

Digital Image Processing

CHAPTER 1: INTRODUCTION

WHAT IS A DIGITAL IMAGE?

- A **digital image** can be defined to be a function of two real variables, $f(x,y)$, where x and y are spatial coordinates, and the amplitude f at a given pair of coordinates is called the intensity of the image at that point.
- Every digital image is composed of a finite number of elements, called pixels, each with a particular location and a finite value.

WHAT IS DIGITAL IMAGE PROCESSING?

- ❑ **Digital image processing:** analysis, manipulation, storage, and display of graphical images from sources such as photographs, drawings, and video.
- ❑ The process is composed of **3 steps**
 - **Input:** the image is either directly captured with a digital camera or an analog image is digitized in a scanner to obtain the binary data a computer can understand.
 - **Processing:** may include any process including image enhancement, restoration, compression, segmentation, or watermarking as well as extraction of image features.
 - **Output:** consists of the display/printing of the processed image, or the features extracted from the input image.

CHAPTER 1: INTRODUCTION

- Interest in DIP methods stem from 2 principal applications:
 - Improvement of pictorial information for human interpretation
 - Processing of image data for storage, transmission, and representation for autonomous machine perception.

BARTLANE CABLE PICTURE TRANSMISSION SYSTEM

- Early 1920s: **Bartlane** cable picture transmission system
 - Reduced the time required to transport a picture across the Atlantic from more than a week to less than three hours.
 - Pictures were coded for cable transmission and then reconstructed at the receiving end on a telegraph printer fitted with type faces simulating a halftone pattern.
 - The early Bartlane systems were capable of coding images in five distinct brightness levels. This was increased to fifteen levels in 1929.



FIGURE 1.1 A digital picture produced in 1921 from a coded tape by a telegraph printer with special type faces. (McFarlane.)

OTHER HISTORICAL DEVELOPMENTS

- 1964: Pictures of the moon transmitted by Ranger 7 were processed by a computer at the Jet Propulsion Lab to correct various types of image distortion inherent in the on-board TV camera.
- Experience with Ranger 7 led to an improvement in image enhancement and restoration in other projects.
 - Surveyor missions to the moon
 - Mariner series of flyby missions to Mars
 - Apollo manned flights to the moon

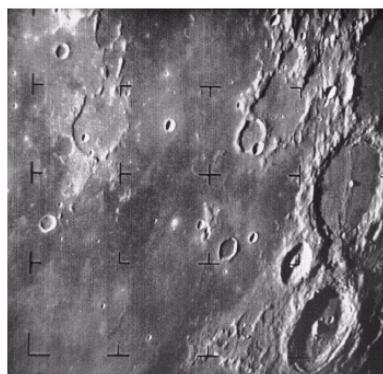


FIGURE 1.4 The first picture of the moon by a U.S. spacecraft. *Ranger 7* took this image on July 31, 1964 at 9:09 A.M. EDT, about 17 minutes before impacting the lunar surface. (Courtesy of NASA.)

APPLICATIONS OF IMAGE PROCESSING

- Early 1970's: Invention of computerized axial tomography
- Applications of image processing
 - For human interpretation
 - Geography
 - Archeology.
 - Physics
 - Astronomy
 - Biology
 - Nuclear medicine
 - Law enforcement
 - Defense
 - For machine perception
 - Automatic character recognition
 - Industrial machine vision for product assembly and inspection
 - Military recognizance
 - Automatic processing of fingerprints
 - Screening of X-rays and blood samples
 - Machine processing of aerial and satellite imagery for weather prediction and environment assessment

ENERGY SOURCES FOR IMAGES

- Electromagnetic energy spectrum: Principal source of energy
 - The full range of **frequencies**, from radio waves to gamma rays, that characterizes light.
 - Gamma rays: highest energy, shortest wavelength
 - Radio waves: lowest energy, longest wavelength
- **Electromagnetic radiation** can be described in terms of a stream of **photons**, each traveling in a wave-like pattern, moving at the **speed of light** and carrying some amount of energy.
- The amount of energy a photon has makes it sometimes behave more like a **wave** and sometimes more like a **particle**. This is called the "**wave-particle duality**" of **light**.
- Low energy photons (such as radio) behave more like waves, while higher energy photons (such as X-rays) behave more like particles.
- Other sources of energy: acoustic, ultrasonic, electronic

ELECTROMAGNETIC SPECTRUM

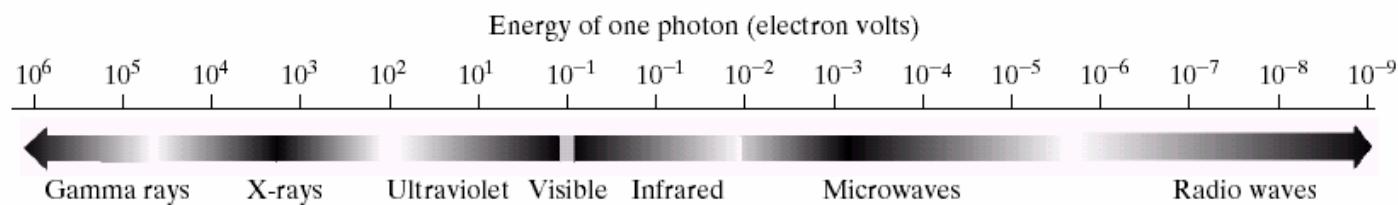


FIGURE 1.5 The electromagnetic spectrum arranged according to energy per photon.

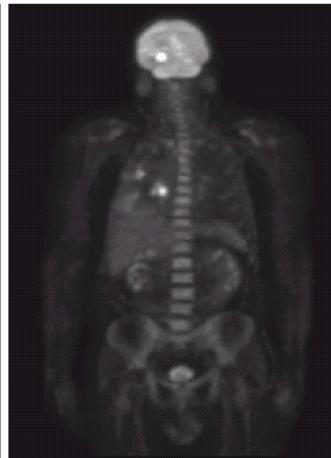
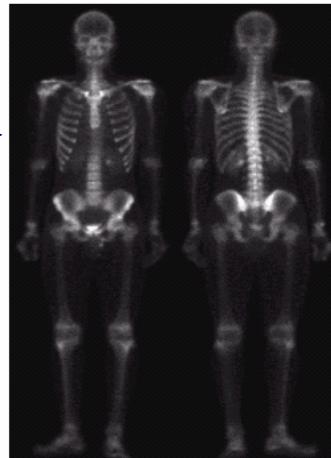
GAMMA-RAY

Major uses: Nuclear medicine & astronomical observations

a
b
c
d

FIGURE 1.6
Examples of gamma-ray imaging. (a) Bone scan. (b) PET image. (c) Cygnus Loop. (d) Gamma radiation (bright spot) from a reactor valve.
(Images courtesy of (a) G.E. Medical Systems, (b) Dr. Michael E. Casey, CTI PET Systems, (c) NASA, (d) Professors Zhong He and David K. Wehe, University of Michigan.)

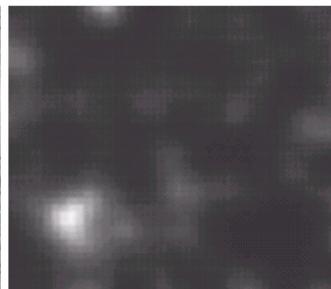
Bone scan: locates sites of bone pathology



← **PET:** another major modality of nuclear imaging

One sample of a sequence that constitutes a 3d rendition of the patient.

A star in the constellation Cygnus exploded about 15K years ago.



← **Gamma radiation from a valve in a nuclear reactor**

The superheated stationary gas cloud (Cygnus Loop) glows in a spectacular array of colors.

X-RAY

X-rays are among the oldest sources of EM radiation used for imaging.

Major uses: Medical diagnostics, astronomy, angiography, CAT scans

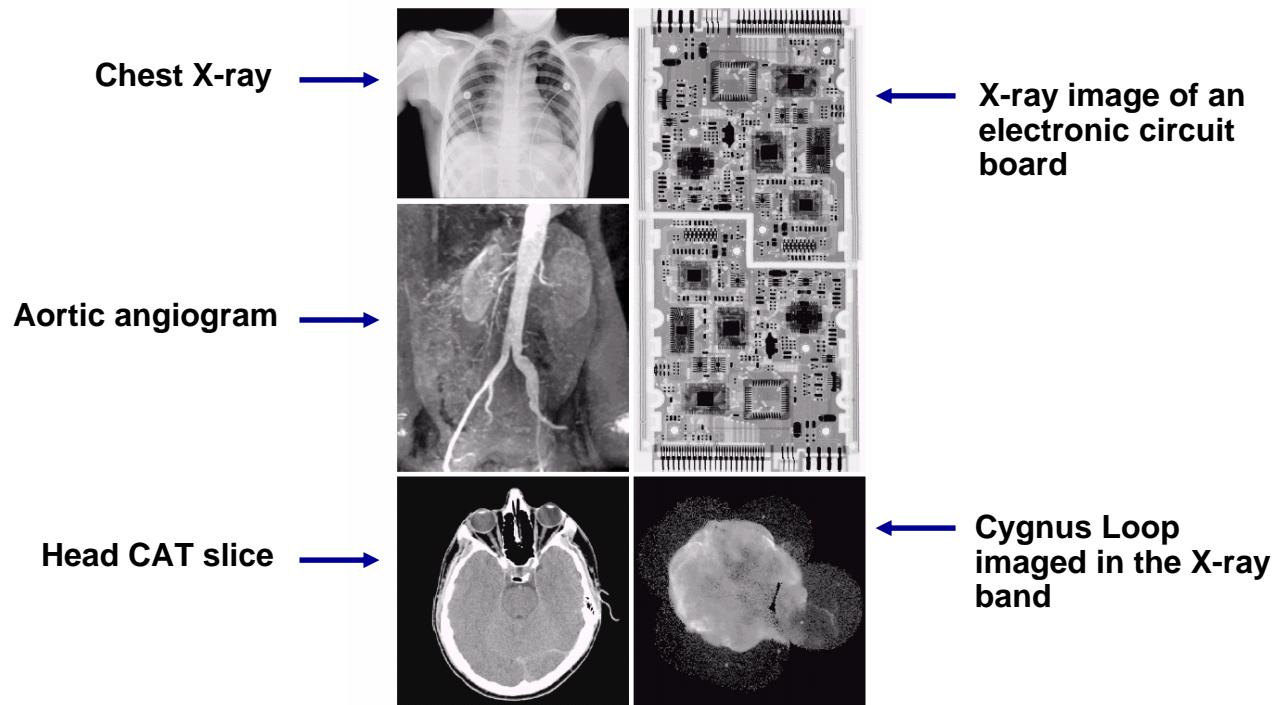


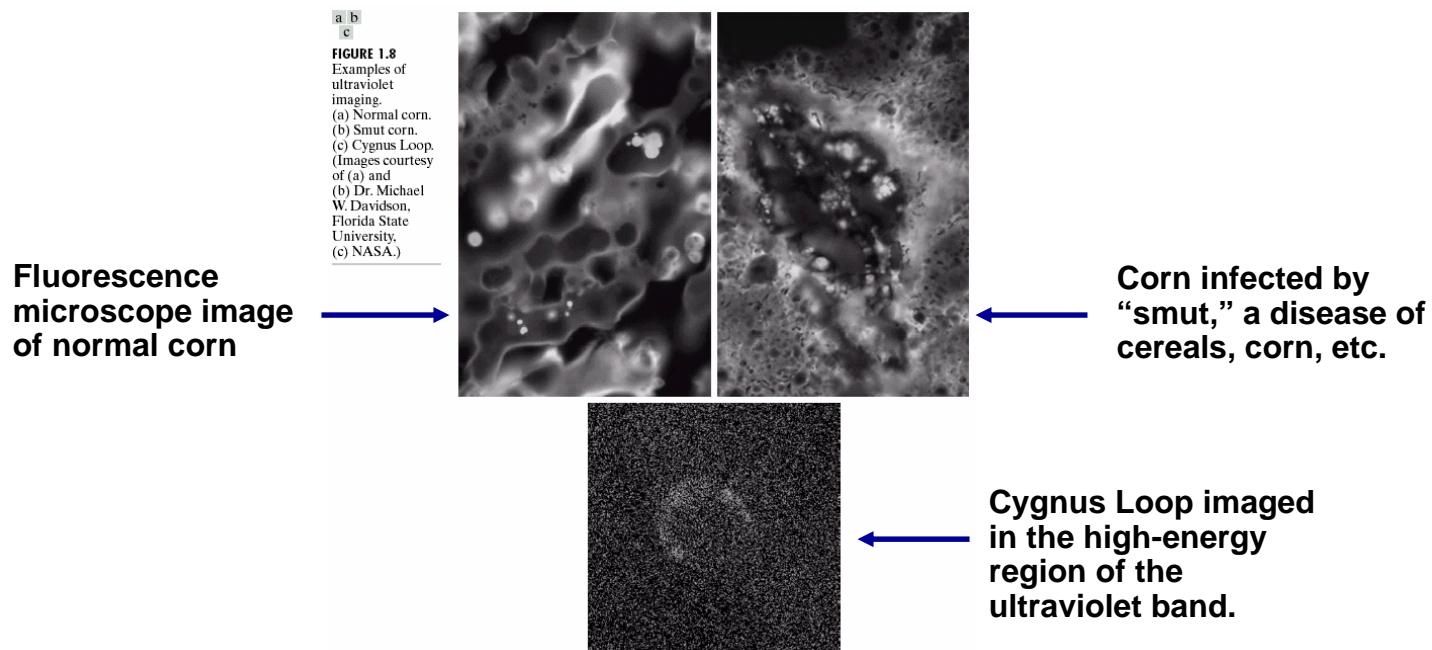
FIGURE 1.7 Examples of X-ray imaging. (a) Chest X-ray. (b) Aortic angiogram. (c) Head CT. (d) Circuit boards. (e) Cygnus Loop. (Images courtesy of (a) and (c) Dr. David R. Pickens Dept. of Radiology & Radiological Sciences, Vanderbilt University Medical Center, (b) Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, (d) Mr. Joseph E. Pascente, Lixi, Inc., and (e) NASA.)

ULTRAVIOLET

Major uses: lithography, industrial inspection, microscopy, lasers, biological imaging, astronomical observations.

Fluorescence is a phenomenon discovered in the middle of 19th century.

Fluorescence microscopy is a excellent method for studying materials that can be made to fluoresce (either in natural form or when treated with chemicals capable of fluorescing).



VISIBLE AND INFRARED – LIGHT MICROSCOPE

Major uses: light microscopy, astronomy, remote sensing, industry, law enforcement.

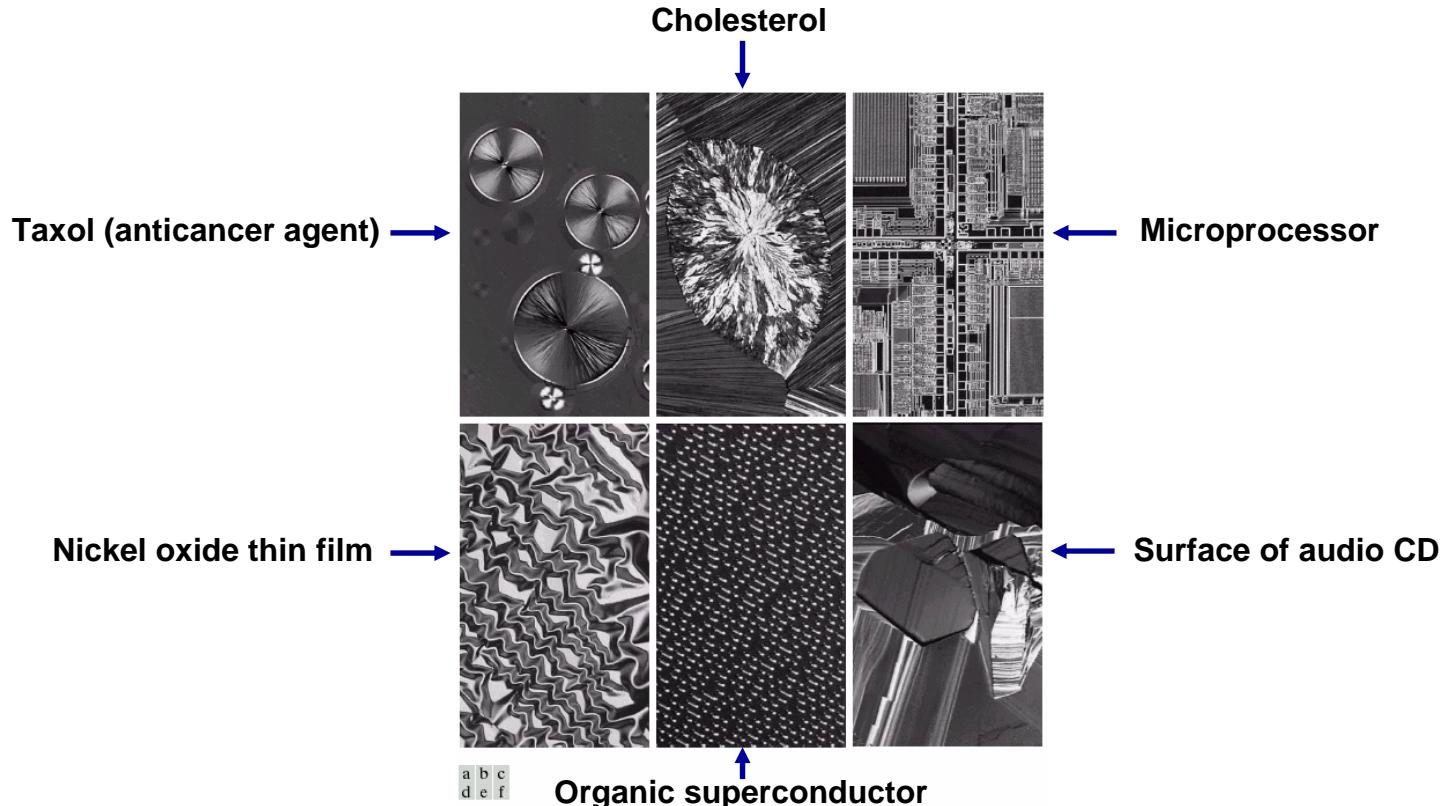


FIGURE 1.9 Examples of light microscopy images. (a) Taxol (anticancer agent)—magnified 250×. (b) Cholesterol—40×. (c) Microprocessor—60×. (d) Nickel oxide thin film—600×. (e) Surface of audio CD—1750×. (f) Organic superconductor—450×. (Images courtesy of Dr. Michael W. Davidson, Florida State University.)

VISIBLE AND INFRARED – REMOTE SENSING

TABLE 1.1
Thematic bands
in NASA's
LANDSAT
satellite.

Remote sensing:
usually includes several
bands in the visual and
infrared regions.

Band No.	Name	Wavelength (μm)	Characteristics and Uses
1	Visible blue	0.45–0.52	Maximum water penetration
2	Visible green	0.52–0.60	Good for measuring plant vigor
3	Visible red	0.63–0.69	Vegetation discrimination
4	Near infrared	0.76–0.90	Biomass and shoreline mapping
5	Middle infrared	1.55–1.75	Moisture content of soil and vegetation
6	Thermal infrared	10.4–12.5	Soil moisture; thermal mapping
7	Middle infrared	2.08–2.35	Mineral mapping

LANDSAT obtains
and transmits
images of the Earth
from space, for
purposes of
monitoring
environmental
conditions.

Multispectral imaging:

One image for each
band in the above
table.

The differences
between visual and
infrared image
features are quite
noticeable.

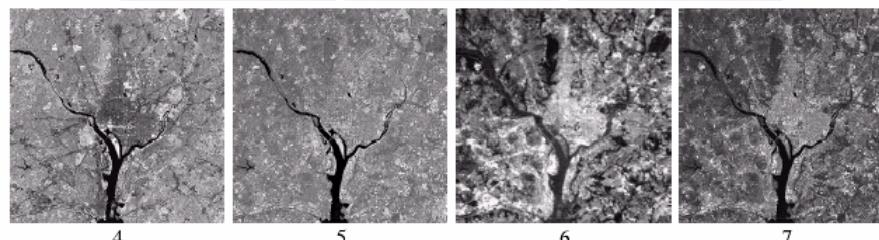
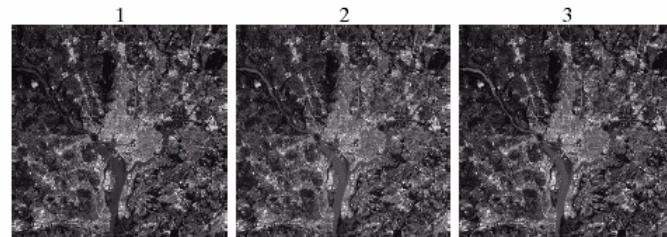


FIGURE 1.10 LANDSAT satellite images of the Washington, D.C. area. The numbers refer to the thematic bands in Table 1.1. (Images courtesy of NASA.)

VISIBLE AND INFRARED – WEATHER OBSERVATION AND PREDICTION

An image of a **hurricane** taken by a satellite using sensors in the visible and infrared bands.



FIGURE 1.11
Multispectral image of Hurricane Andrew taken by NOAA GEOS (Geostationary Environmental Operational Satellite) sensors. (Courtesy of NOAA.)

INFRARED – HUMAN SETTLEMENTS (THE AMERICAS)

FIGURE 1.12
Infrared satellite
images of the
Americas. The
small gray map is
provided for
reference.
(Courtesy of
NOAA.)



These images are part of
**Nighttime Lights of the
World** data set.

This set provides a
global inventory of
human settlements.



VISIBLE – AUTOMATED VISUAL INSPECTION OF MANUFACTURED GOODS

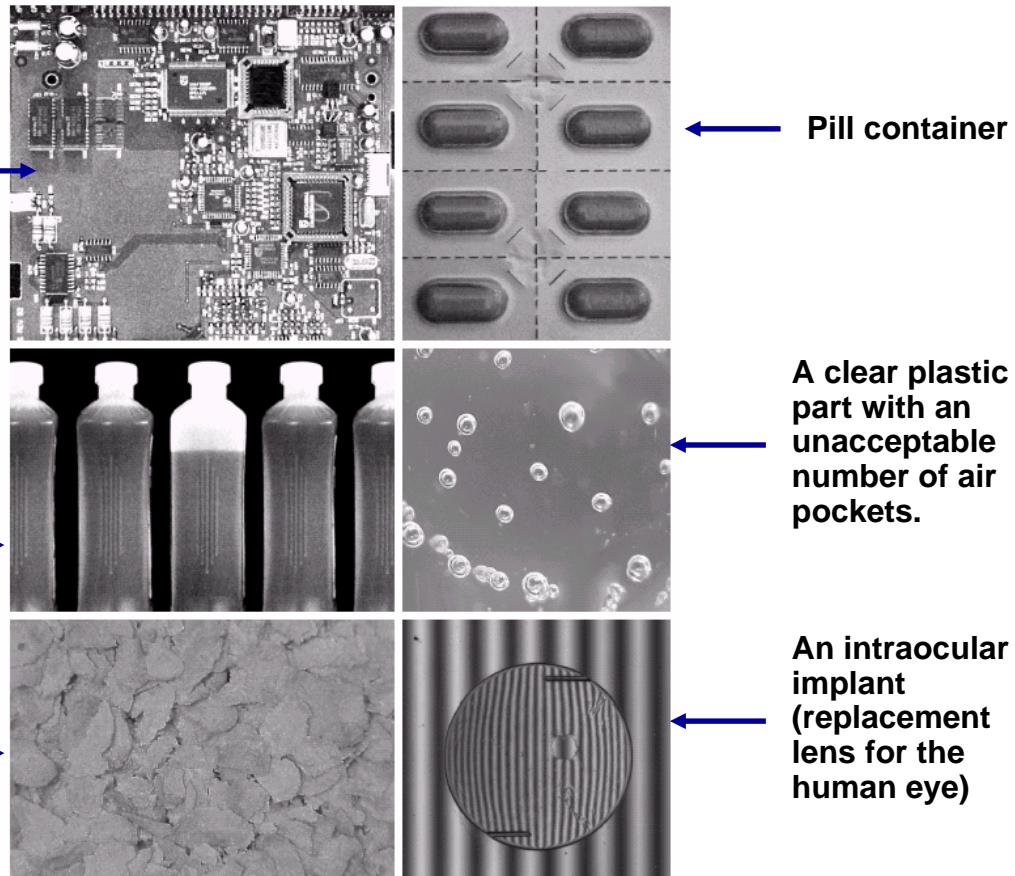
A circuit board controller (the back square is a missing component)

a
b
c
d
e
f

FIGURE 1.14
Some examples of manufactured goods often checked using digital image processing. (a) A circuit board controller. (b) Packaged pills. (c) Bottles. (d) Bubbles in clear-plastic product. (e) Cereal. (f) Image of intraocular implant. (Fig. (f) courtesy of Mr. Pete Sites, Perceptics Corporation.)

A bottle that is not filled up to an acceptable level.

A batch of cereal inspected for color and the presence of anomalies such as burned flakes.



VISIBLE – OTHER APPLICATIONS

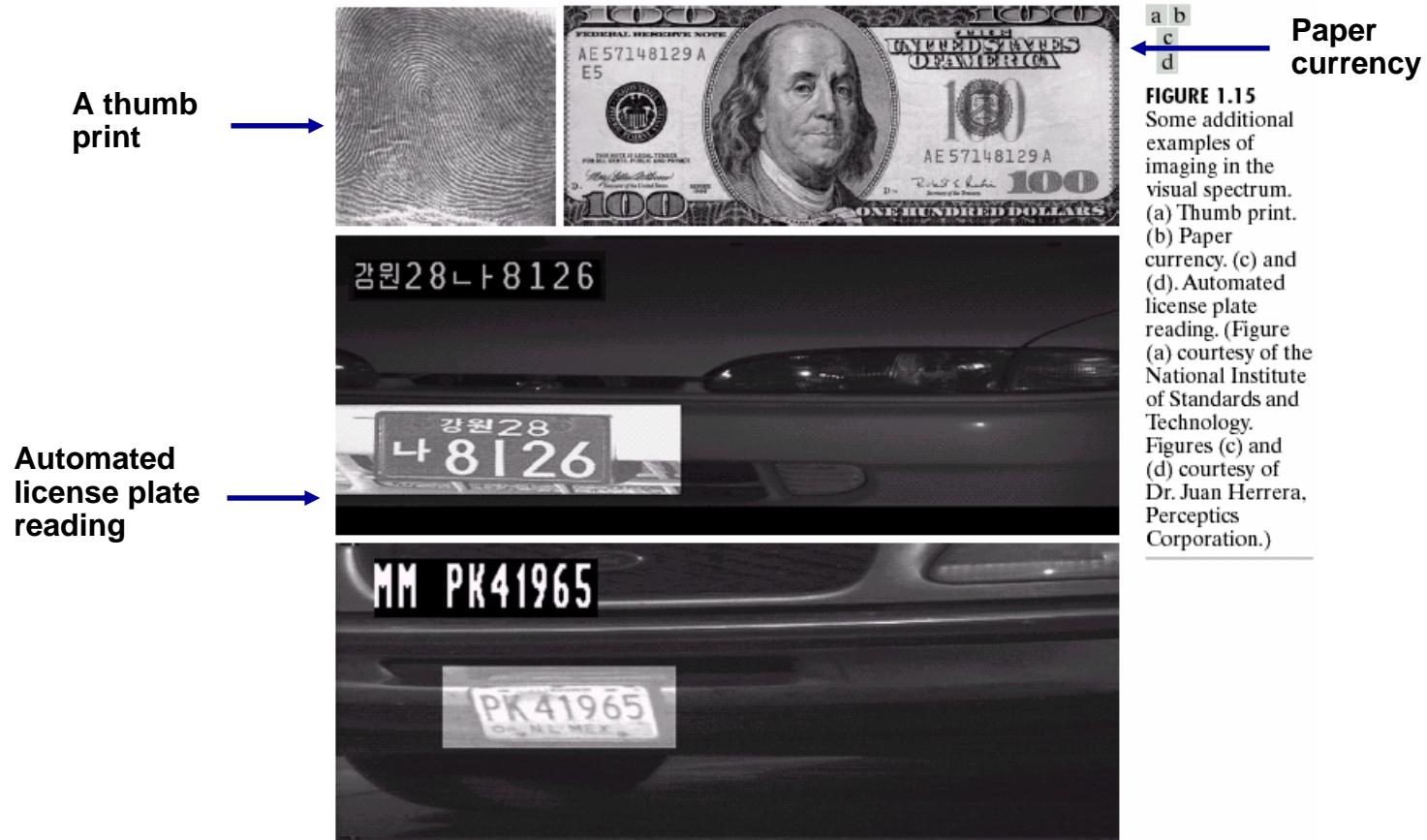


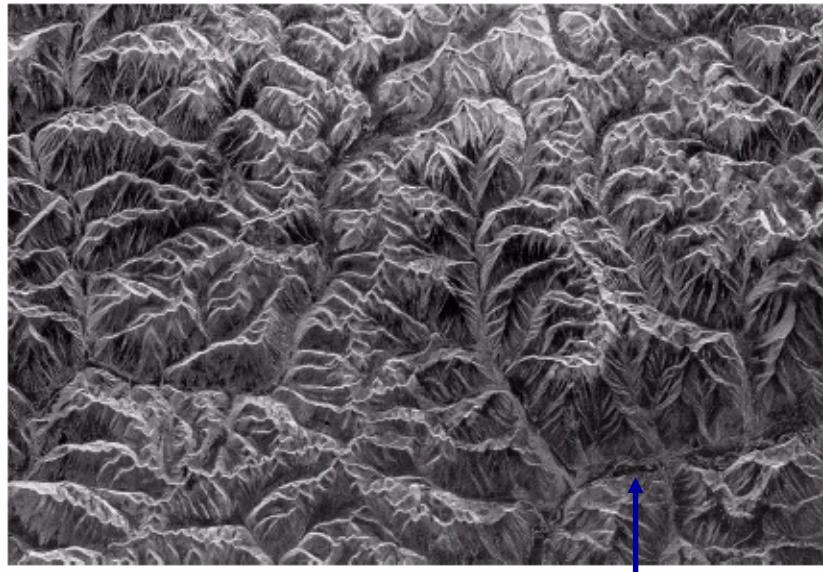
FIGURE 1.15
Some additional examples of imaging in the visual spectrum.
(a) Thumb print.
(b) Paper currency.
(c) and
(d). Automated license plate reading. (Figure (a) courtesy of the National Institute of Standards and Technology. Figures (c) and (d) courtesy of Dr. Juan Herrera, Perceptics Corporation.)

MICROWAVE – IMAGING RADAR

FIGURE 1.16
Spaceborne radar
image of
mountains in
southeast Tibet.
(Courtesy of
NASA.)

Imaging radar

- has the unique ability to collect data over virtually any region at any time, regardless of weather or ambient lightning conditions.
- Works like a flash camera with its own illumination.
- Uses an antenna and digital computer processing to record images.

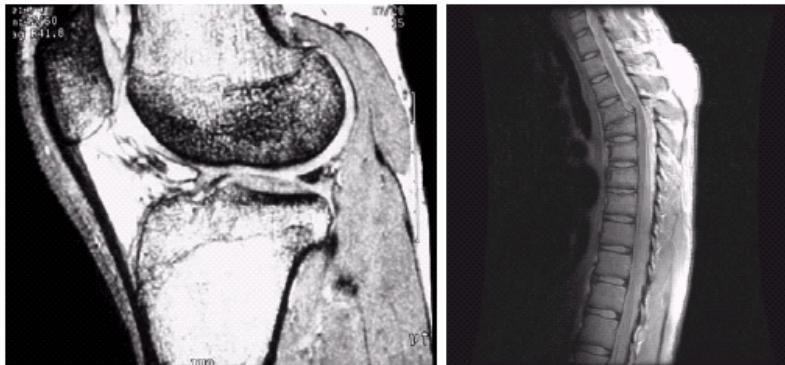


Note the **clarity** and **detail** of the image, unencumbered by clouds or other atmospheric conditions that normally interfere with image in the visual band!

Lhasa River

RADIO – MRI

Major uses: medicine and astronomy



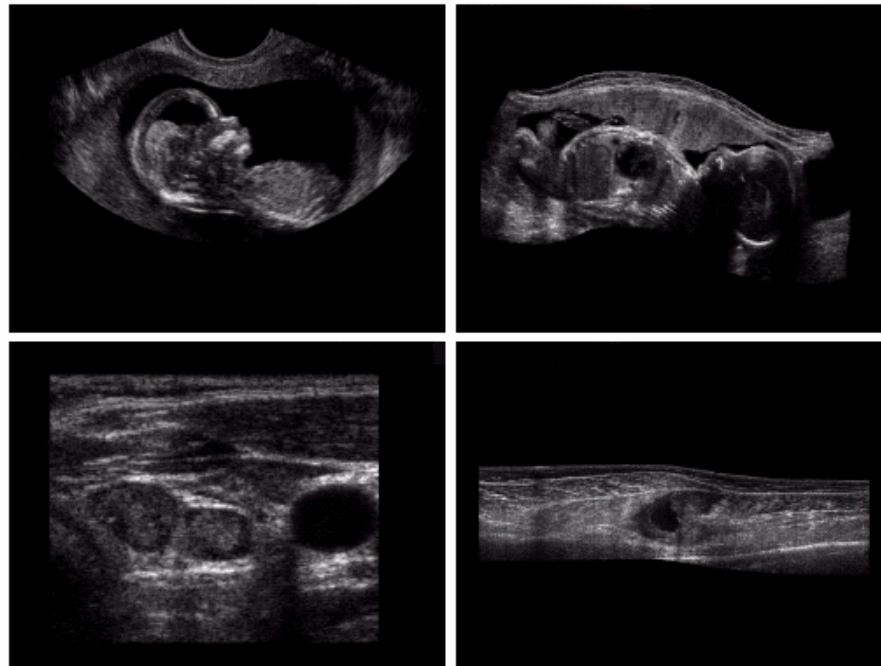
a b

FIGURE 1.17 MRI images of a human (a) knee, and (b) spine. (Image (a) courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, and (b) Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Magnetic resonance imaging (MRI):

- places a patient in a powerful magnet and passes radio waves through the body in short pulses.
- each pulse causes a responding pulse of radio waves to be emitted by the patient's tissues.
- produces a 2D picture of section of the patient.

OTHER IMAGING MODALITIES - ULTRASOUND IMAGING

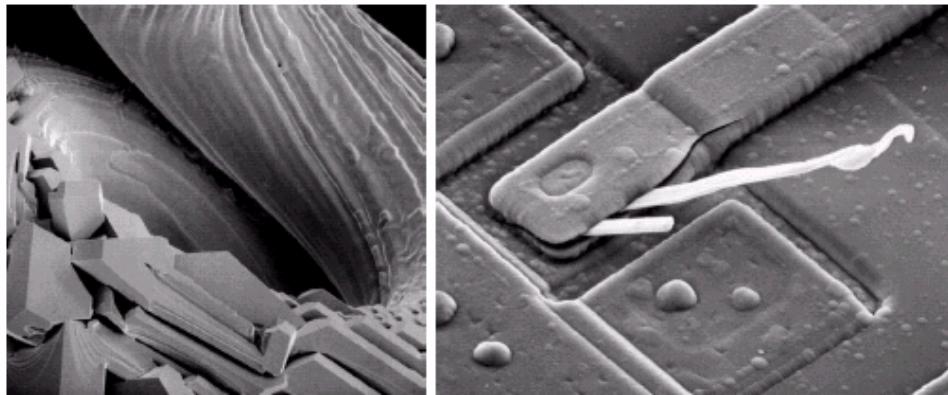


a b
c d

FIGURE 1.20
Examples of ultrasound imaging. (a) Baby.
(2) Another view of baby.
(c) Thyroids.
(d) Muscle layers showing lesion.
(Courtesy of Siemens Medical Systems, Inc., Ultrasound Group.)

Ultrasound imaging: millions of HF sound pulses and echoes are sent and received each second.

OTHER IMAGING MODALITIES - ELECTRON MICROSCOPY



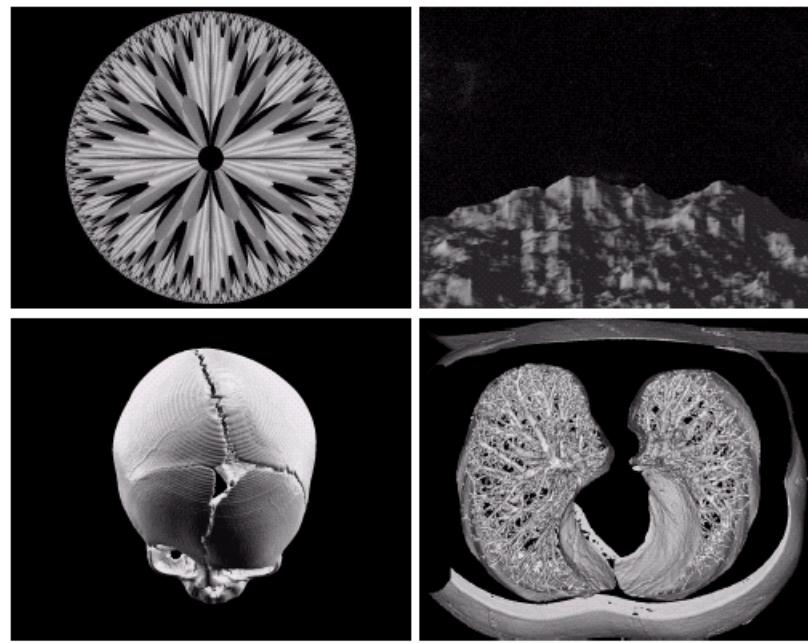
a b

FIGURE 1.21 (a) 250 \times SEM image of a tungsten filament following thermal failure. (b) 2500 \times SEM image of damaged integrated circuit. The white fibers are oxides resulting from thermal destruction. (Figure (a) courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene; (b) courtesy of Dr. J. M. Hudak, McMaster University, Hamilton, Ontario, Canada.)

Electron microscopes

- use a focused beam of electrons instead of light.
- are capable of very high magnification (10,000X or more).
- transmission electron microscope (TEM), scanning electron microscope (SEM)

OTHER IMAGING MODALITIES - COMPUTER-GENERATED OBJECTS



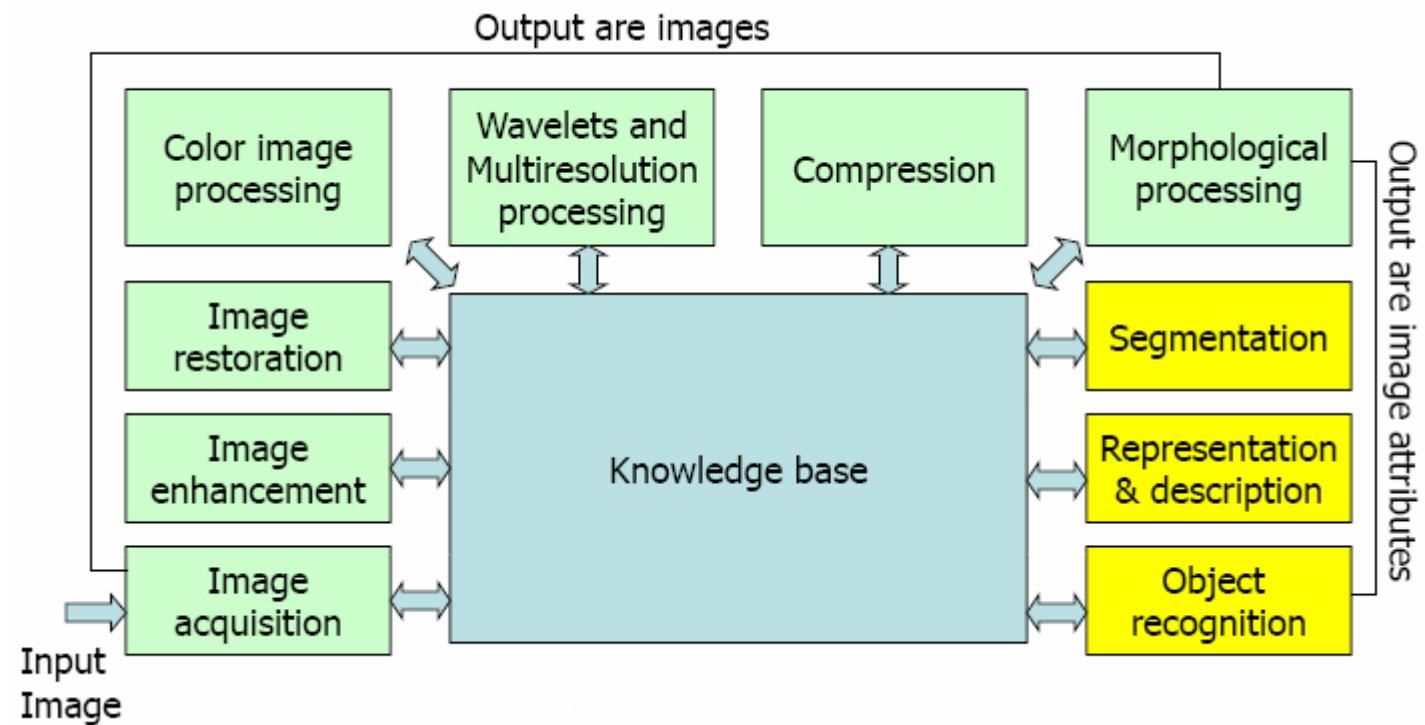
a b
c d

FIGURE 1.22
(a) and (b) Fractal images. (c) and (d) Images generated from 3-D computer models of the objects shown. (Figures (a) and (b) courtesy of Ms. Melissa D. Binde, Swarthmore College. (c) and (d) courtesy of NASA.)

Images that are not obtained from **physical** objects.



FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING



WHAT ARE THE FUNDAMENTAL STEPS?

- **Image acquisition**: capturing an image in digital form.
- **Image enhancement**: making an image look better in a subjective way.
- **Image restoration**: improving the appearance of an image objectively.
- **Color image processing**: color models and basic color processing
- Wavelets and multiresolution processing: Wavelet transform in one and two dimensions.
- **Image compression**: reducing the stored and transmitted image data.
- **Morphological image processing**: extracting image components that are useful in the representation and description of shape.
- **Image segmentation**: partitioning an image into its constituent parts or objects.
- **Representation and description**: boundary representation vs. region representation. Boundary descriptors vs. region descriptors.
- **Recognition**: assigning a label to an object based on its descriptors.

COMPONENTS OF AN IMAGE PROCESSING SYSTEM

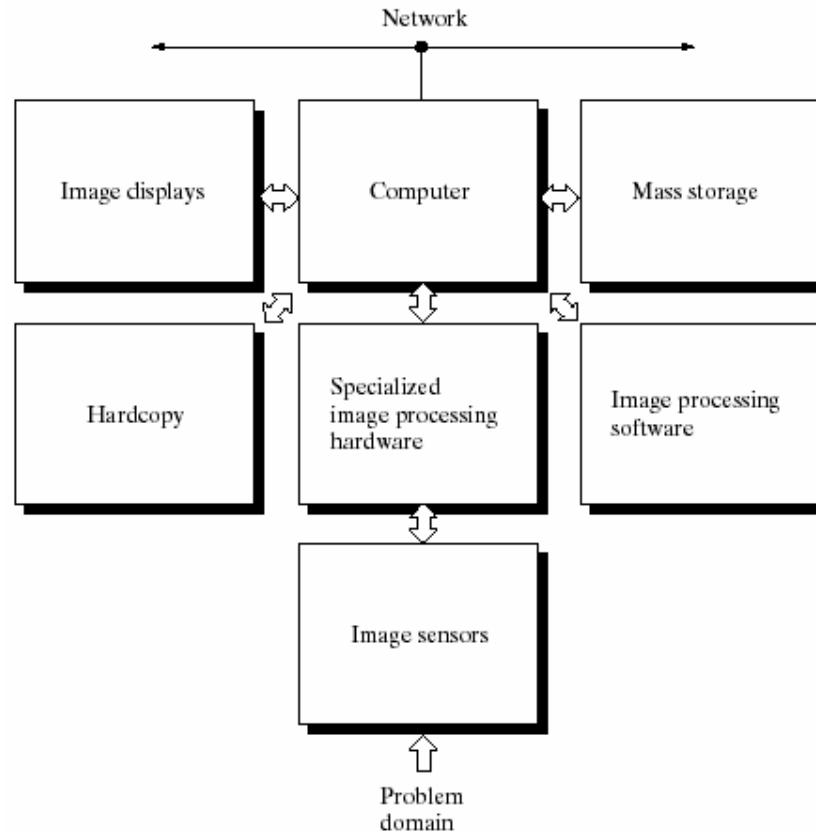


FIGURE 1.24
Components of a
general-purpose
image processing
system.

WHAT ARE THE COMPONENTS?

- **Image sensors and specialized IP HW**
 - A physical device that is sensitive to the energy radiated by the object.
 - A digitizer that converts the output of the physical sensing device into digital form.
 - HW that performs other primitive operations.
- **Computer:** Can range from a PC to a supercomputer.
- **Image processing SW:** Specialized modules that perform specific tasks.
- **Mass storage**
 - Short term storage: main memory, frame buffers
 - On-line storage: magnetic disks and optical disks
 - Archival storage: magnetic tapes and optical disks in “jukeboxes”
- **Hard copy:** all types of printers, film cameras, optical disks (CDs and DVDs).
- **Networking:** bandwidth is the key consideration.

Image Processing



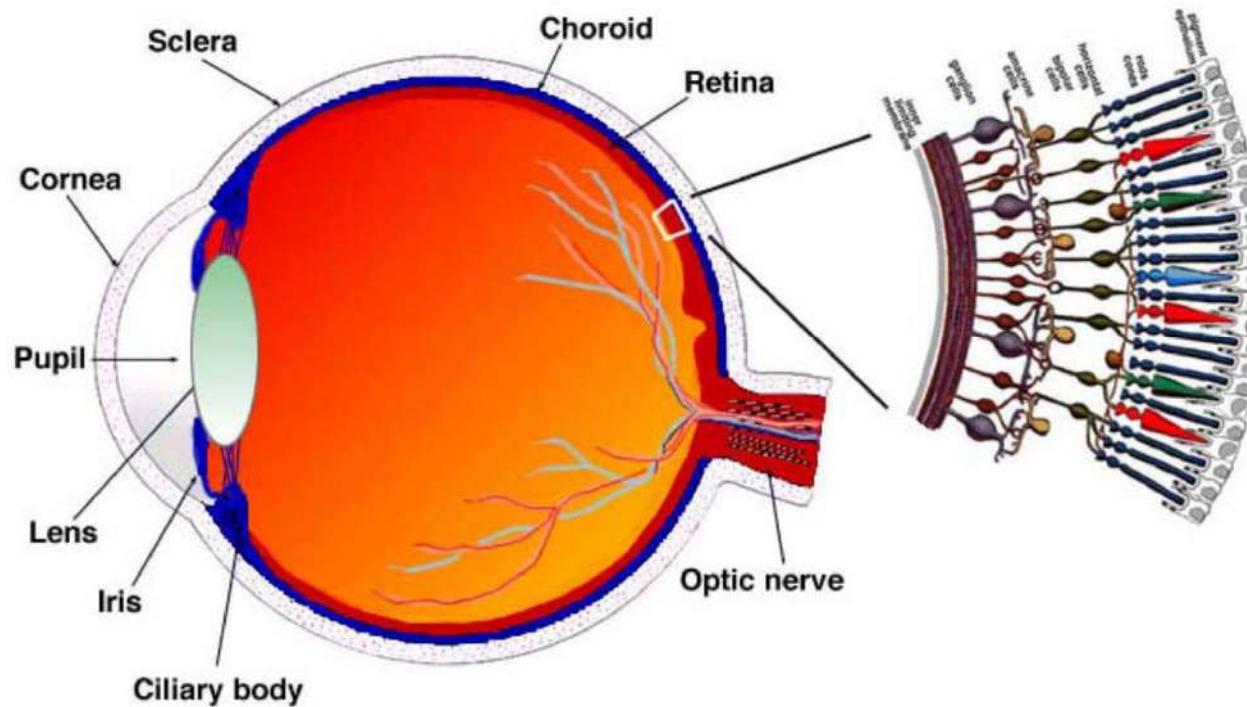
Ch2: Digital image Fundamentals

C. Gonzalez and R. Wood
3rd edition

Digital Image Fundamentals

- Human Visual System
- Image sensing and acquisition
- Image Sampling and Quantization
- Digital Image Representation
- Image Resolution
- Image Interpolation
- Pixel's Relationships
- Distance Measures

Human Visual System



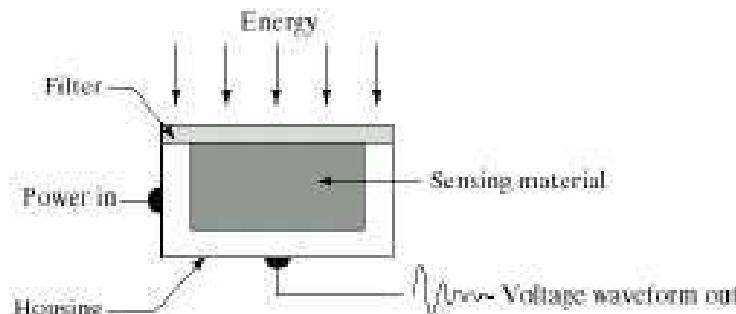
- Light from objects is imaged on the **Retina**.
 - Light hits the retina, which contains photosensitive cells called receptors: **rods and cones**.
 - These cells convert the spectrum into a few discrete values.

Image sensing and acquisition

Sensors

a
b
c

FIGURE 2.12
(a) Single imaging sensor.
(b) Line sensor.
(c) Array sensor.

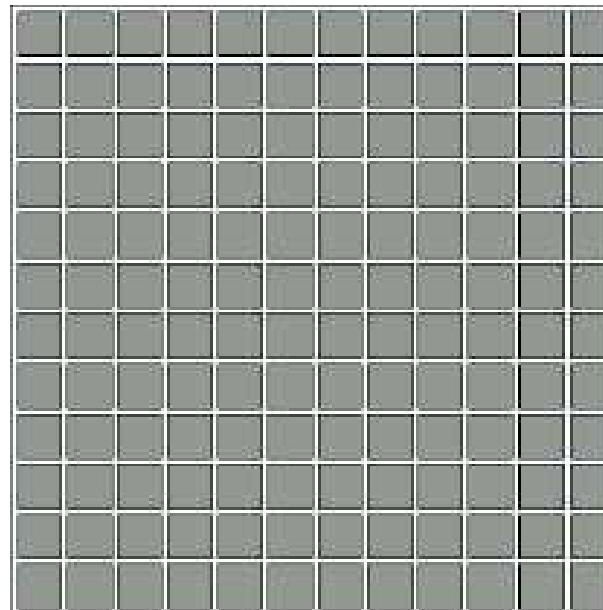


Single

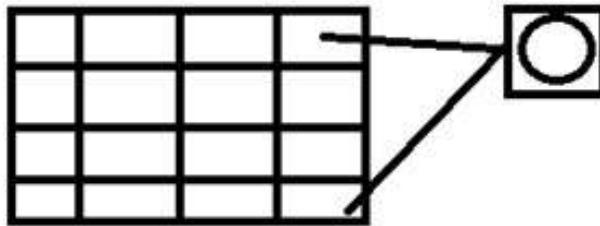
3 principal
sensor
arrangements



Illumination energy is
transformed into
digital images.



Sensors Array



A sensor: sense the intensity of striking photon

- The light falls on the object, then the light reflects back after striking the object and allowed to enter inside the camera.
- When photons of light strike on the chip (sensor) , it is held as a small electrical charge in each photo sensor.
- The response of each sensor is directly equal to the amount of light or (photon) energy striked on the surface of the sensor.

Single Sensor

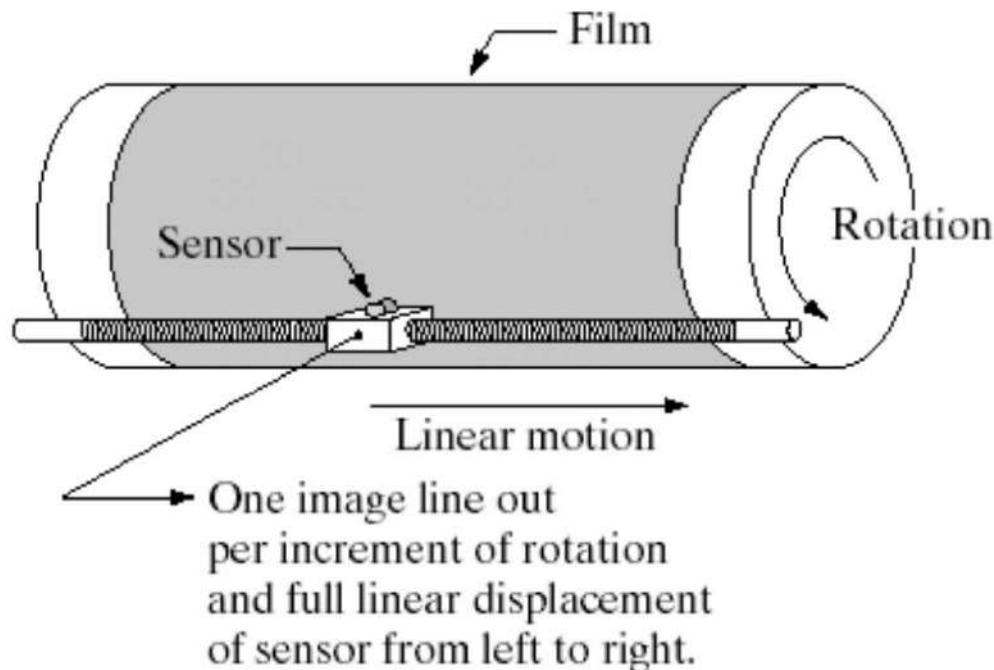


FIGURE 2.13 Combining a single sensor with motion to generate a 2-D image.

Sensor Strips

Typical arrangement in most flat bed scanners

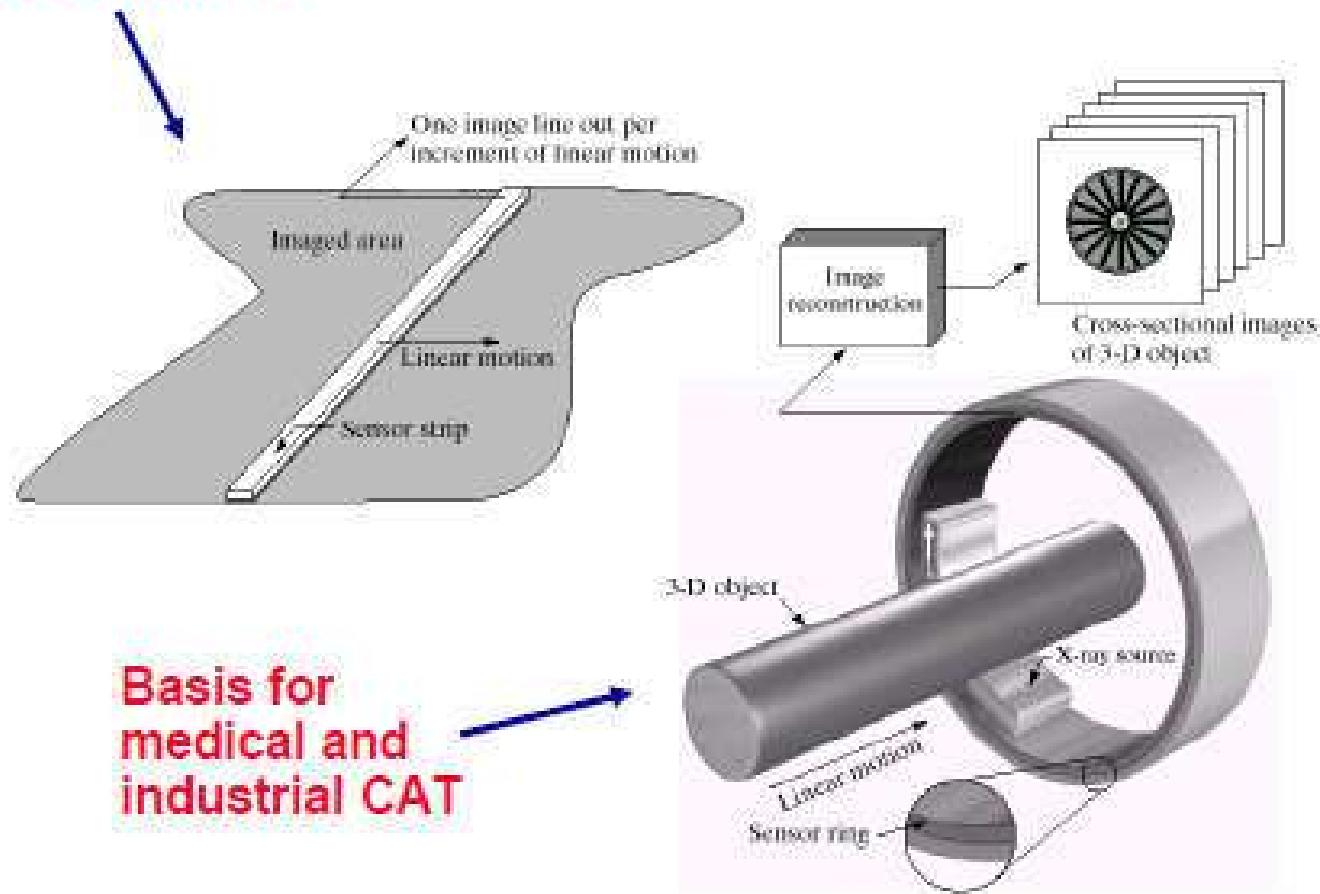


FIGURE 2.14 (a) Image acquisition using a linear sensor strip; (b) Image acquisition using a circular sensor strip.

Sensor Arrays (CCD array)

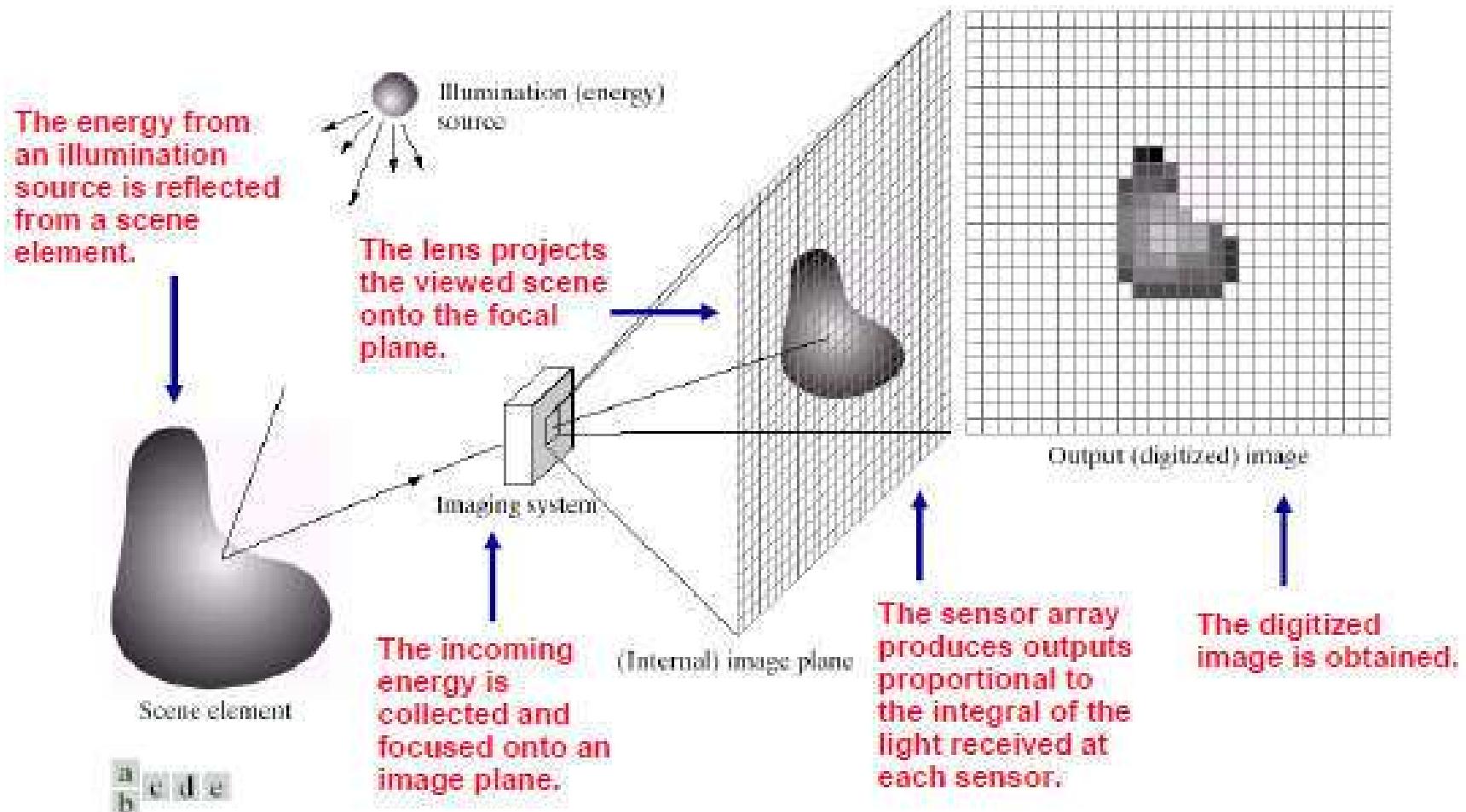


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Image sampling and quantization

- ❑ In order to process the image, it must be saved on computer.
- ❑ The image output in real life is continuous voltage waveform.
- ❑ But computer deals with digital images not with continuous images, thus: continuous images should be converted into digital form.
continuous image (in real life) → digital (computer)

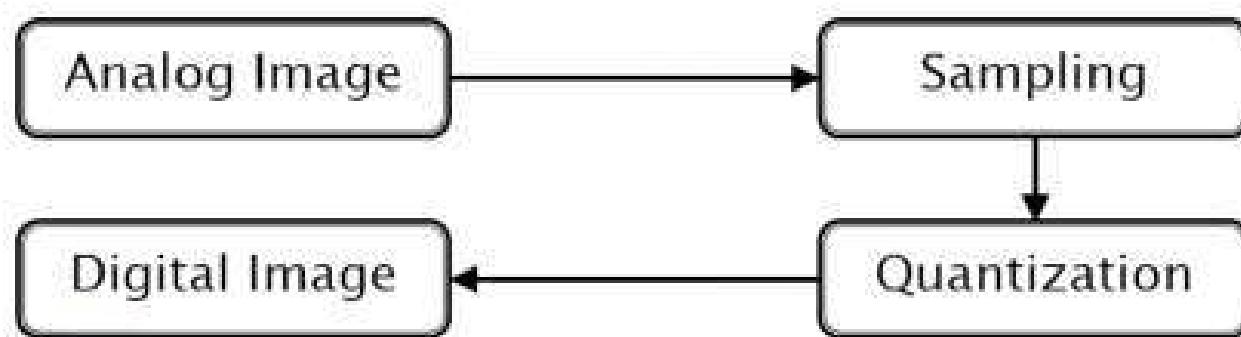


Image sampling and quantization

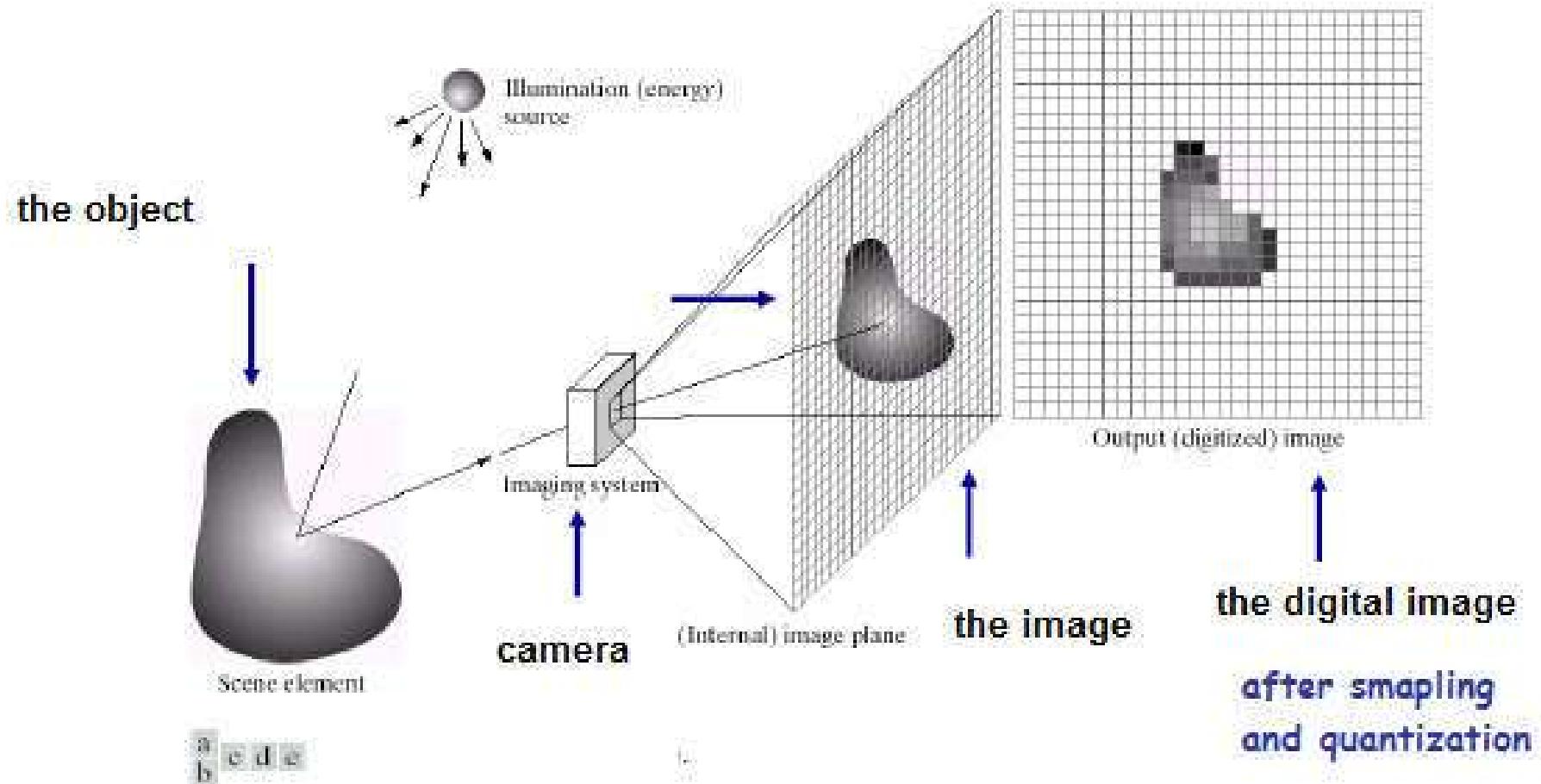


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Image sampling and quantization

continuous image (in real life) → digital (computer)

To do this we use Two processes:

sampling and **quantization**.

- **Sampling**: digitizing the coordinate values
- **Quantization**: digitizing the amplitude values

How does the computer digitize the continuous image?

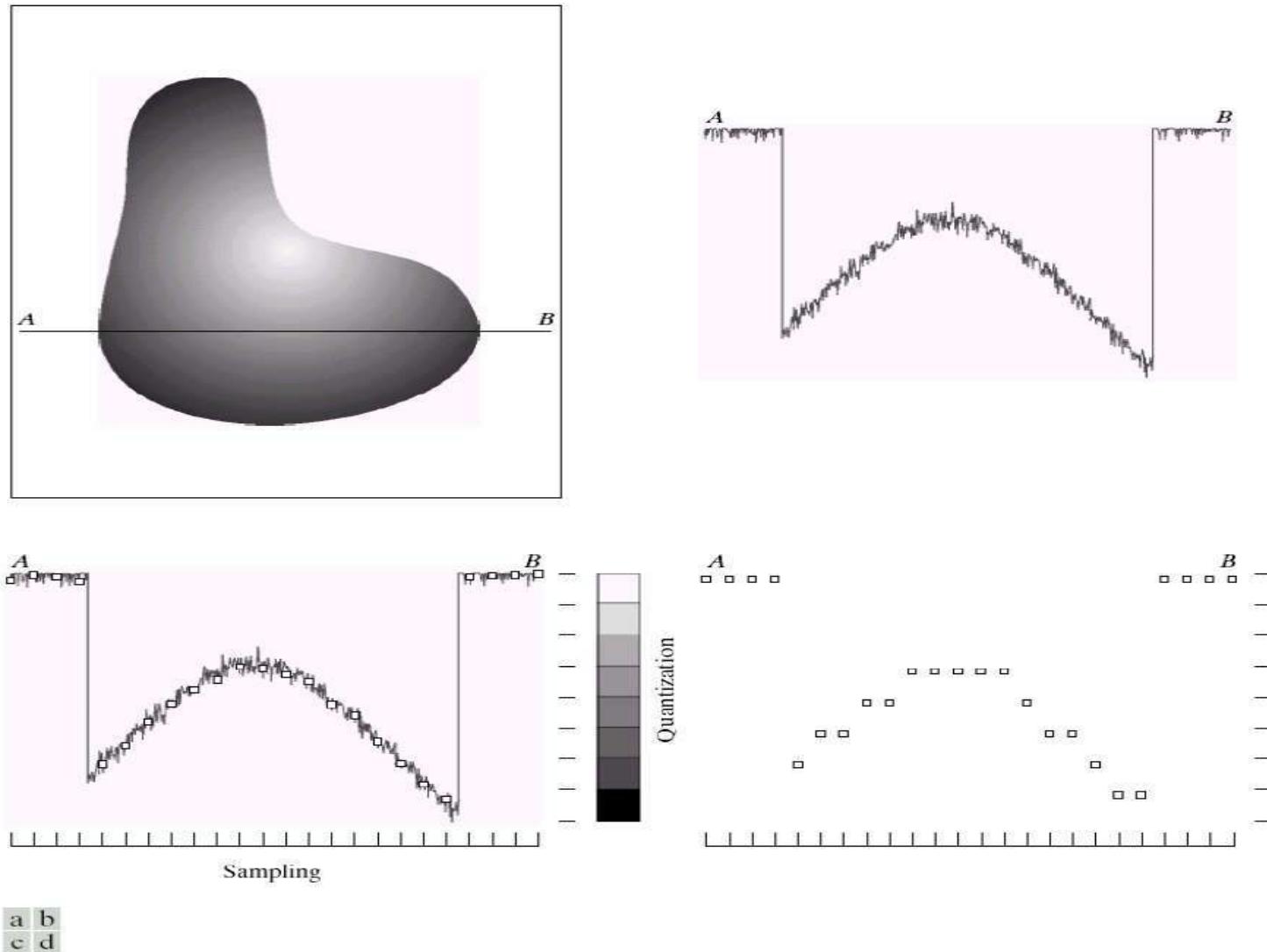
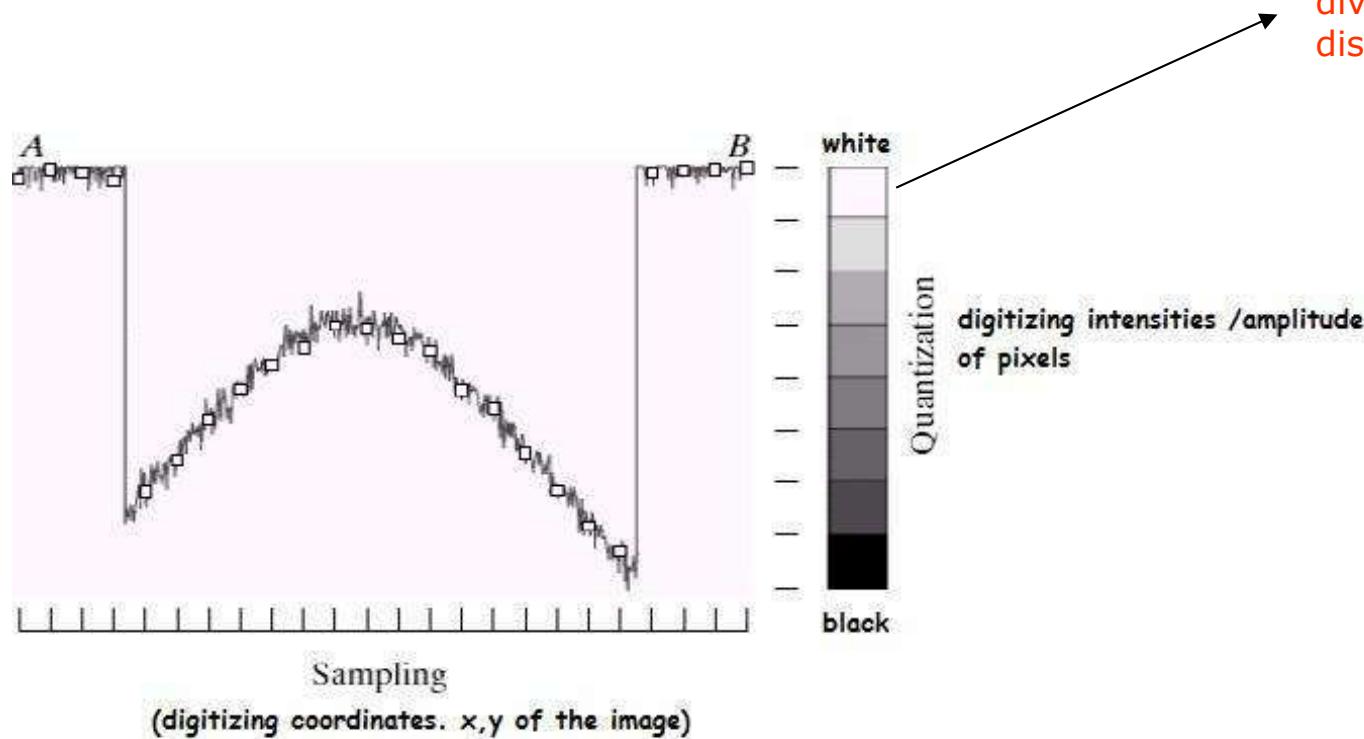


FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

How does the computer digitize the continuous image?

Sampling: digitizing coordinates

Quantization: digitizing intensities



Gray-level scale that divides gray-level into 8 discrete levels

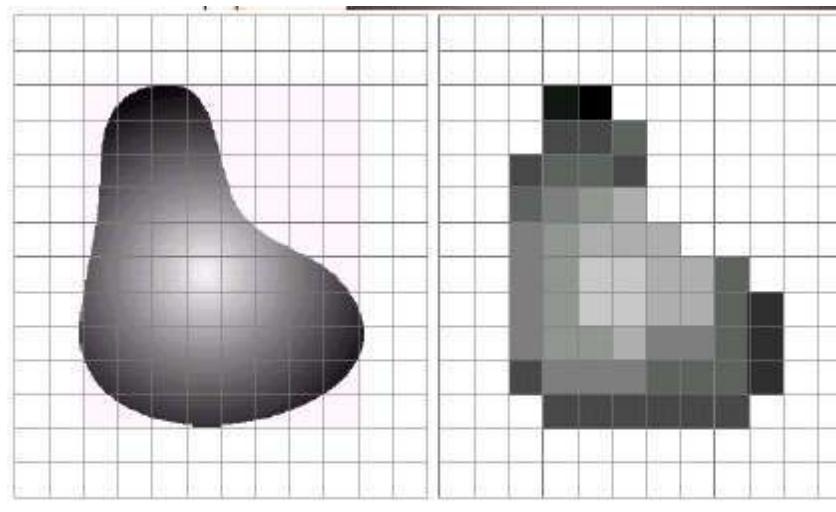
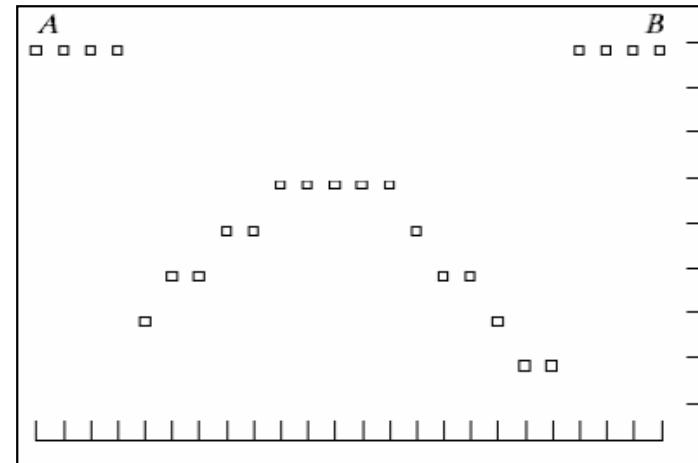
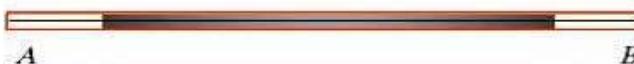
Quantization: converting each sample gray-level value into discrete digital quantity.

sample is a small white square, located by a vertical tick mark as a point x,y

How does the computer digitize the continuous image?

Now:

the digital scanned line AB
representation on computer:

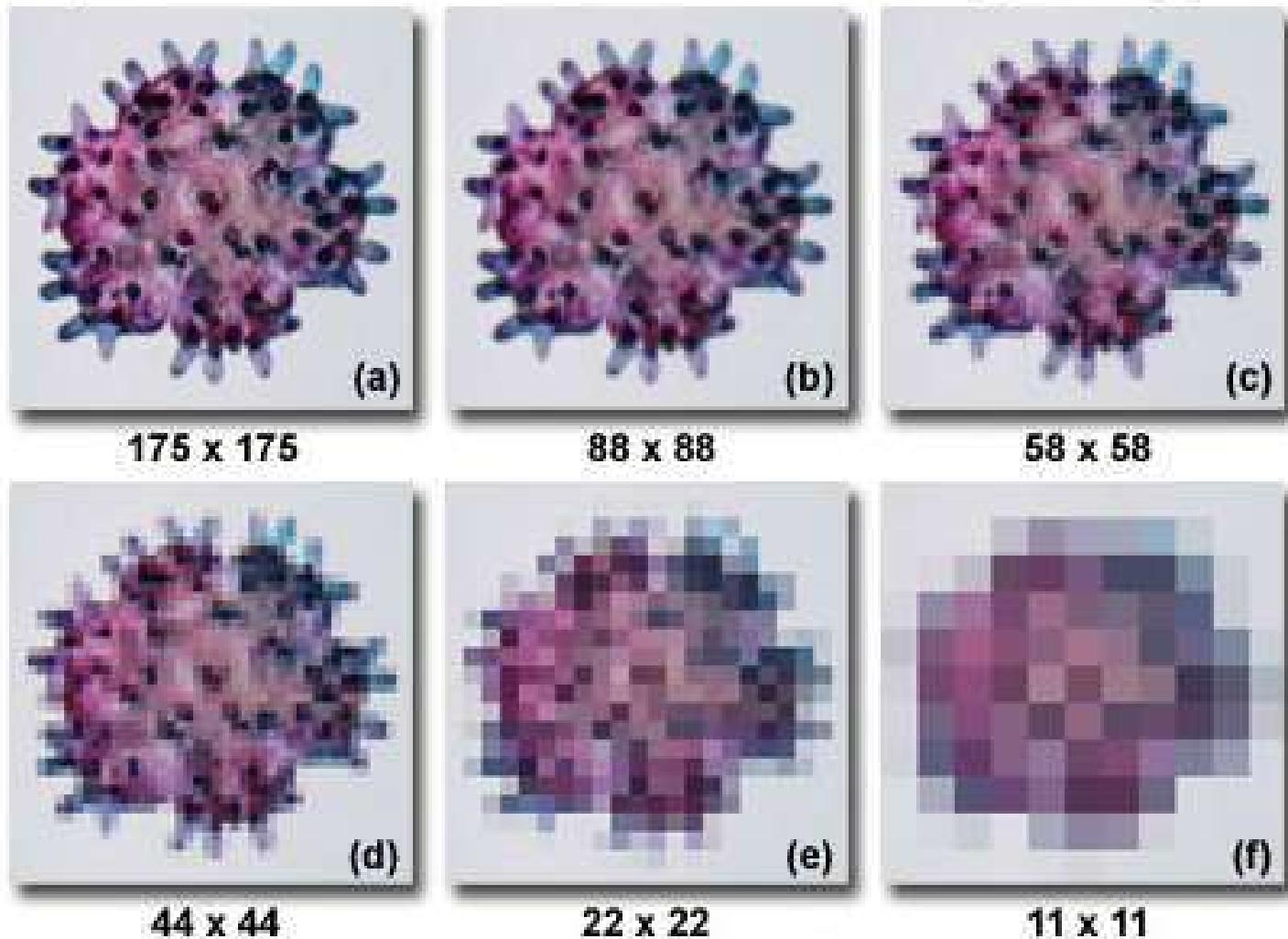


a b

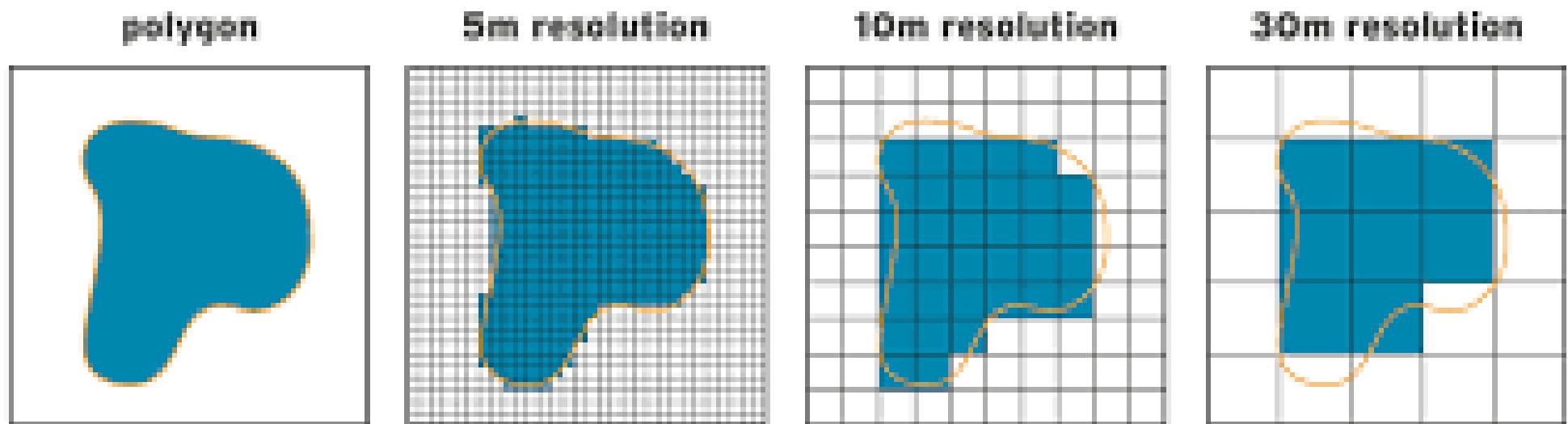
FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Sampling Example

Spatial Resolution Effect on Pixelation in Digital Images



Sampling Example



Smaller cell size
Higher resolution

Larger cell size
Lower resolution

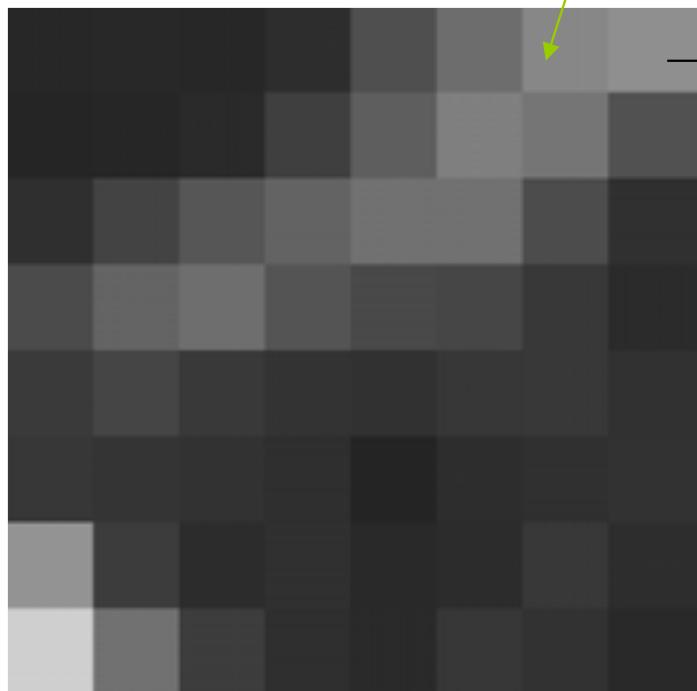
DIGITAL IMAGE REPRESENTATION

What is digital Image?

- **digital image:** function of 2 variables, $f(x,y)$, where x and y are **spatial** coordinates, and f at any pair of coordinates (x,y) is called intensity or gray level of the image at that point.
- **Pixels (pels):** Elements of the digital image , each has intensity.
- **Intensity** of pixel: the amplitude **غزاره** of *gray level (in gray scale images)*

Note: *images can be: binary, grayscale, color.*

What is digital image?



pixel



The image consists of finite number of pixels ($f(x,y)$)

Every pixel Is an intersection تقاطع between a row and a column.

every pixel has intensity كثافة

☞ **Ex:**

$$f(4,3) = 123$$

Refers to a pixel existing on the intersection between row 4 with column 3, and its intensity is 123.

Remember: images can be: binary, grayscale, color.

Binary Images

Binary images are images that have been quantized to two values, usually denoted 0 and 1, but often with pixel values 0 and 255, representing black and white.



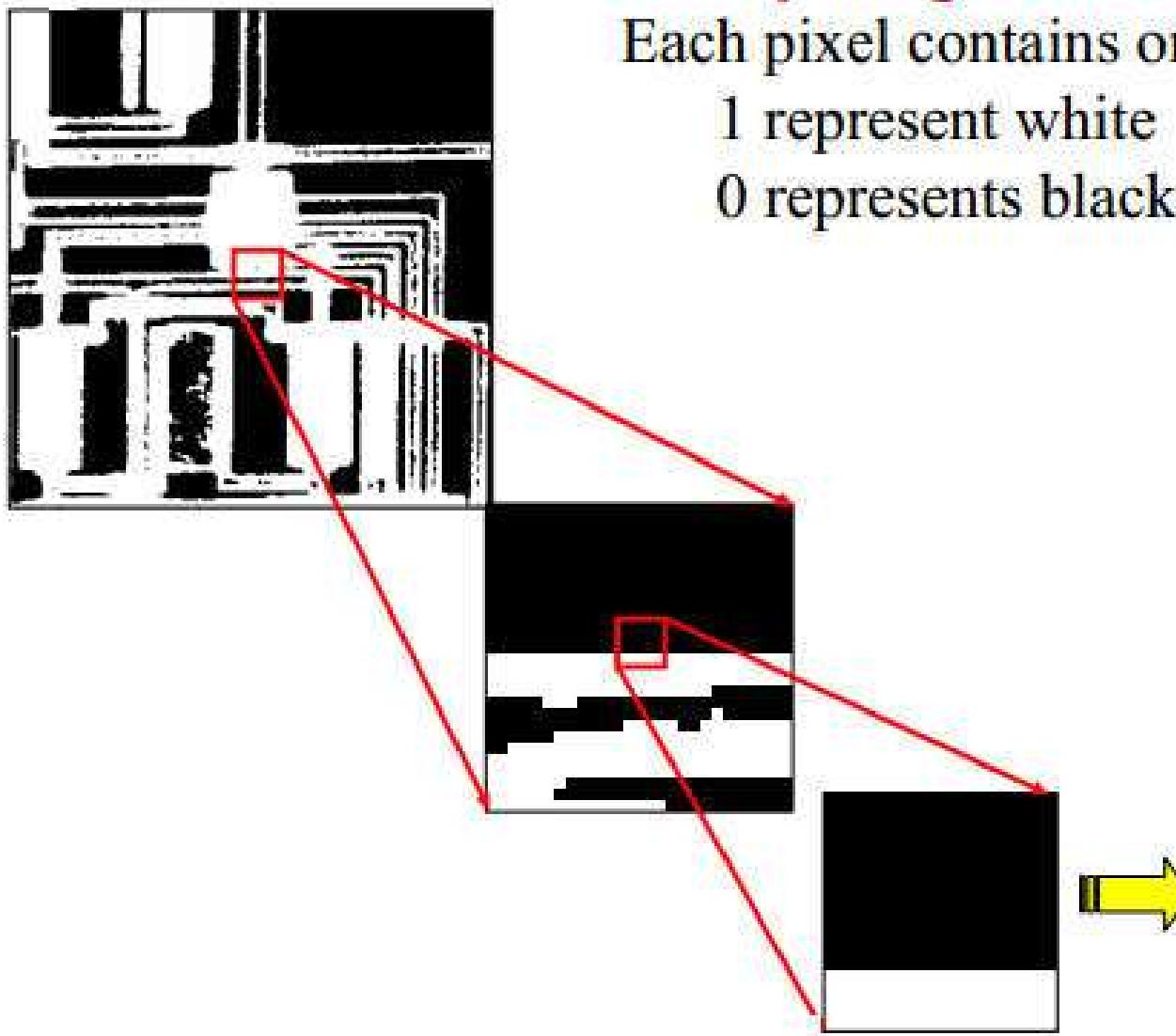
Binary Images

Binary image or black and white image

Each pixel contains one bit :

1 represent white

0 represents black



Binary data

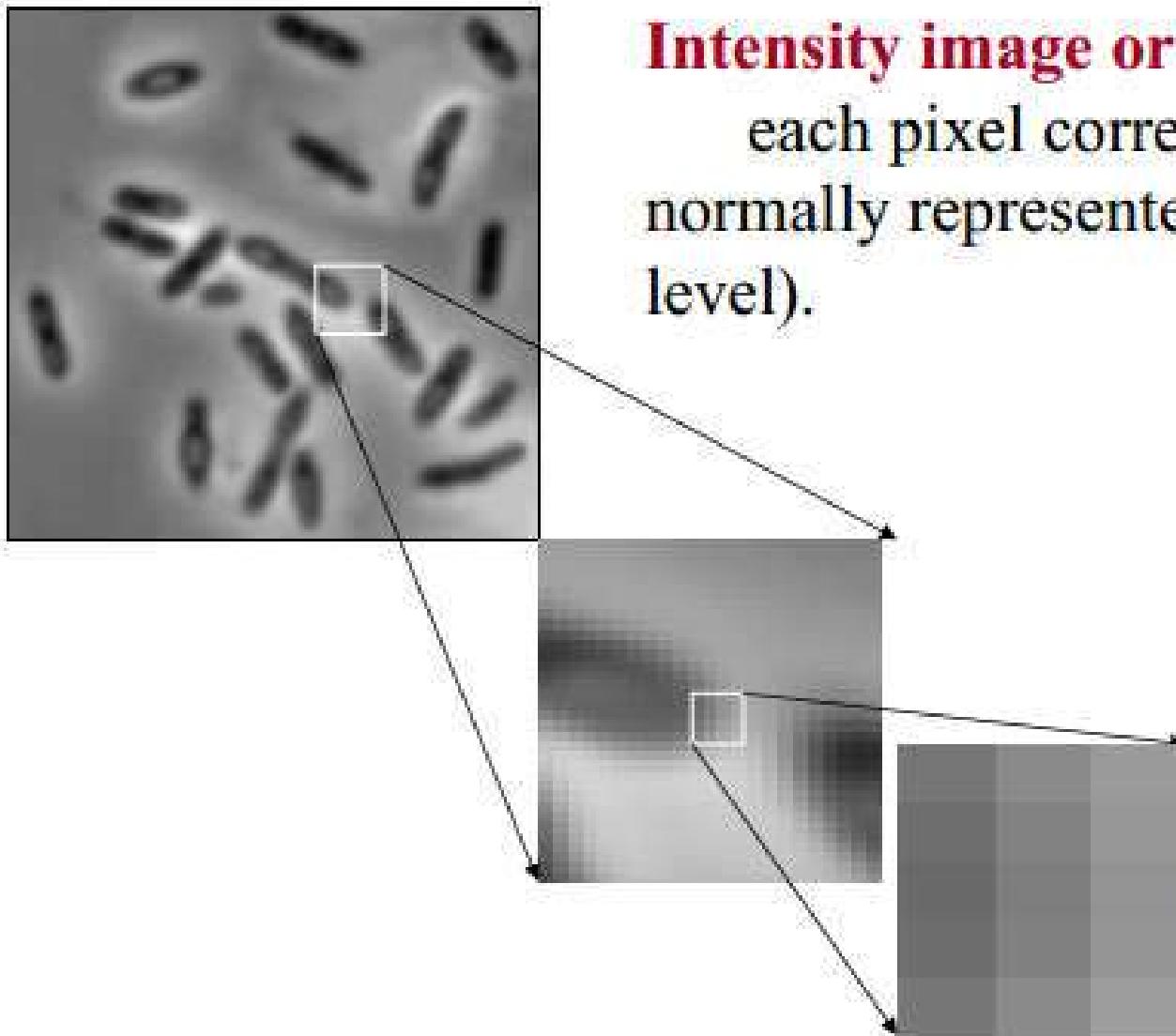
0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	1

Grayscale Images -monochromatic

- A grayscale (or graylevel) image is simply one in which the only colors are shades of gray (0 – 255)



Grayscale Images -monochromatic

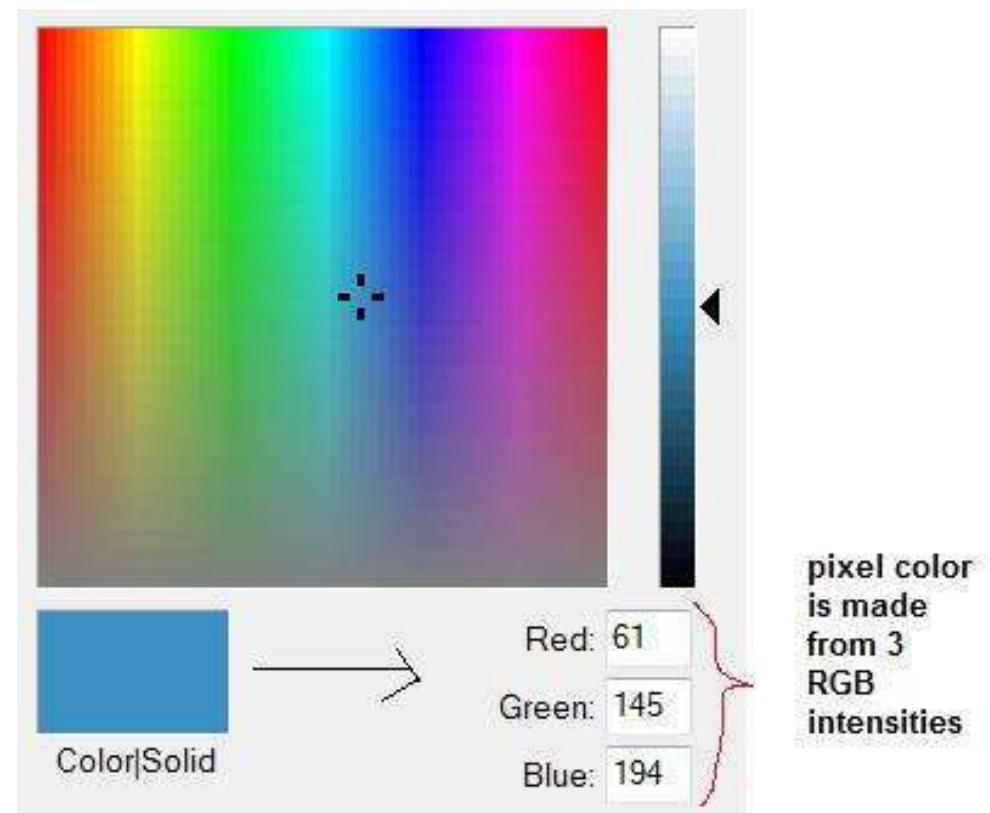


Intensity image or monochrome image

each pixel corresponds to light intensity normally represented in gray scale (gray level).

Color Images - chromatic

- ❑ Color image: A color image contains pixels each of which holds three intensity values corresponding to the red, green, and blue or(RGB)



Color Images - chromatic



Color image or RGB image:
each pixel contains a vector
representing red, green and
blue components.

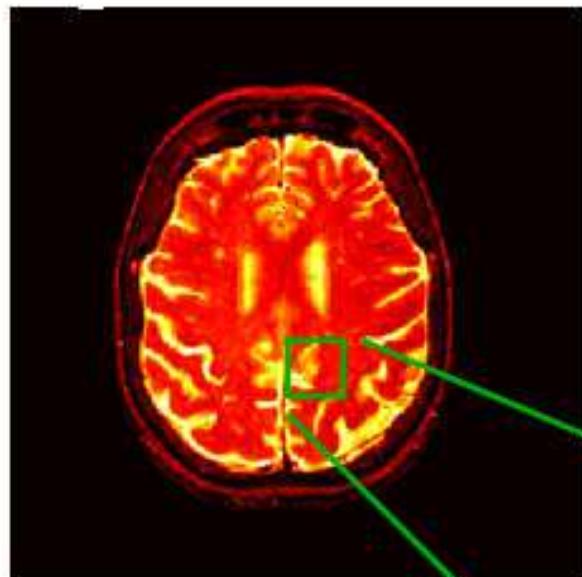
RGB components

10	10	16	28	
9	65	70	56	43
15	32	99	70	56
32	21	60	90	96
	54	85	85	43
		32	65	87
			92	99

Index Image

Index image

Each pixel contains index number pointing to a color in a color table



1	4	9
6	4	7
6	5	2

Index value

Color Table

Index No.	Red component	Green component	Blue component
1	0.1	0.5	0.3
2	1.0	0.0	0.0
3	0.0	1.0	0.0
4	0.5	0.5	0.5
5	0.2	0.8	0.9
...

Representing digital images

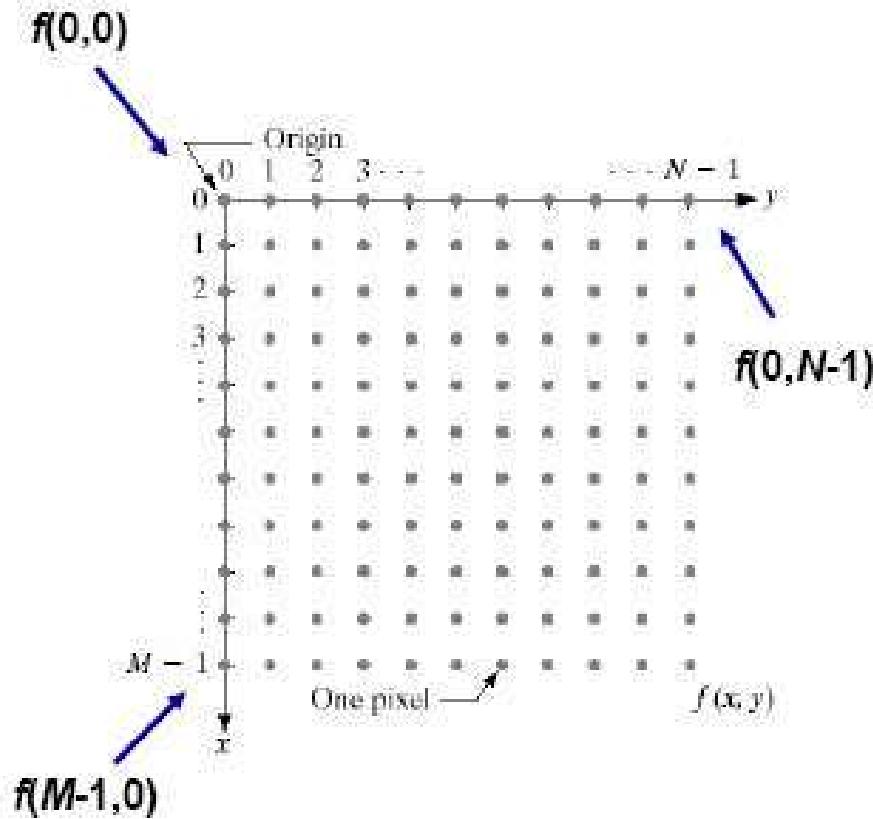


FIGURE 2.18

Coordinate convention used in this book to represent digital images.

L : # of discrete gray levels

$$L = 2^k$$

$$b = M \times N \times k$$

Every pixel has a # of bits.

Digital Image Size

- The size of a digital image is determined by its dimensions ($M \times N$) multiplied by the number of bits k required to store the intensity levels ($L = 2^k$).

$$\text{image size} = M \times N \times k \text{ (bits)}$$

- Typical values of b are:
 - $k = 1$: black and white (binary) images.
 - $k = 8$: grayscale (256 gray levels), or indexed color images
 - $k = 24$: RGB color image.

Representing digital images

The notation introduced in the preceding paragraph allows us to write the complete $M \times N$ digital image in the following compact matrix form:

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N - 1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N - 1) \\ \vdots & \vdots & & \vdots \\ f(M - 1, 0) & f(M - 1, 1) & \cdots & f(M - 1, N - 1) \end{bmatrix}. \quad (2.4-1)$$

The right side of this equation is by definition a digital image. Each element of this matrix array is called an *image element*, *picture element*, *pixel*, or *pel*. The terms *image* and *pixel* will be used throughout the rest of our discussions to denote a digital image and its elements.

In some discussions, it is advantageous to use a more traditional matrix notation to denote a digital image and its elements:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}. \quad (2.4-2)$$

Clearly, $a_{ij} = f(x = i, y = j) = f(i, j)$, so Eqs. (2.4-1) and (2.4-2) are identical matrices.

Pixels!

- Every pixel has # of bits (k)
- Q: Suppose a pixel has 1 bit, how many gray levels can it represent?
Answer: 2 intensity levels only, black and white.
1 Bit (0,1) → 0:black , 1: white
- Q: Suppose a pixel has 2 bit, how many gray levels can it represent?
Answer: 4 gray intensity levels
2 Bit (00, 01, 10 ,11).

Now ..

if we want to represent 256 intensities of grayscale, how many bits do we need?

Answer: 8 bits → which represents: $2^8=256$

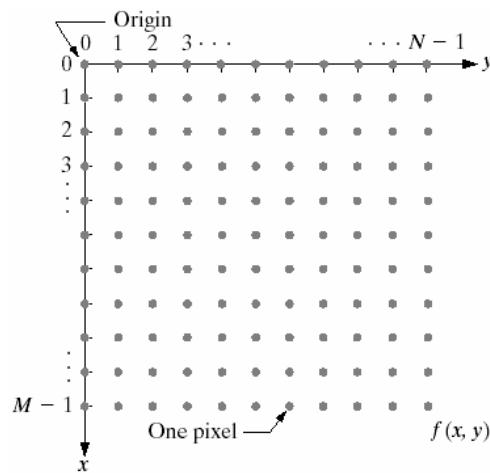
so, the gray intensities (L) that the pixel can hold, is calculated according to according to number of bits it has (k).

$$L = 2^k$$

and they are integers in the interval **[0, L-1]**

Image Size

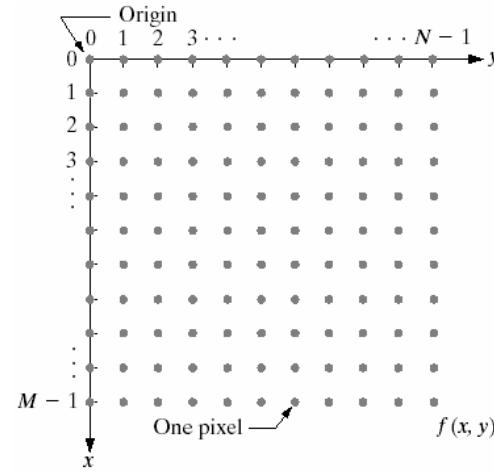
(200 × 200) 8-bit grayscale image



256 intensities

Size= 200×200×8
size= 320,000 bit

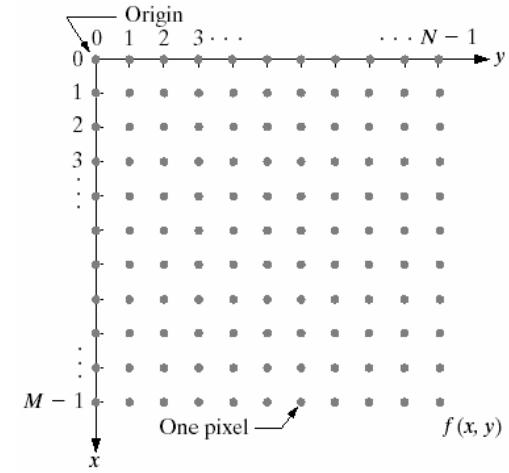
(200 × 200) 1-bit binary image



2 intensities:
Black(0) and white(1)

Size= 200×200×1
size= 40,000 bit

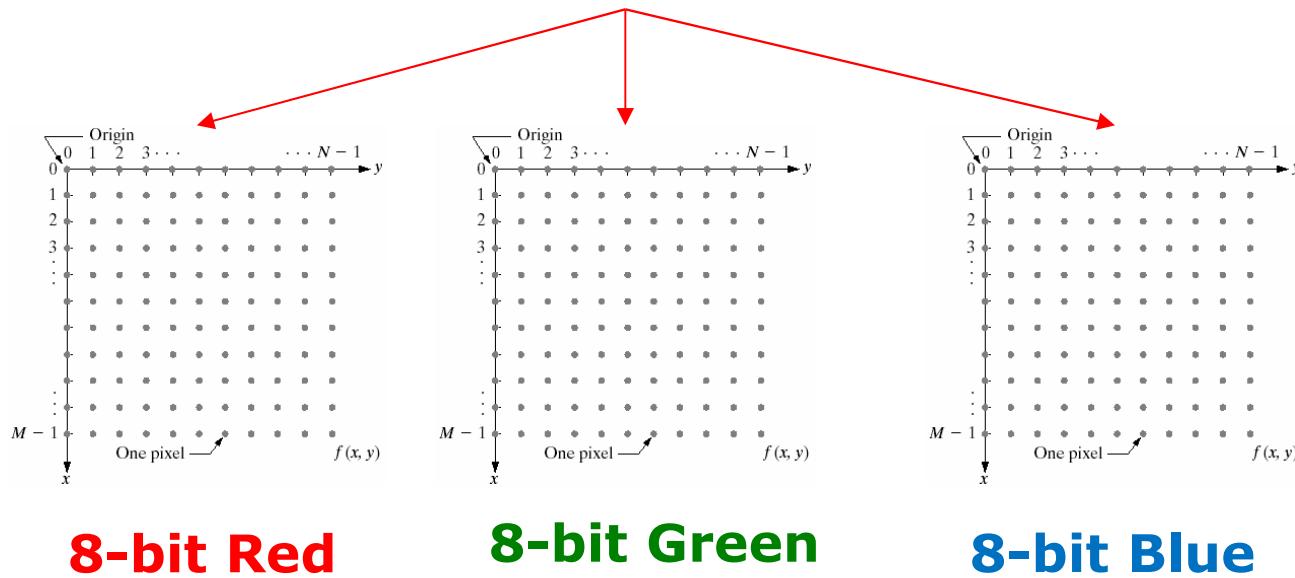
(200 × 200) 8-bit binary image



2 intensities:
Black(0) and white(255)
Size= 200×200×8
size= 320,000 bit

Image Size

(200 × 200) 24-bit color image



Size= 200×200×8×3
Size= 960,000 bit

Number of storage of bits:

$N * M$: the no. of pixels in all the image.

K: no. of bits in each pixel

L: grayscale levels the pixel can represent

$$L = 2^k$$

$$\text{all bits in image} = N * N * k$$

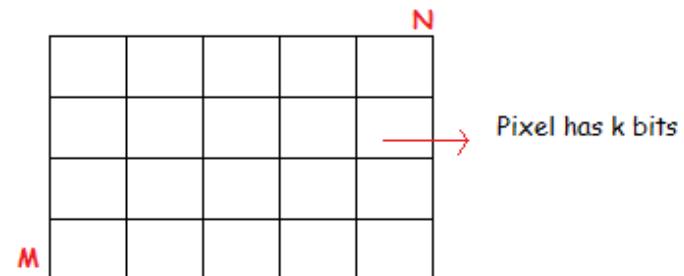


TABLE 2.1

Number of storage bits for various values of N and k .

$$\begin{aligned}\text{NO of pixels} &= N * M \\ \text{NO of bits} &= N * M * k\end{aligned}$$

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

Number of storage of bits:

EX: Here: N=32, K=3, L = $2^3 = 8$

of pixels=N*N = 1024 . (because in this example: M=N)

of bits = N*N*K = 1024*3= 3072

N/k	1 (L = 2)	2 (L = 4)	3 (L = 8)	4 (L = 16)	5 (L = 32)	6 (L = 64)	7 (L = 128)	8 (L = 256)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

N=M in this table, which means no. of horizontal pixels= no. of vertical pixels. And thus:

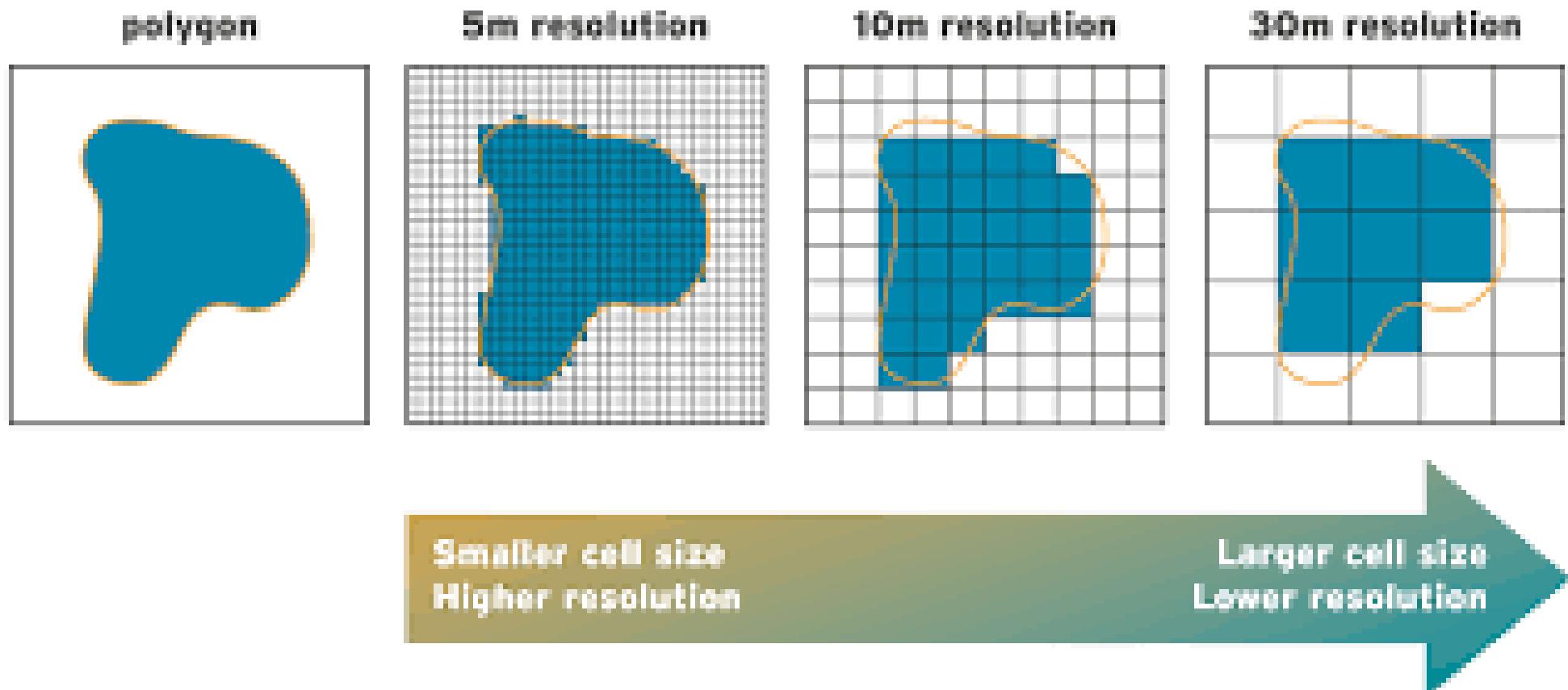
of pixels in the image= N*N

SPATIAL AND INTENSITY RESOLUTION

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
 - Vision specialists will often talk about pixel size
 - Graphic designers will talk about dots per inch (DPI)

Spatial Resolution

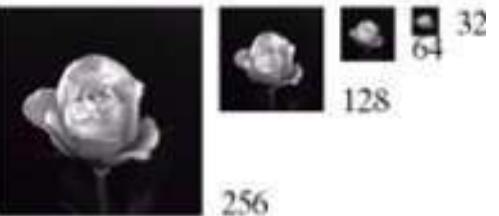


Reducing Spatial Resolution (subsampling)

The **lower resolution** images are smaller than the original



1024



Same # of bits in all
images (same gray level)
different # of pixels

subSampling is performed by deleting rows and columns from the original image.

Spatial and gray-level resolution

Re sampling

(pixel replication)

A special case of nearest neighbor zooming.

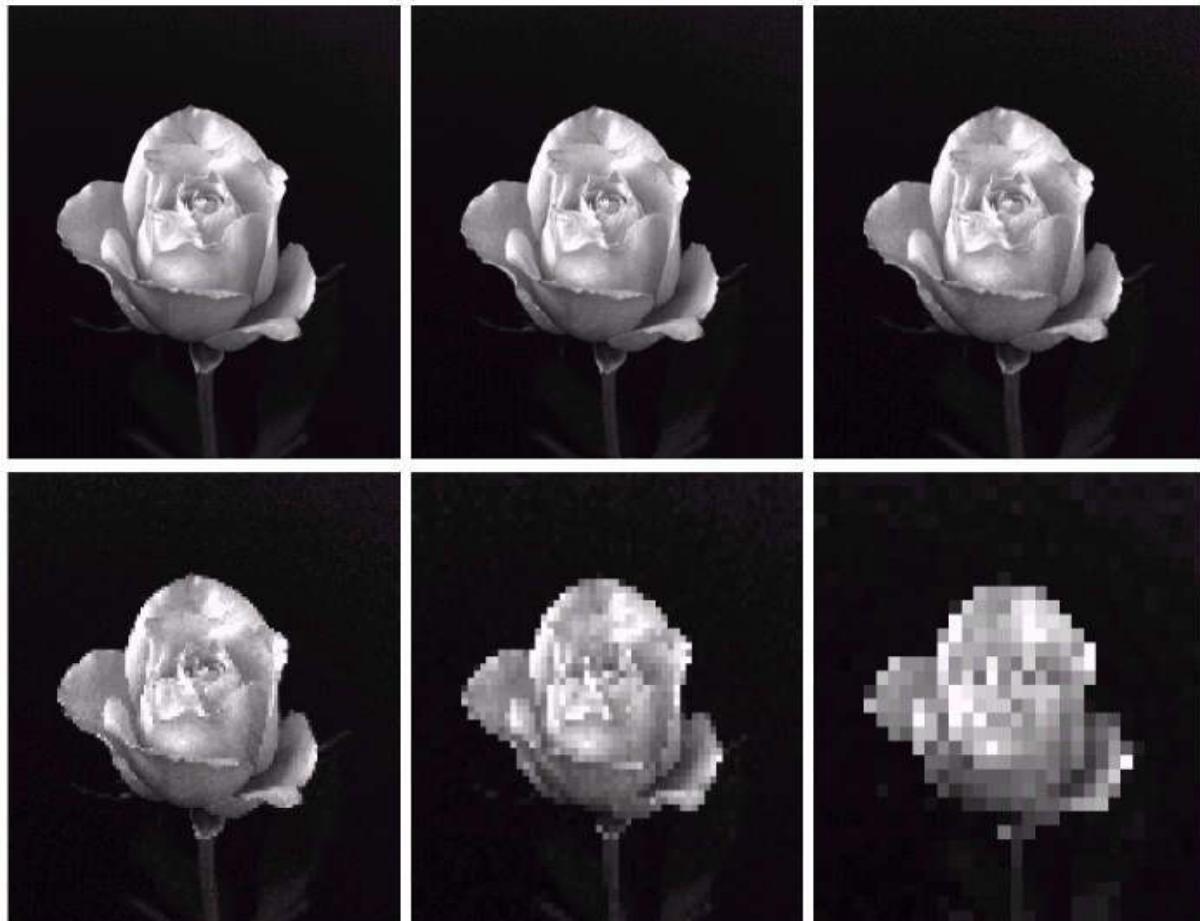


FIGURE 2.20 (a) 1024×1024 , 8-bit image. (b) 512×512 image resampled into 1024×1024 pixels by row and column duplication. (c) through (f) 256×256 , 128×128 , 64×64 , and 32×32 images resampled into 1024×1024 pixels.

Resampling is performed by row and column duplication

Spatial Resolution (cont...)



Spatial Resolution (cont...)



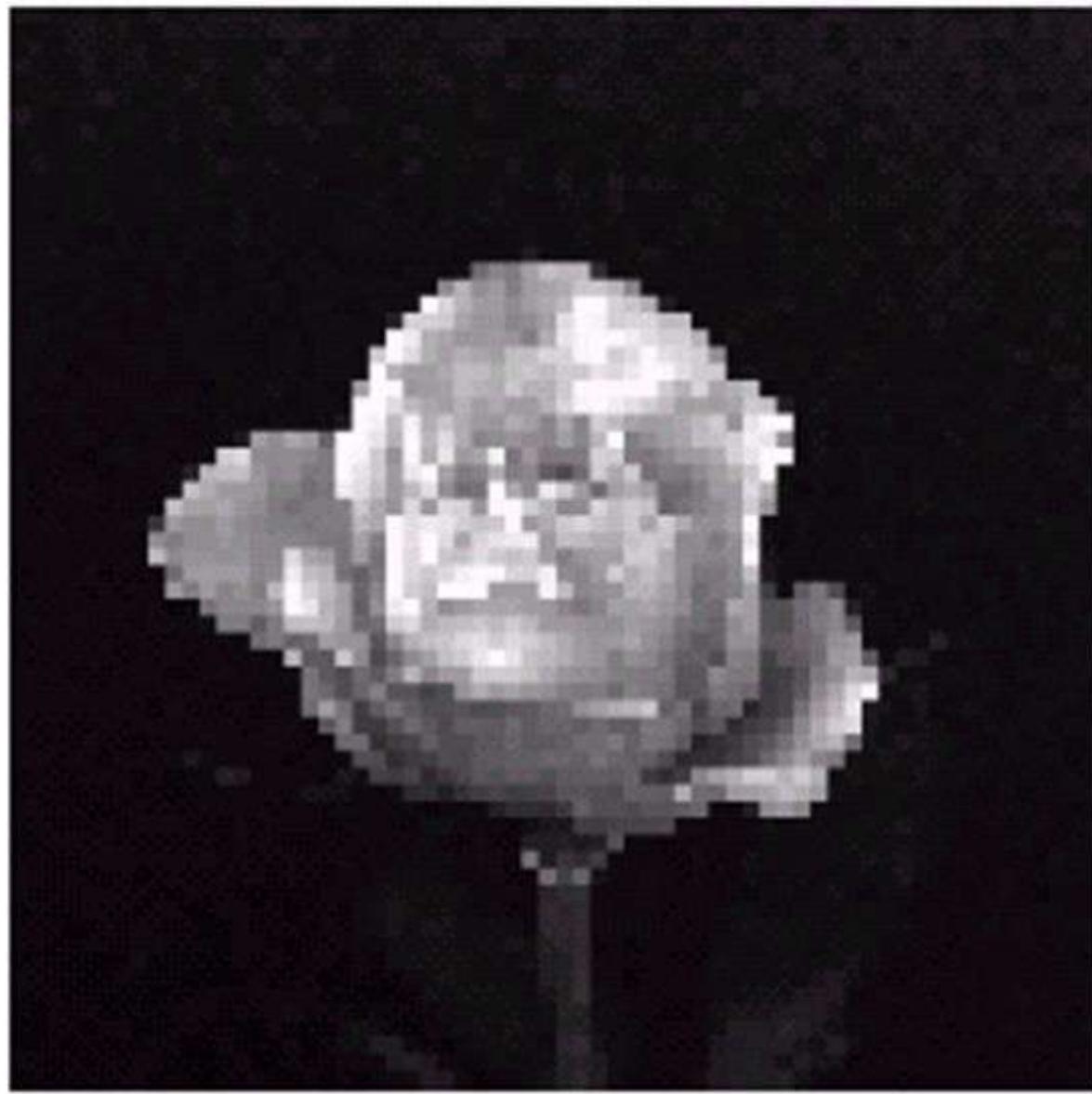
Spatial Resolution (cont...)



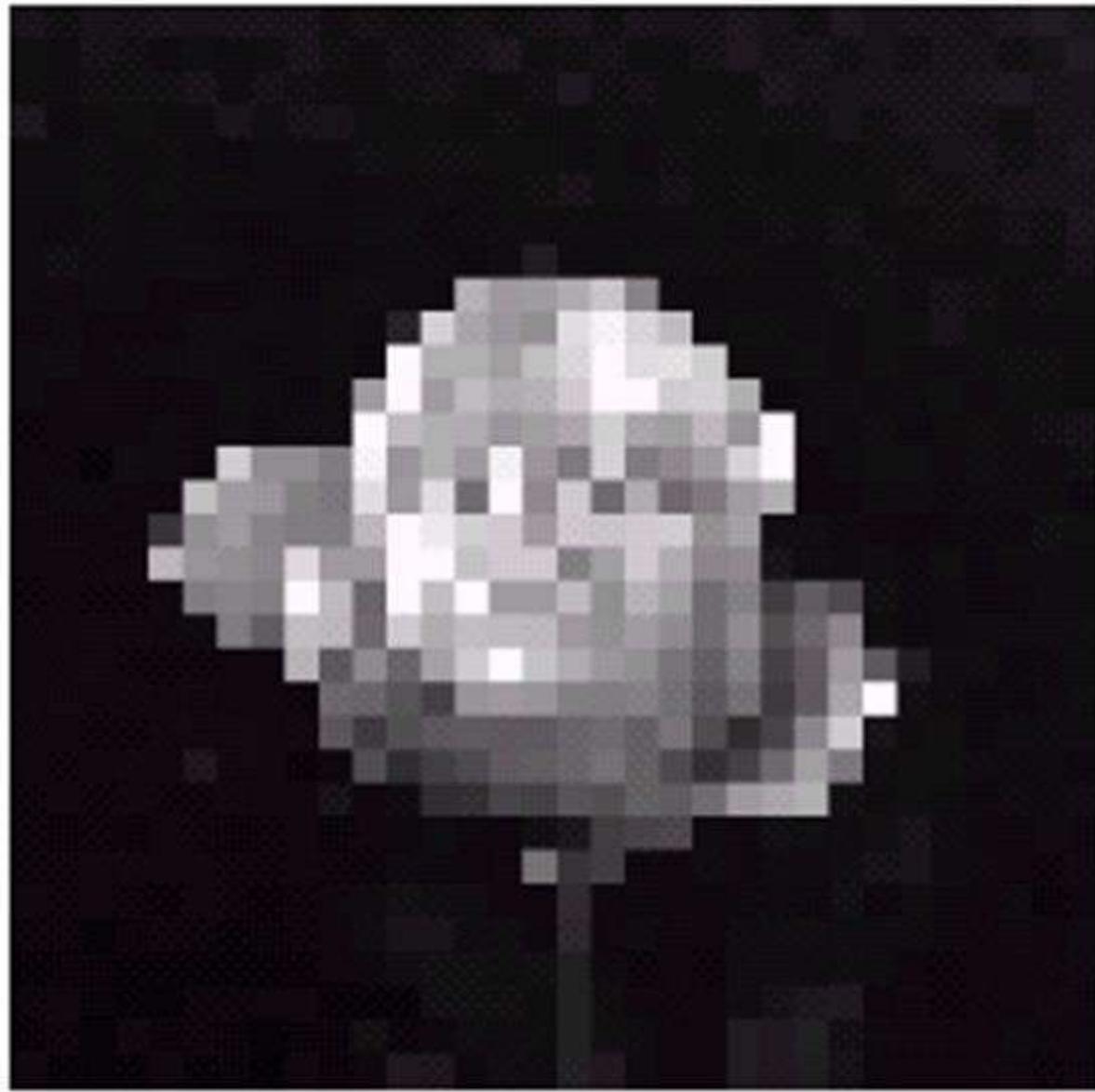
Spatial Resolution (cont...)



Spatial Resolution (cont...)



Spatial Resolution (cont...)



Checkerboard Effect

- ❑ **Checkerboard effect** caused when spatial resolution of images are very low.
- ❑ If we compare the original image with the other images, we find checkerboard patterns at the edges.
- ❑ This effect is visible in the 128×128 and more pronounced in 64×64 and 32×32 .



Checkerboard Effect

Checker board effect is observed by leaving unchanged the number of grey levels and varying the spatial resolution while keeping the display area unchanged. The checkerboard effect is caused by pixel replication, that is, lower resolution images were duplicated in order to fill the display area.

Intensity 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

Reducing IR

Here we keep the number of samples constant and **reduce** the number of intensity levels

In these images, the # of samples is constant but the # of gray levels was reduced from 256 to 32.

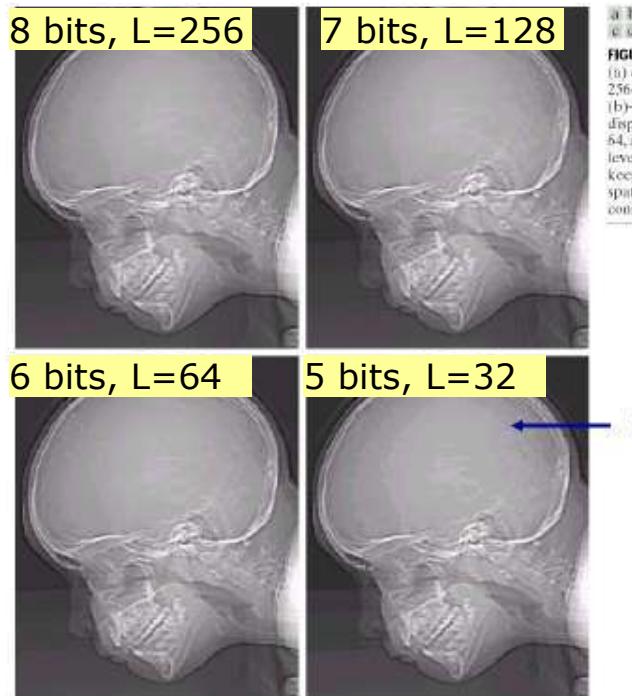


FIGURE 2.21
(a) 452 × 374,
256-level image.
(b)–(d) Image
displayed in 128,
64, and 32 gray
levels, while
keeping the
spatial resolution
constant.

In these images, the # of samples is constant but the # of gray levels was reduced from 16 to 2.

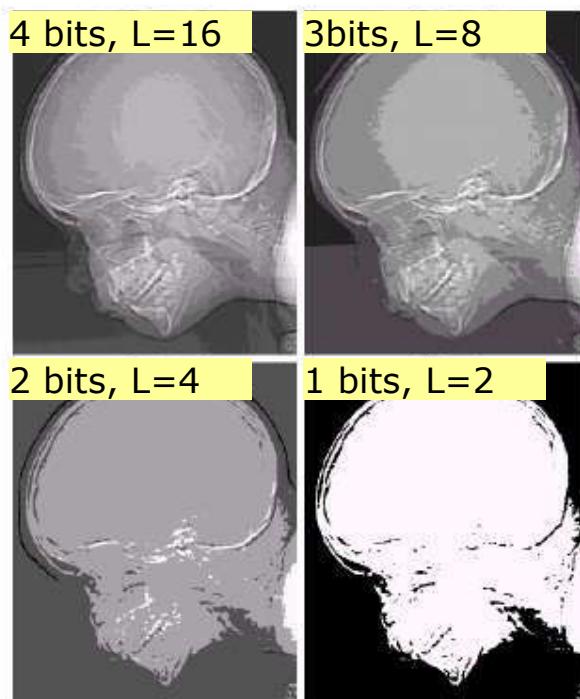


FIGURE 2.21
(Continued)
(e)–(h) Image
displayed in 16, 8,
4, and 2 gray
levels. (Original
courtesy of
Dr. David
R. Pickens,
Department of
Radiology &
Radiological
Sciences,
Vanderbilt
University
Medical Center.)

In this example,

all images have 452*374 pixels, but different # of bits per pixel (as shown in yellow)

different # of bits in all images
(different gray level)

same # of pixels

👉 Remember that:

every pixel has # of bits k

And # of gray levels is $L=2^k$

More pixels → more resolution

More bit/pixel → more accuracy

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)



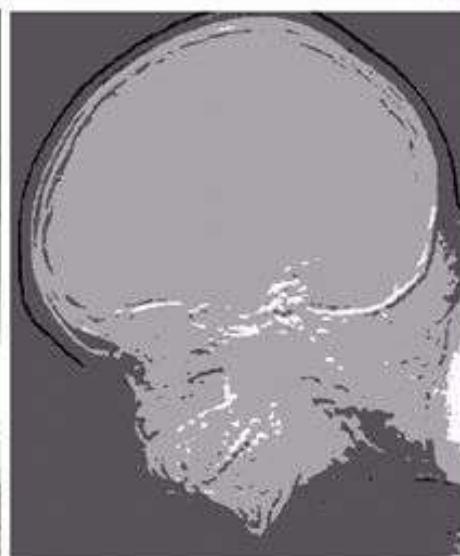
16 grey levels (4 bpp)



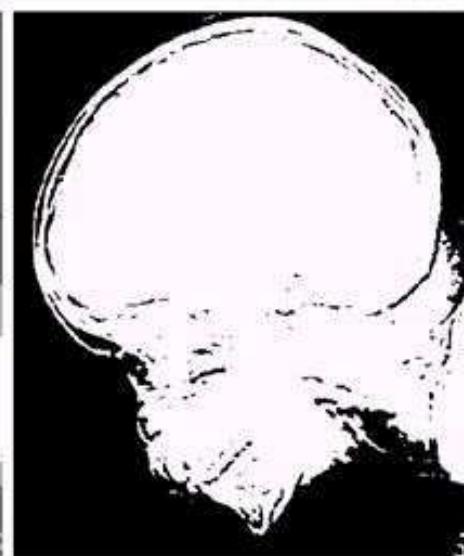
8 grey levels (3 bpp)



4 grey levels (2 bpp)



2 grey levels (1 bpp)



Intensity Level Resolution (cont...)



Intensity Level Resolution (cont...)



Intensity Level Resolution (cont...)



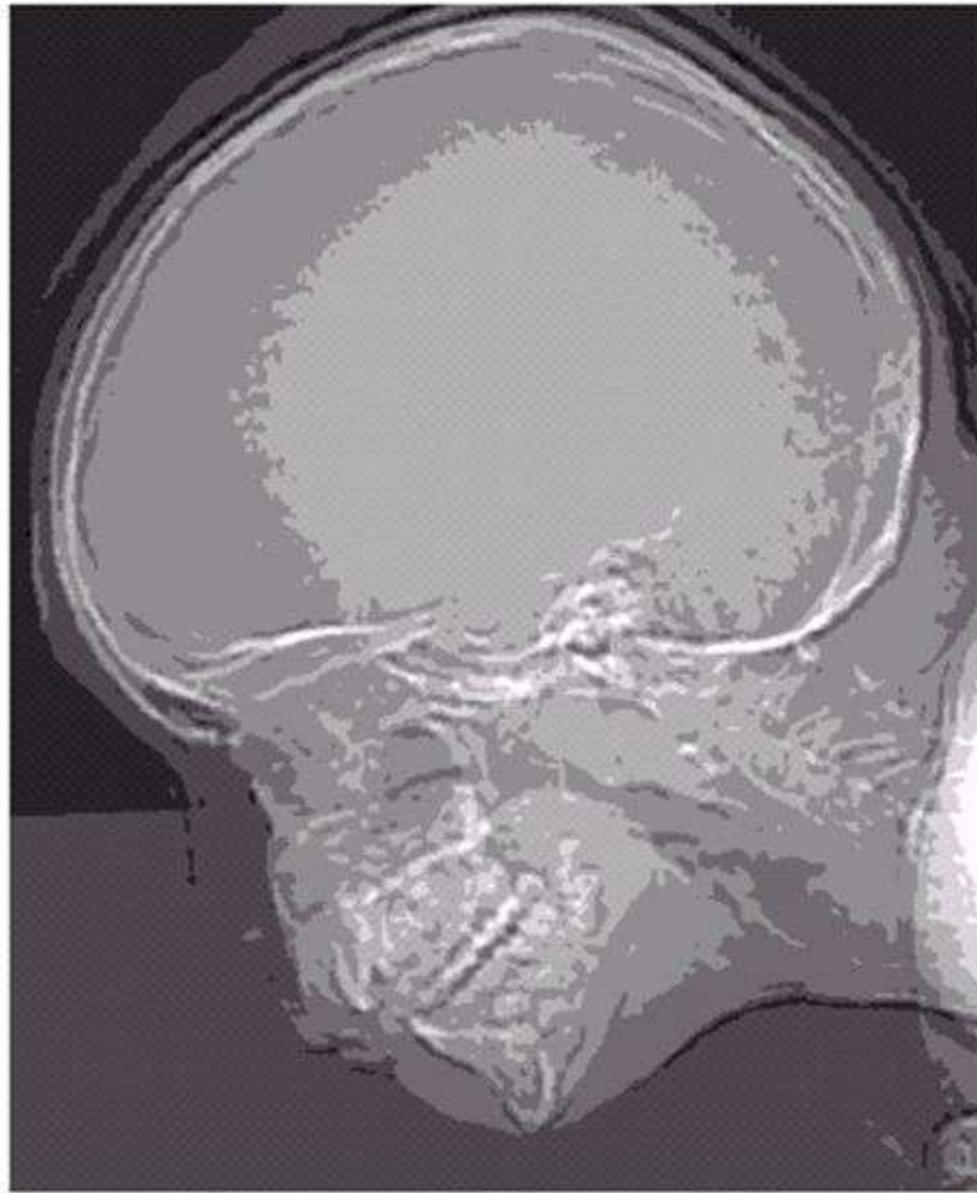
Intensity Level Resolution (cont...)



Intensity Level Resolution (cont...)



Intensity Level Resolution (cont...)



Intensity Level Resolution (cont...)



Intensity Level Resolution (cont...)



False Contouring

- ▣ **Effect of False Contouring** – Under the low intensity resolution it has an imperceptible set of very fine ridge like structure in areas of smooth gray levels (particularly in the skull).
- ▣ This effect cause by the use of an **insufficient number of gray levels in smooth areas** of a digital image, is called False Contouring.
- ▣ False contouring is generally quite visible in images displayed using 16 or less uniformly spaced gray levels

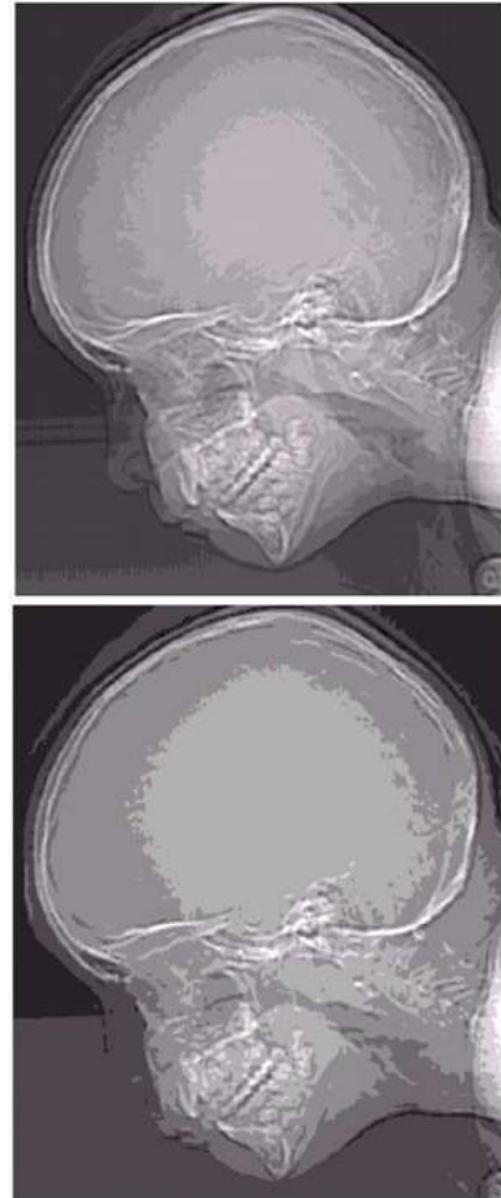


IMAGE INTERPOLATION

Image Interpolation

Image Interpolation is the process of using known data to estimate values at unknown location.

Image interpolation is used for zooming, shrinking, rotating, geometric corrections.

- Zooming + shrinking are Image **resizing** tasks and come under image re-sampling methods and we will study them in the following

Zooming (over sampling) images

Zooming requires 2 steps:

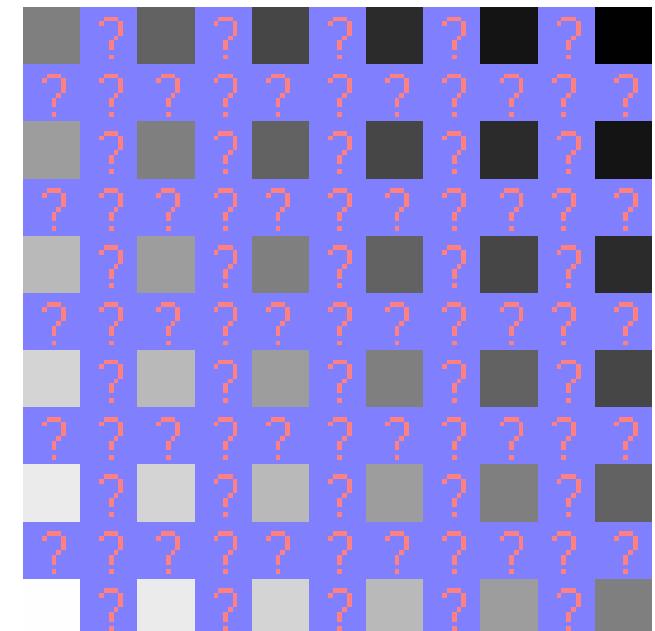
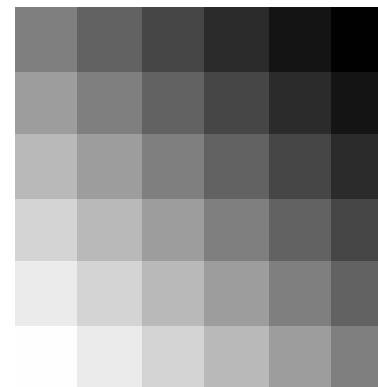
- The creation of new pixel locations.
- The assignment of gray levels to these new locations.

Two techniques for zooming:

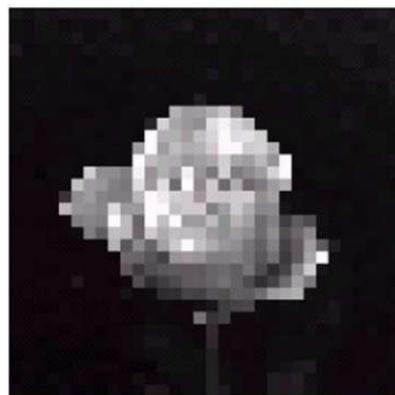
1. Nearest neighbor interpolation
2. Bilinear interpolation
3. Bicubic interpolation

Nearest Neighbor Interpolation

- The creation of new pixel locations.
- The assignment of gray levels to these new locations.



+ ve : Nearest neighbor is fast
-ve: it produces a
checkerboard effect like this!

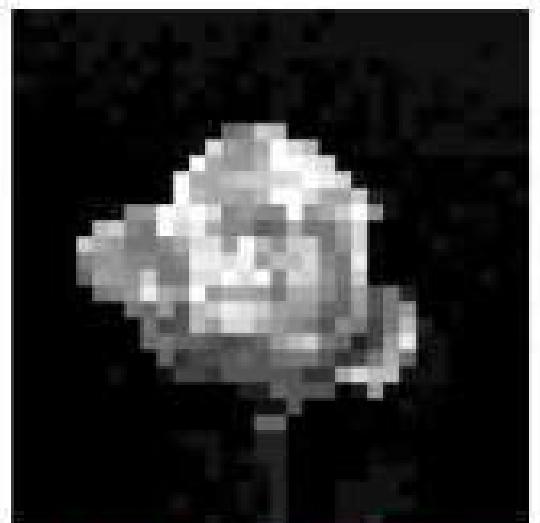




128x128 \Rightarrow *1024x1024*



64x64 \Rightarrow *1024x1024*



32x32 \Rightarrow *1024x1024*

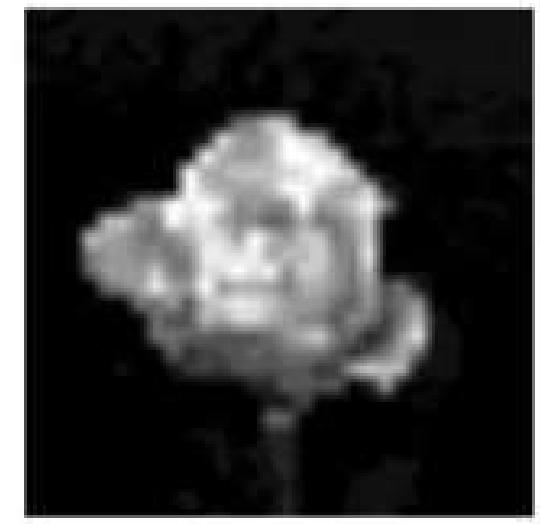
Fig: Images zoomed from 128 x 128, 64 x 64 and 32 x 32 pixels to 1024 x 1024 pixels using nearest neighbor grey-level interpolation.



128x128 \Rightarrow *1024x1024*



64x64 \Rightarrow *1024x1024*

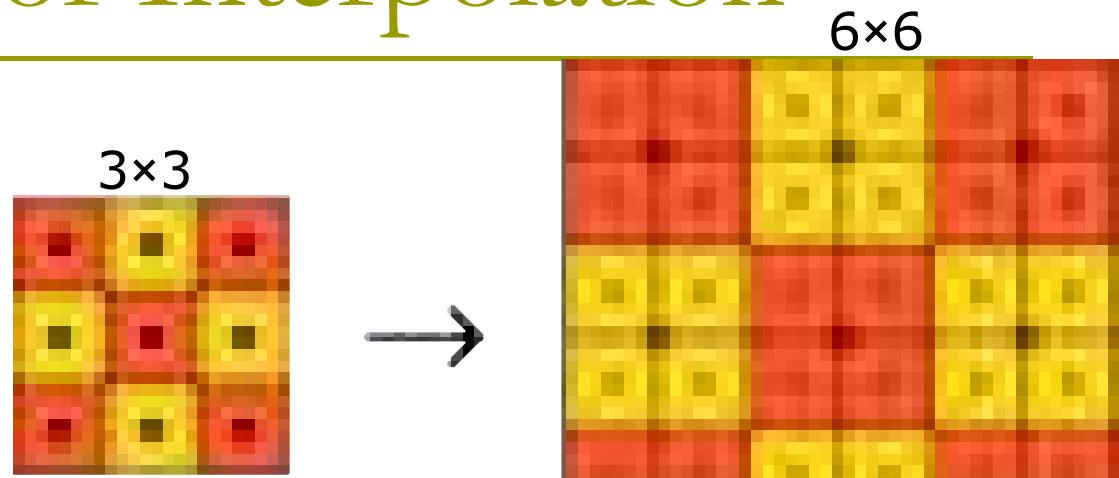


32x32 \Rightarrow *1024x1024*

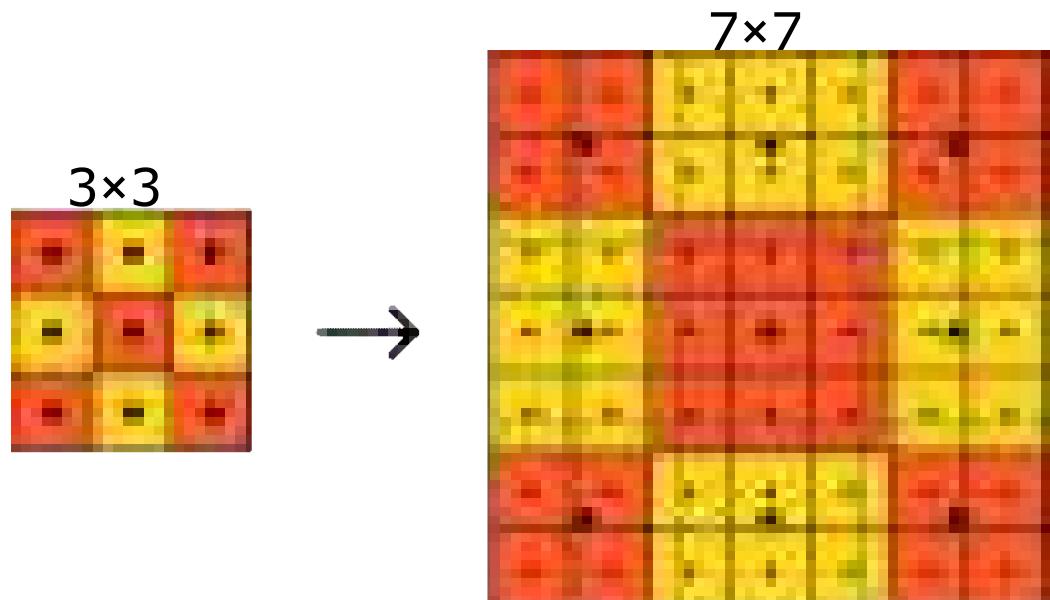
Fig: Images zoomed from 128 x 128, 64 x 64 and 32 x 32 pixels to 1024 x 1024 pixels using bilinear interpolation.

Nearest Neighbor Interpolation

- First, the texture is taken, represented by a point at the center of each of the pixels.



- Then stretched to the size of the final scaled image and overlaid onto it.



- Next, for each point in the scaled image, the nearest point in the texture is found.

Nearest Neighbor Interpolation

Nearest Neighbor Example

$$I(R \times C) = \begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) \\ (2,1) & (2,2) & (2,3) & (2,4) \\ (3,1) & (3,2) & (3,3) & (3,4) \\ (4,1) & (4,2) & (4,3) & (4,4) \end{bmatrix}$$

R=r & C=c

$$S_r = \frac{R}{\bar{R}} = 0.5 \quad S_c = \frac{C}{\bar{C}} = 0.5$$

$$J(\hat{R} \times \hat{C}) = \begin{bmatrix} (1,1) & (1,2) & (1,3) & (1,4) & (1,5) & (1,6) & (1,7) & (1,8) \\ (2,1) & (2,2) & (2,3) & (2,4) & (2,5) & (2,6) & (2,7) & (2,8) \\ (3,1) & (3,2) & (3,3) & (3,4) & (3,5) & (3,6) & (3,7) & (3,8) \\ (4,1) & (4,2) & (4,3) & (4,4) & (4,5) & (4,6) & (4,7) & (4,8) \\ (5,1) & (5,2) & (5,3) & (5,4) & (5,5) & (5,6) & (5,7) & (5,8) \\ (6,1) & (6,2) & (6,3) & (6,4) & (6,5) & (6,6) & (6,7) & (6,8) \\ (7,1) & (7,2) & (7,3) & (7,4) & (7,5) & (7,6) & (7,7) & (7,8) \\ (8,1) & (8,2) & (8,3) & (8,4) & (8,5) & (8,6) & (8,7) & (8,8) \end{bmatrix}$$

$$[(r_f, c_f)] = [(S_r * \hat{r}, S_c * \hat{c})] =$$

$$\begin{bmatrix} (0.5, 0.5) & (0.5, 1) & (0.5, 1.5) & (0.5, 2) & (0.5, 2.5) & (0.5, 3) & (0.5, 3.5) & (0.5, 4) \\ (1, 0.5) & (1, 1) & (1, 1.5) & (1, 2) & (1, 2.5) & (1, 3) & (1, 3.5) & (1, 4) \\ (1.5, 0.5) & (1.5, 1) & (1.5, 1.5) & (1.5, 2) & (1.5, 2.5) & (1.5, 3) & (1.5, 3.5) & (1.5, 4) \\ (2, 0.5) & (2, 1) & (2, 1.5) & (2, 2) & (2, 2.5) & (2, 3) & (2, 3.5) & (2, 4) \\ (2.5, 0.5) & (2.5, 1) & (2.5, 1.5) & (2.5, 2) & (2.5, 2.5) & (2.5, 3) & (2.5, 3.5) & (2.5, 4) \\ (3, 0.5) & (3, 1) & (3, 1.5) & (3, 2) & (3, 2.5) & (3, 3) & (3, 3.5) & (3, 4) \\ (3.5, 0.5) & (3.5, 1) & (3.5, 1.5) & (3.5, 2) & (3.5, 2.5) & (3.5, 3) & (3.5, 3.5) & (3.5, 4) \\ (4, 0.5) & (4, 1) & (4, 1.5) & (4, 2) & (4, 2.5) & (4, 3) & (4, 3.5) & (4, 4) \end{bmatrix}$$

$$[(\bar{r}, \bar{c})] = [Round\{(r_f, c_f)\}] =$$

$$\begin{bmatrix} (1,1) & (1,1) & (1,2) & (1,2) & (1,3) & (1,3) & (1,4) & (1,4) \\ (1,1) & (1,1) & (1,2) & (1,2) & (1,3) & (1,3) & (1,4) & (1,4) \\ (2,1) & (2,1) & (2,2) & (2,2) & (2,3) & (2,3) & (2,4) & (2,4) \\ (2,1) & (2,1) & (2,2) & (2,2) & (2,3) & (2,3) & (2,4) & (2,4) \\ (3,1) & (3,1) & (3,2) & (3,2) & (3,3) & (3,3) & (3,4) & (3,4) \\ (3,1) & (3,1) & (3,2) & (3,2) & (3,3) & (3,3) & (3,4) & (3,4) \\ (4,1) & (4,1) & (4,2) & (4,2) & (4,3) & (4,3) & (4,3.5) & (4,4) \\ (4,1) & (4,1) & (4,2) & (4,2) & (4,3) & (4,3) & (4,4) & (4,4) \end{bmatrix}$$

$$J(\hat{r}, \hat{c}) = I(\bar{r}, \bar{c})$$

Nearest Neighbor Interpolation

Example: Suppose A 2x2 pixels image will be enlarged 2 times by the nearest neighbor method:

1. Stretch the original image to the size of the new scaled image
2. overlaid the stretched grid of the original image onto the new scaled image.
3. For any point in the overlay, look for the closest pixel in the original image, and assign its gray level to the new pixel in the grid. (copy)

50	100
180	250

original image

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

New scaled image

Nearest Neighbor Interpolation

50	100
180	250

original image

50	100
180	250

180	250
-----	-----

Stretch the original image to the size of the new scaled image

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

New scaled image

?	?	?	?
50	100	?	?
?	?	?	?
?	?	?	?

180	250	?	?
?	?	?	?
?	?	?	?
?	?	?	?

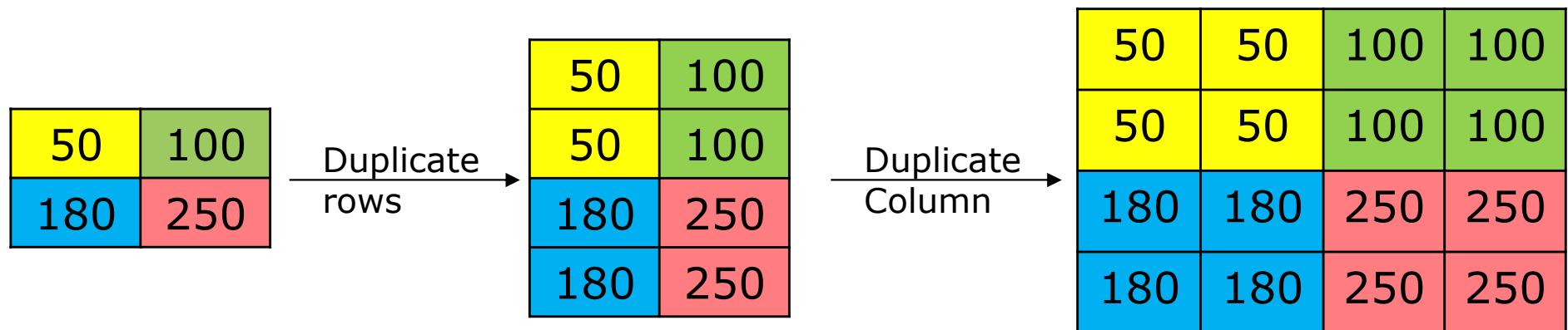
overlaid the stretched original image onto it.

Next, for each point in the scaled image, the nearest point in the texture is found.

50	50	100	100
50	50	100	100
180	180	250	250
180	180	250	250

Nearest Neighbor Interpolation

- ❑ Pixel replication (re sampling) is a special case that is applicable when the size of the image needs to be increased an integer number of times (like 2 times not 1.5 for example).
- ❑ **Example:** if we want to enlarge the following image from 2×2 into 4×4



Shrinking

- Similar to image zooming.

Shrinking an image an integer number of times

- Pixel replication is replaced by row&column deletion.

Bilinear Interpolation:

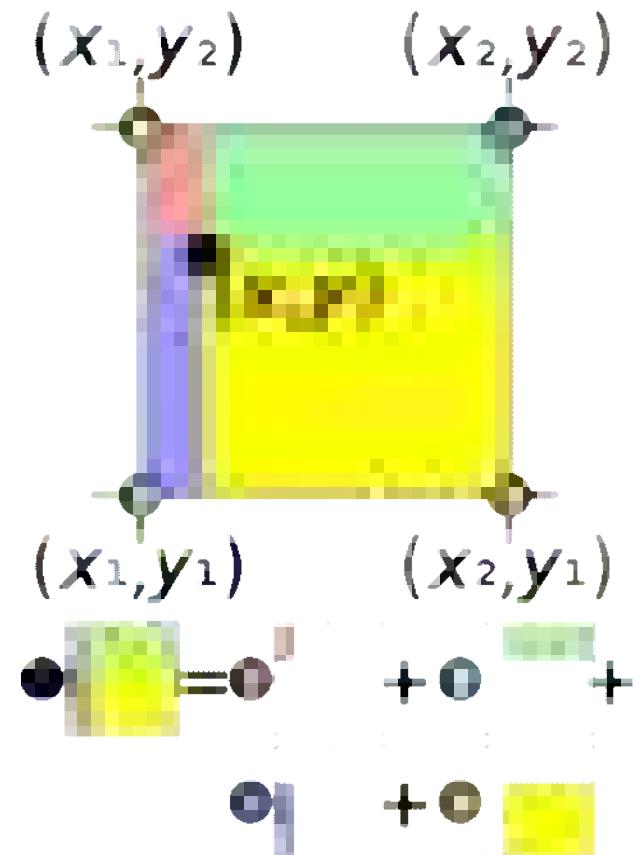
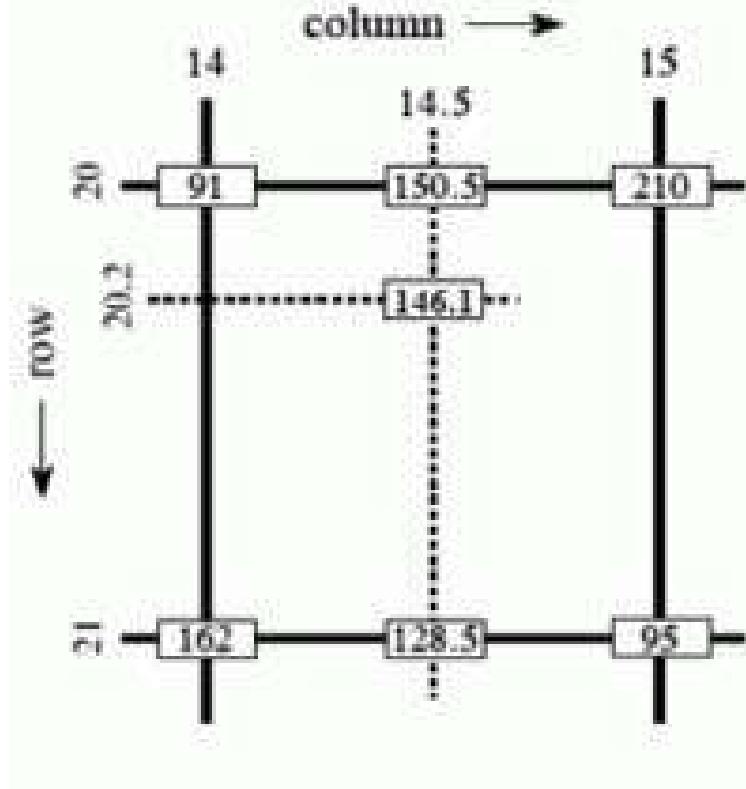
- Here we use the **4 nearest neighbours** to estimate the intensity at a given location.
- Let (x,y) denote the **coordinates** of the location to which we want to assign an intensity value and let $v(x,y)$ denote that value, then:

$$v(x,y) = ax + by + cxy + d$$

- Here the 4 coefficients are determined from the 4 equations in 4 unknowns using the 4 nearest neighbours of point (x,y) .

Bilinear Interpolation

- ❑ The closer point has more influence than the farther point.
- ❑ Thus, the weights are normalized distances between the unknown point and each of the end points.



Bilinear Interpolation

Bilinear Interpolation

Example

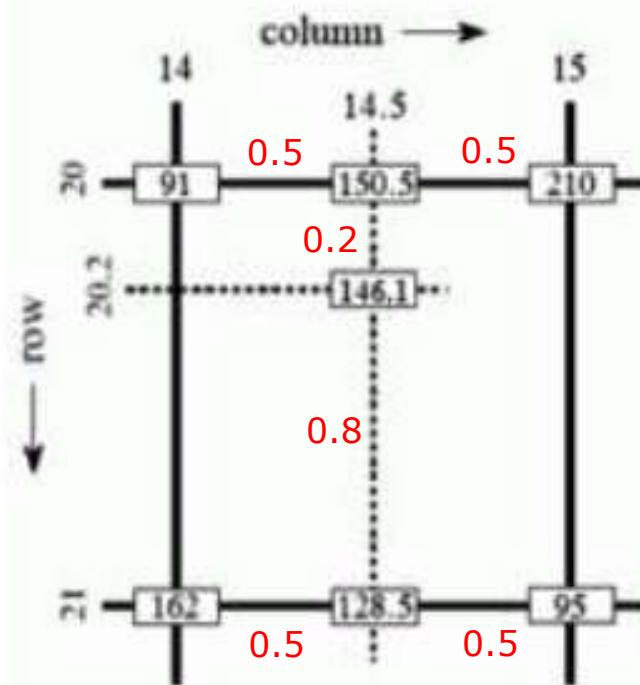
- we observe that the intensity value at the pixel computed to be at row 20.2, column 14.5 can be calculated by first linearly interpolating between the values at column 14 and 15 on each rows 20 and 21, giving

$$I_{20,14.5} = \frac{15 - 14.5}{15 - 14} * 91 + \frac{14.5 - 14}{15 - 14} * 210 = 150.5$$

$$I_{21,14.5} = \frac{15 - 14.5}{15 - 14} * 162 + \frac{14.5 - 14}{15 - 14} * 95 = 128.5$$

and then interpolating linearly between these values, giving

$$I_{20.2,14.5} = \frac{21 - 20.2}{21 - 20} * 150.5 + \frac{20.2 - 20}{21 - 20} * 128.5 = 146.1$$



Bicubic Interpolation:

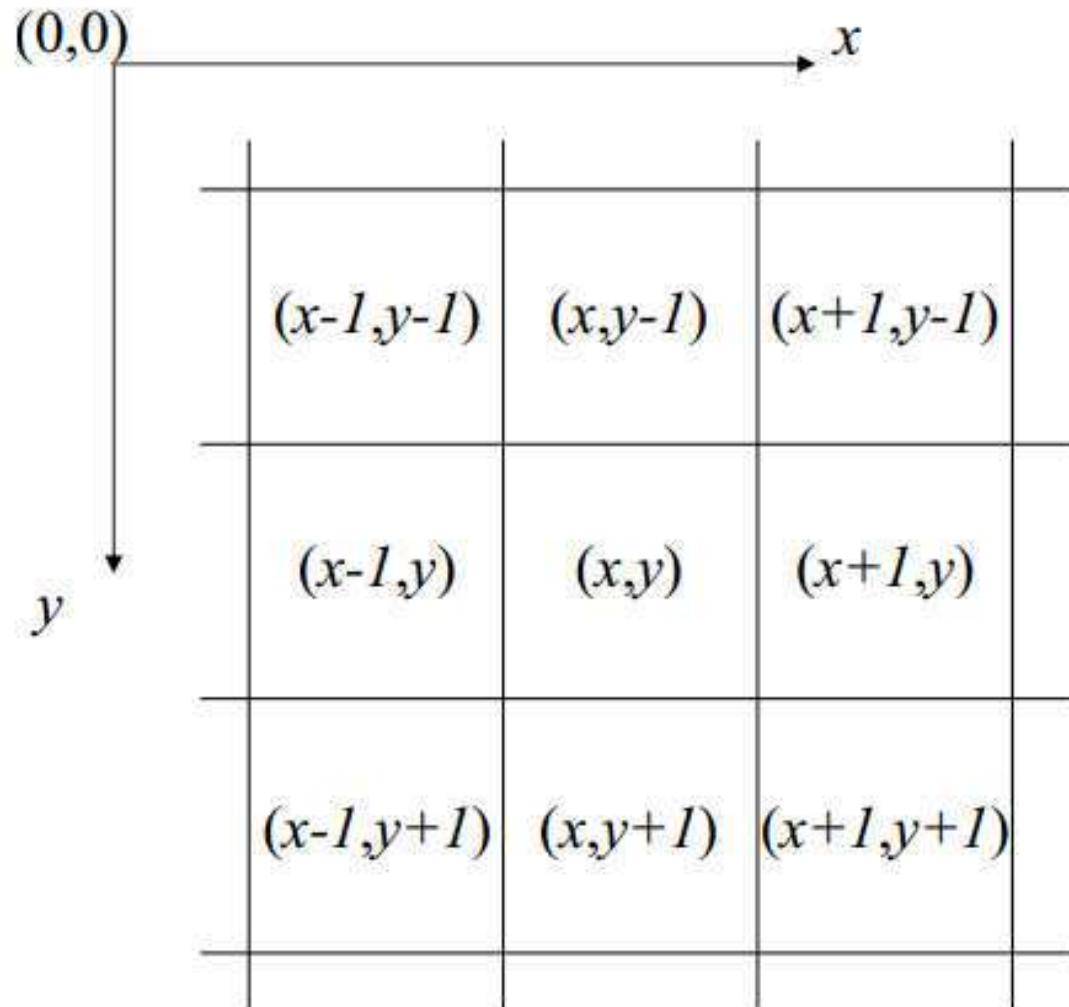
- ❑ Involves 16 nearest neighbours of a point.

$$v(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j$$

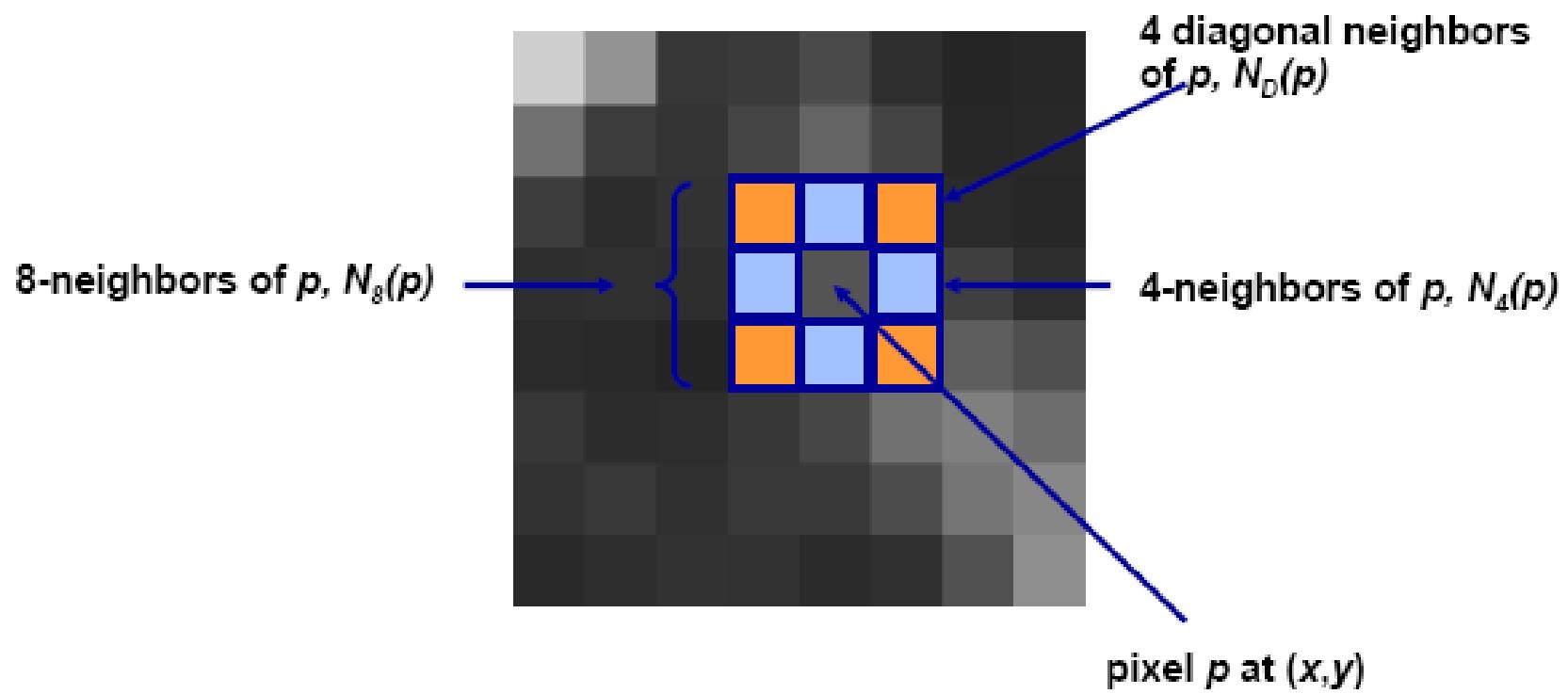
- ❑ **Bicubic Interpolation** method determines the gray level value (or color) from the weighted average of the 16 closest pixels to the specified input coordinates.
- ❑ The image is sharper and more clarity than that produced by Nearest neighbor and Bilinear Interpolation .
- ❑ It needs much more calculation and time to find weighting..

PIXEL'S RELATIONSHIPS

Some basic relationships between pixels



Neighbors of a pixel



Relationships Between Pixels: Neighbors of a Pixel

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

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

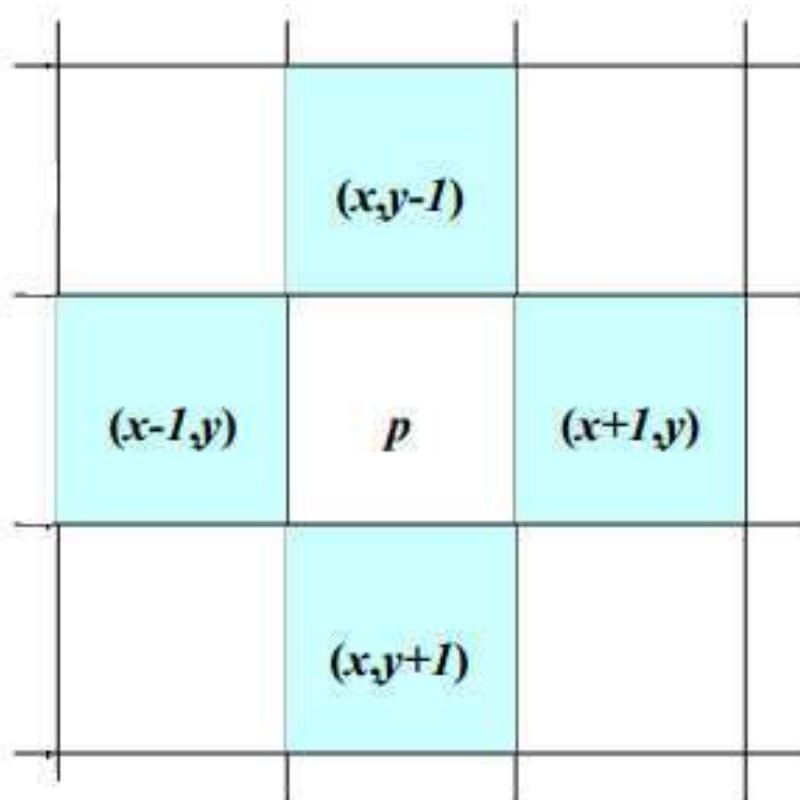
This set of pixels, called the *4-neighbors* of p , is denoted by $N_4(p)$. Each pixel is a unit distance from (x, y) , and some of the neighbors of p lie outside the digital image if (x, y) is on the border of the image.

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 $N_D(p)$. These points, together with the 4-neighbors, are called the *8-neighbors* of p , denoted by $N_8(p)$. As before, some of the points in $N_D(p)$ and $N_8(p)$ fall outside the image if (x, y) is on the border of the image.

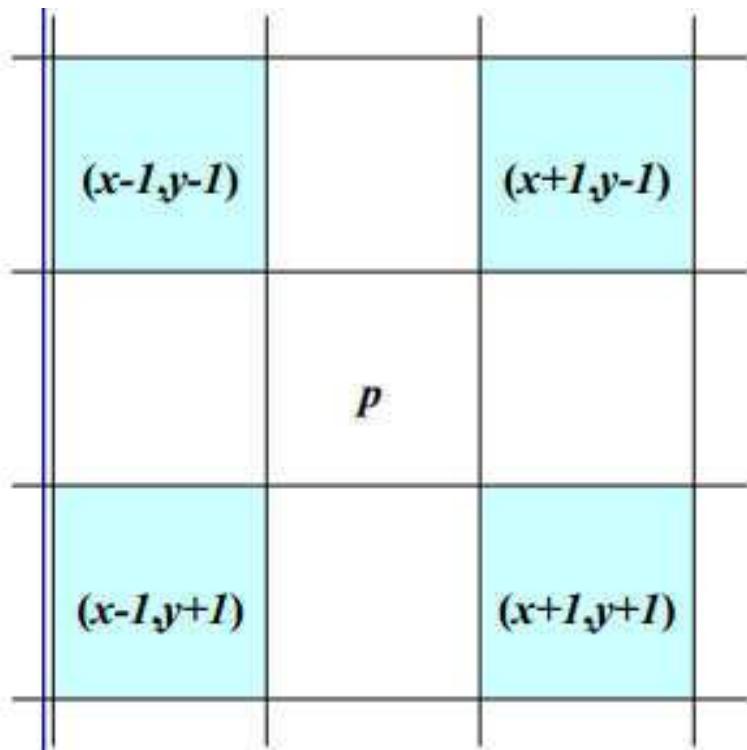
Neighbors of a pixel



4-neighbors of p :

$$N_4(p) = \left\{ (x-1, y), (x+1, y), (x, y-1), (x, y+1) \right\}$$

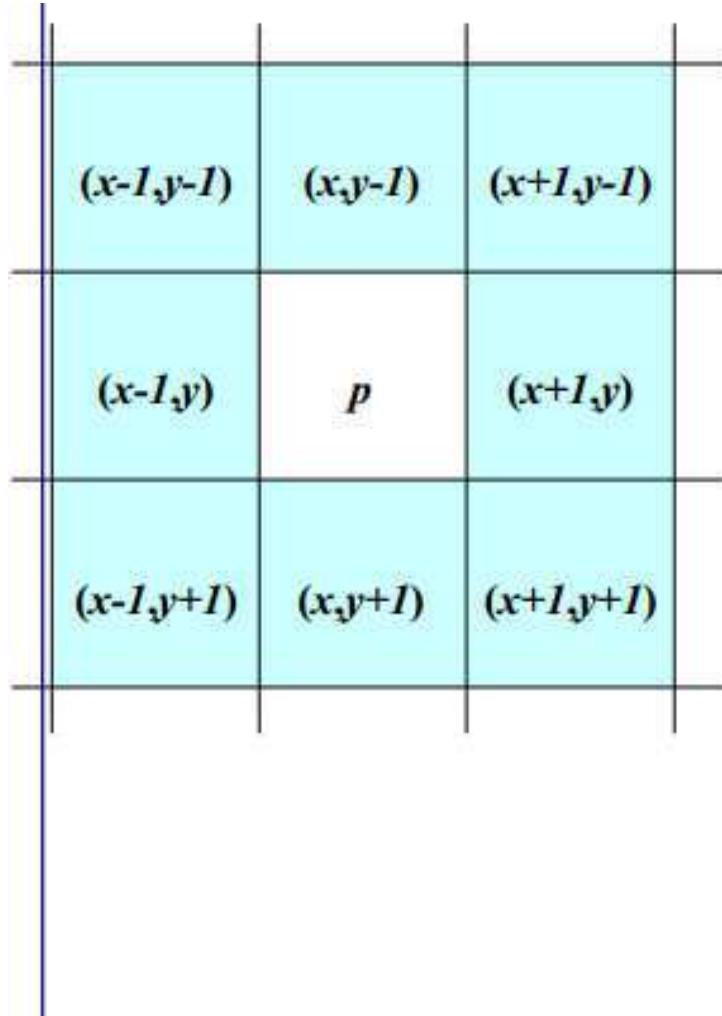
Neighbors of a pixel



Diagonal neighbors of p

$$N_D(p) = \{(x-1,y-1), (x+1,y-1), (x-1,y+1), (x+1,y+1)\}$$

Neighbors of a pixel



8-neighbors of p :

$$N_8(p) = \left\{ \begin{array}{l} (x-1,y-1) \\ (x,y-1) \\ (x+1,y-1) \\ (x-1,y) \\ (x+1,y) \\ (x-1,y+1) \\ (x,y+1) \\ (x+1,y+1) \end{array} \right\}$$

Adjacency

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

Not adjacent adjacent

Adjacent pixels:

They must be neighbors
and have gray value
from the same set V

Adjacency

V : set of gray level values (L), (V is a subset of L .)

3 types of adjacency

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

0 1 1

0 1 0

0 0 1

Arrangement of
pixels in a binary
image

0 1 1

0 1 0

0 0 1

4-adjacent pixels

0 1 1

0 1 0

0 0 1

8 - adjacent pixels

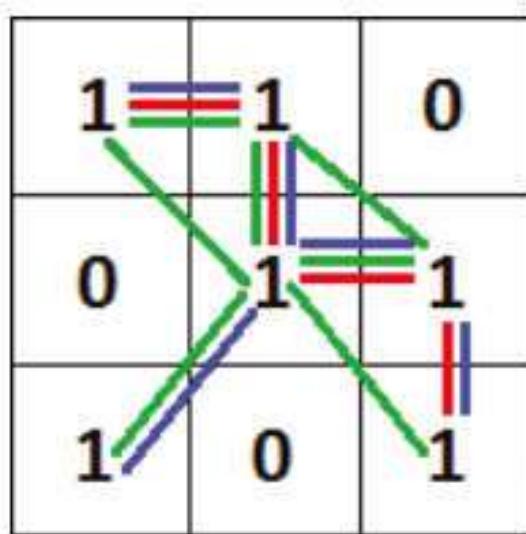
0 1 1

0 1 0

0 0 1

m - adjacent pixels

Adjacency



4-adjacency
8-adjacency
m-adjacency

Example on m-adjacency

Let $V = \{2,3\}$, then:

3	1	3
3	2 (q)	4
(p) 2	0	1

p and q are not m-adjacent

3	1	3 (q)
3	(p) 2	4
2	0	1

p and q are m-adjacent

Digital Path

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

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

where

$$(x_0, y_0) = (x, y), (x_n, y_n) = (s, t), \text{ and pixels } (x_i, y_i) \text{ and } (x_{i-1}, y_{i-1})$$

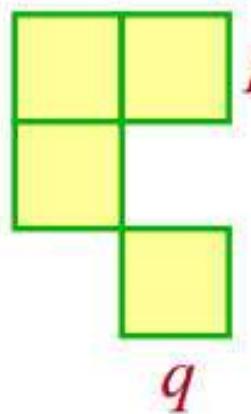
are adjacent for

$$1 \leq i \leq n.$$

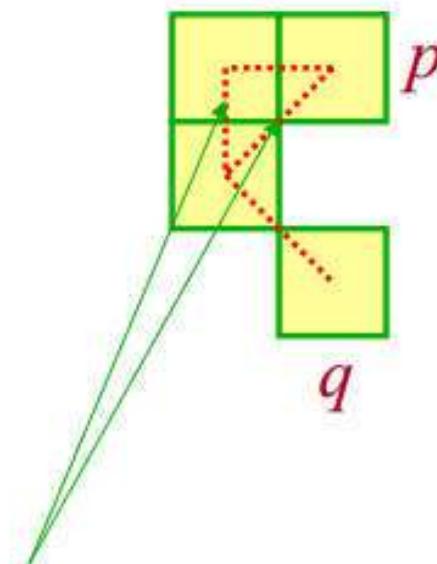
In this case, n is the *length of the path*.

Digital Path

- We can define 4-, 8-, or m-paths depending on the type of adjacency specified.

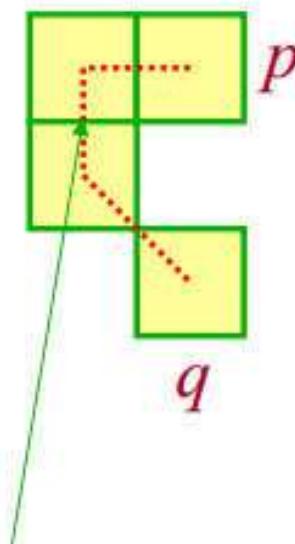


8-path



8-path from p to q
results in some ambiguity

m-path



m-path from p to q
solves this ambiguity

Digital Path

Let $V = \{2,3\}$, then the 4-path doesn't exist between p and q in the following image:

3	1	1	3 (q)
1	3	2	4
3	2	1	4
(p) 2	0	0	1

The diagram shows a 4x4 grid representing a digital path. The grid has columns labeled 3, 1, 1, and 3 (q). Rows are labeled 1, 3, 2, 4. A path is shown from (p) 2 at the bottom-left to 3 (q) at the top-right, consisting of moves (0,1), (1,0), (0,1), (1,0), (0,1), (1,0), (0,1), (1,0). Red lines highlight the path: a vertical line from (p) 2 to the first 3, a horizontal line from the first 3 to the second 3, a vertical line from the second 3 to the first 2, a horizontal line from the first 2 to the second 2, a vertical line from the second 2 to the first 1, a horizontal line from the first 1 to the second 1, a vertical line from the second 1 to the first 0, and a horizontal line from the first 0 to the second 0.

Digital Path

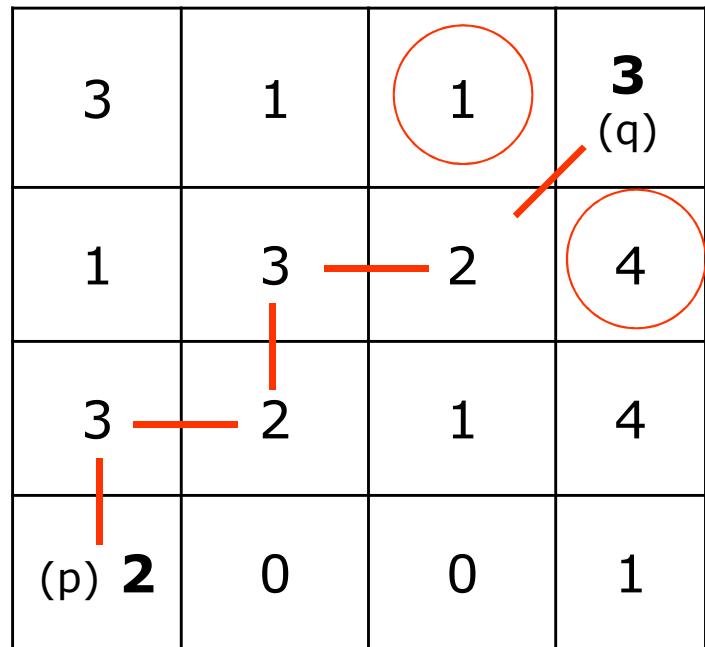
Let $V = \{2,3\}$, then there are many 8-paths between p and q in the following image:

3	1	1	3 (q)
1	3 — 2	4	
3 — 2	1	4	
(p) 2	0	0	1

3	1	1	3 (q)
1	3	2	4
3	2	1	4
(p) 2	0	0	1

Digital Path

Let $V = \{2,3\}$, then there are one m-paths between p and q in the following image:



Digital Path (Example)

According to $V = \{2,3,5\}$, If we want to find p and q pixels 4-, 8- and m-path, (Figure 5)

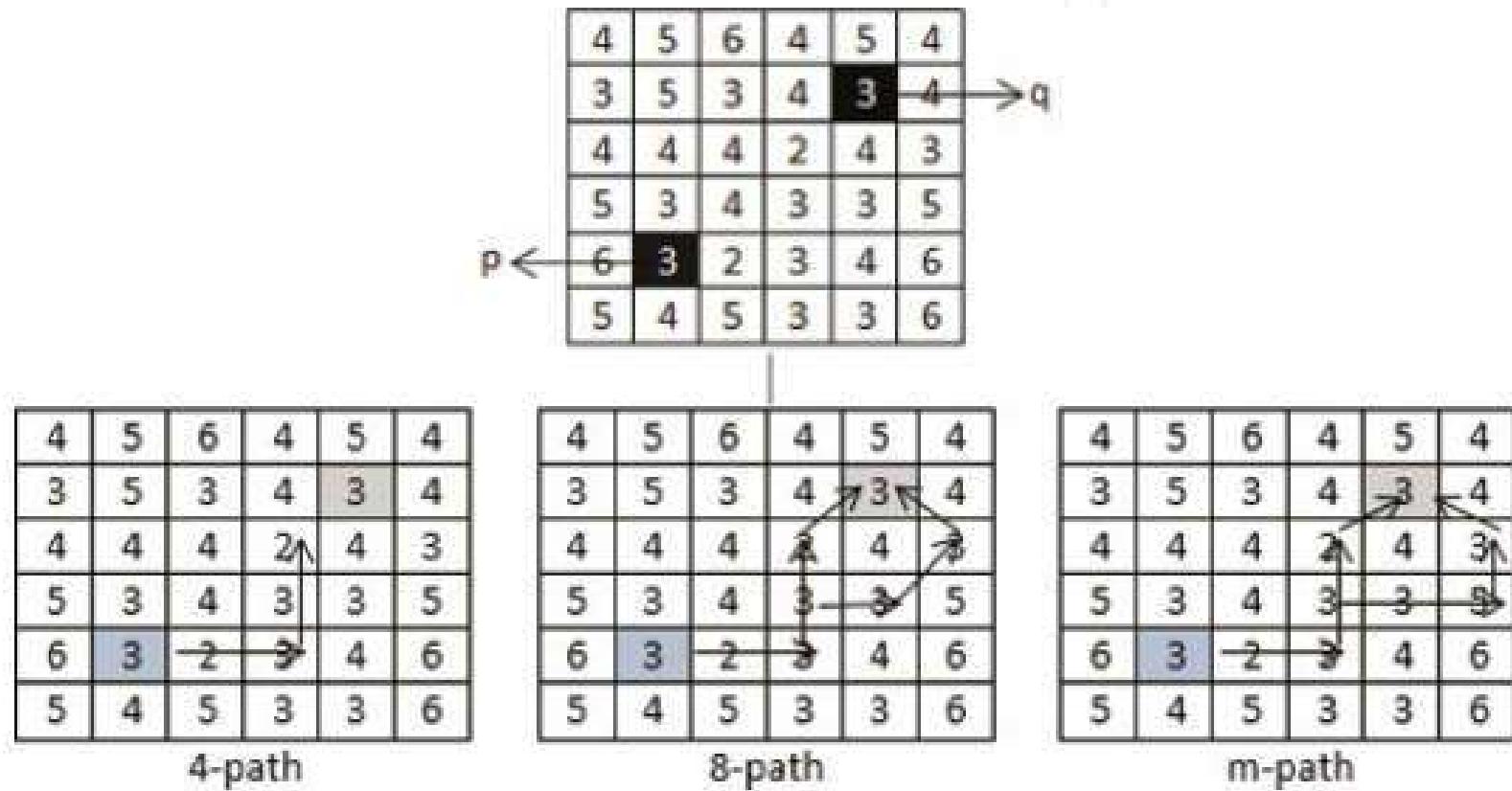


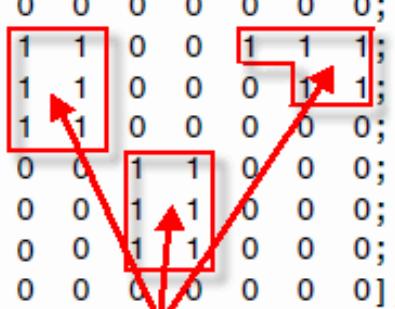
Figure 5. Finding 4-, 8-, m-path between p and q pixels.

Connectivity

- S : a subset of pixels in an image.
- Two pixels p and q are said to be **connected** in S if there exists a path between them consisting entirely of pixels in S .
- For any pixel p in S , the set of pixels that are connected to it in S is called a **connected component** of S .
- If S has only one connected component, it is called a **connected set**.

BW = [0 0 0 0 0 0 0 0;
0 1 1 0 0 1 1 1;
0 1 1 0 0 0 1 1;
0 1 1 0 0 0 0 0;
0 0 0 1 1 0 0 0;
0 0 0 1 1 0 0 0;
0 0 0 1 1 0 0 0;
0 0 0 0 0 0 0 0];

Connected Components



4-adjacency: 3 connected components
8-adjacency: 2 connected components

Connectivity

Let $V = \{1\}$

Based on 4-adjacency

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

3 connected component

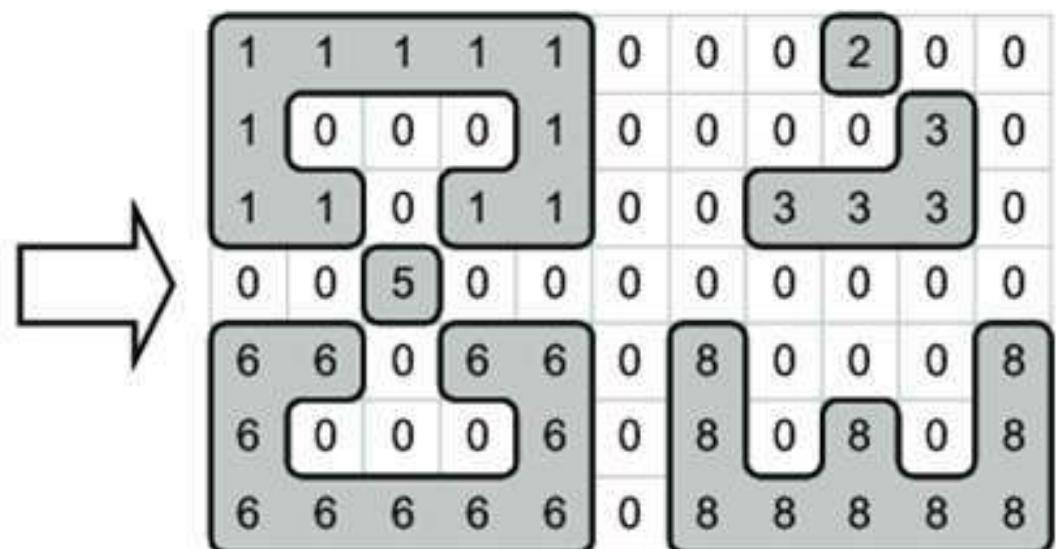
Based on 8-adjacency

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

1 connected component

Example on Connectivity

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

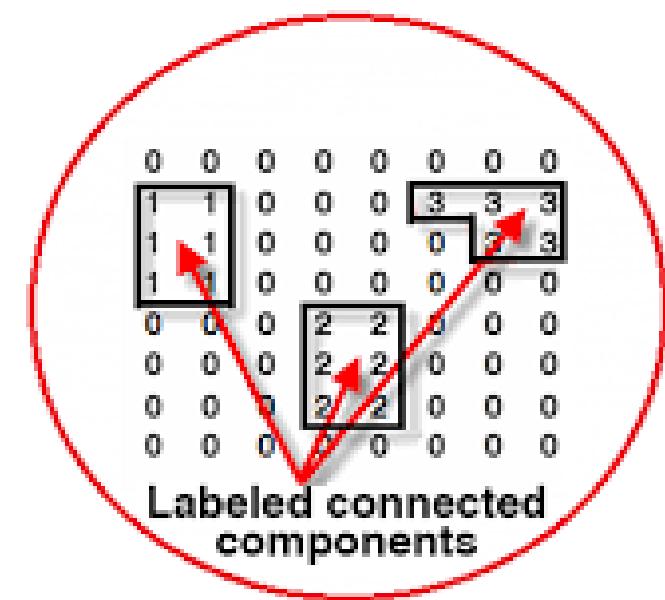
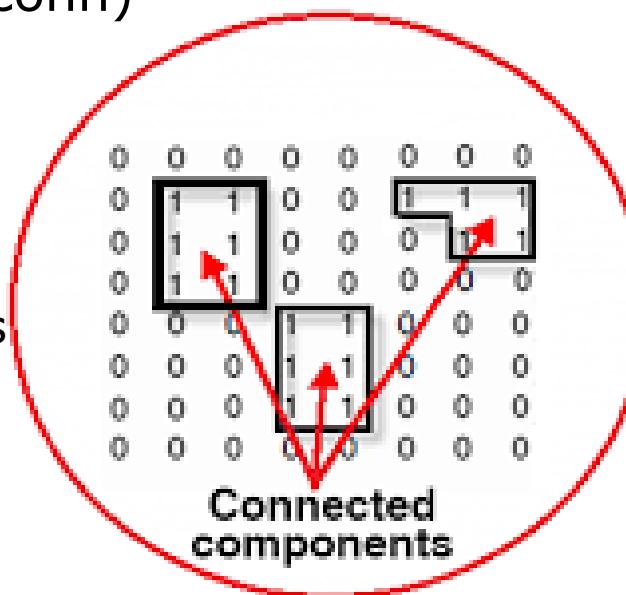


4-adjacency: 6 connected components
8-adjacency: 3 connected components

Example: Connectivity in Matlab

[l,n] = bwlabel(bw, conn)

4-adjacency: 3
connected components



4-adjacency: 4
connected components

A 8x8 binary image matrix with four connected components. The first component (top-left) is yellow and contains 5 white pixels. The second component (center) is blue and contains 4 white pixels. The third component (top-right) is red and contains 3 white pixels. The fourth component (bottom-right) is green and contains 2 white pixels. A red circle highlights the components.

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

(a)

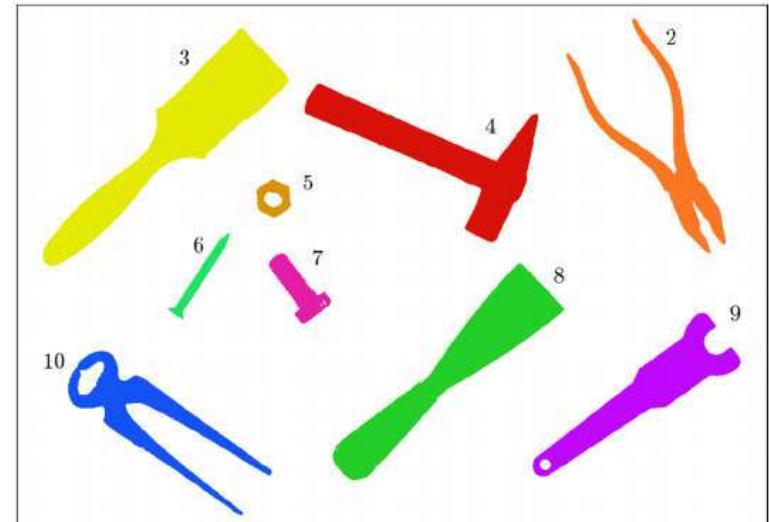
A 8x8 binary image matrix where the connected components from the previous diagram are labeled. Component '1' is yellow, component '2' is blue, component '3' is red, and component '4' is green. A red circle highlights the labeled components.

0	0	0	0	0	0	0	0
1	0	0	0	2	2	2	2
1	1	1	0	0	2	0	0
1	1	0	0	0	0	2	0
0	0	0	3	0	0	2	0
0	0	3	3	0	0	0	0
0	0	0	0	0	0	4	4
0	0	0	0	0	0	4	4

(b)

Regions and Boundaries

- R : a subset of pixels in an image.
- R is a **region** of the image if R is a connected set.
- The **boundary** of a region R is the set of pixels in the region that have one or more neighbors that are not in R .



The circled pixel is NOT a member of boundary if 4-connectivity is used between region and background. It is if 8-connectivity is used.

0	0	0	0	0
0	1	1	0	0
0	1	1	0	0
0	1	(1)	1	0
0	1	1	1	0
0	0	0	0	0

Regions and Boundaries

- R : a subset of pixels in an image.
- R is a **region** of the image if R is a connected set.
- The **boundary** of a region R is the set of pixels in the region that have one or more neighbors that are not in R .

The circled pixel is NOT a member of boundary if 4-connectivity is used between region and background. It is if 8-connectivity is used.

0	0	0	0	0
0	1	1	0	0
0	1	1	0	0
0	1	(1)	1	0
0	1	1	1	0
0	0	0	0	0

Foreground and background

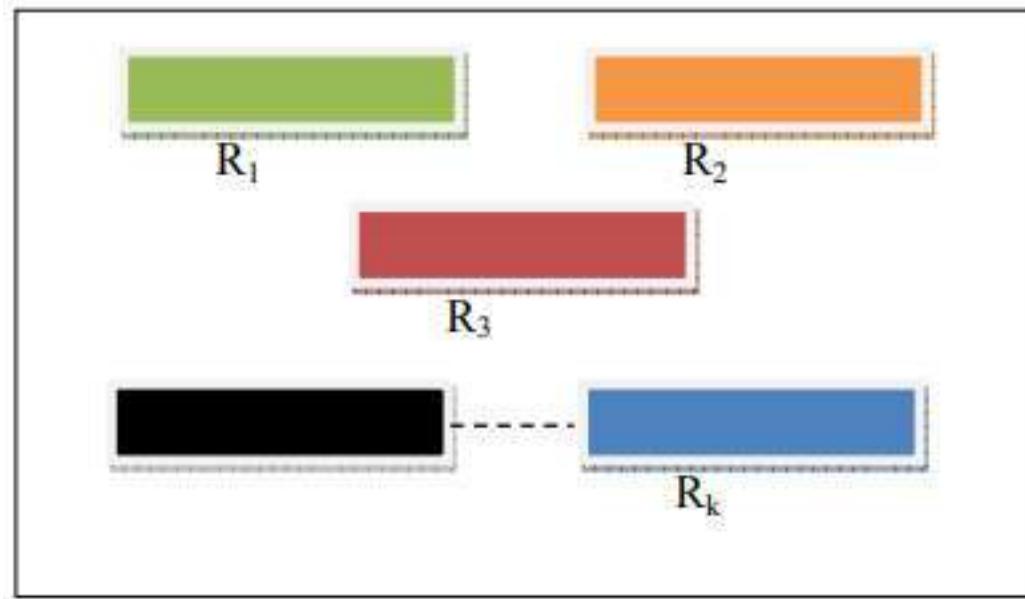
Suppose that the image contains K disjoint regions R_k none of which touches the image border .

R_u : the union of all regions .

$(R_u)^c$: is the complement .

so R_u is called foreground , and $(R_u)^c$: is the background .

Foreground and Background



$R_1 \cup R_2 \cup R_3 \cup \dots \cup R_k = \text{Foreground}$

$(R_1 \cup R_2 \cup R_3 \cup \dots \cup R_k)^c = \text{Background}$

0	1	1
0	1	0
0	0	1

0	1	1
0	1	0
0	0	1

0	1	1
0	1	0
0	0	1

$$\left. \begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \right\} R_i$$

0	0	0	0	0
0	1	1	0	0
0	1	1	0	0
0	1	(1)	1	0
0	1	1	1	0
0	0	0	0	0

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

a	b	c
d	e	f

FIGURE 2.25 (a) An arrangement of pixels. (b) Pixels that are 8-adjacent (adjacency is shown by dashed lines; note the ambiguity). (c) m -adjacency. (d) Two regions that are adjacent if 8-adjacency is used. (e) The circled point is part of the boundary of the 1-valued pixels only if 8-adjacency between the region and background is used. (f) The inner boundary of the 1-valued region does not form a closed path, but its outer boundary does.

DISTANCE MEASURES

Distance measures

If we have 3 pixels: p,q,z:

*p with (x,y)
q with (s,t)
z with (v,w)*

Then:

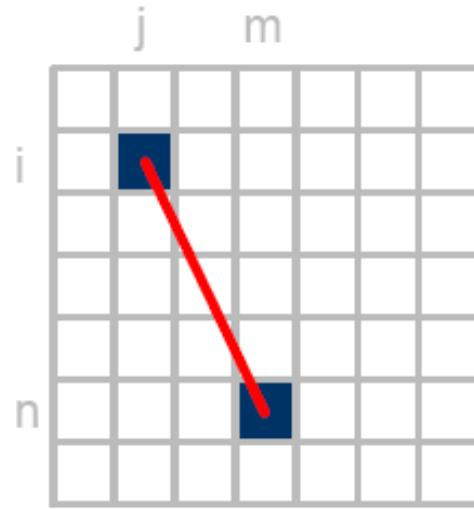
$$D(p,q) = 0 \text{ iff } p = q$$

$$D(p,q) = D(q,p)$$

$$D(p,z) \leq D(p,q) + D(q,z)$$

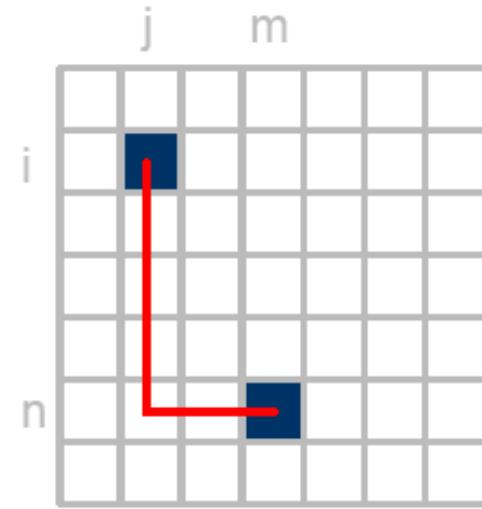
- Euclidean distance between p and q : $D_e(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$
- D_4 distance: $D_4(p,q) = |x-s| + |y-t|$
- D_8 distance: $D_8(p,q) = \max(|x-s|, |y-t|)$
- D_4 and D_8 distances between p and q are independent of any paths that might exist between the points.
- For m -adjacency, D_m distance between two points is defined as the shortest m -path between the points.

Distance measures



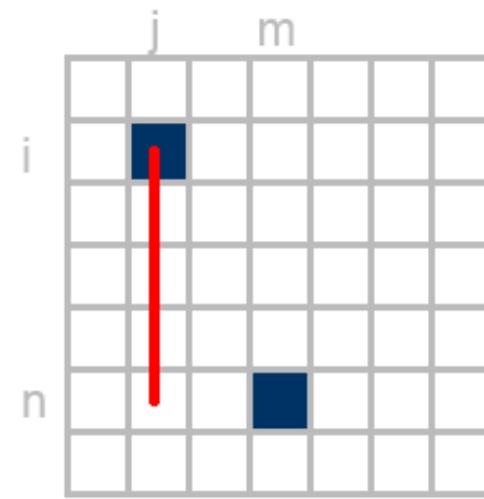
Euclidean Distance

$$= \sqrt{(i-n)^2 + (j-m)^2}$$



City Block Distance

$$= |i-n| + |j-m|$$



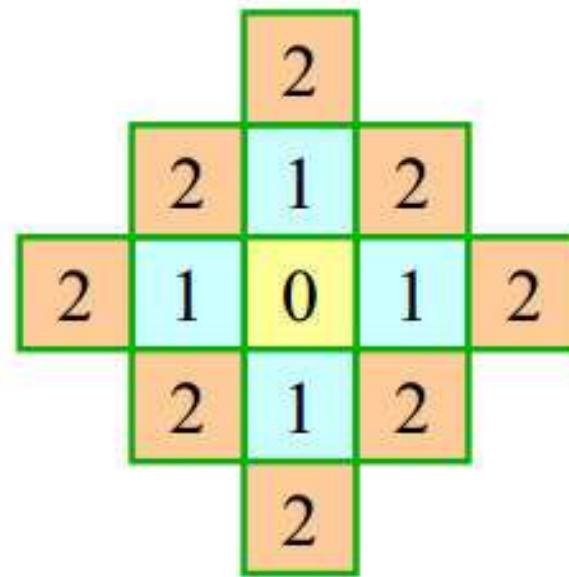
Chessboard Distance

$$= \max[|i-n|, |j-m|]$$

Distance measures

D_4 -distance (*city-block distance*) is defined as

$$D_4(p, q) = |x - s| + |y - t|$$



Pixels with $D_4(p) = 1$ is 4-neighbors of p .

Distance measures

D_8 -distance (*chessboard distance*) is defined as

$$D_8(p, q) = \max(|x - s|, |y - t|)$$

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

Pixels with $D_8(p) = 1$ is 8-neighbors of p .

Example

Compute the distance between the two pixels
using the three distances :

q:(1,1)

P: (2,2)

Euclidian distance : $((1-2)^2 + (1-2)^2)^{1/2} = \sqrt{2}$.

D4(City Block distance): $|1-2| + |1-2| = 2$

D8(chessboard distance) : $\max(|1-2|, |1-2|) = 1$

(because it is one of the 8-neighbors)

	1	2	3
1	q		
2		p	
3			

Distance Measures

Example :

Use the city block distance to prove 4-neighbors ?

Pixel A : $|2-2| + |1-2| = 1$

Pixel B: $|3-2|+|2-2|= 1$

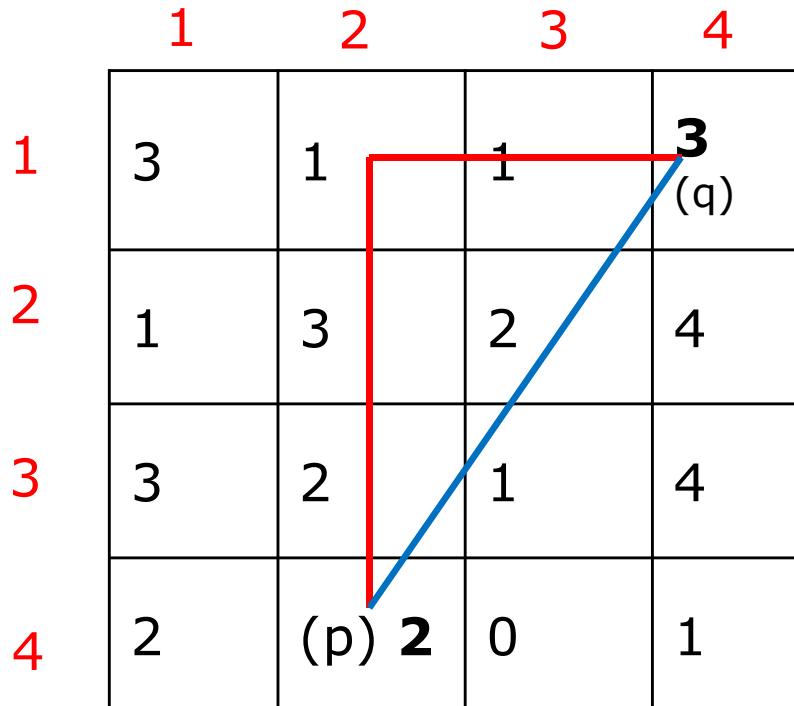
Pixel C: $|2-2|+|2-3| =1$

Pixel D: $|1-2| + |2-2| = 1$

	1	2	3
1		d	
2	a	p	c
3		b	

Now as a homework try the chessboard distance to proof the 8-neighbors!!!!

Example: Distance Measures

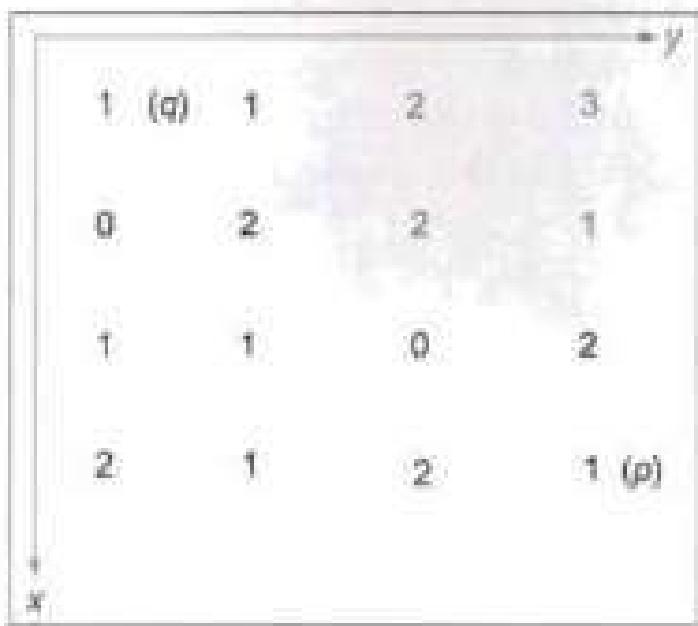


$$D_4 = |1-4| + |2-4|$$
$$D_4 = 3 + 2 = 5$$

$$D_8 = \max(|1-4|, |2-4|)$$
$$D_8 = \max(3, 2) = 3$$

$$\text{Euc} = ((1-4)^2 + (2-4)^2)^{1/2}$$
$$\text{Euc} = (3^2 + 2^2)^{1/2}$$
$$\text{Euc} = (9 + 4)^{1/2}$$
$$= 3.6$$

Example: Distance Measures



Computation of Euclidian distance:

The coordinates of *q* are (0, 0)

The coordinates of *p* are (3, 3)

$$\begin{aligned} D_e &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\ &= \sqrt{(0 - 3)^2 + (0 - 3)^2} \\ &= \sqrt{3^2 + 3^2} \\ &= \sqrt{18} \\ &= 3\sqrt{2} \end{aligned}$$

Computation of D_1 distance:

$$\begin{aligned} D_1 &= |0 - 3| + |0 - 3| \\ &= 3 + 3 \\ &= 6 \end{aligned}$$

Computation of D_∞ distance:

$$\begin{aligned} D_\infty &= \max(|0 - 3|, |0 - 3|) \\ &= 3 \end{aligned}$$

Image Processing



Ch3: Intensity Transformation
and spatial filters

Image Enhancement?

☞ **Enhancement :** is to process an image so that the result is more suitable than the original image for a *specific* application.

Enhancing an image provides better contrast and a more detailed image as compare to non enhanced image. It is used to enhance medical images, images captured in remote sensing, images from satellite e.t.c

Enhancement techniques fall into 2 types:

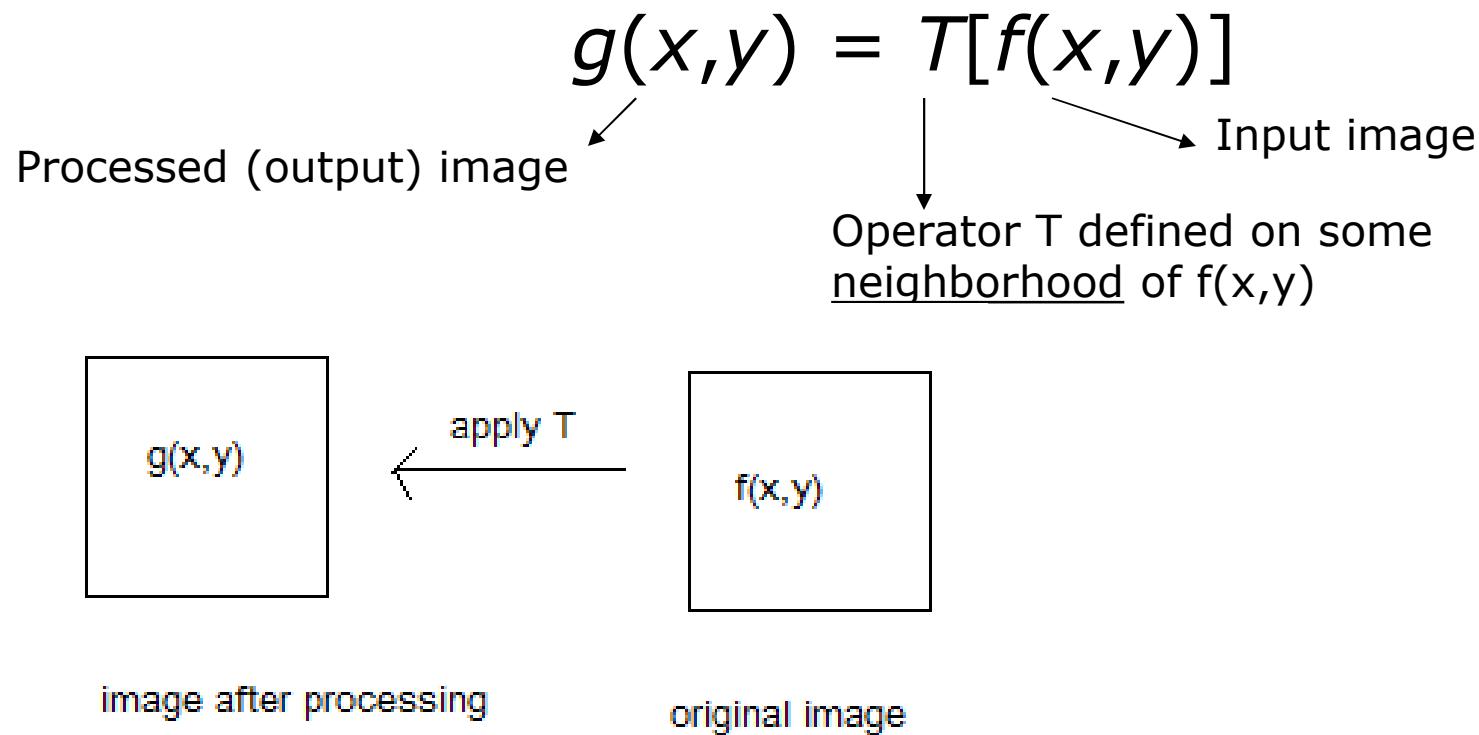
- **Spatial domain:** direct manipulation of pixels in the image plane
- **Frequency domain:** modifying frequency spectrum of the image.

☞ *In this chapter, we are going to discuss spatial domain techniques*

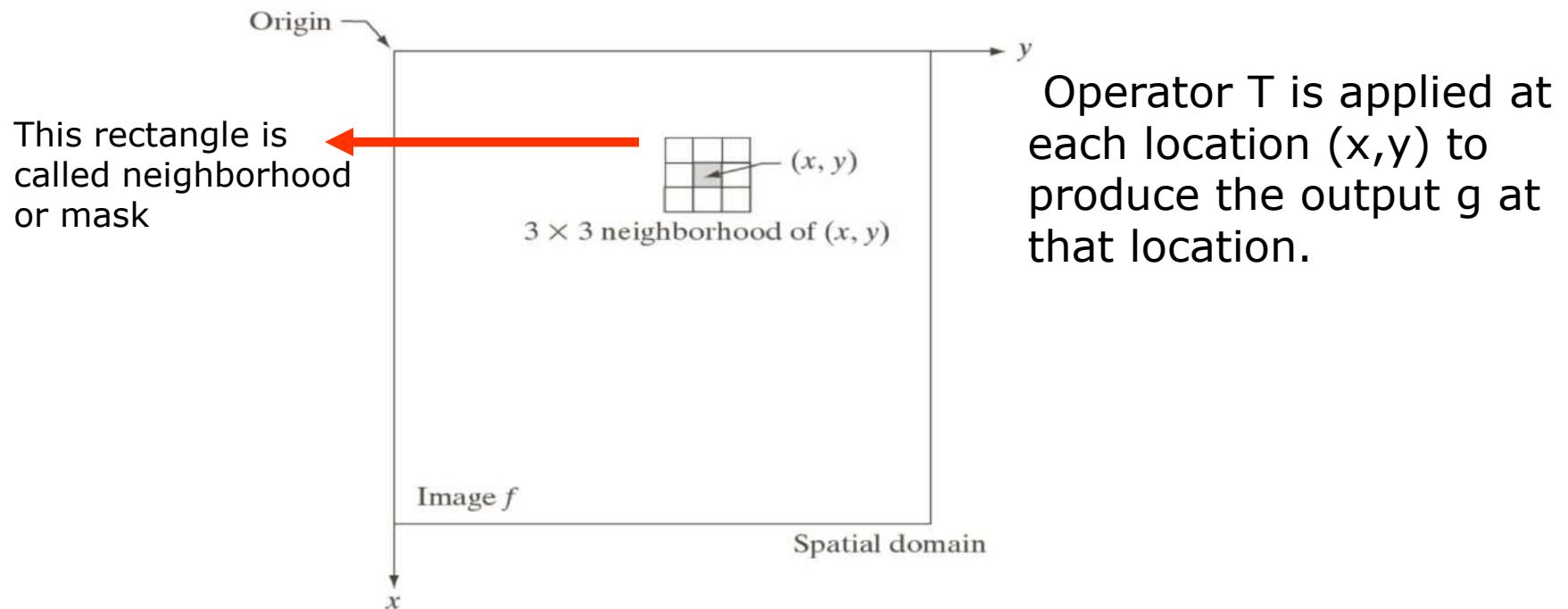
Enhancement in Spatial Domain

- Spatial domain: aggregate of pixels composing an image

Spatial domain processes:



Defining a Neighborhood (T)



types of neighborhood:

1. **intensity transformation**: neighborhood of size 1×1
2. **spatial filter** (or mask ,kernel, template or window): neighborhood of larger size , like in the above example.

Intensity Transformations Functions

- Operation deals with pixel intensity values individually
- The transformed output pixel value does not depend on any of the neighboring pixel value of the input image
- The intensity values are altered using particular transformation techniques
- Here, T is called intensity transformation function or (mapping, gray level function)

$$g(x,y) = T[\underbrace{f(x,y)}_r]$$

s **r**

$$s = T(r)$$

where **r** is the pixels of the input image and **s** is the pixels of the output image

Spatial Filtering

- Mask is a small matrix useful for blurring, sharpening, edge-detection and more.
- New image is generated by applying the mask matrix on the input image.
- The output pixel values thus depend on the neighboring input pixel values.
- The mask may be of any dimension 3X3 4X4

Spatial Filtering

120	190	140	150	200
17	21	30	8	27
89	123	150	73	56
10	178	140	150	18
190	14	76	69	87

x

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

=

	98			

Intensity transformations functions

Intensity transformation functions fall into 2 approaches:

1) Basic intensity transformations

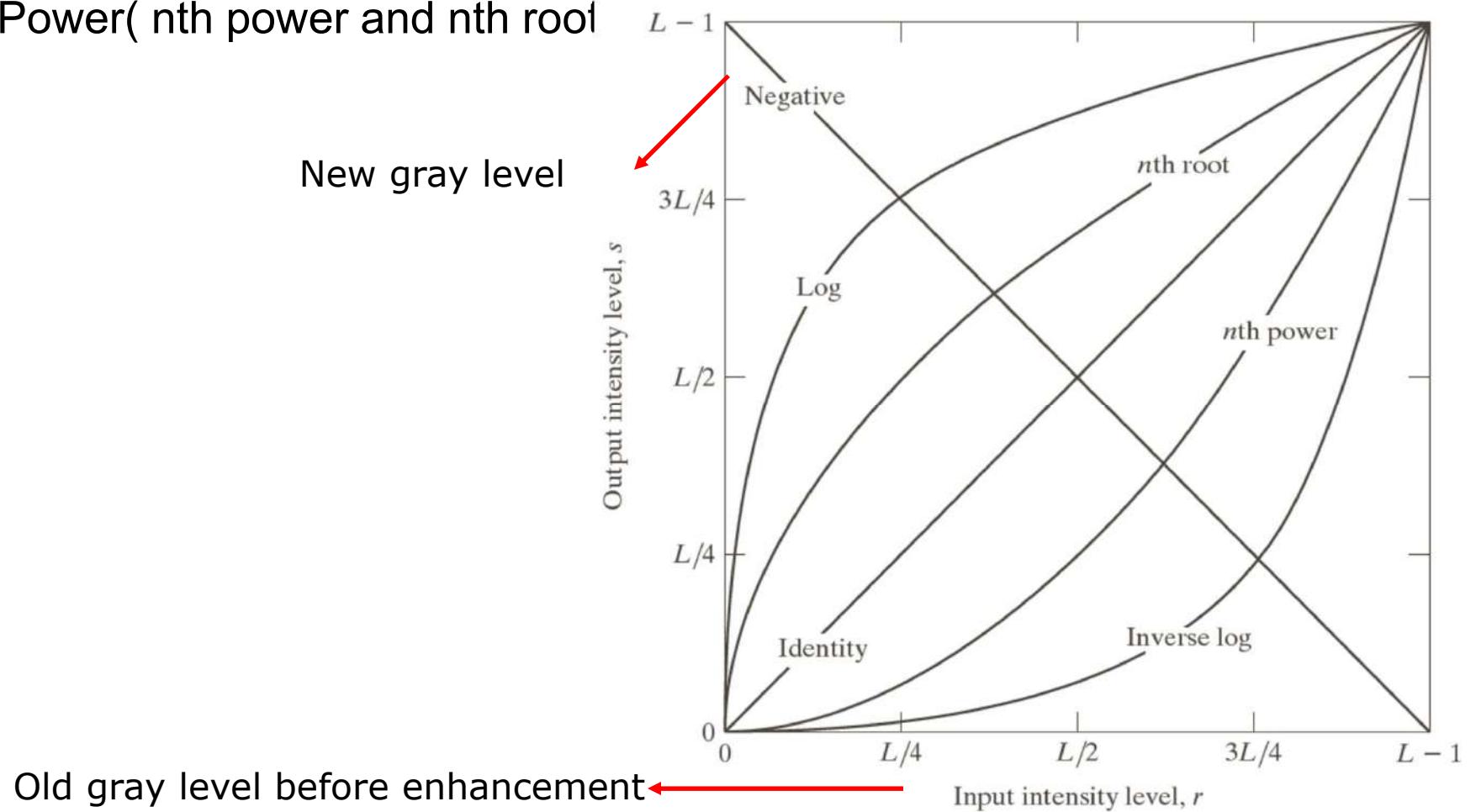
- a) Linear (negative and identity).
- b) logarithmic (Log and Inverse Log) .
- c) Power(nth power and nth root).

2) piecewise Linear transformation functions.

- a) Contrast stretching, thresholding
- b) Gray-level slicing
- c) Bit-plane slicing

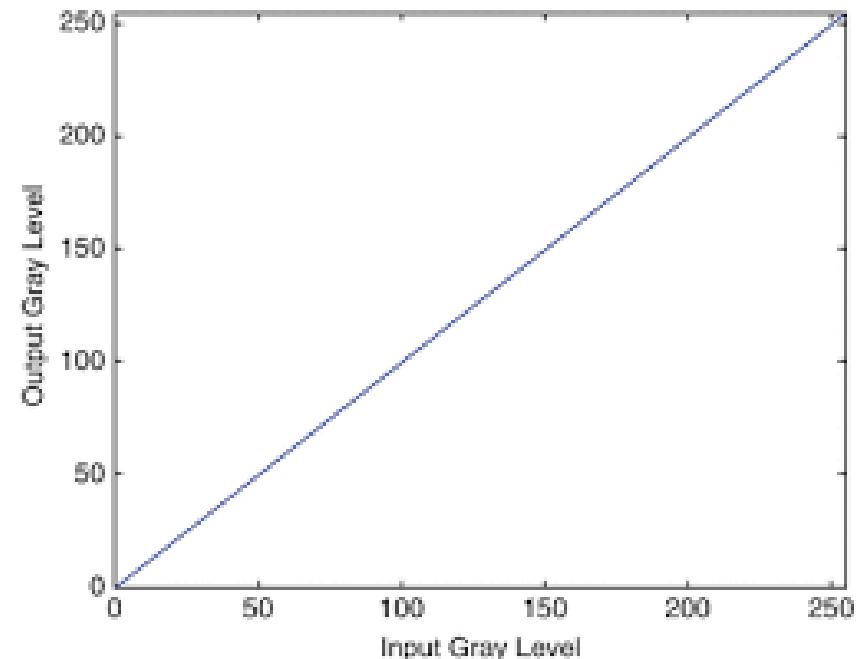
Basic Intensity (gray level) Transformations

- a) Linear (negative and identity).
- b) logarithmic (Log and Inverse Log) .
- c) Power(nth power and nth root)



Identity Transformation Function

- In **Identity transformation**, each value of the input image is directly mapped to each other value of output image.
- Thus, the output intensities are identical to the input intensities. It has been shown below:



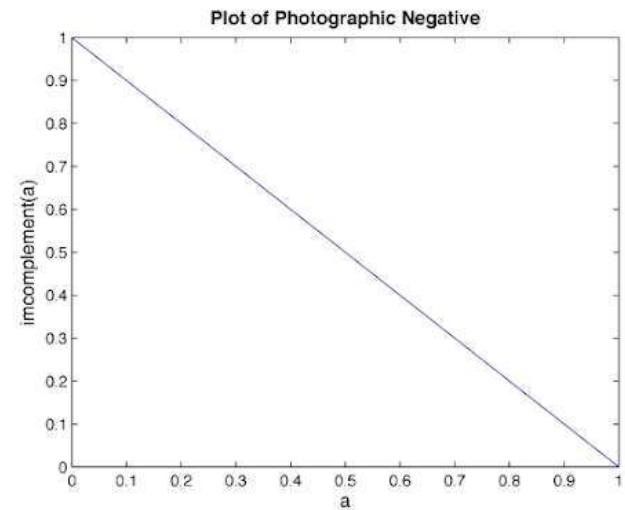
Negative Transformation Function

- ❑ Negative images are useful for enhancing white or grey detail embedded in dark regions of an image.
- ❑ The negative of an image with gray levels in the range [0,L-1] is obtained by using the expression

$$s = L - 1 - r$$

(L-1) = Maximum pixel value where L is the number of gray levels in the input image

r = input Pixel value of an original image



Negative Transformation Function

The negative of an image with intensity levels in the range [0,L-1] is obtained by using the negative transformation :

$$s = L - 1 - r$$

Image (r)



Image (s) after applying T (negative)



Advantages of negative :

- ✓ Produces an equivalent of a photographic negative.
- ✓ Enhances white or gray detail embedded in dark regions especially when the black areas are dominant in size..

Negative Transformation Function

The negative of an image with intensity levels in the range [0,L-1] is obtained by using the negative transformation :

$$s = L-1-r$$

Example

the following matrix represents the pixels values of an 8-bit image (r) , apply negative transform and find the resulting image pixel values.

solution:

$$L = 2^8 = 256$$

$$s = L-1-r$$

$$s = 255-r$$

Apply this transform to each pixel to find the negative

Image (r)			
100	110	90	95
98	140	145	135
89	90	88	85
102	105	99	115

Image (s)

155	145	165	160
157	115	110	120
166	165	167	170
153	150	156	140 s

Negative Transformation Function

Exercise:

the following matrix represents the pixels values of a 5-bit image (r) , apply negative transform and find the resulting image pixel values.

solution:

Image (s)

Image (r)			
21	26	29	30
19	21	20	30
16	16	26	31
19	18	27	23

Image Negation (Example)



a b

FIGURE 3.4
(a) Original
digital
mammogram.
(b) Negative
image obtained
using the negative
transformation in
Eq. (3.2-1).
(Courtesy of G.E.
Medical Systems.)

Shows a small lesion.

$$s = L - 1 - r$$

Produces an equivalent of a photographic negative.
Enhances white or gray detail embedded in dark regions.

Log Transformation Function

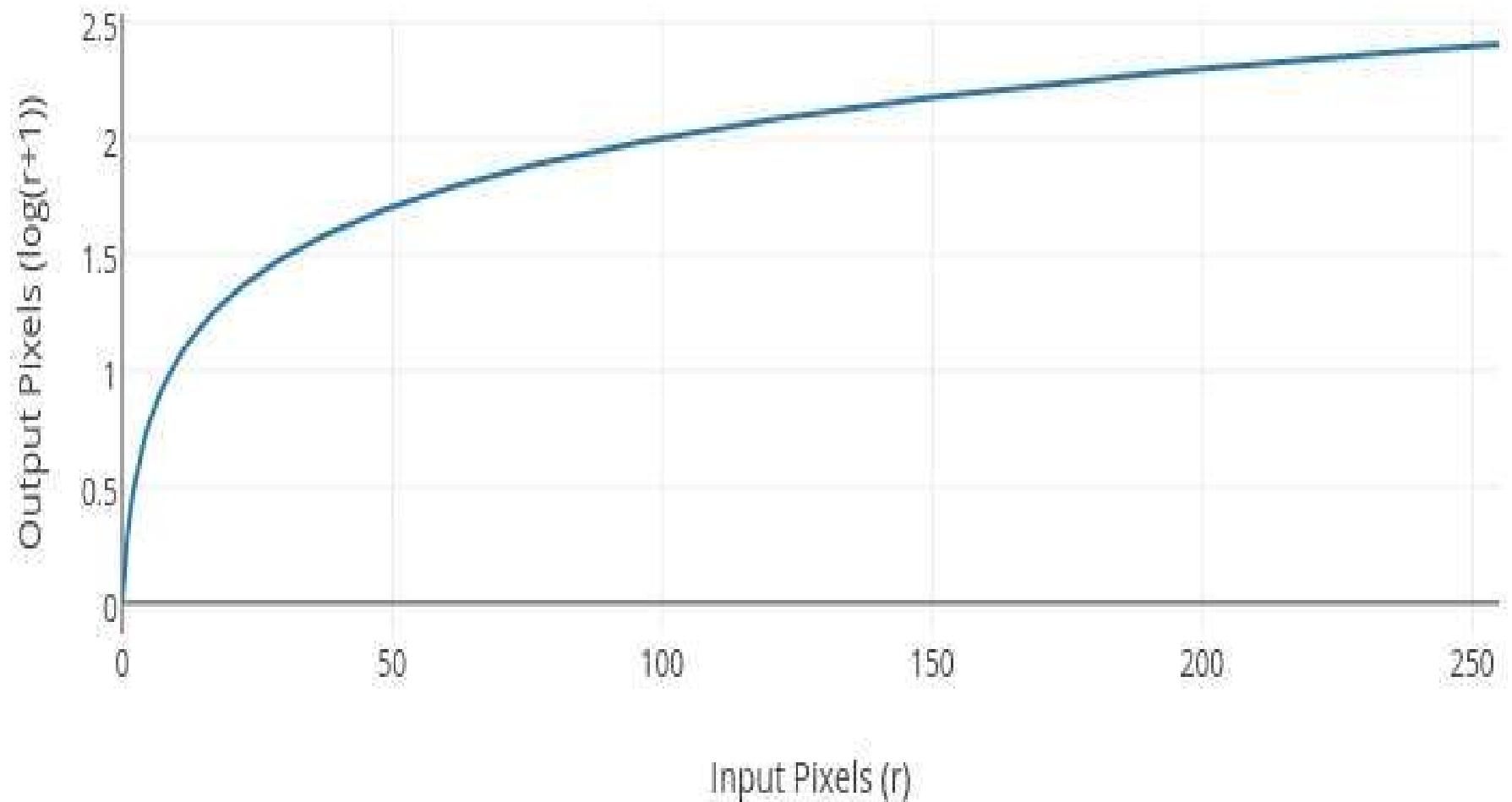
- The general form is:

$$s=c \log(1+r) \quad r>0$$

- **(1+r)**: All the input intensity values have been incremented by 1 because the input values vary from 0 to 255 and the $\log(0)$ is **not defined**.
- **r** varies from (0-255), $\log(1+r)$ ranges from (0-2.4).
- The role of the multiplier **C** is to make the log-transformed pixel values span the entire range of 256 grayscale levels available for the output image.

$$255 = c \log(r_{max} + 1) \quad c = 255 / \log(r_{max} + 1)$$

Log Transformation Function



Log Transformation Function

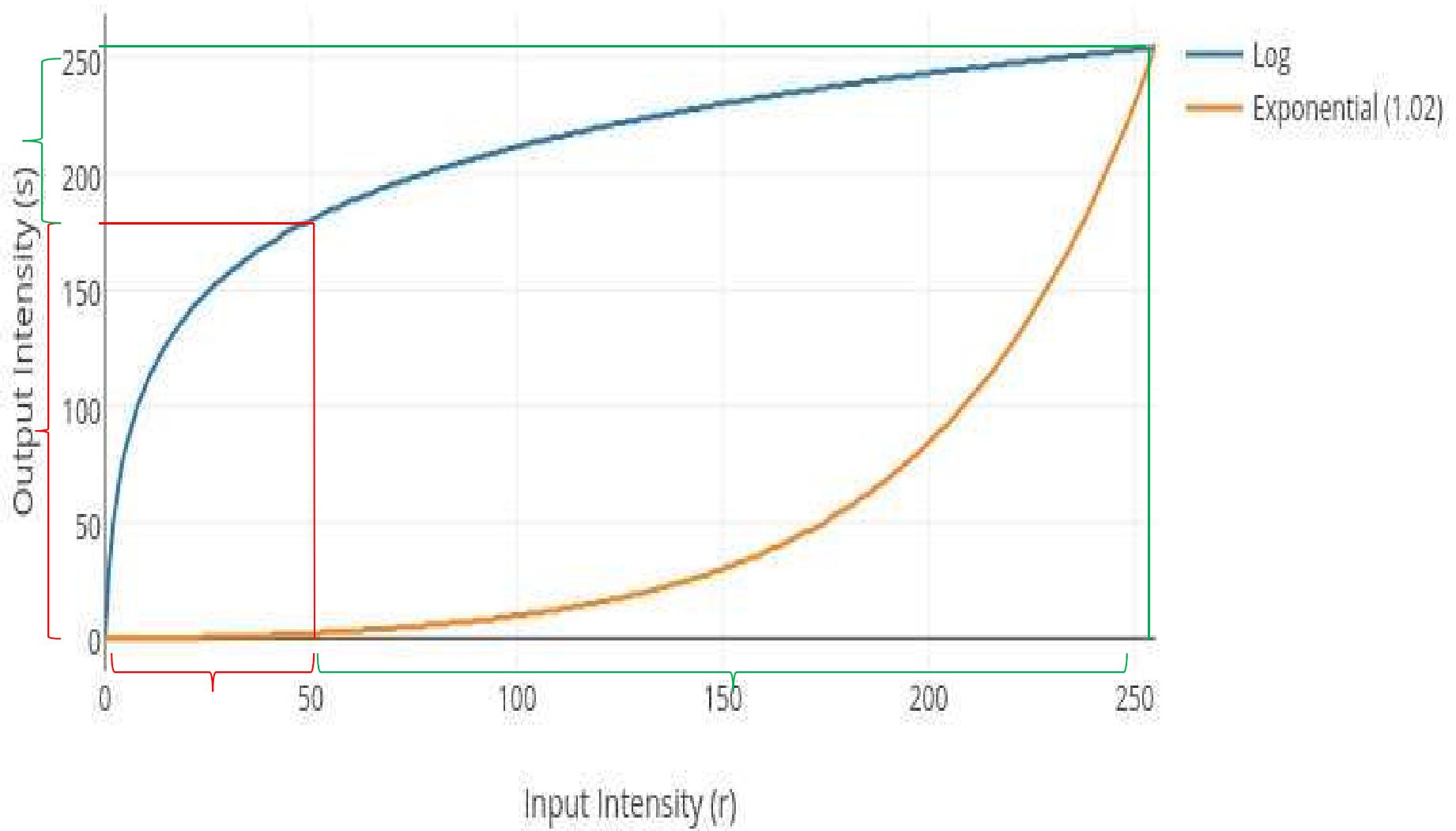
- *The general form is:*

$$S=c \log(1+r) \quad r \geq 0$$

where c is a constant, and it is assumed that

- It maps a narrow range of low gray-level values in the input image into a wider range of output levels.
- The opposite is true of higher values of input levels.
- Used to expand the values of dark pixels in an image while compressing the higher-level values.
- The opposite is true of the inverse log.
- The power-law transformations are much more versatile for this purpose than the log transformation

Log Transformation Function



Log Transformation Function

Input range



Output range after applying log transformation function



Intensity values of 0 and 15 in the input are mapped to 0 and 127 in the output.

The exact opposite phenomenon takes place at the other end of the spectrum. For example, pixels with intensities of 205 and 255 are mapped to 245 and 255 by the log transform.

Exponential or Inverse-Log Transformation

- The general form is:

$$s = T(r) = c(b^r - 1)$$

- Typically, b is chosen to lie close to 1.

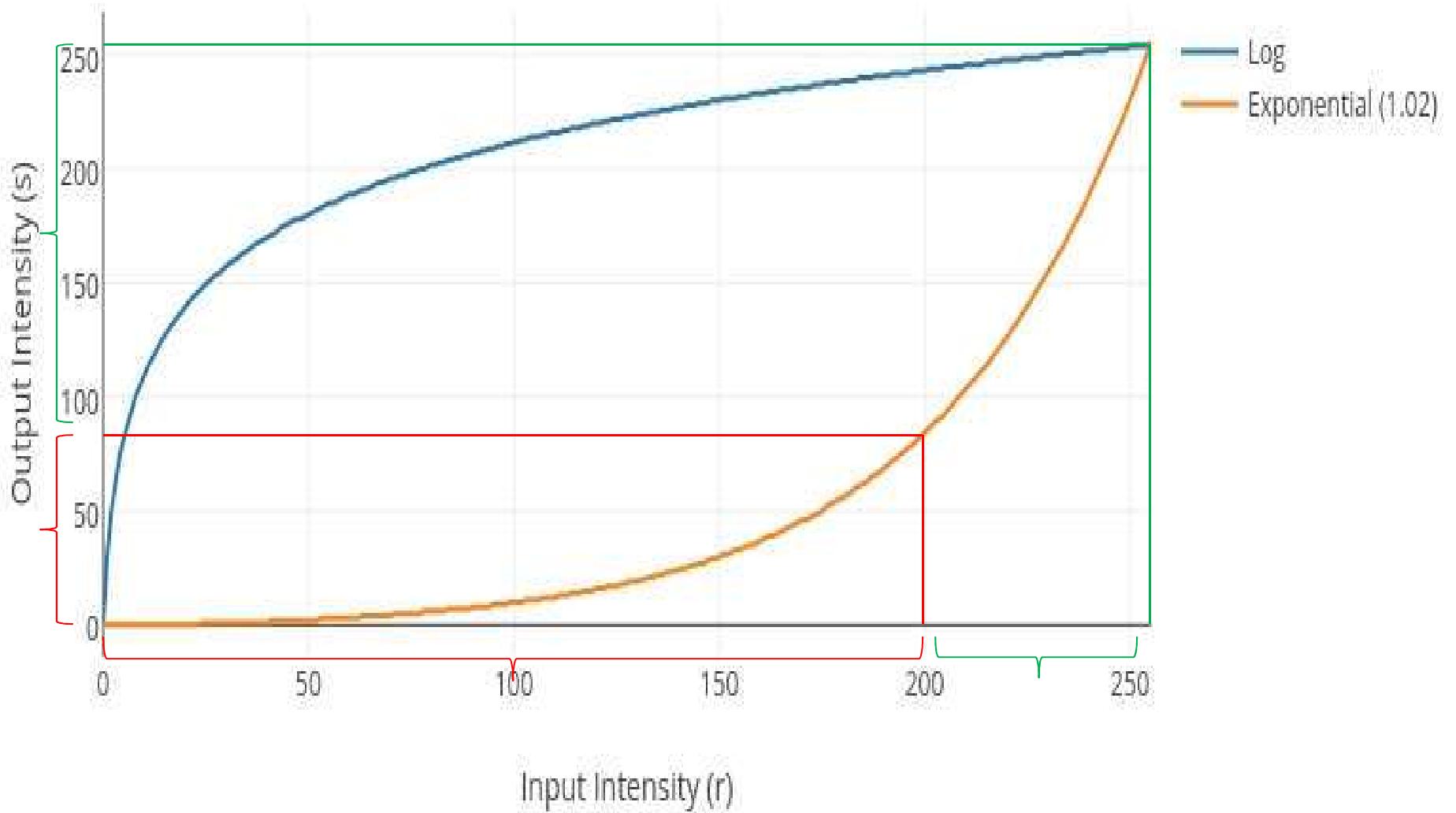
Input range



Output range after applying inverse log transformation function



Inverse Log Transformation Function



Example: Log Trans. Function



This example proves that the log transform can be effective in editing pictures that have been captured *against the light source* by digging out contrast information from the darker regions of an image at the cost of the brighter segments

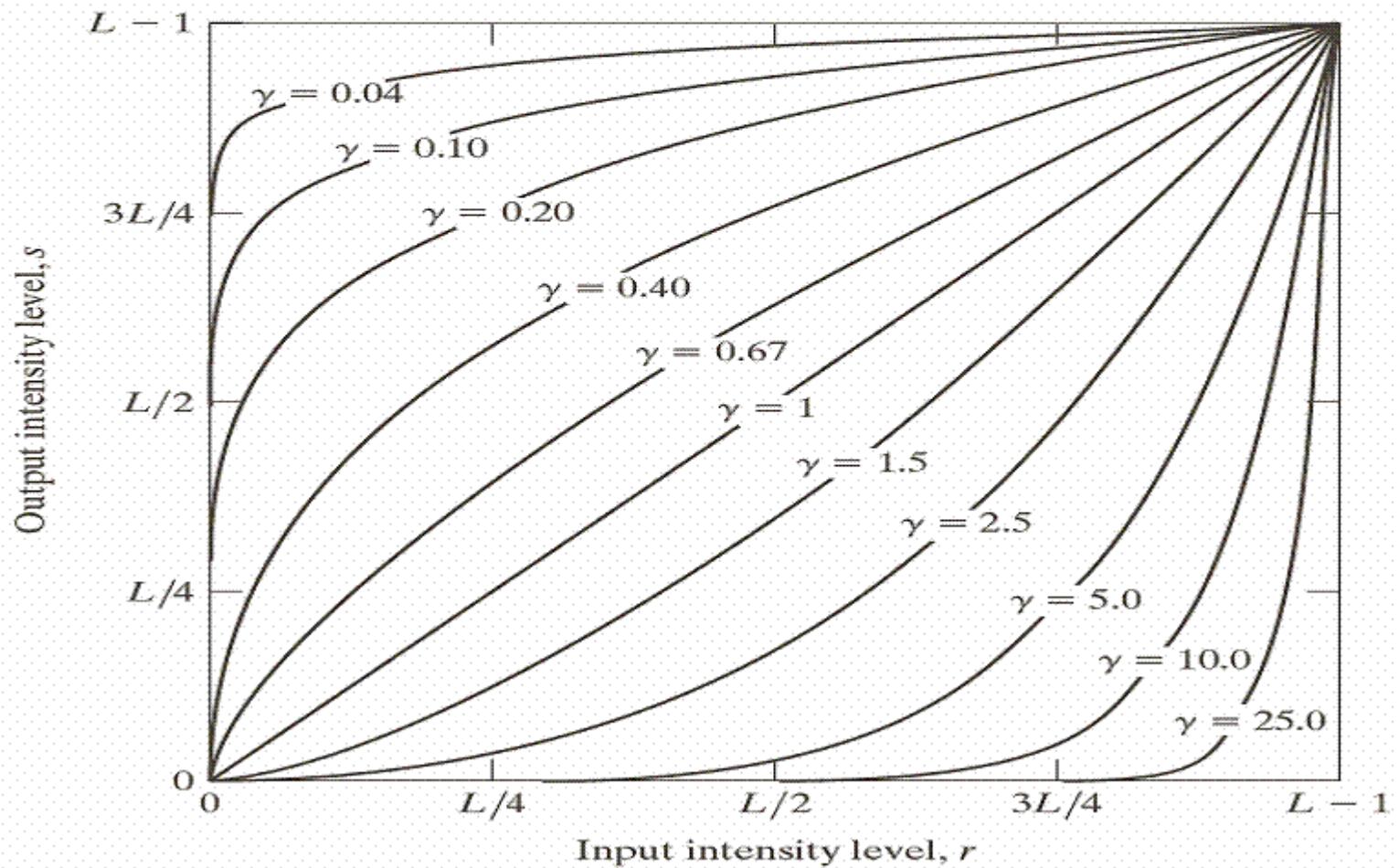
Power-Law Transformation

- Power-law transformations have the basic **form**

$$S=cr^y$$

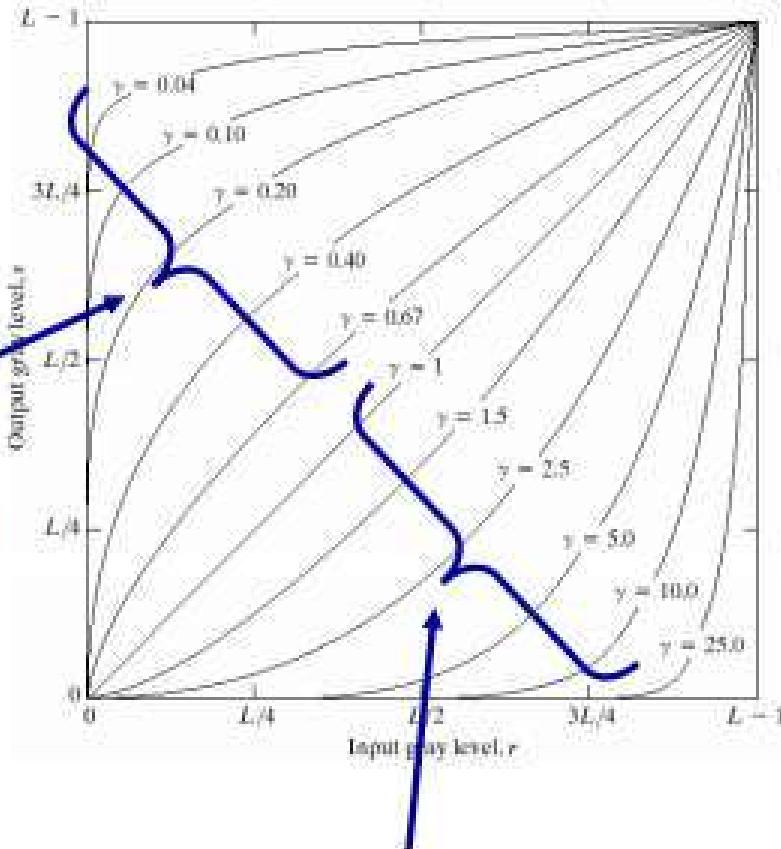
- Where c and gamma are positive constants.
- Plots of s versus r for various values of gamma are shown in **Fig. 3.6**.
- As in the case of the log transformation, power-law curves with fractional values of gamma map a **narrow range** of dark input values into a **wider range** of output values, with the opposite being true for higher values of input levels.

FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases). All curves were scaled to fit in the range shown.



Power-law transformation

Map a narrow range of dark input values into a wider range of output values.



Map a wide range of light input values into a narrower range of output values.

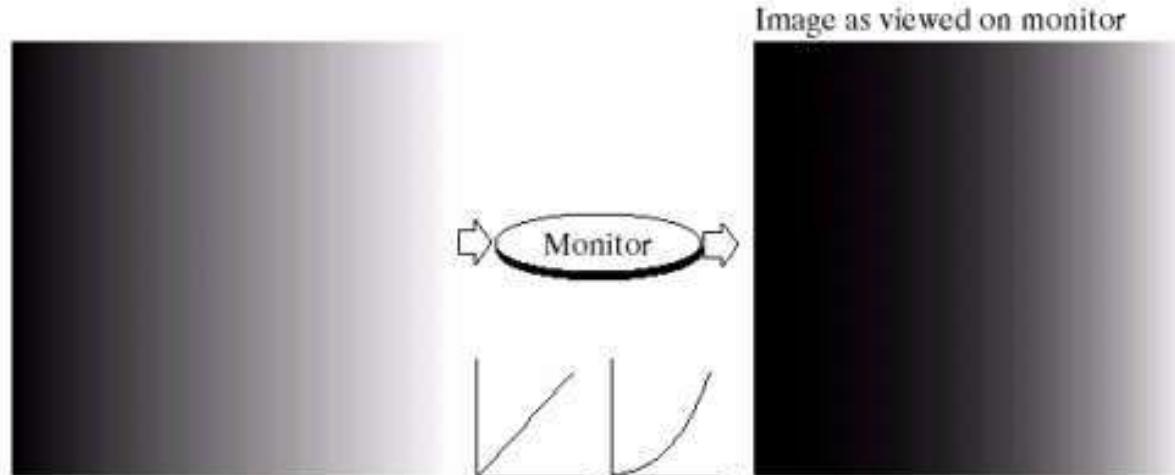
FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases).

Power-Law Transformation

a b
c d

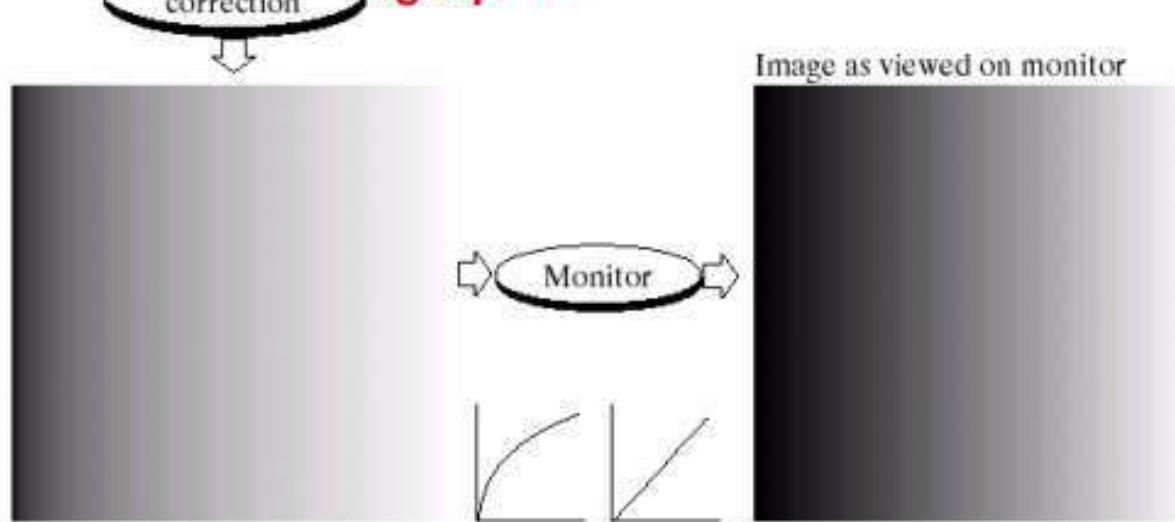
FIGURE 3.7

- (a) Linear-wedge gray-scale image.
- (b) Response of monitor to linear wedge.
- (c) Gamma-corrected wedge.
- (d) Output of monitor.



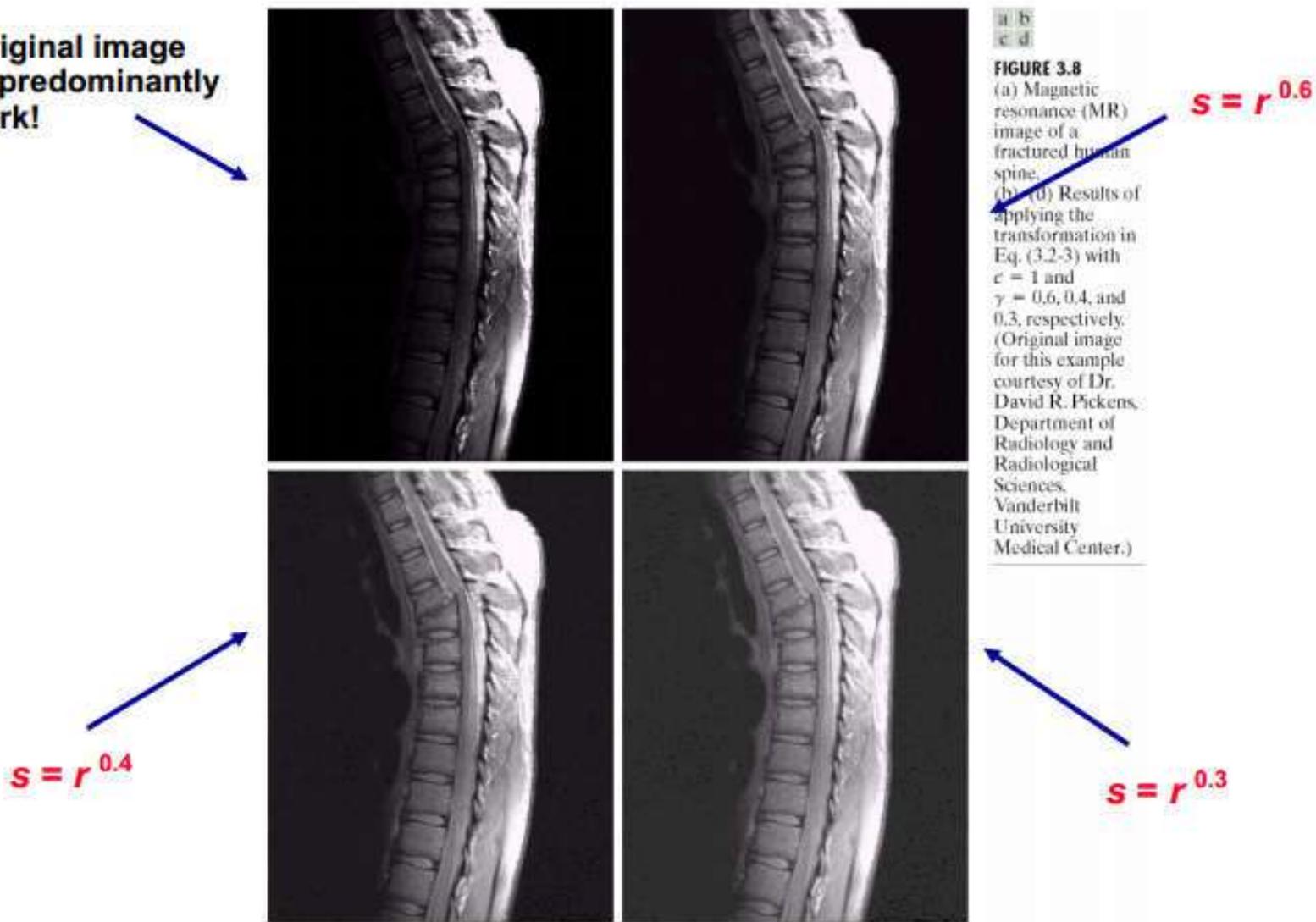
A variety of devices for image capture, printing and display respond according to power law.

Such systems tend to produce images that are darker than intended.



Example: Power-Law Transformation

Original image
is predominantly
dark!



a b
c d

FIGURE 3.8
(a) Magnetic resonance (MR) image of a fractured human spine.
(b)-(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3 , respectively.
(Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

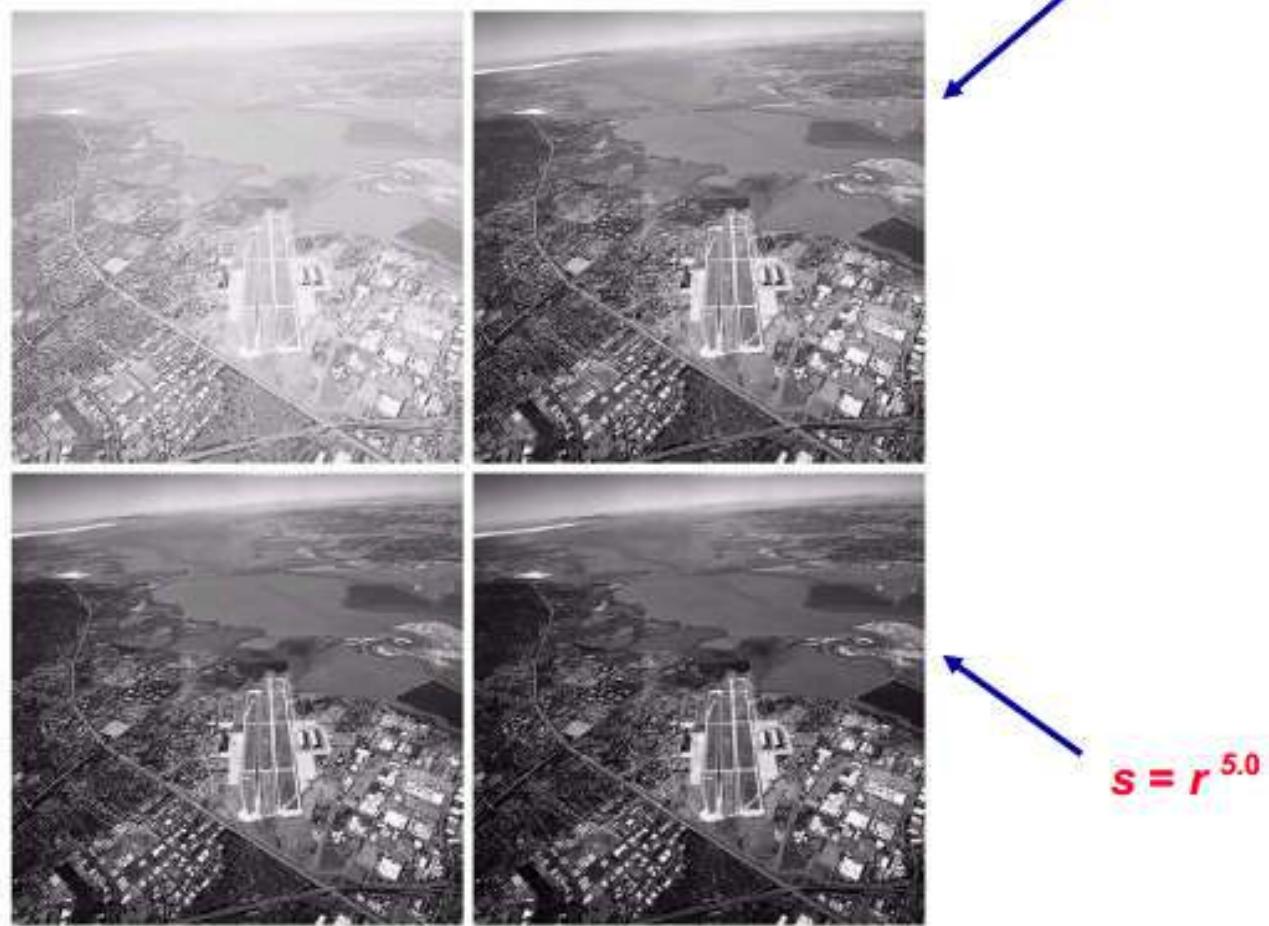
Example: Power-Law Transformation

Original image
has a washed-out
appearance!

a b
c d

FIGURE 3.9

(a) Aerial image.
(b)–(d) Results of
applying the
transformation in
Eq. (3.2-3) with
 $c = 1$ and
 $\gamma = 3.0, 4.0,$ and
 5.0 , respectively.
(Original image
for this example
courtesy of
NASA.)



Piecewise Linear Transformation Functions



Contrast Stretching and
Thresholding

Contrast Stretching

Contrast can be simply explained as the difference between maximum and minimum pixel intensity in an image.

The following example shows three 8-bit images with different contrast level

100	100	100	100
100	100	100	100
100	100	100	100
100	100	100	100

Contrast level
 $100 - 100 = 0$
(No contrast)

0	10	20	30
40	60	80	90
100	120	150	180
200	220	240	255

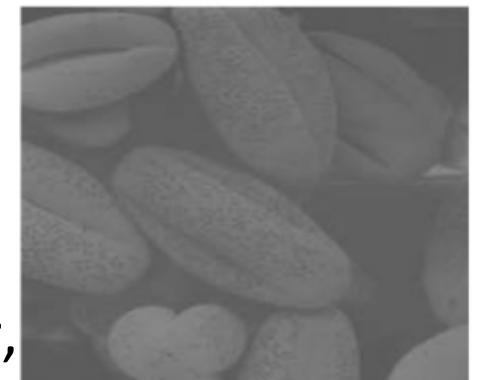
Contrast level
 $255 - 0 = 255$
(High-contrast)

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	150

Contrast level
 $150 - 100 = 50$
Low-contrast

Contrast Stretching

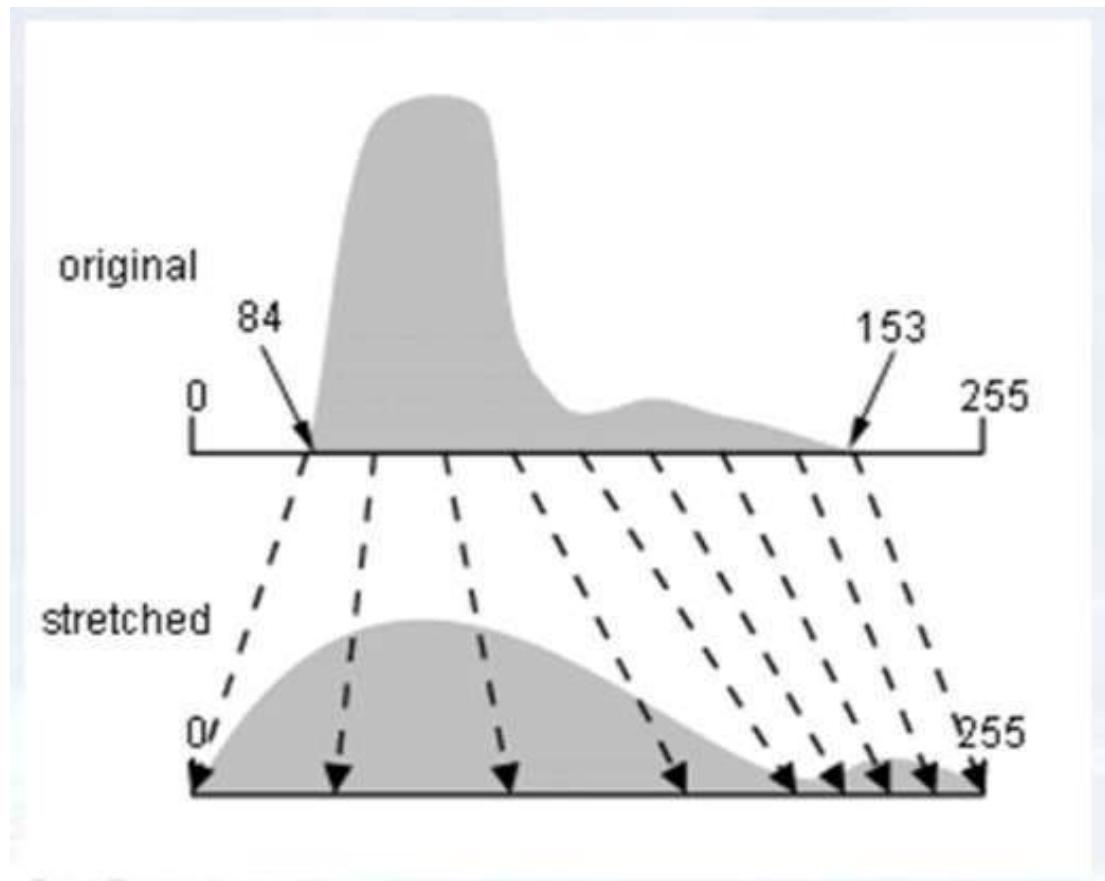
- One of the simplest **piecewise linear** functions is a contrast-stretching transformation.
- **Low-contrast** images can result from
 - poor illumination,
 - lack of dynamic range in the imaging sensor,
 - or even wrong setting of a lens aperture during image acquisition.
- The idea behind contrast stretching is to **increase** the **dynamic range** of the gray levels in the image being processed.



Contrast Stretching

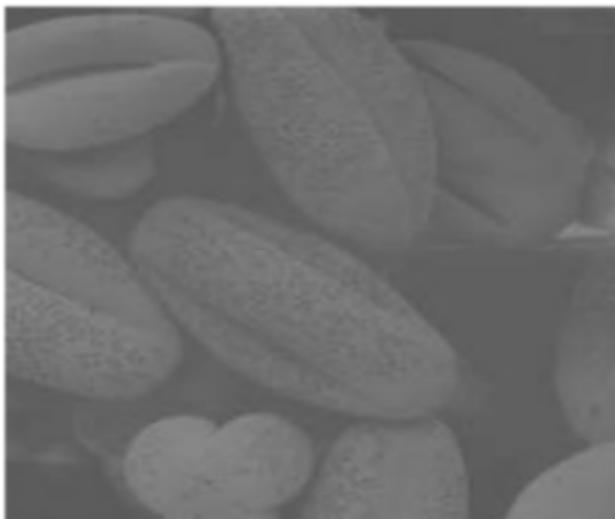
Contrast stretching is also known as normalization. The quality of image is enhanced by stretching the range of intensity values.

- ❑ Is a process that **expands** the range of intensity levels in an image so that it spans the **full intensity range** of the display device.

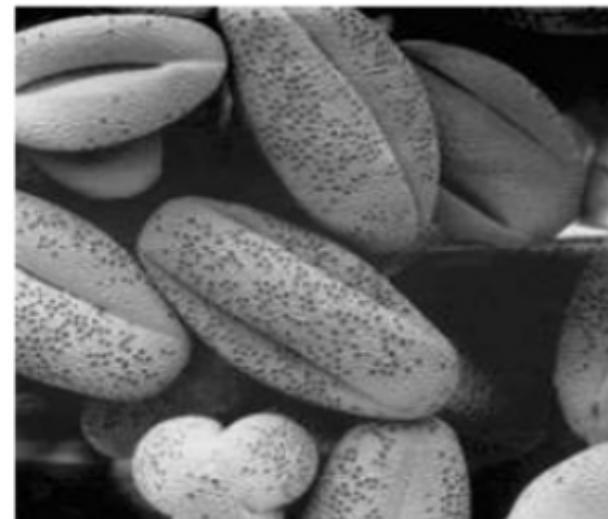


Contrast Stretching

Contrast stretching, which means that the bright pixels in the image will become brighter and the dark pixels will become darker, this means : higher contrast image.



Original Image



Contrast Enhanced Image

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [80 – 150]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

Where I is the input intensity, I_{new} is the output intensity after contrast stretching, $NewMax$ and $NewMin$ are the new intensity range (0 – 255), Max and Min are the input intensity range (80 – 150)

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

$$(255-0)/(151-100)] \\ 255/51 = 5$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching

0			

Output Image

$$I = 100 : (100-100) * [(255-0)/(151-100)] + 0 \\ I = 100 \rightarrow I_{new} = 0$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

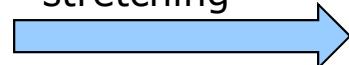
After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25		

Output Image

$$\begin{aligned} I &= 105 : (105-100) * 5 + 0 \\ I &= 105 \rightarrow I_{new} = 5 * 5 = 25 \end{aligned}$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

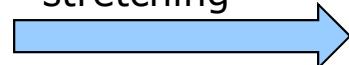
After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25	35	

Output Image

$$I = 107 : (107-100) * 5 + 0$$

$$I = 107 \rightarrow I_{new} = 7 * 5 = 35$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

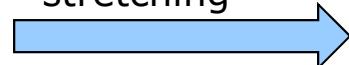
After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25	35	0

Output Image

$$\begin{aligned} I &= 100 : (100-100) * 5 + 0 \\ I &= 100 \rightarrow I_{new} = 0 * 5 = \mathbf{0} \end{aligned}$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

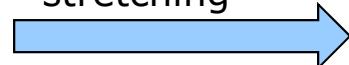
After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25	35	0
100			

Output Image

$$I = 120 : (120-100) * 5 + 0$$

$$I = 120 \rightarrow I_{new} = 20 * 5 = \mathbf{100}$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

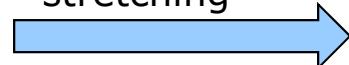
After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25	35	0
100	125		

Output Image

$$I = 125 : (125-100) * 5 + 0$$

$$I = 125 \rightarrow I_{new} = 25 * 5 = 125$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching

0	25	35	0
100	125	100	

Output Image

$$I = 120 : (120-100) * 5 + 0$$

$$I = 120 \rightarrow I_{new} = 20 * 5 = 100$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25	35	0
100	125	100	150

Output Image

$$I = 130 : (130-100) * 5 + 0$$

$$I = 130 \rightarrow I_{new} = 30 * 5 = 150$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

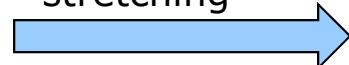
After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25	35	0
100	125	100	150
75			

Output Image

$$I = 115 : (115-100) * 5 + 0$$

$$I = 115 \rightarrow I_{new} = 15 * 5 = 75$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching

0	25	35	0
100	125	100	150
75	85		

Output Image

$$\begin{aligned} I &= 117 : (117-100) * 5 + 0 \\ I &= 117 \rightarrow I_{new} = 17 * 5 = 85 \end{aligned}$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast stretching
→

0	25	35	0
100	125	100	150
75	85	60	

Output Image

$$\begin{aligned} I &= 112 : (112-100) * 5 + 0 \\ I &= 112 \rightarrow I_{new} = 12 * 5 = 60 \end{aligned}$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast stretching
→

0	25	35	0
100	125	100	150
75	85	60	90

Output Image

$$\begin{aligned} I &= 118 : (118-100) * 5 + 0 \\ I &= 118 \rightarrow I_{new} = 18 * 5 = 90 \end{aligned}$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching



0	25	35	0
100	125	100	150
75	85	60	90
150			

Output Image

$$I = 130 : (130-100) * 5 + 0$$

$$I = 130 \rightarrow I_{new} = 30 * 5 = \mathbf{150}$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching

0	25	35	0
100	125	100	150
75	85	60	90
150	200		

Output Image

$$I = 140 : (140-100) * 5 + 0$$

$$I = 140 \rightarrow I_{new} = 40 * 5 = 200$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching

0	25	35	0
100	125	100	150
75	85	60	90
150	200	225	

Output Image

$$I = 145 : (145-100) * 5 + 0$$

$$I = 145 \rightarrow I_{new} = 45 * 5 = 225$$

Contrast Stretching

□ Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min) \frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast
stretching

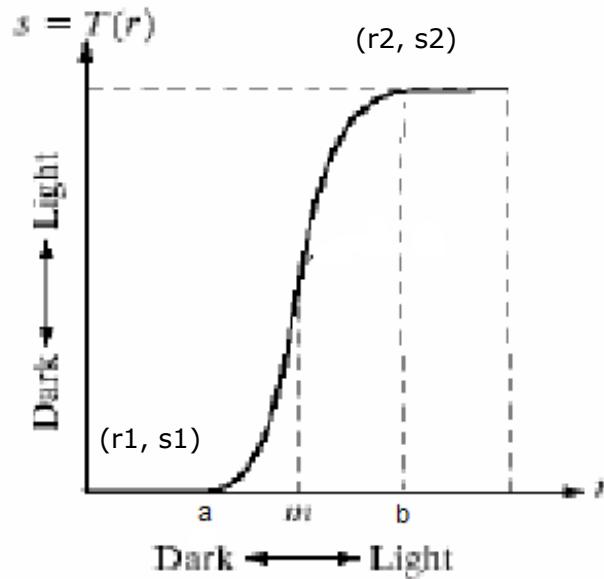
0	25	35	0
100	125	100	150
75	85	60	90
150	200	225	225

Output Image

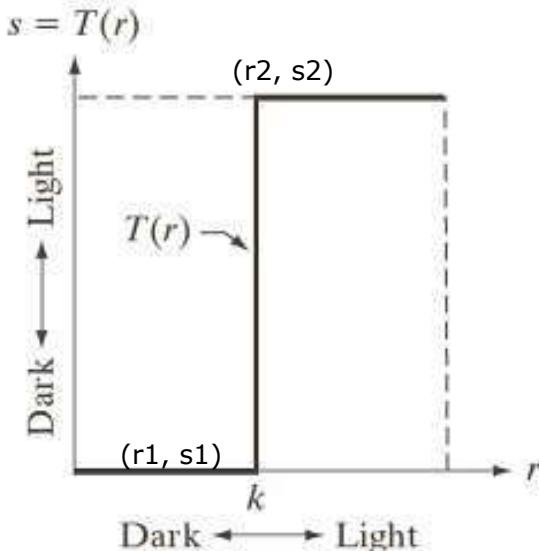
$$I = 151 : (151-100) * 5 + 0$$

$$I = 151 \rightarrow I_{new} = 51 * 5 = 225$$

Contrast Stretching and Thresholding



Contrast stretching

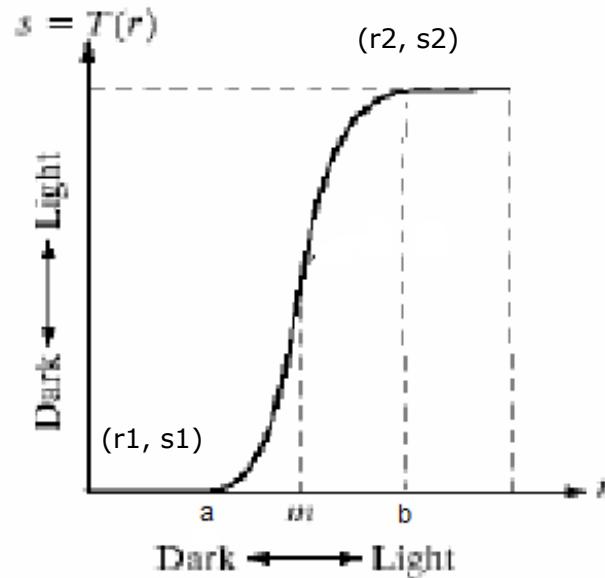


Thresholding:

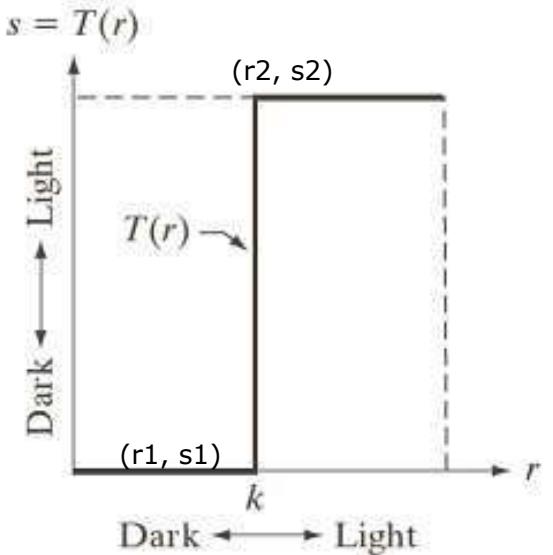
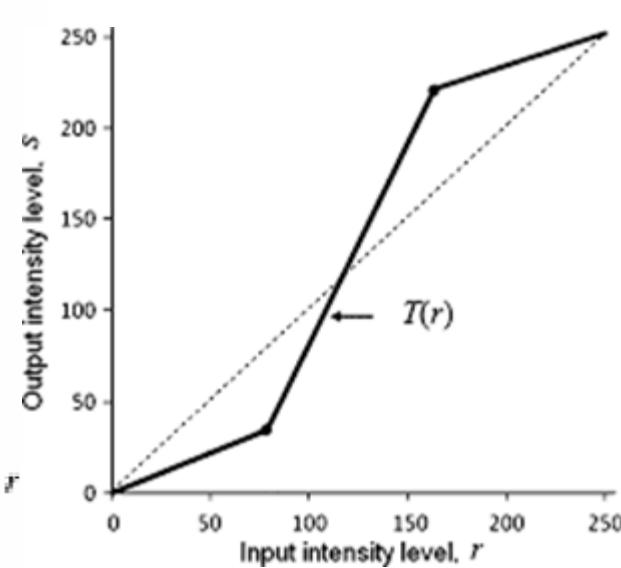
Assume that
a: rmin,
b:rmax,
k : intensity

Contrast stretching: $(r_1,s_1)=(r_{\min},0)$, $(r_2,s_2)=(r_{\max},L-1)$
Thresholding: $(r_1,s_1)=(k,0)$, $(r_2,s_2)=(k,L-1)$

Contrast Stretching and Thresholding



Contrast stretching



Thresholding:

Assume that

a: rmin,

b:rmax,

k : intensity

bright pixels in the image will become brighter and the dark pixels will become darker, this means : higher contrast image.

Contrast stretching: $(r_1, s_1) = (r_{\min}, 0)$, $(r_2, s_2) = (r_{\max}, L-1)$

Thresholding: $(r_1, s_1) = (k, 0)$, $(r_2, s_2) = (k, L-1)$

Example: Contrast Stretching

Image (r)



Image (s) after applying T
(contrast stretching)



Notice that the intensity transformation function T , made the pixels with dark intensities darker and the bright ones even more brighter, this is called **contrast stretching**

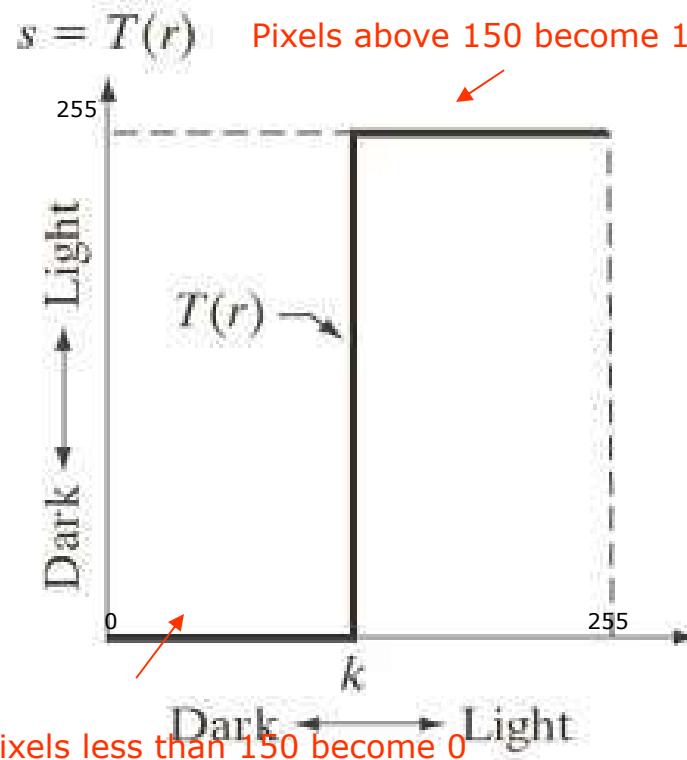
Thresholding

Remember that:

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

Or

$$s = T(r)$$



Example: suppose $m=150$ (called threshold),

(الصورة الاصلية) if r (or pixel intensity in image f) is above this threshold it becomes 1 in s (or pixel intensity in image g) (الصورة بعد التعديل), otherwise it becomes zero.

$$T = \begin{cases} \text{If } f(x,y) \geq 150; g(x,y) = 1 \\ \text{If } f(x,y) < 150; g(x,y) = 0 \end{cases}$$

Or simply...

$$T = \begin{cases} \text{If } r \geq 150; s = 1 \\ \text{If } r < 150; s = 0 \end{cases}$$

This is called
thresholding,
and it produces
a binary image!

Thresholding

Image (r)



Image (s) after applying T
(Thresholding)



Notice that the intensity transformation function T , convert the pixels with dark intensities into black and the bright pixels into white. Pixels above threshold is considered bright and below it is considered dark, and this process is called **thresholding**.

Example: Contrast Stretching and Thresholding



8-bit image with low contrast



After contrast stretching

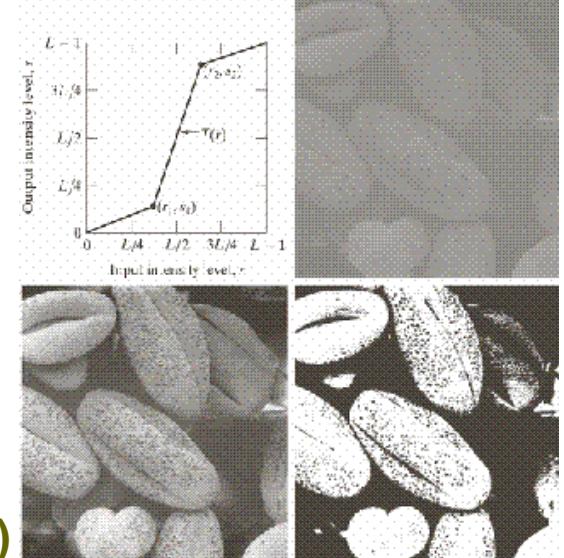
$$(r_1, s_1) = (r_{\min}, 0), (r_2, s_2) = (r_{\max}, L-1)$$



Thresholding function

$$(r_1, s_1) = (m, 0), (r_2, s_2) = (m, L-1)$$

m : mean intensity level in the image



Exercise on Contrast Stretching and Thresholding

Exercise:

the following matrix represents the pixels values of a 8-bit image (r) , apply thresholding transform assuming that the threshold m=95, find the resulting image pixel values.

solution:

Image (s)

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

Image (r)

110	120	90	130
91	94	98	200
90	91	99	100
82	96	85	90

Piecewise Linear Transformation Functions



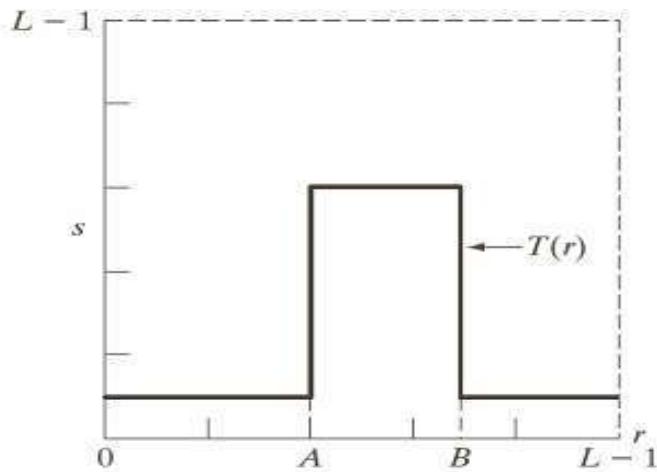
**Intensity-Level Slicing and
Bit-Plane Slicing**

Intensity-Level Slicing (gray level slicing)

Highlighting a specific range of intensities in an image.

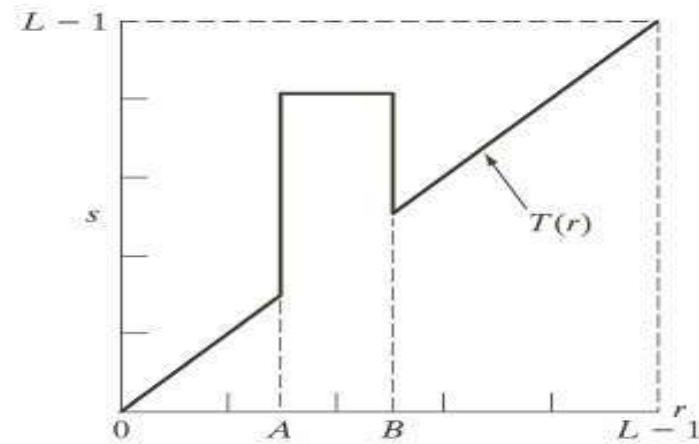
In other words, we segment certain gray level regions from the rest of the image.

Approach 1



display in one value(e.g white) all the values in the range of interest , and in another (e.g black) all other intensities

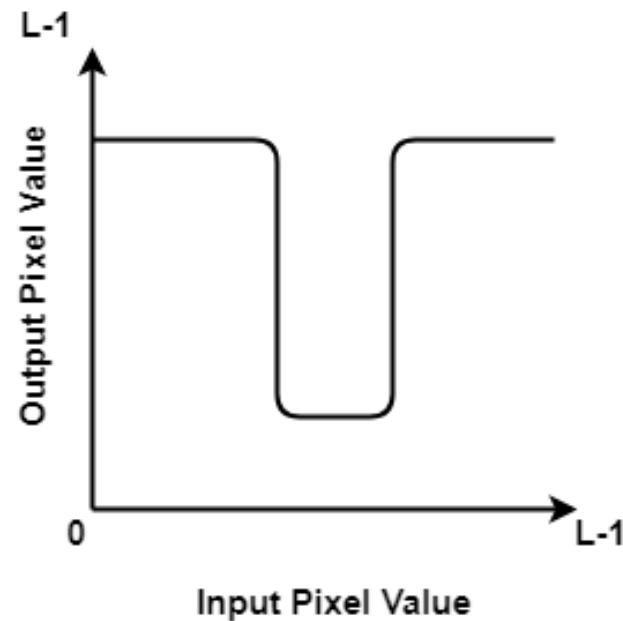
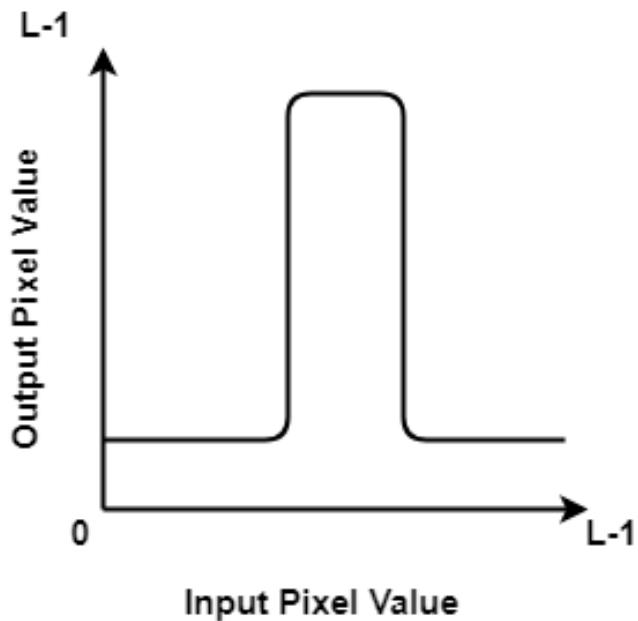
Approach 2



Brightens or darkens the desired range of intensities but leaves all other intensity levels in the image unchanged. It preserves the background

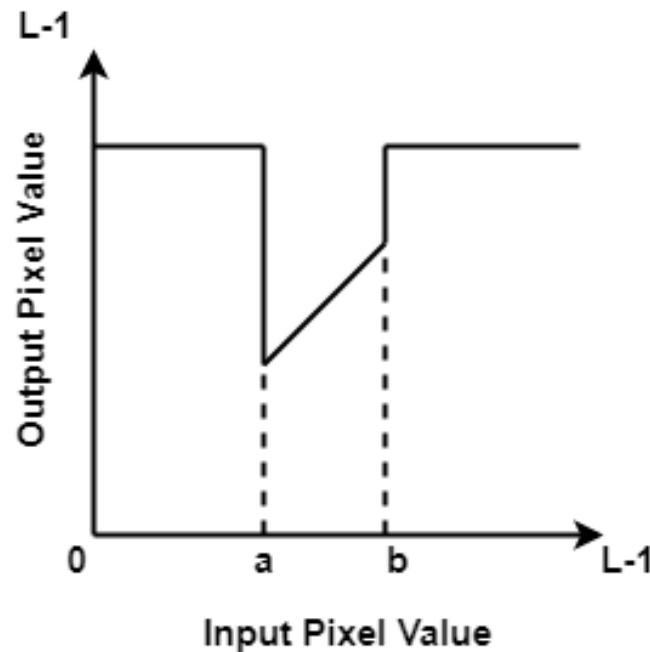
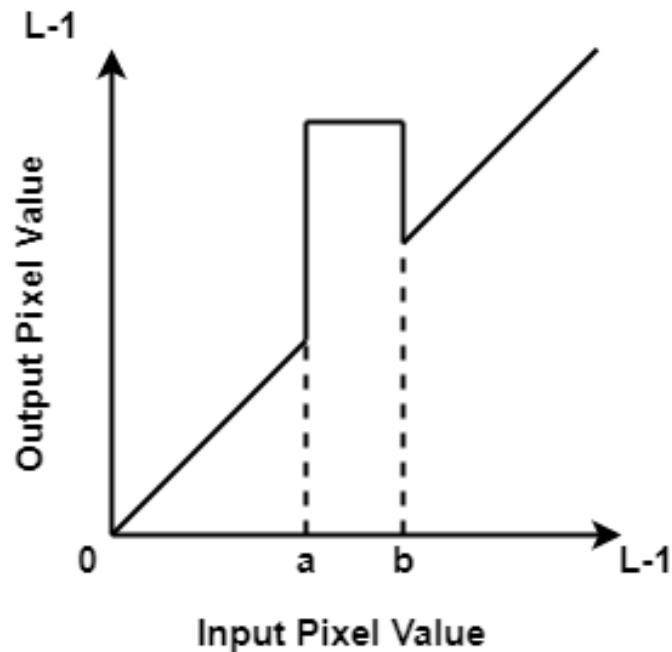
Intensity-Level Slicing: Approach 1

In the first approach, we display the desired range of intensities in white and suppress all other intensities to black or vice versa. This results in a binary image. The transformation function for both the cases is shown below.



Intensity-Level Slicing: Approach 2

In the second approach, we brighten or darken the desired range of intensities(a to b as shown below) and leave other intensities unchanged or vice versa. The transformation function for both the cases, first where the desired range is changed and second where it is unchanged, is shown below.



```

clc; clear all;

i=imread('kidney.tif');      % should be graylevel image
j=double(i);
k=double(i);
[row,col]=size(j);
T1=input('Enter the Lowest threshold value:');
T2=input('Enter the Highest threshold value:');
for x=1:row
    for y=1:col
        if((j(x,y)>T1) && (j(x,y)<T2))
            j(x,y)=i(x,y);
            k(x,y)=255;
        else
            j(x,y)=0;
            k(x,y)=0;
        end
    end
end

```



```

subplot(311), imshow(i), title('Original image')
subplot(312), imshow(uint8(j)), title('Graylevel slicing with background')
subplot(313), imshow(uint8(k)), title('Graylevel slicing without background')

```

Intensity-Level Slicing (gray level slicing)

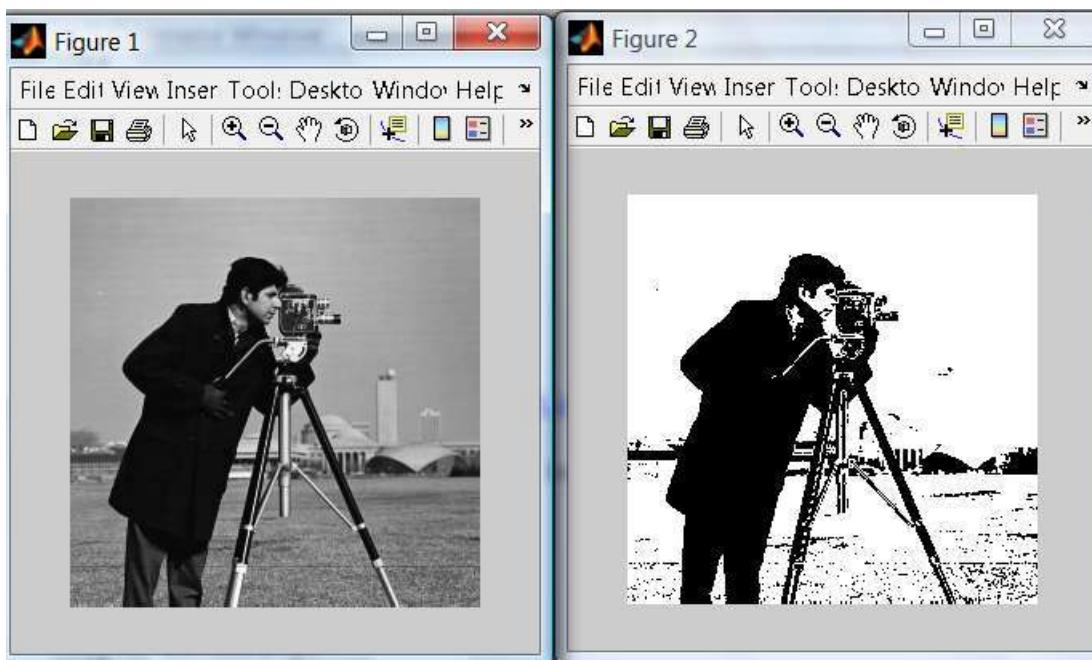
example: apply intensity level slicing in Matlab to read cameraman image , then If the pixel intensity in the old image is between (100 → 150) convert it in the new image into 255 (white). Otherwise convert it to 0 (black).

Solution:

```
x=imread('cameraman.tif');
y=x;
[w h]=size(x);
for i=1:w
    for j=1:h
        if x(i,j)>=100 && x(i,j)<=150
            y(i,j)=255;
        else
            y(i,j)=0;
        end
    end
end
figure, imshow(x);
figure, imshow(y);
```

Intensity-Level Slicing (gray level slicing)

example: apply intensity level slicing in Matlab to read cameraman image , then If the pixel intensity in the old image is between (100 → 150) convert it in the new image into 255 (white). Otherwise convert it to 0 (black).



Intensity-Level Slicing (gray level slicing)

example: apply intensity level slicing in Matlab to read cameraman image , then If the pixel intensity in the old image is between (100 → 150) convert it in the new image into 255 (white). Otherwise it leaves it the same.

Solution:

```
x=imread('cameraman.tif');
y=x;
[w h]=size(x);
for i=1:w
    for j=1:h
        if x(i,j)>=100 && x(i,j)<=200
            y(i,j)=255;
        else
            y(i,j)=x(i,j);
        end
    end
end
figure, imshow(x);
figure, imshow(y);
```

Intensity-Level Slicing (gray level slicing)

example: apply intensity level slicing in Matlab to read cameraman image , then If the pixel intensity in the old image is between (100 → 150) convert it in the new image into 255 (white). Otherwise it leaves it the same.



Exercise: Intensity-Level Slicing (gray level slicing)

Example: apply intensity level slicing to the following image, where:

- If the intensity in the image is between (50 → 100) convert it in the new image into 255 and preserve the background
- If the intensity in the image is between (50 → 100) convert it in the new image into 255, else convert it to 0.

110	120	130	135
100	94	98	200
30	60	70	30
28	29	25	27

Original Image

110	120	130	135
100	255	255	200
30	255	255	30
28	29	25	27

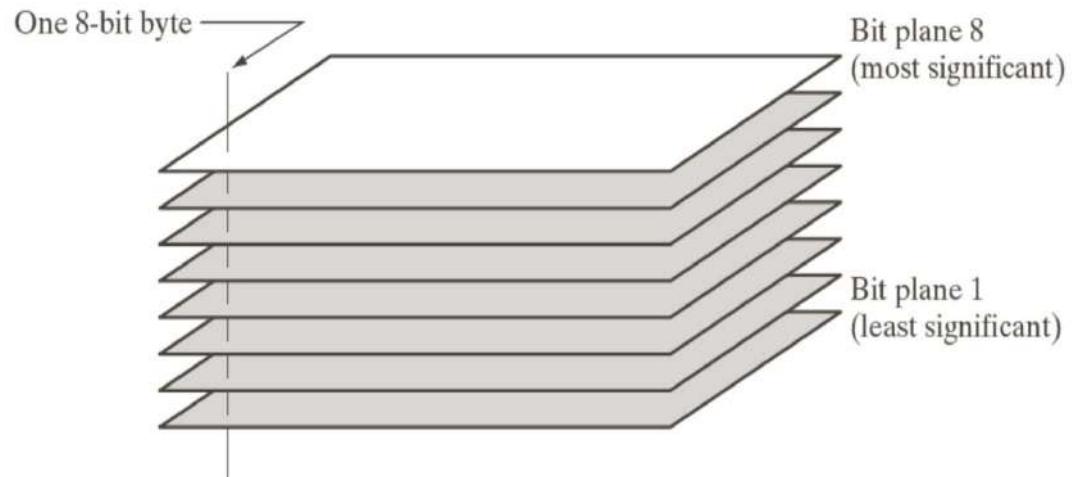
Sliced Image (a)
approach 2

0	0	0	0
0	255	255	0
0	255	255	0
0	0	0	0

Sliced Image (b)
approach 1

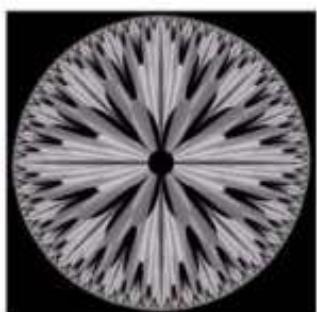
Bit-Plane Slicing

- Remember that pixels are digital numbers composed of 8 bits. Instead of highlighting gray-level range, we could highlight the contribution made by each bit.

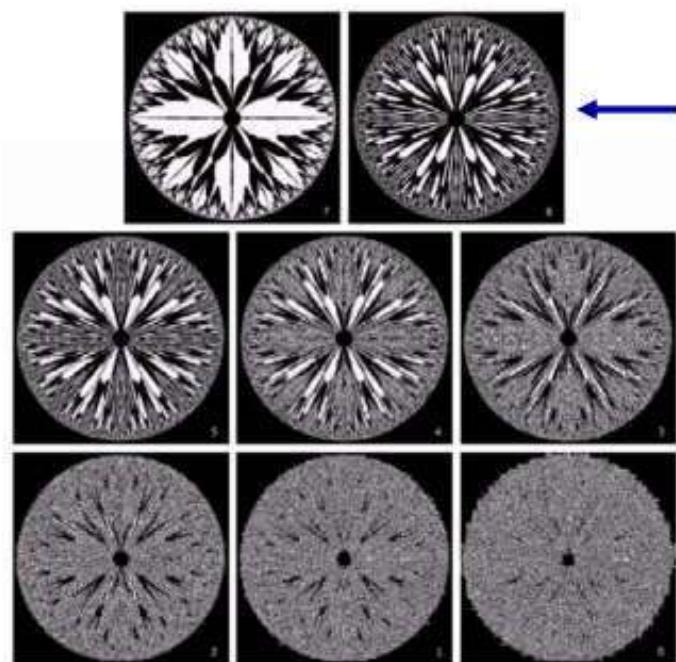
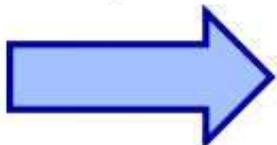


8-bit Image composed of 8 1-bit planes

Bit-Plane Slicing



Slicing into
8 bit planes



Higher-order bits contain
the majority of the visually
significant data.

Other bit planes
contain subtle
details.

FIGURE 3.13 An 8-bit fractal image. (A fractal is an image generated from mathematical formulas.) (Courtesy of Ms. Molina; D. Blaha, Swarthmore College, Swarthmore, PA.)

Bit-plane slicing :

- Suppose that each pixel in an image is represented by **8 bits**.

- Imagine that the image is composed of **eight 1-bit planes**, ranging from bit-plane 0 to bit-plane 7
- **plane 0** contains all the lowest order bits and **plane 7** contains all the high-order bits.
- The **higher-order** bits (especially the top four) contain the majority of the **visually significant data**.
- The other bit planes contribute to more **subtle details** in the image.
- **Separating** a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image, a process that aids in determining the adequacy of the number of bits used to quantize each pixel. Also, this type of decomposition is useful for image **compression**.

3. Bit-Plane Slicing



a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 500×1192 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.

3. Bit-Plane Slicing



a b c

FIGURE 3.15 Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5. Compare (c) with Fig. 3.14(a).

3. Bit-Plane Slicing (example)

We have to use bit get and bit set to extract 8 images;

0 1 1 0 0 1 0 0

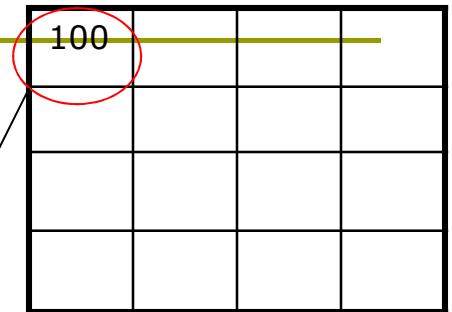


Image of bit1:
00000000

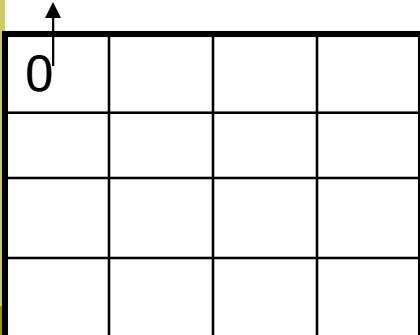


Image of bit2:
00000000

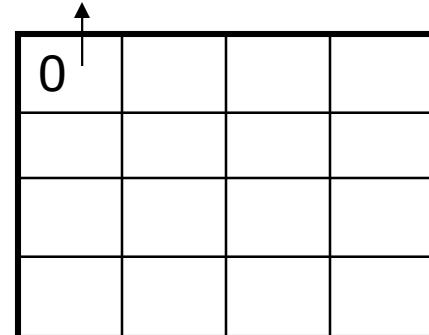


Image of bit3:
00000100

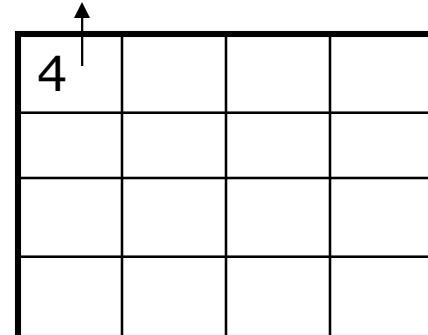


Image of bit4:
00000000

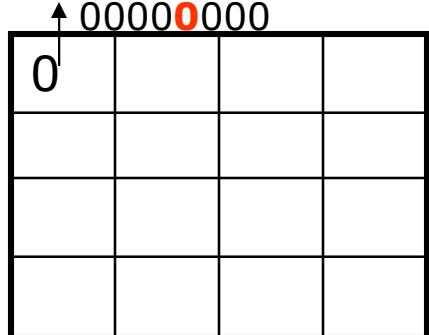


Image of bit5:
00000000

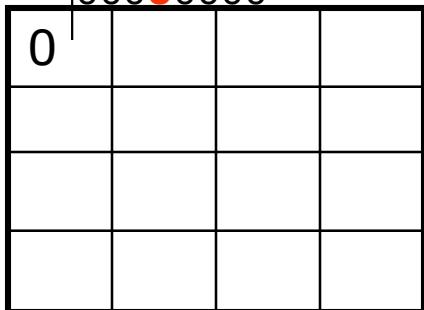


Image of bit6:
00100000

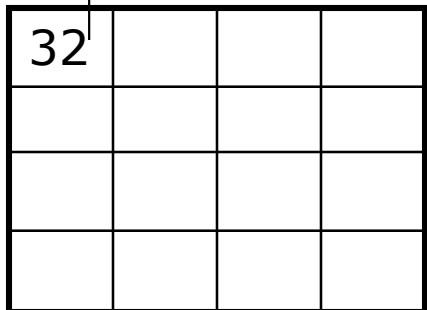


Image of bit7:
01000000

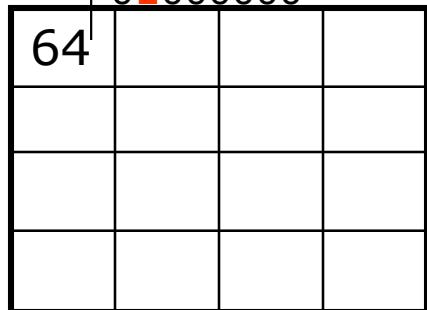
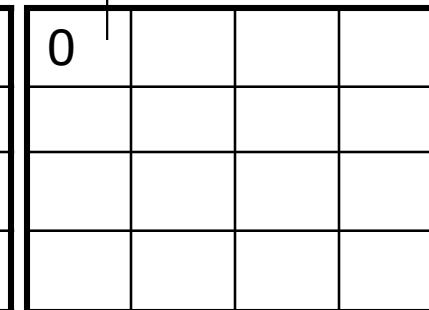


Image of bit8:
00000000



Bit-Plane Slicing in MATLAB

example: apply bit-plane slicing in Matlab to read cameraman image , then extract the image of bit 6.

Solution:

```
x=imread('cameraman.tif');
y=x*0;
[w h]=size(x);
for i=1:w
    for j=1:h
        b=bitget(x(i,j),6);
        y(i,j)=bitset(y(i,j),6,b);

    end
end

figure, imshow(x);
figure, imshow(y);
```



ENHANCEMENT USING ARITHMETIC/LOGIC OPERATIONS

ENHANCEMENT USING ARITHMETIC/LOGIC OPERATIONS

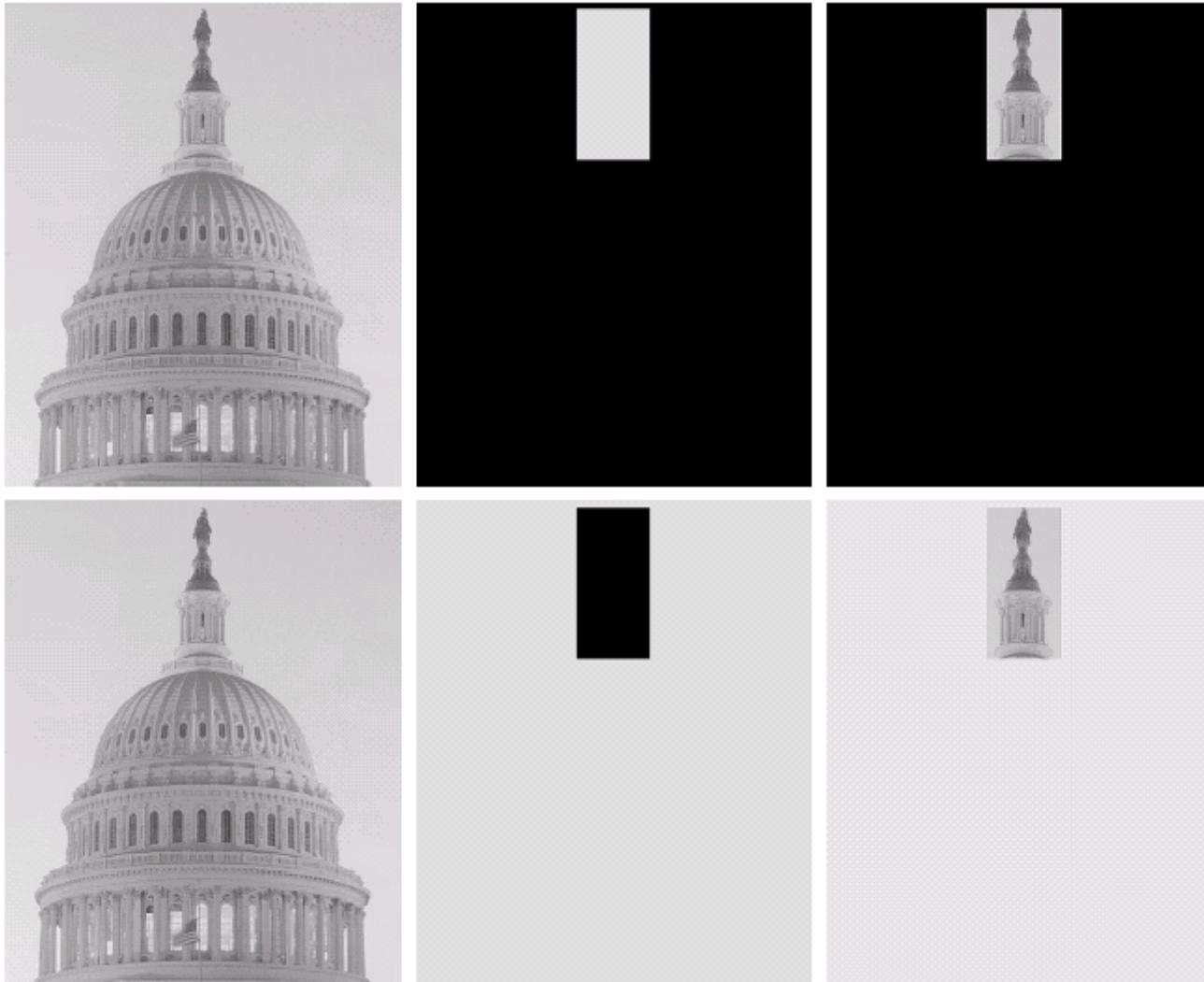
□ Arithmetic operations

- Subtraction }
 - Addition }
 - Multiplication: used as a masking operation
 - Division
- Most useful in image enhancement

□ Logical operations

- AND }
 - OR }
 - NOT
 - Frequently used in conjunction with morphological operations.
- Used for masking } For these operations, gray scale pixel values are processed as strings of binary numbers.

And/Or Masks



a b c
d e f

FIGURE 3.27
(a) Original image. (b) AND image mask.
(c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask.
(f) Result of operation OR on images (d) and (e).

Logical Operations for Masking

Consider the following is a pixel in the image:

Anding with 1's will
give same pixel

$$\begin{array}{r} \mathbf{10101011} \\ \mathbf{AND} \\ \mathbf{11111111} \\ \hline \mathbf{10101011} \end{array}$$

Oring with 0's will
give same pixel

$$\begin{array}{r} \mathbf{10101011} \\ \mathbf{OR} \\ \mathbf{00000000} \\ \hline \mathbf{10101011} \end{array}$$

$$\begin{array}{r} \mathbf{10101011} \\ \mathbf{AND} \\ \mathbf{00000000} \\ \hline \mathbf{00000000} \end{array}$$

Anding with 0's
will give all zero ,
so it will hide the
pixel

$$\begin{array}{r} \mathbf{10101011} \\ \mathbf{OR} \\ \mathbf{11111111} \\ \hline \mathbf{11111111} \end{array}$$

Oring with 1's
will give all ones,
so it will hide the
pixel

Image Subtraction: Example

Subtraction of two images is often used to detect motion. Consider the case where nothing has changed in a scene; the image resulting from subtraction of two sequential images is filled with zeros - a black image. If something has moved in the scene, subtraction produces a nonzero result at the location of the movement.

Image Subtraction: Motion Tracking



background image



live image



difference image

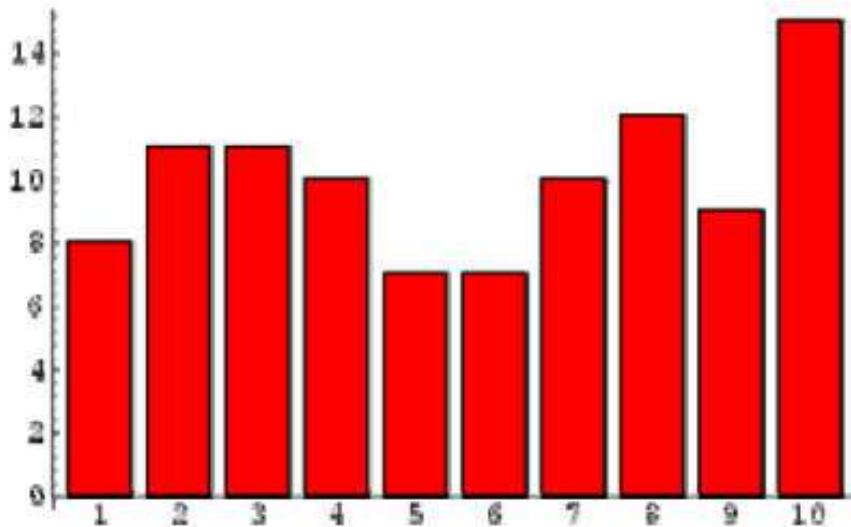


HISTOGRAM EQUALIZATION

Histogram

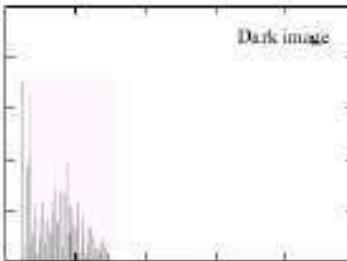
What is a histogram?

- A graph indicating the number of times each gray level occurs in the image, i.e. **frequency** of the **brightness value** in the image
- The image histogram carries important information about the image content
- Algorithm:
 - Assign zero values to all elements of the array h_f ;
 - For all pixels (x,y) of the image f , increment $h_f[f(x,y)]$ by 1.

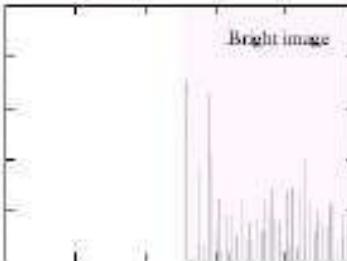


Histogram of the image:

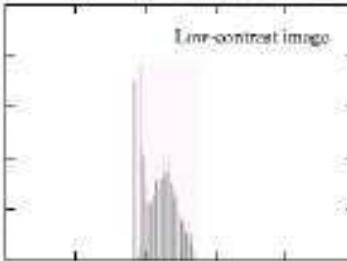
Dark image



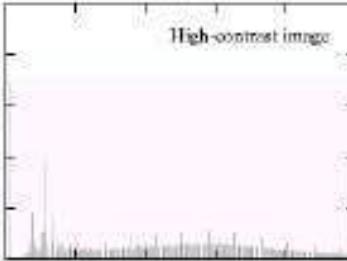
Bright image



Low contrast image

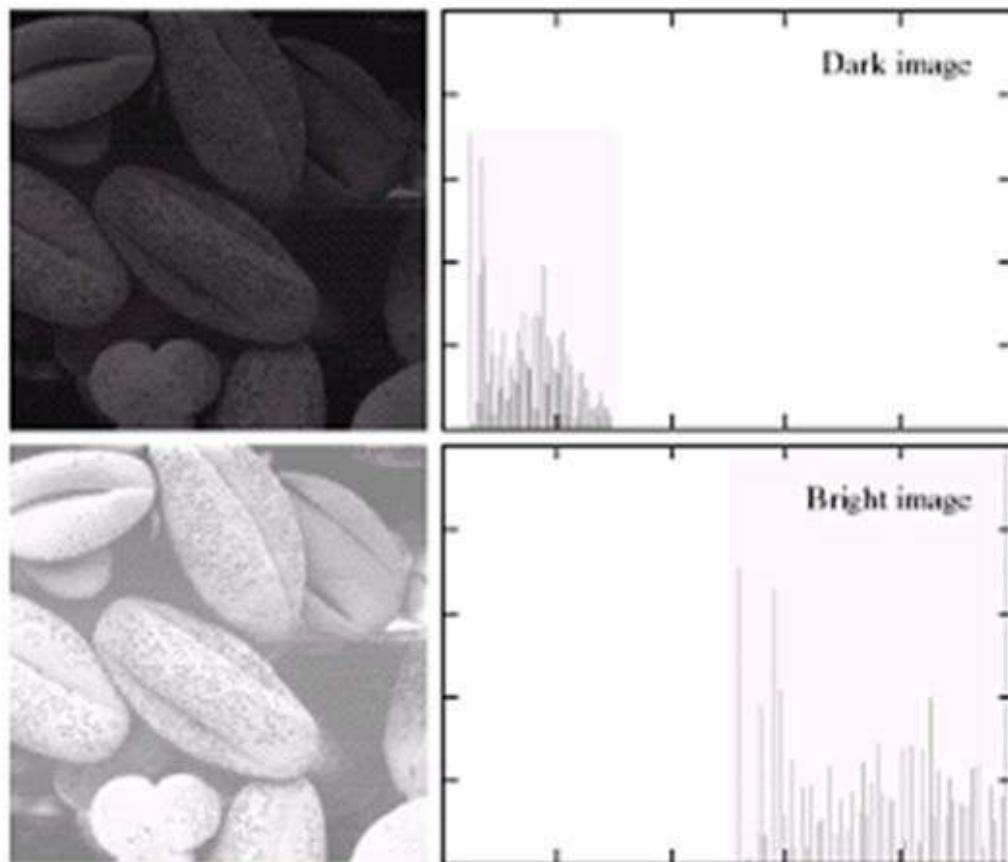


High contrast image



Compare the four images and their histograms.

Example



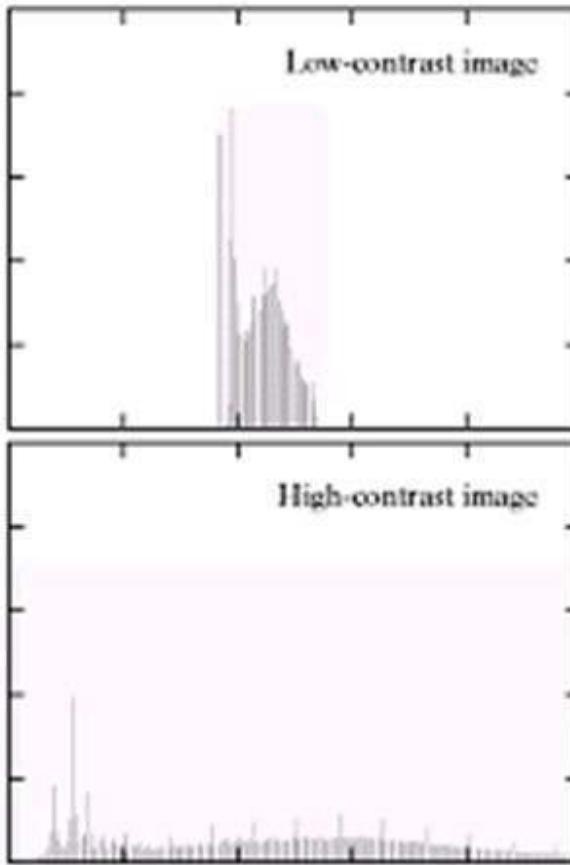
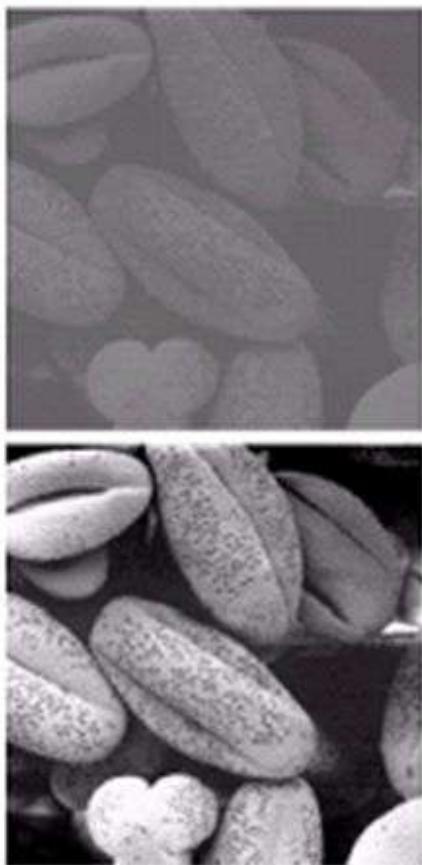
Dark image

Components of histogram are concentrated on the low side of the gray scale.

Bright image

Components of histogram are concentrated on the high side of the gray scale.

Example



Low-contrast image

**histogram is
narrow and
centered toward
the middle of the
gray scale**

High-contrast image
**histogram covers
broad range of the
gray scale and the
distribution of pixels
is not too far from
uniform, with very
few vertical lines
being much higher
than the others**

Histogram

- ❑ Histogram of a digital image

$$h(r_k) = n_k$$

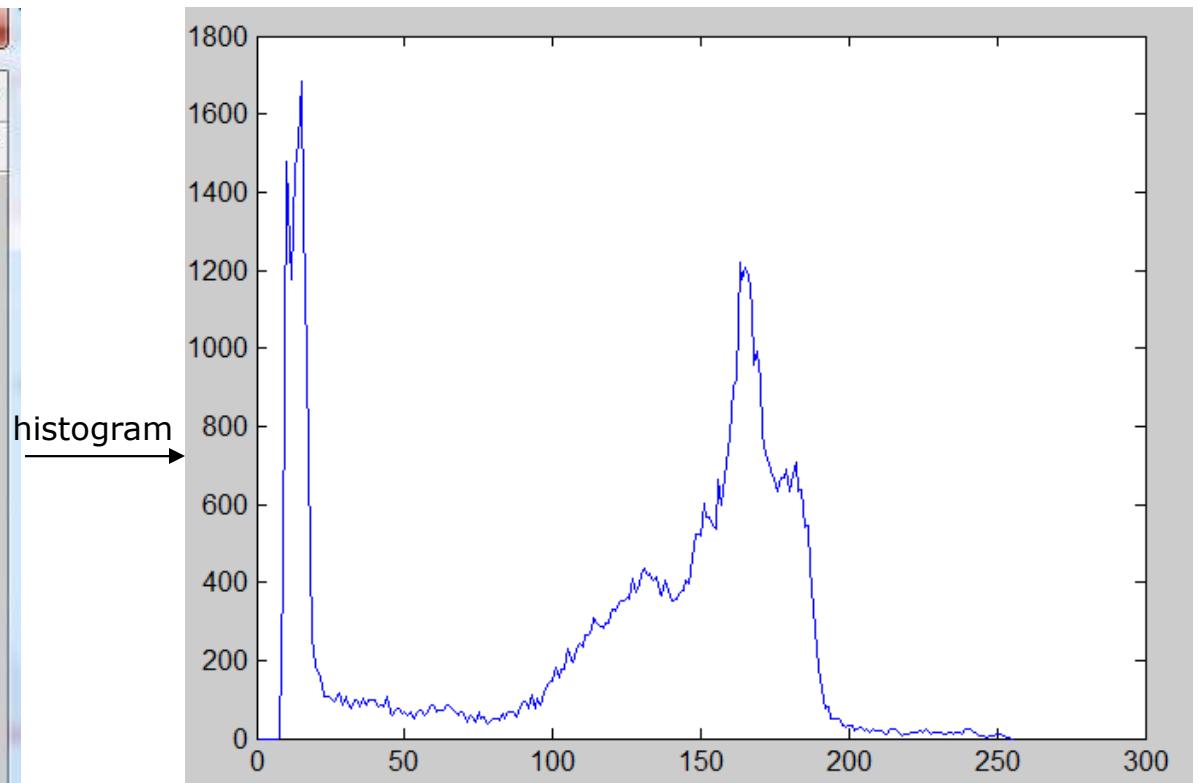
Where:

r_k : kth gray level

n_k : # of pixels with having gray level r_k

- ❑ We manipulate Histogram for image enhancement.
- ❑ Histogram data is useful in many applications like image compression and segmentation.

Histogram of the image:



$$h(r_k) = n_k$$

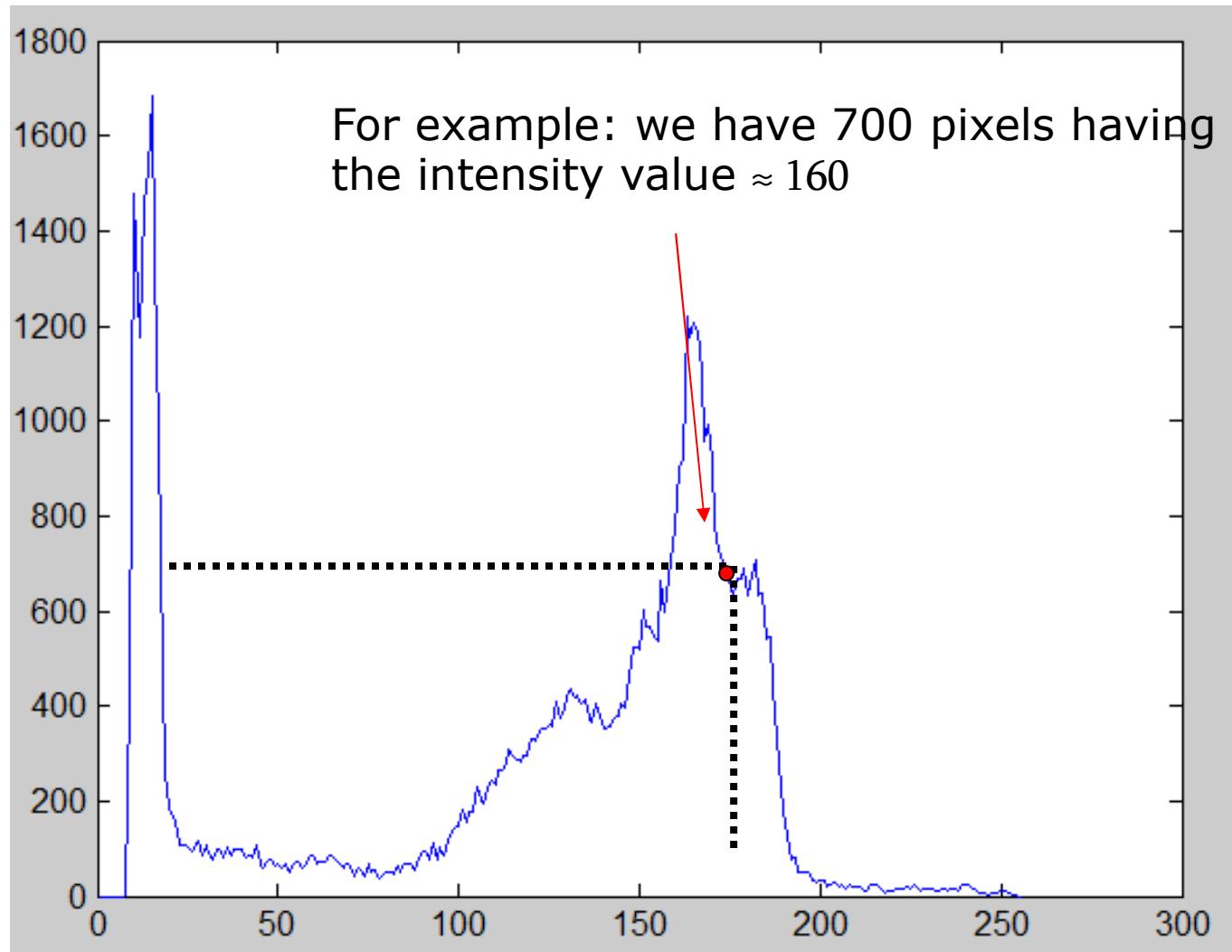
Where:

r_k : kth gray level

n_k : # of pixels with having gray level r_k

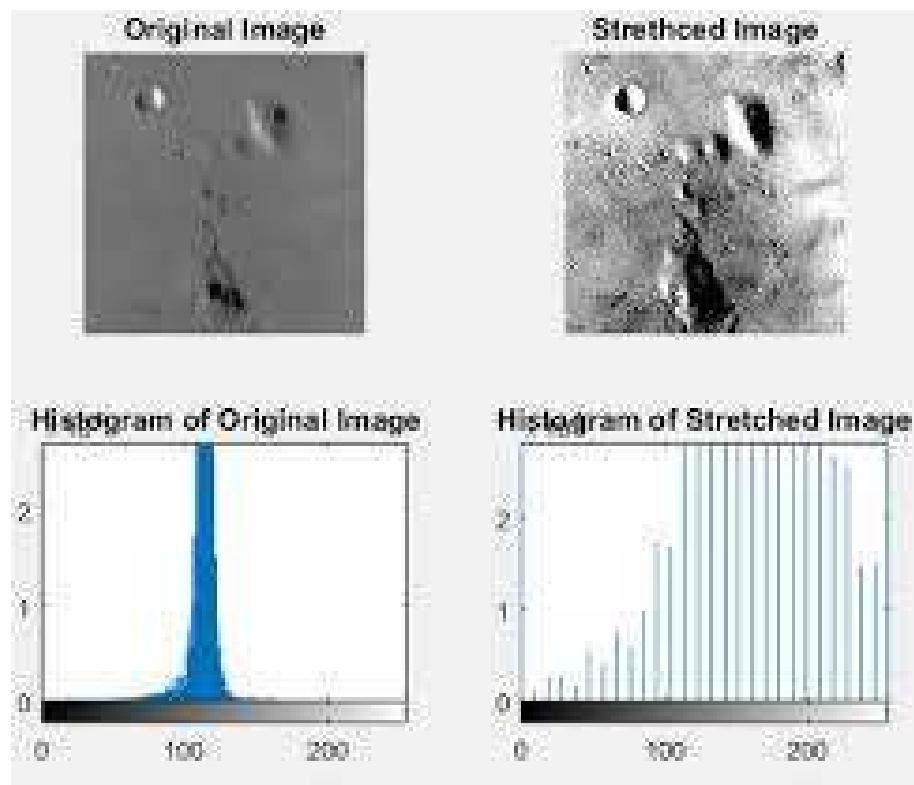
هو تمثيل لعدد البكسل في كل قيمة لونية من درجات gray levels

Histogram of the image:



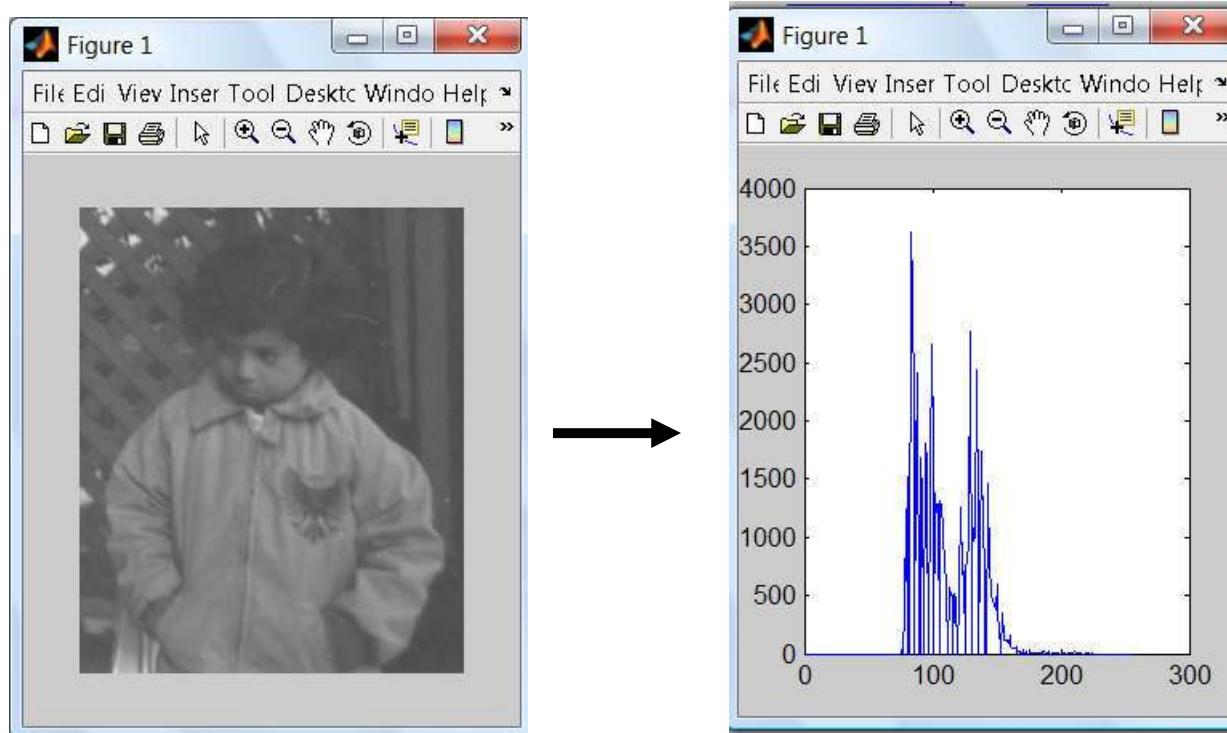
Histogram Equalization

- ❑ **Histogram Equalization** is an image processing technique used to **improve contrast** in images .
- ❑ It accomplishes this by effectively **spreading** out the **most frequent** intensity values, i.e. stretching out the intensity range of the **image**.



Histogram equalization

We have this image in matlab called pout.tif, when we plot its histogram it is showed like this:

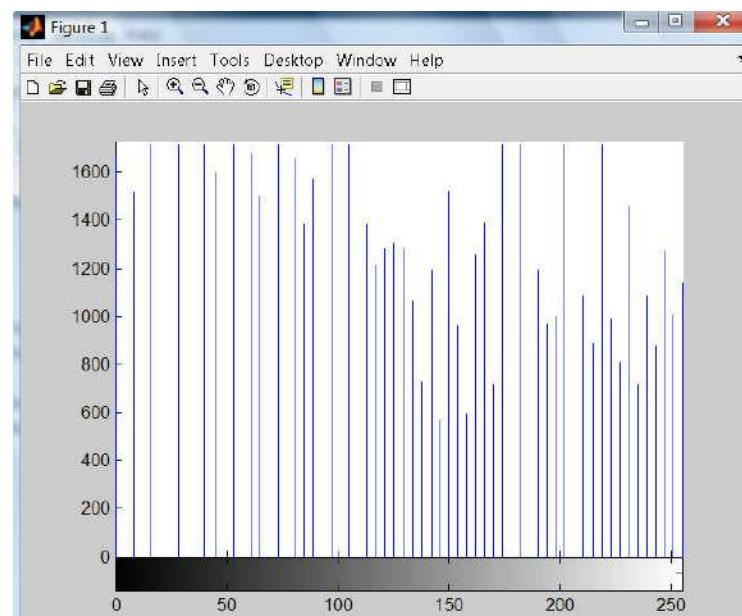
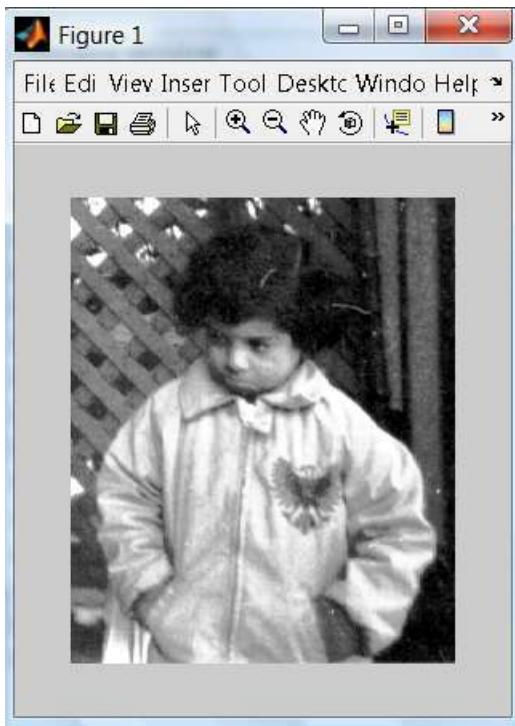


Notice that the pixels intensity values are concentrated on the middle (low contrast)

Histogram equalization

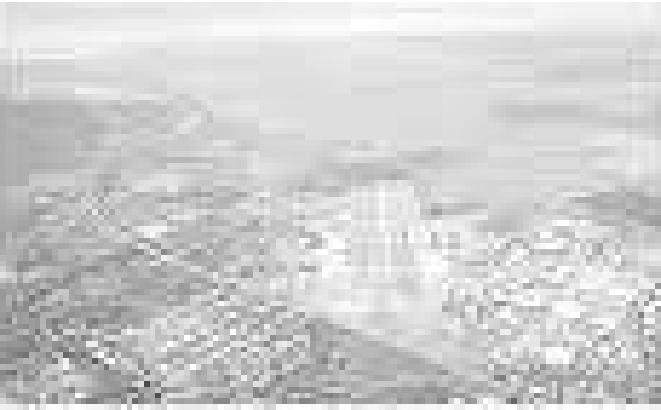
histogram equalization : is the process for adjusting image intensities to enhance contrast.

In matlab : we use **histeq** function



Histogram produces pixels having values that are distributed throughout the range

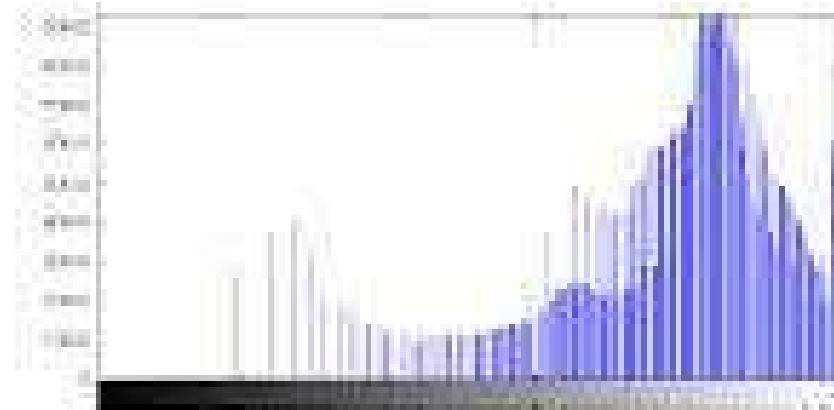
Example: Histogram equalization



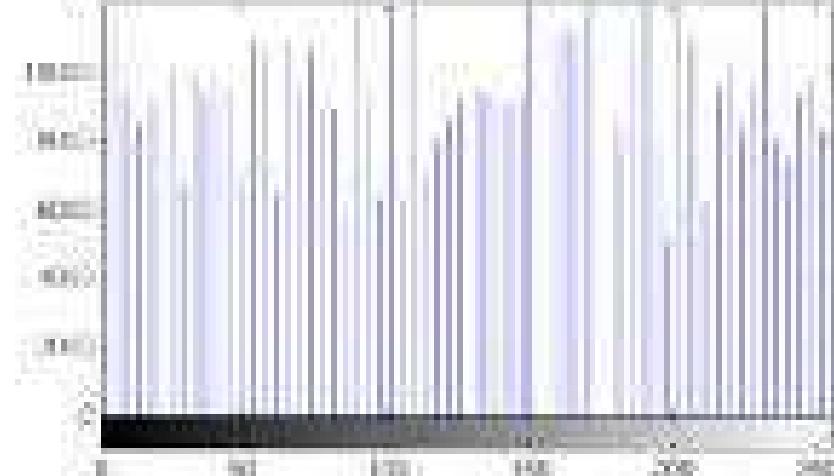
(a)



(b)



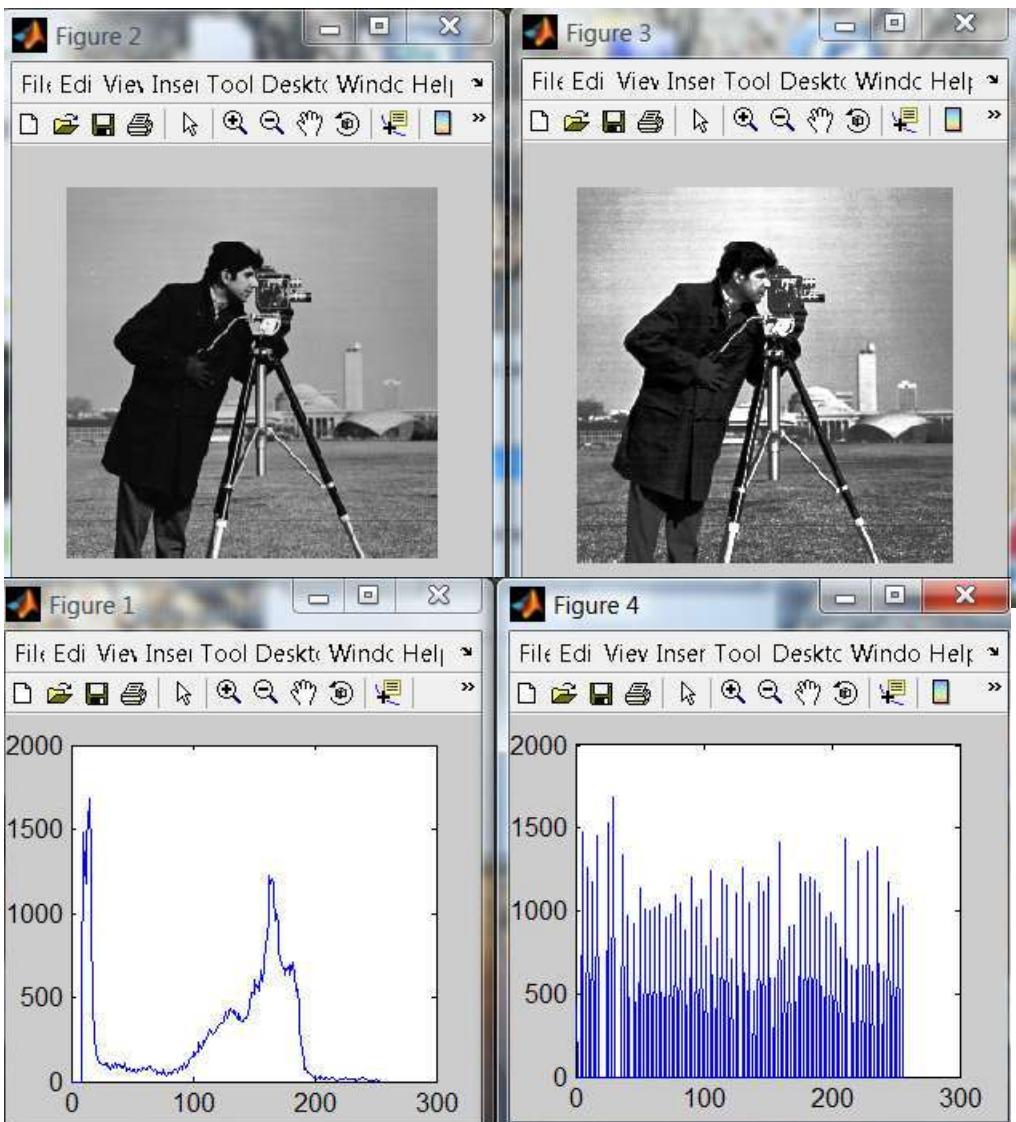
(c)



(d)

Fig. 1. In (a) the contrast is too low, contrast is enhanced in (b) using histogram equalization. The histograms in (c) show

Histogram equalization of the image:



Notice that histogram equalization does not always produce a good result

Equalization (mathematically)

$$g(x) = [(T(X)/n) * L] - 1$$

$$g(x) = (L/n) \cdot T(X) - 1$$

Where,

$G(X)$: the new image after equalization

L : No of gray levels 2^K

n : No of pixels

$T(x)$: cumulative sum of each gray level

Example: Histogram Equalization

- Apply histogram equalization to the following 3-bit image:

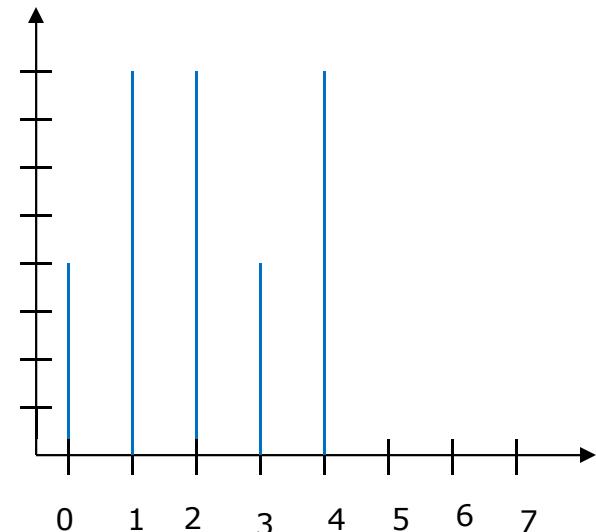
$$L = 2^3 = 8$$

$$\frac{L}{n} = \frac{8}{32} = 0.25$$

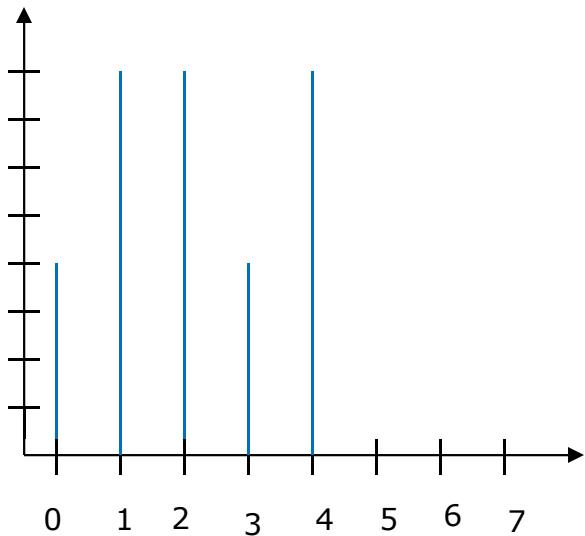
$$\begin{aligned} n &= 4 * 8 \\ &= 32 \end{aligned}$$

0	1	1	2	2	3	4	4
0	1	1	2	2	3	4	4
0	1	1	2	2	3	4	4
0	1	1	2	2	3	4	4

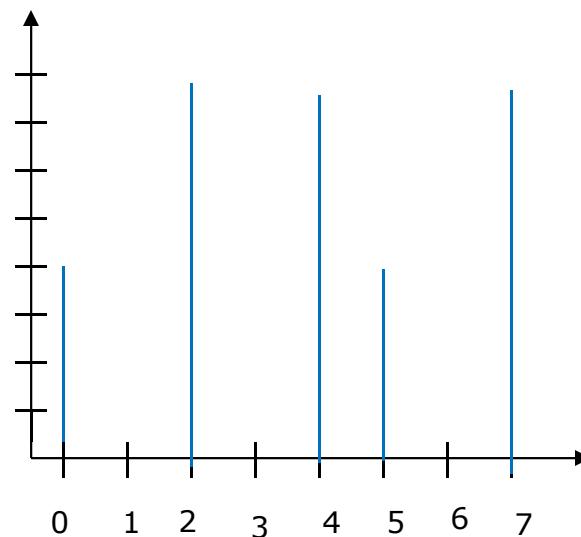
r	Freq.(r)	T(r)	S= (L/n) * T(r) - 1
0	4	4	$0.25 * 4 - 1 = 0$
1	8	12	$0.25 * 12 - 1 = 2$
2	8	20	$0.25 * 20 - 1 = 4$
3	4	24	$0.25 * 24 - 1 = 5$
4	8	32	$0.25 * 32 - 1 = 7$



Example: Histogram Equalization



Before histogram
equalization



After histogram
equalization

Equalization

L grayl evels	X عدد البكسل لكل Graylevel	T(X) مجموع تراكمي للبكسل	G(x)
0	1	1	0
1	3	4	0
2	5	9	1
3	6	15	2
4	6	21	4
5	6	27	5
6	2	29	6
7	3	32	7

Assume that we have (3bits per pixels) or 8 levels of grayscale, and we want to equalize the following image example.

$$G(x) = (L/n) \cdot T(x) - 1$$

$$= (8/32) \cdot T(x) - 1$$

8 عدد ال
graylevel

عدد البكسلات الكلي
No of pixels

Example

Table 2.1: Example of histogram equalization.

Original value x	Frequency $\#(x)$	Cumulative frequency $\delta(x)$	New value $f(x)$
0	1	1	0
1	9	10	2
2	8	18	5
3	6	24	7
4	1	25	7
5	1	26	8
6	1	27	8
7	1	28	8
8	2	30	9
9	0	30	9

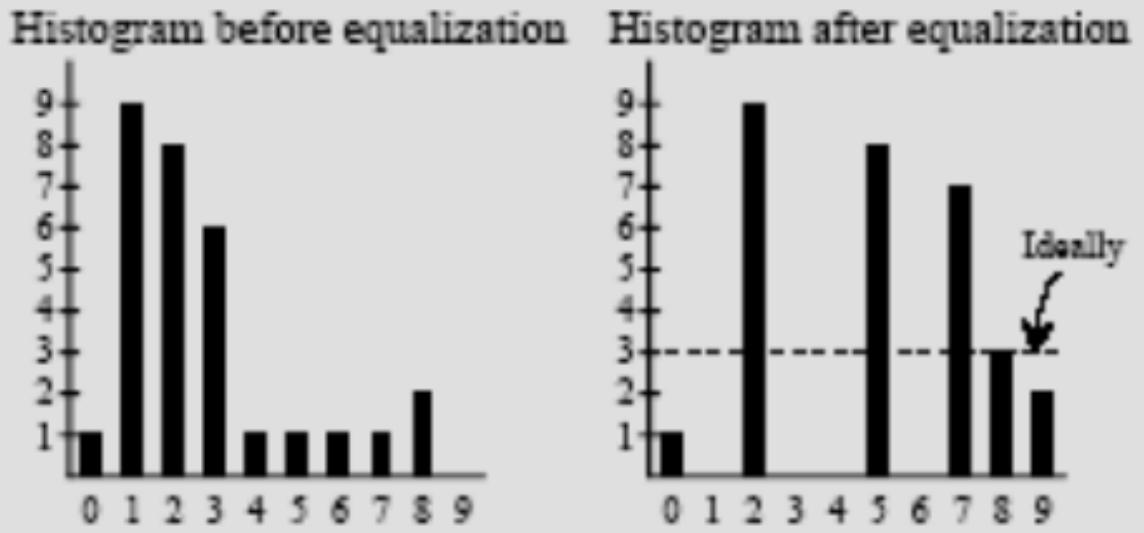
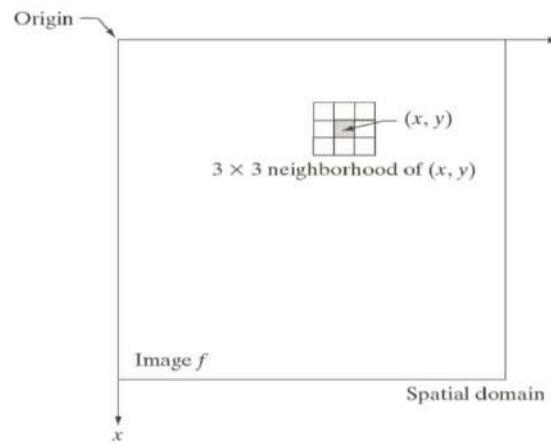


Figure 2.10: Example of histogram equalization.

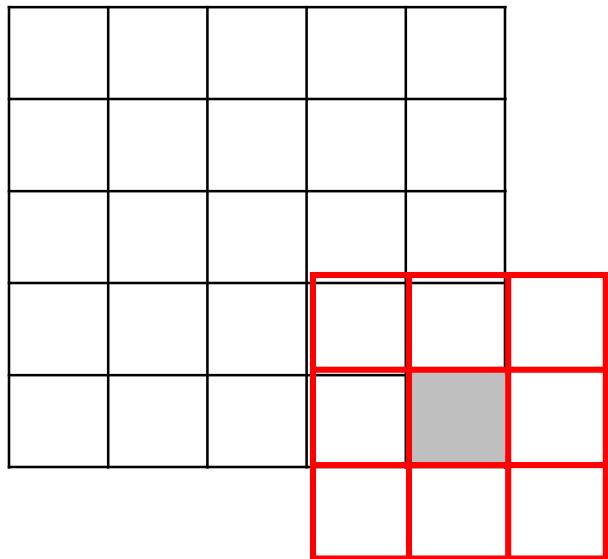
Spatial filters

Remember that types of neighborhood:

- **intensity transformation**: neighborhood of size 1×1
- **spatial filter** (or mask ,kernel, template or window): neighborhood of larger size , like 3×3 mask
- It is the **procedure** of moving the location of neighbourhood and performing a predefined operation. so the spatial filter mask is moved from point to point in an image. At each point (x,y) , the response of the filter is calculated



Spatial Filtering



$$\begin{aligned} R = & w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + w(0,0)f(x,y) + \dots \\ & + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \end{aligned}$$

Spatial Filters

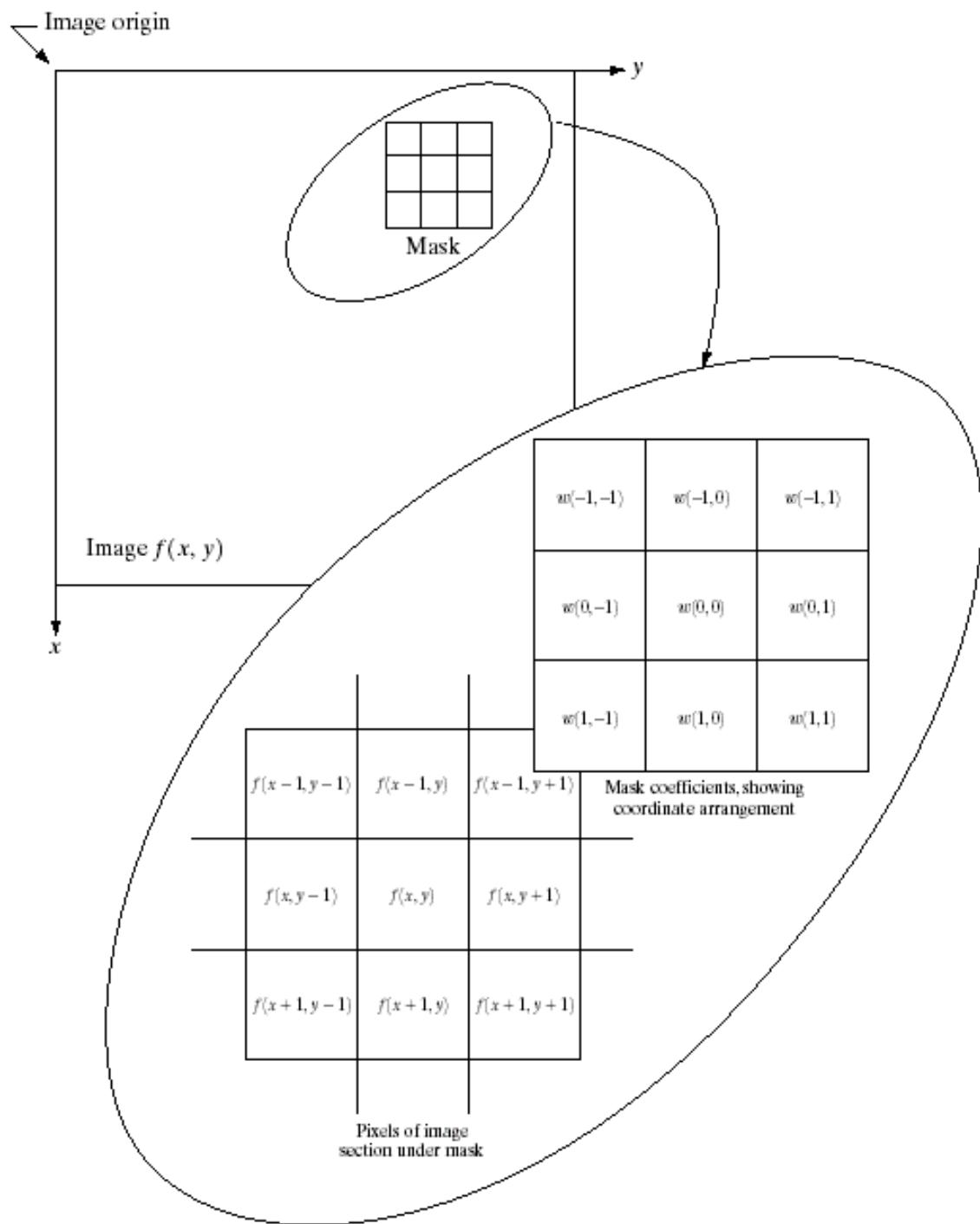


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3×3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

Spatial filters

- The filter mask is moved from point to point in an image.
- At each point (x,y) , the response of the filter is calculated.
- Linear spatial filtering
 - 3x3 mask
 - $$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + w(0,0)f(x,y) + \dots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1)$$
 - $w(0,0)$ coincides with $f(x,y)$ indicating that the mask is centered at $f(x,y)$.
- Our focus will be on masks of odd sizes.
- Image of size $M \times N$, filter mask of size $m \times n$.

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

where $a = (m-1)/2$ and $b = (n-1)/2$

Spatial filters

		Padded f								
		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
↙ Origin		$f(x, y)$								
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	2	3	0	0	0
0	0	0	0	0	4	5	6	0	0	0
0	0	0	0	0	7	8	9	0	0	0
(a)		(b)								
↖ Initial position for w		Full correlation result								
1	2	3	0	0	0	0	0	0	0	0
4	5	6	0	0	0	0	0	0	0	0
7	8	9	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	9	8	7
0	0	0	0	1	0	0	0	0	6	5
0	0	0	0	0	0	0	0	0	3	2
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
(c)		(d)								
Cropped correlation result		(e)								
0	0	0	0	0	0	0	0	0	0	0
0	9	8	7	0	0	0	0	0	0	0
0	6	5	4	0	0	0	0	0	0	0
0	3	2	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Spatial filters

- 1)Smoothing filters - low pass
- 2)Sharpening filters – high pass

Spatial filters : Smoothing (low pass)

Use for:

- 1)blurring
- 2) noise reduction.

How it works? The value of every pixel is replaced by the average of the gray levels in the neighborhood.

Type of smoothing filters:

1. Standard average
 2. weighted average.
 3. Median filter
-
- The diagram consists of three horizontal curly braces. The first brace groups the first two items (Standard average and weighted average) under the label "linear". The second brace groups the third item (Median filter) under the label "Order statistics".

Spatial filters : Smoothing

linear smoothing : averaging kernels

The output (response) of a smoothing, linear spatial filter is simply the **average** of the pixels contained in the neighborhood of the filter mask.

Desirable effect: the most application of smoothing is **noise reduction**, because random noise typically consists of sharp transitions in gray levels,

Undesirable effect: the undesirable side effect is **blur edges**. edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels.

Standard average

1	1	1
1	1	1
1	1	1

$$\frac{1}{9} \times$$

weighted average.

1	2	1
2	4	2
1	2	1

$$\frac{1}{16} \times$$



A box filter



The basic strategy is to reduce blurring.

General implementation:

General implementation for filtering an $M \times N$ image with a weighted averaging filter of size $m \times n$ (m and n odd) is given by the expression:

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

Spatial filters : Smoothing Standard and weighted Average- example

110	120	90	130
91	94	98	200
90	91	99	100
82	96	85	90

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

The mask is moved from point to point in an image. At each point (x,y) , the response of the filter is calculated

Standard averaging filter:

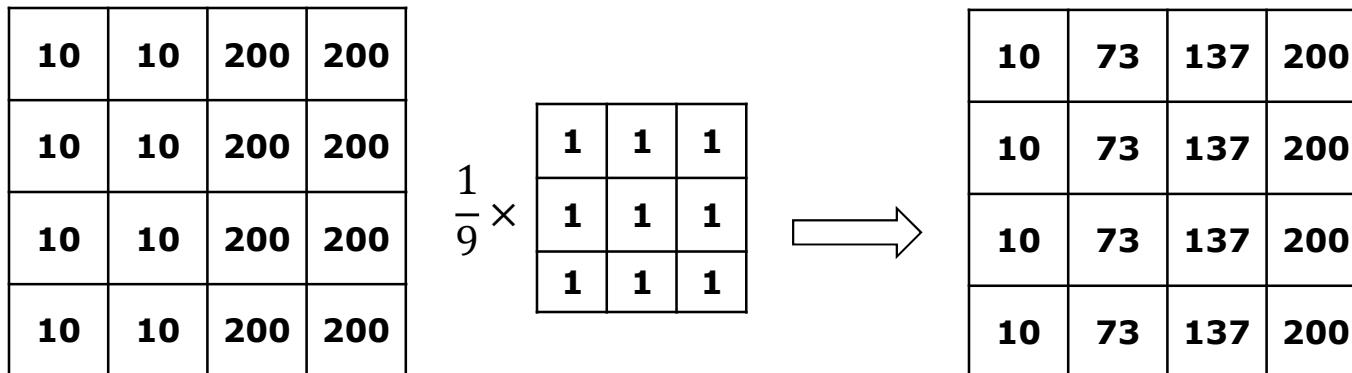
$$(110 + 120 + 90 + 91 + 94 + 98 + 90 + 91 + 99) / 9 = 883 / 9 = 98.1$$

Weighted averaging filter:

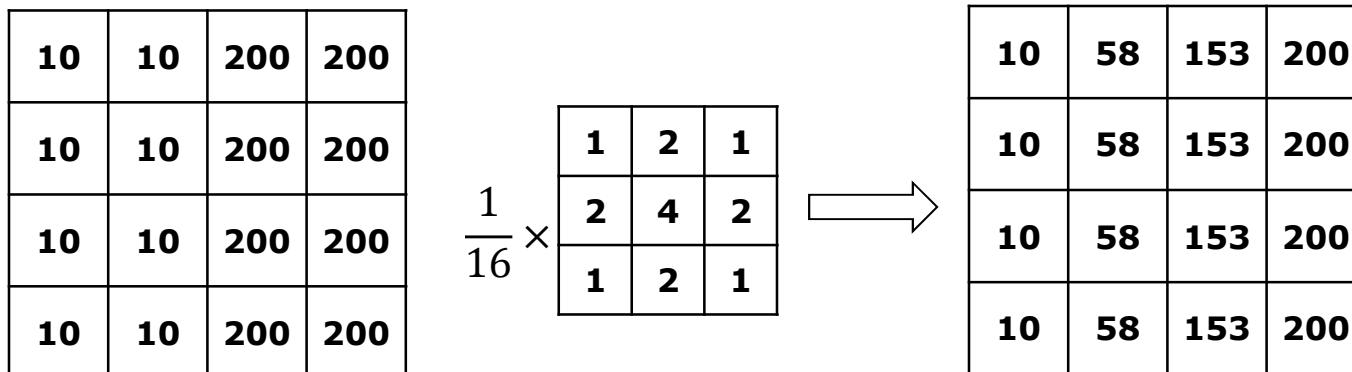
$$(110 + 2 \times 120 + 90 + 2 \times 91 + 4 \times 94 + 2 \times 98 + 90 + 2 \times 91 + 99) / 16 = 97.8$$

The difference between the std. average and the weighed average

The following image represent the gray values of an edge. The following example will explain the effect of standard average on it:

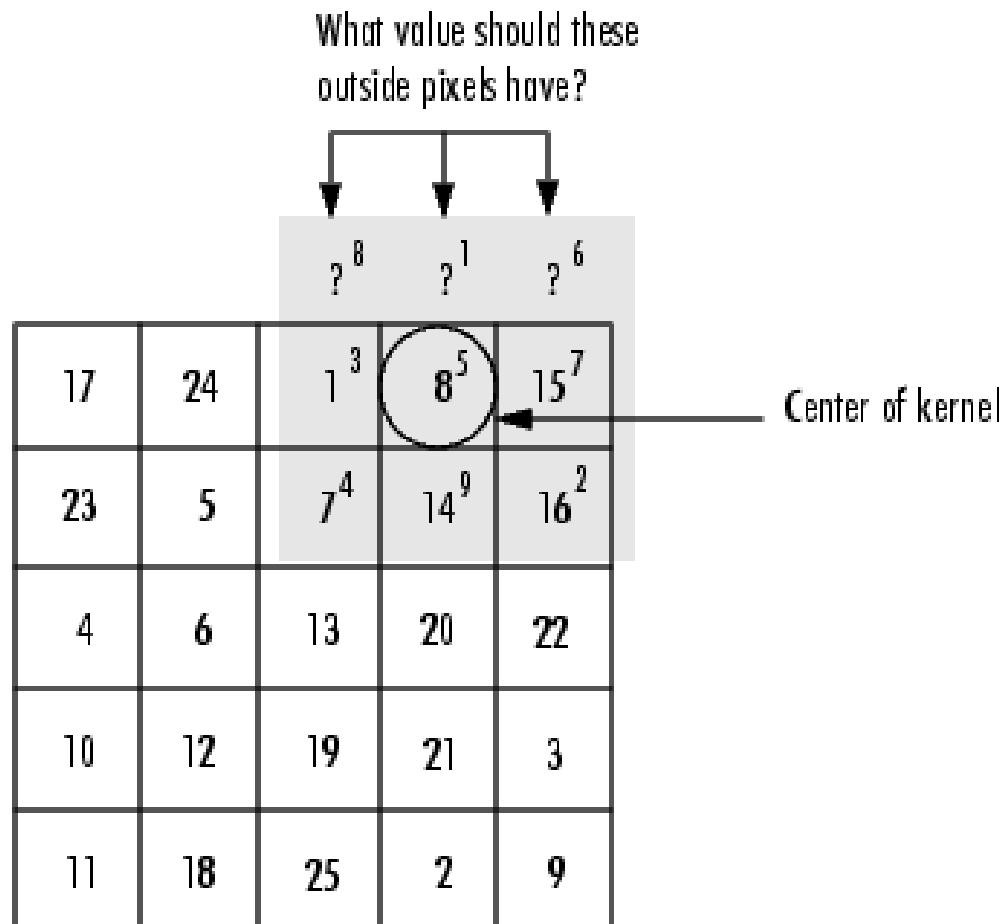


After applying weighted average:



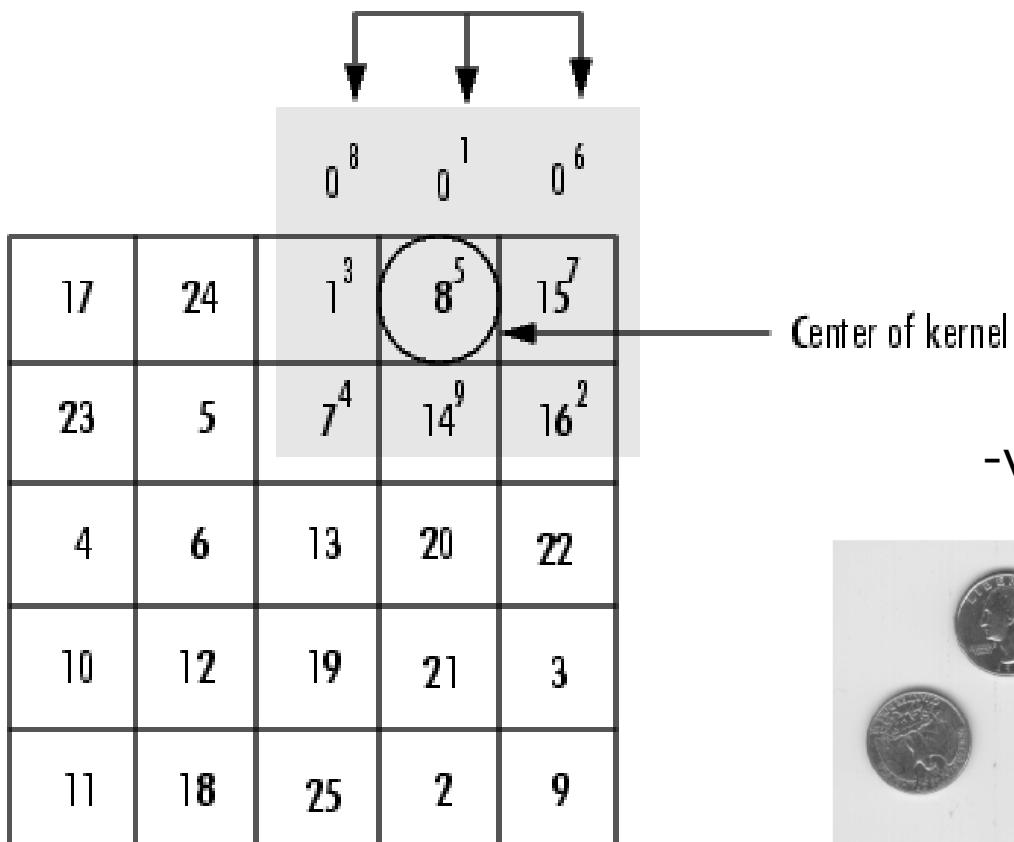
As we note applying weighted average reduce the amount of blurring on the edges

What happens when the Values of the Kernel Fall Outside the Image??!



First solution :Zero padding,

Outside pixels are assumed to be 0.

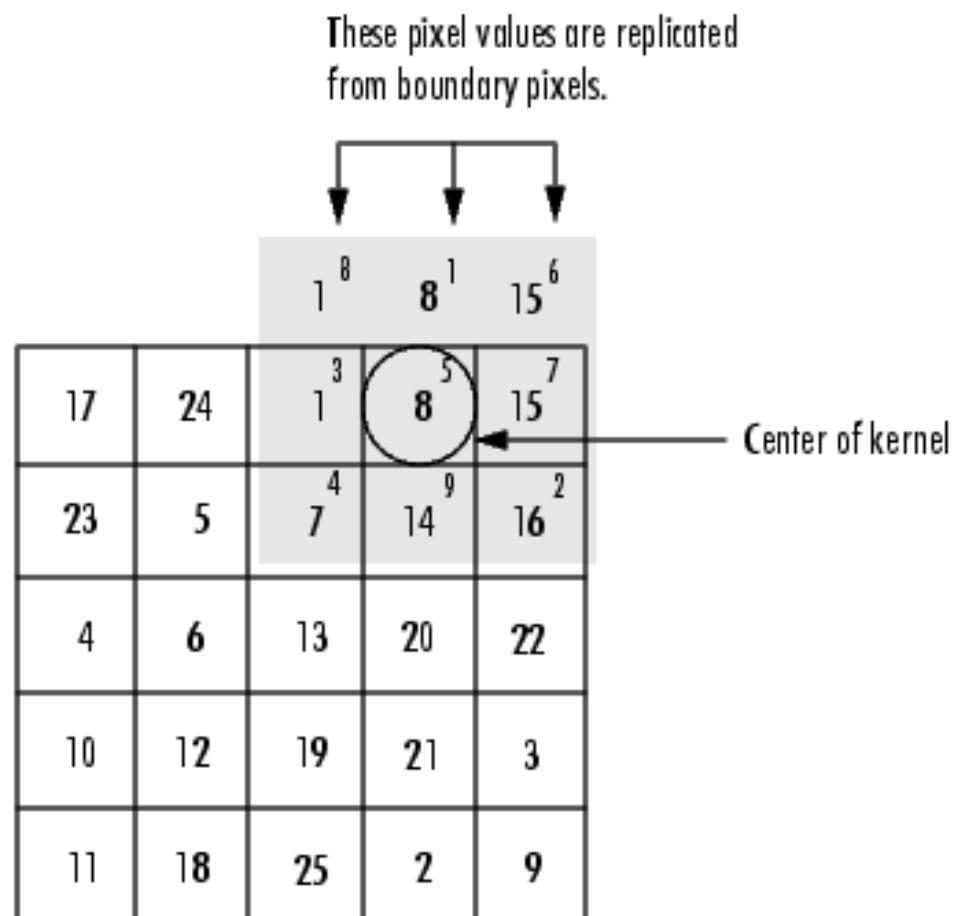


Original Image



Filtered Image with Black Border

border padding



Example on the standard average filter with zero padding

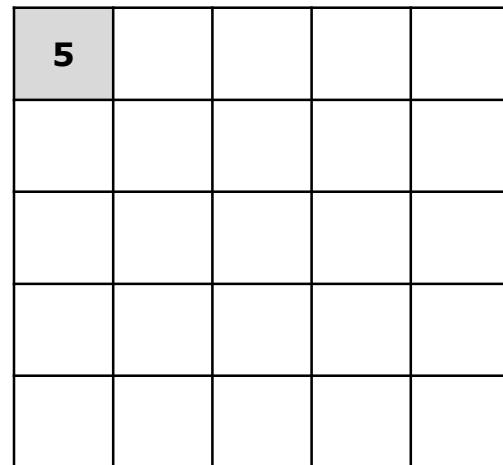
$1/9$	$1/9$	$1/9$			
$1/9$	13	12	10	11	12
$1/9$	10	11	12	10	11
$1/9$	11	13	200	15	14
10	12	13	13	14	
10	11	12	13	11	

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \equiv \begin{array}{|c|c|c|} \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline \end{array}$$

$$13 \cdot 1/9 + 12 \cdot 1/9 + 10 \cdot 1/9 + 11 \cdot 1/9 = 5$$

OR

$$(13*1 + 12*1 + 10*1 + 11*1)/9 = 5$$



Example on the standard average filter with zero padding

1/9	1/9	1/9		
13	12	10	11	12
1/9	1/9	1/9		
10	11	12	10	11
1/9	1/9	1/9		
11	13	200	15	14
10	12	13	13	14
10	11	12	13	11

5	8			

$$13*1/9 + 12*1/9 + 10*1/9 + 10*1/9 + 11*1/9 + 12*1/9 = 8$$

OR

$$(13*1 + 12*1 + 10*1 + 10*1 + 11*1 + 12*1)/9 = 8$$

Example on the standard average filter with zero padding

		1/9	1/9	1/9
13	12	10	11	12
	1/9	1/9	1/9	
10	11	12	10	11
	1/9	1/9	1/9	
11	13	200	15	14
10	12	13	13	14
10	11	12	13	11

5	8	7		

$$12 * 1/9 + 10 * 1/9 + 11 * 1/9 + 11 * 1/9 + 12 * 1/9 + 10 * 1/9 = 7$$

OR

$$(12 * 1 + 10 * 1 + 11 * 1 + 11 * 1 + 12 * 1 + 10 * 1) / 9 = 7$$

Example on the standard average filter with zero padding

		$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
13	12	10	11	12
10	11	12	10	11
11	13	200	15	14
10	12	13	13	14
10	11	12	13	11

5	8	7	7	

$$(10*1 + 11*1 + 12*1 + 12*1 + 10*1 + 11*1)/9 = 7$$

Example on the standard average filter with zero padding

13	12	10	11	12		1/9	1/9	1/9
10	11	12	10	11		1/9	1/9	1/9
11	13	200	15	14				
10	12	13	13	14				
10	11	12	13	11				

5	8	7	7	5

$$(11*1+12*1+10*1+11*1)/9 = 5$$

Example on the standard average filter with zero padding

	13	12	10	11	12
1/9	1/9	1/9			
	10	11	12	10	11
1/9	1/9	1/9			
	11	13	200	15	14
1/9	1/9	1/9			
	10	12	13	13	14
	10	11	12	13	11

5	8	7	7	5
8				

$$(13*1+12*1+10*1+11*1+11*1+11*1+13*1)/9 = 8$$

Example on the standard average filter with zero padding

13	12	10	11	12
1/9	1/9	1/9		
10	11	12	10	11
1/9	1/9	1/9		
11	13	200	15	14
1/9	1/9	1/9		
10	12	13	13	14
10	11	12	13	11

5	8	7	7	5
8	32			

$$(13*1+12*1+10*1+10*1+11*1+12*1+11*1+13*1/9+200*1)/9 = 32$$

Example on the standard average filter with zero padding

13	12	10	11	12
	1/9	1/9	1/9	
10	11	12	10	11
	1/9	1/9	1/9	
11	13	200	15	14
	1/9	1/9	1/9	
10	12	13	13	14
10	11	12	13	11

5	8	7	7	5
8	32	33		

$$(12*1+10*1+11*1+11*1+12*1+10*1+13*1+200*1/9+15*1)/9 = 33$$

Example on the standard average filter with zero padding

13	12	10	11	12
		1/9	1/9	1/9
10	11	12	10	11
		1/9	1/9	1/9
11	13	200	15	14
		1/9	1/9	1/9
10	12	13	13	14
10	11	12	13	11

5	8	7	7	5
8	32	33	33	

$$(10*1+11*1+12*1+12*1+10*1+11*1+200*1+15*1/9+14*1)/9 = 33$$

Example on the standard average filter with zero padding

13	12	10	11	12	
			1/9	1/9	1/9
10	11	12	10	11	
			1/9	1/9	1/9
11	13	200	15	14	
			1/9	1/9	1/9
10	12	13	13	14	
10	11	12	13	11	

5	8	7	7	5
8	32	33	33	8

$$(11*1+12*1+10*1+11*1+15*1+14*1)/9 = 8$$

Example on the standard average filter with zero padding

You have to apply same process till to the end of the image

13	12	10	11	12
10	11	12	10	11
11	13	200	15	14
10	12	13	13	14
10	11	12	13	11

$\begin{matrix} & & \\ 1/9 & & 1/9 & & 1/9 \\ & & & & \\ \end{matrix}$

$\begin{matrix} & & \\ 1/9 & & 1/9 & & 1/9 \\ & & & & \\ \end{matrix}$

$\begin{matrix} & & \\ 1/9 & & 1/9 & & 1/9 \\ & & & & \\ \end{matrix}$

$$(13*1+14*1+13*1+11*1)/9 = 6$$

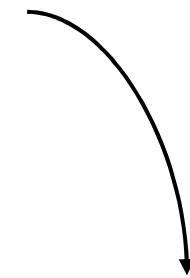


Original Image



Filtered Image with Black Border

5	8	7	7	5
8	32	33	33	8
7	32	33	34	9
7	32	34	34	9
5	8	8	8	6



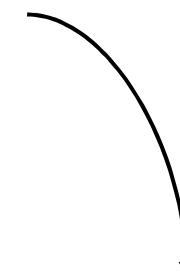
As you can see, we successfully remove the noise (200) and the gray values are very close to each other except the pixels on boundary it have low gray value because of zero padding and this will produce black border and we can use replicate border to eliminate black border.

Example on the standard average filter with replicate border

Spatial filtering with replicate border

13	12	10	11	12
10	11	12	10	11
11	13	200	15	14
10	12	13	13	14
10	11	12	13	11

12	11	11	11	11
12	32	33	33	12
11	32	33	34	13
11	32	34	34	13
10	11	12	12	12



As you can see, the gray values of boundary's pixel are not affected too much with averaging filter

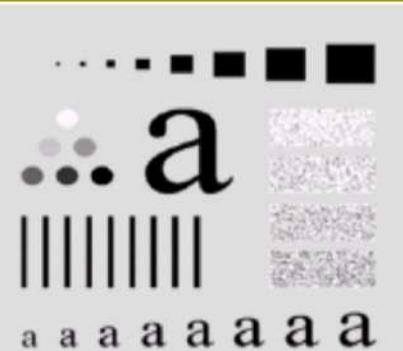
Spatial filters : Smoothing

Averaging effects: blurring + reducing noise

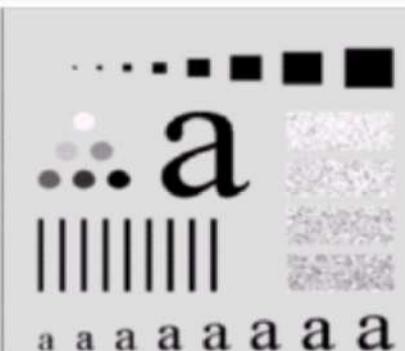
Original image



3 x 3 averaging



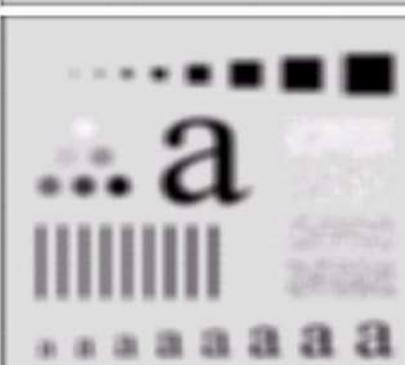
5 x 5 averaging



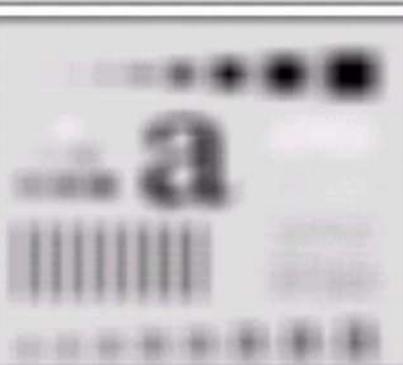
9 x 9 averaging



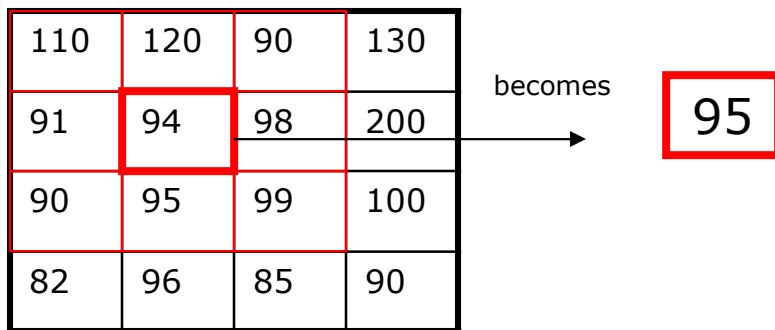
15 x 15 averaging



35 x 35 averaging



Spatial filters : Smoothing order statistics: Median filter



Steps:

1. Sort the pixels in ascending order:

90, 90, 91, 94, 95, 96, 98, 99, 110, 120

2. replace the original pixel value by the median :

95

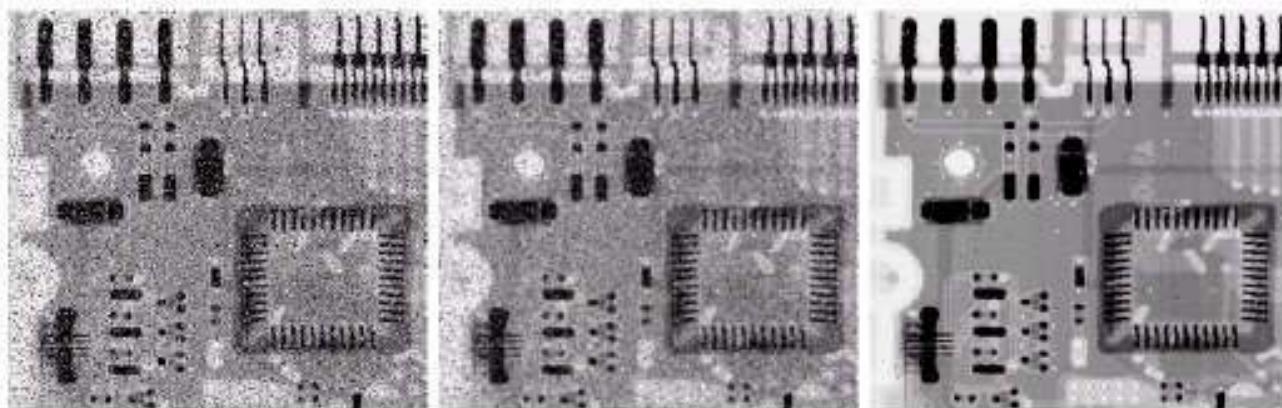
the **median** filter is based on ordering (**ranking**) the pixels , then *replacing the value of a pixel by the median of the gray levels in the neighborhood of that pixel.*

Median filters are quite **popular** because, for certain types of random noise, they provide **excellent noise reduction + less blurring** than linear smoothing filters of similar size.

Median filters are particularly effective in the presence of *impulse noise, also called salt-and-pepper noise* because of its appearance as white and black dots superimposed on an image.

Spatial filters : Smoothing
order statistics: Median filter
use : blurring + reduce salt and pepper noise

Which one has removed the
salt-and-pepper noise??



The original
image with salt
and pepper noise

The smoothed
image using
averaging

The smoothed image
using median

a
b

Example on a Median Filter with replicate border

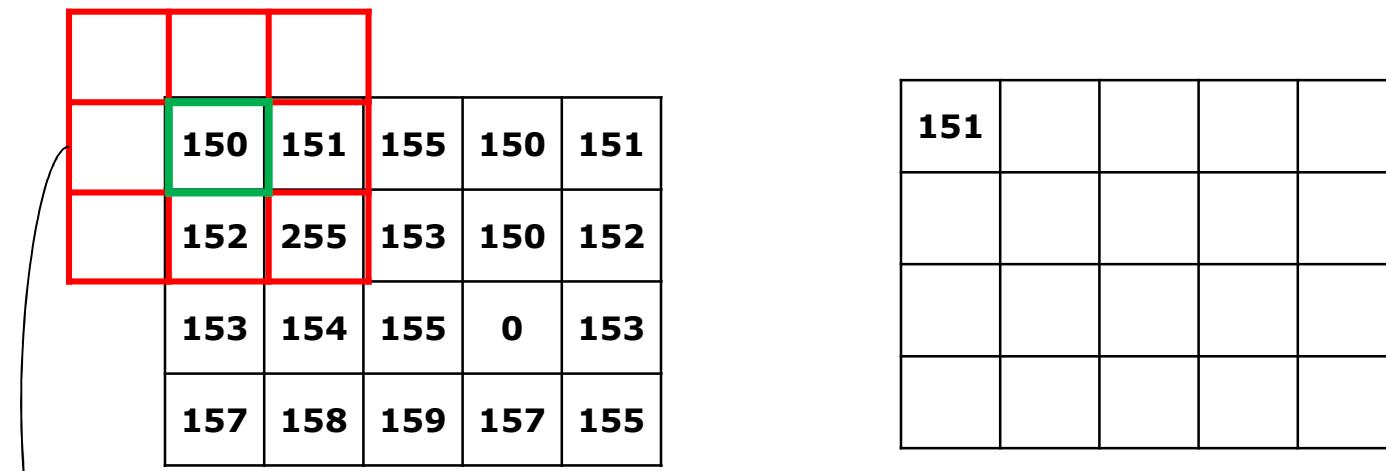
Apply 3×3 median filter on the following image.

Salt noise

Pepper noise

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

Example on a Median Filter with replicate border



With replicate border

{ 150, 150, 151, 150, 150, 151, 151, 152, 152, 255}

After sorting:

{ 150, 150, 150, 150, 151, 151, 152, 152, 255}

The median value is 151

Example on a Median Filter with replicate border

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

151	152			

With replicate border

{150, 151, 155, 150, 151, 155, 152, 255, 153}

After sorting:

{ 150, 150, 151, 151, 152, 153, 155, 155, 255}

The median value is 152

Example on a Median Filter with replicate border

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

151	152	151		

With replicate border

{151, 155, 150, 151, 155, 150, 255, 153, 150}

After sorting:

{ 150, 150, 150, 151, 151, 153, 155, 155, 255}

The median value is 151

Example on a Median Filter with replicate border

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

151	152	151	151	

With replicate border

{155, 150, 151, 155, 150, 151, 153, 150, 152}

After sorting:

{150, 150, 150, 151, 151, 152, 153, 155, 155}

The median value is 151

Example on a Median Filter with replicate border

150	151	155	150	151	
152	255	153	150	152	
153	154	155	0	153	
157	158	159	157	155	

With a 3x3 kernel, the central element (150) is highlighted with a red box. The element at index (3,3) (151) is highlighted with a green box.

151	152	151	151	151

With replicate border

{150, 151, 151, 150, 151, 151, 150, 152, 152}

After sorting:

{150, 150, 150, 151, 151, 151, 151, 152, 152}

The median value is 151

Example on a Median Filter with replicate border

	150	151	155	150	151
	152	255	153	150	152
	153	154	155	0	153
	157	158	159	157	155

151	152	151	151	151
152				

With replicate border

{150, 150, 151, 152, 152, 255, 153, 153, 154}

After sorting:

{150, 150, 151, 152, 152, 152, 153, 153, 154, 255}

The median value is 152

Example on a Median Filter with replicate border

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

151	152	151	151	151
152	153			

With replicate border

{150, 151, 155, 152, 255, 153, 153, 154, 155}

After sorting:

{150, 151, 152, 153, 153, 154, 155, 155, 255}

The median value is 153

Example on a Median Filter with replicate border

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

151	152	151	151	151
152	153	153		

With replicate border

{151, 155, 150, 255, 153, 150, 154, 155, 0}

After sorting:

{0, 150, 150, 151, 153, 154, 155, 155, 255}

The median value is 153

Example on a Median Filter with replicate border

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

151	152	151	151	151
152	153	153	152	

With replicate border

{155, 150, 151, 153, 150, 152, 155, 0, 153}

After sorting:

{0, 150, 150, 151, 152, 153, 153, 155, 155}

The median value is 152

Example on a Median Filter with replicate border

150	151	155	150	151	
152	255	153	150	152	
153	154	155	0	153	
157	158	159	157	155	

151	152	151	151	151
152	153	153	152	151

With replicate border

{150, 151, 151, 150, 152, 152, 0, 153, 153}

After sorting:

{0, 150, 150, 151, 151, 152, 152, 153, 153}

The median value is 151

Example on a Median Filter with replicate border

150	151	155	150	151
152	255	153	150	152
153	154	155	0	153
157	158	159	157	155

**After applying
3x3 Median filter**

151	152	151	151	151
152	153	153	152	151
154	155	155	153	153
157	157	157	155	155

- As we note, the salt and pepper noise is removed completely from this image without burring effect

Spatial filters : Sharpening (high pass)

3.6 Sharpening Spatial Filters

- The principal **objective** of sharpening is to highlight **transition** in intensity.
- Sharpening is accomplished by spatial **differentiation**.
- Image differentiation **enhances edges** and other discontinuities (such as noise) and deemphasizes areas with slowly varying gray-level values

Spatial filters : Sharpening (high pass)

Example on a Sharpening:

f

50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

$\nabla^2 f$

-10	-20	20	10
-10	-20	20	10
-10	-20	20	10
-10	-20	20	10

$g = f + \nabla^2 f$

40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

3.6.1 Foundation

- The **derivatives** of a digital function are defined in terms of **differences**
- Definitions of the **first and 2nd-order** derivatives of a 1-D function $f(x)$ are the differences:

f	50	50	60	90	100	100	100
-----	----	----	----	----	-----	-----	-----

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

$\frac{\partial f}{\partial x}$	0	10	30	10	0	0
---------------------------------	---	----	----	----	---	---

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

$\frac{\partial^2 f}{\partial x^2}$		10	20	-20	-10	0
-------------------------------------	--	----	----	-----	-----	---

3.6.2 Using the Second Derivative for Image Sharpening—The Laplacian

- The Laplacian, for a function (image) $f(x, y)$ of two variables, is defined and given below:

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

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

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

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

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

Ex: apply the second derivative
on the shaded pixel

20	20	20	20
20	10	10	10
20	10	10	10
20	10	10	10

→ $\nabla^2 f = 20 + 20 + 10 + 10 - 4 * 10$
 $\nabla^2 f = 20$

→ $\nabla^2 f = 10 + 10 + 10 + 10 - 4 * 10$
 $\nabla^2 f = 0$

Spatial filters : Sharpening (high pass)

1.LAPLACE

2.Unsharp

3.High boost

4.sobel

2nd Derivative for Image Sharpening—The Laplacian

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

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

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

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

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

2nd Derivative for Image Sharpening—The Laplacian

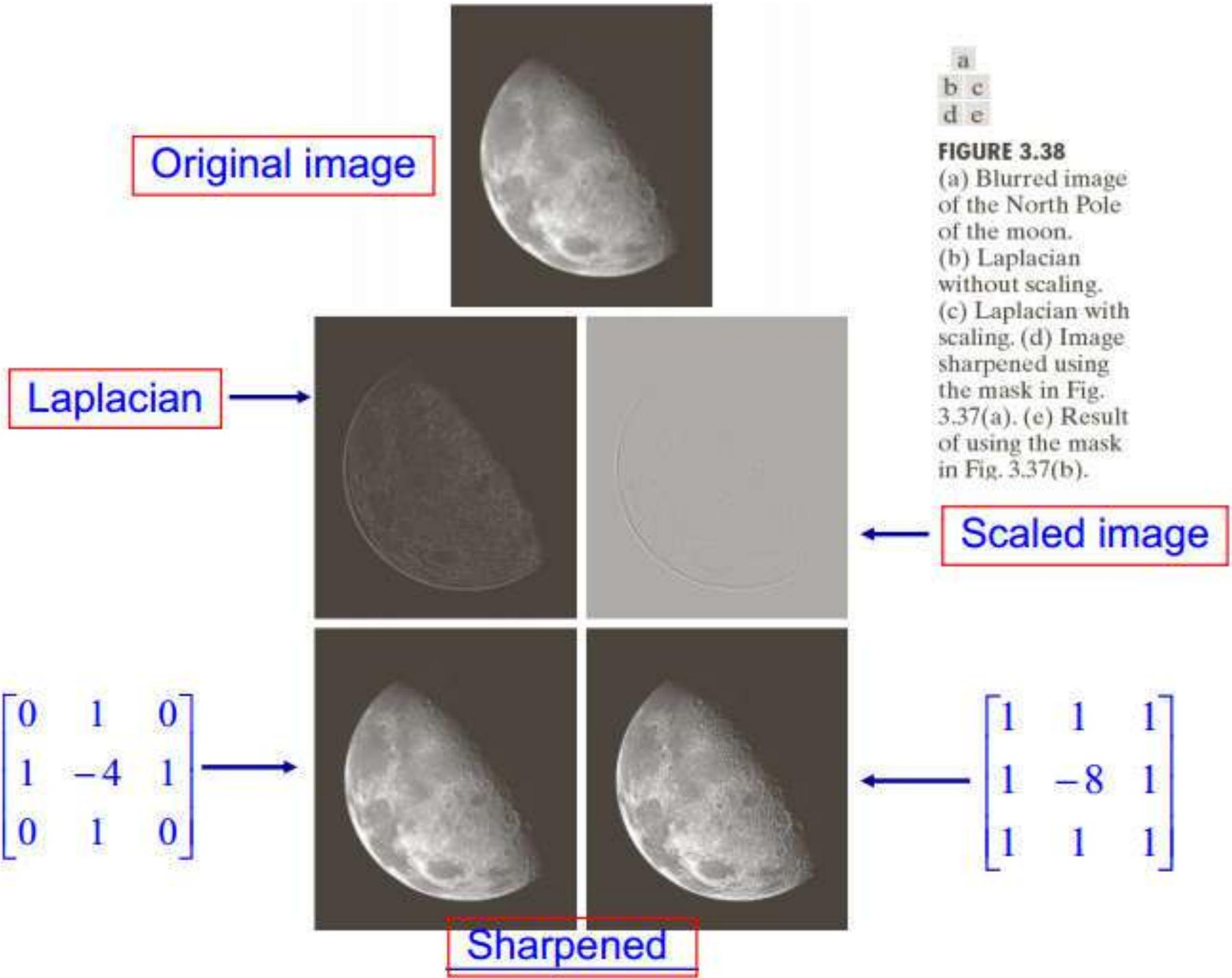
$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad h_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{or} \quad h_4 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Spatial filters : Sharpening

1) LAPLACE (2nd derivative)

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



a
b c
d e

FIGURE 3.38

(a) Blurred image of the North Pole of the moon.
 (b) Laplacian without scaling.
 (c) Laplacian with scaling.
 (d) Image sharpened using the mask in Fig. 3.37(a).
 (e) Result of using the mask in Fig. 3.37(b).

The Laplacian (Better Implementation)

In practice we use the Laplacian for Image enhancement
In a better way by increasing (decreasing)
the center coefficient by 1; e.g.

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

Composite Laplacian Filter

$g(x, y) = f(x, y) + \nabla^2 f(x, y)$ if the center coefficient of the Laplacian mask is positive.

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

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

$$g(x, y) = f(x, y) + 4f(x, y) - f(x+1, y) - f(x-1, y) - f(x, y+1) - f(x, y-1)$$

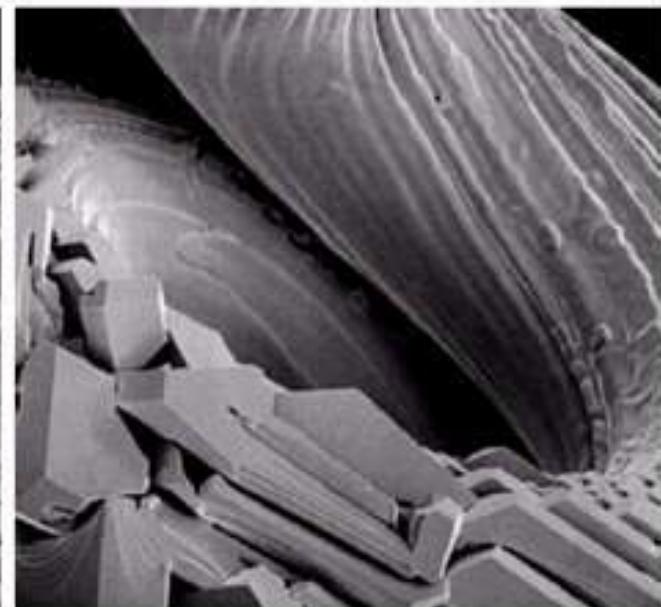
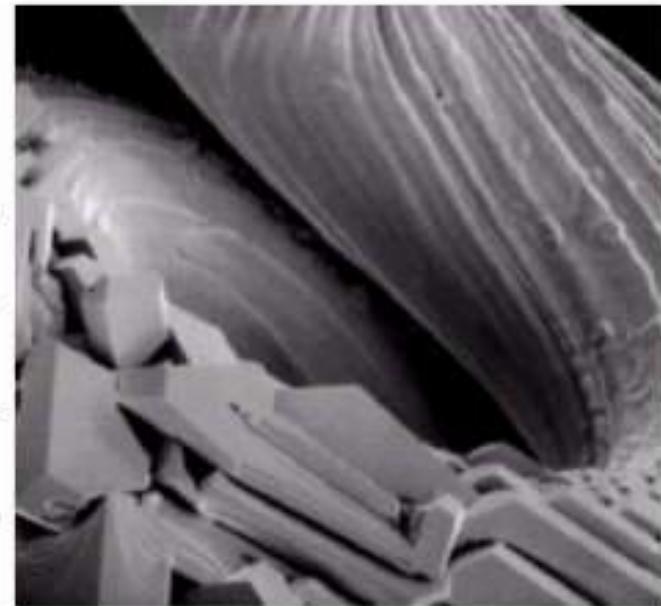
$$g(x, y) = 5f(x, y) - f(x+1, y) - f(x-1, y) - f(x, y+1) - f(x, y-1)$$

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

The Laplacian (more practical masks)

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

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



a b c
d e

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

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

f

50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

Laplacian

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

$\nabla^2 f$

-10	-20	20	10
-10	-20	20	10
-10	-20	20	10
-10	-20	20	10

$g = f + \nabla^2 f$

40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

Composite
Laplacian

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

40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

0	-1	0		
-1	50	60	90	100
4				
0	50	60	90	100
	50	60	90	100
50	60	90	100	

$\nabla^2 f$			
-10			

$g = f + \nabla^2 f$			
40			

$$\begin{aligned}\nabla^2 f &= 4 * 50 - 50 - 50 - 60 - 50 \\&= 200 - 210 \\&= -10\end{aligned}$$

$$\begin{aligned}g(x, y) &= 50 - 10 \\g(x, y) &= 40\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

0	-1	0	
50 -1	60 4	90 -1	100
50 0	60 -1	90 0	100
50	60	90	100
50	60	90	100

$\nabla^2 f$			
-10	-20		

$g = f + \nabla^2 f$			
40	40		

$$\begin{aligned}\nabla^2 f &= 4 * 60 - 50 - 60 - 60 - 90 \\&= 240 - 260 \\&= -20\end{aligned}$$

$$\begin{aligned}g(x, y) &= 60 - 20 \\g(x, y) &= 40\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

		0	-1	0
50	60	90	100	
50	60	90	100	
50	60	90	100	
50	60	90	100	

$$\nabla^2 f$$

-10	-20	20	

$$g = f + \nabla^2 f$$

40	40	110	

$$\begin{aligned}\nabla^2 f &= 4 * 90 - 90 - 90 - 60 - 100 \\&= 360 - 340 \\&= 20\end{aligned}$$

$$\begin{aligned}g(x, y) &= 90 + 20 \\g(x, y) &= 110\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

50	60	90 -1	100 4
50	60	90 0	100 -1
50	60	90	100
50	60	90	100

$\nabla^2 f$			
-10	-20	20	10

$g = f + \nabla^2 f$			
40	40	110	110

$$\begin{aligned}\nabla^2 f &= 4 * 100 - 100 - 100 - 100 - 90 \\ &= 400 - 390 \\ &= 10\end{aligned}$$

$$\begin{aligned}g(x, y) &= 100 + 10 \\ g(x, y) &= 110\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

0	50 -1	60 0	90	100
-1	50 4	60 -1	90	100
0	50 -1	60 0	90	100
	50	60	90	100

$\nabla^2 f$

-10	-20	20	10
-10			

$g = f + \nabla^2 f$

40	40	110	110
40			

$$\begin{aligned}\nabla^2 f &= 4 * 50 - 50 - 50 - 50 - 60 \\&= 200 - 210 \\&= -10\end{aligned}$$

$$\begin{aligned}g(x, y) &= 50 - 10 \\g(x, y) &= 40\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

50 0	60 -1	90 0	100
50 -1	60 4	90 -1	100
50 0	60 -1	90 0	100
50	60	90	100

$\nabla^2 f$

-10	-20	20	10
-10	-20		

$g = f + \nabla^2 f$

40	40	110	110
40	40		

$$\begin{aligned}\nabla^2 f &= 4 * 60 - 50 - 60 - 60 - 90 \\&= 240 - 260 \\&= -20\end{aligned}$$

$$\begin{aligned}g(x, y) &= 60 - 20 \\g(x, y) &= 40\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

50	60 0	90 -1	100 0
50	60 -1	90 4	100 -1
50	60 0	90 -1	100 0
50	60	90	100

$\nabla^2 f$

-10	-20	20	10
-10	-20	20	

$g = f + \nabla^2 f$

40	40	110	110
40	40	110	

$$\begin{aligned}\nabla^2 f &= 4 * 90 - 90 - 90 - 60 - 100 \\ &= 360 - 340 \\ &= 20\end{aligned}$$

$$\begin{aligned}g(x, y) &= 90 + 20 \\ g(x, y) &= 110\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

50	60	90	100	0	0
50	60	90	100	-1	0
50	60	90	100	4	-1
50	60	90	100	0	-1

$\nabla^2 f$

-10	-20	20	10
-10	-20	20	10

$g = f + \nabla^2 f$

40	40	110	110
40	40	110	110

$$\begin{aligned}\nabla^2 f &= 4 * 100 - 100 - 100 - 100 - 90 \\ &= 400 - 390 \\ &= 10\end{aligned}$$

$$\begin{aligned}g(x, y) &= 100 + 10 \\ g(x, y) &= 110\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

0 -1
-1 4
0 -1

-10	-20	20	10
-10	-20	20	10
-10	-20	20	10
-10	-20	20	10
-10	-20	20	10

40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

$$\begin{aligned}\nabla^2 f &= 4 * 100 - 100 - 100 - 100 - 90 \\&= 400 - 390 \\&= 10\end{aligned}$$

$$\begin{aligned}g(x, y) &= 100 + 10 \\g(x, y) &= 110\end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

0	-1	0
-1	50 5	60 -1
0	50	60
50	60	90
50	60	90

g

40			

$$\begin{aligned} g(x, y) &= 5 * 50 - 50 - 50 - 60 - 50 \\ &= 250 - 210 \\ &= 40 \end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

0	-1	0	
50 -1	60 5	90 -1	100
50 0	60 -1	90 0	100
50	60	90	100
50	60	90	100

g

40	40		

$$\begin{aligned} g(x, y) &= 5 * 60 - 50 - 60 - 60 - 90 \\ &= 3000 - 260 \\ &= 40 \end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

				0	-1	0
50	60	90	100	-1	5	-1
50	60	90	100	0	-1	0
50	60	90	100			
50	60	90	100			

g

40	40	110	

$$\begin{aligned} g(x, y) &= 5 * 90 - 90 - 90 - 60 - 100 \\ &= 450 - 340 \\ &= 110 \end{aligned}$$

Laplacian vs. Composite Laplacian

Example: Apply the Laplacian and composite Laplacian filters on the following blurred edge

50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

A 5x5 input image grid with values: Row 1: 50, 60, 90, 100, 0; Row 2: 50, 60, 90, 100, 0; Row 3: 50, 60, 90, 100, 0; Row 4: 50, 60, 90, 100, 0; Row 5: 50, 60, 90, 100, 0. The value 100 is highlighted in red.

A 3x3 kernel is applied to the center pixel (90). The kernel is: $\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$. The central value 5 is highlighted in green.

<i>g</i>			
40	40	110	110
40	40	110	110
40	40	110	110
40	40	110	110

$$\begin{aligned} g(x, y) &= 5 * 100 - 100 - 100 - 100 - 90 \\ &= 500 - 390 \\ &= 110 \end{aligned}$$

Spatial filters : Sharpening LAPLACE – 2st derivative

Use: for highlighting fine detail or enhancing detail that has been blurred.

Example: apply the following laplace on the highlighted pixel

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

153	157	156	153	155
159	156	158	156	159
155	158	154	156	160
154	157	158	160	160
157	157	157	156	155

$$154*4 - 158 - 156 - 158 - 158 = -14$$

So the value after filter = -14

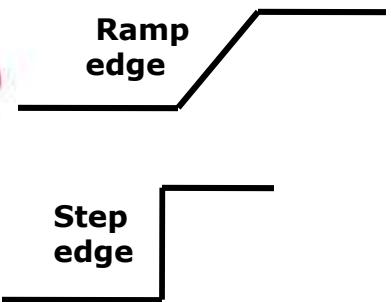
We call the resultant image: **sharpened image**.

Filtered image=original + sharpened image

The value in the filter image=154-14 =140

Sharpening Spatial filters

- Used for **highlighting fine detail or enhancing detail that has been blurred.**
- Averaging is analogous to **integration**.
- Sharpening is analogous to **differentiation**.
- A Basic defintion of the first-order derivative of 1-D function $f(x)$ is the difference:
 - $f'(x) = f(x + 1) - f(x)$
 - Must be zero in flat areas
 - Must be non-zero at the onset of a gray-level step or ramp
 - Must be nonzero along ramps
- Second derivatives
 - $f''(x) = f(x + 1) + f(x - 1) - 2f(x)$
 - Must be zero in flat areas
 - Must be non-zero at the onset and end of a gray-level step or ramp
 - Must be zero along ramps of constant slope



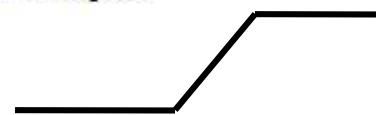
- Used for **highlighting fine detail** or **enhancing detail that has been blurred**.
- Averaging is analogous to **integration**.
- Sharpening is analogous to **differentiation**.
- A **Basic definition of the first-order derivative of 1-D function $f(x)$** is the difference:
 - $f'(x) = f(x + 1) - f(x)$
 - Must be zero in flat areas
 - Must be non-zero at the onset of a gray-level step or ramp
 - Must be nonzero along ramps

Second derivatives

- $f''(x) = f(x + 1) + f(x - 1) - 2f(x)$
- Must be zero in flat areas
- Must be non-zero at the onset and end of a gray-level step or ramp
- Must be zero along ramps of constant slope

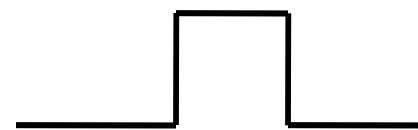
Ramp edge

5	5	5	6	7	8	9	9	9
---	---	---	---	---	---	---	---	---



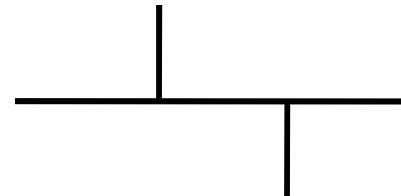
1st derivative

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



2nd derivative

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



Spatial filters : Sharpening

1st VS 2nd derivative sharpening

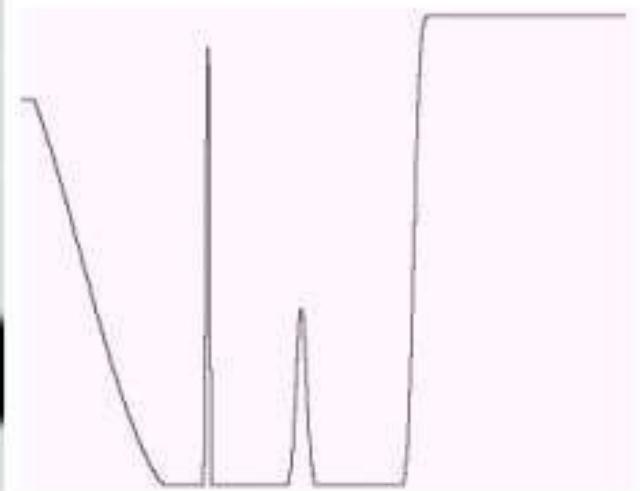
1st derivative sharpening produces thicker edges in an image

1st derivative sharpening has stronger response to gray level change

2nd derivative sharpening has stronger response to fine details, such as thin lines and isolated points.

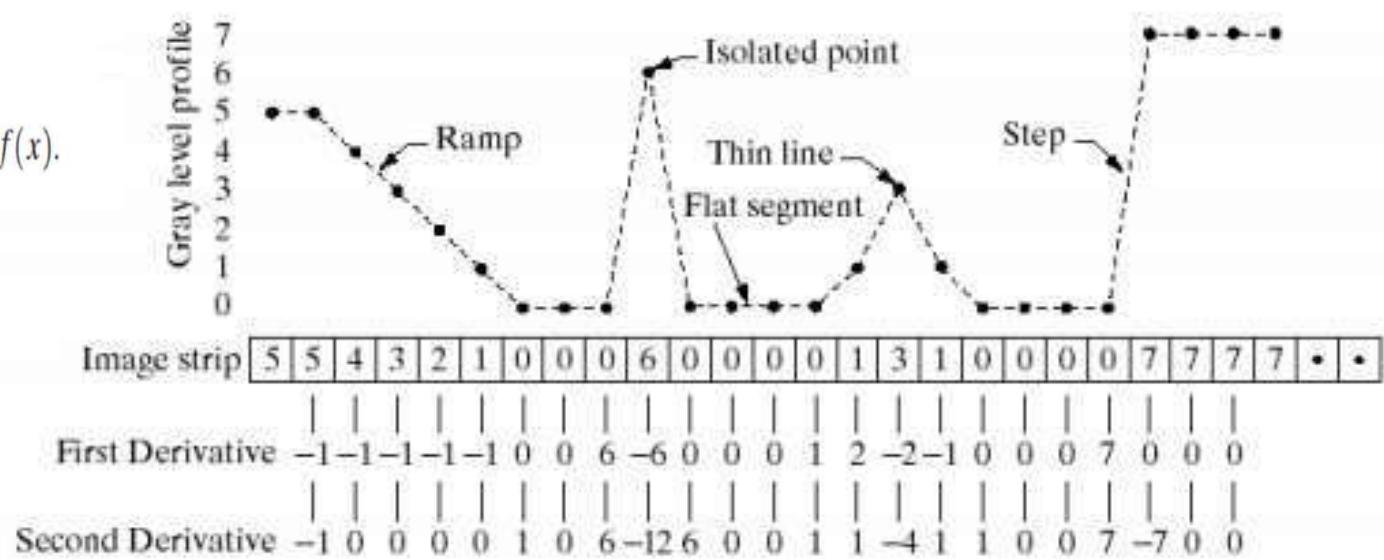
2nd derivative sharpening has double response to gray level change

1st vs 2nd derivative



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

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



Unsharp Masking

- ❑ This operation is referred to as edge enhancement, edge crispening or *unsharp masking*.
- ❑ To make edges slightly sharper and crisper.
- ❑ A very popular practice in industry (where the name came).
- ❑ Subtracting a blurred version of an image from the image itself.

$$f_s(x, y) = f(x, y) - \bar{f}(x, y)$$

↑ ↑
Original image Blurred image

Unsharp Masking

To sharpen images by **subtracting** a blurred or **unsharp (smoothed)** version of an image from the original image itself.

- This process, called unsharp masking, **consists of**:
 1. Blur the original image.
Let $\bar{f}(x, y)$ denote the **blurred** image.
 2. Subtract the blurred image from the original image (the resulting difference is called the mask).

$$g_{mask}(x, y) = f(x, y) - \bar{f}(x, y)$$

3. Add the mask to the original.

$$g(x, y) = f(x, y) + g_{mask}(x, y)$$

Unsharp masking and Highboost filtering expressions:

- Let $\bar{f}(x, y)$ denote the blurred image.
- The mask is obtained as:

$$g_{mask}(x, y) = f(x, y) - \bar{f}(x, y)$$

- A weighted portion of mask is added to original:

$$g(x, y) = f(x, y) + k * g_{mask}(x, y)$$

- **Unsharp Making:** when $k=1$
- **Highboost filtering:** when $k > 1$



FIGURE 3.40

- (a) Original image.
- (b) Result of blurring with a Gaussian filter.
- (c) Unsharp mask.
- (d) Result of using unsharp masking.
- (e) Result of using highboost filtering.

Example

Example: Apply Unsharp filter on the following blurred edge

f			
50	60	90	100
50	60	90	100
50	60	90	100
50	60	90	100

Std. avg		
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

53	67	83	97
53	67	83	97
53	67	83	97
53	67	83	97

$mask = f - f'$			
-3	-7	7	3
-3	-7	7	3
-3	-7	7	3
-3	-7	7	3

$$g_{Unsharp} = f + mask$$

47	53	97	103
47	53	97	103
47	53	97	103
47	53	97	103

As we note, the difference between the resulted edge (53,97) is larger than the original edge (60,90), but if this sharpening is not enough we will try to enlarge the value of the mask using high-boost filter by multiplying with value >1

$$g_{highboost} = f + 3 * mask$$

41	39	111	109
41	39	111	109
41	39	111	109
41	39	111	109

1st Derivatives of Enhancement – The Gradient

First derivatives in image processing are implemented using the magnitude of the gradient.

For a function $f(x, y)$, the gradient of f at coordinates (x, y) is defined as the 2-dimensional column vector:

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

$\begin{bmatrix} G_x \\ G_y \end{bmatrix}$			
$\begin{bmatrix} G_x \\ G_y \end{bmatrix}$			
$\begin{bmatrix} G_x \\ G_y \end{bmatrix}$			
$\begin{bmatrix} G_x \\ G_y \end{bmatrix}$			

1st Derivatives of Enhancement – The Gradient

The magnitude of the vector is given by:

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

The magnitude of the gradient vector is often referred to As the *gradient*.

Because of the computational burden of the above Eq., it is common Practice to approximate the magnitude of the gradient using absolute values as :

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

This Eq. is simpler to compute.

1st Derivatives of Enhancement – The Gradient

We can now define digital approximations to the preceding equations, and from there formulate the appropriate filter masks.

We will use the following notation to denote image points in a 3x3 region:

...

1st Derivatives of Enhancement – The Gradient

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

**The center point, z_5 , denotes $f(x, y)$,
 z_1 denotes $f(x-1, y-1)$, and so on.**

1st Derivatives of Enhancement – The Gradient

a
b c
d e

FIGURE 3.44
A 3×3 region of an image (the z 's are gray-level values) and masks used to compute the gradient at point labeled z_5 . All masks coefficients sum to zero, as expected of a derivative operator.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

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

-1	0	0	-1
0	1	1	0

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

Roberts operators

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



The weight value of 2 is used to achieve some smoothing by giving more importance to the center point.

Sobel operators

$$\nabla f \approx |(z_7+2z_8+z_9)-(z_1+2z_2+z_3)| + |(z_3+2z_6+z_9)-(z_1+2z_4+z_7)|$$

Robert's Operators

Example: Apply the Robert's operators on the pixel Z_5

Z_1	Z_2	Z_3	
Z_4	Z_5 -1	Z_6 0	
Z_7	Z_8 0	Z_9 1	

Z_1	Z_2	Z_3	
Z_4	Z_5 0	Z_6 -1	
Z_7	Z_8 1	Z_9 0	

$$G_x = Z_5 * -1 + Z_6 * 0 + Z_8 * 0 + Z_9 * 1$$

$$G_y = Z_5 * 0 + Z_6 * -1 + Z_8 * 1 + Z_9 * 0$$

$$G_x = Z_9 - Z_5$$

$$G_y = Z_8 - Z_6$$

$$\text{Gradient} = |Z_9 - Z_5| + |Z_8 - Z_6|$$

Sobel Operators

Example: Apply the Sobel operators on the pixel Z_5

Z_1	Z_2	Z_3	
-1	-2	-1	
Z_4	Z_5	Z_6	
0	0	0	
Z_7	Z_8	Z_9	
1	2	1	

Z_1	Z_2	Z_3	
-1	0	1	
Z_4	Z_5	Z_6	
-2	0	2	
Z_7	Z_8	Z_9	
-1	0	1	

$$G_x = Z_1 * -1 + Z_2 * -2 + Z_3 * -1 + \\ Z_4 * 0 + Z_5 * 0 + Z_6 * 0 + Z_7 * 1 + Z_8 \\ * 2 + Z_9 * 1$$

$$G_x = -Z_1 - 2Z_2 - Z_3 + Z_7 + 2Z_8 + Z_9$$

$$G_x = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)$$

$$G_x = Z_1 * -1 + Z_4 * -2 + Z_7 * -1 + \\ Z_2 * 0 + Z_5 * 0 + Z_8 * 0 + Z_3 * 1 + Z_6 \\ * 2 + Z_9 * 1$$

$$G_x = -Z_1 - 2Z_4 - Z_7 + Z_3 + 2Z_6 + Z_9$$

$$G_x = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)$$

$$\text{Gradient} = |(Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)| + |(Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)|$$

Sobel Operators

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

Detects horizontal edges

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

Detects vertical edges

Sobel operators

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

we can apply the sobel horizontal kernel or the sobel vertical kernel or both and adding them together.

Example: Find the gradient of the following shaded pixels:

	P1	P2	P4			
	10	10	10	200	200	200
P1	10	10	10	200	200	200
P3	10	10	10	200	200	200
	100	100	100	20	20	20
	100	100	100	20	20	20
	100	100	100	20	20	20

Gx	Gy
-1	-1
0	0
1	2
-2	0
-1	1

$$Gx(P1) = -10 - 20 - 10 + 0 + 0 + 0 + 10 + 20 + 10 \\ = 0$$

10	-1	10	-2	10	-1	200	200	200
10	0	10	0	10	0	200	200	200
10	1	10	2	10	1	200	200	200
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20

10	-1	10	0	10	1	200	200	200
10	-2	10	0	10	2	200	200	200
10	-1	10	0	10	1	200	200	200
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20

$$Gy(P1) = -10 + 0 + 10 - 20 + 0 + 20 - 10 + 0 + 10 \\ = 0$$

$$\nabla(P1) = |Gx| + |Gy|$$

$$\nabla(P1) = 0$$

The gradient is zero because P1 is located within a constant area

Example: Find the gradient of the following shaded pixels:

Gx(P2)

10	10	-1	10	-2	200	-1	200	200
10	10	0	10	0	200	0	200	200
10	10	1	10	2	200	1	200	200
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20

Gy(P2)

10	10	-1	10	0	200	1	200	200
10	10	-2	10	0	200	2	200	200
10	10	-1	10	0	200	1	200	200
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20
100	100	100	20	20	20	20	20	20

$$Gx(p2) = -10 - 20 - 200 + 0 + 0 + 0 + 10 + 20 + 200 = 0$$

$$Gy(p2) = -10 + 0 + 200 - 20 + 0 + 400 - 10 + 0 + 200 = 760$$

$$\nabla(P2) = |Gx| + |Gy|$$

$$\nabla(P2) = 0 + 760 = 760$$

The Gy is non-zero because P2 is located on a vertical edge

Example: Find the gradient of the following shaded pixels:

Gx(P3)

10	10	10	200	200	200
10 -1	10 -2	10 -1	200	200	200
10 0	10 0	10 0	200	200	200
100 1	100 2	100 1	20	20	20
100	100	100	20	20	20
100	100	100	20	20	20

Gy(P3)

10	10	10	200	200	200
10 -1	10 0	10 1	200	200	200
10 -2	10 0	10 2	200	200	200
100 -1	100 0	100 1	20	20	20
100	100	100	20	20	20
100	100	100	20	20	20

$$Gx(P3) = -10 - 20 - 10 + 0 + 0 + 0 + 100 + 200 + 100 = 360$$

$$Gy(P3) = -10 + 0 + 10 - 20 + 0 + 20 - 100 + 0 + 100 = 0$$

$$\begin{aligned} \nabla(P3) &= |Gx| + |Gy| \\ \nabla(P3) &= 0 + 360 = 360 \end{aligned}$$

The Gx here is non-zero because P3 is located on a horizontal edge

Example: Find the gradient of the following shaded pixels:

Gx(P4)

10	10	10	200	200	200
10	10 -1	10 -2	200 -1	200	200
10	10 0	10 0	200 0	200	200
100	100 1	100 2	20 1	20	20
100	100	100	20	20	20
100	100	100	20	20	20

Gy(P4)

10	10	10	200	200	200
10	10 -1	10 0	200 1	200	200
10	10 -2	10 0	200 2	200	200
100	100 -1	100 0	20 1	20	20
100	100	100	20	20	20
100	100	100	20	20	20

$$Gx(P4) = -10 - 20 - 200 + 0 + 0 + 0 + 100 + 200 + 20 = 90$$

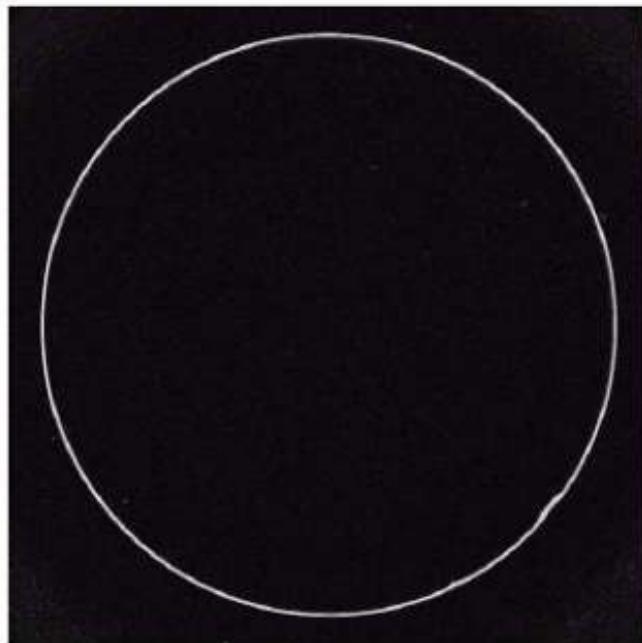
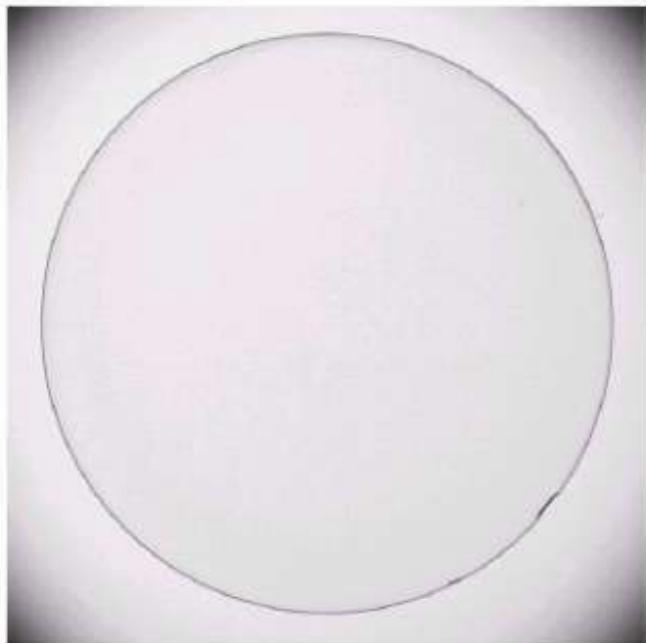
$$Gy(P4) = -10 + 0 + 200 - 20 + 0 + 400 - 100 + 0 + 20 = 490$$

$$\begin{aligned} \nabla(P4) &= |Gx| + |Gy| \\ \nabla(P4) &= 90 + 490 = 580 \end{aligned}$$

The Gx and Gy here are non-zero because P4 is located on a horizontal and vertical edges

1st Derivatives of Enhancement – The Gradient

Gradient is frequently used in industrial inspection.



a b

FIGURE 3.45
Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock).
(b) Sobel gradient.
(Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)



Constant or slowly varying shades of gray have been eliminated.

Combining spatial Enhancement Methods

In this chapter we have focused attention thus far on individual enhancement approaches.

Frequently, a given enhancement task will require application of *several complementary enhancement techniques* in order to achieve an acceptable result.

In this section we illustrate by means of an example how to combine several of the approaches developed in this chapter to address a difficult enhancement task.

Combining spatial Enhancement Methods

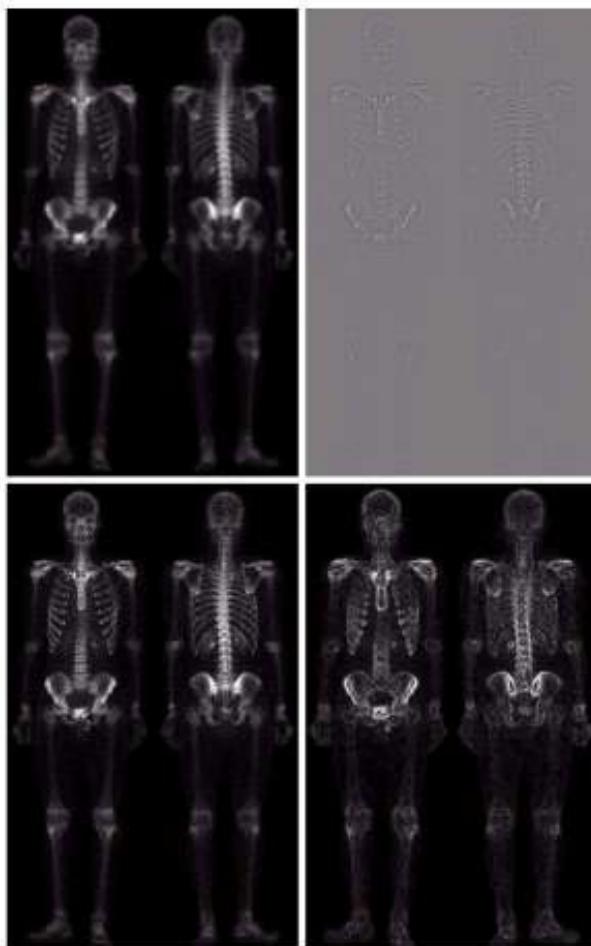


FIGURE 3.46
(a) Image of whole body bone scan.
(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).

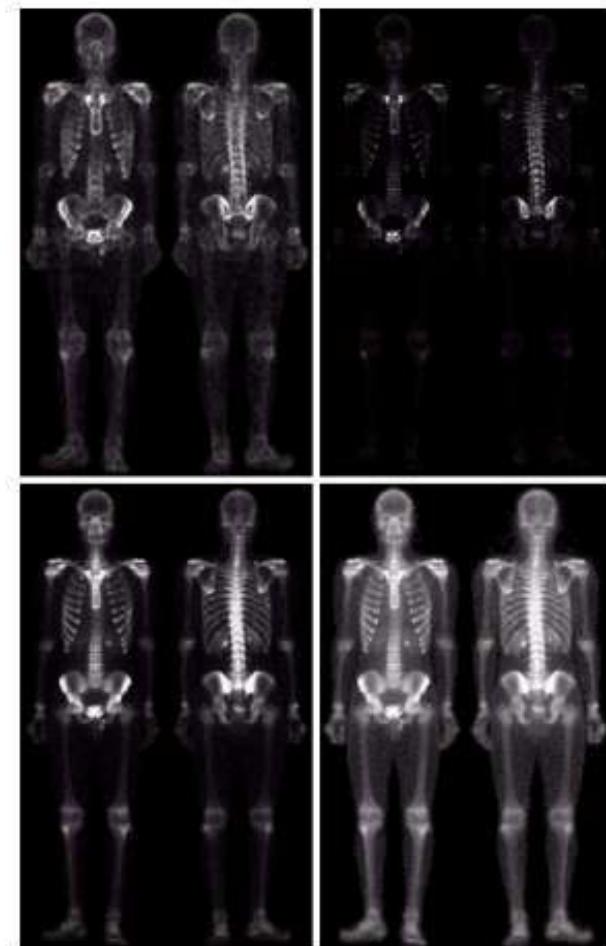


FIGURE 3.46
(Continued)
(e) Sobel image smoothed with a 5×5 averaging filter. (f) Mask image formed by the product of (c) and (e).
(g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)

Combining spatial Enhancement Methods

- Our **objective** is to enhance the image by sharpening it and by bringing out more of the skeletal detail.
- The narrow dynamic range of the gray levels and high noise content make this image **difficult** to enhance.
- The **strategy** we will follow is:
 - to utilize the **Laplacian** to highlight fine detail, and
 - the **gradient** to enhance prominent edges.
 - a smoothed version of the gradient image will be used to mask the Laplacian image
 - Finally, we will attempt to increase the dynamic range of the gray levels by using a **gray-level transformation**.



Thank
you

Conclusions

With image enhancement we try to make images look better according to **subjective** criteria.

Contrast enhancement of a gray image can be achieved by **manipulating the gray values** of the pixels so that they become more diverse.

This can be done by defining a **transformation** that converts the distribution of the gray values to a pre-specified shape. The choice of this shape may be totally arbitrary.

Smoothing spatial filters have been used for blurring and for noise reduction, while **sharpening spatial filters** have been used to highlight fine details in the image or to enhance detail that has been blurred ...

MATLAB (average +sobel+laplace)

Fspecial : for choosing the filter:

```
X=fspecial('average',[3 3])
```

```
p=fspecial('laplacian', 0)
```

```
v=fspecial('sobel') → horizontal sobel
```

```
Y=v' → vertical sobel
```

Imfilter : for applying filter.

```
m= imread('cameraman.tif');
```

```
f= imfilter(m,x) → this command will apply average filter on image
```

```
Fp=imfilter(m,p) → this command will apply laplace filter on image
```

```
Imshow(fp) → this command will show the laplace sharpened image
```

```
imshow(m+fp) → this command will show the filtered image after applying  
la place
```

MATLAB – cont. (unsharp + highboost)

Applying unsharp::

```
X=imread('cameraman.tif')  
p=fspecial('average',[3 3])  
xblur=imfilter(x,p) → for bluring
```

Mask= xblur-x → for subtracting the mask

Y=x+mask → for unsharp the image

Revise slide 57 + 58

For highboost , only multiply the mask any any constant in the last step: $y=x+3*mask$ for example

MATLAB – cont. (border padding)

Ex.

```
X=imread('cameraman.tif')
p=fspecial('laplacian', 0)
Xp=imfilter(x,p, 'replicate')
```

This command will apply border padding
instead of zero padding

Image Processing



Ch4:
Filtering in frequency domain

Image Transform

We can enhance the image in two ways:

- 1) spatial domain**
- 2) frequency domain**

In spatial we access the pixels themselves.

In frequency we convert the image into Fourier transform to make changes on this domain.

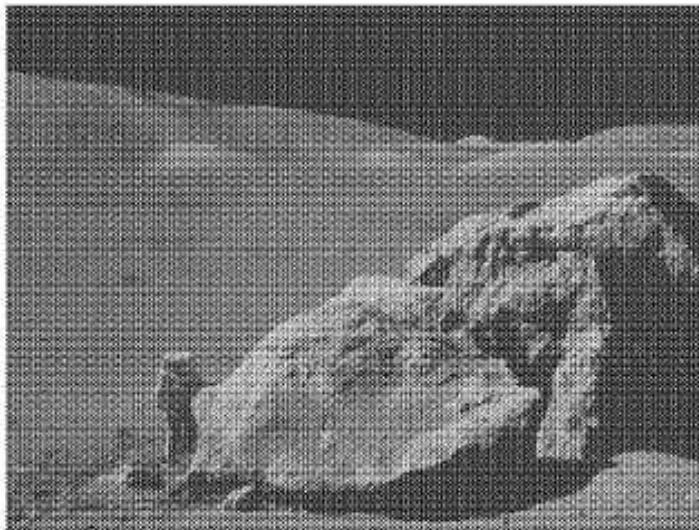
Image Transform

Image Transforms refer to the mathematical mapping of image pixels from the spatial domain to some other domain, where certain other characteristics of the image can be seen in a better manner.

Image Transform

In some cases, image processing tasks are best formulated by transforming the input image, carrying the specified task in a *transform domain*, and applying the inverse transform to return to the *spatial domain*.

Applications on Image Transform

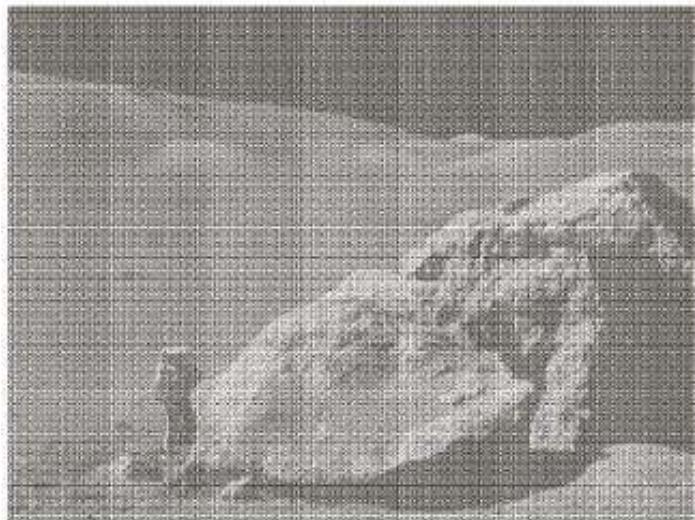


Original image

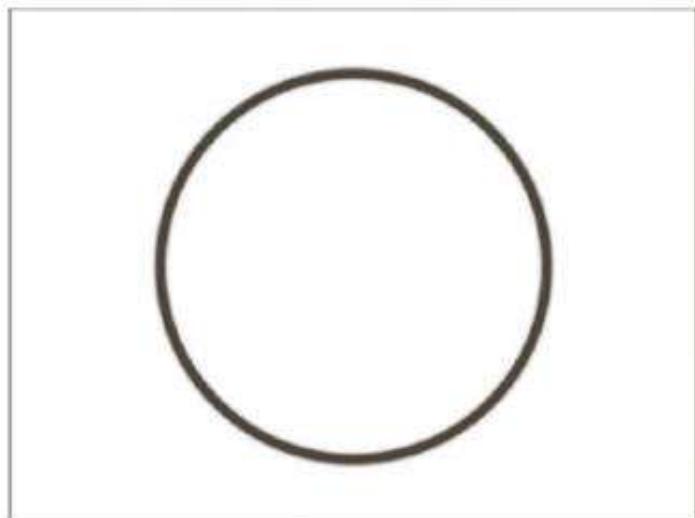


Filtered image in spatial
domain (average filter)

Applications on Image Transform

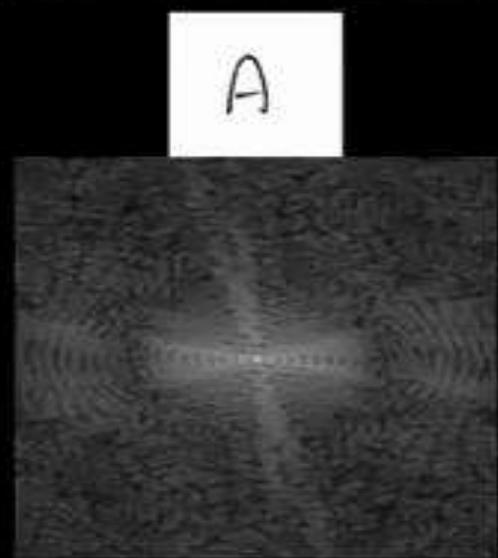
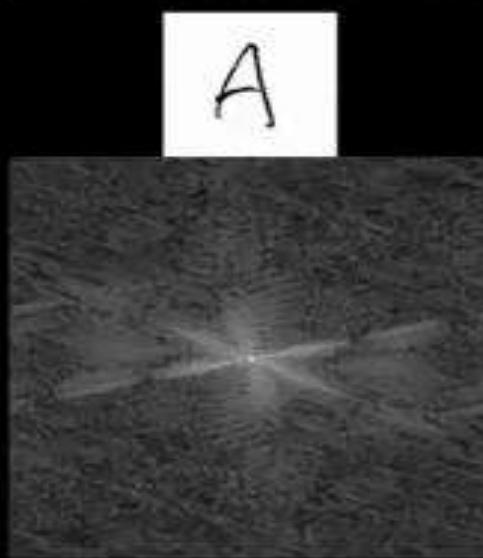
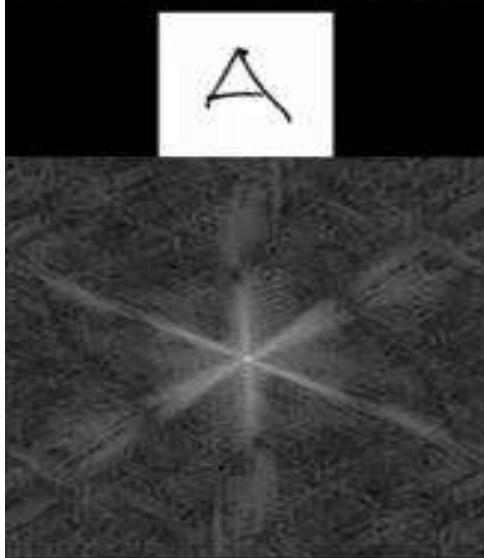
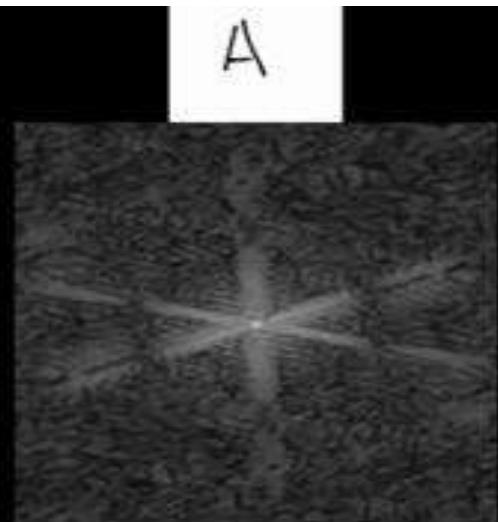
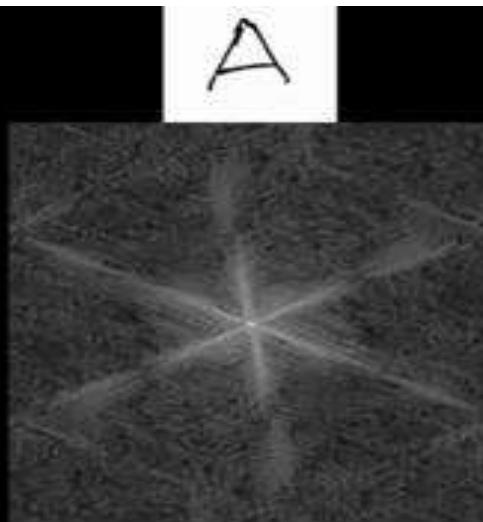
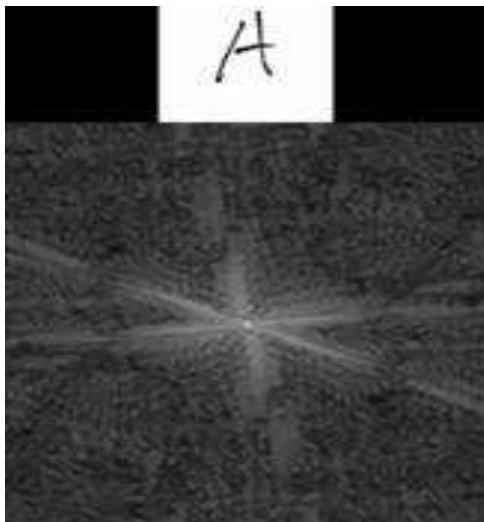
a
bc
d**FIGURE 2.40**

(a) Image corrupted by sinusoidal interference. (b) Magnitude of the Fourier transform showing the bursts of energy responsible for the interference. (c) Mask used to eliminate the energy bursts. (d) Result of computing the inverse of the modified Fourier transform. (Original image courtesy of NASA.)

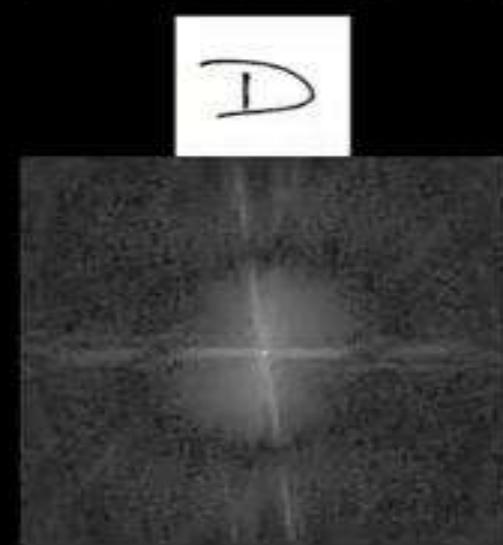
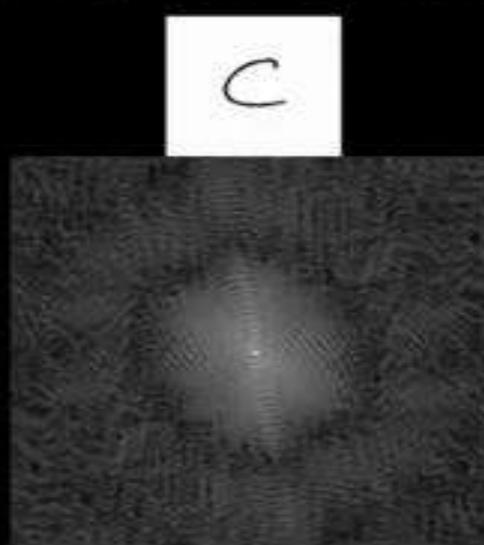
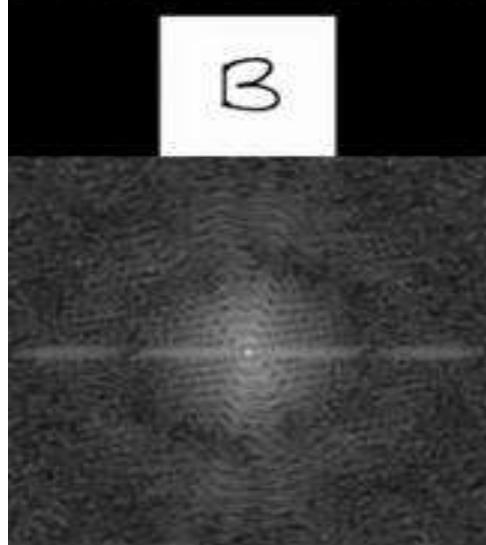
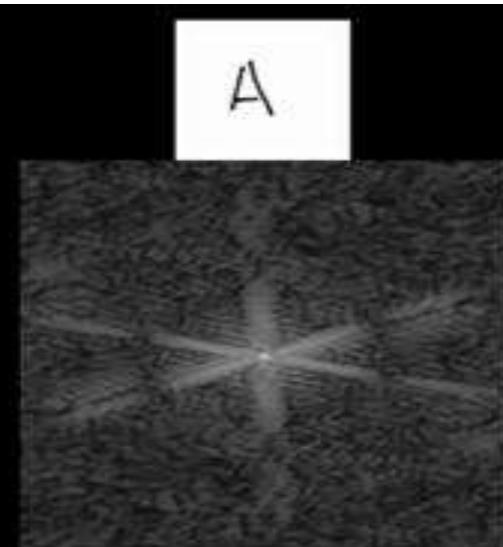
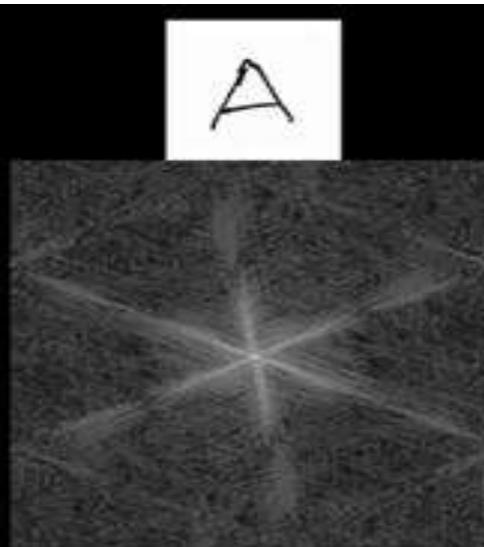
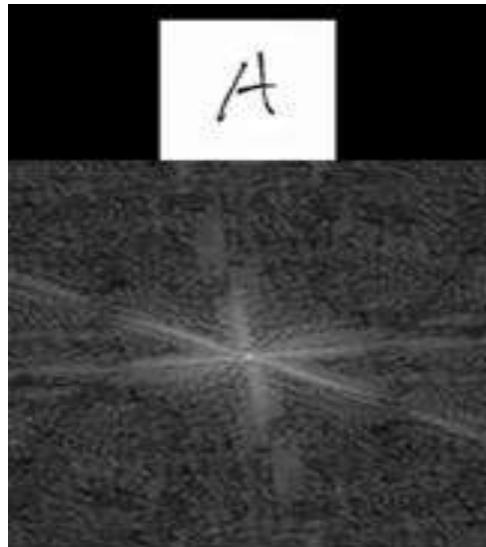


Filtered image in
frequency domain

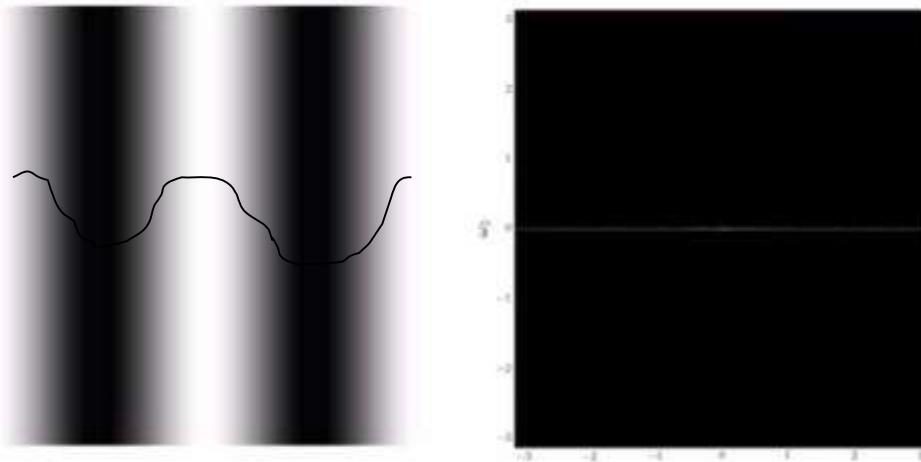
Applications on Image Transform



Applications on Image Transform

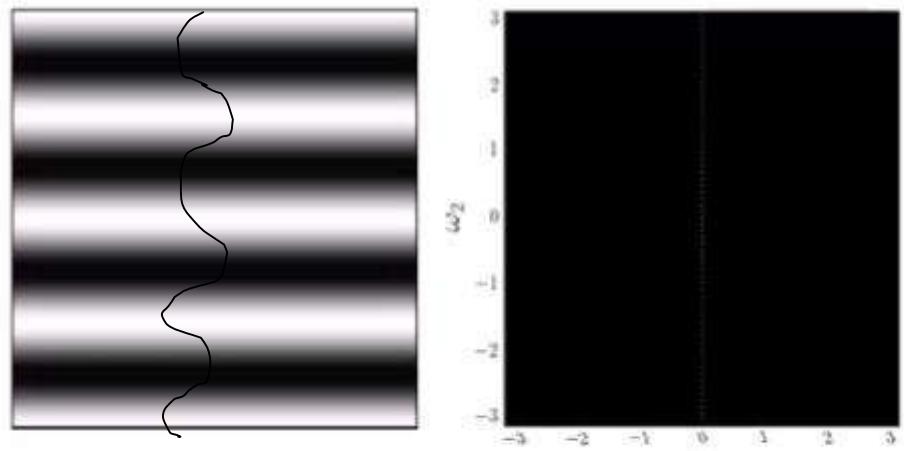


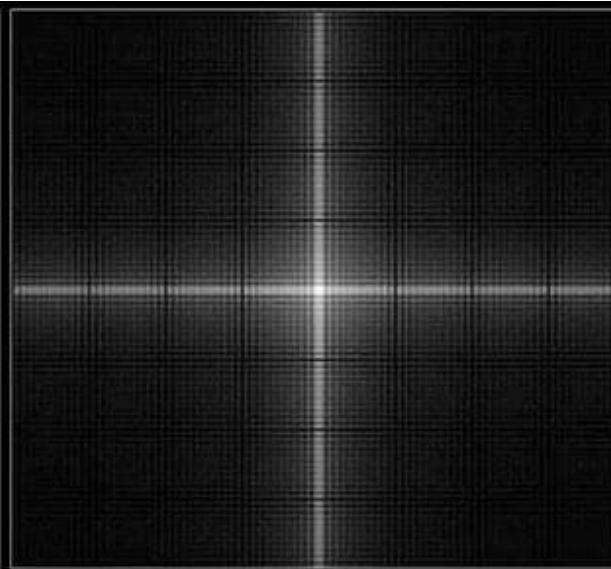
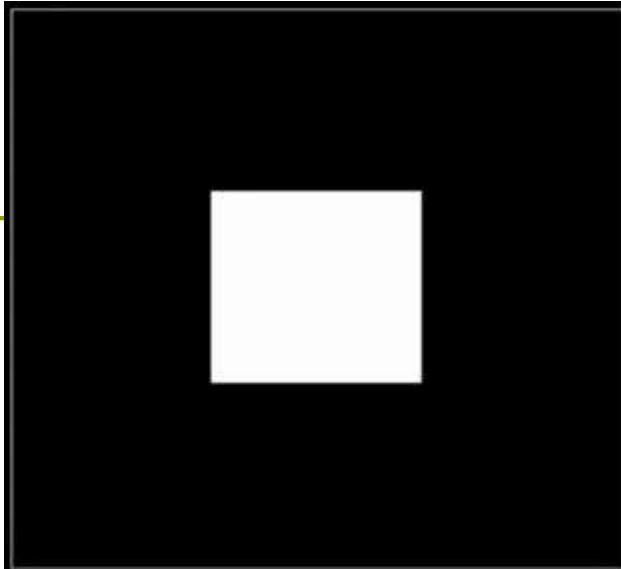
If we try to put a 2D sine waves through a Fourier transform



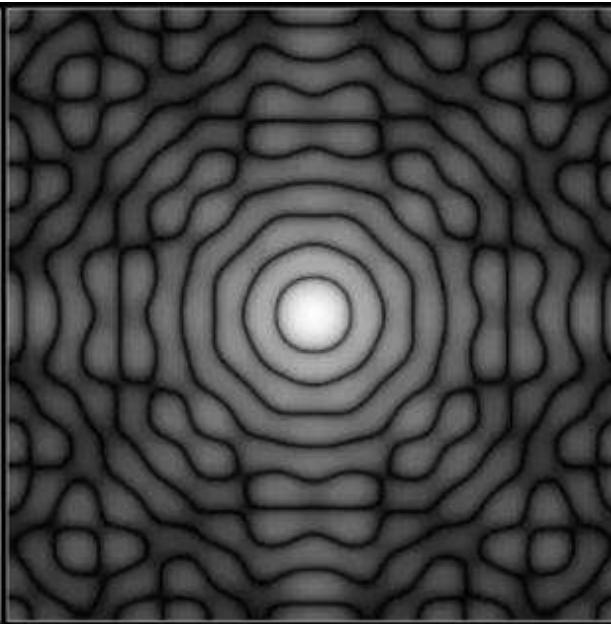
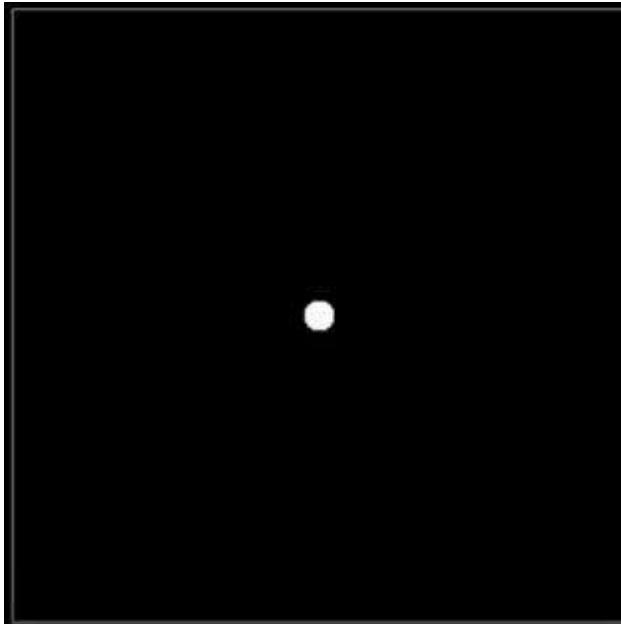
In the case where sine waves are either vertical or horizontal you see a corresponding vertical or a horizontal line of speck's in the Fourier Transform

For example, in the case where we have horizontal sine waves we see a horizontal line of speck's, this means that our image is a linear combination of many horizontal sine waves (this picture is constantly going up and down)



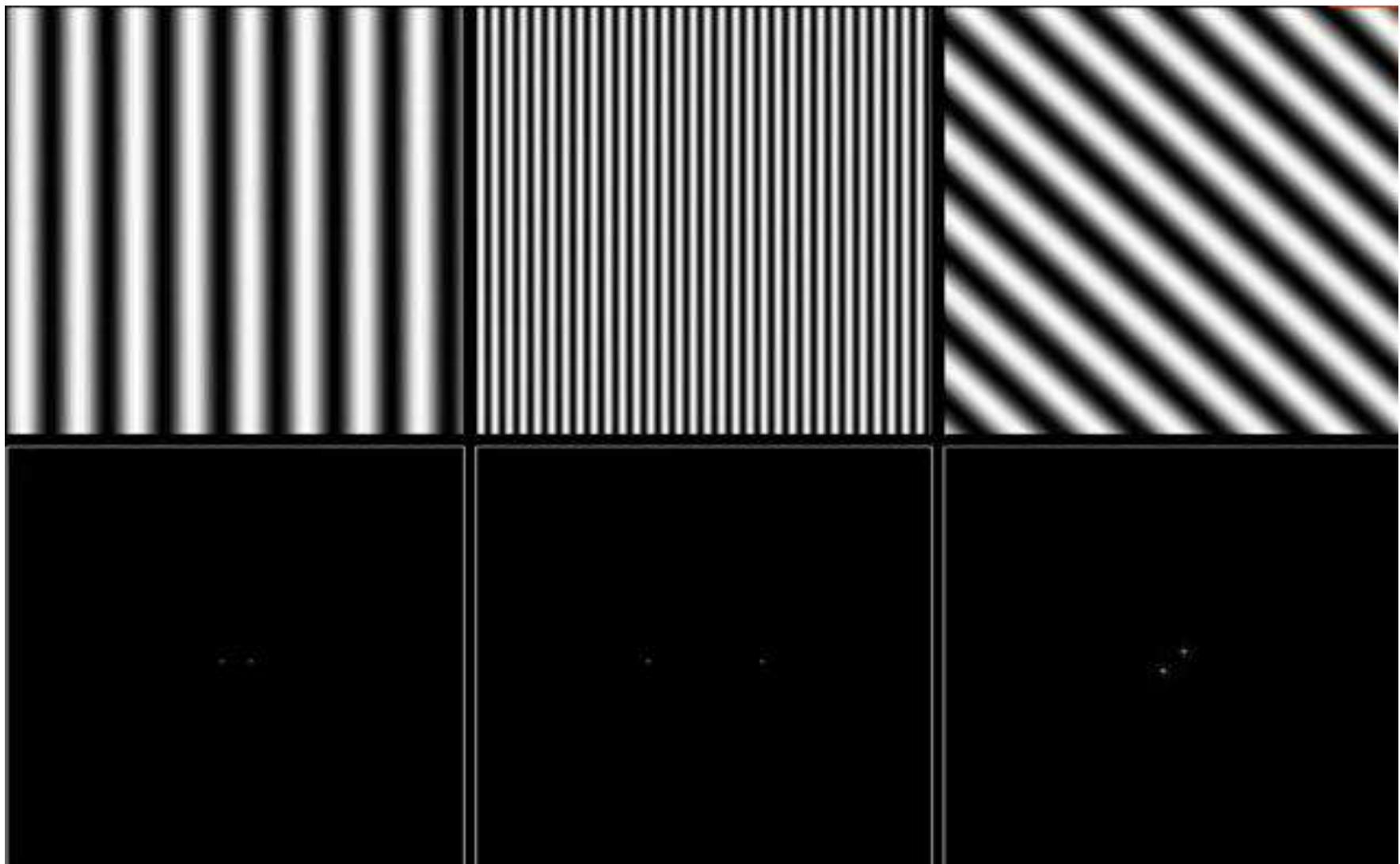


A square with its
Fourier transform

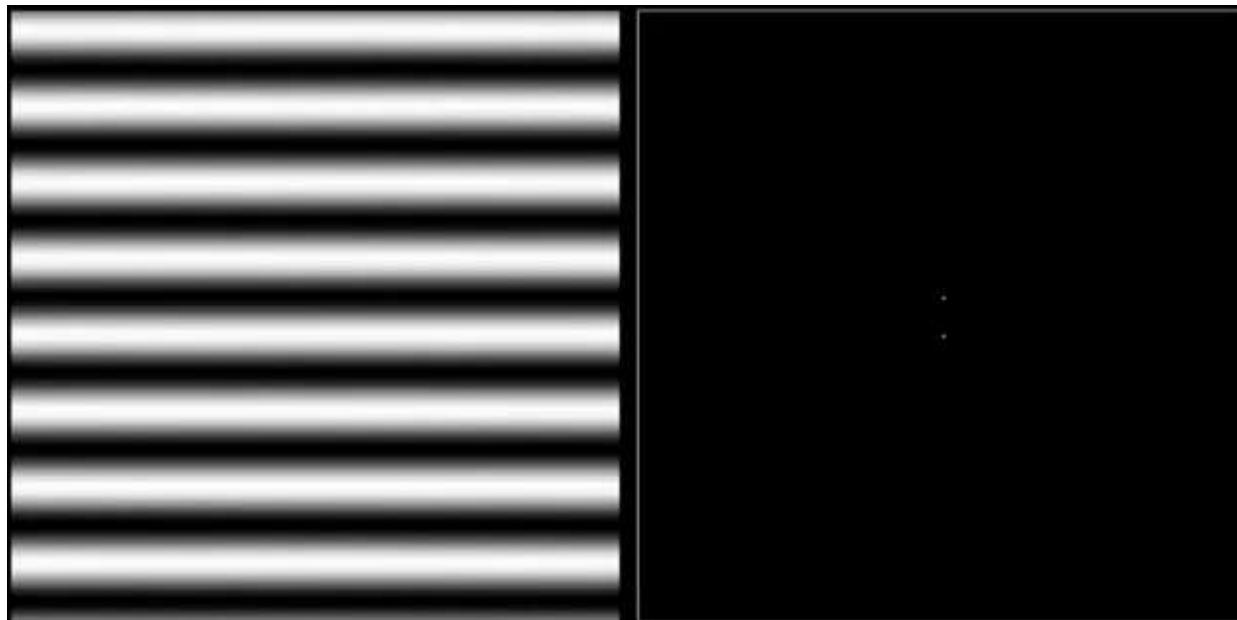


A circle with its
Fourier transform

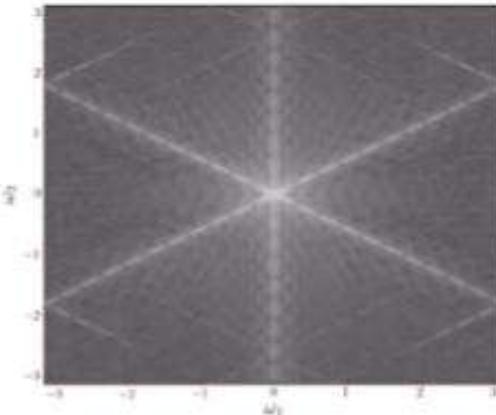
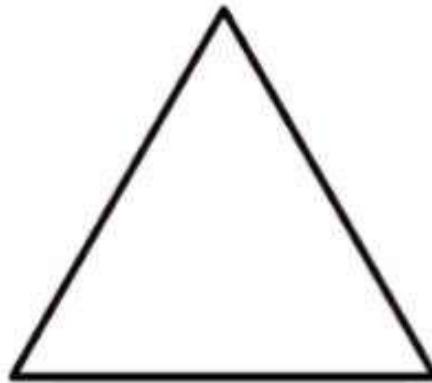
2D sine waves with its Fourier transform



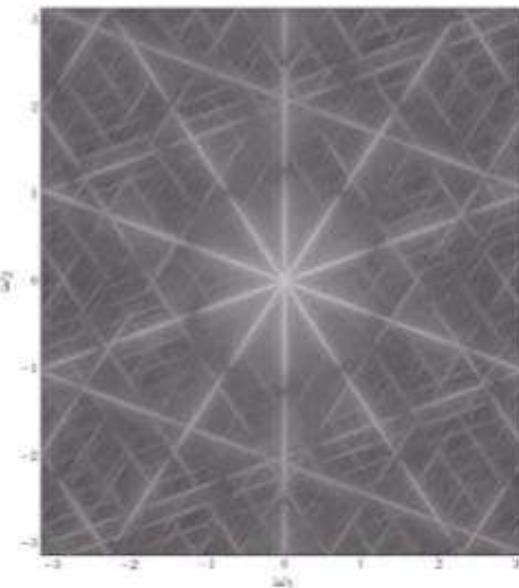
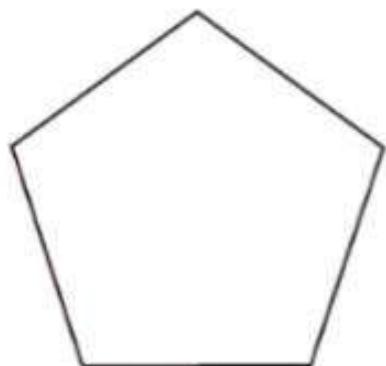
A vertical 2D sine waves with its Fourier transform



The number of the streaks on the middle is the same number of sides in our polygon

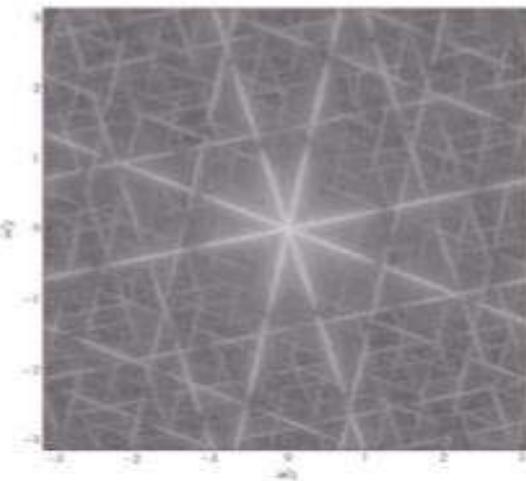
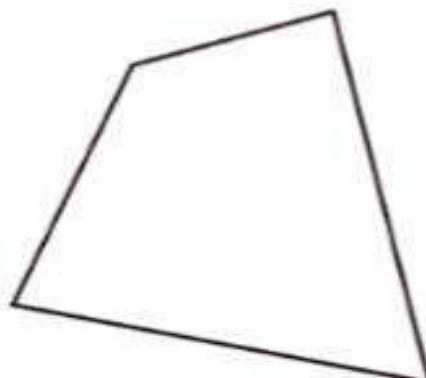
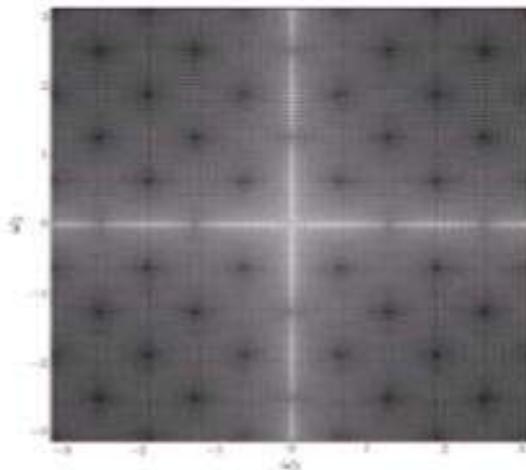


We see three streaks for the triangle

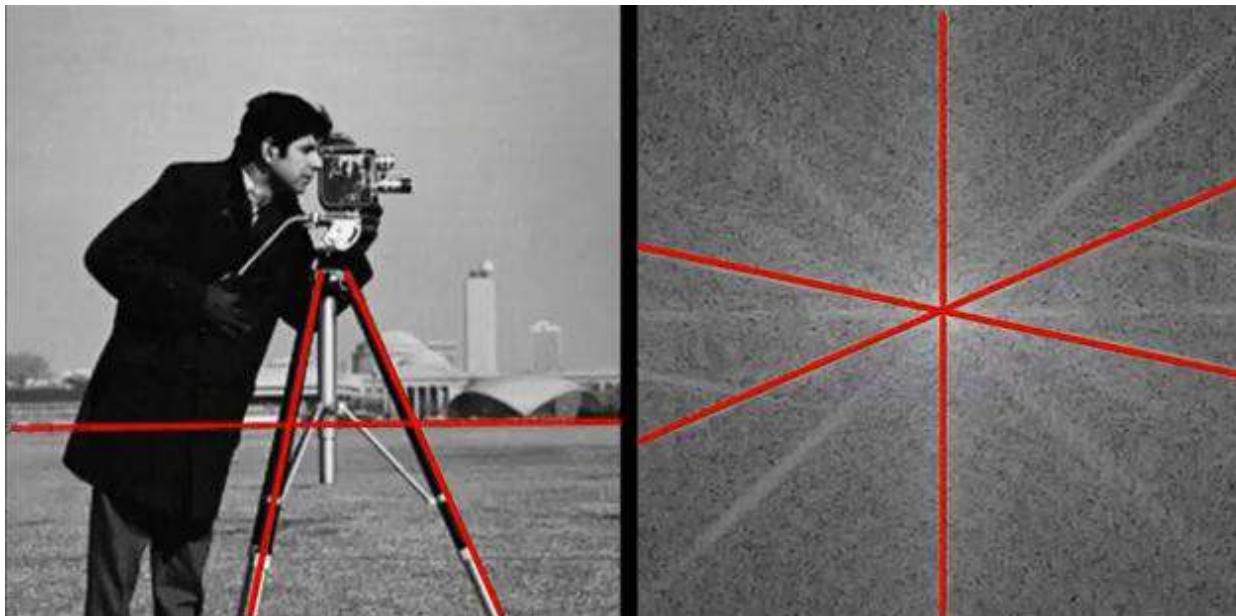
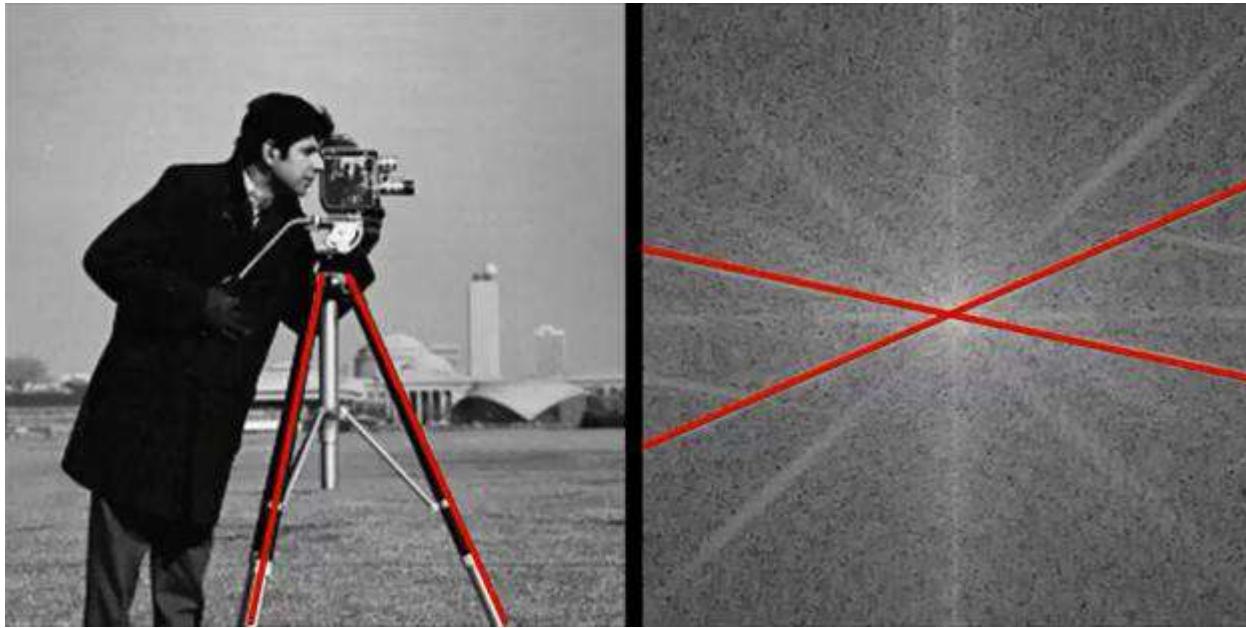


We see five streaks for the pentagon

However, for the square we see only two streaks this is because each streak in the middle represents line going in a certain direction



Therefore, if you use a quadrilateral with no parallel edges we will observe the expected four streaks.



Fourier transform

Fourier Series: Any function that periodically repeats itself can be expressed as the sum of complex exponentials (sines and/or cosines) of different frequencies, each multiplied by a different coefficient.

- **Fourier transform:** Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weighting function

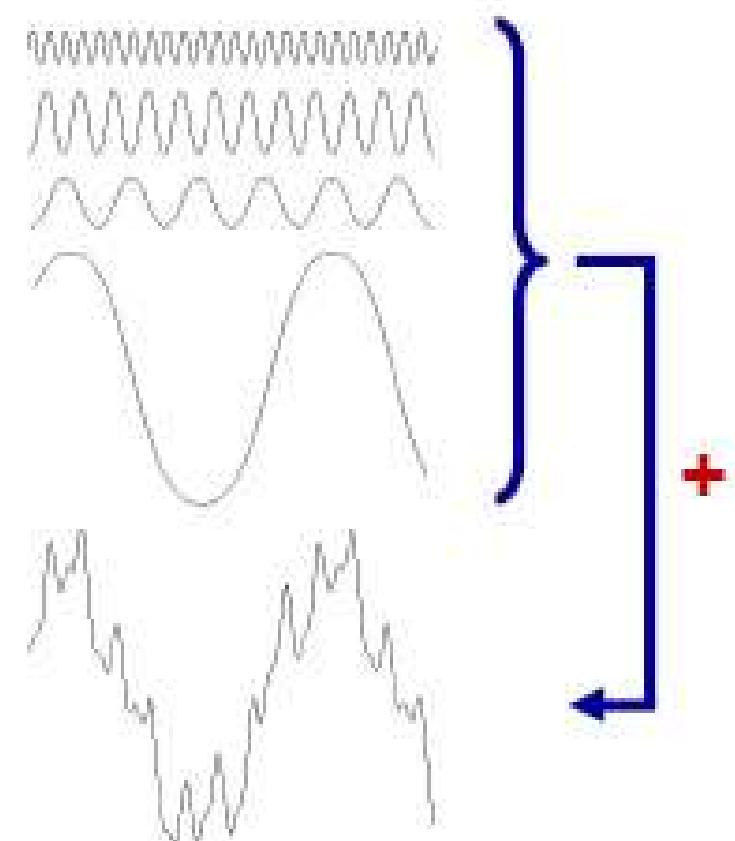


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum

The function at the bottom is the sum of the four functions above it.

Fourier transform

Each component of Fourier transform $F(u)$ is composed of the sum of the all values of the function $f(x) \rightarrow$ where it's multiplied by sines and cosines of the various frequencies

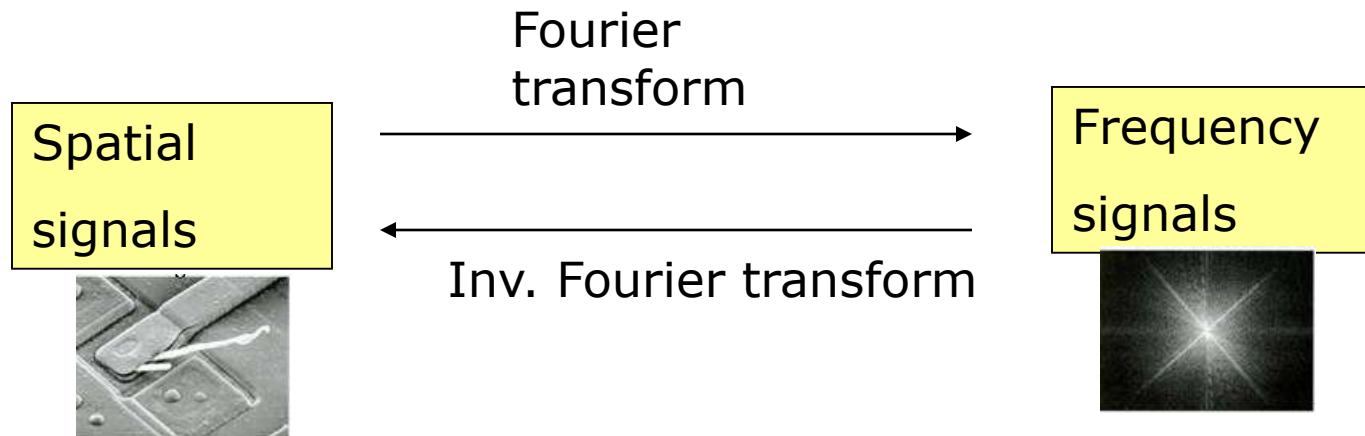
Because u determines the frequency of the components of the transform \rightarrow it is called a frequency domain

Fourier Transform vs. Glass Prism

- **Prism:** device that separates light into various color components each depending on its wavelength (frequency) content.
- **Fourier Transform:** mathematical prism that separates a function into various components based on frequency contents.



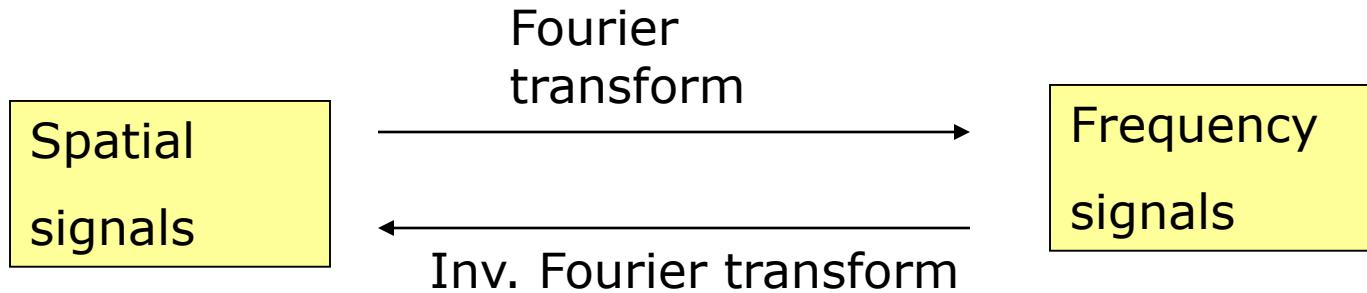
Fourier → spatial spatial → Fourier



Important characteristic of this function that can be reconstructed (recovered) completely via an inverse process with no loss of information

Fourier → spatial

spatial → Fourier



There are four cases to study:

- 1) 1-D continuous case (FFT/IFFT2)
- 2) 1-d discrete case (DFT/IDFT)
- 3) 2-d continuous case (FFT2 / IFFT2)
- 4) 2-d discrete case (DFT2 / IDFT2)

Fourier → spatial

spatial → Fourier

هناك اربع انواع من اشارات ال *spatial frequency* ممكن تحويلها الى اشارات

There are four cases to study:

1) 1-D continuous signal case

تحويل اشارة مستمرة ذات .. frequency الى spatial بعد واحد من .. مثال: اشارات الصوت

2) 1-D discrete case

تحويل اشارة متقطعة ذات .. digital sound الى spatial بعد واحد من .. مثال: اشارات الصوت المخزنة حاسوبيا (الرقمية) :

3) 2-D continuous case

تحويل اشارة مستمرة ثنائية الابعاد من .. الصورة التي لم تخزن بعد في .. frequency الى spatial مثال: الصورة التي لم تخزن بعد في الحاسوب

4) 2-D discrete case

تحويل اشارة متقطعة ثنائية الابعاد من .. digital image الى spatial مثال: اشارات الصورة المخزنة حاسوبيا (الرقمية) :

1) 1-D continuous signal case

The Fourier Transform (FT) is given by:

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

□ The **Inverse** Fourier Transform is given by:

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

2) 1-D discrete signal case

Suppose $f = [f_0, f_1, f_2, \dots, f_{N-1}]$

is a sequence of length N. Define its discrete Fourier transform

$$F = [F_0, F_1, F_2, \dots, F_{N-1}]$$

where $F_u = \frac{1}{N} \sum_{x=0}^{N-1} \exp[-2\pi i \frac{xu}{N}] \cdot f_x$

2) 1-D discrete signal case

The Fourier Transform (FT) is given by: $e^{\pm j\theta} = \cos(\theta) \pm j\sin(\theta)$

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \quad \text{for } u = 0, 1, 2, \dots, M - 1.$$

Simple periodic patterns

□ The **Inverse** Fourier Transform is given by:

$$f(x) = \sum_{u=0}^{M-1} F(u) e^{j2\pi ux/M} \quad \text{for } x = 0, 1, 2, \dots, M - 1.$$

Note:

➤ In the discrete case ...
the integrals are replaced by sums.

Matlab Implementation:
Functions fft, ifft, fftshift.

2) 1-D discrete signal case – example

Q: Suppose you have the following discrete signal array, convert $f(0)$ to Fourier transform?

$f(x)$	50	10	30	10
--------	----	----	----	----

We use the previous formula to convert:

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \quad \text{for } u = 0, 1, 2, \dots, M - 1.$$

M=4

$$F(0) = \frac{1}{4} (f(0) * e^{-j*2*\pi*0*0/4} + f(1) * e^{-j*2*\pi*0*1/4} + f(2) * e^{-j*2*\pi*0*2/4} + f(3) * e^{-j*2*\pi*0*3/4})$$

$$= \frac{1}{4} (50 * e^0 + 10 * e^0 + 30 * e^0 + 10 * e^0)$$

$$= \frac{1}{4}(50*1 + 10 * 1 + 30 * 1 + 10 * 1)$$

$$= 25$$

$F(u)$	25			
--------	----	--	--	--

NOTE that $F(0) = \text{average of } f(x)$

2) 1-D discrete signal case – example – continued.

$f(x)$	50	10	30	10
--------	----	----	----	----

والآن نعيد المعادلة لكل $F(u)$ لنحصل على كل القيم، حاول ان تحول $f(1)$ الى fourier حسب المعادلة

$$\begin{aligned} F(1) &= \frac{1}{4} (f(0) * e^{-j*2*\pi*1*0/4} + f(1) * e^{-j*2*\pi*1*1/4} + f(2) * e^{-j*2*\pi*1*2/4} + f(3) * e^{-j*2*\pi*1*3/4}) \\ &= \frac{1}{4} (50 * e^0 + 10 * e^{-j\pi/2} + 30 * e^{-j\pi} + 10 * e^{-j*3\pi/2}) \end{aligned}$$

Note that in the result we will have an imaginary part since there is j which means $\sqrt{-1}$

i.e. $F(u)$ consists of two parts: real $R(u)$ and imaginary $I(u)$.

2) 1-D discrete signal case

$$F(0) = \sum_{x=0}^3 f(x) = [f(0) + f(1) + f(2) + f(3)] \\ = 1 + 2 + 4 + 4 = 11$$

The next value of $F(u)$ is

$$F(1) = \sum_{x=0}^3 f(x) e^{-j2\pi(1)x/4} \\ = 1e^0 + 2e^{-j\pi/2} + 4e^{-j\pi} + 4e^{-j3\pi/2} = -3 + 2j$$

Similarly, $F(2) = -(1 + 0j)$ and $F(3) = -(3 + 2j)$. Observe that *all* values of $f(x)$ are used in computing *each* term of $F(u)$.

If instead we were given $F(u)$ and were asked to compute its inverse, we would proceed in the same manner, but using the inverse transform. For instance,

$$f(0) = \frac{1}{4} \sum_{u=0}^3 F(u) e^{j2\pi u(0)} \\ = \frac{1}{4} \sum_{u=0}^3 F(u) \\ = \frac{1}{4} [11 - 3 + 2j - 1 - 3 - 2j] \\ = \frac{1}{4} [4] = 1$$

$f(x)$	1	2	4	4
--------	---	---	---	---

A memory aid for evaluating $e^{j\theta}$

$$\begin{aligned}\theta = 0 &\Rightarrow e^{j\theta} = +1 \\ \theta = \pi/2 &\Rightarrow e^{j\theta} = +j \\ \theta = \pi &\Rightarrow e^{j\theta} = -1 \\ \theta = 3\pi/2 &\Rightarrow e^{j\theta} = -j \\ \theta = 2\pi &\Rightarrow e^{j\theta} = +1 \\ \theta = 5\pi/2 &\Rightarrow e^{j\theta} = +j\end{aligned}$$

and so on...

Example

$f(x)$	1	2	4	4
--------	---	---	---	---

$$F(u) = \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \quad \text{for } u = 0, 1, 2, \dots, M-1.$$

M=4

$$F(0) = (f(0) * e^{-j*2*\pi*0*0/4} + f(1) * e^{-j*2*\pi*0*1/4} + f(2) * e^{-j*2*\pi*0*2/4} + f(3) * e^{-j*2*\pi*0*3/4})$$

$$= (1 * e^0 + 2 * e^0 + 4 * e^0 + 4 * e^0)$$

$$= (1*1 + 2*1 + 4*1 + 4*1)$$

$$F(0) = 11$$

Example (cont.)

M=4

$$\begin{aligned} F(1) &= (f(0) * e^{-j*2*\pi*1*0/4} + f(1) * e^{-j*2*\pi*1*1/4} + f(2) * e^{-j*2*\pi*1*2/4} + f(3) * e^{-j*2*\pi*1*3/4}) \\ &= (1 * e^0 + 2 * e^{-j\pi/2} + 4 * e^{-j\pi} + 4 * e^{-j3\pi/2}) \end{aligned}$$

$$=(1 * 1 + 2 * -j + 4 * -1 + 4 * j)$$

$$=(1 - 2j - 4 + 4j)$$

$$F(1) = -3 + 2j$$

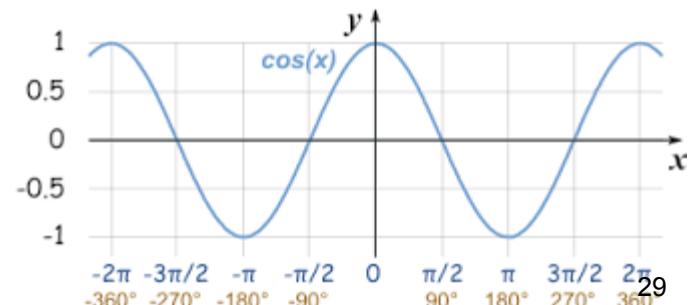
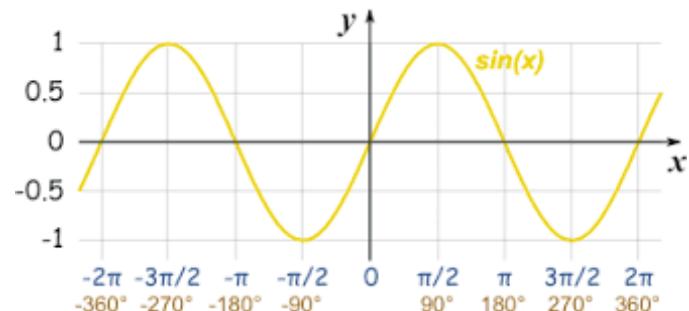
$$e^{\pm j\theta} = \cos(\theta) \pm j\sin(\theta)$$

By substituting in the above equation :

$$e^{-j\pi/2} = \cos(\pi/2) - j\sin(\pi/2)$$

$$e^{-j\pi/2} = 0 - j * 1$$

$$e^{-j\pi/2} = -j$$



Example (cont.)

$$e^{\pm j\theta} = \cos(\theta) \pm j\sin(\theta)$$

By substituting in the above equation :

$$e^{-j\Pi} = \cos(\Pi) - j^*\sin(\Pi)$$

$$e^{-j\Pi} = -1 - j^* 0$$

$$e^{-j\Pi} = -1$$

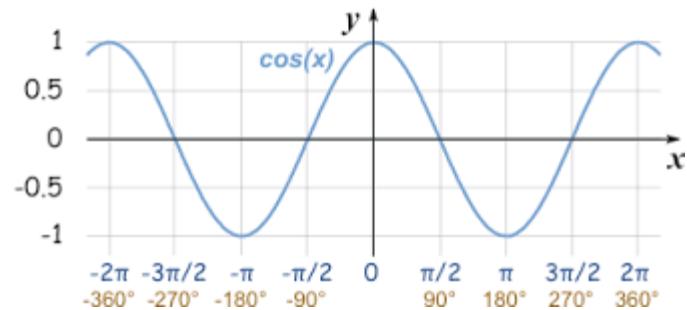
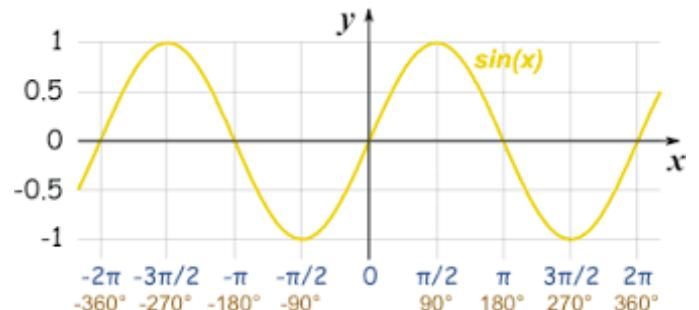
$$e^{\pm j\theta} = \cos(\theta) \pm j\sin(\theta)$$

By substituting in the above equation :

$$e^{-j^*3\Pi/2} = \cos(3\Pi/2) - j^*\sin(3\Pi/2)$$

$$e^{-j\Pi} = 0 - j^* -1$$

$$e^{-j\Pi} = +j$$



Polar Coordinates

$$f(x) = F(u)$$

$$\begin{bmatrix} 1 \\ 2 \\ 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 11 \\ -3 + 2j \\ -1 \\ -3 - 2j \end{bmatrix}$$

$$\begin{array}{c}
 \begin{array}{ccccc}
 & & \nearrow & & \\
 & & & & \\
 \left[\begin{array}{c} 11 \\ -3 \\ -1 \\ -3 \end{array} \right] & & & & \left[\begin{array}{c} 0 \\ 2 \\ 0 \\ -2 \end{array} \right]
 \end{array} \\
 \text{Real} \quad \text{Imag}
 \end{array}$$

$F(u)$ is a matrix of complex numbers and it is not useful and if we convert it into a polar coordinates it will become more useful information.

Firstly, we will decompose the complex matrix into two matrices (real and imaginary), then we can calculate the polar coordinate (Magnitude and Phase angle) which are more useful

Phase , magnitude, power spectrum in 1-D.

$$F(u) = |F(u)| e^{-j\phi(u)},$$

where $|F(u)| = [R^2(u) + I^2(u)]^{1/2}$ and $\phi(u) = \tan^{-1} \left[\frac{I(u)}{R(u)} \right]$

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

1

power spectrum

3) 2-D continuous signal case

The Fourier Transform (FT) is given by:

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy$$

□ The **Inverse** Fourier Transform is given by:

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)} du dv$$

Note:

- 2-D case is an extension of the 1-D concepts.
- Mostly applied for 2-D Image Processing.

4) 2-D discrete signal case

The Fourier Transform (FT) is given by:

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

□ The **Inverse** Fourier Transform is given by:

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

2-D discrete signal case – example

Q: Suppose you have the following discrete image, convert $f(0,0)$ to Fourier transform?

We use the previous formula to convert: الحل:

$f(x,y)$		
5	10	20
5	15	5

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

M=3, N=2

$F(0,0) =$

$$1/6 (f(0,0) * e^{-j*2*\pi*(0*0/3+ 0*0/2)} + f(0,1) * e^{-j*2*\pi*(0*0/3+ 0*1/2)} + f(0,2) * e^{-j*2*\pi*(0*0/3+ 0*2/2)} + f(1,0) * e^{-j*2*\pi*(0*1/3+ 0*0/2)} + f(1,1) * e^{-j*2*\pi*(0*1/3+ 0*1/2)} + f(1,2) * e^{-j*2*\pi*(0*1/3+ 0*2/2)})$$

$$= 1/6 (5 * e^0 + 10 * e^0 + 20 * e^0 + 5 * e^0 + 15 * e^0 + 5 * e^0)$$

$$= 1/6(60) = 10$$

NOTE that $F(0,0) = \text{average of } f(x,y)$

نعيد المعادلة لكل بكسلات الصورة لنحو من Fourier الى spatial وسنرى ان هناك كما تعلمبا سابقا رقم صحيح ورقم تخيلي لكل بكسل تقريبا.

$F(u,v)$		
10		

Properties of the Fourier Transform

- $F(0,0)$ - value is by far the largest component of the image,
- Other frequency components are usually much smaller,
- The magnitude of $F(X,Y)$ decreases quickly
 - Instead of displaying the $|F(u,v)|$ we display $\log(1 + |F(u,v)|)$ real function usually

Image Spectra

$F(u, v)$ is a complex number; its real and imaginary parts are not particularly informative in them selves; it is far more useful to think of the magnitude and phase of $F(u, v)$.

Image Spectra

$|F(u, v)|$ is the magnitude and $\phi(u, v)$ is the phase. They can be written as:

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

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

The above equations allow us to decompose an array of complex coefficients into an array of magnitudes and an array of phases.

Phase , magnitude, power spectrum in 2-D.

magnitude



phase



Polar coordinates: $|F(u, v)| = [R^2(u, v) + I^2(u, v)]^{1/2}$ $\phi(u, v) = \tan^{-1} \left[\frac{I(u, v)}{R(u, v)} \right]$

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

Image Spectra

The amplitude, A , is the size of the variation –the height of a peak.

The phase, ϕ , is the position of the start of a cycle relative to some reference point (e. g., the origin)

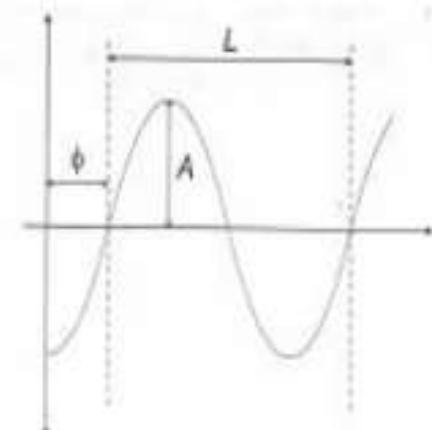


Image Spectra

The **magnitudes** correspond to the **amplitude** of the basis images in the Fourier representation.

The array of magnitudes is termed **amplitude spectrum** of the image.

The array of phases is termed **phase spectrum** of the image.

When the term 'spectrum' is used on its own, the **amplitude spectrum** is normally implied. This is because the phases are generally less significant for the purpose of interpretation.

Image Spectra

The amplitude spectrum is sensitive to the presence of particular features in the an image.

Without amplitude information, we can no longer determine the relative brightnesses of the features, but we can at least see the boundaries between them, which aids recognition.

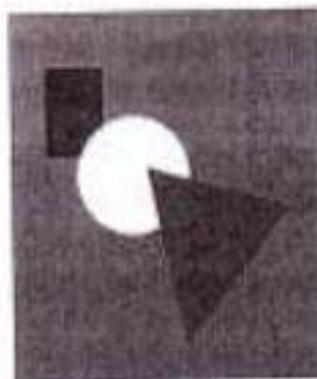
The phase spectrum encodes the features location in the image.

Without phase information, the spatial coherence of the image is disrupted and becomes impossible to recognize features of interest.

Because phase is so crucial to maintaining image integrity, phase spectrum is left untouched and manipulate only the amplitude spectrum.

Image Spectra

Examples of an amplitude and phase of an image:



(a)



(b)

amplitude:



(c)

Phase spectrum appears to be somewhat noisy!

Figure 8.8 A synthetic image and its spectra. (a) Image. (b) Amplitude spectrum. (c) Phase spectrum.

Image Spectra

If we attempt to reconstruct the image after destroying either the phase or amplitude information, then the reconstruction will fail! **WHY?**

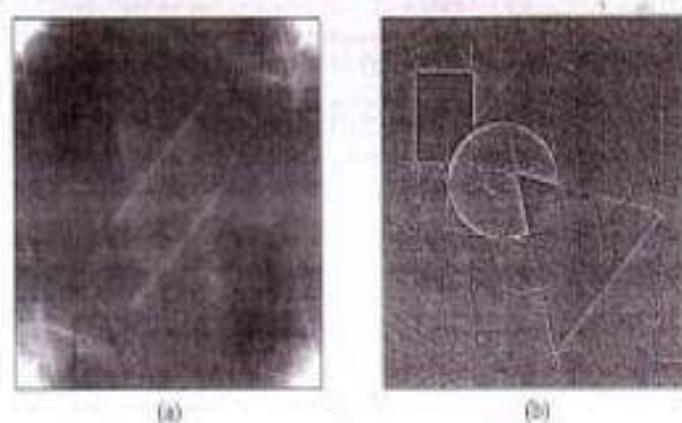
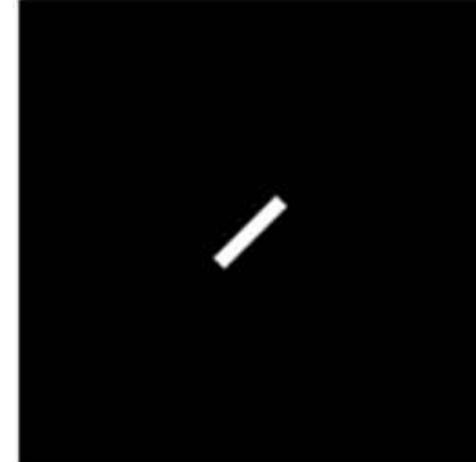


Figure 8.9 Reconstructions of an image from its spectra. (a) After destroying phase information. (b) After destroying amplitude information.

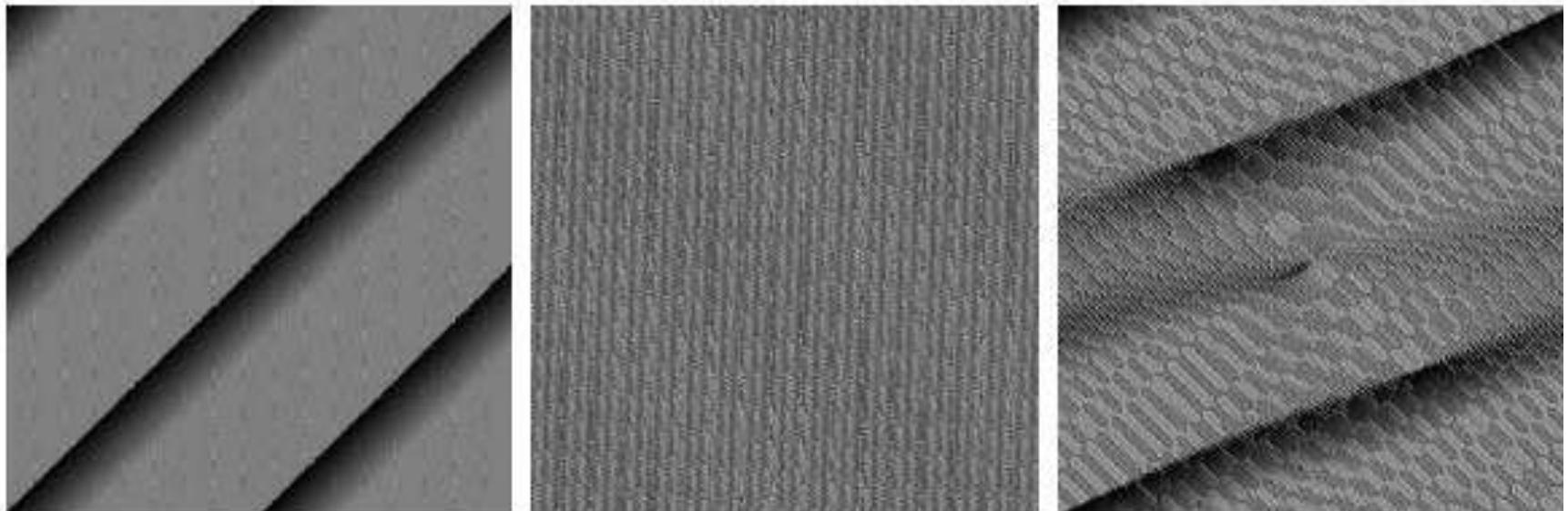
After destroying phase

After destroying amplitude

Example

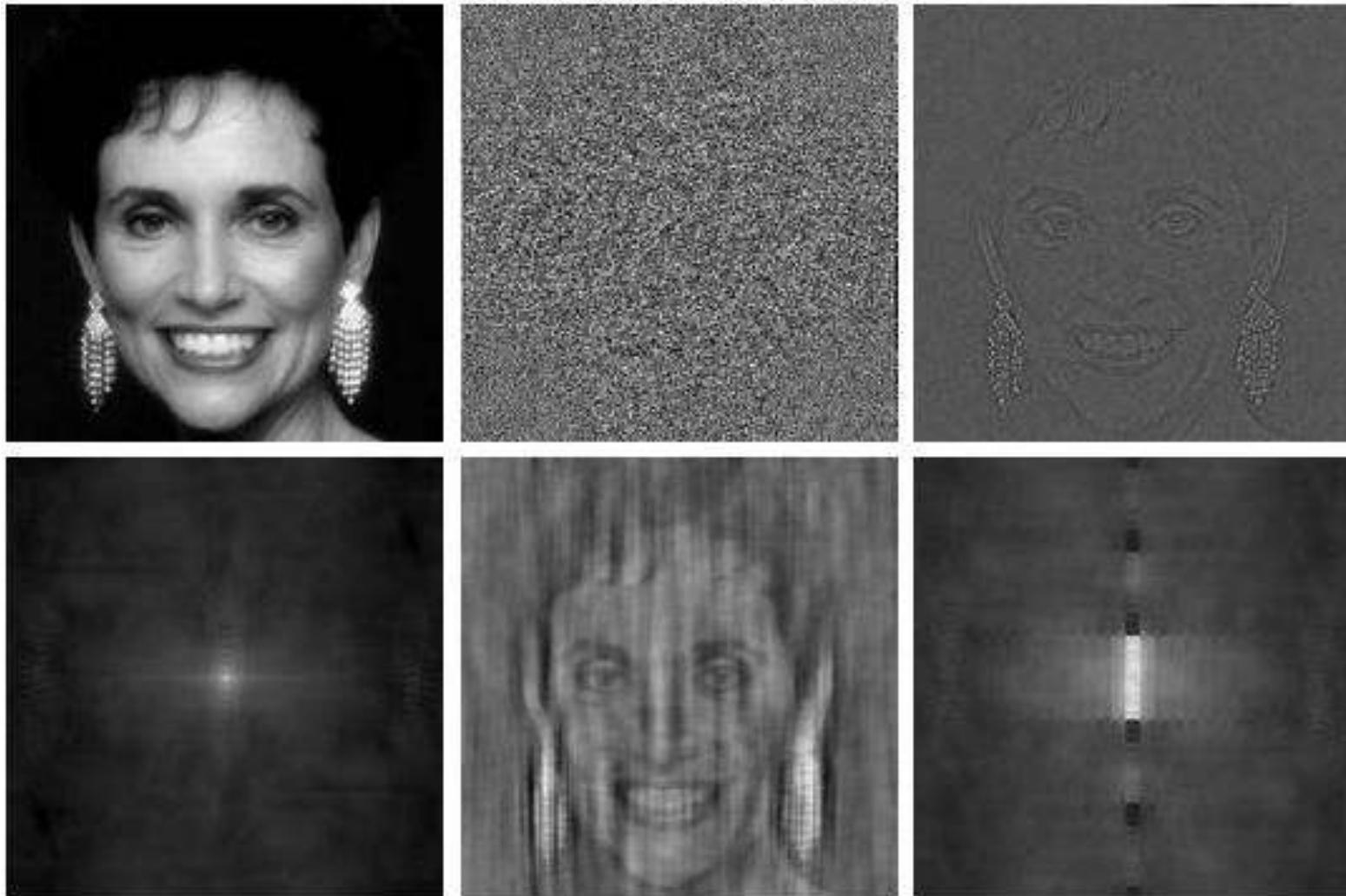


Phase Angle



a b c

FIGURE 4.26 Phase angle array corresponding (a) to the image of the centered rectangle in Fig. 4.24(a), (b) to the translated image in Fig. 4.25(a), and (c) to the rotated image in Fig. 4.25(c).



a b c
d e f

FIGURE 4.27 (a) Woman. (b) Phase angle. (c) Woman reconstructed using only the phase angle. (d) Woman reconstructed using only the spectrum. (e) Reconstruction using the phase angle corresponding to the woman and the spectrum corresponding to the rectangle in Fig. 4.24(a). (f) Reconstruction using the phase of the rectangle and the spectrum of the woman.

Shifting the origin

a b

FIGURE 4.3

(a) Image of a 20×40 white rectangle on a black background of size 512×512 pixels.

(b) Centered Fourier spectrum shown after application of the log transformation given in Eq. (3.2-2). Compare with Fig. 4.2.

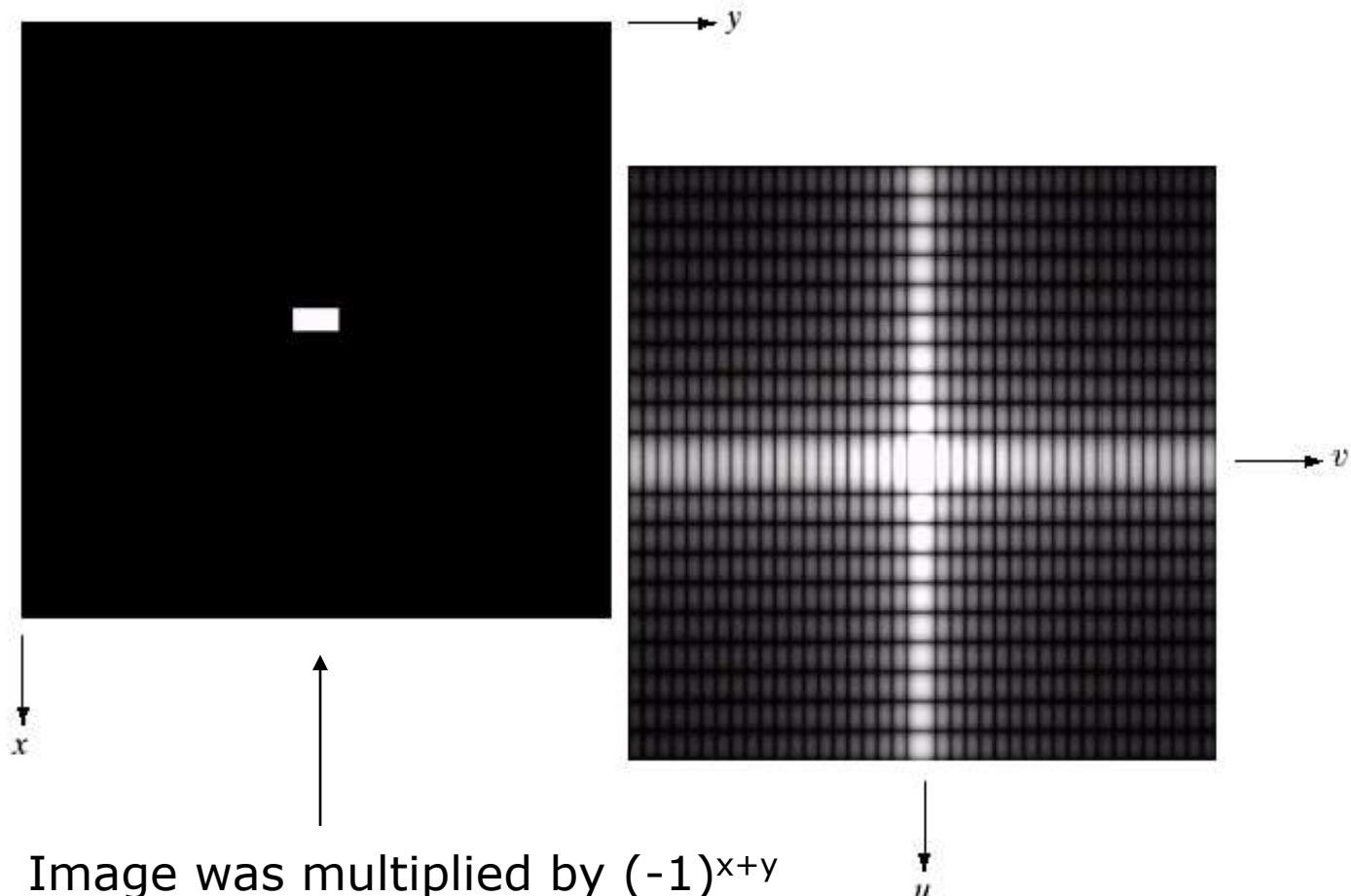


Image was multiplied by $(-1)^{x+y}$ prior to computing the transform

Shifting the origin

a b
c d

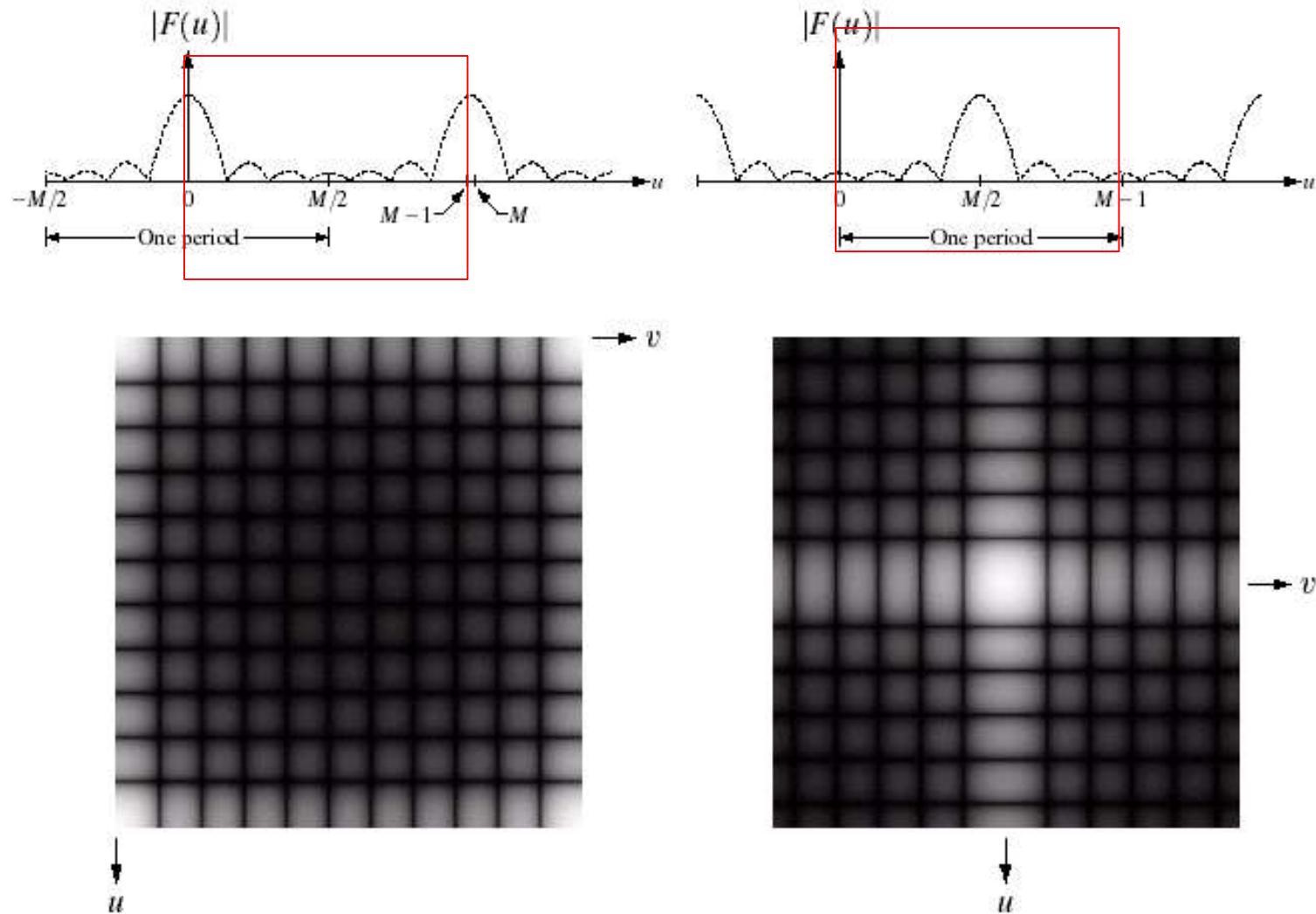
FIGURE 4.34

(a) Fourier spectrum showing back-to-back half periods in the interval $[0, M - 1]$.

(b) Shifted spectrum showing a full period in the same interval.

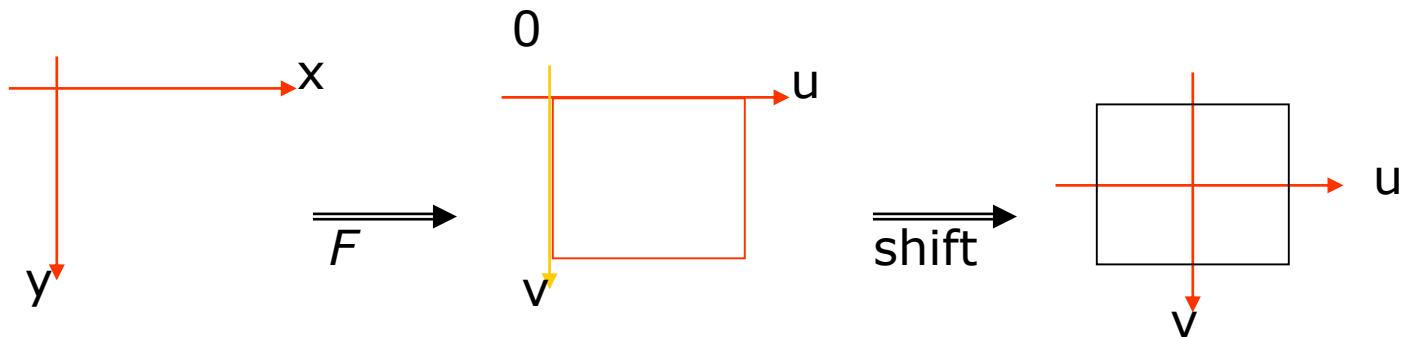
(c) Fourier spectrum of an image, showing the same back-to-back properties as (a), but in two dimensions.

(d) Centered Fourier spectrum.



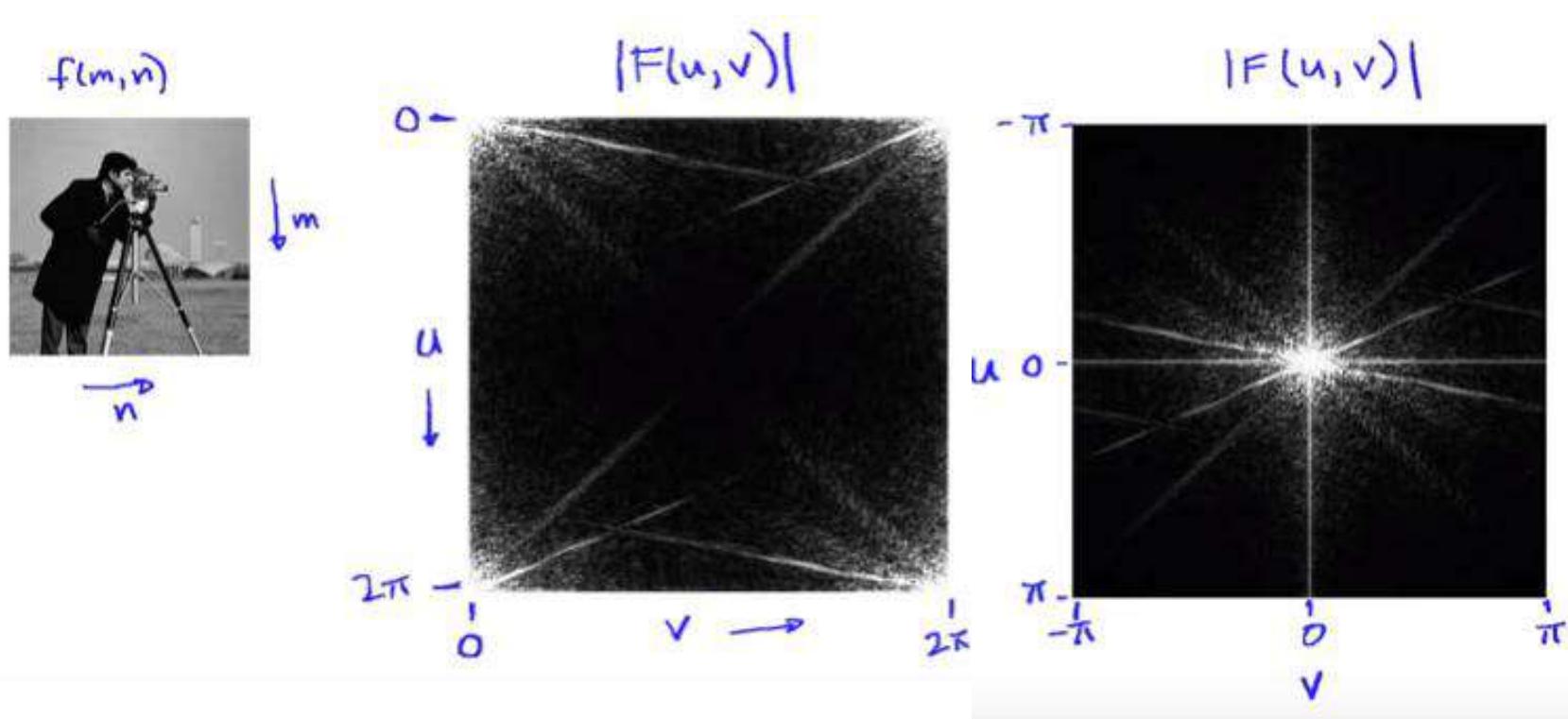
Shifting the origin

Frequency axis □



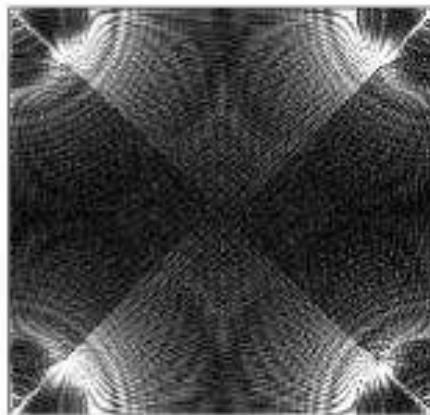
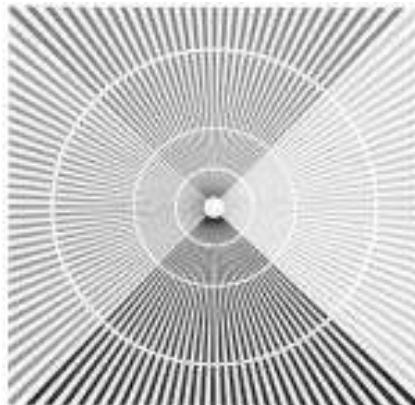
$$\Im[f(x, y)(-1)^{x+y}] = F(u - M/2, v - N/2)$$

Shifting the origin

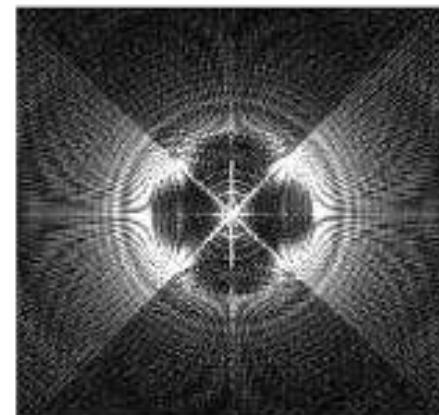


Shifting Example

Original ray image \mathbf{X}



choice 1: $\mathbf{Y} = \text{fft2}(\mathbf{X})$
Low-frequency at four corners



choice 2: $\mathbf{Y} = \text{fftshift}(\text{fft2}(\mathbf{X}))$
Low-frequency at the center

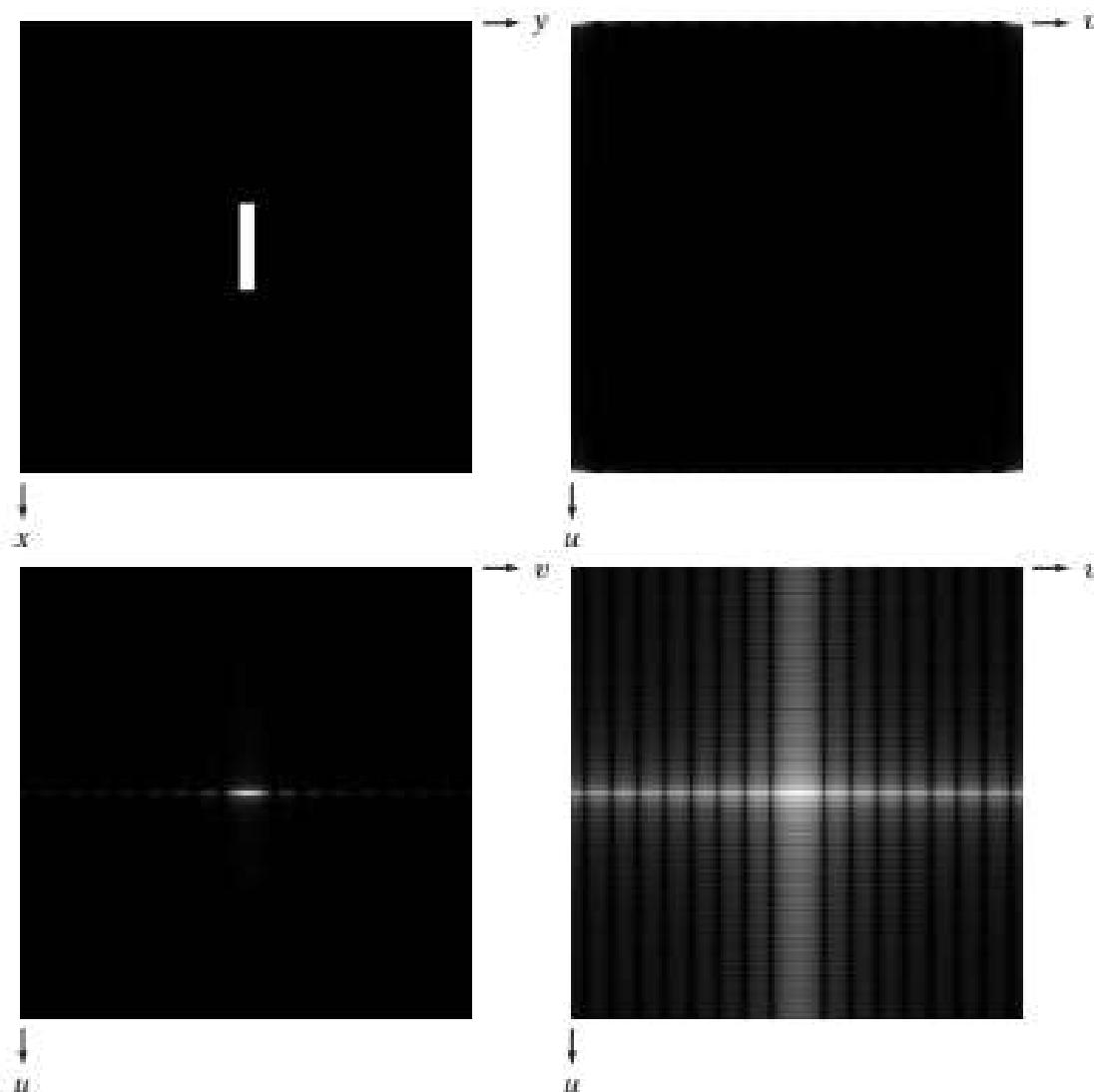
FFTSIFT Shift zero-frequency component to center of spectrum.

Shifting the origin

a b
c d

FIGURE 4.24

- (a) Image.
(b) Spectrum
showing bright spots
in the four corners.
(c) Centered
spectrum. (d) Result
showing increased
detail after a log
transformation. The
zero crossings of the
spectrum are closer in
the vertical direction
because the rectangle
in (a) is longer in that
direction. The
coordinate
convention used
throughout the book
places the origin of
the spatial and
frequency domains at
the top left.



Shifting the origin

$$\Im[f(x,y)(-1)^{x+y}] = F(u - M/2, v - N/2)$$



Fourier Transform This shifts the origin of $F(u,v)$ to $(M/2, N/2)$.

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



dc component

If $f(x,y)$ is an image, the value of DFT at the origin is equal to the average gray level of the image.

$$f(x,y) \text{ is real: } F(u,v) = F^*(-u,-v) \Leftrightarrow |F(u,v)| = |F(-u,-v)|$$

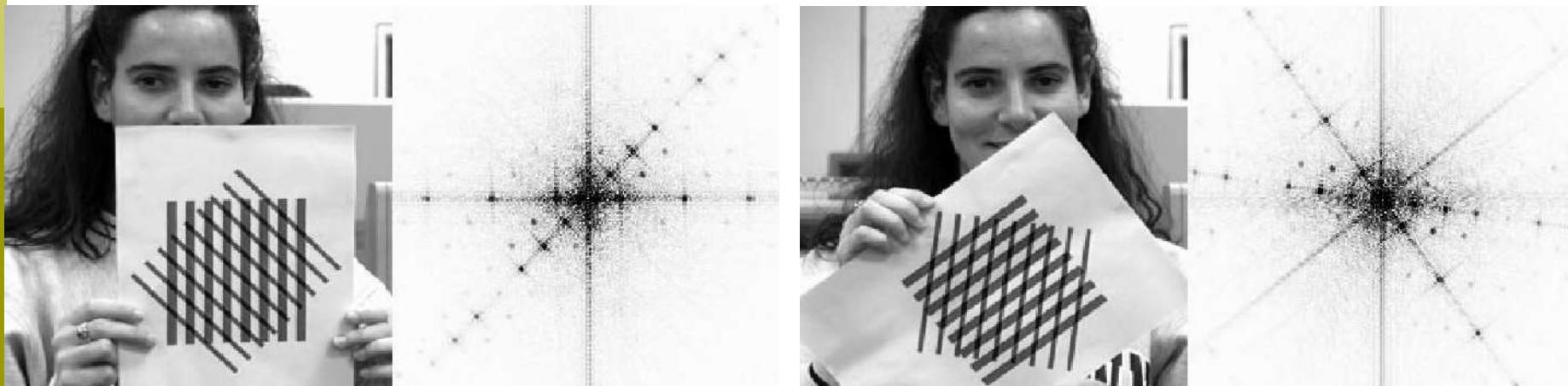


$$\Delta u = \frac{1}{M\Delta x} \quad \text{and} \quad \Delta v = \frac{1}{N\Delta y}$$

spectrum is symmetric

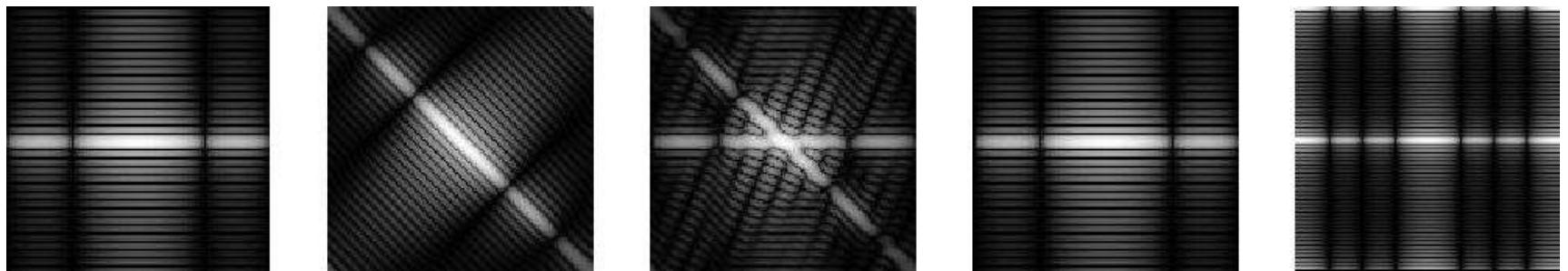
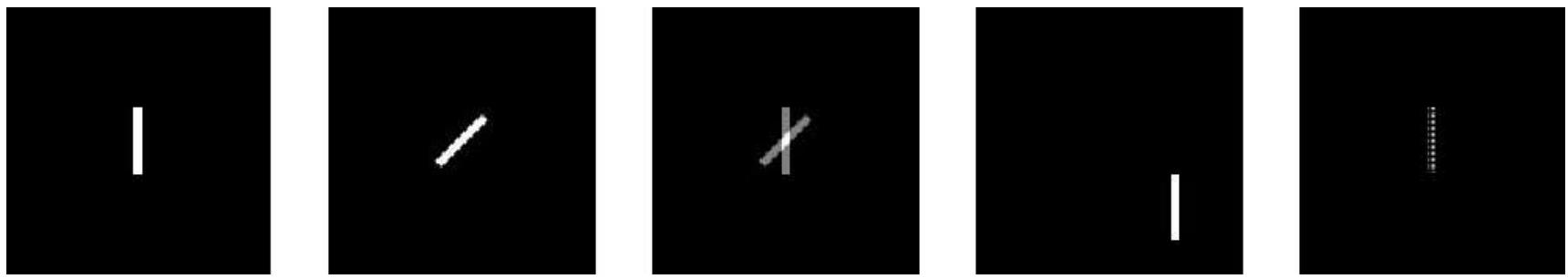
Properties of the Fourier Transform

- The edges on the image appears as point series in perpendicular direction in Fourier transform of the image and vice versa.



Properties of the Fourier Transform

Image-space



original

rotation

linearity

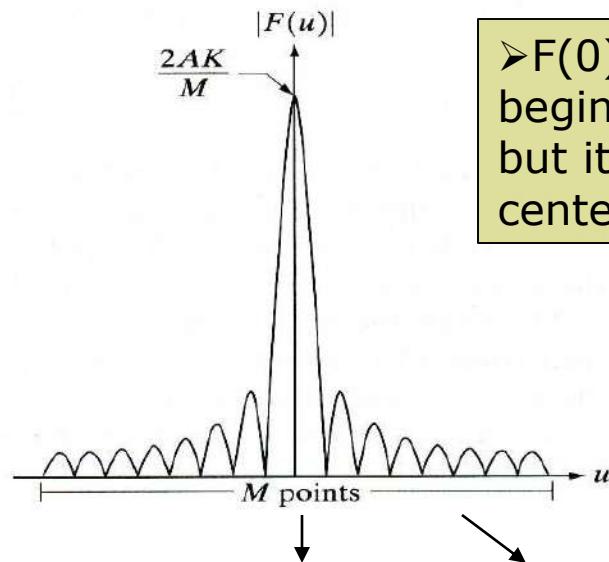
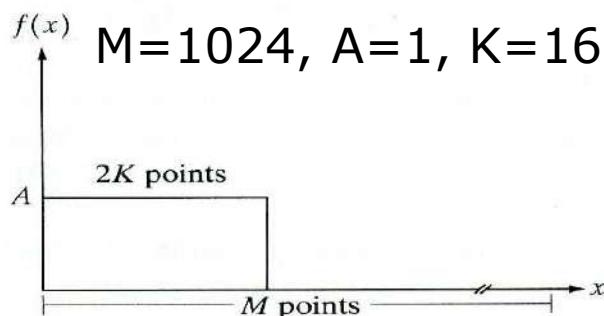
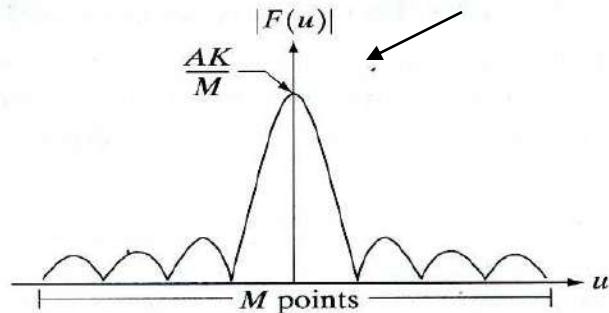
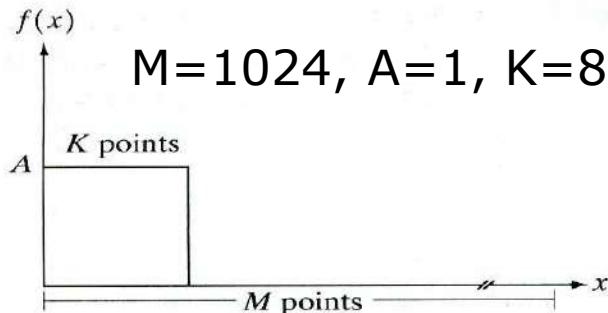
shift

scale

Frequency space

Example

$F(0) = \text{average of } f(x)$



lower frequency

Higher frequency

➤ $F(0)$ must appear at the beginning of the curve but it was shifted to the center

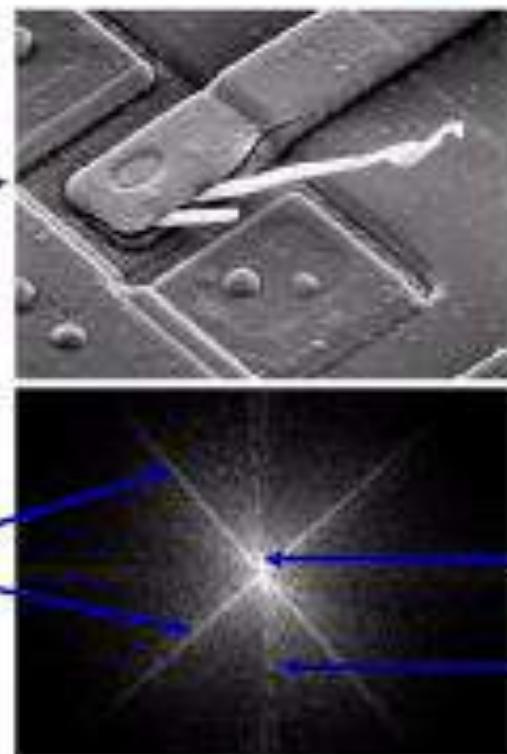
Fourier VS spatial characteristics

Frequency is directly related to the **rate of change**.

DFT components can be associated with patterns of **intensity variations** in an image.

2 principal features:

1. Strong edges that run approx. at $\pm 45^\circ$.
2. 2 white oxide protrusions.



Correspond to the strong edges.

Corresponds to the short protrusion.

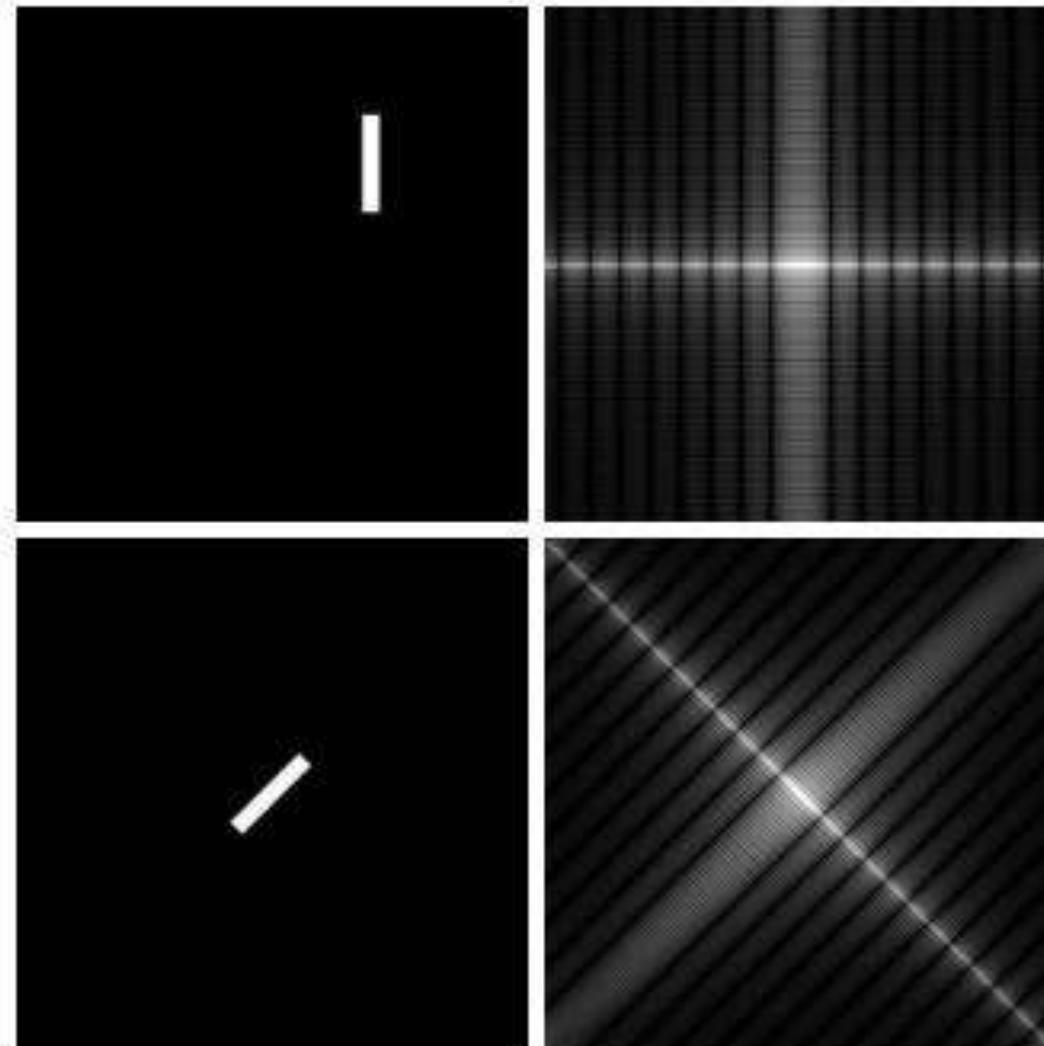
Corresponds to the long protrusion.



FIGURE 4.4
(a) SEM image of a dechamfered integrated circuit.
(b) Fourier spectrum of (a).
(Original image courtesy of Dr. I. M. Hudak,
Rensselaer Polytechnic Institute for
Materials Research, McGraw-Hill
University, Hamilton,
Ontario, Canada.)

- Try `fftshift` to do this in Matlab

Some properties of the 2-D DFT: Rotation & Translation



a b
c d

FIGURE 4.25
(a) The rectangle in Fig. 4.24(a) translated, and (b) the corresponding spectrum.
(c) Rotated rectangle, and (d) the corresponding spectrum. The spectrum corresponding to the translated rectangle is identical to the spectrum corresponding to the original image in Fig. 4.24(a).

Some properties of the 2-D DFT: Rotation

- Rotation by an angle in the SD causes rotation by same angle in the FD.

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0).$$

Some properties of the 2-D DFT: translation

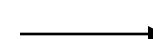
- Multiplication by complex exponentials in the spatial domain (SD) has the effect of translation in the FD.
- Frequency modulation concept.

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

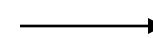
$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-j2\pi(ux_0/M + vy_0/N)},$$

How to convert from spatial to frequency in Matlab. (**fft2 /ifft2**)

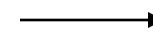
```
X= imread('cameraman.tif');  
Y= fft2(double(X));  
imshow(X);  
Figure, imshow(Y);
```



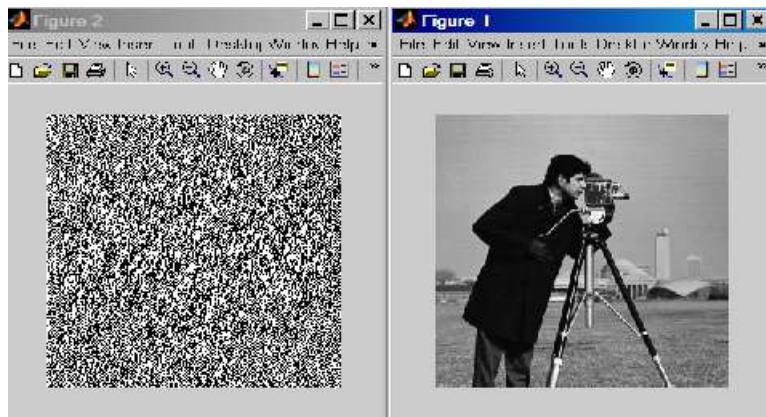
Read image in spatial



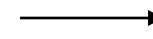
Convet image to Fourier



Display both the spatial & the Fourier versions from cameraman



```
z= uint8(ifft2(Y));  
Figure, imshow(z);
```



Convert image to spatial

How to get the magnitude and phase from the fouriour (**abs / angle**)

```
X= imread('cameraman.tif');  
Y= fft2(double(X));  
Mag= abs(Y);  
Phase = unwrap(angle(Y));
```

→ *Get the magnitude*
→ *Get the phase*

- You can reconstruct the image from mag alone or from phase alone??
- To do this use ifft2 not with y , but with mag , or with phase.
- but in both cases ,you will not get the original image.

Image Processing



Ch4:
Filtering in frequency domain
part2

Usefulness of FT Concepts:

- FT concepts are essential for understanding the **spectral** (FD) properties of signals and systems and have very wide applications such as filtering, image compression, image feature representation etc.

Frequency Domain Filtering & Spatial Domain Filtering

Similar jobs can be done in the spatial and frequency domains

Filtering in the spatial domain can be easier to understand

Filtering in the frequency domain can be much faster – especially for large images

Frequency Domain Filtering

- **Frequencies** are the amount by which grey values change with distance.
- **High frequency components** are characterized by large changes in grey values over small distances; (edges and noise)
- **Low frequency components** are parts characterized by little change in the gray values. (backgrounds, skin textures)
- **High pass filter**: if it “passes over” the high frequency components, and reduces or eliminates low frequency components.
- **Low pass filter**: if it “passes over” the low frequency components, and reduces or eliminates high frequency components.

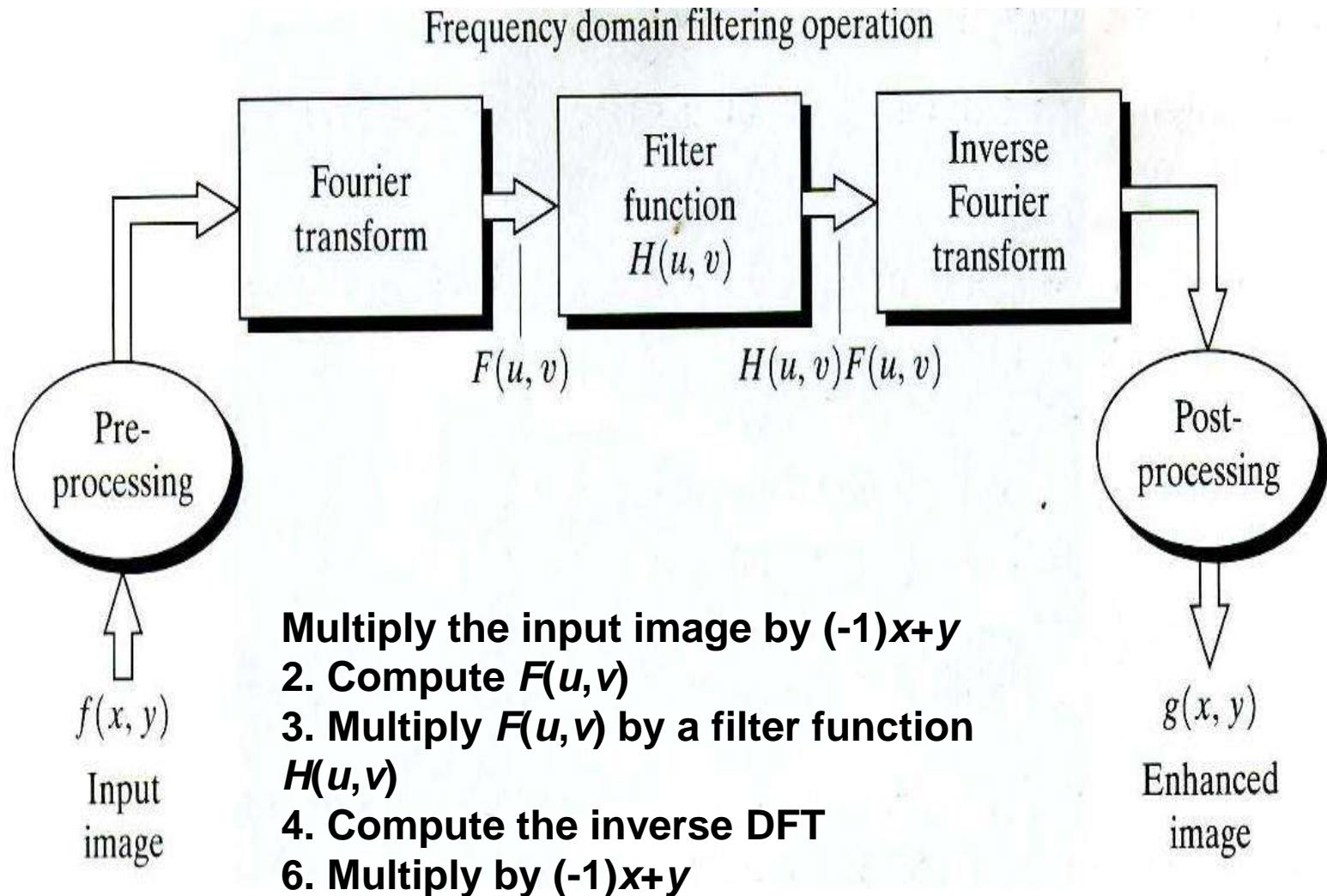
introduction

- It consists of modifying the FT of an input image and then finding the IFT to get the output image.
- Mathematically its given by:

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

Filtered Image = $\mathfrak{F}^{-1}[G(u, v)]$.

Steps of frequency domain filter



Correspondence between FD and SD filtering:

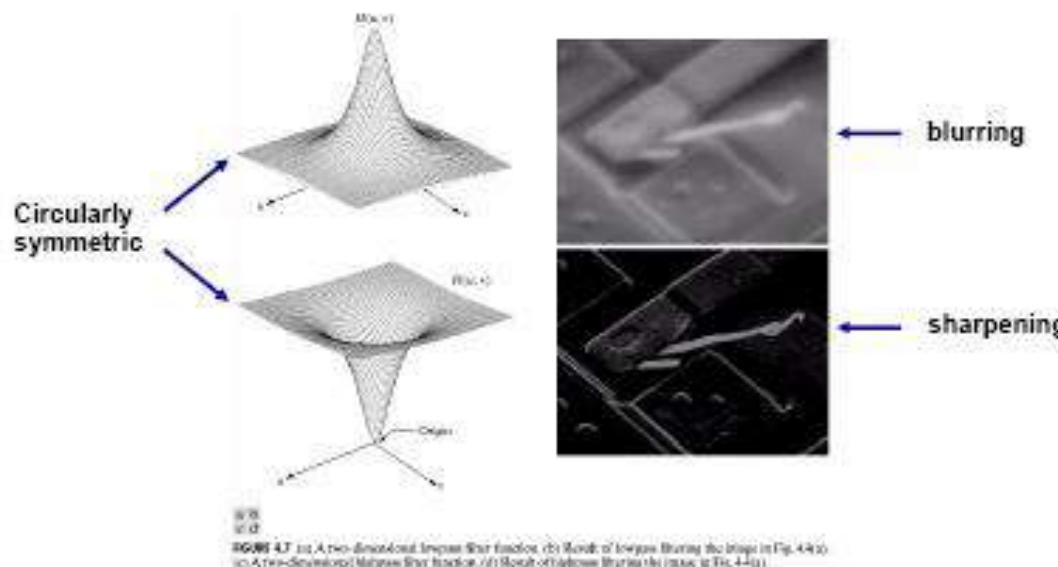
- SD: filtering is given by discrete convolution
- FD: Discrete convolution in FD is equivalent to multiplication

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

- Filtering can be done in either domain.

Low pass & High pass filters in FD

- ❑ **Low-pass filter:** A filter that attenuates high frequencies while passing low frequencies.- used for **blurring (smoothing)**
- ❑ **High-pass filter:** A filter that attenuates low frequencies while passing high frequencies. used for **sharpening**



Low pass & High pass filters in FD

1) Low pass - Smoothing filters

- 1.1) Ideal lowpass filters (very sharp)
- 1.2) Butterworth lowpass filters
- 1.3) Gaussian lowpass filters (very smooth)

More smooth
in the edge of
cut-off
frequency

- ▣ Butterworth filter parameter: filter order
- ▣ High values: filter has the form of the ideal filter.
- ▣ Low values: filter has the form of the Gaussian filter.

2) High pass- Sharpening filters

- 2.1) Ideal highpass filters
- 2.2) Butterworth highpass filters
- 2.3) Gaussian highpass filters

1) Low pass – *image smoothing*

Low-pass filter: A filter that attenuates high frequencies while passing low frequencies.- used for **blurring (smoothing)**

Image smoothing using FD filters

- Noise is usually high frequency.
- Image details and edges have high frequency characteristics.
- Hence noise and details removal is usually termed smoothing or blurring.
- Smoothing is achieved by lowpass filters (LPFs).
- **3 types** of LPFs will be studied i.e. Ideal LPF, Butterworth LPF and Gaussian LPF.

1) Low pass - *image Smoothing*

Smoothing filters:

1.1) Ideal low pass filters (ILPF)

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

1.2) Butterworth low pass filters (BLPF)

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

1.3) Gaussian low pass filters (GLPF)

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

1.1) 2-D Ideal low pass filter ILPF

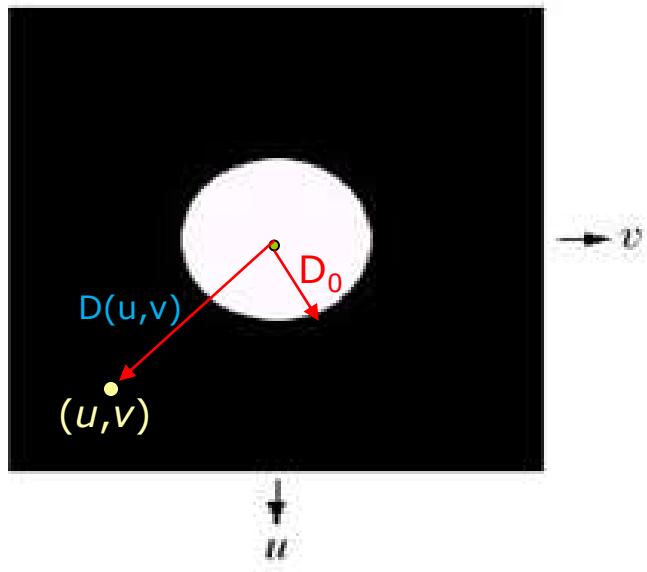
The simplest lowpass filter is ILPF, it (cuts off) all high frequency components that are at distance greater than a specified distance: D_0 from the center

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

Where

- ◻ $D(u, v)$ is the distance from (u, v) to the center of the frequency rectangle
- ◻ Image size: $M \times N$
- ◻ Center of the frequency rectangle: $(u, v) = (M/2, N/2)$
- ◻ Distance to the center: $D(u, v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$

$H(u,v)$



Example on 2-D Ideal
low pass filter ILPF

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



Center of the frequency rectangle:

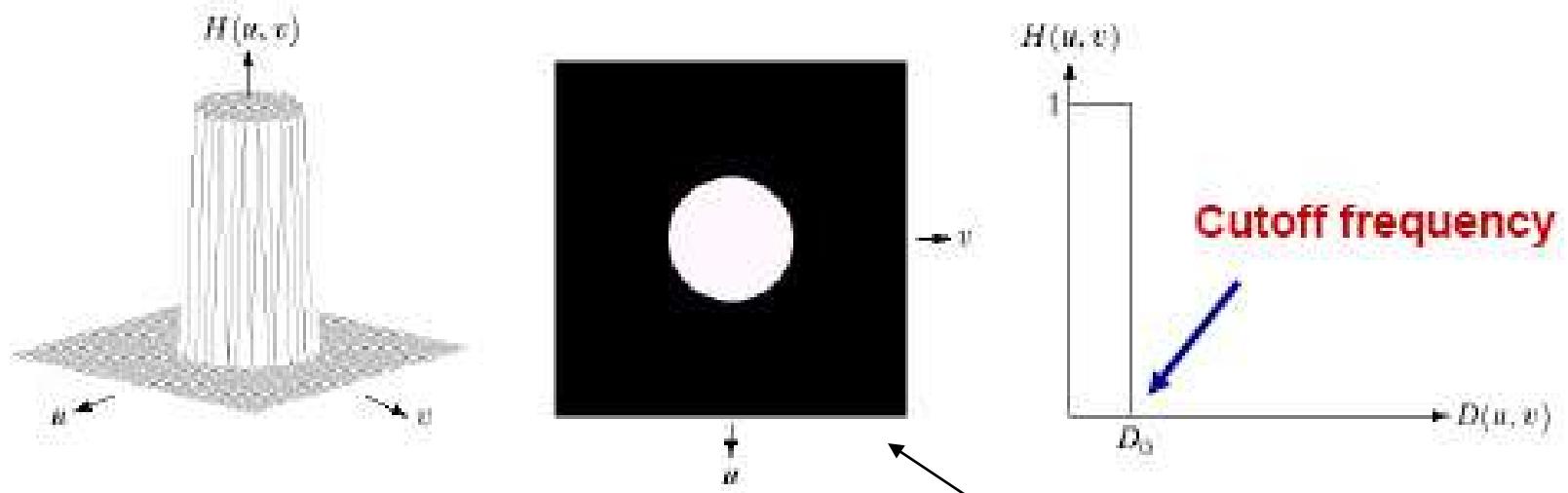
$$(M/2, N/2) = (5,5)$$

Suppose $D_0 = 3$

$$\begin{aligned} \text{The value of } H1 &= [(5-5)^2 + (5-7)^2]^{1/2} \\ &= 2 < 3 \text{ then } H1 = 1 \end{aligned}$$

$$\begin{aligned} \text{The value of } H2 &= [(5-9)^2 + (5-2)^2]^{1/2} \\ &= 5 > 3 \text{ then } H2 = 0 \end{aligned}$$

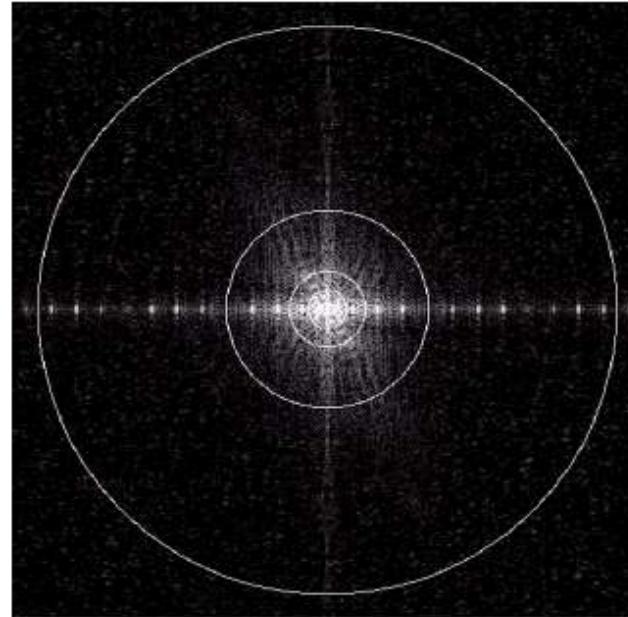
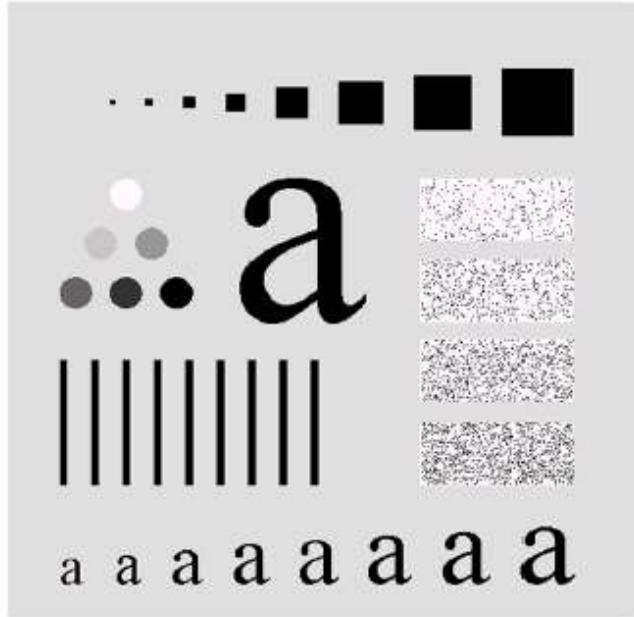
How does ILPF function works?



• ضرب قيم هذا الماسك $H(u, v)$ بالقيم المكافئة لها بالصورة $F(u, v)$. فنحصل فقط على ال frequencies داخل الدائرة

• دائمًا منتصف الصورة تحمل اهم المعلومات عنها (اي معظم معلومات الصورة), بينما تتوزع التفاصيل

• كل ما يكبر نصف قطر الدائرة (اي D_0), كل ما حصلنا على صورة اقرب للاصل. (انظر الشريحة التالية)



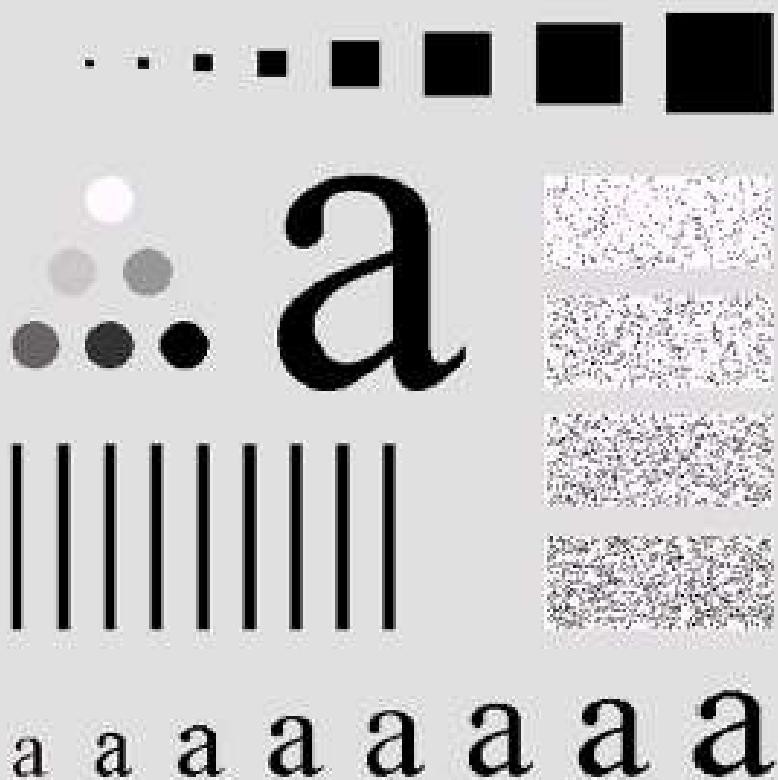
كل ما أكبر نصف قطر الدائرة (اي D_0),
كل ما حصلنا على صورة اقرب للاصل

<u>Radius</u>	<u>total image power %</u>
5	92.0
15	94.6
30	96.4
80	98
230 pixels	99.5

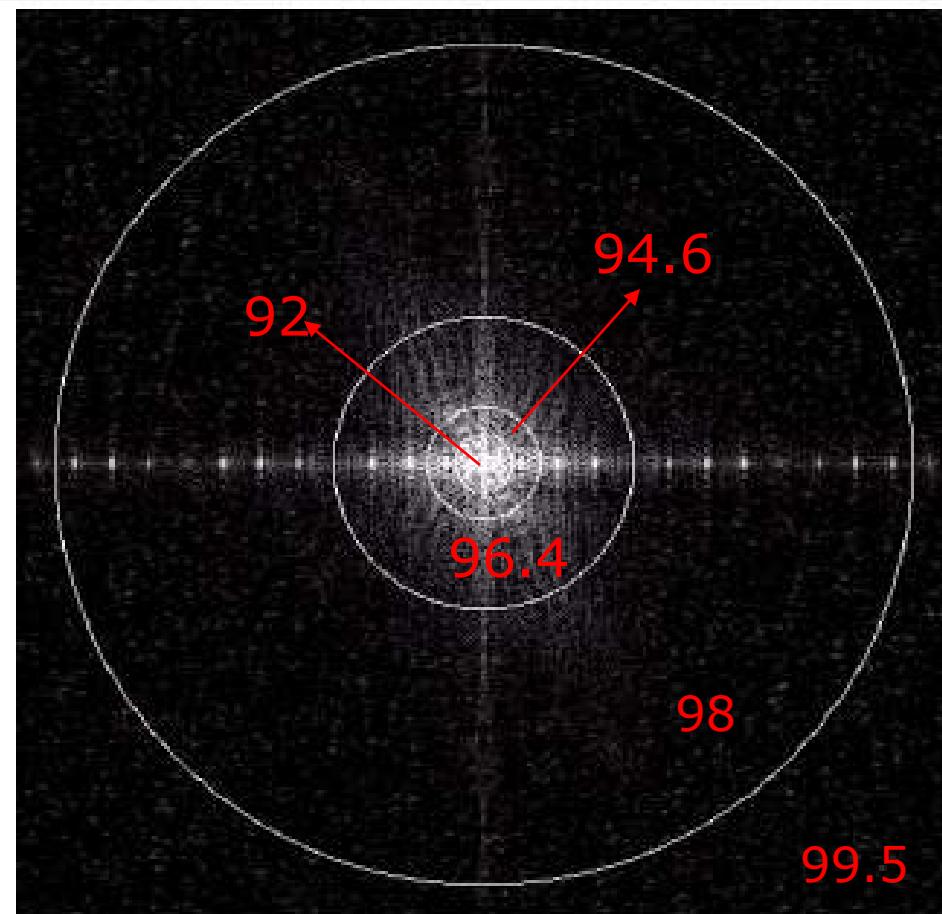
دائماً منتصف الصورة تحمل اهم المعلومات عنها (اي معظم معلومات الصورة)، بينما تتوزع التفاصيل

ILPF: example

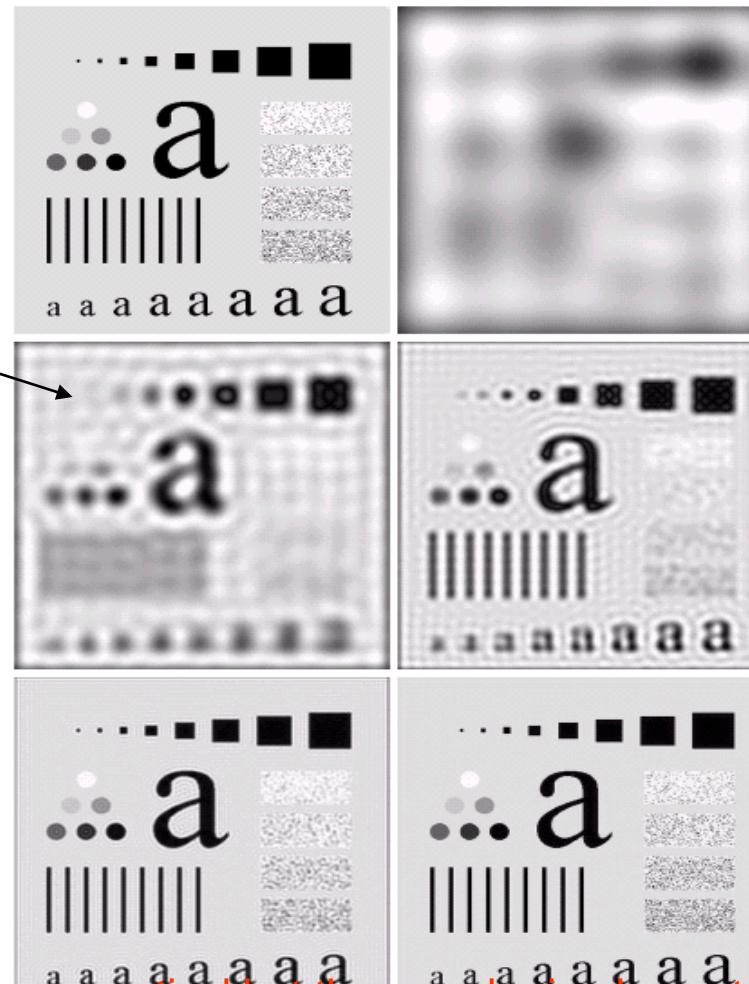
original



Freq.



Ideal Low Pass Filter results



Original image

Ringing is a characteristic of ideal Filter

Result of filtering with ideal low pass filter of radius 15

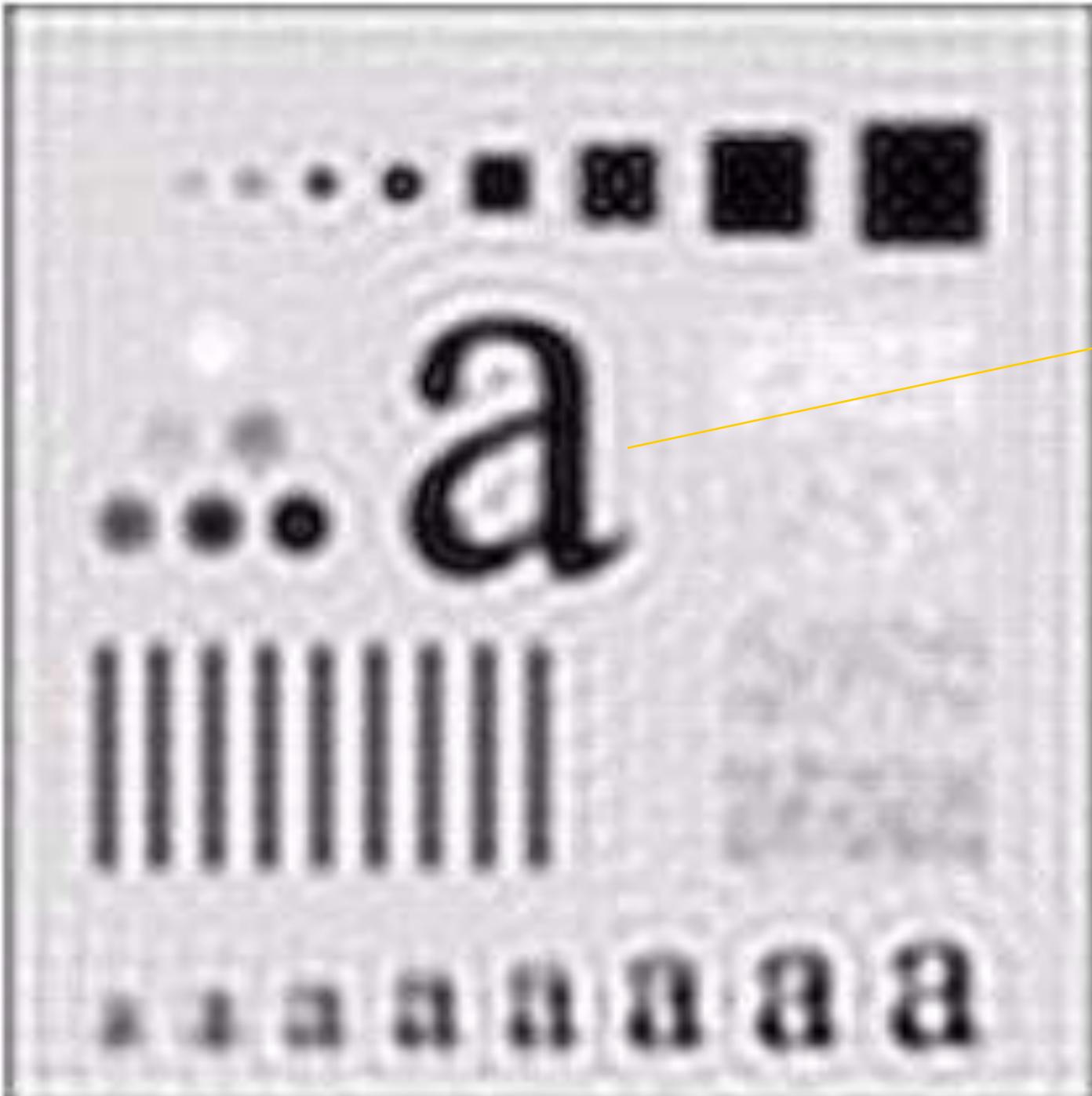
Result of filtering with ideal low pass filter of radius 80

Result of filtering with ideal low pass filter of radius 5

Result of filtering with ideal low pass filter of radius 30

Result of filtering with ideal low pass filter of radius 230

Ideal lowpass filtering is not very practical but they can be implemented on a computer to study their behavior.¹⁷

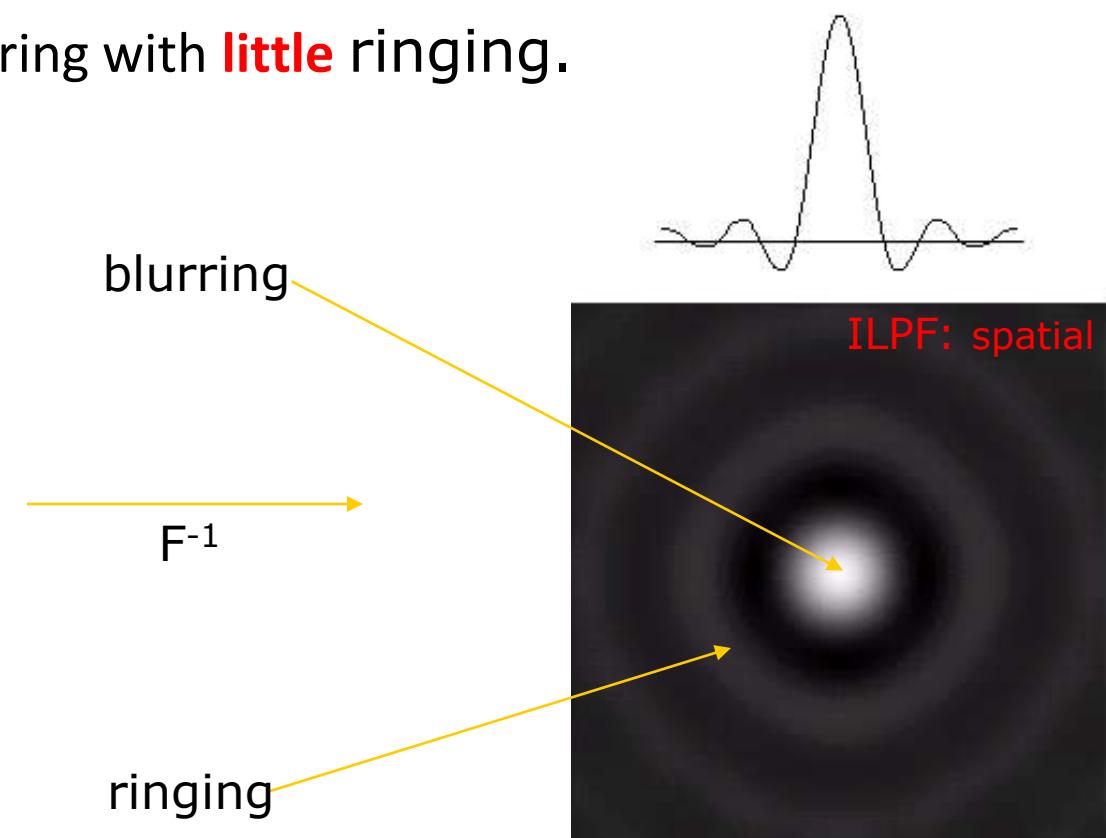
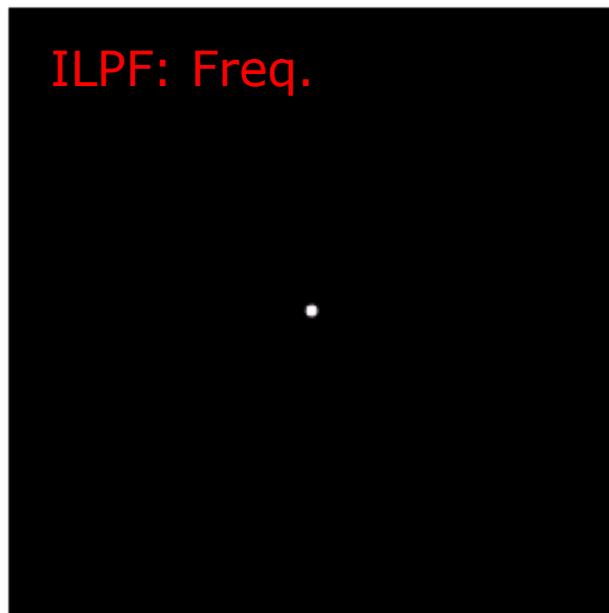


Ringing
effect

Blurring and ringing feature of ILPFs:

The ILPF has **sharp** cutoffs or discontinuities which cause ringing.

- The center of lobe is the cause for blurring **but** the outer smaller lobes cause ringing.
- We want to achieve blurring with **little** ringing.



1.2) Butterworth low pass filter - BLPF:

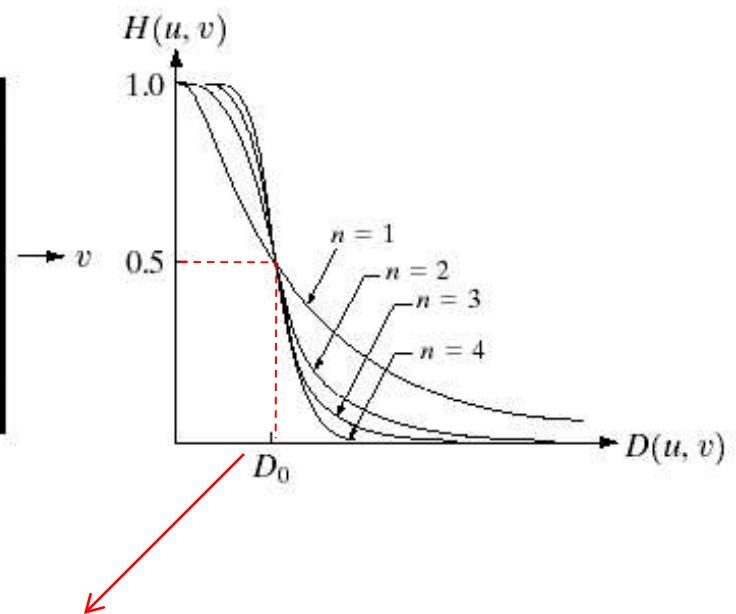
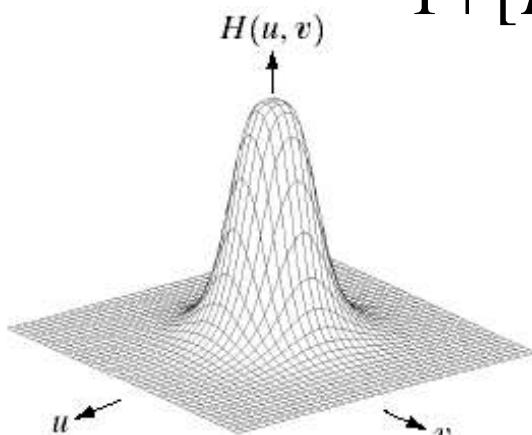
- ILPF transfer function has sharp discontinuities.
- BLPF transfer function **does not** have **sharp** discontinuities in order to **reduce** ringing.

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

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

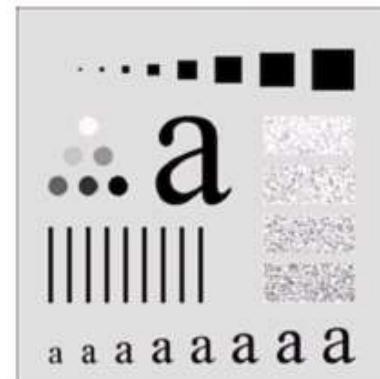
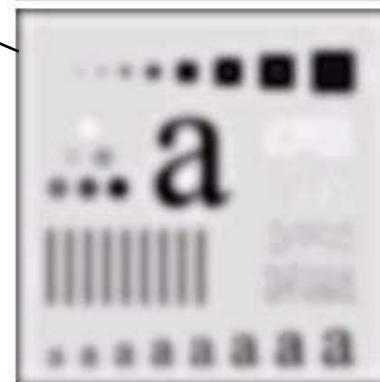
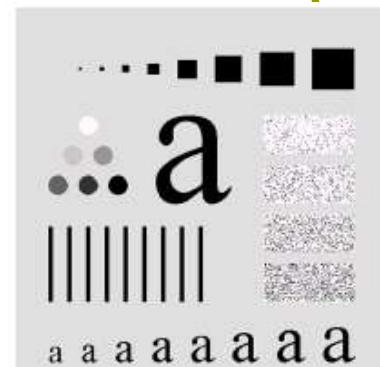
Butterworth low pass filter - BLPF:

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



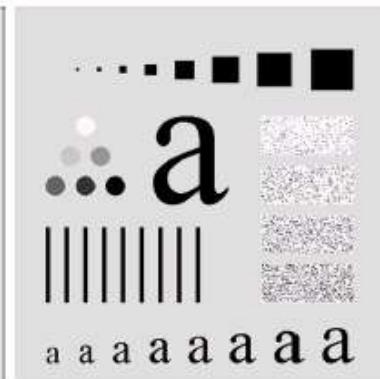
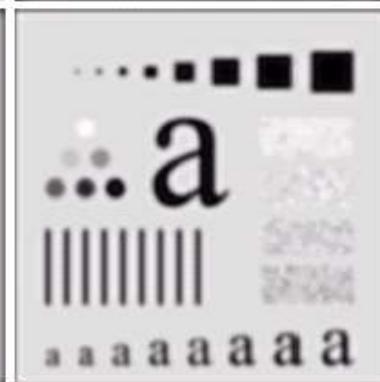
Butterworth Lowpass Filter (cont...)

Original image



Ringing is not visible in
any of these images.

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



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

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

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

Ringing increases with filter order (n) as seen in SD

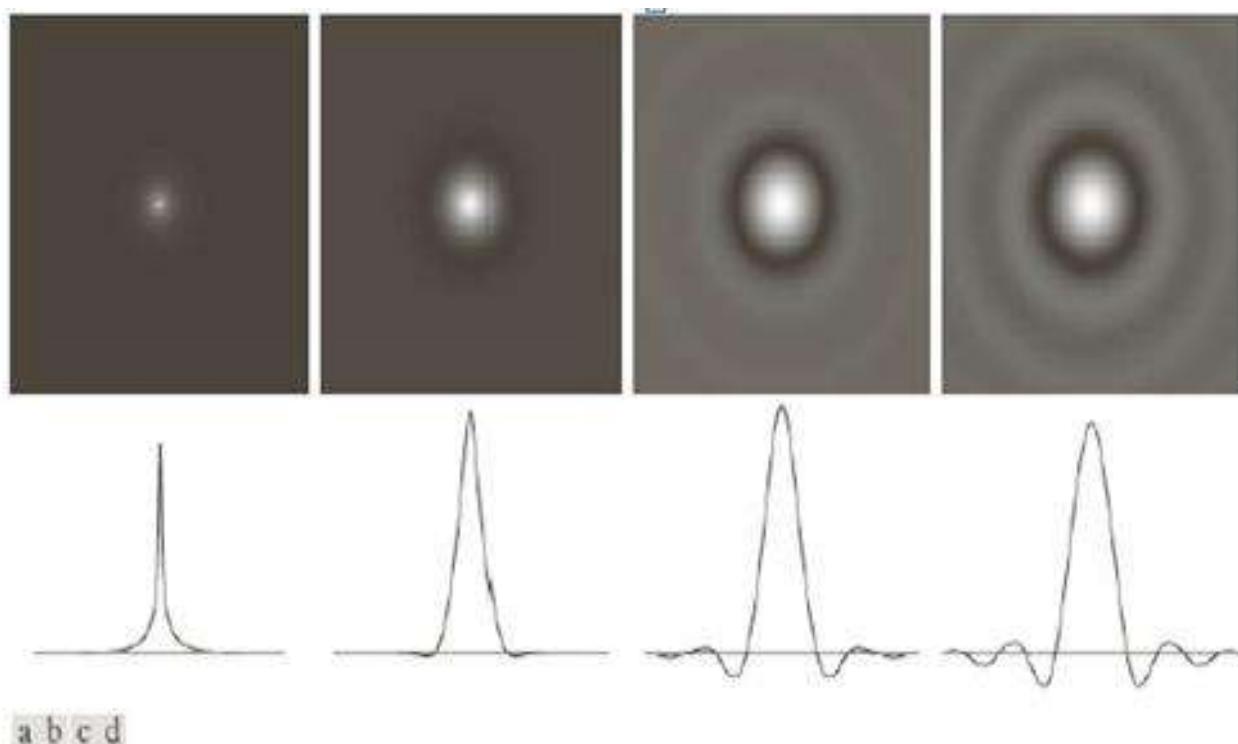
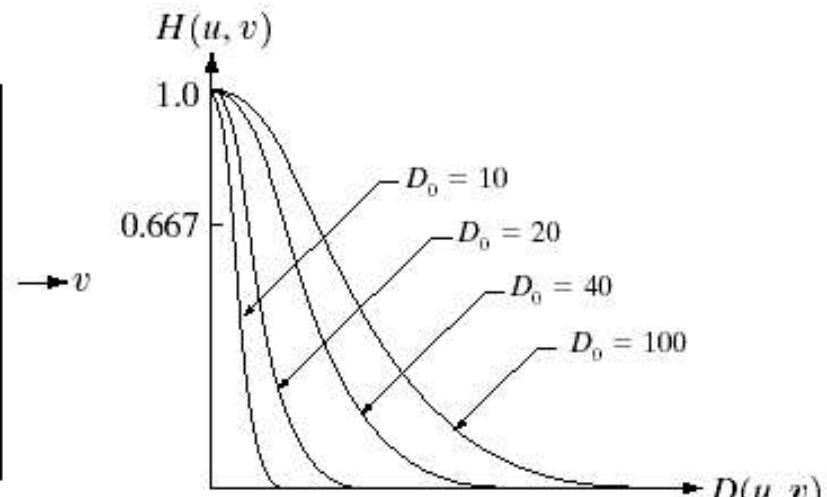
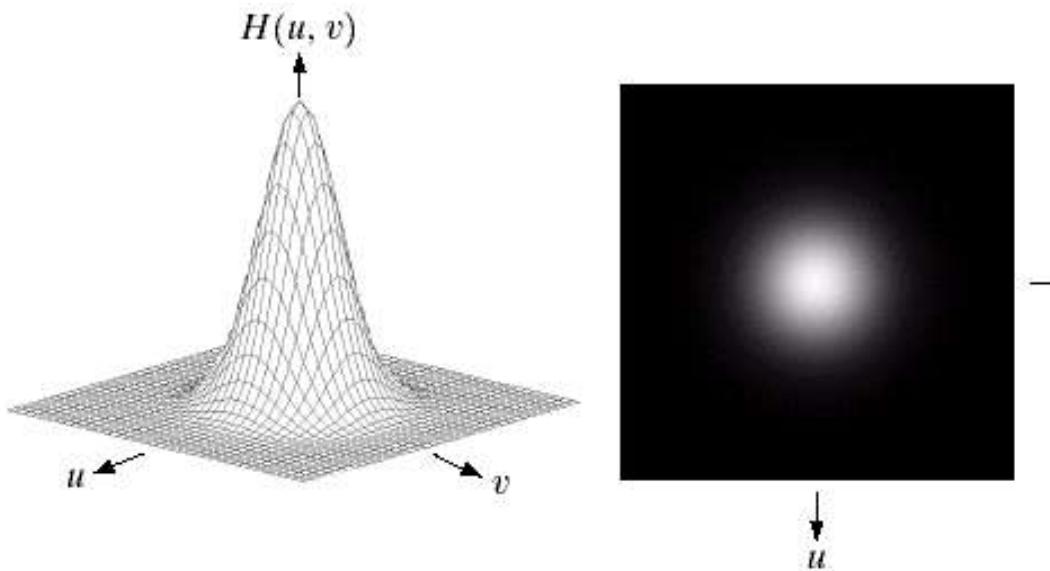


FIGURE 4.46 (a)-(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding intensity profiles through the center of the filters (the size in all cases is 1000×1000 and the cutoff frequency is 5). Observe how ringing increases as a function of filter order.

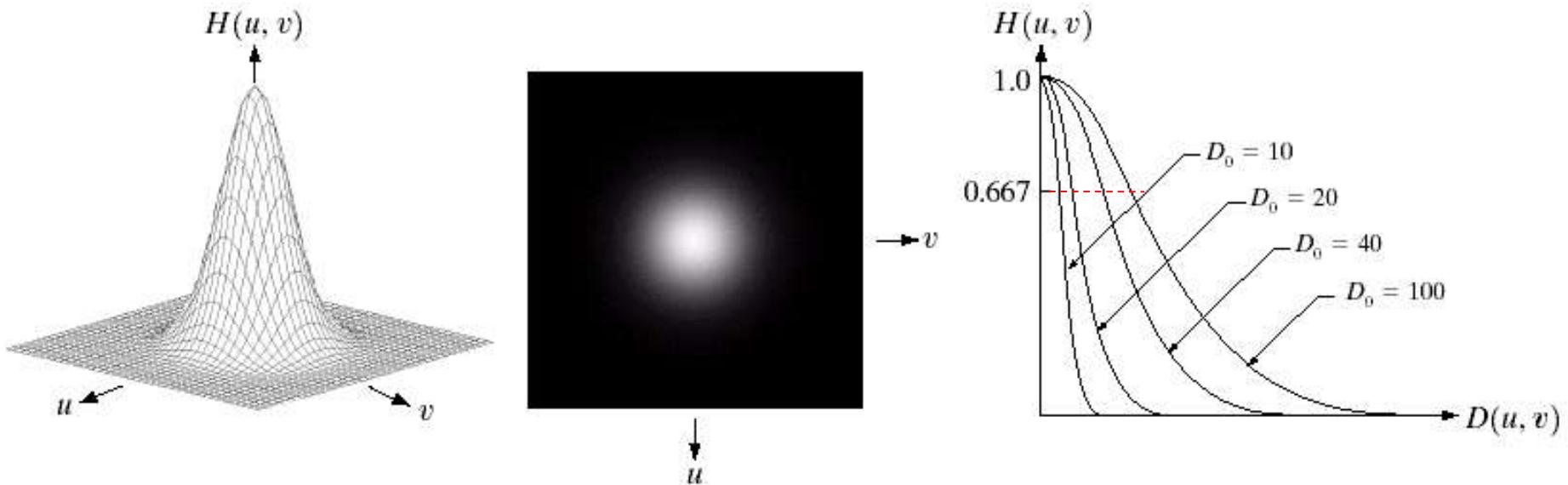
1.3) Gaussian Lowpass Filters -GLPF

The transfer function of a Gaussian lowpass filter is defined as

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



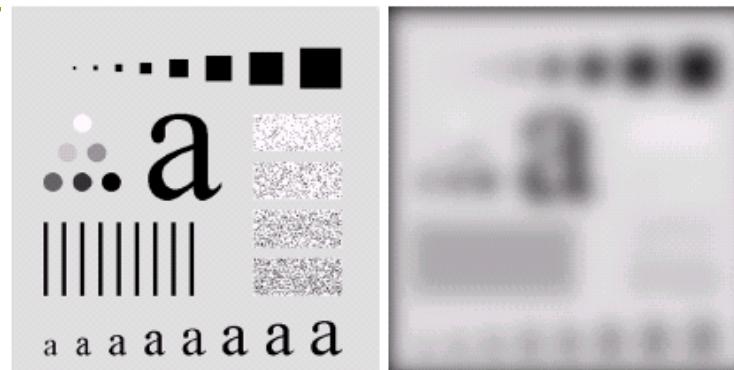
Gaussian low pass filter - GLPF:



Gaussian Lowpass Filters (cont...)

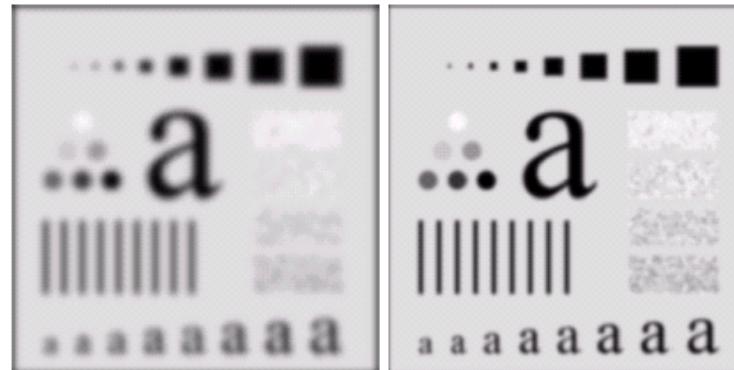
No ringing

Original image



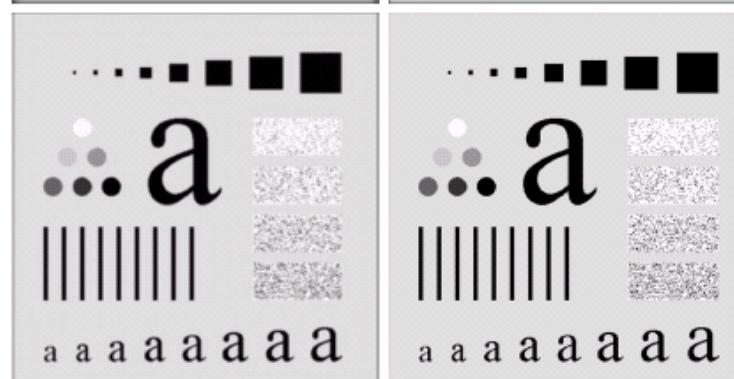
Result of filtering with Gaussian filter with cutoff radius 5

Result of filtering with Gaussian filter with cutoff radius 15



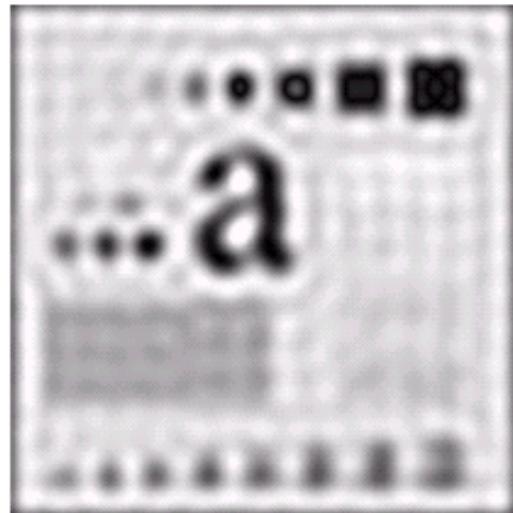
Result of filtering with Gaussian filter with cutoff radius 30

Result of filtering with Gaussian filter with cutoff radius 85

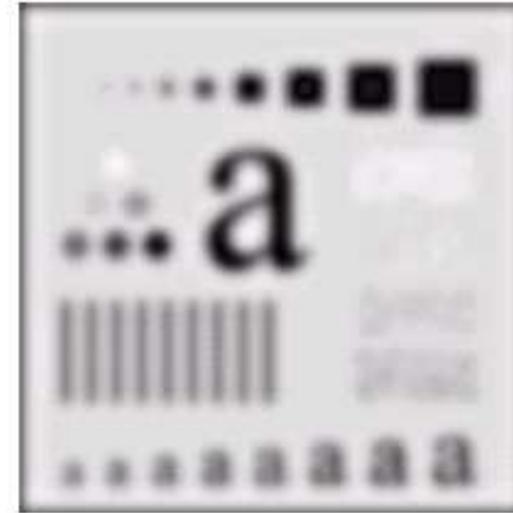


Lowpass Filters Compared

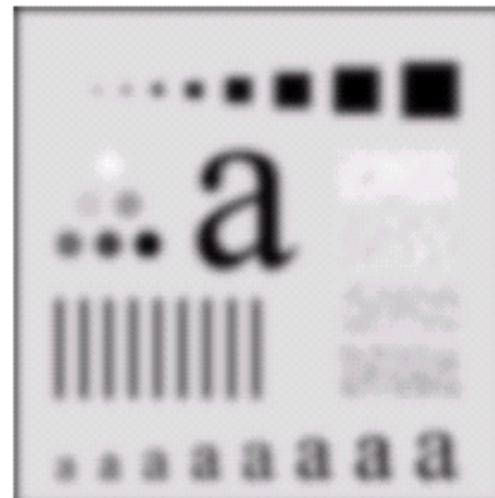
Result of
filtering with
ideal low pass
filter of radius
15



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



Result of
filtering with
Gaussian filter
with cutoff
radius 15



Practical applications

A lowpass Gaussian filter is used to connect broken text

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.



Machine recognition systems have difficulty in reading broken characters.

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.



GLPF with $D_0 = 80$

Practical applications:

Different Lowpass Gaussian filters used
to remove blemishes in a photograph



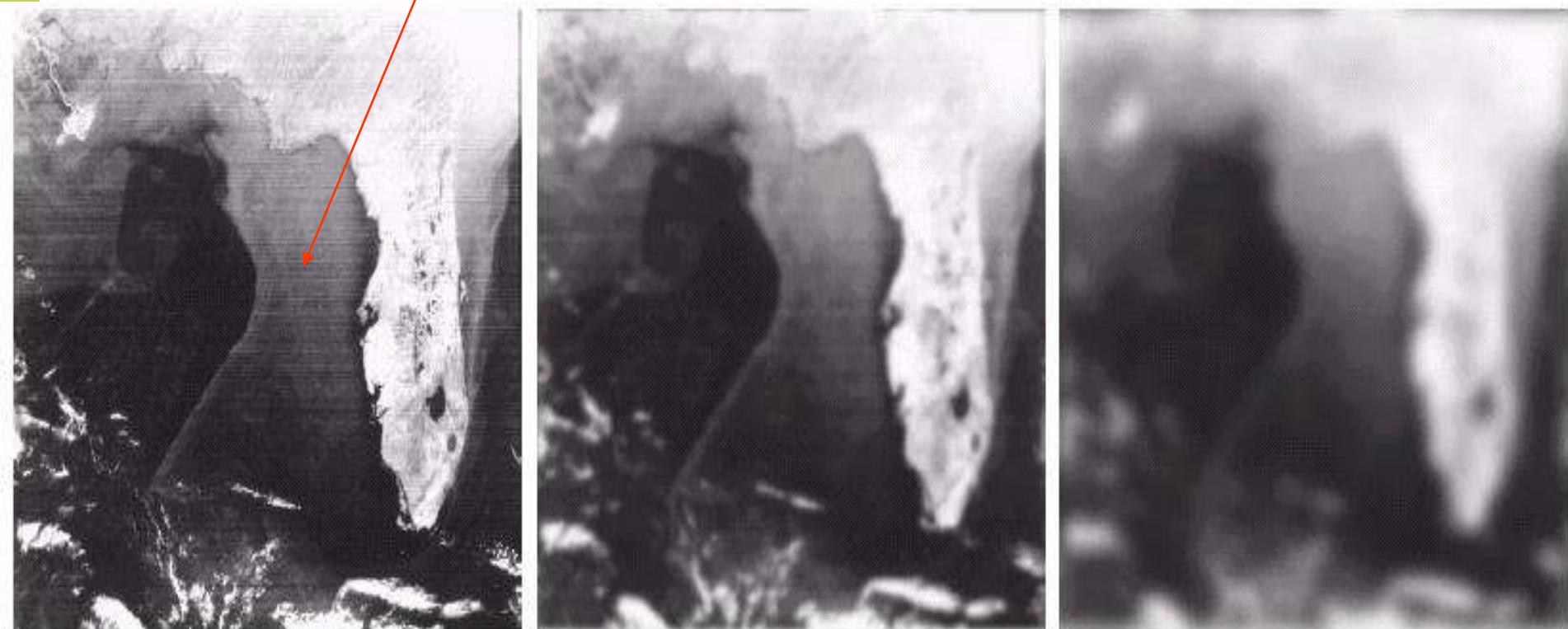
Practical applications:

588x600

Scan line

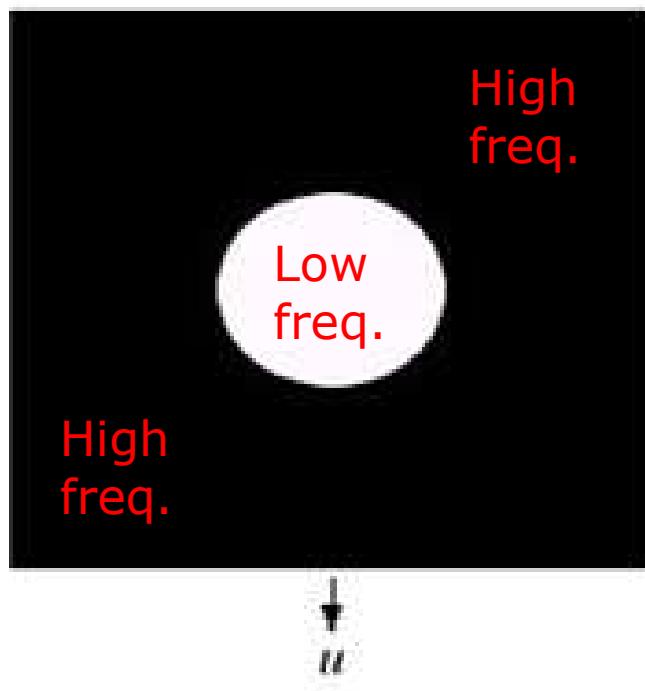
GLPF, $D_0=30$

GLPF, $D_0=10$

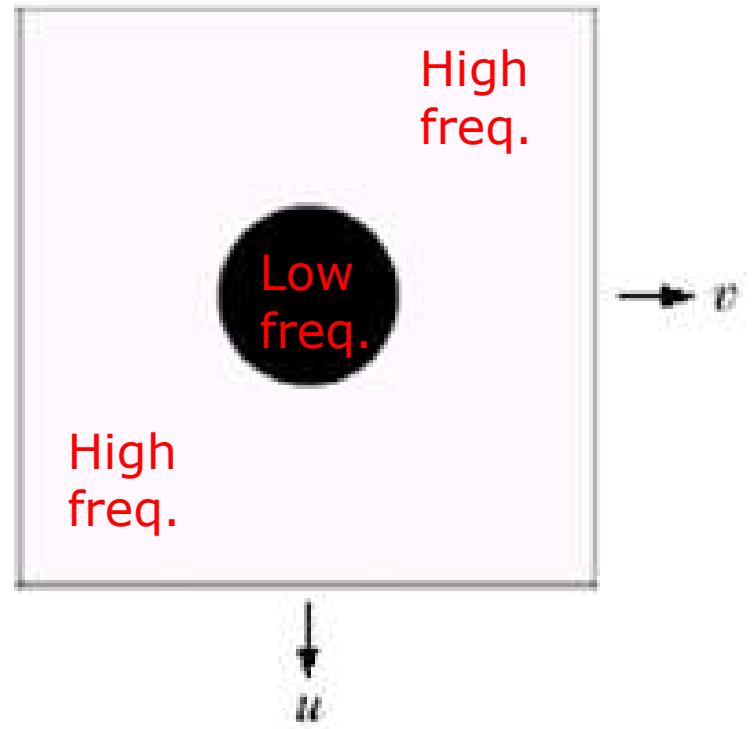


2) High pass - *image Sharpening*

Low-pass filter



High-pass filter



2) High pass - *image Sharpening*

- ❑ Blurring (smoothing) is achieved by attenuating the **HF**(high frequency) components of DFT of an image. (low pass filters)
- ❑ Sharpening is achieved by attenuating the **LF** (low frequency) components of DFT of an image. Where Edges and fine detail in images are associated with high frequency components (high pass filters)

High pass filters – only pass the high frequencies, drop the low ones. High pass frequencies are precisely the reverse of low pass filters, so:

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

2) High pass - *image Sharpening*

Sharpening filters:

2.1) Ideal high pass filters (IHPF)

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

2.2) Butterworth high pass filters (BHPF)

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

2.3) Gaussian high pass filters (GHPF)

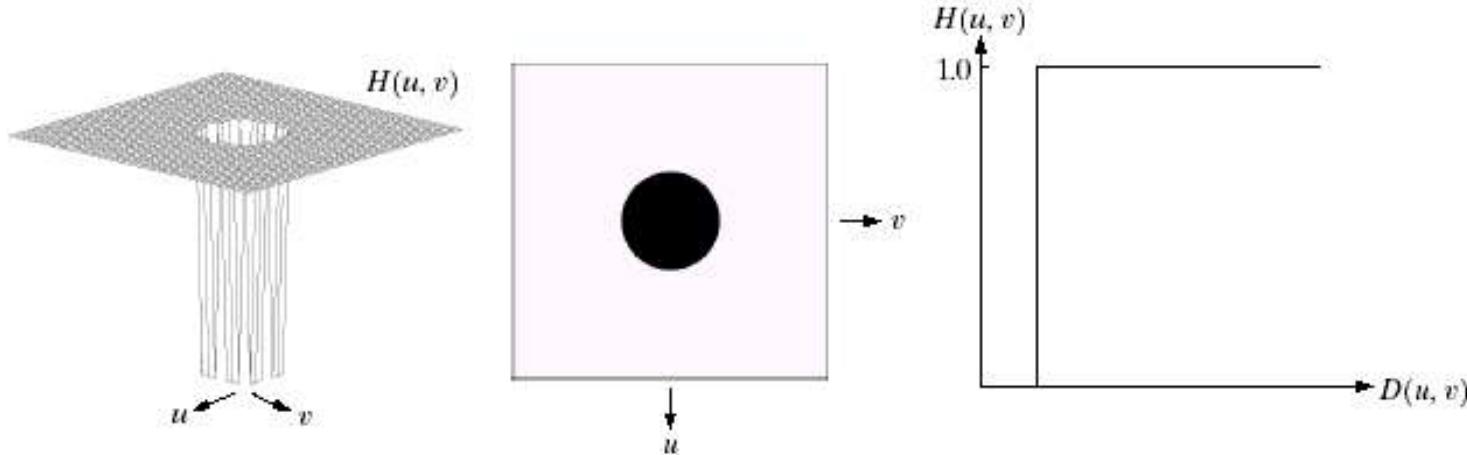
$$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2}$$

2.1) Ideal High Pass Filters

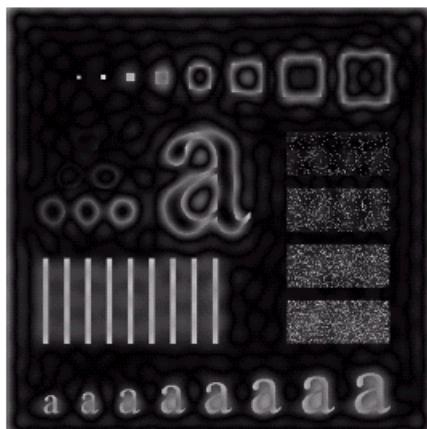
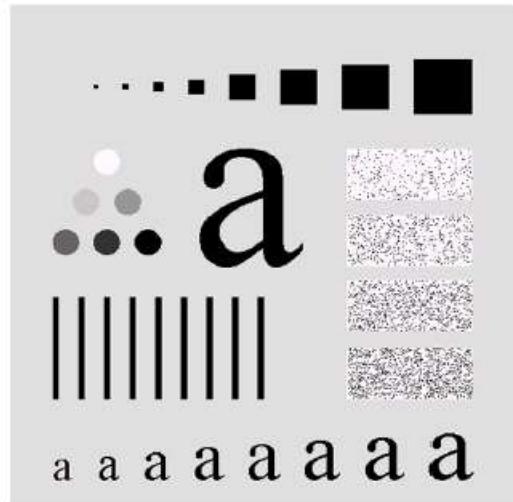
The ideal high pass filter is given as:

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

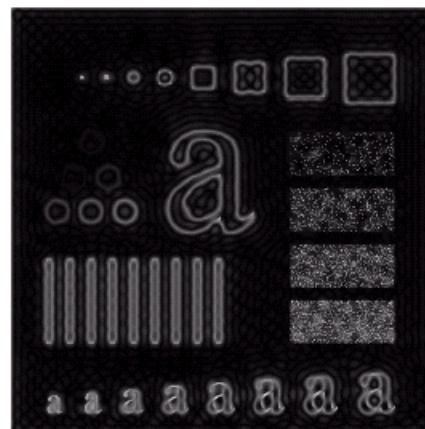
where D_0 is the cut off distance as before



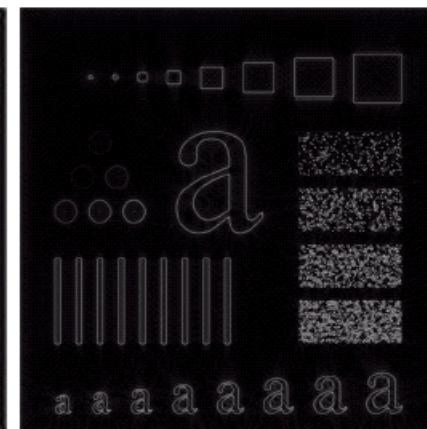
Ideal High Pass Filters (cont...)



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



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



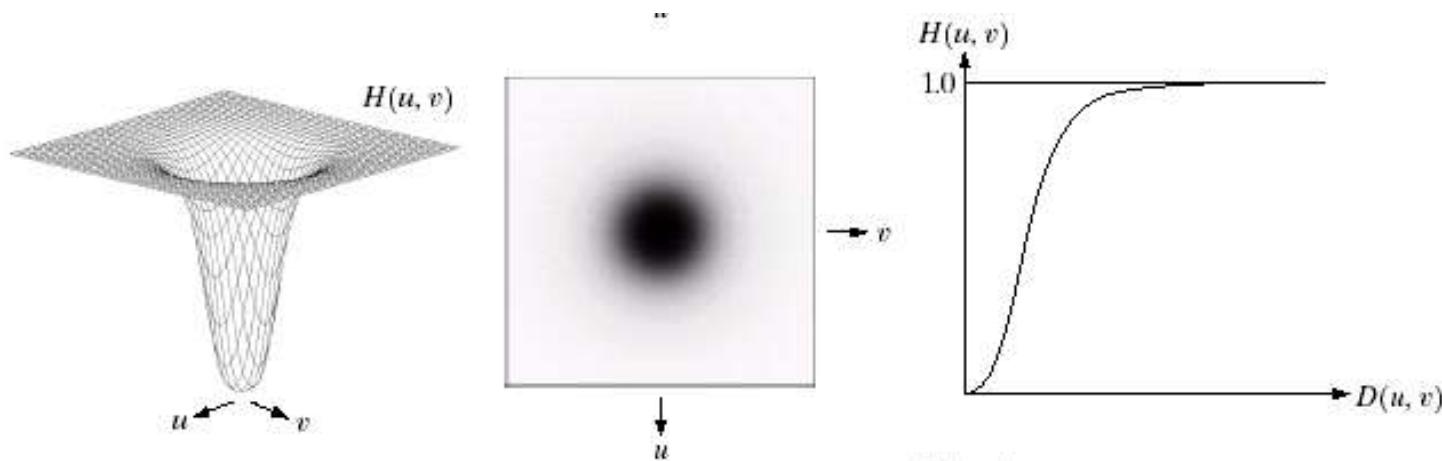
Results of ideal
high pass filtering
with $D_o = 80$

Butterworth High Pass Filters

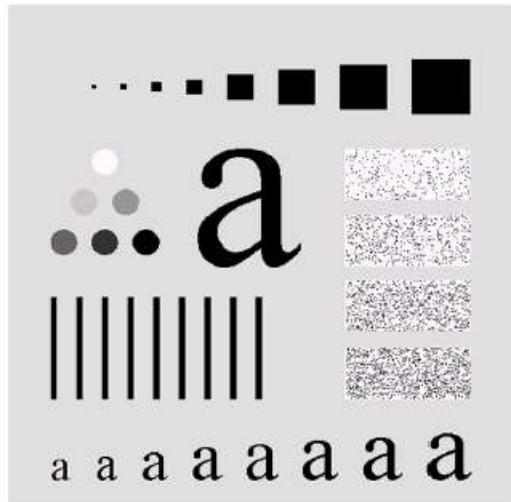
The Butterworth high pass filter is given as:

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

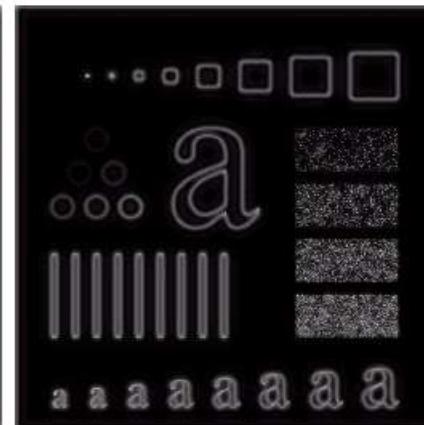
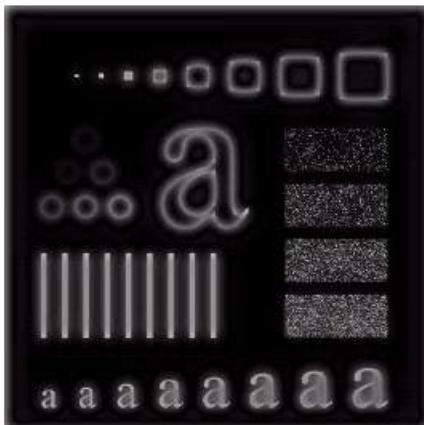
where n is the order and D_0 is the cut off distance as before



Butterworth High Pass Filters (cont...)

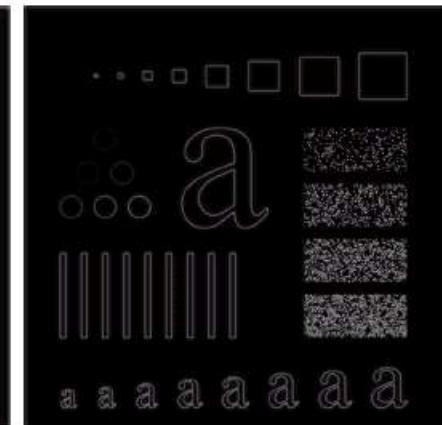


Results of
Butterworth
high pass
filtering of
order 2 with
 $D_o = 15$



Results of Butterworth high pass
filtering of order 2 with $D_o = 30$

Results of
Butterworth
high pass
filtering of
order 2 with
 $D_o = 80$

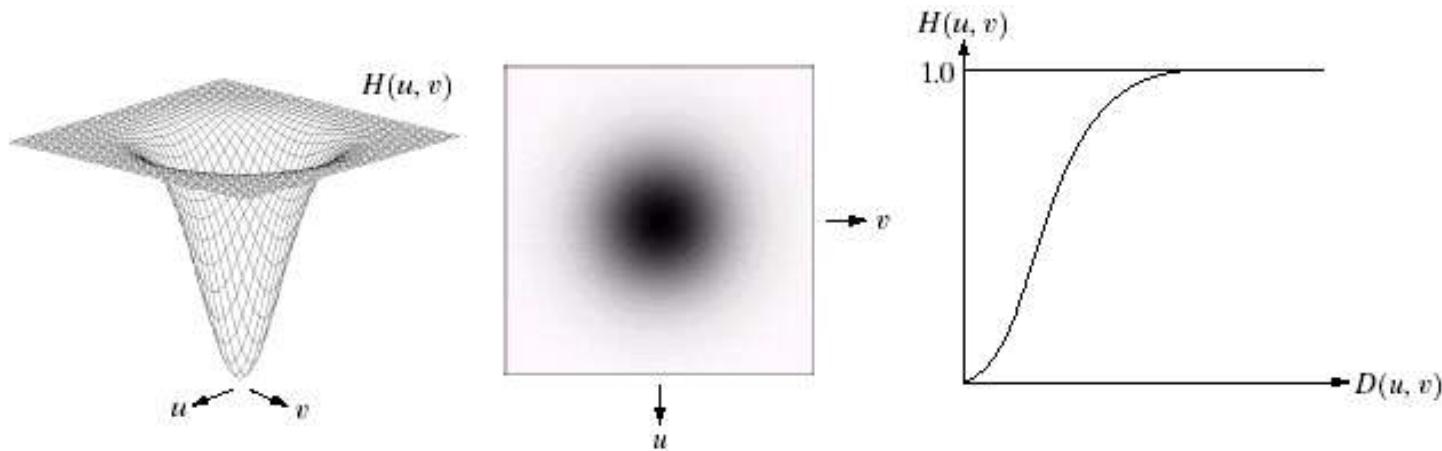


Gaussian High Pass Filters

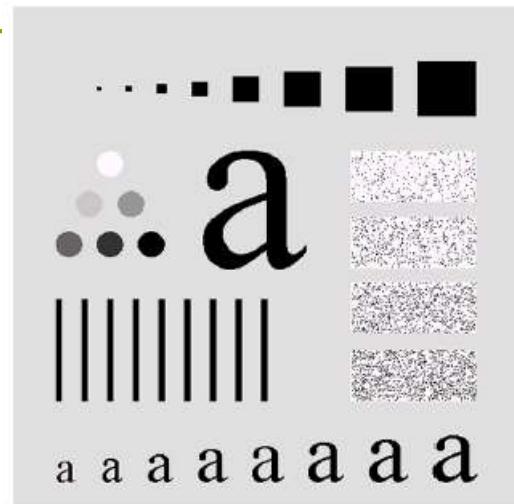
The Gaussian high pass filter is given as:

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

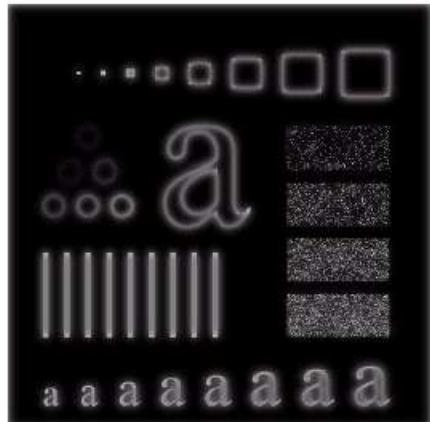
where D_0 is the cut off distance as before



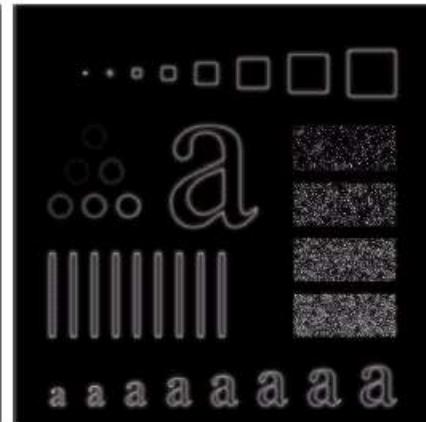
Gaussian High Pass Filters (cont...)



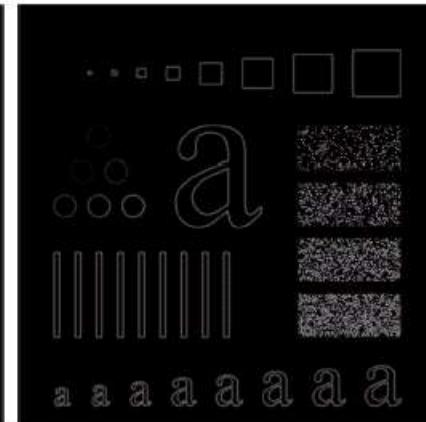
Results of
Gaussian
high pass
filtering with
 $D_0 = 15$



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



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



Highpass Filter Comparison

BF represents a transition between the sharpness of the **IF** and the smoothness of the **GF**

يعني يعتبر BF مرحلة وسطية بين IF و GF
حيث يعتبر IF حاد للغاية بينما يمتاز GF
بنعومته

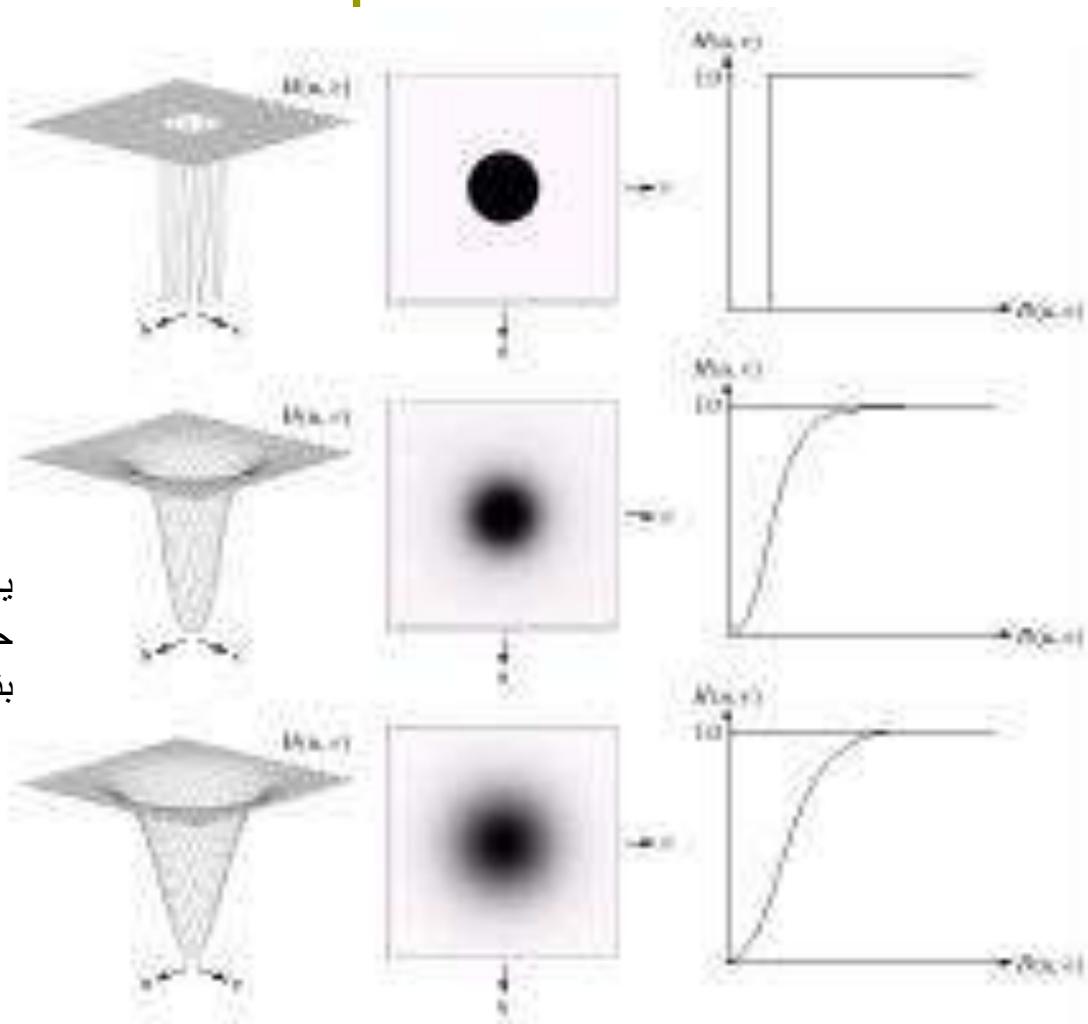
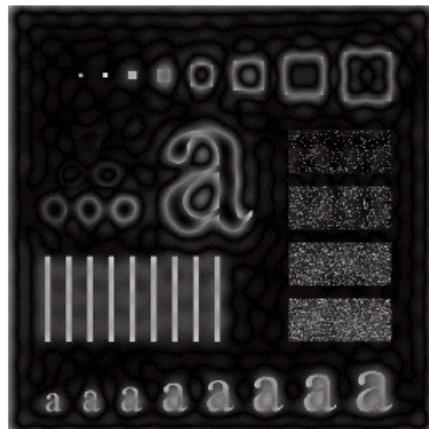
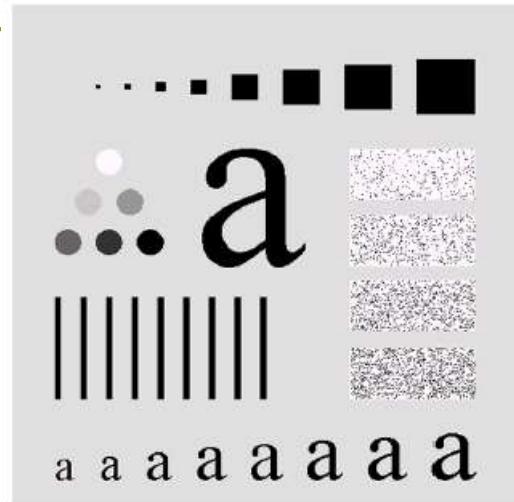
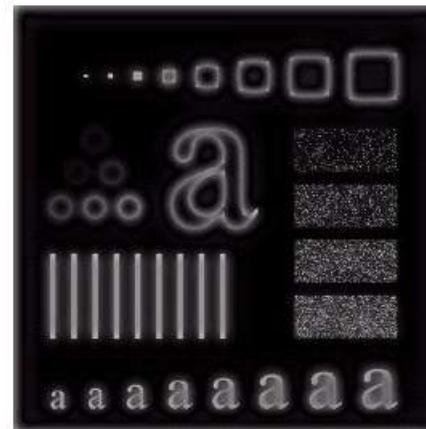


FIGURE 4.27 Top row: 3D surface plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

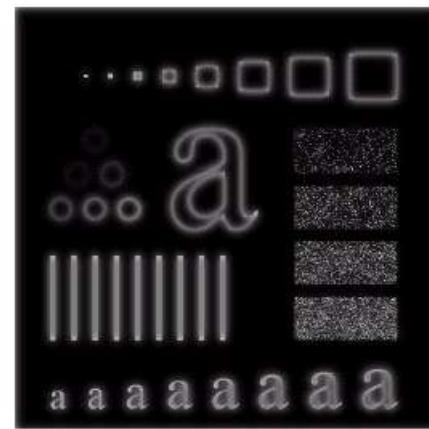
Highpass Filter Comparison



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

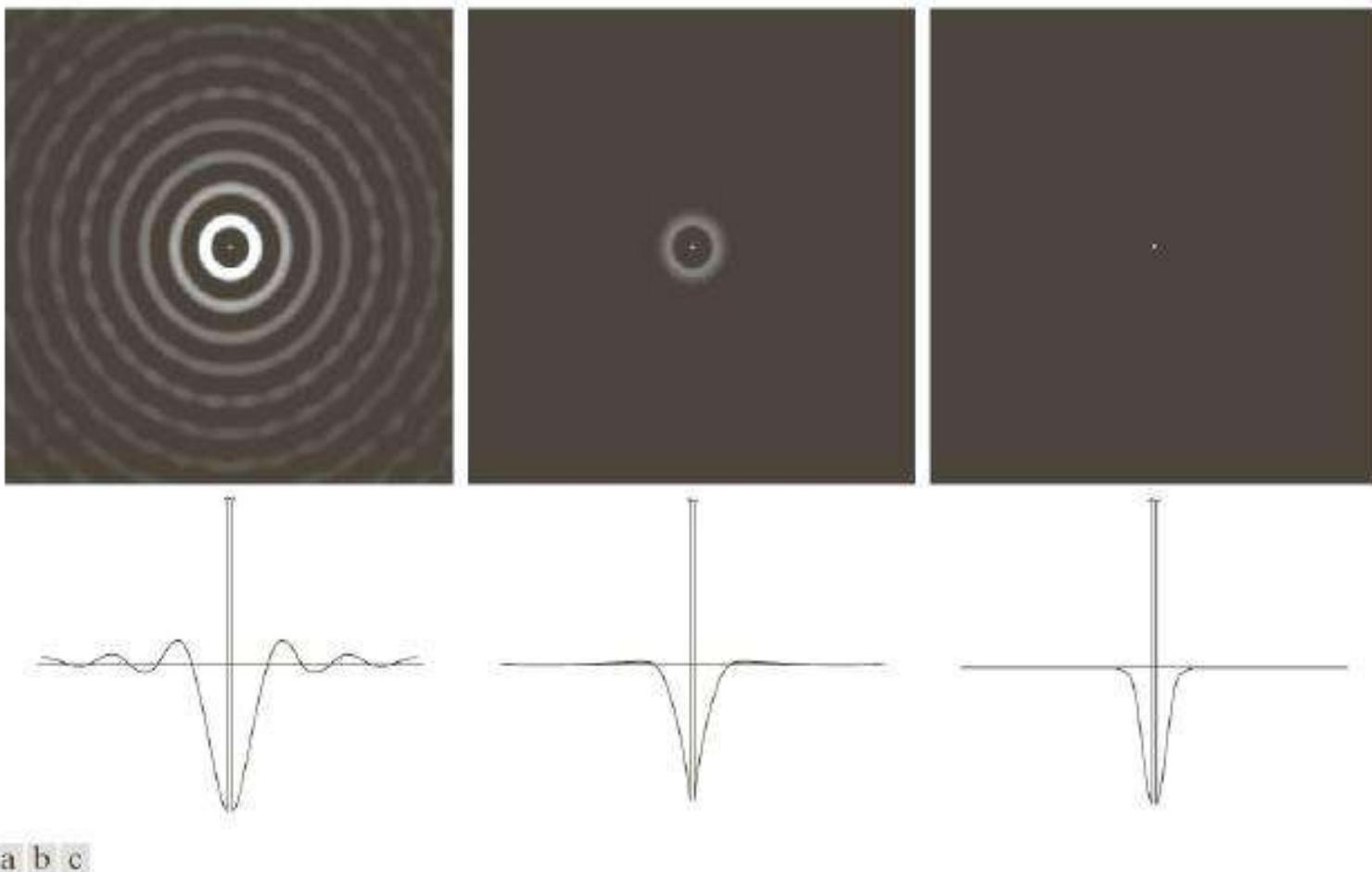


Results of Butterworth
high pass filtering of
order 2 with $D_o = 15$



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

The 3 HPF SD representation



a b c

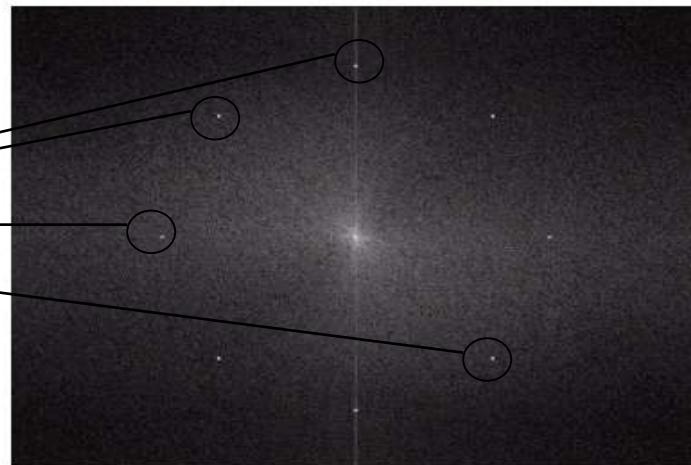
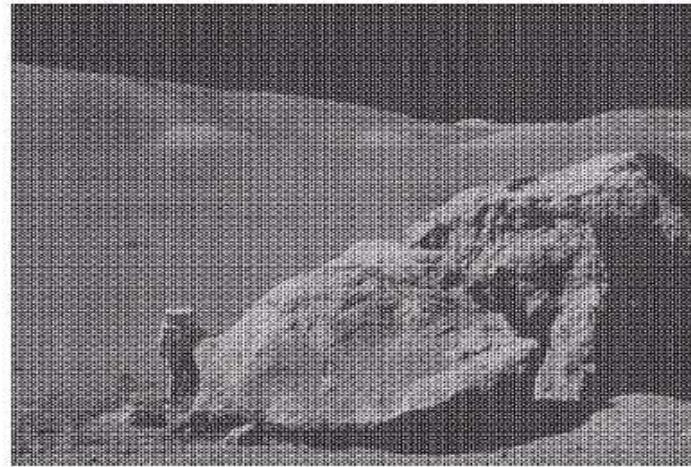
FIGURE 4.53 Spatial representation of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding intensity profiles through their centers.

Periodic Noise

a
b

FIGURE 5.5

(a) Image corrupted by sinusoidal noise.
(b) Spectrum (each pair of conjugate impulses corresponds to one sine wave).
(Original image courtesy of NASA.)



Periodic noise can be reduced in via frequency domain

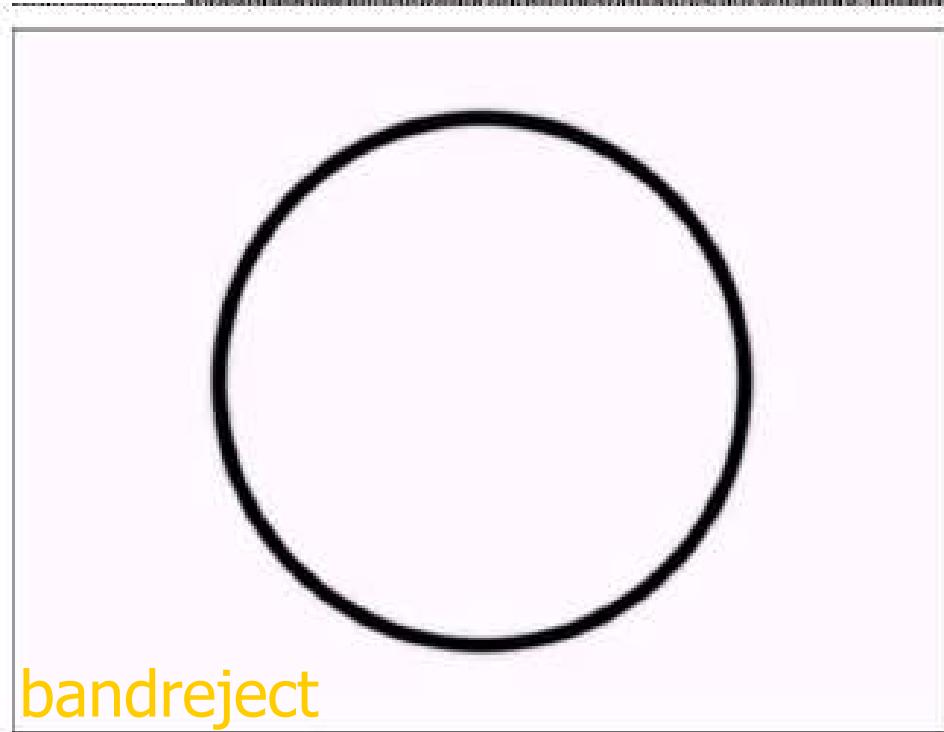
Are generated due to electrical or electromechanical interference during image acquisition



noisy



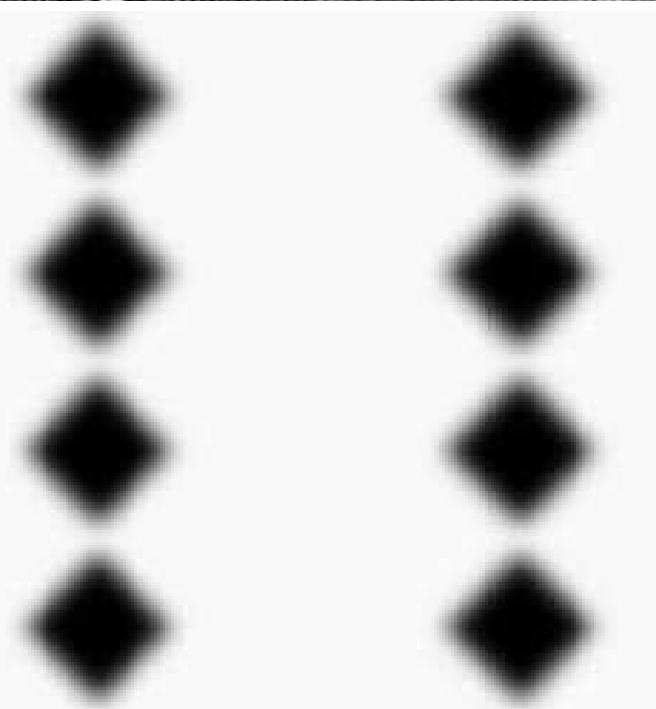
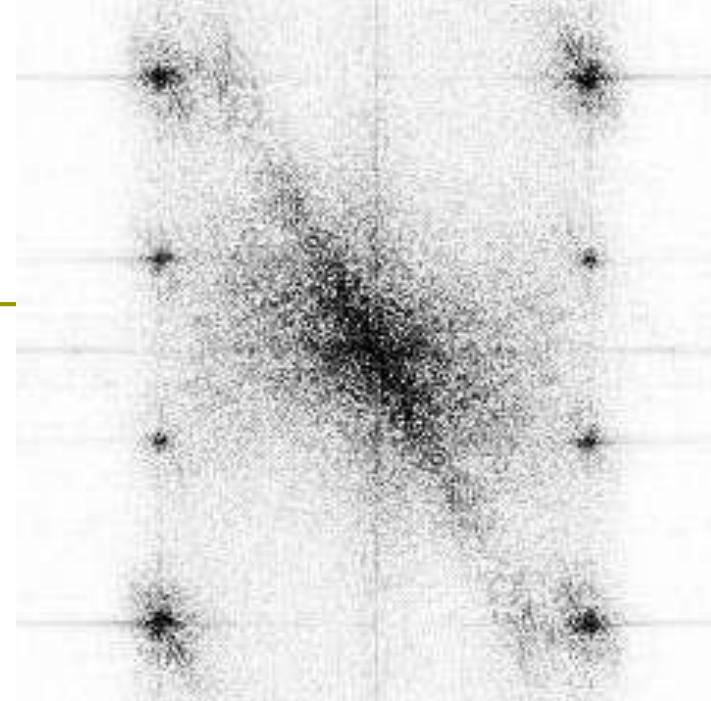
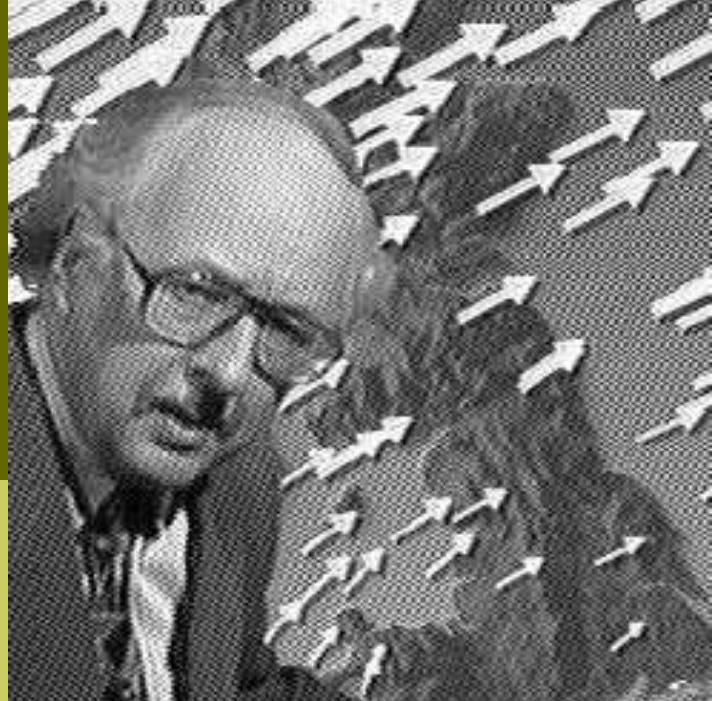
spectrum

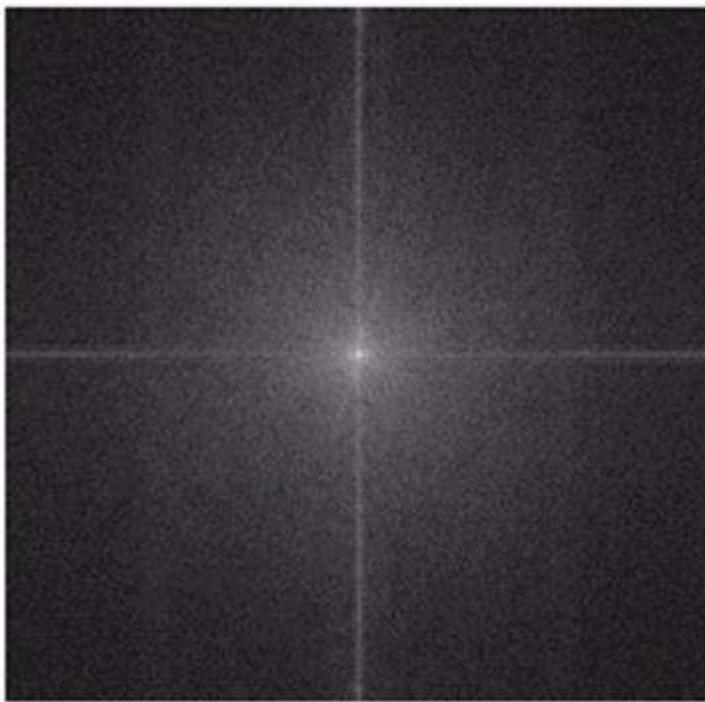


bandreject



filtered

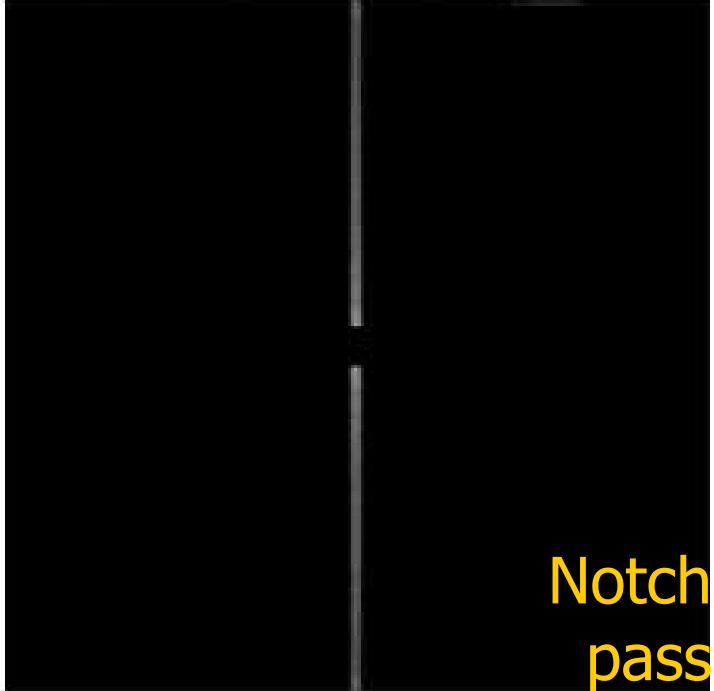




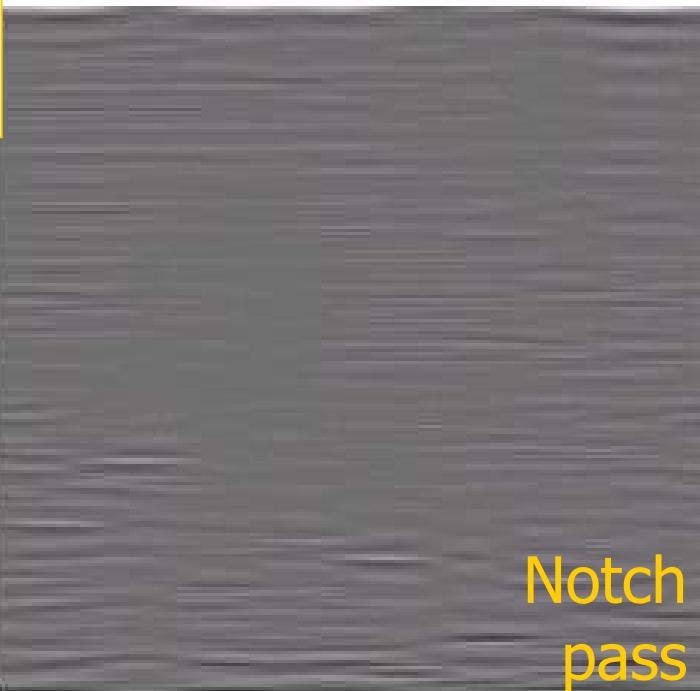
Horizontal
Scan lines



DFT



Notch
pass



Notch
pass



Notch
reject

Image Processing



Ch6: Color image Processing

Color Image Processing

- ❑ Color in image Processing is motivated by two factors:
 1. Color is a powerful descriptor that often simplifies object identifications and extraction from the scene.
 2. Human can discern thousands of color shades and intensities, compared to about only two dozen shades of gray.

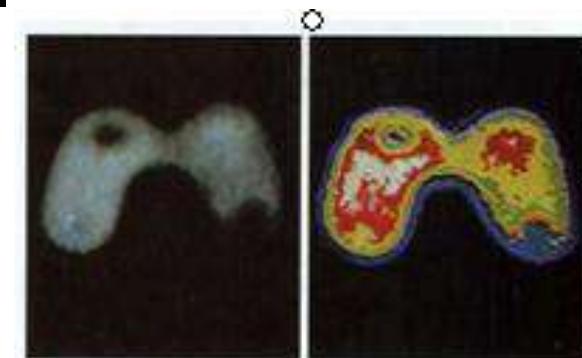
Image color processing areas

Areas of color image processing:

1) ***Full-color processing:*** images are acquired with a full-color Sensor , such as TV camera or scanner.



2) ***Pseudo-color processing:*** a color is assigned to a particular monochrome intensity or range of intensities.



Pseudo-color processing:

Pseudo-color processing: assignment of colors to gray values (in monochrome/grayscale images) based on a specified criterion.

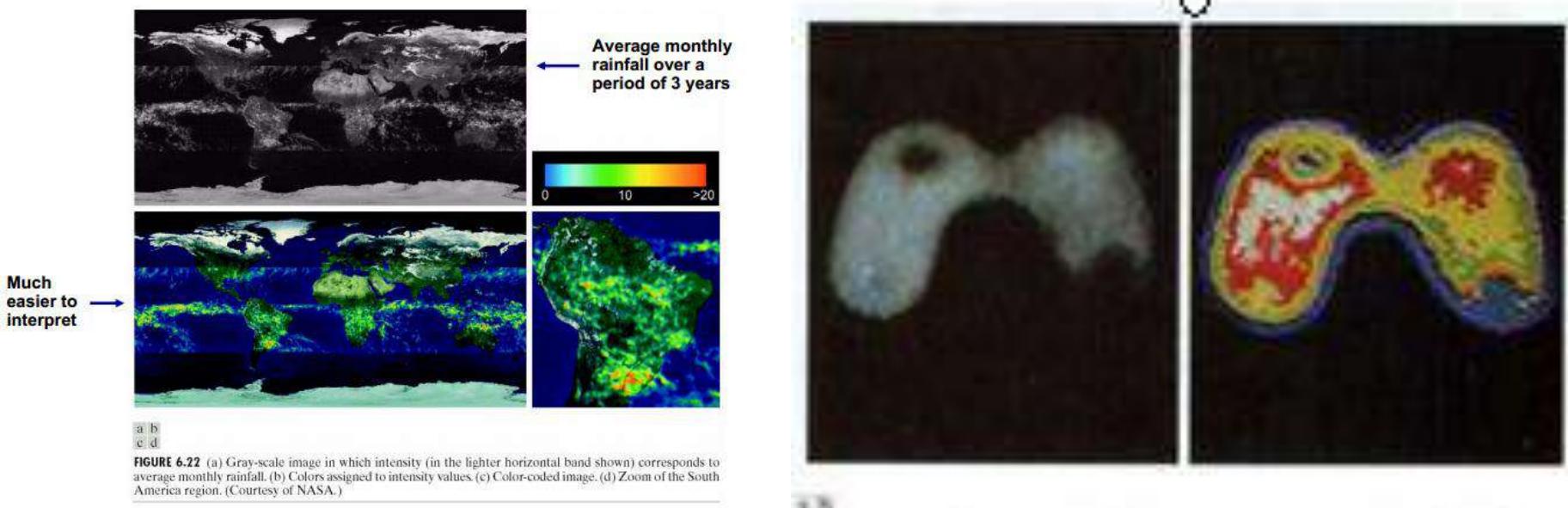


Image color processing areas

- In the past, most digital image color processing was done at the pseudo-color level.

- Today, full-color image processing techniques are used in a broad range of applications. Like publishing, internet and visualizing.

Color spectrum

Isaac Newton discovered that: When passing a beam of sunlight through a prism, the light is decomposed into a spectrum of colors : violet, blue, green, yellow, orange, red

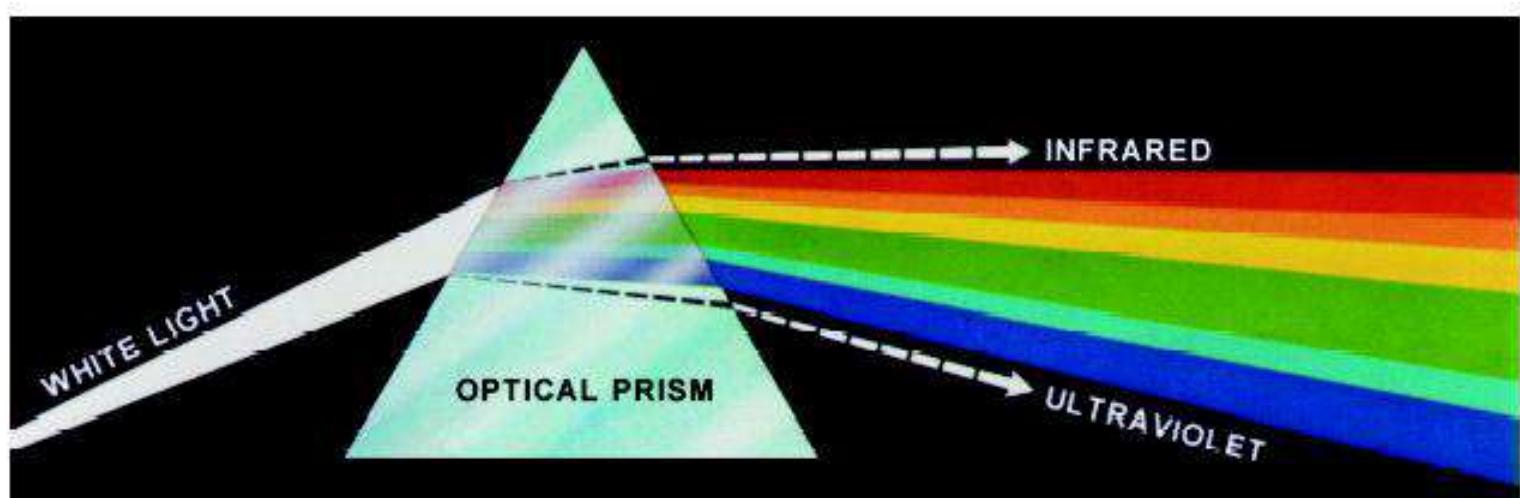


FIGURE 6.1 Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lamp Business Division.)

Electromagnetic spectrum

- Visible light spans the electromagnetic spectrum from approximately 400 to 700 nm.

Ultraviolet – visible light – infrared

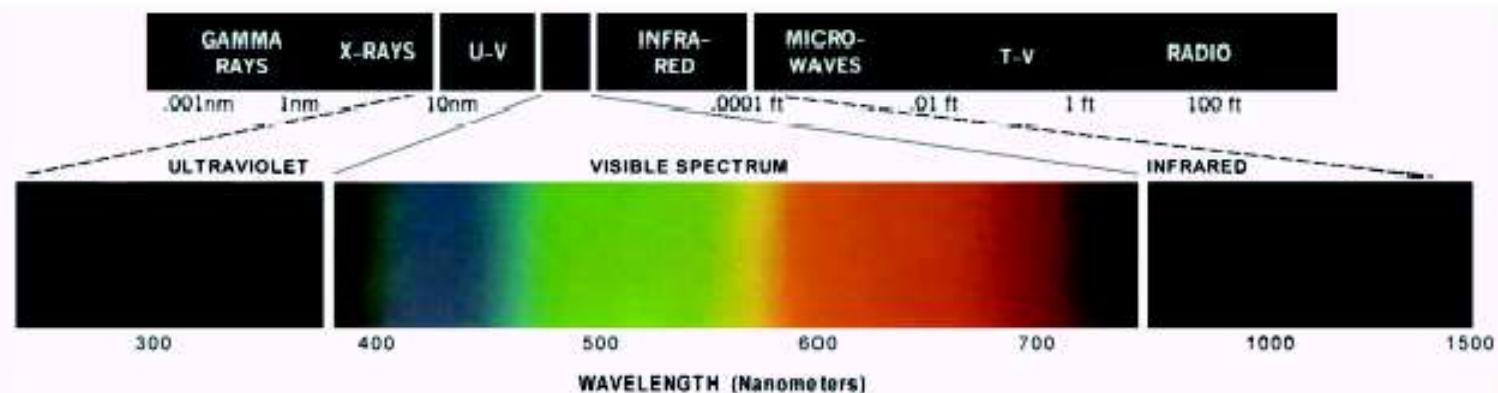
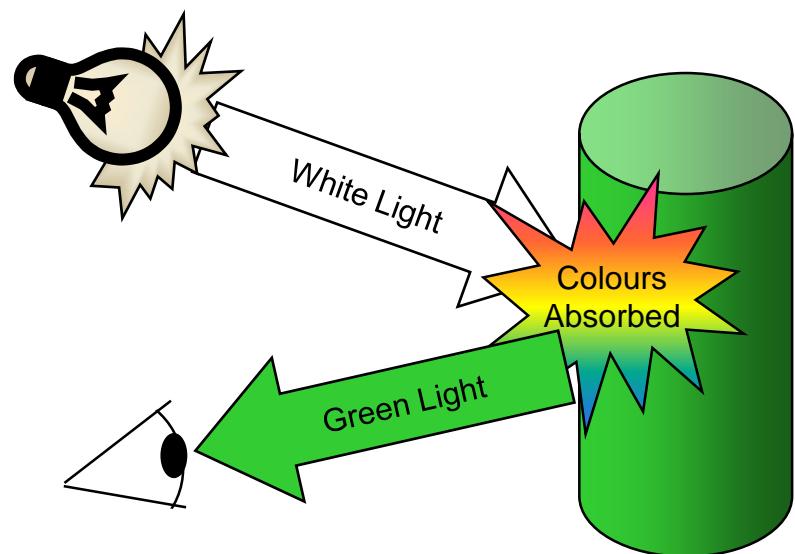


FIGURE 6.2 Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lamp Business Division.)

Color spectrum

The colors that human perceive in an object are determined by the nature of the light reflected from that object.

Example: Green objects reflect light with wavelengths in 500-570 nm while absorbing most of the energy at other wavelengths.



Color Spectrum

- Characterization of light is central to the science of color.
 - **Achromatic light:** its only attribute is its intensity (gray level refers to a scalar measure of intensity that ranges from black to white).
 - **Chromatic light:** spans the EM spectrum from 400nm to 700nm.

Note: Color may be characterized by:

brightness and **Chromaticity**

Chromatic light quantities:

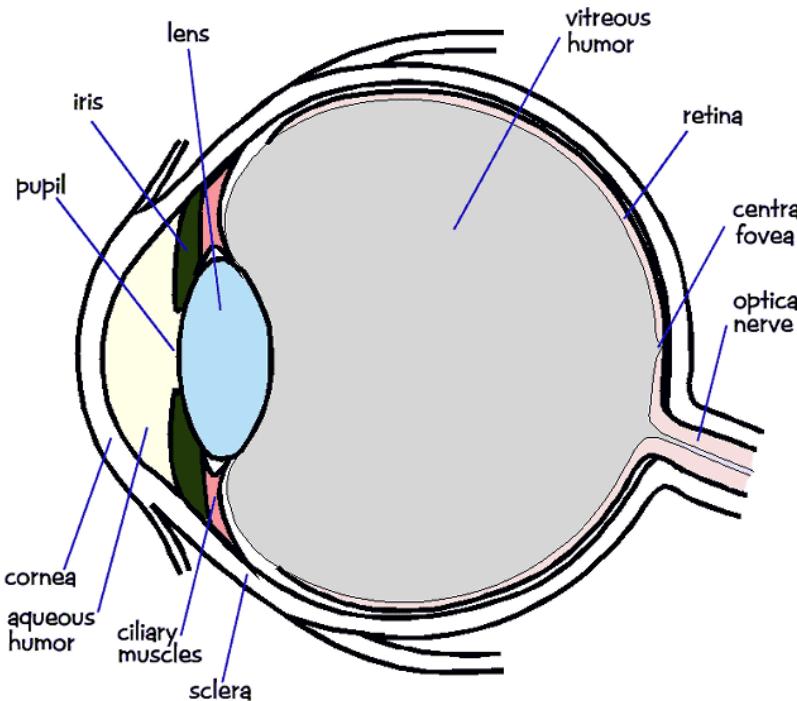
3 basic quantities to describe the quality of a chromatic light source:

- 1) **Radiance (W)** كمية الطاقة المنبعثة من الجسم (الاشعاع) total amount of energy that flows from the light source.
- 2) **Luminance (Im):** كمية الطاقة التي يحصل عليها الناظر the amount of energy an observer perceives from a light source.
- 3) **Brightness:** الاضاءة a subjective descriptor that is impossible measure in practice. It embodies the achromatic notation of intensity (color sensation). It is very hard to measure.

Example: light produced from a source operating in the far infrared region.

- ❑ Radiance: significant!
- ❑ Luminance: hardly perceived!

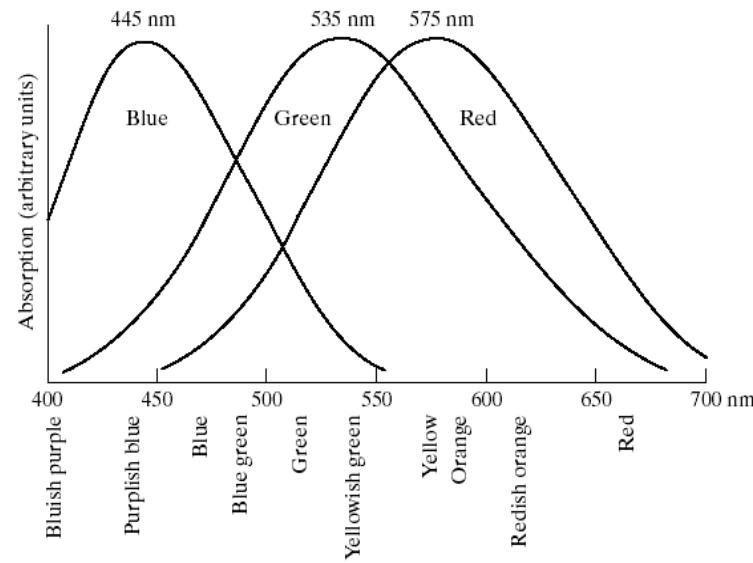
Human visual system



- Color perception
 - Light hits the retina, which contains photosensitive cells.
 - These cells convert the spectrum into a few discrete values.

Human visual system

- There are two types of photosensitive cells:
 - Cones
 - Sensitive to colored light, but not very sensitive to dim light.
 - Rods
 - Sensitive to achromatic light.
- We perceive color using three different types of cones.
 - Each one is sensitive in a different region of the spectrum.
 - 440 nm (BLUE)
 - 545 nm (GREEN)
 - 580 nm (RED)
 - They have different sensitivities



Additive and subtractive colors

Due to the absorption characteristics of the human eye, color are seen as variable combinations of the primary colors

Additive primary colors: **RGB**

use in the case of light sources such as color monitors.

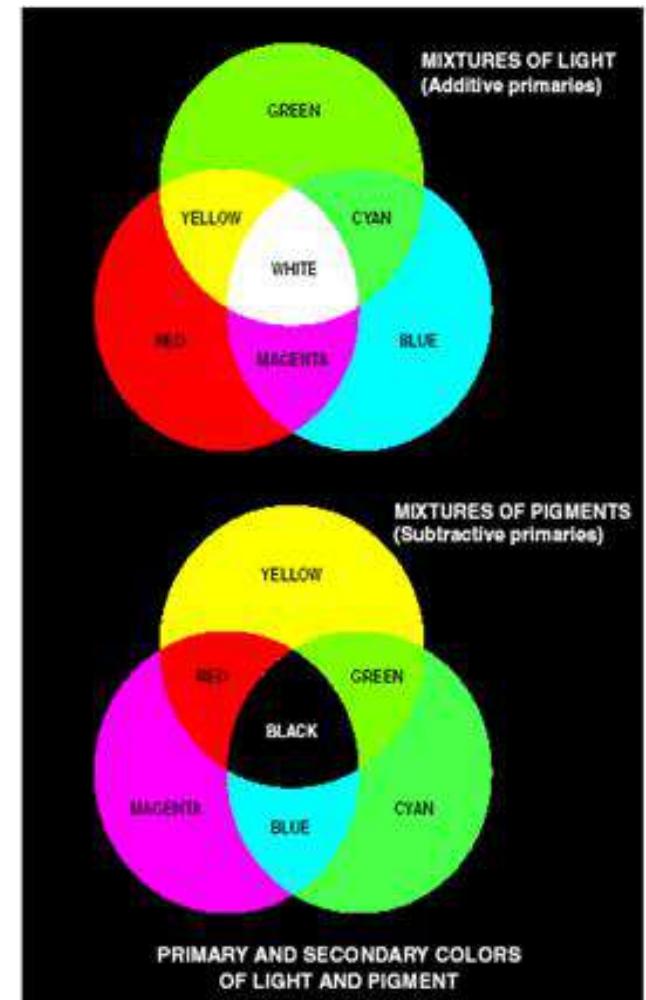
The primary colors can be added to produce the secondary colors of light.

RGB add together to get white.

Subtractive primary colors: **CMY**

use in the case of pigments in printing devices.

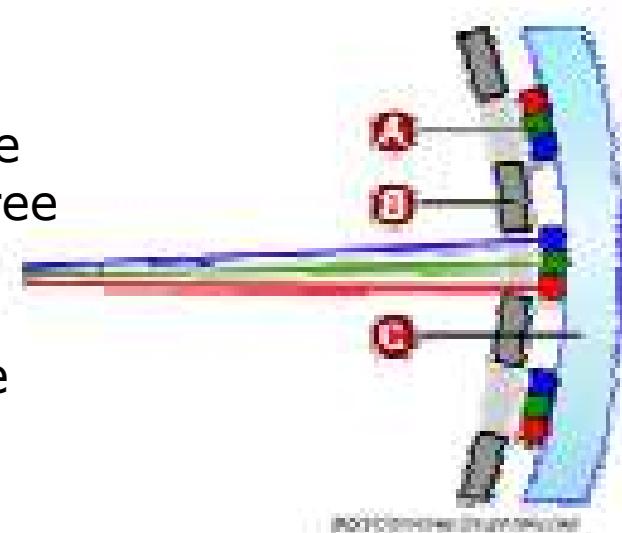
CMY add together to get Black



Color TV

A color TV is an example of the additive nature of light colors:

1. When a color TV needs to create a red dot, it fires the red beam at the red phosphor. Similarly for green and blue dots.
2. To create a white dot, red, green and blue beams are fired simultaneously -- the three colors mix together to create white.
3. To create a black dot, all three beams are turned off as they scan past the dot. All other colors on a TV screen are combinations of red, green and blue.



Color models

Color model (Or color space or system) is: A color model is a specification of colors in a standard way.

Color model: *A specification of coordinate system and a subspace where each color is represented by a single point.*

Color models to study:

Hardware-oriented models:

- RGB (red, green, blue) model for color monitors and video cameras
- CMY (cyan, magenta, yellow) model for color printing

Image processing oriented (developing algorithms based on color description that are natural intuitive to humans):

- HSI (hue, saturation, intensity) decouples the color and gray-scale information in an image.

RGB model

Images represented with the RGB model have 3 (components) matrixes: Red matrix, Green matrix, AND Blue matrix

If 8 bits are used for each pixel, we have a 24-bit RGB image.

(this means **PIXEL DEPTH** is: 24 bits per pixel)

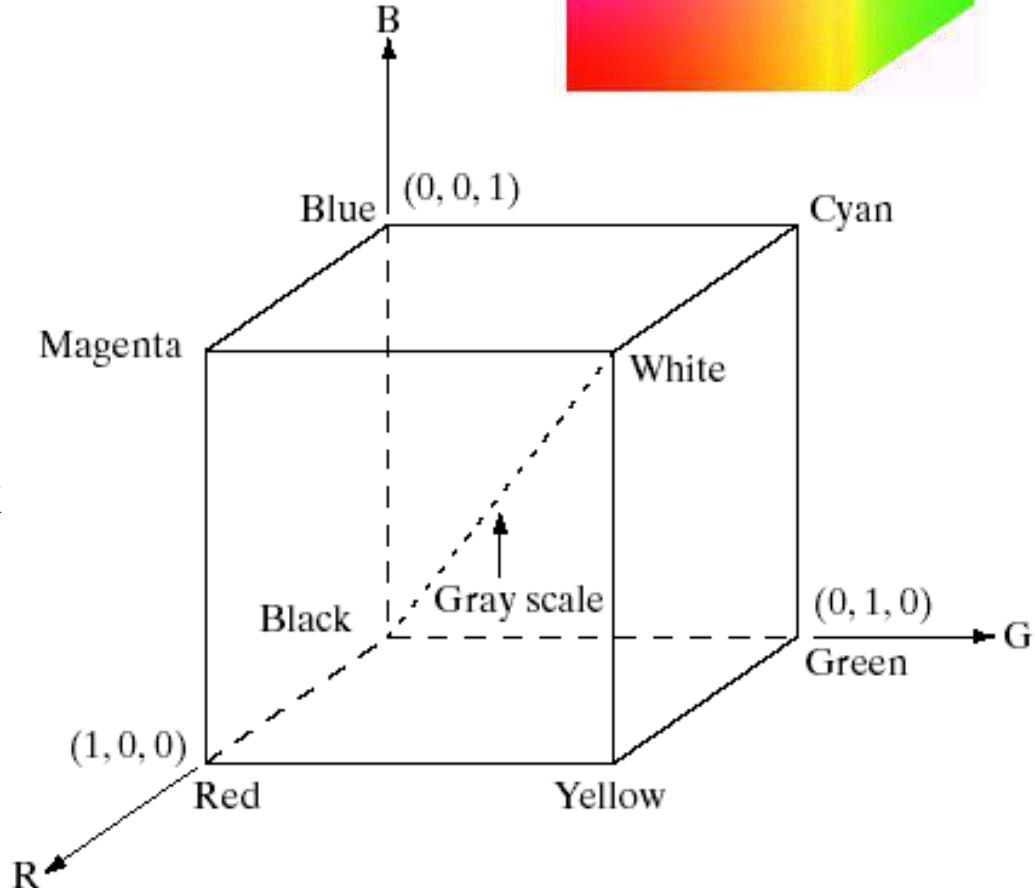
Pixel depth is #of bits used to represent each pixel in the color image

*i.e: each color appears in its primary spectral components :R,G,B ,
This model is based on Cartesian coordinate system.*

24-bit color cube:

Note that :

- Red, green , blue colors are at three corners
- Cyan, yellow , magenta are in the other corners
- Black is in the center.
- Grayscale extends from black to white along a line.
- Each color is represented by a point in or on the unit cube.
- pixel depth here is 24 bits.



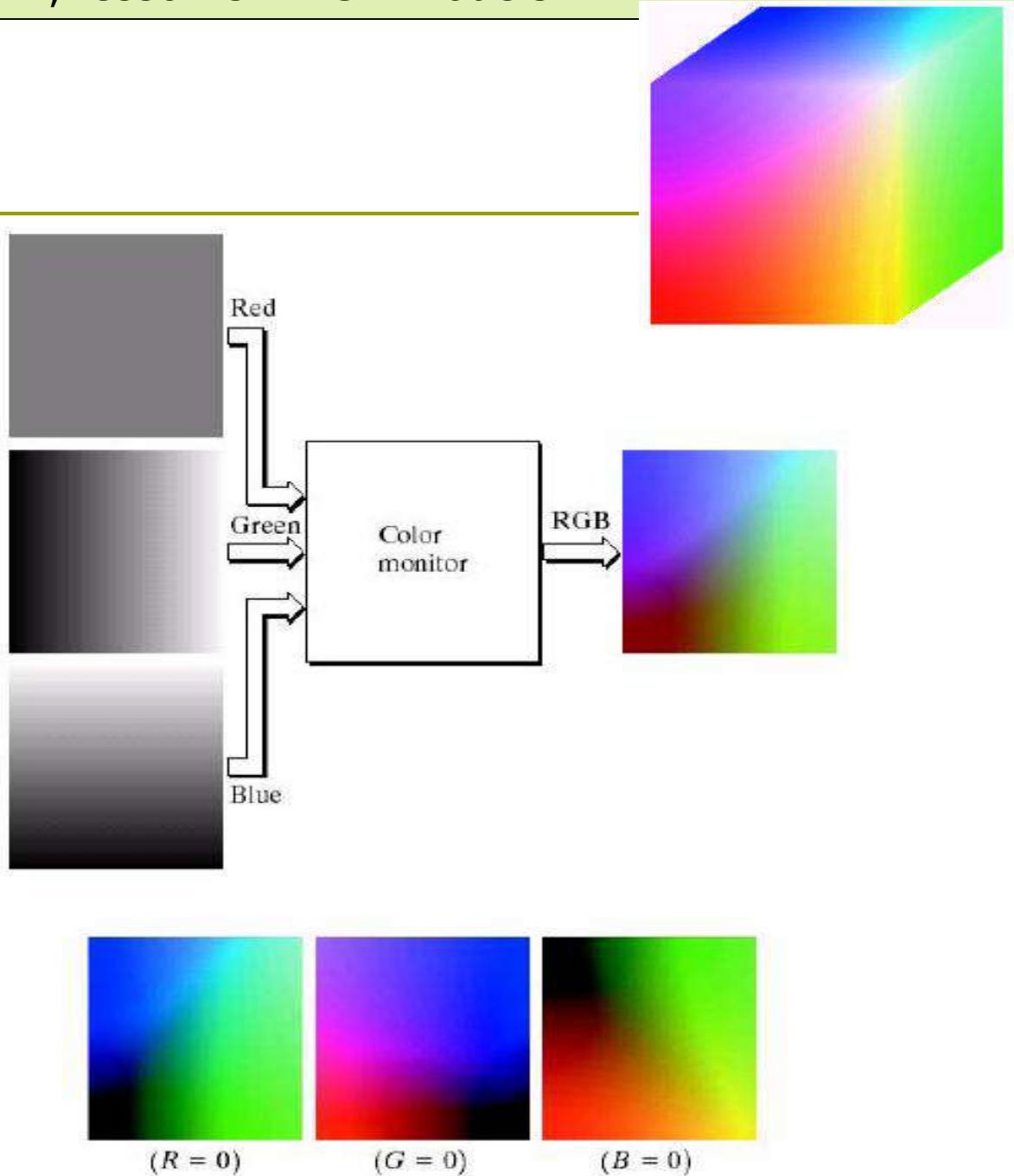
The total # of colors in 24-bit color image is: $(2^8)^3 = 16,777,216$ colors

RGB model

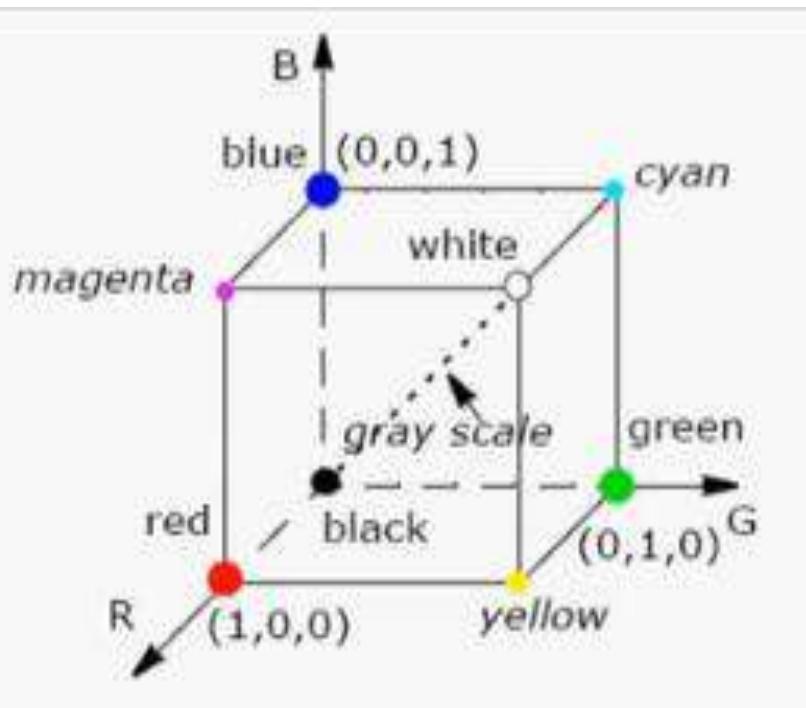
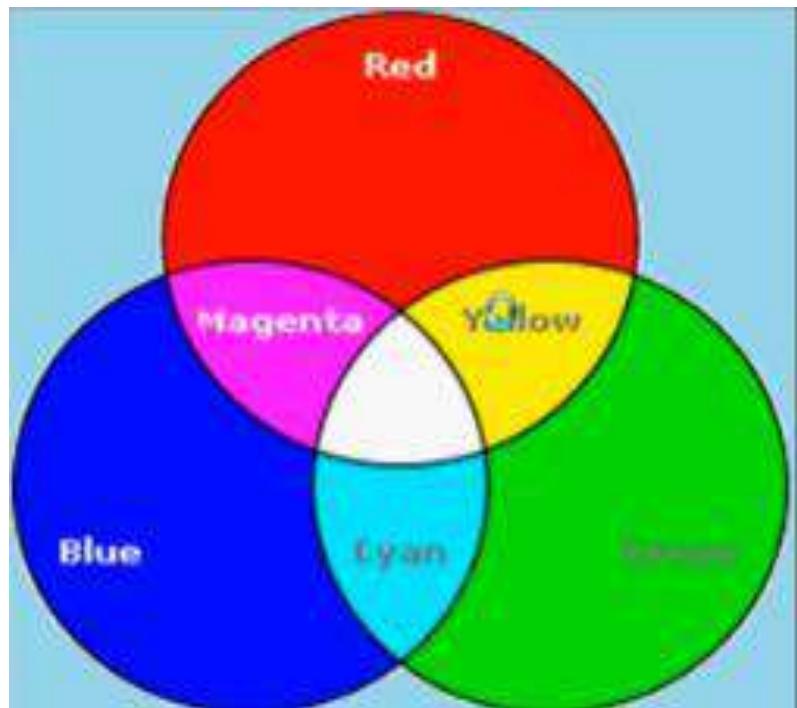
a
b

FIGURE 6.9

- (a) Generating the RGB image of the cross-sectional color plane ($127, G, B$).
(b) The three hidden surface planes in the color cube of Fig. 6.8.



RGB Color Model

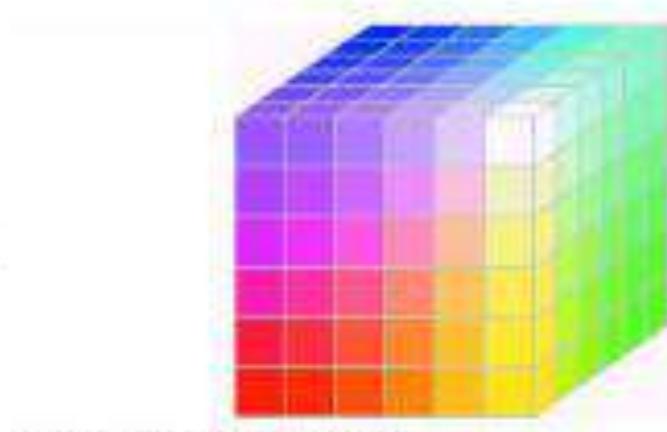
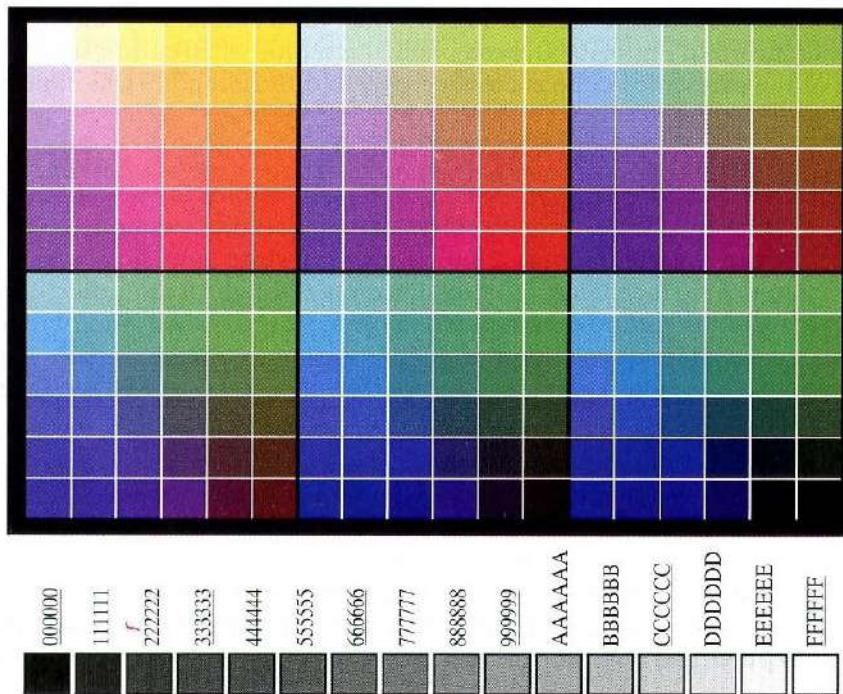


The Safe Colours:

A subset of colours

Most systems use 256 colours

216 colours are common to most OS.



RGB surface color cube

Each **surface** has 36 colors: $6 \times 36 = 216$

RGB Hex Triplet Color Chart

E-mail-ware... What a concept!

If you find this chart helpful, send
mail to Doug and say "Thanks!"
jacobson@pop.c-com.net

NON-DITHERING COLORS

FFFFFF	FFCCFF	FF99FF	FF66FF	FF33FF	FF00FF
FFFFCC	FFCCCC	FF99CC	FF66CC	FF33CC	FF00CC
FFFF99	FFCC99	FF9999	FF6699	FF3399	FF0099
FFFF66	FFCC66	FF9966	FF6666	FF3366	FF0066
FFFF33	FFCC33	FF9933	FF6633	FF3333	FF0033
FFFF00	FFCC00	FF9900	FF6600	FF3300	FF0000
EEEEE0	CCCCF0	CC99FF	CC66FF	CC33FF	CC00FF
000000	CCCCF0	CC99FF	CC66FF	CC33FF	CC00FF
CCCCCC	CCCCF0	CC99FF	CC66FF	CC33FF	CC00FF
BBBBBB	CCCCF0	CC99FF	CC66FF	CC33FF	CC00FF
AAAAAA	CCCCF0	CC99FF	CC66FF	CC33FF	CC00FF
999999	CCCC99	CC9999	CC6699	CC3399	CC0099
888888	CCCC66	CC9966	CC6666	CC3366	CC0066
777777	CCCC33	CC9933	CC6633	CC3333	CC0033
666666	CCCC00	CC9900	CC6600	CC3300	CC0000
555555	9999FF	9999FF	9966FF	9933FF	9900FF
444444	99FFCC	9999CC	9966CC	9933CC	9900CC
333333	99FF99	99CC99	999999	996699	993399
222222	99FF66	99CC66	999966	996666	993366
111111	99FF33	99CC33	999933	996633	993333
000000	99FF00	99CC00	999900	996600	993300
F00000	66FFFF	66CCFF	6699FF	6633FF	6600FF
EE0000	66FFCC	66CCCC	6699CC	6633CC	6600CC
DD0000	66FF99	66CC99	669999	663399	660099
CC0000	66FF66	66CC66	669966	663366	660066
BB0000	66FF33	66CC33	669933	663333	660033
AA0000	66FF00	66CC00	669900	663300	660000
990000	33FFFF	33CCFF	3399FF	3366FF	3333FF
880000	33FFCC	33CCCC	3399CC	3366CC	3333CC
770000	33FF99	33CC99	339999	336699	333399
660000	33FF66	33CC66	339966	336666	333366
550000	33FF33	33CC33	339933	336633	333333
440000	33FF00	33CC00	339900	336600	333300
330000	00FFFF	00CCFF	0099FF	0066FF	0033FF
220000	00FFCC	00CCCC	0099CC	0066CC	0033CC
110000	00FF99	00CC99	009999	006699	003399
000000	00FF66	00CC66	009966	006666	003366
	00FF33	00CC33	009933	006633	003333
	00FF00	00CC00	009900	006600	003300

Hex	00	33	66	99	CC	FF
Decimal	0	51	102	153	204	255

$$(6)^3 = 216$$



Trichromatic Coefficients:

The amounts of red, green blue needed to form any particular color are called **tristimulus values** and are denoted X,Y, and Z respectively.

Now Suppose: X: red , Y: green, Z: blue

Then, a color is specified by its trichromatic coefficients as follows:

$$x = X / (X+Y+Z)$$

$$y = Y / (X+Y+Z)$$

$$z = Z / (X+Y+Z)$$

Where (**x + y + z = 1**)

Trichromatic Coefficients- cont.:

Example: suppose we have the following color with R,G,B as defined, then calculate trichromatic coefficients (x,y,z) as follows:

$$x = R / (R+G+B) \rightarrow 255 / (255+194+14) = 0.5504$$

$$y = G / (R+G+B) \rightarrow 194 / (255+194+14) = 0.419$$

$$z = B / (R+G+B) \rightarrow 14 / (255+194+14) = 0.030$$



Chromaticity diagram

- Specifying colors systematically can be achieved using the CIE chromacity diagram, which shows color composition as a function of x (red) and y (green).
- On this diagram, the x-axis represents the proportion of red and the y-axis represents the proportion of green used .

$$x + y + z = 1$$

- The proportion of blue used in a color is calculated as:

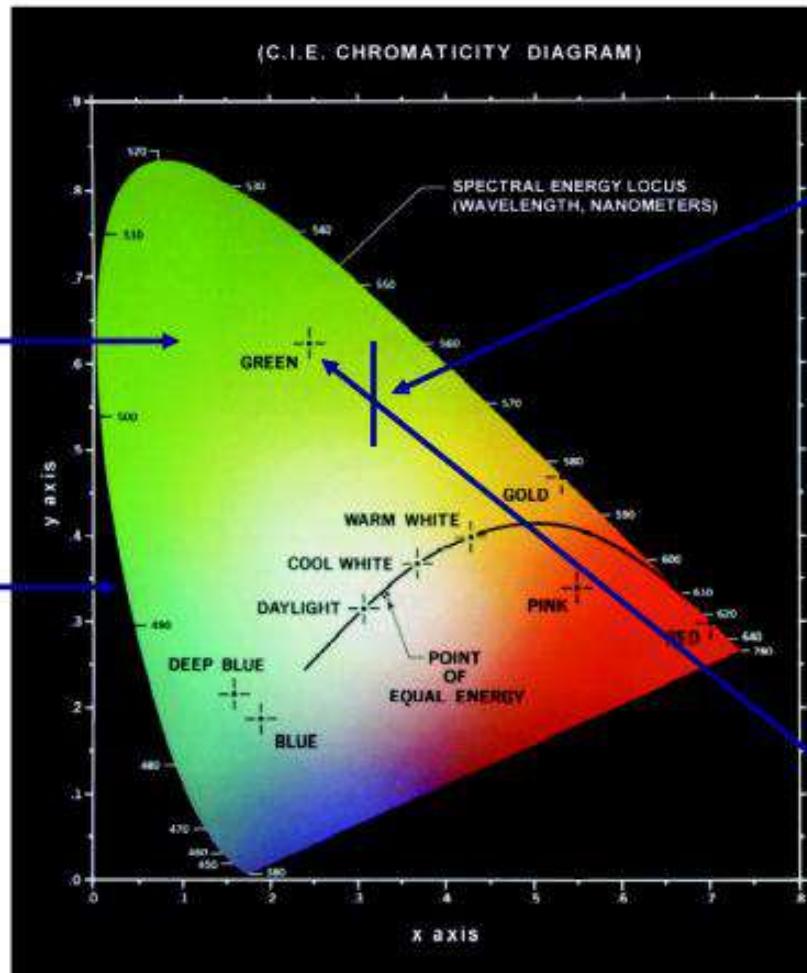
$$z = 1 - (x + y)$$

Chromaticity diagram

FIGURE 6.5
Chromaticity
diagram.
(Courtesy of the
General Electric
Co., Lamp
Business
Division.)

Any point within
the diagram
represents some
mixture of
spectrum colors.

Points on the
boundary are
pure colors in
the visible
spectrum.



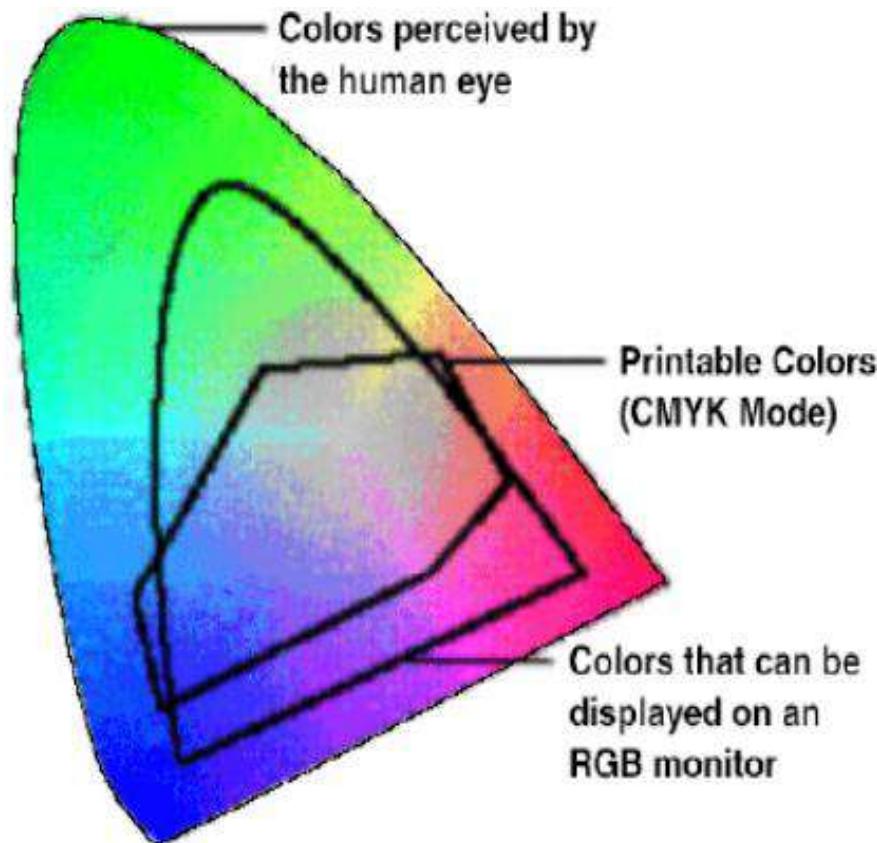
A straight line segment joining any 2 points defines all the different color variations that can be obtained by mixing these 2 colors additively.

x axis: red
y axis: green

Colors on the
boundary:
have highest saturation

x: 25%, y: 62%
so: z: 13%.

Color gamut



A line segment indicates all colors that can be produced by mixing two colors corresponding to the end points of the line.

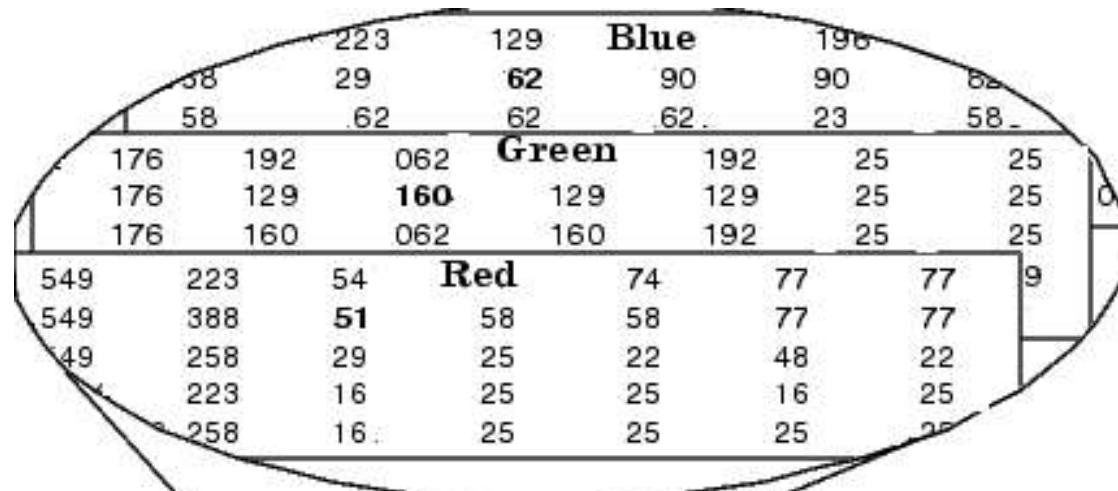
Each color model has different color range (or gamut). RGB model has a larger gamut than CMY. Therefore, some color that appears on a screen may not be printable and is replaced by the closest color in the CMY gamut.

RGB Color Values

Here is a table with common RGB color values:

R	G	B	Hex Value	Color
0	0	0	000000	Black
255	0	0	FF0000	Red
0	255	0	00FF00	Green
0	0	255	0000FF	Blue
255	255	0	FFFF00	Yellow
255	0	255	FF00FF	Magenta
0	255	255	00FFFF	Cyan
255	128	128	FF8080	Bright Red
128	255	128	80FF80	Bright Green
128	128	255	8080FF	Bright Blue
64	64	64	404040	Dark Grey
128	128	128	808080	Intermediate Grey
192	192	192	C0C0C0	Bright Grey
255	255	255	FFFFFF	White

RGB model

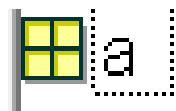


RGB model - Matlab-

Assume we want to read the following image and display it in matlab.

```
>>a=imread('aqaba.jpg');
```

```
>> imshow(a);
```

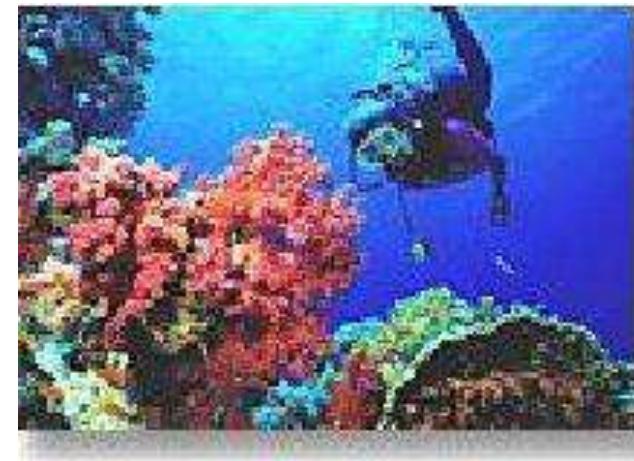


160 rows

<160x229x3 u... uint8

229 columns

3 matrixes (RGB)



RGB model - Matlab- continued.

In image (a),

Red matrix : $a(:,:,1)$;

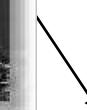
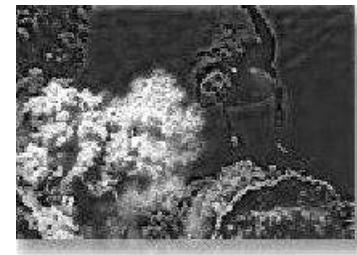
Green matrix: $a(:,:,2)$;

Blue matrix: $a(:,:,3)$;

RGB model - Matlab- continued.

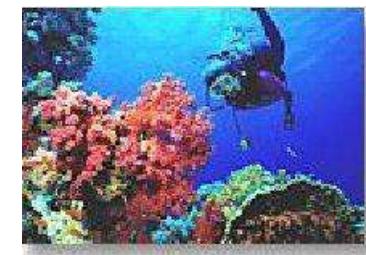
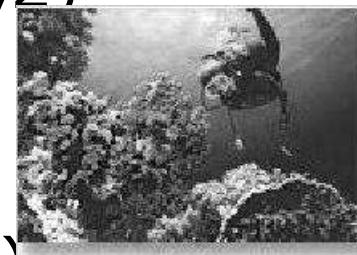
To show the **red** matrix $a(:,:,1)$

```
>>imshow(a(:,:,1));
```



To show the **green** matrix $a(:,:,2)$

```
>>imshow(a(:,:,2));
```



To show the **blue** matrix $a(:,:,3)$

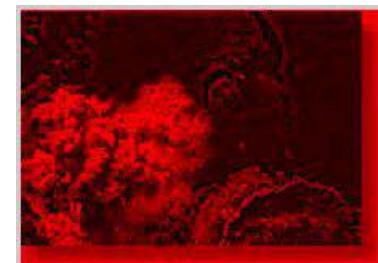
```
>>imshow(a(:,:,3));
```



RGB model - Matlab- continued.

To show the image , but with emphasizing the **red** matrix $a(:,:,1)$

```
>> y= a;      (save a copy of image in y)
>> y(:,:2)=0; (put zeros in the green matrix pixels)
>>y(:,:3)=0; (put zeros in the blue matrix pixels)
>>imshow(y);   (show the new image)
```



Exercise:

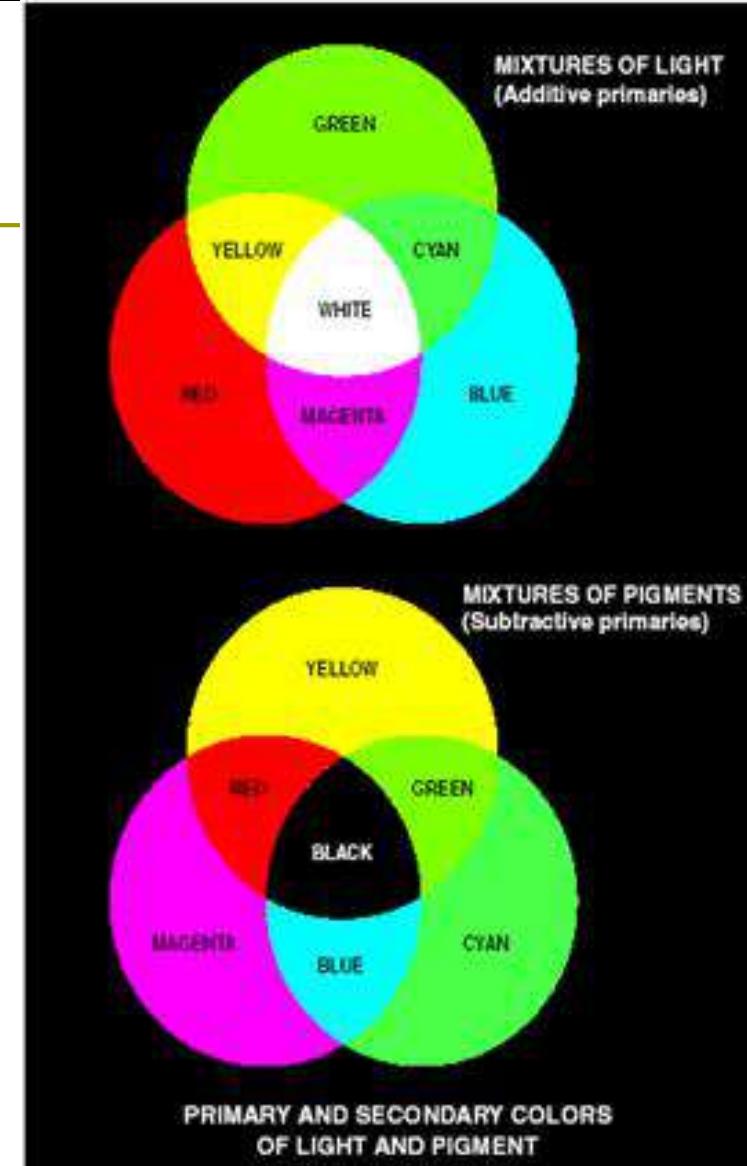
1. *do the same for green matrix*
2. *do the same for bleu matrix*

CMY model -

Secondary colors

- ❑ Magenta (R+B)
- ❑ Cyan (G+B)
- ❑ Yellow (R+G)

❑ They are considered as :
secondary colors of light
but: primary colors of pigment الصبغة

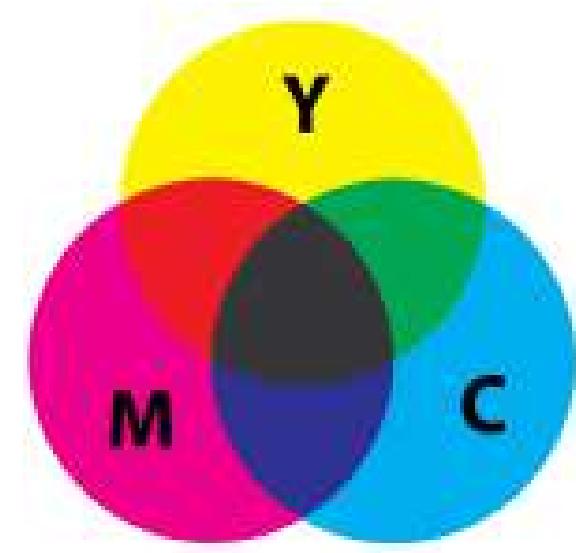


Used in:

- ❑ Color printer and copier
- ❑ Deposit colored pigment on paper
- ❑ Relationship with RGB model

Convert CMY to RGB model

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = 255 - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

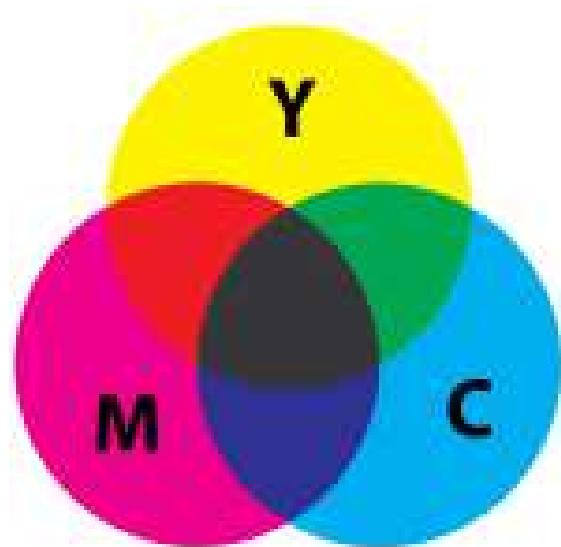


Example :surface coated with pure cyan does not contain red ($C = 1 - R$)

Combining cyan, magenta, and yellow should give black .
In practice, combining these color for printing produces a muddy-looking black. So, in order to produce true black, a forth color, black, is added, giving rise to the CMYK color model.

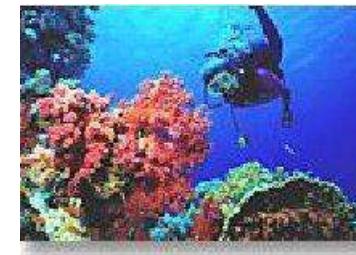
CMYK Color Model

- Combining cyan, magenta, and yellow should give black .
- In practice, combining these color for printing produces a muddy-looking black.
- So, in order to produce true black, a forth color, black, is added, giving rise to the **CMYK** color model.



Convert RGB to CMY model- Matlab

```
>>a=imread('aqaba.jpg');  
>> imshow(a); →  
→>b=255-a;  
>>Imshow(b); →
```



Or you may write:

```
→>b=a;  
>>b(:,:,1)=255-a(:,:,1);  
>>b(:,:,2)=255-a(:,:,2);  
>>b(:,:,3)=255-a(:,:,3);  
>>Imshow(b);
```

HSI model vs. RGB Model

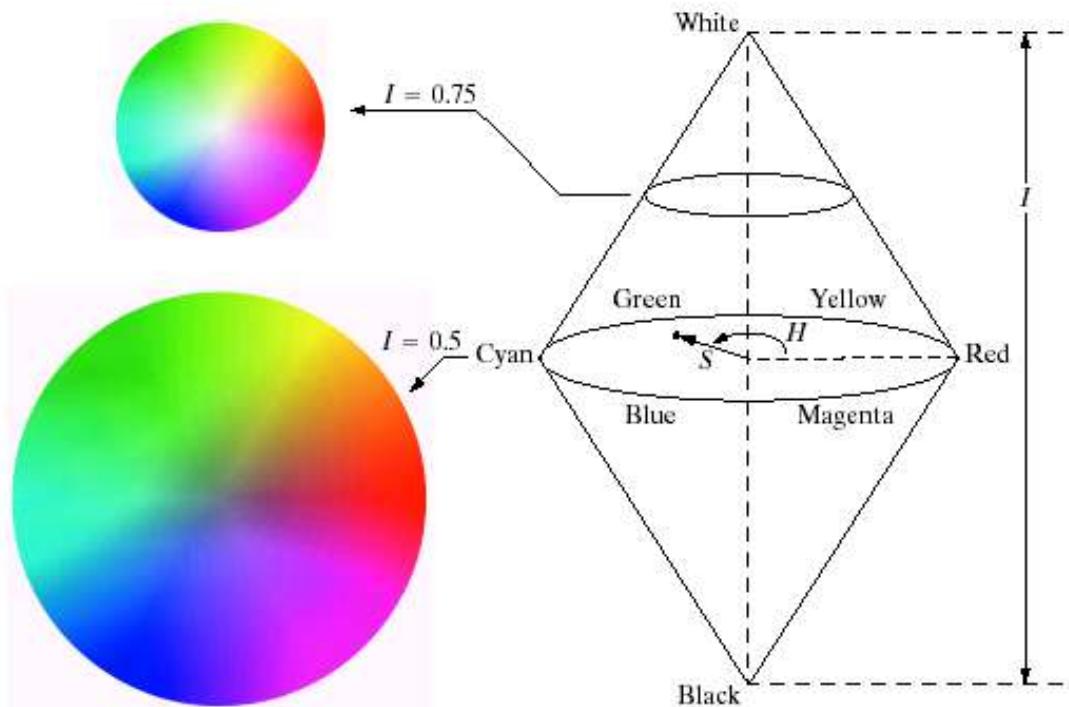
- RGB is useful for hardware implementations and it is not related to the way in which the human visual system works.
- However, RGB is not a particularly intuitive way in which to describe colors.
- Rather when people describe colors they tend to use hue, saturation, and brightness.
- RGB is great for color generation(as images captured from cameras, etc.), but HSI is great for color description.

HSI model

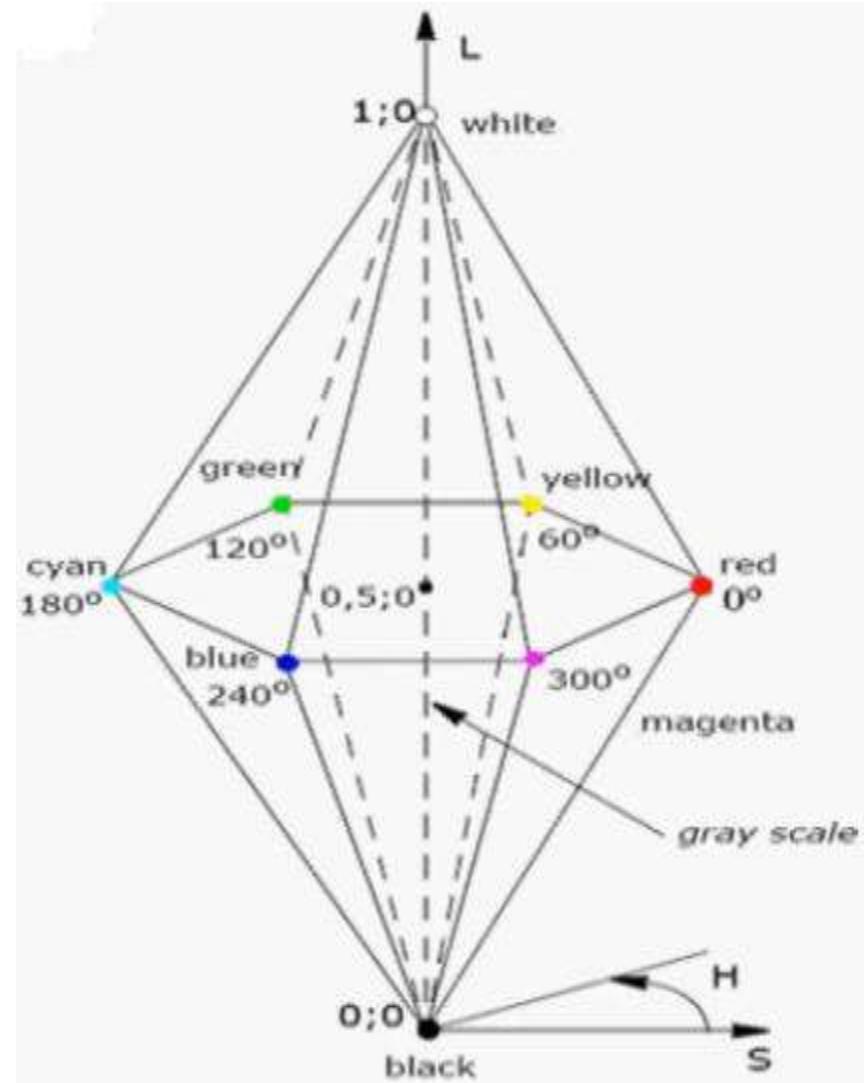
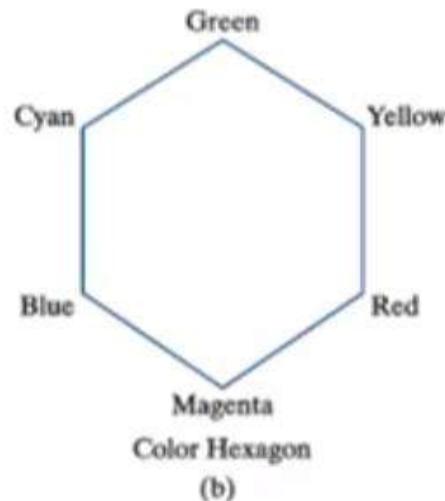
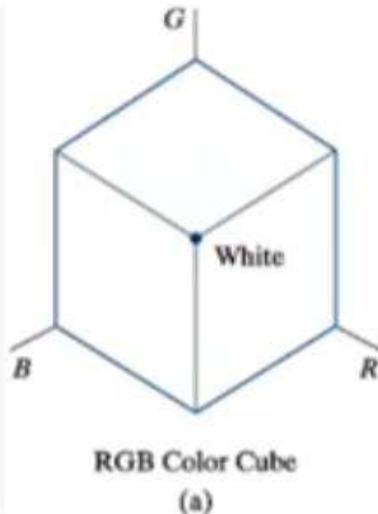


HSI Model

- ◻ Uniform: equal (small) steps give the same perceived color changes.
- ◻ Hue is encoded as an angle (0 to 2π).
- ◻ Saturation is the distance to the vertical axis (0 to 1).
- ◻ Intensity is the height along the vertical axis (0 to 1).



HSI model vs. RGB Model



HSI model

Humans describe a color object by its :
hue, saturation, and brightness.

Hue: a color attribute that describes a pure color. its dominant wavelength in a mixture of light waves.

Example: when we call an object :red, orange or green, we are referring to its Hue

Saturation: gives a measure of the degree to which pure color is diluted by white light. Or relative purity

Example: Pink is less saturated than red

Brightness: a subjective descriptor that is practically impossible to measure.. Its chromatic notation of intensity

$$\text{Chromaticity} = \text{Hue} + \text{Saturation}$$

→ The HSI model decouples the intensity component from the color-carrying information or chromacity (hue & saturation).

HSI Model



(Left) Image of food originating from a digital camera.
(Center) Saturation value of each pixel decreased 20%.
(Right) Saturation value of each pixel increased 40%.

Convert RGB to HSI

The RBG values have been normalized to the range [0,1].

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

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

$$I = \frac{1}{3}(R+G+B)$$

Convert HSI to RGB

The HSI values are given in the interval [0,1].

The applicable equations depend on the values of H .

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

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

GB sector ($120^\circ \leq H < 240^\circ$): first subtract 120° from it.

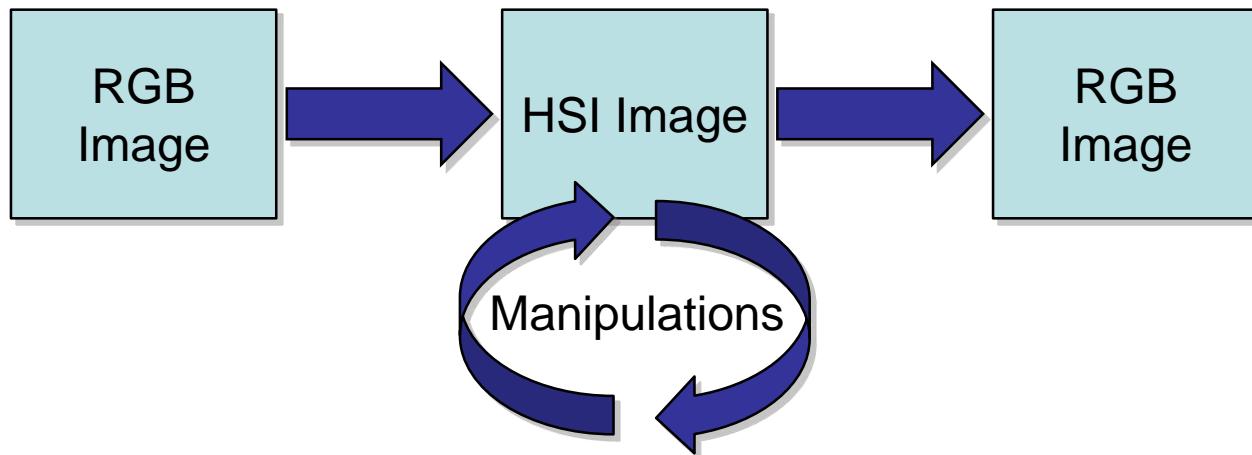
$$R = I(1 - S) \quad G = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right] \quad B = 3I - (R + G)$$

BR sector ($240^\circ \leq H \leq 360^\circ$): first subtract 240° from it.

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

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 manipulations under HSI.
 - Finally convert the image back from HSI to RGB.



Convert RGB to HSV in matlab

Convert RGB to HSV

```
>>a=imread('aqaba.jpg');  
>> imshow(a);  
>>b= rgb2HSV(a);
```

Convert RGB to HSV

```
>>c= HSV2RGB(b);
```

Color Image Processing



Pseudo-color image
processing

Image color processing areas

Areas of color image processing:

1) ***Full-color processing:*** images are acquired with a full-color Sensor , such as TV camera or scanner.



2) ***Pseudo-color processing:*** a color is assigned to a particular monochrome intensity or range of intensities.



Pseudo-color

Pseudo-color image processing: assignment of colors to gray values (in monochrome/grayscale images) based on a specified criterion.

يعني تحويل صورة grayscale الى صورة ملونة: مثال تلوين الافلام الابيض والاسود القديمة.

- The **principal use** الاستخدام of pseudo-color is for human visualization and interpretation of gray-scale events in images.
- A **principal motivation** المحفزات for using color is that humans can detect thousands of color shades and intensities!
- **Intensity slicing** and **color coding** is a simple example of pseudo-color image processing.

→ Pseudo-color(false color) اللون الخادع Is different than true color اللون الاصلي

Intensity slicing

Intensity slicing and color coding is the simplest example of pseudo-color image processing.

Image can be viewed as 3-d function (intensity VS spatial coordinates). This method creates planes as intersection in image intensity level, and assign a color to each level.

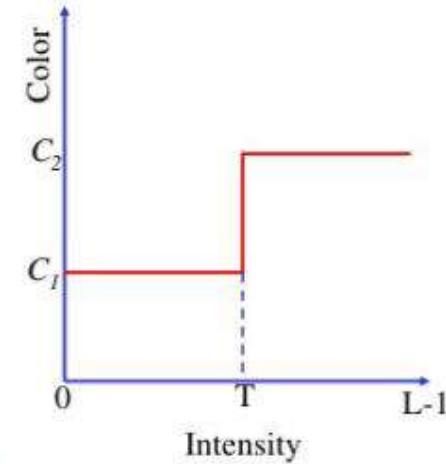
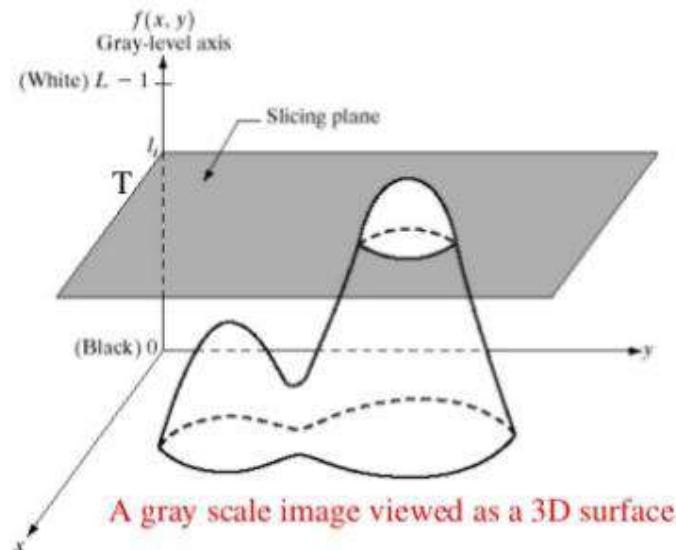
Example: if we have a gray level image of 8 bit per pixel, then we have 256 intensities

- If you want to create a color image with 5 color only, you can create 5 levels only, and assign a color for each:
- 0- 25: black, 26 – 90: red, 91 – 150: green, 155 – 200: yellow, 201: 255 :white.

Formula:

$$g(x, y) = \begin{cases} C_1 & \text{if } f(x, y) \leq T \\ C_2 & \text{if } f(x, y) > T \end{cases}$$

C_1 = Color No. 1
 C_2 = Color No. 2



Intensity slicing

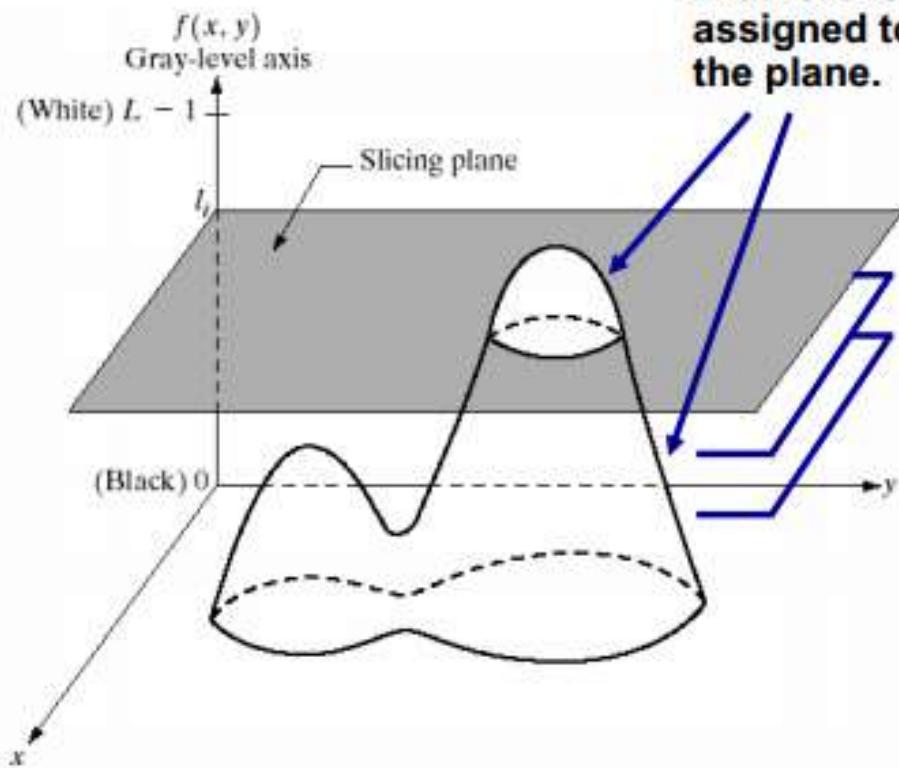


FIGURE 6.18 Geometric interpretation of the intensity-slicing technique.

Algorithm:

$[0, L-1]$: gray-scale

I_0 : black
 I_{L-1} : white

P planes: I_1, I_2, \dots, I_P

P planes partition the gray-scale into $P+1$ intervals:
 V_1, V_2, \dots, V_{P+1}

Color assignments:

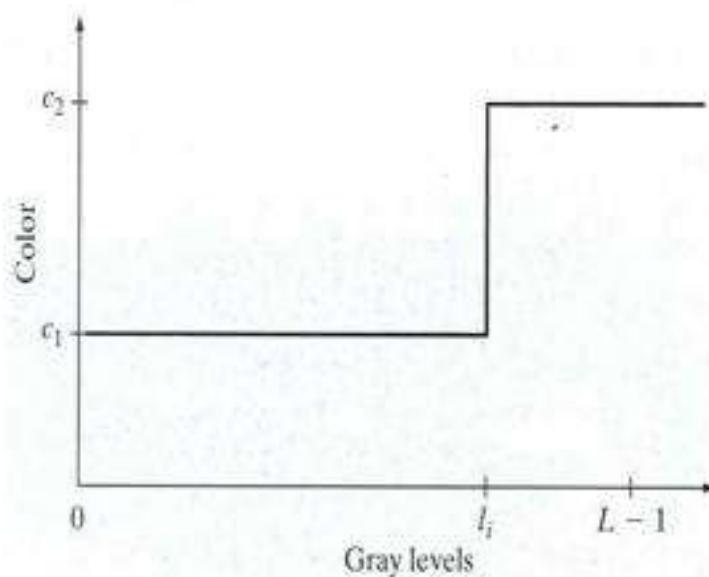
$$f(x,y) = c_k \text{ if } f(x,y) \in V_k$$

Intensity slicing

Suppose you want to have a color image with 2 color only , then you have one plane that divide the intensities into two intervals → two colors.

For example for intensity from $0 \rightarrow L_i$: Color c1.

From $L_i \rightarrow L - 1$: Color c2

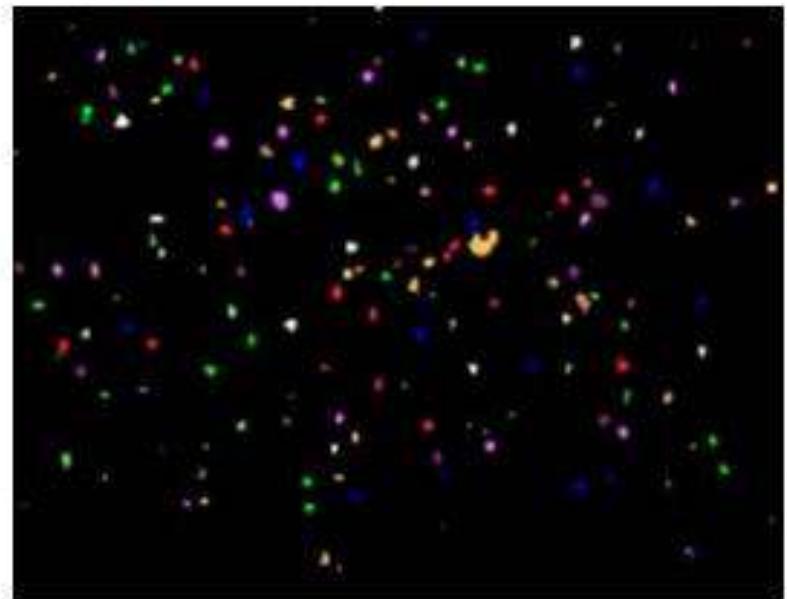


IE 6.19 An alternative representation of the intensity-slicing technique.

Intensity slicing into 8 colors



Gray image



8 different colors

Intensity slicing into 8 colors



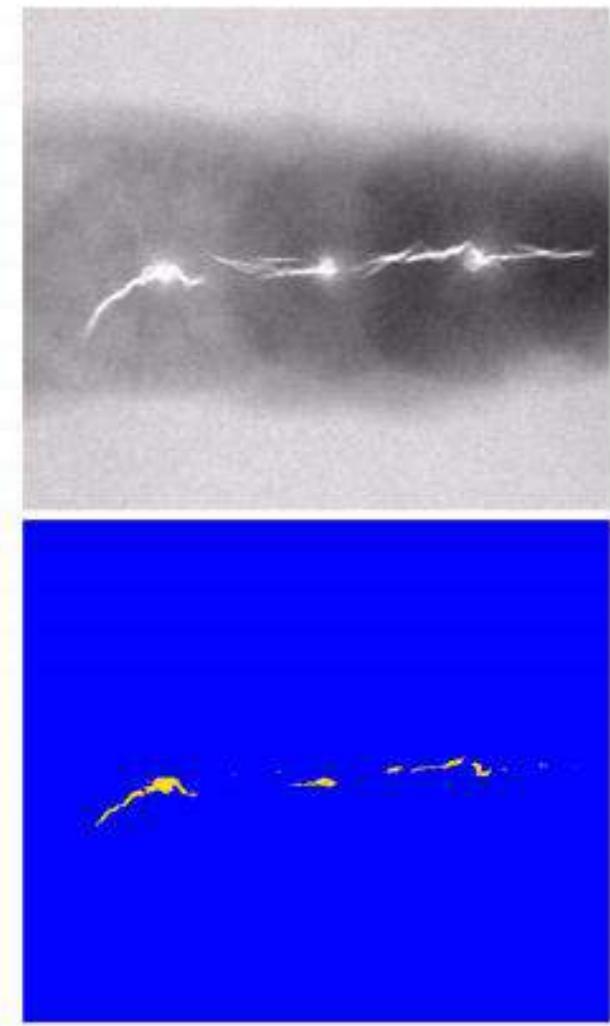
Gray image



8 different colors

```
e=imread('shades8.png');
eg = rgb2gray(e);
imshow(e)
% K R G B Y M C W
my8colors = [0 0 0; 1 0 0; 0 1 0; 0 0 1; 1 1 0; 1 0 1; 0 1 1; 1 1 1];
e8 = grayslice(eg, 8);
figure, imshow(e8, my8colors)
```

Intensity slicing into 2 colors



Hence, gray levels of value **255** in an 8-bit image coming from such a system automatically imply a **problem** with the weld!

Intensity slicing applications –maps الخرائط

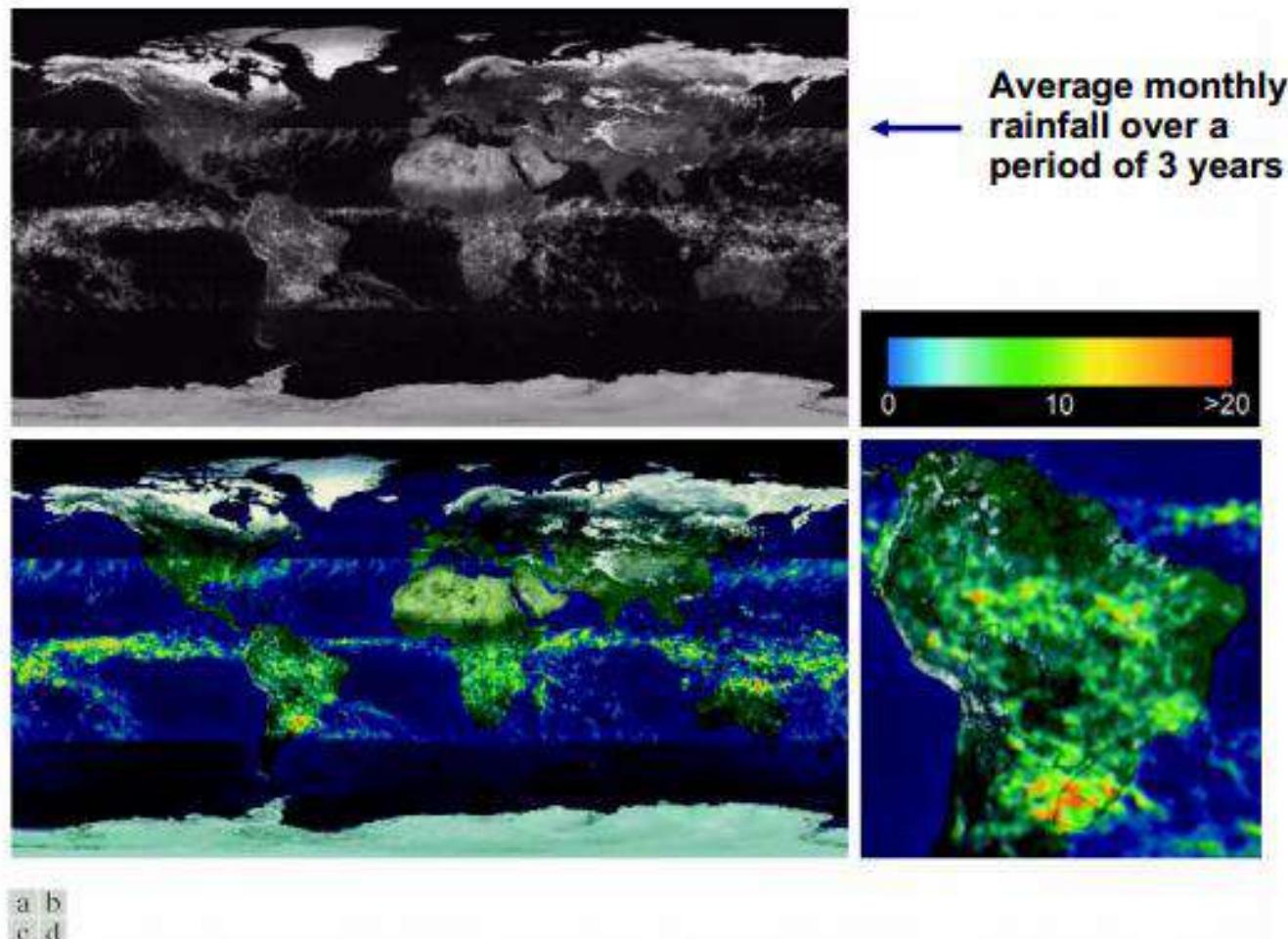


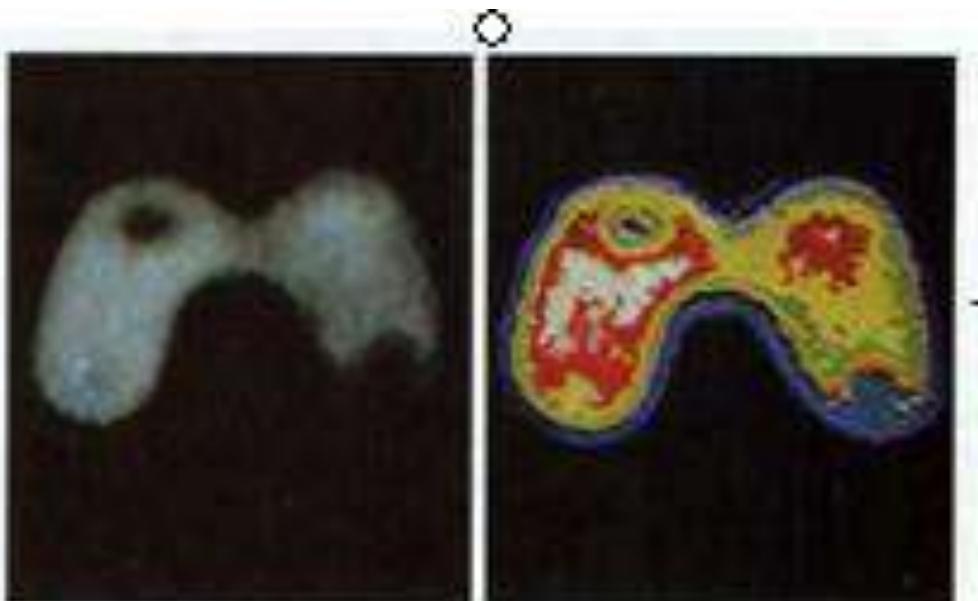
FIGURE 6.22 (a) Gray-scale image in which intensity (in the lighter horizontal band shown) corresponds to average monthly rainfall. (b) Colors assigned to intensity values. (c) Color-coded image. (d) Zoom of the South America region. (Courtesy of NASA.)

Intensity slicing applications – medicine

الطب

اختبار الغدة الدرقية Thyroid phantom:

عملنا slicing لثمانية الوان



→ A different color is assigned to each region without regard for the meaning of the gray levels in the image.

Gray level to color transformation

- Intensity slicing discussed in previous slide is very simple technique, there are other transforms that have better results.
- One technique is called: gray level to color transformation. It performs 3 independent transformations then combines them together into one resulting color image. The three results can then serve as the red, green, and blue components of a color image

- يعني لكي نحوال الصورة السكنية الى ملونة ، نعمل منها 3 صور سكنية اخرى ثم ندمجهم معا في صورة واحدة ملونة. كل صورة سكنية تدل على لون من الوان RGB
- نعتبر طريقة piecewise linear intensity slicing بينما هذه الطريقة nonlinear

Gray level to color transformation

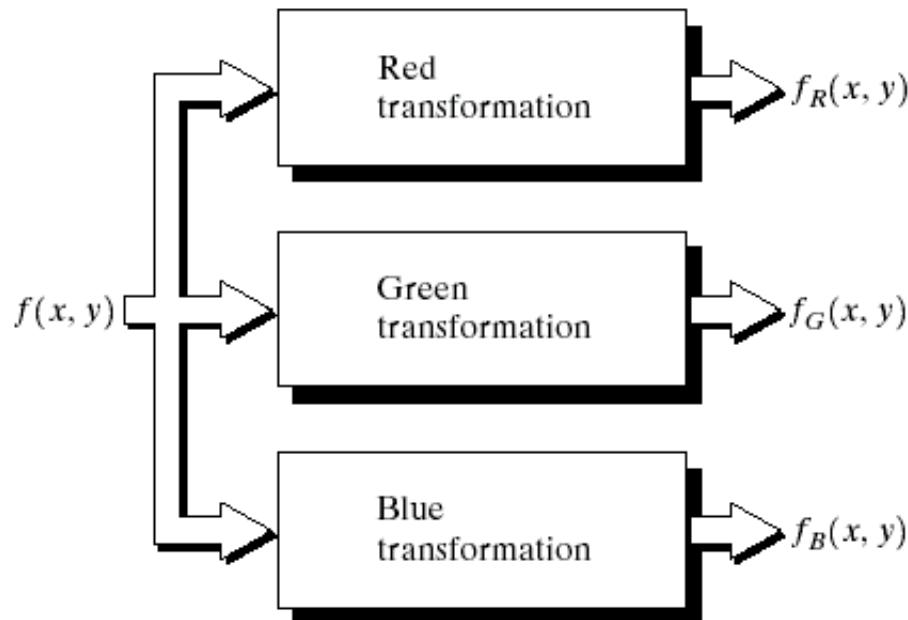


FIGURE 6.23 Functional block diagram for pseudocolor image processing. f_R , f_G , and f_B are fed into the corresponding red, green, and blue inputs of an RGB color monitor.

Basics of Full-Color Image Processing

Two major categories of full-color IP approaches

Each component image is processed individually, and a composite color image is formed from the components.

Color pixels (which are vectors) are directly processed.

Two conditions have to be satisfied for per-color-component and vector-based processing to be equivalent.

The process has to be applicable to both scalars and vectors.

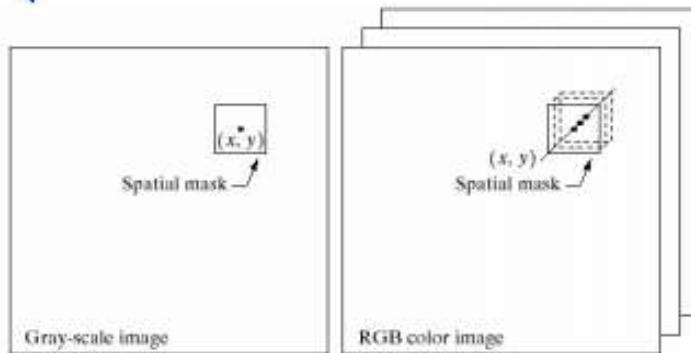
The operation on each component of a vector must be independent of the other components.

Example:

Suppose the process is neighborhood averaging.

Same result would be obtained using the **scalar and vector methods**.

a b
FIGURE 6.29
Spatial masks for gray-scale and RGB color images.



Color Transformation

- **General Trans:** $g(x,y) = T[f(x,y)]$:
pixels values are 3 or 4.
- **Special Case Trans:** $s_i = T_i(r_1, r_2, \dots, r_n)$:
 - s_i and r_i : variables denoting the color components
 - $\{T_1, T_2, \dots, T_n\}$: set of transformation functions
 - For RGB color space, $n=3$.
 - For CMYK color space, $n=4$.

Color Transformation

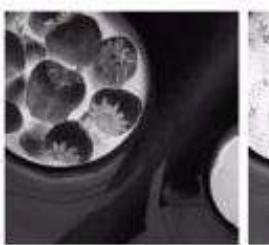
- Example: *intensity modification*
 - $g(x,y)=kf(x,y)$, $0 < k < 1$
 - Any color space can be used.
 - **RGB** color space: $s_i=kr_i$, $i=1,2,3$
 - **CMY** color space: $s_i=kr_i+(1-k)$, $i=1,2,3$
 - **HSI** color space: $s_3=kr_3$, $s_1=r_1$, $s_2=r_2$
 - In this case, although the HSI transformation involves the fewest # of operations, the computations required to convert an RGB or CMY(K) image into HSI more than offsets the advantages! (**The output is the same!**).

Color-Space Components of a Full-color image



Full color

FIGURE 6.30 A full-color image and its various color-space components. (Original image courtesy of Med-Data Interactive.)



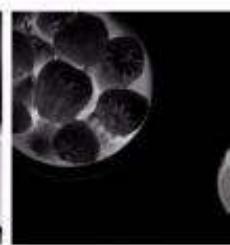
Cyan



Magenta



Yellow



Black

← CMYK



Red



Green



Blue

← RGB



Hue



Saturation



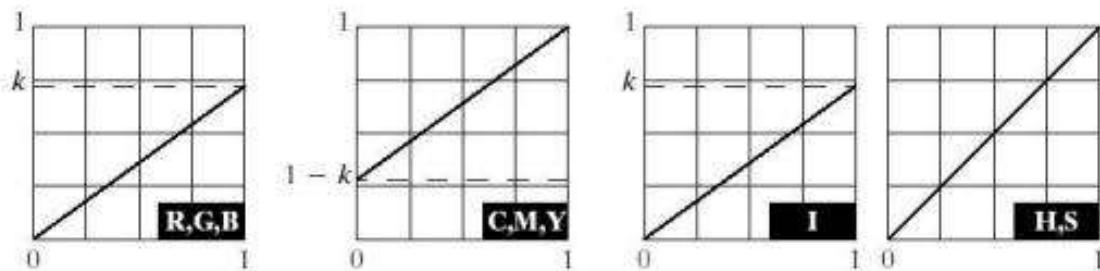
Intensity

← HSI

Modified Intensity of the Full-Color Image

a b
c d e

FIGURE 6.31
Adjusting the intensity of an image using color transformations.
(a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting $k = 0.7$).
(c)–(e) The required RGB, CMY, and HSI transformation functions.
(Original image courtesy of MedData Interactive.)



RGB color space: $s_i = kr_i, i=1,2,3$

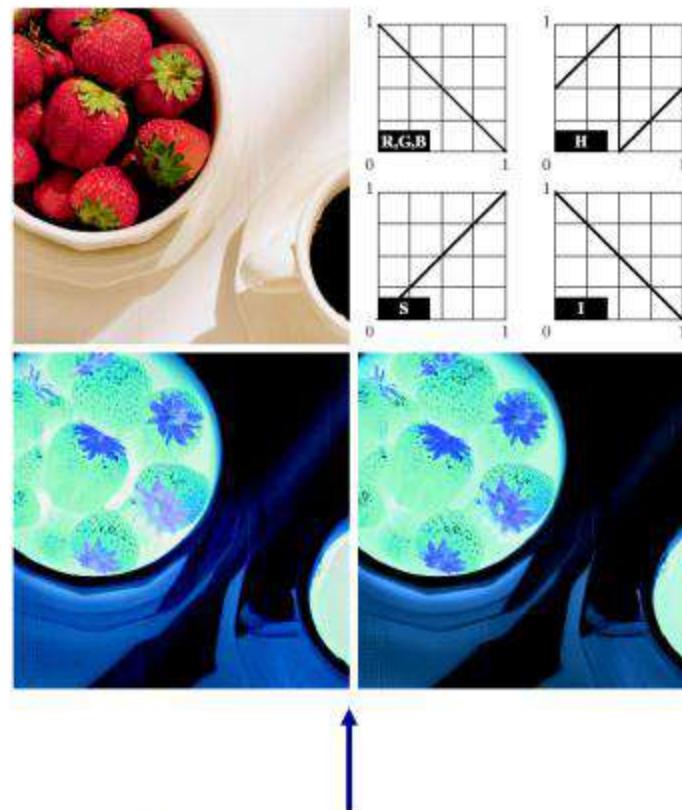
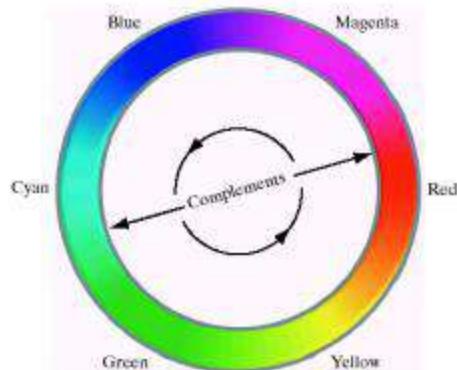
CMY color space: $s_i = kr_i + (1-k), i=1,2,3$

HSI color space: $s_3 = kr_3, s_1 = r_1, s_2 = r_2$

Color Complements

The hues directly opposite one another on the color circle are called **complements**.

Complements are analogous to gray-scale negatives: they are useful in enhancing detail embedded in dark region of a color image.



a
b
c
d

FIGURE 6.33
Color complement transformations.
(a) Original image.
(b) Complement transformation functions.
(c) Complement of (a) based on the RGB mapping functions.
(d) An approximation of the RGB complement using HSI transformations.

The RGB complement transformation functions used here do not have a straightforward HSI color space equivalent.

Tone and Color Correction

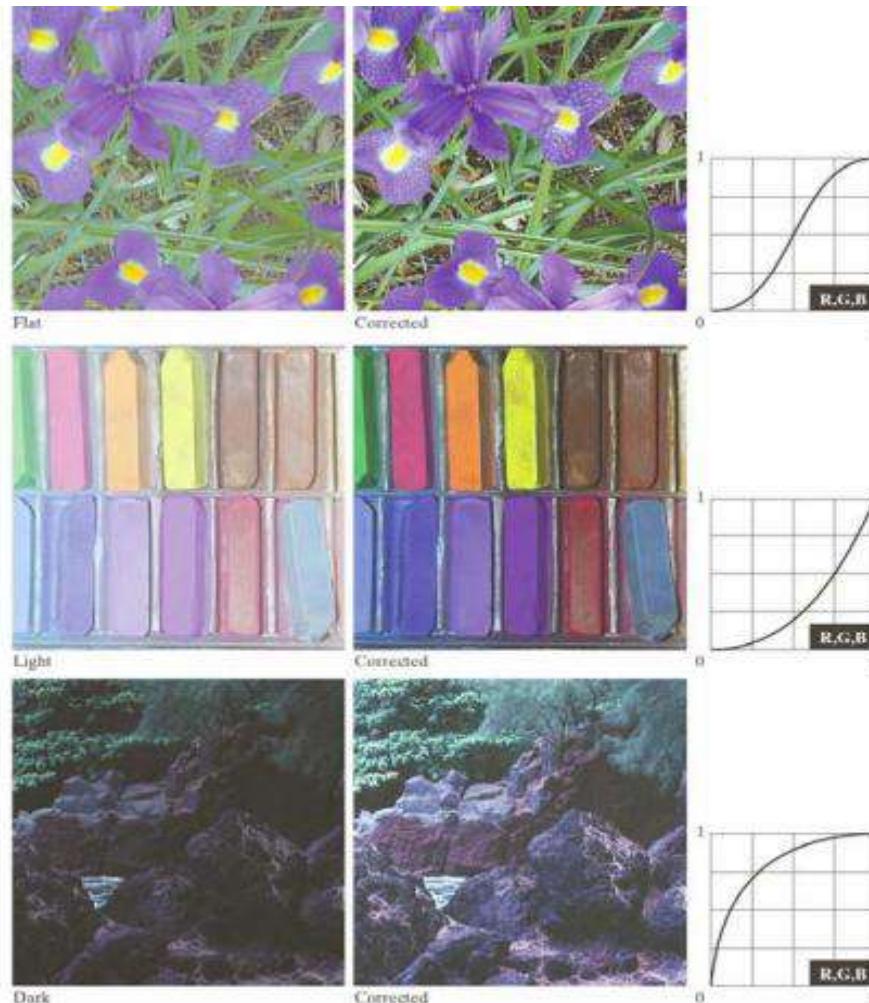


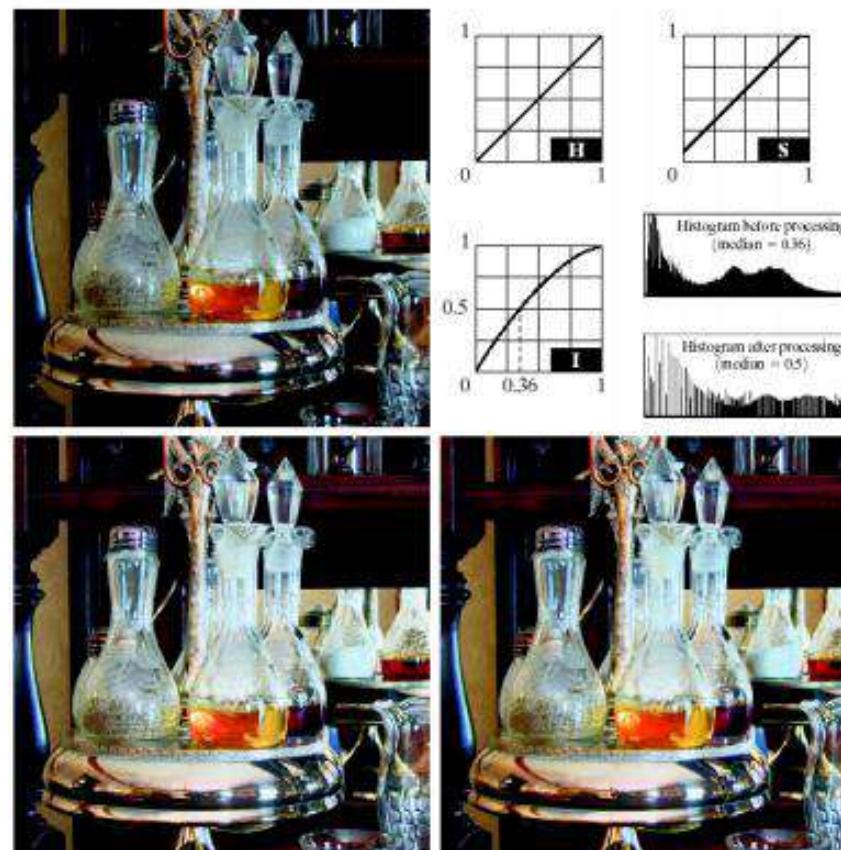
FIGURE 6.35 Tonal corrections for flat, light (high key), and dark (low key) color images. Adjusting the red, green, and blue components equally does not always alter the image hues significantly.

Histogram Processing

It is generally unwise to histogram equalize the components of a color image independently. This results in erroneous color.

A more logical approach is to spread the color intensities uniformly, leaving the colors themselves (e.g., hues) unchanged.

The HSI color space is ideally suited to this approach.



The intensity component is histogram equalized.

The saturation component is increased after histogram equalization

a
b
c
d

FIGURE 6.37
Histogram equalization (followed by saturation adjustment) in the HSI color space.

Color Image Smoothing

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

$$\bar{c}(x, y) = \left(\begin{array}{l} \frac{1}{K} \sum_{(x, y) \in S_{xy}} R(x, y) \\ \frac{1}{K} \sum_{(x, y) \in S_{xy}} G(x, y) \\ \frac{1}{K} \sum_{(x, y) \in S_{xy}} B(x, y) \end{array} \right) \quad \longleftarrow$$

Smoothing by neighborhood averaging can be carried out using either individual color planes or the RGB color vectors.

Smoothed Image

By smoothing only the intensity plane,
the pixels in the smoothed image
maintain their original hue and
saturation – and their original color!



a b c

FIGURE 6.40 Image smoothing with a 5×5 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 Image Sharpening

$$\nabla^2[c(x, y)] = \begin{bmatrix} \nabla^2 R(x, y) \\ \nabla^2 G(x, y) \\ \nabla^2 B(x, y) \end{bmatrix}$$

The Laplacian

The **Laplacian** of a full-color image can be obtained by computing the Laplacian of each component plane separately.

Sharpened Image

Obtained by combining the Laplacian of the intensity plane with the unmodified hue and saturation planes.



a b c

FIGURE 6.41 Image sharpening with the Laplacian. (a) Result of processing each RGB channel. (b) Result of processing the intensity component and converting to RGB. (c) Difference between the two results.

Matlab Example: RGB to r g b

```
%rgb2r-g-b
rgb=imread('colors.jpg');
imshow(rgb);
title('RGB image');
r= rgb(:,:,1);
figure, imshow(r)
title('RED image');
g= rgb(:,:,2);
figure,imshow(g)
title('GREEN image');
b= rgb(:,:,3);
figure,imshow(b)
title('BLUE image');
```

Matlab Example: RGB to r g b

```
function colourdisplay
%To demonstrate the splitting of an image into its primary
colours
A = imread('peppers.png');
subplot(2,2,1); imshow(A)
title('RGB image');
Redimage = A(:,:,1);
subplot(2,2,2); imshow(Redimage);
title('Red image');
Greenimage = A(:,:,2);
subplot(2,2,3); imshow(Greenimage);
title('Green image');
Blueimage = A(:,:,3);
subplot(2,2,4); imshow(Blueimage);
title('Blue image');
```

Matlab Example: Filtering Colored Images

```
%rgb2HSI and filter ... on I.  
rgb=imread('peppers.png');  
imshow(rgb);  
title('RGB image');  
h = rgb2hsi(rgb);  
H= h(:,:,1); S= h(:,:,2); I= h(:,:,3);  
w = fspecial('average', 25);  
I_filtered = imfilter(I, w, 'replicate');  
h = cat(3, H, S, I_filtered);  
f = hsi2rgb(h);  
f = min(f, 1);  
figure, imshow(f);
```

Image Processing



Ch7:
Image compression

IMAGE COMPRESSION

- Every day, an enormous amount of digital information is stored, processed and transmitted.
- The entire catalog of the Library of Congress (the world's largest library) is available electronically.
- The storage and communications requirements for information storage are immense!

MULTIMEDIA STORAGE REQUIREMENTS

- ❑ The importance of image compression is emphasized by the huge amount of data in raster images:
 - ❑ a typical gray-scale image of 512x512 pixels, each represented by 8 bits, contains 256 kilobytes of data.
 - ❑ With the color information, the number of bytes is tripled.

MULTIMEDIA STORAGE REQUIREMENTS

- ❑ If we talk about video images of 25 frames per second, even a one second of colored film requires approximately 19 megabytes of memory
- ❑ Thus, the necessity for compression is obvious.

IMAGE COMPRESSION PURPOSE

- ❑ The purpose of compression is to code the image data into a compact form, minimizing both the number of bits in the representation, and the distortion caused by the compression.

IMAGE COMPRESSION TECHNIQUES

- Two broad categories of compression techniques:
 - Lossless techniques (or *information preserving*, or *reversible*)
 - No information is lost. (100% reconstruction of the original data.)
 - Useful in image archiving (e.g., storage of legal or medical records)
 - Lossy techniques (or *irreversible*)
 - A certain amount of information loss is acceptable.
 - Useful in applications such as broadcast TV, videoconferencing, and facsimile transmission.

Performance criteria in image compression

The three main criteria of measuring the performance of an image compression algorithm are:

- ***Compression efficiency.***
- ***Distortion caused by the compression algorithm.***
- ***The speed of the compression and decompression process.***

Performance criteria: Compression efficiency

There are various ways of expressing the amount of compression that has been achieved. One way is to compute the **compression ratio**, defined as

$$C = \frac{n}{n_c}$$

where **n** is the number of information carrying units used in the uncompressed dataset and **n_c** is the number of units in the compressed dataset.

Larger values of C indicate better compression.

Performance criteria: Distortion

- **Subjective**, if the quality is evaluated by humans.
- **objective** measures the distortion which is calculated as the *difference* between the original and the reconstructed image by a predefined function.
 - mean absolute error (MAE).
 - mean square error (MSE).
 - peak-to-peak signal to noise ratio (PSNR).

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - x_i|$$

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - x_i)^2$$

$$\text{PSNR} = 10 \cdot \log_{10} [255^2 / \text{MSE}] \quad (\text{assuming } k=8)$$

Lossless Compression Techniques:

- Run-Length Encoding (RLE).**
- Huffman Coding.**
- BITPLANE Coding.**
- Golomb-Rice Coding.**
- Arithmetic Coding.**
- LEMPEL-ZIV.**

Lossy Compression Techniques:

❑ Transforms.

- Discrete Cosine Transform (DCT) -> JPEG**
- Wavelets JPEG2000.**

Lossy Compression Techniques:

- 1) Block Transform coding /*using DCT*(JPEG)
- 1) Wavelet transform coding (JPEG 2000)

1) Block transform coding (JPEG)

Some compression techniques are based on modifying the transform of the image.

Transform coding: *using a reversible , linear transform (like Fourier) to map the image into a set of transform coefficients, which are then quantized and coded.*

For most natural images, a significant number of coefficients have small magnitudes and can be quantized (or discarded) , and this may cause little non visual distortion.

اي من الممكن ضغط الصور بتحويل الصورة من spatial الى اي نوع اخر مثل Fourier ومن ثم التعديل عليه ، واهمال بعض قيمها (لتحقيق ضغط الصورة) وبعدها اعادة الصورة الى spatial. وسيتسبب اهمال بعض القيم باختلاف الصورة المضغوطة عن الاصلية قليلا ولكنه تغيير غير ملحوظ.

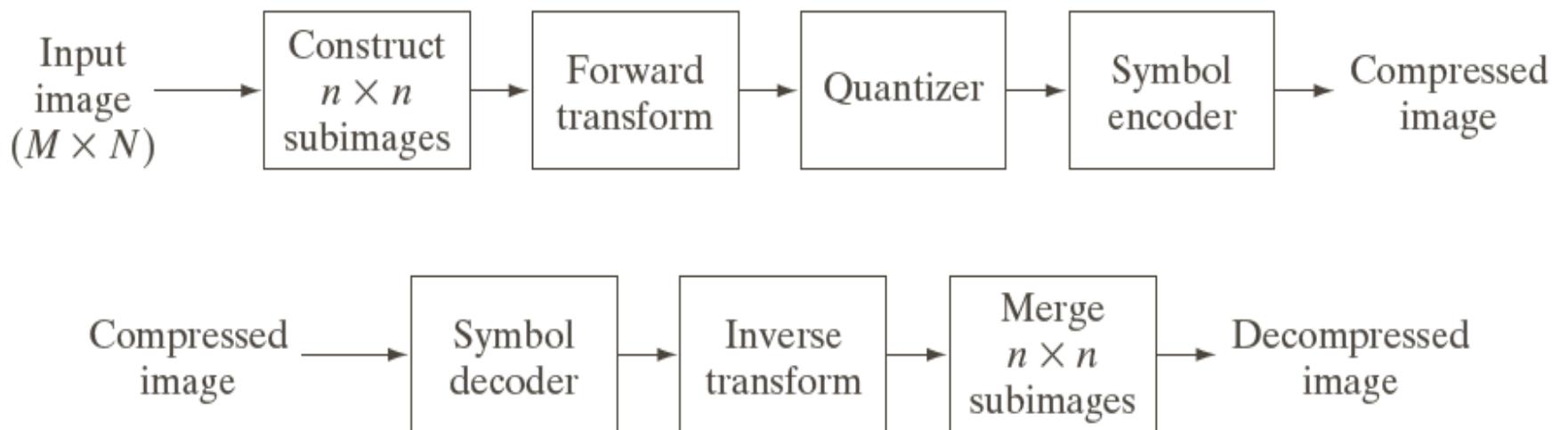
Block transform coding

there are a variety of transformations, including:

1. DFT(discrete Fourier transform)
2. DCT (discrete cousin transform)

كما ذكرنا ، فاننا سنحول الصورة من SPATIAL الى نوع اخر مثل: DFT أو DCT او,,الخ, ومن ثم سنعدل الصورة بان نحذف بعض المعلومات و من ثم سنرجعها الى SPATIAL لتحقق ضغط الصورة، (كما في الشكل التالي). واختيار نوع ال TRANSFORMATION يعتمد على بعض العوامل التي سنذكرها في الشريحة التالية.

Encoder and Decoder



Block transform coding

□ نقوم بطريقة block transform coding او لا بتقسيم الصورة الى Fourier(DFT) من $8 * 8$ مثلاً، ومن ثم تحويله الى نوع اخر مثل (blocks أو DCT وبعدها عمل quantization أي حذف بعض المعلومات الغير مهمة وبعد ان نحصل على الصورة المضغوطة ممكن ارجاعها الى spatial, بعد ان تكون قد دمجنا ال blocks. (في السไลدات التالية توضيح لهذه الخطوات)

□ تحويل الصورة من شكل الى اخر (اي من Fourier الى spatial مثلاً) ليس هو المسؤول عن ضغط الصورة. بل مرحلة ال quantization التي تلي مرحلة التحويل هي المسؤولة عن ضغط الصورة.

Transform selection

- As we mentioned in the last slide, when we compress the image, we can convert it to a some 2D transformation (like DFT,DCT,..etc). But how to select the most appropriate transform to obtain the best compression??
- **The choice of a particular transform depends on:**
 - 1) the amount of reconstruction error that can be tolerated (يمكن ان يُحتمل).
 - 2) the computational resources available.
- **Important Note: Compression is achieved during the quantization not during the transformation step.**

اي ان تحويل الصورة من شكل الى اخر (اي من spatial الى Fourier مثلا) ليس هو المسؤول عن ضغط الصورة. بل مرحلة ال quantization التي تلي مرحلة التحويل هي المسؤولة عن ضغط الصورة.

Why DCT ??

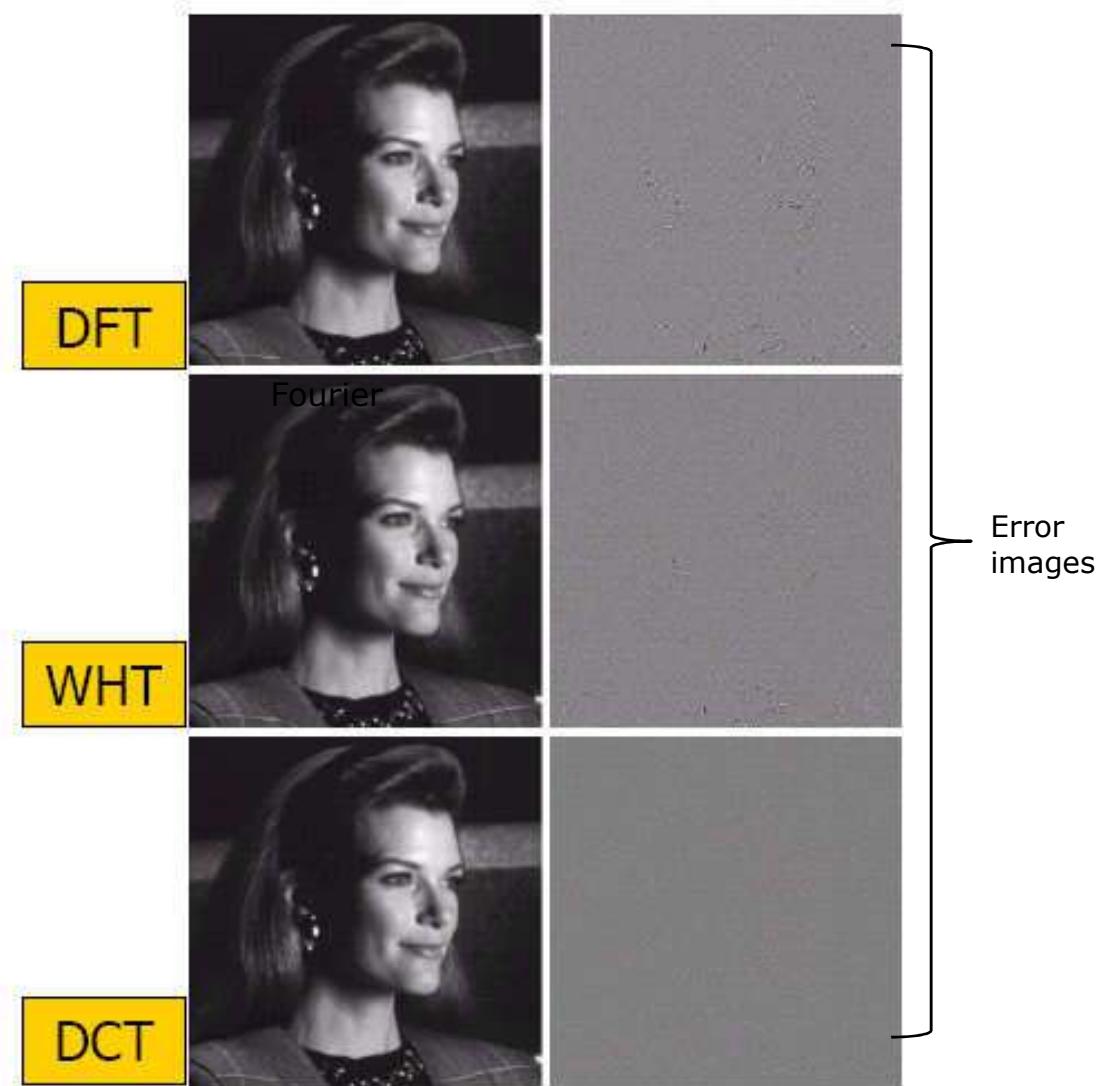
DCT creates a frequency-based representation of the image discarding some of the high frequencies to create redundancy and hence achieve compression.

DCT achieves a set of real-valued coefficients.

The DCT is preferred to the DFT because it packs a given amount of information into fewer coefficients.

Comparison between 3 transforms results

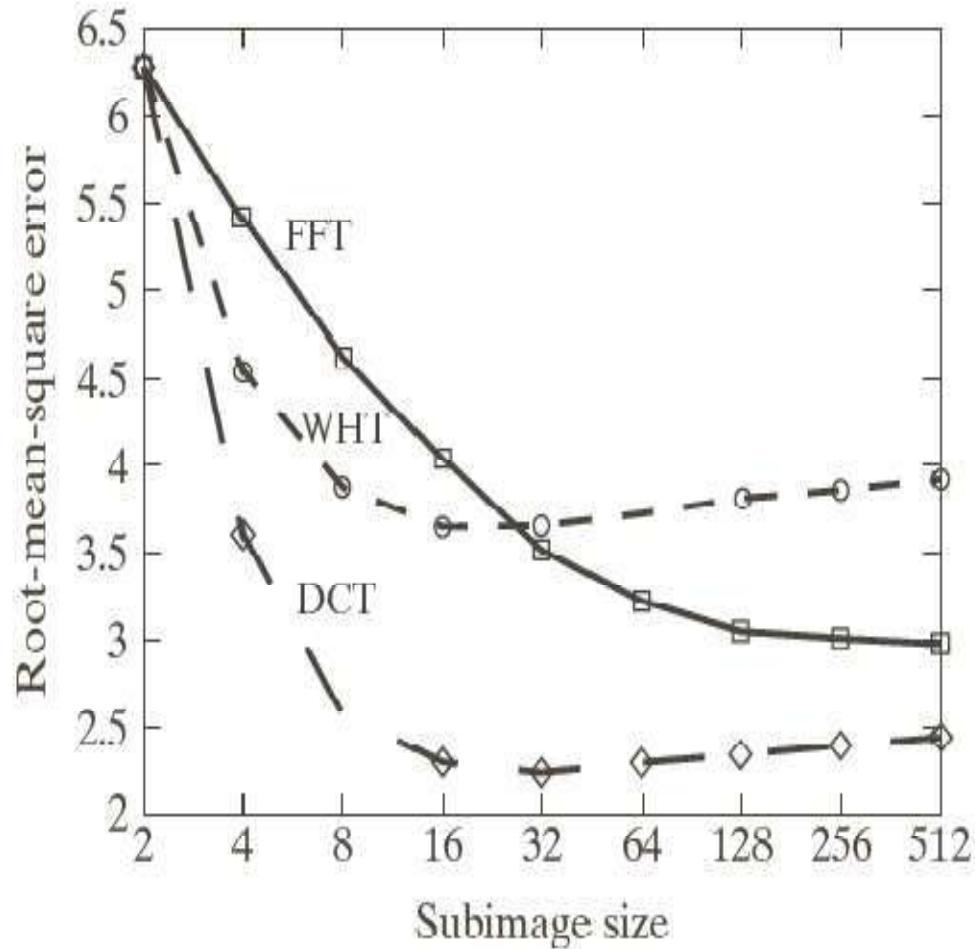
The same image is compressed using three transforms, DFT, WHE, DCT



Compare between the error images in the right.

The best result for error image is for DCT.

Comparison between 3 transforms RMSE *(root mean square error)*



Why Blocks ??

JPEG on an entire image **Disadvantages:**

- ❑ It is demanding computationally;
- ❑ Discarding high frequencies from the spectrum generated by the transform will have the effect of a low pass filter, blurring all parts of the image to the same degree.

Solution:

- ❑ Perform the transform on small areas (**blocks**) of the image, e. g. 8x8.

Transform selection – cont.

Consider image $f(x,y)$ of size $N \times N$. and suppose that the discrete transform to be applied on it is: $T(u,v)$.

$T(u,v)$ transform is obtained by the following formula:

$$T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y)g(x, y, u, v) \quad \dots \text{ For } u, v = 0, 1, 2, \dots, N-1.$$

$f(x,y)$, similarly, is obtained by the following formula:

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v)h(x, y, u, v) \quad \dots \text{ For } x, y = 0, 1, 2, \dots, N-1.$$

- ✓ In the above equations, $g(x, y, u, v)$ and $h(x, y, u, v)$ are called the **forward and inverse transformation kernels** (or **basis functions** or basis images)
- ✓ The $T(u, v)$ for $u, v = 0, 1, 2, \dots, N-1$ are called **Transform coefficients**

DCT (Discrete Cosine Transform)

A Transform that can be applied on images, and can be given in the following formula:

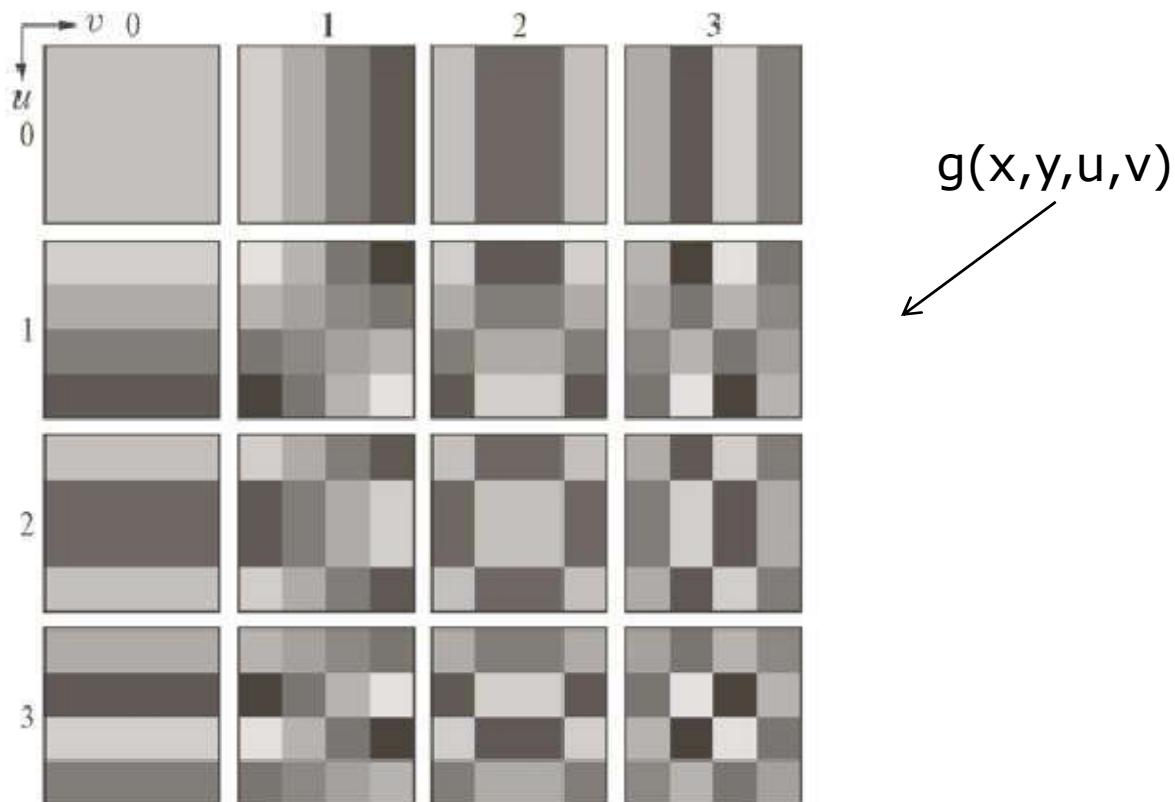
$$\begin{aligned}g(x, y, u, v) &= h(x, y, u, v) \\&= \alpha(u)\alpha(v) \cos\left[\frac{(2x + 1)u\pi}{2N}\right] \cos\left[\frac{(2y + 1)v\pi}{2N}\right]\end{aligned}$$

where

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

DCT (cont'd)

- ❑ Basis set of functions for a 4x4 image (i.e., cosines of different frequencies).

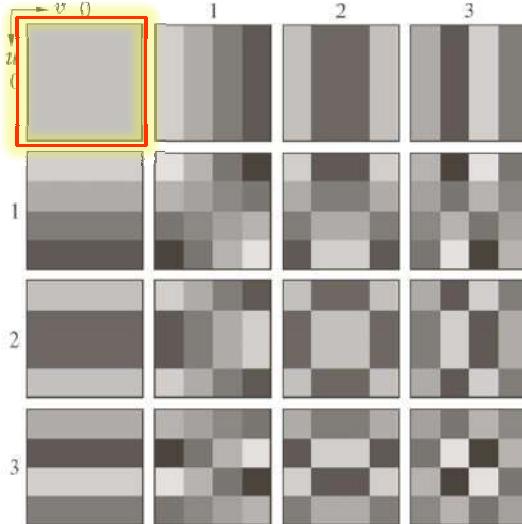


كيف يُعمل DCT؟

على افتراض عندنا صورة $f(x,y)$ ، فاننا نقوم بتقسيمها الى صور صغيرة فرعية بحجم 4×4 أو 8×8 ... ومن ثم لكل **block** نقوم بتطبيق المعادلة التي درسناها بسلайд 5,

$$T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y)g(x, y, u, v)$$

فمثلا لكي نحصل على $T(0,0)$ (وهذا الحد يسمى *dc* الصورة ويحمل كل طاقة الصورة مثل *Fourier*) نقوم بجمع ناتج ضرب كل بكسل بالصورة ب $g(x,y,0,0)$ (المربع الاحمر)، اي ستضرب كل بكسلات الصورة بأول مربع وتجمع هذه نتائج الضرب ، وهذا ينتج اول قيمة في **DCT transform** للصورة. ونكرر هذه العملية للحصول على $T(0,1)$ حيث تضرب كل بكسلات الصوره بالمربع الثاني و تجمع النتائج.. وتكرر العملية لآخر قيمة وهي $T(4,4)$ ، ومن ثم نكرر DCT للصورة الفرعية الثانية، فالثالثة، الى ان ننتهي و هكذا تكون قد حصلنا على تمثيل DCT لجميع الصور الفرعية للصورة الاصلية.



DCT .. Sub image size selection

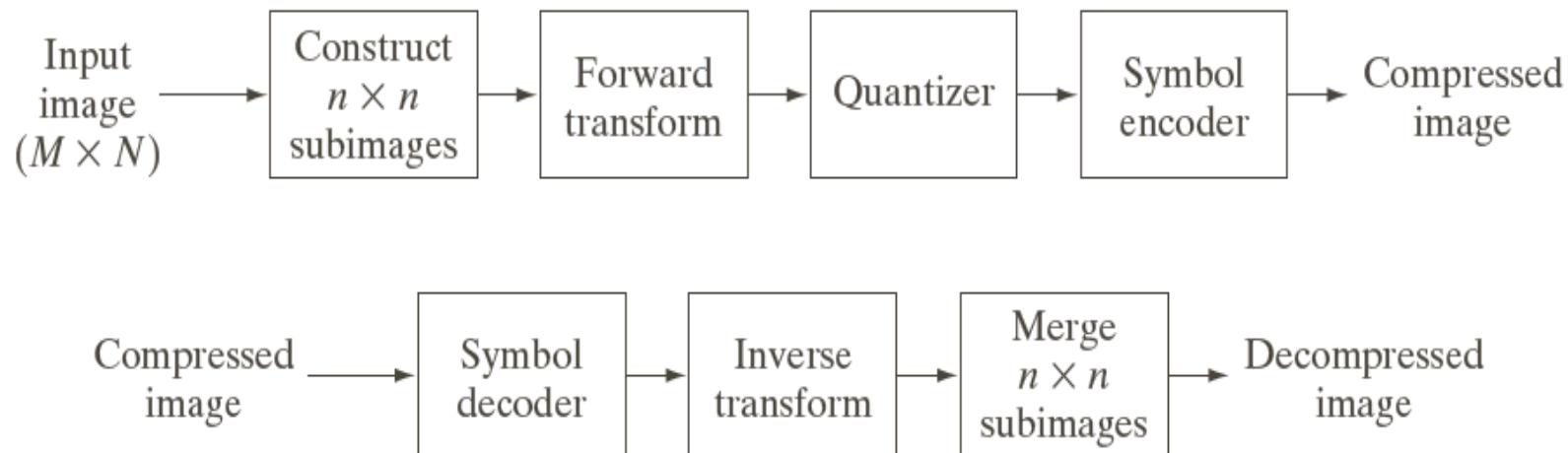
- We mentioned in the previous slide that the image is divided into sub images, but how to select the most appropriate sub images size?
- In most applications, images are divided to reduce the redundancy between adjacent sub images. the sub image size n must be an integer power of 2. (The size should be power of 2 مربعات كاملة to simplify the computation. the most popular sub image sizes are: 8x8, 16x16

□ ملاحظة: كل ما زاد حجم الصور الفرعية ,sub image size كل ما زادت درجة الضغط .computation complexity ، و ايضاً ستزيد الحسابات المعقدة compression level

Sub image size ↑ .. compression level ↑ .. computation complexity ↑

DCT and Image Compression

Remember from slide 4 that image compression using block transform coding contains the following steps:



So we studied the first step (constructing sub images), and the second step (forward transform –DCT). Then the third step is **quantization**. Which means to set the frequency coefficients in the transformed matrix whose intensities are less than a defined threshold to zero (these coefficients cannot be recovered during decoding), these intensities are not visualized by human eye, so the compressed image will be very close to the original image with a little error of course.

DCT and Image Compression –

- JPEG compression as an example (lossy)

JPEG compression uses DCT as its transform.

In the **JPEG** image compression algorithm,

1. the input image is divided into 8-by-8 or 16-by-16 blocks
2. The quantity 2^{n-1} is subtracted from each pixel value
3. then two-dimensional **DCT** is computed for each block.
4. The DCT coefficients are then quantized, coded, and transmitted.
5. The JPEG receiver (or JPEG file reader) decodes the quantized DCT coefficients,
6. It computes the inverse two-dimensional DCT of each block,
7. The quantity 2^{n-1} is added to each pixel value
8. then it puts the blocks back together into a single image.

For typical images, many of the DCT coefficients have values close to zero; these coefficients can be **discarded** without seriously affecting the quality of the reconstructed image.

DCT and Image Compression : JPEG compression as an example (lossy compression)

Although there will be some loss of quality in the reconstructed image, it will be clearly recognizable.

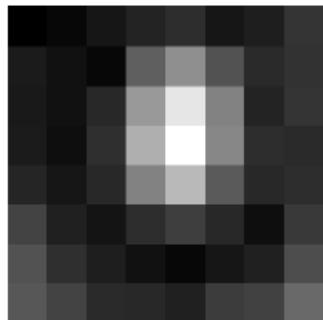
The following example shows cameraman compressed image (in the right) and it is recognizable although almost 85% of the DCT coefficients were discarded.



Image Courtesy of MIT

Quantizer

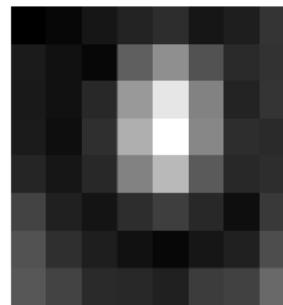
- note that we can quantize some frequency coefficients more heavily than others by simply increasing Q
- this leads to the idea of a quantization matrix
- we start with an image block (e.g. 8x8 pixels)



52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Quantizer

- next we apply a transform (e.g. 8x8 DCT)



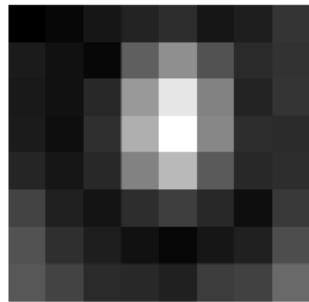
52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

DCT

-415	-30	-61	27	56	-20	-2	0
4	-22	-61	10	13	-7	-9	5
-47	7	77	-25	-29	10	5	-6
-49	12	34	-15	-10	6	2	2
12	-7	-13	-4	-2	2	-3	3
-8	3	2	-6	-2	1	4	2
-1	0	0	-2	-1	-3	4	-1
0	0	-1	-4	-1	0	1	2

Quantizer

- and quantize with a varying Q



52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

DCT

-415	-30	-61	27	56	-20	-2	0
4	-22	-61	10	13	-7	-9	5
-47	7	77	-25	-29	10	5	-6
-49	12	34	-15	-10	6	2	2
12	-7	-13	-4	-2	2	-3	3
-8	3	2	-6	-2	1	4	2
-1	0	0	-2	-1	-3	4	-1
0	0	-1	-4	-1	0	1	2

Q mtx

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

$$x_q = \text{round}\left(\frac{x}{Q}\right)$$

نطبق هذه المعادلة و النتيجة
بالشريحة اللاحقة

Quantizer

- note that higher frequencies are quantized more heavily

Q mtx	[16 11 10 16 24 40 51 61]
	[12 12 14 1 14 16 24 40 57 69 56]
	[14 17 22 29 51 87 80 62]
	[18 22 37 56 68 109 103 77]
	[24 35 55 64 81 104 113 92]
	[49 64 78 87 103 121 126 101]
	[72 92 95 98 112 100 103 99]

- in result, many high frequency coefficients are simply wiped out

DCT	quantized DCT
[-415 -30 -61 27 56 -20 -2 0]	[-26 -3 -6 2 2 -1 0 0]
4 -22 -61 10 13 -7 -9 5	0 -2 -4 1 1 0 0 0
-47 7 77 -25 -29 10 5 -6	-3 1 5 -1 -1 0 0 0
-49 12 34 -15 -10 6 2 2	-4 1 2 -1 0 0 0 0
12 -7 -13 -4 -2 2 -3 3	1 0 0 0 0 0 0 0
-8 3 2 -6 -2 1 4 2	0 0 0 0 0 0 0 0
-1 0 0 -2 -1 -3 4 -1	0 0 0 0 0 0 0 0
0 0 -1 -4 -1 0 1 2	0 0 0 0 0 0 0 0

DCT	quantized DCT
[-415 -30 -61 27 56 -20 -2 0]	[-26 -3 -6 2 2 -1 0 0]
4 -22 -61 10 13 -7 -9 5	0 -2 -4 1 1 0 0 0
-47 7 77 -25 -29 10 5 -6	-3 1 5 -1 -1 0 0 0
-49 12 34 -15 -10 6 2 2	-4 1 2 -1 0 0 0 0
12 -7 -13 -4 -2 2 -3 3	1 0 0 0 0 0 0 0
-8 3 2 -6 -2 1 4 2	0 0 0 0 0 0 0 0
-1 0 0 -2 -1 -3 4 -1	0 0 0 0 0 0 0 0
0 0 -1 -4 -1 0 1 2	0 0 0 0 0 0 0 0

Quantizer

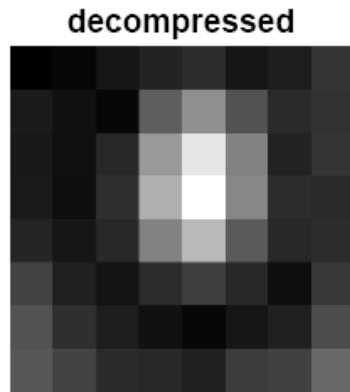
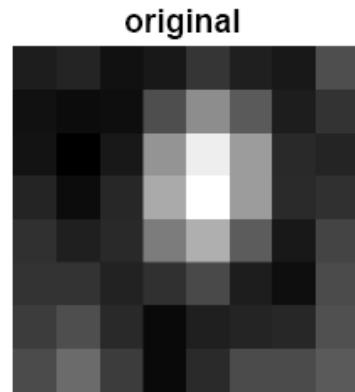
- this saves a lot of bits, but we no longer have an exact replica of original image block

DCT	quantized DCT
[-415 -30 -61 27 56 -20 -2 0]	[-26 -3 -6 2 2 -1 0 0]
4 -22 -61 10 13 -7 -9 5	0 -2 -4 1 1 0 0 0
-47 7 77 -25 -29 10 5 -6	-3 1 5 -1 -1 0 0 0
-49 12 34 -15 -10 6 2 2	-4 1 2 -1 0 0 0 0
12 -7 -13 -4 -2 2 -3 3	1 0 0 0 0 0 0 0
-8 3 2 -6 -2 1 4 2	0 0 0 0 0 0 0 0
-1 0 0 -2 -1 -3 4 -1	0 0 0 0 0 0 0 0
0 0 -1 -4 -1 0 1 2	0 0 0 0 0 0 0 0

inverse DCT	original pixels
[60 63 55 58 70 61 58 80]	[52 55 61 66 70 61 64 73]
58 56 56 83 108 88 63 71	63 59 55 90 109 85 69 72
60 52 62 113 150 116 70 67	62 59 68 113 144 104 66 73
66 56 68 122 156 116 69 72	63 58 71 122 154 106 70 69
69 62 65 100 120 86 59 76	67 61 68 104 126 88 68 70
68 68 61 68 78 60 53 78	79 65 60 70 77 68 58 75
74 82 67 54 63 64 65 83	85 71 64 59 55 61 65 83
83 96 77 56 70 83 83 89	87 79 69 68 65 76 78 94

Quantizer

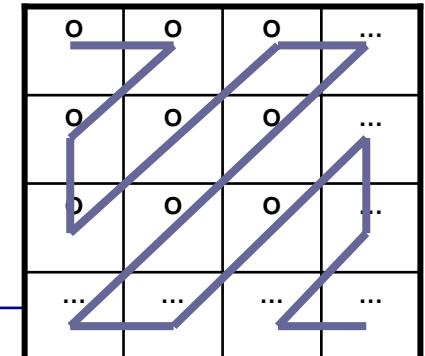
- note, however, that visually the blocks are not very different



- we have saved lots of bits without much “perceptual” loss
- this is the reason why JPEG and MPEG work

JPEG Compression: STEPS

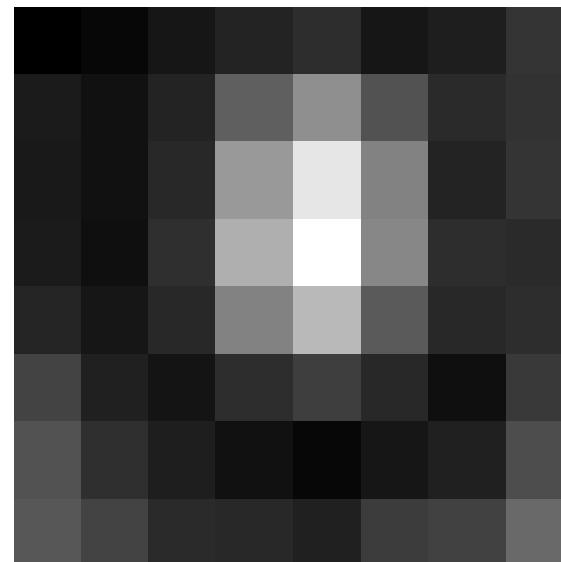
- ❑ For each 8x8 image block, perform the following steps:
 - Subtracting 128 from each value (for $n=8 \Rightarrow 2^{n-1} = 128$)
 - Apply DCT
 - Normalization by dividing by matrix Q (**the lossy part**
 - Turning most of the elements zero)
 - Change the matrix into a vector by reading the nonzero element in a **zigzag** fashion.



JPEG De-Compression: STEPS

- Inverse the operations:
 - Decode Huffman encoding and RLE
 - Put the vector back to 8x8 matrix
 - Multiply by Q
 - Inverse DCT
 - Shift back by 128

A CASE STUDY: TEST SUB-IMAGE (8 x 8)



A CASE STUDY: AN 8X8 BLOCK

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

A CASE STUDY: LEVEL SHIFTING

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-59	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-67	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

The quantity 2^{n-1} is subtracted from each pixel value.

$$n=8 \Rightarrow 2^{n-1} = 128$$

A CASE STUDY: APPLICATION OF DCT

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

A CASE STUDY: NORMALIZATION MATRIX

$Z(u, v)$

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

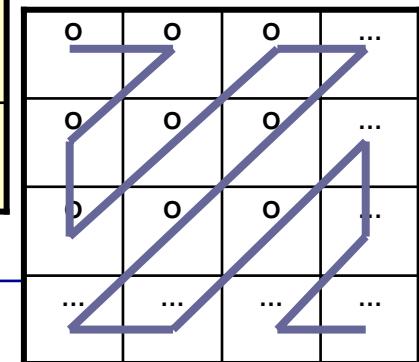
A CASE STUDY: QUANTIZATION

DCT coefficients
are quantized
using the formula

$$\hat{T}(u,v) = \text{round} \left[\frac{T(u,v)}{Z(u,v)} \right]$$

38 consecutive
zeros!

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



A CASE STUDY: ZIGZAG REORDERING

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



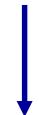
a special **EOB**
Huffman code word
indicates that the
remainder of the
coefficients are
zeros.

A CASE STUDY: COMPLETELEY CODED ARRAY

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001



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



001 100101 11100110 110110 0110 11110100 000 1010

of bits needed to store the 8x8 block = $64 \times 8 = 512$

of bits after JPEG compression = 92

Compression ratio = $512/92 \Rightarrow 5.6:1$

A CASE STUDY: DECOMPRESSION BEGINS

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

A CASE STUDY: DENORMALIZATION

$$\dot{T}(u, v) = \hat{T}(u, v)Z(u, v)$$

-416	-33	-60	32	48	0	0	0
12	-24	-56	0	0	0	0	0
-42	13	80	-24	-40	0	0	0
-56	17	44	-29	0	0	0	0
18	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

A CASE STUDY: INVERSE DCT

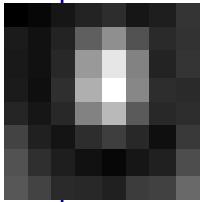
-70	-64	-61	-64	-69	-66	-58	-50
-72	-73	-61	-39	-30	-40	-54	-59
-68	-78	-58	-9	13	-12	-48	-64
-59	-77	-57	0	22	-13	-51	-60
-54	-75	-64	-23	-13	-44	-63	-56
-52	-71	-72	-54	-54	-71	-71	-54
-45	-59	-70	-68	-67	-67	-61	-50
-35	-47	-61	-66	-60	-48	-44	-44

A CASE STUDY: LEVEL SHIFTING

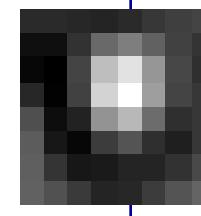
The quantity 2^{n-1}
is added from
each pixel value.
 $n=8 \Rightarrow 2^{n-1} = 128$

58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

A CASE STUDY: DIFFERENCE



52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94



58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84



-6	-9	-6	2	11	-1	-6	-5
7	4	-1	1	11	-3	-5	3
2	9	-2	-6	-3	-12	-14	9
-6	7	0	-4	-5	-9	-7	1
-7	8	4	-1	11	4	3	-2
3	8	4	-4	2	11	1	1
2	2	5	-1	-6	0	-2	5
-6	-2	2	6	-4	-4	-6	10

A CASE STUDY: Matlab: compress

block = [

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94];

b = double(block) - 128;

bd = dct2(b);

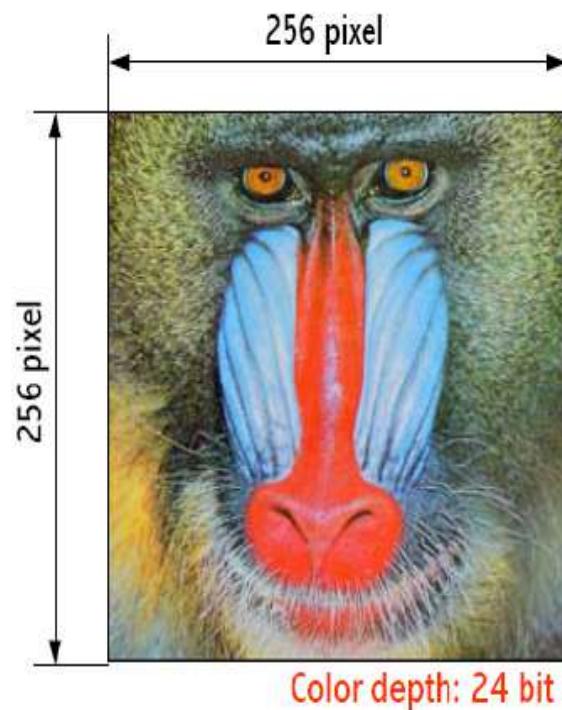
A CASE STUDY: Matlab: compress

```
q=[16 11 10 16 24 40 51 61;...  
    12 12 14 19 26 58 60 55;...  
    14 13 16 24 40 57 69 56;...  
    14 17 22 29 51 87 80 62;...  
    18 22 37 56 68 109 103 77;...  
    24 35 55 64 81 104 113 92;...  
    49 64 78 87 103 121 120 101;...  
    72 92 95 98 112 100 103 99];  
  
bq = round(bd./q);
```

A CASE STUDY: Matlab: Uncompress

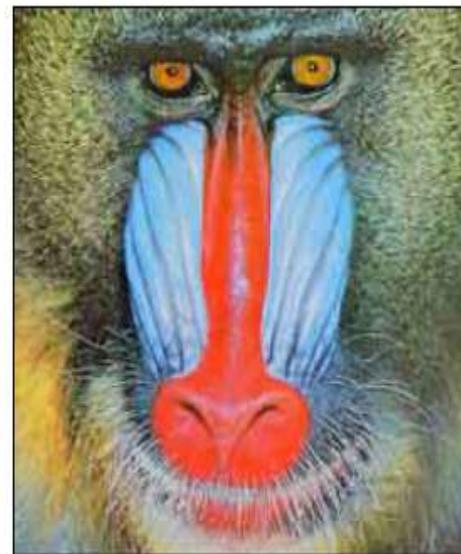
```
bq2 = bq.*q;  
bd2 = idct2(bq2)  
b2 = round(bd2+128)  
% diff  
double(block) - double(b2)
```

JPEG EXAMPLE



$$\text{Size of Data Chunk} = 256 \cdot 256 \cdot 3 = 192 \text{ kB}$$

Compressed image (JPEG)



File Size = 18.7 kB

Compression ratio = 10,27 !!

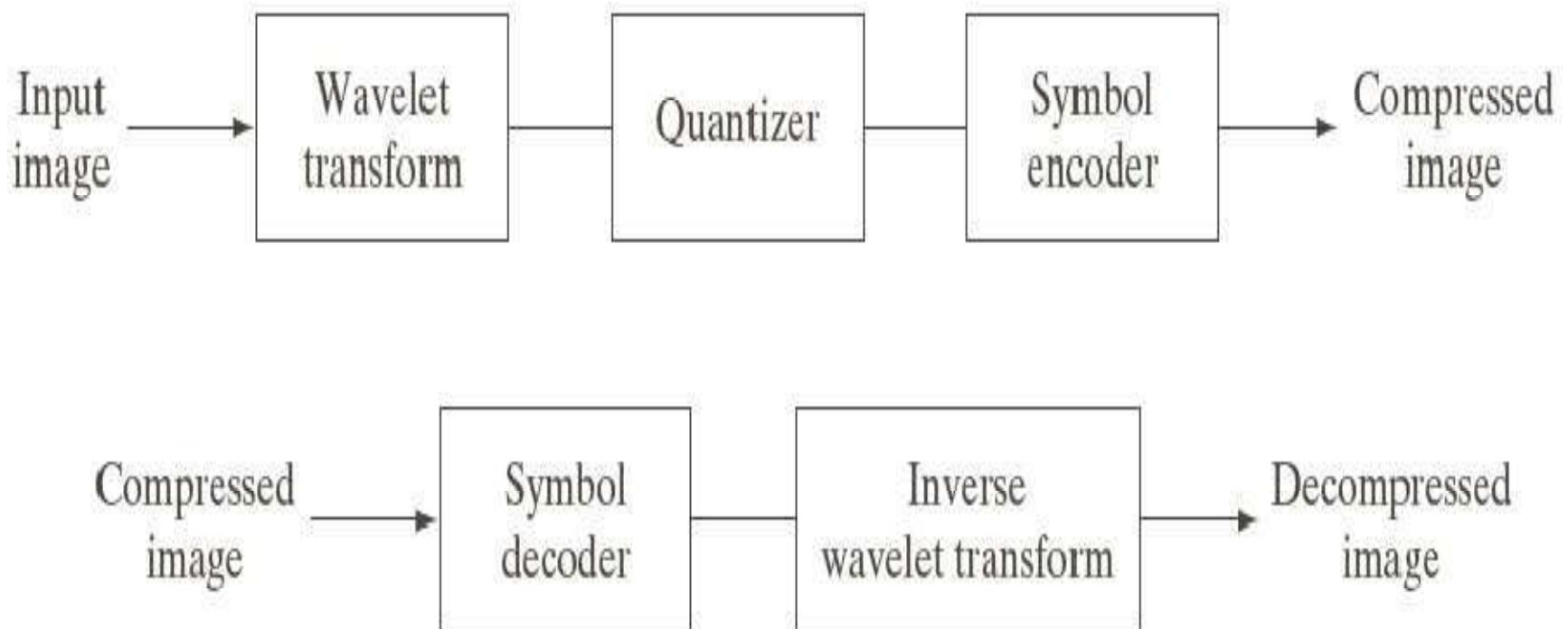
2) Wavelet Coding (JPEG 2000)

Wavelet coding is another technique used in compression, and is based on the idea that the transform coefficients of the images pixels can be coded more efficiently than the original pixels.

some wavelets of the image pack most of its visual information into small number of coefficients, the remaining coefficients can be quantized into zero with a little image distortion.

Wavelet Coding

Encoder and Decoder



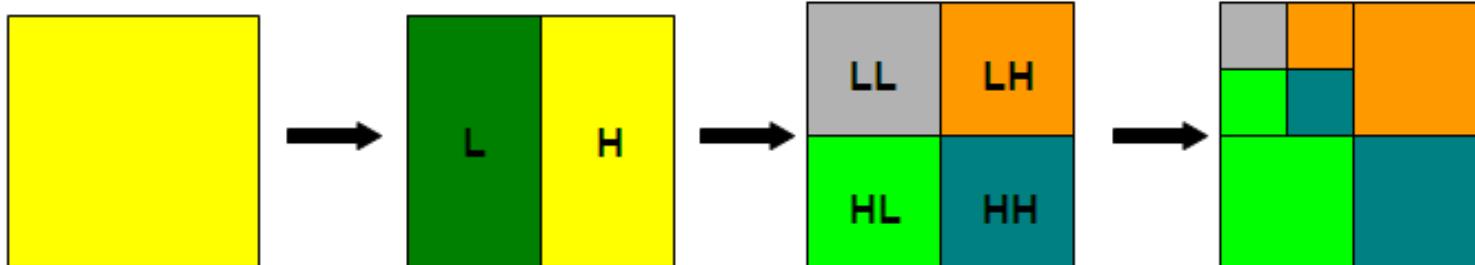
Wavelet Coding شرح

Remember that The human eyes are less sensitive to high frequency components.

اي ان العين المجردة لا تلاحظ الترددات العالية بل تلاحظ الترددات المنخفضة , تذكر من وحدة 4 عندما ذكرنا ان معظم المعلومات تتركز في الترددات المنخفضة (منتصف دائرة الصور بال frequency domain بينما تتوزع التفاصيل بالترددات البعيدة او العالية)

□ بطريقة wavelet نقوم بتقسيم الصورة الى مستويات من الترددات (وليس الى صور فرعية كما في DCT) , حيث بالمستوى الاول , نقوم بفصل الترددات العالية H للصورة عن الترددات المنخفضة L , ومن ثم بالمستوى الثاني نقسم جزء الترددات العالية H الى جزئين: ترددات عالية عليا HH, وترددات عالية دنيا LH, وكذلك الامر للترددات المنخفضة L, فهي تقسم لترددات منخفضة عليا LH وترددات منخفضة دنيا LL. وبعدها ننتقل الى المستوى الثالث, حيث نعيد تقسيم كل تردد مرة اخرى..ويمكن ان نقسم لمستوى رابع وخامس و سادس,, الخ حسب الاختيار (انظر شريحة 24).

□ ولنستفيد من compression في ال wavelet على ادنى الترددات الدنيا (اي معظم تركيز المعلومات و الطاقة للصورة), وهكذا نختصر حفظ كل الصورة و من ثم نحقق ضغط الصورة المطلوب مع وجود فرق بسيط Error بين الصورة الاصل و الصورة المضغوطة (بالتفاص



Wavelet selection

As in block transform selection, the appropriate wavelet type must be selected carefully.

The most widely used wavelet-based compression are:

- 1) Daubechies wavelets,
- 2) biorthogonal wavelets.
- 3) Extra..

Comparison between different wavelets

- (1) Harr wavelets : the simplest
- (2) Daubechies wavelets : the most popular imaging wavelets
- (3) Symlets : an extension of Daubechies with increased symmetry
- (4) Cohen-Daubechies-Feauveau wavelets : biorthogonal wavelets

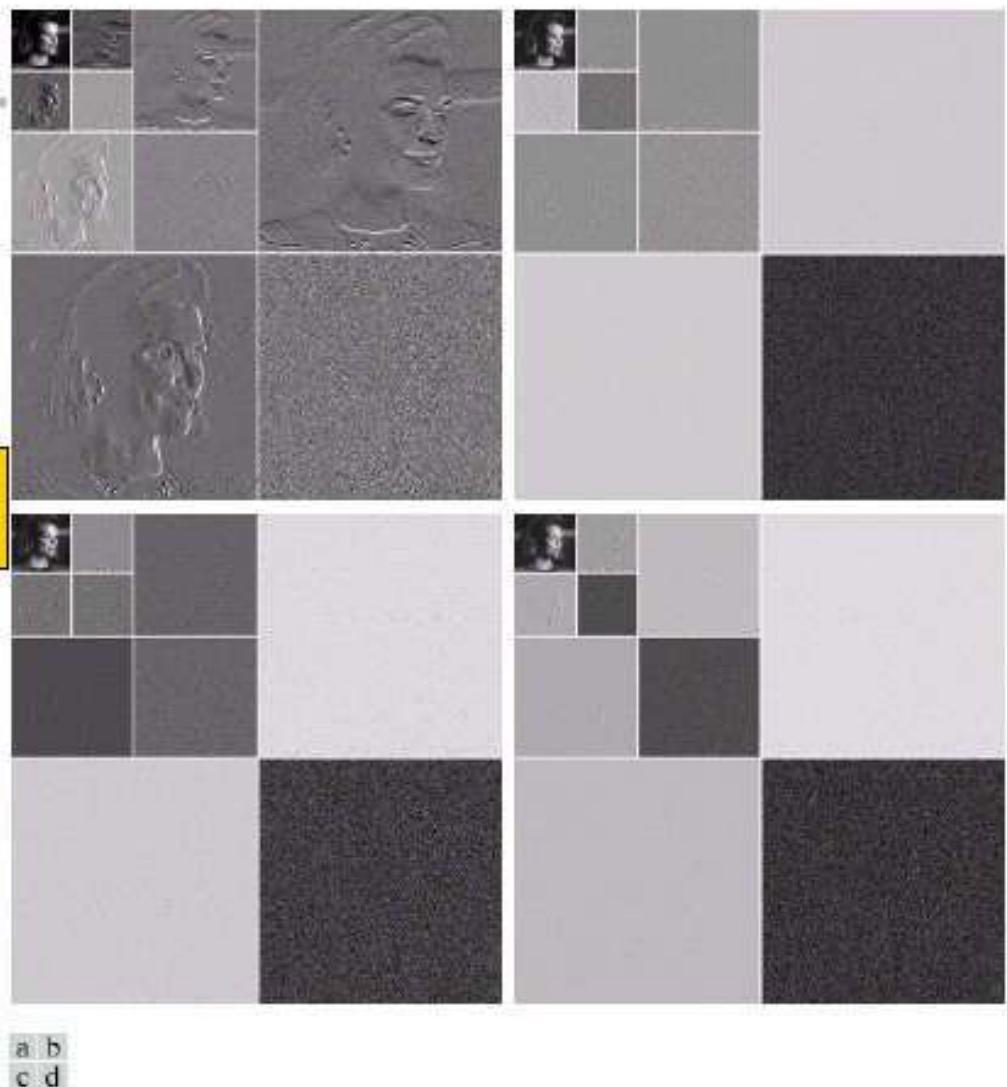


FIGURE 8.42 Wavelet transforms of Fig. 8.23 with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies-Feauveau biorthogonal wavelets.

Decomposition level selection

اختيار عدد مستويات التقسيم

Decomposition level selection is an important factor that affect wavelet computational complexity and reconstruction error .

P-scale fast wavelet transform involves P filter iterations. (i.e we call our wavelet example in the last slide a 3-scale fast transform since it includes 3 iterations).

The number of forward and inverse transforms increases with the number of decomposition levels.

Quantization of coefficients at more decomposition levels impacts larger areas of the reconstructed image

In applications like: searching image databases, or transmitting images, the **factors the determine the number of transform levels** are:

- 1) *The resolution of the stored images*
- 2) *The scale of the lowest useful approximations*

Decomposition level impact on wavelet coding the 512×512 image

Decomposition Level (Scales or Filter Bank Iterations)	Approximation Coefficient Image	Truncated Coefficients (%)	Reconstruction Error (rms)
1	256×256	74.7%	3.27
2	128×128	91.7%	4.23
3	64×64	95.1%	4.54
4	32×32	95.6%	4.61
5	16×16	95.5%	4.63

Three levels are enough!

Digital Image Processing

CHAPTER 9: MORPHOLOGICAL IMAGE PROCESSING

Morphological Image Processing

Morphology: a branch of biology that deals with the form and structure of animals and plants. “Morphing” in Biology which means “changing a shape”.

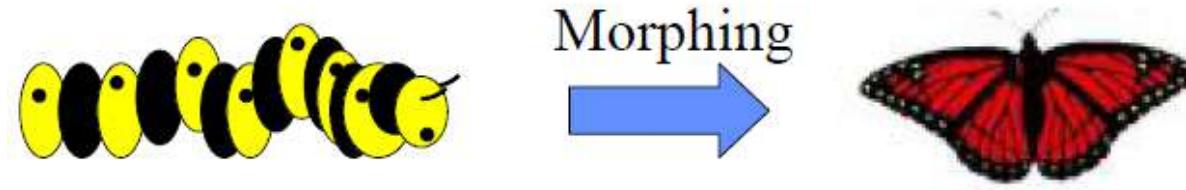
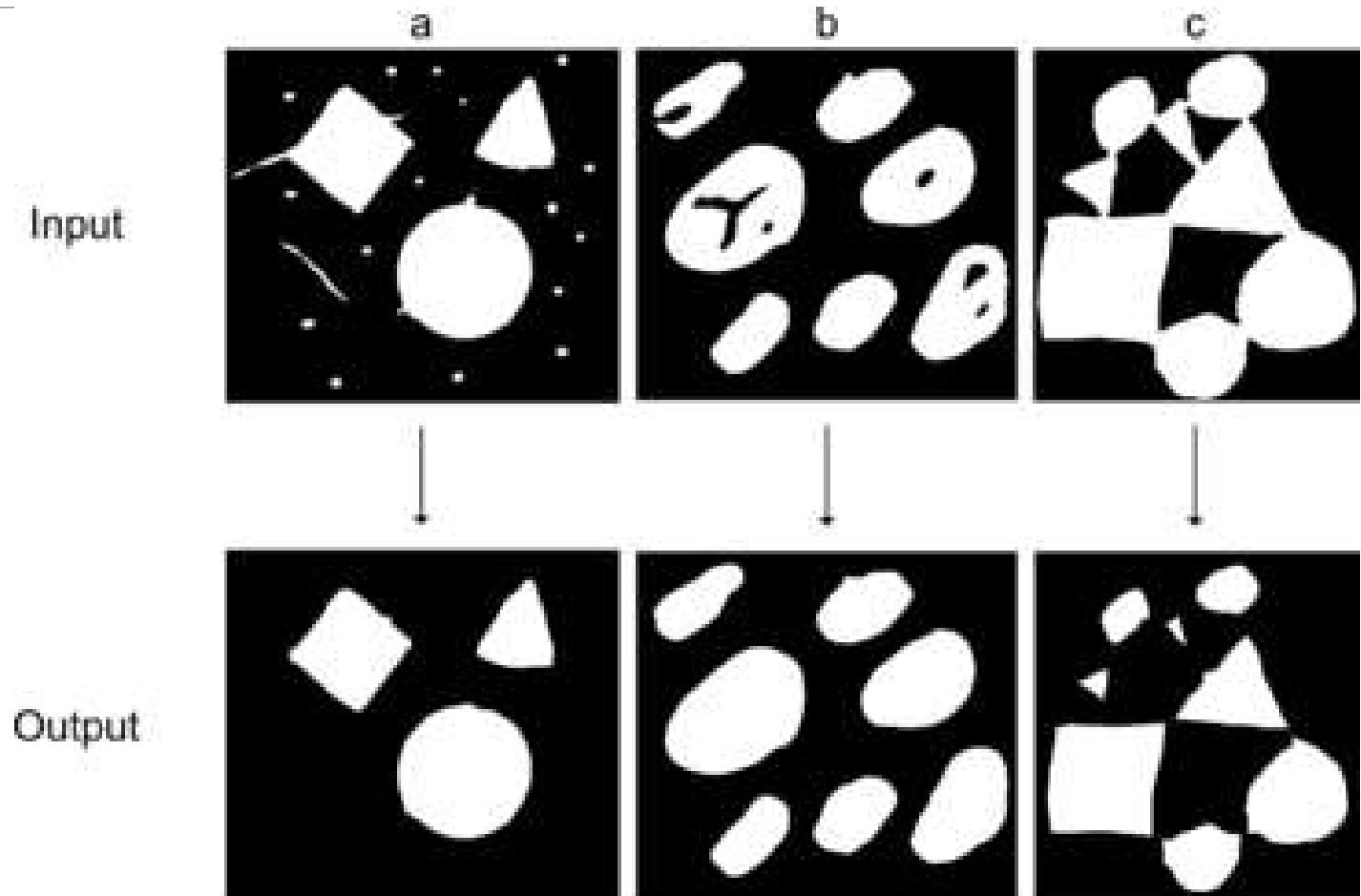


Image morphological operations are used to manipulate object shapes such as thinning, thickening, and filling.

Mathematical morphology: extraction of image components that are useful in the representation and description of regional shape, such as boundaries and skeletons.

Morphological Image Processing

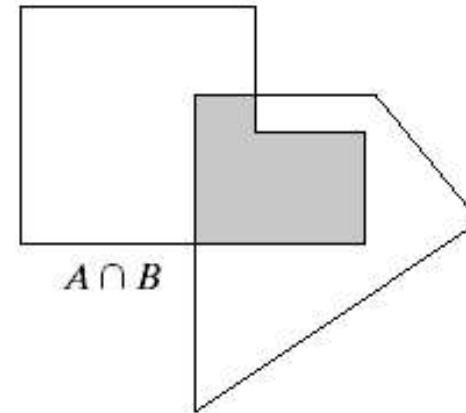
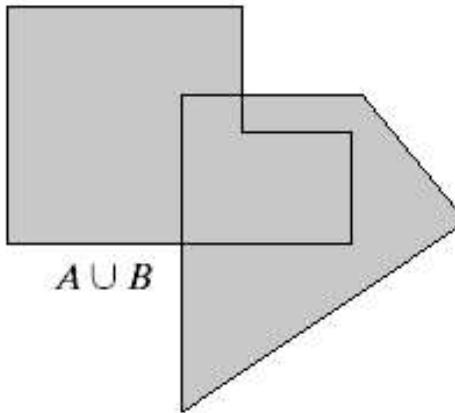
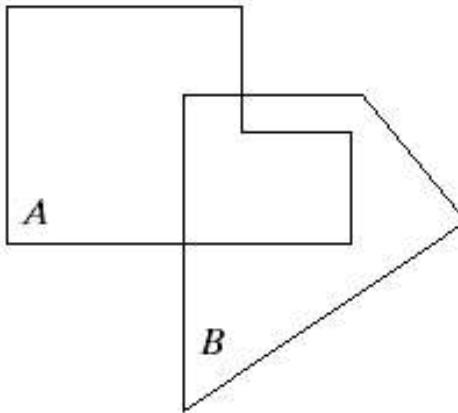
Morphological operations are typically applied to remove imperfections introduced during segmentation, and so typically operate on binary images.



Morphological Image Processing

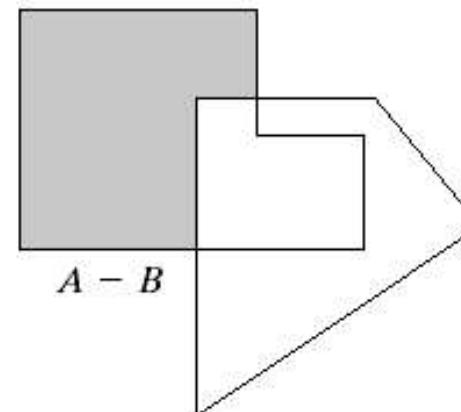
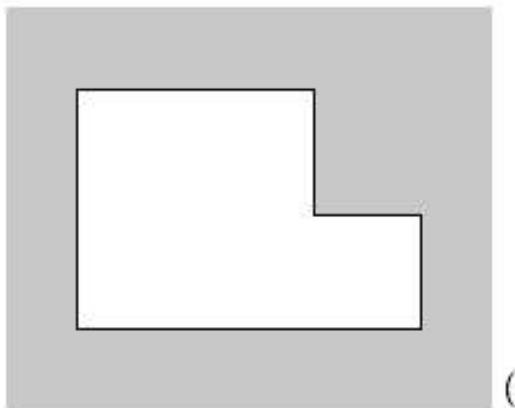
- The language of mathematical morphology is **set theory**.
- **Sets** in mathematical morphology represent objects in an image.
- In binary images, the sets are **members** of Z^2 consists of x,y coordinates.
- In gray scale images, the set represents components in Z^3 which means x,y coordinates and discrete intensity value.
- Until this chapter, the I/O of DIP methods were images.
- Now, we have a transition to DIP methods where I are images but O are attributes extracted from those images.

Basic Set Operations



a	b	c
d	e	

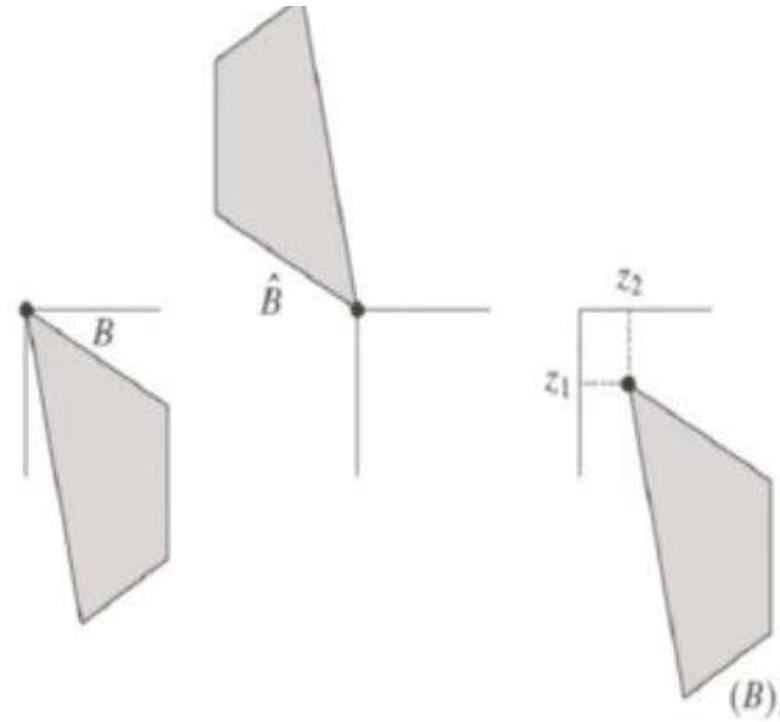
FIGURE 9.1
(a) Two sets A and B . (b) The union of A and B .
(c) The intersection of A and B . (d) The complement of A .
(e) The difference between A and B .



Translation and Reflection

- The concept of set **reflection** and **translation** are extensively used in mathematical morphology.
- In **reflection** the set of points in B whose (x, y) coordinates have been replaced by (-x, -y).
- In **translation** the set of points in B whose (x, y) coordinates have been replaced by (x+z₁, y+z₂).

$$\hat{B} = \left\{ \bar{w} \mid \bar{w} = -\bar{b}, \text{ for } \bar{b} \in B \right\}$$
$$(B)_z = \left\{ \bar{c} \mid \bar{c} = \bar{b} + \bar{z}, \text{ for } \bar{b} \in B \right\}$$



a b c

FIGURE 9.1
(a) A set, (b) its reflection, and
(c) its translation by z .

Structure Elements

- **Structure elements (SE): a shape mask used in basic morphological operations.**
- Structuring elements can be any size and make any shape digitally representable, each has an origin.
- Unlike spatial kernels, morphological kernels/ structuring elements do not require numerical values.

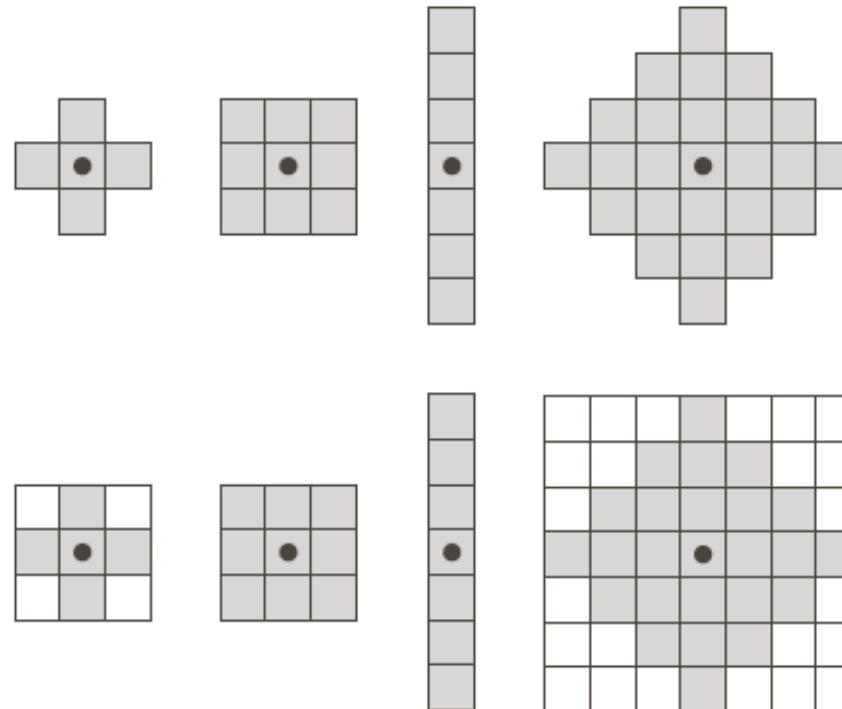


FIGURE 9.2 First row: Examples of structuring elements. Second row: Structuring elements converted to rectangular arrays. The dots denote the centers of the SEs.

Basic Morphological Operations

- Fundamentally **morphological image processing** is very 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**.
- **Dilation** and **erosion**: Operations fundamental to morphological processing.
- **Dilation** expands objects in the image, and **erosion** shrinks it.
- Many morphological algorithms in this chapter are based on these 2 primitives.

Dilation

- **Dilation of A and B is defined as follows**

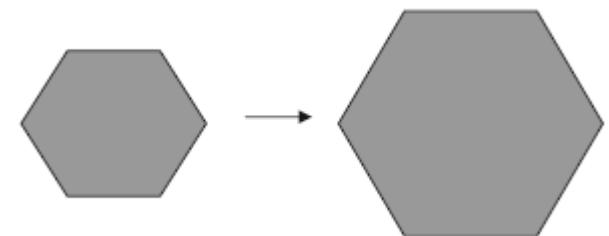
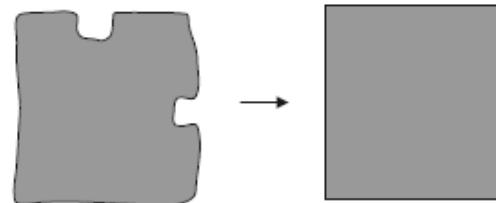
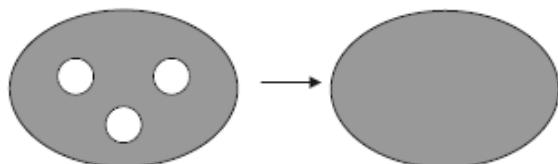
$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

- A : Input Image , B: Structure Element
- Obtain the reflection of B about its origin, and shift this reflection by z. if it overlaps with A, output is 1.
- In other words:
 - Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

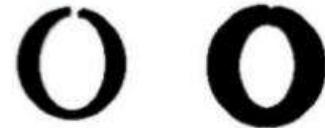
0	0 0	0 1	0 0	0	0
0	0 1	0 1	0 1	0	0
0	0 0	1 1	1 0	0	0
0	0	1	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Dilation

- Dilation allows objects to expand, then potentially filling in small holes and connecting disjoint object.
- Dilation expands the connected sets of 1s of a binary image.
- It can be used for
 1. Expanding shapes:
 2. Filling holes, gaps and gulfs:



Dilation can repair breaks



Dilation can repair intrusions

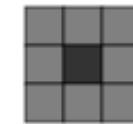
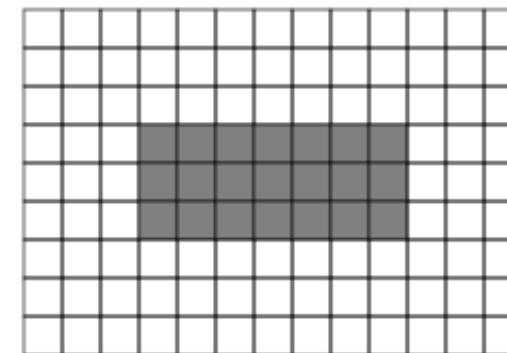


Watch out: Dilation enlarges objects

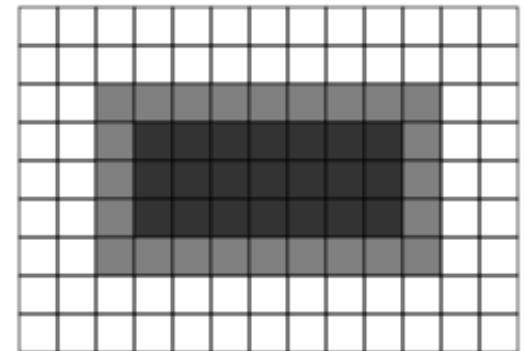
Example on Dilation

Dilation expands the connected sets of 1s of a binary image.

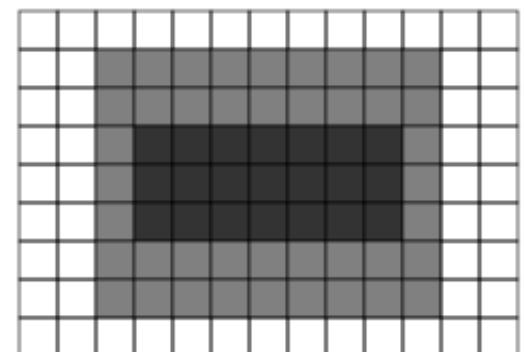
The shaded pixels are considered 1s in a binary image which refer to the object.



H, 3x3, origin
at the center



H, 5x3, origin
at the center



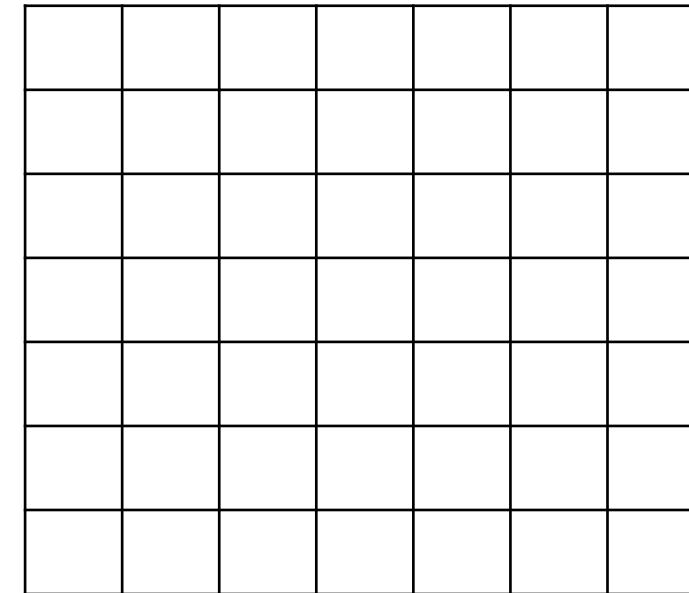
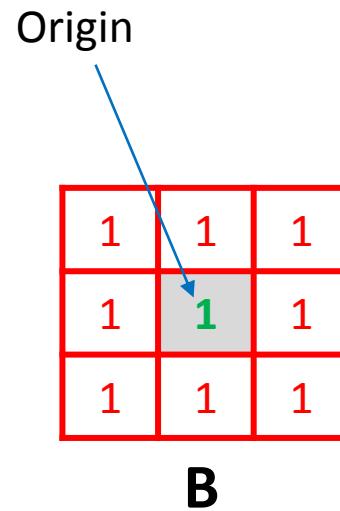
Dilation enlarges a set.

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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



Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0 (like logical OR operation)

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the
output is 0

0	0	0	0	0		

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the output is 1

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the output is 1

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the output is 1

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the output is 1

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
A							

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the
output is 1

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the output is 0

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the output is 1

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the output is 1

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the
output is 1

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the
output is 1

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

There is any
1-pixel in B
coincide with
1-pixel in A ?

Yes, then the
output is 1

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the
output is 0

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

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	1
A	1	1	1	1	1	1	1

There is any
1-pixel in B
coincide with
1-pixel in A ?

No, then the
output is 0

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

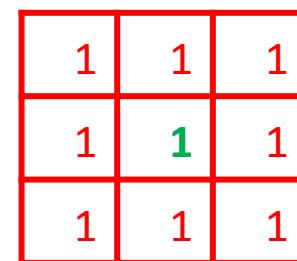
Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A



B

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

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	1	0
1	1	1
0	1	0

B

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

$A \oplus B$

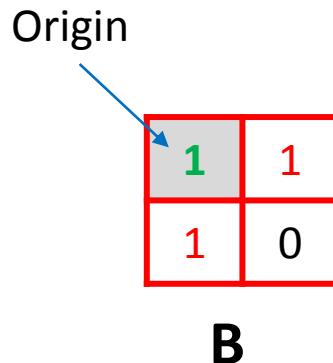
Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A



$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0					

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0						

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1			

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1	0				

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1	0	0	

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1	0	0	0

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1	0	0	0
0					

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1	0	0	0
0	1				

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1	0	0	0
0	1	1			

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	1	0	0	0
0	1	1	0		

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

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

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

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

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

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

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

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

$A \oplus B$

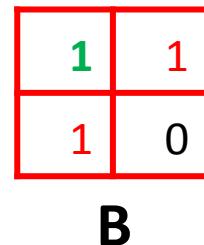
Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A



B

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

$A \oplus B$

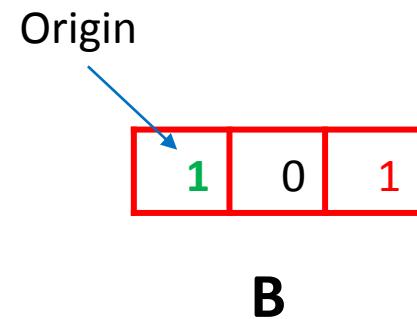
Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A



$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	0	0	0
1				

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	0			

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	0	1		

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	0	1	0	

$A \oplus B$

Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	0	1	0	0

$A \oplus B$

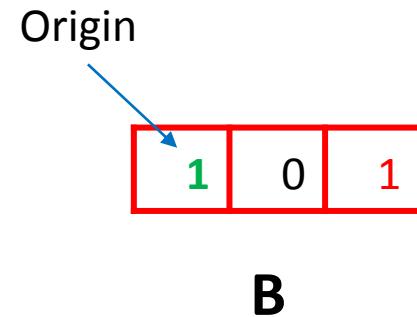
Example on Dilation Technique

Dilation Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary OR operation of the corresponding elements in A.

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

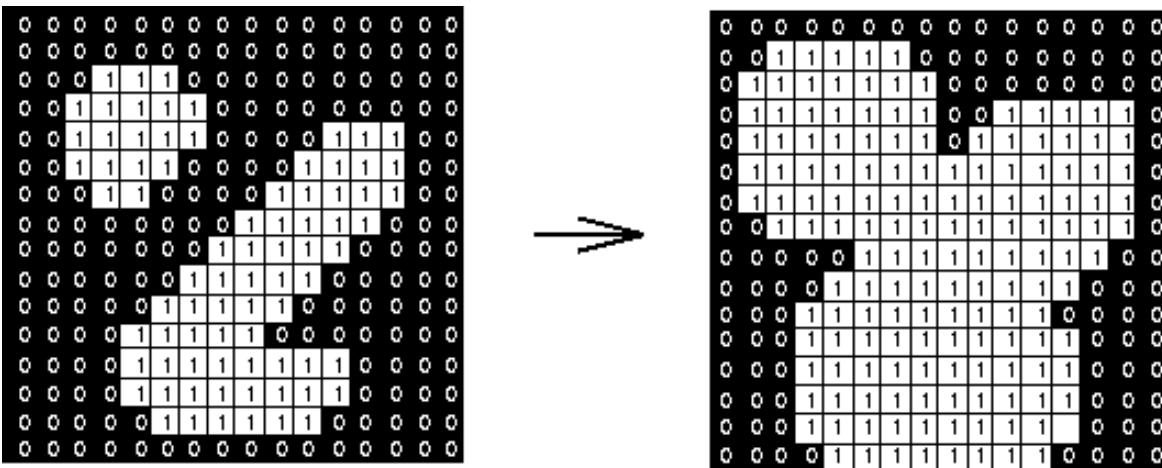
A



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

$A \oplus B$

Example on Dilation

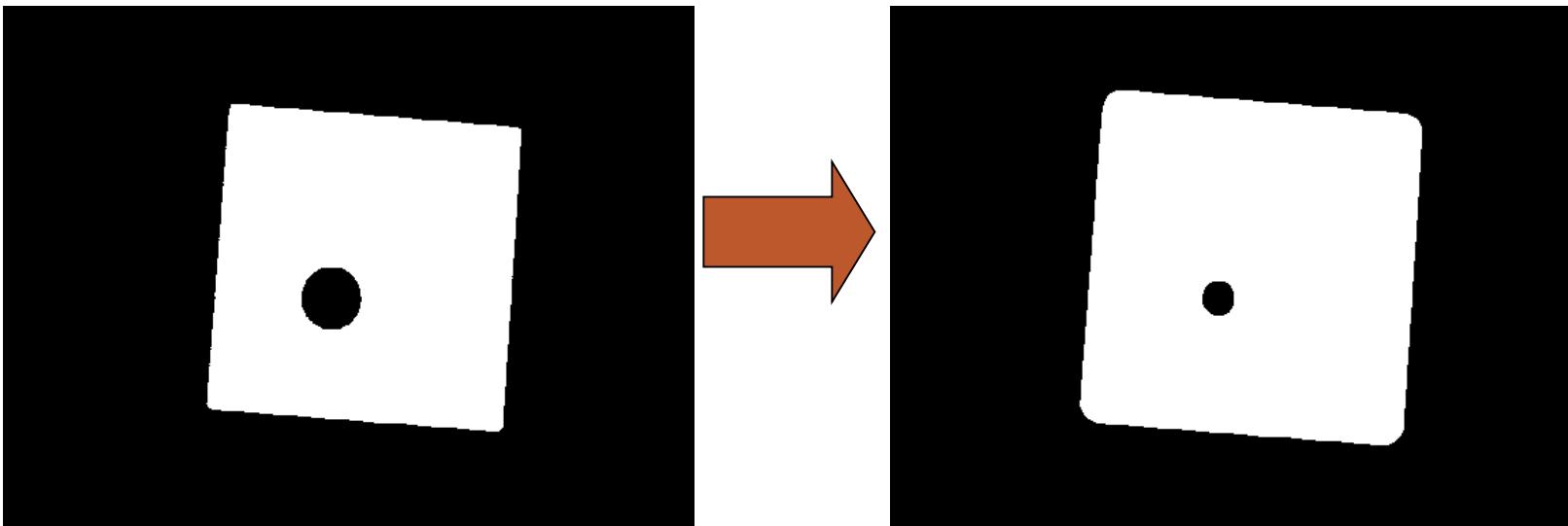


**Dilation enlarges foreground,
shrinks background**

1	1	1
1	1	1
1	1	1

Set of coordinate points =
{ (-1, -1), (0, -1), (1, -1),
(-1, 0), (0, 0), (1, 0),
(-1, 1), (0, 1), (1, 1) }

Example on Dilation

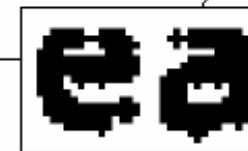


Application of Dilation: Bridging Gaps

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



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



a
b
c

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

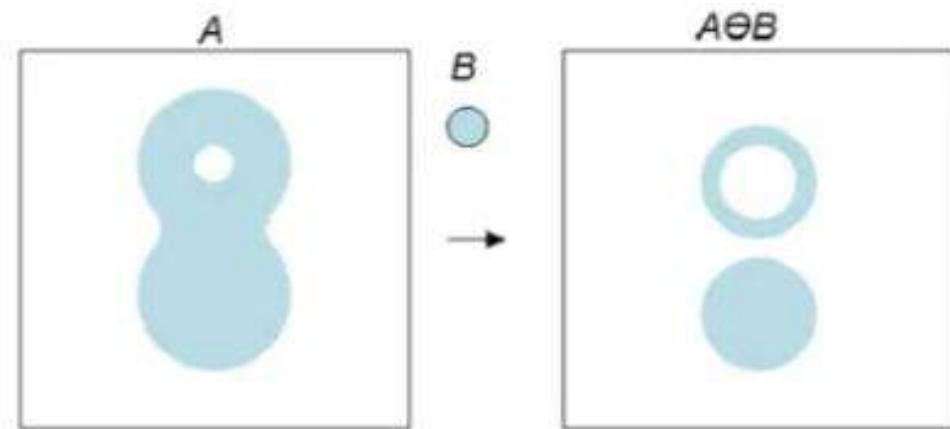
0	1	0
1	1	1
0	1	0

Erosion

- **Erosion of A by B.**

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

- Shift B by z, if it is completely inside A, output a 1.
- **Erosion enlarges holes, breaks thin parts, shrinks objects.**



Erosion

- ❑ **Erosion** shrinks objects by etching away (eroding) their boundaries. Erosion shrinks the connected sets of 1s of a binary image.

- ❑ It can be used for

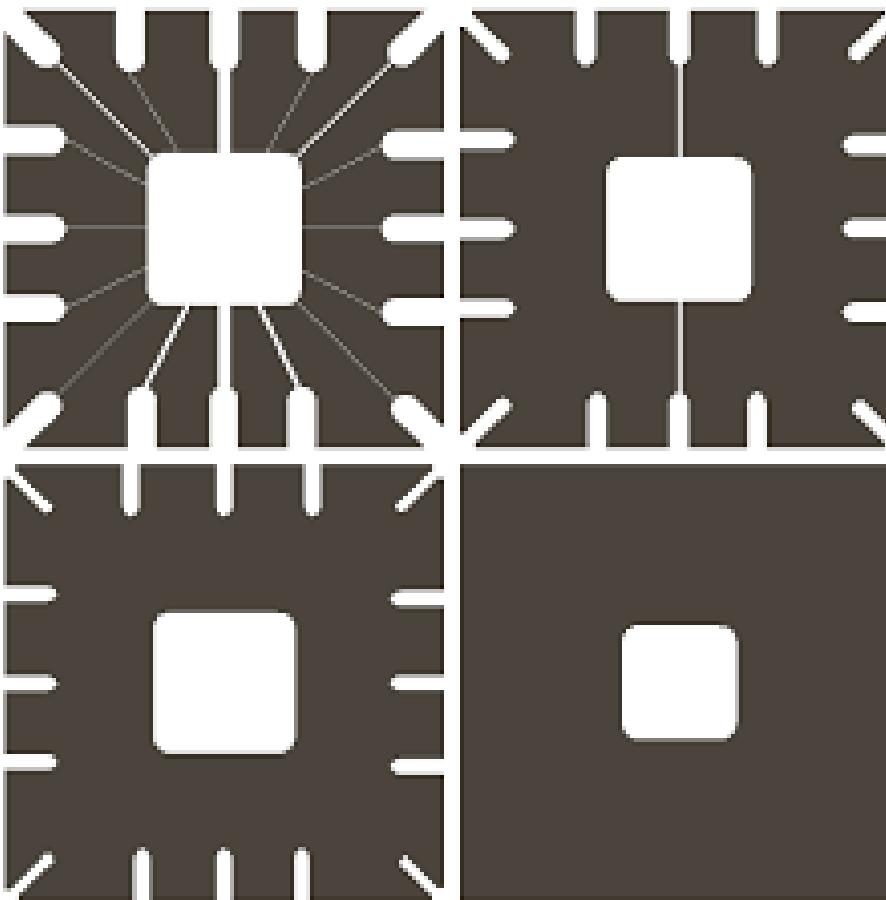
1. Shrinking shapes:



2. Removing bridges, branches and small protrusions:



Erosion



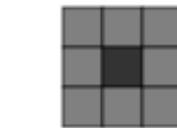
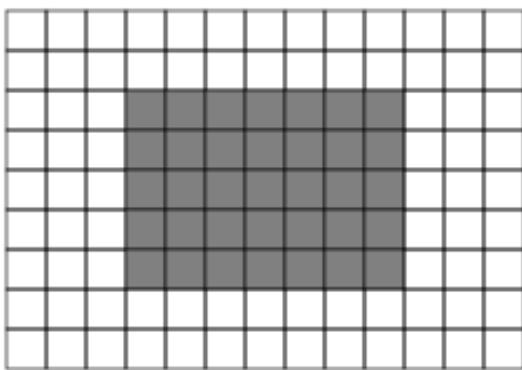
a b
c d

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

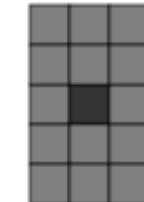
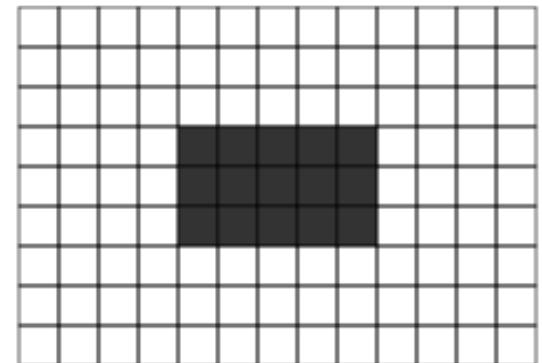
Example on Erosion

Erosion shrinks the connected sets of 1s of a binary image.

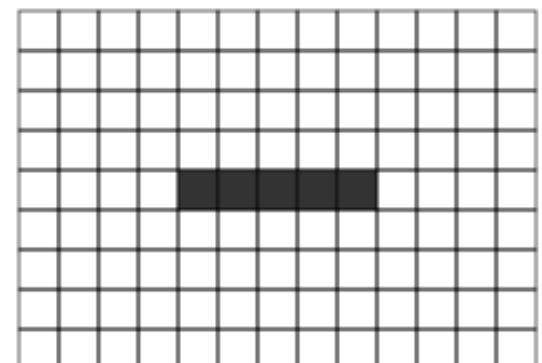
The object is deleted completely if the SE is larger than the entire object



H, 3x3, origin
at the center



H, 5x3, origin
at the center



Erosion shrinks a set

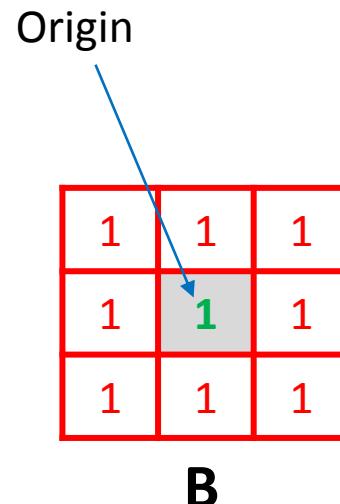
Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A



Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the output is 0 (like logical AND operation)

A

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

0	0						

When applying
Erosion on 0-
pixels in A it will
stay 0 in the
eroded image.

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the output is 0

A

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the output is 0

A

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the output is 0

A

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

0	0	0	0	0	0	0
0	0	0	0	0	0	

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

0	0	0	0	0	0	0
0	0	0	0	0	0	0

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

0	0	0	0	0	0	0
0	0	0	0	0	0	0

A

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
 - By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the output is 0

A

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

Yes, then the
output is 1

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

Are all 1-pixels
in B coincide
with 1-pixels
in A ?

No, then the
output is 0

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

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

1	1	1
1	1	1
1	1	1

B

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

$A \ominus B$

Example on Erosion Technique

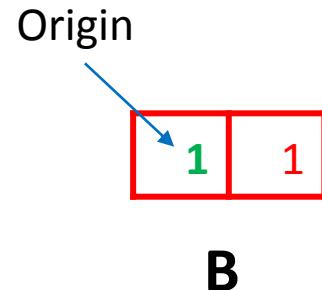
Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

Apply Erosion with padding Zero

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

A



$A \ominus B$

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

0	0	0	0	0
1				

$A \ominus B$

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	1			

$A \ominus B$

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	1	1		

$A \ominus B$

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	1	1	1	

$A \ominus B$

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

0	0	0	0	0
1	1	1	1	0

$A \ominus B$

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A

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

$A \ominus B$

Example on Erosion Technique

Erosion Technique:

- Move B over A, placing origin at each pixel.
- By considering 1-pixel location in B, compute the binary AND operation of the corresponding elements in A.

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

A



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

$A \ominus B$

An Application of Erosion

In general, **dilation** does not fully restore eroded objects.



a b c

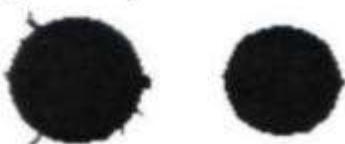
FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

Application of Erosion

Erosion can split apart joined objects



Erosion can strip away extrusions



Watch out: Erosion shrinks objects



Original image



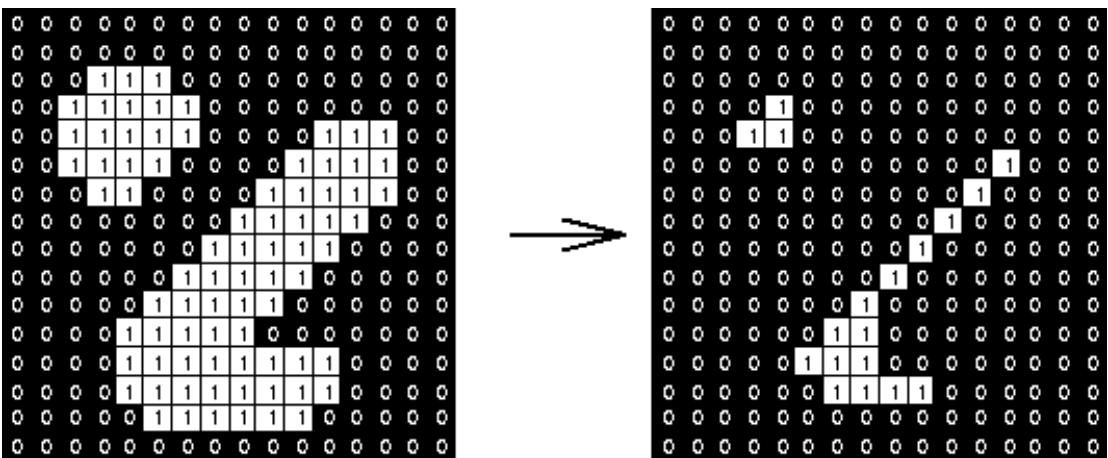
Erosion by 3*3
square structuring
element



Erosion by 5*5 square
structuring element

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

Example on Erosion

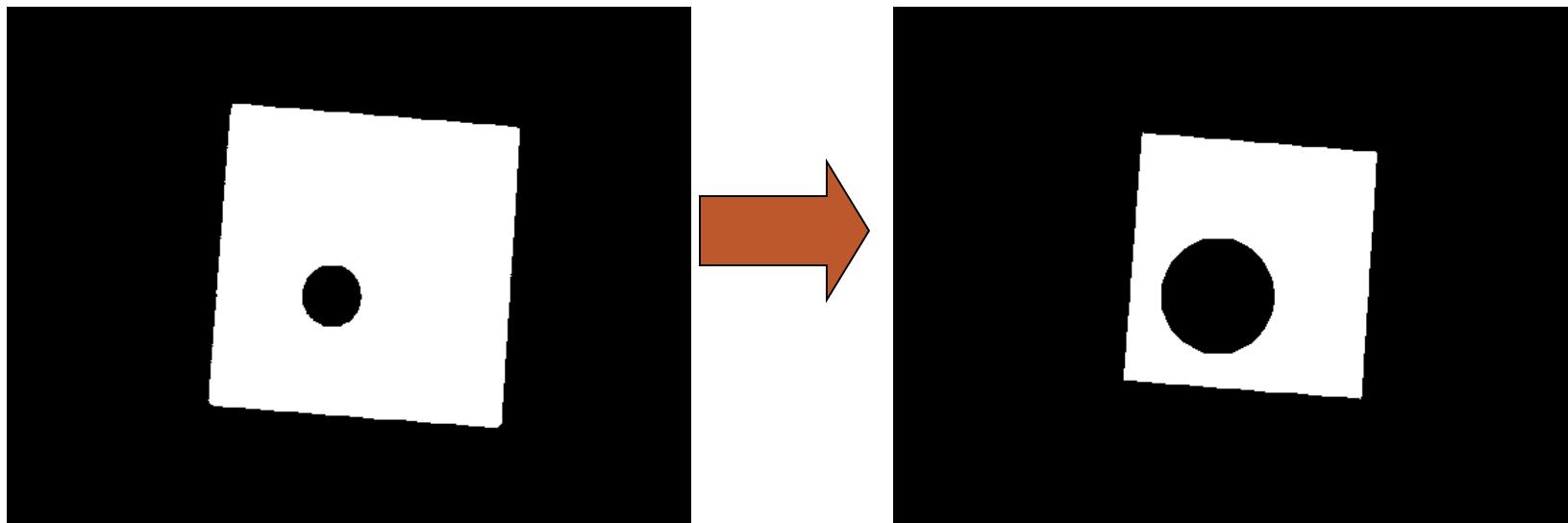


**Erosion shrinks foreground,
enlarges Background**

1	1	1
1	1	1
1	1	1

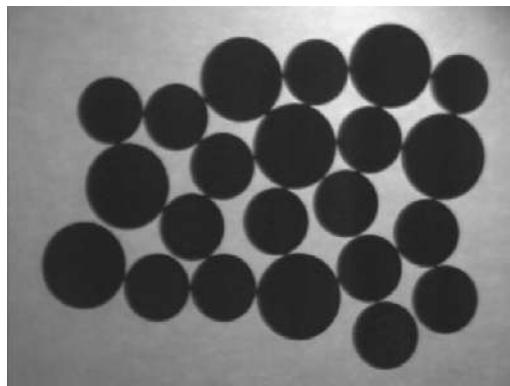
Set of coordinate points =
 $\{ (-1, -1), (0, -1), (1, -1),$
 $(-1, 0), (0, 0), (1, 0),$
 $(-1, 1), (0, 1), (1, 1) \}$

Example on Erosion

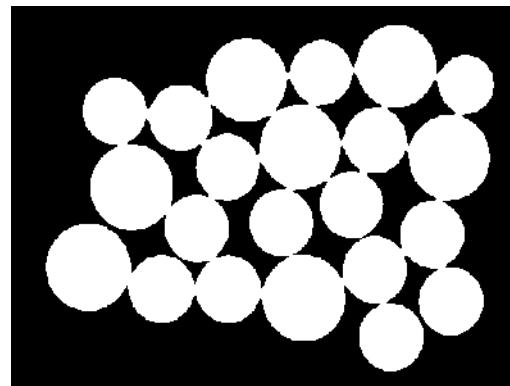


Application on Erosion: Counting Coins

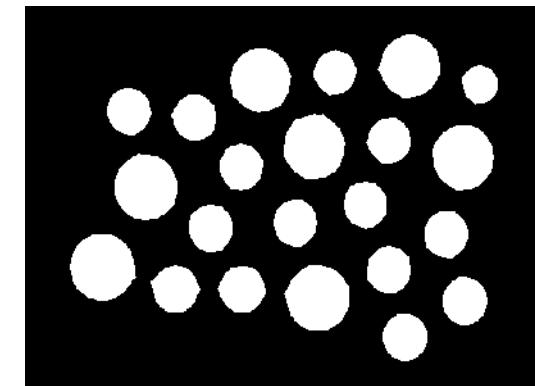
- ❑ Counting coins is difficult because they touch each other!
- ❑ Solution: Thresholding and Erosion separates them!



Original



Thresholding



Erosion

Morphological Operations

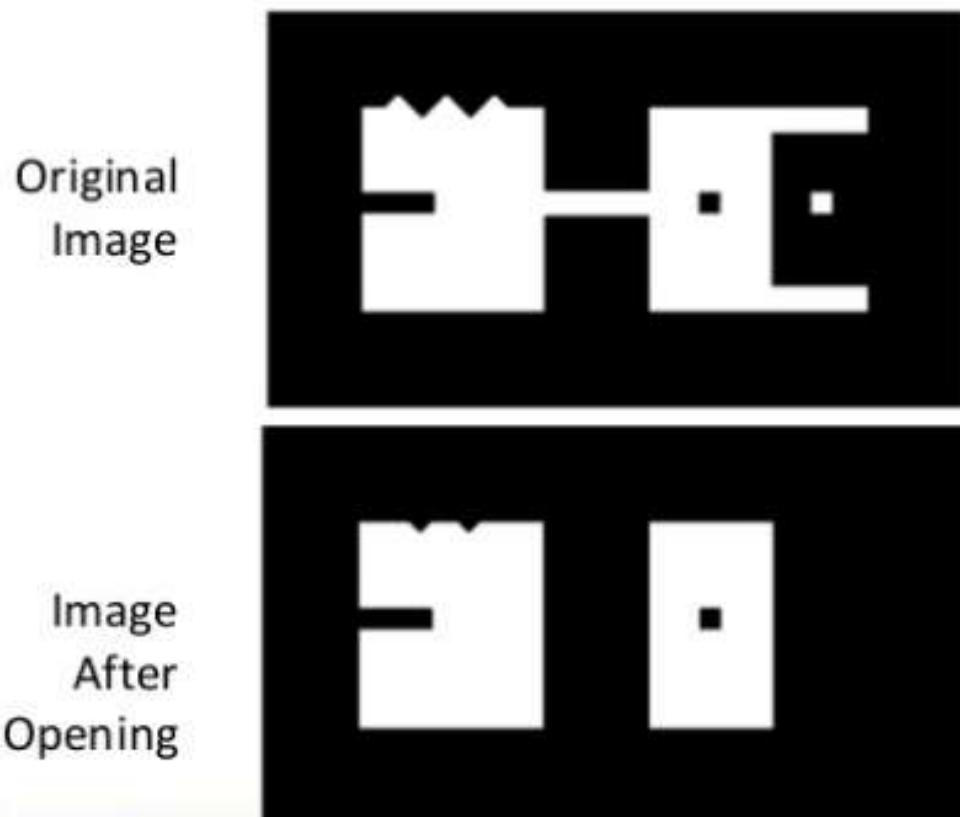
OPENING AND CLOSING

OPENING

- **Opening A by B:**

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

- Erosion of A by B, followed by a dilation of the result by B.
- It can be used to eliminate all pixels in regions that are too small to contain the structuring element.
 - Generally smooths the contour of an object
 - Breaks narrow isthmuses
 - Eliminates thin protrusions



CLOSING

- Closing of A by B:

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

- Dilation of A by B, followed by an erosion of the result by B.
- It can be used to connects objects that are close to each other.
 - Also tends to smooth sections of contours.
 - Generally fuses narrow breaks and long thin gulfs
 - Eliminates small holes
 - Fills gaps in the contour

Original
Image

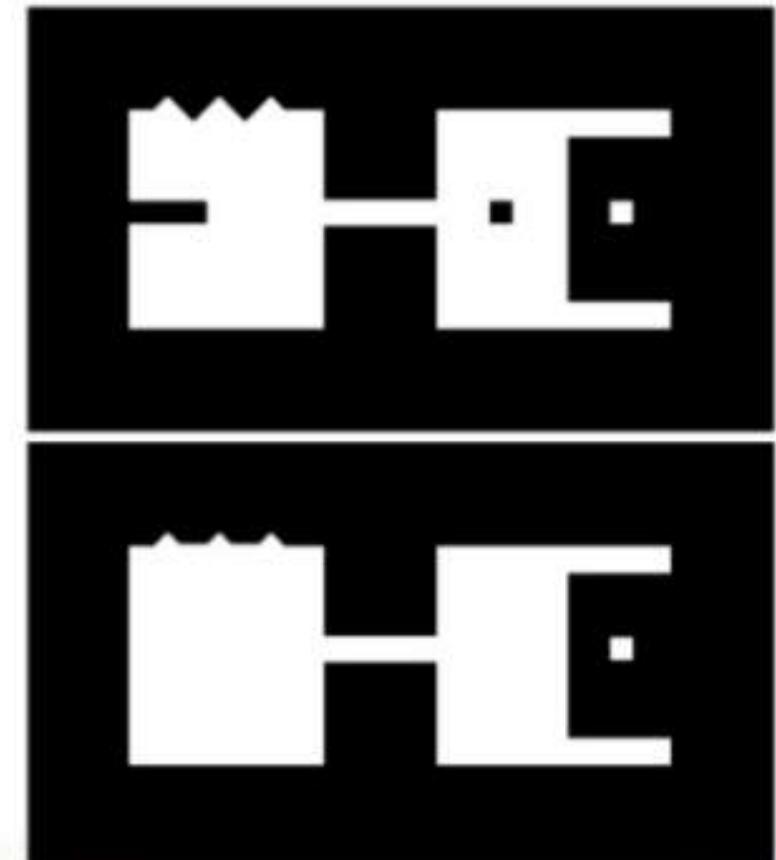
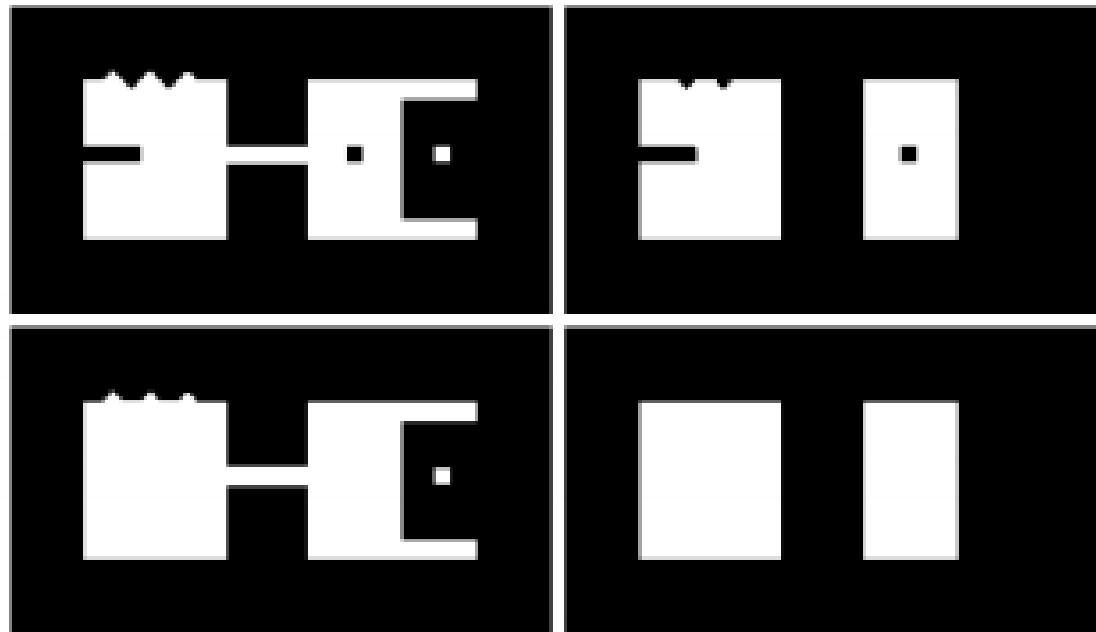


Image
After
Closing

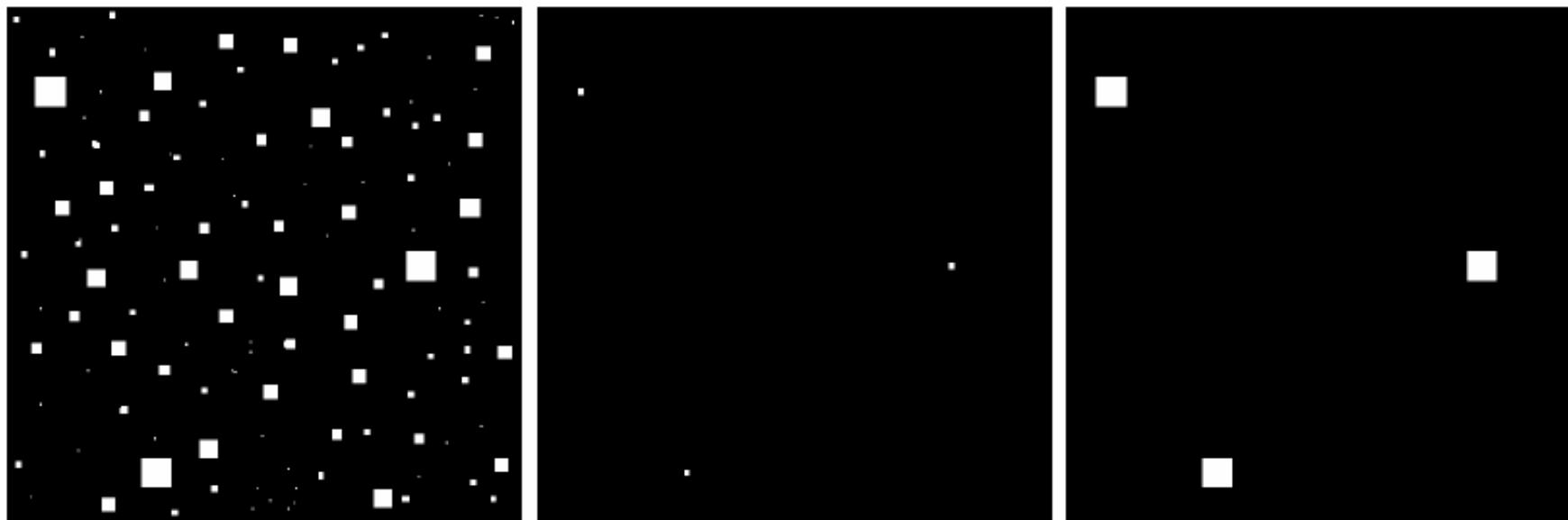
Opening and Closing



(a) Original. (b) Opening. (c) Closing. (d) Closing of (b).
The structuring element is a square.

An Application of Opening

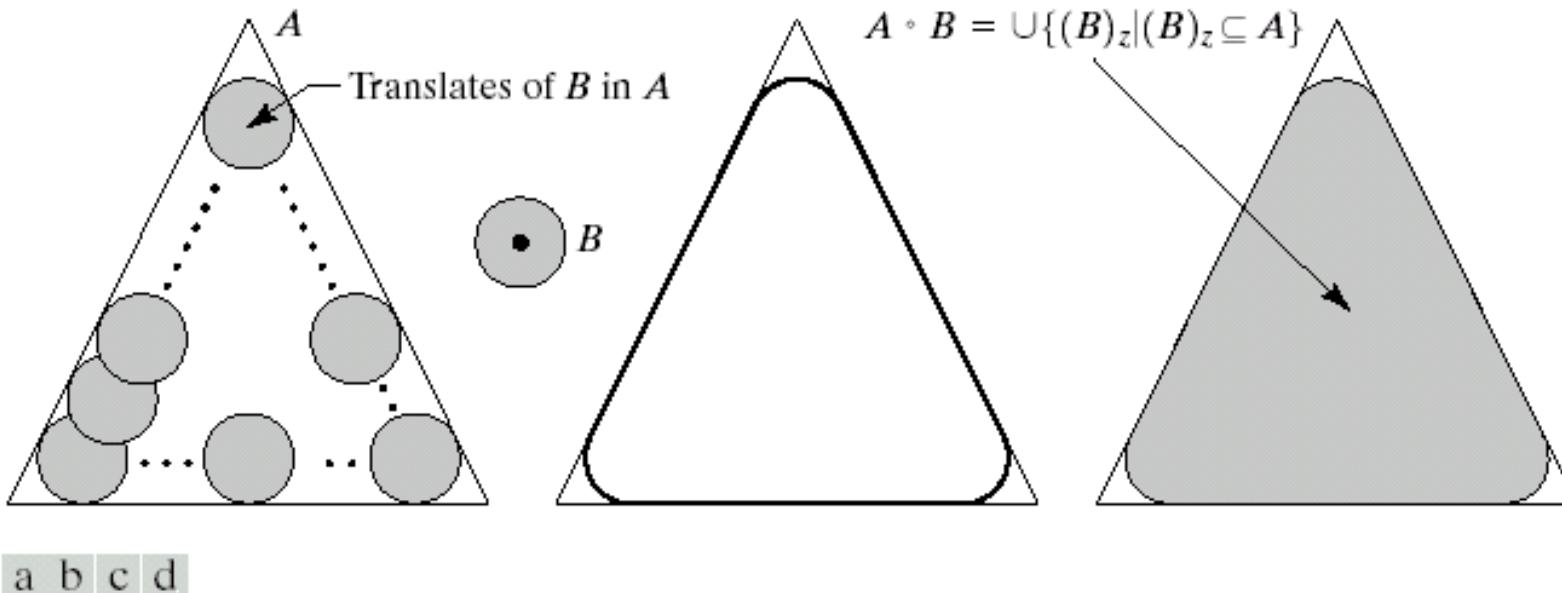
You can delete unwanted objects by using **Opening**.



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.

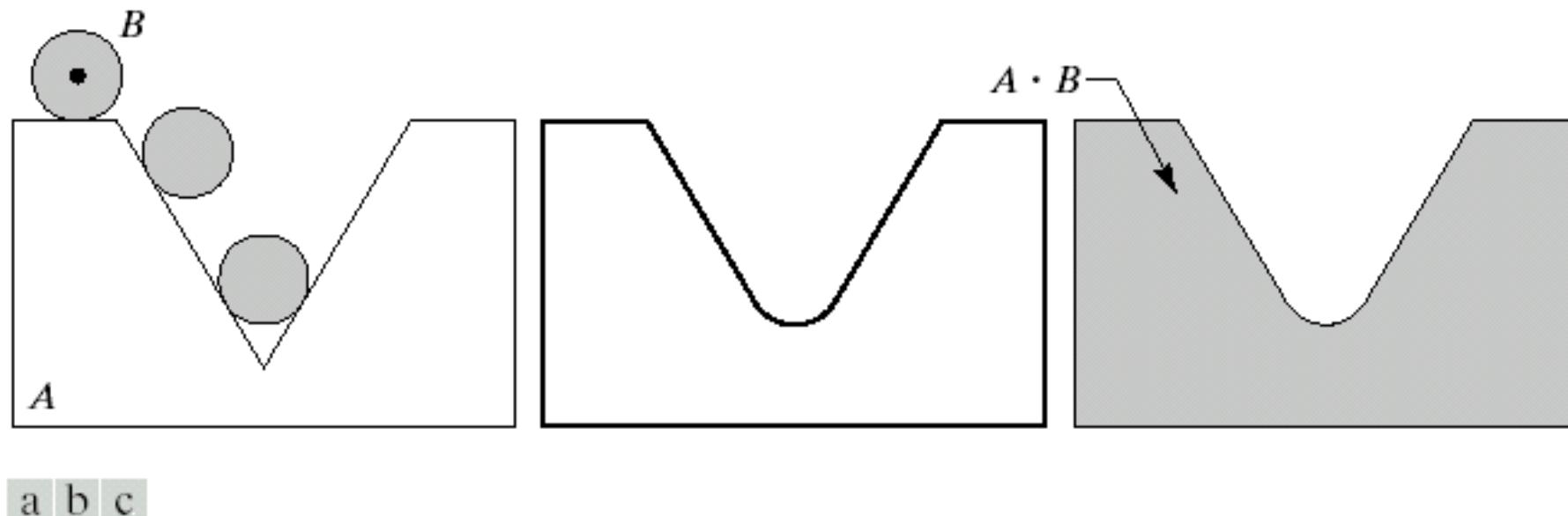
Opening



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). (c) The heavy line is the outer boundary of the opening. (d) Complete opening (shaded).

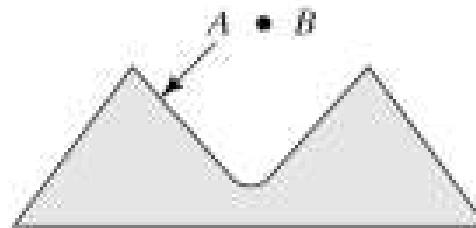
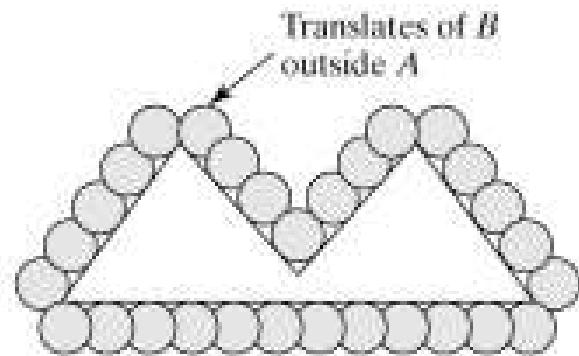
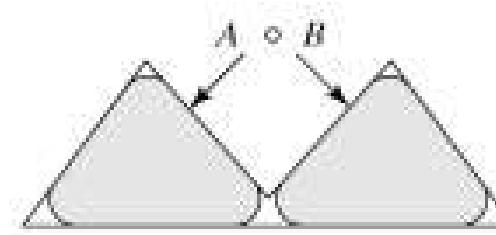
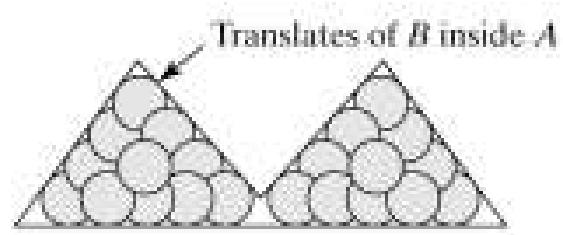
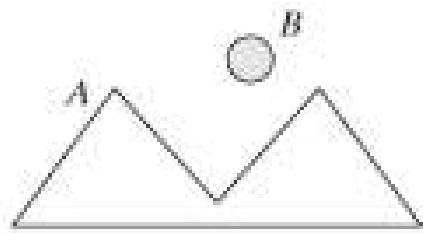
Closing



a b c

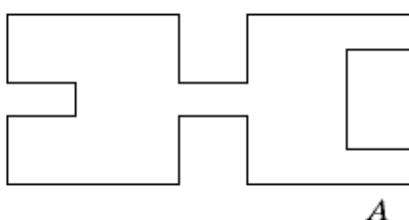
FIGURE 9.9 (a) Structuring element B “rolling” on the outer boundary of set A . (b) Heavy line is the outer boundary of the closing. (c) Complete closing (shaded).

Opening and Closing

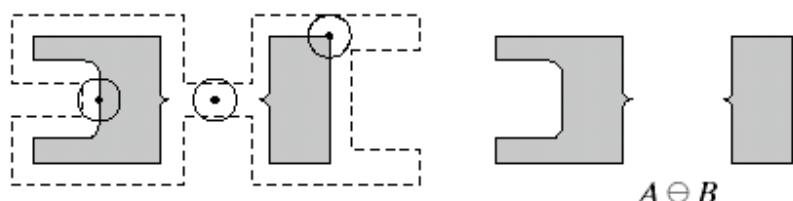


a
b c
d e
f g
h i

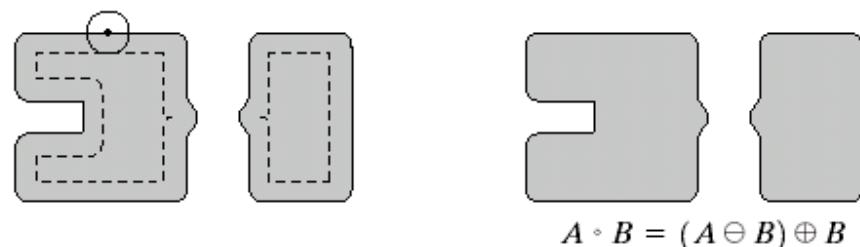
FIGURE 9.10
Morphological opening and closing. The structuring element is the small circle shown in various positions in (b). The dark dot is the center of the structuring element.



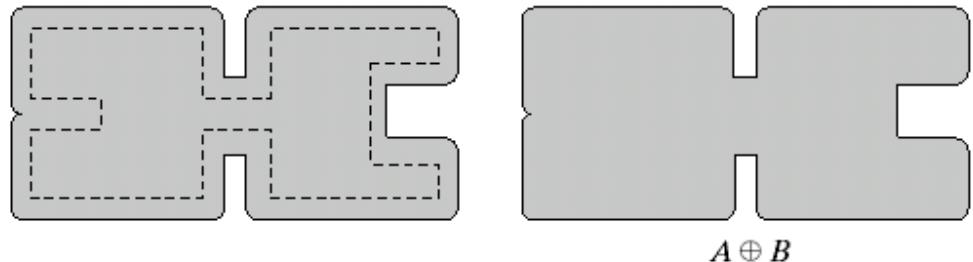
Opening and Closing



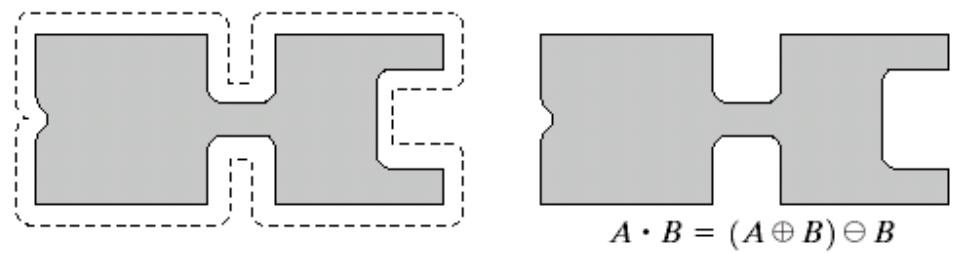
$$A \ominus B$$



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



$$A \ominus B$$



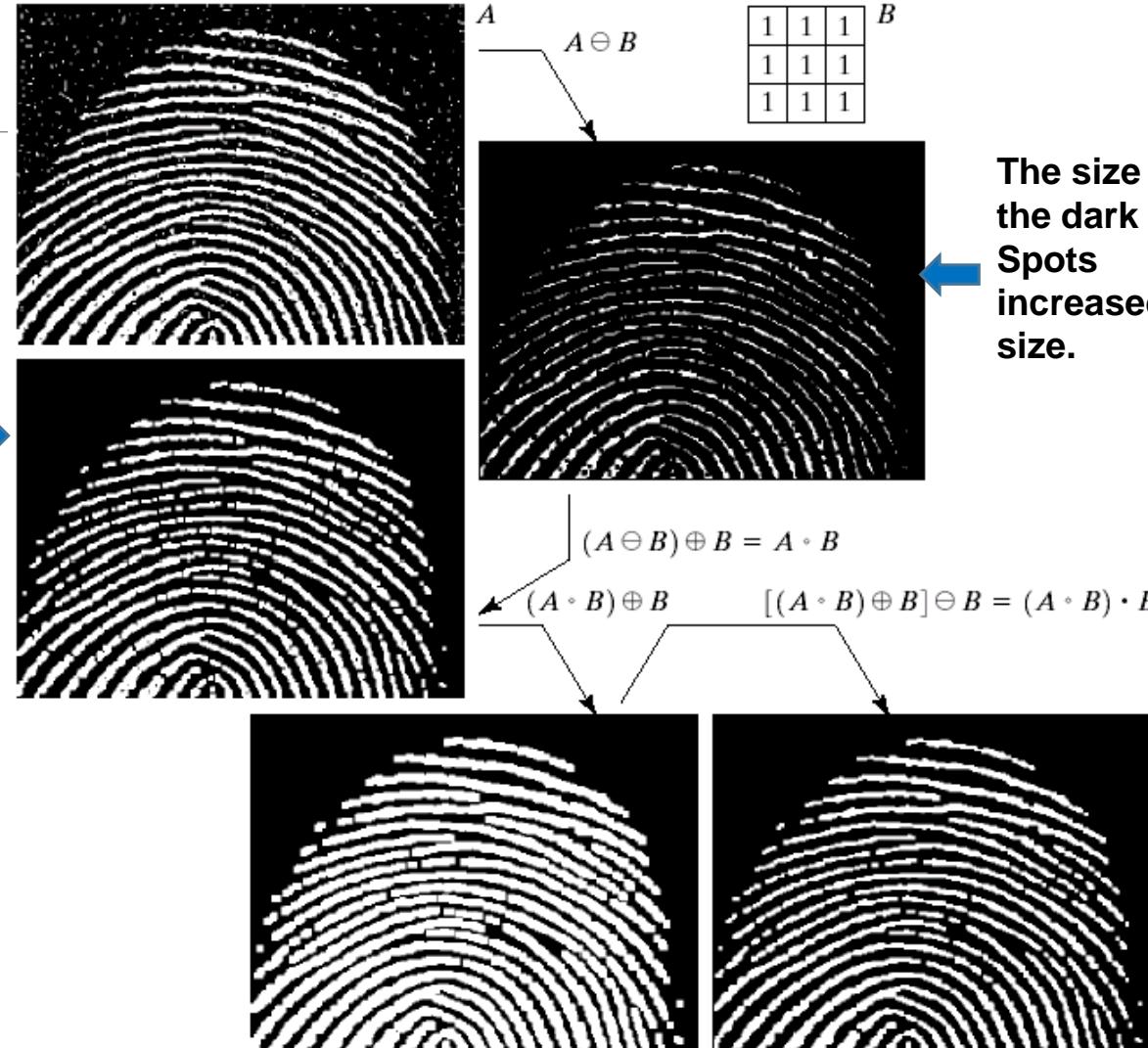
$$A \circ B = (A \ominus B) \oplus B$$

Morphological Filter: Opening followed by Closing

Morphological operations can be used to construct filters.

The noise manifests itself as light elements on a dark background and as dark elements on the light components of the fingerprint.

The opening operation reduced the noise components in size or deleted them completely. However, it created new gaps between the fingerprint ridges.



The size of the dark Spots increased in size.

Most of the gaps were restored but the ridges were thickened.

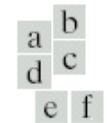


FIGURE 9.11

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

The noise has been eliminated With minimal distortion to the fingerprint.

Example on Opening

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

A

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

$A \ominus B$

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

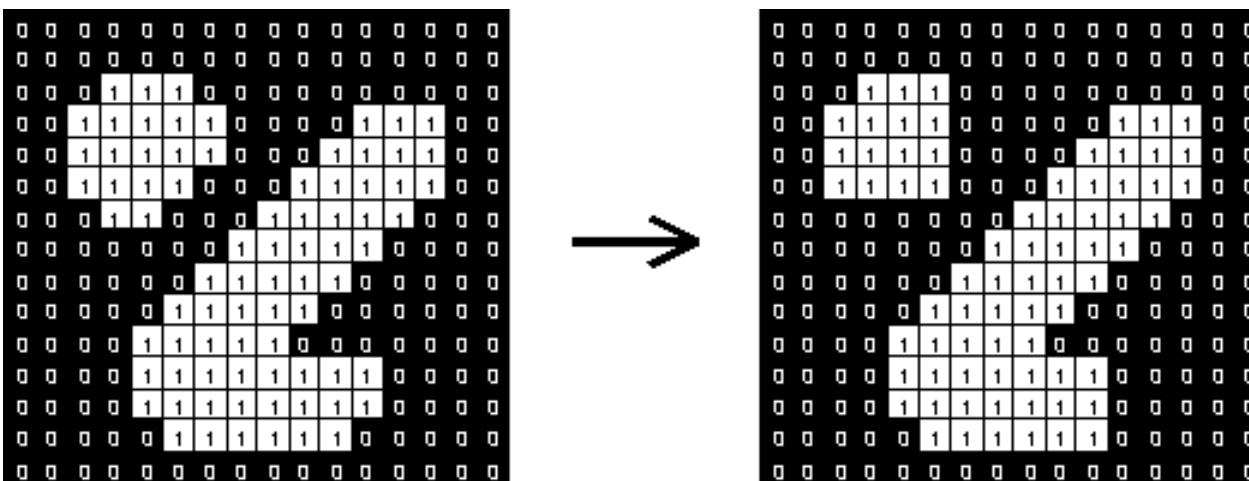
$(A \ominus B) \oplus B$

0	1	0
1	1	1
0	1	0

B

Example on Opening

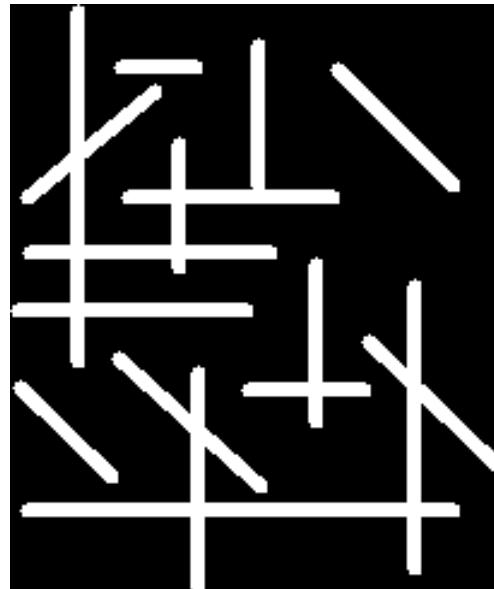
Structuring element: 3x3 square



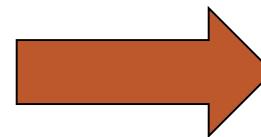
Application on Opening

Extract the horizontal and vertical lines separately

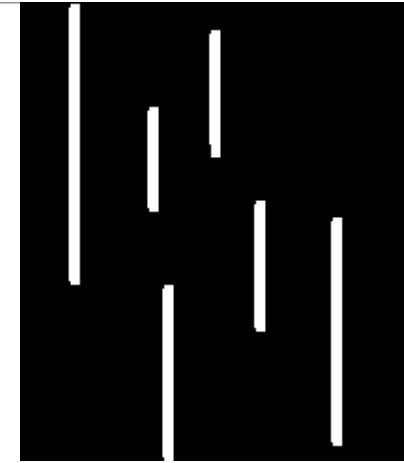
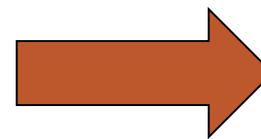
- ❑ There are a few glitches in rightmost image where the diagonal lines cross vertical lines.
- ❑ These could easily be eliminated, however, using a slightly longer structuring element.



3×9

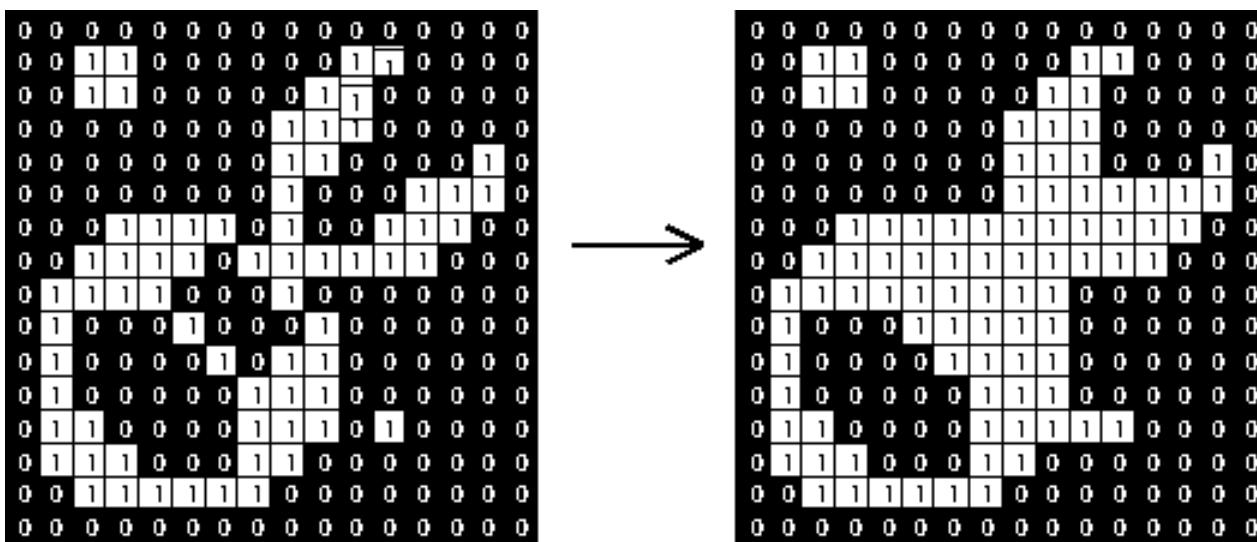


9×3



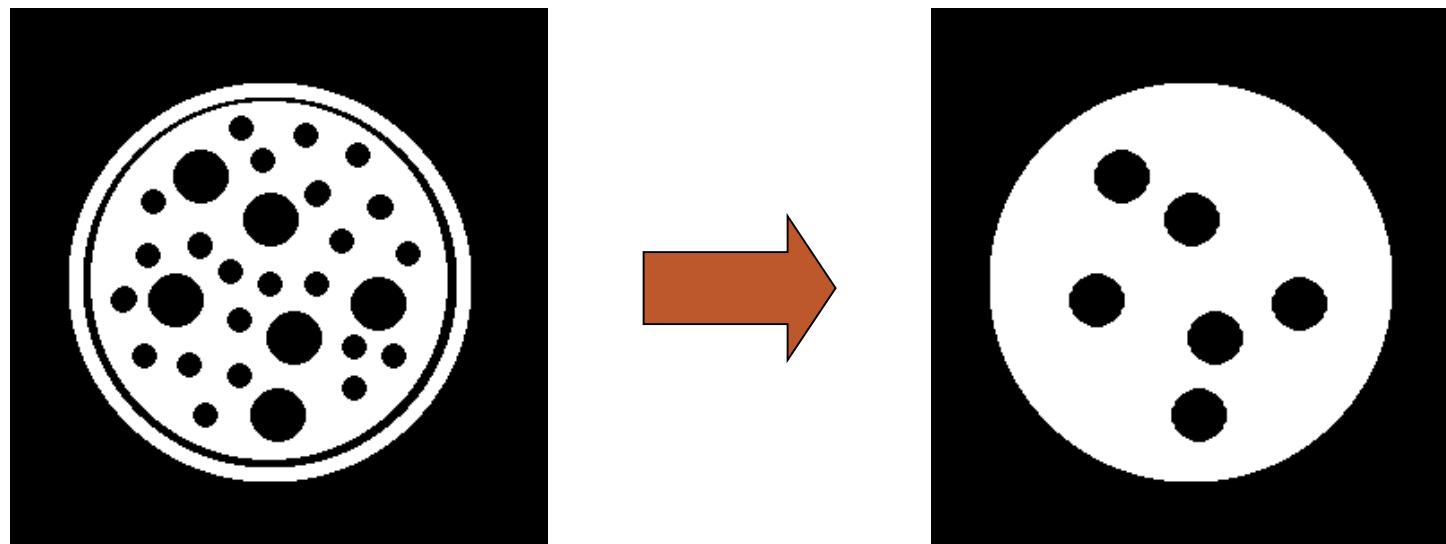
Example on Closing

Structuring element: 3x3 square



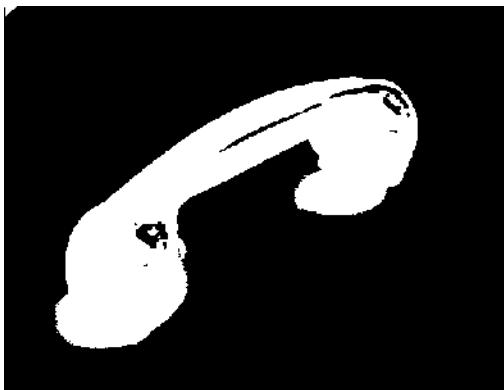
Application on Closing

- ❑ Closing operation with a 22 pixel disc
- ❑ Closes small holes in the foreground



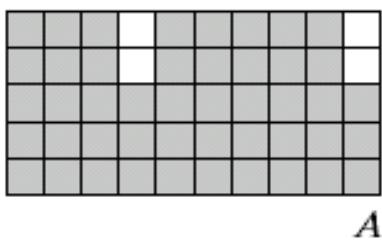
Application on Closing

- 1. Threshold**
- 2. Closing with disc of size 20**

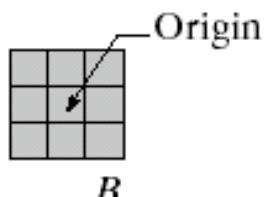


Boundary Extraction

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



$$A \ominus B$$



$$\beta(A)$$



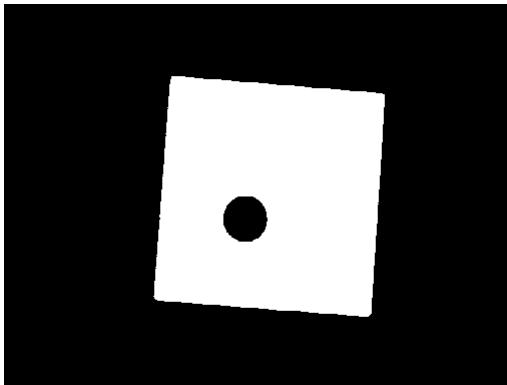
Boundary Extraction

- To detect the inner boundary:
 1. Erode input image
 2. Subtract eroded image from input image
 3. Edges remain!

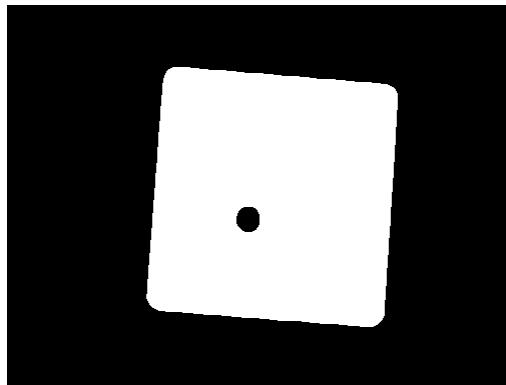
Boundary Extraction

- To detect the outer boundary:

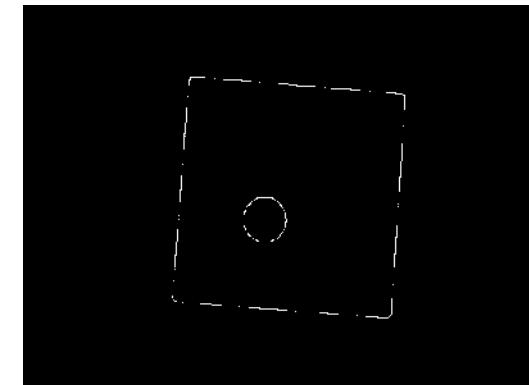
1. Dilate input image
2. Subtract input image from dilated image
3. Edges remain!



A



$A \oplus B$



$(A \oplus B) - A$

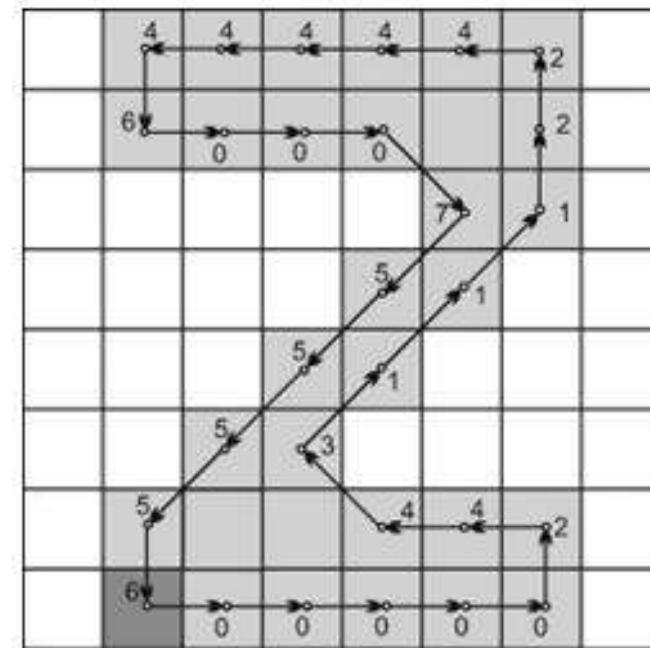
Than You

Digital Image Processing

CHAPTER 11: REPRESENTATION AND DESCRIPTION

Representation & Description

- ❑ After segmentation, the resulting aggregate of segmented pixels usually is represented and described in a form suitable for further computer processing.



Representation & Description

Objective:

To represent and describe information embedded in an image in other forms that are more suitable than the image itself.

Benefits:

- Easier to understand
- Require fewer memory,
- faster to be processed

What kind of information we can use?

Boundary, shape, Region, Texture, Relation between regions

Representation & Description

- ❑ Representing a region involves 2 choices:
 1. We can represent the region in terms of its external characteristics (its boundary)
 2. We can represent the region in terms of its internal characteristics (the pixels contained in a region)
- ❑ After representation, the next task is to describe the region based on the chosen representation.
- ❑ For example, a region may be represented by its boundary, and the boundary described by features such as its length and other features.

Representation

- **External representation** is chosen when the primary focus is on shape characteristics.
- **Internal representation** is chosen when the primary focus is on regional properties such as color and texture.
- Sometimes, **both** ways are used.

Common Representation

- Common external representation methods are:

- **Chain code**
- **Polygonal approximation**
- **Signature**
- **Boundary segments**
- **Skeleton (medial axis)**

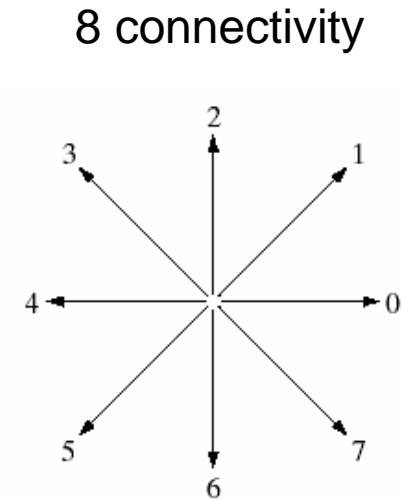
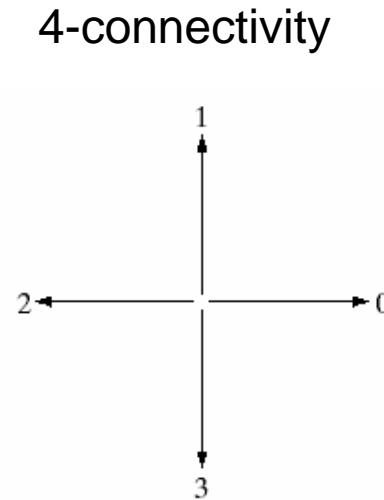
Chain Codes

- ❑ **Represent a boundary** by a connected sequence of straight-line segments of specified length and direction.
 - ❑ Directions are coded using the numbering scheme:

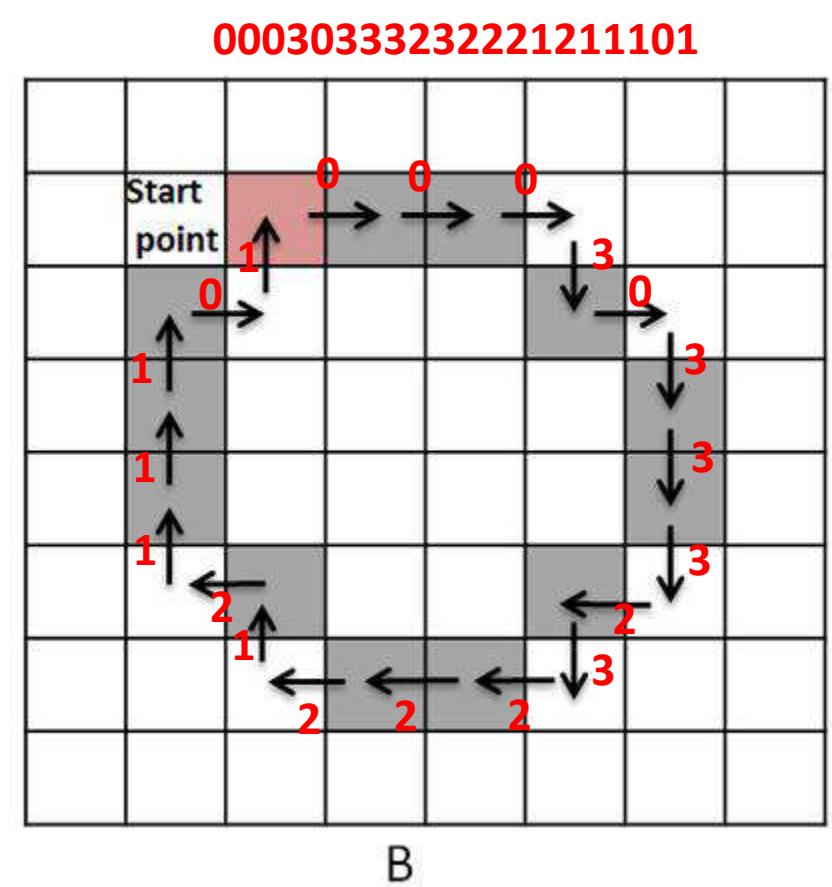
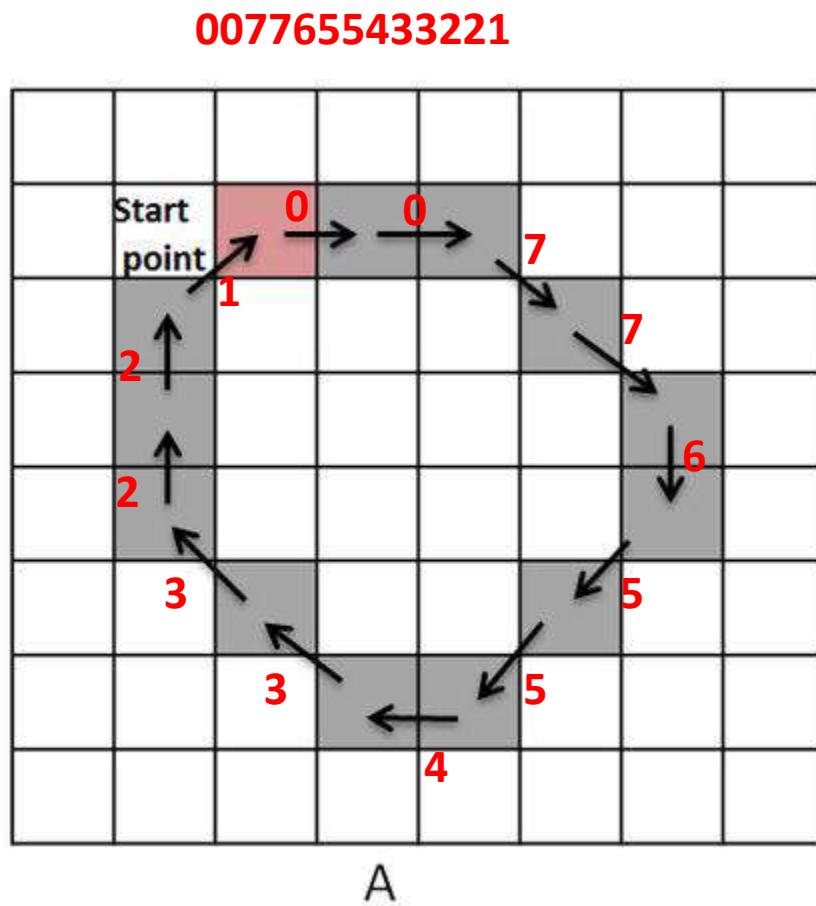
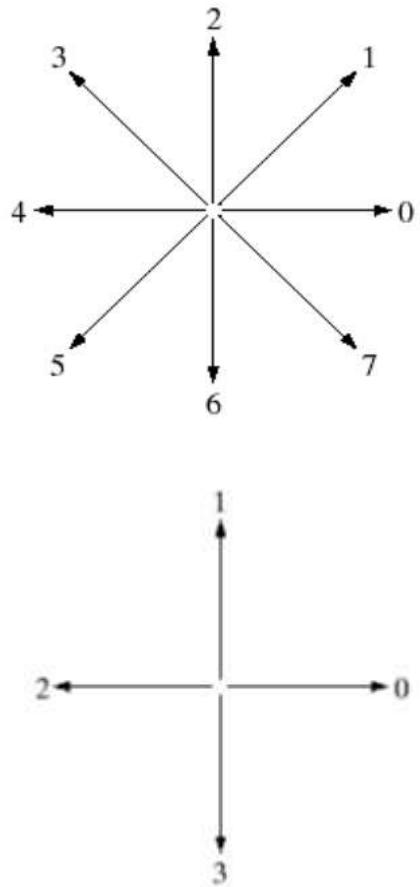
a b

FIGURE 11.1

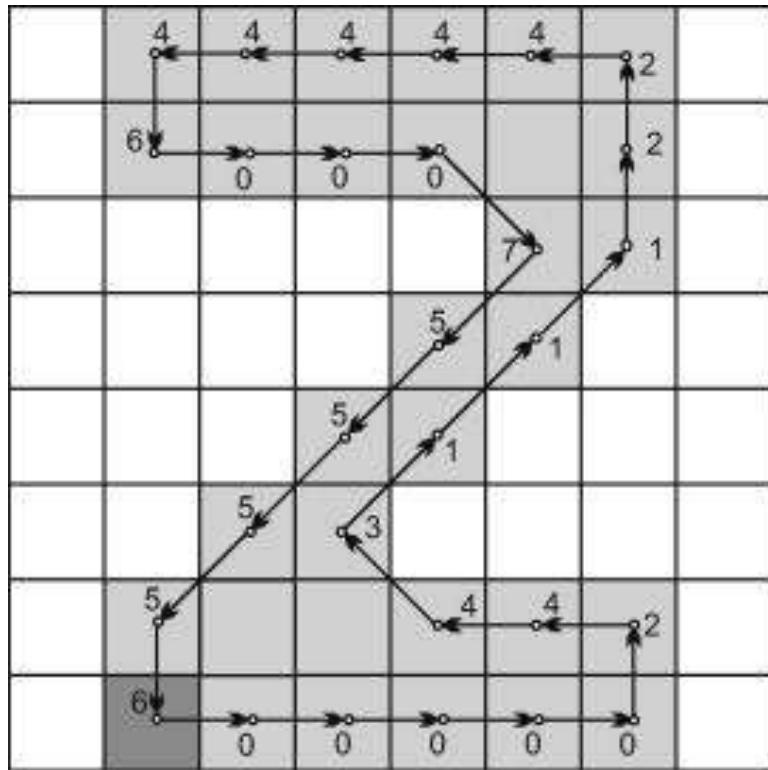
Direction numbers for
 (a) 4-directional chain code, and
 (b) 8-directional chain code.



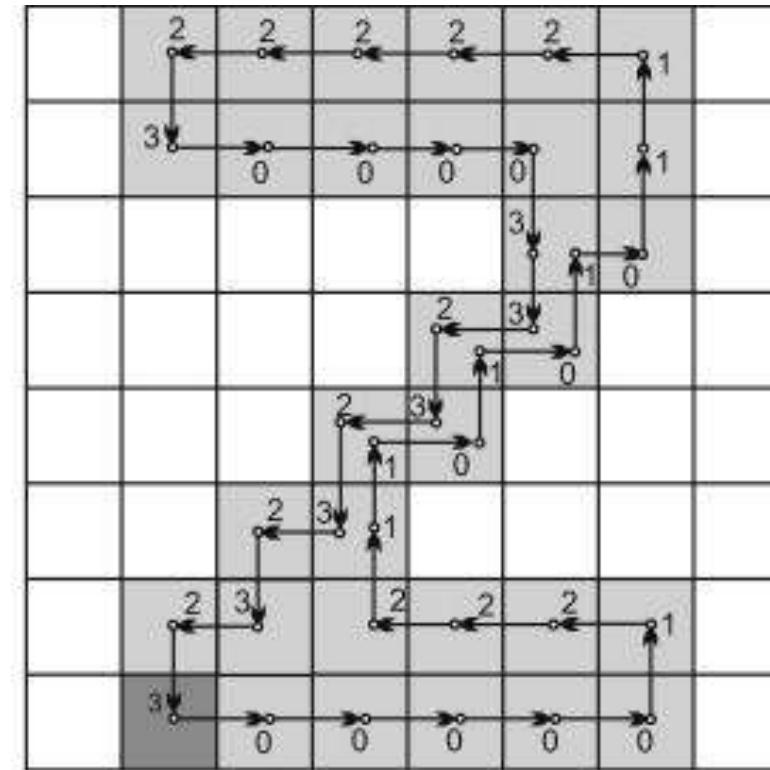
Chain Codes Example



Chain Codes Example



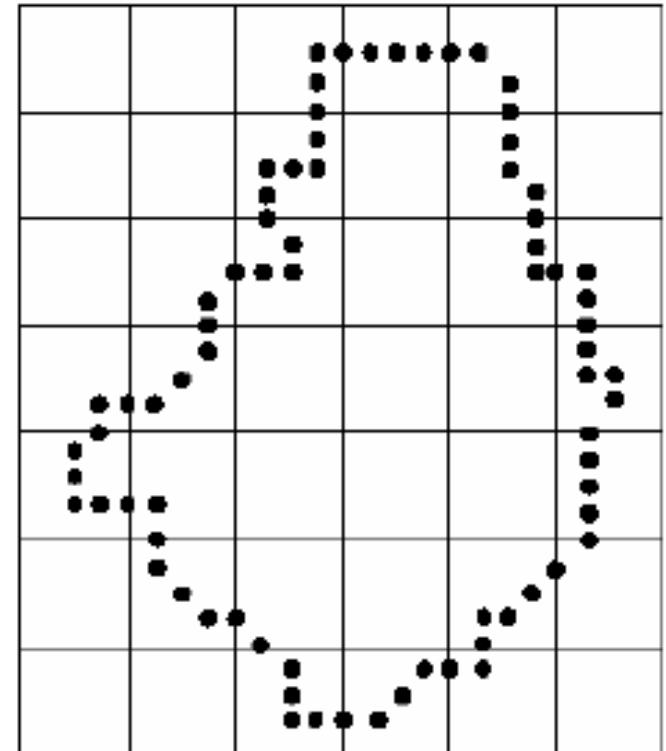
(a)



(b)

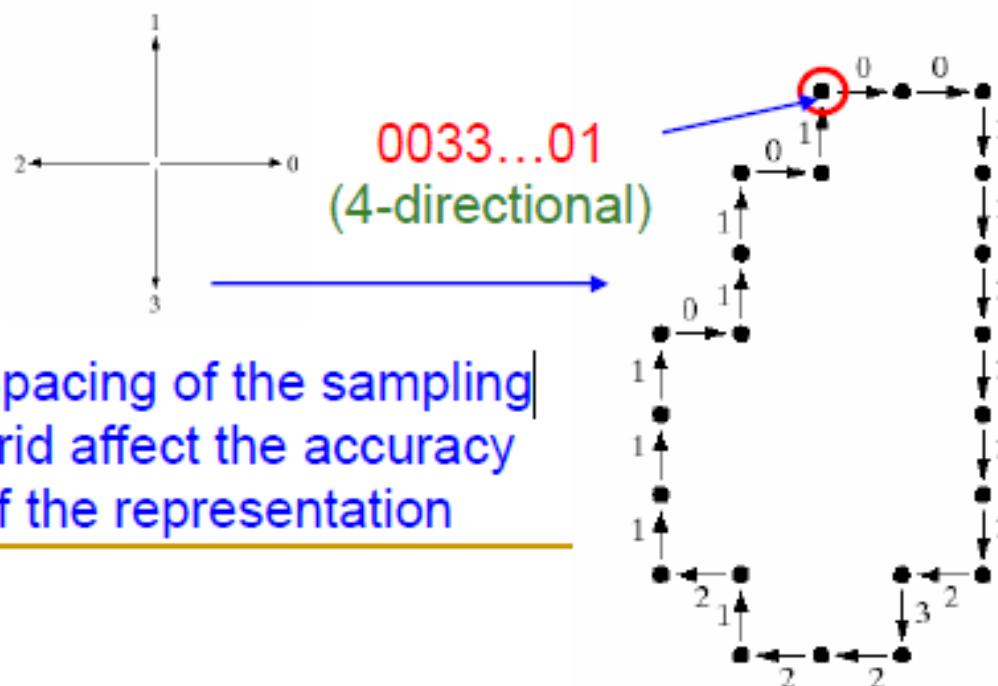
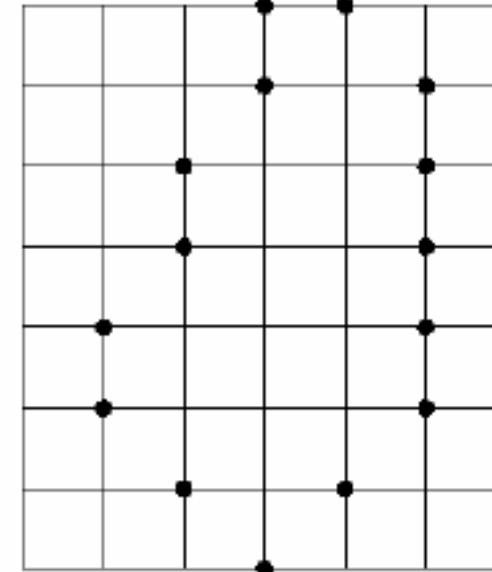
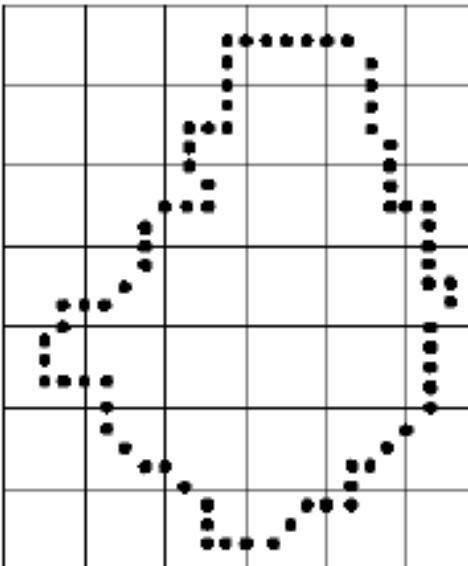
Generation of Chain Codes

- Walking along the boundary in clockwise direction, and assigning a direction to the segments connecting every pair of pixels.
- Problems:
 - Long chain
 - Sensitive to noise
- Remedies?
 - Resampling using larger grid spacing.

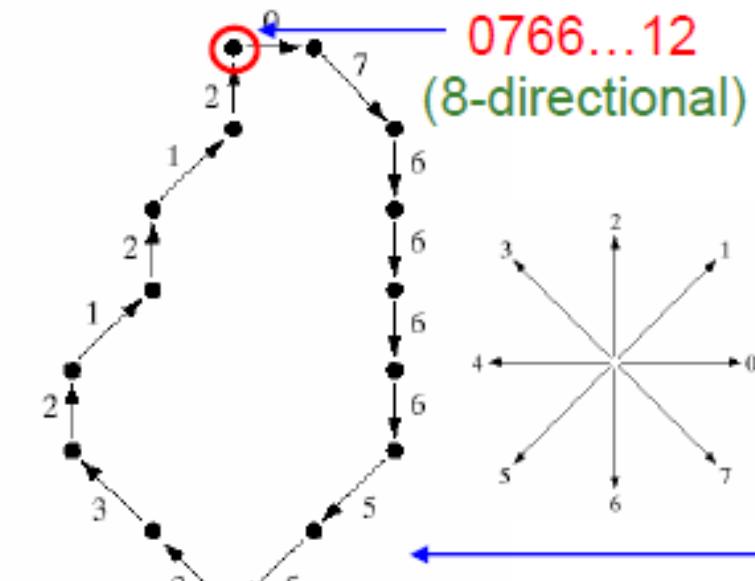


Resampling for Chain Codes

As the boundary traversed, a boundary point is assigned to each node ...



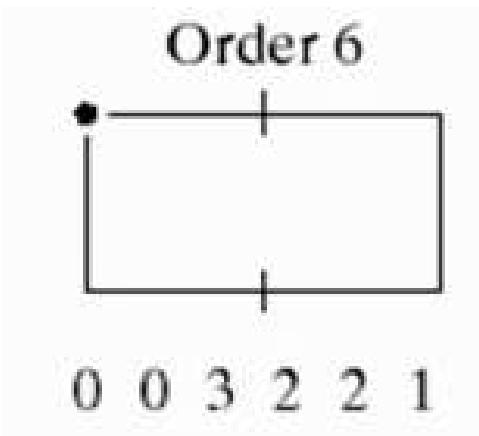
Spacing of the sampling grid affect the accuracy of the representation



Normalization for Chain Codes

- The chain code of the boundary depends on the starting point; solution: **Normalization**.
 - Make the chain code a circular sequence
 - Redefine the starting point which gives an integer of minimum magnitude.
 - E.g. 101003333222 normalized to 003333222101

Normalization for Chain Codes



Treat a chain code as a **circular sequence** and redefine the starting point so that the resulting sequence of numbers forms an integer of minimum magnitude

3 2 2 1 0 0 Normalization → 0 0 3 2 2 1
2 2 1 0 0 3 → 0 0 3 2 2 1
1 0 0 3 2 2 → 0 0 3 2 2 1

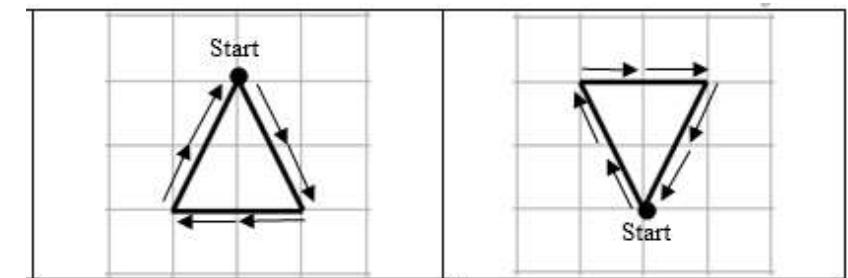
Normalization for Chain Codes

□ Normalize for rotation:

- Using first difference (FD) obtained by counting the number of direction changes in counterclockwise direction that separate 2 adjacent elements of codes.
- E.g FD of a 4-direction chain code 10103322 is 3133030

□ Normalize for scaling:

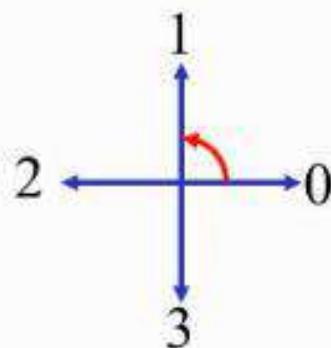
- Altering the size of the resampling grid.



Normalization for Chain Codes

The first difference of a chain code: counting the number of direction change (in counterclockwise) between 2 adjacent elements of the code.

Example: Chain code : The first



difference

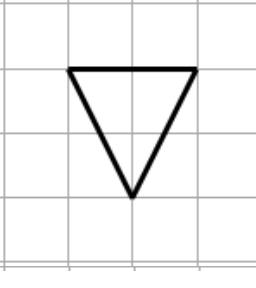
- 0 → 1
- 0 → 2
- 0 → 3
- 2 → 3
- 2 → 0
- 2 → 1

Example:

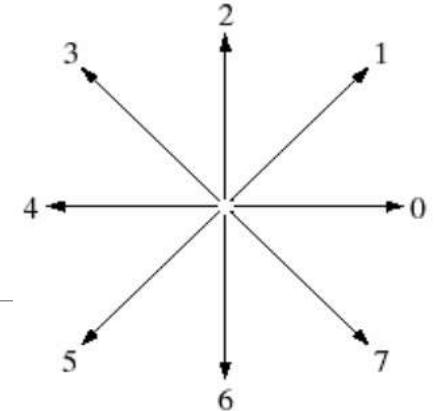
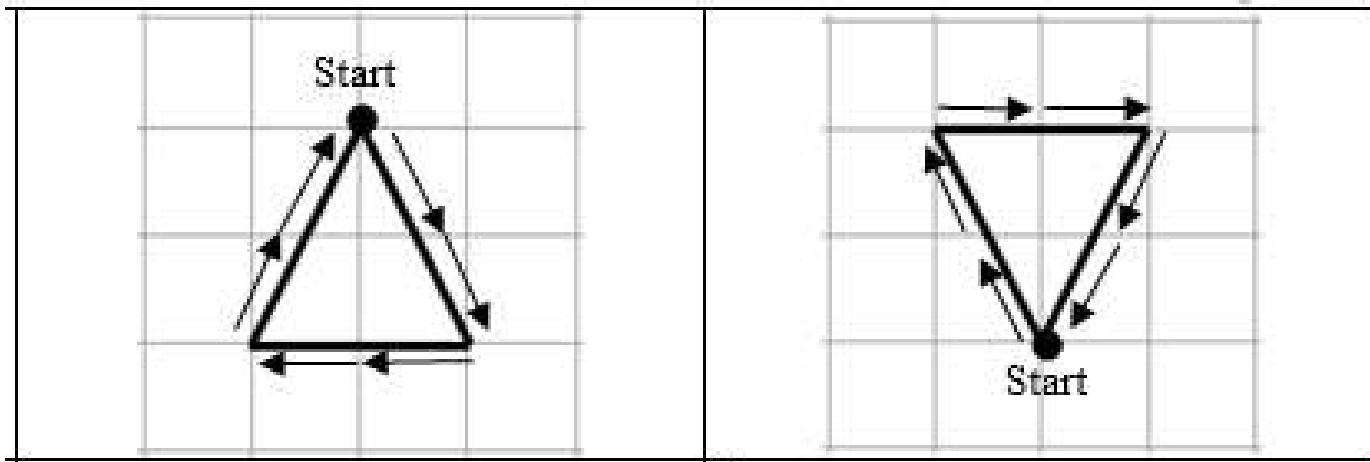
- a chain code: 10103322

- The first difference = 3133030
- Treating a chain code as a circular sequence, we get the first difference = 33133030

The first difference is rotational invariant.



Example on the Chain Code



Chain code

7 7 4 4 1 1

0 0 5 5 3 3

Normalization for rotation

0 5 0 5 0 6

0 5 0 6 0 5

Normalization for starting point

0 5 0 5 0 6

0 5 0 5 0 6

Thank you
