

# Image and Video Processing

## Lectures:

- Days: Monday, Thursday & Friday
- slot: H

**Labs:** Assignment based

**Google group:**

<https://groups.google.com/d/forum/ivpw18>

**Textbook:**

“Digital Image Processing”, Rafael C. Gonzalez & Richard E. Woods, 3<sup>rd</sup> ed.

# Course outcomes

At the end of the course the students will be able to:

- Describe the fundamentals of image and video processing and their applications
- Develop familiarity and implement basic image and video processing techniques & algorithms.
- Select and apply appropriate techniques to real problems in image and video analysis.

# Introduction

This lecture will cover:

- What is digital image and it's processing?
- Brief History of DIP
- Sample applications of DIP
- Key stages in DIP
- The human visual system
- Imaging beyond visible spectrum

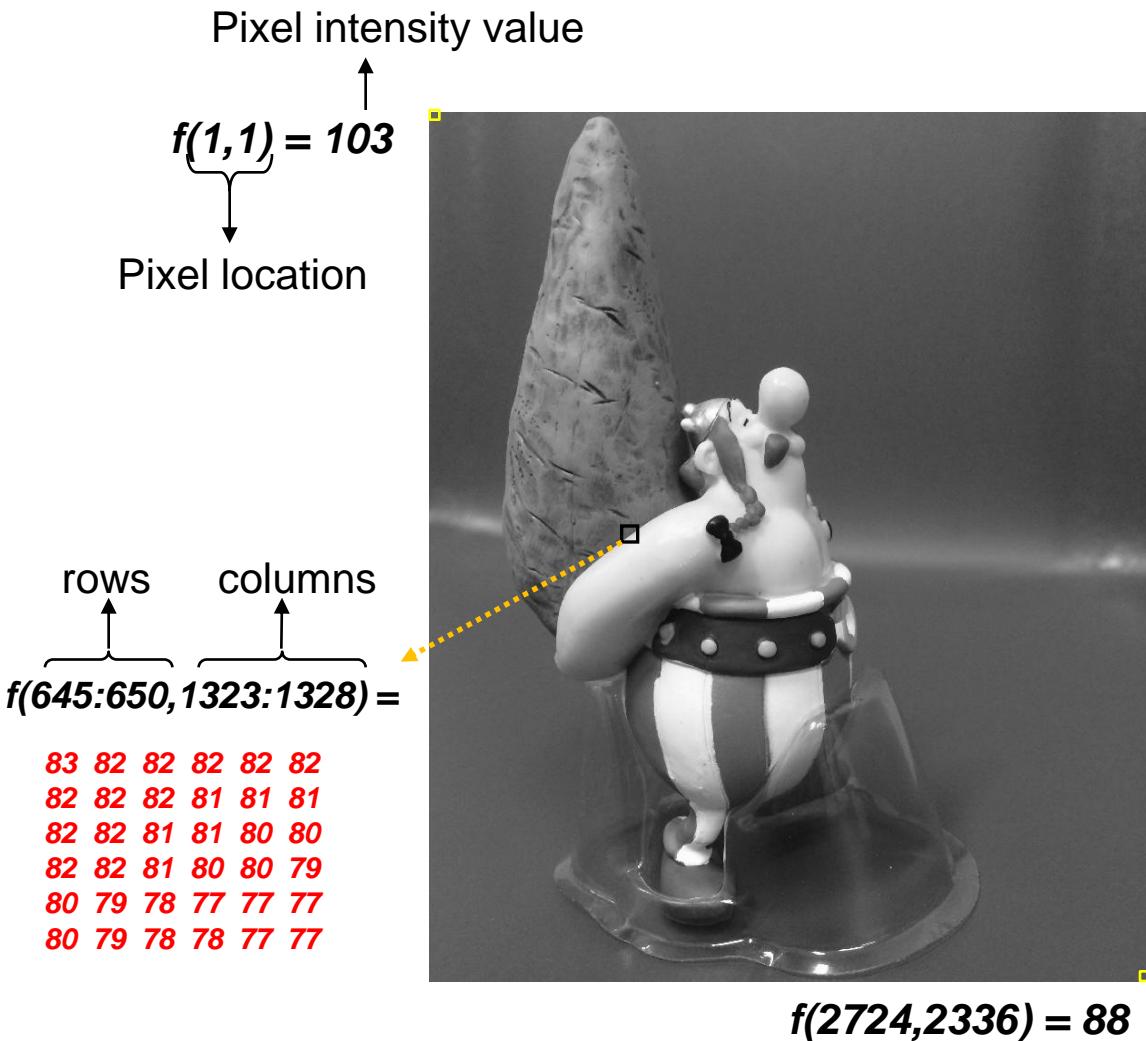
# What is a Digital Image?

An image is a two-dimensional function  $f(x,y)$ , where  $x$  and  $y$  are the **spatial** (plane) coordinates, and the amplitude of  $f$  at any pair of coordinates  $(x,y)$  is called the intensity of the image at that level.

If  $x,y$  and the **amplitude** values of  $f$  are **finite** and **discrete quantities**, we call the image a **digital image**.

A digital image is composed of a finite number of elements called **pixels**, each of which has a particular location and value.

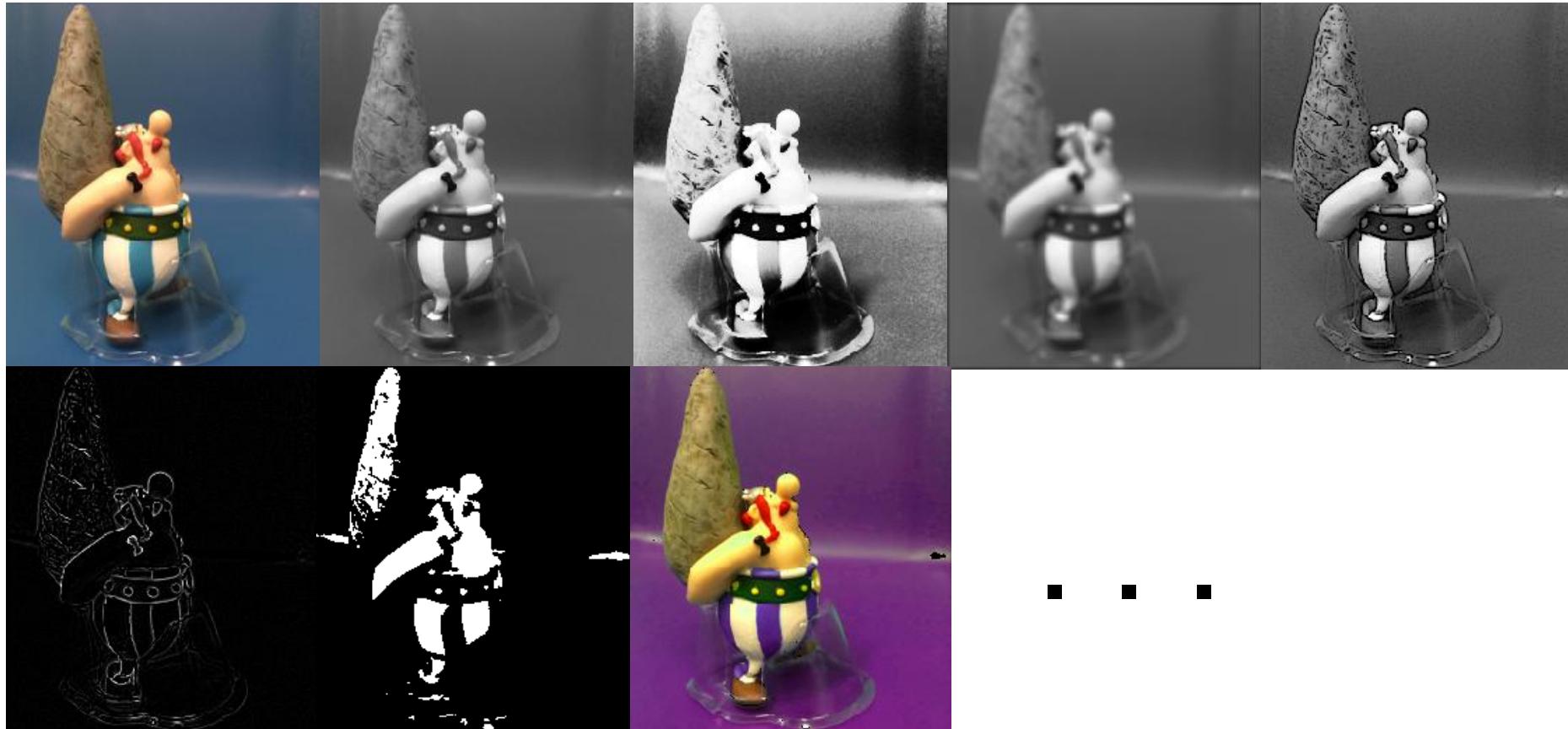
# What is a Digital Image? (contd.)



Consider the following image (2724x2336 pixels) to be 2D function or a **matrix** with **rows** and **columns**

In **8-bit** representation Pixel intensity values range between **0 (Black)** and **255 (White)**

# Digital Image Processing



# Digital Image Processing

Motivated by following principal objectives:

- Improvement of pictorial information for human perception and interpretation
- Processing of image data for storage, transmission and representation for autonomous machine application

# History of DIP

**Early 1920s:** One of the first applications of digital imaging was in the newspaper industry

- The Bartlane cable picture transmission service
- Photographs were transmitted by cable between London and New York
- Pictures were coded for cable transfer and reconstructed at the receiving end on a telegraph printer using specially designed printing blocks.

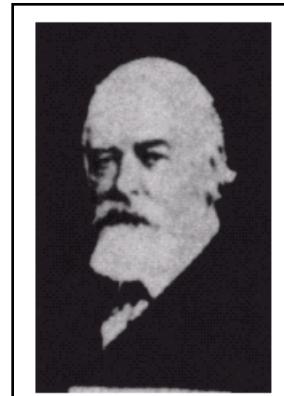


Early digital image

# History of DIP (cont...)

**Mid to late 1920s:** Improvements to the Bartlane system resulted in higher quality images

- New reproduction processes based on photographic techniques
- Increased number of tones in reproduced images



Improved  
digital image

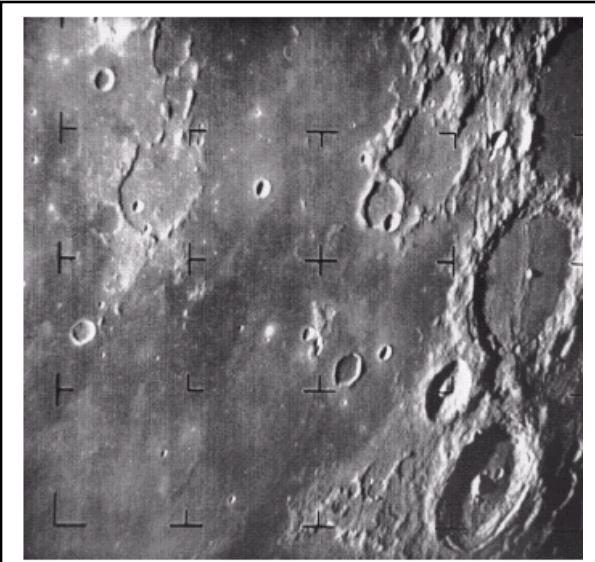


Early 15 tone digital  
image

# History of DIP (cont...)

**1960s:** Improvements in computing technology and the onset of the space race led to a surge of work in digital image processing

- **1964:** Computers used to improve the quality of images of the moon taken by the *Ranger 7* probe
- Such techniques were used in other space missions

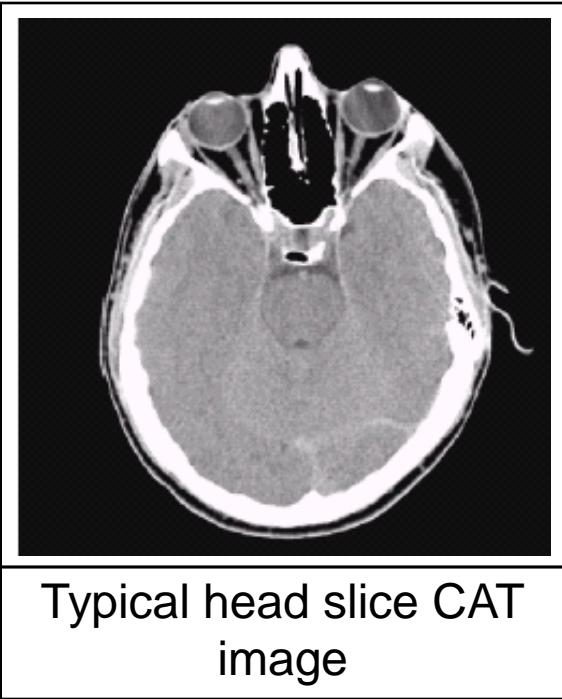


A picture of the moon taken by the Ranger 7 probe minutes before landing

# History of DIP (cont...)

**1970s:** Digital image processing begins to be used in medical applications

- **1979:** Sir Godfrey N. Hounsfield & Prof. Allan M. Cormack share the Nobel Prize in medicine for the invention of tomography, the technology behind Computerised Axial Tomography (CAT) scans



Typical head slice CAT image

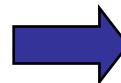
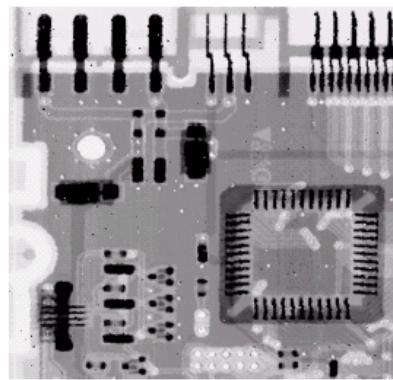
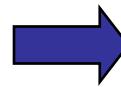
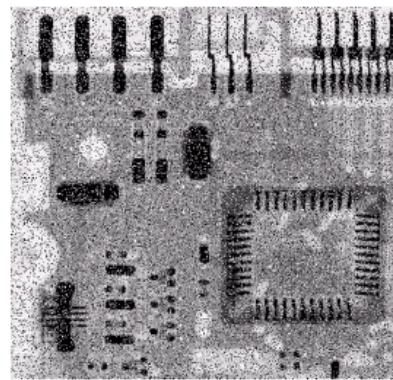
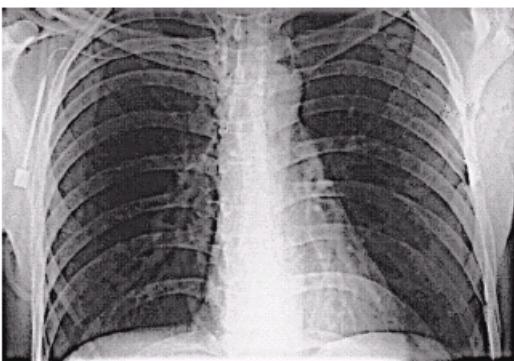
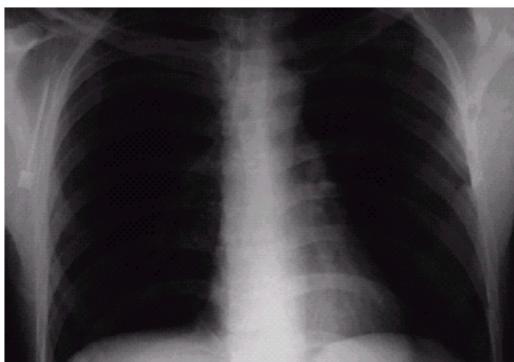
# History of DIP (cont...)

**1980s - Today:** The use of digital image processing techniques has exploded and they are now used for all kinds of tasks in all kinds of areas

- Image enhancement/restoration
- Medical visualisation
- Remote sensing & GIS
- Industrial inspection
- Law enforcement
- Human computer interfaces
- Special effects in movies

# Applications: Image Enhancement

One of the most common uses of DIP techniques: improve quality, remove noise etc

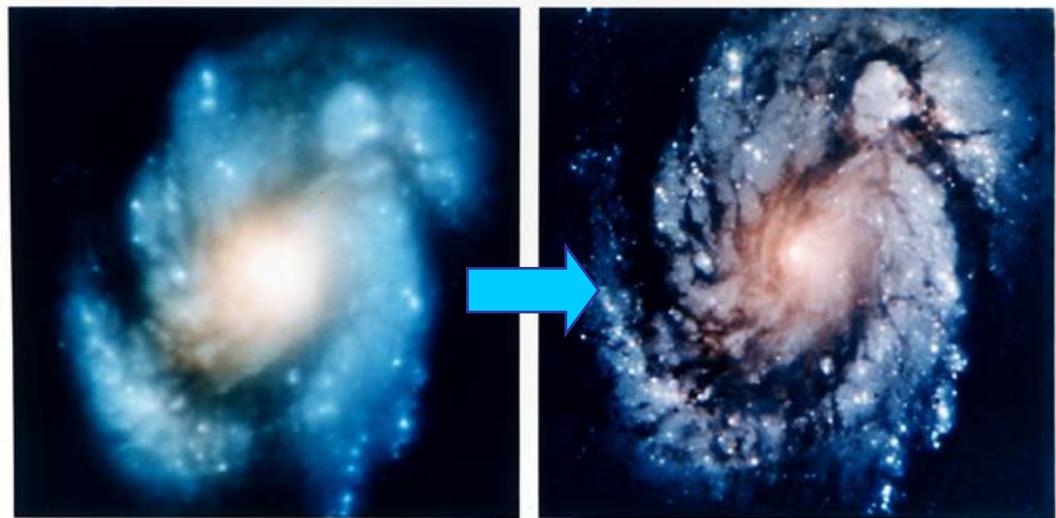


# Applications : The Hubble Telescope

Launched in 1990 the Hubble telescope can take images of very distant objects

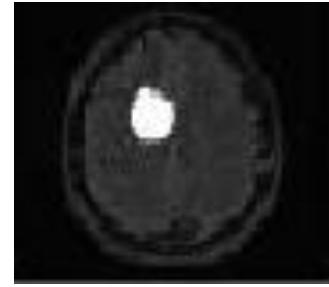
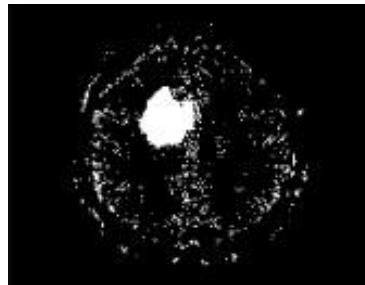
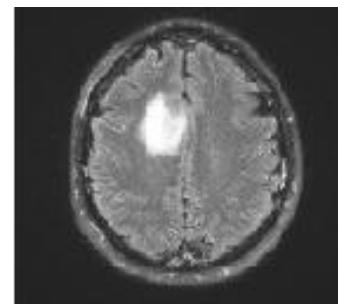
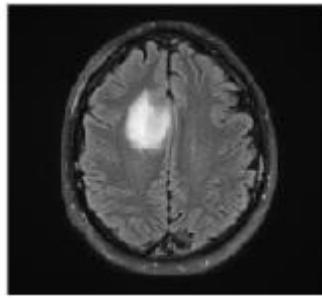
However, an incorrect mirror made many of Hubble's images useless

Image processing techniques were used to fix this



# Applications : Medicine

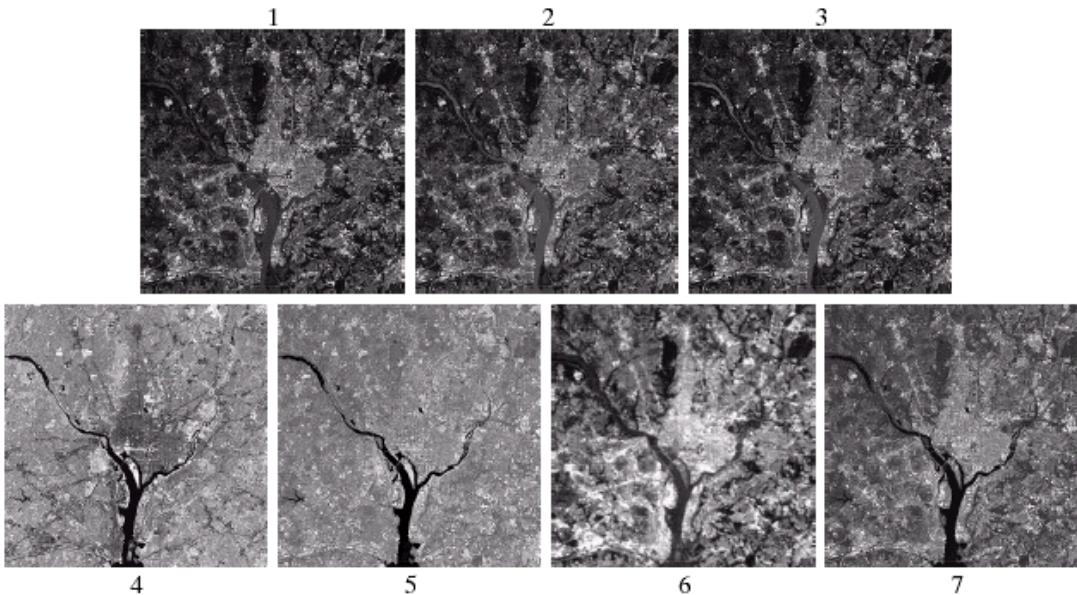
## Detection of brain tumor



# Applications : GIS

## Geographic Information Systems

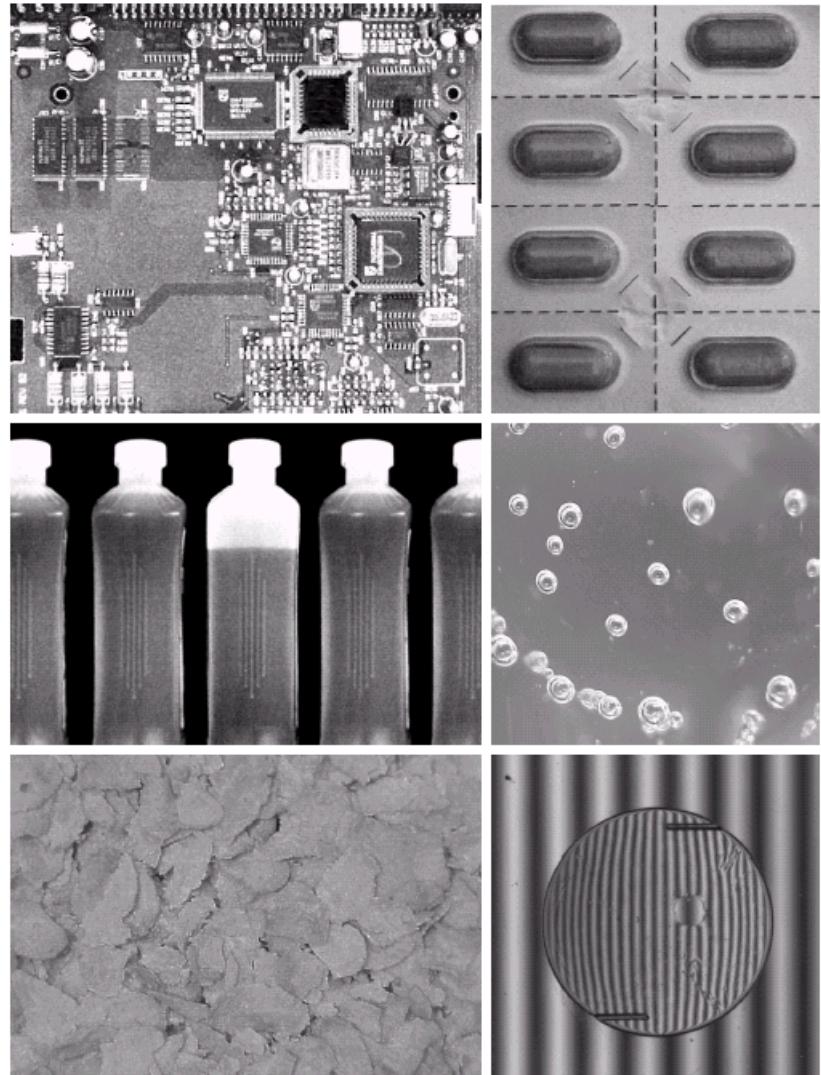
- Digital image processing techniques are used extensively to manipulate satellite imagery
- Terrain classification
- Astronomy



# Applications : Industrial Inspection

Human operators are expensive, slow and at times unreliable.

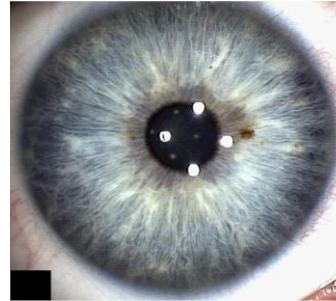
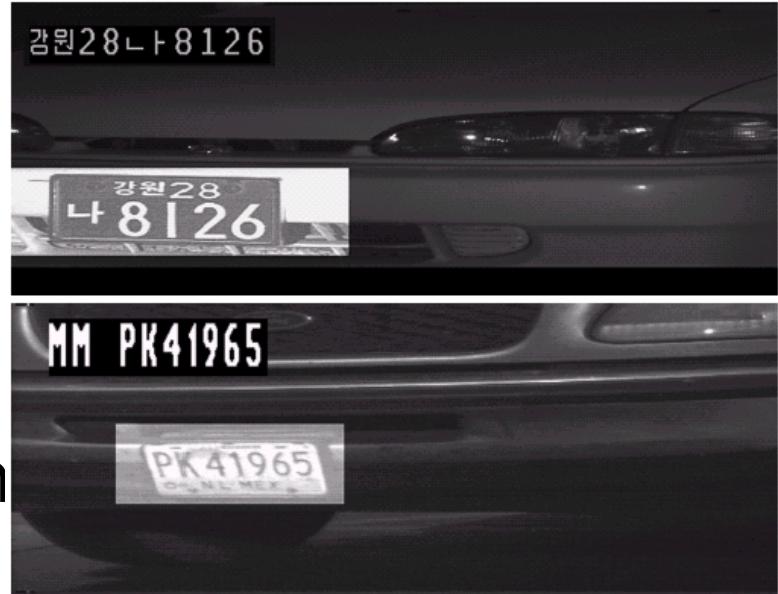
Industrial vision systems are used in all kinds of industries



# Applications : Law Enforcement

Image processing techniques are used extensively by law enforcers

- Number plate recognition for speed cameras/ automated toll systems
- Biometrics and forensics.

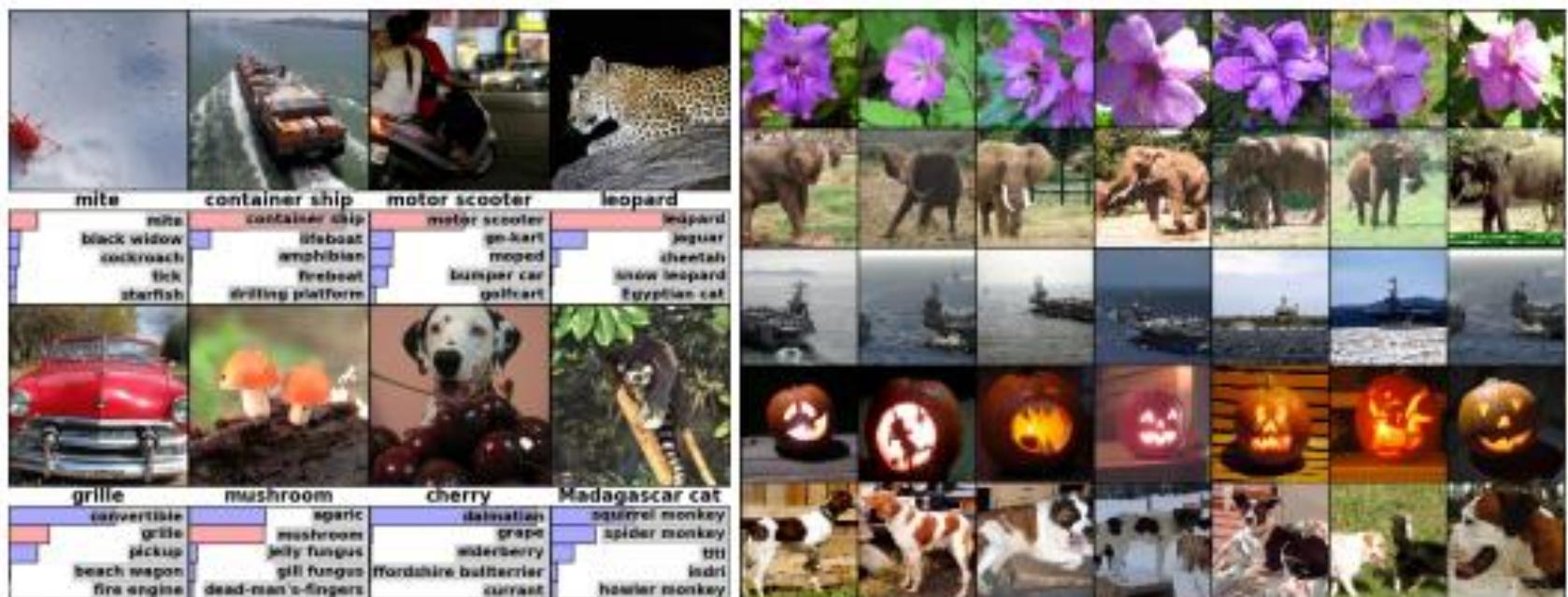


# Applications : Special Effects in movies

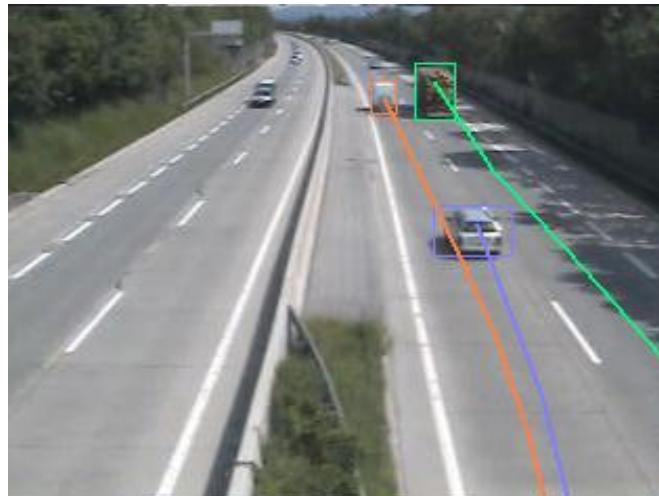
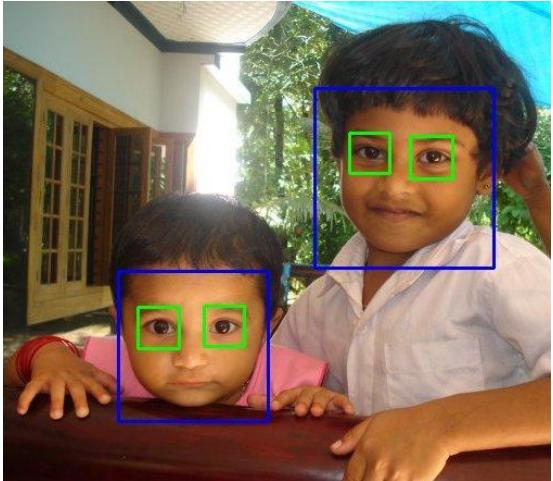
Artistic effects are used to make images more visually appealing, to add special effects and to make composite images



# Image classification and tagging



# Object Detection, Recognition and Tracking



# Applications : Law Enforcement

Video surveillance using  
CCTV camera

- Pedestrian and vehicle detection and tracking



Traffic and safety in highways,  
metro stations, Airport,  
railways etc.

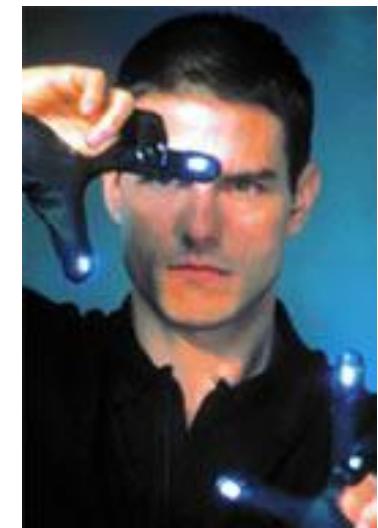
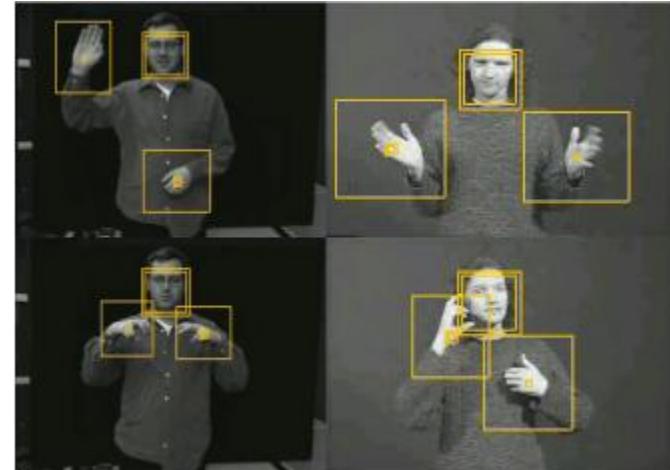


Homeland security for human lives  
and property in banks, shopping  
malls, smart homes, buildings etc.

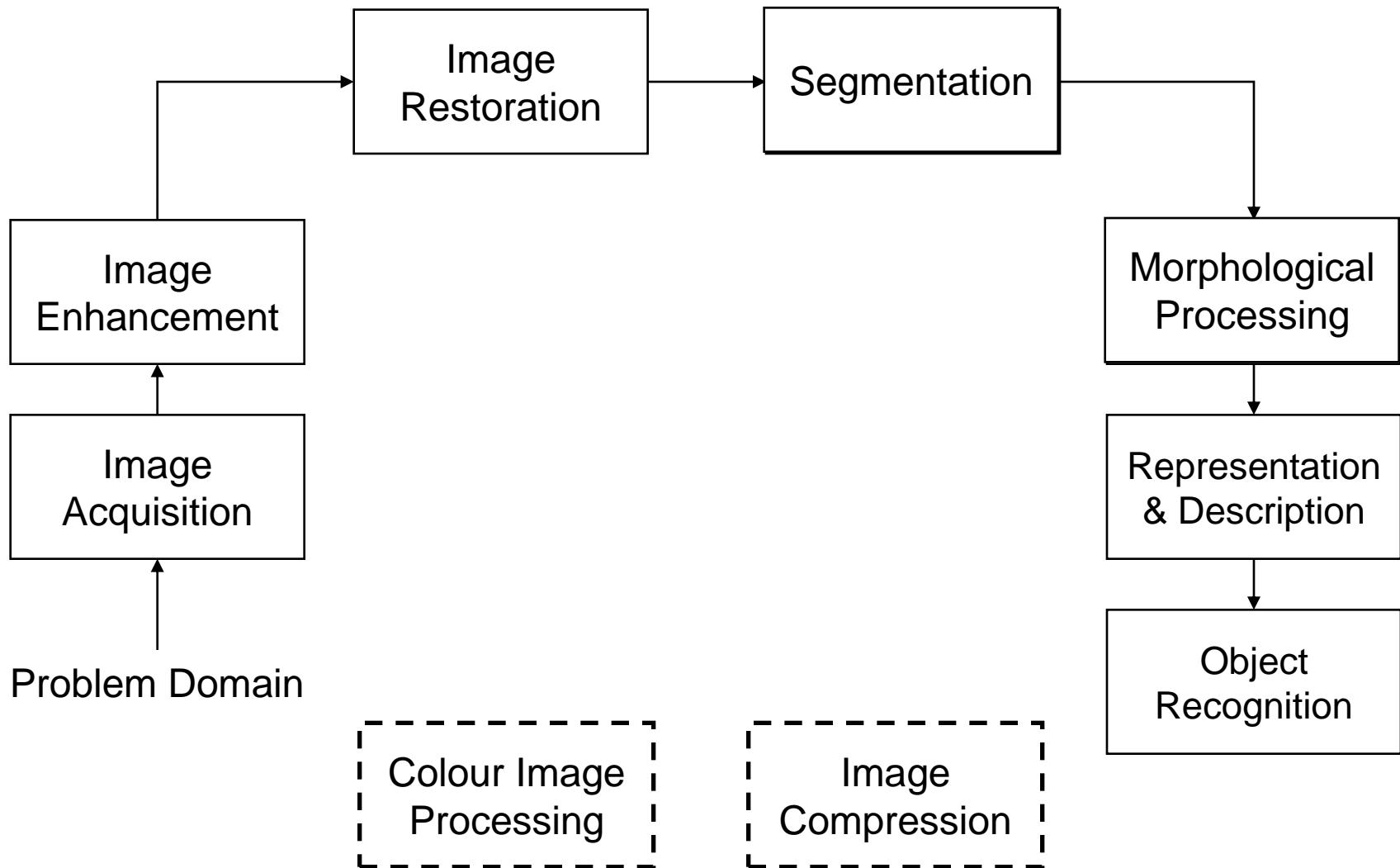
# Applications : HCI

Try to make human computer interfaces more natural

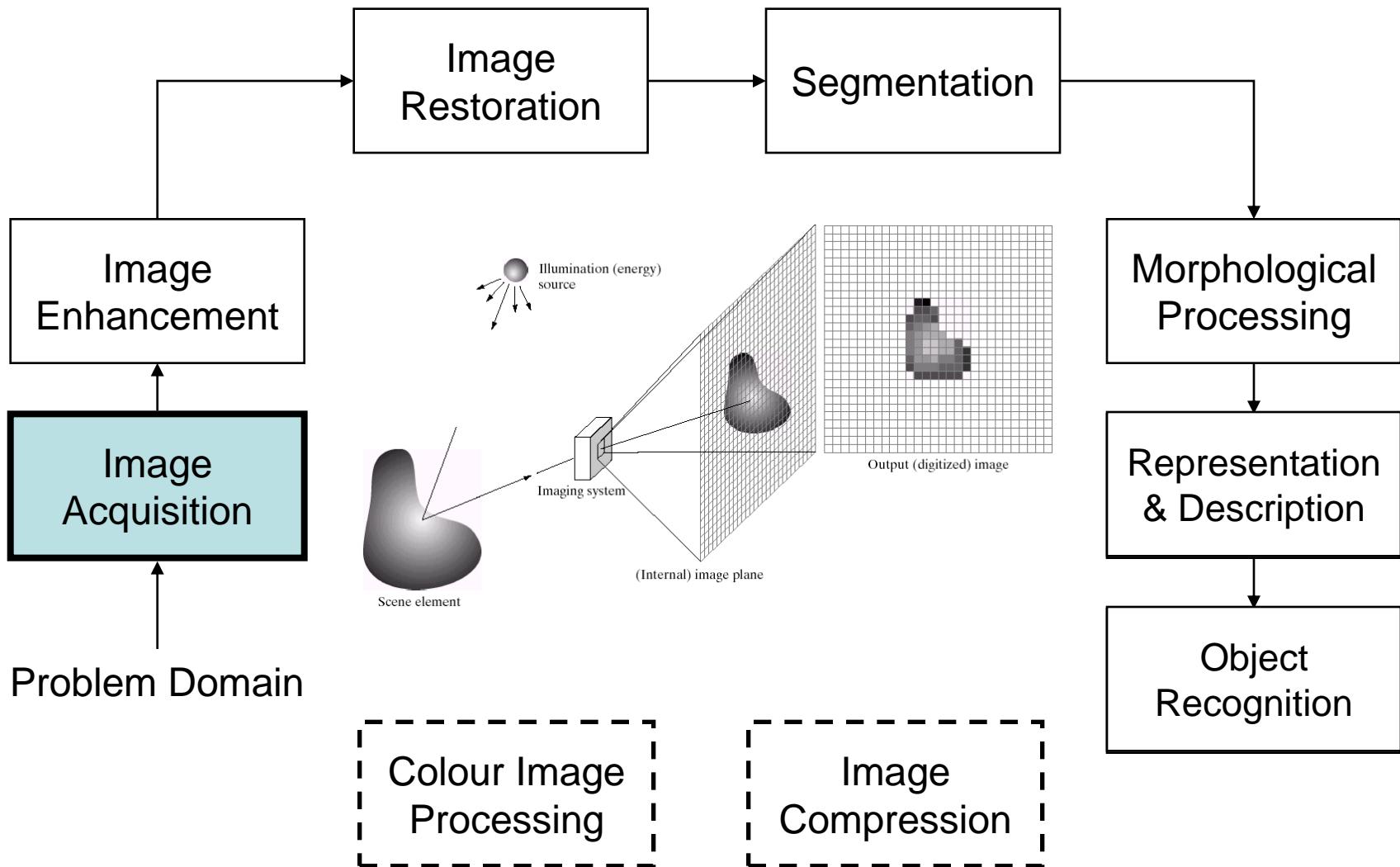
- Face recognition
- Gesture recognition



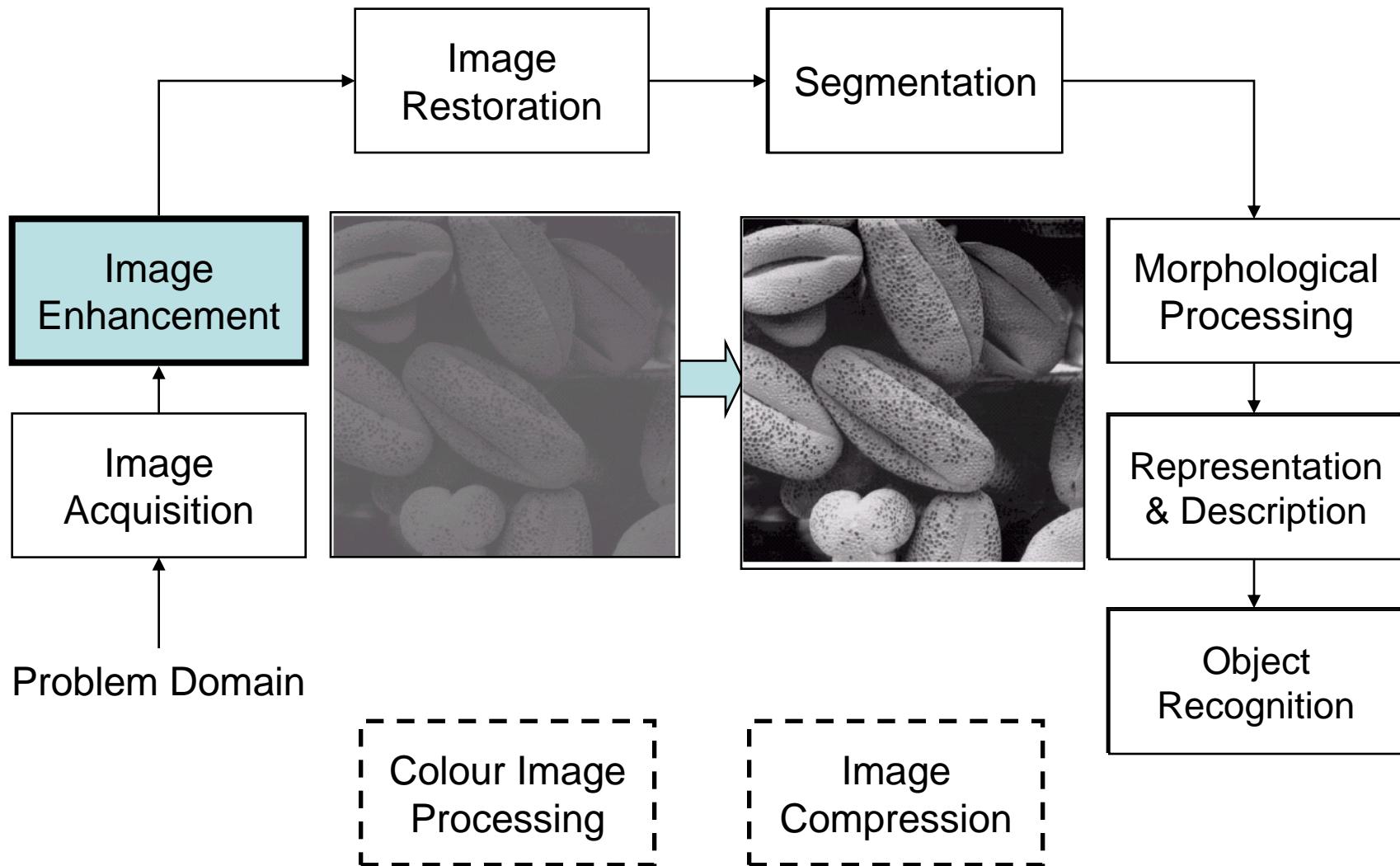
# Key Stages in Digital Image Processing



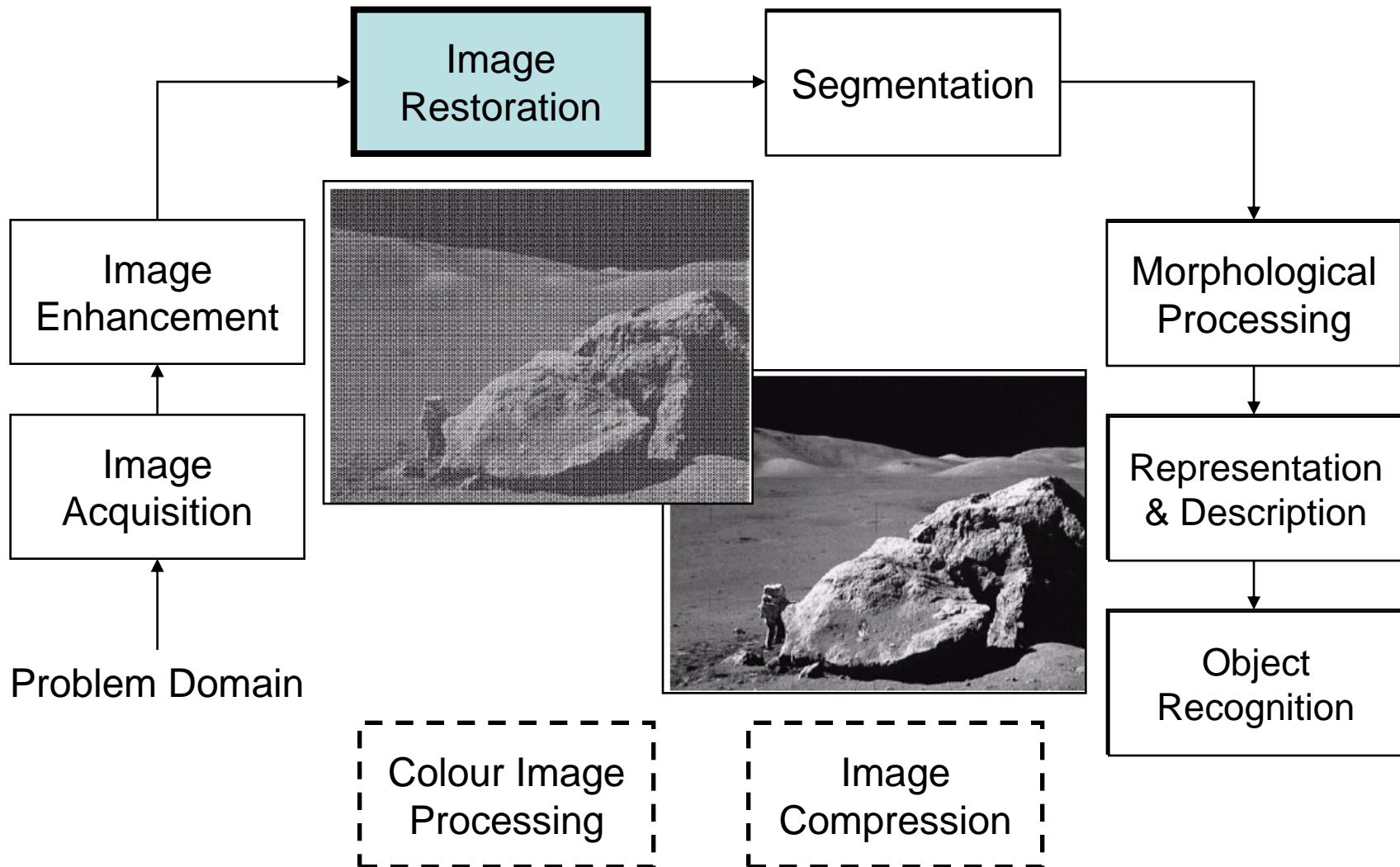
# Key Stages in Digital Image Processing: Image Acquisition



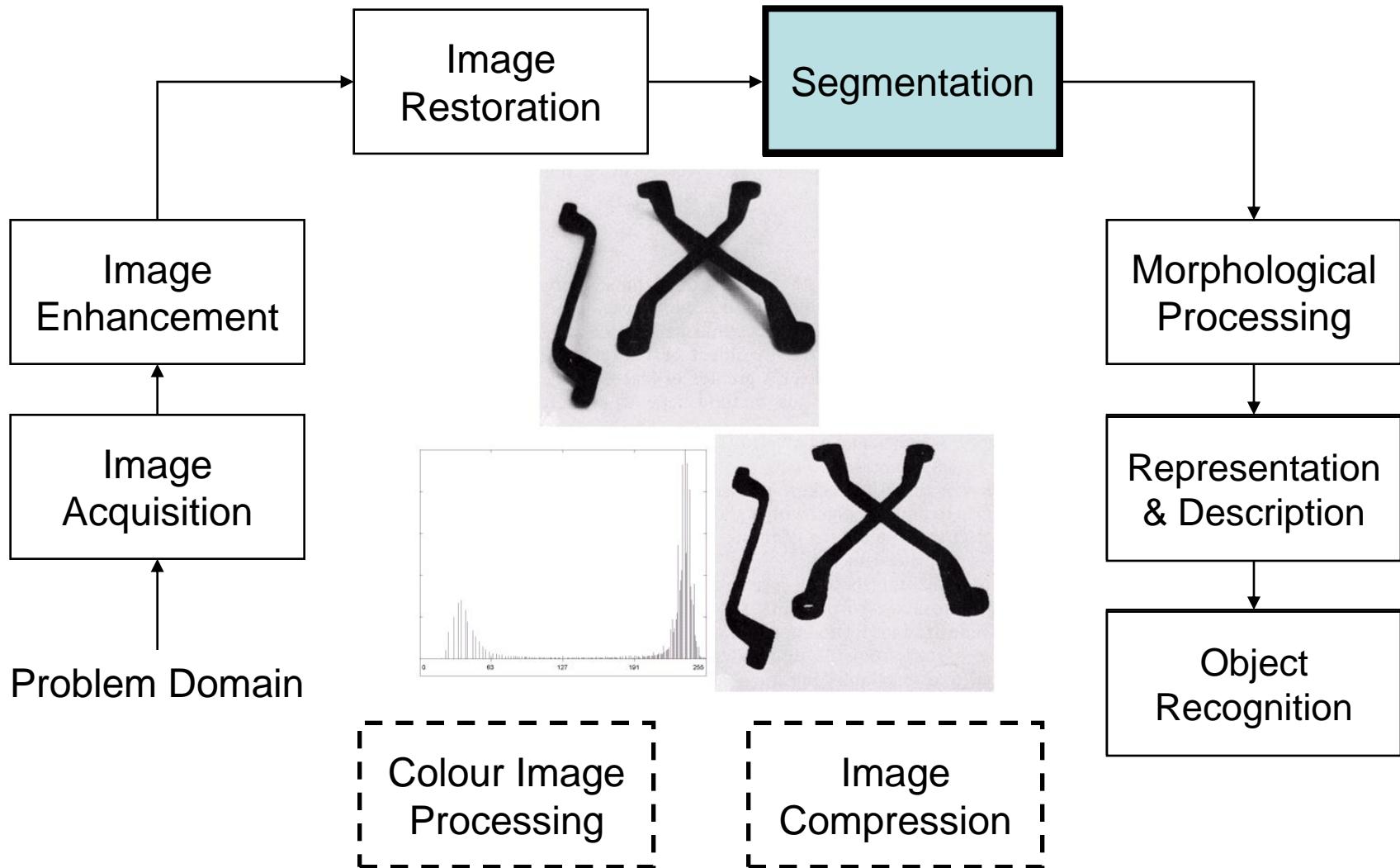
# Key Stages in Digital Image Processing: Image Enhancement



# Key Stages in Digital Image Processing: Image Restoration

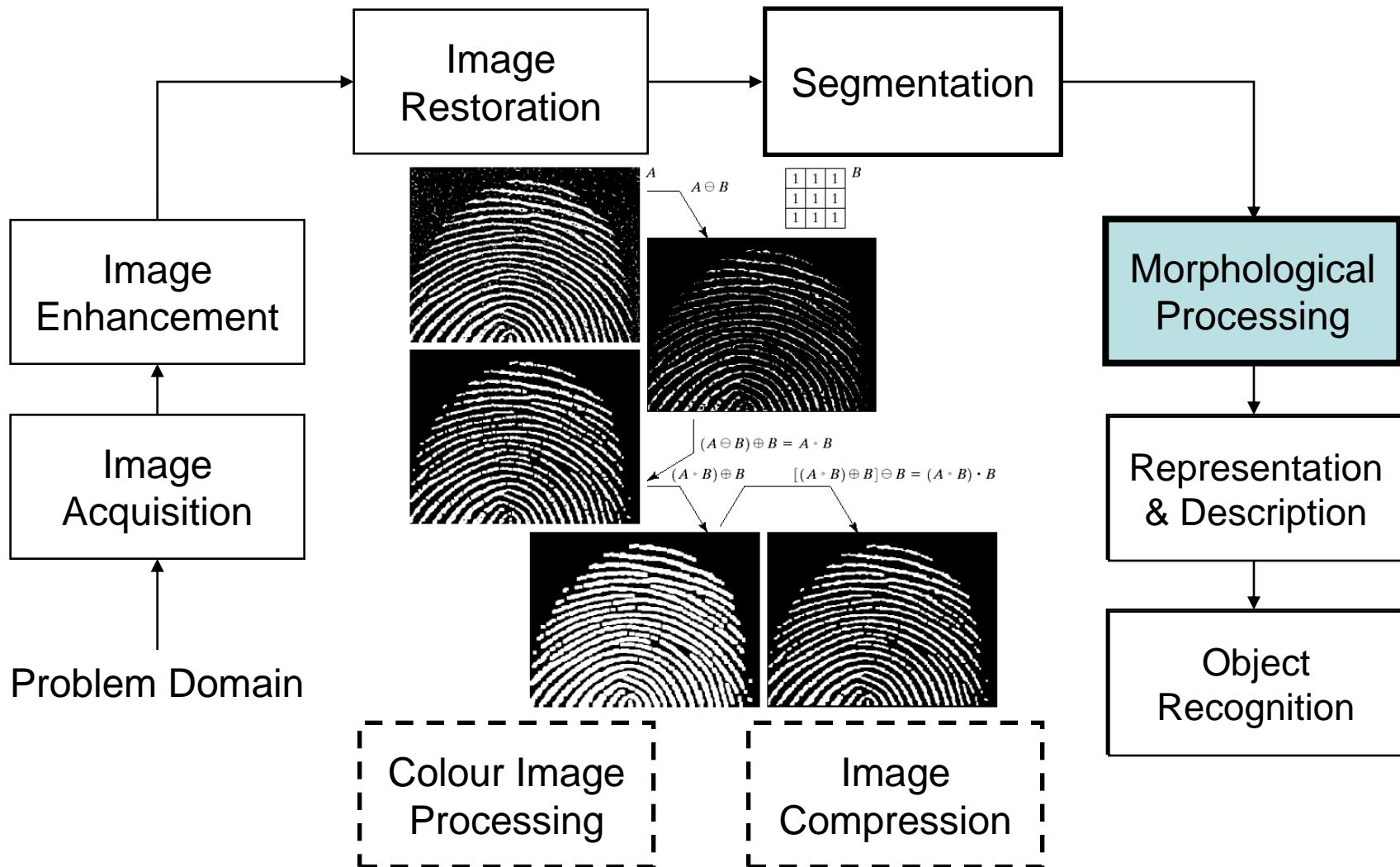


# Key Stages in Digital Image Processing: Segmentation

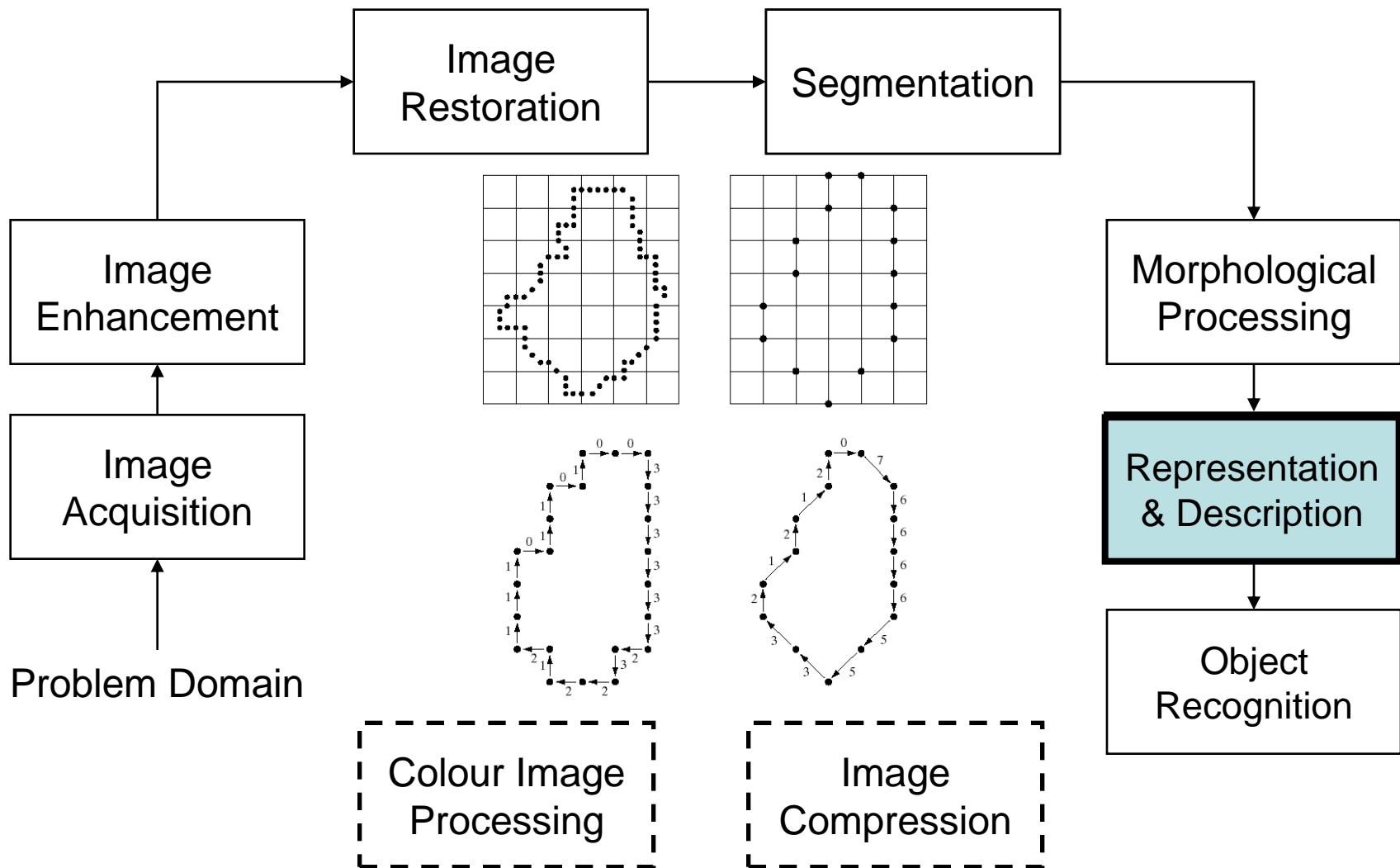


# Key Stages in Digital Image Processing:

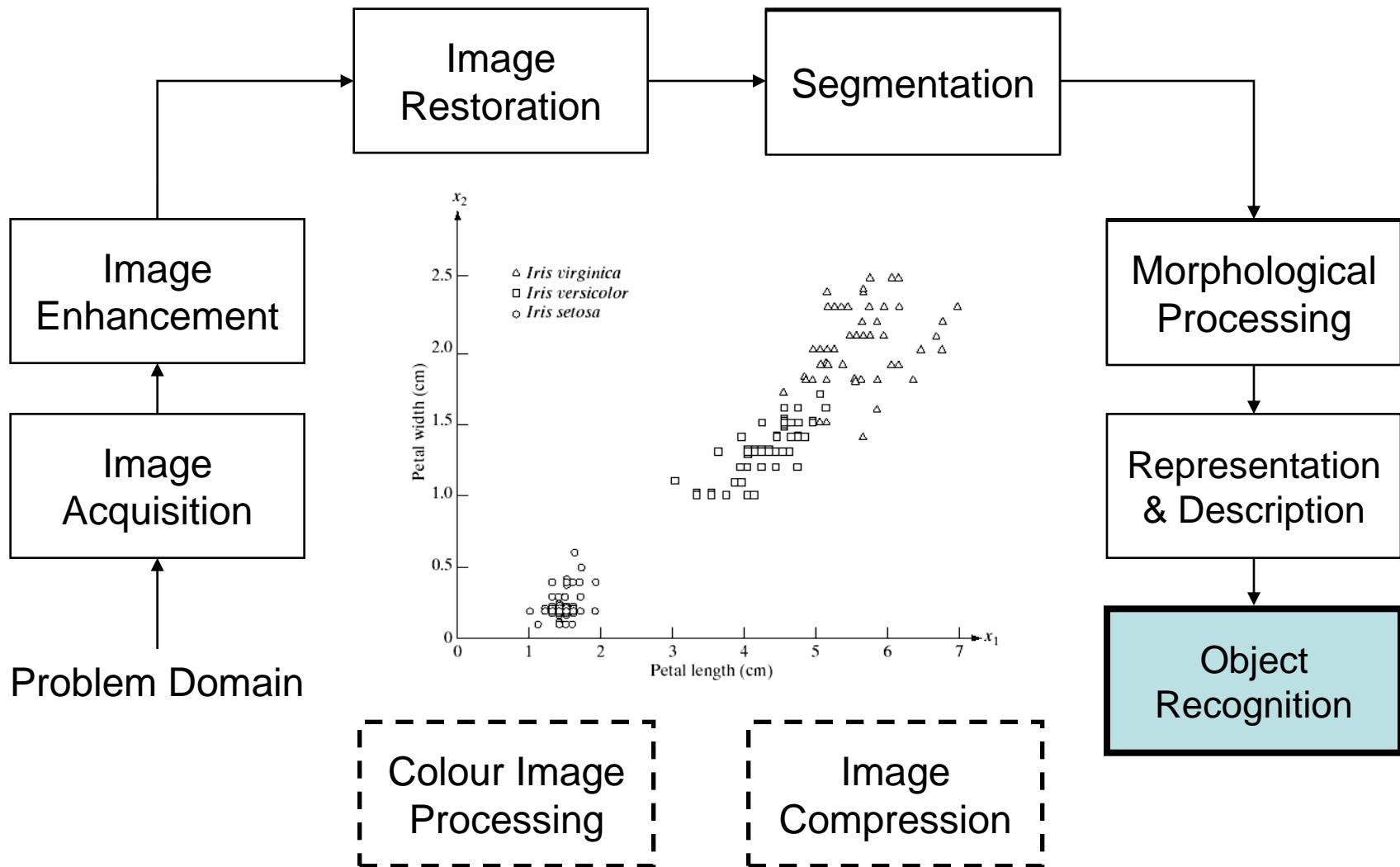
## Morphological Processing



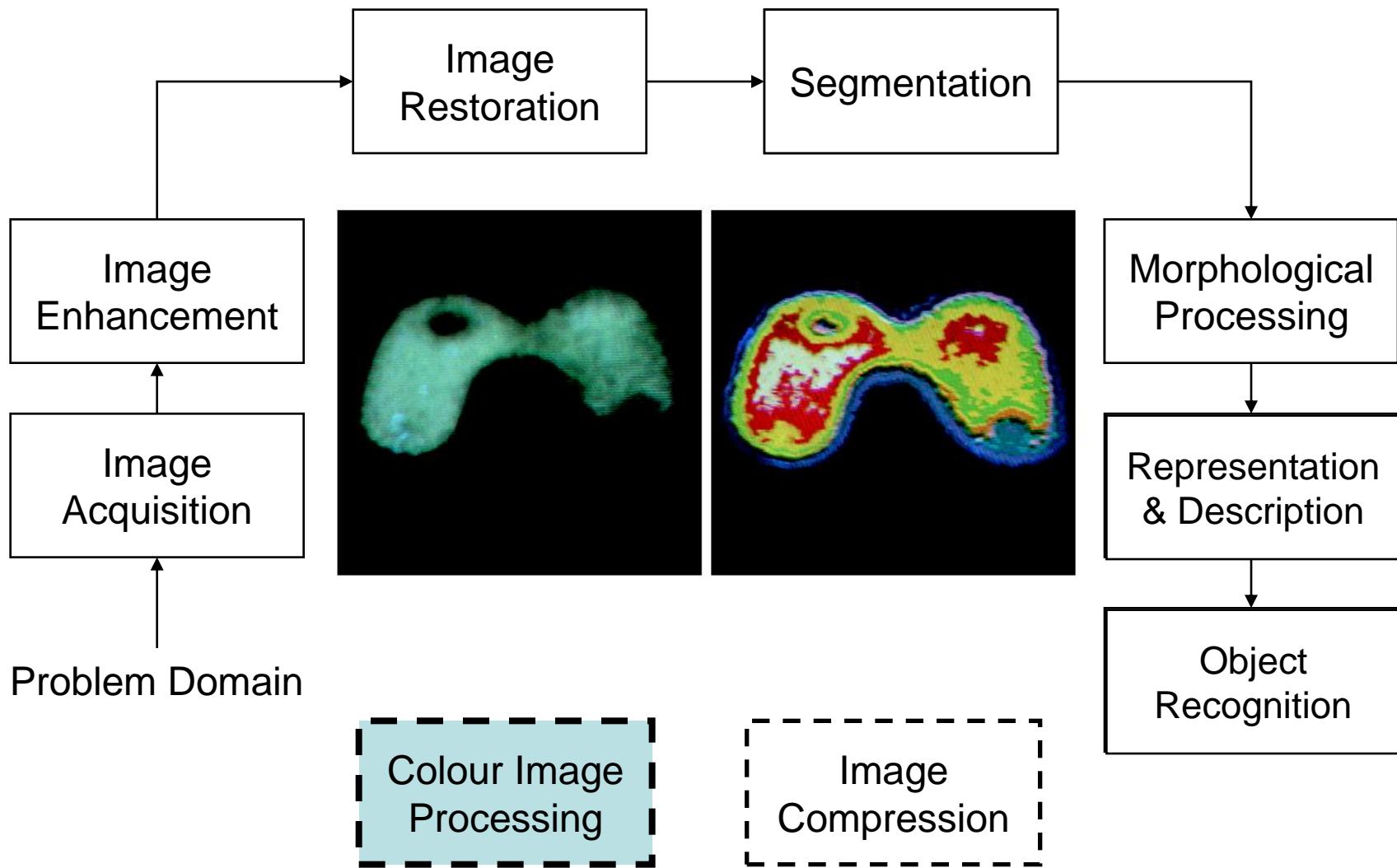
# Key Stages in Digital Image Processing: Representation & Description



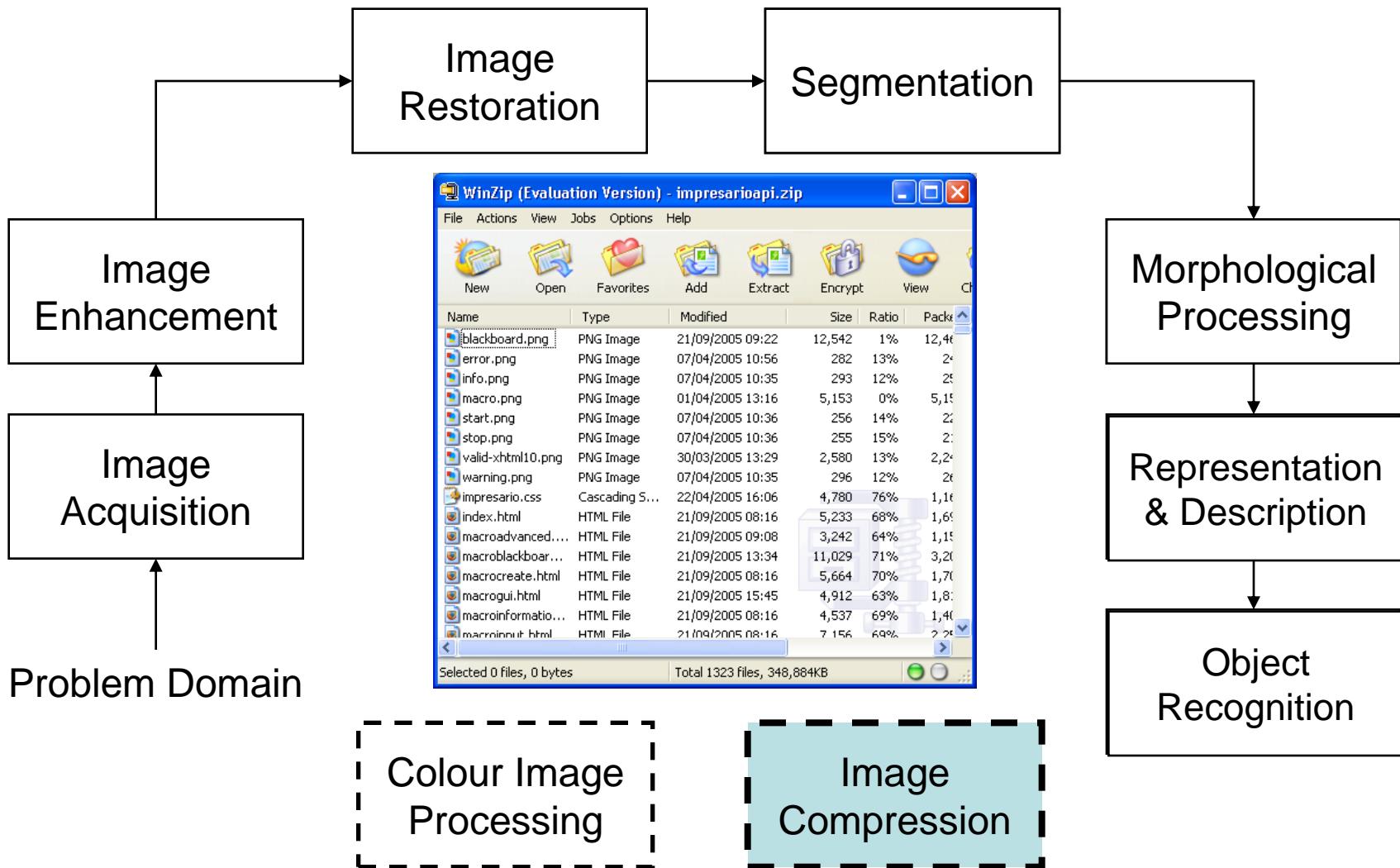
# Key Stages in Digital Image Processing: Object Recognition



# Key Stages in Digital Image Processing: Colour Image Processing



# Key Stages in Digital Image Processing: Image Compression



# Scope of this course

The continuum from image processing to computer vision can be broken up into low-, mid- and high-level processes

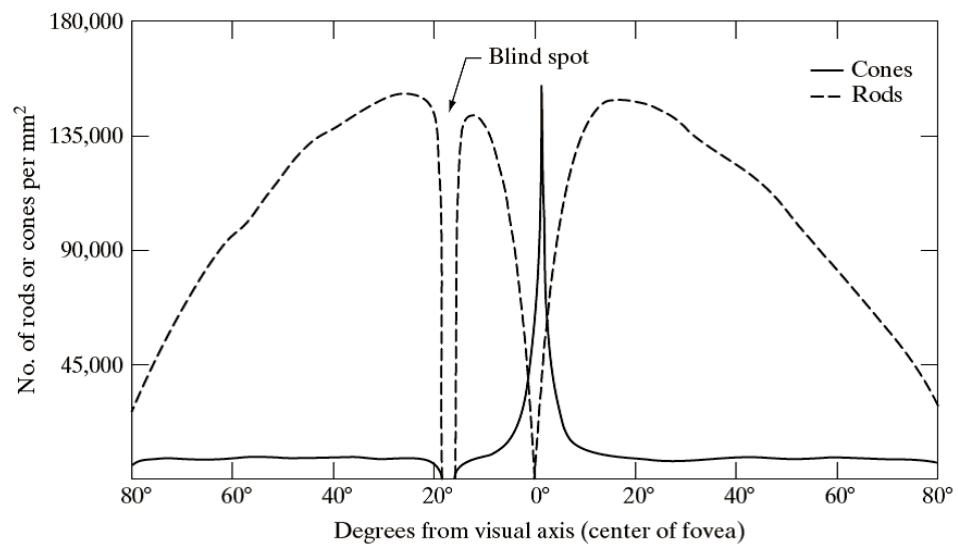
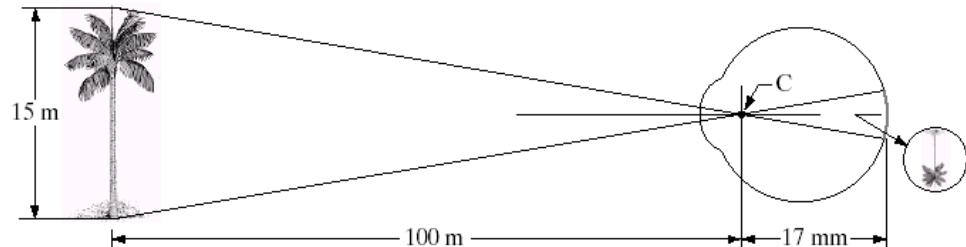
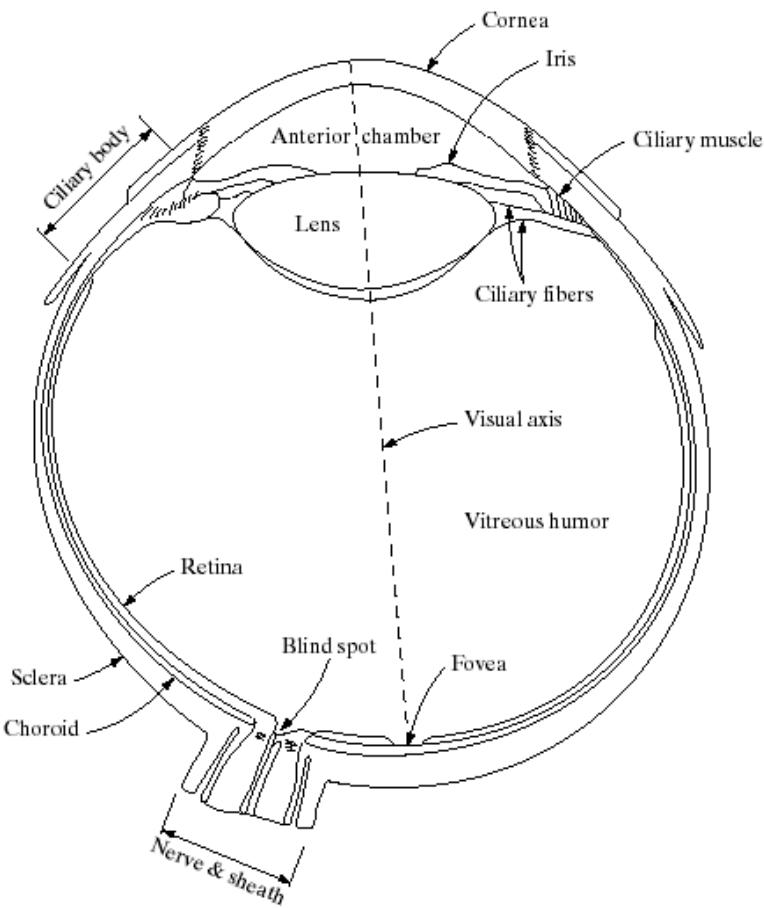
Low Level Process	Mid Level Process	High Level Process
<b>Input:</b> Image <b>Output:</b> Image	<b>Input:</b> Image <b>Output:</b> Attributes	<b>Input:</b> Attributes <b>Output:</b> Understanding
<b>Examples:</b> Noise removal, image sharpening	<b>Examples:</b> Object recognition, segmentation	<b>Examples:</b> Scene understanding, autonomous navigation

In this course we will stop here

# The Human Visual System

- Understanding of the human visual system can help in appreciating how computer vision systems might be designed.
- Human intuition and analysis based on visual judgement can play a significant role in choice of one technique over other.
- Given the complexity of vision model we will just look at a rudimentary aspects of the human visual system

# Structure Of The Human Eye



# Blind-Spot Experiment

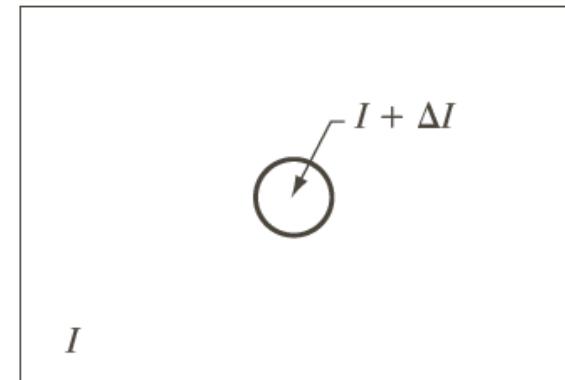
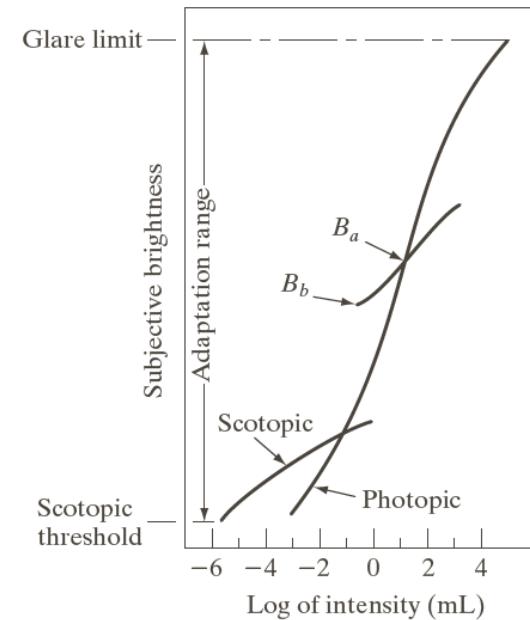
- Draw an image similar to that below on a piece of paper (the dot and cross are about 6 inches apart)



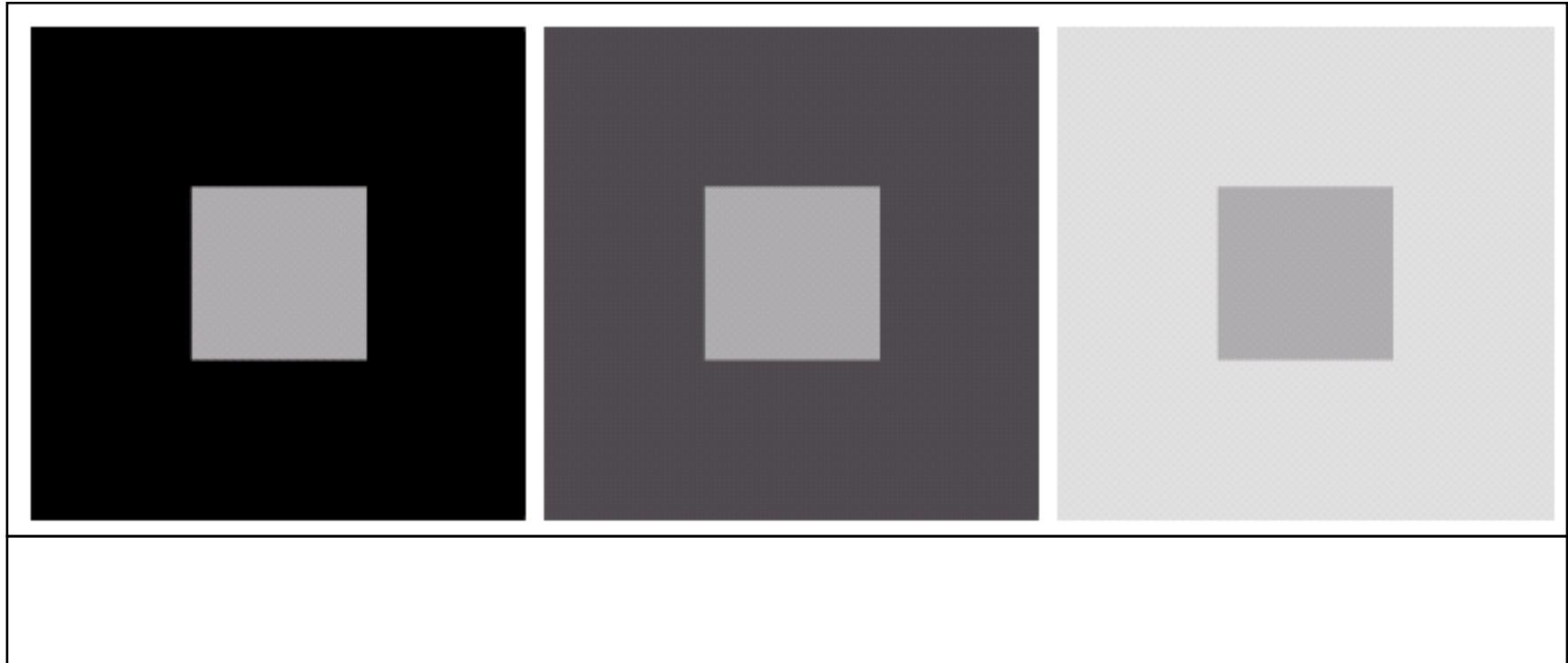
- Close your right eye and focus on the cross with your left eye
- Hold the image about 20 inches away from your face and move it slowly towards you
- The dot should disappear!

# Brightness Adaptation & Discrimination

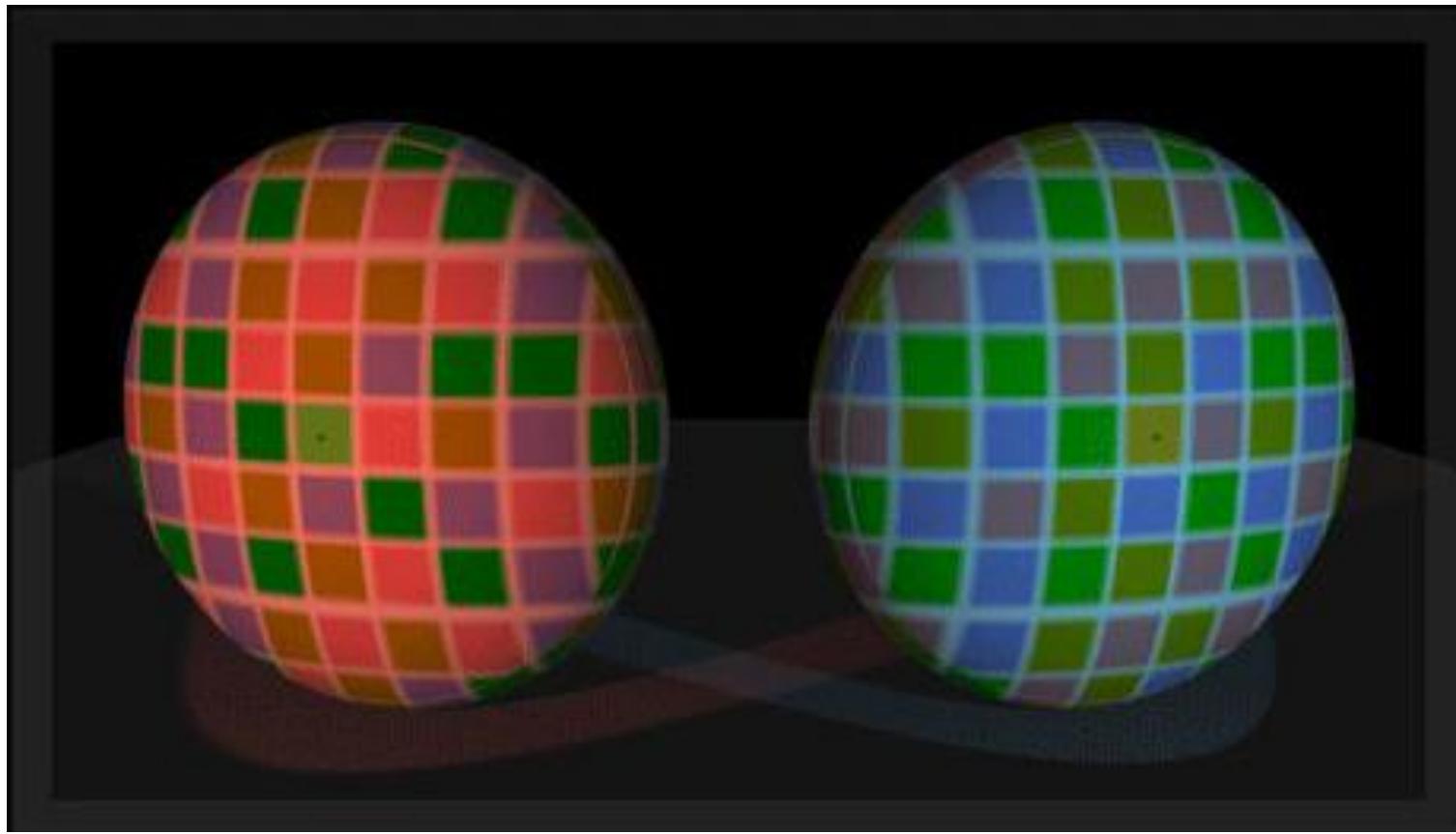
- The human visual system can perceive approximately  $10^{10}$  different light intensity levels
- However, at any one time we can only discriminate between a much smaller number – *brightness adaptation*
- Similarly, the *perceived intensity* of a region is related to the light intensities of the regions surrounding it



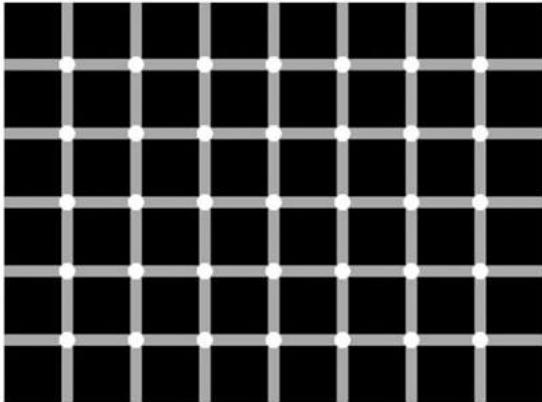
# Brightness Adaptation & Discrimination (cont...)



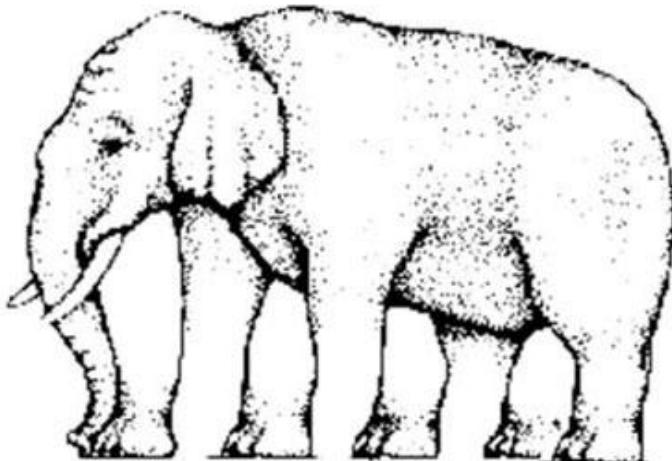
# Color Adaptation



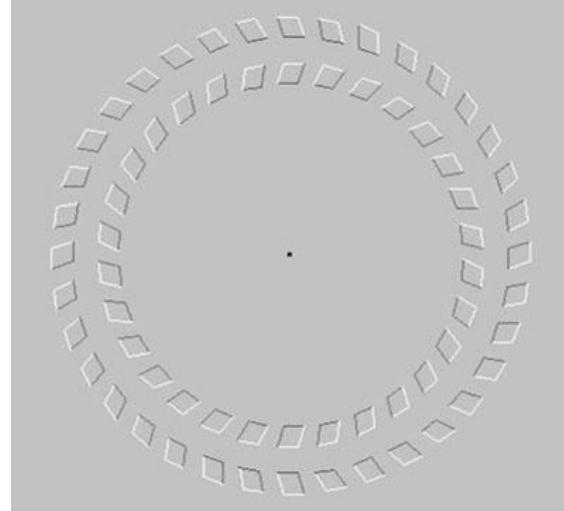
# Optical Illusions



Are the dots in between the squares white, black or grey?



How many legs does this elephant have?

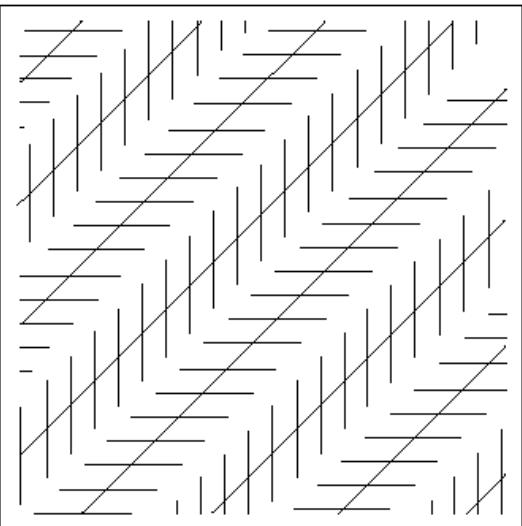
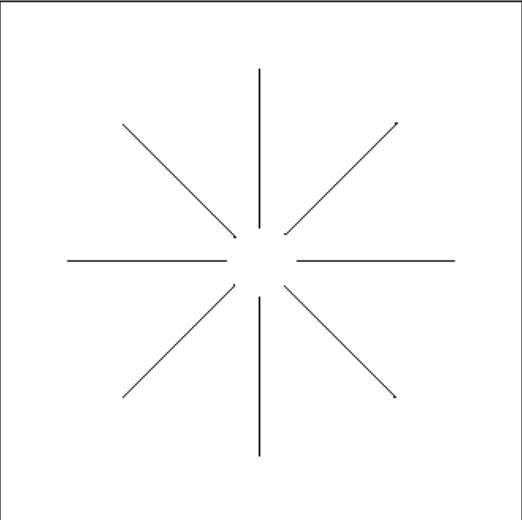
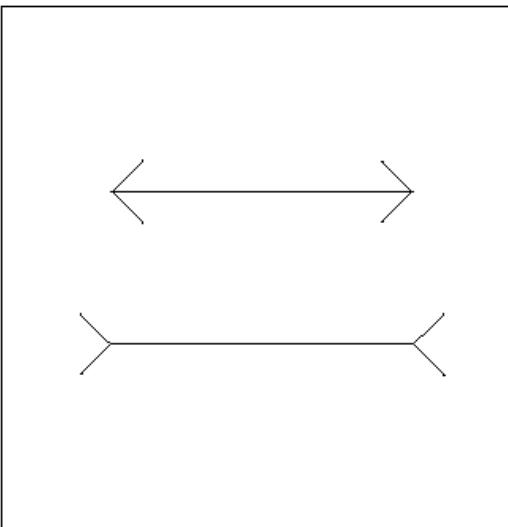
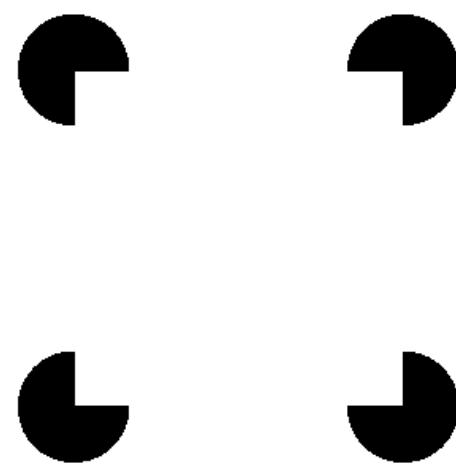
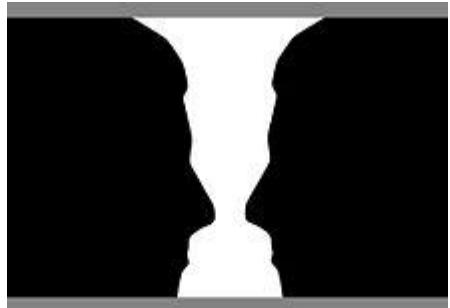
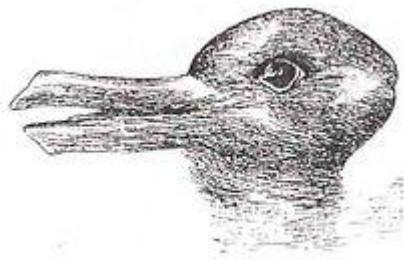
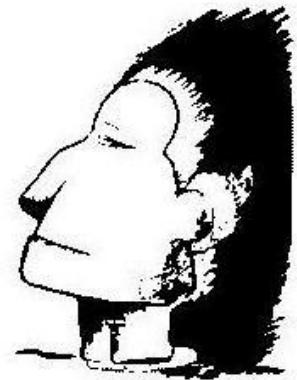


Is this the face of a lady or a word?

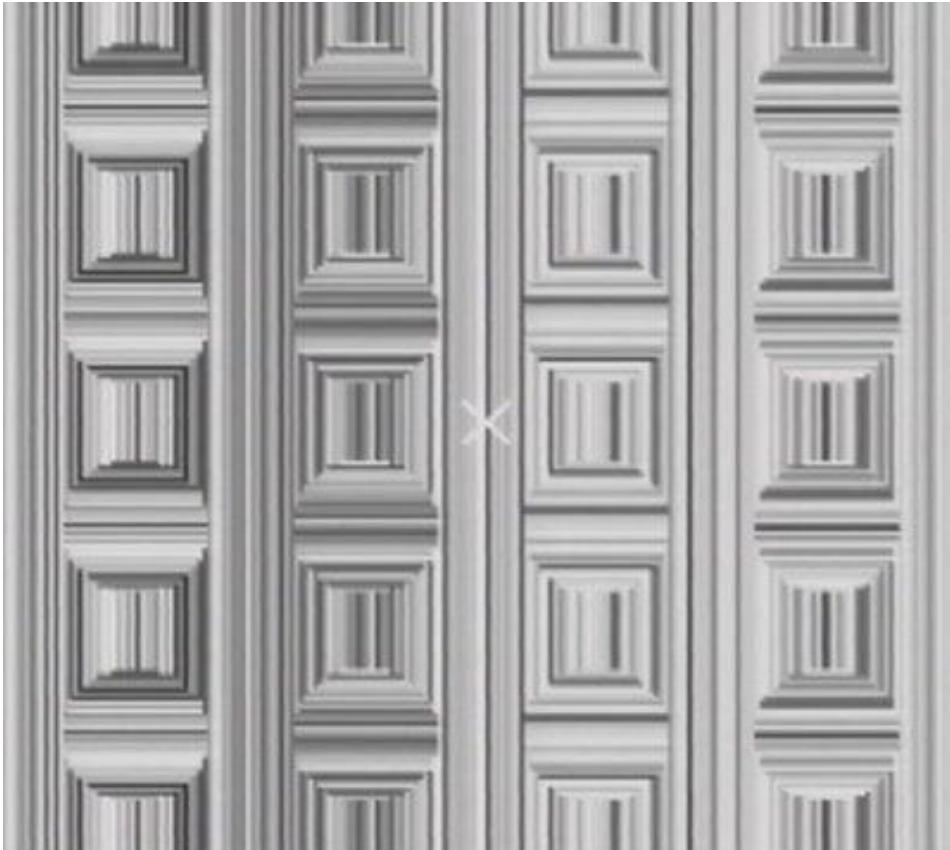


Focus on the dot in the middle and then move your head backwards and forwards.

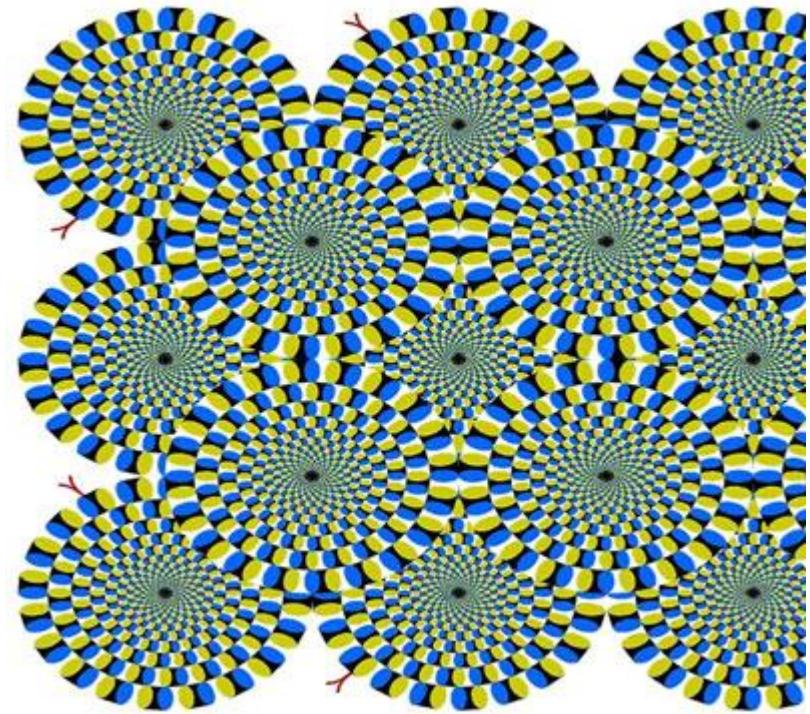
# Optical Illusions



# Optical Illusions



Stare at the cross in the middle of the image and think circles



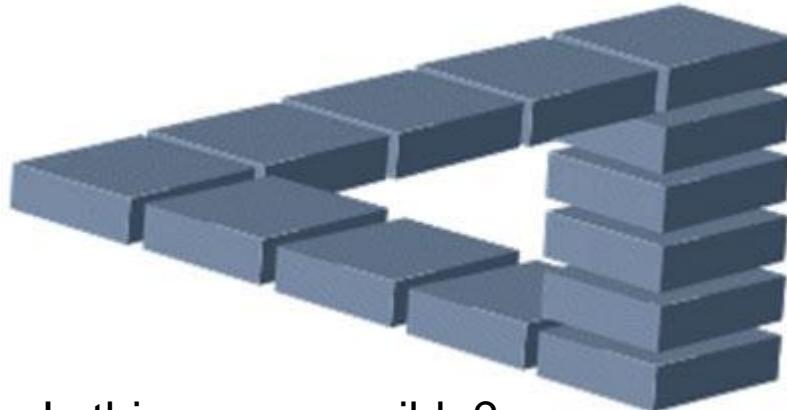
Is this picture still or moving?

# Optical Illusions



Is the ladder going up or down?

Focus on the 4 dots in the middle of the picture for 30 seconds. Then look at a blank wall - who do you see? Maybe blink your eyes a few times to find out.

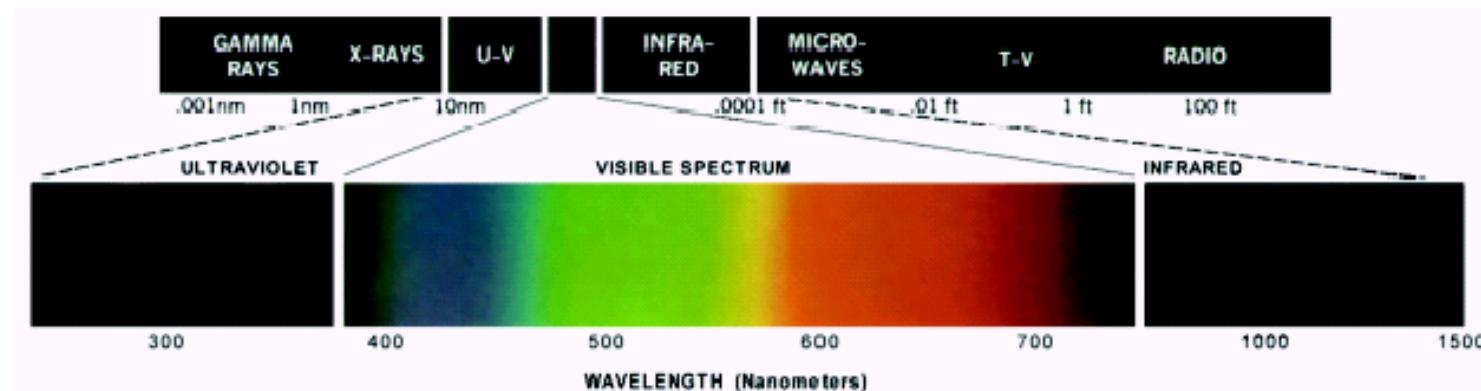


Is this even possible?



# Sources of Digital Images

- The principal source for the images is the **electromagnetic (EM) energy spectrum**.
- The electromagnetic spectrum is split up according to the wavelengths of different forms of energy

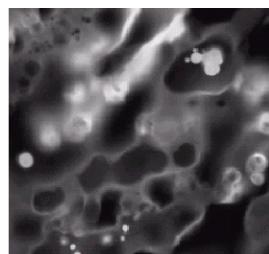


# Imaging in EM spectrum

Positron Emission Tomography



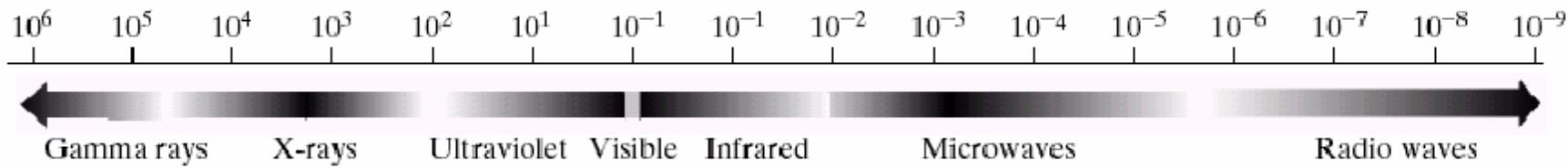
Fluorescence image of corn



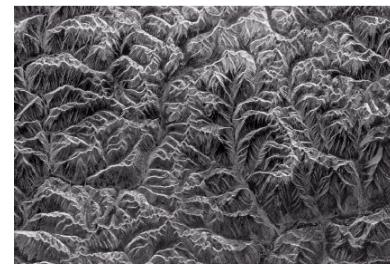
infrared Imaging



Energy of one photon (electron volts)



Chest X-ray



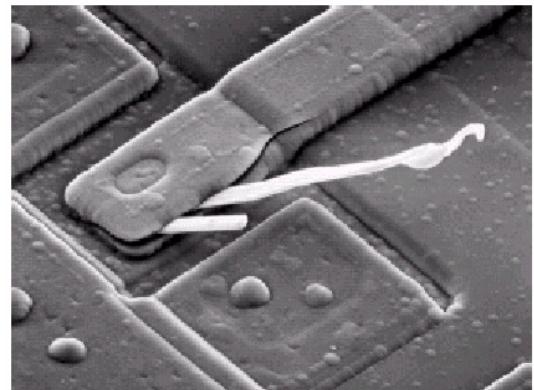
Radar Image

# Other Non-Electro-Magnetic Imaging Modalities

Ultrasound imaging



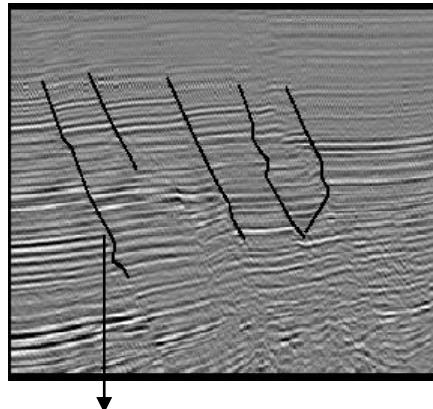
Electron microscopy



visible



seismic

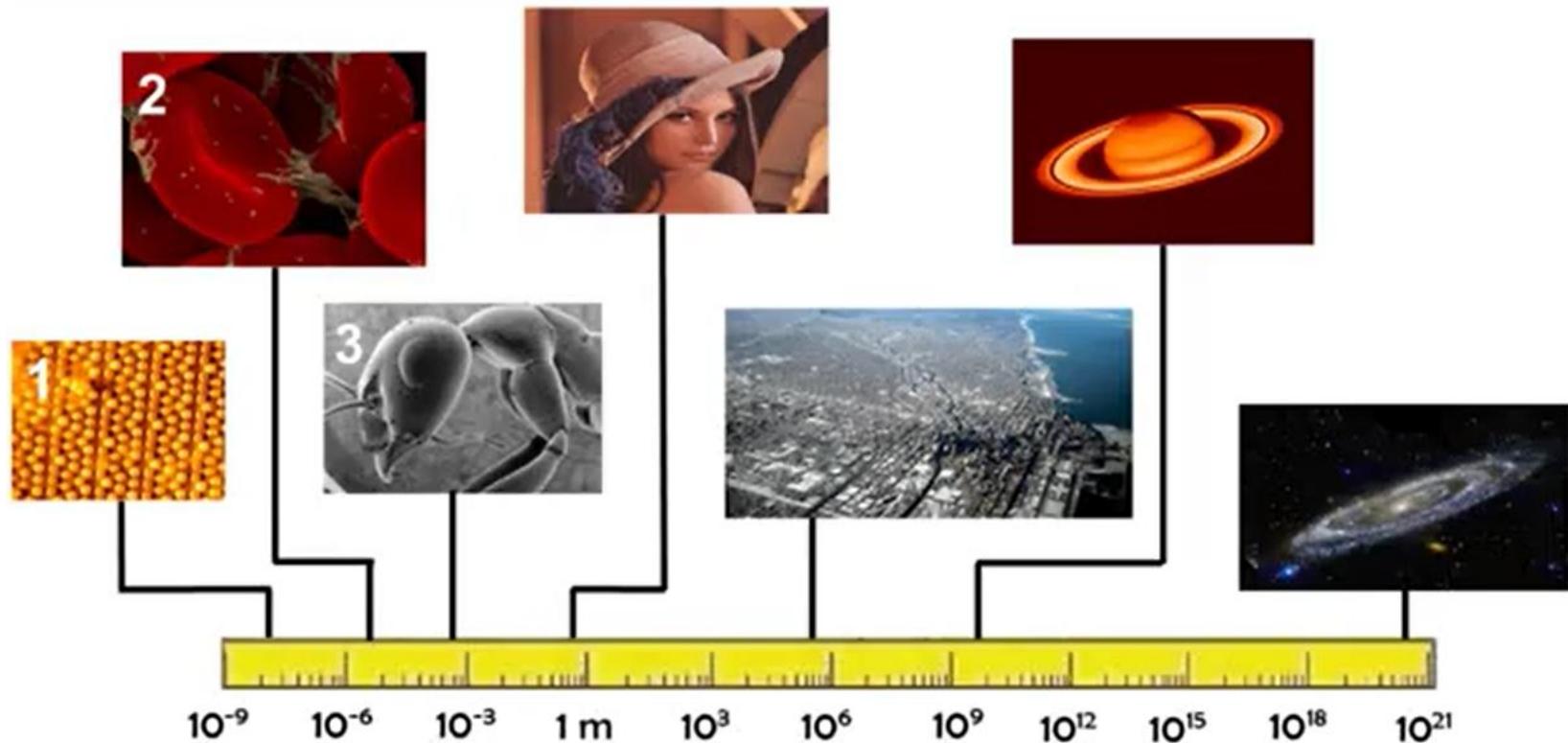


potential locations of oil/gas



Computer generated

# Range/scale of imaged objects



We have looked at:

- What is digital image and video processing?
- History of DIP
- Some important applications of DIP
- Key stages in DIP
- The human visual system
- Imaging beyond visible spectrum

Next we will begin to delve in details of the stages in DIP...

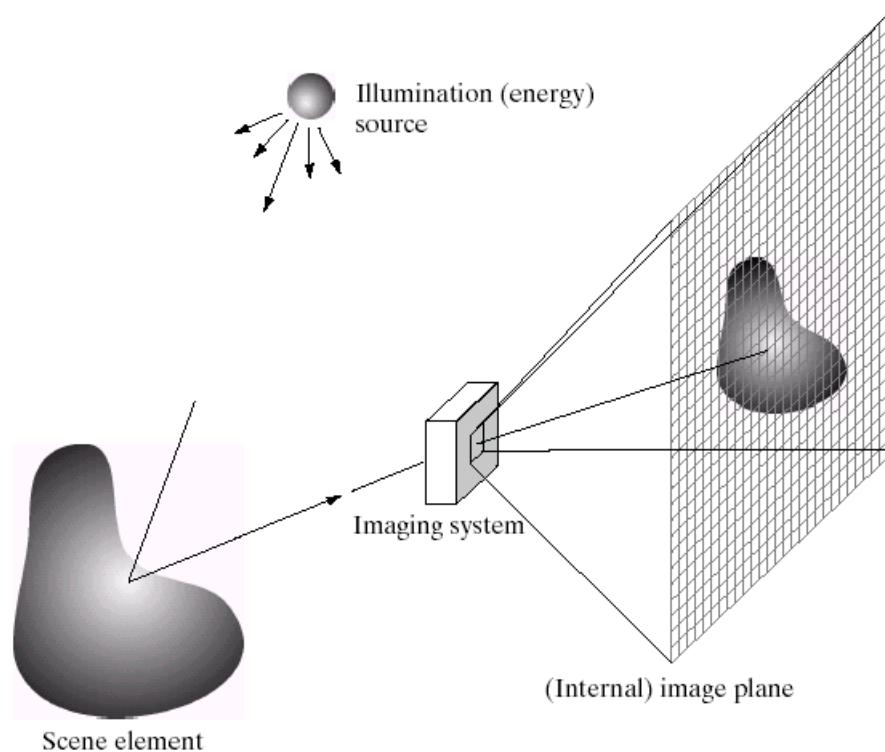
# Contents

This lecture will cover:

- Image acquisition, formation and sensing
- Sampling, quantisation
- Image representation
- Spatial, Intensity and temporal resolution

# Image Acquisition

Images are typically generated by *illuminating a scene* and absorbing the energy reflected by the objects in that scene



- Typical notions of illumination and scene can be way off:
  - X-rays of a skeleton
  - Ultrasound of an unborn baby
  - Electro-microscopic images of molecules

# Image Formation

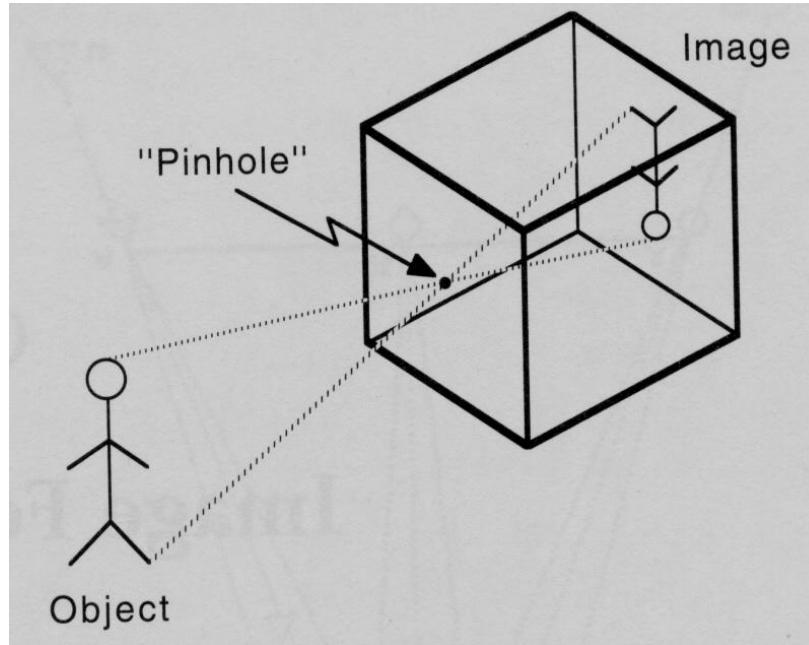
There are two parts to the image formation process:

- The **geometry of image formation**, which determines where in the image plane the projection of a point in the scene will be located.
- The **physics of light**, which determines the brightness of a point in the image plane as a function of illumination and surface properties.

# Pinhole camera

This is the simplest device to form an image of a 3D scene on a 2D surface.

Straight rays of light pass through a “pinhole” and form an inverted image of the object on the image plane.

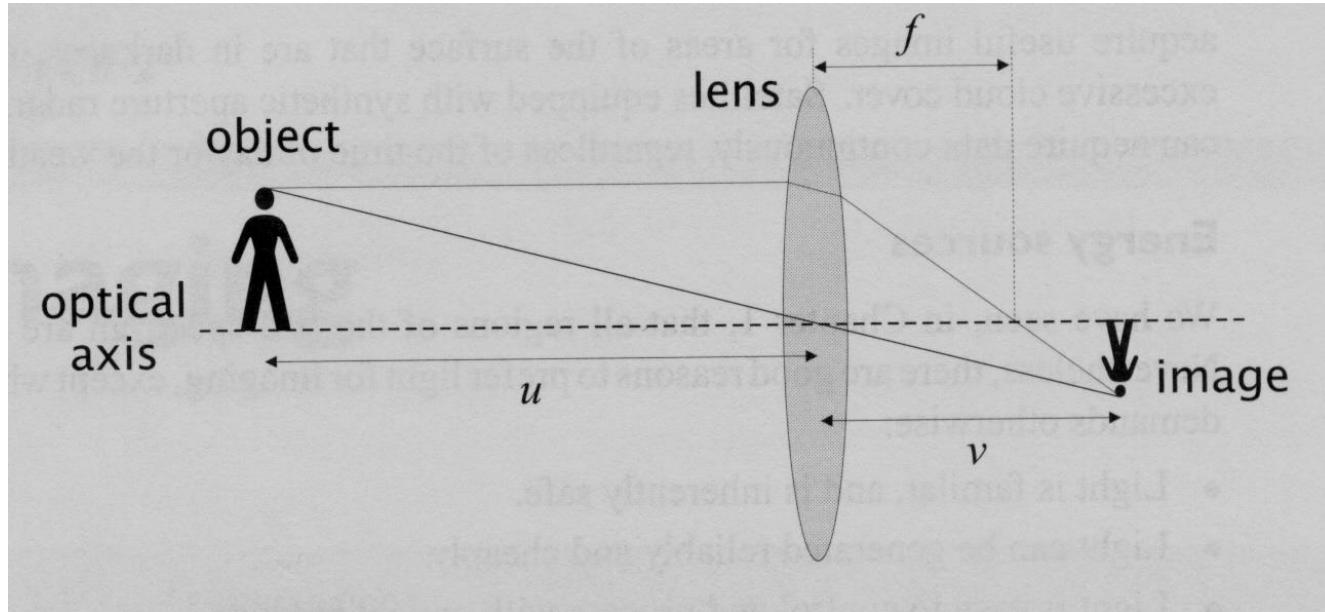


$$x = \frac{fx}{Z}$$

$$y = \frac{fy}{Z}$$

# Camera optics

- In practice, the aperture must be larger to admit more light.
- Lenses are placed to in the aperture to focus the bundle of rays from each scene point onto the corresponding point in the image plane



# A Simple Photometric Model

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

$f(x, y)$ : intensity at the point  $(x, y)$

$i(x, y)$ : illumination at the point  $(x, y)$

(the amount of source illumination incident on the scene)

$r(x, y)$ : reflectance/transmissivity at the point  $(x, y)$

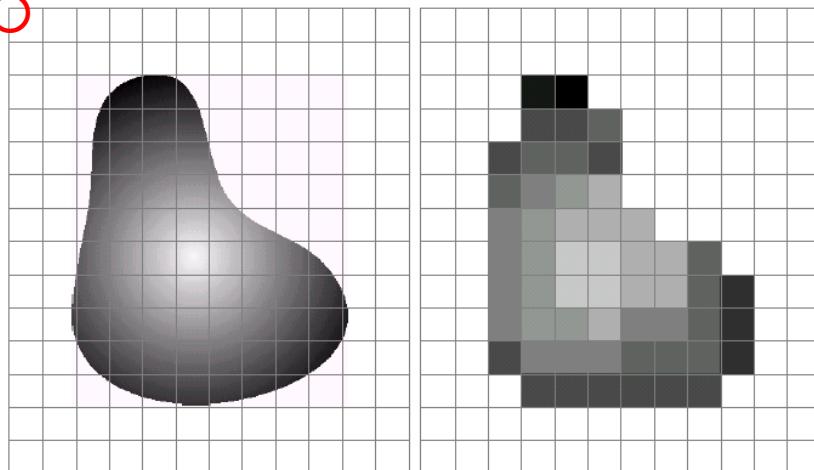
(the amount of illumination reflected/transmitted by the object)

where  $0 < i(x, y) < \infty$  and  $0 < r(x, y) < 1$

# Digital Image

- Function  $f : A \rightarrow B$ , where  $A$ : domain,  $B$ : range
  - Continuous –  $A, B$ : continuous
  - Discrete –  $A$ : discrete,  $B$ : continuous
  - Digital –  $A, B$ : discrete
- Scene  $g(x,y,z)$ : a 3-D continuous function
- Digital image  $f(x,y)$ : a 2-D digital function

Origin O

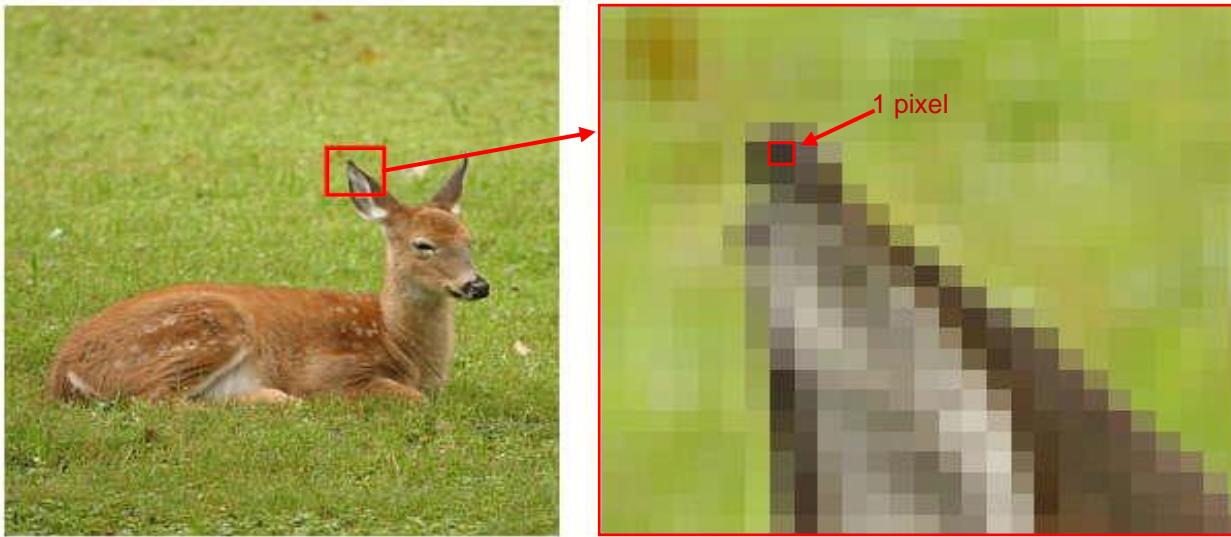


Pixel: picture element

# Digital Image (cont...)

Pixel values typically represent gray levels, colours, heights, opacities etc

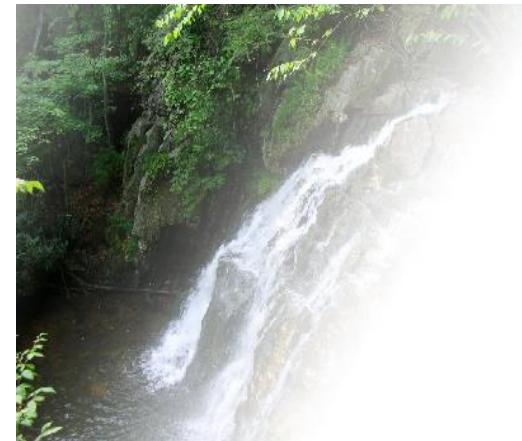
**Remember** *digitization* implies that a digital image is an *approximation* of a real scene



# Image types

Common image formats include:

- 1 sample per point (B&W or Grayscale)
- 3 samples per point (Red, Green, and Blue)
- 4 samples per point (Red, Green, Blue, and “Alpha”,  
a.k.a. Opacity)

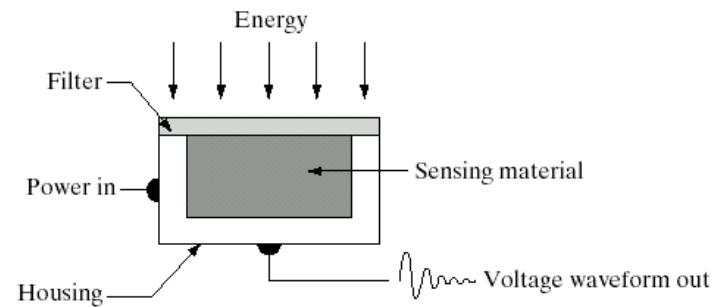


For most of this course we will focus on grey-scale images

# Digital cameras

A digital camera replaces film with a sensor array.

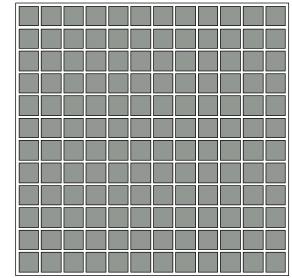
- Each cell in the array is light-sensitive diode that converts photons to electrons
- Two common types
  - Charge Coupled Device (CCD)
  - Complementary metal oxide semiconductor (CMOS)



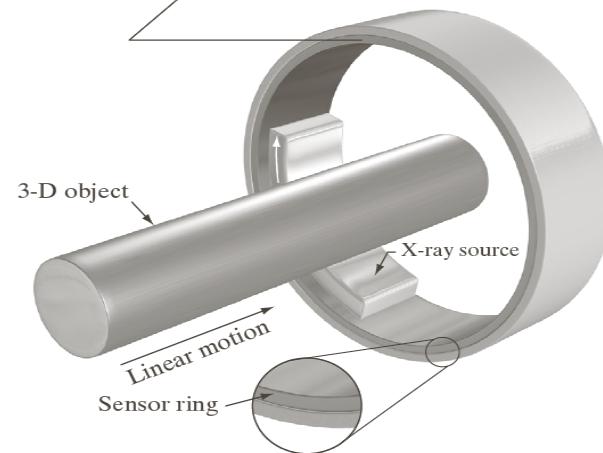
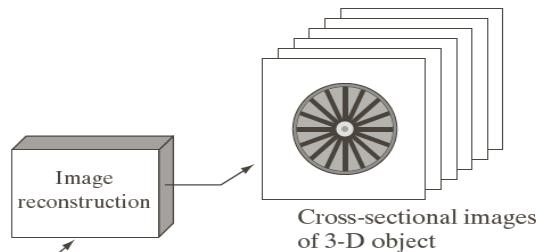
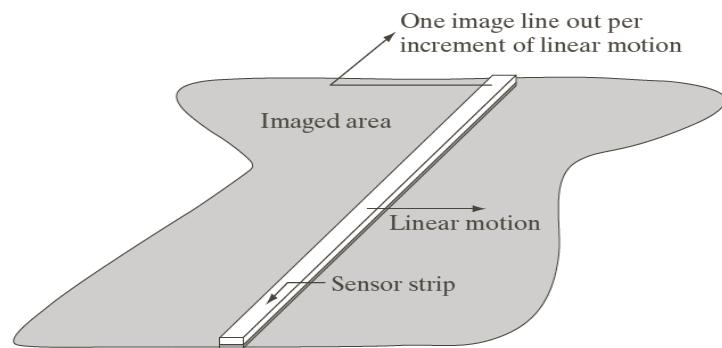
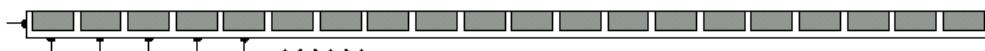
# Image Sensing

Sensors can be arranged in fixed 2D array or a line strip or a ring

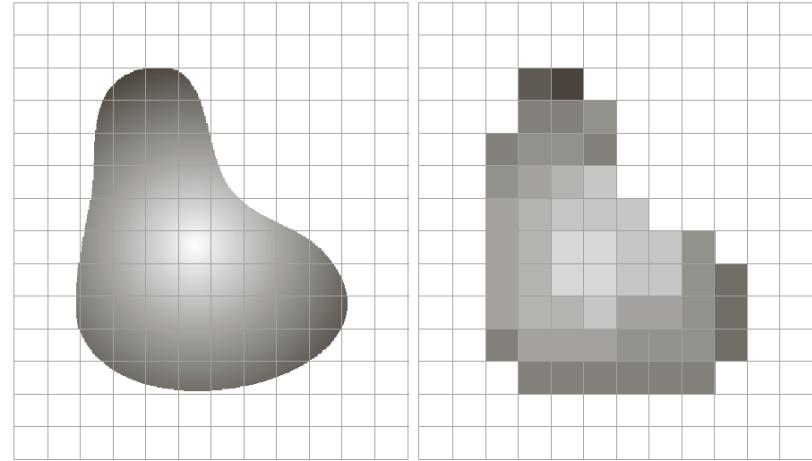
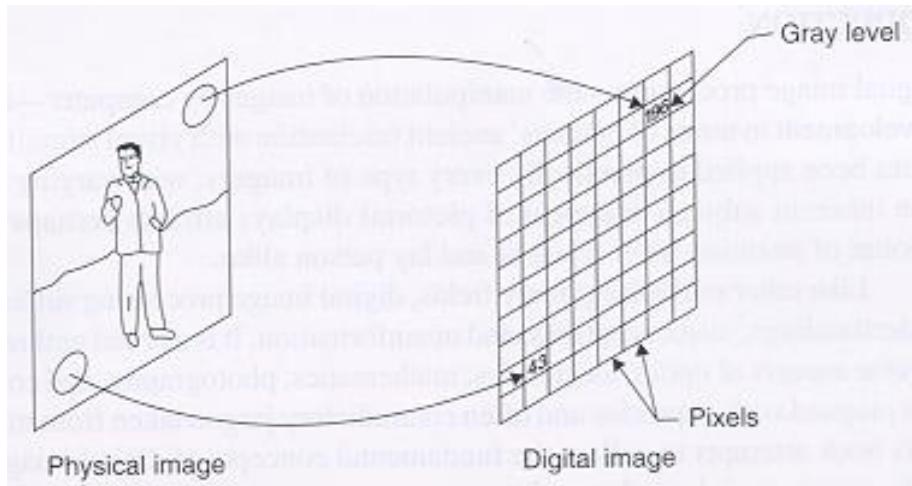
Array of Image Sensors



Line of Image Sensors



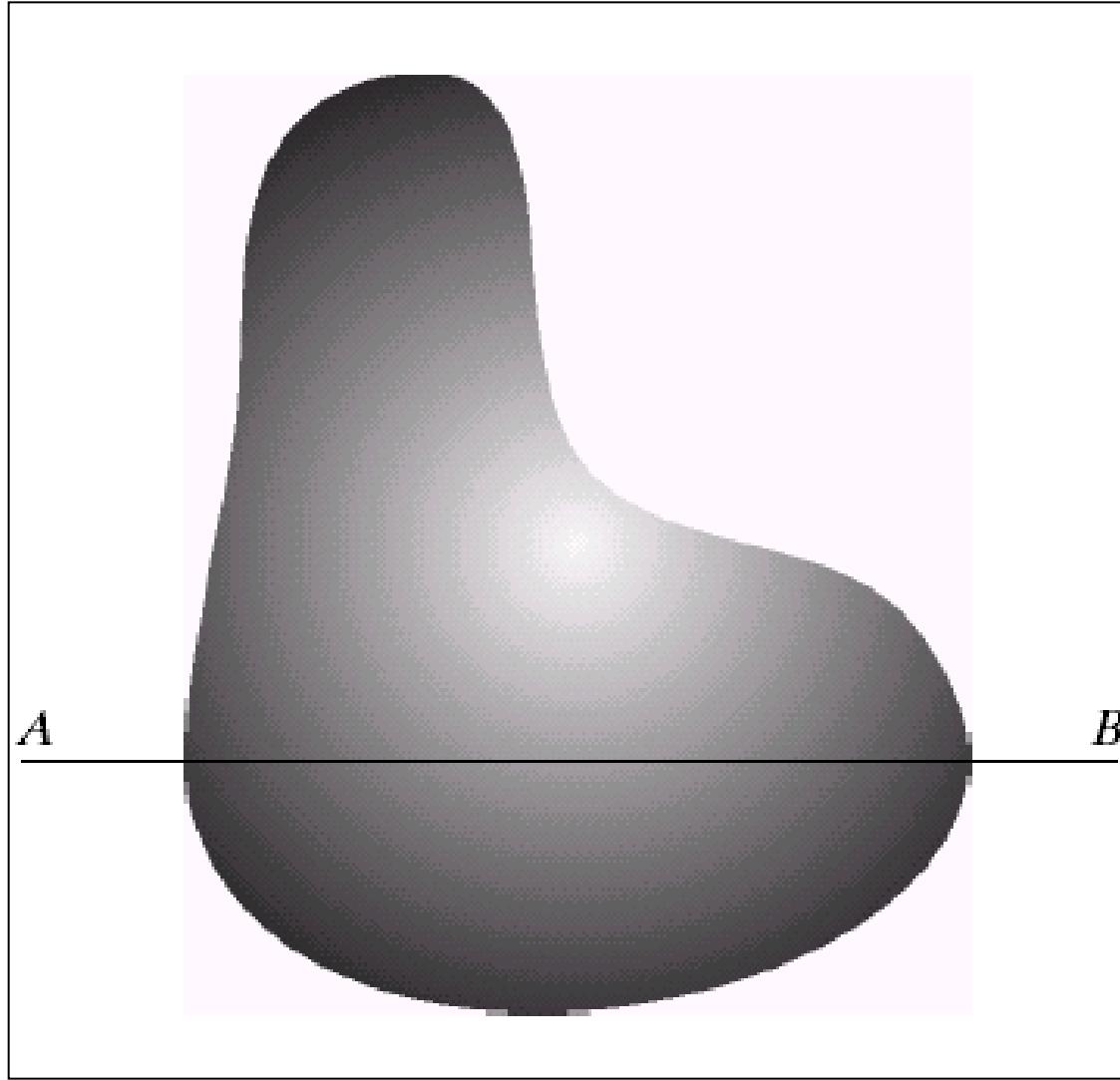
# Image Sampling And Quantisation



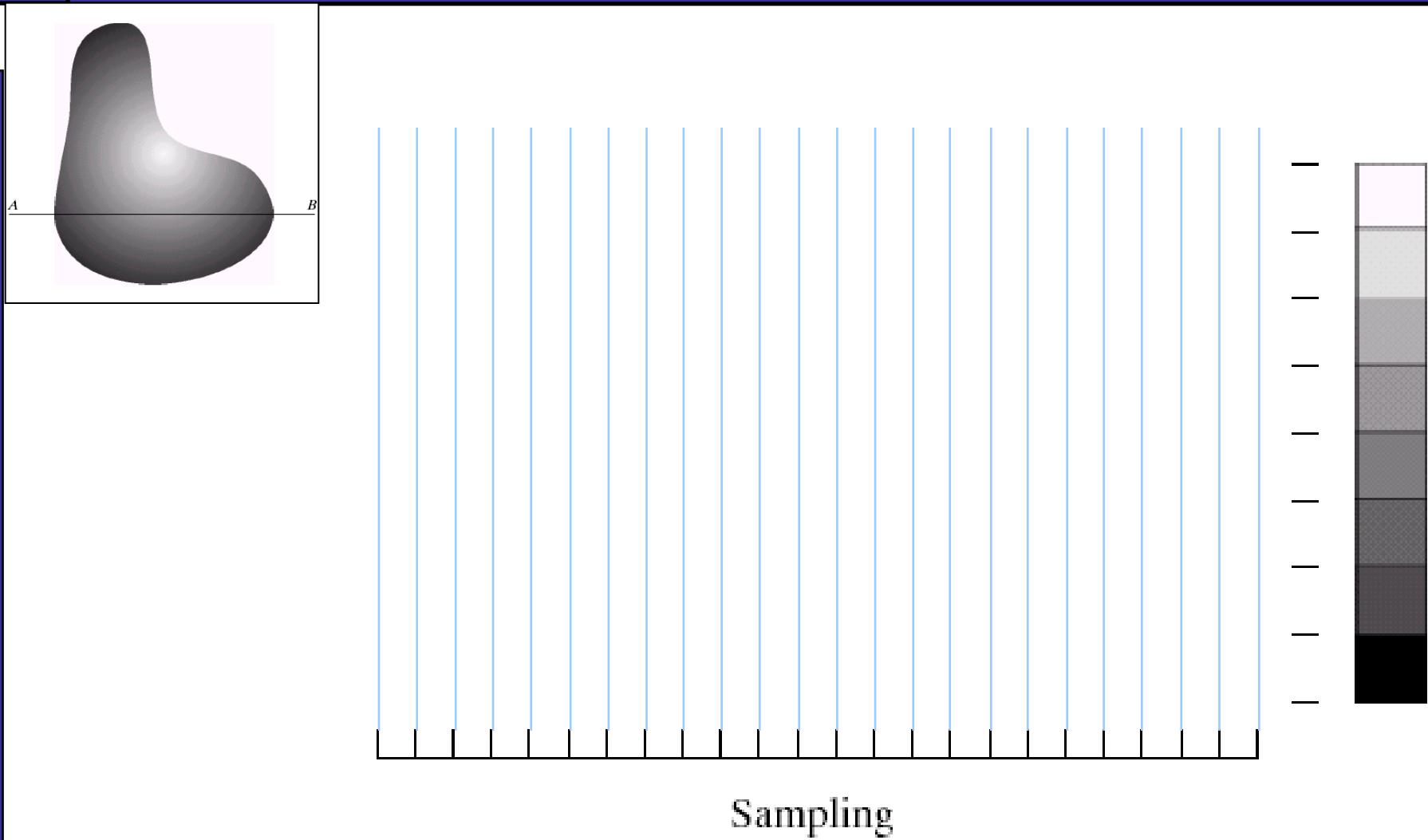
**Sampling:** measure the value of an image at a finite number of points.

**Quantization:** represent measured value (i.e., voltage) at the sampled point by a finite set of integer values.

# Image Sampling And Quantisation



# Image Sampling And Quantisation

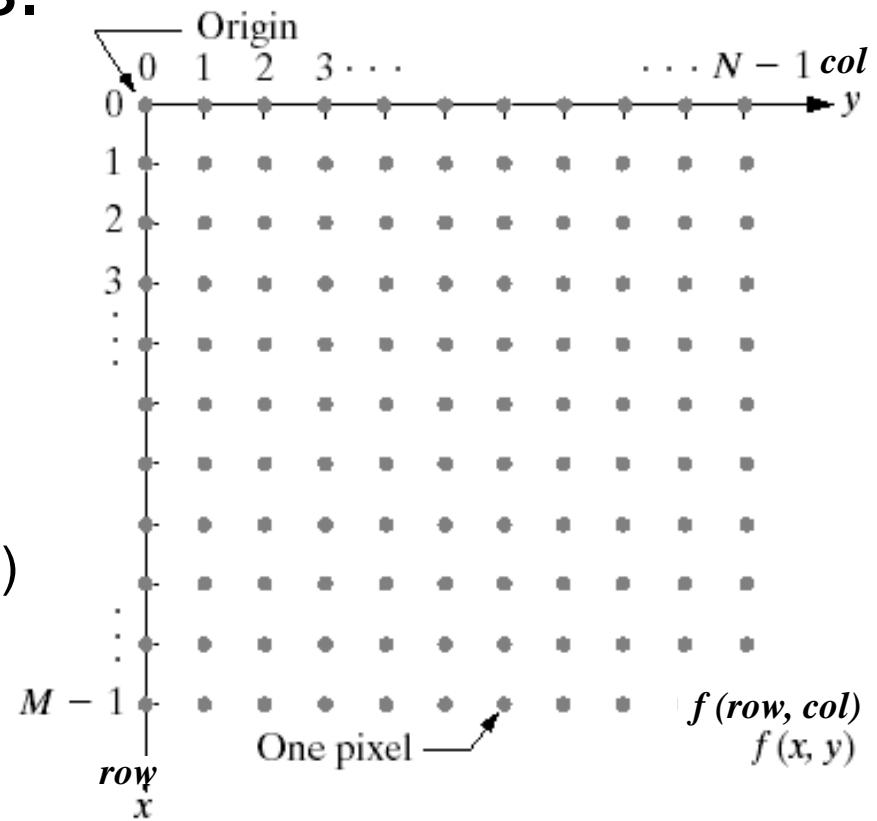


# Image Representation

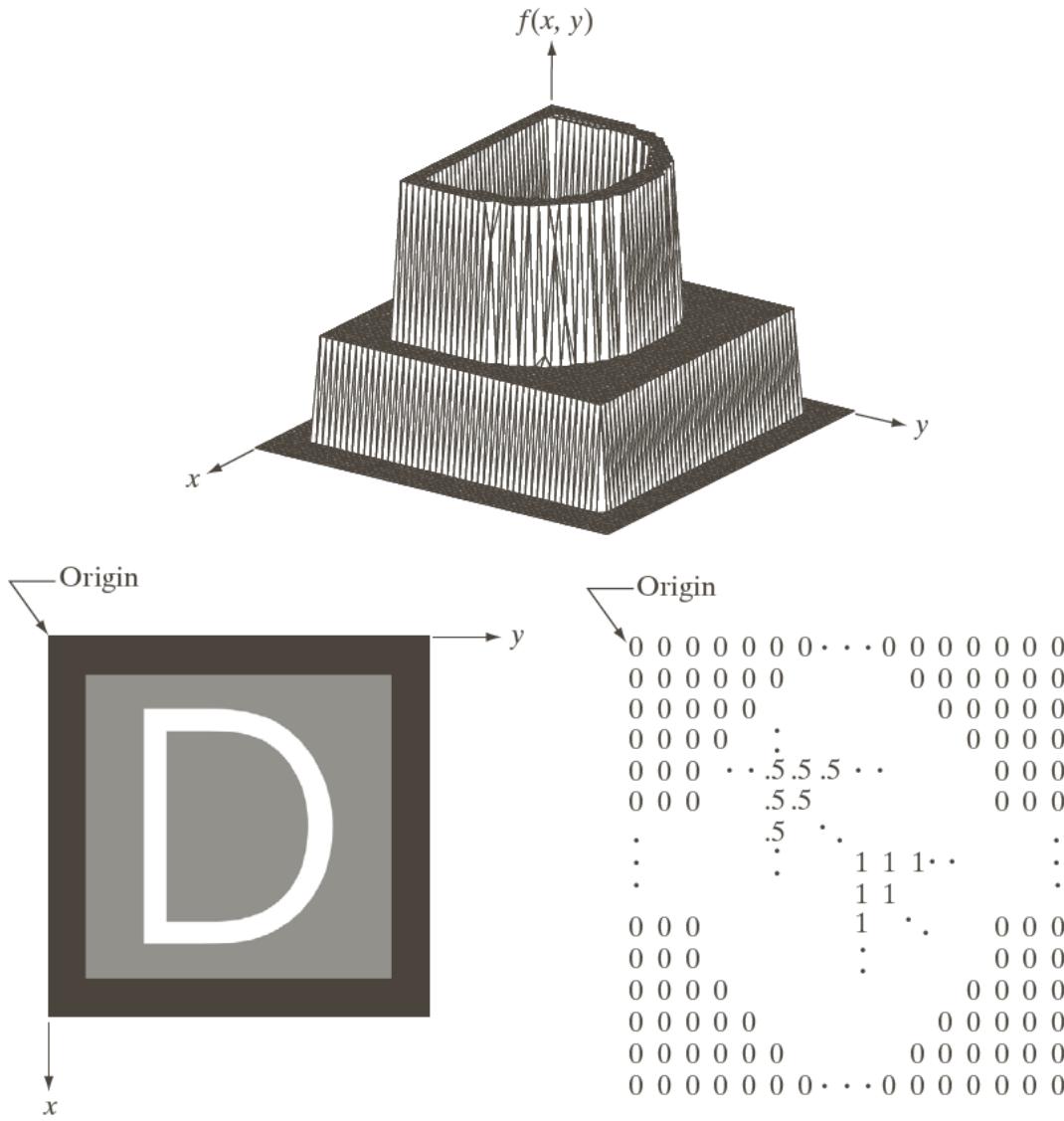
A digital image can easily be represented as matrices composed of  $M$  rows and  $N$  columns of pixel values.

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & \dots & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix}$$

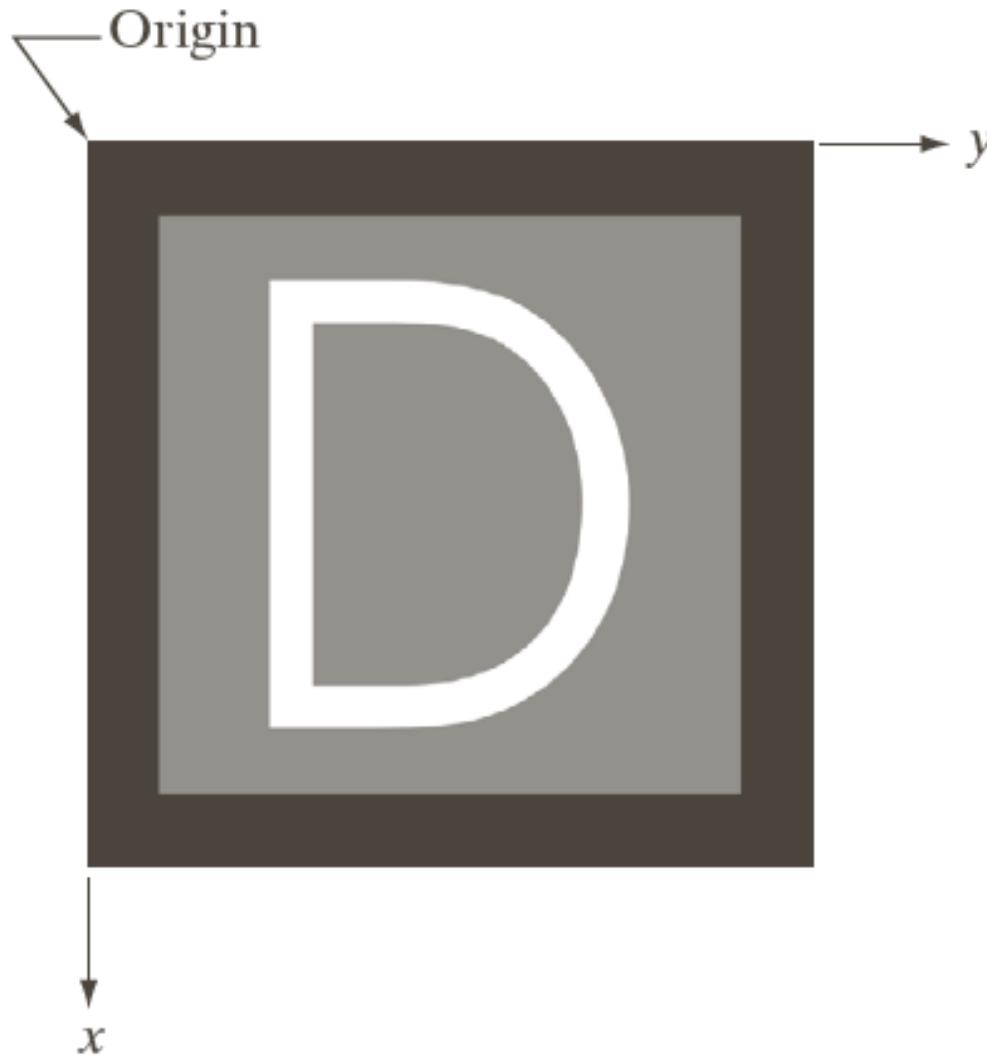
Pixel values are most often grey levels in the range 0-255(black-white)



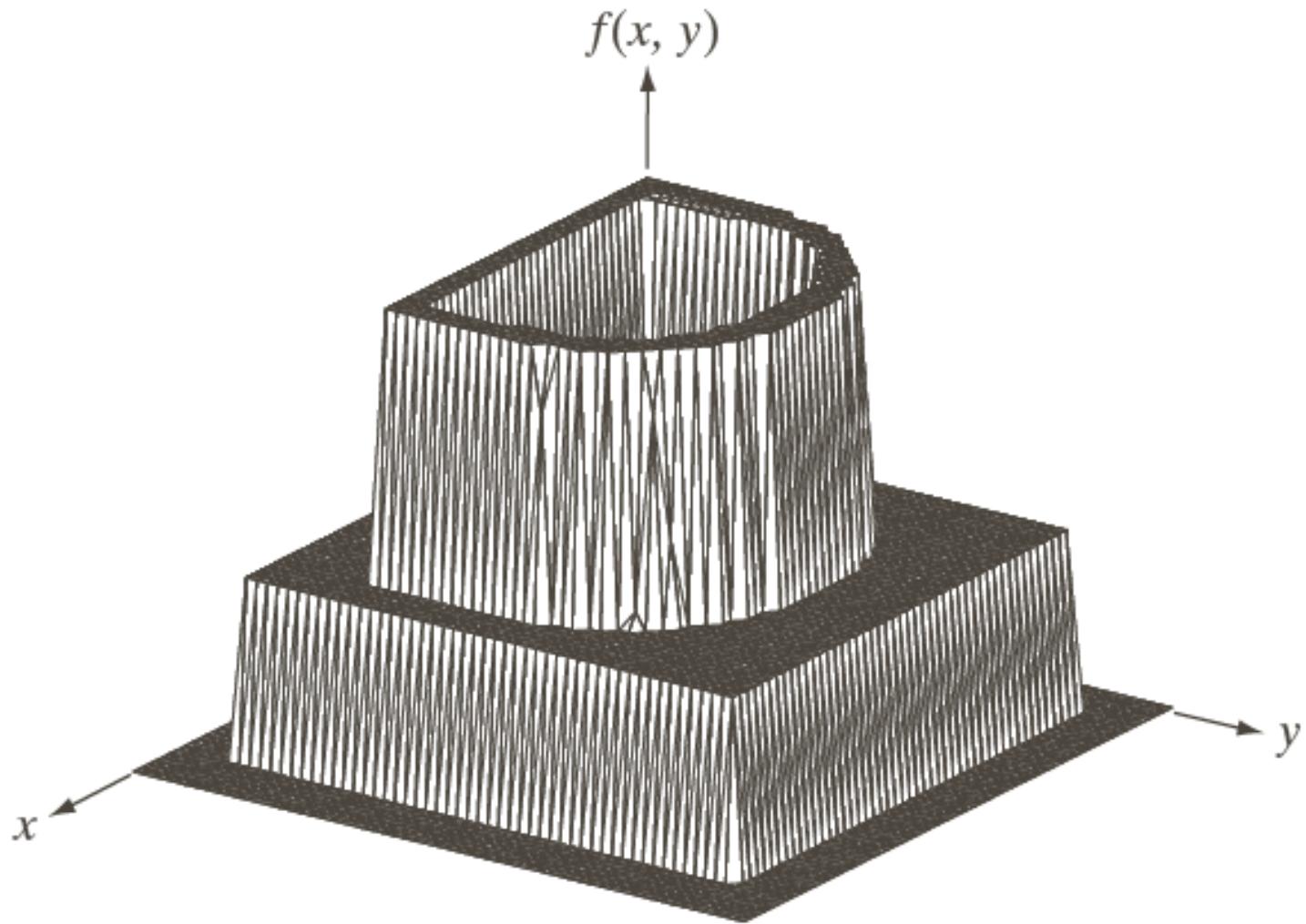
# Image Representation



# Image Representation



# Image Representation



# Image Representation

Origin

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	0 0 0 0 0
0 0 0 0	:
0 0 0 . . . 5 5 5 . . .	0 0 0
0 0 0 . 5 5	0 0 0
.	. 5 . .
:	. . . 1 1 1 . . .
:	1 1
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 0 0 0 0 0
0 0 0 0 0 0 . . . 0 0 0 0 0 0 0	

# Spatial Resolution

*The spatial resolution* of an image is determined by how sampling was carried out

Spatial resolution simply refers to the smallest discernable detail in an image



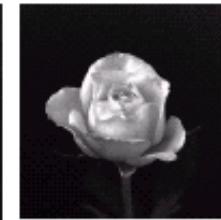
# Spatial Resolution (cont...)



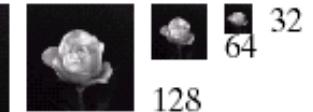
1024



512



256



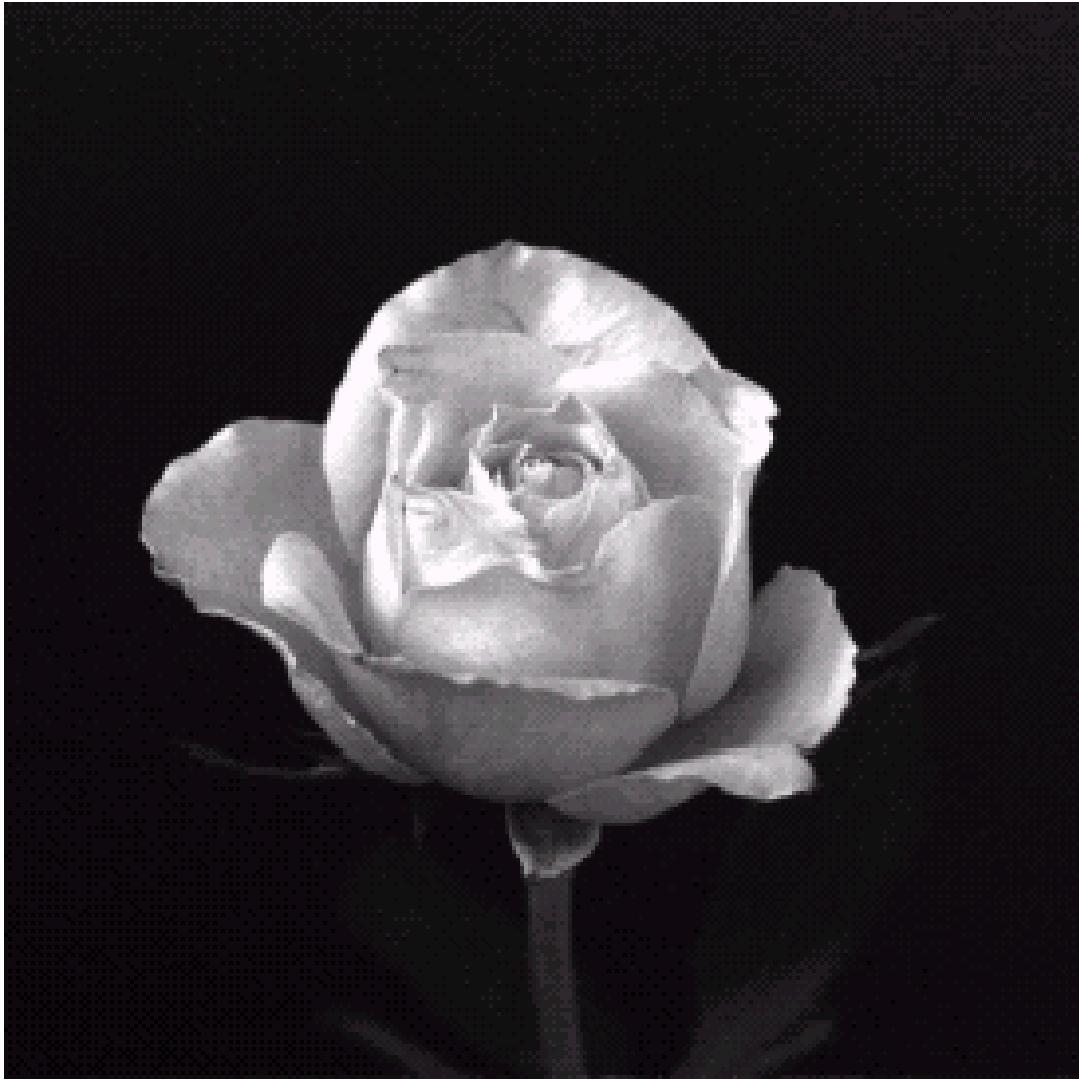
32

64

# Spatial Resolution (cont...)



# Spatial Resolution (cont...)



# Spatial Resolution (cont...)



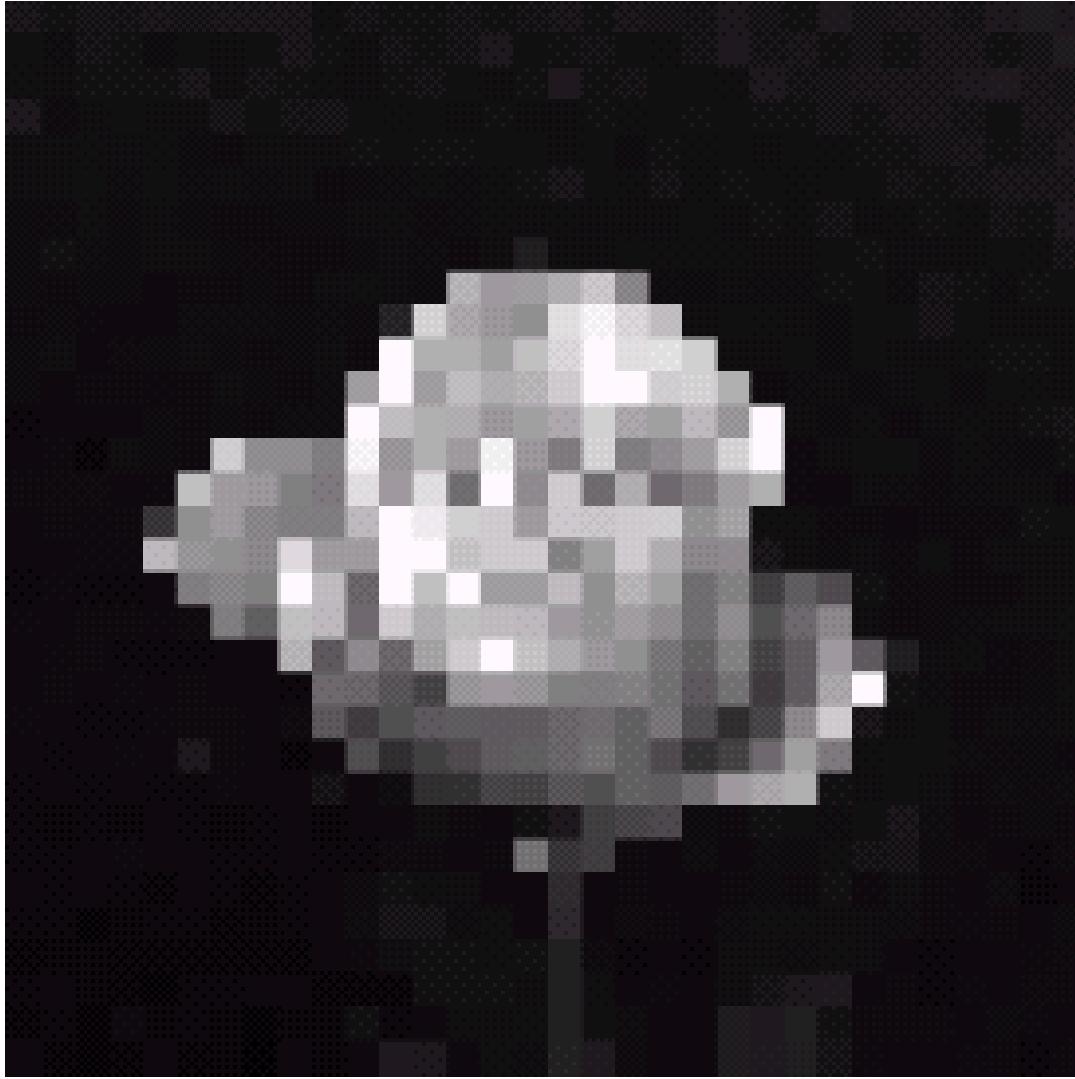
# Spatial Resolution (cont...)



# Spatial Resolution (cont...)



# Spatial Resolution (cont...)



# Intensity Level Resolution

*Intensity level resolution* refers to the number of intensity levels used to represent the image

- The more intensity levels used, the finer the level of detail discernable in an image
- Intensity level resolution is usually given in terms of the number of bits used to store each intensity level

Number of Bits	Number of Intensity Levels	Examples
1	2	0, 1
2	4	00, 01, 10, 11
4	16	0000, 0101, 1111
8	256	00110011, 01010101
16	65,536	1010101010101010

# Intensity Level Resolution (cont...)

256 grey levels (8 bits per pixel)



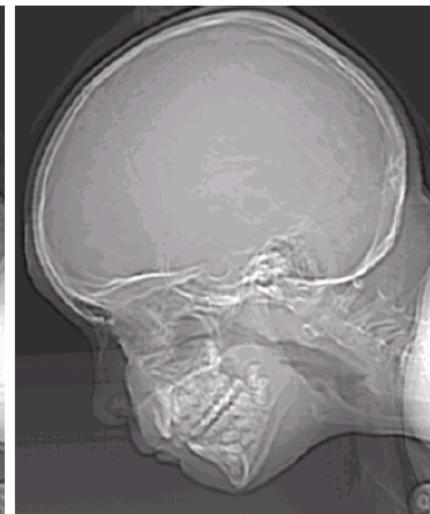
128 grey levels (7 bpp)



64 grey levels (6 bpp)



32 grey levels (5 bpp)



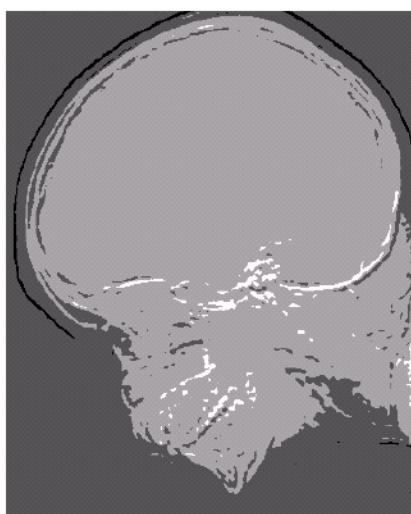
16 grey levels (4 bpp)



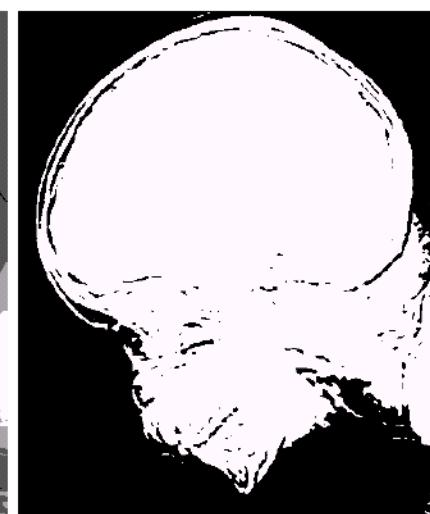
8 grey levels (3 bpp)



4 grey levels (2 bpp)



2 grey levels (1 bpp)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Resolution: How Much Is Enough?

The big question with resolution is always *how much is enough?*

- This all depends on what is in the image and what you would like to do with it
- Key questions include
  - Does the image look aesthetically pleasing?
  - Can you see what you need to see within the image?

# Resolution: How Much Is Enough? (cont...)



The picture on the right is fine for counting the number of cars, but not for reading the number plate

# Intensity Level Resolution (cont...)



Low Detail



Medium Detail



High Detail

# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Intensity Level Resolution (cont...)



# Temporal Resolution

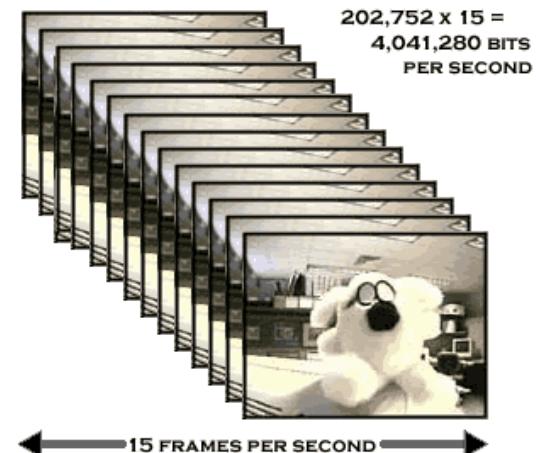
A Digital video can be thought of as a sequence of digital image. A video playback device creates the illusion of full motion by displaying a rapid sequence of changing images on a display device.



# Temporal resolution

## Frame Rate

- animation is an illusion caused by the rapid display of still images.
- television and movies play at 25-30 fps but acceptable playback can be achieved with 12-15 fps.



# Summary

We have looked at:

- Image acquisition, formation and sensing
- What is a Digital Image?
- Sampling, quantisation
- Image representation
- Spatial, Intensity, temporal resolution

Next time we start to look basic terminologies, relationship between pixels and common image operations

## What we discussed last time

- Image acquisition, formation and sensing
- Sampling, quantisation
- Image representation
- Spatial, Intensity and temporal resolution

- Q1. Consider an image with 100 lines and 1000 pixels per line. Each pixel can take 256 different values. The total amount of bits needed to store that image is \_\_\_\_ ?
- Q2. A color video has frame rate of 30 frames per second. Considering that spatial resolution of each frame (images) is  $1000 \times 1000$  pixels and every color component is stored using 8 bits then a minute of (uncompressed) video will occupy ?

Q3. An imaging system and object surface has illumination and reflectance range as:

$$1 \leq i(x, y) \leq 100 \text{ and } 0.5 \leq r(x, y) \leq 1$$

The acquired image is stored as grayscale with 8 bits per pixel. An intensity value of say 25 at any location will get stored as \_\_\_\_?

This lecture will cover:

- Neighbourhood and Connectivity
- Connected component labeling
- Distance measures
- Basic Image Operations

# Neighbors of a Pixel

A pixel  $p$  at coordinates  $(x,y)$  has four *horizontal* and *vertical* neighbors whose coordinates are given by:

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

	$(x, y-1)$	
$(x-1, y)$	$P(x, y)$	$(x+1, y)$
	$(x, y+1)$	

This set of pixels, called the *4-neighbors* of  $p$ , is denoted by  $N_4(p)$ .

# Neighbors of a Pixel

The four *diagonal* neighbors of  $p$  have coordinates:

$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$

and are denoted by  $N_D(p)$ .

$(x-1, y+1)$		$(x+1, y-1)$
	$P(x,y)$	
$(x-1, y-1)$		$(x+1, y+1)$

These points, together with the 4-neighbors, are called the 8-neighbors of  $p$ , denoted by  $N_8(p)$ .

$(x-1, y+1)$	$(x, y-1)$	$(x+1, y-1)$
$(x-1, y)$	$P(x,y)$	$(x+1, y)$
$(x-1, y-1)$	$(x, y+1)$	$(x+1, y+1)$

# Pixel Connectivity

Connectivity between pixels is important:

- Because it is used to determine components/regions in an image, trace contours and establish boundaries of objects

Two pixels are connected if:

- They are “neighbors” in some sense (i.e.  $N_4(p)$ ,  $N_8(p)$ , ...)
- Their gray levels satisfy a specified criterion of similarity (e.g. equality, ...)

# Adjacency

Let  $V$ : a set of intensity values used to define adjacency.

In a binary image,  $V = \{1\}$ , if we are referring to adjacency of pixels with value 1.

In a gray-scale image, the idea is the same, but  $V$  typically contains more elements, for example,  $V = \{180, 181, 182, \dots, 200\}$

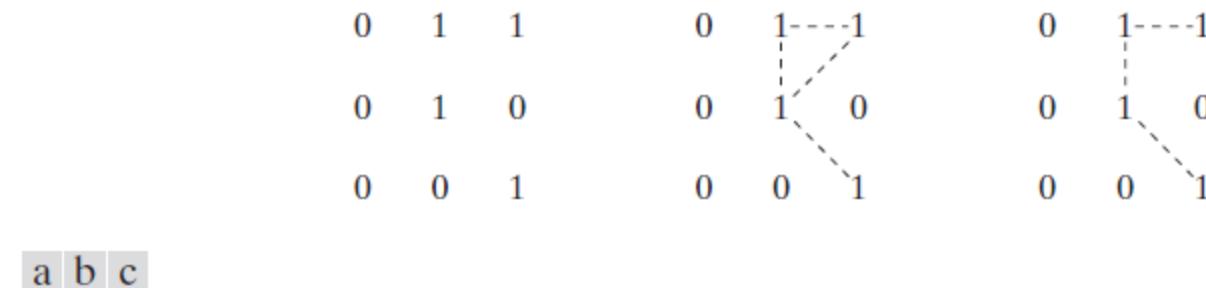
# Types of Adjacency

1. **4-adjacency:** Two pixels  $p$  and  $q$  with values from  $V$  are 4-adjacent if  $q$  is in the set  $N_4(p)$ .
2. **8-adjacency:** Two pixels  $p$  and  $q$  with values from  $V$  are 8-adjacent if  $q$  is in the set  $N_8(p)$ .
3. **m-adjacency (mixed):** Two pixels  $p$  and  $q$  with values from  $V$  are m-adjacent if :
  - $q$  is in  $N_4(p)$  **or**
  - $q$  is in  $N_D(p)$  **and** the set  $N_4(p) \cap N_4(q)$  has no pixel whose values are from  $V$  (no intersection)

# Types of Adjacency

Mixed adjacency eliminates the ambiguities that often arise when 8-adjacency is used.

Eg.



**FIGURE 2.26** (a) Arrangement of pixels; (b) pixels that are 8-adjacent (shown dashed) to the center pixel; (c)  $m$ -adjacency.

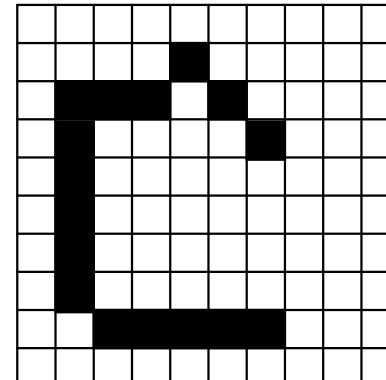
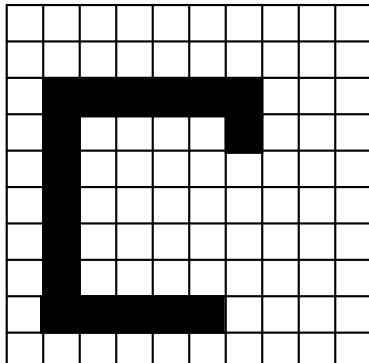
# A Digital Path

A digital path (or curve) from pixel  $p$  with coordinate  $(x,y)$  to pixel  $q$  with coordinate  $(s,t)$  is a sequence of distinct pixels:

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n),$$

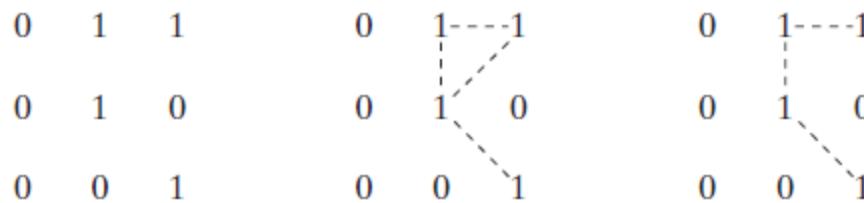
where  $(x_0, y_0) = (x, y)$  and  $(x_n, y_n) = (s, t)$  and

$(x_i, y_i)$  and  $(x_{i-1}, y_{i-1})$  are adjacent for  $1 \leq i \leq n$



# A Digital Path

Return to the previous example:



a b c

**FIGURE 2.26** (a) Arrangement of pixels; (b) pixels that are 8-adjacent (shown dashed) to the center pixel; (c)  $m$ -adjacency.

In figure (b) the paths between the top right and bottom right pixels are 8-paths. And the path between the same 2 pixels in figure (c) is  $m$ -path

# Connectivity

- Let  $R$  represent a subset of pixels in an image.
- A pixel pair  $p$  and  $q$  are said to be connected in  $R$  if there exists a path between them consisting entirely of pixels in  $R$ .
- A **connected component** is a maximal set of pixels in  $R$  that is connected.
- There can be more than one such set within a given  $R$ .
- Set  $R$  is a **connected set** if it has only one connected component.

# Region and Boundary

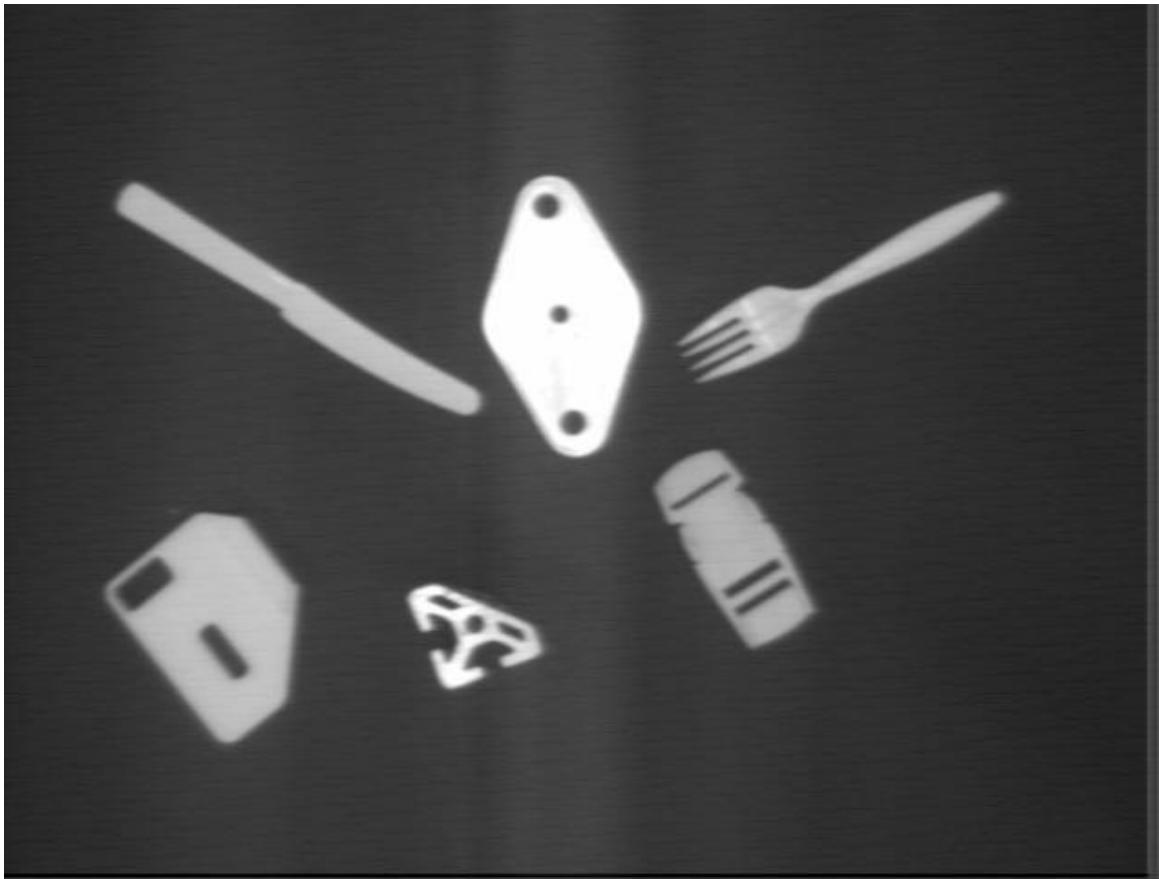
## Region

Let  $R$  be a subset of pixels, we call  $R$  a region of the image if  $R$  is a connected set.

## Boundary

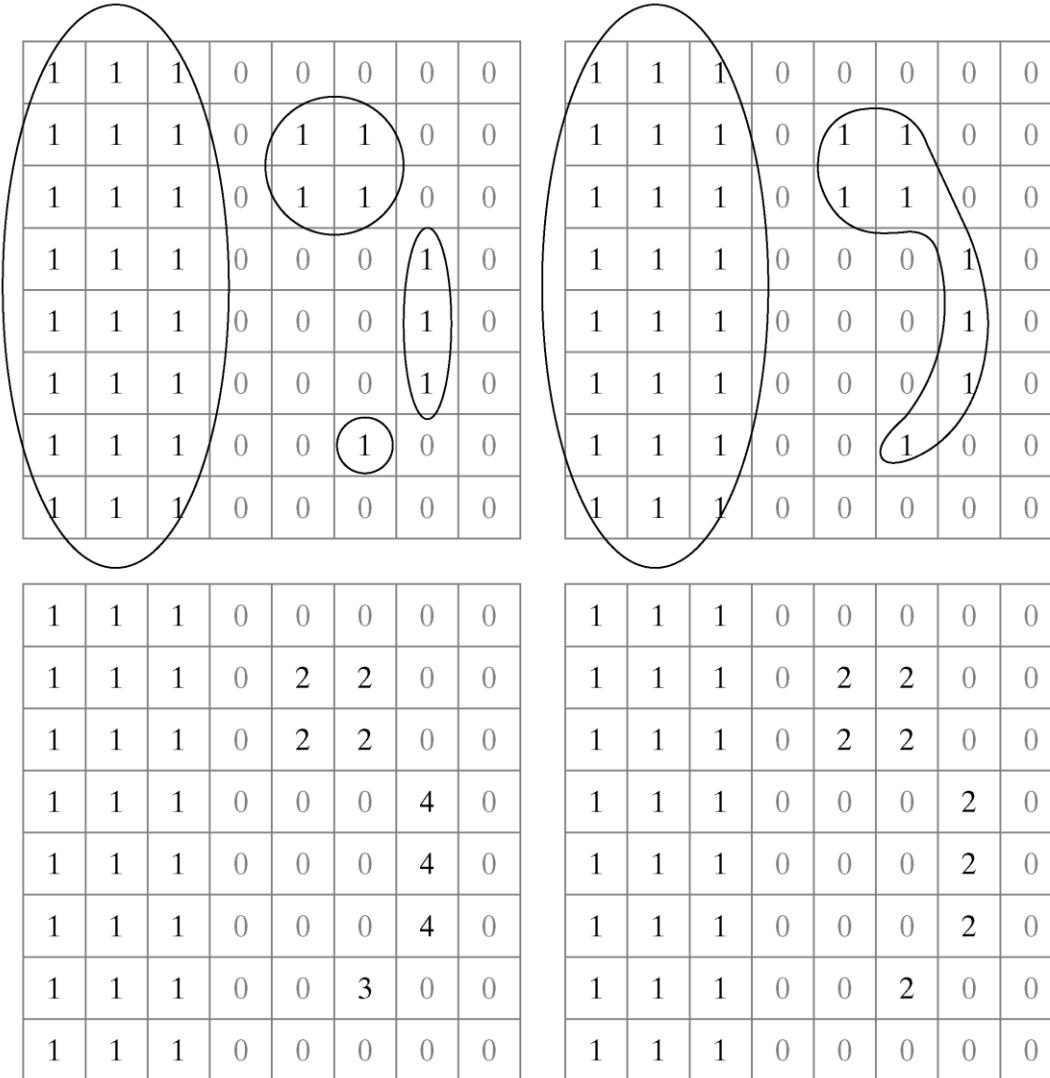
The *boundary* (also called *border* or *contour*) of a region  $R$  is the set of pixels in the region that have one or more neighbors that are not in  $R$ .

# Application



Need to **SEGMENT** image into separate **COMPONENTS**(regions)

# Connected Components



a	b
c	d

**FIGURE 9.19**

Connected components  
(a) Four 4-connected components.

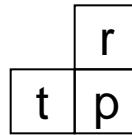
(b) Two 8-connected components.

(c) Label matrix obtained using 4-connectivity

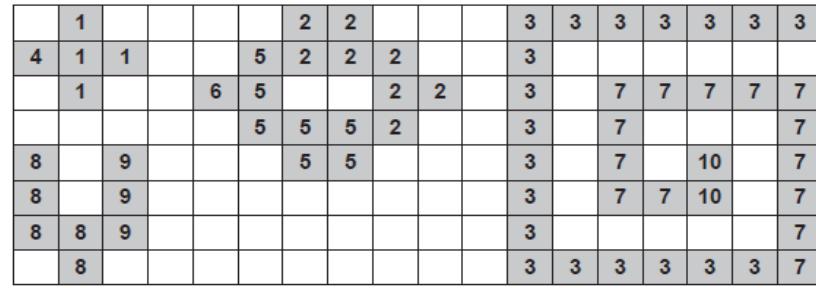
(d) Label matrix obtained using 8-connectivity.

# Connected Components Labeling

## First Pass



- Examine pixels in scanline order
- When we visit point p, points r and t have been visited and labeled.
- If  $p=0$ : no action;
- If  $p=1$ : check r and t.
  - both  $r$  and  $t = 0$ ; assign new label to p;
  - only one of  $r$  and  $t$  is a 1. assign its label to p;
  - both  $r$  and  $t$  are 1:
    - same label => assign it to p;
    - different label=> assign lowest no. to p and



Mark labels in neighbors as equivalent (they are the same.)

# Connected Components Labeling

## □ Resolve Equivalences

- **Union-Find** Structure and operations
- Obtain transitive closure using **Floyd-Warshall** (F-W) algorithm

## Second Pass

- examine the pixels in scanline order
- Replace the label with equivalent component label

$$\begin{aligned}
 4 &\equiv 1 \\
 6 &\equiv 5, 2 \\
 9 &\equiv 8 \\
 10 &\equiv 7, 3
 \end{aligned}$$

	1				2	2			3	3	3	3	3	3	3
1	1	1			2	2	2	2	3						
	1				2	2			2	2	3	3	3	3	3
						2	2	2	2	3	3	3			3
8		8				2	2			3	3		3		3
8		8							3	3	3	3			3
8	8	8							3						3
	8								3	3	3	3	3	3	3

(d)

# Distance Measures

For pixels  $p$ ,  $q$  and  $z$ , with coordinates  $(x,y)$ ,  $(s,t)$  and  $(v,w)$ , respectively,  $D$  is a distance function if:

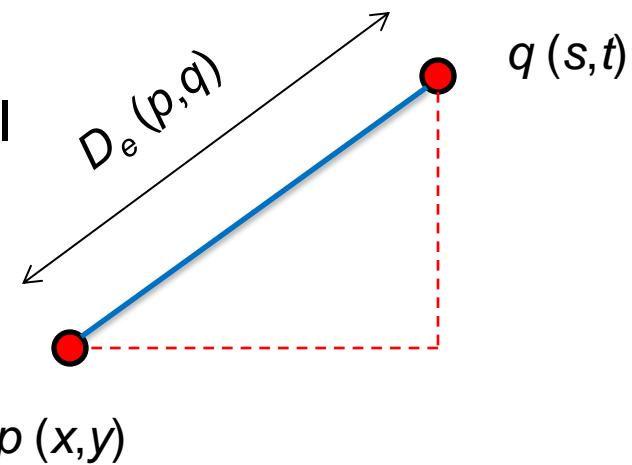
- (a)  $D(p,q) \geq 0$  ( $D(p,q) = 0$  iff  $p = q$ ),
- (b)  $D(p,q) = D(q,p)$ , and
- (c)  $D(p,z) \leq D(p,q) + D(q,z)$ .

# Distance Measures

The ***Euclidean Distance*** between  $p$  and  $q$  is defined as:

$$D_e(p, q) = [(x - s)^2 + (y - t)^2]^{1/2}$$

Pixels having a distance less than or equal to some value  $r$  from  $(x,y)$  are the points contained in a disk of radius  $r$  centered at  $(x,y)$

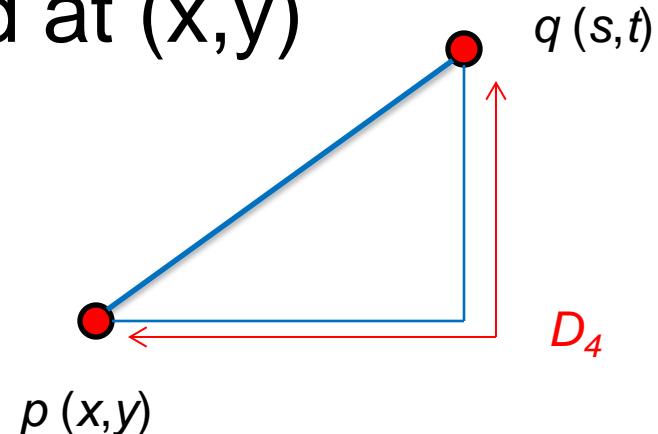


# Distance Measures

The  $D_4$  **distance** (also called **city-block distance**) between  $p$  and  $q$  is defined as:

- $D_4(p, q) = |x - s| + |y - t|$
- forms a Diamond centered at  $(x, y)$
- e.g. pixels with  $D_4 \leq 2$  from  $p$

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

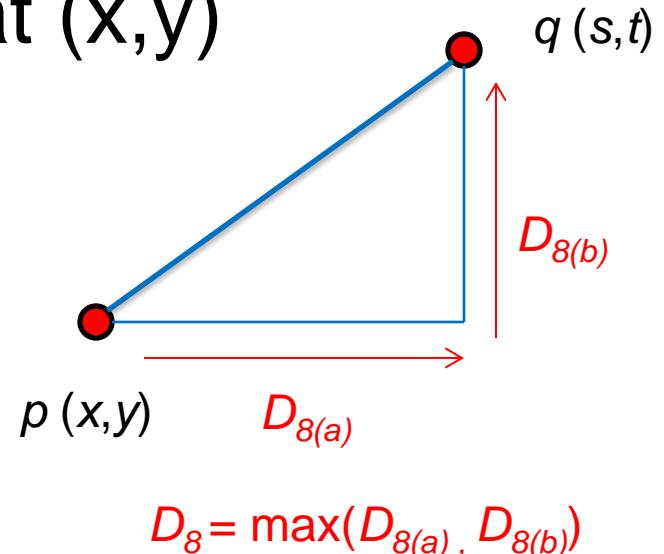


# Distance Measures

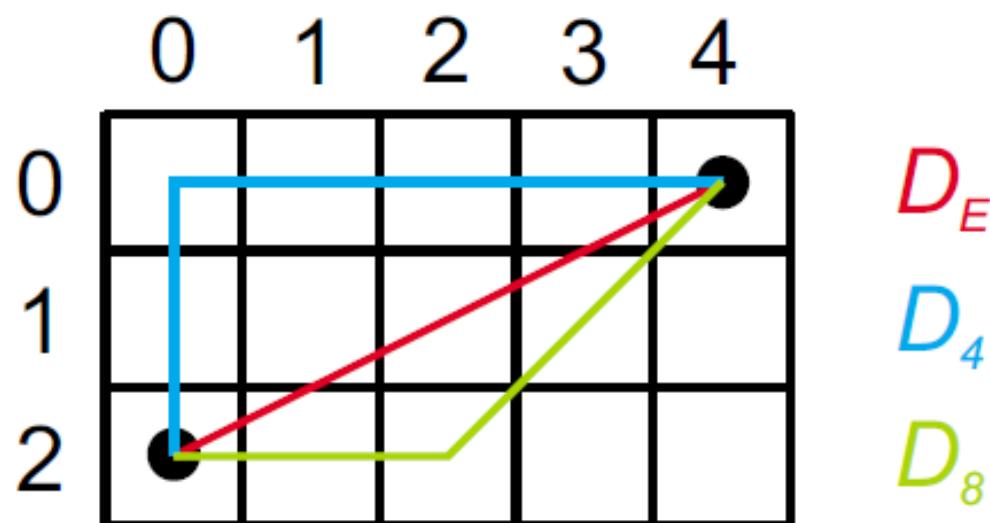
The  **$D_8$  distance** (also called **chessboard distance**) between  $p$  and  $q$  is defined as:

- $D_8(p, q) = \max(|x - s|, |y - t|)$
- forms a square centered at  $(x, y)$
- e.g. pixels with  $D_8 \leq 2$  from  $p$

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2



$$D_8 = \max(D_{8(a)}, D_{8(b)})$$

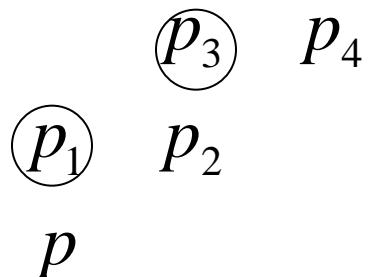


# Distance Measures

$D_4$  and  $D_8$  distances between p and q are independent of any paths that exist between the points

**D<sub>m</sub> distance:** is defined as the shortest m-path between the points. In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors.

e.g. assume     $p, p_2, p_4 = 1$   
 $p_1, p_3 = \text{can have either } 0 \text{ or } 1$



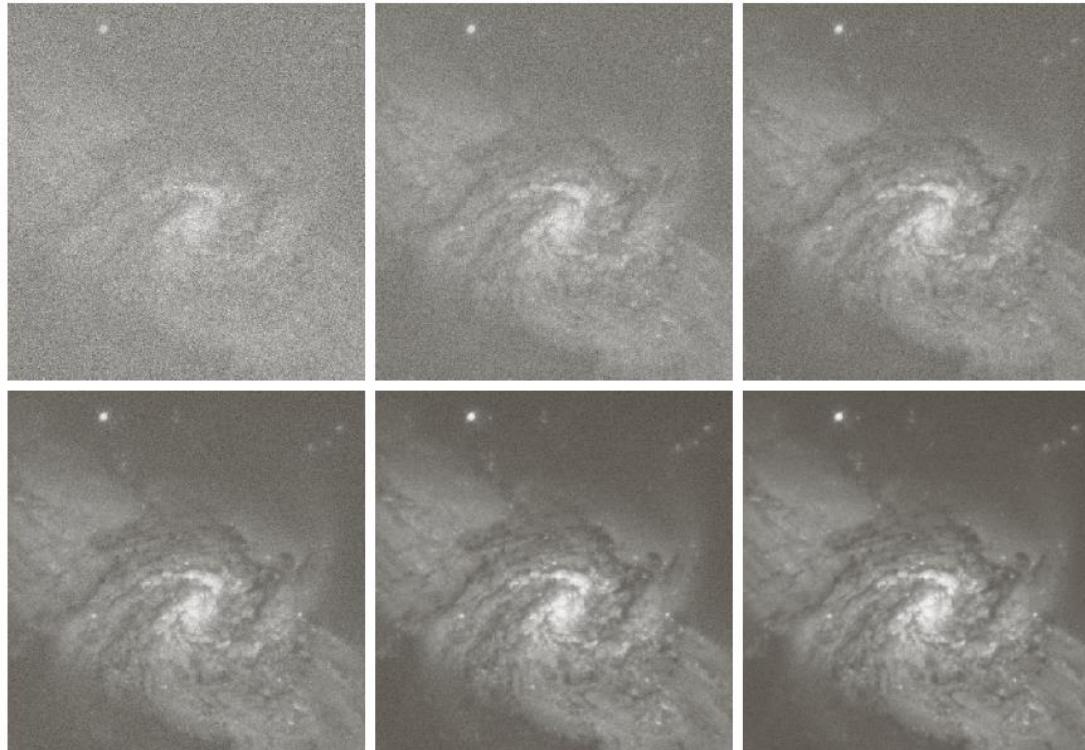
If only connectivity of pixels valued 1 is allowed, and  $p_1$  and  $p_3$  are 0, the m-distance between  $p$  and  $p_4$  is 2.

If either  $p_1$  or  $p_3$  is 1, the distance is 3.

If both  $p_1$  and  $p_3$  are 1, the distance is 4  
( $pp_1p_2p_3p_4$ )

# Basic Image Operations

## Arithmetic Operations - Addition

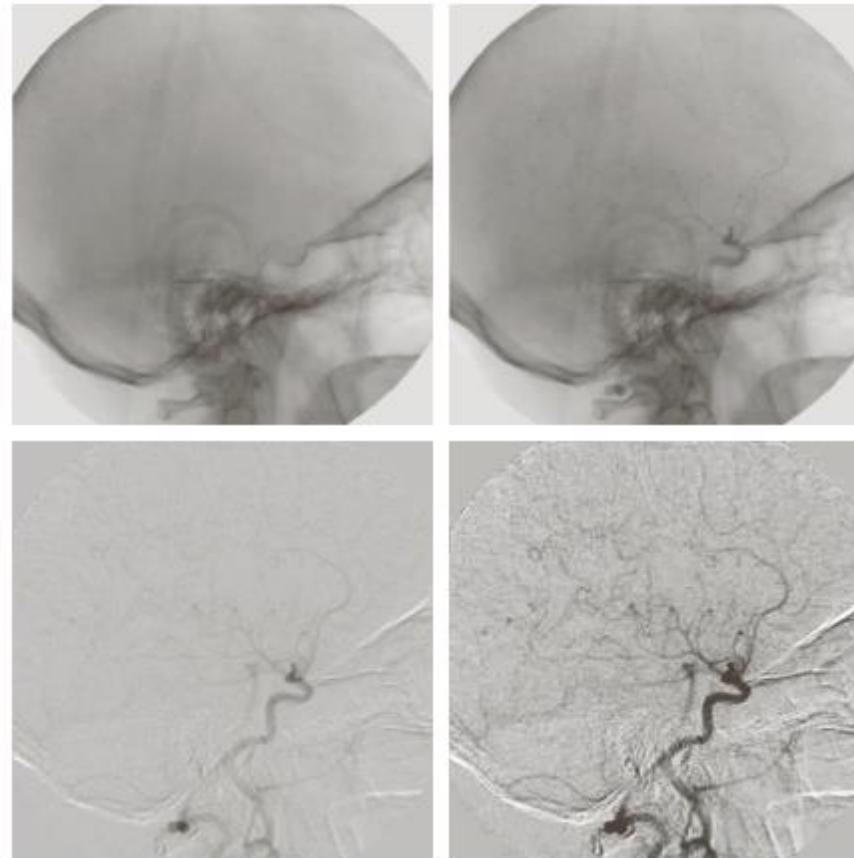


a b c  
d e f

**FIGURE 2.26** (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

# Basic Image Operations

## Arithmetic Operations - Subtraction



a b  
c d

**FIGURE 2.28**  
Digital subtraction angiography.  
(a) Mask image.  
(b) A live image.  
(c) Difference between (a) and (b). (d) Enhanced difference image.  
(Figures (a) and (b) courtesy of The Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

# Basic Image Operations

## Arithmetic Operations - Multiplication

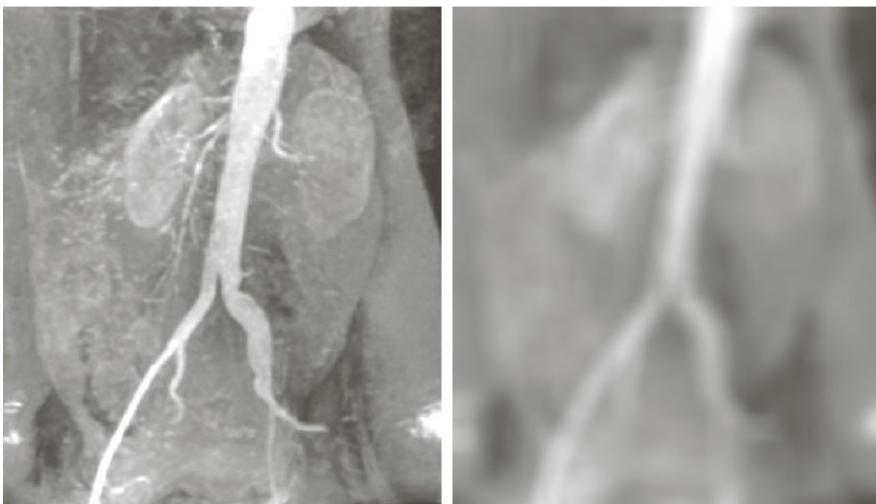
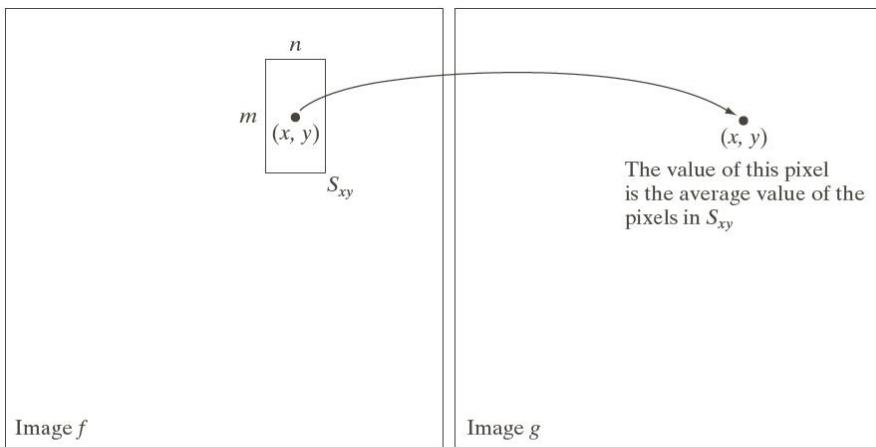


a | b | c

**FIGURE 2.30** (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

# Basic Image Operations

## Spatial Operations – Neighborhood operations

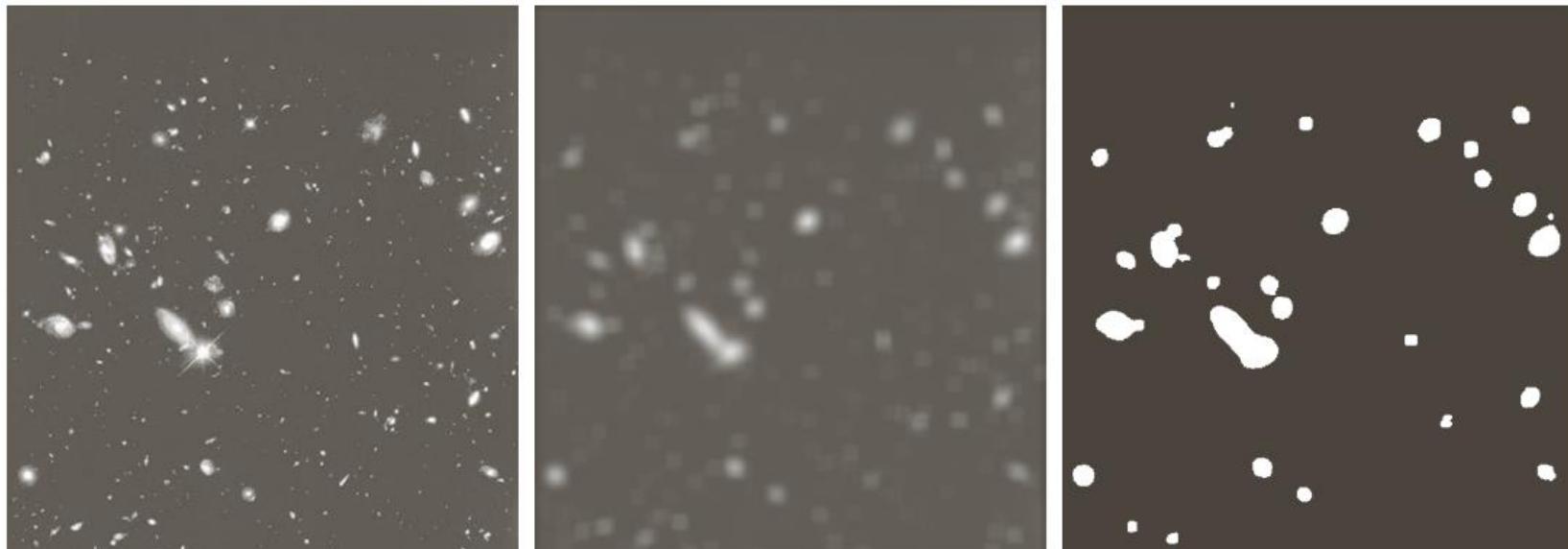


a	b
c	d

**FIGURE 2.35**  
Local averaging using neighborhood processing. The procedure is illustrated in (a) and (b) for a rectangular neighborhood. (c) The aortic angiogram discussed in Section 1.3.2. (d) The result of using Eq. (2.6-21) with  $m = n = 41$ . The images are of size  $790 \times 686$  pixels.

# Basic Image Operations

## Spatial Operations – Neighborhood operations



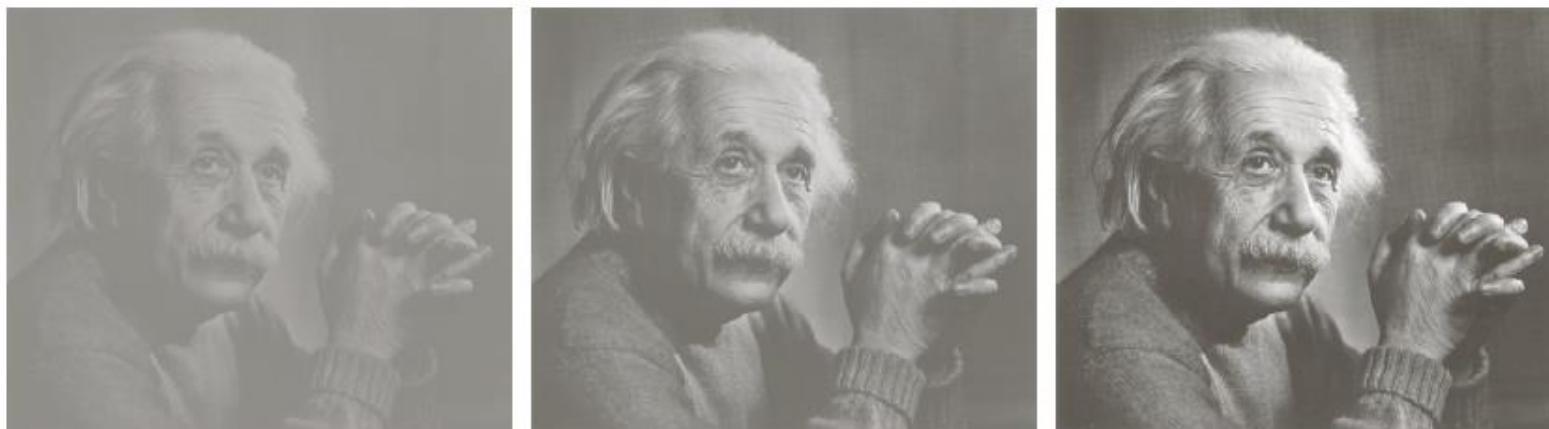
a b c

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

# Basic Image Operations

**Probabilistic methods** – used in many ways.

Ex.- Treat intensity values as random variable, then the mean and variance can be useful measure of image characteristics.



a b c

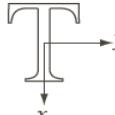
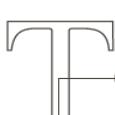
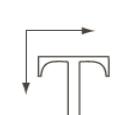
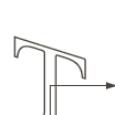
**FIGURE 2.41**  
Images exhibiting  
(a) low contrast,  
(b) medium  
contrast, and  
(c) high contrast.

# Basic Image Operations

## Spatial Operations – Geometric transformations

**TABLE 2.2**

Affine transformations based on Eq. (2.6–23).

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v \cos \theta - w \sin \theta$ $y = v \cos \theta + w \sin \theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

# Basic Image Operations

Image Resizing:

Down-sampling and  
Up-sampling

Interpolation methods:

1. Nearest neighbour technique
2. Bilinear technique
3. Bicubic technique



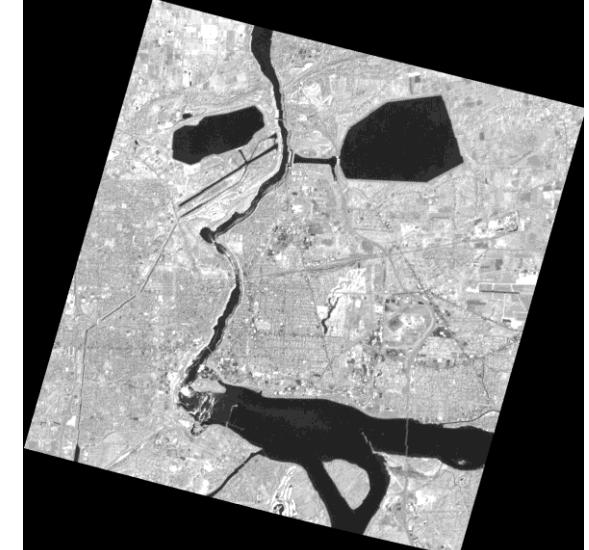
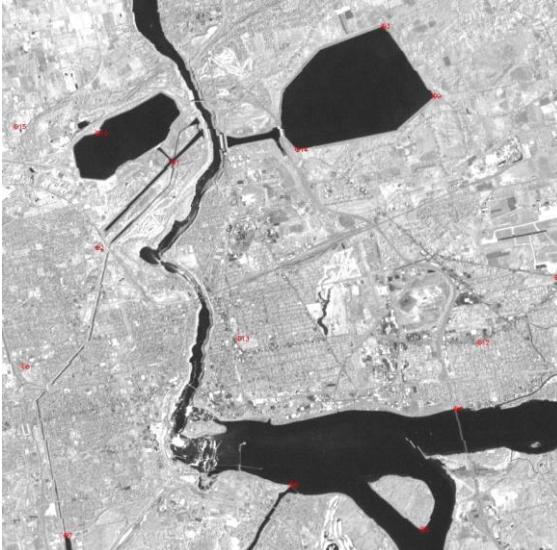
128 → 1024

64 → 1024

# Basic Image Operations

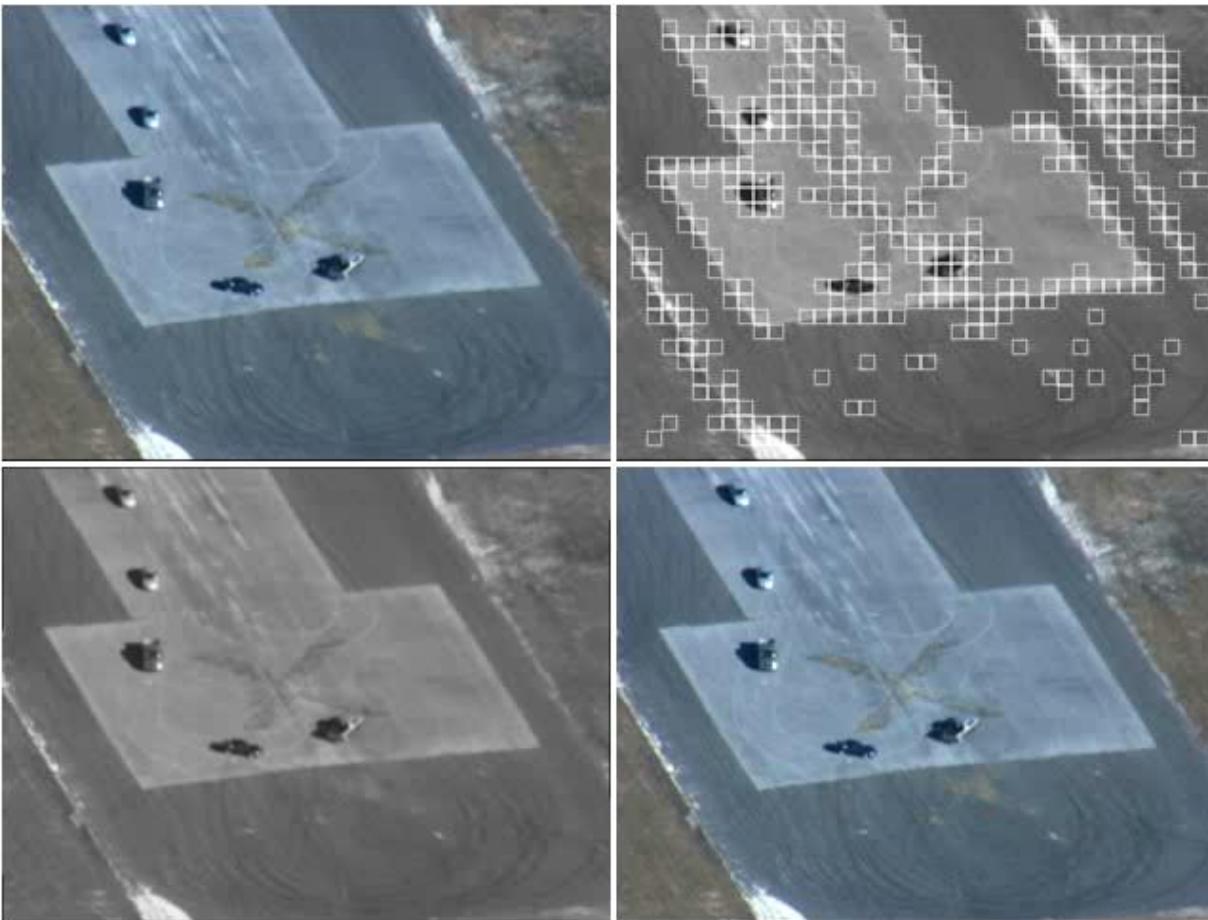
## Application – Image Registration

Image Registration is a process of estimating optimal transformation between two images which need to be aligned



# Basic Image Operations

## Application– Image Registration



# Summary

We have looked at:

- Pixel neighbourhood and connectivity
- Distance measures
- Connected component labeling
- Basic Image Operations

Next time we start to look at techniques for image enhancement

Last class we have looked at some basic Image Operations.

**Q.** Image subtraction is used often in industrial applications for detecting missing components in a product assembly. The approach is to store a "golden" image that corresponds to a correct assembly. This image is then subtracted from incoming images of the same product. Ideally, the differences would be zero if the new products are assembled correctly. Difference images for products with missing components would be nonzero in the regions where they differ from the golden image, **What conditions do you think have to be met in practice for this method to work?**

# Image Enhancement Methods

There are two broad categories of image enhancement techniques

- Spatial domain techniques
  - Direct manipulation of image pixels
- Frequency domain techniques
  - Manipulation of Fourier transform or wavelet transform of an image

For the moment we will concentrate on techniques that operate in the spatial domain

# Spatial domain Image Enhancement

Over the next few lectures we will look at image enhancement techniques working in the spatial domain:

- What is image enhancement?
- Different kinds of image enhancement
- Point processing
- Histogram processing
- Neighbourhood operations

# What Is Image Enhancement?

Image enhancement is the process of making images more useful

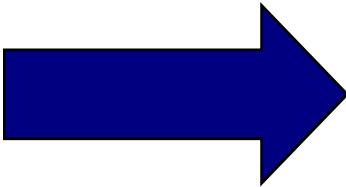
The reasons for doing this include:

- Highlighting interesting detail in images
- Removing noise from images
- Making images more visually appealing

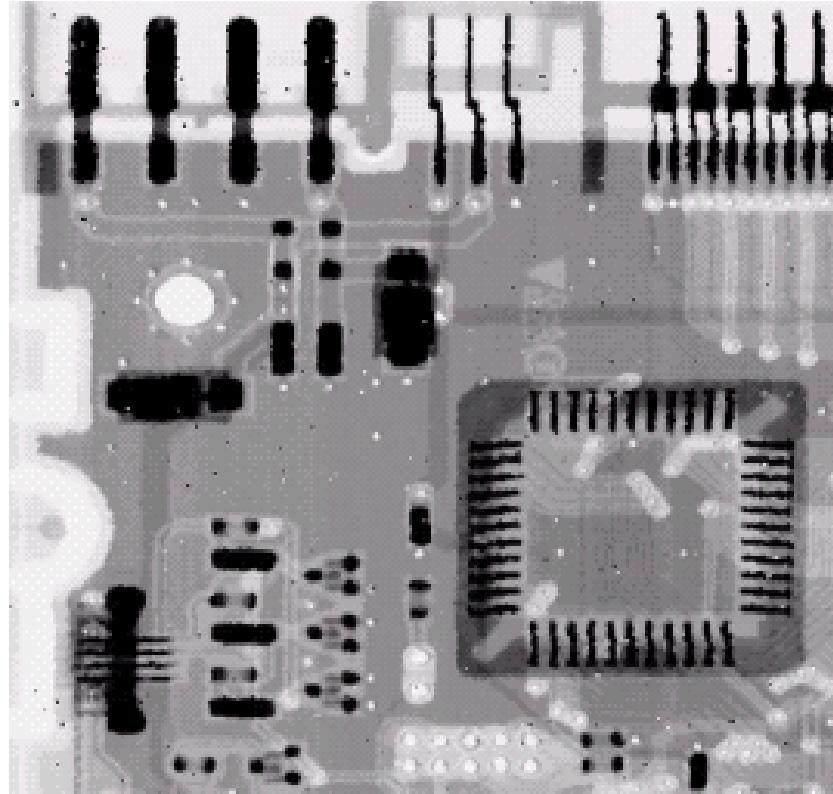
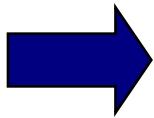
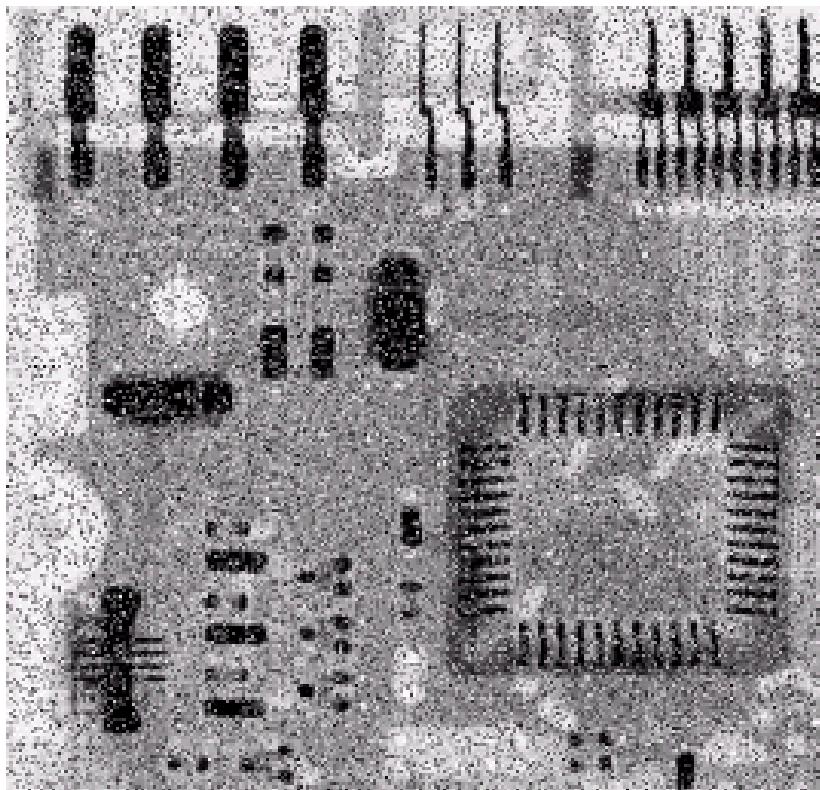
# Image Enhancement Examples



# Image Enhancement Examples (cont...)



# Image Enhancement Examples (cont...)



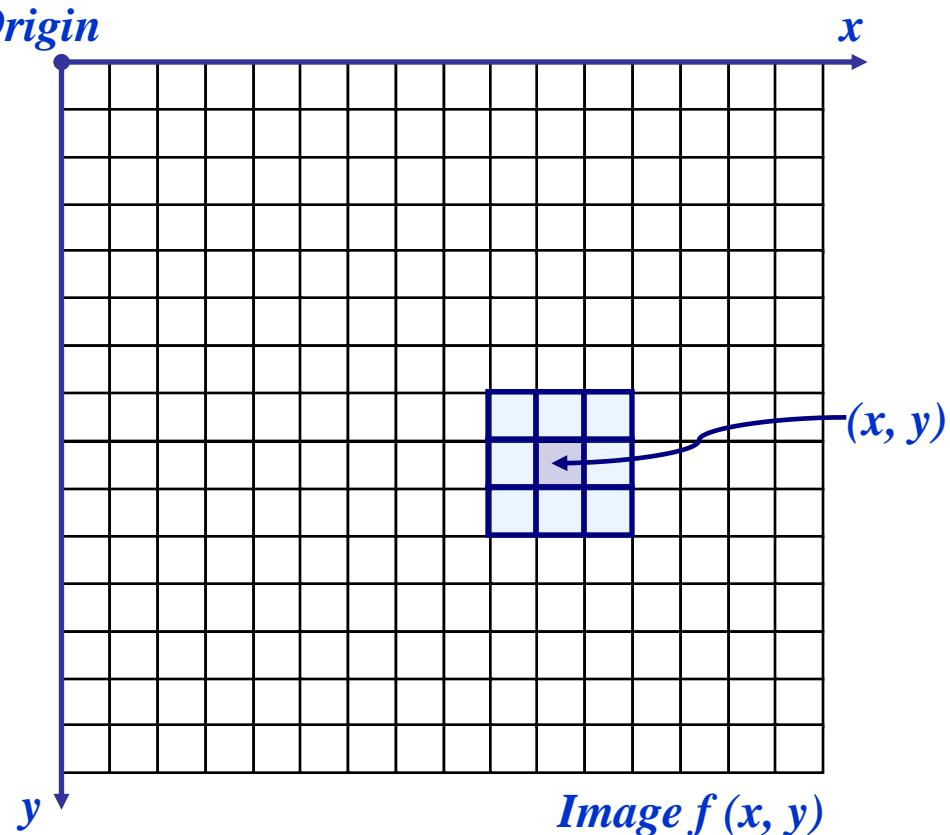
# Image Enhancement Examples (cont...)



# Basic Spatial Domain Image Enhancement

- Most spatial domain enhancement operations can be reduced to the form

- $g(x, y) = T[f(x, y)]$
- where  $f(x, y)$  is the input image,  $g(x, y)$  is the processed image and  $T$  is some operator defined over some neighbourhood of  $(x, y)$



# Point Processing

- The simplest spatial domain operations occur when the neighbourhood is simply the pixel itself
- In this case  $T$  is referred to as a *grey level transformation function* or a *point processing operation*
- Point processing operations take the form
  - $s = T(r)$
- where  $s$  refers to the processed image pixel value and  $r$  refers to the original image pixel value

# A Note About Grey Levels

So far when we have spoken about image grey level values we have said they are in the range [0, 255]

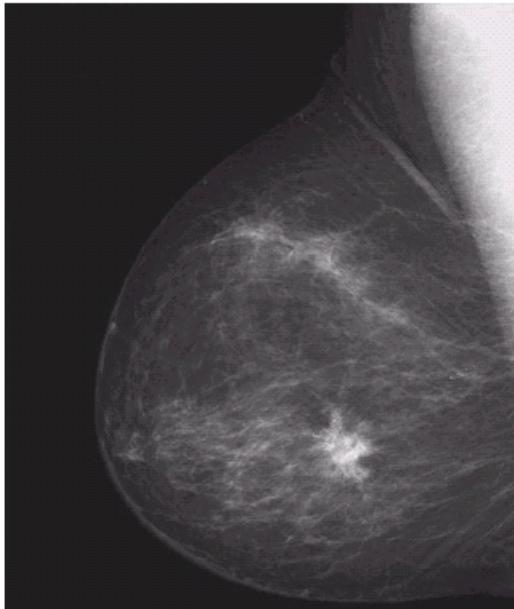
- Where 0 is black and 255 is white
- The range [0,255] stems from display technologies

We need not be restricted to this range. For many of the image processing operations in this lecture grey levels are assumed to be given in the range [0.0, 1.0]

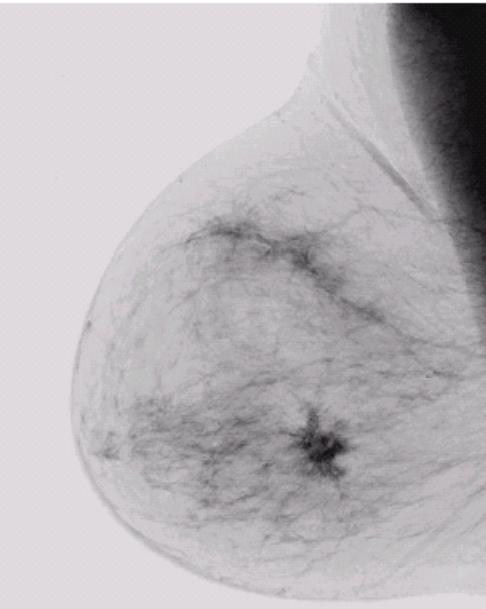
# Point Processing Example: Negative Images

- Negative images are useful for enhancing white or grey detail embedded in dark regions of an image
  - Related to human visual perception and adaptation in different brightness range

Original  
Image



$$s = 1.0 - r$$



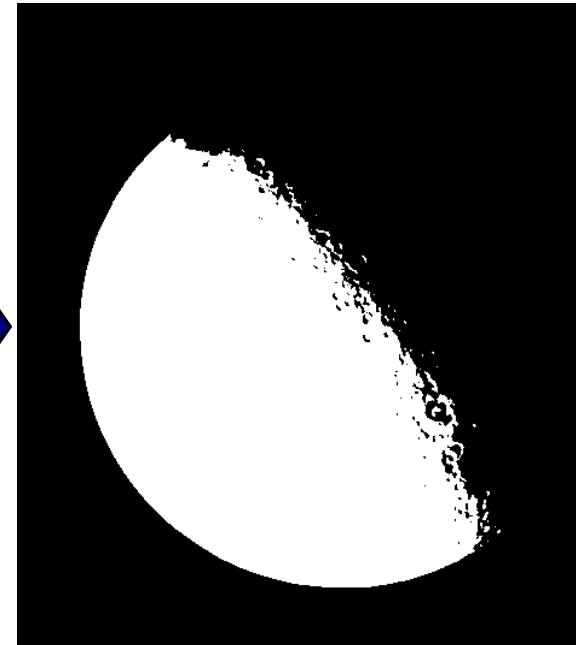
Negative  
Image

# Point Processing Example: Thresholding

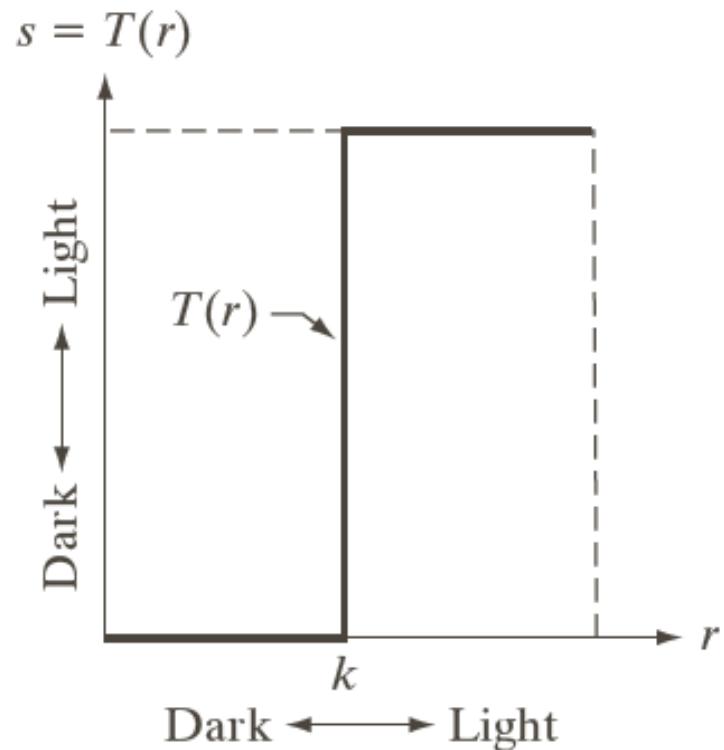
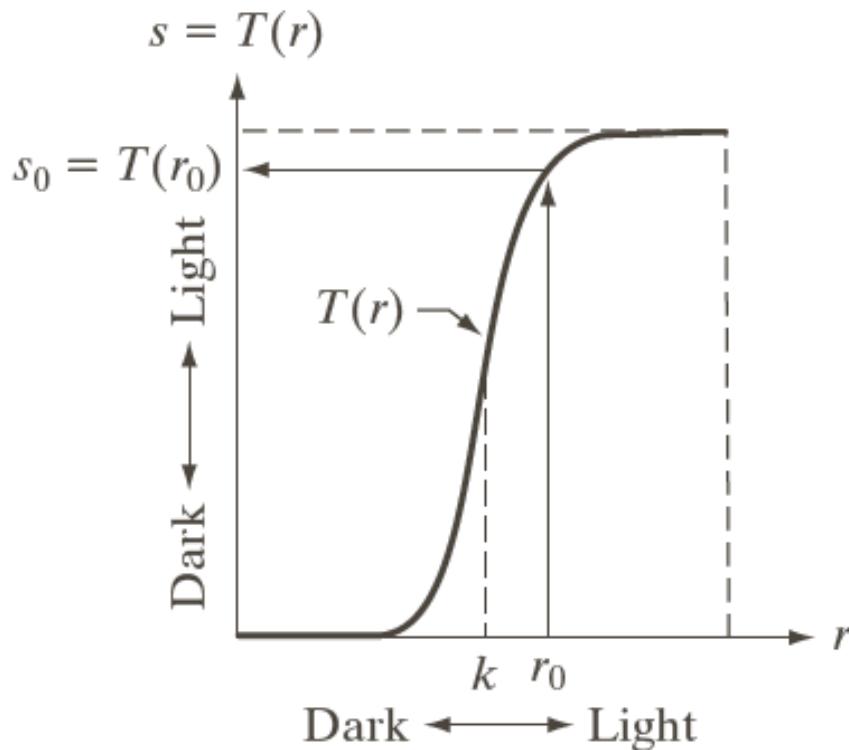
- Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background



$$s = \begin{cases} 1.0 & r > \text{threshold} \\ 0.0 & r \leq \text{threshold} \end{cases}$$

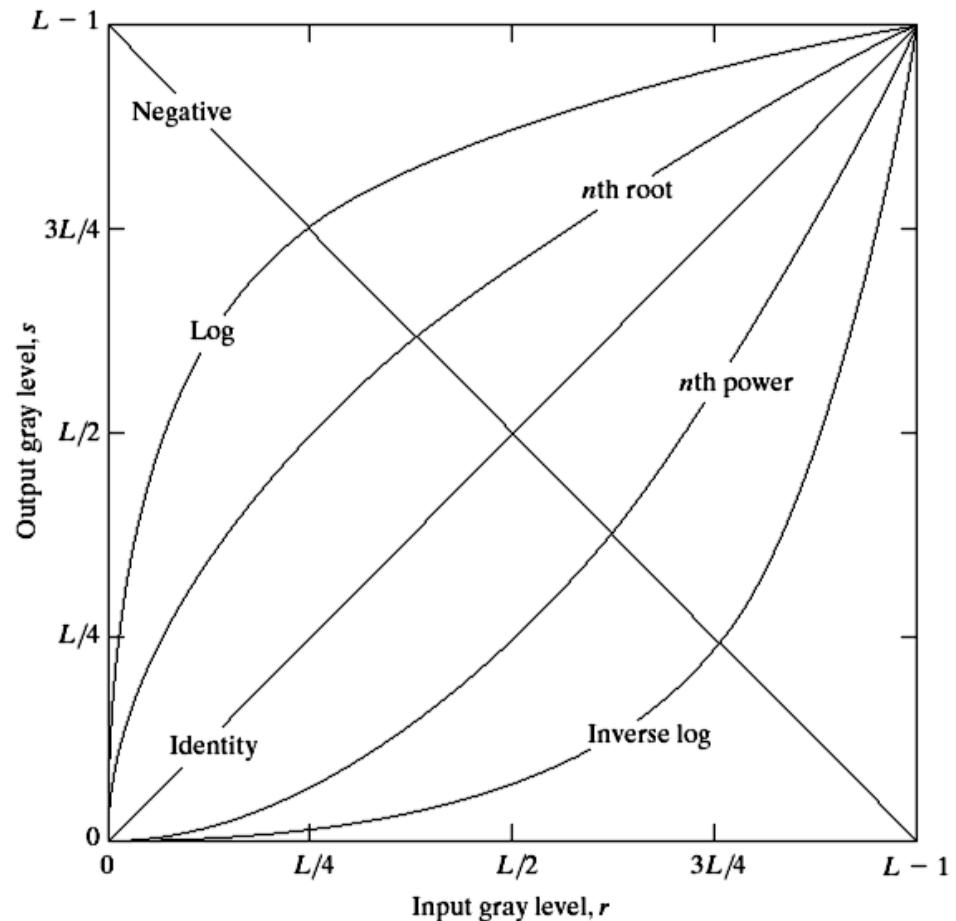


# Intensity Transformations



# Basic Grey Level Transformations

- There are many different kinds of grey level transformations
- Three of the most common are shown here
  - Linear
    - Negative/Identity
  - Logarithmic
    - Log/Inverse log
  - Power law
    - $n^{\text{th}}$  power/ $n^{\text{th}}$  root



# Logarithmic Transformations

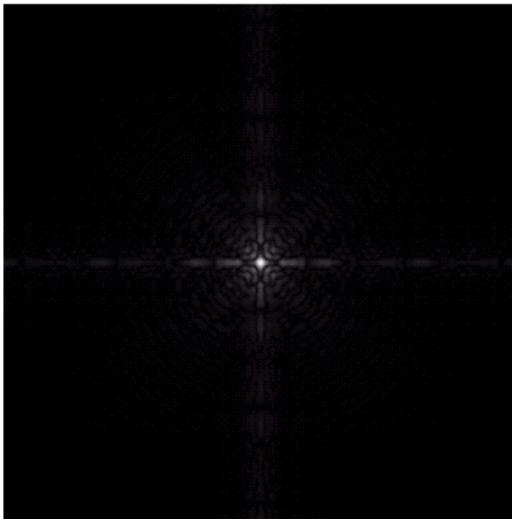
- The general form of the log transformation is

$$\bullet s = c * \log(1 + r)$$

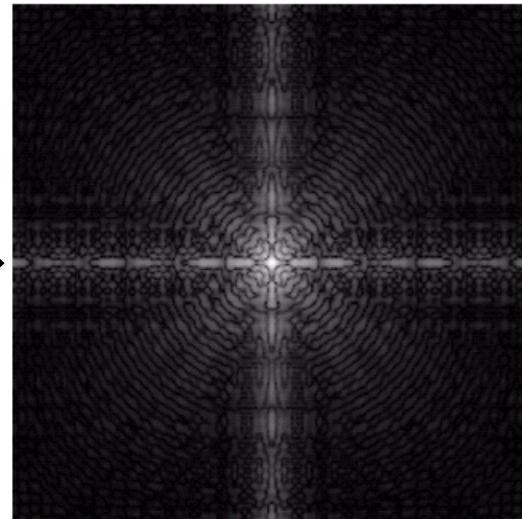
- We usually set  $c$  to 1 and it is assumed that  $r \geq 0$
- The log transformation maps a narrow range of low input grey level values into a wider range of output values
- The inverse log transformation performs the opposite transformation

# Logarithmic Transformations (cont...)

- Log functions are particularly useful when the input grey level values may have an extremely large range of values
- In the following example the Fourier transform of an image is put through a log transform to reveal more detail



$$s = \log(1 + r)$$

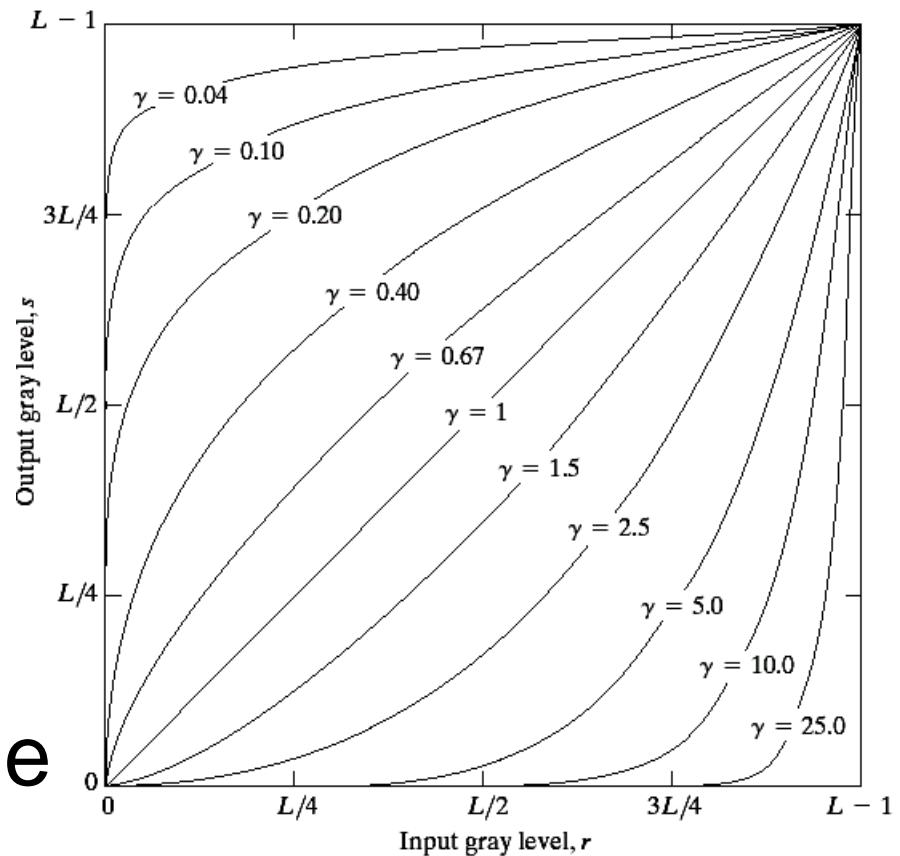


# Power Law Transformations

- Power law transformations have the following form

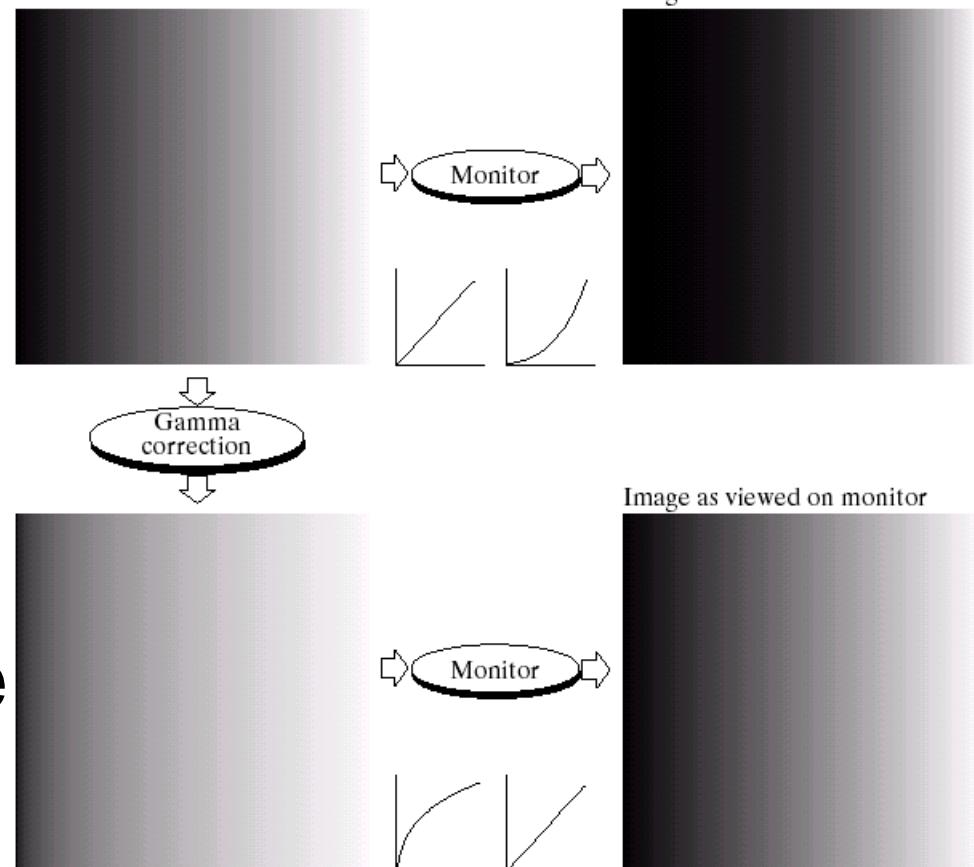
$$s = c * r^\gamma$$

- Map a narrow range of dark input values into a wider range of output values or vice versa
- Varying  $\gamma$  gives a whole family of curves



# Gamma Correction

- Some of you might be familiar with gamma correction of computer monitors
- Problem is that display devices do not respond linearly to different intensities
  - Can be corrected by preprocessing the image appropriately

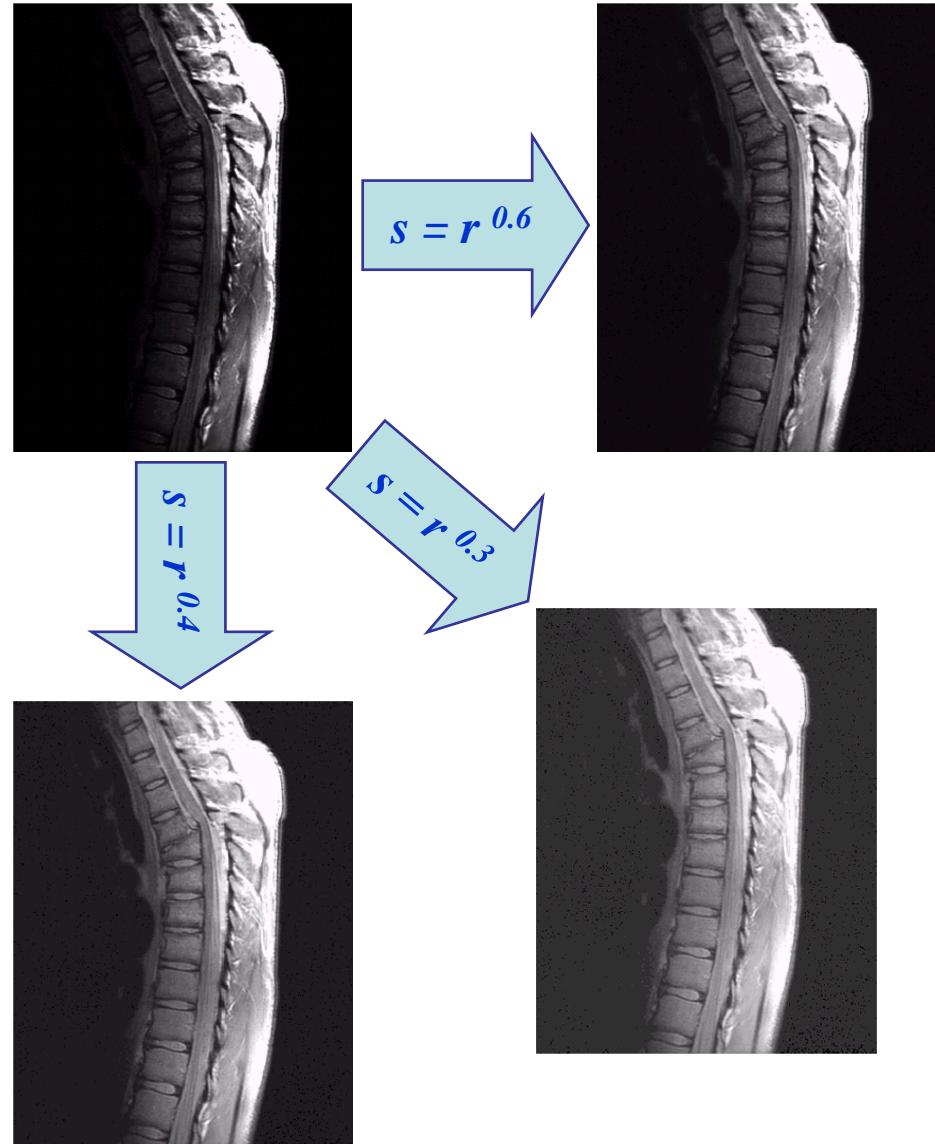


# Power Law Example



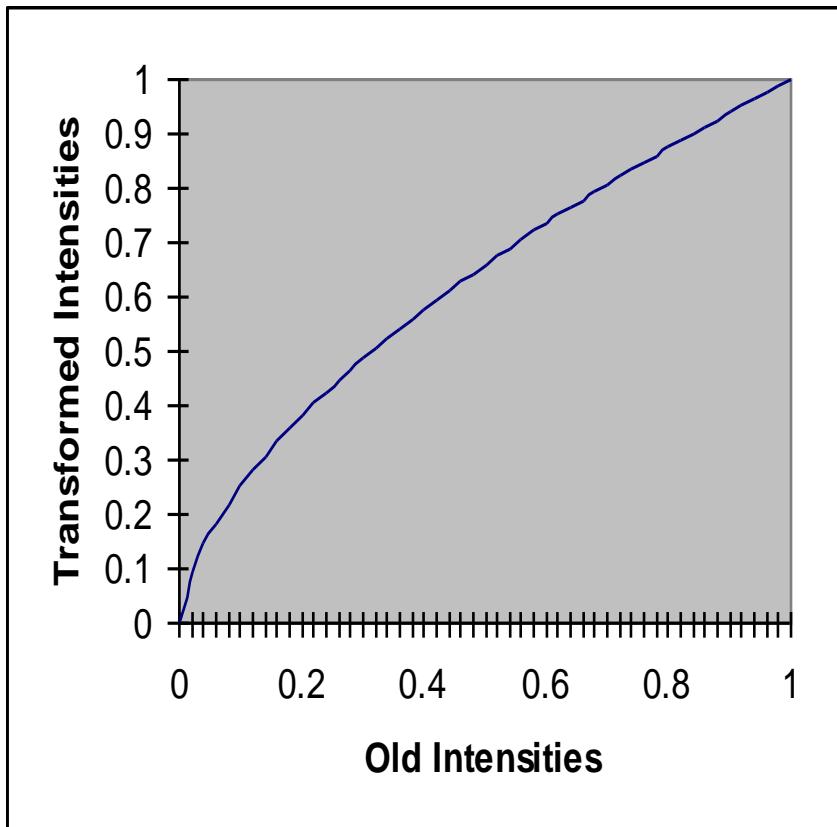
# Power Law Example (cont...)

- The images to the right show a magnetic resonance (MR) image of a fractured human spine
- Different curves highlight different detail



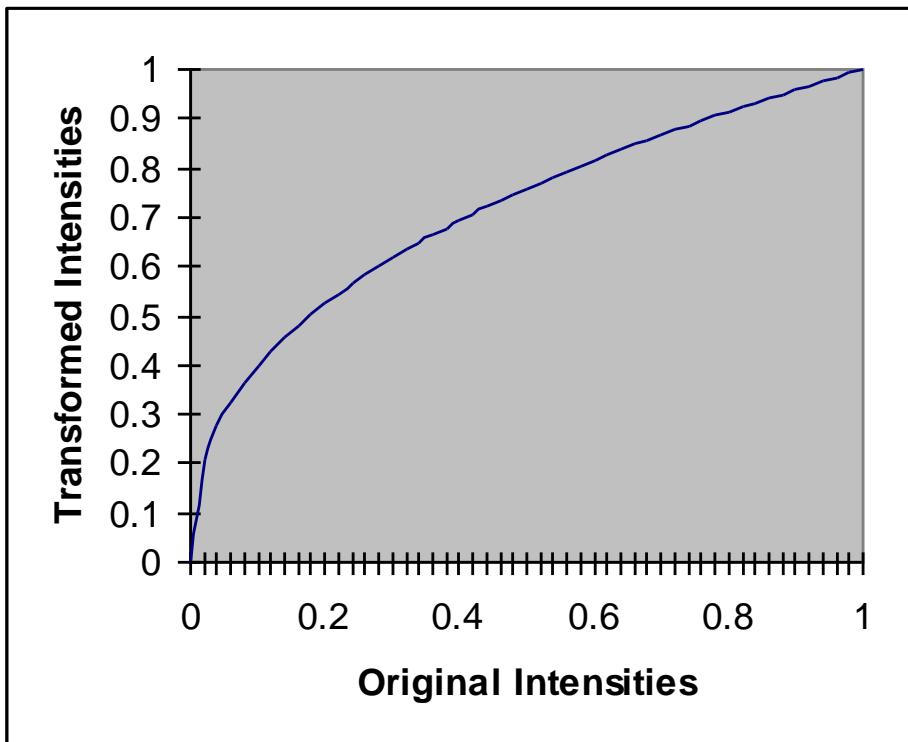
# Power Law Example (cont...)

$$\gamma = 0.6$$



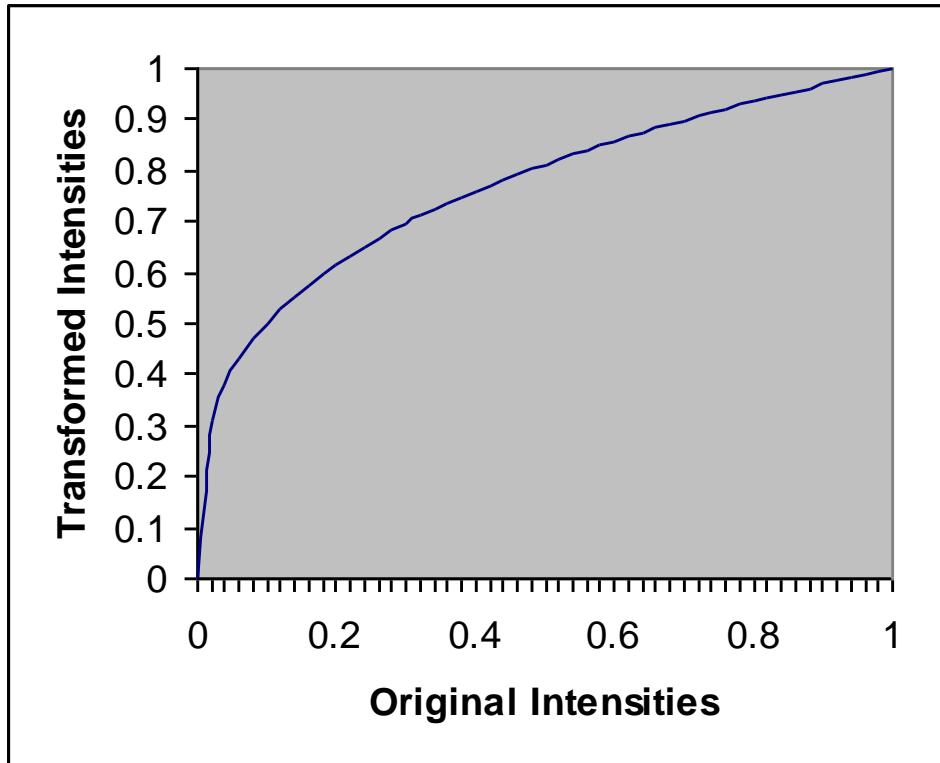
# Power Law Example (cont...)

$$\gamma = 0.4$$



# Power Law Example (cont...)

$$\gamma = 0.3$$

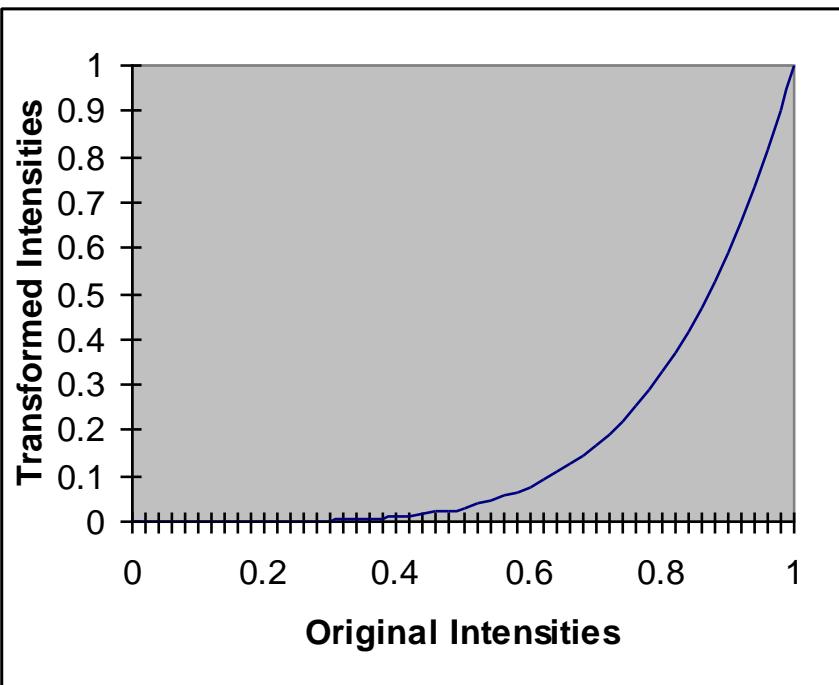


# Power Law Example



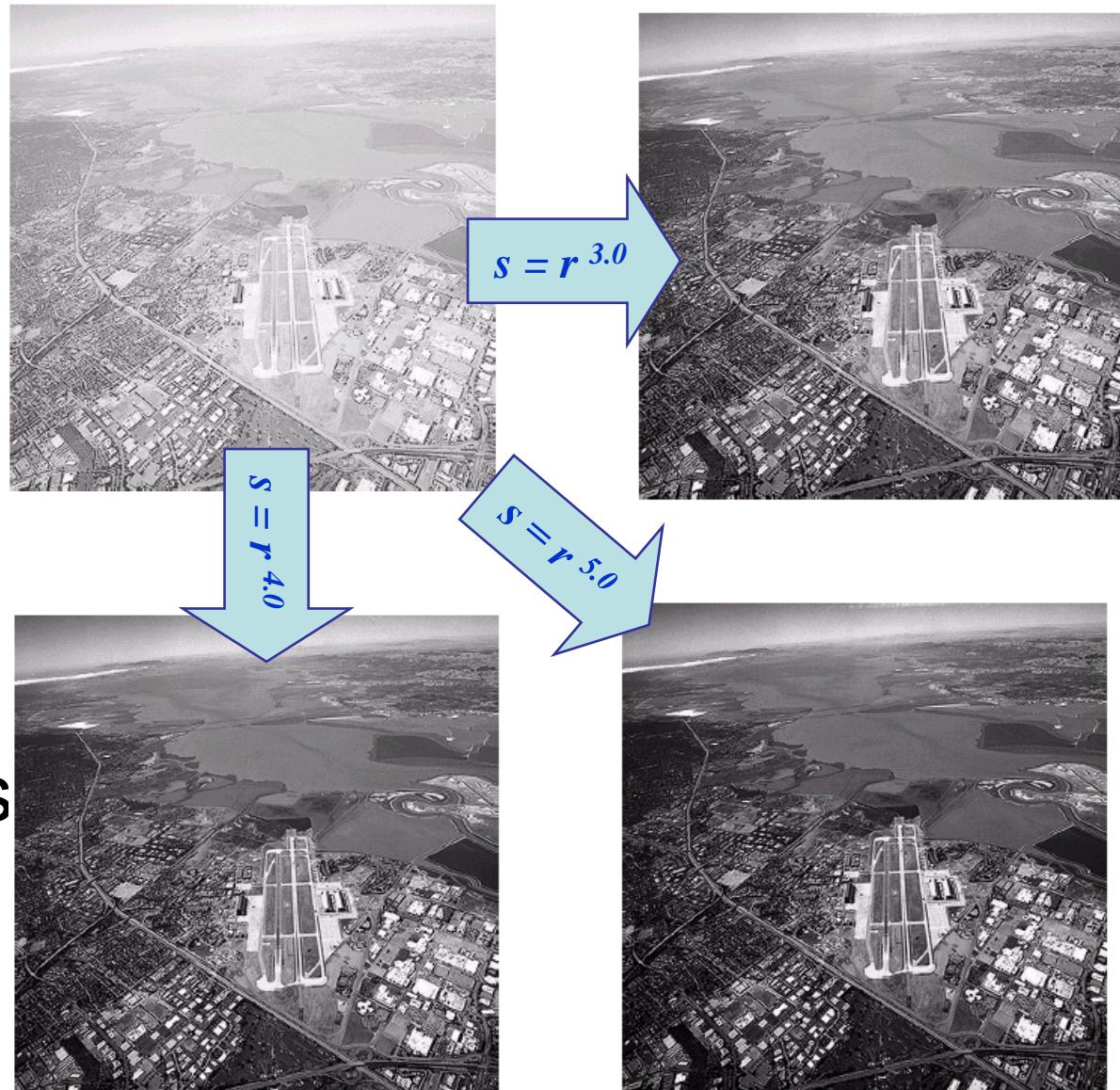
# Power Law Example (cont...)

$$\gamma = 5.0$$



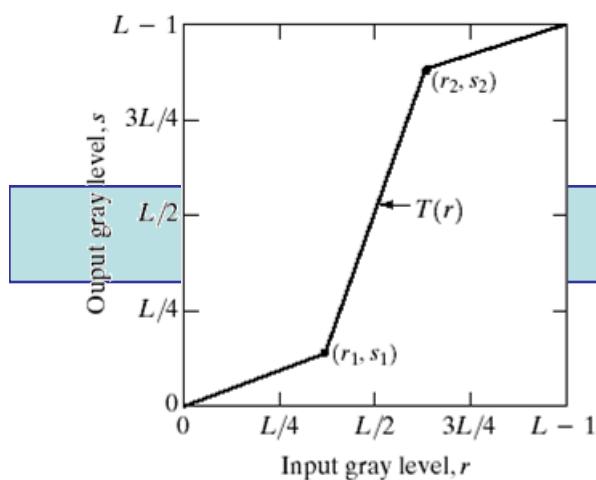
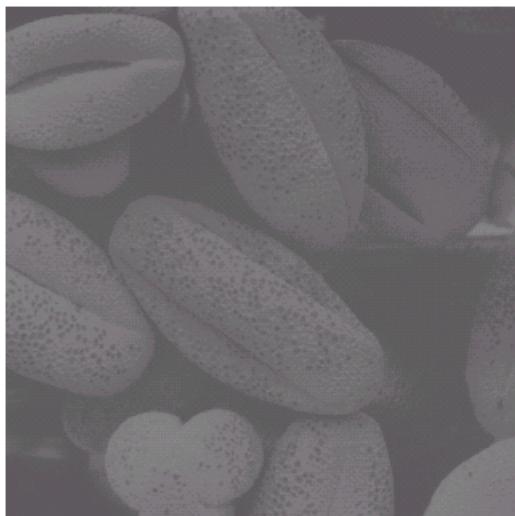
# Power Law Transformations (cont...)

- An aerial photo of a runway is shown
- This time power law transforms are used to darken the image
- Different curves highlight different detail



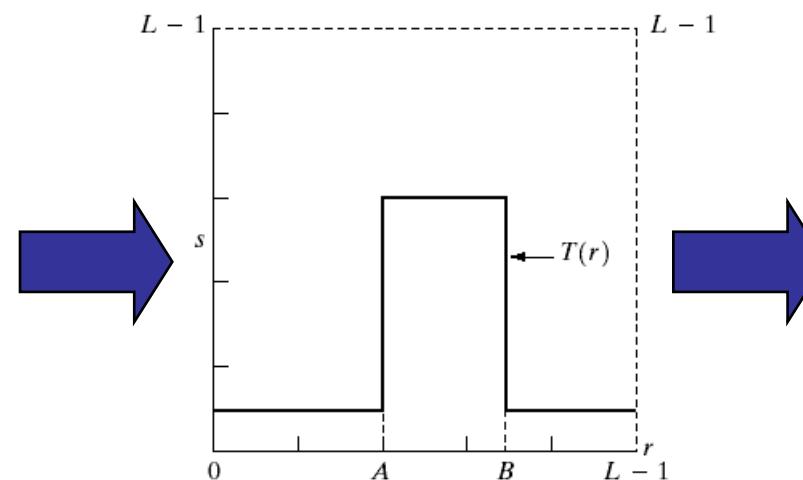
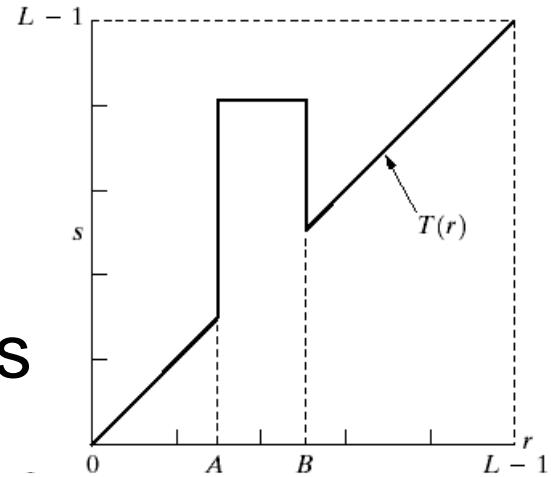
# Piecewise Linear Transformation Functions

- Rather than using a well defined mathematical function we can use arbitrary user-defined transforms
- The images below show a contrast stretching linear transform to add contrast to a poor quality image



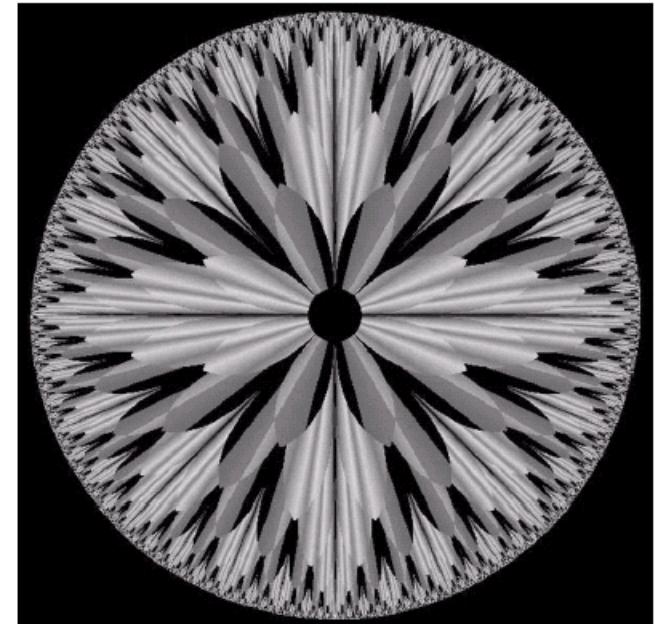
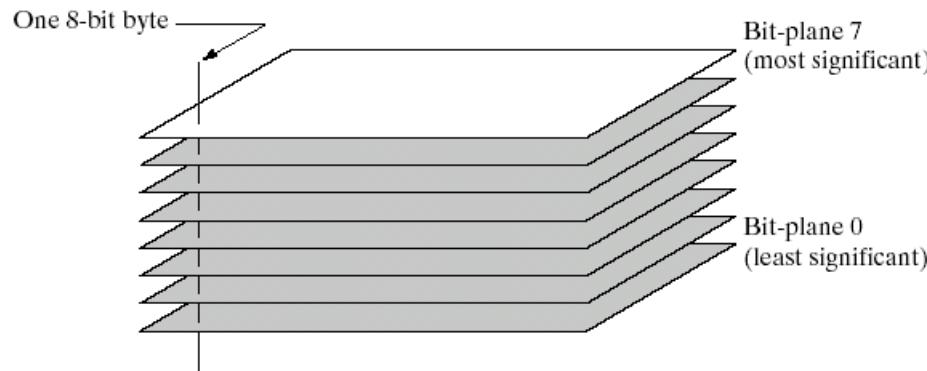
# Gray Level Slicing

- Highlights a specific range of grey levels
  - Similar to thresholding
  - Other levels can be suppressed or maintained
  - Useful for highlighting features in an image

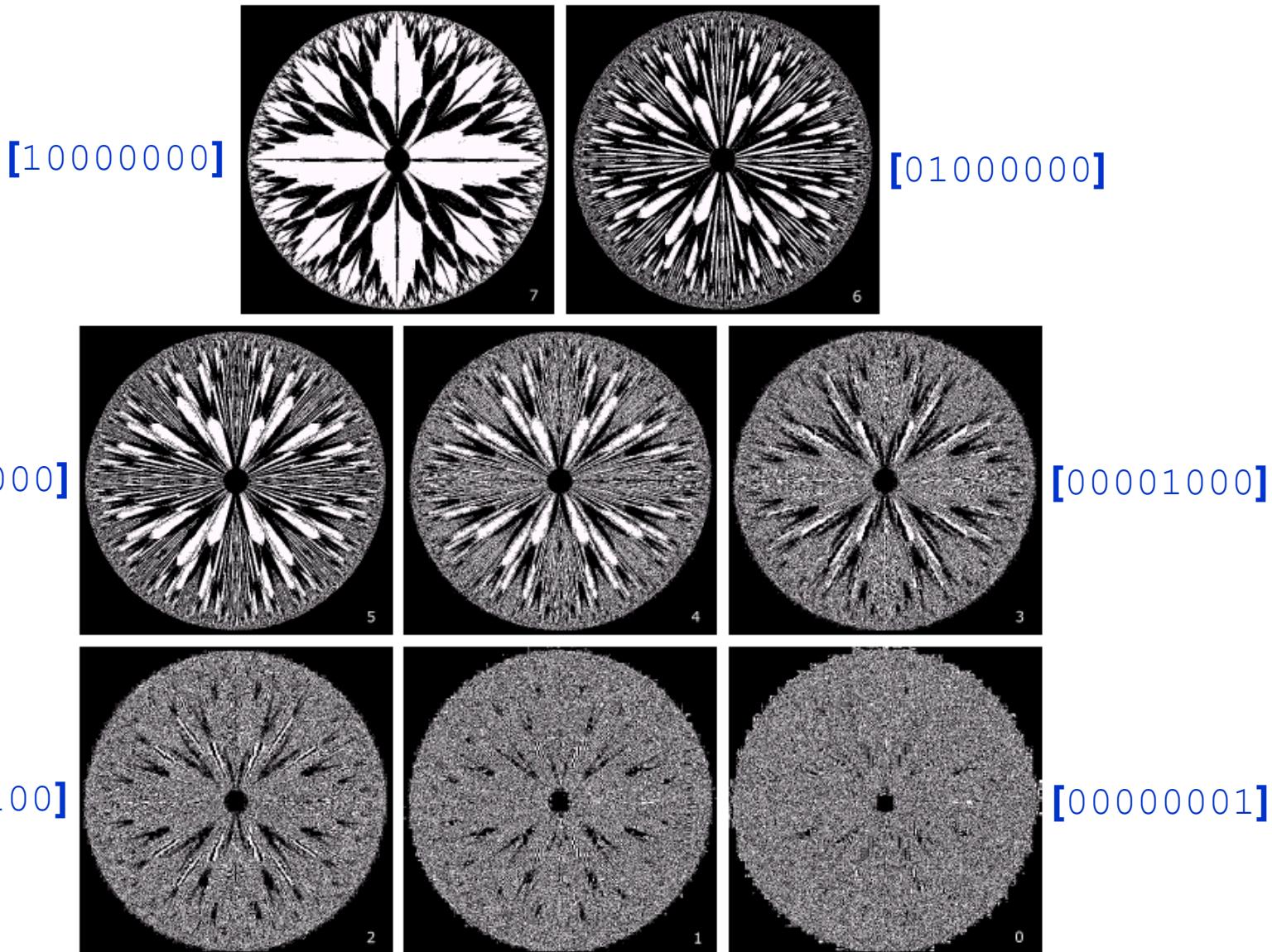


# Bit Plane Slicing

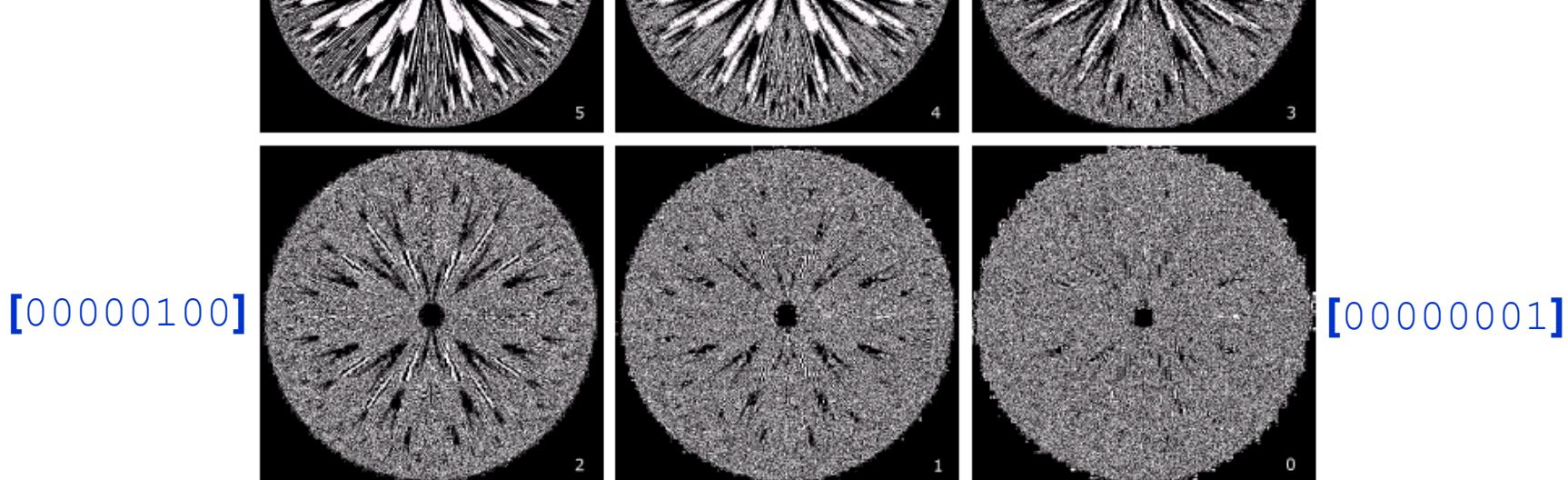
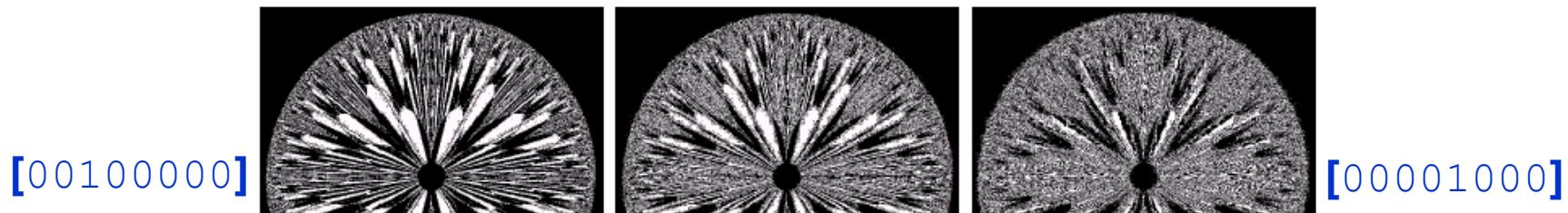
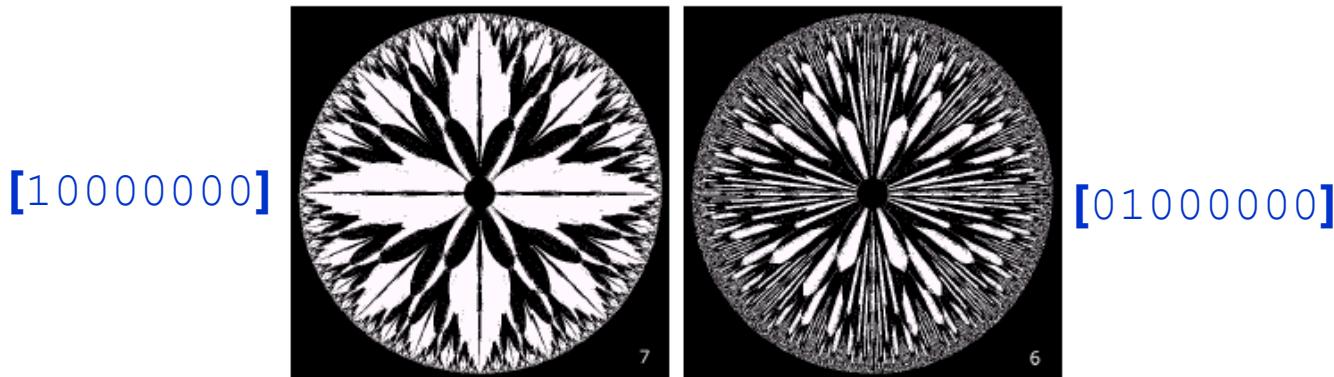
- Often by isolating particular bits of the pixel values in an image we can highlight interesting aspects of that image
  - Higher-order bits usually contain most of the significant visual information
  - Lower-order bits contain subtle details



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



a b c  
d e f  
g h i

**FIGURE 3.14** (a) An 8-bit gray-scale image of size  $500 \times 1192$  pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

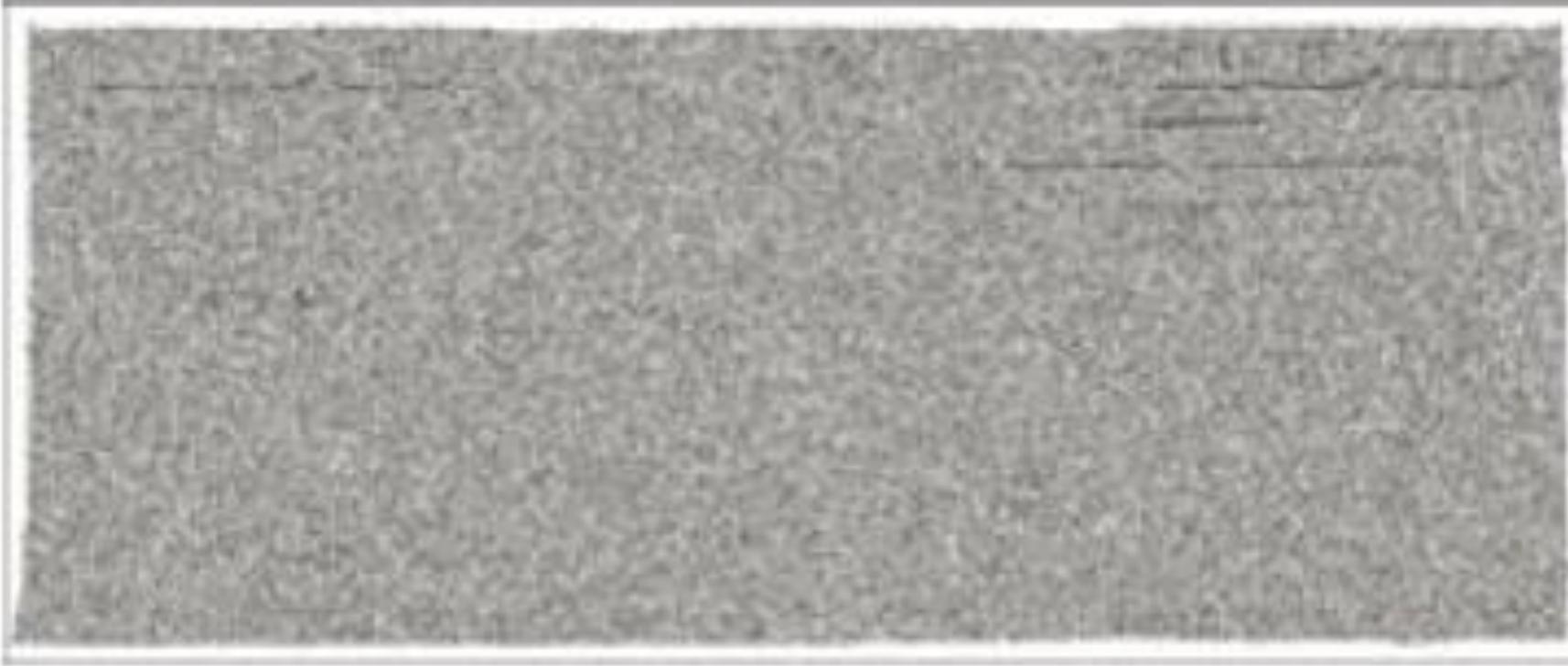
# Bit Plane Slicing (cont...)



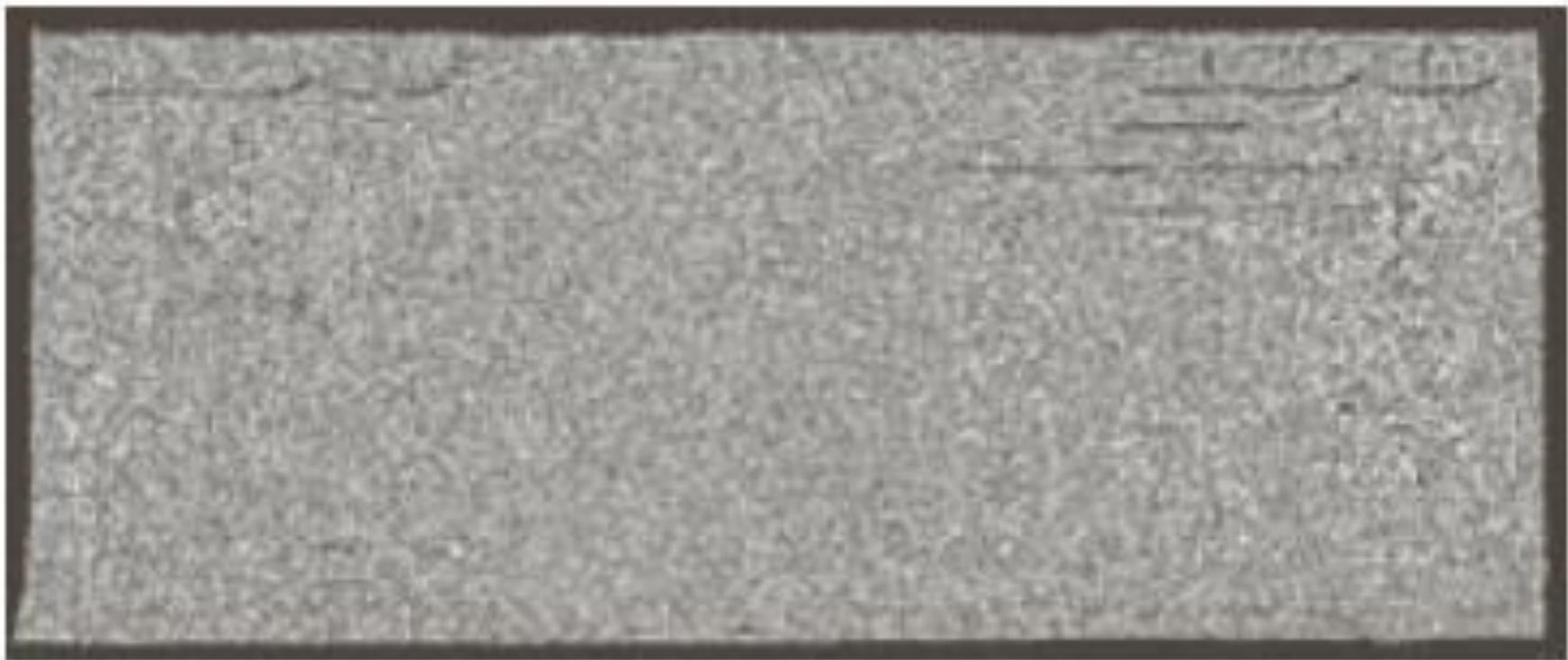
# Bit Plane Slicing (cont...)



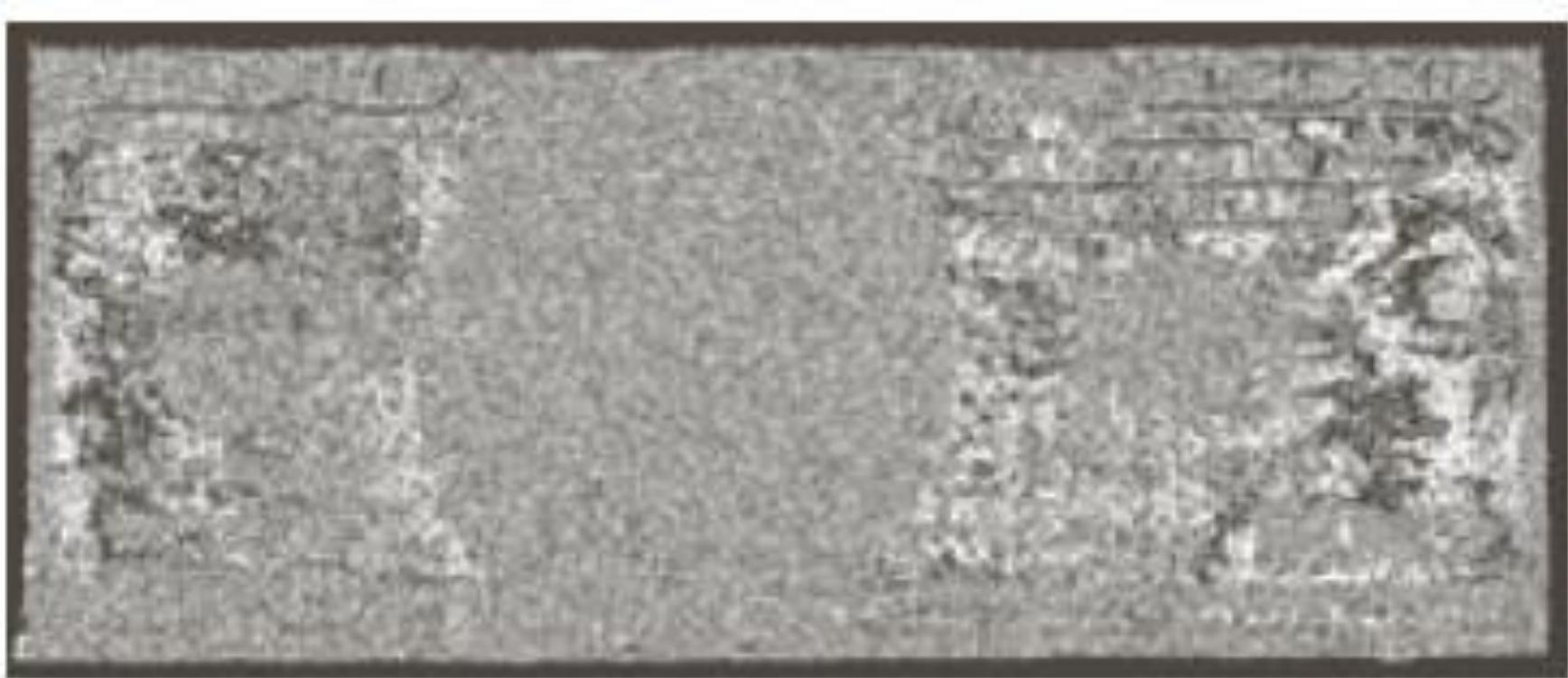
# Bit Plane Slicing (cont...)



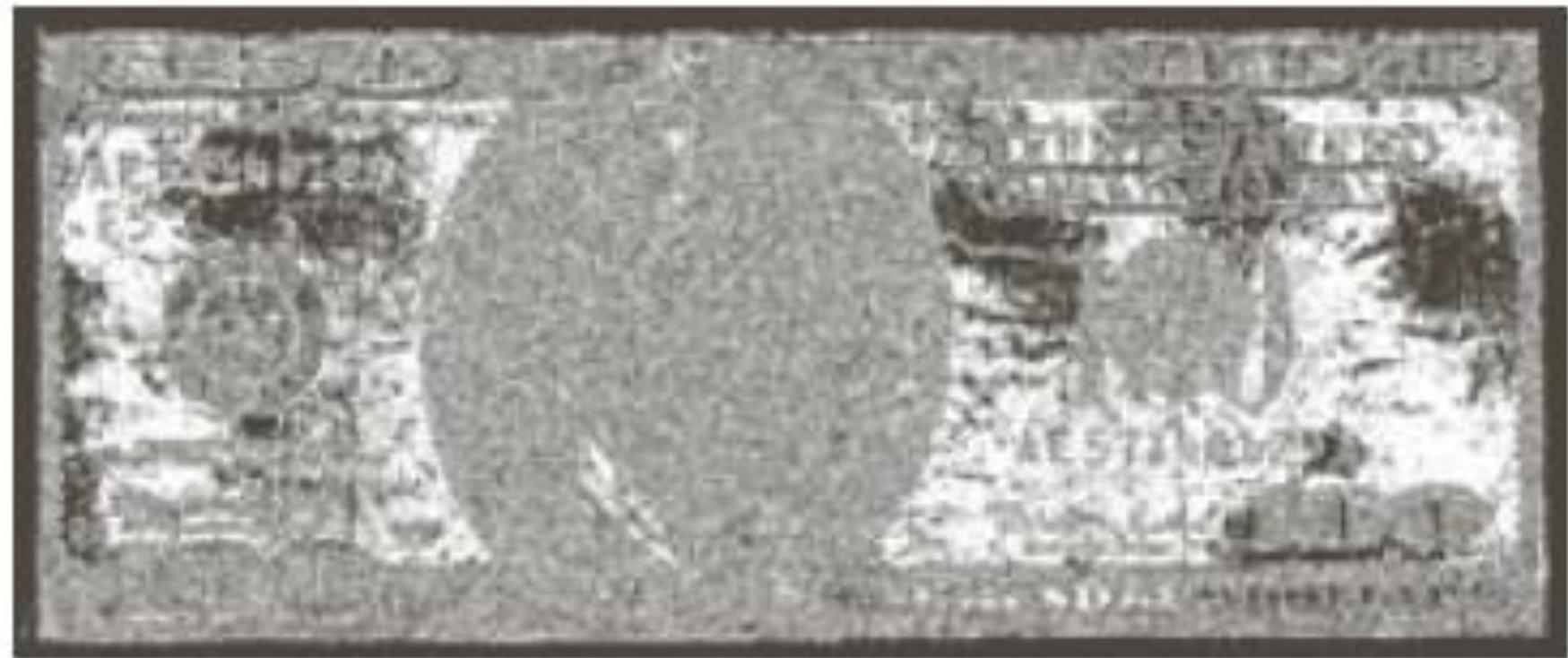
# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



Reconstructed image  
using only bit planes 8  
and 7



Reconstructed image  
using only bit planes 8, 7  
and 6



Reconstructed image  
using only bit planes 7, 6  
and 5

# Summary of Point Processing Enhancement

- We have looked at different kinds of point processing image enhancement
  - Negative images
  - Thresholding
  - Logarithmic transformation
  - Power law transforms
  - Grey level slicing
  - Bit plane slicing

Next lecture we will see histogram based processing techniques

# Image & Video Processing

Image Enhancement  
(Histogram Processing)

# Contents

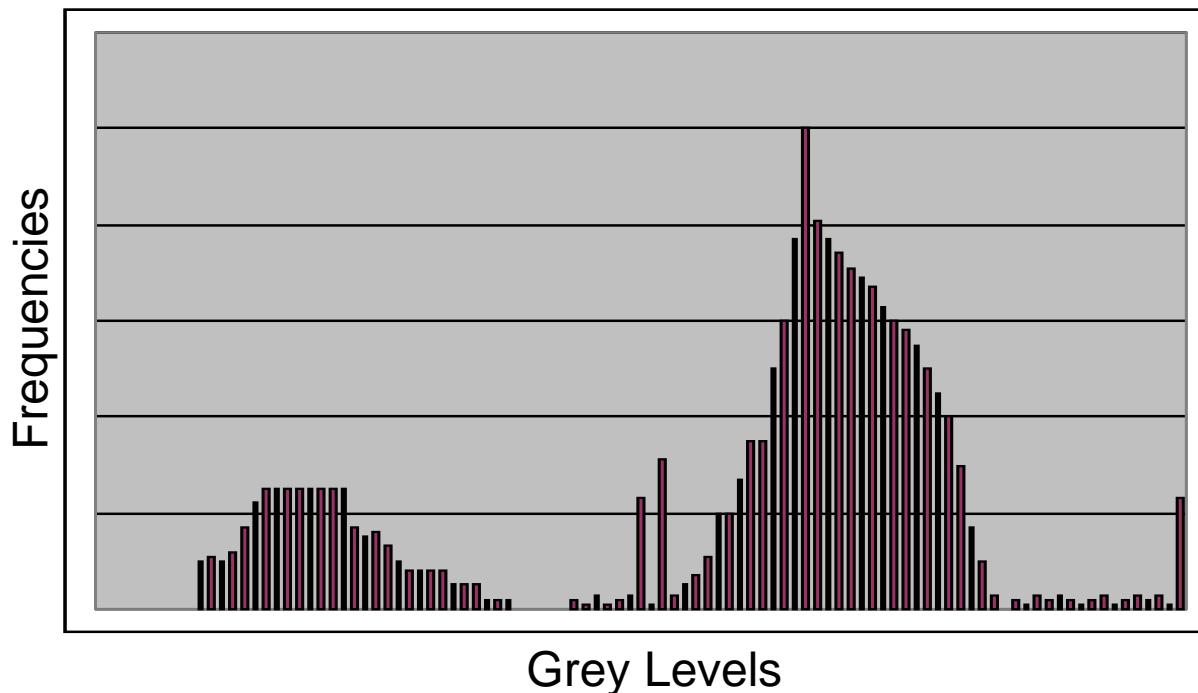
we are looking at image enhancement techniques working in the spatial domain:

- What is image enhancement?
- Different kinds of image enhancement
- Point processing
- **Histogram processing**
- Neighbourhood operations

# Image Histograms

The histogram of an image shows us the distribution of grey levels in the image

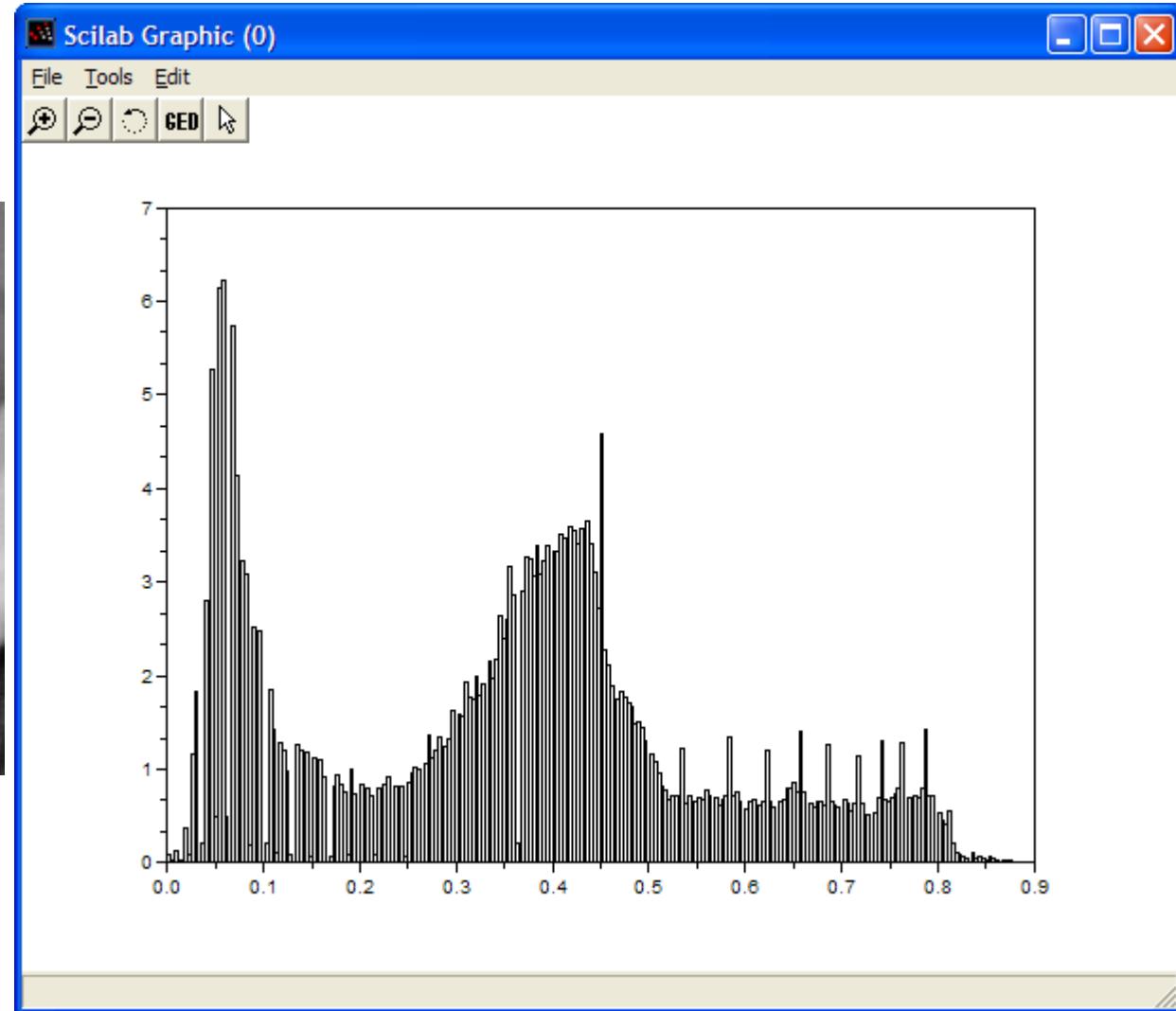
Massively useful in image processing,  
especially in segmentation



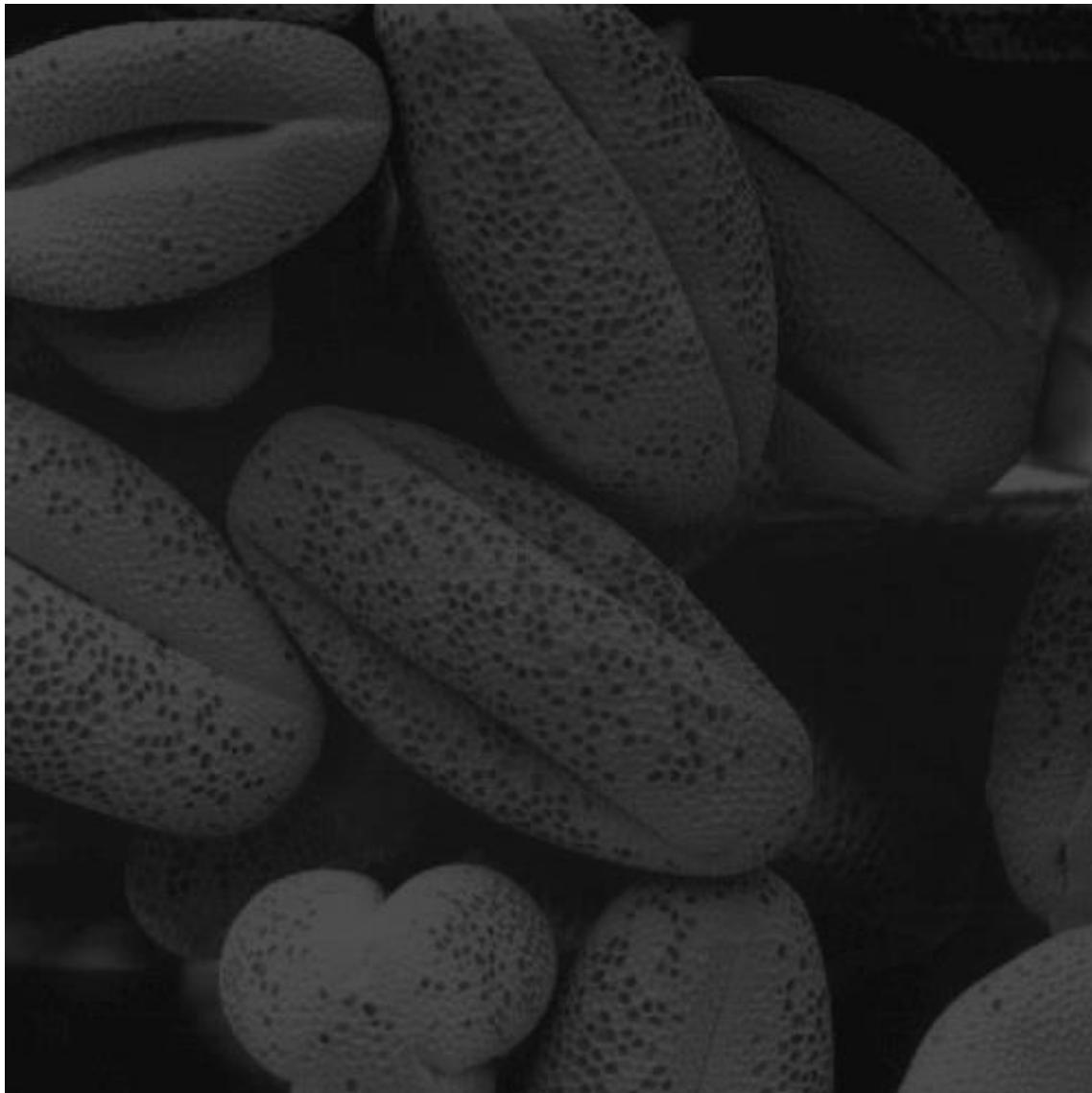
# Histogram Examples



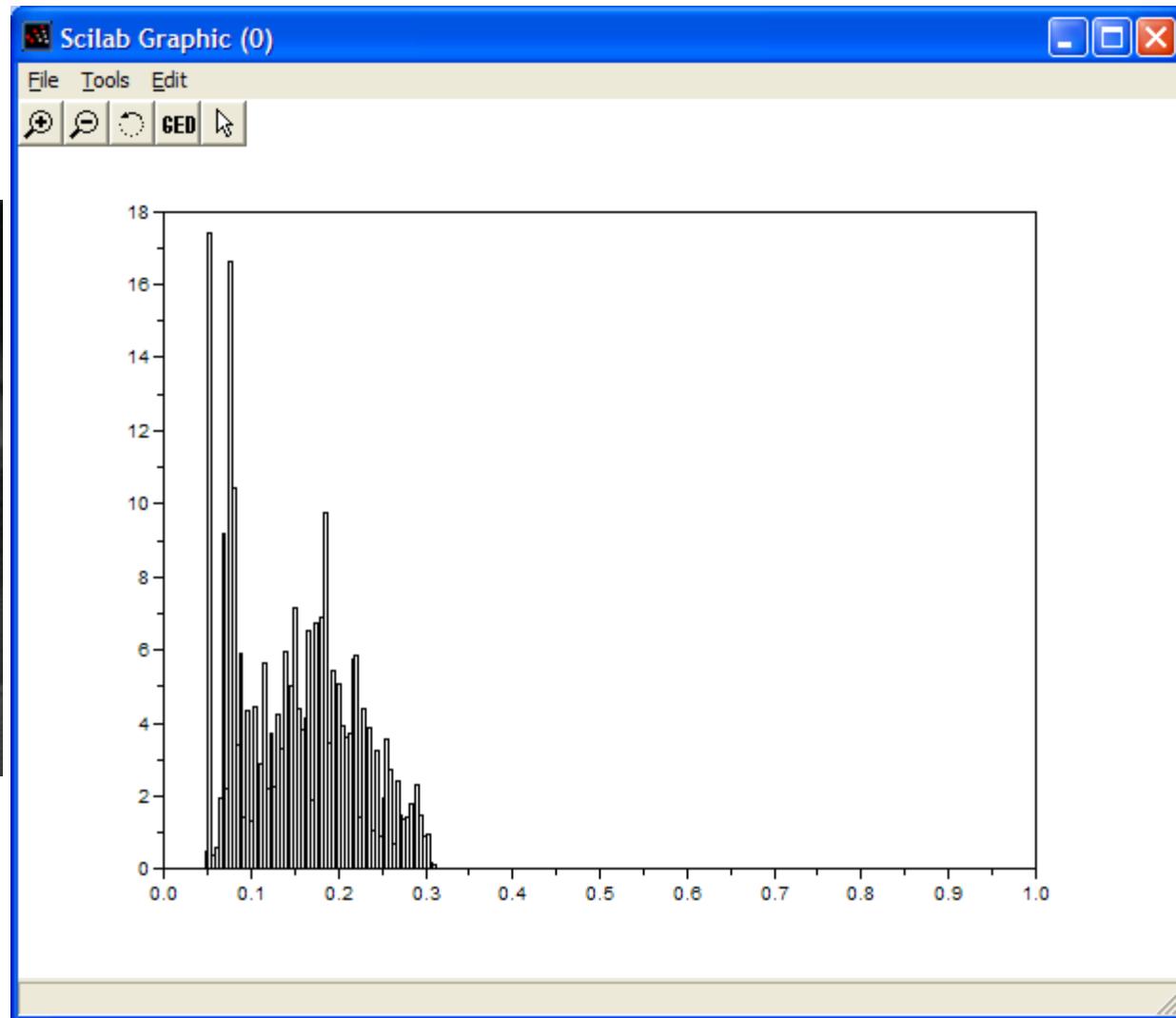
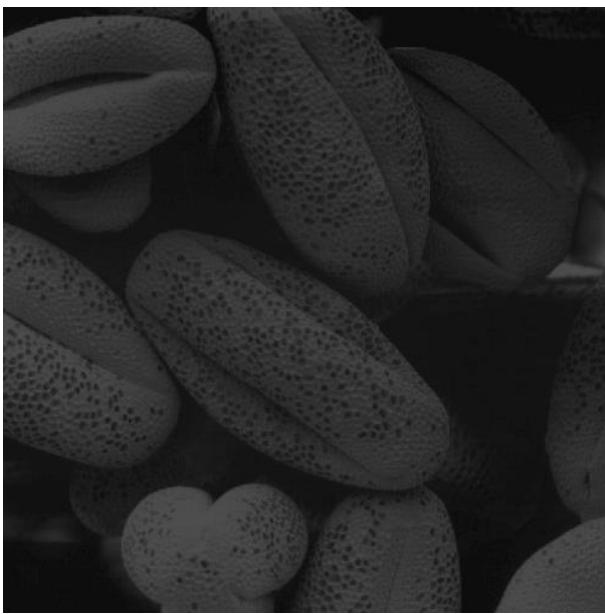
# Histogram Examples (cont...)



# Histogram Examples (cont...)



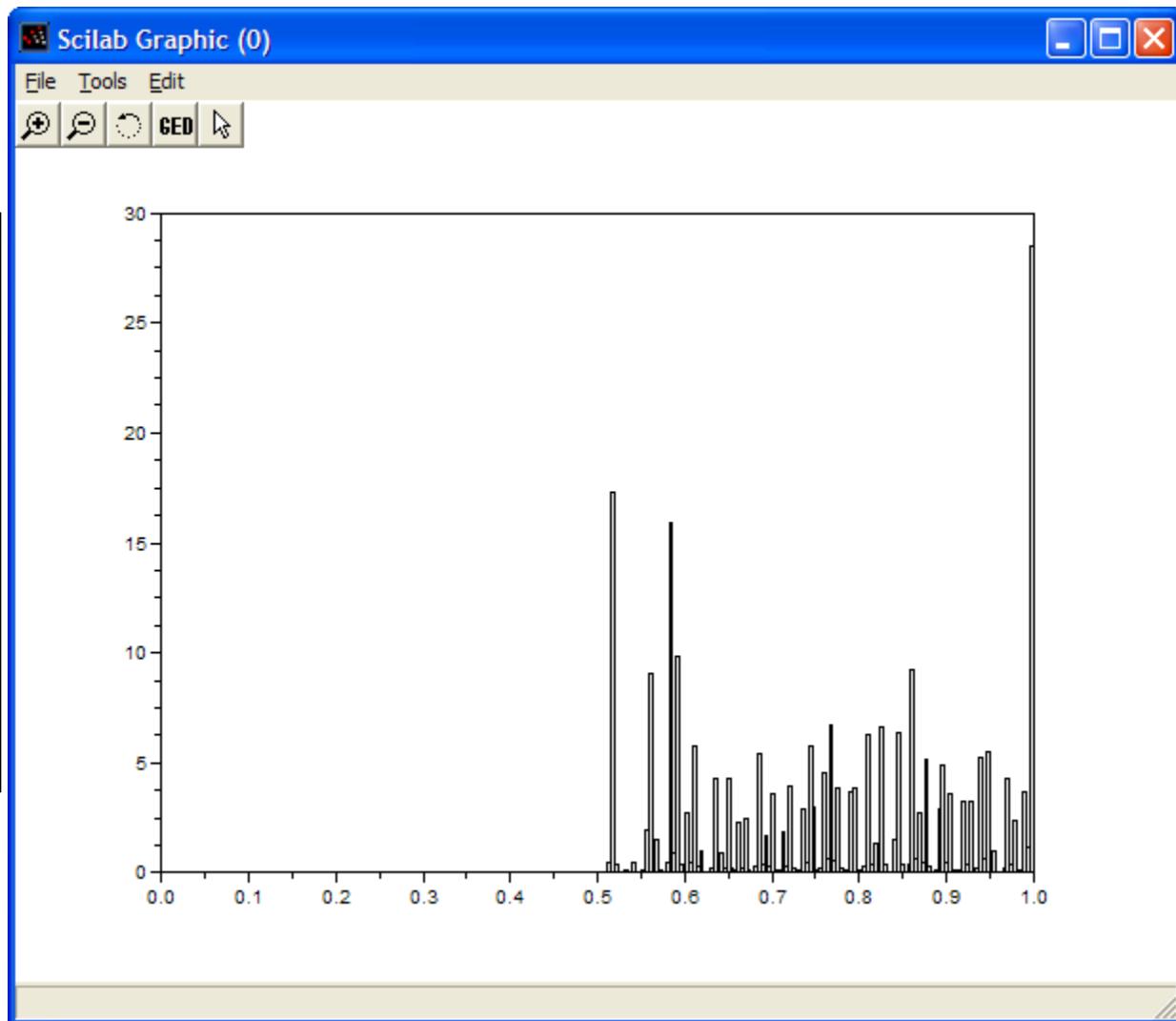
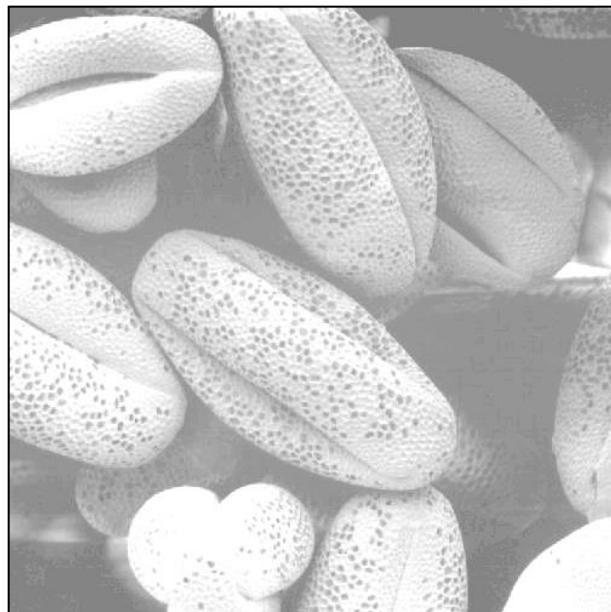
# Histogram Examples (cont...)



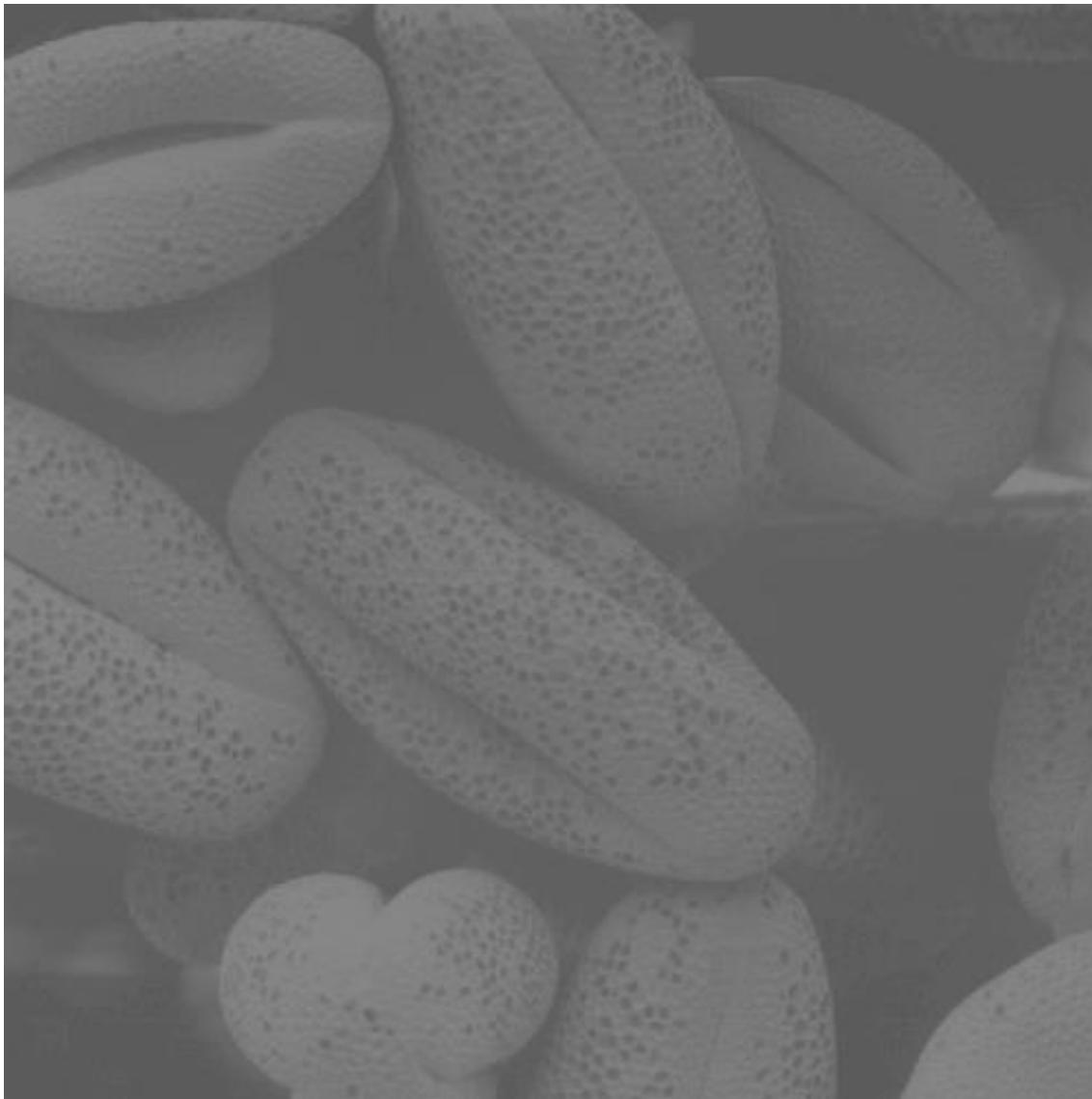
# Histogram Examples (cont...)



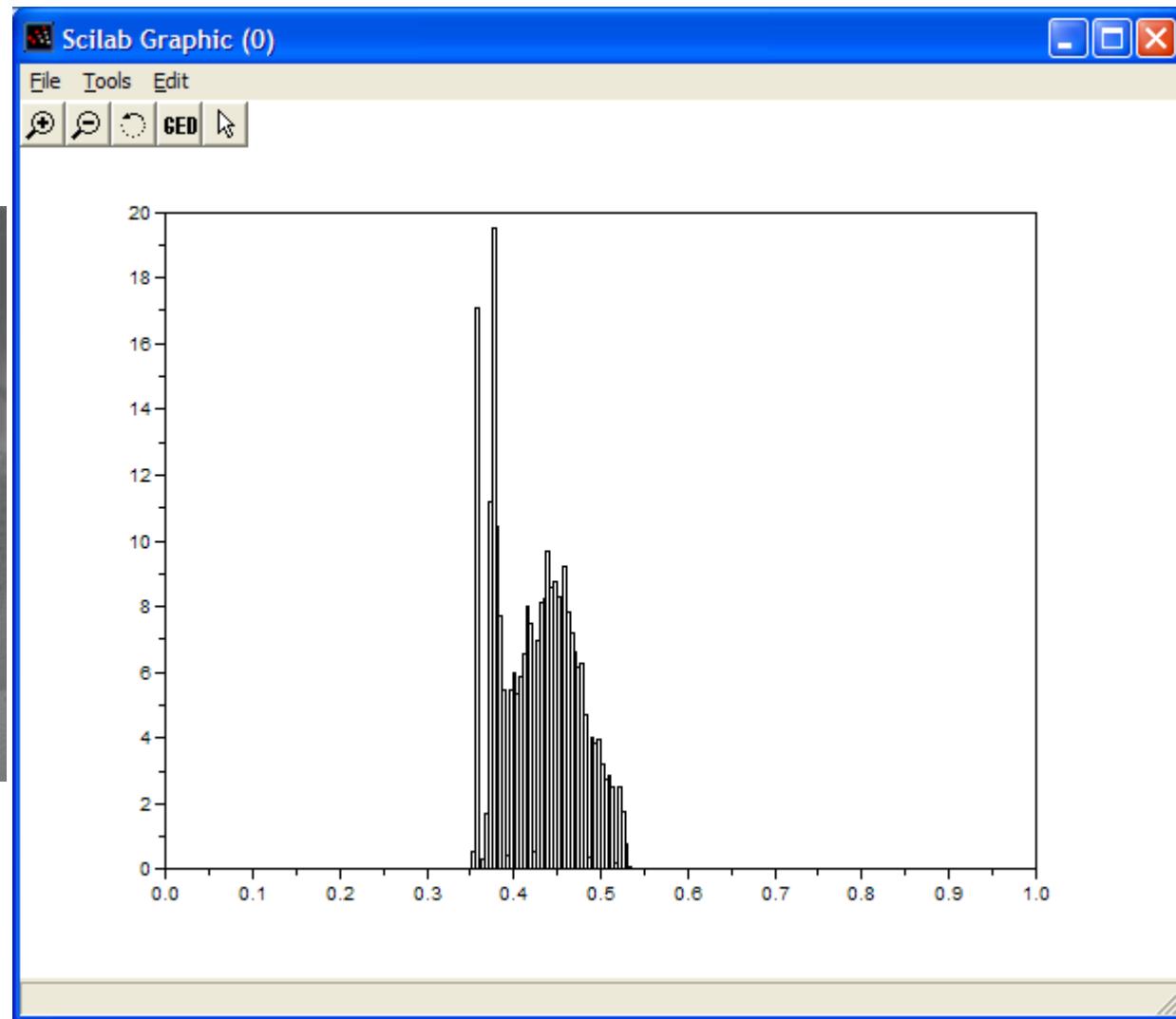
# Histogram Examples (cont...)



# Histogram Examples (cont...)



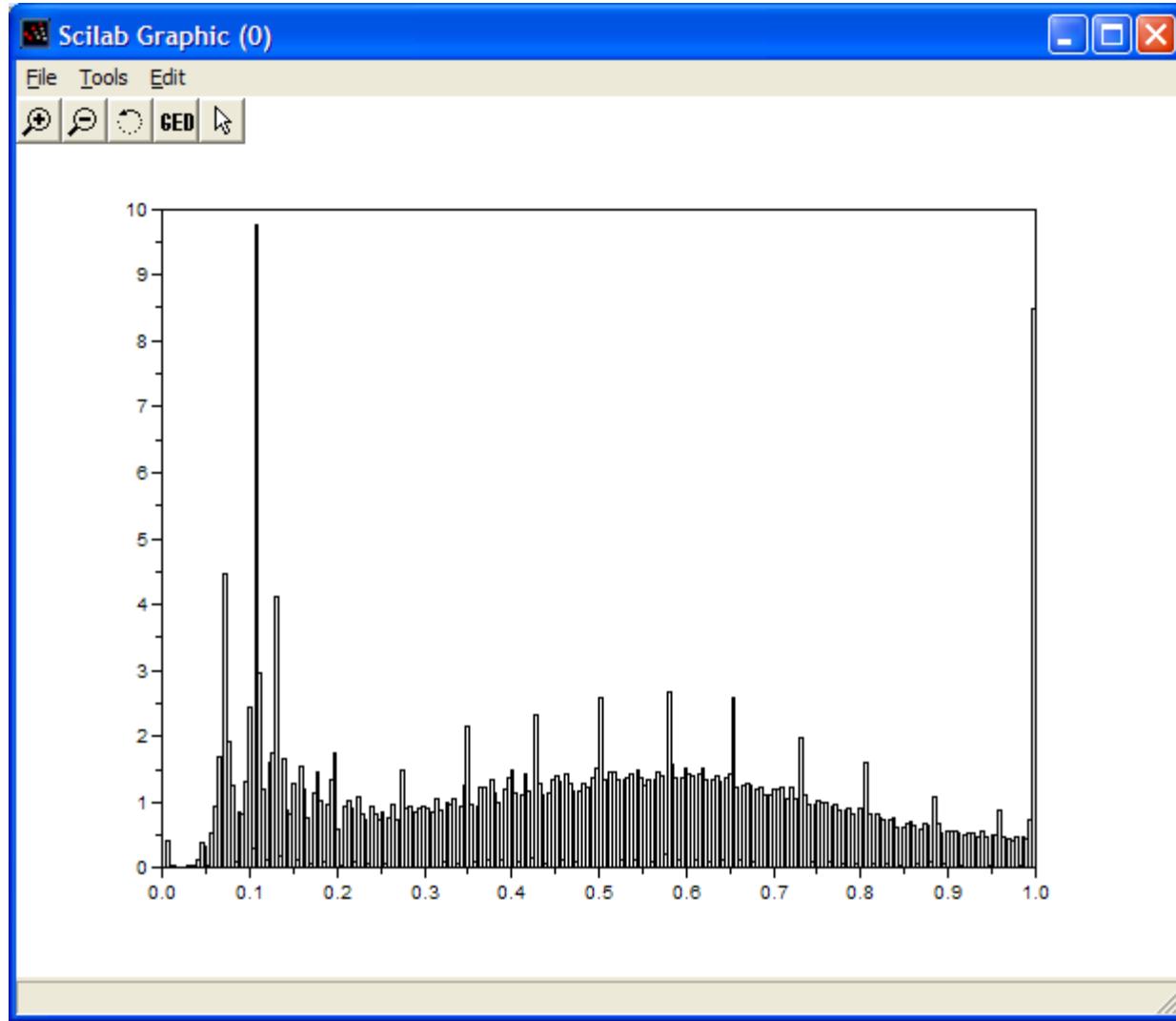
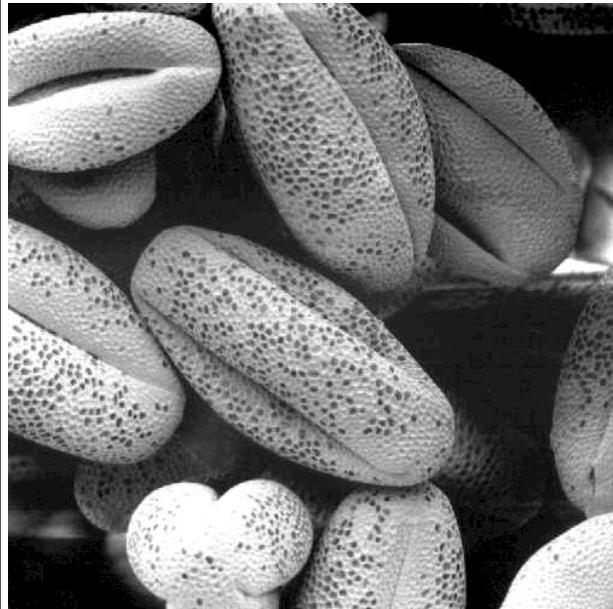
# Histogram Examples (cont...)



# Histogram Examples (cont...)



# Histogram Examples (cont...)

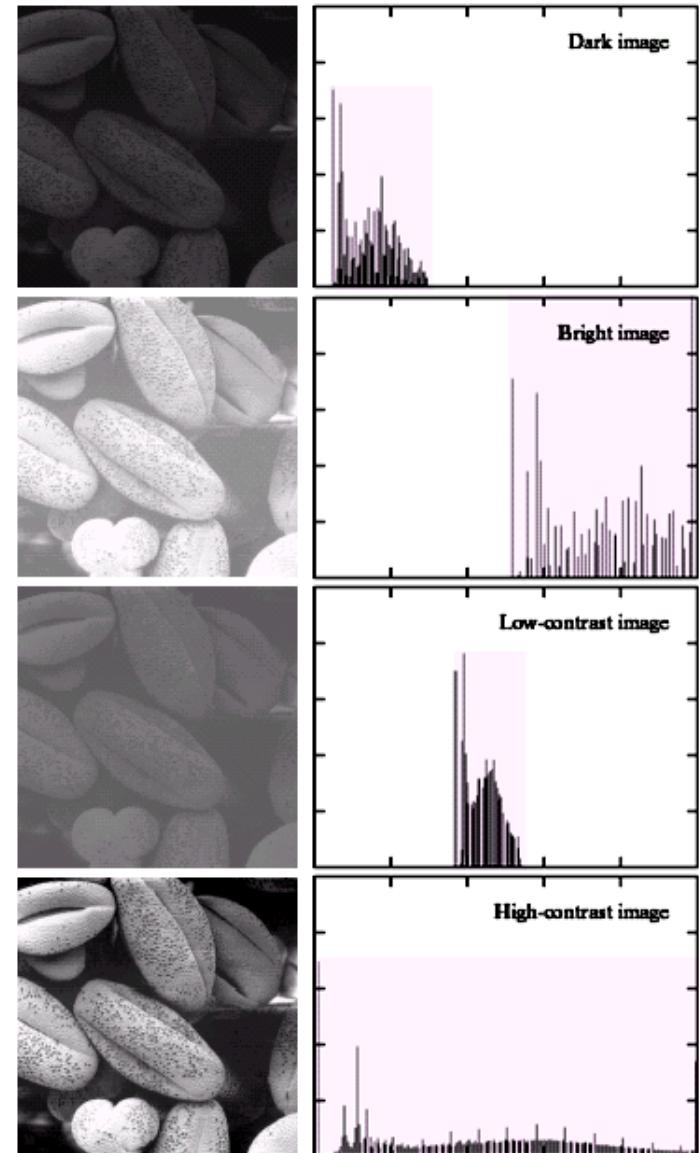


# Histogram Examples (cont...)

A selection of images and their histograms

Notice the relationships between the images and their histograms

Note that the high contrast image has the most evenly spaced histogram

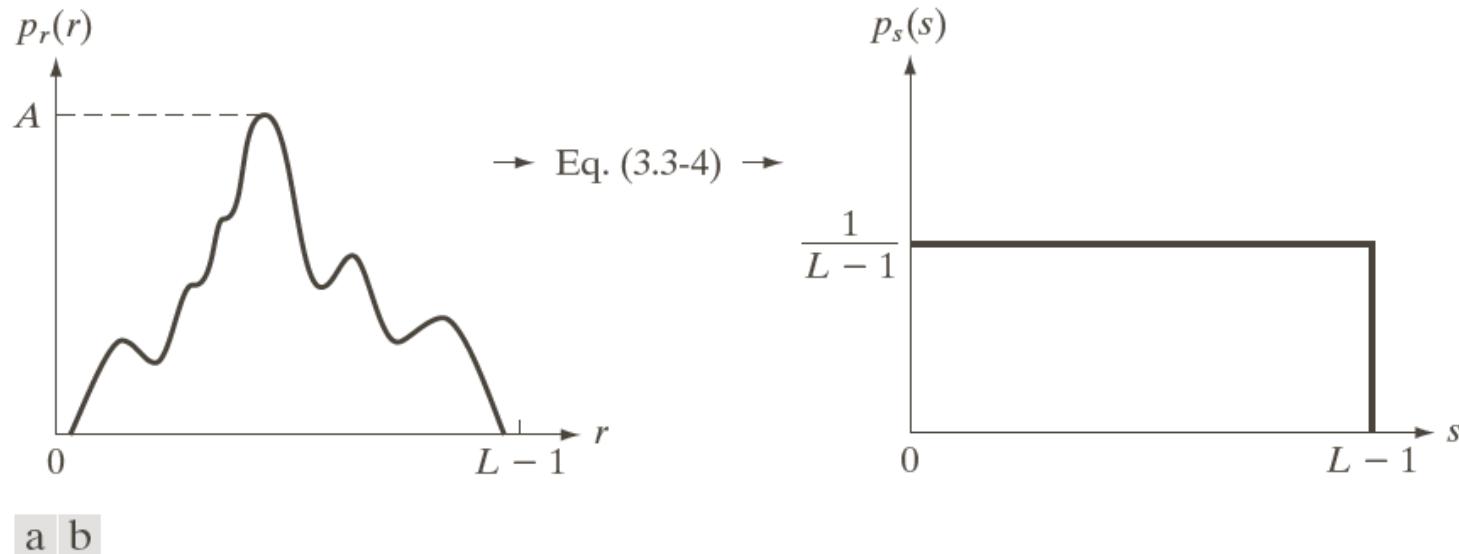


# Contrast Stretching

- Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images
- We can fix images that have poor contrast by applying a pretty simple contrast specification
- The interesting part is how do we decide on this transformation function?



# Histogram Equalisation

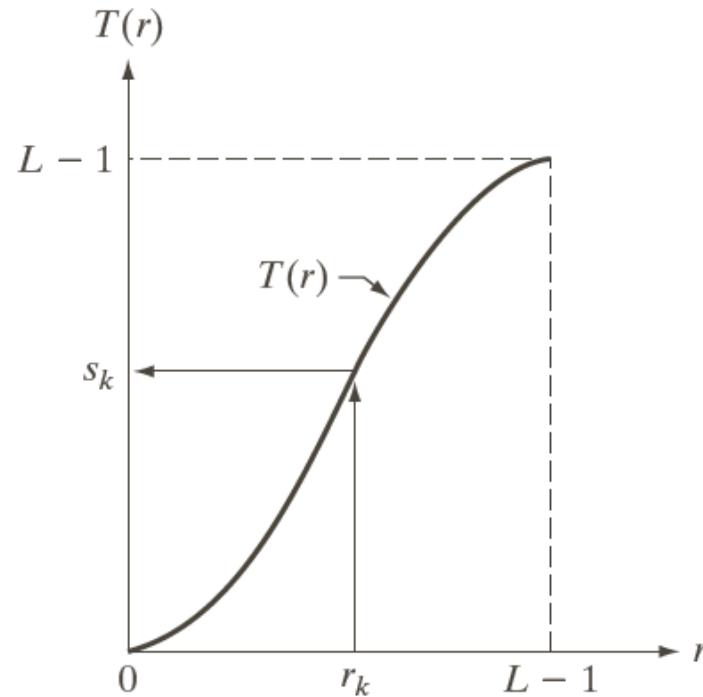
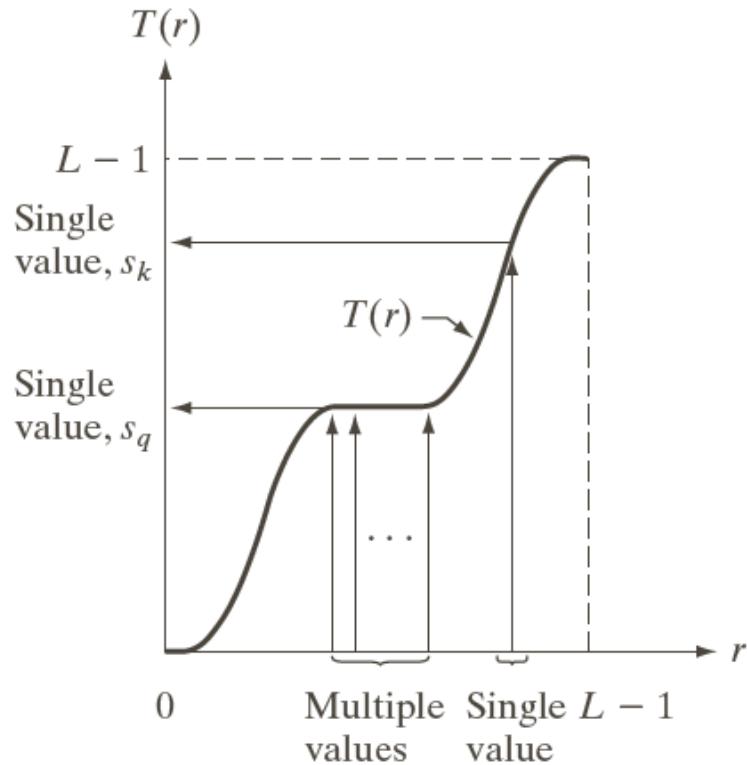


**FIGURE 3.18** (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels,  $r$ . The resulting intensities,  $s$ , have a uniform PDF, independently of the form of the PDF of the  $r$ 's.

# Histogram Equalisation

The transformation function should be:

- 1) Monotonically increasing
- 2)  $0 \leq T(r) \leq L-1$



a b

**FIGURE 3.17**  
(a) Monotonically increasing function, showing how multiple values can map to a single value.  
(b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

# Histogram Equalisation

The formula for histogram equalisation for discrete values is given where

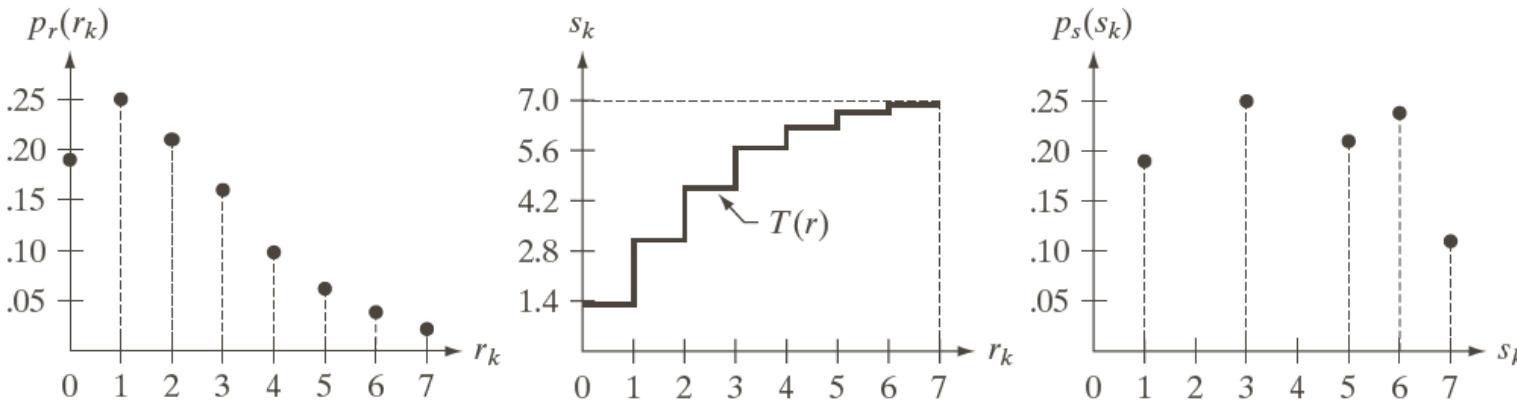
- $r_k$ : input intensity
- $s_k$ : processed intensity
- $k$ : the intensity range (e.g 0.0 – 1.0)
- $n_j$ : the frequency of intensity  $j$
- $MN$ : Total pixels

$$\begin{aligned}s_k &= T(r_k) \\ &= (L - 1) \sum_{j=0}^k p_r(r_j) \\ &= \frac{(L - 1)}{MN} \sum_{j=0}^k n_j\end{aligned}$$

# Equalisation Examples

$r_k$	$n_k$	$p_r(r_k) = n_k/MN$
$r_0 = 0$	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

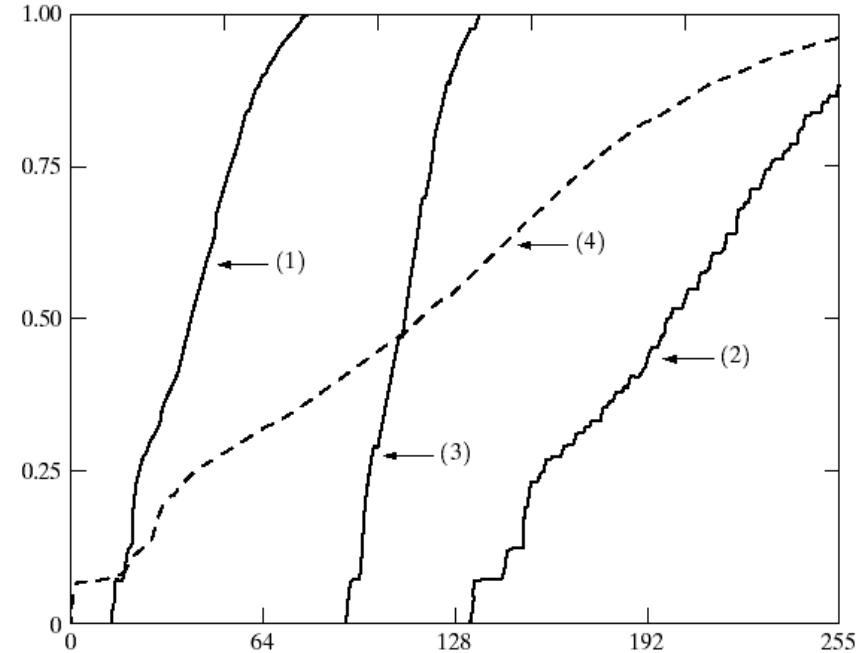
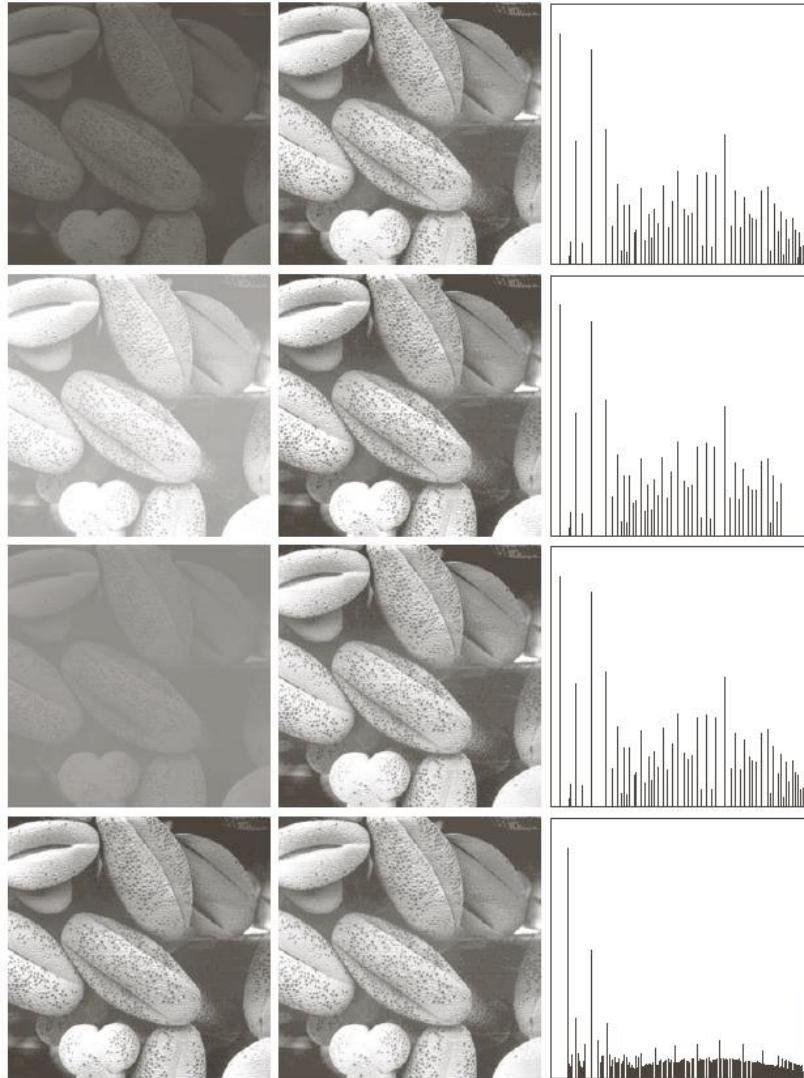
**TABLE 3.1**  
 Intensity distribution and histogram values for a 3-bit,  $64 \times 64$  digital image.



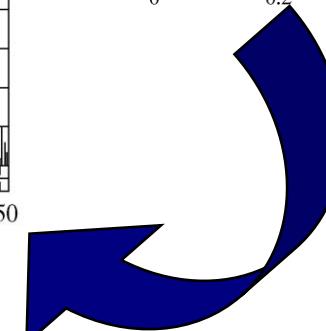
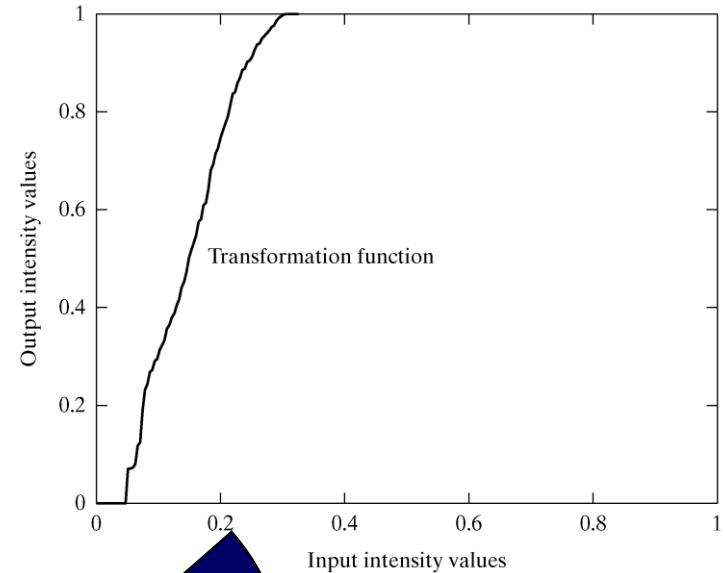
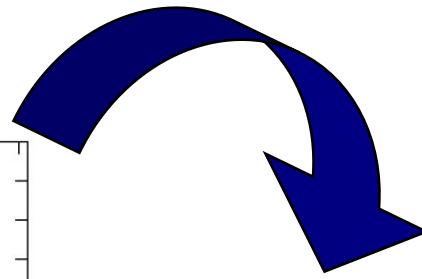
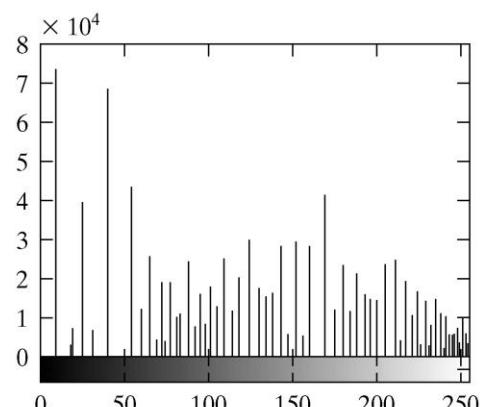
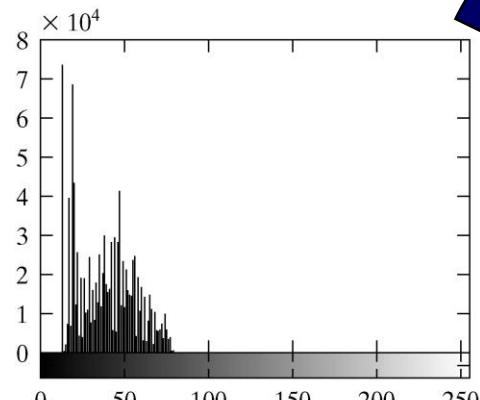
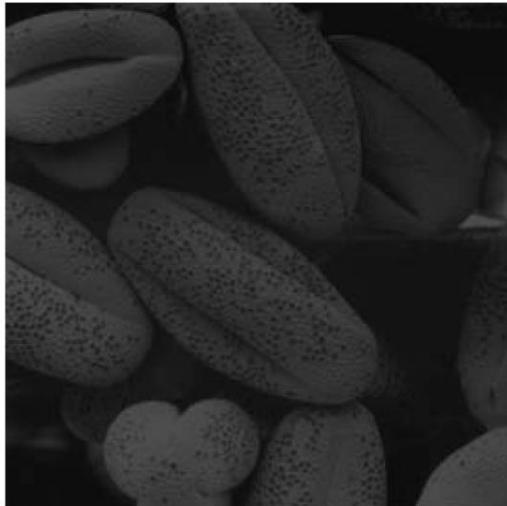
a b c

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

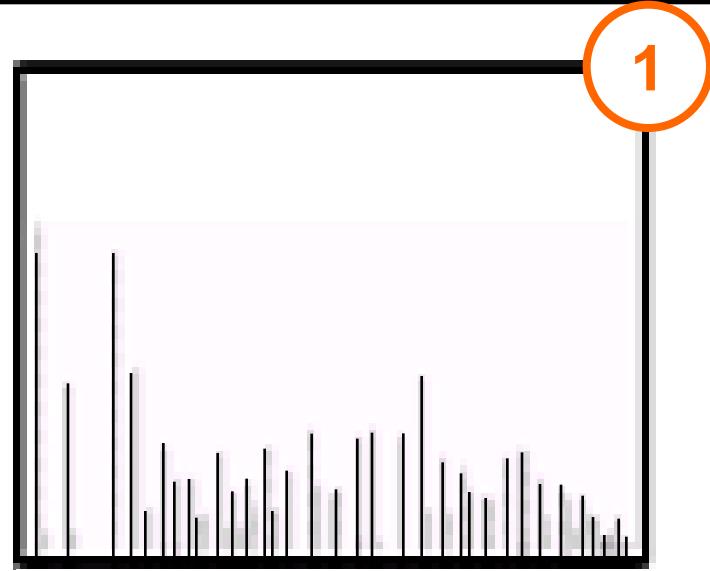
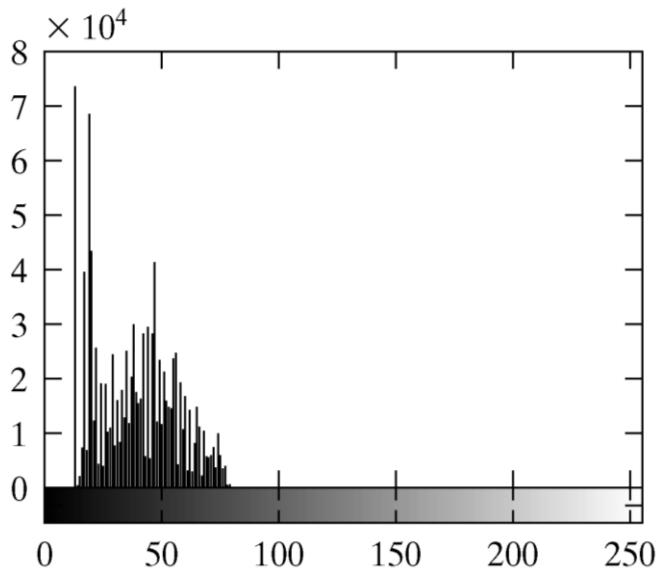
# Equalisation Examples



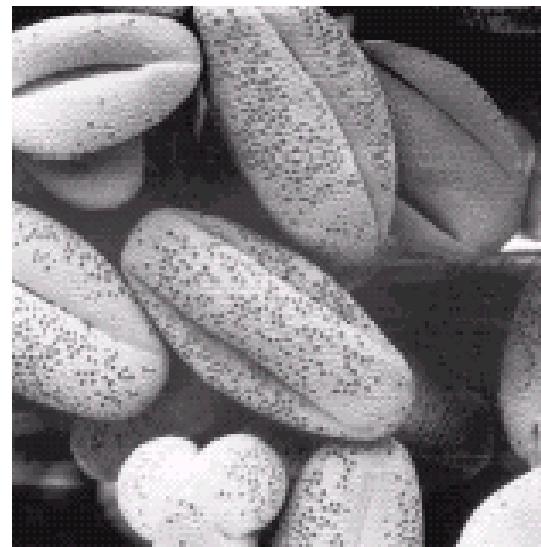
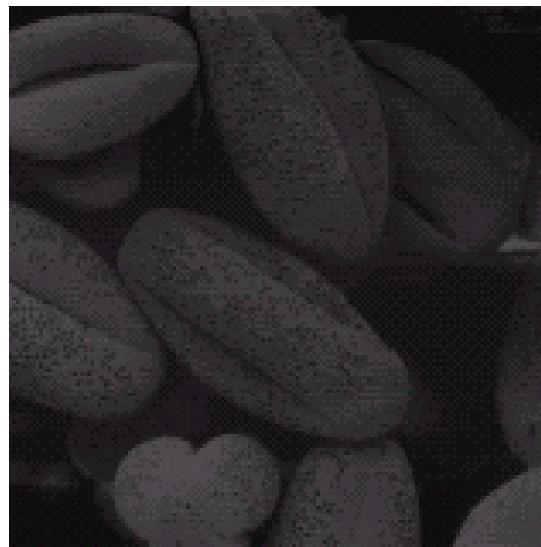
# Equalisation Transformation Function



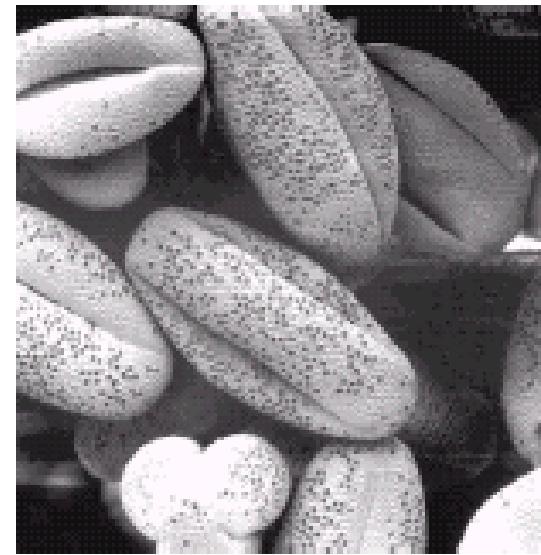
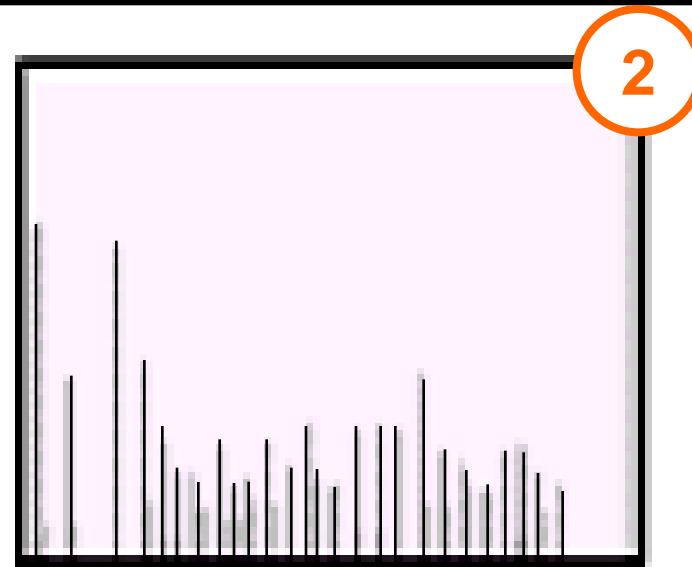
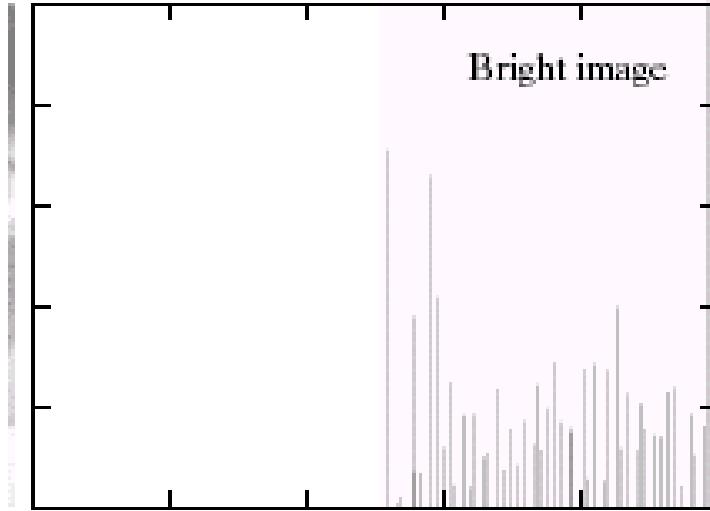
# Equalisation Examples



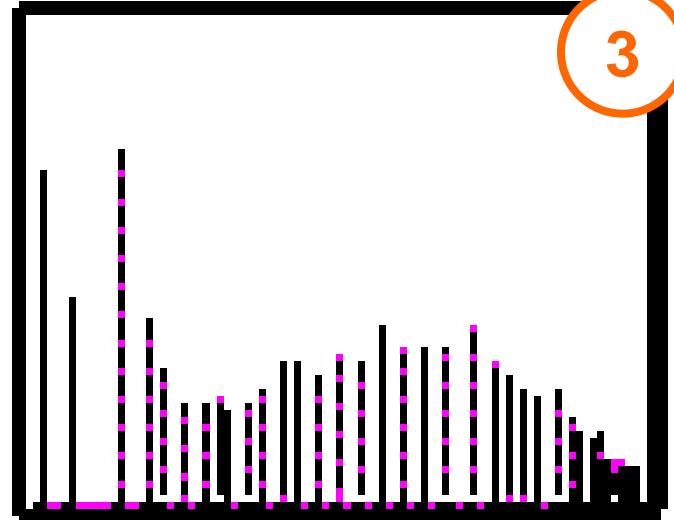
1



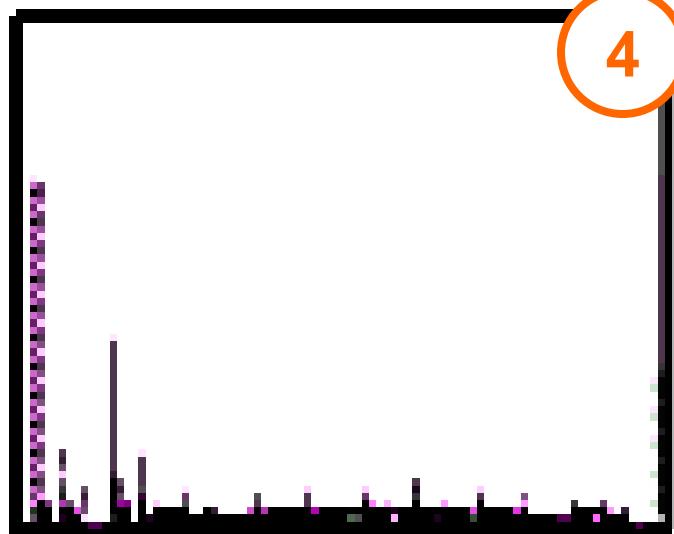
# Equalisation Examples



# Equalisation Examples (cont...)



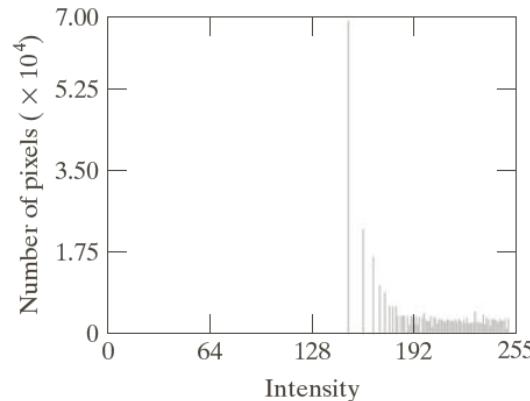
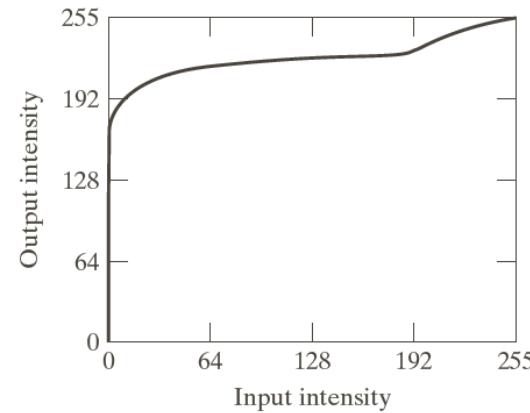
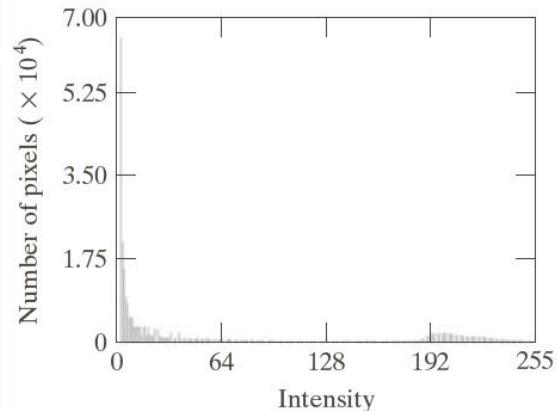
3



4

# Histogram Matching (Specification)

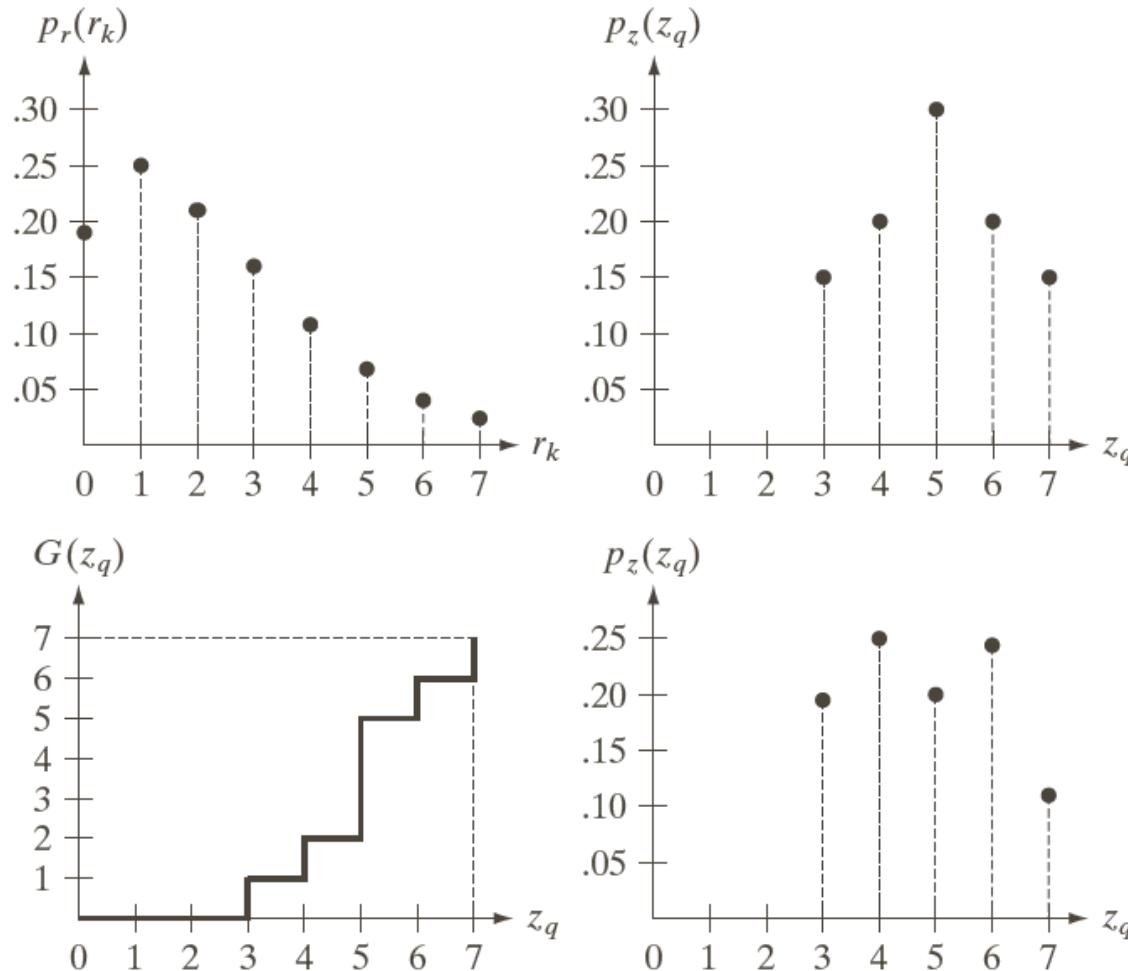
- For some applications, enhancing based on uniform histogram is not best approach



# Histogram Matching (Specification)

- Useful to be able to specify the shape of the histogram that we wish (guess!) the processed image to have
- Method to generate a processed image that has a specified histogram is called ***Histogram Matching or Histogram Specification***

# Histogram Matching (Specification)

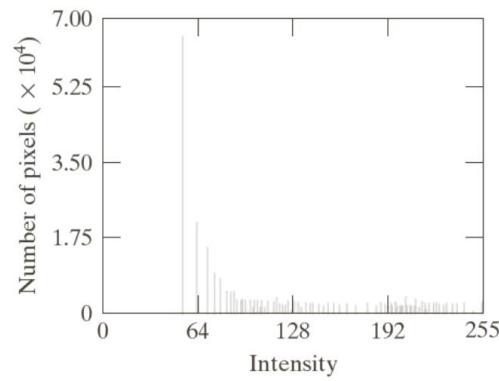
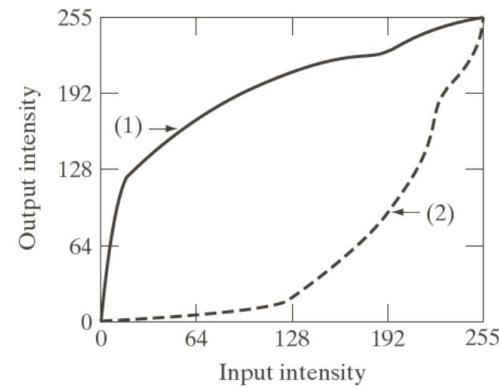
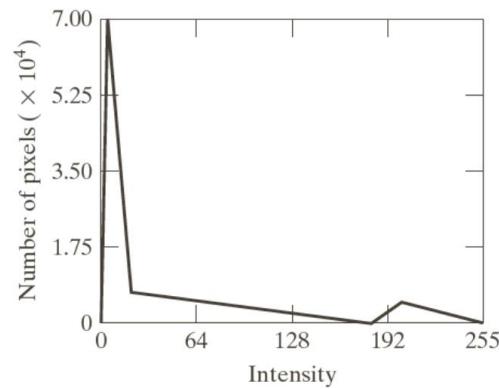
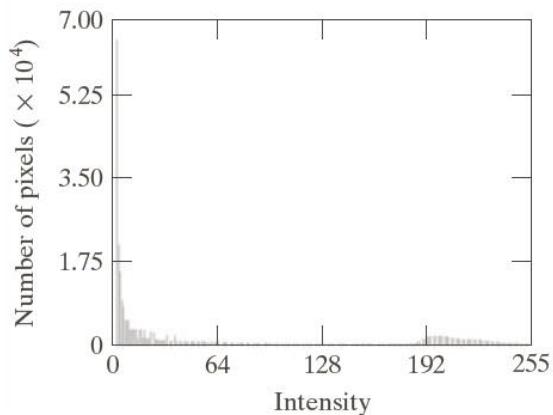
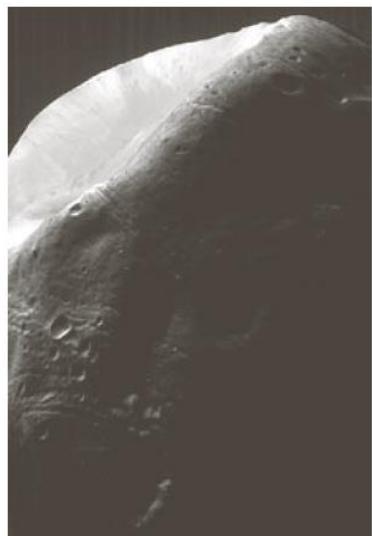


a	b
c	d

**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).

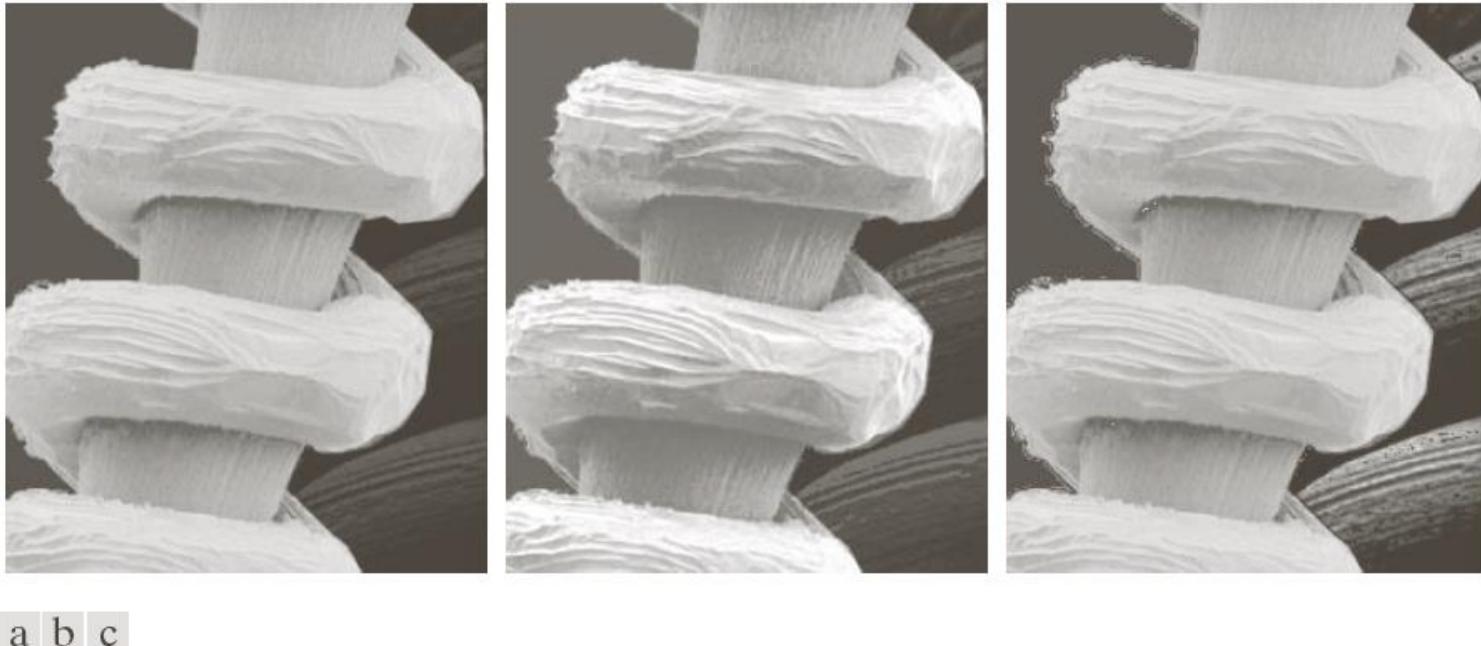
# Histogram Matching (Specification)



a  
c  
b  
d

**FIGURE 3.25**  
(a) Specified histogram.  
(b) Transformations.  
(c) Enhanced image using mappings from curve (2).  
(d) Histogram of (c).

# Local Histogram processing



a b c

**FIGURE 3.27** (a) SEM image of a tungsten filament magnified approximately 130×. (b) Result of global histogram equalization. (c) Image enhanced using local histogram statistics. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

We have looked at:

- Different kinds of image enhancement
- Histograms
- Histogram equalisation
- Histogram matching

Next time we will start to look at point processing and some neighbourhood operations

# Image & Video Processing

Image Enhancement  
(Spatial Filtering)

# Contents

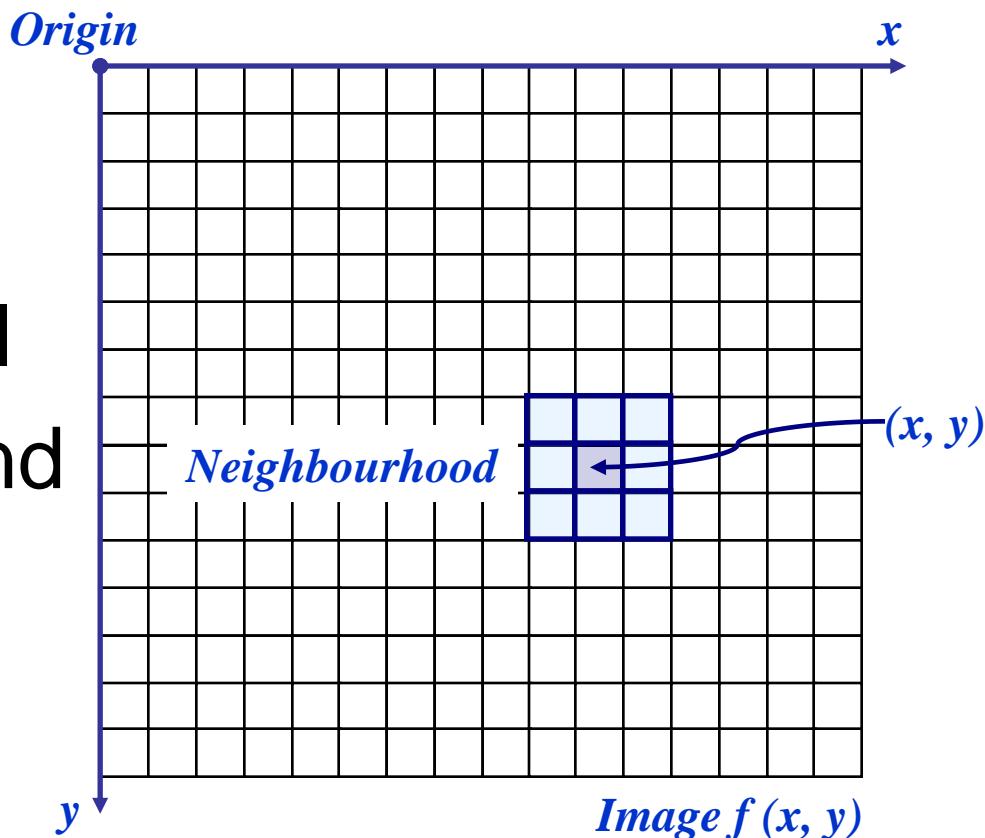
In this lecture we will look at spatial filtering techniques:

- Neighbourhood operations
- Spatial filtering : Correlation and convolution
- Smoothing operations
- Sharpening filters
  - 1<sup>st</sup> derivative filters
  - 2<sup>nd</sup> derivative filters
- Combining filtering techniques

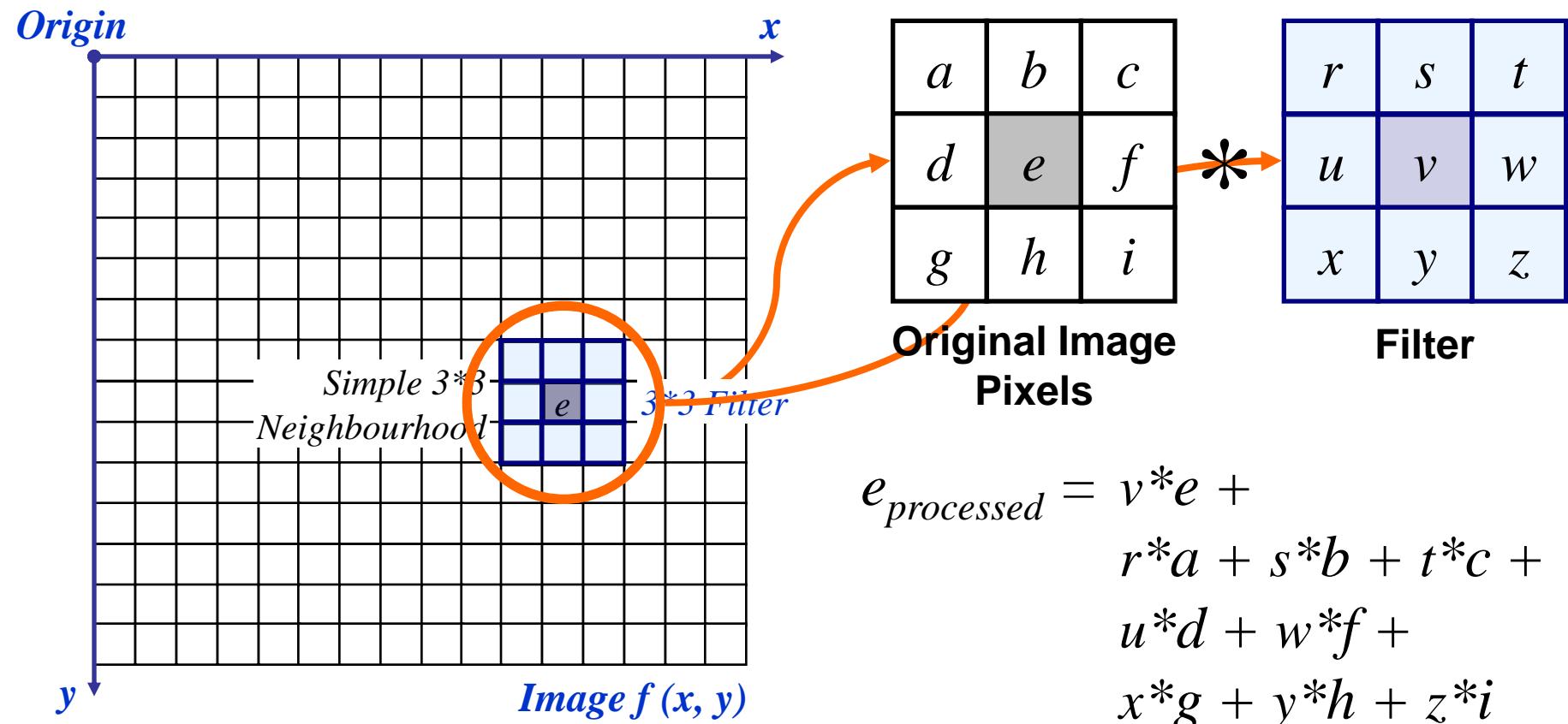
# Neighbourhood Operations

Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations

Neighbourhoods are mostly a rectangle around a central pixel  
(Although any size and shape are possible)

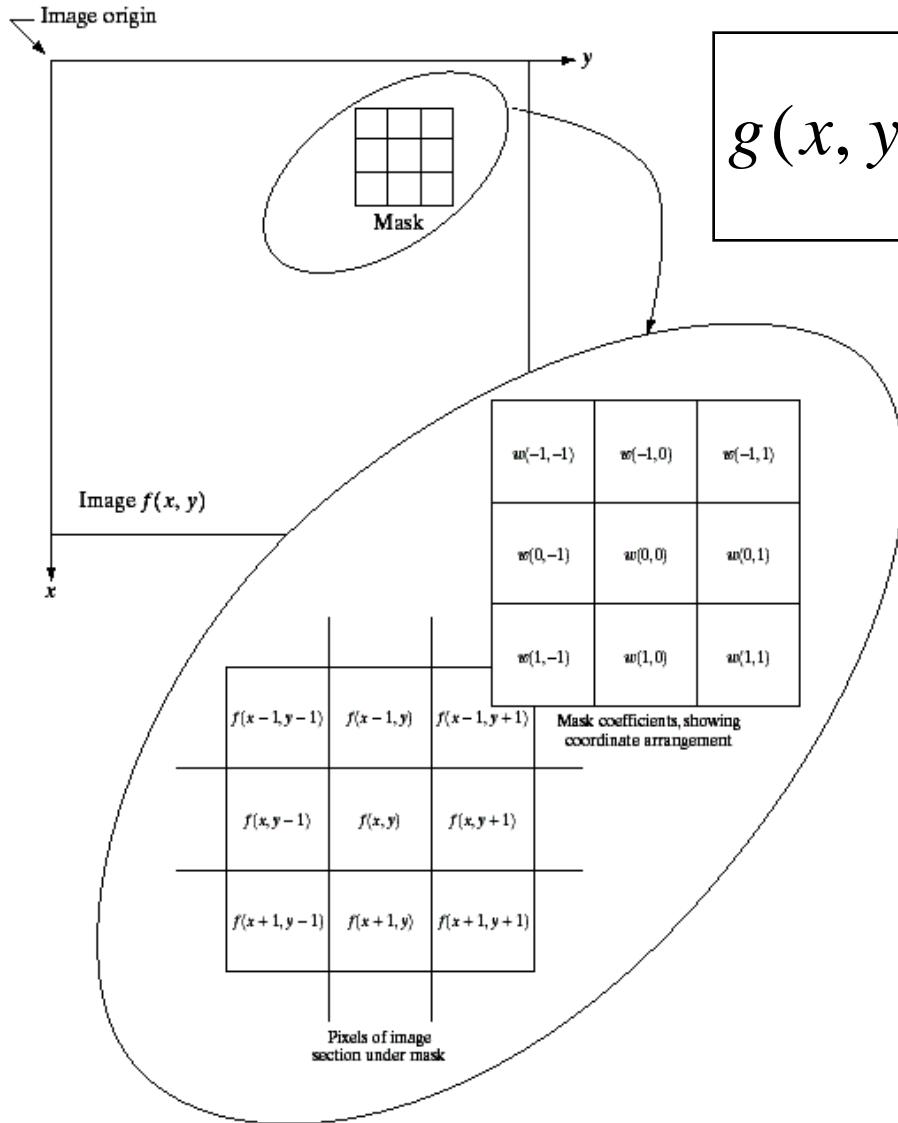


# The Spatial Filtering Process



The above is repeated for every pixel in the original image to generate the filtered image

# Spatial Filtering: Equation Form



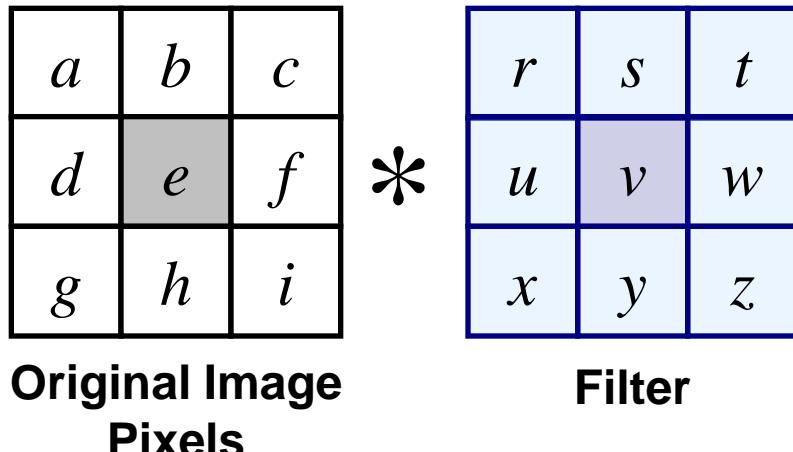
$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

# Correlation & Convolution

The filtering we have been talking so far is referred to as *correlation* with the filter itself. *Convolution* is a similar operation, with just one subtle difference.



$$e_{processed} = v^*e + z^*a + y^*b + x^*c + w^*d + u^*f + t^*g + s^*h + r^*i$$

For symmetric filters it makes no difference

# Smoothing Spatial Filters

One of the simplest spatial filtering operations we can perform is a smoothing operation

- Simply average all of the pixels in a neighbourhood around a central value
- Especially useful in removing noise from images
- Also useful for highlighting gross detail

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

Simple averaging filter

# Weighted Smoothing Filters

More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function

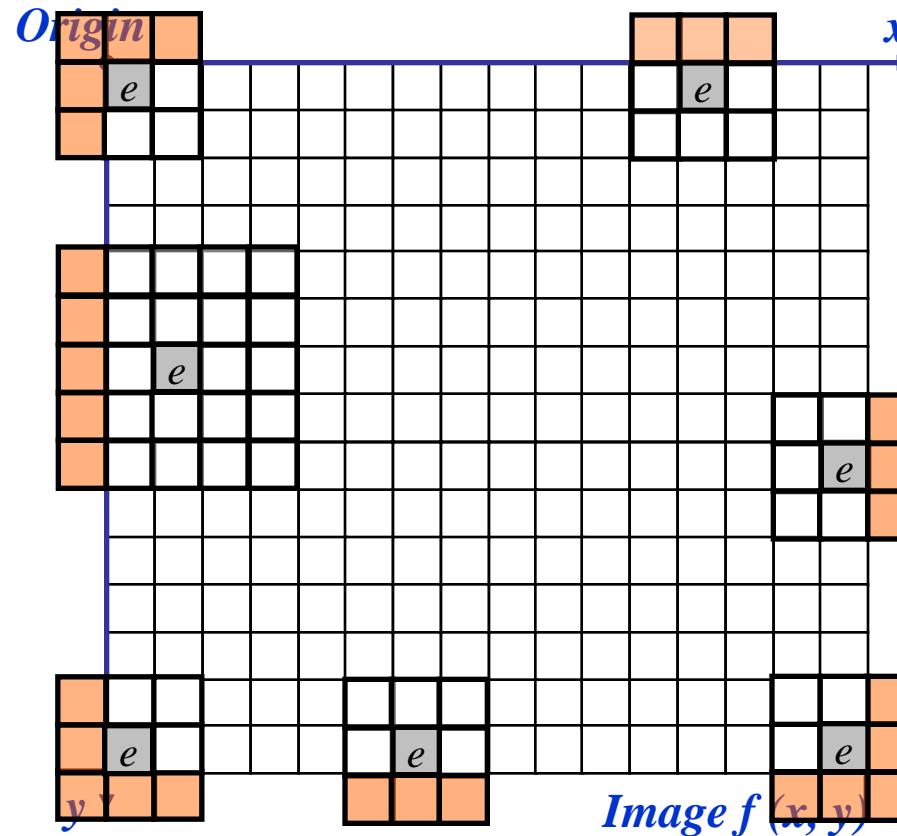
- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

$1/_{16}$	$2/_{16}$	$1/_{16}$
$2/_{16}$	$4/_{16}$	$2/_{16}$
$1/_{16}$	$2/_{16}$	$1/_{16}$

Weighted  
averaging filter

# What Happens At The Edges!

At the edges of an image we are missing pixels to form a neighbourhood

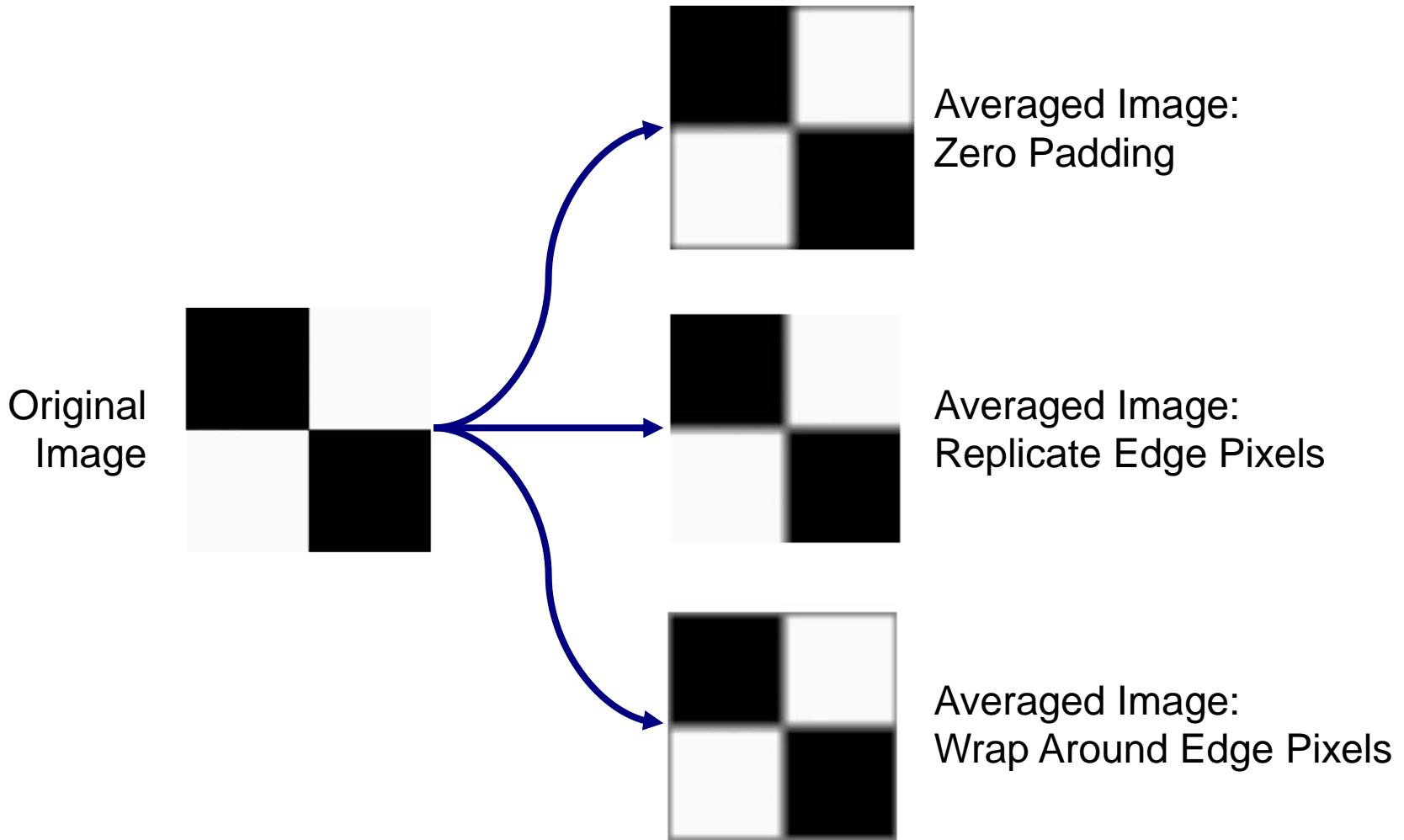


# What Happens At The Edges! (cont...)

There are a few approaches to dealing with missing edge pixels:

- Omit missing pixels
  - Can add extra code and slow down processing
- Pad the image
  - Typically with either all white or all black pixels
- Replicate border pixels
- Truncate the image
- Allow pixels *wrap around* the image
  - Can cause some strange image artefacts

# What Happens At The Edges!



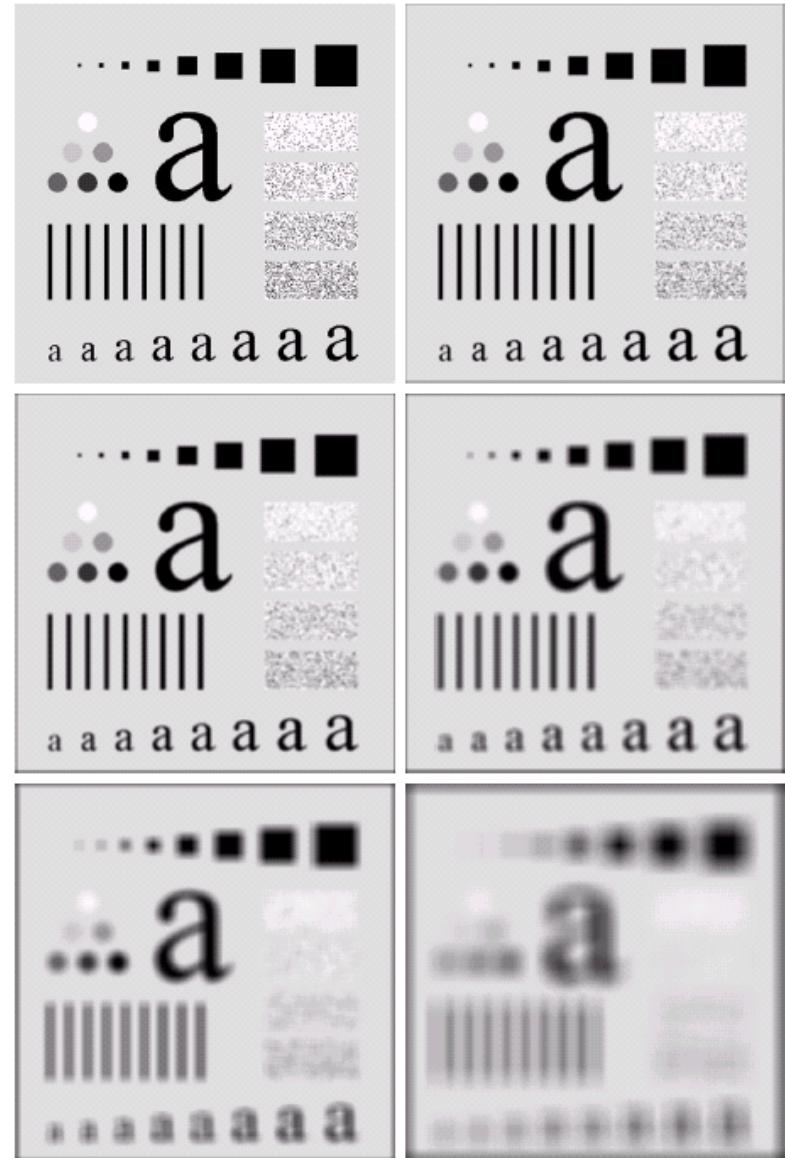
# Image Smoothing Example

The image at the top left is an original image of size 500\*500 pixels

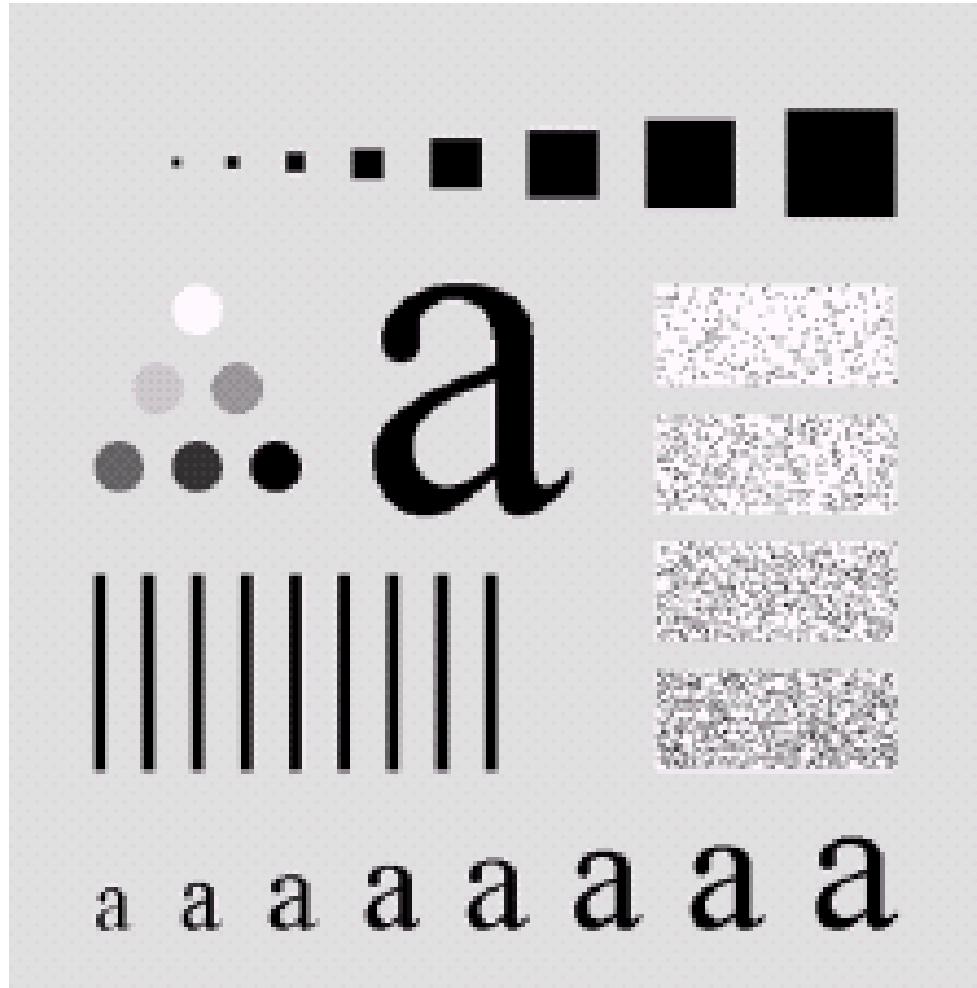
The subsequent images show the image after filtering with an averaging filter of increasing sizes

- 3, 5, 9, 15 and 35

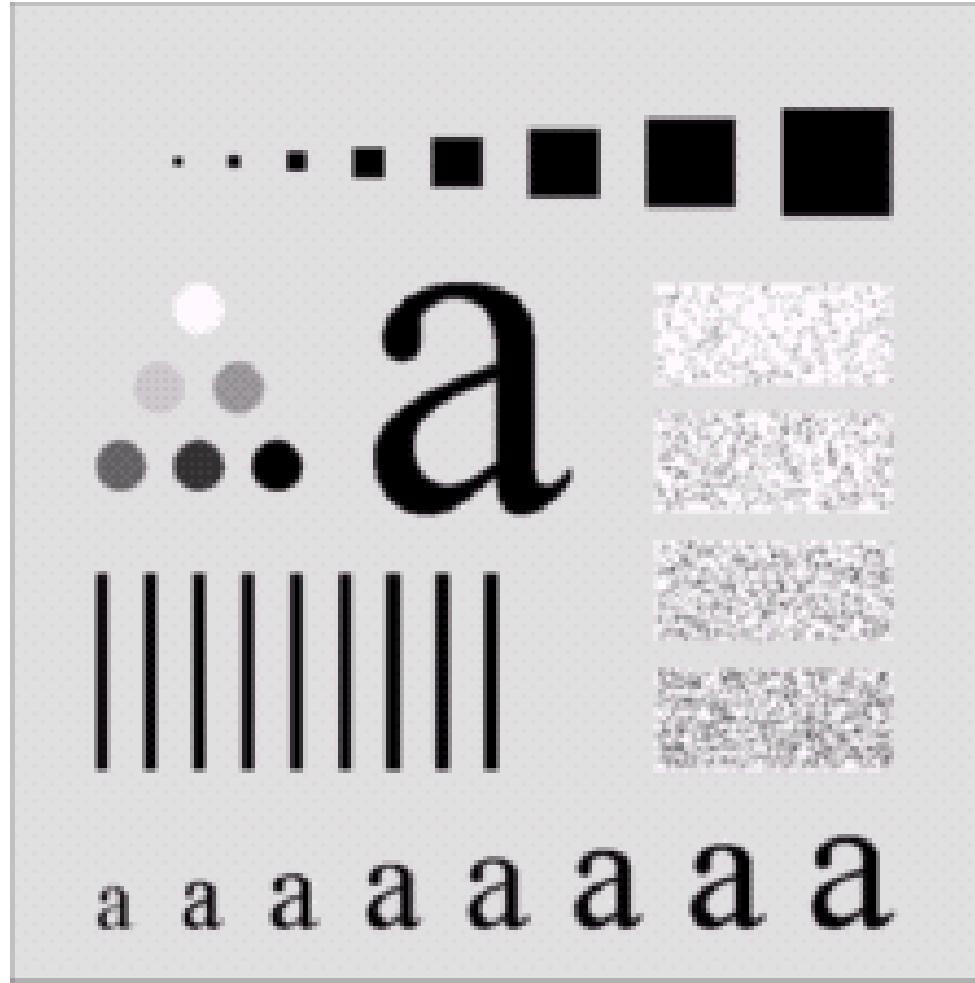
Notice how detail begins to disappear



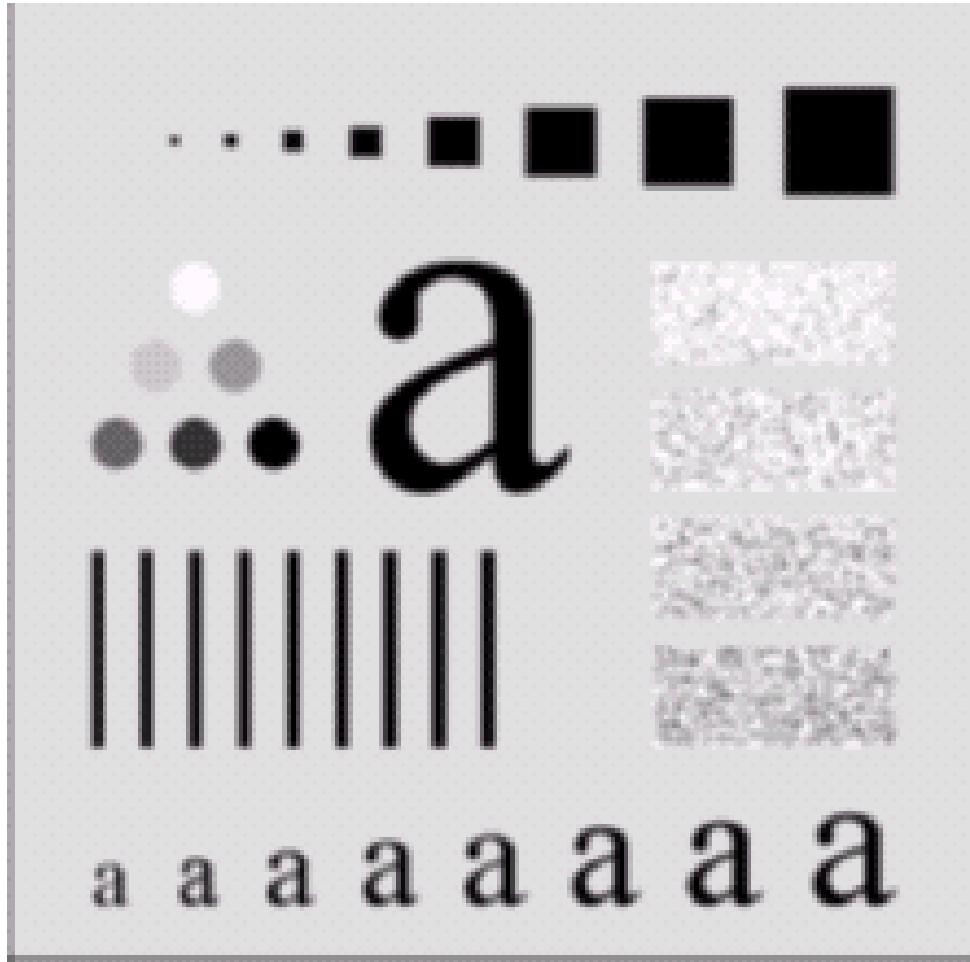
# Image Smoothing Example



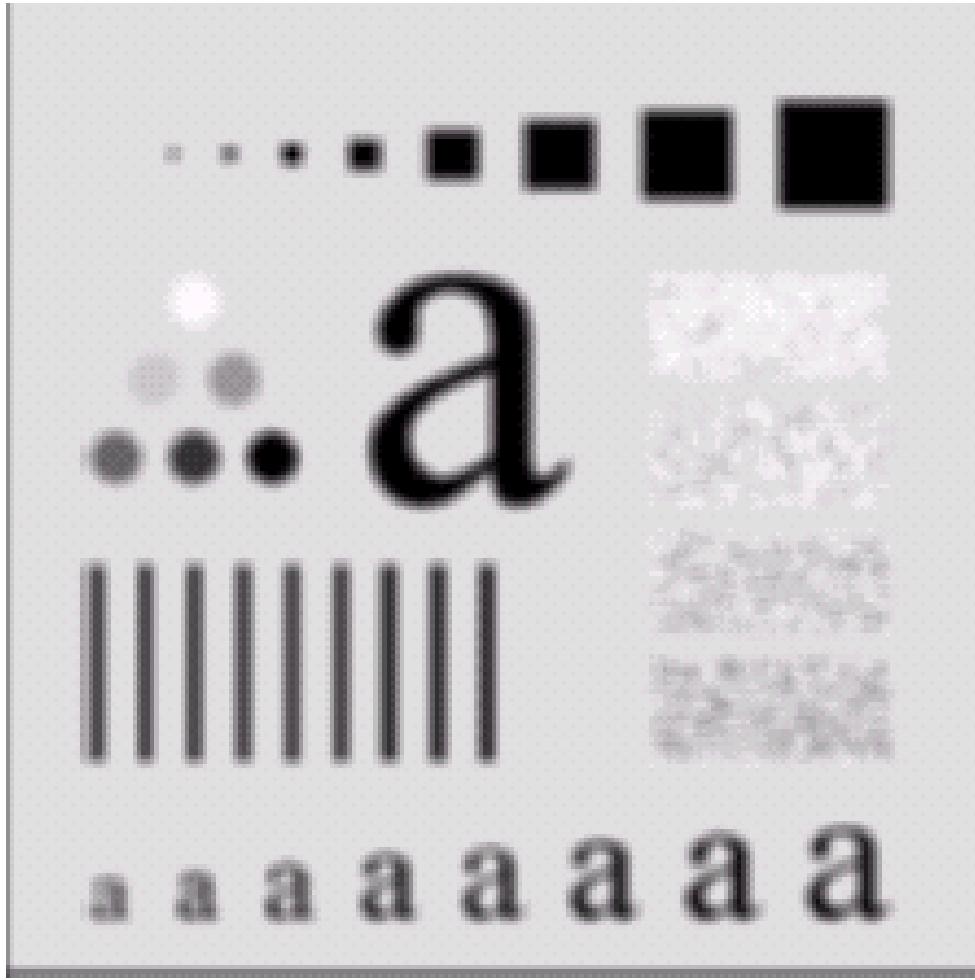
# Image Smoothing Example



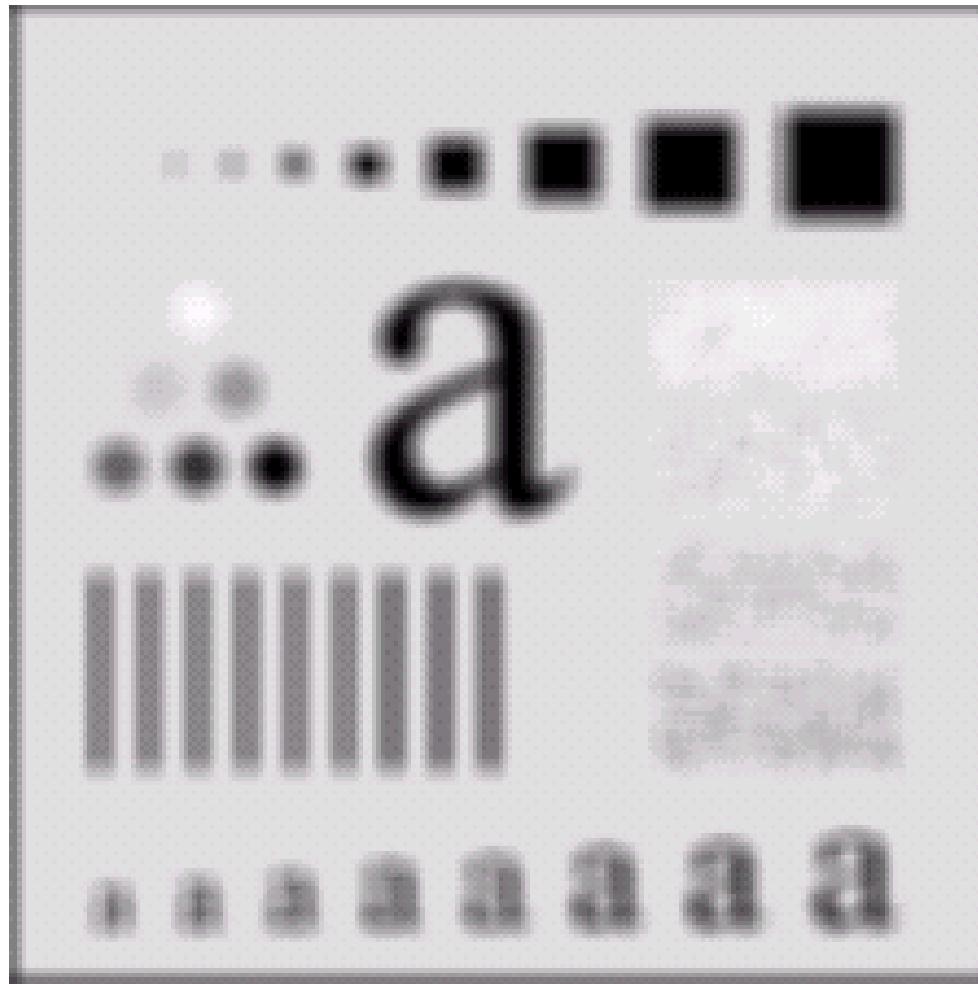
# Image Smoothing Example



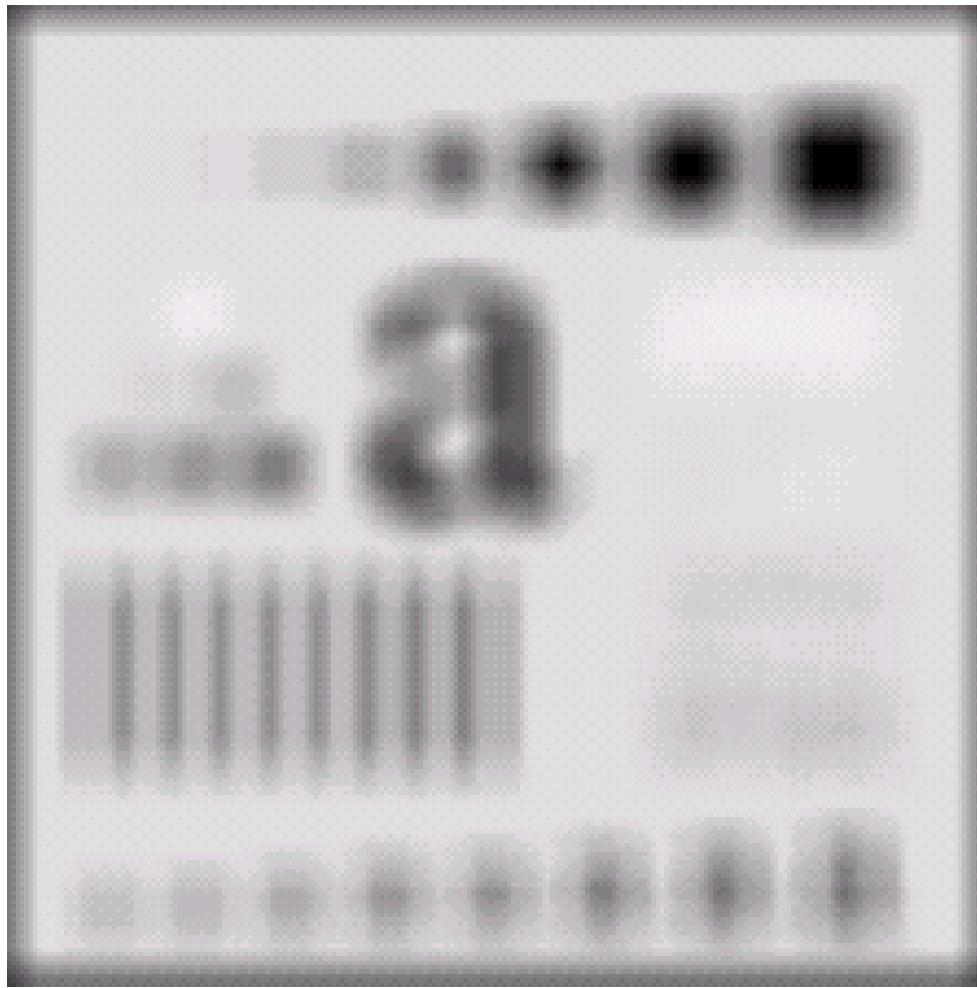
# Image Smoothing Example



# Image Smoothing Example



# Image Smoothing Example

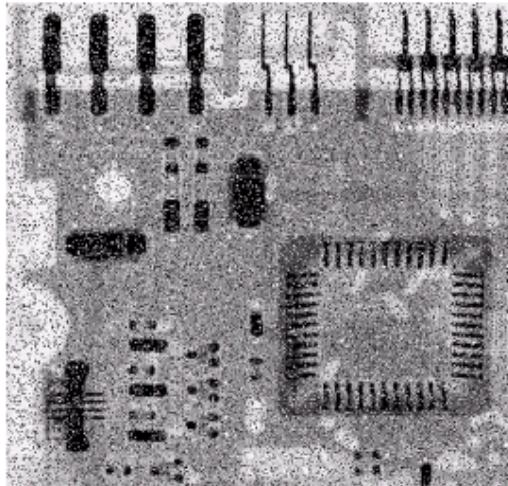


# Order Statistics Filters

Some simple neighbourhood operations include:

- **Min:** Set the pixel value to the minimum in the neighbourhood
- **Max:** Set the pixel value to the maximum in the neighbourhood
- **Mean:** Set the pixel value to the mean in the neighbourhood
- **Median:** The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median).

# Averaging Filter Vs. Median Filter Example



Original Image  
With Noise

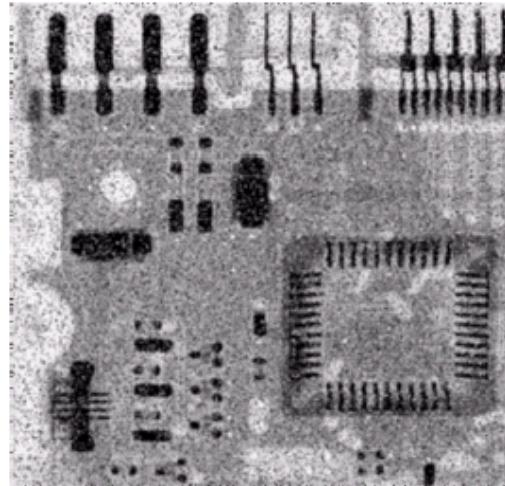


Image After  
Averaging Filter

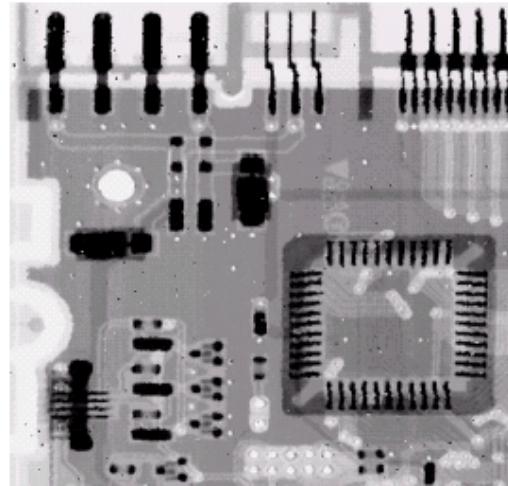
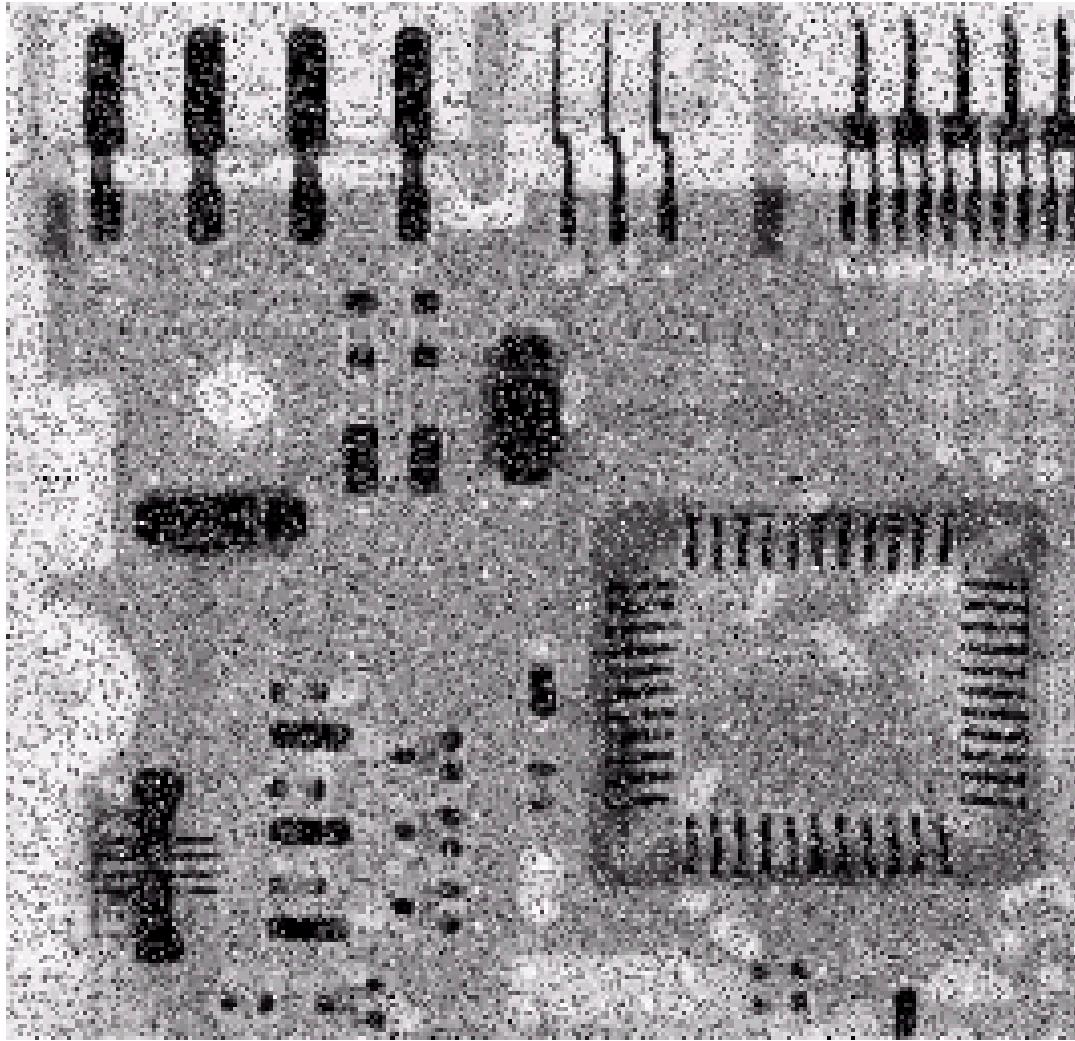


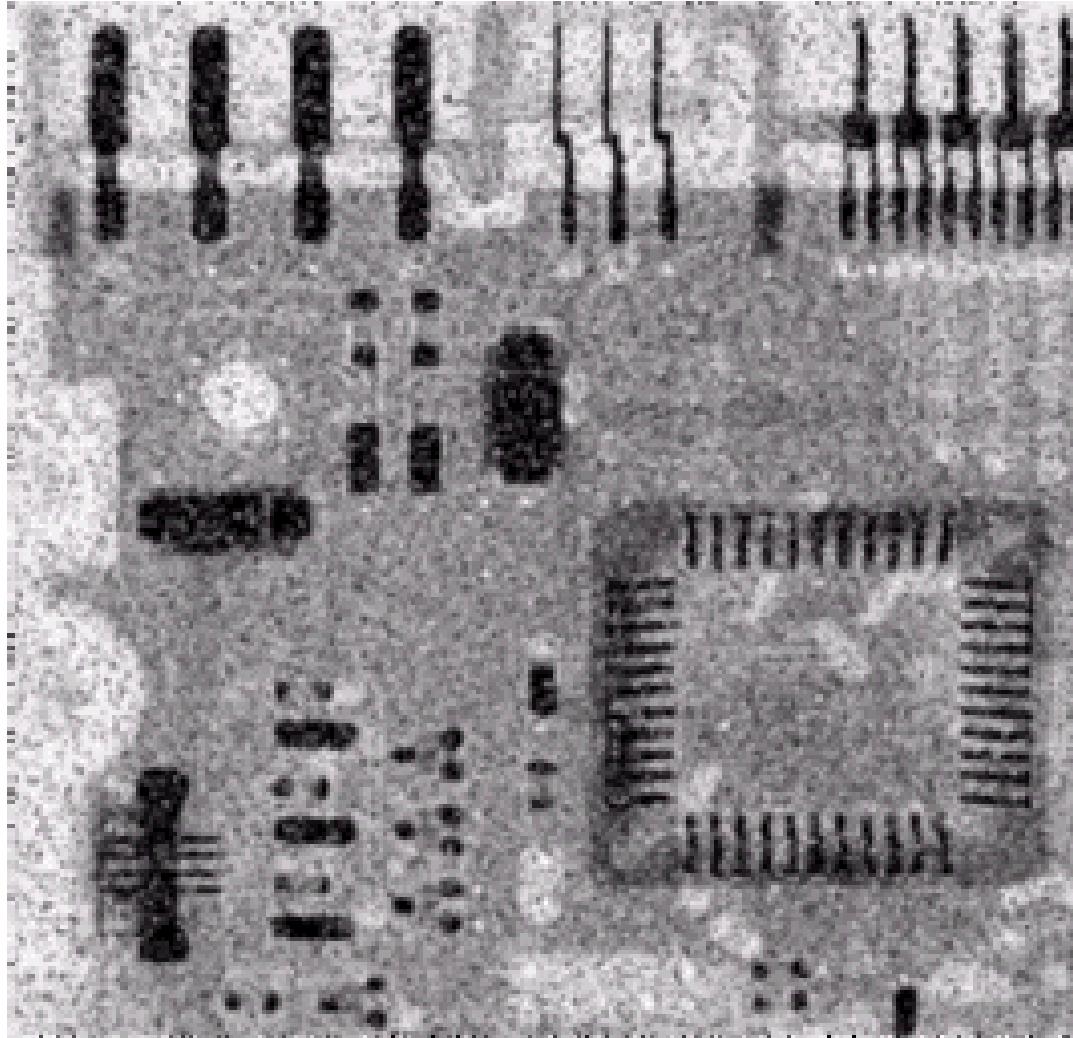
Image After  
Median Filter

Many times a median filter works better than an averaging filter for removing random noise

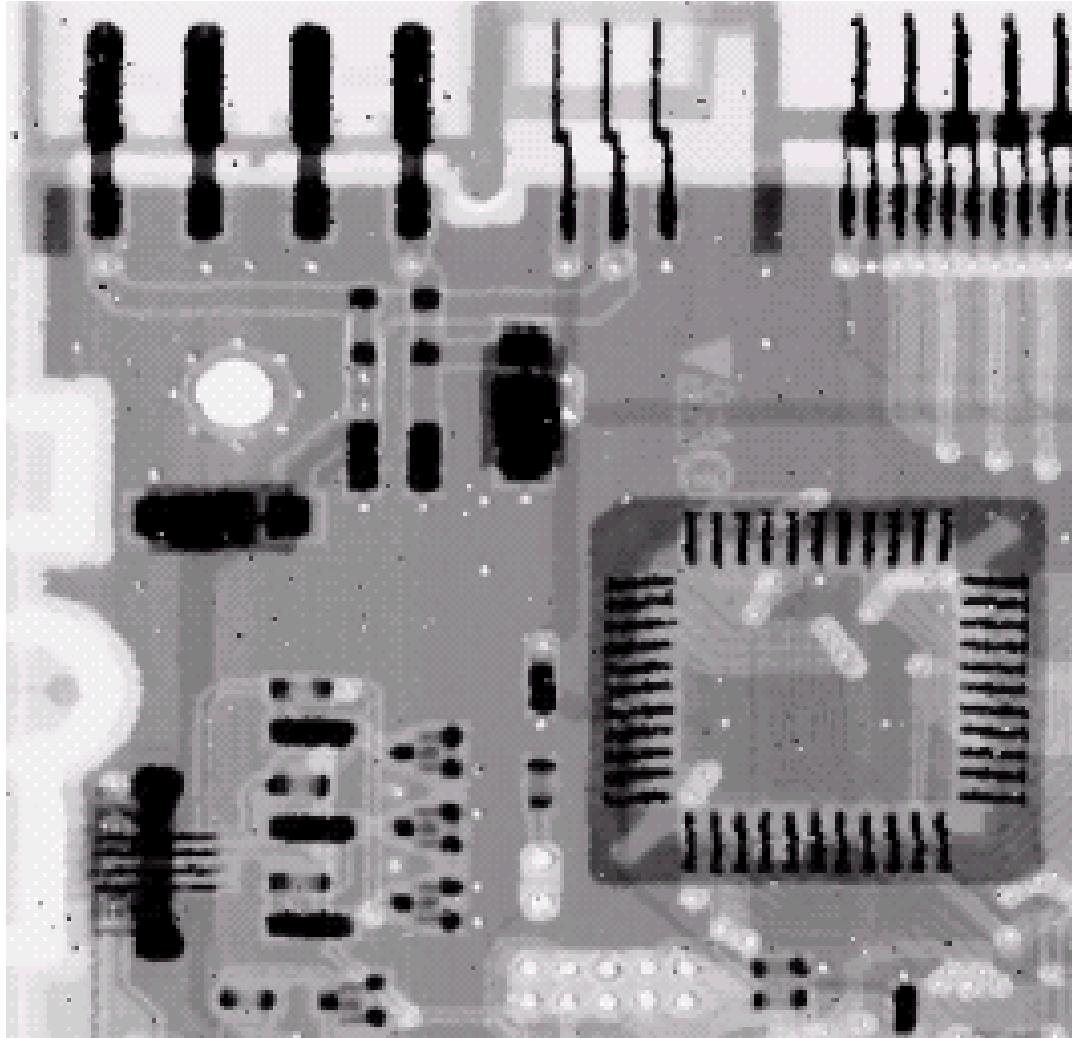
# Averaging Filter Vs. Median Filter Example



# Averaging Filter Vs. Median Filter Example



# Averaging Filter Vs. Median Filter Example



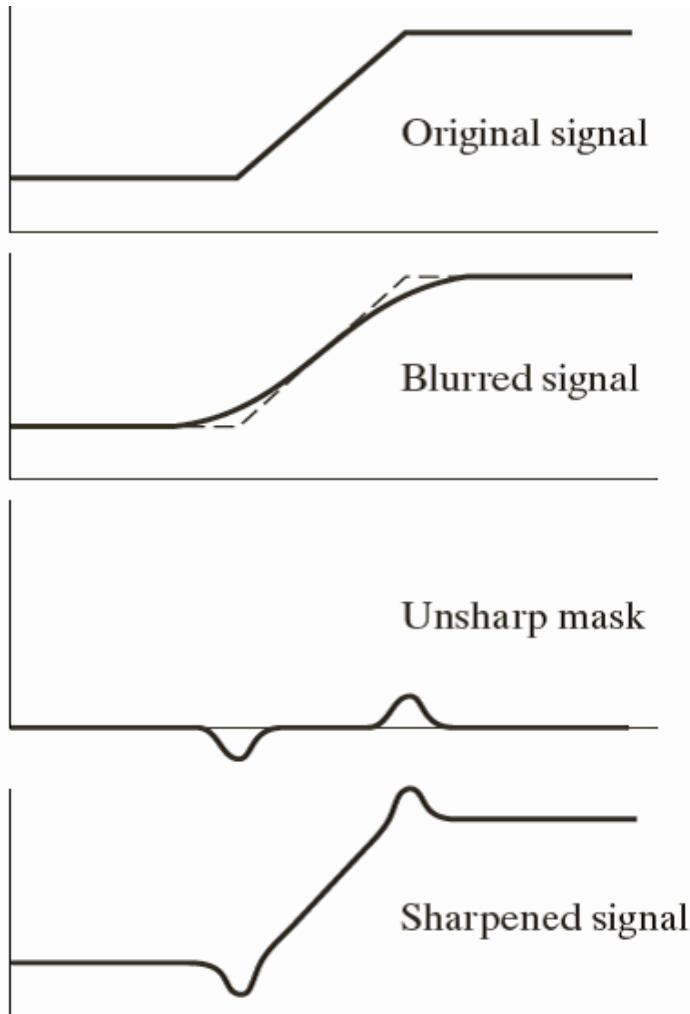
# Sharpening Spatial Filters

*Sharpening spatial filters* seek to highlight fine detail

- Remove blurring from images
- Highlight edges

Sharpening filters are based on *spatial differentiation*

# Unsharp masking & Highboost filtering



# Spatial Differentiation

Differentiation measures the *rate of change* of a function. The formula for the **1<sup>st</sup> derivative** of a function is as follows:

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

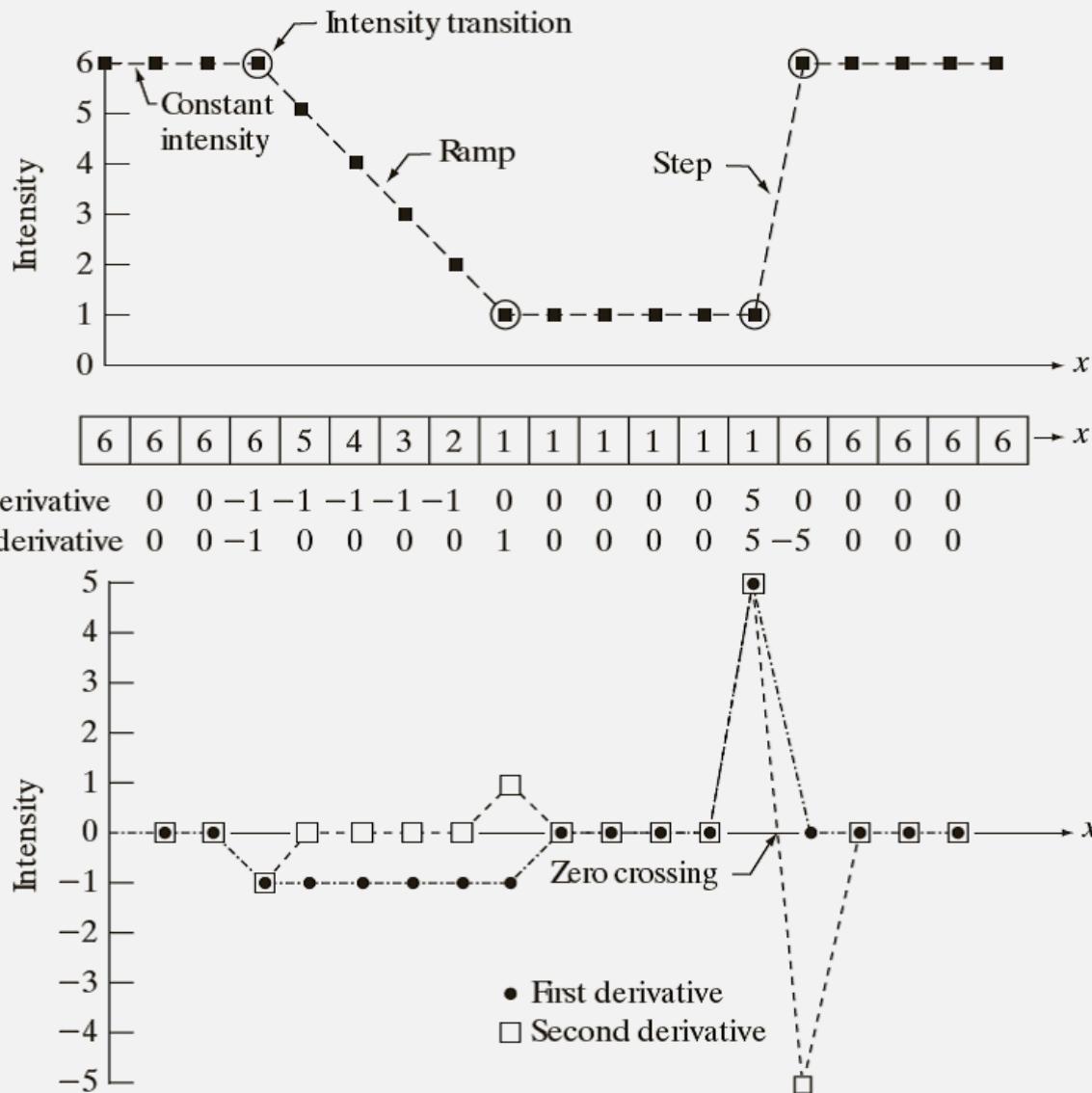
It's just the difference between subsequent values and measures the rate of change of the function

# 2<sup>nd</sup> Derivative

The formula for the 2<sup>nd</sup> derivative of a function is as follows:

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

Simply takes into account the values both before and after the current value



**Let's consider a simple 1 dimensional example**

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.

# Using Second Derivatives For Image Enhancement

The 2<sup>nd</sup> derivative is more useful for image enhancement than the 1<sup>st</sup> derivative

- Stronger response to fine detail
- Simpler implementation
- We will come back to the 1<sup>st</sup> order derivative later on

The first sharpening filter we will look at is the *Laplacian*

- Isotropic
- One of the simplest sharpening filters
- We will look at a digital implementation

# The Laplacian

The Laplacian is defined as follows:

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

where the partial 1<sup>st</sup> order derivative in the  $x$  direction is defined as follows:

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

and in the  $y$  direction as follows:

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

# The Laplacian (cont...)

So, the Laplacian can be given as follows:

$$\begin{aligned}\nabla^2 f = & [f(x+1, y) + f(x-1, y) \\& + f(x, y+1) + f(x, y-1)] \\& - 4f(x, y)\end{aligned}$$

We can easily build a filter based on this

0	1	0
1	-4	1
0	1	0

# Variants On The Simple Laplacian

There are lots of slightly different versions of the Laplacian that can be used:

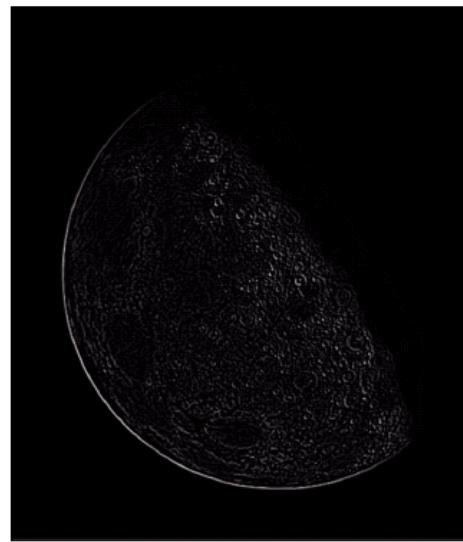
0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1
0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

# The Laplacian (cont...)

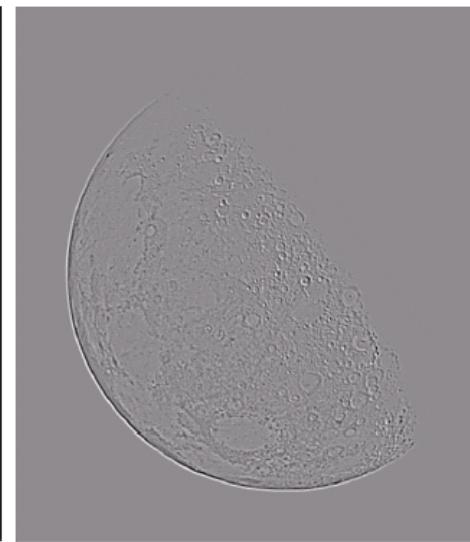
Applying the Laplacian to an image we get a new image that highlights edges and other discontinuities



Original  
Image



Laplacian  
Filtered Image



Laplacian  
Filtered Image  
Scaled for Display

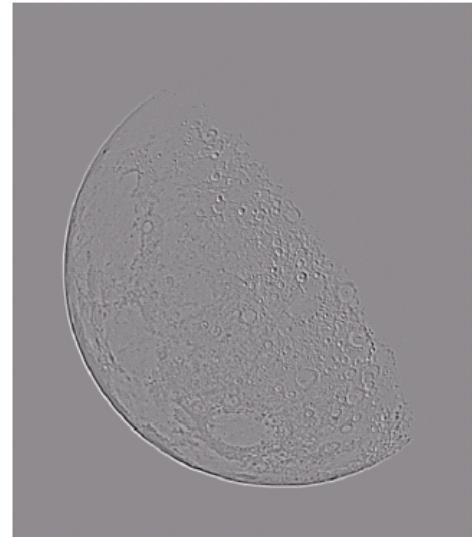
# But That Is Not Very Enhanced!

The result of a Laplacian filtering is not an enhanced image

We have to do more work in order to get our final image

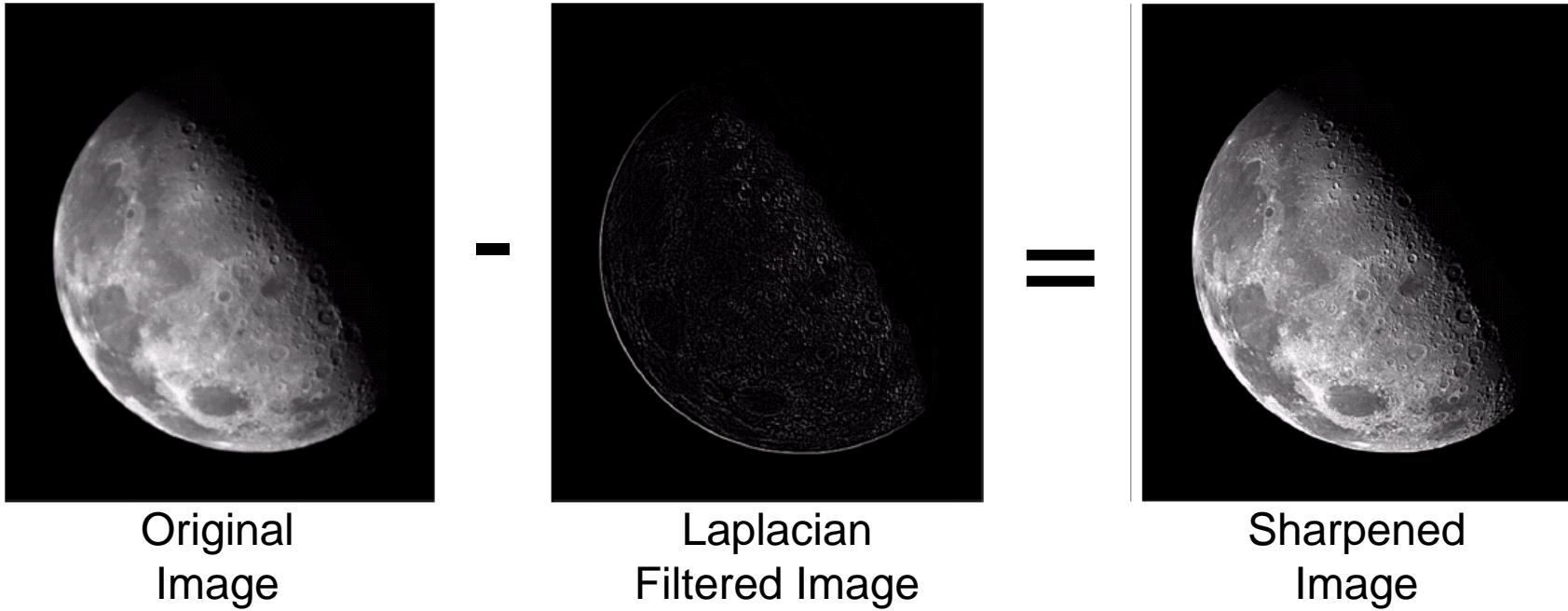
Subtract the Laplacian result from the original image to generate our final sharpened enhanced image

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



Laplacian  
Filtered Image  
Scaled for Display

# Laplacian Image Enhancement



Important to keep in mind which variant of Laplacian is used to decide whether to add or subtract

# Laplacian Image Enhancement



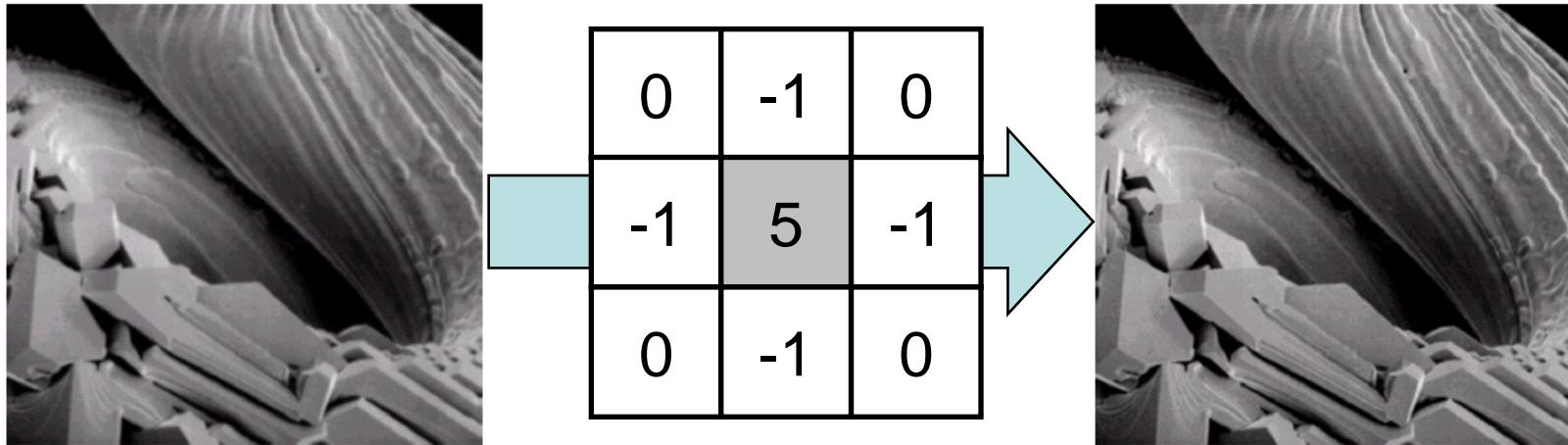
# Simplified Image Enhancement

The entire enhancement can be combined into a single filtering operation

$$\begin{aligned} g(x, y) &= f(x, y) - \nabla^2 f \\ &= f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1) \\ &\quad - 4f(x, y)] \\ &= 5f(x, y) - f(x+1, y) - f(x-1, y) \\ &\quad - f(x, y+1) - f(x, y-1) \end{aligned}$$

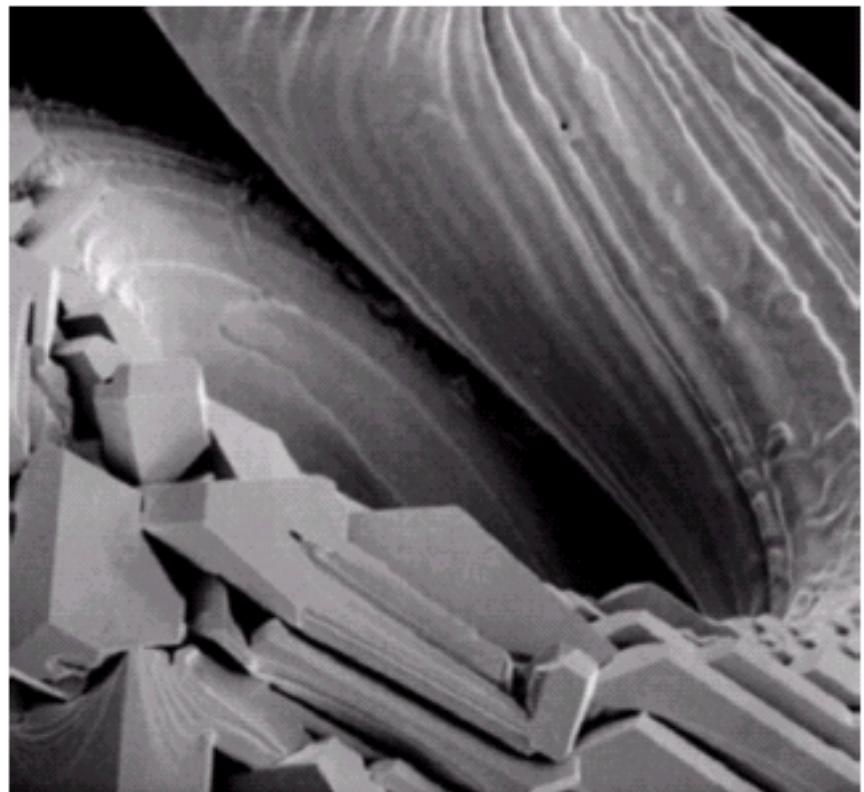
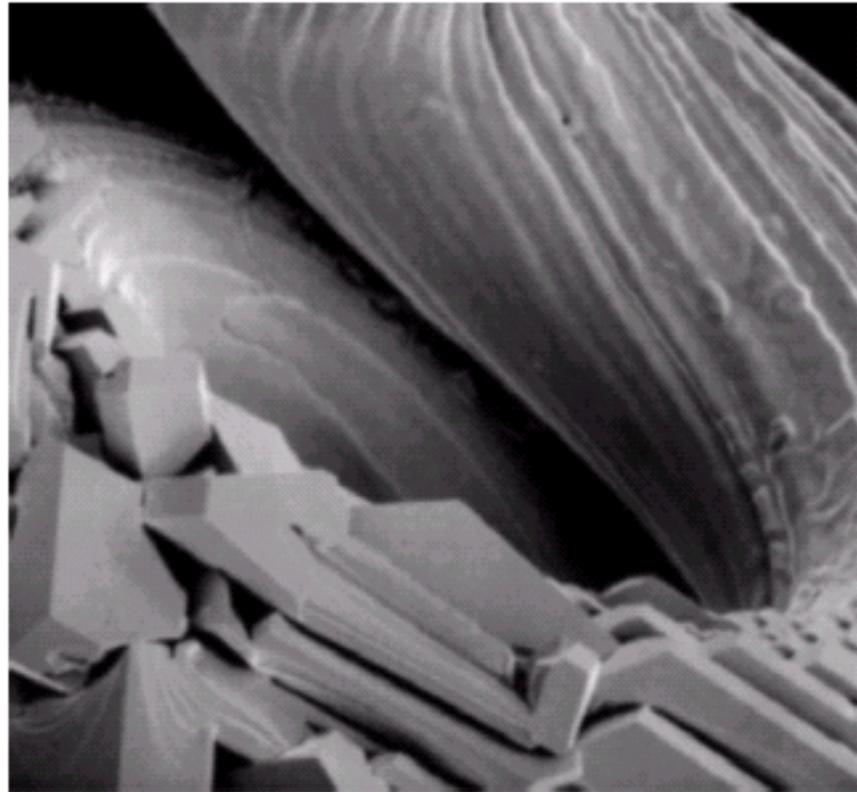
# Simplified Image Enhancement (cont...)

This gives us a new filter which does the whole job for us in one step



Q: What will be the combined filter if the diagonal directions also considered.

# Simplified Image Enhancement (cont...)



# 1<sup>st</sup> Derivative Filtering

First derivative filtering is implemented using the magnitude of the gradient.

For a function  $f(x, y)$  the gradient of  $f$  at coordinates  $(x, y)$  is given as the column vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

# 1<sup>st</sup> Derivative Filtering (cont...)

The magnitude of this vector is given by:

$$\begin{aligned}\nabla f &= \text{mag}(\nabla f) \\ &= [G_x^2 + G_y^2]^{1/2} \\ &= \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2}\end{aligned}$$

For practical reasons this can be simplified as:

$$\nabla f \approx |G_x| + |G_y|$$

# 1<sup>st</sup> Derivative Filtering (cont...)

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

-1	0	0	-1
0	1	1	0

Roberts

-1	1
----	---

-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

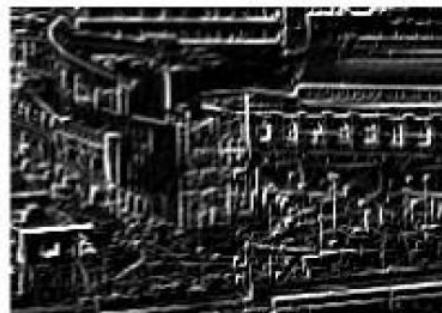
Prewitt

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

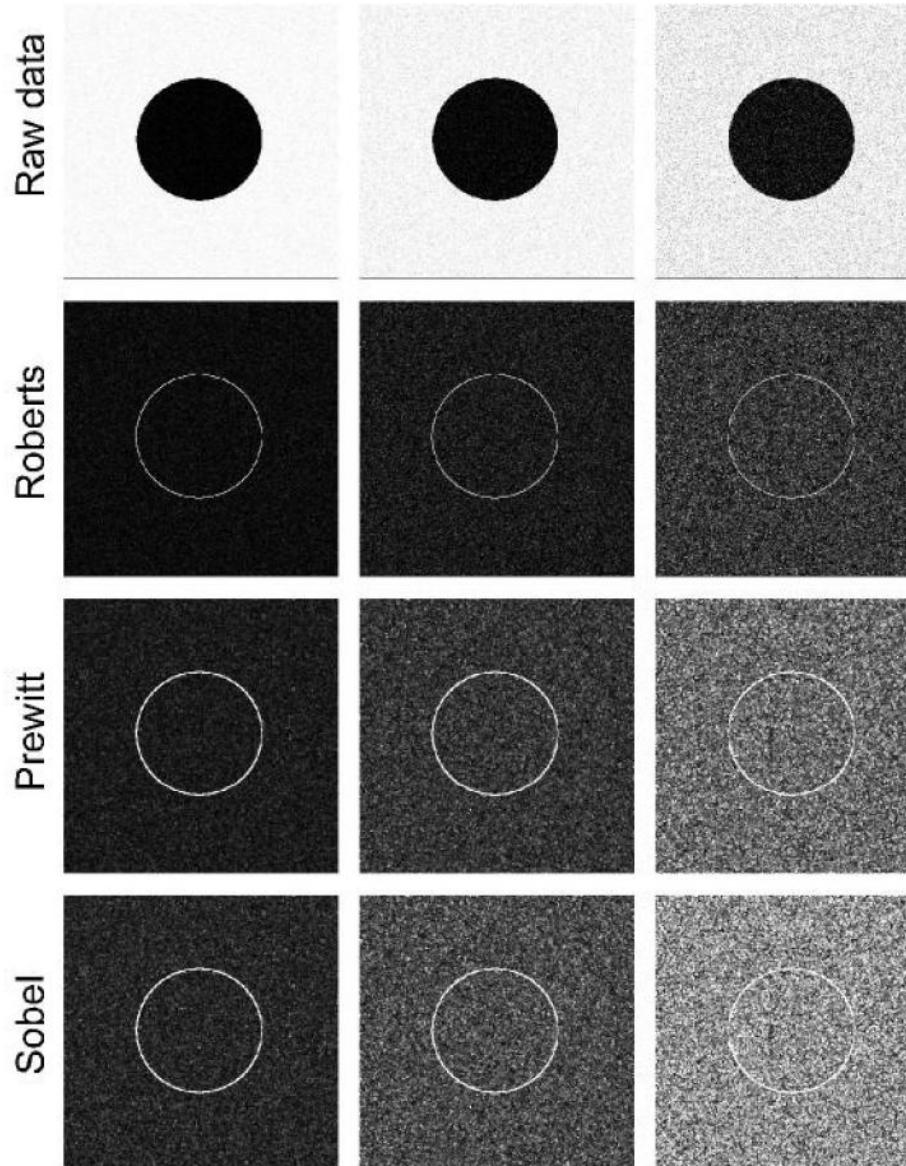
Sobel

Given a 3\*3 region of an image the following edge detection filters can be used

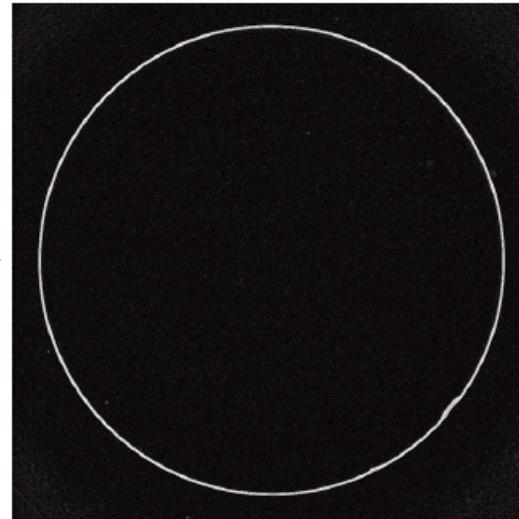
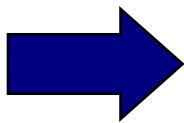
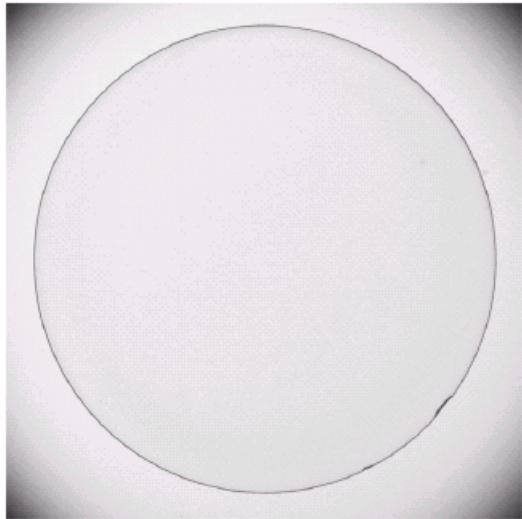
# 1<sup>st</sup> Derivative Filtering (cont...)



# 1<sup>st</sup> Derivative Filtering (cont...)



# Application example



An image of a contact lens which is enhanced in order to find defects (at four and five o'clock in the image) more obvious

Sobel filters are typically used for edge detection

# Combining Spatial Enhancement Methods

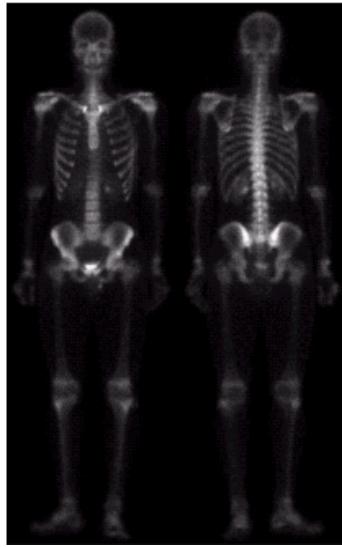
Successful image enhancement is typically not achieved using a single operation

Rather we combine a range of techniques in order to achieve a final result

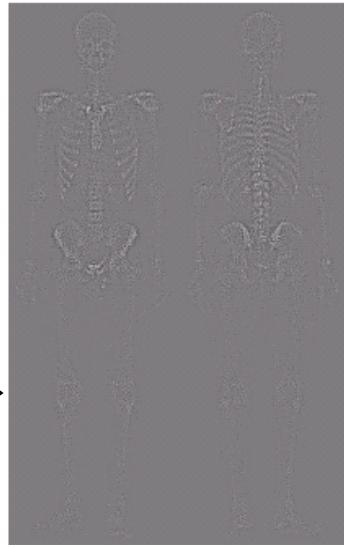
This example will focus on enhancing the bone scan to the right



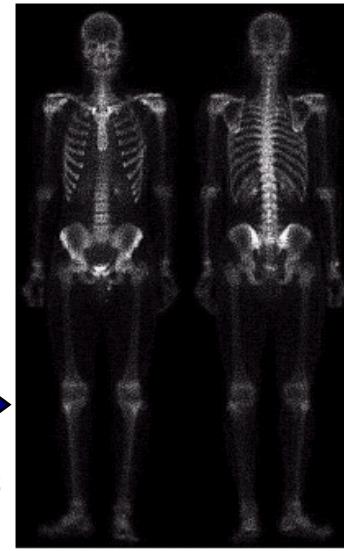
# Combining Spatial Enhancement Methods (cont...)



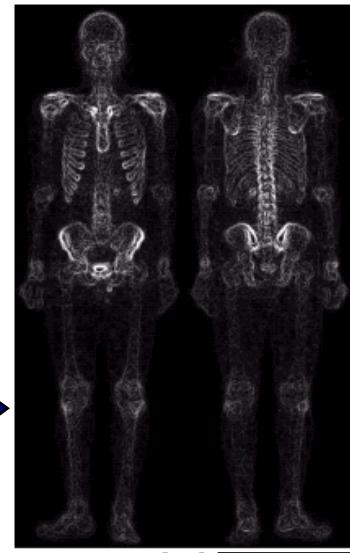
(a) Laplacian filter of bone scan (a)



(b) Sharpened version of bone scan achieved by subtracting (a) and (b)



(c) Sobel filter of bone scan (a)



(d)

# Combining Spatial Enhancement Methods (cont...)

The product of (c) and (e) which will be used as a mask

(e)

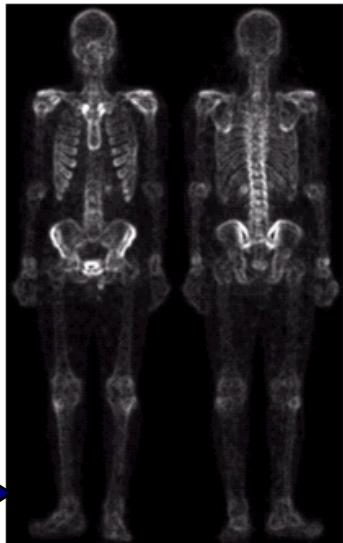


Image (d) smoothed with  
a 5\*5 averaging filter

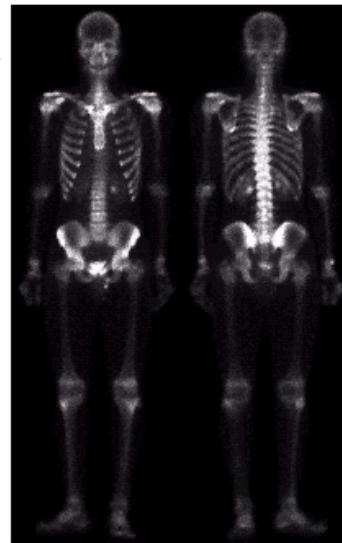
Sharpened image  
which is sum of (a)  
and (f)

(f)

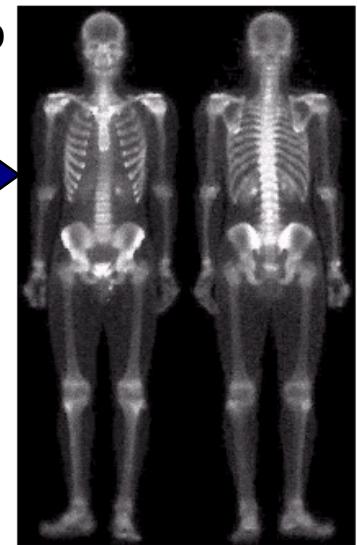


Result of applying a  
power-law trans. to  
(g)

(g)

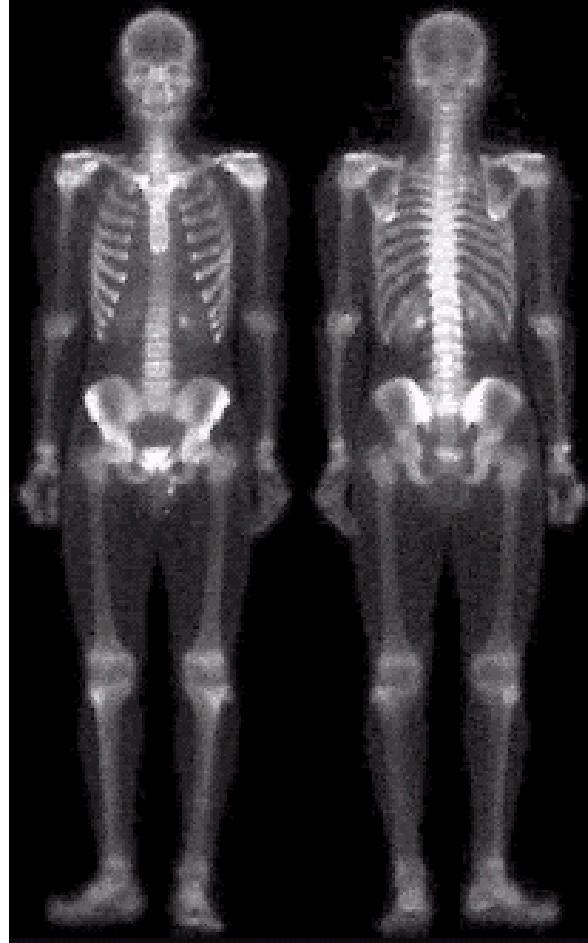
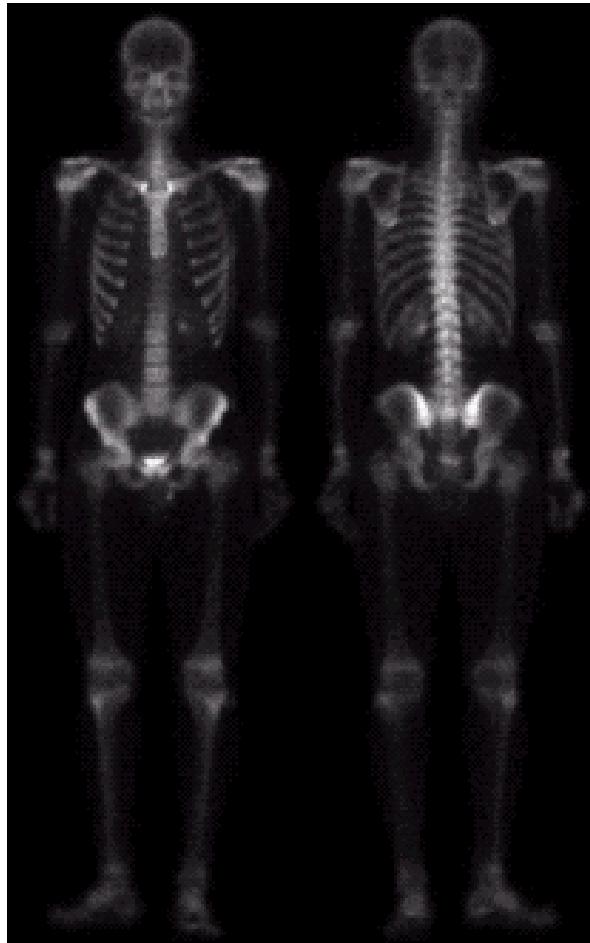


(h)



# Combining Spatial Enhancement Methods (cont...)

Compare the original and final images



# **Image & Video Processing**

---

## **Frequency Domain Filtering**

# Contents

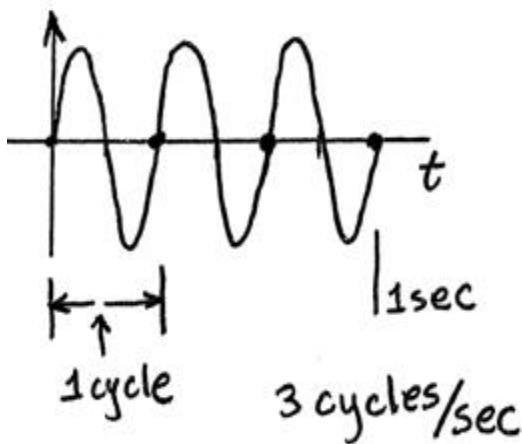
---

In this lecture we will introduce frequency domain/space and look at mathematical tools to go from time/spatial domain to frequency domain

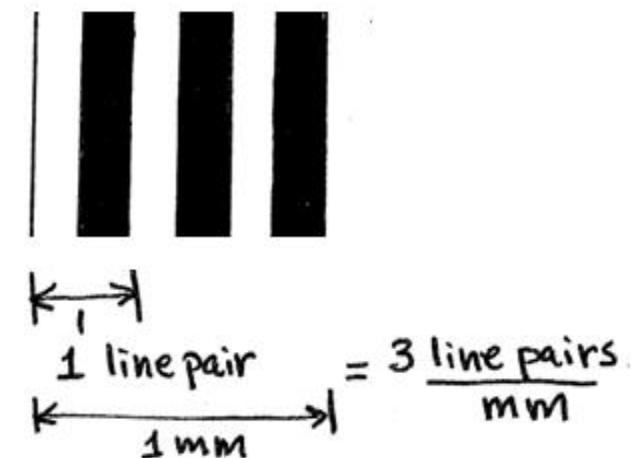
- What is Frequency domain?
- Frequency in Images
- Why Frequency analysis?
- The Fourier series
- The Fourier transform

# Frequency Domain/Space

- The term frequency comes up a lot in physics, as some variation in time, describing the characteristics of some periodic motion or behaviour.



UNIT OF MEASUREMENT	
Hz	line pairs / mm
"cycles/sec	pixels / mm



Frequency in the  
Time domain (1D)

Frequency in the  
Spatial domain (2D)

# Frequency Domain/Space

---

- A signal (wave in time or image in spatial domain) can be decomposed or separated into a sum of sinusoids of different frequencies, amplitude and phase.
- *1D Audio Example:* Consider a complicated sound played on a piano or a guitar. We can describe this sound in two related ways:
  - **Temporal Domain:** Sample the amplitude of the sound many times a second, which gives an approximation to the sound as a function of time
  - **Frequency Domain:** Analyse the sound in terms of the pitches of the notes, or the amplitude of each frequency (fundamental plus harmonics) which make up the sound up.

# Frequency in Image?

---

- Image can be represented in two ways:
  - **Spatial Domain**: Brightness along a line can be recorded as a set of values measured at equally spaced distances apart (represented as 2D array of pixel measurements)
  - **Frequency Domain**: as a set of spatial frequency values/component (represented as 2D grid of spatial frequencies)
- In short, frequency in image has to do with intensity (brightness or color) variation across the image, i.e. it is a function of spatial coordinates, rather than time.
- Example: If an image represented in frequency space has *high* frequencies then it means that the image has sharp edges or details

# Frequency in Image?

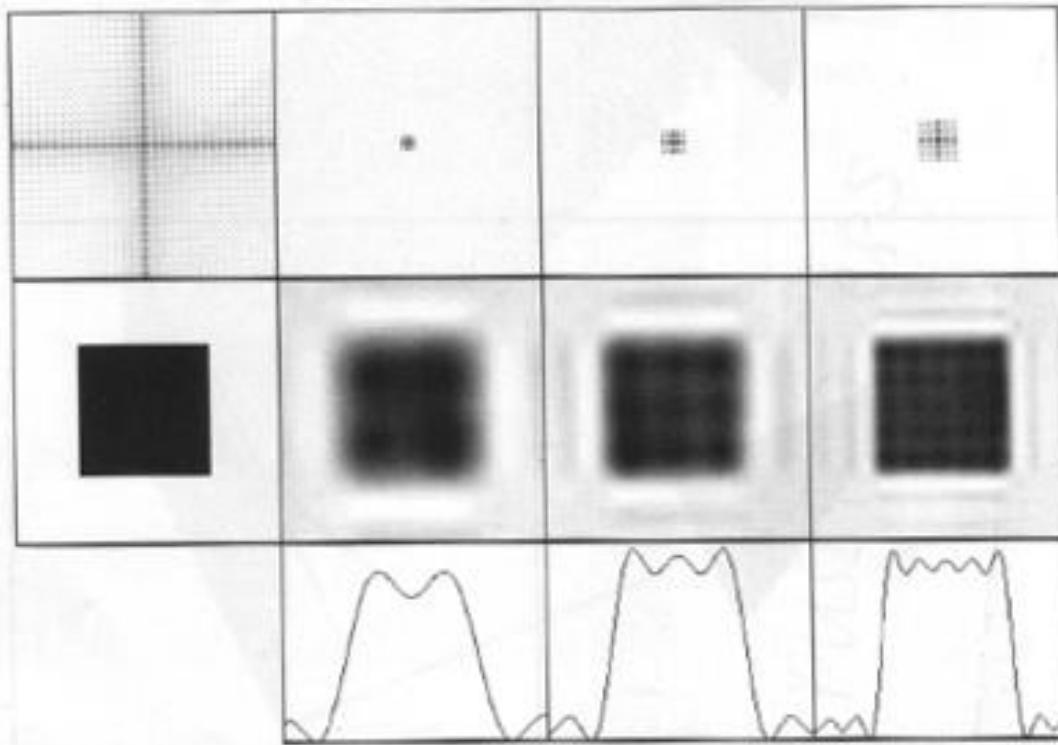


Fig 1. Images in the spatial domain are in the middle row, and their frequency space are shown on the top row. The bottom row shows the varying brightness of the horizontal line through the center of an image.

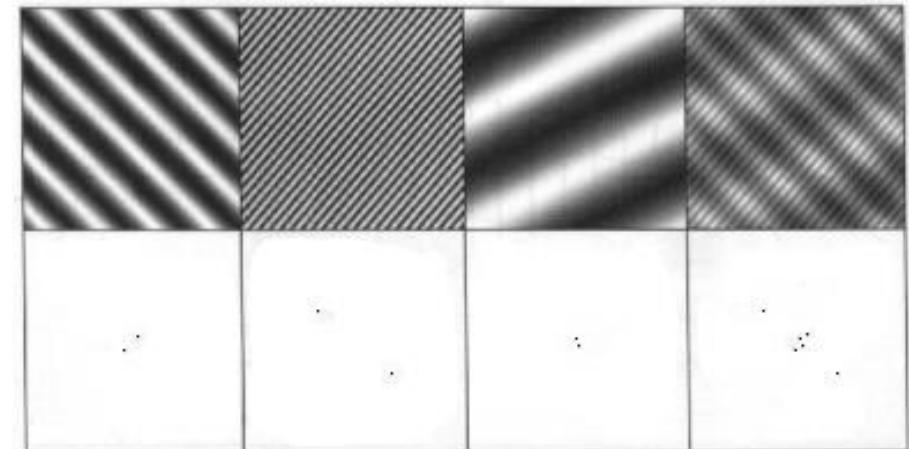
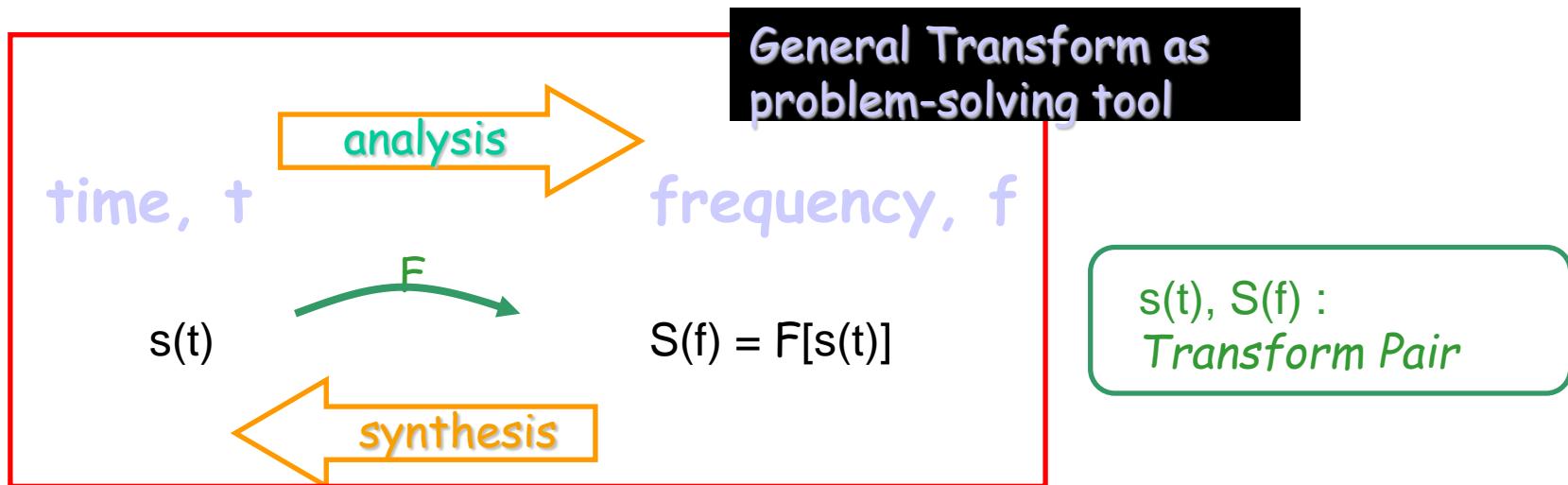


Fig 2. Images with perfectly sinusoidal variations in brightness: The first three images are represented by two dots. You can easily see that the position and orientation of those dots have something to do with what the original image looks like. The 4th image is the sum of the first three.

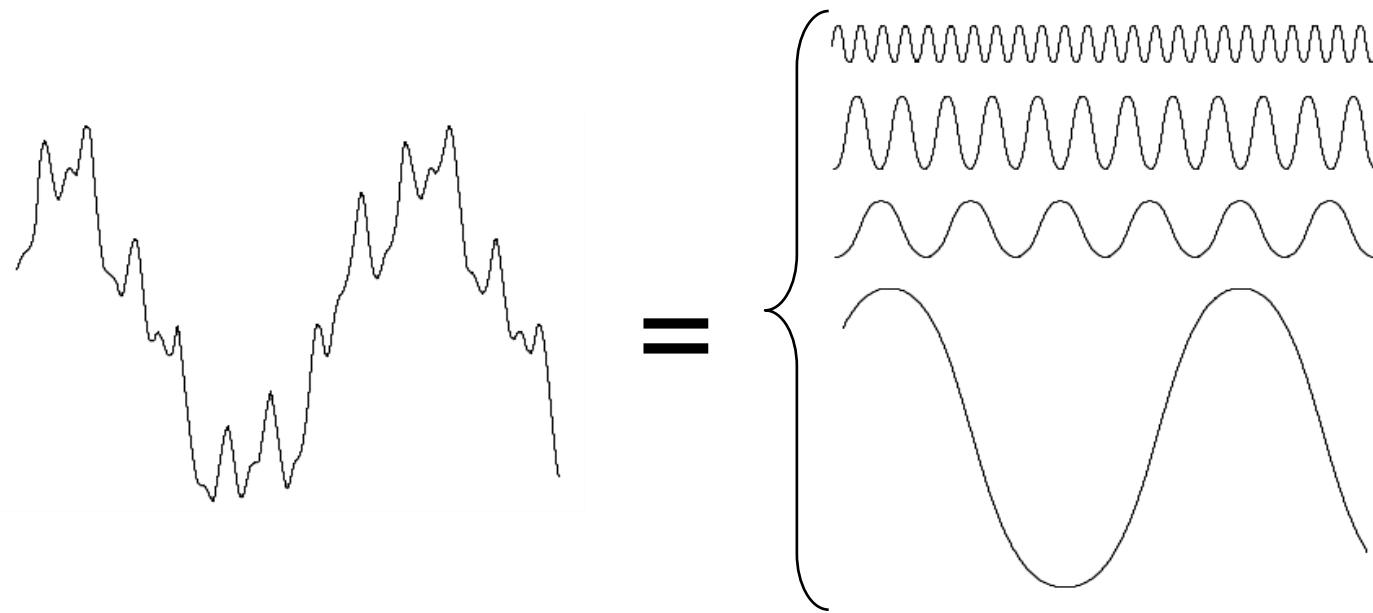
# Why Frequency analysis?

- Makes large filtering operations much faster.
- Powerful & complementary to time/spatial domain analysis techniques.
- Several transforms that makes it easy to go forwards and backwards from the spacial domain to the frequency space: Fourier, Laplace, wavelet, etc.



# The Big Idea

**Fourier Series and Transforms** – given by mathematician **Joseph Fourier**.  
One of the most important mathematical theories in modern engineering



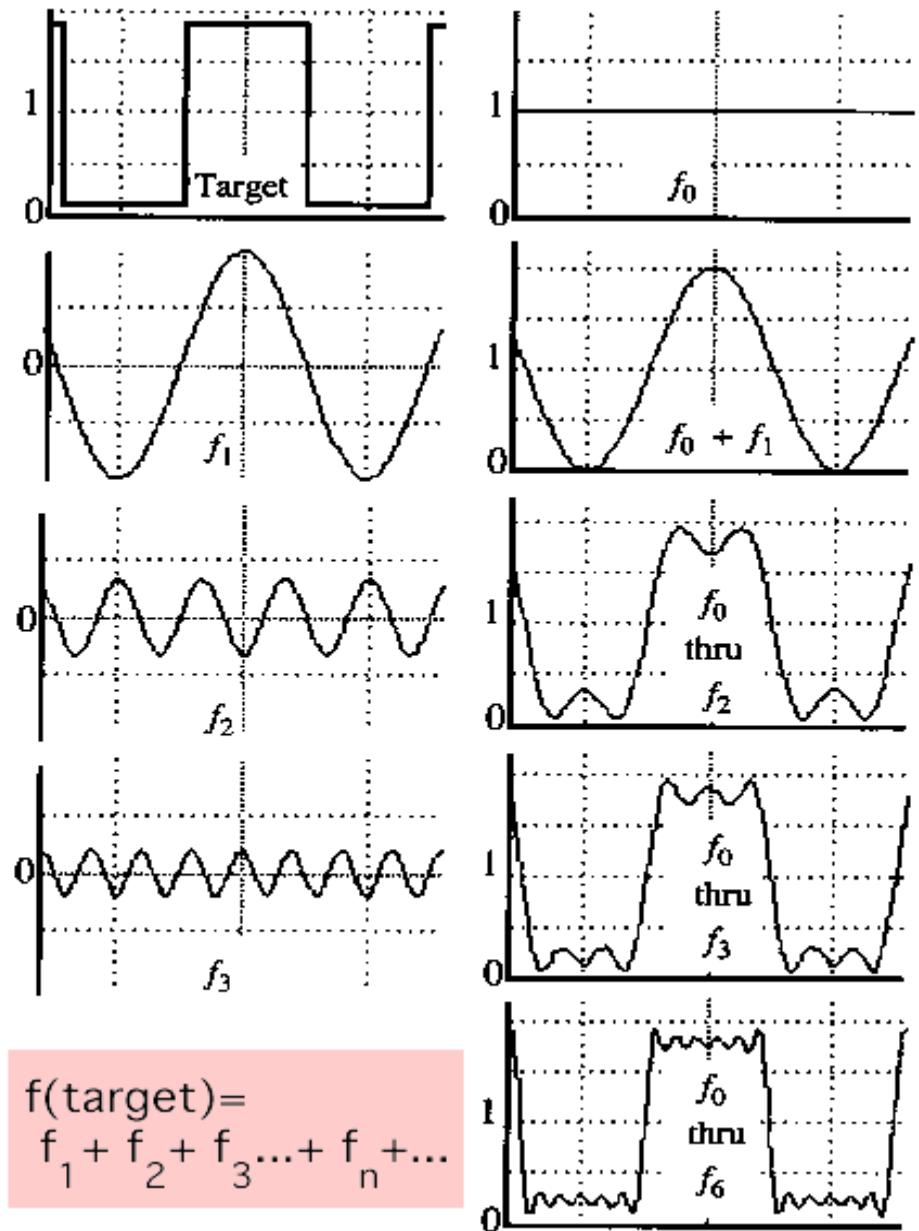
Any function that periodically repeats itself can be expressed as a sum of sines and cosines of different frequencies each multiplied by a different coefficient – a *Fourier series*

# A Sum of Sinusoids

- Our building block:

$$A \sin(\omega x + \phi)$$

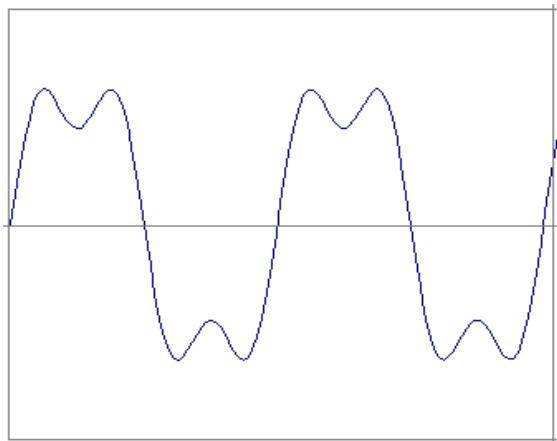
- Add enough of them to get any signal  $f(x)$  you want!
- Which one encodes the coarse vs. fine structure of the signal?



# Time and Frequency

---

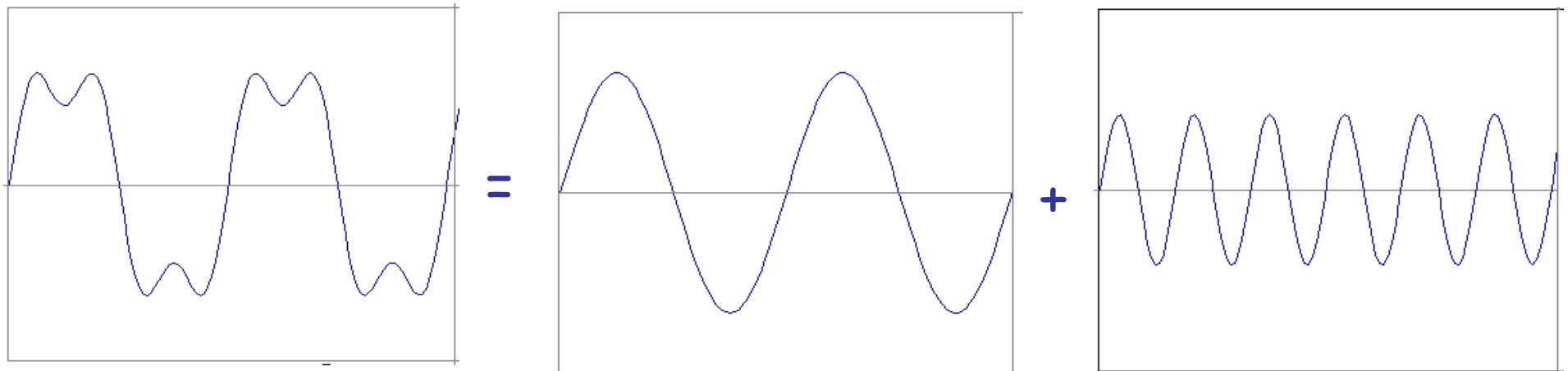
- example :  $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi (3f) t)$



# Time and Frequency

---

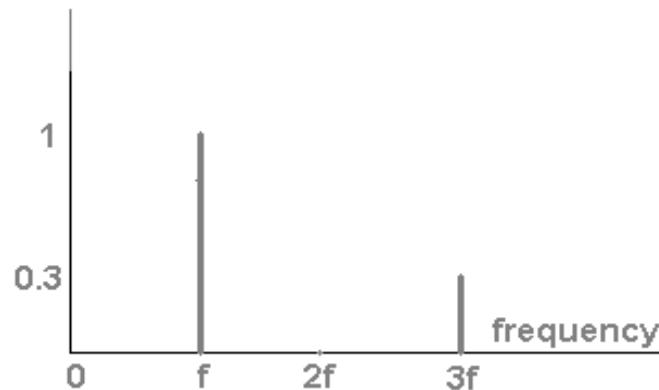
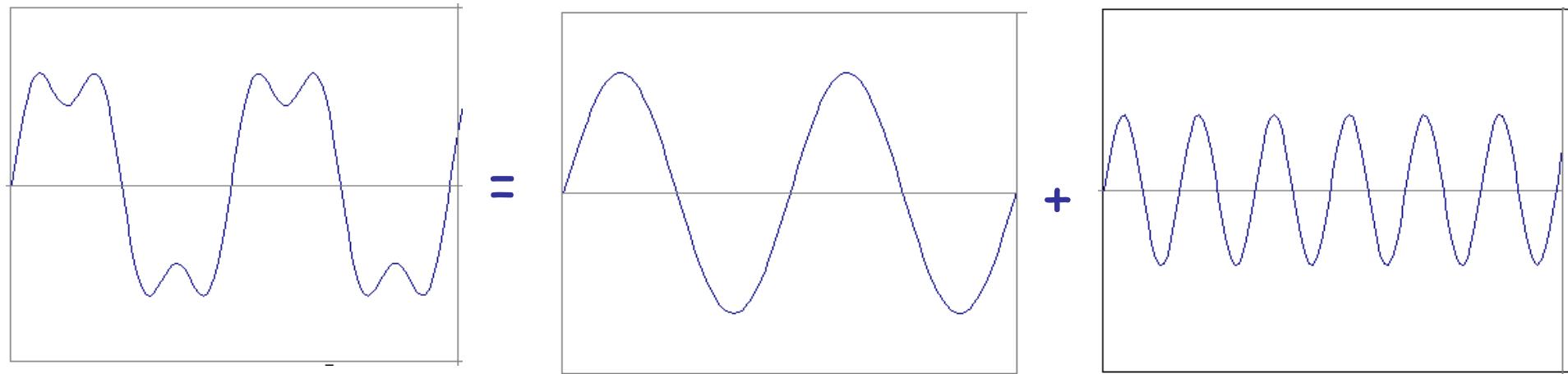
- example :  $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi (3f) t)$



# Frequency Spectra

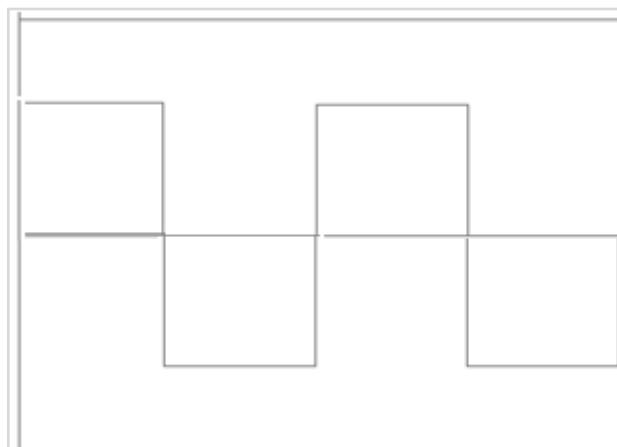
---

- example :  $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi (3f) t)$

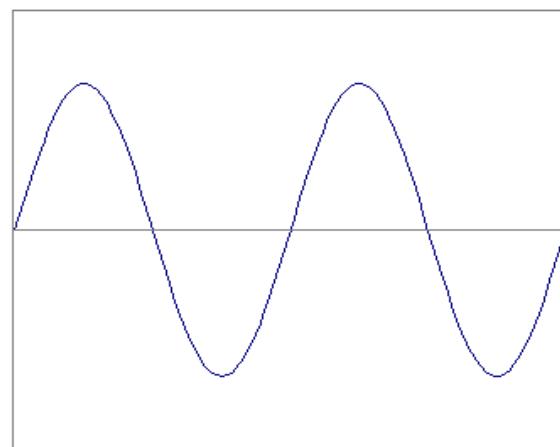


# Frequency Spectra

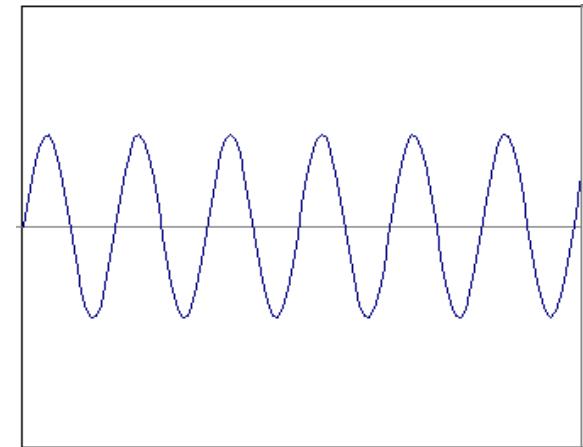
---



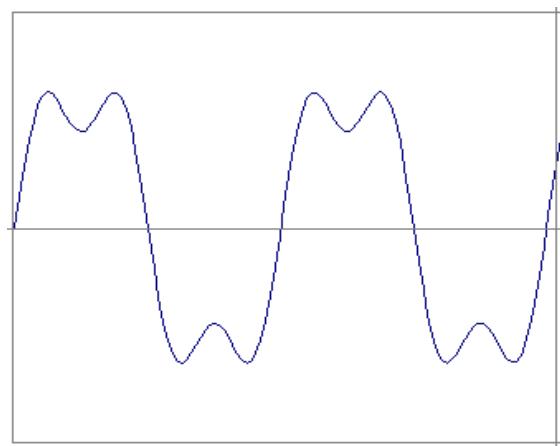
=



+

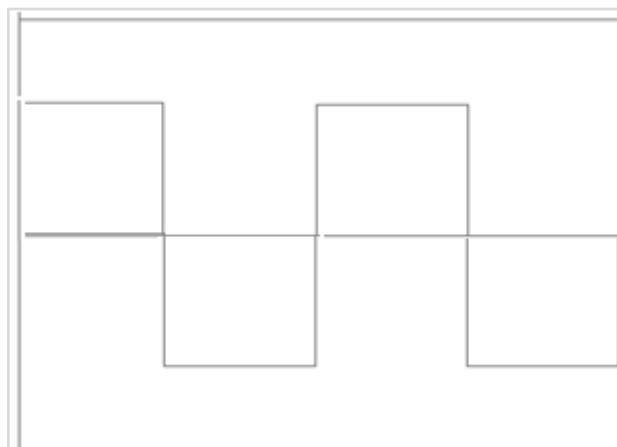


=

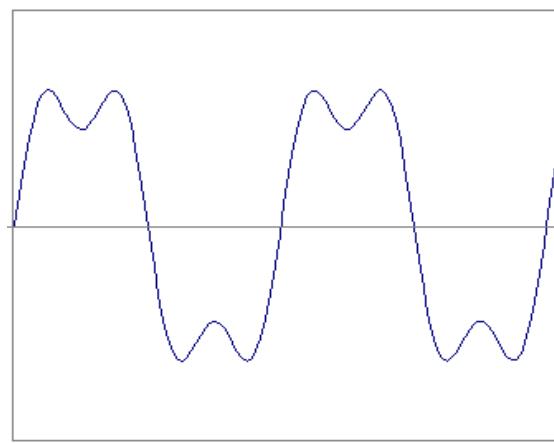


# Frequency Spectra

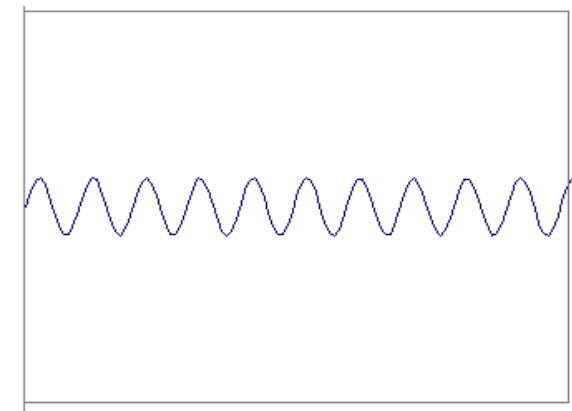
---



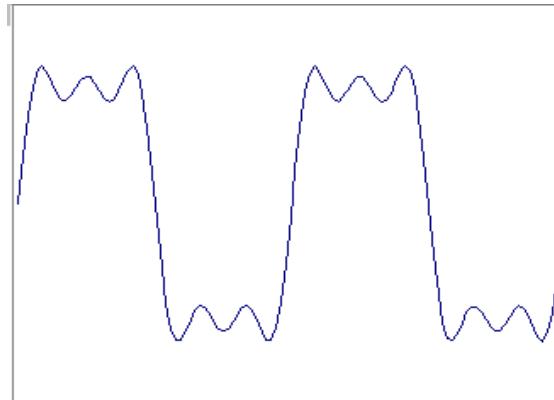
=



+

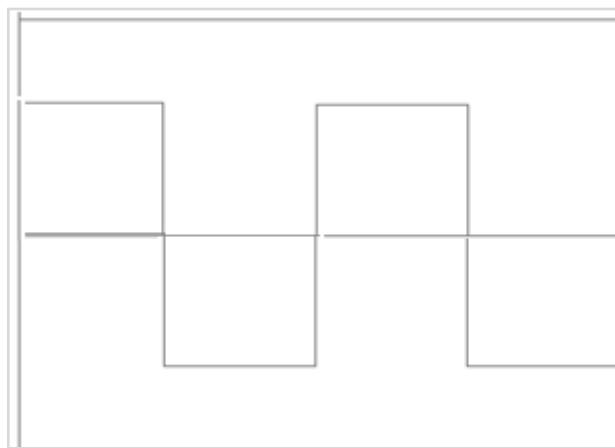


=

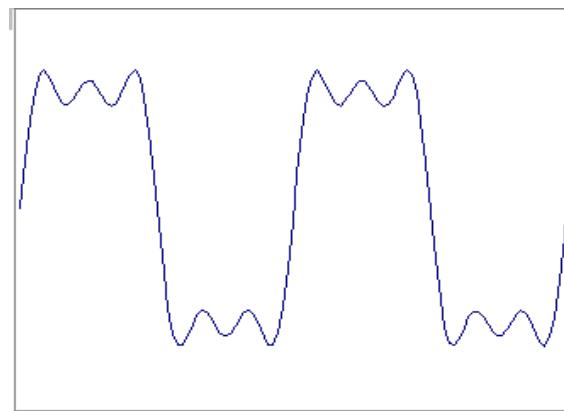


# Frequency Spectra

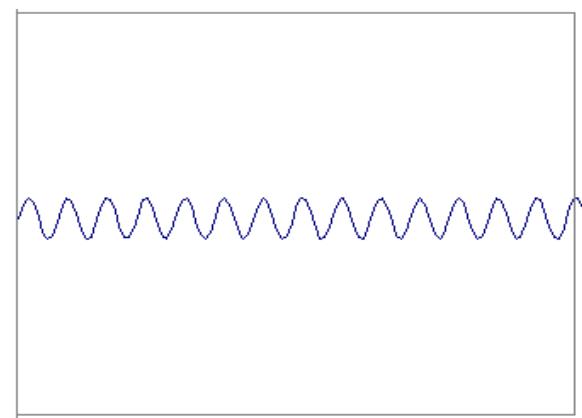
---



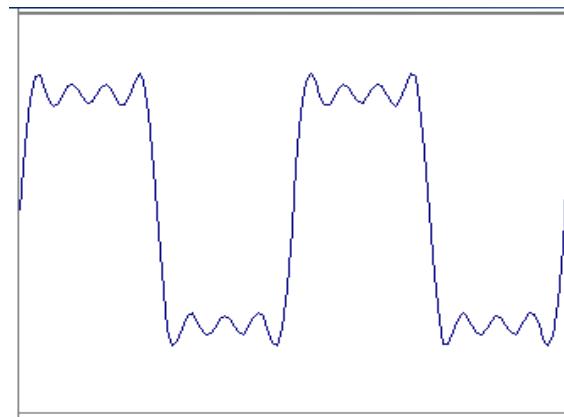
=



+

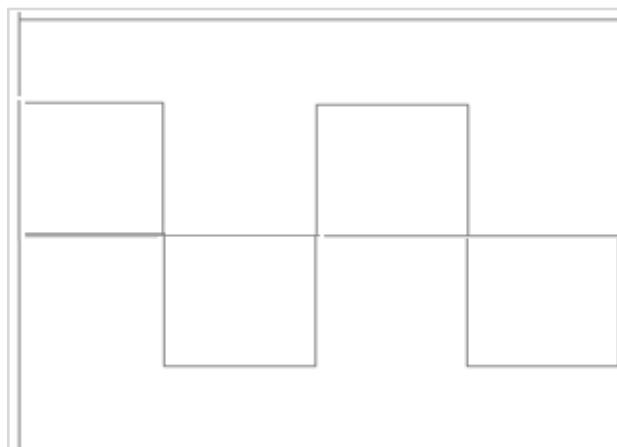


=

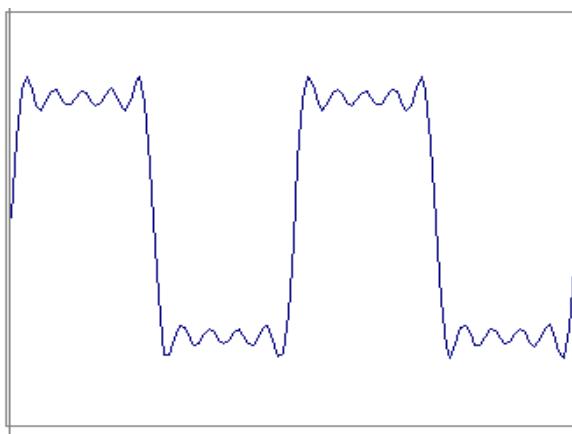


# Frequency Spectra

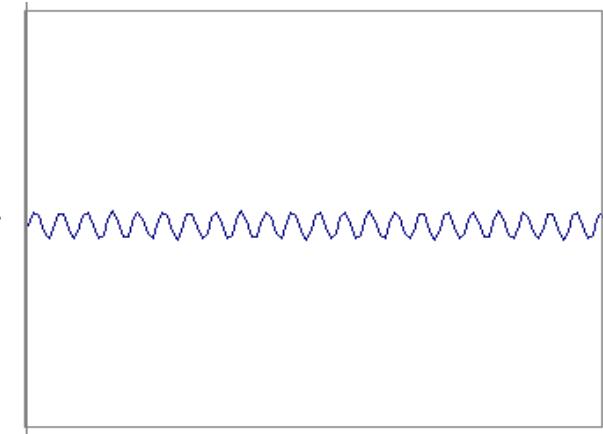
---



=



+

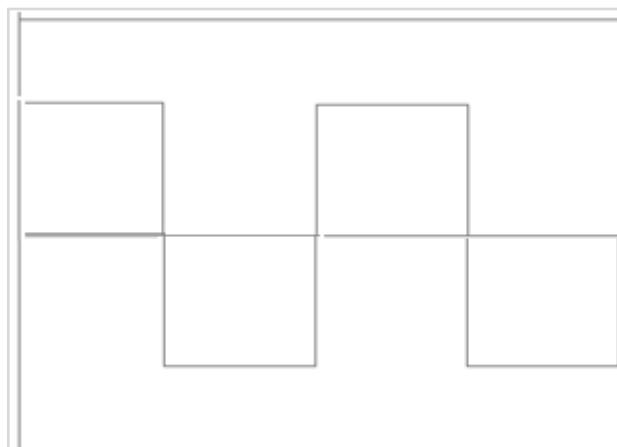


=



# Frequency Spectra

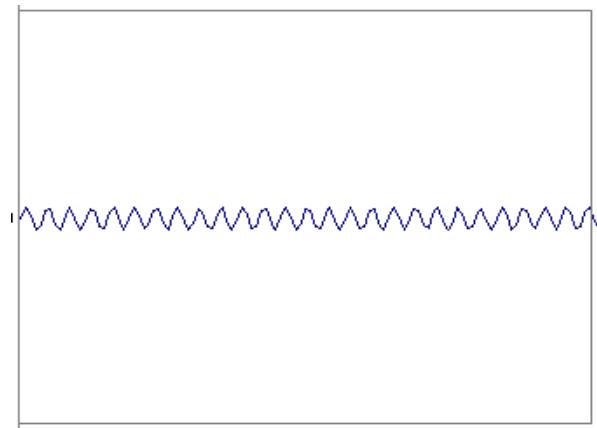
---



=



+

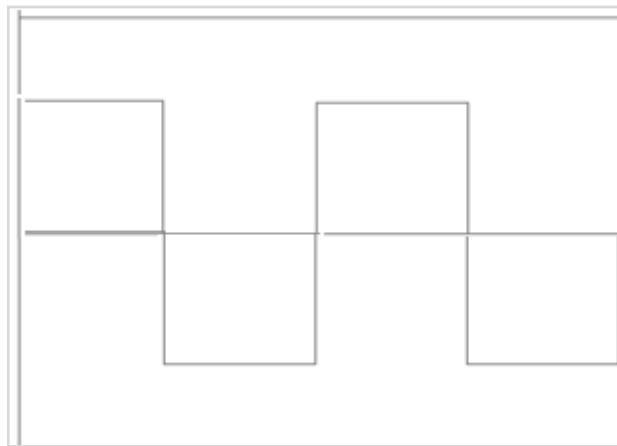


=



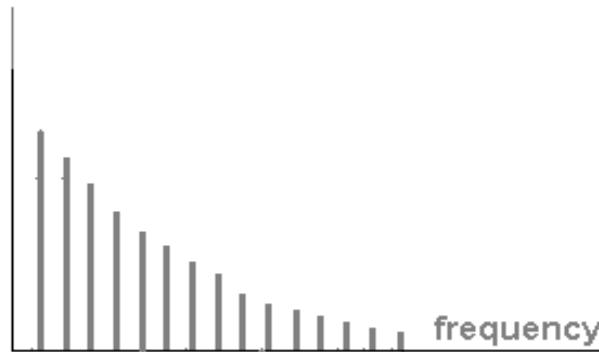
# Frequency Spectra

---



=

$$A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt)$$



# Fourier Series

---

- A general representation of function  $f(t)$  that is periodic with period  $T$ , by Fourier series as:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + \sum_{n=1}^{\infty} b_n \sin(n\omega_0 t)$$

$$a_0 = \frac{2}{T} \int_{t_0}^{t_0+T} f(t) dt$$

$$a_n = \frac{2}{T} \int_{t_0}^{t_0+T} f(t) \cos n\omega_0 t dt \quad n = 1, 2, \dots$$

$$b_n = \frac{2}{T} \int_{t_0}^{t_0+T} f(t) \sin n\omega_0 t dt \quad n = 1, 2, \dots$$

# Fourier Series

---

$$\begin{aligned}f(t) &= \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega_0 t) + \sum_{n=1}^{\infty} b_n \sin(n\omega_0 t) \\&= \frac{a_0}{2} + \sum_{n=1}^{\infty} \sqrt{a_n^2 + b_n^2} \left( \frac{a_n}{\sqrt{a_n^2 + b_n^2}} \cos \omega_n t + \frac{b_n}{\sqrt{a_n^2 + b_n^2}} \sin \omega_n t \right)\end{aligned}$$

$$= \frac{a_0}{2} + \sum_{n=1}^{\infty} \sqrt{a_n^2 + b_n^2} (\cos \theta_n \cos \omega_n t + \sin \theta_n \sin \omega_n t)$$

$$= C_0 + \sum_{n=1}^{\infty} C_n \cos(\omega_n t - \theta_n)$$

$$f(t) = C_0 + \sum_{n=1}^{\infty} C_n \cos(\omega_n t - \theta_n)$$

harmonic amplitude

phase angle

$$C_n = \sqrt{a_n^2 + b_n^2}$$

$$\theta_n = \tan^{-1} \left( \frac{b_n}{a_n} \right)$$

# Complex Form of the Fourier Series

---

$$e^{jn\omega_0 t} = \cos n\omega_0 t + j \sin n\omega_0 t , \quad e^{-jn\omega_0 t} = \cos n\omega_0 t - j \sin n\omega_0 t$$

$$\cos n\omega_0 t = \frac{1}{2} (e^{jn\omega_0 t} + e^{-jn\omega_0 t}) , \quad \sin n\omega_0 t = \frac{1}{2j} (e^{jn\omega_0 t} - e^{-jn\omega_0 t}) = -\frac{j}{2} (e^{jn\omega_0 t} - e^{-jn\omega_0 t})$$

$$\begin{aligned} f(t) &= \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos n\omega_0 t + \sum_{n=1}^{\infty} b_n \sin n\omega_0 t \\ &= \frac{a_0}{2} + \frac{1}{2} \sum_{n=1}^{\infty} a_n (e^{jn\omega_0 t} + e^{-jn\omega_0 t}) - \frac{j}{2} \sum_{n=1}^{\infty} b_n (e^{jn\omega_0 t} - e^{-jn\omega_0 t}) \\ &= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[ \frac{1}{2} (a_n - jb_n) e^{jn\omega_0 t} + \frac{1}{2} (a_n + jb_n) e^{-jn\omega_0 t} \right] \\ &= c_0 + \sum_{n=1}^{\infty} [c_n e^{jn\omega_0 t} + c_{-n} e^{-jn\omega_0 t}] \\ &= \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t} \end{aligned}$$

# Complex Form of the Fourier Series

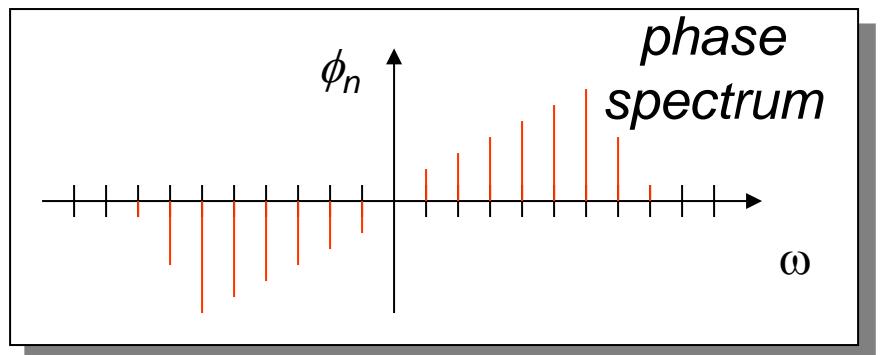
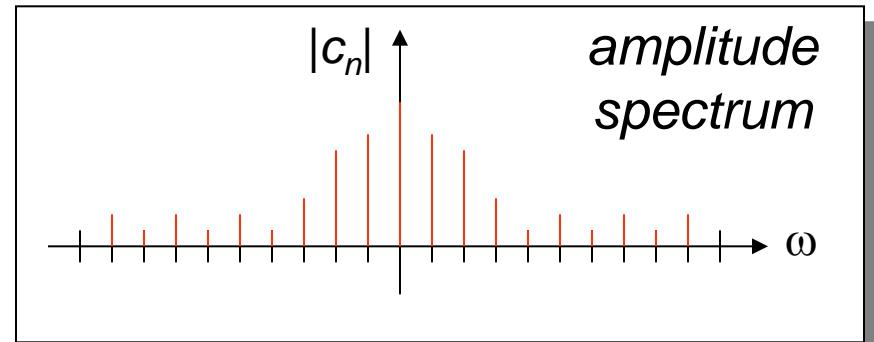
$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t}$$

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-jn\omega_0 t} dt$$

If  $f(t)$  is real,

$$\rightarrow c_{-n} = c_n^* \quad |c_n| = |c_{-n}| = \frac{1}{2} \sqrt{a_n^2 + b_n^2}$$

$$\phi_n = \tan^{-1} \left( -\frac{b_n}{a_n} \right)$$



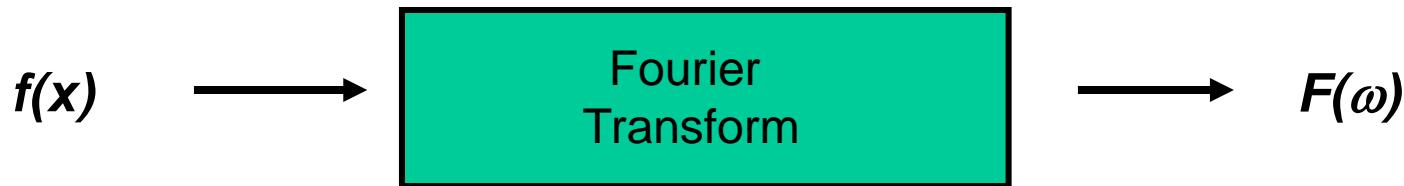
# Fourier Transform

---

- We have seen that periodic signals can be represented with the Fourier series
- Can **aperiodic signals** be analyzed in terms of frequency components?
- Yes, and the Fourier transform provides the tool for this analysis.
- **Aperiodic signals** can be treated as periodic with period tending to infinity in the limit
- The major difference w.r.t. the discrete line spectra of periodic signals is that the **spectra of aperiodic signals** are continuous.

# Fourier Transform

- We want to understand the frequency  $\omega$  of our signal. So, let's reparametrize the signal by  $\omega$  instead of  $x$ :



- For every  $\omega$  from 0 to inf,  $F(\omega)$  holds the amplitude  $A$  and phase  $\phi$  of the corresponding sine

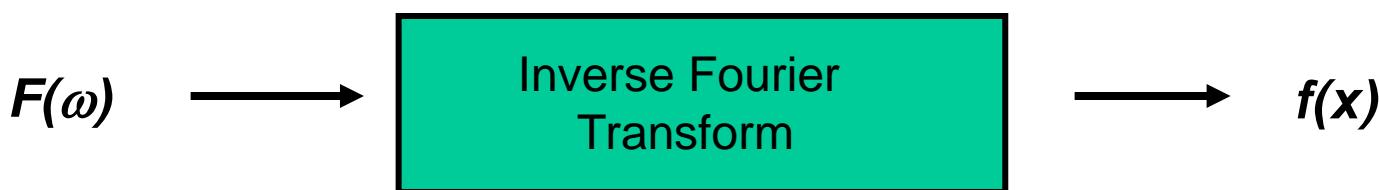
$$A \sin(\omega x + \phi)$$

- How can  $F$  hold both? Complex number trick!

$$F(\omega) = R(\omega) + j I(\omega)$$

$$A = \pm \sqrt{R(\omega)^2 + I(\omega)^2}$$

$$\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$$



# Fourier Transform – more formally

---

Represent the signal as an infinite weighted sum of an infinite number of sinusoids

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx$$

Note:  $e^{j\theta} = \cos \theta + j \sin \theta \quad j = \sqrt{-1}$

Arbitrary function  $\longrightarrow$  Single Analytic Expression

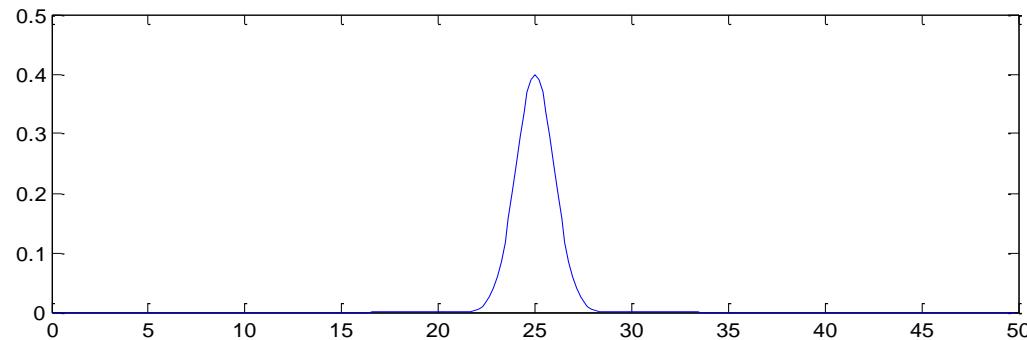
Spatial Domain ( $x$ )  $\longrightarrow$  Frequency Domain ( $u$ )  
(Frequency Spectrum  $F(u)$ )

Inverse Fourier Transform (IFT)

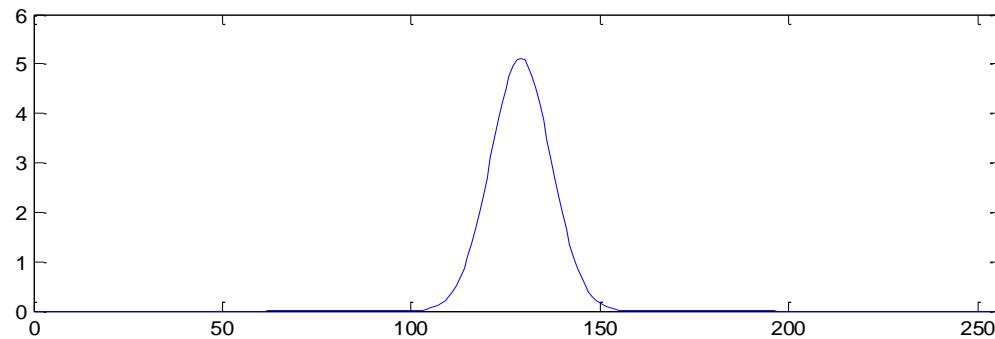
$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

# Famous Fourier Transforms

---



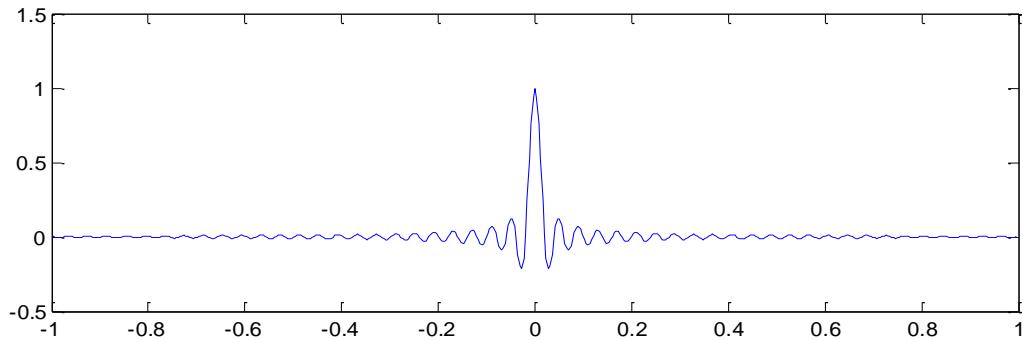
Gaussian



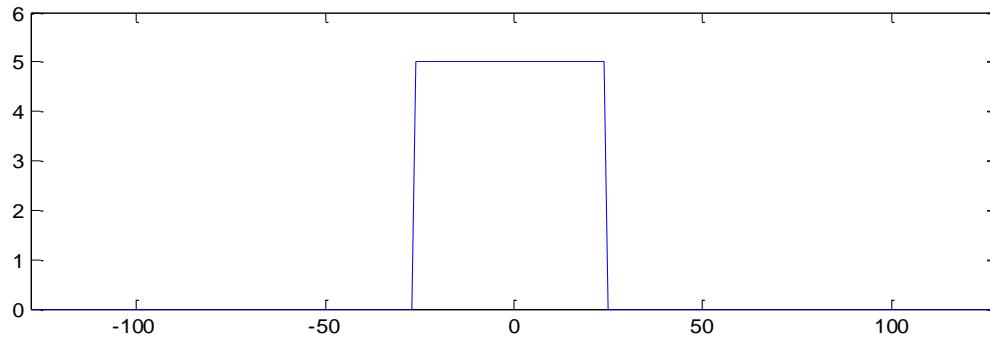
Gaussian

# Famous Fourier Transforms

---



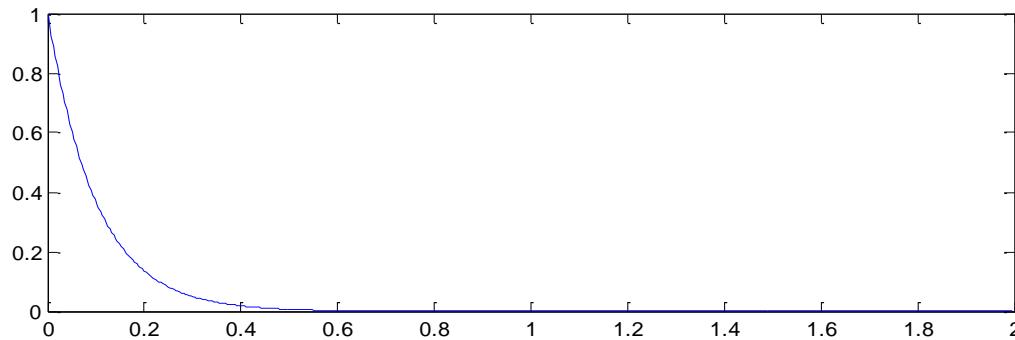
Sinc function



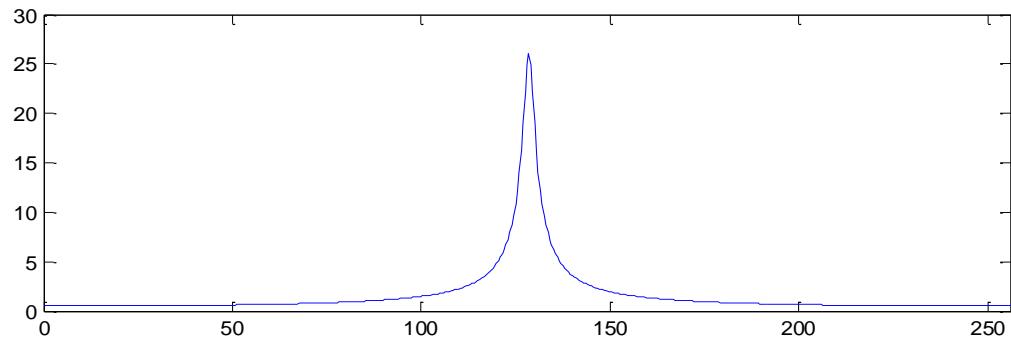
Square wave

# Famous Fourier Transforms

---



Exponential



Lorentzian

# **Image & Video Processing**

---

## **Frequency Domain Filtering**

# Contents

---

In this lecture we will look at DFT and image enhancement in the frequency domain

- Discrete Fourier Transform (2D)
- DFT and Images
- DFT Properties
- Image Processing in the frequency domain
- Frequency domain filters for enhancement
  - Image smoothing
  - Image sharpening

# The Discrete Fourier Transform in 2D

---

The *Discrete Fourier Transform* of  $f(x, y)$ , for  $x = 0, 1, 2 \dots M-1$  and  $y = 0, 1, 2 \dots N-1$ , denoted by  $F(u, v)$ , is given by the equation:

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

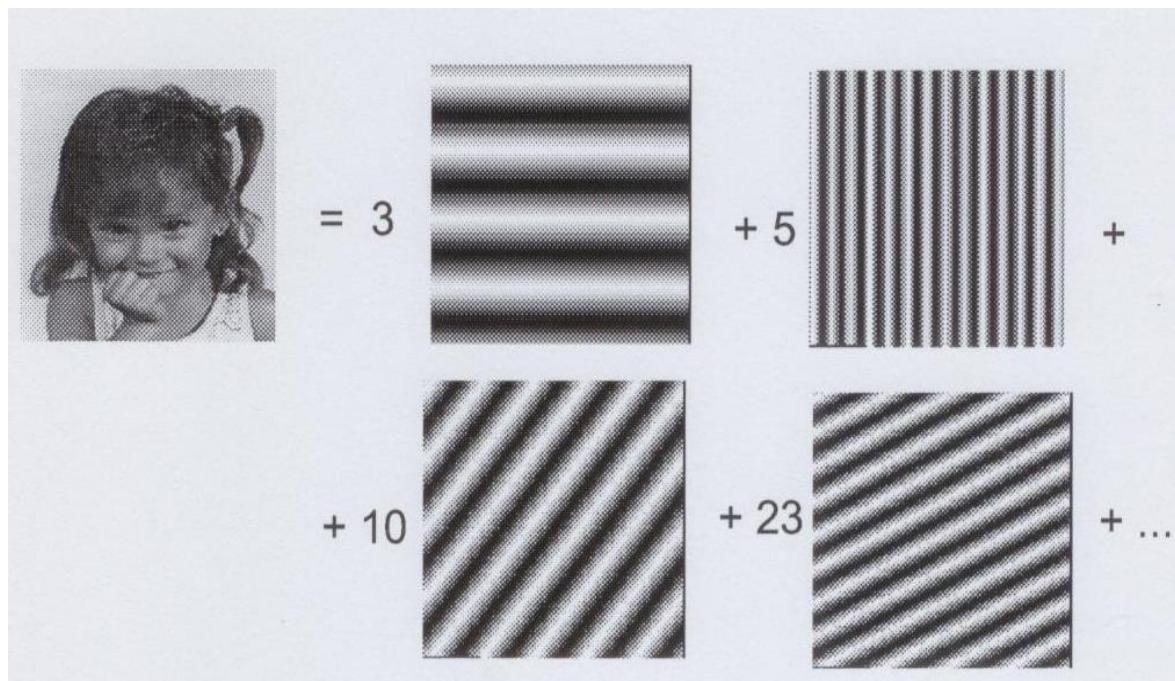
for  $u = 0, 1, 2 \dots M-1$  and  $v = 0, 1, 2 \dots N-1$ .

Inverse Discrete Fourier Transform is given by equation:

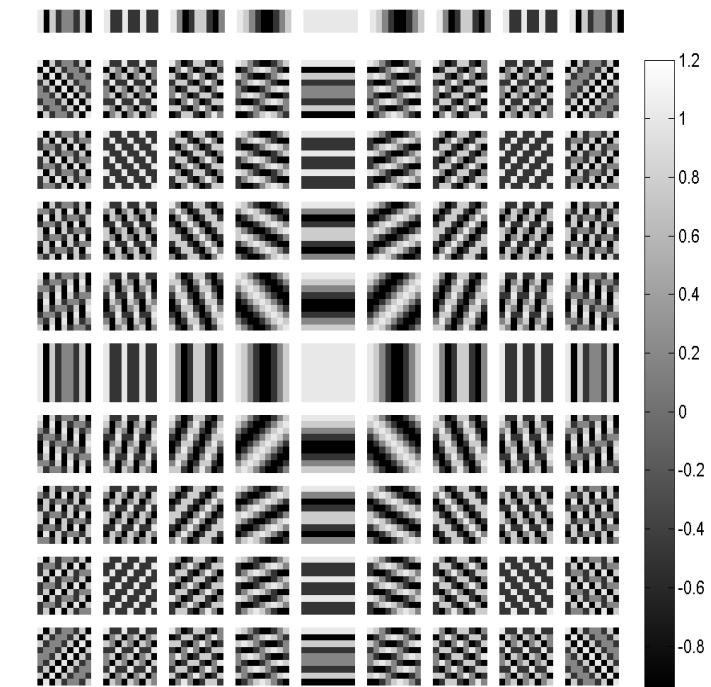
$$f(x, y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}$$

# DFT & Images

Interpretation: 2D cos/sin functions

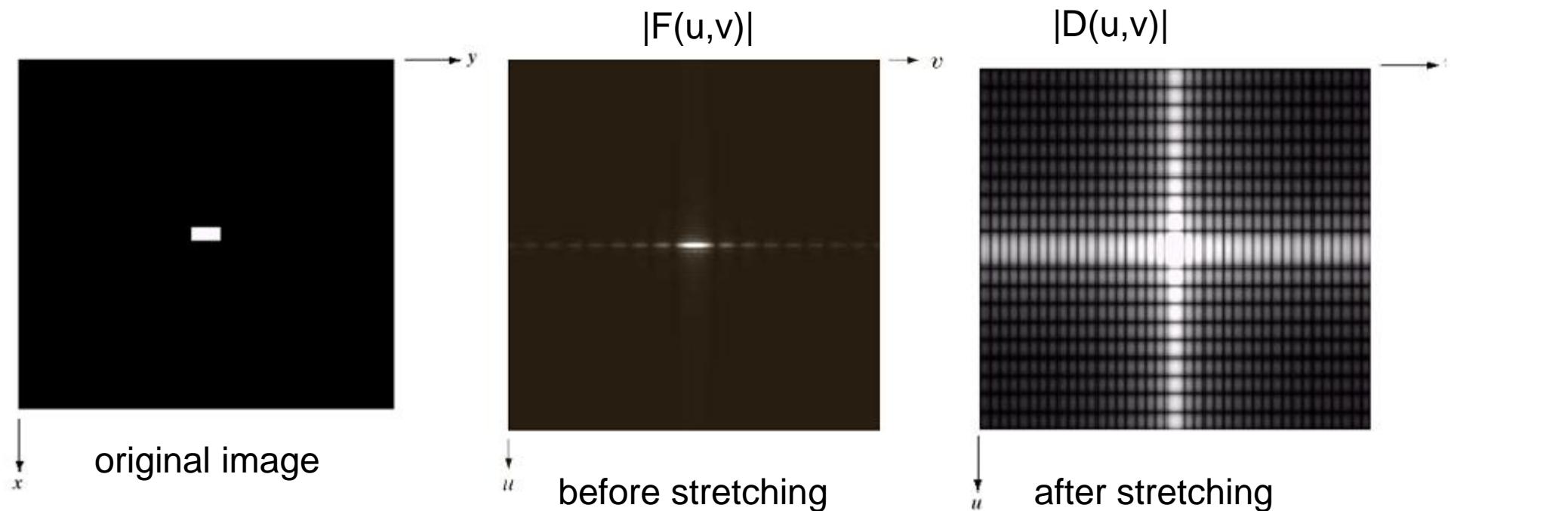


2-D Fourier basis



# Visualizing DFT

- Typically, we visualize  $|F(u,v)|$
- The dynamic range of  $|F(u,v)|$  is typically very large
- Apply log transformation:  $D(u, v) = c \log(1 + |F(u, v)|)$

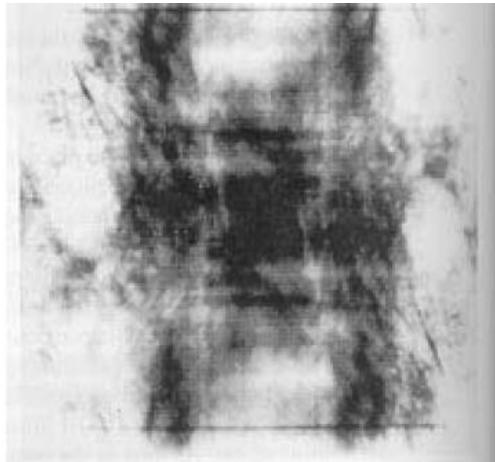
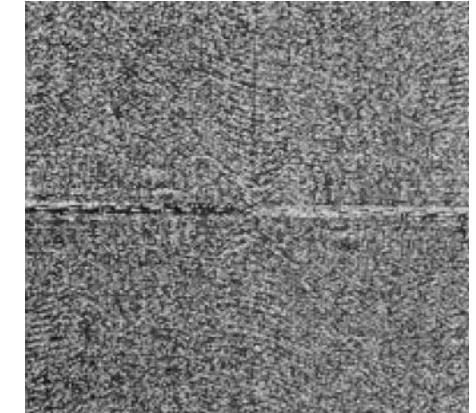
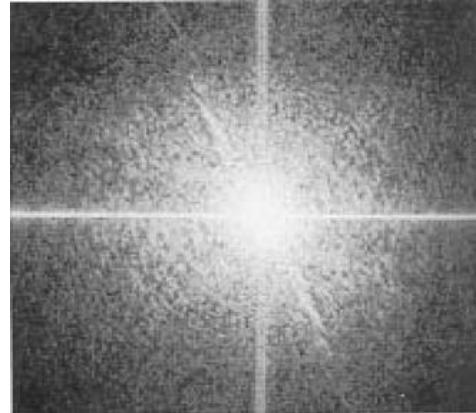


# Magnitude and Phase of DFT

- What is more important?

magnitude

phase



Reconstructed  
image using  
magnitude  
only



Reconstructed  
image using  
Phase only

# another example: amplitude vs. phase

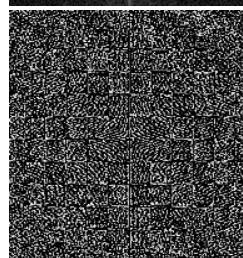
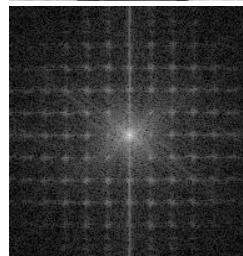
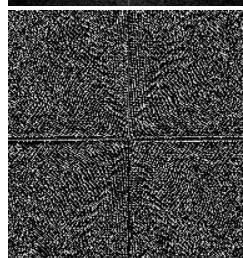
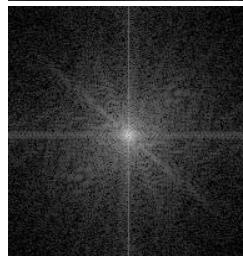
$A = \text{"Aron"}$

$FA = \text{fft2}(A)$

$\log(\text{abs}(FA))$

$\text{angle}(FA)$

$\text{ifft2}(\text{abs}(FA), \text{angle}(FP))$



$P = \text{"Phyllis"}$

$FP = \text{fft2}(P)$

$\log(\text{abs}(FP))$

$\text{angle}(FP)$

$\text{ifft2}(\text{abs}(FP), \text{angle}(FA))$

# DFT Properties: Separability

---

- The 2D DFT can be computed using 1D transforms **only**:

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

kernel is  
separable: 
$$e^{-j2\pi(\frac{ux+vy}{N})} = e^{-j2\pi(\frac{ux}{N})} e^{-j2\pi(\frac{vy}{N})}$$

- Let's set: 
$$\sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(\frac{vy}{N})} = F(x, v)$$

- Then: 
$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} e^{-j2\pi(\frac{ux}{N})} F(x, v)$$

# Fast Fourier Transform

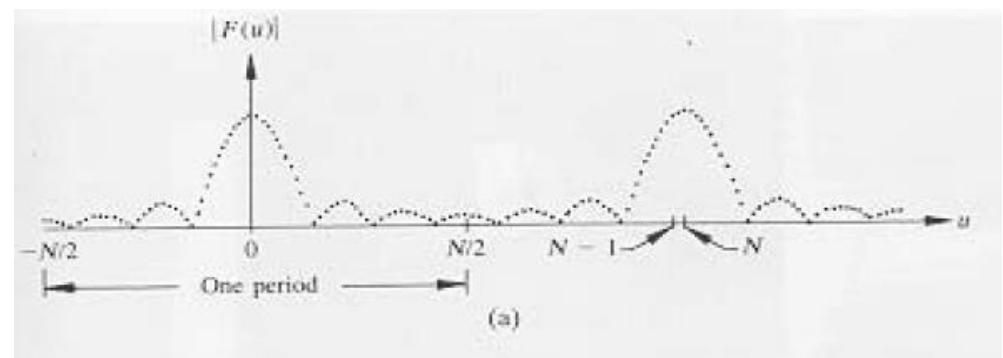
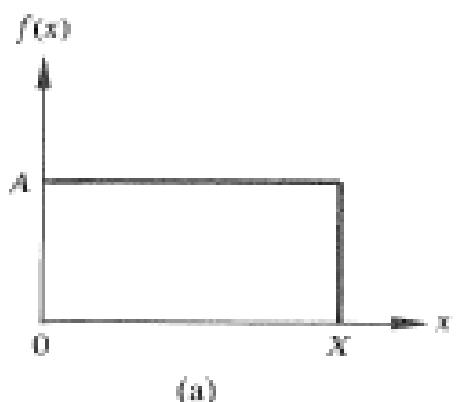
---

- The development of the *Fast Fourier Transform (FFT)* algorithm allows the Fourier transform to be carried out in a reasonable amount of time
- Based on Divide and conquer strategy and exploits property (like Separability) of the 2D transform
- Reduces the amount of time required to perform a Fourier transform by a factor of  **$MN \log MN$**  times
- Discrete, 2-D Fourier & inverse Fast Fourier transforms are implemented in Matlab as `fft2` and `ifft2`

# DFT Properties: Periodicity

- The DFT and its inverse are periodic with period N

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



# DFT Properties: Translation

---

$$f(x,y) \longleftrightarrow F(u,v)$$

- Translation in spatial domain:

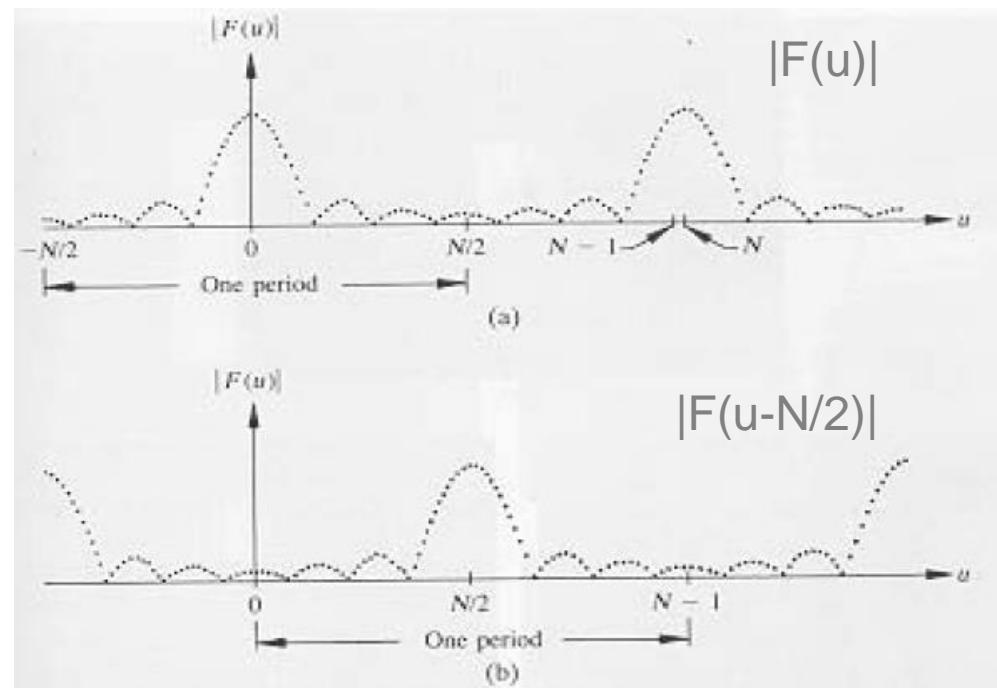
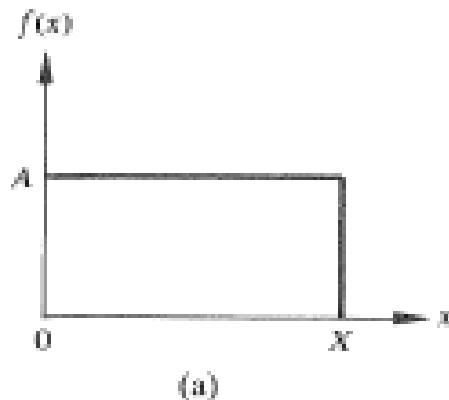
$$f(x - x_0, y - y_0) \longleftrightarrow F(u, v)e^{-j2\pi(\frac{ux_0+vy_0}{N})}$$

- Translation in frequency domain:

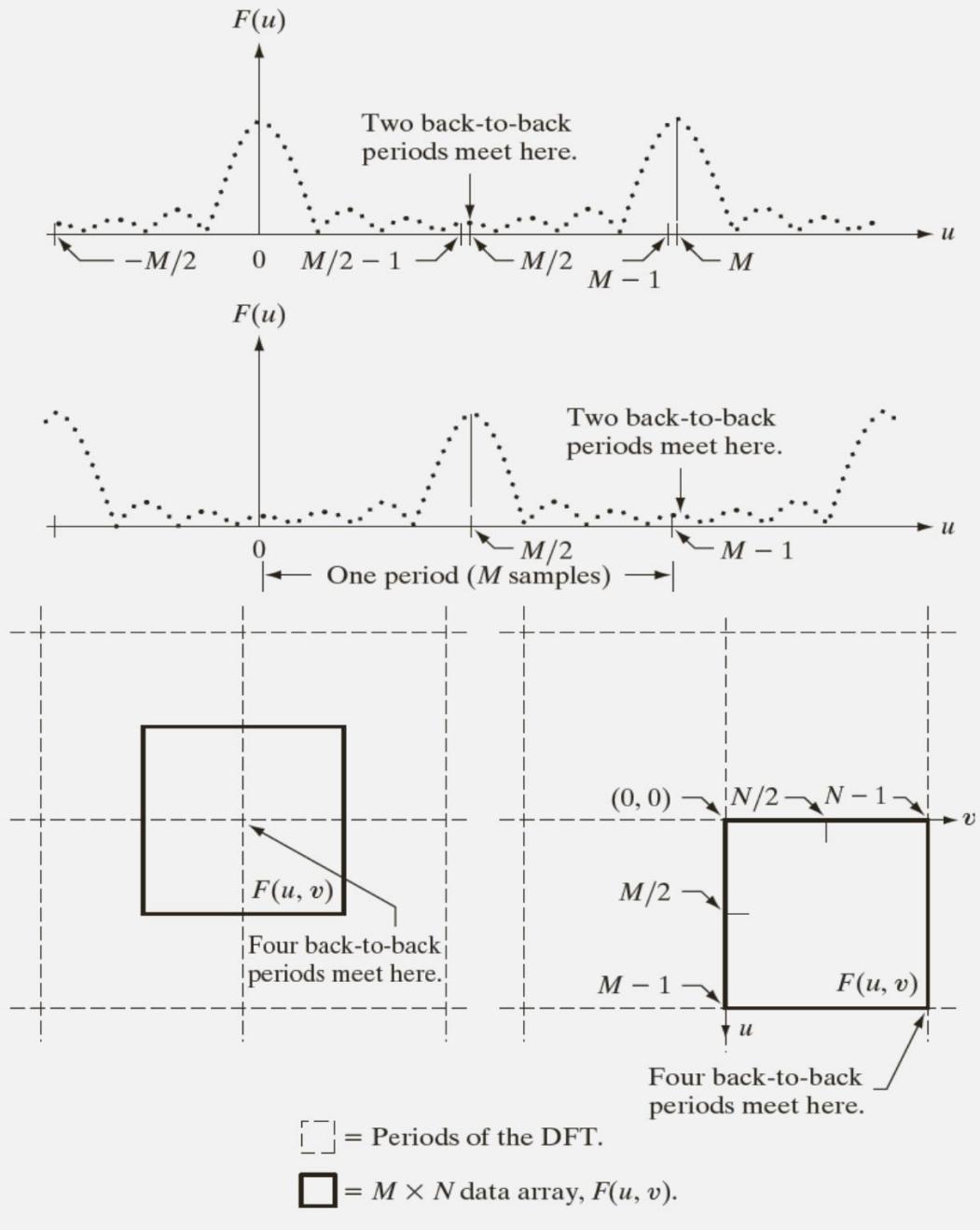
$$f(x, y)e^{j2\pi(\frac{u_0x+v_0y}{N})} \longleftrightarrow F(u - u_0, v - v_0)$$

# DFT Properties: Translation (cont'd)

- To show a **full period**, we need to translate the origin of the transform at  $u=N/2$  (or at  $(N/2, N/2)$  in 2D)



## Periodicity & Translation of the transform

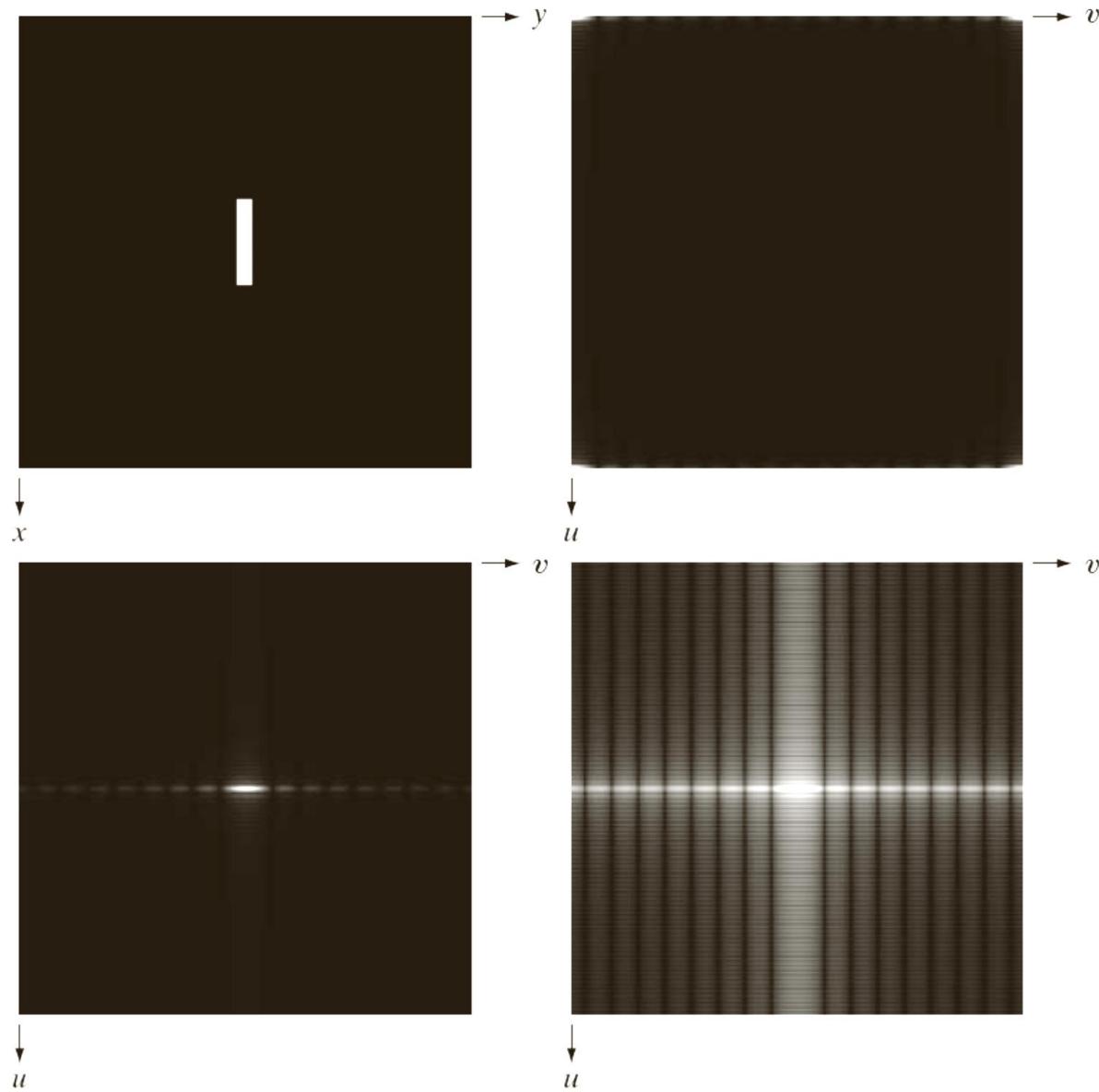


a  
b  
c d

**FIGURE 4.23**

Centering the Fourier transform.  
 (a) A 1-D DFT showing an infinite number of periods.  
 (b) Shifted DFT obtained by multiplying  $f(x)$  by  $(-1)^x$  before computing  $F(u)$ .  
 (c) A 2-D DFT showing an infinite number of periods. The solid area is the  $M \times N$  data array,  $F(u, v)$ , obtained with Eq. (4.5-15). This array consists of four quarter periods.  
 (d) A Shifted DFT obtained by multiplying  $f(x, y)$  by  $(-1)^{x+y}$  before computing  $F(u, v)$ . The data now contains one complete, centered period, as in (b).

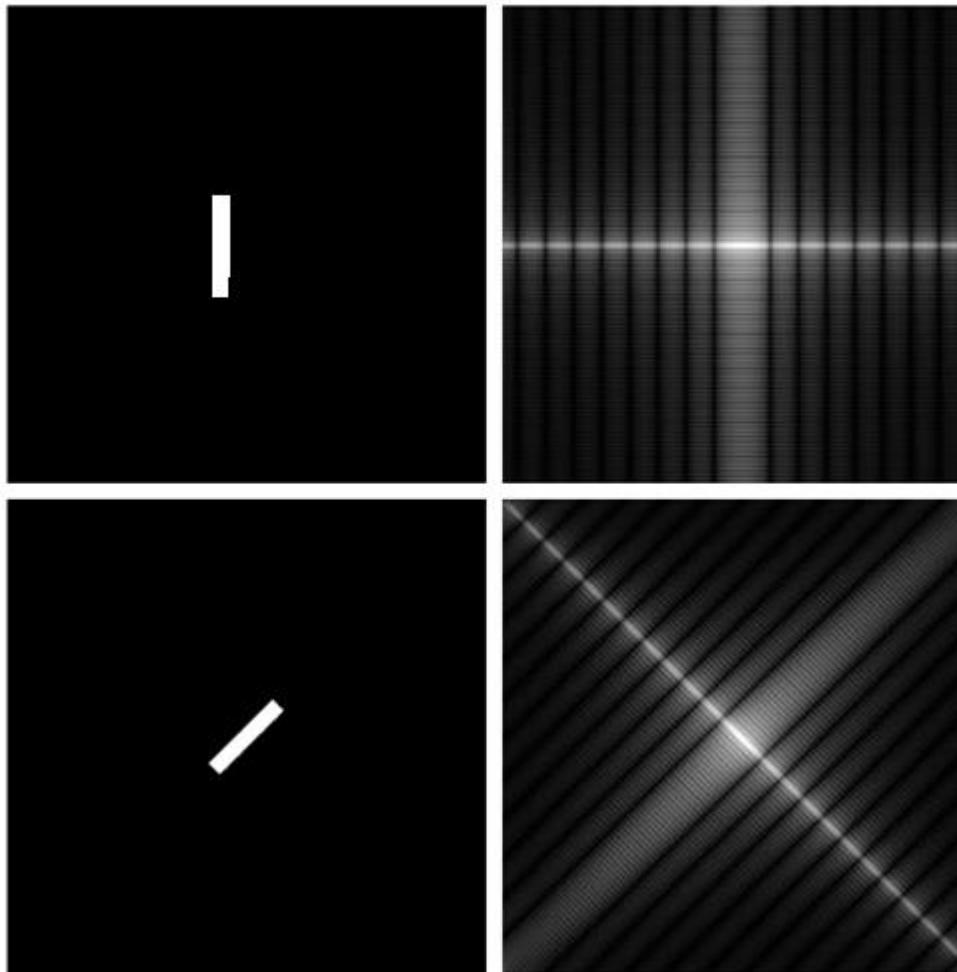
# Centering the spectrum and log enhancement



# DFT Properties: Rotation

---

- Rotating  $f(x,y)$  by  $\theta$  rotates  $F(u,v)$  by  $\theta$



# DFT Properties: Multiplication and Convolution

---

Spatial Domain ( $x$ )		Frequency Domain ( $u$ )
$g = f * h$	$\longleftrightarrow$	$G = FH$
$g = fh$	$\longleftrightarrow$	$G = F * H$

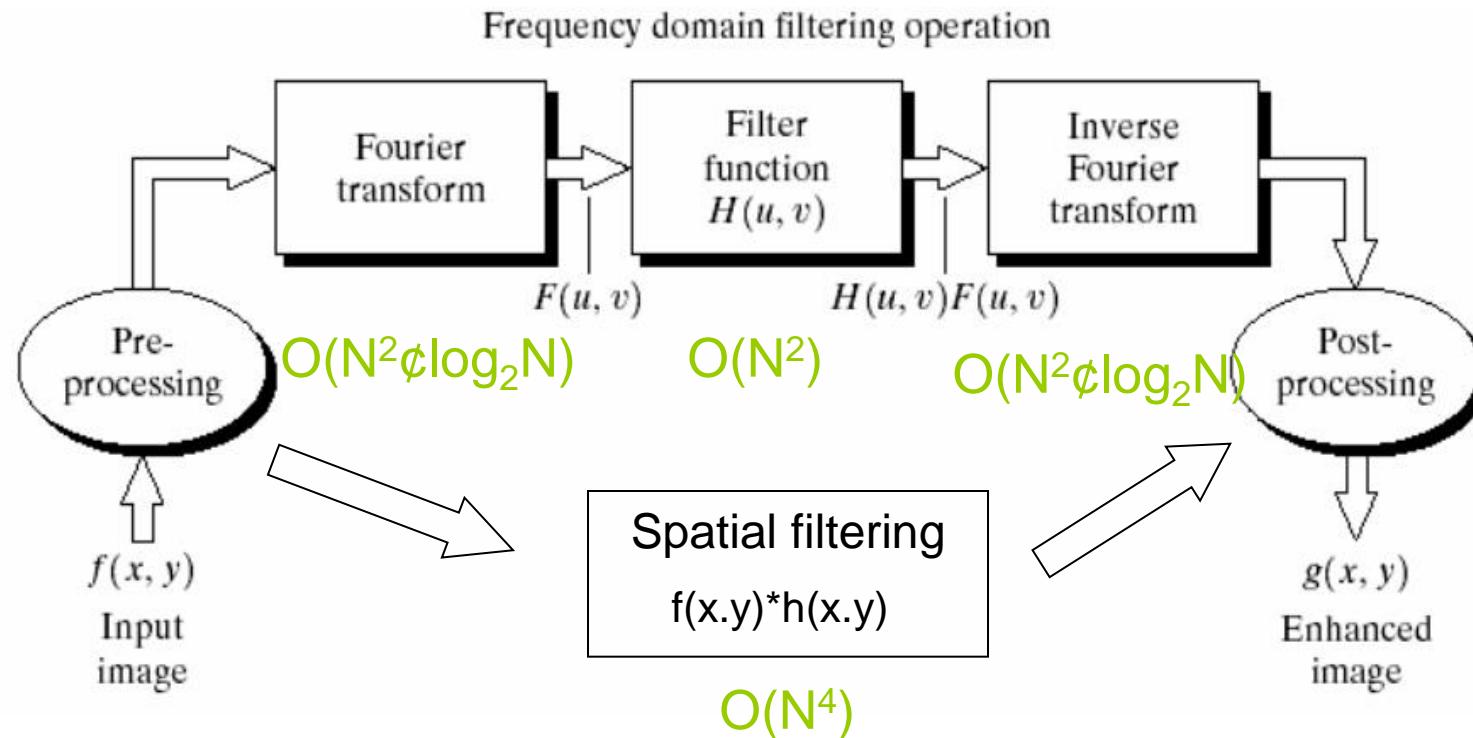
So, we can find  $g(x)$  by Fourier transform

$$\begin{array}{ccccccc} g & = & f & * & h \\ \uparrow \text{IFT} & & \downarrow \text{FT} & & \downarrow \text{FT} \\ G & = & F & \times & H \end{array}$$

# Filtering Image in Frequency Domain

To filter an image in the frequency domain:

1. Compute  $F(u,v)$  the DFT of the image
2. Multiply  $F(u,v)$  by a filter function  $H(u,v)$
3. Compute the inverse DFT of the result



# DFT Properties (contd.)

---

$$F[f(x, y) + g(x, y)] = F[f(x, y)] + F[g(x, y)] \quad (\text{Addition})$$

$$af(x, y) \xrightarrow{\quad} aF(u, v) \quad (\text{Scale})$$

$$F(0,0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \quad (\text{average})$$

$$F(u, v) = F^*(-u, -v) \quad (\text{conjugate symmetric})$$

$$|F(u, v)| = |F(-u, -v)| \quad (\text{symmetric})$$

$$\frac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v) \quad (\text{Differentiation})$$

$$(-jx)^n f(x, y) \Leftrightarrow \frac{\partial^n F(u, v)}{\partial u^n}$$

# **IMAGE ENHANCEMENT IN FREQUENCY DOMAIN**

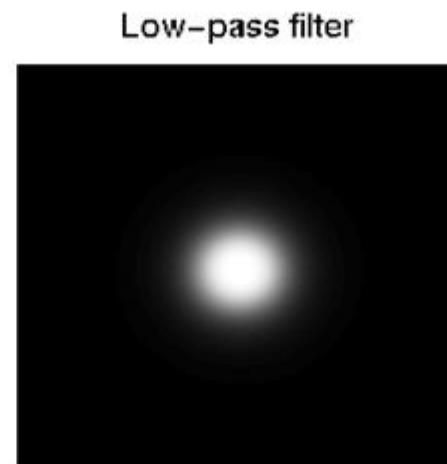
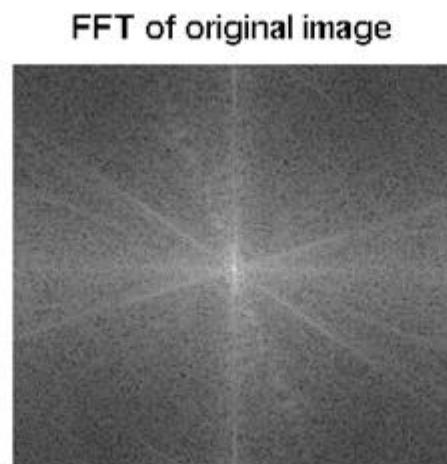
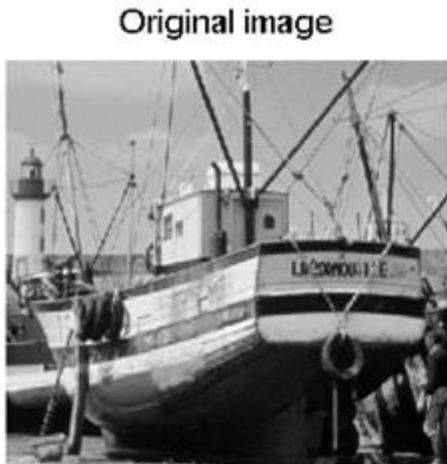
Image Smoothing

Image Sharpening

# Smoothing: Averaging / Lowpass Filtering

Smoothing/Blurring results from:

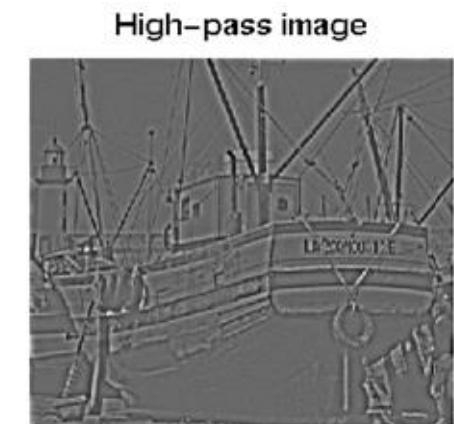
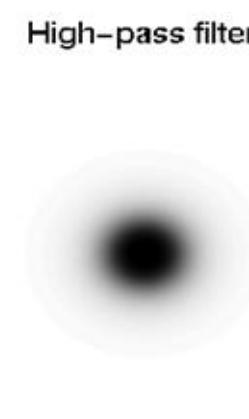
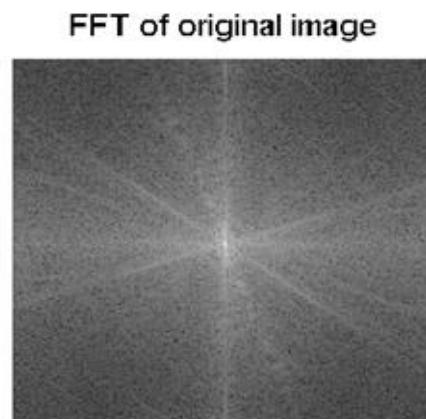
- Pixel averaging in the spatial domain:
  - Each pixel in the output is a weighted average of its neighbors.
  - Is a convolution whose weight matrix sums to 1.
- Lowpass filtering in the frequency domain:
  - High frequencies are diminished or eliminated



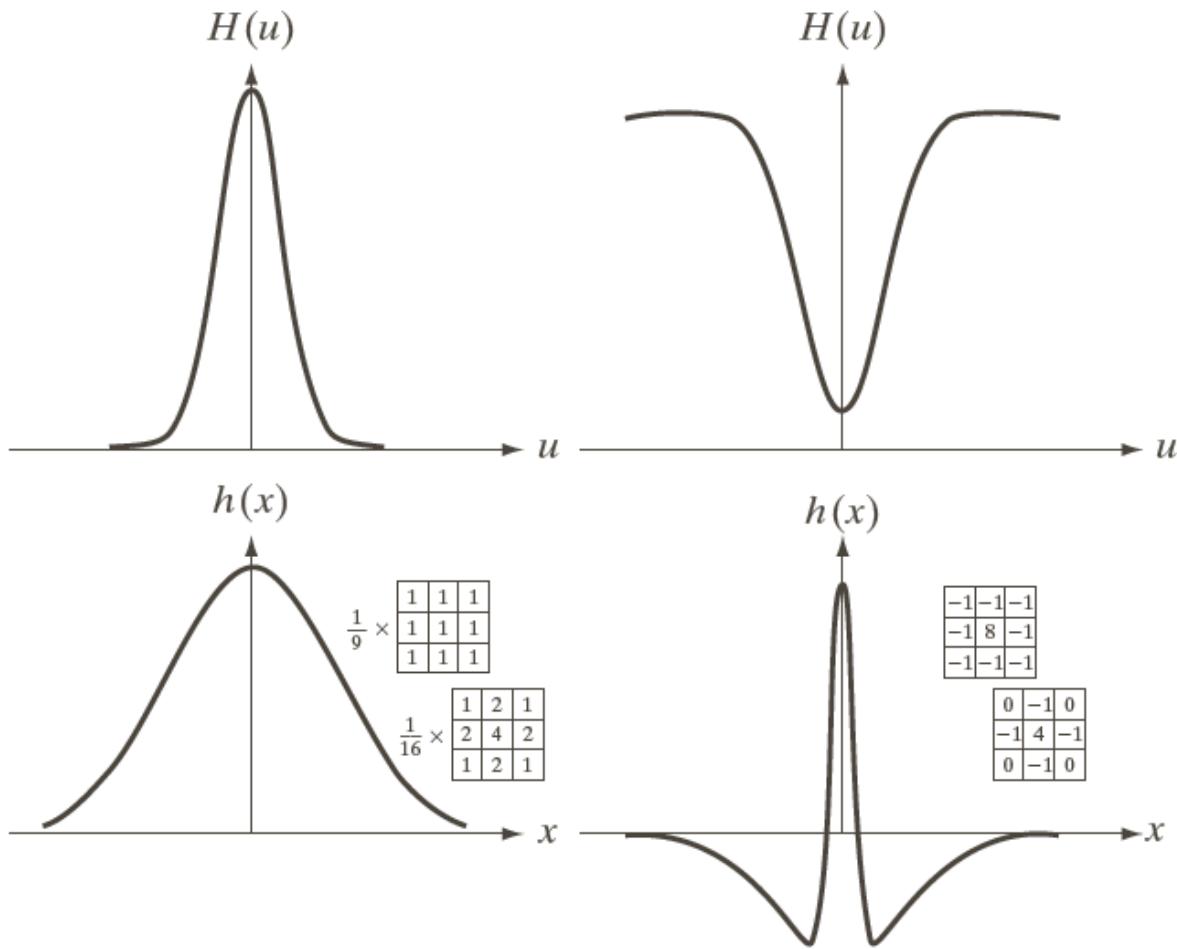
# Sharpening: Differencing / Highpass Filtering

Sharpening results from adding to the image, a copy of itself that has been:

- Pixel-differenced in the spatial domain:
  - Each pixel in the output is a difference between itself and a weighted average of its neighbors.
  - Is a convolution whose weight matrix sums to 0.
- Highpass filtered in the frequency domain:
  - High frequencies are enhanced or amplified.



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



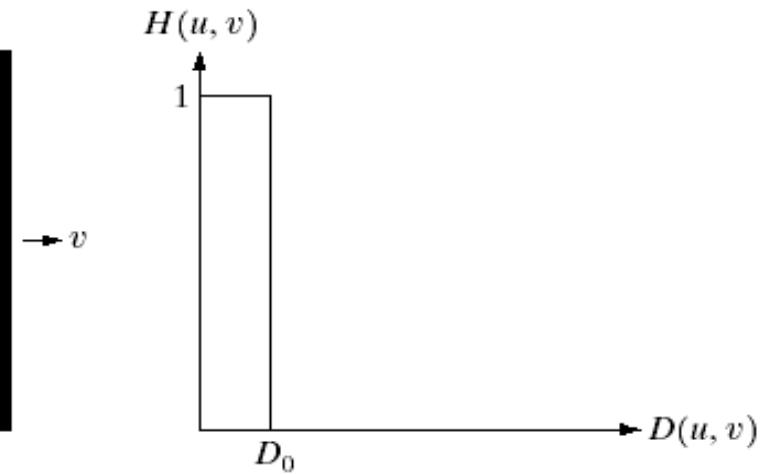
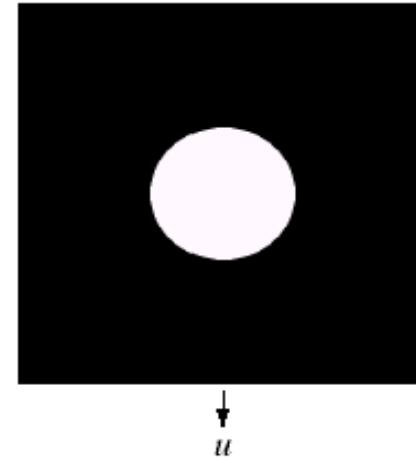
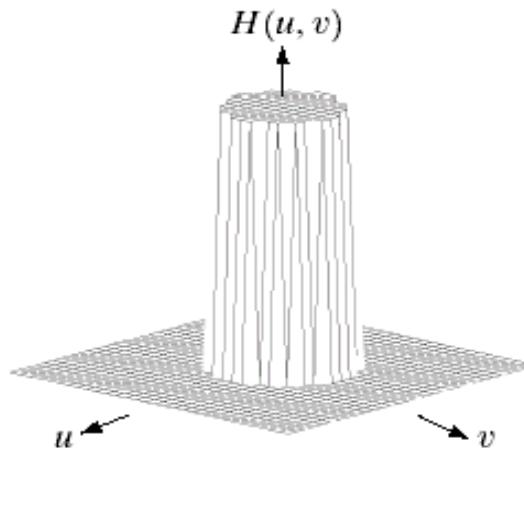
a	c
b	d

**FIGURE 4.37**

- (a) A 1-D Gaussian lowpass filter in the frequency domain.
- (b) Spatial lowpass filter corresponding to (a).
- (c) Gaussian highpass filter in the frequency domain.
- (d) Spatial highpass filter corresponding to (c). The small 2-D masks shown are spatial filters we used in Chapter 3.

# Ideal Low Pass Filter

Simply cut off all high frequency components 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 Filter (cont...)

---

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

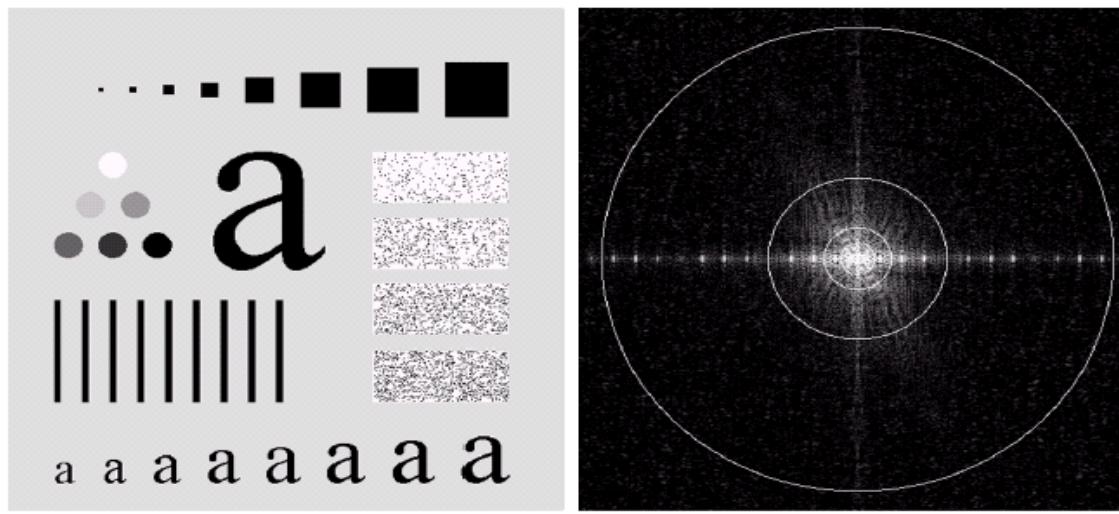
$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$$

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

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

# Ideal Low Pass Filter (cont...)

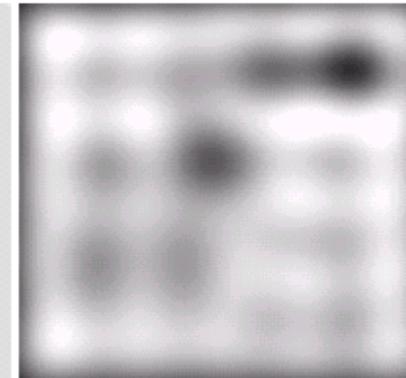
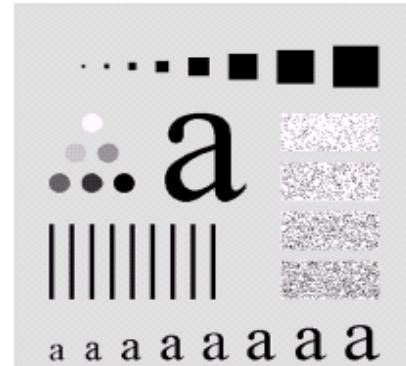
---



Above we show an image, it's Fourier spectrum and a series of ideal low pass filters of radius 5, 15, 30, 80 and 230 superimposed on top of it

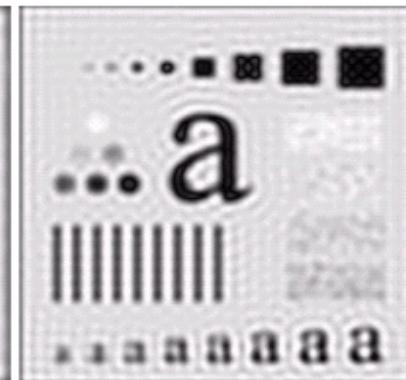
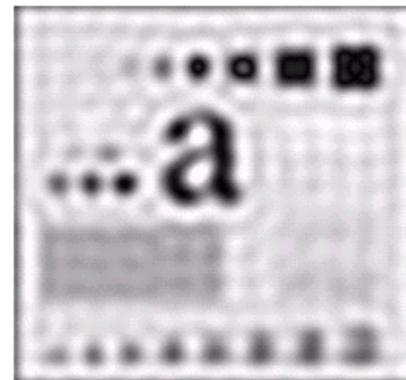
# Ideal Low Pass Filter (cont...)

Original image



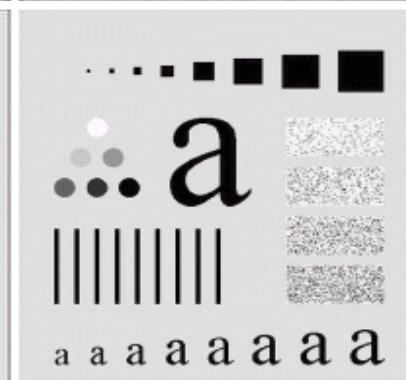
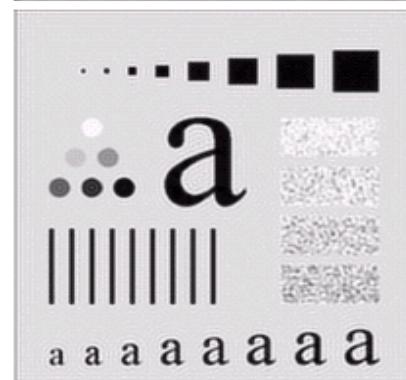
Result of filtering with ideal low pass filter of radius 5

Result of filtering with ideal low pass filter of radius 15



Result of filtering with ideal low pass filter of radius 30

Result of filtering with ideal low pass filter of radius 80



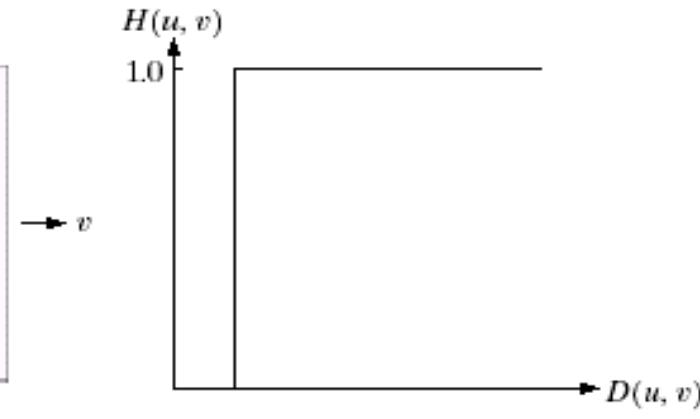
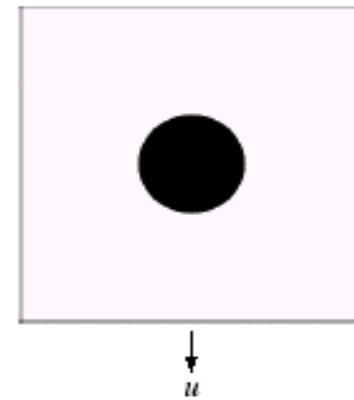
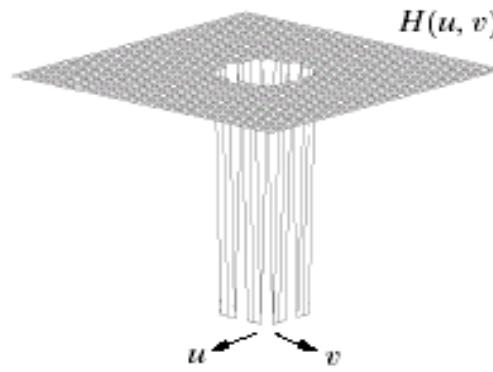
Result of filtering with ideal low pass filter of radius 230

# Ideal High Pass Filters

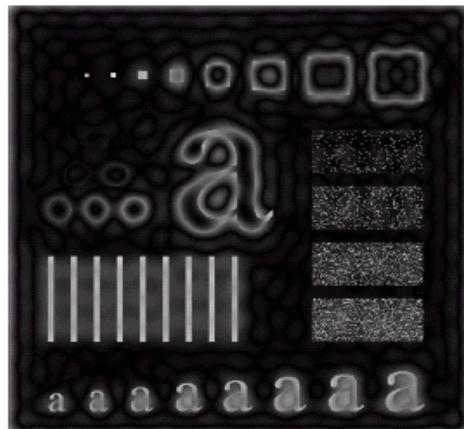
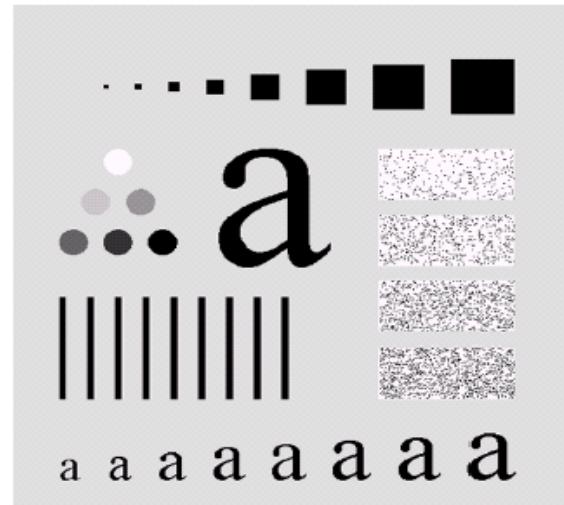
The ideal high pass filter is given as:

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

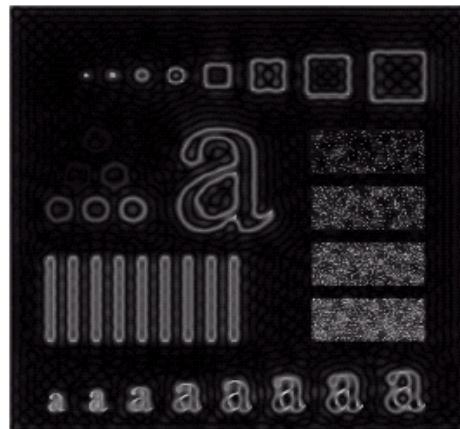
where  $D_0$  is the cut off distance as before



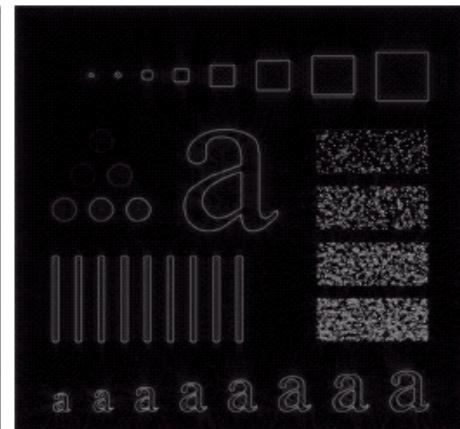
# Ideal High Pass Filters (cont...)



Results of ideal  
high pass filtering  
with  $D_0 = 15$



Results of ideal  
high pass filtering  
with  $D_0 = 30$



Results of ideal  
high pass filtering  
with  $D_0 = 80$

# Ideal Band Pass Filters

The ideal band pass filter is given as:

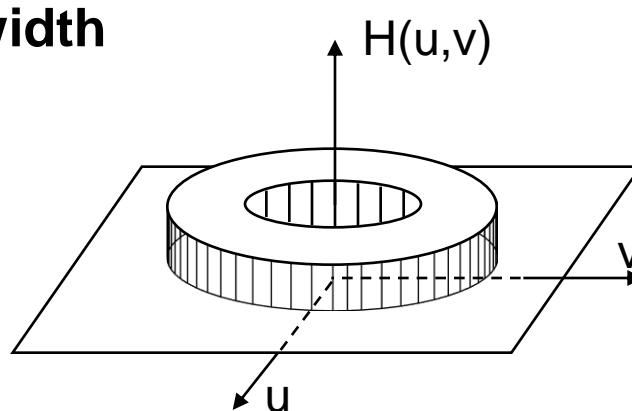
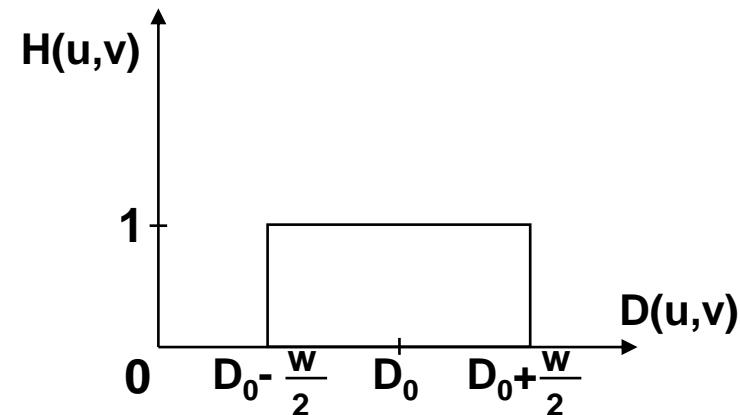
$$H(u,v) = \begin{cases} 0 & D(u,v) \leq D_0 - \frac{w}{2} \\ 1 & D_0 - \frac{w}{2} \leq D(u,v) \leq D_0 + \frac{w}{2} \\ 0 & D(u,v) > D_0 + \frac{w}{2} \end{cases}$$

where

$$D(u,v) = \sqrt{u^2 + v^2}$$

$D_0$  = cut off frequency

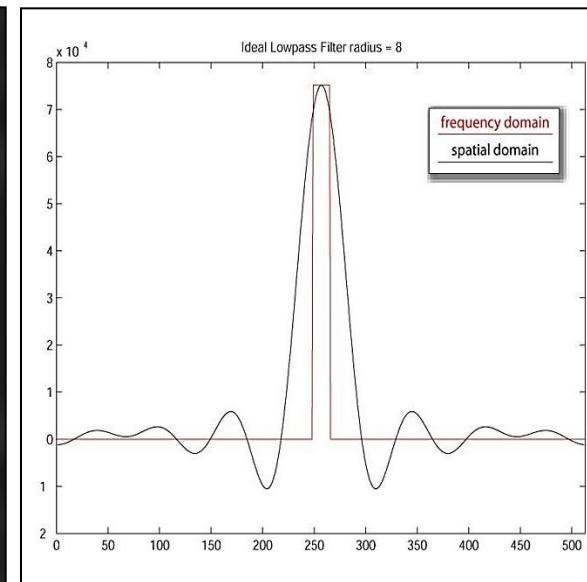
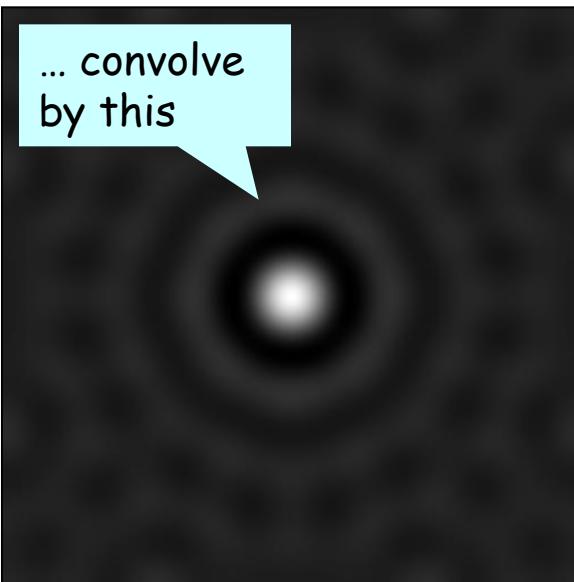
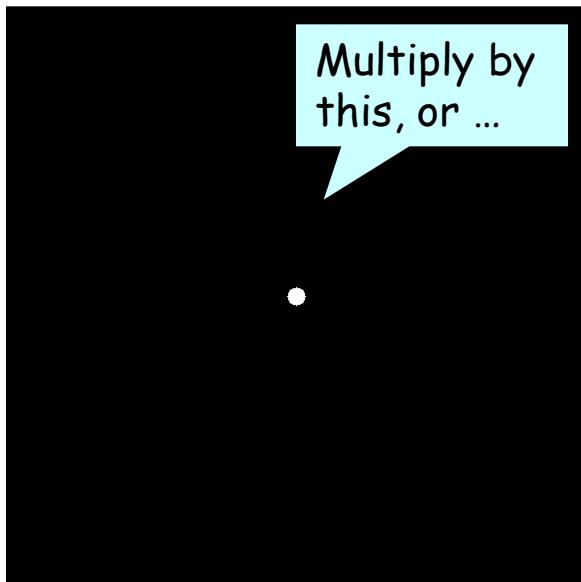
$w$  = band width



# The Ringing problem

Ideal Filters Do Not Produce Ideal Results

Image size: 512x512  
FD filter radius: 8



Fourier Domain Rep.

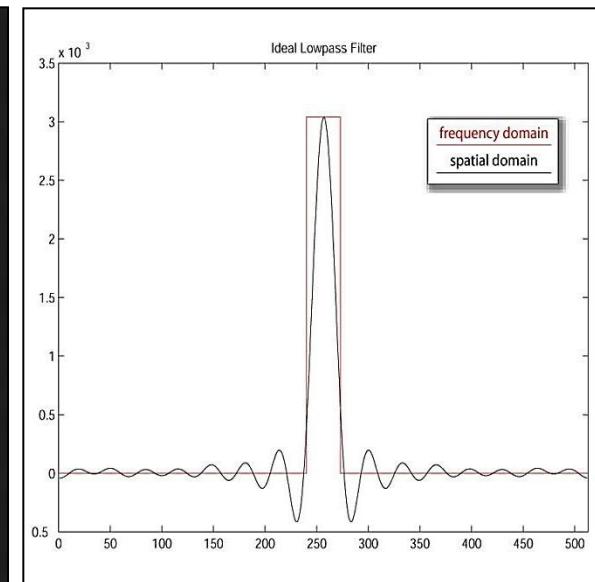
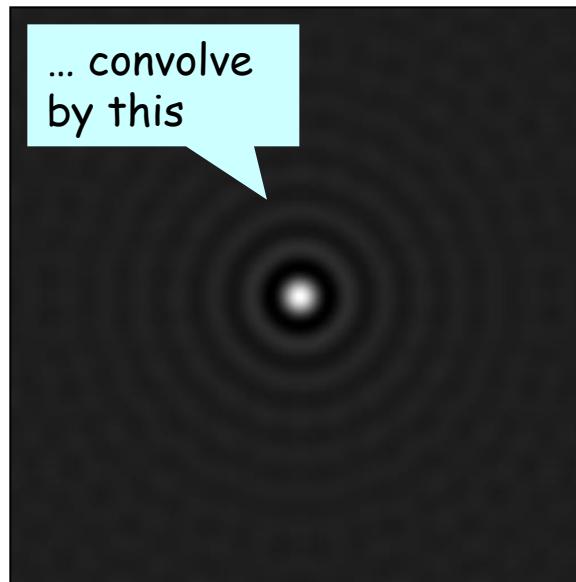
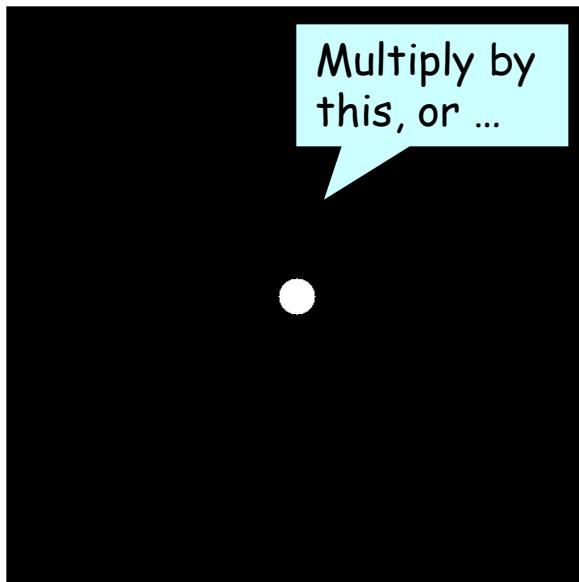
Spatial Representation

Central Profile

# The Ringing problem

Ideal Filters Do Not Produce Ideal Results

Image size: 512x512  
FD filter radius: 16



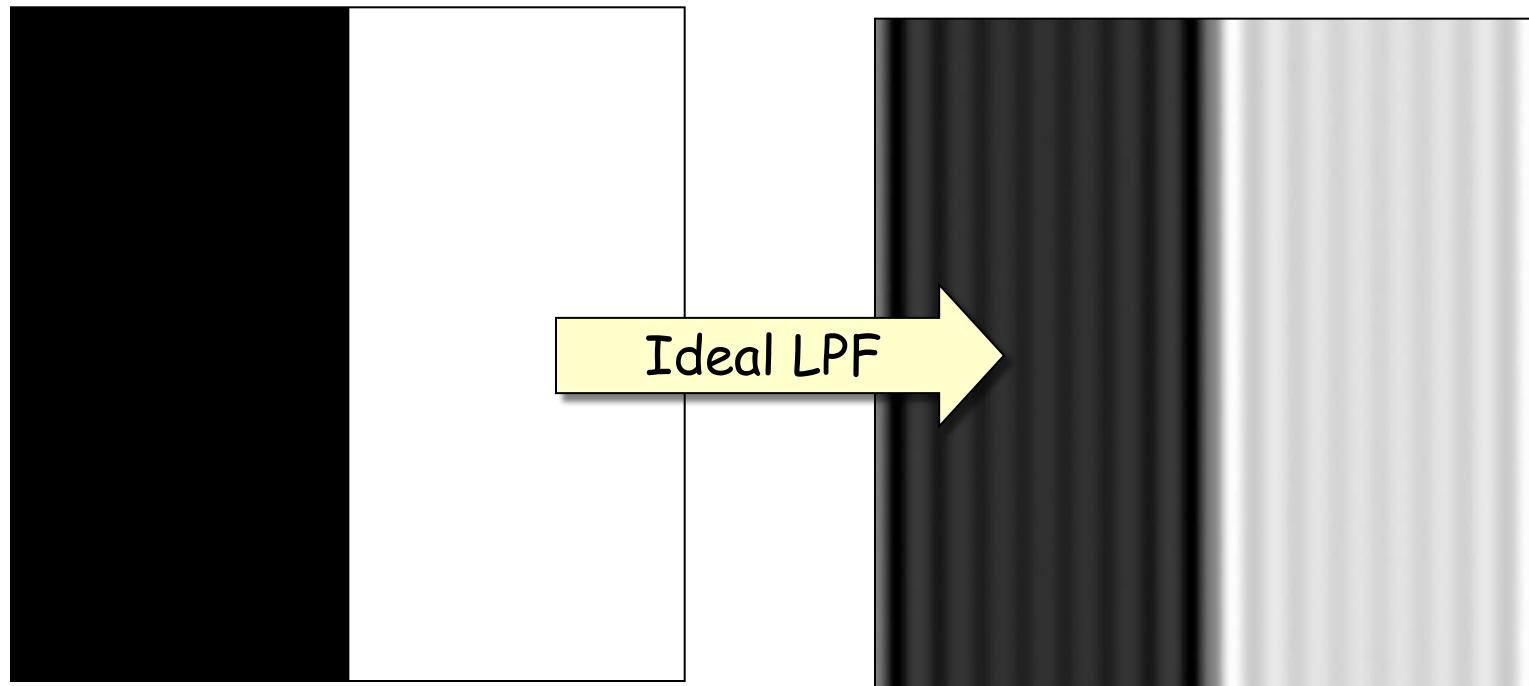
Fourier Domain Rep.

Spatial Representation

Central Profile

$\uparrow D_0$  —————→  $\downarrow$  Ringing radius +  $\downarrow$  blur

# The Ringing problem

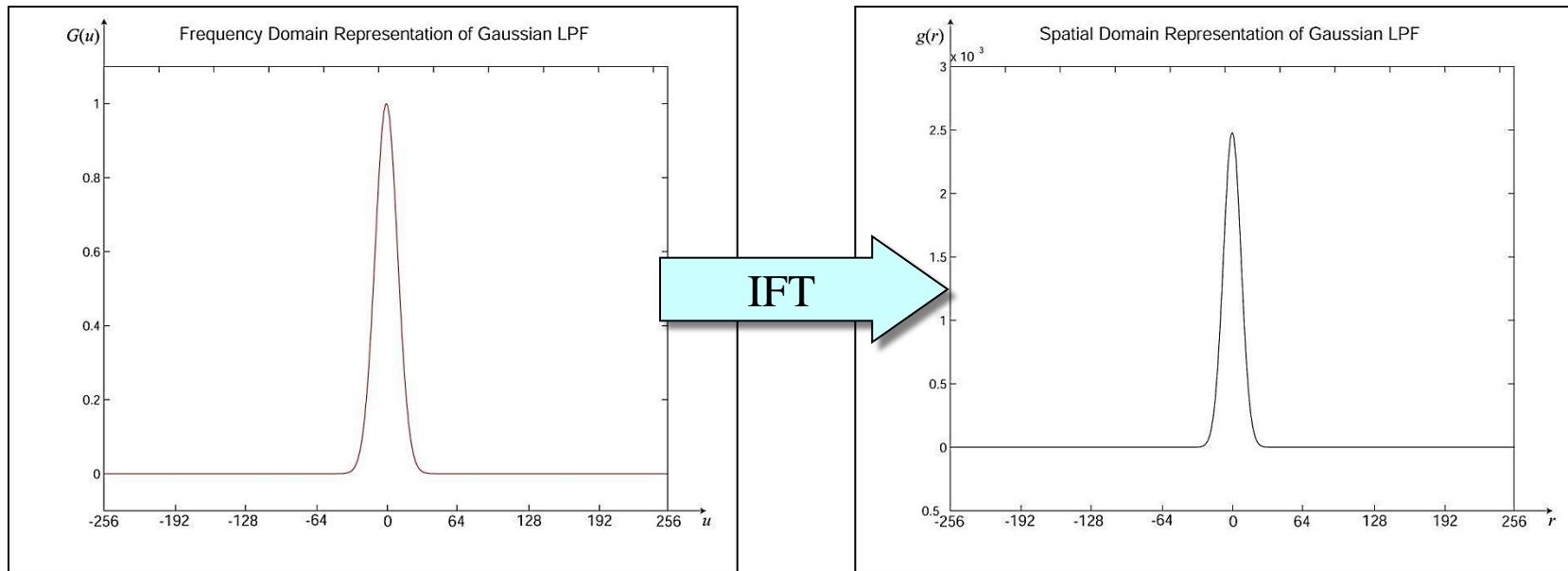


Blurring the image above w/ an ideal lowpass filter...

...distorts the results with ringing or ghosting.

# Optimal Filter

## The Gaussian

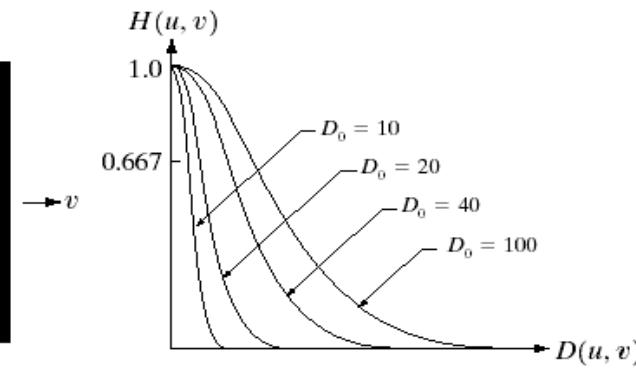
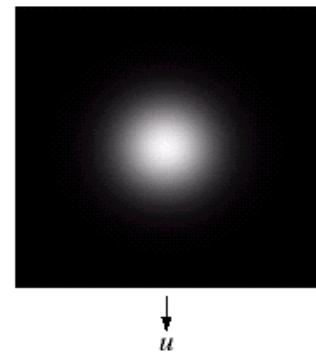
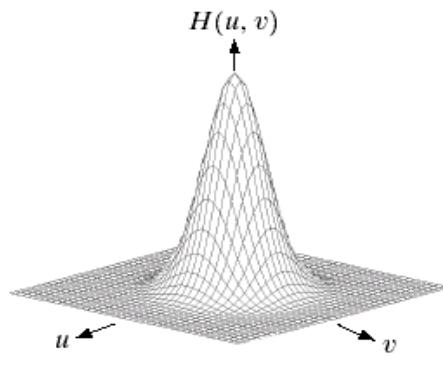


The Gaussian filter optimizes it by providing the sharpest cutoff possible without ringing.

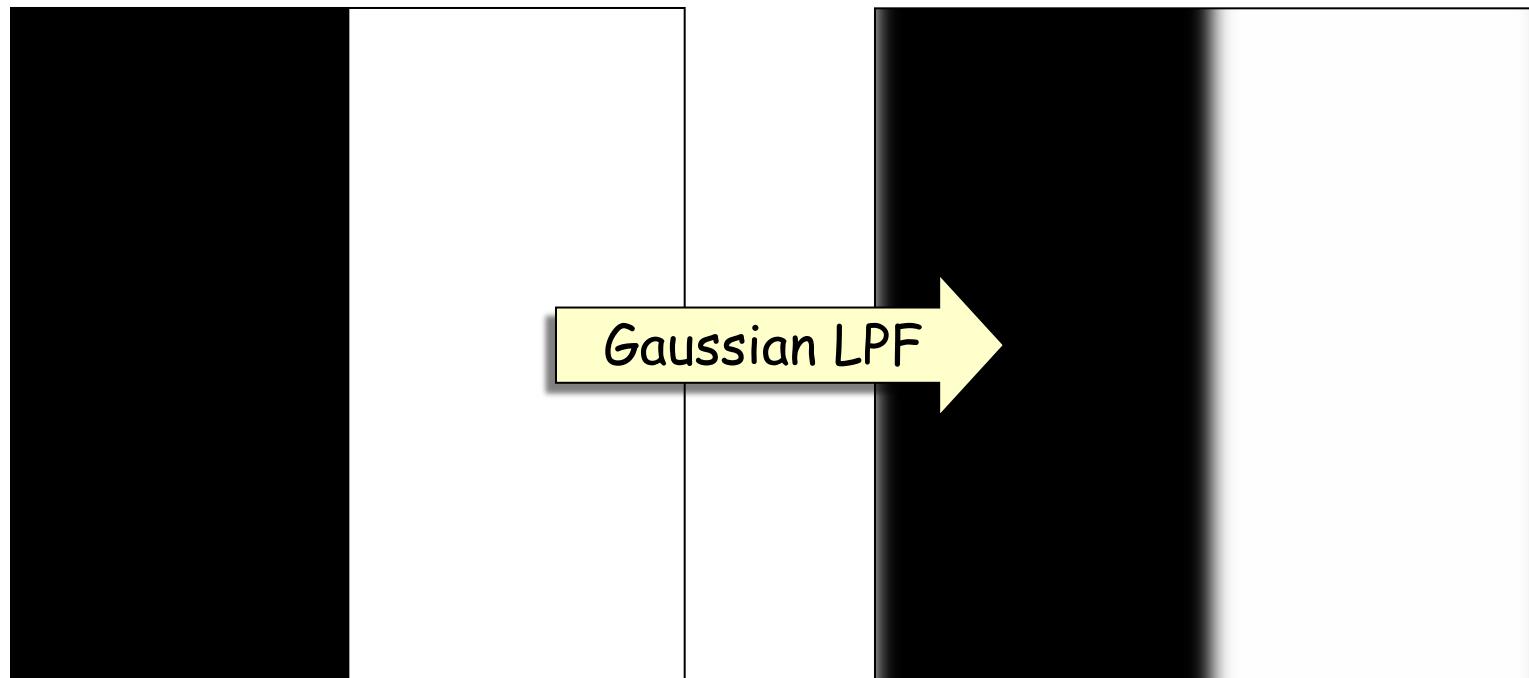
# Gaussian Lowpass Filters

The transfer function of a Gaussian lowpass filter is defined as:

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



# Optimal Filter: The Gaussian

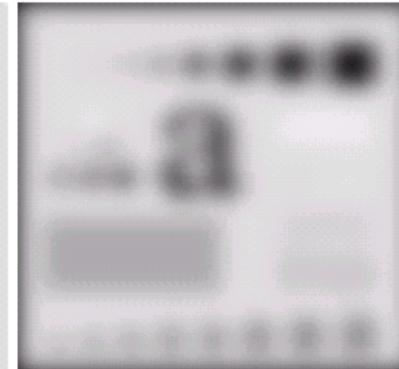
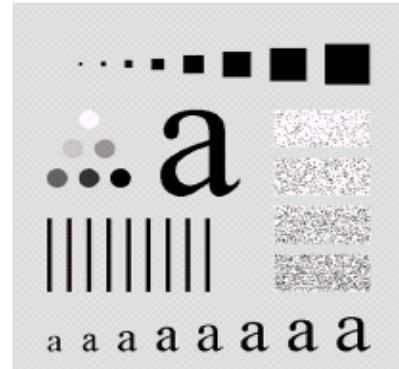


With a gaussian lowpass filter, the image above ...

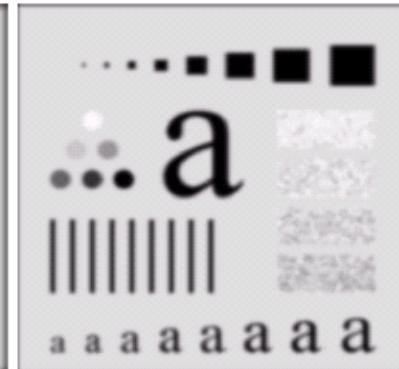
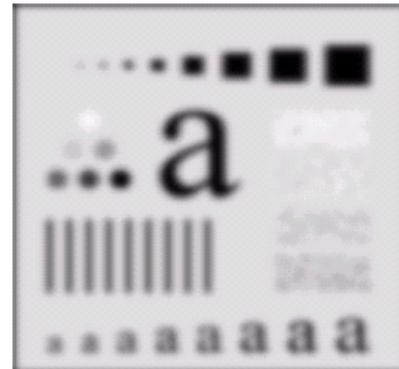
... is blurred without ringing or ghosting.

# Gaussian Lowpass Filters (cont...)

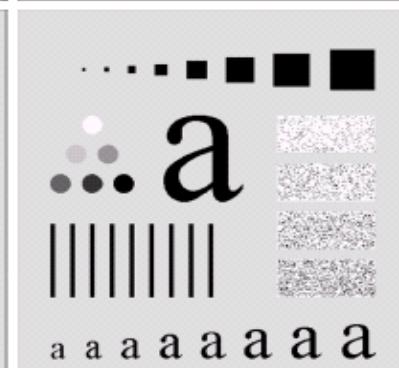
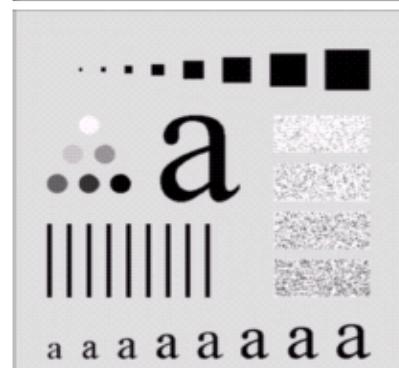
Original image



Result of filtering with Gaussian filter with cutoff radius 15



Result of filtering with Gaussian filter with cutoff radius 85



Result of filtering with Gaussian filter with cutoff radius 5

Result of filtering with Gaussian filter with cutoff radius 30

Result of filtering with Gaussian filter with cutoff radius 230

# Resolution Sequence

Original Image

$$\sigma_0 = 0$$



# Resolution Sequence

Gaussian  
LPF

$$\sigma_1 = 1$$



# Resolution Sequence

Gaussian  
LPF  
 $\sigma_2 = 2$



# Resolution Sequence

Gaussian  
LPF

$$\sigma_3 = 4$$



# Resolution Sequence

Gaussian  
LPF

$$\sigma_4 = 8$$

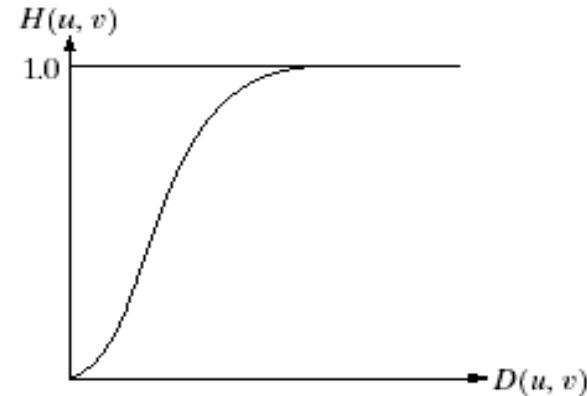
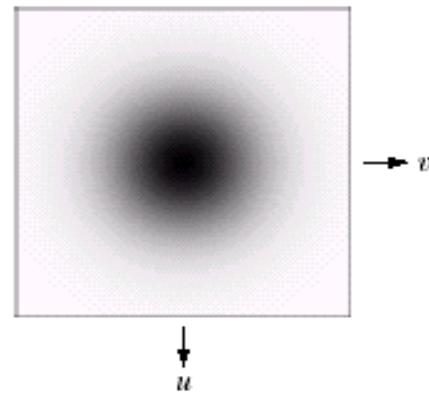
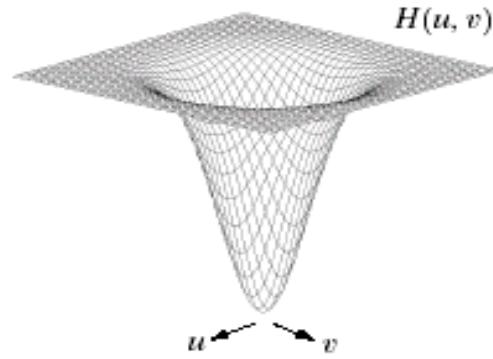


# Gaussian High Pass Filters

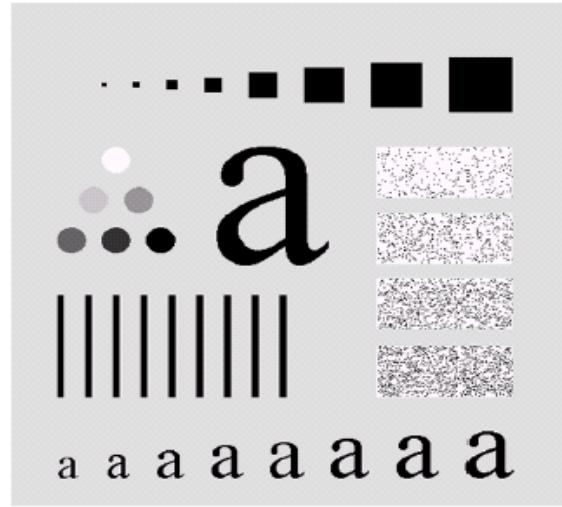
The Gaussian high pass filter is given as:

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

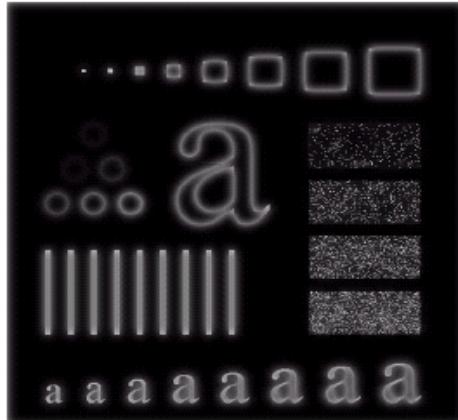
where  $D_0$  is the cut off distance as before



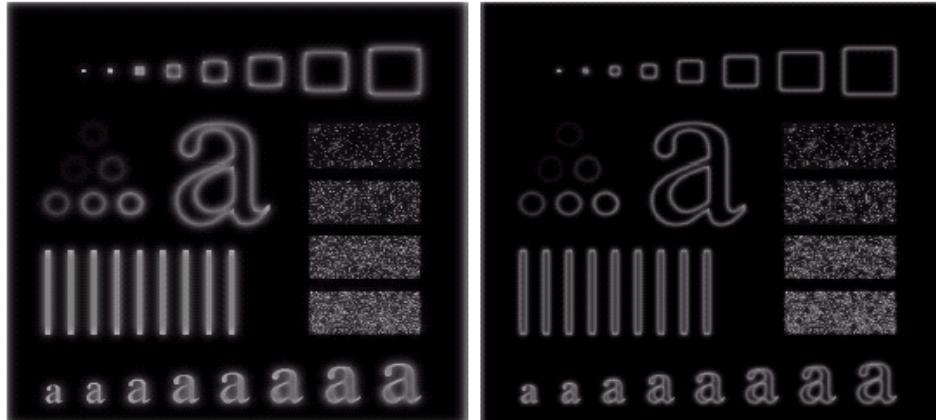
# Gaussian High Pass Filters (cont...)



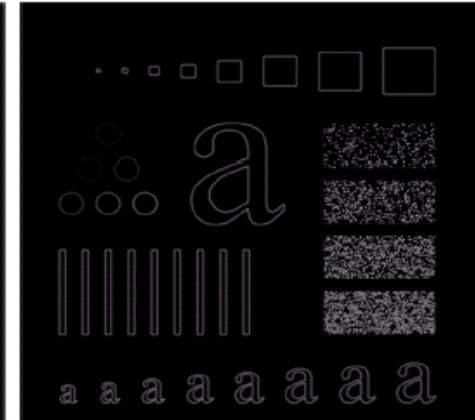
Results of Gaussian high pass filtering with  $D_0 = 15$



Results of Gaussian high pass filtering with  $D_0 = 30$



Results of Gaussian high pass filtering with  $D_0 = 80$



# Highpass Sequence

Difference between original image and Gaussian LPF image at  $\sigma_9 = 256$ .

$$\mathbf{J} = \mathbf{I} - [\mathbf{I} * g(\sigma_9)].$$



# Highpass Sequence

Difference between  
original image and  
Gaussian LPF image  
at  $\sigma_8 = 128$ .

$$\mathbf{J} = \mathbf{I} - [\mathbf{I} * g(\sigma_8)].$$



# Highpass Sequence

Difference between original image and Gaussian LPF image at  $\sigma_7 = 64$ .

$$\mathbf{J} = \mathbf{I} - [\mathbf{I} * g(\sigma_7)].$$



# Highpass Sequence

Difference between original image and Gaussian LPF image at  $\sigma_6 = 32$ .

$$\mathbf{J} = \mathbf{I} - [\mathbf{I} * g(\sigma_6)].$$



# Highpass Sequence

Difference between original image and Gaussian LPF image at  $\sigma_5 = 16$ .

$$\mathbf{J} = \mathbf{I} - [\mathbf{I} * g(\sigma_5)].$$



# Sharpening above a Specific Scale.

An image is sharpened by taking a linear combination of the image and a highpass filtered version of itself. The scale of the sharpening can be controlled via the cutoff of the HPF and a multiplicative constant. In the following examples the image has been sharpened via

$$\mathbf{I}_{\text{hfe}, \sigma} = \mathbf{I} + \alpha \mathbf{I}_{\text{hpf}, \sigma} = \mathbf{I} + \alpha (\mathbf{I} - [\mathbf{I} * g(\sigma)]) = (1 + \alpha) \mathbf{I} - \alpha [\mathbf{I} * g(\sigma)],$$

where  $g$  is a 2D Gaussian with  $\sigma \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$  and  $\alpha$  is a scale factor, usually in  $(0, 2)$ . After the computation, each image was histogram matched to  $\mathbf{I}$ .

$\sigma_0 = 0$  Original Image

# Sharpening above a Specific Scale



CSX SW1500 at N. Charleston, SC, SEP 1999  
Photo by T. Moses (c)

$$\sigma_0 = 1, \alpha=1$$

# Sharpening above a Specific Scale



$$\sigma_0 = 2, \alpha=1$$

# Sharpening above a Specific Scale



$$\sigma_0 = 4, \alpha=1$$

# Sharpening above a Specific Scale



CSX SW1500 at N. Charleston, SC, SEP 1999  
Photo by T. Moses (c)

$$\sigma_0 = 8, \alpha=1$$

# Sharpening above a Specific Scale



$$\sigma_0 = 16, \alpha=1$$

# Sharpening above a Specific Scale



$$\sigma_0 = 32, \alpha=1$$

# Sharpening above a Specific Scale



$$\sigma_0 = 64, \alpha=1$$

# Sharpening above a Specific Scale



# Bandpass Sequence

Difference between  
Gaussian LPF images:  
 $\sigma_6 = 32$  and  $\sigma_7 = 64$ .



$$\mathbf{J} = \mathbf{I} * [g(\sigma_6) - g(\sigma_7)] = [\mathbf{I} * g(\sigma_6)] - [\mathbf{I} * g(\sigma_7)].$$

# Bandpass Sequence

Difference between Gaussian LPF images:  
 $\sigma_5 = 16$  and  $\sigma_6 = 32$ .



$$\mathbf{J} = \mathbf{I} * [g(\sigma_5) - g(\sigma_6)] = [\mathbf{I} * g(\sigma_5)] - [\mathbf{I} * g(\sigma_6)].$$

# Bandpass Sequence

Difference between Gaussian LPF images:  
 $\sigma_4 = 8$  and  $\sigma_5 = 16$ .



$$\mathbf{J} = \mathbf{I} * [g(\sigma_4) - g(\sigma_5)] = [\mathbf{I} * g(\sigma_4)] - [\mathbf{I} * g(\sigma_5)].$$

# Emphasizing a Specific Pass Band.

An image can be bandpass filtered by subtracting two differently Gaussian filtered copies of it. That specific band can be emphasized in the image by adding it back to the image. In the following examples the image has been emphasized via

$$\begin{aligned}\mathbf{I}_{\text{bpe}, \sigma_0, \sigma_1} &= \mathbf{I} + \alpha \mathbf{I}_{\text{bpf}, \sigma_0, \sigma_1} \\ &= \mathbf{I} + \alpha \left[ \mathbf{I} - \left( [\mathbf{I} * g(\sigma_0)] - [\mathbf{I} * g(\sigma_1)] \right) \right] \\ &= (1 + \alpha) \mathbf{I} - \alpha \left( \mathbf{I} * [g(\sigma_0) - g(\sigma_1)] \right),\end{aligned}$$

where  $\sigma_0, \sigma_1 \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$  and  $\alpha$  is a scale factor, usually in  $(0, 2)$ . After the computation, each image was histogram matched to  $\mathbf{I}$ .

$$(\sigma_1, \sigma_0) = (1, 0), \alpha=1$$

# Emphasizing a Specific Pass Band



CSX SW1500 at N. Charleston, SC, SEP 1999  
Photo by T. Moses (c)

$$(\sigma_1, \sigma_0) = (2, 1), \alpha=1$$

# Emphasizing a Specific Pass Band



CSX SW1500 at N. Charleston, SC, SEP 1999  
Photo by T. Moses (c)

$$(\sigma_1, \sigma_0) = (4,2), \alpha=1$$

# Emphasizing a Specific Pass Band



$$(\sigma_1, \sigma_0) = (8,4), \alpha=1$$

# Emphasizing a Specific Pass Band



CSX SW1500 at N. Charleston, SC, SEP 1999  
Photo by T. Moses (c)

$$(\sigma_1, \sigma_0) = (16, 8), \alpha=1$$

# Emphasizing a Specific Pass Band



CSX SW1500 at N. Charleston, SC, SEP 1999  
Photo by T. Moses (c)

$$(\sigma_1, \sigma_0) = (32, 16), \alpha=1$$

# Emphasizing a Specific Pass Band



$$(\sigma_1, \sigma_0) = (64, 32), \alpha=1$$

# Emphasizing a Specific Pass Band



CSX SW1500 at N. Charleston, SC, SEP 1999  
Photo by T. Moses (c)

# Noise Enhancement: the Problem with Sharpening

## Effects of Noise on Enhancement of HF



original image



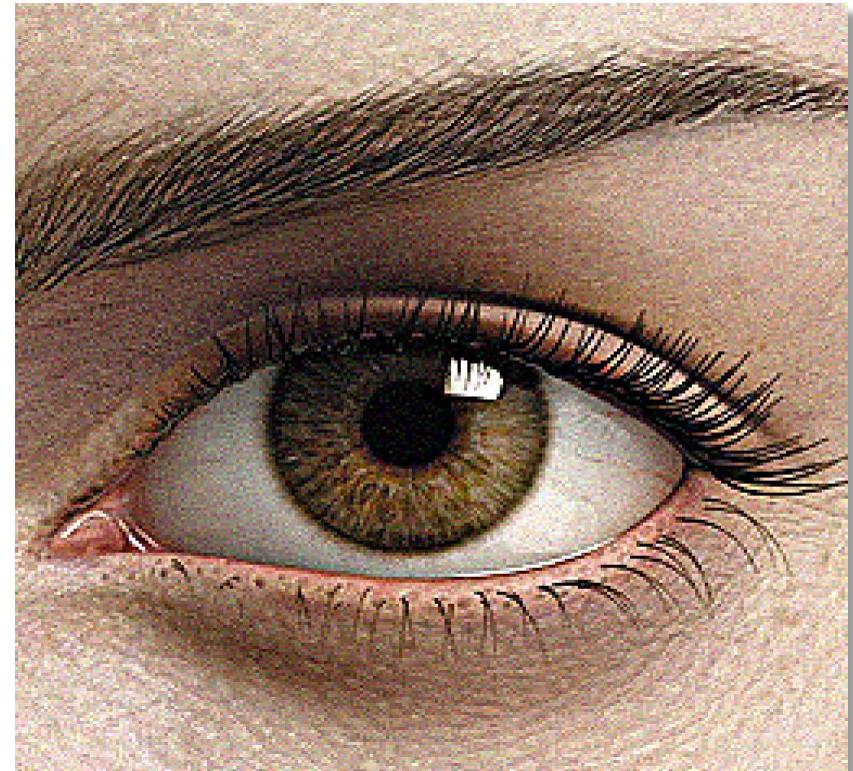
HF enhanced original

# Noise Enhancement: the Problem with Sharpening

## Effects of Noise on Enhancement of HF



noisy image



HF enhanced noisy image

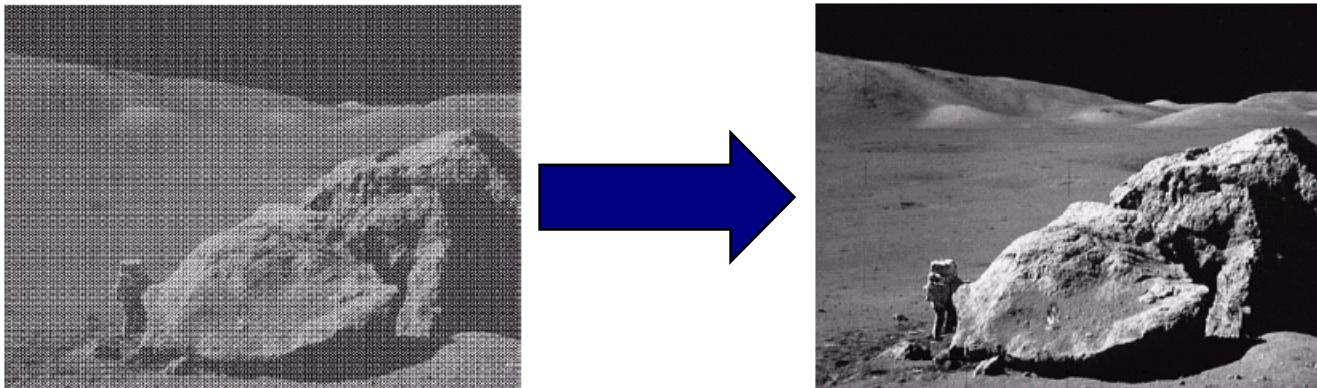
# Image & Video Processing

Image Restoration:  
Noise Removal

# What is Image Restoration?

Image restoration attempts to restore images that have been degraded

- Identify the degradation process and attempt to remove it in order to go back to the “original”
- Similar to image enhancement, but more objective

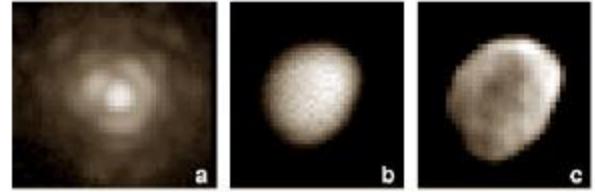


# Applications

Started from the 1950s

- Scientific explorations
- Legal investigations
- Film making and archival
- Image and video (de-)coding
- ...
- Consumer photography

Example of image restoration  
Asteroid Vesta



# Degradation causes

## Degradation examples:



- original



- optical blur



- motion blur



- spatial quantization (discrete pixels)

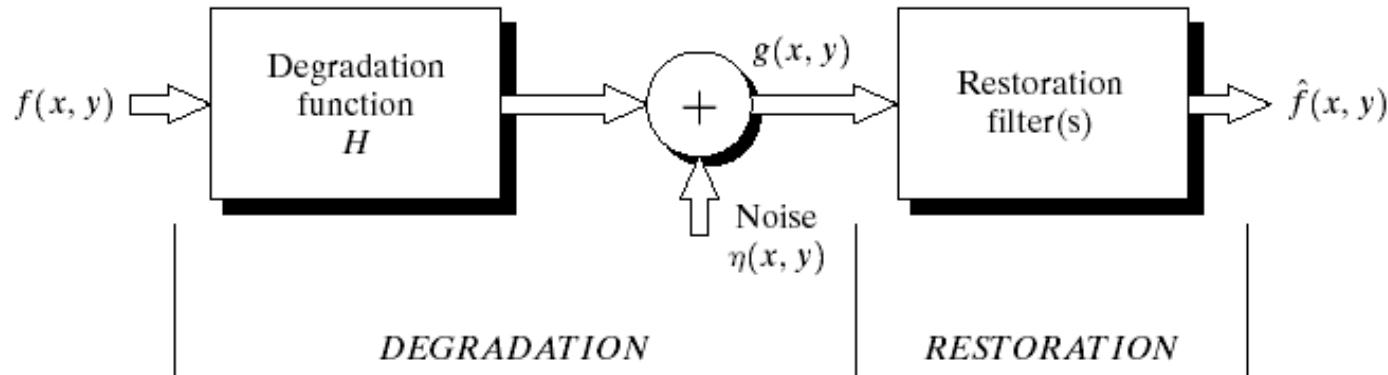


- additive intensity noise

### Causes:

- Camera: translation, shake, out-of-focus ...
- Environment: scattered and reflected light
- Device noise: CCD/CMOS sensor and circuitry
- Quantization noise
- Transmission error

# A Model for Image Distortion/Restoration



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

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

where  $f(x, y)$  is the original image pixel,  $H$  is the degradation function,  $\eta(x, y)$  is the noise term and  $g(x, y)$  is the resulting noisy pixel

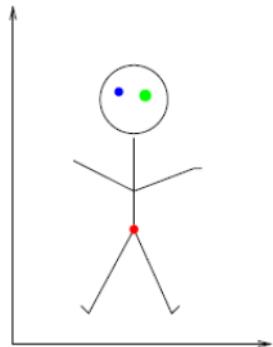
# Assumptions for the Distortion Model

- Noise
  - Independent of spatial location
    - Exception: periodic noise ..
  - Uncorrelated with image
- Degradation function  $H$ 
  - Linear
  - Position-invariant

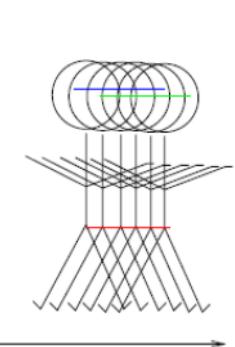
$$g(x, y) = h(x, y) * F(x, y) + \eta(x, y)$$

**Step #1: image degraded only by noise.**

Input Image

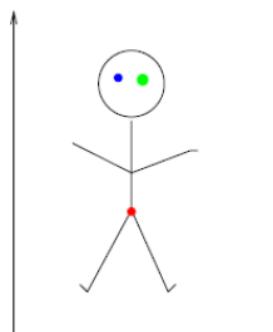


Blurred by Camera linear motion

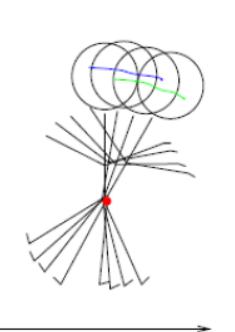


SPACE-INVARIANT RESPONSE - each point on image gives same response just shifted in position.

Input Image



Blurred by Camera Rotation

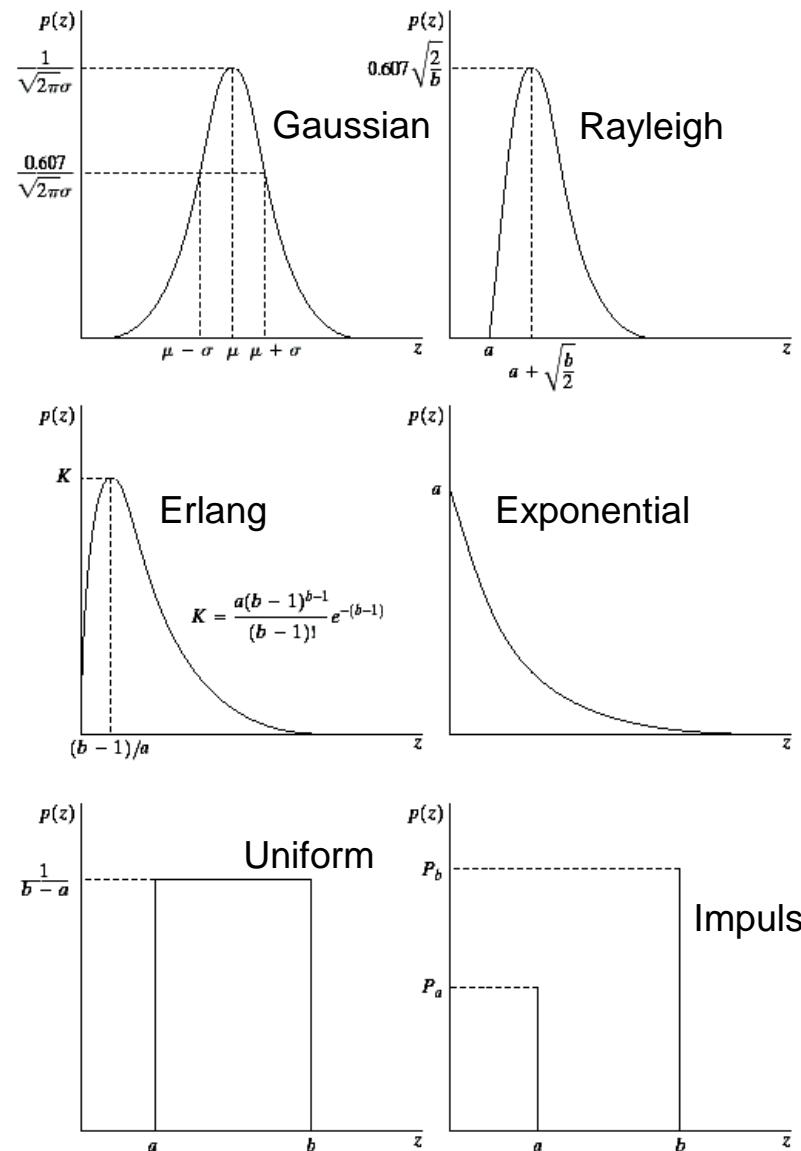


SPACE-VARIANT RESPONSE - each point on image gives a different response

# Noise Models

There are many different models for the image noise term  $\eta(x, y)$ :

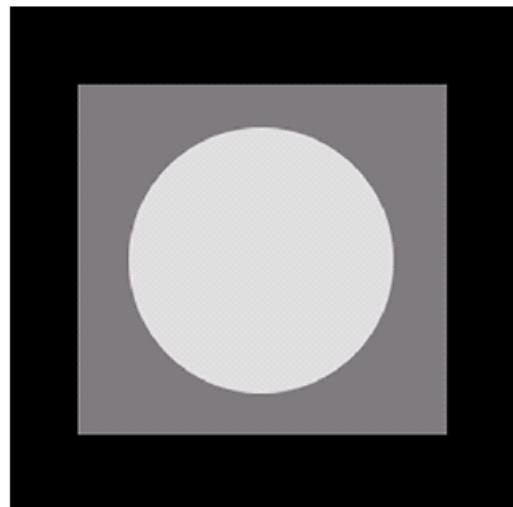
- Gaussian
  - Most common model
- Rayleigh
- Erlang
- Exponential
- Uniform
- Impulse
  - *Salt and pepper* noise



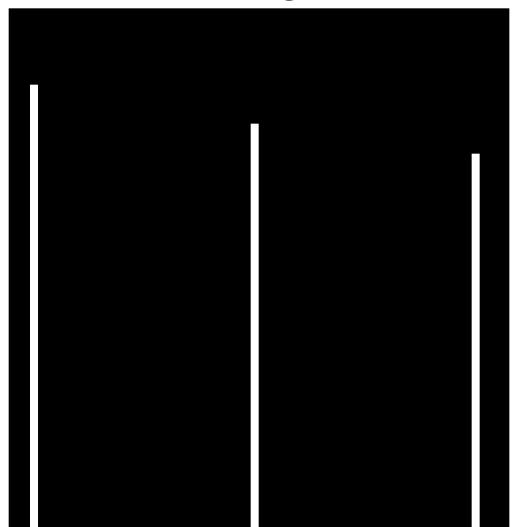
# Noise Example

The test pattern to the right is ideal for demonstrating the addition of noise

The following slides will show the result of adding noise based on various models to this image



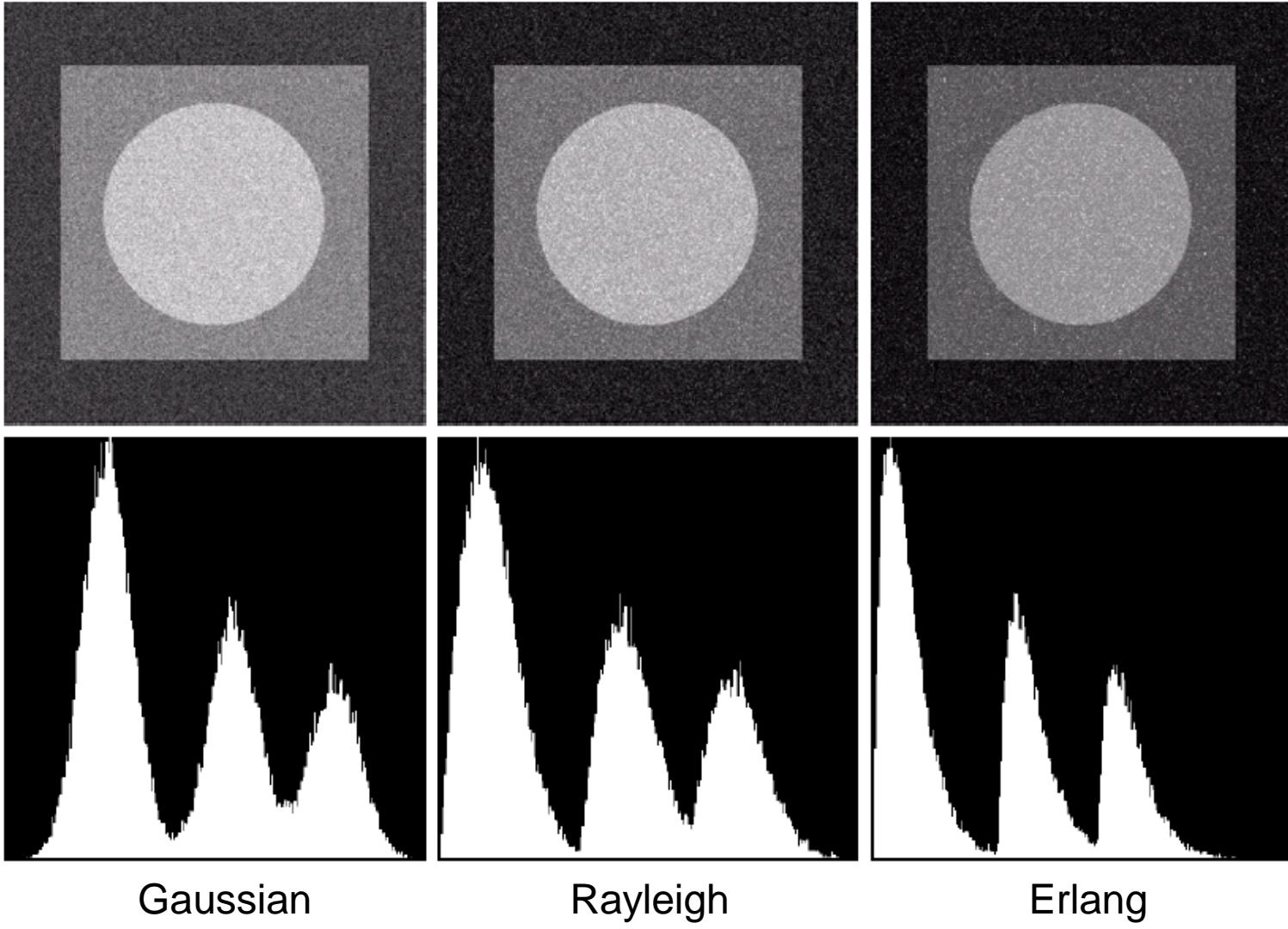
Image



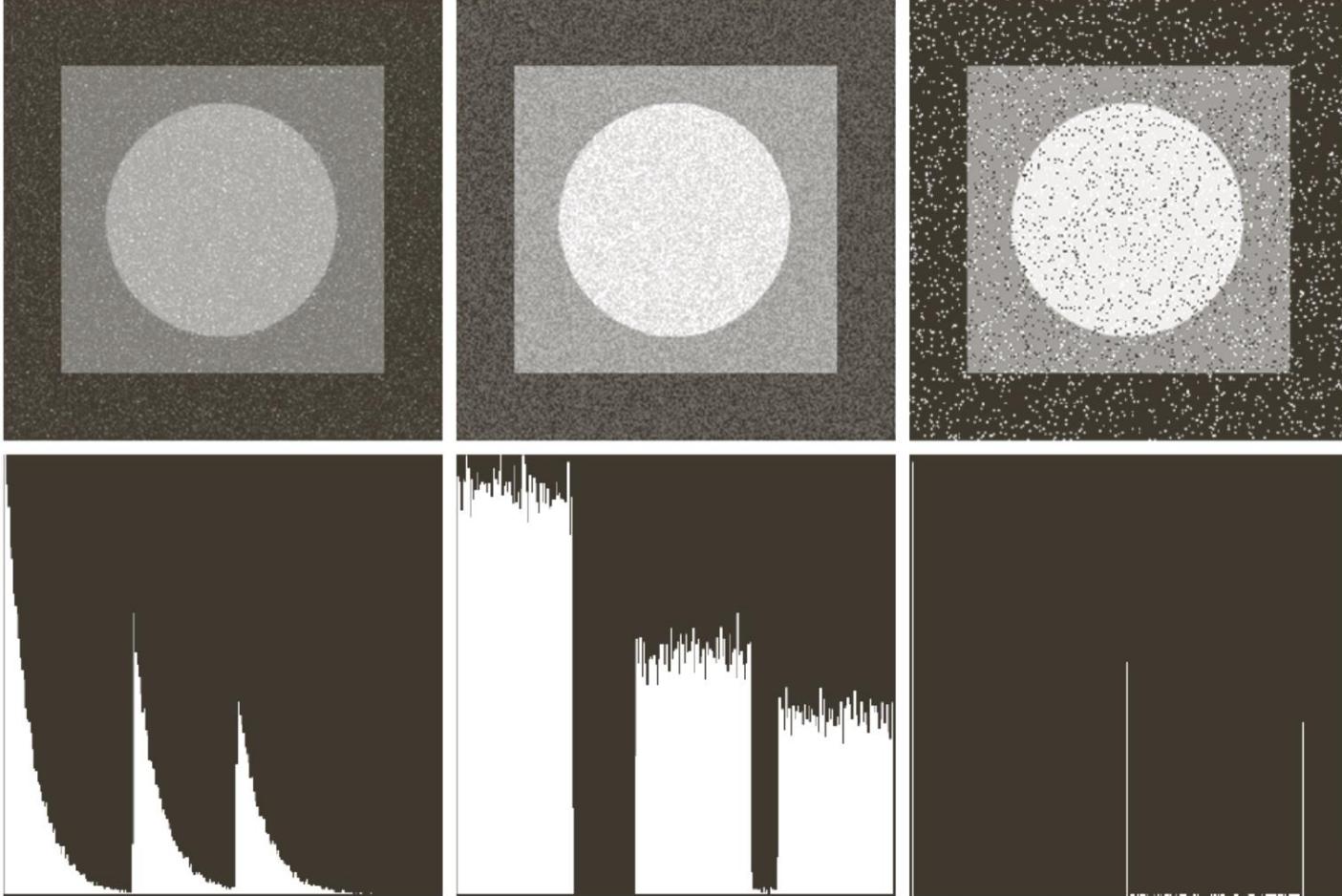
Histogram



# Noise Example (cont...)



# Noise Example (cont...)

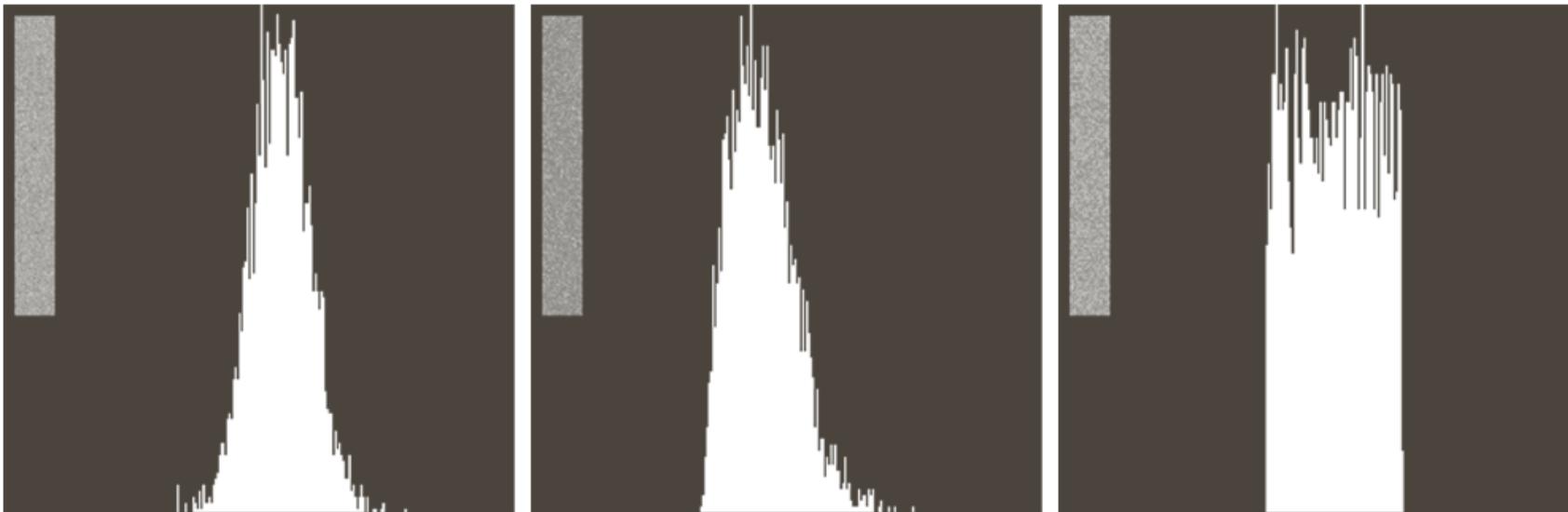


Exponential

Uniform

Impulse

# Estimation of noise parameters



a | b | c

**FIGURE 5.6** Histograms computed using small strips (shown as inserts) from (a) the Gaussian, (b) the Rayleigh, and (c) the uniform noisy images in Fig. 5.4.

# Filtering to Remove Noise

We can use spatial filters of different kinds to remove different kinds of noise

The *arithmetic mean* filter is a very simple one and is calculated as follows:

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

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

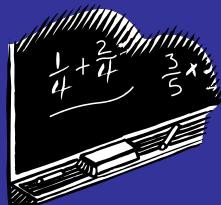
This is implemented as the simple smoothing filter

Blurs the image to remove noise

# Other Means

There are different kinds of mean filters all of which exhibit slightly different behaviour:

- Geometric Mean
- Harmonic Mean
- Contraharmonic Mean



# Other Means (cont...)

There are other variants on the mean which can give different performance

**Geometric Mean:**

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

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

# Noise Removal Examples

Original  
Image

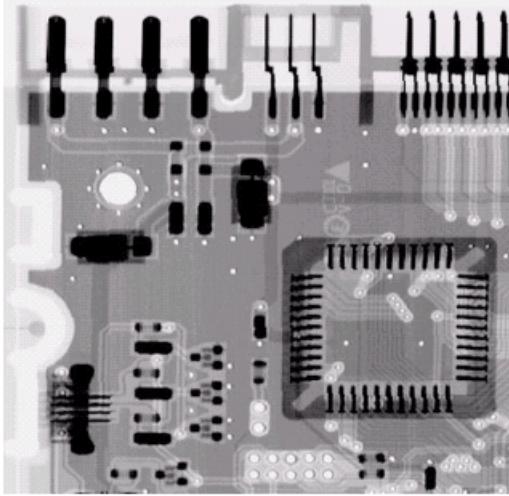
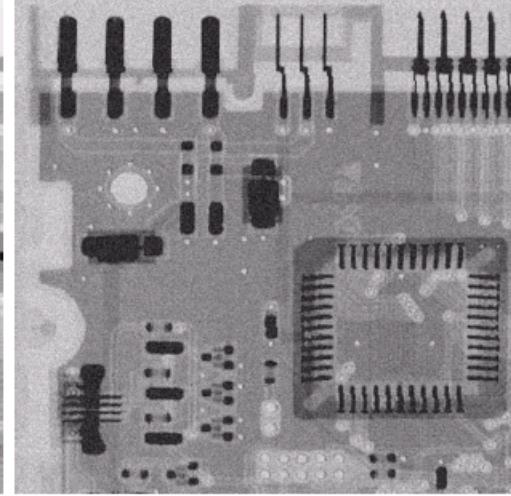
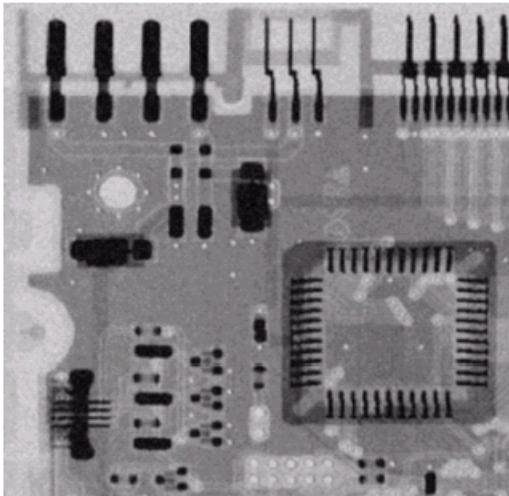


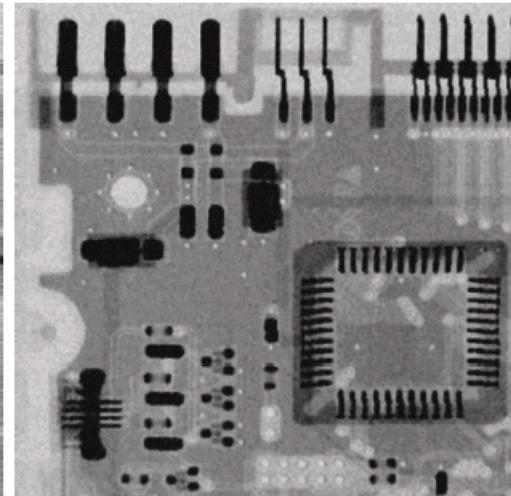
Image  
Corrupted  
By Gaussian  
Noise

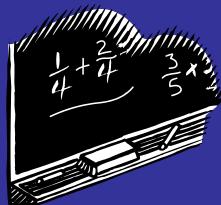


After A  $3 \times 3$   
Arithmetic  
Mean Filter



After A  $3 \times 3$   
Geometric  
Mean Filter





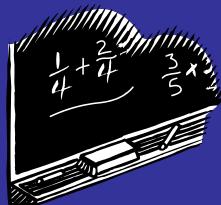
# Other Means (cont...)

## Harmonic Mean:

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

Works well for salt noise, but fails for pepper noise

Also does well for other kinds of noise such as Gaussian noise



# Other Means (cont...)

## Contraharmonic Mean:

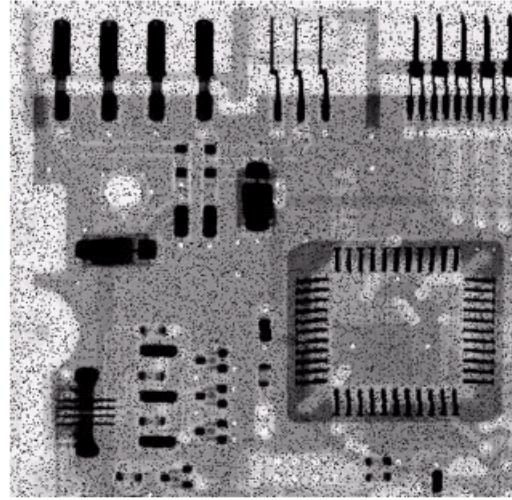
$$\hat{f}(x, y) = \frac{\sum_{(s,t) \in S_{xy}} g(s, t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} g(s, t)^Q}$$

$Q$  is the *order* of the filter and adjusting its value changes the filter's behaviour

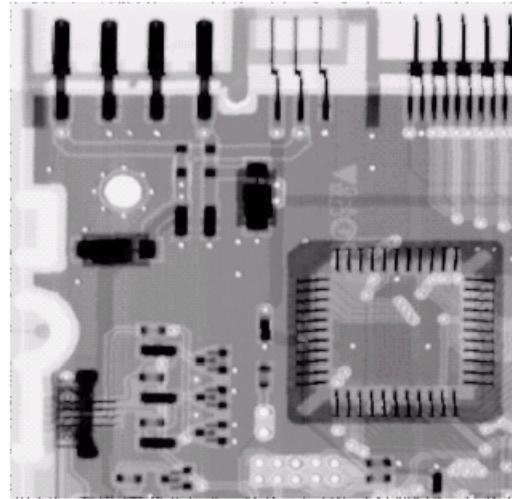
Positive values of  $Q$  eliminate pepper noise  
Negative values of  $Q$  eliminate salt noise

# Noise Removal Examples (cont...)

Image  
Corrupted  
By Pepper  
Noise



Result of  
Filtering Above  
With  $3 \times 3$   
Contraharmonic  
 $Q=1.5$



# Noise Removal Examples (cont...)

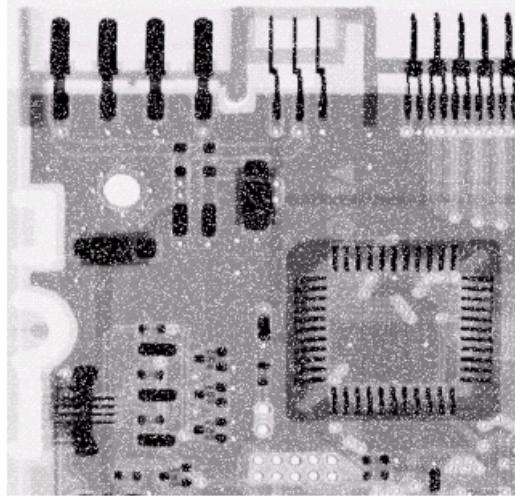
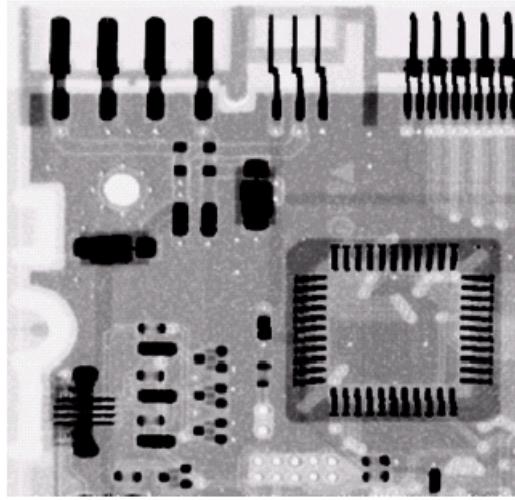


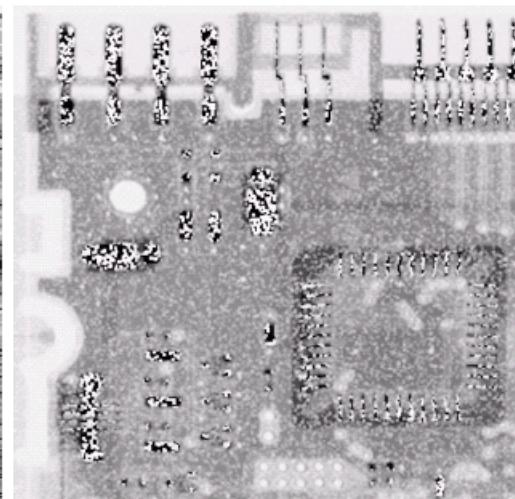
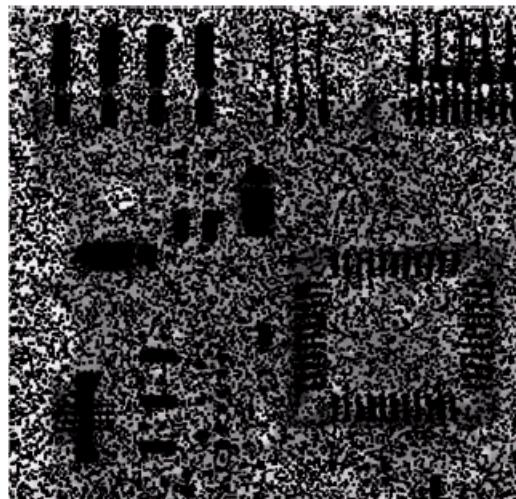
Image  
Corrupted  
By Salt  
Noise



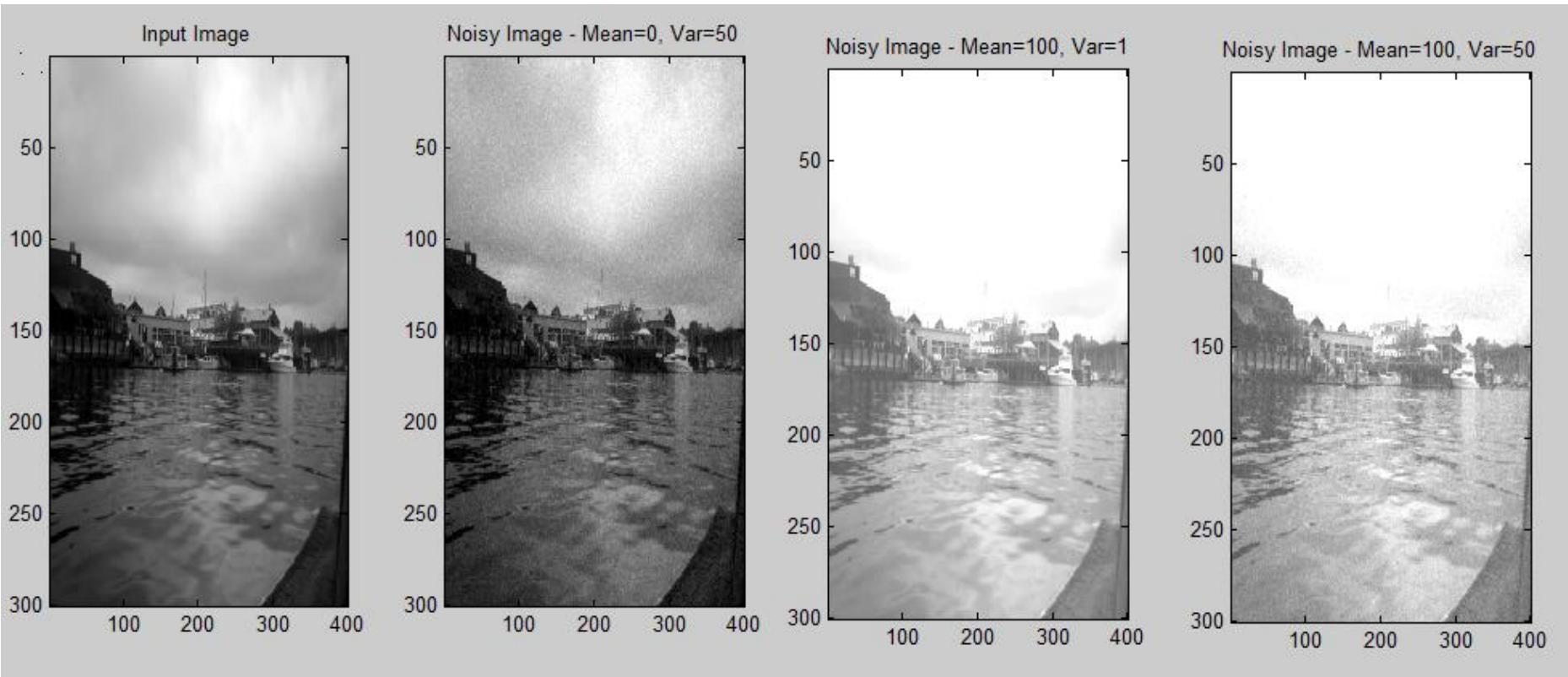
Result of  
Filtering Above  
With  $3 \times 3$   
Contraharmonic  
 $Q=-1.5$

# Contraharmonic Filter: Caution!

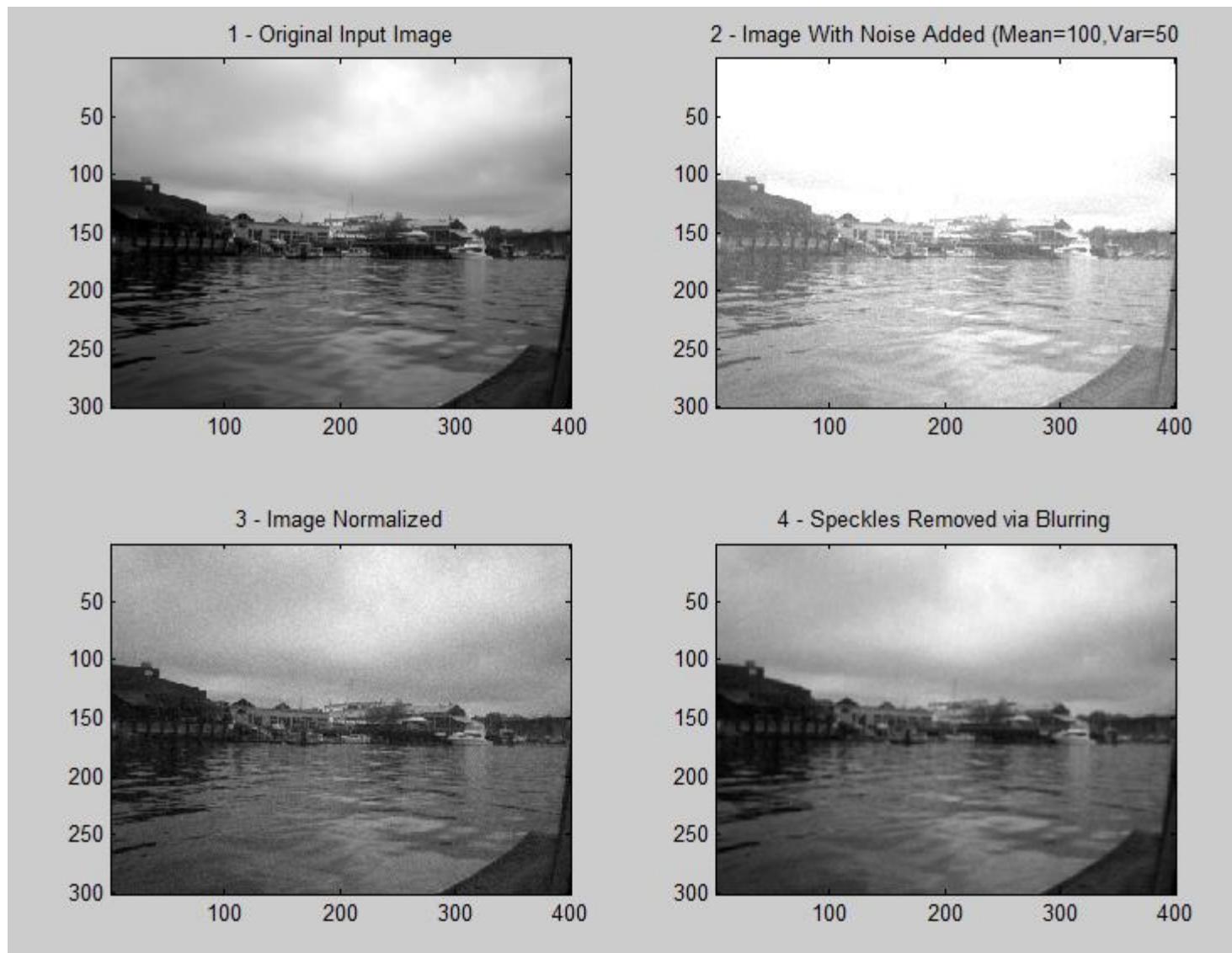
Choosing the wrong value for Q when using the contraharmonic filter can have drastic results



# Filtering of Gaussian noise



# Filtering of Gaussian noise (contd.)



# Rank Filters (non linear filtering)

Spatial filters that are based on ordering the pixel values that make up the neighbourhood operated on by the filter

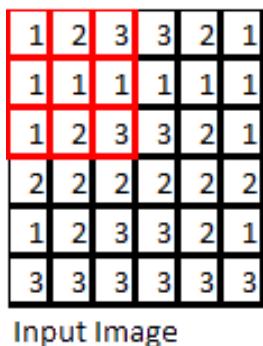
Useful spatial filters include

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

## Median Filter:

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

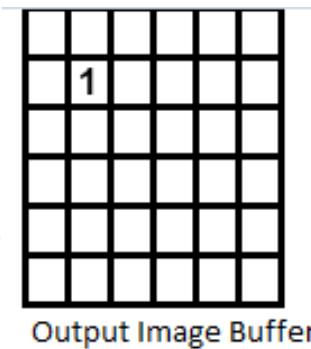
- Excellent at noise removal, without the smoothing effects that can occur with other smoothing filters
  - Particularly good when salt and pepper noise is present



1	2	3
1	1	1
1	2	3

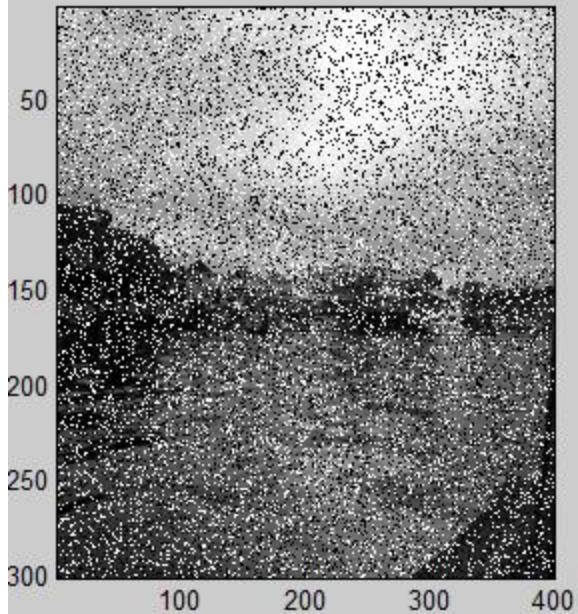
1  
1  
1  
1  
1  
2  
2  
3  
3

← Median value  
(Sort the pixels by value FIRST)

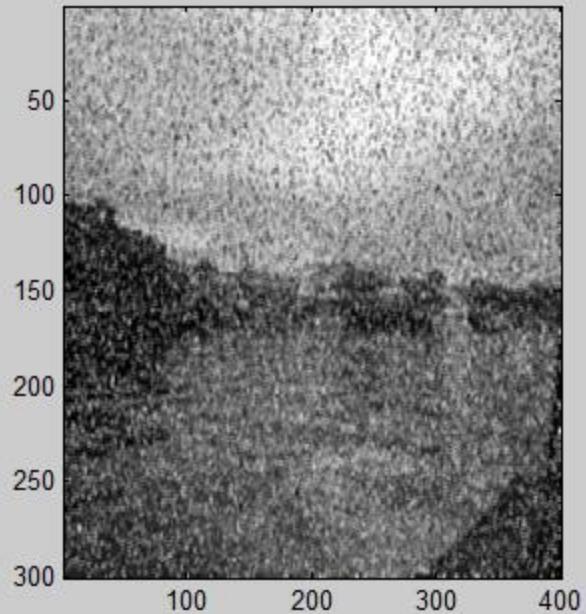


# Mean Vs. Median filter

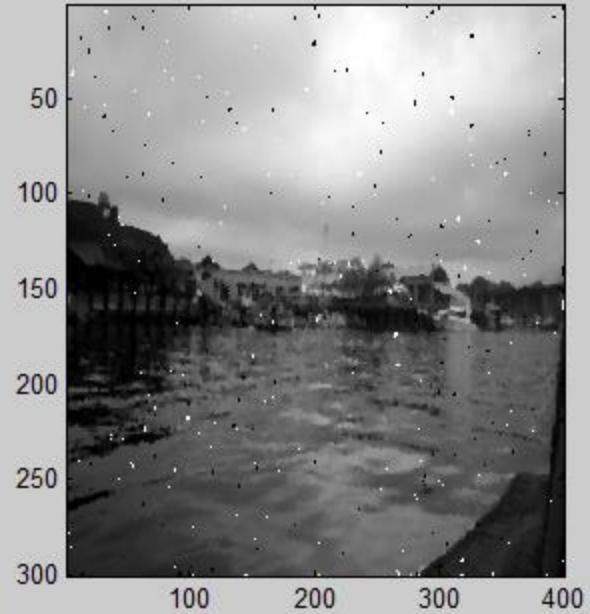
Input Image with 25% Salt & Pepper Noise



Noise Removed via 3x3 Smoothing Filter

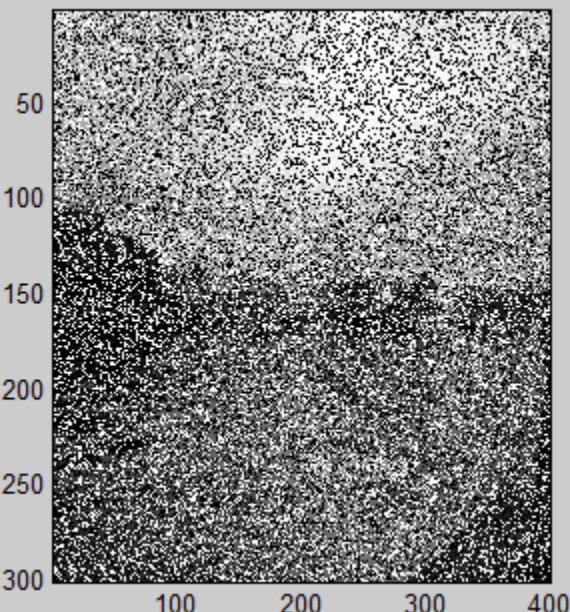


Noise Removed via 3x3 Median Filter

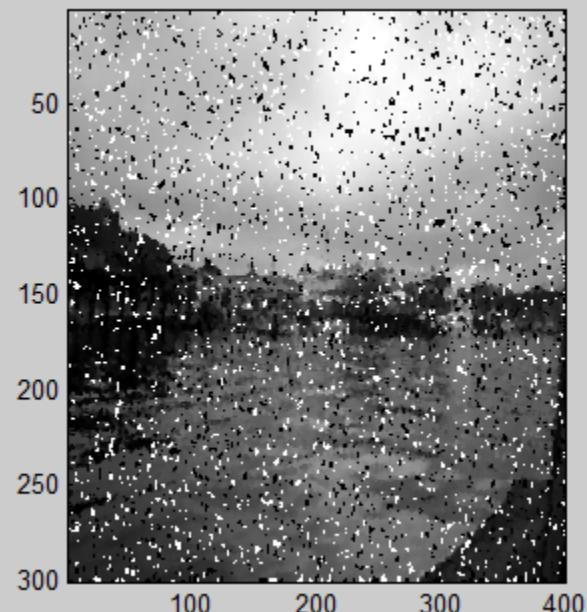


# Noise Removal Examples

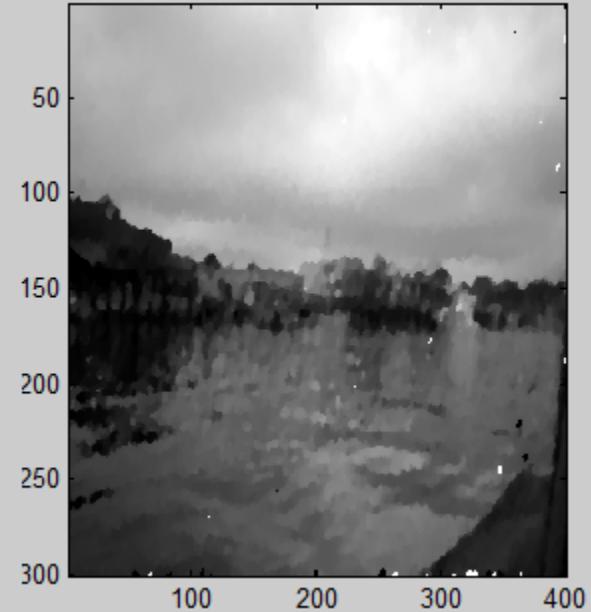
Input Image with 50% Salt & Pepper Noise



Noise Removed via 3x3 Median Filter

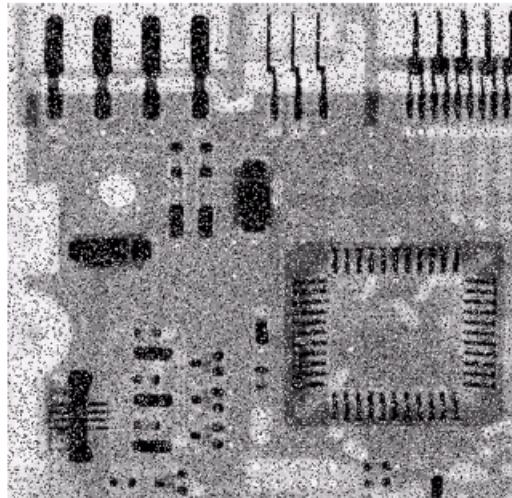


Noise Removed via 5x5 Median Filter

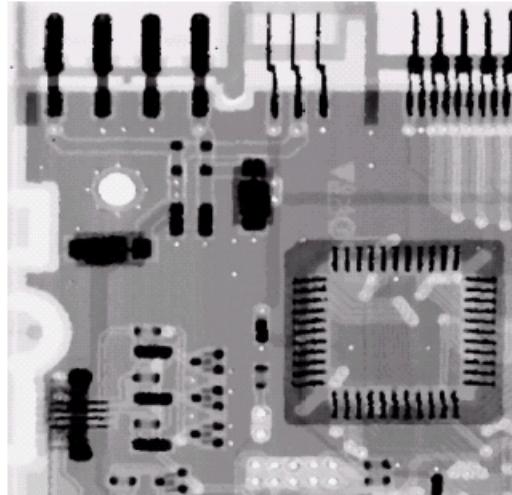


# Noise Removal Examples (contd)

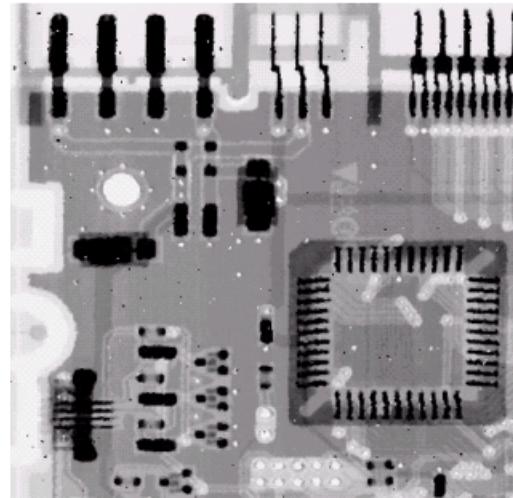
Image  
Corrupted  
By Salt And  
Pepper Noise



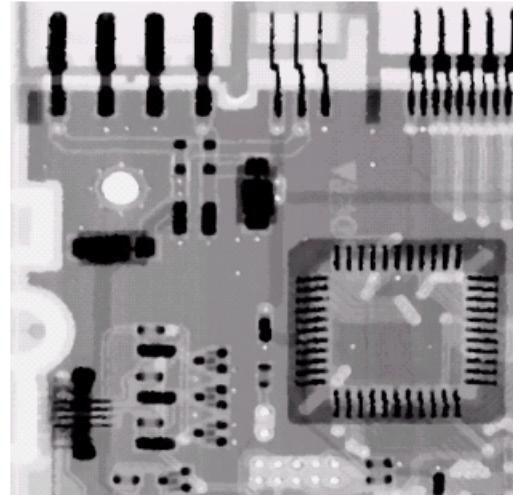
Result of 2  
Passes With  
A  $3 \times 3$  Median  
Filter



Result of 1  
Pass With A  
 $3 \times 3$  Median  
Filter



Result of 3  
Passes With  
A  $3 \times 3$  Median  
Filter



# Max and Min Filter

## Max Filter:

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

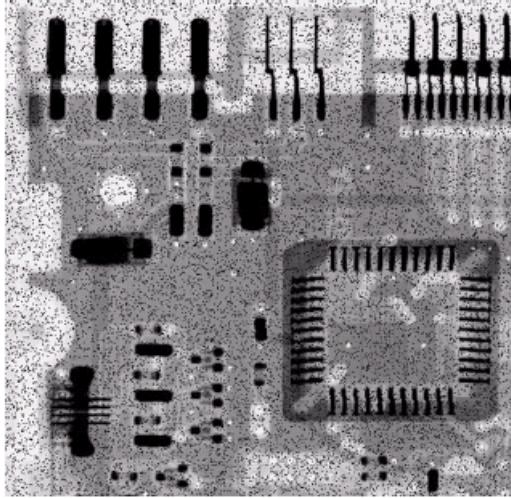
## Min Filter:

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

Max filter is good for pepper noise and min is good for salt noise

# Noise Removal Examples (cont...)

Image  
Corrupted  
By Pepper  
Noise



Result Of  
Filtering  
Above  
With A  $3 \times 3$   
Max Filter

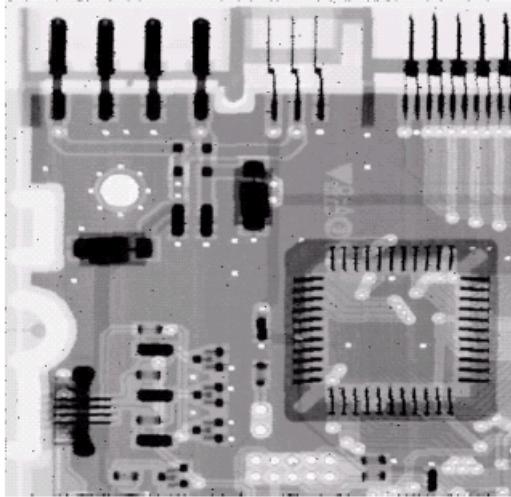
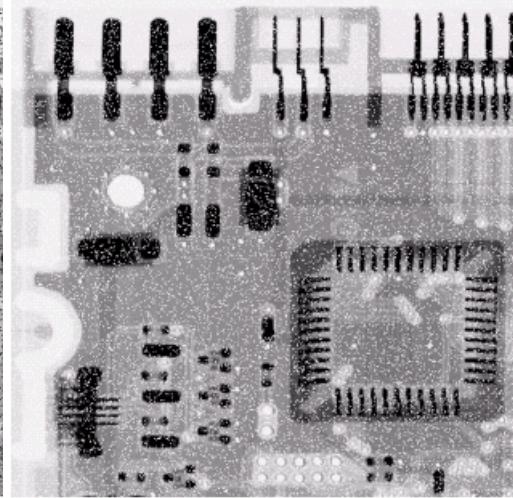
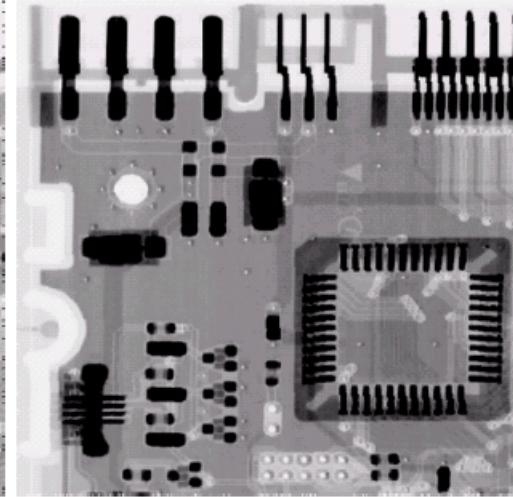


Image  
Corrupted  
By Salt  
Noise



Result Of  
Filtering  
Above  
With A  $3 \times 3$   
Min Filter



# Alpha-Trimmed Mean Filter

## Alpha-Trimmed Mean Filter:

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

We can delete the  $d/2$  lowest and  $d/2$  highest grey levels

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

# Noise Removal Examples (cont...)

Image  
Corrupted  
By Uniform  
Noise

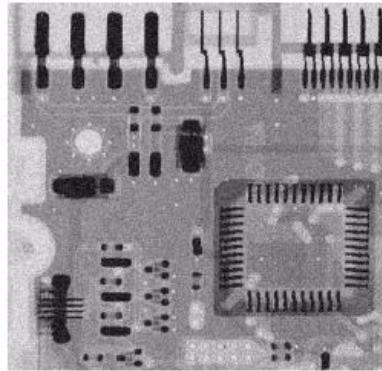
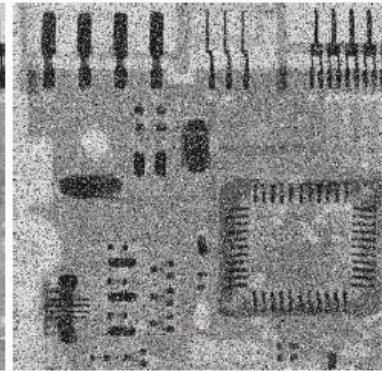
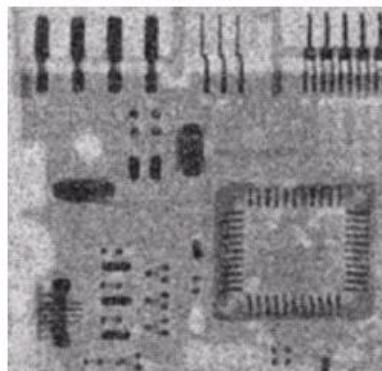


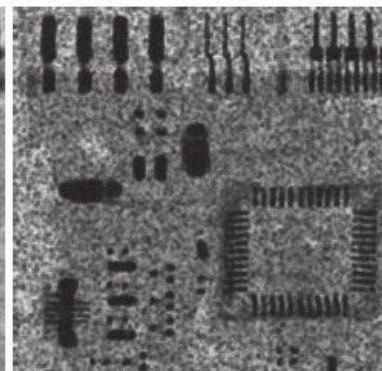
Image Further  
Corrupted  
By Salt and  
Pepper Noise



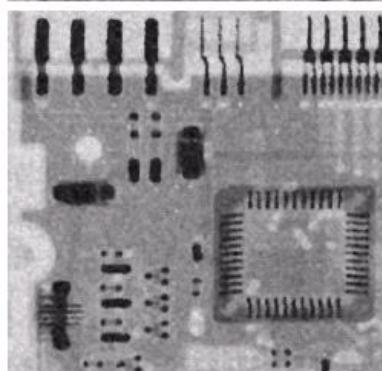
Filtered By  
5\*5 Arithmetic  
Mean Filter



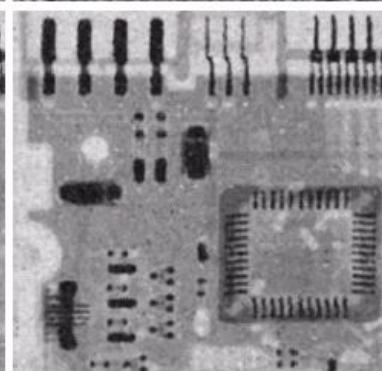
Filtered By  
5\*5 Geometric  
Mean Filter



Filtered By  
5\*5 Median  
Filter



Filtered By  
5\*5 Alpha-Trimmed  
Mean Filter



# Adaptive Filters

The filters discussed so far are applied to an entire image without any regard for how image characteristics vary from one point to another

The behaviour of **adaptive filters** changes depending on the characteristics of the image inside the filter region

# Adaptive filter: Neighborhood-based

## Adaptive local noise reduction filter

- Response based on 4 quantities
  - Local variance, **variance of noise**,  $g(x,y)$ , and local mean
- Behavior of filter
  - If variance of noise is zero, return  $g(x,y)$
  - If the local variance is high compared to the variance of noise, return a value close to  $g(x,y)$
  - If the two variances are equal, return the mean value

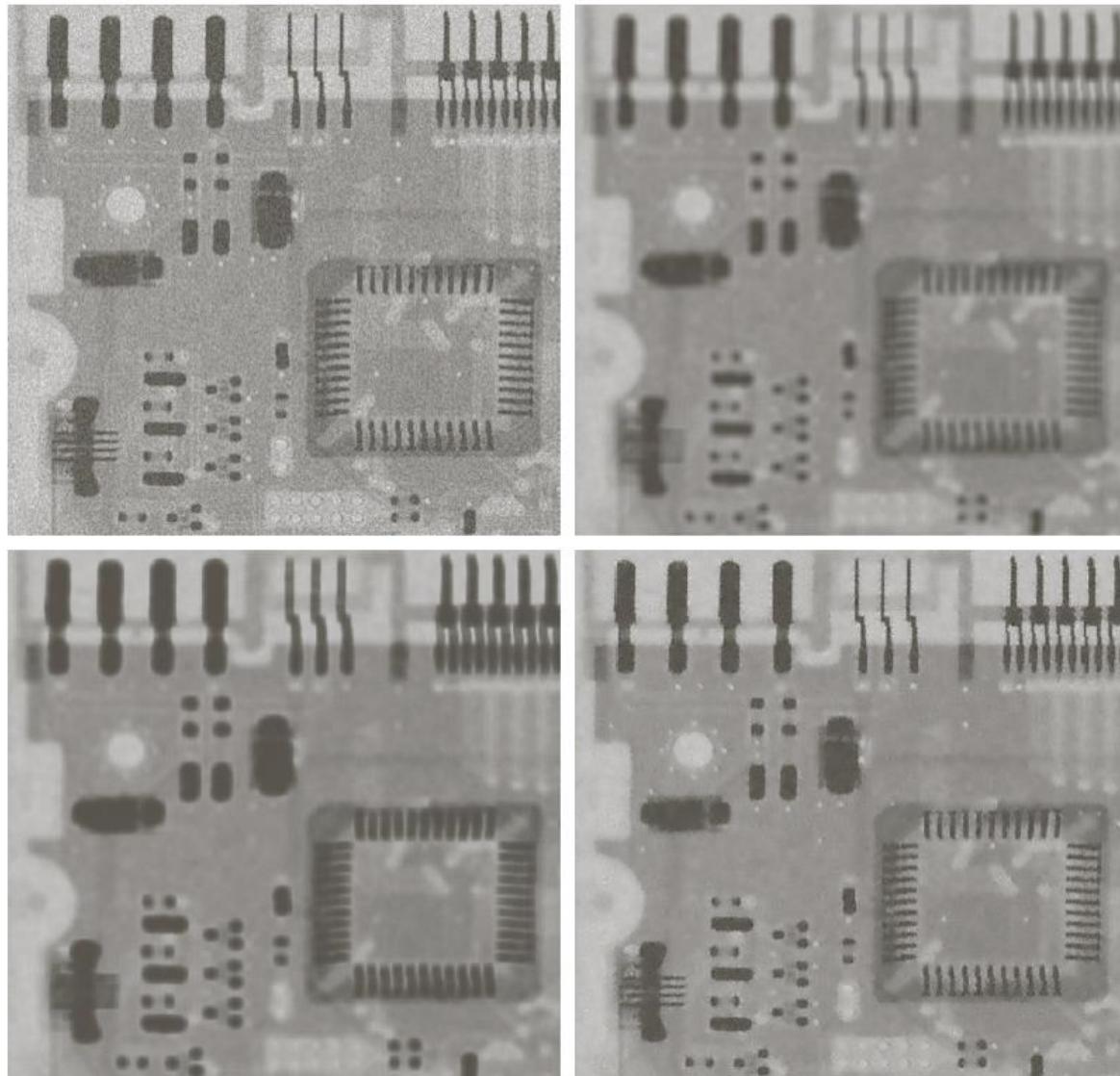
$$\hat{f}(x,y) = g(x,y) - \frac{S_h^2}{S_L^2} [g(x,y) - m_L]$$

# Adaptive filter: Neighborhood-based

a  
b  
c  
d

**FIGURE 5.13**

- (a) Image corrupted by additive Gaussian noise of zero mean and variance 1000.  
(b) Result of arithmetic mean filtering.  
(c) Result of geometric mean filtering.  
(d) Result of adaptive noise reduction filtering. All filters were of size  $7 \times 7$ .



# Adaptive Median Filtering

The median filter performs relatively well on impulse noise as long as the spatial density of the impulse noise is not large

The adaptive median filter can handle much more spatially dense impulse noise, and does less distortion

The key insight in the adaptive median filter is that the filter size changes depending on the characteristics of the image

# Adaptive Median Filtering (cont...)

The adaptive median filter has following purposes:

- Remove spatially dense impulse noise
- Reduce distortion

First examine the following notation:

- $z_{min}$  = minimum grey level in  $S_{xy}$
- $z_{max}$  = maximum grey level in  $S_{xy}$
- $z_{med}$  = median of grey levels in  $S_{xy}$
- $z_{xy}$  = grey level at coordinates  $(x, y)$
- $S_{max}$  = maximum allowed size of  $S_{xy}$

# Adaptive Median Filtering (cont...)

Level A:  $A1 = z_{med} - z_{min}$

$A2 = z_{med} - z_{max}$

If  $A1 > 0$  and  $A2 < 0$ , Go to level B

Else increase the window size

If window size  $\leq S_{max}$  repeat level A

Else output  $z_{med}$

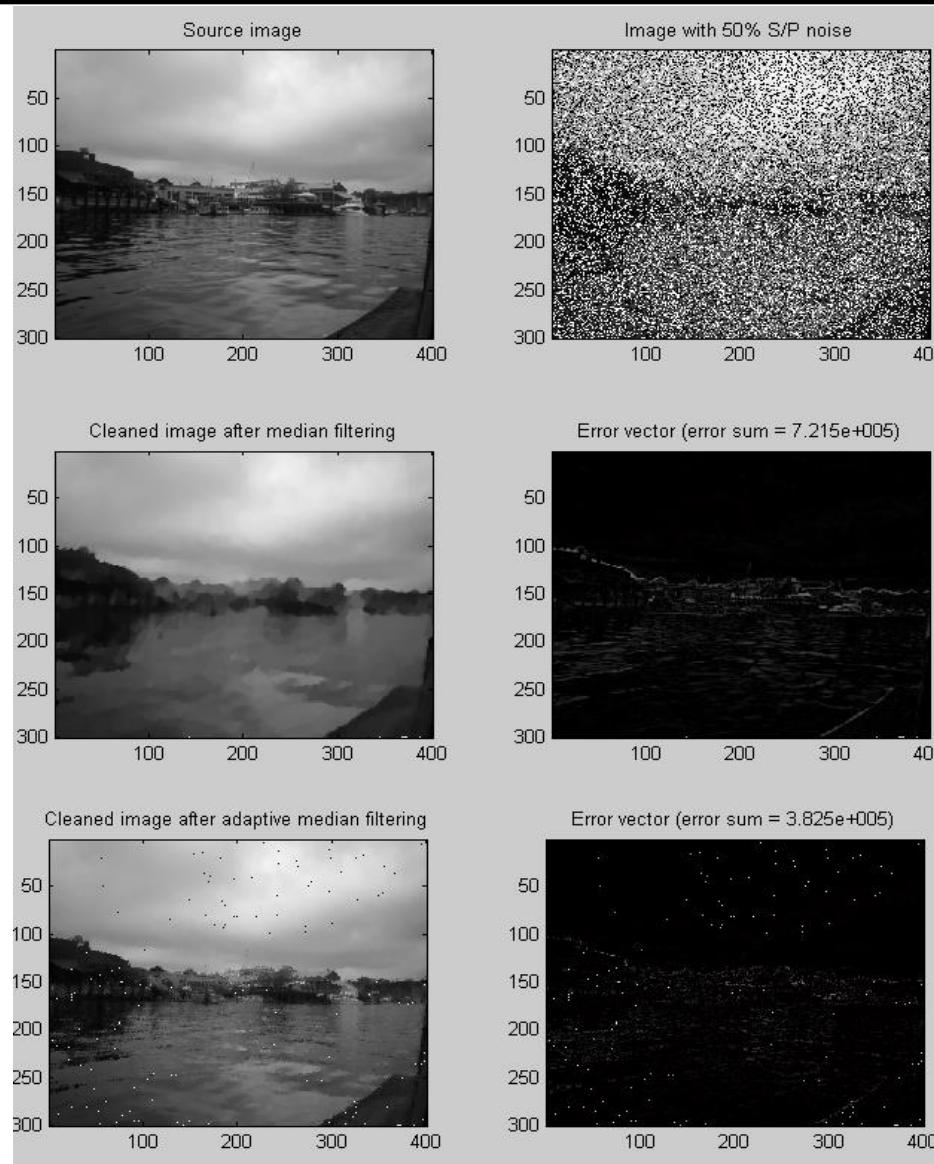
Level B:  $B1 = z_{xy} - z_{min}$

$B2 = z_{xy} - z_{max}$

If  $B1 > 0$  and  $B2 < 0$ , output  $z_{xy}$

Else output  $z_{med}$

# Simple Adaptive Median Filtering Example



Using only  
level B and  
filter 7 x 7

# Adaptive Median Filtering Examples

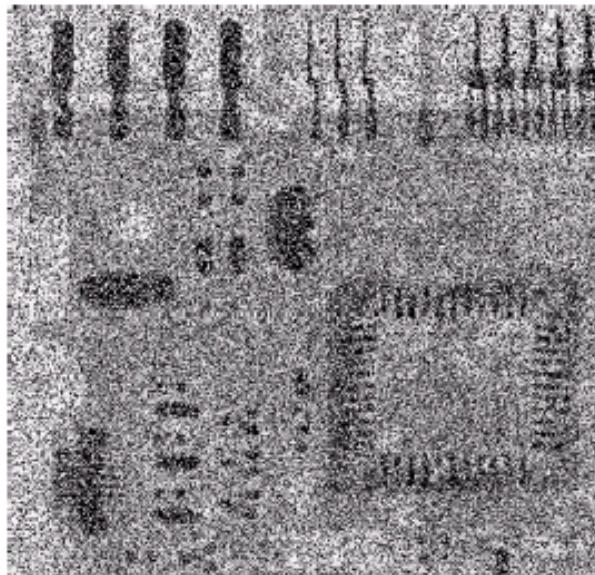
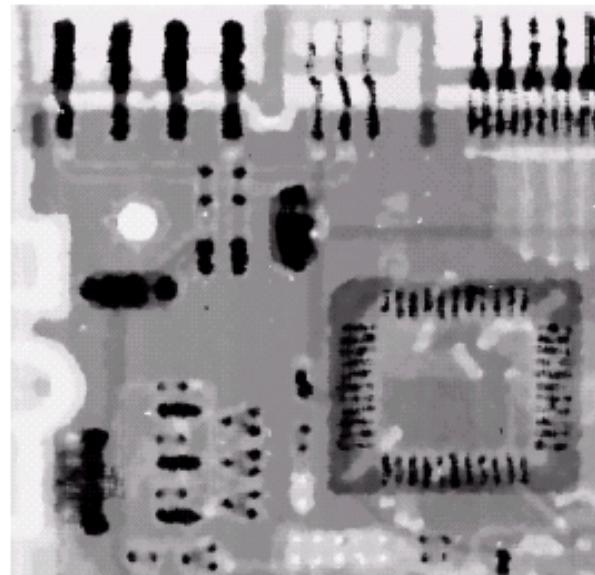
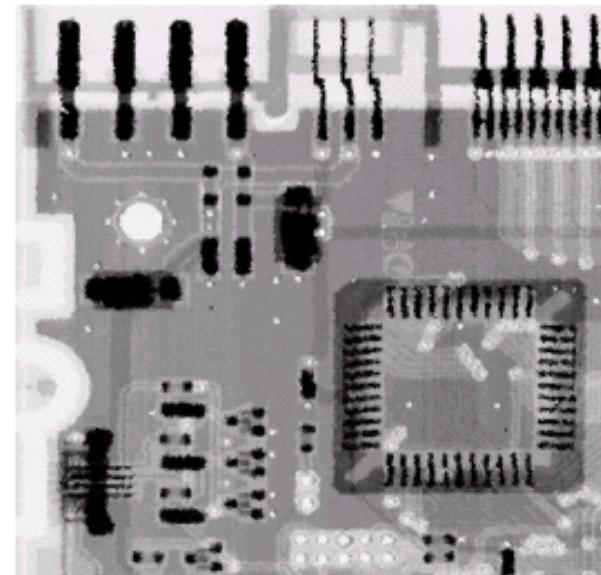


Image corrupted by salt  
and pepper noise with  
probabilities  $P_a = P_b = 0.25$



Result of filtering with a 7  
\* 7 median filter



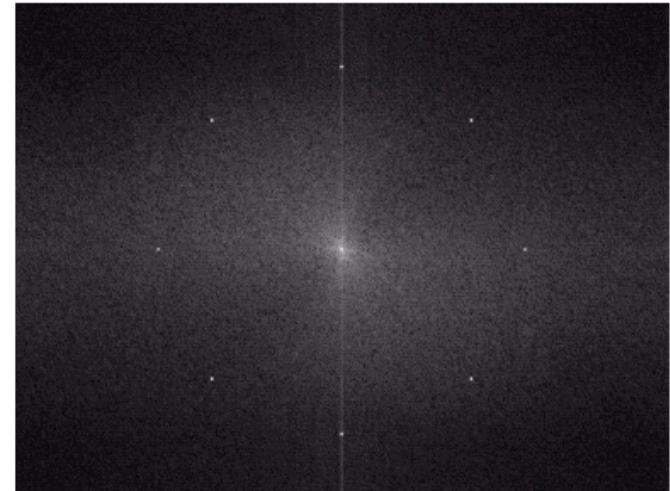
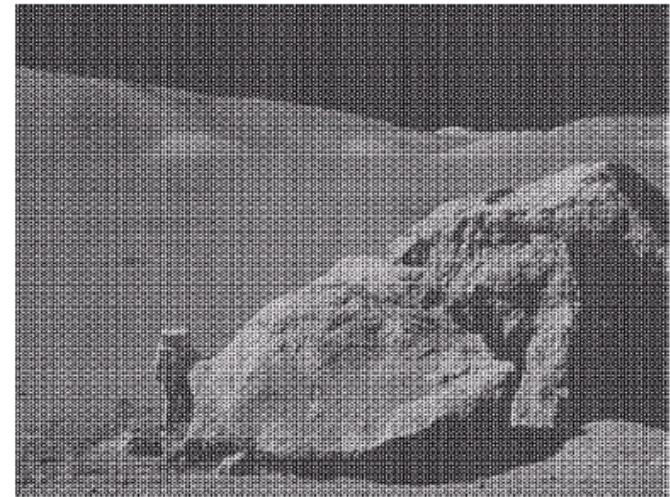
Result of adaptive median  
filtering with  $S_{max} = 7$

# Periodic Noise removal

Typically arises due to electrical or electromagnetic interference

Gives rise to regular noise patterns in an image

Frequency domain techniques in the Fourier domain are most effective at removing periodic noise



# Band Reject Filters

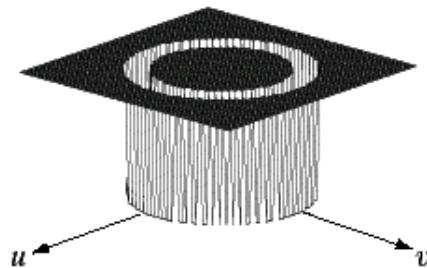
Removing periodic noise from an image involves removing a particular range of frequencies from that image

*Band reject* filters can be used for this purpose  
An ideal band reject filter is given as follows:

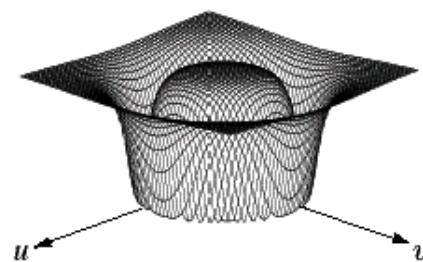
$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u, v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u, v) > D_0 + \frac{W}{2} \end{cases}$$

# Band Reject Filters (cont...)

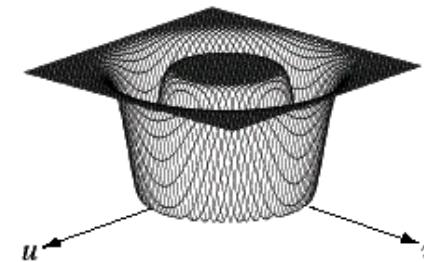
The ideal band reject filter is shown below, along with Butterworth and Gaussian versions



Ideal Band  
Reject Filter



Butterworth  
Band Reject  
Filter (of order 1)

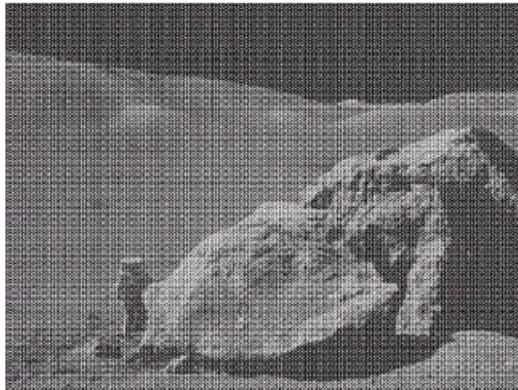


Gaussian  
Band Reject  
Filter

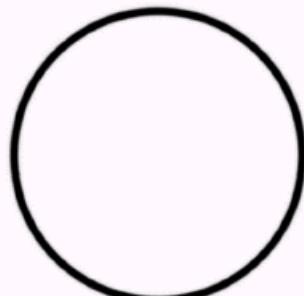
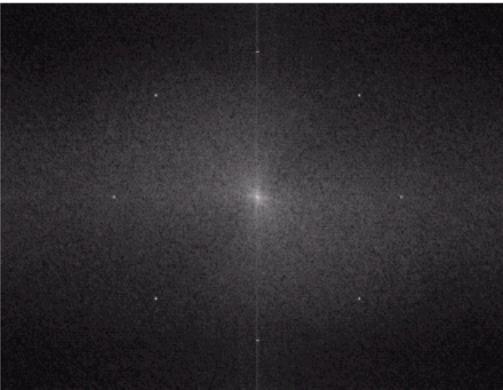
Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u, v) = \frac{1}{1 + \left[ \frac{DW}{D^2 - D_0^2} \right]^{2n}}$	$H(u, v) = 1 - e^{-\left[ \frac{D^2 - D_0^2}{DW} \right]^2}$

# Band Reject Filter Example

Image corrupted by sinusoidal noise



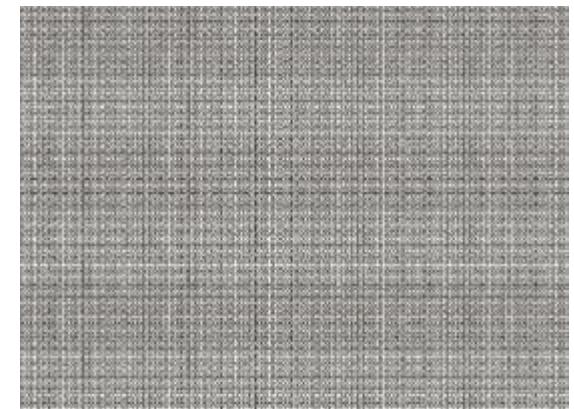
Fourier spectrum of corrupted image



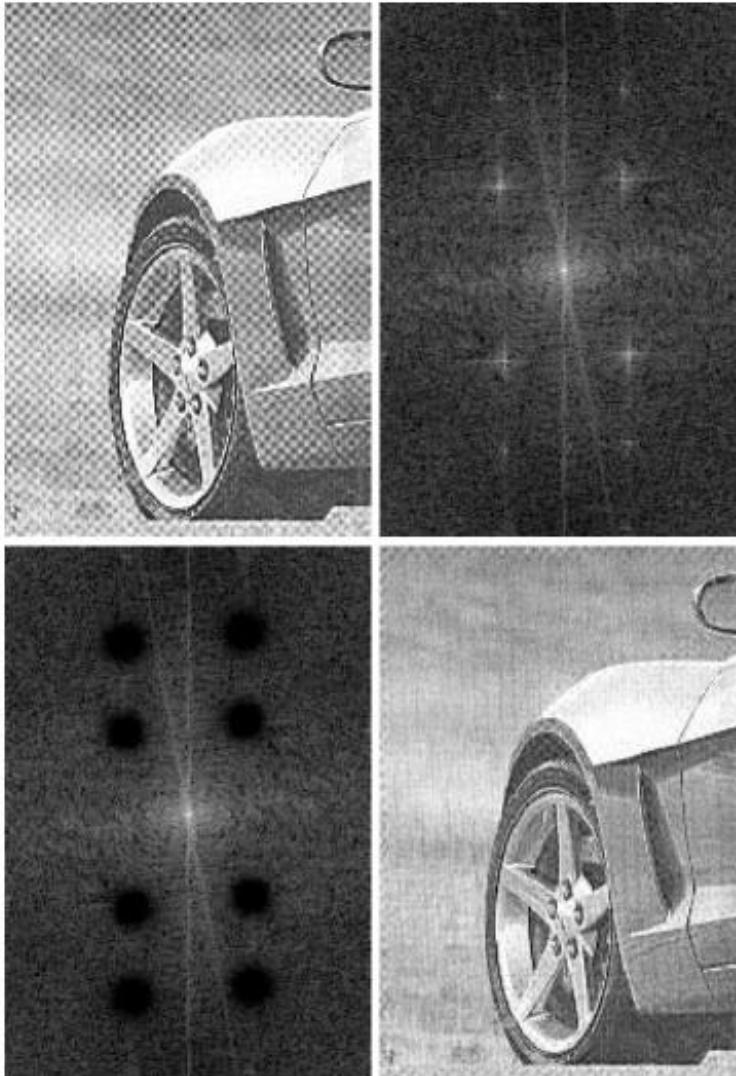
Butterworth band reject filter



Filtered image



# Notch Filter



a b  
c d

**FIGURE 4.64**

- (a) Sampled newspaper image showing a moiré pattern.
- (b) Spectrum.
- (c) Butterworth notch reject filter multiplied by the Fourier transform.
- (d) Filtered image.

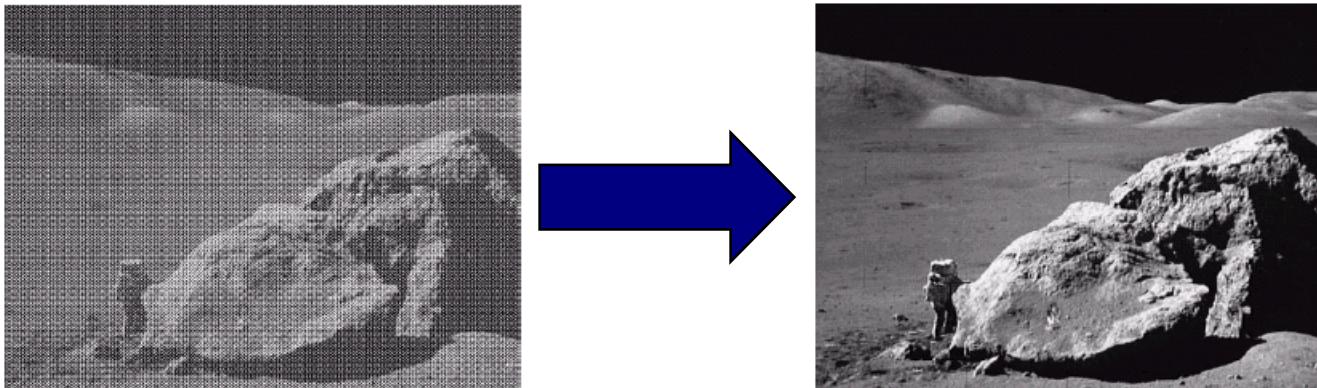
# Image & Video Processing

Image Restoration:  
Noise Removal

# What is Image Restoration?

Image restoration attempts to restore images that have been degraded

- Identify the degradation process and attempt to remove it in order to go back to the “original”
- Similar to image enhancement, but more objective

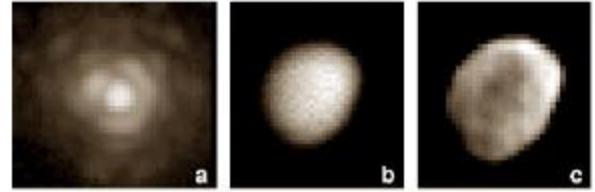


# Applications

Started from the 1950s

- Scientific explorations
- Legal investigations
- Film making and archival
- Image and video (de-)coding
- ...
- Consumer photography

Example of image restoration  
Asteroid Vesta



# Degradation causes

## Degradation examples:



- original



- optical blur



- motion blur



- spatial quantization (discrete pixels)

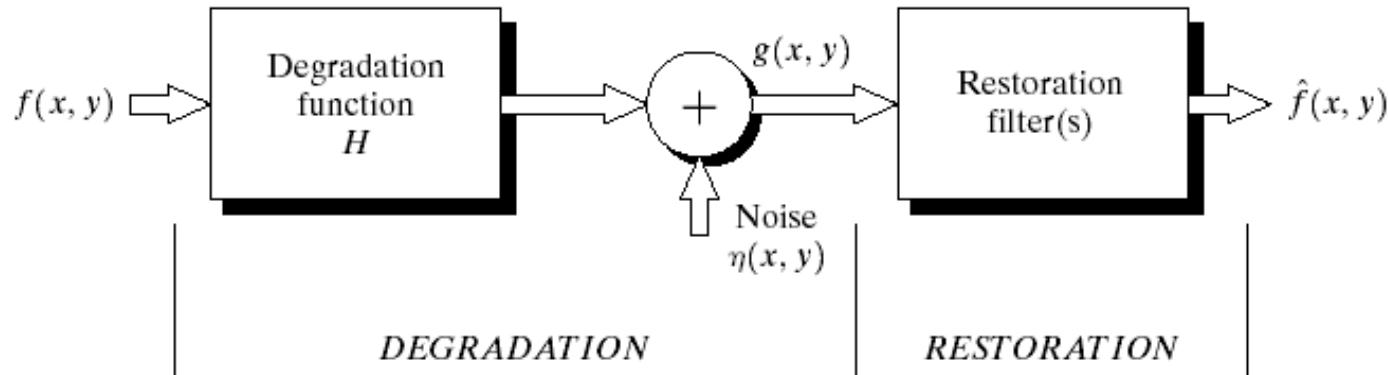


- additive intensity noise

### Causes:

- Camera: translation, shake, out-of-focus ...
- Environment: scattered and reflected light
- Device noise: CCD/CMOS sensor and circuitry
- Quantization noise
- Transmission error

# A Model for Image Distortion/Restoration



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

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

where  $f(x, y)$  is the original image pixel,  $H$  is the degradation function,  $\eta(x, y)$  is the noise term and  $g(x, y)$  is the resulting noisy pixel

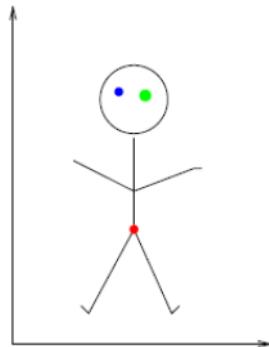
# Assumptions for the Distortion Model

- Noise
  - Independent of spatial location
    - Exception: periodic noise ..
  - Uncorrelated with image
- Degradation function  $H$ 
  - Linear
  - Position-invariant

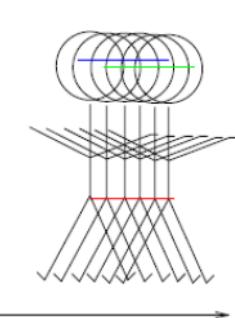
$$g(x, y) = h(x, y) * F(x, y) + \eta(x, y)$$

**Step #1: image degraded only by noise.**

Input Image

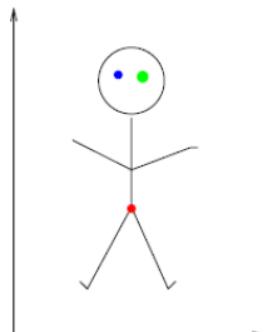


Blurred by Camera linear motion

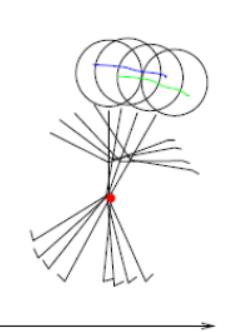


SPACE-INVARIANT RESPONSE - each point on image gives same response just shifted in position.

Input Image



Blurred by Camera Rotation

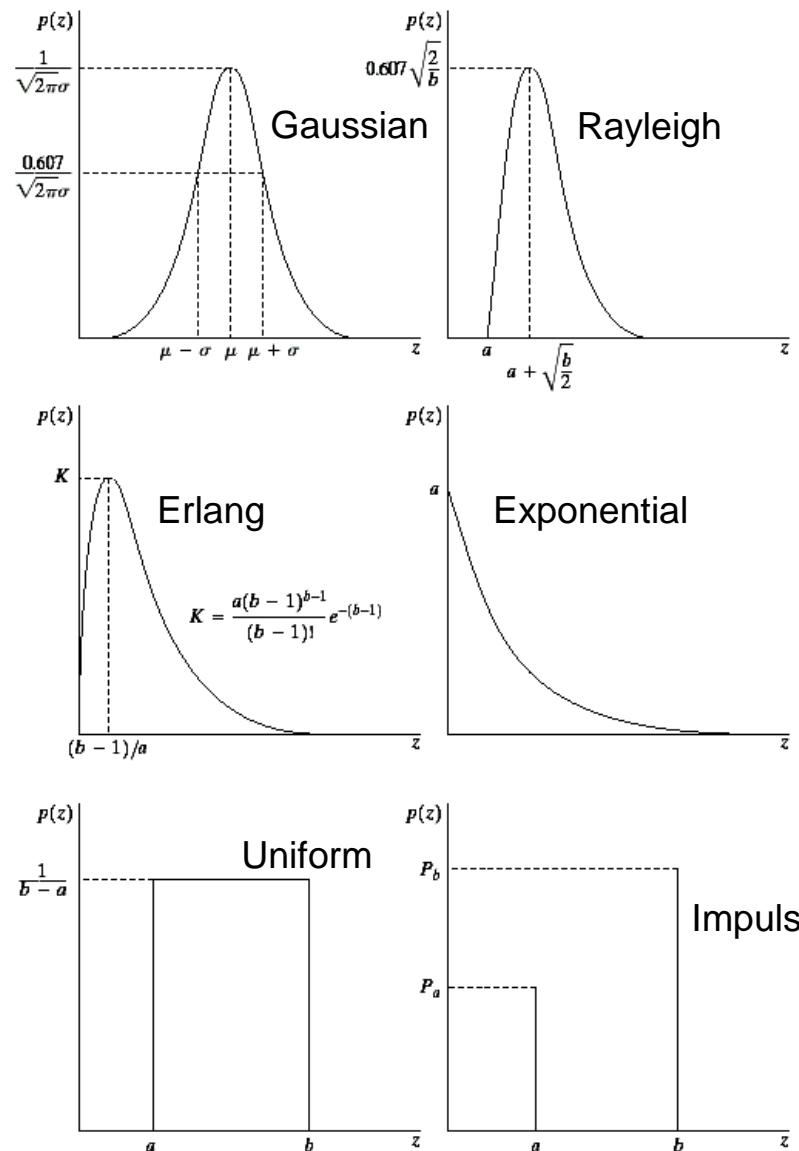


SPACE-VARIANT RESPONSE - each point on image gives a different response

# Noise Models

There are many different models for the image noise term  $\eta(x, y)$ :

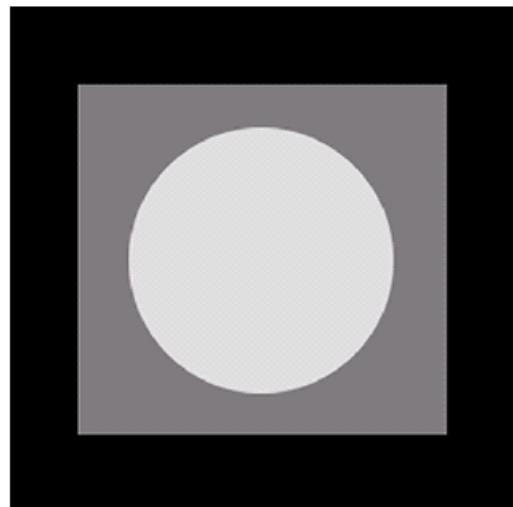
- Gaussian
  - Most common model
- Rayleigh
- Erlang
- Exponential
- Uniform
- Impulse
  - *Salt and pepper* noise



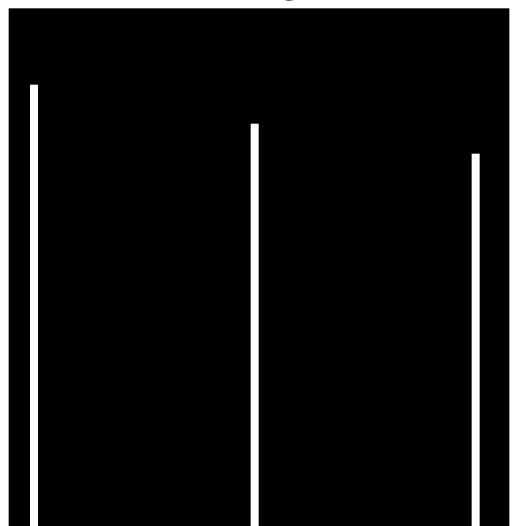
# Noise Example

The test pattern to the right is ideal for demonstrating the addition of noise

The following slides will show the result of adding noise based on various models to this image



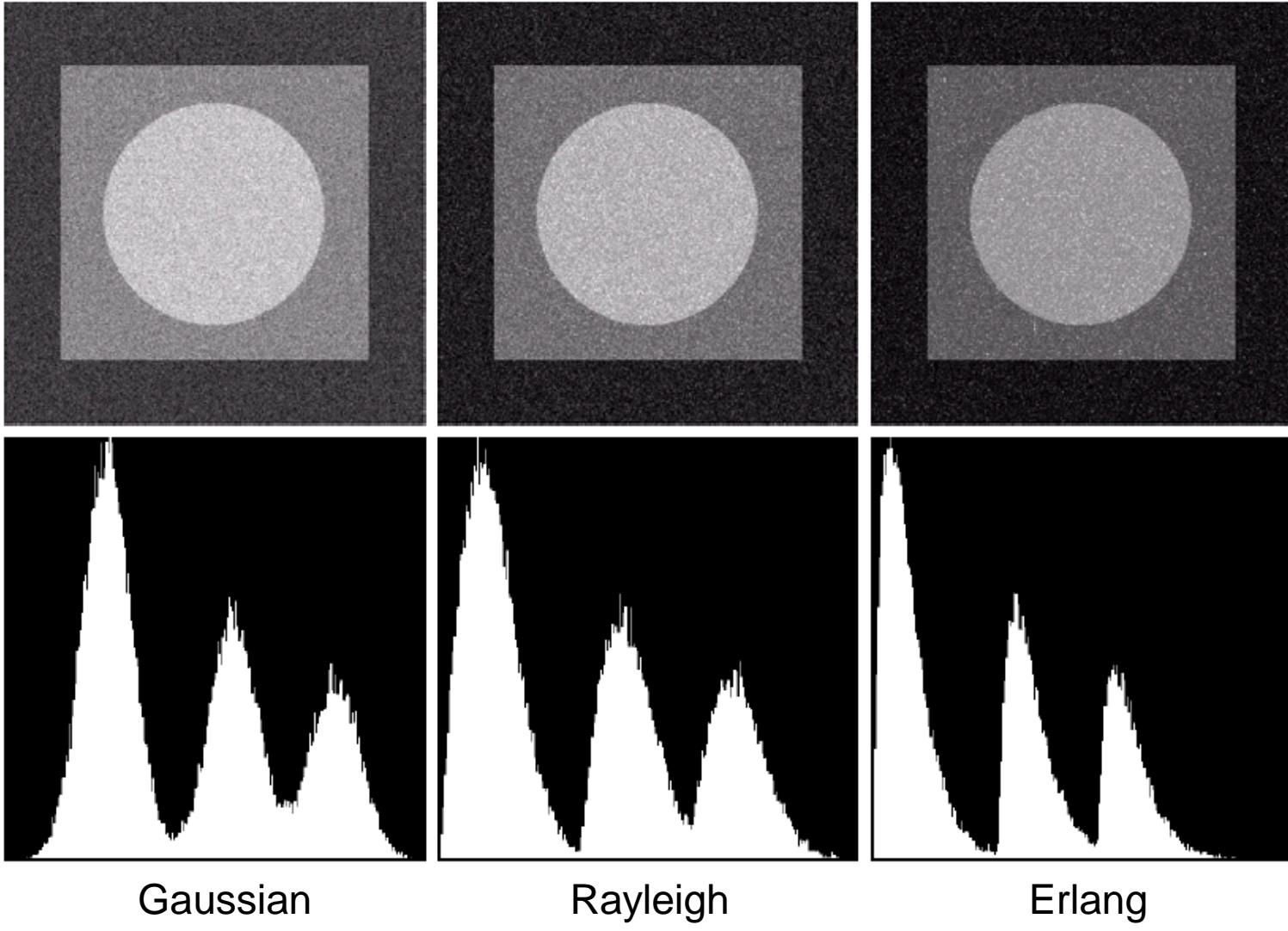
Image



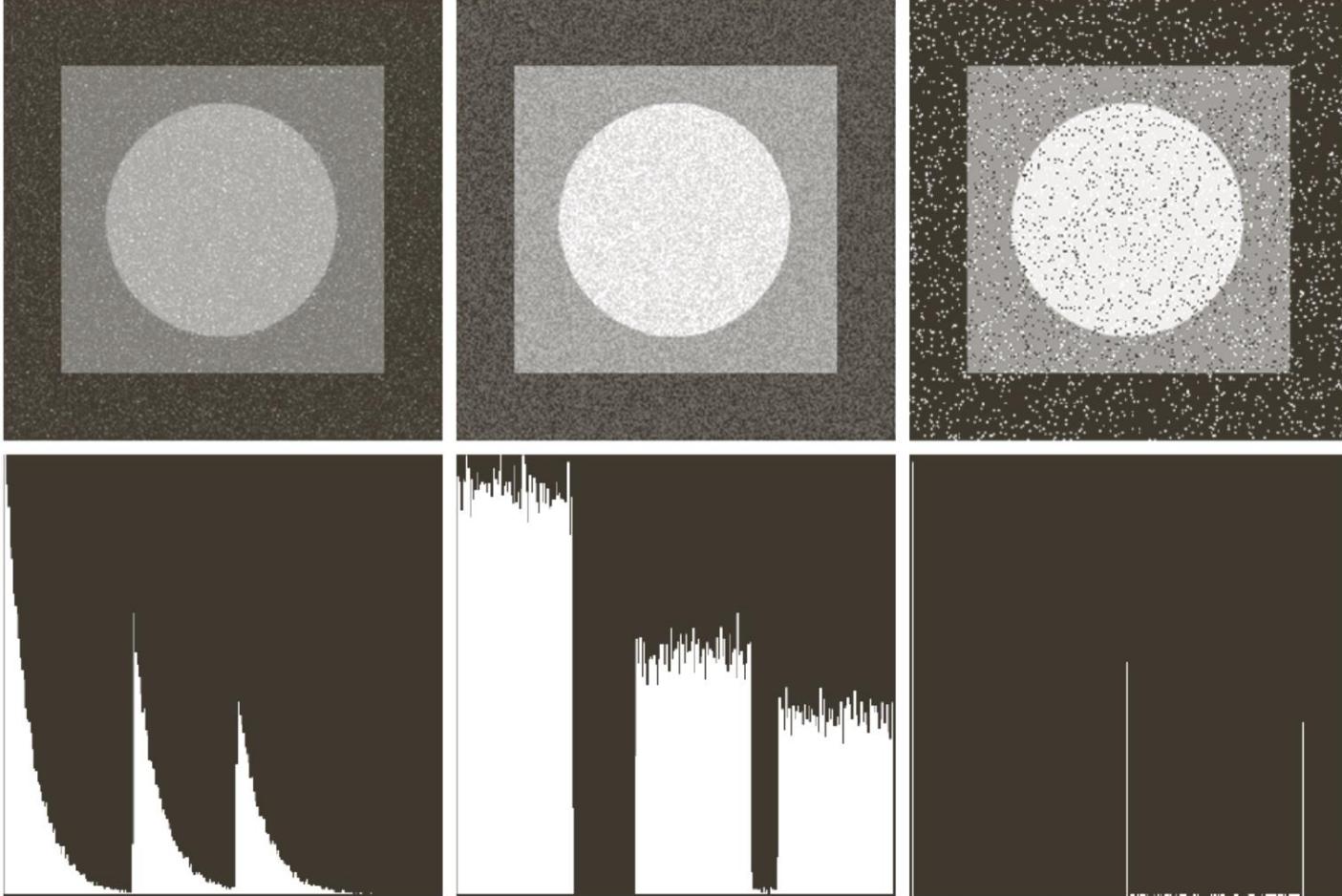
Histogram



# Noise Example (cont...)



# Noise Example (cont...)

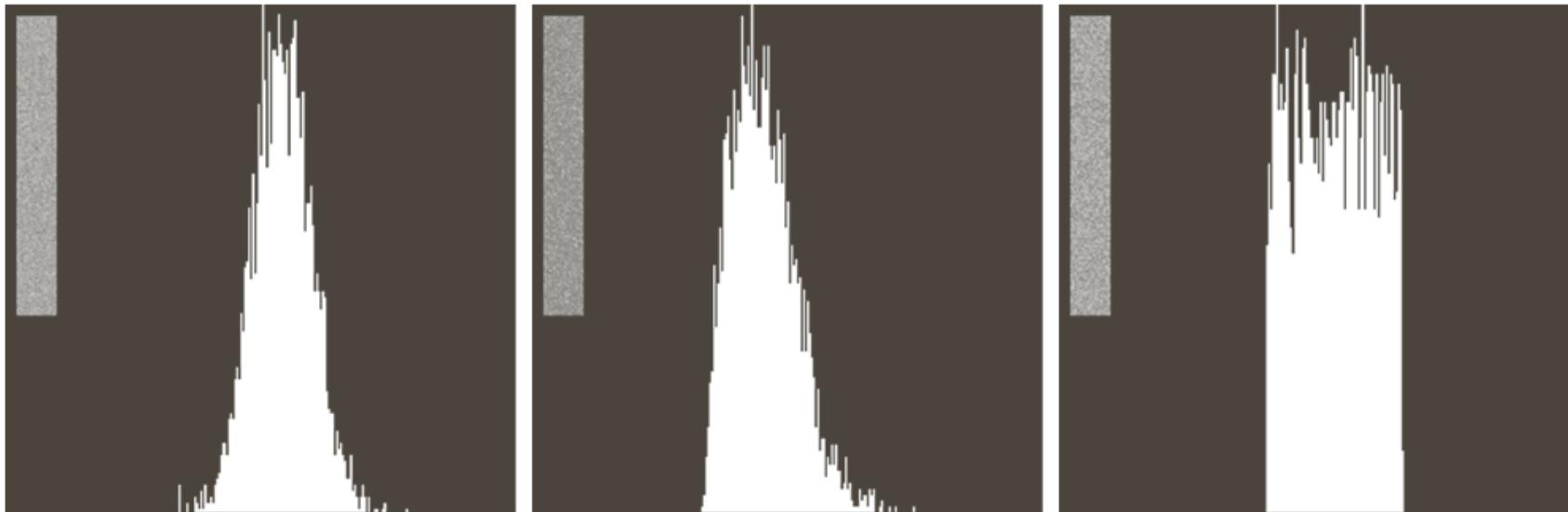


Exponential

Uniform

Impulse

# Estimation of noise parameters



a | b | c

**FIGURE 5.6** Histograms computed using small strips (shown as inserts) from (a) the Gaussian, (b) the Rayleigh, and (c) the uniform noisy images in Fig. 5.4.

# Filtering to Remove Noise

We can use spatial filters of different kinds to remove different kinds of noise

The *arithmetic mean* filter is a very simple one and is calculated as follows:

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

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

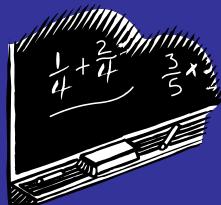
This is implemented as the simple smoothing filter

Blurs the image to remove noise

# Other Means

There are different kinds of mean filters all of which exhibit slightly different behaviour:

- Geometric Mean
- Harmonic Mean
- Contraharmonic Mean



# Other Means (cont...)

There are other variants on the mean which can give different performance

**Geometric Mean:**

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

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

# Noise Removal Examples

Original  
Image

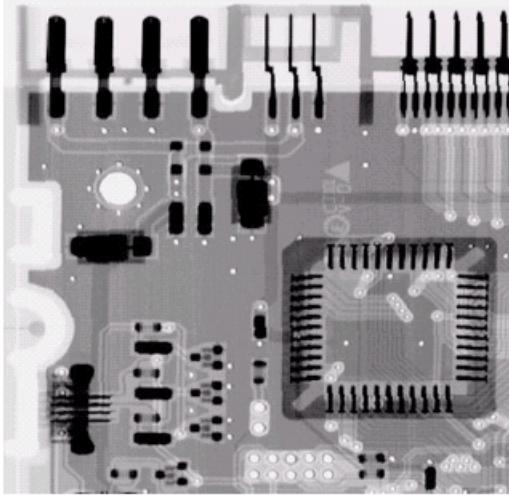
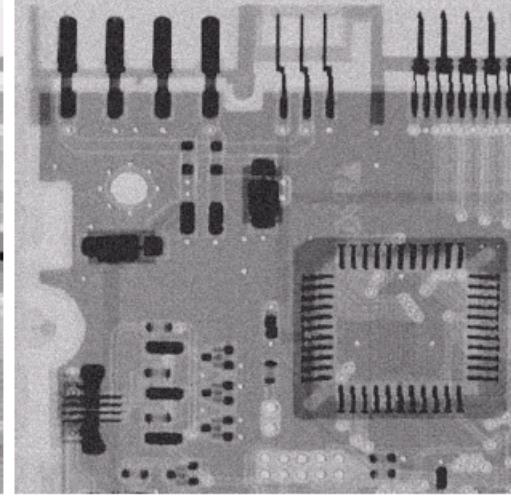
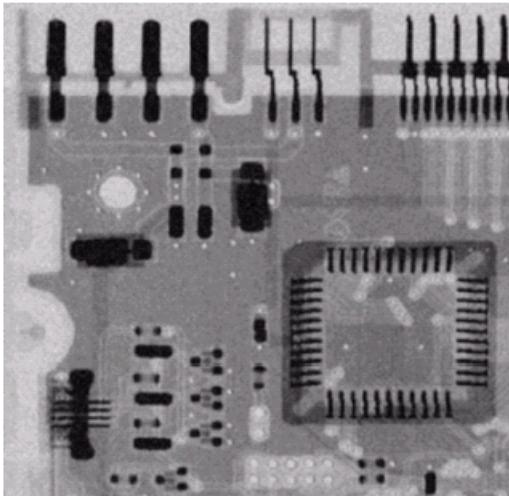


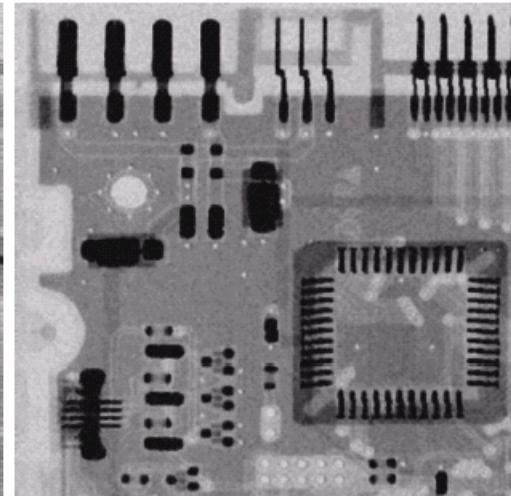
Image  
Corrupted  
By Gaussian  
Noise

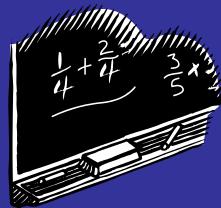


After A 3\*3  
Arithmetic  
Mean Filter



After A 3\*3  
Geometric  
Mean Filter





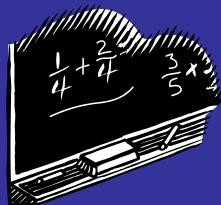
# Other Means (cont...)

## Harmonic Mean:

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

Works well for salt noise, but fails for pepper noise

Also does well for other kinds of noise such as Gaussian noise



# Other Means (cont...)

## Contraharmonic Mean:

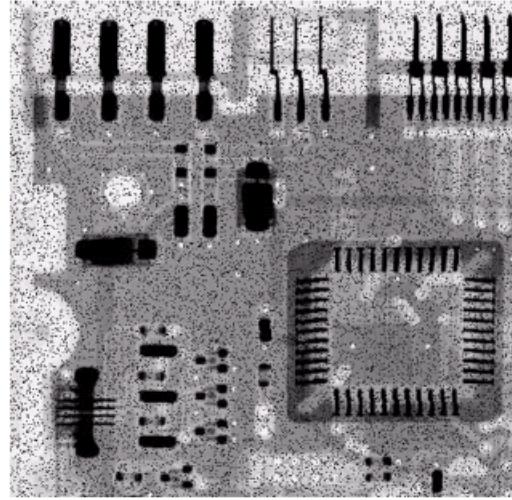
$$\hat{f}(x, y) = \frac{\sum_{(s,t) \in S_{xy}} g(s, t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} g(s, t)^Q}$$

$Q$  is the *order* of the filter and adjusting its value changes the filter's behaviour

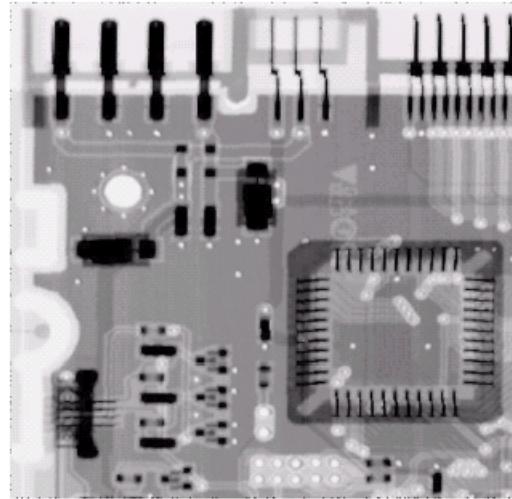
Positive values of  $Q$  eliminate pepper noise  
Negative values of  $Q$  eliminate salt noise

# Noise Removal Examples (cont...)

Image  
Corrupted  
By Pepper  
Noise



Result of  
Filtering Above  
With  $3 \times 3$   
Contraharmonic  
 $Q=1.5$



# Noise Removal Examples (cont...)

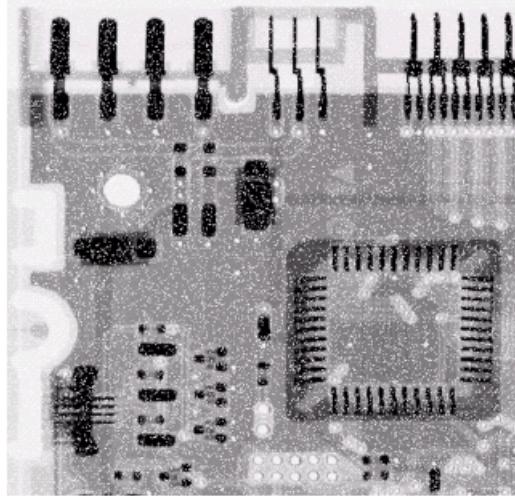
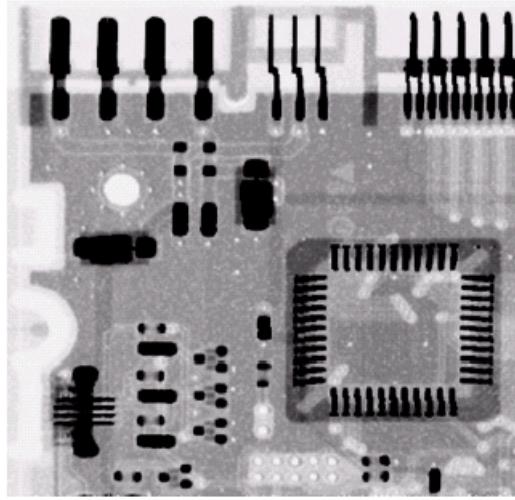


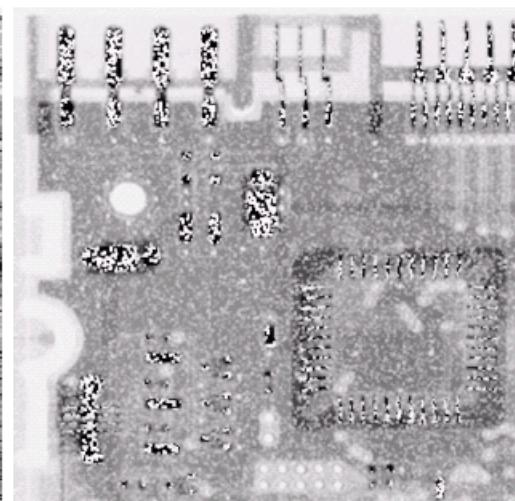
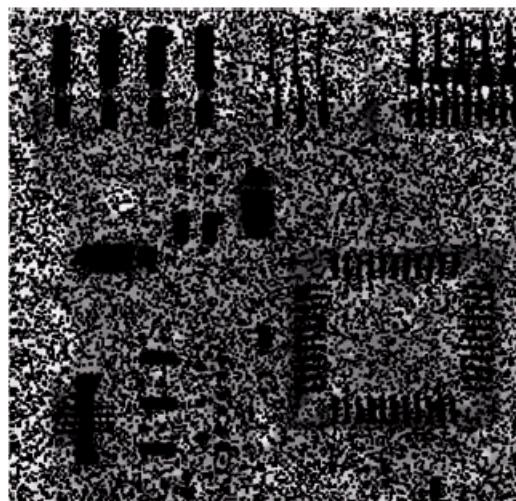
Image  
Corrupted  
By Salt  
Noise



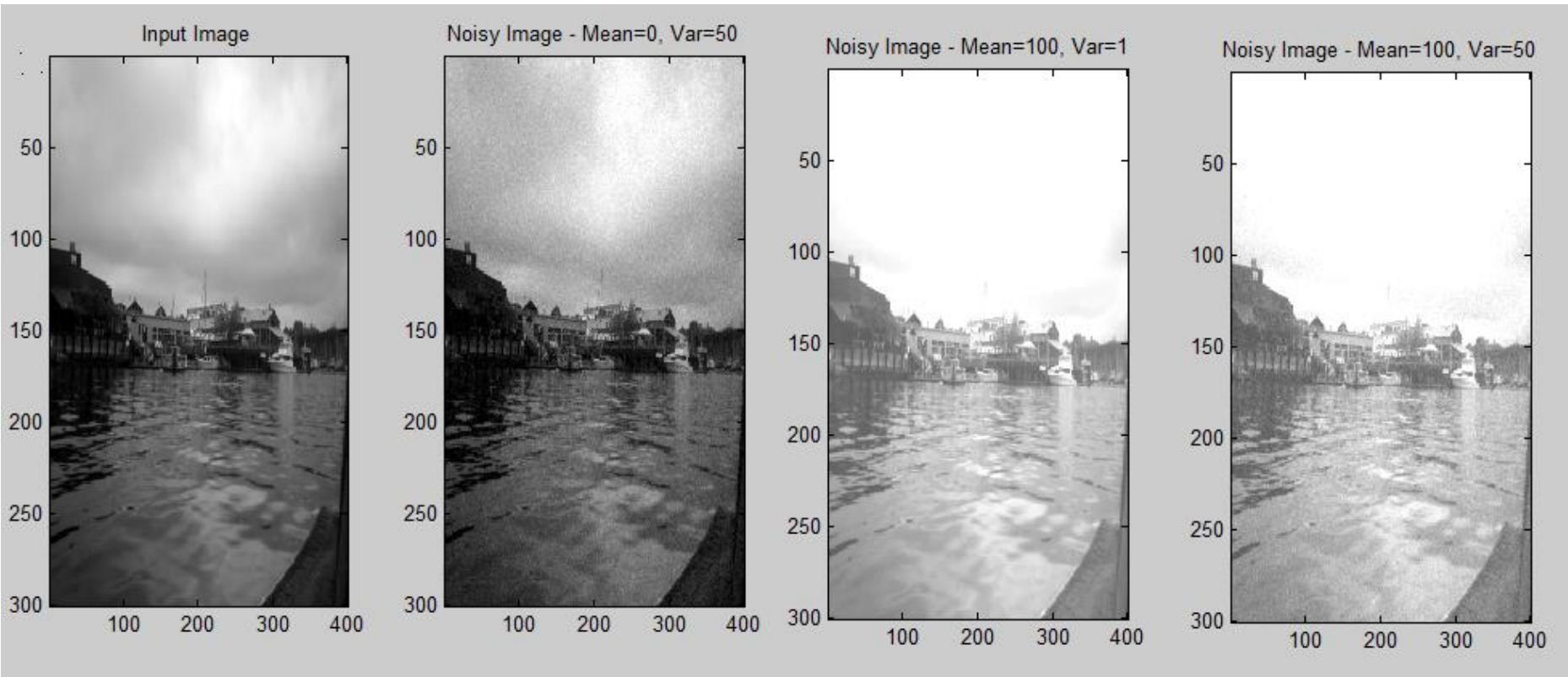
Result of  
Filtering Above  
With  $3 \times 3$   
Contraharmonic  
 $Q=-1.5$

# Contraharmonic Filter: Caution!

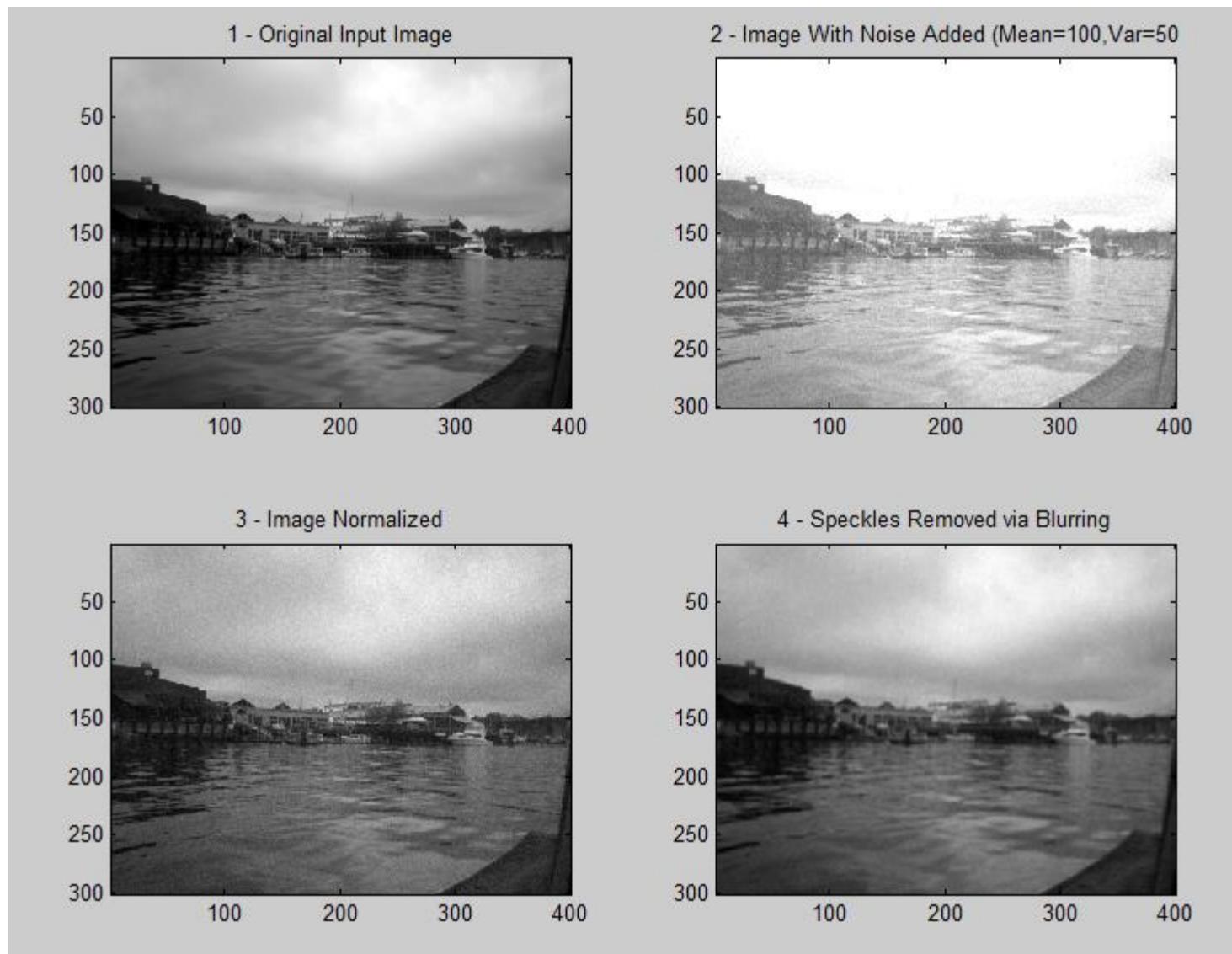
Choosing the wrong value for Q when using the contraharmonic filter can have drastic results



# Filtering of Gaussian noise



# Filtering of Gaussian noise (contd.)



# Rank Filters (non linear filtering)

Spatial filters that are based on ordering the pixel values that make up the neighbourhood operated on by the filter

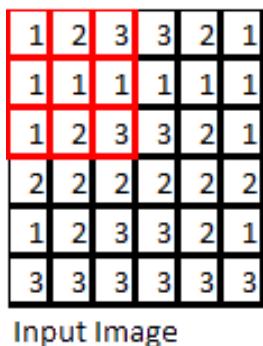
Useful spatial filters include

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

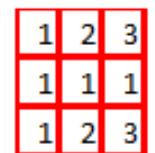
# Median Filter:

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

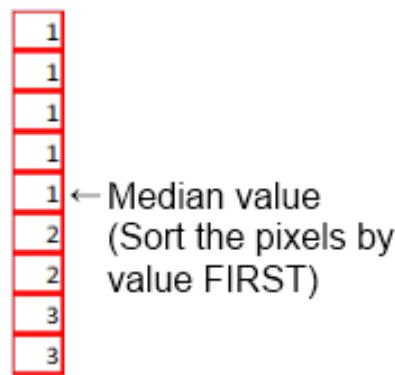
- Excellent at noise removal, without the smoothing effects that can occur with other smoothing filters
  - Particularly good when salt and pepper noise is present



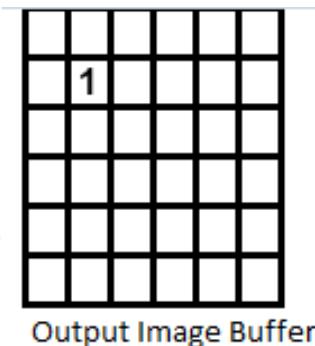
### Input Image



Values to sort



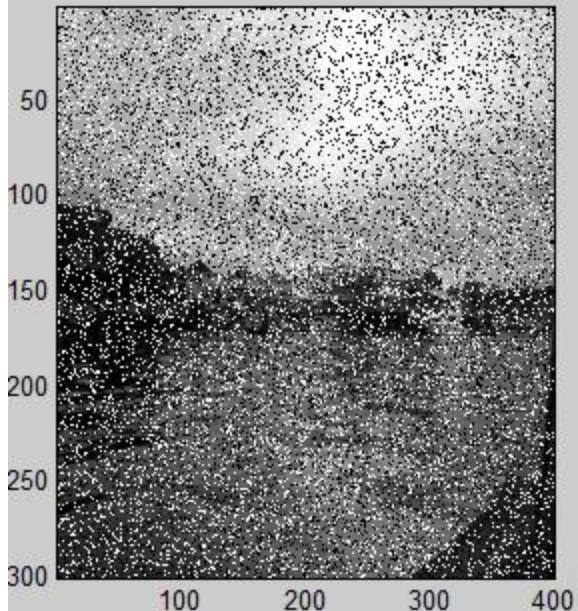
Median value  
(Sort the pixels by  
value FIRST)



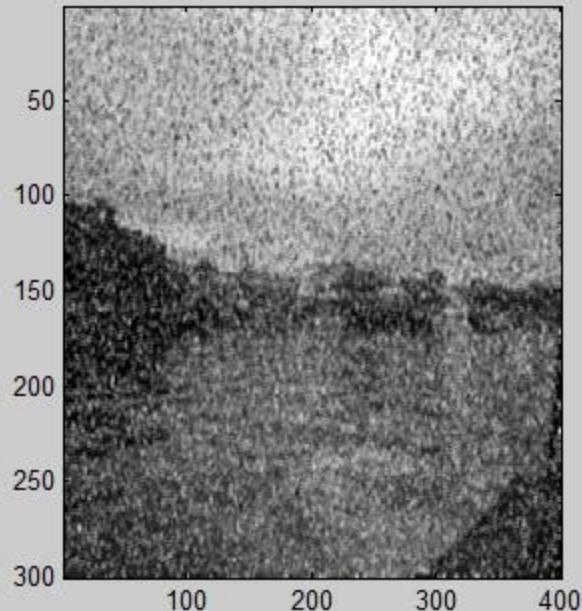
## Output Image Buffer

# Mean Vs. Median filter

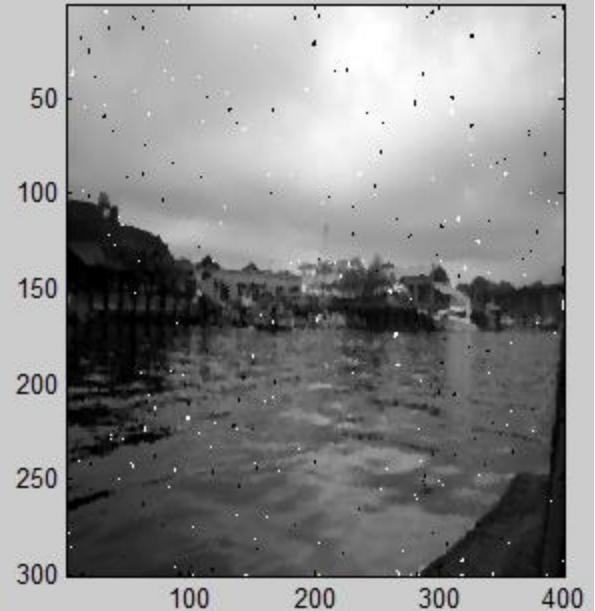
Input Image with 25% Salt & Pepper Noise



Noise Removed via 3x3 Smoothing Filter

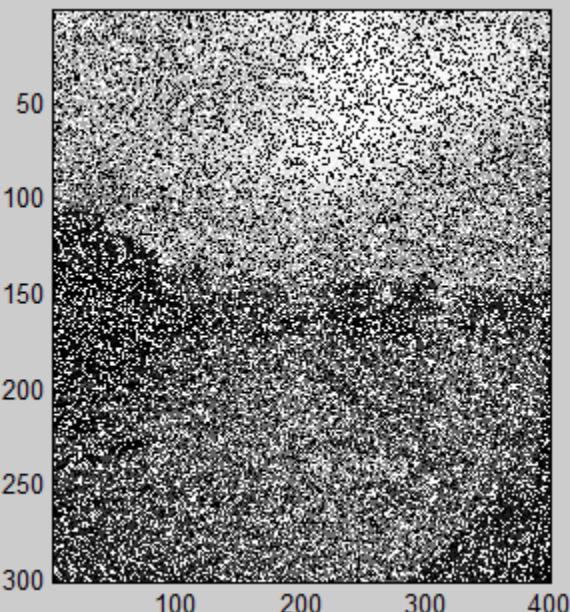


Noise Removed via 3x3 Median Filter

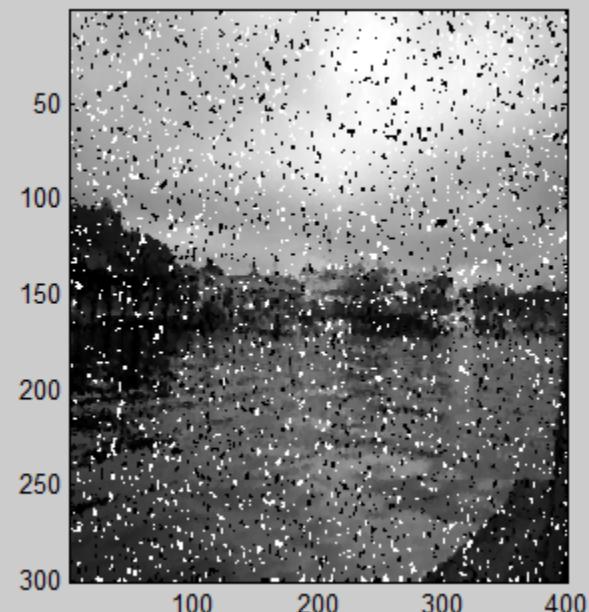


# Noise Removal Examples

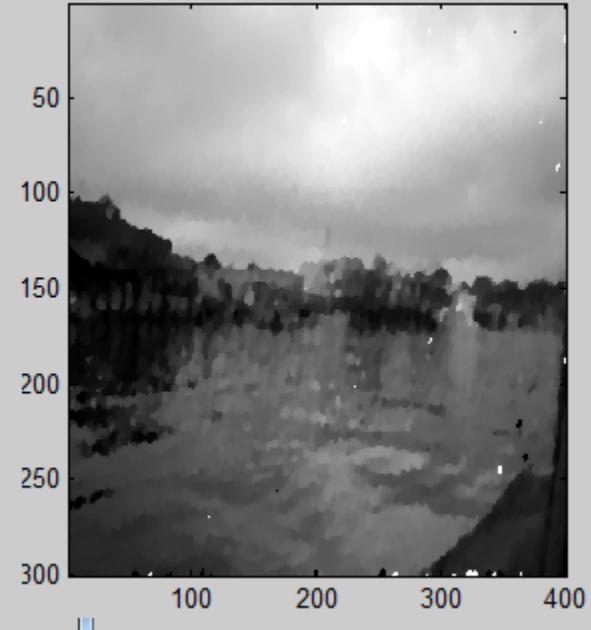
Input Image with 50% Salt & Pepper Noise



Noise Removed via 3x3 Median Filter

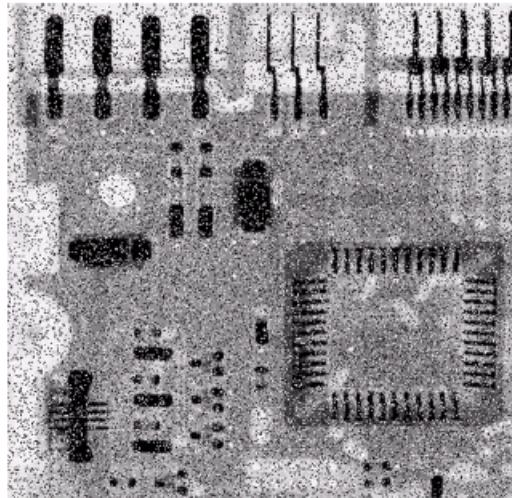


Noise Removed via 5x5 Median Filter

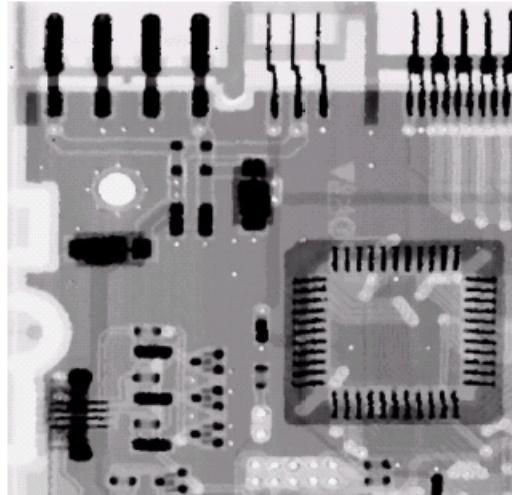


# Noise Removal Examples (contd)

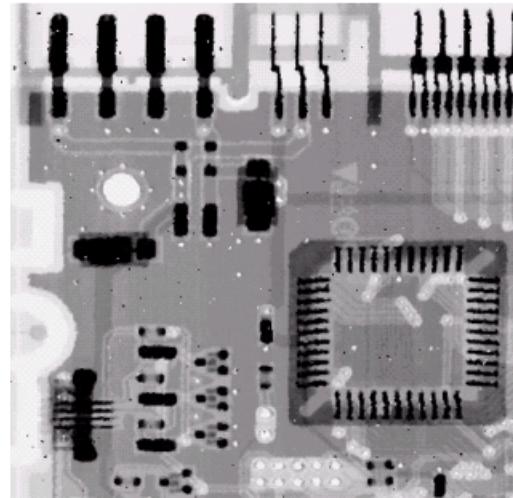
Image  
Corrupted  
By Salt And  
Pepper Noise



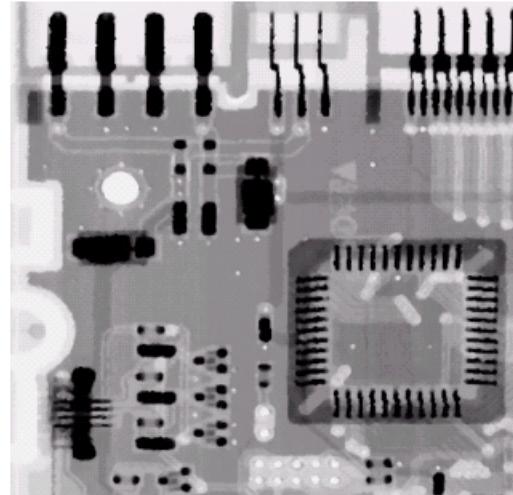
Result of 2  
Passes With  
A  $3 \times 3$  Median  
Filter



Result of 1  
Pass With A  
 $3 \times 3$  Median  
Filter



Result of 3  
Passes With  
A  $3 \times 3$  Median  
Filter



# Max and Min Filter

## Max Filter:

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

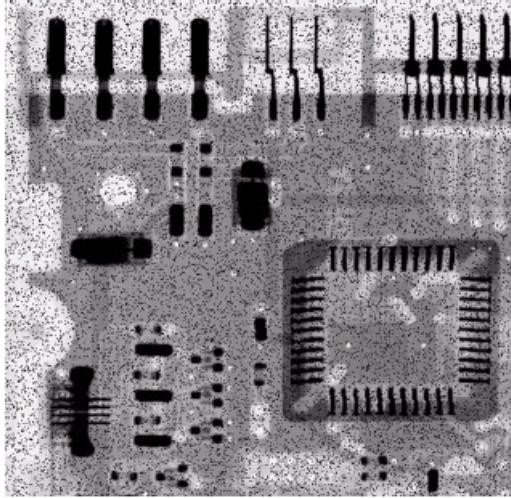
## Min Filter:

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

Max filter is good for pepper noise and min is good for salt noise

# Noise Removal Examples (cont...)

Image  
Corrupted  
By Pepper  
Noise



Result Of  
Filtering  
Above  
With A  $3 \times 3$   
Max Filter

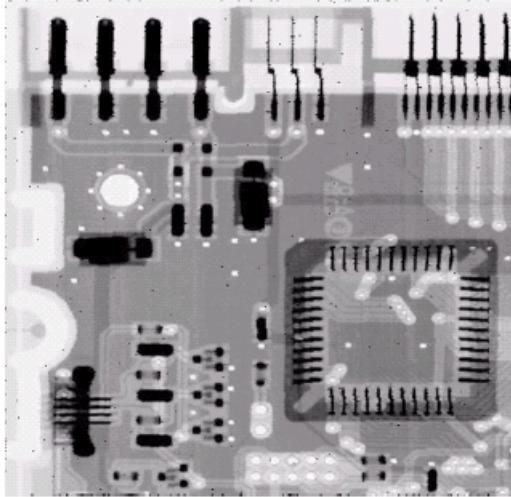
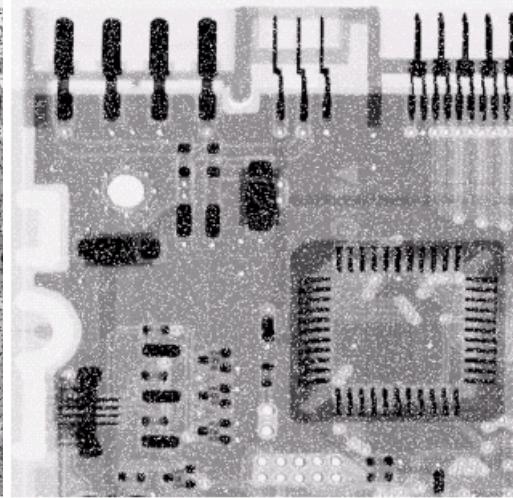
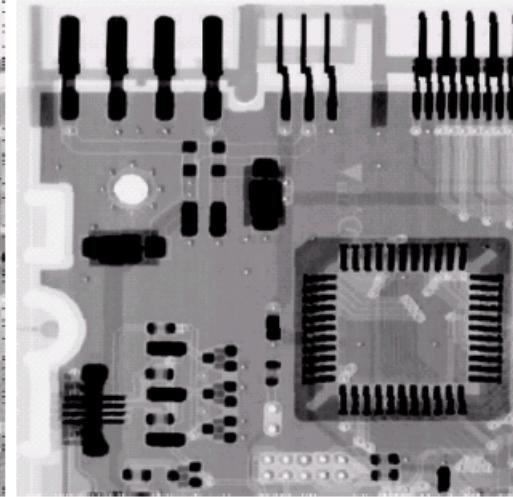


Image  
Corrupted  
By Salt  
Noise



Result Of  
Filtering  
Above  
With A  $3 \times 3$   
Min Filter



# Alpha-Trimmed Mean Filter

## Alpha-Trimmed Mean Filter:

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

We can delete the  $d/2$  lowest and  $d/2$  highest grey levels

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

# Noise Removal Examples (cont...)

Image  
Corrupted  
By Uniform  
Noise

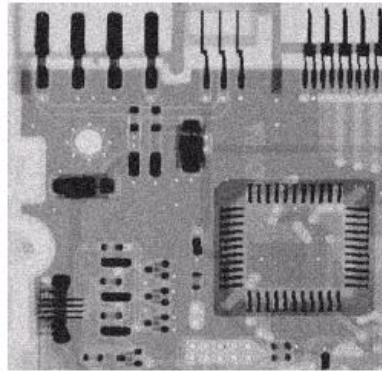
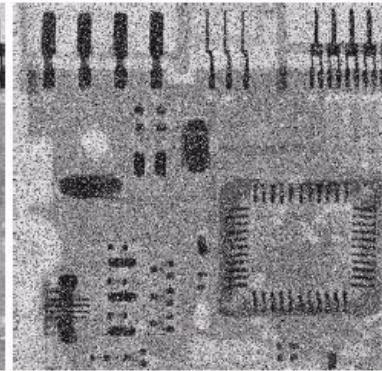
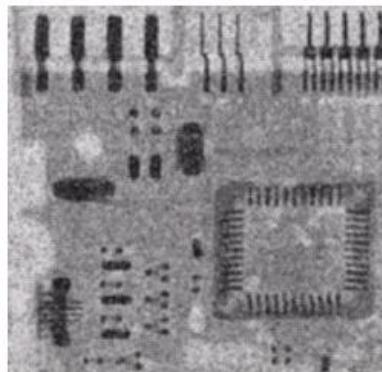


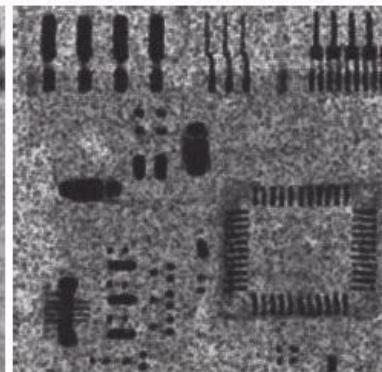
Image Further  
Corrupted  
By Salt and  
Pepper Noise



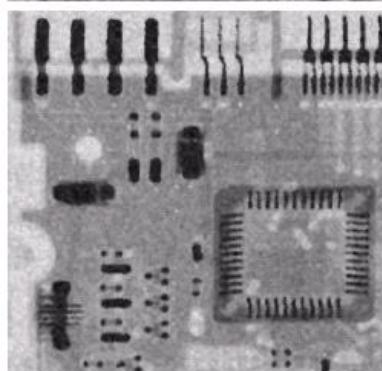
Filtered By  
5\*5 Arithmetic  
Mean Filter



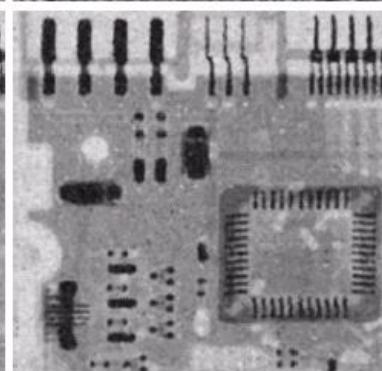
Filtered By  
5\*5 Geometric  
Mean Filter



Filtered By  
5\*5 Median  
Filter



Filtered By  
5\*5 Alpha-Trimmed  
Mean Filter



# Adaptive Filters

The filters discussed so far are applied to an entire image without any regard for how image characteristics vary from one point to another

The behaviour of **adaptive filters** changes depending on the characteristics of the image inside the filter region

# Adaptive filter: Neighborhood-based

## Adaptive local noise reduction filter

- Response based on 4 quantities
  - Local variance, **variance of noise**,  $g(x,y)$ , and local mean
- Behavior of filter
  - If variance of noise is zero, return  $g(x,y)$
  - If the local variance is high compared to the variance of noise, return a value close to  $g(x,y)$
  - If the two variances are equal, return the mean value

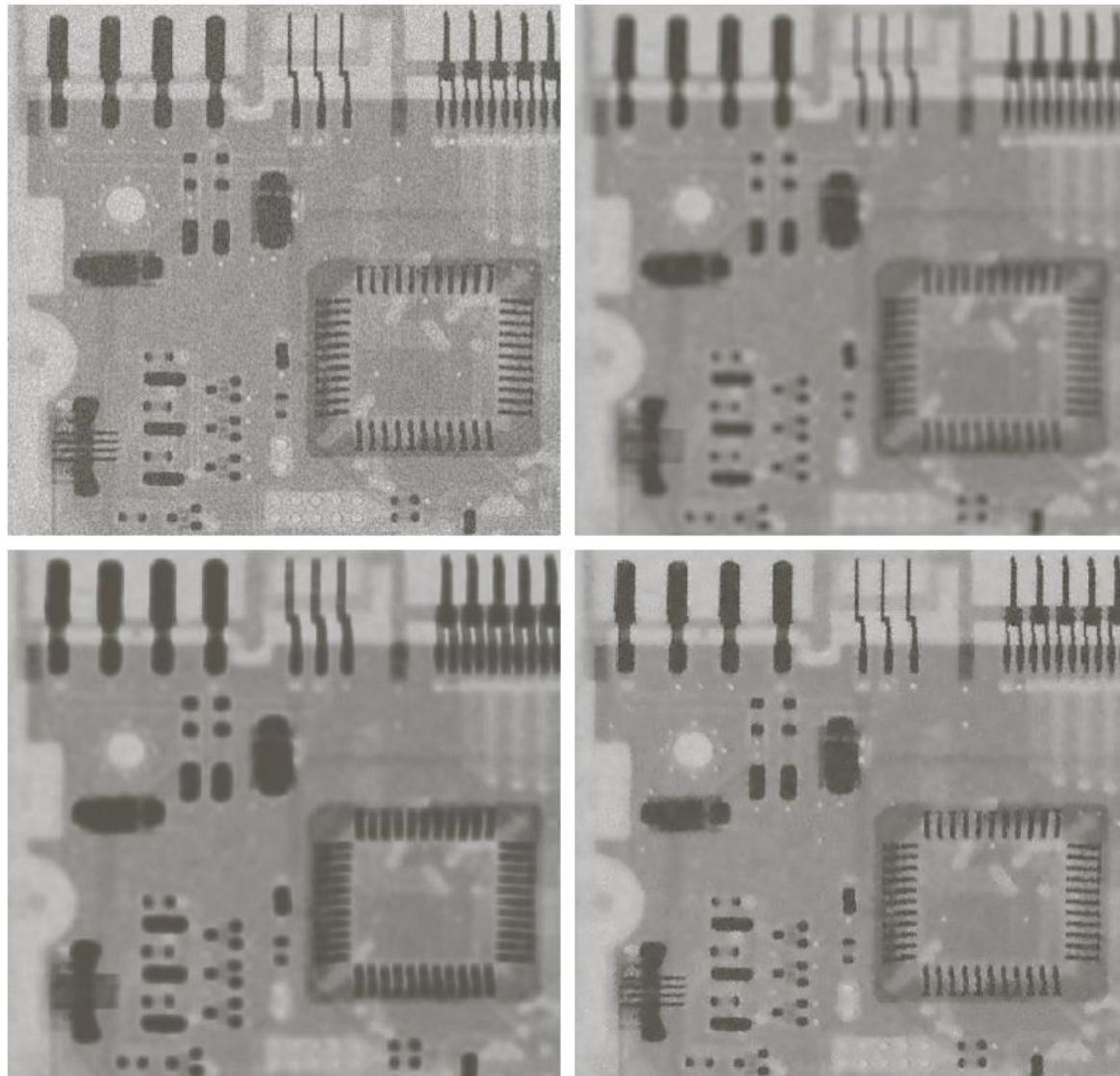
$$\hat{f}(x,y) = g(x,y) - \frac{S_h^2}{S_L^2} [g(x,y) - m_L]$$

# Adaptive filter: Neighborhood-based

a  
b  
c  
d

**FIGURE 5.13**

- (a) Image corrupted by additive Gaussian noise of zero mean and variance 1000.  
(b) Result of arithmetic mean filtering.  
(c) Result of geometric mean filtering.  
(d) Result of adaptive noise reduction filtering. All filters were of size  $7 \times 7$ .



# Adaptive Median Filtering

The median filter performs relatively well on impulse noise as long as the spatial density of the impulse noise is not large

The adaptive median filter can handle much more spatially dense impulse noise, and does less distortion

The key insight in the adaptive median filter is that the filter size changes depending on the characteristics of the image

# Adaptive Median Filtering (cont...)

The adaptive median filter has following purposes:

- Remove spatially dense impulse noise
- Reduce distortion

First examine the following notation:

- $z_{min}$  = minimum grey level in  $S_{xy}$
- $z_{max}$  = maximum grey level in  $S_{xy}$
- $z_{med}$  = median of grey levels in  $S_{xy}$
- $z_{xy}$  = grey level at coordinates  $(x, y)$
- $S_{max}$  = maximum allowed size of  $S_{xy}$

# Adaptive Median Filtering (cont...)

Level A:  $A1 = z_{med} - z_{min}$

$A2 = z_{med} - z_{max}$

If  $A1 > 0$  and  $A2 < 0$ , Go to level B

Else increase the window size

If window size  $\leq S_{max}$  repeat level A

Else output  $z_{med}$

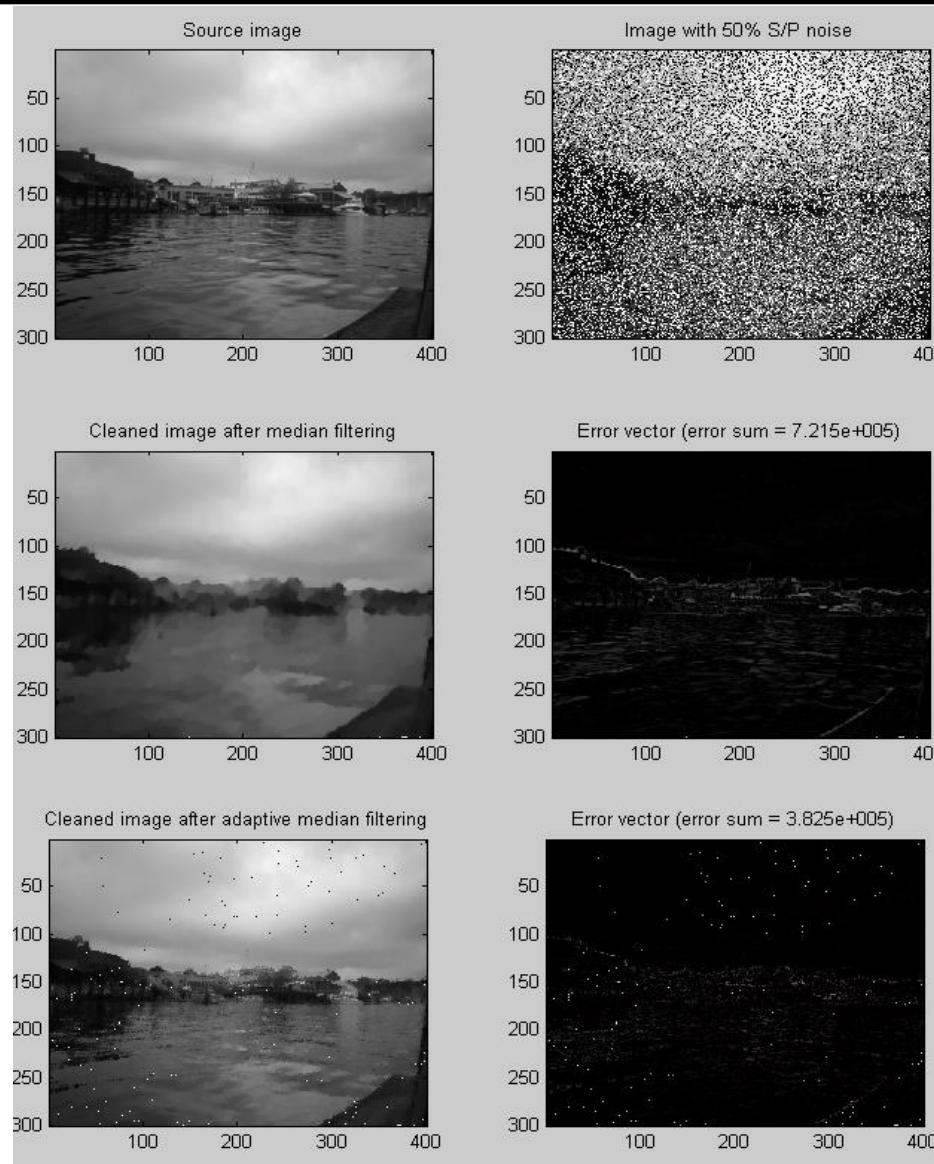
Level B:  $B1 = z_{xy} - z_{min}$

$B2 = z_{xy} - z_{max}$

If  $B1 > 0$  and  $B2 < 0$ , output  $z_{xy}$

Else output  $z_{med}$

# Simple Adaptive Median Filtering Example



Using only  
level B and  
filter 7 x 7

# Adaptive Median Filtering Examples

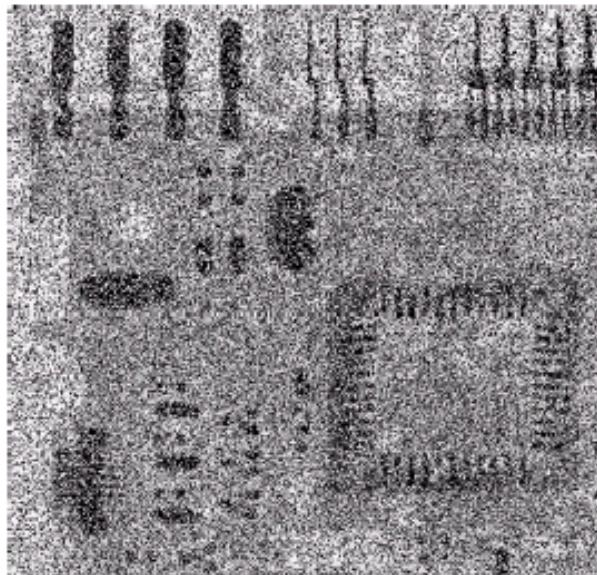
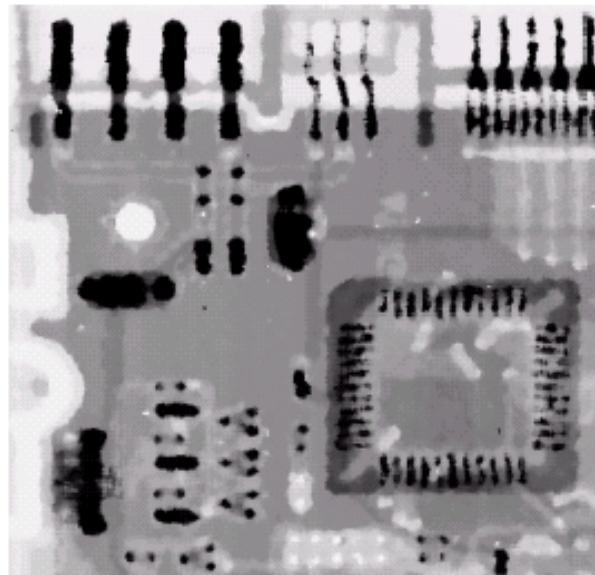
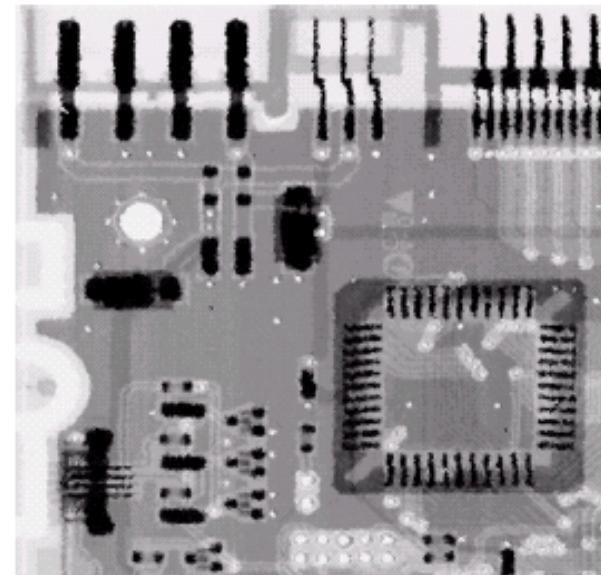


Image corrupted by salt  
and pepper noise with  
probabilities  $P_a = P_b = 0.25$



Result of filtering with a 7  
\* 7 median filter



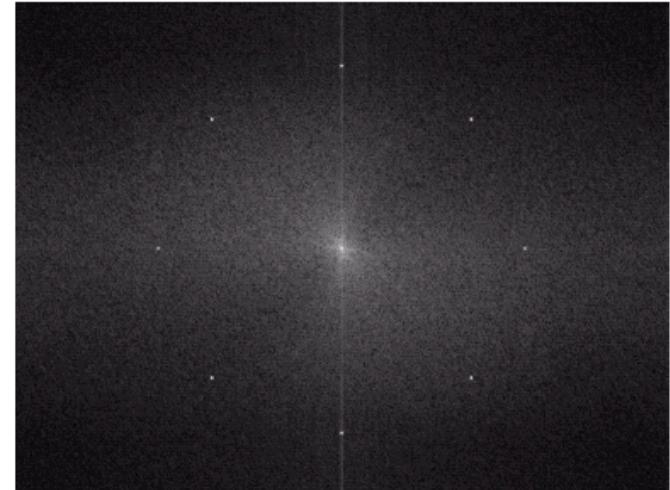
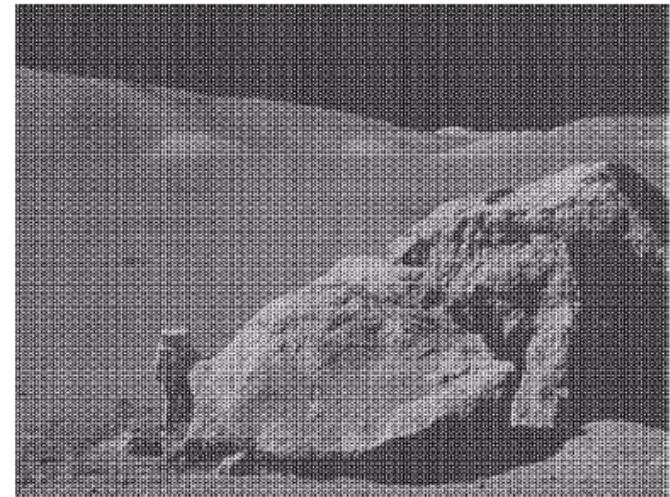
Result of adaptive median  
filtering with  $S_{max} = 7$

# Periodic Noise removal

Typically arises due to electrical or electromagnetic interference

Gives rise to regular noise patterns in an image

Frequency domain techniques in the Fourier domain are most effective at removing periodic noise



# Band Reject Filters

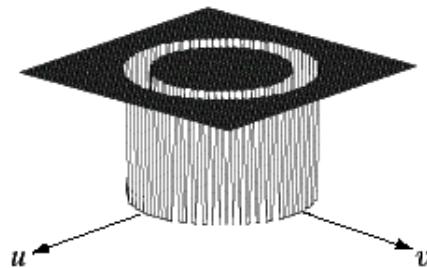
Removing periodic noise from an image involves removing a particular range of frequencies from that image

*Band reject* filters can be used for this purpose  
An ideal band reject filter is given as follows:

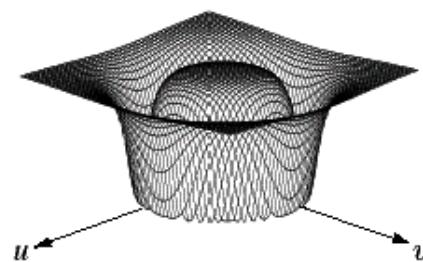
$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u, v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u, v) > D_0 + \frac{W}{2} \end{cases}$$

# Band Reject Filters (cont...)

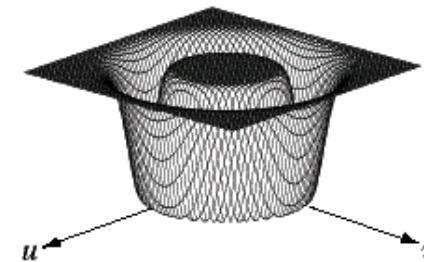
The ideal band reject filter is shown below, along with Butterworth and Gaussian versions



Ideal Band  
Reject Filter



Butterworth  
Band Reject  
Filter (of order 1)

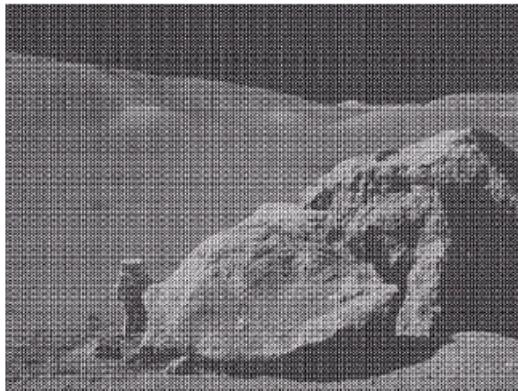


Gaussian  
Band Reject  
Filter

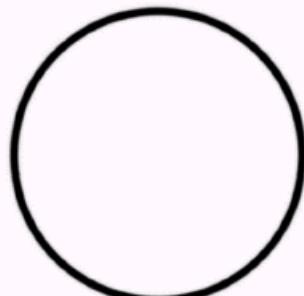
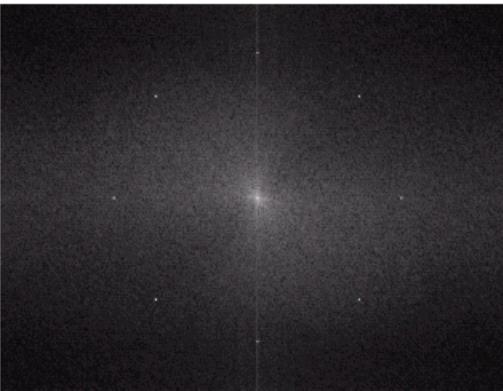
Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u, v) = \frac{1}{1 + \left[ \frac{DW}{D^2 - D_0^2} \right]^{2n}}$	$H(u, v) = 1 - e^{-\left[ \frac{D^2 - D_0^2}{DW} \right]^2}$

# Band Reject Filter Example

Image corrupted by sinusoidal noise



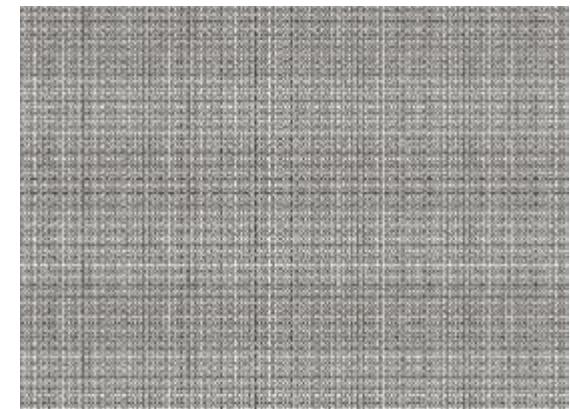
Fourier spectrum of corrupted image



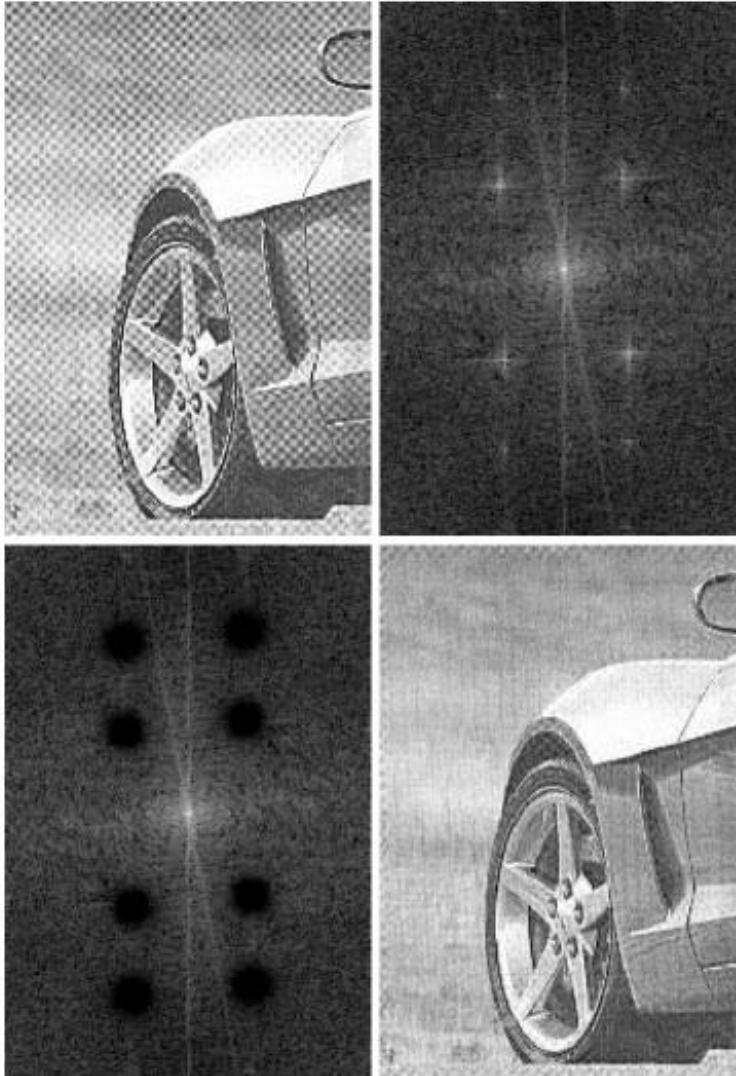
Butterworth band reject filter



Filtered image



# Notch Filter



a b  
c d

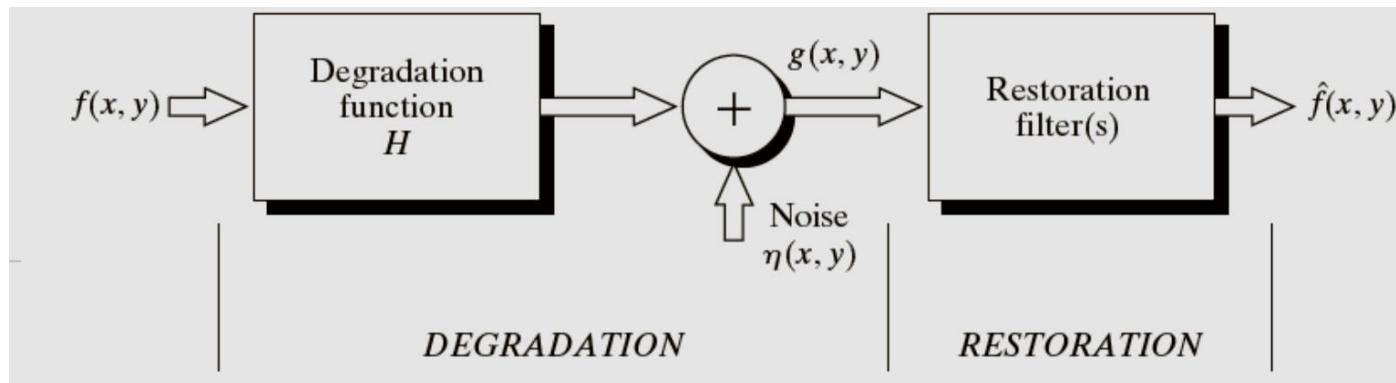
**FIGURE 4.64**

- (a) Sampled newspaper image showing a moiré pattern.
- (b) Spectrum.
- (c) Butterworth notch reject filter multiplied by the Fourier transform.
- (d) Filtered image.

# Restoration From Degradation

# Linear, position-invariant degradation model

- Many types of degradation can be *approximated* by linear, position-invariant processes.  
$$g(x,y) = f(x,y) * h(x,y) + \eta(x,y)$$
  
$$G(u,v) = F(u,v)H(u,v) + N(u,v)$$



- Image deconvolution: find  $H(u,v)$  and apply inverse process

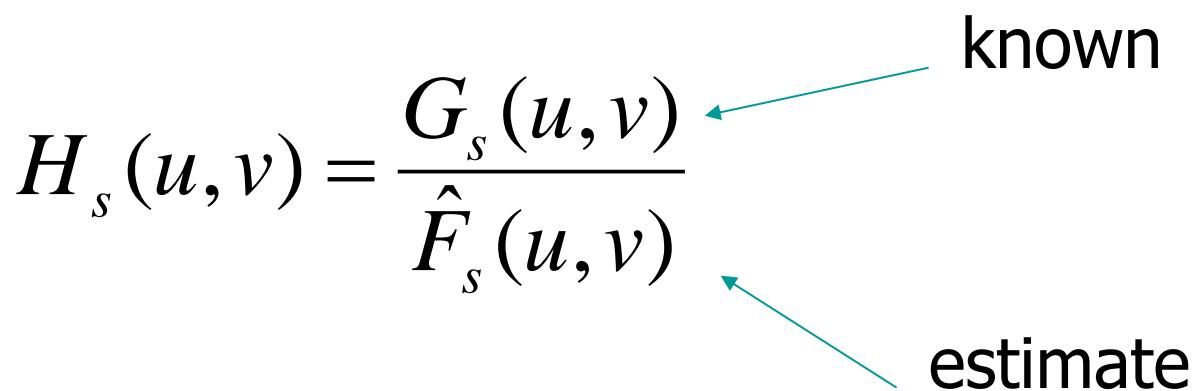
# Estimating the Degradation Function

- By image observation: Take a region in the image with
  - Simple structure
  - Strong signal content (negligible noise)
- Estimate the original image in the window

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

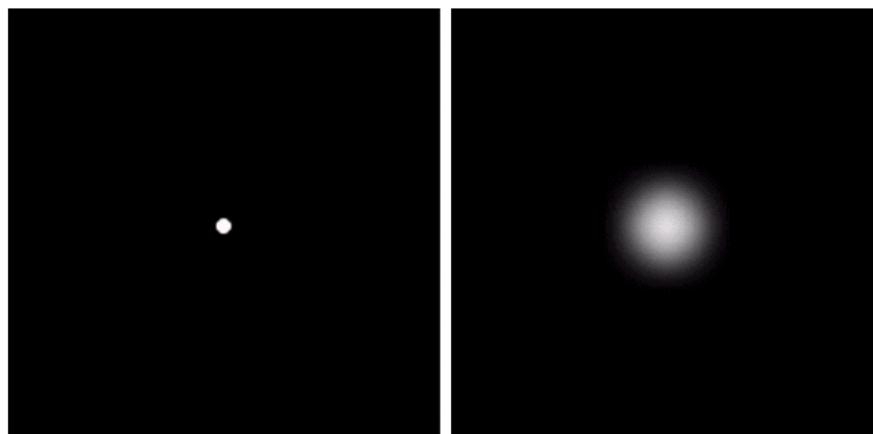
known

estimate



# Estimating the Degradation Function

- By experimentation: If the image acquisition system is available, then obtain **impulse response**
  - $G(u,v) = H(u,v)F(u,v) + N(u,v)$
  - $H(u,v) = G(u,v) / A$



a b

**FIGURE 5.24**  
Degradation estimation by impulse characterization.  
(a) An impulse of light (shown magnified).  
(b) Imaged (degraded) impulse.

# Estimating the Degradation Function

- By modeling: Ex. Atmospheric turbulence

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

original



$k=0.001$



$k=0.0025$



$k=0.00025$

# Estimating the Degradation Function

- Derive a **mathematical model**: ex. Motion blur

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

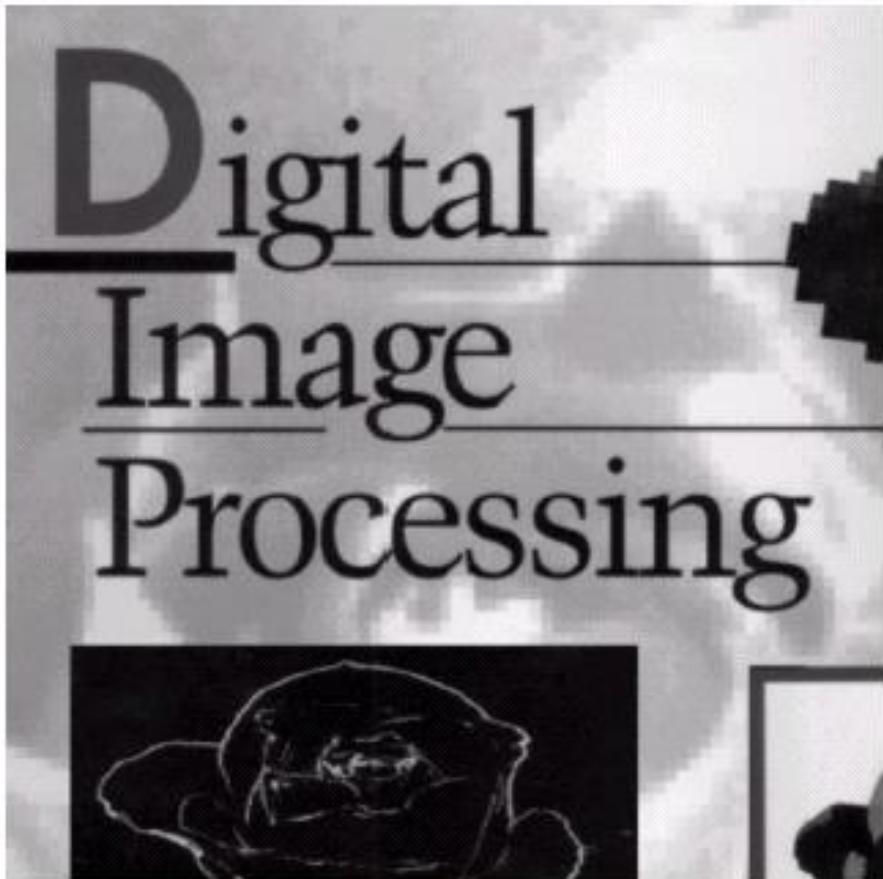
Fourier  
transform

Planar motion

$$G(u, v) = F(u, v) \int_0^T e^{-j2\pi[ux_0(t)+vy_0(t)]} dt$$

# Estimating the Degradation Function

original



Apply motion model



# Inverse Filtering

- Compute an estimate by simply dividing the transform of degraded image by the degradation function

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

Estimate of  
original image

Unknown  
noise

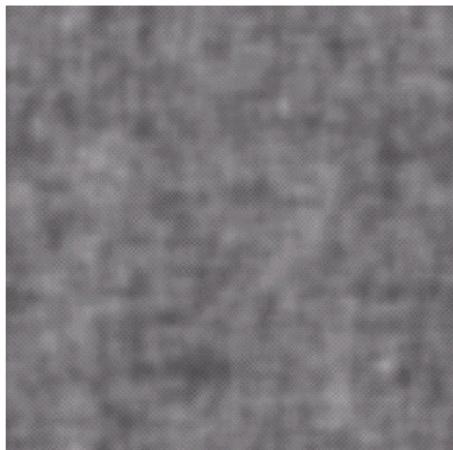
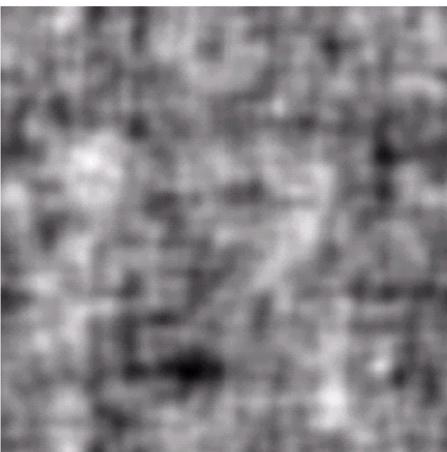
Problem: 0 or small values

- Suffers from noise amplification

# Psuedo inverse filtering

- Limiting the analysis to frequencies near origin

Full  
inverse  
filter  
for  
 $k=0.0025$



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

$$\hat{H}(u, v) = \begin{cases} 1/H(u, v), & |u^2 + v^2| \leq \eta \\ 0, & |u^2 + v^2| > \eta \end{cases}$$

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

Cut  
Outside  
40%

Degraded  
Image



Cut  
Outside  
85%

# Wiener Filtering

- Incorporates both the degradation and statistical characteristics of noise
- Objective function: find an estimate  $\hat{f}$  of  $f$  such that the mean square error between them is minimized

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

$$\hat{F}(u, v) = \frac{H^*(u, v) S_f(u, v)}{S_f(u, v) |H(u, v)|^2 + S_n(u, v)} G(u, v)$$

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

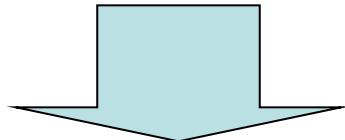
$S_n(u, v) = |N(u, v)|^2 = \text{power spectrum of the noise}$

# Wiener Filtering

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

Constant

Unknown



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

# Inverse vs. Wiener Filtering



a b c

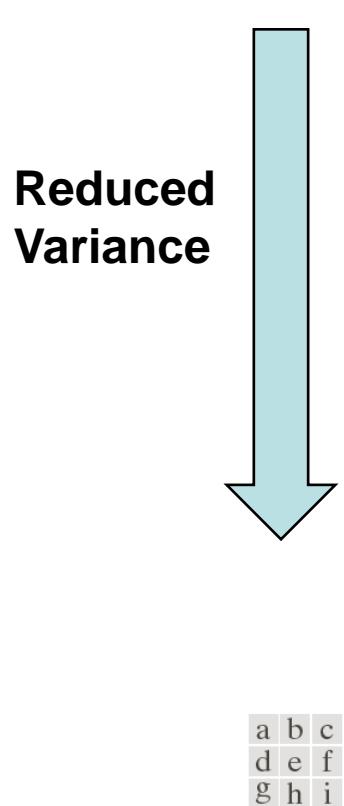
**FIGURE 5.28** Comparison of inverse- and Wiener filtering. (a) Result of full inverse filtering of Fig. 5.25(b). (b) Radially limited inverse filter result. (c) Wiener filter result.

Full inverse  
filtering

Radially limited  
inverse filtering

Wiener filtering  
(K was chosen interactively)

# Further comparison



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

# Quantitative evaluation of image restoration

- Some Performance Measures for evaluating Image restoration method where the original image and reconstructed image from its degraded version is available are as follows:
  - Signal-to-Noise Ratio (SNR)
  - Mean Square Error (MSE)
  - Root Mean Square Error (RMSE)
  - Peak Signal-to-Noise Ratio (PSNR)

- Signal to noise ratio

$$SNR = \frac{\sum_{x=1}^m \sum_{y=1}^n |F(x, y)|^2}{\sum_{x=1}^m \sum_{y=1}^n |N(x, y)|^2}$$

- ***Mean square error:***

$$MSE = \frac{1}{m \times n} \sum_{i=1}^m \sum_{j=1}^n [I'(i, j) - I(i, j)]^2$$

Where  $I$  is the original image and  $I'$  is the reconstructed image.

- ***Root mean square error:***

$$RMSE = \sqrt{MSE}$$

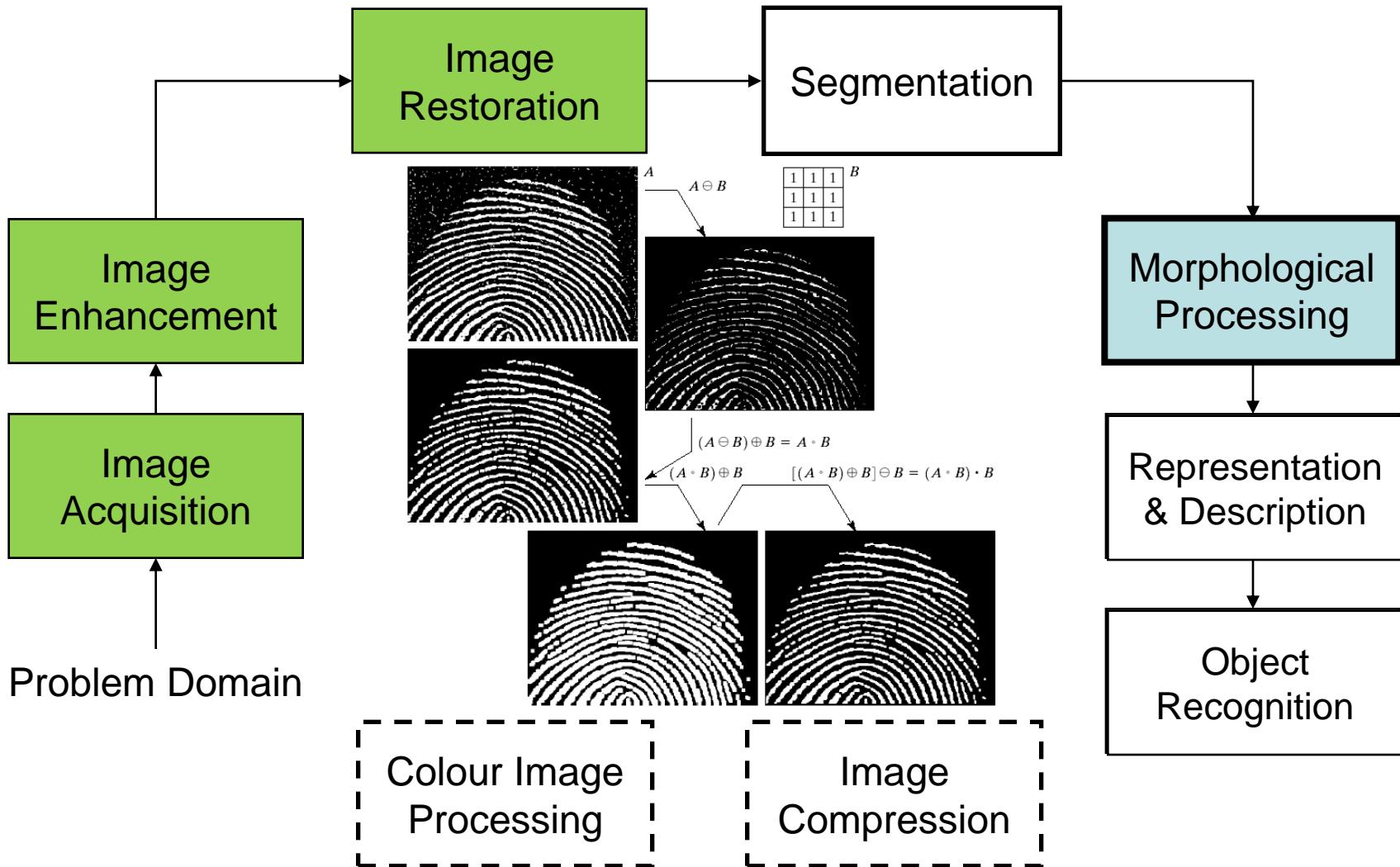
- ***Peak signal-to-noise ratio:***

$$PSNR = 20 \log_{10} \left[ \frac{255}{RMSE} \right]$$

# Image and Video Processing

## Morphological Image Processing

# Course Outline



Morphological operations can be used to remove imperfections in the segmented image and provide information on the form and structure of a region shape

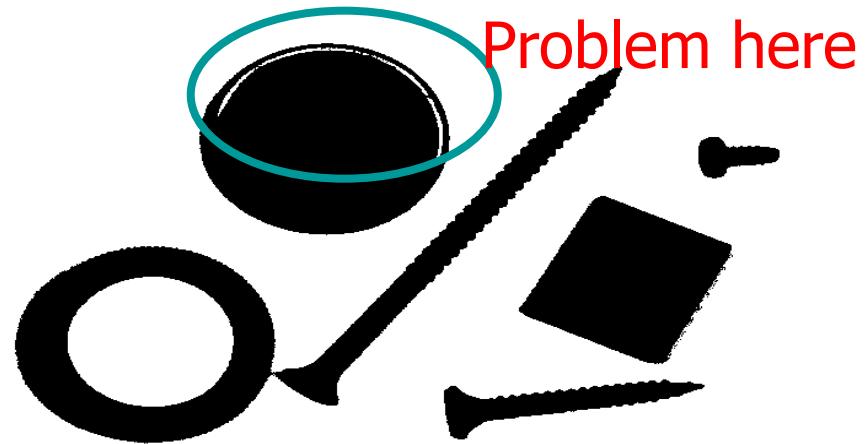
In this topic we will consider

- What is morphology?
- Simple morphological operations
- Compound operations
- Morphological algorithms

# What Is Morphology?

- Morphological image processing (or *morphology*) describes a range techniques for extracting image components that are useful in the representation & description of region shape such as boundaries, skeletons etc.
- Some of the operations are frequently applied as post-processing to remove imperfections introduced during segmentation, and so typically operate on binary images.

# Quick Example 1



How do we fill “missing pixels”?

# Quick Example 2



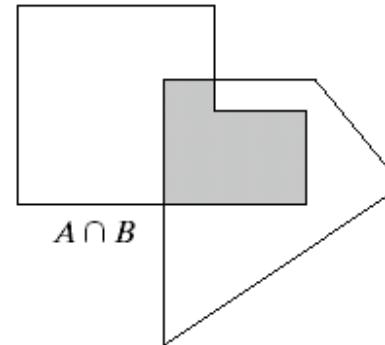
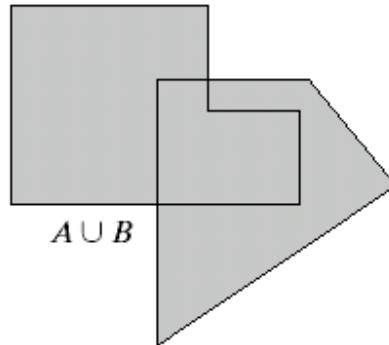
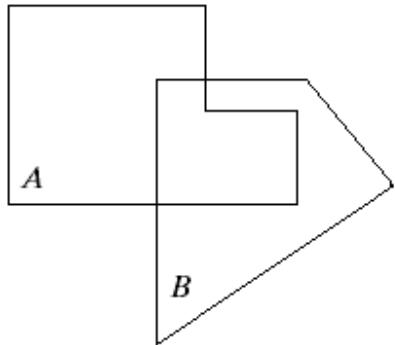
Image after segmentation



Image after segmentation and  
morphological processing

# Preliminaries

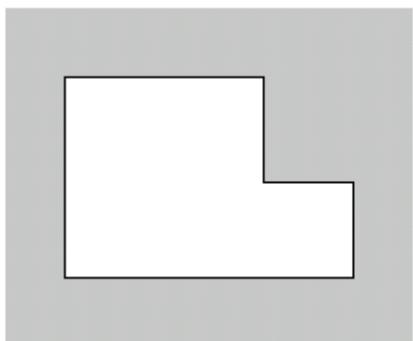
- Language of mathematical morphology is set theory



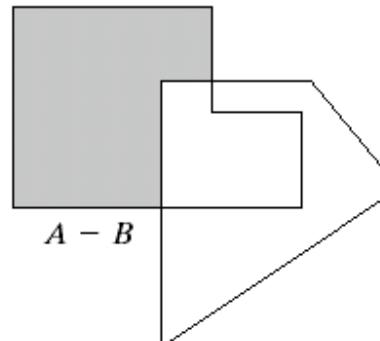
a	b	c
d	e	

**FIGURE 9.1**

- (a) Two sets  $A$  and  $B$ . (b) The union of  $A$  and  $B$ .  
(c) The intersection of  $A$  and  $B$ . (d) The complement of  $A$ .  
(e) The difference between  $A$  and  $B$ .

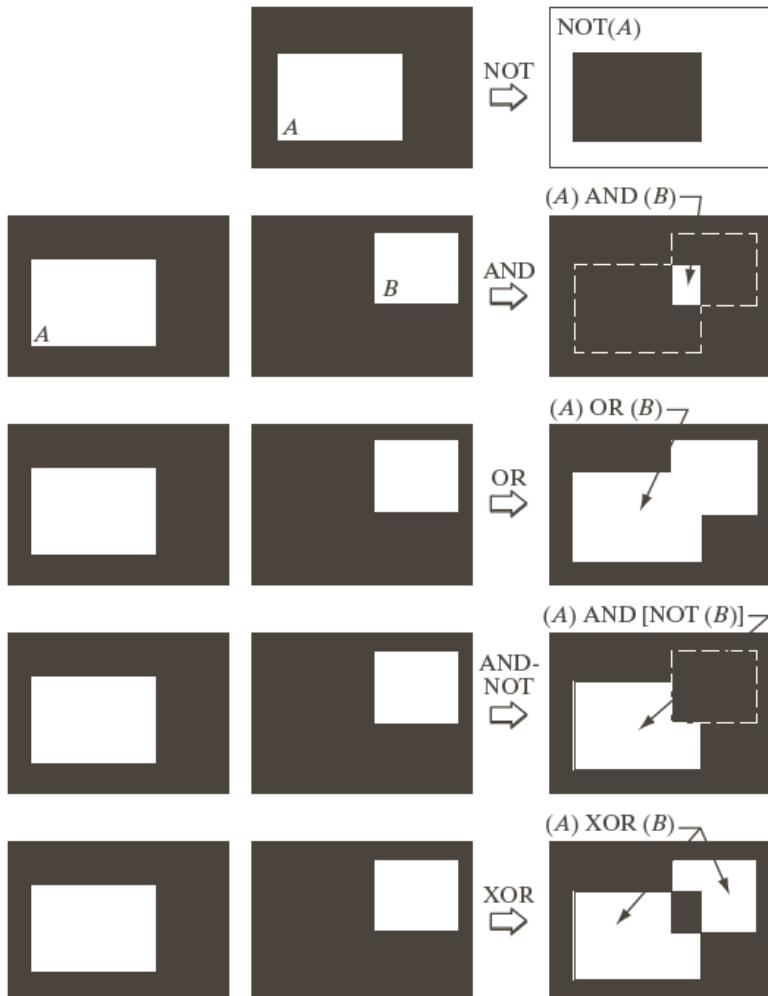


$(A)^c$



# Preliminaries

- Logic operations involving binary images



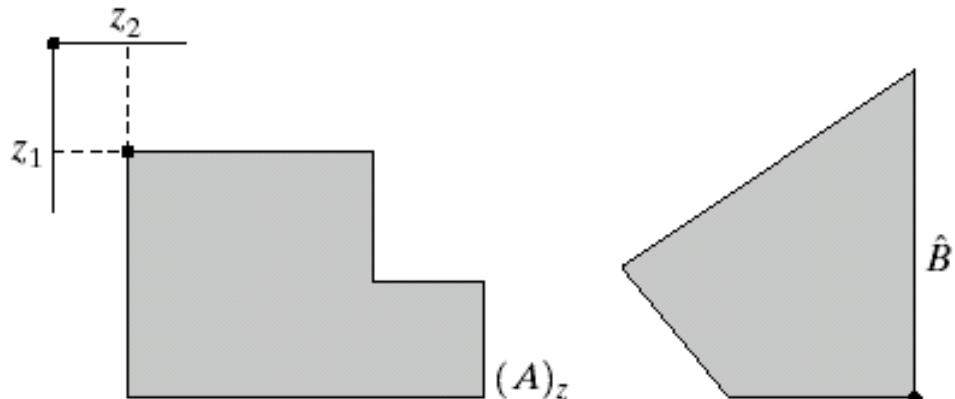
**FIGURE 2.33**  
Illustration of  
logical operations  
involving  
foreground  
(white) pixels.  
Black represents  
binary 0s and  
white binary 1s.  
The dashed lines  
are shown for  
reference only.  
They are not part  
of the result.

# Preliminaries

- Reflection and Translation:

$$\hat{B} = \{w \mid w \in -b, \text{ for } b \in B\}$$

$$(A)_z = \{c \mid c \in a + z, \text{ for } a \in A\}$$



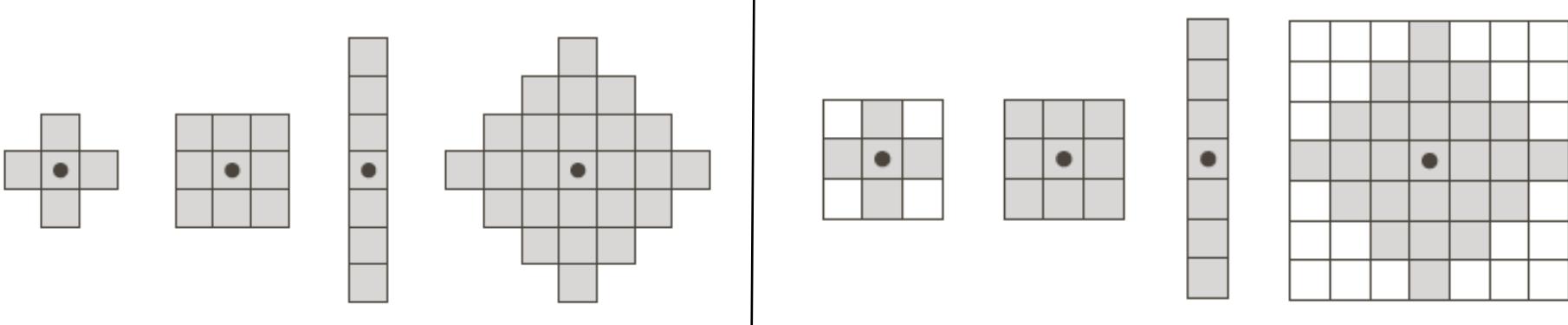
a b

**FIGURE 9.2**

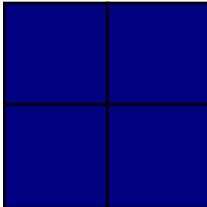
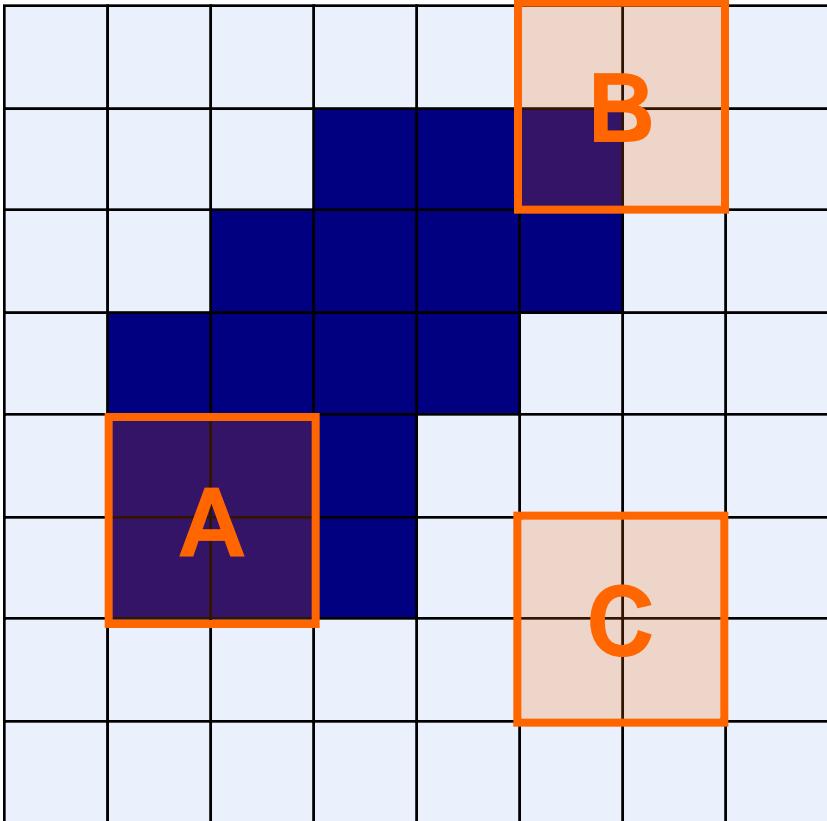
(a) Translation of  $A$  by  $z$ .  
(b) Reflection of  $B$ . The sets  $A$  and  $B$  are from Fig. 9.1.

# Structuring Elements

- Structuring elements are small sets/sub-images used to probe an image under study
- For each SE, define its origin
- shape and size must be adapted to geometric properties for the objects
  - For simplicity we will use rectangular structuring elements with their origin at the middle pixel



# Structuring Elements, Hits & Fits



Structuring Element

**Fit:** All *on pixels* in the structuring element cover *on pixels* in the image

**Hit:** Any *on pixel* in the structuring element covers an *on pixel* in the image

Basic morphological processing operations are based on these simple ideas (informally)

# Fitting & Hitting

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	<b>B</b>	1	1	1	0	<b>C</b>	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	<b>A</b>	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

Structuring  
Element 1

0	1	0
1	1	1
0	1	0

Structuring  
Element 2

# Fundamental Operations

- Fundamentally morphological image processing is very much like spatial filtering
- The structuring element is moved across every pixel in the original image to give a pixel in a new processed image
- The value of this new pixel depends on the operation performed
- There are two basic morphological operations: **erosion** and **dilation**

Erosion of image A by structuring element B represented as sets in  $\mathbb{Z}^2$ , is defined as:

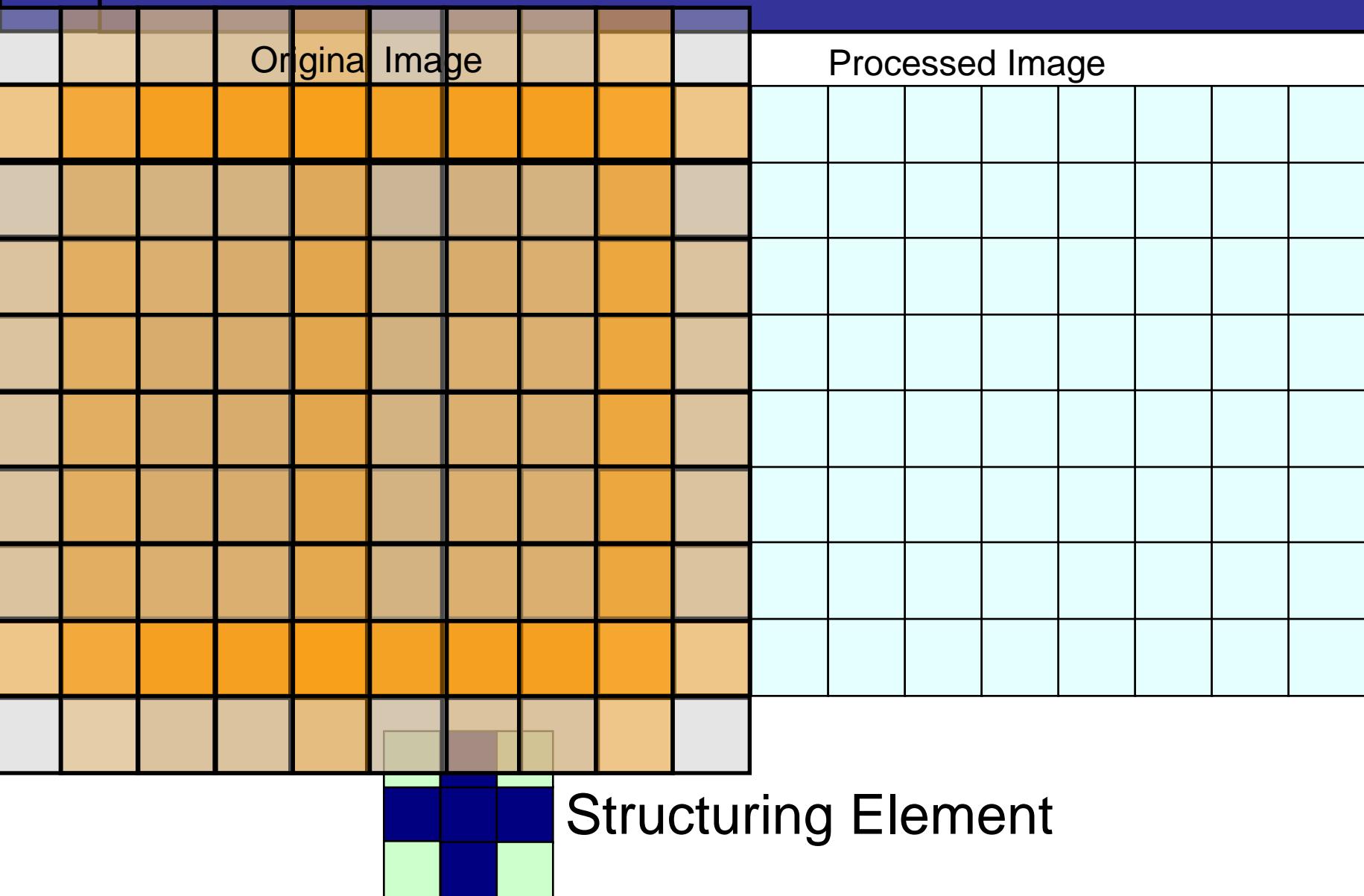
$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

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

Informally, the structuring element B is positioned with its origin at  $(x, y)$  and the new pixel value is determined using the rule:

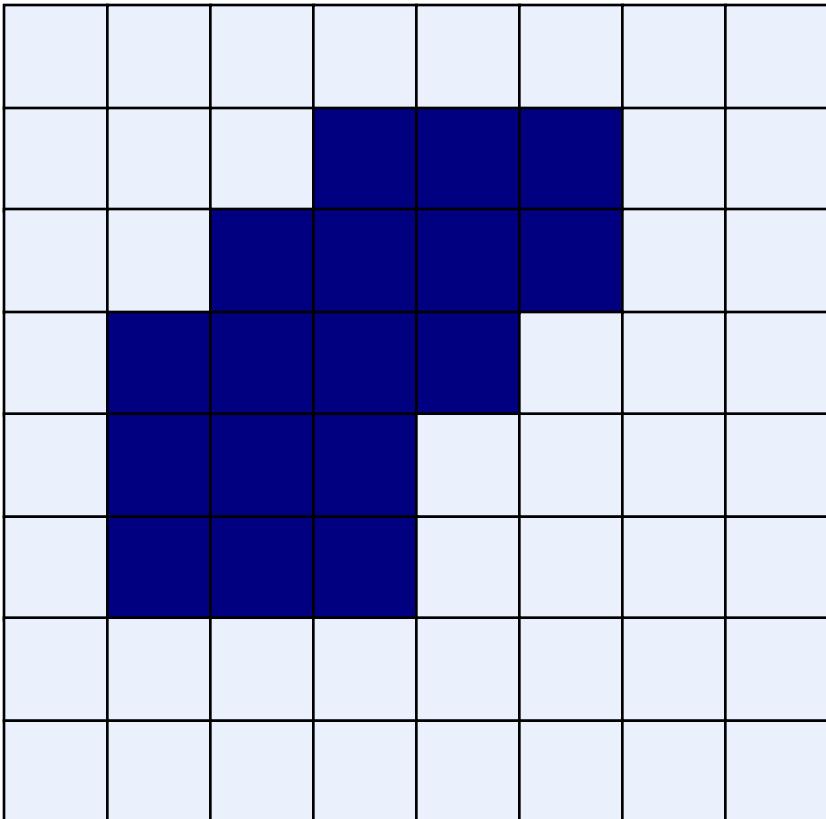
$$g(x, y) = \begin{cases} 1 & \text{if } B \text{ fits } A \\ 0 & \text{otherwise} \end{cases}$$

# Erosion Example

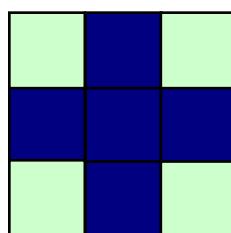
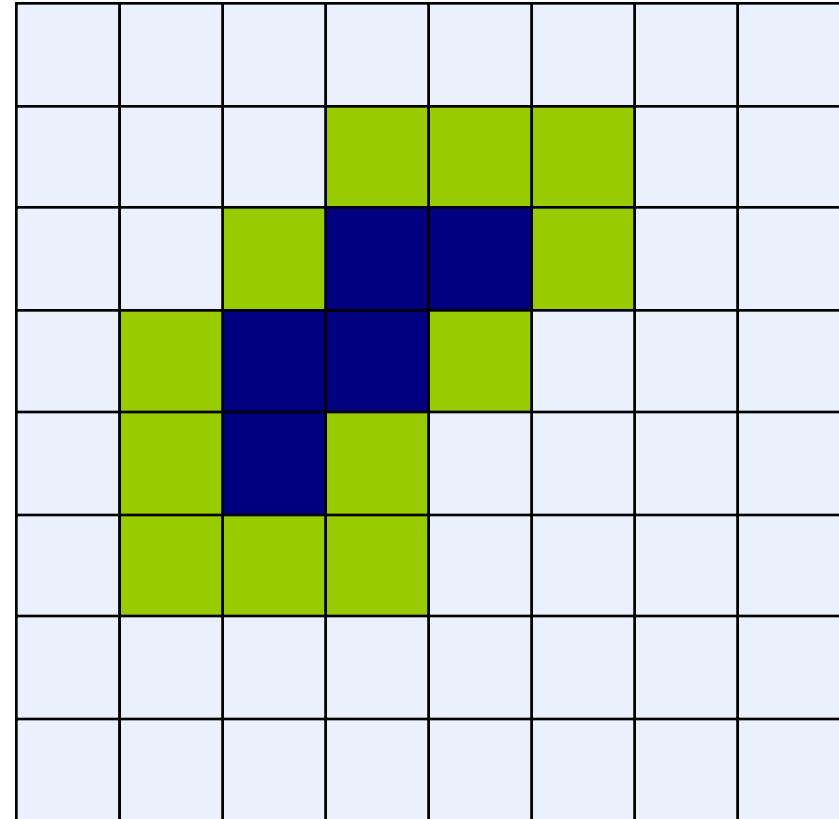


# Erosion Example

Original Image



Processed Image With Eroded Pixels



Structuring Element

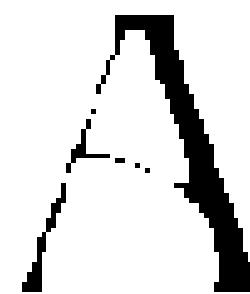
# Erosion Example 1



Original image



Erosion by  $3 \times 3$   
square structuring  
element

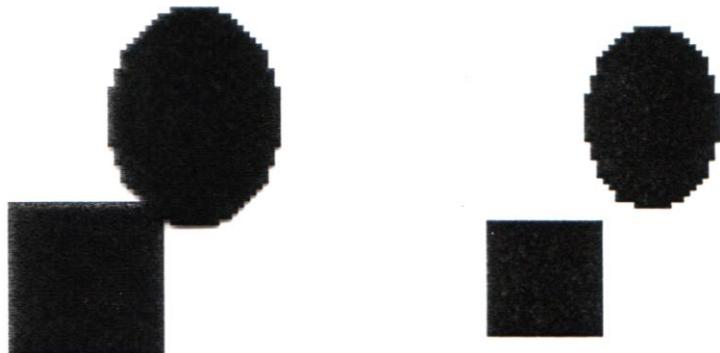


Erosion by  $5 \times 5$   
square structuring  
element

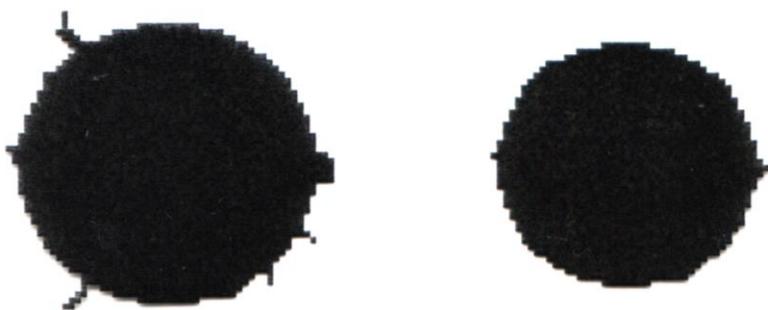
**Watch out:** In these examples a 1 refers to a black pixel!

# What Is Erosion For?

Erosion can split apart joined objects

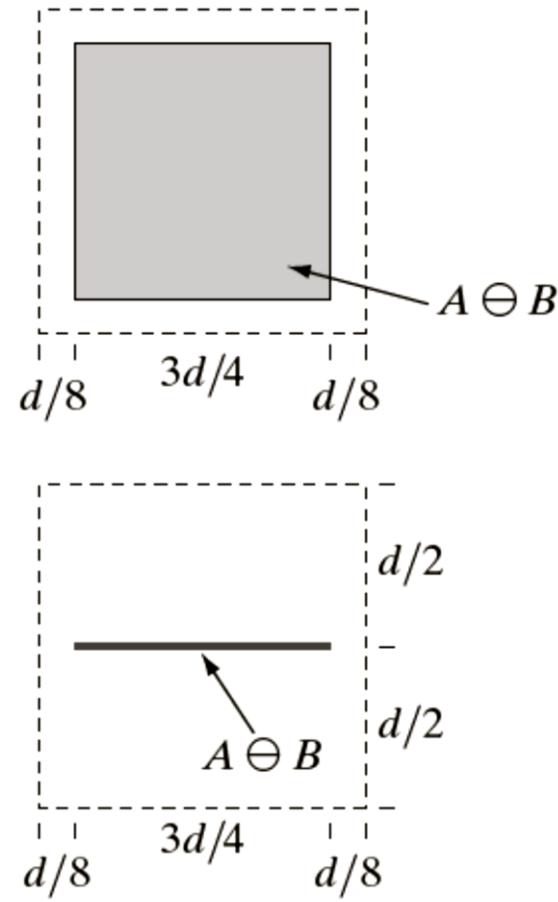
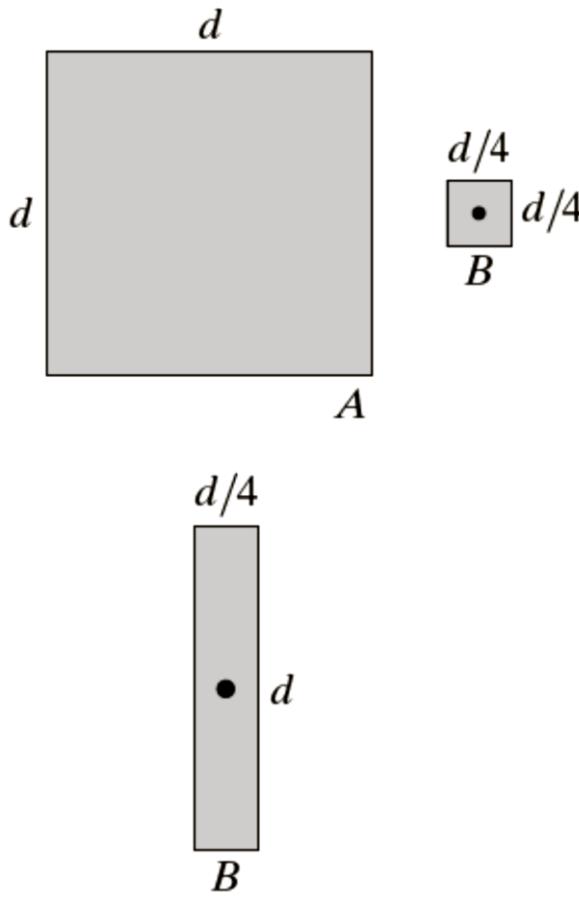


Erosion can strip away extrusions

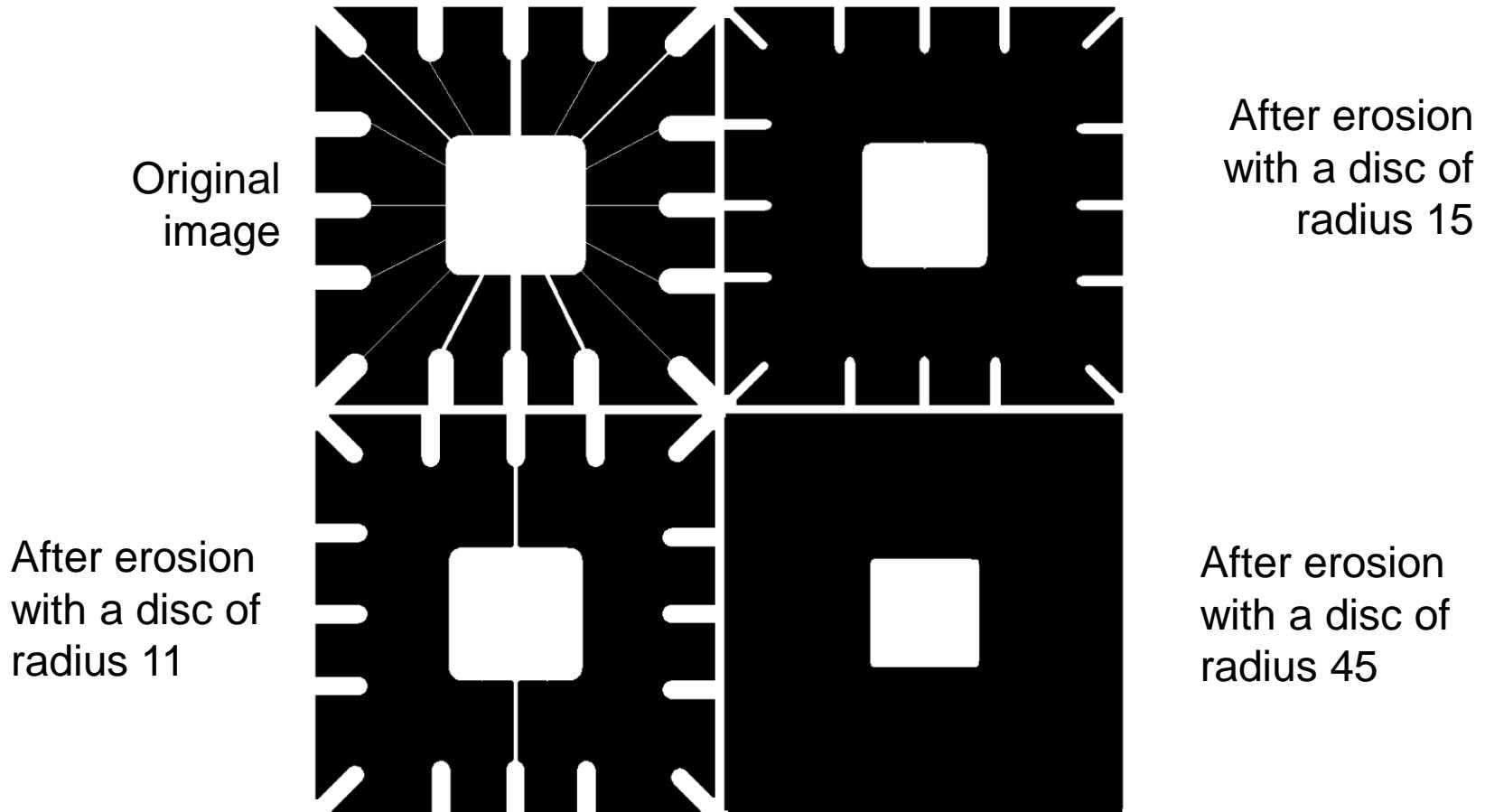


**Watch out:** Erosion shrinks objects

# Example 2



# Erosion Example 3



Dilation of image  $A$  by structuring element  $B$  is given by:

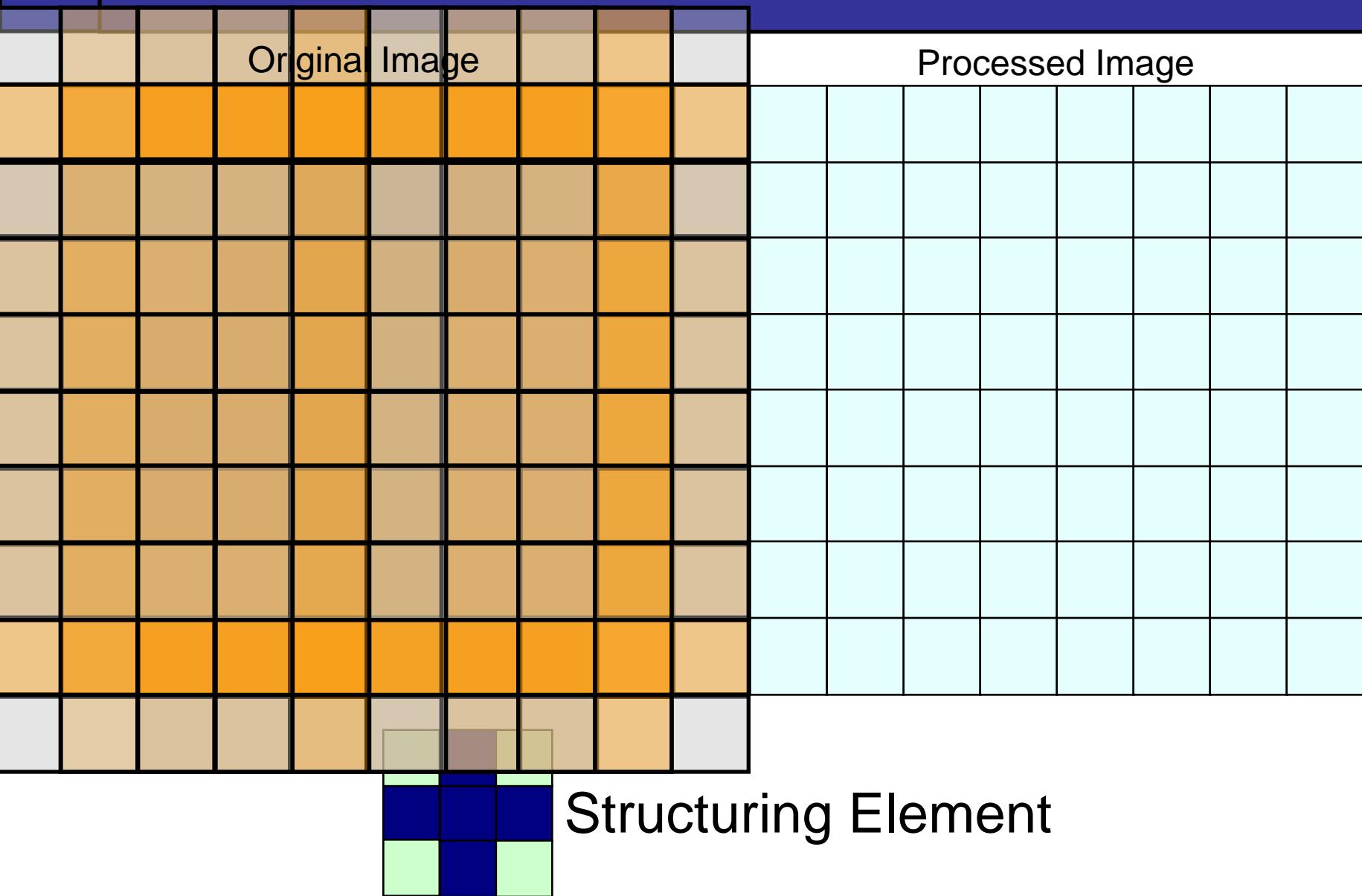
$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\}$$

The set of all displacements  $z$ , such that  $\hat{B}$  and  $A$  overlap by at least one element.

Informally: The structuring element  $B$  (if symmetric with origin at center, else reflected) is positioned with its origin at  $(x, y)$  and the new pixel value is determined using the rule:

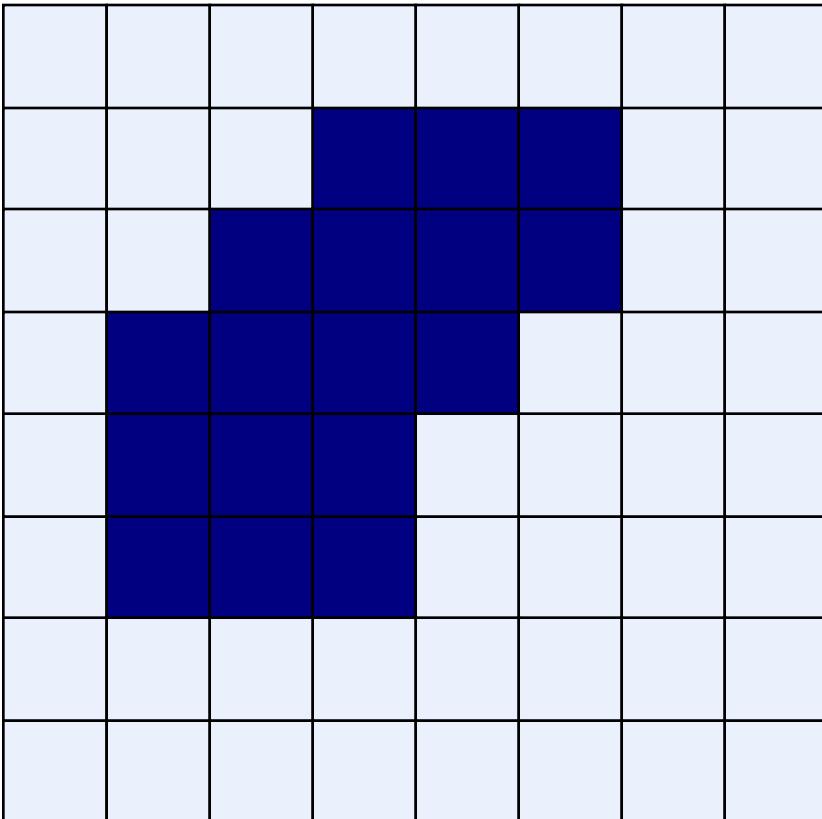
$$g(x, y) = \begin{cases} 1 & \text{if } B \text{ hits } A \\ 0 & \text{otherwise} \end{cases}$$

# Dilation Example

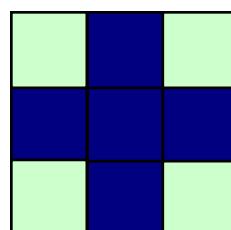
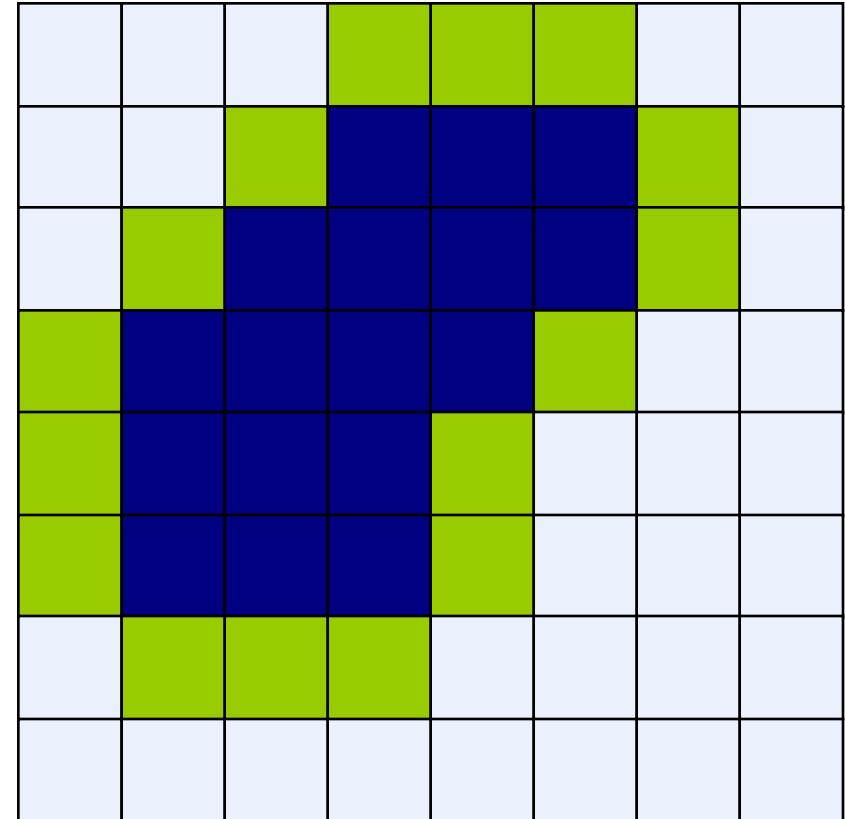


# Dilation Example

Original Image



Processed Image With Dilated Pixels



Structuring Element

# Dilation Example 1



Original image



Dilation by  $3 \times 3$   
square structuring  
element



Dilation by  $5 \times 5$   
square structuring  
element

**Watch out:** In these examples a 1 refers to a black pixel!

# Dilation Example 2

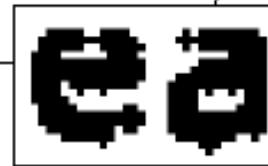
Original image

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.



After dilation

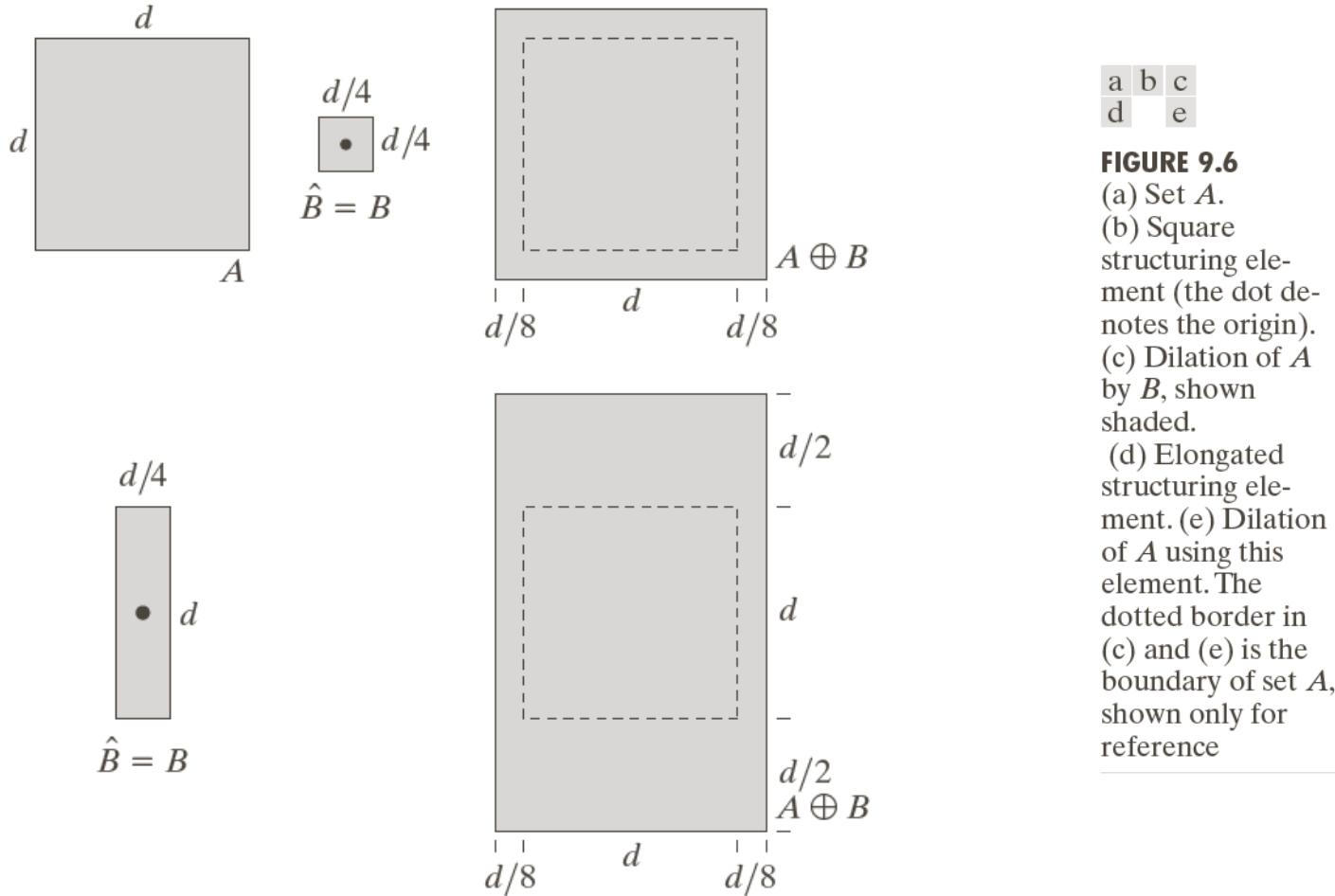
Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



0	1	0
1	1	1
0	1	0

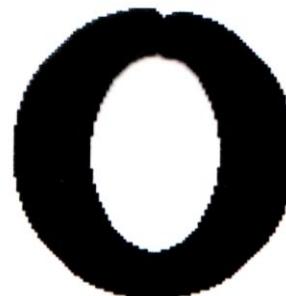
Structuring element

# Dilation Example 3

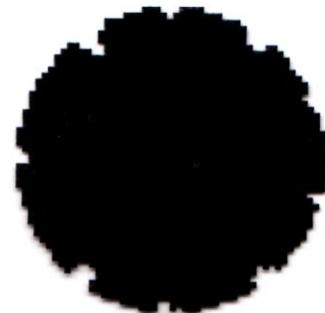


# What Is Dilation For?

Dilation can repair breaks



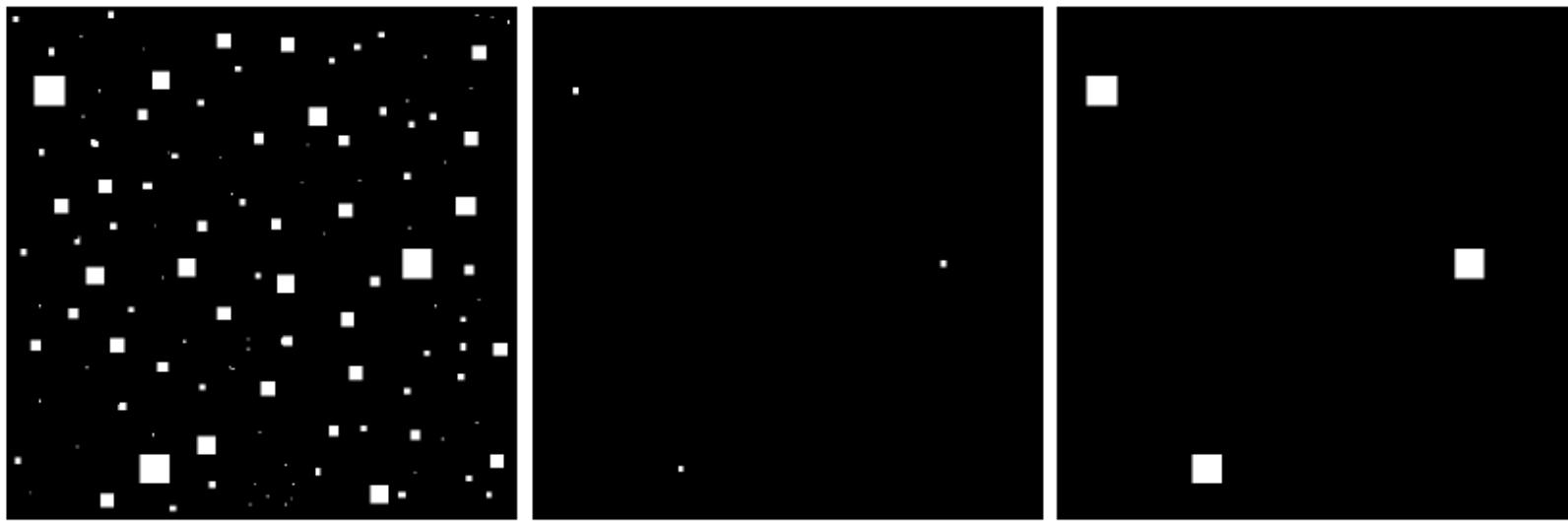
Dilation can repair intrusions



**Watch out:** Dilation enlarges objects

# What Is Dilation For?

Combined with erosion: eliminate irrelevant details



a b c

**FIGURE 9.7** (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

structuring element  $B = 13 \times 13$  pixels of gray level 1

# Compound Operations

More interesting morphological operations can be performed by performing combinations of erosions and dilations

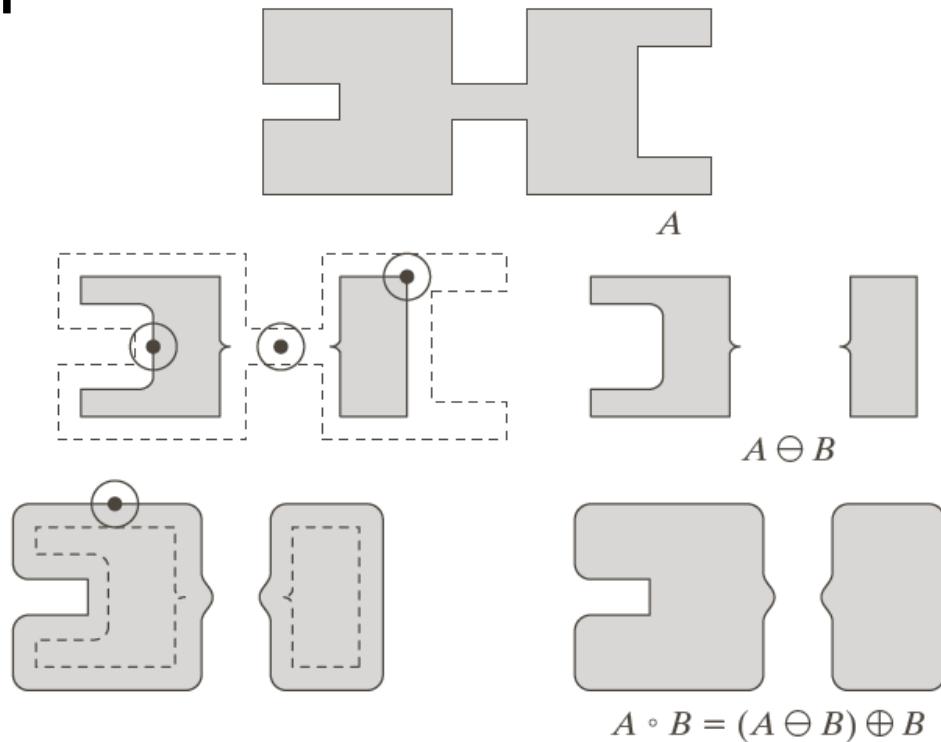
The most widely used of these *compound operations* are:

- Opening
- Closing

# Opening

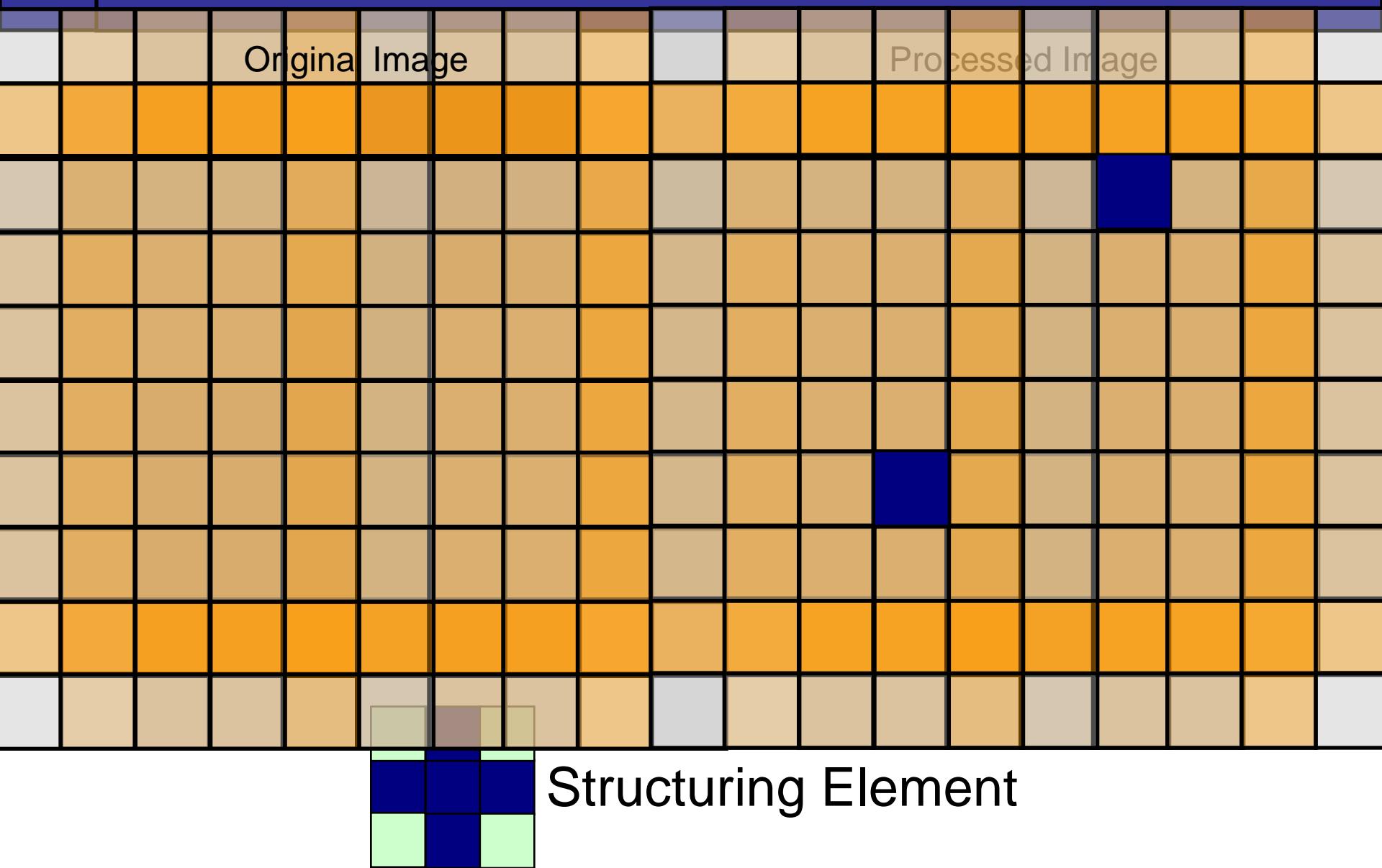
The opening of image  $f$  by structuring element  $s$ , denoted  $f \circ s$  is simply an erosion followed by a dilation

$$f \circ s = (f \ominus s) \oplus s$$



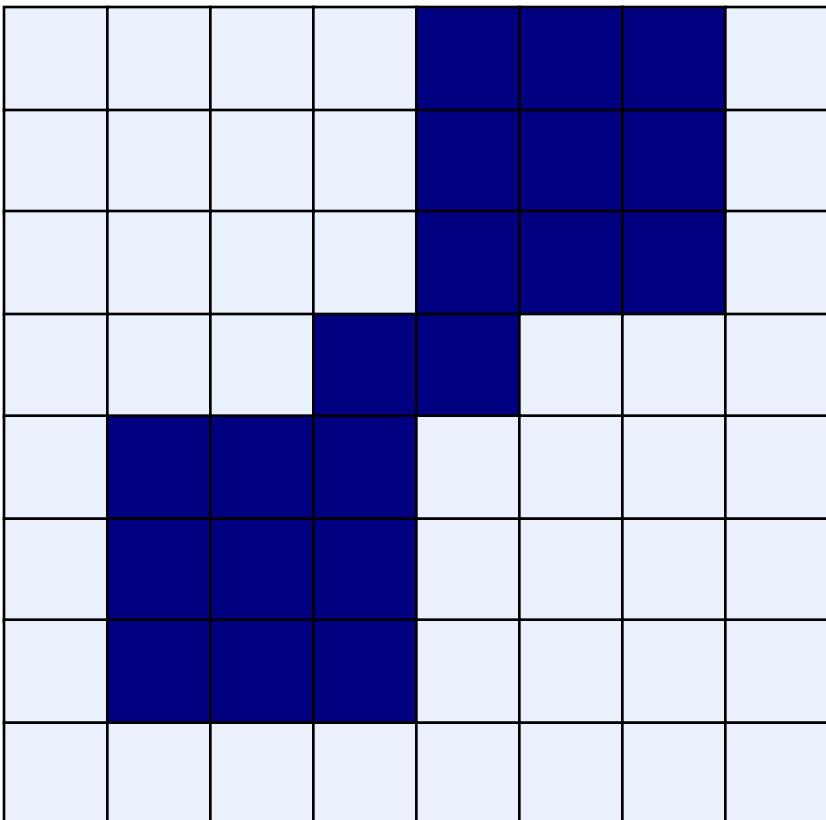
Note a disc shaped structuring element is used

# Opening Example

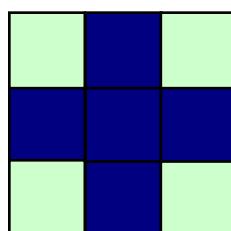
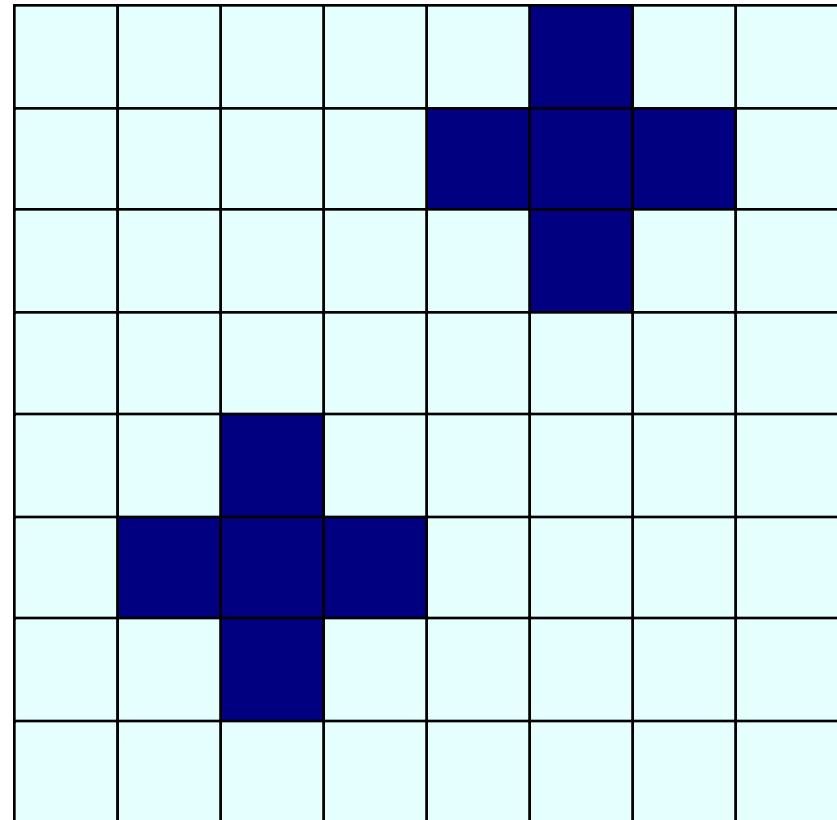


# Opening Example

Original Image

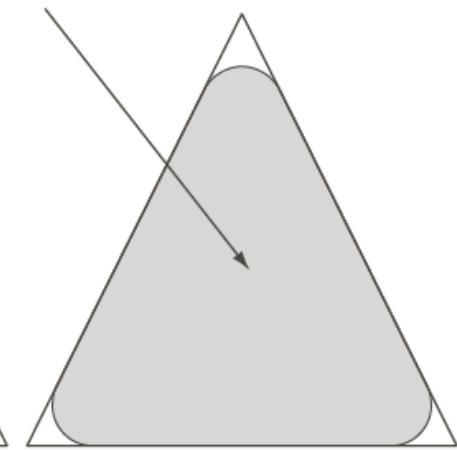
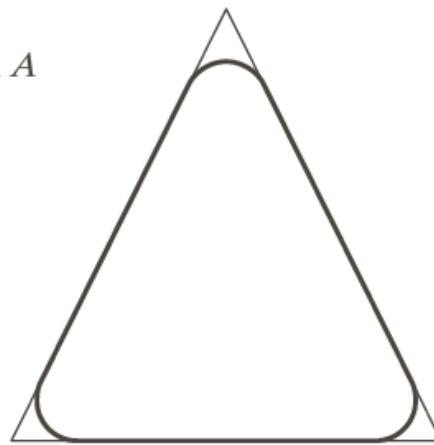
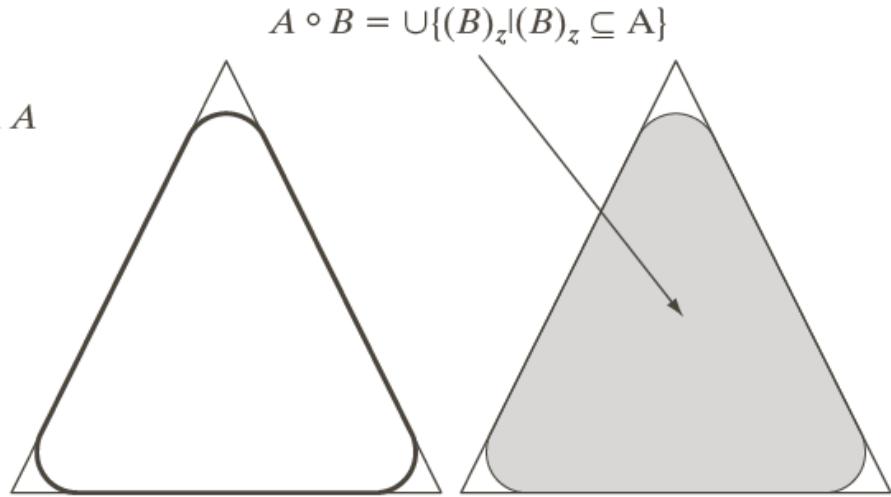
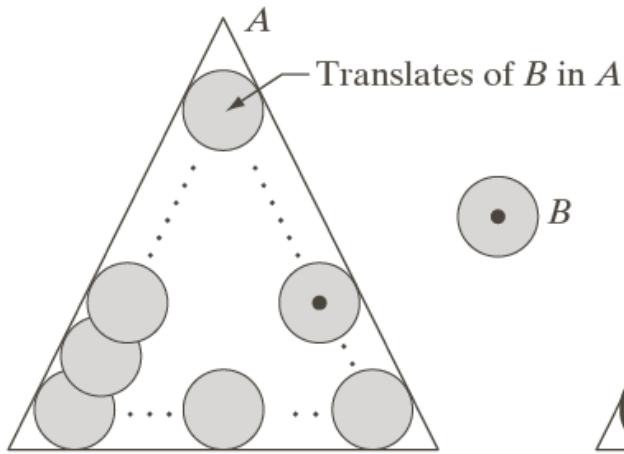


Processed Image



Structuring Element

# Opening Examples



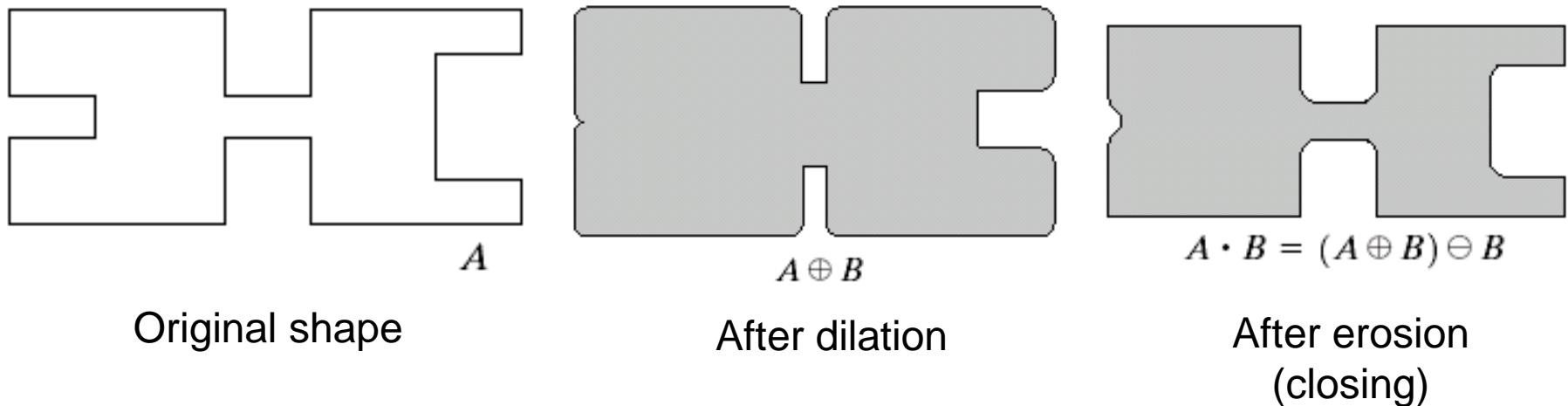
a b c d

**FIGURE 9.8** (a) Structuring element  $B$  “rolling” along the inner boundary of  $A$  (the dot indicates the origin of  $B$ ). (b) Structuring element. (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded). We did not shade  $A$  in (a) for clarity.

$$A \circ B = \cup\{(B)_z | (B)_z \subseteq A\}$$

The closing of image  $f$  by structuring element  $s$ , denoted  $f \cdot s$  is simply a dilation followed by an erosion

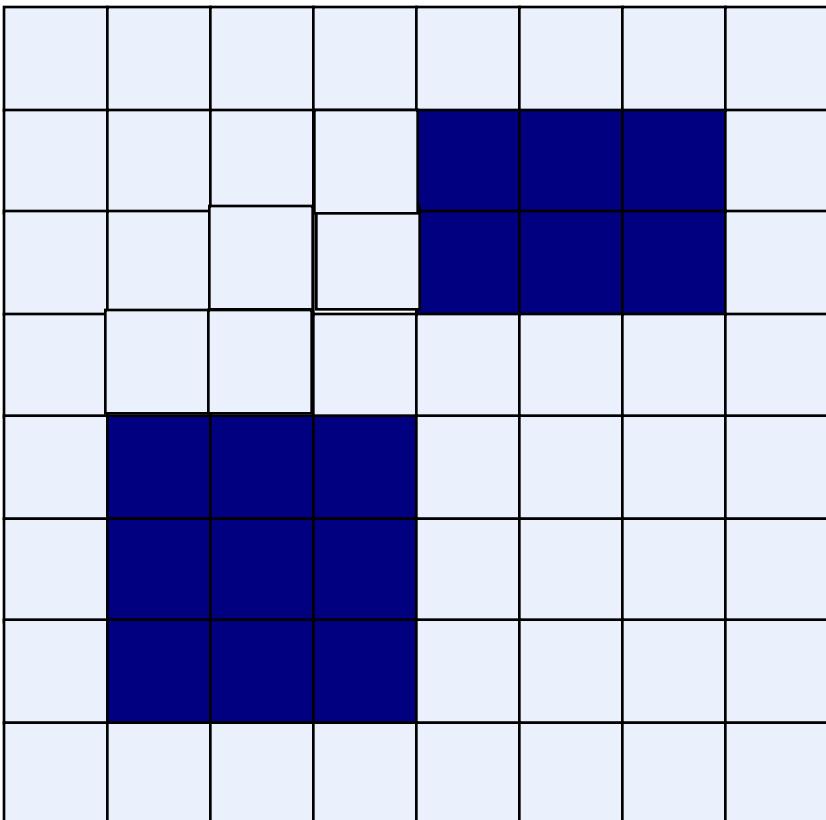
$$f \cdot s = (f \oplus s) \ominus s$$



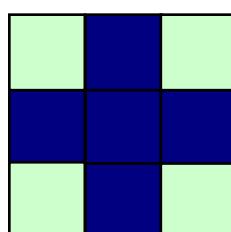
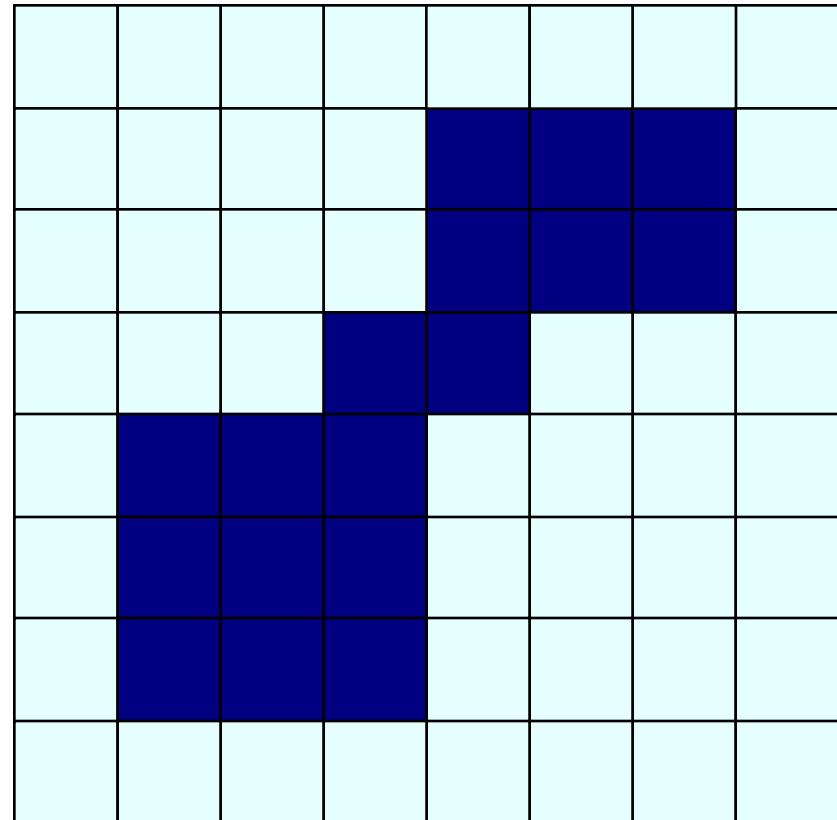
Note a disc shaped structuring element is used

# Closing Example

Original Image

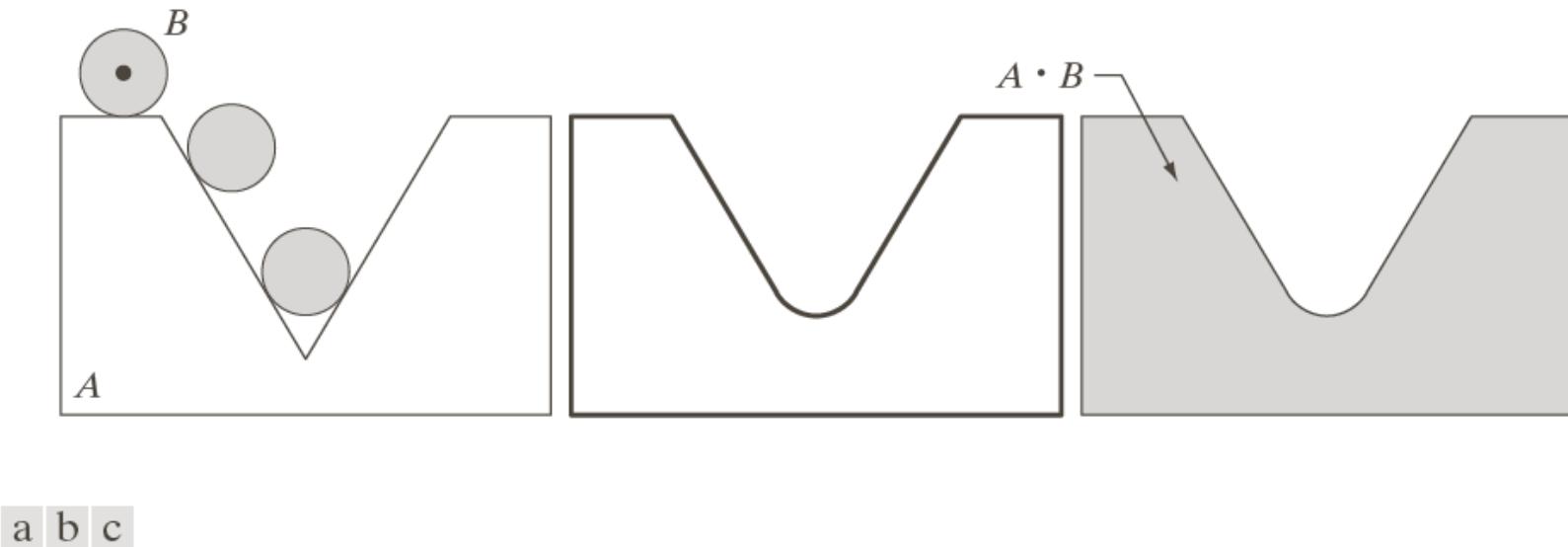


Processed Image



Structuring Element

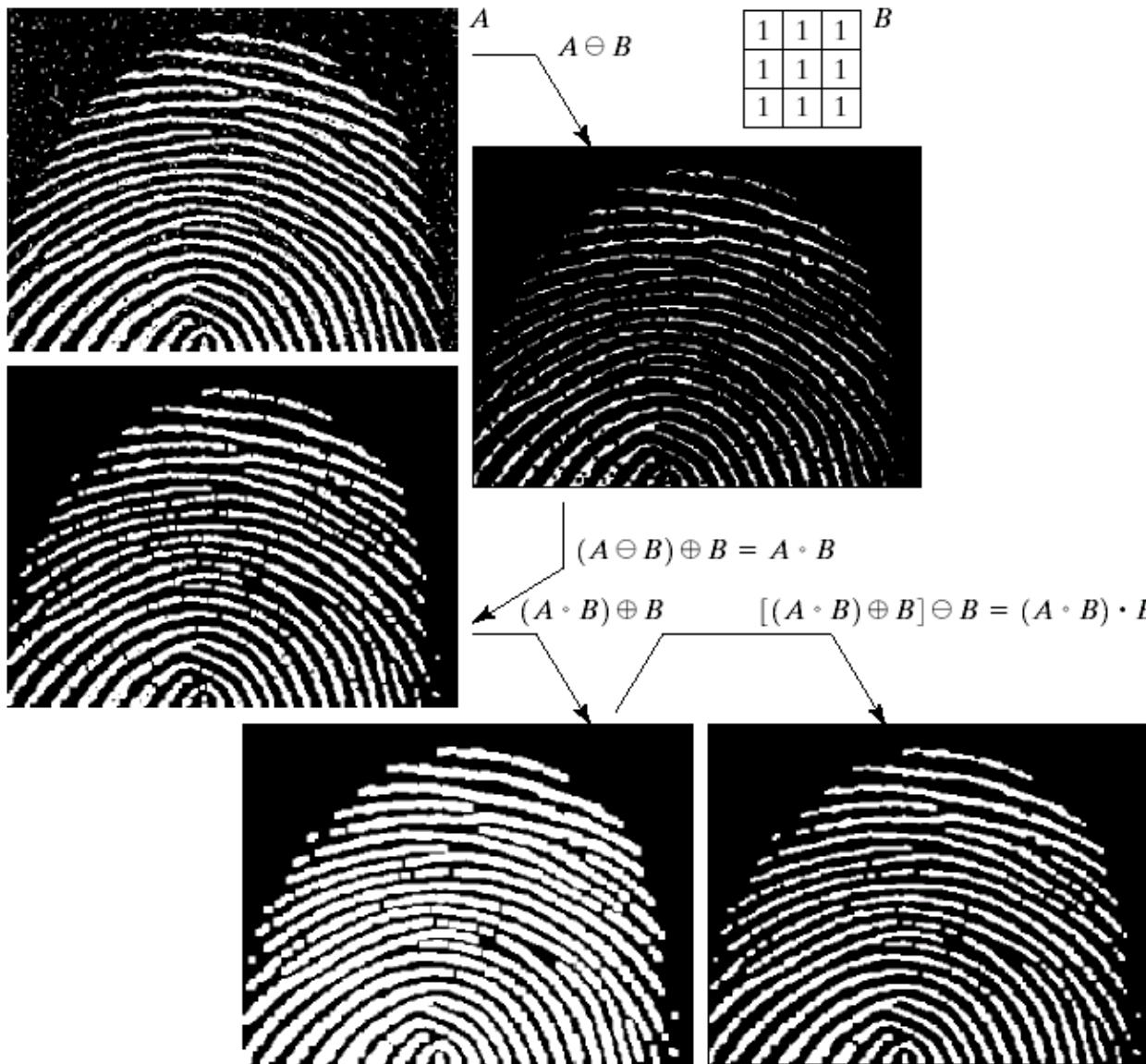
# Closing Example



a b c

**FIGURE 9.9** (a) Structuring element  $B$  “rolling” on the outer boundary of set  $A$ . (b) The heavy line is the outer boundary of the closing. (c) Complete closing (shaded). We did not shade  $A$  in (a) for clarity.

# Morphological Processing Example



# Morphological Algorithms

Using the simple technique we have looked at so far we can begin to consider some more interesting morphological algorithms

Such as:

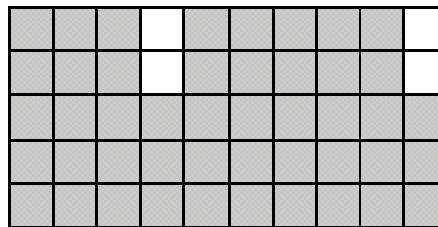
- Boundary extraction
- Region filling
- Extraction of connected components
- Hit or Miss Transformation
- Convex Hull
- Thinning/thickening

# Boundary Extraction

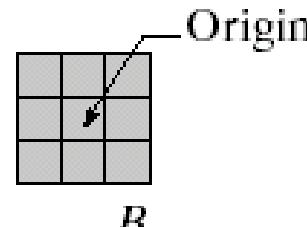
Extracting the boundary (or outline) of an object is often extremely useful

The boundary can be given simply as

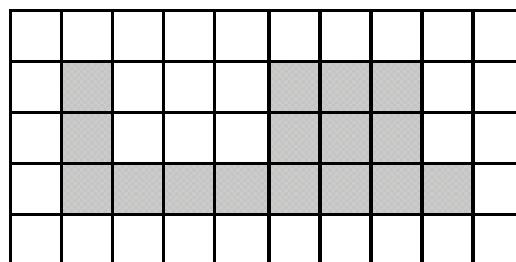
$$\beta(A) = A - (A \ominus B)$$



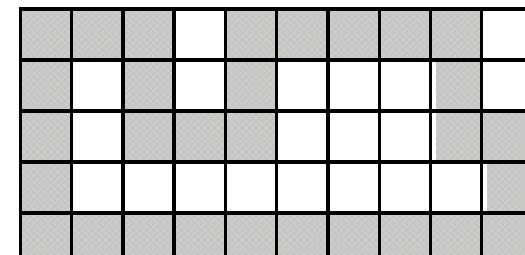
*A*



*B*



*A ⊖ B*



$\beta(A)$



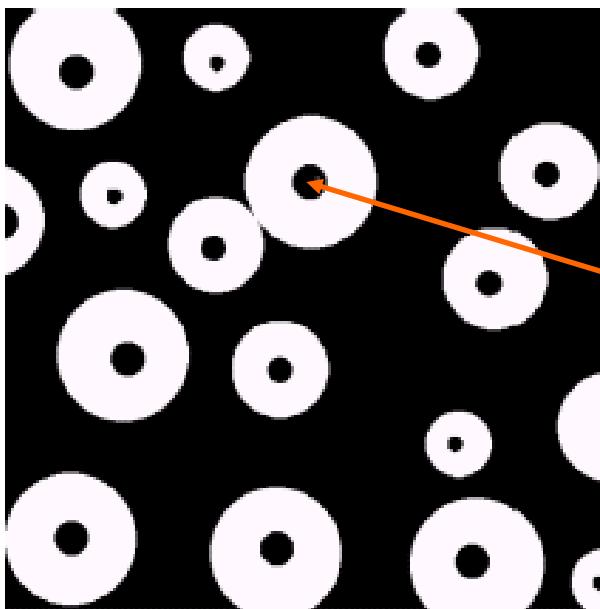
# Boundary Extraction Example

A simple image and the result of performing boundary extraction using a square  $3 \times 3$  structuring element



# Region Filling

Given a pixel inside a boundary, *region filling* attempts to fill that boundary with object pixels (1s)



Given a point inside here, can we fill the whole circle?

# Region Filling (cont...)

The key equation for region filling is

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \dots$$

Where  $X_0$  is simply the starting point inside the boundary,  $B$  is a simple structuring element and  $A^c$  is the complement of  $A$

This equation is applied repeatedly until  $X_k$  is equal to  $X_{k-1}$

Finally the result is unioned with the original boundary

# Region Filling Step By Step

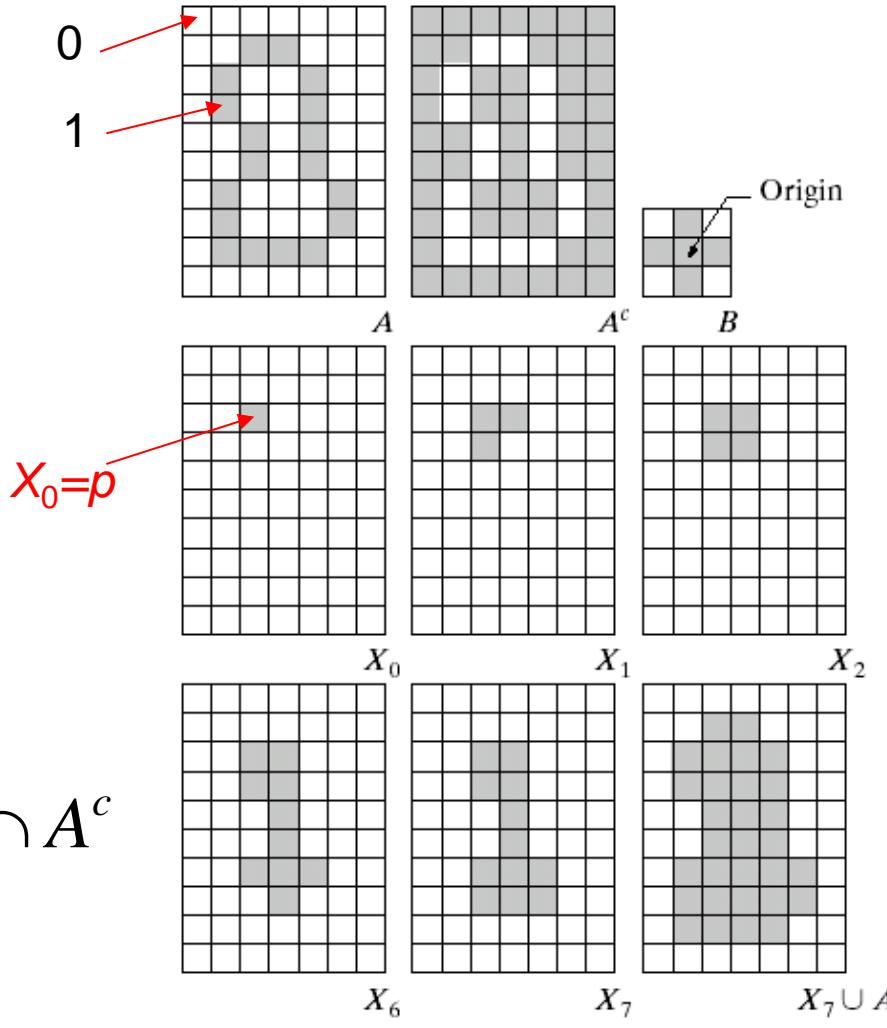
a	b	c
d	e	f
g	h	i

**FIGURE 9.15**

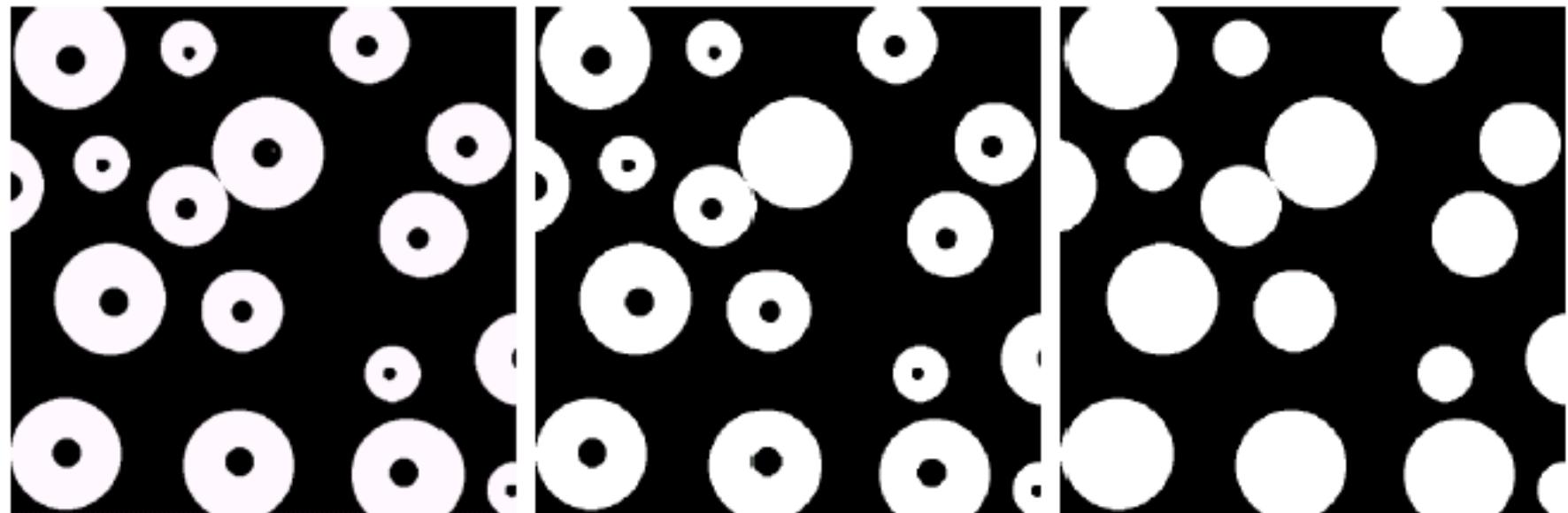
Region filling.

- (a) Set  $A$ .
- (b) Complement of  $A$ .
- (c) Structuring element  $B$ .
- (d) Initial point inside the boundary.
- (e)–(h) Various steps of Eq. (9.5-2).
- (i) Final result [union of (a) and (h)].

$$X_k = (X_{k-1} \oplus B) \cap A^c$$



# Region Filling Example

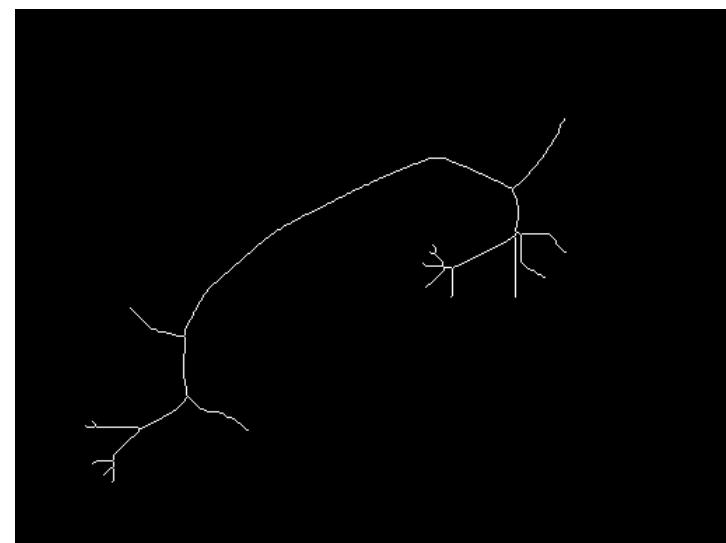
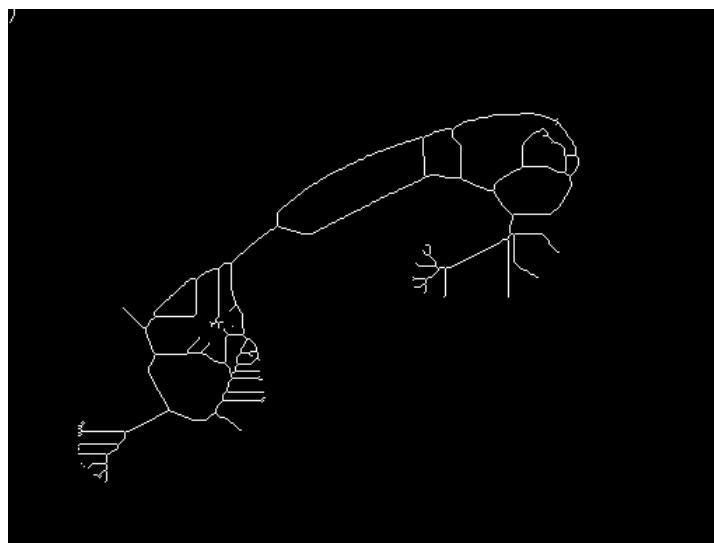
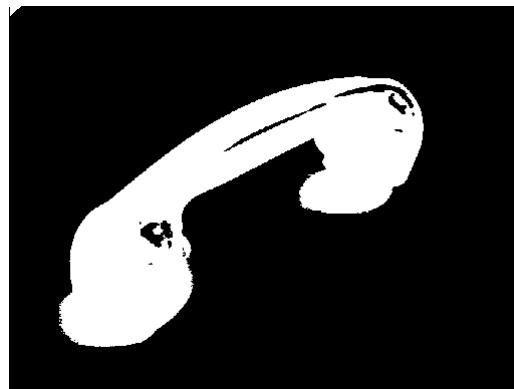


Original Image

One Region  
Filled

All Regions  
Filled

# Utility for further processing

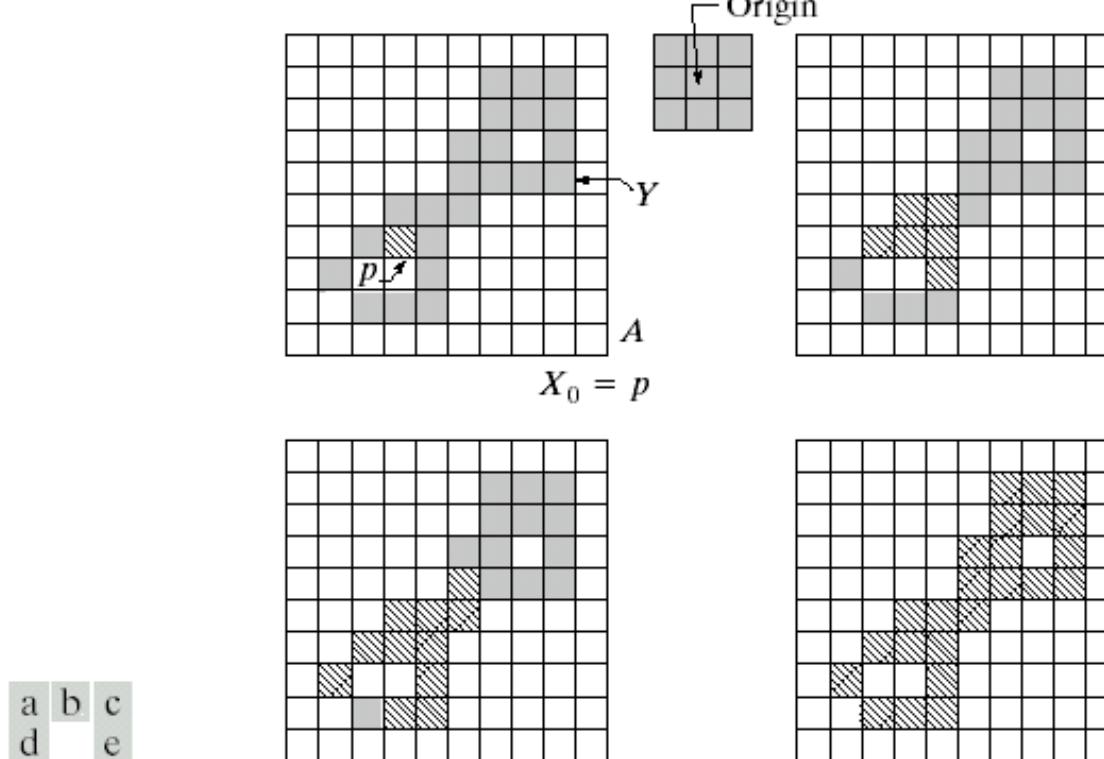


skeleton before closing

skeleton after closing

# Extraction of connected components

$$X_k = (X_{k-1} \oplus B) \cap A$$



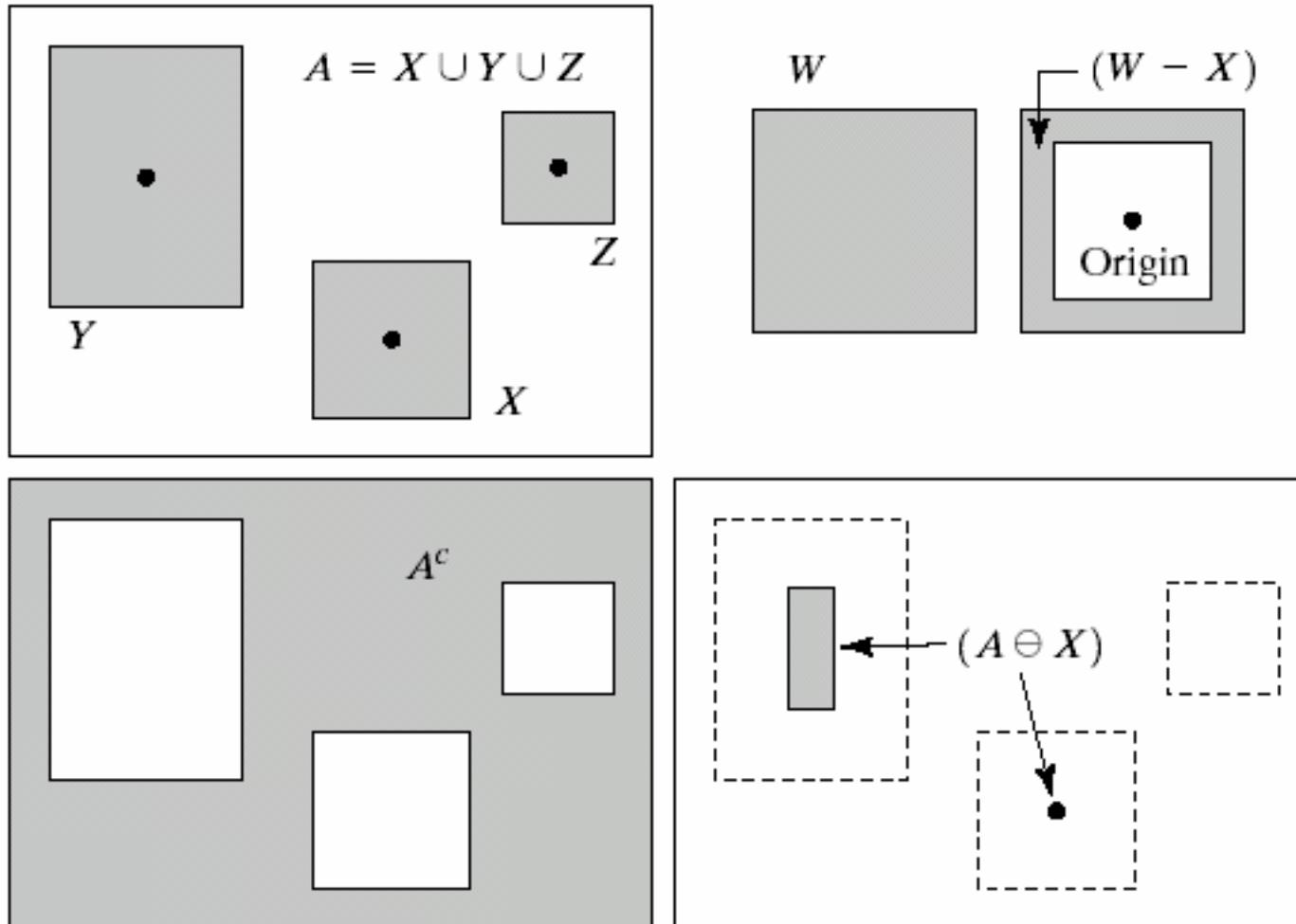
**FIGURE 9.17** (a) Set  $A$  showing initial point  $p$  (all shaded points are valued 1, but are shown different from  $p$  to indicate that they have not yet been found by the algorithm). (b) Structuring element. (c) Result of first iterative step. (d) Result of second step. (e) Final result.

# The Hit-or-Miss Transformation

- Used to **look for particular patterns** of foreground and background pixels
- Very basic tool for shape detection
- Input:
  - Binary Image
  - Group of Structuring Elements, containing 0s, 1s and don't cares(!)

# The Hit-or-Miss Transformation

$$A \Theta B = (A \Theta X) \cap [A^c \Theta (W - X)] \quad B = (X, W - X)$$

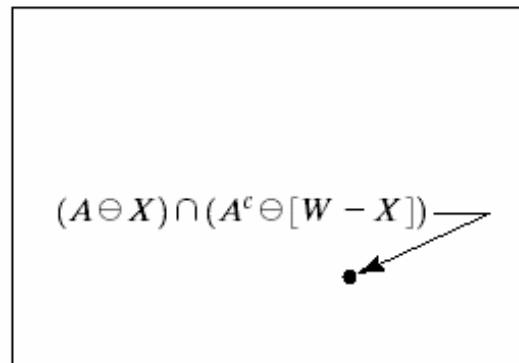
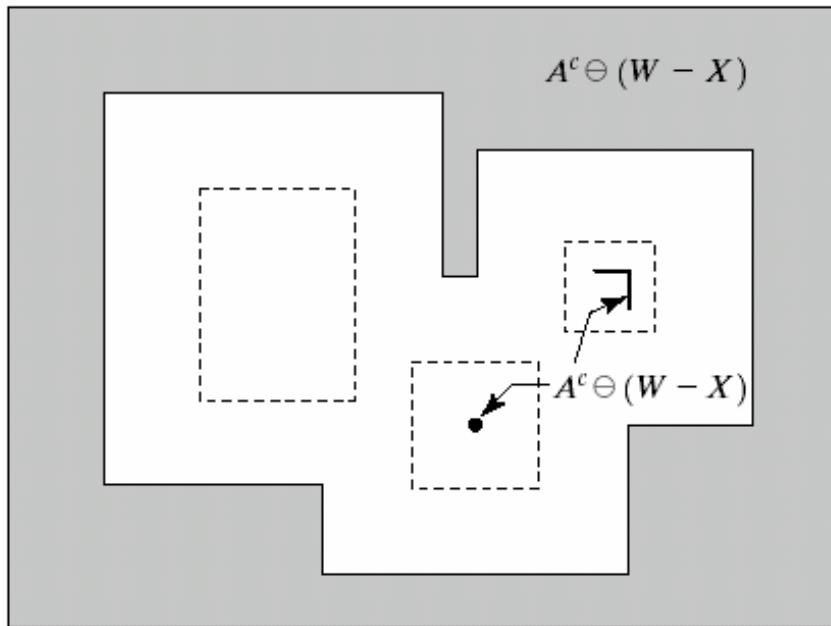
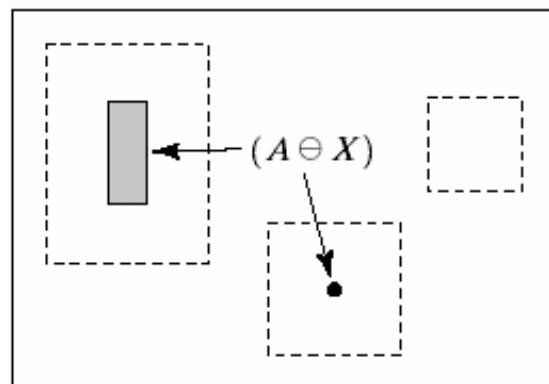


a	b
c	d
e	
f	

**FIGURE 9.12**  
 (a) Set  $A$ . (b) A window,  $W$ , and the local background of  $X$  with respect to  $W$ ,  $(W - X)$ .  
 (c) Complement of  $A$ . (d) Erosion of  $A$  by  $X$ .  
 (e) Erosion of  $A^c$  by  $(W - X)$ .  
 (f) Intersection of (d) and (e), showing the location of the origin of  $X$ , as desired.

# The Hit-or-Miss Transformation

$$A \Theta B = (A \ominus X) \cap [A^c \ominus (W - X)] \quad B = (X, W - X)$$

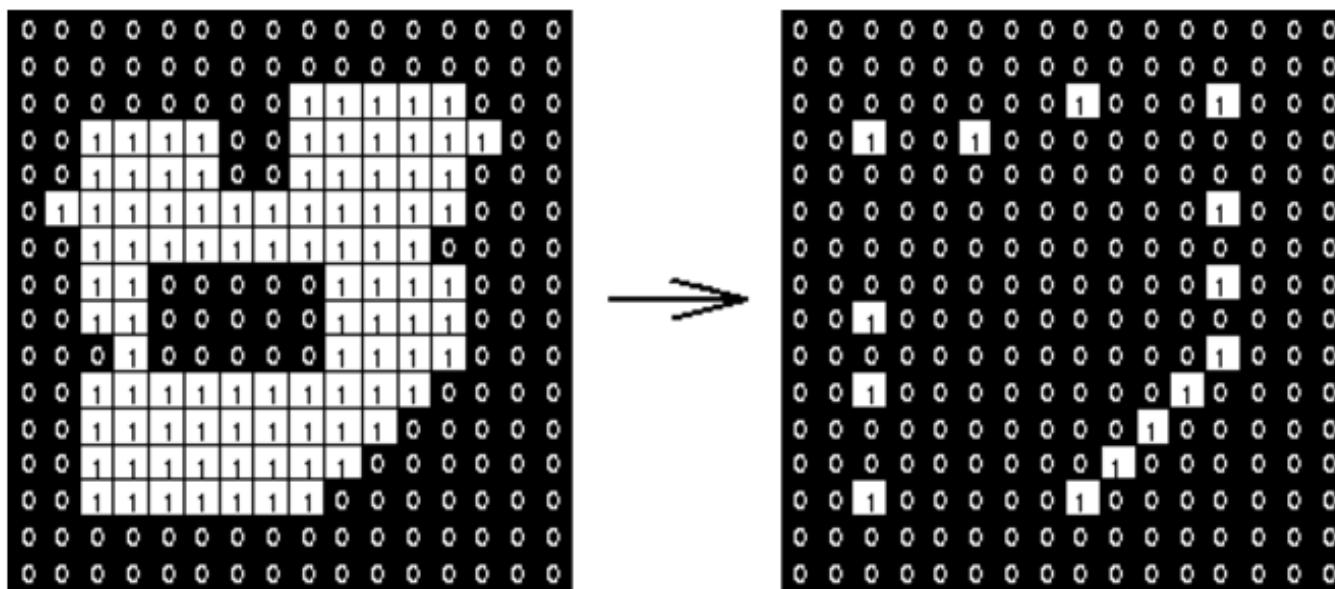


a	b
c	d
e	
f	

**FIGURE 9.12**  
 (a) Set  $A$ . (b) A window,  $W$ , and the local background of  $X$  with respect to  $W$ ,  $(W - X)$ .  
 (c) Complement of  $A$ . (d) Erosion of  $A$  by  $X$ .  
 (e) Erosion of  $A^c$  by  $(W - X)$ .  
 (f) Intersection of (d) and (e), showing the location of the origin of  $X$ , as desired.

# Corner Detection

- To find all the right angle convex corners of a region in a given image as shown below?



# Corner Detection

- Structuring Elements representing four corners
- Contains 0s, 1s and *don't care*'s

x	1	x
0	1	1
0	0	x

x	1	x
1	1	0
x	0	0

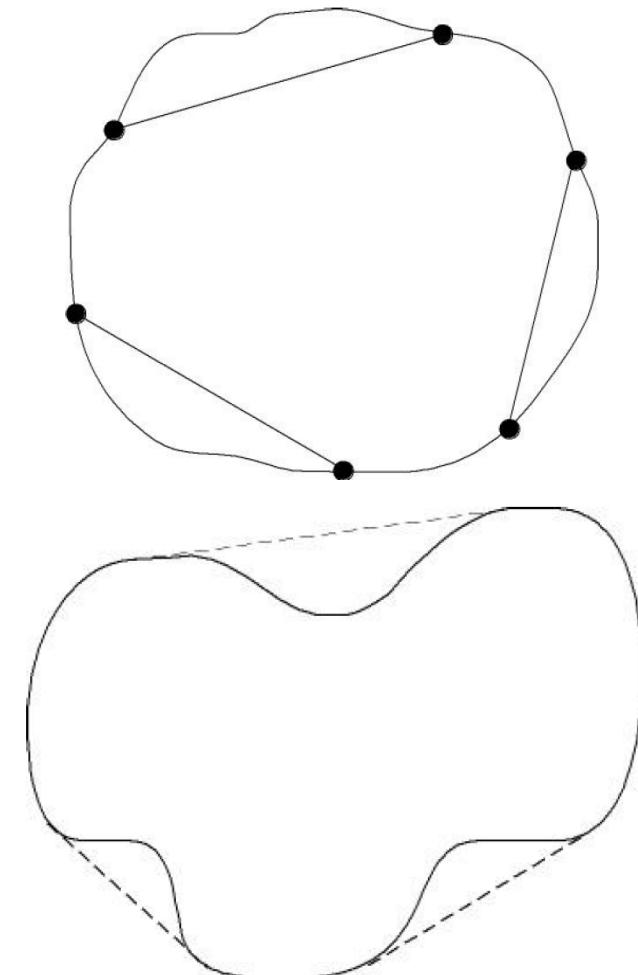
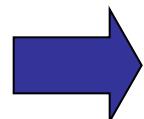
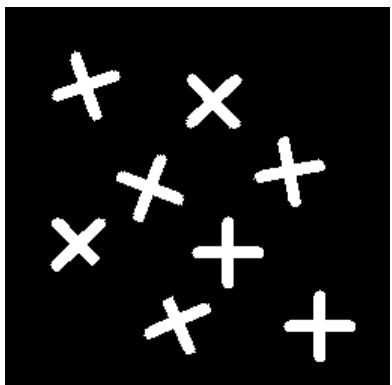
x	0	0
1	1	0
x	1	x

0	0	x
0	1	1
x	1	x

- Apply each Structuring Element
- Use OR operation to combine the four results

# Convex Hull

- A set  $A$  is said to be **convex** if the straight line segment joining any two points in  $A$  lies entirely within  $A$ .
- The **convex hull**  $H$  of an arbitrary set  $S$  is the smallest convex set containing  $S$ .



# Convex Hull

Let  $B^i$ ,  $i = 1, 2, 3, 4$ , represent the four structuring elements.

The procedure consists of implementing the equation:

$$X_k^i = (X_{k-1} \circledast B^i) \cup A$$

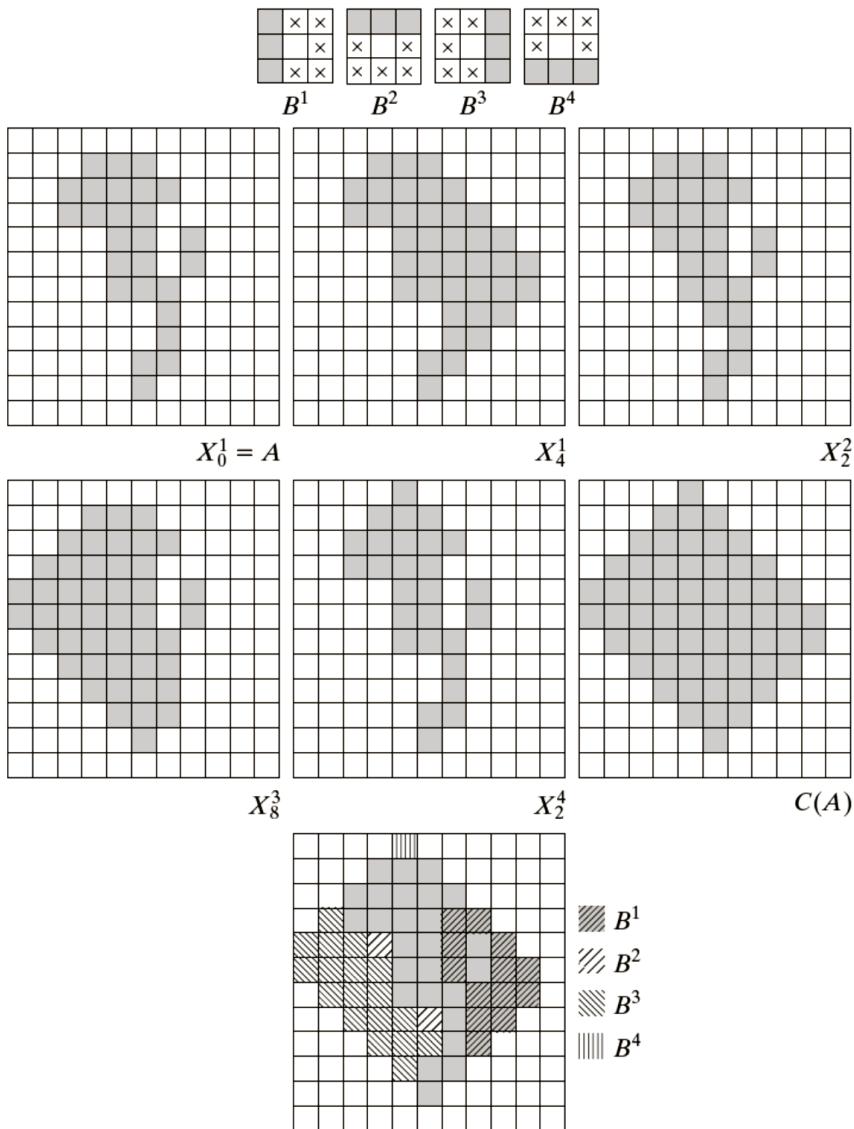
$$i = 1, 2, 3, 4 \text{ and } k = 1, 2, 3, \dots$$

with  $X_0^i = A$ .

When the procedure converges, or  $X_k^i = X_{k-1}^i$ , let  $D^i = X_k^i$ ,  
the convex hull of A is

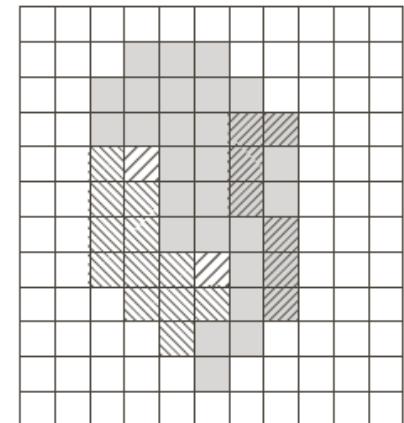
$$C(A) = \bigcup_{i=1}^4 D^i$$

# Convex Hull



**FIGURE 9.19**

(a) Structuring elements. (b) Set  $A$ . (c)–(f) Results of convergence with the structuring elements shown in (a). (g) Convex hull. (h) Convex hull showing the contribution of each structuring element.



**FIGURE 9.20**

Result of limiting growth of the convex hull algorithm to the maximum dimensions of the original set of points along the vertical and horizontal directions.

1. Used to **remove** selected **foreground pixels** from binary images
2. After edge detection, lines are often **thicker than one pixel.**
3. Thinning can be used to thin those line to **one pixel width.**
4. Several applications, but is particularly useful for [skeletonization](#)

- The thinning of a set  $A$  by a structuring element  $B$ , defined:  
$$\begin{aligned} A \otimes B &= A - (A \oplus B) \\ &= A \cap (A \oplus B)^c \end{aligned}$$
- A more useful expression for thinning  $A$  symmetrically is based on a sequence of structuring elements:

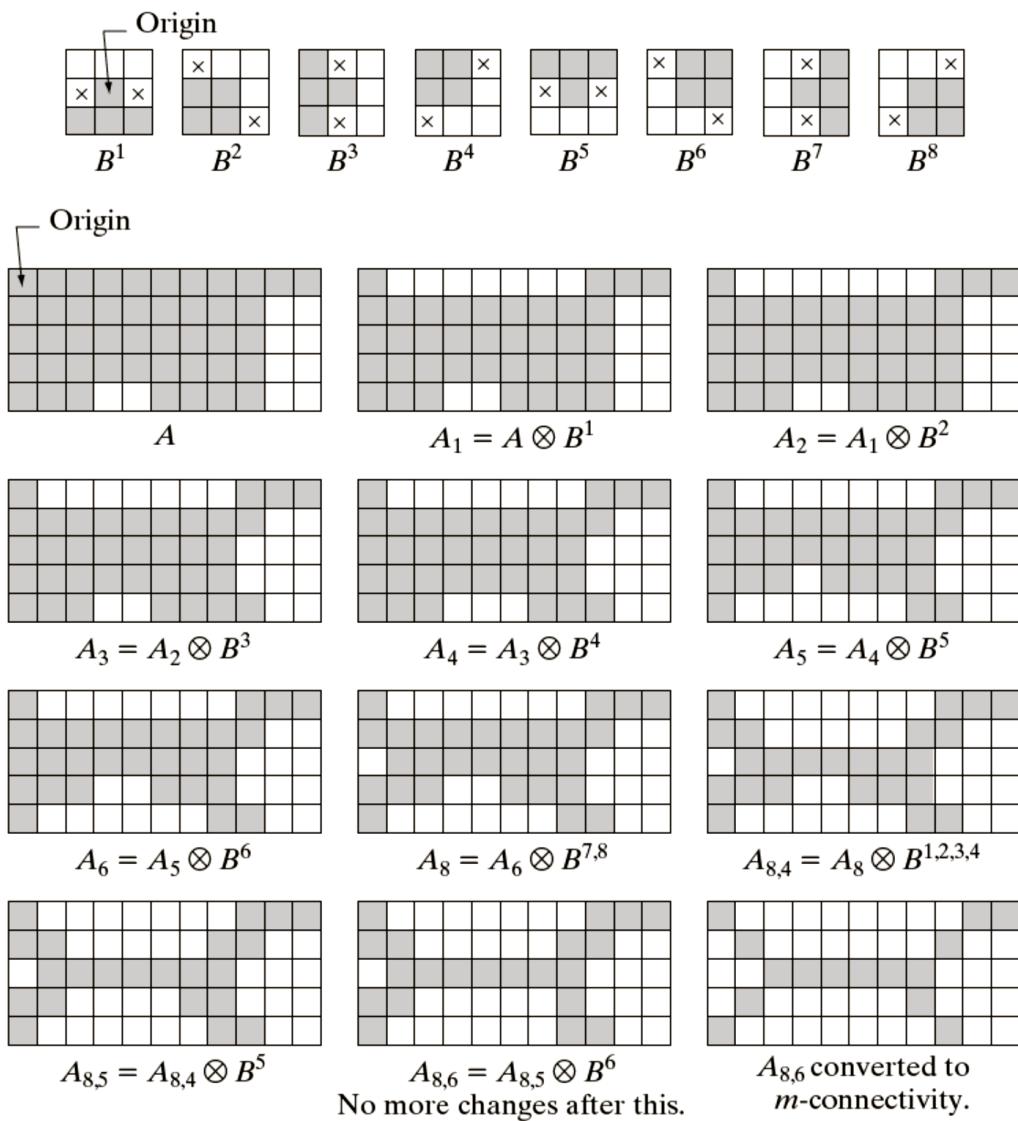
$$\{B\} = \{B^1, B^2, B^3, \dots, B^n\}$$

where  $B^i$  is a rotated version of  $B^{i-1}$

The thinning of  $A$  by a sequence of structuring element  $\{B\}$

$$A \otimes \{B\} = (((((A \otimes B^1) \otimes B^2) \dots) \otimes B^n))$$

# Thining



**FIGURE 9.21** (a) Sequence of rotated structuring elements used for thinning. (b) Set  $A$ . (c) Result of thinning with the first element. (d)-(i) Results of thinning with the next seven elements (there was no change between the seventh and eighth elements). (j) Result of using the first four elements again. (l) Result after convergence. (m) Conversion to  $m$ -connectivity.

# Summary

The purpose of morphological processing is primarily to remove imperfections added during segmentation

The basic operations are *erosion* and *dilation*

Using the basic operations we can perform *opening* and *closing*

More advanced morphological operation can then be implemented using combinations of all of these

# Image and Video Processing

Colour Image Processing

## ■ Motive

- Color is a powerful descriptor that often simplifies object identification and extraction from a scene.
- Human can discern thousands of color shades and intensities, compared to about only two dozen shades of gray.

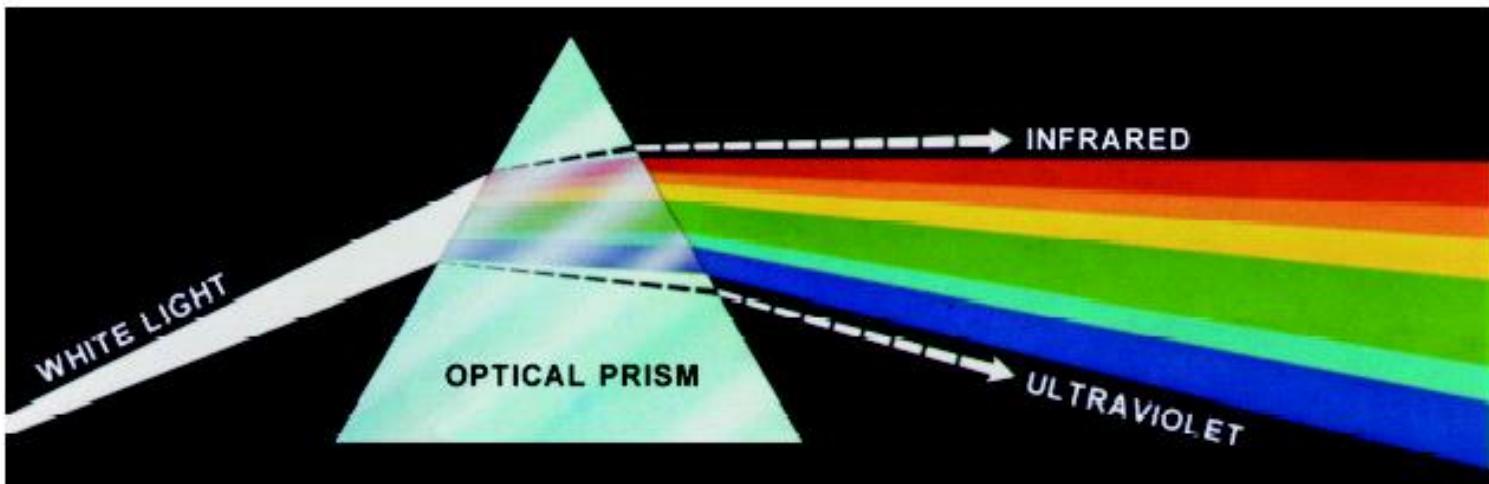
# Introduction

We'll look at color image processing, covering:

- Color fundamentals and models
  - Color spectrum vs. electromagnetic spectrum
  - CIE standard, R, G, B as the primary colors
  - Chromaticity diagram
  - Color models – RGB, HSI etc.
- Color Processing
  - Grey to color - Psuedo coloring
  - Processing using RGB model vs. HSI model
  - Color Enhancement, Segmentation etc.

# Color Spectrum

In 1666 Sir Isaac Newton discovered that when a beam of sunlight passes through a glass prism, the emerging beam is split into a spectrum of colours

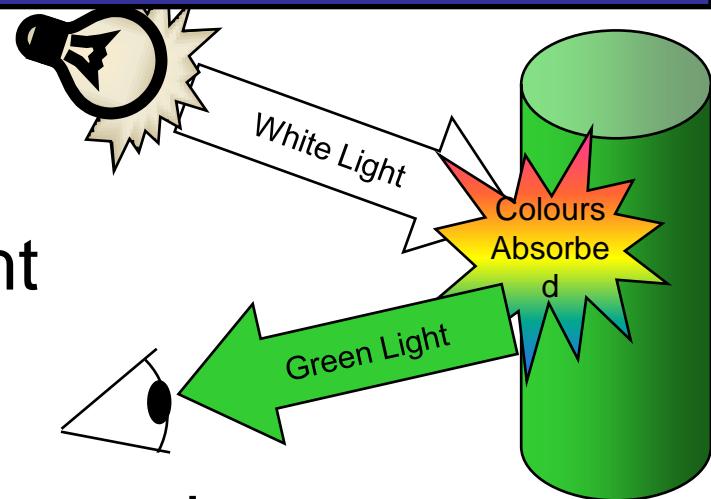


# Some Questions?

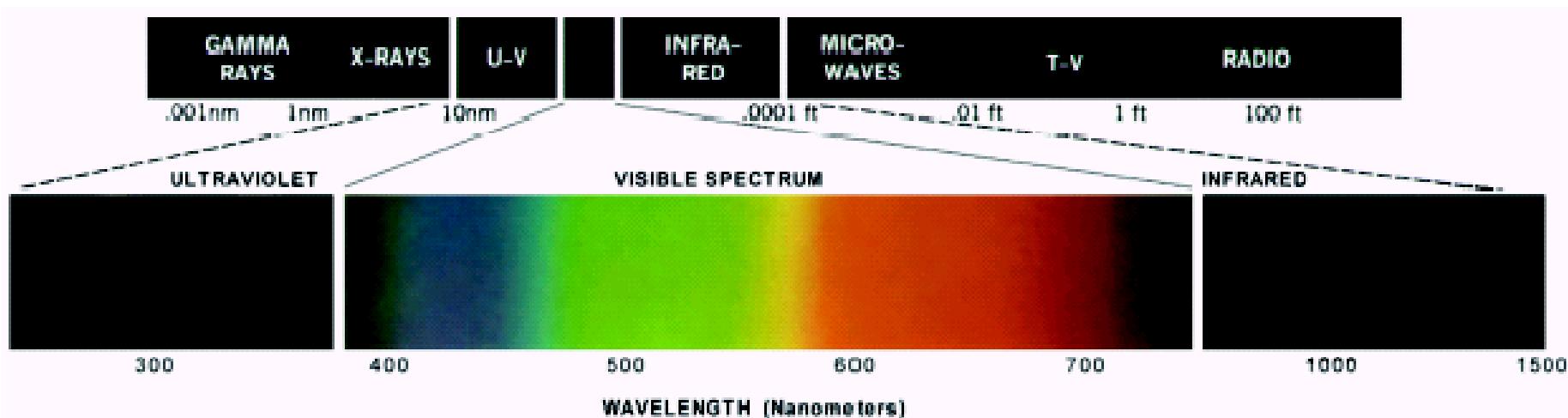
- What does it mean when we say an object is in a certain color?
- Why are the primary colors of human vision red, green, and blue?
- Is it true that different proportions of red, green, and blue can produce **all** the visible color?
- What kind of color model is the most suitable one to describe human vision?  
What is more suitable for digital processing?

# Colors of human vision

The colors that humans and most animals perceive in an object are determined by the nature of the light reflected from the object



Chromatic light spans the electromagnetic spectrum from approximately 400 to 700nm



# Colors of human vision (cont..)

- As we mentioned before human colour vision is achieved through 6 to 7 million cones in each eye
- Approximately 66% of these cones are sensitive to red light, 33% to green light and 3% to blue light
  - For this reason, red, green, and blue are referred to as the primary colors of human vision.
  - Tristimulus: the amount of R, G, B needed to form any color (X, Y, Z)
  - Trichromatic coefficients: x, y, z

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

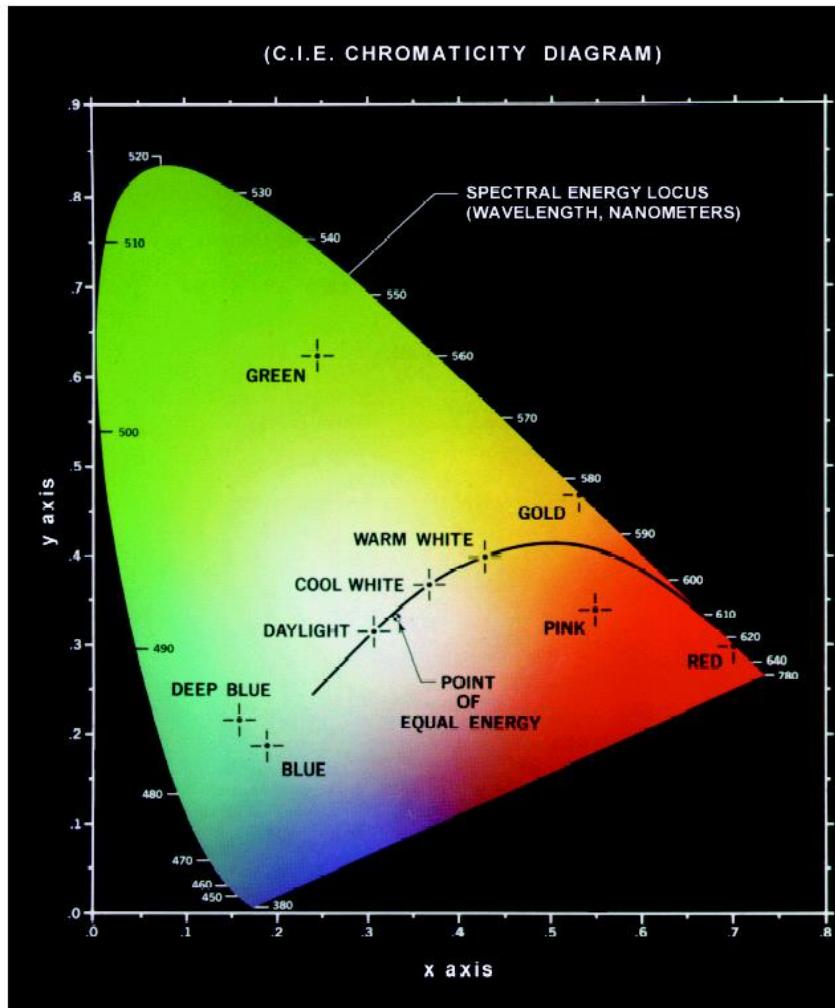
$$z = \frac{Z}{X + Y + Z}$$

$$x + y + z = 1$$

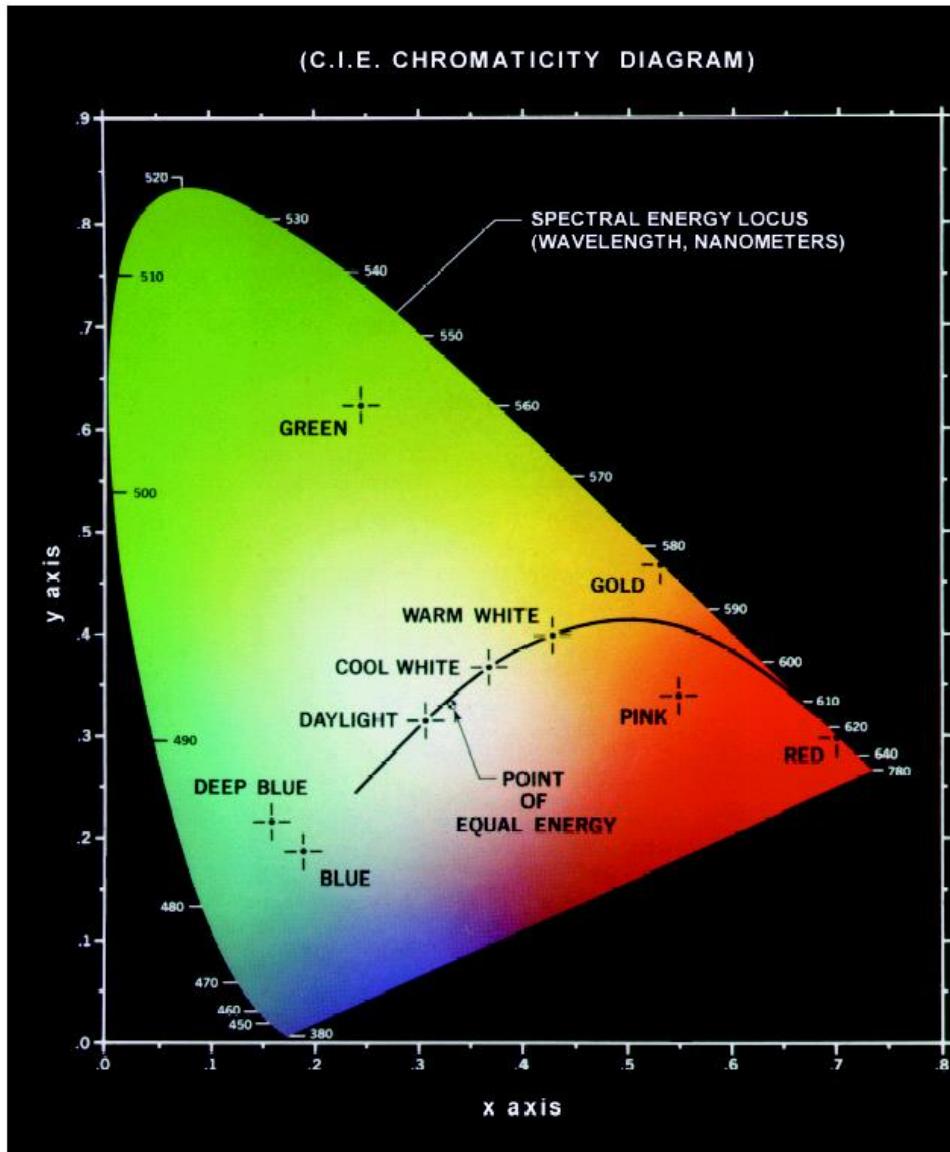
# CIE Chromacity Diagram

- Specifying colors systematically can be achieved using the **CIE chromacity diagram**
- On this diagram the x-axis represents the proportion of red and the y-axis represents the proportion of red used
- The proportion of blue used in a color is calculated as:

$$z = 1 - (x + y)$$



# CIE Chromacity Diagram (cont...)



Green: 62% green, 25% red and 13% blue

Red: 32% green, 67% red and 1% blue

# CIE Chromacity Diagram (cont...)

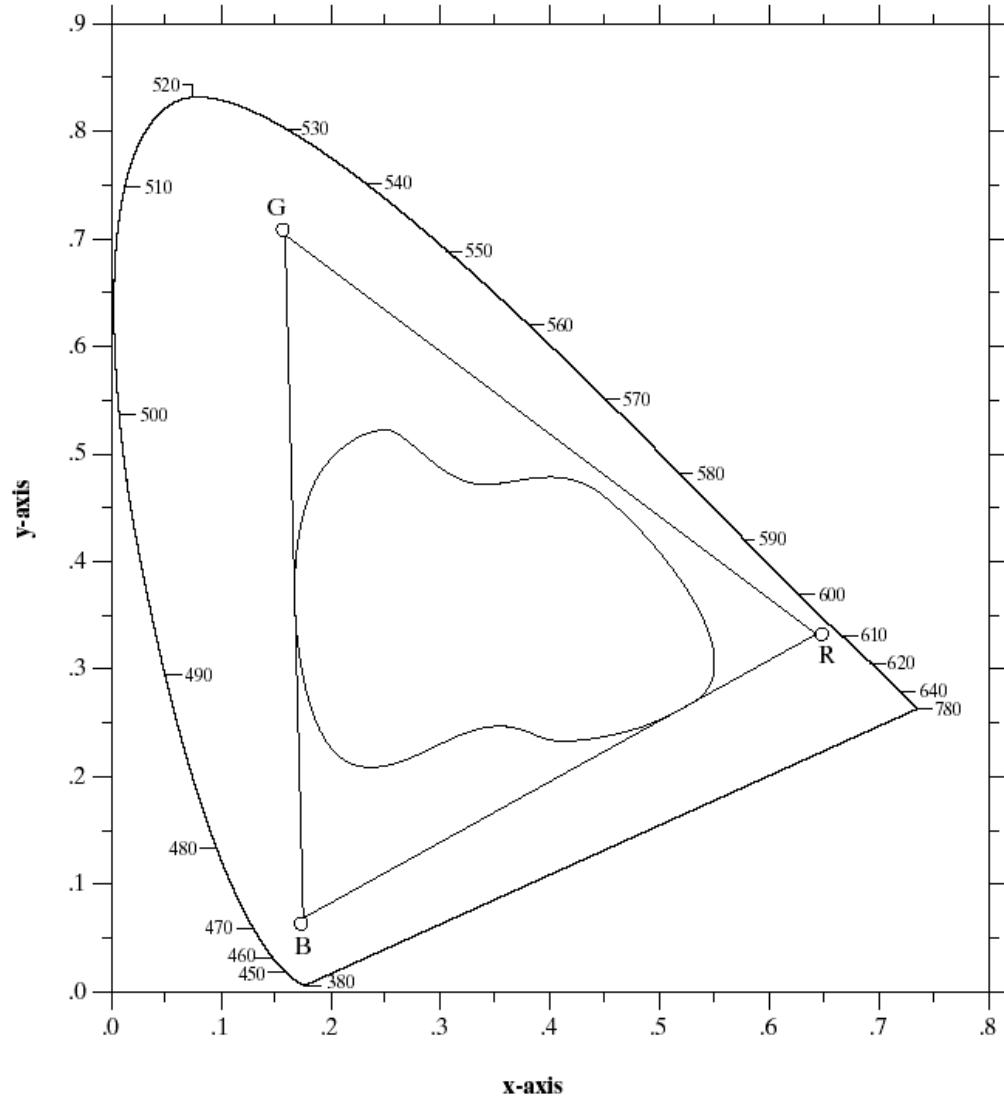
Any colour located on the boundary of the chromacity chart is fully saturated

The point of equal energy has equal amounts of each colour and is the CIE standard for pure white

Any straight line joining two points in the diagram defines all of the different colours that can be obtained by combining these two colours additively

This can be easily extended to three points

# CIE Chromacity Diagram (cont...)



This means the entire colour range cannot be displayed based on any three colours

The triangle shows the typical colour gamut produced by RGB monitors

The strange shape is the gamut achieved by high quality colour printers

# Colour Models

From the previous discussion it should be obvious that there are different ways to model colour

We will consider two very popular models used in colour image processing:

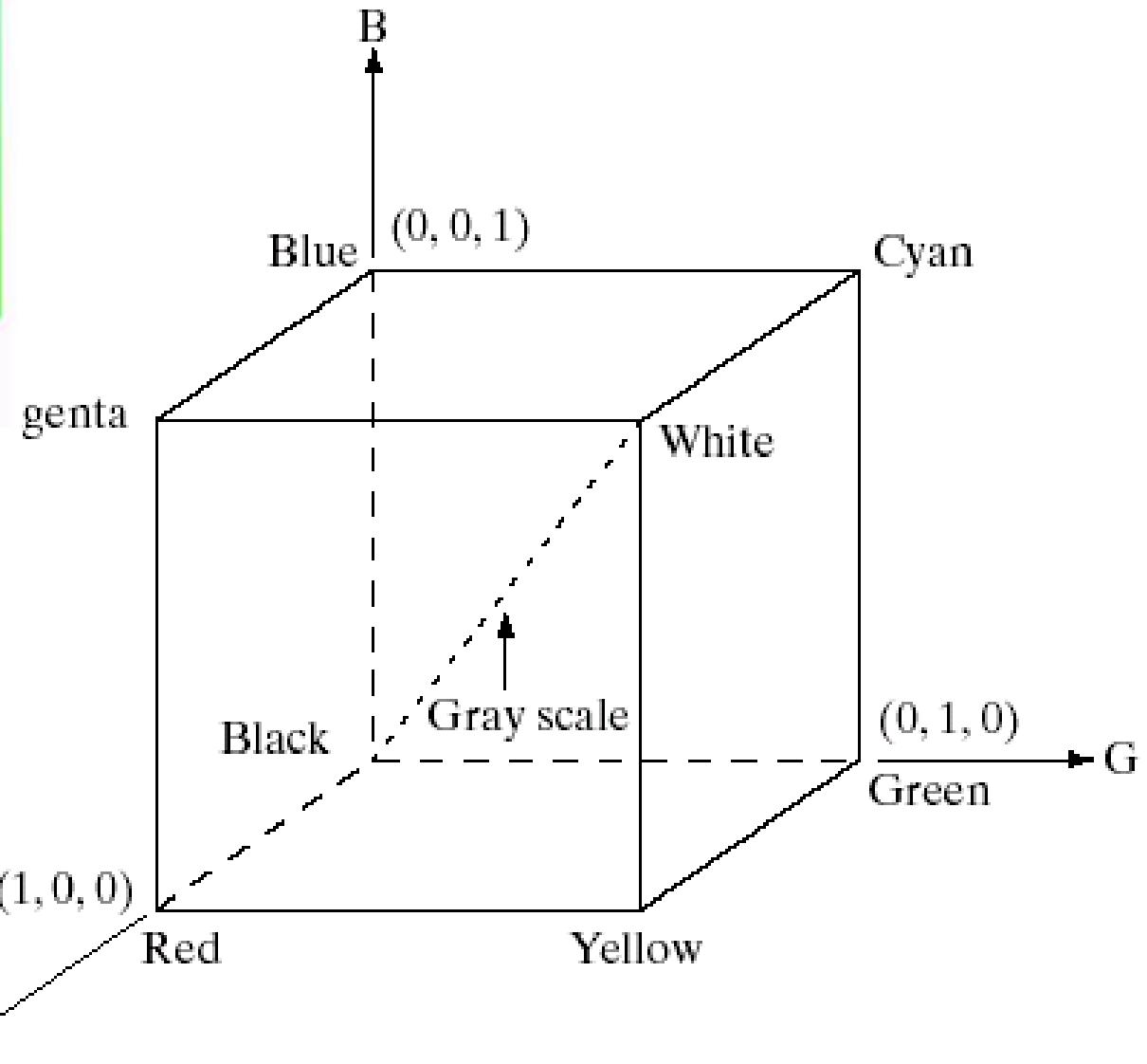
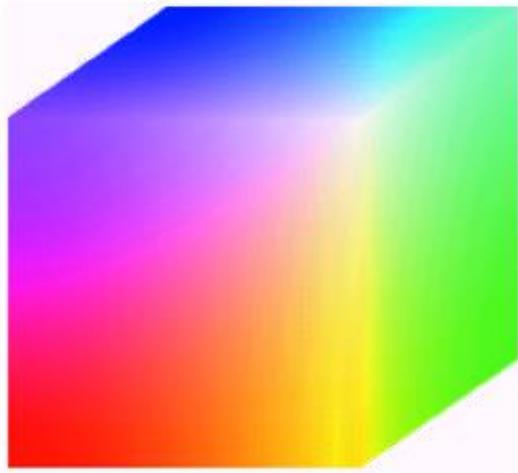
- RGB (**R**ed **G**reen **B**lue)
- HSI (**H**ue **S**aturation **I**ntensity)

In the RGB model each colour appears in its primary spectral components of red, green and blue

The model is based on a Cartesian coordinate system

- RGB values are at 3 corners
- Cyan magenta and yellow are at three other corners
- Black is at the origin
- White is the corner furthest from the origin
- Different colours are points on or inside the cube represented by RGB vectors

# RGB (cont...)



# RGB (cont...)

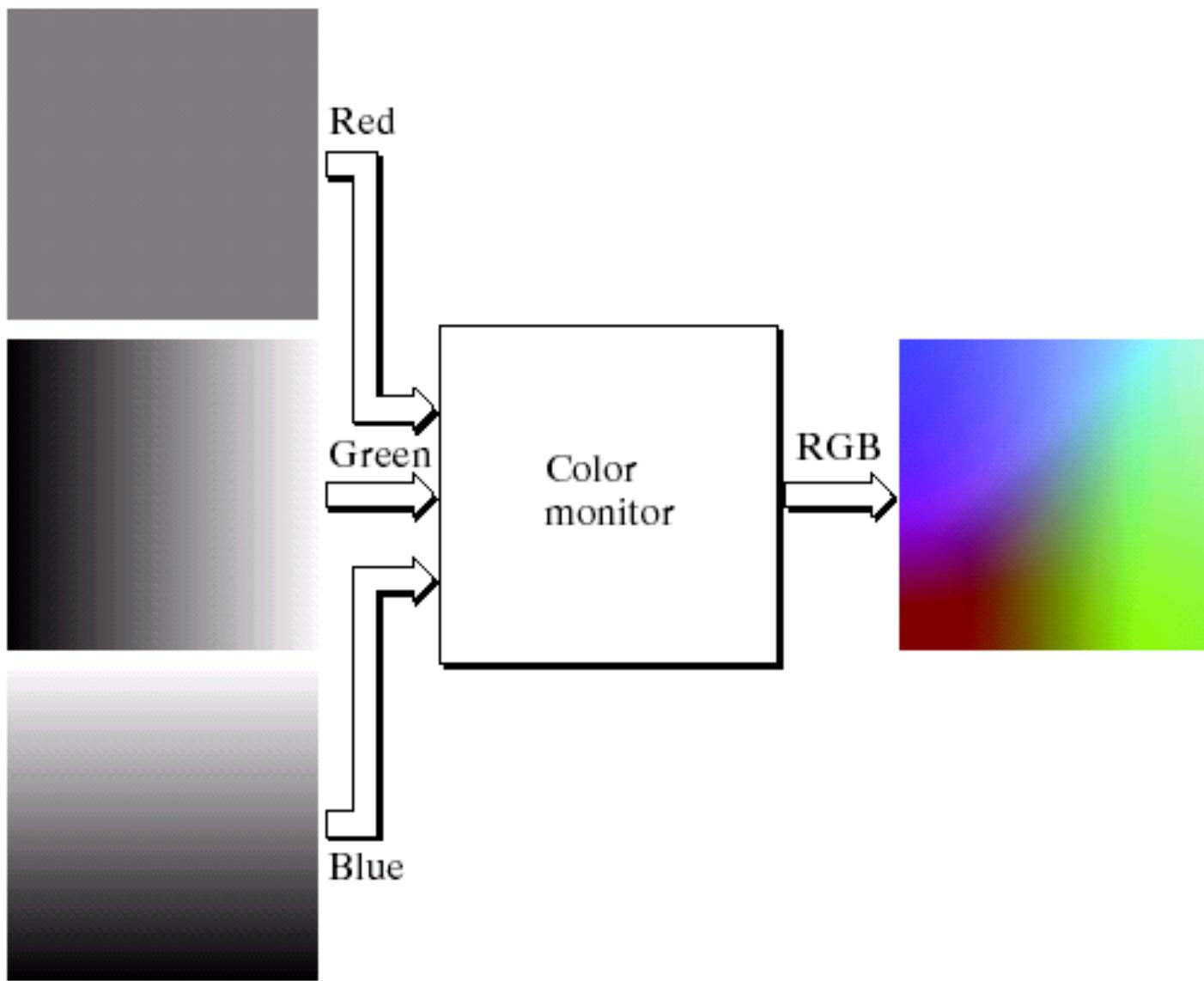
Images represented in the RGB colour model consist of three component images – one for each primary colour

When fed into a monitor these images are combined to create a composite colour image

The number of bits used to represent each pixel is referred to as the colour depth

A 24-bit image is often referred to as a full-colour image as it allows  $(2^8)^3 = 16,777,216$  colours

# RGB (cont...)



# The HSI Colour Model

- RGB is useful for hardware implementations and is related to the way in which the human visual system works
- However, RGB is not a particularly intuitive way in which to describe colours
- Rather when people describe colours they tend to use **hue**, **saturation** and **brightness**
- HSI model is particularly good for colour description and manipulation

# The HSI Colour Model (cont...)

The HSI model uses three measures to describe colours:

- **Hue**: dominant color perceived by an observer (say yellow, orange or red)
- **Saturation**: Gives a measure of relative purity or how much white is mixed with a pure colour (hue)
- **Intensity**: Intensity is the same achromatic notion that we have seen in grey level images.

# HSI, Intensity & RGB

Intensity can be extracted from RGB images – which is not surprising if we stop to think about it

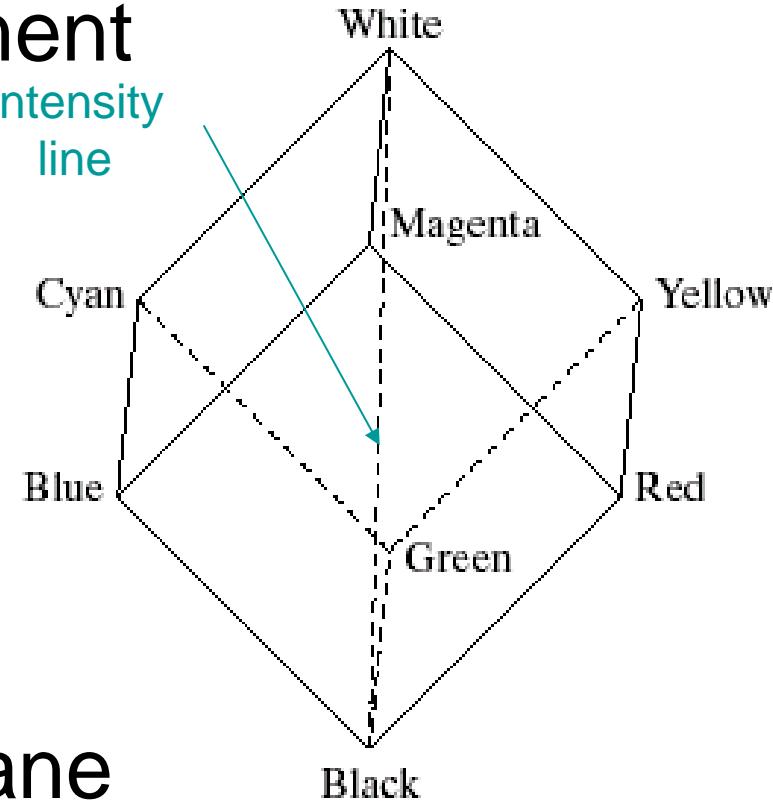
Remember the diagonal on the RGB colour cube that we saw previously ran from black to white

Now consider if we stand this cube on the black vertex and position the white vertex directly above it

# HSI, Intensity & RGB (cont...)

Now the intensity component of any colour can be determined by passing a plane *perpendicular* to the intensity axis and containing the colour point

The intersection of the plane with the intensity axis gives us the intensity component of the colour

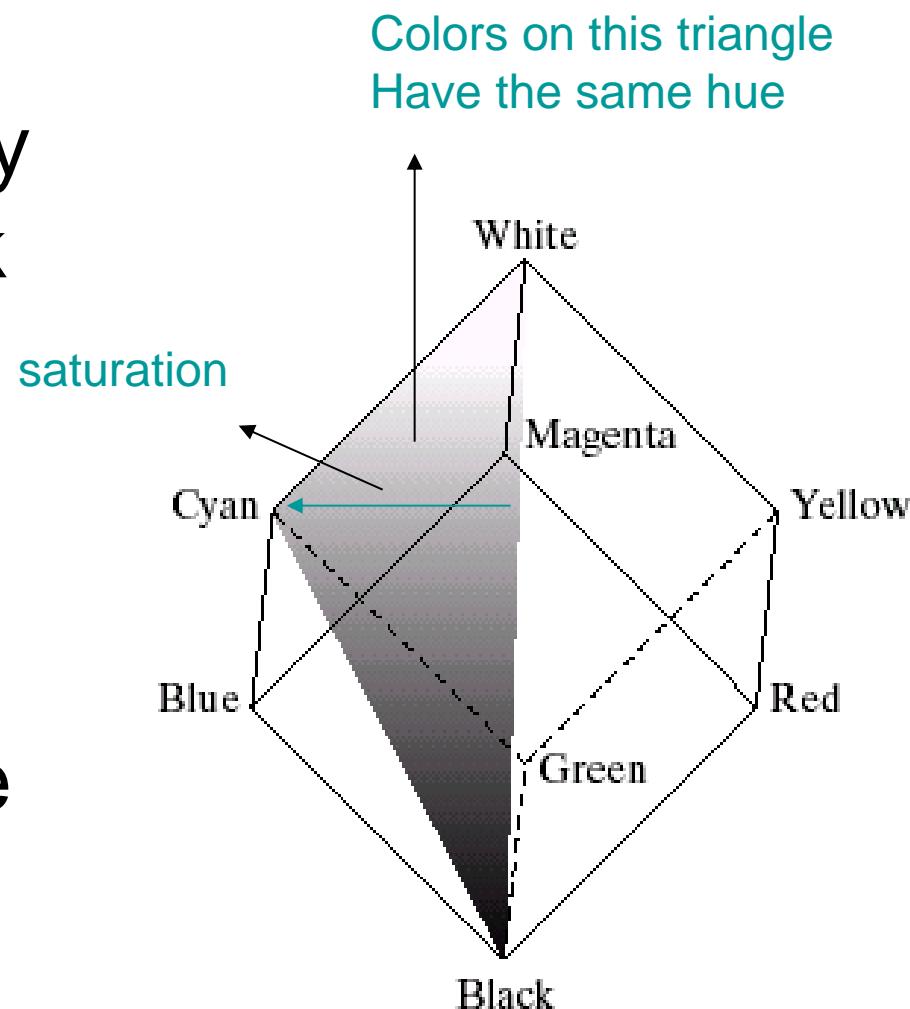


# HSI, Hue & RGB

In a similar way we can extract the hue from the RGB colour cube

Consider a plane defined by the three points cyan, black and white

All points contained in this plane must have the same hue (cyan) as black and white cannot contribute hue information to a colour

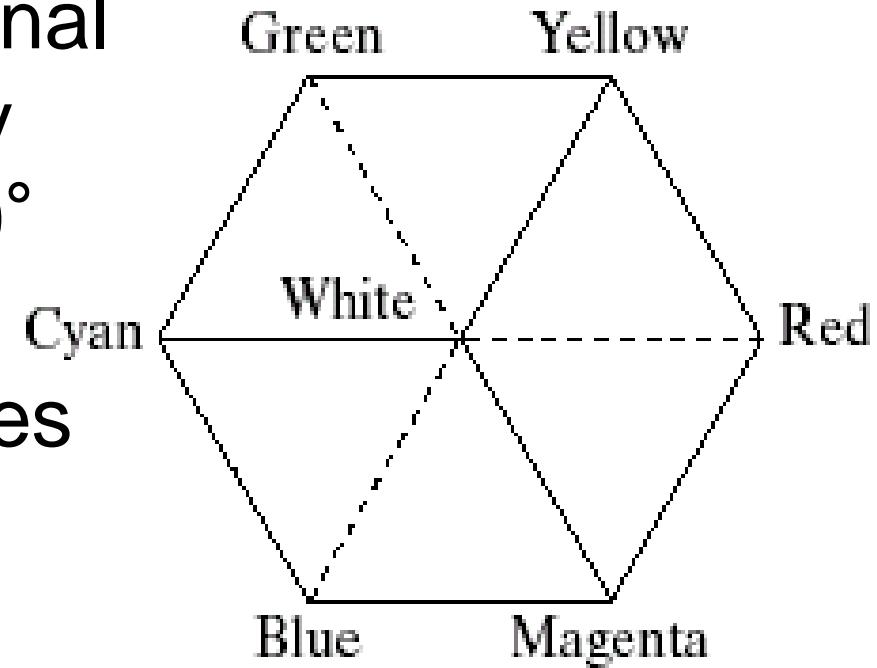


# The HSI Colour Model

Consider if we look straight down at the RGB cube as it was arranged previously

We would see a hexagonal shape with each primary colour separated by  $120^\circ$  and secondary colours at  $60^\circ$  from the primaries

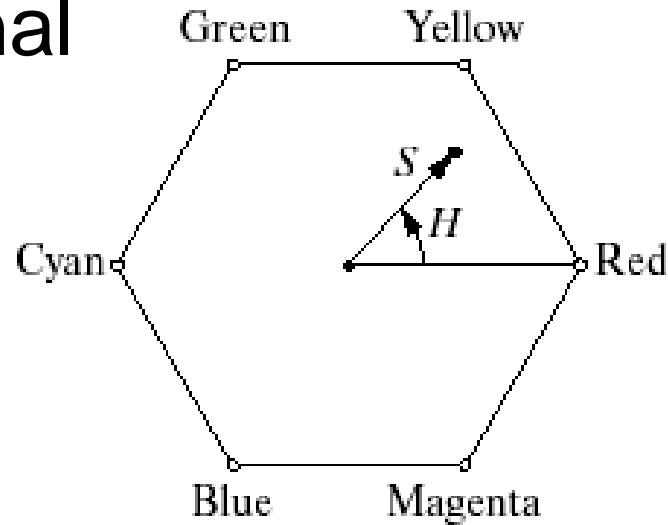
So the HSI model is composed of a vertical intensity axis and the locus of colour points that lie on planes perpendicular to that axis



# The HSI Colour Model (cont...)

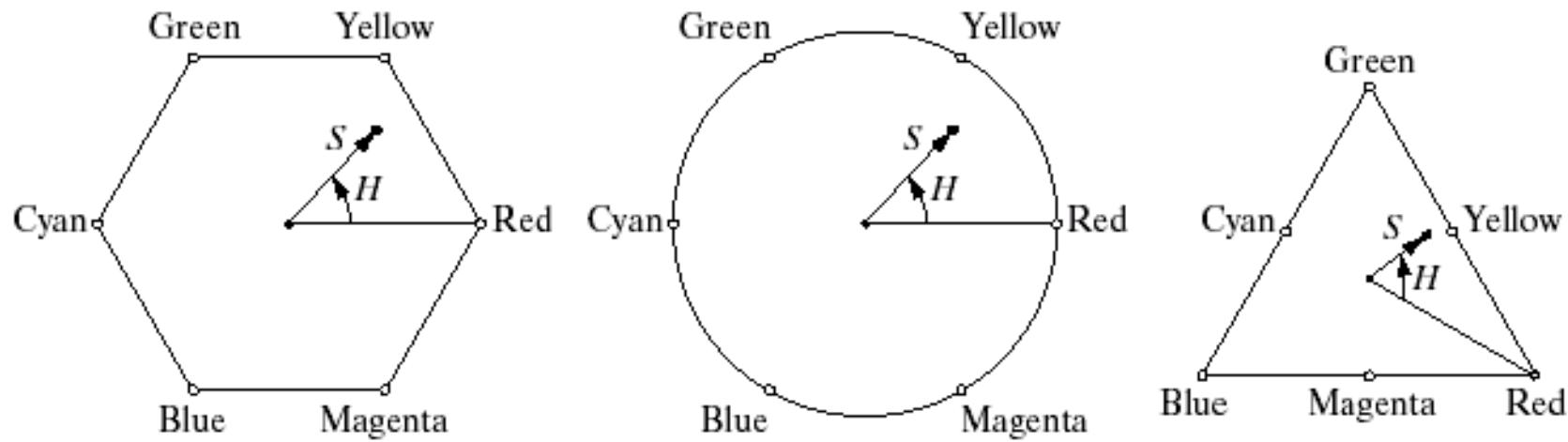
To the right we see a hexagonal shape and an arbitrary colour point

- The hue is determined by an angle from a reference point, usually red
- The saturation is the distance from the origin to the point
- The intensity is determined by how far up the vertical intensity axis this hexagonal plane sits (not apparent from this diagram)

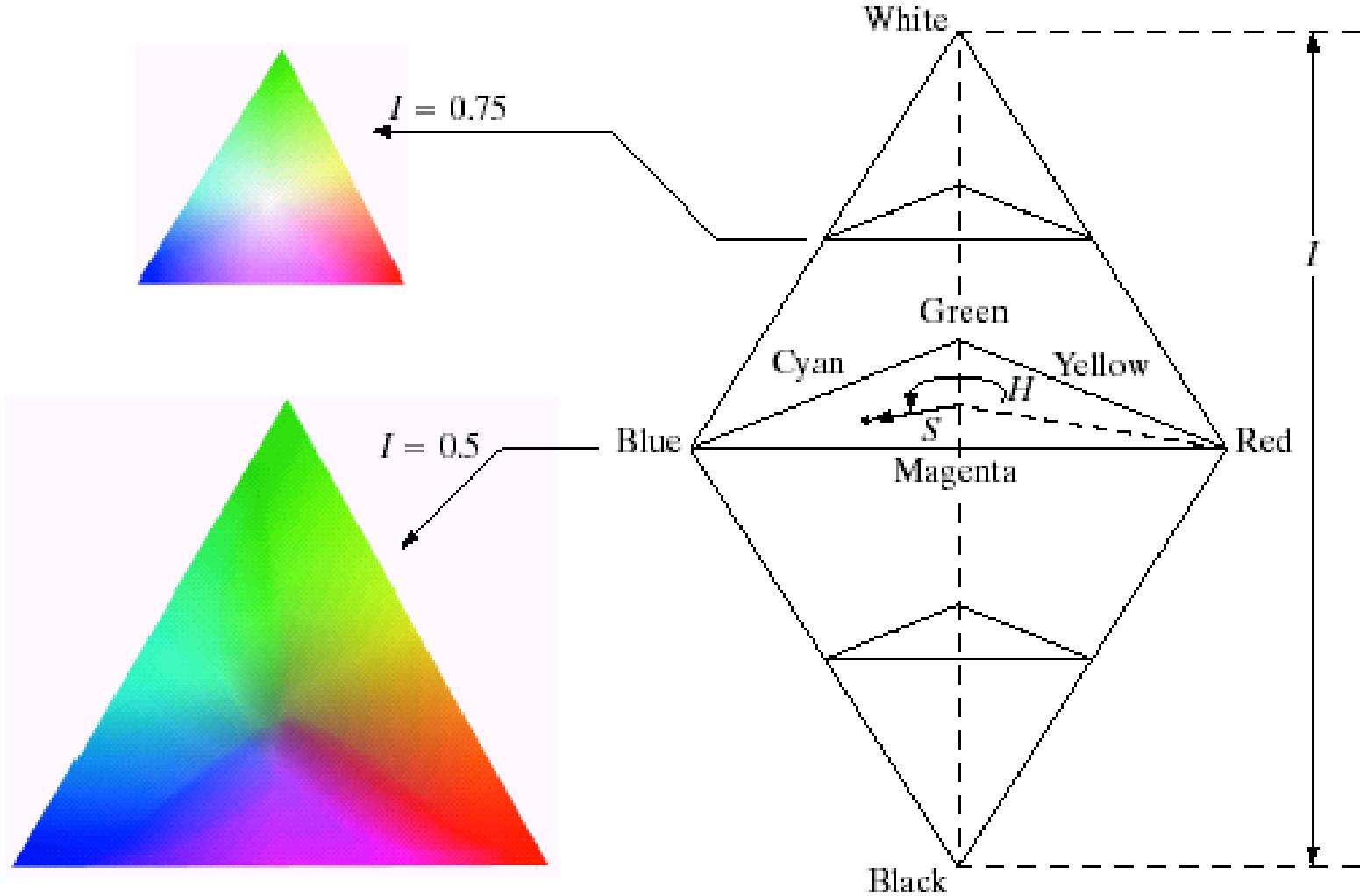


# The HSI Colour Model (cont...)

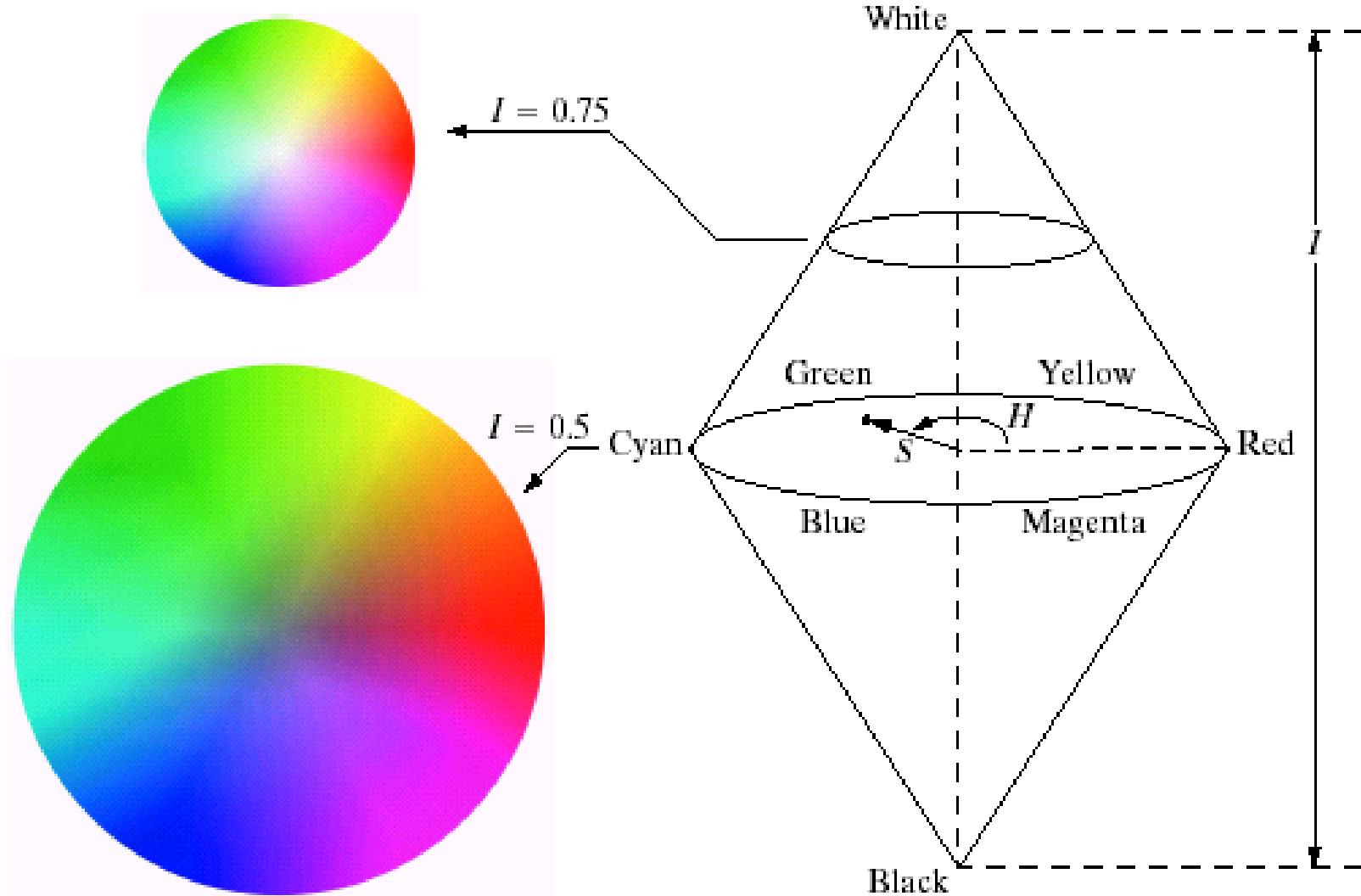
Because the only important things are the angle and the length of the saturation vector this plane is also often represented as a circle or a triangle



# HSI Model Examples



# HSI Model Examples



# Converting From RGB To HSI

Given a colour as R, G, and B its H, S, and I values are calculated as follows:

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \quad \theta = \cos^{-1} \left\{ \frac{\frac{1}{2} [(R-G) + (R-B)]}{[(R-G)^2 + (R-B)(G-B)]^{\frac{1}{2}}} \right\}$$

$$S = 1 - \frac{3}{(R+G+B)} [\min(R, G, B)] \quad I = \frac{1}{3}(R+G+B)$$

# Converting From HSI To RGB

Given a colour as H, S, and I it's R, G, and B values are calculated as follows:

- RG sector ( $0 \leq H < 120^\circ$  )

$$R = I \left[ 1 + \frac{S \cos H}{\cos(60 - H)} \right] \quad G = 3I - (R + B) \quad B = I(1 - S)$$

- GB sector ( $120^\circ \leq H < 240^\circ$  )

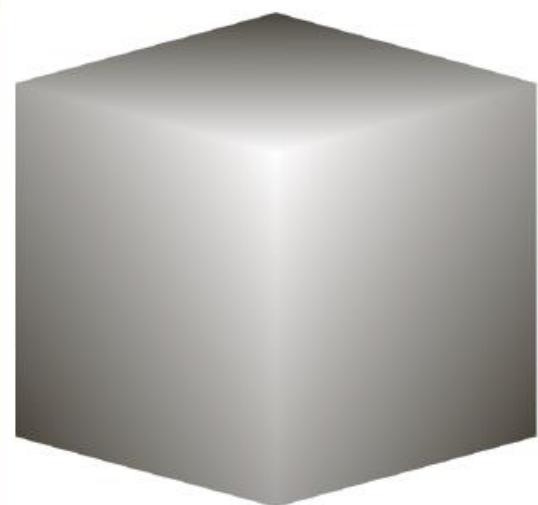
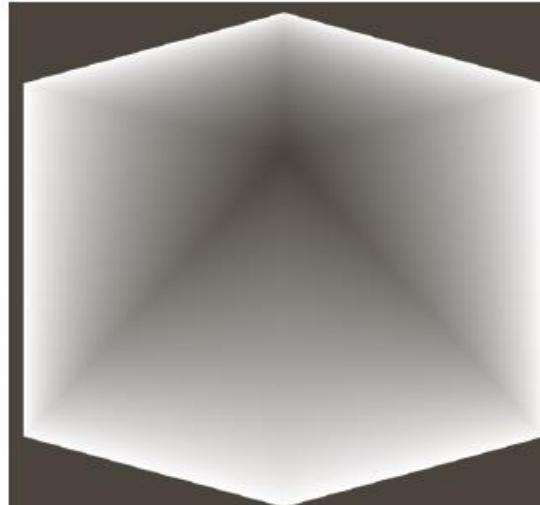
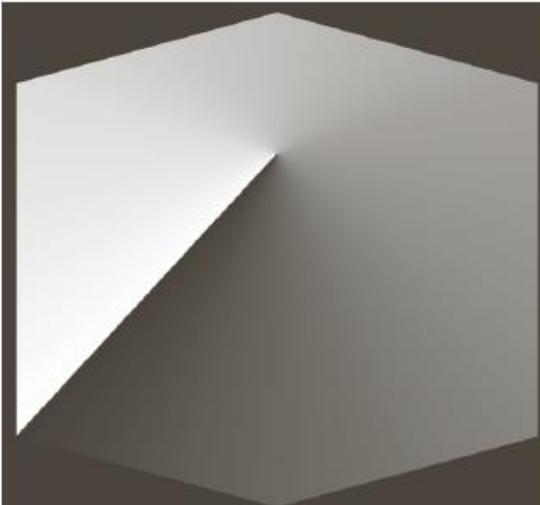
$$R = I(1 - S) \quad G = I \left[ 1 + \frac{S \cos(H - 120)}{\cos(H - 60)} \right] \quad B = 3I - (R + G)$$

# Converting From HSI To RGB (cont...)

- BR sector ( $240^\circ \leq H \leq 360^\circ$ )

$$R = 3I - (G + B) \quad G = I(1 - S) \quad B = I \left[ 1 + \frac{S \cos(H - 240)}{\cos(H - 180)} \right]$$

# HSI & RGB



H, S, and I Components of RGB Colour Cube

# Question

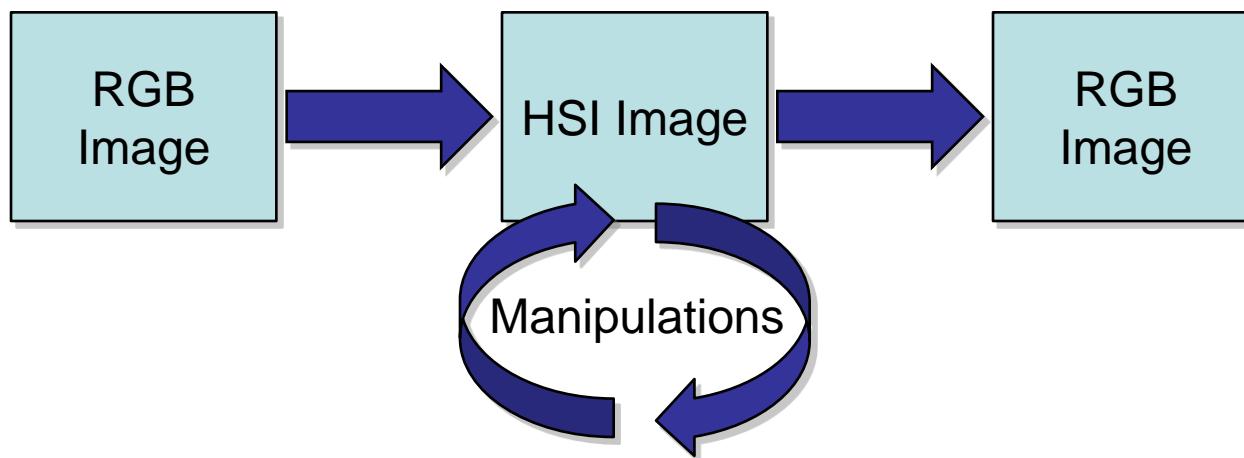
- Consider the following RGB image, in which the squares are fully saturated and each of the colors is at maximum intensity. An HSI image is generated from this image. Describe the appearance of each HSI component image.



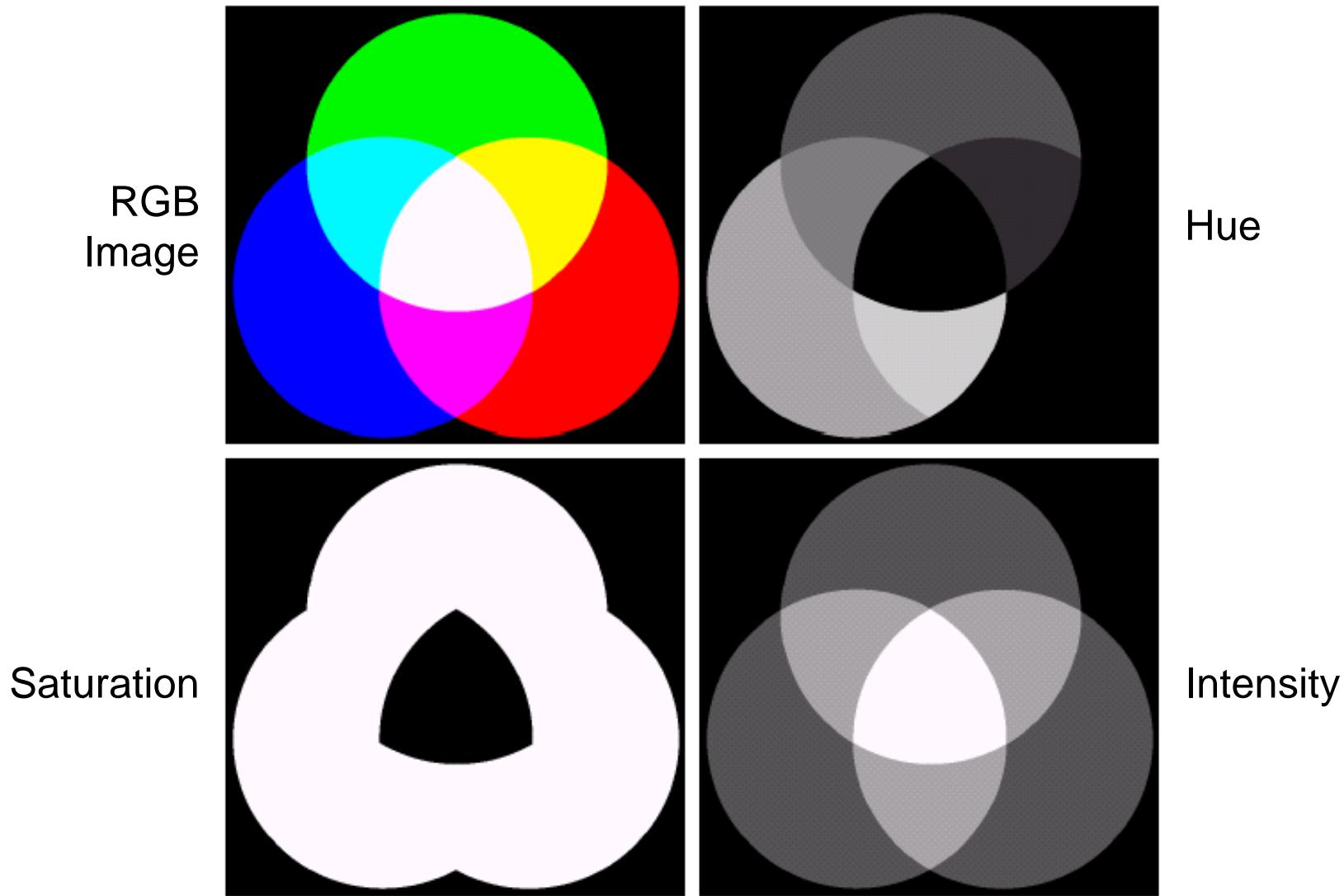
# Manipulating Images In The HSI Model

In order to manipulate an image under the HSI model we:

- First convert it from RGB to HSI
- Perform our manipulations under HSI
- Finally convert the image back from HSI to RGB

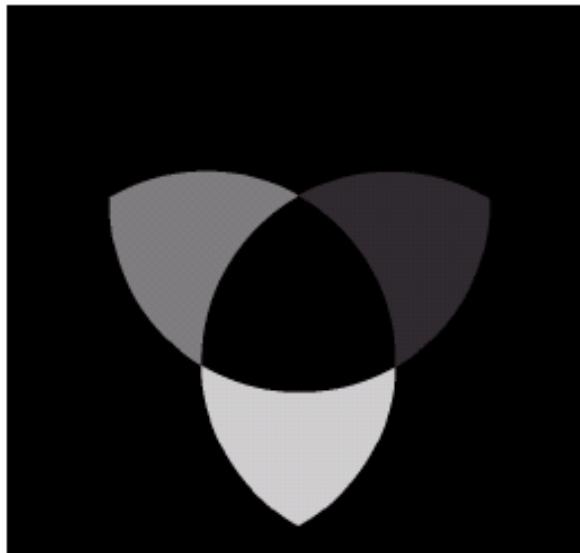


# RGB → HSI → RGB

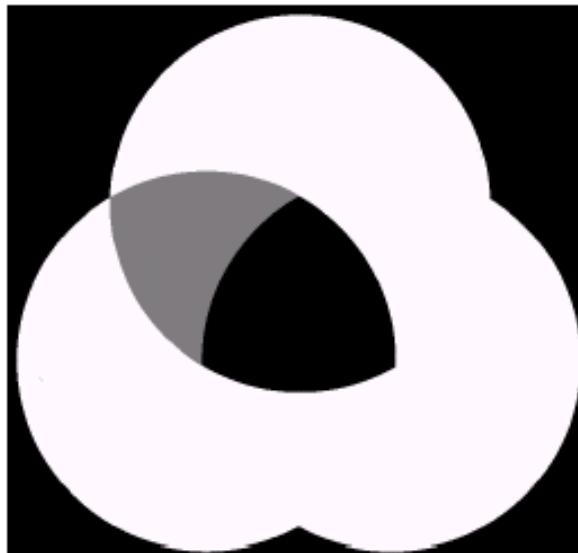


# RGB → HSI → RGB (cont...)

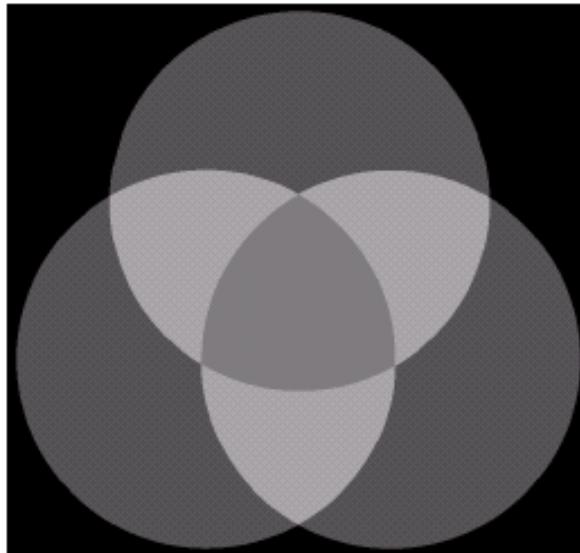
Hue



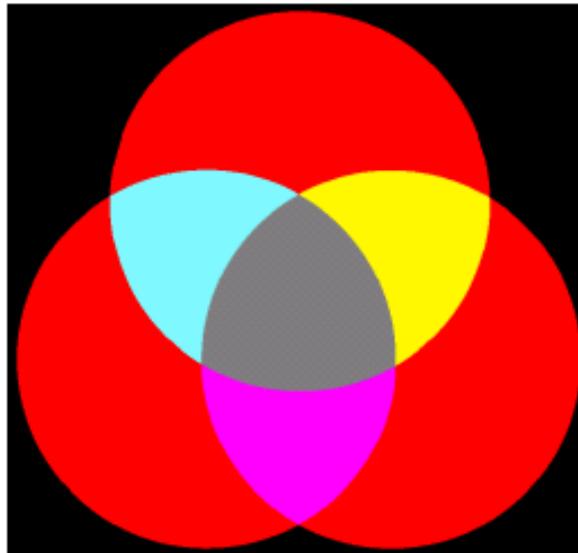
Saturation



Intensity



RGB Image



# Color image processing

- **Color image processing is divide into two major area:**
  - *Pseudo-Color* Processing
  - *Full-Color* Processing

# Pseudo-color image processing: Motivation



# Pseudo-color image processing: Motivation

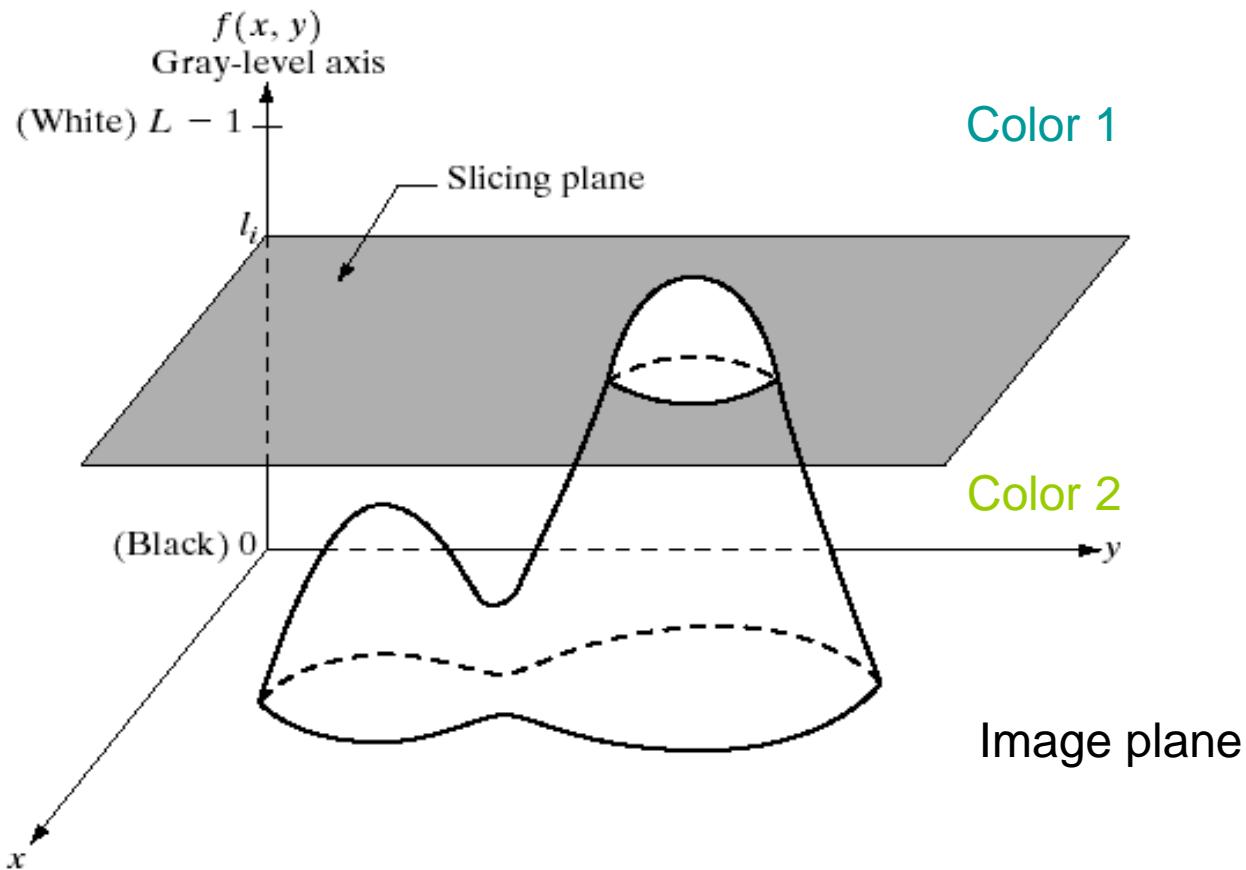


# Pseudo-color image processing

- Assign colors to gray values based on a specified criterion
  - For human visualization and interpretation of gray-scale events
- 
- Intensity slicing
  - Gray level to color transformations

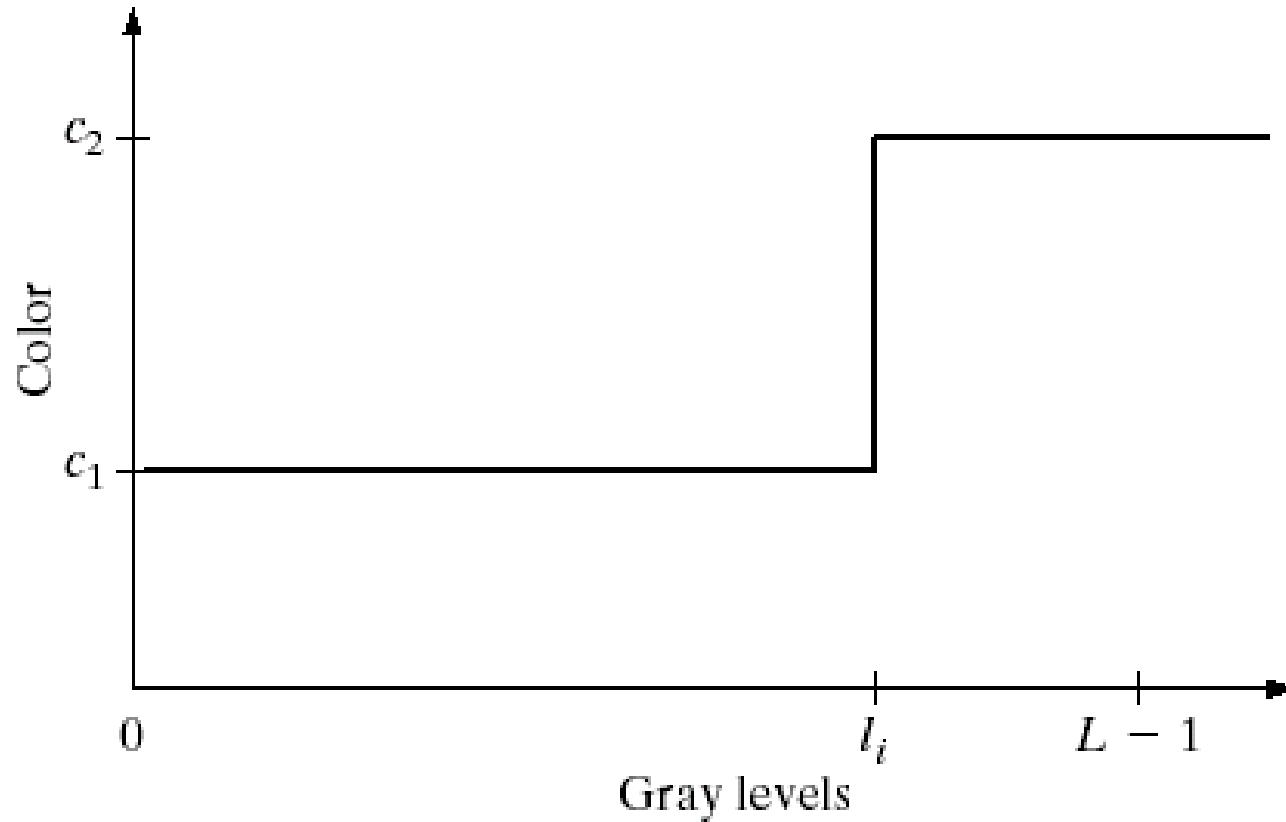
# Intensity slicing

- 3-D view of intensity image



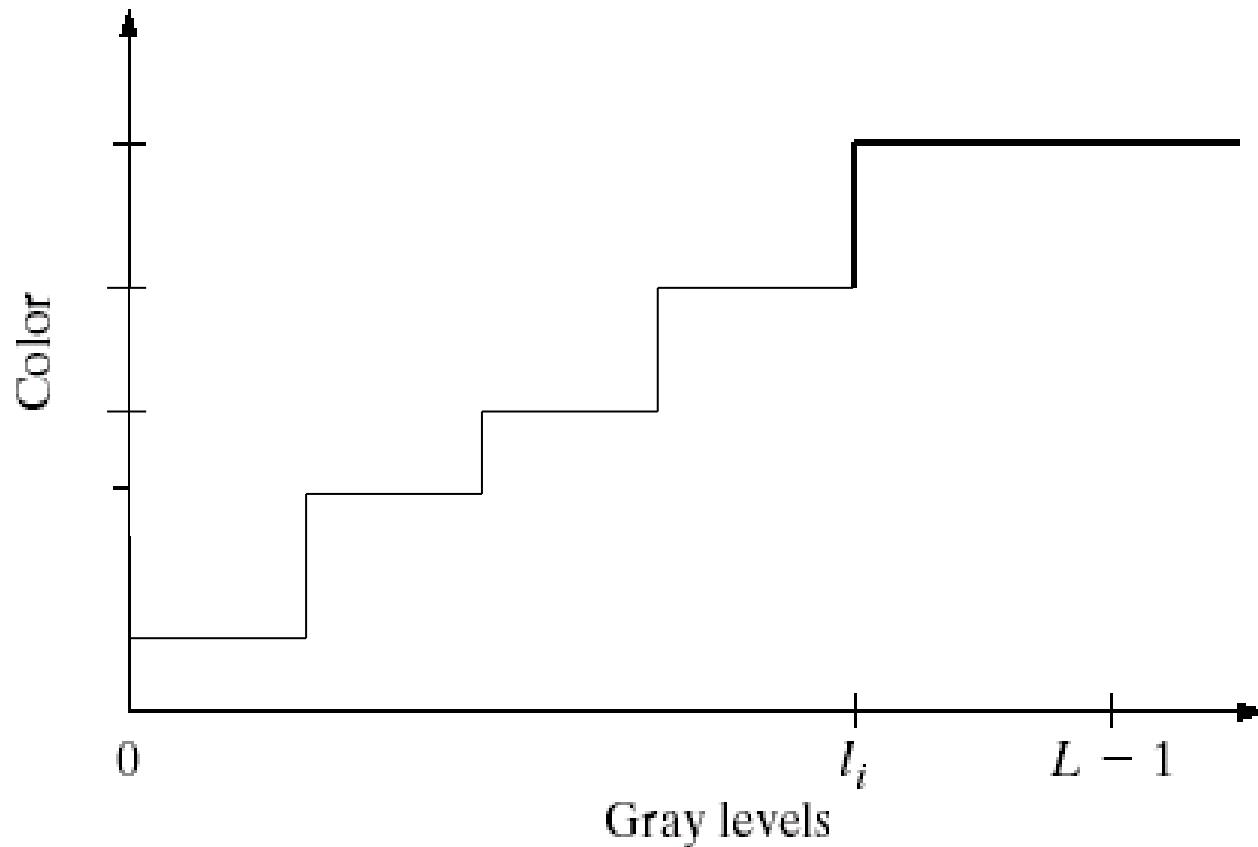
# Intensity slicing (cont.)

- Alternative representation of intensity slicing

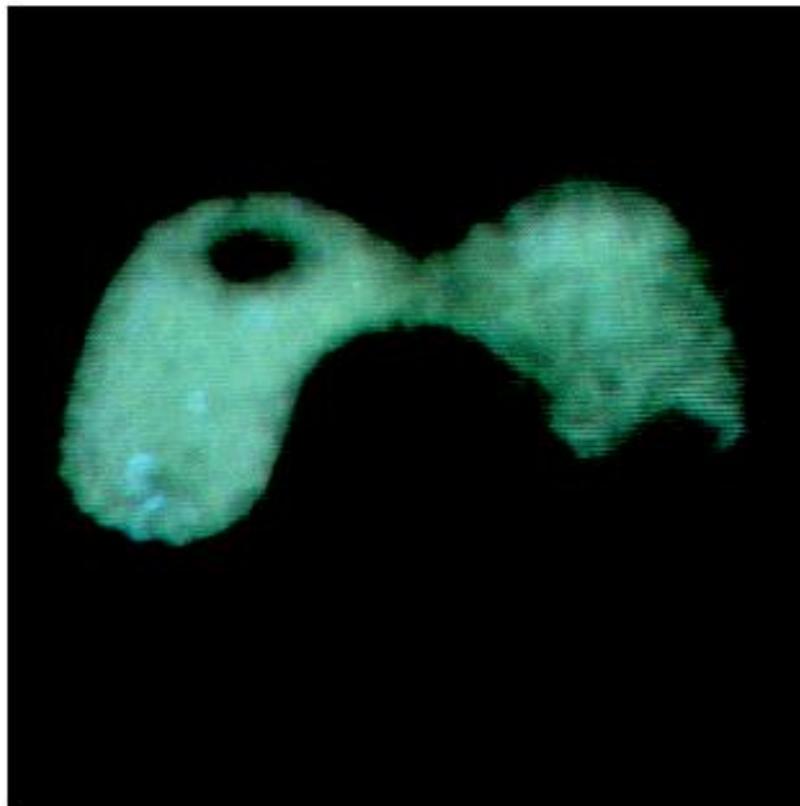


# Intensity slicing (cont.)

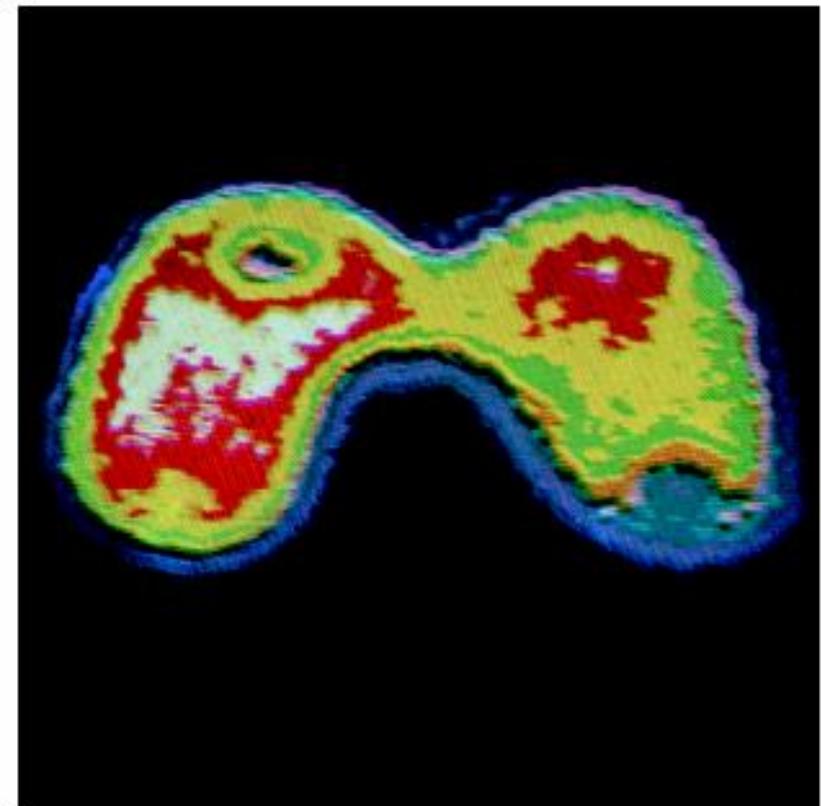
- More slicing plane, more colors



# Example Applications



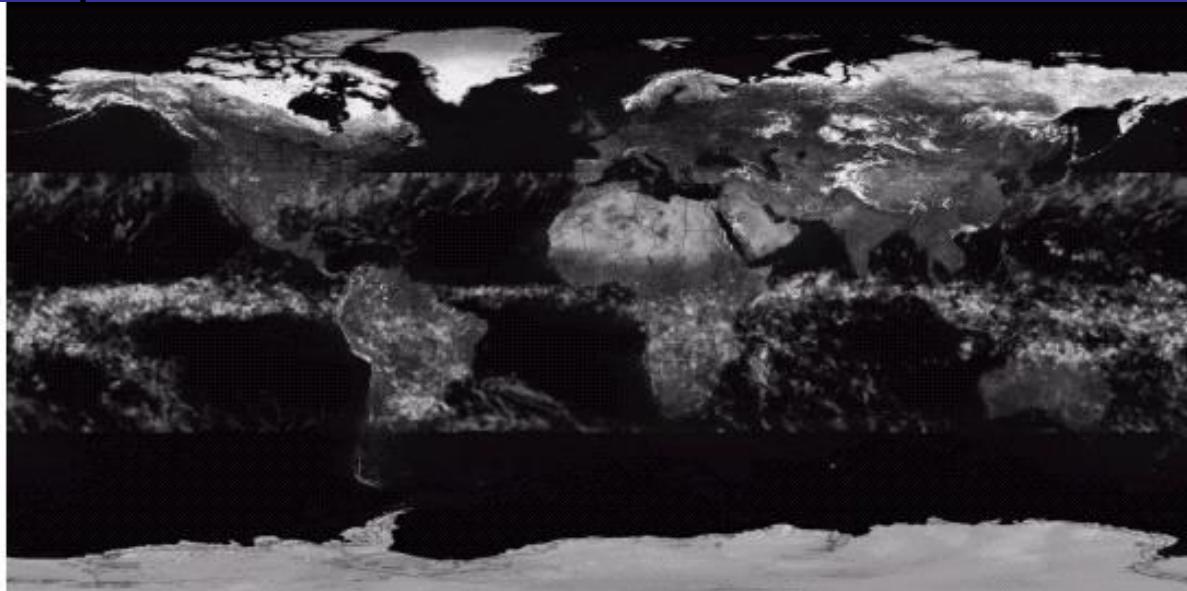
Radiation test pattern



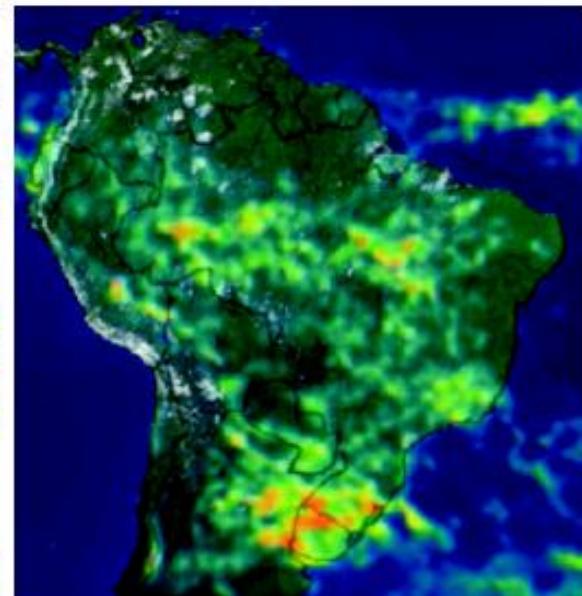
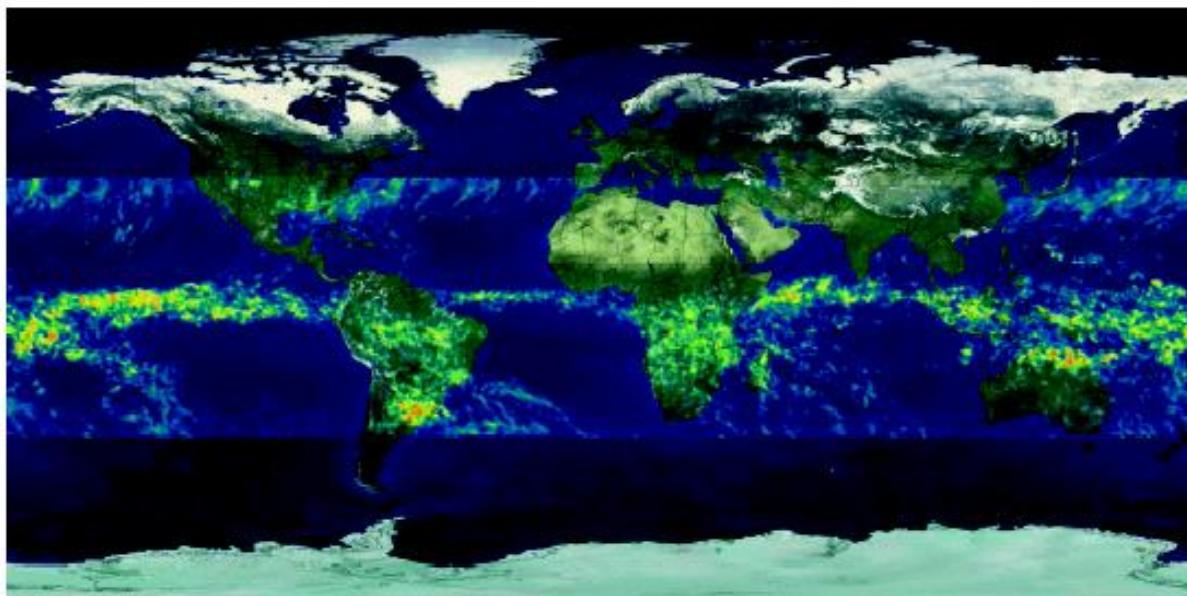
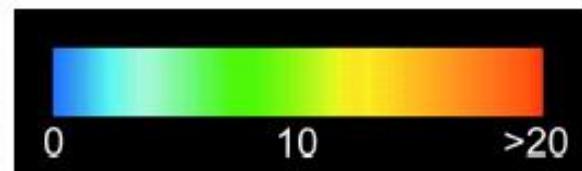
8 color regions

\* See the gradual gray-level changes

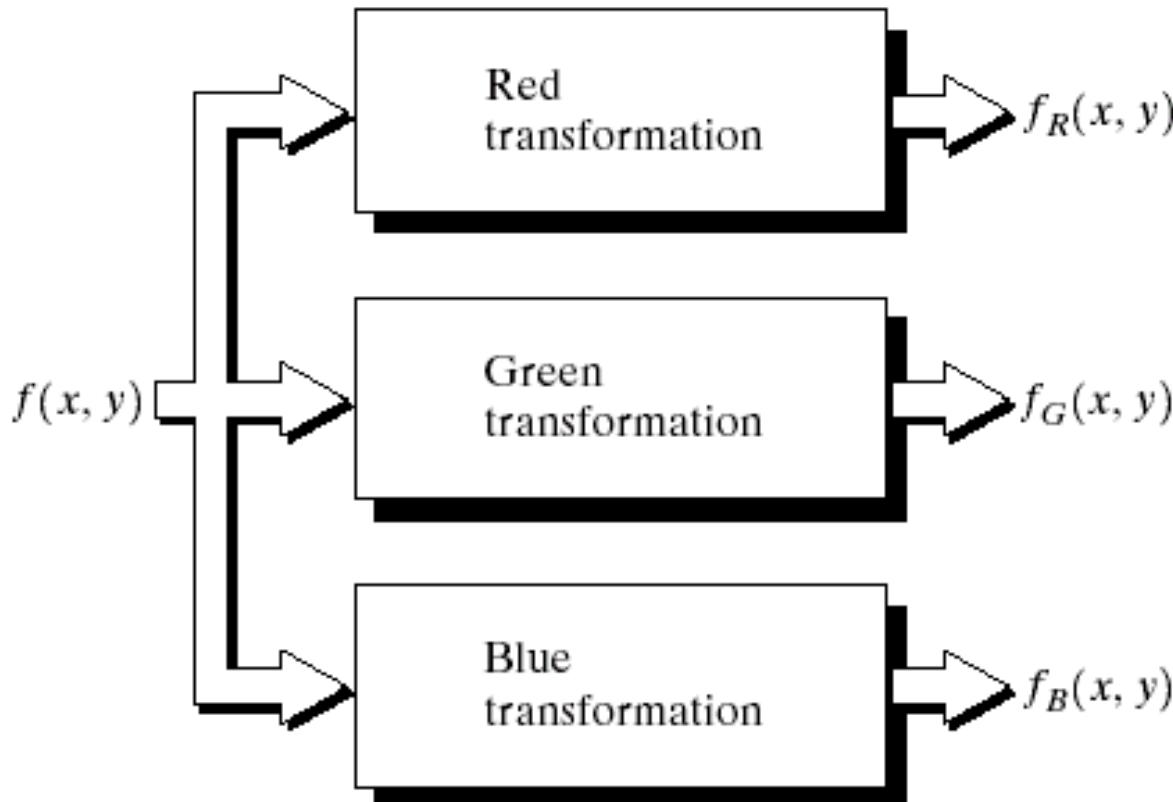
# Example Applications



Rainfall statistics



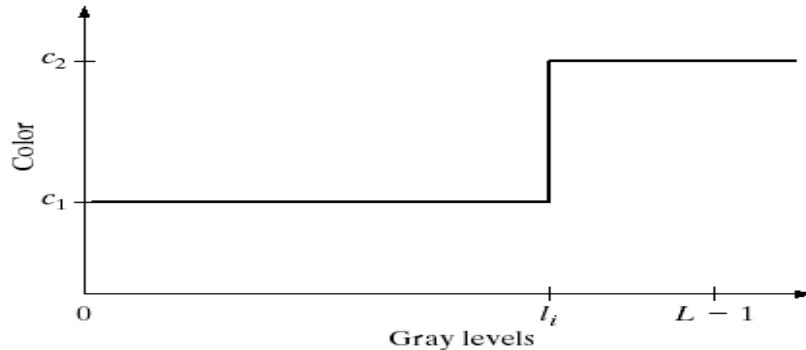
# Gray level to color transformation



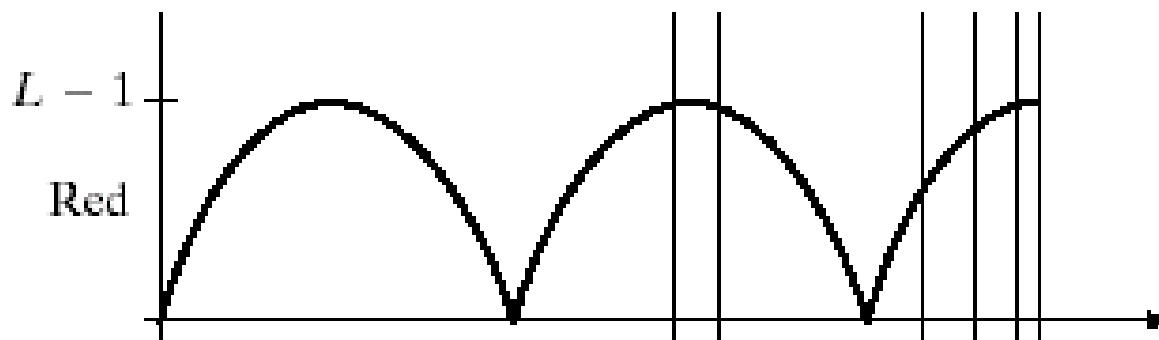
**FIGURE 6.23** Functional block diagram for pseudocolor image processing.  $f_R$ ,  $f_G$ , and  $f_B$  are fed into the corresponding red, green, and blue inputs of an RGB color monitor.

# Gray level to color transformation

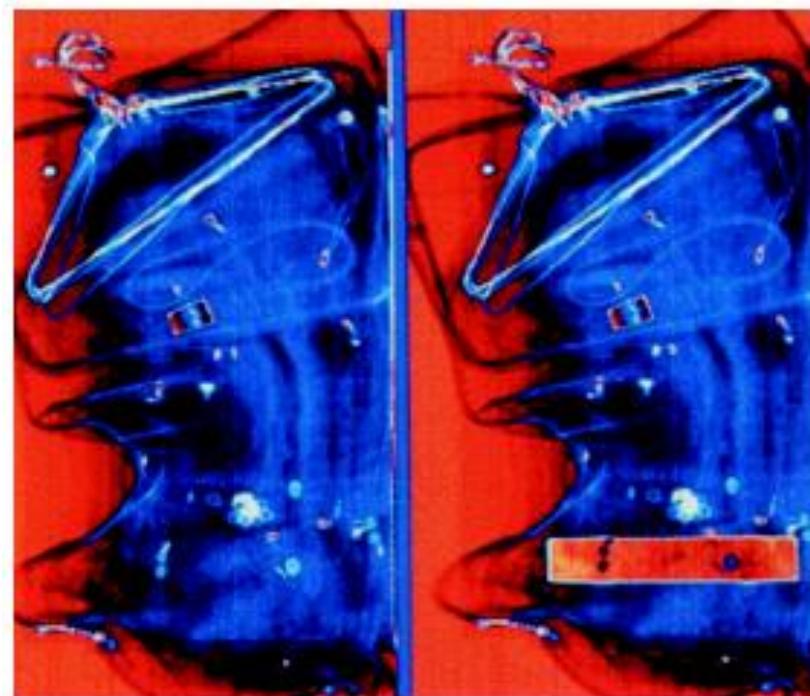
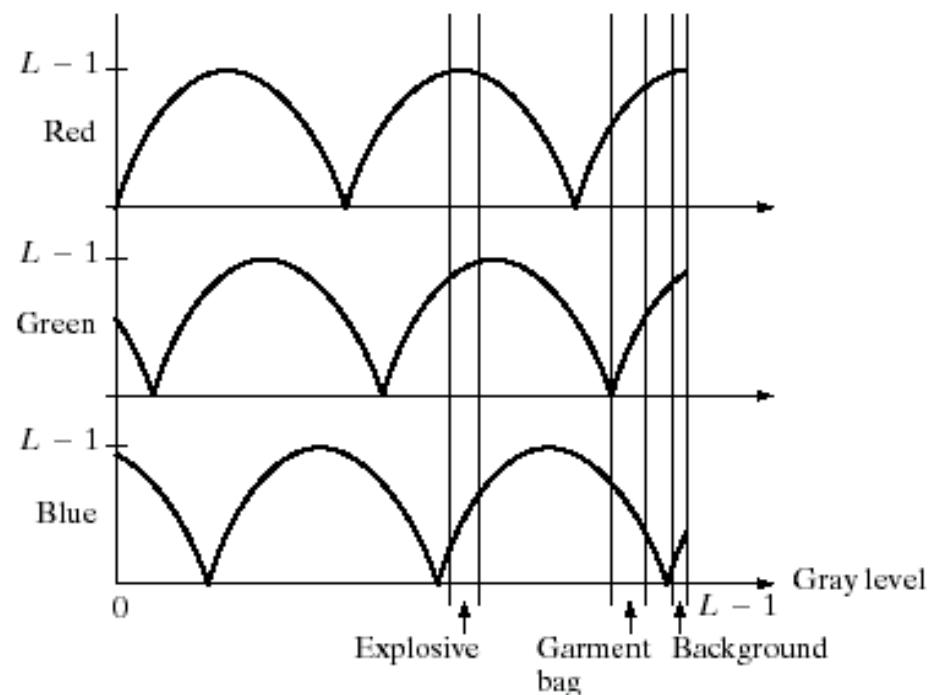
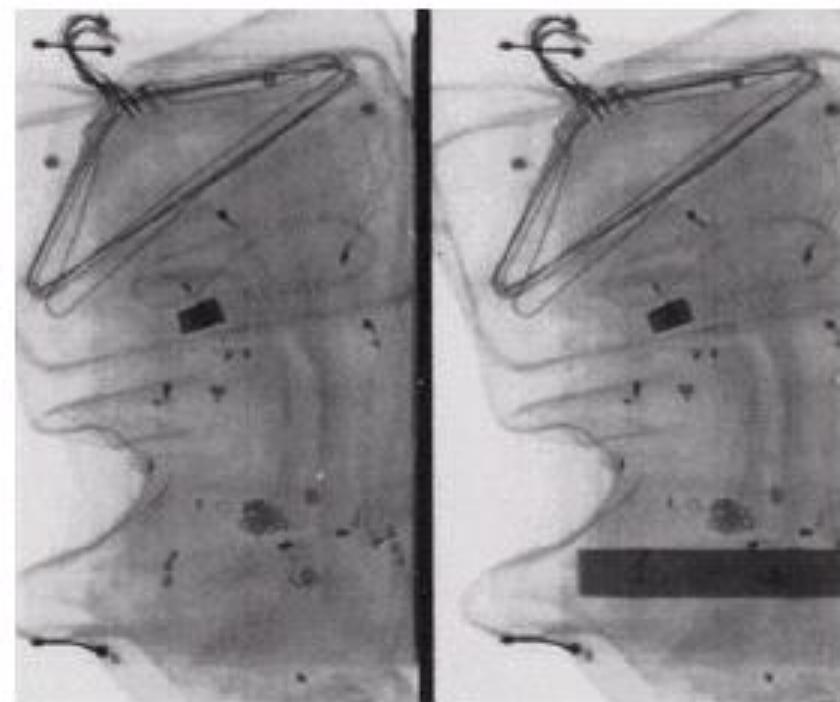
- Intensity slicing: piecewise linear transformation



- General Gray level to color transformation

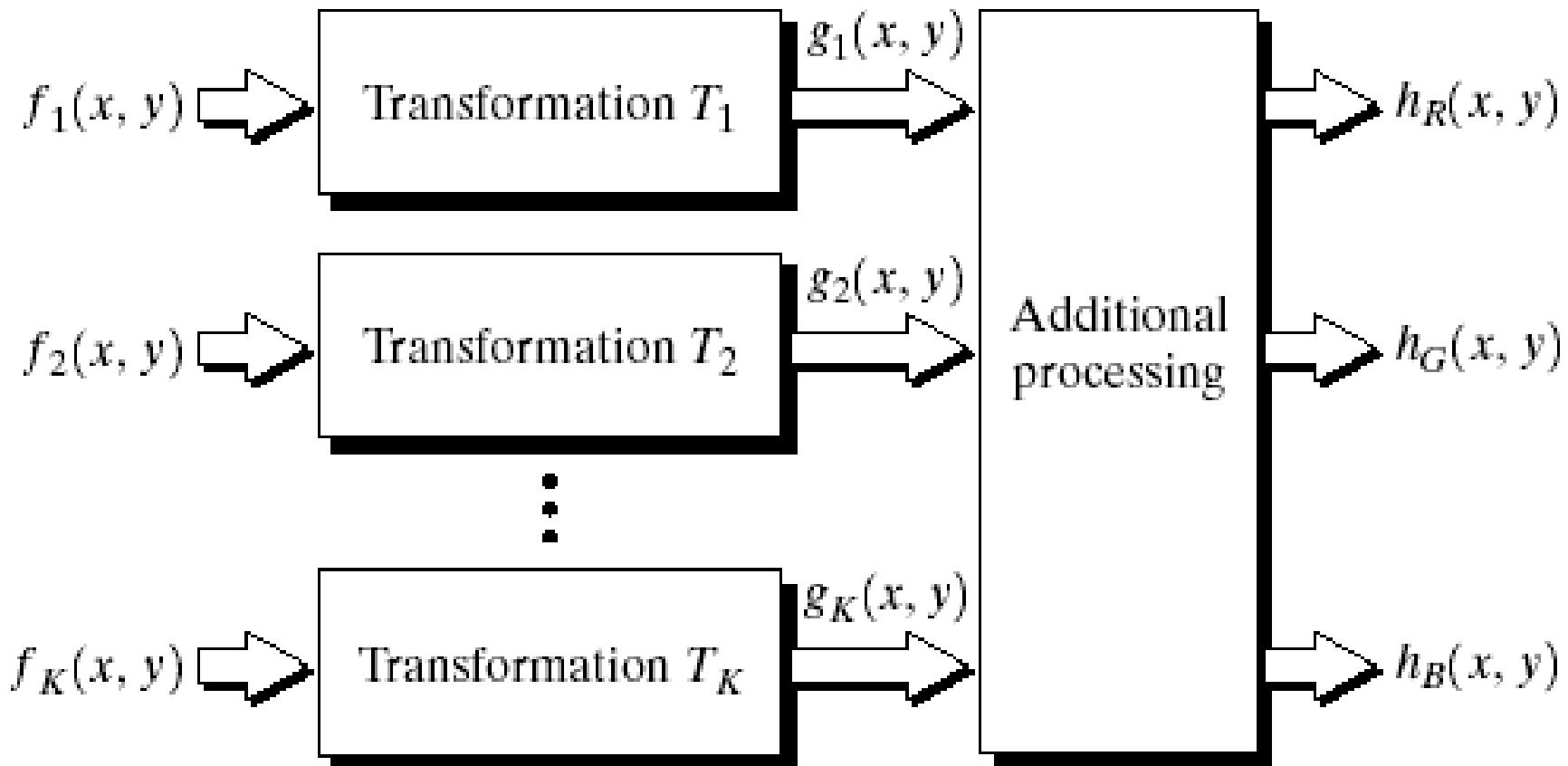


# Application



# Combine several monochrome images

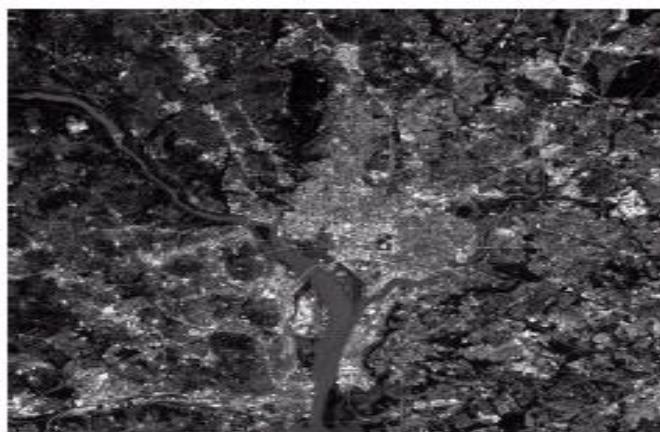
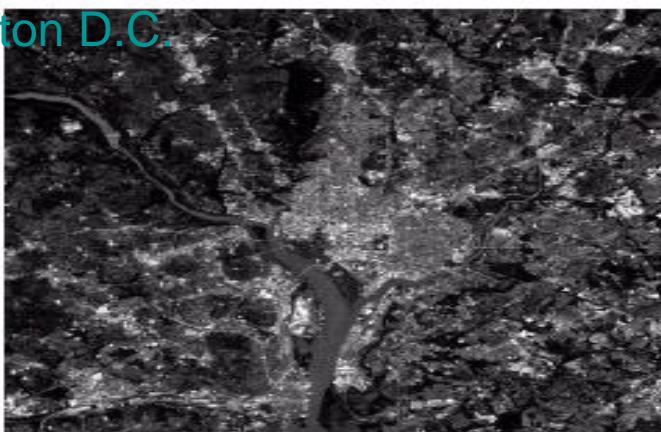
Example: multi-spectral images



Washington D.C.

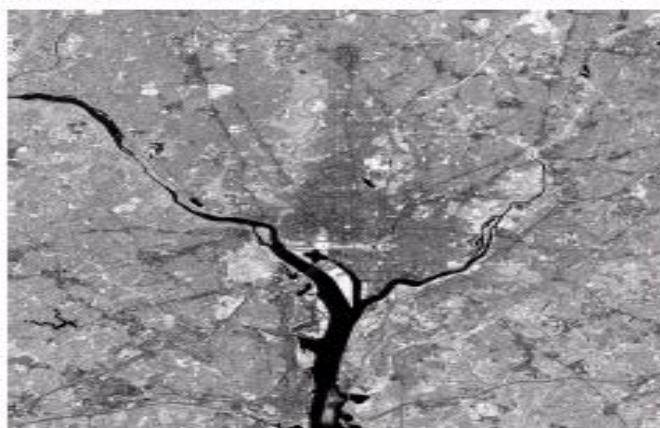
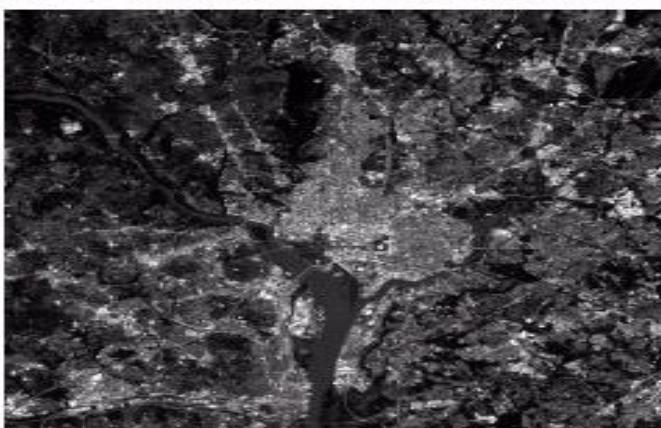
48

R

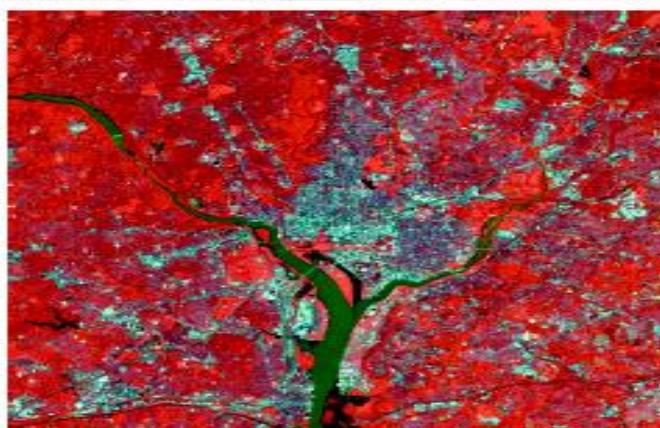


G

B



Near  
Infrared  
(sensitive  
to biomass)



R+G+B

near-infrared+G+B

# Full Color Image Processing

- A pixel at  $(x, y)$  is a **vector** in the color space
  - RGB color space

$$\mathbf{c}(x, y) = \begin{bmatrix} R(x, y) \\ G(x, y) \\ B(x, y) \end{bmatrix}$$

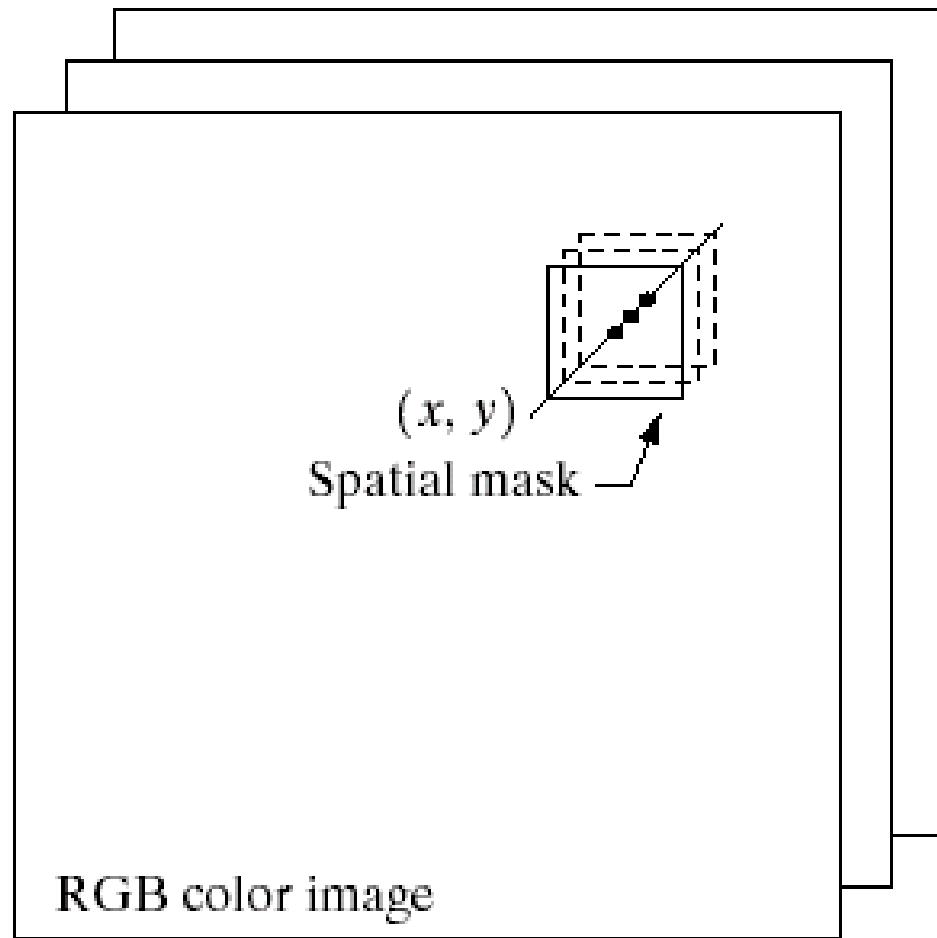
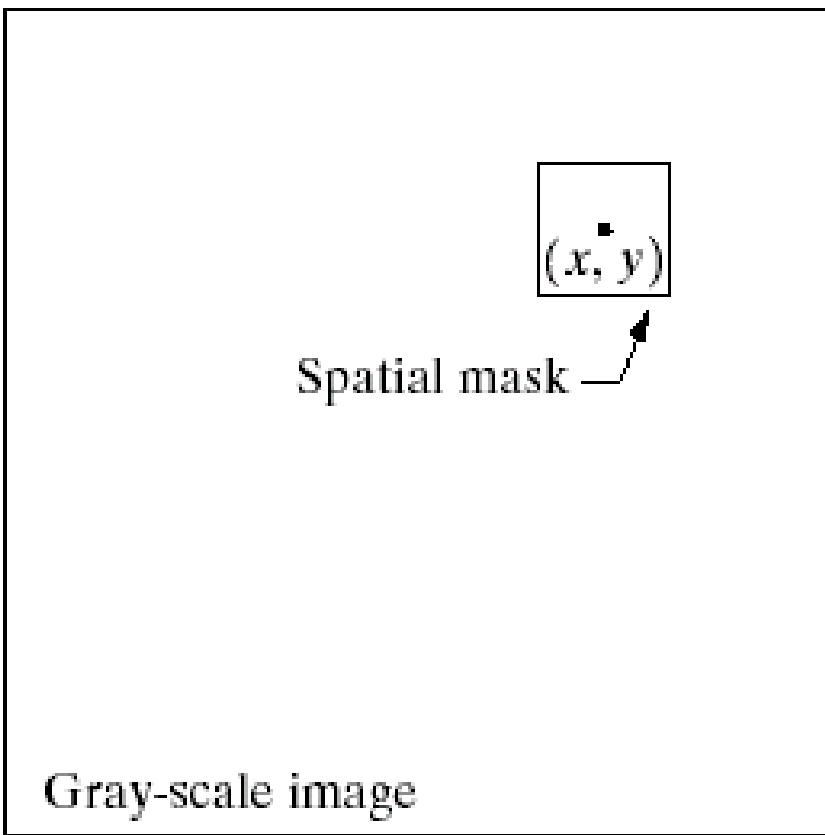
c.f. gray-scale image

$$f(x, y) = I(x, y)$$

# How to deal with color vector?

- 
- Per-color-component processing
    - Process each color component
  - Vector-based processing
    - Process the color vector of each pixel
  - When can the above methods be equivalent?
    - Process can be applied to both scalars and vectors
    - Operation on each component of a vector must be independent of the other component

# Example: spatial mask



# Two spatial processing categories

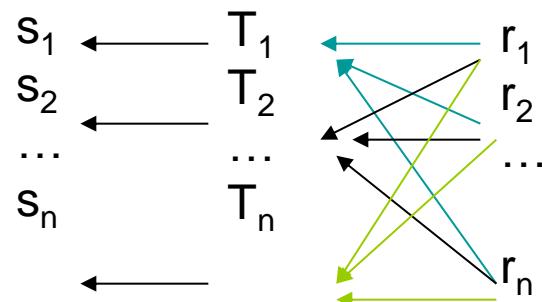
- Similar to gray scale processing studied before, we have two major categories
- **Pixel-wise** processing
- **Neighborhood** processing

# Color transformation

- Similar to gray scale transformation
  - $g(x,y) = T[f(x,y)]$
- Color transformation

$$s_i = T_i(r_1, r_2, \dots, r_n), \quad i = 1, 2, \dots, n$$

$g(x,y)$                                      $f(x,y)$

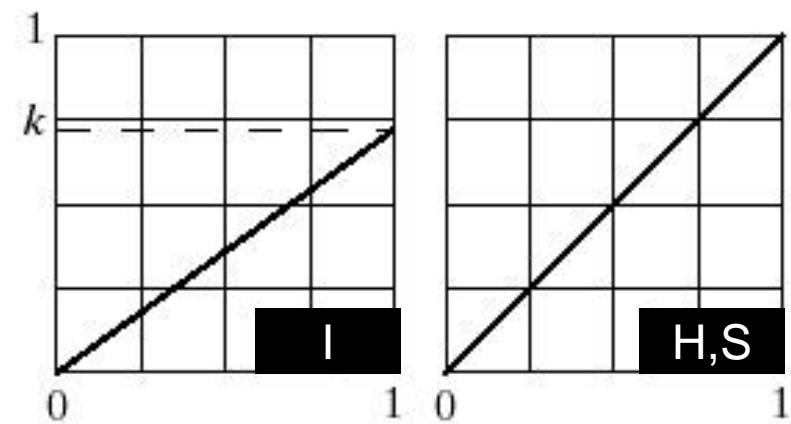
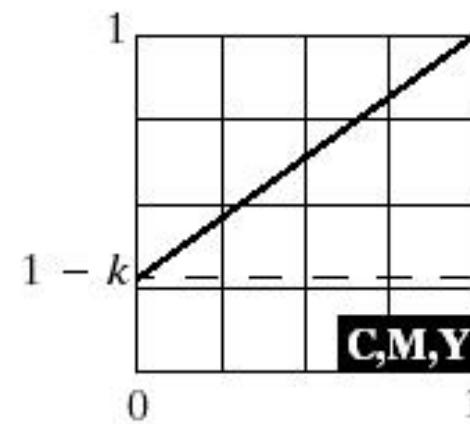
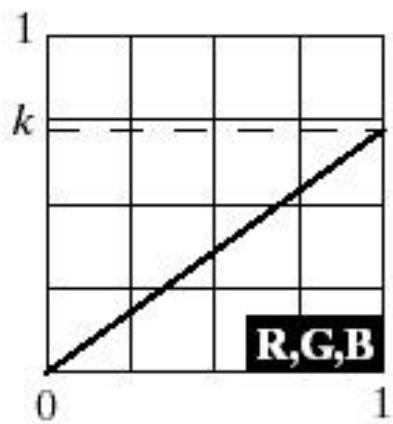


# Use which color model in color transformation?

- RGB  $\leftrightarrow$  HSI
- **Theoretically**, any transformation can be performed in any color model
- **Practically**, some operations are better suited to specific color model

# Example: modify intensity of a color image

- **Example:**  $g(x,y) = k f(x,y)$ ,  $0 < k < 1$
- **HSI color space**
  - Intensity:  $s_3 = k r_3$
  - Note: transform to HSI requires complex operations
- **RGB color space**
  - For each R,G,B component:  $s_i = k r_i$





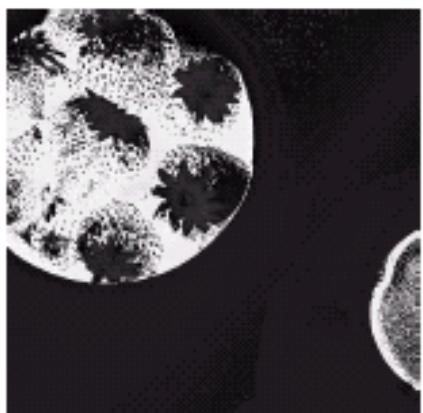
Red



Green



Blue



Hue



Saturation

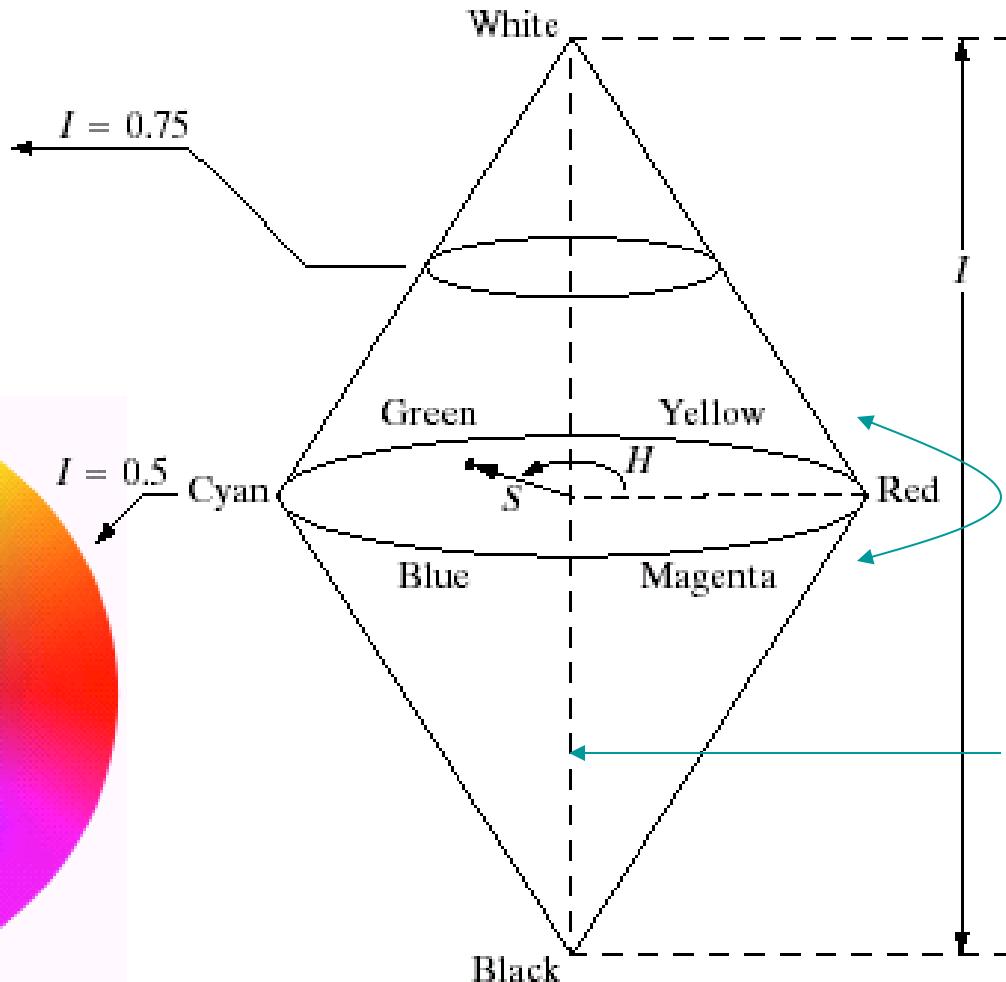
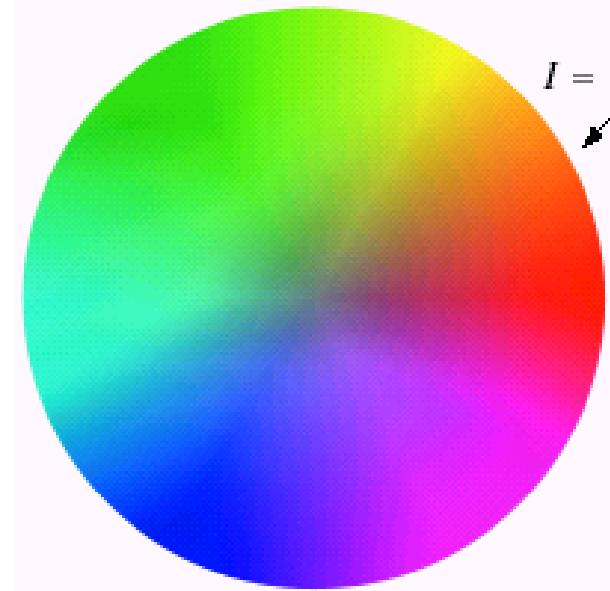
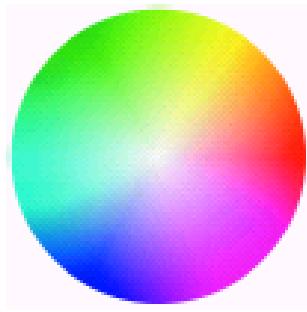


Intensity



Full color

# Problem of using Hue component

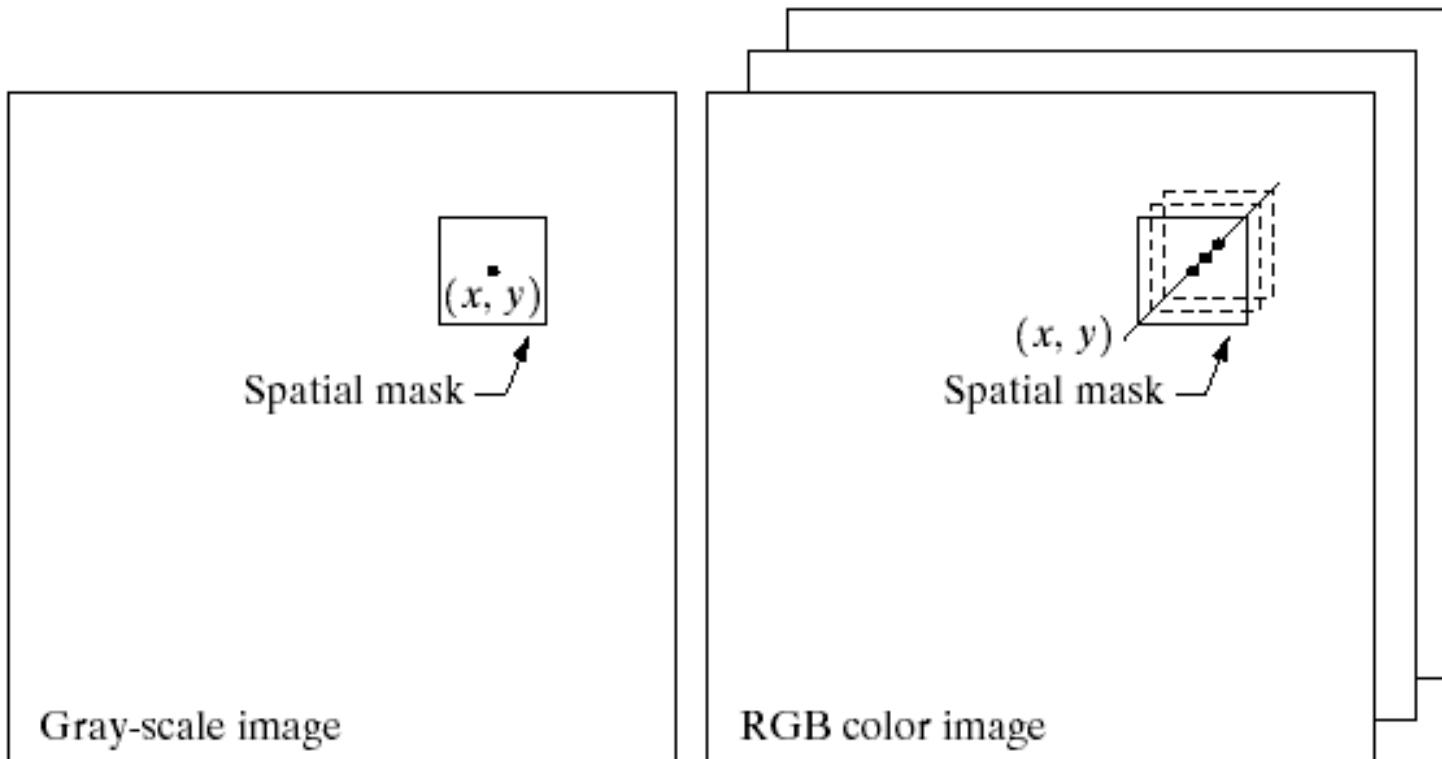


dis-continuous

Un-defined  
over gray  
axis

# Color image smoothing

- Neighborhood processing



# Color image smoothing: averaging mask

$$\bar{\mathbf{c}}(x, y) = \frac{1}{K} \sum_{(x, y) \in S_{xy}} \mathbf{c}(x, y)$$

vector processing

 Neighborhood  
Centered at (x,y)

$$\bar{\mathbf{c}}(x, y) = \begin{bmatrix} \frac{1}{K} \sum_{(x, y) \in S_{xy}} R(x, y) \\ \frac{1}{K} \sum_{(x, y) \in S_{xy}} G(x, y) \\ \frac{1}{K} \sum_{(x, y) \in S_{xy}} B(x, y) \end{bmatrix}$$

per-component processing

original



R



G



G



H



S



I

# Example: 5x5 smoothing mask

RGB model

Smooth I  
in HSI model

difference

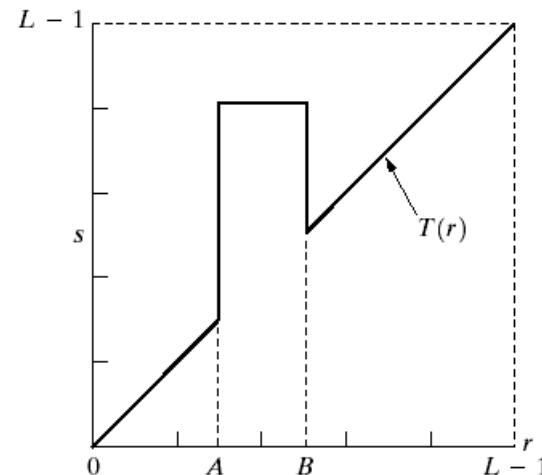
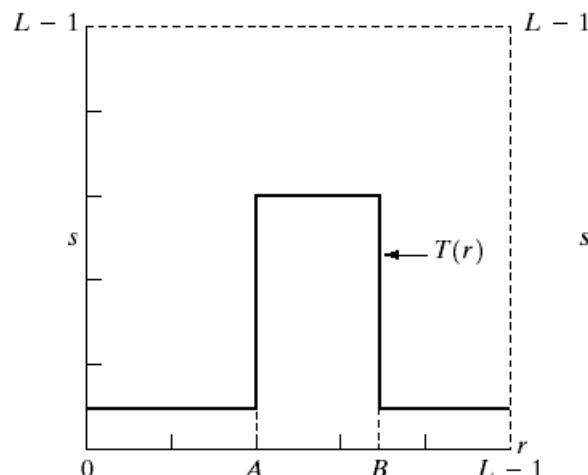


a b c

**FIGURE 6.40** Image smoothing with a  $5 \times 5$  averaging mask. (a) Result of processing each RGB component image. (b) Result of processing the intensity component of the HSI image and converting to RGB. (c) Difference between the two results.

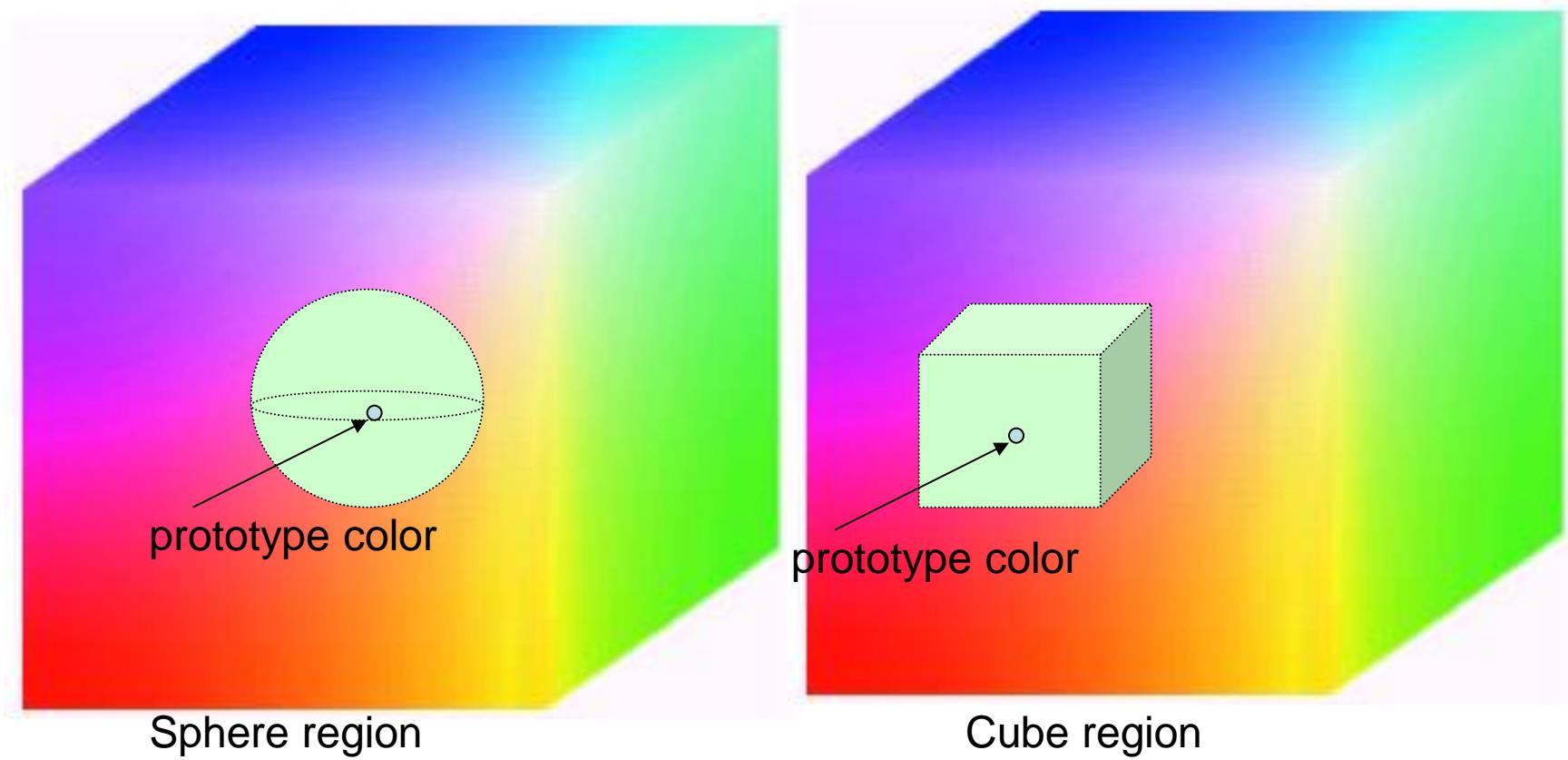
# Color slicing

- **Highlighting/Extracting a specific range of colors in an image**
- **Use the region defined by the colors as a mask for further processing**
- Recall the gray level slicing



# Implementation of color slicing

- How to take a **region of colors** of interest?



# Implementation of color slicing

1. Colors of interest are enclosed by ***cube*** (or ***hypercube*** for  $n>3$ )

$$s_i = \begin{cases} 0.5 & \text{if } \left[ |r_j - a_j| > \frac{W}{2} \right]_{\text{any } 1 \leq j \leq n} , \quad i = 1, 2, \dots, n \\ r_i & \text{otherwise} \end{cases}$$

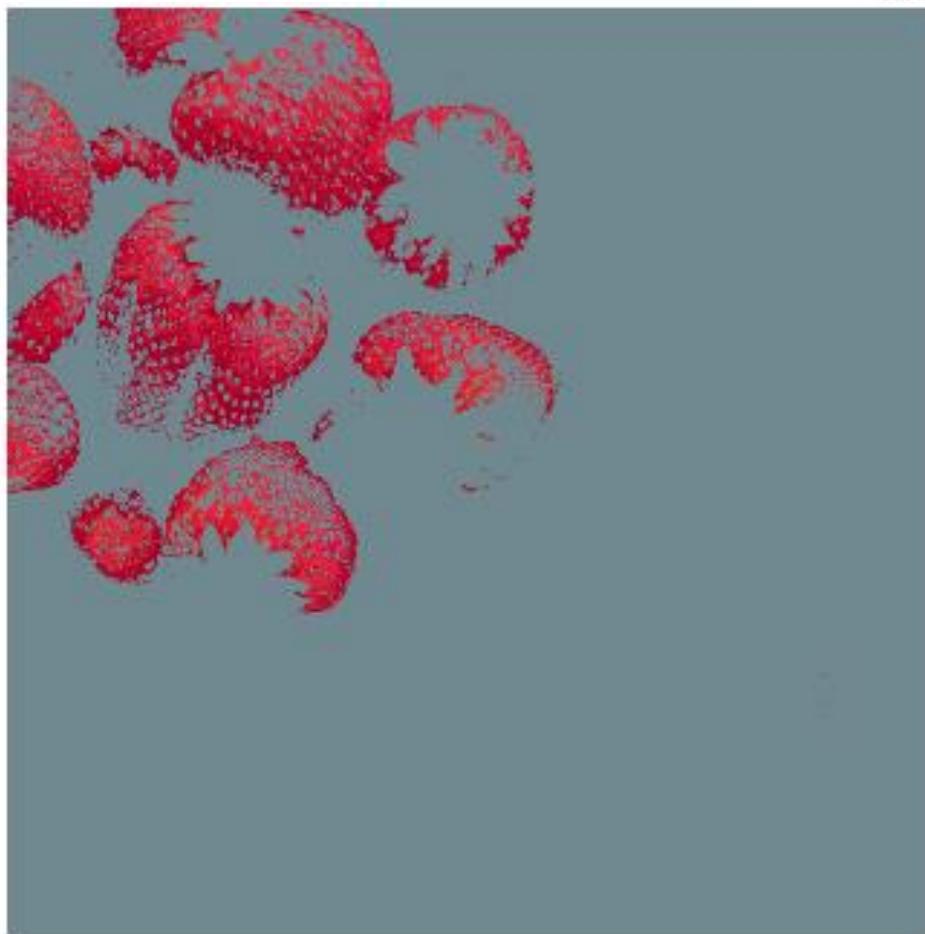
2. Colors of interest are enclosed by ***Sphere***

$$s_i = \begin{cases} 0.5 & \text{if } \sum_{j=1}^n (r_j - a_j)^2 > R_0^2 , \quad i = 1, 2, \dots, n \\ r_i & \text{otherwise} \end{cases}$$

# Example



Full color

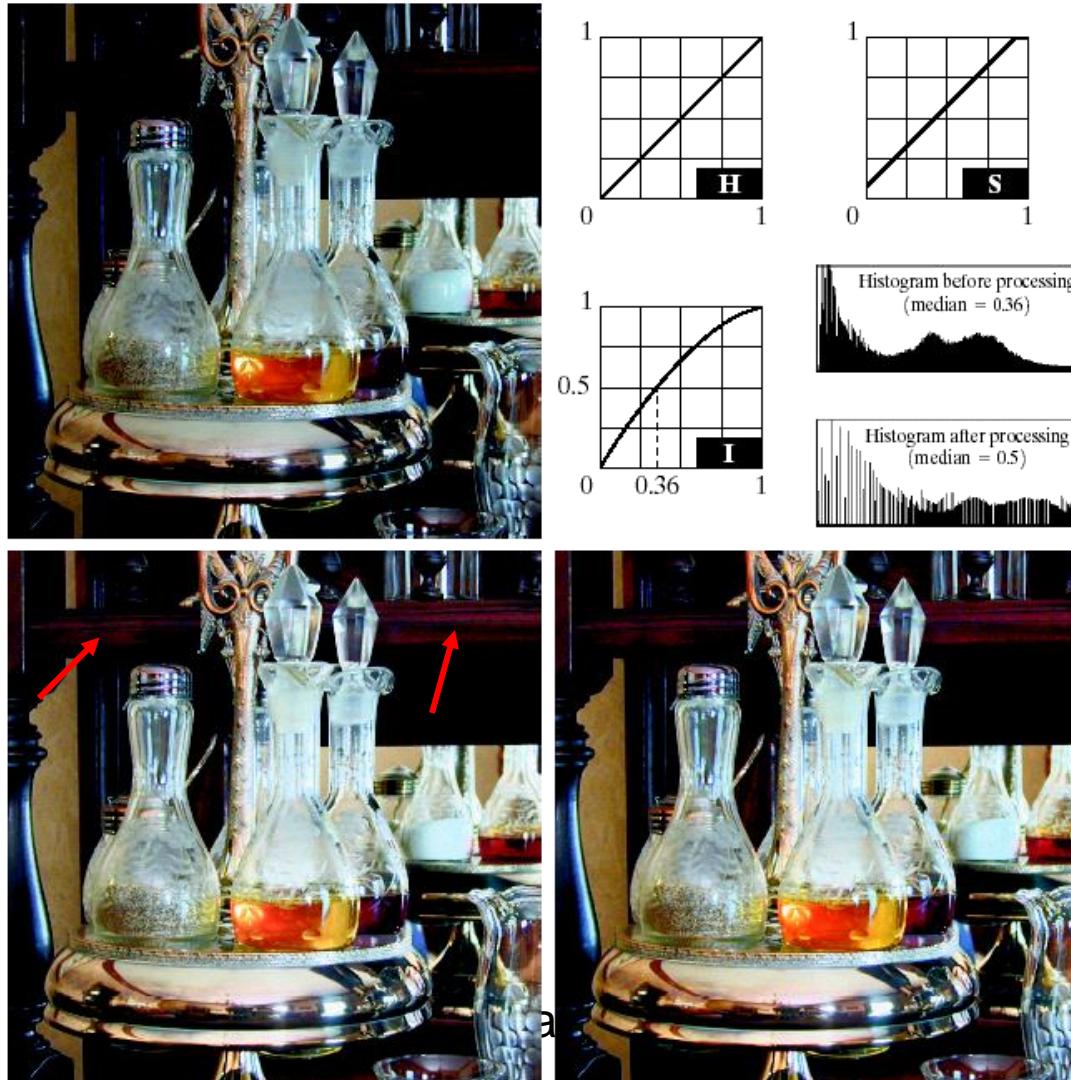


cube



sphere

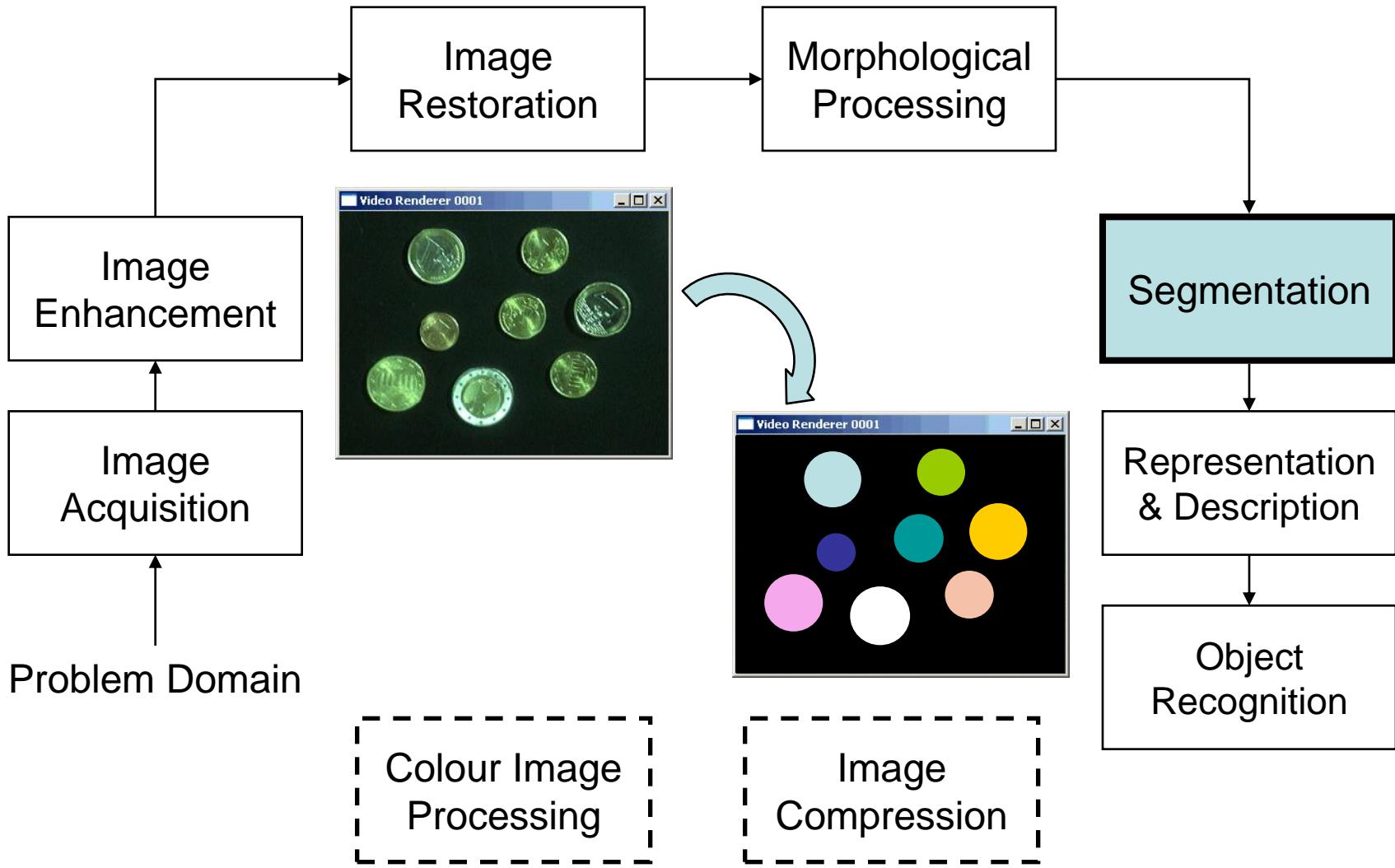
# Histogram Processing



# Image and Video Processing

Segmentation

# Course Outline



# What is segmentation?

- Segmentation is the process of dividing an image into separate/non-overlapping regions.
- All of the pixels in an image must belong to some region or another.
- A region might be an object, a part of an object or the background
- A region can be specified by either defining the pixels that constitute it, or the pixels that bound it.

# An Example

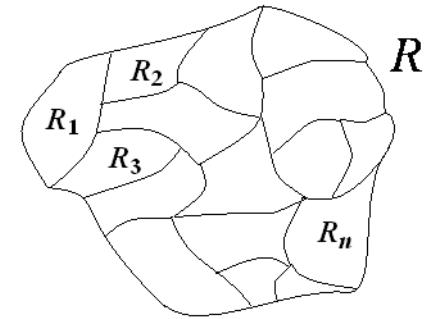
Identify the regions that you would segment this image into?



# Region definition

## Basic Formulation:

- (a)  $\bigcup_{i=1}^n R_i = R$
- (b)  $R_i$  is a connected region,  $i = 1, 2, \dots, n$
- (c)  $R_i \cap R_j = \emptyset$  for all i and j,  $i \neq j$
- (d)  $P(R_i) = \text{TRUE}$  for  $i = 1, 2, \dots, n$
- (e)  $P(R_i \cup R_j) = \text{FALSE}$  for  $i \neq j$



$P(R_i)$  is a homogeneity predicate property defined over the points in set  $R_i$

Ex.  $P(R_i) = \text{TRUE}$ , if all pixels in  $R_i$  have the 'same' gray level.

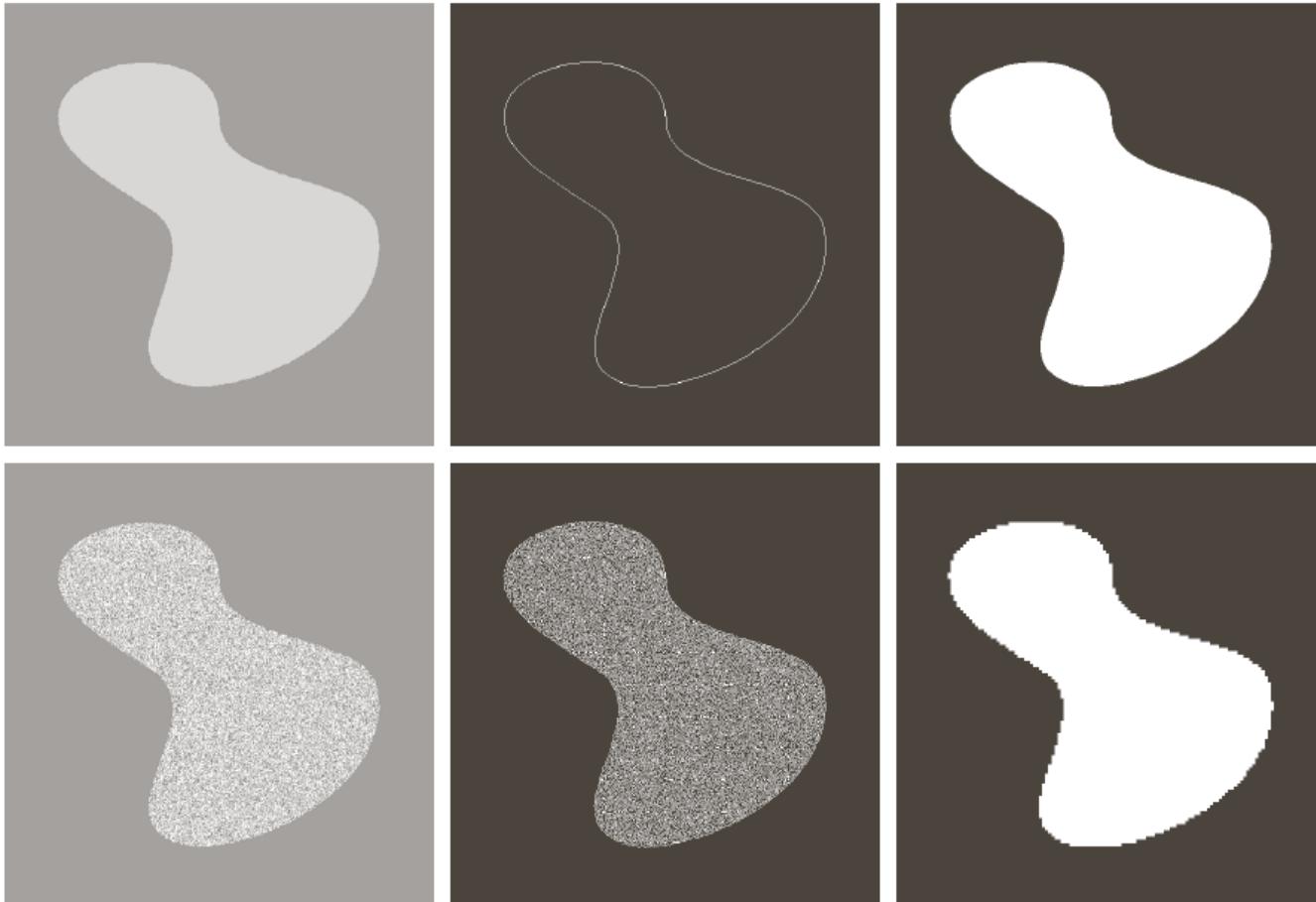
# Basic Principle

- Segmentation algorithms are often based on one of the following two basic properties of intensity values:
- **Similarity**: Partitioning an image into regions that are similar according to a set of predefined criteria.
- **Discontinuity**: Detecting boundaries of regions based on local discontinuity in intensity.

# Similarity vs Discontinuity

## Features

- intensity
- texture
- edge sharpness
- any other relevant feature(s)



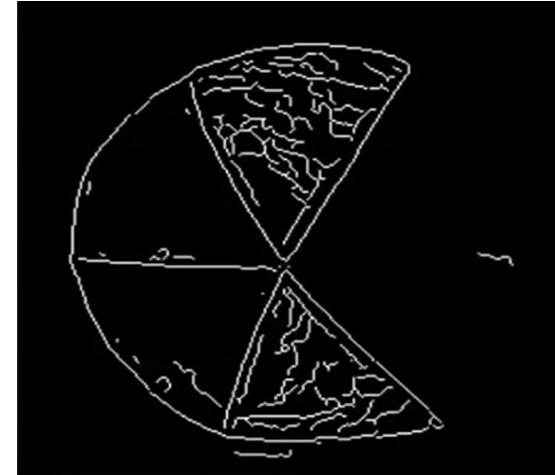
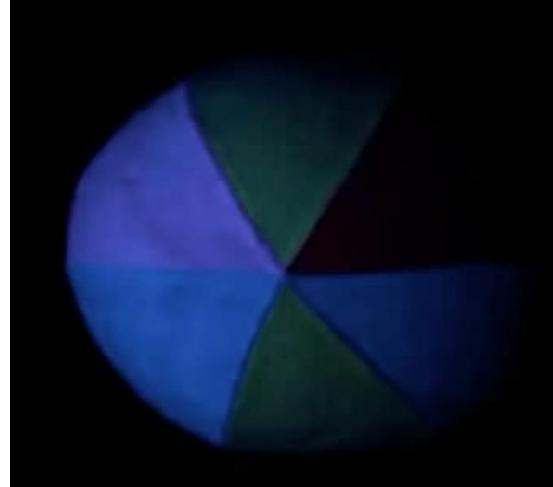
**FIGURE 10.1** (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large 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.

a b c  
d e f

# Image Segmentation Methods

- **Edge based segmentation [*Discontinuity*]**
  - Finding boundary between adjacent regions i.e. Detecting edges that separate regions from each other.
- **Threshold based segmentation [*Similarity*]**
  - Finding regions by grouping pixels with similar intensities i.e. Based on pixel intensities (shape of histogram is often used for automation).
- **Region based segmentation [*Similarity*]**
  - Finding regions directly using growing or splitting i.e. Grouping similar pixels (with e.g. region growing or merge & split).
- **Motion based segmentation [*Similarity*]**
  - Finding regions by comparing successive frames of a video sequence to identify regions that correspond to moving objects

# Segmentation Using Discontinuities



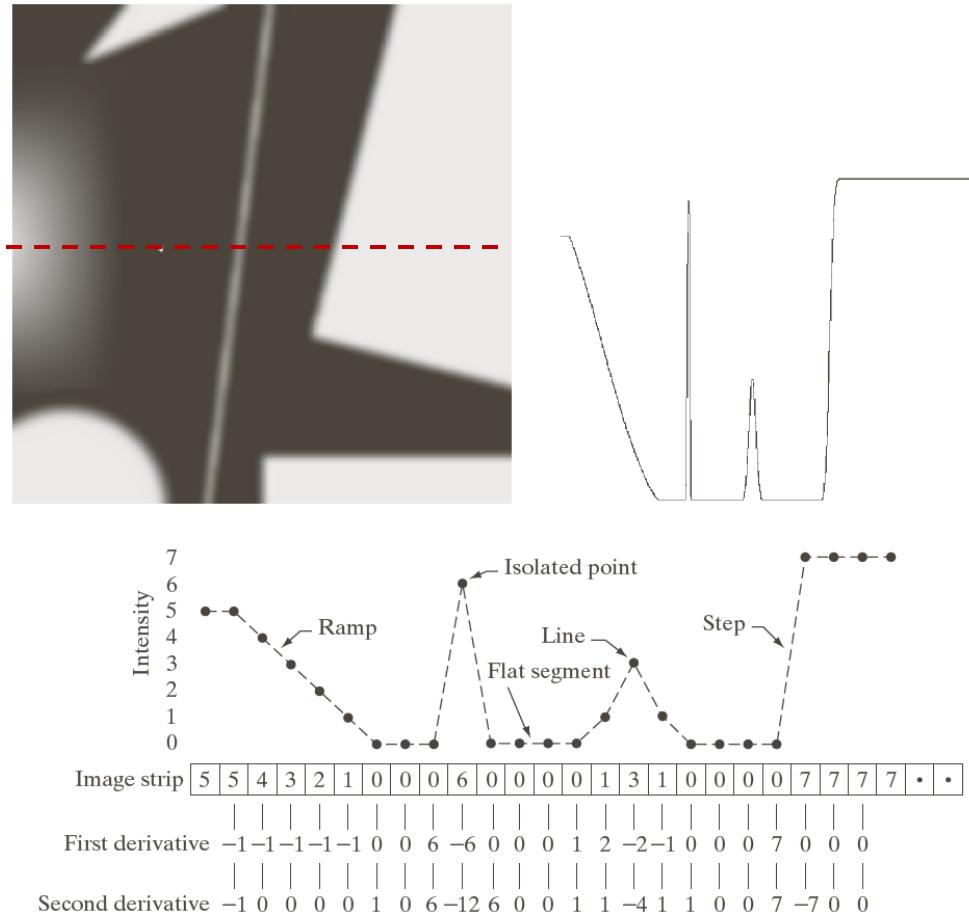
There are three basic types of grey level discontinuities in digital images:

- Points
- Lines
- Edges



# Point, line & edge detection

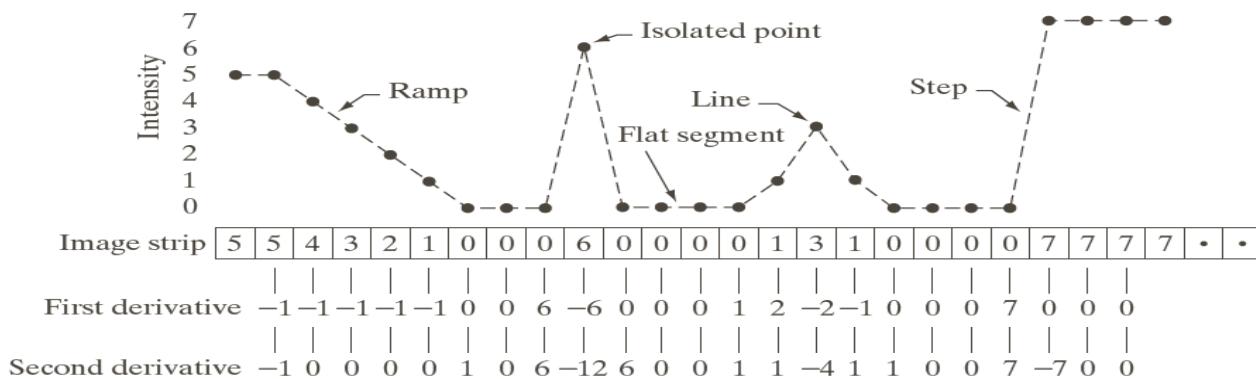
- Basic discontinuities in images can be detected from the 1<sup>st</sup> and 2<sup>nd</sup> order derivatives of the intensity profile
- Sensitive to image noise
- noise filtering needed



**FIGURE 10.2** (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).

# Characteristics of First and Second Order Derivatives

- 1<sup>st</sup> order derivatives generally produce thicker edges in image
  - 2<sup>nd</sup> order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise
  - Second-order derivatives produce a double-edge response at ramp and step transition in intensity
  - The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light



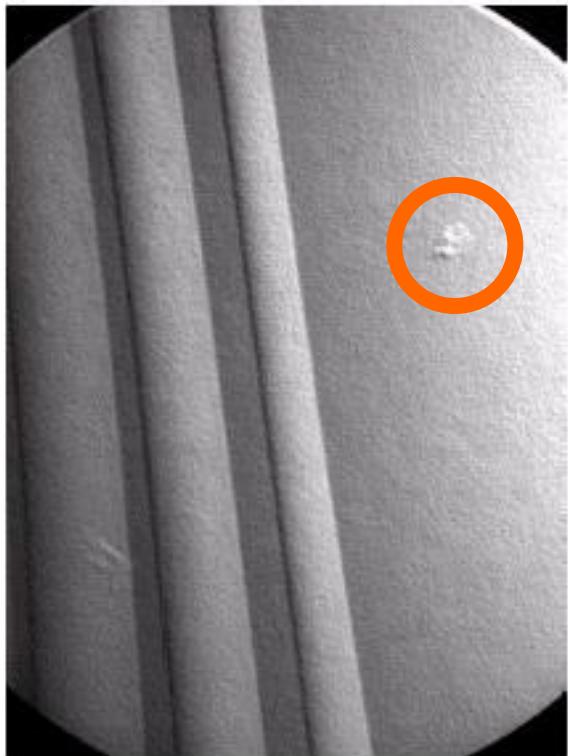
# Point Detection

Point detection can be achieved simply using the mask below:

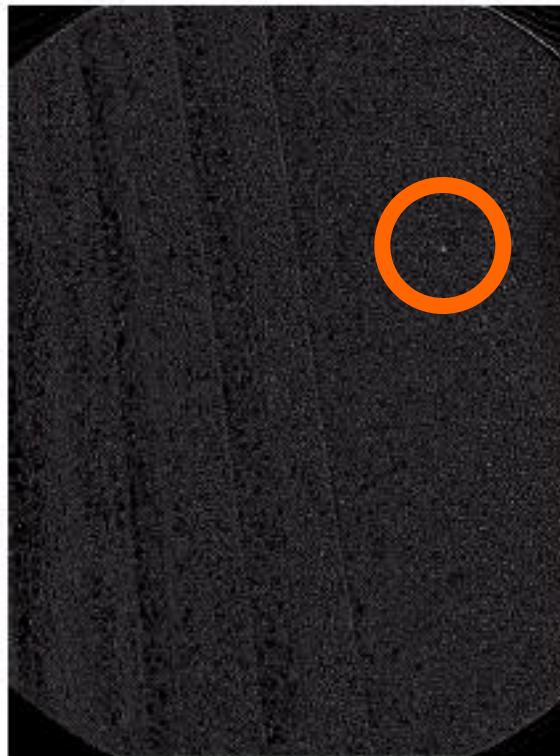
-1	-1	-1
-1	8	-1
-1	-1	-1

Points are detected at those pixels in the subsequent filtered image that are above a set threshold

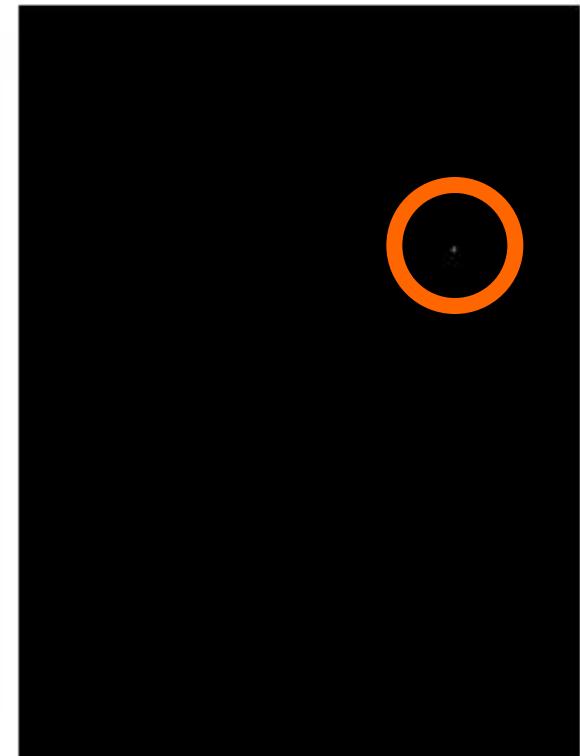
# Point Detection (cont...)



X-ray image of  
a turbine blade



Result of point  
detection



Result of  
thresholding

# Line Detection

- The next level of complexity is to try to detect lines
- The masks below will extract lines that are one pixel thick and running in a particular direction

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

Horizontal

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

+45°

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

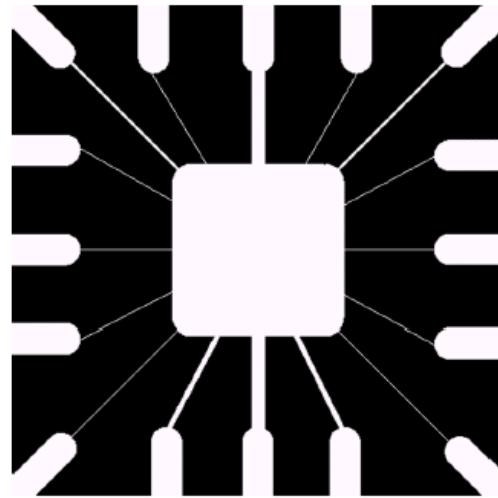
Vertical

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

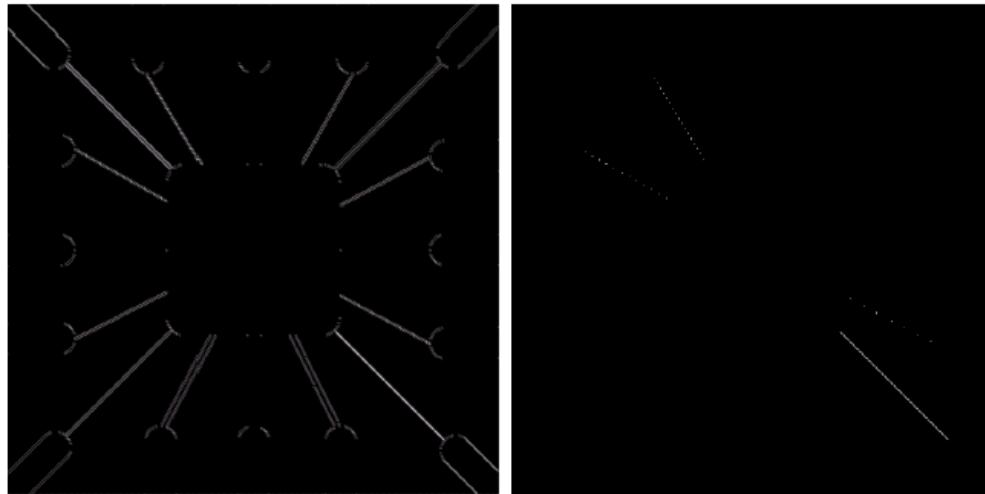
-45°

# Line Detection (cont...)

Binary image of a wire bond mask



After  
processing  
with  $-45^\circ$  line  
detector



Result of  
thresholding  
filtering result

# Edge Detection

- **What is an edge?** An edge is a set of connected pixels that lie on the boundary between two regions. Edges are pixels where there is local transition of image intensities

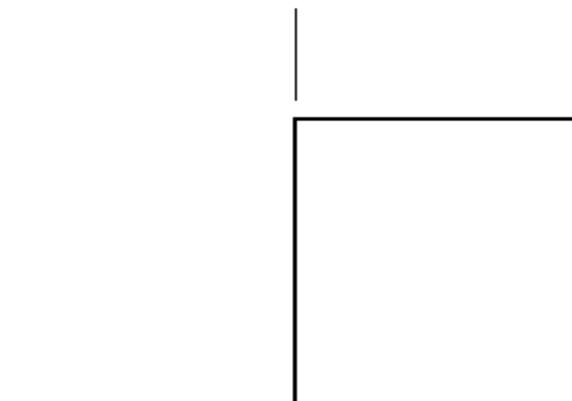
Model of an ideal digital edge



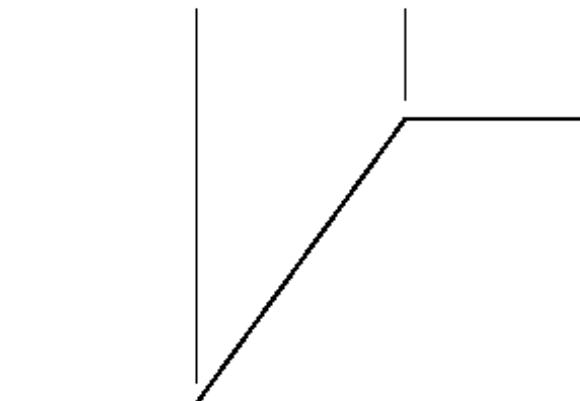
Model of a ramp digital edge



Gray-level profile  
of a horizontal line  
through the image



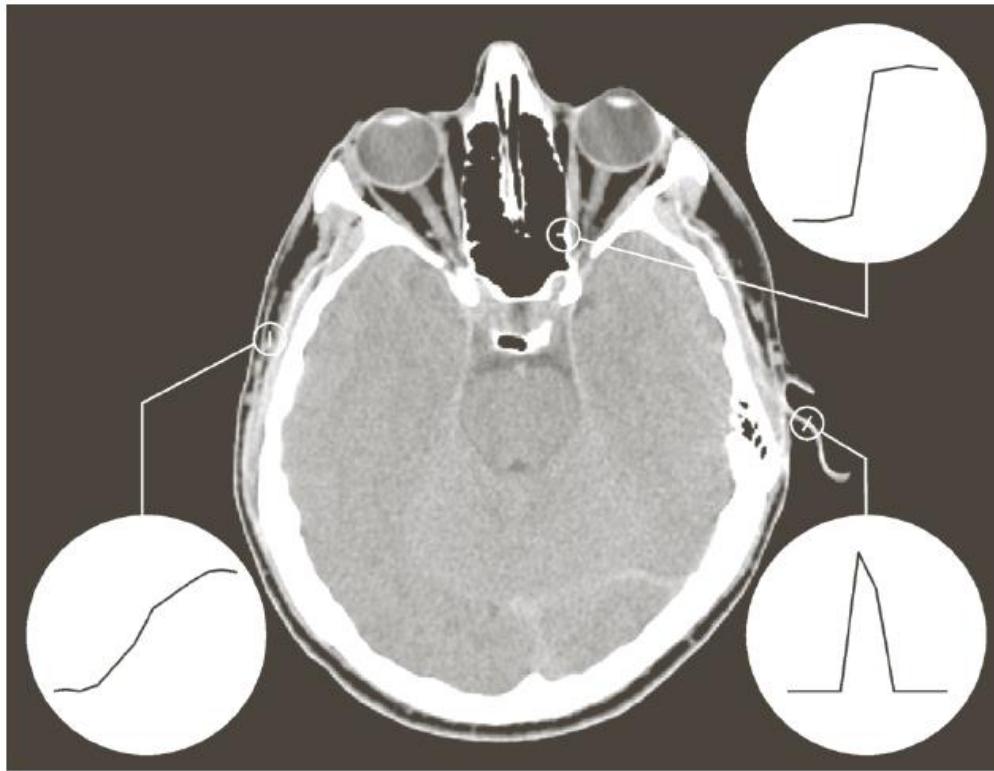
Gray-level profile  
of a horizontal line  
through the image



# Edge Detection

- **Why Edge detection?**
  - Helps in determining object boundaries and segmentation
  - More compactly represents salient features of the scene than pixels
  - Further features can be extracted from the edges of an image (e.g., corners, lines etc.)
  - These features can be used by higher-level computer vision algorithms (e.g., recognition)

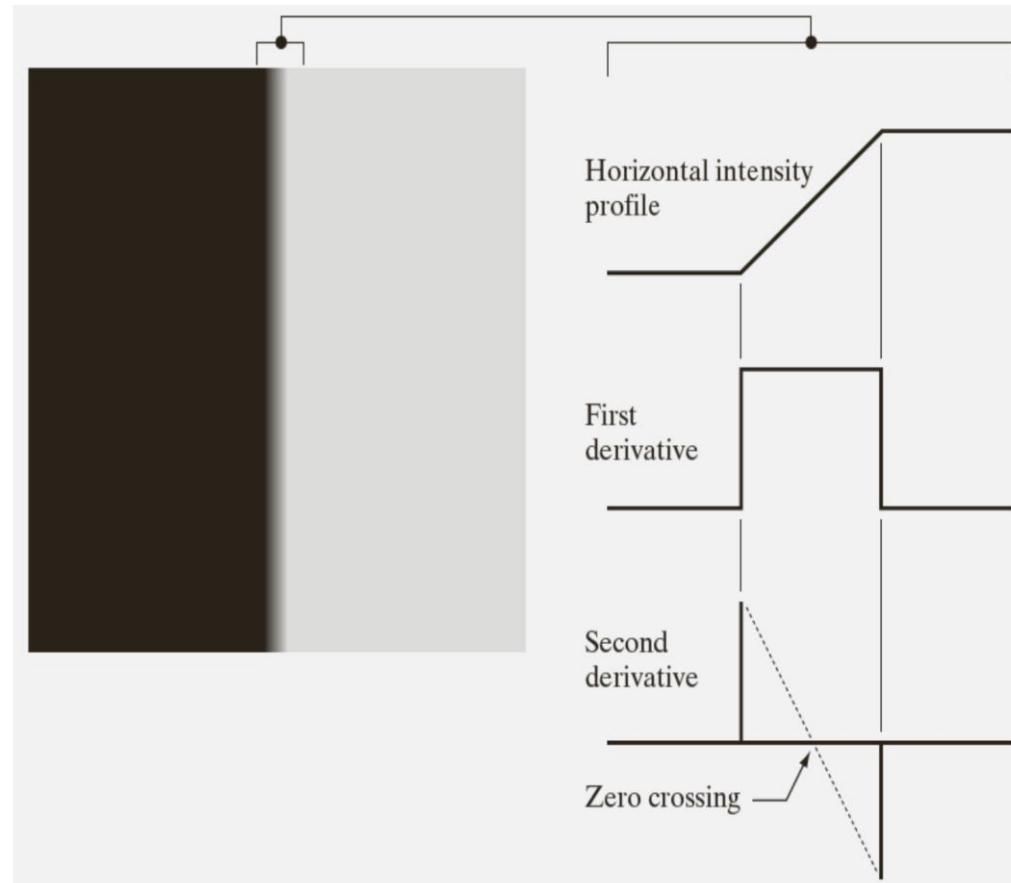
# Edges in reality



**FIGURE 10.9** A  $1508 \times 1970$  image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and “step” profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

# Edges & Derivatives

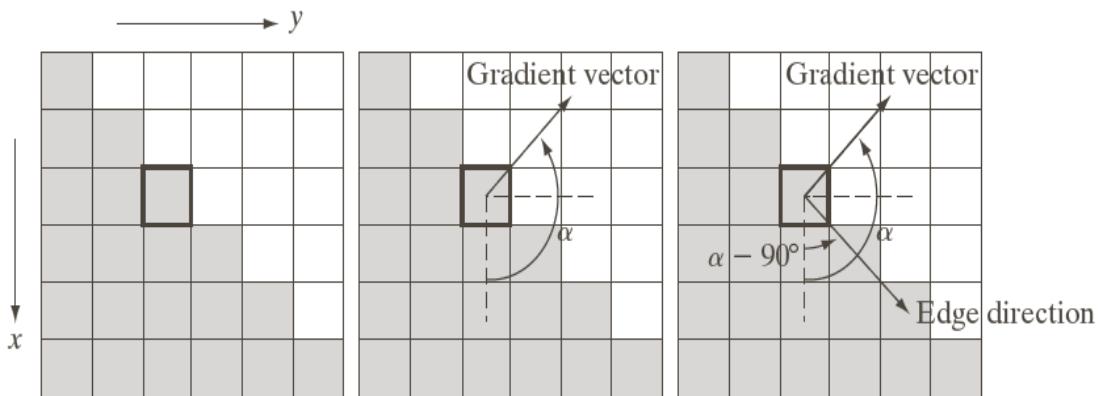
- Recall earlier discussions on how derivatives are used to find discontinuities



# Edge Detection

- Finding the edge strength and direction
- Basic Edge Detection by Using First-Order Derivative

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$



Gradient magnitude (strength):  $M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$

Gradient direction:  $\alpha(x, y) = \tan^{-1} \left[ \frac{g_y}{g_x} \right]$

- The direction of the edge  $\phi = \alpha - 90^\circ$

# Common Edge Detectors

- Given a 3\*3 region of an image the following edge detection filters can be used

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

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Prewitt

-1	0
0	1
0	1
1	0

Roberts

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

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

Sobel

# Edge Detection Example

Original Image



Horizontal Gradient Component



Vertical Gradient Component



Combined Edge Image



# Edge Detection Example



# Edge Detection Example



# Edge Detection Example



# Edge Detection Example



# Edge Detection Problems

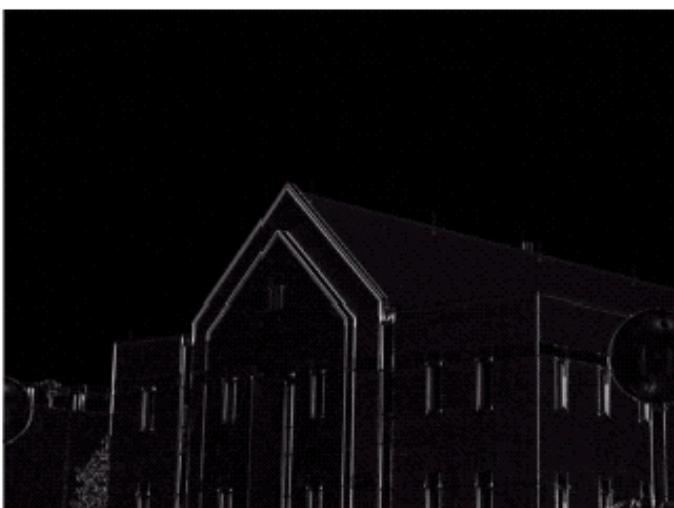
- Often, problems arise in edge detection in that there is too much detail
- For example, the brickwork in the previous example
- One way to overcome this is to smooth images prior to edge detection

# Edge Detection Example With Smoothing

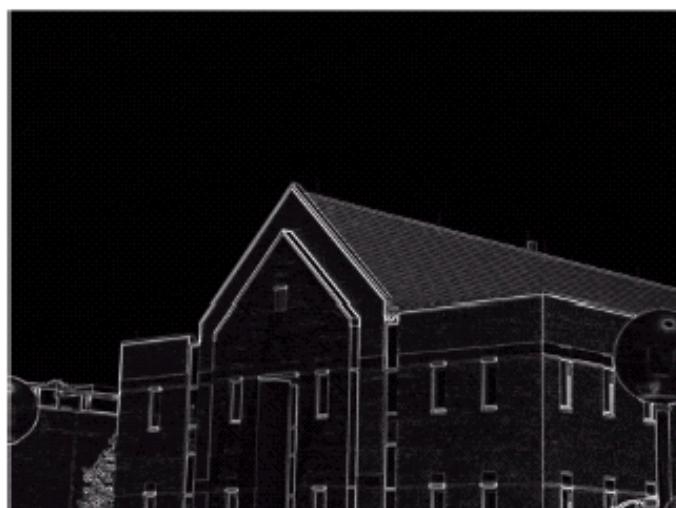
Original Image



Horizontal Gradient Component



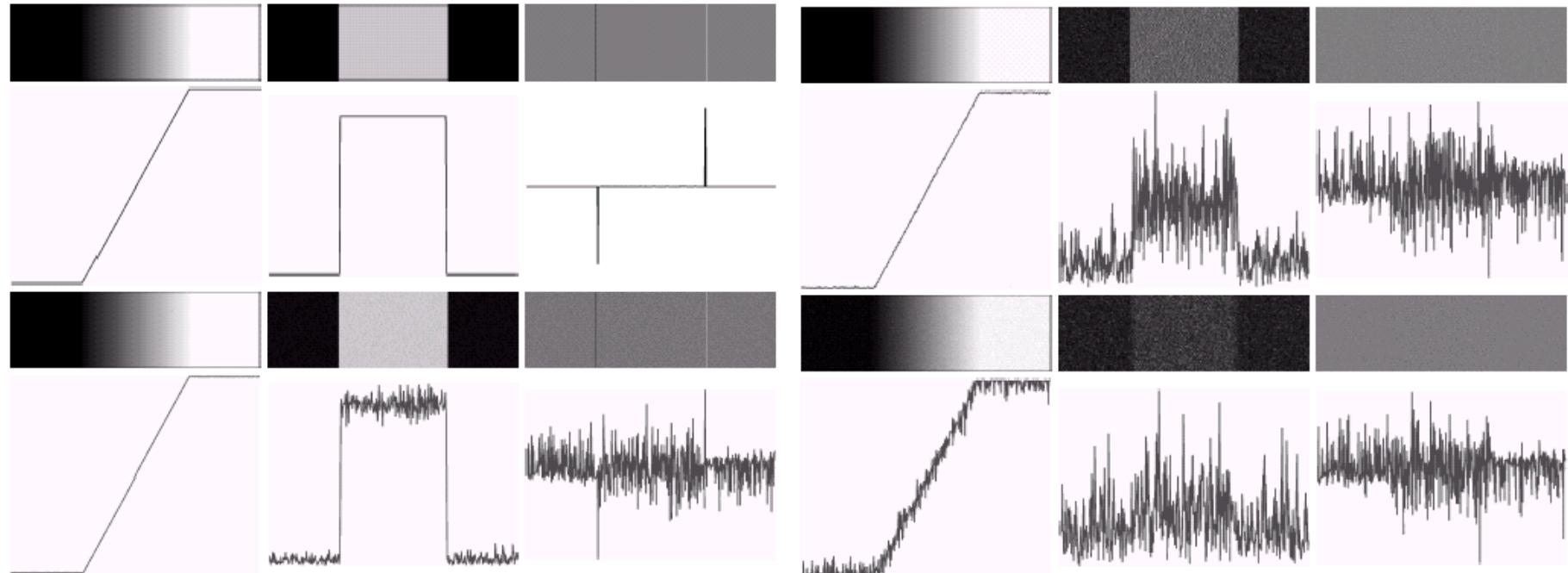
Vertical Gradient Component



Combined Edge Image

# Derivatives & Noise

- Derivative based edge detectors are extremely sensitive to noise



# Importance of Scale

- Structures exist at multiple scales



# Scale space



A simple image pyramid

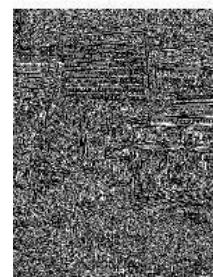


Image scale space generated by varying sigma of a smoothing function

# Feature scale space

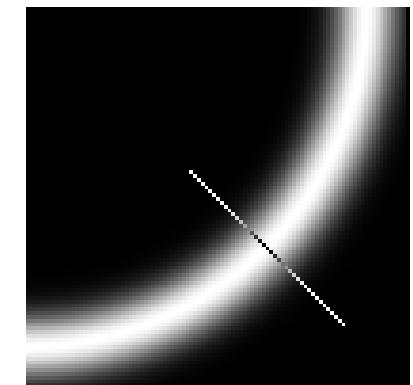


original

 $\sigma = 1.5$  $\sigma = 3.0$  $\sigma = 6.0$  $\sigma = 12.0$  $\sigma = 1.0$  $\sigma = 2.0$  $\sigma = 4.0$  $\sigma = 10.0$

# Steps in edge detection

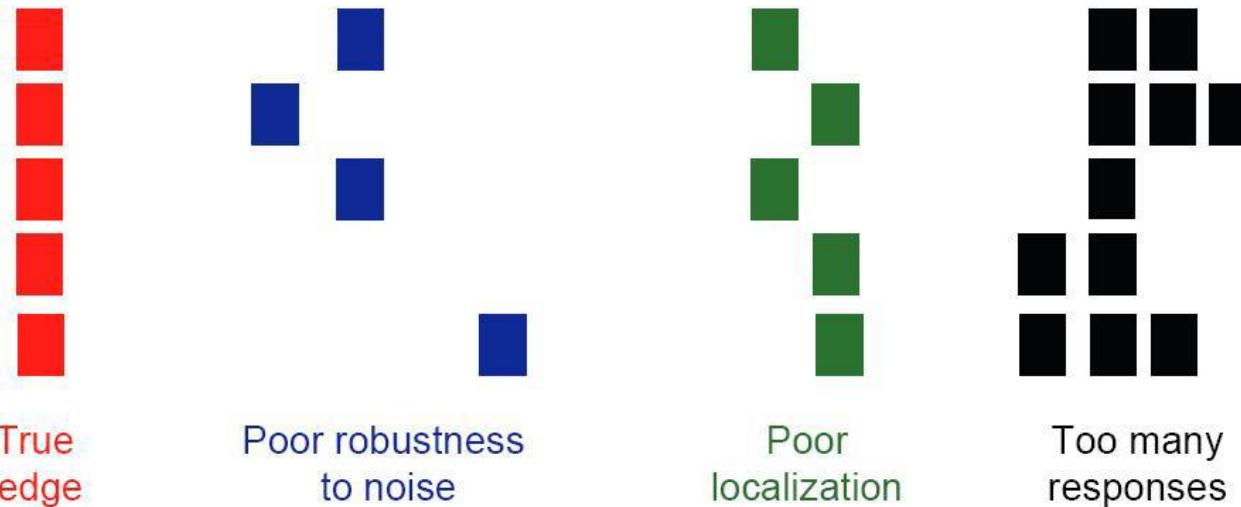
1. Image smoothing for reduction of noise and irrelevant details
2. Detection of edge points
3. Edge localization - Most edges are not sharp dropoffs. Extracting the ideal edge is thus a matter of finding the curve with peak/optimal gradient magnitude.
4. Edge Linking



# Optimal edge detector

Criteria for an “optimal” edge detector:

- **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
- **Good localization:** the edges detected must be as close as possible to the true edges
- **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



## 2<sup>nd</sup> Derivative Methods

- Finding optimal edges (maxima of gradient magnitude) is equivalent to finding *places where the second derivative is zero*
- The zeroes may not fall exactly on a pixel. We can isolate these zeroes by finding *zero crossings*: places where one pixel is positive and a neighbor is negative (or vice versa)
- zero crossings form closed paths and extremely sensitive to noise

# 2<sup>nd</sup> Derivative Methods: Laplacian

We encountered the 2<sup>nd</sup>-order derivative based Laplacian filter already – isotropic i.e rotationally invariant

0	-1	0
-1	4	-1
0	-1	0

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

The Laplacian is typically not used by itself as it is too sensitive to noise .It is combined with a smoothing Gaussian filter for purpose of edge detection

# Laplacian Of Gaussian (Marr-Hildreth edge detector)

- To reduce the noise effect and also to adjust the scale, the image is first smoothed.
- When the filter chosen is a Gaussian, we call it the LoG edge detector.

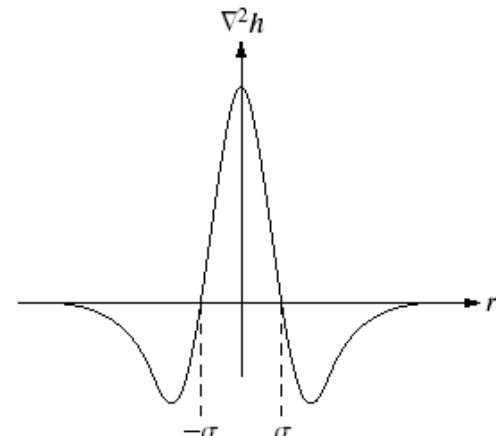
$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$\sigma$  controls smoothing/adjusts scale

- It can be shown that:

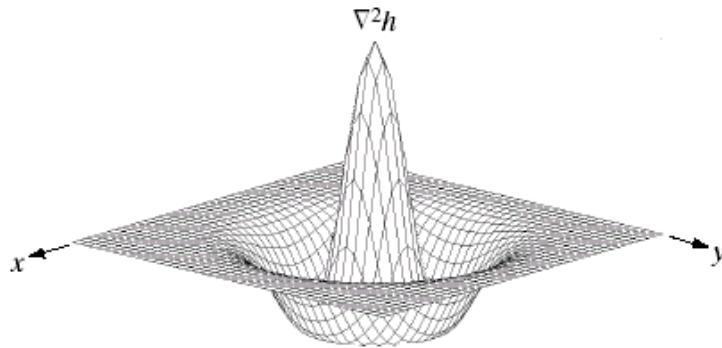
$$\nabla^2[f(x, y) * G(x, y)] = \nabla^2G(x, y) * f(x, y)$$

$$\nabla^2G(x, y) = \left(\frac{r^2 - \sigma^2}{\sigma^4}\right)e^{-r^2/2\sigma^2}, (r^2 = x^2 + y^2)$$

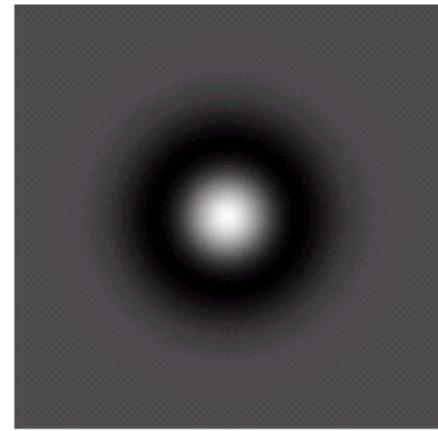


# Laplacian Of Gaussian (Marr-Hildreth edge detector)

The Laplacian of Gaussian function can be approximated over a discrete spatial neighborhood to give a convolution kernel which looks like a Mexican hat.



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0



$n$  (size of LOG filter) should be smallest odd integer greater or equal to  $6\sigma$

# Example Result



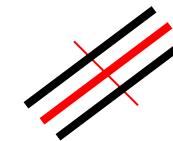
a	b
c	d

**FIGURE 10.22**

(a) Original image of size  $834 \times 1114$  pixels, with intensity values scaled to the range  $[0, 1]$ . (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using  $\sigma = 4$  and  $n = 25$ . (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

# The Canny Edge Detection

- Smooth the image with a Gaussian filter with spread  $\sigma$ .
- Compute gradient magnitude and direction at each pixel of the smoothed image.
- Zero out any pixel response  $\leq$  the two neighboring pixels on either side of it, along the direction of the gradient. This is called nonmaximum suppression.



- Use double thresholding to segregate ‘strong’ (high-magnitude) and ‘weak’ edge pixels.
- Visit strong edge pixels and do connectivity analysis to keep only those weak edge pixels which are connected to them

# The Canny Edge Detection



a b  
c d

**FIGURE 10.25**

(a) Original image of size  $834 \times 1114$  pixels, with intensity values scaled to the range  $[0, 1]$ .  
(b) Thresholded gradient of smoothed image.  
(c) Image obtained using the Marr-Hildreth algorithm.  
(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

# Edge Linking

- Set of pixels from edge detecting algorithms, seldom define a boundary completely because of noise, breaks in the boundary etc.
- Therefore, Edge detecting algorithms are typically followed by linking and other detection procedures, designed to assemble edge pixels into meaningful boundaries.
- 2 types – local and global

# Local Processing

- Analyse the characteristics of pixels in a small neighbourhood ( 3x3, or 5x5 ) about every point that has undergone edge detection.
- All points that are similar are linked, forming a boundary of pixels that share some common properties.
  - strength of the response of the gradient operator used to produce the edge pixel
  - direction of the gradient.

$$|\nabla f(x, y) - \nabla(x_0, y_0)| \leq E,$$

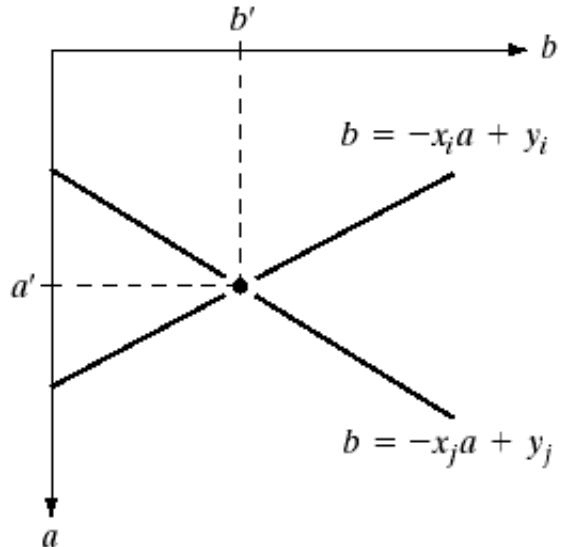
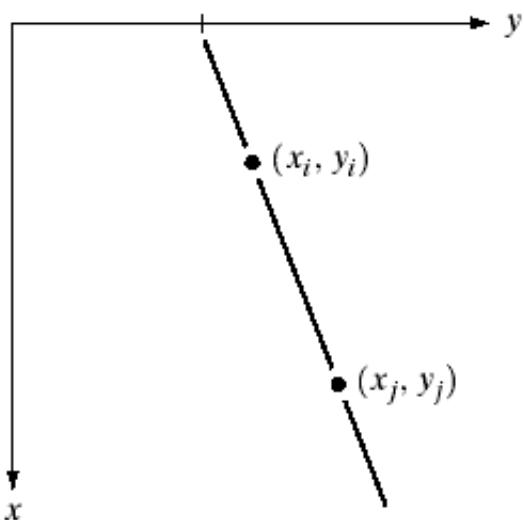
$$|\alpha(x, y) - \alpha(x_0, y_0)| < A$$

# Global Processing via Hough Transform

Hough transform: a way of finding edge points in an image that lie on curves of specified shape.

Example:  $xy$ -plane v.s.  $ab$ -plane (parameter space)

$$y_i = ax_i + b$$



a b

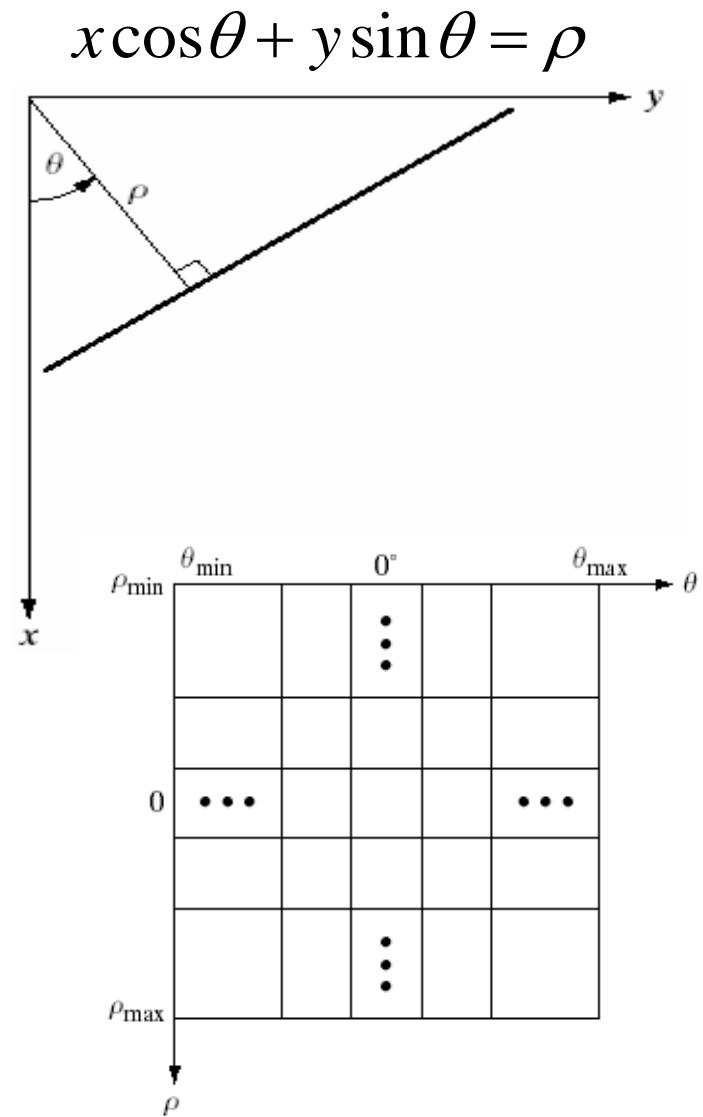
**FIGURE 10.17**  
(a)  $xy$ -plane.  
(b) Parameter space.

# Hough Transform

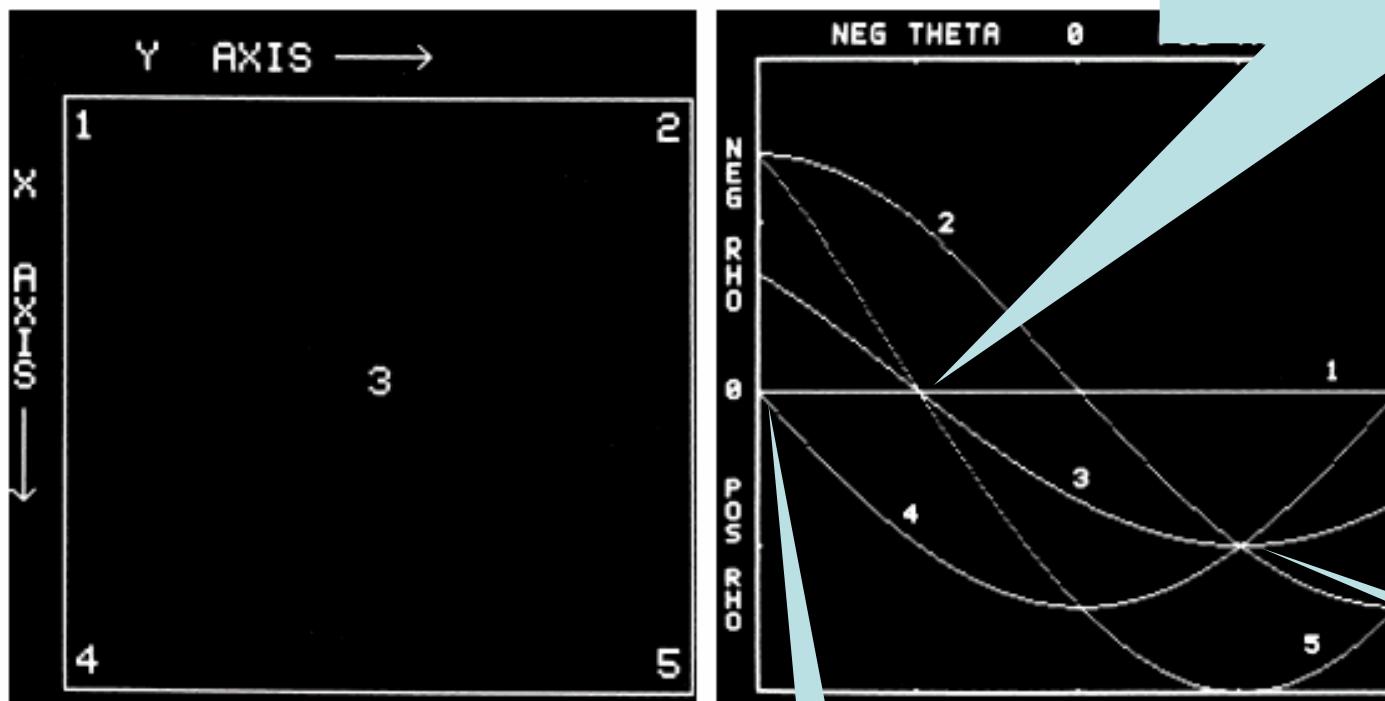
The Hough transform consists of finding all pairs of values of  $\theta$  and  $\rho$  which satisfy the equations that pass through  $(x,y)$ .

These are accumulated in what is basically a 2-dimensional histogram.

When plotted these pairs of  $\theta$  and  $\rho$  will look like a **sine** wave. The process is repeated for all appropriate  $(x,y)$  locations.



# Hough Transform



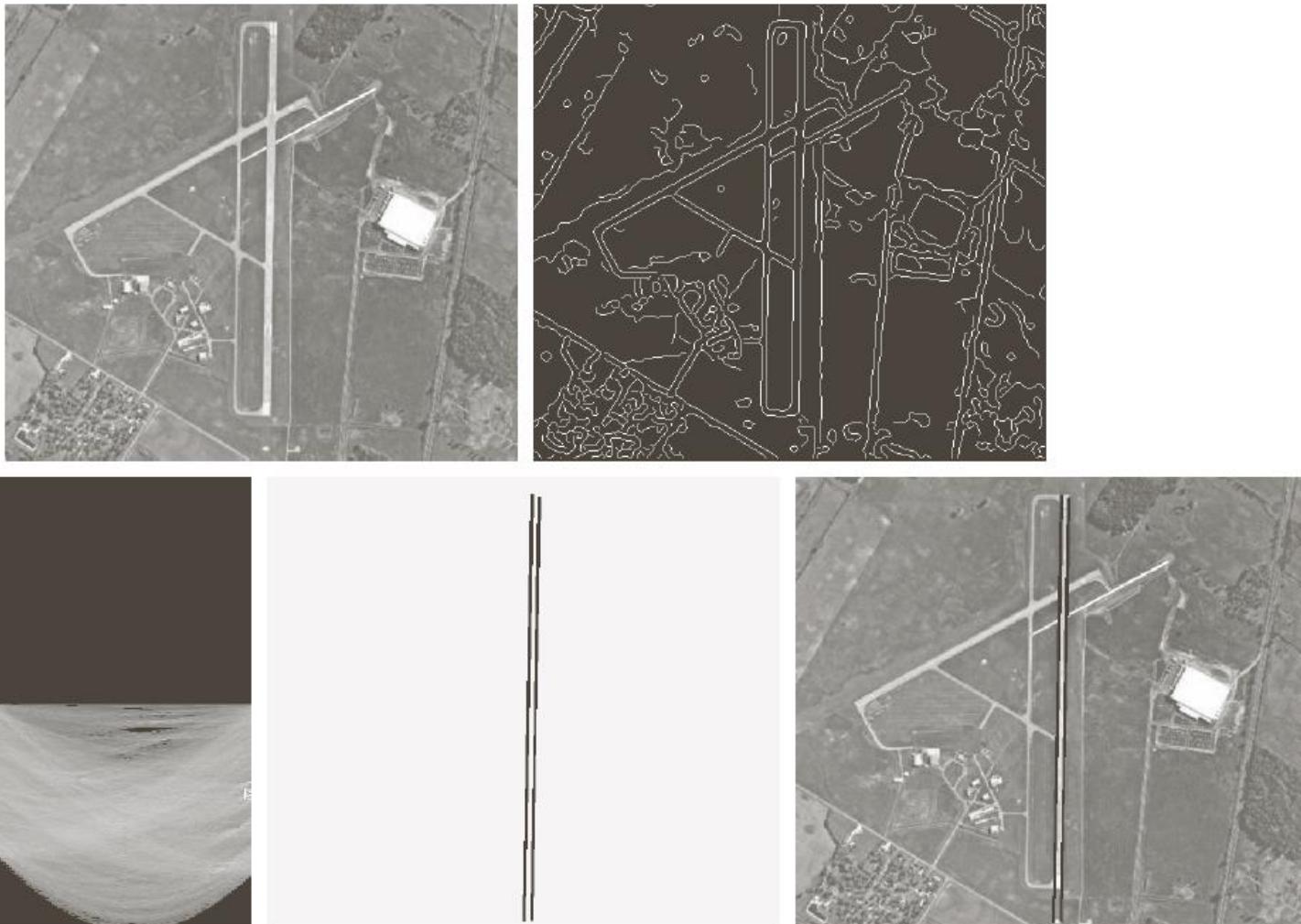
The intersection of the  
curves corresponding to  
points 1,3,5

**FIGURE 10.20**  
Illustration of the  
Hough transform.  
(Courtesy of Mr.  
D. R. Cate, Texas  
Instruments, Inc.)

2,3,4

1,4

# Hough Transform



**FIGURE 10.34** (a) A  $502 \times 564$  aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

# What about Circles?



# Summary

- In this Part, we have begun looking at segmentation, and in particular edge detection
- Edge detection is massively important as it is in many cases the first step to object recognition
- We saw methods of edge linking, especially Hough transform

# Thresholding based segmentation

- Now we will continue to look at the problem of segmentation, this time though in terms of thresholding
- In particular we will look at:
  - What is thresholding?
  - Simple Global thresholding
  - Optimal Global thresholding: Otsu's Algorithm
  - Adaptive/dynamic/local thresholding

# Thresholding

- Thresholding is popular approach because of its intuitive property, simplicity of implementation and computation speed
- We have talked about simple single value thresholding already
- Single value thresholding can be given mathematically as follows:

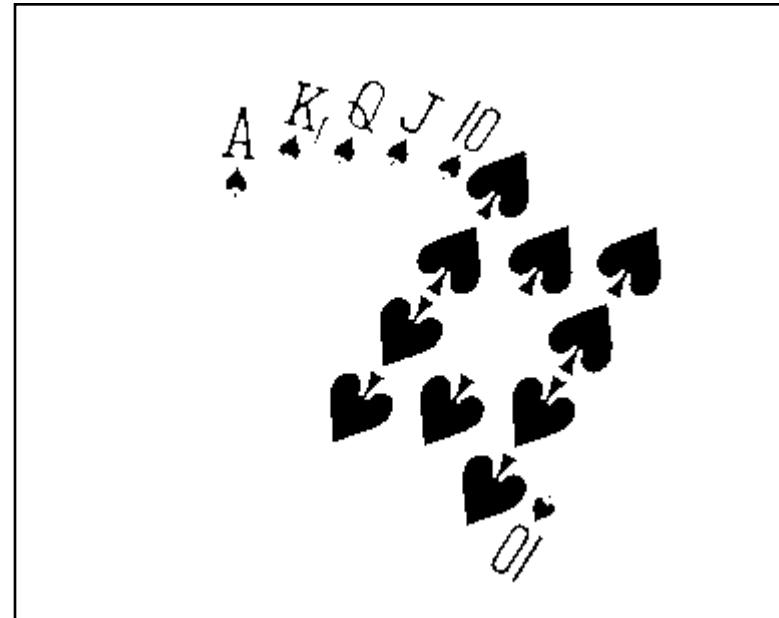
$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$

# Thresholding Example

- Imagine a poker playing robot that needs to visually interpret the cards in its hand



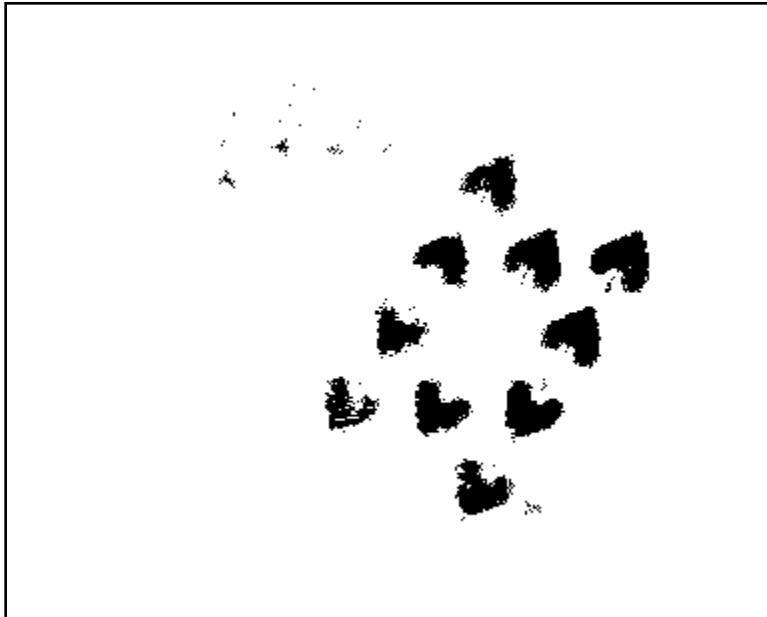
Original Image



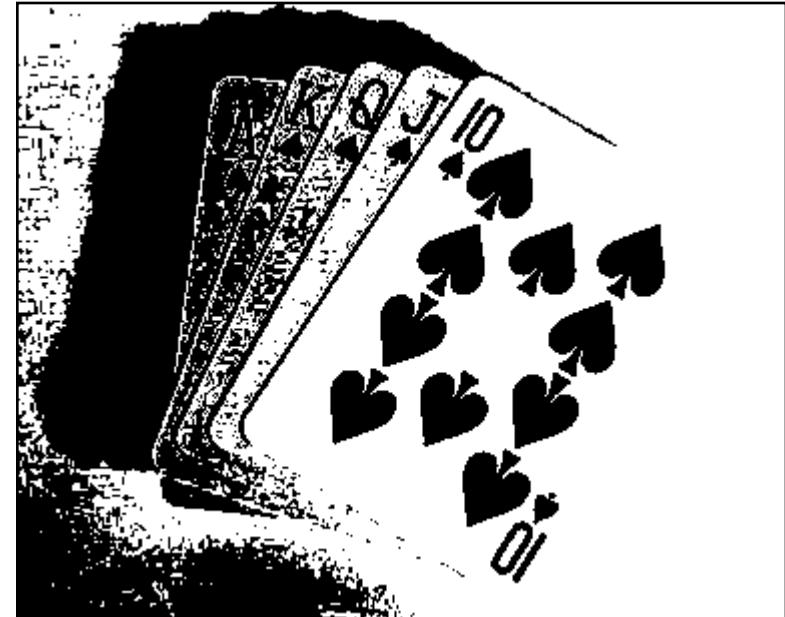
Thresholded Image

# But Be Careful

- If you get the threshold wrong the results can be disastrous



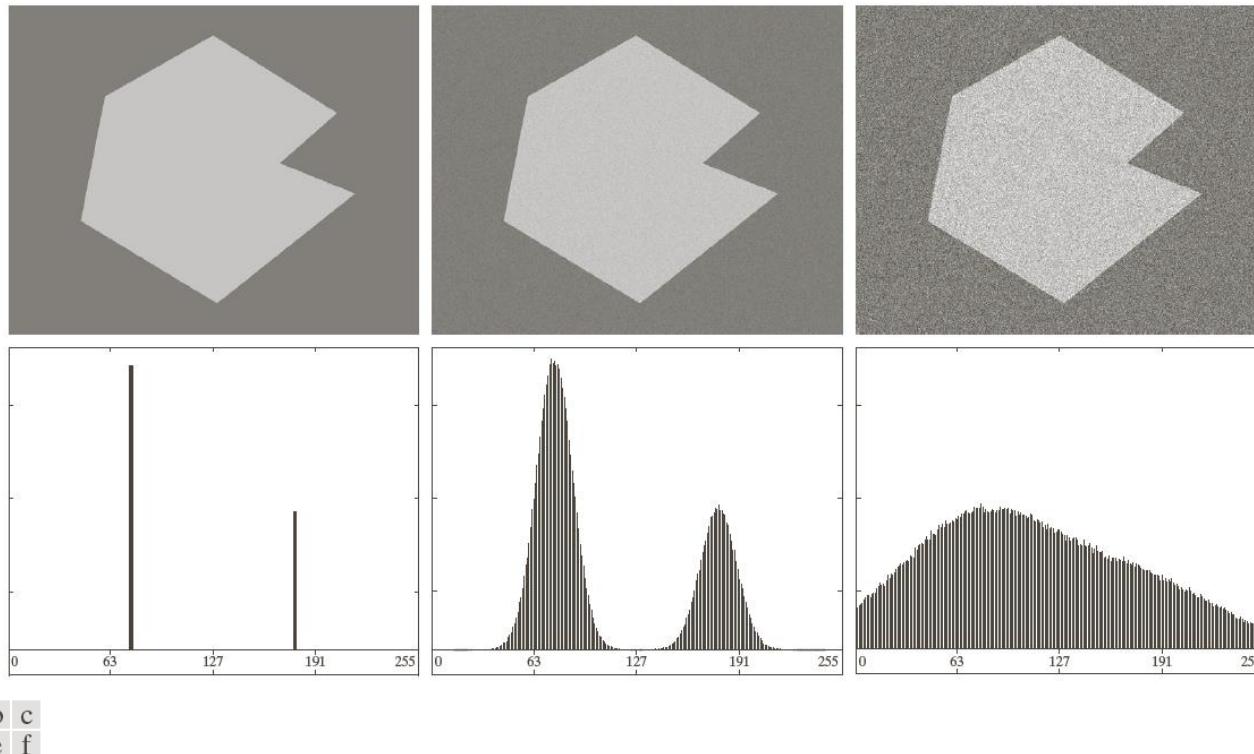
Threshold Too Low



Threshold Too High

# Noise in Thresholding

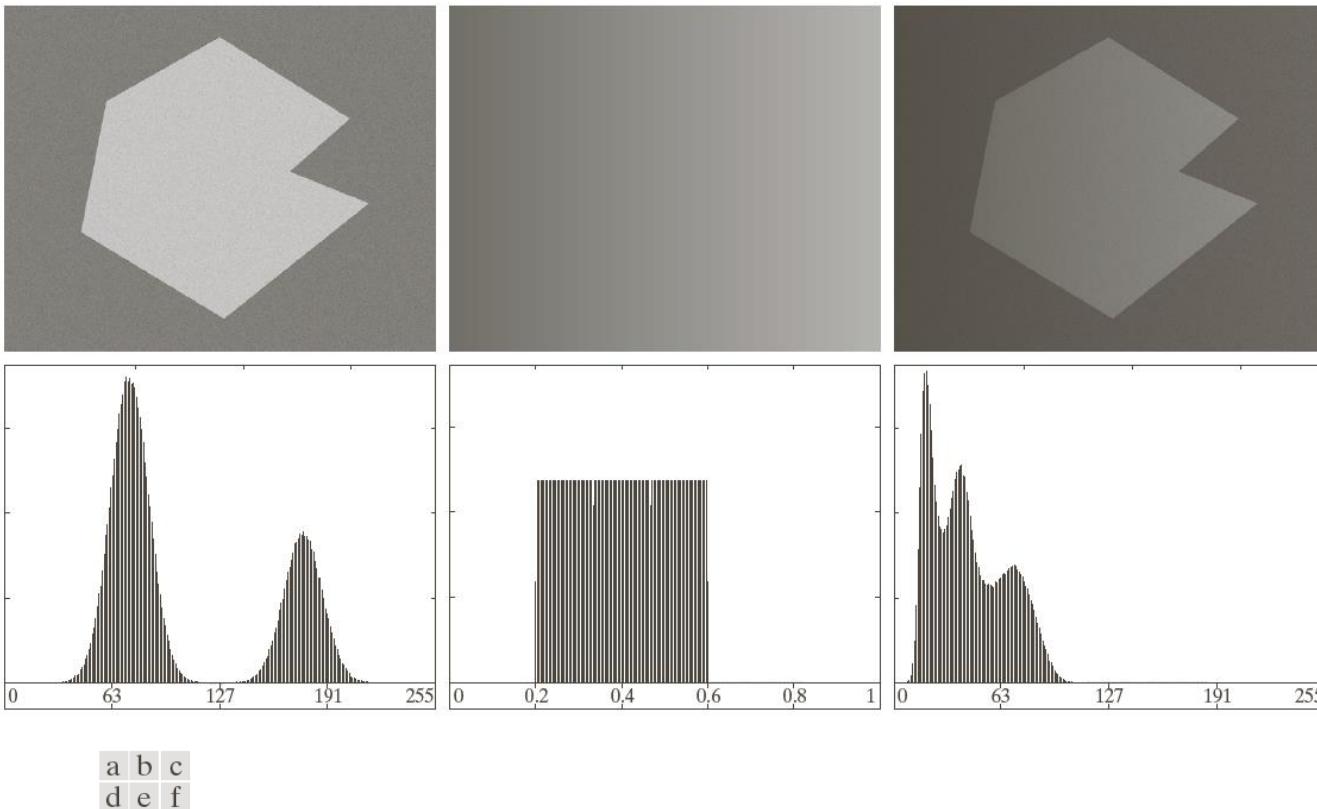
- Difficulty in determining the threshold due to noise



**FIGURE 10.36** (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

# Illumination in Thresholding

- Difficulty in determining the threshold due to non uniform illumination or reflectance



**FIGURE 10.37** (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

# Basic Global Thresholding

- Based on the histogram of an image
- Partition the image histogram using a single global threshold
- The success of this technique very strongly depends on how well the histogram can be partitioned

# Basic Global Thresholding Algorithm

- The basic global threshold,  $T$ , is calculated as follows:
  1. Select an initial estimate for  $T$  (typically the average grey level in the image)
  2. Segment the image using  $T$  to produce two groups of pixels:  $G_1$  consisting of pixels with grey levels  $> T$  and  $G_2$  consisting of pixels with grey levels  $\leq T$
  3. Compute the average grey levels of pixels in  $G_1$  to give  $\mu_1$  and  $G_2$  to give  $\mu_2$

# Basic Global Thresholding Algorithm

4. Compute a new threshold value:

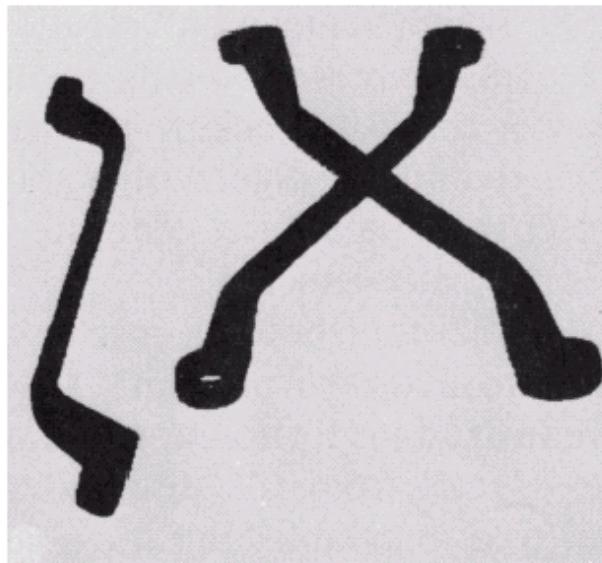
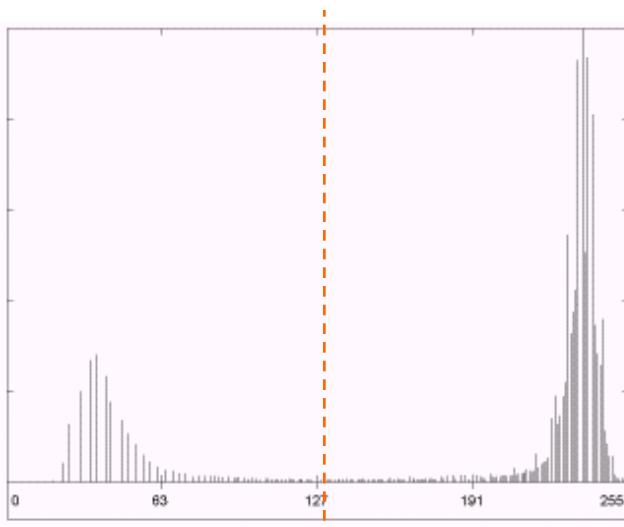
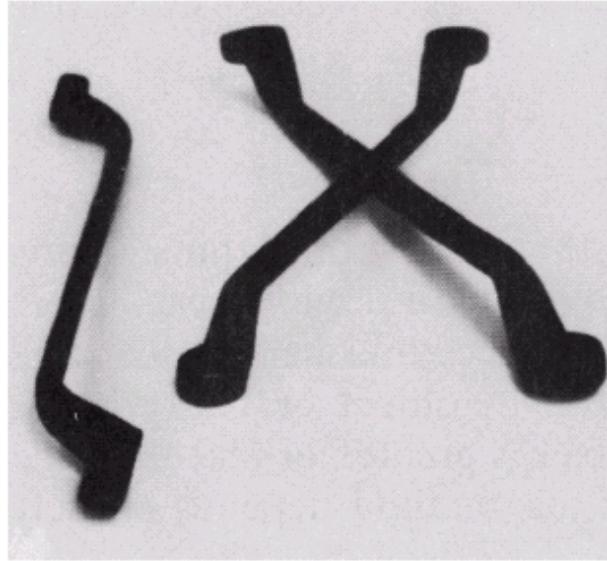
$$T = \frac{\mu_1 + \mu_2}{2}$$

5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit  $T_\infty$

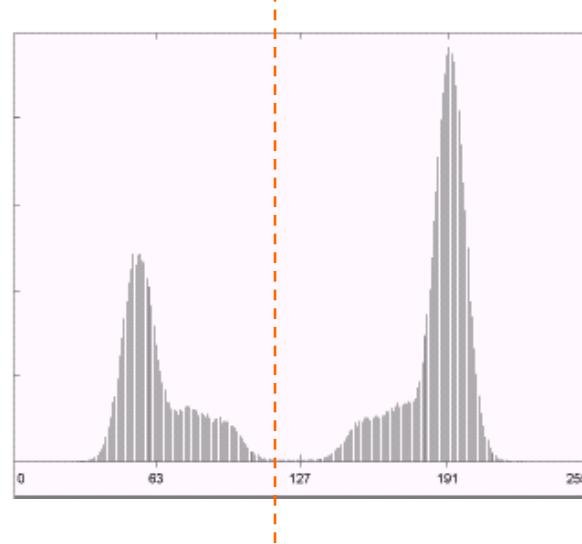
- This algorithm works very well for finding thresholds when the histogram is suitable

10

# Thresholding Example 1



# Thresholding Example 2



# Optimum Global Thresholding (Otsu's Method)

- Otsu's method selects the threshold by minimizing the within-class variance
  - Try to make each cluster as tight as possible thus (hopefully!) minimizing their overlap.
- Assumptions: Histogram (and the image) are *bimodal and uniform illumination*



# Otsu's Method (contd..)

The *weighted within-class variance* is:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Where the class probabilities are estimated as:

$$q_1(t) = \sum_{i=1}^t P(i) \quad q_2(t) = \sum_{i=t+1}^I P(i)$$

And the class means are given by:

$$\mu_1(t) = \sum_{i=1}^t \frac{iP(i)}{q_1(t)} \quad \mu_2(t) = \sum_{i=t+1}^I \frac{iP(i)}{q_2(t)}$$

# Otsu's Method (contd..)

Finally, the individual class variances are:

$$\sigma_1^2(t) = \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \sigma_2^2(t) = \sum_{i=t+1}^I [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

- Run through the full range of  $t$  values [1,256] and pick the value that minimizes  $\sigma_w^2(t)$ .
- But the relationship between the within-class and between-class variances can be exploited to generate a recursion relation that permits a much faster calculation.
- *Between-class variance* is the sum of weighted squared distances between the class means and the grand mean

# Otsu's Method (contd..)

The total variance is the sum of the within-class variances (weighted) and the between class variance. After some algebra, we can express the total variance as...

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\text{Within-class, from before}} + \underbrace{q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2}_{\text{Between-class, } \sigma_B^2(t)}$$

Since the total is constant and independent of  $t$ , the effect of changing the threshold is merely to move the contributions of the two terms back and forth.

So, *minimizing the within-class variance is the same as maximizing the between-class variance.*

The nice thing about this is that we can compute the quantities in  $\sigma_B^2(t)$  recursively as we run through the range of  $t$  values.

# Otsu's Method (Finally...)

*Initialization...*     $q_1(1) = P(1); \mu_1(0) = 0$

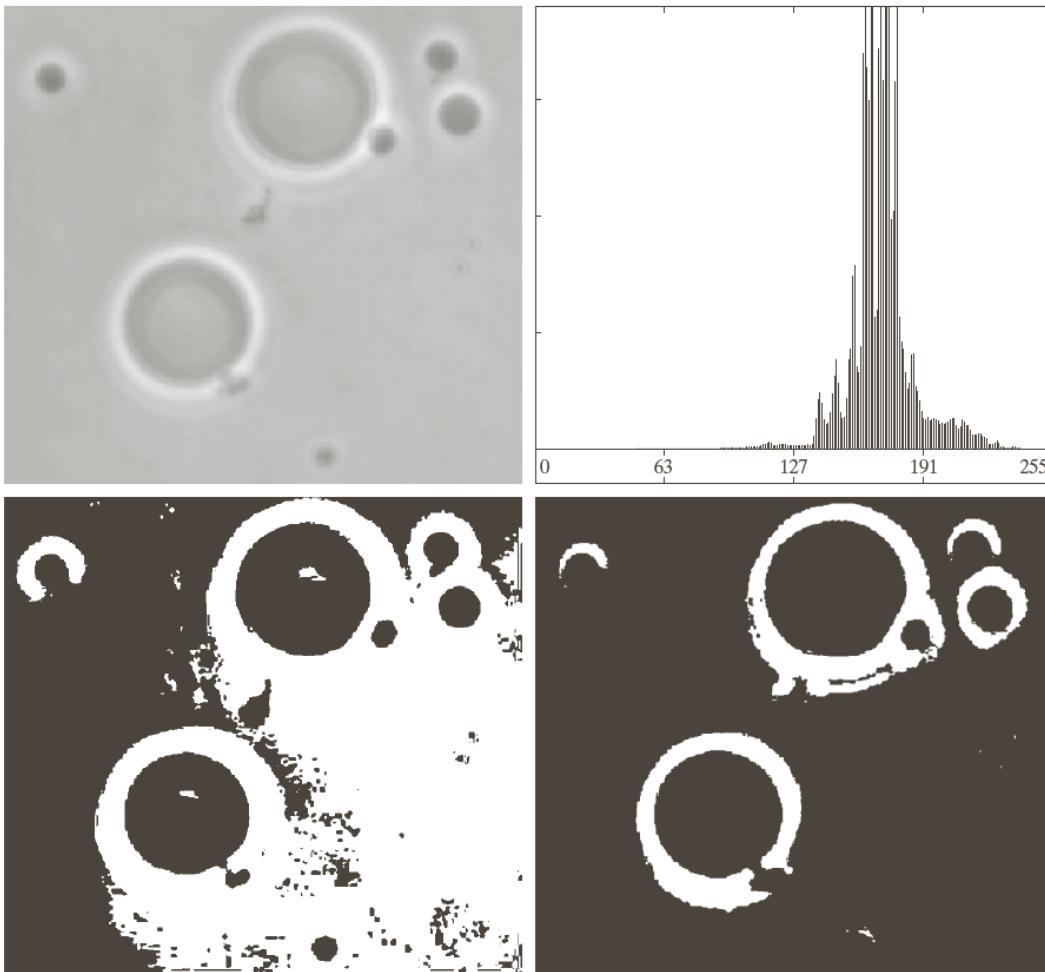
*Recursion...*

$$q_1(t+1) = q_1(t) + P(t+1)$$

$$\mu_1(t+1) = \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)}$$

$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)}$$

# Otsu's example



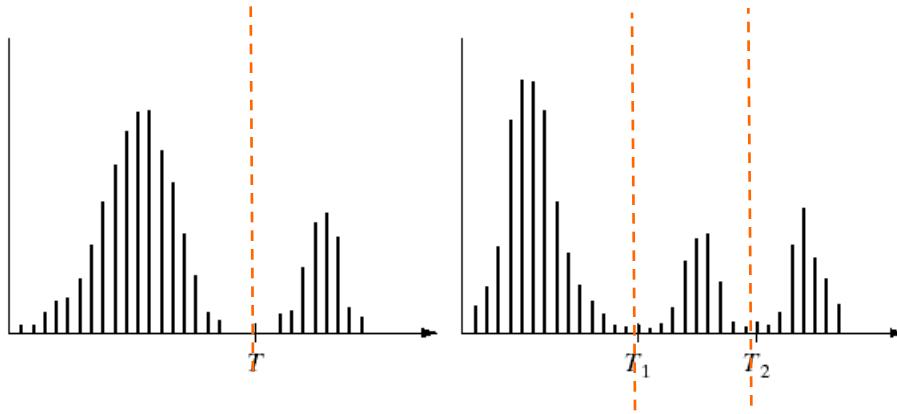
a b  
c d

**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.)

# Problems With Single Value Thresholding

- Single value thresholding only works for bimodal histograms
- Images with other kinds of histograms need more than a single threshold



# Otsu's method extended

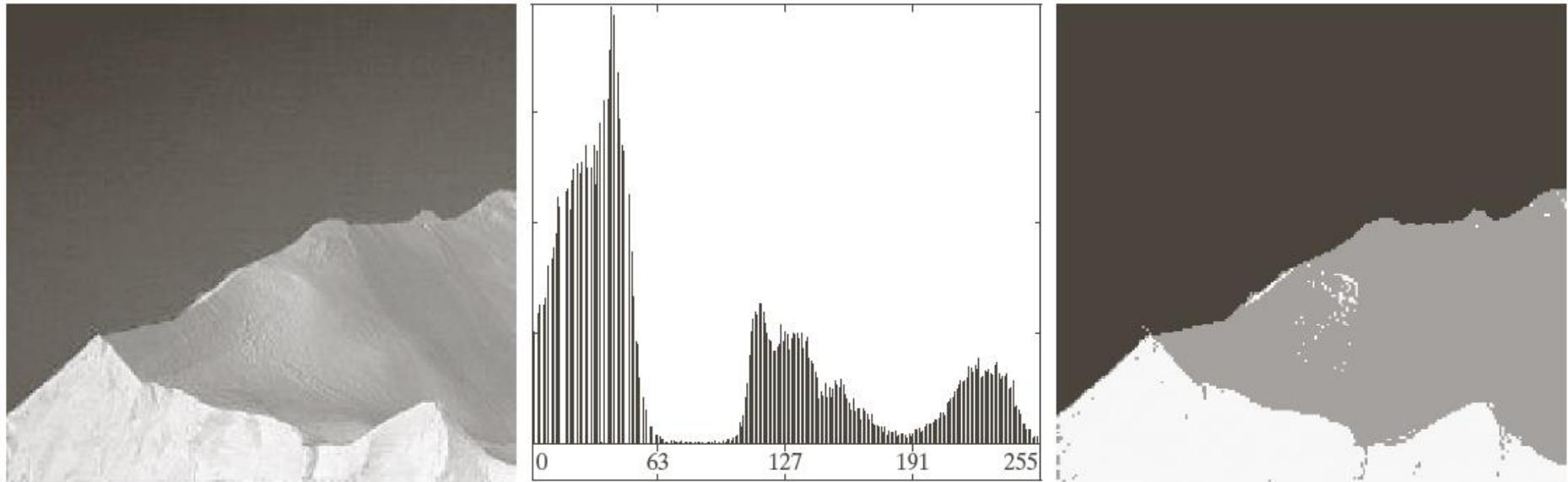
- The method may be extended to multiple thresholds
- In practice for more than 2 thresholds (3 segments) more advanced methods are employed.
- For three classes, the between-class variance is:

$$\sigma_B^2(k_1, k_2) = P_1(k_1)[m_1(k_1) - m_G] + P_2(k_1, k_2)[m_2(k_1, k_2) - m_G] + P_3(k_2)[m_3(k_2) - m_G]$$

- The thresholds are computed by searching all pairs for values:

$$(k_1^*, k_2^*) = \max_{0 \leq k_1 < k_2 \leq L-1} \{\sigma_B^2(k_1, k_2)\}$$

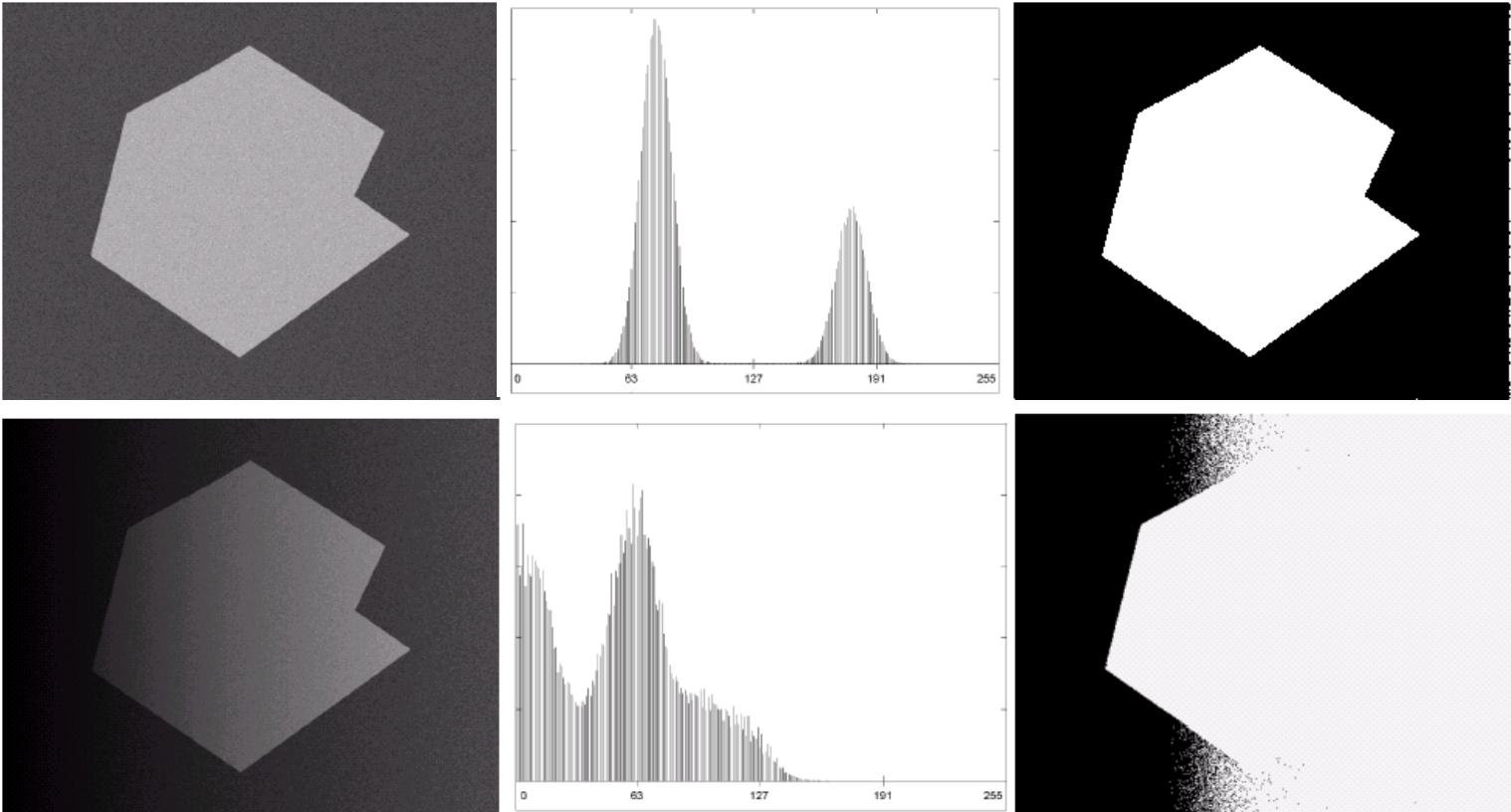
# Otsu's dual threshold example



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.)

# Single Value Thresholding and Illumination



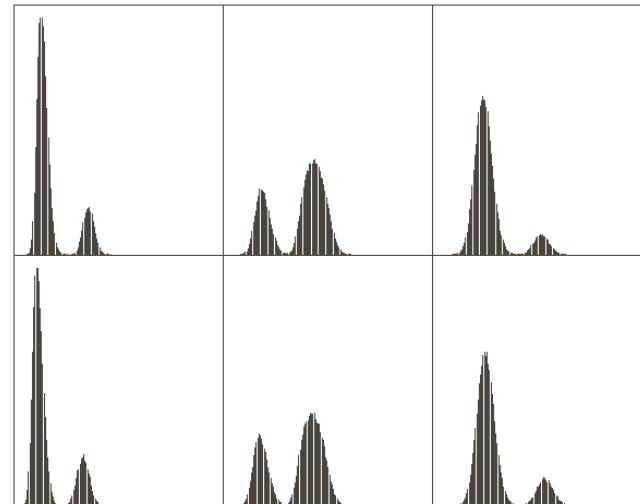
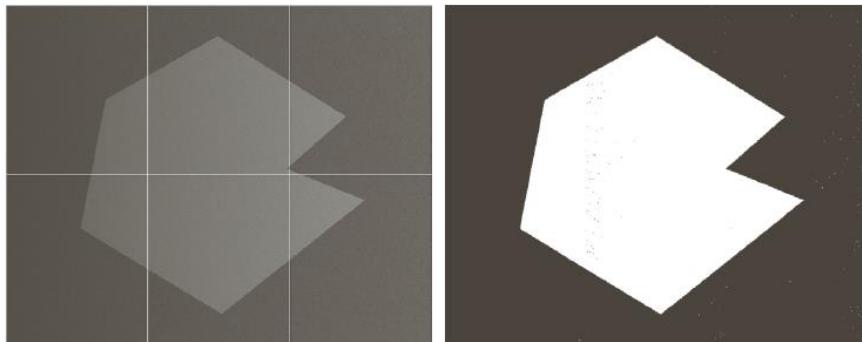
- Uneven illumination can really upset a single valued thresholding scheme

# Basic Adaptive Thresholding

- An approach to handling situations in which single value thresholding will not work is to divide an image into sub images and threshold these individually
- Since the threshold for each pixel depends on its location within an image this technique is said to *adaptive*

# Basic Adaptive Thresholding Example

- The image below shows an example of using adaptive thresholding with the image shown previously



# Variable thresholding based on local processing

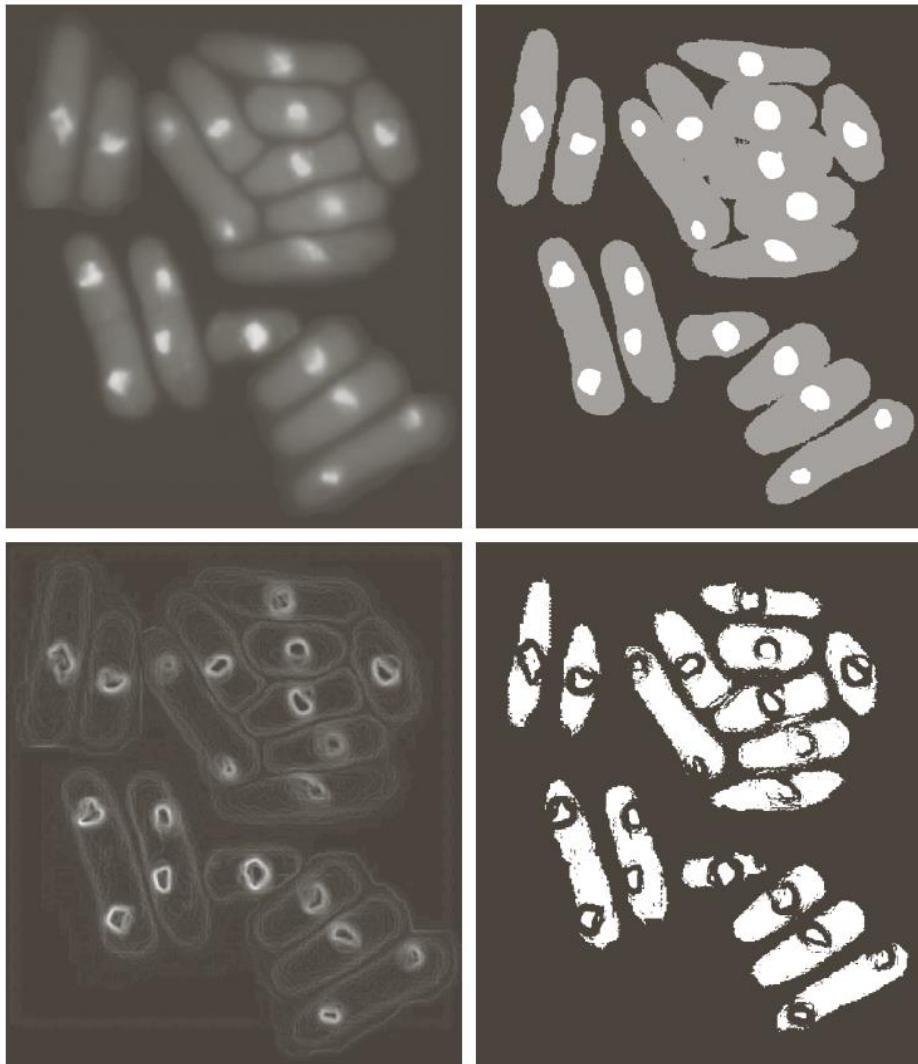
## Use of local image properties.

- Compute a threshold for every single pixel in the image based on its neighborhood ( $m_{xy}$ ,  $\sigma_{xy}$ , ...).

$$g(x, y) = \begin{cases} 1 & Q(\text{local properties}) \text{ is true} \\ 0 & Q(\text{local properties}) \text{ is false} \end{cases}$$

$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} \text{true} & f(x, y) > a\sigma_{xy} \text{ AND } f(x, y) > bm_{xy} \\ \text{false} & \text{otherwise} \end{cases}$$

# Example of local thresholding



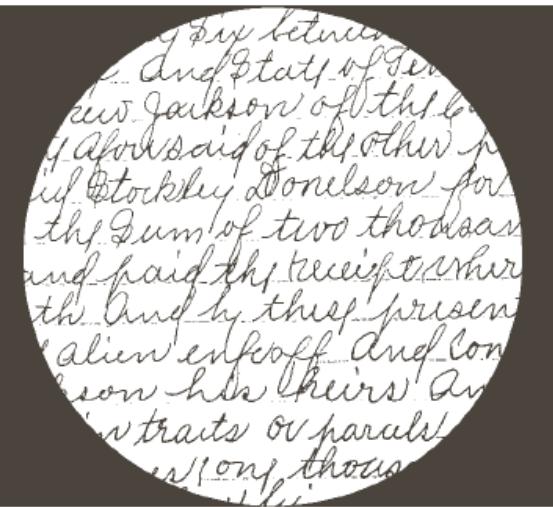
a b  
c d

**FIGURE 10.48**

- (a) Image from Fig. 10.43.  
(b) Image segmented using the dual thresholding approach discussed in Section 10.3.6.  
(c) Image of local standard deviations.  
(d) Result obtained using local thresholding.

# local thresholding using moving average

Ind County Six between Stockley  
of Knox. And State of Tennessee  
Andrew Jackson off the County  
date above said of the other part  
paid Stockley Donelson for A  
of the sum of two thousand  
and paid the receipt wherit  
hath And by these presents  
by alien enforff And Confir  
Jackson his heirs And C  
certain traits or parcels of La  
and ares one thousand p[ounds]  
and bushell and his



Ind County Six between Stockley  
of Knox. And State of Tennessee  
Andrew Jackson off the County  
date above said of the other part  
paid Stockley Donelson for A  
of the sum of two thousand  
and paid the receipt wherit  
hath And by these presents  
by alien enforff And Confir  
Jackson his heirs And C  
certain traits or parcels of La  
and ares one thousand p[ounds]  
and bushell and his

Ind County Six between Stockley  
of Knox. And State of Tennessee  
Andrew Jackson off the County  
date above said of the other part  
paid Stockley Donelson for A  
of the sum of two thousand  
and paid the receipt wherit  
hath And by these presents  
by alien enforff And Confir  
Jackson his heirs And C  
certain traits or parcels of La  
and ares one thousand p[ounds]  
and bushell and his

a b c

**FIGURE 10.49** (a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

# Summary

- In this lecture we looked at thresholding based segmentation techniques
- We saw the basic global and ostu's global thresholding algorithm
- We also saw adaptive thresholding methods

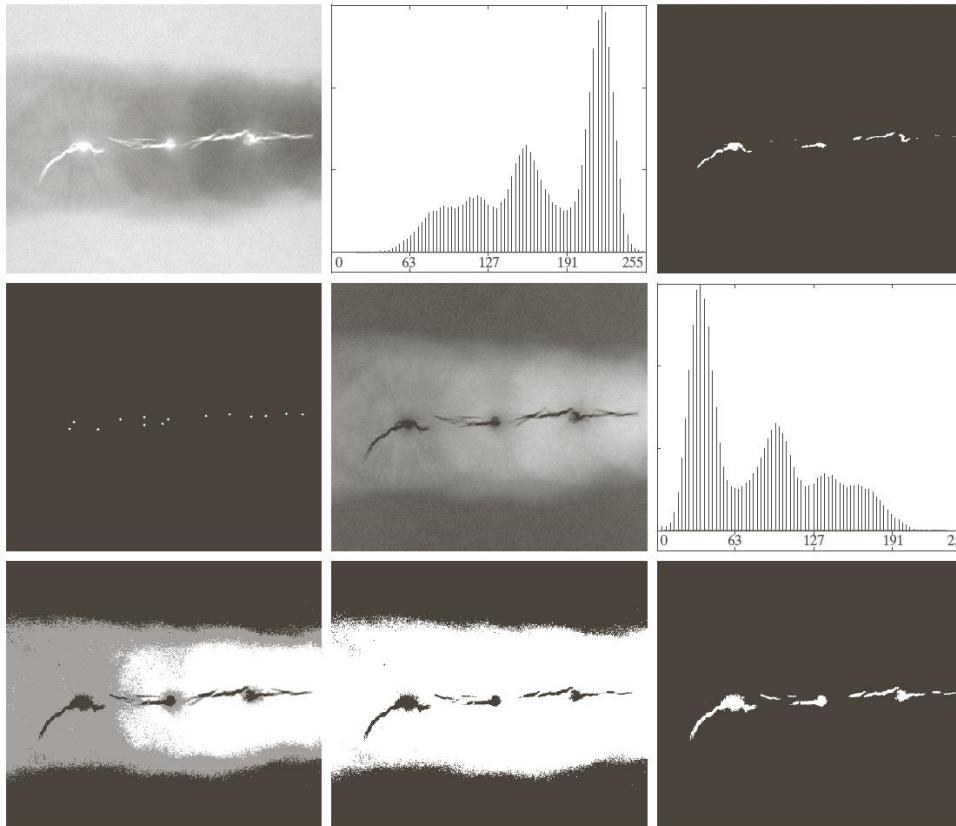
# Region based segmentation

- **Thresholding** used global criterion to decide whether a pixel belong to a region
- **Region growing:** Conversely, local methods will accumulate pixels into a region: starting from a single pixel we would inspect the neighbours, adding them to the region if they are similar.
- Issues to be addressed: 1) What region properties are to be used 2) How do we decide that a pixel should be part of the region.

# Basic region growing algorithm

- Let  $f(x,y)$  be input image,  $S(x,y)$  denote seed array, and  $Q$  be the predicate
- **Algorithm**
  - Find all connected components in  $S(x,y)$  and erode them to 1 pixel.
  - Form image  $f_q(x,y)=1$  if  $f(x,y)$  satisfies the predicate  $Q$ .
  - Form image  $g(x,y)=1$  for all pixels in  $f_q(x,y)$  that are 8-connected to any seed point in  $S(x,y)$ .
  - Label each connected component in  $g(x,y)$  with a different label.

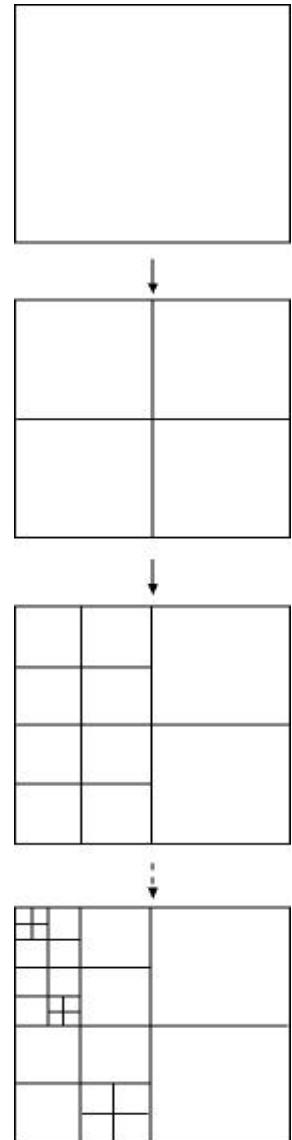
# Region Growing



The weld is very bright. The predicate used for region growing is to compare the absolute difference between a seed point and a pixel to a threshold. If the difference is below it we accept the pixel as crack.

# Region Splitting and Merging

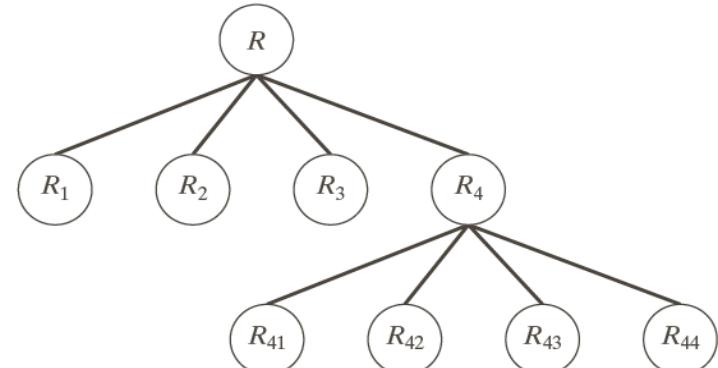
- Previous method is accurate to pixel level but slow.
- Alternate is to split the image to sub-images that do not satisfy a predicate
- If only splitting was used, the final partition would contain adjacent regions with identical properties.
- A merging step follows that merges adjacent regions satisfying the predicate.
- Execution is more rapid but the outlines it generates are less accurate.



# Region Splitting and Merging

- **Algorithm** (Based on quadtrees/*quadimages*. The root of the tree corresponds to the image)
  - Split into four disjoint quadrants any region  $R_i$  for which  $Q(R_i) = \text{FALSE}$ .
  - When no further splitting is possible, merge any adjacent regions  $R_i$  and  $R_k$  for which  $Q(R_i \cup R_k) = \text{TRUE}$ .
  - Stop when no further merging is possible.
- A maximum quadregion size is specified beyond which no further splitting is carried out.

	$R_1$	$R_2$
$R_3$	$R_{41}$	$R_{42}$
	$R_{43}$	$R_{44}$



# Region Splitting and Merging

Characteristics of the region of interest:

- Standard deviation greater than the background (which is near zero) and the central region (which is smoother).
- Mean value greater than the mean of background and less than the mean of the central region.
- Predicate:

$$Q = \begin{cases} \text{true} & \sigma > \alpha \text{ AND } 0 < m < b \\ \text{false} & \text{otherwise} \end{cases}$$

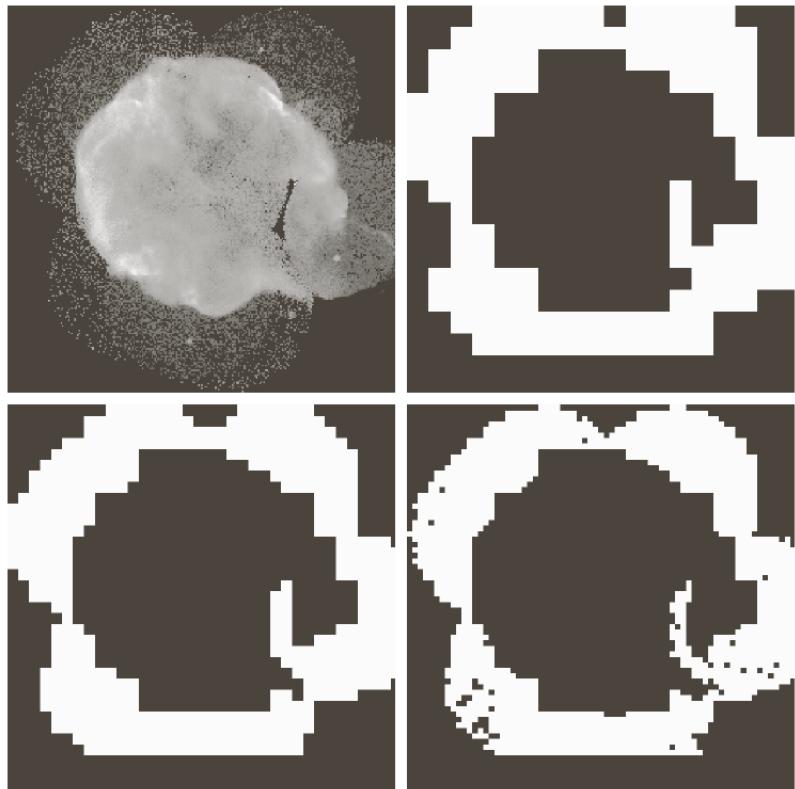
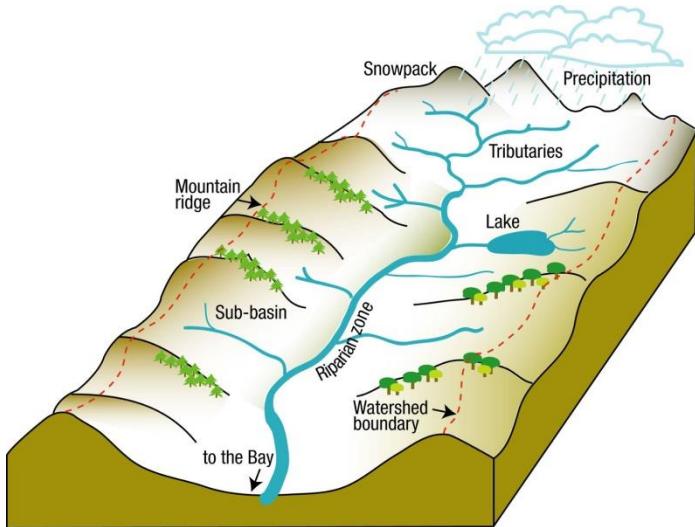


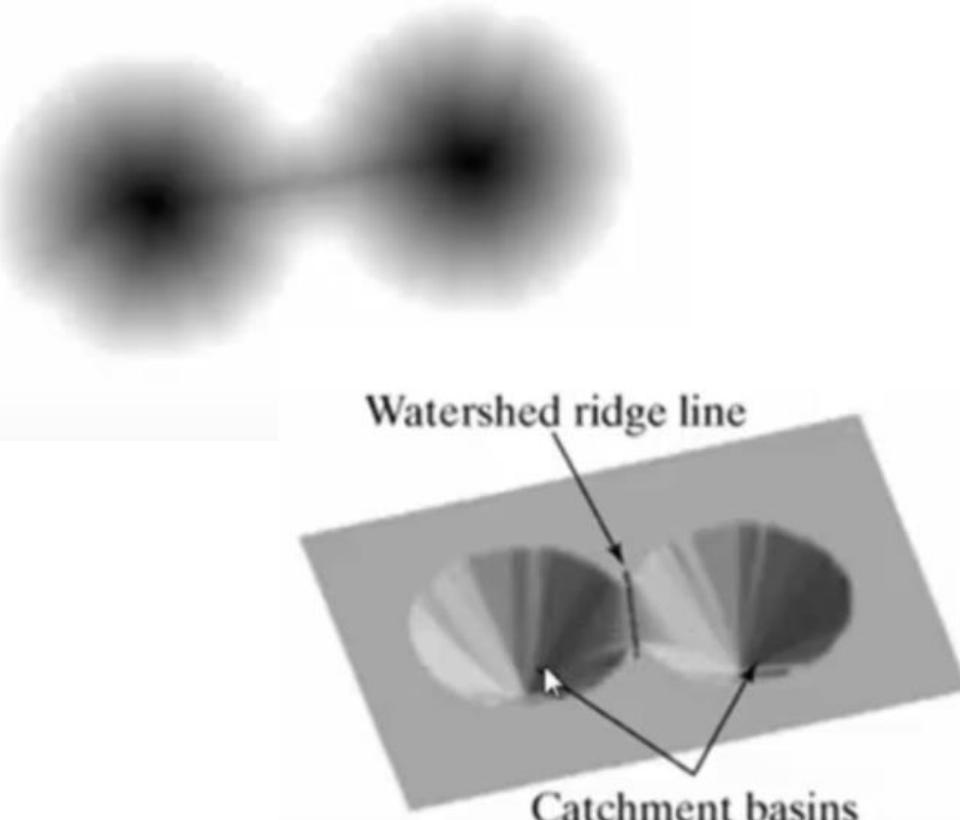
Image of the Cygnus Loop. We want to segment the outer ring of less dense matter

# Watershed Transformation

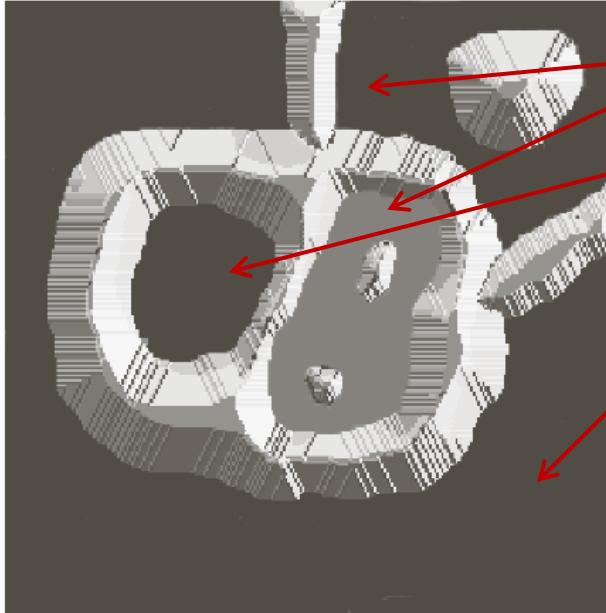
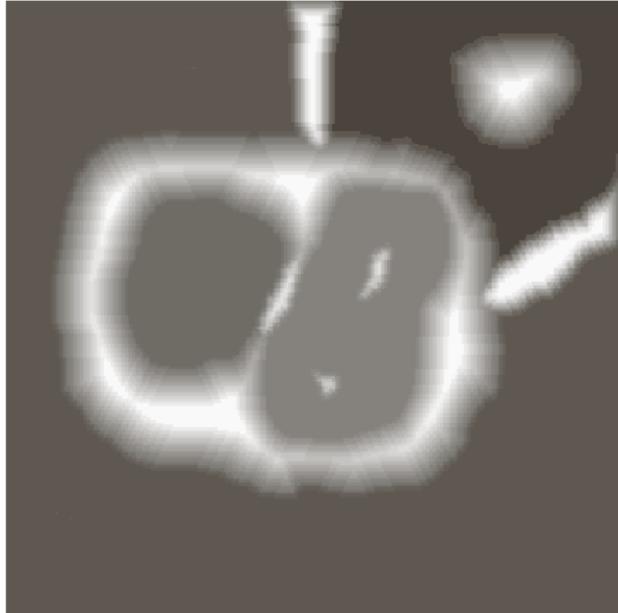
Watershed is a transformation on grayscale images. The aim of this technique is to segment the image, typically when two regions-of-interest are close to each other—i.e, their edges touch.



A geographical watershed



# Visualising Watershed

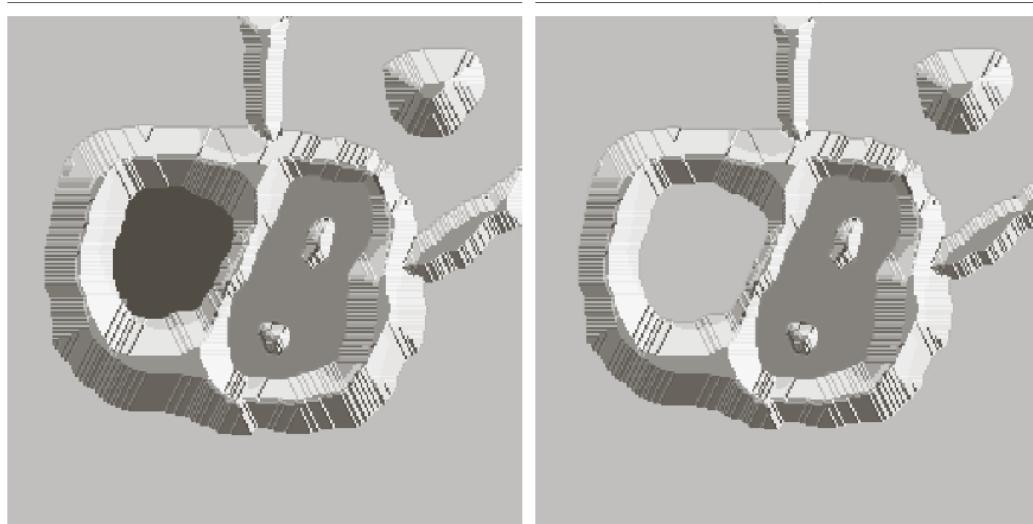


Regional minima

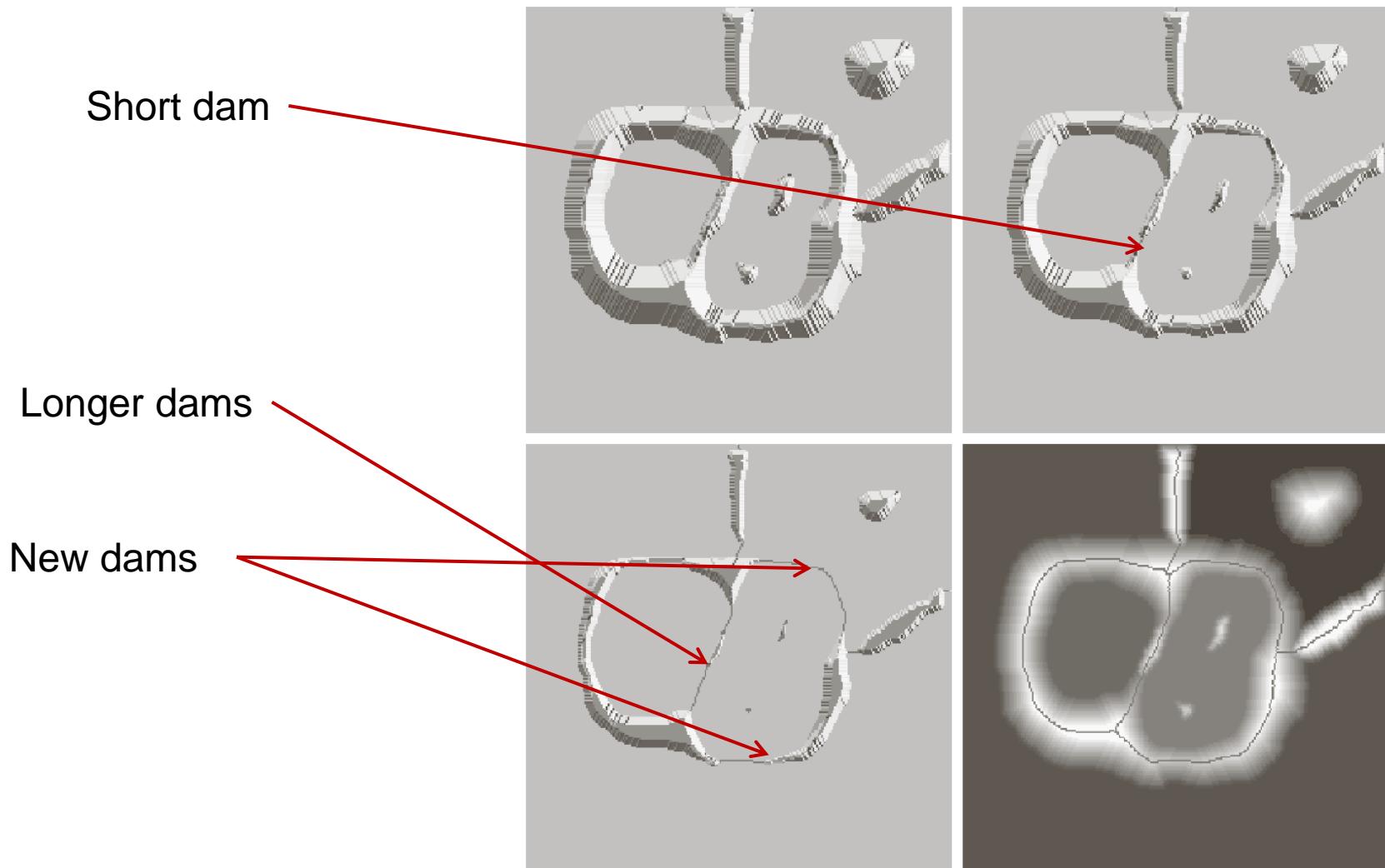
- Image data may be interpreted as a topographic surface.
- The height is proportional to the image intensity.
- Backsides of structures are shaded for better visualization.

# Watershed by flooding

- A hole is punched in each regional minimum and the topography is flooded by water from below through the holes.
- When the rising water is about to merge in catchment basins, a dam (barriers in the form of pixels) is built to prevent merging.
- At the max level of flooding only the tops of the dams will be visible. These continuous and connected boundaries act as partition and image is said to be segmentation.

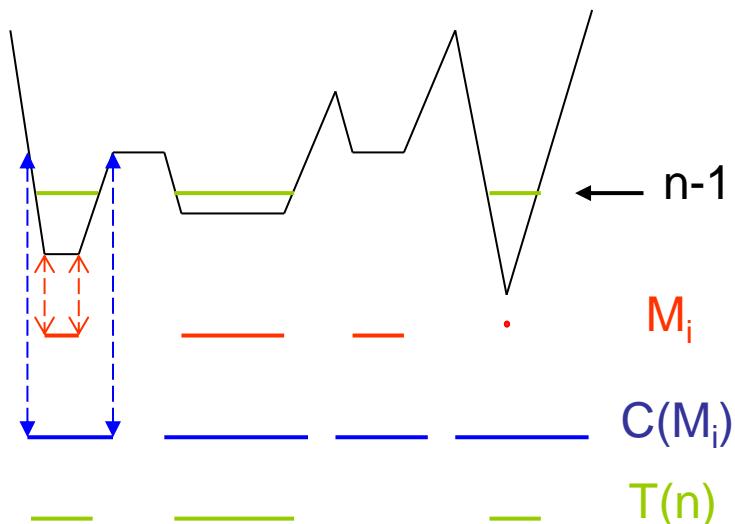


# Morphological Watershed



# Watershed Transform

- Denote  $M_1, M_2, \dots, M_R$  as the sets of the coordinates of the points in the regional minima of an image  $g(x,y)$
- Denote  $C(M_i)$  as the coordinates of the points in the catchment basin associated with regional minimum  $M_i$ .
- Denote  $T[n]$  as the set of coordinates  $(s,t)$  for which  $g(s,t) < n$
- Flood the topography in integer flood increments from  $n=\min+1$  to  $n=\max+1$
- At each flooding, the topography is viewed as a binary image



# Watershed Transform

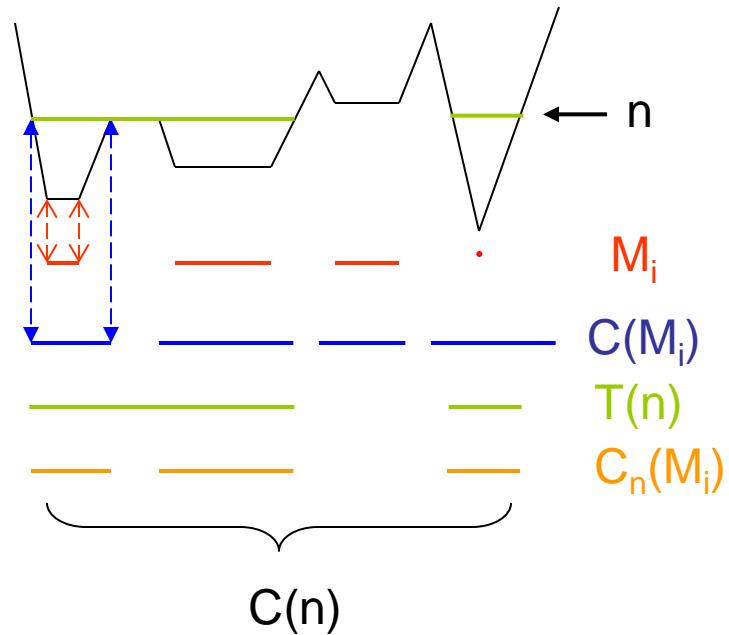
- Denote  $C_n(M_i)$  as the set of coordinates of points in the catchment basin associated with minimum  $M_i$  at flooding stage  $n$ .

- $C_n(M_i) = C(M_i) \cap T[n]$
  - $C_n(M_i) \subseteq T[n]$

- Denote  $C[n]$  as the union of the flooded catchment basin portions at stage  $n$ :

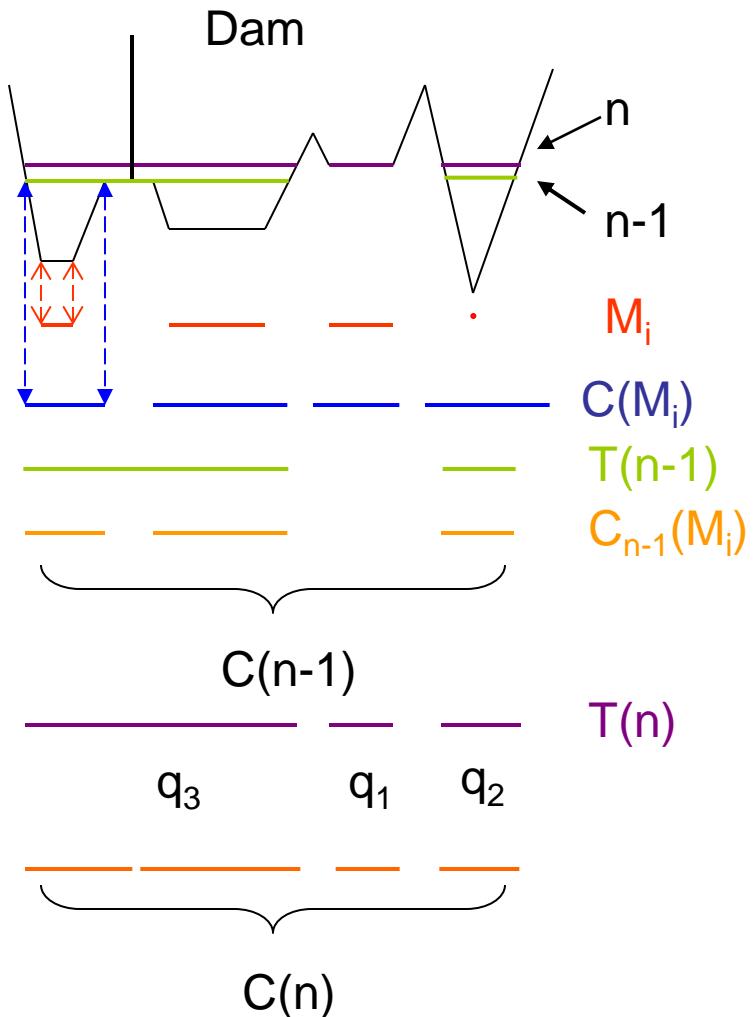
$$C[n] = \bigcup_{i=1}^R C_n(M_i) \text{ and } C[\max + 1] = \bigcup_{i=1}^R C(M_i)$$

- Initialization
  - Let  $C[min+1] = T[min+1]$
- At each step  $n$ , assume  $C[n-1]$  has been constructed. The goal is to obtain  $C[n]$  from  $C[n-1]$

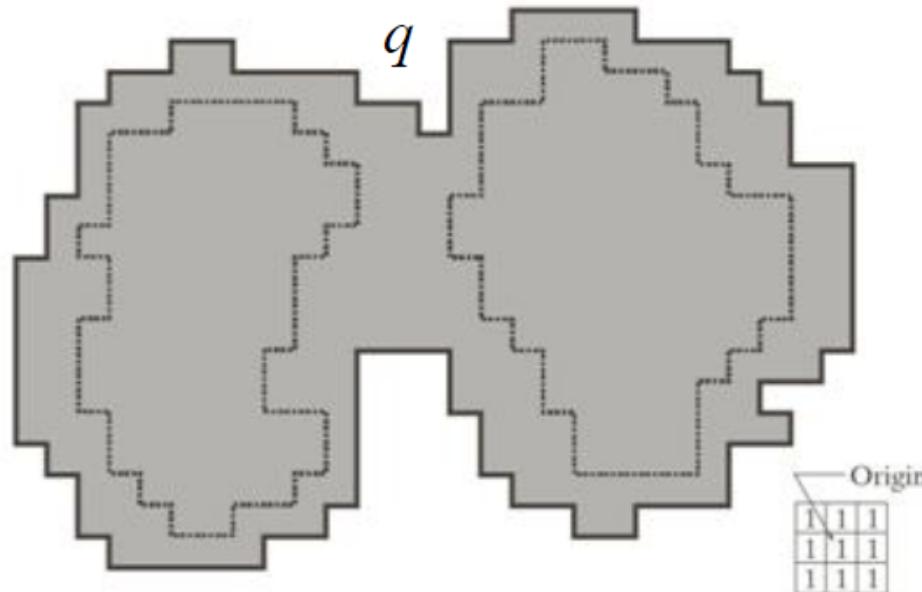


# Watershed Transform

- Denote  $Q[n]$  as the set of connected components in  $T[n]$ .
- For each  $q \in Q[n]$ , there are three possibilities
  - $q \cap C[n-1]$  is empty ( $q_1$ )
    - A new minimum is encountered
    - $q$  is incorporated into  $C[n-1]$  to form  $C[n]$
  - $q \cap C[n-1]$  contains one connected component of  $C[n-1]$  ( $q_2$ )
    - $q$  is incorporated into  $C[n-1]$  to form  $C[n]$
  - $q \cap C[n-1]$  contains more than one connected components of  $C[n-1]$  ( $q_3$ )
    - A ridge separating two or more catchment basins has been encountered
    - A dam has to be built within  $q$  to prevent overflow between the catchment basins
- Repeat the procedure until  $n=\max+1$

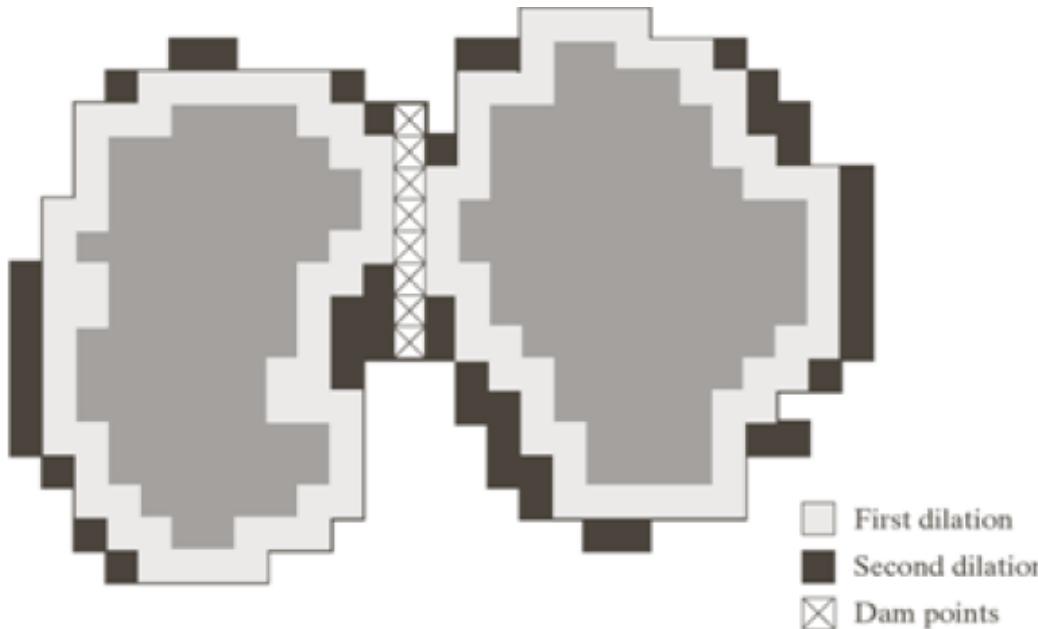


# Dam Construction



- Each of the connected components is dilated by the SE shown, subject to:
  1. The center of the SE has to be contained in  $q$ .
  2. The dilation cannot be performed on points that would cause the sets being dilated to merge.

# Dam Construction



## Conditions

1. Center of SE in  $q$ .
2. No dilation if merging.

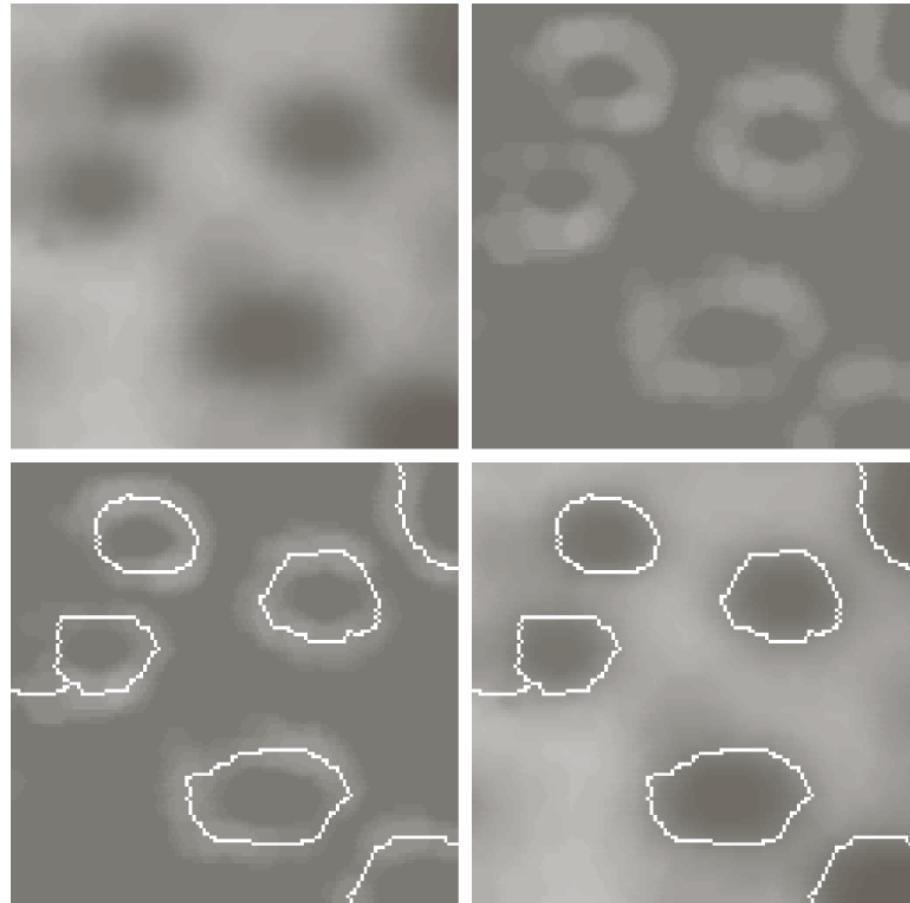
- In the first dilation, condition 1 was satisfied by every point and condition 2 did not apply to any point.
- In the second dilation, several points failed condition 1 while meeting condition 2 (the points in the perimeter which is broken).

# Application

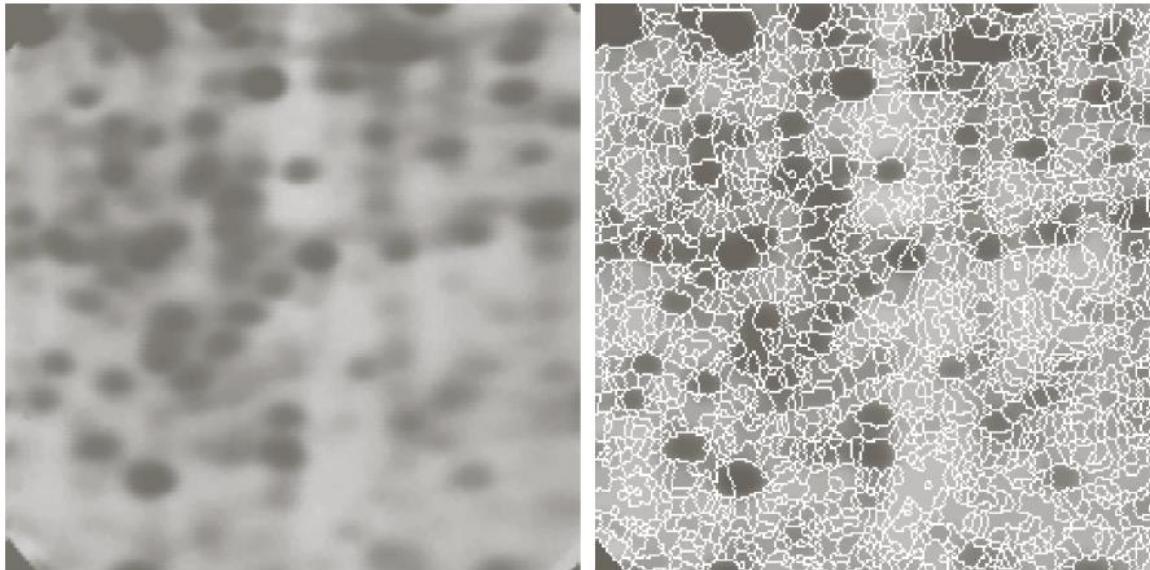
- A common application is the extraction of nearly uniform, blob-like objects from their background.
- For this reason it is generally applied to the gradient of the image and the catchment basins correspond to the blob like objects.

a	b
c	d

**FIGURE 10.56**  
(a) Image of blobs.  
(b) Image gradient.  
(c) Watershed lines.  
(d) Watershed lines superimposed on original image.



# Problems

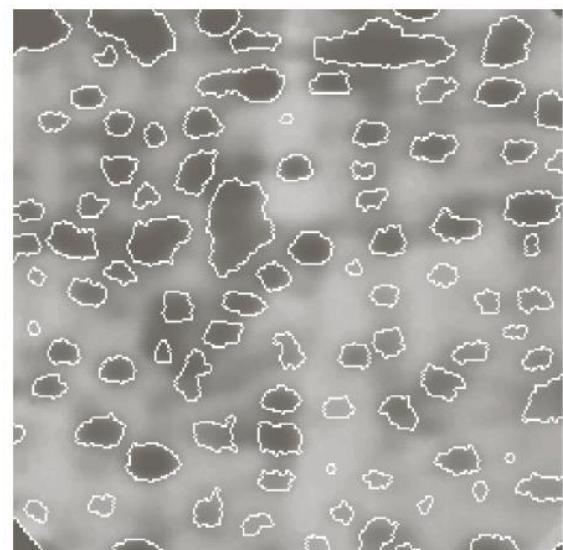
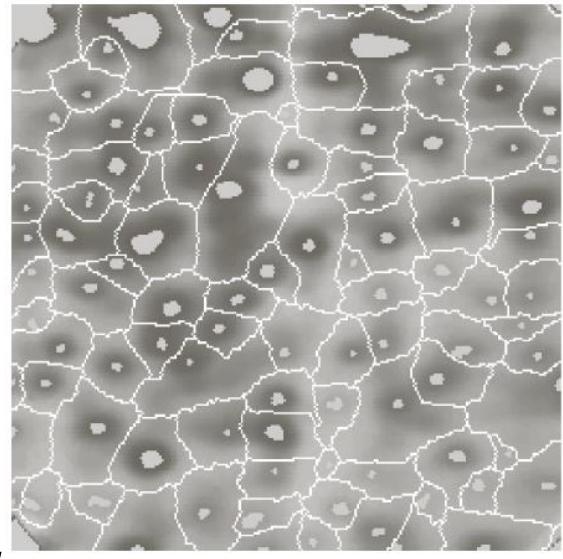


- Noise and local minima lead generally to *oversegmentation*.
- The result is not useful.
- Solution: limit the number of allowable regions by additional knowledge.

# Use of Markers

## Markers (connected components):

- *internal*, associated with the objects
- *external*, associated with the background.
- Here the problem is the large number of local minima.
- Smoothing may eliminate them.
- Define an *internal marker* (*after smoothing*):
  - Region surrounded by points of higher altitude.
  - They form connected components.
  - All points in the connected component have the same intensity.



# Image and Video Processing

Description & Representation

# Region description

- After segmentation, we get pixels along the boundary or pixels contained in a region.
- The next stage of the processing scheme is to extract characteristic information from these regions, that can be subsequently used to assign an identity to the region (by comparing with the information derived from known regions).

# Region description (contd.)

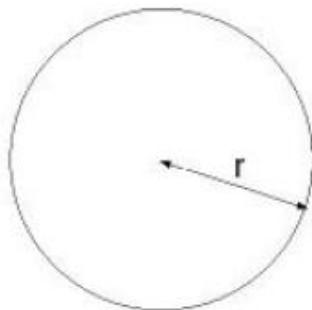
- The method chosen to describe a region will be closely coupled to the problem being solved.
- Ask following questions:
  - What problem am I trying to solve?
  - What information do I need to solve this problem?
  - How do I find that information from the image?

# Region Descriptors

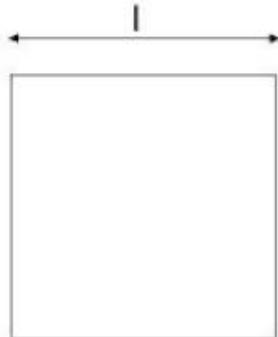
- Simple descriptors:
  - Area: # of pixels in the region (can compute histogram of labelled image)
  - Perimeter: length of its boundary (how to derive it ? )
  - Compactness:  $(\text{perimeter})^2/\text{area}$
  - Diameter, Major/Minor axis, orientation etc.
  - Mean/Median, Min/Max of the intensity levels

# Question

Q: What shape gives the minimal compactness ?



$$\text{compactness} = 4\pi$$



$$\text{compactness} = 16$$



$$\text{compactness} = \frac{4(a+b)^2}{ab}$$

# Region Representation

- Representing a region involves two choices:
  - Using external characteristics, e.g. boundary
  - Using internal characteristics, e.g. texture
- Common external representation methods are:
  - Chain code
  - Polygonal approximation
  - Signature
  - Boundary segments
  - Skeleton (medial axis)

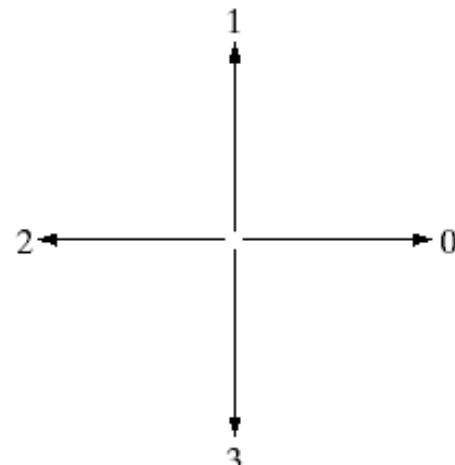
# Method 1: Chain Codes

- Represent a boundary by a connected sequence of straight-line segments of specified length and direction.
- Directions are coded using the numbering scheme

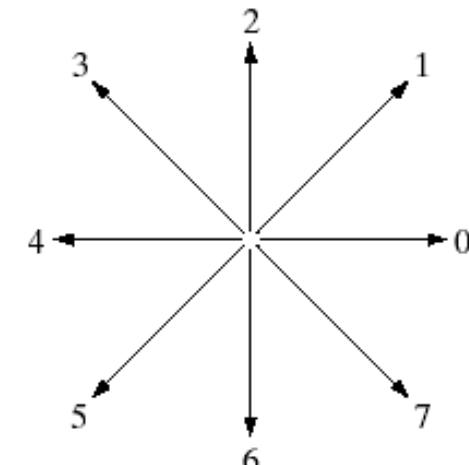
a b

**FIGURE 11.1**

Direction numbers for  
(a) 4-directional  
chain code, and  
(b) 8-directional  
chain code.



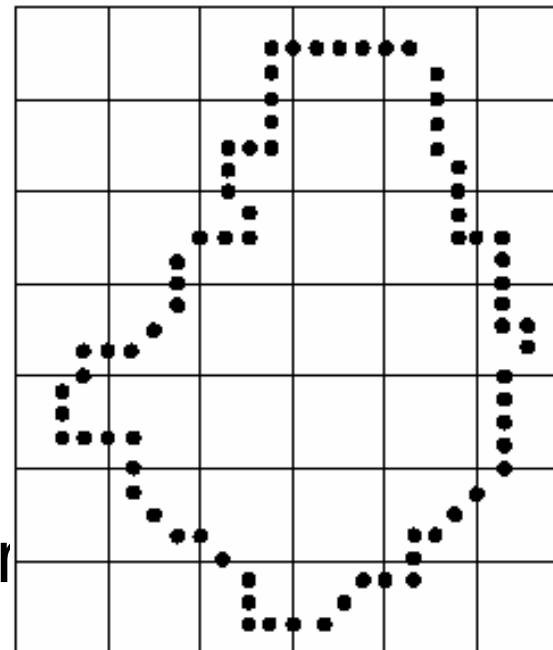
4-connectivity



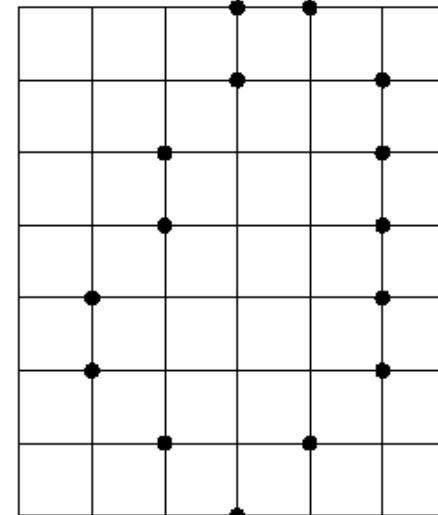
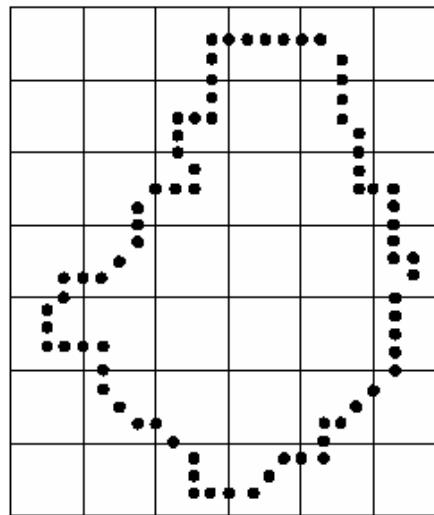
8 connectivity

# Generation of Chain Codes

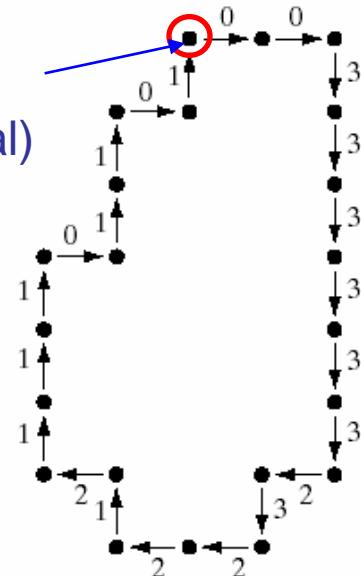
- Walking along the boundary in clockwise direction, for every pair of pixels assign the direction.
- Problems:
  - Long chain
  - Sensitive to noise
- Remedies?
  - Resampling using larger grid spacing.



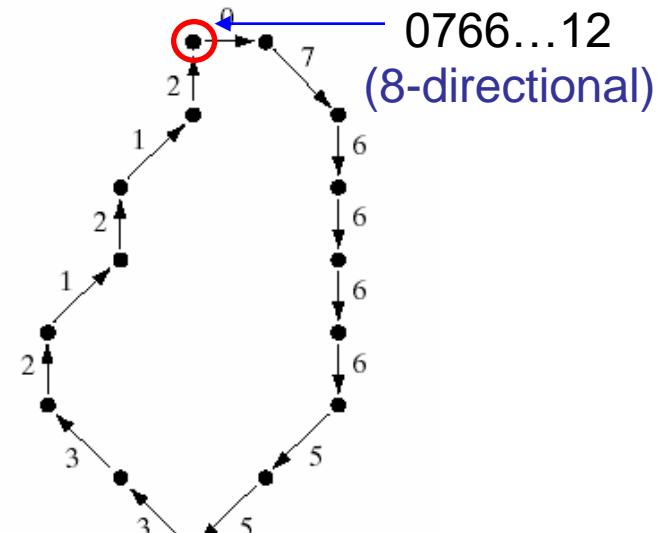
# Resampling for Chain Codes



0033...01  
(4-directional)



Spacing of the sampling grid affect the accuracy of the representation



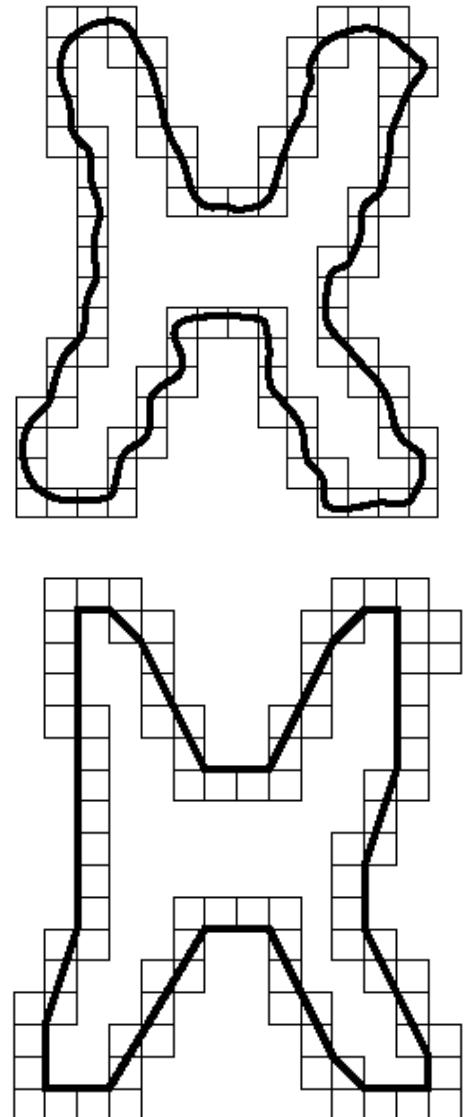
0766...12  
(8-directional)

# Normalization for Chain Codes

- With respect to starting point:
  - Make the chain code a circular sequence
  - Redefine the starting point which gives an integer of minimum magnitude
  - E.g. 101003333222 normalized to 003333222101
- For rotation:
  - Using first difference (FD) obtained by counting the number of direction changes in counterclockwise direction
  - E.g FD of a 4-direction chain code 10103322 is 3133030
- For scaling:
  - Altering the size of the resampling grid.

# Method 2: Polygonal Approximation

- Major disadvantage of chain codes is that the noise directly affects the code.
- Replace approximately linear segments of the boundary by straight line segments.
- The boundary of an object is thereby reduced to a polygon.
- Goal: to capture the “essence” of the boundary shape with the fewest possible polygonal segments.

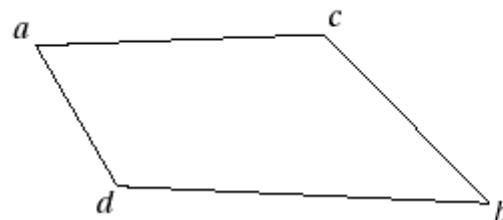
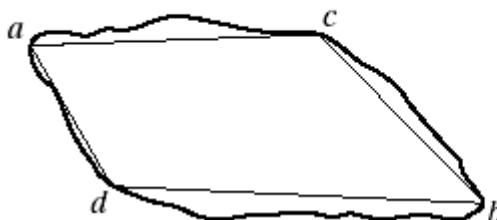
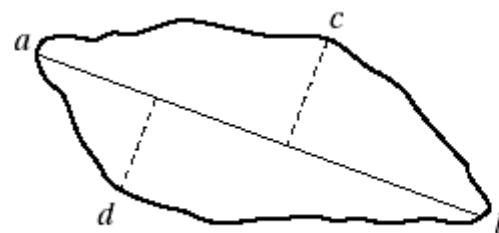


# Merging Technique

- Repeat the following steps:
  - Merging unprocessed points along the boundary until the least-square error line fit exceed the threshold.
  - Store the parameters of the line
  - Reset the LS error to 0
- Intersections of adjacent line segments form vertices of the polygon.
- Problem: vertices of the polygon do not always correspond to inflections (e.g. corners) in the original boundary.<sub>12</sub>

# Splitting Technique

- Subdivide a segment successively into two parts until a specified criterion is met.
- For example

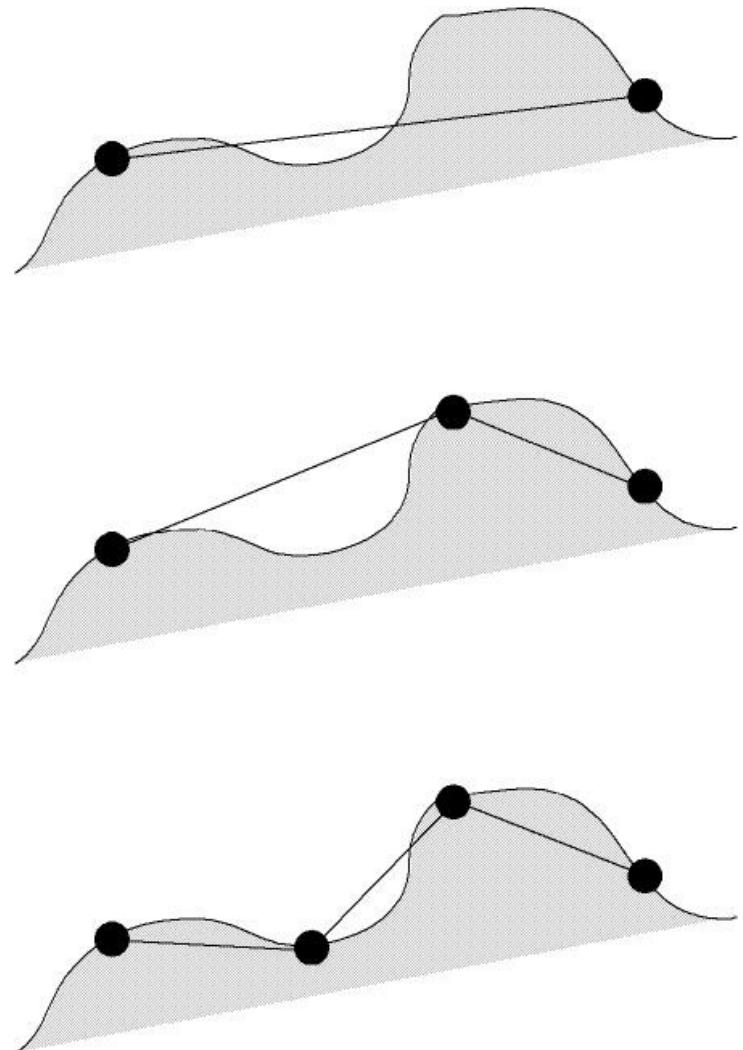


a	b
c	d

**FIGURE 11.4**  
(a) Original boundary.  
(b) Boundary divided into segments based on extreme points. (c) Joining of vertices.  
(d) Resulting polygon.

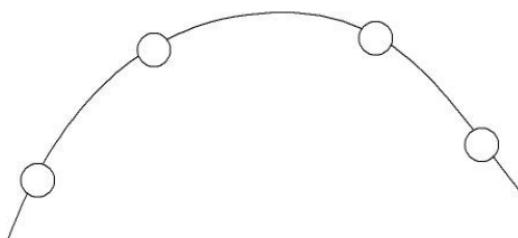
# Polyline splitting

- Firstly compute the distance from each point onto the line.
- The maximum distance is found, if it exceeds some threshold, then a vertex is inserted to bisect the original line.
- The process is then repeated for the newly formed segments.



# Spline curves

- If any portion of the boundary is curved, then polygon representation becomes inefficient, i.e. many small segments are generated.
- Inserting curved sections instead of linear ones may ameliorate this situation
- Splines provide a compact representation of curves using two end points and two intermediate control points.
- Adjacent curve sections will share end points to ensure continuity,



# Method 3: Signature

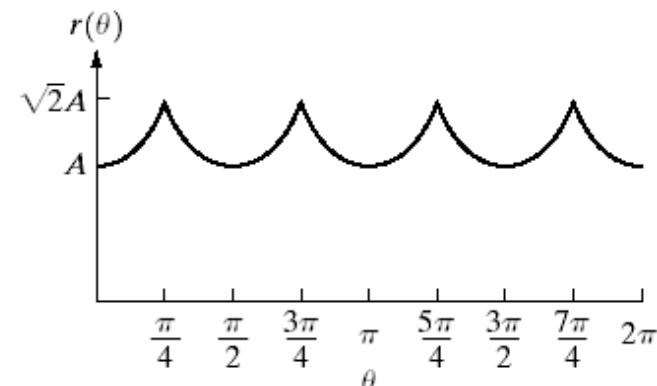
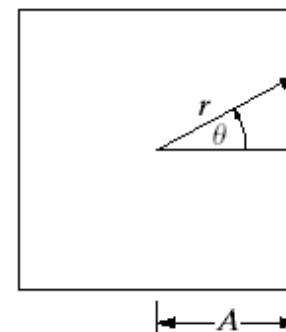
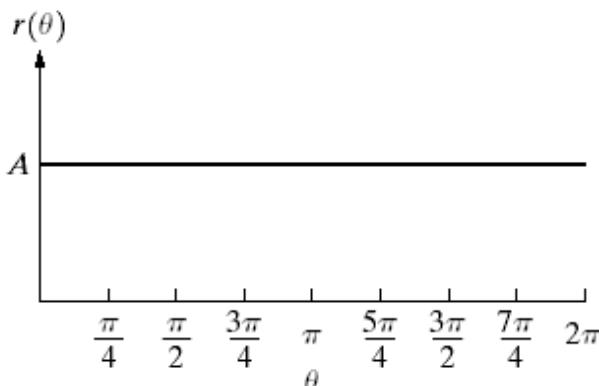
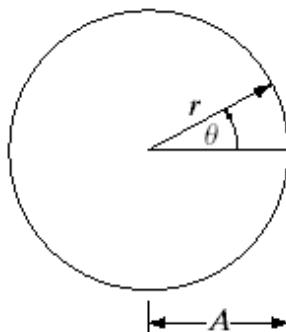
- Reduce the boundary representation to a 1D function by creating a plot of distances from a point to the border as a radius is rotated about the point.

a b

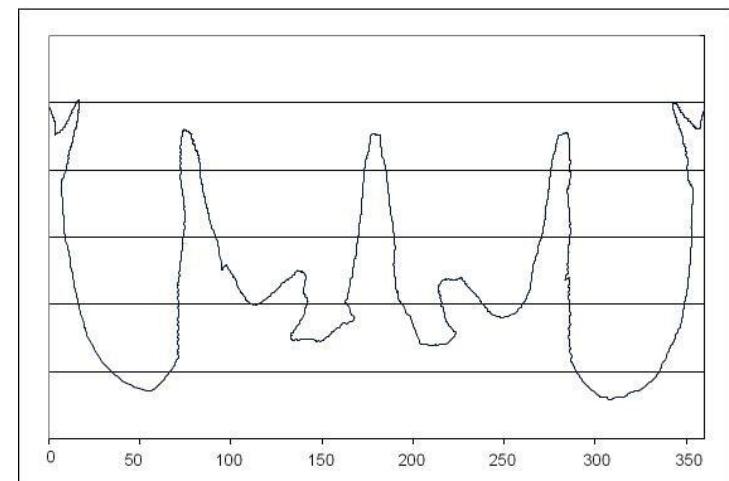
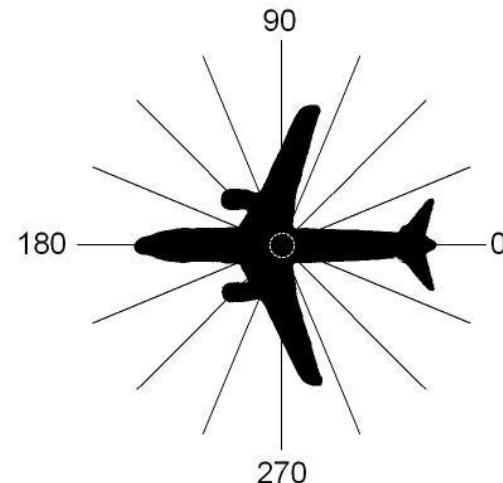
**FIGURE 11.5**

Distance-versus-angle signatures.  
In (a)  $r(\theta)$  is constant. In (b),  
the signature consists of  
repetitions of the pattern

$r(\theta) = A \sec \theta$  for  
 $0 \leq \theta \leq \pi/4$  and  
 $r(\theta) = A \csc \theta$  for  
 $\pi/4 < \theta \leq \pi/2$ .



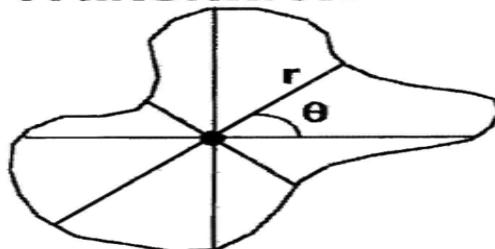
- Real objects will generate more complex curves, but they may still be recognisable,
- Recognition will be achieved by comparison of the curve with the curve derived from known objects.



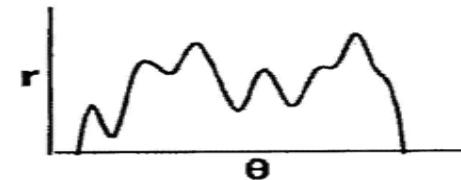
# Fourier Descriptor

Taking the dist. vs. angle one step further we get ...

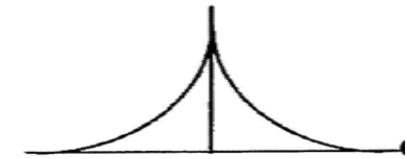
**Translation**



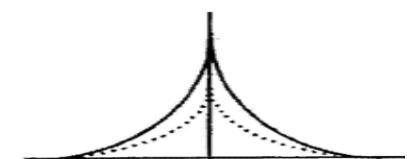
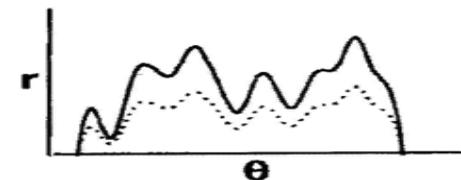
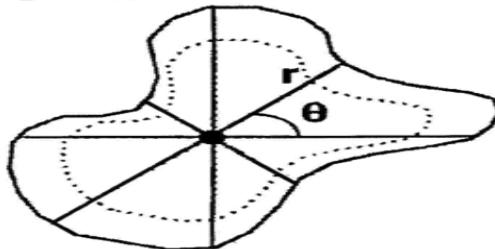
**Boundary Representation**



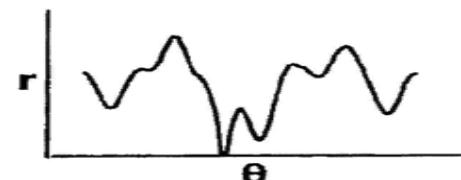
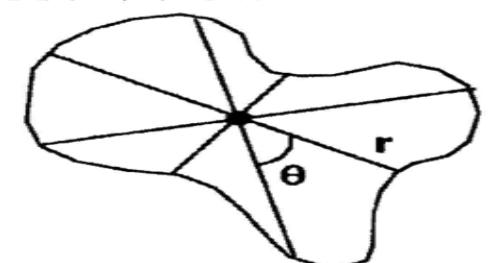
**Fourier Transform**

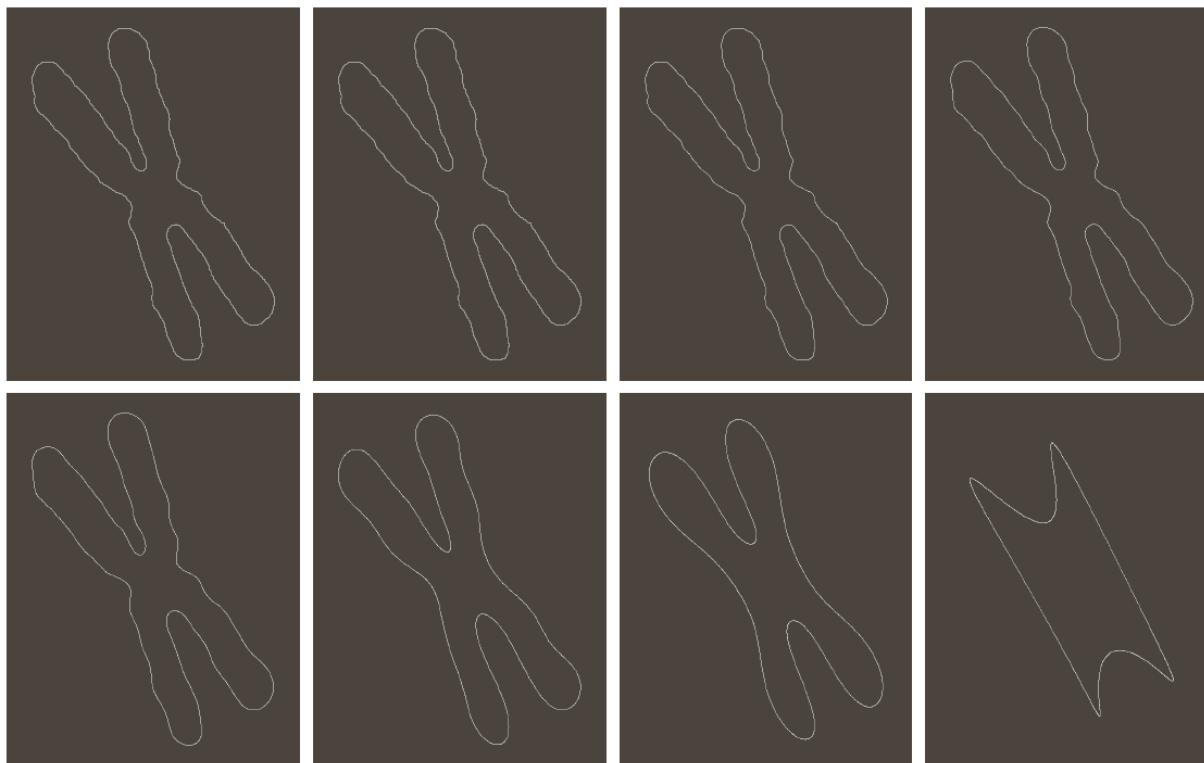


**Scale**



**Rotation**



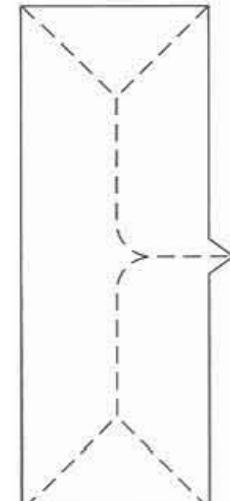
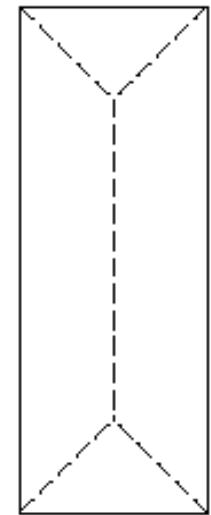


a	b	c	d
e	f	g	h

**FIGURE 11.20** (a) Boundary of human chromosome (2868 points). (b)–(h) Boundaries reconstructed using 1434, 286, 144, 72, 36, 18, and 8 Fourier descriptors, respectively. These numbers are approximately 50%, 10%, 5%, 2.5%, 1.25%, 0.63%, and 0.28% of 2868, respectively.

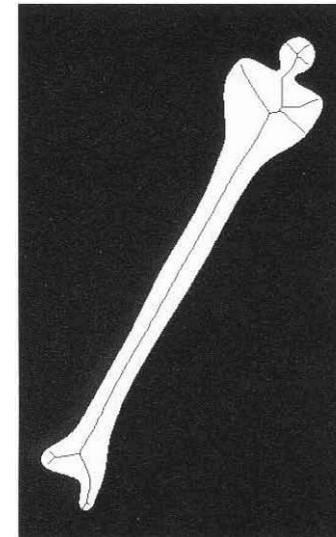
# Skeletons

- Obtaining the skeleton of the region via thinning.
  - By morphology: no provision for keeping the skeleton connected
  - By Medial Axis Transformation (MAT): The MAT of region R with border B is found as:
    - For each point p in R, we find its closest neighbor in B.
    - If p has more than one, it belongs to the Medial Axis (skeleton) of R.



# Skeletons

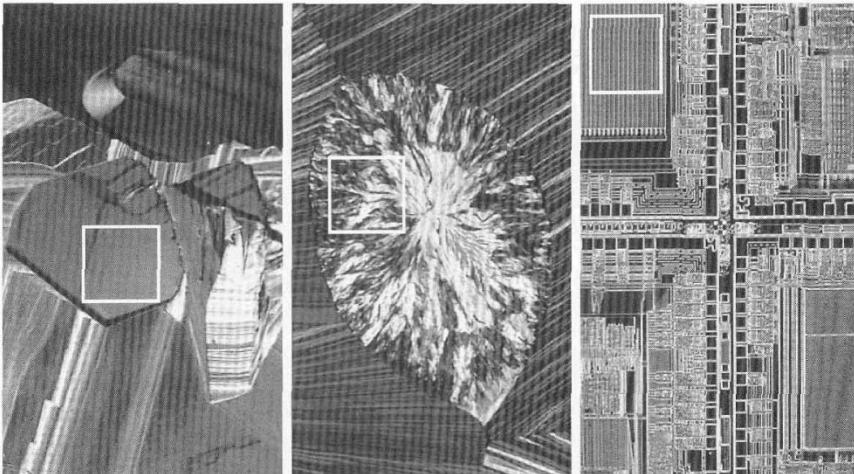
- To improve computational efficiency, edge points of a region are iteratively deleted if:
  - End points are not deleted
  - Connectedness is not broken
  - No excessive erosion is caused
- Step1:
  - (a)  $2 \leq N(p_1) \leq 6$
  - (b)  $T(p_1) = 1$
  - (c)  $p_2 \cdot p_4 \cdot p_6 = 0$
  - (d)  $p_4 \cdot p_6 \cdot p_8 = 0$
- Step2:
  - (c')  $p_2 \cdot p_4 \cdot p_8 = 0$
  - (d')  $p_2 \cdot p_6 \cdot p_8 = 0$



$p_9$	$p_2$	$p_3$	0	0	1
$p_8$	$p_1$	$p_4$	0	$p_1$	1
$p_7$	$p_6$	$p_5$	0	0	1

# Statistical Moments for Texture

- Statistical approaches
  - smooth, coarse, regular



- nth moment:

$$u_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i)$$

$$m = \sum_{i=0}^{L-1} z_i p(z_i)$$

- 2th moment:

- is a measure of gray level contrast(relative smoothness)

- 3th moment:

- is a measure of the skewness of the histogram

- 4th moment:

- is a measure of its relative flatness

- 5th and higher moments:

- are not so easily related to histogram shape

# Moment Invariant

2D Moment of order (p+q)

$$m_{pq}(S) = \sum_{(x,y) \in S} x^p y^q f(x, y)$$

$$M_{H1} = \eta_{20} + \eta_{02}$$

$$M_{H2} = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$M_{H3} = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$M_{H4} = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\begin{aligned} M_{H5} &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[(\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2] \end{aligned}$$

$$\begin{aligned} M_{H6} &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ &\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \end{aligned}$$

$$\begin{aligned} M_{H7} &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned}$$

Central moment

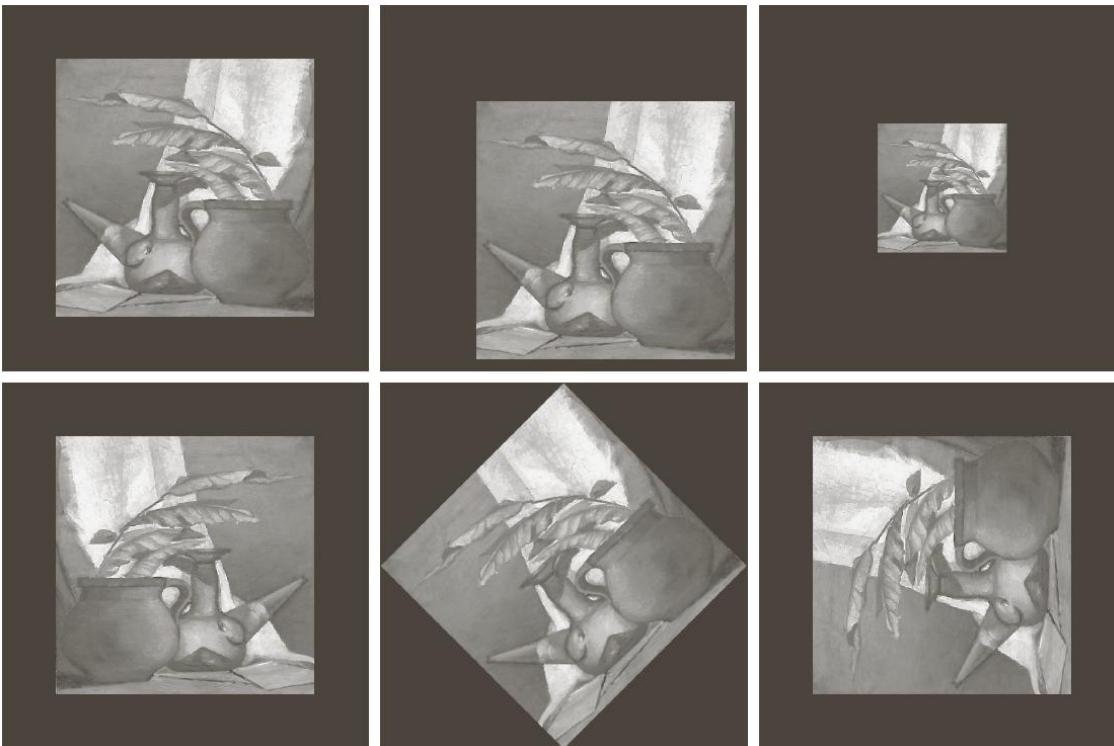
$$\mu_{pq}(S) = \sum_{(x,y) \in S} (x - \bar{x})^p (y - \bar{y})^q f(x, y), \text{ where}$$

$$\bar{x} = \frac{m_{10}(S)}{m_{00}(S)}, \bar{y} = \frac{m_{01}(S)}{m_{00}(S)}$$

Normalized central moment

$$\eta_{pq}(S) = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \gamma = \frac{p+q}{2} + 1, p+q = 2, 3, \dots$$

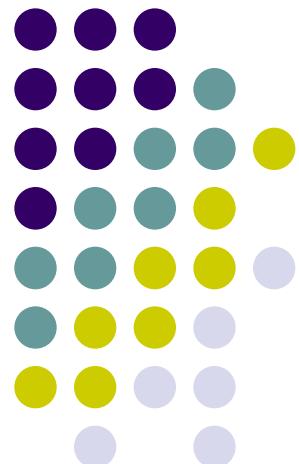
# Moment Invariant



Moment Invariant	Original Image	Translated	Half Size	Mirrored	Rotated 45°	Rotated 90°
$\phi_1$	2.8662	2.8662	2.8664	2.8662	2.8661	2.8662
$\phi_2$	7.1265	7.1265	7.1257	7.1265	7.1266	7.1265
$\phi_3$	10.4109	10.4109	10.4047	10.4109	10.4115	10.4109
$\phi_4$	10.3742	10.3742	10.3719	10.3742	10.3742	10.3742
$\phi_5$	21.3674	21.3674	21.3924	21.3674	21.3663	21.3674
$\phi_6$	13.9417	13.9417	13.9383	13.9417	13.9417	13.9417
$\phi_7$	-20.7809	-20.7809	-20.7724	20.7809	-20.7813	-20.7809

- Common external representation methods are:
  - Chain code
  - Polygonal approximation
  - Signature
  - skeleton
- Descriptors
  - Boundary descriptor
  - Region descriptor

# Motion based Segmentation in videos





# Visual motion

- Powerful cue used by humans to extract objects and regions.
- Motion arises from
  - Objects moving in the scene.
  - Relative displacement between the sensing system and the scene





# Moving object segmentation

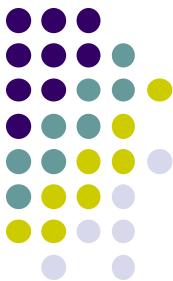
- Visual world is dynamic and we constantly come across video scenes with many moving objects like vehicle, pedestrians etc.
- Automated analysis of such dynamic video scene activities/events require 1) detection 2) classification 3) tracking
- Applications - Video surveillance, Traffic monitoring, Human action recognition, Human-computer interaction etc.



# Background Subtraction

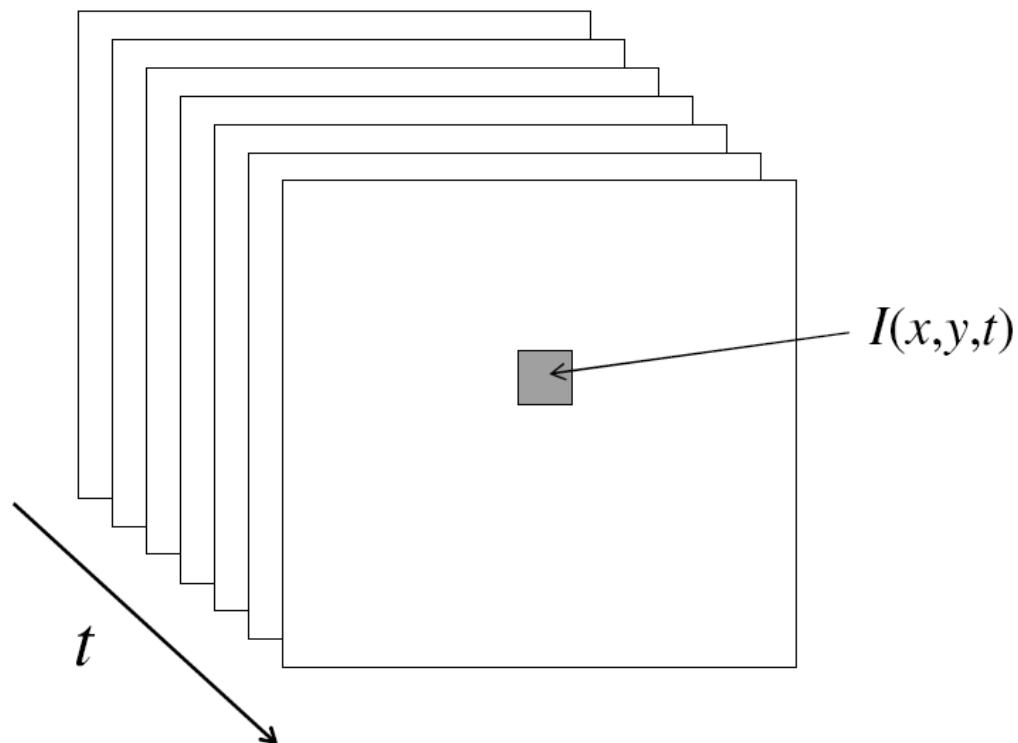
- Given a video frame from a static camera observing a scene, goal is to separate the moving foreground objects from the static *background* (*part of the scene that does not change position with time*)





# Video as an “Image Stack”

- A video is a sequence of frames captured over time
- Now our image data is a function of space ( $x, y$ ) and time ( $t$ )



# Simple Approach

Image at time  $t$ :

$$I(x, y, t)$$



Background at time  $t$ :

$$B(x, y, t)$$



$$| > Th$$

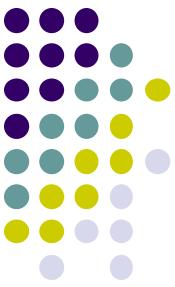
1. Estimate the background for time  $t$ .
2. Subtract the estimated background from the input frame.
3. Apply a threshold,  $Th$ , to the absolute difference to get the **foreground mask**.

But, how can we estimate the background?

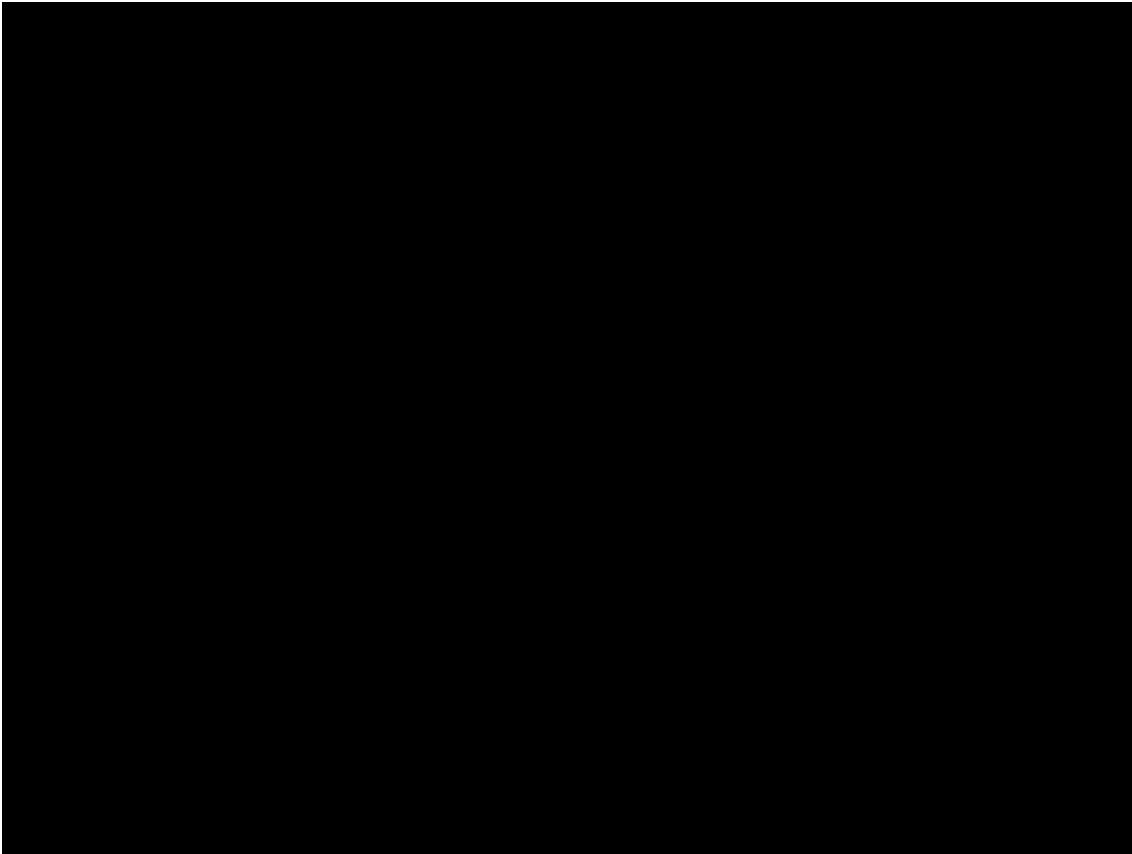


# Why Background modeling?

- Several factor cause change in the background like
  - Illumination changes
  - Natural wind motions causing background parts to move
  - Shadows, flicker etc.
  - Moving object itself may halt/become stationary (eg. car gets parked)



# Test Video



# Frame Differencing

- Background is estimated to be the previous frame.  
Background subtraction equation then becomes:

$$B(x, y, t) = I(x, y, t - 1)$$



$$|I(x, y, t) - I(x, y, t - 1)| > Th$$

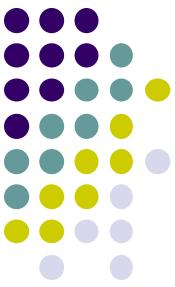
- Depending on the object structure, speed, frame rate and global threshold, this approach may or may **not** be useful (usually **not**).



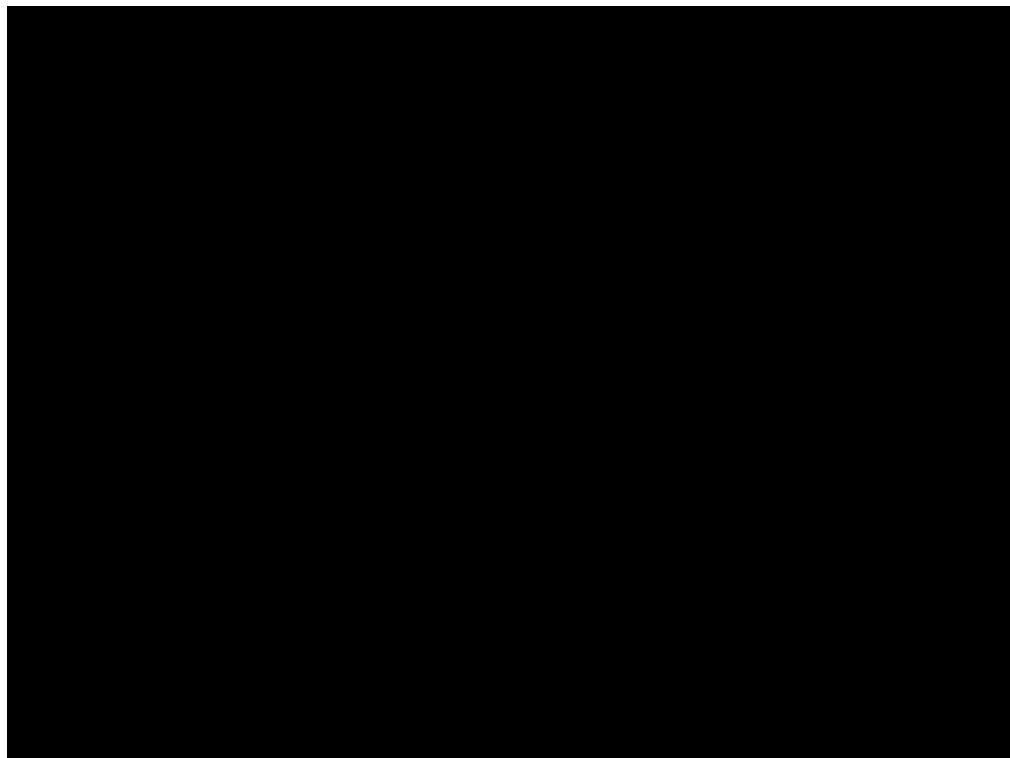
—

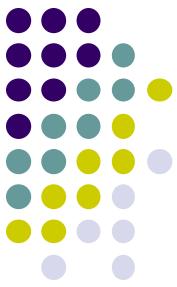


$$| > Th$$



# Result





# Frame Differencing

- Major flaws
  - For objects with uniformly distributed intensity values, the interior pixels are interpreted as part of the background.
  - Objects must be continuously moving otherwise it becomes part of the background.
- Two major advantages.
  - modest computational load.
  - background model is highly adaptive
- A challenge with this method is in determining the threshold value.

# Frame Differencing

$Th = 25$



$Th = 50$



$Th = 100$



$Th = 200$



The accuracy of this approach is dependent on object speed, frame rate, threshold value.



# Mean Filter

- ▶ In this case the background is the mean of the previous  $n$  frames:

$$B(x, y, t) = \frac{1}{n} \sum_{i=0}^{n-1} I(x, y, t - i)$$

↓

$$|I(x, y, t) - \frac{1}{n} \sum_{i=0}^{n-1} I(x, y, t - i)| > Th$$

- ▶ For  $n = 10$ :

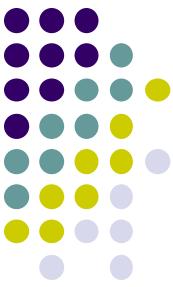
Estimated Background



Foreground Mask



where  $N$  is the number of preceding images taken for averaging. This averaging refers to averaging corresponding pixels in the given images.  $N$  would depend on the frame rate and the amount of movement in the video.<sup>3</sup>



# Mean Filter



- ▶ For  $n = 50$ :  
Estimated Background



Foreground Mask



# Median Filter

- ▶ Assuming that the background is more likely to appear in a scene, we can use the median of the previous  $n$  frames as the background model:

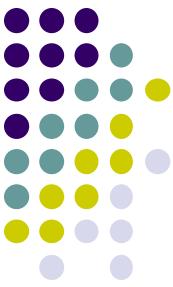
$$B(x, y, t) = \text{median}\{I(x, y, t - i)\}$$
$$\downarrow$$
$$|I(x, y, t) - \text{median}\{I(x, y, t - i)\}| > Th \text{ where}$$
$$i \in \{0, \dots, n - 1\}.$$

Estimated Background



Foreground Mask

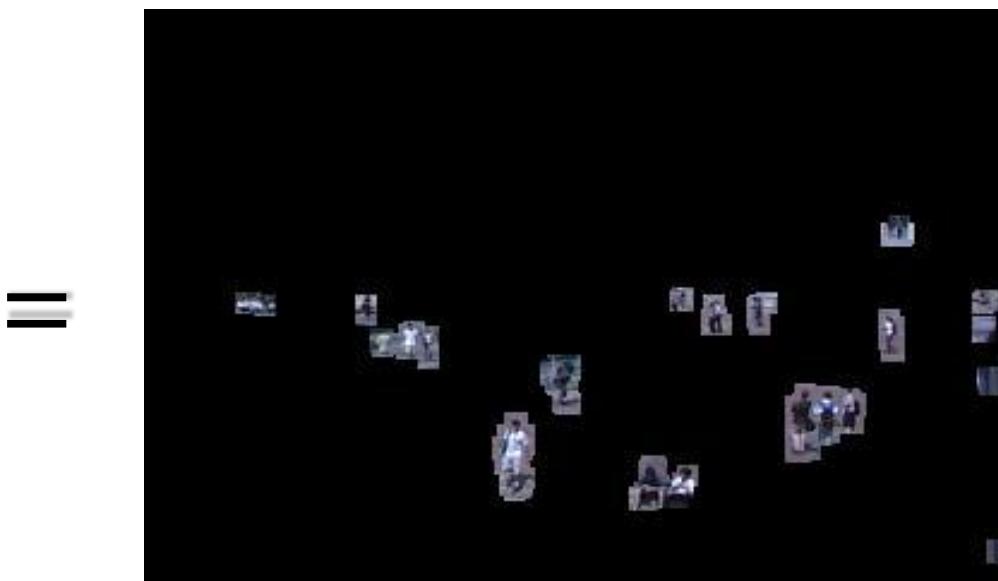




# Average/Median Image



# Background Subtraction

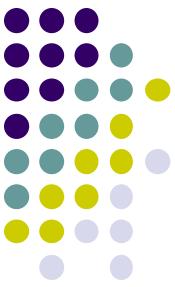




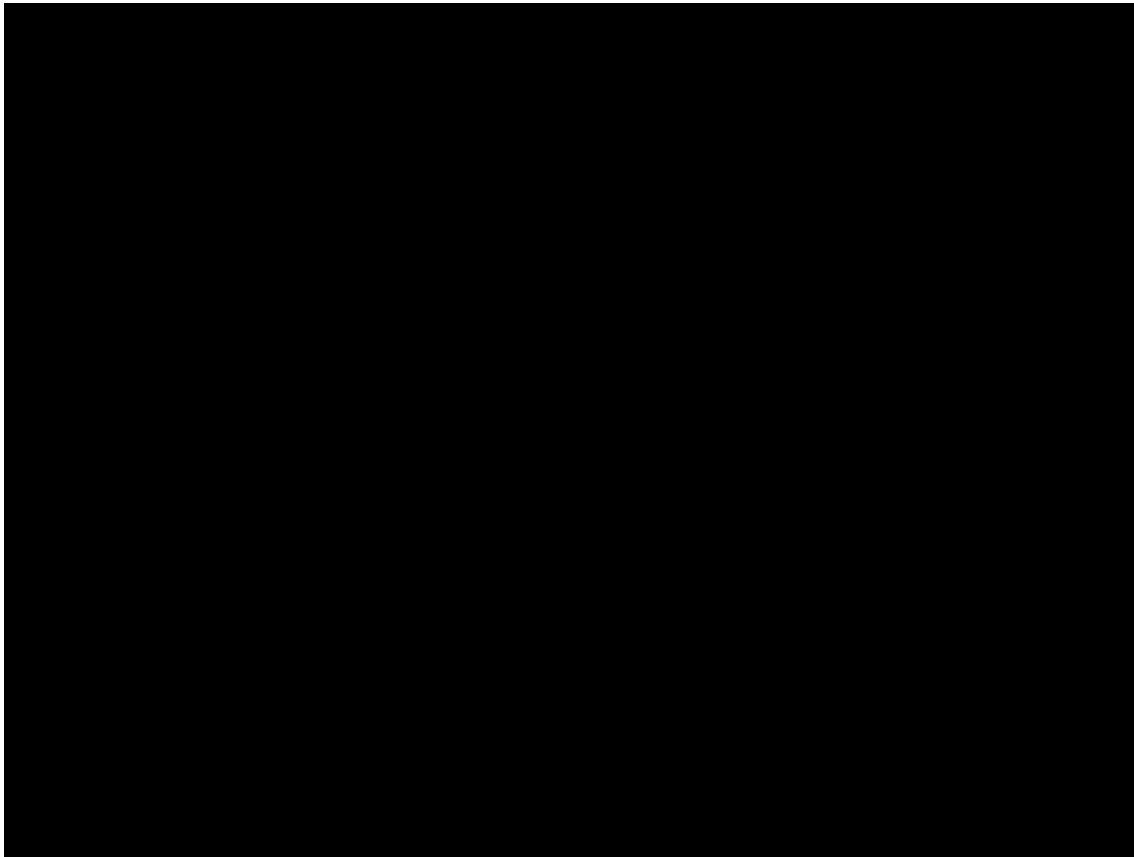
# Approximate Median Filter

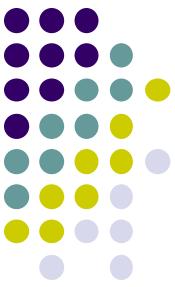
To overcome the large memory and computation requirement:

- If a pixel in the current frame has a value larger than the corresponding background pixel, the background pixel is incremented by 1.
- Likewise, if the current pixel is less than the background pixel, the background is decremented by one.
- The background eventually converges to an estimate where half the input pixels are greater than the background, and half are less than the background—approximately the median (convergence time will vary based on frame rate and amount movement in the scene.)

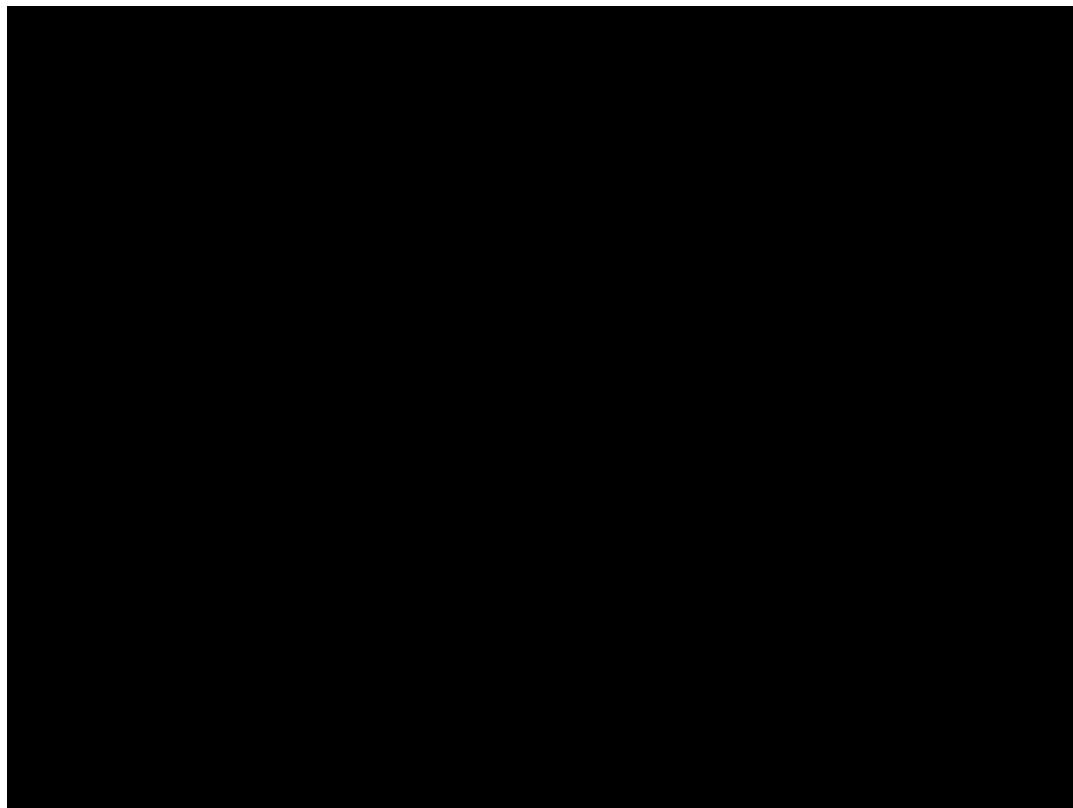


# Background model





# Results





# Main Shortcoming

## Disadvantages:

- ▶ There is **another** major problem with these simple approaches:

$$|I(x, y, t) - B(x, y, t)| > Th$$

1. There is one global threshold,  $Th$ , for all pixels in the image.
2. And even a bigger problem:

**this threshold is *not* a function of  $t$ .**

- ▶ So, these approaches will not give good results in the following conditions:

- ▶ if the background is bimodal,
- ▶ if the scene contains many, slowly moving objects (mean & median),
- ▶ if the objects are fast and frame rate is slow (frame differencing),
- ▶ and if general lighting conditions in the scene change with time!

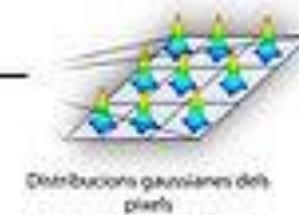


# Running Gaussian Average

- For every pixel, fit one Gaussian PDF distribution ( $\mu, \sigma$ ) on the most recent  $n$  frames (this gives the background PDF).
- To accommodate for change in background over time (e.g. due to illumination changes or non-static background objects), at every frame, every pixel's mean and variance must be updated.
- background PDF update (running average):

$$\mu_{t+1} = \alpha F_t + (1 - \alpha) \mu_t$$

$$\sigma_{t+1}^2 = \alpha(F_t - \mu_t)^2 + (1 - \alpha)\sigma_t^2$$



Distribuciones gaussianas del fondo



# Running Gaussian Average

- Initialization: For variance, we can, for example, use the variance in x and y from a small window around each pixel and take  $\mu_0 = I_0$
- We can now classify a pixel as background if its current intensity lies within some confidence interval of its distribution's mean.

$$\frac{|(I_t - \mu_t)|}{\sigma_t} > k \rightarrow \text{Foreground}$$

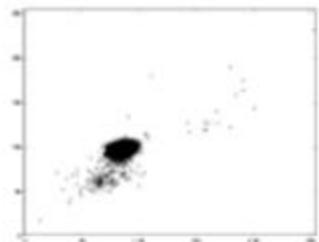
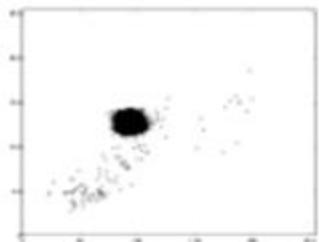
$$\frac{|(I_t - \mu_t)|}{\sigma_t} \leq k \rightarrow \text{Background}$$

- It does not however cope with multimodal background.

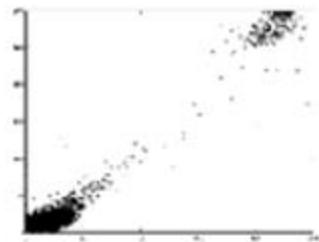


# Gaussian Mixture Model

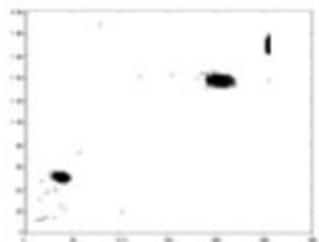
- A background pixel may belong to more than one pattern class each having a specific PDF



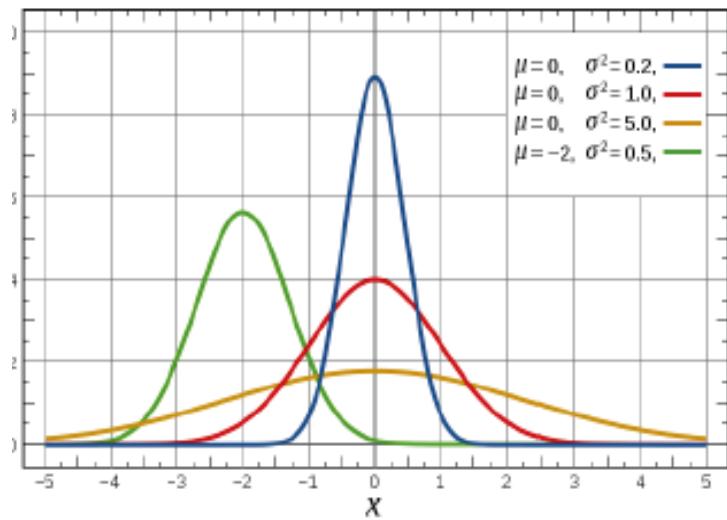
(a)



(b)



(c)





# Mixture model

- Model the history of a pixel distribution by a weighted combination of K adaptive gaussians to cope with multimodal background distributions;

$$p(x) = \sum_{j=1}^K w_j \cdot N(x | \mu_j, \Sigma_j) \quad \sum_{j=1}^K w_j = 1 \quad \text{and} \quad 0 \leq w_j \leq 1$$

- The parameters of Gaussians - weights, mean and variance. **How to learn/update the background model?**
- The mixture of Gaussians actually models both the foreground and the background: **how to pick only the distributions modeling the background?**

# Model Adaptation



- ▶ An on-line K-means approximation is used to update the Gaussians.
- ▶ If a new pixel value,  $X_{t+1}$ , can be matched to one of the existing Gaussians (within  $2.5\sigma$ ), that Gaussian's  $\mu_{i,t+1}$  and  $\sigma_{i,t+1}^2$  are updated as follows:

$$\mu_{i,t+1} = (1 - \rho)\mu_{i,t} + \rho X_{t+1}$$

and

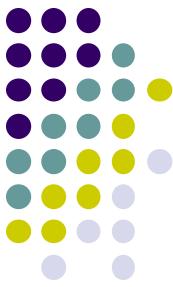
$$\sigma_{i,t+1}^2 = (1 - \rho)\sigma_{i,t}^2 + \rho(X_{t+1} - \mu_{i,t+1})^2$$

where  $\rho = \alpha \mathcal{N}(X_{t+1} | \mu_{i,t}, \sigma_{i,t}^2)$  and  $\alpha$  is a learning rate.

- ▶ Prior weights of all Gaussians are adjusted as follows:

$$\omega_{i,t+1} = (1 - \alpha)\omega_{i,t} + \alpha(M_{i,t+1})$$

where  $M_{i,t+1} = 1$  for the matching Gaussian and  $M_{i,t+1} = 0$  for all the others.



# Model Adaptation

- The mean and variance for unmatched distributions remain the same.
- If none of the K distributions match the current pixel value, the least probable distribution is discarded and
- A new distribution with the current value as its mean value, an initially high variance, and low prior weight, is added.

# Background Model Estimation

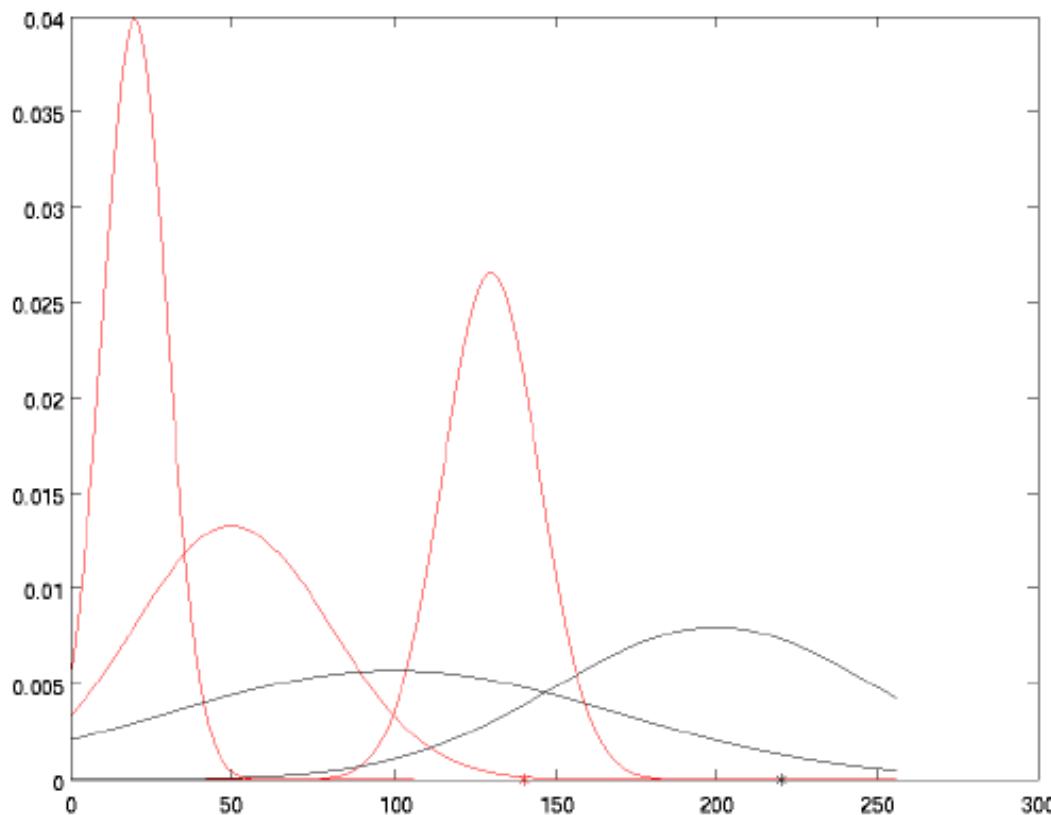
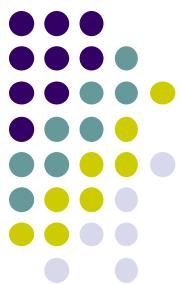


- ▶ Heuristic: the Gaussians with the **most supporting evidence** and **least variance** should correspond to the background (**Why?**).
- ▶ The Gaussians are ordered by the value of  $\omega/\sigma$  (high support & less variance will give a high value).
- ▶ Then simply the first  $B$  distributions are chosen as the background model:

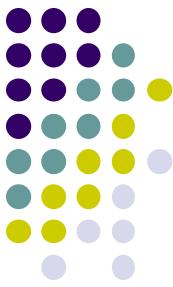
$$B = \operatorname{argmin}_b (\sum_{i=1}^b \omega_i > T)$$

where  $T$  is minimum portion of the image which is expected to be background.

# Background Model Estimation



- ▶ After background model estimation **red** distributions become the background model and **black** distributions are considered to be foreground.



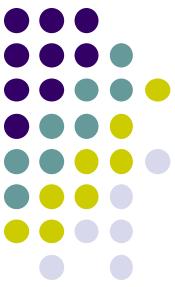
# Advantages Vs Shortcomings

## Advantages:

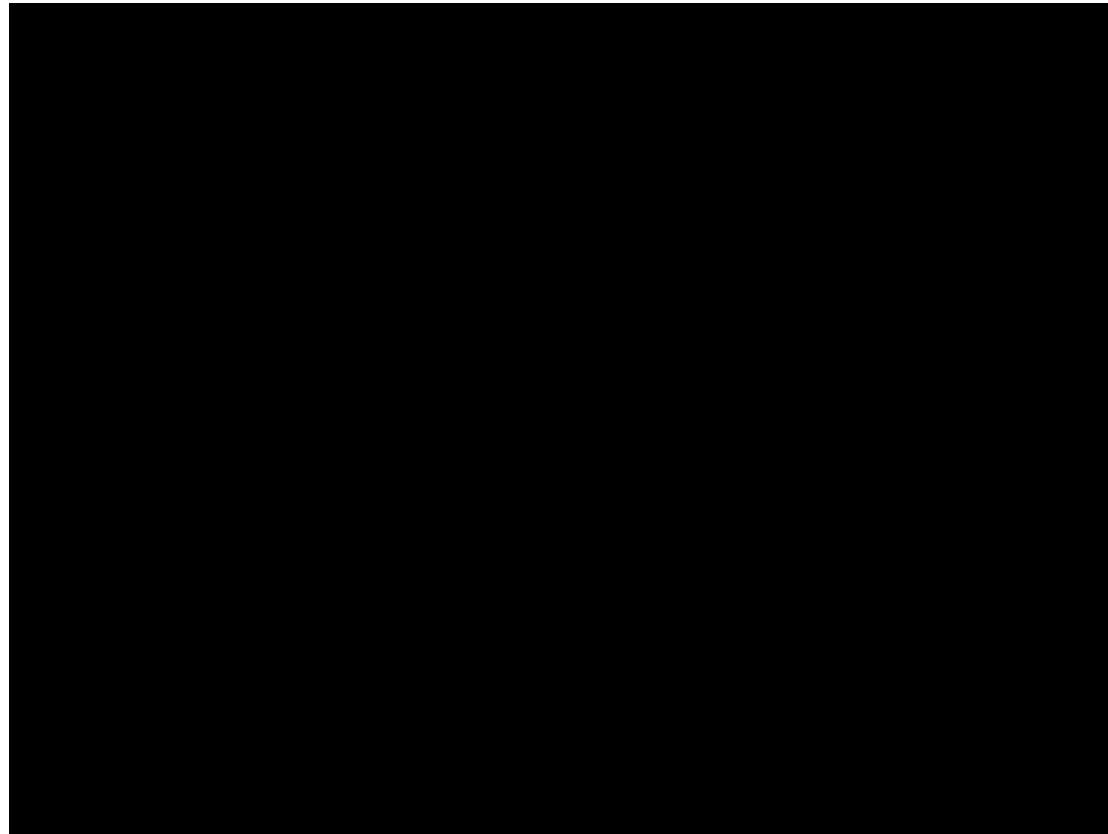
- Same as adaptive Gaussians but now can also cope with multimodal background distributions

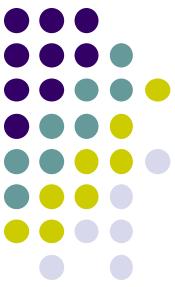
## Disadvantages:

- ▶ Cannot deal with sudden, drastic lighting changes!
- ▶ Initializing the Gaussians is important (median filtering).
- ▶ There are relatively many parameters, and they should be selected intelligently.

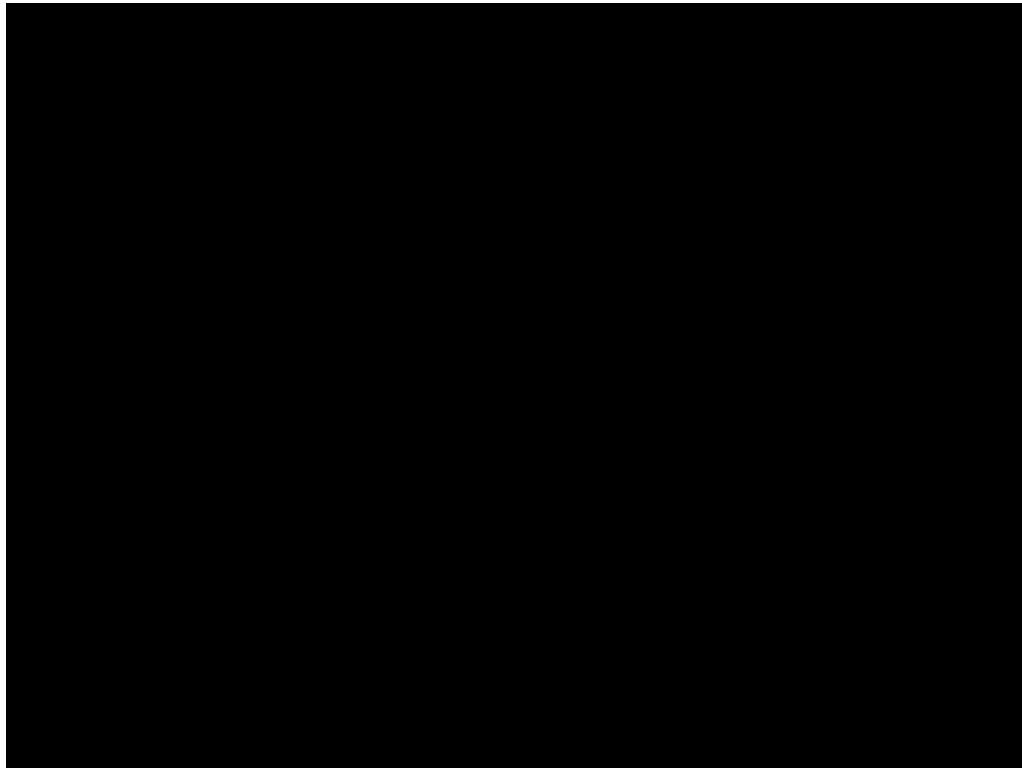


# Background model





# Results





# Summary

- ▶ Simple background subtraction approaches such as **frame differencing**, **mean** and **median** filtering, are pretty fast.
  - ▶ However, their global, constant thresholds make them **insufficient** for challenging real-world problems.
- ▶ **Adaptive background mixture model** approach can handle challenging situations: such as bimodal backgrounds, long-term scene changes and repetitive motions in the clutter.
- ▶ Adaptive background mixture model can further be improved by **incorporating temporal information**, or **using some regional background subtraction approaches in conjunction with it**.