

Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 01

Introduction

What is Digital Image Processing?

1. Digital Signal Processing

- *Digital image processing* can be regarded as a subarea of *Digital Signal Processing*.
- Because of that, in the next slides some basic concepts about Digital Signal Processing will be presented.
- Then Digital Image Processing will be introduced.

1. Digital Signal Processing

- Some applications:
 - Image processing;
 - Video processing;
 - Audio processing;
 - Speech Analysis/Synthesis;
 - Biomedical Engineering;
 - Communications systems;
 - Space exploration; etc.

1. Digital Signal Processing

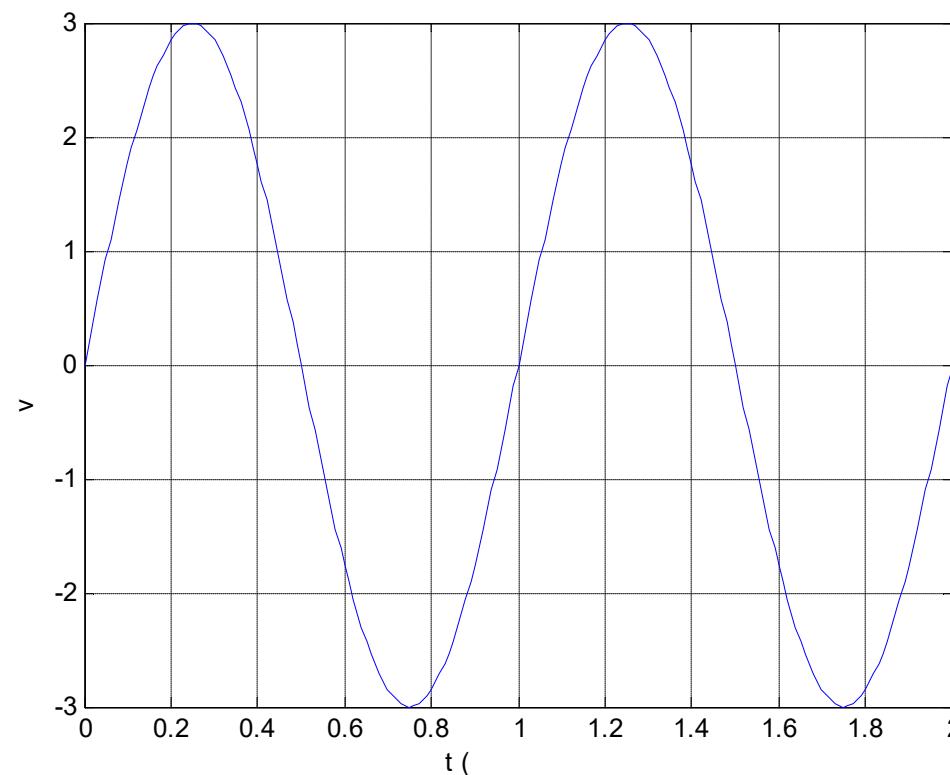
- What is a signal?
 - A **signal** is a description of how one parameter is related to another parameter, typically describing physical phenomena.
 - ✓ For example, a voltage that varies with time.
 - If these parameters assume a continuous range of values, we will call this a **continuous signal**. Most signals observed in nature are continuous.
 - **Analog signals** are those that have the “same aspect” of a continuous signal observed in nature.
 - The **analog-to-digital** conversion allows for the signal to be processed by **digital computers**.

1. Digital Signal Processing

- What is a discrete-time signal?
 - A discrete-time signal results from the **sampling** of the original analog signal:
 - ✓ **Sampling** is the discretization of the *independent variable*.

1. Digital Signal Processing

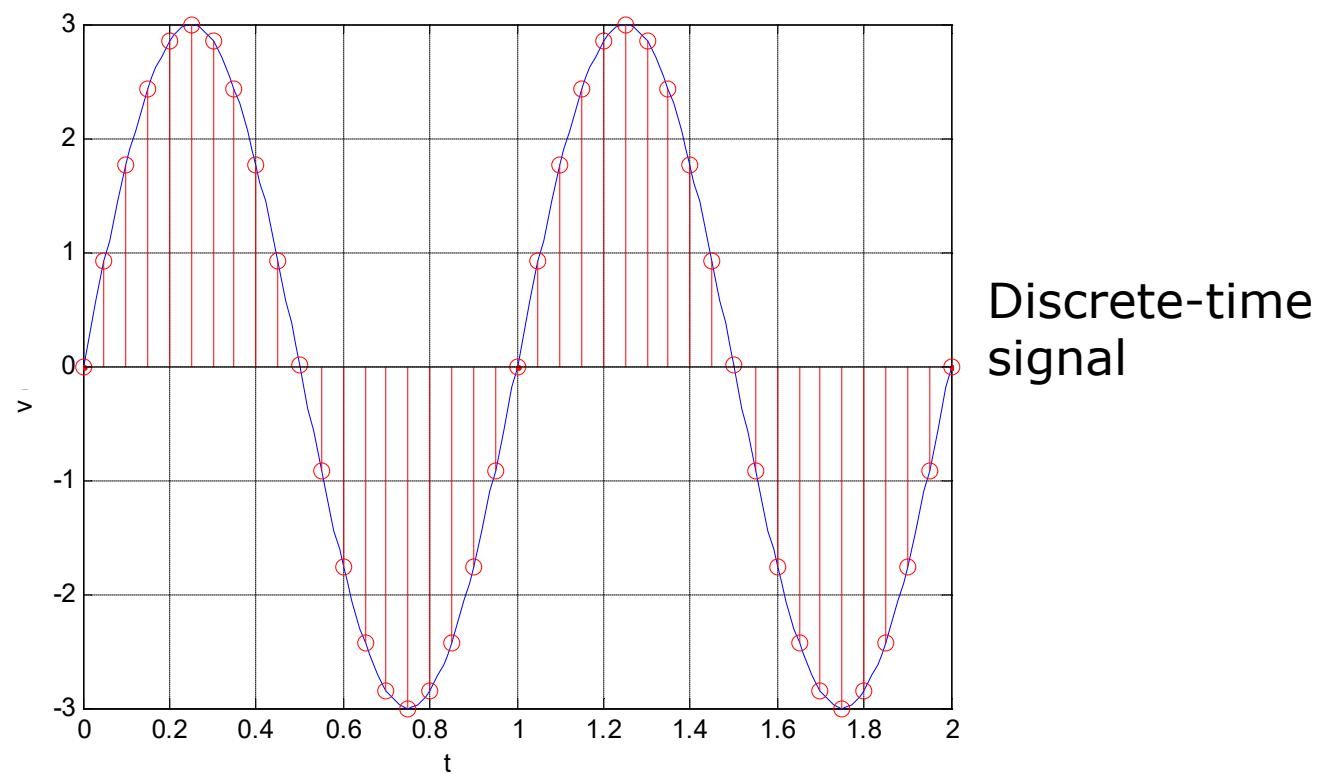
- What is a discrete-time signal?
 - Example: voltage (v) that varies with time (t).



1. Digital Signal Processing

- What is a discrete-time signal?

➤ **Sampling:** discretization of the *independent variable t* .



1. Digital Signal Processing

- What is a discrete-time signal?

➤ Discrete-time signals:

- ✓ Are those that may be represented as a sequence of numbers:

$$\{x[n], n \in \mathbb{Z}\}.$$

- ✓ **x[n]** is the amplitude of the signal at time **nT**, where **T** (**sampling period**) represents the time interval between two consecutive instants where the signal is defined.

- ✓ Thus, an alternative notation would be:

$$\{x[nT], n \in \mathbb{Z}\}.$$

1. Digital Signal Processing

- What is a discrete-time signal?

- $\{x[n], n \in Z\}$

$x[n] \rightarrow \dots, x[0], x[1], x[2], x[3], x[4], \dots$

n represent the nth position of a sample in the array.

- $\{x[nT], n \in Z\}$

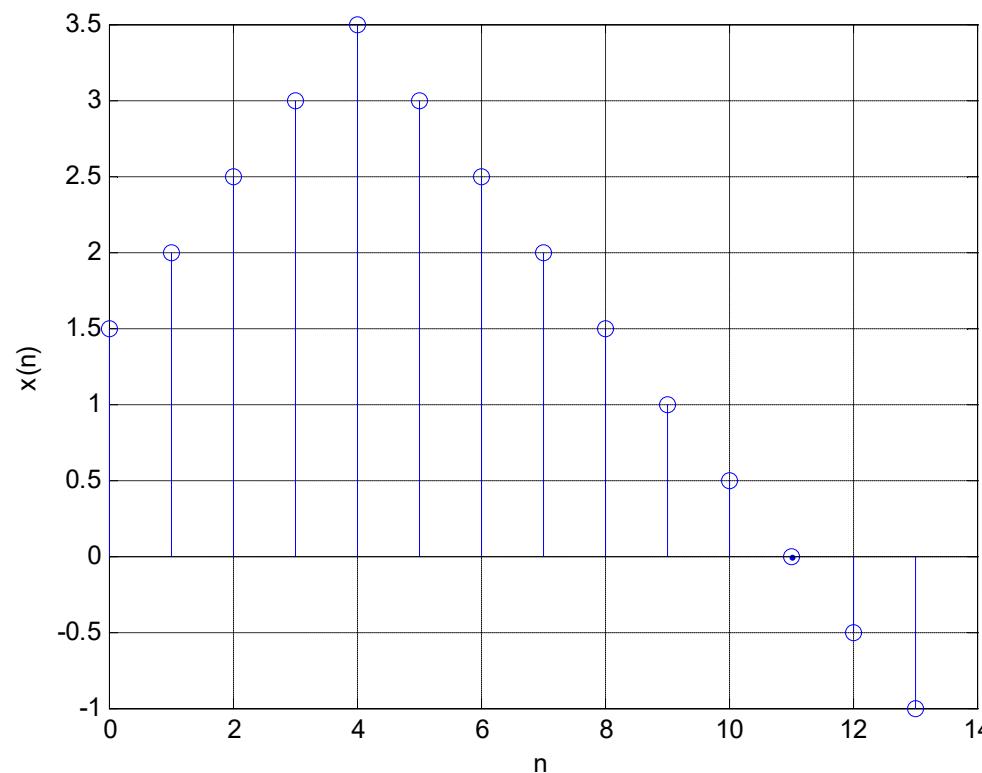
$x[nT] \rightarrow \dots, x[0], x[0.03], x[0.06], x[0.09], x[0.12], \dots$

nT represent the time at which the signal was sampled.
In this example, the distance in time between two consecutive samples is 0.03 seconds.

1. Digital Signal Processing

- What is a discrete-time signal?

➤ Example: $x[n] = [1.5 \ 2.0 \ 2.5 \ 3.0 \ 3.5 \ 3.0 \ 2.5 \ 2.0 \ 1.5 \ 1.0 \ 0.5 \ 0 \ -0.5 \ -1.0]$

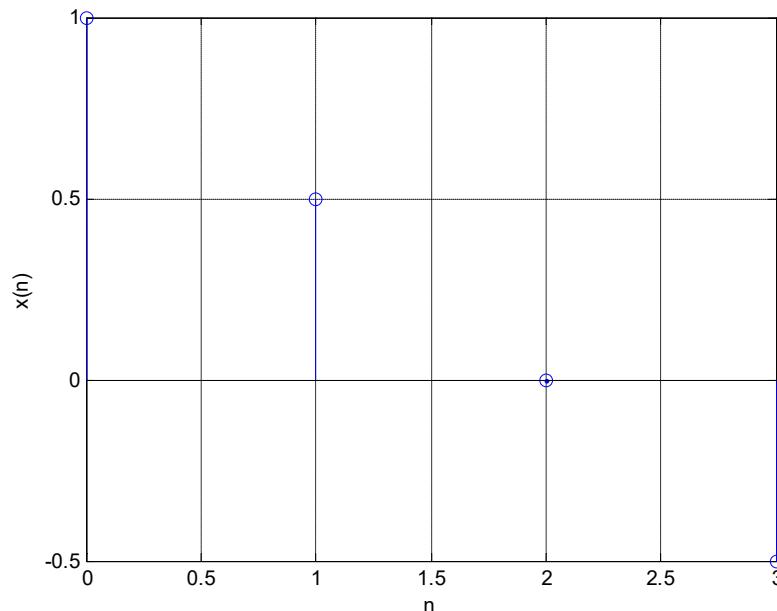


1. Digital Signal Processing

- What is a discrete-time signal?

➤ Example: $x[n] = [1.0 \ 0.5 \ 0 \ -0.5]$

$$x(0) \cdot \delta(n) \quad x(1) \cdot \delta(n-1) \quad x(2) \cdot \delta(n-2) \quad x(3) \cdot \delta(n-3)$$



$$x[n] = \sum_{k=-\infty}^{+\infty} x[k] \cdot \delta[n-k]$$

1. Digital Signal Processing

- What is a discrete-time signal?

➤ IMPORTANT:

- ✓ Although the term discrete-**time** signal is widely used to describe all kinds of discrete signals, time is not always the parameter to appear as the independent variable of acquired signals.

1. Digital Signal Processing

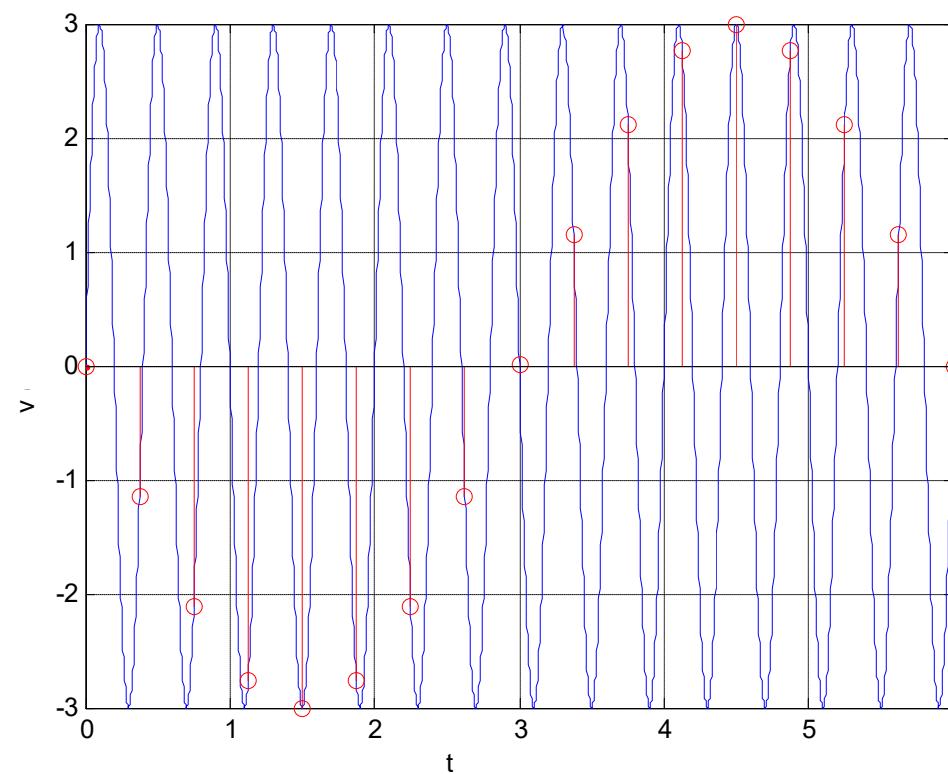
- What is a discrete-time signal?

➤ IMPORTANT:

- ✓ Although the term discrete-**time** signal is widely used to describe all kinds of discrete signals, time is not always the parameter to appear as the independent variable of acquired signals.
- ✓ A geophysicist might acquire measurements of rock density at equally spaced distances along the surface of the earth. In this case, the independent variable is **distance**.
- ✓ To keep things general, it is better to simply label the independent variable as **sample number**.

1. Digital Signal Processing

- What is a discrete-time signal?
 - Problem: sampling frequency.

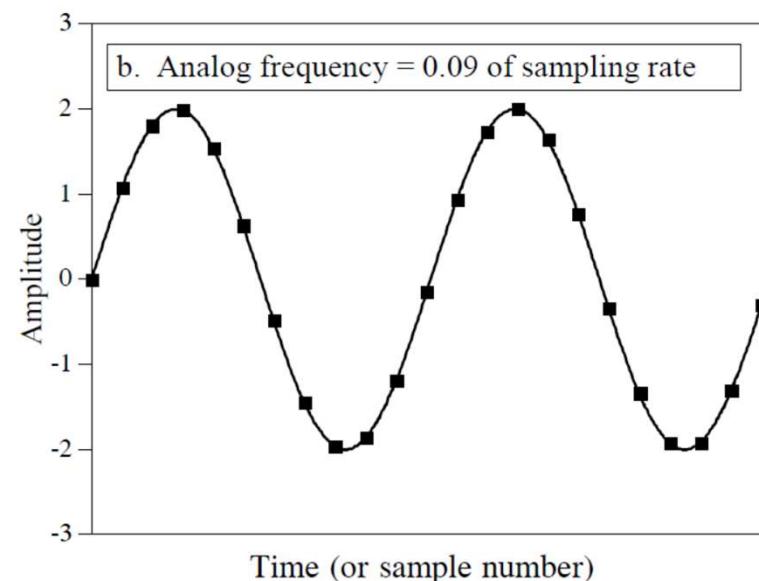
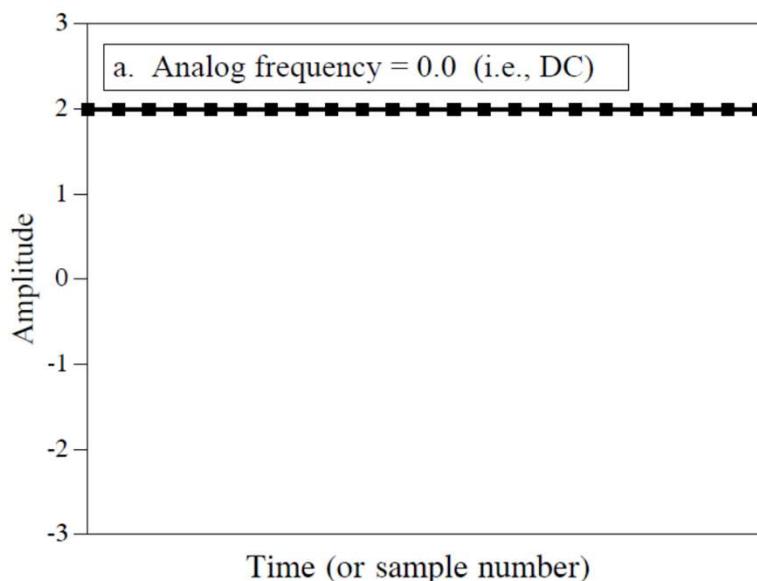


1. Digital Signal Processing

- Nyquist–Shannon sampling theorem:
 - Suppose you sample a continuous signal in some manner. If you can exactly reconstruct the analog signal from the samples, you must have done the sampling *properly*.
 - The sampling theorem indicates that a continuous signal can be properly sampled, *only if it does not contain frequency components above one-half of the sampling rate*.
- ✓ Consider an analog signal composed of frequencies between DC and 3 kHz. To properly digitize this signal it must be sampled at 6,000 samples/sec (6 kHz) or higher.

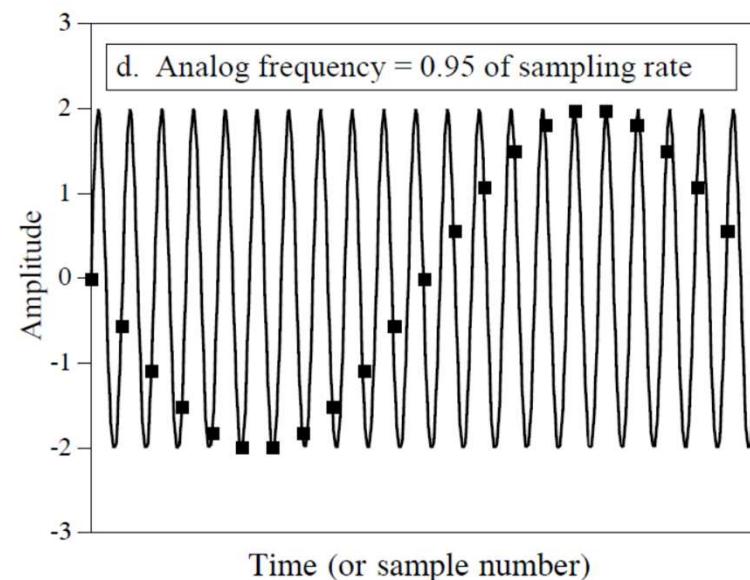
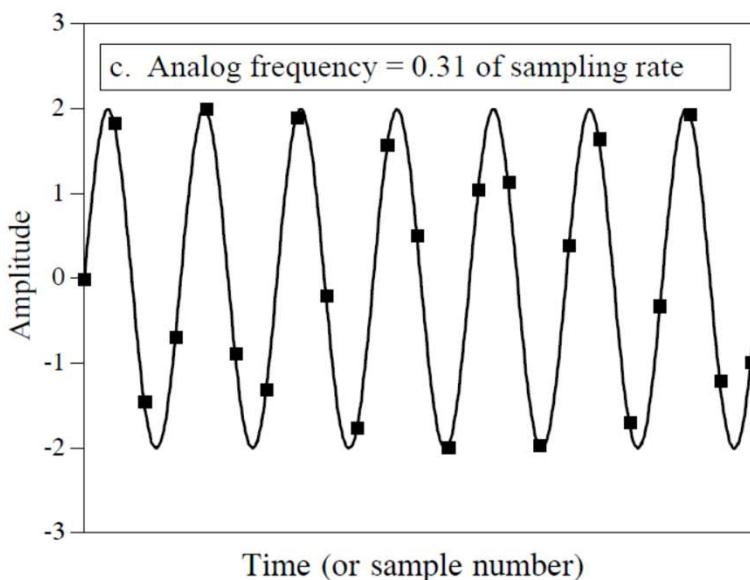
1. Digital Signal Processing

- Nyquist–Shannon sampling theorem: examples.



1. Digital Signal Processing

- Nyquist–Shannon sampling theorem: examples.

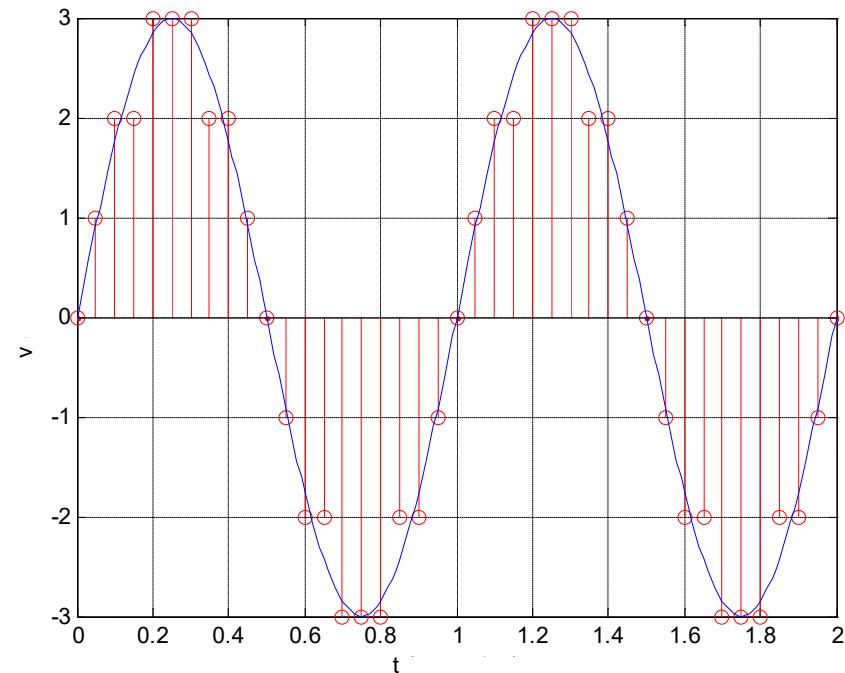
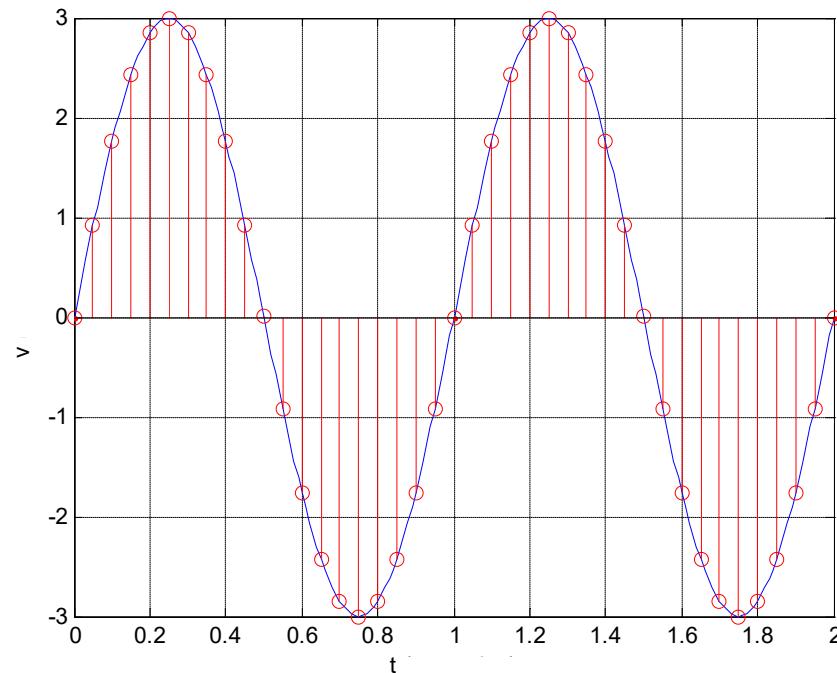


1. Digital Signal Processing

- What is a digital signal?
 - A **digital signal** results from the **quantization** of a discrete-time signal:
 - ✓ **Quantization** is the discretization of the *dependent variable*.

1. Digital Signal Processing

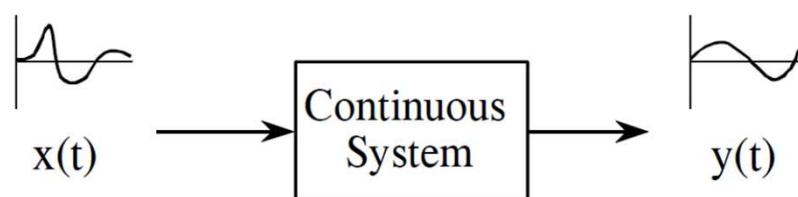
- What is a digital signal?



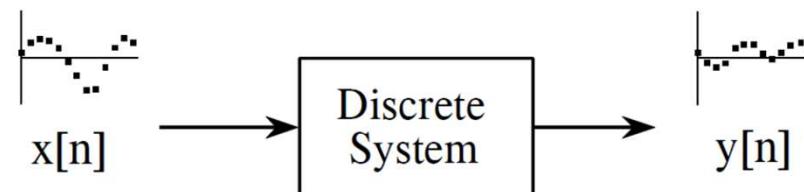
1. Digital Signal Processing

- What is a discrete system?

- A **system** is any process that produces an output signal in response to an input signal
- In **continuous systems** input and output signals are continuous.



- In **discrete systems** input and output signals are discrete.

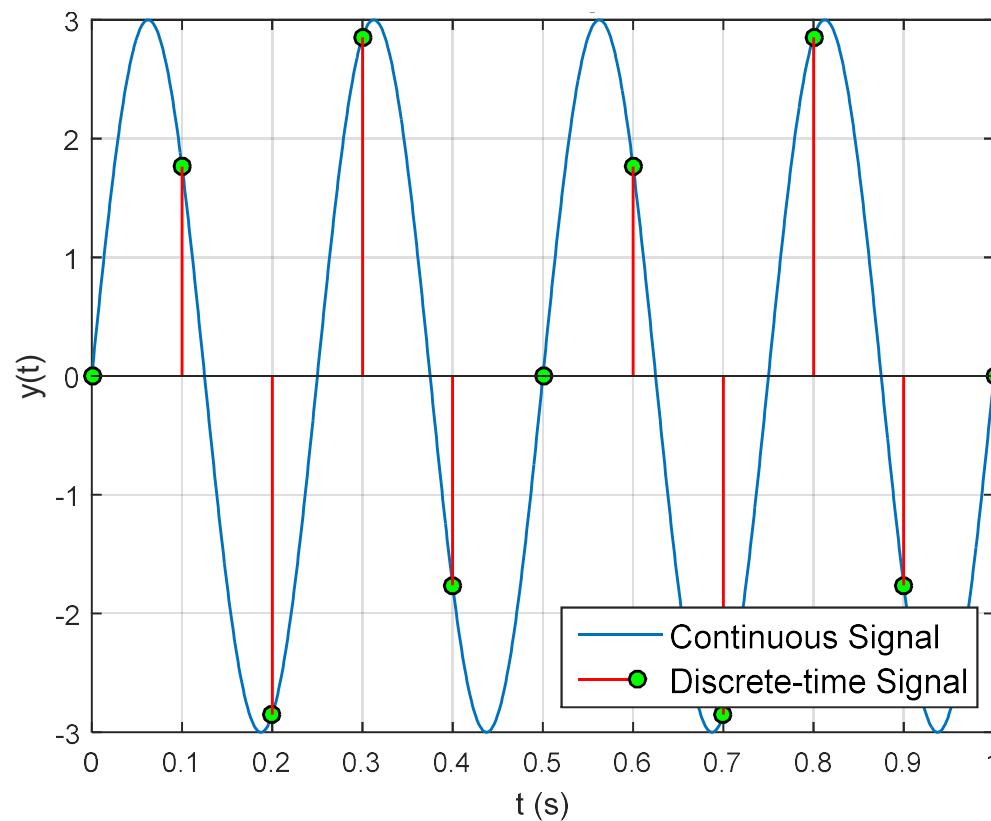


1. Digital Signal Processing

- What is Digital Signal Processing?
 - It is the area in Electrical and Computer Engineering/Computer Science that deals with de **systems** that manipulate **digital signals**.
 - Thus, the development of Digital Signal Processing is intimately tied to the development of the digital computer.

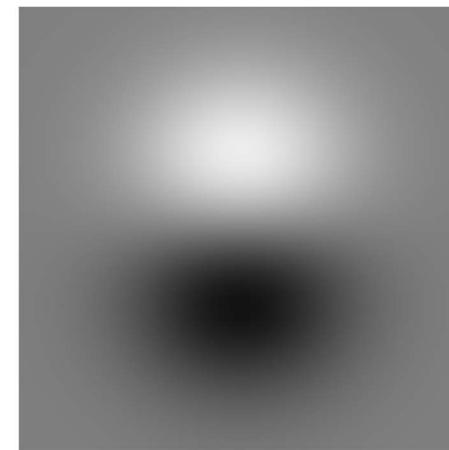
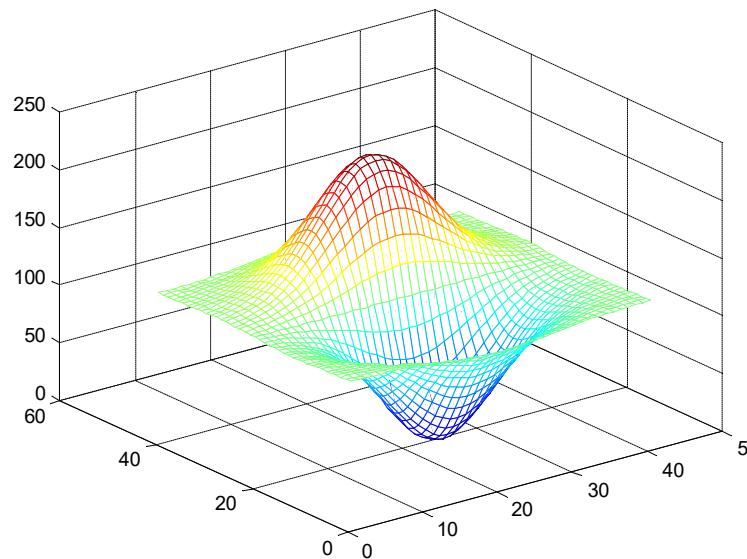
1. Digital Signal Processing

- MATLAB: s24Sampling.m



2. Digital Image Processing

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



2. Digital Image Processing

- MATLAB: s26ImageFunc.m

```
clear all
close all

x = -2:0.1:2;
y = -2:0.1:2;

for i = 1:length(x)
    for j = 1:length(y)
        z(i,j) = round(255*(0.5+ x(i).*exp(-x(i)^2 - y(j)^2)));
    end
End

imshow(uint8(z));
figure
mesh(z);
```

MATLAB Primer

https://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf

2. Digital Image Processing

- When x , y , and the amplitude values of f are all finite, discrete quantities, we call the image a **digital image**.
- **Digital image processing** refers to processing digital images by means of a digital computer.
- Note that a digital image is composed of a finite number of elements, each of which has a particular location and value.
- These elements are referred to as picture elements or **pixels**.
- The human visual system is limited to the visual band of the electromagnetic (EM) spectrum.

2. Digital Image Processing

- Imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves.
 - They can operate on images generated by sources that humans are not accustomed to associating with images. These include:
 - Ultrasound;
 - electron microscopy; and
 - computer-generated images.
- Thus, digital image processing encompasses a wide and varied field of applications.

2. Digital Image Processing

- **Low-level processes:** involve primitive operations such as denoising, contrast enhancement and image sharpening. Inputs and outputs are images.
- **Mid-level:** involves tasks such as segmentation, description of objects and classification of individual objects. Inputs are images, but outputs are attributes extracted from those images.
- **High-level:** involves “making sense” of an ensemble of recognized objects, as in image analysis, and performing the cognitive functions normally associated with vision.
- There are no clear-cut boundaries in the continuum from low-level processes at one end to high-level processes at the other.

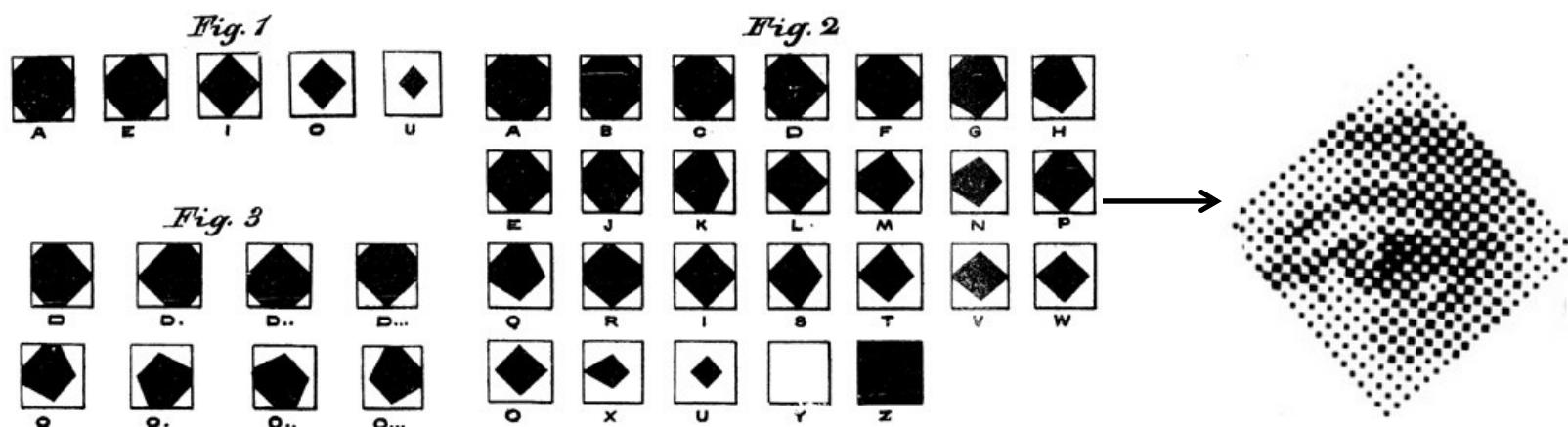
3. The Origins

- One of the first applications of digital images was in the newspaper industry, when pictures were first sent by submarine cable between London and New York (Bartlane cable picture transmission system).
 - Digital picture produced in 1921 from a coded tape by a telegraph printer with special type faces simulating a halftone pattern.



3. The Origins

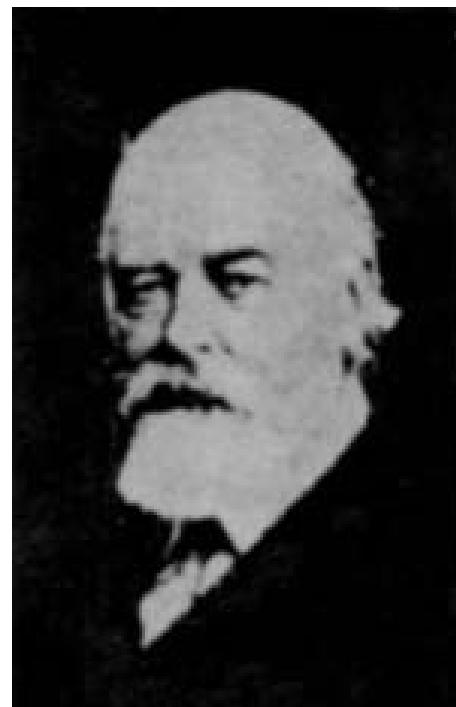
- One of the first applications of digital images was in the newspaper industry, when pictures were first sent by submarine cable between London and New York (Bartlane cable picture transmission system).
 - Digital picture produced in 1921 from a coded tape by a telegraph printer with special type faces simulating a halftone pattern.



<http://www.jmcvey.net/cable/elements/index.htm>

3. The Origins

- Photographic reproduction made in 1922 from a tape punched after the signals had crossed the Atlantic twice. The system was capable of coding images in five distinct levels of gray.



3. The Origins

- ✓ This capability was increased to 15 levels in 1929.



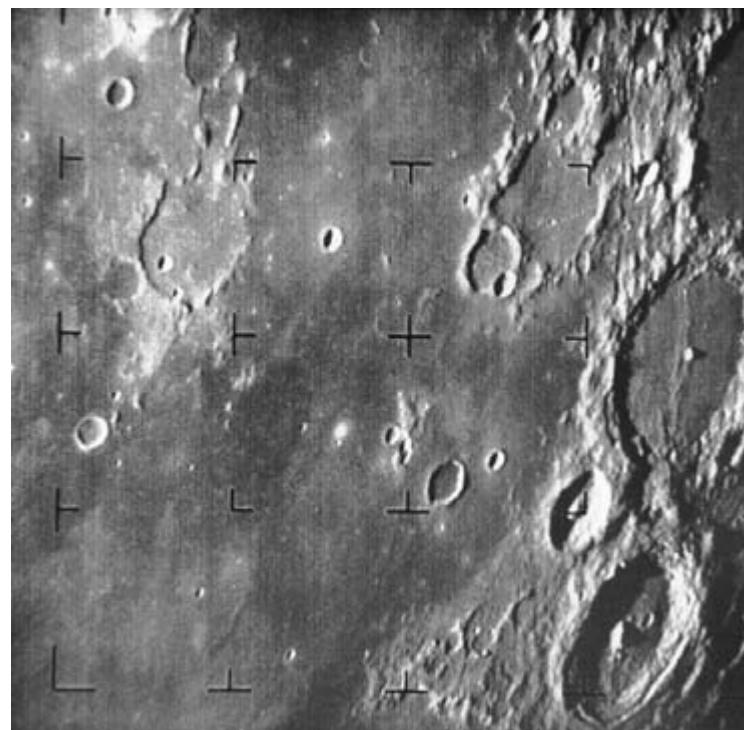
3. The Origins

- The examples just cited are not considered digital image processing results because computers were not involved in their creation.
- Thus, the history of digital image processing is intimately tied to the development of the digital computer.
- The idea of a computer goes back to the invention of the abacus in Asia Minor, more than 5000 years ago.
- But the first computers powerful enough to carry out meaningful image processing tasks appeared only in the early 1960s.



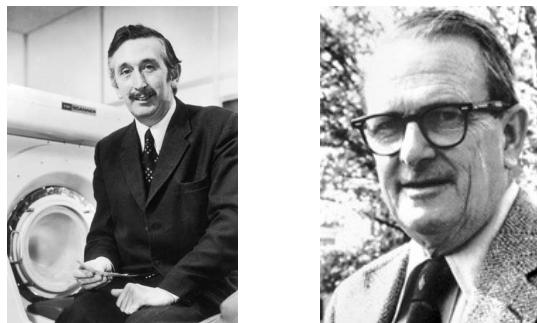
3. The Origins

- In 1964, pictures of the moon were transmitted by Ranger 7 and processed by a computer to correct various types of image distortion inherent in the on-board television camera.



3. The Origins

- In the late 1960s and early 1970s image processing techniques began to be used in medical imaging, remote Earth resources observations, and astronomy.
- The invention in the early 1970s of computerized axial tomography (CAT) is one of the most important events in the application of image processing in medical diagnosis.
- Tomography was invented independently by Sir Godfrey N. Hounsfield and Professor Allan M. Cormack, who shared the 1979 Nobel Prize in Medicine for their invention.

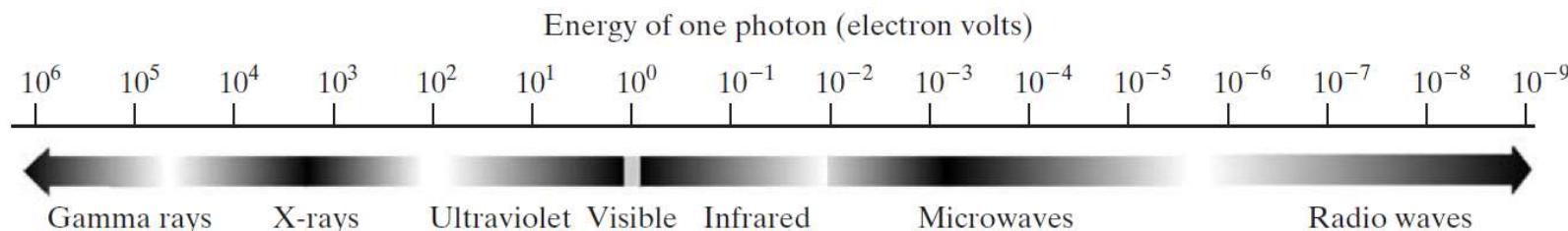


4. Examples of Fields that Use Digital Image Processing

- From the 1960s until the present, the field of image processing has grown vigorously.
- The principal energy source for images in use today is the electromagnetic energy spectrum.
- Other important sources of energy include acoustic, ultrasonic, and electronic (in the form of electron beams used in electron microscopy).
- Synthetic images, used for modeling and visualization, are generated by computer.

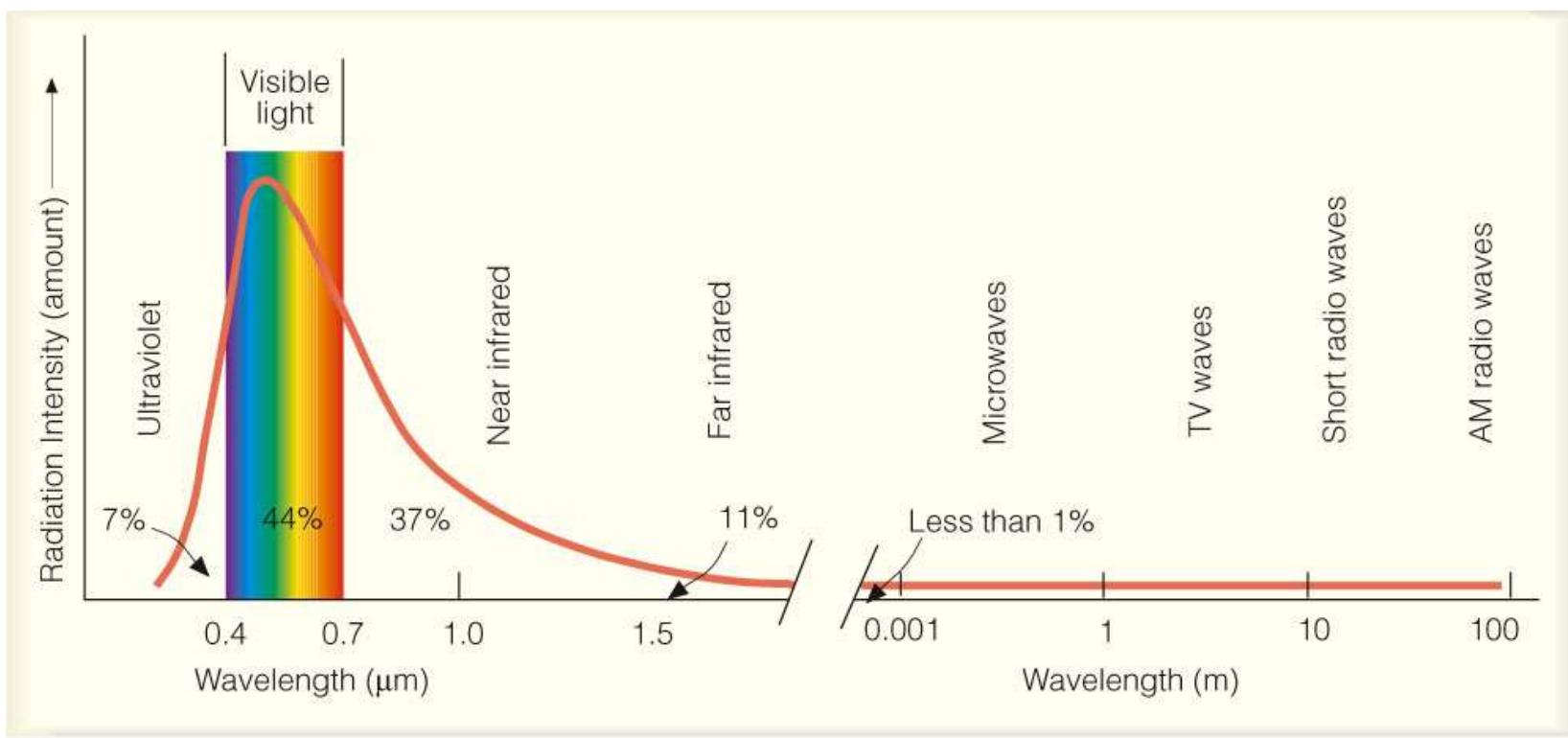
4. Examples of Fields that Use Digital Image Processing

- Electromagnetic waves can be conceptualized as propagating sinusoidal waves of varying wavelengths.
- They can be thought of as a stream of massless particles, each traveling in a wavelike pattern and moving at the speed of light.
- Each massless particle contains a certain amount (or bundle) of energy. Each bundle of energy is called a photon.
- If spectral bands are grouped according to energy per photon, we obtain:



4. Examples of Fields that Use Digital Image Processing

- Curiosity (solar radiation):

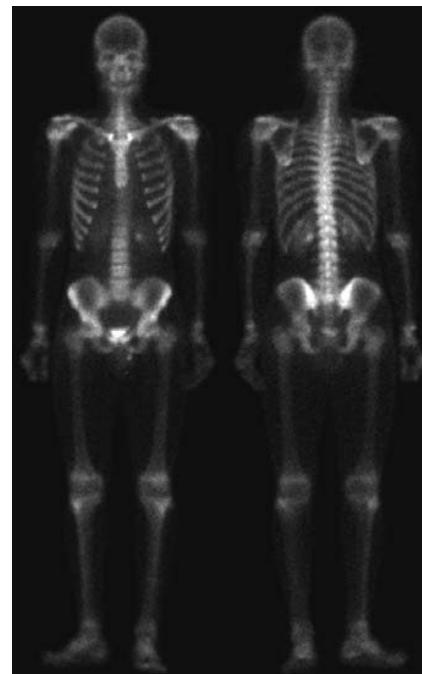


© 2007 Thomson Higher Education

4. Examples of Fields that Use Digital Image Processing

- Gamma-Ray Imaging

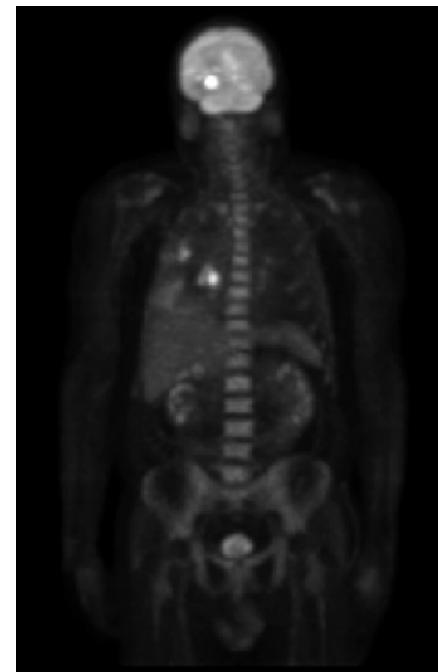
- Nuclear medicine: one approach is to inject a patient with a radioactive isotope that emits gamma rays as it decays. Complete bone scan.



4. Examples of Fields that Use Digital Image Processing

- Gamma-Ray Imaging

- Nuclear medicine: another major modality of nuclear imaging called positron emission tomography (PET).



4. Examples of Fields that Use Digital Image Processing

- X-ray Imaging

- Chest X-ray generated simply by placing the patient between an X-ray source and a film sensitive to X-ray energy.



4. Examples of Fields that Use Digital Image Processing

- X-ray Imaging

➤ Angiography is another major application in an area called *contrast enhancement radiography*. This procedure is used to obtain images of blood vessels.



4. Examples of Fields that Use Digital Image Processing

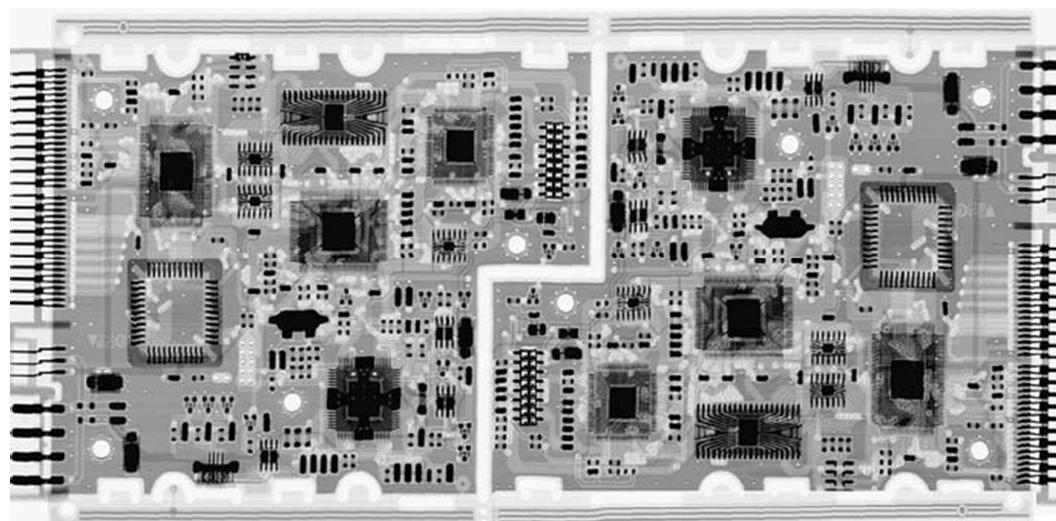
- X-ray Imaging
 - Computerized axial tomography (CAT).



4. Examples of Fields that Use Digital Image Processing

- X-ray Imaging

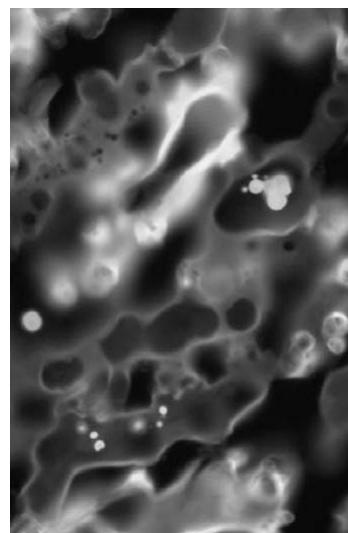
- X-ray image of an electronic circuit board used to examine circuit boards for flaws in manufacturing, such as missing components or broken traces.



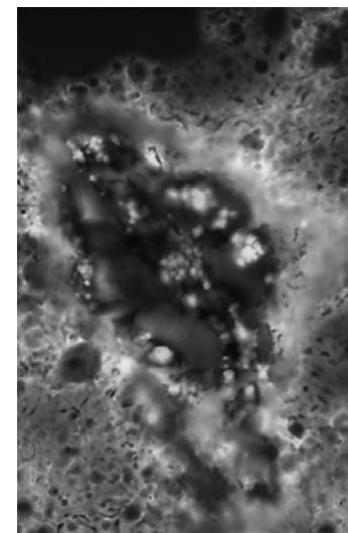
4. Examples of Fields that Use Digital Image Processing

- Imaging in the Ultraviolet Band

- Fluorescence microscopy: is an excellent method for studying materials that can be made to fluoresce, either in their natural form or when treated with chemicals capable of fluorescing.



Normal corn



Smut corn

4. Examples of Fields that Use Digital Image Processing

- Imaging in the Visible and Infrared Bands
 - The visual band of the electromagnetic spectrum is the most familiar in all our activities,
 - ✓ It is not surprising that imaging in this band outweighs by far all the others in terms of scope of application.
 - The infrared band often is used in conjunction with visual imaging, so they have been grouped for the purpose of illustration.

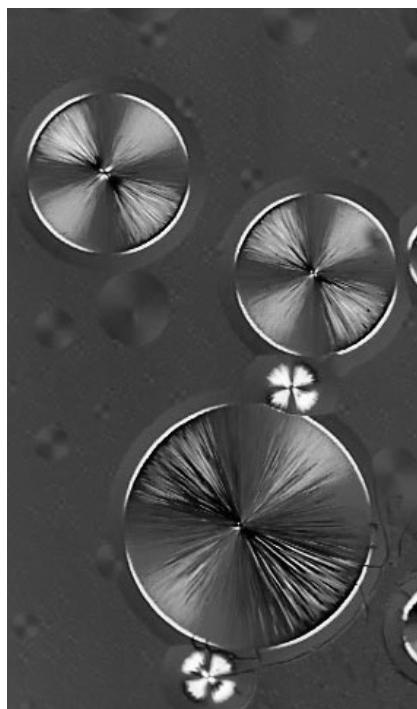


4. Examples of Fields that Use Digital Image Processing

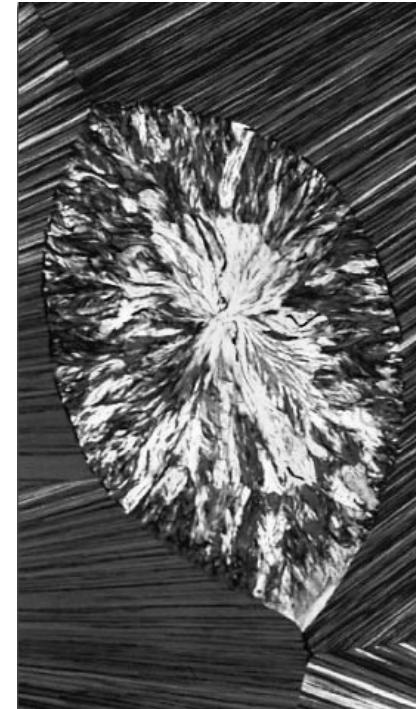
- Imaging in the Visible and Infrared Bands

➤ Light microscopy:

Taxol (anticancer agent)



Cholesterol

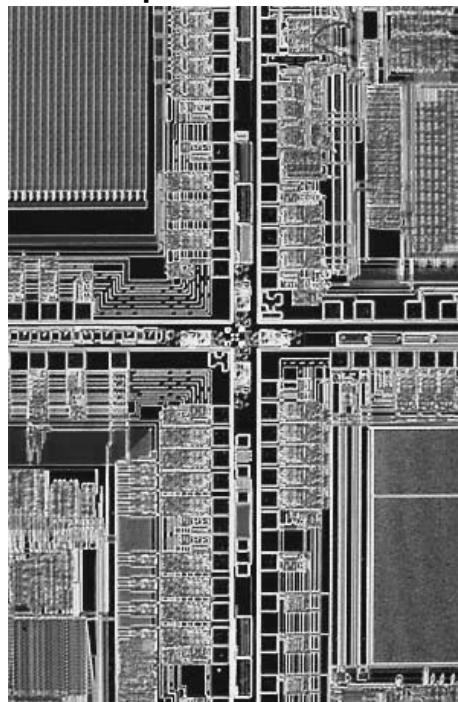


4. Examples of Fields that Use Digital Image Processing

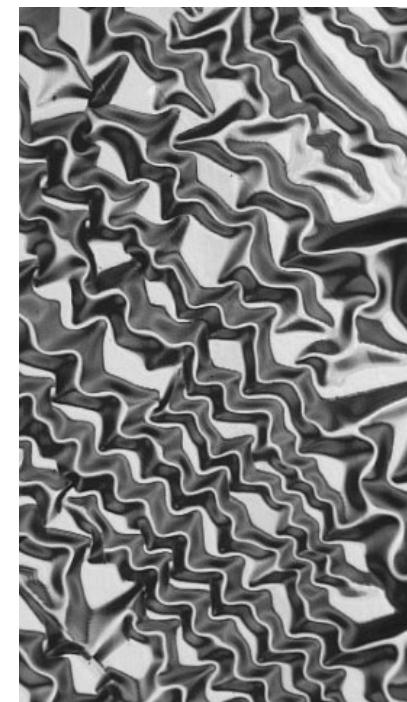
- Imaging in the Visible and Infrared Bands

➤ Light microscopy:

Microprocessor



Nickel oxide thin film



4. Examples of Fields that Use Digital Image Processing

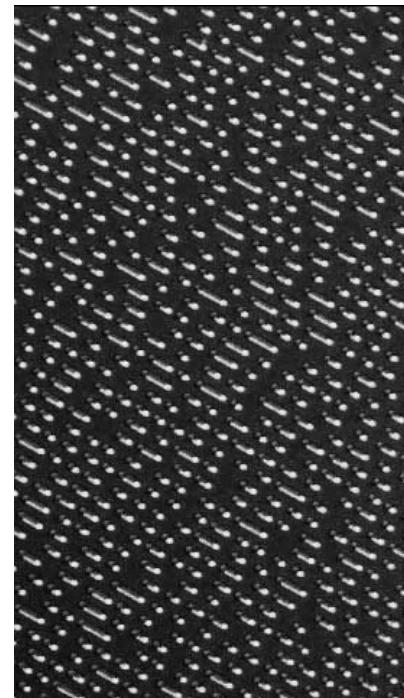
- Imaging in the Visible and Infrared Bands

➤ Light microscopy:

Organic superconductor



Surface of audio CD



4. Examples of Fields that Use Digital Image Processing

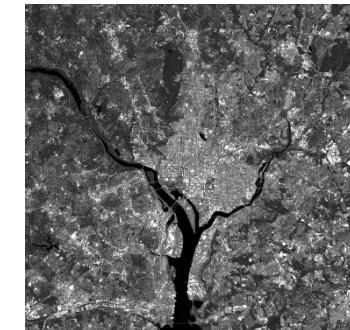
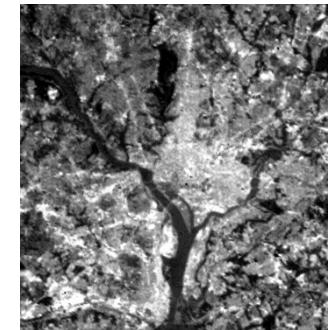
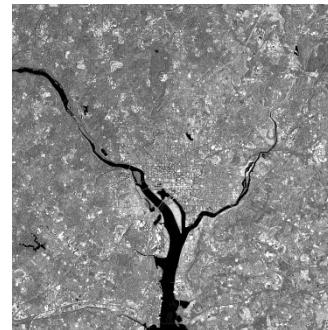
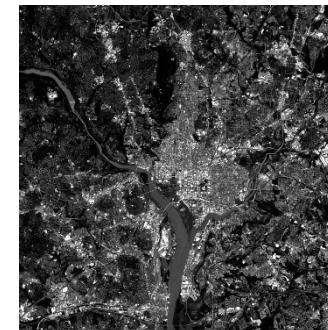
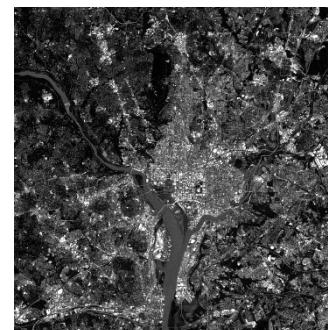
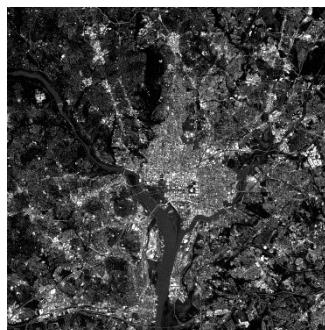
- Imaging in the Visible and Infrared Bands

➤ Another major area of visual processing is **remote sensing** (monitoring environmental conditions)

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

4. Examples of Fields that Use Digital Image Processing

- Imaging in the Visible and Infrared Bands
 - LANDSAT satellite images of the Washington, D.C. area (thematic bands from 1 to 7):



4. Examples of Fields that Use Digital Image Processing

- Imaging in the Visible and Infrared Bands

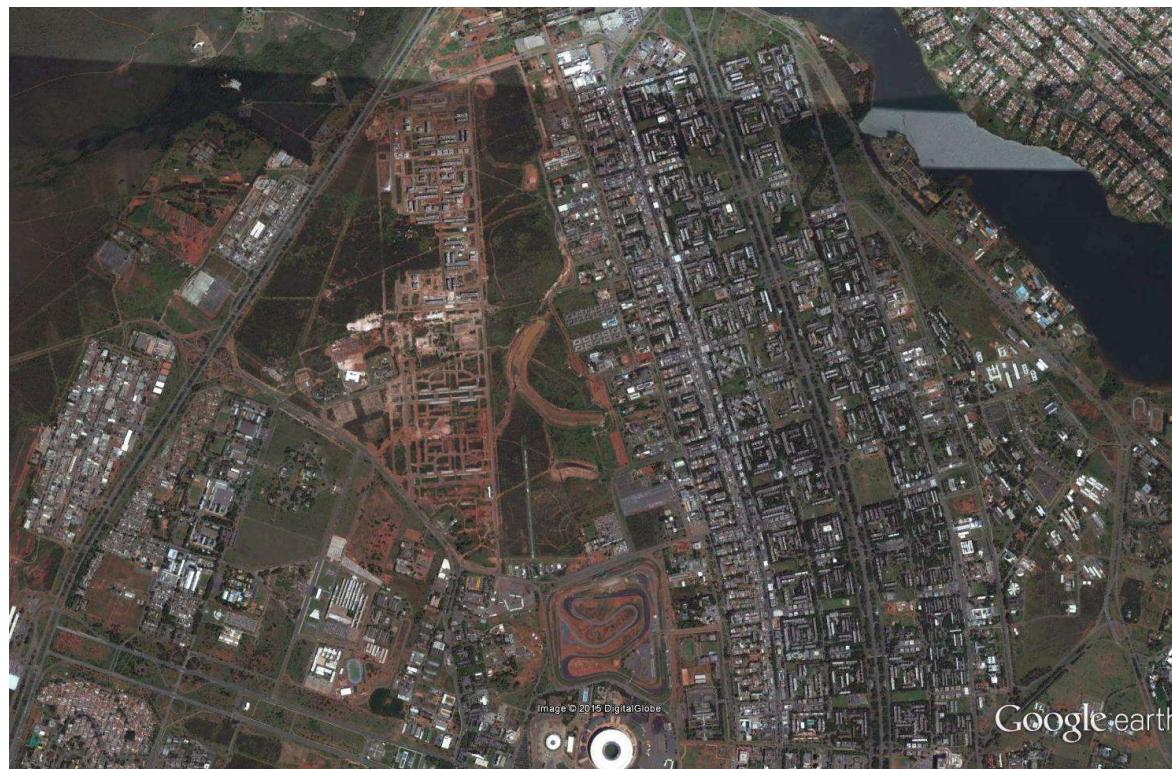
➤ Setor Noroeste, Brasília, Brazil, 2002:



4. Examples of Fields that Use Digital Image Processing

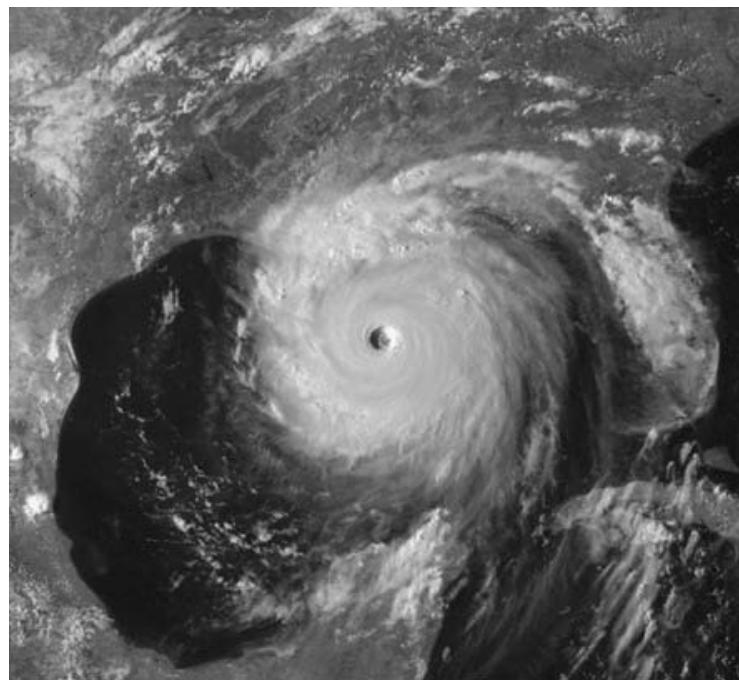
- Imaging in the Visible and Infrared Bands

➤ Setor Noroeste, Brasília, Brazil, 2015:



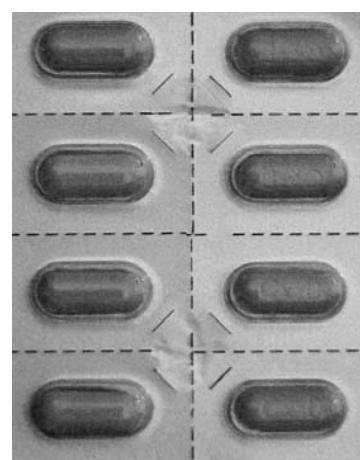
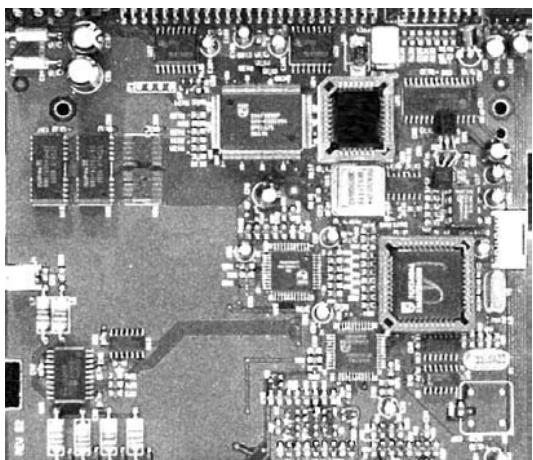
4. Examples of Fields that Use Digital Image Processing

- Imaging in the Visible and Infrared Bands
 - Weather observation and prediction (Multispectral image of Hurricane – visible and infrared bands):



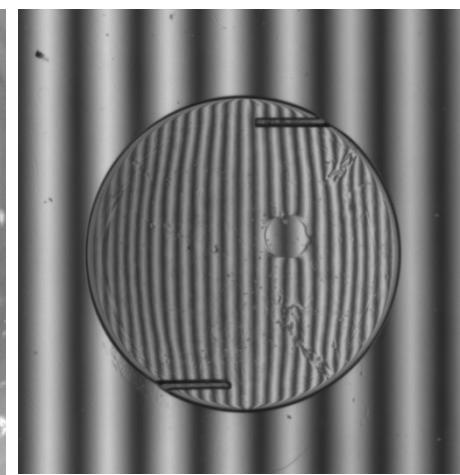
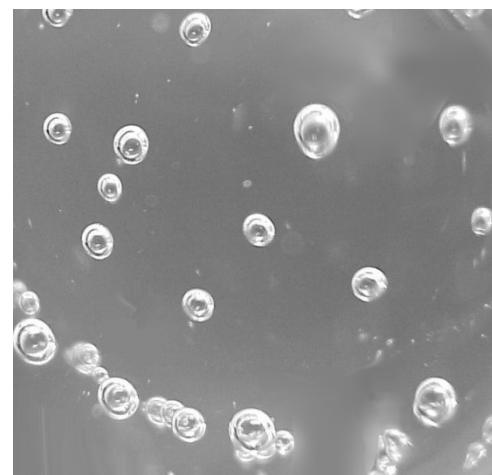
4. Examples of Fields that Use Digital Image Processing

- Imaging in the Visible and Infrared Bands
 - A major area of imaging in the visual spectrum is in automated visual inspection of manufactured goods.



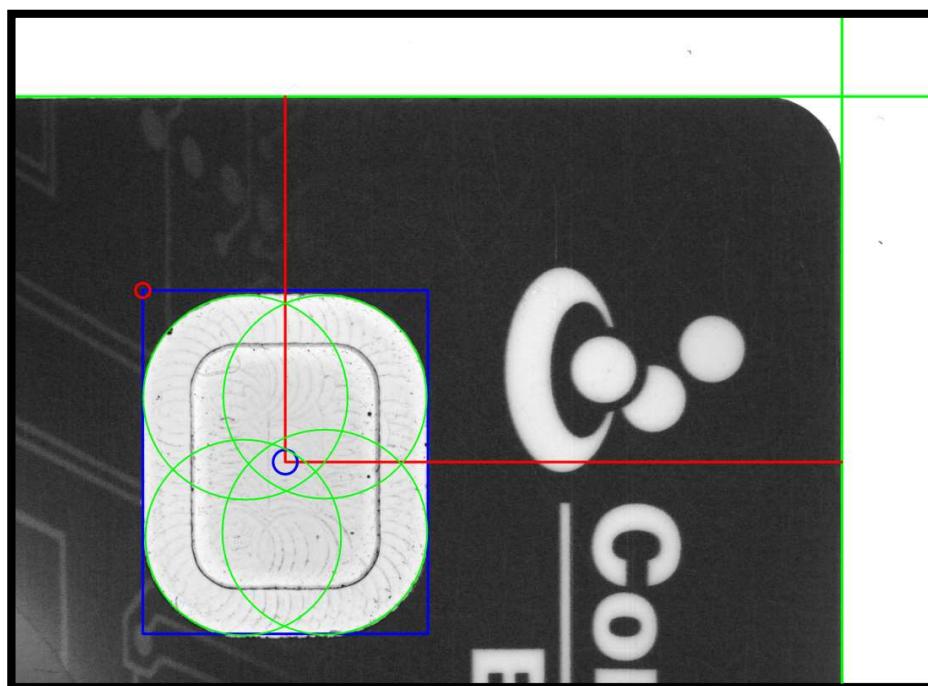
4. Examples of Fields that Use Digital Image Processing

- Imaging in the Visible and Infrared Bands
 - A major area of imaging in the visual spectrum is in automated visual inspection of manufactured goods.



4. Examples of Fields that Use Digital Image Processing

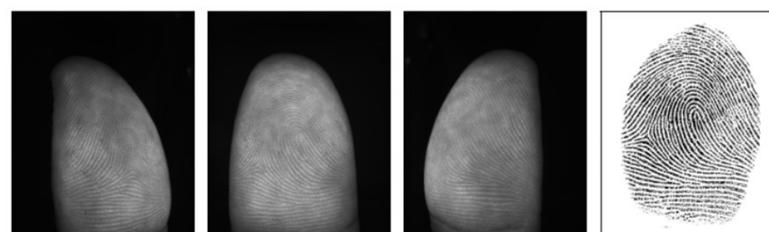
- Imaging in the Visible and Infrared Bands
 - A major area of imaging in the visual spectrum is in automated visual inspection of manufactured goods.



4. Examples of Fields that Use Digital Image Processing

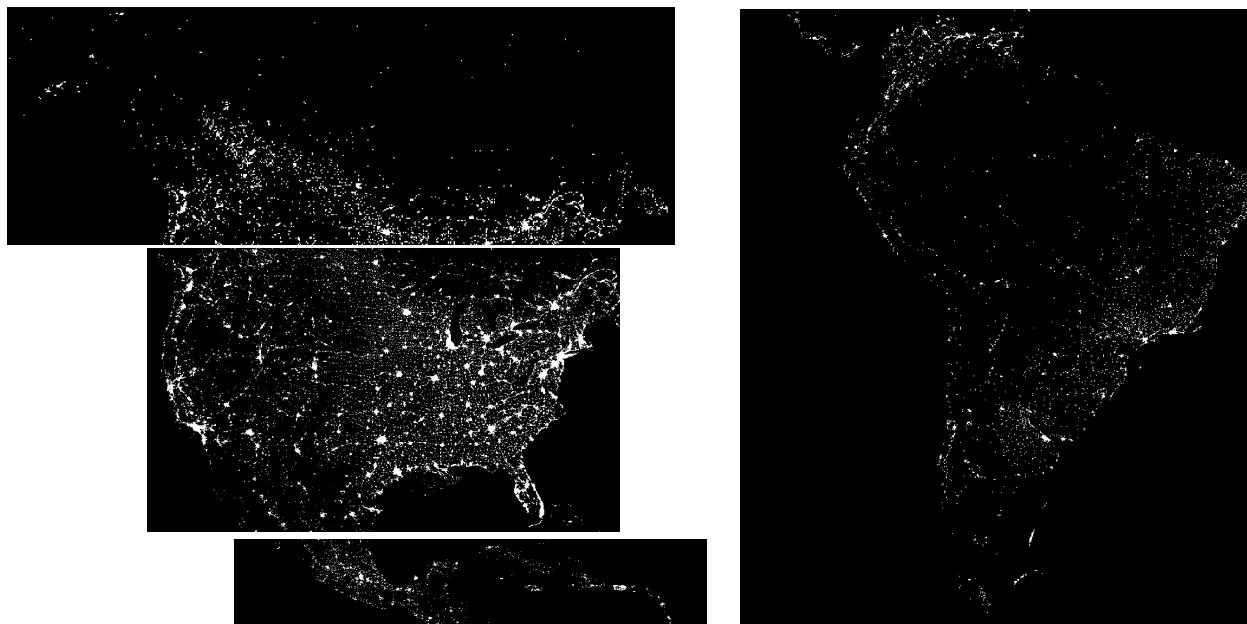
- Imaging in the Visible and Infrared Bands

➤ Other applications:



4. Examples of Fields that Use Digital Image Processing

- Imaging in the Visible and Infrared Bands
 - ✓ Global inventory of human settlements (infrared):



Nighttime Lights of the World data set

4. Examples of Fields that Use Digital Image Processing

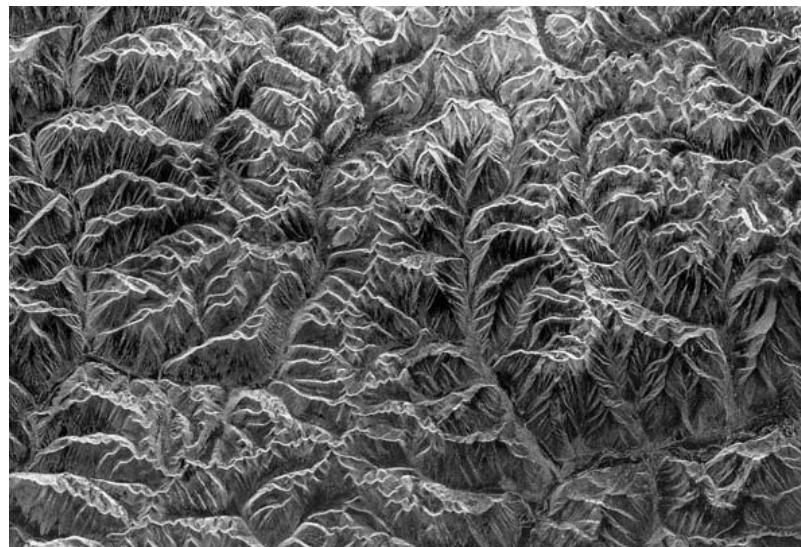
- Imaging in the Visible and Infrared Band:

- ✓ 3D video depth map:



4. Examples of Fields that Use Digital Image Processing

- Imaging in the Microwave Band
 - ✓ The dominant application of imaging in the microwave band is **radar**.



Mountainous area of southeast Tibet

4. Examples of Fields that Use Digital Image Processing

- Imaging in the Radio Band:

- ✓ In medicine radio waves are used in magnetic resonance imaging (MRI).

Knee



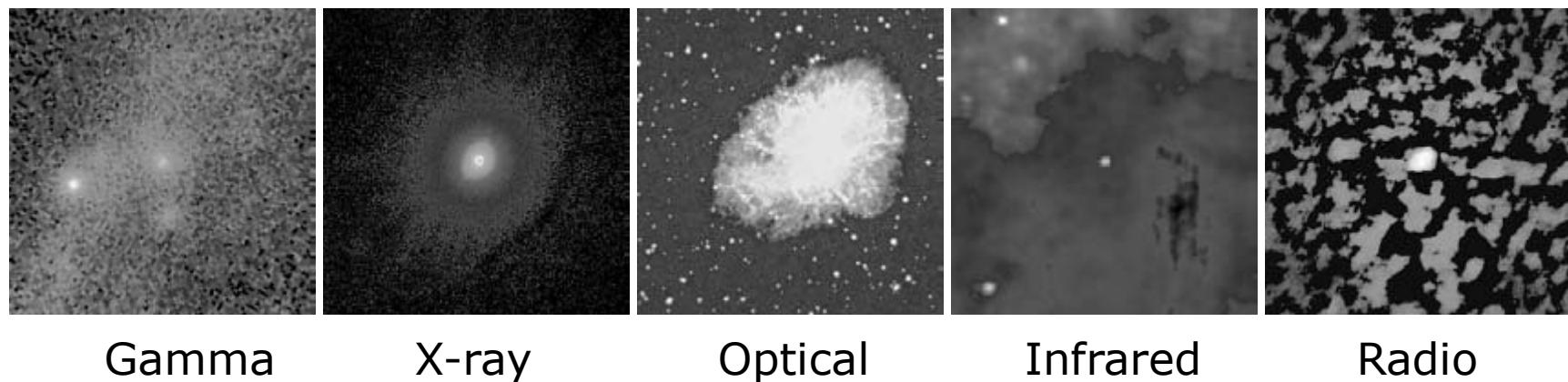
Spine



4. Examples of Fields that Use Digital Image Processing

- Multispectral imaging

✓ Astronomy (images of the Crab Pulsar covering the electromagnetic spectrum):



Gamma

X-ray

Optical

Infrared

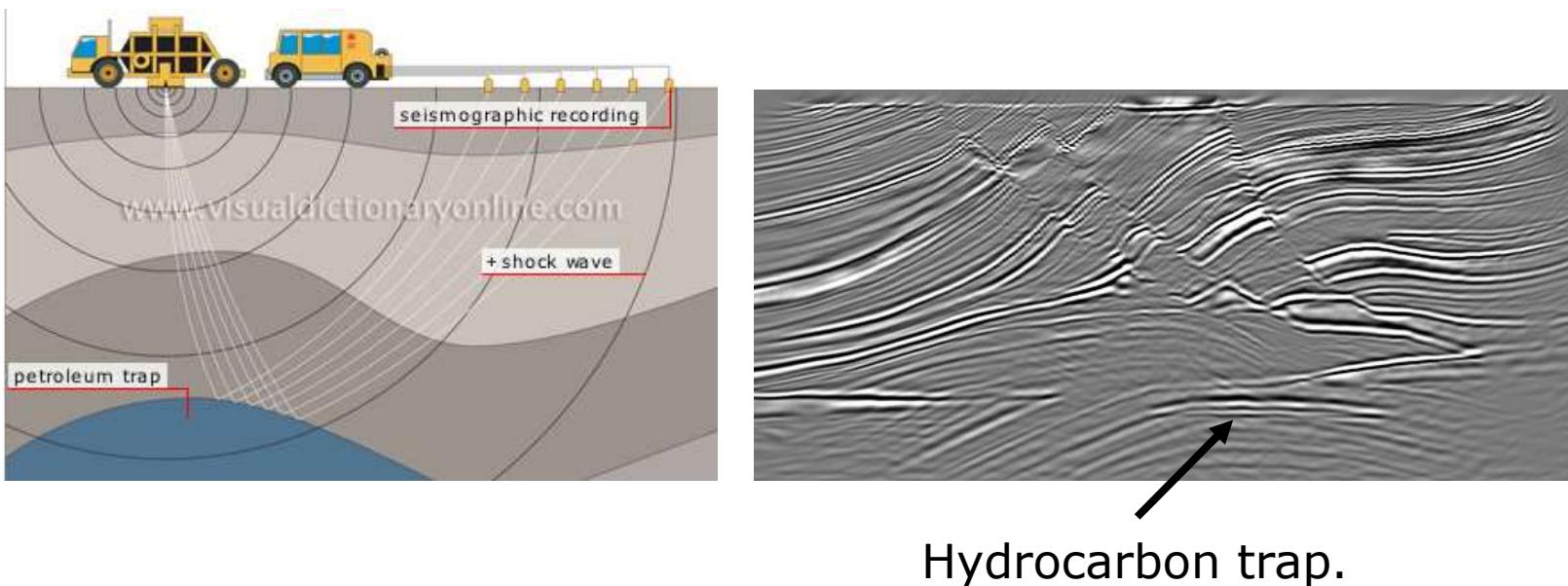
Radio

4. Examples of Fields that Use Digital Image Processing

- Examples in which Other Imaging Modalities Are Used

✓ Acoustic imaging:

➤ Geology (mineral and oil exploration):

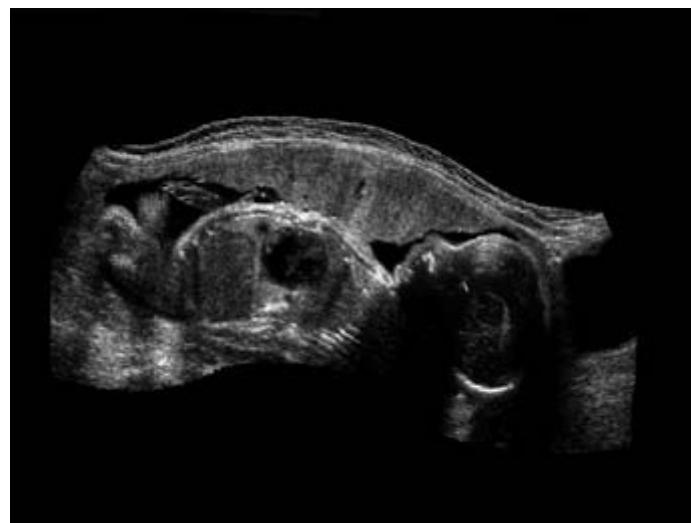


4. Examples of Fields that Use Digital Image Processing

- Examples in which Other Imaging Modalities Are Used

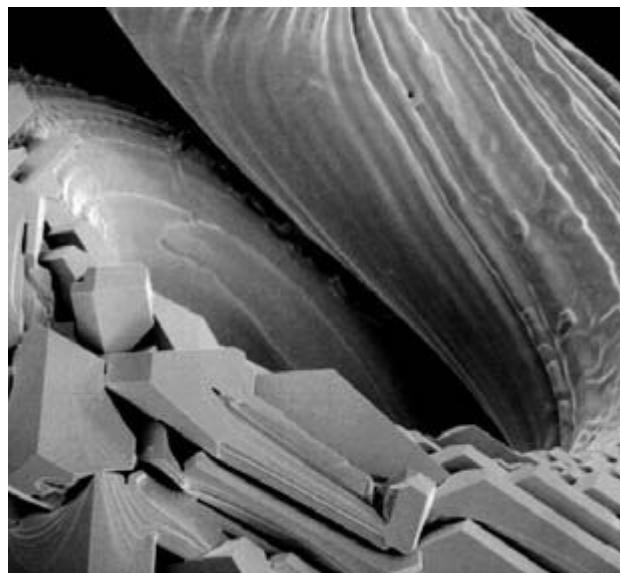
- ✓ Acoustic imaging:

- Obstetrics (ultrasonography):

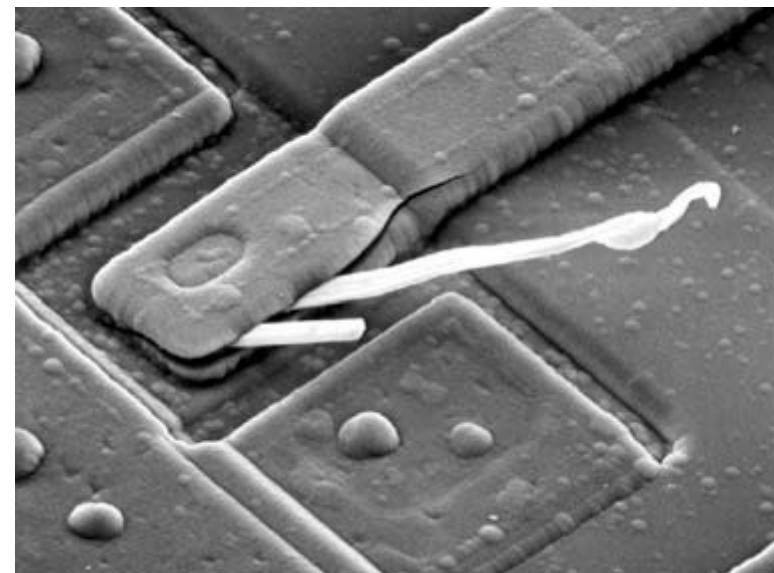


4. Examples of Fields that Use Digital Image Processing

- Examples in which Other Imaging Modalities Are Used
 - ✓ Electron microscopy (up to x10.000 magnification):



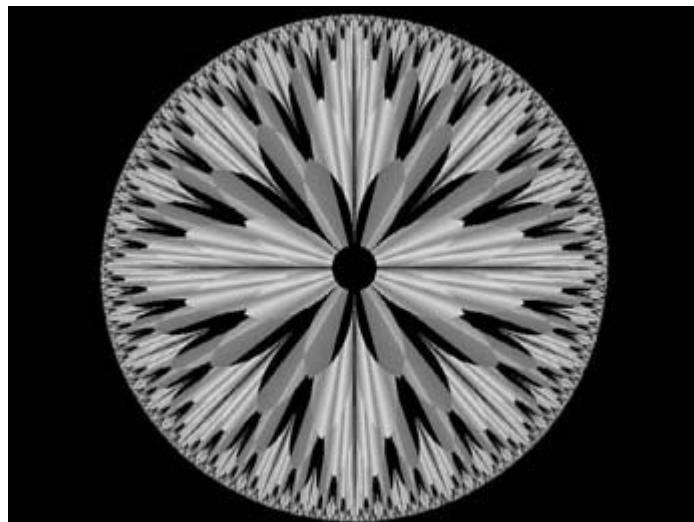
Tungsten filament
(x250)



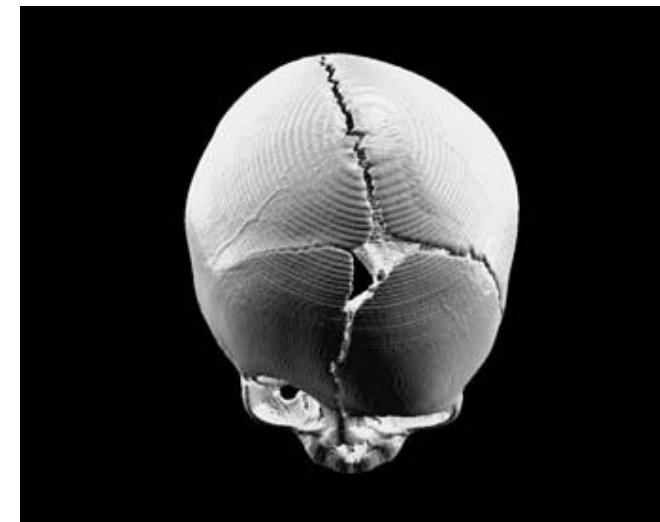
Integrated circuit
(x2500)

4. Examples of Fields that Use Digital Image Processing

- Examples in which Other Imaging Modalities Are Used
 - ✓ Computer-generated images:



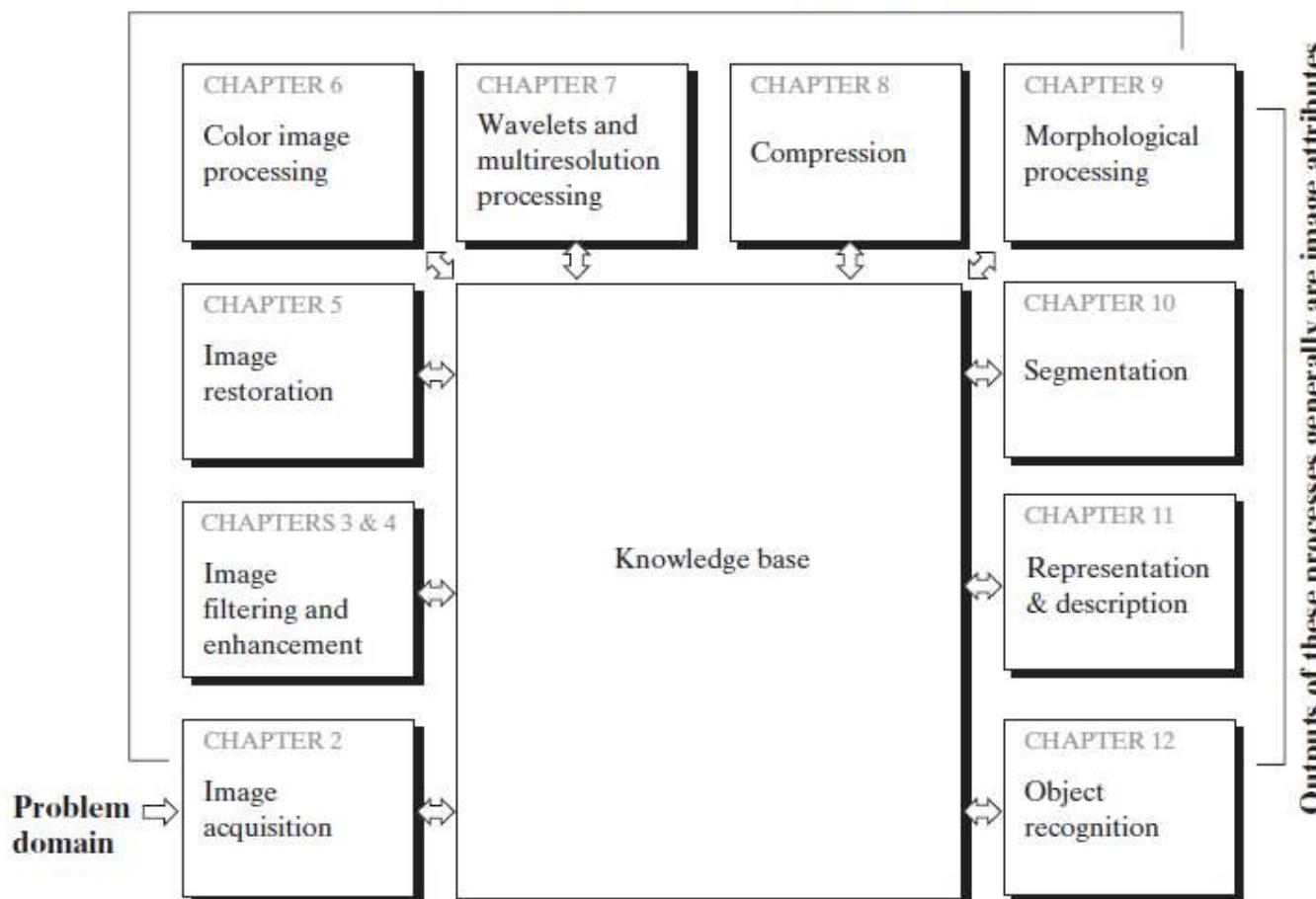
Fractals



3-D modeling

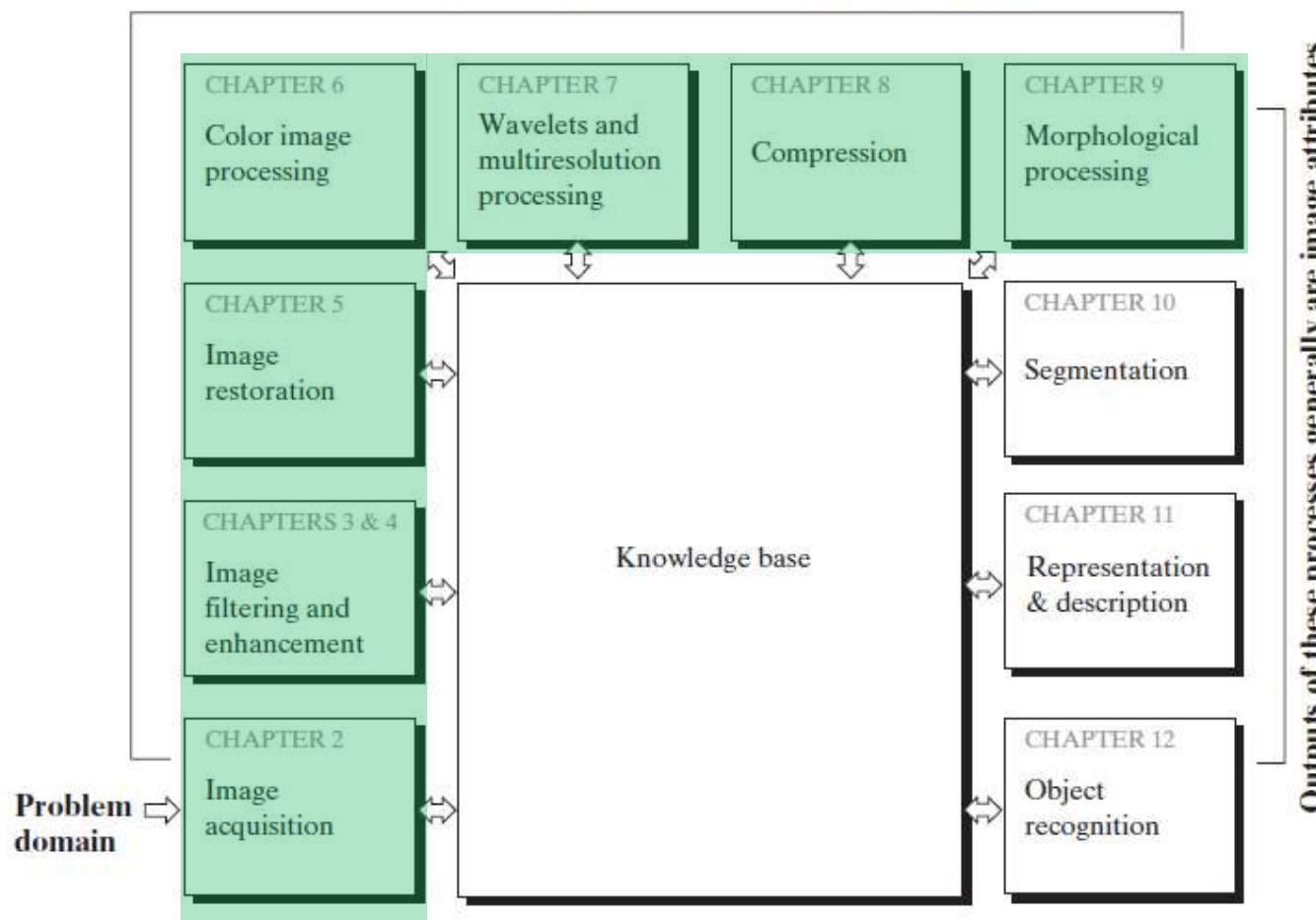
5. Fundamental Steps in Digital Image Processing

Outputs of these processes generally are images



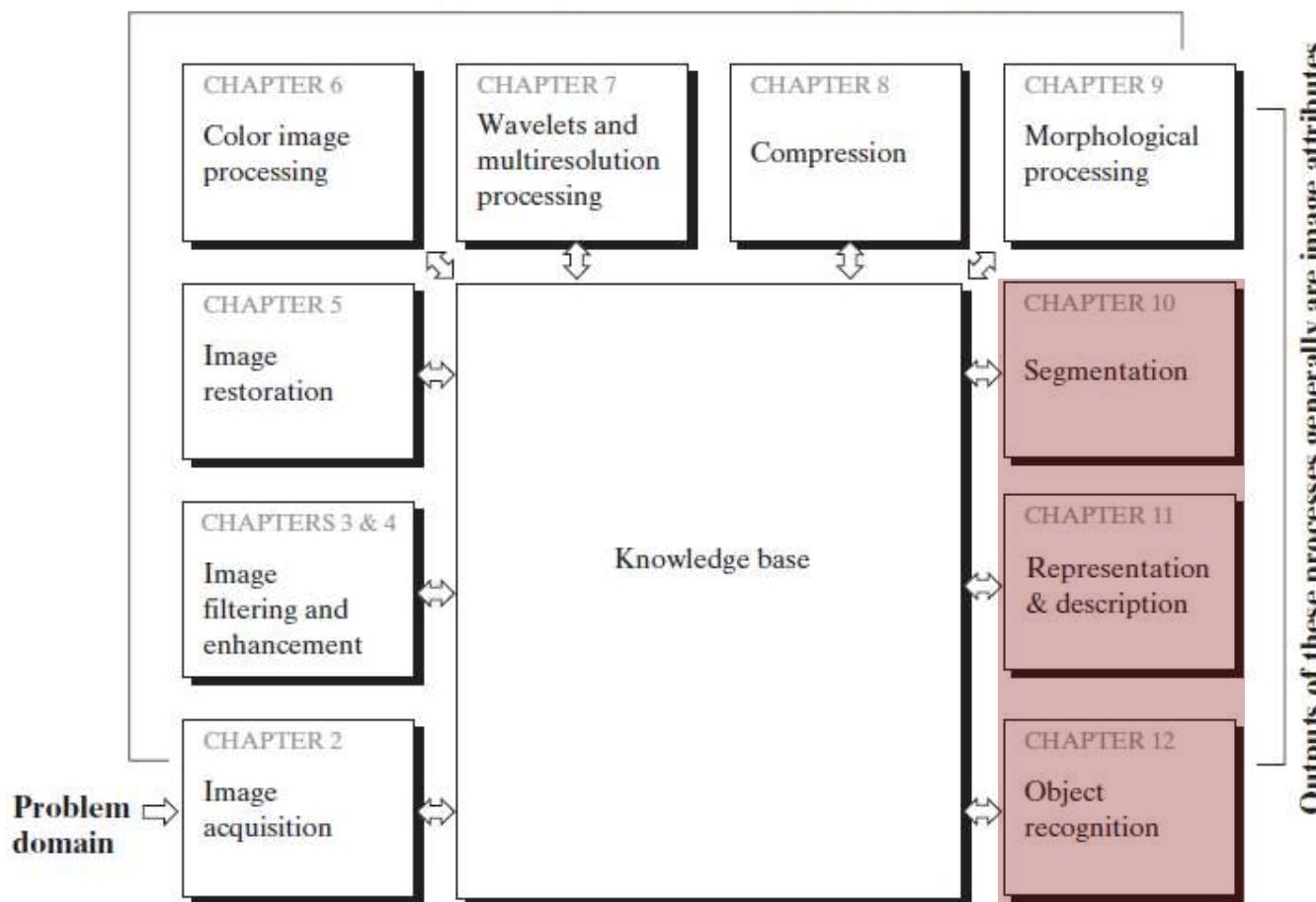
5. Fundamental Steps in Digital Image Processing

Outputs of these processes generally are images



5. Fundamental Steps in Digital Image Processing

Outputs of these processes generally are images



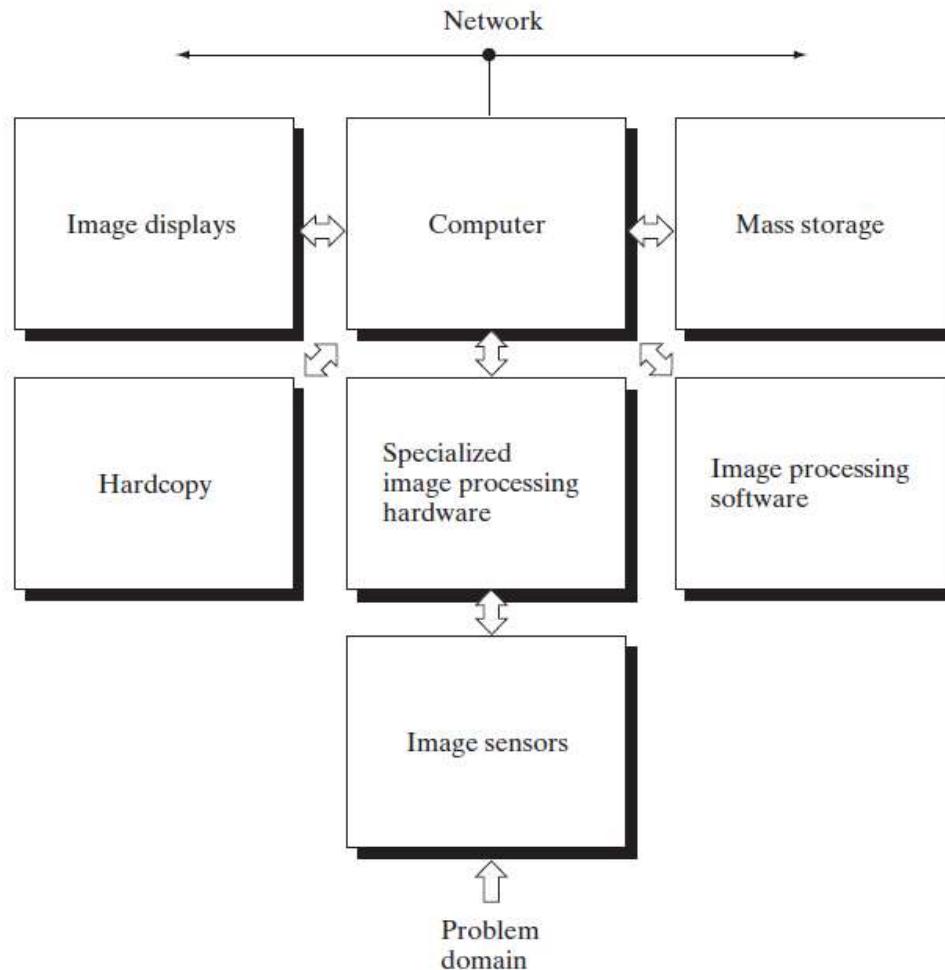
5. Fundamental Steps in Digital Image Processing

- ***Image acquisition***: could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves preprocessing.
- ***Image enhancement***: brings out detail that is obscured, or simply to highlight certain features of interest (subjective improvement) in an image.
- ***Image restoration***: deals with objectively improving the appearance of an image.
- ***Color image processing***: covers basic color processing in a digital domain.
- ***Wavelets***: are the foundation for representing images in various degrees of resolution. Used for image data compression and pyramidal representation.

5. Fundamental Steps in Digital Image Processing

- **Compression**: deals with techniques for reducing the storage required to save an image, or the bandwidth required to transmit it.
- **Morphological processing**: deals with tools for extracting image components that are useful in the representation and description of shape.
- **Segmentation**: partitions an image into its constituent parts or objects.
- **Representation and description**: transforms raw data into a form suitable for subsequent processing.
- **Recognition**: is the process that assigns a label (e.g., "vehicle") to an object based on its descriptors.

6. Components of an Image Processing System



6. Components of an Image Processing System

- ***Image sensors***: physical device that is sensitive to the energy radiated by the object we wish to image.
- ***Specialized hardware***: digitizer plus hardware that performs other primitive operations.
- ***Computer***: general-purpose computer.
- ***Software***: consists of specialized modules that perform specific tasks.
- ***Mass storage capability***: when dealing with thousands, or even millions, of images, providing adequate storage in an image processing system can be a challenge.
- ***Image displays***: are mainly color monitors.

6. Components of an Image Processing System

- **Hardcopy:** devices for recording images (include laser printers, film cameras, heat-sensitive devices, inkjet units, and digital units, such as CD-ROM disks).
- **Networking:** image transmission.

Suggested reading: Gonzales & Woods DIP - Chapter 1

Sample Book Material

http://www.imageprocessingplace.com/DIP-3E/dip3e_sample_book_material.htm

Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 02

Digital Image Fundamentals

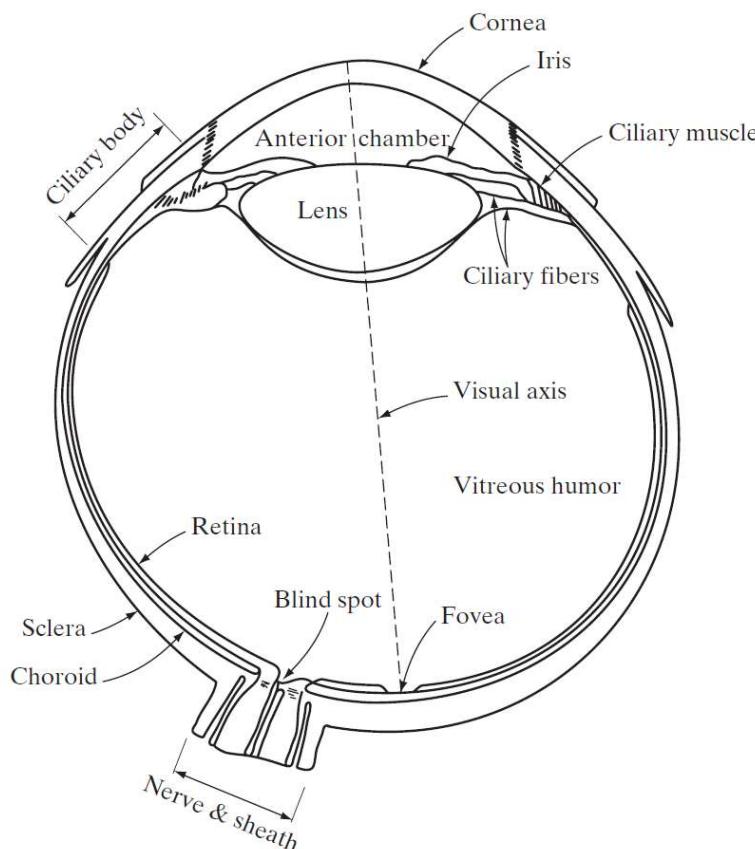
1. Elements of Visual Perception

- Digital image processing field is built on a foundation of mathematical and probabilistic formulations.
- Human intuition and analysis play a central role in the choice of one technique versus another → Subjective, visual judgments.



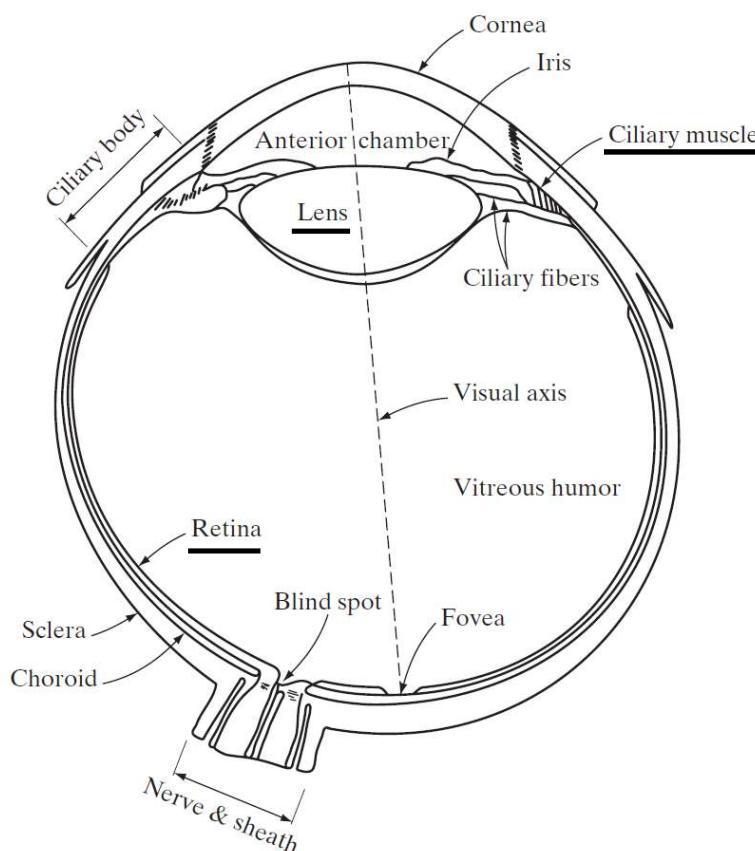
1.1 Structure of the Human Eye

- The **eye** is nearly a sphere, with an average diameter of approximately 20 mm.



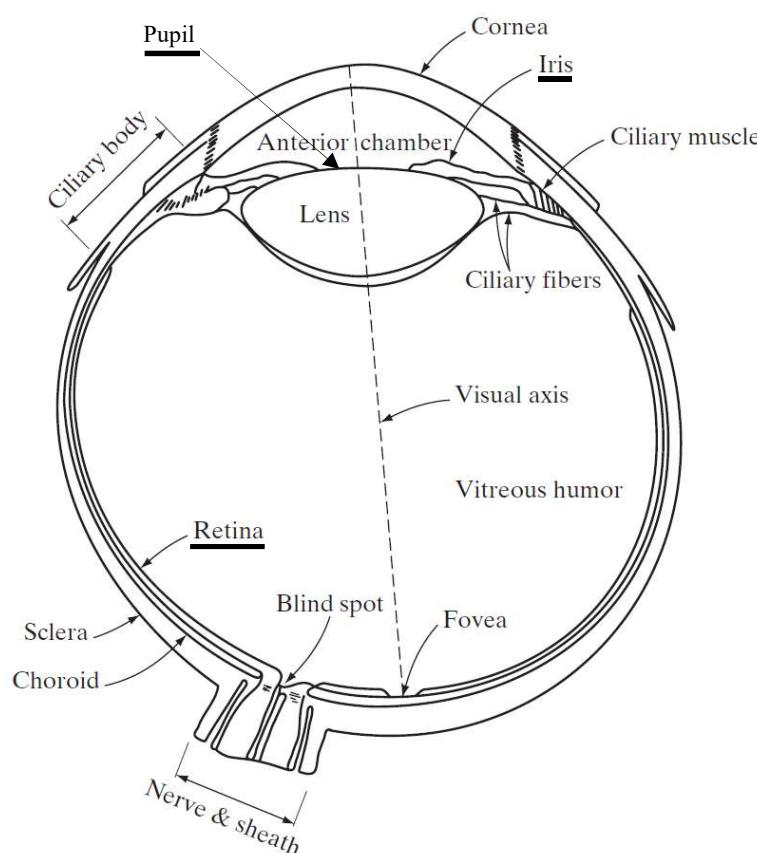
1.1 Structure of the Human Eye

- The **ciliary muscle** changes the shape of the **lens** in order to focus light on the **retina**.



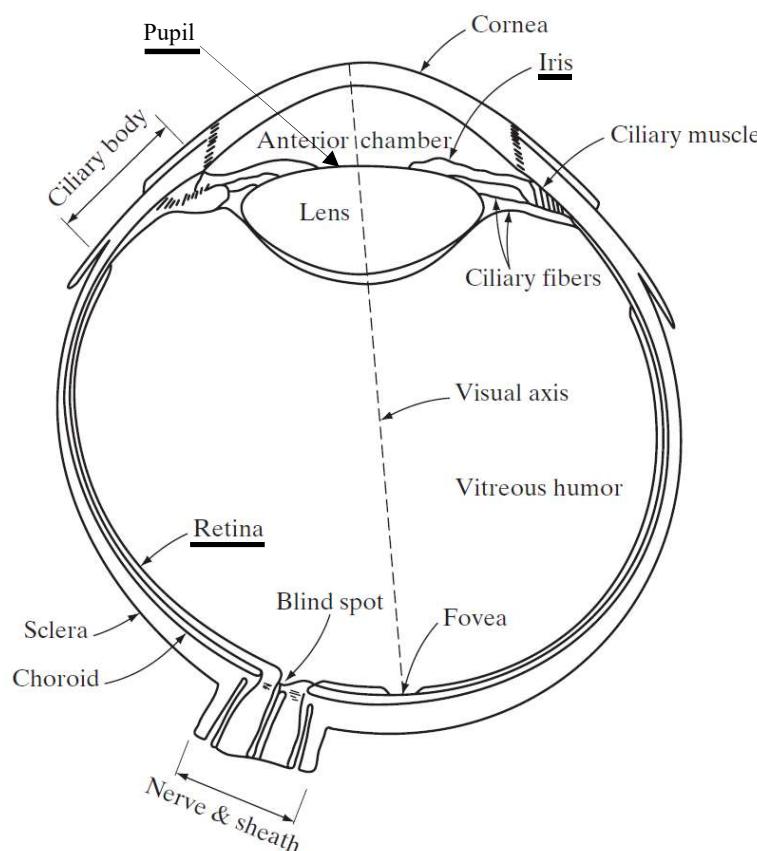
1.1 Structure of the Human Eye

- The **iris** is responsible for controlling the diameter and size of the **pupil** and thus the amount of light reaching the **retina**.



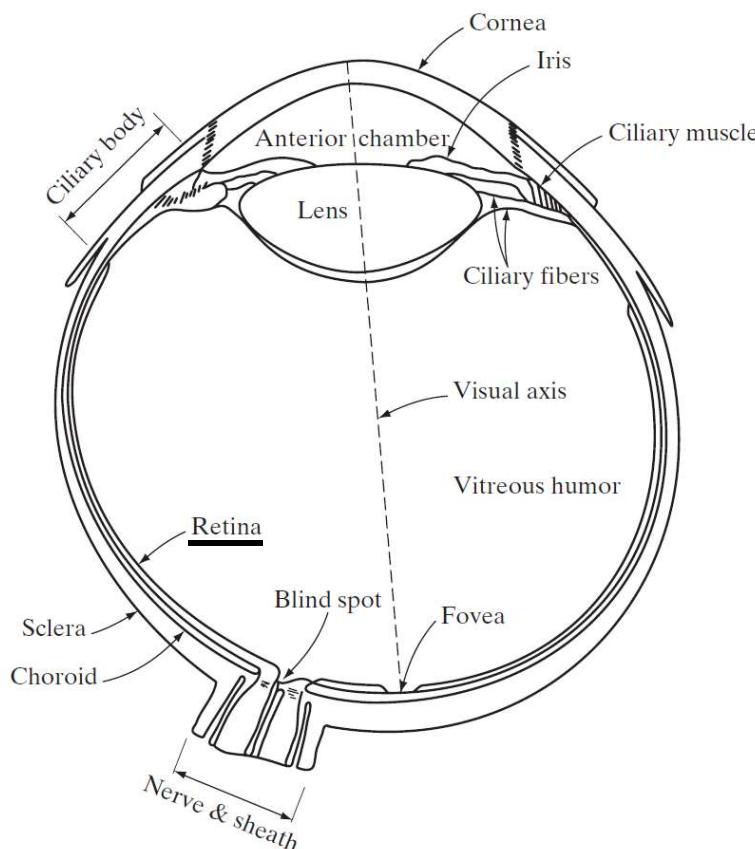
1.1 Structure of the Human Eye

- The **pupil** is a hole located in the centre of the **iris** that allows light to strike the **retina**.



1.1 Structure of the Human Eye

- The **retina** is the innermost membrane of the eye. Light from an object outside the eye is imaged on the retina.

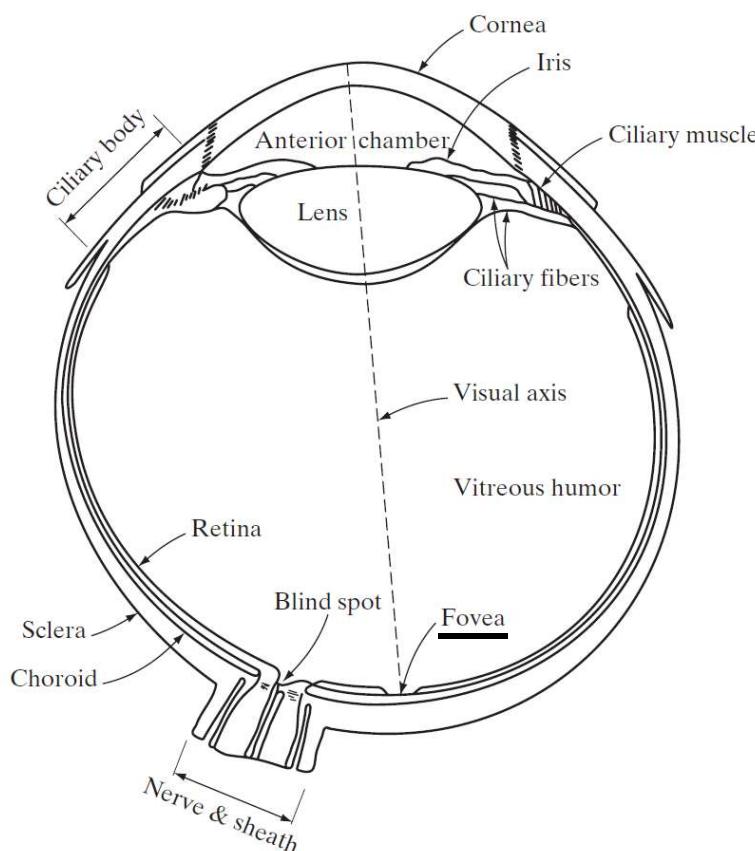


1.1 Structure of the Human Eye

- Vision is afforded by the distribution of discrete light receptors over the surface of the retina.
 - There are two classes of receptors: **cones** and **rods**.
- The cones in each eye number between 6 and 7 million. They are located primarily in the central portion of the retina, called the **fovea**.
 - Humans can resolve **fine details** with these cones.
 - Highly sensitive to color.
 - Cone vision is called **photopic** or **bright-light vision**.
- We can view the fovea as a square sensor array of size 1.5 mm x 1.5 mm.

1.1 Structure of the Human Eye

- The cones in each eye number between 6 and 7 million. They are located primarily in the **fovea**.

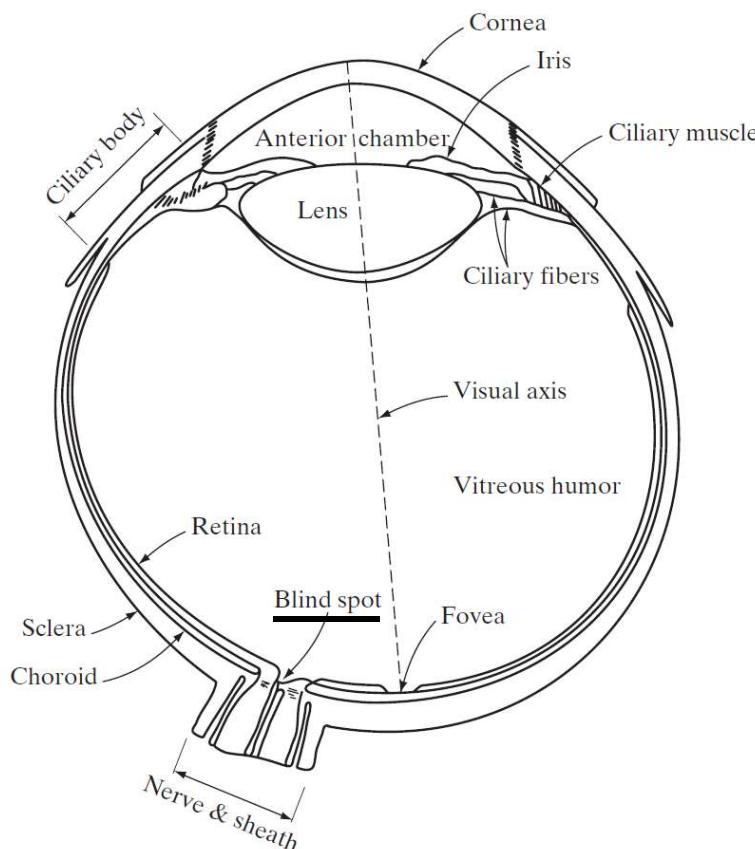


1.1 Structure of the Human Eye

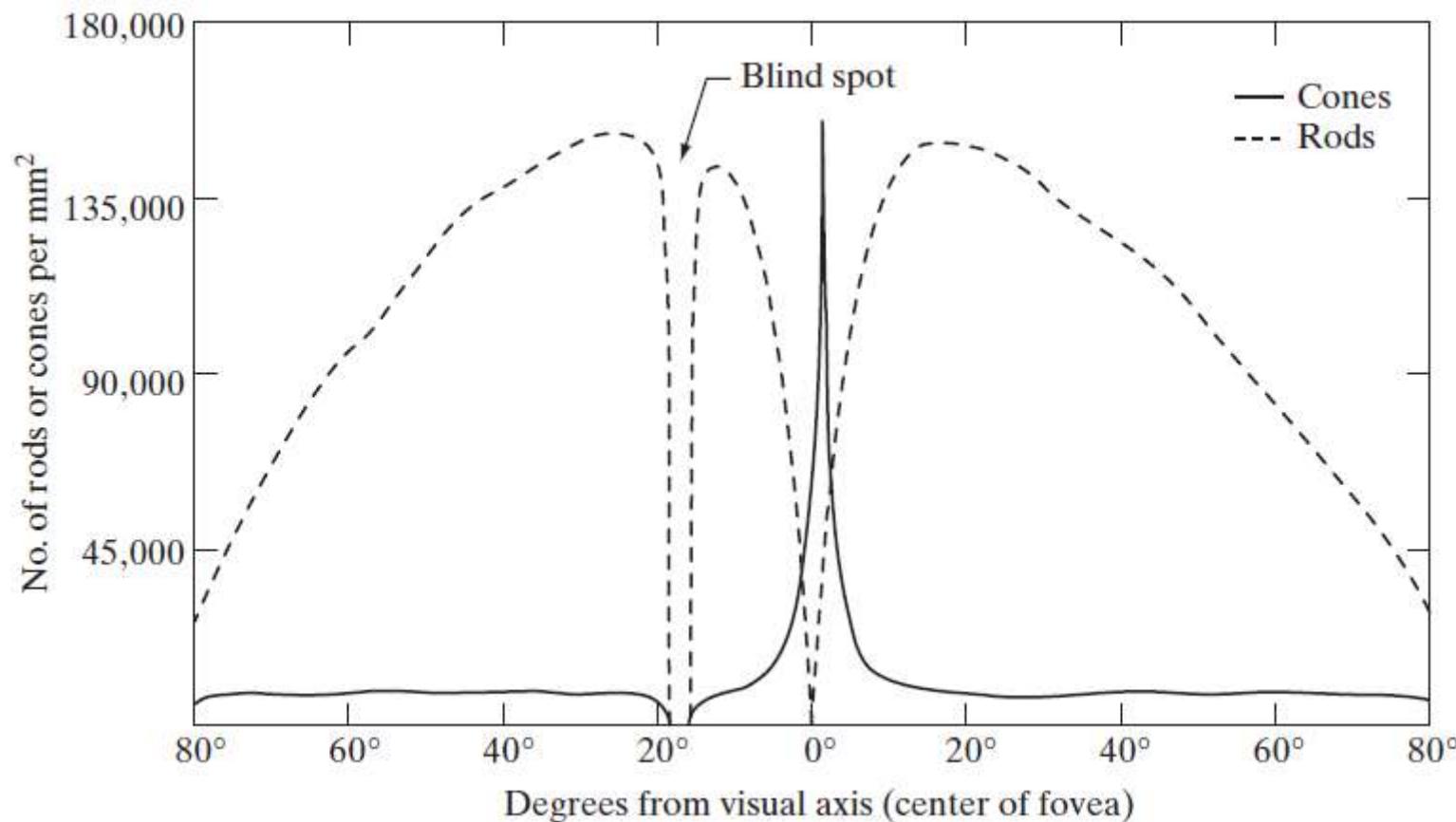
- The number of rods is much larger: 75 to 150 million distributed over the retinal surface.
 - Rods serve to give a general, **overall picture** of the field of view.
 - They are not involved in color vision.
 - Sensitive to low levels of illumination (**scotopic** or **dim-light vision**).
 - For example, objects that appear brightly colored in daylight when seen by moonlight appear as colorless forms because only the rods are stimulated
- **Blind spot:** region of emergence of the optic nerve from the eye (the absence of receptors).

1.1 Structure of the Human Eye

- **Blind spot:** region of emergence of the optic nerve from the eye (the absence of receptors).

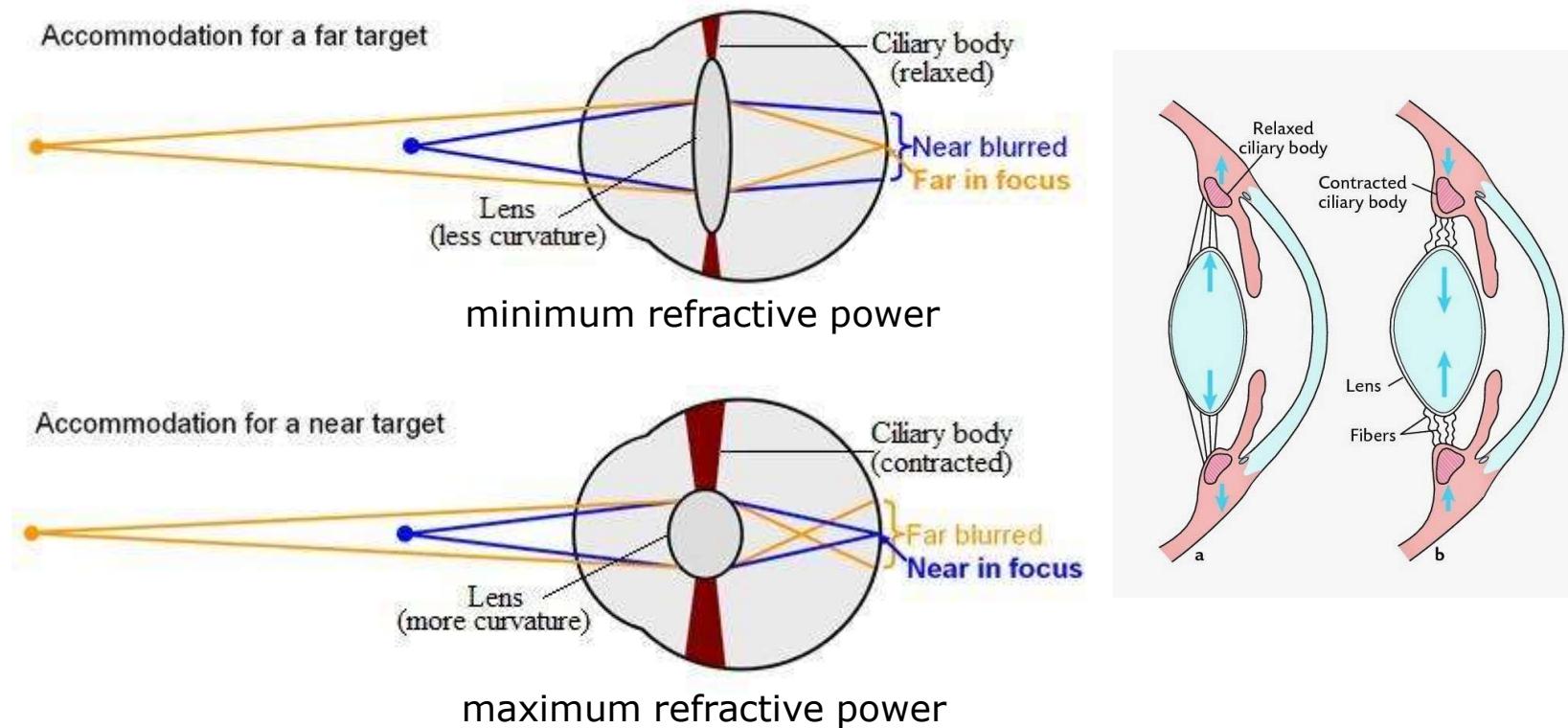


1.1 Structure of the Human Eye



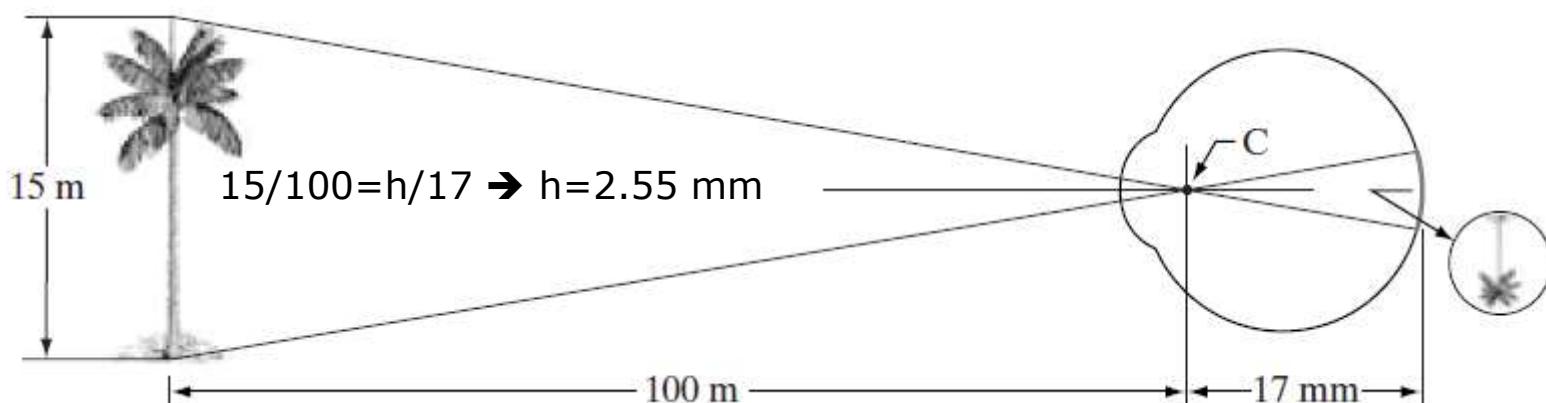
1.2 Image Formation in the Eye

- **Focal length:** distance between the center of the lens and the retina. Varies from 14 mm (maximum refractive power) to about 17 mm (minimum refractive power).



1.2 Image Formation in the Eye

- **Focal length:** distance between the center of the lens and the retina. Varies from 14 mm to about 17 mm.

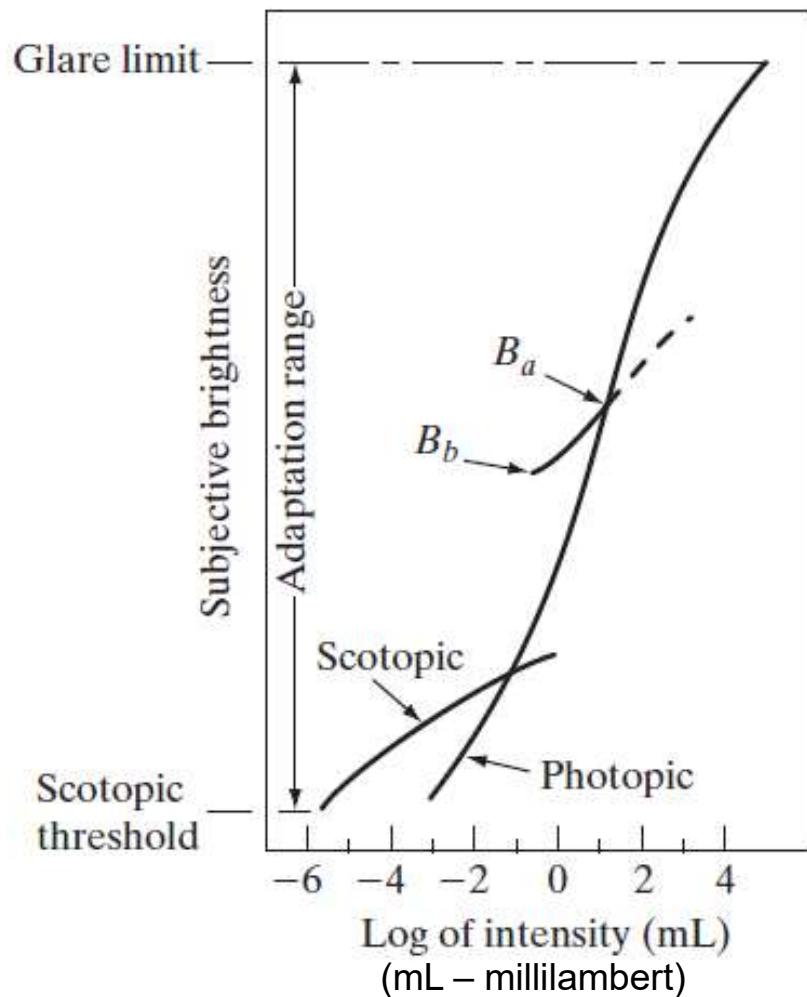


- The retinal image is reflected primarily in the area of the fovea.
- Perception then takes place by the relative excitation of light receptors, which transform radiant energy into electrical impulses that are ultimately decoded by the brain.

1.3 Brightness Adaptation and Discrimination

- The range of light intensity levels to which the human visual system can adapt is enormous:
 - on the order of 10^{10} from the *scotopic* threshold to the *glare* limit.
- Experimental evidence indicates that subjective brightness is a logarithmic function of the light intensity incident on the eye.

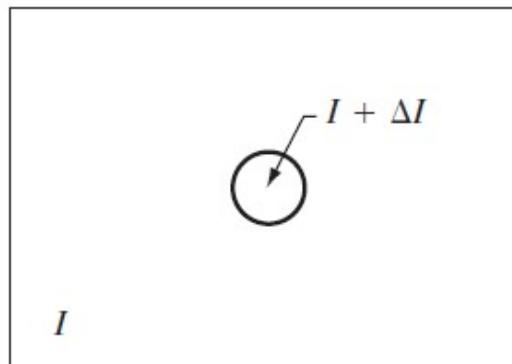
1.3 Brightness Adaptation and Discrimination



- The visual system cannot operate over such a range simultaneously.
- Rather, it accomplishes this large variation by changes in its overall sensitivity, a phenomenon known as **brightness adaptation**.
- For any given set of conditions, the current sensitivity level of the visual system is called the **brightness adaptation level** (e.g. B_a).
- In this example, stimuli below B_b are perceived as indistinguishable blacks.

1.3 Brightness Adaptation and Discrimination

- The ability of the eye to discriminate between changes in light intensity at any specific adaptation level is also of considerable interest.
- A classic experiment: *background + flashes*.



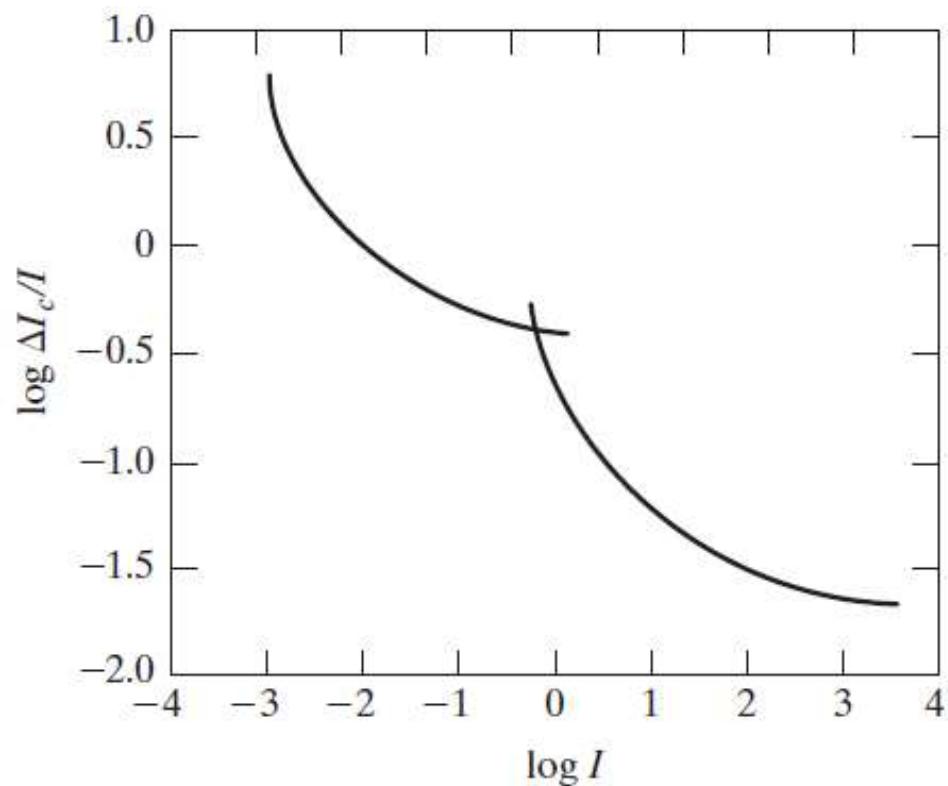
- *Weber ratio*: $\Delta I_c/I \rightarrow \Delta I_c$ is the increment of illumination discriminable 50% of trials with background illumination I . A small value means “good” brightness discrimination, a large value means “poor” brightness discrimination.

1.3 Brightness Adaptation and Discrimination

- Typical plot of Weber ratio as a function of intensity.

Brightness discrimination is poor (the Weber ratio is large) at low levels of illumination. Vision is carried out by activity of the rods

Brightness discrimination improves (the Weber gets smaller) at high levels of illumination. Vision is carried out by activity of cones.



1.3 Brightness Adaptation and Discrimination

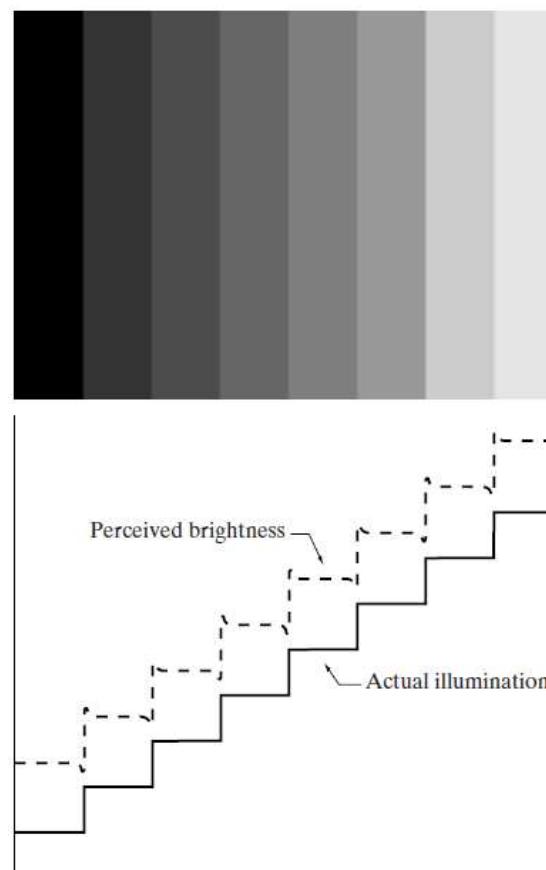
- Another experiment: background illumination is held constant and the intensity of the other source is now allowed to vary incrementally from never being perceived to always being perceived.
 - The typical observer can discern a total of one to two dozen different intensity changes.
- Roughly, this result is related to the number of different intensities a person can see at any one point in a monochrome image.
- This result does not mean that an image can be represented by such a small number of intensity
 - As the eye roams about the image, the average background changes, thus allowing a *different* set of incremental changes to be detected

1.3 Brightness Adaptation and Discrimination

- However, perceived brightness is not a simple function of intensity:
 - Mach Bands.
 - Simultaneous contrast.
 - Other examples.

1.3 Brightness Adaptation and Discrimination

- Mach bands: undershoot or overshoot around the boundary of regions of different intensities



1.3 Brightness Adaptation and Discrimination

- Mach bands: undershoot or overshoot around the boundary of regions of different intensities



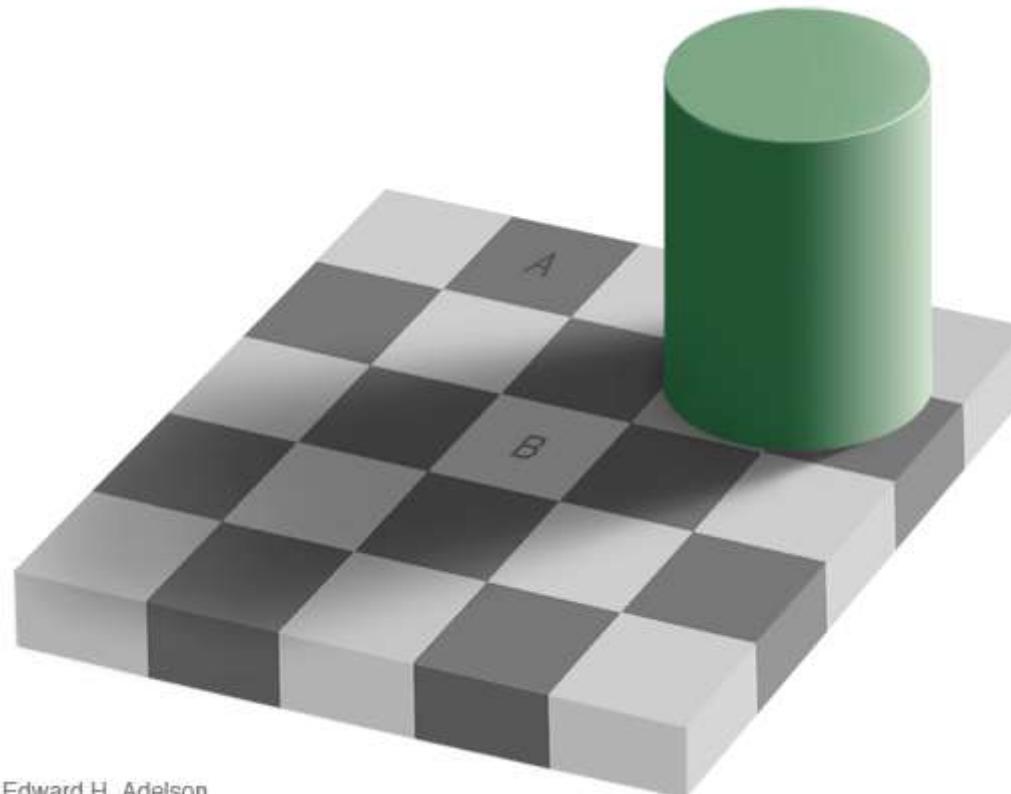
1.3 Brightness Adaptation and Discrimination

- Simultaneous contrast: perceived brightness does not depend simply on its intensity.



1.3 Brightness Adaptation and Discrimination

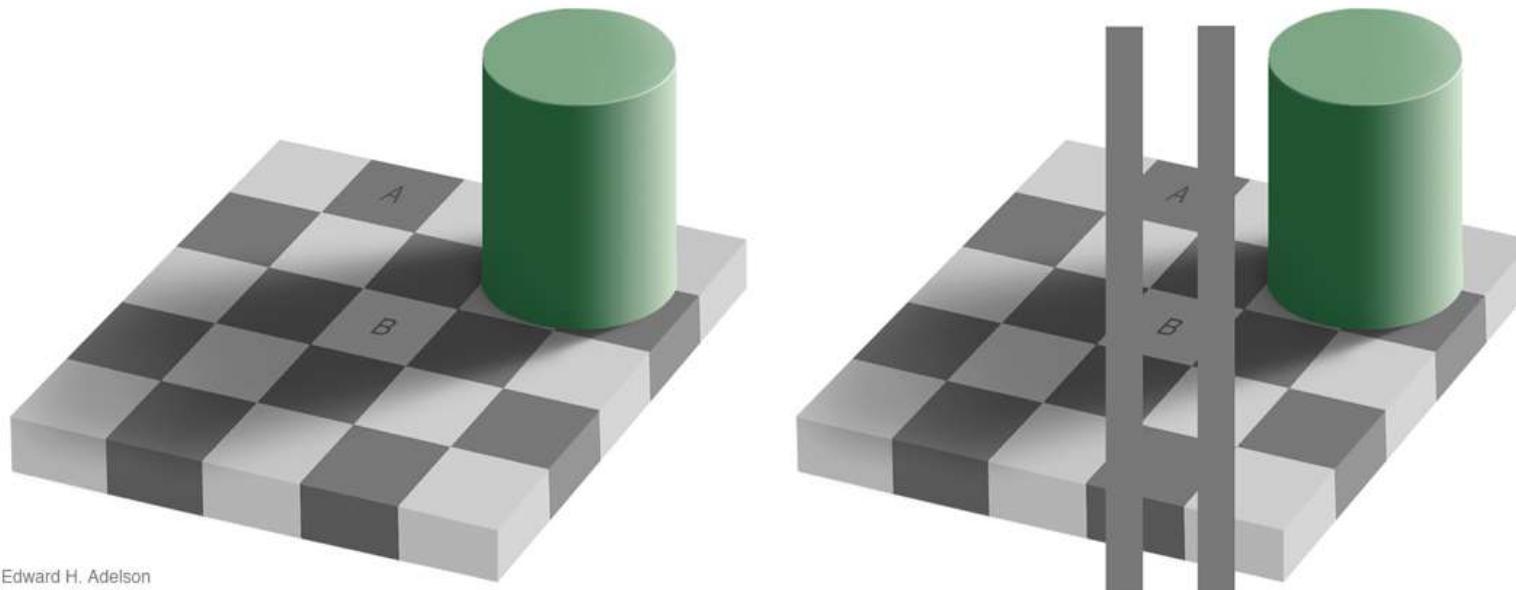
- Intensities of A and B are the same?



Edward H. Adelson

1.3 Brightness Adaptation and Discrimination

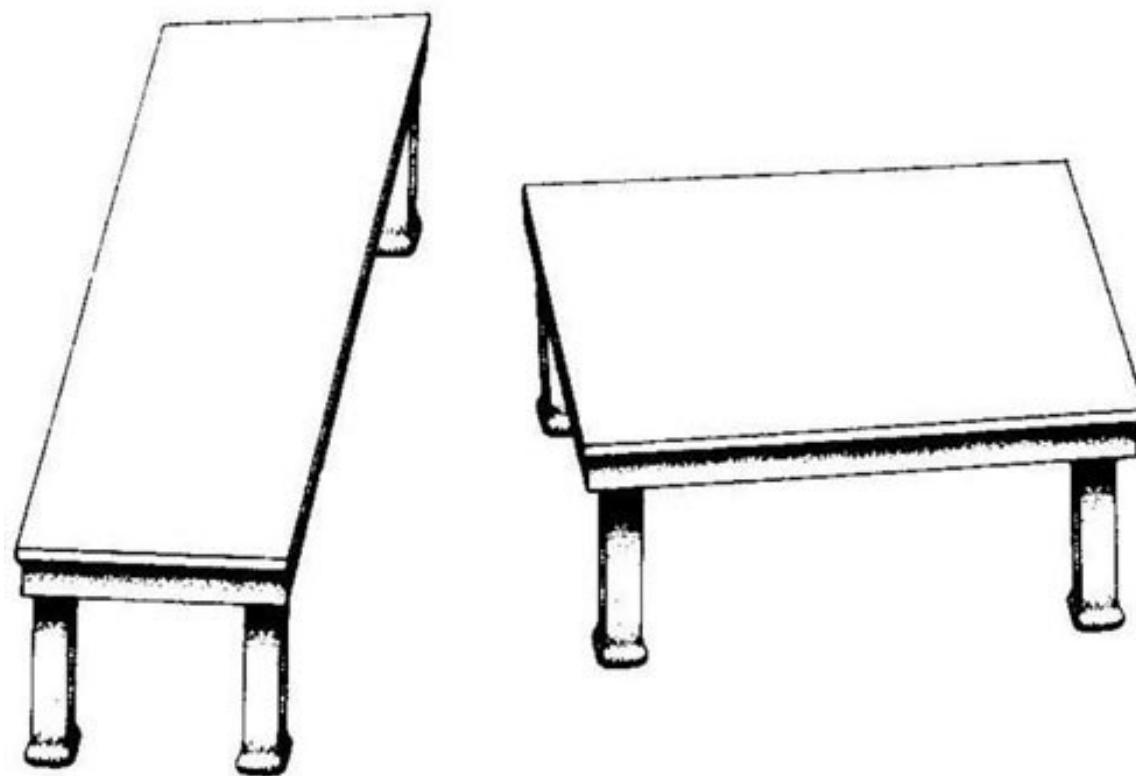
- Intensities of A and B are the same?



http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

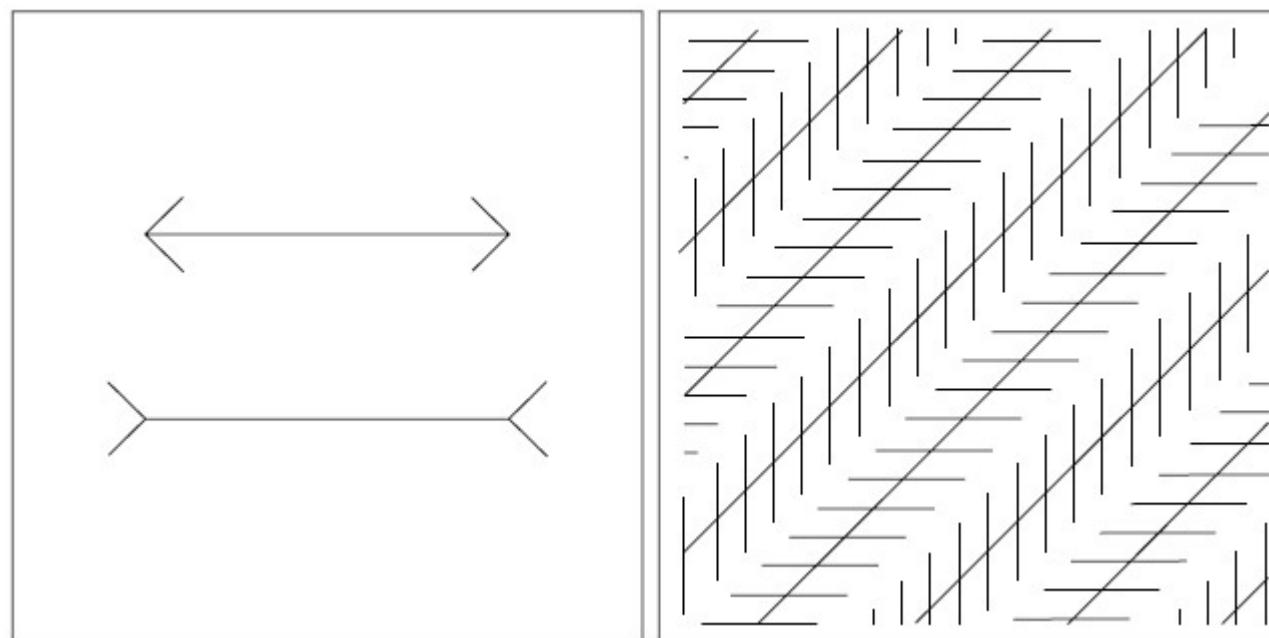
1.4 Optical Illusions

- Turning the Tables: the tabletop on the left is identical in shape and size to the one on the right.



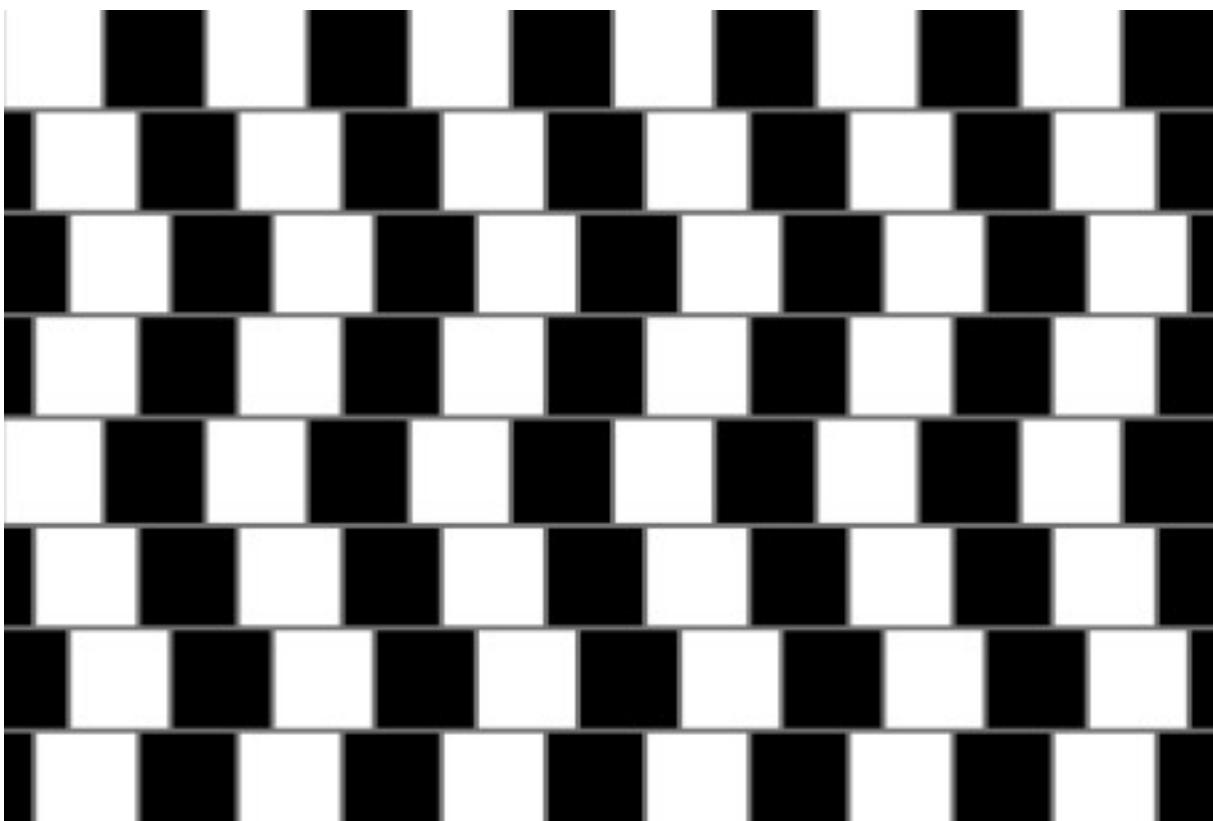
1.4 Optical Illusions

- Left: The two horizontal line segments are of the same length, but one appears shorter than the other; Right: all lines that are oriented at 45° are equidistant and parallel.



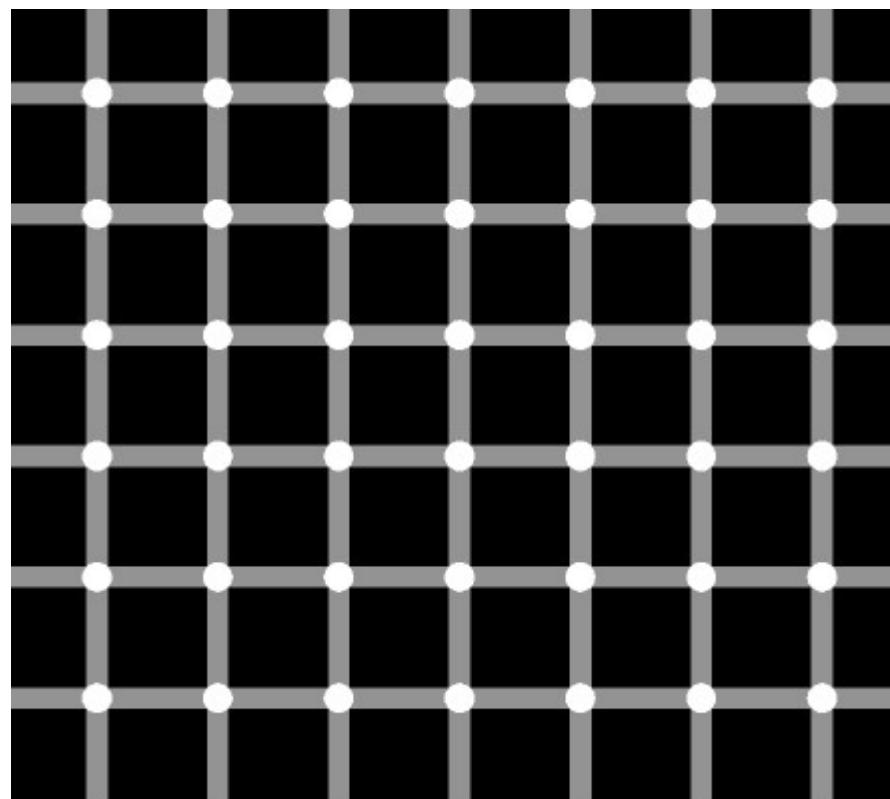
1.4 Optical Illusions

- Lines are parallel.



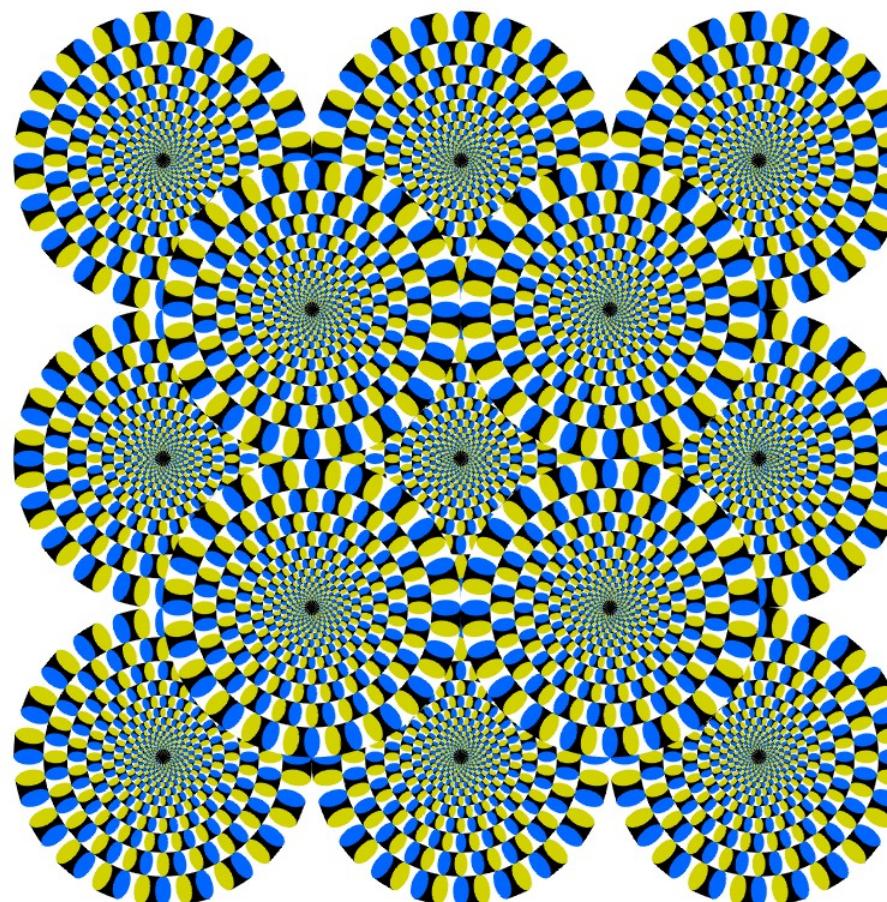
1.4 Optical Illusions

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

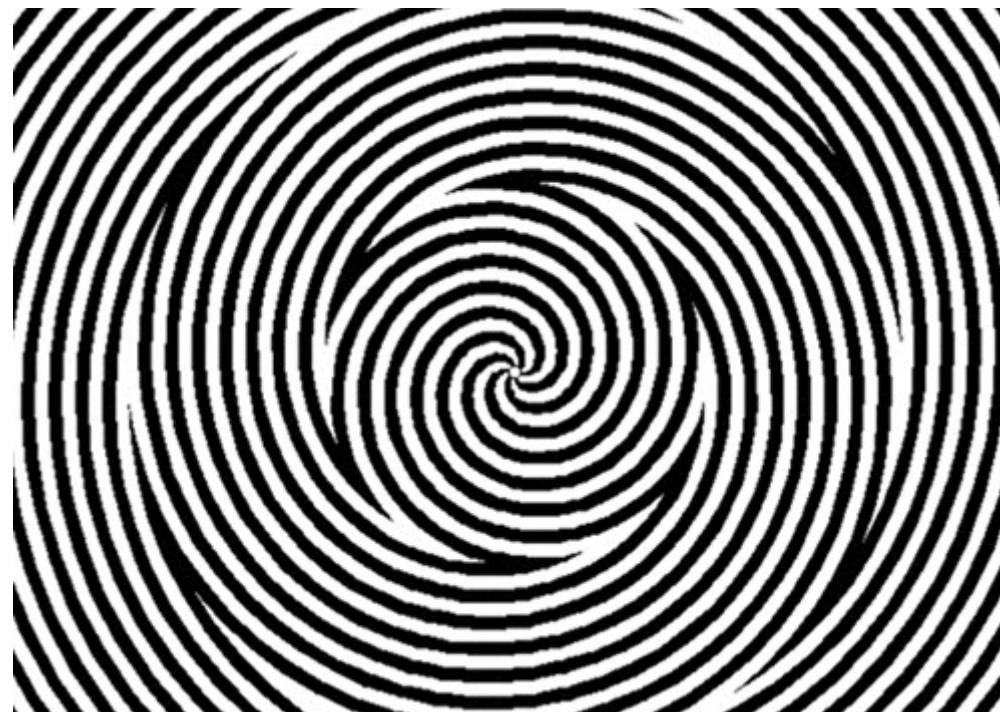


1.4 Optical Illusions

- Rotation snakes.



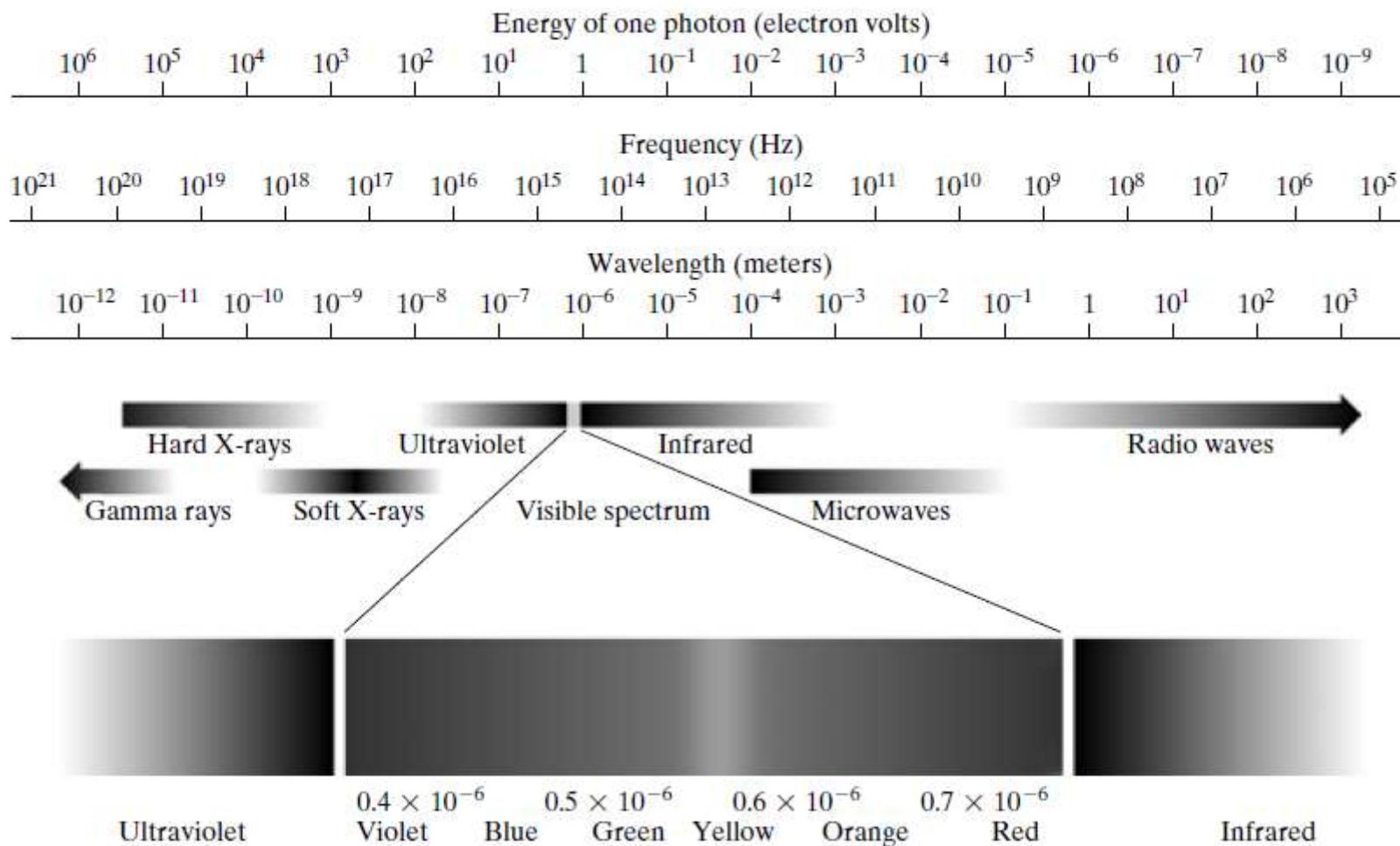
1.4 Optical Illusions



1.4 Optical Illusions



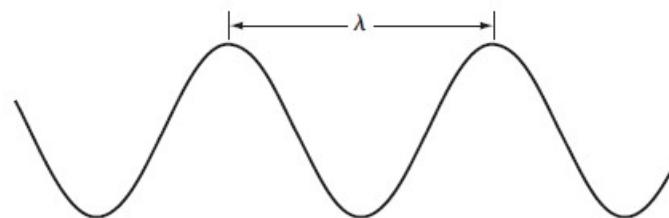
2. Light and the Electromagnetic Spectrum



2. Light and the Electromagnetic Spectrum

- The electromagnetic spectrum can be expressed in terms of wavelength, frequency, or energy.
- Wavelength λ (m) and frequency ν (Hz) are related by the expression

$$\lambda = \frac{c}{\nu}$$



where c is the speed of light (2.988×10^8 m/s).

- The energy (eV) is given by

$$E = h\nu$$

where h ($4,136 \times 10^{-15}$ eV) is Plank's constant.

2. Light and the Electromagnetic Spectrum

- Electromagnetic waves can be visualized as propagating sinusoidal waves with wavelength λ .
- Or they can be thought of as a stream of massless particles, each traveling in a wavelike pattern and moving at the speed of light.
 - Each massless particle contains a certain amount (or bundle) of energy.
 - ✓ Each bundle of energy is called a photon.
- Energy is proportional to frequency.
 - The higher frequency the more energy per photon.

2. Light and the Electromagnetic Spectrum

- Light is a particular type of electromagnetic radiation that can be sensed by the human eye.

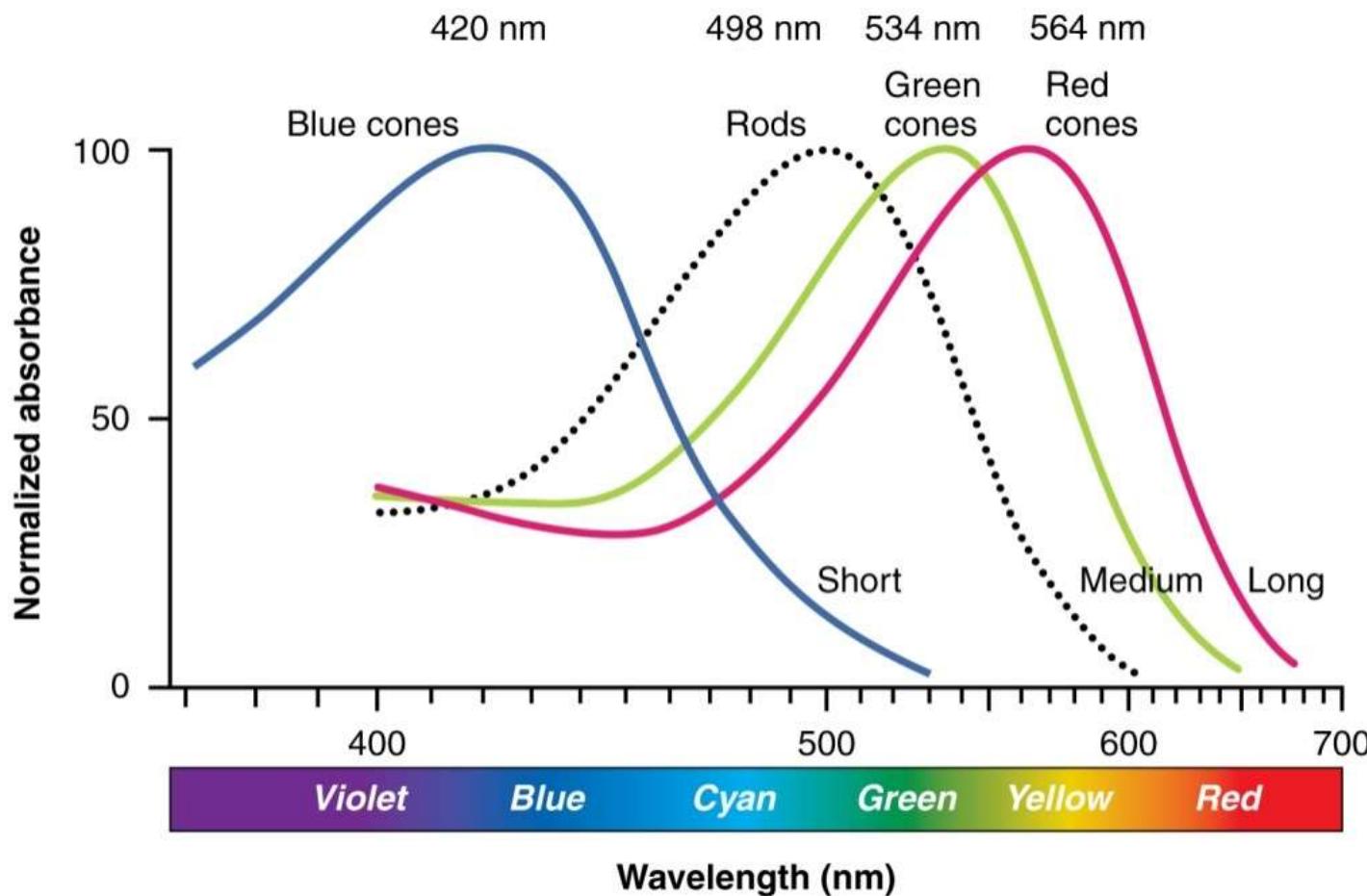


- **For convenience**, the color spectrum is divided into six broad regions: violet, blue, green, yellow, orange, and red.
 - The colors that humans perceive in an object are determined by the nature of the light reflected from the object.

2. Light and the Electromagnetic Spectrum

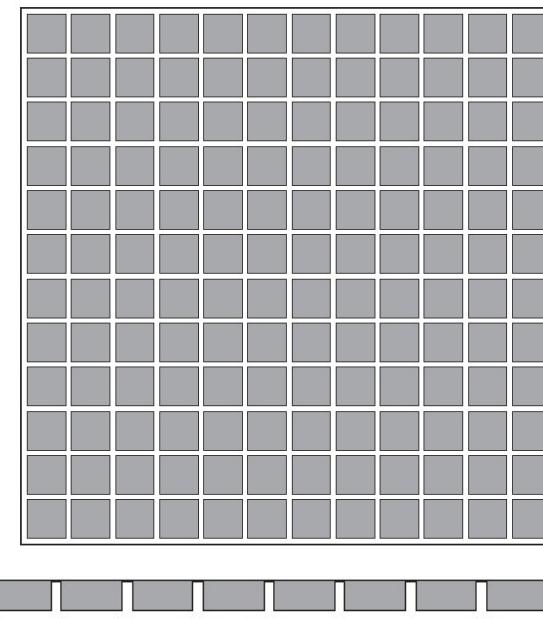
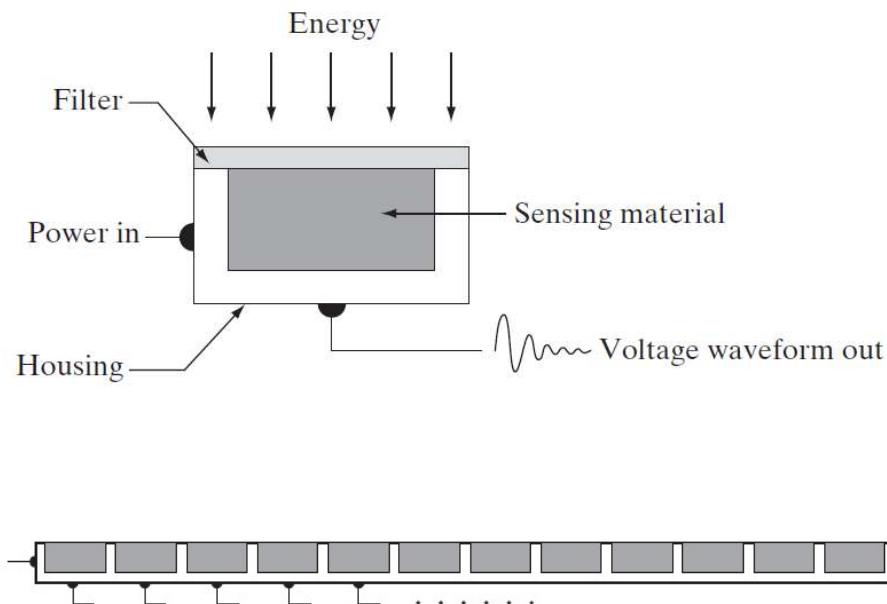
- Body that reflects light and is relatively balanced in all visible wavelengths appears white to the observer.
- However, a body that favors reflectance in a limited range of the visible spectrum exhibits some shades of color.
- Light that is void of color is called **achromatic** or **monochromatic light**.
- The term **gray level** generally is used to describe monochromatic intensity because it ranges from black, to grays, and finally to white.

2. Light and the Electromagnetic Spectrum



3. Image Sensing and Acquisition

- Incoming energy is transformed into a voltage. A digital quantity is obtained from each sensor by digitizing its response.
- Sensor arrangements used to transform illumination energy into digital images:



3.1 Image Acquisition Using a Single Sensor

- Single sensor:

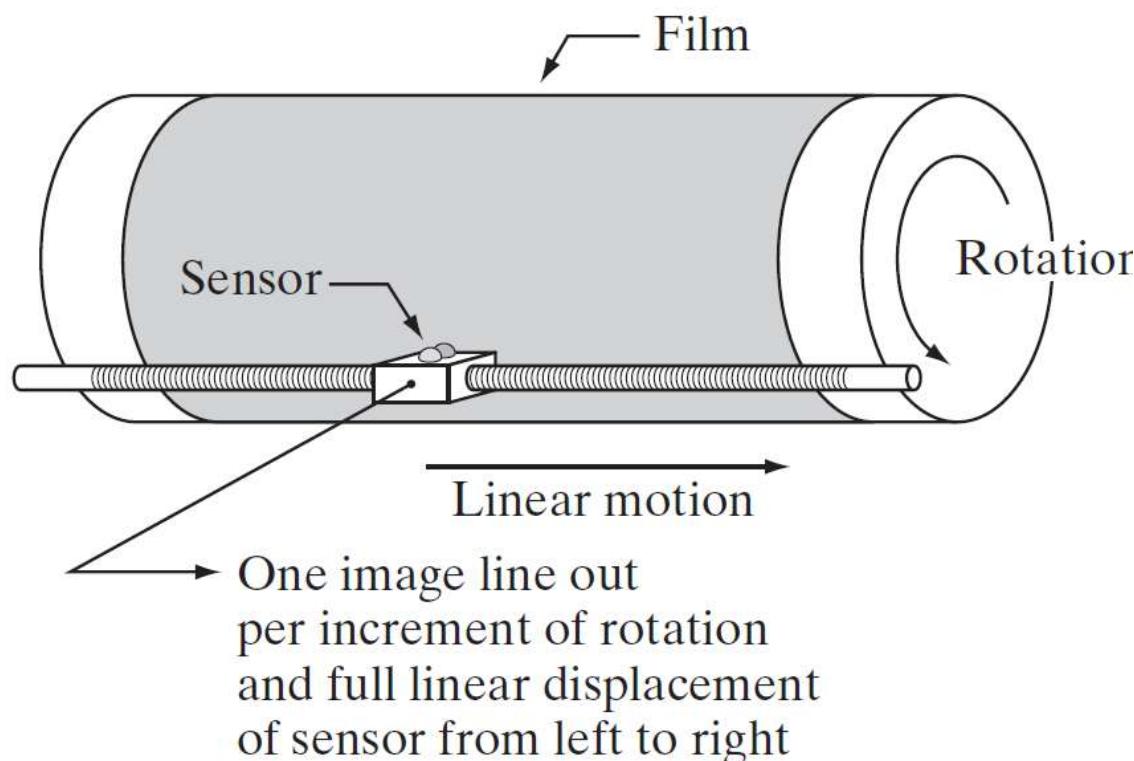


FIGURE 2.13
Combining a single sensor with motion to generate a 2-D image.

3.2 Image Acquisition Using Sensor Strips

- In-line arrangement of sensors:

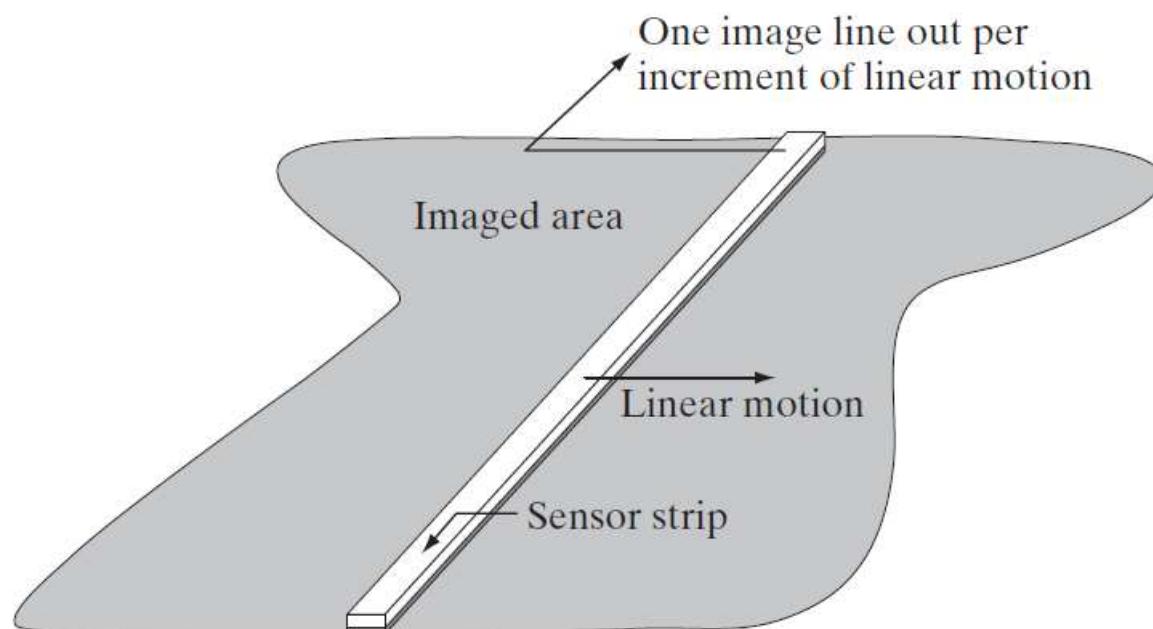


FIGURE 2.14 (a) Image acquisition using a linear sensor strip.

3.3 Image Acquisition Using Sensor Arrays

- Sensors arranged in the form of a 2-D array.

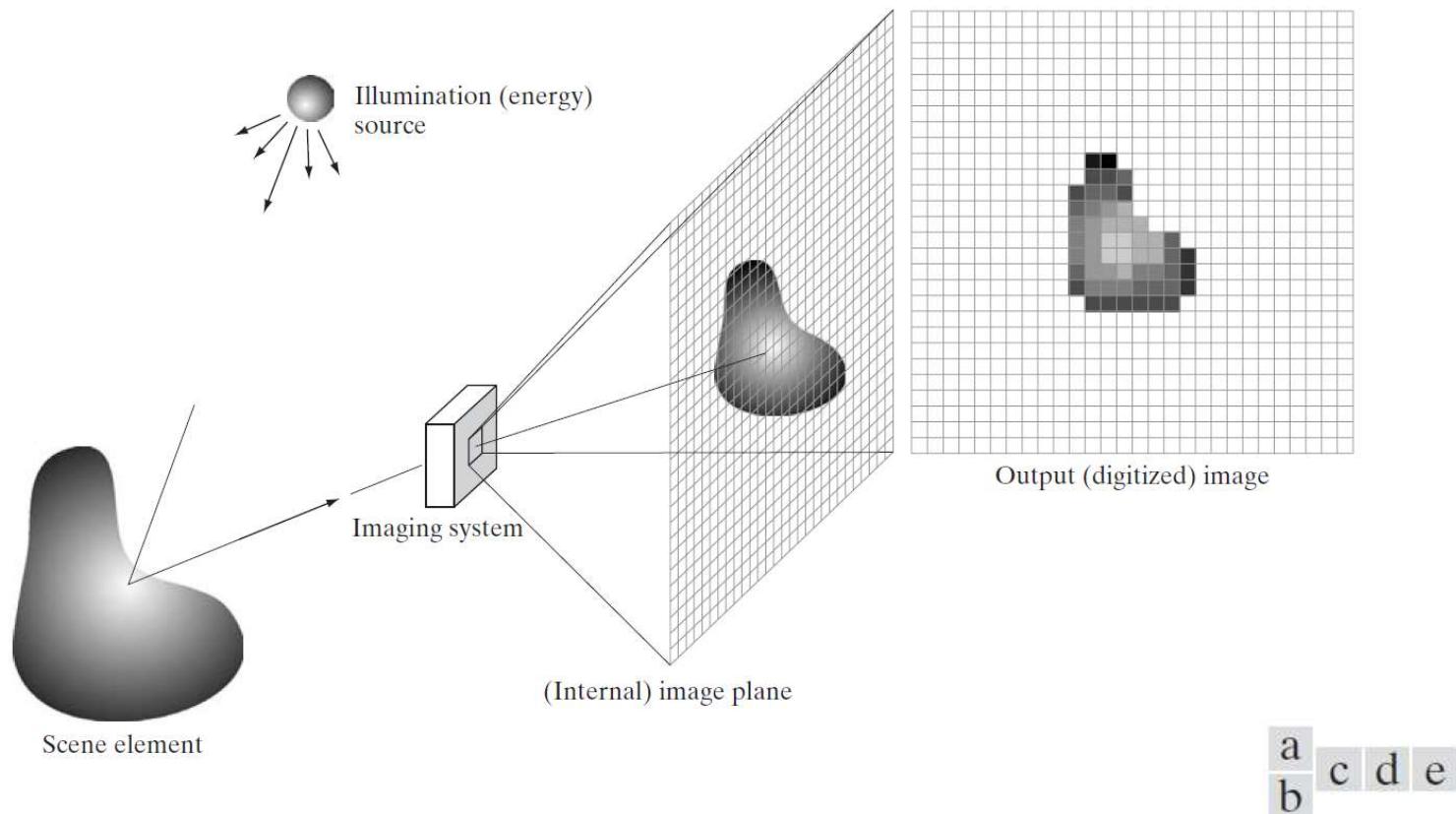


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

3.4 A Simple Image Formation Model

- As introduced before, we denote images by two dimensional functions of the form $f(x,y)$.
- The function $f(x,y)$ may be characterized by two components: (1) the amount of source **illumination incident** on the scene, and (2) the amount of **illumination reflected** by the objects in the scene.
- These are called the illumination and reflectance components and are denoted by $i(x,y)$ and $r(x,y)$.

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

$$0 < i(x, y) < \infty \text{ (Im/m}^2\text{)}$$

$$0 < r(x, y) < 1.$$

3.4 A Simple Image Formation Model

- Let the intensity (gray level) of a monochrome image at any coordinates (x_o, y_o) be denoted by

$$l = f(x_o, y_o)$$

- The interval $[L_{\min}, L_{\max}]$ is called the gray scale.

$$L_{\min} \leq l \leq L_{\max}$$



- Common practice is to shift this interval numerically to the $[0, L-1]$ interval where 0 is considered black and $L-1$ is considered white on the gray scale.

4. Image Sampling and Quantization

- To create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: sampling and quantization.
- An image may be continuous with respect to the x- and y-coordinates, and also in amplitude.
- To convert it to digital form, we have to sample the function in both coordinates and in amplitude.
 - Digitizing the **coordinate values** is called **sampling**.
 - Digitizing the **amplitude values** is called **quantization**.

4. Image Sampling and Quantization

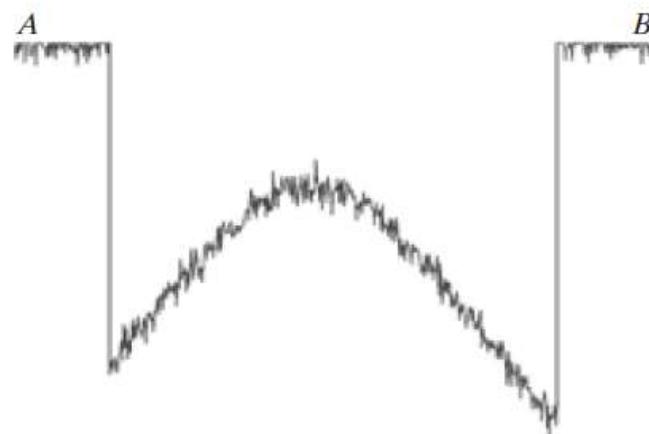
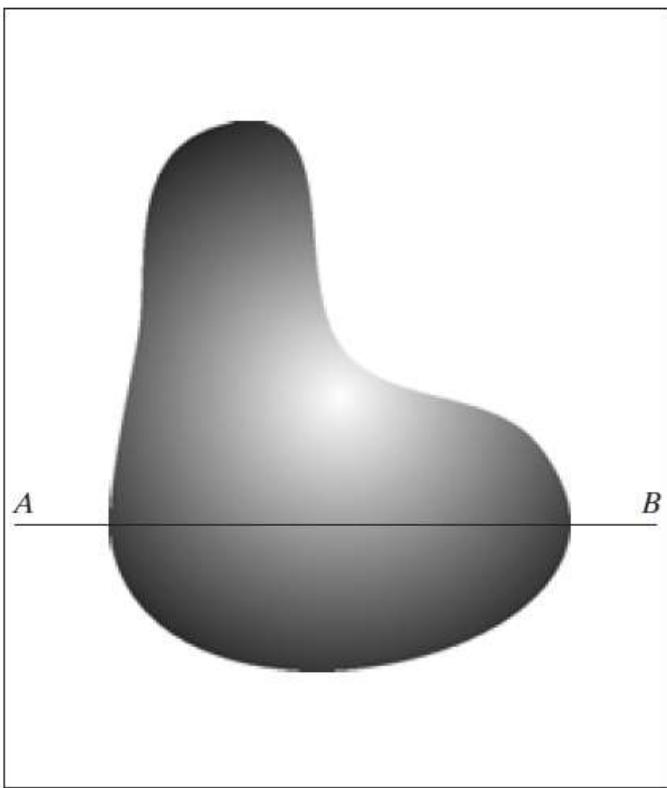


FIGURE 2.16
Generating a digital image.
(a) Continuous image. (b) A scan line from A to B in the continuous image, used to illustrate the concepts of sampling and quantization.

4. Image Sampling and Quantization

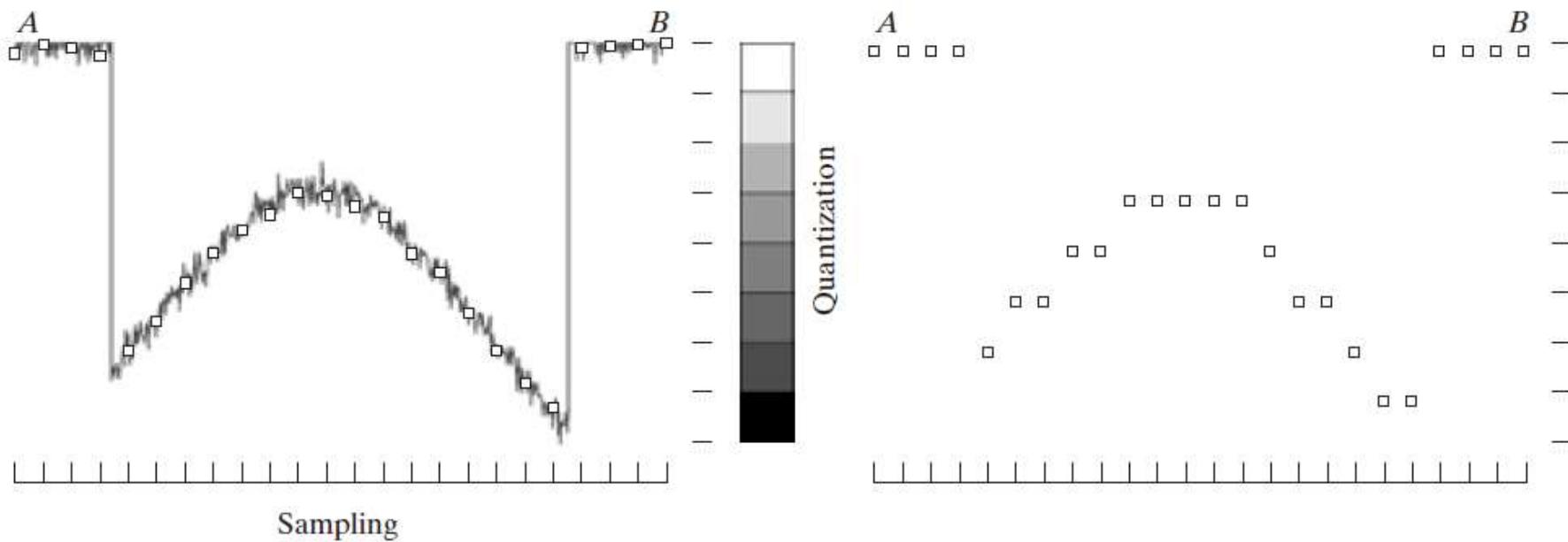
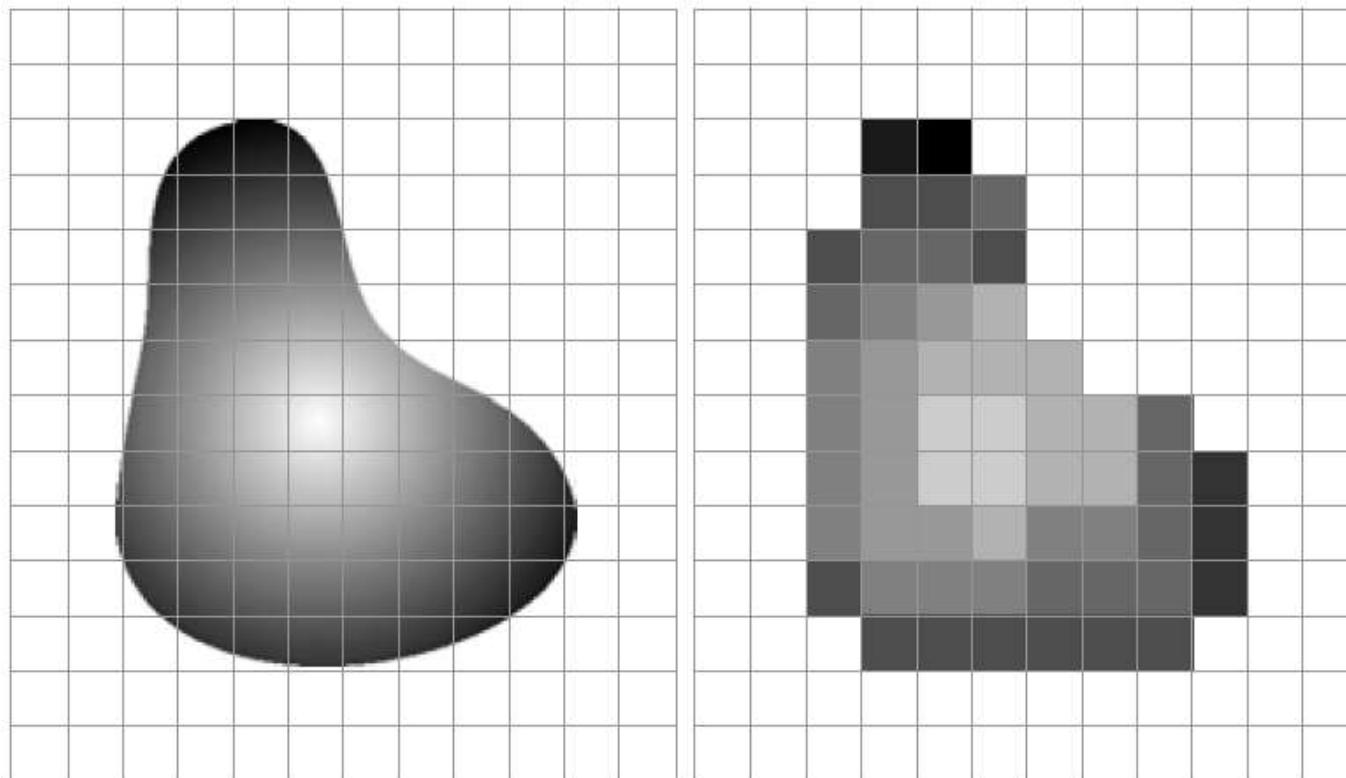


FIGURE 2.16
(c) Sampling and quantization.
(d) Digital scan line.

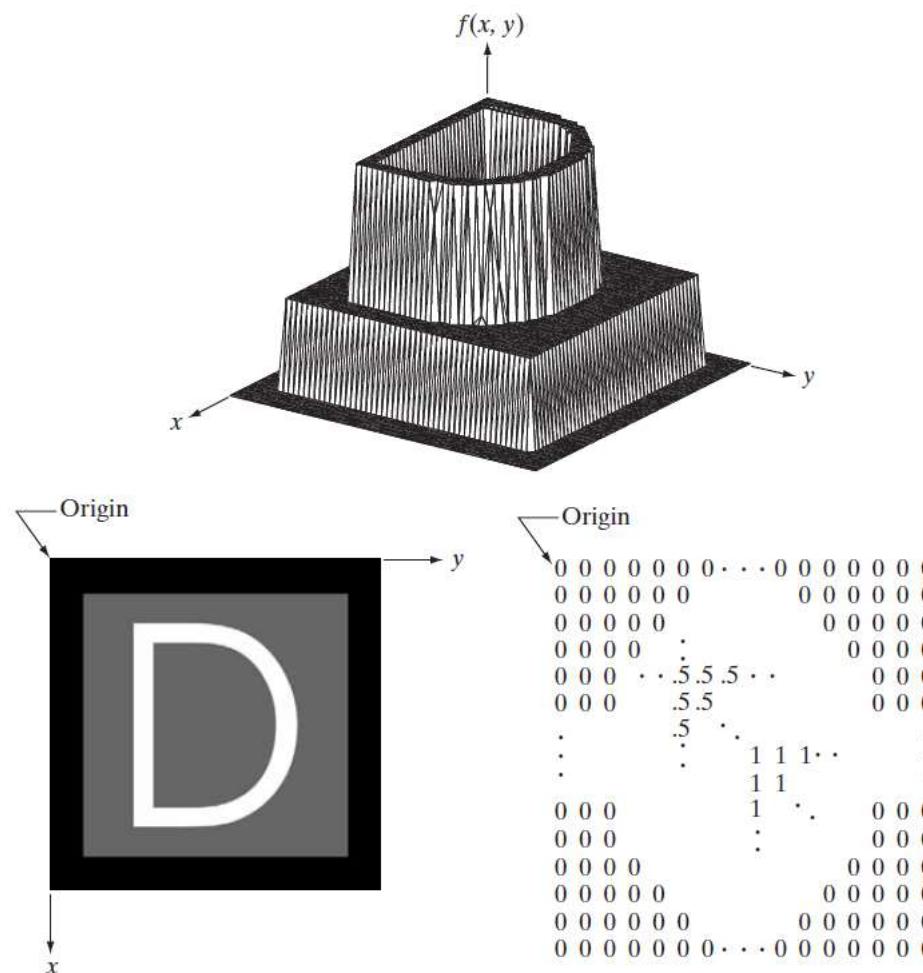
4. Image Sampling and Quantization



a | b

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

4. Image Sampling and Quantization



a
b c

FIGURE 2.18
 (a) Image plotted as a surface.
 (b) Image displayed as a visual intensity array.
 (c) Image shown as a 2-D numerical array (0, .5, and 1 represent black, gray, and white, respectively).

4.1 Representing Digital Images

- In equation form, we write the representation of an $M \times N$ numerical array as:

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

- Or,

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix}$$

- Each element of this matrix is called an **image element, picture element or pixel**.

4.1 Representing Digital Images

- This digitization process requires that decisions be made regarding the values for M, N, and for the number, L, of discrete intensity levels.
- There are no restrictions placed on M and N, other than they have to be positive integers. However, due to storage and quantizing hardware considerations, the number of intensity levels typically is an integer power of 2:

$$L = 2^k$$

- We assume that the discrete levels are equally spaced and that they are integers in the interval [0, L-1].
- The number, b , of bits required to store a digitized image is

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

4.1 Representing Digital Images

- Example ($M = 8$ row, $N = 8$ column , $k = 8$ bits):

	0	1	2	3	4	5	6	7	N
0	8	9	43	98	15	13	10	14	
1	9	11	94	50	15	13	10	14	
2	9	11	161	29	14	40	96	17	
3	9	14	170	21	15	148	173	19	
4	9	23	153	17	23	168	166	19	
5	9	53	110	15	43	175	168	20	
6	10	101	67	15	74	174	167	20	
7	11	141	41	14	117	173	164	20	

$b = 8 \times 8 \times 8 = 512$ bits

M ↓

4.1 Representing Digital Images

- MATLAB functions:

- `clear all`
- `close all`
- `x = zeros(M,N)`
- `x = ones(M,N)`
- `y = uint8(x)`
- `imshow(y)`
- `figure`

4.2 Spatial and Intensity Resolution

- Intuitively, **spatial resolution** is a measure of the smallest discernible detail in an image.
- Can be stated in a number of ways, with **line pairs per unit distance**, and **dots (pixels) per unit distance**.
- **Image size** by itself does not tell the complete story. To say that an image has, say, a resolution 1024×1024 pixels is not a meaningful statement without stating the spatial dimensions encompassed by the image.

4.2 Spatial and Intensity Resolution

- Intuitively, **spatial resolution** is a measure of the smallest discernible detail in an image.



FIGURE 2.19 A 1024×1024 , 8-bit image subsampled down to size 32×32 pixels. The number of allowable gray levels was kept at 256.

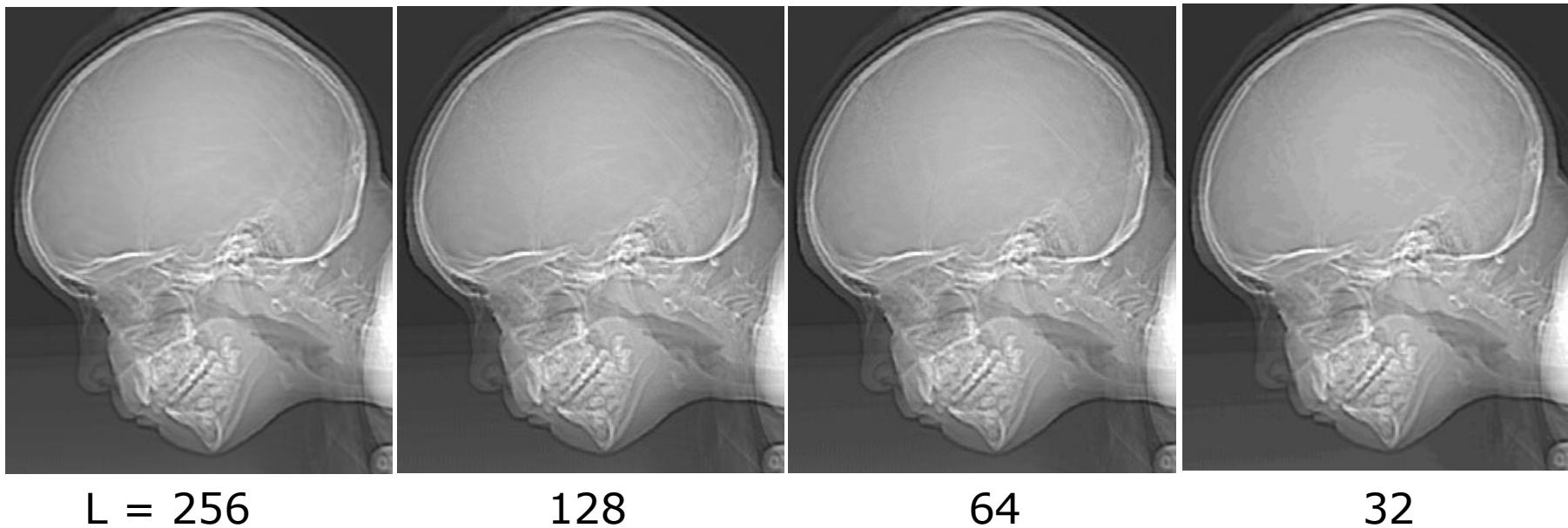
4.2 Spatial and Intensity Resolution



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.

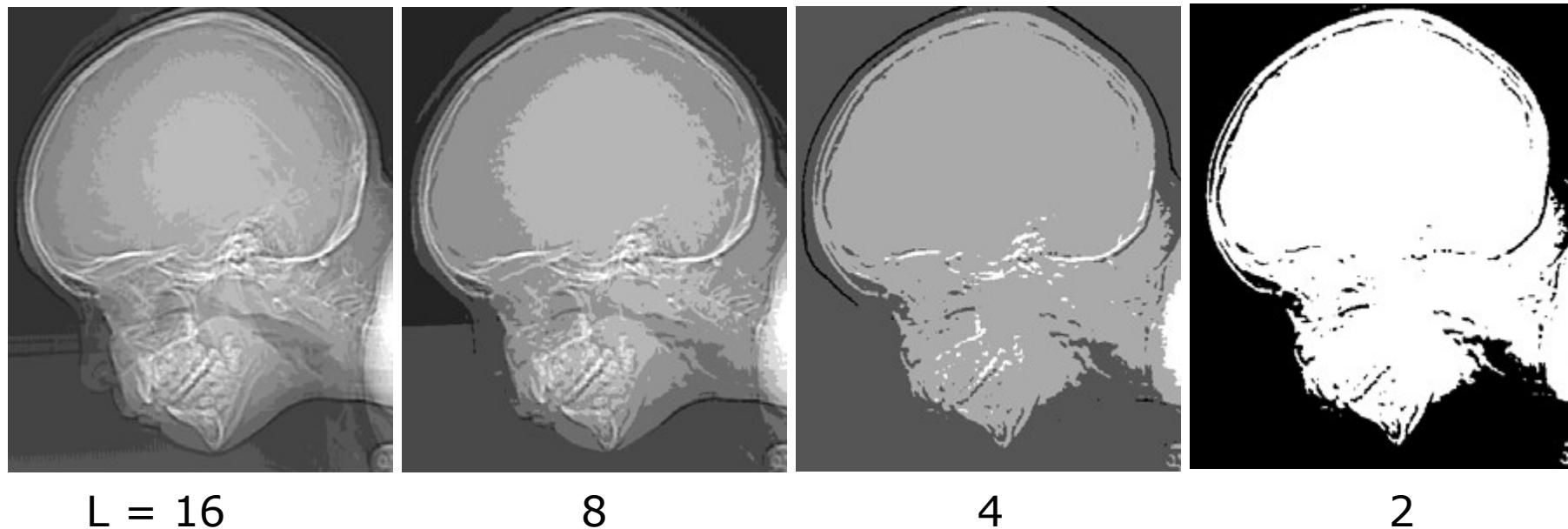
4.2 Spatial and Intensity Resolution

- **Intensity resolution** refers to the smallest discernible change in intensity level.
- For example, an image whose intensity is quantized into 256 levels has 8 bits of intensity resolution.



4.2 Spatial and Intensity Resolution

- **Intensity resolution** refers to the smallest discernible change in intensity level.
- For example, an image whose intensity is quantized into 256 levels has 8 bits of intensity resolution.



L = 16

8

4

2

4.2 Spatial and Intensity Resolution

- Effects on image quality produced by varying N and k simultaneously (Huang [1965]).



a b c

FIGURE 2.22 (a) Image with a low level of detail. (b) Image with a medium level of detail. (c) Image with a relatively large amount of detail. (Image (b) courtesy of the Massachusetts Institute of Technology.)

4.2 Spatial and Intensity Resolution

- Effects on image quality produced by varying N ($N=M$) and k simultaneously (Huang [1965]).
- Points lying on an **isopreference curve** correspond to images of equal subjective quality

For a fixed value of N , the perceived quality for this type of image is nearly independent of the number of intensity levels used.

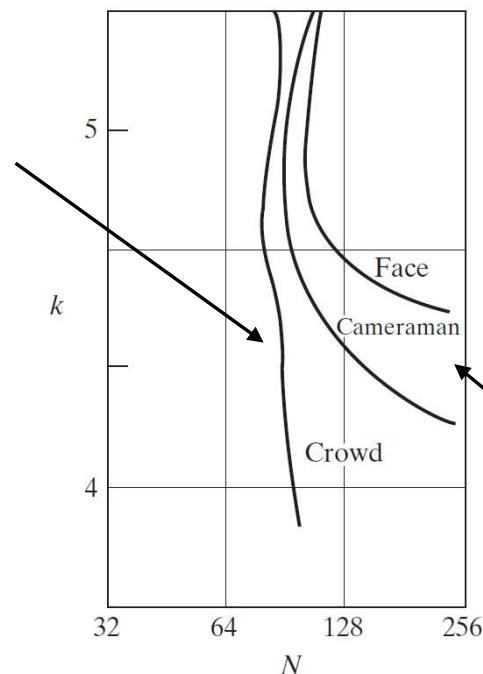
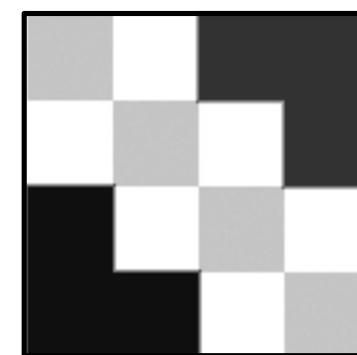
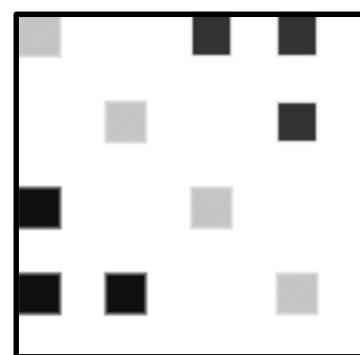


FIGURE 2.23
Typical
isopreference
curves for the
three types of
images in
Fig. 2.22.

Quality remained the same in some intervals in which the number of samples was increased, but the number of intensity levels actually decreased.

4.3 Image Interpolation

- **Interpolation** (resampling/resizing: shrinking and zooming) is the process of using known data to estimate values at unknown locations.
- Suppose that an image of size 4×4 pixels has to be enlarged 2 times to 8×8 pixels.
 - **Nearest neighbor interpolation** assigns to each new location the intensity of its nearest neighbor in the original image.



4.3 Image Interpolation

- **Bilinear interpolation:**

$$f(x, y) = \sum_{i=0}^1 \sum_{j=0}^1 a_{ij} x^i y^j = a_{00} + a_{10}x + a_{01}y + a_{11}xy$$

a_{00}	a_{10}
a_{01}	a_{11}

$$a_{00} = f(0, 0)$$

$$a_{10} = f(1, 0) - f(0, 0)$$

$$a_{01} = f(0, 1) - f(0, 0)$$

$$a_{11} = f(1, 1) + f(0, 0) - (f(1, 0) + f(0, 1))$$

- **Bicubic interpolation:**

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

5. Some Basic 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 $N_4(p)$.

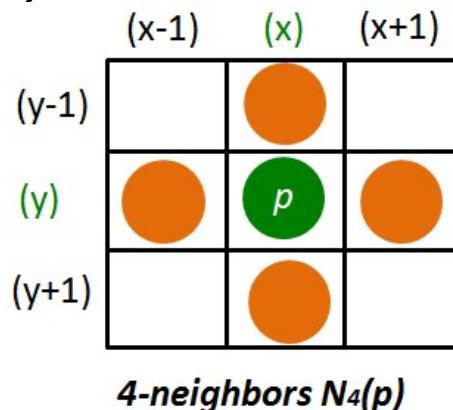


Image: <http://cse19-iiith.vlabs.ac.in/theory.php?exp=diff>

5. Some Basic Relationships between Pixels

- **Neighbors of a Pixel:**

- The four **diagonal neighbors** of p have coordinates

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

and are denoted by $N_D(p)$.

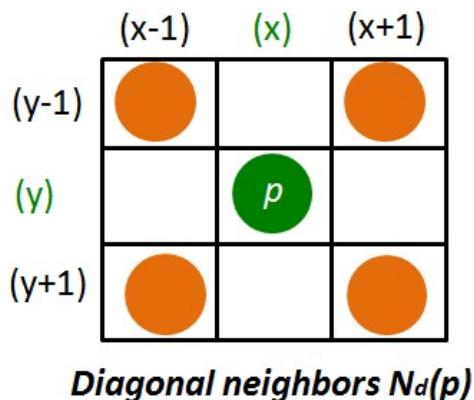


Image: <http://cse19-iiith.vlabs.ac.in/theory.php?exp=diff>

5. Some Basic Relationships between Pixels

- **Neighbors of a Pixel:**

- Diagonal neighbors together with the 4-neighbors are called the **8-neighbors** of p , denoted by

$$N_8(p) = N_4(p) \cup N_D(p)$$

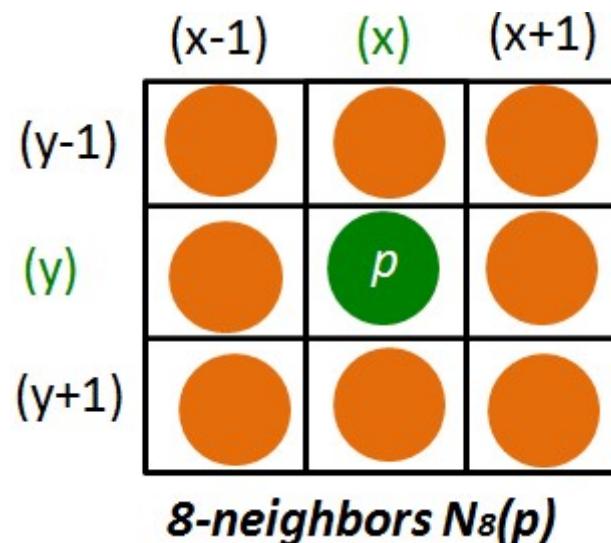


Image: <http://cse19-iiith.vlabs.ac.in/theory.php?exp=diff>

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **4-adjacent** if q is in the set $N_4(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **8-adjacent** if q is in the set $N_8(p)$. If $V = \{1\}$:

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	1	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two pixels p and q with values from V (set of intensity values used to define adjacency) are **m-adjacent** if:

- a. q is in $N_4(p)$; or
- b. q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V .

- Is introduced to eliminate the ambiguities that often arise when 8-adjacency is used.

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

$$\begin{aligned} N_4(p) \cap N_4(q) \\ \notin V \end{aligned}$$

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

$$\begin{aligned} N_4(p) \cap N_4(q) \\ \notin V \end{aligned}$$

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

$N_4(p) \cap N_4(q)$
 $\in V$

$N_4(p) \cap N_4(q)$
 $\notin V$

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

$N_4(p) \cap N_4(q)$
 $\notin V$

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

$N_4(p) \cap N_4(q)$
 $\in V$

$N_4(p) \cap N_4(q)$
 $\notin V$

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ **m-adjacent:**

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	1	0	0	0	0
0	0	0	1	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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

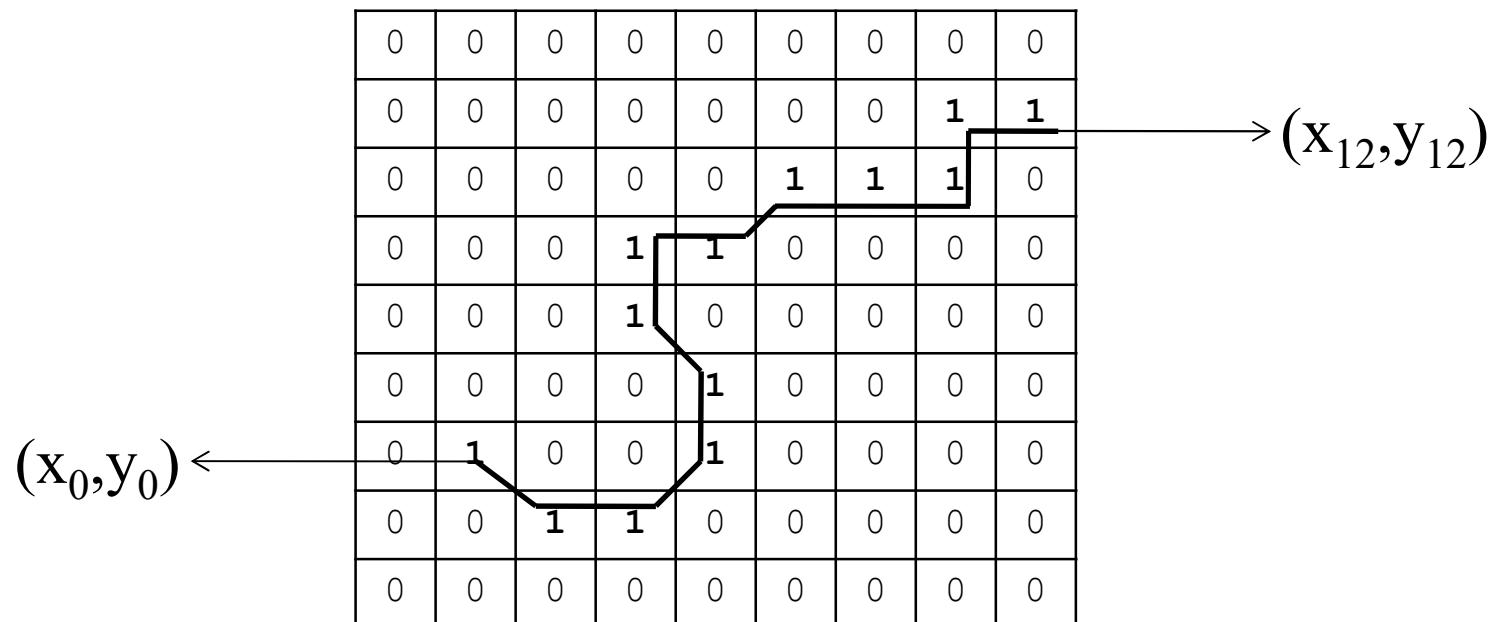
➤ A (digital) path (or curve) from pixel $p(x,y)$ to pixel $q(s,t)$ with is a sequence of distinct pixels with coordinates:

$$(x,y) = (x_0, y_0), (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) = (s, t)$$

- Pixels (x_i, y_i) 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.
 - We can define 4-, 8-, or m-paths depending on the type of adjacency specified.

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**



5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- If $(x_0, y_0) = (x_n, y_n)$ the path is a **closed** path.

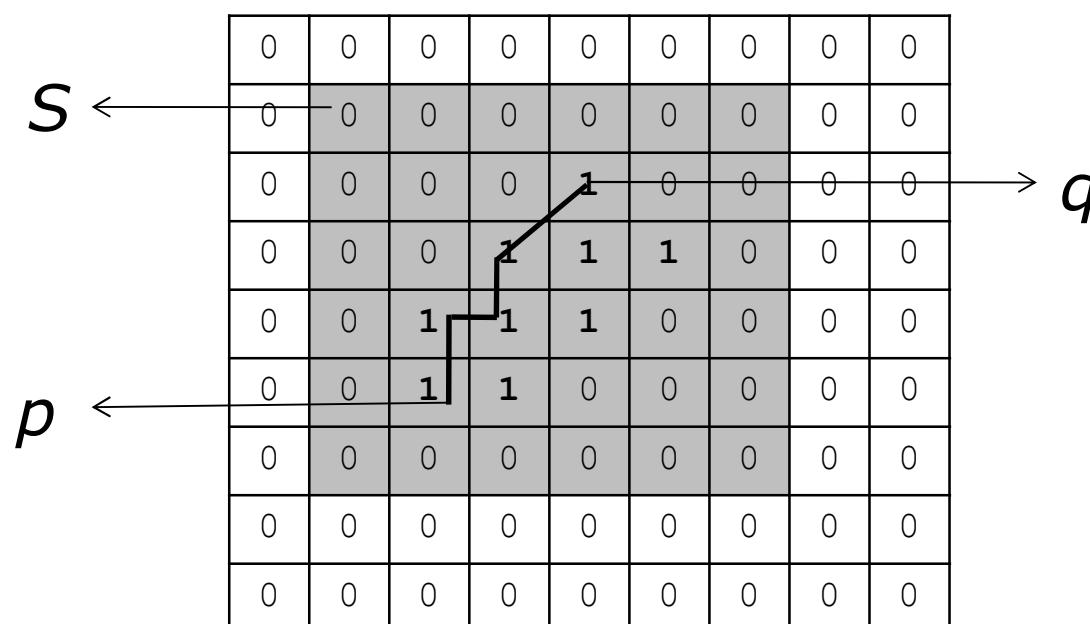
$(x_0, y_0) = (x_{10}, y_{10})$ ←

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

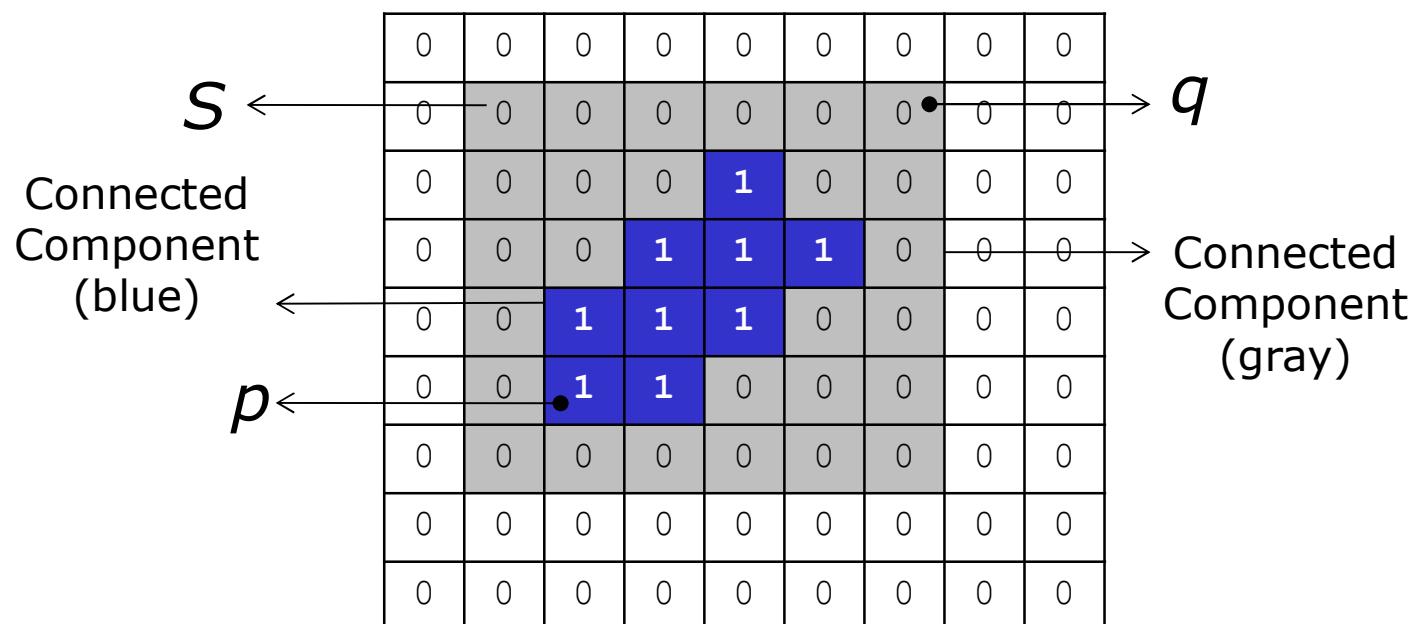
- Let S represent a subset of pixels in an image. Two pixels p and q are said to be **connected** in S if there exists a path between them consisting entirely of pixels in S .



5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- For any pixel p in S , the set of pixels that are connected to it in S is called a **connected component** of S .



5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- If it only has one connected component, then S is called a **connected set**.

S ← → Connected Set

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- Let R be a subset of pixels in an image. We call R a **region** of the image if R is a connected set.

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Two regions, R_i and R_j are said to be adjacent if their union forms a connected set. For our definition to make sense, the type of adjacency used must be specified.

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- In our example, R_1 and R_2 are 8-adjacent.

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ Regions that are not adjacent are said to be **disjoint**. In our example, R_1 and R_2 are disjoint if 4-adjacency is considered.

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

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- Suppose that an image contains k disjoint regions.
- Let R_u denote the union of all the k regions, and let $(R_u)^c$ denote its complement.

	0	0	0	0	0	0	0	0	0
R_1	1	1	1	0	0	0	0	0	0
	1	1	1	0	1	0	0	0	0
R_2	0	0	0	1	1	1	0	1	0
	0	0	1	1	1	0	1	1	
	0	0	1	1	0	0	1	1	0
	0	0	0	0	0	1	1	0	0
$(R_u)^c$	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ We call R_u the **foreground** and $(R_u)^c$ the **background**.

0	0	0	0	0	0	0	0	0
R_1	1	1	1	0	0	0	0	0
	1	1	1	0	1	0	0	0
R_2	0	0	0	1	1	1	0	1
	0	0	1	1	1	0	1	1
	0	0	1	1	0	0	1	1
	0	0	0	0	0	1	1	0
$(R_u)^c$	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

➤ The **boundary** (also called the border or contour) of a region R is the set of points that are adjacent to points in the complement of R (inner border).

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

(inner) border ←

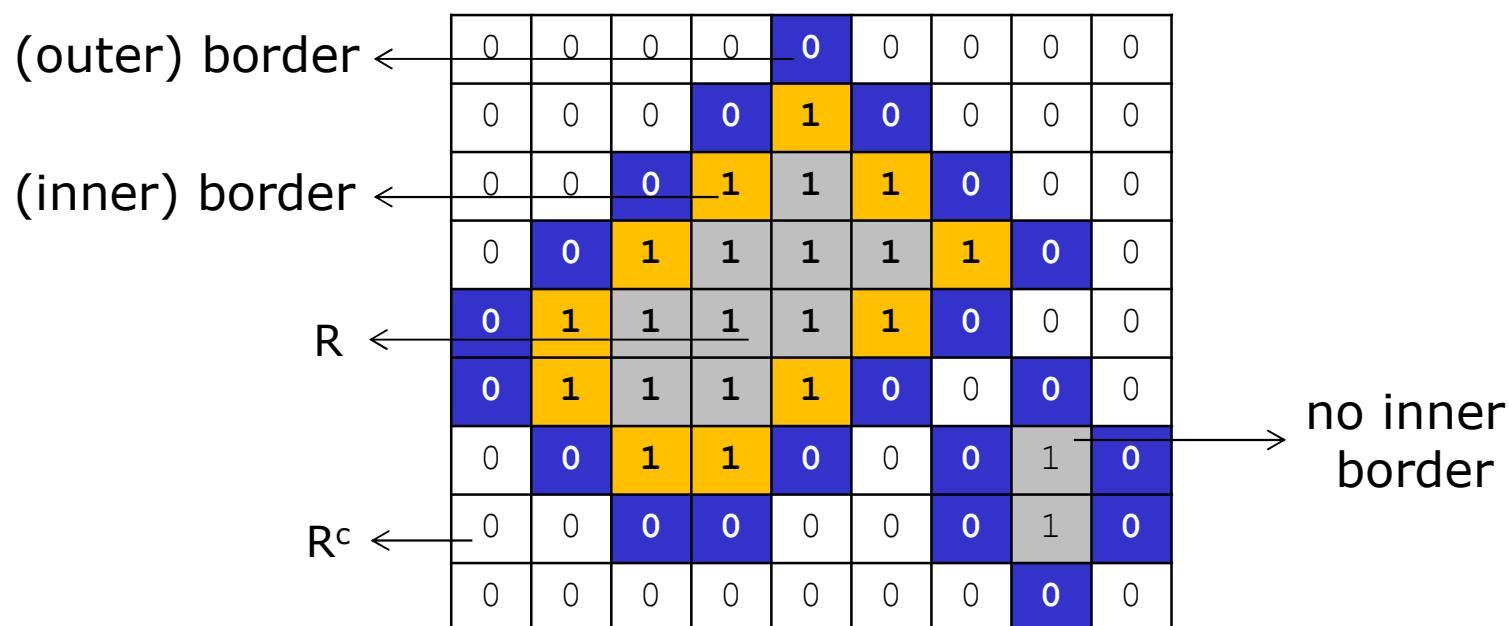
R ←

R^c ←

5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- Border-following algorithms usually are formulated to follow the **outer border**.



5. Some Basic Relationships between Pixels

- **Adjacency, Connectivity, Regions, and Boundaries**

- The concept of an **edge** is found frequently in discussions dealing with regions and **boundaries**.
- There is a key difference between these concepts, however.
 - ✓ The **boundary** of a finite region forms a **closed path**.
 - ✓ **Edges** are formed from pixels with **derivative values** that exceed a preset threshold (intensity discontinuities).

5. Some Basic Relationships between Pixels

- **Distance Measures**

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

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

5. Some Basic Relationships between Pixels

- **Distance Measures**

➤ Euclidean distance

$$D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$$

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0
2	0	0	0	1	1	1	0	0	0
3	0	0	1	1	1	1	1	0	0
4	0	1	1	1	1	1	0	0	0
5	0	1	1	1	1	0	0	0	0
6	0	0	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0

→ (2,3)

$$D_e(p, q) = \sqrt{(5-2)^2 + (1-3)^2} \approx 3,6$$

→ (5,1)

5. Some Basic Relationships between Pixels

- **Distance Measures**

➤ D_4 distance (called the city-block distance):

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

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0
2	0	0	0	1	1	1	0	0	0
3	0	0	1	1	1	1	1	0	0
4	0	1	1	1	1	1	0	0	0
5	0	1	1	1	1	0	0	0	0
6	0	0	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0

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

→ (2,3)

$$D_4(p, q) = |5 - 2| + |1 - 3| = 5$$

→ (5,1)

5. Some Basic Relationships between Pixels

- **Distance Measures**

➤ D_8 distance (called the chessboard distance):

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

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0
2	0	0	0	1	1	1	0	0	0
3	0	0	1	1	1	1	1	0	0
4	0	1	1	1	1	1	0	0	0
5	0	1	1	1	1	0	0	0	0
6	0	0	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0

→ (2,3)

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

$$D_8(p, q) = \max(|5 - 2|, |1 - 3|) = 3$$

→ (5,1)

5. Some Basic Relationships between Pixels

- **Distance Measures**

➤ D_m distance (shortest m-path between the points two pixels $p \in q$ ($V=\{1\}$)):

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

$$D_m(p,q) = 2$$

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

$$D_m(p,q) = 3$$

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

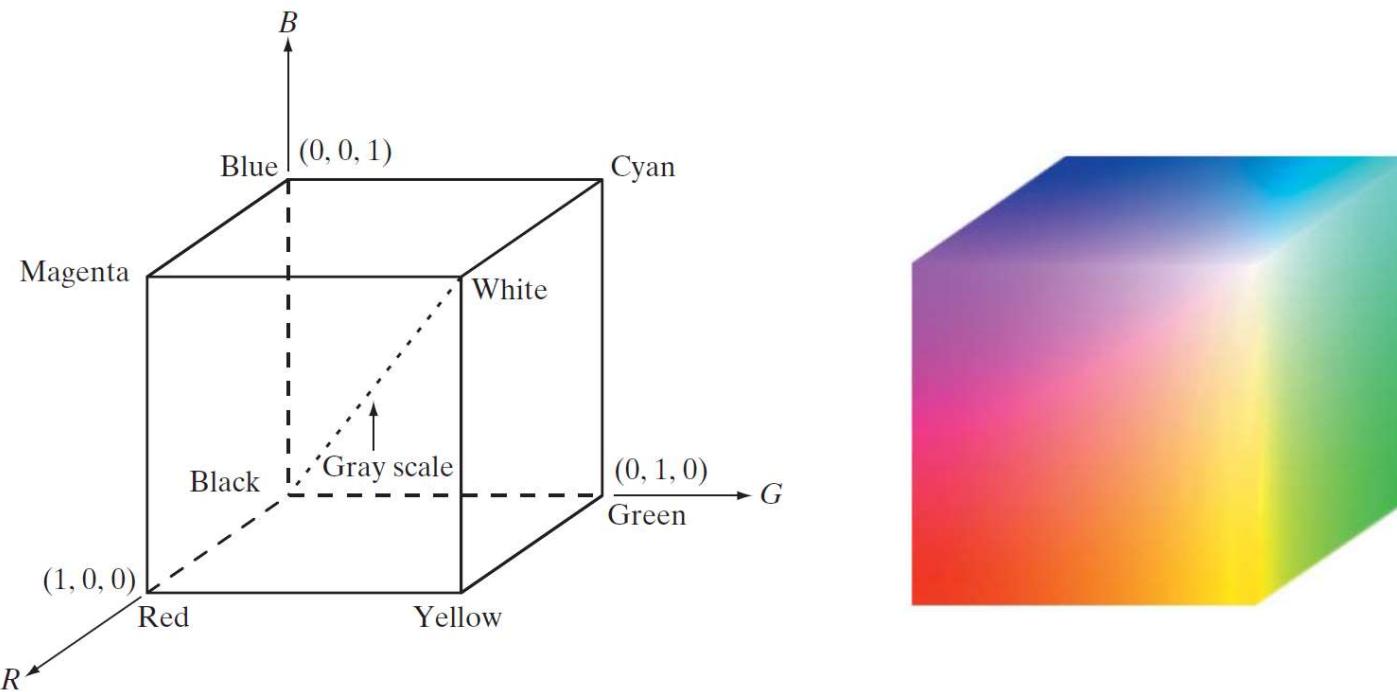
$$D_m(p,q) = 3$$

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

$$D_m(p,q) = 4$$

6. Color Models

- **RGB** ($24 \text{ bits} \rightarrow 256^3 \text{ colors} \rightarrow 16.777.216 \text{ of colors}$):



6. Color Models

- **RGB** (24 bits → 256³ colors → 16.777.216 of colors):

R



G



B



6. Color Models

- **YC_bC_r:**

- ✓ The human visual system is more sensitive to luminance (brightness) than to color.
- ✓ However, RGB does not take advantage of this fact. The luminance is spread over the components R, G and B, making them equally relevant.
- ✓ Another useful representation, called YC_bC_r, separates the luminance component Y from the color components, C_b and C_r, also called chrominance.
- ✓ The Y component can be viewed as a grayscale version of the color image.

6. Color Models

- **RGB ↔ YCbCr:**

$$Y = 0,299R + 0,587G + 0,114B$$

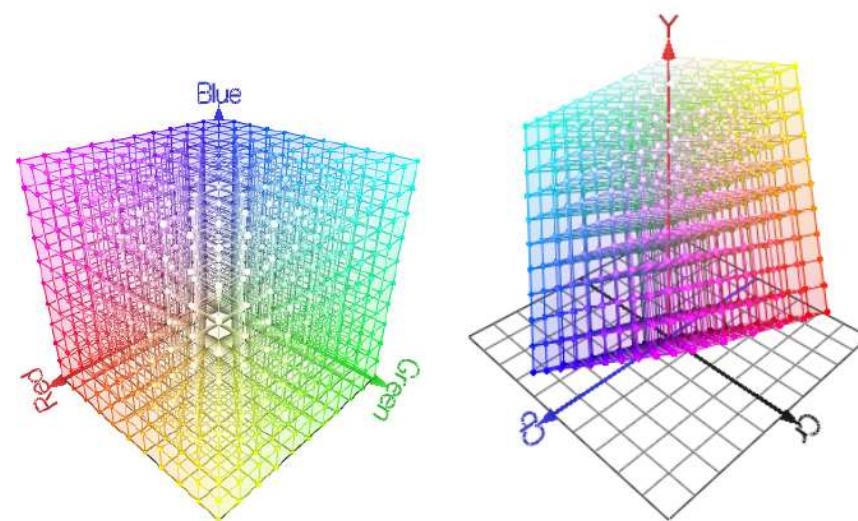
$$C_b = 0,564(B - Y)$$

$$C_r = 0,713(R - Y)$$

$$R = Y + 1,402C_r$$

$$G = Y - 0,344C_b - 0,714C_r$$

$$B = Y + 1,772C_b$$

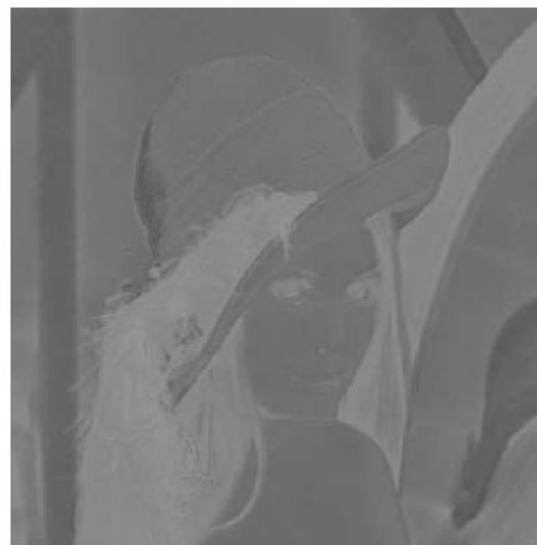


6. Color Models

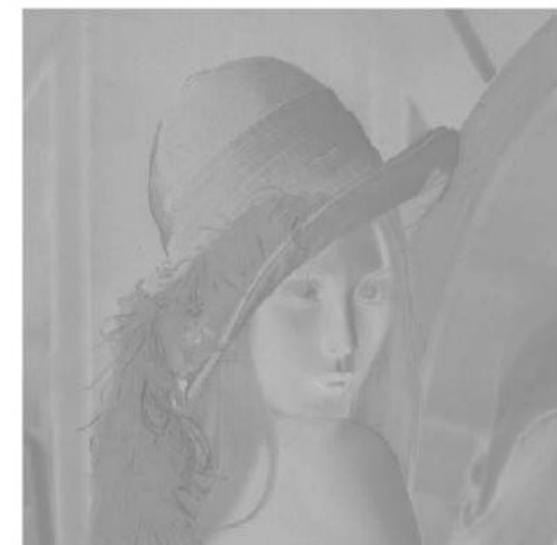
- **RGB → YCbCr:**



Y



Cb



Cr

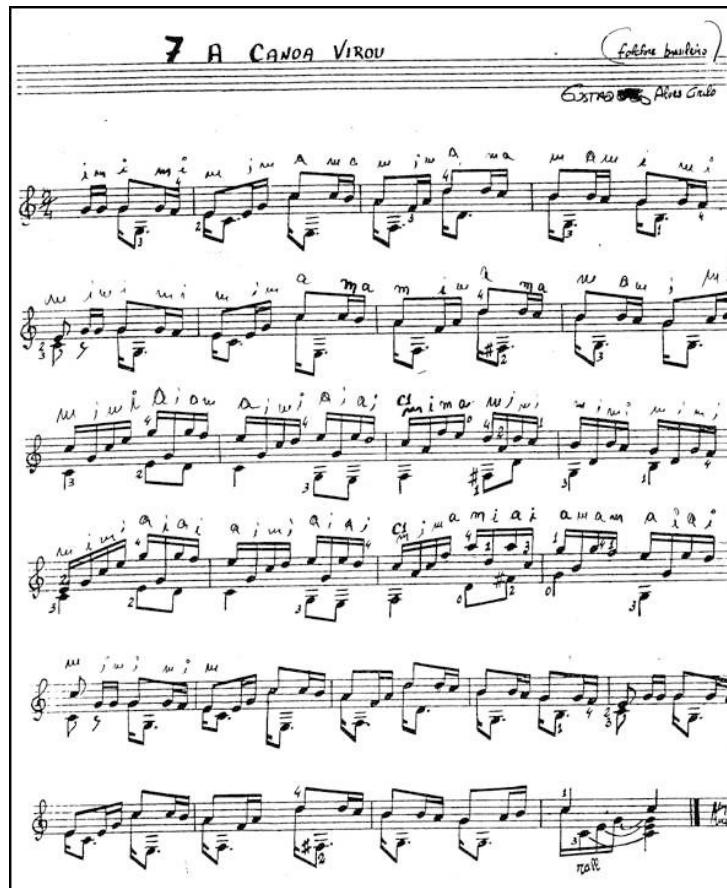
6. Color Models

- **Grayscale images** (*8 bits* → 256 gray levels):



6. Color Models

- **Binary images** ($1 \text{ bit} \rightarrow 2 \text{ levels}$):



7. Mathematical Tools

- **Array versus Matrix Operations**

➤ *Array product:*

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

➤ *Matrix product:*

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

7. Mathematical Tools

• Linear versus Nonlinear operations

- The output of a linear operation due to the sum of two inputs is the same as performing the operation on the inputs individually and then summing the results (*additivity*).
- The output of a linear operation to a constant times an input is the same as the output of the operation due to the original input multiplied by that constant (*homogeneity*).

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

$$\begin{aligned} H[a_i f_i(x, y) + a_j f_j(x, y)] &= a_i H[f_i(x, y)] + a_j H[f_j(x, y)] \\ &= a_i g_i(x, y) + a_j g_j(x, y) \end{aligned}$$

7. Mathematical Tools

• Arithmetic Operations

- Arithmetic operations between images are array operations.
- Operations are performed between corresponding pixel pairs in f and g .

$$s(x, y) = f(x, y) + g(x, y)$$

$$d(x, y) = f(x, y) - g(x, y)$$

$$p(x, y) = f(x, y) \times g(x, y)$$

$$v(x, y) = f(x, y) \div g(x, y)$$

7. Mathematical Tools

• Arithmetic Operations

- Example (*image averaging/denoising*):

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

- The assumption is that at every pair of coordinates (x, y) the noise is uncorrelated† and has zero average value.
- If the noise satisfies the constraints just stated, and an image \bar{g} is formed by averaging K different noisy images,

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^K g_i(x, y)$$

7. Mathematical Tools

• Arithmetic Operations

➤ Example (*image averaging/denoising*):

✓ Then it follows that,

$$E\{\bar{g}(x, y)\} = f(x, y)$$

✓ Variance and standard deviation of \bar{g} :

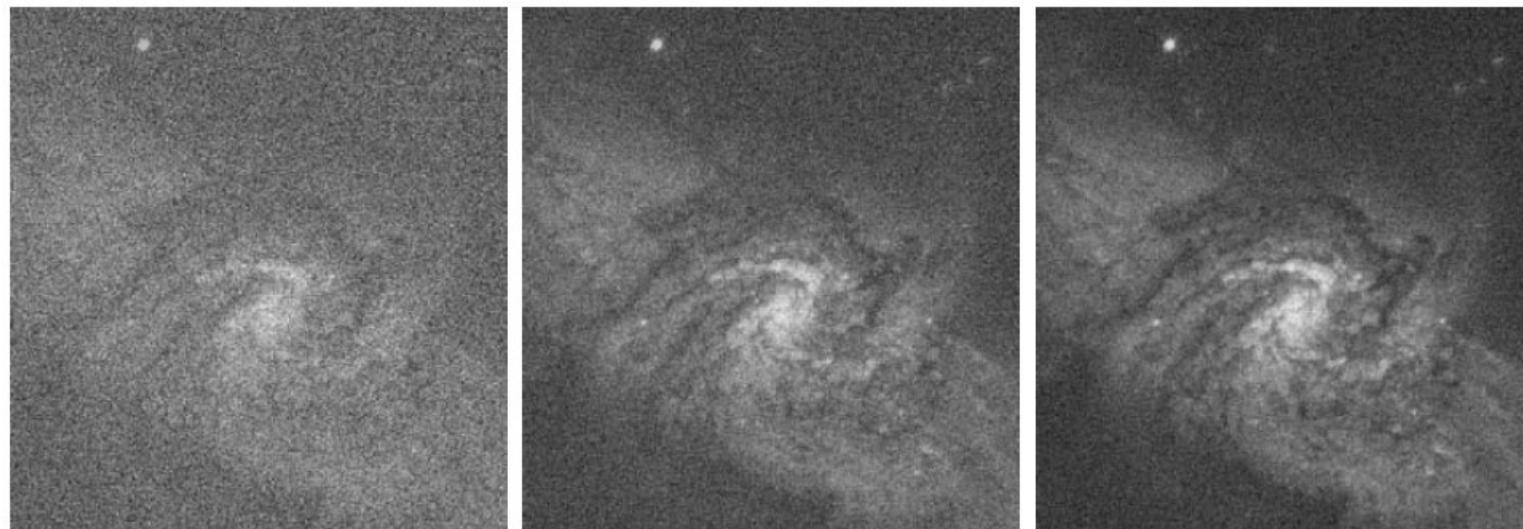
$$\sigma_{\bar{g}(x,y)}^2 = \frac{1}{K} \sigma_{\eta(x,y)}^2$$

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}} \sigma_{\eta(x,y)}$$

7. Mathematical Tools

- **Arithmetic Operations**

- Example (*image averaging/denoising*):



7. Mathematical Tools

- **Arithmetic Operations**

- Example (*image averaging/denoising*):

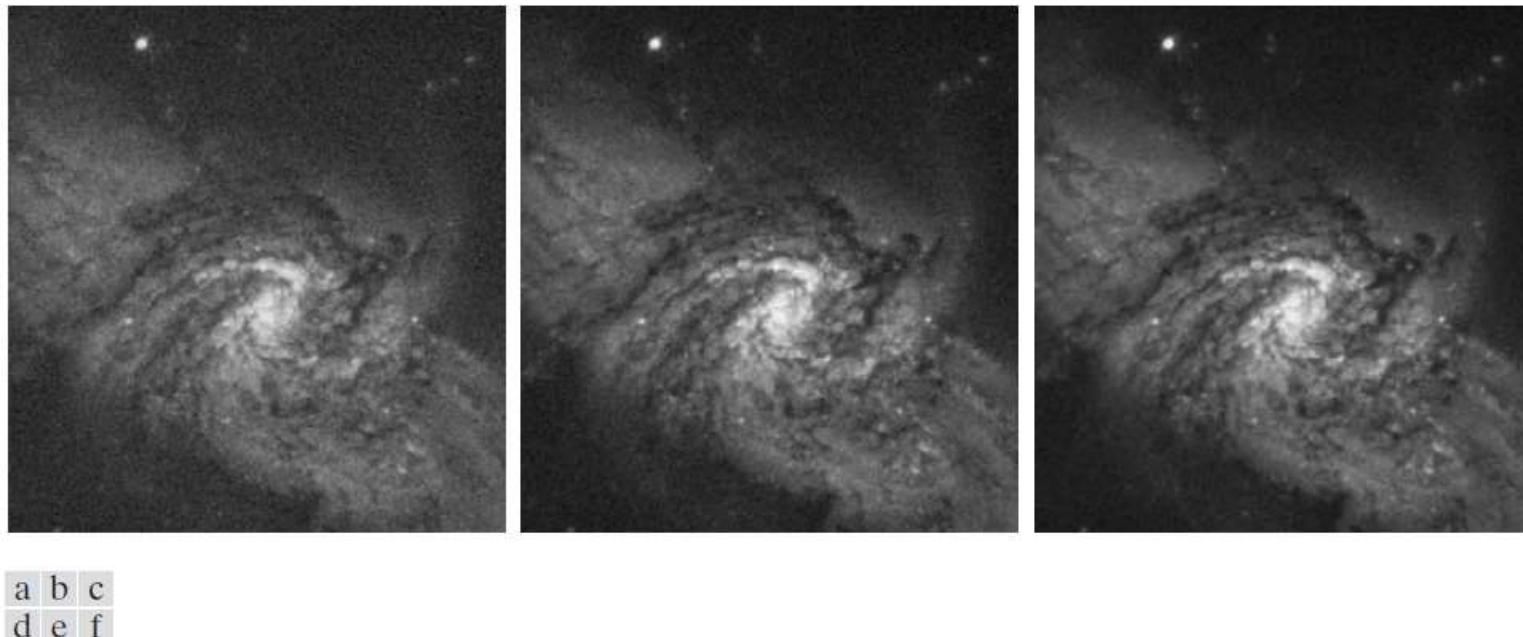


FIGURE 2.26 (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

7. Mathematical Tools

- **Arithmetic Operations**

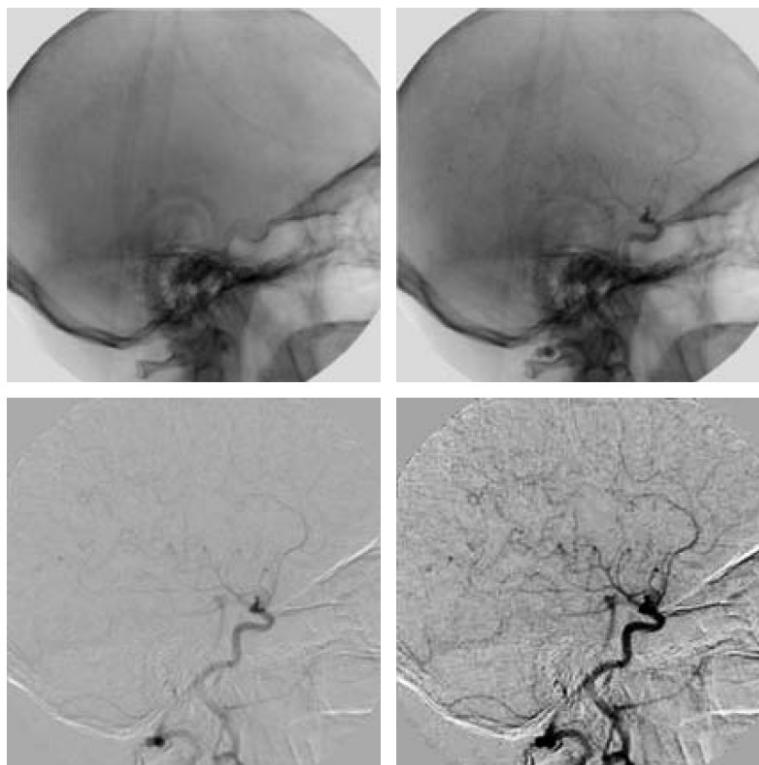
- MATLAB: s160Lena.m



7. Mathematical Tools

• Arithmetic Operations

- Example (*image subtraction/mask mode radiography*):



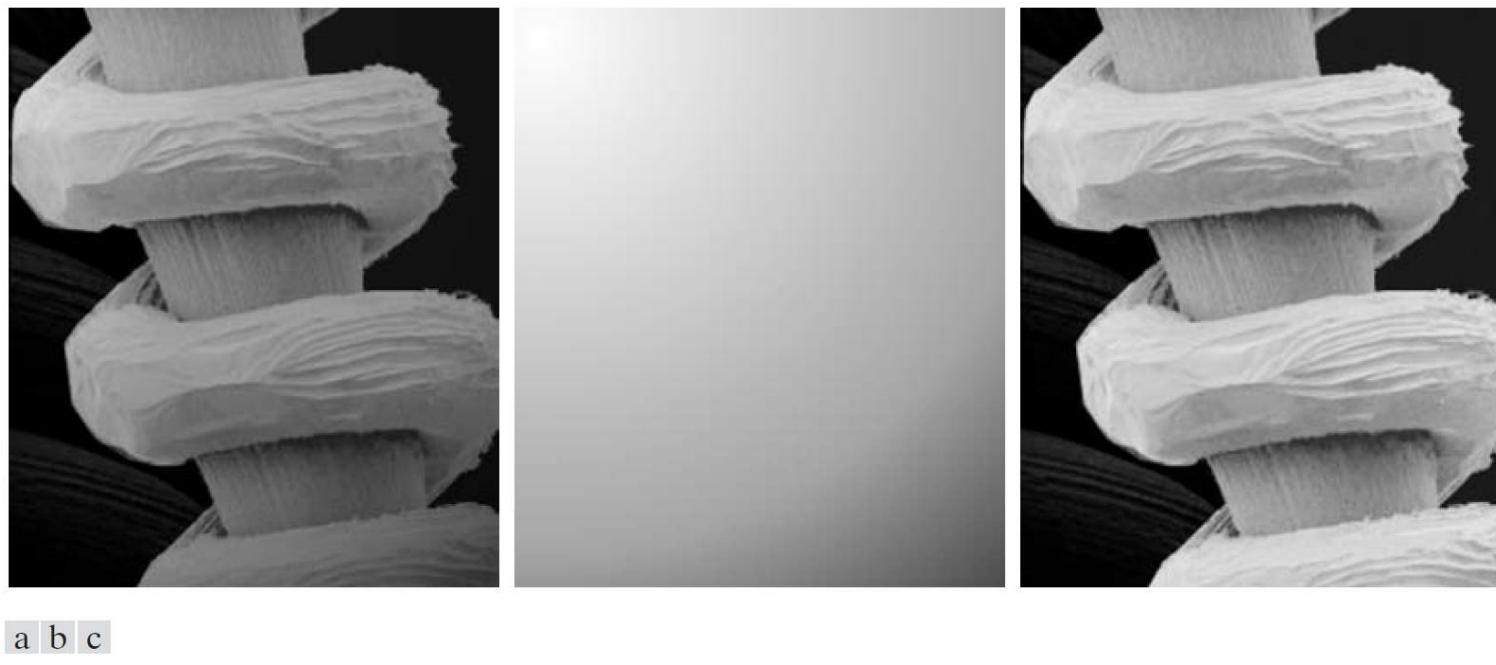
a	b
c	d

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

7. Mathematical Tools

- **Arithmetic Operations**

- Example (*image multiplication/shading correction*):



a b c

FIGURE 2.29 Shading correction. (a) Shaded SEM image of a tungsten filament and support, magnified approximately 130 times. (b) The shading pattern. (c) Product of (a) by the reciprocal of (b). (Original image courtesy of Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

7. Mathematical Tools

- **Arithmetic Operations**

- Example (*image multiplication/region of interest*):



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

7. Mathematical Tools

• Arithmetic Operations

- A few comments about implementing image arithmetic operations.
- In practice, most images are displayed using 8 bits (or three separate 8-bit channels).
- Thus, we expect image values to be in the range from 0 to 255.
- Given an image f , an approach that guarantees that the full range of an arithmetic operation between images is “captured” into a fixed number of $K+1$ intensity levels is as follows:

$$f_m = f - \min(f) \quad f_s = K \left[\frac{f_m}{\max(f_m)} \right]$$

7. Mathematical Tools

• Set and Logical Operations

- Let A be a set composed of ordered pairs of real numbers. If $a = (a_1, a_2)$ is an element of A , then we write $a \in A$
- Similarly, if a is not an element of A , we write $a \notin A$
- The set with no elements is called the null or empty set and is denoted by the symbol \emptyset .
- A set is specified by the contents of two braces: $\{\cdot\}$. For example: $C = \{w | w = -d, d \in D\}$
- One way in which sets are used in image processing is to let the elements of sets be the coordinates of pixels.

7. Mathematical Tools

• Set and Logical Operations

- If every element of a set A is also an element of a set B, then A is said to be a *subset* of B, denoted as

$$A \subseteq B$$

- The *union* of two sets A and B is denoted by

$$C = A \cup B$$

- Similarly, the *intersection* of two sets A and B, denoted by

$$D = A \cap B$$

- Two sets A and B are said to be *disjoint or mutually exclusive* if they have no common elements,

$$A \cap B = \emptyset$$

7. Mathematical Tools

• Set and Logical Operations

- The set *universe*, U , is the set of all elements in a given application.
- The *complement* of a set A is the set of elements that are not in A :

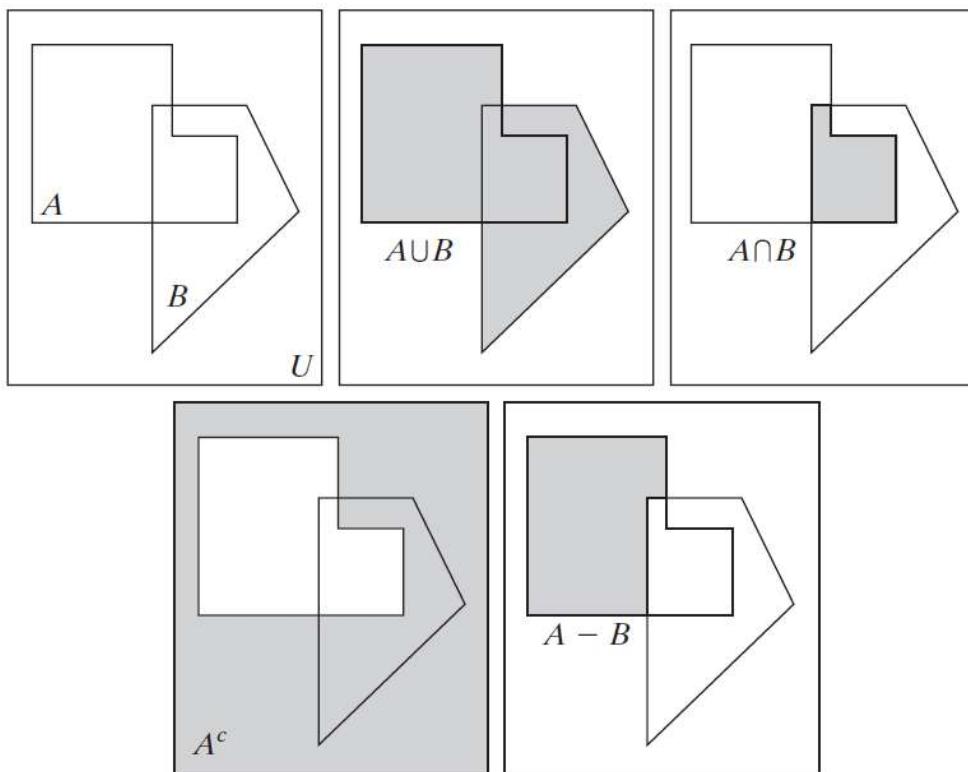
$$A^c = \{w | w \notin A\}$$

- The *difference* of two sets A and B is defined as

$$A - B = \{w | w \in A, w \notin B\} = A \cap B^c$$

7. Mathematical Tools

- Set and Logical Operations



a	b	c
d	e	

FIGURE 2.31

(a) Two sets of coordinates, A and B , in 2-D space. (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 . In (b)–(e) the shaded areas represent the members of the set operation indicated.

7. Mathematical Tools

- **Set and Logical Operations**

- Example:

- Let the elements of a gray-scale image be represented by a set A whose elements are triplets of the form (x, y, z) , where x and y are spatial coordinates and z denotes intensity.
 - We can define the complement of A as the set

$$A^c = \{(x, y, K - z) | (x, y, z) \in A\}$$

- Let A denote an 8-bit gray-scale image ($K=255$).

7. Mathematical Tools

- **Set and Logical Operations**

- Example:

- Let the elements of a gray-scale image be represented by a set A whose elements are triplets of the form (x, y, z) , where x and y are spatial coordinates and z denotes intensity.
 - We can define the complement of A as the set

$$A^c = \{(x, y, K - z) | (x, y, z) \in A\}$$

- Let A denote an 8-bit gray-scale image ($K=255$).

7. Mathematical Tools

- **Set and Logical Operations**

- Example:

- ✓ Suppose that we want to form the negative of A using set operations. We simply form the set

$$A^c = \{(x, y, 255 - z) | (x, y, z) \in A\}$$

7. Mathematical Tools

- **Set and Logical Operations**

- Example:

- ✓ The union of two gray-scale sets A and B may be defined as the set

$$A \cup B = \left\{ \max_z(a, b) \mid a \in A, b \in B \right\}$$

- ✓ Array formed from the maximum intensity between pairs of spatially corresponding elements.

7. Mathematical Tools

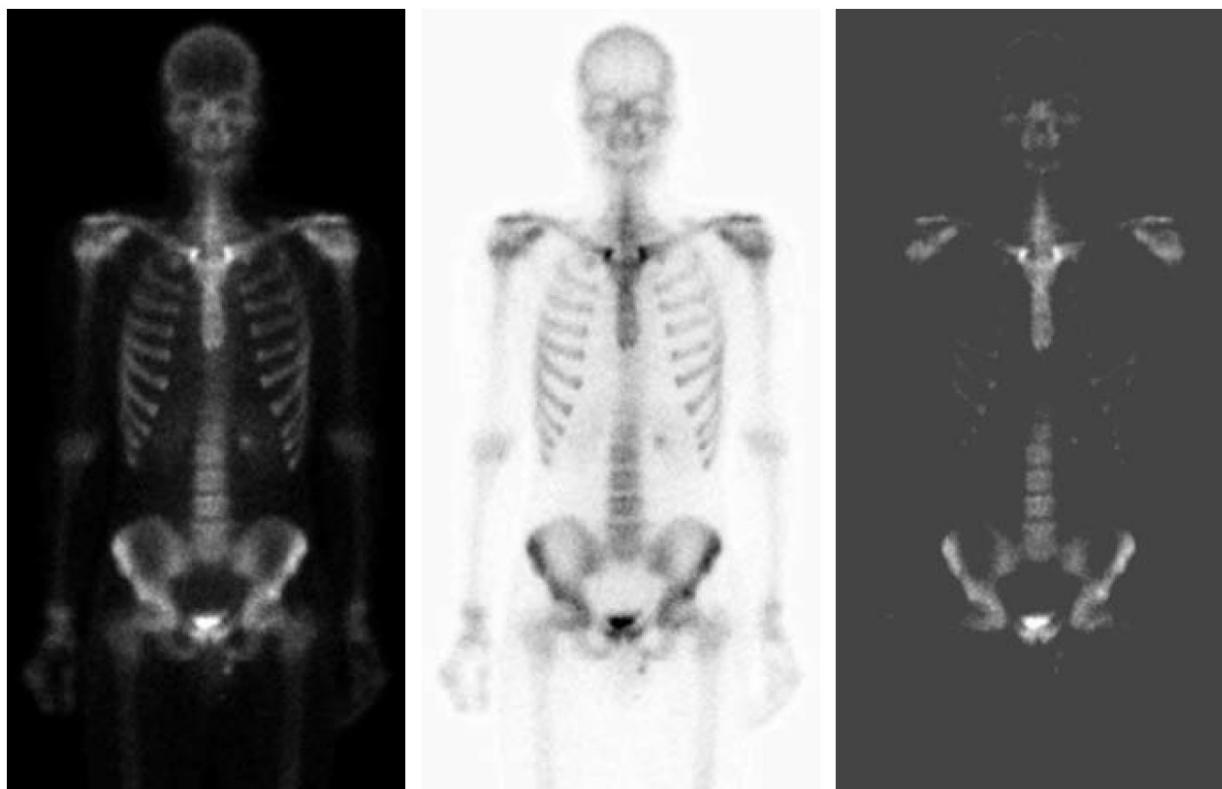
- **Set and Logical Operations**

- Example:

- ✓ Suppose that A again represents the image in, and let B denotes a rectangular array of the same size as A, but in which all values of z are equal to 3 times the mean intensity, m , of the elements of A.
 - ✓ The result of performing the set union, is show in the next slide.

7. Mathematical Tools

- Set and Logical Operations



a b c

FIGURE 2.32 Set operations involving gray-scale images.
(a) Original image. (b) Image negative obtained using set complementation.
(c) The union of (a) and a constant image.
(Original image courtesy of G.E. Medical Systems.)

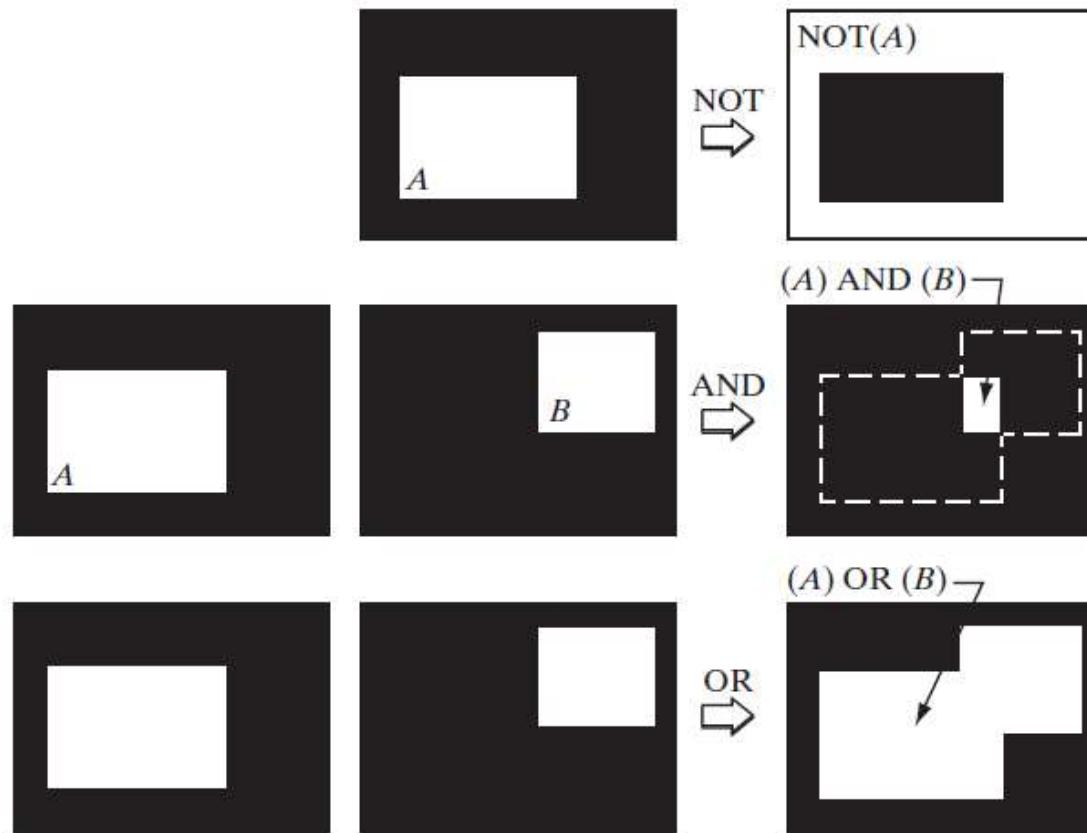
7. Mathematical Tools

• Set and Logical Operations

- When dealing with binary images, we can think of *foreground* (1-valued) and *background* (0-valued) sets of pixels.
- In this case, it is common practice to refer to *union*, *intersection*, and *complement* as the OR, AND, and NOT *logical operations*.

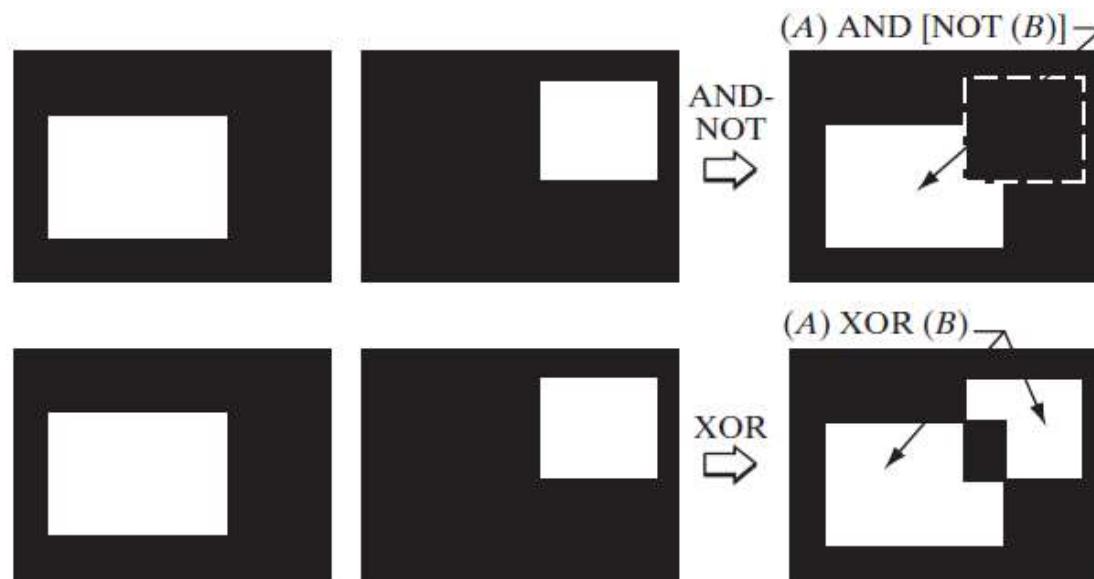
7. Mathematical Tools

- Set and Logical Operations



7. Mathematical Tools

- Set and Logical Operations



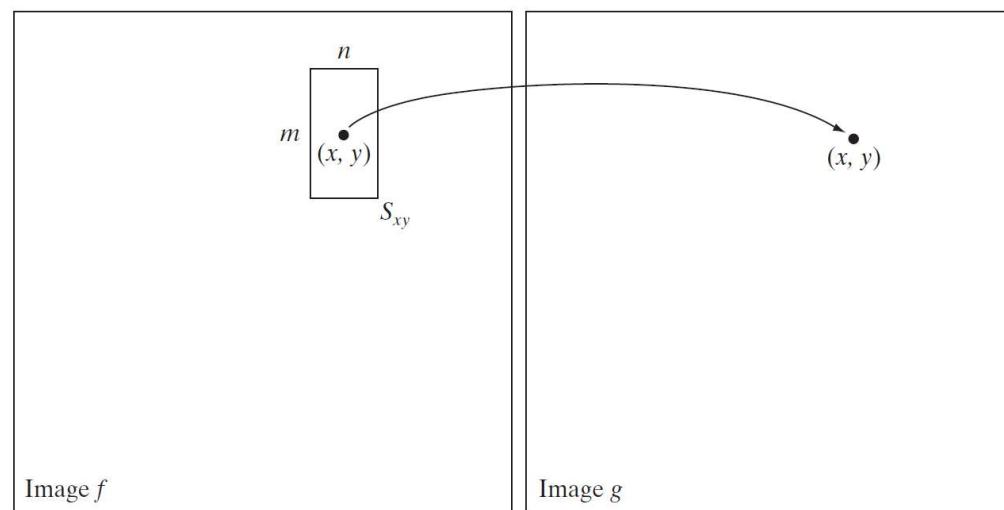
7. Mathematical Tools

• Spatial Operations

- *Single-pixel operations*: alter the values pixels based on their intensity.

$$s = T(z)$$

- *Neighborhood operations*: alter the values of pixels based on their neighborhood.



7. Mathematical Tools

- **Spatial Operations**

- *Geometric spatial*: modify the spatial relationship between pixels in an image.
- The transformation of coordinates may be expressed as

$$(x, y) = T\{(v, w)\}$$

- For example,

$$(x, y) = T\{(v, w)\} = (v/2, w/2)$$

shrinks the original image to half its size in both spatial directions.

7. Mathematical Tools

• Spatial Operations

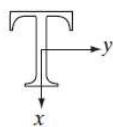
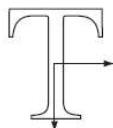
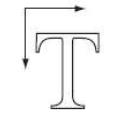
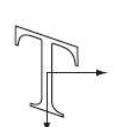
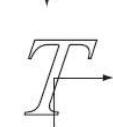
- One of the most commonly used spatial coordinate transformations is the *affine transform*:

$$[x \ y \ 1] = [v \ w \ 1] \mathbf{T} = [v \ w \ 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

- This transformation can scale, rotate, translate, or shear a set of coordinate points, depending on the value chosen for the elements of matrix T.

7. Mathematical Tools

- Spatial Operations

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

7. Mathematical Tools

• Spatial Operations

- *Image registration:* we have available the input and output images, but the specific transformation that produced the output image from the input generally is unknown.
- The problem, then, is to estimate the transformation function and then use it to register the two images.
- For example, suppose that we have a set of four tie points each in an input and a reference image. A simple model based on a bilinear approximation is given by

$$x = c_1v + c_2w + c_3vw + c_4$$

$$y = c_5v + c_6w + c_7vw + c_8$$

7. Mathematical Tools

• Spatial Operations

- If we have four pairs of corresponding tie points in both images, we can write eight equations and use them to solve for the eight unknown coefficients.
- These coefficients constitute the model that transforms the pixels of one image into the locations of the pixels of the other to achieve registration.

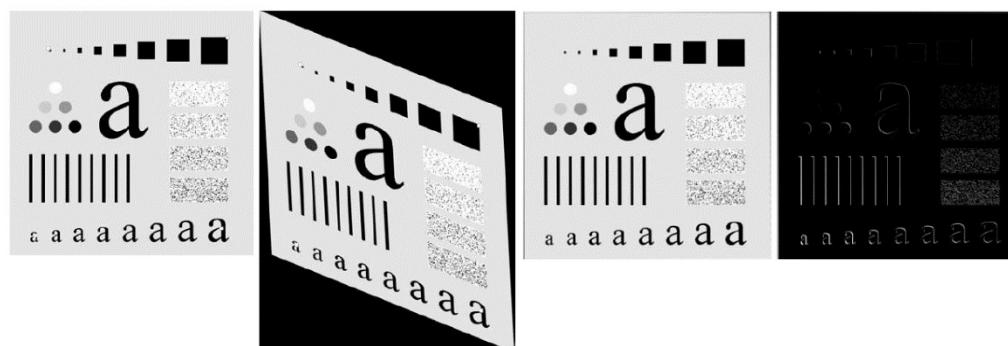


FIGURE 2.37
Image registration.
(a) Reference image.
(b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners.
(c) Registered image (note the errors in the border).
(d) Difference between (a) and (c), showing more registration errors.

7. Mathematical Tools

• Vector and Matrix Operations

- Color images are formed in RGB color space by using red, green, and blue component images.
- Each pixel has three components, which can be organized in the form of a column vector.

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

- Thus, an RGB color image of size MxN can be represented by three component images of this size, or by a total of MN 3-D vectors.
- Once pixels have been represented as vectors we have at our disposal the tools of vector-matrix theory.

7. Mathematical Tools

• Image Transforms

- In some cases, image processing tasks are best formulated by transforming the input images, carrying the specified task in a transform domain, and applying the inverse transform to return to the spatial domain.
- A particularly important class of 2-D linear transforms, can be expressed in the general form.

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)r(x, y, u, v)$$

where f is the input image, r is called the forward transformation kernel, and the equation is evaluated for and $u = 0, 1, \dots, M-1$ and $v = 0, 1, \dots, N-1$.

7. Mathematical Tools

• Image Transforms

- T is called the *forward transform* of f .
- Given T we can recover f using the *inverse transform*.

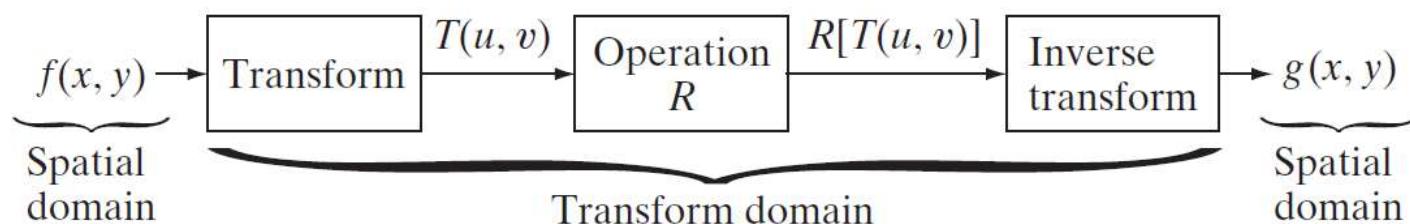
$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v)s(x, y, u, v)$$

- The equation is evaluated for $x = 0, 1, \dots, M-1$ and $y = 0, 1, \dots, N-1$.
- s is called the inverse transformation kernel.

7. Mathematical Tools

• Image Transforms

- General approach for operating in the linear transform domain.



7. Mathematical Tools

• Image Transforms

- General approach for operating in the linear transform domain.

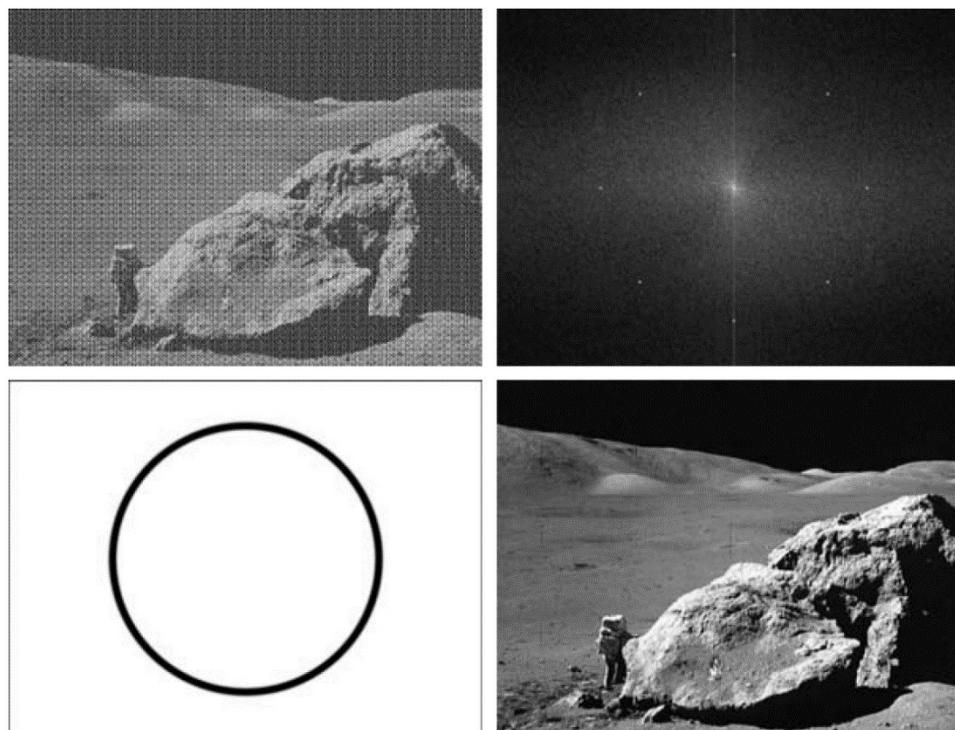


FIGURE 2.40

(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.)

7. Mathematical Tools

• Probabilistic Methods

- We may treat intensity values as random quantities.

✓ Probability: $p(z_k) = \frac{n_k}{MN}$

✓ Mean: $m = \sum_{k=0}^{L-1} z_k p(z_k)$

✓ Variance: $\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k)$

- We use these statistical quantities to derive intensity enhancement algorithms.

MATLAB Tips and Examples

- **Useful functions**

```
clear all  
close all  
clc  
I = imread('cameraman.tif');  
[X, map] = gray2ind(I, 16);  
II = ind2gray(X, map);  
imshow(II);  
imwrite()  
imresize()  
rgb2ycbcr()
```

Suggested reading: Gonzales & Woods DIP - Chapter 2

Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 03

Intensity Transformation and Spatial Filtering

1. Introduction

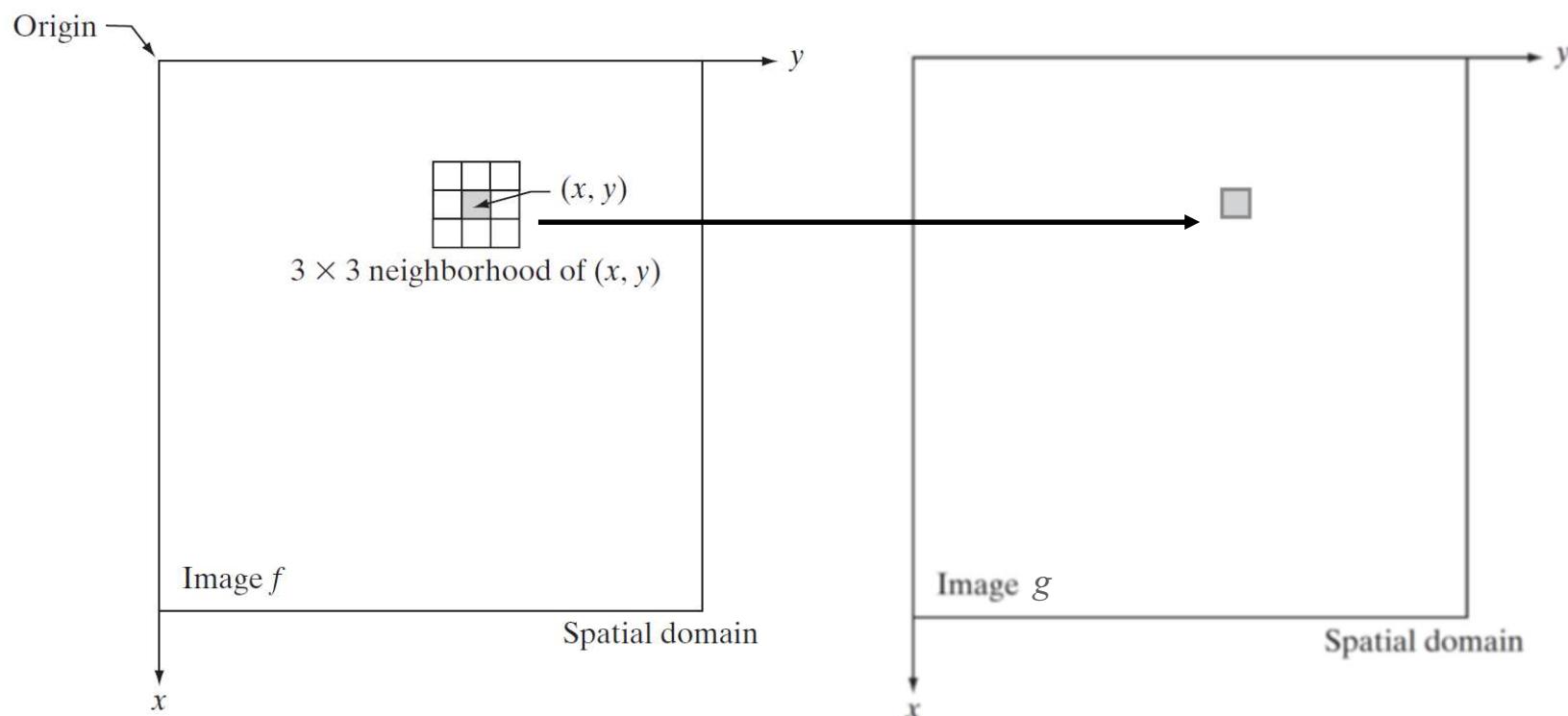
- All the image processing techniques discussed in this topic are implemented in the spatial domain, which is simply the plane containing the pixels of an image.
- Later we will discuss techniques implemented in the frequency domain.
- The spatial domain processes we discuss in this chapter can be denoted by the expression:

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

where $f(x, y)$ is the input image, $g(x, y)$ is the output image, and T is an operator on f defined over a neighborhood of point (x, y) .

1. Introduction

- A 3×3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

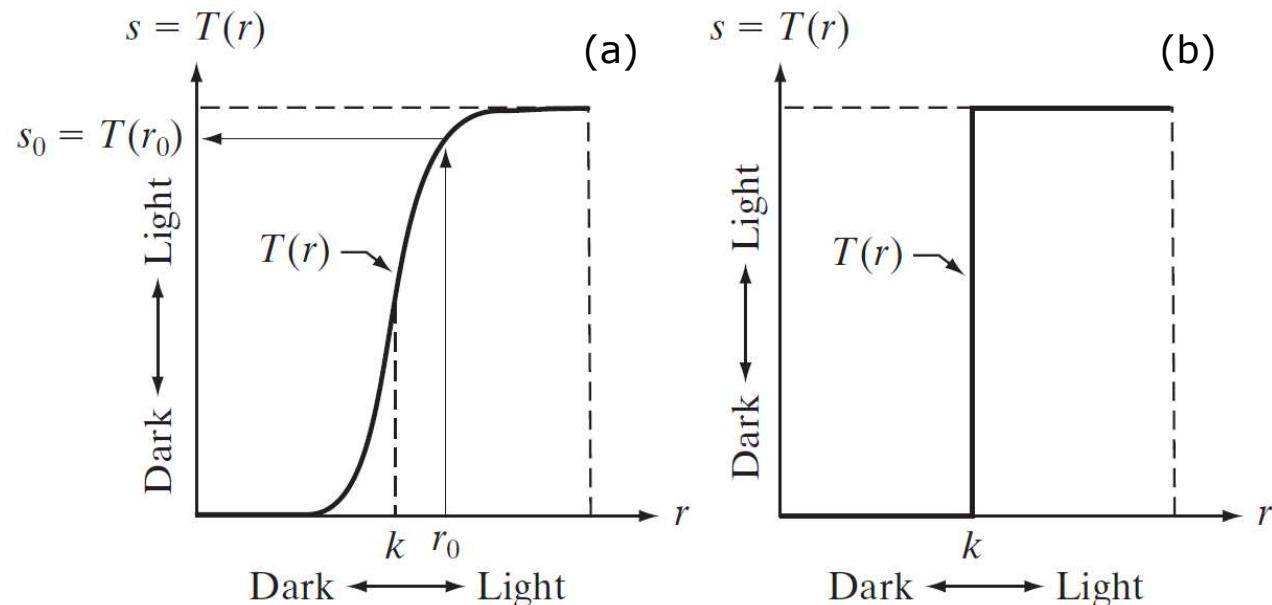


1. Introduction

- The smallest possible neighborhood is of size 1×1 . In this case, T becomes an intensity transformation function of the form,

$$s = T(r)$$

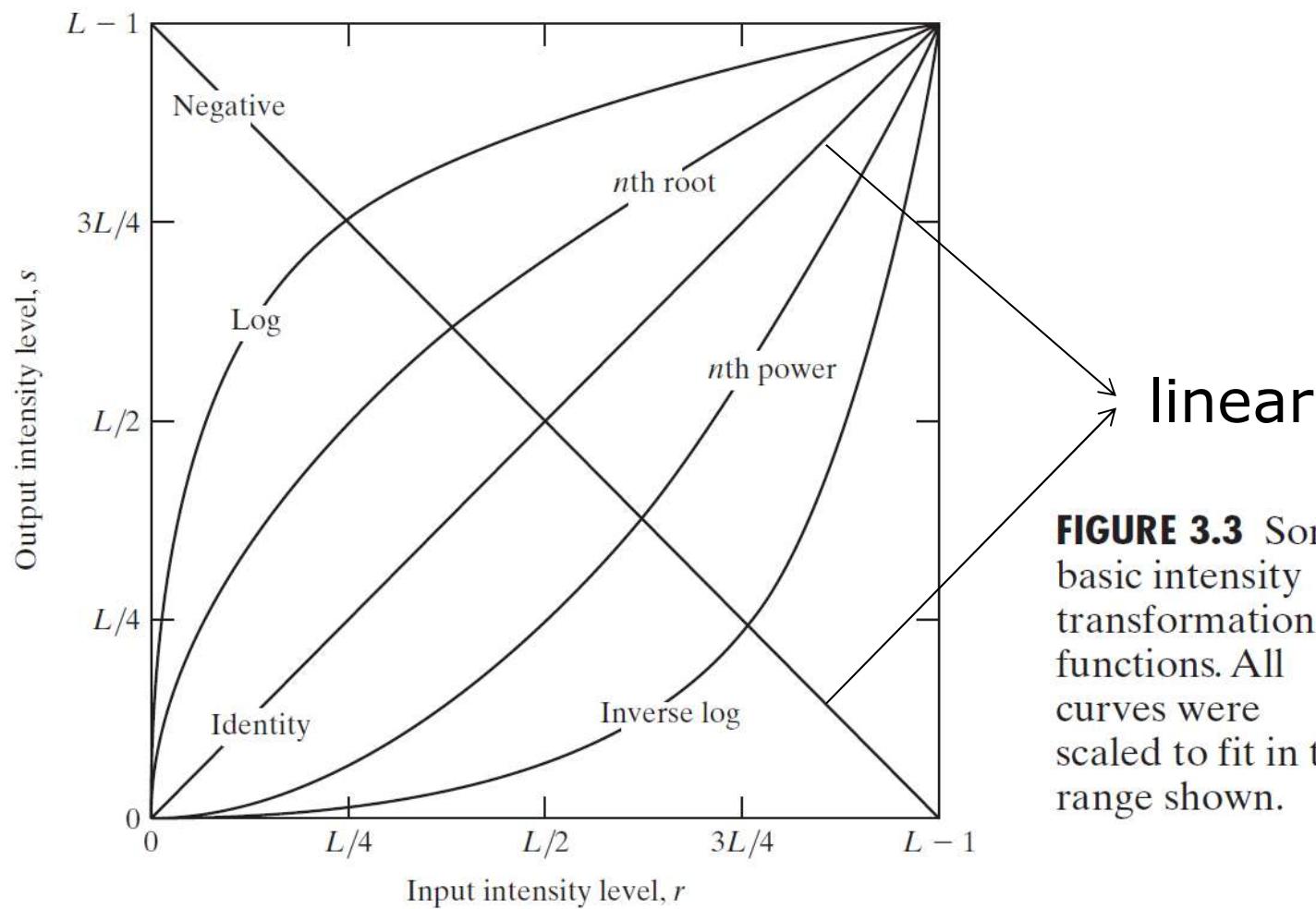
- Examples: *contrast stretching* e *thresholding*.



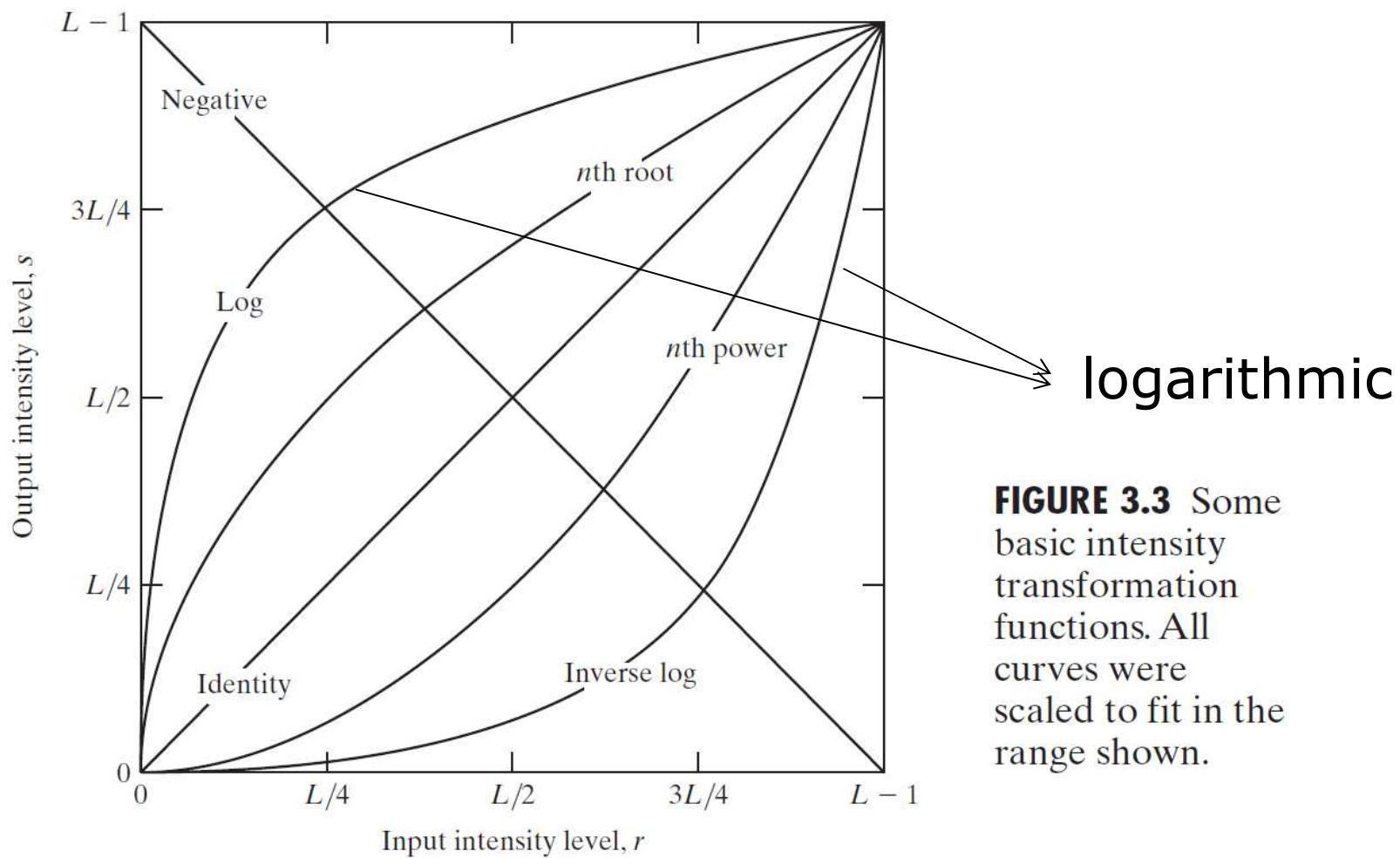
a b

FIGURE 3.2
Intensity transformation functions.
(a) Contrast-stretching function.
(b) Thresholding function.

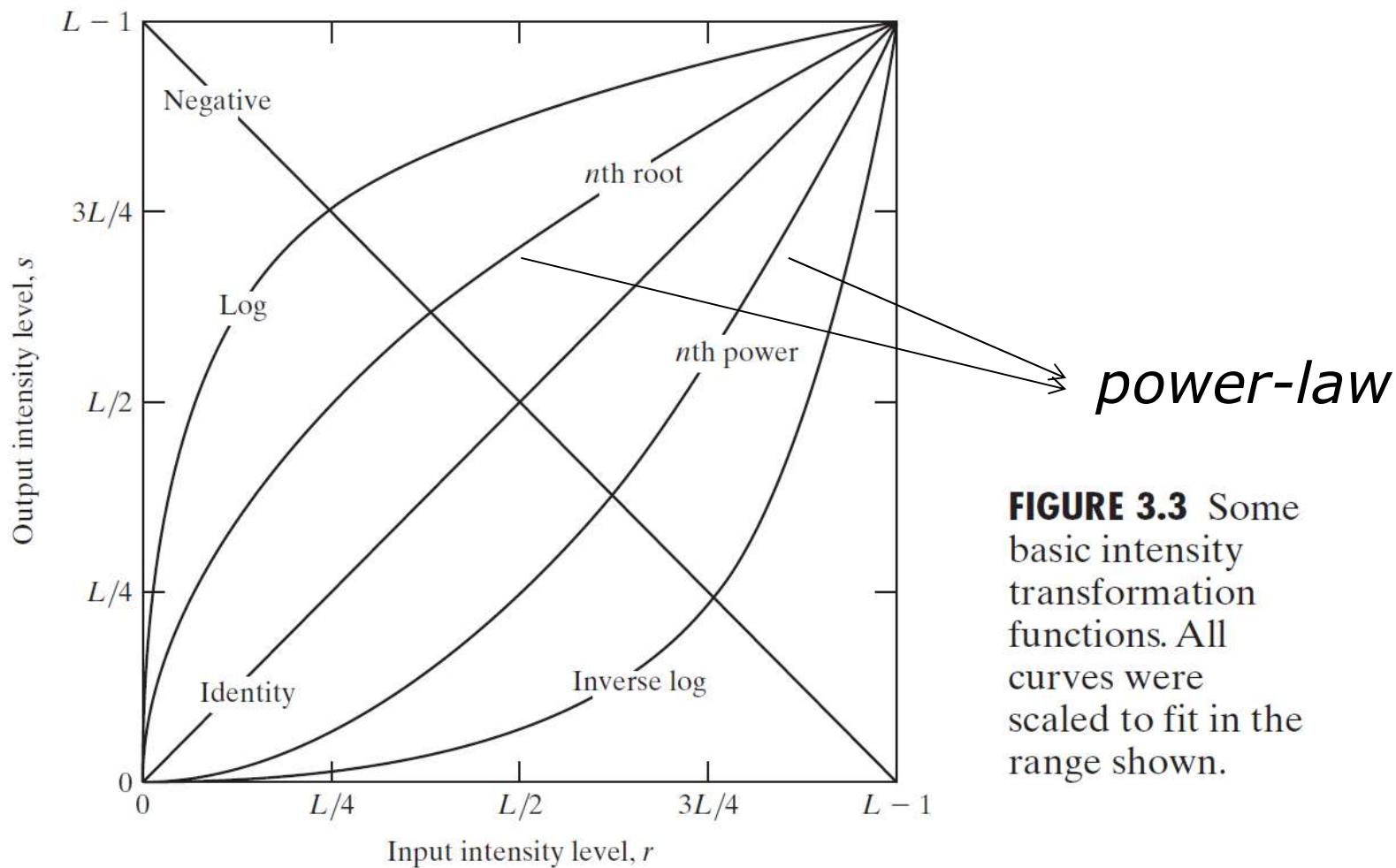
2. Some Basic Intensity Transformation Functions



2. Some Basic Intensity Transformation Functions

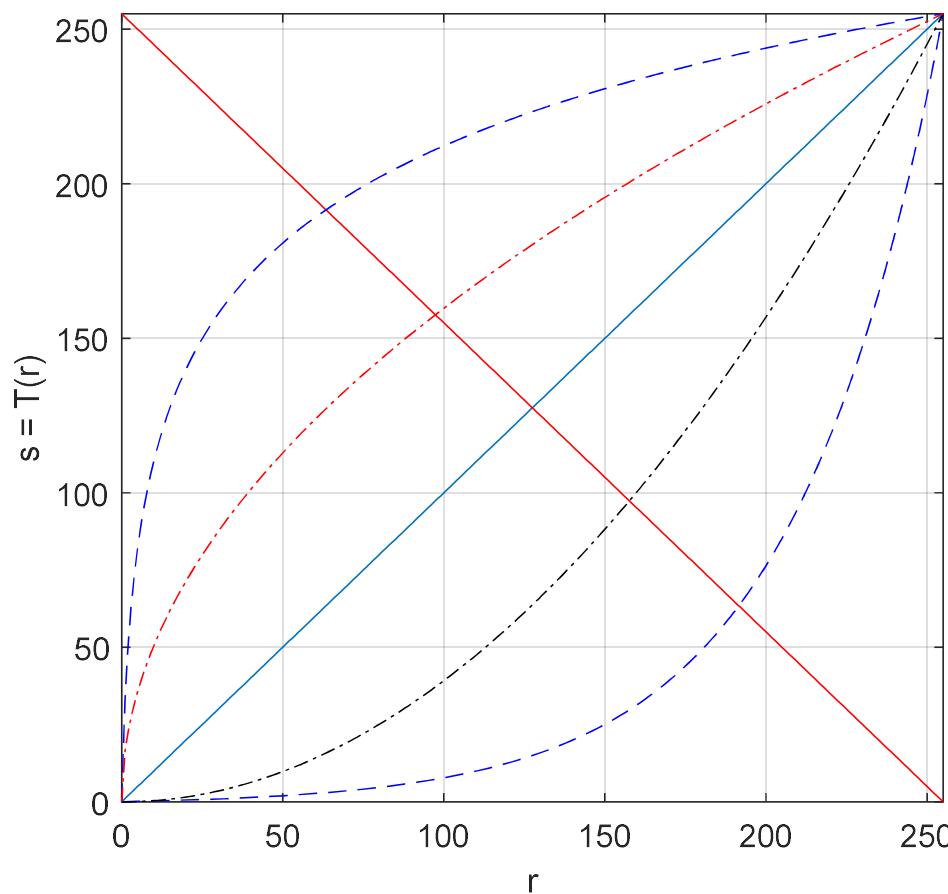


2. Some Basic Intensity Transformation Functions



2. Some Basic Intensity Transformation Functions

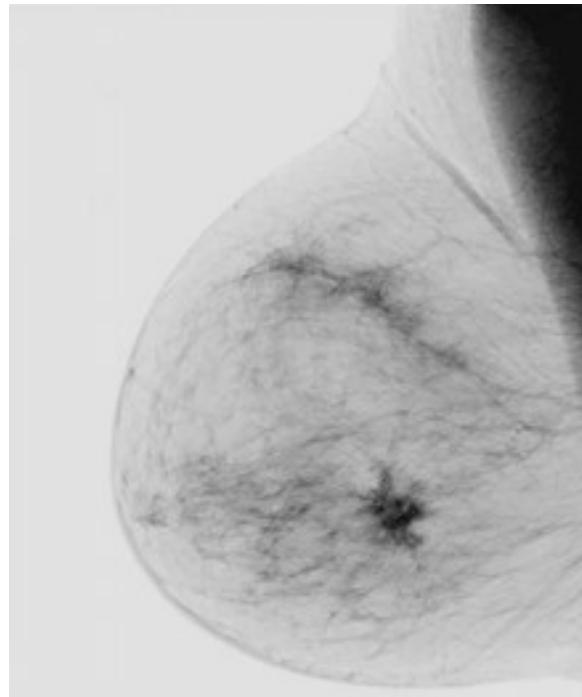
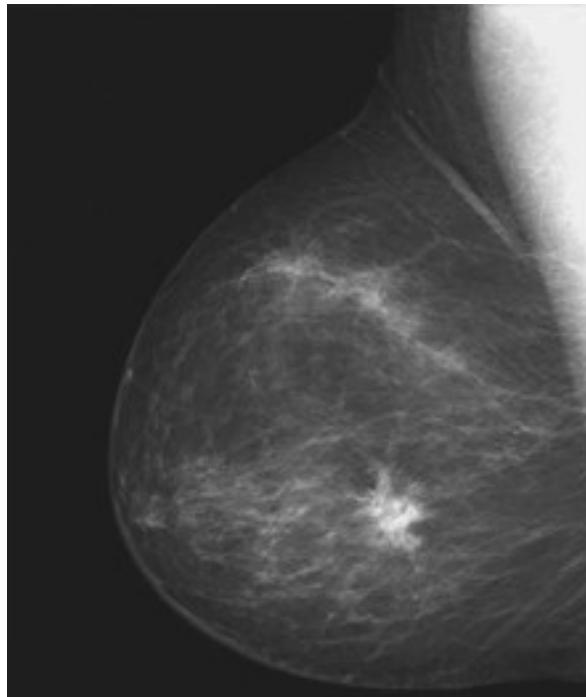
- MATLAB: s09intensity_transformations.m



2. Some Basic Intensity Transformation Functions

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

$$s = L - 1 - r$$



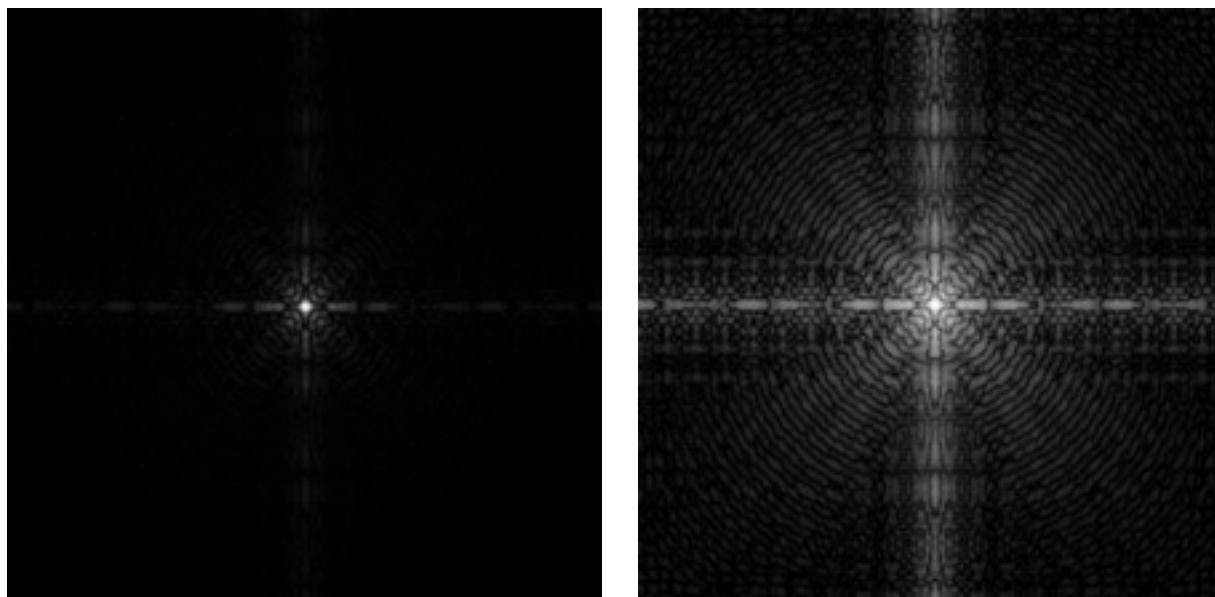
a b

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.)

2. Some Basic Intensity Transformation Functions

- **Log Transformations:** We use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values. The opposite is true of the inverse log transformation.

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



a | b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation in Eq. (3.2-2) with $c = 1$.

2. Some Basic Intensity Transformation Functions

- **Power-Law (Gamma) Transformations:** $S = Cr^\gamma$

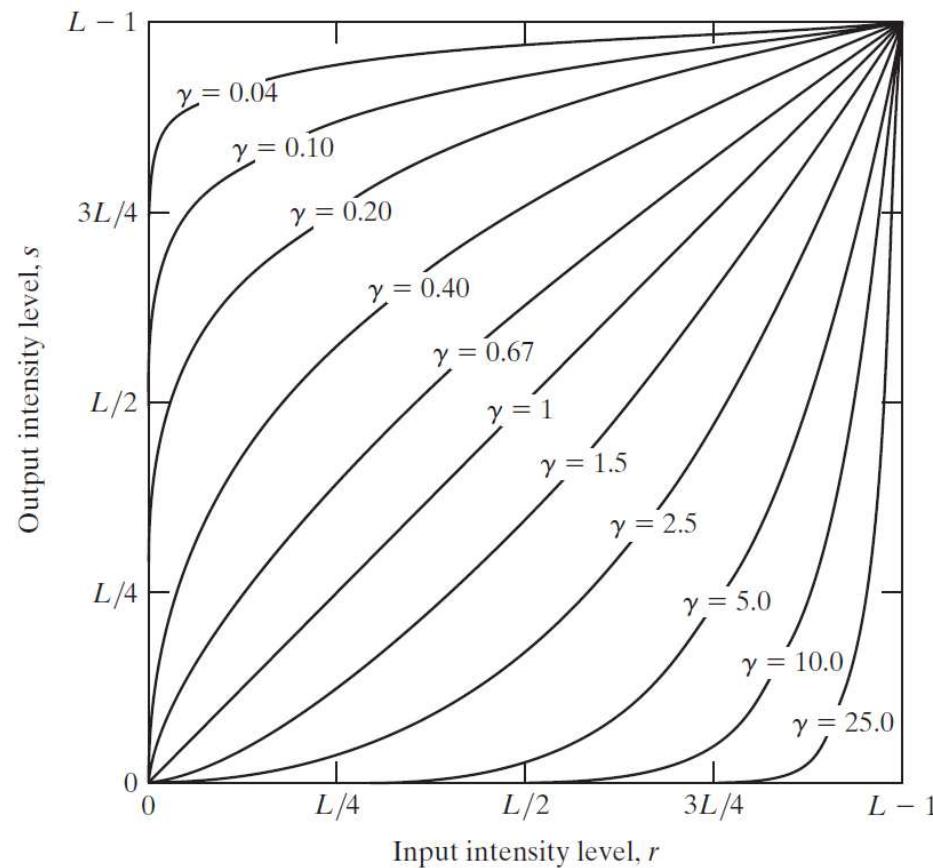


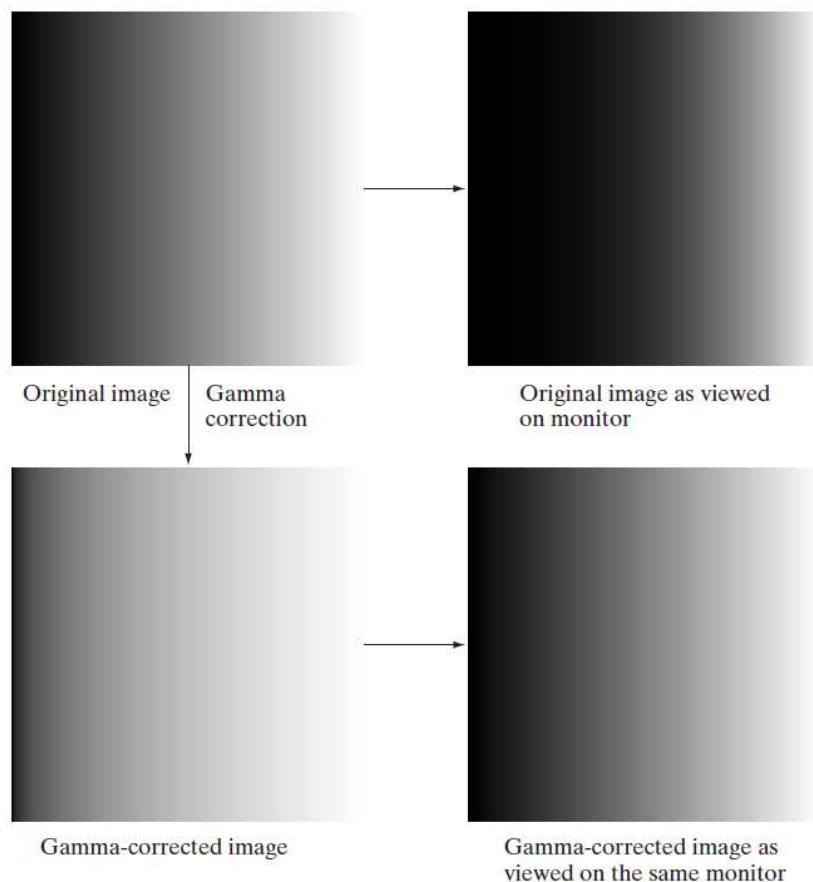
FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases). All curves were scaled to fit in the range shown.

2. Some Basic Intensity Transformation Functions

- A variety of **devices** used for image capture, printing, and display **respond according to a power-law**.
- By convention, the exponent in the power-law equation is referred to as **gamma**.
- The process used to correct these power-law response phenomena is called **gamma correction**.
- Gamma correction is important if displaying an image accurately on a computer screen is of concern.

2. Some Basic Intensity Transformation Functions

- Example of gamma correction:



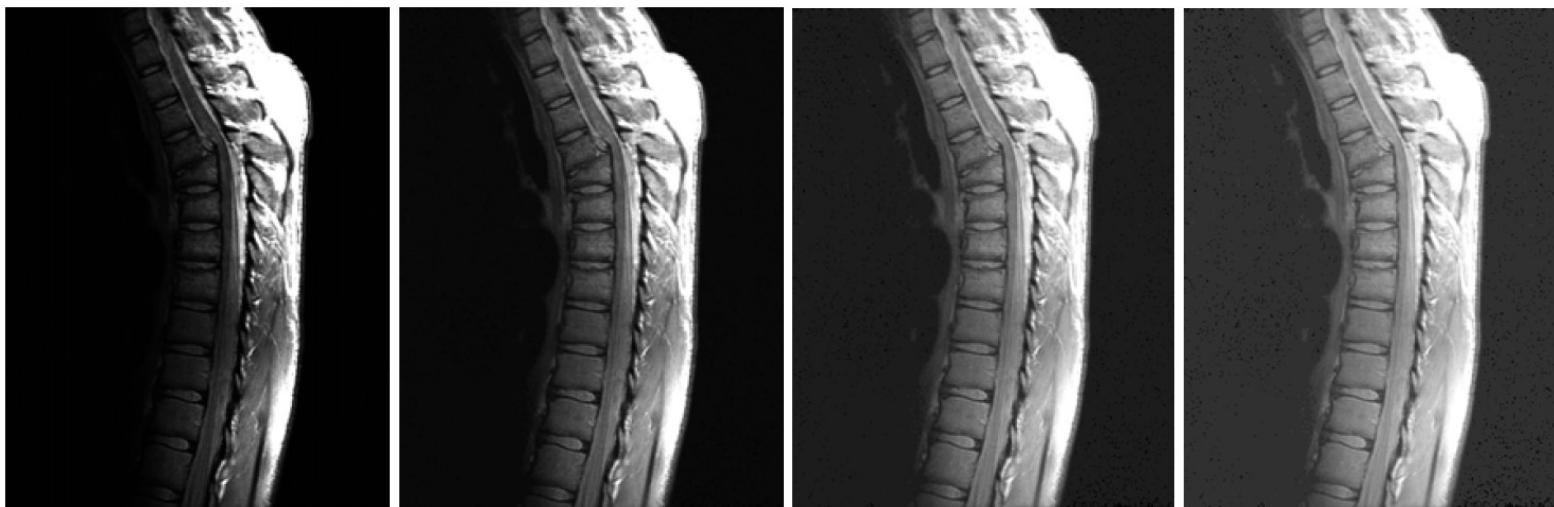
a
b
c
d

FIGURE 3.7

(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).

2. Some Basic Intensity Transformation Functions

- Power-law general-purpose contrast manipulation:



a | b | c | d

FIGURE 3.8
(a) Magnetic resonance image (MRI) 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 courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

2. Some Basic Intensity Transformation Functions

- Power-law general-purpose contrast manipulation:



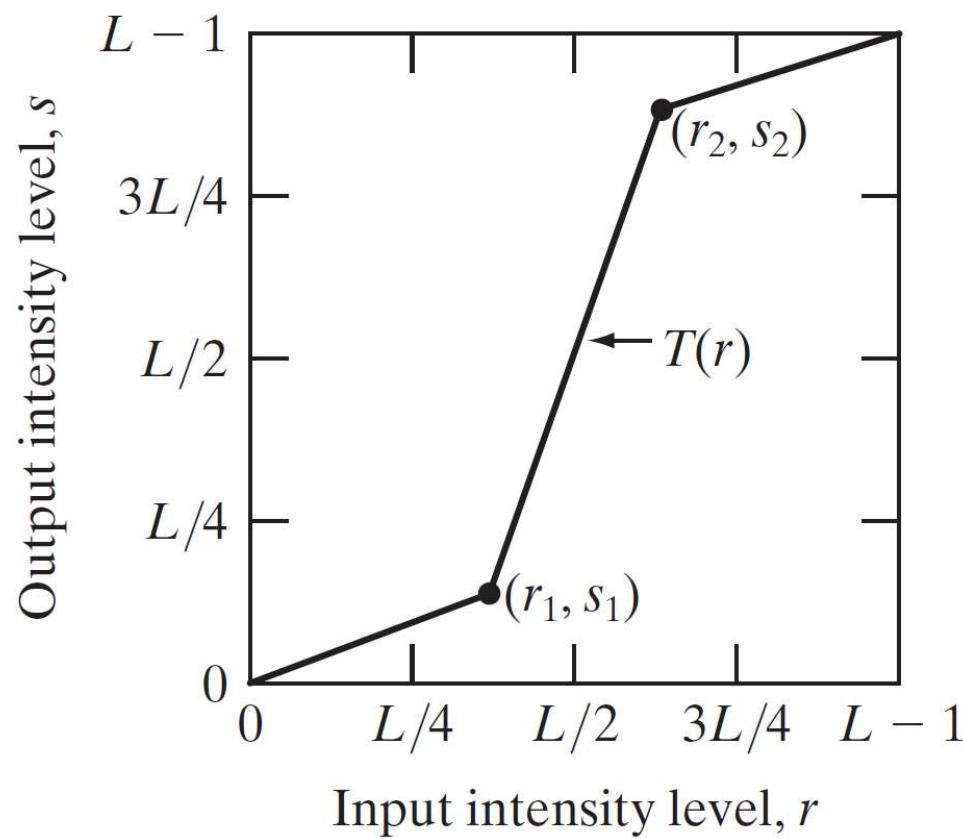
a b
c d

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.)

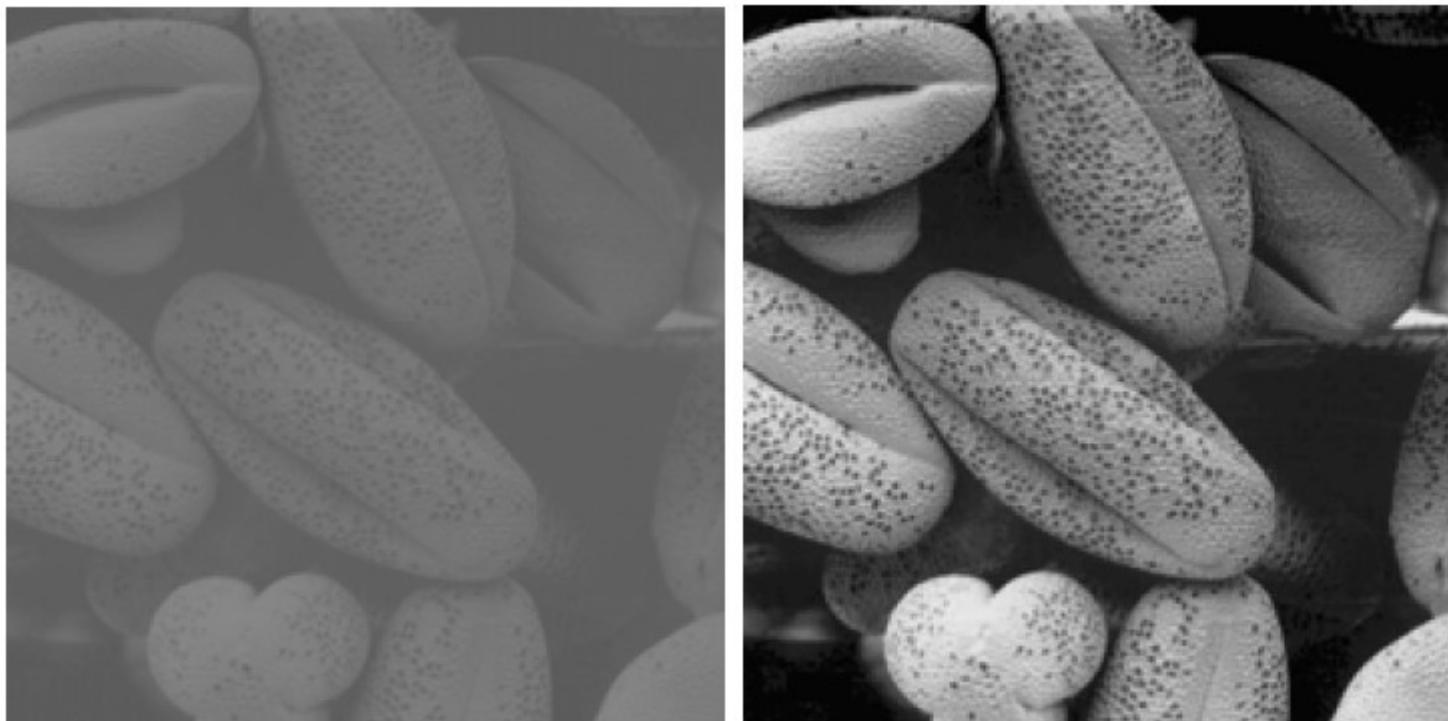
2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **contrast stretching**



2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **contrast stretching**



2. Some Basic Intensity Transformation Functions

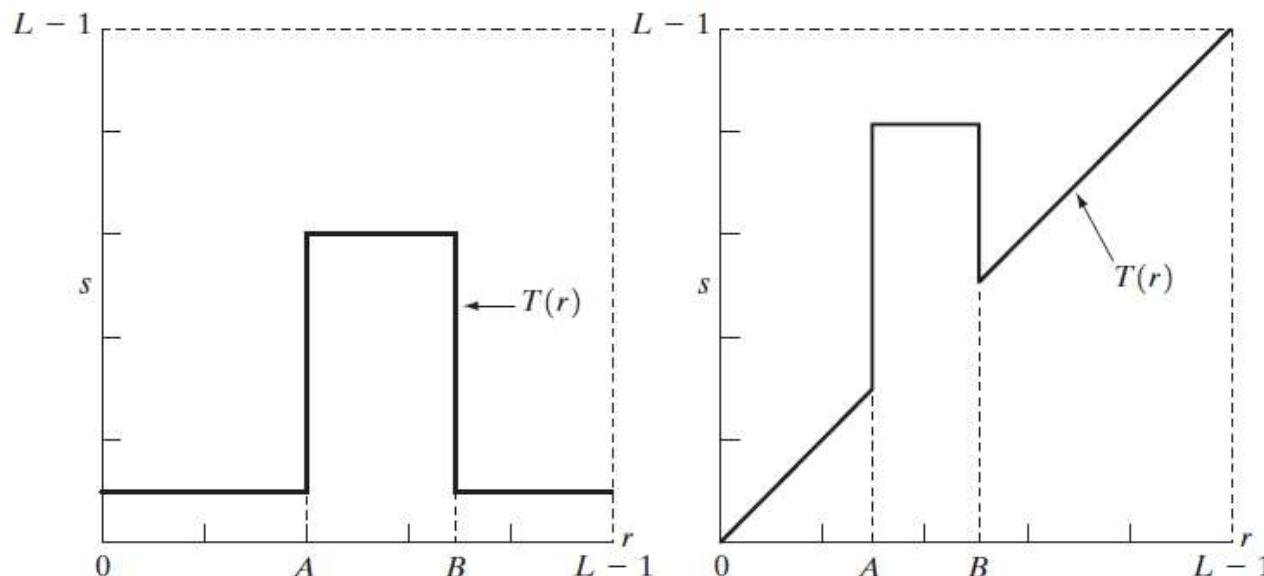
- Piecewise-Linear Transformation Functions:
thresholding

$$s = \begin{cases} 0, & \text{se } r \leq m \\ 255, & \text{se } r > m \end{cases}$$



2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **Intensity-level slicing**

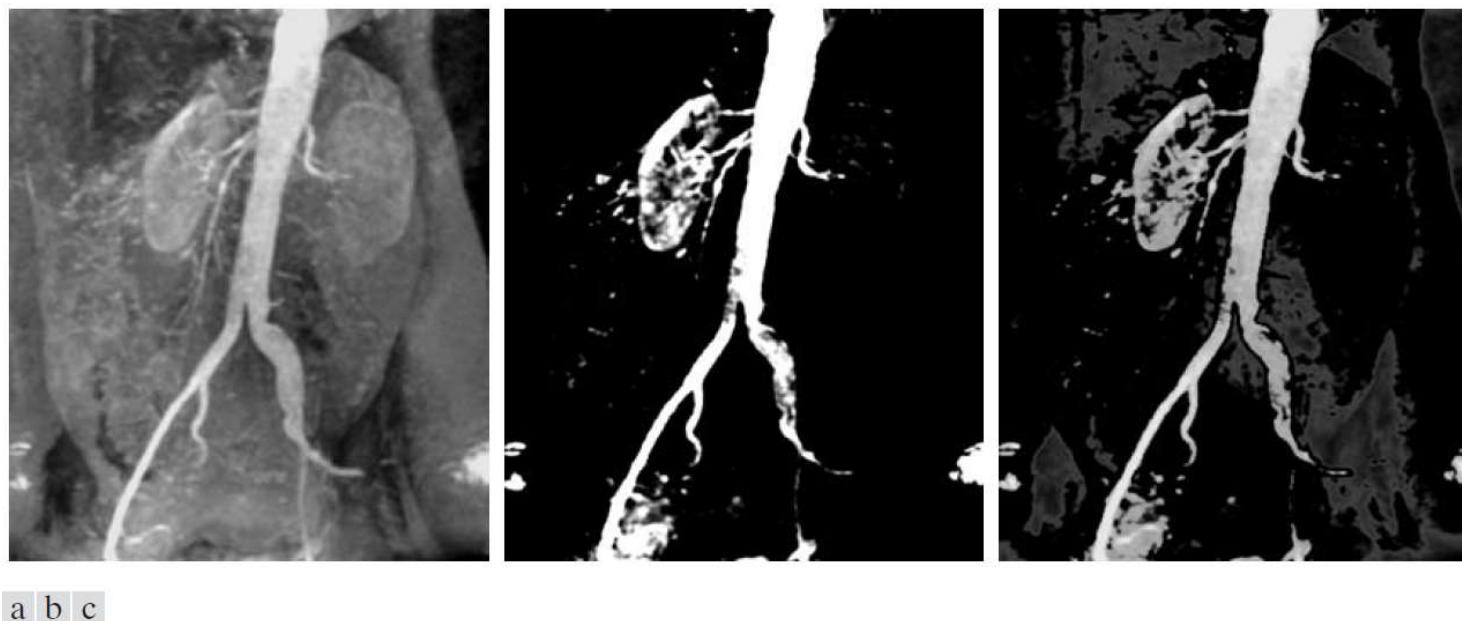


a | b

FIGURE 3.11 (a) This transformation highlights intensity range $[A, B]$ and reduces all other intensities to a lower level. (b) This transformation highlights range $[A, B]$ and preserves all other intensity levels.

2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **Intensity-level slicing**

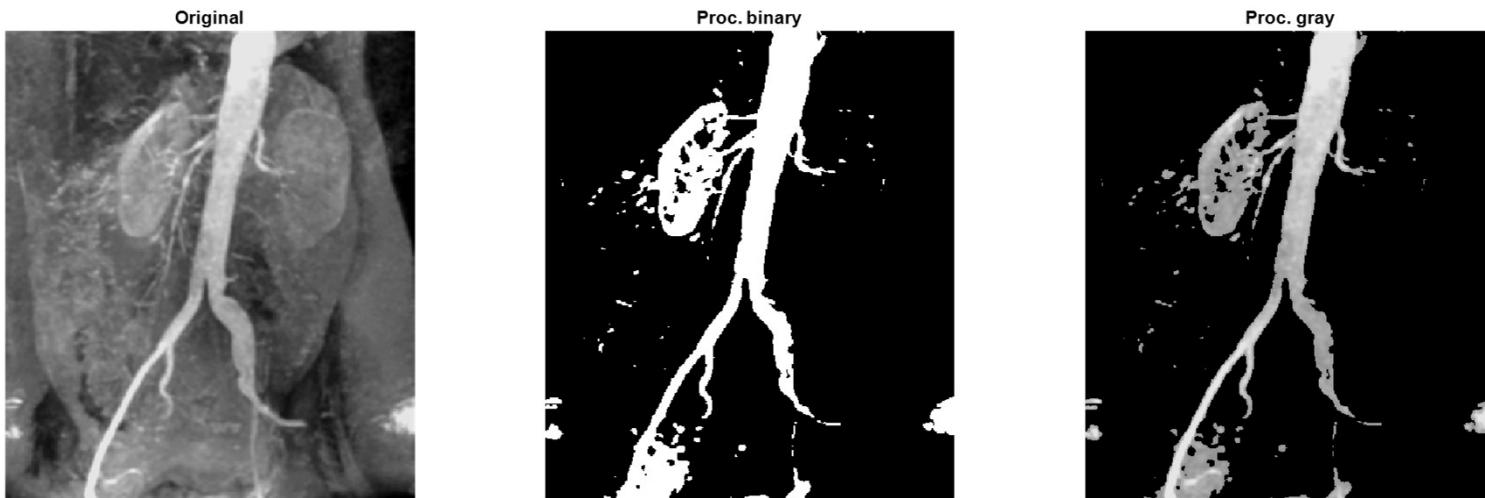


a b c

FIGURE 3.12 (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected area set to black, so that grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)

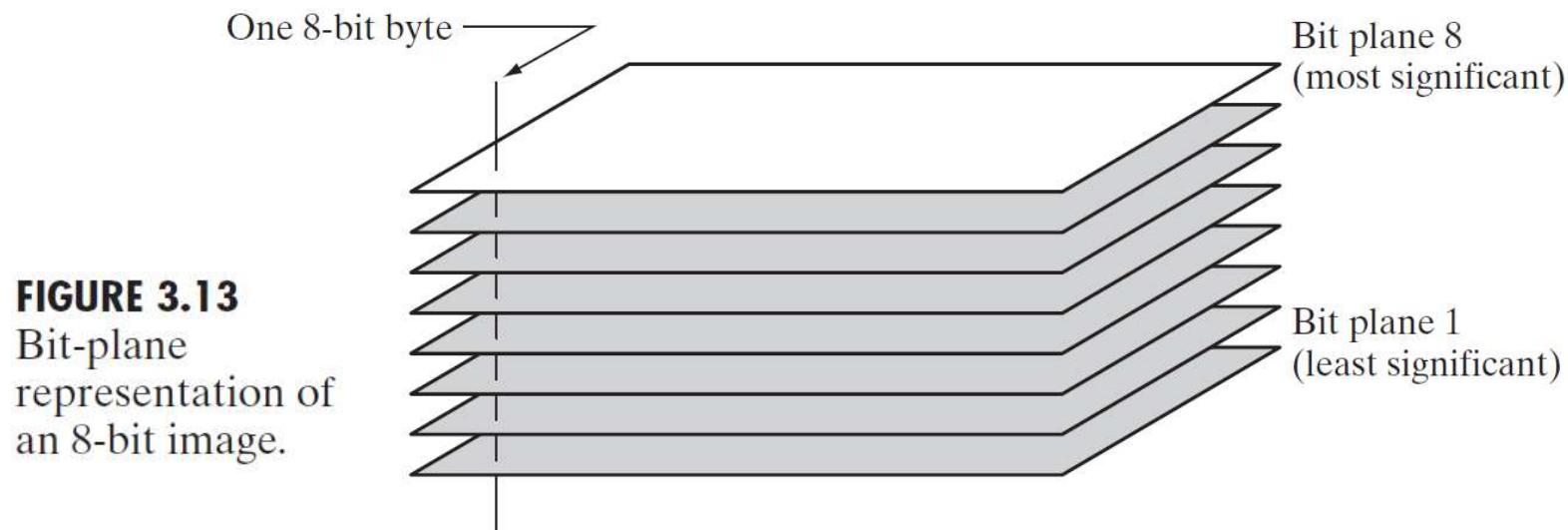
2. Some Basic Intensity Transformation Functions

- **MATLAB:** s24kidney.m



2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **bit-plane slicing**.
- Instead of highlighting **intensity-level ranges**, we could highlight the contribution made to total image appearance by **specific bits**.



2. Some Basic Intensity Transformation Functions

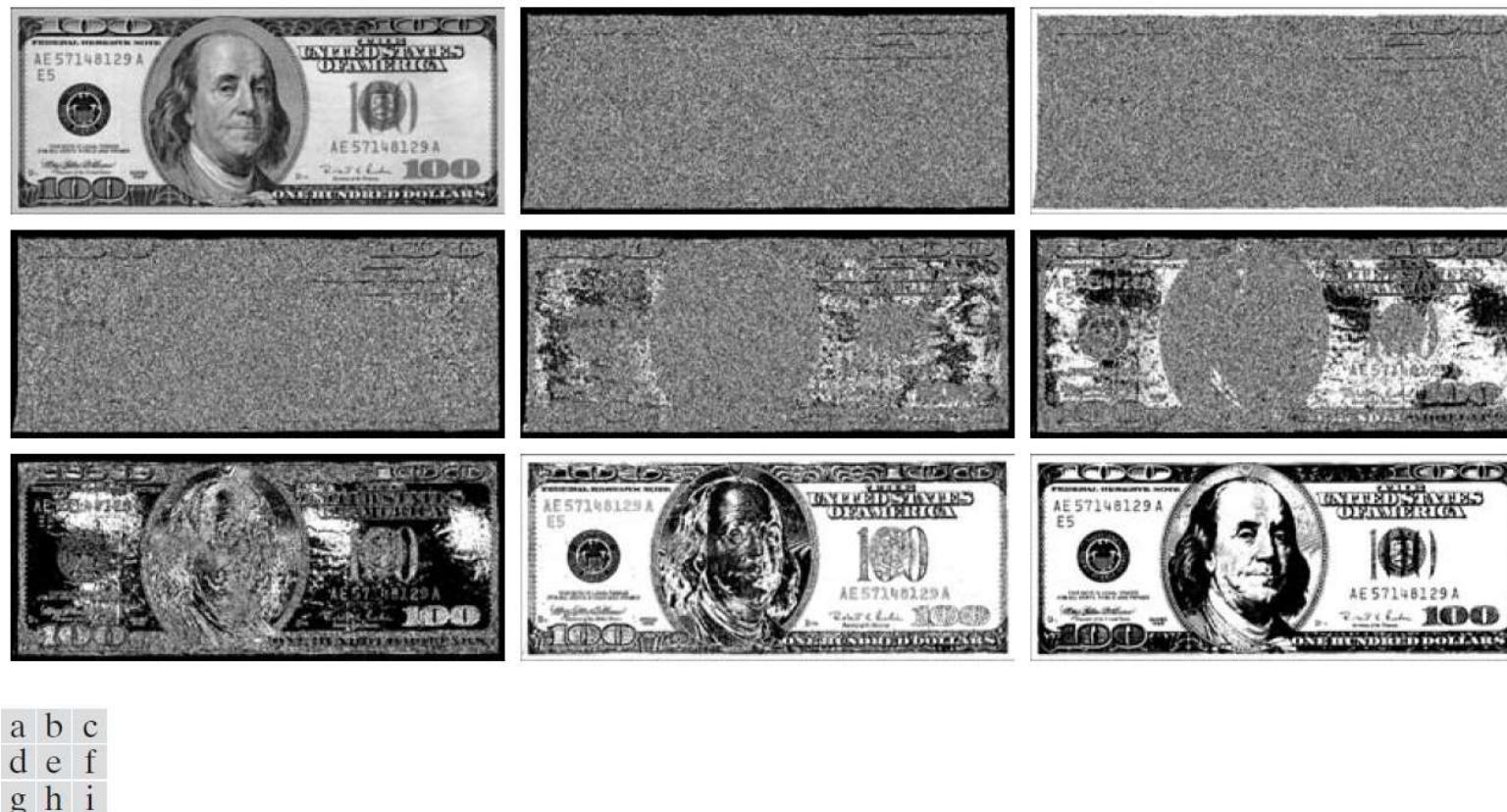
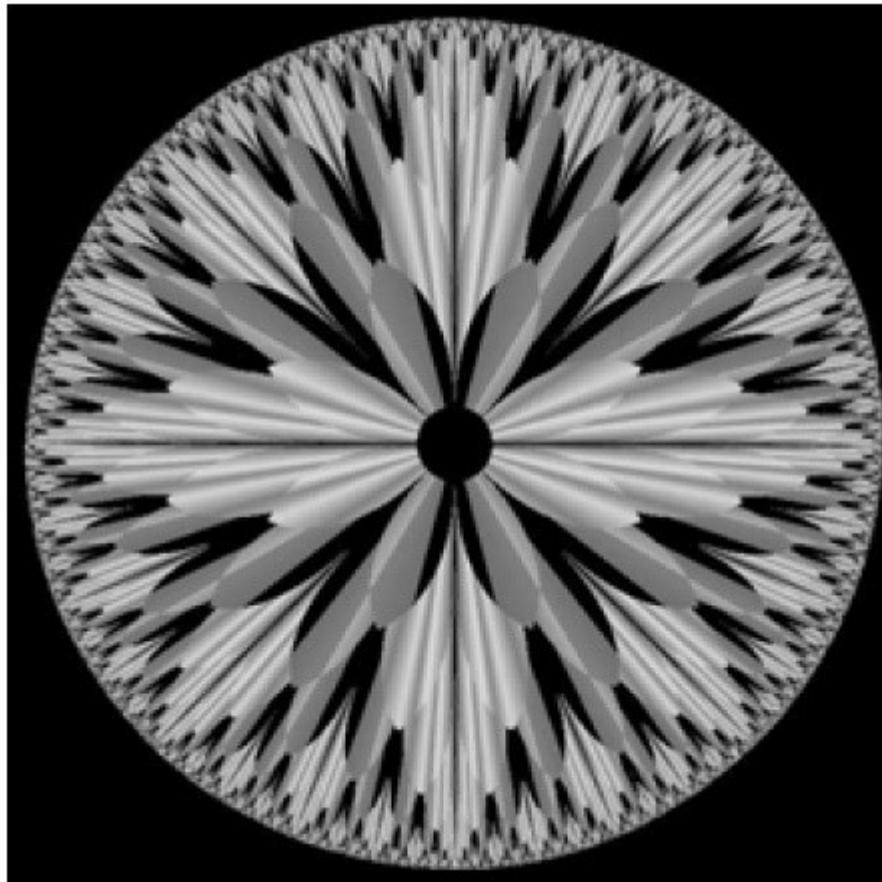


FIGURE 3.14 (a) An 8-bit gray-scale image of size 500×1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

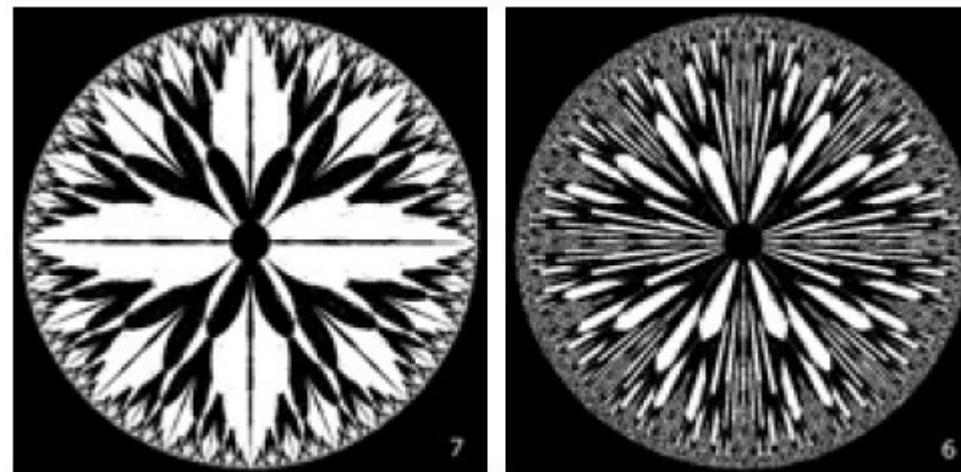
2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **bit-plane slicing.**



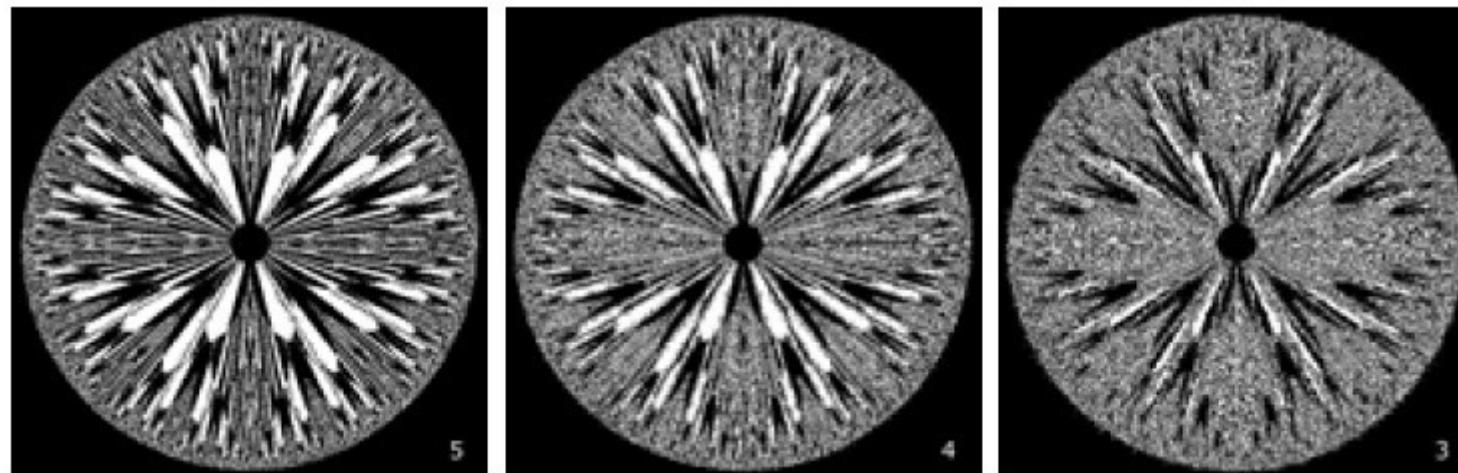
2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **bit-plane slicing.**



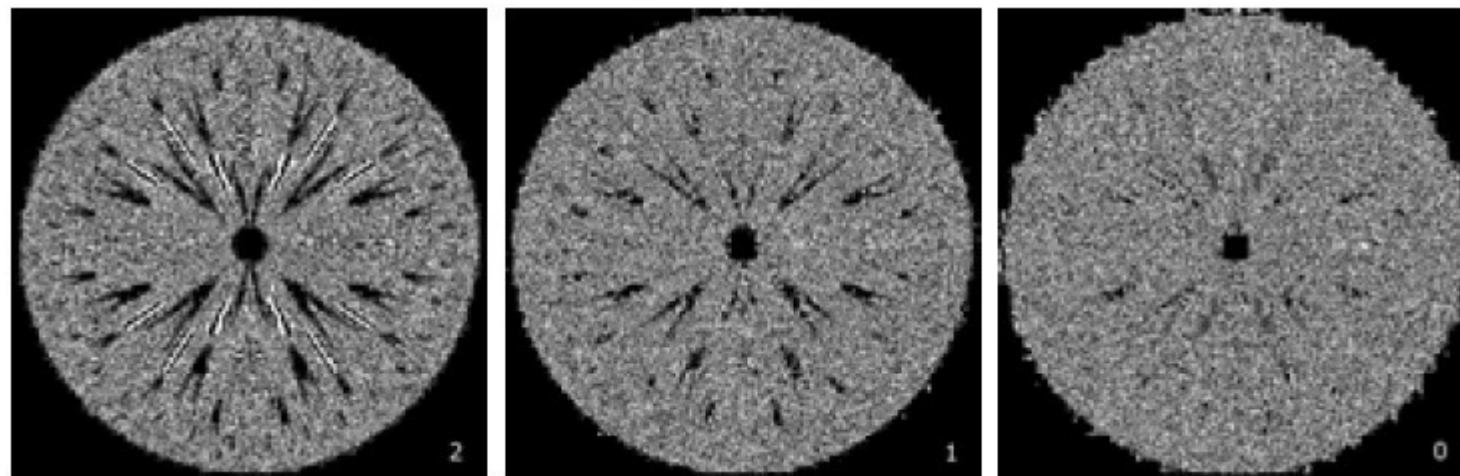
2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **bit-plane slicing.**



2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **bit-plane slicing.**



2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **bit-plane slicing**.
- Decomposing an image into its bit planes is useful for analyzing the relative **importance of each bit-plane** in the image.
- This type of decomposition is useful for image compression, in which fewer than all planes are used in reconstructing an image.



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).

2. Some Basic Intensity Transformation Functions

- Piecewise-Linear Transformation Functions: **bit-plane slicing.**



3. Histogram Processing

- The histogram of a digital image with intensity levels in the range [0 L-1] is a discrete function,

$$h(r_k) = n_k,$$

where r_k is the k th intensity value and n_k the number of pixels in the image with intensity r_k .

- It is common practice to normalize a histogram by dividing each of its components by the total number of pixels N in the image,

$$p(r_k) = n_k / N.$$

- Loosely speaking, $p(r_k)$ is an estimate of the probability of occurrence of intensity level in an image.

3. Histogram Processing

- Example:

2	8	4	8	7	5	6	6
7	5	1	9	4	1	7	2
4	5	2	4	2	2	6	3
8	1	1	4	4	3	4	6
2	8	2	3	1	7	5	7
2	6	2	8	1	0	3	1
1	3	4	3	8	0	7	8
1	5	0	1	9	2	2	7

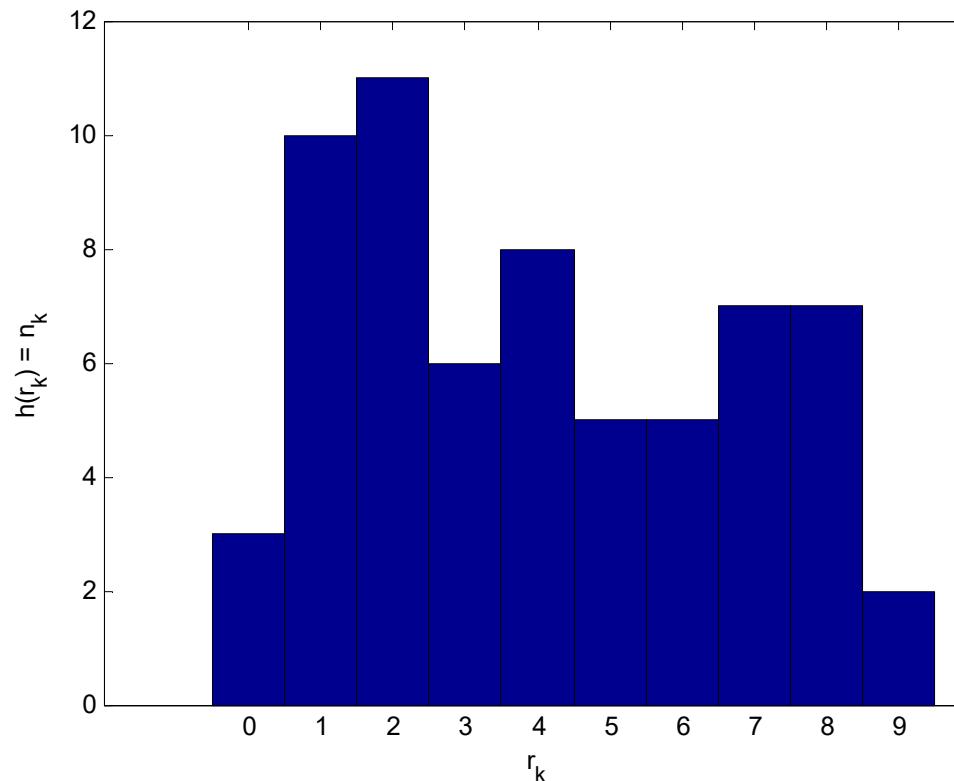
$$r_k = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

$$h(r_k) = 3, 10, 11, 6, 8, 5, 5, 7, 7, 2 = n_k$$

$$p(r_k) = n_k / 64$$

3. Histogram Processing

- Example:

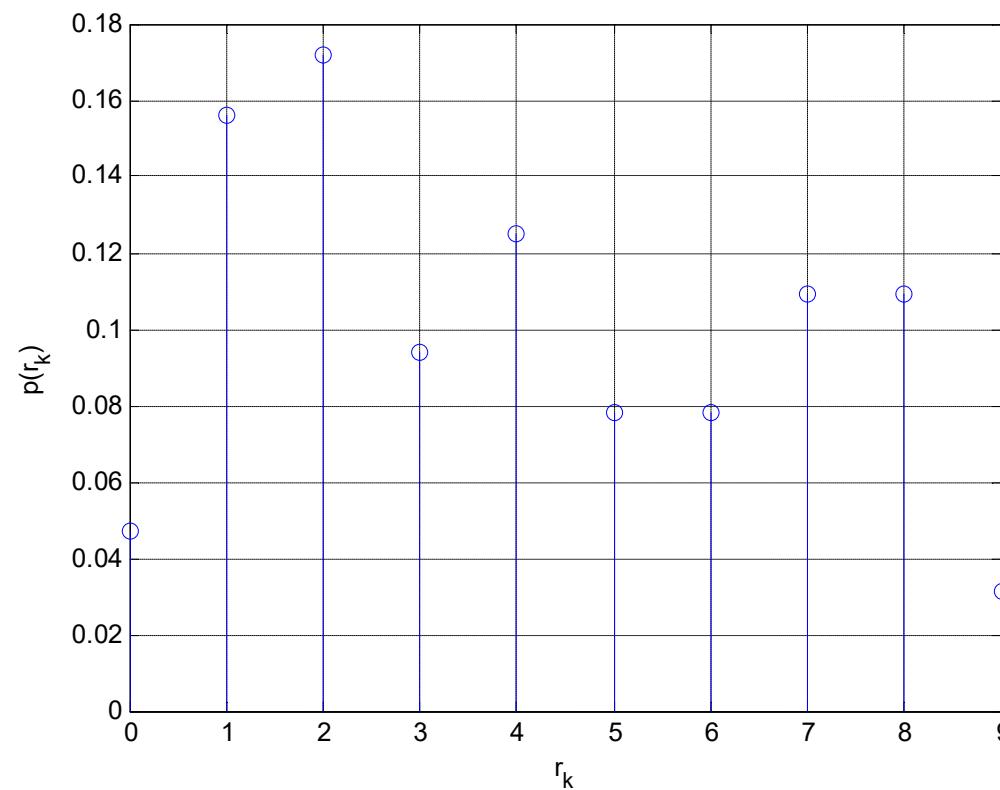


- MATLAB: `hist()`

3. Histogram Processing

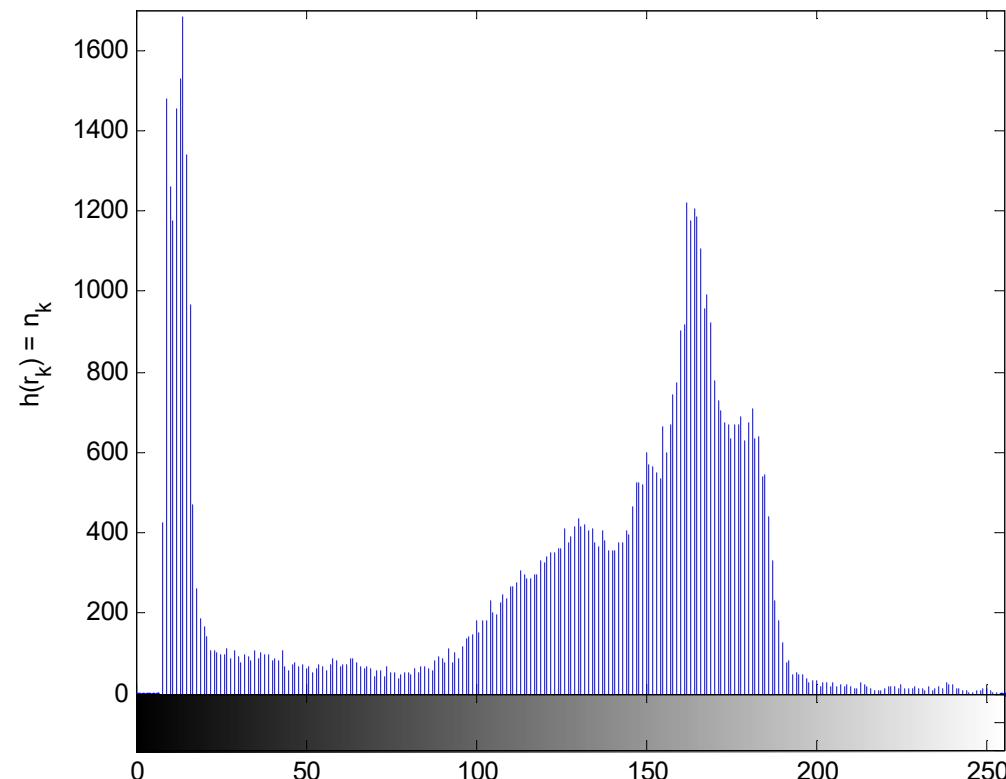
- Example:

$$p(r_k) = n_k / 64$$



3. Histogram Processing

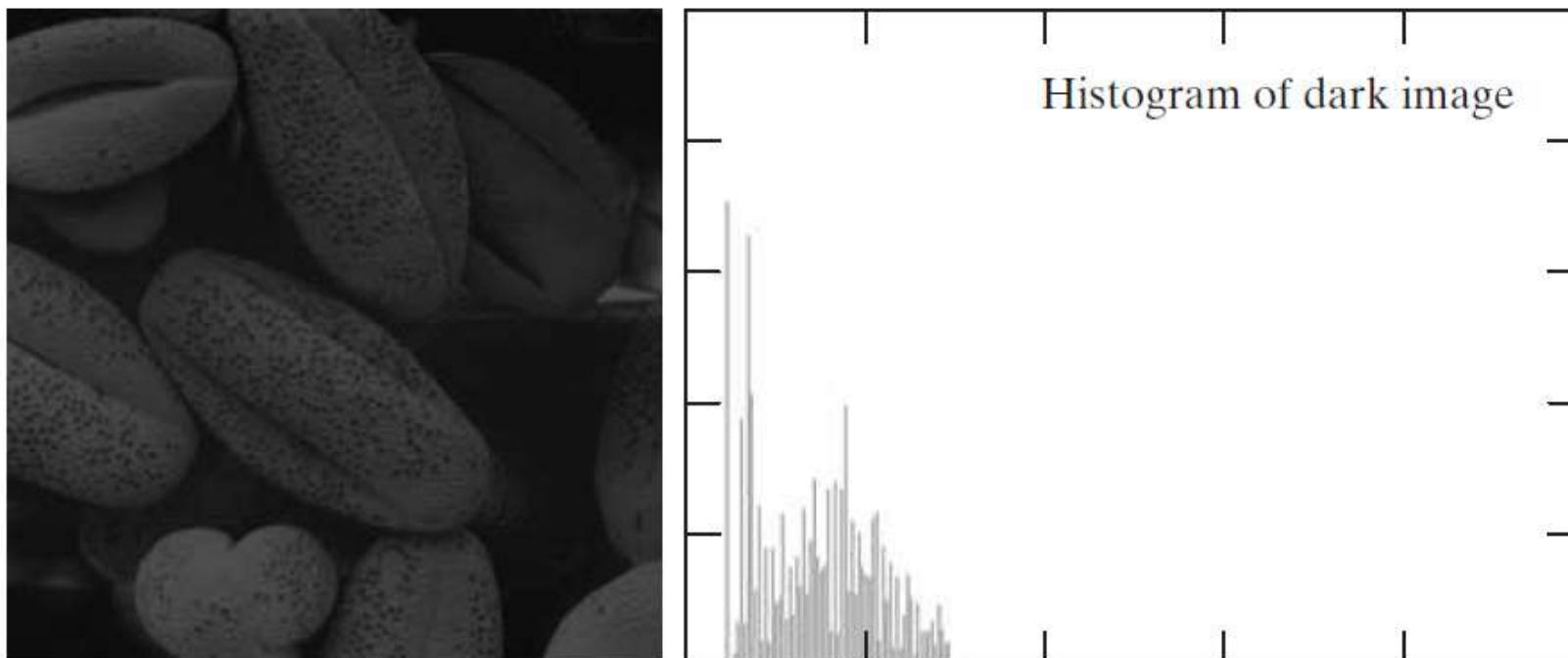
- Example:



- MATLAB imhist();

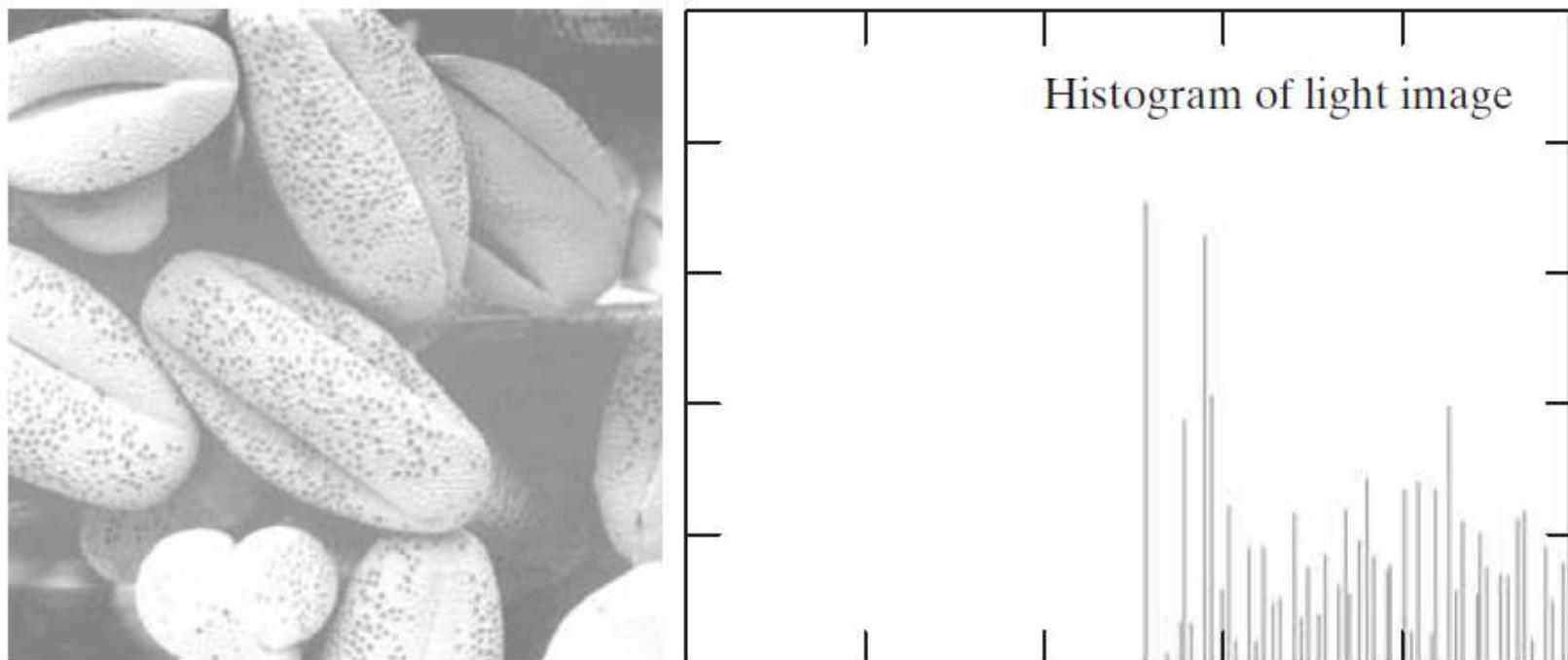
3. Histogram Processing

- Example:



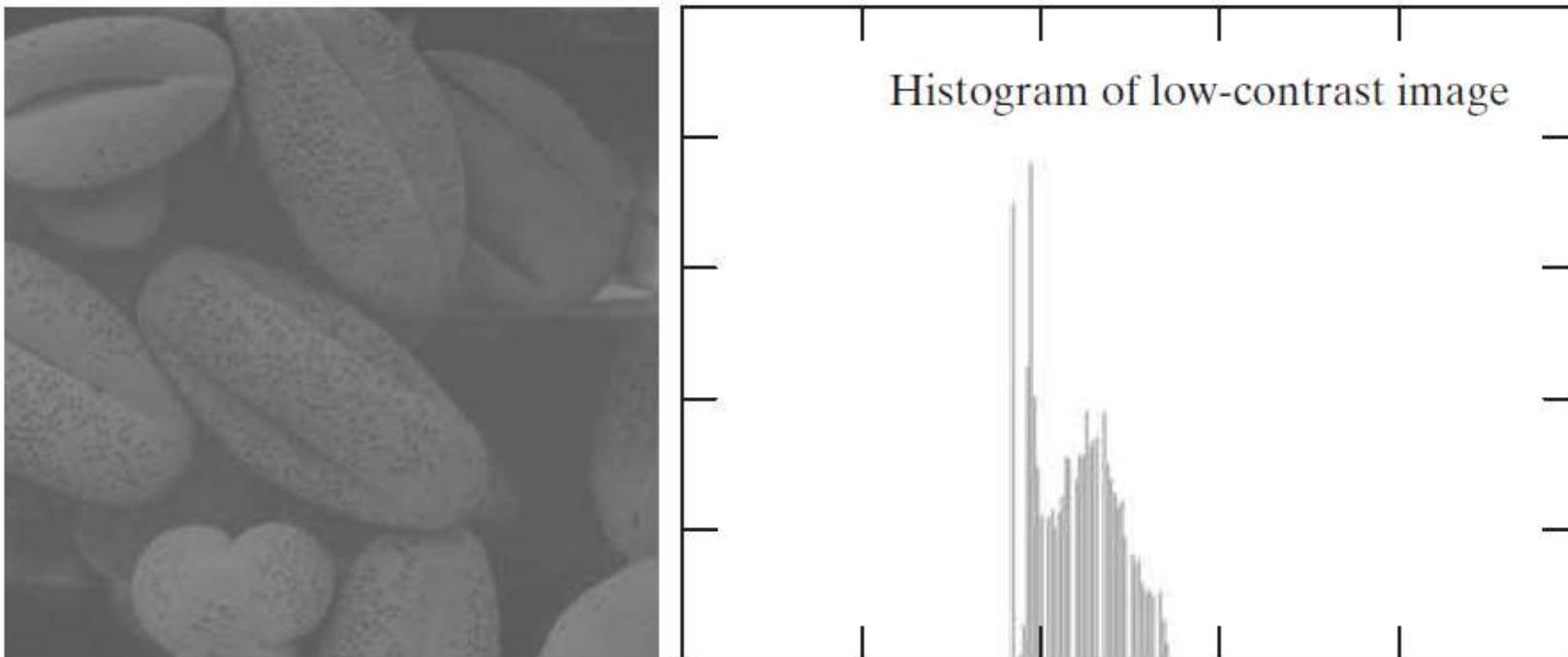
3. Histogram Processing

- Example:



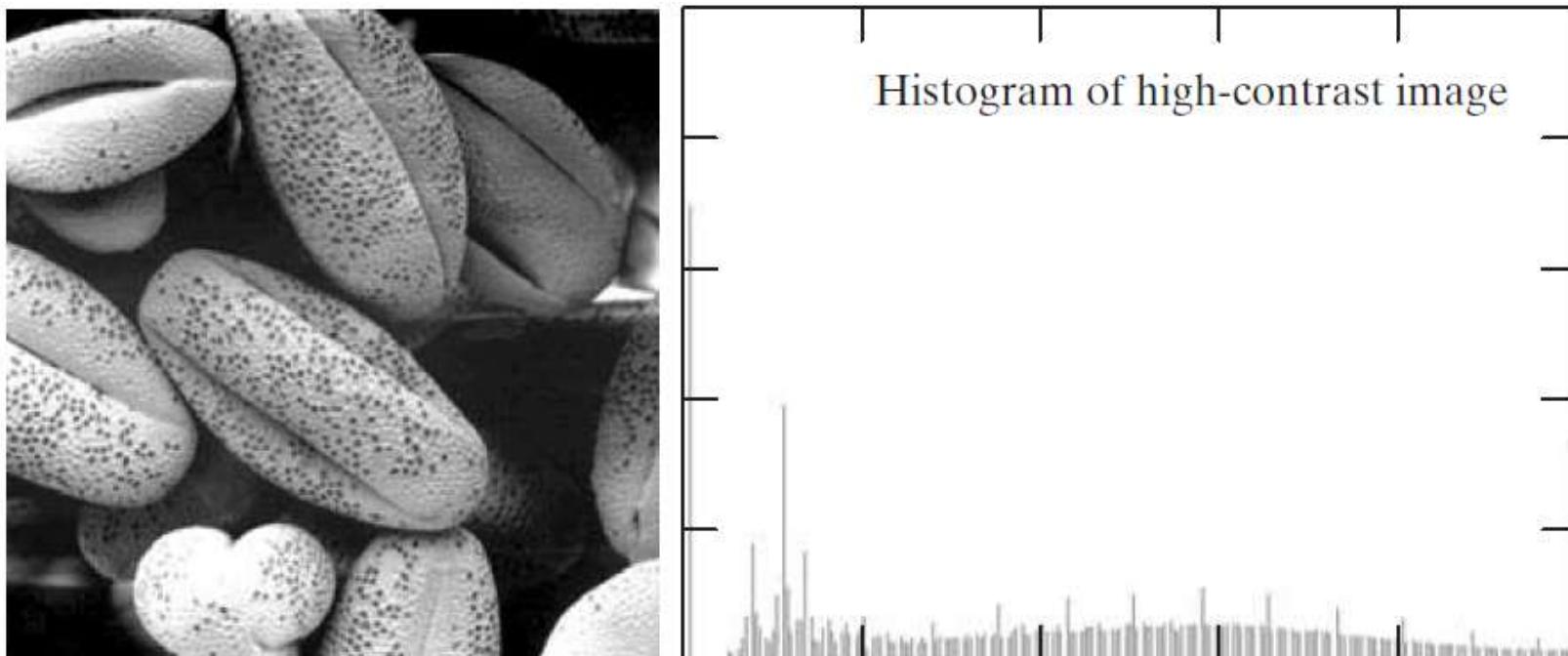
3. Histogram Processing

- Example:



3. Histogram Processing

- Example:



3. Histogram Processing

- Histogram Equalization:

- ✓ Let the variable r denote the intensities of an image to be processed.
- ✓ We assume that r is in the range $[0, L-1]$, with $r=0$ representing black and $r = L-1$ representing white.
- ✓ For r satisfying these conditions, we focus attention on transformations of the form

$$s = T(r) \quad 0 \leq r \leq L - 1$$

that produces an output intensity level s for every pixel in the input image having intensity r .

3. Histogram Processing

- Histogram Equalization

- ✓ We assume that,

- a) $T(r)$ is a monotonically increasing function in the interval $0 \leq r \leq L-1$ and

- b) $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$.

- ✓ In some formulations to be discussed later, we use the inverse

$$r = T^{-1}(s) \quad 0 \leq s \leq L - 1$$

- in which case we change condition a) to

- a') $T(r)$ is a strictly monotonically increasing function in the interval $0 \leq r \leq L-1$.

3. Histogram Processing

- Histogram Equalization:

- a. $T(r)$ is a monotonically increasing function:

- ✓ Guarantees that output intensity values will never be less than corresponding input values.

- b. $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$:

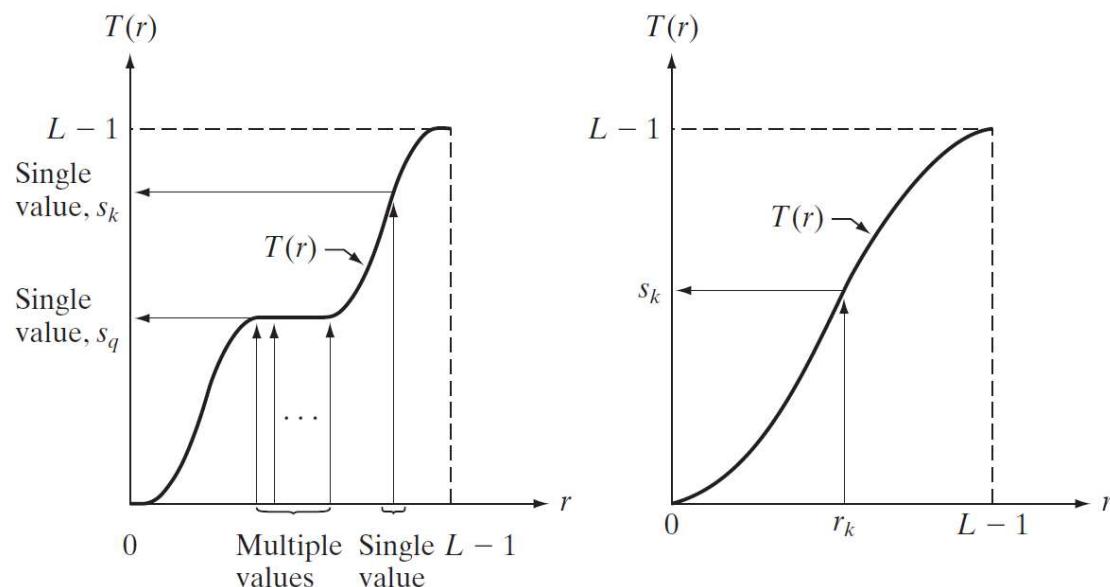
- ✓ Guarantees that the range of output intensities is the same as the input.

- a'. $T(r)$ is a strictly monotonically increasing function:

- ✓ Guarantees that the mappings from s back to r will be one-to-one, thus preventing ambiguities.

3. Histogram Processing

- Histogram Equalization:



a b

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

[†]Recall that a function $T(r)$ is *monotonically increasing* if $T(r_2) \geq T(r_1)$ for $r_2 > r_1$. $T(r)$ is a *strictly monotonically increasing* function if $T(r_2) > T(r_1)$ for $r_2 > r_1$. Similar definitions apply to monotonically decreasing functions.

3. Histogram Processing

- Histogram Equalization:
 - ✓ The intensity levels in an image may be viewed as random variables in the interval [0, L-1].
 - ✓ A fundamental descriptor of a random variable is its probability density function (PDF).
 - ✓ Let $p_r(r)$ and $p_s(s)$ denote the PDFs of r and s , respectively.
 - ✓ A fundamental result from basic probability theory (<https://goo.gl/HChKEI>):

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|$$

3. Histogram Processing

- Histogram Equalization:

- ✓ A transformation function of particular importance in image processing has the form,

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw$$

- ✓ The right side of this equation is recognized as a scaled version of the cumulative distribution function (CDF) of random variable r .
- ✓ To find the corresponding to the transformation, we use,

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|$$

3. Histogram Processing

- Histogram Equalization:

✓ A transformation function of particular importance in image processing has the form,

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw$$

$$\begin{aligned} \frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= (L - 1) \frac{d}{dr} \left[\int_0^r p_r(w) dw \right] \\ &= (L - 1)p_r(r) \end{aligned}$$

3. Histogram Processing

- Histogram Equalization:

✓ A transformation function of particular importance in image processing has the form,

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw$$

$$\begin{aligned} p_s(s) &= p_r(r) \left| \frac{dr}{ds} \right| \\ &= p_r(r) \left| \frac{1}{(L - 1)p_r(r)} \right| \\ &= \frac{1}{L - 1} \quad 0 \leq s \leq L - 1 \end{aligned}$$

3. Histogram Processing

- Histogram Equalization:

- ✓ We recognize the form of $p_s(s)$ as a uniform probability density function.
- ✓ Simply stated, we have demonstrated that performing the suggested intensity transformation yields a random variable, s , characterized by a uniform PDF.

$$p_s(s) = \frac{1}{L - 1} \quad 0 \leq s \leq L - 1$$

- ✓ It is important to note that $p_s(s)$ is always uniform, independently of the form of $p_r(r)$.

3. Histogram Processing

- Histogram Equalization:

✓ We recognize the form of $p_s(s)$ as a uniform probability density function.

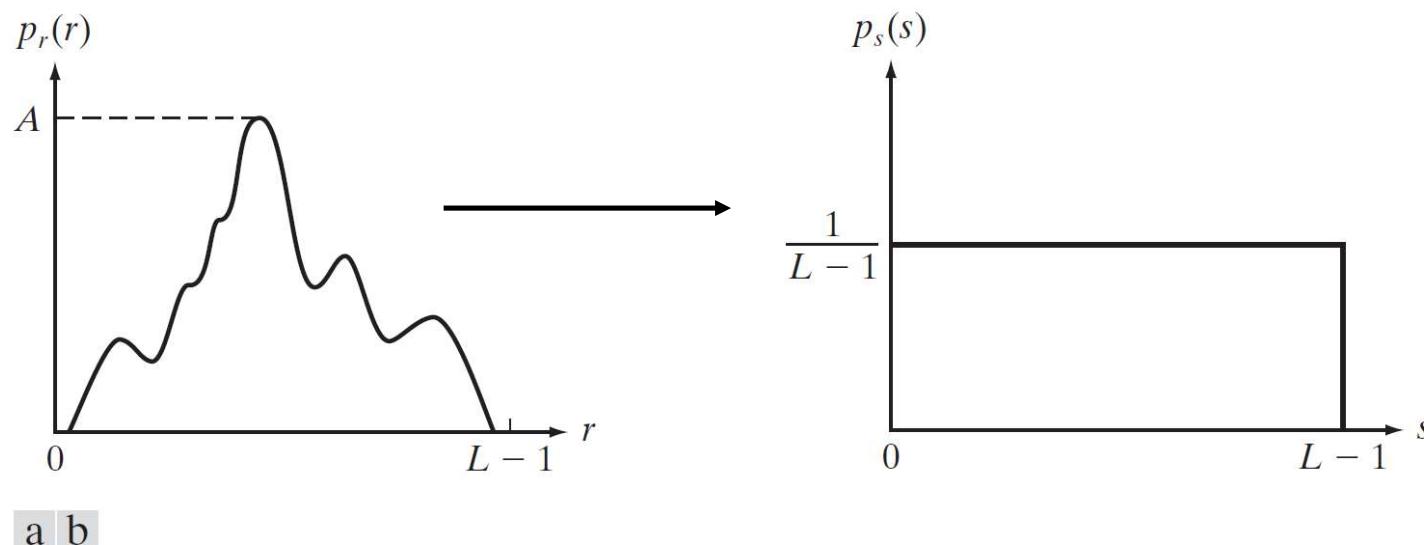


FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r . The resulting intensities, s , have a uniform PDF, independently of the form of the PDF of the r 's.

3. Histogram Processing

- Example: Suppose that the (continuous) intensity values in an image have the PDF,

$$p_r(r) = \begin{cases} \frac{2r}{(L-1)^2} & \text{for } 0 \leq r \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

then,

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

- Suppose next that we form a new image with intensities, s , obtained using this transformation.
- Suppose $L = 10$ and the particular case $r = 3$.

3. Histogram Processing

- Suppose $L = 10$ and the particular case $r = 3$.
- Then, $s = T(r) = r^2/9 = 1$
- We can verify that the PDF of the intensities in the new image is uniform.

$$\begin{aligned} p_r(r) &= \frac{2r}{(L-1)^2} & p_s(s) &= p_r(r) \left| \frac{dr}{ds} \right| = \frac{2r}{(L-1)^2} \left| \left[\frac{ds}{dr} \right]^{-1} \right| \\ && &= \frac{2r}{(L-1)^2} \left| \left[\frac{d}{dr} \frac{r^2}{L-1} \right]^{-1} \right| \\ && &= \frac{2r}{(L-1)^2} \left| \frac{(L-1)}{2r} \right| = \frac{1}{L-1} \end{aligned}$$

3. Histogram Processing

- Histogram Equalization (for discrete values):
 - ✓ For discrete values, we deal with probabilities (histogram values) and summations instead of PDFs and integrals.
 - ✓ As mentioned earlier, the probability of occurrence of intensity level in a digital image is approximated by,

$$p_r(r_k) = \frac{n_k}{MN} \quad k = 0, 1, 2, \dots, L - 1$$

where MN is the total number of pixels in the image, n_k is the number of pixels that have intensity r_k , and L is the number of possible intensity levels in the image.

3. Histogram Processing

- Histogram Equalization (for discrete values):
 - ✓ The discrete form of the previously proposed transformation is,

$$\begin{aligned}s_k &= T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) \\ &= \frac{(L - 1)}{MN} \sum_{j=0}^k n_j \quad k = 0, 1, 2, \dots, L - 1\end{aligned}$$

3. Histogram Processing

- Example:

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

TABLE 3.1
Intensity distribution and histogram values for a 3-bit, 64×64 digital image.

3. Histogram Processing

- Example:

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

$$s_0 = T(r_0) = 7 \sum_{j=0}^0 p_r(r_j) = 7 p_r(r_0) = 1.33$$

3. Histogram Processing

- Example:

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

$$s_1 = T(r_1) = 7 \sum_{j=0}^1 p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08$$

3. Histogram Processing

- Example:

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

$$s_2 = 4.55, s_3 = 5.67, s_4 = 6.23, s_5 = 6.65, s_6 = 6.86, s_7 = 7.00$$

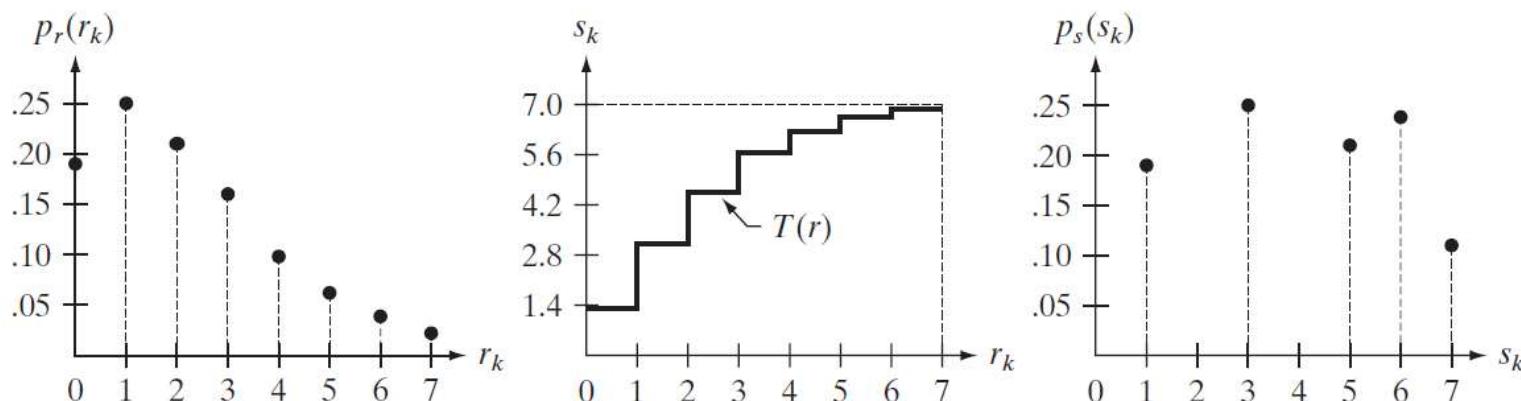
3. Histogram Processing

- Example:

r_k	n_k	$p_r(r_k) = n_k/MN$	
$r_0 = 0$	790	0.19	$s_0 = 1.33 \rightarrow 1$
$r_1 = 1$	1023	0.25	$s_1 = 3.08 \rightarrow 3$
$r_2 = 2$	850	0.21	$s_2 = 4.55 \rightarrow 5$
$r_3 = 3$	656	0.16	$s_3 = 5.67 \rightarrow 6$
$r_4 = 4$	329	0.08	$s_4 = 6.23 \rightarrow 6$
$r_5 = 5$	245	0.06	$s_5 = 6.65 \rightarrow 7$
$r_6 = 6$	122	0.03	$s_6 = 6.86 \rightarrow 7$
$r_7 = 7$	81	0.02	$s_7 = 7.00 \rightarrow 7$

3. Histogram Processing

- Example:



a b c

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

3. Histogram Processing

- MATLAB

```
% Histogram equalization
clc
clear all
close all

I = imread('tire.tif');
[h, w] = size(I);
n = length(I(:));

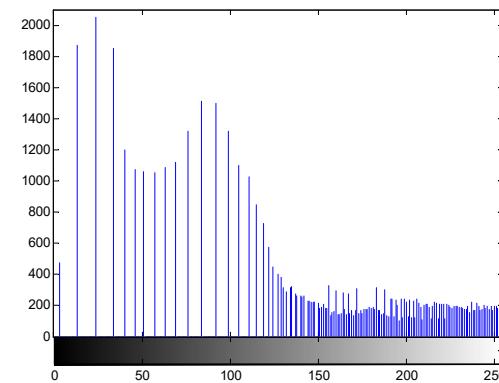
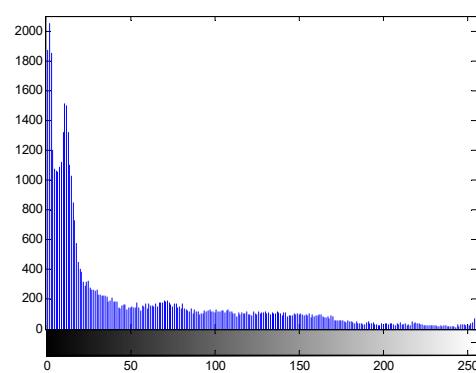
% Probability Mass Function (PMF)
[nk rk] = hist(I(:), [0:255]);
stem(rk, nk,'.'); grid on; title('Original Image Histogram (PMF)');

% Building the transformation function (CDF)
ps = nk/n;
sk = cumsum(ps);
figure; stem(rk,255*sk,'.'); grid on; title('Transformation Function (CDF)');
xlabel('r_k'); ylabel('s_k = T(r_k)')
figure; imshow(I)

% Equalization
for i = 1:h
    for j = 1:w
        Ih(i,j) = round(255*sk(I(i,j)+1));
    end
end
figure; imshow(uint8(Ih))
```

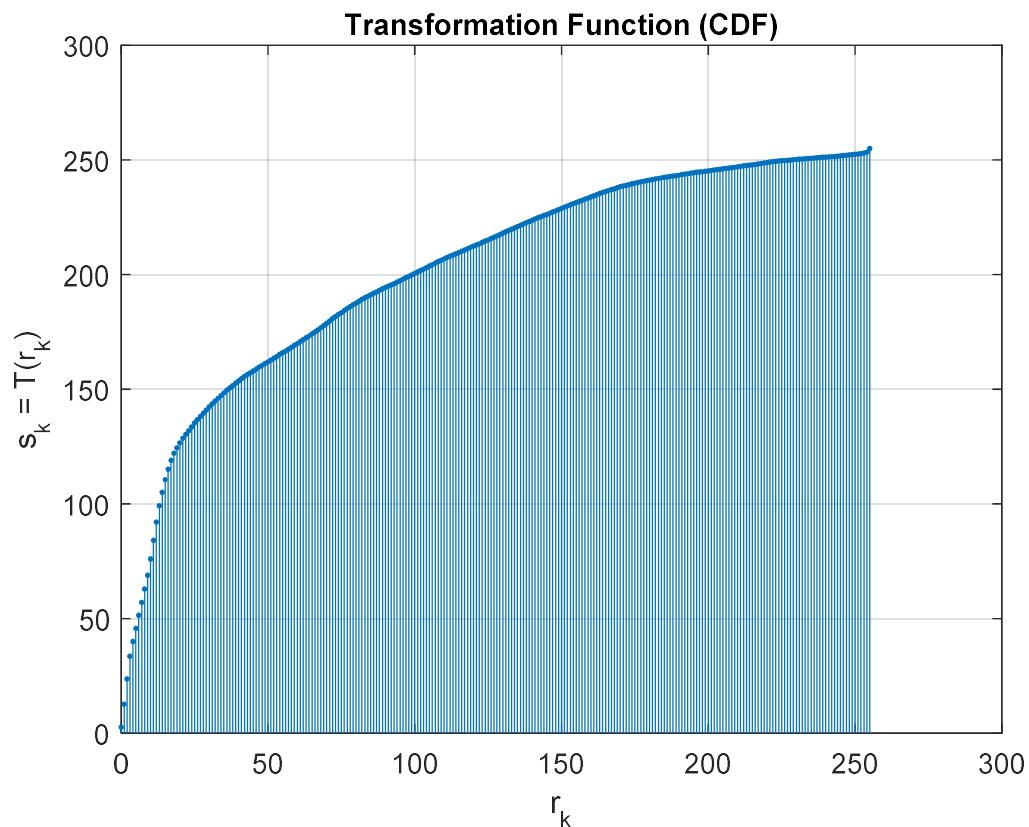
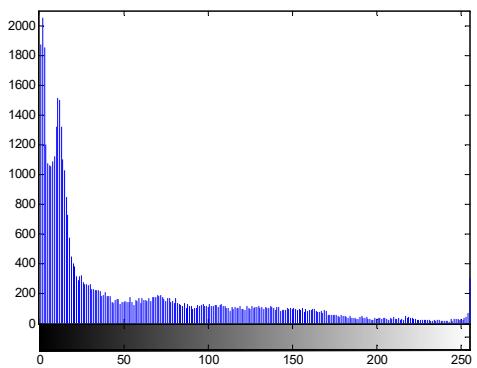
3. Histogram Processing

- Example:



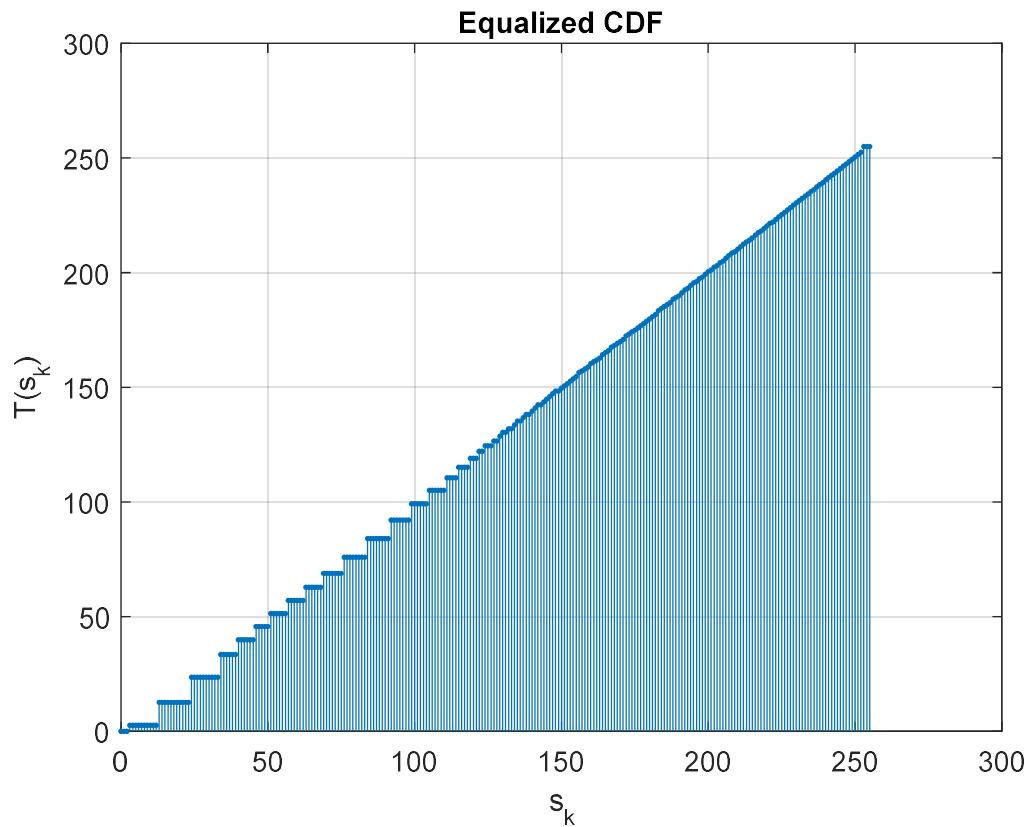
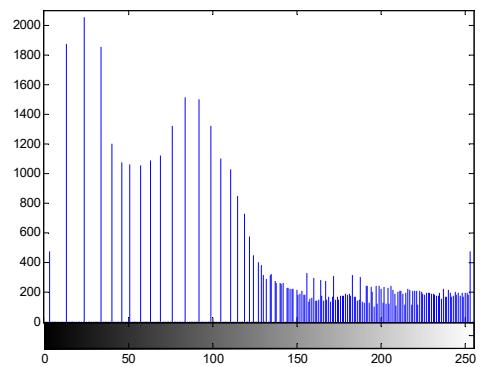
3. Histogram Processing

- Example:



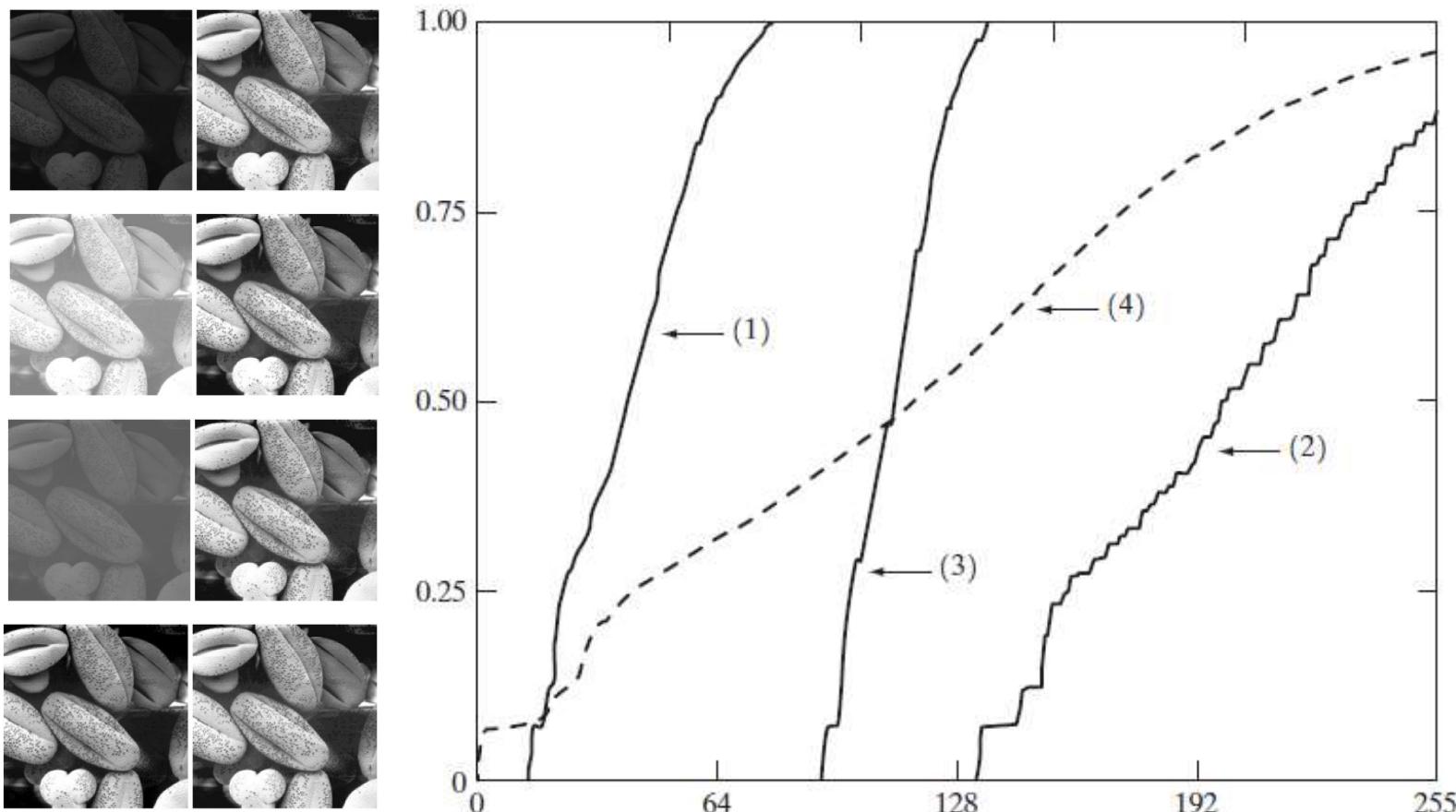
3. Histogram Processing

- Example:



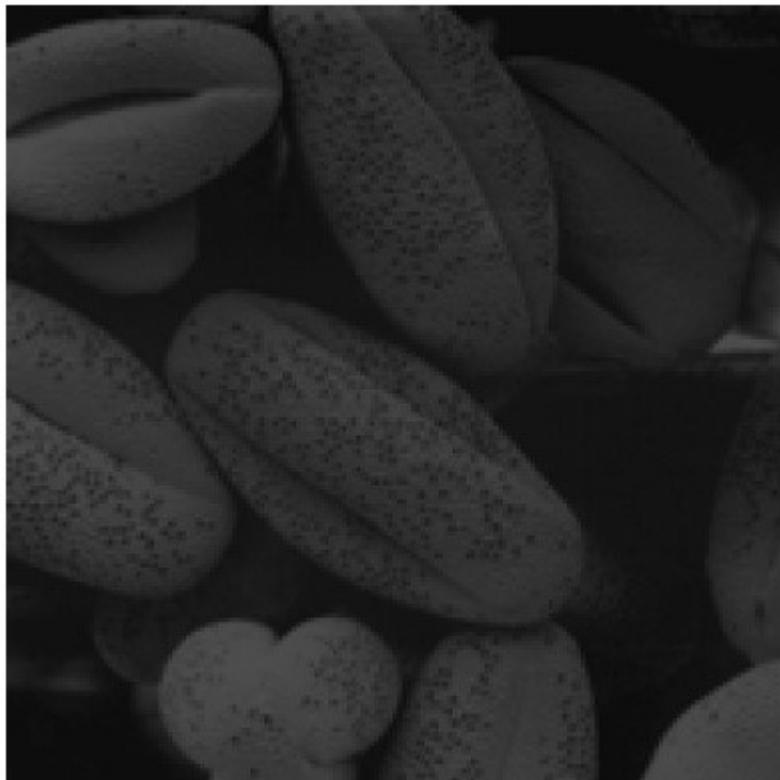
3. Processamento de Histograma

- Other examples (1-4):



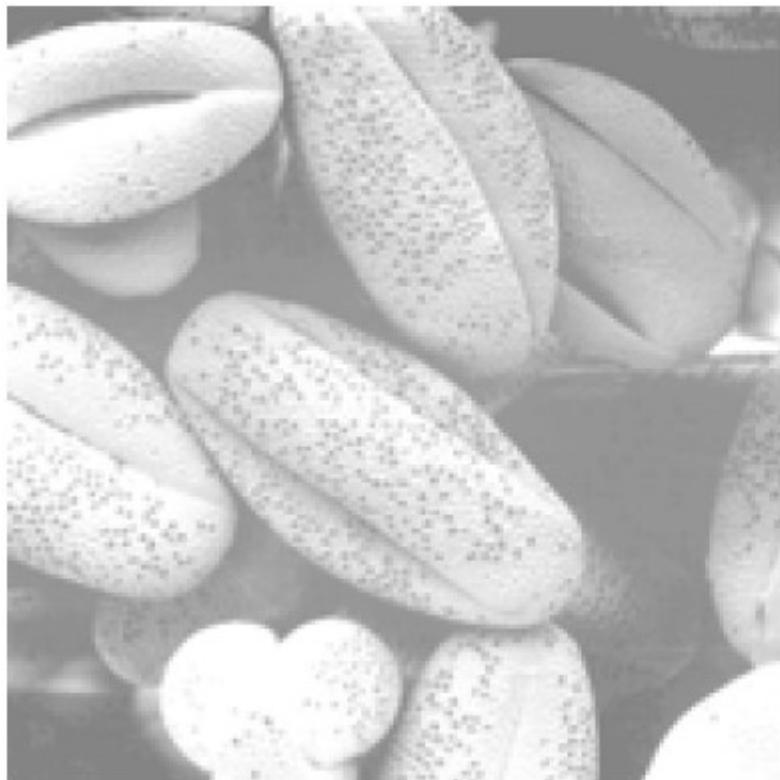
3. Histogram Processing

- Other examples (1):



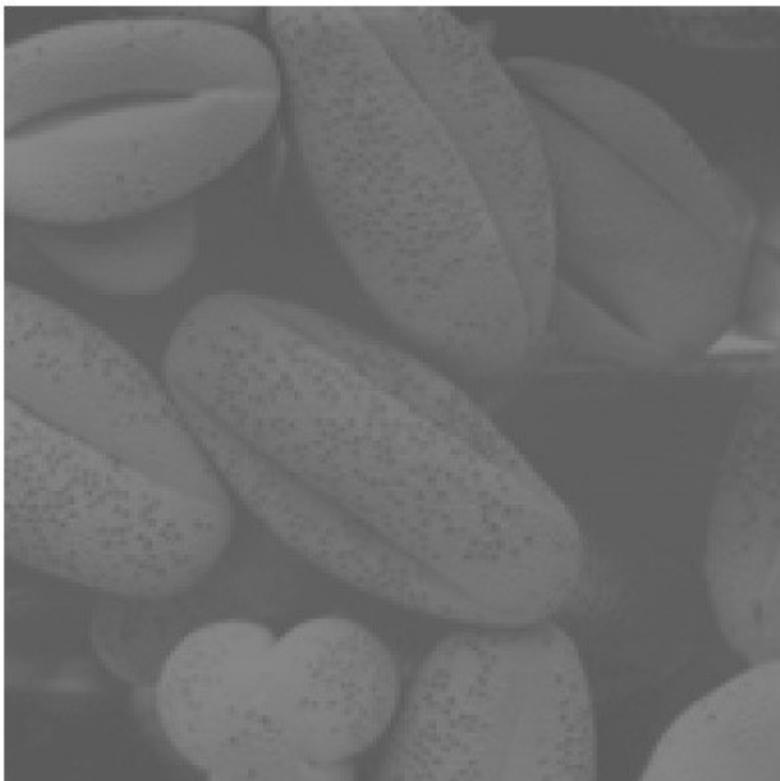
3. Histogram Processing

- Other examples (2):



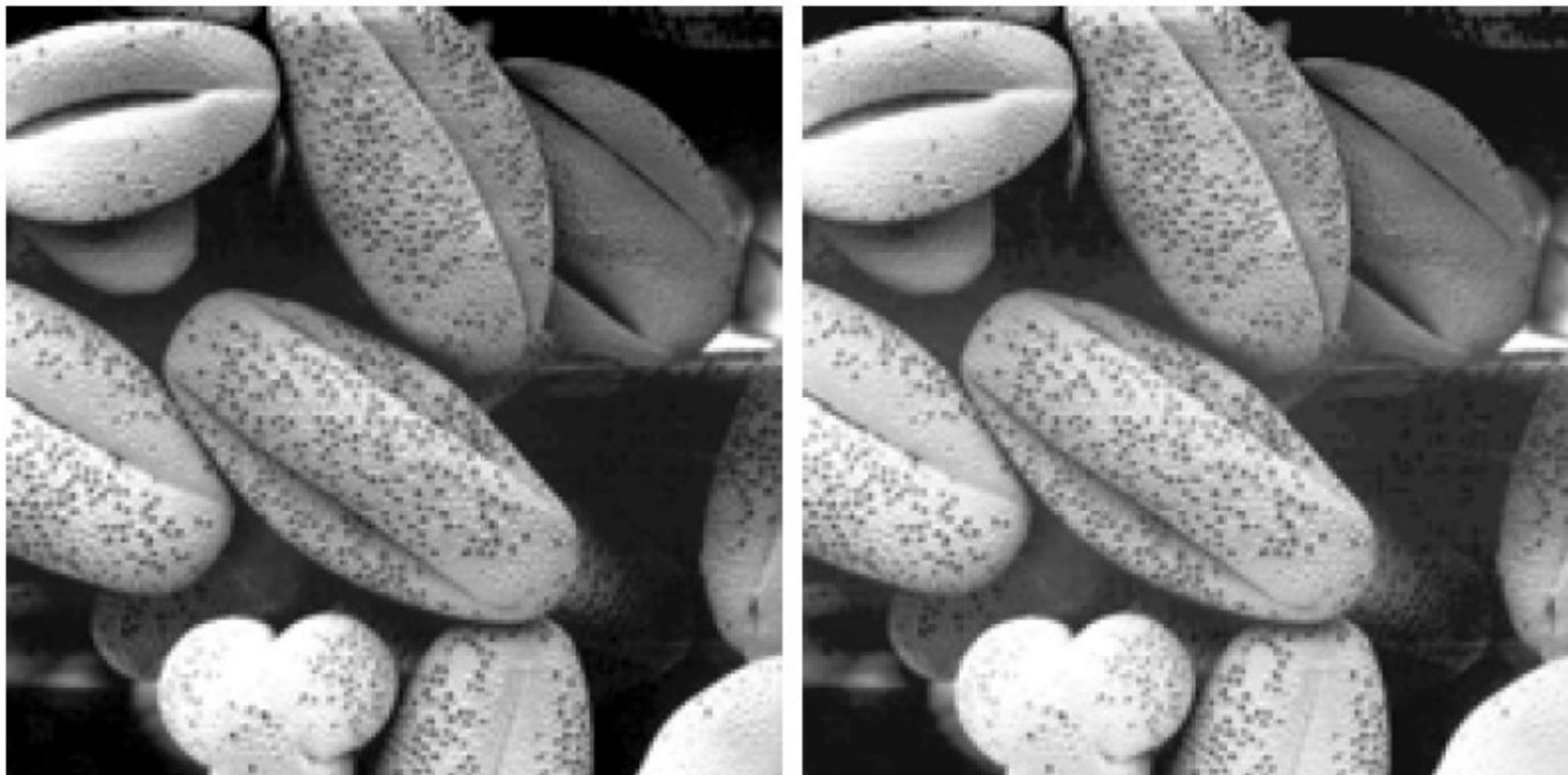
3. Histogram Processing

- Other examples (3):



3. Histogram Processing

- Other examples (4):



3. Histogram Processing

- Histogram Matching (Specification):
 - ✓ It is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have.
 - ✓ The method used to generate a processed image that has a specified histogram is called **histogram matching** or **histogram specification**.

- ✓ Let s be a random variable s with the property,

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw$$

- ✓ Suppose next that we define a random variable z with the property,

$$G(z) = (L - 1) \int_0^z p_z(t) dt = s$$

3. Histogram Processing

- Histogram Matching (Specification):

- ✓ It then follows from these two equations that,

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

and, therefore, that z must satisfy the condition,

$$z = G^{-1}[T(r)] = G^{-1}(s)$$

- ✓ $T(r)$ can be obtained, since $p_r(r)$ has been estimated from the input image.
 - ✓ Similarly, the transformation function $G(z)$ can be obtained because $p_z(z)$ is given.

3. Histogram Processing

- Histogram Matching (Specification):

- ✓ Discrete formulation

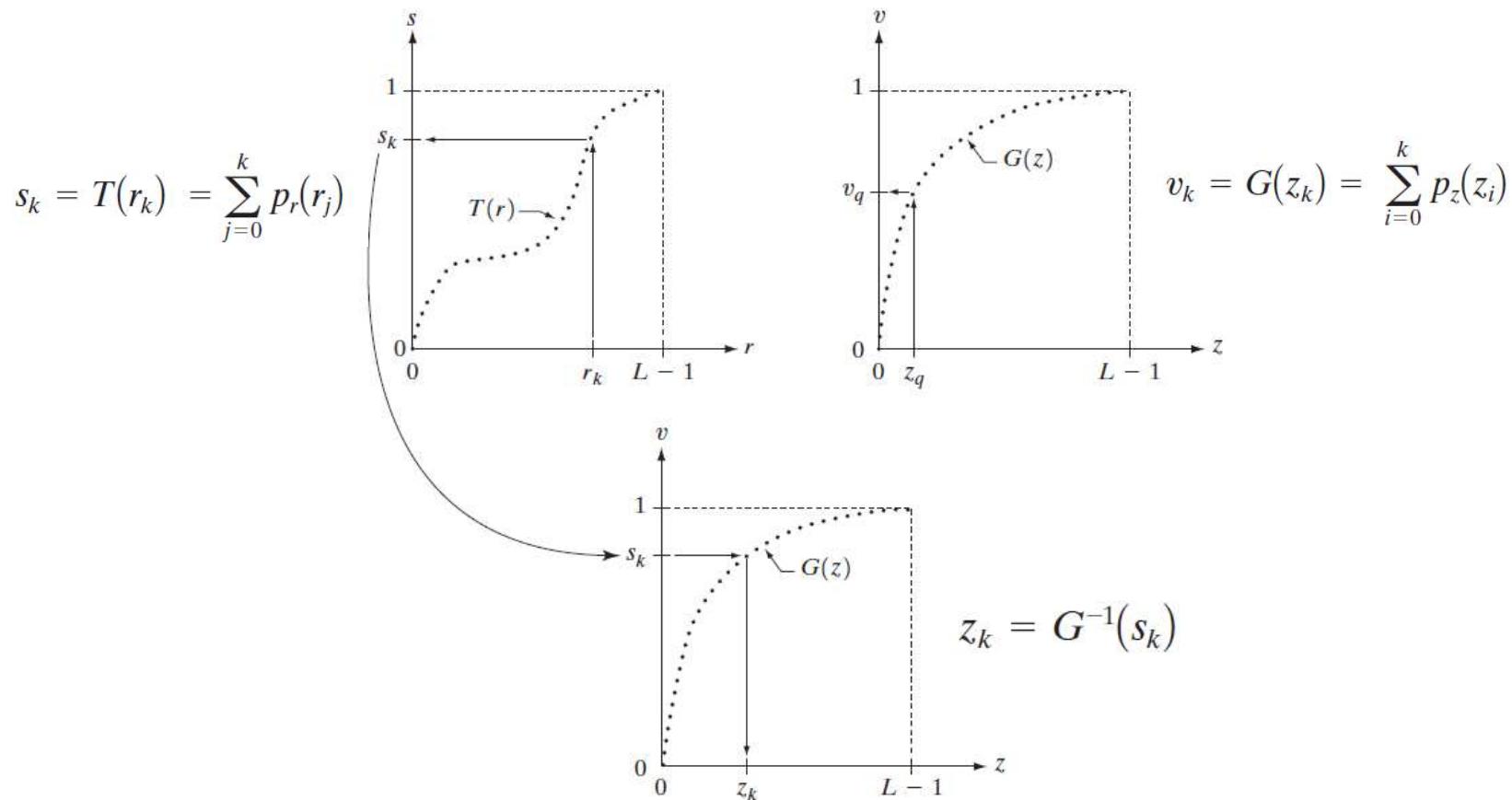
$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j)$$
$$k = 0, 1, 2, \dots, L - 1$$

$$G(z_q) = (L - 1) \sum_{i=0}^q p_z(z_i)$$

$$G(z_q) = s_k \longrightarrow z_q = G^{-1}(s_k)$$

3. Histogram Processing

- Histogram Matching (Specification):



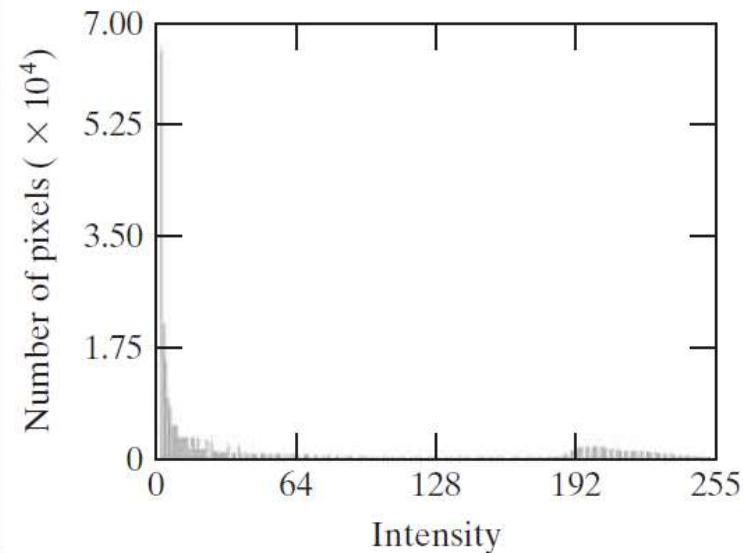
3. Histogram Processing

- Histogram Matching (Specification):

a b

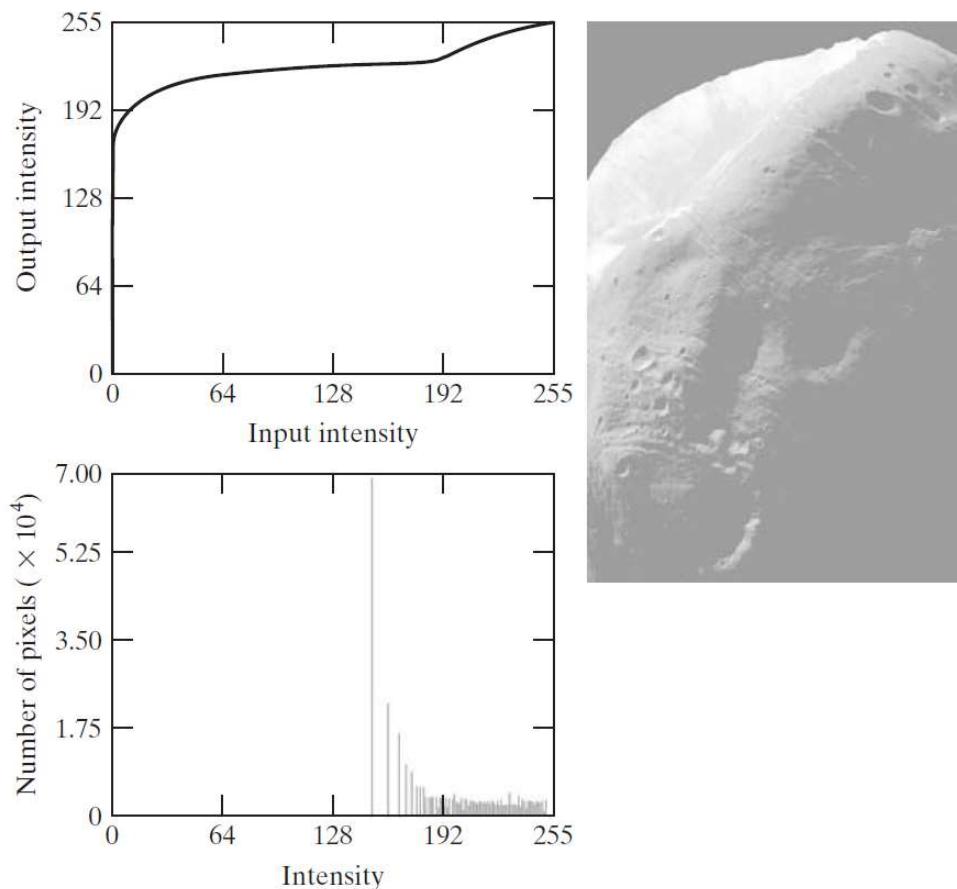
FIGURE 3.23

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



3. Histogram Processing

- Histogram Matching (Specification):



a
b
c

FIGURE 3.24

- (a) Transformation function for histogram equalization.
(b) Histogram-equalized image (note the washed-out appearance).
(c) Histogram of (b).

3. Histogram Processing

- Histogram Matching (Specification):

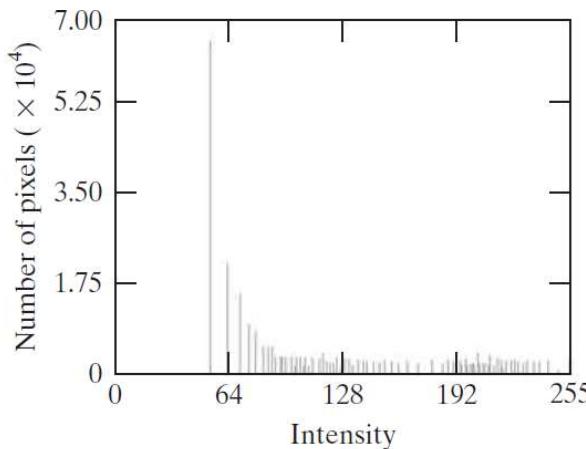
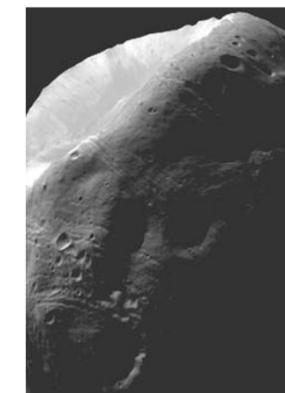
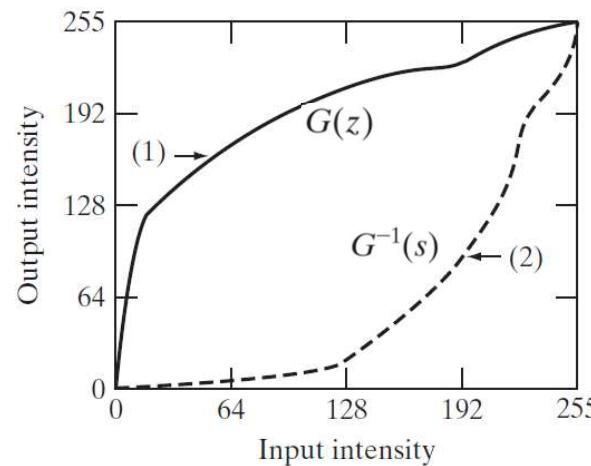
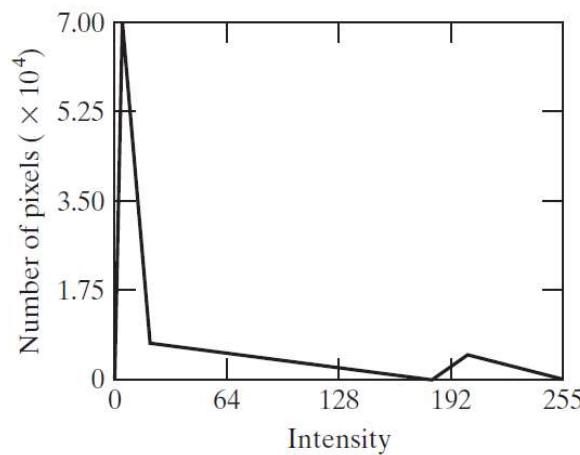
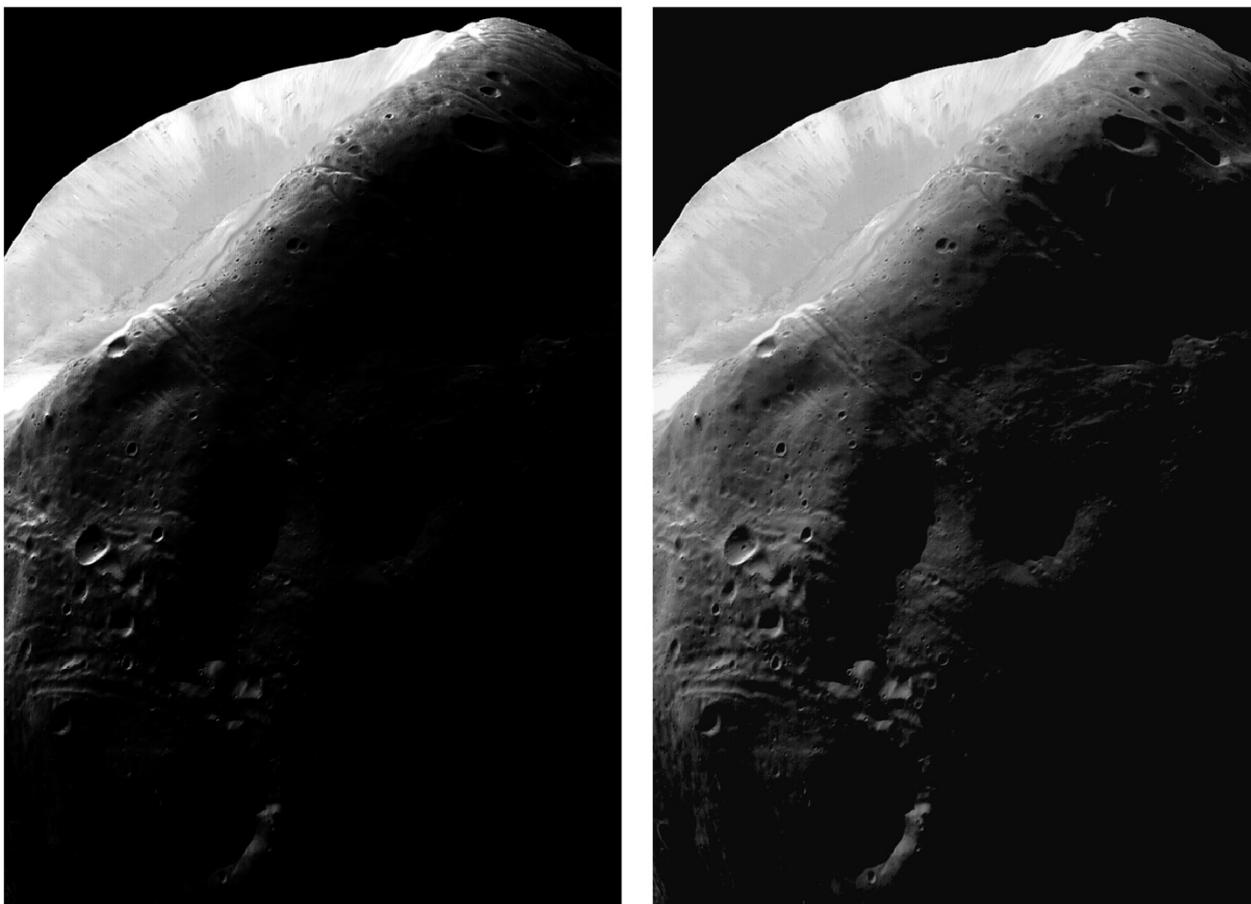


FIGURE 3.25
(a) Specified histogram.
(b) Transformations.
(c) Enhanced image using mappings from curve (2).
(d) Histogram of (c).

3. Histogram Processing

- MATLAB: s076histspec.m



3. Histogram Processing

- Local Histogram Processing:

- ✓ The histogram processing techniques previously described are easily adapted to local enhancement.
- ✓ The procedure is to define a neighborhood and move its center from pixel to pixel. At each location, the histogram of the points in the neighborhood is computed.

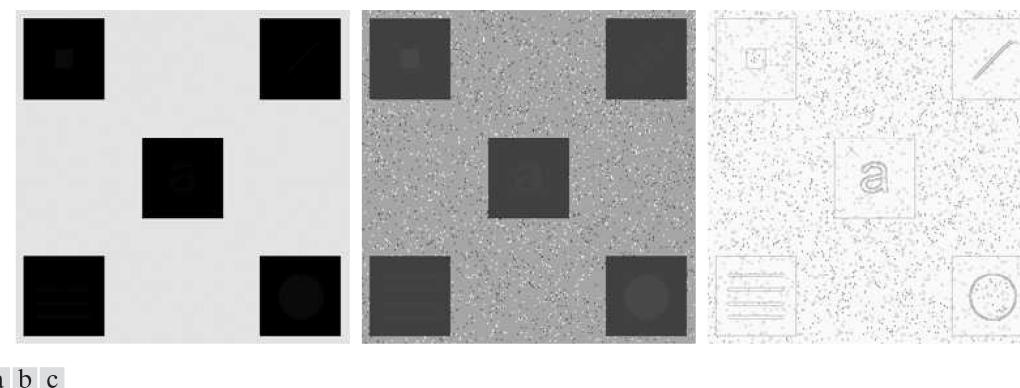


FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3×3 .

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

- ✓ The n th moment of r about its mean m is defined as

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i)$$

where

$$m = \sum_{i=0}^{L-1} r_i p(r_i)$$

- ✓ The second moment (variance) is particularly important

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i)$$

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

- ✓ Example:

$$I = \begin{bmatrix} 0 & 0 & 1 & 1 & 2 \\ 1 & 2 & 3 & 0 & 1 \\ 3 & 3 & 2 & 2 & 0 \\ 2 & 3 & 1 & 0 & 0 \\ 1 & 1 & 3 & 2 & 2 \end{bmatrix}$$

$$L = 4 \rightarrow r_k = [0 \ 1 \ 2 \ 3]$$

$$p(r_k) = [0.24 \ 0.28 \ 0.28 \ 0.20]$$

$$m = 0 \cdot 0.24 + 1 \cdot 0.28 + 2 \cdot 0.28 + 3 \cdot 0.20 = 1.44$$

$$\begin{aligned} \mu_2 &= (0 - 1.44)^2 \cdot 0.24 + (1 - 1.44)^2 \cdot 0.28 + \\ &\quad (2 - 1.44)^2 \cdot 0.28 + (3 - 1.44)^2 \cdot 0.20 = 1.1264 \end{aligned}$$

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

- ✓ It is common practice to estimate the mean and the variance sample values, without computing the histogram:

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

$$\sigma^2 = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x, y) - m]^2$$

- ✓ Global mean and variance may be computed over an entire image
- ✓ However, a more powerful use of these parameters is in local enhancement.

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

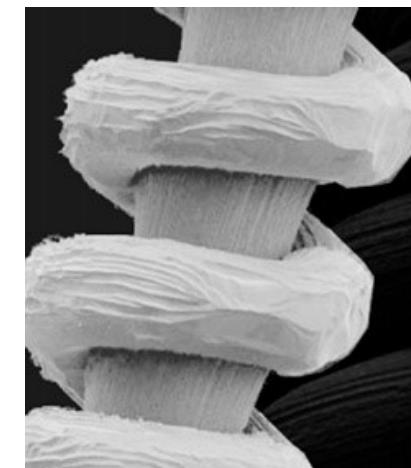
➤ In this particular case, the problem is to enhance dark areas while leaving the light area as unchanged as possible because it does not require enhancement.

➤ A measure of whether an area is relatively light or dark at a point (x, y) is to compare the average local intensity $m_{S_{xy}}$ to the average image intensity M_G .

➤ Candidate for processing:

$$m_{S_{xy}} \leq k_0 M_G$$

k_0 is positive, less than 1.0.



3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

➤ Enhancing areas that have low contrast.

Candidates:

$$\sigma_{S_{xy}} \leq k_2 D_G$$

where $\sigma_{S_{xy}}$ is the local standard deviation, D_G is the global standard deviation and k_2 is a positive constant, less than 0.1.

➤ Finally, we need to restrict the lowest values of contrast we are willing to accept,

$$k_1 D_G \leq \sigma_{S_{xy}}$$

with $k_1 < k_2$.

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

➤ A pixel that meets all the conditions for local enhancement is processed simply by multiplying it by a specified constant, E , to increase the value of its intensity level relative to the rest of the image:

$$g(x, y) = \begin{cases} E \cdot f(x, y) & \text{if } m_{S_{xy}} \leq k_0 m_G \text{ AND } k_1 \sigma_G \leq \sigma_{S_{xy}} \leq k_2 \sigma_G \\ f(x, y) & \text{otherwise} \end{cases}$$

- Choosing these parameters generally requires a bit of experimentation.
- In this example, $E = 4.0$, $k_0 = 0.4$, $k_1 = 0.02$ and $k_2 = 0.4$. A 3x3 window was used.

3. Histogram Processing

- Using Histogram Statistics for Image Enhancement:

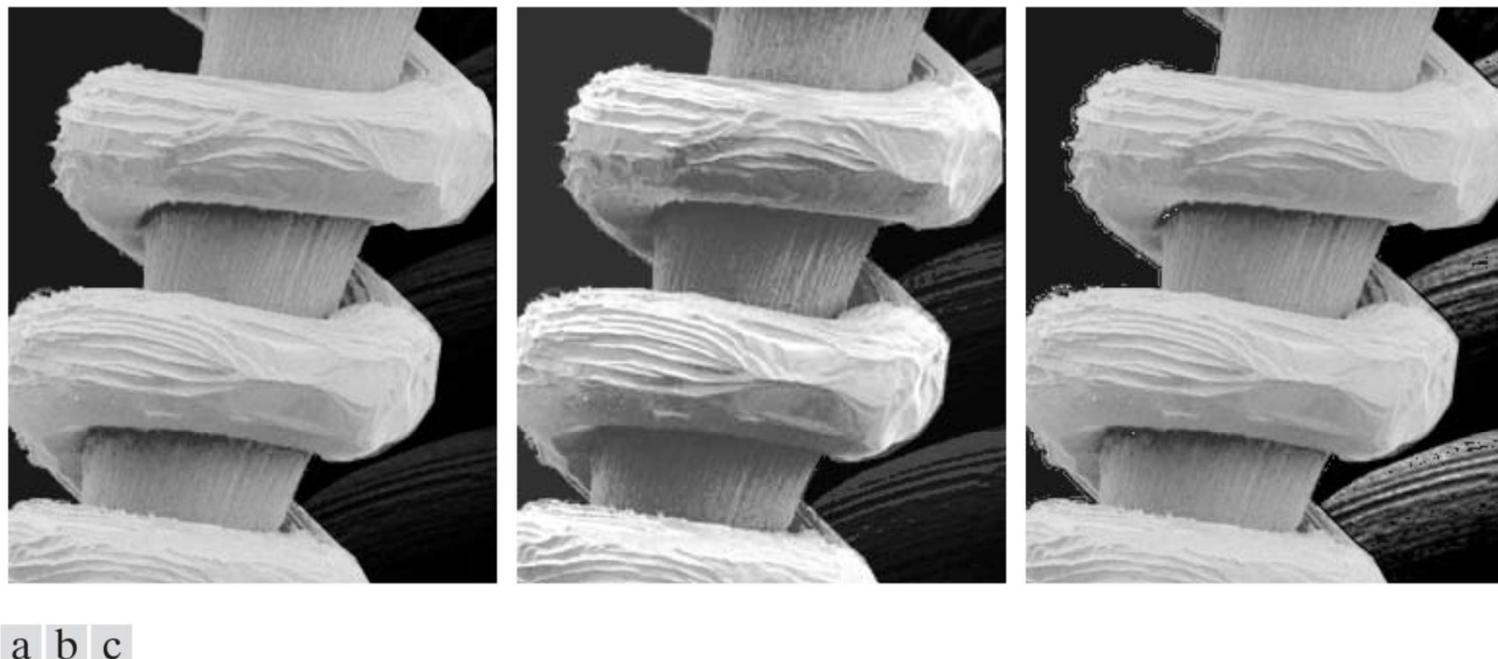


FIGURE 3.27 (a) SEM image of a tungsten filament magnified approximately 130×. (b) Result of global histogram equalization. (c) Image enhanced using local histogram statistics. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

4. Fundamentals of Spatial Filtering

- The Mechanics of Spatial Filtering:
 - The name *filter* is borrowed from frequency domain processing.
 - “Filtering” refers to accepting (passing) or rejecting certain frequency components.
 - For example, a filter that passes low frequencies is called a **lowpass filter**.
 - If the operation performed on the image pixels is **linear**, then the filter is called a **linear spatial filter**. Otherwise, the filter is **nonlinear**.

4. Fundamentals of Spatial Filtering

- The Mechanics of Spatial Filtering:

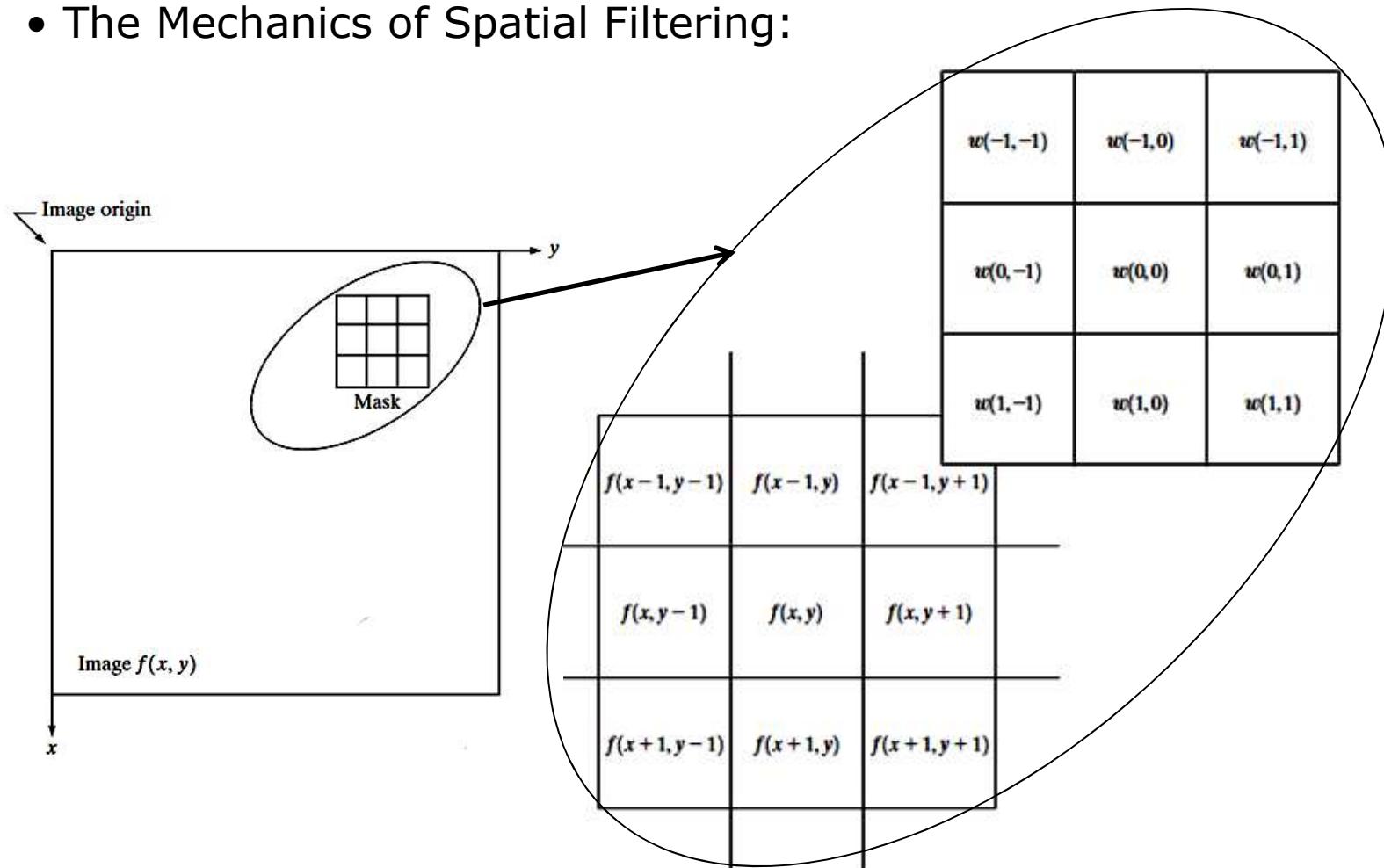
- For a mask (filter) of size $m \times n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a e b are positive integers.
- At any point (x, y) in the image, the response, $g(x,y)$, of the filter is the sum of products of the filter coefficients and the image pixels encompassed by the filter:

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

where x and y are varied so that each pixel in w visits every pixel in f .

4. Fundamentals of Spatial Filtering

- The Mechanics of Spatial Filtering:

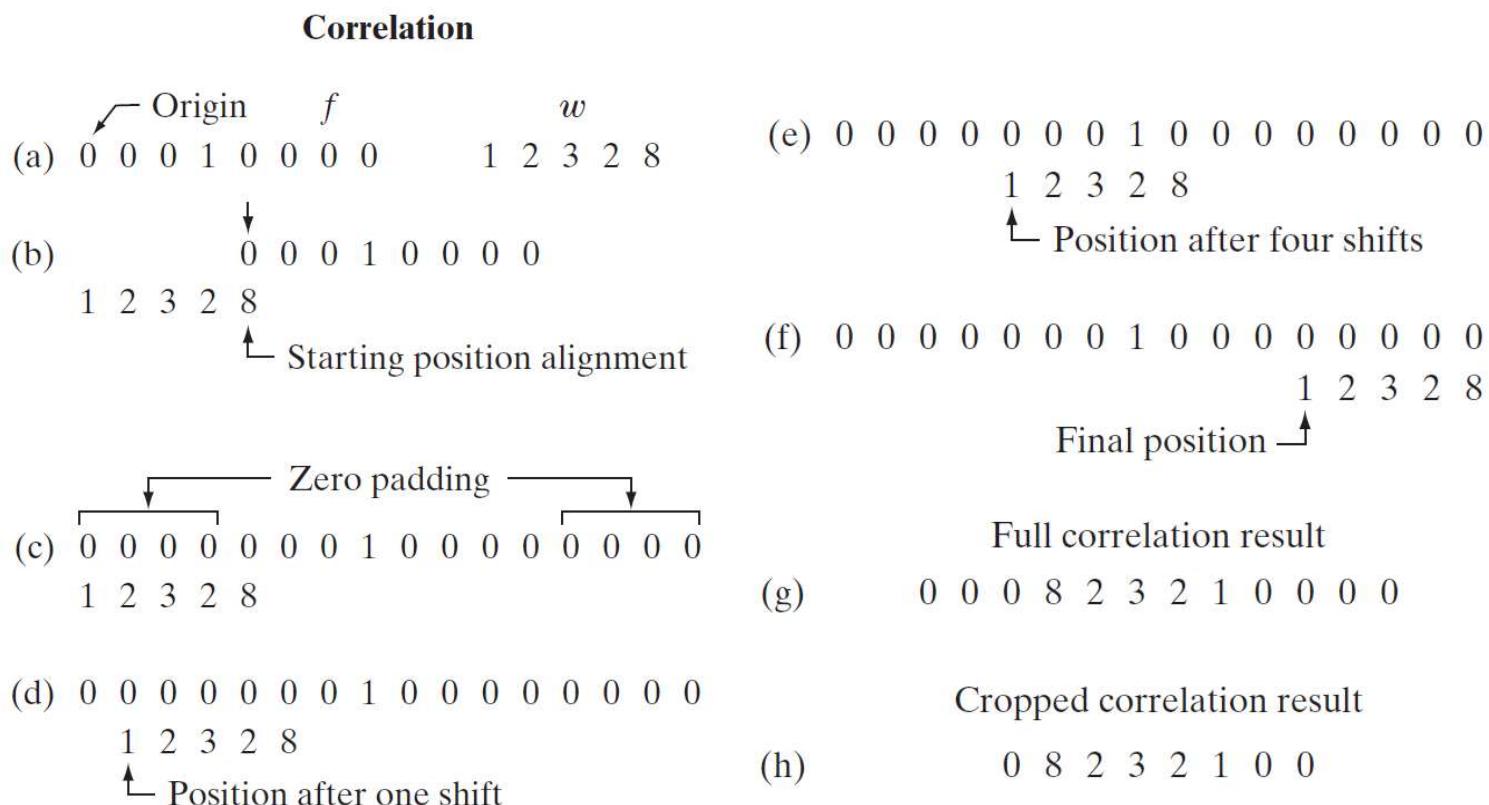


4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:
 - **Correlation** is the process of moving a filter mask over the image and computing the sum of products at each location, exactly as explained in the previous slides.
 - The mechanics of **convolution** are the same, except that the filter is first rotated by 180°.

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:



4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:

Convolution

↙ Origin f w rotated 180°

$\begin{matrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{matrix}$	$\begin{matrix} 8 & 2 & 3 & 2 & 1 \end{matrix}$	(i) $\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$ (m) $\begin{matrix} 8 & 2 & 3 & 2 & 1 \end{matrix}$
---	---	--

$\begin{matrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 8 & 2 & 3 & 2 & 1 \end{matrix}$	(j)	(n) $\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$ $\begin{matrix} 8 & 2 & 3 & 2 & 1 \end{matrix}$
--	-----	--

$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$ (k) $\begin{matrix} 8 & 2 & 3 & 2 & 1 \end{matrix}$	Full convolution result $\begin{matrix} 0 & 0 & 0 & 1 & 2 & 3 & 2 & 8 & 0 & 0 & 0 & 0 \end{matrix}$ (o)
--	--

$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$ (l) $\begin{matrix} 8 & 2 & 3 & 2 & 1 \end{matrix}$	Cropped convolution result $\begin{matrix} 0 & 1 & 2 & 3 & 2 & 8 & 0 & 0 \end{matrix}$ (p)
--	---

4. Fundamentals of Spatial Filtering

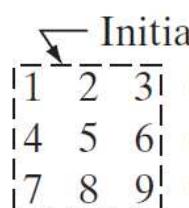
- Spatial Correlation and Convolution:
 - We call a function that contains a single 1 with the rest being 0s a **discrete unit impulse**.
 - So we conclude that **correlation** of a function with a discrete unit impulse yields a 180° **rotated** version of the function at the location of the impulse.
 - **Convolution** is a cornerstone of **linear system theory** and will be addressed in more detail later.
 - Convolving a function with a unit impulse yields a **copy** of the function at the location of the impulse.

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:

4. Fundamentals of Spatial Filtering

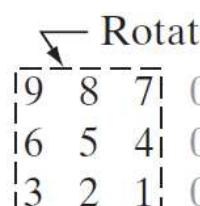
- Spatial Correlation and Convolution:

Initial position for w	Full correlation result	Cropped correlation result
	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)	(e)

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

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:

Rotated w	Full convolution result	Cropped convolution result
	0 0	0 0
(f)	(g)	(h)
0 0	0 0	0 1 2 3 0 0 4 5 6 0 0 7 8 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0
0 0	0 0	0 0

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

4. Fundamentals of Spatial Filtering

- Spatial Correlation and Convolution:
 - If, instead of containing a single 1, image f had contained a **region equal to w** , the value of the correlation function (after normalization) would have been **maximum** when was **centered on that region**.
 - Thus, correlation can be used also to find **matches** between images (MATLAB: `s95corrMatching`).
 - Using correlation or convolution to perform spatial filtering is a matter of preference.
 - What is important is that the filter mask used in a given filtering task corresponds to the intended operation.

4. Fundamentals of Spatial Filtering

- Vector Representation of Linear Filtering
 - Response, R, of a mask either for correlation or convolution,

$$R = w_1 z_1 + w_2 z_2 + \dots + w_{mn} z_{mn}$$

$$\begin{aligned} &= \sum_{k=1}^{mn} w_k z_k \\ &= \mathbf{w}^T \mathbf{z} \end{aligned}$$

where the ws are the coefficients of an m x n filter and the zs are the corresponding image intensities encompassed by the filter.

4. Fundamentals of Spatial Filtering

- Vector Representation of Linear Filtering

➤ Example:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 3.31

Another representation of a general 3×3 filter mask.

$$R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9$$

$$\begin{aligned} &= \sum_{k=1}^9 w_k z_k \\ &= \mathbf{w}^T \mathbf{z} \end{aligned}$$

4. Fundamentals of Spatial Filtering

- Generating Spatial Filter Masks
 - The filter coefficients are selected based on what the filter is supposed to do.
 - Keep in mind that all we can do with linear filtering is to implement a sum of products.
 - For example, suppose that we want to replace the pixels in an image by the average intensity of a neighborhood centered on those pixels.

$$w_i = 1/9 \quad R = \frac{1}{9} \sum_{i=1}^9 z_i$$

4. Fundamentals of Spatial Filtering

- Generating Spatial Filter Masks
 - In some applications, we have a continuous function of two variables, and the objective is to obtain a spatial filter mask based on that function.
 - For example, a Gaussian function of two variables

$$h(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is the standard deviation and, as usual, we assume that coordinates x and y are integers.

- Then, to generate the filter mask from this function, we sample it about its center.

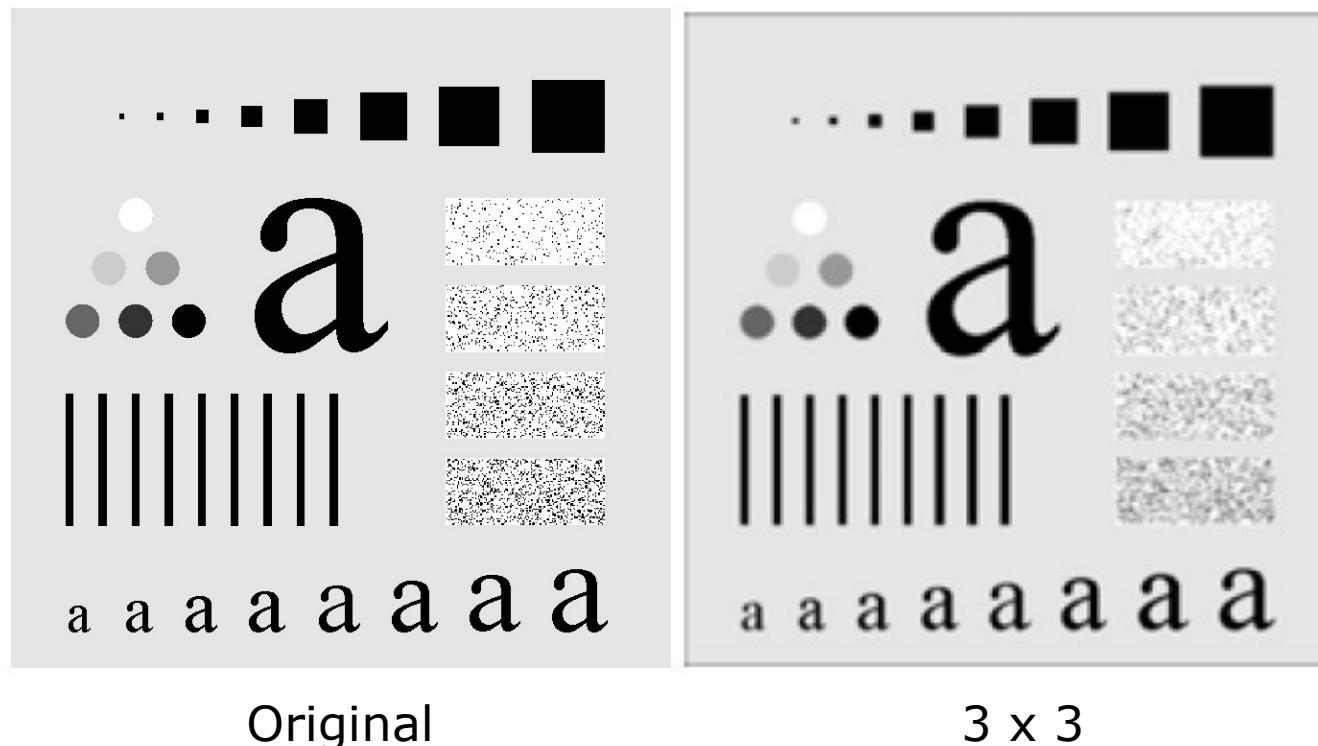
5. Smoothing Spatial Filters

- Smoothing spatial filters are used for blurring and for noise reduction.
- The output of a **smoothing, linear spatial filter** is simply the **average** of the pixels contained in the neighborhood of the filter mask.
- They also are referred to **lowpass filters**.

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

5. Smoothing Spatial Filters

- Example:



5. Smoothing Spatial Filters

- Example:



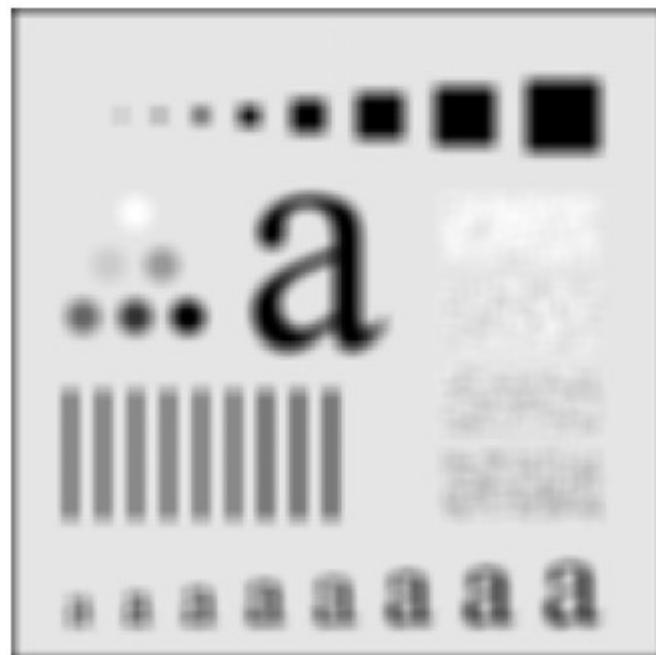
5 x 5



9 x 9

5. Smoothing Spatial Filters

- Example:



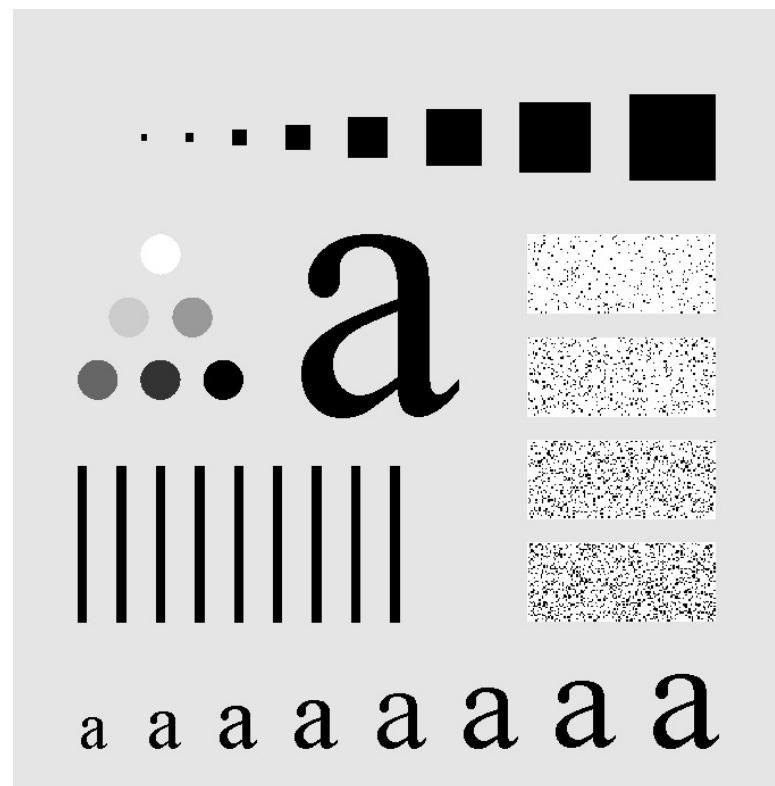
15 x 15



35 x 35

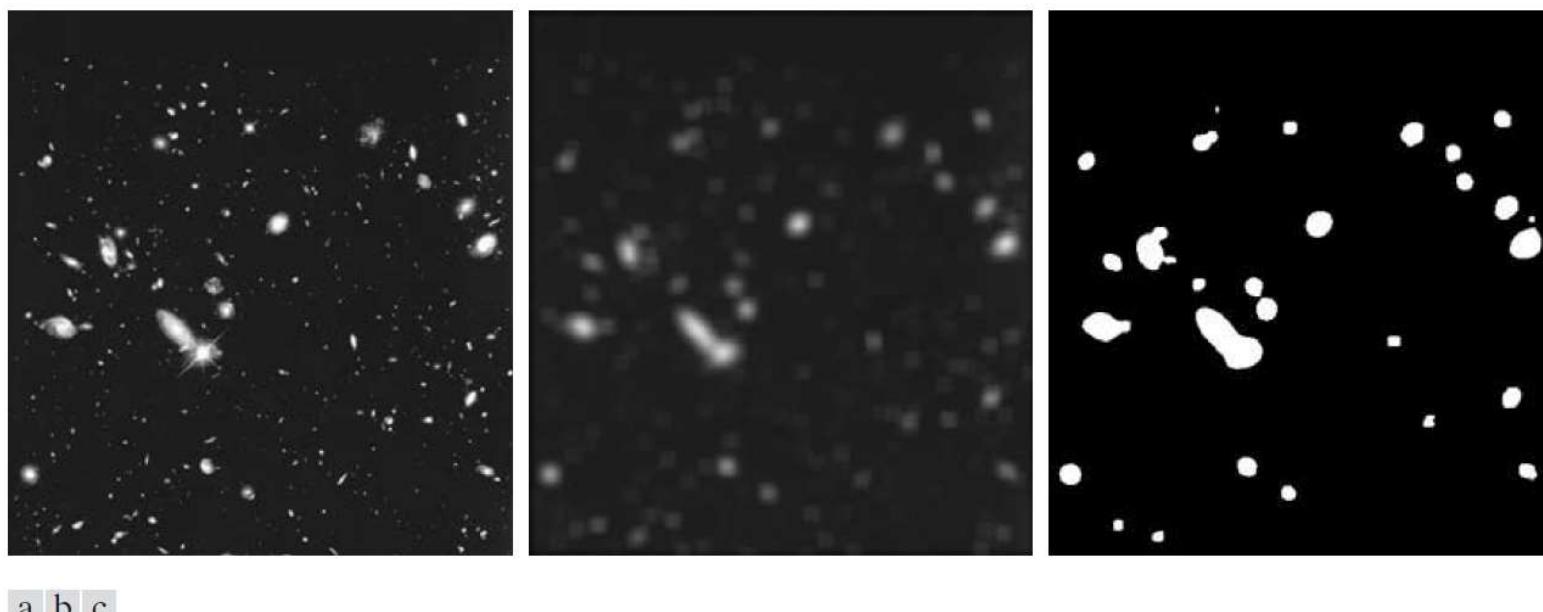
5. Smoothing Spatial Filters

- MATLAB: s104characters.m



5. Smoothing Spatial Filters

- An important application of spatial averaging is to blur an image for the purpose of getting a gross representation of objects of interest.



a b c

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

5. Smoothing Spatial Filters

- MATLAB: s106hubble.m



5. Smoothing Spatial Filters

- Order-Statistic (Nonlinear) Filters
 - **Median filter:** 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.
 - Causes less blurring than linear smoothing filters of similar size.

✓ Example:

10	20	20
20	15	20
20	25	100

✓ Sorted pixels: 10, 15, 20, 20, 20, 20, 20, 25, 100

5. Smoothing Spatial Filters

- Example:

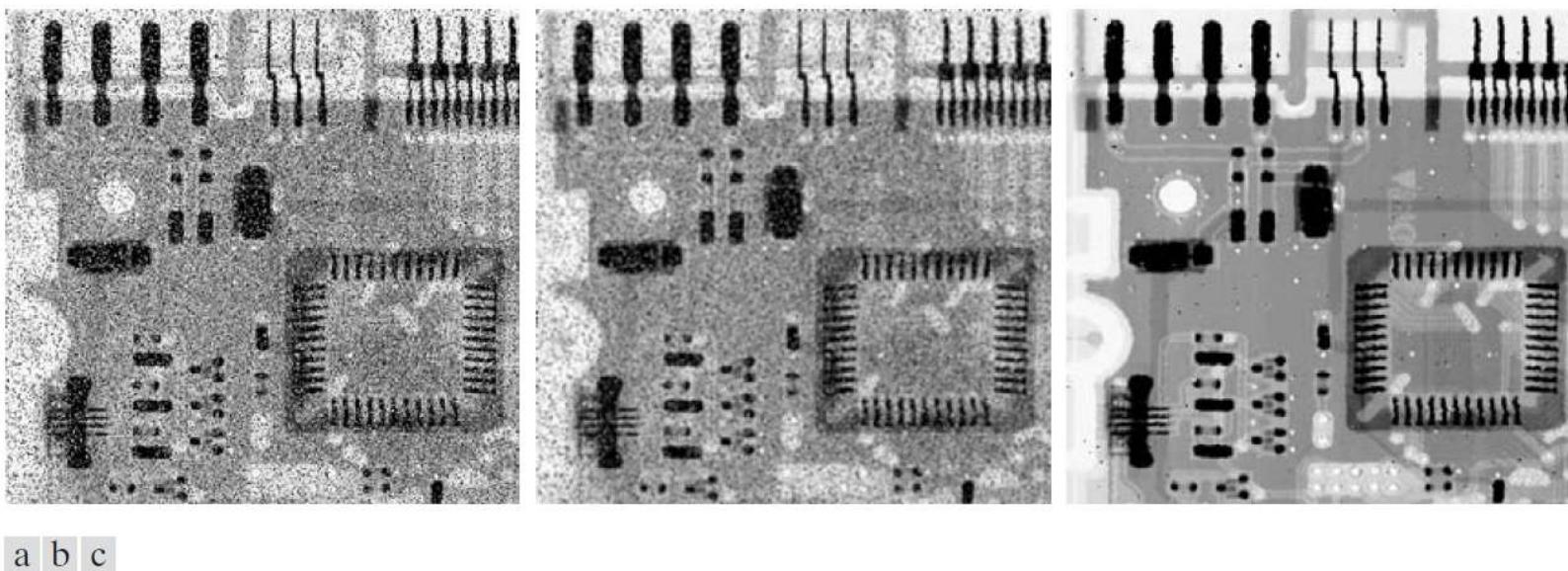
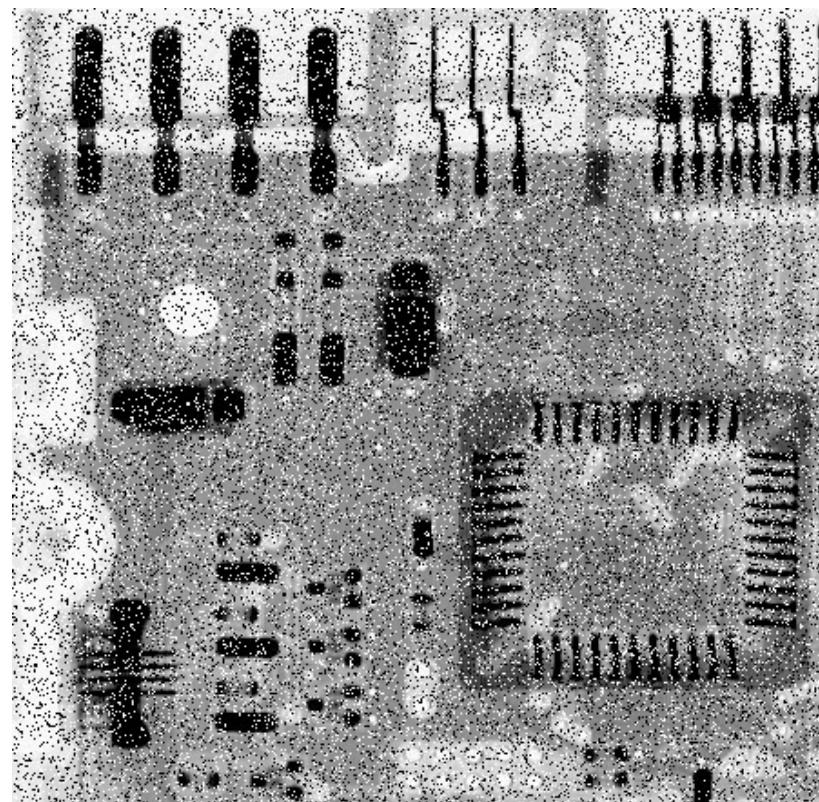


FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

5. Smoothing Spatial Filters

- MATLAB: s109circuit.m



5. Smoothing Spatial Filters

- The median represents the 50th percentile of a ranked set of numbers.
- Using the 100th percentile results in the so-called **max filter**, useful for finding the brightest points in an image.

$$R = \max\{z_k | k = 1, 2, \dots, 9\}$$

- The 0th percentile filter is the **min filter**, used for the opposite purpose.

$$R = \min\{z_k | k = 1, 2, \dots, 9\}$$

6. Sharpening Spatial Filters

- Because **smoothing** is analogous to integration, it is logical to conclude that **sharpening** can be accomplished by spatial differentiation.
- Image differentiation enhances edges and other discontinuities and deemphasizes areas with slowly varying intensities.
- Thus, the principal objective of sharpening is to highlight transitions in intensity.

6. Sharpening Spatial Filters

- The derivatives of a digital function are defined in terms of differences.
- However, we require that any definition we use for a **first derivative**:
 1. must be zero in areas of constant intensity;
 2. must be nonzero at the onset of an intensity step or ramp; and
 3. must be nonzero along ramps.
- A basic definition of the first-order derivative of a one-dimensional function f is the difference

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

6. Sharpening Spatial Filters

- Similarly, any definition of a **second derivative** must:
 1. must be zero in constant areas;
 2. must be nonzero at the onset and end of an intensity step or ramp; and
 3. must be zero along ramps of constant slope.
- We define the second-order derivative of as the difference

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

6. Sharpening Spatial Filters

- Second-order derivative

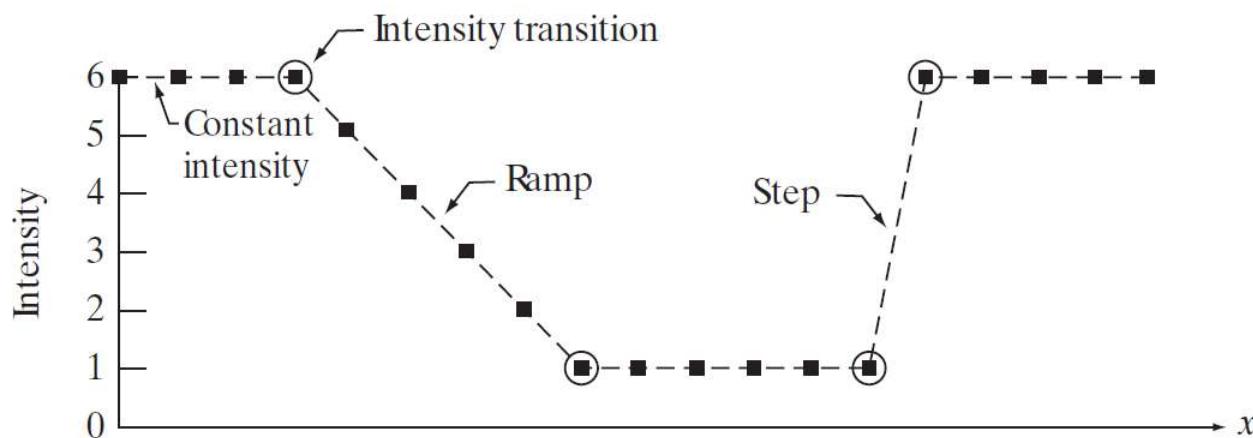
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

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

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

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

6. Sharpening Spatial Filters

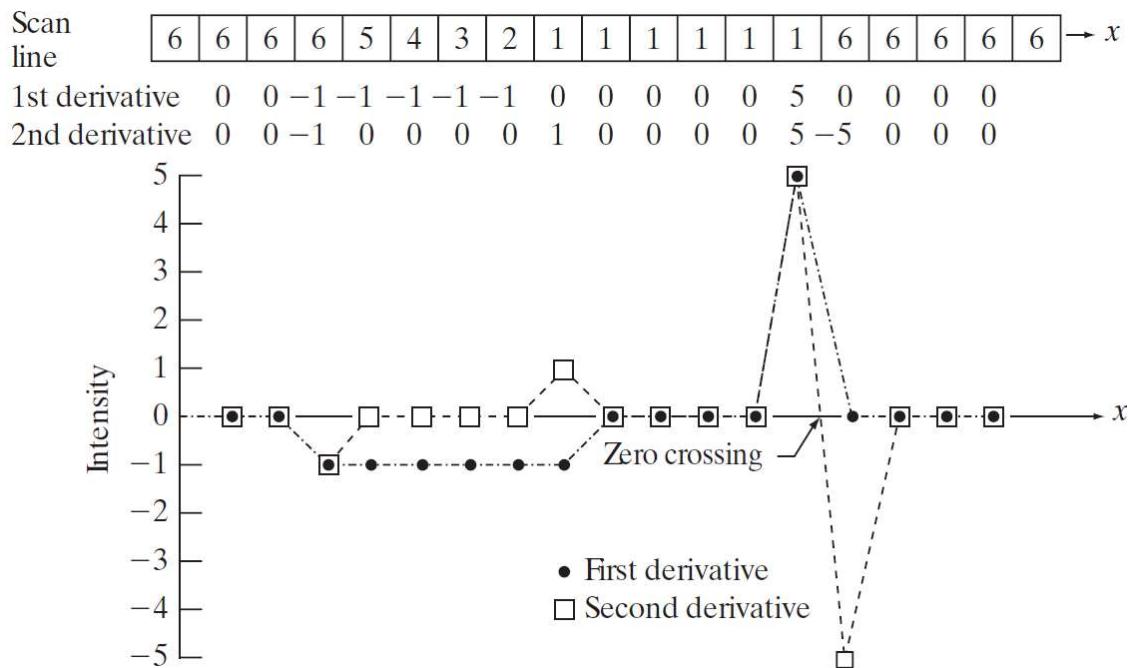


Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	6	6	6	6	$\rightarrow x$
1st derivative	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0	5	0	0	0	
2nd derivative	0	0	-1	0	0	0	0	1	0	0	0	0	0	5	-5	0	0	

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

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

6. Sharpening Spatial Filters



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

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

6. Sharpening Spatial Filters

- Using the second derivative – **The Laplacian:**
 - The approach basically consists of defining a **discrete** formulation of the **second-order derivative** and then constructing a **filter mask** based on that formulation.
 - We are interested in **isotropic filters**, whose response is independent of the direction of the discontinuities in the image to which the filter is applied.
 - The simplest isotropic derivative operator is the Laplacian, which, for a function of two variables (image), is defined as

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

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:

➤ We must express this equation in discrete form.

✓ In x-direction we have:

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

✓ In y-direction we have:

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

➤ It follows that the discrete Laplacian of two variables is

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

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:
 - The previous equation can be implemented using following filter masks

0	1	0
1	-4	1
0	1	0

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

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:
 - The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms

1	1	1
1	-8	1
1	1	1

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

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:
 - Because the Laplacian is a derivative operator, its use **highlights** intensity discontinuities in an image and **deemphasizes** regions with slowly varying intensity levels.
 - The basic way in which we use the Laplacian for image sharpening is

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

where f and g are the input and sharpened images, respectively.

6. Sharpening Spatial Filters

- Using the second derivative – The Laplacian:

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

➤ $c = -1$ if

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

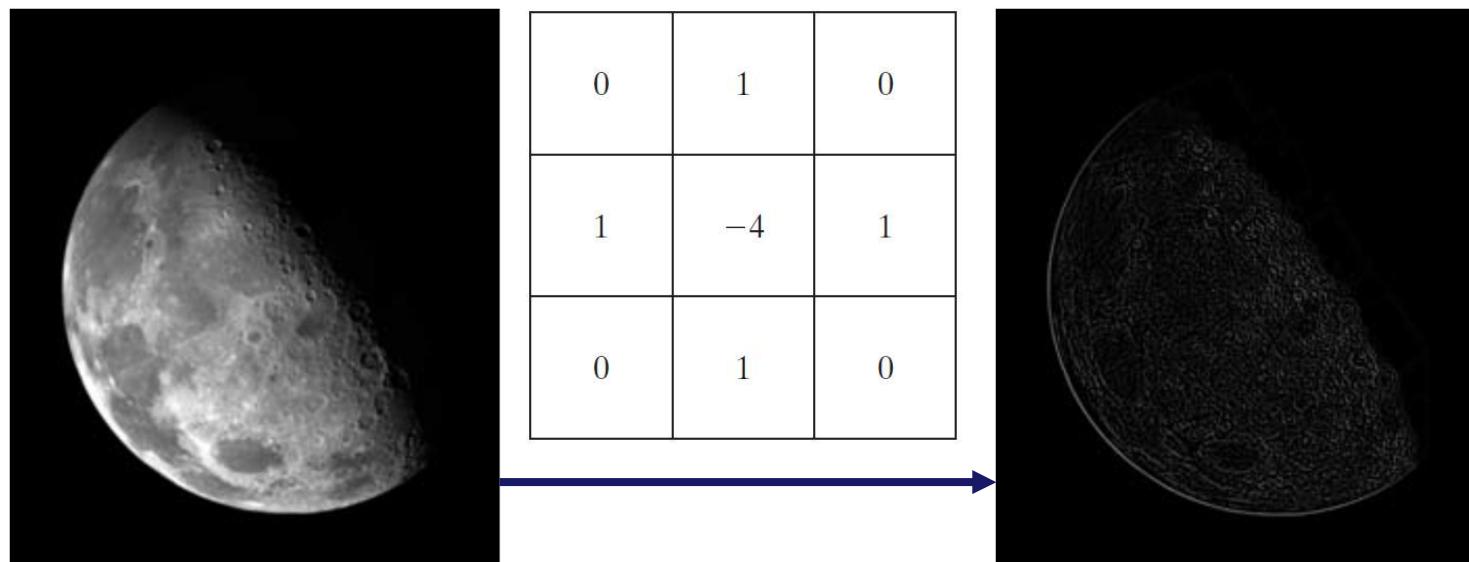
➤ $c = 1$ if

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

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

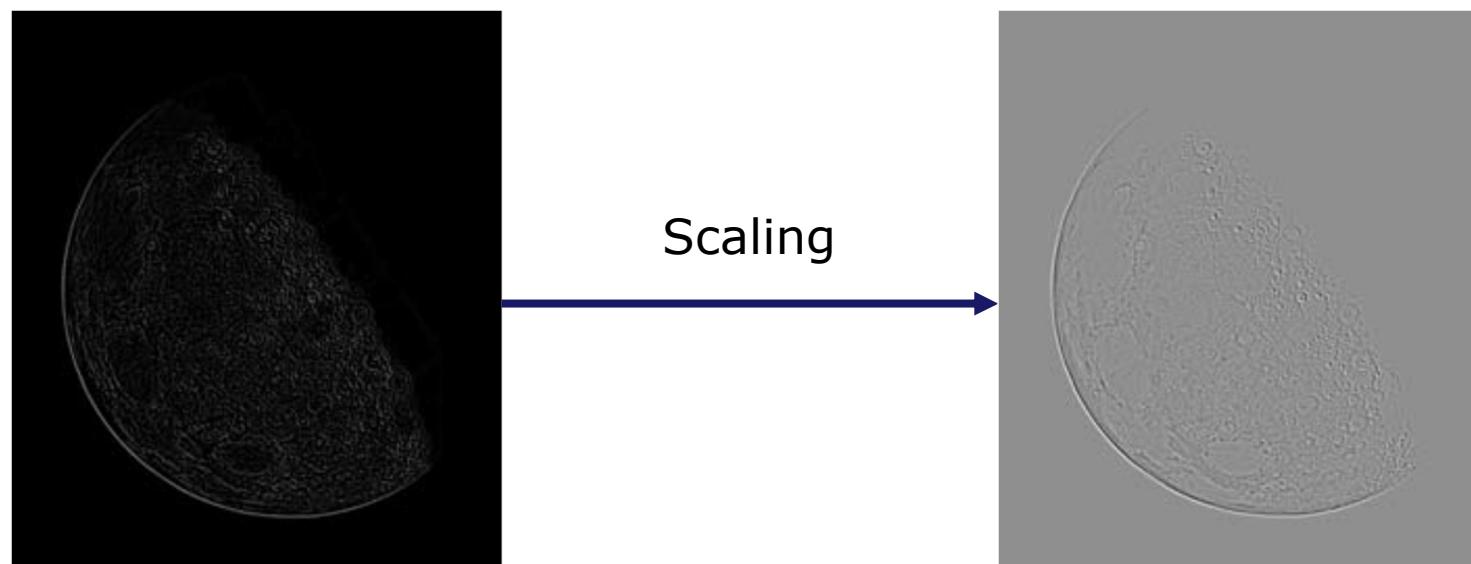
6. Sharpening Spatial Filters

- Example:



6. Sharpening Spatial Filters

- Example:



6. Sharpening Spatial Filters

- Example:

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



$g(x, y)$

$f(x, y)$

$c[\nabla^2 f(x, y)]$

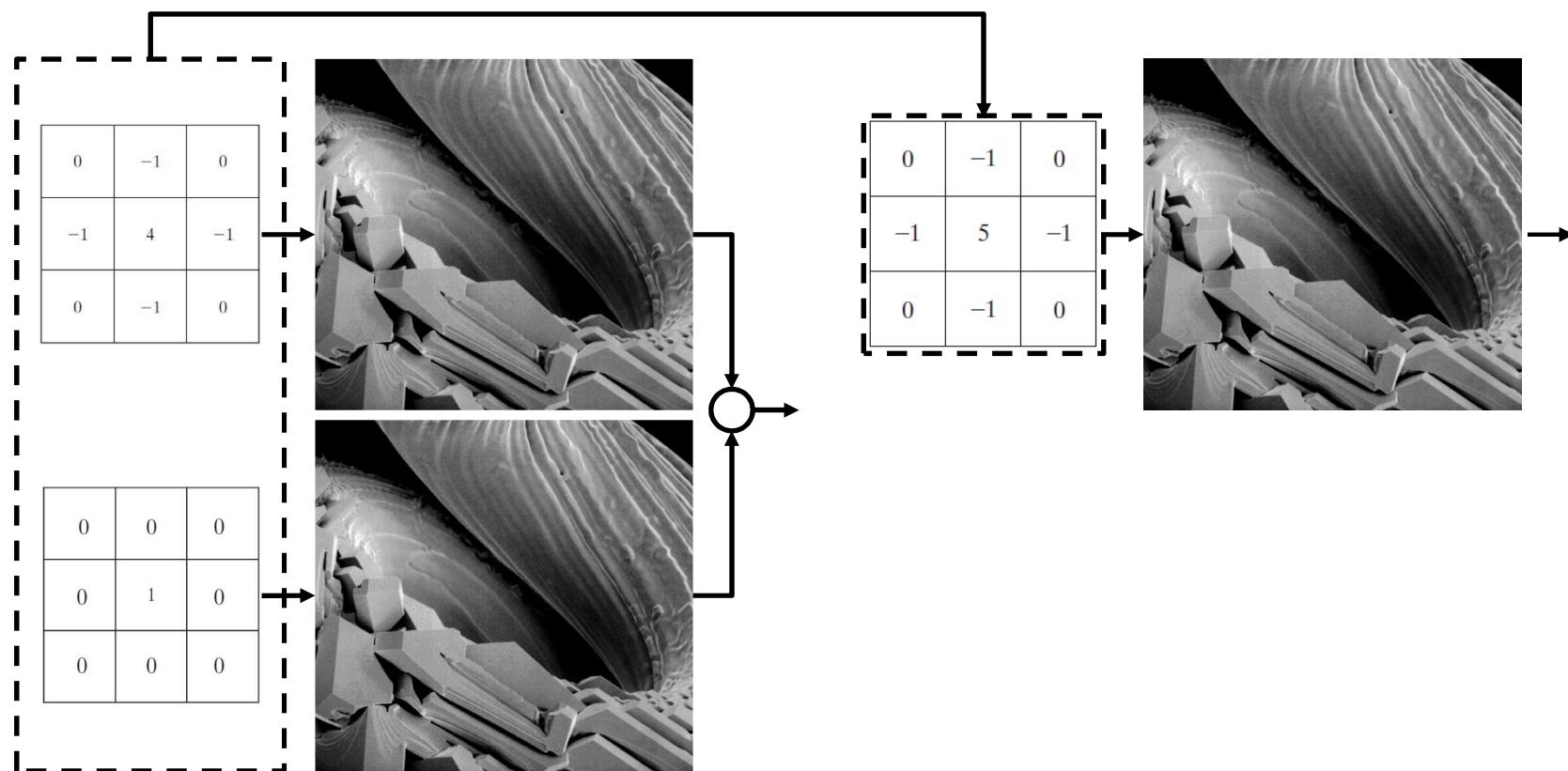
6. Sharpening Spatial Filters

- MATLAB: s126Laplacian.m



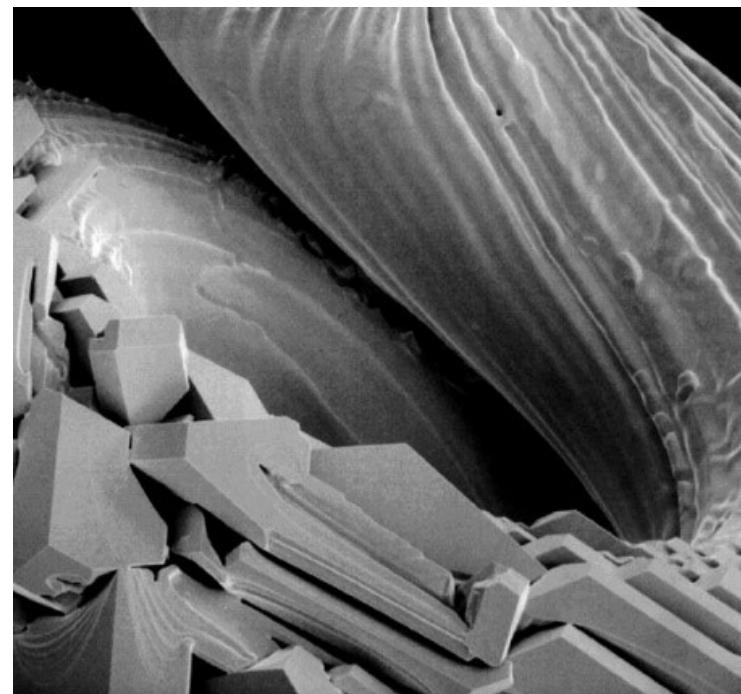
6. Sharpening Spatial Filters

- Simplification



6. Sharpening Spatial Filters

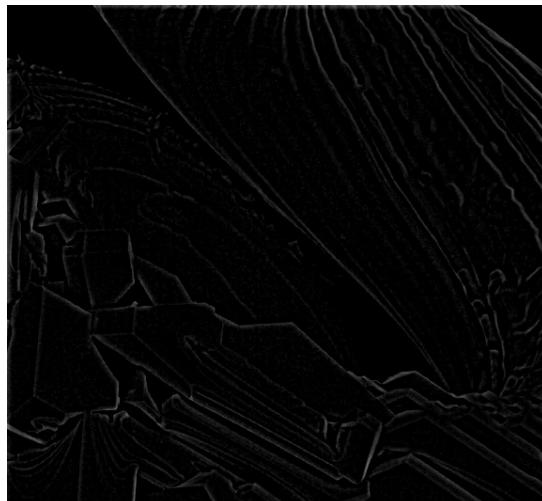
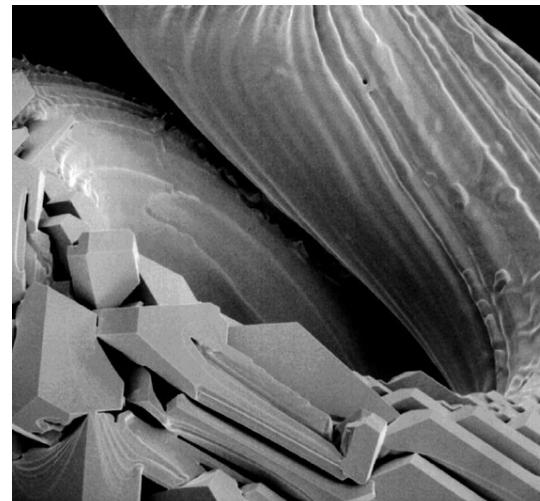
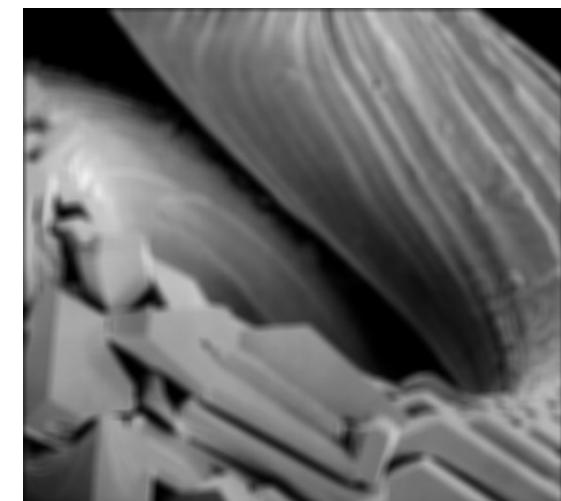
- MATLAB: s128LaplacianSimpl.m



6. Sharpening Spatial Filters

- **Unsharp Masking** Filtering: consists of subtracting an unsharp (smoothed) version of an image from the original image.

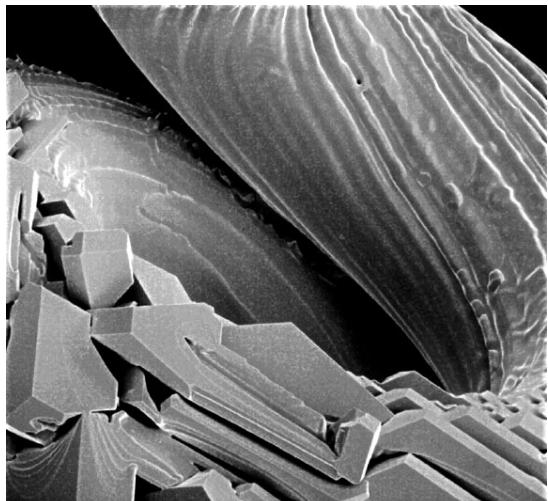
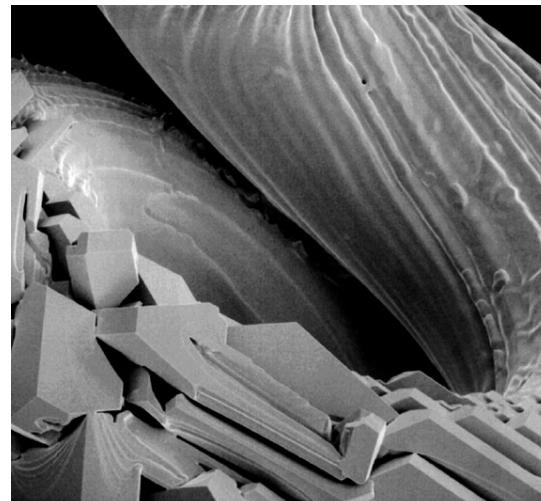
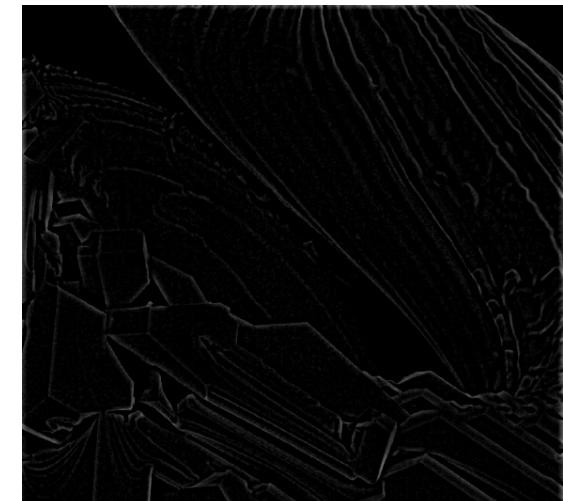
$$g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y)$$

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

6. Sharpening Spatial Filters

- Unsharp Masking Filtering ($0 > k \geq 1$)

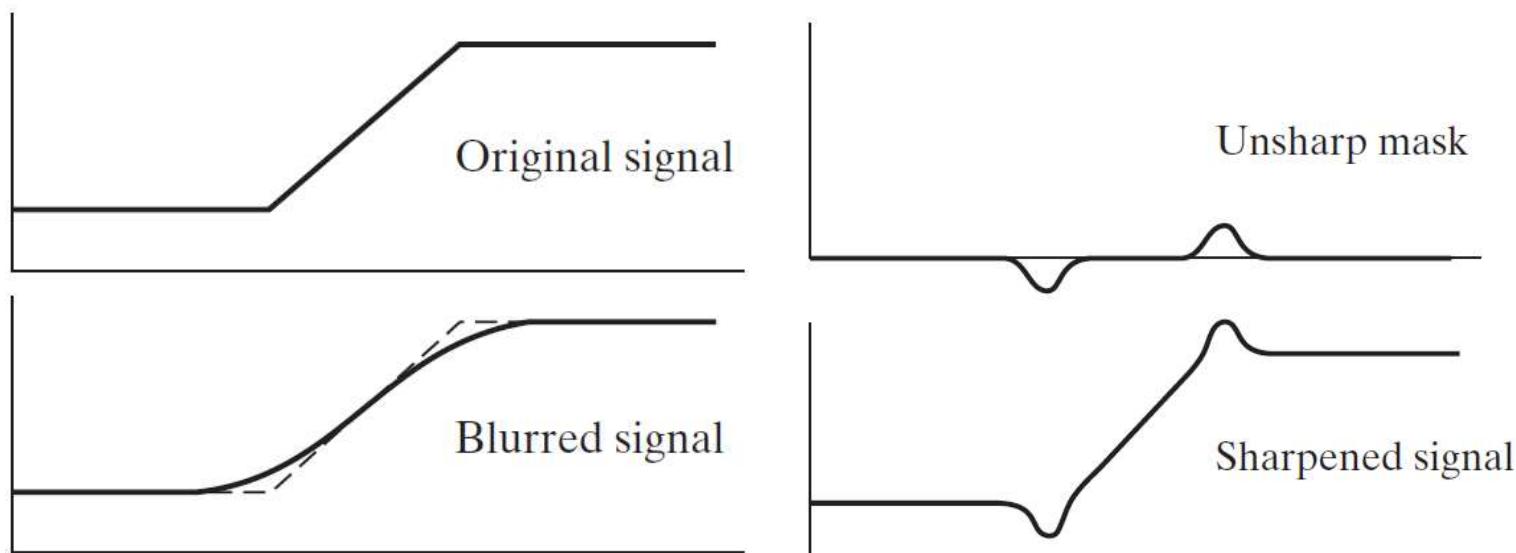
$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y)$$

 $g(x, y)$  $f(x, y)$  $g_{\text{mask}}(x, y)$

6. Sharpening Spatial Filters

- Unsharp Masking Filtering ($0 > k \geq 1$)

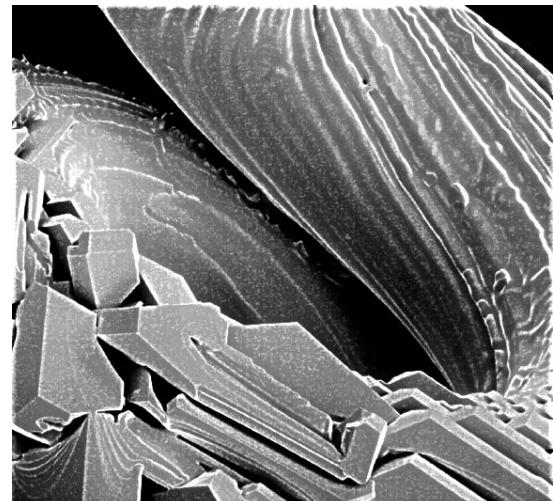
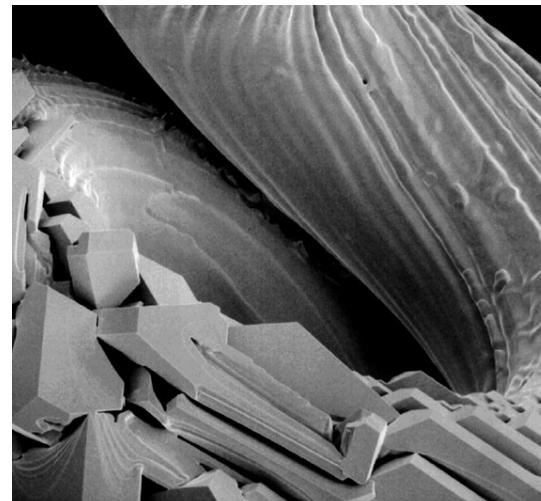
$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y)$$



6. Sharpening Spatial Filters

- **High-Boost** Filtering ($k > 1$)

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

 $g(x, y)$  $f(x, y)$  $k * g_{\text{mask}}(x, y)$

6. Sharpening Spatial Filters

- MATLAB: s133HighBoost.m



6. Sharpening Spatial Filters

- Using First-Order Derivatives - **The Gradient**

- First derivatives in image processing are implemented using the *magnitude of the gradient*.
- The gradient of a function f at coordinates (x, y) is defined as the two-dimensional column vector

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

- This vector points in the direction of the greatest rate of change of f at location (x, y) .

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient

- The **magnitude** of vector ∇f , denoted as $M(x, y)$, is

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

- It is common practice to refer to M as the *gradient image*.
 - In some implementations, it is more suitable computationally to approximate the squares and square root operations by absolute values:

$$M(x, y) \approx |g_x| + |g_y|$$

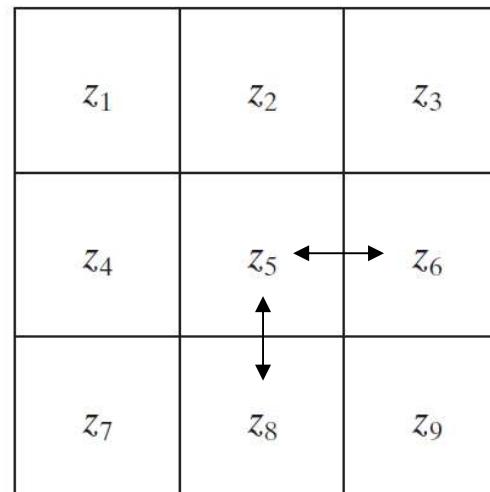
6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient
 - In order to simplify the discussion that follows, we will use the following notation:

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient
 - The simplest approximations to a first-order derivative are

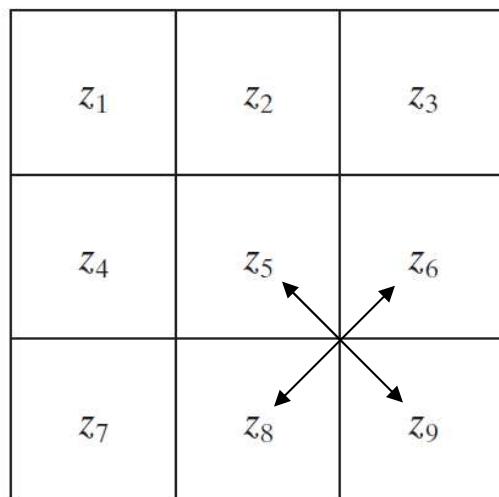


$$g_y = (z_6 - z_5)$$

$$g_x = (z_8 - z_5)$$

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient
 - Two other definitions proposed by Roberts [1965]
 - ✓ Roberts cross gradient operators



$$g_x = (z_9 - z_5) \quad g_y = (z_8 - z_6)$$

-1	0
0	1

6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient
 - Approximations using a 3x3 neighborhood centered on z_5 are as follows:
 - ✓ Sobel operators

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

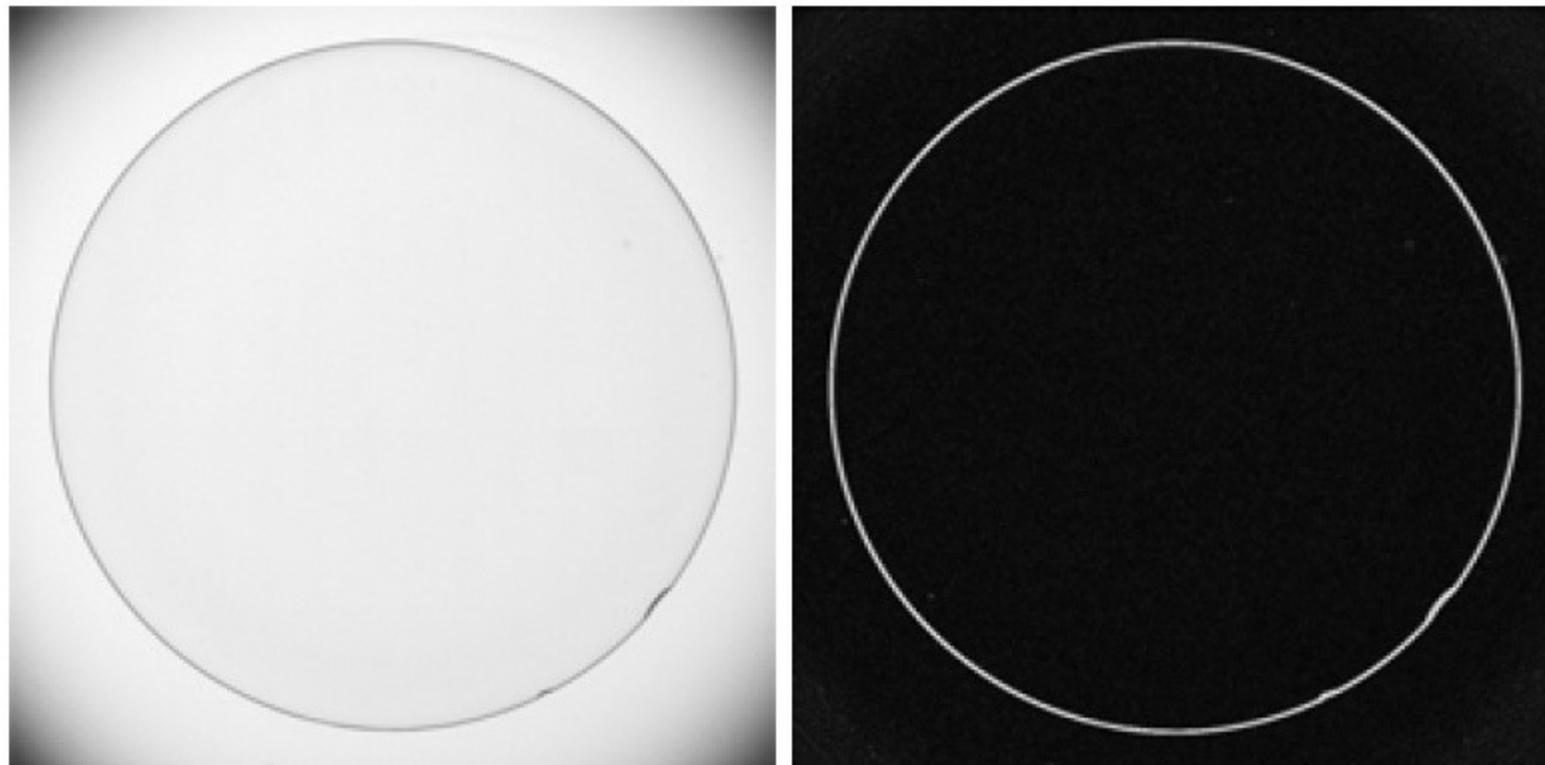
$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

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

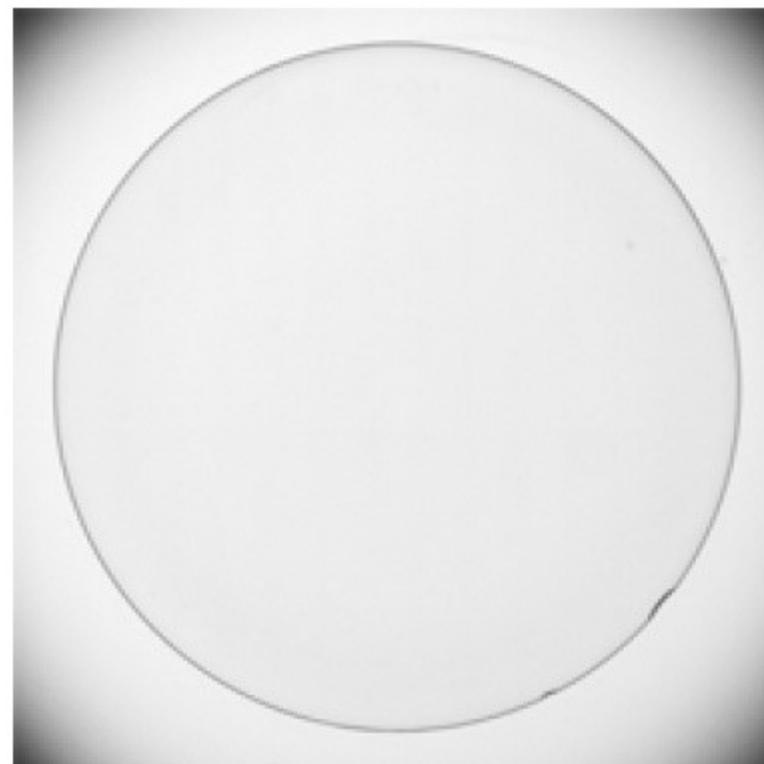
6. Sharpening Spatial Filters

- Using First-Order Derivatives - The Gradient



6. Sharpening Spatial Filters

- MATLAB: s141Sobel.m



Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 04

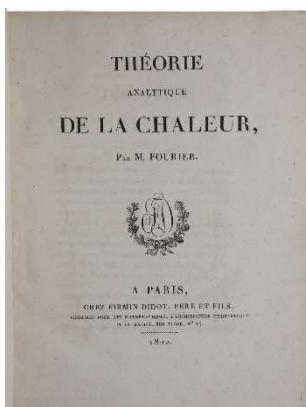
Filtering in the Frequency Domain

1. Background



- **Jean Baptiste Joseph Fourier** was a French mathematician born in 1768.
- The contribution for which he is most remembered was outlined in a memoir in 1807 and published in 1822 in his book, *La Théorie Analytique de la Chaleur* (**The Analytic Theory of Heat**).
- Basically, Fourier's contribution in this field states that **any periodic function** can be expressed as the **sum of sines and/or cosines** of different frequencies, each multiplied by a different coefficient (we now call this sum a **Fourier series**).
- 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 weighing function. The formulation in this case is the **Fourier transform**.

1. Background



- Fourier presented a paper in 1807 to the *Institut de France* on the use of sinusoids to represent temperature distributions.
- Among the reviewers were two of history's most famous mathematicians, Joseph Louis **Lagrange** (1736-1813), and Pierre Simon de **Laplace** (1749-1827).
- **Laplace** and the other reviewers **voted to publish the paper**.
- **Lagrange** adamantly **protested**.
- The *Institut de France* bowed to the prestige of Lagrange, and **rejected Fourier's work**.
- Only after Lagrange died that the paper was finally published, **some 15 years later**.

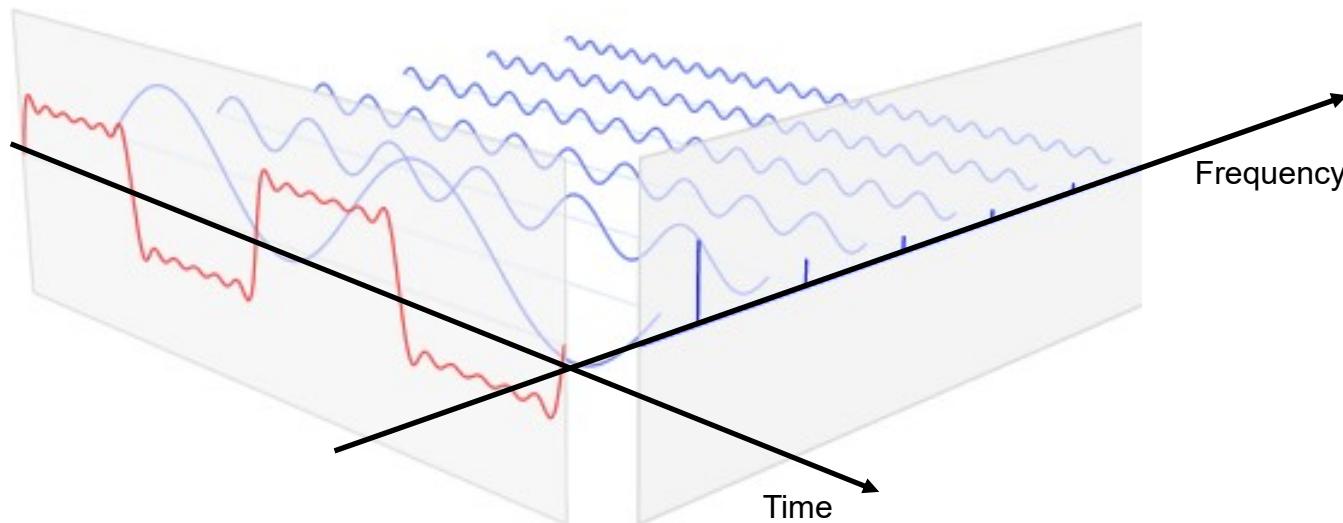
2. Preliminary Concepts

- Both representations share a important characteristic:
 - A function, expressed in either a Fourier series or transform, **can be reconstructed completely** via with no loss of information.



2. Preliminary Concepts

- The advent of digital **computers** and the “discovery” of a **fast Fourier transform** (FFT) algorithm in the early 1960s (more about this later) revolutionized the field of signal processing.



2. Preliminary Concepts

- Complex numbers

$$C = R + jI$$

- Conjugate

$$C^* = R - jI$$

- Polar coordinates

$$C = |C|(\cos \theta + j \sin \theta)$$

$$|C| = \sqrt{R^2 + I^2}$$

$$\theta = \arctan(I/R)$$

- Exponential form

$$e^{j\theta} = \cos \theta + j \sin \theta \longrightarrow C = |C| e^{j\theta}$$

2. Preliminary Concepts

- The preceding equations are applicable also to complex functions.

$$F(u) = R(u) + jI(u)$$

$$|F(u)| = \sqrt{R(u)^2 + I(u)^2}$$

$$\theta(u) = \arctan[I(u)/R(u)]$$

- A function $f(t)$ of a continuous variable t that is periodic with period, T , can be expressed as the **sum of sines and cosines multiplied by appropriate coefficients**.

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j \frac{2\pi n}{T} t}$$

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j \frac{2\pi n}{T} t} dt \quad \text{for } n = 0, \pm 1, \pm 2, \dots$$

2. Preliminary Concepts

- Fourier series (Trigonometric):

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi n}{T}t\right) + b_n \sin\left(\frac{2\pi n}{T}t\right) \right]$$

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nw_0 t) + b_n \sin(nw_0 t)] \quad w_0 = \frac{2\pi}{T}$$

$$a_0 = \frac{2}{T} \int_0^T f(t) dt$$

$$a_n = \frac{2}{T} \int_0^T f(t) \cos(nw_0 t) dt$$

$$b_n = \frac{2}{T} \int_0^T f(t) \sin(nw_0 t) dt$$

2. Preliminary Concepts

- Fourier series (Exponential):

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nw_0 t) + b_n \sin(nw_0 t)]$$

$$\begin{aligned} e^{jn w_0 t} &= \cos(n w_0 t) + j \sin(n w_0 t) & \cos(n w_0 t) &= \frac{1}{2} (e^{j n w_0 t} + e^{-j n w_0 t}) \\ e^{-j n w_0 t} &= \cos(n w_0 t) - j \sin(n w_0 t) & \xrightarrow{\hspace{1cm}} & \sin(n w_0 t) = \frac{1}{2j} (e^{j n w_0 t} - e^{-j n w_0 t}) \end{aligned}$$

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[\frac{a_n}{2} (e^{j n w_0 t} + e^{-j n w_0 t}) + \frac{b_n}{2j} (e^{j n w_0 t} - e^{-j n w_0 t}) \right]$$

2. Preliminary Concepts

- Fourier series (Exponential):

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[\frac{a_n}{2} (e^{jn\omega_0 t} + e^{-jn\omega_0 t}) + \frac{b_n}{2j} (e^{jn\omega_0 t} - e^{-jn\omega_0 t}) \right]$$

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[\left(\frac{a_n - jb_n}{2} \right) e^{jn\omega_0 t} + \left(\frac{a_n + jb_n}{2} \right) e^{-jn\omega_0 t} \right]$$

2. Preliminary Concepts

- Fourier series (Exponential):

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[\left(\frac{a_n - jb_n}{2} \right) e^{jn\omega_0 t} + \left(\frac{a_n + jb_n}{2} \right) e^{-jn\omega_0 t} \right]$$

- Taking

$$n \geq 1$$

$$c_0 = \frac{1}{2}a_0, \quad c_n = \frac{1}{2}(a_n - jb_n), \quad c_{-n} = \frac{1}{2}(a_n + jb_n)$$

- It follows that

$$f(t) = c_0 + \sum_{n=1}^{\infty} c_n e^{jn\omega_0 t} + \sum_{n=1}^{\infty} c_{-n} e^{-jn\omega_0 t}$$

2. Preliminary Concepts

- Fourier series (Exponential):

$$f(t) = c_0 + \sum_{n=1}^{\infty} c_n e^{jn\omega_0 t} + \sum_{n=1}^{\infty} c_{-n} e^{-jn\omega_0 t}$$

$$f(t) = \sum_{n=1}^{\infty} c_{-n} e^{-jn\omega_0 t} + c_0 e^{j0\omega_0 t} + \sum_{n=1}^{\infty} c_n e^{jn\omega_0 t}$$

$$f(t) = \sum_{n=-\infty}^{-1} c_n e^{-jn\omega_0 t} + c_0 e^{j0\omega_0 t} + \sum_{n=1}^{\infty} c_n e^{jn\omega_0 t}$$

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

2. Preliminary Concepts

- Fourier series (Exponential):

$$c_n = \frac{1}{2} (a_n - jb_n)$$

$$a_n = \frac{2}{T} \int_0^T f(t) \cos(nw_0 t) dt$$

$$b_n = \frac{2}{T} \int_0^T f(t) \sin(nw_0 t) dt$$

$$c_n = \frac{1}{2} \left[\frac{2}{T} \int_0^T f(t) \cos(nw_0 t) dt - j \frac{2}{T} \int_0^T f(t) \sin(nw_0 t) dt \right]$$

$$= \frac{1}{T} \int_0^T f(t) [\cos(nw_0 t) - j \sin(nw_0 t)] dt$$

2. Preliminary Concepts

- Fourier series (Exponential):

$$c_n = \frac{1}{T} \int_0^T f(t) [\cos(nw_0 t) - j \sin(nw_0 t)] dt$$

$$= \frac{1}{T} \int_0^T f(t) e^{-jnw_0 t} dt$$

2. Preliminary Concepts

- Fourier series (Exponential):
- In summary

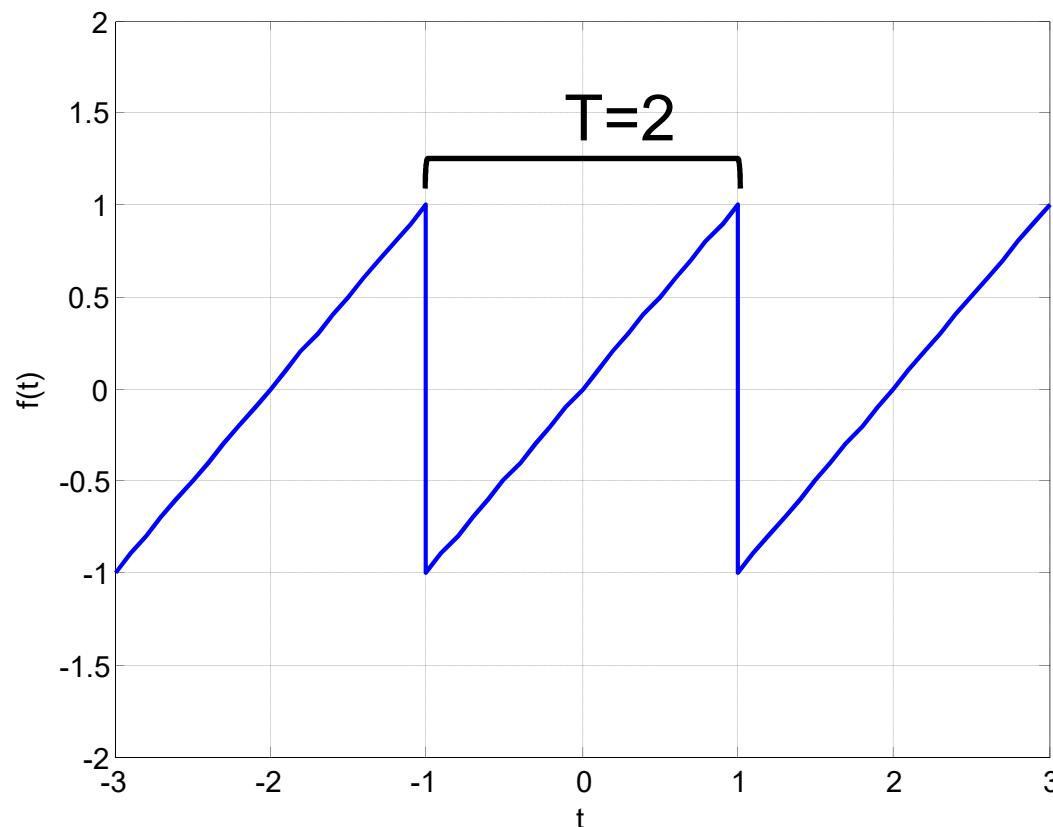
$$f(t) = \sum_{n=-\infty}^{\infty} F(n)e^{jw_0nt} \quad w_0 = \frac{2\pi}{T}$$
$$F(n) = \frac{1}{T} \int_0^T f(t)e^{-jw_0nt} dt, \quad n = 0, \pm 1, \pm 2, \dots$$

$f(t)$ is complex, continuous and periodic with period T

$F(n)$ is complex, discrete and non-periodic

2. Preliminary Concepts

- Example:



2. Preliminary Concepts

- Example:

$$\begin{aligned} F(n) &= \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j\frac{2\pi n}{T}t} dt = \frac{1}{2} \int_{-1}^1 t e^{-j\frac{2\pi n}{2}t} dt = \\ &= \frac{1}{2} \int_{-1}^1 t [\cos(\pi nt) - j \sin(\pi nt)] dt \\ &= \frac{1}{2} \int_{-1}^1 t \cos(\cancel{\pi nt}) dt - \frac{j}{2} \int_{-1}^1 t \sin(\pi nt) dt \end{aligned}$$

2. Preliminary Concepts

- Example:

$$F(n) = -\frac{j}{2} \int_{-1}^1 t \sin(\pi n t) dt = -j \int_0^1 t \sin(\pi n t) dt$$

$$= -j \left[\frac{\sin(\pi n t)}{(\pi n)^2} - t \frac{\cos(\pi n t)}{\pi n} \right]_0^1$$

$$= -j \left[\frac{\sin(\pi n) - \pi n \cos(\pi n)}{(\pi n)^2} \right]$$

$$\boxed{\int x \sin ax dx = -\frac{\sin ax}{a^2} - \frac{x \cos ax}{a} + C}$$

2. Preliminary Concepts

- Example:

$$\begin{aligned} &= -j \left[\frac{\sin(\pi n) - \pi n \cos(\pi n)}{(\pi n)^2} \right] \\ &= -j \left[\frac{-\cos(\pi n)}{\pi n} \right] = -j \left[\frac{-(-1)^n}{\pi n} \right] \end{aligned}$$

$$F(n) = j \frac{(-1)^n}{\pi n}, n \neq 0$$

$$F(0) = \frac{1}{T} \int_0^T f(t) e^{-j\omega_0 n t} dt = \int_{-1}^1 t dt = 0, n = 0$$

2. Preliminary Concepts

- Example:

$$\begin{aligned} f(t) &= \sum_{n=-\infty}^{\infty} F(n)e^{j\frac{2\pi n}{T}t} = \sum_{n=-\infty}^{\infty} j \frac{(-1)^n}{\pi n} e^{j\pi nt} \\ &= \sum_{n=-\infty}^{\infty} j \frac{(-1)^n}{\pi n} [\cos(\pi nt) + j \sin(\pi nt)], \quad n \neq 0 \\ &= \sum_{n=-\infty}^{\infty} j \frac{(-1)^n}{\pi n} \cos(\pi nt) - \sum_{n=-\infty}^{\infty} \frac{(-1)^n}{\pi n} \sin(\pi nt) \end{aligned}$$

0 

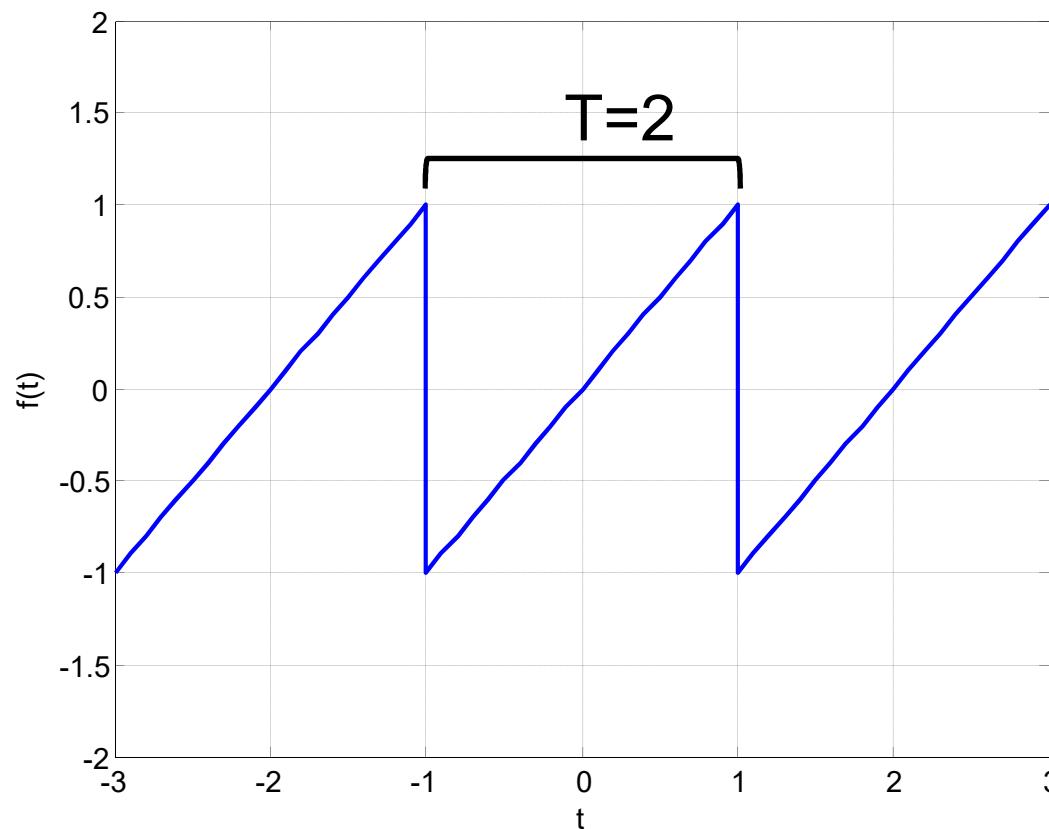
2. Preliminary Concepts

- Example

$$\begin{aligned}f(t) &= -\sum_{n=-\infty}^{\infty} \frac{(-1)^n}{\pi n} \sin(\pi n t) \\&= -2 \sum_{n=1}^{\infty} \frac{(-1)^n}{\pi n} \sin(\pi n t)\end{aligned}$$

2. Preliminary Concepts

- MATLAB: s23FourierSeriesSawtooth.m



2. Preliminary Concepts

- Impulses and Their Sifting Property
 - A **unit impulse** of a continuous variable t is defined as

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases}$$

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

- An impulse may be viewed as a spike of infinity amplitude and zero duration, having unit area.

2. Preliminary Concepts

- Impulses and Their Sifting Property
 - An impulse has the so called **sifting property** with respect to integration

$$\int_{-\infty}^{\infty} f(t) \delta(t) dt = f(0)$$

provided that $f(t)$ is continuous at $t = 0$.

- A more general statement

$$\int_{-\infty}^{\infty} f(t) \delta(t - t_0) dt = f(t_0)$$

2. Preliminary Concepts

- Impulses and Their Sifting Property

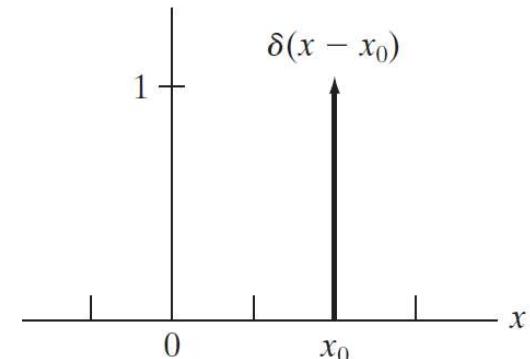
- The **unit discrete impulse**

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases} \quad \sum_{x=-\infty}^{\infty} \delta(x) = 1$$

- *The sifting property*

$$\sum_{x=-\infty}^{\infty} f(x) \delta(x) = f(0)$$

$$\sum_{x=-\infty}^{\infty} f(x) \delta(x - x_0) = f(x_0)$$

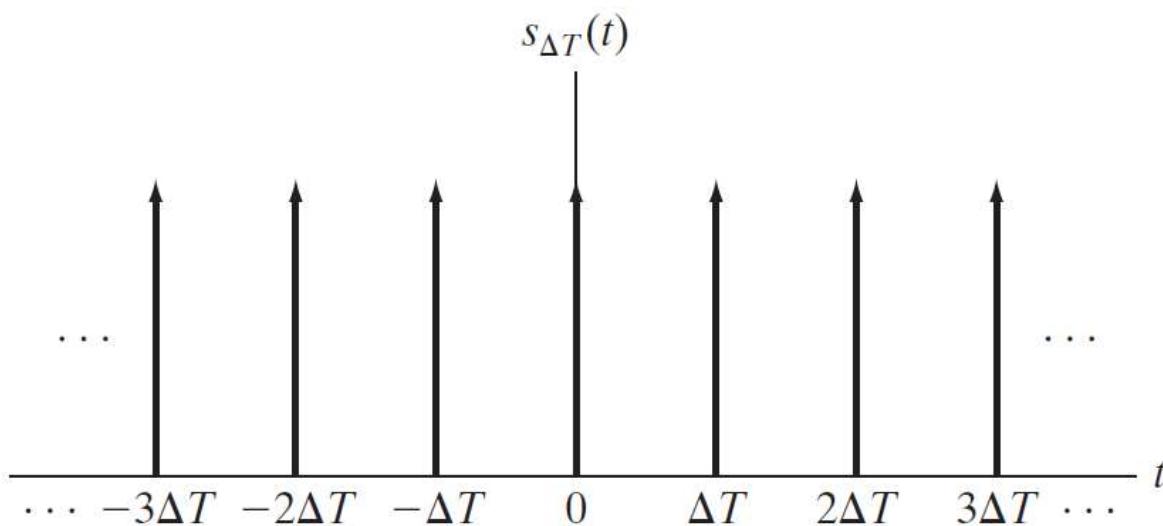


2. Preliminary Concepts

- Impulses and Their Sifting Property

- **Impulse train**

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$



2. Preliminary Concepts

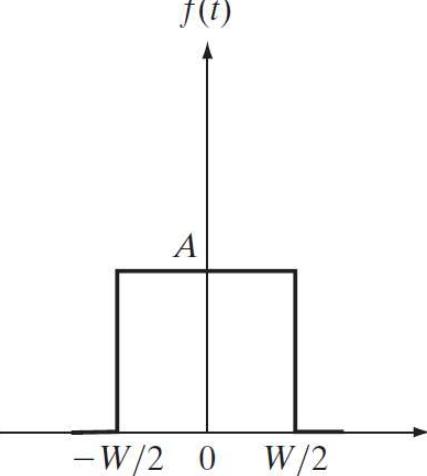
- The Fourier Transform of Functions of One Continuous Variable

$$\mathfrak{F}\{f(t)\} = F(\mu) = \int_{-\infty}^{\infty} f(t) e^{-j2\pi\mu t} dt$$

$$\mathfrak{F}^{-1}\{F(\mu)\} = f(t) = \int_{-\infty}^{\infty} F(\mu) e^{j2\pi\mu t} d\mu$$

2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable

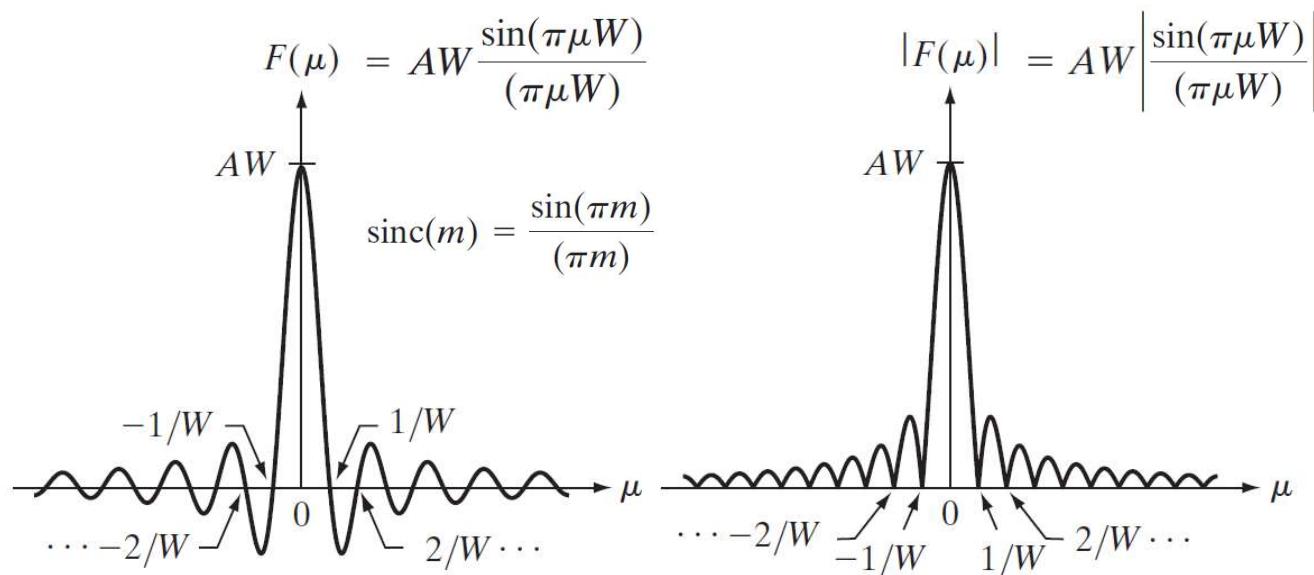


$$\begin{aligned}
 F(\mu) &= \int_{-\infty}^{\infty} f(t) e^{-j2\pi\mu t} dt = \int_{-W/2}^{W/2} A e^{-j2\pi\mu t} dt \\
 &= \frac{-A}{j2\pi\mu} [e^{-j2\pi\mu t}]_{-W/2}^{W/2} = \frac{-A}{j2\pi\mu} [e^{-j\pi\mu W} - e^{j\pi\mu W}] \\
 &= \frac{A}{j2\pi\mu} [e^{j\pi\mu W} - e^{-j\pi\mu W}] \\
 &= AW \frac{\sin(\pi\mu W)}{(\pi\mu W)}
 \end{aligned}$$

$\int e^{cx} dx = \frac{1}{c} e^{cx}$

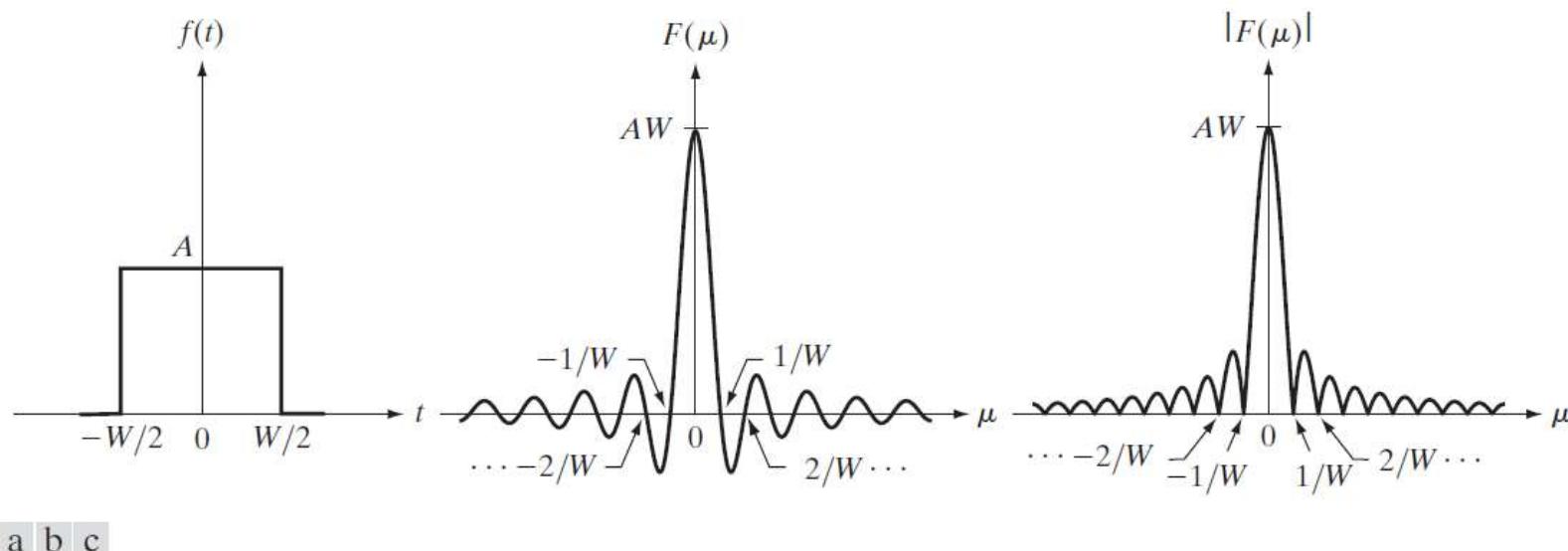
2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable
 - In general, the Fourier transform contains complex terms, and it is customary for display purposes to work with the magnitude of the transform, which is called the **frequency spectrum**.



2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable



a b c

FIGURE 4.4 (a) A simple function; (b) its Fourier transform; and (c) the spectrum. All functions extend to infinity in both directions.

2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable
 - The Fourier transform of a unit impulse located at the origin

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t) e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{\infty} e^{-j2\pi\mu t} \delta(t) dt \\ &= e^{-j2\pi\mu 0} = e^0 \\ &= 1 \end{aligned}$$

2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable
 - Similarly, the Fourier transform of an impulse located at $t = t_0$

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} \delta(t - t_0) e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{\infty} e^{-j2\pi\mu t} \delta(t - t_0) dt \\ &= e^{-j2\pi\mu t_0} \\ &= \cos(2\pi\mu t_0) - j \sin(2\pi\mu t_0) \end{aligned}$$

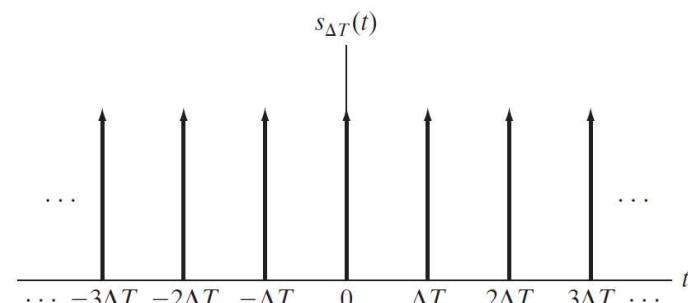
2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable
 - The **impulse train** below is periodic so that it can be expressed as a Fourier series

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j \frac{2\pi n}{\Delta T} t}$$

$$c_n = \frac{1}{\Delta T} \int_{-\Delta T/2}^{\Delta T/2} s_{\Delta T}(t) e^{-j \frac{2\pi n}{\Delta T} t} dt$$



2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable

➤ The preceding equation becomes

$$c_n = \frac{1}{\Delta T} \int_{-\Delta T/2}^{\Delta T/2} \delta(t) e^{-j\frac{2\pi n}{\Delta T}t} dt$$

$$= \frac{1}{\Delta T} e^0$$

$$= \frac{1}{\Delta T}$$

➤ The Fourier series expansion then becomes

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{\Delta T}t} \longrightarrow s_{\Delta T}(t) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}$$

2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable

- Our objective is to obtain the Fourier transform of this expression

$$s_{\Delta T}(t) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{j \frac{2\pi n}{\Delta T} t}$$

- Because summation is a linear process, obtaining the **Fourier transform of a sum** is the same as obtaining the **sum of the transforms** of the individual components.

$$\Im \left\{ e^{j \frac{2\pi n}{\Delta T} t} \right\}$$

2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable

- Suppose we have $f(t)$ and we know the Fourier Transform of a function $g(t)$

$$f(t) = e^{j\frac{2\pi}{\Delta T}t} \quad G(\mu) = \delta(\mu - 1/\Delta T)$$

then

$$\begin{aligned} g(t) &= \int_{-\infty}^{+\infty} G(\mu) e^{j2\pi\mu t} d\mu \\ &= \int_{-\infty}^{+\infty} \delta(\mu - 1/\Delta T) e^{j2\pi\mu t} d\mu = e^{j\frac{2\pi}{\Delta T}t} \end{aligned}$$

It follows that $f(t) = g(t)$

2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable

➤ If $f(t) = g(t)$

then $\mathcal{F}\{f(t)\} = \mathcal{F}\{g(t)\}$

and $f(t) = e^{j\frac{2\pi}{\Delta T}t} \longleftrightarrow F(\mu) = \delta(\mu - 1/\Delta T)$

2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable

➤ Therefore, if

$$f(t) = e^{j\frac{2\pi}{\Delta T}t} \longleftrightarrow F(\mu) = \delta(\mu - 1/\Delta T)$$

and

$$s_{\Delta T}(t) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}$$

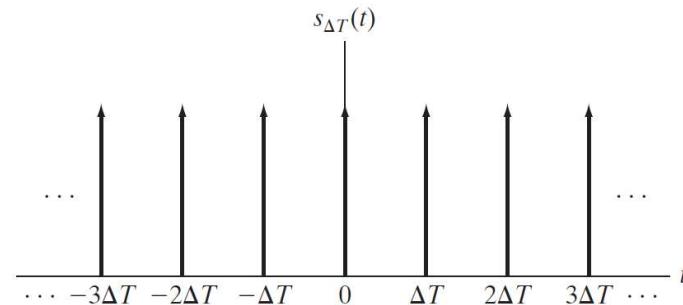
then

$$\begin{aligned} S(\mu) &= \Im\{s_{\Delta T}(t)\} = \Im\left\{\frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}\right\} = \frac{1}{\Delta T} \Im\left\{\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T}t}\right\} \\ &= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right) \end{aligned}$$

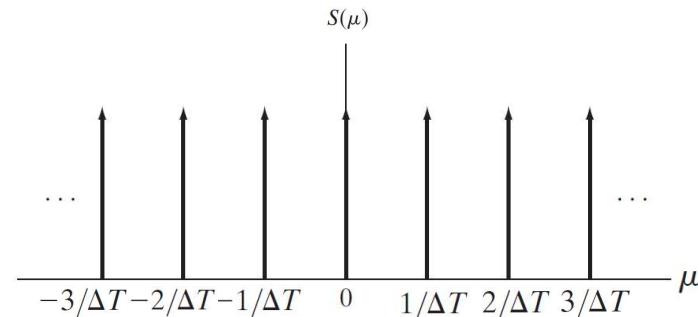
2. Preliminary Concepts

- The Fourier Transform of Functions of One Continuous Variable

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$



$$S(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right)$$



2. Preliminary Concepts

- Convolution

$$f(t) \star h(t) = \int_{-\infty}^{\infty} f(\tau) h(t - \tau) d\tau$$

- Fourier transform of the convolution

$$\begin{aligned}\Im\{f(t) \star h(t)\} &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\tau) h(t - \tau) d\tau \right] e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{\infty} f(\tau) \left[\int_{-\infty}^{\infty} h(t - \tau) e^{-j2\pi\mu t} dt \right] d\tau\end{aligned}$$

2. Preliminary Concepts

- Convolution

- It can be shown that (exercise),

$$\Im\{h(t - \tau)\} = H(\mu)e^{-j2\pi\mu\tau}$$

- From the previous equations

$$\Im\{f(t) \star h(t)\} = \int_{-\infty}^{\infty} f(\tau) [H(\mu)e^{-j2\pi\mu\tau}] d\tau$$

$$= H(\mu) \int_{-\infty}^{\infty} f(\tau) e^{-j2\pi\mu\tau} d\tau$$

$$= H(\mu) F(\mu)$$

2. Preliminary Concepts

- Convolution

- This result is one-half of the **convolution theorem** and is written as

$$f(t) \star h(t) \Leftrightarrow H(\mu) F(\mu)$$

- Following a similar development would result in the other half of the convolution theorem:

$$f(t)h(t) \Leftrightarrow H(\mu) \star F(\mu)$$

3. Sampling and Fourier Transform of Sampled Functions

- Sampling

- One way to model sampling is to multiply $f(t)$ by a **sampling function** equal to a train of impulses ΔT units apart

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t) \delta(t - n\Delta T)$$

- The **value** of each sample is then given by the “strength” of the weighted impulse, which we obtain by integration. That is, the value, f_k , of an arbitrary sample in the sequence is given by

$$f_k = \int_{-\infty}^{\infty} f(t) \delta(t - k\Delta T) dt = f(k\Delta T)$$

$$k = \dots, -2, -1, 0, 1, 2, \dots$$

3. Sampling and Fourier Transform of Sampled Functions

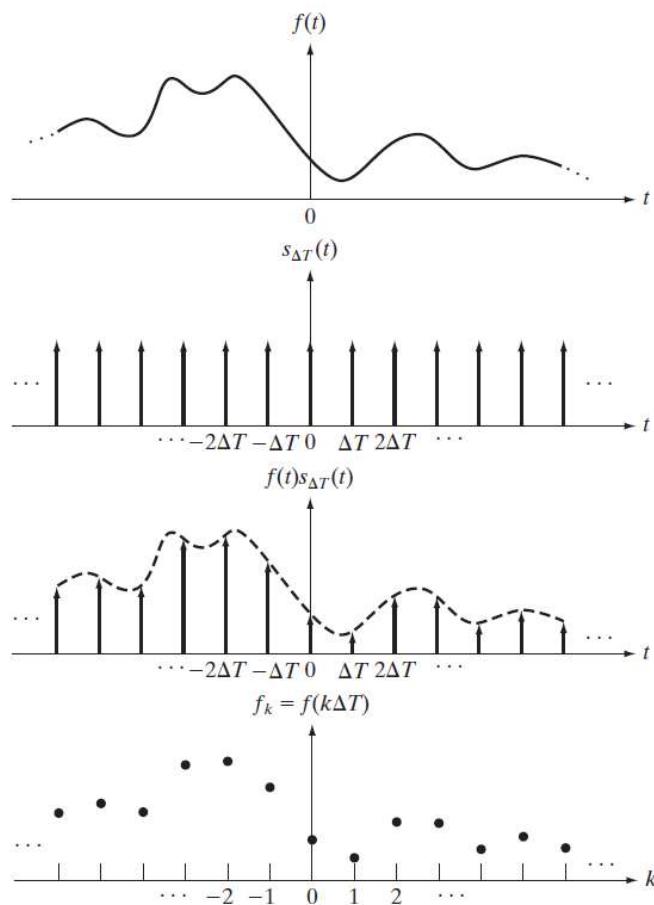
- Sampling

$$f(t)$$

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T)$$

$$\begin{aligned} f_k &= \int_{-\infty}^{\infty} f(t)\delta(t - k\Delta T) dt \\ &= f(k\Delta T) \end{aligned}$$



a
b
c
d

FIGURE 4.5
 (a) A continuous function.
 (b) Train of impulses used to model the sampling process.
 (c) Sampled function formed as the product of (a) and (b).
 (d) Sample values obtained by integration and using the sifting property of the impulse. (The dashed line in (c) is shown for reference. It is not part of the data.)

3. Sampling and Fourier Transform of Sampled Functions

- Fourier Transform of Sampled Functions

➤ From the convolution theorem

$$\begin{aligned}\tilde{F}(\mu) &= \Im\{\tilde{f}(t)\} \\ &= \Im\{f(t)s_{\Delta T}(t)\} \\ &= F(\mu) \star S(\mu)\end{aligned}$$

➤ And we know that

$$S(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right)$$

3. Sampling and Fourier Transform of Sampled Functions

- Fourier Transform of Sampled Functions

➤ But directly from the definition

$$\begin{aligned}\tilde{F}(\mu) &= F(\mu) \star S(\mu) \\ &= \int_{-\infty}^{\infty} F(\tau) S(\mu - \tau) d\tau & S(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right) \\ &= \frac{1}{\Delta T} \int_{-\infty}^{\infty} F(\tau) \sum_{n=-\infty}^{\infty} \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\ &= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} F(\tau) \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\ &= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F\left(\mu - \frac{n}{\Delta T}\right)\end{aligned}$$

3. Sampling and Fourier Transform of Sampled Functions

- Fourier Transform of Sampled Functions

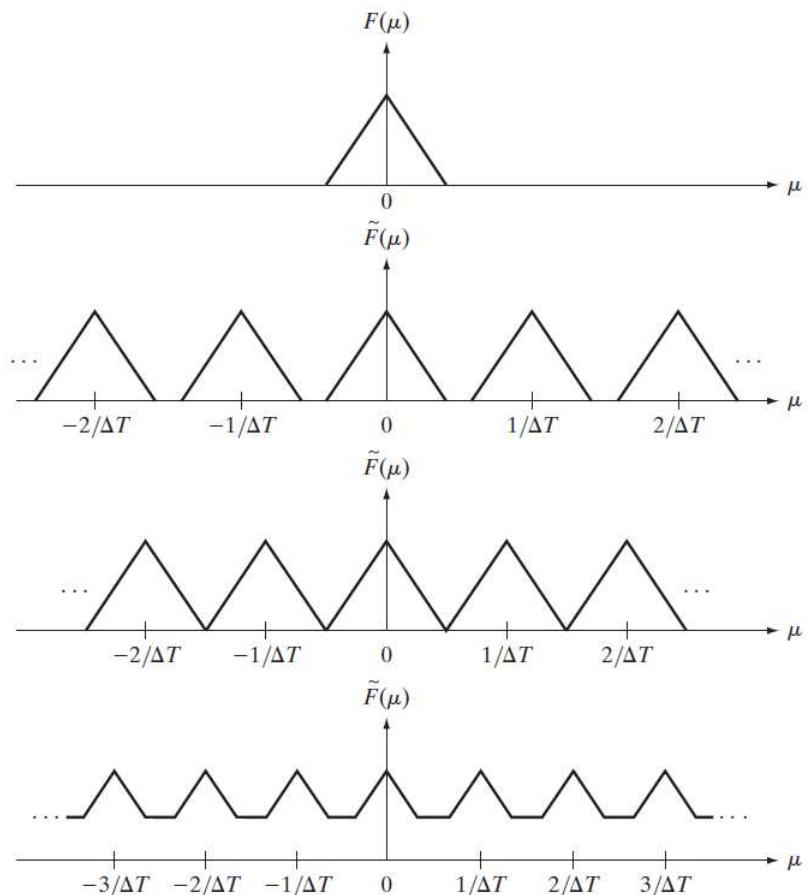
- This summation shows that the Fourier transform of a sampled function is an **infinite, periodic** sequence of **copies** of $F(\mu)$, the transform of the original continuous function $f(t)$.

$$\tilde{F}(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F\left(\mu - \frac{n}{\Delta T}\right)$$

- The separation between copies is determined by the value of $1/\Delta T$.
- Observe that although $\tilde{f}(t)$ is a sampled function, its transform $\tilde{F}(\mu)$ is continuous because it consists of copies of $F(\mu)$ which is a continuous function.

3. Sampling and Fourier Transform of Sampled Functions

- The Sampling Theorem



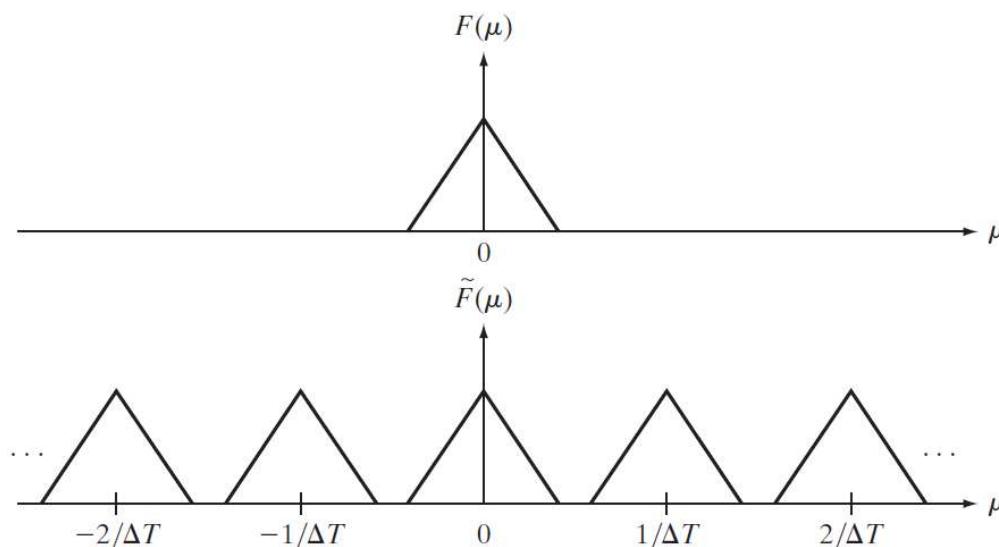
a
b
c
d

FIGURE 4.6

(a) Fourier transform of a band-limited function.
 (b)–(d)
 Transforms of the corresponding sampled function under the conditions of over-sampling, critically-sampling, and under-sampling, respectively.

3. Sampling and Fourier Transform of Sampled Functions

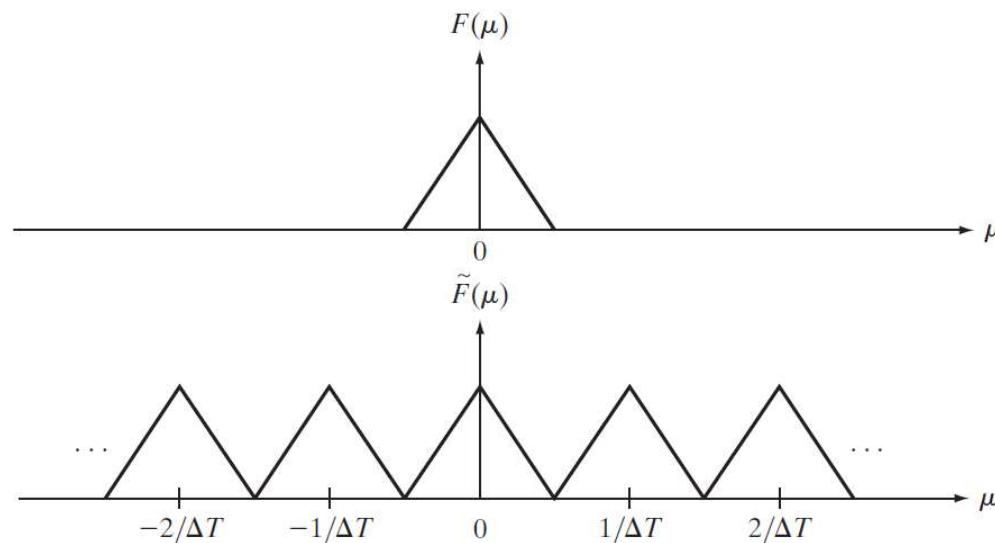
- The Sampling Theorem



- **Over-sampling:** the sampling rate was high enough to provide sufficient separation between the periods and thus preserve the integrity of $F(\mu)$.

3. Sampling and Fourier Transform of Sampled Functions

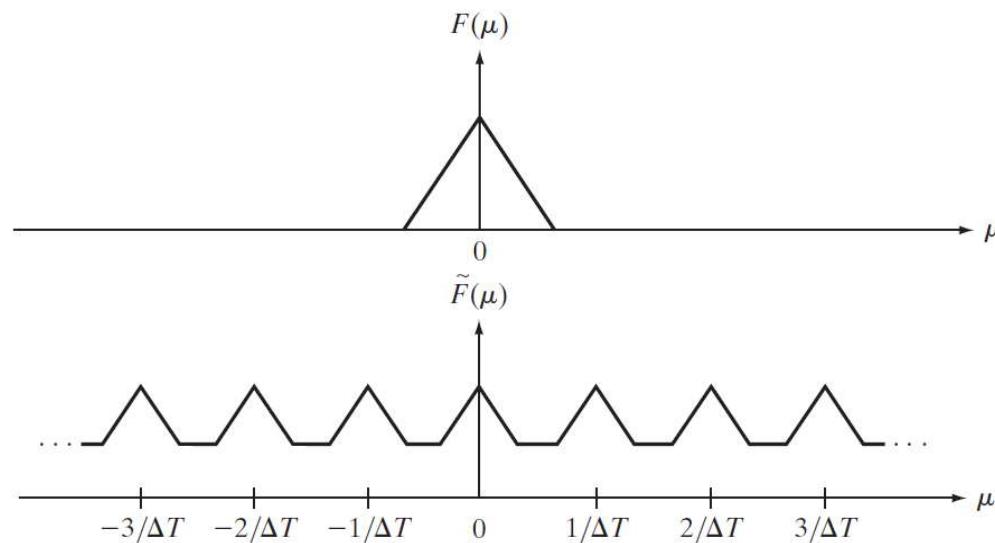
- The Sampling Theorem



- **Critically-sampling:** the sampling rate was just enough to preserve $F(\mu)$.

3. Sampling and Fourier Transform of Sampled Functions

- The Sampling Theorem

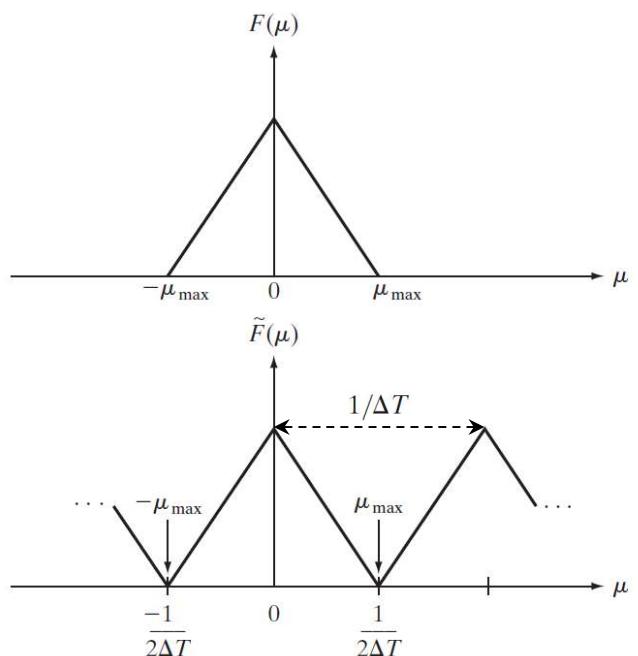


- **Under-sampling:** the sampling rate was below the minimum required to maintain distinct copies of $F(\mu)$ and thus failed to preserve the original transform.

3. Sampling and Fourier Transform of Sampled Functions

- The Sampling Theorem

- A function $f(t)$ whose Fourier transform is zero for values of frequencies outside a finite interval (band) about the origin is called a **band-limited** function.



a
b

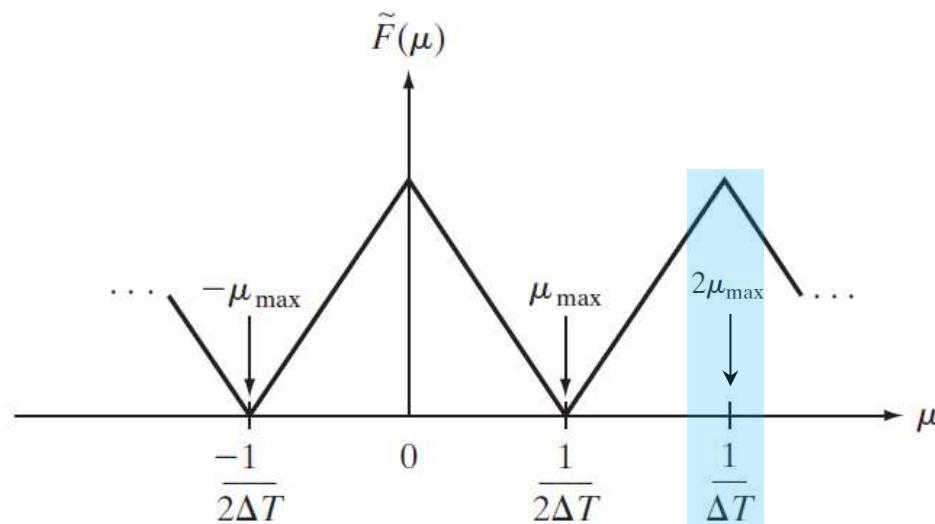
FIGURE 4.7
 (a) Transform of a band-limited function.
 (b) Transform resulting from critically sampling the same function.

3. Sampling and Fourier Transform of Sampled Functions

- The Sampling Theorem

- Extracting from $\tilde{F}(\mu)$ a **single period** that is equal to $F(\mu)$ is possible if the separation between copies is sufficient.
- Sufficient separation is guaranteed if

$$\frac{1}{\Delta T} > 2\mu_{\max}$$



3. Sampling and Fourier Transform of Sampled Functions

- The Sampling Theorem

- This equation indicates that a **continuous, band-limited** function can be recovered completely from a set of its samples if the samples are acquired at a rate **exceeding twice the highest frequency content of the function.**

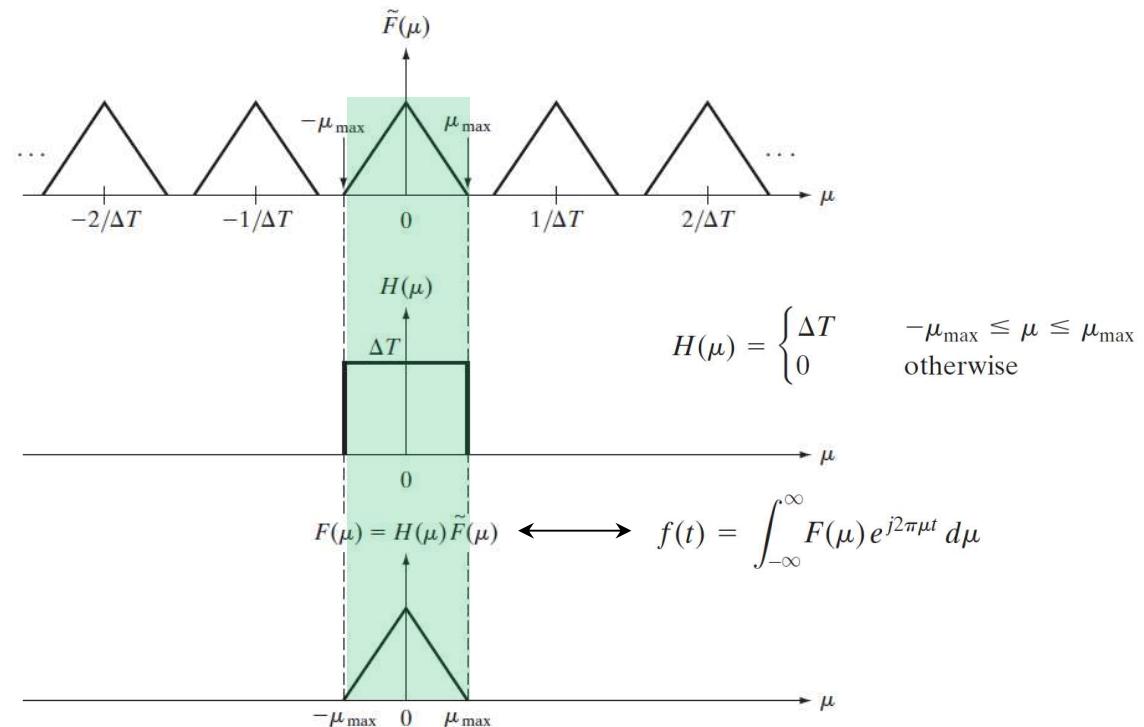
$$\mu_s > 2\mu_{\max}$$

- This result is known as the **sampling theorem**.
- A sampling rate equal to exactly twice the highest frequency is called the **Nyquist rate**.

3. Sampling and Fourier Transform of Sampled Functions

- The Sampling Theorem

- Extracting one period of the transform of a band-limited function using an **ideal lowpass filter**.

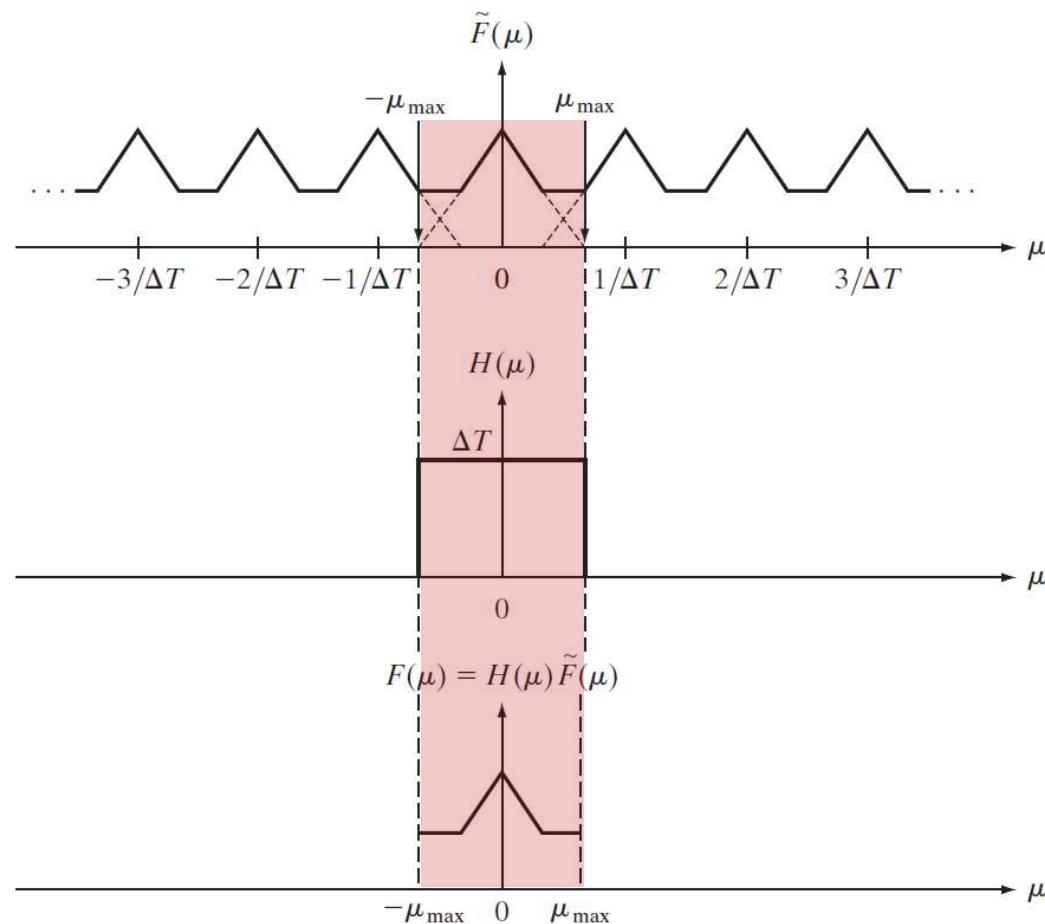


3. Sampling and Fourier Transform of Sampled Functions

- The Sampling Theorem
 - Function $H(\mu)$ is called a **lowpass filter** because it passes frequencies at the low end of the frequency range but it eliminates all higher frequencies.
 - It is called also an **ideal lowpass filter** because of its infinitely rapid transitions in amplitude.
 - We will have much more to say about filtering later.

3. Sampling and Fourier Transform of Sampled Functions

- Aliasing



3. Sampling and Fourier Transform of Sampled Functions

- Aliasing

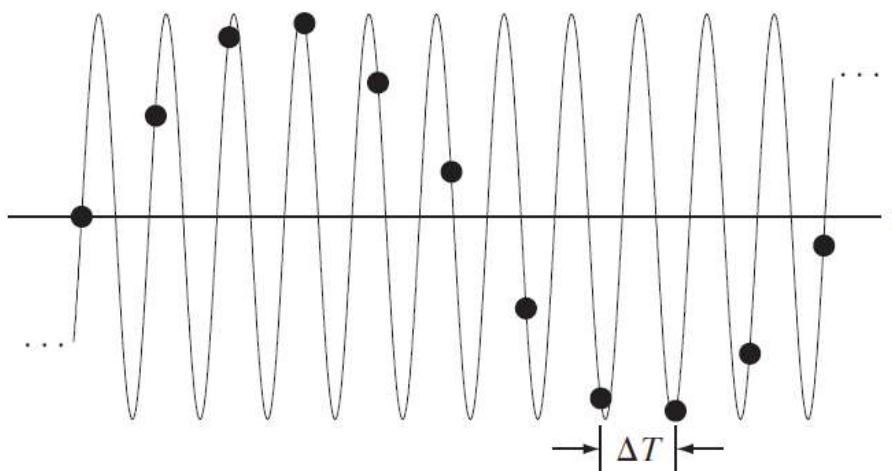


FIGURE 4.10 Illustration of aliasing. The under-sampled function (black dots) looks like a sine wave having a frequency much lower than the frequency of the continuous signal. The period of the sine wave is 2 s, so the zero crossings of the horizontal axis occur every second. ΔT is the separation between samples.

3. Sampling and Fourier Transform of Sampled Functions

- Function reconstruction
 - Using the convolution theorem

$$\begin{aligned} f(t) &= \mathcal{I}^{-1}\{F(\mu)\} \\ &= \mathcal{I}^{-1}\{H(\mu)\tilde{F}(\mu)\} \\ &= h(t) \star \tilde{f}(t) = \tilde{f}(t) \star h(t) \end{aligned}$$

3. Sampling and Fourier Transform of Sampled Functions

- Function reconstruction

➤ Using the convolution theorem

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t) \delta(t - n\Delta T)$$

$$h(t) = \mathfrak{I}^{-1}\{H(u)\} = \int_{-\mu_{\max}}^{\mu_{\max}} \Delta T e^{j2\pi u t} du = \frac{\Delta T}{\pi t} \sin(2\pi\mu_{\max} t)$$

$$= 2\Delta T \mu_{\max} \frac{\sin(\pi 2\mu_{\max} t)}{\pi 2\mu_{\max} t} = 2\Delta T \mu_{\max} \text{sinc}(2\mu_{\max} t)$$

$$= 2\Delta T \frac{1}{2\Delta T} \text{sinc}\left(\frac{t}{\Delta T}\right) = \text{sinc}\left(\frac{t}{\Delta T}\right)$$

$$\boxed{\int e^{cx} dx = \frac{1}{c} e^{cx}}$$

$$\boxed{\text{sinc}(m) = \frac{\sin(\pi m)}{(\pi m)}}$$

3. Sampling and Fourier Transform of Sampled Functions

- Function reconstruction

➤ Using the convolution theorem

$$f(t) = \tilde{f}(t) \star h(t) = \int_{-\infty}^{\infty} \tilde{f}(\tau) h(t - \tau) d\tau$$

$$f(t) = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(\tau) \delta(\tau - n\Delta T) h(t - \tau) d\tau$$

$$= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau) h(t - \tau) \delta(\tau - n\Delta T) d\tau$$

$$= \sum_{n=-\infty}^{\infty} f(n\Delta T) h(t - n\Delta T)$$

$$\boxed{h(t) = \text{sinc}\left(\frac{t}{\Delta T}\right) \rightarrow h(t - n\Delta T) = \text{sinc}\left(\frac{t - n\Delta T}{\Delta T}\right)}$$

$$= \sum_{n=-\infty}^{\infty} f(n\Delta T) \text{sinc}\left[\frac{(t - n\Delta T)}{\Delta T}\right]$$

4. The Discrete Fourier Transform (DFT) of One Variable

- Obtaining the DFT from the Continuous Transform of a Sampled Function.

- In practice, we work with a finite number of samples in time and in frequency domain.
- First we find $\tilde{F}(\mu)$ in terms of the sampled function $\tilde{f}(t)$ itself.

$$\begin{aligned}
 \tilde{F}(\mu) &= \int_{-\infty}^{\infty} \tilde{f}(t) e^{-j2\pi\mu t} dt \\
 &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(t) \delta(t - n\Delta T) e^{-j2\pi\mu t} dt \\
 &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} f(t) \delta(t - n\Delta T) e^{-j2\pi\mu t} dt \\
 &= \sum_{n=-\infty}^{\infty} f_n e^{-j2\pi\mu n\Delta T}
 \end{aligned}$$

$$\boxed{\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t) \delta(t - n\Delta T)}$$

4. The Discrete Fourier Transform (DFT) of One Variable

- Obtaining the DFT from the Continuous Transform of a Sampled Function
 - Although f_n is a discrete function, its Fourier $\tilde{F}(\mu)$ is continuous and infinitely periodic with period $1/\Delta T$ (see slide 48).
 - All we need to characterize $\tilde{F}(\mu)$ is one period, and sampling one period is the basis for the DFT.
 - Suppose that we want to obtain M equally spaced samples of $\tilde{F}(\mu)$ taken from $\mu = 0$ to $\mu = 1/\Delta T$.
 - This is accomplished by taking the samples at the following frequencies:

$$\mu = \frac{m}{M\Delta T} \quad m = 0, 1, 2, \dots, M-1$$

4. The Discrete Fourier Transform (DFT) of One Variable

- Obtaining the DFT from the Continuous Transform of a Sampled Function

➤ Substituting $\mu = \frac{m}{M\Delta T}$ $m = 0, 1, 2, \dots, M - 1$

into

$$\tilde{F}(\mu) = \sum_{n=-\infty}^{\infty} f_n e^{-j2\pi\mu n \Delta T}$$

and letting F_m denote the result, then

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi m n / M} \quad m = 0, 1, 2, \dots, M - 1$$

- This is the **discrete Fourier transform** we are seeking. A set $\{f_n\}$ consisting of M samples of $f(t)$ yields a set $\{F_m\}$ of M complex discrete values.

4. The Discrete Fourier Transform (DFT) of One Variable

- Obtaining the DFT from the Continuous Transform of a Sampled Function

- Conversely, given $\{F_m\}$ we can recover the sample set $\{f_n\}$ by using the IDFT.

$$f_n = \frac{1}{M} \sum_{m=0}^{M-1} F_m e^{j2\pi mn/M} \quad n = 0, 1, 2, \dots, M-1$$

- Demonstration

$$\begin{aligned}
 f_n &= \frac{1}{M} \sum_{m=0}^{M-1} \left\{ \sum_{k=0}^{M-1} f_k e^{-j2\pi mk/M} \right\} e^{j2\pi nm/M} \\
 &= \sum_{k=0}^{M-1} f_k \left\{ \frac{1}{M} \sum_{m=0}^{M-1} e^{j2\pi \frac{m(k-n)}{M}} \right\} = \\
 &= \sum_{k=0}^{M-1} f_k \delta(k-n) = f_n, \quad n = 0 \dots M-1
 \end{aligned}$$

$$\frac{1}{M} \sum_{m=0}^{M-1} e^{j2\pi \frac{m(k-n)}{M}} = \frac{1}{M} \sum_{m=0}^{M-1} e^{j2\pi \frac{m\alpha}{M}}, \quad \alpha = (k-n)$$

$$\frac{1}{M} \sum_{m=0}^{M-1} e^{j2\pi \frac{m\alpha}{M}} = 0, \text{ for } \alpha \neq 0$$

$$\frac{1}{M} \sum_{m=0}^{M-1} e^{j2\pi \frac{m\alpha}{M}} = 1, \text{ for } \alpha = 0$$

$$\frac{1}{M} \sum_{m=0}^{M-1} e^{j2\pi \frac{m(k-n)}{M}} = \delta(k-n)$$

4. The Discrete Fourier Transform (DFT) of One Variable

- Obtaining the DFT from the Continuous Transform of a Sampled Function

- It is more intuitive, especially in two dimensions, to use the notation x and y for image coordinate variables and u and v for frequency variables.

$$F(u) = \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \quad u = 0, 1, 2, \dots, M-1$$

$$f(x) = \frac{1}{M} \sum_{u=0}^{M-1} F(u) e^{j2\pi ux/M} \quad x = 0, 1, 2, \dots, M-1$$

- It can be shown that both the forward and inverse discrete transforms are **infinitely periodic**, with period M .

$$f(x) = f(x + kM) \quad F(u) = F(u + kM)$$

4. The Discrete Fourier Transform (DFT) of One Variable

- Obtaining the DFT from the Continuous Transform of a Sampled Function

- The discrete equivalent of the convolution

$$f(x) \star h(x) = \sum_{m=0}^{M-1} f(m)h(x-m) \quad x = 0, 1, 2, \dots, M-1$$

- Because in the preceding formulations the functions are periodic, their convolution is also periodic.
 - For this reason, the process inherent in this equation often is referred to as **circular convolution**, and is a direct result of the periodicity of the DFT and its inverse.
 - Finally, the convolution theorem is applicable also to discrete variables.

4. The Discrete Fourier Transform (DFT) of One Variable

- Relationship between the sampling and frequency intervals

- If $f(x)$ consists of M samples of a function $f(t)$ taken ΔT units apart, the duration of the record comprising the set

$$\{f(x)\}, x = 0, 1, 2, \dots, M - 1$$

is

$$T = M\Delta T$$

- The corresponding spacing Δu , in the discrete frequency domain follows from

$$\Delta u = \frac{1}{M\Delta T} = \frac{1}{T}$$

- The entire frequency range spanned by the M components of the DFT is

$$\Omega = M\Delta u = \frac{1}{\Delta T}$$

4. The Discrete Fourier Transform (DFT) of One Variable

- Relationship between the sampling and frequency intervals
 - Thus, we see that the resolution in frequency, Δu , of the DFT depends on the duration T over which the continuous function, $f(t)$, is sampled.

$$\Delta u = \frac{1}{M\Delta T} = \frac{1}{T}$$

- The range of frequencies spanned by the DFT depends on the sampling interval.

$$\Omega = M\Delta u = \frac{1}{\Delta T}$$

- Observe that both expressions exhibit inverse relationships with respect to T and ΔT .

5. Extension to Functions of Two Variables

- The 2-D Impulse and Its Sifting Property

➤ The impulse of two continuous variables is given by

$$\delta(t, z) = \begin{cases} \infty & \text{if } t = z = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t, z) dt dz = 1$$

➤ The 2-D impulse exhibits the sifting property under integration

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) \delta(t, z) dt dz = f(0, 0)$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) \delta(t - t_0, z - z_0) dt dz = f(t_0, z_0)$$

5. Extension to Functions of Two Variables

- The 2-D Impulse and Its Sifting Property

- For discrete variables

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y) \delta(x, y) = f(0, 0)$$

- For an impulse located at coordinates (x_0, y_0)

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y) \delta(x - x_0, y - y_0) = f(x_0, y_0)$$

5. Extension to Functions of Two Variables

- The 2-D Impulse and Its Sifting Property
 - For an impulse located at coordinates (x_0, y_0)

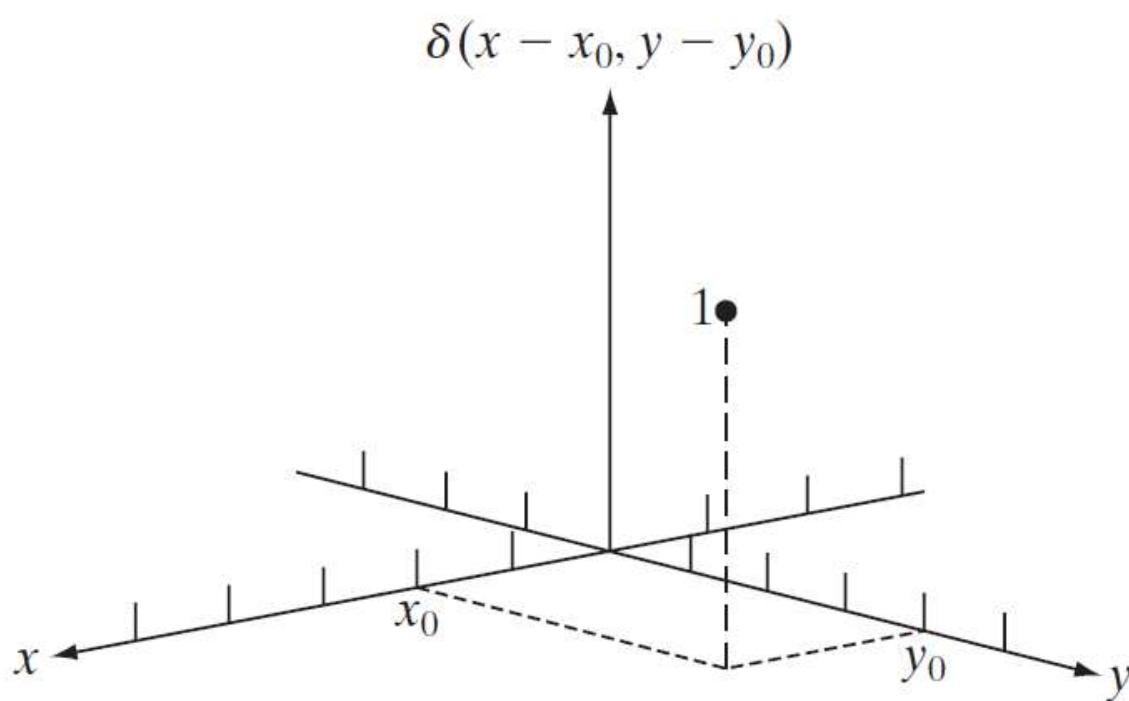


FIGURE 4.12
Two-dimensional unit discrete impulse. Variables x and y are discrete, and δ is zero everywhere except at coordinates (x_0, y_0) .

5. Extension to Functions of Two Variables

- The 2-D Continuous Fourier Transform Pair
 - Let $f(t, z)$ be a continuous function of two continuous variables, t and z .
 - The two-dimensional, continuous Fourier transform pair is given by the expressions

$$F(\mu, \nu) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) e^{-j2\pi(\mu t + \nu z)} dt dz$$

$$f(t, z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\mu, \nu) e^{j2\pi(\mu t + \nu z)} d\mu d\nu$$

5. Extension to Functions of Two Variables

- The 2-D Continuous Fourier Transform Pair

➤ Example

$$F(\mu, \nu) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) e^{-j2\pi(\mu t + \nu z)} dt dz$$

$$= \int_{-T/2}^{T/2} \int_{-Z/2}^{Z/2} A e^{-j2\pi(\mu t + \nu z)} dt dz$$

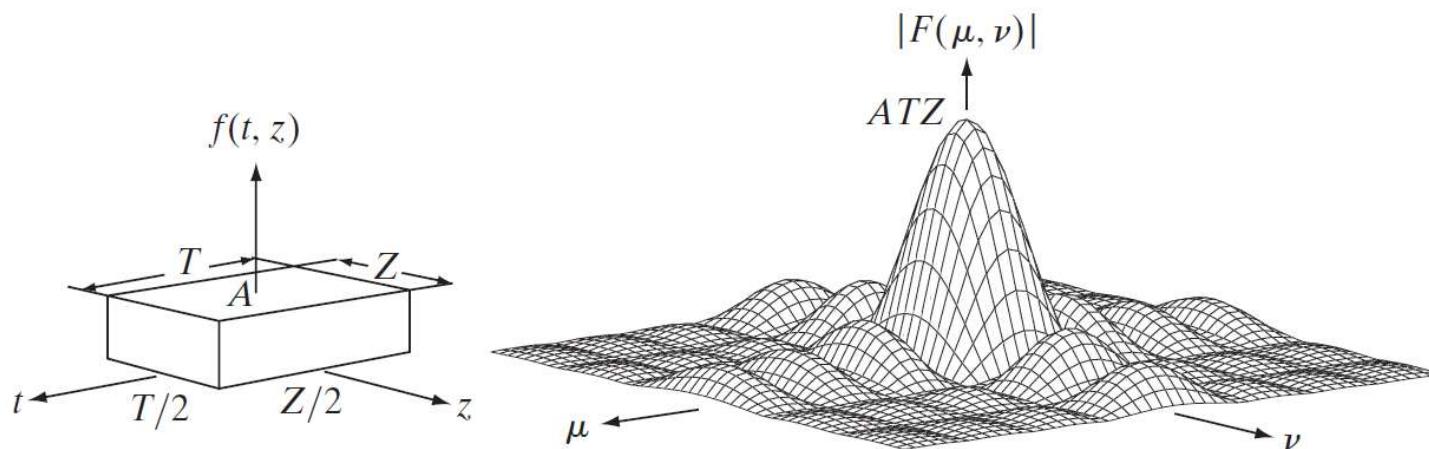
$$= ATZ \left[\frac{\sin(\pi\mu T)}{(\pi\mu T)} \right] \left[\frac{\sin(\pi\nu Z)}{(\pi\nu Z)} \right]$$

$$|F(\mu, \nu)| = ATZ \left| \frac{\sin(\pi\mu T)}{(\pi\mu T)} \right| \left| \frac{\sin(\pi\nu Z)}{(\pi\nu Z)} \right|$$

5. Extension to Functions of Two Variables

- The 2-D Continuous Fourier Transform Pair

➤ Example



a b

FIGURE 4.13 (a) A 2-D function, and (b) a section of its spectrum (not to scale). The block is longer along the t -axis, so the spectrum is more “contracted” along the μ -axis. Compare with Fig. 4.4.

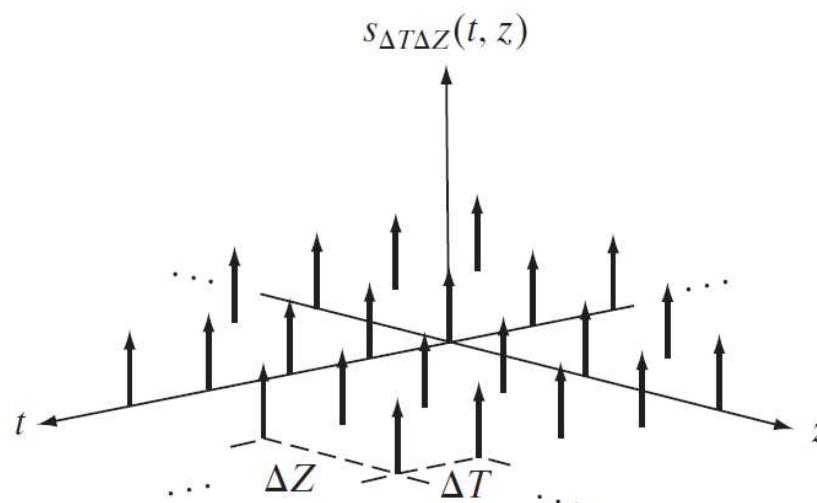
5. Extension to Functions of Two Variables

- Two-Dimensional Sampling and the 2-D Sampling Theorem

➤ 2-D impulse train

$$s_{\Delta T \Delta Z}(t, z) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - m\Delta T, z - n\Delta Z)$$

$$F(\mu, \nu) = 0 \quad \text{for } |\mu| \geq \mu_{\max} \text{ and } |\nu| \geq \nu_{\max}$$



5. Extension to Functions of Two Variables

- Two-Dimensional Sampling and the 2-D Sampling Theorem
 - The 2-D sampling theorem states that a continuous, band-limited function can be recovered with no error from a set of its samples if the sampling intervals are

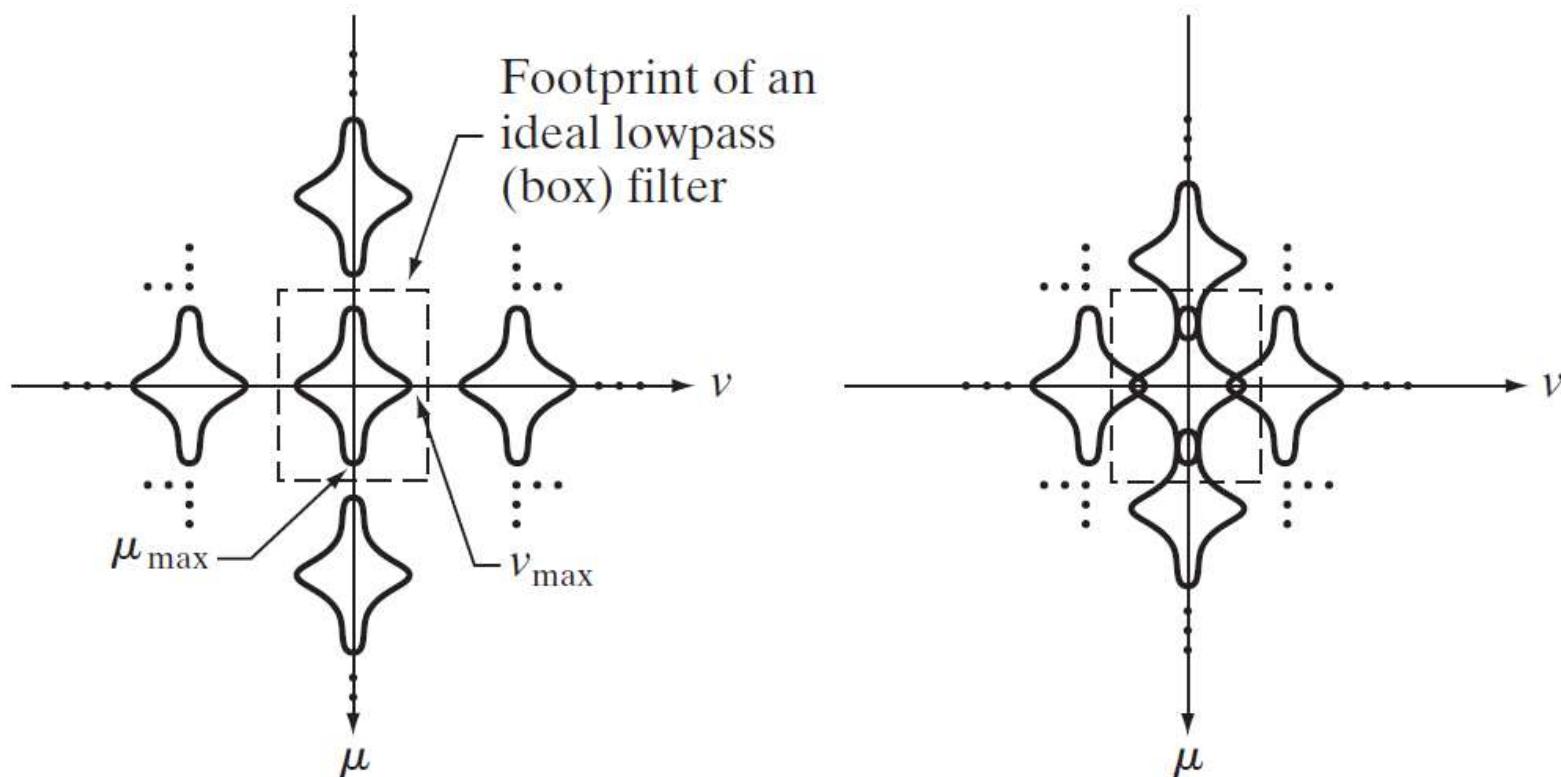
$$\frac{1}{\Delta T} > 2\mu_{\max} \quad \frac{1}{\Delta Z} > 2\nu_{\max}$$

- Stated another way, no information is lost if a 2-D, band-limited, continuous function is represented by samples acquired at rates greater than twice the highest frequency content of the function in both the μ - ν -directions.

$$\mu_s > 2\mu_{\max} \quad \nu_s > 2\nu_{\max}$$

5. Extension to Functions of Two Variables

- Aliasing in Images



5. Extension to Functions of Two Variables

- Aliasing in Images (sampling)
 - Suppose an imaging system whose number of samples it can take is fixed at 96×96 pixels.
 - If we use this system to digitize checkerboard patterns, it will be able to resolve patterns that are up to 96×96 pixels in which the size of each square is 1×1 pixels.
 - What happens when the detail (the size of the checkerboard squares) is less than one camera pixel?

5. Extension to Functions of Two Variables

- Aliasing in Images (sampling)

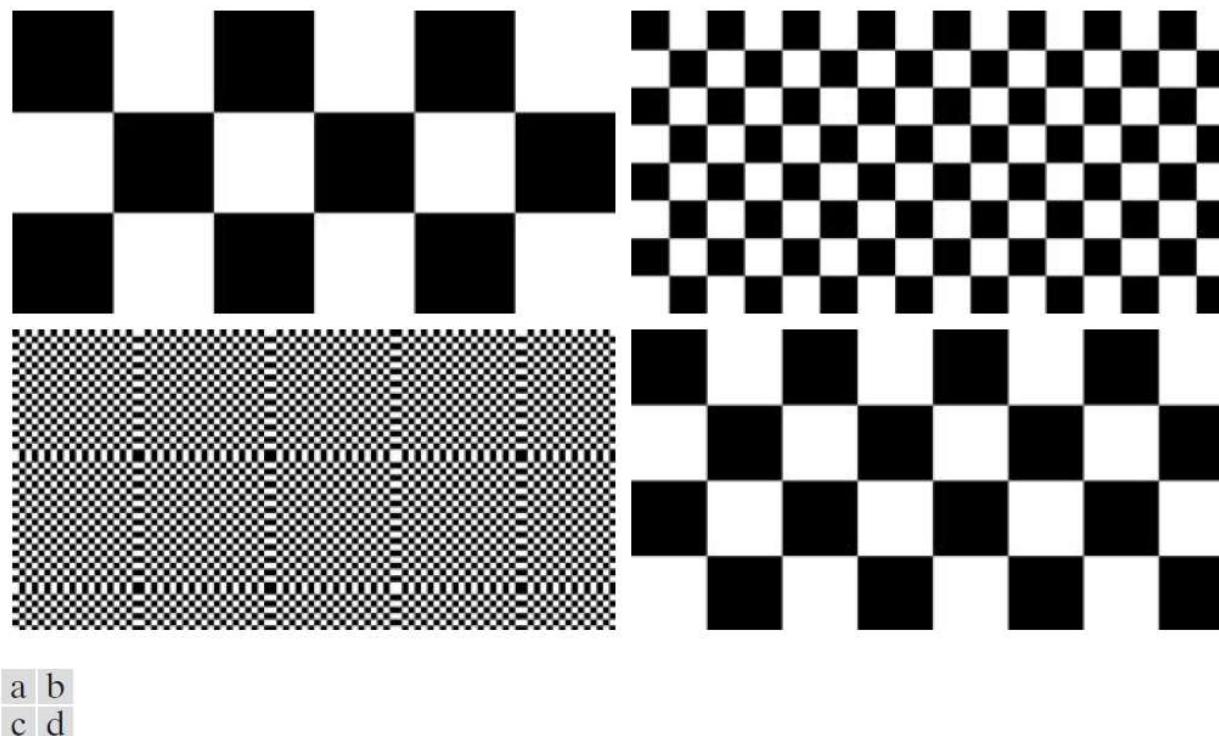


FIGURE 4.16 Aliasing in images. In (a) and (b), the lengths of the sides of the squares are 16 and 6 pixels, respectively, and aliasing is visually negligible. In (c) and (d), the sides of the squares are 0.9174 and 0.4798 pixels, respectively, and the results show significant aliasing. Note that (d) masquerades as a “normal” image.

5. Extension to Functions of Two Variables

- Aliasing in Images (interpolation and resampling)

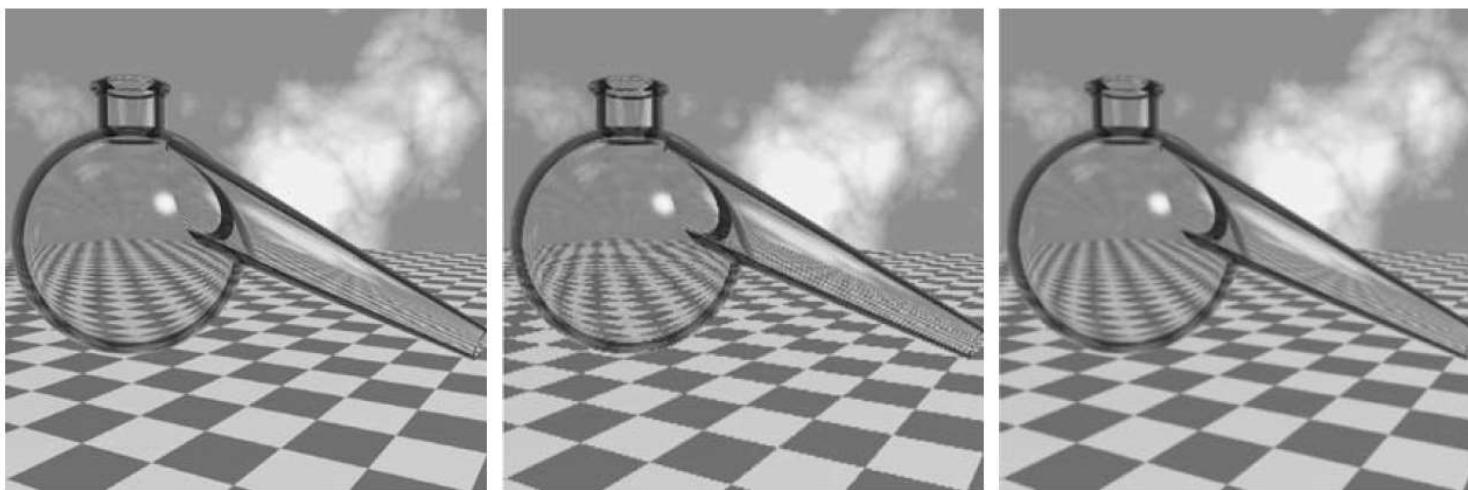


a b c

FIGURE 4.17 Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a 3×3 averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)

5. Extension to Functions of Two Variables

- Aliasing in Images (interpolation and resampling)
 - When you work with images that have strong edge content, the effects of aliasing are seen as block-like image components, called *jaggies*.

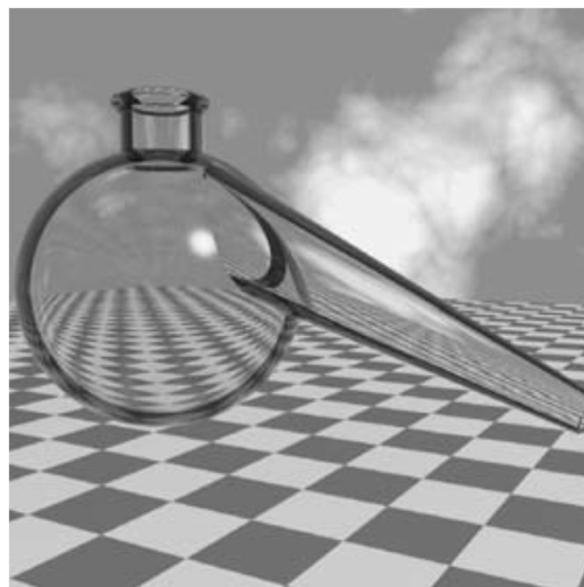


a b c

FIGURE 4.18 Illustration of jaggies. (a) A 1024×1024 digital image of a computer-generated scene with negligible visible aliasing. (b) Result of reducing (a) to 25% of its original size using bilinear interpolation. (c) Result of blurring the image in (a) with a 5×5 averaging filter prior to resizing it to 25% using bilinear interpolation. (Original image courtesy of D. P. Mitchell, Mental Landscape, LLC.)

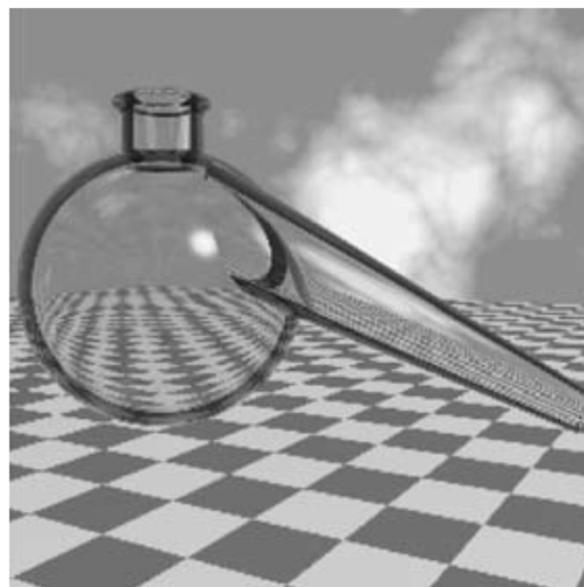
5. Extension to Functions of Two Variables

- Aliasing in Images (interpolation and resampling)
 - When you work with images that have strong edge content, the effects of aliasing are seen as block-like image components, called *jaggies*.



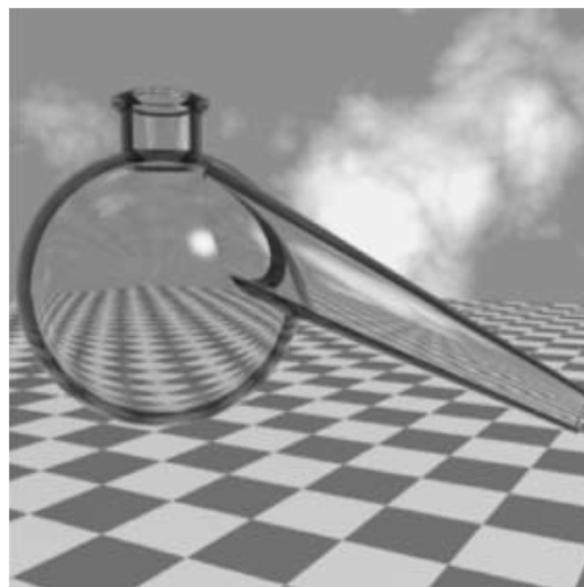
5. Extension to Functions of Two Variables

- Aliasing in Images (interpolation and resampling)
 - When you work with images that have strong edge content, the effects of aliasing are seen as block-like image components, called *jaggies*.



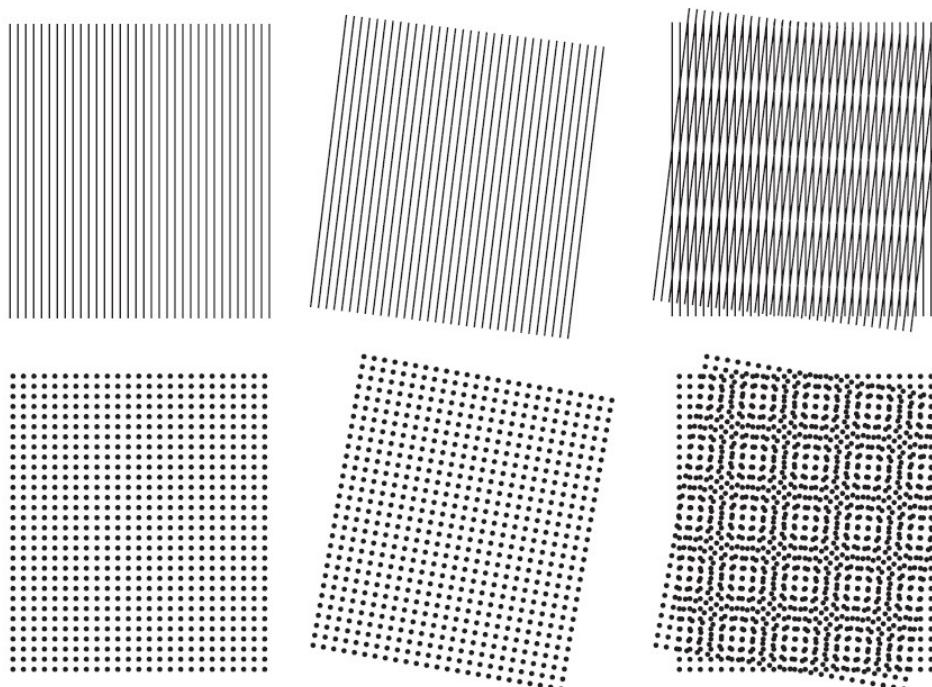
5. Extension to Functions of Two Variables

- Aliasing in Images (interpolation and resampling)
 - When you work with images that have strong edge content, the effects of aliasing are seen as block-like image components, called *jaggies*.



5. Extension to Functions of Two Variables

- Aliasing in Images (Moiré patterns)
 - This problem arises routinely when scanning media print, such as newspapers and magazines, or in images with periodic components.



a	b	c
d	e	f

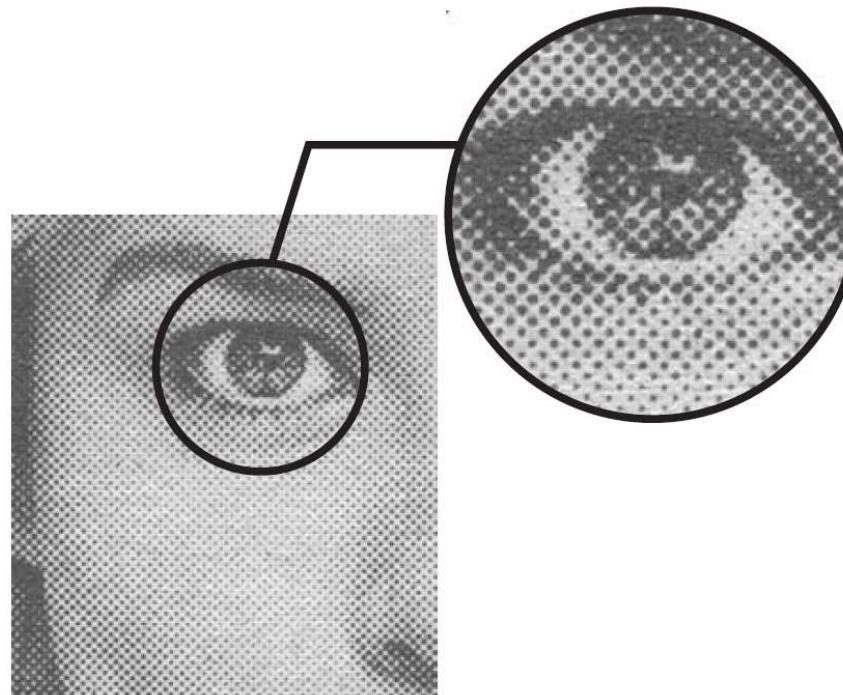
FIGURE 4.20

Examples of the moiré effect.
These are ink drawings, not digitized patterns.
Superimposing one pattern on the other is equivalent mathematically to multiplying the patterns.

5. Extension to Functions of Two Variables

- Aliasing in Images (Moiré patterns)
 - Printed materials make use of halftone dots, which are black dots whose sizes and joining schemes are used to simulate gray tones.

FIGURE 4.22
A newspaper image and an enlargement showing how halftone dots are arranged to render shades of gray.



5. Extension to Functions of Two Variables

- Aliasing in Images (Moiré patterns)
 - Printed materials make use of halftone dots, which are black dots whose sizes and joining schemes are used to simulate gray tones.



FIGURE 4.21
A newspaper image of size 246×168 pixels sampled at 75 dpi showing a moiré pattern. The moiré pattern in this image is the interference pattern created between the $\pm 45^\circ$ orientation of the halftone dots and the north-south orientation of the sampling grid used to digitize the image.

5. Extension to Functions of Two Variables

- The 2-D Discrete Fourier Transform and Its Inverse

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

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

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

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

6. Some Properties of the 2-D DFT

- Relationships Between Spatial and Frequency Intervals
 - Let ΔT and ΔZ denote the separations between samples.
 - Then, the separations between the corresponding discrete, frequency domain variables are given by

$$\Delta u = \frac{1}{M\Delta T} \quad \Delta v = \frac{1}{N\Delta Z}$$

6. Some Properties of the 2-D DFT

- Translation

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

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi(x_0u/M + y_0v/N)}$$

- Rotation

$$x = r \cos \theta \quad y = r \sin \theta \quad u = \omega \cos \varphi \quad v = \omega \sin \varphi$$

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$$

6. Some Properties of the 2-D DFT

- Periodicity

- The 2-D Fourier transform and its inverse are infinitely periodic in the u and v directions; that is,

$$F(u, v) = F(u + k_1 M, v) = F(u, v + k_2 N) = F(u + k_1 M, v + k_2 N)$$

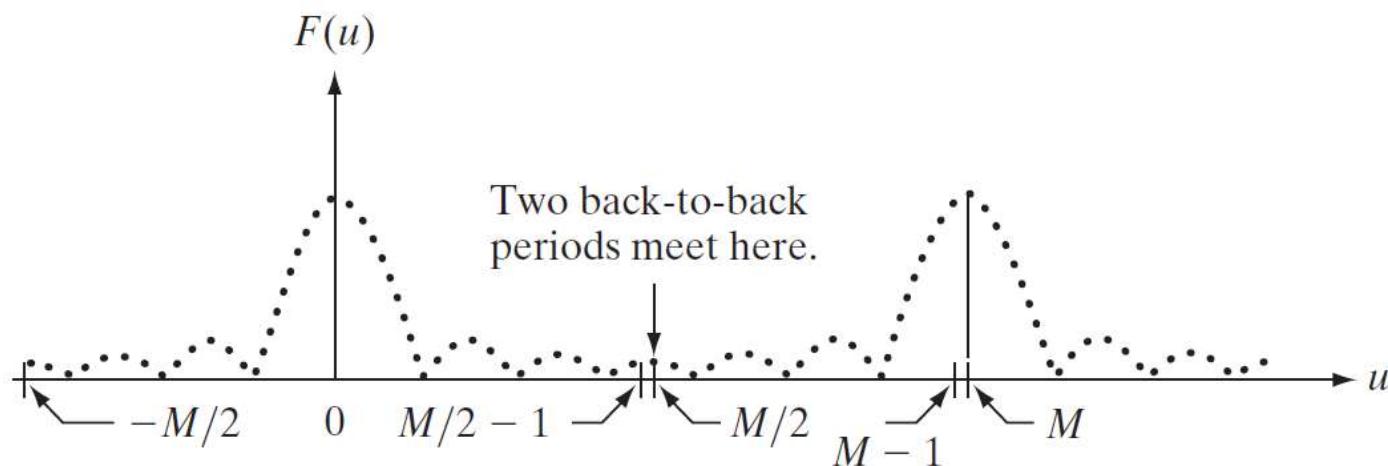
$$f(x, y) = f(x + k_1 M, y) = f(x, y + k_2 N) = f(x + k_1 M, y + k_2 N)$$

where M and N are the number of samples in u and v directions.

6. Some Properties of the 2-D DFT

- Periodicity

- Consider the 1-D spectrum. The transform data in the interval from 0 to $M-1$ consists of two back-to-back half periods meeting at point $M/2$.



6. Some Properties of the 2-D DFT

- Periodicity

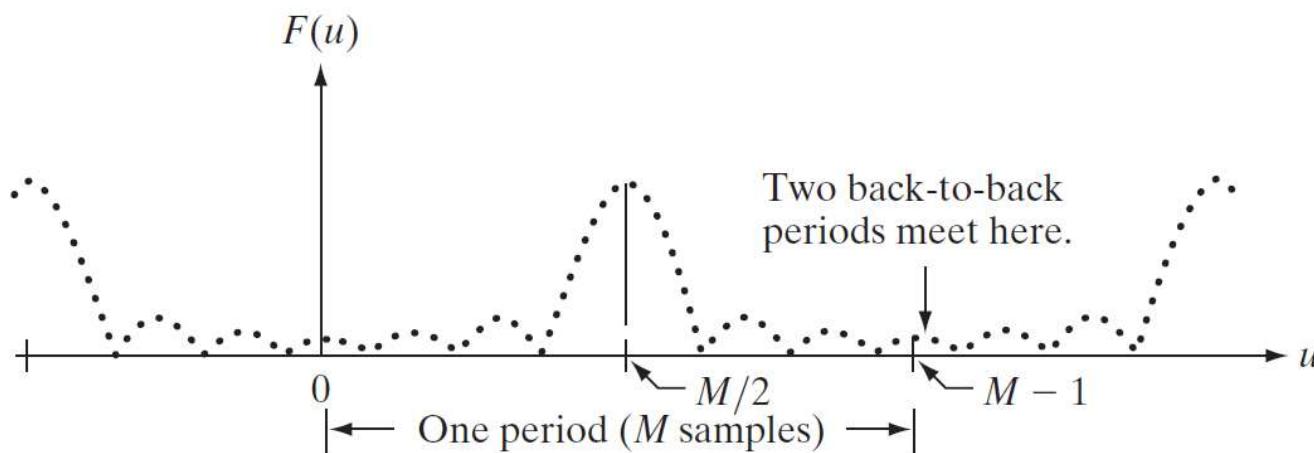
- For display and filtering purposes, it is more convenient to have in this interval a complete period of the transform in which the data are contiguous.
- We know that $f(x)e^{j2\pi(u_0x/M)} \Leftrightarrow F(u - u_0)$
- In other words, multiplying $f(x)$ by the exponential term shifts the data so that the origin, $F(0)$, is located at u_0 .
- If we let $u_0 = M/2$, $f(x)(-1)^x \Leftrightarrow F(u - M/2)$
- That is, shifts the data so that $F(0)$ is at the center of the interval $[0, M-1]$.

6. Some Properties of the 2-D DFT

- Periodicity

- For display and filtering purposes, it is more convenient to have in this interval a complete period of the transform in which the data are contiguous.

$$f(x)(-1)^x \Leftrightarrow F(u - M/2)$$

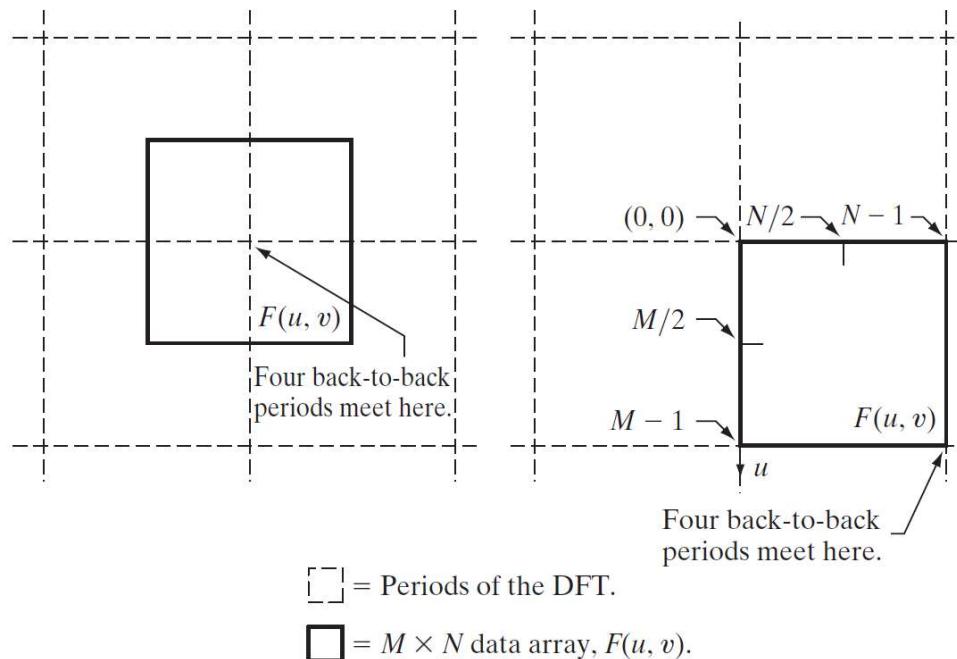


6. Some Properties of the 2-D DFT

- Periodicity

- In 2-D the situation is more difficult to graph, but the principle is the same.

$$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$$



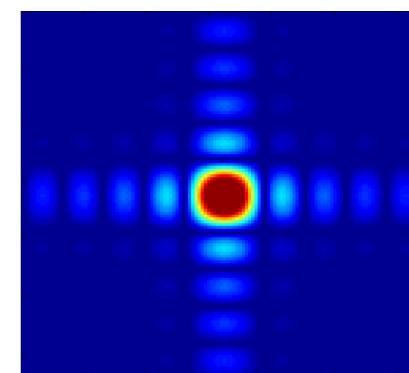
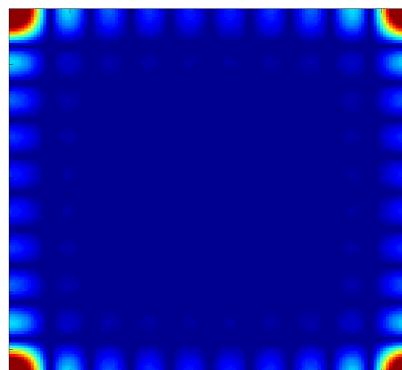
6. Some Properties of the 2-D DFT

- Periodicity

- In 2-D the situation is more difficult to graph, but the principle is the same.

$$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$$

- Using this equation shifts the data so that $F(0,0)$ is at the center of the frequency rectangle defined by the intervals and $[0 M-1]$, $[0 N-1]$.



6. Some Properties of the 2-D DFT

- Symmetry

- An important result from functional analysis is that any **real or complex function**, $w(x, y)$, can be expressed as the sum of an **even** and an **odd** part (each of which can be real or complex):

$$w(x, y) = w_e(x, y) + w_o(x, y)$$

- Where,

$$w_e(x, y) \triangleq \frac{w(x, y) + w(-x, -y)}{2}$$

$$w_o(x, y) \triangleq \frac{w(x, y) - w(-x, -y)}{2}$$

- It follows that

$$w_e(x, y) = w_e(-x, -y) \quad w_o(x, y) = -w_o(-x, -y)$$

6. Some Properties of the 2-D DFT

- Symmetry

- **Even** functions are said to be **symmetric** and **odd** functions are **antisymmetric**.
- When we talk about symmetry we are referring to symmetry about the center point of a sequence
- It is more convenient to think only in terms of **nonnegative indices**, in which case the definitions of evenness and oddness become

$$w_e(x, y) = w_e(M - x, N - y)$$

$$w_o(x, y) = -w_o(M - x, N - y)$$

where M and N are the number of rows and columns of a 2-D array.

6. Some Properties of the 2-D DFT

- Symmetry
 - Although **evenness** and **oddness** are visualized easily for continuous functions, **these concepts are not as intuitive** when dealing with **discrete** sequences.
 - ✓ Even sequences
 - $f(x) = f(M - x)$
 - $f(0)$ is immaterial in the test for evenness.

6. Some Properties of the 2-D DFT

- Symmetry

- Although **evenness** and **oddness** are visualized easily for continuous functions, **these concepts are not as intuitive** when dealing with **discrete** sequences.
 - ✓ Even sequences
 - Example

$$f = \{f(0) \quad f(1) \quad f(2) \quad f(3)\} = \{2 \quad 1 \quad 1 \quad 1\}$$

$$M = 4 \quad f(x) = f(4 - x)$$

$$f(0) = f(4), \quad f(2) = f(2),$$

$$f(1) = f(3), \quad f(3) = f(1)$$

6. Some Properties of the 2-D DFT

- Symmetry
 - Although **evenness** and **oddness** are visualized easily for continuous functions, **these concepts are not as intuitive** when dealing with **discrete** sequences.
 - ✓ Odd sequences
 - $g(x) = -g(M - x)$
 - When **M** is **even**, a 1-D **odd** sequence has the property that the points at position **0** and **M/2** are **always zero**.
 - When **M** is **odd**, the first term still has to be **0**, but the **remaining** terms form pairs with **equal value** but **opposite sign**.

6. Some Properties of the 2-D DFT

- Symmetry

- Although **evenness** and **oddness** are visualized easily for continuous functions, **these concepts are not as intuitive** when dealing with **discrete** sequences.
 - ✓ Odd sequences
 - Example

$$g = \{g(0) \ g(1) \ g(2) \ g(3)\} = \{0 \ -1 \ 0 \ 1\}$$

- However, the sequence below is neither odd nor even, although the “basic” structure appears to be odd.

$$g = \{0 \ -1 \ 0 \ 1 \ 0\}$$

6. Some Properties of the 2-D DFT

- Symmetry

- Although **evenness** and **oddness** are visualized easily for continuous functions, **these concepts are not as intuitive** when dealing with **discrete** sequences.
 - ✓ The same basic considerations hold in 2-D.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -2 & 0 & 2 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

- ✓ As an **exercise**, you use the previous equations to convince yourself that this 2-D sequence is **odd**.

6. Some Properties of the 2-D DFT

- Symmetry properties

- Now, a number of important symmetry properties of the DFT and its inverse can be establish.
- A property used frequently is that the Fourier transform of a **real** function, $f(x, y)$, is **conjugate symmetric**,

$$F^*(u, v) = F(-u, -v)$$

- If $f(x, y)$ is **imaginary**, its Fourier transform is **conjugate antisymmetric**,

$$F^*(-u, -v) = -F(u, v)$$

6. Some Properties of the 2-D DFT

- Symmetry properties

- Now, a number of important symmetry properties of the DFT and its inverse can be establish.

	Spatial Domain [†]	Frequency Domain [†]
1)	$f(x, y)$ real	$\Leftrightarrow F^*(u, v) = F(-u, -v)$
2)	$f(x, y)$ imaginary	$\Leftrightarrow F^*(-u, -v) = -F(u, v)$
3)	$f(x, y)$ real	$\Leftrightarrow R(u, v)$ even; $I(u, v)$ odd
4)	$f(x, y)$ imaginary	$\Leftrightarrow R(u, v)$ odd; $I(u, v)$ even
5)	$f(-x, -y)$ real	$\Leftrightarrow F^*(u, v)$ complex
6)	$f(-x, -y)$ complex	$\Leftrightarrow F(-u, -v)$ complex
7)	$f^*(x, y)$ complex	$\Leftrightarrow F^*(-u - v)$ complex
8)	$f(x, y)$ real and even	$\Leftrightarrow F(u, v)$ real and even
9)	$f(x, y)$ real and odd	$\Leftrightarrow F(u, v)$ imaginary and odd
10)	$f(x, y)$ imaginary and even	$\Leftrightarrow F(u, v)$ imaginary and even
11)	$f(x, y)$ imaginary and odd	$\Leftrightarrow F(u, v)$ real and odd
12)	$f(x, y)$ complex and even	$\Leftrightarrow F(u, v)$ complex and even
13)	$f(x, y)$ complex and odd	$\Leftrightarrow F(u, v)$ complex and odd

TABLE 4.1 Some symmetry properties of the 2-D DFT and its inverse. $R(u, v)$ and $I(u, v)$ are the real and imaginary parts of $F(u, v)$, respectively. The term *complex* indicates that a function has nonzero real and imaginary parts.

[†]Recall that x, y, u , and v are *discrete* (integer) variables, with x and u in the range $[0, M - 1]$, and y , and v in the range $[0, N - 1]$. To say that a complex function is *even* means that its real *and* imaginary parts are even, and similarly for an odd complex function.

6. Some Properties of the 2-D DFT

- Fourier Spectrum and Phase Angle.
 - Because the 2-D DFT is complex in general, it can be expressed in polar form:

$$F(u, v) = |F(u, v)| e^{j\phi(u, v)}$$

where

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

is the **frequency spectrum** and

$$\phi(u, v) = \arctan \left[\frac{I(u, v)}{R(u, v)} \right]$$

is the **phase angle**.

6. Some Properties of the 2-D DFT

- Fourier Spectrum and Phase Angle.

- The **power spectrum** is defined as

$$\begin{aligned} P(u, v) &= |F(u, v)|^2 \\ &= R^2(u, v) + I^2(u, v) \end{aligned}$$

- The Fourier transform of a **real** function is **conjugate symmetric**, which implies that the spectrum has **even symmetry** about the origin:

$$|F(u, v)| = |F(-u, -v)|$$

- The phase angle exhibits the following **odd symmetry** about the origin:

$$\phi(u, v) = -\phi(-u, -v)$$

6. Some Properties of the 2-D DFT

- Fourier Spectrum and Phase Angle.

➤ Finally, it follows from

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

that

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

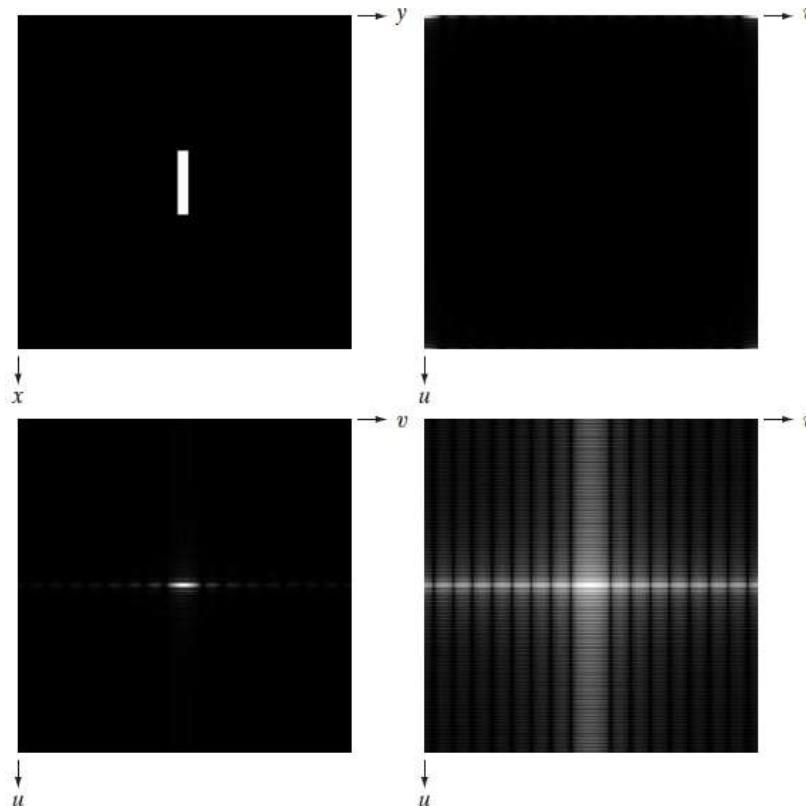
which indicates that the **zero-frequency term** is **proportional** to the **average value** of $f(x, y)$.

$$\begin{aligned} F(0, 0) &= MN \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \\ &= M\bar{f}(x, y) \end{aligned}$$

6. Some Properties of the 2-D DFT

- Fourier Spectrum and Phase Angle.

➤ Example (spectrum)



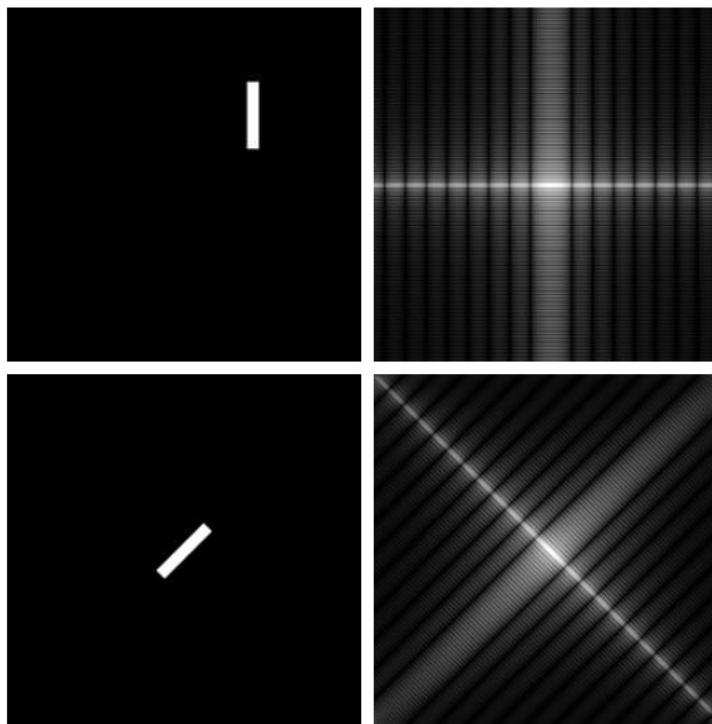
a
b
c
d

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.

6. Some Properties of the 2-D DFT

- Fourier Spectrum and Phase Angle.

- Example (spectrum)

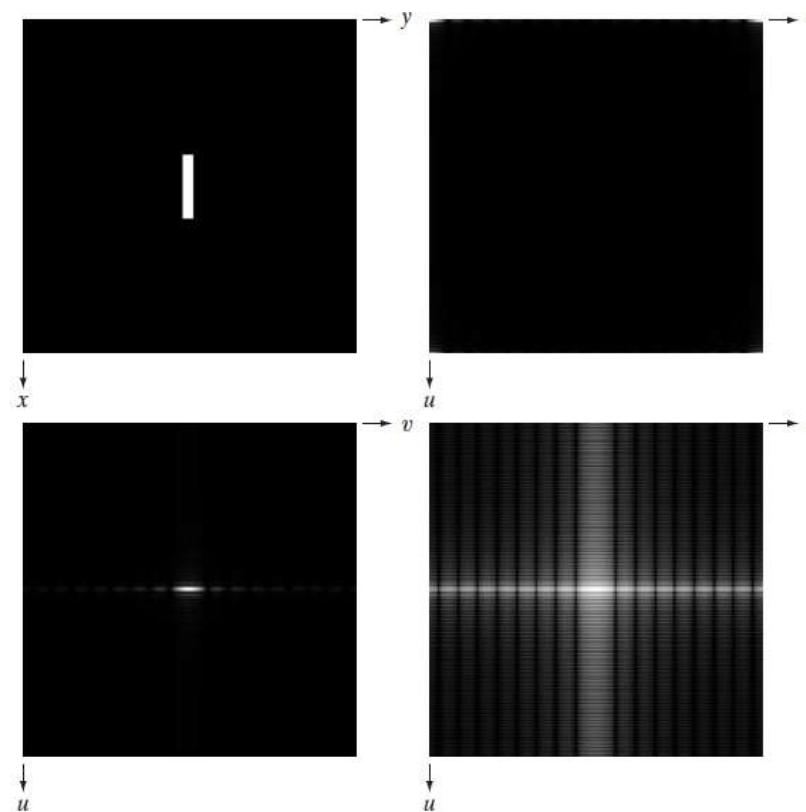


a	b
c	d

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).

6. Some Properties of the 2-D DFT

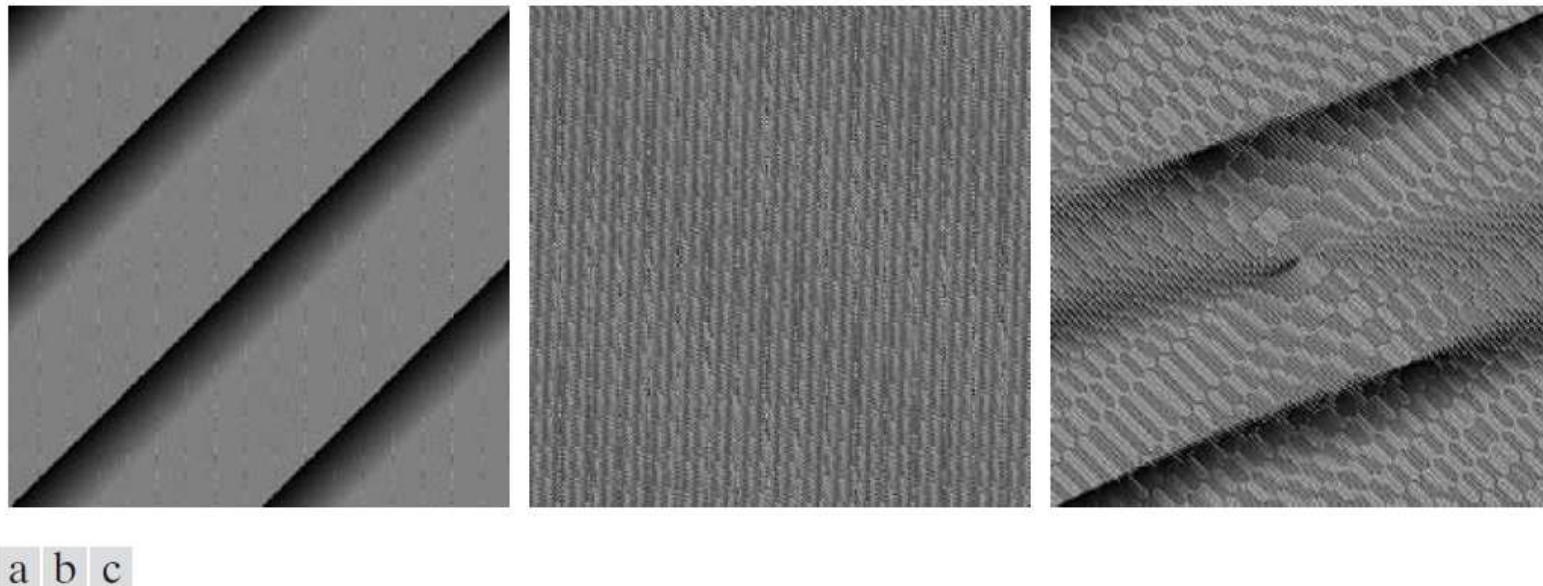
- MATLAB: s113Spectrum.m



6. Some Properties of the 2-D DFT

- Fourier Spectrum and Phase Angle.

- 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).

6. Some Properties of the 2-D DFT

- The 2-D Convolution Theorem

$$f(x, y) \star h(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n)$$
$$x = 0, 1, 2, \dots, M - 1 \quad \text{and} \quad y = 0, 1, 2, \dots, N - 1$$

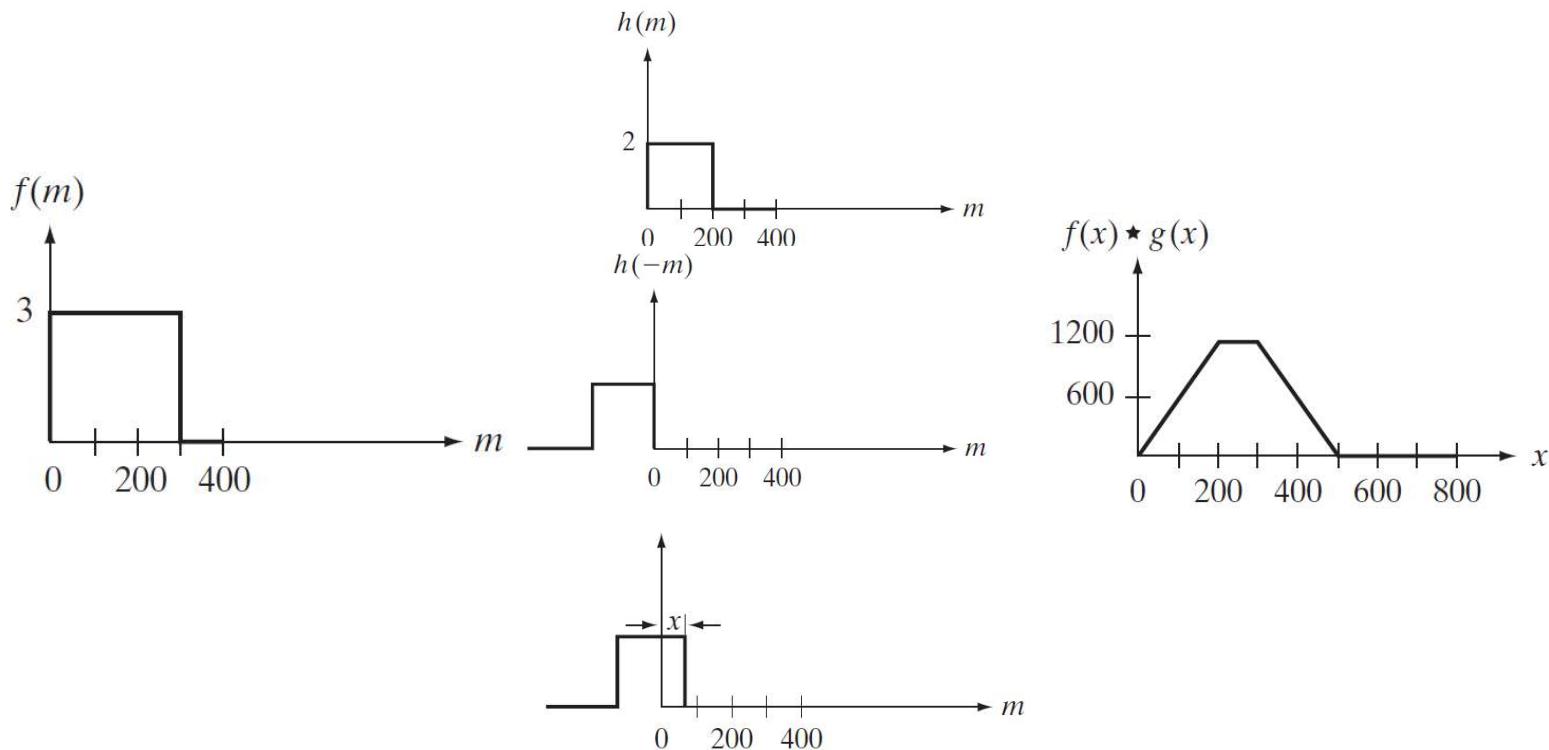
$$f(x, y) \star h(x, y) \Leftrightarrow F(u, v)H(u, v)$$

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

6. Some Properties of the 2-D DFT

- The 2-D Convolution Theorem

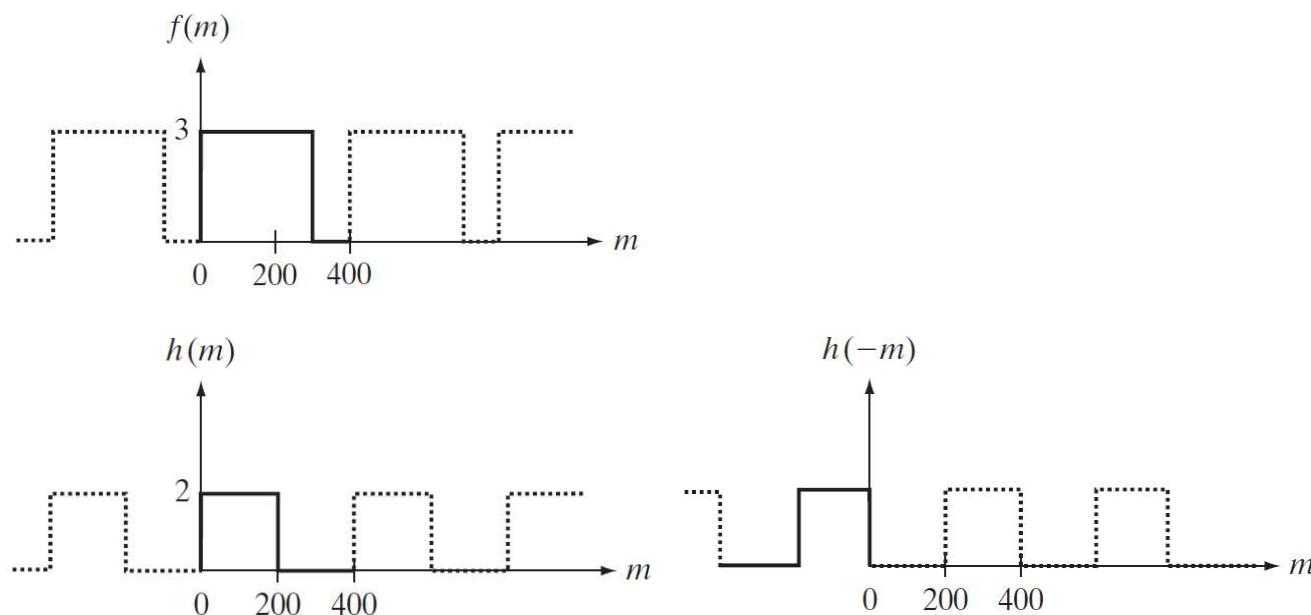
➤ 1-D example: convolution of two functions, f and h .



6. Some Properties of the 2-D DFT

- The 2-D Convolution Theorem

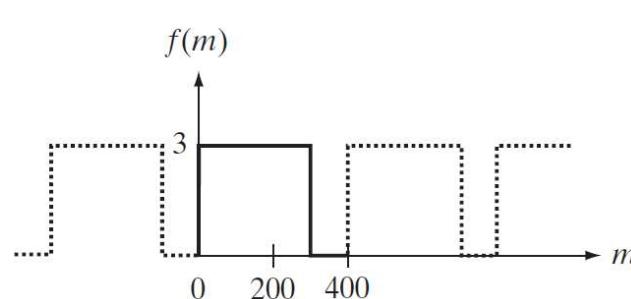
➤ 1-D example: if we use the DFT and the convolution theorem to obtain the same result, we must take into account the **periodicity inherent in the expression for the DFT**.



6. Some Properties of the 2-D DFT

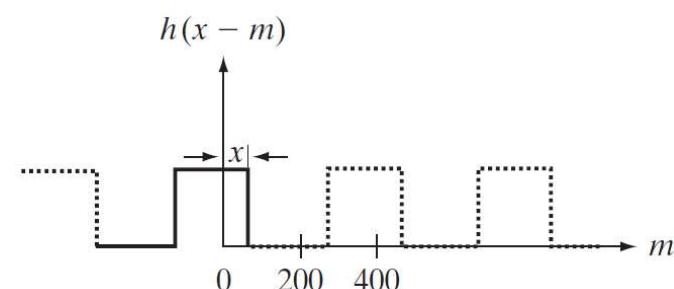
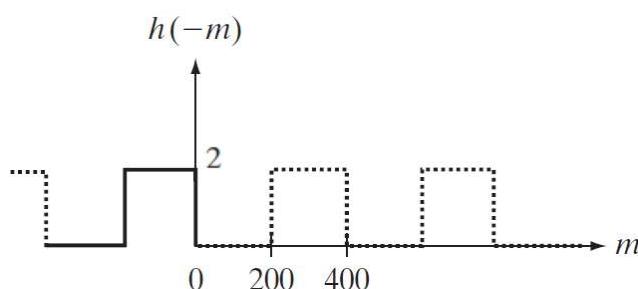
- The 2-D Convolution Theorem

➤ 1-D example: if we use the DFT and the convolution theorem to obtain the same result, we must take into account the periodicity inherent in the expression for the DFT.



$$f(x) \star h(x) = \sum_{m=0}^{M-1} f(m)h(x-m)$$

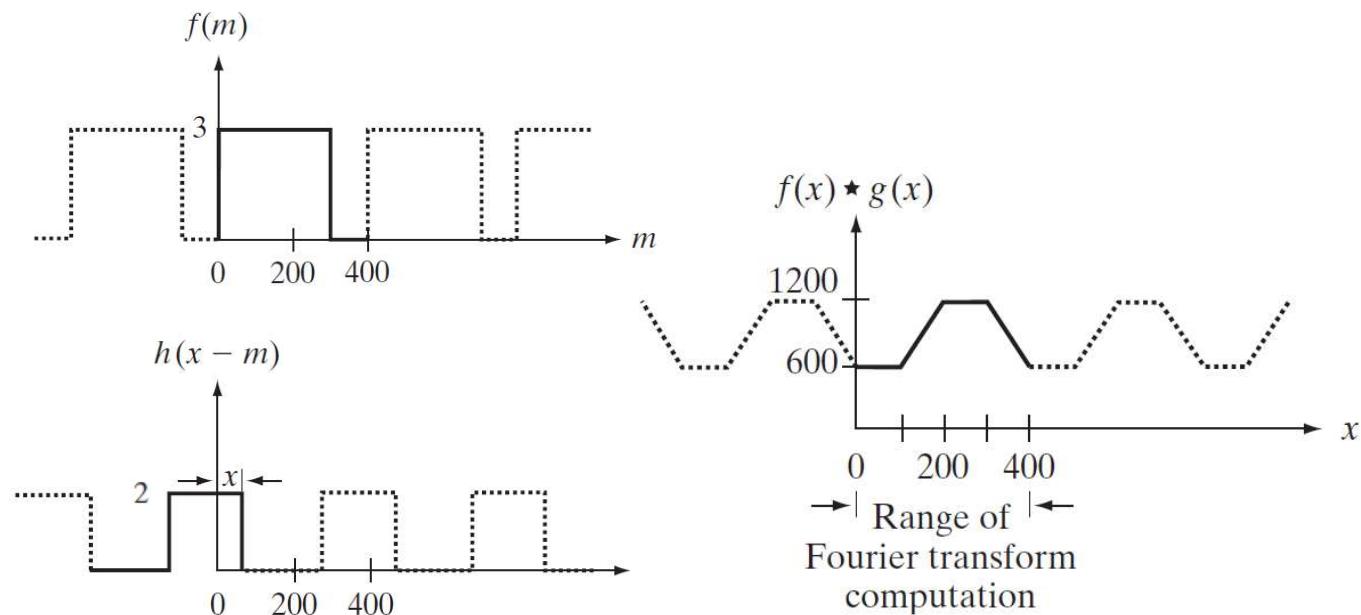
$$x = 0, 1, 2, \dots, M-1$$



6. Some Properties of the 2-D DFT

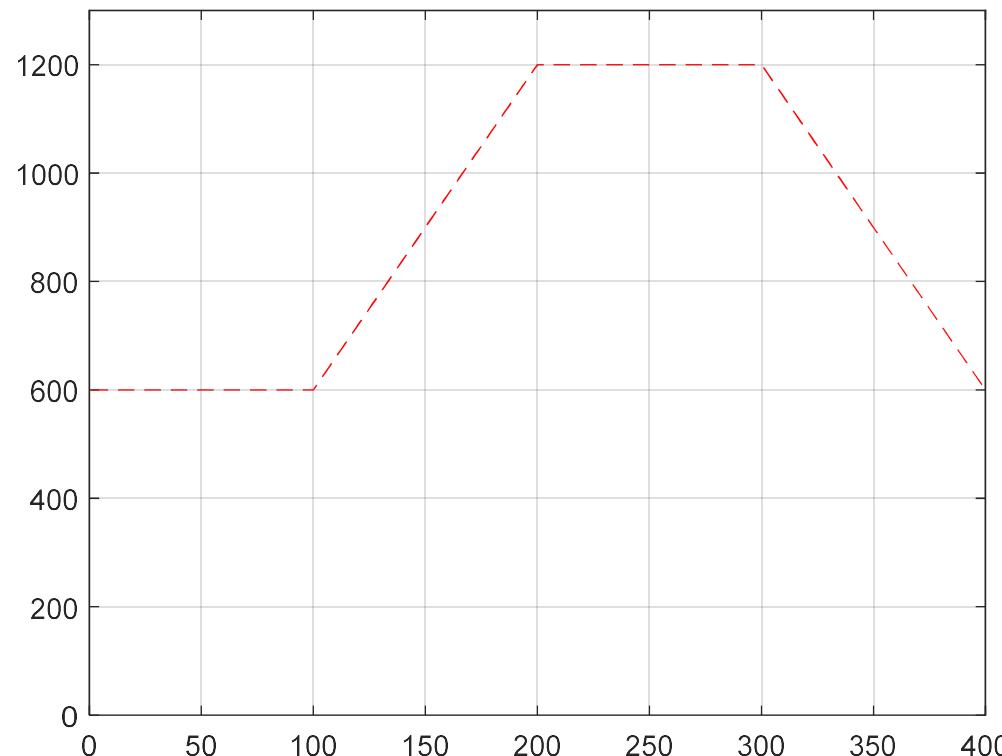
- The 2-D Convolution Theorem

➤ 1-D example: if we use the DFT and the convolution theorem to obtain the same result, we must take into account the periodicity inherent in the expression for the DFT.



6. Some Properties of the 2-D DFT

- MATLAB: s120ConvolutionPadding.m



6. Some Properties of the 2-D DFT

- The 2-D Convolution Theorem
 - The **closeness of the periods** is such that they interfere with each other to cause what is commonly referred to as **wraparound error**.
 - Fortunately, the solution to the wraparound error problem is simple.
 - Consider two functions, $f(x)$ and $h(x)$ composed of **A and B samples**, respectively.
 - We need to append P zeros to both functions.

$$P \geq A + B - 1$$

6. Some Properties of the 2-D DFT

- The 2-D Convolution Theorem
 - Visualizing a similar example in 2-D would be more difficult, but the conclusions are similar.
 - Let $f(x, y)$ and $h(x, y)$ be two image arrays of sizes $A \times B$ and $C \times D$ pixels, respectively.
 - Wraparound error in their circular convolution can be avoided by **padding** these functions with zeros, as follows

$$f_p(x, y) = \begin{cases} f(x, y) & 0 \leq x \leq A - 1 \text{ and } 0 \leq y \leq B - 1 \\ 0 & A \leq x \leq P \text{ or } B \leq y \leq Q \end{cases}$$

$$h_p(x, y) = \begin{cases} h(x, y) & 0 \leq x \leq C - 1 \text{ and } 0 \leq y \leq D - 1 \\ 0 & C \leq x \leq P \text{ or } D \leq y \leq Q \end{cases}$$

$$P \geq A + C - 1 \quad Q \geq B + D - 1$$

7. The Basics of Filtering in the Frequency Domain

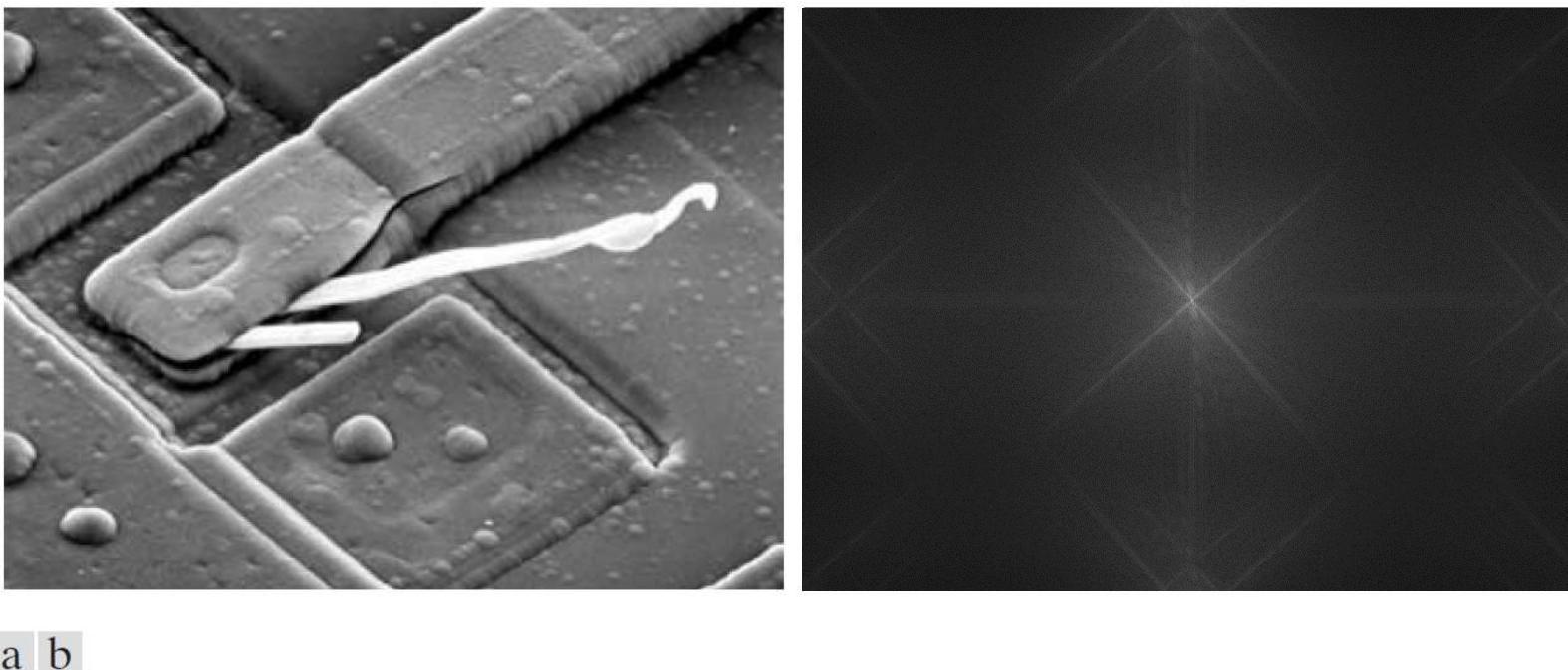
- Additional Characteristics of the Frequency Domain
 - Filtering techniques in the frequency domain are based on modifying the DFT of a signal to achieve a specific objective.
 - **Frequency** is directly related to **spatial rates of change**.
 - ✓ The **slowest** varying frequency component is proportional to the **average intensity** of an image.
 - ✓ As we move **away from the origin** of the shifted transform, the high frequencies correspond to the **fast varying intensity** components of an image.

7. The Basics of Filtering in the Frequency Domain

- Additional Characteristics of the Frequency Domain
 - The two components of the transform to which we have access are:
 - ✓ The transform **magnitude** (spectrum); and
 - ✓ The phase **angle**.
 - Visual analysis of the **phase** component generally is **not very useful**.
 - The **spectrum**, however, provides some **useful visual information**.

7. The Basics of Filtering in the Frequency Domain

- Additional Characteristics of the Frequency Domain

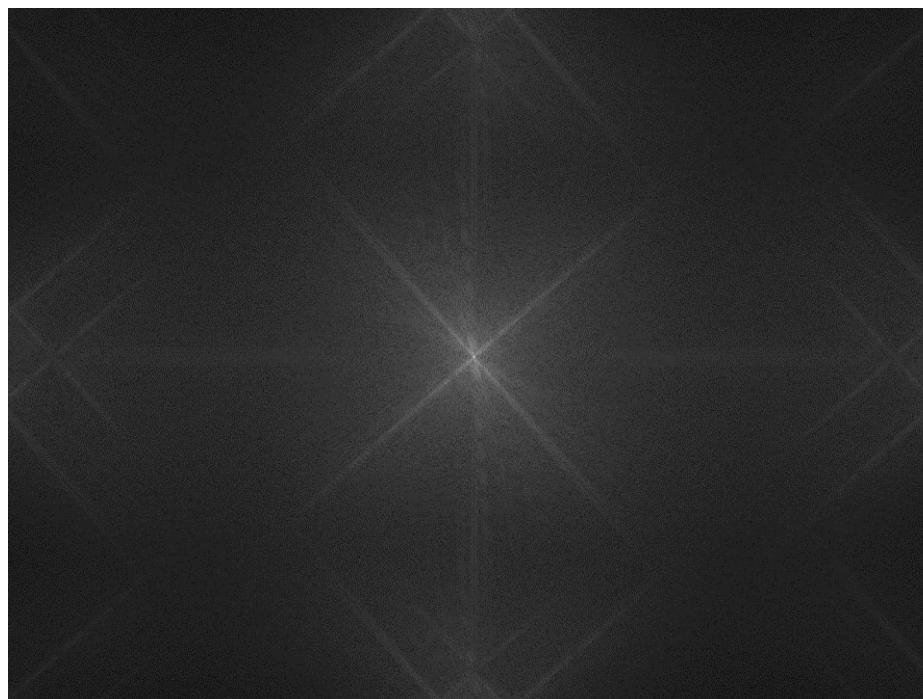


a b

FIGURE 4.29 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

7. The Basics of Filtering in the Frequency Domain

- MATLAB: s125Spectrum.m



7. The Basics of Filtering in the Frequency Domain

- Frequency Domain Filtering Fundamentals
 - Given a digital image $f(x,y)$ of $M \times N$ pixels, the basic filtering equation in which we are interested has the form:
$$g(x, y) = \mathfrak{F}^{-1}[H(u, v)F(u, v)]$$
 - $F(u,v)$ is the DFT of the input image, $f(x, y)$, and $H(u,v)$ is the filter; $g(x, y)$ is the filtered image.
 - We are now in a position to consider the filtering process in some detail.
 - One of the simplest filters we can construct is a filter $H(u, v)$ that is 0 at the center of the transform and 1 elsewhere.

7. The Basics of Filtering in the Frequency Domain

- Frequency Domain Filtering Fundamentals

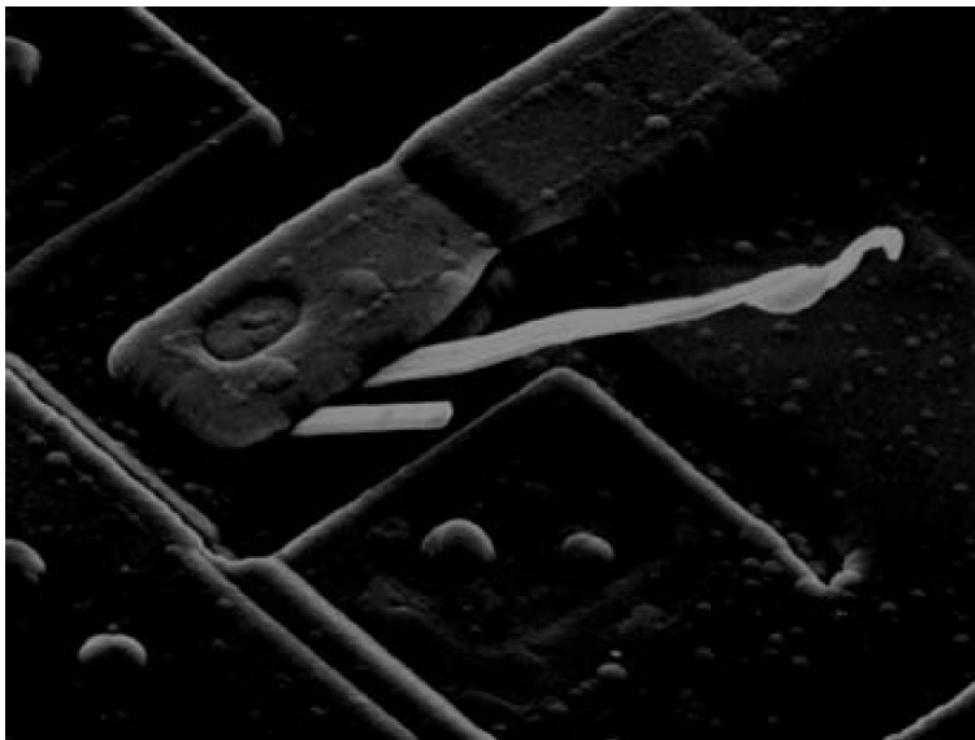


FIGURE 4.30
Result of filtering
the image in
Fig. 4.29(a) by
setting to 0 the
term $F(M/2, N/2)$
in the Fourier
transform.

7. The Basics of Filtering in the Frequency Domain

- Frequency Domain Filtering Fundamentals

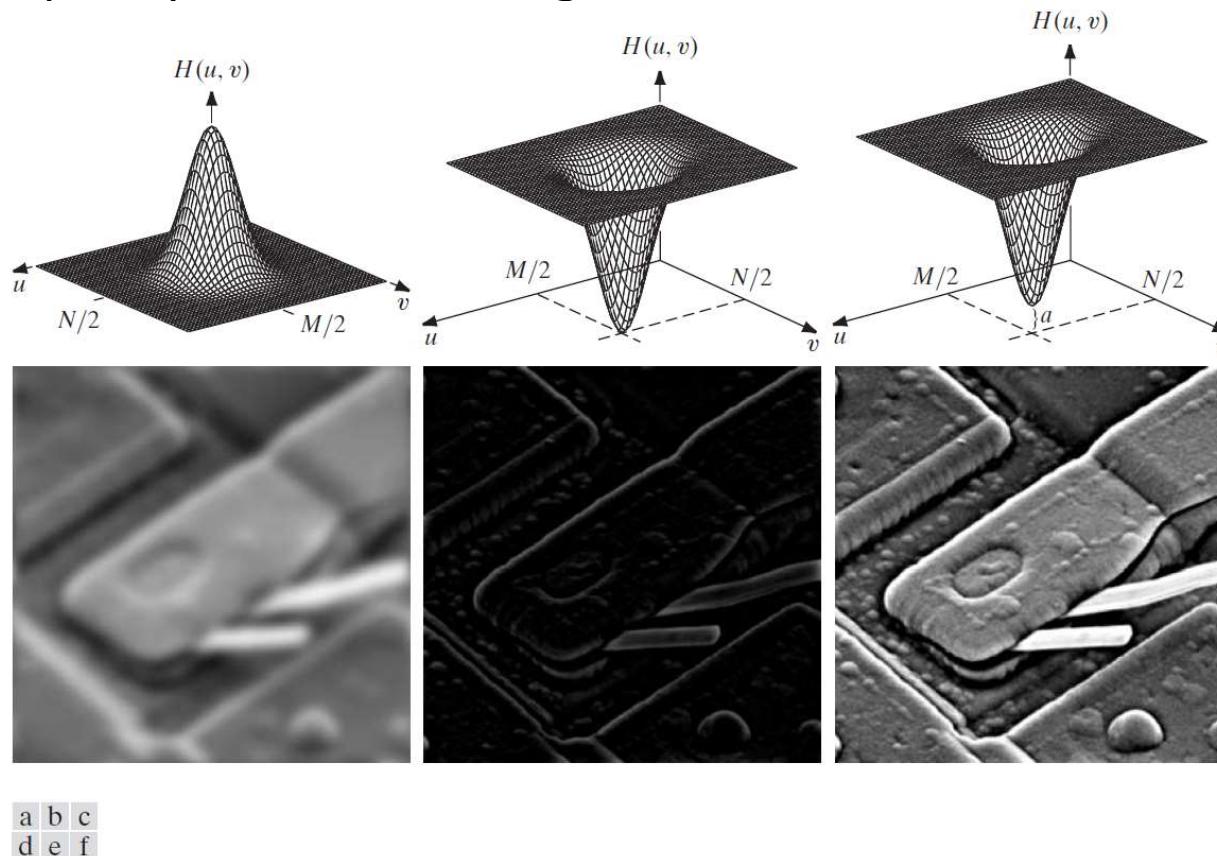
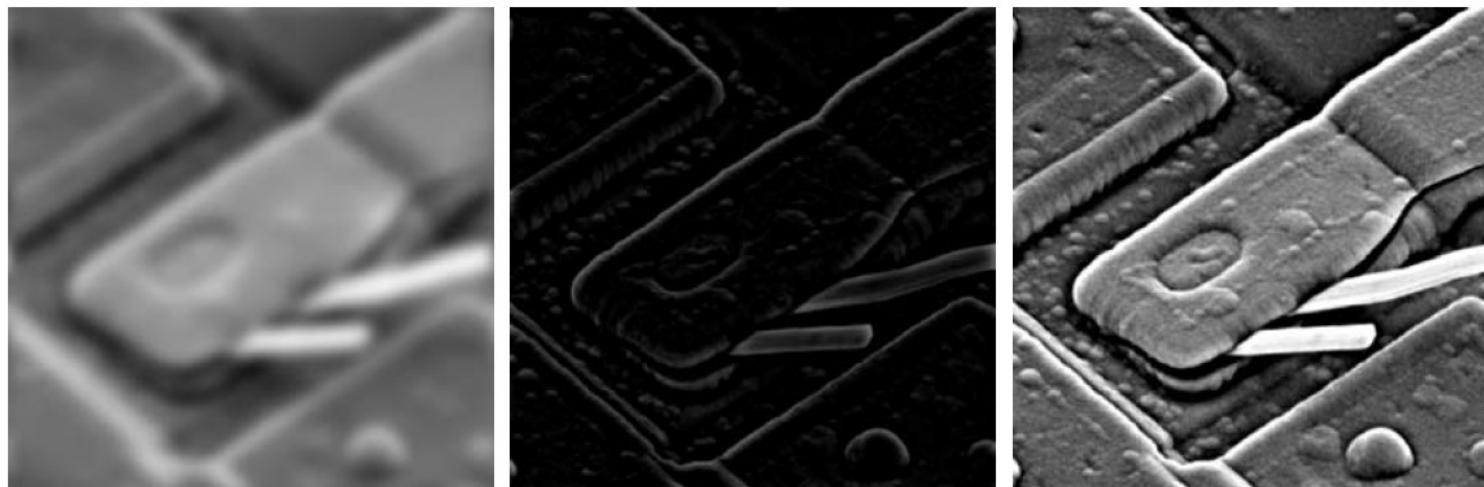


FIGURE 4.31 Top row: frequency domain filters. Bottom row: corresponding filtered images obtained using Eq. (4.7-1). We used $a = 0.85$ in (c) to obtain (f) (the height of the filter itself is 1). Compare (f) with Fig. 4.29(a).

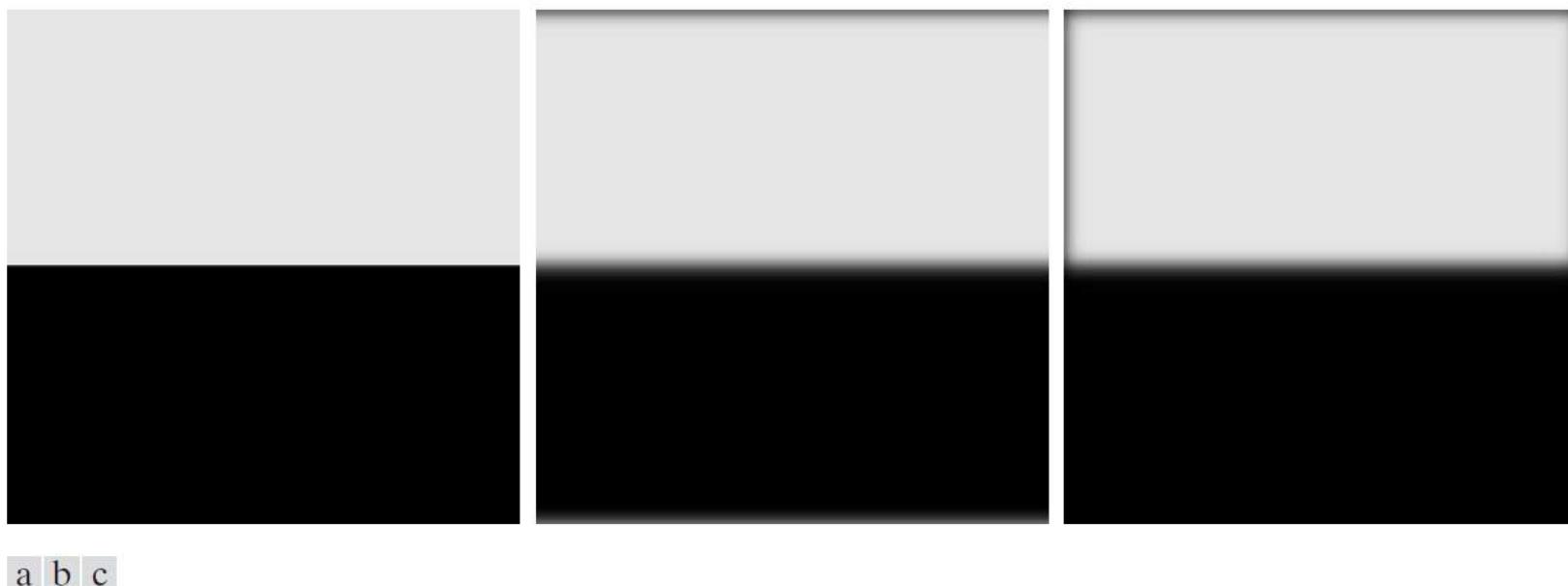
7. The Basics of Filtering in the Frequency Domain

- MATLAB: s130FilterExamples.m



7. The Basics of Filtering in the Frequency Domain

- Frequency Domain Filtering Fundamentals

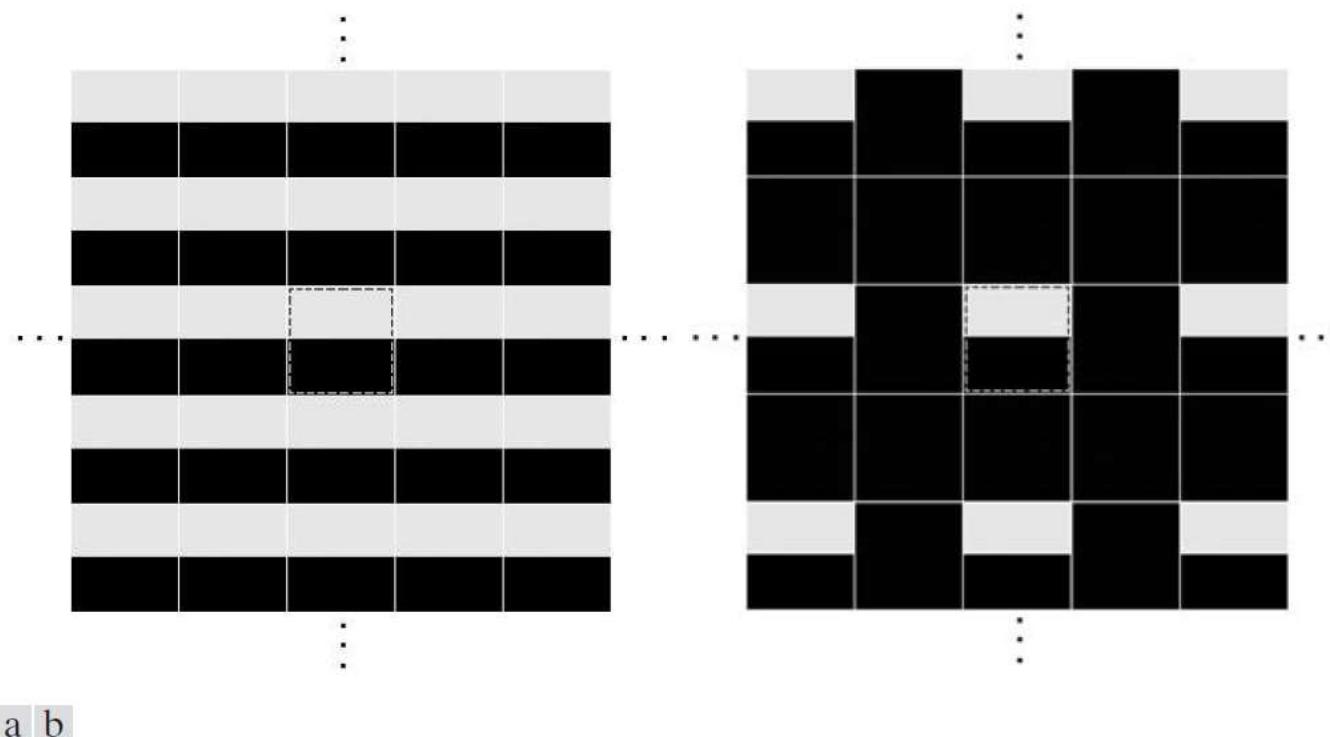


a b c

FIGURE 4.32 (a) A simple image. (b) Result of blurring with a Gaussian lowpass filter without padding. (c) Result of lowpass filtering with padding. Compare the light area of the vertical edges in (b) and (c).

7. The Basics of Filtering in the Frequency Domain

- Frequency Domain Filtering Fundamentals



a b

FIGURE 4.33 2-D image periodicity inherent in using the DFT. (a) Periodicity without image padding. (b) Periodicity after padding with 0s (black). The dashed areas in the center correspond to the image in Fig. 4.32(a). (The thin white lines in both images are superimposed for clarity; they are not part of the data.)

7. The Basics of Filtering in the Frequency Domain

- MATLAB: s129CircConv.m



7. The Basics of Filtering in the Frequency Domain

- Frequency Domain Filtering Fundamentals
 - **Low frequencies** in the transform are related to **slowly varying intensity** components in an image.
 - **High frequencies** are caused by **sharp transitions in intensity**
 - A filter that attenuates high frequencies while passing low frequencies (appropriately called a **lowpass filter**) would blur an image.
 - A filter with the opposite property (called a **highpass filter**) would enhance sharp detail.

7. The Basics of Filtering in the Frequency Domain

- Summary of Steps for Filtering in the Frequency Domain
 - I. Given an input image $f(x, y)$ of size $M \times N$ obtain the padding parameters P and Q from $P = 2M$ e $Q = 2N$.
 - II. Form a padded image $f_p(x, y)$ of size $P \times Q$.
 - III. Compute DFT of $f_{ps}(x, y) \rightarrow F_{ps}(u, v)$.
 - IV. Generate a real, symmetric filter function $H(u, v)$ of size $P \times Q$, with center at coordinates $P/2, Q/2$.

7. The Basics of Filtering in the Frequency Domain

- Summary of Steps for Filtering in the Frequency Domain

V. Calculate $G(u, v) = H(u, v).F_{ps}(u, v)$.

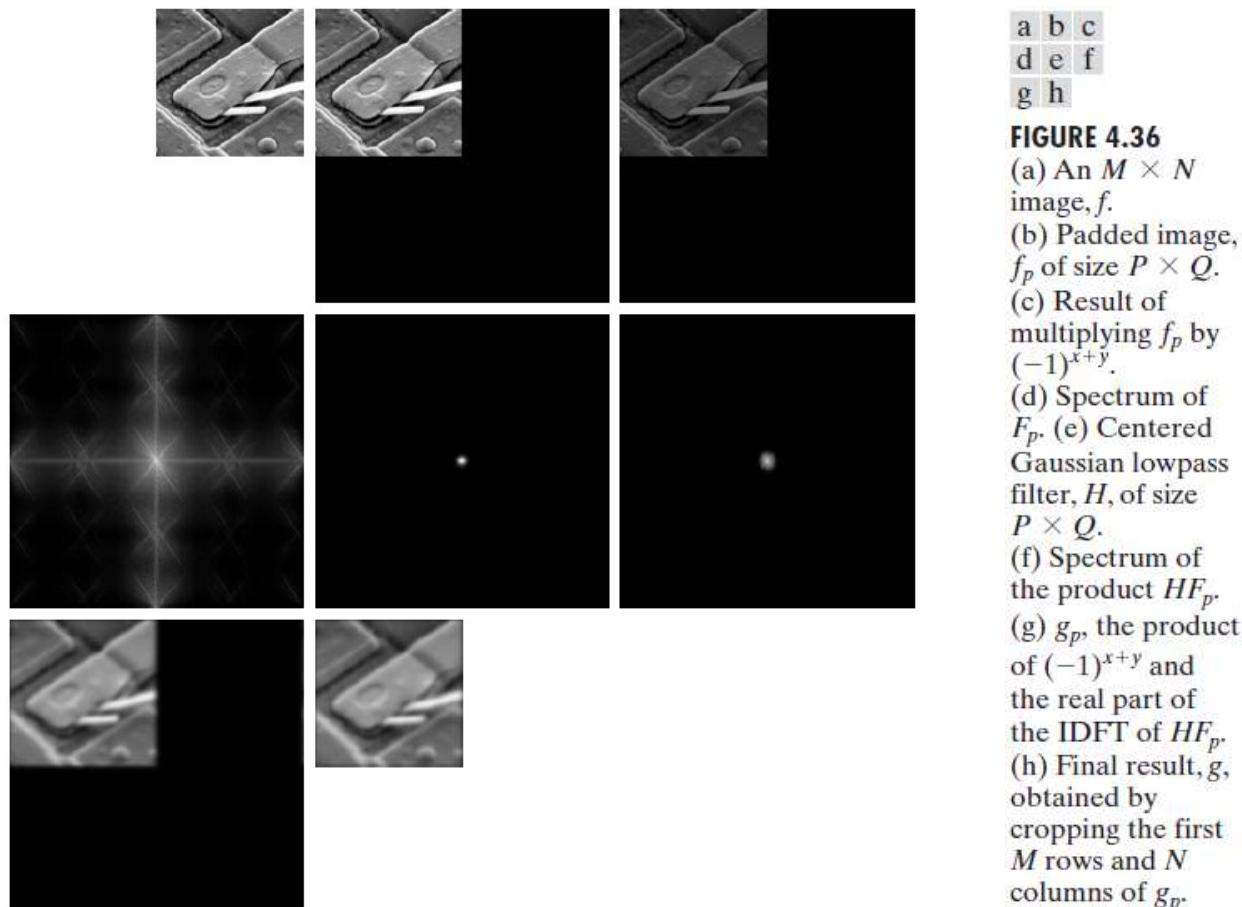
VI. Obtain the processed image:

$$g_p(x, y) = \{\text{real}[\mathfrak{J}^{-1}[G(u, v)]]\}(-1)^{x+y}$$

VII. Obtain the final processed result $g(x, y)$ by extracting the $M \times N$ region from the top, left quadrant $g_p(x, y)$.

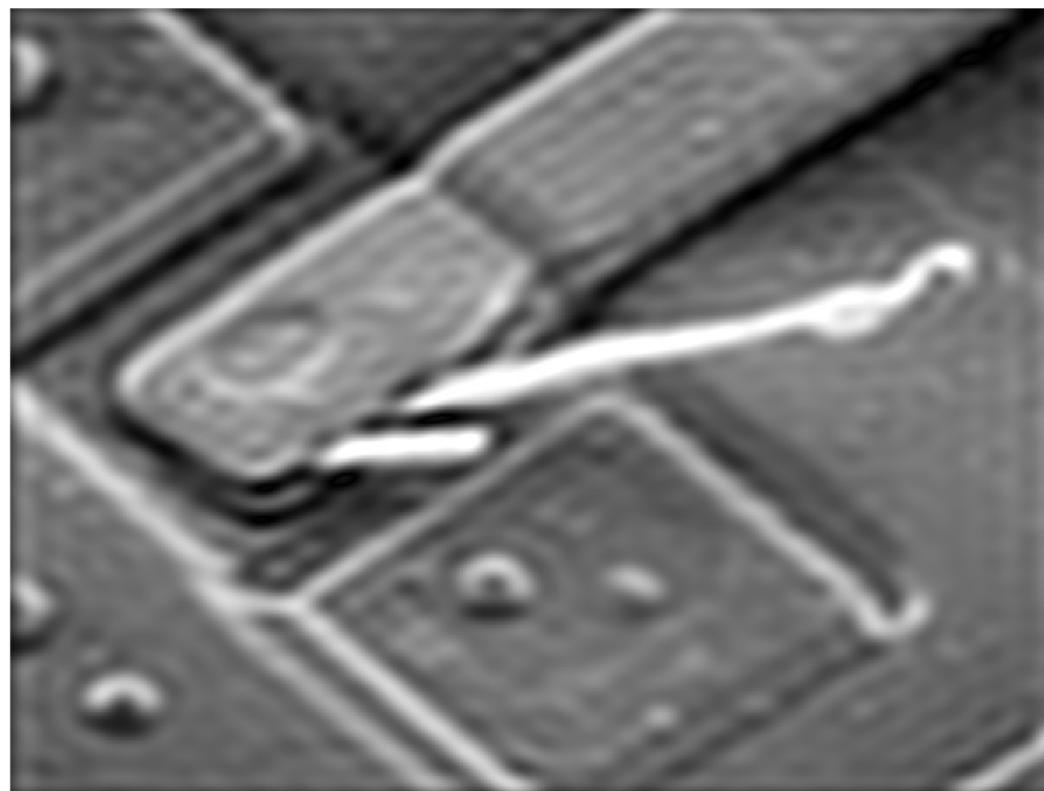
7. The Basics of Filtering in the Frequency Domain

- Summary of Steps for Filtering in the Frequency Domain



7. The Basics of Filtering in the Frequency Domain

- MATLAB: s138FilterAlg.m



7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains
 - The link between filtering in the spatial and frequency domains is the convolution theorem.
 - Given a filter $H(u, v)$, suppose that we want to find its equivalent representation in the spatial domain.
 - If we let $f(x, y) = \delta(x, y)$, then $F(u, v) = 1$.
 - The filtered output is

$$\mathcal{F}^{-1}\{F(u, v).H(u, v)\} = \mathcal{F}^{-1}\{H(u, v)\}$$

which is the corresponding filter in the spatial domain,
 $h(x, y)$.

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains
 - Because this filter can be obtained from the response of a frequency domain filter to an impulse, $h(x, y)$ sometimes is referred to as the **impulse response of $H(u, v)$** .
 - And because the impulse response is of finite length, such filters are called **finite impulse response** (FIR) filters.
 - These are the **only types** of linear spatial filters **considered in this course**.

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains
 - In practice, **we prefer** to implement **convolution** filtering using **small filter masks**.
 - However, filtering concepts are **more intuitive** in the **frequency domain**.
 - One way to take advantage of the properties of both domains is:
 1. To specify a filter in the frequency domain;
 2. Compute its IDFT; and
 3. Use the resulting, full-size spatial filter as a guide for constructing smaller spatial filter.

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

➤ Example (lowpass filter):

- ✓ Let $H(u)$ denote the 1-D frequency domain Gaussian filter:

$$H(u) = A e^{-u^2/2\sigma^2}$$

- ✓ The corresponding filter in the spatial domain is obtained by taking the inverse Fourier transform of $H(u)$:

$$h(x) = \sqrt{2\pi}\sigma A e^{-2\pi^2\sigma^2x^2}$$

- ✓ When $H(u)$ has a broad profile (large value of s), $h(x)$ has a narrow profile, and vice versa.

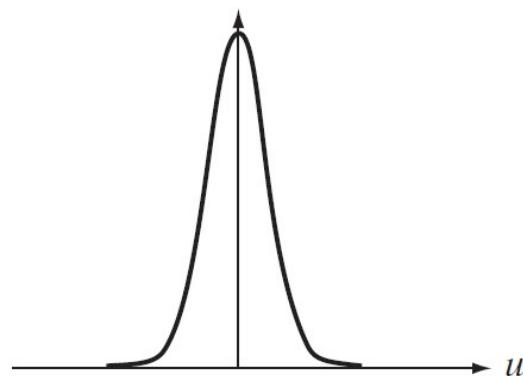
7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

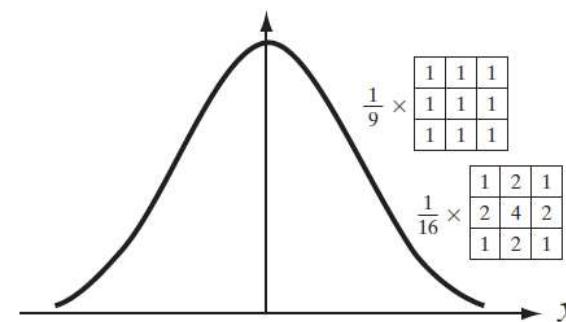
➤ Example (lowpass filter):

- ✓ Suppose that we want to use the shape of $h(x)$ as a guide for specifying the coefficients of a small spatial mask

$$H(u) = Ae^{-u^2/2\sigma^2}$$



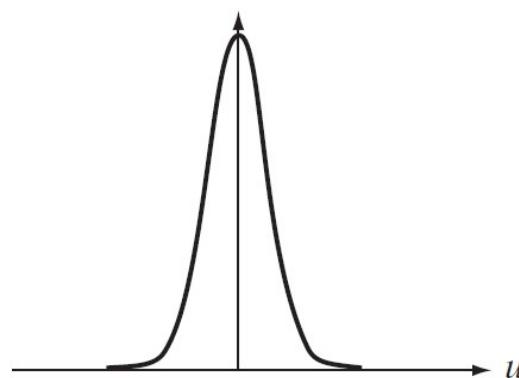
$$h(x) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2x^2}$$



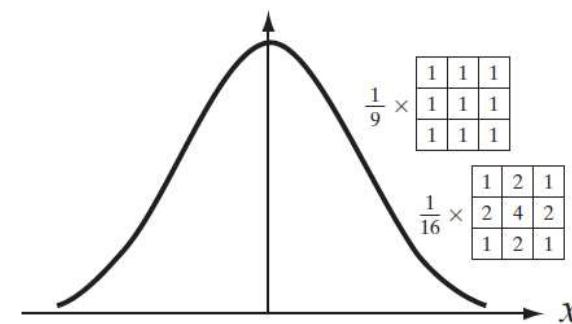
7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains
 - Example (lowpass filter):
 - ✓ All coefficients are positive.

$$H(u) = Ae^{-u^2/2\sigma^2}$$



$$h(x) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2x^2}$$



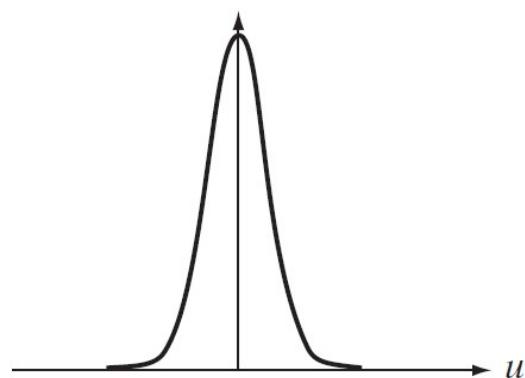
7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

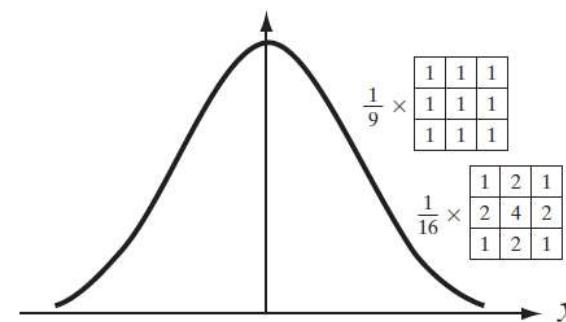
➤ Example (lowpass filter):

- ✓ The **narrower** the **frequency** domain **filter**, the more it will attenuate the low frequencies, resulting in increased blurring.

$$H(u) = Ae^{-u^2/2\sigma^2}$$



$$h(x) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2x^2}$$



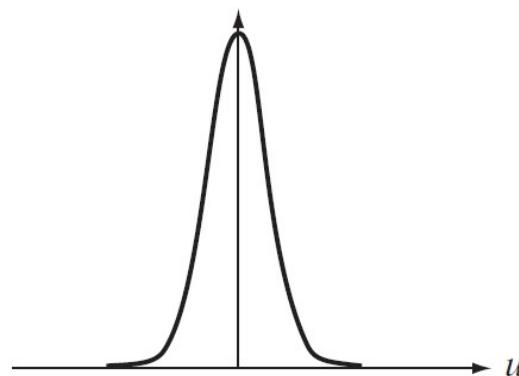
7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

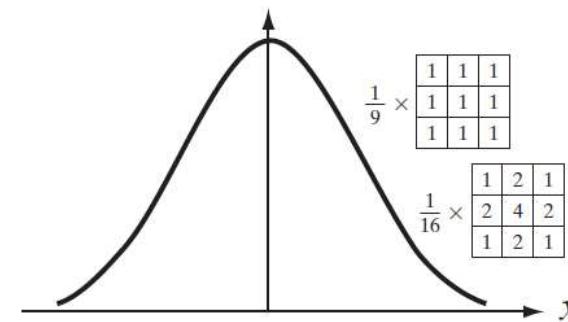
➤ Example (lowpass filter):

- ✓ In the **spatial** domain, this means that a **larger mask** must be used to **increase blurring**.

$$H(u) = Ae^{-u^2/2\sigma^2}$$



$$h(x) = \sqrt{2\pi}\sigma Ae^{-2\pi^2\sigma^2x^2}$$



7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

➤ Example (highpass filter):

- ✓ Difference of Gaussians in the frequency domain:

$$H(u) = Ae^{-u^2/2\sigma_1^2} - Be^{-u^2/2\sigma_2^2}$$

$$A \geq B \text{ and } \sigma_1 > \sigma_2$$

- ✓ The corresponding filter in the spatial domain is

$$h(x) = \sqrt{2\pi}\sigma_1 A e^{-2\pi^2\sigma_1^2x^2} - \sqrt{2\pi}\sigma_2 B e^{-2\pi^2\sigma_2^2x^2}$$

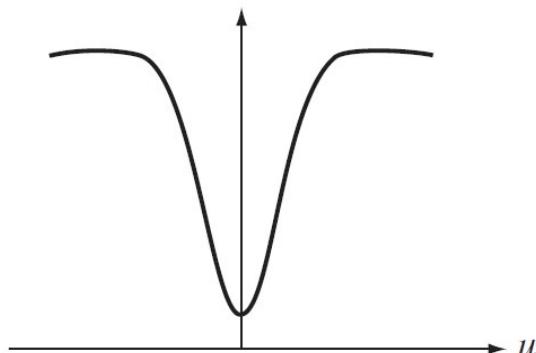
7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

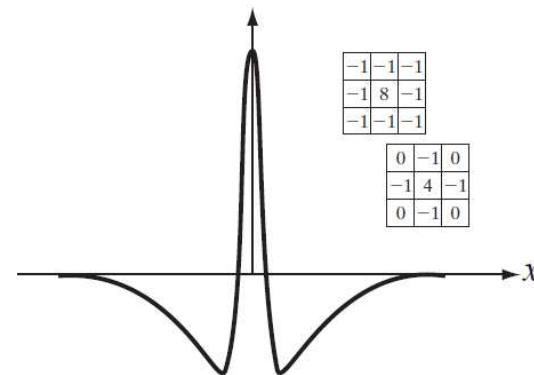
➤ Example (highpass filter):

- Has a positive center term with negative terms on either side.

$$H(u) = Ae^{-u^2/2\sigma_1^2} - Be^{-u^2/2\sigma_2^2}$$



$$h(x) = \sqrt{2\pi}\sigma_1 A e^{-2\pi^2\sigma_1^2x^2} - \sqrt{2\pi}\sigma_2 B e^{-2\pi^2\sigma_2^2x^2}$$



7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

➤ Example (lowpass and highpass filter):

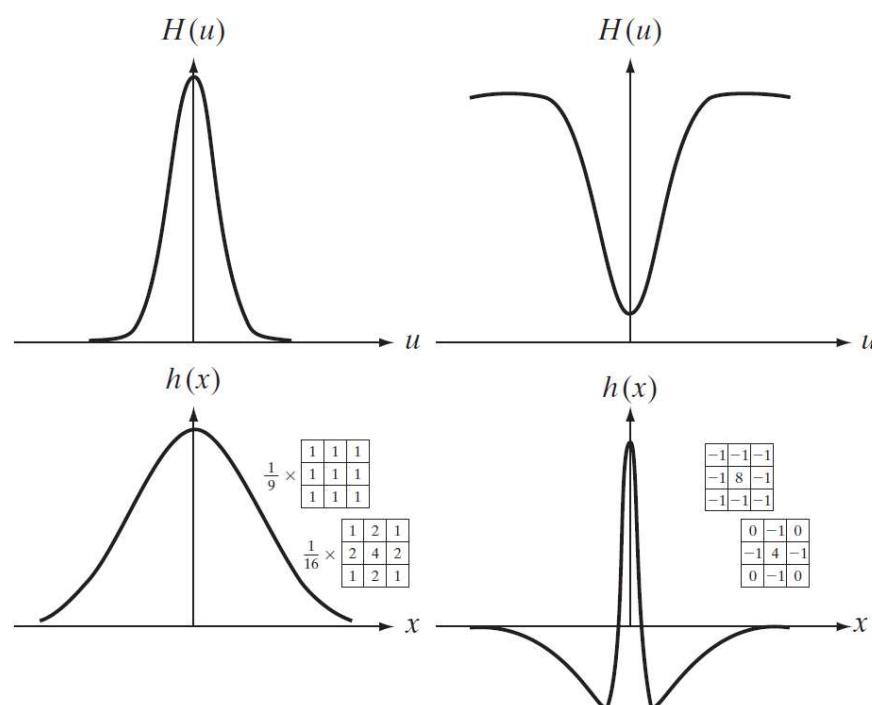


FIGURE 4.37

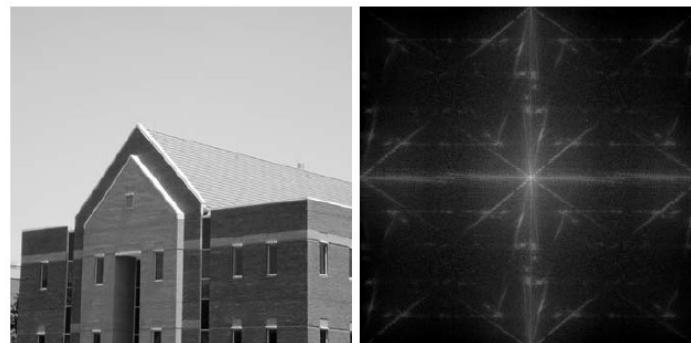
(a) A 1-D Gaussian lowpass filter in the frequency domain.
 (b) Spatial lowpass filter corresponding to (a).
 (c) Gaussian highpass filter in the frequency domain.
 (d) Spatial highpass filter corresponding to (c). The small 2-D masks shown are spatial filters we used in Chapter 3.

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains
 - Frequency domain can be viewed as a “laboratory”.
 - Some tasks that would be difficult to formulate in the spatial domain become easier in the frequency domain.
 - Once we have selected a filter in the frequency domain, the implementation is usually done in the spatial domain.
 - One approach is to specify small spatial masks that attempt to capture the “essence” of the full filter function in the spatial domain.
 - There are more formal approaches to design 2-D digital filters.

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains
 - In this example, we start with a spatial mask and show how to generate the filter in the frequency domain.
 - To keep matters simple, we use the 3x3 Sobel vertical edge detector.
 - The figure below shows a pixel image that we wish to filter and its spectrum.



a b

FIGURE 4.38
(a) Image of a
building, and
(b) its spectrum.

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains
 - Figure 4.39(a) shows the Sobel mask, $h(x, y)$.
 - Because the input image is of size 600×600 pixels and the filter is of size 3×3 we avoid wraparound error by padding f and h to size 602×602 pixels.
 - The Sobel mask exhibits odd symmetry. To maintain this symmetry, we place $h(x, y)$ so that its center is at the center of the padded array forming $h_p(x, y)$.

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

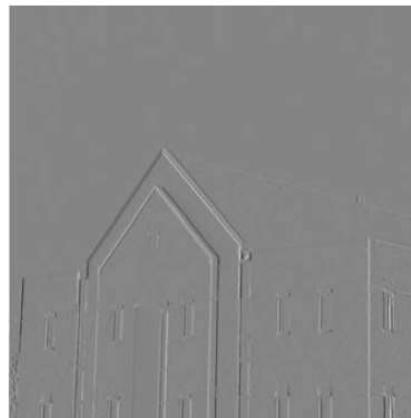
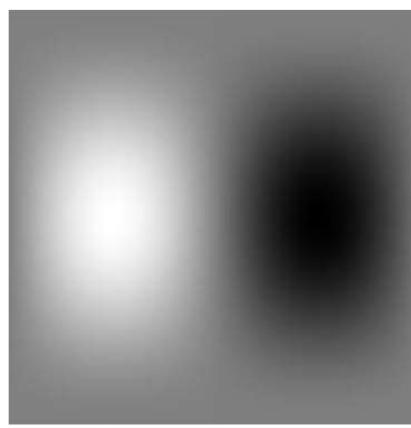
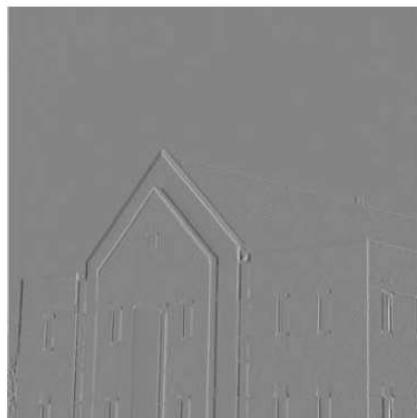
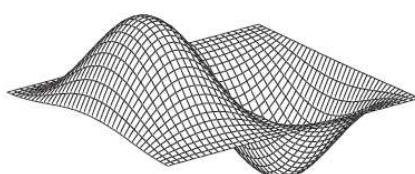
➤ The procedure used to generate $H(u, v)$ is:

1. Multiply $h_p(x, y)$ by $(-1)^{x+y}$ to center the frequency domain filter;
2. Compute the forward DFT of the result in 1;
3. Set the real part of the resulting DFT to 0 to account for parasitic real parts - $H(x, y)$ has to be purely imaginary; and
4. Multiply the result by $(-1)^{u+v}$ (reverses the multiplication of $H(u, v)$ by $(-1)^{u+v}$ which is implicit when $h(x, y)$ was moved to the center of $h_p(x, y)$).

7. The Basics of Filtering in the Frequency Domain

- Correspondence Between Filtering in the Spatial and Frequency Domains

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



a	b
c	d

FIGURE 4.39
(a) A spatial mask and perspective plot of its corresponding frequency domain filter. (b) Filter shown as an image. (c) Result of filtering Fig. 4.38(a) in the frequency domain with the filter in (b). (d) Result of filtering the same image with the spatial filter in (a). The results are identical.

7. The Basics of Filtering in the Frequency Domain

- MATLAB: s153FilterGen.m

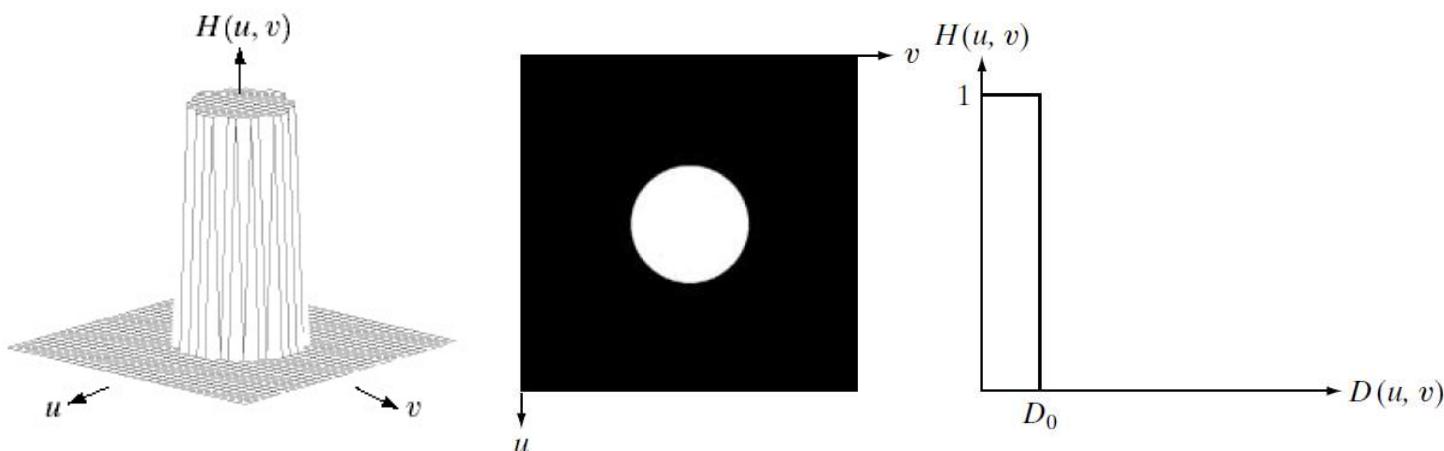


8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

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

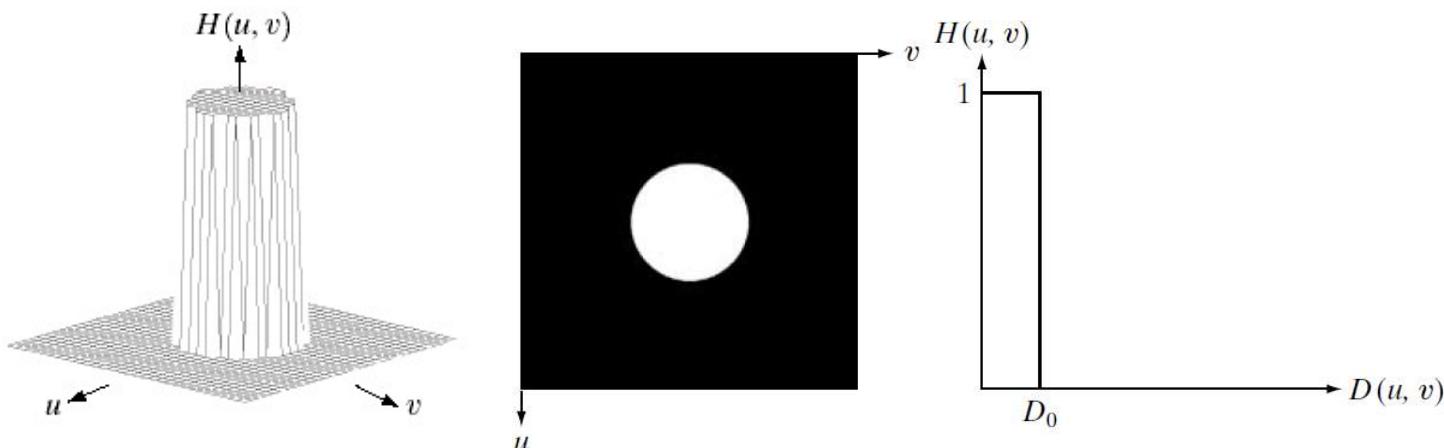
$$D(u, v) = [(u - P/2)^2 + (v - Q/2)^2]^{1/2}$$



8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

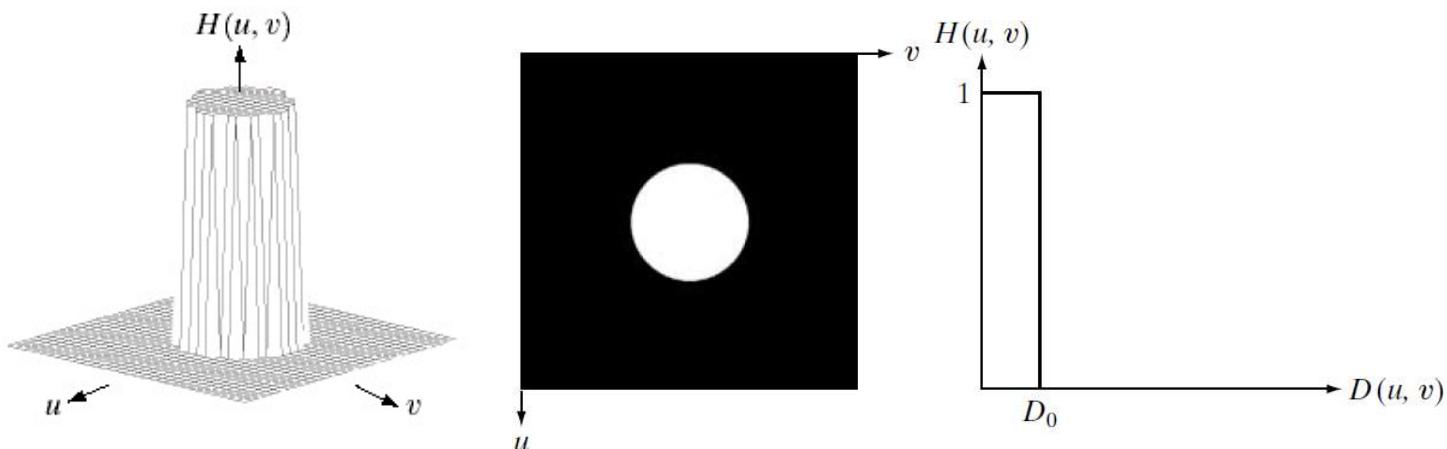
- The name **ideal** indicates that all frequencies on or inside a circle of radius D_0 are passed, without attenuation.
- For an ILPF cross section, the point D_0 of transition between 1 and 0 is called the *cutoff frequency*.



8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

- The sharp cutoff frequencies of an ILPF cannot be realized with electronic components, although they certainly can be simulated in a computer.
- The lowpass filters introduced in this chapter are compared by studying their behavior as a function of the same cutoff frequencies.



8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

- One way to establish a set of standard cutoff frequency loci is to compute circles that enclose specified amounts of **total image power** P_T .
 - This quantity is obtained by summing the components of the power spectrum of the padded images at each point

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

$$\begin{aligned} P(u, v) &= |F(u, v)|^2 \\ &= R^2(u, v) + I^2(u, v) \end{aligned}$$

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

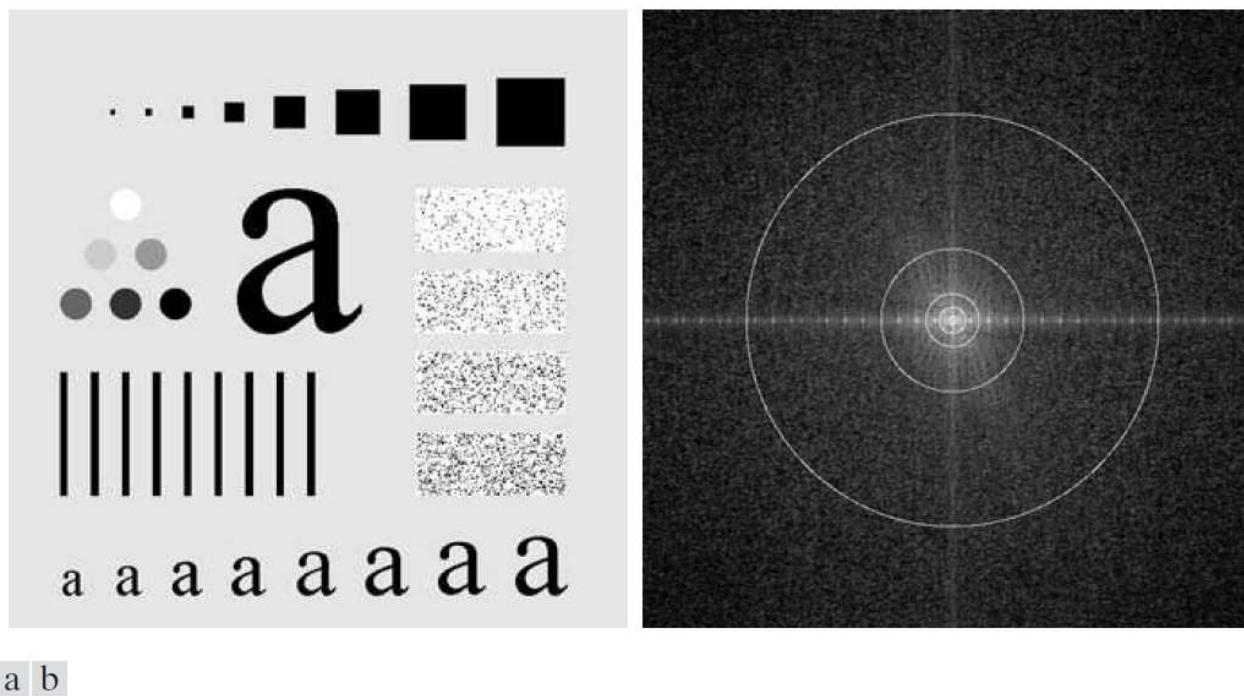
- If the DFT has been centered, a circle of radius D_0 with origin at the center of the frequency rectangle encloses α percent of the power, where

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

and the summation is taken over values of (u, v) that lie inside the circle or on its boundary.

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters



a b

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

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

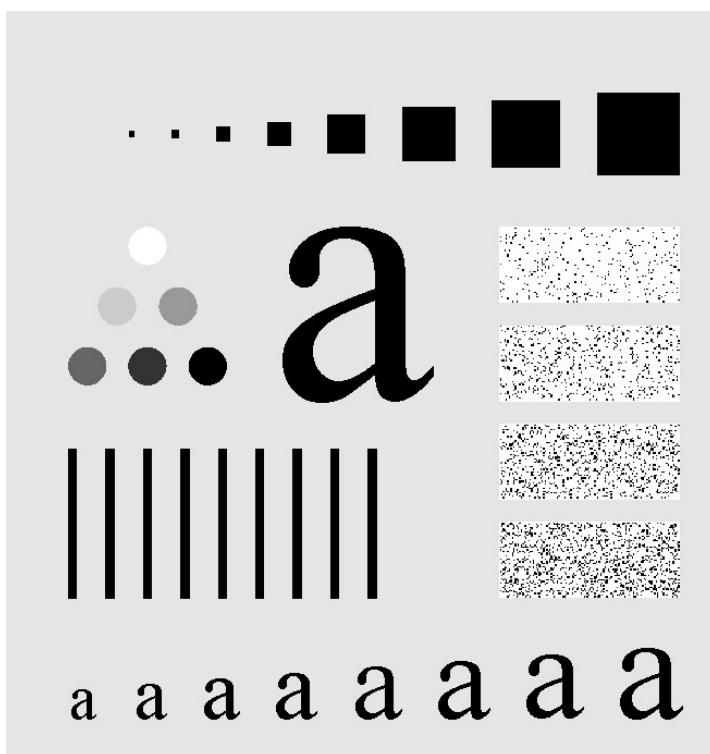


FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters



FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters



FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters



FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

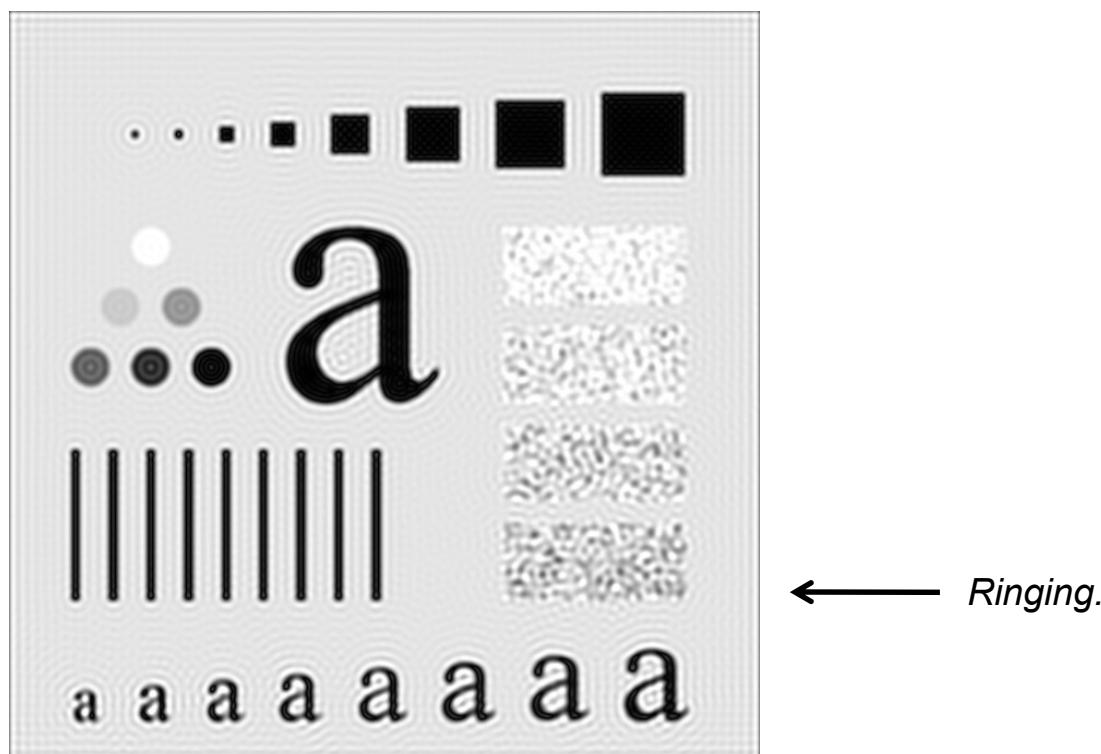


FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

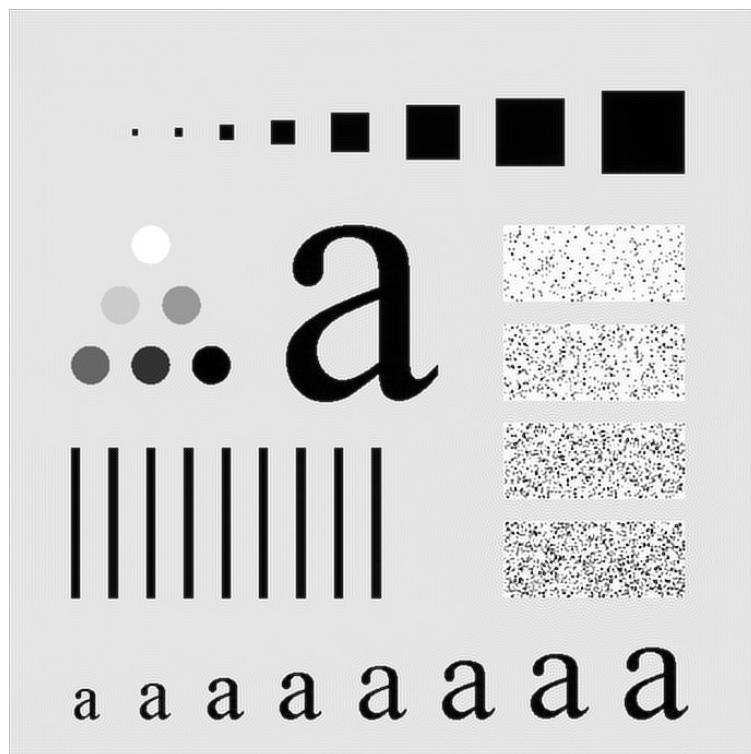
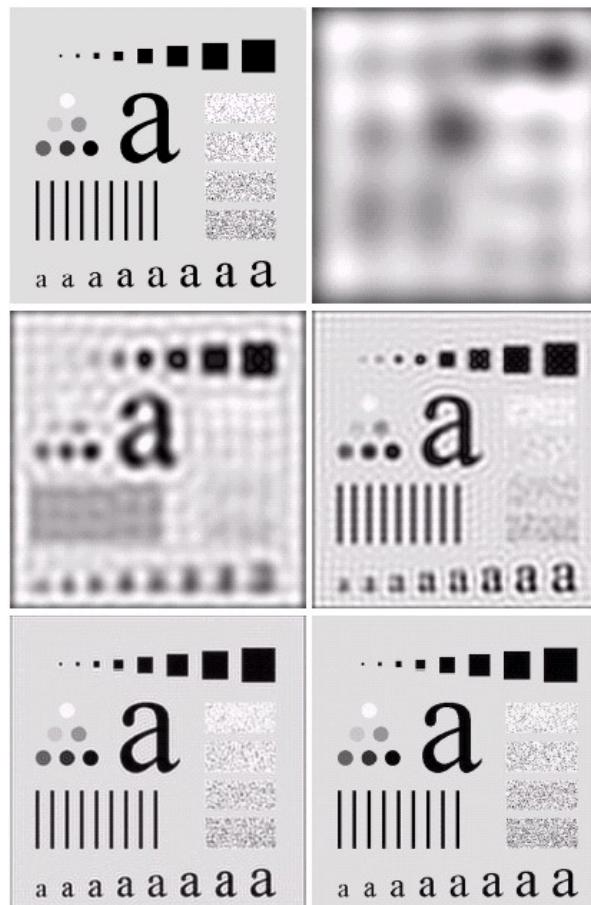


FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

8. Image Smoothing Using Frequency Domain Filters

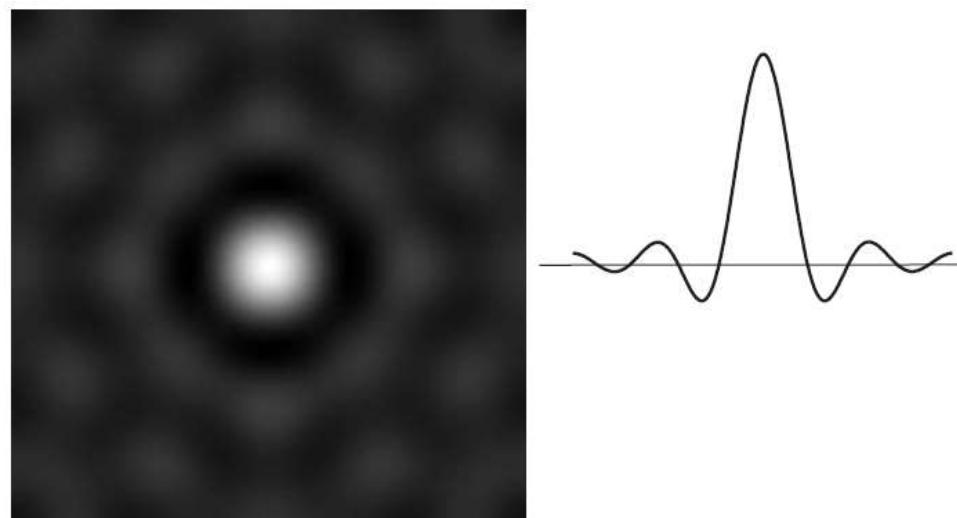
- Ideal Lowpass Filters



8. Image Smoothing Using Frequency Domain Filters

- Ideal Lowpass Filters

- The blurring and ringing properties of ILPFs can be explained using the convolution theorem.



a b

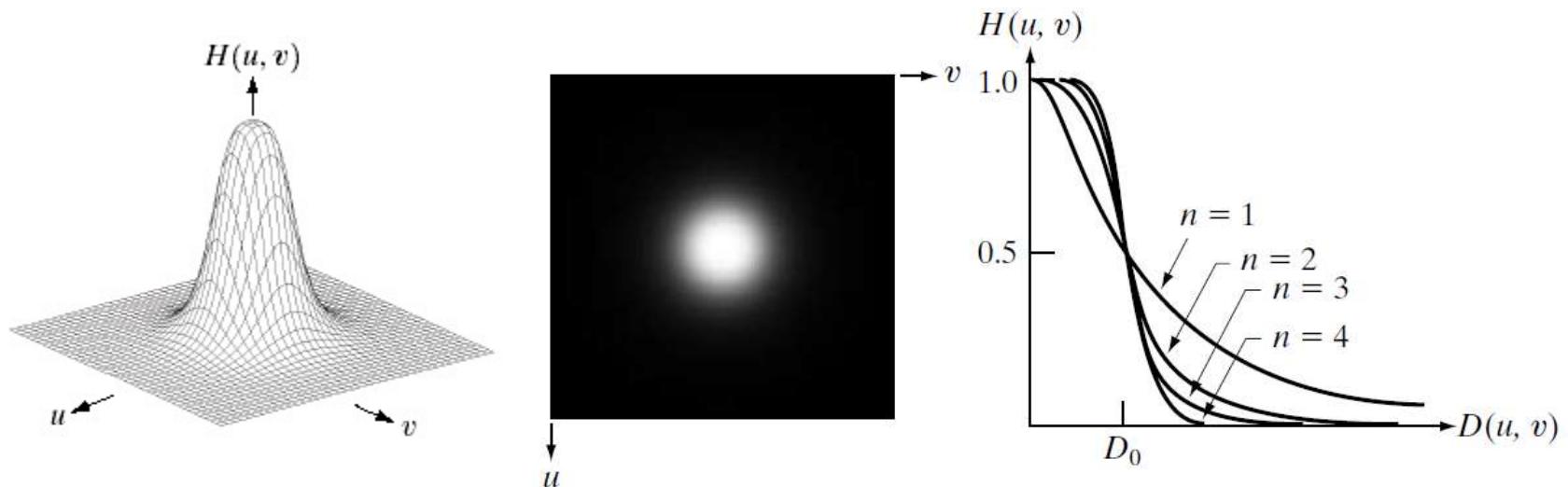
FIGURE 4.43
(a) Representation in the spatial domain of an ILPF of radius 5 and size 1000×1000 .
(b) Intensity profile of a horizontal line passing through the center of the image.

8. Image Smoothing Using Frequency Domain Filters

- Butterworth Lowpass Filters

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

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

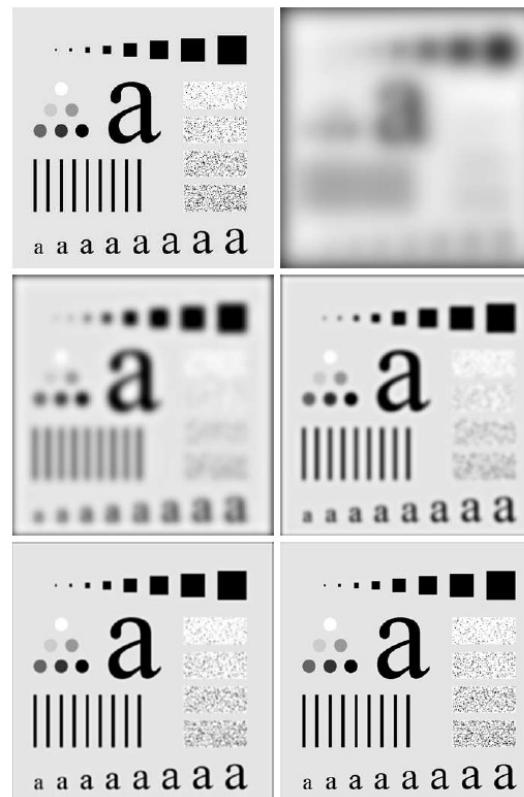


8. Image Smoothing Using Frequency Domain Filters

- Butterworth Lowpass Filters
 - Unlike the ILPF, the **BLPF** transfer function **does not have a sharp discontinuity** that gives a clear cutoff between passed and filtered frequencies.
 - For filters with smooth transfer functions, cutoff frequency defined as those for which $H(u, v)$ is **down to a certain fraction of its maximum** value (eg. 50%).
 - A BLPF of **order 1** has **no ringing** in the spatial domain. Ringing generally is **imperceptible** in filters of **order 2**, but can become **significant** in filters of **higher order**.

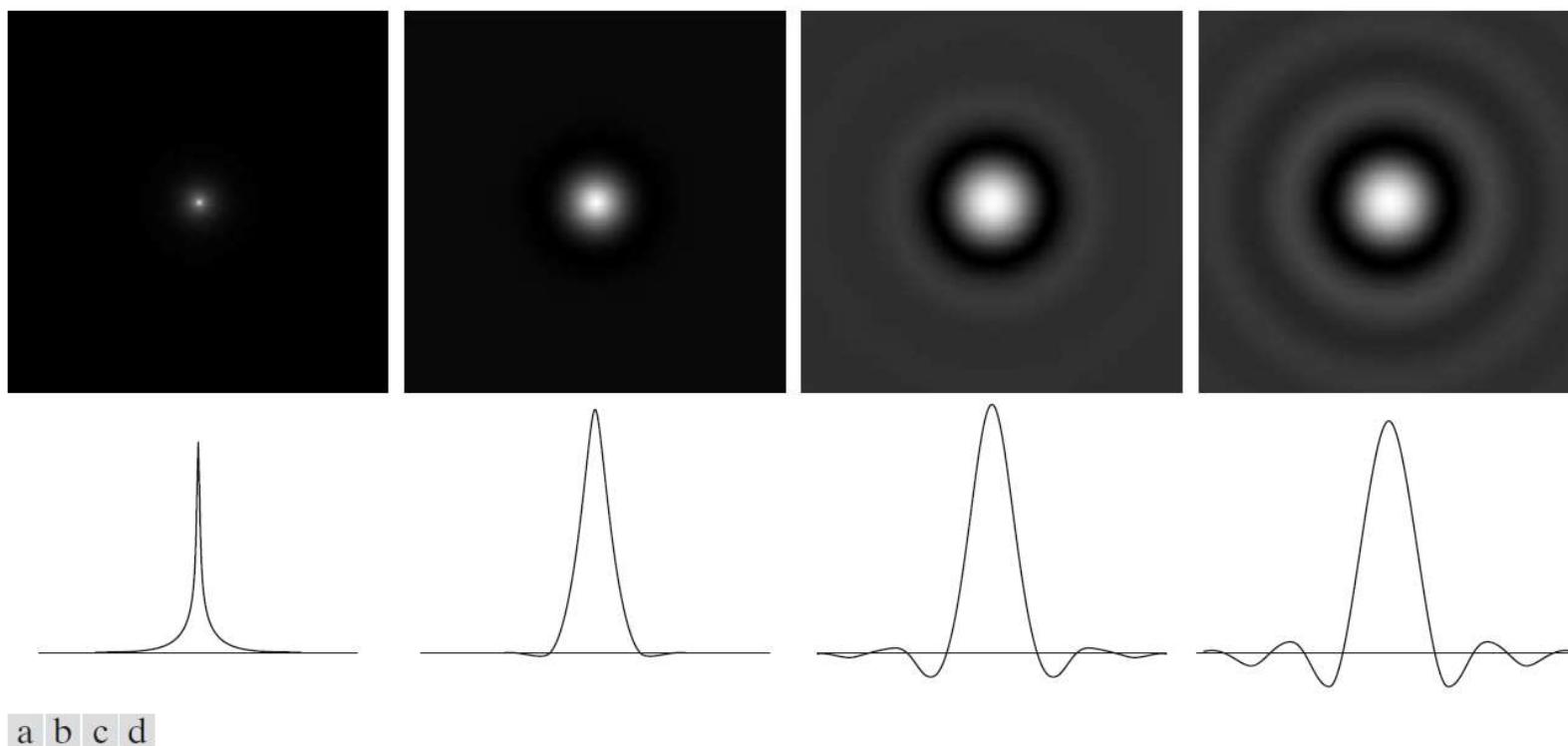
8. Image Smoothing Using Frequency Domain Filters

- Butterworth Lowpass Filters
 - BLPFs of order 2. Cutoff frequencies same as before.



8. Image Smoothing Using Frequency Domain Filters

- Butterworth Lowpass Filters



a b c d

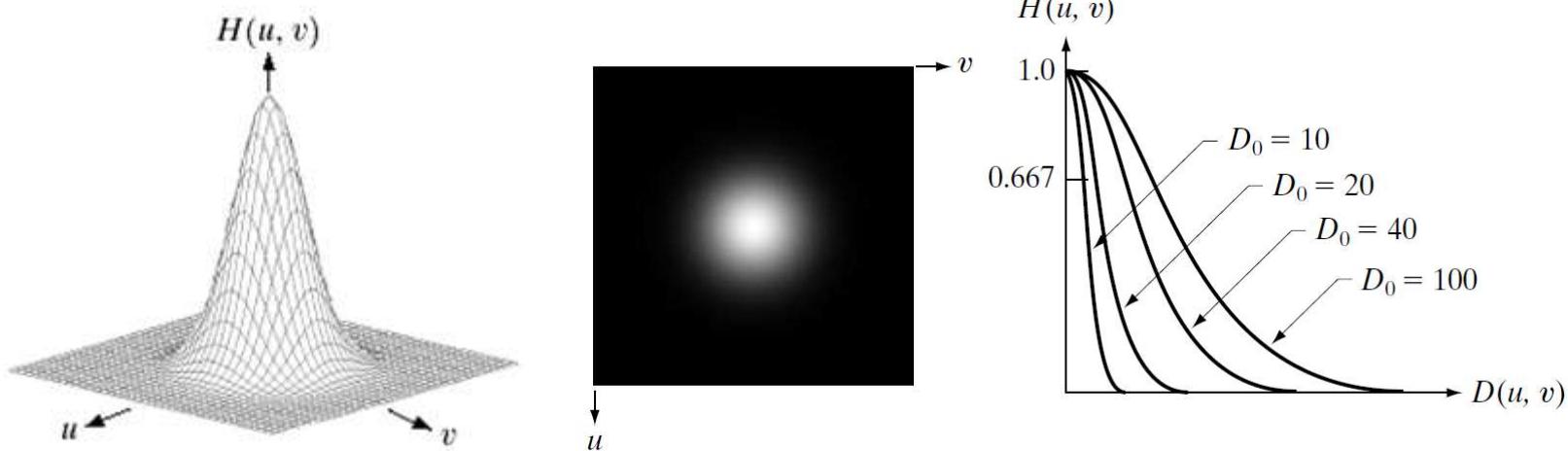
FIGURE 4.46 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding intensity profiles through the center of the filters (the size in all cases is 1000×1000 and the cutoff frequency is 5). Observe how ringing increases as a function of filter order.

8. Image Smoothing Using Frequency Domain Filters

- Gaussian Lowpass Filters

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

- D_0 is the cutoff frequency. When $D(u, v) = D_0$, the GLPF is down to 0.667 of its maximum value.



8. Image Smoothing Using Frequency Domain Filters

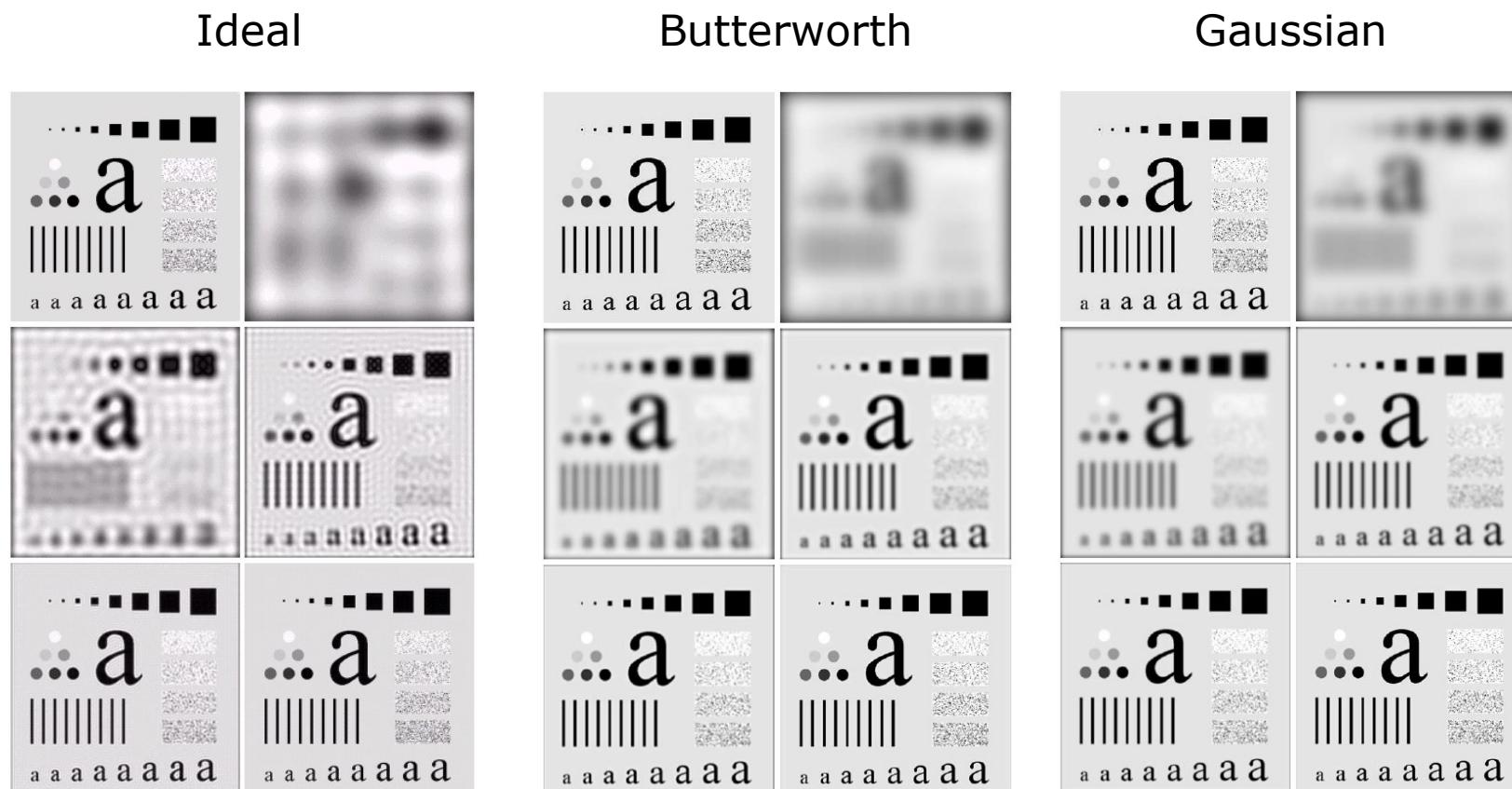
- Gaussian Lowpass Filters

➤ Cutoff frequencies same as before. No ringing effect.



8. Image Smoothing Using Frequency Domain Filters

- Comparison



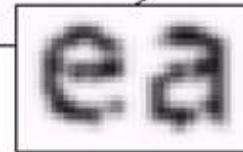
8. Image Smoothing Using Frequency Domain Filters

- Additional Examples of Lowpass Filtering

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

FIGURE 4.49
(a) Sample text of low resolution (note broken characters in magnified view).
(b) Result of filtering with a GLPF (broken character segments were joined).

8. Image Smoothing Using Frequency Domain Filters

- Additional Examples of Lowpass Filtering



a b c

FIGURE 4.50 (a) Original image (784×732 pixels). (b) Result of filtering using a GLPF with $D_0 = 100$. (c) Result of filtering using a GLPF with $D_0 = 80$. Note the reduction in fine skin lines in the magnified sections in (b) and (c).

8. Image Smoothing Using Frequency Domain Filters

- Additional Examples of Lowpass Filtering



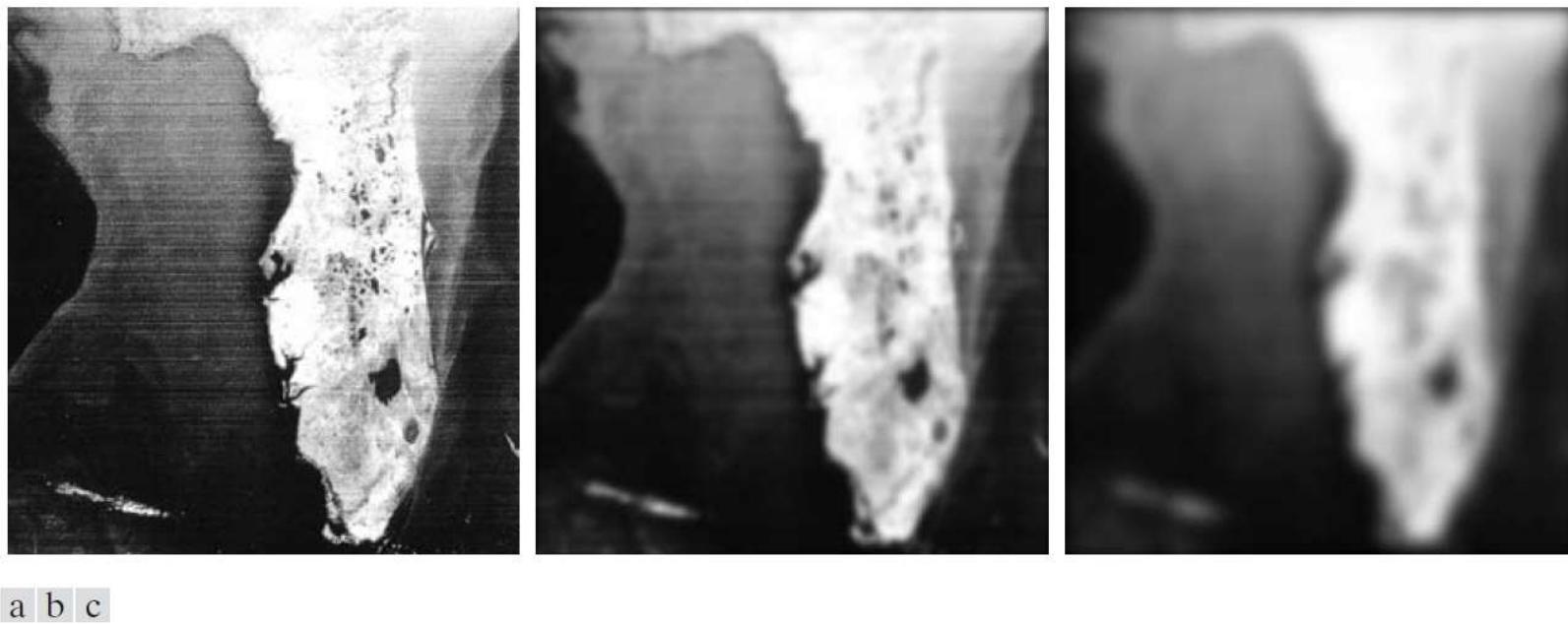
8. Image Smoothing Using Frequency Domain Filters

- Additional Examples of Lowpass Filtering



8. Image Smoothing Using Frequency Domain Filters

- Additional Examples of Lowpass Filtering
 - Here, the objective is to blur out as much detail as possible while leaving large features recognizable.



a b c

FIGURE 4.51 (a) Image showing prominent horizontal scan lines. (b) Result of filtering using a GLPF with $D_0 = 50$. (c) Result of using a GLPF with $D_0 = 20$. (Original image courtesy of NOAA.)

9. Image Sharpening Using Frequency Domain Filters

- Highpass Filtering

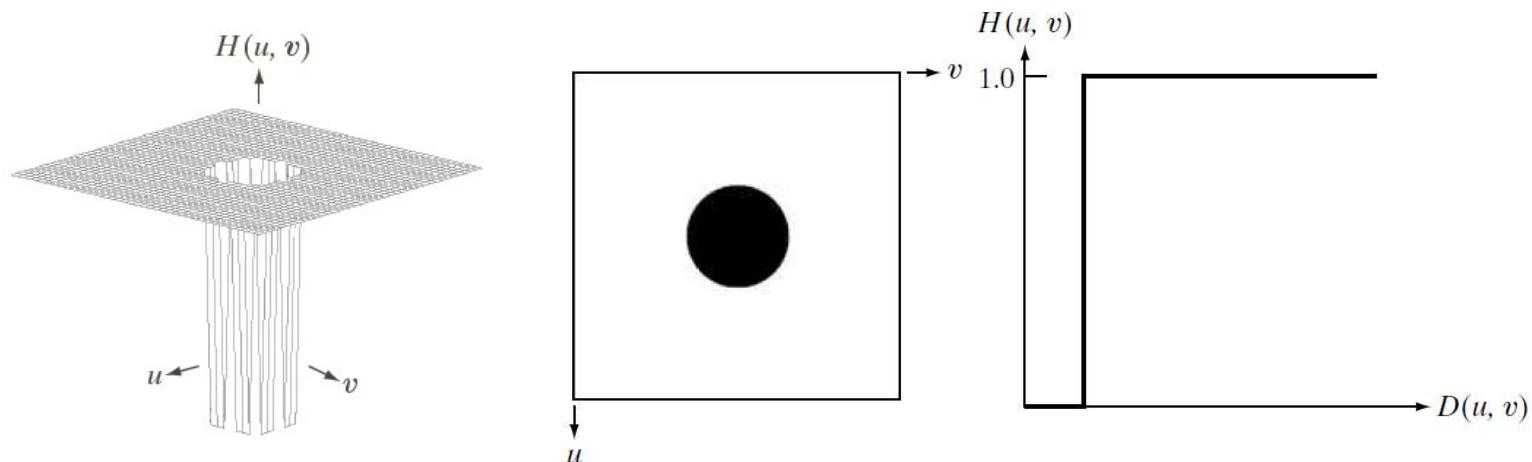
- A highpass filter is obtained from a given lowpass filter using the equation

$$H_{\text{HP}}(u, v) = 1 - H_{\text{LP}}(u, v)$$

9. Image Sharpening Using Frequency Domain Filters

- Ideal Highpass Filters

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



9. Image Sharpening Using Frequency Domain Filters

- Ideal Highpass Filters

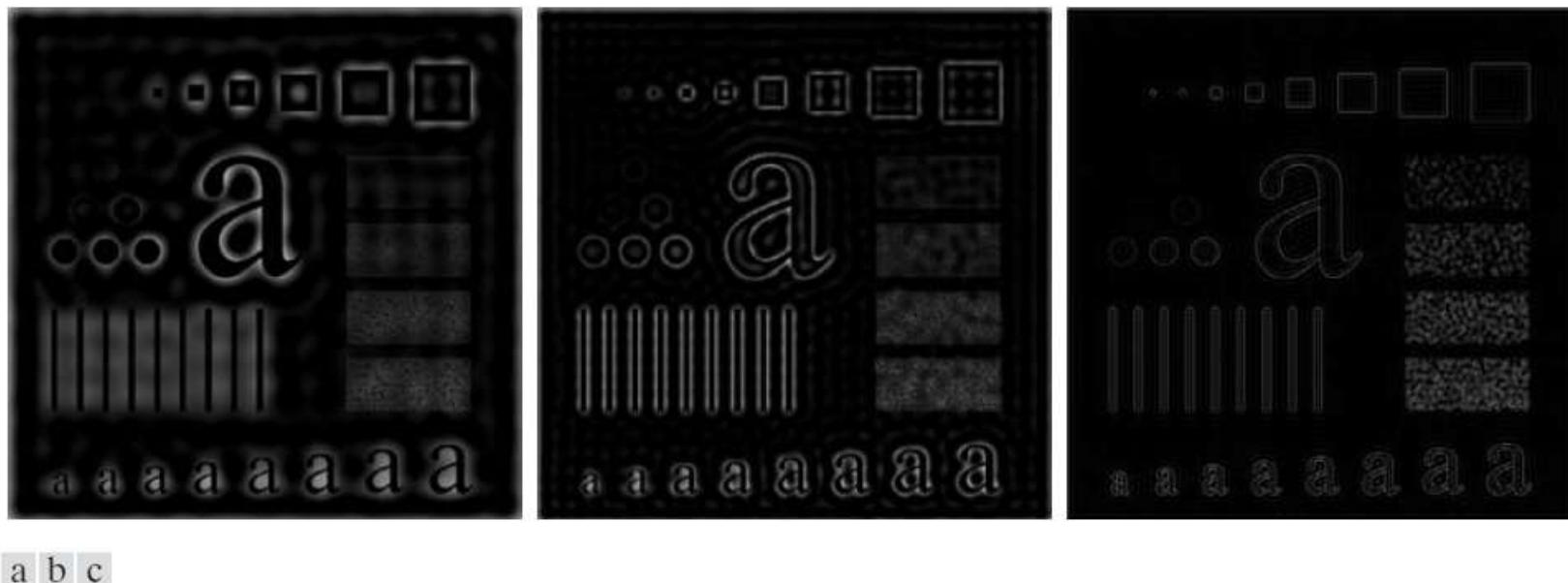
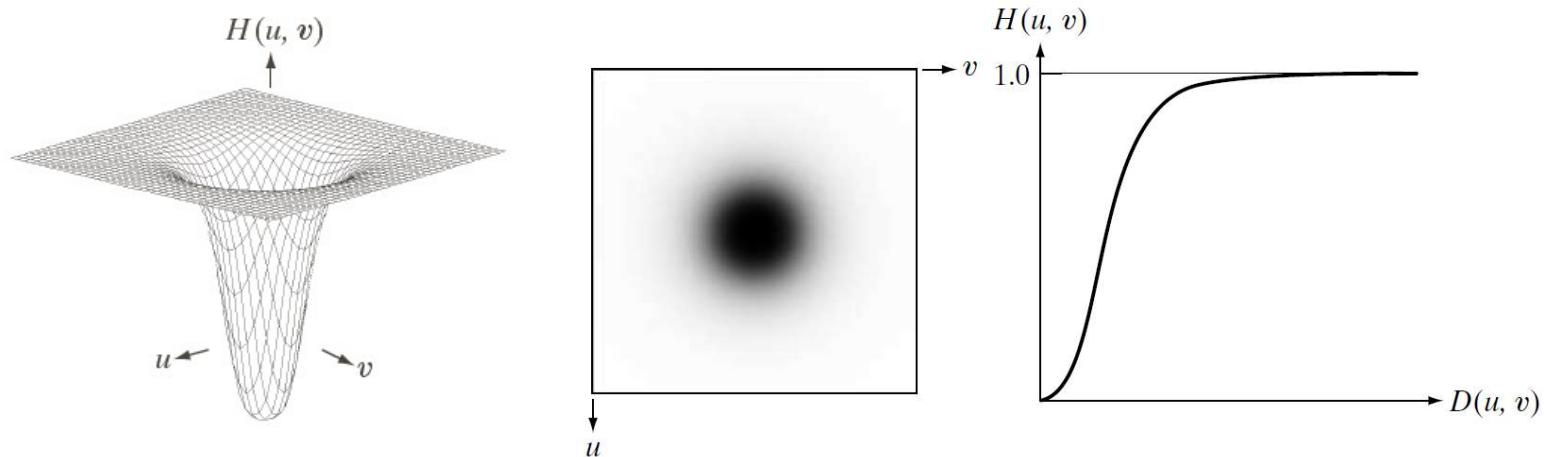


FIGURE 4.54 Results of highpass filtering the image in Fig. 4.41(a) using an IHPF with $D_0 = 30, 60$, and 160 .

9. Image Sharpening Using Frequency Domain Filters

- Butterworth Highpass Filters

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



9. Image Sharpening Using Frequency Domain Filters

- Butterworth Highpass Filters

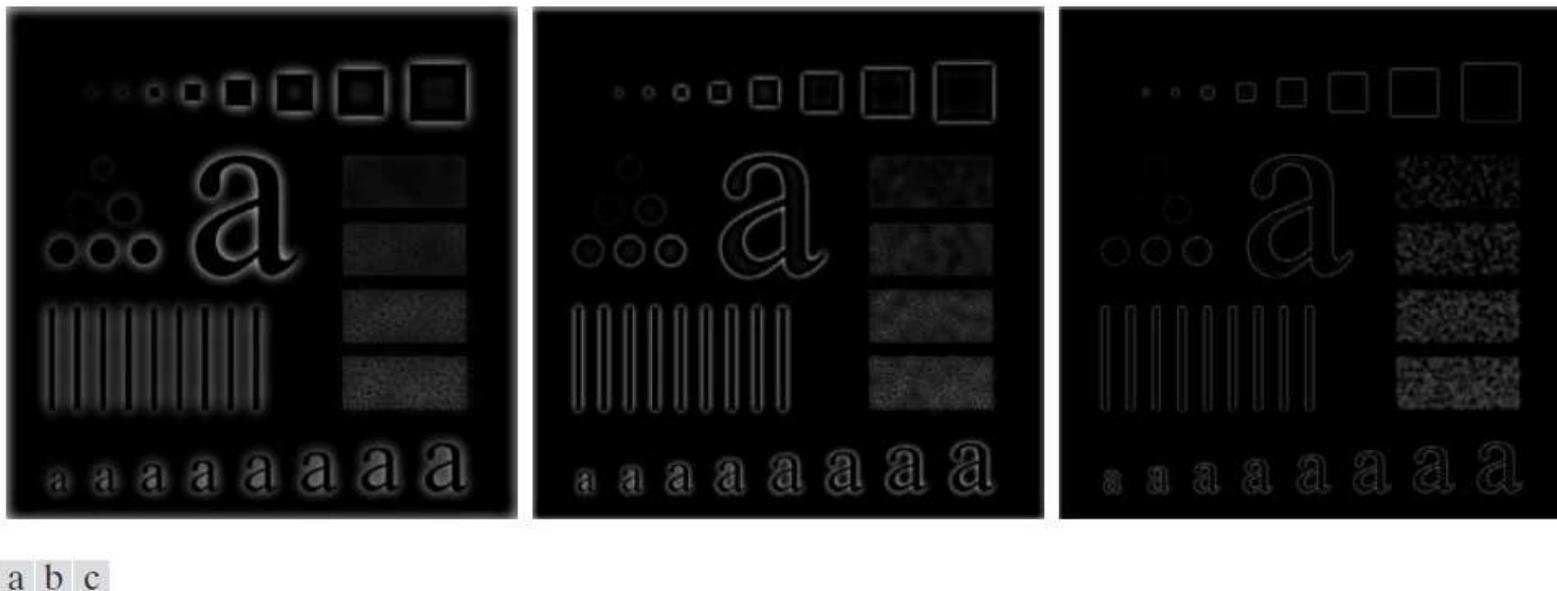
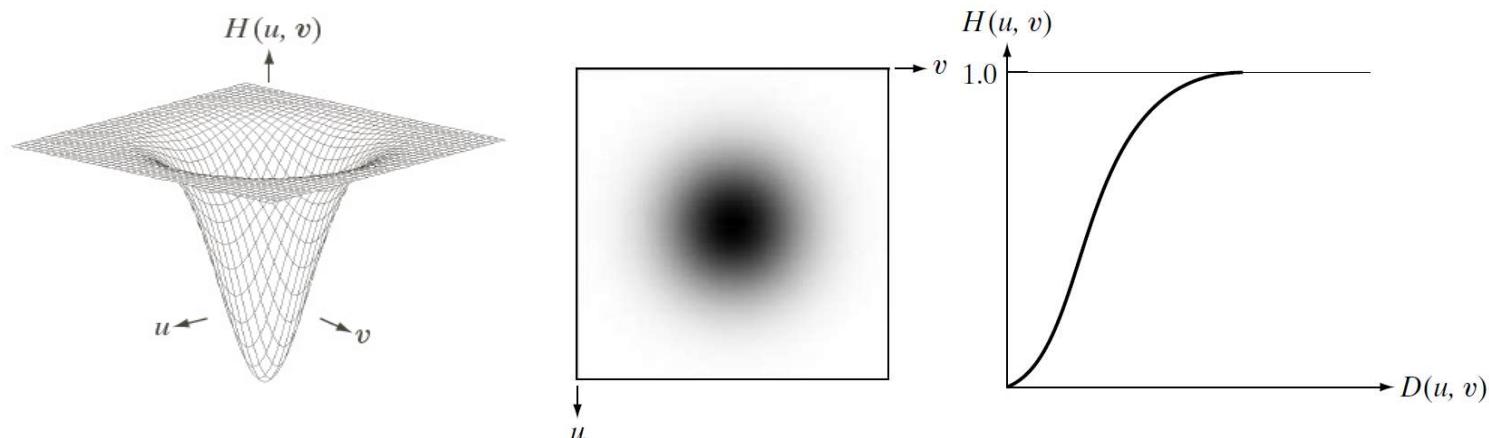


FIGURE 4.55 Results of highpass filtering the image in Fig. 4.41(a) using a BHPF of order 2 with $D_0 = 30, 60$, and 160 , corresponding to the circles in Fig. 4.41(b). These results are much smoother than those obtained with an IHPE.

9. Image Sharpening Using Frequency Domain Filters

- Gaussian Highpass Filters

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



9. Image Sharpening Using Frequency Domain Filters

- Gaussian Highpass Filters

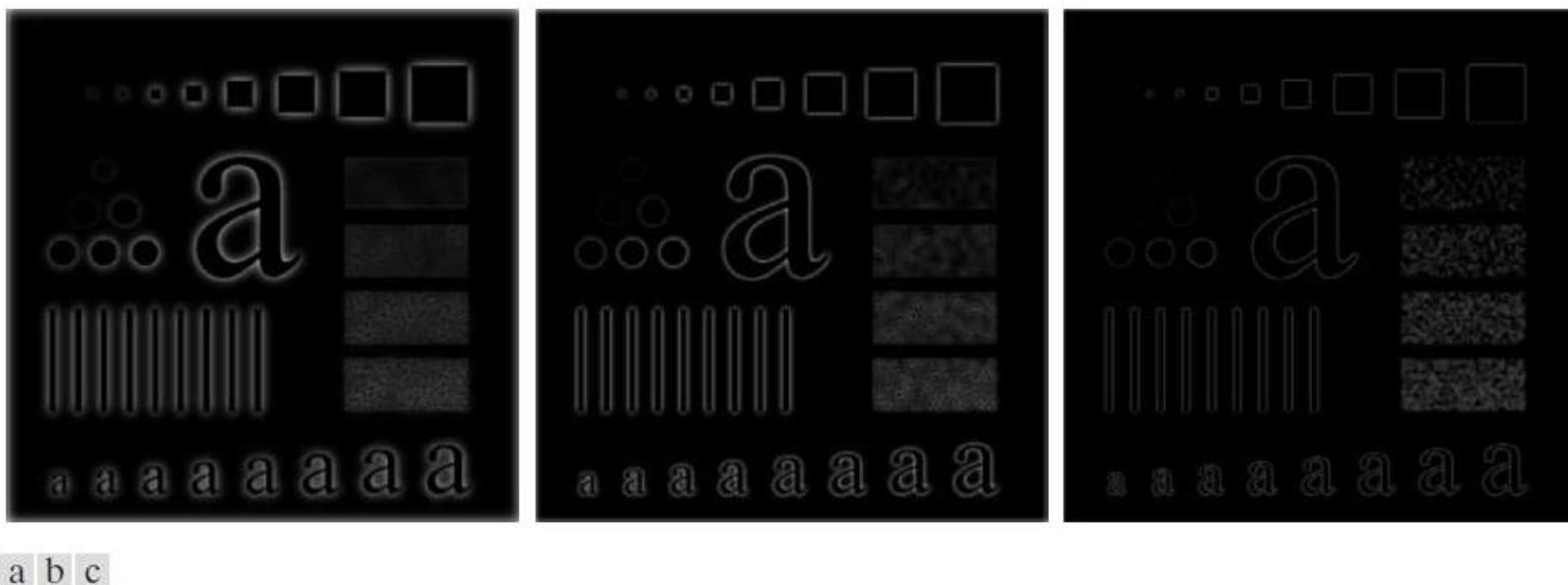


FIGURE 4.56 Results of highpass filtering the image in Fig. 4.41(a) using a GHPF with $D_0 = 30, 60$, and 160 , corresponding to the circles in Fig. 4.41(b). Compare with Figs. 4.54 and 4.55.

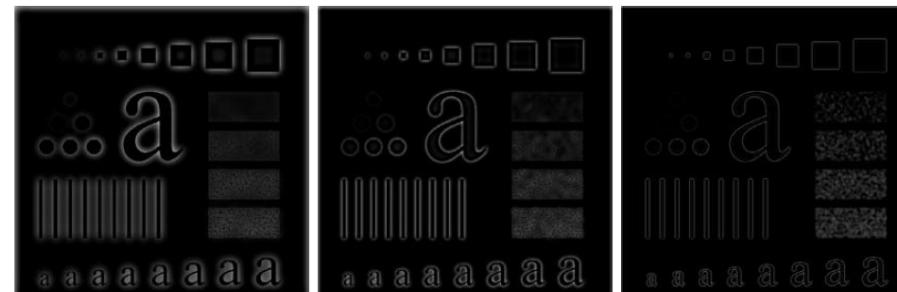
9. Image Sharpening Using Frequency Domain Filters

- Comparison

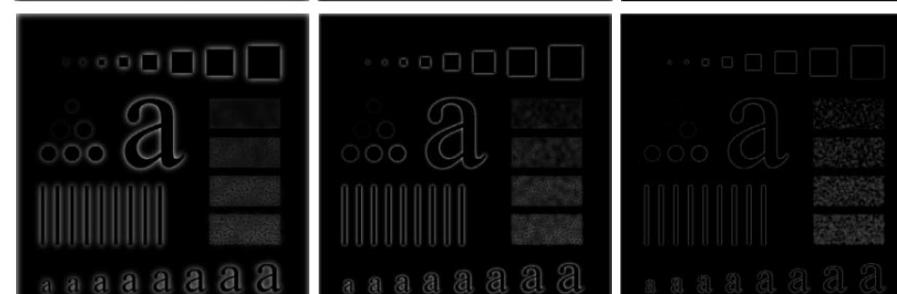
➤ Ideal



➤ Butterworth

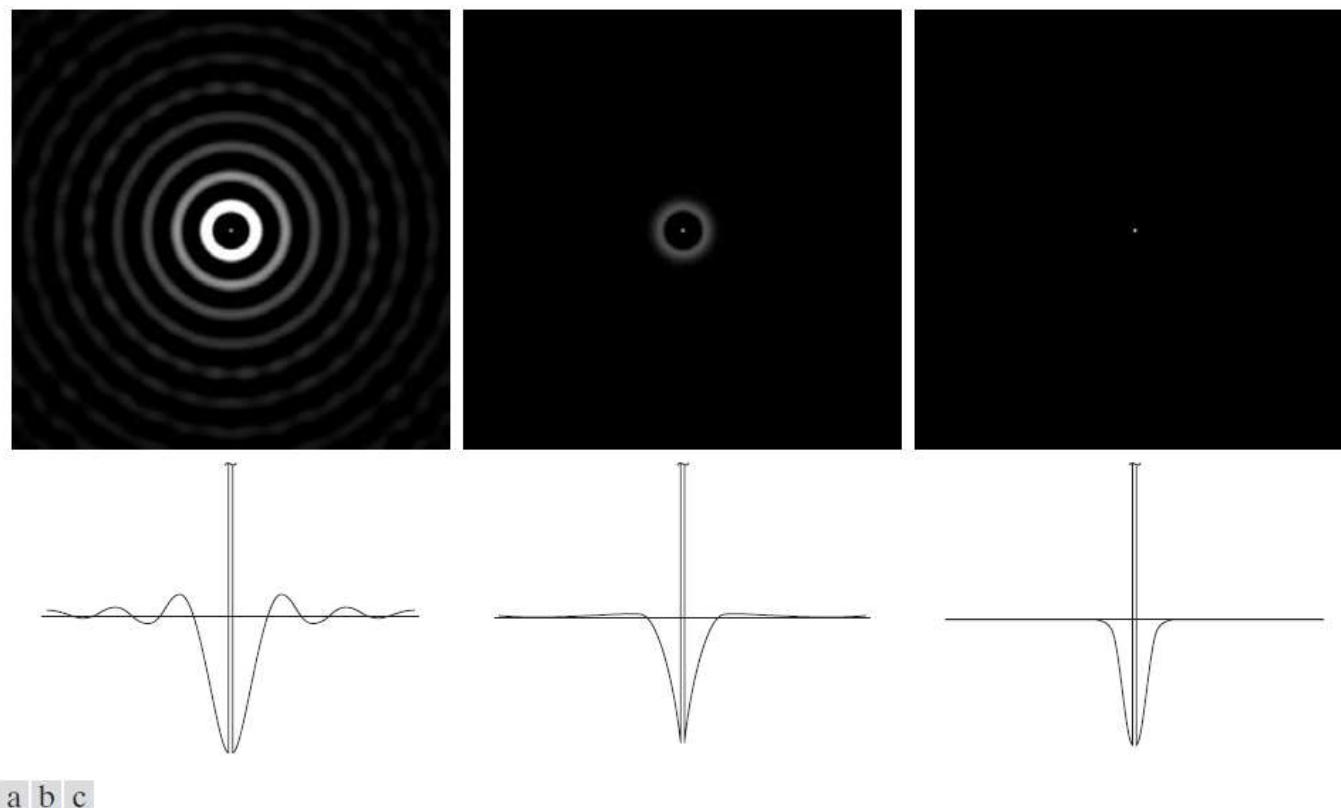


➤ Gaussian



9. Image Sharpening Using Frequency Domain Filters

- Spatial 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.

9. Image Sharpening Using Frequency Domain Filters

- Example



a | b | c

FIGURE 4.57 (a) Thumb print. (b) Result of highpass filtering (a). (c) Result of thresholding (b). (Original image courtesy of the U.S. National Institute of Standards and Technology.)

9. Image Sharpening Using Frequency Domain Filters

- Low/highpass filters equations

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

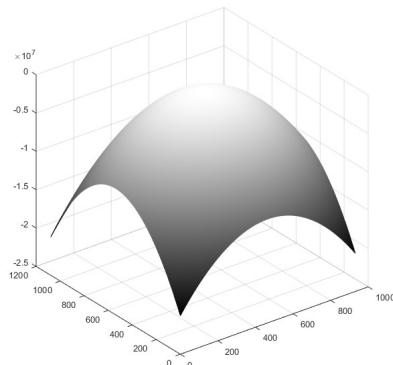
Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$	$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}}$	$H(u, v) = e^{-D^2(u,v)/2D_0^2}$

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

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

9. Image Sharpening Using Frequency Domain Filters

- The **Laplacian** in the Frequency Domain
 - It can be shown that the Laplacian can be implemented in the frequency domain using the filter



$$H(u, v) = -4\pi^2(u^2 + v^2)$$

$$\left(\frac{\partial}{\partial t}\right)^m \left(\frac{\partial}{\partial z}\right)^n f(t, z) \Leftrightarrow (j2\pi\mu)^m (j2\pi\nu)^n F(\mu, \nu)$$

$$\frac{\partial^m f(t, z)}{\partial t^m} \Leftrightarrow (j2\pi\mu)^m F(\mu, \nu); \frac{\partial^n f(t, z)}{\partial z^n} \Leftrightarrow (j2\pi\nu)^n F(\mu, \nu)$$

or, with respect to the center of the frequency rectangle, using the filter

$$\begin{aligned} H(u, v) &= -4\pi^2[(u - P/2)^2 + (v - Q/2)^2] \\ &= -4\pi^2 D^2(u, v) \end{aligned}$$

9. Image Sharpening Using Frequency Domain Filters

- The Laplacian in the Frequency Domain

➤ Then, the Laplacian image is obtained as:

$$\nabla^2 f(x, y) = \mathfrak{J}^{-1}\{H(u, v)F(u, v)\}$$

➤ As explained before, enhancement is achieved using the equation:

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

➤ Here, $c = -1$ because $H(u, v)$ is negative. In the frequency domain, is written as

$$\begin{aligned} g(x, y) &= \mathfrak{J}^{-1}\{F(u, v) - H(u, v)F(u, v)\} \\ &= \mathfrak{J}^{-1}\{[1 - H(u, v)]F(u, v)\} \\ &= \mathfrak{J}^{-1}\{[1 + 4\pi^2 D^2(u, v)]F(u, v)\} \end{aligned}$$

9. Image Sharpening Using Frequency Domain Filters

- MATLAB: s193LaplacianFreq.m



a b

FIGURE 4.58
(a) Original, blurry image.
(b) Image enhanced using the Laplacian in the frequency domain. Compare with Fig. 3.38(e).

9. Image Sharpening Using Frequency Domain Filters

- Unsharp Masking, Highboost Filtering, and High-Frequency-Emphasis Filtering

✓ First a mask is defined $g_{\text{mask}}(x, y) = f(x, y) - f_{\text{LP}}(x, y)$

with $f_{\text{LP}}(x, y) = \mathfrak{F}^{-1}[H_{\text{LP}}(u, v)F(u, v)]$

where $H_{\text{LP}}(u, v)$ is a lowpass filter and $F(u, v)$ is the Fourier transform of $f(x, y)$.

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

✓ This expression defines **unsharp** masking when $k = 1$ and **highboost** filtering when $k > 1$.

9. Image Sharpening Using Frequency Domain Filters

- Unsharp Masking, Highboost Filtering, and High-Frequency-Emphasis Filtering

- ✓ We can express the preceding result entirely in terms of frequency domain computations involving a lowpass filter:

$$g(x, y) = \mathfrak{F}^{-1}\left\{\left[1 + k * [1 - H_{LP}(u, v)]\right]F(u, v)\right\}$$

- ✓ We can express this result in terms of a highpass filter:

$$g(x, y) = \mathfrak{F}^{-1}\left\{\left[1 + k * H_{HP}(u, v)\right]F(u, v)\right\}$$

- ✓ The expression contained within the square brackets is called a **high-frequency emphasis** filter.
 - ✓ The constant, k , gives control over the proportion of high frequencies that influence the final result.

9. Image Sharpening Using Frequency Domain Filters

- Unsharp Masking, Highboost Filtering, and High-Frequency-Emphasis Filtering
 - ✓ A slightly more general formulation of high-frequency-emphasis filtering is the expression

$$g(x, y) = \mathfrak{J}^{-1}\left\{ [k_1 + k_2 * H_{HP}(u, v)]F(u, v)\right\}$$

where $k_1 \geq 0$ gives controls of the offset from the origin and $k_2 \geq 0$ controls the contribution of high frequencies.

9. Image Sharpening Using Frequency Domain Filters

- Unsharp Masking, Highboost Filtering, and High-Frequency-Emphasis Filtering

$$\begin{aligned} k_1 &= 0.5 \\ k_2 &= 0.75 \end{aligned}$$

$$D_0 = 40$$

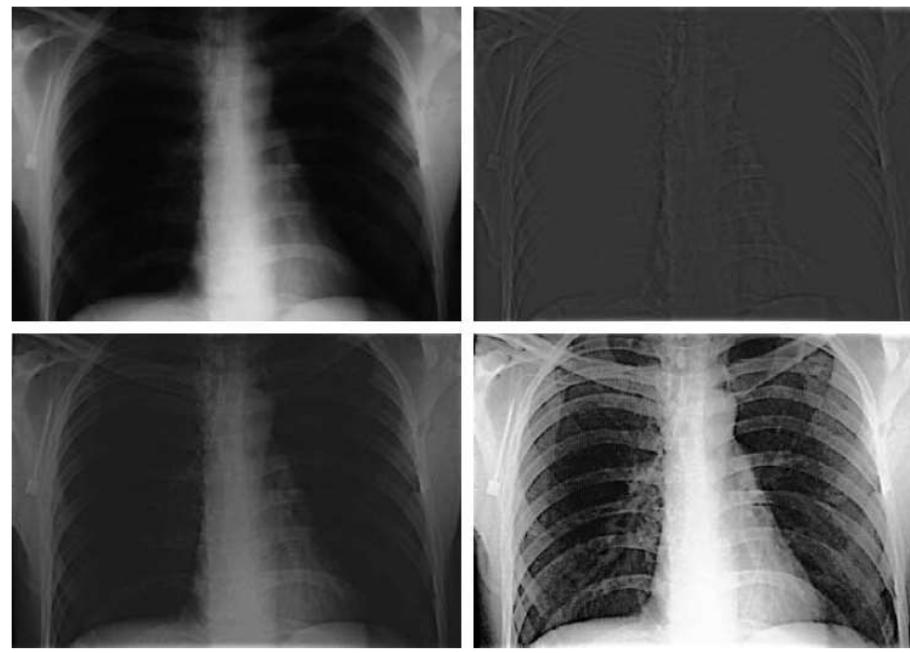
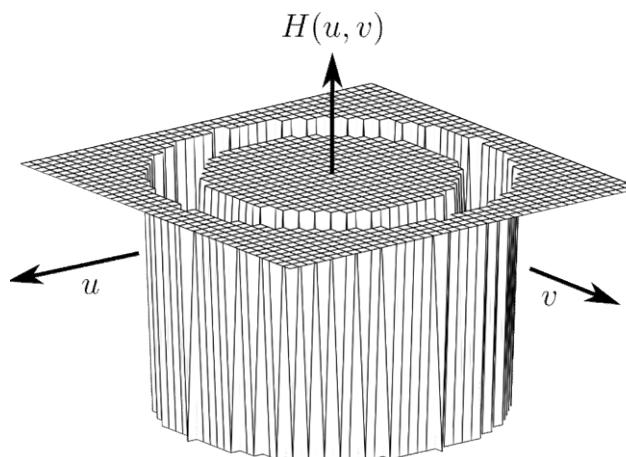


FIGURE 4.59 (a) A chest X-ray image. (b) Result of highpass filtering with a Gaussian filter. (c) Result of high-frequency-emphasis filtering using the same filter. (d) Result of performing histogram equalization on (c). (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

10. Selective Filtering

- Bandreject and Bandpass Filters
 - Ideal Bandreject Filters

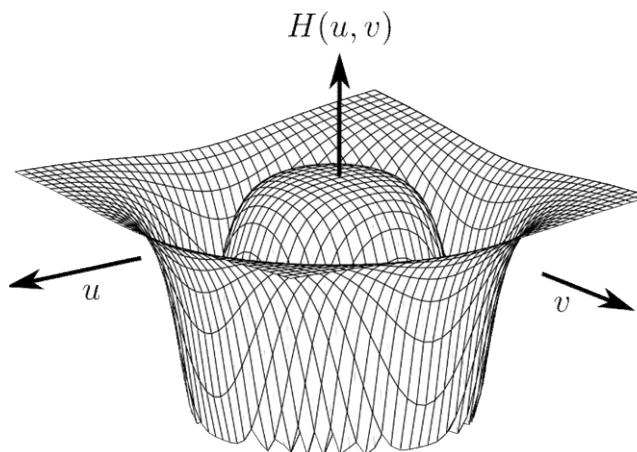
$$H(u, v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$$



10. Selective Filtering

- Bandreject and Bandpass Filters
 - Butterworth Bandreject Filters

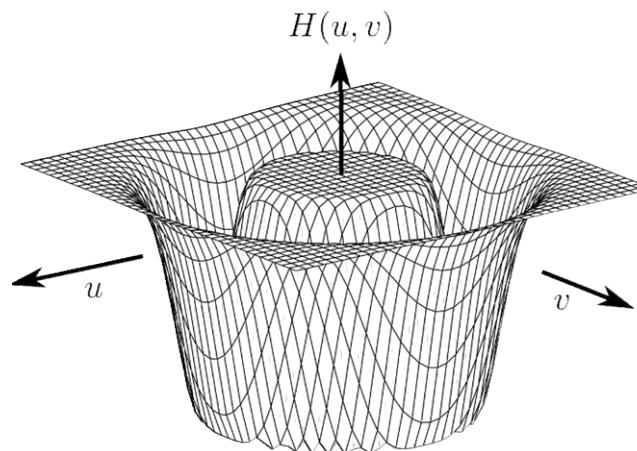
$$H(u, v) = \frac{1}{1 + \left[\frac{DW}{D^2 - D_0^2} \right]^{2n}}$$



10. Selective Filtering

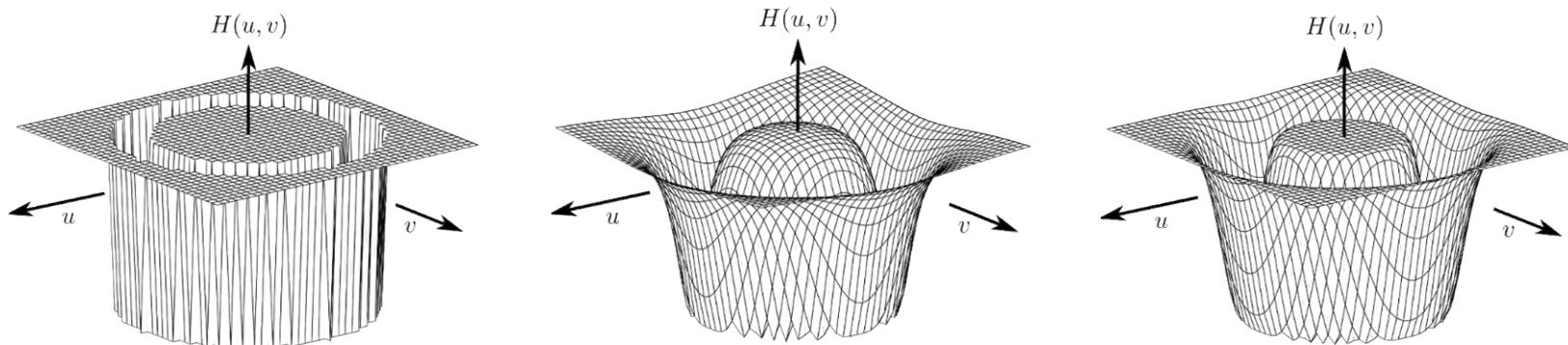
- Bandreject and Bandpass Filters
 - Gaussian Bandreject Filters

$$H(u, v) = 1 - e^{-\left[\frac{D^2 - D_0^2}{DW}\right]^2}$$



10. Selective Filtering

- Bandreject and Bandpass Filters



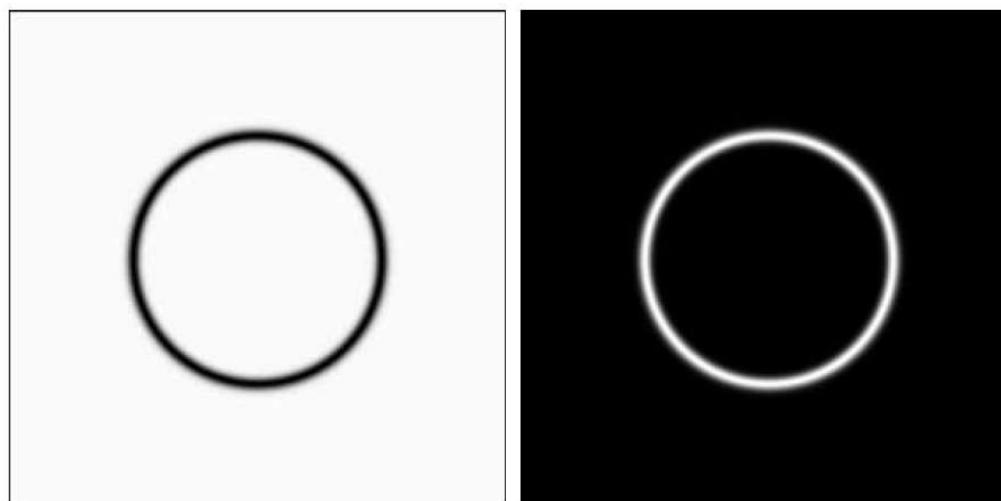
Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u, v) = \frac{1}{1 + \left[\frac{DW}{D^2 - D_0^2} \right]^{2n}}$	$H(u, v) = 1 - e^{-\left[\frac{D^2 - D_0^2}{DW} \right]^2}$

- D is the distance $D(u,v)$ from the center of the filter, D_0 is the radial center of the band, W is the width of the band (upper-lower cutoff frequency) and n is the order of the Butterworth filter.

10. Selective Filtering

- Bandreject and Bandpass Filters
 - A bandpass filter is obtained from a bandreject filter in the same manner that we obtained a highpass filter from a lowpass filter:

$$H_{BP}(u, v) = 1 - H_{BR}(u, v)$$



a b

FIGURE 4.63
(a) Bandreject
Gaussian filter.
(b) Corresponding
bandpass filter.
The thin black
border in (a) was
added for clarity; it
is not part of the
data.

10. Selective Filtering

- MATLAB: s204FilterImplem.m

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

10. Selective Filtering

- Notch Filters

- Notch filters rejects (or passes) frequencies in a predefined neighborhood about the center of the frequency rectangle.
- They are **symmetric about the origin**, so a notch with center at (u_0, v_0) must have a corresponding notch at location $(-u_0, -v_0)$.
- **Notch reject filters** are constructed as products of **highpass filters** whose centers have been **translated to the centers of the notches**. The general form is:

$$H_{\text{NR}}(u, v) = \prod_{k=1}^Q H_k(u, v) H_{-k}(u, v)$$

- where H_k and H_{-k} are highpass filters whose centers are at (u_k, v_k) and $(-u_k, -v_k)$.

10. Selective Filtering

- Notch Filters

- These centers are specified with respect to the center of the frequency rectangle, $(M/2, N/2)$.
- The distance computations for each filter are thus carried out using the expressions

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

and

$$D_{-k}(u, v) = [(u - M/2 + u_k)^2 + (v - N/2 + v_k)^2]^{1/2}$$

- For example, the following is a Butterworth notch reject filter of order n , containing **three notch pairs**

$$H_{\text{NR}}(u, v) = \prod_{k=1}^3 \left[\frac{1}{1 + [D_{0k}/D_k(u, v)]^{2n}} \right] \left[\frac{1}{1 + [D_{0k}/D_{-k}(u, v)]^{2n}} \right]$$

10. Selective Filtering

- Notch Filters

- A notch pass filter is obtained from a notch reject filter using the expression

$$H_{NP}(u, v) = 1 - H_{NR}(u, v)$$

- Example

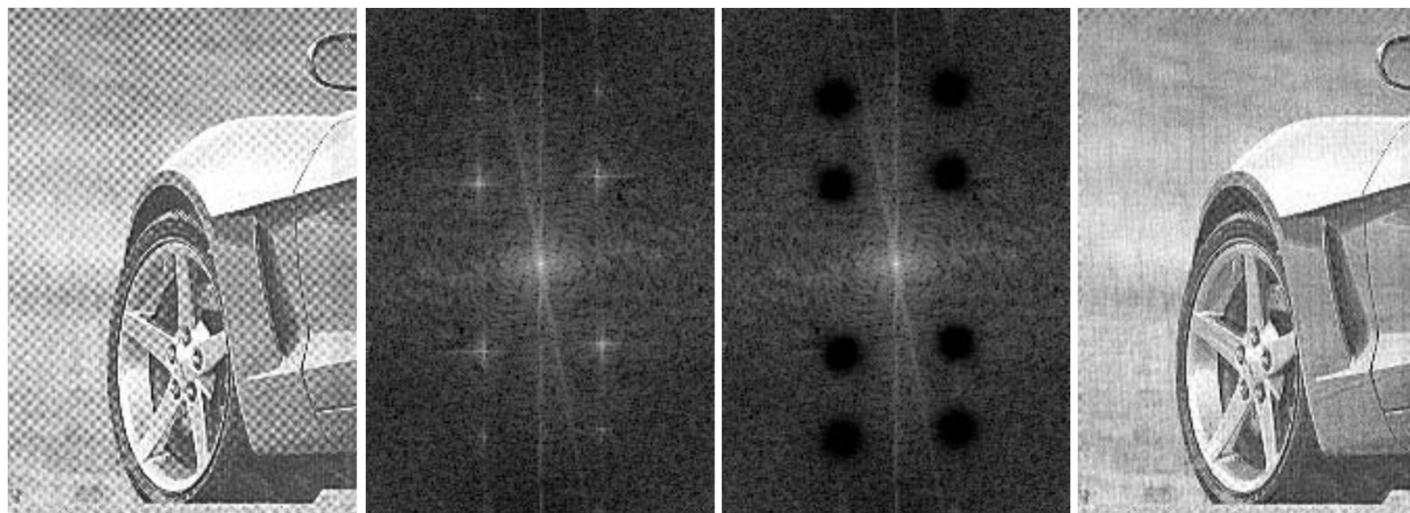


FIGURE 4.64
(a) Sampled newspaper image showing a moiré pattern.
(b) Spectrum.
(c) Butterworth notch reject filter multiplied by the Fourier transform.
(d) Filtered image.

10. Selective Filtering

- MATLAB: s206Notch.m



Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 05

Morphological Image Processing

1. Preliminaries

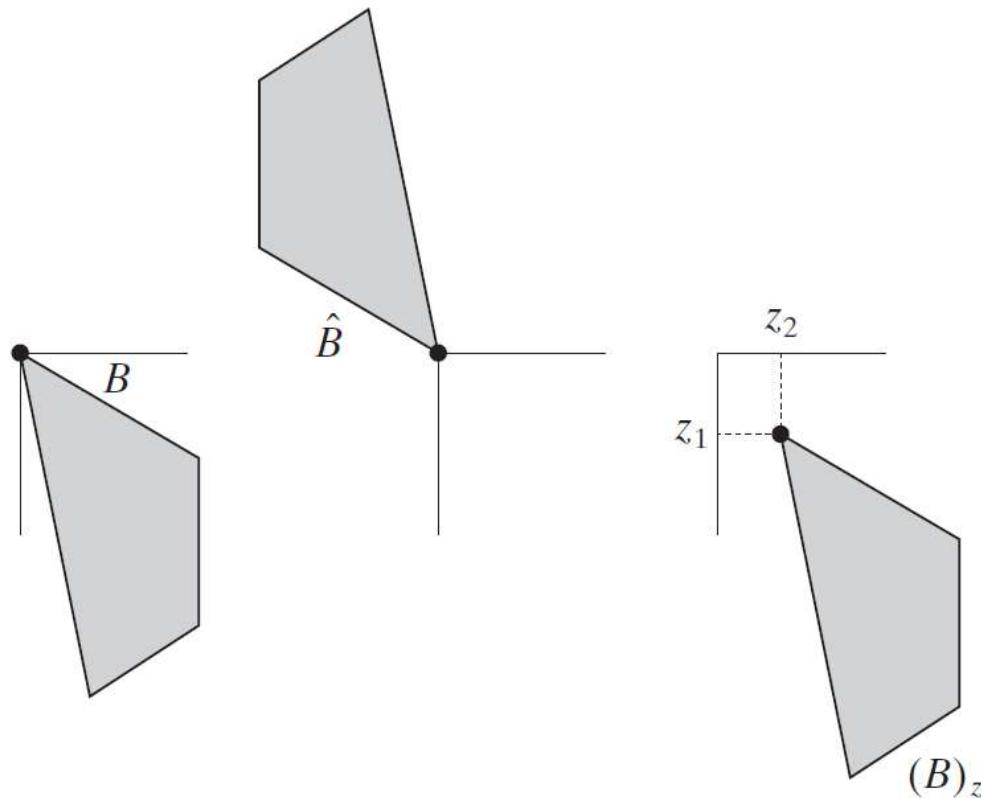
- The language of mathematical morphology is **set theory**.
- **Sets** in mathematical morphology represent **objects in an image**.
- **Binary images**, can be represented as sets whose components are in \mathbb{Z}^2 . Coordinates (x, y) of black (or white) pixels.
- **Grayscale images** can be represented as sets whose components are in \mathbb{Z}^3 (x, y , gray level).
- In this case, two components of each element of the set refer to the coordinates of a pixel, and the third corresponds to its discrete intensity value.

1. Preliminaries

- The concepts of set **reflection** and **translation** are used extensively in morphology.
- Reflection $\hat{B} = \{w|w = -b, \text{ for } b \in B\}$
- If B is the set of pixels (2-D points) representing an object in an image, then \hat{B} is simply the set of points in whose coordinates have been replaced by $(-x, -y)$.
- The translation $(B)_z = \{c|c = b + z, \text{ for } b \in B\}$
- If B is the set of pixels representing an object in an image, then $(B)_z$ is the set of points in whose coordinates have been replaced by $(x+z_1, y+z_2)$.

1. Preliminaries

- Reflection and translation



a b c

FIGURE 9.1

(a) A set, (b) its reflection, and (c) its translation by z .

1. Preliminaries

- **Structuring elements** (SEs) are small sets or subimages used to probe an image for properties of interest.

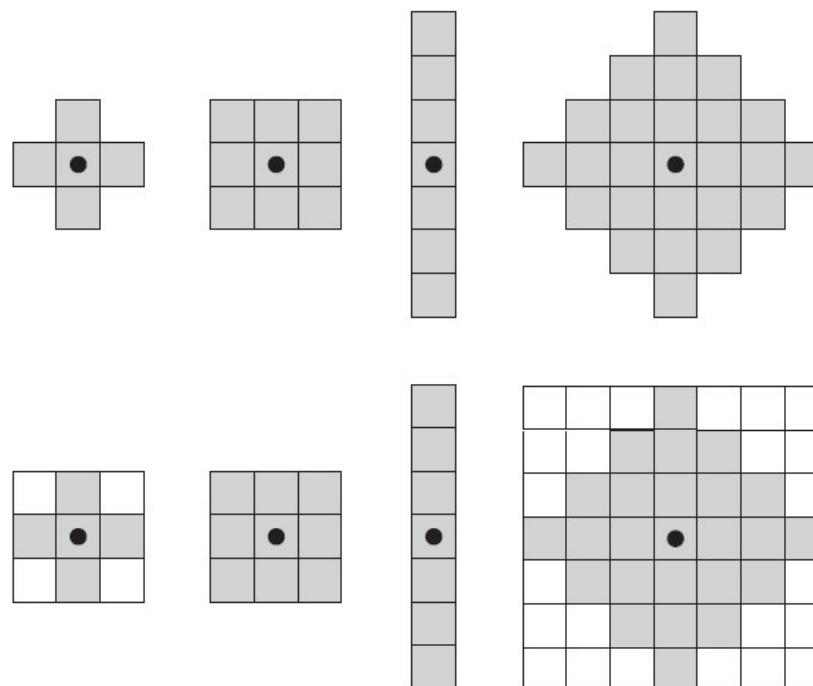


FIGURE 9.2 First row: Examples of structuring elements. Second row: Structuring elements converted to rectangular arrays. The dots denote the centers of the SEs.

1. Preliminaries

- **Structuring elements** (SEs) are small sets or subimages used to probe an image for properties of interest.

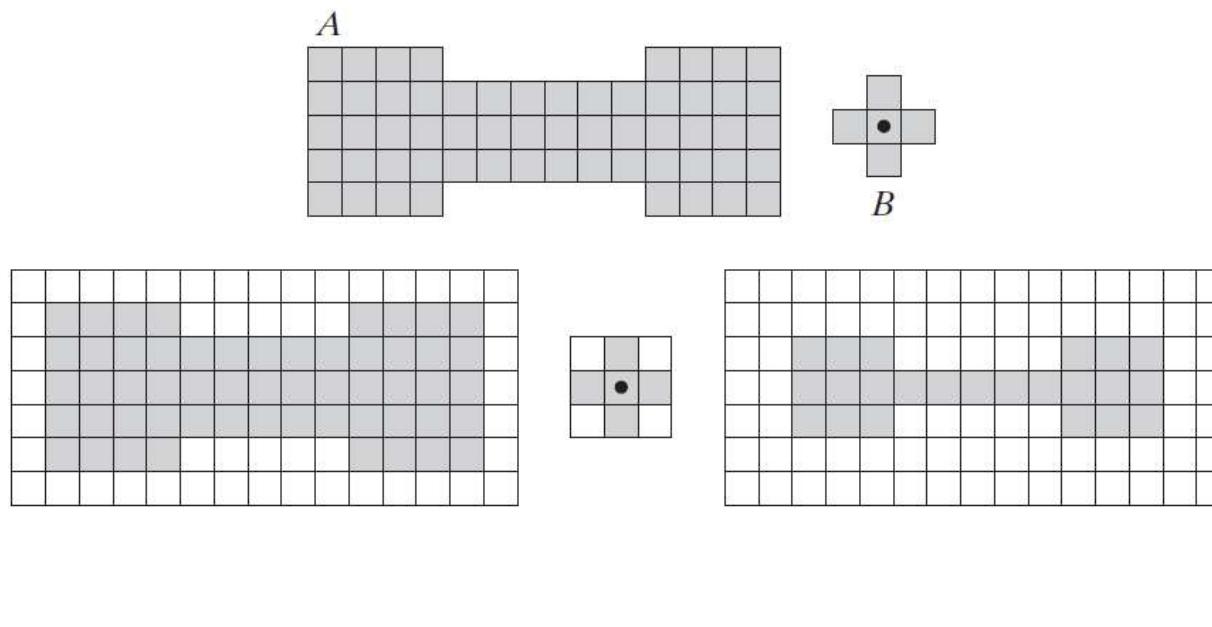
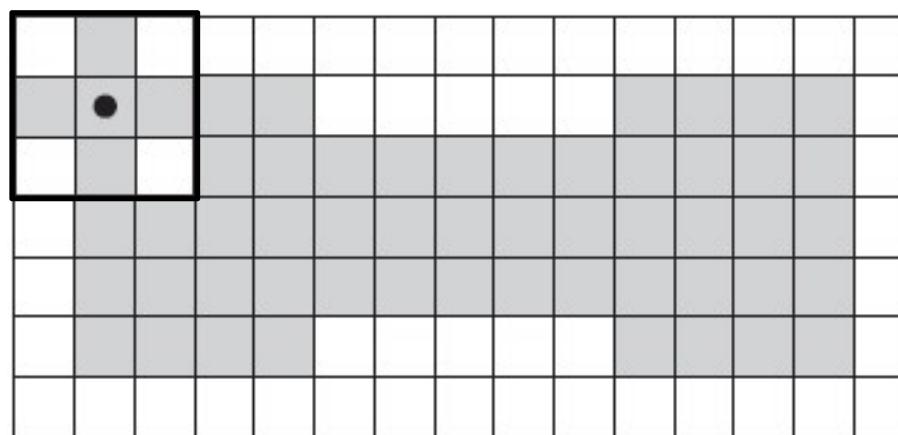


FIGURE 9.3 (a) A set (each shaded square is a member of the set). (b) A structuring element. (c) The set padded with background elements to form a rectangular array and provide a background border. (d) Structuring element as a rectangular array. (e) Set processed by the structuring element.

1. Preliminaries

- Example

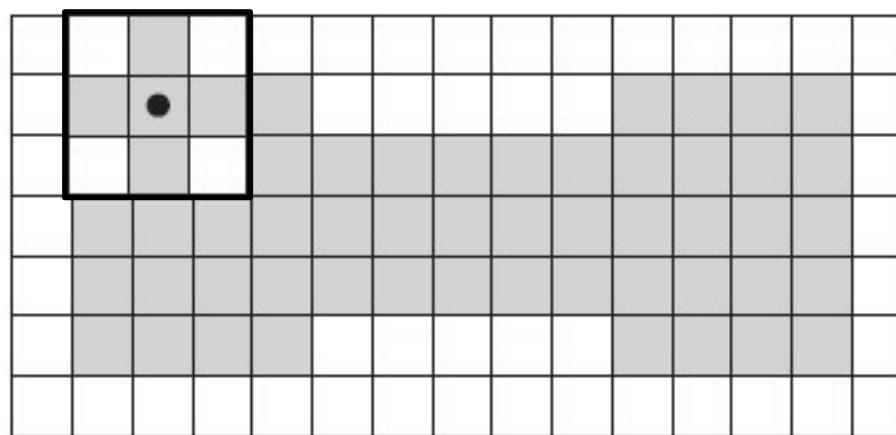
- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



1. Preliminaries

- Example

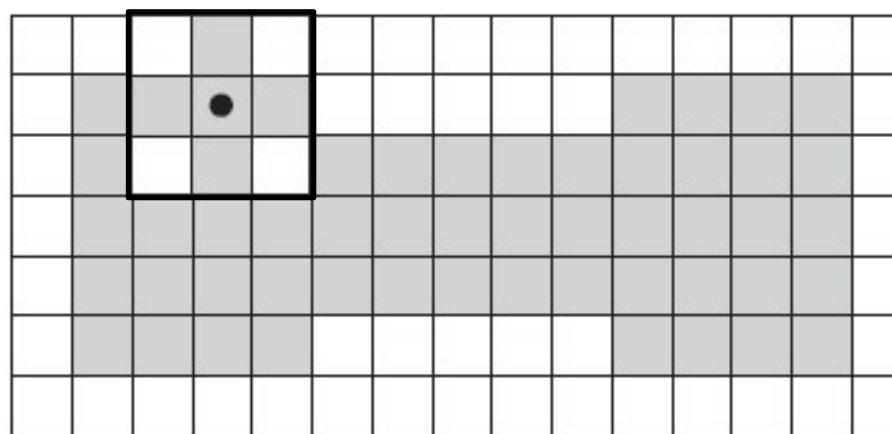
- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



1. Preliminaries

- Example

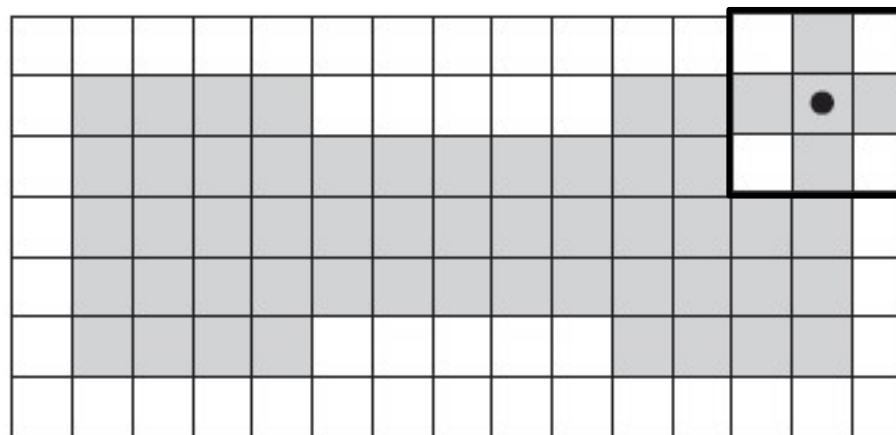
- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



1. Preliminaries

- Example

- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



1. Preliminaries

- Example

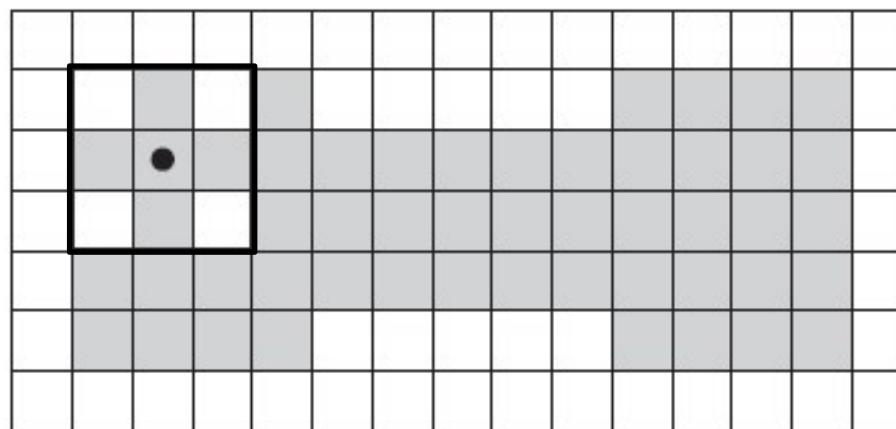
- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



1. Preliminaries

- Example

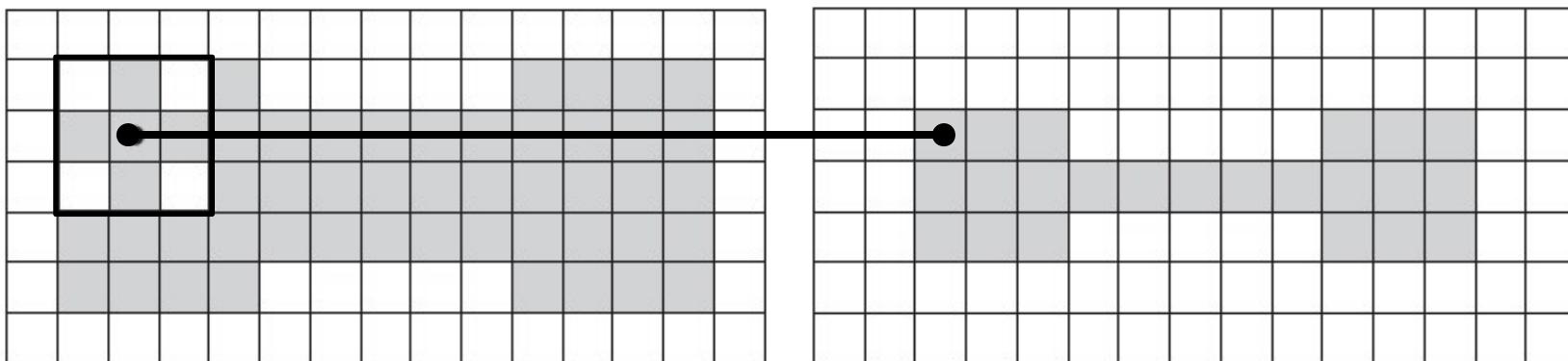
- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



1. Preliminaries

- Example

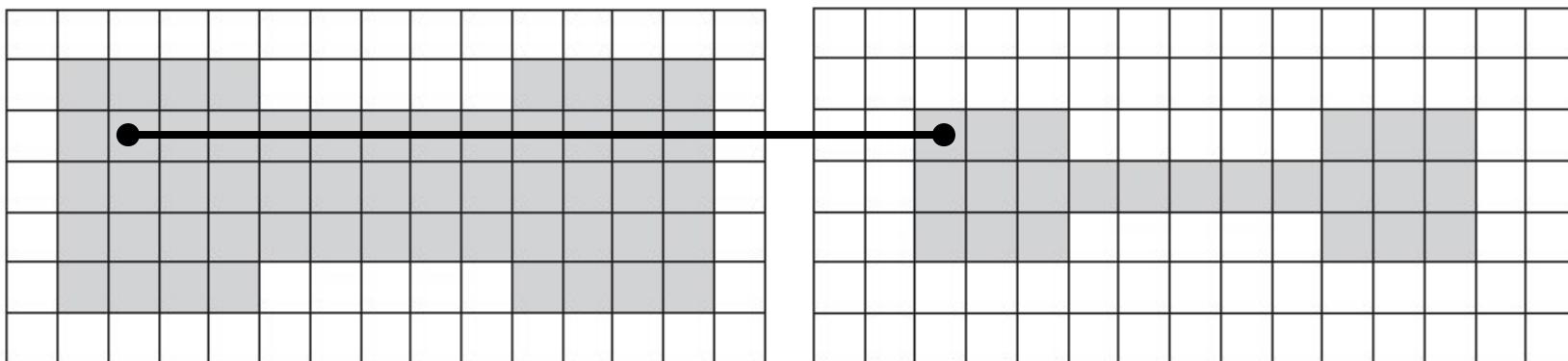
- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



1. Preliminaries

- Example

- Create a new set by running B over A so that the origin of B visits every element of A. At each location of the origin of B, if B is completely contained in A, mark that location as a member of the new set (shown shaded); else mark it as not being a member of the new set (shown not shaded).



2. Erosion and Dilation

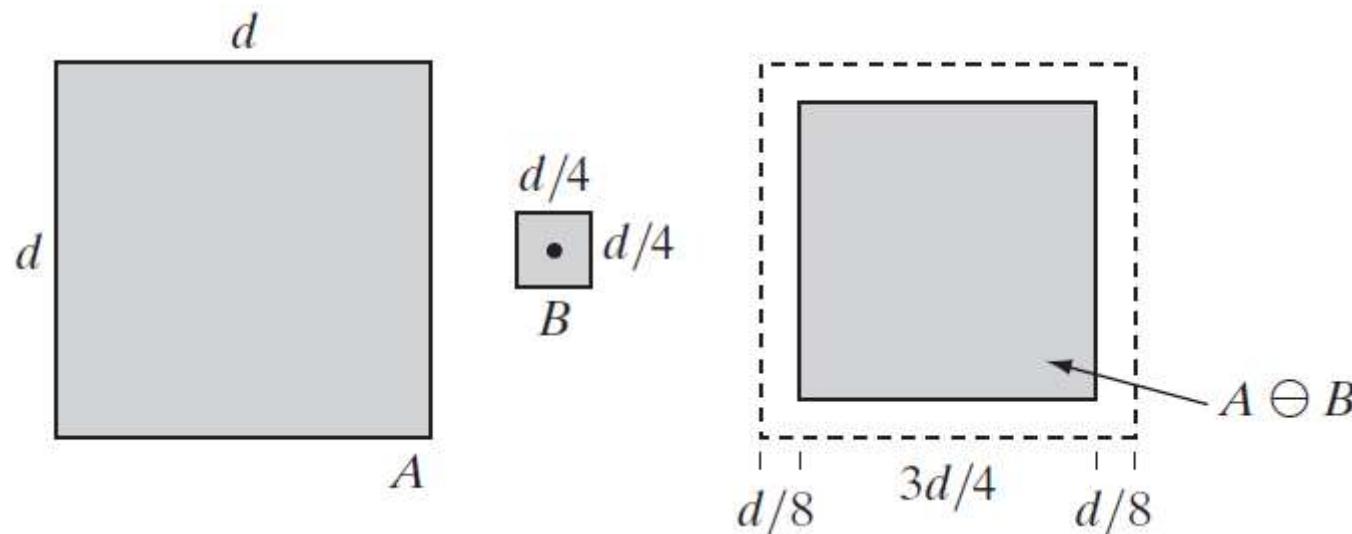
- **Erosion** and **dilation** are fundamental to morphological processing.
- In fact, many of the morphological algorithms discussed here are based on these two primitive operations.

2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **erosion** of A by B is defined as

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

- This equation indicates that the erosion of A by B is the set of all points z such that B translated by z is contained in A .

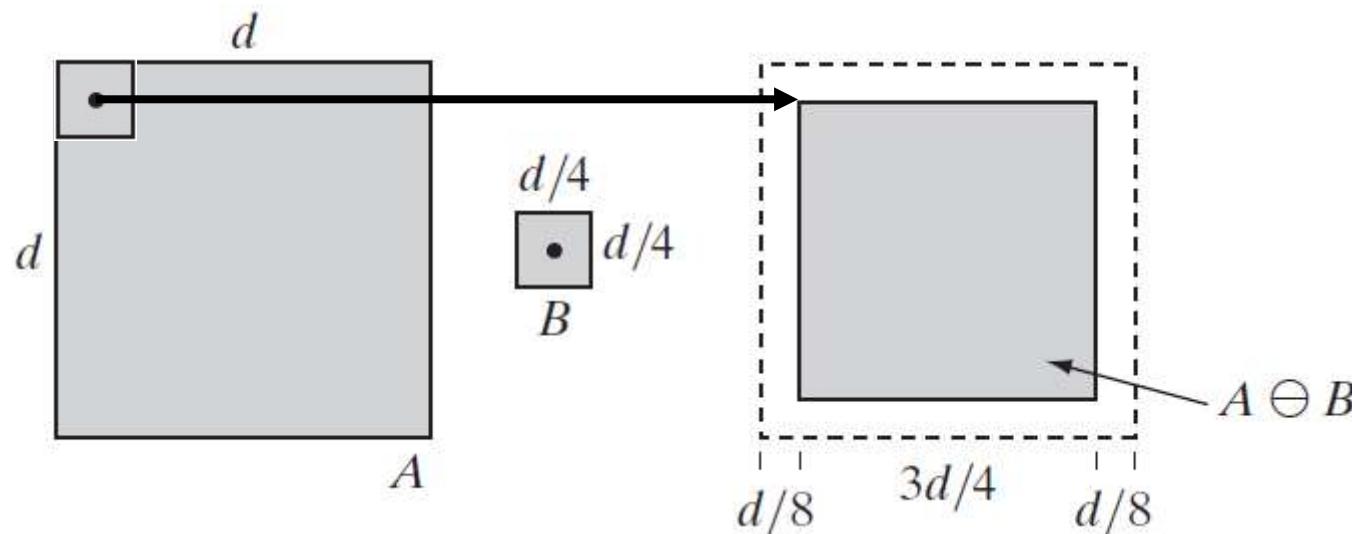


2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **erosion** of A by B is defined as

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

- This equation indicates that the erosion of A by B is the set of all points z such that B translated by z is contained in A .

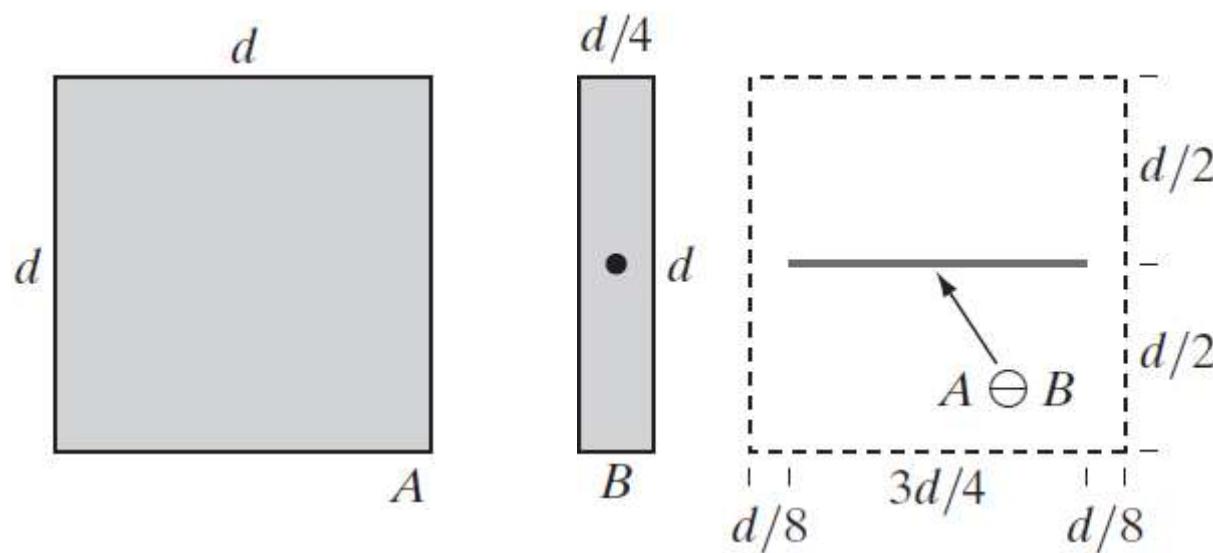


2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **erosion** of A by B is defined as

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

- This equation indicates that the erosion of A by B is the set of all points z such that B translated by z is contained in A .

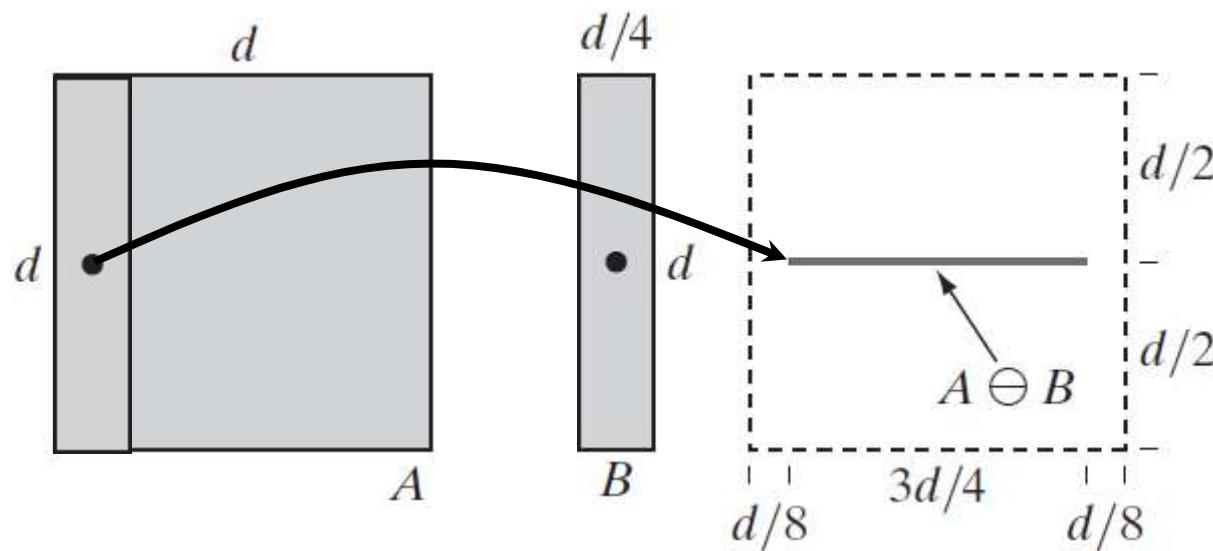


2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **erosion** of A by B is defined as

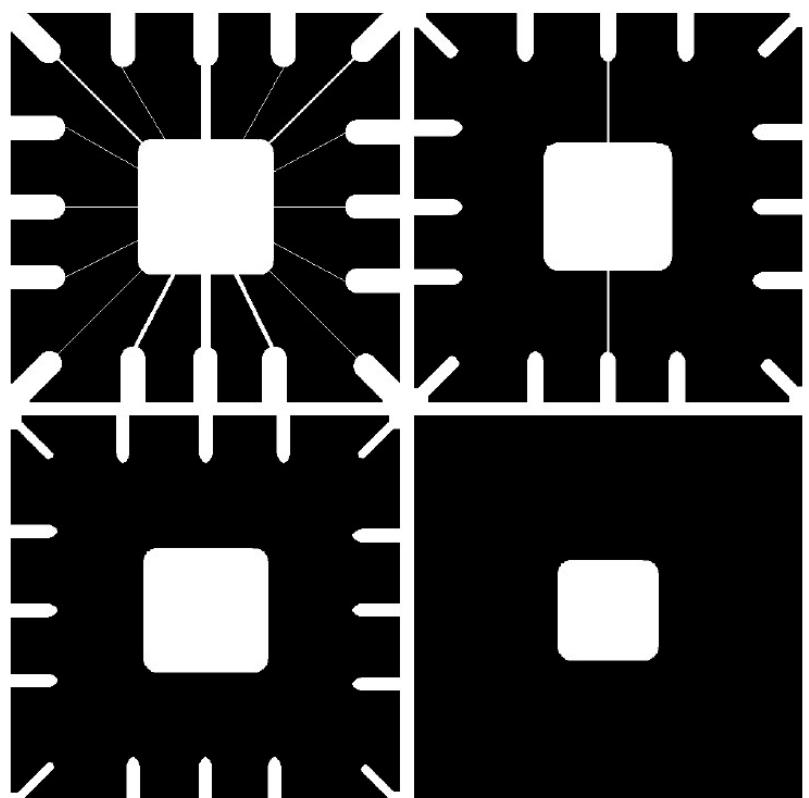
$$A \ominus B = \{z | (B)_z \subseteq A\}$$

- This equation indicates that the erosion of A by B is the set of all points z such that B translated by z is contained in A .



2. Erosion and Dilation

- Suppose that we wish to remove the lines connecting the center region to the border pads.

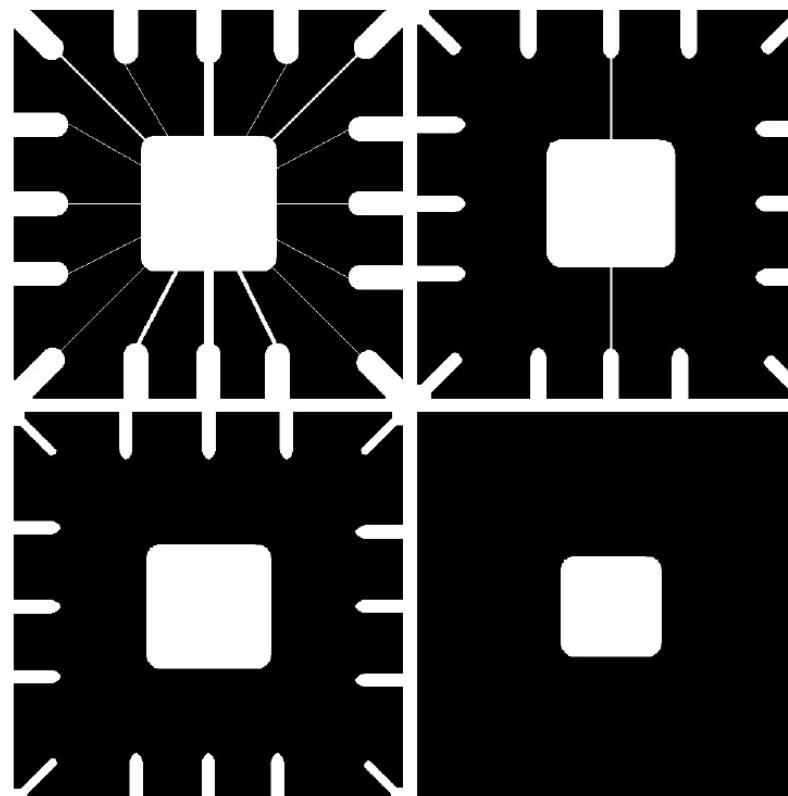


a	b
c	d

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.

2. Erosion and Dilation

- MATLAB: s22Erosion.m



2. Erosion and Dilation

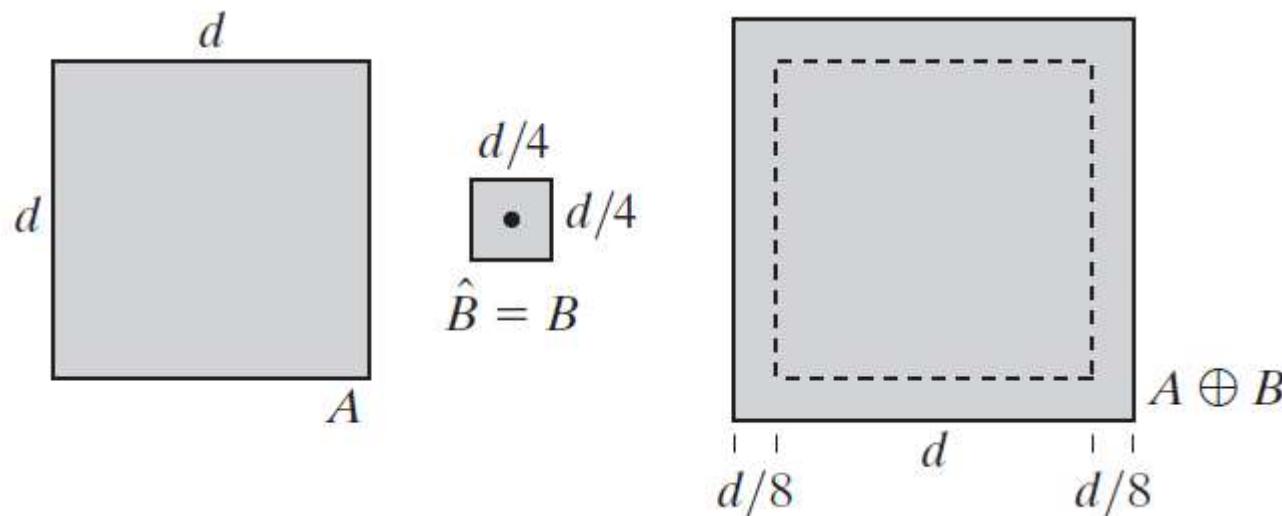
- We see from previous examples that **erosion shrinks** or **thins** objects in a binary image.
- In fact, we can view **erosion** as a morphological **filtering operation**.
 - Details smaller than the structuring element are filtered (removed) from the image.
- In the last example, erosion performed the function of a “line filter.”

2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **dilation** of A by B is defined as

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

- The dilation of A by B then is the set of all displacements z , such that A and \hat{B} overlap by at least one element.

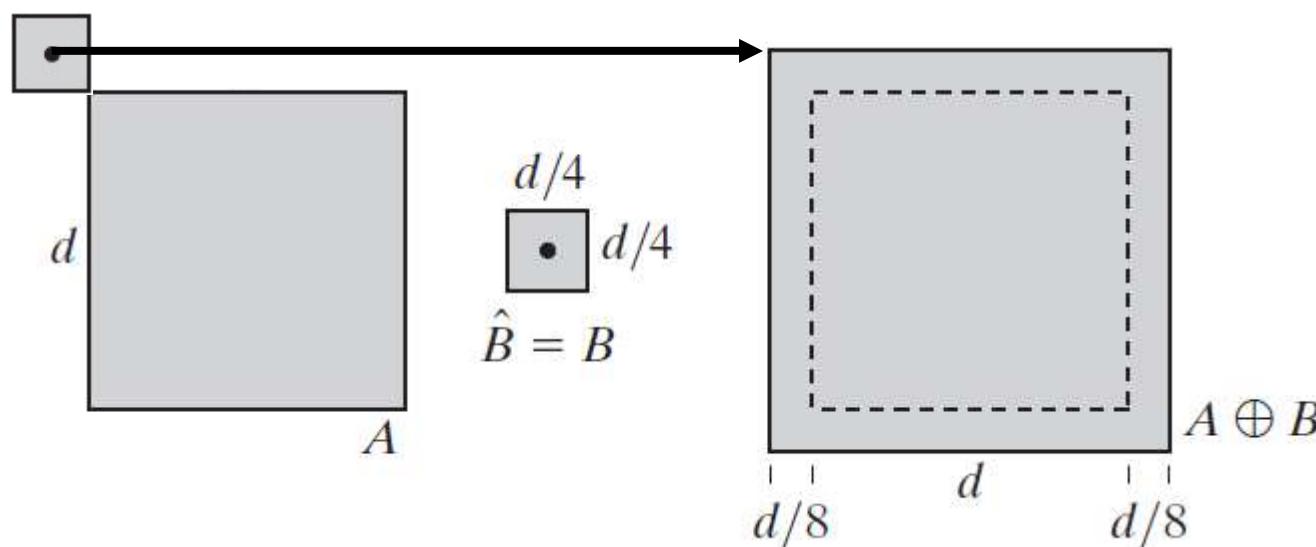


2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **dilation** of A by B is defined as

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

- The dilation of A by B then is the set of all displacements z , such that A and \hat{B} overlap by at least one element.

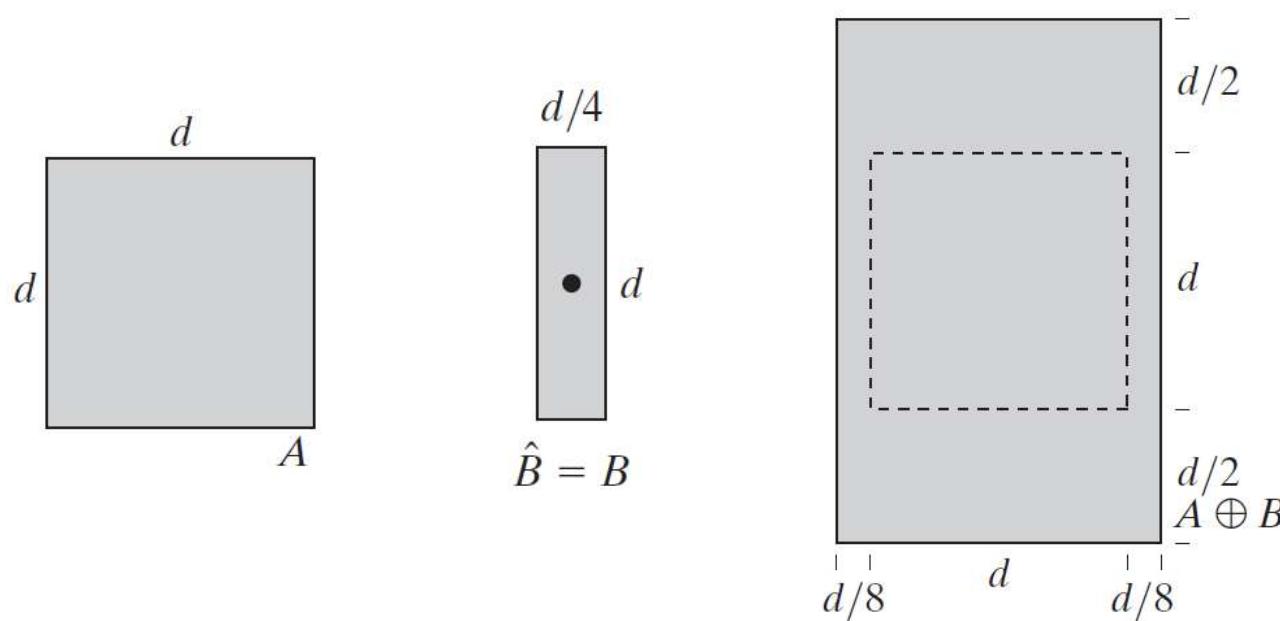


2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **dilation** of A by B is defined as

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

- The dilation of A by B then is the set of all displacements z , such that A and \hat{B} overlap by at least one element.

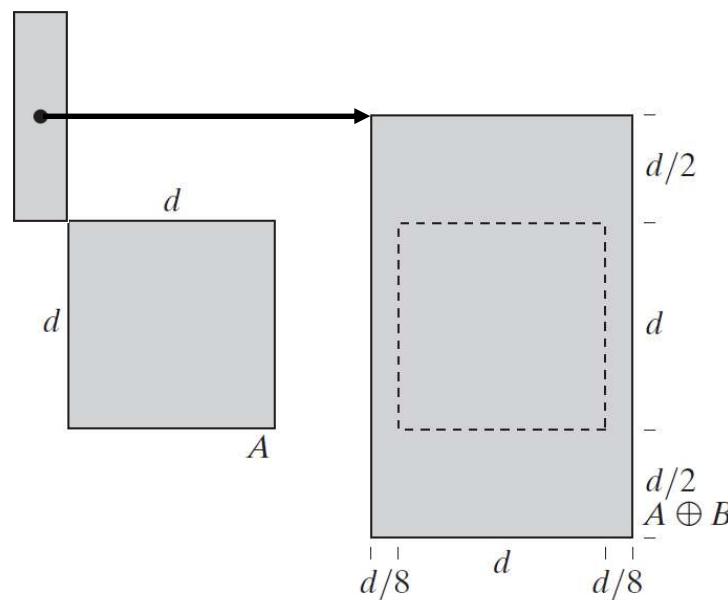


2. Erosion and Dilation

- With A and B as sets in \mathbb{Z}^2 , the **dilation** of A by B is defined as

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

- The dilation of A by B then is the set of all displacements z , such that A and \hat{B} overlap by at least one element.



2. Erosion and Dilation

- One of the simplest applications of dilation is for 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.



0	1	0
1	1	1
0	1	0

a b c

FIGURE 9.7

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

2. Erosion and Dilation

- MATLAB: s29Dilation.m

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.



0	1	0
1	1	1
0	1	0

2. Erosion and Dilation

- Duality
 - Erosion and dilation are duals of each other with respect to set complementation and reflection.

$$(A \ominus B)^c = A^c \oplus \hat{B} \quad (A \oplus B)^c = A^c \ominus \hat{B}$$

- The duality property is useful particularly when the structuring element is symmetric with respect to its origin
- We can obtain the erosion of an image by simply by dilating its background with the same structuring element and complementing the result.
- The proof of this result is left as an exercise.

2. Erosion and Dilation

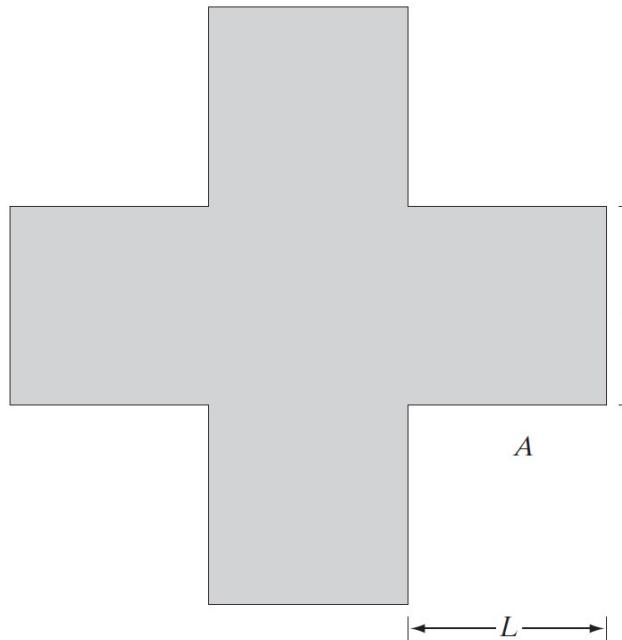
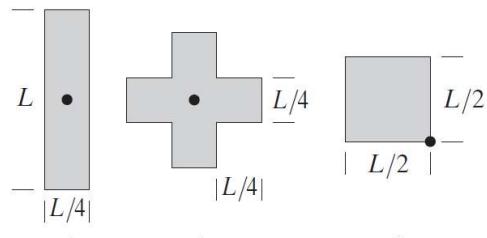
- Exercise

Let A denote the set shown shaded in the following figure. Refer to the structuring elements shown (the black dots denote the origin). Sketch the result of the following morphological operations:

(a) $(A \ominus B^1) \oplus B^3$

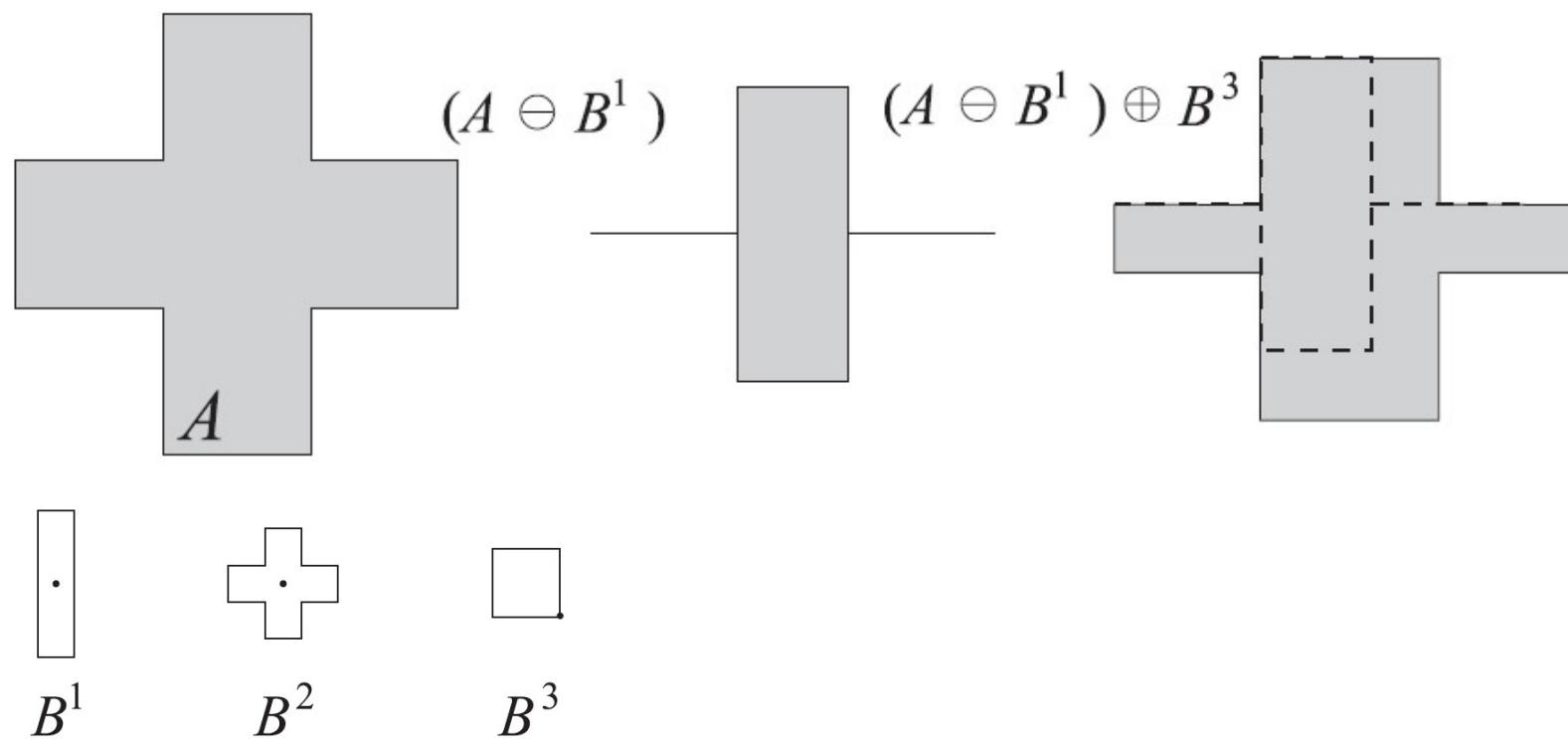
(b) $(A \oplus B^1) \ominus B^3$

(c) $(A \oplus B^3) \ominus B^2$



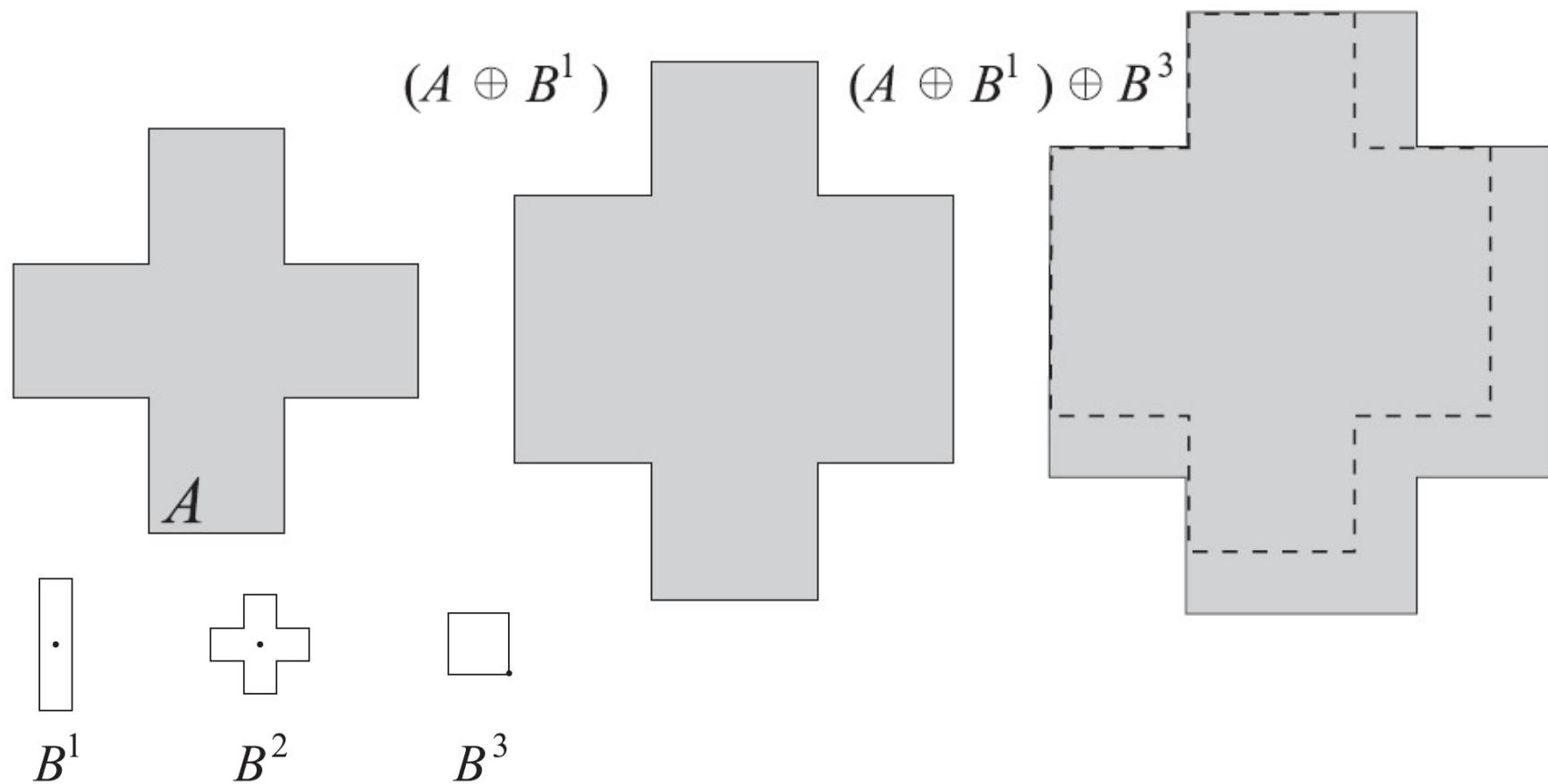
2. Erosion and Dilation

- Exercise



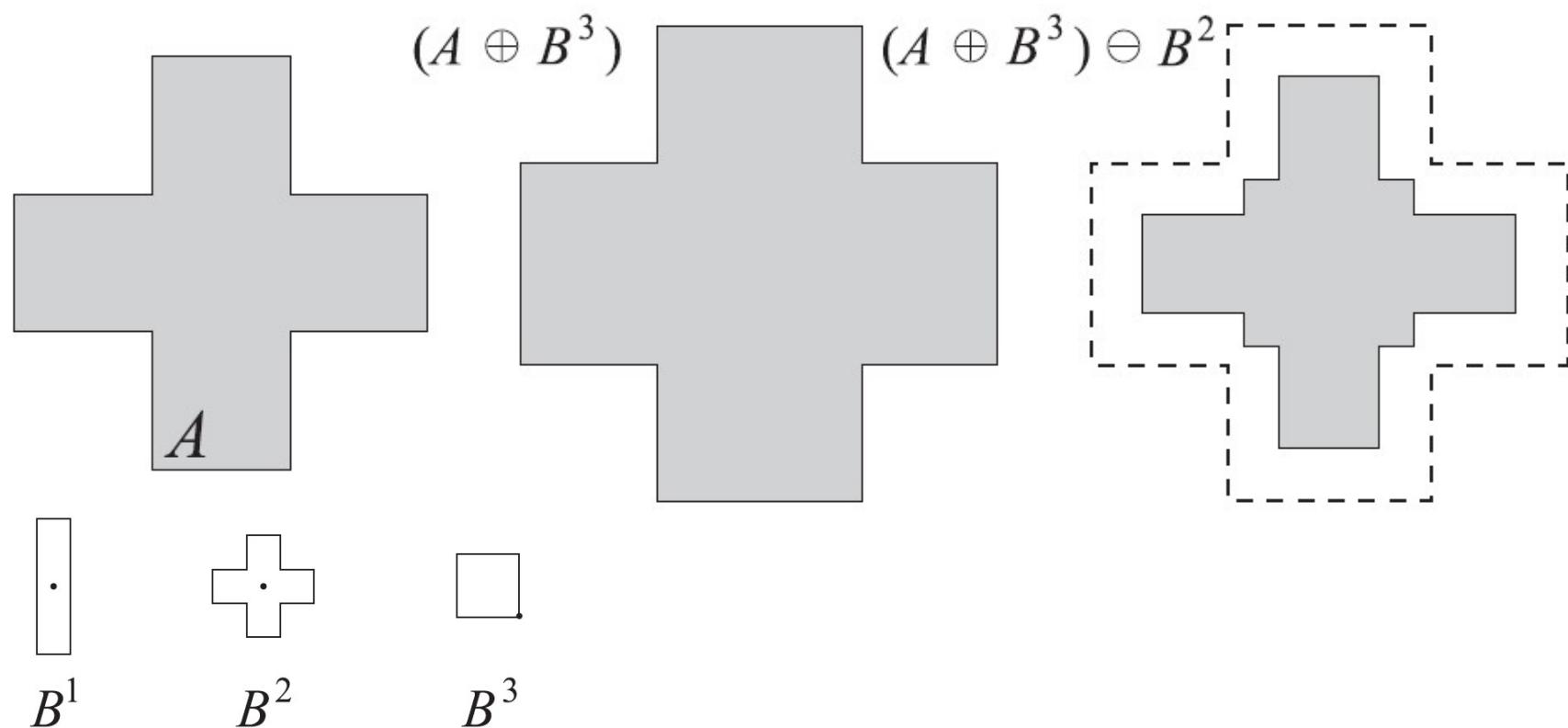
2. Erosion and Dilation

- Exercise



2. Erosion and Dilation

- Exercise



3. Opening and Closing

- As you have seen, dilation expands the components of an image and erosion shrinks them.
- Opening generally:
 - smoothes the contour of an object
 - breaks narrow isthmuses; and
 - eliminates thin protrusions.
- Closing generally:
 - fuses narrow breaks and long thin gulfs;
 - eliminates small holes, and
 - Fills gaps in the contour.

3. Opening and Closing

- The opening of set A by structuring element B is defined as

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

- Thus, the opening of A by B is the erosion of A by B followed by a dilation of the result by B.
- Similarly, the closing of set A by structuring element B is defined as

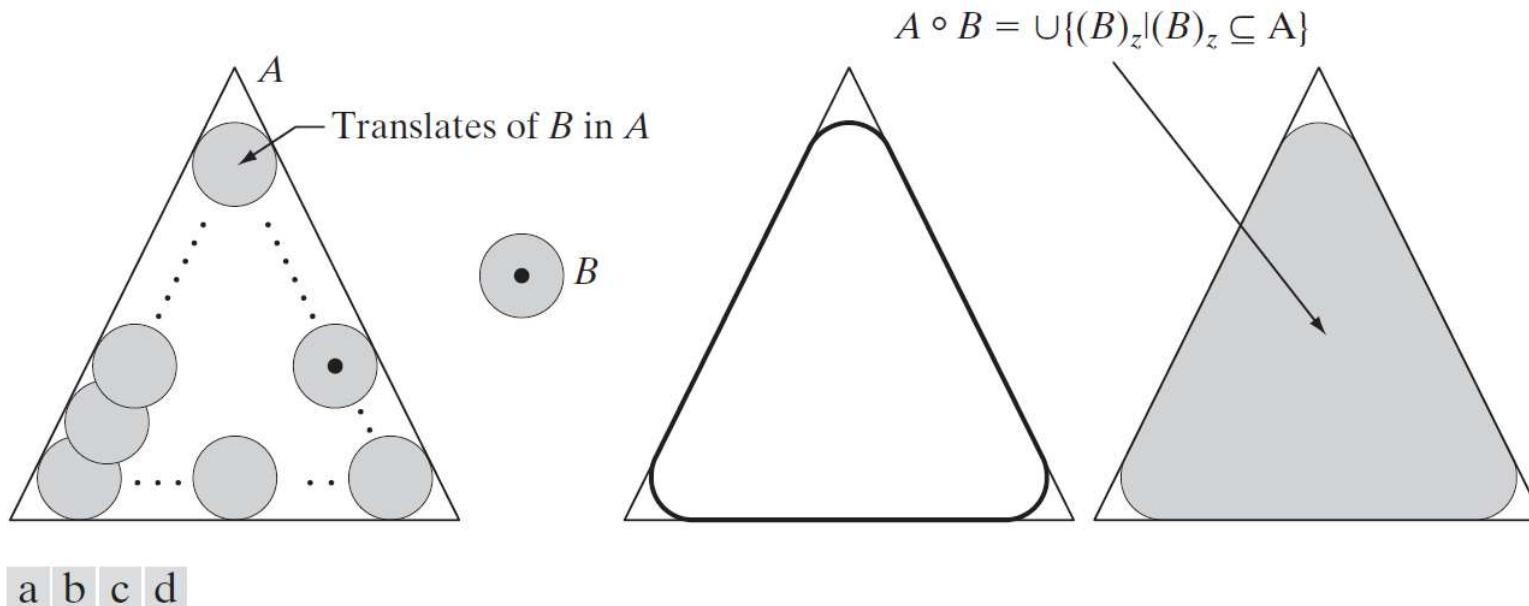
$$A \bullet B = (A \oplus B) \ominus B$$

- The closing of A by B is simply the dilation of A by B followed by the erosion of the result by B.

3. Opening and Closing

- Geometric interpretation

➤ Opening



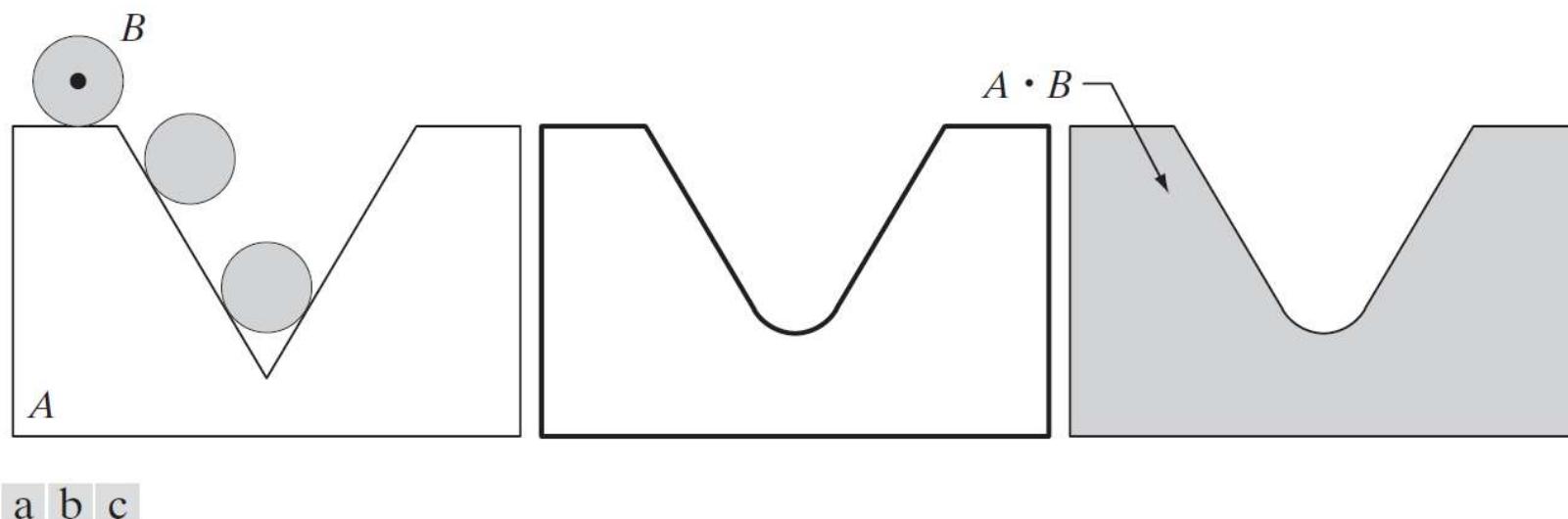
a b c d

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

3. Opening and Closing

- Geometric interpretation

➤ Closing

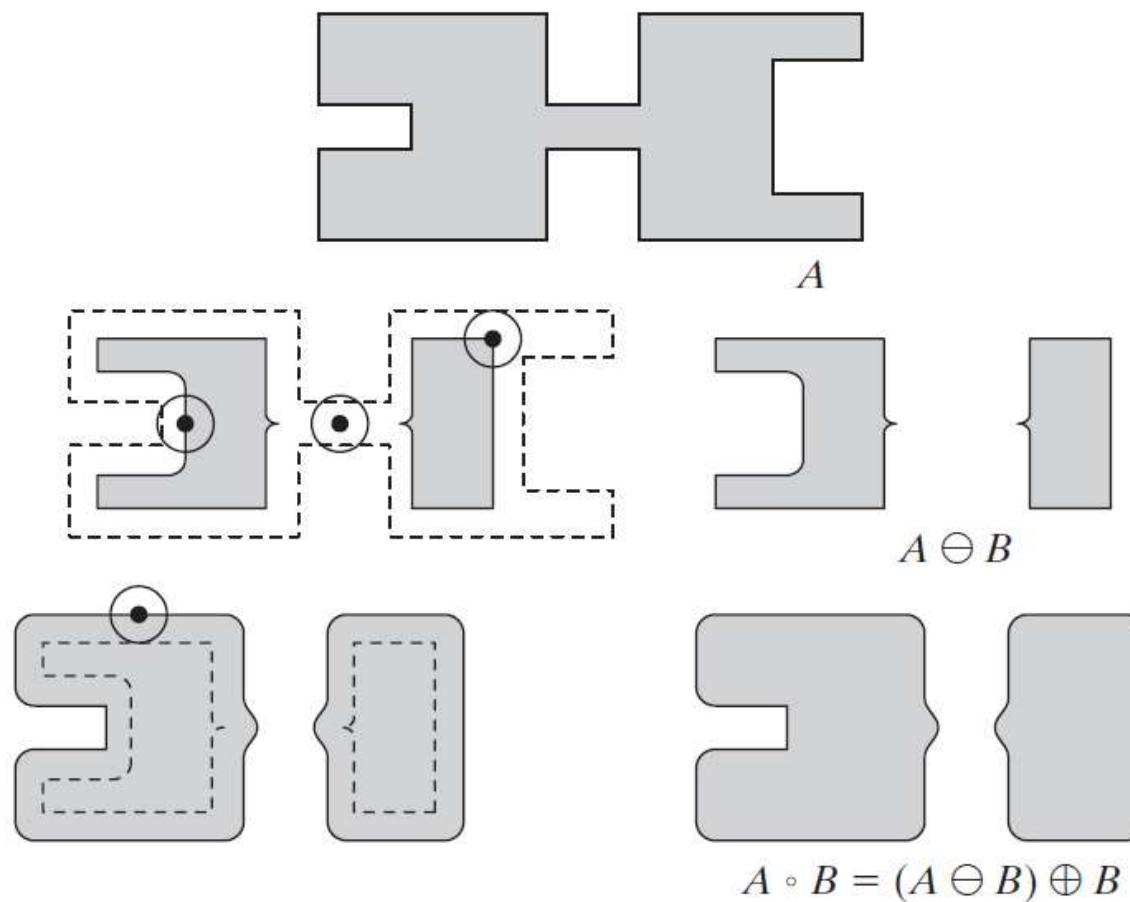


a b c

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

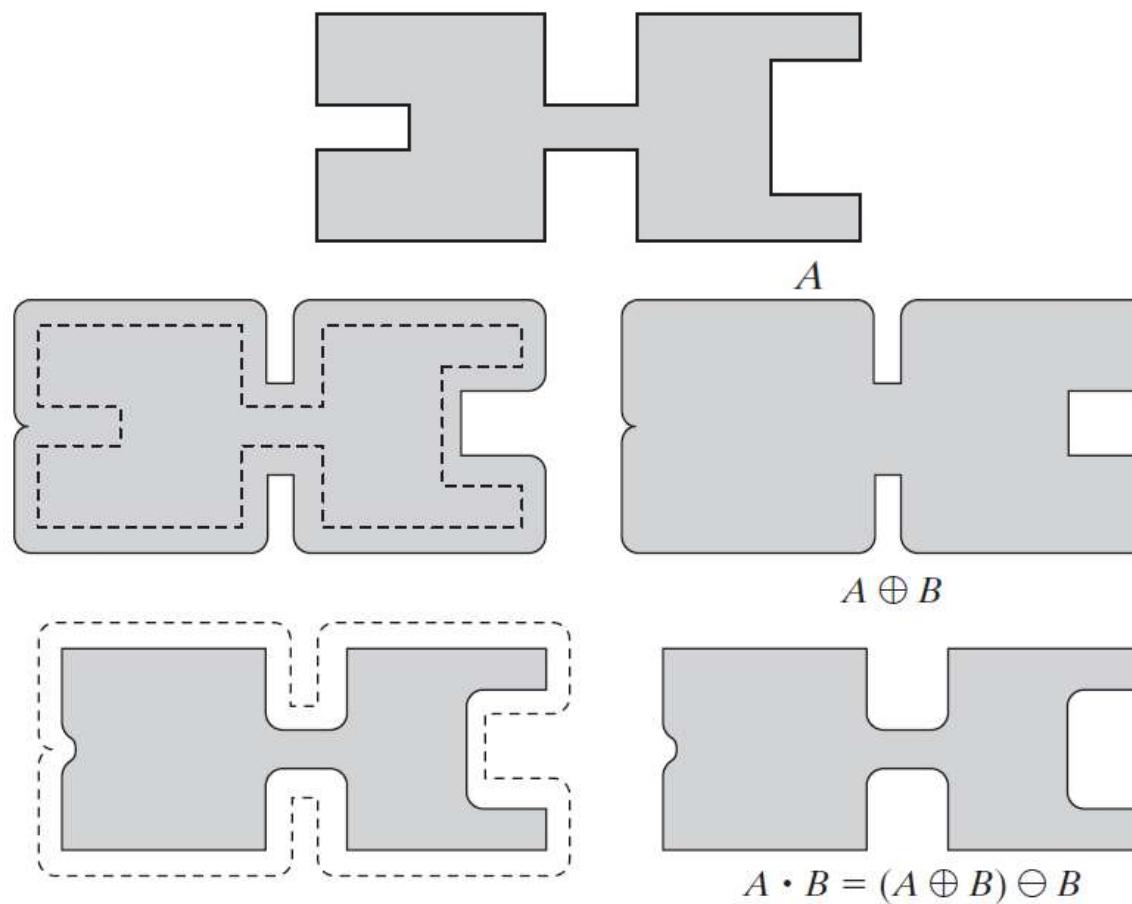
3. Opening and Closing

- Example



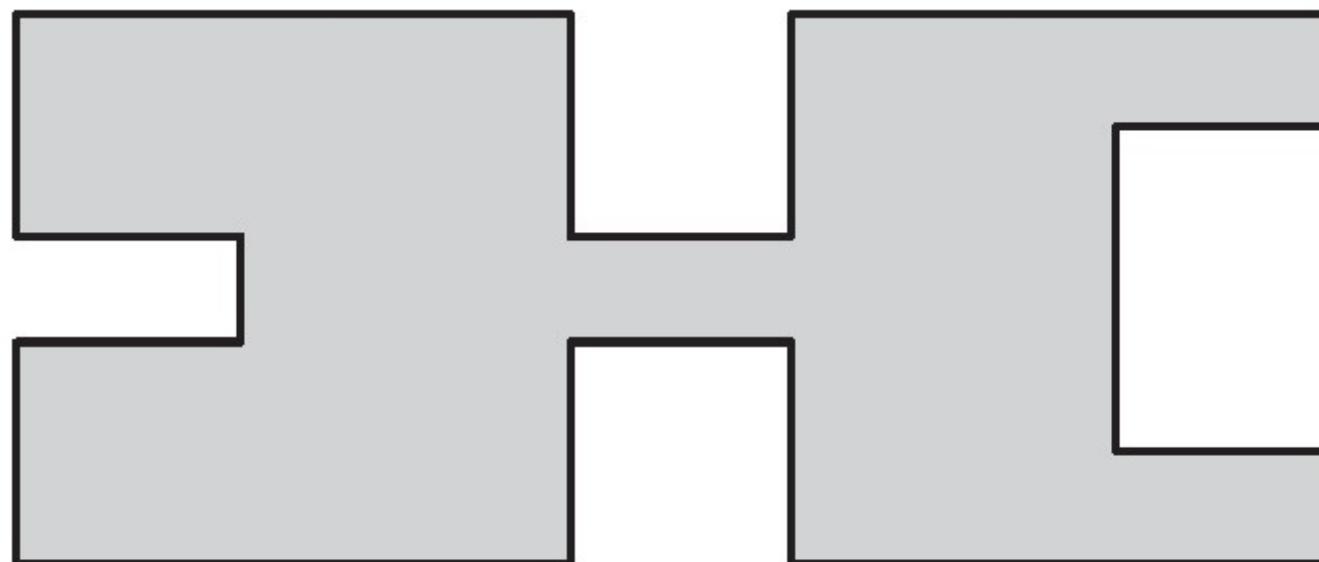
3. Opening and Closing

- Example



3. Opening and Closing

- MATLAB: s37OpenClose.m



A

3. Opening and Closing

- As in the case with dilation and erosion, **opening** and **closing** are **duals** of each other with respect to set complementation and reflection.

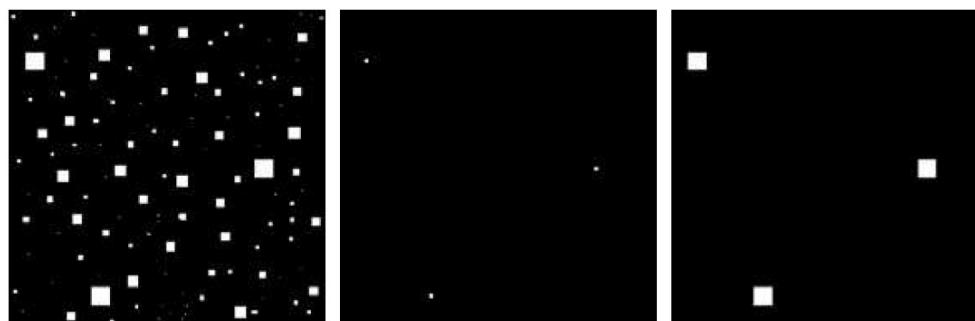
$$(A \bullet B)^c = (A^c \circ \hat{B}) \quad (A \circ B)^c = (A^c \bullet \hat{B})$$

- The opening operation satisfies the following properties:
 - (a) $A \circ B$ is a subset (subimage) of A .
 - (b) If C is a subset of D , then $C \circ B$ is a subset of $D \circ B$.
 - (c) $(A \circ B) \circ B = A \circ B$.
- Similarly, the closing operation satisfies the following properties:
 - (a) A is a subset (subimage) of $A \bullet B$.
 - (b) If C is a subset of D , then $C \bullet B$ is a subset of $D \bullet B$.
 - (c) $(A \bullet B) \bullet B = A \bullet B$.

3. Opening and Closing

- Exercise

Consider the three binary images shown in the following figure. The image on the left is composed of squares of sizes 1, 3, 5, 7, 9, and 15 pixels on the side. The image in the middle was generated by eroding the image on the left with a square structuring element of 1s, of size 13×13 pixels, with the objective of eliminating all the squares, except the largest ones. Finally, the image on the right is the result of dilating the image in the center with the same structuring element, with the objective of restoring the largest squares. You know that erosion followed by dilation is the opening of an image, and you know also that opening generally does not restore objects to their original form. Explain why full reconstruction of the large squares was possible in this case.



3. Opening and Closing

- Example

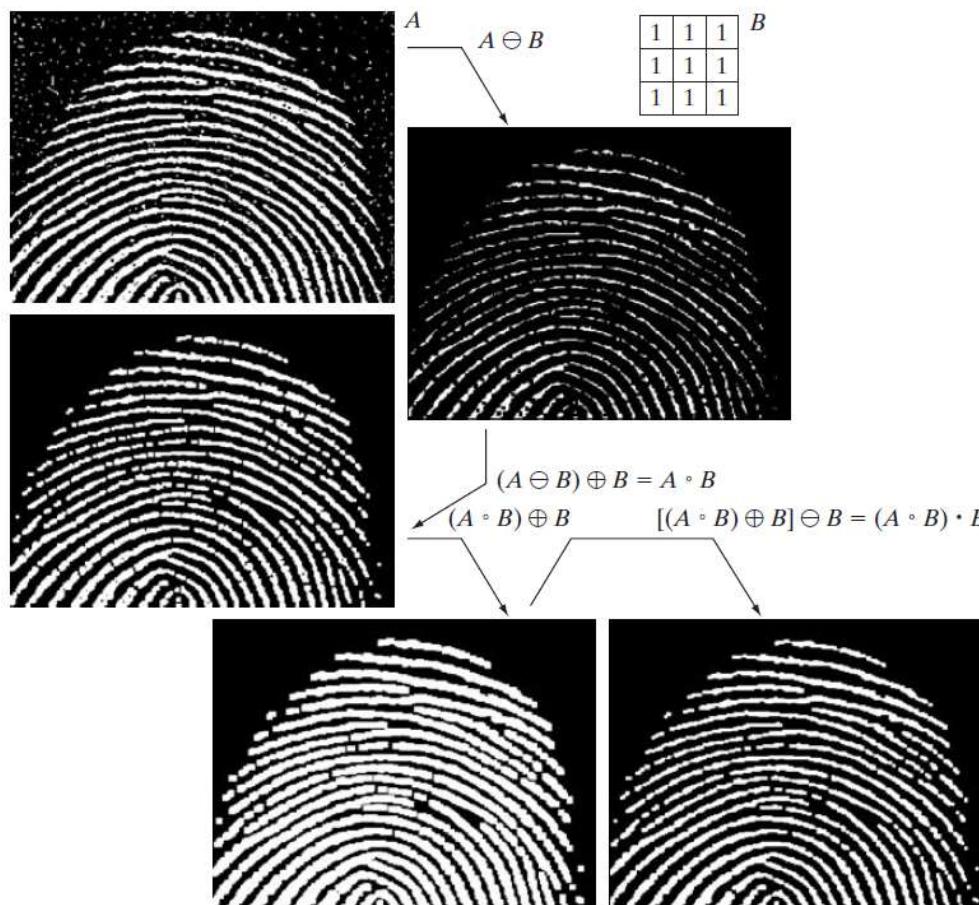
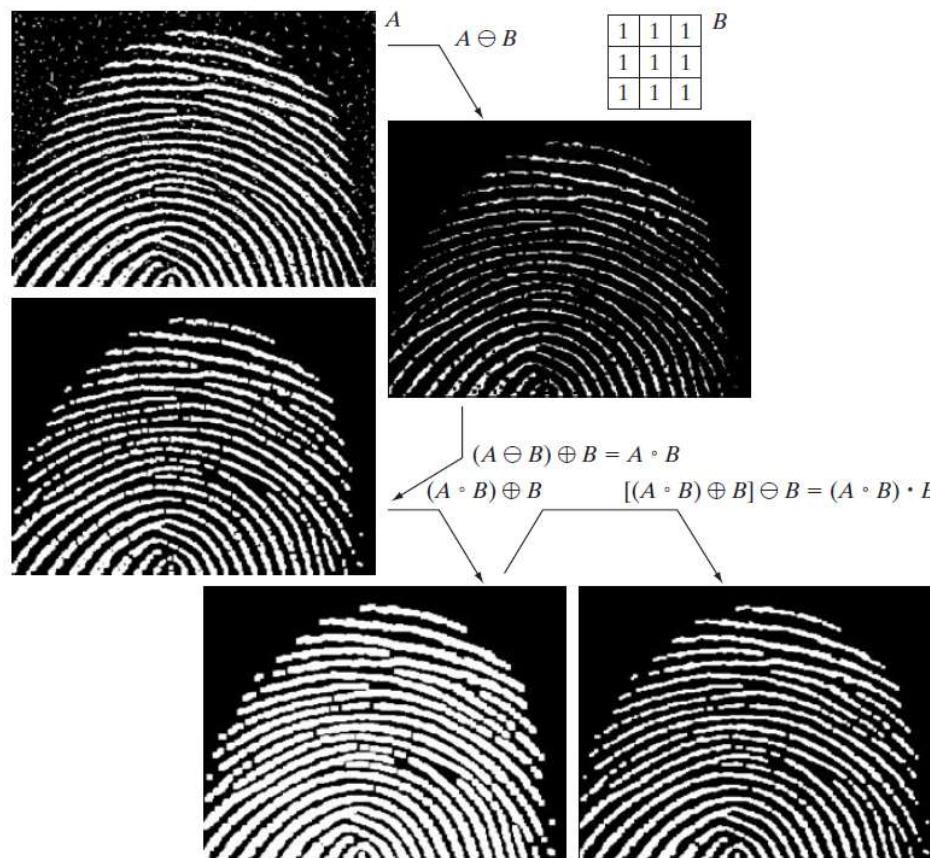


FIGURE 9.11

- Noisy image.
 - Structuring element.
 - Eroded image.
 - Opening of A .
 - Dilation of the opening.
 - Closing of the opening.
- (Original image courtesy of the National Institute of Standards and Technology.)

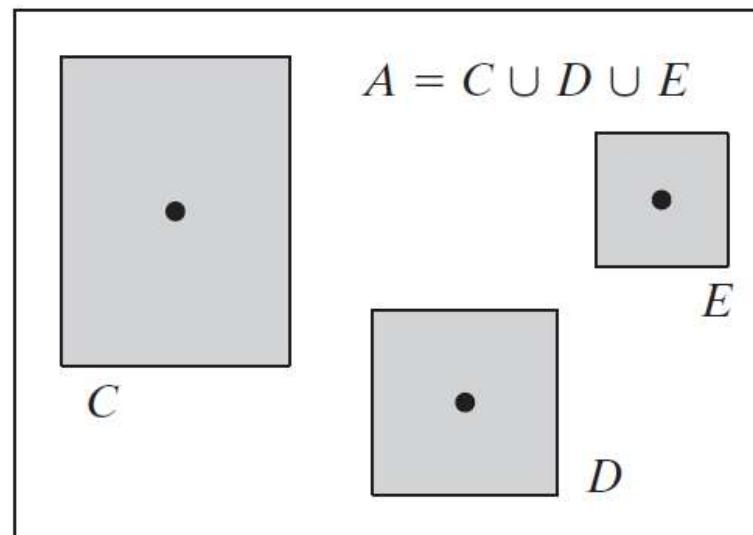
3. Opening and Closing

- MATLAB: s41Fingerprint.m



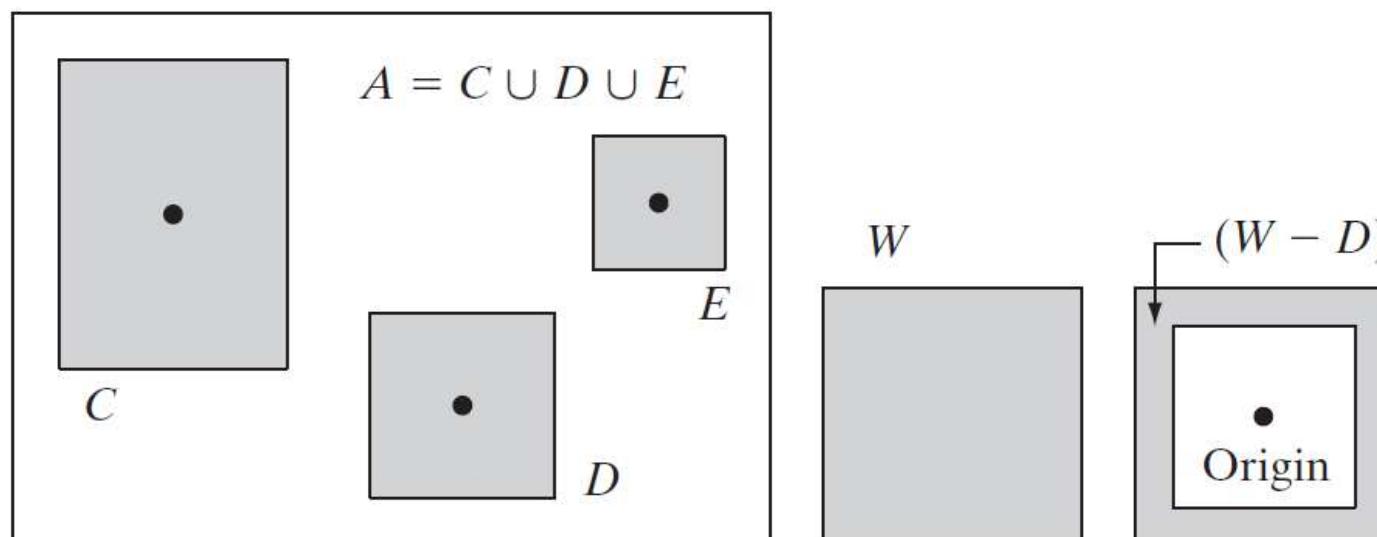
4. The Hit-or-Miss Transformation

- The morphological **hit-or-miss** transformation is a basic tool for **shape detection**.
- The objective is to find the location of one of the shapes, say, D.



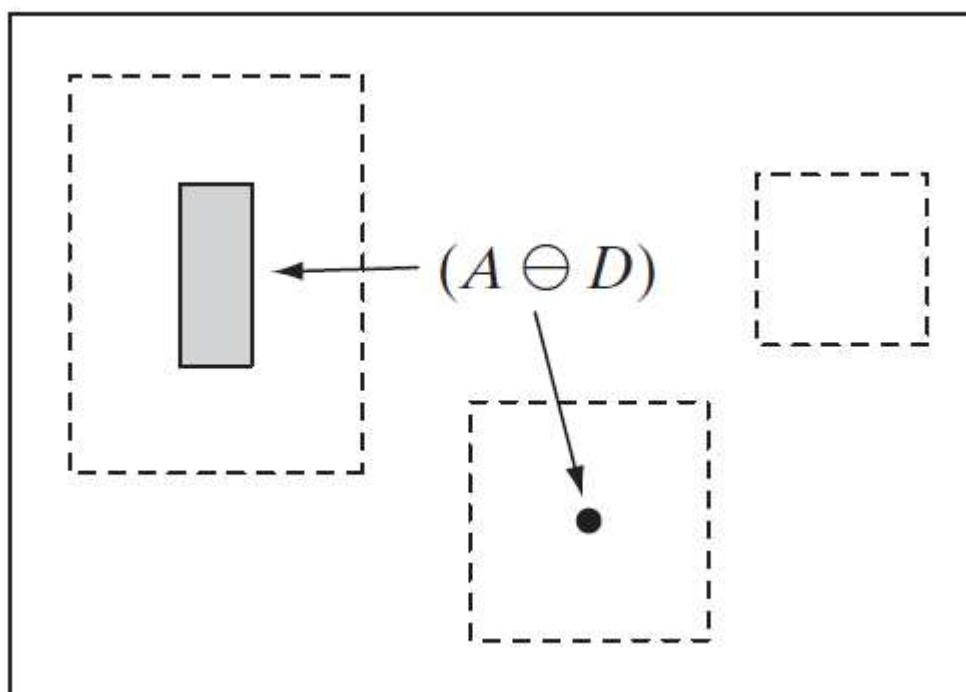
4. The Hit-or-Miss Transformation

- Let D be enclosed by a small window, W.
- The local background of D with respect to W is defined as the set difference ($W - D$).



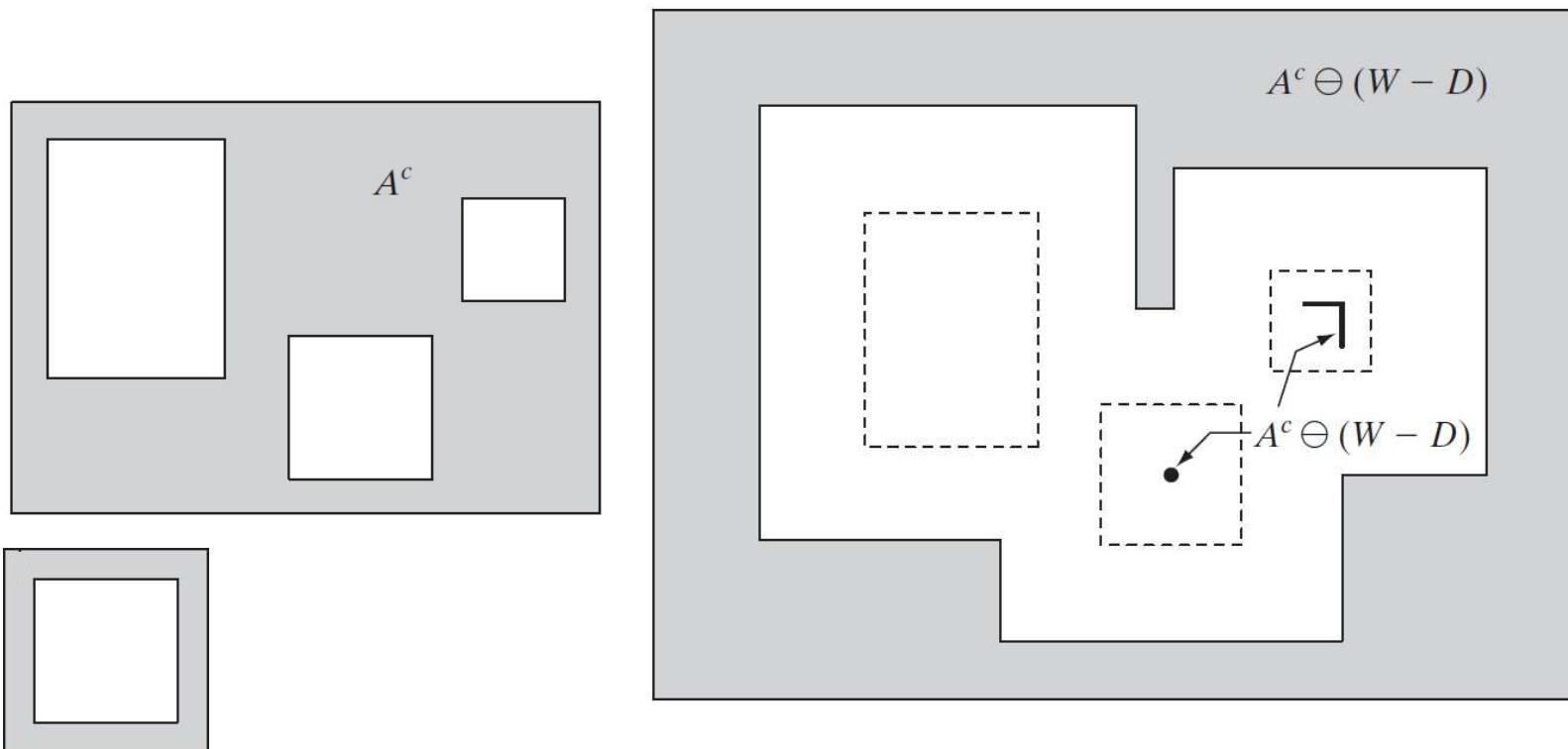
4. The Hit-or-Miss Transformation

- Erosion of A by D (the dashed lines are included for reference).



4. The Hit-or-Miss Transformation

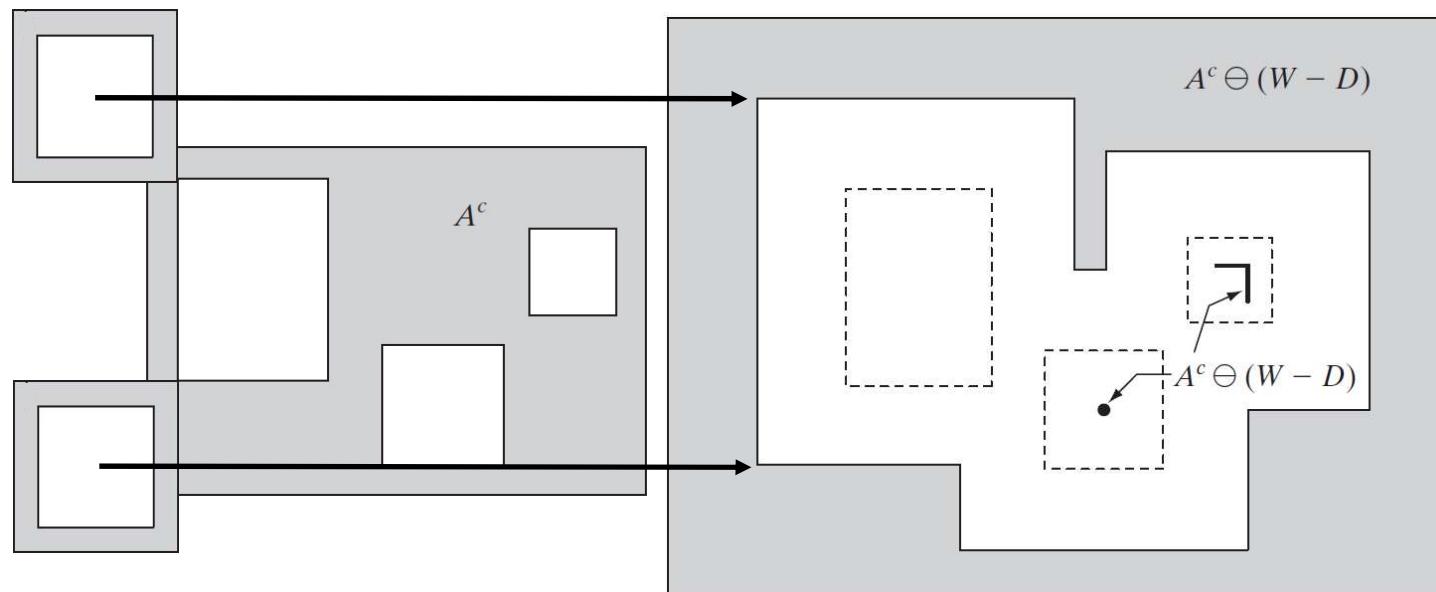
- Erosion of the complement of A by the local background set ($W - D$).



4. The Hit-or-Miss Transformation

- Erosion of the complement of A by the local background set ($W - D$).

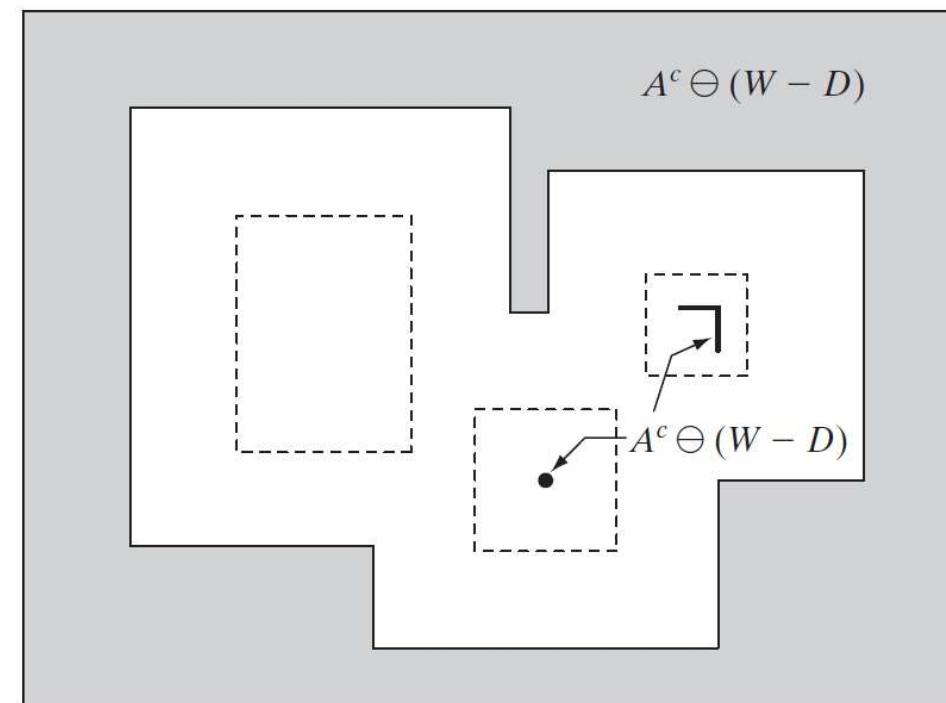
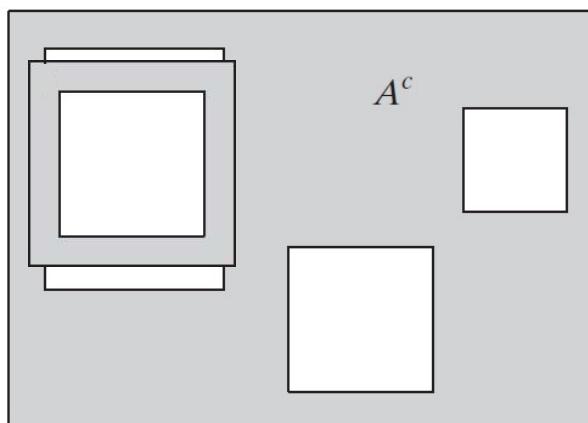
Grey regions match.



4. The Hit-or-Miss Transformation

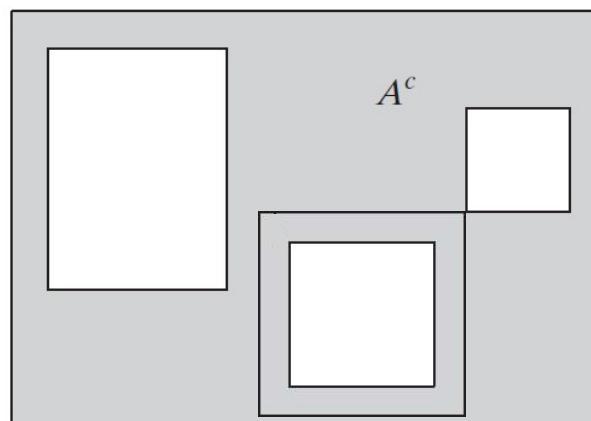
- Erosion of the complement of A by the local background set ($W - D$).

Grey regions do not match.

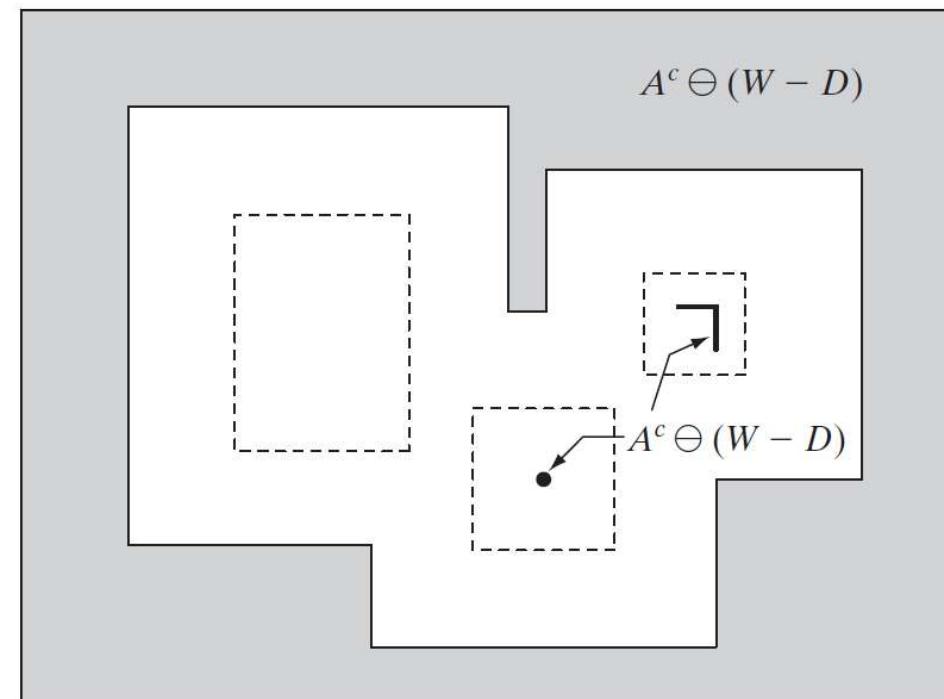


4. The Hit-or-Miss Transformation

- Erosion of the complement of A by the local background set ($W - D$).

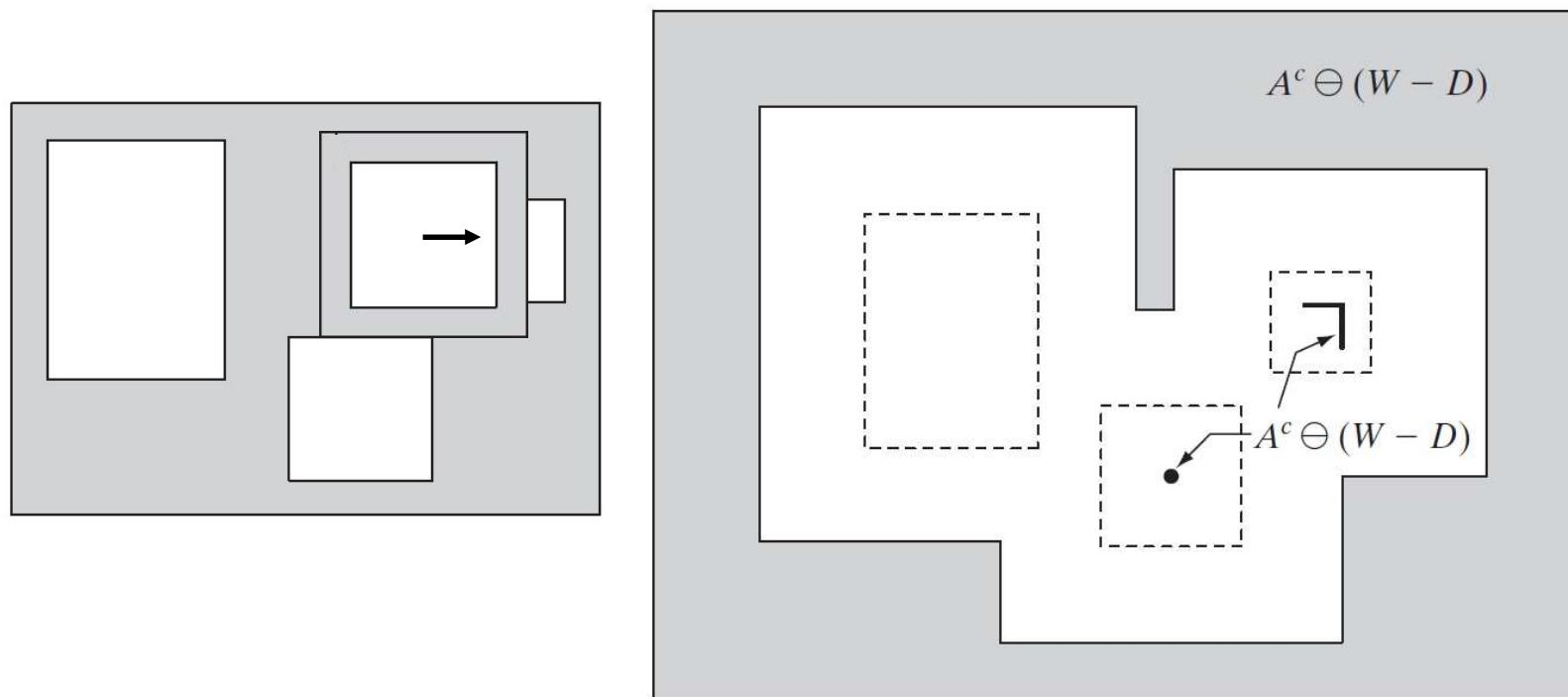


Grey regions match only
in one single point.



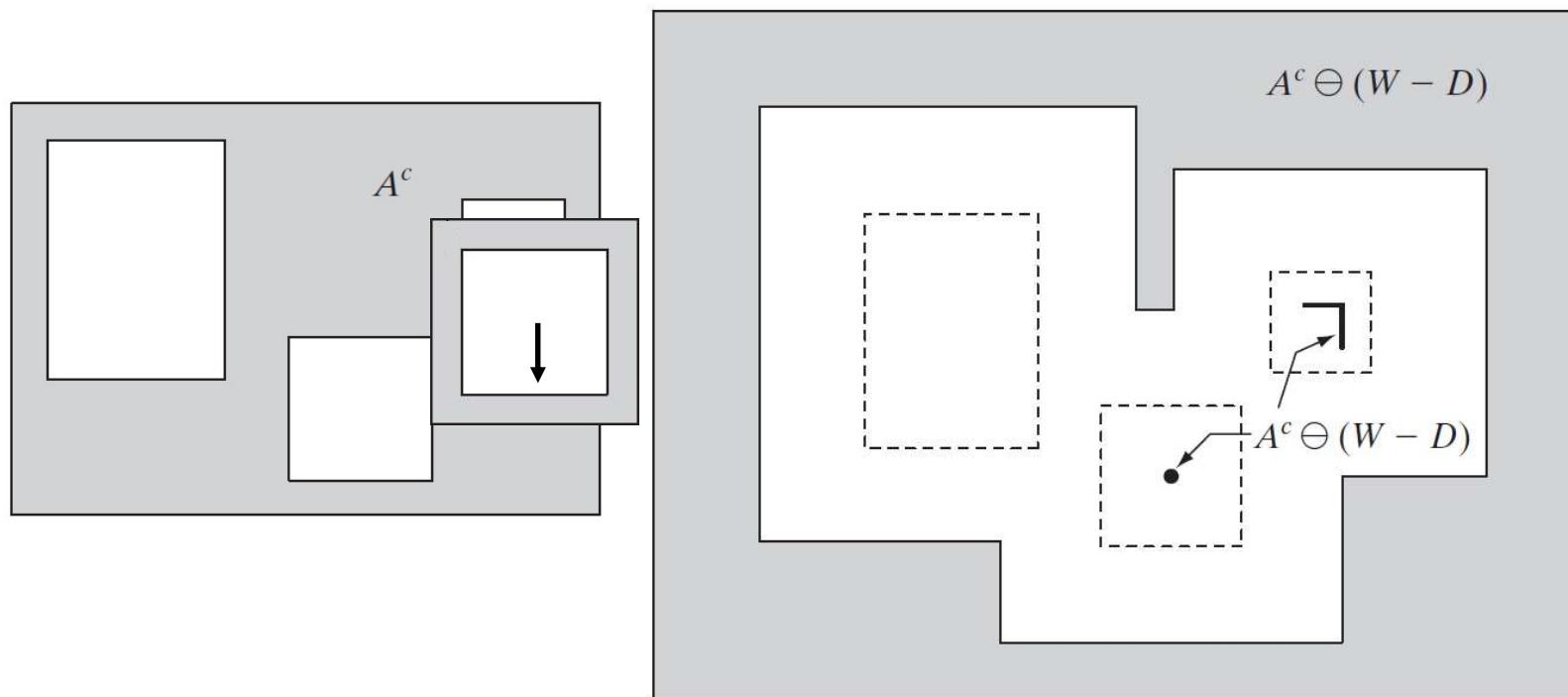
4. The Hit-or-Miss Transformation

- Erosion of the complement of A by the local background set ($W - D$).



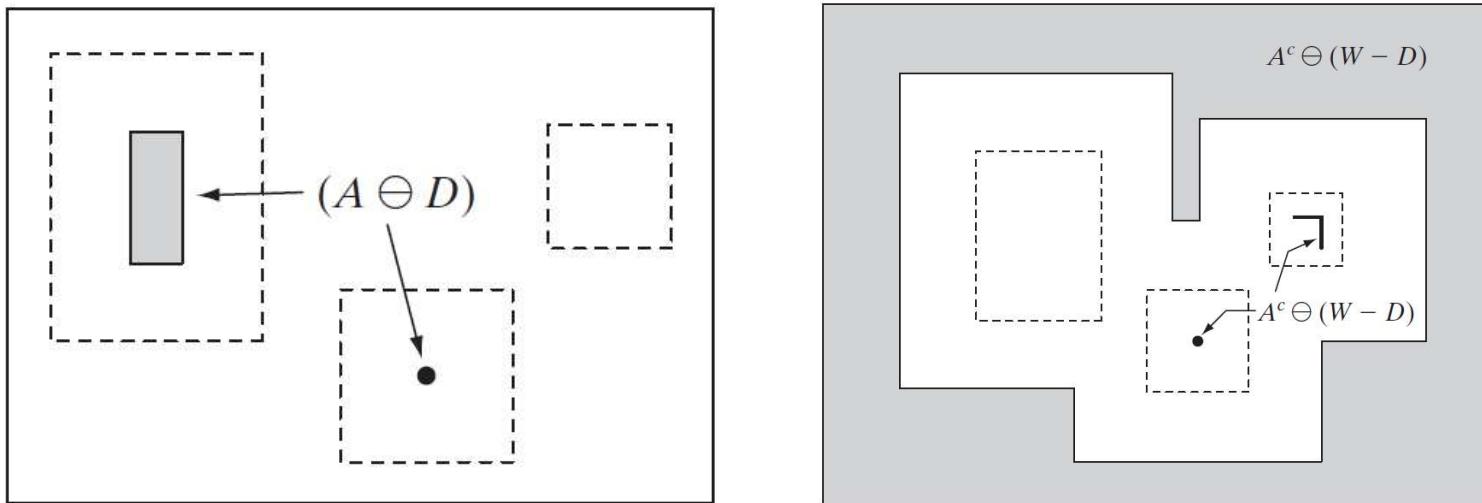
4. The Hit-or-Miss Transformation

- Erosion of the complement of A by the local background set ($W - D$).



4. The Hit-or-Miss Transformation

- The intersection is precisely the location sought.



$$A \circledast B = (A \ominus D) \cap [A^c \ominus (W - D)] \quad B = (B_1, B_2)$$

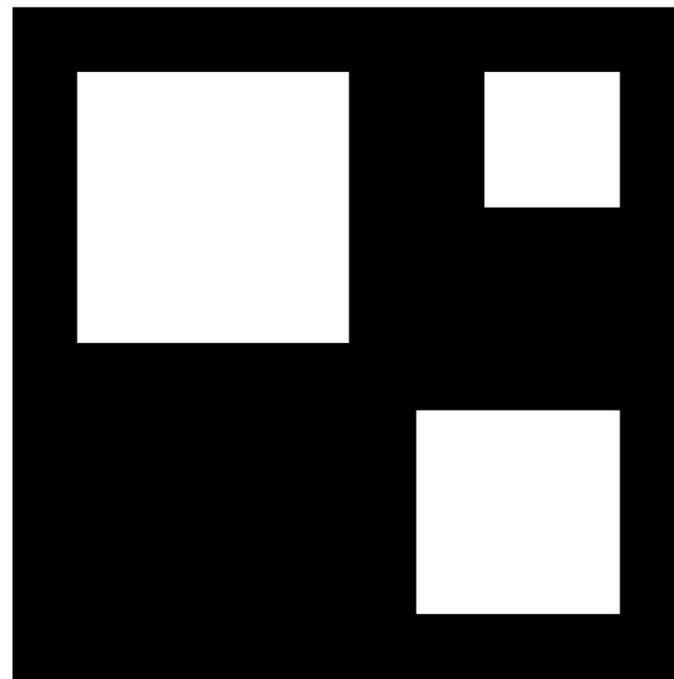
$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$
$$B_1 = D \quad B_2 = (W - D)$$

4. The Hit-or-Miss Transformation

- Using a structuring element B_1 associated with objects and an element B_2 associated with the background is based on the assumption that **objects must be disjoint sets**.
 - Each object has at least a **one-pixel-thick background** around it.
- Sometimes, we may want to detect certain patterns of 1s and 0s within a set, in which case a **background is not required**.
- In such instances, the **hit-or-miss transformation** is reduced to a **simple erosion**.
 - It is still a set of matches, but without the requirement of a background match for detecting individual objects.
- This simplified pattern detection scheme is used in some of the following algorithms.

4. The Hit-or-Miss Transformation

- MATLAB: s52HitMiss.m



5. Boundary Extraction

- The boundary of a set A is defined as

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

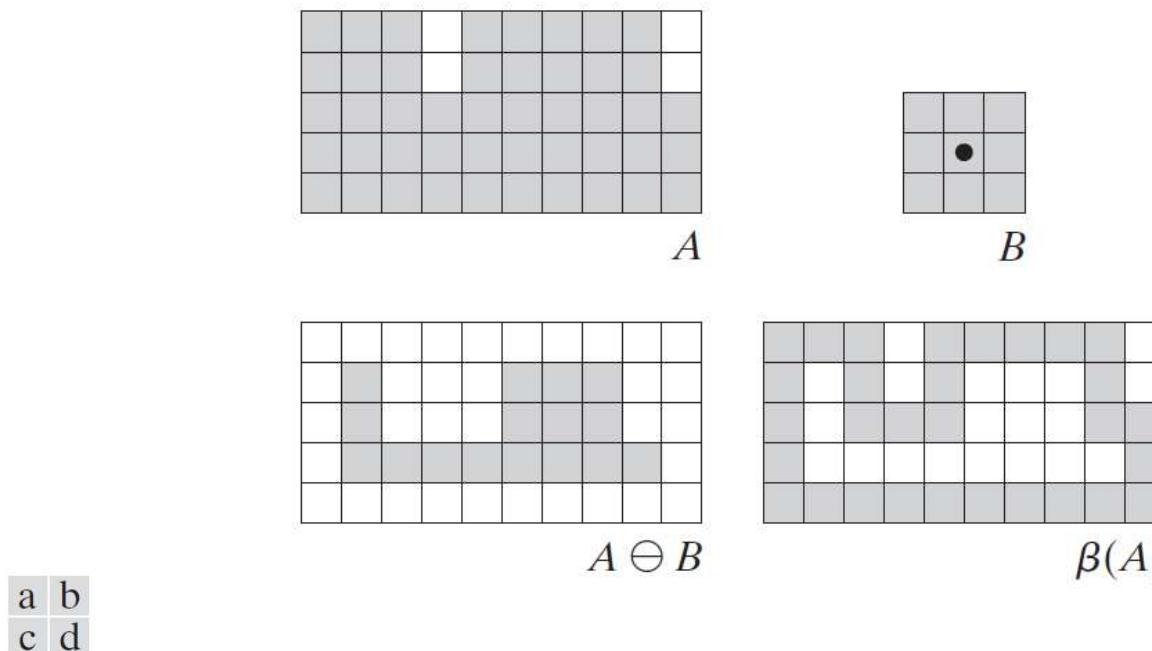


FIGURE 9.13 (a) Set A . (b) Structuring element B . (c) A eroded by B . (d) Boundary, given by the set difference between A and its erosion.

5. Boundary Extraction

- MATLAB: s55Boundary.m



a b

FIGURE 9.14
(a) A simple
binary image, with
1s represented in
white. (b) Result
of using
Eq. (9.5-1) with
the structuring
element in
Fig. 9.13(b).

6. Hole Filling

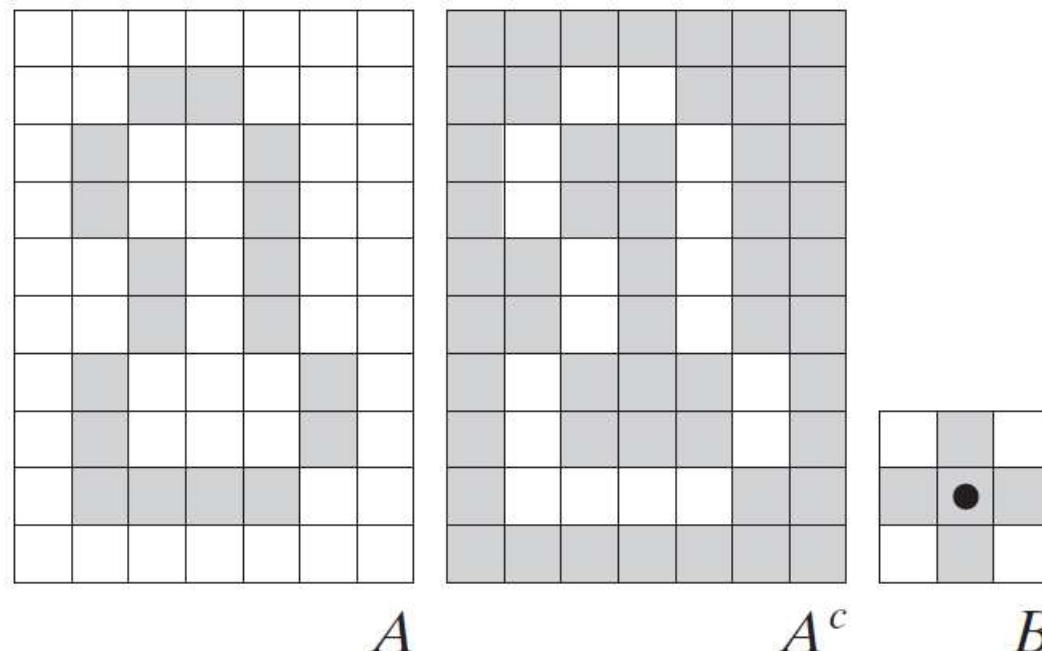
- A **hole** may be defined as a background region surrounded by a connected border of foreground pixels.
- Let A denote a set whose elements are 8-connected boundaries, each boundary enclosing a background region (i.e., a hole).
- Given a point in each hole, the objective is to fill all the holes with 1s.
- First form an array, X_0 , of 0s (the same size as the array containing A), except at the locations in X_0 corresponding to the given point in each hole, which we set to 1.
- Then, we apply the following procedure,

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

6. Hole Filling

- Then, we apply the following procedure,

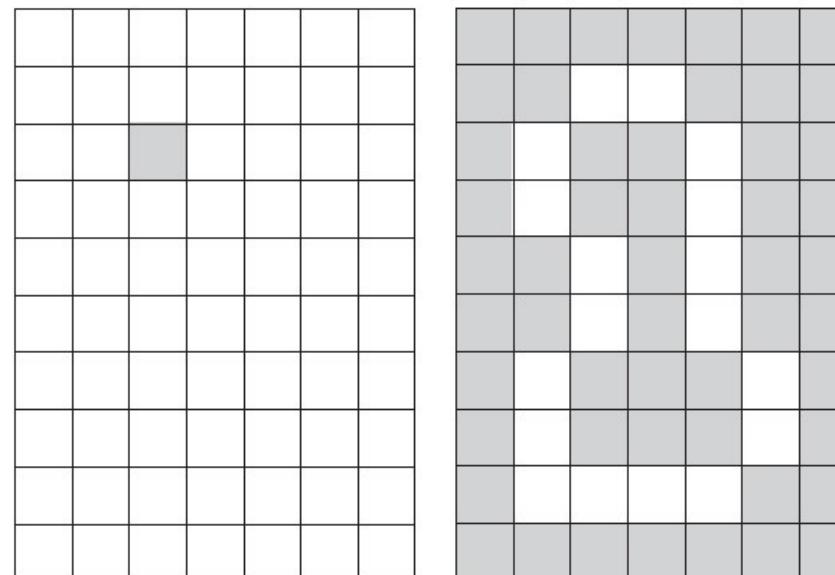
$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \dots$$



6. Hole Filling

- Then, we apply the following procedure,

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

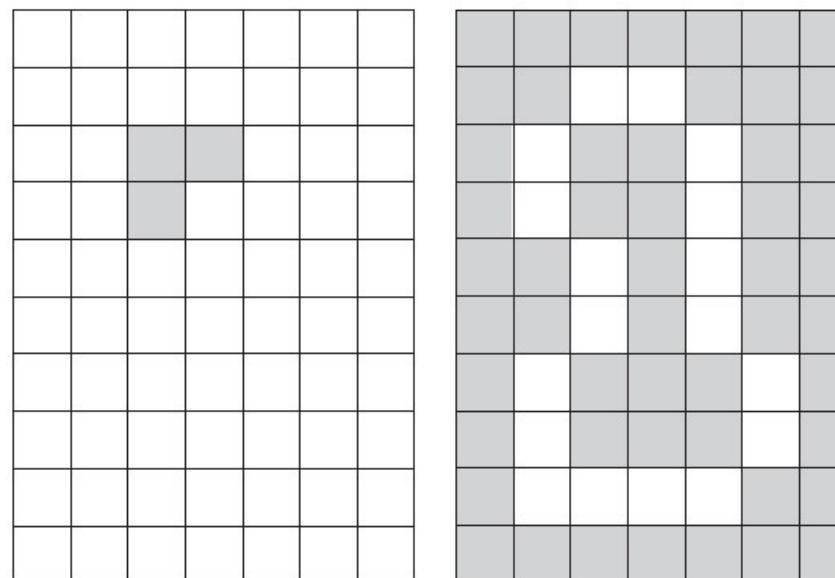


X_0

6. Hole Filling

- Then, we apply the following procedure,

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

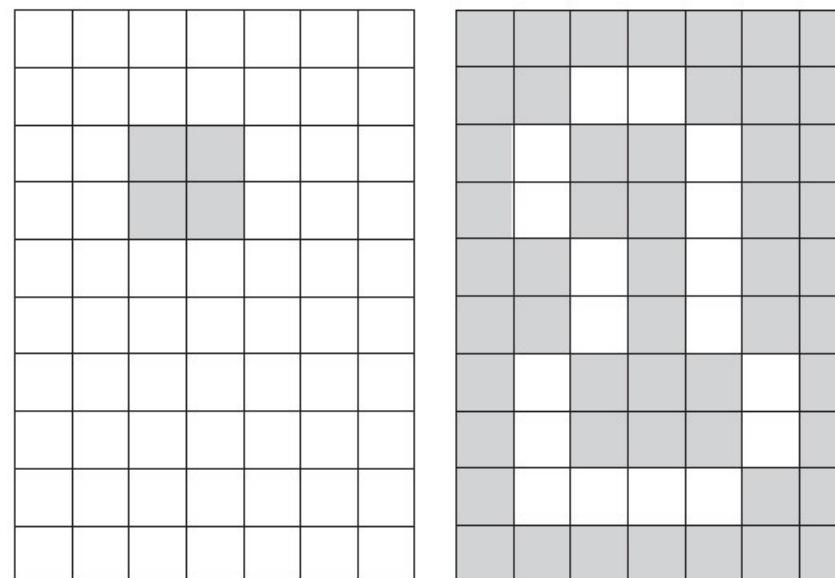


X_1

6. Hole Filling

- Then, we apply the following procedure,

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

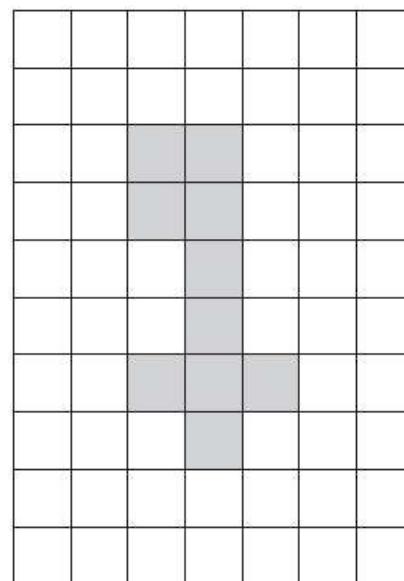
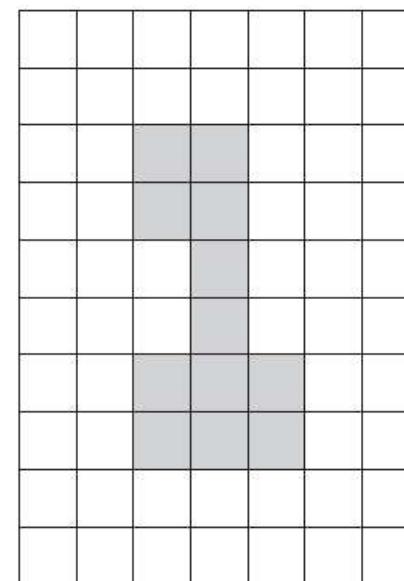
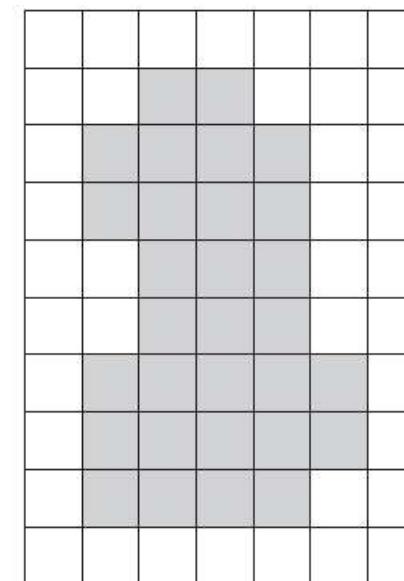


X_2

6. Hole Filling

- Then, we apply the following procedure,

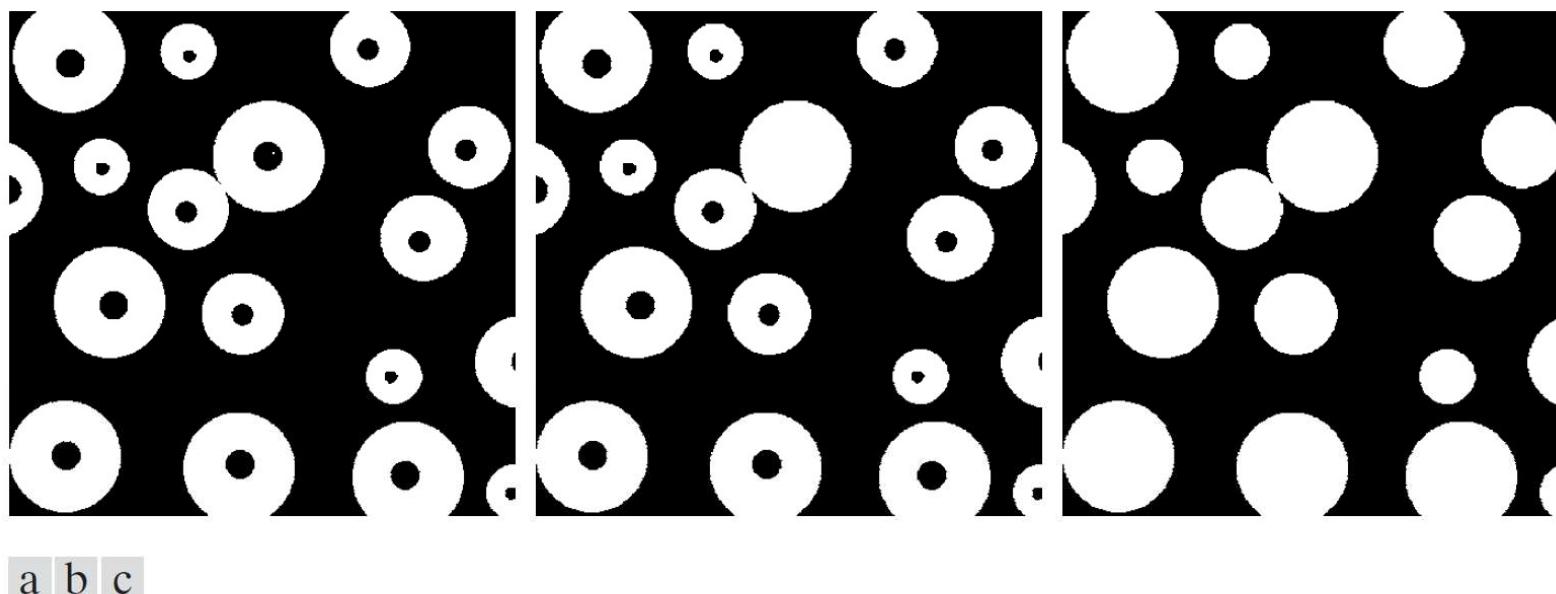
$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \dots$$

 X_6  X_8  $X_8 \cup A$

6. Hole Filling

- Example/ Exercise: Fig0916(a)(region-filling-reflections).tif

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



a b c

FIGURE 9.16 (a) Binary image (the white dot inside one of the regions is the starting point for the hole-filling algorithm). (b) Result of filling that region. (c) Result of filling all holes.

7. Extraction of Connected Components

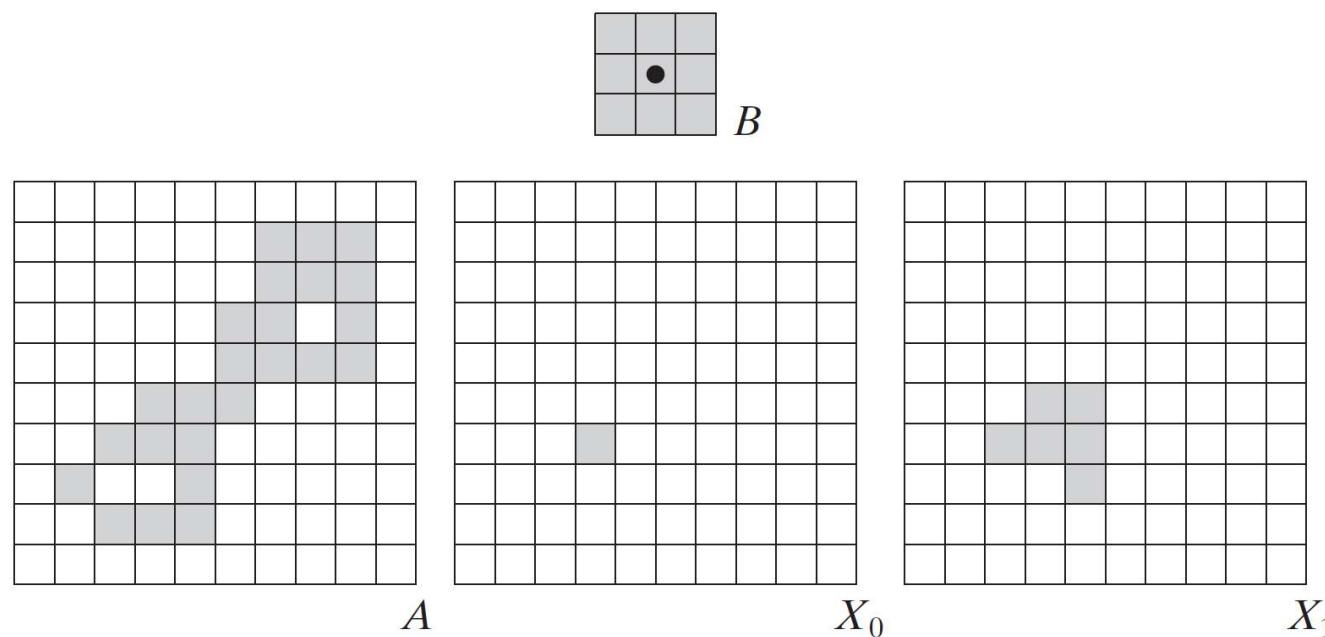
- Extraction of connected components from a binary image is central to many automated image analysis applications.
- Let A be a set containing one or more connected components.
- Form an array, X_0 , whose elements are 0s, except at each location known to correspond to a point in each connected component in A , which we set to 1.
- Then, we apply the following procedure,

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

7. Extraction of Connected Components

- Then, we apply the following procedure,

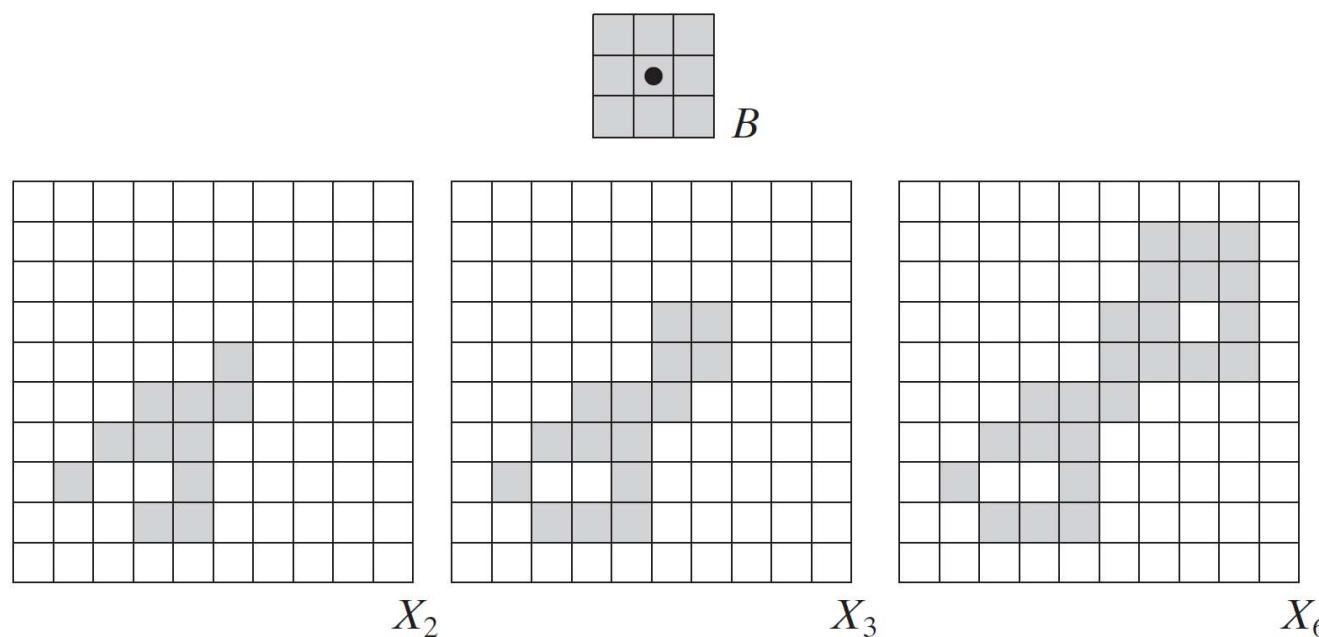
$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1, 2, 3, \dots$$



7. Extraction of Connected Components

- Then, we apply the following procedure,

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



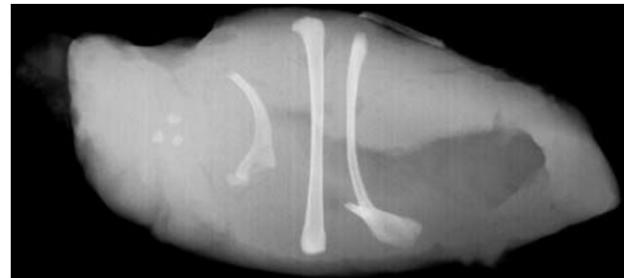
7. Extraction of Connected Components

- Example/Exercise: X-ray image of a chicken breast that contains bone fragments.
- It is of considerable interest to be able to detect such objects in processed food before packaging and/or shipping.
- Fig0918(a)(Chickenfilet with bones).tif



7. Extraction of Connected Components

- In this particular case, the density of the bones is such that their nominal intensity values are different from the background.
- This makes extraction of the bones from the background a simple matter by using a single threshold.



7. Extraction of Connected Components

- We use erosion to make sure that only objects of “significant” size remain.
- In this example, we define as significant any object that remains after erosion with a 5x5 structuring element of 1s



7. Extraction of Connected Components

- We use erosion to make sure that only objects of “significant” size remain.
- In this example, we define as significant any object that remains after erosion with a 5x5 structuring element of 1s.



7. Extraction of Connected Components

- The next step is to analyze the size of the objects that remain. We label these objects by extracting the connected components in the image.

a
b
c d

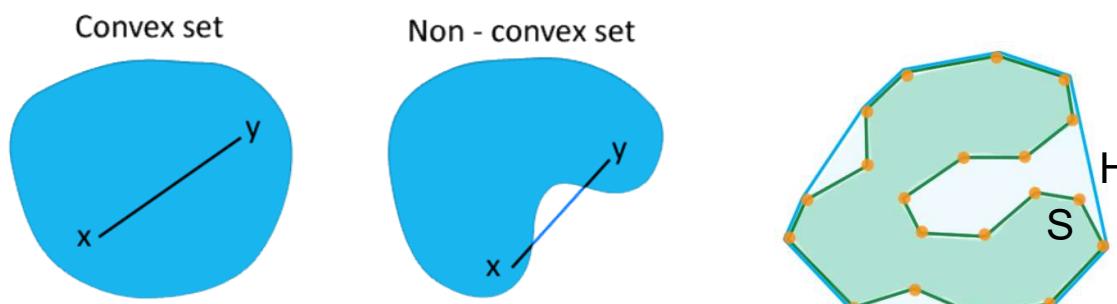
FIGURE 9.18
 (a) X-ray image of chicken filet with bone fragments.
 (b) Thresholded image.
 (c) Image eroded with a 5×5 structuring element of 1s.
 (d) Number of pixels in the connected components of (c).
 (Image courtesy of NTB Elektronische Geraete GmbH, Diepholz, Germany, www.ntbxray.com.)



Connected component	No. of pixels in connected comp
01	11
02	9
03	9
04	39
05	133
06	1
07	1
08	743
09	7
10	11
11	11
12	9
13	9
14	674
15	85

8. Convex Hull

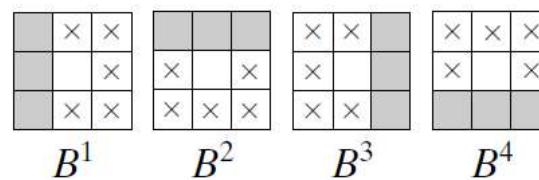
- A set A is said to be convex if the straight line segment joining any two points in A lies entirely within A
- The *convex hull* H of an arbitrary set S is the smallest convex set containing S .
- The set difference is called the convex deficiency of S .
- The convex hull and convex deficiency are useful for object description



https://www.easycalculation.com/mathematics-dictionary/convex_set.html

8. Convex Hull

- Let B^i , $i = 1..4$ represent the following four structuring elements



- The procedure consists of implementing the equation

$$X_k^i = (X_{k-1} \circledast B^i) \cup A \quad i = 1, 2, 3, 4 \quad \text{and} \quad k = 1, 2, 3, \dots$$

$$X_0^i = A$$

- When the procedure converges, we let

$$D^i = X_k^i$$

- Then the convex hull of A is

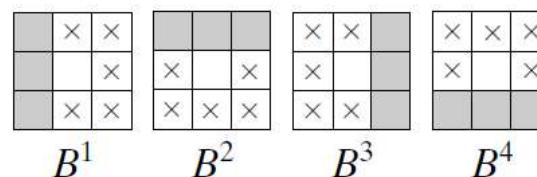
$$C(A) = \bigcup_{i=1}^4 D^i$$

8. Convex Hull

- In other words, the method consists of iteratively applying the hit-or-miss transform to A with B^i .
- The union of the four resulting D s constitutes the convex hull.
- We are using the simplified implementation of the hit-or-miss transform in which no background match is required, as discussed before.
- The x entries indicate “don’t care” conditions.
- A structuring element is said to have found a match in A if the 3x3 region of A under the structuring element mask at that location matches the pattern of the mask.

8. Convex Hull

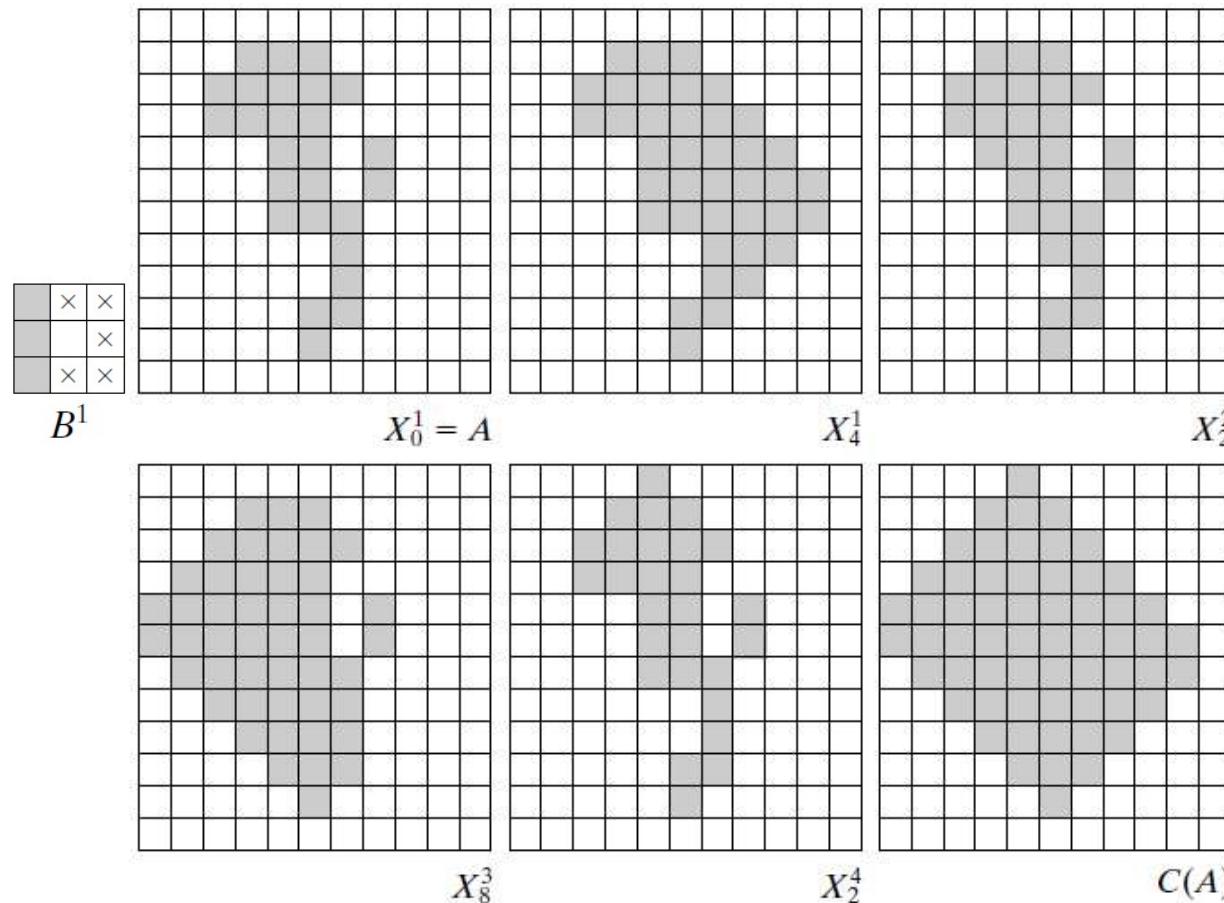
- For a particular mask, a pattern match occurs when the center of the 3×3 region in A is 0, and the three pixels under the shaded mask elements are 1.



- The values of the other pixels in the region do not matter.
- Observe that B^i is a clockwise rotation of B^{i-1} by 90° .

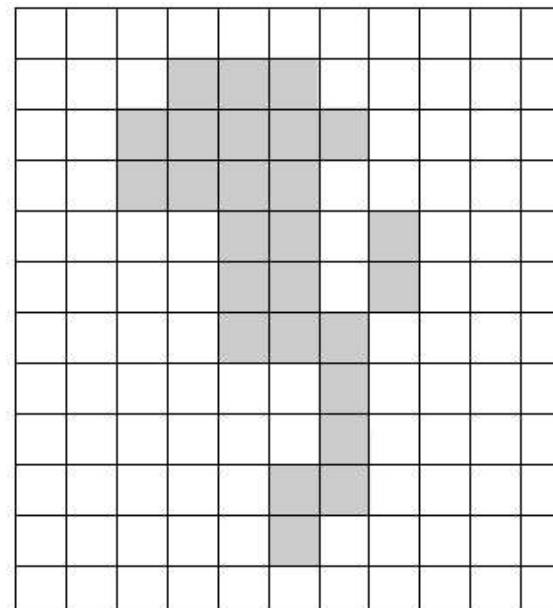
8. Convex Hull

- Example



8. Convex Hull

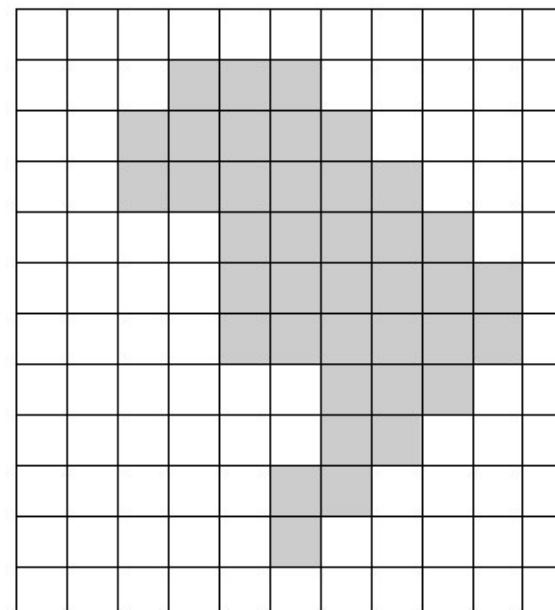
- Example



A

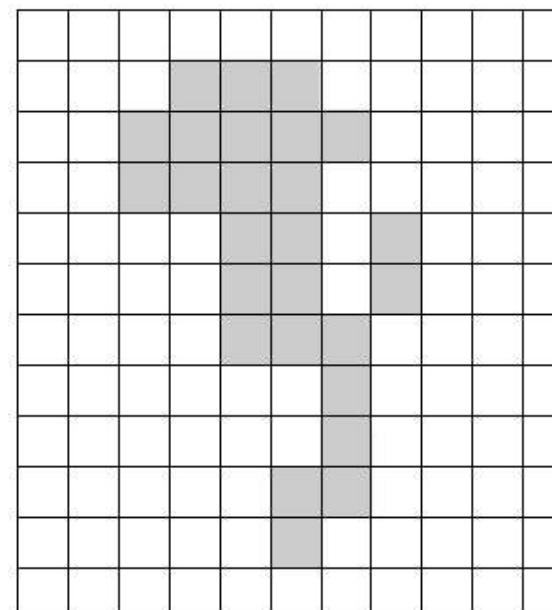
8. Convex Hull

- Example


$$X_4^1$$

8. Convex Hull

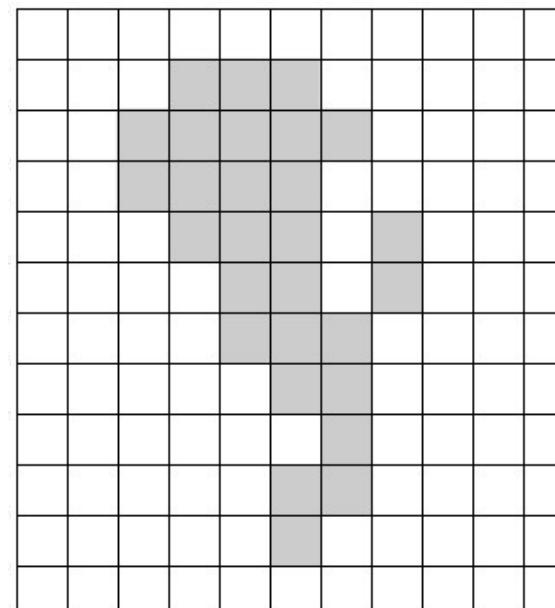
- Example



A

8. Convex Hull

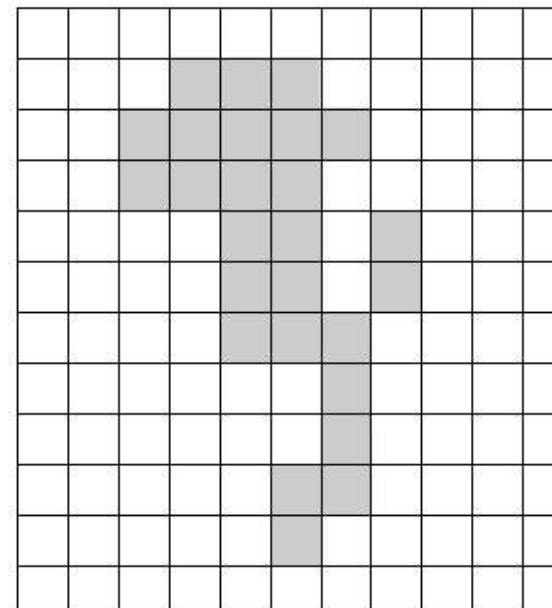
- Example



$$X_2^2$$

8. Convex Hull

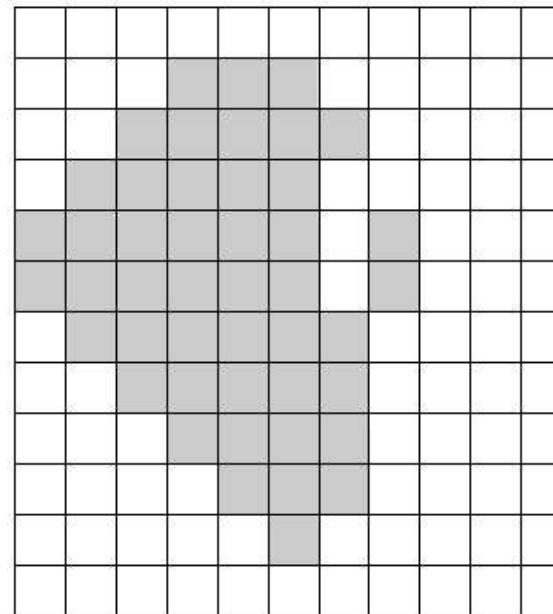
- Example



A

8. Convex Hull

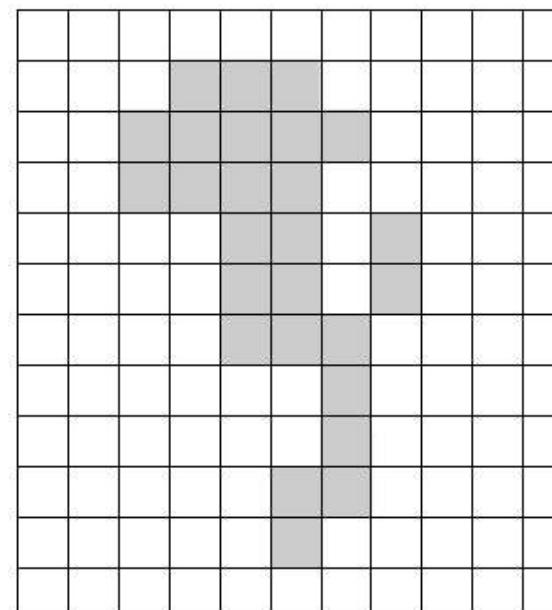
- Example



$$X_8^3$$

8. Convex Hull

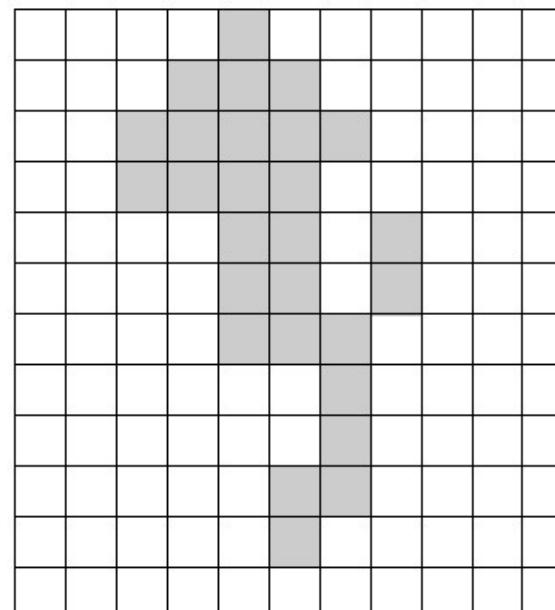
- Example



A

8. Convex Hull

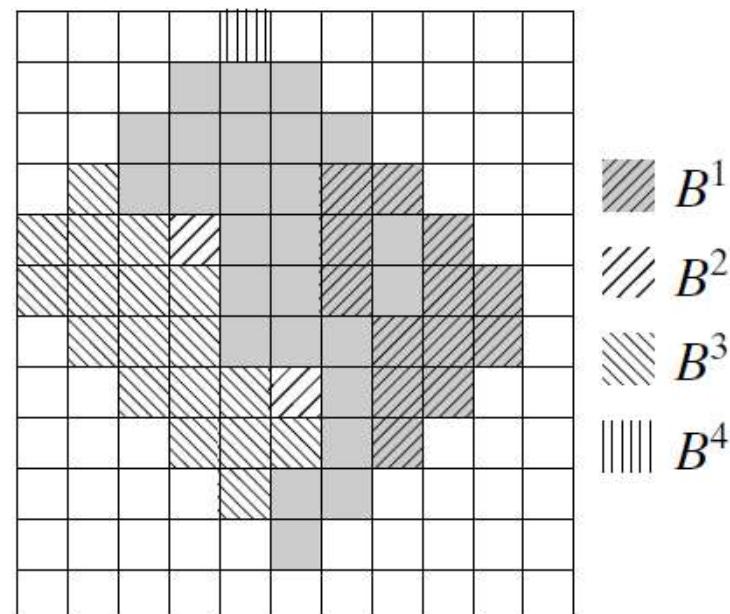
- Example



$$X_2^4$$

8. Convex Hull

- Example



8. Convex Hull

- Example

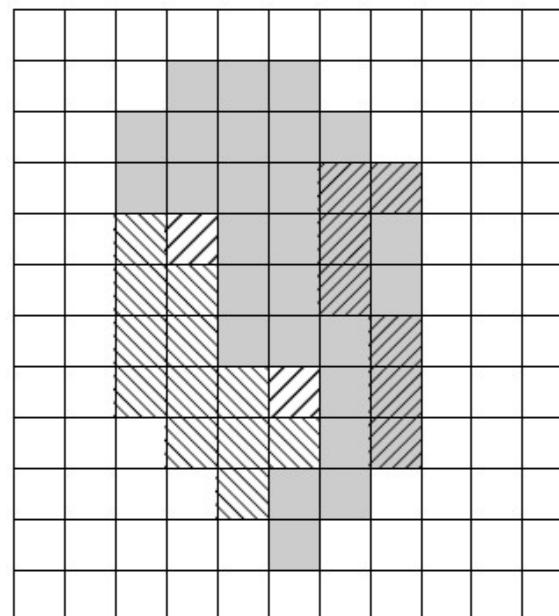


FIGURE 9.20
Result of limiting growth of the convex hull algorithm to the maximum dimensions of the original set of points along the vertical and horizontal directions.

9. Thinning

- The **thinning** of a set A by a structuring B can be defined in terms of the hit-or-miss transform

$$A \otimes B = A - (A \circledast B)$$

- A more useful expression for thinning A symmetrically is based on a **sequence of structuring elements**

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

- The entire process below is repeated until no further changes occur.

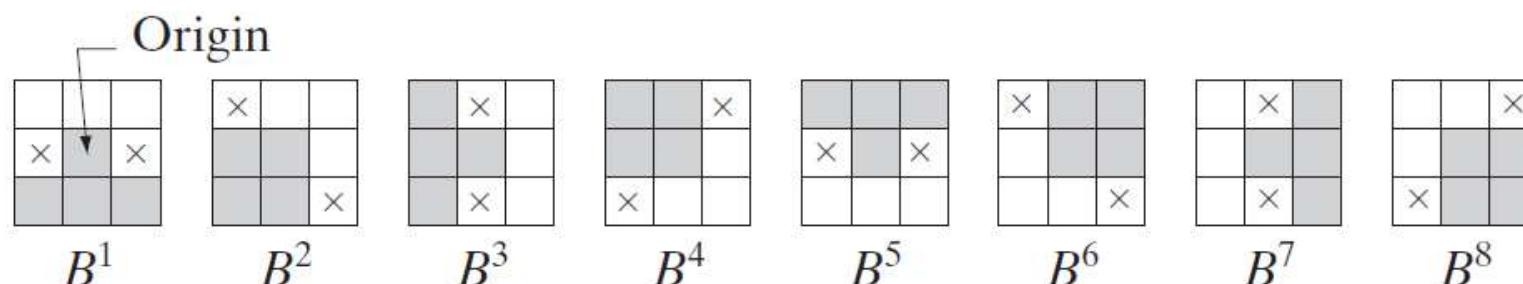
$$A \otimes \{B\} = ((\dots((A \otimes B^1) \otimes B^2) \dots) \otimes B^n)$$

9. Thinning

- The thinning of a set A by a structuring B can be defined in terms of the hit-or-miss transform:

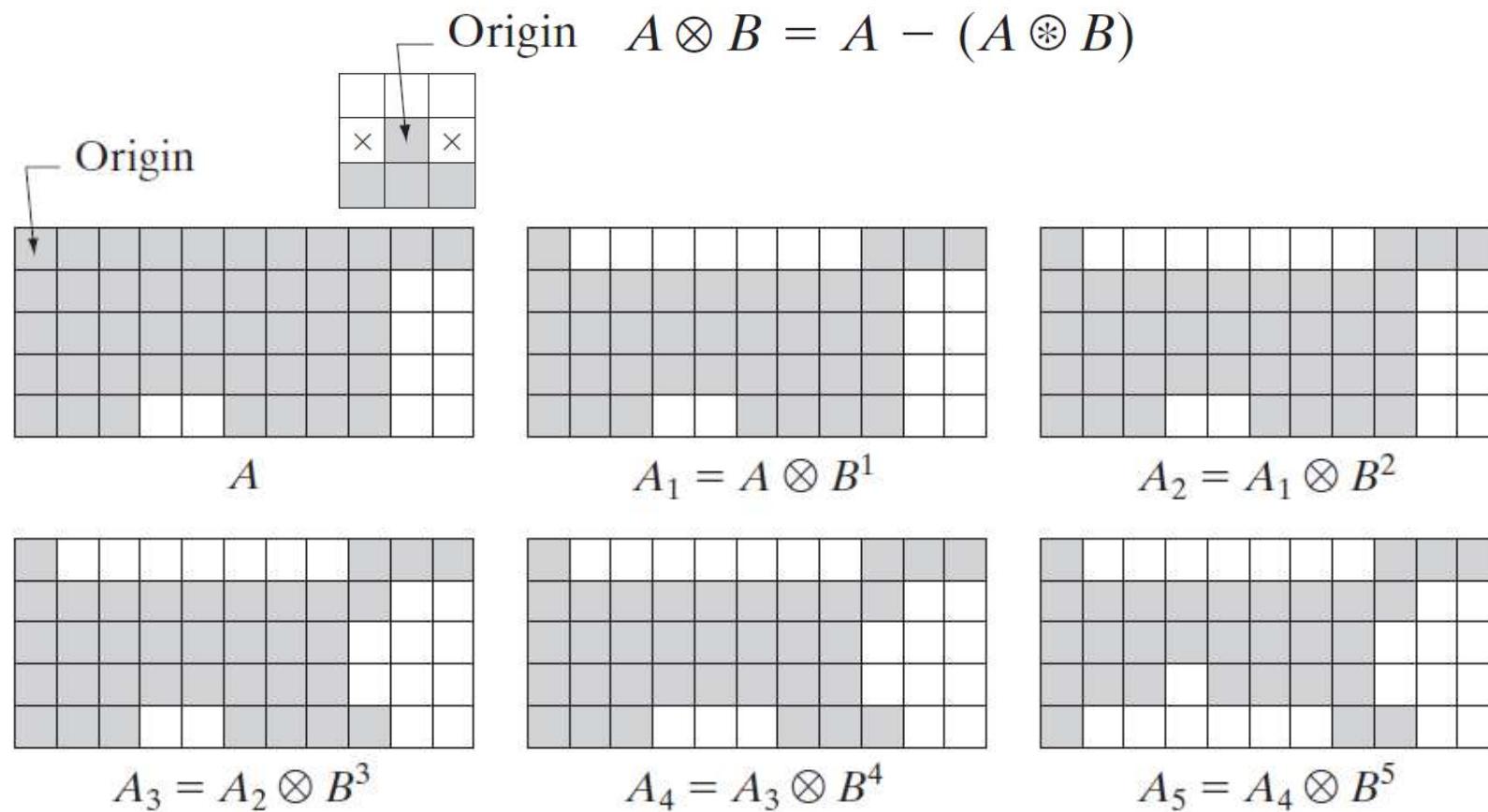
$$\{B\} = \{B^1, B^2, B^3, \dots, B^n\}$$

- B^i is a clockwise rotation of B^{i-1} .



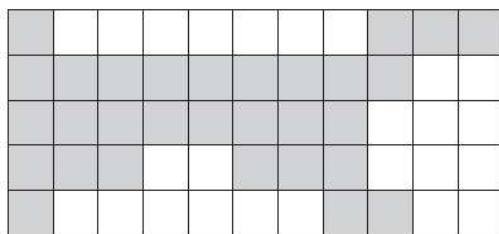
9. Thinning

- Example

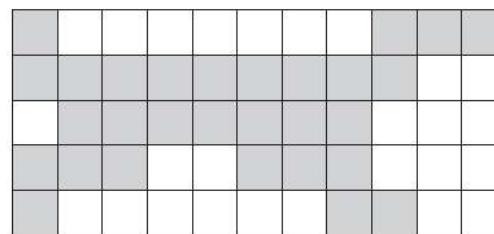


9. Thinning

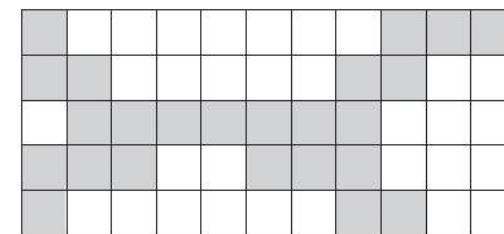
- Example



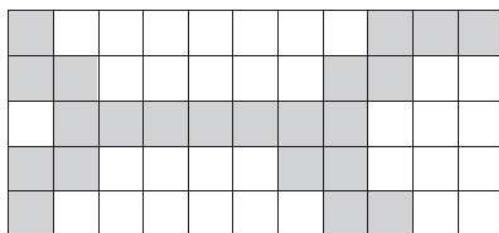
$$A_6 = A_5 \otimes B^6$$



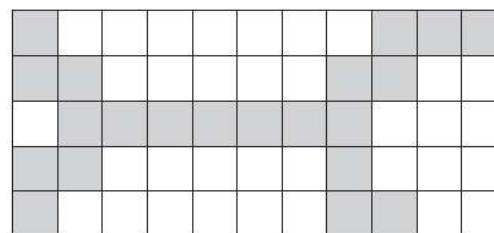
$$A_8 = A_6 \otimes B^{7,8}$$



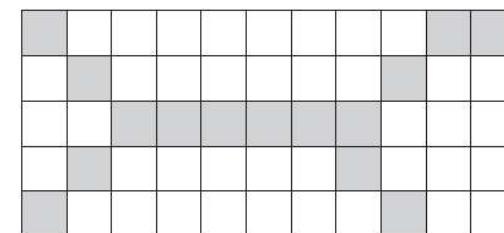
$$A_{8,4} = A_8 \otimes B^{1,2,3,4}$$



$$A_{8,5} = A_{8,4} \otimes B^5$$



$A_{8,6} = A_{8,5} \otimes B^6$
No more changes after this.



$A_{8,6}$ converted to
 m -connectivity.

a		
b	c	d
e	f	g
h	i	j
k	l	m

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

9. Thinning

- MATLAB: s90Thinning.m



9. Thinning

- Example



9. Thinning

- Example



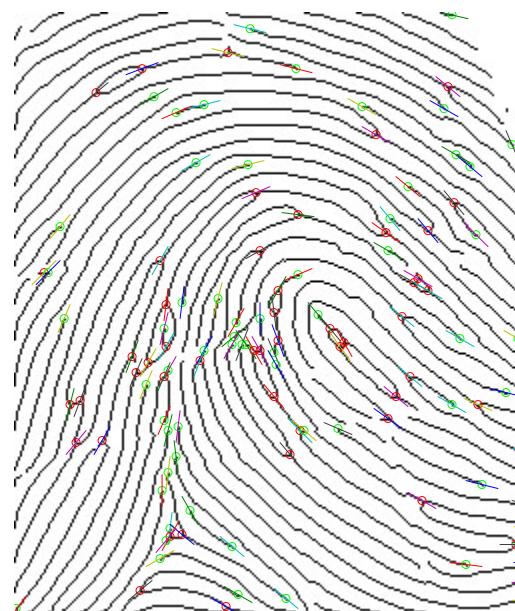
9. Thinning

- Example



9. Thinning

- Example



9. Thickening

- **Thickening** is the morphological **dual of thinning** and is defined by the expression

$$A \odot B = A \cup (A \circledast B)$$

where is B a structuring element suitable for thickening. As in thinning, thickening can be defined as a sequential operation

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

- The structuring elements used for thickening have the same form as those shown before, but with all 1s and 0s interchanged.

9. Thickening

- However, a separate algorithm for thickening is seldom used in practice.
- Instead, the usual procedure is to thin the background of the set in question and then complement the result.
- In other words, to thicken a set A ,
 1. We form $C = A^C$;
 2. Thin C ; and then
 3. Form C^C .
- Thickening by this method usually is followed by postprocessing to remove disconnected points.

9. Thickening

- Example:

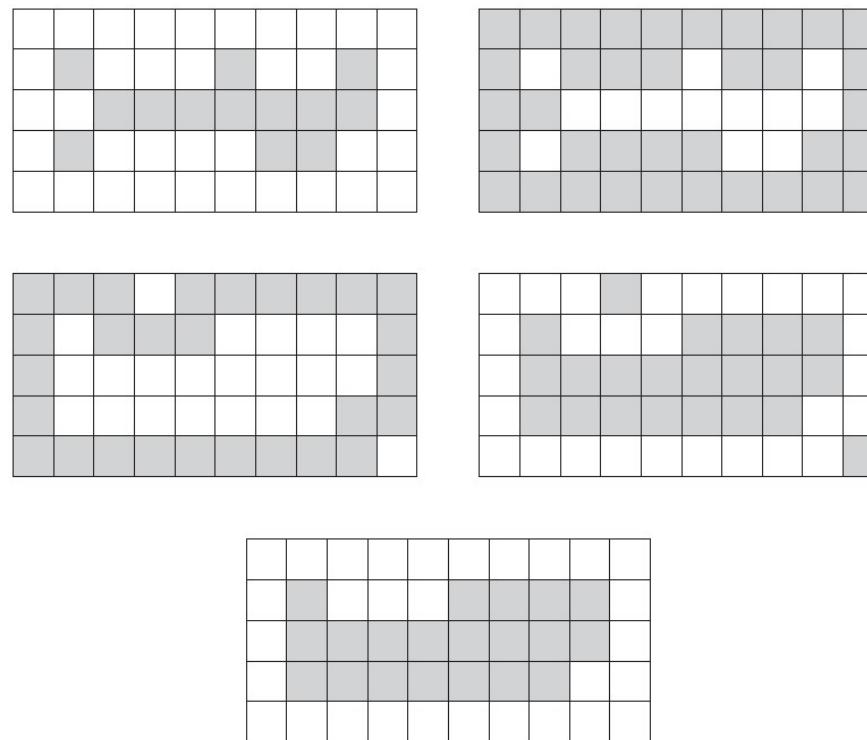


FIGURE 9.22 (a) Set A . (b) Complement of A . (c) Result of thinning the complement of A . (d) Thickened set obtained by complementing (c). (e) Final result, with no disconnected points.

10. Skeletons

- The skeleton of A can be expressed in terms of erosions and openings:

$$S(A) = \bigcup_{k=0}^K S_k(A) \quad \text{Successive erosions.}$$

$$S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$$

- $S(A)$ can be obtained as the union of the skeleton **subsets** $S_k(A)$.
- K is the last iterative step before A erodes to an empty set.

$$K = \max\{k | (A \ominus kB) \neq \emptyset\}$$

10. Skeletons

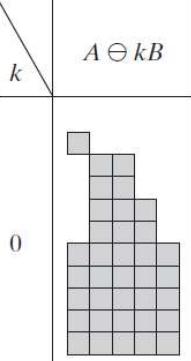
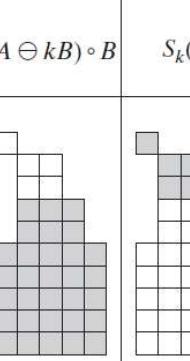
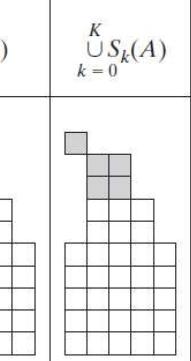
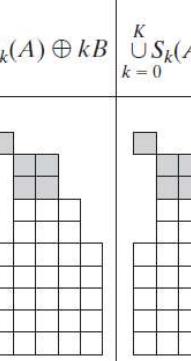
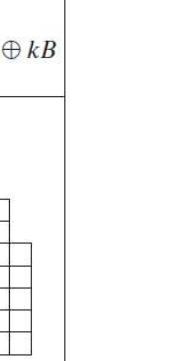
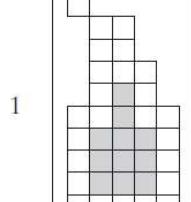
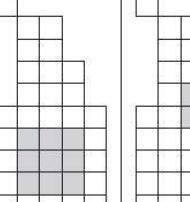
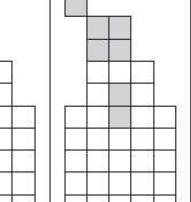
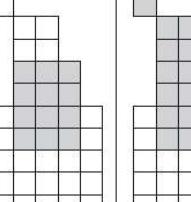
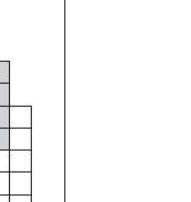
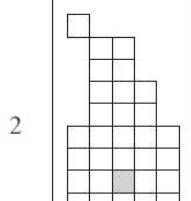
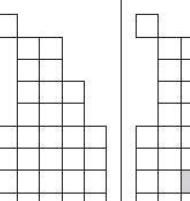
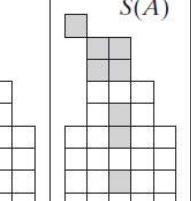
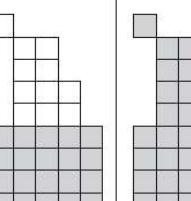
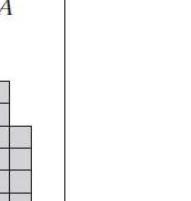
- Example:

$$K = \max\{k | (A \ominus kB) \neq \emptyset\}$$

$$S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$$

$$S(A) = \bigcup_{k=0}^K S_k(A)$$

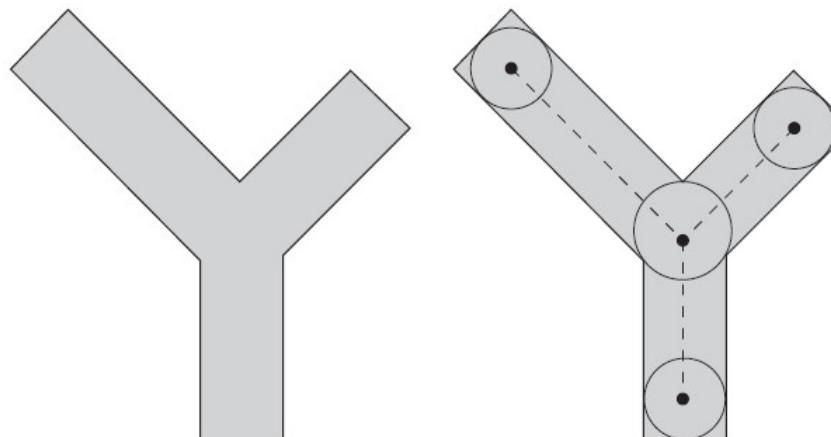
$$A = \bigcup_{k=0}^K (S_k(A) \oplus kB)$$

$k \backslash$	$A \ominus kB$	$(A \ominus kB) \circ B$	$S_k(A)$	$\bigcup_{k=0}^K S_k(A)$	$S_k(A) \oplus kB$	$\bigcup_{k=0}^K S_k(A) \oplus kB$
0						
1						
2						

B

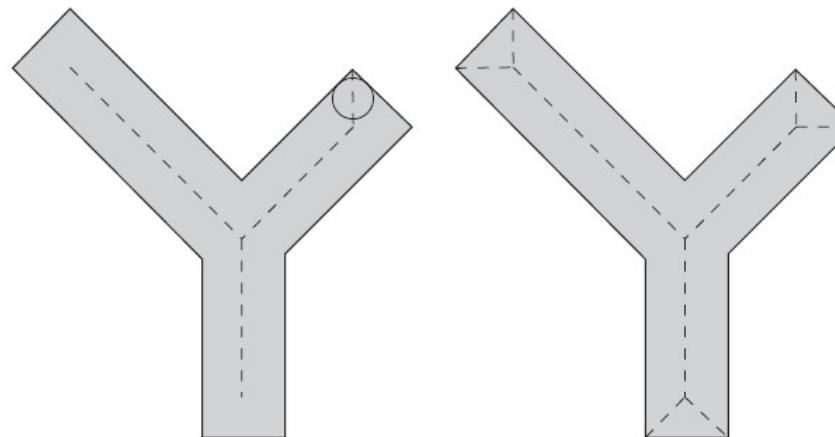

10. Skeletons

- Intuition
 - If z is a point of $S(A)$ and $(D)_z$ is the **largest disk** centered at z and contained in A , one cannot find a larger disk (not necessarily centered at z) containing $(D)_z$ and included in A . The disk is called **a maximum disk**.
 - The disk $(D)_z$ touches the boundary of A at two or more different places.



10. Skeletons

- Intuition
 - If z is a point of $S(A)$ and $(D)_z$ is the **largest disk** centered at z and contained in A , one cannot find a larger disk (not necessarily centered at z) containing $(D)_z$ and included in A . The disk is called **a maximum disk**.
 - The disk $(D)_z$ touches the boundary of A at two or more different places.



10. Skeletons

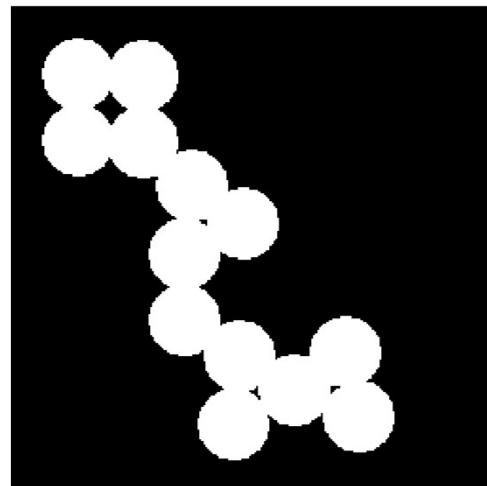
- Example:

```
clear all
close all
clc
BW1 = imread('circles.png');
imshow(BW1);
BW2 = bwmorph(BW1, 'thin', Inf);
figure
imshow(BW2)
BW3 = bwmorph(BW1, 'skel', Inf);
figure
imshow(BW3)
```

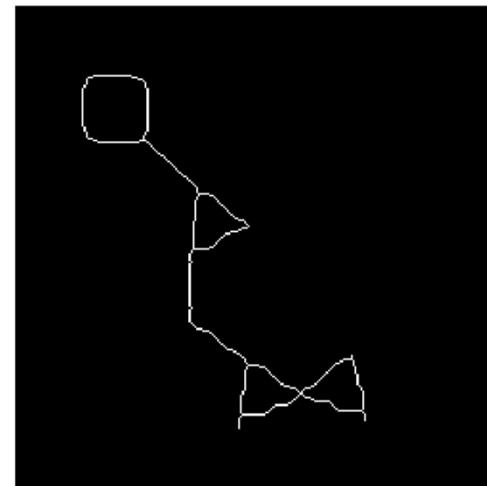
10. Skeletons

- Example:

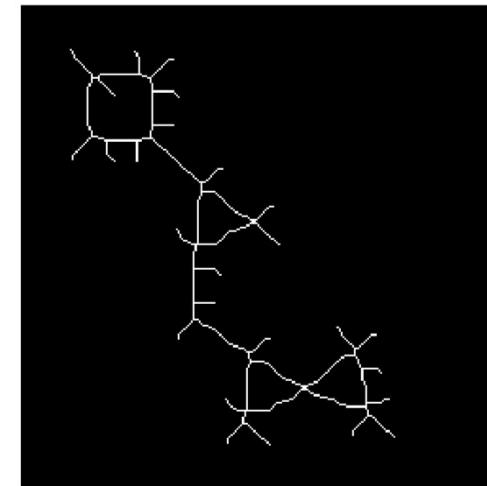
Original



Thinning



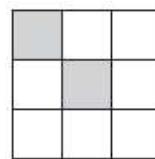
Skeletonization



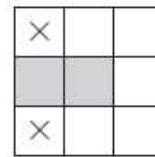
11. Pruning

- **Pruning** methods are an essential **complement to thinning and skeletonizing** algorithms because these procedures tend to leave parasitic components that need to be “**cleaned up**” by postprocessing.
- Thinning of an input set with a sequence B of structuring elements designed to detect only end points achieves the desired result.
- First we calculate,

$$X_1 = A \otimes \{B\}$$



B^5, B^6, B^7, B^8 (rotated 90°)



B^1, B^2, B^3, B^4 (rotated 90°)

11. Pruning

- The next step is to “restore” the character to its original form. To do so first we form a set containing all end points in X_1 .

$$X_2 = \bigcup_{k=1}^8 (X_1 \circledast B^k)$$

- Next we dilate the end points (3 times, for instance), using set A as a delimiter.

$$X_3 = (X_2 \oplus H) \cap A$$

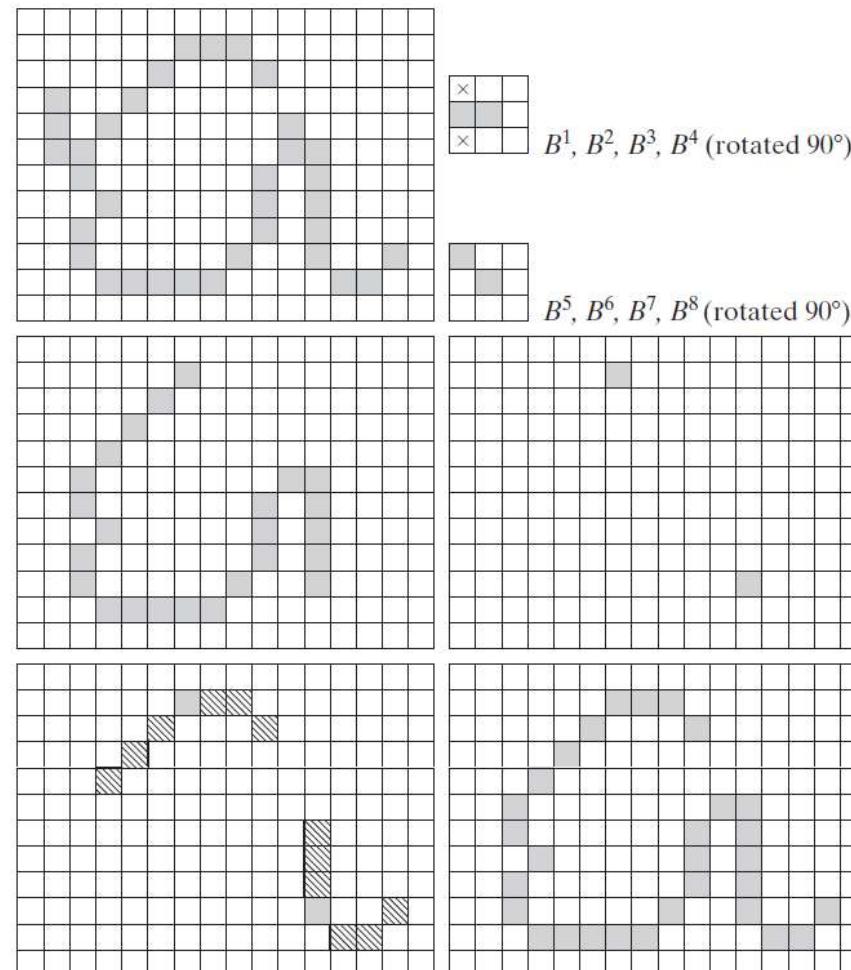
where H is a 3x3 structuring element of 1s and the intersection with A is applied after each step.

- Finally, the union of X_1 and X_3 yields the desired result,

$$X_4 = X_1 \cup X_3$$

11. Pruning

- Example:

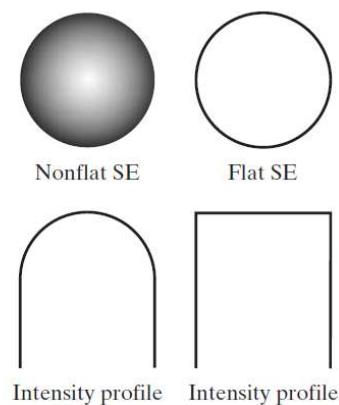


a b
c
d e
f g

FIGURE 9.25
(a) Original image. (b) and (c) Structuring elements used for deleting end points. (d) Result of three cycles of thinning. (e) End points of (d). (f) Dilation of end points conditioned on (a). (g) Pruned image.

12. Gray-Scale Morphology

- In this section, we extend to gray-scale images the basic operations of dilation, erosion, opening, and closing.
- Structuring elements in gray-scale morphology belong to one of two categories: **nonflat** and **flat**.
- Gray-scale SEs are used infrequently in practice.
- All the examples are based on symmetrical, flat structuring elements of unit height whose origins are at the center.



12. Gray-Scale Morphology

- Erosion

- The *erosion* of f by a flat structuring element b at any location (x, y) is defined as the **minimum value** of the image in the region coincident with b when the origin of b is at (x, y) ,

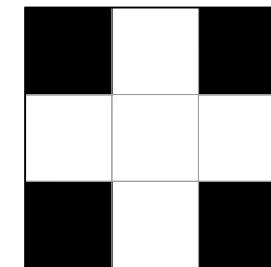
$$[f \ominus b](x, y) = \min_{(s, t) \in b} \{f(x + s, y + t)\}$$

- In general an eroded gray-scale image will be darker than the original.
- The sizes of **bright** features will be **reduced**, and that the sizes of **dark** features will be **increased**

12. Gray-Scale Morphology

- Erosion (flat SE)

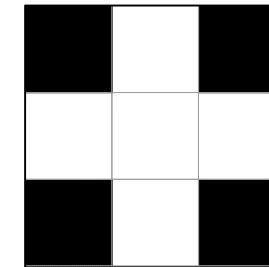
0	0	0	0	0	0	0	100	0	0
0	0	0	0	0	0	50	40	30	0
0	0	0	60	130	0	0	50	0	0
0	0	0	90	120	175	0	0	0	0
0	0	0	0	130	0	175	0	0	0
0	0	90	0	0	120	220	230	0	0
0	0	0	50	0	0	100	0	0	0
0	0	60	5	70	0	0	0	0	0
0	0	0	120	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



12. Gray-Scale Morphology

- Erosion (flat SE)

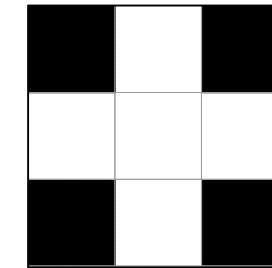
0	0	0	0	0	0	0	100	0	0
0	0	0	0	0	0	50	40	30	0
0	0	0	60	130	0	0	50	0	0
0	0	0	90	120	175	0	0	0	0
0	0	0	0	130	0	175	0	0	0
0	0	90	0	0	120	220	230	0	0
0	0	0	50	0	0	100	0	0	0
0	0	60	5	70	0	0	0	0	0
0	0	0	120	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



12. Gray-Scale Morphology

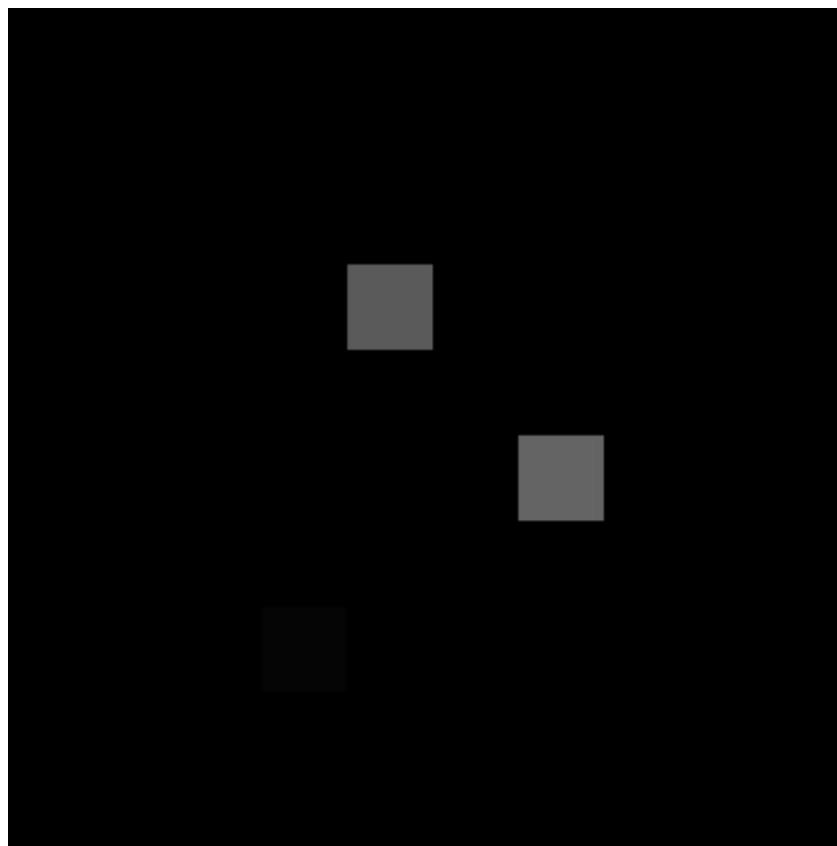
- Erosion (flat SE)

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	30	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	90	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	5	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



12. Gray-Scale Morphology

- Erosion (flat SE)



12. Gray-Scale Morphology

- Dilation (flat SE)
 - Similarly, the *dilation* of f by a flat structuring element b at any location (x, y) is defined as the maximum value of the image in the window outlined by \hat{b} when the origin of \hat{b} is at (x, y) .

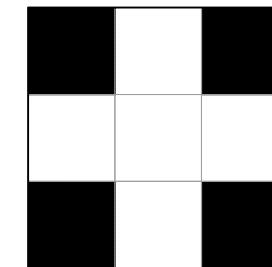
$$[f \oplus b](x, y) = \max_{(s, t) \in b} \{f(x - s, y - t)\}$$

- The **bright** features were **thickened** and the intensities of the **dark** features were **reduced**.

12. Gray-Scale Morphology

- Dilation (flat SE)

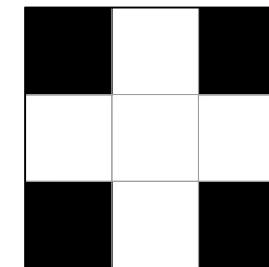
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	40	0	0
0	0	0	60	0	0	0	0	0	0	0
0	0	0	90	120	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	220	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0



12. Gray-Scale Morphology

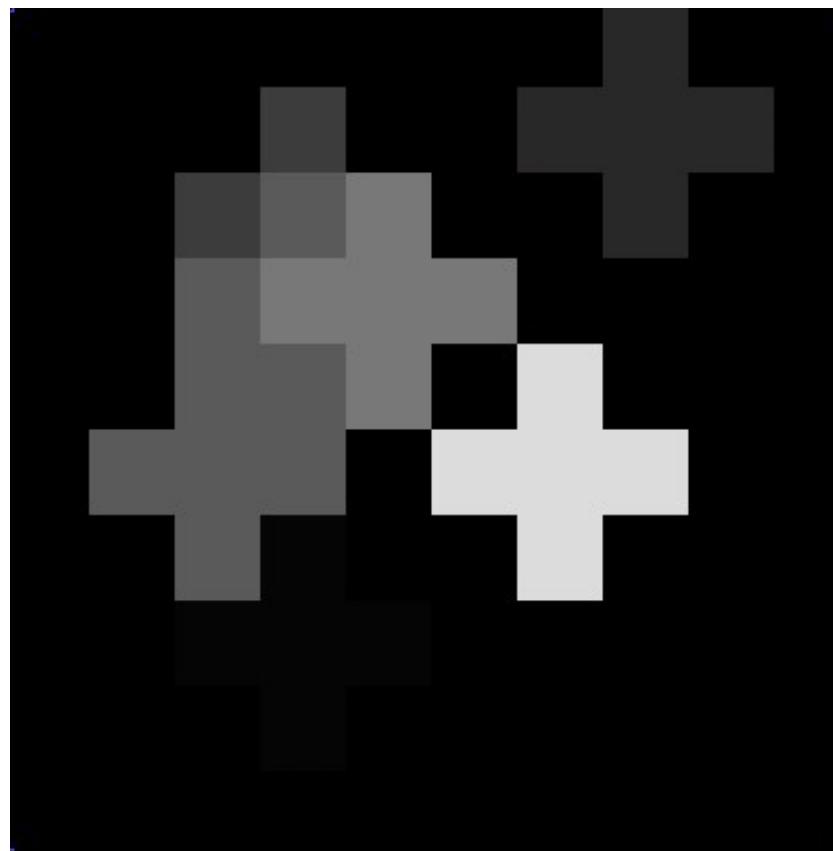
- Dilation (flat SE)

0	0	0	0	0	0	0	40	0	0
0	0	0	60	0	0	40	40	40	0
0	0	60	90	120	0	0	40	0	0
0	0	90	120	120	120	0	0	0	0
0	0	90	90	120	0	220	0	0	0
0	90	90	90	0	220	220	220	0	0
0	0	90	5	0	0	220	0	0	0
0	0	5	5	5	0	0	0	0	0
0	0	0	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



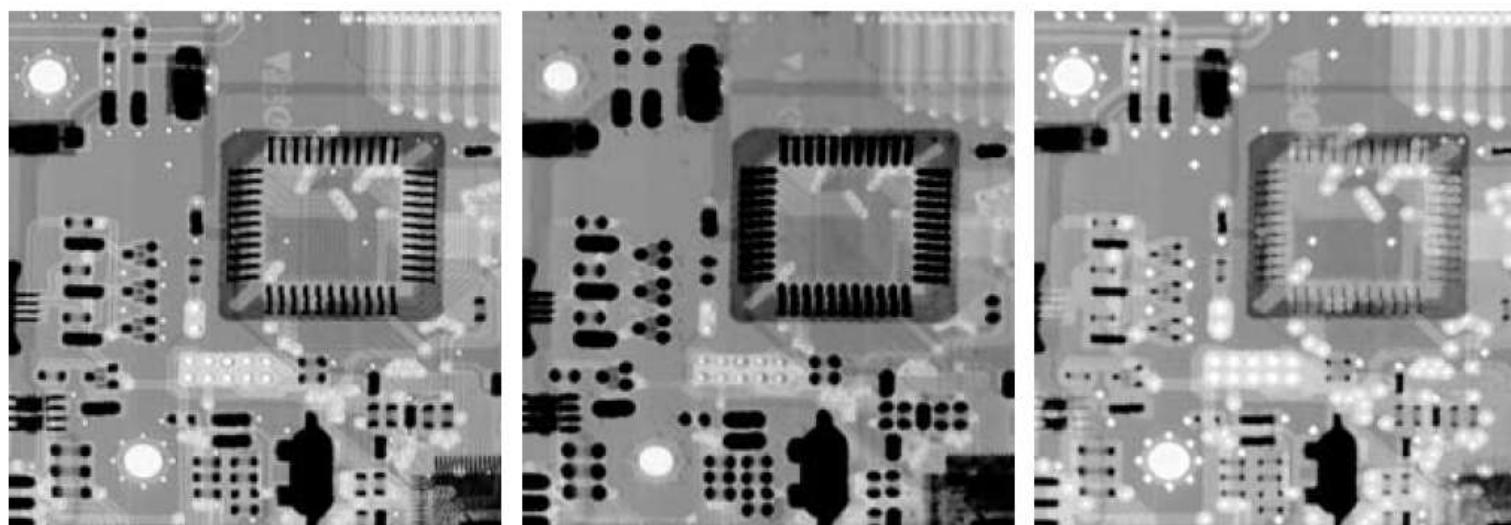
12. Gray-Scale Morphology

- Dilation (flat SE)



12. Gray-Scale Morphology

- Erosion and dilation example (flat SE)



a b c

FIGURE 9.35 (a) A gray-scale X-ray image of size 448×425 pixels. (b) Erosion using a flat disk SE with a radius of two pixels. (c) Dilation using the same SE. (Original image courtesy of Lixi, Inc.)

12. Gray-Scale Morphology

- Erosion and dilation (nonflat SE)
 - The *erosion* of image f by nonflat structuring element b_N , is defined as,

$$[f \ominus b_N](x, y) = \min_{(s, t) \in b_N} \{f(x + s, y + t) - b_N(s, t)\}$$

- The *dilation* using a nonflat SE is defined as,

$$[f \oplus b_N](x, y) = \max_{(s, t) \in b_N} \{f(x - s, y - t) + b_N(s, t)\}$$

12. Gray-Scale Morphology

- Erosion and dilation (nonflat SE)
 - Problems:
 - ✓ Erosion and dilation using a nonflat SE are not bounded in general by the values of f , which can present **problems in interpreting results**.
 - ✓ Potential difficulties in selecting meaningful elements for b_N .
 - ✓ Computational burden.

12. Gray-Scale Morphology

- Erosion and dilation (nonflat SE)
 - Erosion and dilation are duals with respect to function complementation and reflection;

$$(f \ominus b)^c = (f^c \oplus \hat{b})$$

$$(f \oplus b)^c = (f^c \ominus \hat{b})$$

$$f^c = -f(x, y)$$

$$\hat{b} = b(-x, -y)$$

12. Gray-Scale Morphology

- Opening and Closing

- The opening of image f by structuring element b is

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

- Similarly, the closing of f by b is

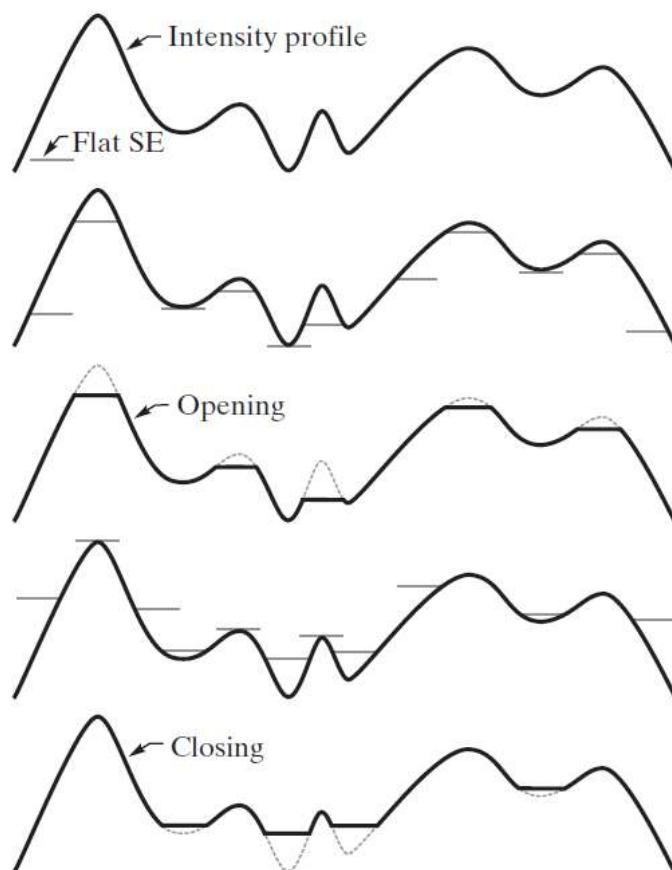
$$f \bullet b = (f \oplus b) \ominus b$$

- The opening and closing for gray-scale images are duals with respect to complementation and SE reflection:

$$(f \bullet b)^c = f^c \circ \hat{b} \quad (f \circ b)^c = f^c \bullet \hat{b}$$

12. Gray-Scale Morphology

- Opening and Closing

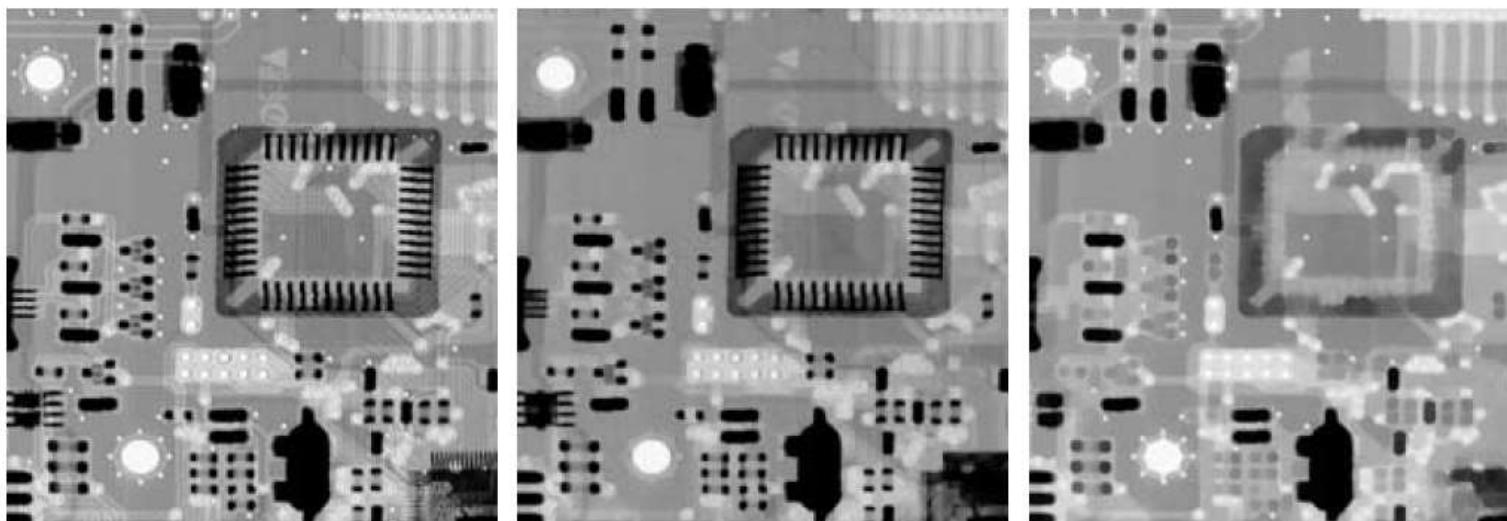


a
b
c
d
e

FIGURE 9.36
Opening and closing in one dimension. (a) Original 1-D signal. (b) Flat structuring element pushed up underneath the signal. (c) Opening. (d) Flat structuring element pushed down along the top of the signal. (e) Closing.

12. Gray-Scale Morphology

- Opening and Closing



a b c

FIGURE 9.37 (a) A gray-scale X-ray image of size 448×425 pixels. (b) Opening using a disk SE with a radius of 3 pixels. (c) Closing using an SE of radius 5.

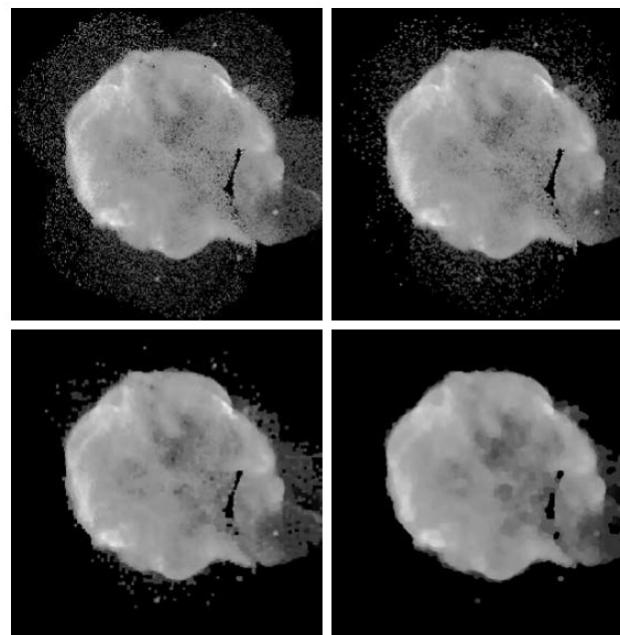
12. Gray-Scale Morphology

- Morphological smoothing

➤ *Alternating sequential filtering*: an opening–closing sequence starts with the original image, but subsequent steps perform the opening and closing on the results.

a
b
c
d

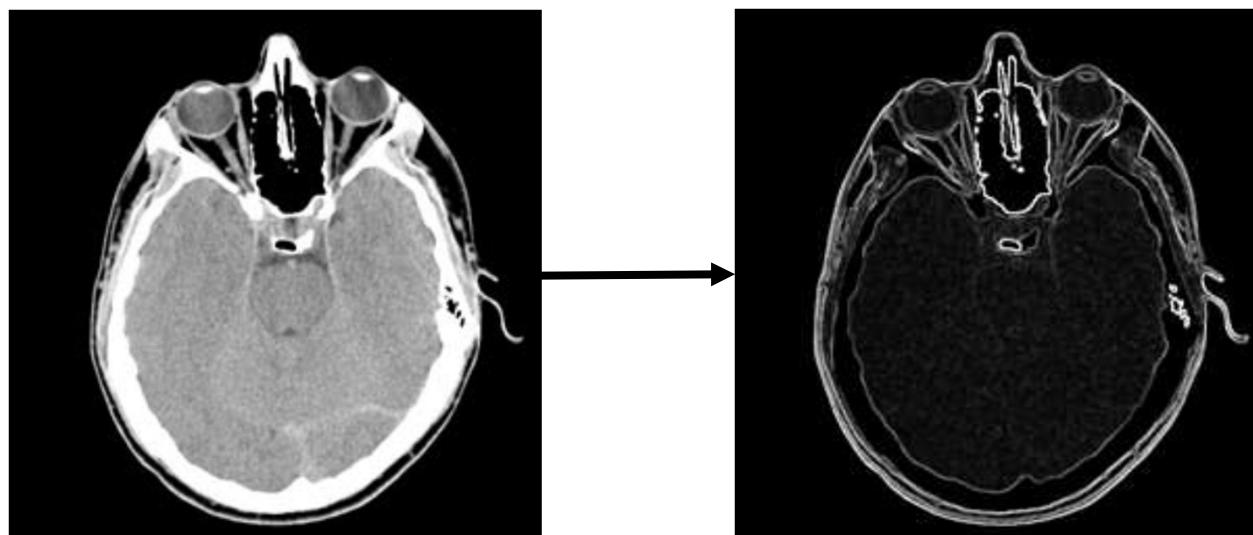
FIGURE 9.38
(a) 566×566 image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope.
(b)–(d) Results of performing opening and closing sequences on the original image with disk structuring elements of radii, 1, 3, and 5, respectively.
(Original image courtesy of NASA.)



12. Gray-Scale Morphology

- Morphological gradient
 - Dilation and erosion can be used in combination with image subtraction to obtain the morphological gradient of an image,

$$g = (f \oplus b) - (f \ominus b)$$



12. Gray-Scale Morphology

- Top-hat and bottom-hat transformations
 - An important use of top-hat transformations is in correcting the effects of nonuniform illumination.
 - Proper illumination plays a central role in the process of extracting objects from the background.
 - The *top-hat transformation* of a grayscale image f is defined as f minus its *opening* (used for light objects on a dark background),

$$T_{\text{hat}}(f) = f - (f \circ b)$$

- The *bottom-hat transformation* f of f is defined as the closing of f minus f (used for the converse),

$$B_{\text{hat}}(f) = (f \bullet b) - f$$

12. Gray-Scale Morphology

- Top-hat and bottom-hat transformations

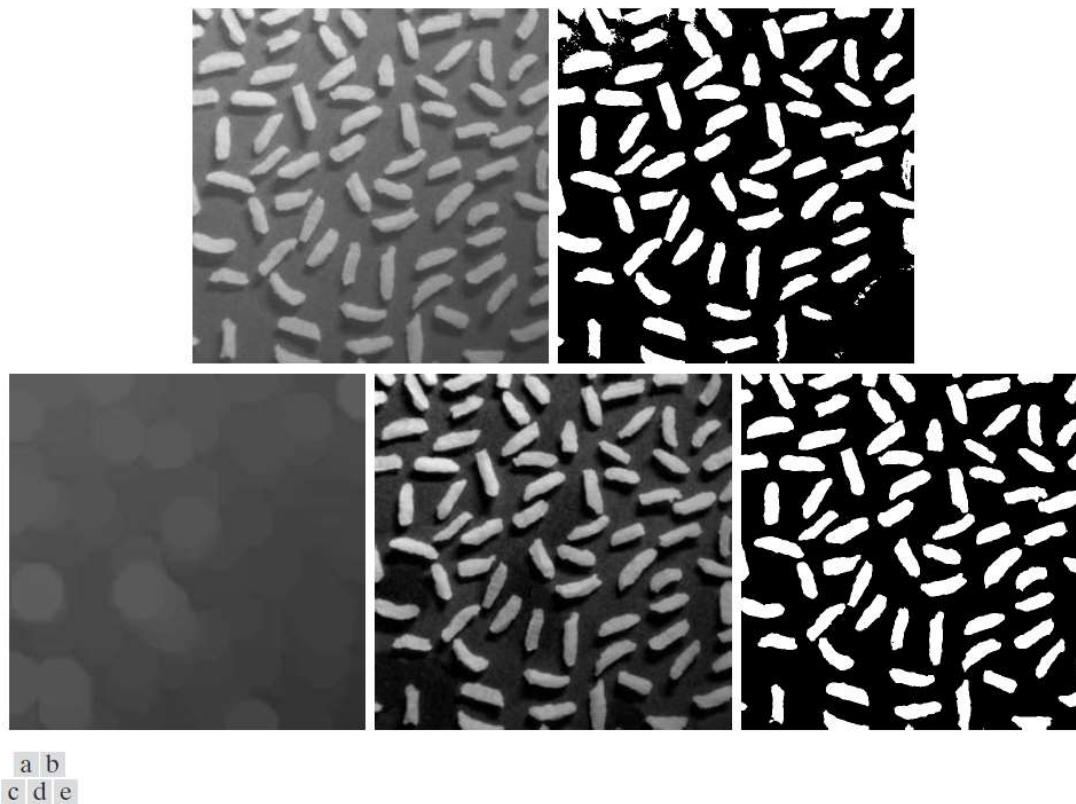


FIGURE 9.40 Using the top-hat transformation for *shading correction*. (a) Original image of size 600×600 pixels. (b) Thresholded image. (c) Image opened using a disk SE of radius 40. (d) Top-hat transformation (the image minus its opening). (e) Thresholded top-hat image.

12. Gray-Scale Morphology

- Top-hat and bottom-hat transformations

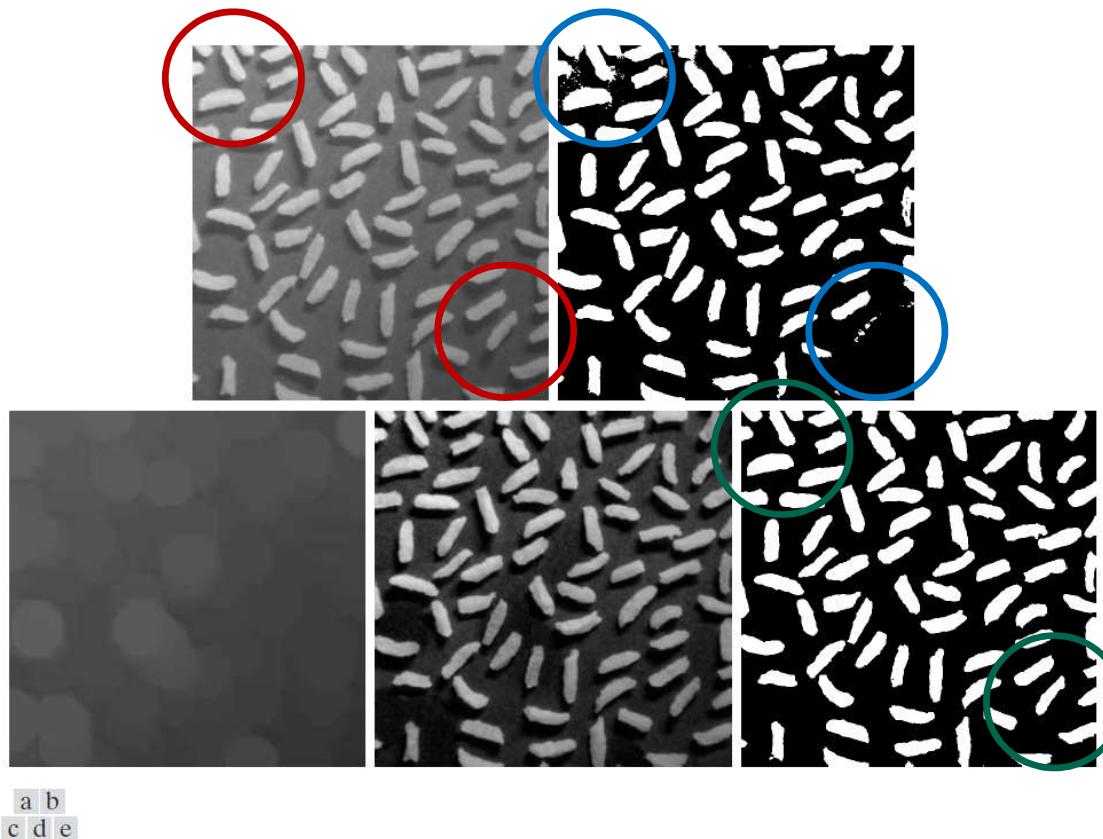


FIGURE 9.40 Using the top-hat transformation for *shading correction*. (a) Original image of size 600×600 pixels. (b) Thresholded image. (c) Image opened using a disk SE of radius 40. (d) Top-hat transformation (the image minus its opening). (e) Thresholded top-hat image.

12. Gray-Scale Morphology

- Granulometry

- In terms of image processing, *granulometry* is a field that deals with determining the size distribution of particles in an image.
- In practice, particles seldom are neatly separated, which makes particle counting by identifying individual particles a difficult task.
- Morphology can be used to estimate particle size distribution indirectly, without having to identify and measure every particle in the image.

12. Gray-Scale Morphology

- Granulometry
 - With particles having regular shapes that are lighter than the background, the method consists of applying openings with SEs of increasing size.
 - For each opening, the sum of the pixel values in the opening is computed.
 - This sum, sometimes called the *surface area*, decreases as a function of increasing SE size because.
 - This procedure yields a 1-D array of such numbers.
 - To emphasize changes between successive openings, we compute the difference between adjacent elements of the 1-D array.

12. Gray-Scale Morphology

- Granulometry

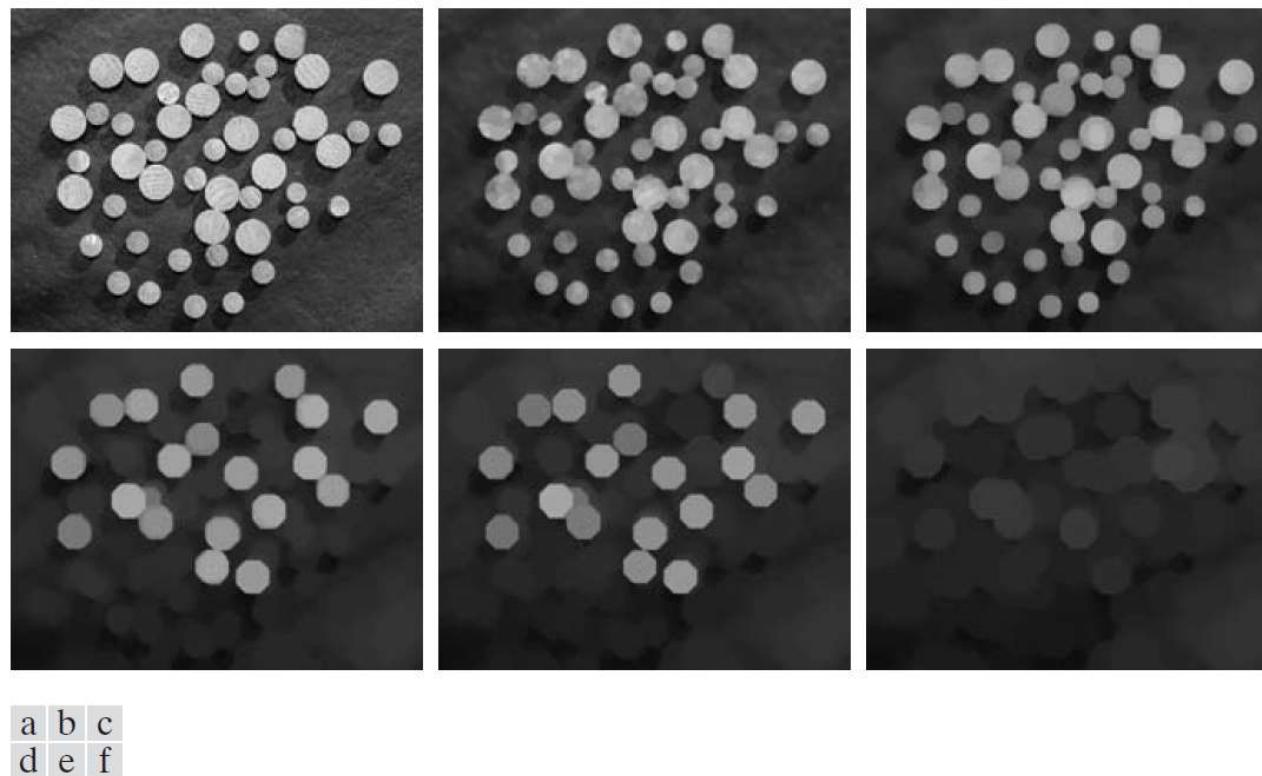
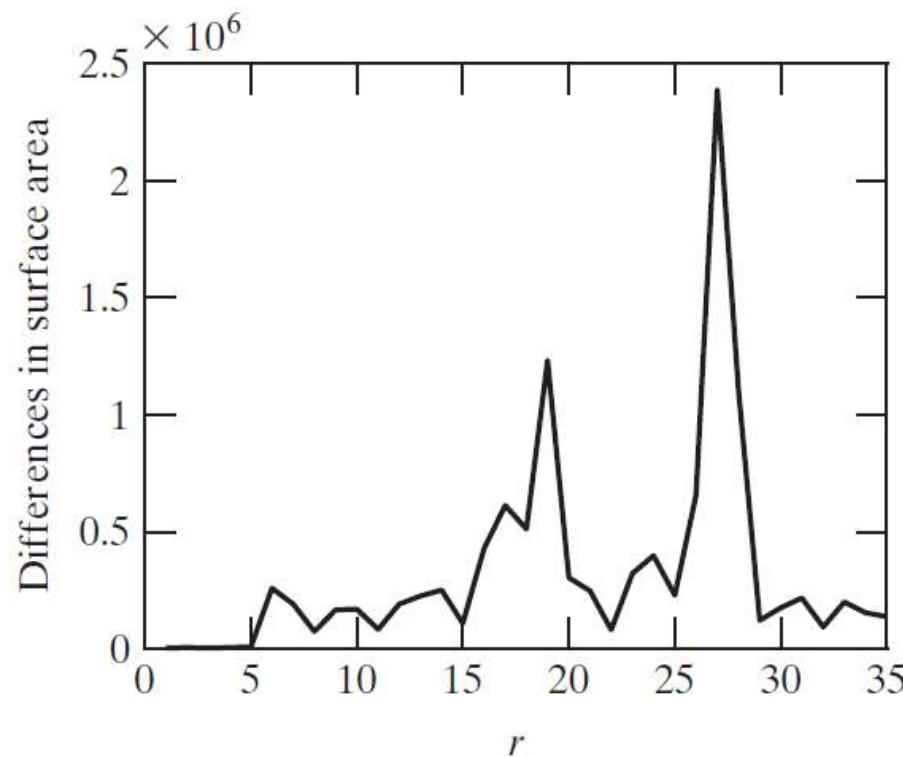


FIGURE 9.41 (a) 531×675 image of wood dowels. (b) Smoothed image. (c)–(f) Openings of (b) with disks of radii equal to 10, 20, 25, and 30 pixels, respectively. (Original image courtesy of Dr. Steve Eddins, The MathWorks, Inc.)

12. Gray-Scale Morphology

- Granulometry



12. Gray-Scale Morphology

- MATLAB: s129Granul.m



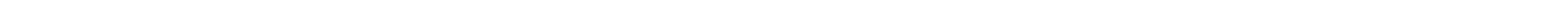
Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 06

Image Segmentation



1. Fundamentals

- Let R represent the **entire spatial region** occupied by an image. We may view image **segmentation** as a process that **partitions** R into n **subregions**, R_1, R_2, \dots, R_n , such that,

(a) $\bigcup_{i=1}^n R_i = R.$

(b) R_i is a connected set, $i = 1, 2, \dots, n.$

(c) $R_i \cap R_j = \emptyset$ for all i and j , $i \neq j.$

(d) $Q(R_i) = \text{TRUE}$ for $i = 1, 2, \dots, n.$

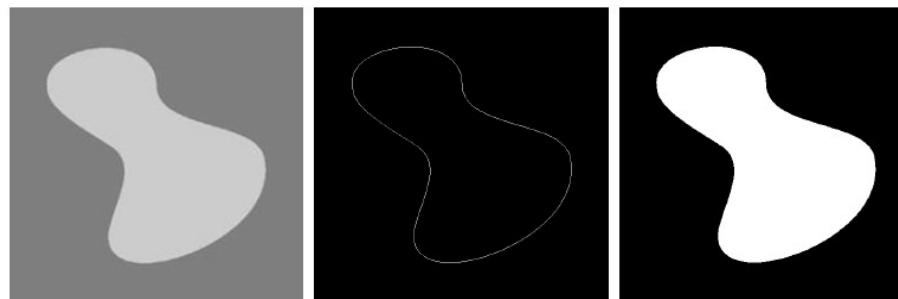
(e) $Q(R_i \cup R_j) = \text{FALSE}$ for any adjacent regions R_i and $R_j.$

1. Fundamentals

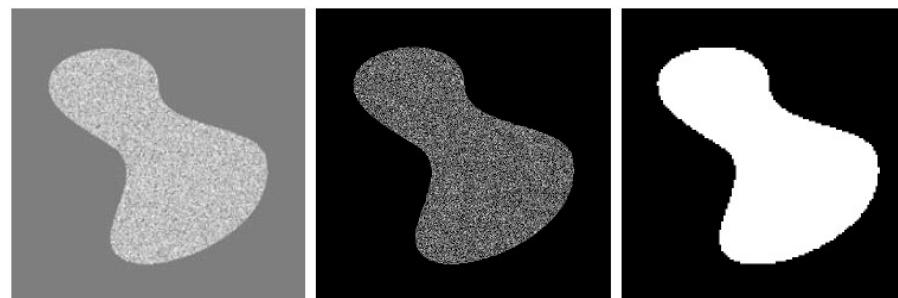
- Let R represent the **entire spatial region** occupied by an image. We may view image **segmentation** as a process that **partitions** R into n **subregions**, R_1, R_2, \dots, R_n , such that,
 - a. Every pixel must be in a region.
 - b. Points in a region must be connected in some predefined sense (e.g., 4- or 8-connected).
 - c. The regions must be disjoint.
 - d. Pixels in a segmented region must satisfy an specific property (e.g., all pixels must have the same intensity level).
 - e. Two adjacent regions must be different in the sense of predicate Q .

1. Fundamentals

- Segmentation algorithms for monochrome images generally are based on one of two basic categories:
 - **Edge-based** segmentation (based on discontinuity)

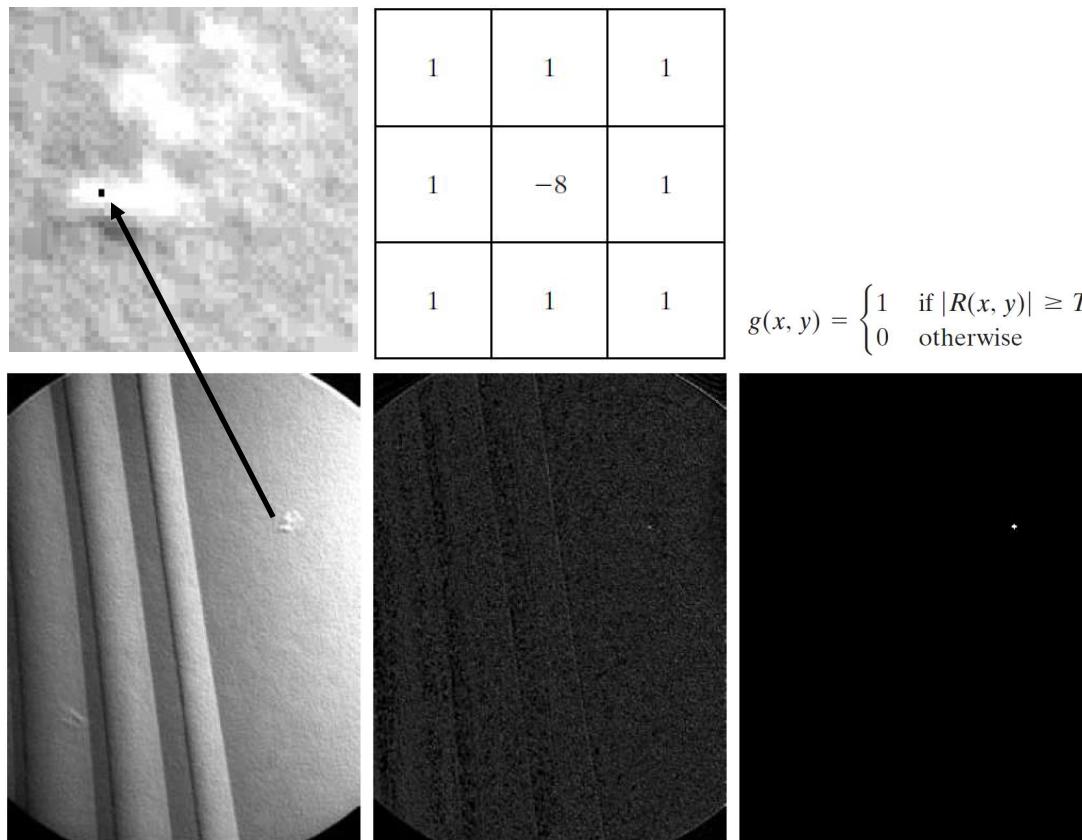


- **Region-based** segmentation (based on similarity)



2. Point, Line, and Edge Detection

- Detection of Isolated Points



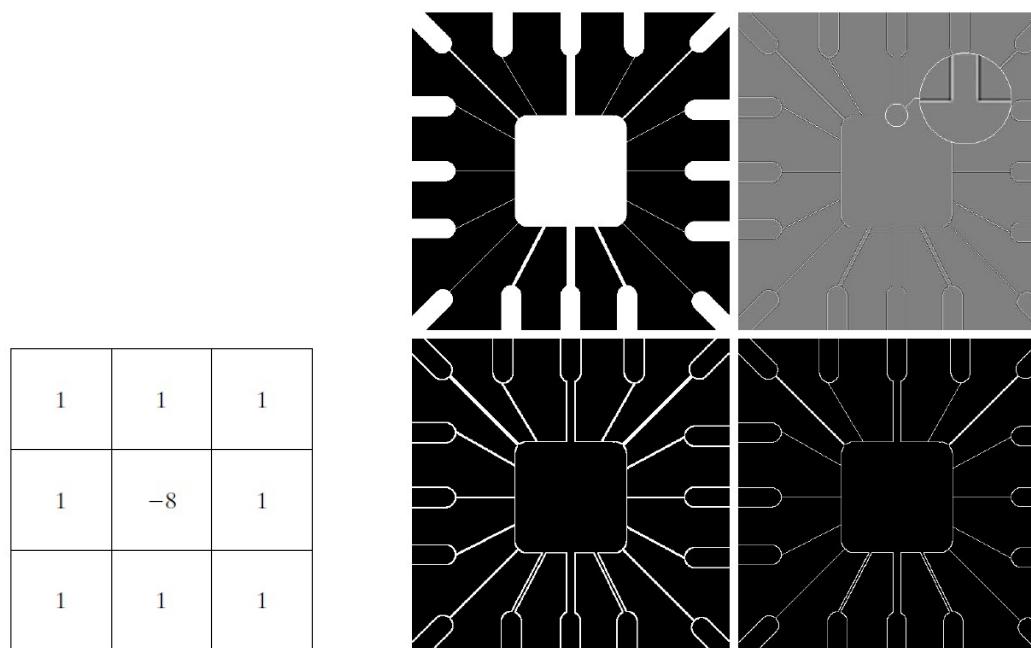
a
b c d

FIGURE 10.4
(a) Point detection (Laplacian) mask.
(b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel.
(c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

2. Point, Line, and Edge Detection

- Line Detection

- We can use the Laplacian mask for line detection also, keeping in mind that the double-line effect of the second derivative must be handled properly.



a b
c d

FIGURE 10.5

(a) Original image.
(b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian.
(c) Absolute value of the Laplacian.
(d) Positive values of the Laplacian.

2. Point, Line, and Edge Detection

- Line Detection

- The Laplacian detector used before is **isotropic**, so its response is **independent of direction**.
- Often, interest lies in detecting lines in **specified directions**.

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

Horizontal

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

+45°

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

Vertical

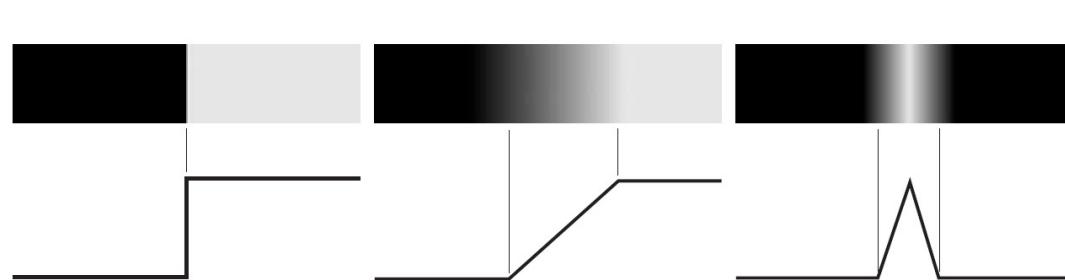
-1	-1	2
-1	2	-1
2	-1	-1

-45°

2. Point, Line, and Edge Detection

- Edge Models

- Edge detection is the approach used most frequently for segmenting images based on abrupt (local) changes in intensity.
- In practice, digital images have edges that are blurred and noisy.
- In such situations, edges are more closely modeled as having an intensity ramp profile.



a b c

FIGURE 10.8
From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models

- The models allow us to write mathematical expressions for edges in the development of image processing algorithms.

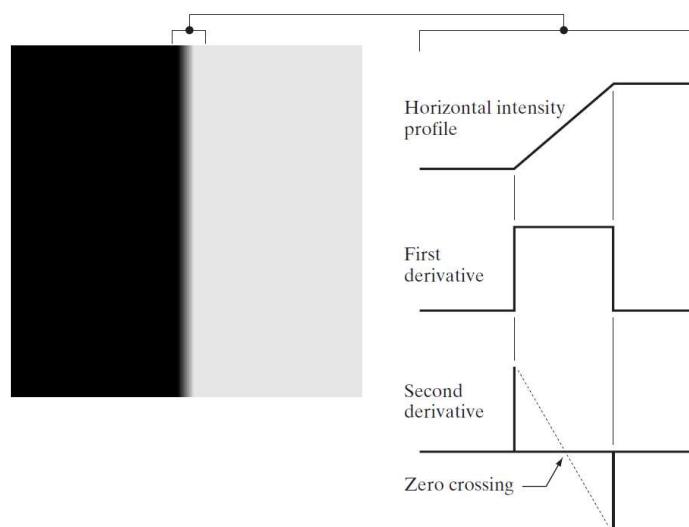


FIGURE 10.9 A 1508×1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and “step” profiles span 9 pixels and 2 pixels, respectively.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

2. Point, Line, and Edge Detection

- Edge Models

- The **magnitude** of the **first derivative** can be used to detect an edge at a point in an image.
- The **sign** of the **second derivative** can be used to determine whether an edge pixel lies on the dark or light side of an edge



2. Point, Line, and Edge Detection

- Edge Models
 - The effect of noise

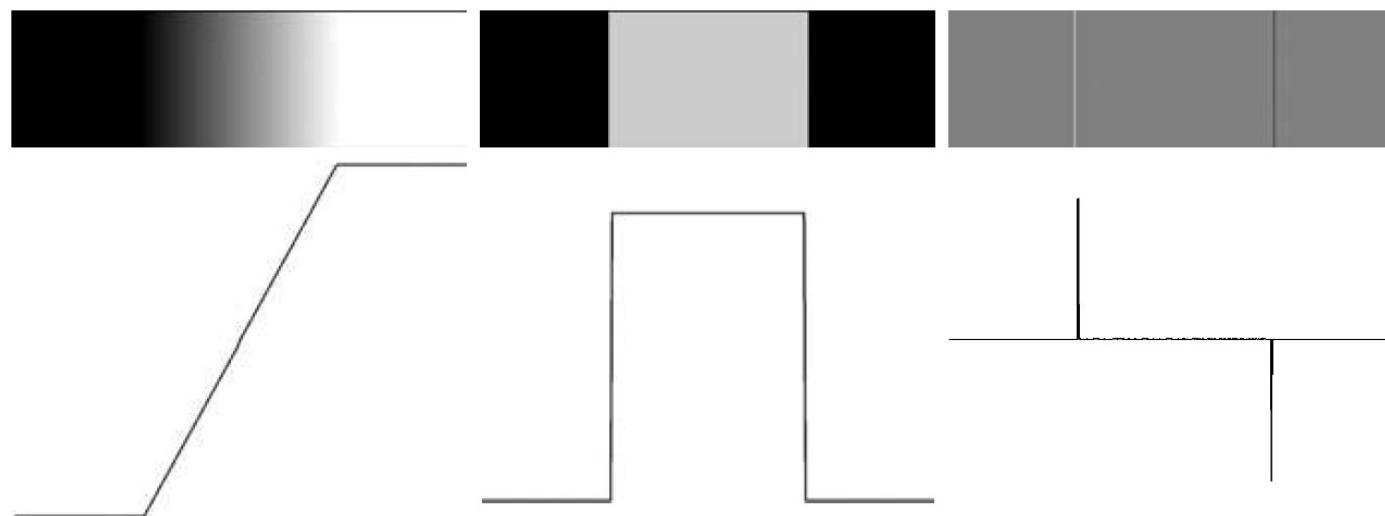


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models
 - The effect of noise

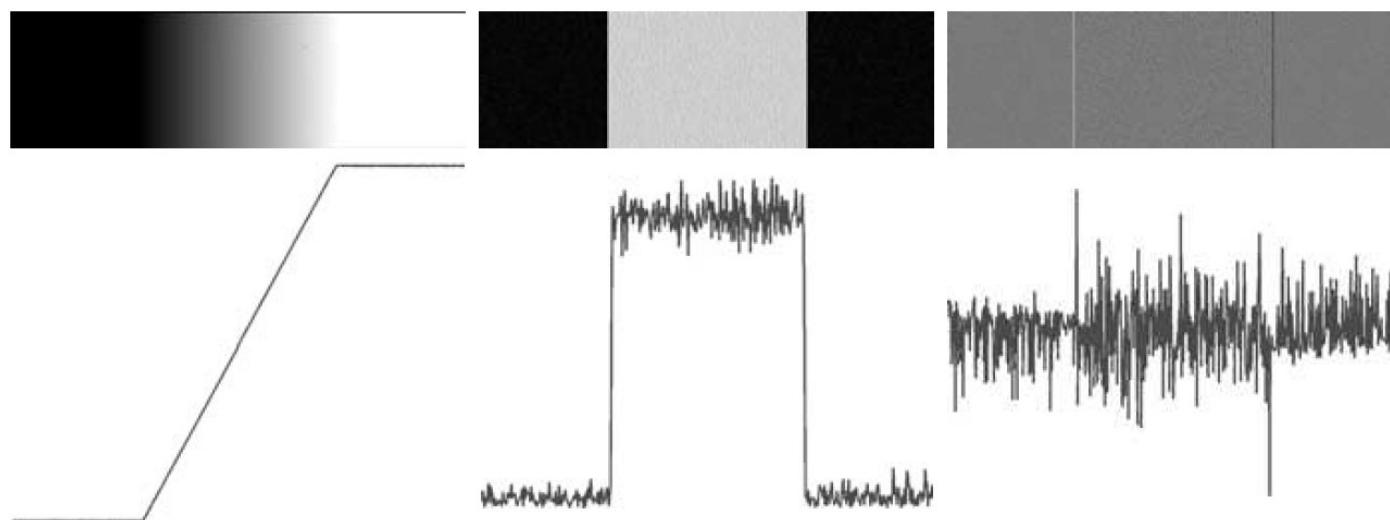


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models
 - The effect of noise

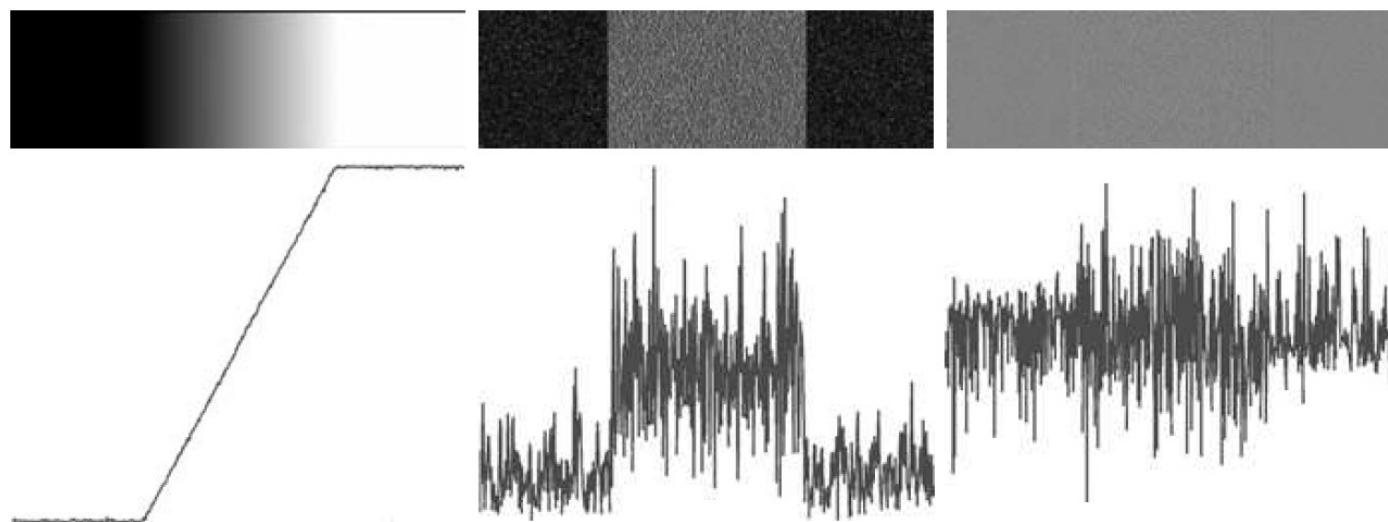


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models
 - The effect of noise

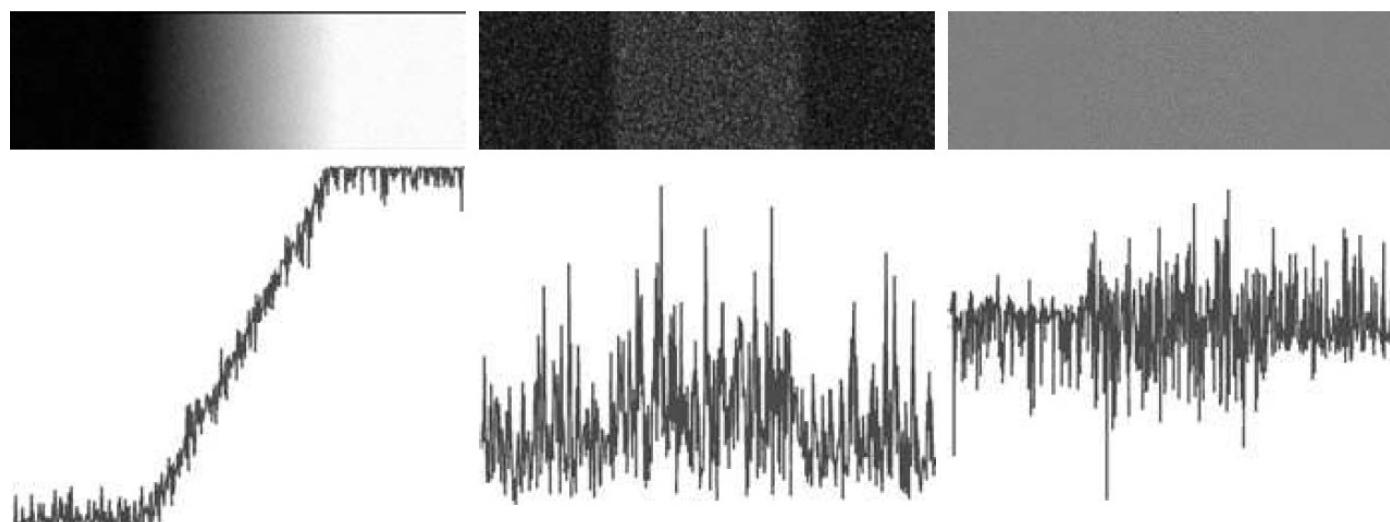


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

2. Point, Line, and Edge Detection

- Edge Models

- We conclude this section by noting that there are three fundamental steps performed in edge detection:
 1. Image smoothing for noise reduction.
 2. Detection of edge points. As mentioned earlier, this is a local operation that extracts from an image all points that are potential candidates to become edge points.
 3. Edge localization. Select from the candidate edge points only the points that are true members of the set of points comprising an edge.

2. Point, Line, and Edge Detection

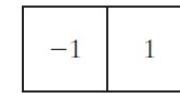
- Basic Edge Detection
 - Gradient operators

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

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

a | b

FIGURE 10.13
One-dimensional
masks used to
implement Eqs.
(10.2-12) and
(10.2-13).



2. Point, Line, and Edge Detection

- Basic Edge Detection
 - Gradient operators

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

-1	0		0	-1
0	1		1	0

Roberts

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

2. Point, Line, and Edge Detection

- Basic Edge Detection

- Gradient operators

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

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

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

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Prewitt

2. Point, Line, and Edge Detection

- Basic Edge Detection
 - Gradient operators

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

-1	-2	-1	-1	0	1	z_1	z_2	z_3
0	0	0	-2	0	2	z_4	z_5	z_6
1	2	1	-1	0	1	z_7	z_8	z_9

Sobel

2. Point, Line, and Edge Detection

- Basic Edge Detection
 - Gradient operators

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

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

Prewitt

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

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

Sobel

a	b
c	d

FIGURE 10.15
Prewitt and Sobel
masks for
detecting diagonal
edges.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

2. Point, Line, and Edge Detection

- Basic Edge Detection



a b
c d

FIGURE 10.16

(a) Original image of size 834×1114 pixels, with intensity values scaled to the range $[0, 1]$.
(b) $|g_x|$, the component of the gradient in the x -direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image.
(c) $|g_y|$, obtained using the mask in Fig. 10.14(g).
(d) The gradient image, $|g_x| + |g_y|$.

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

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

2. Point, Line, and Edge Detection

- Basic Edge Detection



a b
c d

FIGURE 10.18
Same sequence as
in Fig. 10.16, but
with the original
image smoothed
using a 5×5
averaging filter
prior to edge
detection.

2. Point, Line, and Edge Detection

- Basic Edge Detection



a b

FIGURE 10.19
Diagonal edge detection.
(a) Result of using the mask in Fig. 10.15(c).
(b) Result of using the mask in Fig. 10.15(d). The input image in both cases was Fig. 10.18(a).

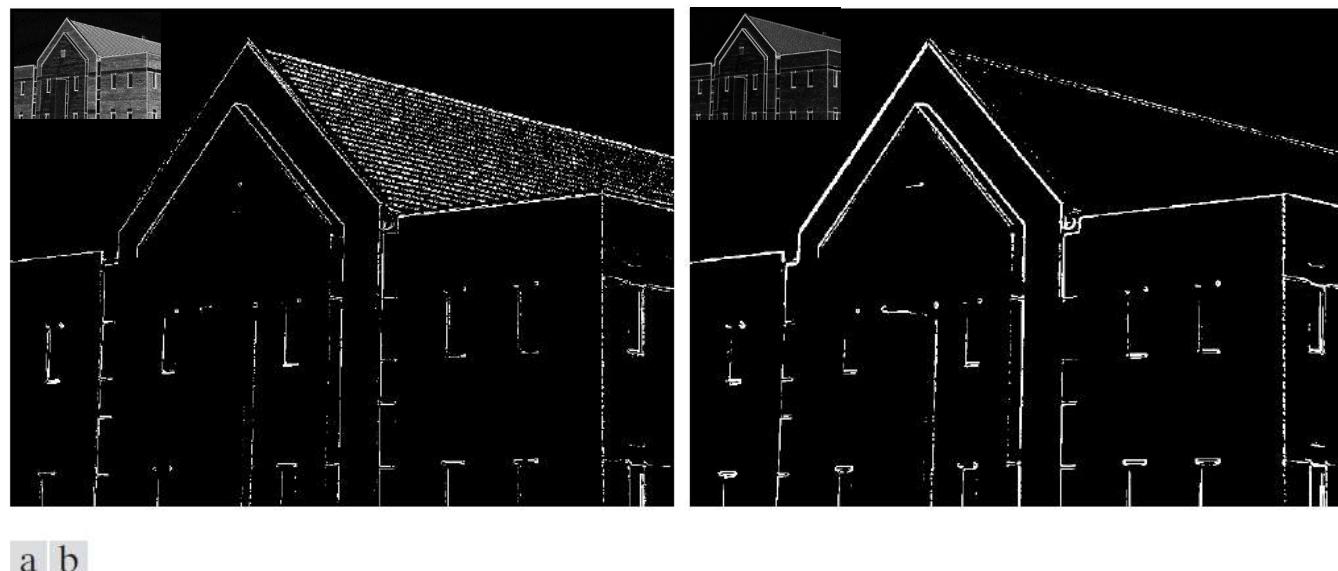


0	1	2
-1	0	1
-2	-1	0

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

2. Point, Line, and Edge Detection

- Basic Edge Detection
 - Combining the gradient with thresholding



a b

FIGURE 10.20 (a) Thresholded version of the image in Fig. 10.16(d), with the threshold selected as 33% of the highest value in the image; this threshold was just high enough to eliminate most of the brick edges in the gradient image. (b) Thresholded version of the image in Fig. 10.18(d), obtained using a threshold equal to 33% of the highest value in that image.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ One of the earliest successful attempts at incorporating **more sophisticated analysis** into the edge-finding process is attributed to Marr and Hildreth.
 - ✓ They argued that:
 - a. Intensity changes are not independent of image scale;
 - b. A sudden intensity change will give rise to a peak or trough in the first derivative or, equivalently, to a zero crossing in the second derivative.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ These ideas suggest that an operator used for edge detection should have two salient features:
 - a. It should be an operator capable of computing a digital approximation of the first or second derivative; and
 - b. It should be capable of being “tuned”: large operators can be used to detect blurry edges and small operators to detect fine details.
 - ✓ Marr and Hildreth argued that the most satisfactory operator fulfilling these conditions is the **Laplacian of the Gaussian** filter $\nabla^2 G$ (LoG).

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

➤ The Marr-Hildreth edge detector

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

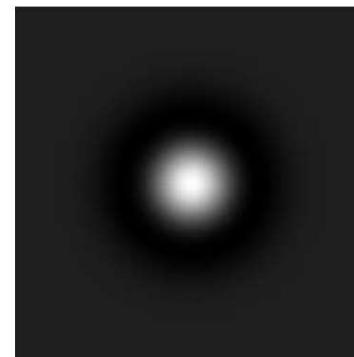
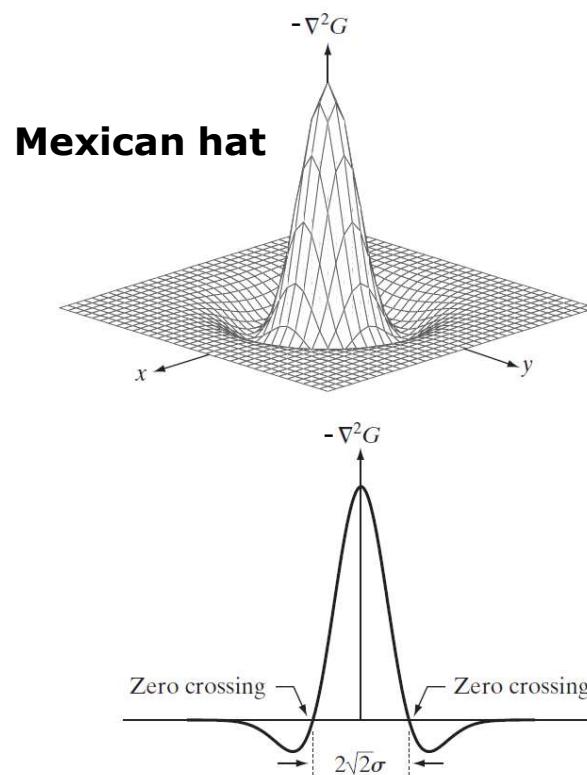
$$\boxed{\frac{d}{dx}e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}}}$$

$$\begin{aligned}\nabla^2 G(x, y) &= \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \\ &= \frac{\partial}{\partial x} \left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right] + \frac{\partial}{\partial y} \left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right] \\ &= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}\end{aligned}$$

$$\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector



a	b
c	d

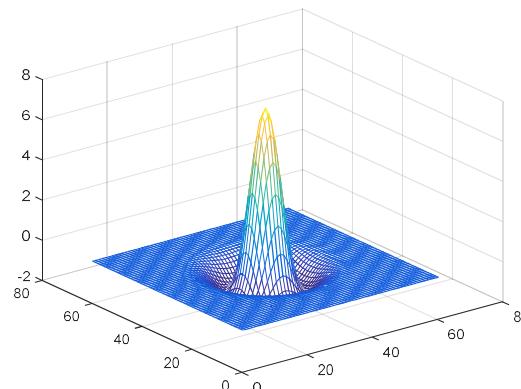
FIGURE 10.21

(a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5×5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ A more effective approach for generating a LoG filter is to **sample the Gaussian** to the desired $n \times n$ size and then **convolve** the resulting array with a **Laplacian mask**.
 - ✓ MATLAB: s29MarrHildFilters.m



2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ The Marr-Hildreth algorithm consists of convolving the LoG filter with an input image, $f(x, y)$,
$$g(x, y) = [\nabla^2 G(x, y)] \star f(x, y)$$

and then finding the zero crossings of $g(x, y)$.

 - ✓ Because these are linear processes,
$$g(x, y) = \nabla^2 [G(x, y) \star f(x, y)]$$

indicating that we can **smooth** the image **first** with a **Gaussian filter** and then compute the **Laplacian** of the result.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ The Marr-Hildreth edge-detection algorithm may be summarized as follows:
 1. Filter the input image with an Gaussian lowpass filter obtained by sampling $G(x, y)$.
 2. Compute the Laplacian of the image resulting from Step 1.
 3. Find the zero crossings of the image from Step 2.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ Recall that about 99.7% of the volume under a 2-D Gaussian lies between $\pm 3\sigma$ about the mean.
 - ✓ Thus, as a rule of thumb, the size of an $n \times n$ LoG discrete filter should be such that n is the smallest odd integer greater than or equal to 6σ .
 - ✓ One approach for finding the **zero crossings** at any pixel, p , of the filtered image, $g(x, y)$, is based on using a **3 x 3 neighborhood** centered at **p** .

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ A zero crossing at p implies that the signs of at least two of its opposing neighboring pixels must differ.
 - There are four cases to test: left/right, up/down, and the two diagonals.
 - ✓ If the values of are being **compared** against a **threshold**, the absolute value of their numerical difference must also exceed the threshold before we can call a zero-crossing pixel.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Marr-Hildreth edge detector
 - ✓ MATLAB: s33MarrHildEdges.m



2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - ✓ The algorithm is more complex.
 - ✓ The performance is superior in general to the edge detectors discussed so far.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - ✓ Canny's approach is based on three basic objectives:
 - a. Low error rate.
 - All edges should be found, and there should be no spurious responses.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - ✓ Canny's approach is based on three basic objectives:
 - b. Edge points should be well localized.
 - The edges located must be as close as possible to the true edges.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - ✓ Canny's approach is based on three basic objectives:
 - c. Single edge point response.
 - The detector should return only one point for each true edge point.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

➤ The Canny edge detector

1. Let $f(x, y)$ denote the input image, $f_s(x, y)$ the **smoothed** image and $G(x, y)$ the Gaussian function

$$f_s(x, y) = G(x, y) \star f(x, y)$$

✓ Gaussian filter, with $\sigma = 1.4$

$$\frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - 2. This operation is followed by computing the **gradient magnitude** and **direction** (angle),

$$g_x = \partial f_s / \partial x \quad g_y = \partial f_s / \partial y$$

$$M(x, y) = \sqrt{g_x^2 + g_y^2}$$

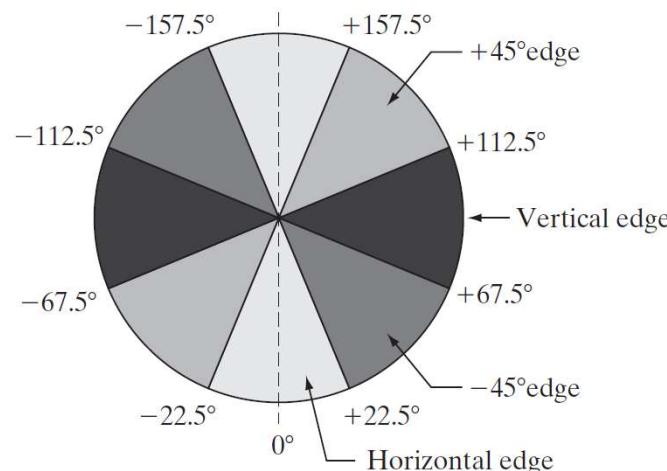
$$\alpha(x, y) = \tan^{-1} \left[\frac{g_y}{g_x} \right]$$

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

- The Canny edge detector

- ✓ In a 3×3 region we can define **four discrete orientations** for an edge passing through the center point of the region: horizontal, vertical, $+45^\circ$, -45° .

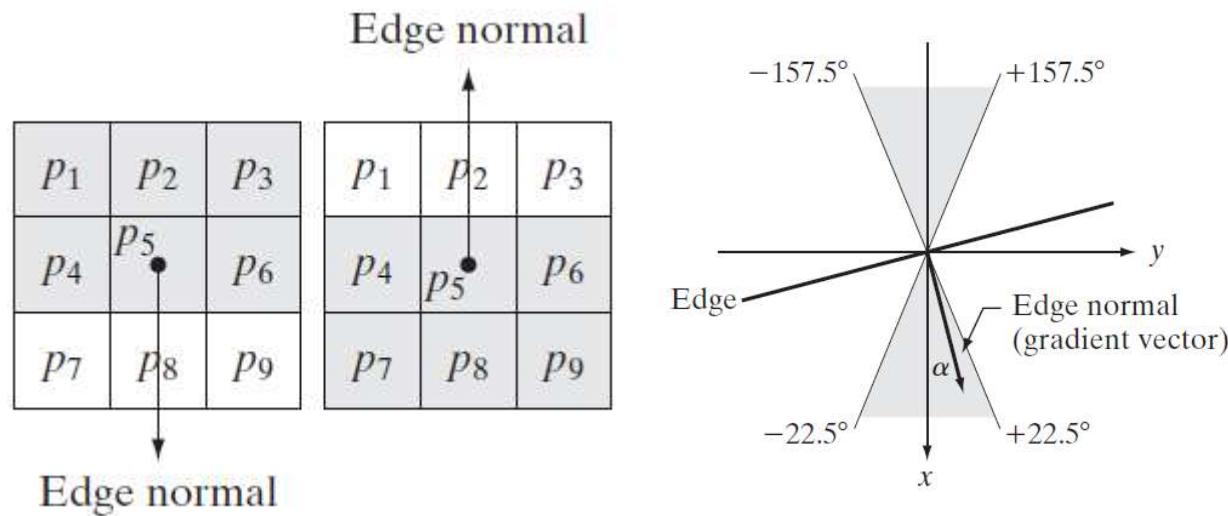


2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

- The Canny edge detector

- ✓ The **edge direction** can be determined from the direction of the **edge normal** (gradient direction) $\alpha(x, y)$.



2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection

➤ The Canny edge detector

3. Suppression of **nonmaxima**

- a. Let d_1, d_2, d_3 and d_4 denote the four basic gradient directions.
- b. For a region centered at every point (x, y) , find the direction, \mathbf{d}_k , that is closest to $\alpha(x, y)$.
- c. If the value of $M(x, y)$ **is less** than at **least one** of its **two neighbors along \mathbf{d}_k** , let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = M(x, y)$.
 - ✓ $g_N(x, y)$ is equal to $M(x, y)$ with the nonmaxima edge points suppressed.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - 4. The final operation is to **threshold** to reduce false edge points.
 - a. Canny's algorithm uses **two thresholds**, a **low** threshold, T_L and a **high** threshold, T_H .
$$g_{NH}(x, y) = g_N(x, y) \geq T_H$$
$$g_{NL}(x, y) = g_N(x, y) \geq T_L$$
 - b. Canny suggested that the **ratio** of the **high** to **low** threshold should be two or three to one.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - c. We eliminate from g_{NL} all the nonzero pixels from g_{NH} by letting,
$$g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y)$$
 - d. The nonzero pixels in g_{NH} and g_{NL} may be viewed as being “**strong**” and “**weak**” edge pixels, respectively.
 - e. Pixels in g_{NH} are assumed to be **valid edge** pixels and are so marked immediately.
 - f. The edges in g_{NH} typically have **gaps**.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - g. Longer edges are formed using the following procedure:
 - 1) Locate the next unvisited pixel, p , in \mathbf{g}_{NH} .
 - 2) Mark as **valid edge pixels** all the **weak** pixels in \mathbf{g}_{NL} that are **connected to p** .
 - 3) If all **nonzero** pixels in \mathbf{g}_{NH} have been **visited** go to Step 4. Else, return to Step 1.
 - 4) Set to **zero** all pixels in \mathbf{g}_{NL} that were **not marked** as **valid** edge pixels (step 2).

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - h. The **final image** is formed by **appending** to $\mathbf{g}_{\text{NH}}(x, y)$ the **nonzero** pixels from $\mathbf{g}_{\text{NL}}(x, y)$.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - The Canny edge detector
 - ✓ Summarizing, the Canny edge detection algorithm consists of the following basic steps:
 1. **Smooth** the input image with a Gaussian filter.
 2. Compute the **gradient** magnitude and angle images.
 3. Apply **nonmaxima suppression** to the gradient magnitude image.
 4. Use **double thresholding** and **connectivity analysis** to detect and link edges.

2. Point, Line, and Edge Detection

- More Advanced Techniques for Edge Detection
 - MATLAB: s44Canny.m



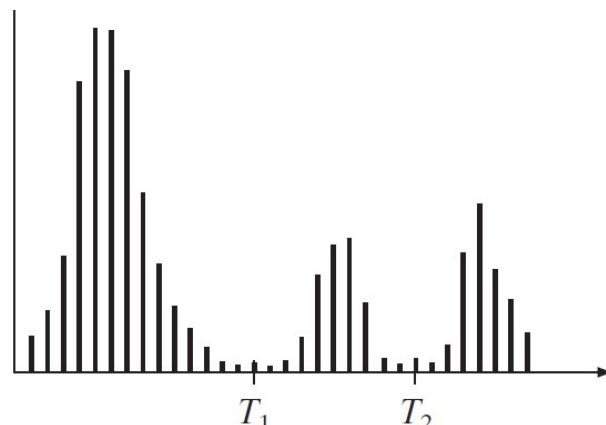
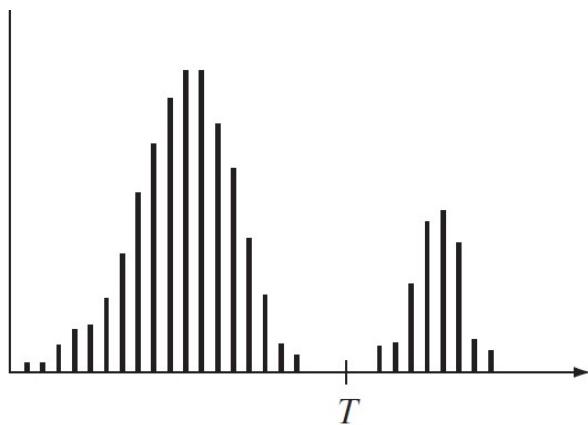
3. Threshold

- The basics of intensity thresholding
 - In the previous section, regions were identified by first finding **edge segments** and then attempting to link the segments into boundaries.
 - In this section, we discuss techniques for partitioning images directly into regions based on **intensity values** and/or properties of these values.

3. Threshold

- The basics of intensity thresholding

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$
$$g(x, y) = \begin{cases} a & \text{if } f(x, y) > T_2 \\ b & \text{if } T_1 < f(x, y) \leq T_2 \\ c & \text{if } f(x, y) \leq T_1 \end{cases}$$



a b

FIGURE 10.35
Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

3. Threshold

- The role of noise in image thresholding

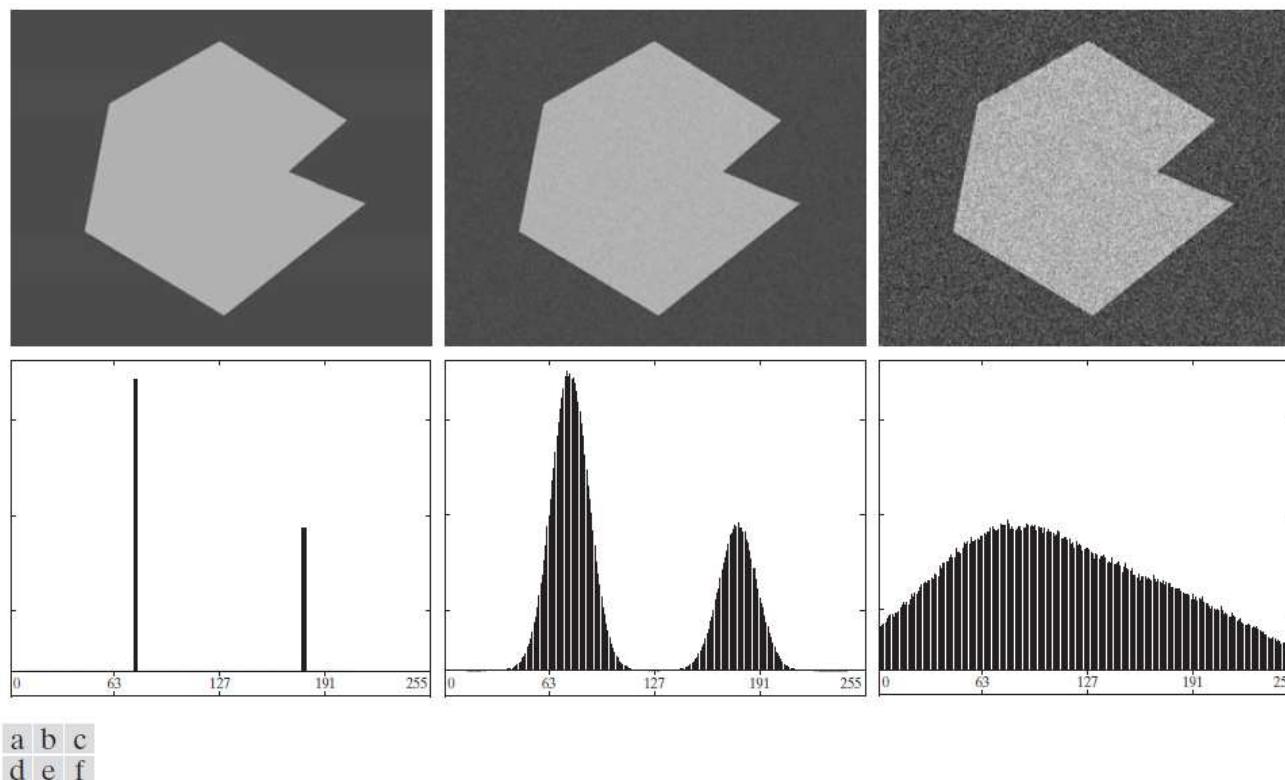


FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

3. Threshold

- The role of illumination and reflectance

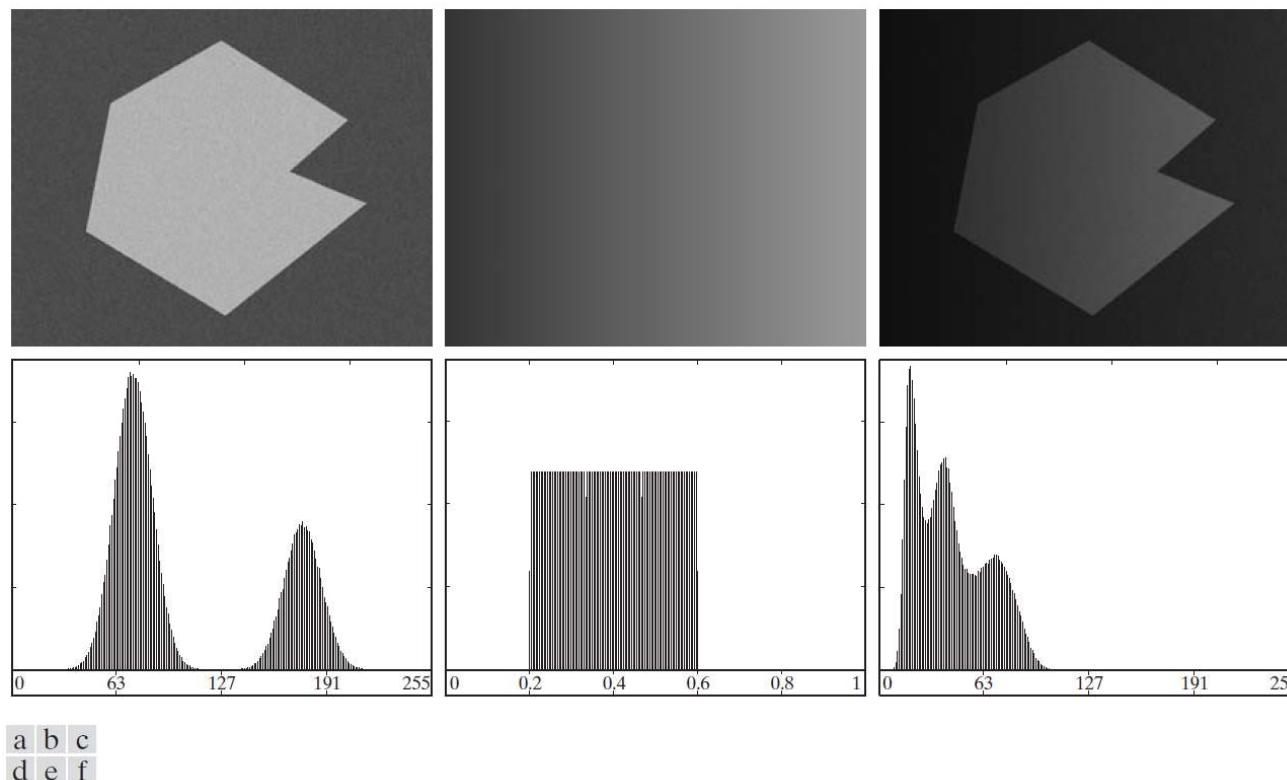


FIGURE 10.37 (a) Noisy image. (b) Intensity ramp in the range $[0.2, 0.6]$. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

3. Threshold

- Basic Global Thresholding
 - Even if global thresholding is a suitable approach, an algorithm capable of estimating automatically the threshold value for each image is required.
 1. Select an initial estimate for the global threshold, T .
 2. Segment the image using T in Eq. (10.3-1). This will produce two groups of pixels: G_1 consisting of all pixels with intensity values $> T$, and G_2 consisting of pixels with values $\leq T$.
 3. Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2 , respectively.
 4. Compute a new threshold value:
$$T = \frac{1}{2}(m_1 + m_2)$$
 - 5. Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined parameter ΔT .

3. Threshold

- Basic Global Thresholding

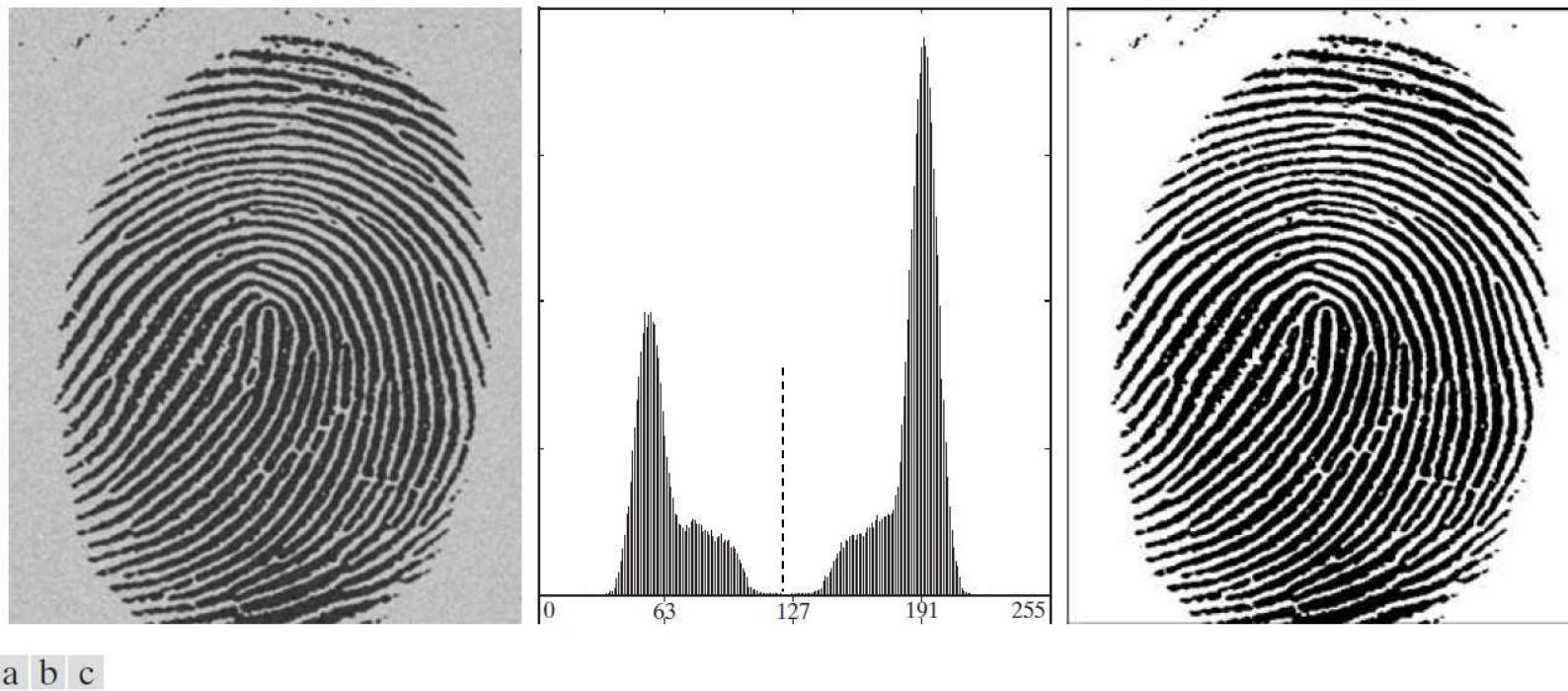


FIGURE 10.38 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)

3. Threshold

- Optimum Global Thresholding Using Otsu's Method
 - Thresholding may be viewed as a **statistical-decision theory problem**.
 - ✓ The objective is to **minimize the average error** incurred in assigning pixels to two or more groups (also called classes).
 - This problem is known to have an elegant closed-form solution known as the **Bayes decision rule**.
 - The solution is **based on the PDF of the intensity levels** of **each class** and the probability that each class occurs in a given application.

3. Threshold

- Optimum Global Thresholding Using Otsu's Method
 - Unfortunately, **estimating PDFs is not a trivial** so the problem usually is simplified by making workable assumptions about the form of the PDFs, such as assuming that they are **Gaussian functions**.
 - Even with **simplifications**, implementing solutions using these assumptions can be complex and **not always well-suited** for practical applications.
 - The approach discussed in this section, called **Otsu's method** is an attractive alternative.
 - It is optimum in the sense that it **maximizes the between-class variance**, a well-known measure used in statistical discriminant analysis.

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

➤ Otsu's algorithm may be summarized as follows:

1. Compute the normalized histogram of the input image. Denote the components of the histogram by $p_i, i = 0, 1, 2, \dots, L - 1$.
2. Compute the cumulative sums, $P_1(k)$, for $k = 0, 1, 2, \dots, L - 1$, using Eq. (10.3-4).

$$P_1(k) = \sum_{i=0}^k p_i$$

3. Compute the cumulative means, $m(k)$, for $k = 0, 1, 2, \dots, L - 1$, using Eq. (10.3-8).

$$m(k) = \sum_{i=0}^k ip_i$$

4. Compute the global intensity mean, m_G , using (10.3-9).

$$m_G = \sum_{i=0}^{L-1} ip_i$$

3. Threshold

- Optimum Global Thresholding Using Otsu's Method
 - Otsu's algorithm may be summarized as follows:

5. Compute the between-class variance, $\sigma_B^2(k)$, for $k = 0, 1, 2, \dots, L - 1$, using Eq. (10.3-17).

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}$$

6. Obtain the Otsu threshold, k^* , as the value of k for which $\sigma_B^2(k)$ is maximum. If the maximum is not unique, obtain k^* by averaging the values of k corresponding to the various maxima detected.
7. Obtain the separability measure, η^* , by evaluating Eq. (10.3-16) at $k = k^*$.

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \quad \sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$$

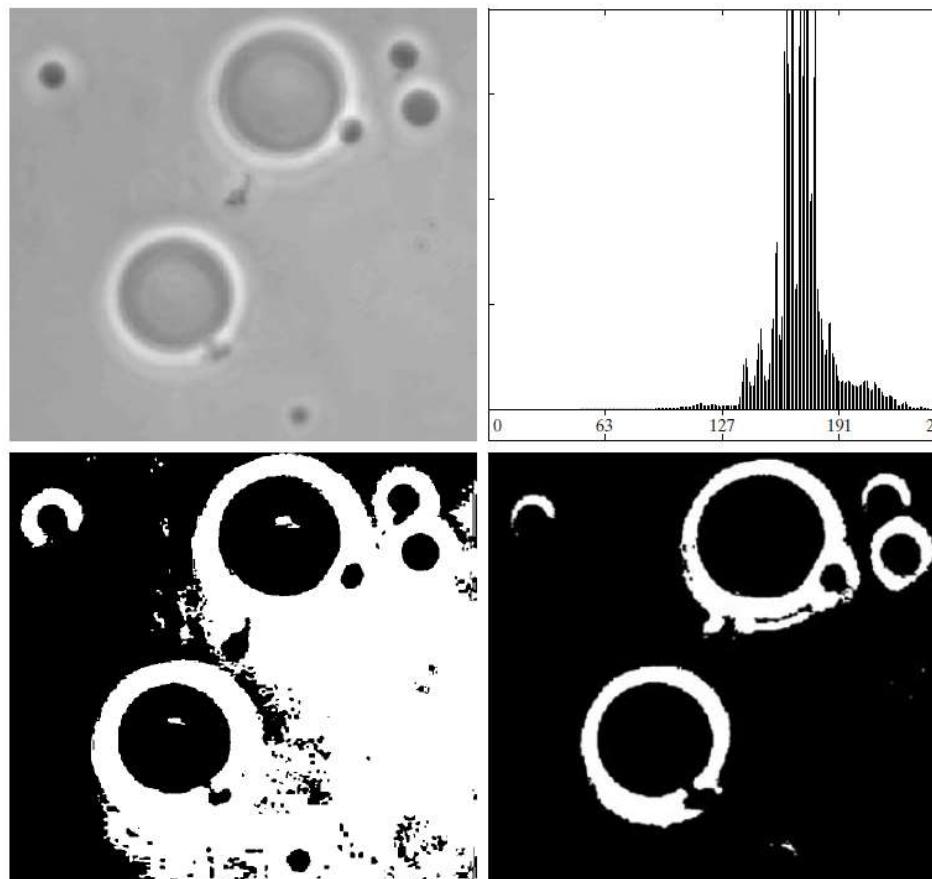
3. Threshold

- Optimum Global Thresholding Using Otsu's Method
 - Once k^* has been obtained, the input image is segmented as before:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > k^* \\ 0 & \text{if } f(x, y) \leq k^* \end{cases}$$

3. Threshold

- Optimum Global Thresholding Using Otsu's Method



a
b
c
d

FIGURE 10.39
(a) Original image.
(b) Histogram (high peaks were clipped to highlight details in the lower values).
(c) Segmentation result using the basic global algorithm from Section 10.3.2.
(d) Result obtained using Otsu's method.
(Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

3. Threshold

- Optimum Global Thresholding Using Otsu's Method

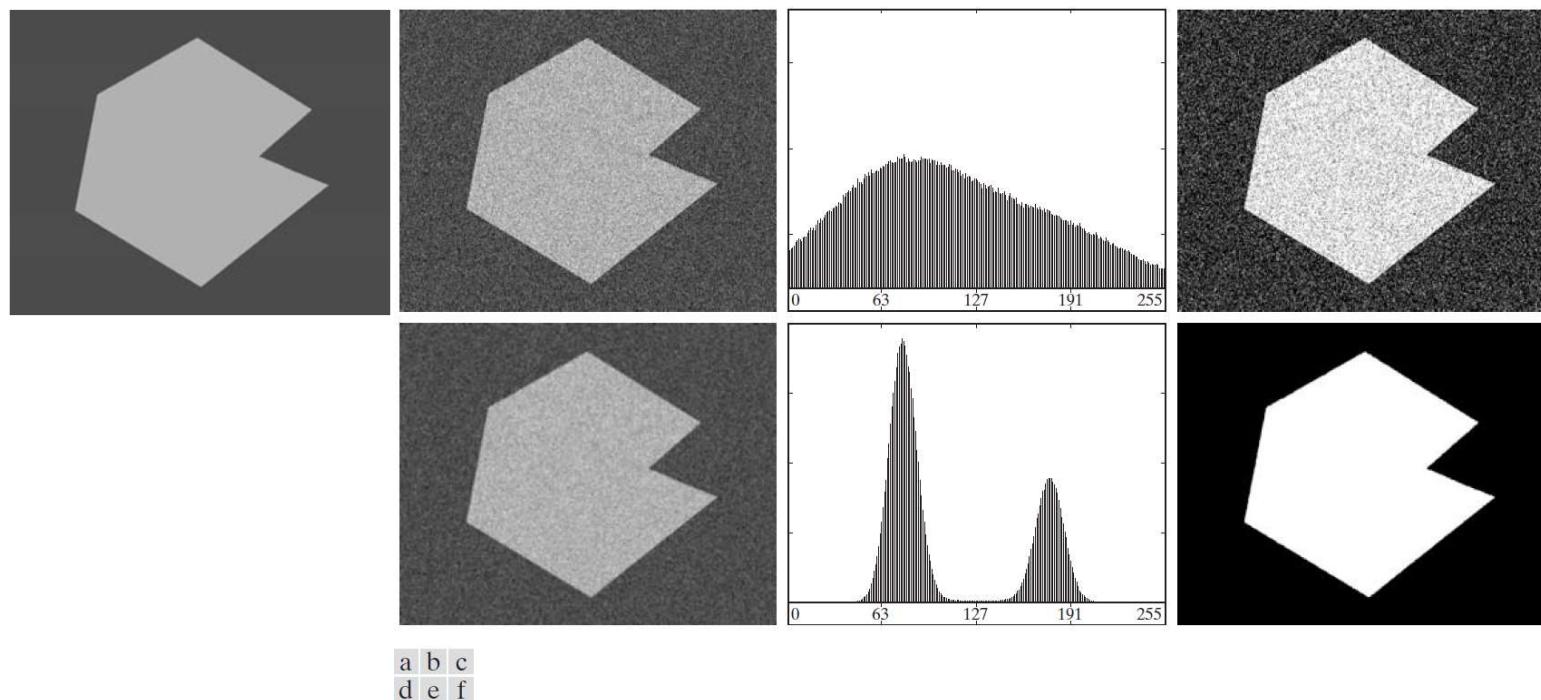


FIGURE 10.40 (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

3. Threshold

- Using Edges to Improve Global Thresholding

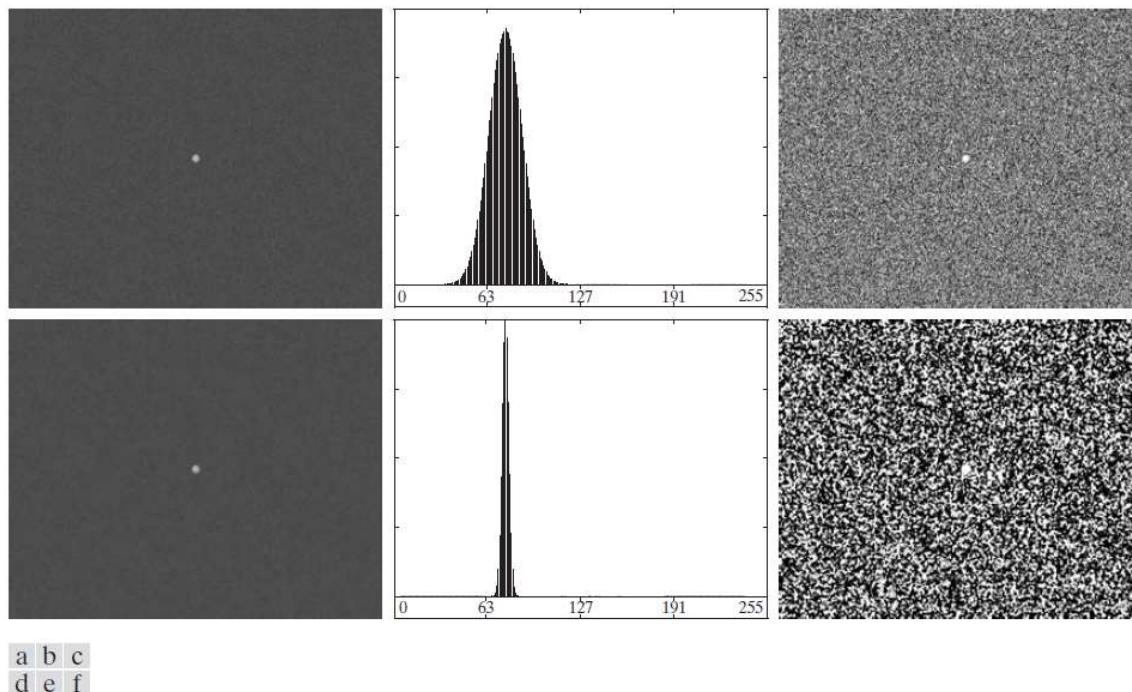


FIGURE 10.41 (a) Noisy image and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method. Thresholding failed in both cases.

3. Threshold

- Using Edges to Improve Global Thresholding

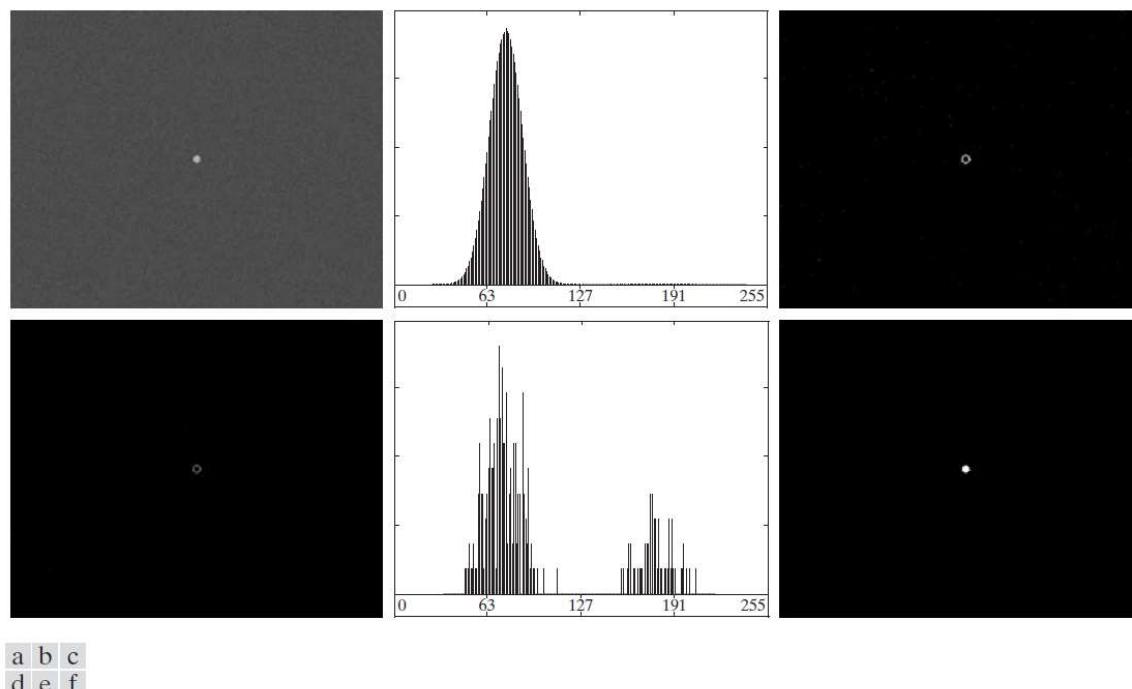


FIGURE 10.42 (a) Noisy image from Fig. 10.41(a) and (b) its histogram. (c) Gradient magnitude image thresholded at the 99.7 percentile. (d) Image formed as the product of (a) and (c). (e) Histogram of the nonzero pixels in the image in (d). (f) Result of segmenting image (a) with the Otsu threshold based on the histogram in (e). The threshold was 134, which is approximately midway between the peaks in this histogram.

3. Threshold

- Using Edges to Improve Global Thresholding

The objective is to detect the bright spots.

This value of T corresponds approximately to the 99.5 percentile.

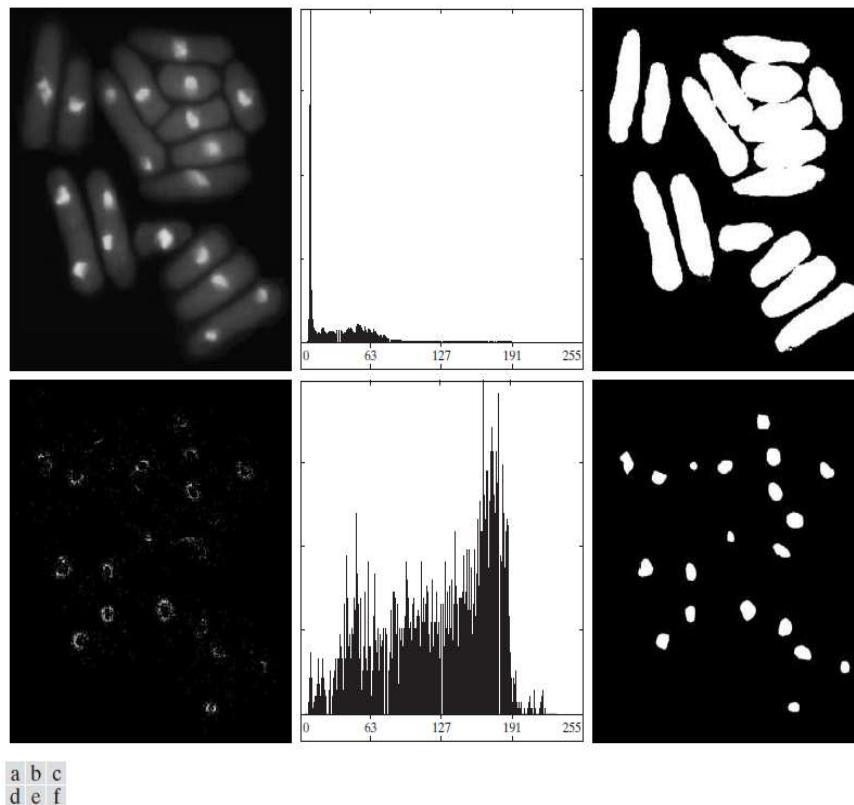


FIGURE 10.43 (a) Image of yeast cells. (b) Histogram of (a). (c) Segmentation of (a) with Otsu's method using the histogram in (b). (d) Thresholded absolute Laplacian. (e) Histogram of the nonzero pixels in the product of (a) and (d). (f) Original image thresholded using Otsu's method based on the histogram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

3. Threshold

- Using Edges to Improve Global Thresholding

➤ The preceding discussion is summarized in the following algorithm:

1. Compute an edge image as either the magnitude of the gradient, or absolute value of the Laplacian, of $f(x, y)$ using any of the methods discussed in Section 10.2.
2. Specify a threshold value, T .
3. Threshold the image from Step 1 using the threshold from Step 2 to produce a binary image, $g_T(x, y)$. This image is used as a *mask image* in the following step to select pixels from $f(x, y)$ corresponding to “strong” edge pixels.
4. Compute a histogram using only the pixels in $f(x, y)$ that correspond to the locations of the 1-valued pixels in $g_T(x, y)$.
5. Use the histogram from Step 4 to segment $f(x, y)$ globally using, for example, Otsu’s method.

3. Threshold

- Multiple Thresholds

➤ In the case of K classes, C_1, C_2, \dots, C_K :

$$m_k = \frac{1}{P_k} \sum_{i \in C_k} i p_i \quad m_G = \sum_{i=0}^{L-1} i p_i \quad P_k = \sum_{i \in C_k} p_i$$

$$\sigma_B^2 = \sum_{k=1}^K P_k (m_k - m_G)^2$$

$$\sigma_B^2(k_1^*, k_2^*, \dots, k_{K-1}^*) = \max_{0 < k_1 < k_2 < \dots < k_{n-1} < L-1} \sigma_B^2(k_1, k_2, \dots, k_{K-1})$$

3. Threshold

- Multiple Thresholds
 - In the case of 3 classes, C_1 , C_2 and C_3

$$m_1 = \frac{1}{P_1} \sum_{i=0}^{k_1} i p_i \quad P_1 = \sum_{i=0}^{k_1} p_i$$

$$m_2 = \frac{1}{P_2} \sum_{i=k_1+1}^{k_2} i p_i \quad P_2 = \sum_{i=k_1+1}^{k_2} p_i$$

$$m_3 = \frac{1}{P_3} \sum_{i=k_2+1}^{L-1} i p_i \quad P_3 = \sum_{i=k_2+1}^{L-1} p_i$$

3. Threshold

- Multiple Thresholds
 - In the case of 3 classes, C_1 , C_2 and C_3

$$\begin{aligned}\sigma_B^2 = & P_1(m_1 - m_G)^2 + \\ & P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2\end{aligned}$$

$$\sigma_B^2(k_1^*, k_2^*) = \max_{0 < k_1 < k_2 < L-1} \sigma_B^2(k_1, k_2)$$

3. Threshold

- Multiple Thresholds
 - In the case of 3 classes, C_1 , C_2 and C_3
 1. The procedure starts by selecting the first value of k_1 (that value is 1).
 2. Next, k_2 is incremented through all its values greater than k_1 and less than $L-1$.
 3. This procedure is repeated until $k_1 = L-3$.
 4. The last step is to look for the maximum value in this array.

3. Threshold

- Multiple Thresholds

- In the case of 3 classes, C_1 , C_2 and C_3
 - ✓ The thresholded image is then given by

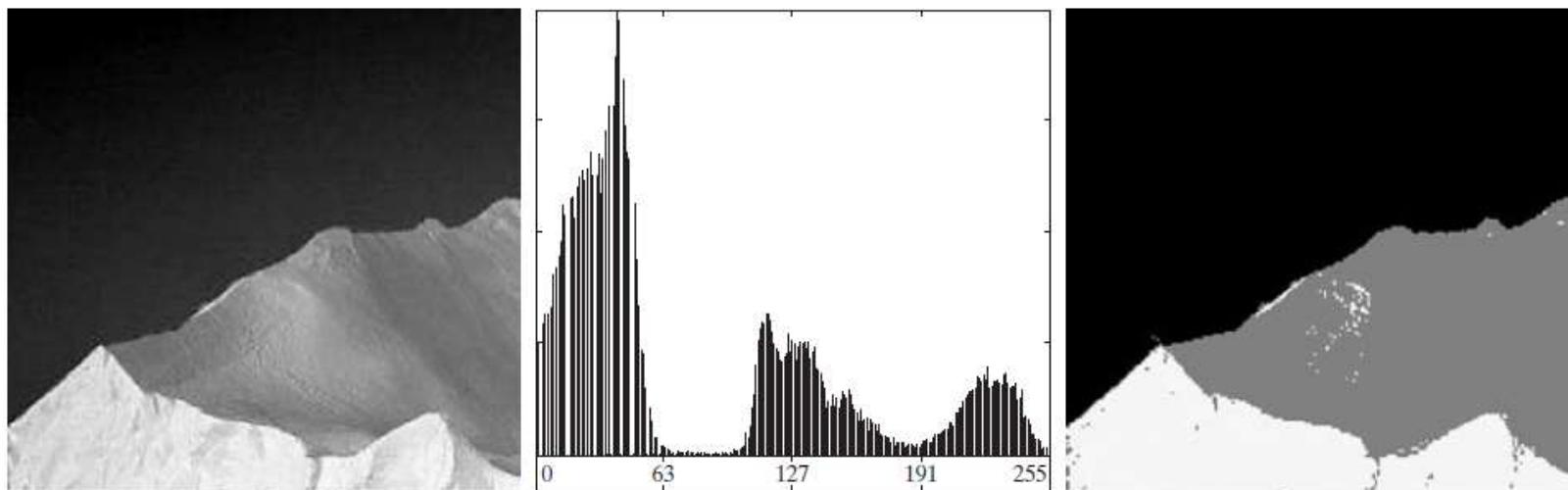
$$g(x, y) = \begin{cases} a & \text{if } f(x, y) \leq k_1^* \\ b & \text{if } k_1^* < f(x, y) \leq k_2^* \\ c & \text{if } f(x, y) > k_2^* \end{cases}$$

- ✓ Separability measure

$$\eta(k_1^*, k_2^*) = \frac{\sigma_B^2(k_1^*, k_2^*)}{\sigma_G^2}$$

3. Threshold

- Multiple Thresholds



a b c

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

3. Threshold

- Variable Thresholding

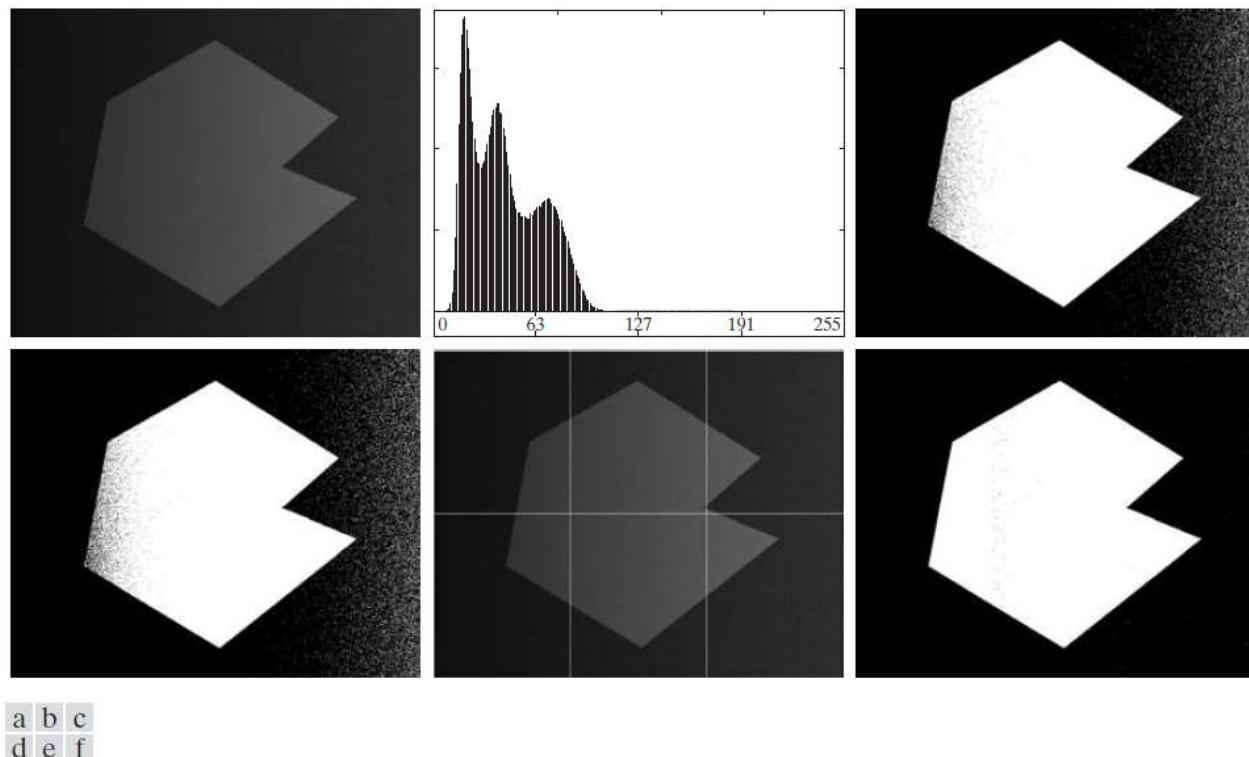


FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 07

Image Transforms

1. Introduction

- In some cases, image **processing tasks** are best formulated by transforming the input images, carrying the specified task in a **transform domain**, and applying the **inverse transform** to return to the spatial domain.
- A particularly important class of **2-D linear transforms**, can be expressed in the general form.

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)r(x, y, u, v)$$

where **f** is the input image, **r** is called the **forward transformation** kernel, and the equation is evaluated for and $u = 0, 1, \dots, M-1$ and $v = 0, 1, \dots, N-1$.

1. Introduction

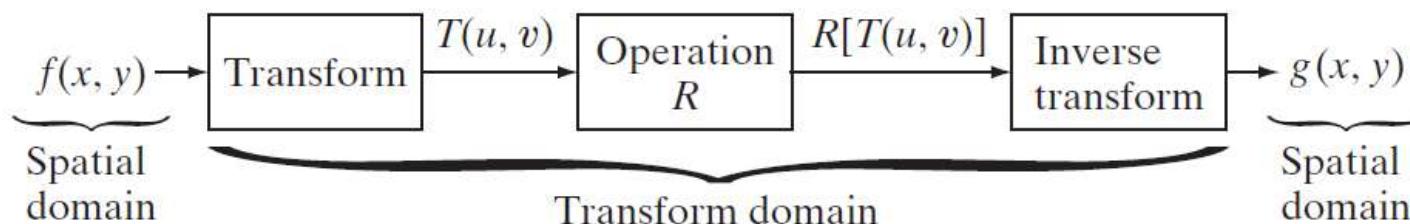
- T is called the **forward transform** of f .
- Given T we can recover f using the **inverse transform**.

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v)s(x, y, u, v)$$

- The equation is evaluated for and $x=0, 1, \dots, M-1$ and $y=0, 1, \dots, N-1$.
- And s is called the **inverse transformation** kernel.

1. Introduction

- By using these transforms, it is possible to express an image as a **combination** of a set **of basic signals**, known as the **basis functions**.
- The image output in the **transformed space** may be analyzed, interpreted, and further processed for implementing **diverse** image processing **tasks**.
- General approach for operating in the linear transform domain.



2. Discrete Fourier Transform

- Substituting the kernels below into the previous equations yields the **Discrete Fourier Transform** pair

$$r(x, y, u, v) = e^{-j2\pi(ux+vy)/n}$$

$$s(x, y, u, v) = \frac{1}{n^2} e^{j2\pi(ux+vy)/n}$$

$$M = N = n$$

- The Discrete Fourier Transform was discussed in “**Topic 04**
- Filtering in the Frequency Domain”.

2. Discrete Fourier Transform

- The 2-D Discrete Fourier Transform and its inverse.

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

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

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

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

- DFT uses a set of **complex exponential** functions.
- Normally used for **general spectral analysis** applications.

3. Discrete Cosine Transform

- **Discrete Cosine Transform** (DCT) is the basis for many image and video **compression algorithms**, especially the baseline JPEG and MPEG standards for compression of **still** and **video** images respectively.
- It is obtained by using the following (equal) kernels:

$$r(x, y, u, v) = s(x, y, u, v)$$

$$= \alpha(u)\alpha(v) \cos\left[\frac{(2x + 1)u\pi}{2n}\right] \cos\left[\frac{(2y + 1)v\pi}{2n}\right]$$

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

3. Discrete Cosine Transform

- DCT uses only (real-valued) cosine functions.
- It translates the correlated data to uncorrelated data.
- The DCT and inverse DCT can be computed using the DFT.
- Example: 8 basis functions of a 4-by-4 matrix.

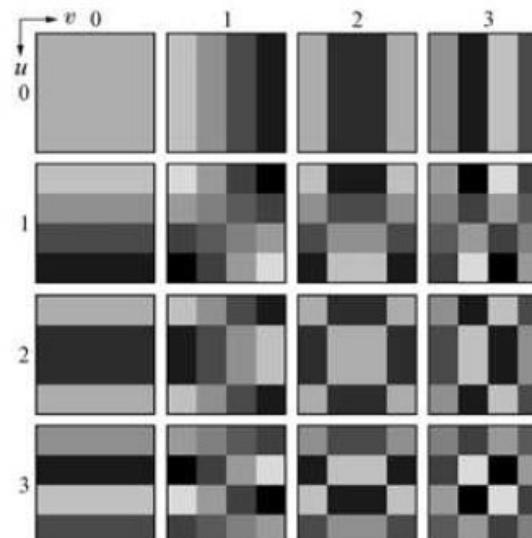


FIGURE 8.23
Discrete-cosine
basis functions for
 $n = 4$. The origin
of each block is at
its top left.

3. Discrete Cosine Transform

- **Mean-square** reconstruction **error** is related directly to the **energy** or **information** packing properties of the transform employed.
- An image $g(x, y)$ can be expressed as a function of its 2-D transform

$$g(x, y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) s(x, y, u, v)$$

$$x, y = 0, 1, 2, \dots, n - 1$$

- The inverse kernel s can be viewed as defining a set of **basis functions** or **basis images**.

3. Discrete Cosine Transform

- This interpretation becomes clearer if we use the notation:

$$\mathbf{G} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) \mathbf{S}_{uv}$$

$$\mathbf{S}_{uv} = \begin{bmatrix} s(0, 0, u, v) & s(0, 1, u, v) & \cdots & s(0, n - 1, u, v) \\ s(1, 0, u, v) & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ s(n - 1, 0, u, v) & s(n - 1, 1, u, v) & \cdots & s(n - 1, n - 1, u, v) \end{bmatrix}$$

- \mathbf{G} contains the pixels of the image and is defined as a linear combination of n^2 matrices of size $n \times n$ that is, \mathbf{S}_{uv} , for $u, v = 0, 1, 2, \dots, n-1$.

3. Discrete Cosine Transform

- We can **define** a transform coefficient **masking function** χ which is constructed to **eliminate** the **basis images** that make the smallest contribution to the total sum

$$\chi(u, v) = \begin{cases} 0 & \text{if } T(u, v) \text{ satisfies a specified truncation criterion} \\ 1 & \text{otherwise} \end{cases}$$

$$\hat{\mathbf{G}} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \chi(u, v) T(u, v) \mathbf{S}_{uv}$$

- The mean-square error between \mathbf{G} and $\hat{\mathbf{G}}$ approximation is

$$e_{ms} = E\left\{ \|\mathbf{G} - \hat{\mathbf{G}}\|^2 \right\} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \sigma_{T(u,v)}^2 [1 - \chi(u, v)]$$

- The total mean-square approximation error thus is the sum of the variances of the discarded transform coefficients.

3. Discrete Cosine Transform

- Transformations that redistribute or pack the most information into the fewest coefficients provide the smallest reconstruction errors.

4. Discrete Wavelet Transform

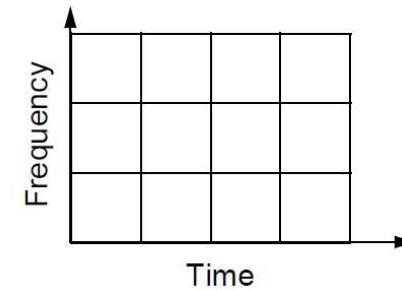
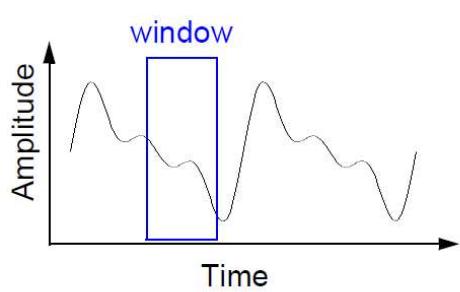
- Signal analysts already have at their disposal an impressive arsenal of tools. Perhaps the most well-known of these is **Fourier analysis**.



- Fourier analysis has a **serious drawback**. In transforming to the frequency domain, **time information is lost**.
- If a signal does not change much over time this drawback is not very important.

4. Discrete Wavelet Transform

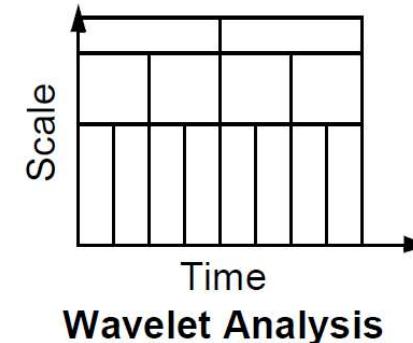
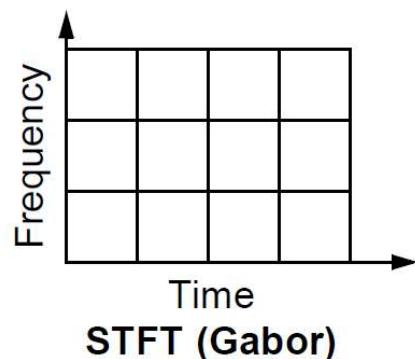
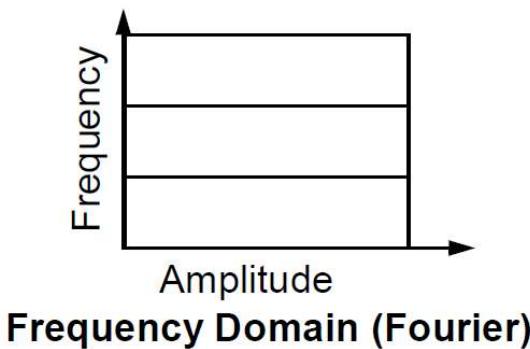
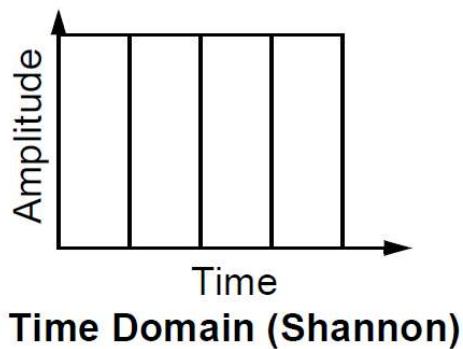
- However, most **interesting signals** contain numerous non-stationary or **transitory characteristics**.
- In an effort to **correct** this deficiency, Dennis Gabor (1946) adapted the Fourier transform to analyze only **a small section** of the signal at a **time**.
 - Short-Time Fourier Transform (STFT)



- Precision is determined by the size of the window.

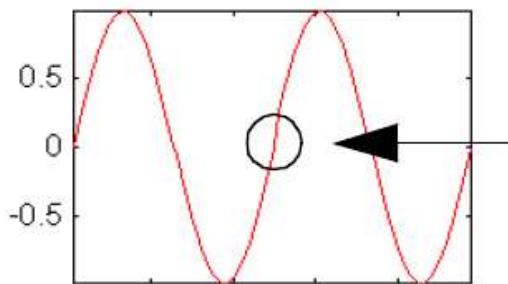
4. Discrete Wavelet Transform

- **Wavelet analysis** represents the next logical step: a windowing technique with variable-sized regions.

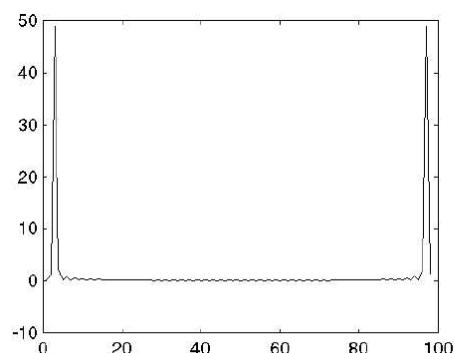


4. Discrete Wavelet Transform

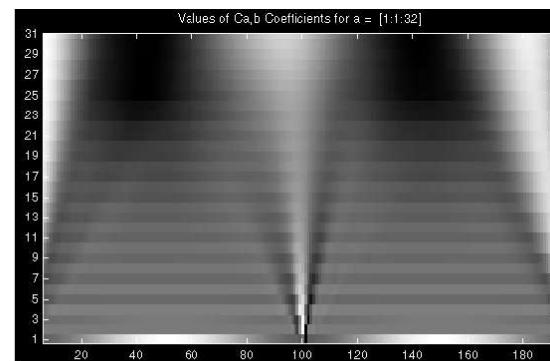
- One major **advantage** afforded by wavelets is the ability to perform **local analysis** — that is, to analyze a localized area of a larger signal.



Sinusoid with a small discontinuity



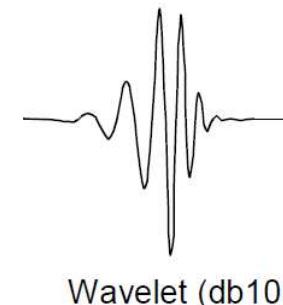
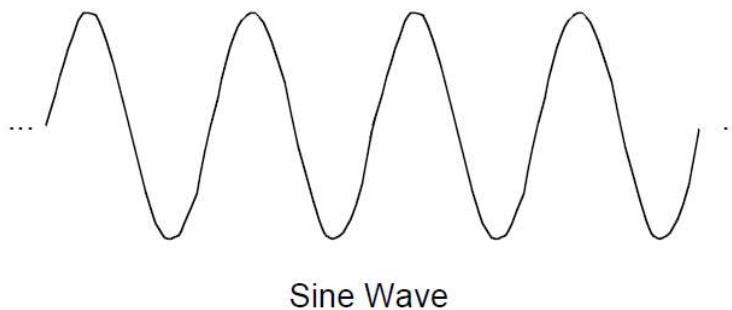
Fourier Coefficients



Wavelet Coefficients

4. Discrete Wavelet Transform

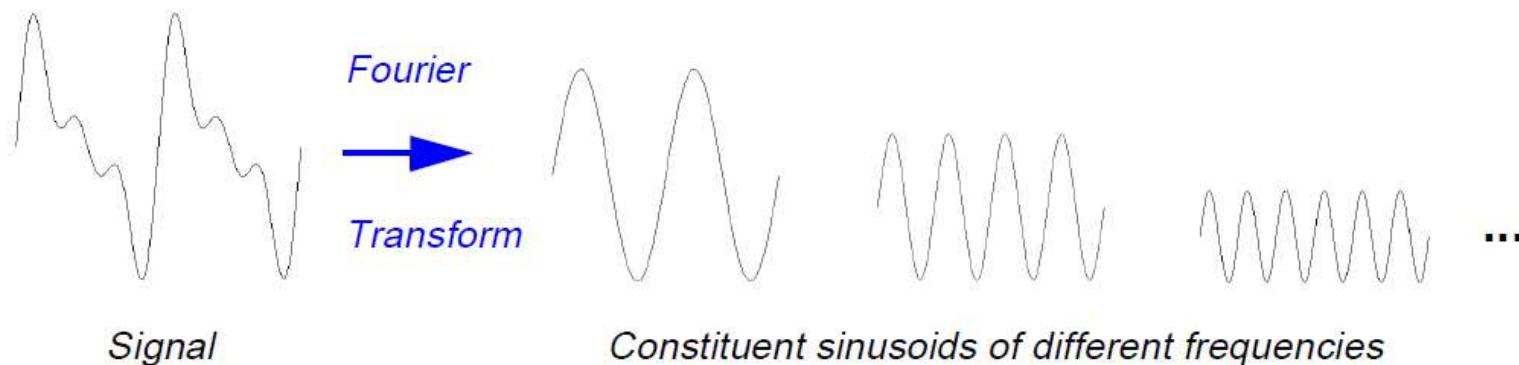
- A wavelet is a waveform of effectively **limited duration** that has an **average** value of **zero**.
- Compare wavelets with sine waves, which are the basis of Fourier analysis:
 - Sinusoids do not have limited duration.
 - Sinusoids are smooth and predictable.
 - Wavelets tend to be irregular and asymmetric.



4. Discrete Wavelet Transform

- Fourier analysis consists of breaking up a signal into sine waves of various frequencies.

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$



4. Discrete Wavelet Transform

- The **Continuous Wavelet Transform** (CWT) of a continuous function $f(x)$, relative to a real-valued wavelet, $\psi(x)$, is defined as

$$W_{\psi}(s, \tau) = \int_{-\infty}^{\infty} f(x)\psi_{s,\tau}(x) dx$$

$$\psi_{s,\tau}(x) = \frac{1}{\sqrt{s}}\psi\left(\frac{x - \tau}{s}\right)$$

where s and τ are called **scale** and **translation** parameters, respectively.

4. Discrete Wavelet Transform

- The function $f(x)$ can be obtained using the **inverse** Continuous Wavelet Transform,

$$f(x) = \frac{1}{C_\psi} \int_0^\infty \int_{-\infty}^\infty W_\psi(s, \tau) \frac{\psi_{s,\tau}(x)}{s^2} d\tau ds$$

$$C_\psi = \int_{-\infty}^\infty \frac{|\Psi(\mu)|^2}{|\mu|} d\mu$$

where $\Psi(\mu)$ is the Fourier transform of $\psi(x)$.

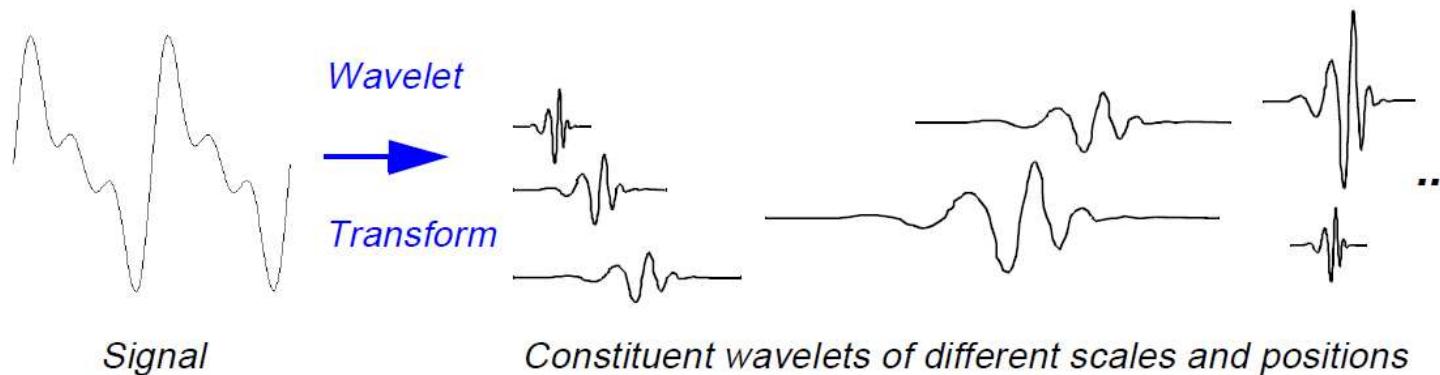
- The previous equations define a reversible transformation as long as the so-called admissibility criterion is satisfied,

$$C_\psi < \infty$$

4. Discrete Wavelet Transform

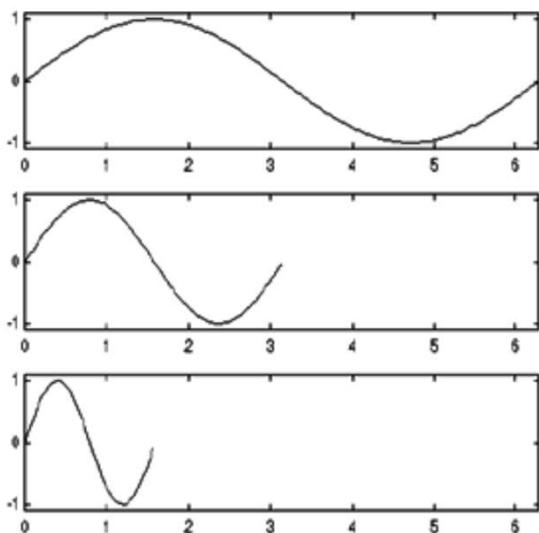
- Similarly, wavelet analysis is the breaking up of a signal into **shifted** and **scaled** versions of the original (or **mother**) wavelet.

$$C(\text{scale}, \text{position}) = \int_{-\infty}^{\infty} f(t)\psi(\text{scale}, \text{position}, t)dt$$



4. Discrete Wavelet Transform

- Scaling a wavelet simply means **stretching** (or compressing) it.



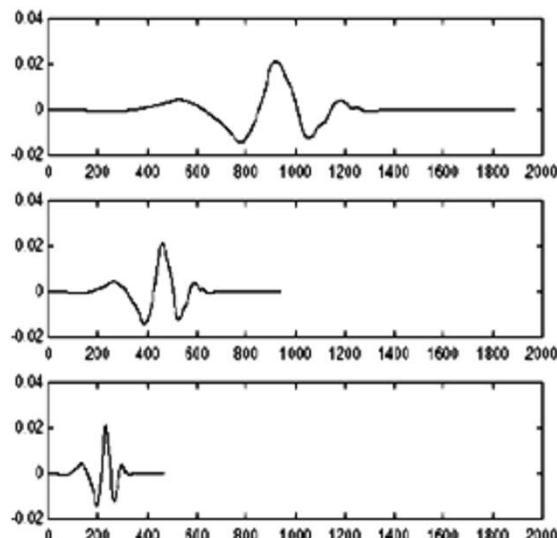
$$f(t) = \sin(t) ; a = 1$$

$$f(t) = \sin(2t) ; a = \frac{1}{2}$$

$$f(t) = \sin(4t) ; a = \frac{1}{4}$$

4. Discrete Wavelet Transform

- The scale factor works exactly the same with wavelets.



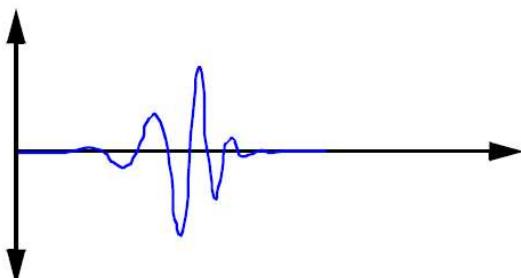
$$f(t) = \psi(t) \quad ; \quad a = 1$$

$$f(t) = \psi(2t) \quad ; \quad a = \frac{1}{2}$$

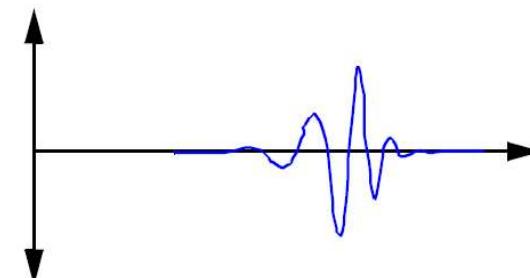
$$f(t) = \psi(4t) \quad ; \quad a = \frac{1}{4}$$

4. Discrete Wavelet Transform

- Shifting a wavelet simply means delaying (or hastening) its onset.



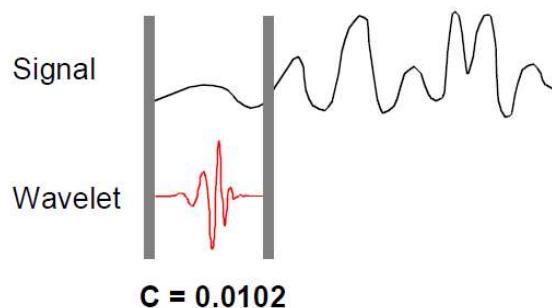
Wavelet function
 $\psi(t)$



Shifted wavelet function
 $\psi(t - k)$

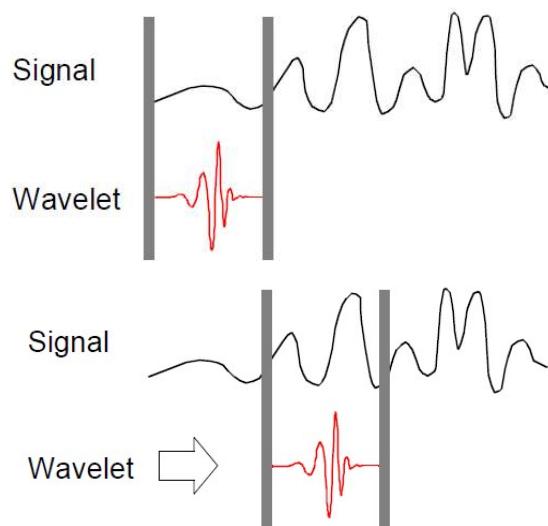
4. Discrete Wavelet Transform

- The CWT is the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet.
- Computing the CWT:
 1. Take a wavelet and **compare** it to a section at the start of the original signal.
 2. Calculate a number, C , that represents **how closely correlated** the wavelet is with this section of the signal.



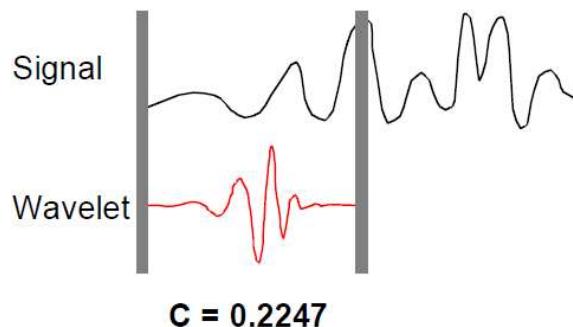
4. Discrete Wavelet Transform

- Computing the CWT:
 3. Shift the wavelet to the right and repeat steps 1 and 2 until you've covered the whole signal.



4. Discrete Wavelet Transform

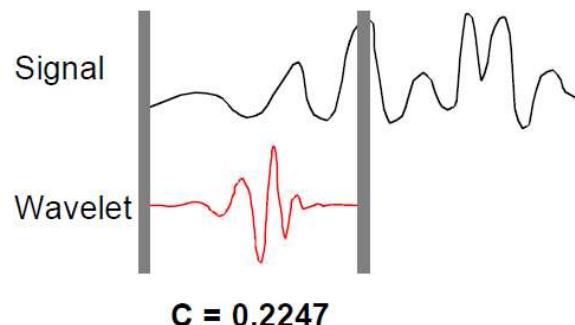
- Computing the CWT:
 4. Scale (stretch) the wavelet and repeat steps 1 through 3.



5. Repeat steps 1 through 4 for all scales.

4. Discrete Wavelet Transform

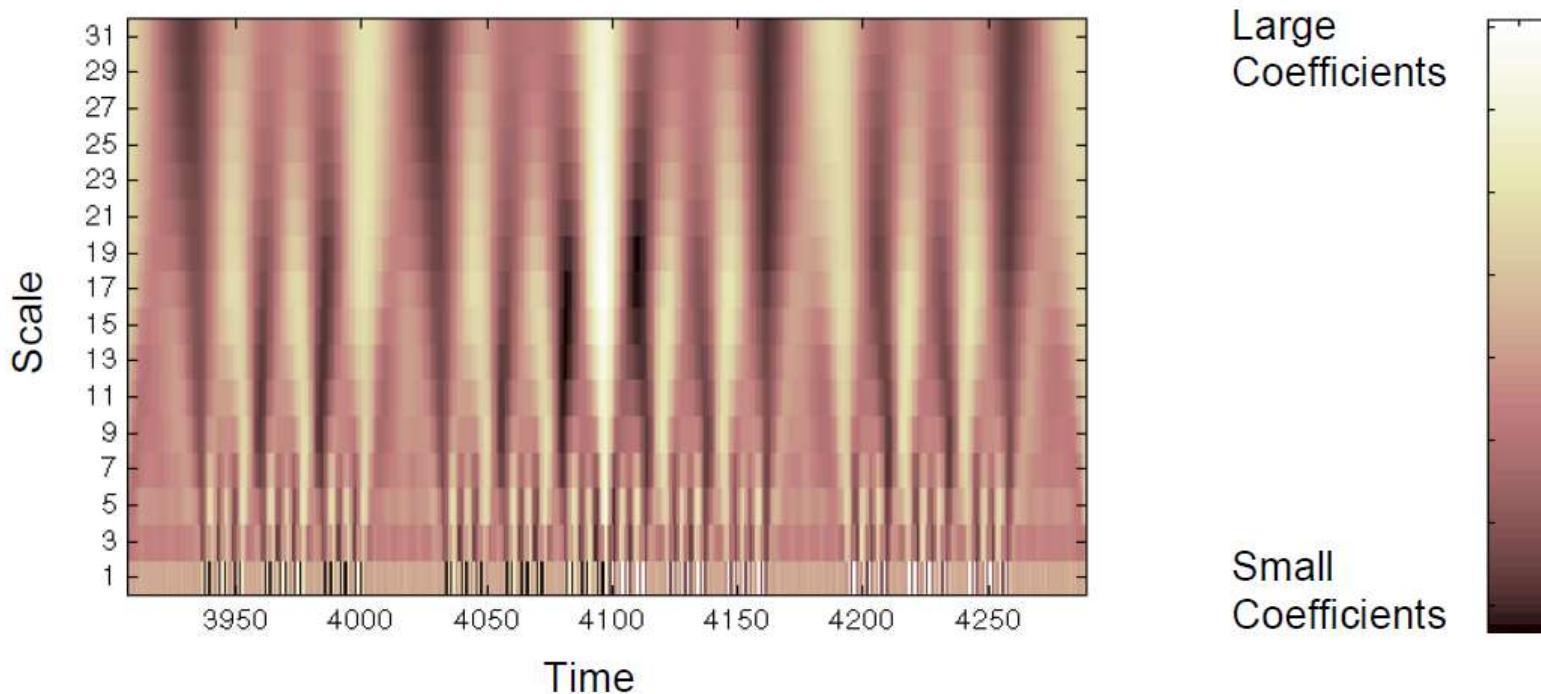
- Computing the CWT:
 4. Scale (stretch) the wavelet and repeat steps 1 through 3.



- 5. Repeat steps 1 through 4 for all scales.
 - When you're done, you'll have the coefficients produced at different scales by different sections of the signal.

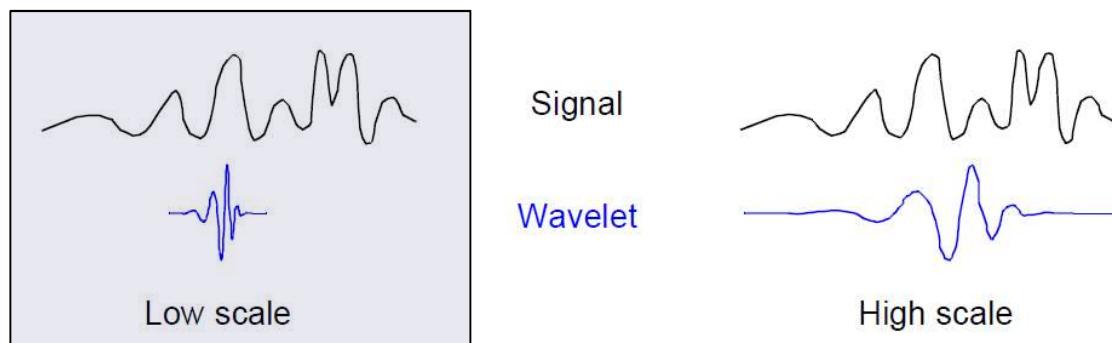
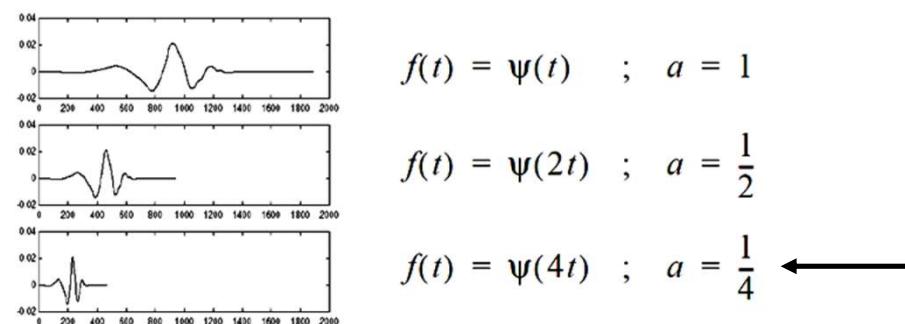
4. Discrete Wavelet Transform

- How to make sense of all these coefficients?



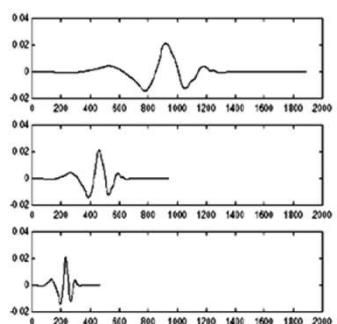
4. Discrete Wavelet Transform

- There is a correspondence between wavelet scales and frequency:
 - Low scale ➔ Compressed wavelet ➔ Rapidly changing details ➔ High frequency



4. Discrete Wavelet Transform

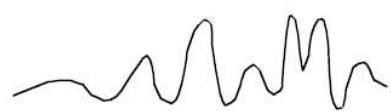
- There is a correspondence between wavelet scales and frequency:
 - High scale $a \rightarrow$ Stretched wavelet \rightarrow Slowly changing, coarse features \rightarrow Low frequency



$$f(t) = \psi(t) ; a = 1 \quad \longleftarrow$$

$$f(t) = \psi(2t) ; a = \frac{1}{2}$$

$$f(t) = \psi(4t) ; a = \frac{1}{4}$$

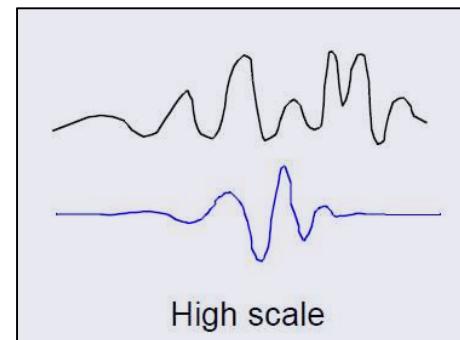


Low scale

Signal



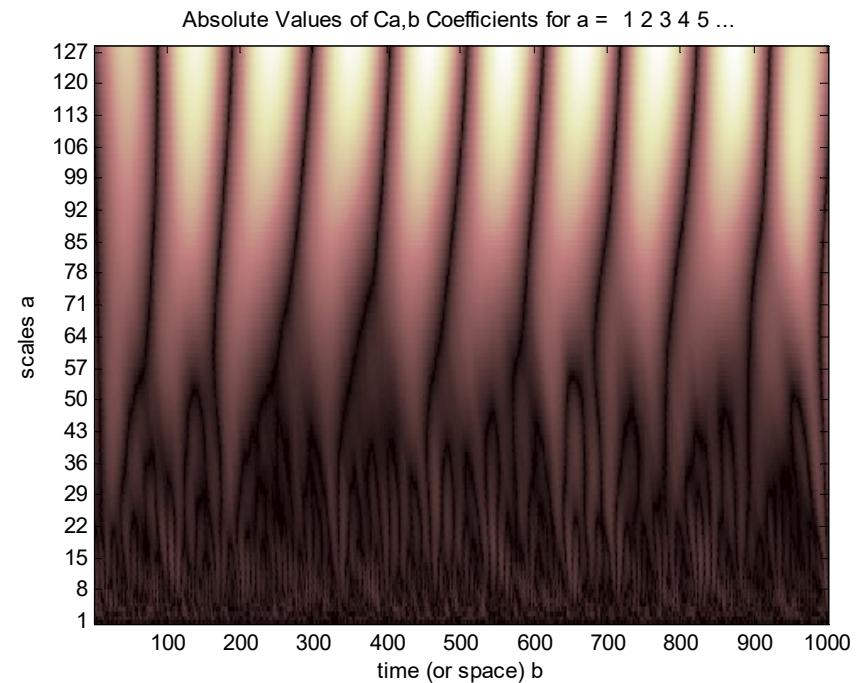
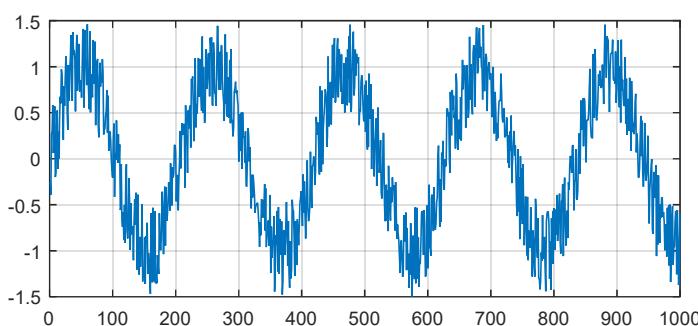
Wavelet



4. Discrete Wavelet Transform

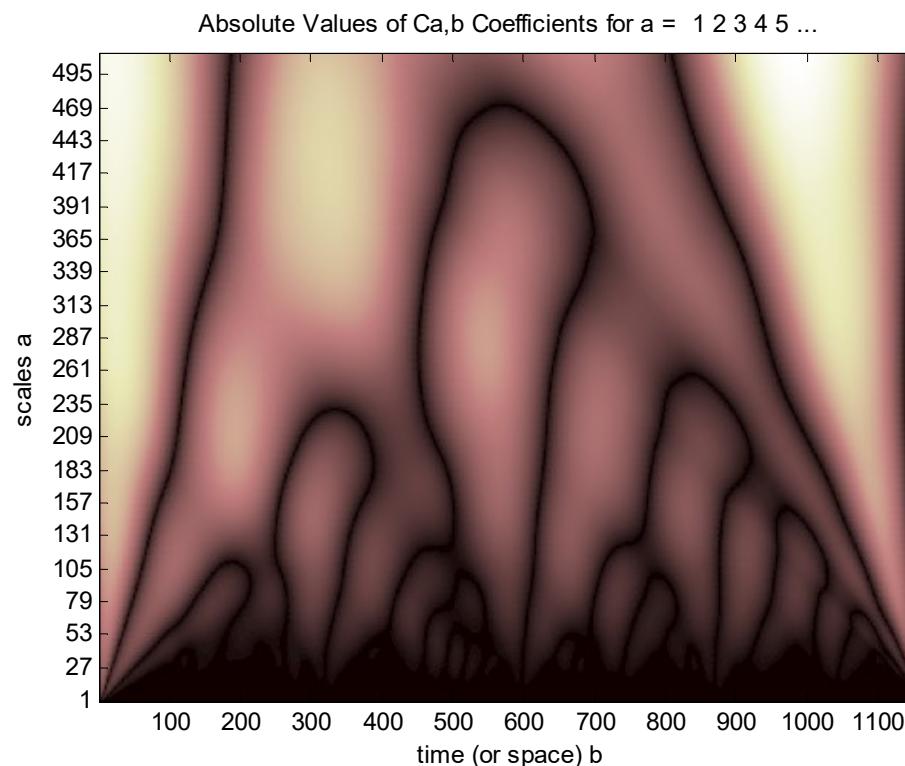
- There is a correspondence between wavelet scales and frequency:

```
load noissin;
c = cwt(noissin,1:128,'db4','plot');
```



4. Discrete Wavelet Transform

- MATLAB: s32Lunar.m



4. Discrete Wavelet Transform

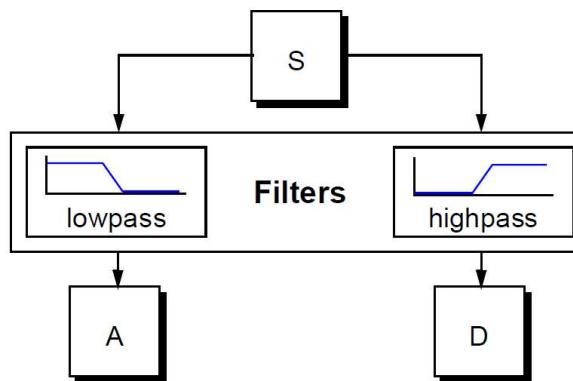
- What is “**continuous**” about the Continuous Wavelet Transform (CWT) are the **scales** at which it operates.
- CWT can operate at **every scale**, from that of the original signal up to some maximum scale which you determine.
- The CWT is also **continuous** in terms of **shifting**.
- What if we choose only a **subset** of **scales** and **positions** at which to make our calculations?
 - It turns out, that if we choose **scales** and **positions based** on **powers of two** then our analysis will just as **accurate**.
- We obtain such an analysis from the **Discrete Wavelet Transform** (DWT).

4. Discrete Wavelet Transform

- An efficient way to implement this scheme **using filters** was developed in 1988 by **Mallat**.
- For many signals, the **low-frequency** content is the most important part. It is what gives the signal its **identity**.
- The **high-frequency** content, on the other hand, imparts **nuance**.
- It is for this reason that, in wavelet analysis, we often speak of **approximations** and **details**.

4. Discrete Wavelet Transform

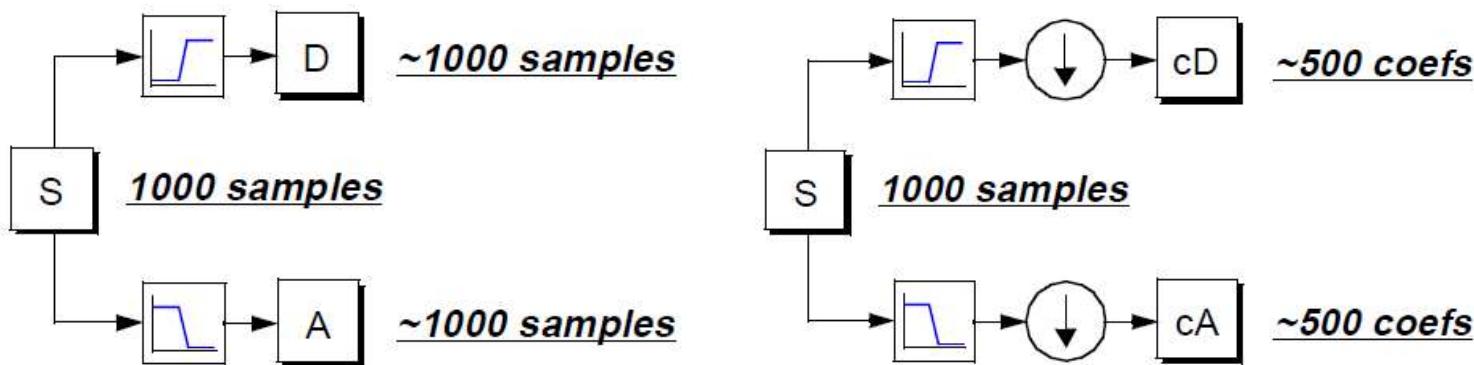
- The **approximations** are the high-scale, low-frequency components of the signal.
- The **details** are the low-scale, high-frequency components.
- The filtering process, at its most basic level, looks like this:



- If we actually perform this operation on a real digital signal, we wind up with **twice as much data** as we started with.

4. Discrete Wavelet Transform

- To correct this problem, we introduce the notion of **downsampling**.
- This simply means **throwing away** every second data point.

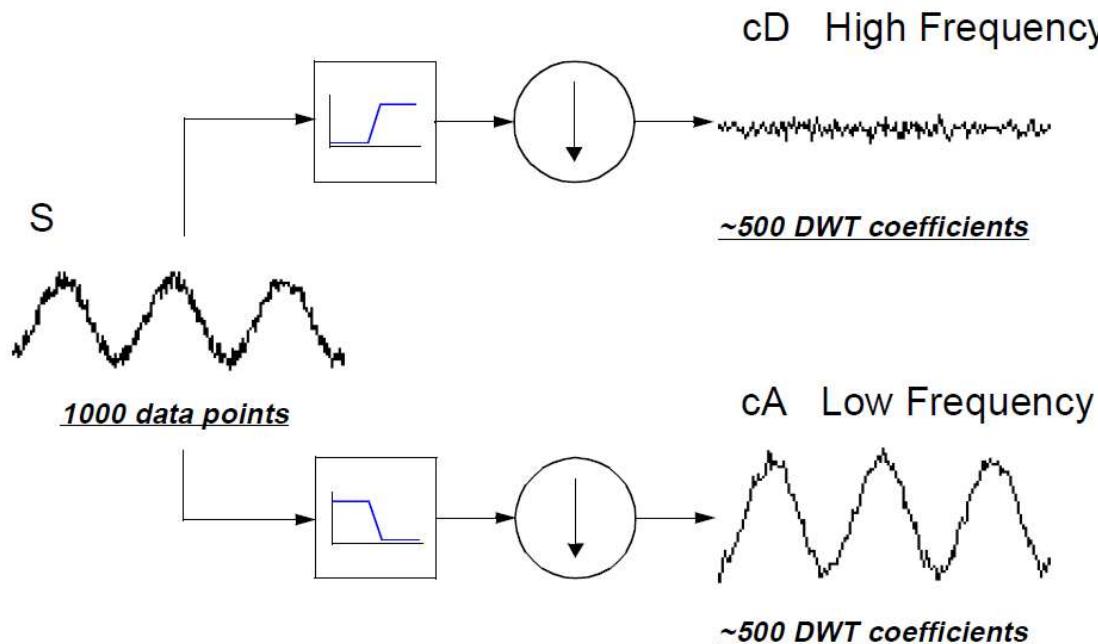


- The process on the right, which includes downsampling, produces **DWT coefficients**.

4. Discrete Wavelet Transform

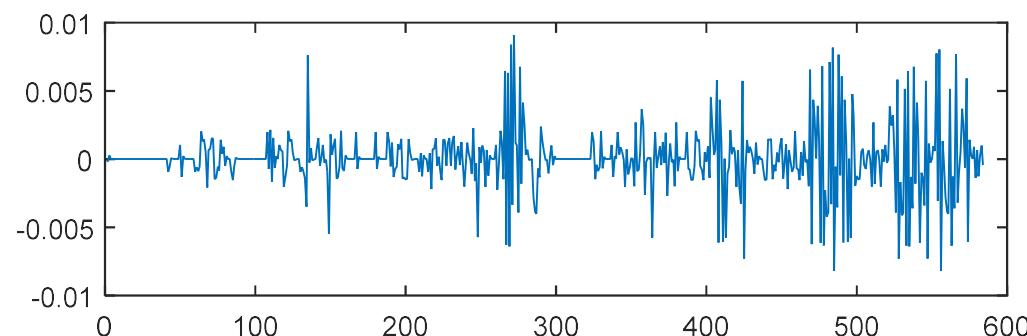
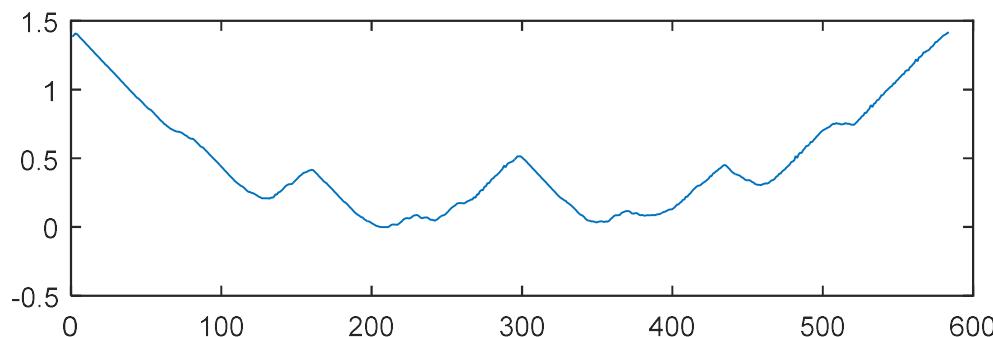
- To correct this problem, we introduce the notion of **downsampling**.

```
s = sin(20.*linspace(0,pi,1000)) + 0.5.*rand(1,1000);  
[cA,cD] = dwt(s,'db2');
```



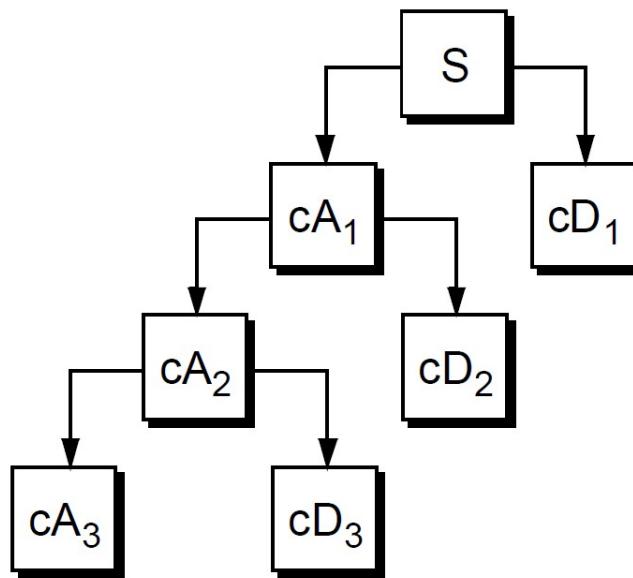
4. Discrete Wavelet Transform

- MATLAB: s39ApproxDet.m



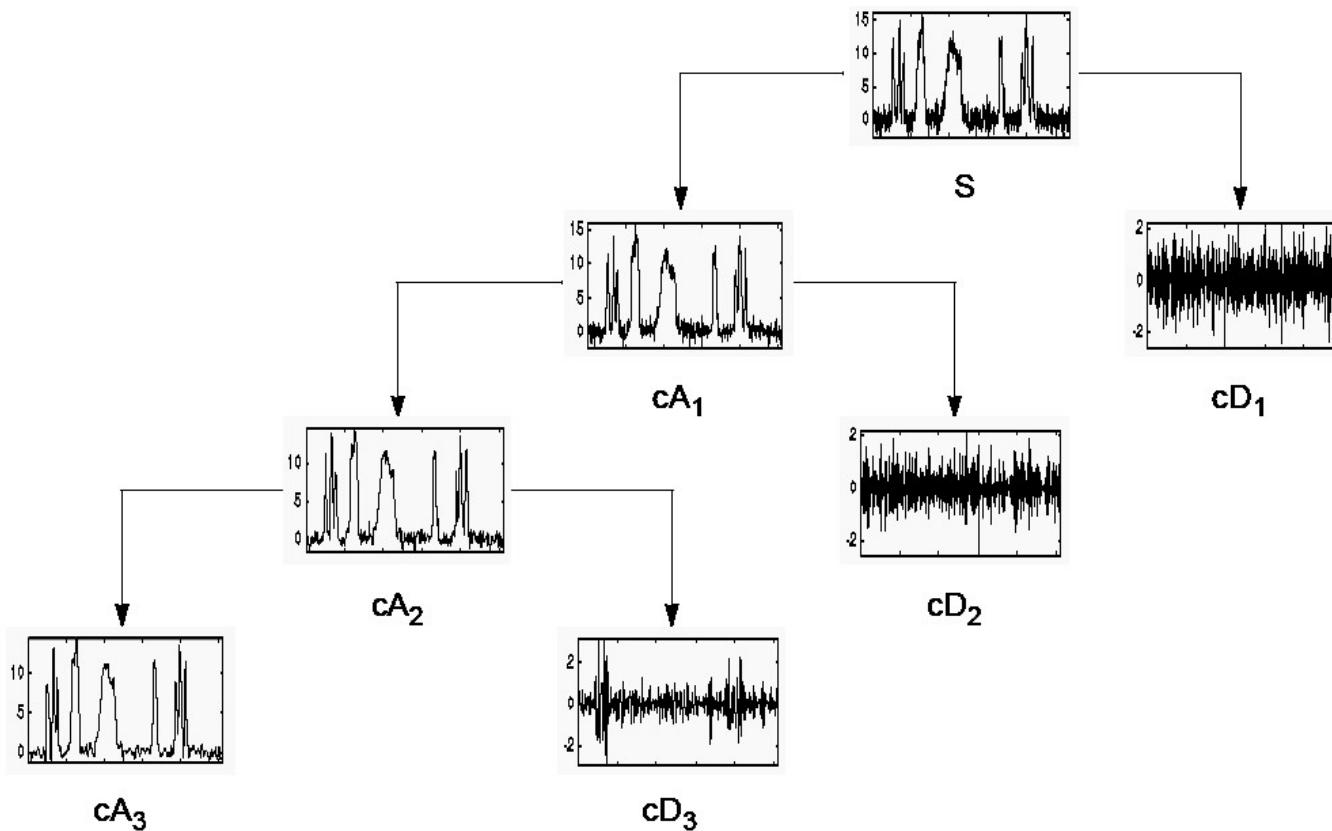
4. Discrete Wavelet Transform

- The decomposition process can be iterated, with **successive approximations** being **decomposed** in turn, so that one signal is broken down into many lower-resolution components.
- This is called the wavelet decomposition tree.



4. Discrete Wavelet Transform

- The decomposition process can be iterated.

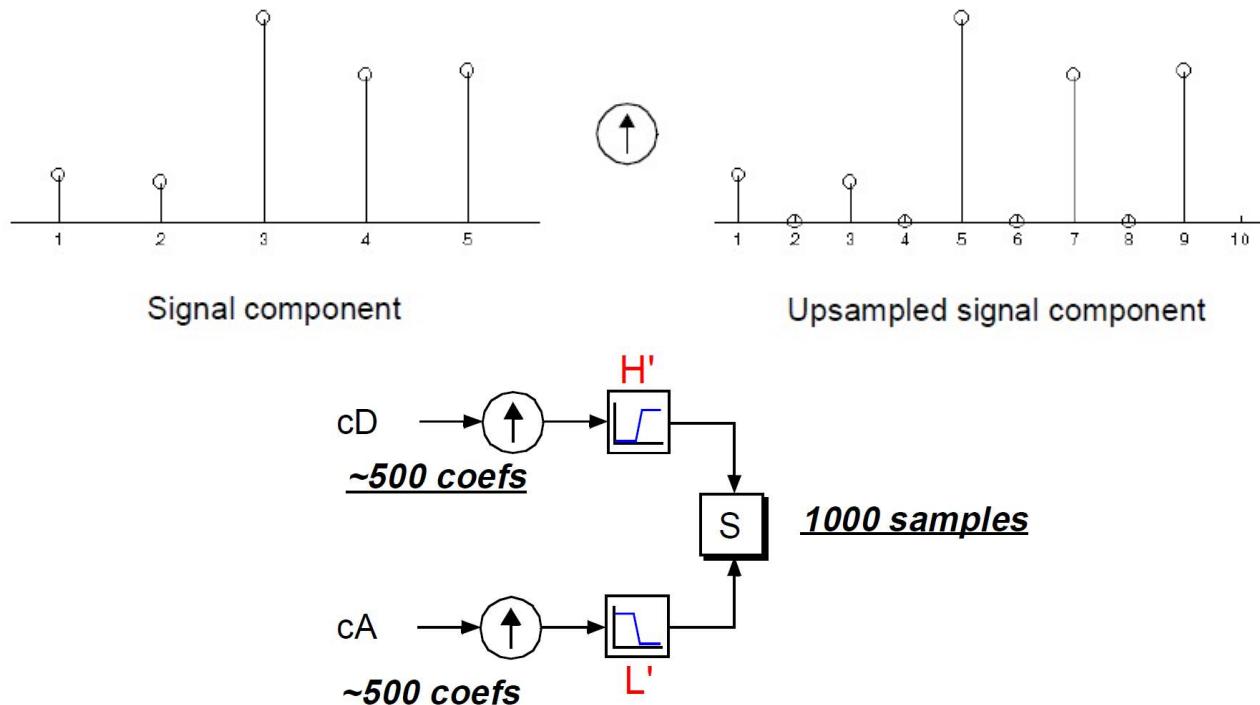


4. Discrete Wavelet Transform

- We have learned how the discrete wavelet transform can be used to analyze, or **decompose**, signals and images.
- The other half of the story is how those components can be **assembled** back into the original signal with no loss of information.
- This process is called **reconstruction**, or **synthesis**.
- The mathematical manipulation that effects synthesis is called the **inverse** Discrete Wavelet Transform (IDWT).

4. Discrete Wavelet Transform

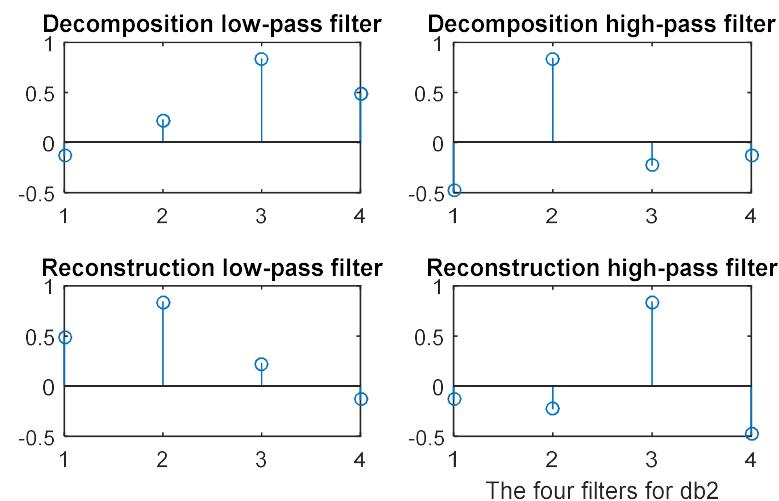
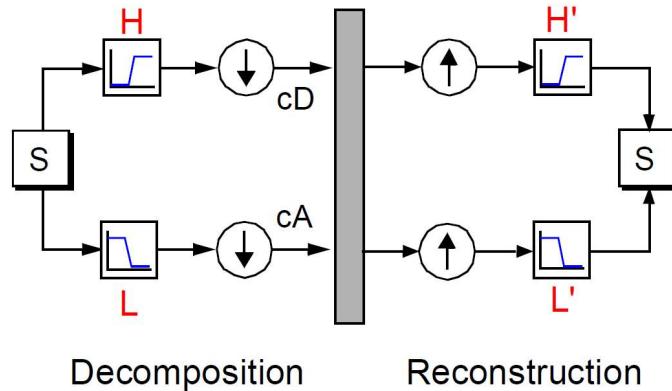
- Wavelet **analysis** involves **filtering** and **downsampling**.
- The wavelet **reconstruction** process consists of **upsampling** and **filtering**.



4. Discrete Wavelet Transform

- By carefully choosing the decomposition and reconstruction filters we can “cancel out” the effects of aliasing caused by the subsampling.

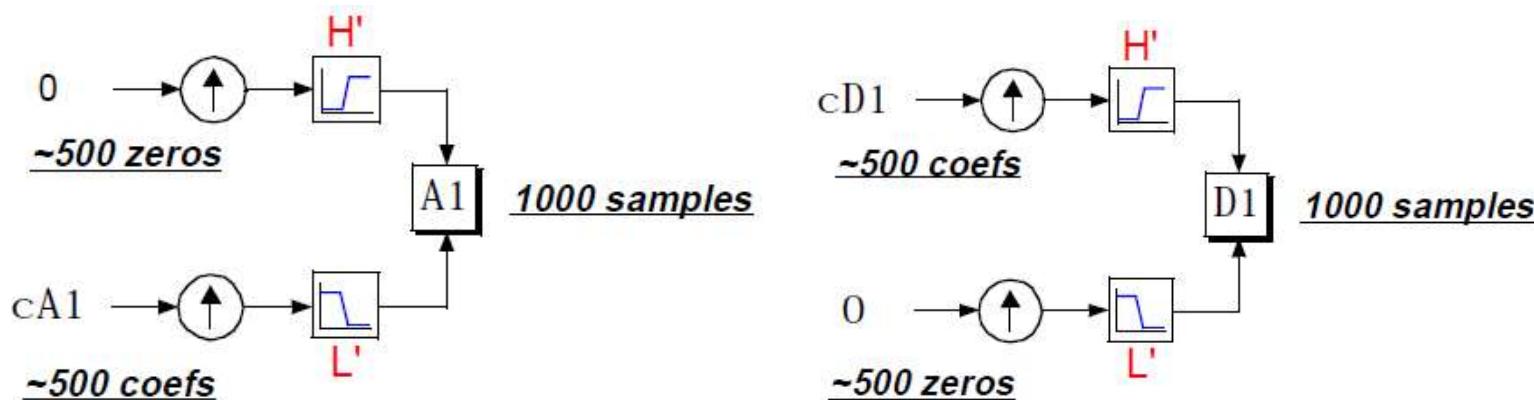
- Quadrature mirror filters:**



- MATLAB: s44Filters.m

4. Discrete Wavelet Transform

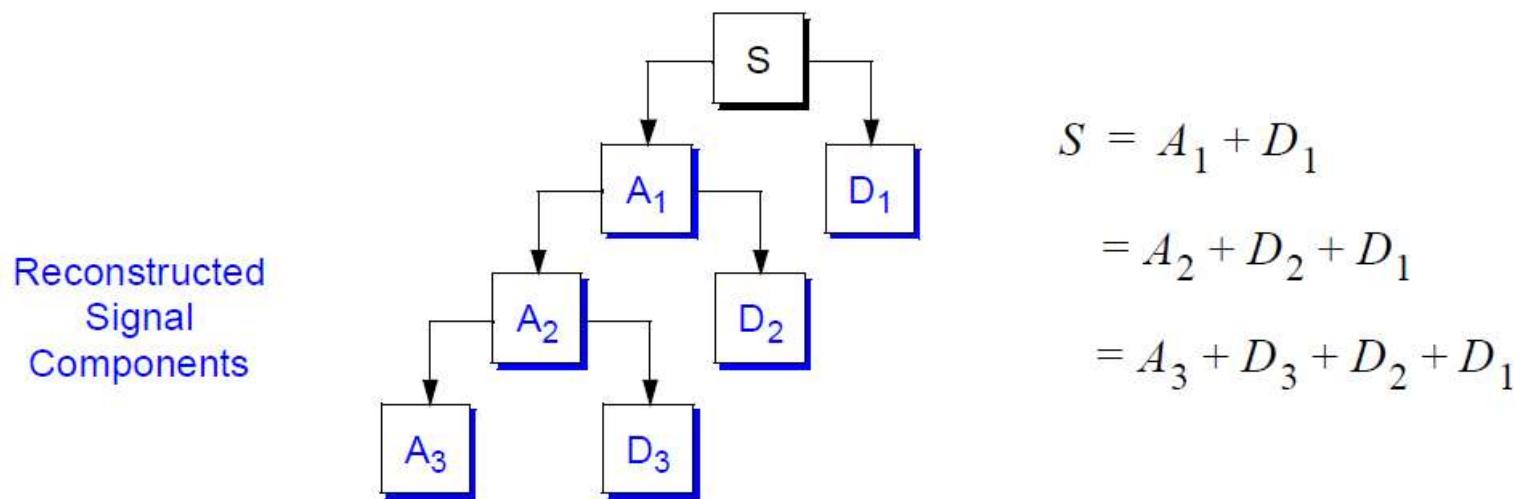
- It is also possible to reconstruct the approximation A and detail D themselves **from their coefficient vectors** c_A and c_D .



- And $S = A_1 + D_1$.

4. Discrete Wavelet Transform

- Extending this technique to a multi-level analysis.



4. Discrete Wavelet Transform

- The choice of **filters** not only determines whether **perfect reconstruction** is possible, it also determines the **shape of the wavelet** we use to perform the analysis.
- To **construct** a **wavelet** of some practical utility, you **seldom** start by **drawing** a **waveform**.
- Instead, it usually makes more sense to **design** the appropriate **filters** and then use them to **create** the **waveform**.

4. Discrete Wavelet Transform

- Example:

- Starting from the reconstruction low-pass filter L'

```
[L, H, Lprime, Hprime] = wfilters('db2')
```

```
Lprime = [0.4830 0.8365 0.2241 -0.1294]
```

- If we reverse the order L' and then multiply every second sample by -1 , we obtain the highpass filter H' :

```
Hprime = [-0.1294 -0.2242 0.8365 -0.4830]
```

- Next, upsample H' by two, inserting zeros in alternate positions:

```
HU = [-0.1294 0 -0.2241 0 0.8365 0 -0.4830 0]
```

4. Discrete Wavelet Transform

- Example:

- Convolve the upsampled vector with the original lowpass filter:

```
H2 = conv(HU,Lprime);
```

- If we iterate this process several more times, repeatedly upsampling and convolving the resultant vector with the four-element filter vector Lprime, a pattern begins to emerge.
 - Scale the final result by $(\sqrt{2})^i$, where i is the number of iterations.
- This result shows that the wavelet's shape is determined entirely by the coefficients of the reconstruction filters.

4. Discrete Wavelet Transform

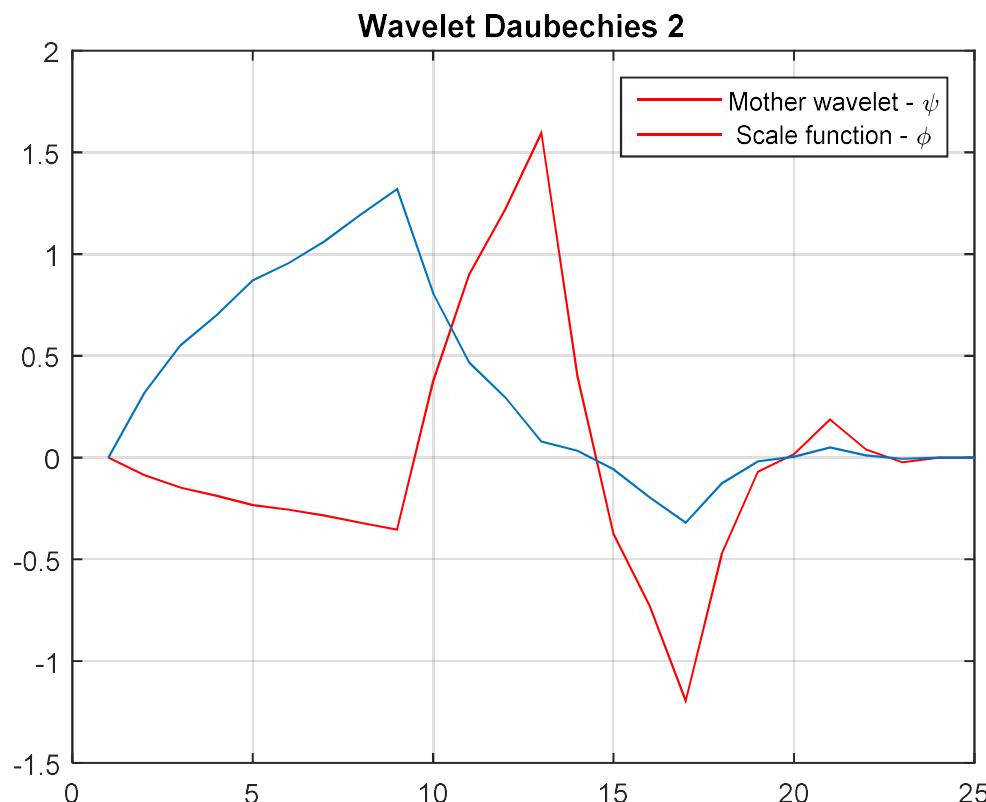
- This relationship has profound implications.
- It means that you cannot choose just any shape, call it a wavelet, and perform an analysis.
- At least, you can't choose an arbitrary wavelet waveform if you want to be able to reconstruct the original signal accurately.
- You are compelled to choose a shape determined by quadrature mirror decomposition filters.

4. Discrete Wavelet Transform

- The **mother wavelet** function ψ is determined by the highpass filter, which also produces the **details** of the wavelet **decomposition**.
- There is an additional function associated with some but not all wavelets. This is the so-called **scaling function**, ϕ .
- It is determined by the lowpass quadrature mirror filters, and thus is associated with the **approximations** of the wavelet **decomposition**.
- Iteratively upsampling and convolving the reconstruction lowpass filter produces a shape approximating the **scaling function**.

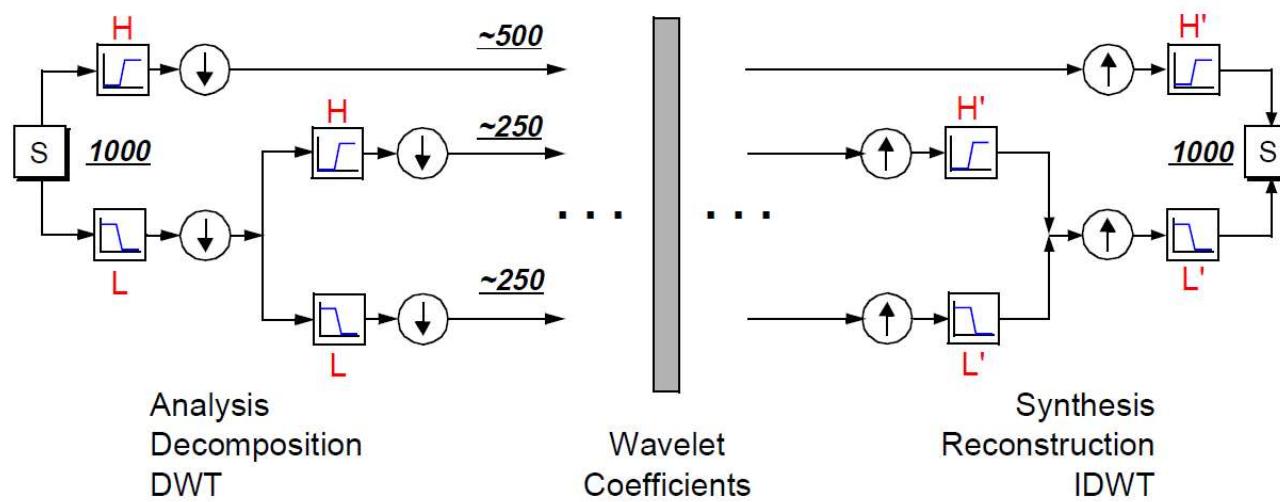
4. Discrete Wavelet Transform

- MATLAB: s53WaveFilters.m



4. Discrete Wavelet Transform

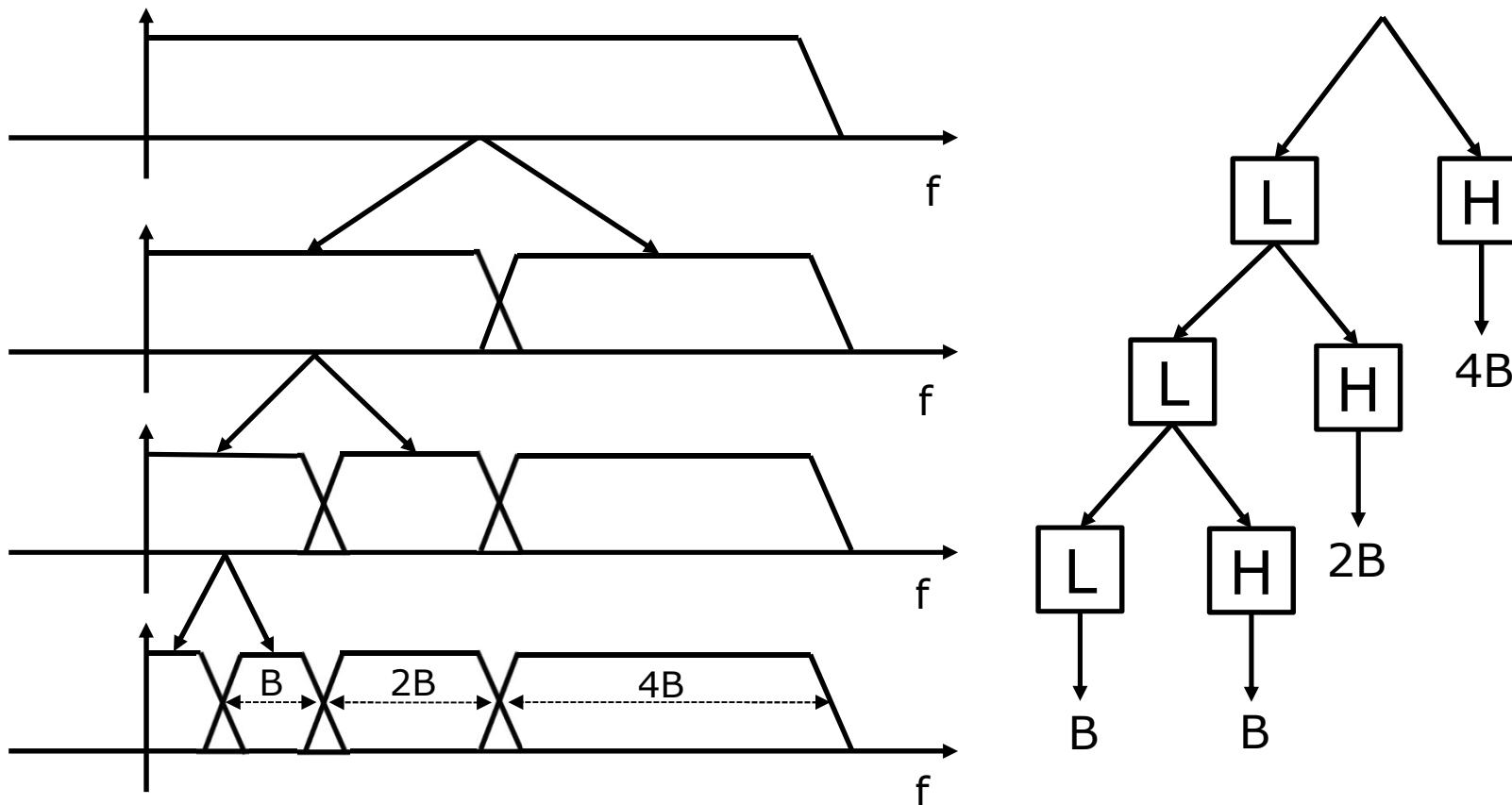
- Multistep analysis-synthesis:



- Of course, there is no point breaking up a signal merely to have the satisfaction of immediately reconstructing it.
- We perform wavelet analysis because the coefficients thus obtained have many known uses, de-noising and compression being foremost among them.

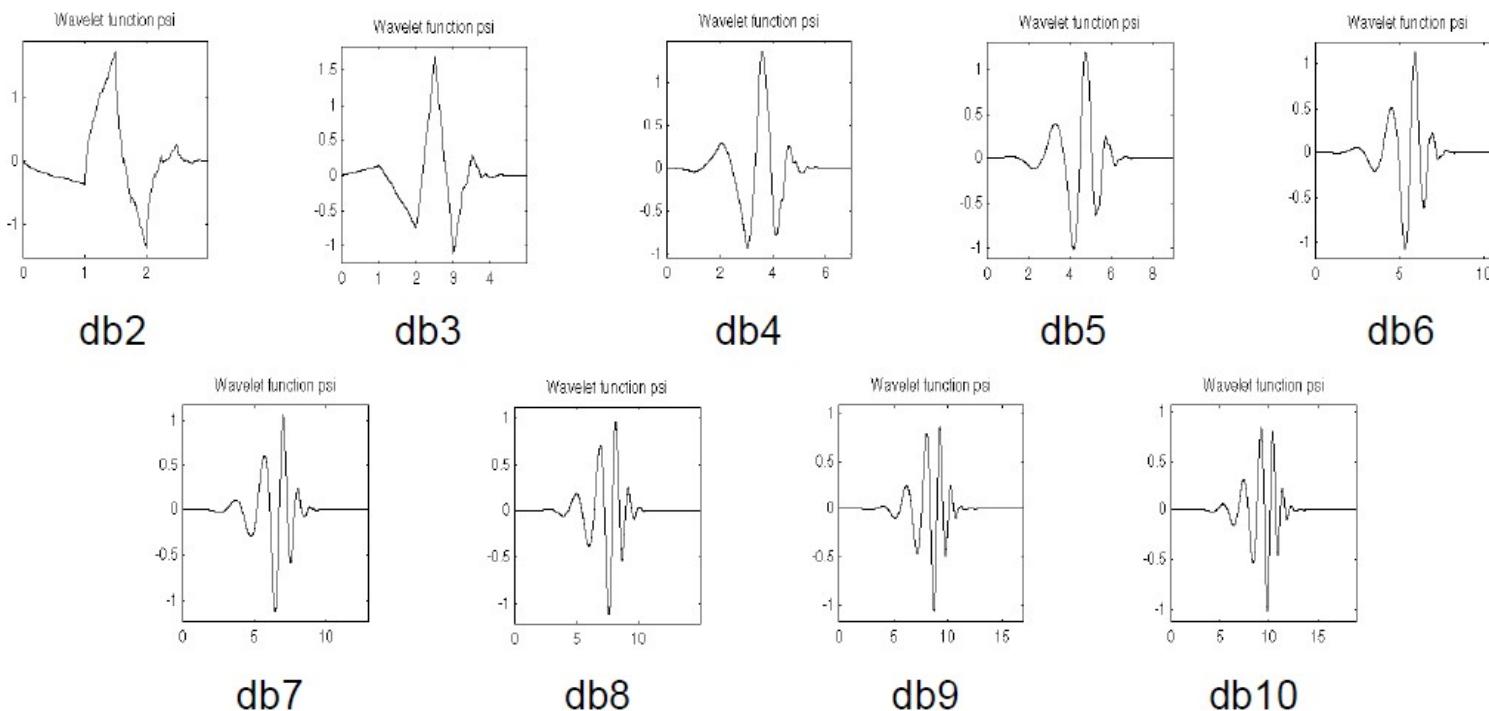
4. Discrete Wavelet Transform

- Multistep analysis-synthesis:



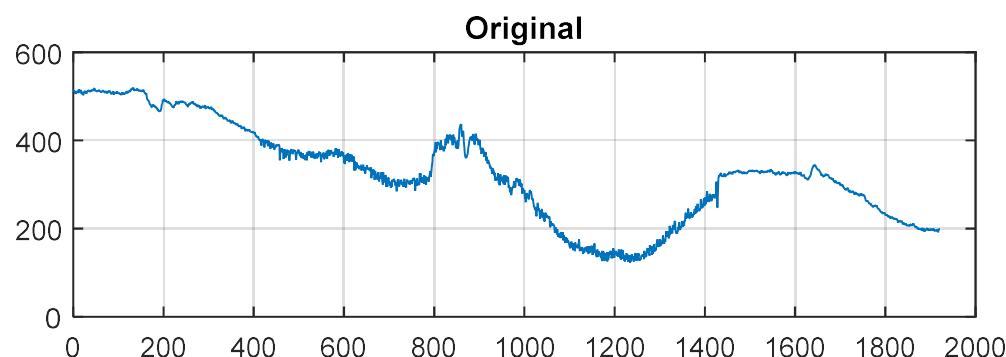
4. Discrete Wavelet Transform

- Examples of wavelets: nine members of the Daubechies family.



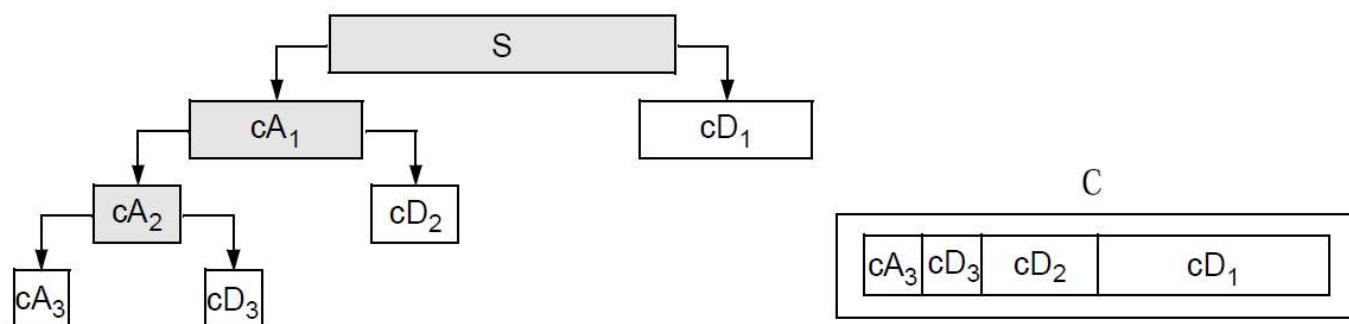
4. Discrete Wavelet Transform

- Example of One-Dimensional Analysis:
 - This example involves a real-world signal — electrical consumption measured over the course of three days.
 - This signal is particularly interesting because of noise introduced when a defect developed in the monitoring equipment as the measurements were being made.
 - Wavelet analysis effectively removes the noise.



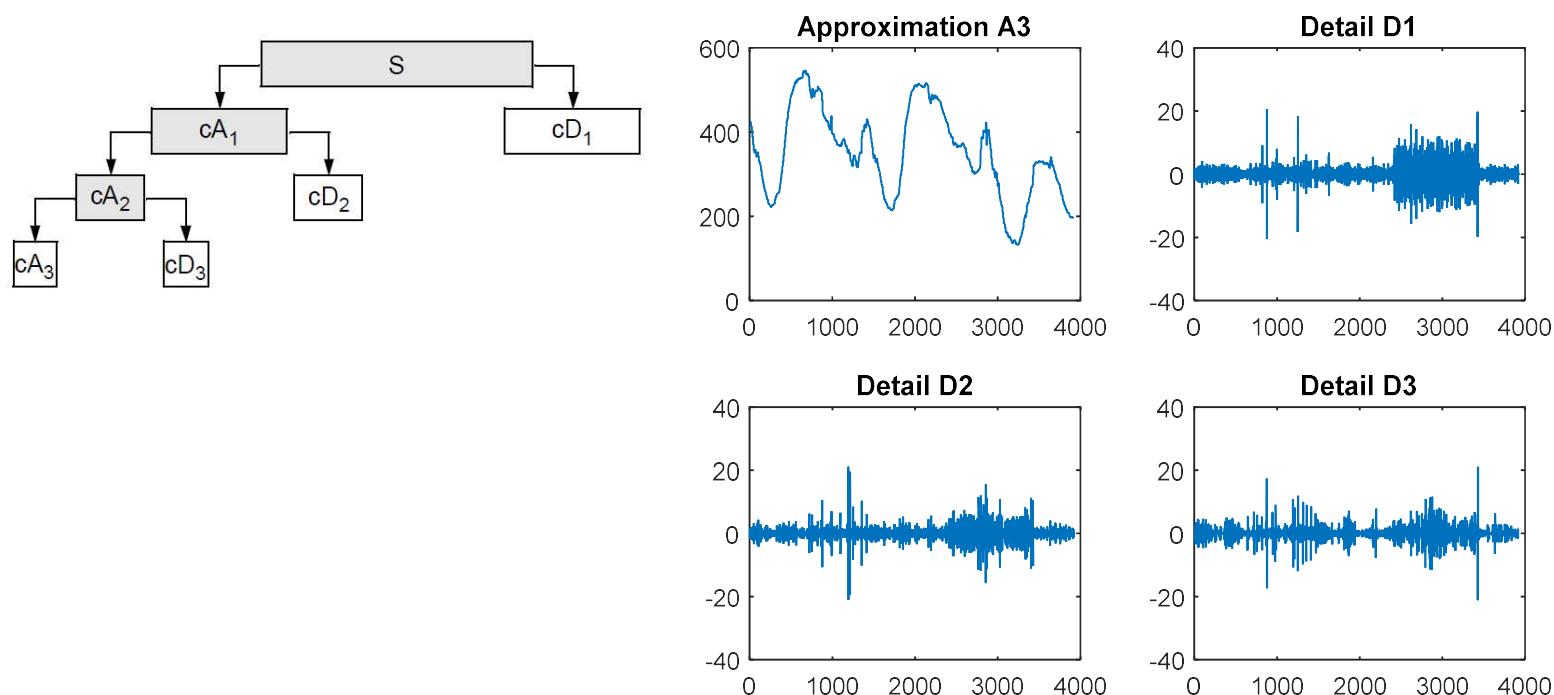
4. Discrete Wavelet Transform

- Example of One-Dimensional Analysis:
 - Level 3 decomposition of the signal (using the 'db3' wavelet).



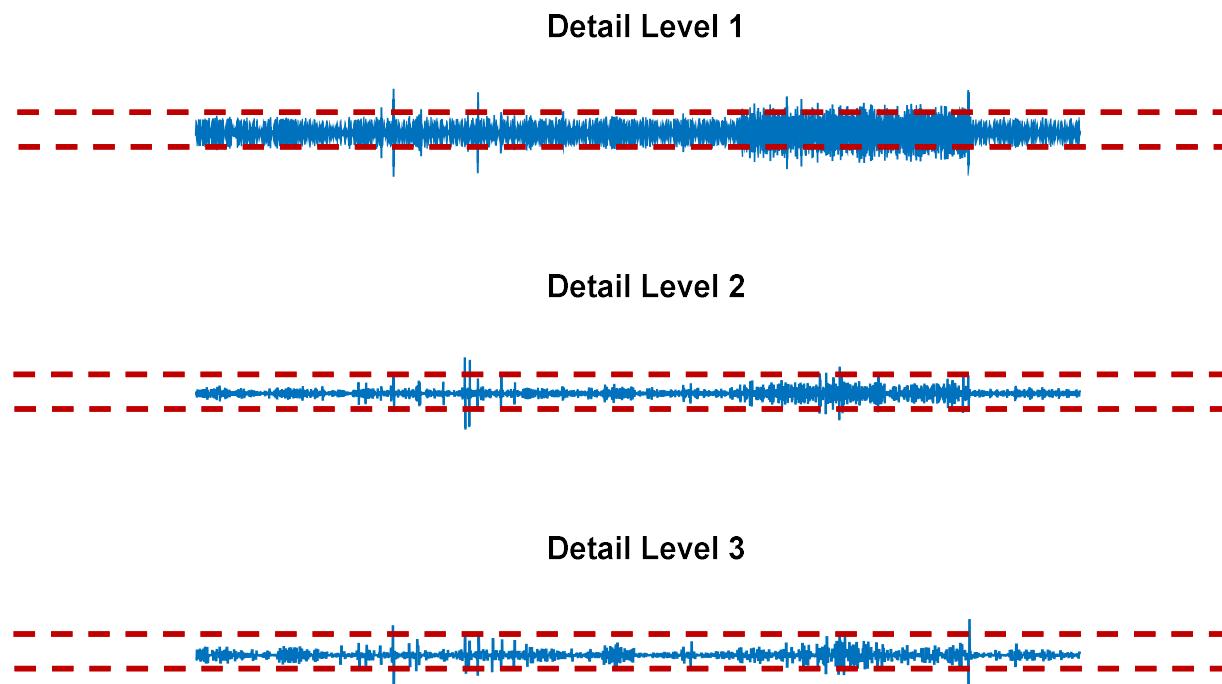
4. Discrete Wavelet Transform

- Example of One-Dimensional Analysis:
 - Level 3 decomposition of the signal (using the 'db3' wavelet).



4. Discrete Wavelet Transform

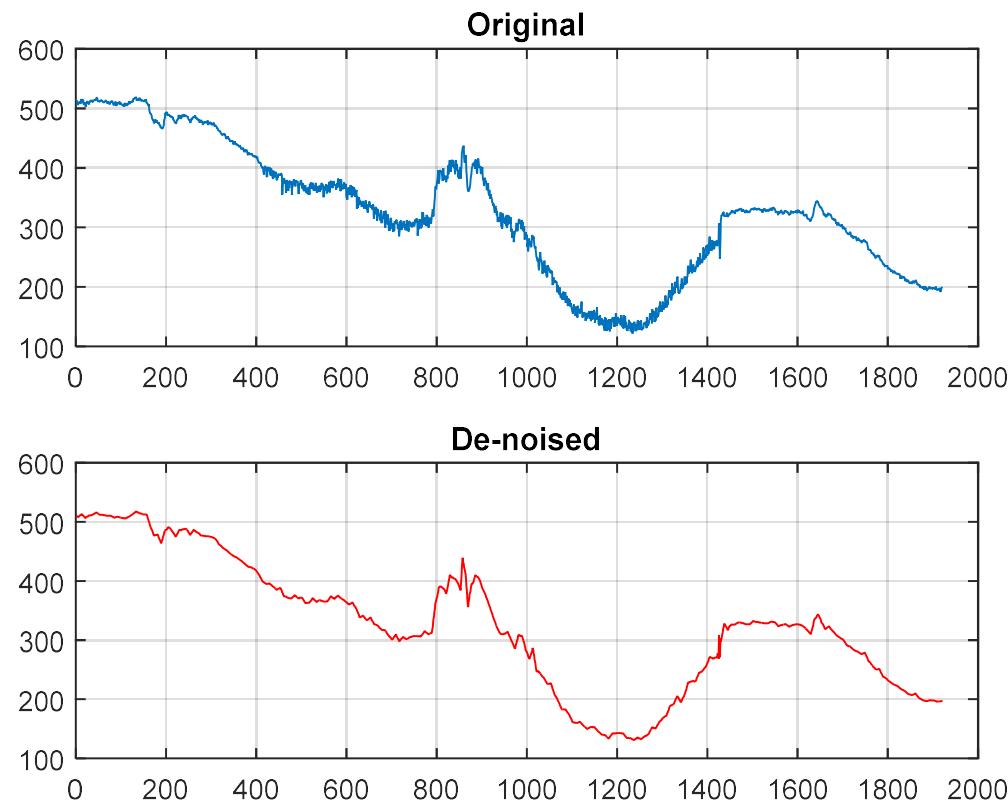
- Example of One-Dimensional Analysis:
 - Setting a threshold



4. Discrete Wavelet Transform

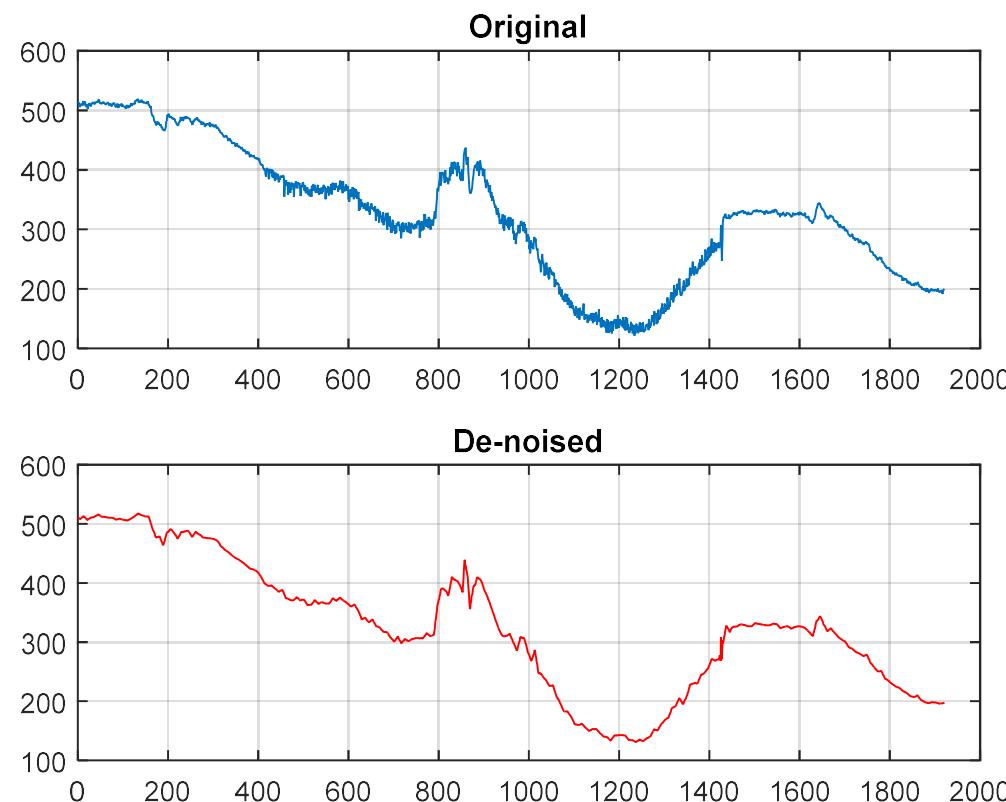
- Example of One-Dimensional Analysis:

➤ Reconstruction



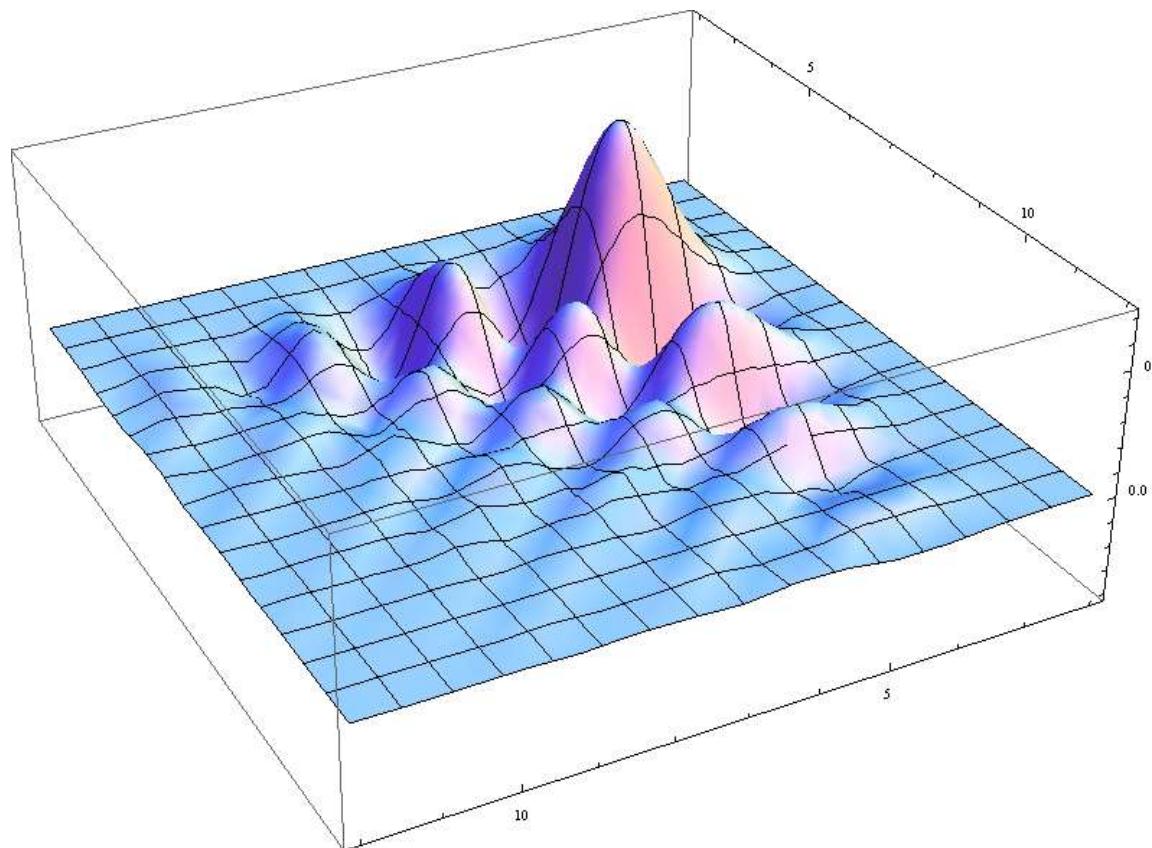
4. Discrete Wavelet Transform

- MATLAB: s61Denoise1D.m



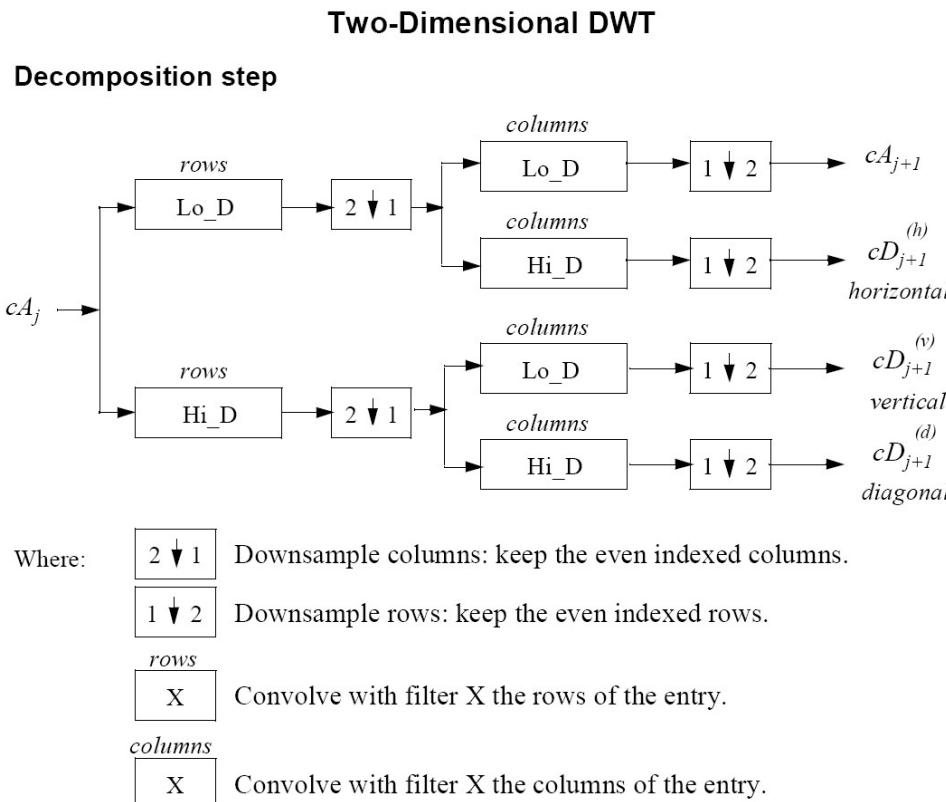
4. Discrete Wavelet Transform

- Two-Dimensional Analysis



4. Discrete Wavelet Transform

- Two-Dimensional Analysis

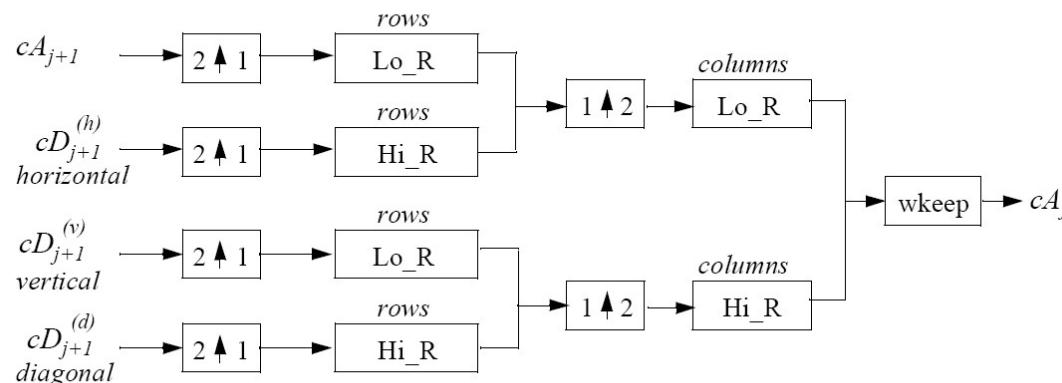


4. Discrete Wavelet Transform

- Two-Dimensional Analysis

Two-Dimensional IDWT

Reconstruction step



Where:



Upsample columns: insert zeros at odd-indexed columns.



Upsample rows: insert zeros at odd-indexed rows.



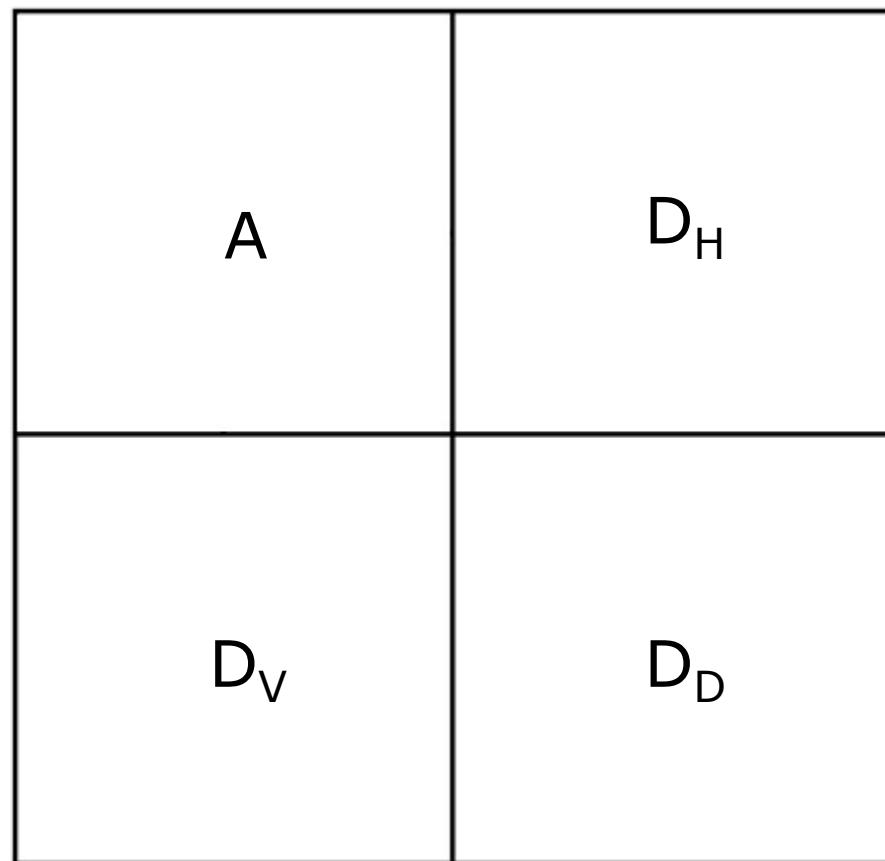
Convolve with filter X the rows of the entry.



Convolve with filter X the columns of the entry.

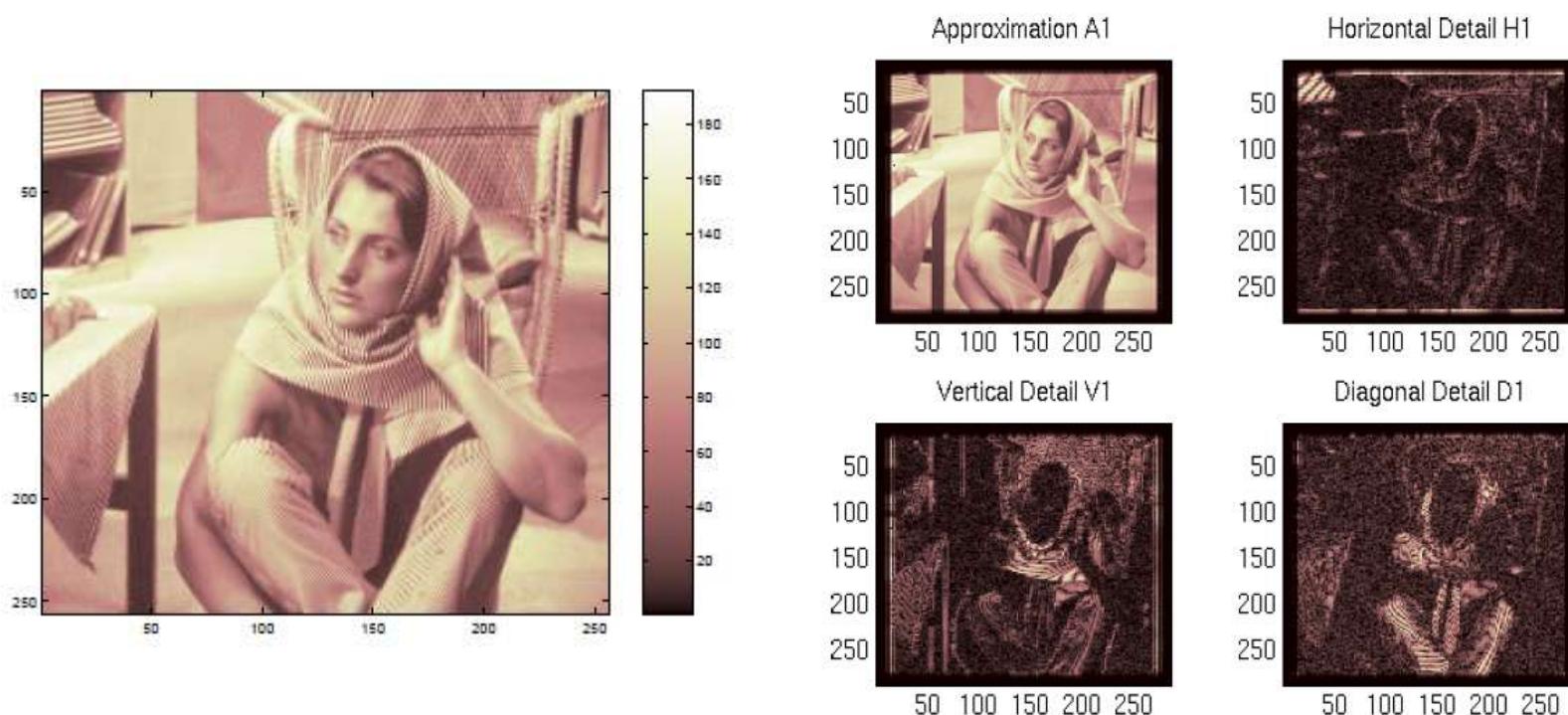
4. Discrete Wavelet Transform

- Two-Dimensional Analysis (one-step decomposition)



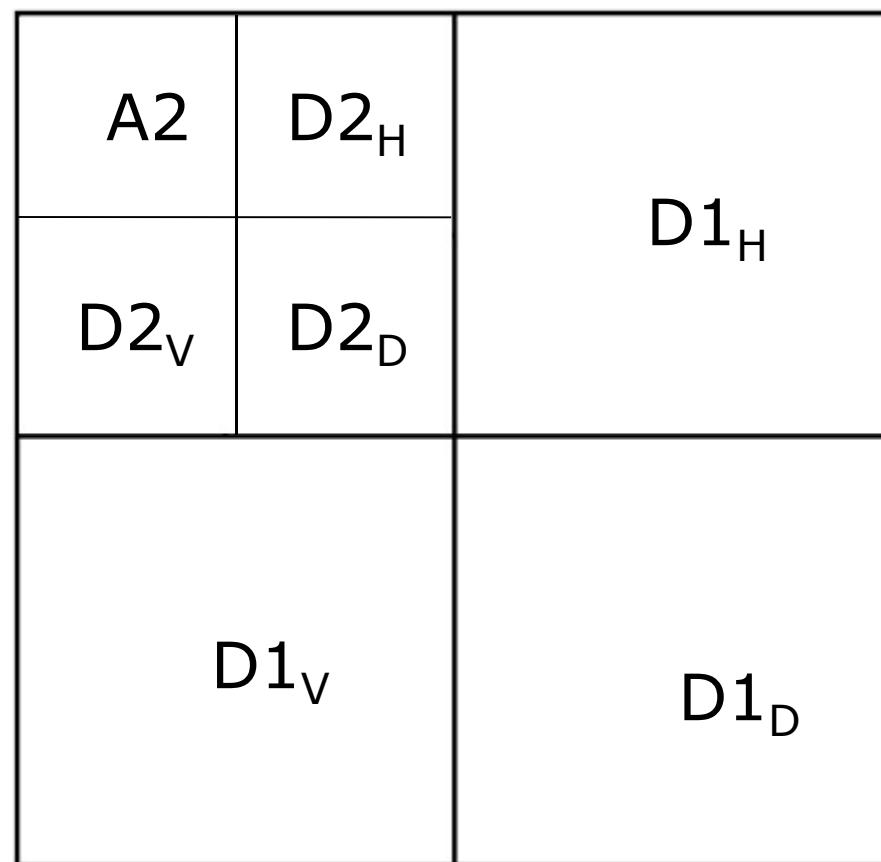
4. Discrete Wavelet Transform

- Two-Dimensional Analysis (one-step decomposition)



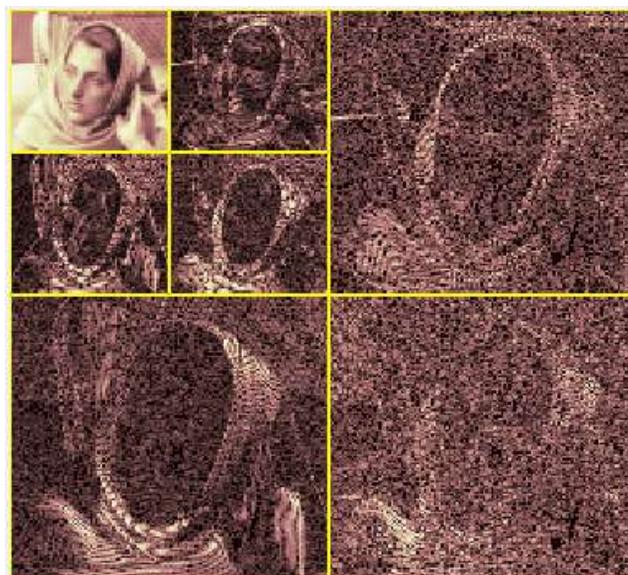
4. Discrete Wavelet Transform

- Two-Dimensional Analysis (multiple-level decomposition)



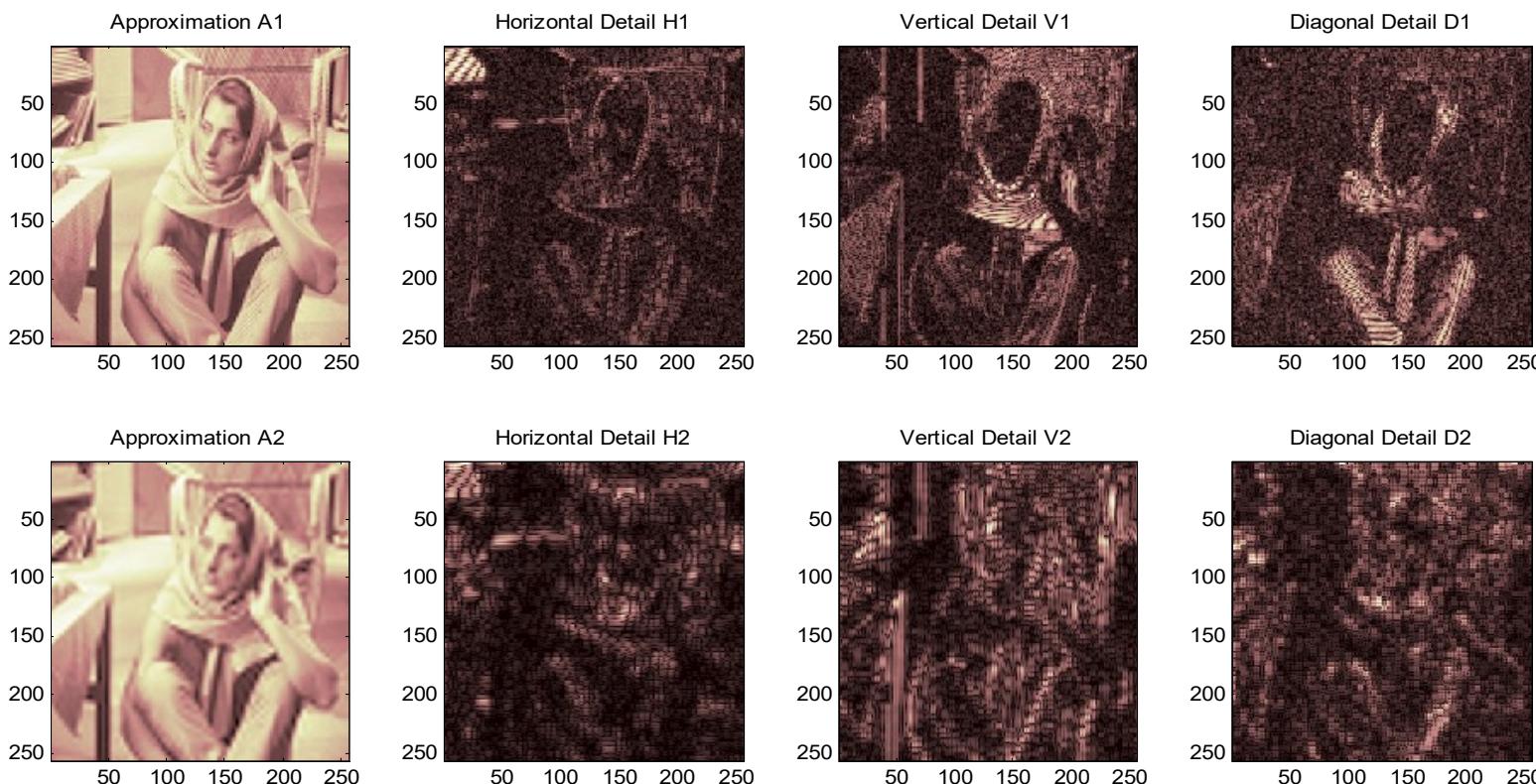
4. Discrete Wavelet Transform

- Two-Dimensional Analysis (multiple-level decomposition)



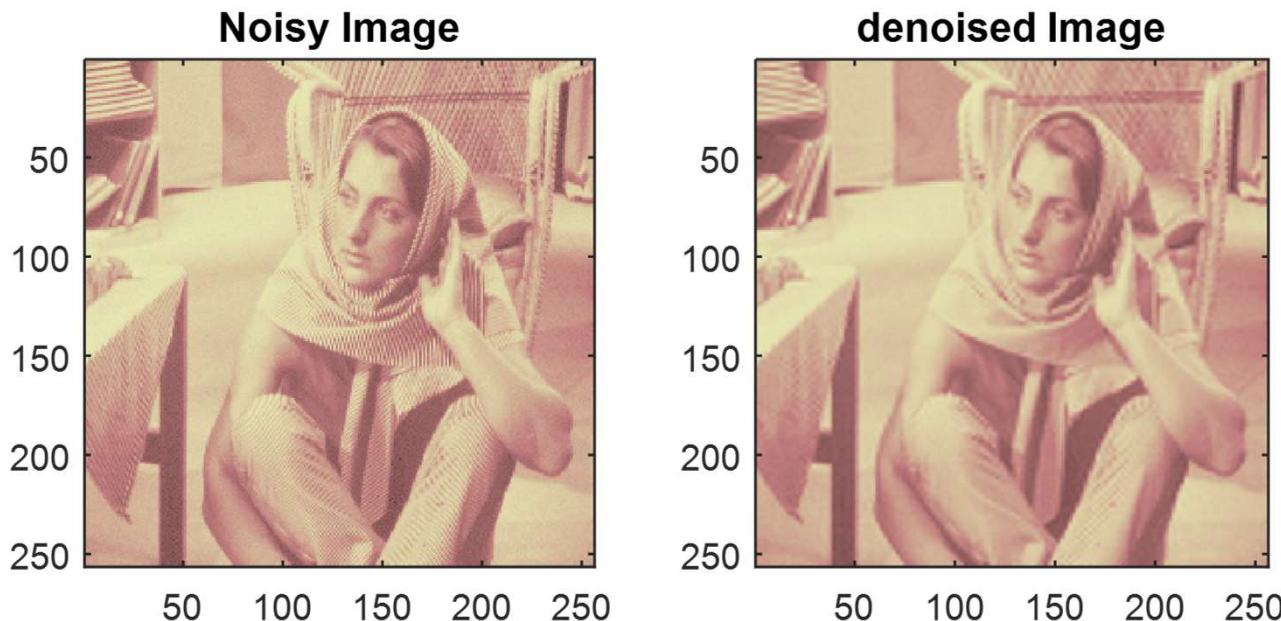
4. Discrete Wavelet Transform

- Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition)



4. Discrete Wavelet Transform

- Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition): s72Denoise2Da.m



4. Discrete Wavelet Transform

- Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition): s73Denoise2Db.m

Noisy Image

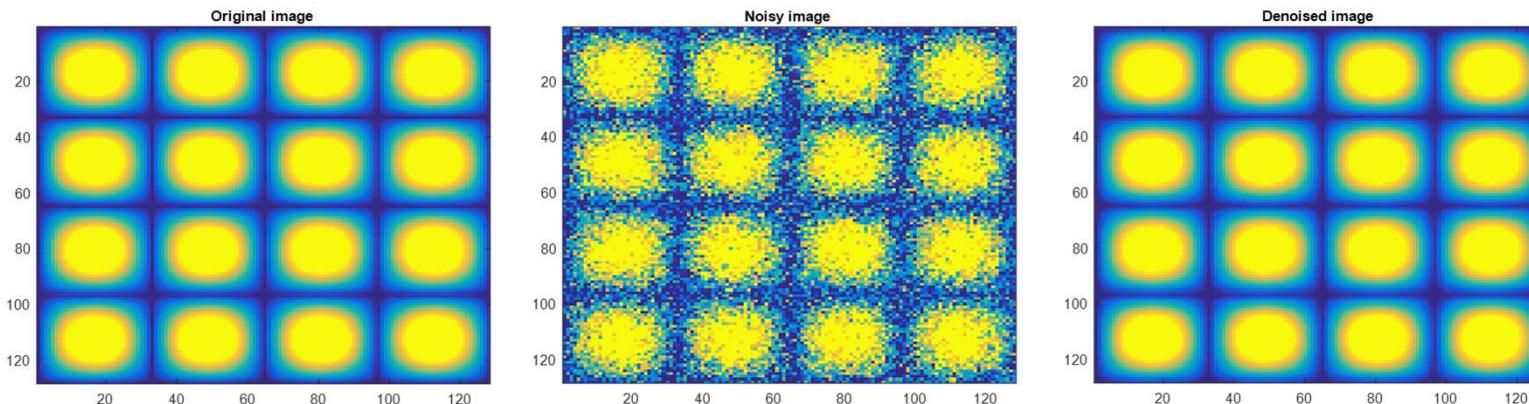


Denoised Image

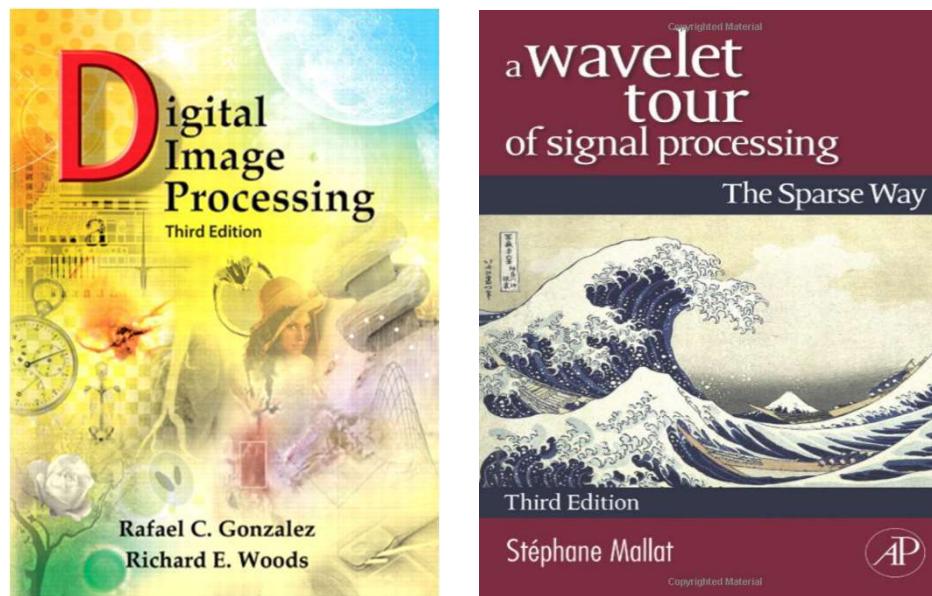


4. Discrete Wavelet Transform

- Example of Two-Dimensional Analysis (Extension to Image Denoising, Two-step Decomposition): s74Denoise2Dc.m



5. Further Reading



Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 08

Image Coding

1. Fundamentals

- To better understand the need for compact image representations, consider the amount of data required to represent a **two-hour** standard definition (**SD**) television movie using $720 \times 480 \times 24$ bit pixel arrays,

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

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

or 224 GB (gigabytes) of data. Twenty-seven 8.5 GB dual-layer DVDs are needed to store it.

1. Fundamentals

- To put a two-hour movie on a single DVD, each frame must be compressed—on average—by a factor of 26.3.
- The compression must be even higher for high definition (**HD**) television, where image resolutions reach 1920x1080x24 bits/image.
- Web page images, high-resolution digital camera photos, streamed video also are compressed routinely to save storage space and reduce transmission time.
- The term **data compression** refers to the process of reducing the amount of data required to represent a given quantity of information.
- In this definition, **data** and **information** are not the same thing.

1. Fundamentals

- Data are the means by which information is conveyed. Various amounts of data can be used to represent the same amount of information.
- Representations that contain irrelevant or repeated information are said to contain **redundant data**.
- If we let b and b' denote the number of data bits in two representations of the same information, the **relative data redundancy** of the representation with bits is

$$R = 1 - \frac{1}{C}$$

where C , commonly called the **compression ratio**, is defined as

$$C = \frac{b}{b'}$$

1. Fundamentals

- Two-dimensional intensity arrays suffer from three principal types of data redundancies that can be identified and exploited:
 1. **Coding redundancy**: more bits than necessary are used to represent the intensities in 2-D intensity arrays.
 2. **Interpixel redundancy**: pixels of most 2-D intensity arrays are correlated spatially (i.e., each pixel is similar to or dependent on neighboring pixels); temporally correlated pixels (i.e., those similar to or dependent on pixels in nearby frames) also replicate information.
 3. **Psychovisual redundancy (Irrelevant information)**: most 2-D intensity arrays contain information that is ignored by the human visual system.

1. Fundamentals

- Two-dimensional intensity arrays suffer from three principal types of data redundancies that can be identified and exploited.



a b c

FIGURE 8.1 Computer generated $256 \times 256 \times 8$ bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy but may exhibit others as well.)

1. Fundamentals

- Coding Redundancy

- Consider the probability mass function defined in "Topic 03 - Intensity Transformation and Spatial Filtering"

$$p_r(r_k) = \frac{n_k}{MN} \quad k = 0, 1, 2, \dots, L - 1$$

- The average number of bits required to represent each pixel is

$$L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

where $l(r_k)$ the number of bits used to represent each value of r_k .

1. Fundamentals

- Coding Redundancy
 - The total number of bits required to represent an image is $M \times N \times L_{avg}$.
 - If the intensities are represented using a natural **m-bit fixed-length code** (code 1), then $L_{avg} = m$.
 - The computer-generated image in Fig. 8.1(a) has the intensity distribution shown in the second column of Table 8.1.
 - If a natural 8-bit binary code (denoted as **code 1** in Table 8.1) is used to represent its 4 possible intensities, the average number of bits for **code 1** is 8 bits.

1. Fundamentals

- Coding Redundancy

- On the other hand, if the scheme designated as **code 2** in Table 8.1 is used

$$L_{\text{avg}} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$

r_k	$p_r(r_k)$	Code 1	$I_1(r_k)$	Code 2	$I_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

TABLE 8.1
Example of variable-length coding.

1. Fundamentals

- Coding Redundancy

- The total number of bits needed to represent the entire image is

$$MNL_{\text{avg}} = 256 \times 256 \times 1.81 = 118,621$$

- The resulting **compression rate** and corresponding **relative redundancy** are

$$C = \frac{256 \times 256 \times 8}{118,621} = \frac{8}{1.81} \approx 4.42$$

$$R = 1 - \frac{1}{4.42} = 0.774$$

- Thus **77.4%** of the data in the original 8-bit 2-D intensity array is **redundant**.

1. Fundamentals

- Coding Redundancy
 - The compression achieved by code 2 results from assigning **fewer bits** to the **more probable** intensity values than to the less probable ones.
 - Note that the best fixed-length code that can be used in this example is the natural **2-bit** counting sequence.
 - But the resulting compression is only or 4:1—about 10% less than the 4.42:1 compression of the variable-length code.
 - As the preceding example shows, coding redundancy is present when the codes assigned to a set of events (such as intensity values) do not take full advantage of the probabilities of the events.

1. Fundamentals

- Spatial and Temporal Redundancy
 - Consider the computer-generated collection of constant intensity lines in Fig. 8.1(b).
 1. All 256 intensities are equally probable.
 2. Because the intensity of each line was selected randomly, its pixels are independent of one another in the vertical direction
 3. Because the pixels along each line are identical, they are maximally correlated in the horizontal direction.

1. Fundamentals

- Spatial and Temporal Redundancy
 - Observation 1 tells us that the image in Fig. 8.1(b) cannot be compressed by variable length coding alone.
 - Observations 2 and 3 reveal a significant spatial redundancy that can be eliminated using a sequence of **run-length pairs**.
 - ✓ Each run-length pair specifies the start of a new intensity and the number of consecutive pixels that have that intensity.
 - ✓ For the image in Fig. 8.1(b), the compression rate is

$$C = (256 \times 256 \times 8)/[(256 + 256) \times 8] \text{ or } 128:1$$

1. Fundamentals

- Spatial and Temporal Redundancy
 - In most images, pixels are correlated spatially and in time.
 - The **information** carried by a **single pixel** is **small**.
 - Much of its visual contribution is redundant in the sense that it **can be inferred** from its **neighbors**.
 - To **reduce the redundancy** associated with spatial and temporal correlation they must be transformed into a more efficient “non-visual” representation.
 - A mapping is said to be **reversible** if the original pixels can be reconstructed without error; otherwise the mapping is said to be **irreversible**.

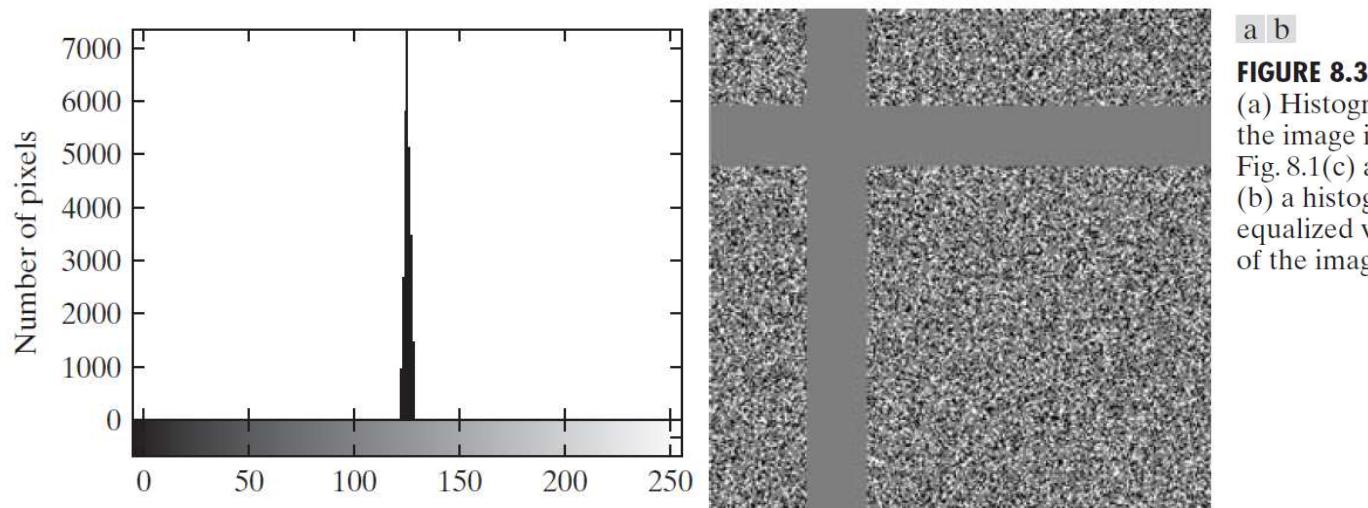
1. Fundamentals

- Psychovisual redundancy (Irrelevant Information)
 - One of the simplest ways to compress a set of data is to remove superfluous data from the set.
 - Information that is **ignored** by the **human visual system** are candidates for **omission**.
 - Thus, the image in Fig. 8.1(c), because it appears to be a homogeneous field of gray, can be represented by its average intensity alone - a single 8-bit value.
 - In this case, the compression rate is

$$C = (256 \times 256 \times 8)/8 \text{ or } 65,536:1$$

1. Fundamentals

- Psychovisual redundancy (Irrelevant Information)
 - However...



a b

FIGURE 8.3
(a) Histogram of the image in Fig. 8.1(c) and (b) a histogram equalized version of the image.

- Because its omission results in a **loss of quantitative information**, its removal is commonly referred to as **quantization**.

1. Fundamentals

- Measuring Image Information

- A question that naturally arises is this: How few bits are actually needed to represent the information in an image?
- **Information theory** provides the mathematical framework to answer this and related questions.
- A random event E with probability $P(E)$ is said to contain

$$I(E) = \log \frac{1}{P(E)} = -\log P(E)$$

units of information

- If $P(E) = 1$ (that is, the event always occurs), $I(E) = 0$.

1. Fundamentals

- Measuring Image Information
 - The **base** of the logarithm determines the **unit** used to measure information.
 - If the **base 2** is selected, the unit of information is the **bit**.
 - Note that if $P(E) = \frac{1}{2}$, $I(E) = 1$ bit.
 - That is, 1 bit is the amount of information conveyed when one of two possible equally likely events occurs.

1. Fundamentals

- Measuring Image Information

- Consider a source of statistically independent random events from a discrete **set** of possible **events** $\{a_1, a_2, \dots, a_J\}$ with associated **probabilities** $\{P(a_1), P(a_2), \dots, P(a_J)\}$,
- The **average information per source output**, called the **entropy** of the source, is

$$H = - \sum_{j=1}^J P(a_j) \log P(a_j)$$

where a_j are called **source symbols**.

- Because they are statistically independent, the source itself is called a **zero-memory source**.

1. Fundamentals

- Measuring Image Information
 - If an image is considered to be the output of an imaginary zero-memory “intensity source,” we can use the histogram of the observed image to estimate the symbol probabilities of the source.
 - Then the intensity source’s entropy becomes

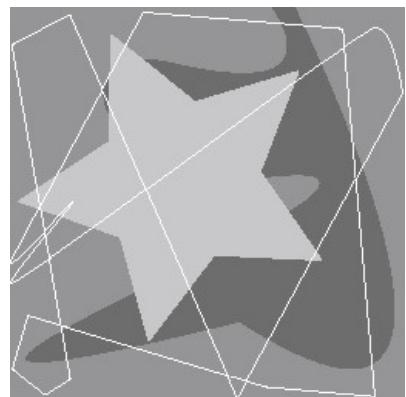
$$\tilde{H} = - \sum_{k=0}^{L-1} p_r(r_k) \log_2 p_r(r_k)$$

- It is **not possible** to **code** the intensity values of the imaginary source (and thus the sample image) **with fewer than \tilde{H}** bits/pixel.

1. Fundamentals

- Measuring Image Information

- The entropy of the image in Fig. 8.1(a) can be estimated by substituting the intensity probabilities from Table 8.1 into previous equation:



r_k	$p_r(r_k)$
$r_{87} = 87$	0.25
$r_{128} = 128$	0.47
$r_{186} = 186$	0.25
$r_{255} = 255$	0.03
r_k for $k \neq 87, 128, 186, 255$	0

$$\begin{aligned}\tilde{H} &= -[0.25 \log_2 0.25 + 0.47 \log_2 0.47 + 0.25 \log_2 0.25 + 0.03 \log_2 0.03] \\ &\approx -[0.25(-2) + 0.47(-1.09) + 0.25(-2) + 0.03(-5.06)] \\ &\approx 1.6614 \text{ bits/pixel}\end{aligned}$$

1. Fundamentals

- Measuring Image Information

- Shannon's first theorem

- ✓ The variable-length code (**code 2**) in Table 8.1 was able to represent the intensities of the image in Fig. 8.1(a) using only 1.81 bits/pixel.

- ✓ Although this is higher than the 1.6614 bits/pixel entropy estimate, Shannon's first theorem — also called the **noiseless coding theorem** (Shannon [1948]) — assures us that the image in Fig. 8.1(a) can be represented with as few as 1.6614 bits/pixel.

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ The **removal** of **irrelevant information** involves a **loss** of real or **quantitative** image information.
 - ✓ Because information is lost, a means of quantifying the nature of the loss is needed.
 - ✓ Two types of criteria can be used for such an assessment: (1) **objective** fidelity criteria and (2) **subjective** fidelity criteria.

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ When information loss can be expressed as a mathematical function, it is said to be based on an **objective** fidelity criterion.
 - An example is the **root-mean-square (rms) error** between two images.

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

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - Another example is the **mean-squared signal-to-noise ratio** (SNR_{ms}),

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

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - But **peak signal-to-noise ratio** (PSNR) is the most commonly objective quality metric used in image processing.

$$\text{PSNR} = 10 \log_{10} \frac{\text{MAX}^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2} \text{ dB}$$

1. Fundamentals

- Measuring Image Information

- Fidelity Criteria

- ✓ While **objective fidelity** criteria offer a simple and convenient way to evaluate information loss, decompressed images are ultimately viewed by humans.
 - ✓ So, measuring image quality by the **subjective evaluations** of people is often more appropriate.
 - ✓ The evaluations may be made using an absolute rating scale or by means of side-by-side comparisons of $\hat{f}(x,y)$ and $f(x,y)$.

1. Fundamentals

- Measuring Image Information

- Fidelity Criteria

- ✓ Absolute rating scale

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

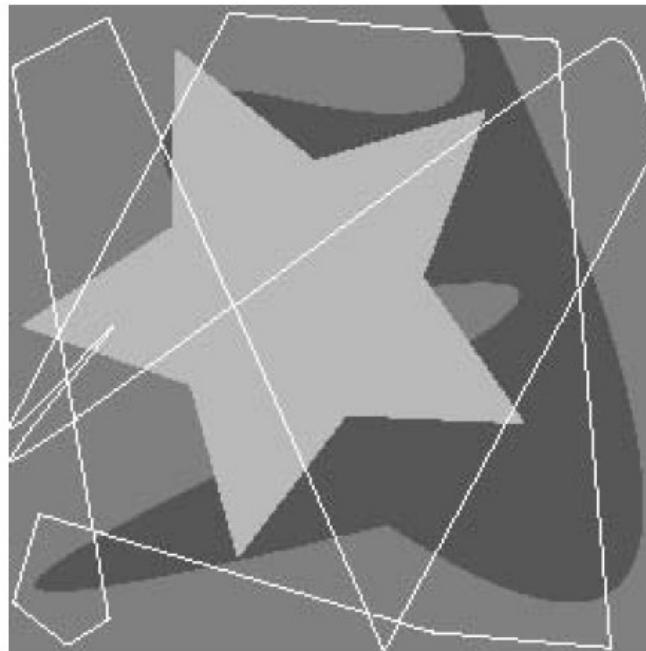
TABLE 8.2
Rating scale of
the Television
Allocations Study
Organization.
(Frendendall and
Behrend.)

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ Side-by-side comparisons of $\hat{f}(x,y)$ and $f(x,y)$.
 - Can be done with a scale such as $\{-3,-2,-1,0,1,2,3\}$ to represent the subjective evaluations:
 - much worse
 - worse
 - slightly worse
 - the same
 - slightly better
 - better
 - much better

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 1 (absolute rating scale)



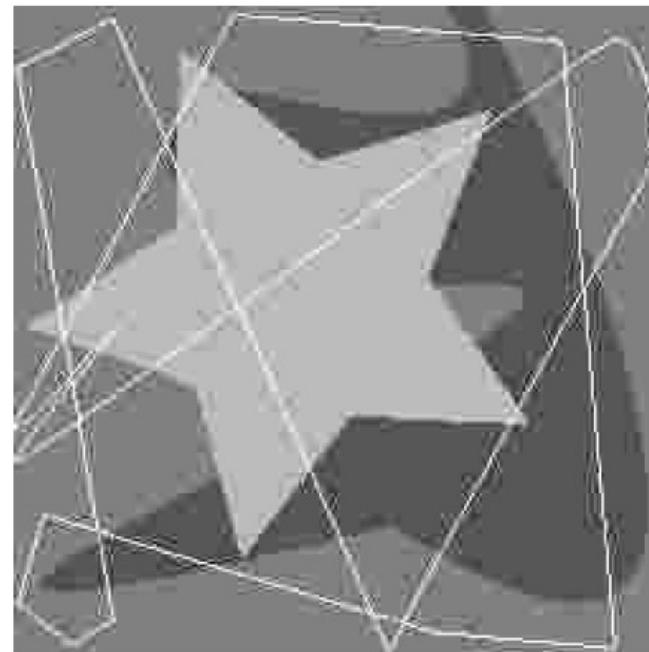
Value	Rating
1	Excellent
2	Fine
3	Passable
4	Marginal
5	Inferior
6	Unusable

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 1 (absolute rating scale)

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 1 (absolute rating scale)



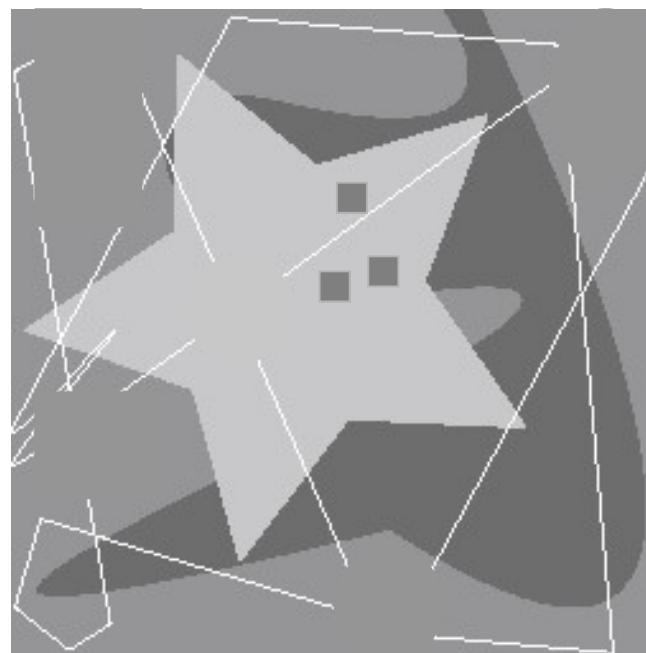
Value	Rating
1	Excellent
2	Fine
3	Passable
4	Marginal
5	Inferior
6	Unusable

1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 1 (absolute rating scale)

1. Fundamentals

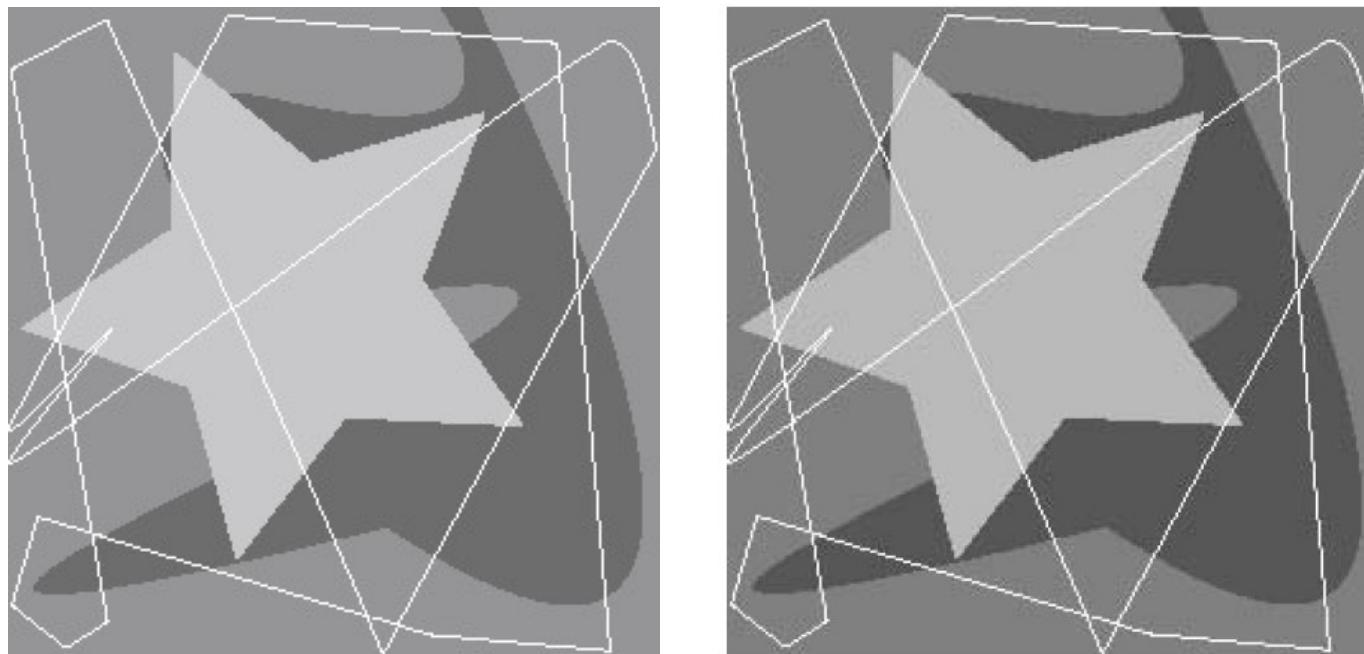
- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 1 (absolute rating scale)



Value	Rating
1	Excellent
2	Fine
3	Passable
4	Marginal
5	Inferior
6	Unusable

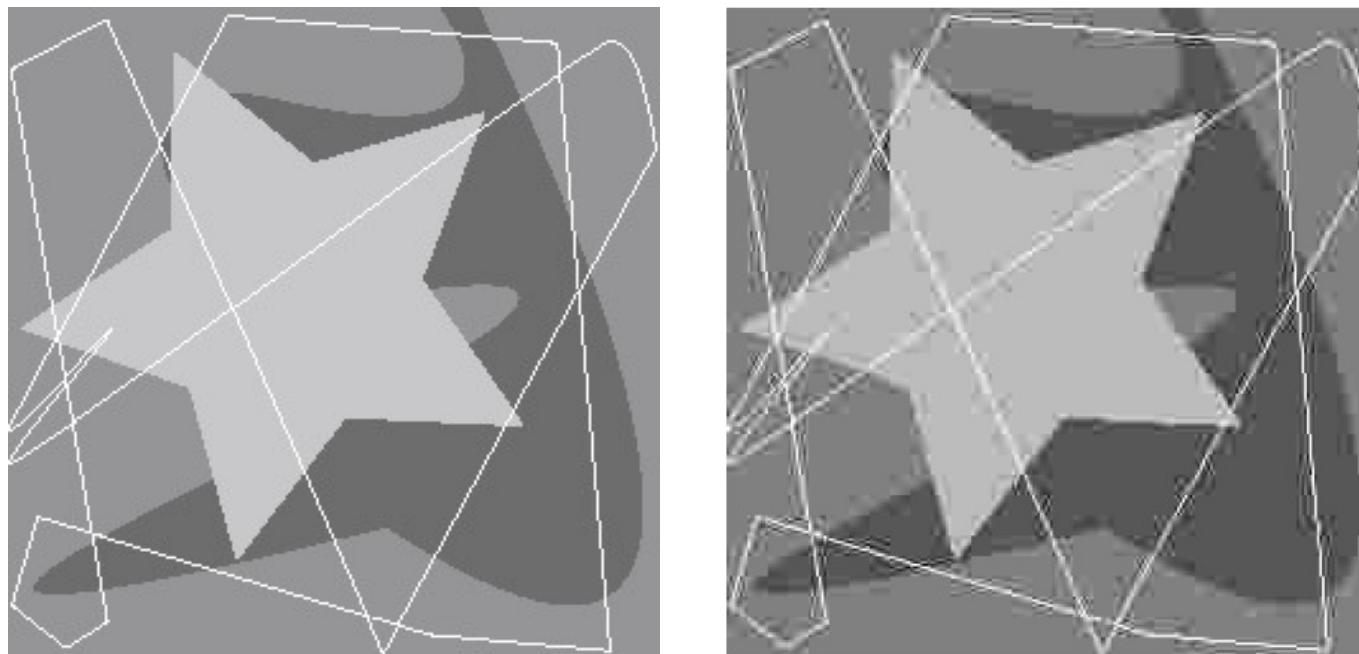
1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 2 (side-by-side)



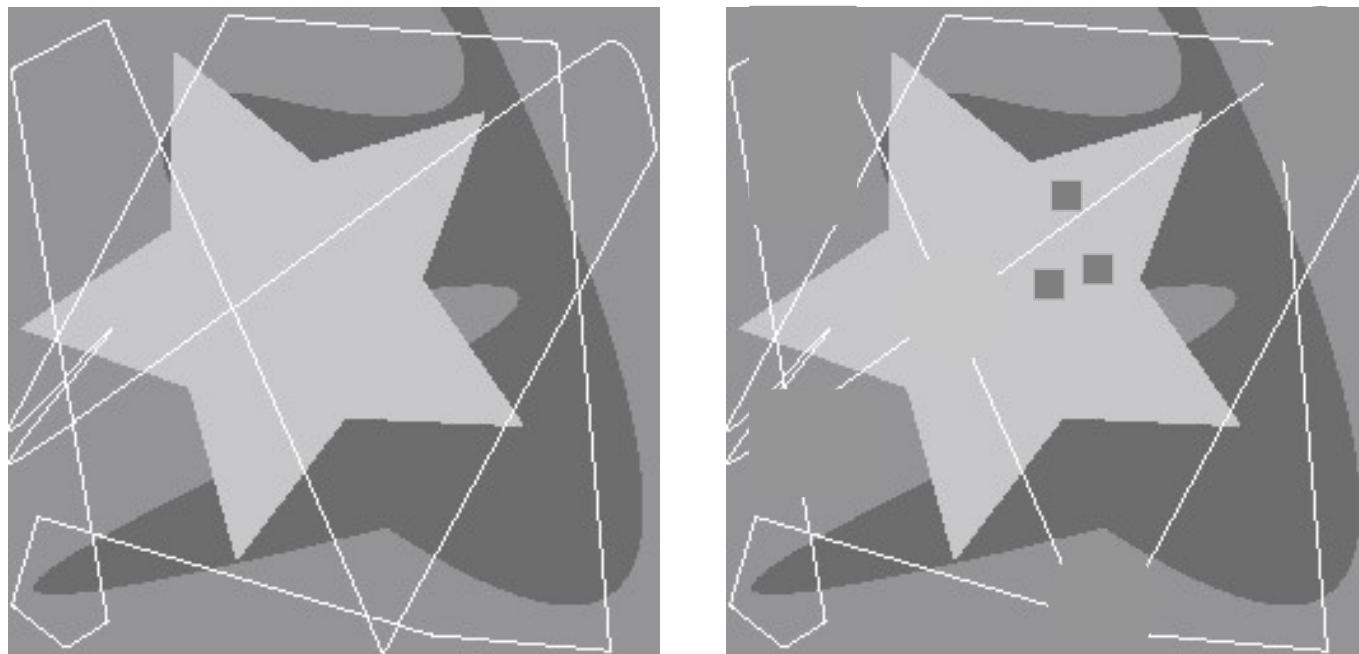
1. Fundamentals

- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 2 (side-by-side)



1. Fundamentals

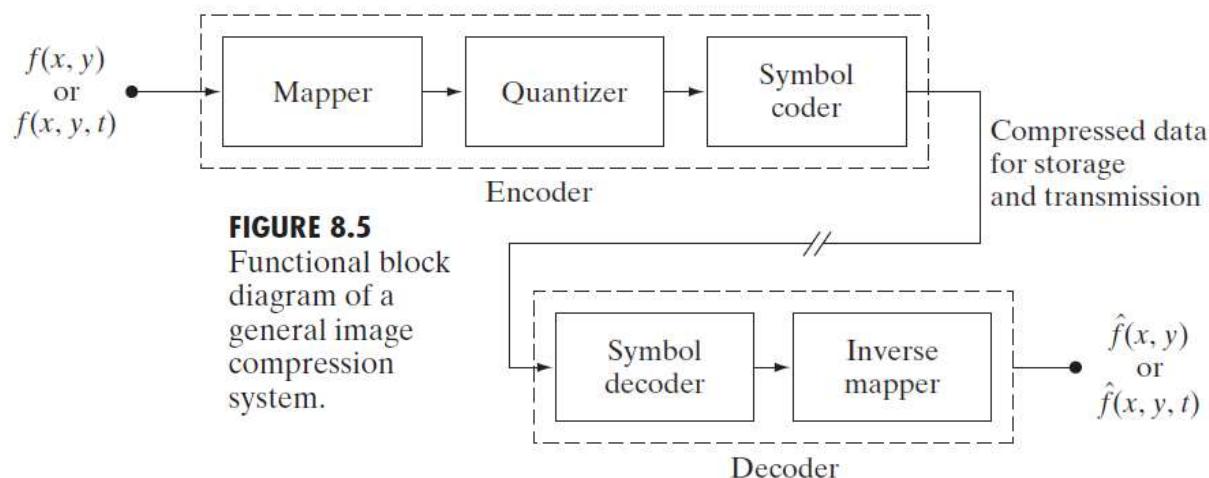
- Measuring Image Information
 - Fidelity Criteria
 - ✓ Example 2 (side-by-side)



1. Fundamentals

- Image Compression Model

- An image compression system is composed of two distinct functional components: an **encoder** and a **decoder**.
- A **codec** is a device or program that is capable of both encoding and decoding.



1. Fundamentals

- Image Compression Model
 - **Mapper:** transforms f into a format designed to reduce spatial and temporal redundancy.
 - **Quantizer:** reduces the accuracy of the mapper's output in accordance with a pre-established fidelity criterion.
 - ✓ The goal is to keep irrelevant information out of the compressed representation.
 - **Symbol coder:** generates a fixed- or variable-length code to represent the quantizer output.
 - Upon its completion, the input image has been processed for the removal of each of the redundancies described before.

1. Fundamentals

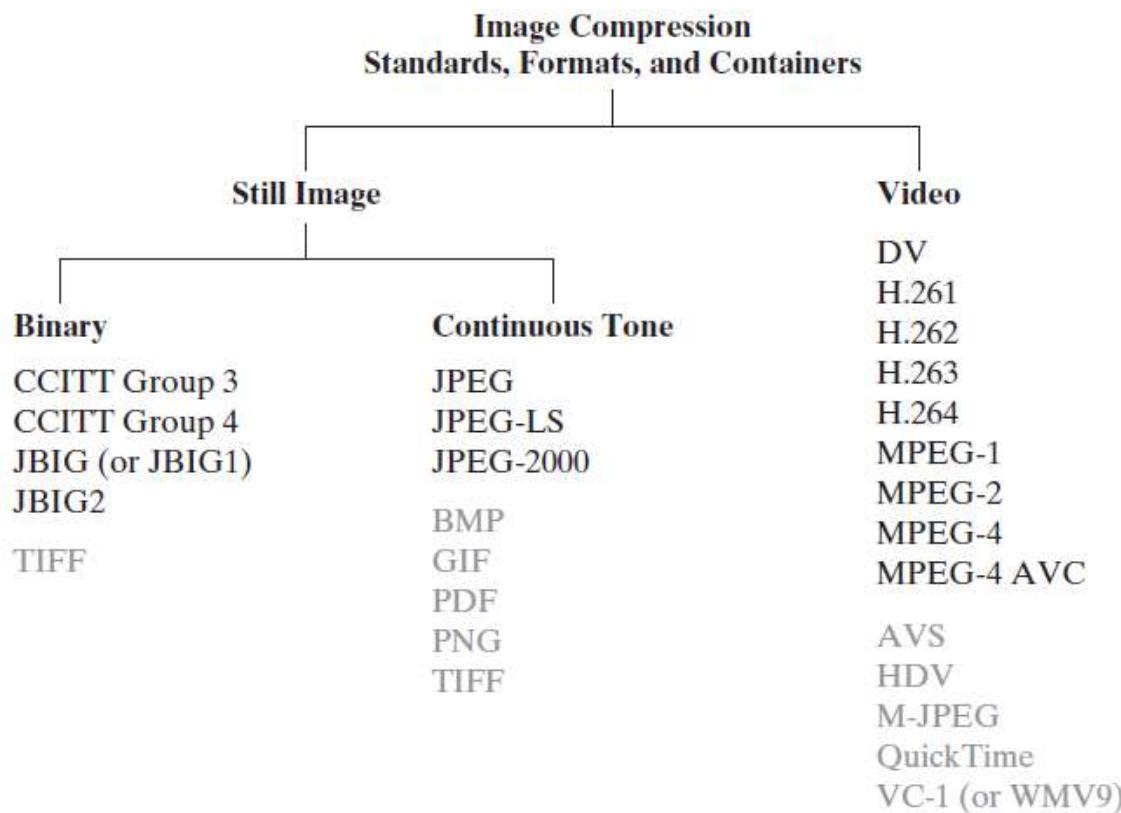
- Image Compression Model
 - **Symbol decoder** and **inverse mapper**: they perform, in reverse order, the inverse operations of the encoder's symbol **encoder** and **mapper**.
 - Because **quantization** results in **irreversible** information **loss**, an inverse quantizer block is **not included** in the general **decoder** model.

1. Fundamentals

- Image Formats, Containers, and Compression Standards
 - **Image file format** is a standard way to organize and store image data.
 - ✓ It defines how the data is arranged and the type of compression—if any—that is used
 - **Image container** is similar to a file format but handles multiple types of image data.
 - **Compression standards** define procedures for compressing and decompressing images.

1. Fundamentals

- Image Formats, Containers, and Compression Standards



1. Fundamentals

- Image Formats, Containers, and Compression Standards
 - The entries in black are international standards sanctioned by the
 - **International Standards Organization** (ISO);
 - **International Electrotechnical Commission** (IEC); and/or
 - **International Telecommunications Union** (ITU-T).

1. Fundamentals

- Image Formats, Containers, and Compression Standards

- Bi-level Still Images

JBIG or JBIG1	ISO/IEC/ ITU-T	A <i>Joint Bi-level Image Experts Group</i> standard for progressive, lossless compression of bi-level images. Continuous-tone images of up to 6 bits/pixel can be coded on a bit-plane basis [8.2.7]. Context sensitive arithmetic coding [8.2.3] is used and an initial low resolution version of the image can be gradually enhanced with additional compressed data.
JBIG2	ISO/IEC/ ITU-T	A follow-on to JBIG1 for bi-level images in desktop, Internet, and FAX applications. The compression method used is content based, with dictionary based methods [8.2.6] for text and halftone regions, and Huffman [8.2.1] or arithmetic coding [8.2.3] for other image content. It can be lossy or lossless.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

- Continuous-tone Still Images

JPEG	ISO/IEC/ ITU-T	A <i>Joint Photographic Experts Group</i> standard for images of photographic quality. Its lossy <i>baseline coding system</i> (most commonly implemented) uses quantized discrete cosine transforms (DCT) on 8×8 image blocks [8.2.8], Huffman [8.2.1], and run-length [8.2.5] coding. It is one of the most popular methods for compressing images on the Internet.
JPEG-LS	ISO/IEC/ ITU-T	A lossless to near-lossless standard for continuous tone images based on adaptive prediction [8.2.9], context modeling [8.2.3], and Golomb coding [8.2.2].
JPEG- 2000	ISO/IEC/ ITU-T	A follow-on to JPEG for increased compression of photographic quality images. Arithmetic coding [8.2.3] and quantized discrete wavelet transforms (DWT) [8.2.10] are used. The compression can be lossy or lossless.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

- Video

MPEG-2	ISO/IEC	An extension of MPEG-1 designed for DVDs with transfer rates to 15 Mb/s. Supports interlaced video and HDTV. It is the most successful video standard to date.
H.264	ITU-T	An extension of H.261–H.263 for videoconferencing, Internet streaming, and television broadcasting. It supports prediction differences within frames [8.2.9], variable block size integer transforms (rather than the DCT), and context adaptive arithmetic coding [8.2.3].
MPEG-4 AVC	ISO/IEC	MPEG-4 Part 10 <i>Advanced Video Coding</i> (AVC). Identical to H.264 above.

1. Fundamentals

- Image Formats, Containers, and Compression Standards

- Continuous-tone Still Images

BMP	Microsoft	<i>Windows Bitmap.</i> A file format used mainly for simple uncompressed images.
GIF	CompuServe	<i>Graphic Interchange Format.</i> A file format that uses lossless LZW coding [8.2.4] for 1- through 8-bit images. It is frequently used to make small animations and short low resolution films for the World Wide Web.
PDF	Adobe Systems	<i>Portable Document Format.</i> A format for representing 2-D documents in a device and resolution independent way. It can function as a container for JPEG, JPEG 2000, CCITT, and other compressed images. Some PDF versions have become ISO standards.

1. Fundamentals

- Image Formats, Containers, and Compression Standards
 - Continuous-tone Still Images

PNG	<i>World Wide Web Consortium</i> (W3C)	<i>Portable Network Graphics.</i> A file format that losslessly compresses full color images with transparency (up to 48 bits/pixel) by coding the difference between each pixel's value and a predicted value based on past pixels [8.2.9].
TIFF	Aldus	<i>Tagged Image File Format.</i> A flexible file format supporting a variety of image compression standards, including JPEG, JPEG-LS, JPEG-2000,JBIG2, and others.

2. Some Basic Compression Methods

- Huffman coding

CCITT
JBIG2
JPEG

MPEG-1,2,4
H.261, H.262,
H.263, H.264

- One of the most popular techniques for removing coding redundancy is due to Huffman.
- The **first step** in Huffman's approach is to create a series of source reductions.

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

2. Some Basic Compression Methods

- Huffman coding

- The **second step** in Huffman's procedure is to code each reduced source, starting with the smallest source and working back to the original source.

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

$$\begin{aligned}
 L_{\text{avg}} &= (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5) \\
 &= 2.2 \text{ bits/pixel}
 \end{aligned}$$

- The entropy of the source is 2.14 *bits/symbol*.

2. Some Basic Compression Methods

- Huffman coding

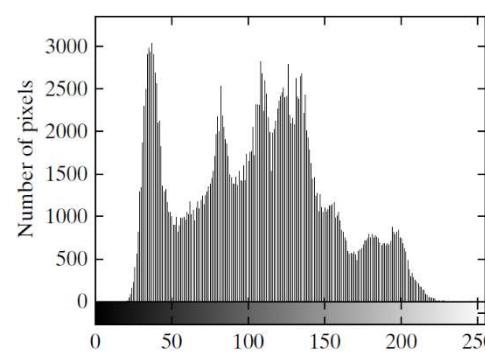
- Huffman's procedure creates the **optimal code** for a set of symbols and probabilities subject to the constraint that the symbols be coded one at a time.
- The code itself is an **instantaneous uniquely decodable block code**.
 - ✓ **Block code**: each source symbol is mapped into a fixed sequence of code symbols.
 - ✓ **Instantaneous**: each **code word**¹ in a string of code symbols can be decoded without referencing succeeding symbols.
 - ✓ **Uniquely decodable**: any string of code symbols can be decoded in only one way.

¹Each piece of information or event is assigned a sequence of code symbols, called a code word.

2. Some Basic Compression Methods

- Huffman coding

➤ Example:



a b

FIGURE 8.9 (a)
A 512×512 8-bit
image, and (b) its
histogram.

- Huffman's procedure was used to encode intensities with 7.428 *bits/pixel* - including the Huffman code table that is required to reconstruct the image.
- The compressed representation exceeds the estimated entropy (7.3838 bits/pixel) only by about 0.6%.

2. Some Basic Compression Methods

- Arithmetic Coding

JBIG1
JBIG2
JPEG-2000
H.264
MPEG-4 AVC

- Unlike the variable-length codes of the previous two sections, arithmetic coding generates **nonblock** codes.
- Example: a five-symbol **message**, $a_1a_2a_3a_3a_4$, from a four-symbol source is coded.
- At the start of the coding process, the **message** is assumed to occupy the entire **half-open interval** $[0, 1)$.
- This interval is subdivided initially into four regions based on the probabilities of each source symbol.

2. Some Basic Compression Methods

- Arithmetic Coding

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

TABLE 8.6
Arithmetic coding example.

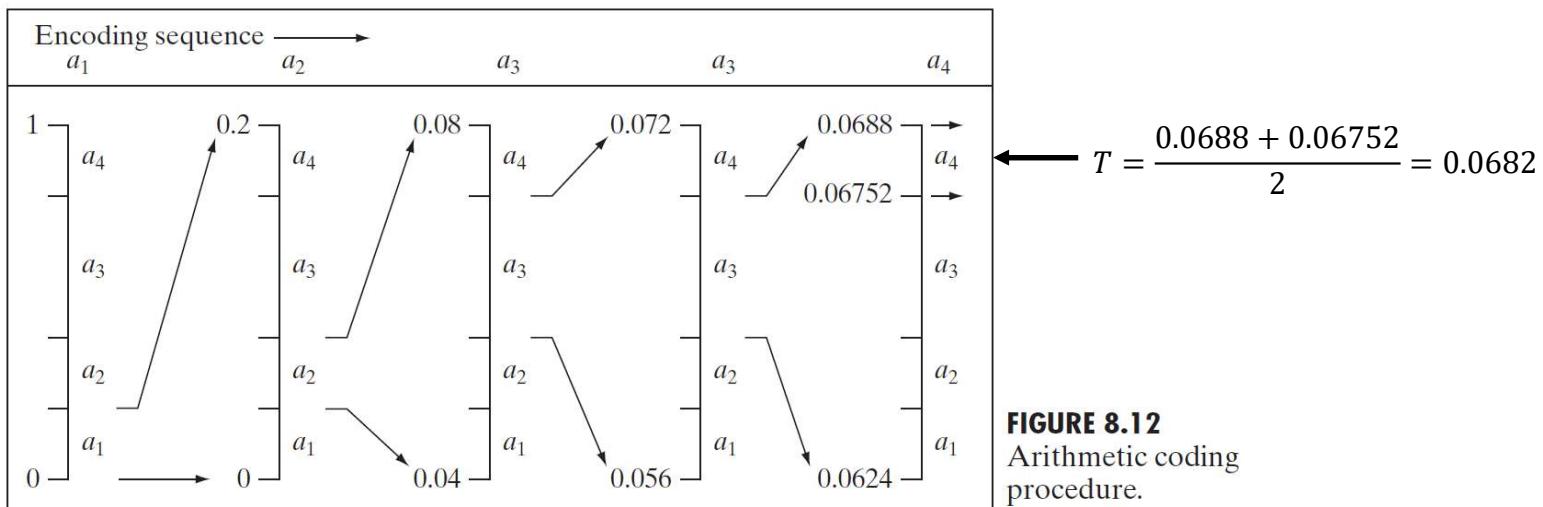


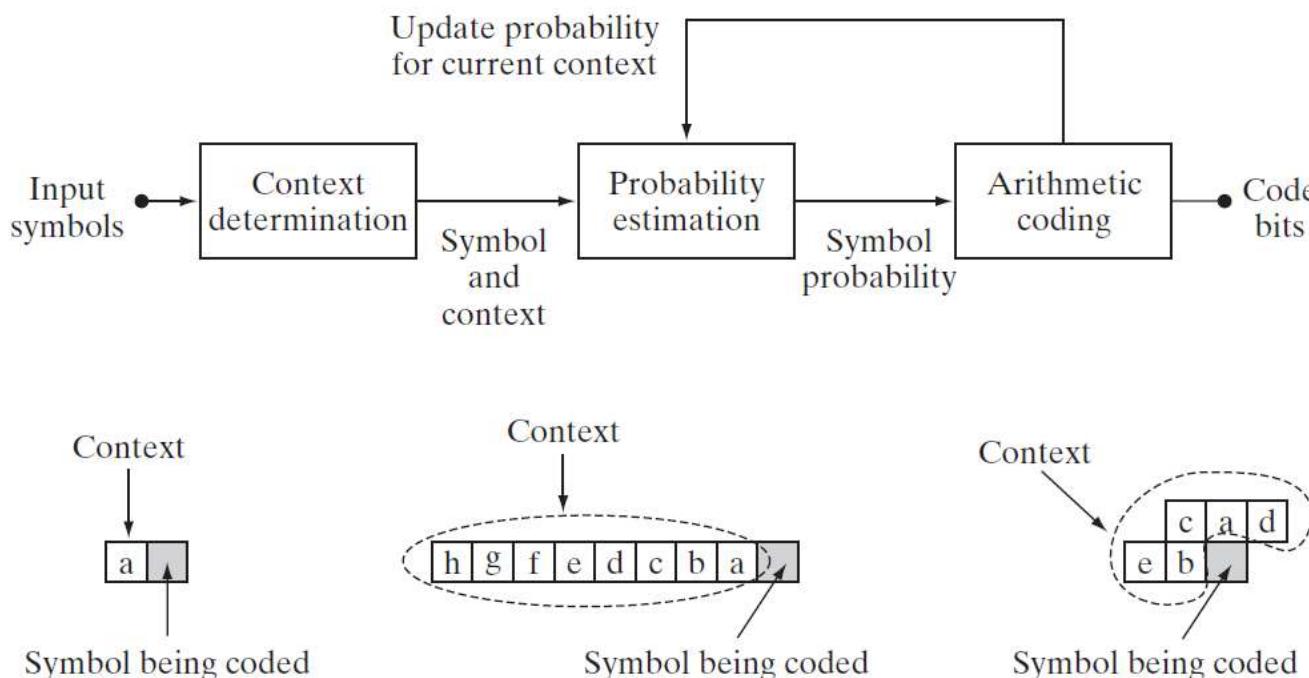
FIGURE 8.12
Arithmetic coding procedure.

2. Some Basic Compression Methods

- Adaptive context dependent probability estimates
 - With accurate input symbol probability models coded, arithmetic coders are near optimal.
 - However, inaccurate probability models can lead to non-optimal results
 - A simple way to improve the accuracy of the probabilities employed is to use an **adaptive, context dependent** probability model.

2. Some Basic Compression Methods

- Adaptive context dependent probability estimates



a
b c d

FIGURE 8.13
 (a) An adaptive, context-based arithmetic coding approach (often used for binary source symbols).
 (b)–(d) Three possible context models.

2. Some Basic Compression Methods

GIF
TIFF
PDF

- Lempel-Ziv-Welch (LZW) Coding
 - A key feature of LZW coding is that it requires no a priori knowledge of the probability of occurrence of the symbols to be encoded.
 - The **coding dictionary** is created while the data are being encoded.
 - Until recently it was protected under a US patent.
 - Has been integrated into a variety of mainstream imaging file formats, including GIF, TIFF and PDF.
 - The PNG format was created to get around LZW licensing requirements.

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding
 - Removes **coding redundancy** and some of the image's **spatial redundancy**.
 - At the onset of the coding process, a codebook or dictionary containing the source symbols to be coded is constructed.
 - For 8-bit monochrome images, the first 256 words of the dictionary are assigned to intensities 0, 1, 2,..., 255.
 - As the encoder sequentially examines image pixels, intensity sequences that are not in the dictionary are placed in algorithmically determined locations.

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

- Example

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

- 9-bit, 512-word dictionary

Dictionary Location	Entry
0	0
1	1
:	:
255	255
256	—
:	:
511	—

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = \emptyset$
2. $C = 39$ ←
3. $P+C = 39$
4. $P+C \in \text{dictionary}:$
 ✓ $P = P+C = 39$

	Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
		39			
	39	39	39	256	39-39
	39	126	39	257	39-126
	126	126	126	258	126-126
	126	39	126	259	126-39
	39	39			
	39-39	126	256	260	39-39-126
	126	126			
	126-126	39	258	261	126-126-39
	39	39			
	39-39	126			
	39-39-126	126	260	262	39-39-126-126
	126	39			
	126-39	39	259	263	126-39-39
	39	126			
	39-126	126	257	264	39-126-126
	126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = 39$
2. $C = 39$
3. $P+C = 39|39$
4. $P+C \notin \text{dictionary}$:

- ✓ Output code for $P = 39$
- ✓ Add $P+C$ to the dictionary
- ✓ $P = C = 39$

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39	39		
39	126	39	256	39-39
126	126	257		39-126
126	39	258		126-126
39	39	259		126-39
39-39	126	260		39-39-126
126	126			
126-126	39	261		126-126-39
39	39			
39-39	126			
39-39-126	126	262		39-39-126-126
126	39			
126-39	39	263		126-39-39
39	126			
39-126	126	264		39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = 39$
2. $C = 126$
3. $P+C = 39|126$
4. $P+C \notin \text{dictionary}$:

- ✓ Output code for $P = 39$
- ✓ Add $P+C$ to the dictionary
- ✓ $P = C = 126$

	Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
		39			
1.	39	39	39	256	39-39
2.	39	126	39	257	39-126
3.	126	126	126	258	126-126
4.	126	39	126	259	126-39
	39	39			
	39-39	126	256	260	39-39-126
	126	126			
	126-126	39	258	261	126-126-39
	39	39			
	39-39	126			
	39-39-126	126	260	262	39-39-126-126
	126	39			
	126-39	39	259	263	126-39-39
	39	126			
	39-126	126	257	264	39-126-126
	126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = 126$
2. $C = 126$
3. $P+C = 126|126$
4. $P+C \notin \text{dictionary}$:

- ✓ Output code for $P = 126$
- ✓ Add $P+C$ to the dictionary
- ✓ $P = C = 126$

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = 126$
2. $C = 39$ ←
3. $P+C = 126|39$
4. $P+C \notin \text{dictionary}$:

- ✓ Output code for $P = 126$
- ✓ Add $P+C$ to the dictionary
- ✓ $P = C = 39$

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	(20)	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = 39$
2. $C = 39$ ←
3. $P+C = 39|39$
4. $P+C \in \text{dictionary}:$

✓ $P = P+C = 39|39$

	Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
		39			
	39	39	39	256	39-39
	39	126	39	257	39-126
	126	126	126	258	126-126
	126	39	126	259	126-39
1.	39	39			
2.	39	39			
3.	39-39	126	256	260	39-39-126
4.	126	126			
	126-126	39	258	261	126-126-39
	39	39			
	39-39	126			
	39-39-126	126	260	262	39-39-126-126
	126	39			
	126-39	39	259	263	126-39-39
	39	126			
	39-126	126	257	264	39-126-126
	126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = 39|39$
2. $C = 126$
3. $P+C = 39|39|126$
4. $P+C \notin \text{dictionary}$:

- ✓ Output code for $P = 39|39$
- ✓ Add $P+C$ to the dictionary
- ✓ $P = C = 126$

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

	Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
		39			
	39	39	39	256	39-39
	39	126	39	257	39-126
	126	126	126	258	126-126
	126	39	126	259	126-39
	39	39			
1. P = 126	39-39	126	256	260	39-39-126
2. C = 126	126	126			
3. P+C = 126 126	126-126	39	258	261	126-126-39
4. P+C ∈ dictionary:	39	39			
✓ P = P+C = 126 126	39-39	126			
	39-39-126	126	260	262	39-39-126-126
	126	39			
	126-39	39	259	263	126-39-39
	39	126			
	39-126	126	257	264	39-126-126
	126	126			

2. Some Basic Compression Methods

- Lempel-Ziv-Welch (LZW) Coding

➤ Example

P: Previous
C: Current

1. $P = 126|126$
2. $C = 39 \leftarrow$
3. $P+C = 126|39|39$
4. $P+C \notin \text{dictionary}$:

- ✓ Output code for $P = 126|126$
- ✓ Add $P+C$ to the dictionary
- ✓ $P = C = 39$

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
	39			
39	39	258	261	126-126-39
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

2. Some Basic Compression Methods

- Run-Length Coding

CCITT
JBIG2
JPEG
M-JPEG
MPEG-1,2,4
BMP

- Images with repeating intensities can often be compressed by representing runs of identical intensities as **run-length pairs**.
- The technique, referred to as **run-length encoding** (RLE) became the standard compression approach in facsimile (FAX) coding.
- Compression is achieved by eliminating a simple form of **spatial redundancy**.
- If there are not enough equal intensity runs, compression is not effective and the **file may expand**.

2. Some Basic Compression Methods

- Run-Length Coding
 - Run-length encoding is particularly **effective** when compressing **binary images**, because adjacent pixels are more likely to be identical.
 - In addition, each image row can be represented by a sequence of lengths only.
 - Algorithm:
 1. Let $s \leftarrow 0$.
 2. While there are bits to encode:
 - a. Read the next n consecutive bits equal to s
 - b. Write n
 - c. $s \leftarrow (s + 1) \bmod 2$

2. Some Basic Compression Methods

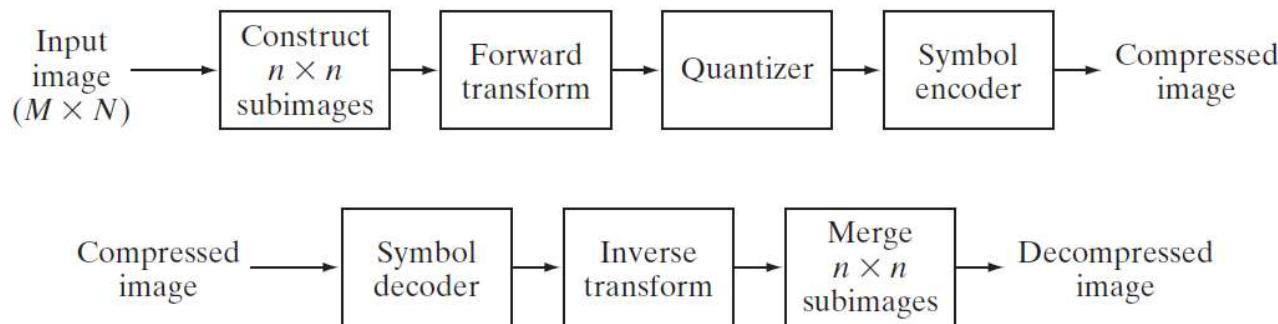
- Block Transform Coding

JPEG
M-JPEG
MPEG-1, 2, 4
H.261, H.262,
H.263, and H.264
DV and HDV
VC-1

- Part of this section was already discussed in “Topic 07 - Image Transforms”
- Here we consider a compression technique that divides an image into **small non-overlapping blocks** and processes the blocks independently.
- A reversible, linear transform is used to map each block into a set of **transform coefficients**, which are then **quantized** and coded.
- For most images, a significant number of the coefficients have **small magnitudes** and can be **coarsely quantized** with little image distortion.

2. Some Basic Compression Methods

- Block Transform Coding



a
b

FIGURE 8.21
A block transform coding system:
(a) encoder;
(b) decoder.

2. Some Basic Compression Methods

- Block Transform Coding
 - Bit allocation
 - ✓ The **reconstruction error** is a **function of** the relative importance of the **coefficients** that are **discarded**.
 - ✓ In most transform coding systems, the retained coefficients are selected on the basis of
 - ✓ maximum variance (**zonal coding**); or
 - ✓ maximum magnitude (**threshold coding**).
 - ✓ The overall process of **truncating**, **quantizing**, and **coding** the coefficients of a transformed subimage is commonly called **bit allocation**.

2. Some Basic Compression Methods

- Block Transform Coding
 - Bit allocation
 - ✓ **Threshold coding:** this result was obtained by keeping the eight largest DCT coefficients.



2. Some Basic Compression Methods

- Block Transform Coding
 - Bit allocation
 - ✓ **Zonal coding:** each coefficient was considered a random variable whose distribution could be computed over the ensemble of all subimages.



2. Some Basic Compression Methods

- Block Transform Coding
 - Bit allocation
 - ✓ **Zonal coding:** the 8 distributions of largest variance were located and used to determine the coefficients that were retained.



2. Some Basic Compression Methods

- Block Transform Coding
 - Bit allocation
 - ✓ In both cases **87.5%** of the DCT coefficients of each subimage were discarded



a b
c d

FIGURE 8.28
Approximations of Fig. 8.9(a) using 12.5% of the 8×8 DCT coefficients:
(a)–(b) threshold coding results;
(c)–(d) zonal coding results. The difference images are scaled by 4.

2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation
 - ✓ The underlying concept is that the transform coefficients of **largest magnitude** make the **most significant contribution**.
 - ✓ There are some ways to threshold a transformed subimage.
 - ✓ We will assume that the threshold can be varied as a **function of the location** of each coefficient within the subimage.
 - In this case, thresholding and quantization can be combined.

2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation

$$\hat{T}(u, v) = \text{round} \left[\frac{T(u, v)}{Z(u, v)} \right]$$

where $\hat{T}(u, v)$ is a thresholded and quantized approximation of $T(u, v)$ and Z is an element of the transform normalization array

$$\mathbf{Z} = \begin{bmatrix} Z(0, 0) & Z(0, 1) & \dots & Z(0, n - 1) \\ Z(1, 0) & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ Z(n - 1, 0) & Z(n - 1, 1) & \dots & Z(n - 1, n - 1) \end{bmatrix}$$

2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation
 - ✓ The denormalized array, denoted $\dot{T}(u, v)$ is an approximation of $\hat{T}(u, v)$.

$$\dot{T}(u, v) = \hat{T}(u, v)Z(u, v)$$

- ✓ The inverse transform of $\dot{T}(u, v)$ yields the decompressed subimage approximation.

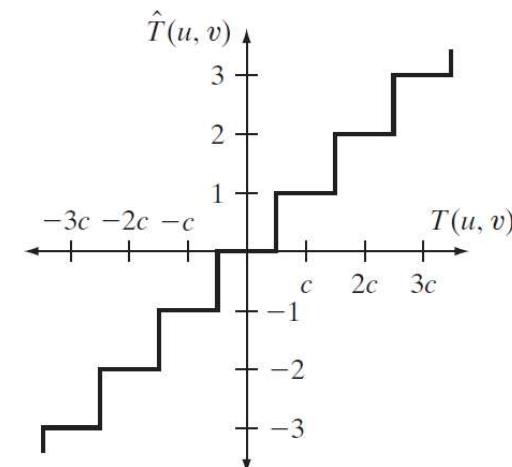
2. Some Basic Compression Methods

- Block Transform Coding

- Threshold coding implementation

- ✓ The graphics bellow depicts the case in which $Z(u, v)$ is assigned a particular value c .
 - ✓ Note that $\hat{T}(u, v)$ assumes an integer value k if and only if

$$kc - \frac{c}{2} \leq T(u, v) < kc + \frac{c}{2}$$



2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation
 - ✓ If $Z(u, v) > 2T(u, v)$, then $\hat{T}(u, v) = 0$ and the transform coefficient is discarded.
 - ✓ When k is represented with a variable-length code that increases in length as the magnitude of increases, the number of bits used to represent $T(u, v)$ is controlled by the value of c .
 - ✓ Thus the elements of Z can be scaled to achieve a variety of compression levels.

2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation
 - ✓ The following array has been used extensively in the JPEG standardization.
 - ✓ Each coefficient is quantized according to its psychovisual importance.

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

2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation



FIGURE 8.31 Approximations of Fig. 8.9(a) using the DCT and normalization array of Fig. 8.30(b): (a) \mathbf{Z} , (b) $2\mathbf{Z}$, (c) $4\mathbf{Z}$, (d) $8\mathbf{Z}$, (e) $16\mathbf{Z}$, and (f) $32\mathbf{Z}$.

2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation



FIGURE 8.31 Approximations of Fig. 8.9(a) using the DCT and normalization array of Fig. 8.30(b): (a) \mathbf{Z} , (b) $2\mathbf{Z}$, (c) $4\mathbf{Z}$, (d) $8\mathbf{Z}$, (e) $16\mathbf{Z}$, and (f) $32\mathbf{Z}$.

2. Some Basic Compression Methods

- Block Transform Coding
 - Threshold coding implementation
 - ✓ Quantization matrix: Z, 2Z, 4Z, 8Z, 16Z, 32Z
 - ✓ Compression: 12, 19, 30, 49, 85, 182:1
 - ✓ rms error: 3.83, 4.93, 6.62, 9.35, 13.94, 22.46

2. Some Basic Compression Methods

- Block Transform Coding
 - One of the most popular and comprehensive continuous tone, still frame compression standards is the **JPEG standard**.
 - It defines three different coding systems.
 - In the baseline system, the input and output data precision is limited to 8 bits, whereas the quantized DCT values are restricted to 11 bits.
 - The compression itself is performed in three sequential steps: DCT computation, quantization, and variable-length code assignment.

2. Some Basic Compression Methods

- Block Transform Coding
 - The image is first **subdivided into blocks** of size 8x8, which are processed left to right, top to bottom.
 - As each block is encountered, its 64 pixels are **level-shifted** by subtracting the quantity 2^{k-1} , where 2^k is the maximum number of intensity levels.
 - **The 2-D discrete cosine transform** of the block is then computed, **quantized** and **reordered** to form a 1-D sequence of quantized coefficients.
 - The JPEG coding procedure is designed to take advantage of the **long runs of zeros** that normally result from the reordering

2. Some Basic Compression Methods

- Block Transform Coding
 - The nonzero AC coefficients are coded using a **variable-length** code that defines the coefficient values and number of preceding zeros.
 - The DC coefficient is **difference coded** relative to the DC coefficient of the previous subimage.
 - **Default coding tables** and quantization arrays are provided for both color and monochrome processing.
 - The user is **free to construct** custom tables and/or arrays.

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Original subimage

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	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Pixels values intensity-shifted by -128

-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

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ DCT of intensity-shifted array

$$\hat{T}(u, v) = \text{round} \left[\frac{T(u, v)}{Z(u, v)} \right]$$

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

-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

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Normalization

$$\hat{T}(u, v) = \text{round} \left[\frac{T(u, v)}{Z(u, v)} \right]$$

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

-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

2. Some Basic Compression Methods

- Block Transform Coding

➤ Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

✓ The coefficients are reordered using the zig-zag scan

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

-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

[-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]

2. Some Basic Compression Methods

- Block Transform Coding

➤ Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

✓ Modified Huffman code tables

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

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

TABLE A.4 JPEG default DC code (luminance).

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	1010 (= EOB)	4			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	11111111000000	17
0/3	100	6	8/3	111111110110111	19
0/4	1011	8	8/4	111111110111000	20
0/5	11010	10	8/5	111111110111001	21
0/6	111000	12	8/6	111111110111010	22
0/7	1111000	14	8/7	111111110111011	23
0/8	1111110110	18	8/8	111111110111100	24
0/9	111111110000010	25	8/9	111111110111101	25
0/A	111111111000011	26	8/A	111111110111110	26

TABLE A.5 JPEG default AC code (luminance).

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Modified Huffman code tables

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



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

- ✓ The resulting compression ratio is 512/92, or about 5.6:1

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Modified Huffman code tables
 - DC coefficients: compute difference between the current DC coefficient dc_i , and that of the previously dc_{i-1} encoded subimage.
 - If $[d_i - d_{i-1}] = [-26 - (-17)]$ or **-9** then the DC difference category is 4.
 - The proper base code for a category 4 is 101 (a 3-bit code).

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Modified Huffman code tables
 - The total length of a completely encoded category 4 DC coefficient is 7 bits.
 - The remaining $k = 4$ bits must be generated from the LSBs of the difference value.
 - If the difference is positive, take the k LSBs bits; else take the negative difference minus 1.

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Modified Huffman code tables
 - $-9_{10} = \mathbf{10111}_2$
 - $-26 \text{ (JPEG)} \rightarrow (101: \mathbf{0111} - 1)_2 \rightarrow 101:0110_2$

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Modified Huffman code tables
 - The nonzero **AC coefficients** of the reordered array are **coded similarly**.
 - The **principal difference** is that each default AC Huffman code word depends also on the number of **zero-valued coefficients** preceding the nonzero coefficient to be coded.

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Modified Huffman code tables
 - Consider the first AC coefficient -3.
 - It belongs to category 2 and is preceded by no zero-valued coefficients.
 - The base code for a category 2/run 0 is 01.
 - The last 2 bits are generated by the same process used to arrive at the LSBs of the DC difference code.

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Modified Huffman code tables
 - $-3_{10} \rightarrow \mathbf{101}_2$
 - -3_{10} (JPEG) $\rightarrow 01: (\mathbf{01}-1)_2 = (01:00)_2$

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Decoding: denormalization

$$\dot{T}(0,0) = \hat{T}(0,0)Z(0,0) = (-26)(16) = -416$$

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

-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

2. Some Basic Compression Methods

- Block Transform Coding

- Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Decoding: denormalization

$$\dot{T}(0, 0) = \hat{T}(0, 0)Z(0, 0) = (-26)(16) = -416$$

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

-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

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Decoding: 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

2. Some Basic Compression Methods

- Block Transform Coding
 - Example: compression and reconstruction of the following subimage with the JPEG baseline standard.
 - ✓ Decoding: level shifting

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

2. Some Basic Compression Methods

- Block Transform Coding

➤ Example: compression and reconstruction of the following subimage with the JPEG baseline standard.

- ✓ Decoding

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	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

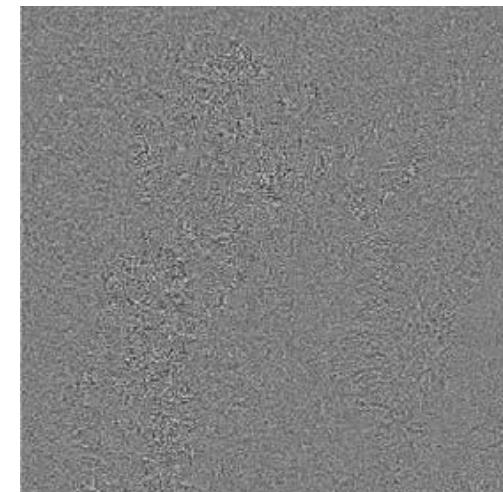
Original

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

Reconstructed

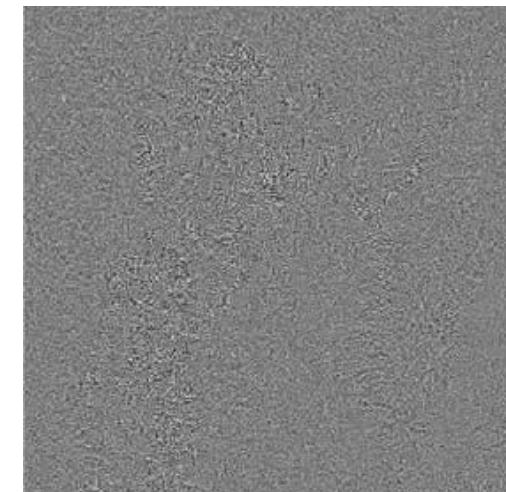
3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
 - This result was obtained by **dividing** the original image into subimages of **size 8 x 8** using the DCT, **truncating** 50% of the resulting coefficients, and **taking the inverse transform** of the truncated coefficient arrays.



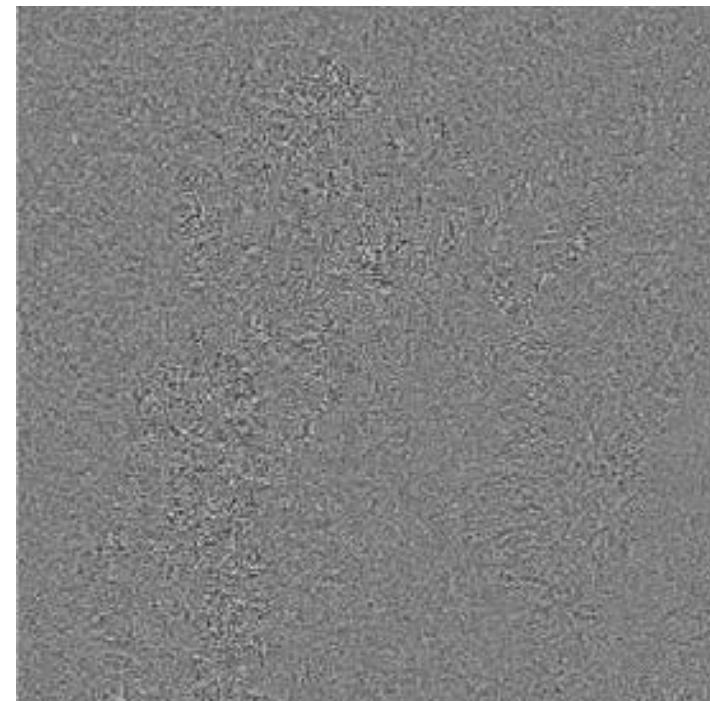
3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
 - The 32 retained coefficients were selected on the basis of maximum magnitude. In this example, the *rms* error was 1.13.



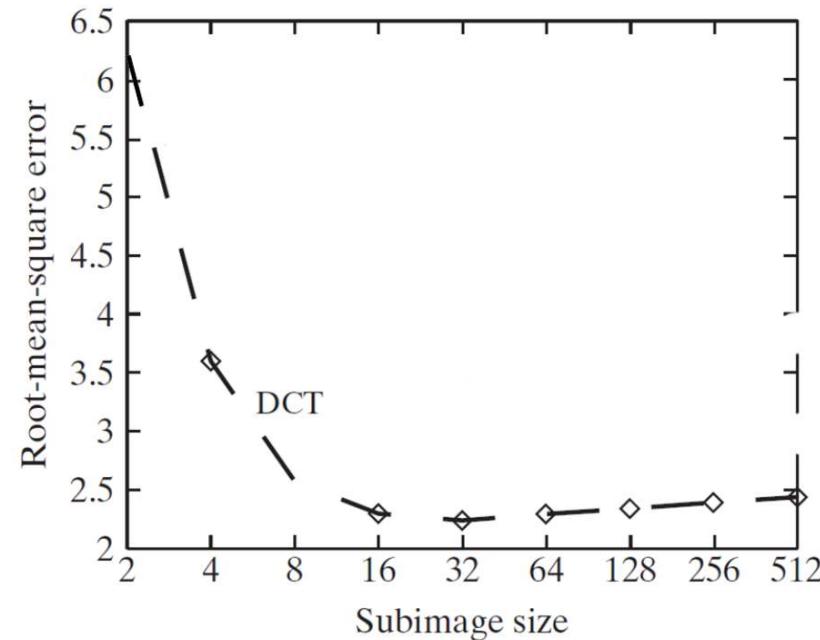
3. Discrete Cosine Transform

- MATLAB: s109Subimage8Size.m



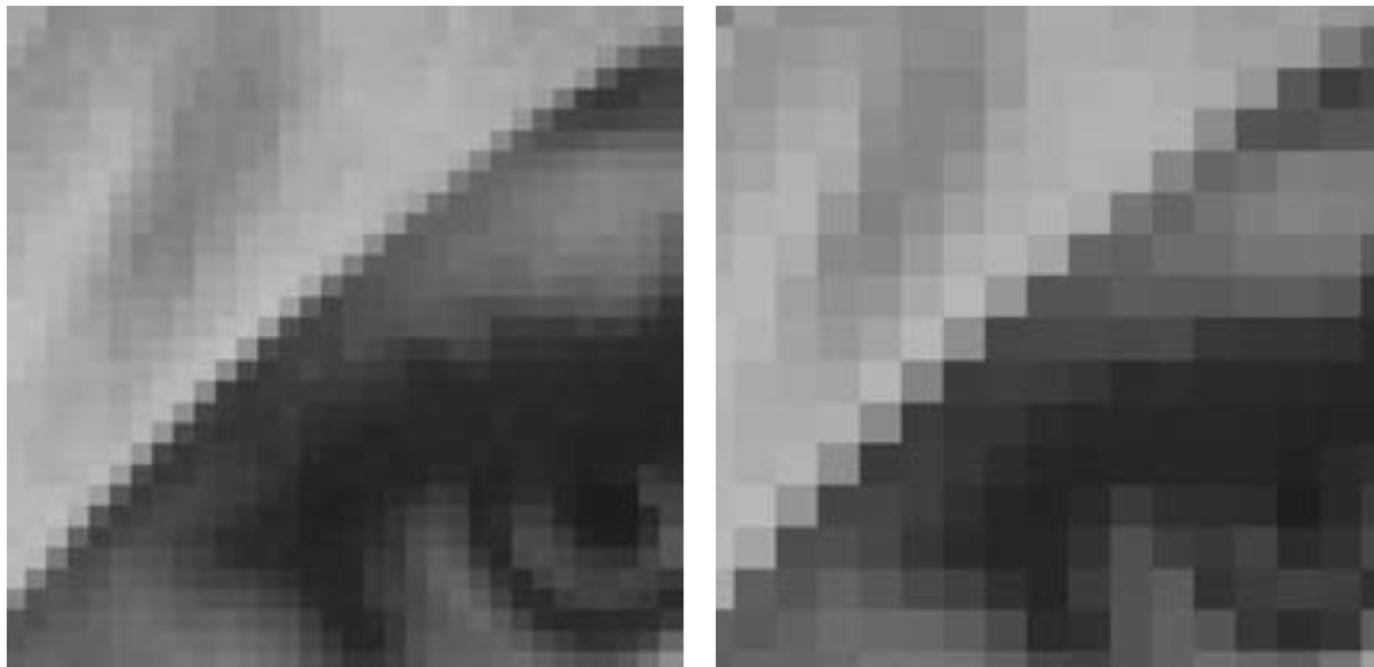
3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
 - Truncating 75% of coefficients



3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.

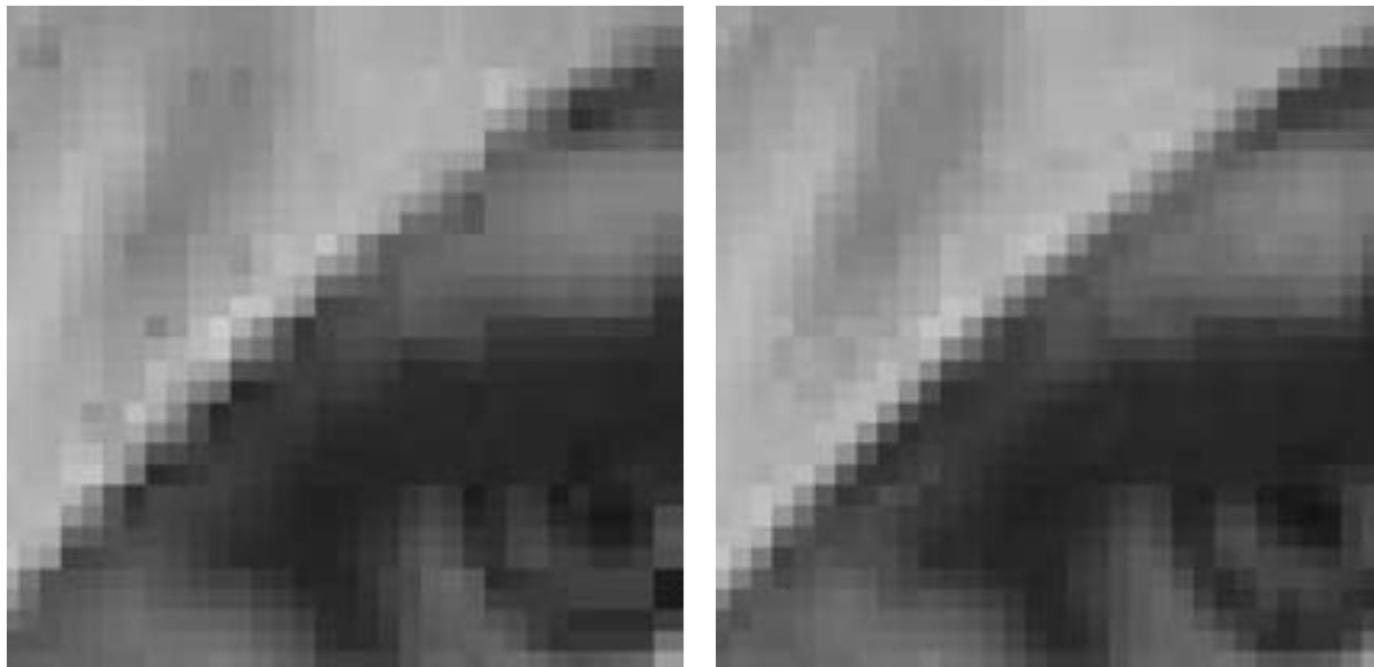


a b c d

FIGURE 8.27 Approximations of Fig. 8.27(a) using 25% of the DCT coefficients and (b) 2×2 subimages, (c) 4×4 subimages, and (d) 8×8 subimages. The original image in (a) is a zoomed section of Fig. 8.9(a).

3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.

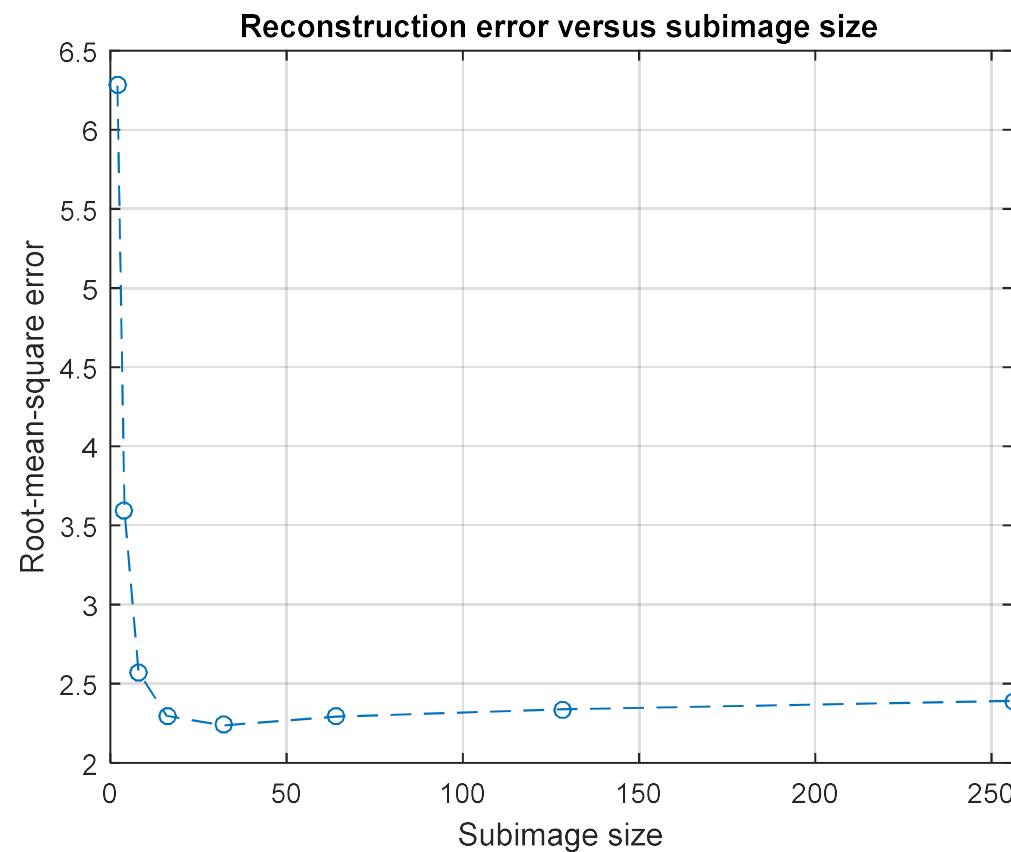


a b c d

FIGURE 8.27 Approximations of Fig. 8.27(a) using 25% of the DCT coefficients and (b) 2×2 subimages, (c) 4×4 subimages, and (d) 8×8 subimages. The original image in (a) is a zoomed section of Fig. 8.9(a).

3. Discrete Cosine Transform

- MATLAB: s113SubimageNSize.m



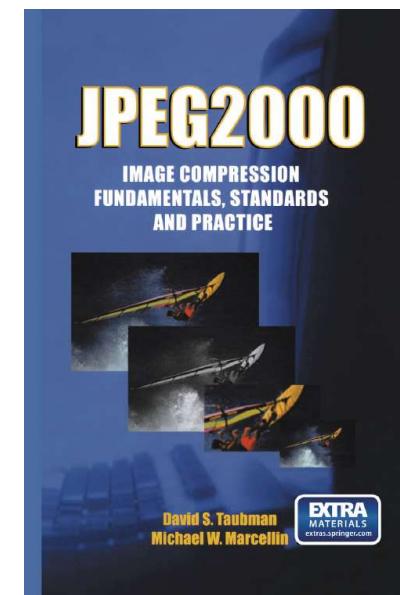
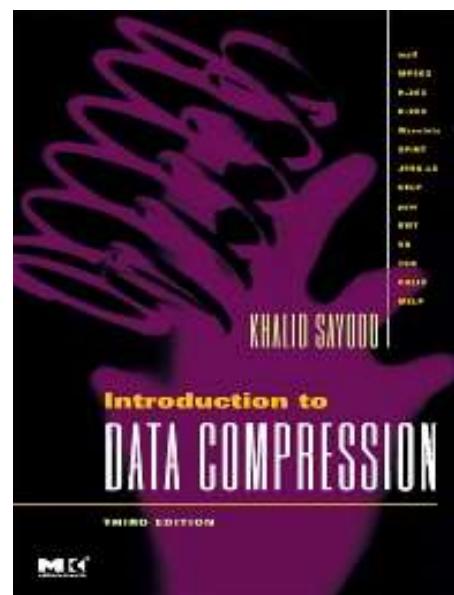
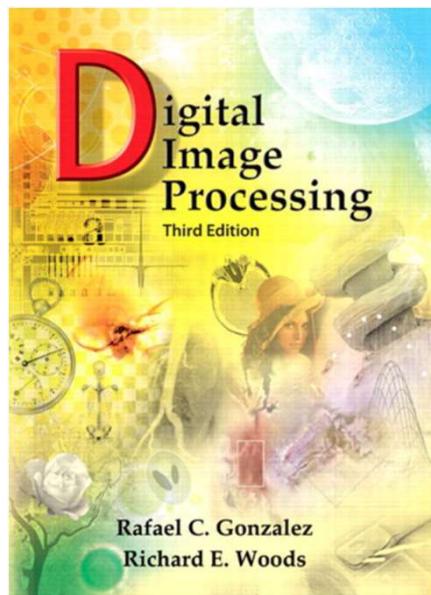
3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
 - The **larger** the **size** the **more decorrelation** the transform coding can achieve.
 - However, the **correlation** between pixels becomes **insignificant** when the distance between pixels becomes large, e.g., **20 pixels**.
 - Images are subdivided so that the **correlation between adjacent** subimages is **reduced** to some **acceptable level**.
 - The **computation** of the subimage **transforms** is **simplified** if the block size is an integer power of 2.

3. Discrete Cosine Transform

- Impact of subimage size on reconstruction error.
 - In adaptive transform coding, a large block **cannot adapt to local statistics** well.
 - **Large size** implies a possibly severe effect of **transmission error** in reconstructed images.
 - As will be shown in video coding, transform coding is used together with motion compensated coding, where blocks of 4, 8 and 16 are most often.
 - In general, both the level of compression and computational complexity increase as the subimage size increases.
 - The most popular subimage sizes are 8x8 and 16x16.

4. Further Reading



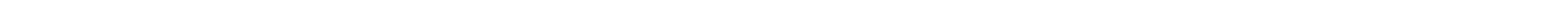
Introduction to Image Processing

Prof. Alexandre Zaghetto
alexandre.zaghetto@mcgill.ca

McGill University
Department of Electrical and Computer Engineering

Topic 09

Video Coding



1. Introduction

- A change of scene

➤ 2000

- ✓ Most viewers receive analogue television via terrestrial, cable or satellite transmission.
- ✓ VHS video tapes are the principal medium for recording and playing TV programs, movies, etc.
- ✓ Cell phones are cell phones, i.e. a mobile handset can only be used to make calls or send SMS messages.
- ✓ Internet connections are slow, primarily over telephone modems for home users.

1. Introduction

- A change of scene

➤ 2000

- ✓ Internet connections are slow, primarily over telephone modems for home users.
- ✓ Web pages are web pages, with static text, graphics and photos and not much else.
- ✓ Video calling requires dedicated video-conferencing terminals and expensive leased lines.
- ✓ Video calling over the internet is possible but slow, unreliable and difficult to set up.

1. Introduction

- A change of scene
 - 2000
 - ✓ Video calling over the internet is possible but slow, unreliable and difficult to set up.
 - ✓ Consumer video cameras, camcorders, use tape media, principally analogue tape.
 - ✓ Home-made videos generally stay within the home.

1. Introduction

- A change of scene

➤ 2010

- ✓ Most viewers receive digital television via terrestrial, cable, satellite or internet
- ✓ Greater choice of channels, electronic programme guides and high definition services.
- ✓ Analogue TV has been switched off in many countries.
- ✓ Many TV programmes can be watched via the internet.

1. Introduction

- A change of scene

➤ 2010

- ✓ DVDs are the principal medium for playing pre-recorded movies and TV programs.
- ✓ Movie downloading, hard-disk recording and playback.
- ✓ Variety of digital media formats.
- ✓ High definition DVDs, Blu-Ray Disks, are increasing in popularity.

1. Introduction

- A change of scene

➤ 2010

- ✓ Cell phones function also as cameras.
- ✓ Internet access speeds continue to get faster, enabling widespread use of video-based web applications.
- ✓ Among other things web pages are movie players with content that changes dynamically.
- ✓ Video calling over the internet is commonplace.
- ✓ Consumer video cameras use hard disk or flash memory card media.

1. Introduction

- A change of scene

➤ 2010

- ✓ Editing, uploading and internet sharing of home videos is widespread.
- ✓ A whole range of illegal activities has been born
 - DVD piracy, movie sharing via the internet etc.
- ✓ Video footage of breaking news is more likely to come from a cell phone than a TV camera.

1. Introduction

- A change of scene
 - We will focus on one technical aspect that is key to the widespread adoption of digital video technology – **video compression**.
 - Digital video data tends to take up a **large amount of storage** or transmission capacity.
 - Video compression or video coding is the process of **reducing the amount of data** required to represent a digital video signal.
 - It is essential for any application in which storage capacity or transmission bandwidth is constrained.

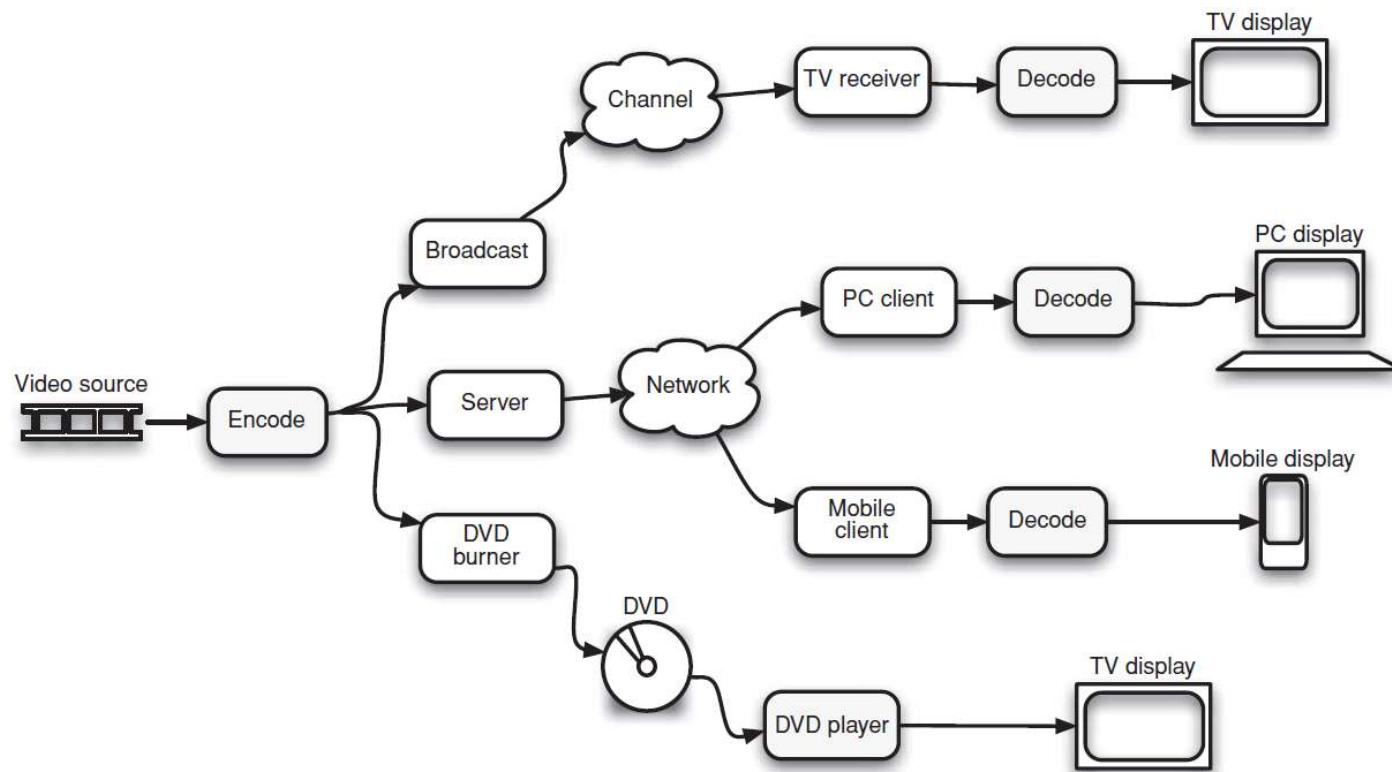
1. Introduction

- A change of scene
 - Almost all consumer applications for digital video fall into this category:
 - ✓ One-way scenario
 - Digital television broadcasting
 - Internet video streaming
 - Mobile video streaming
 - DVD video
 - ✓ Two-way scenario
 - Video calling

1. Introduction

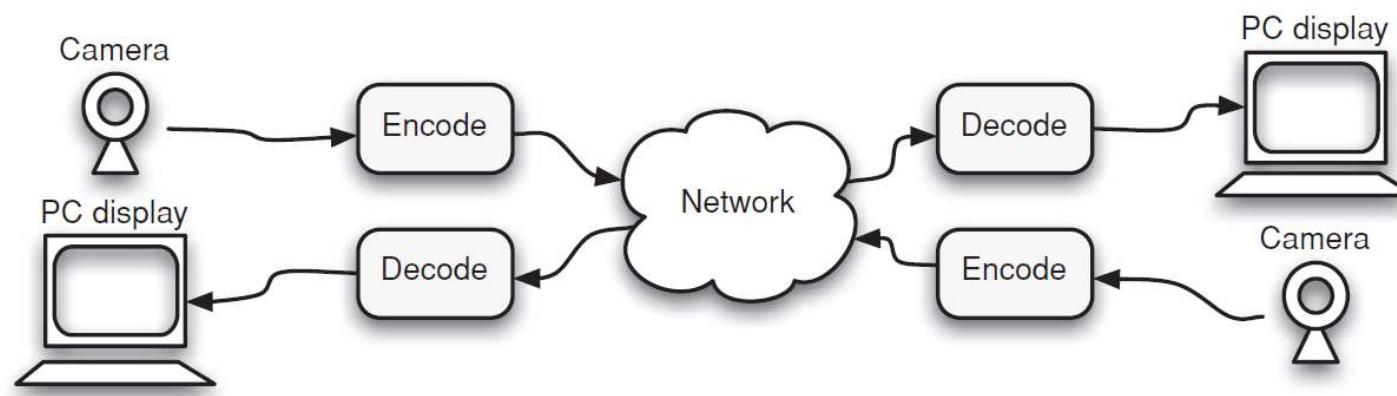
- A change of scene

- Video coding scenarios, one-way



1. Introduction

- A change of scene
 - Video coding scenarios, one-way



1. Introduction

- Driving the change
 - The consumer applications discussed above represent very **large markets**.
 - A TV company that can pack a **larger number of high-quality** channels into the available bandwidth has a market edge over its competitors.
 - **Better video codec** results in a **better product** and therefore a more **competitive product**.
 - This drive to improve video compression technology has led to **significant investment** in video coding **research** and **development** over the last 15-20 years.

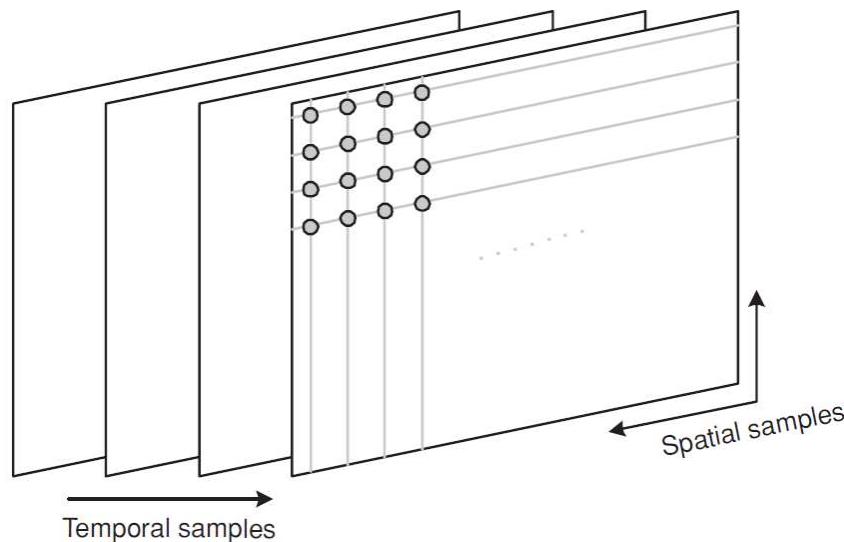
2. Video Formats and Quality

- Natural video scenes
 - Characteristics of a typical natural video scene that are relevant for video processing and compression include:
 - ✓ Spatial characteristics such as;
 - texture variation within scene;
 - number and shape of objects; and
 - colour.
 - ✓ Temporal characteristics such as
 - object motion;
 - changes in illumination; and
 - movement of the camera or viewpoint.

2. Video Formats and Quality

- Capture

- A natural visual scene is **spatially** and **temporally** continuous.
- Representing a visual scene in digital form involves **sampling** the real scene spatially and temporally.



2. Video Formats and Quality

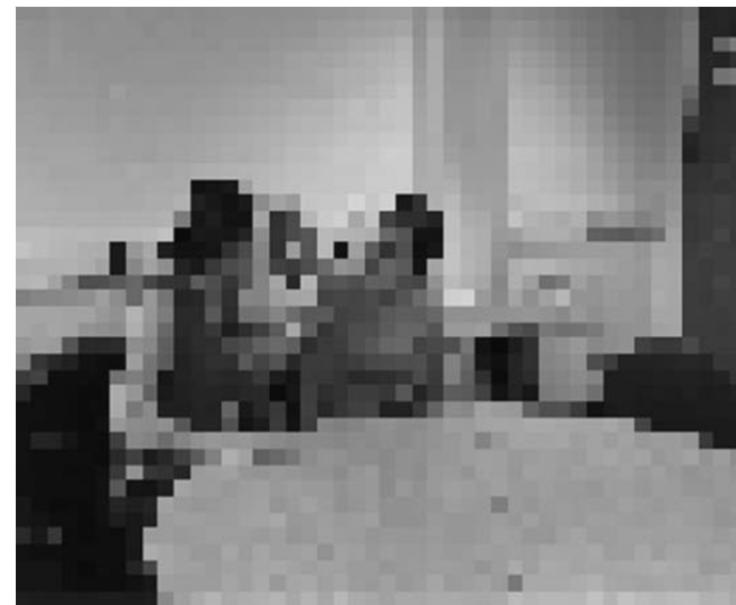
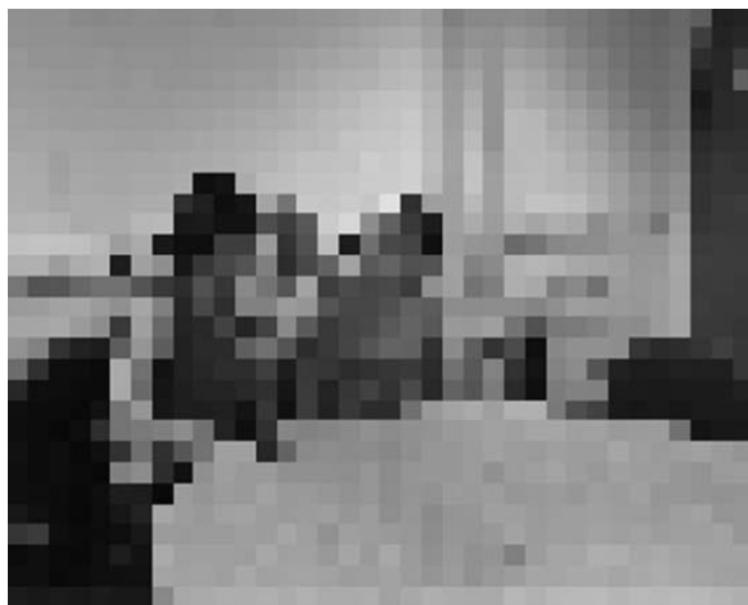
- Capture
 - To obtain a 2-D sampled image, a camera focuses a 2-D projection of the video scene onto a sensor, such as an array of Charge Coupled Devices (CCDs).
 - In the case of **colour image** capture, each colour component is separately **filtered** and projected onto a CCD array

2. Video Formats and Quality

- Spatial sampling
 - The output of a CCD array is an analogue video signal, a varying electrical signal that represents a video image.
 - Sampling the image at a specific time produces a sampled **frame** that has defined values at a set of discrete positions.
 - The number of sampled positions influences the visual quality of the image.

2. Video Formats and Quality

- Spatial sampling



2. Video Formats and Quality

- Temporal sampling
 - A moving video image is formed by taking a rectangular **snapshot** of the signal at periodic time intervals.
 - Playing back the series of snapshots or frames produces the appearance of **motion**.
 - **Higher temporal sampling rate** or frame rate gives apparently **smoother motion** in the video scene
 - ✓ 10 frames/s ➔ very low bit-rate video
 - ✓ 10–20 frames/s ➔ low bit-rate video
 - ✓ 25 or 30 frames/s ➔ Standard Definition TV

2. Video Formats and Quality

- Temporal sampling
 - A moving video image is formed by taking a rectangular **snapshot** of the signal at periodic time intervals.
 - Playing back the series of snapshots or frames produces the appearance of **motion**.
 - **Higher temporal sampling rate** or frame rate gives apparently **smoother motion** in the video scene
 - ✓ 10 frames/s ➔ very low bit-rate video
 - ✓ 10–20 frames/s ➔ low bit-rate video
 - ✓ 25 or 30 frames/s ➔ Standard Definition TV
 - ✓ 50 or 60 frames/s ➔ very smooth motion

2. Video Formats and Quality

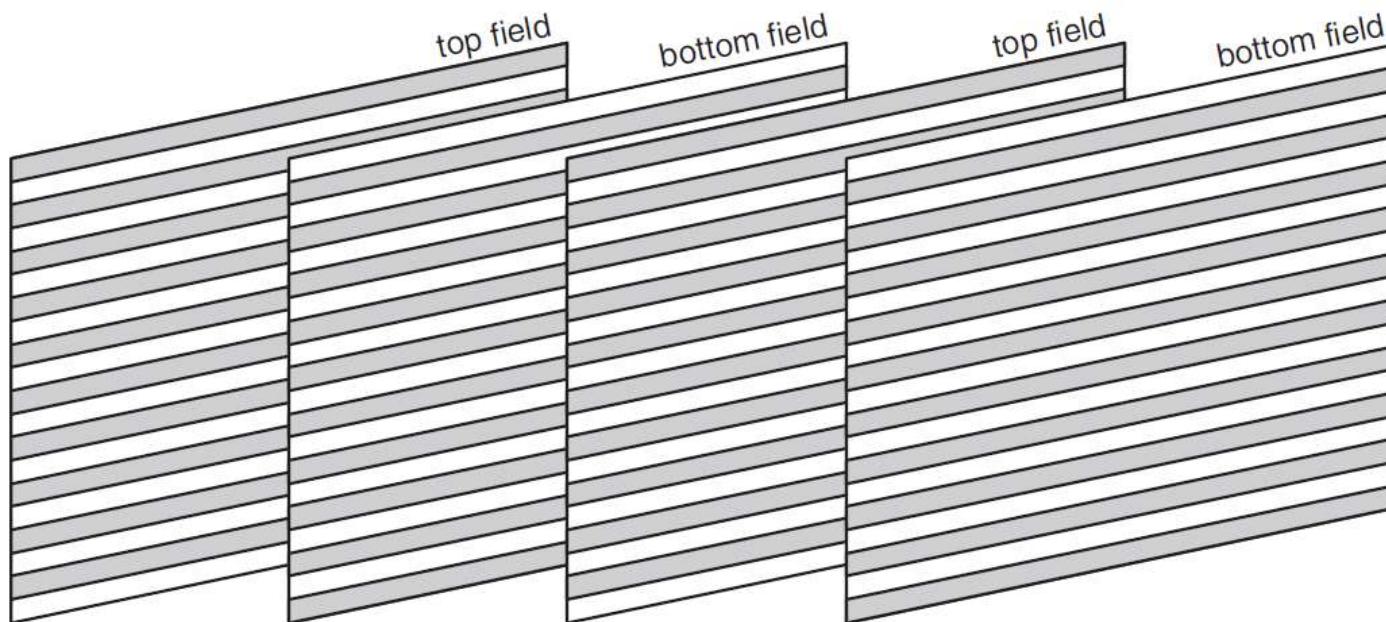
- Frames and fields

- A video signal may be sampled as a series of complete frames (**progressive** sampling) or as a sequence of interlaced fields (**interlaced** sampling).
- In an **interlaced** video sequence, **half** of the data in a frame (**one field**) is typically sampled **at each** temporal sampling **interval**.
- A field may consist of either the odd-numbered or even-numbered lines within a complete video frame.
- An interlaced video sequence typically contains a series of fields, each representing half of the information in a complete video frame

2. Video Formats and Quality

- Frames and fields

- An interlaced video sequence typically contains a series of fields, each representing half of the information in a complete video frame



2. Video Formats and Quality

- Frames and fields

- An interlaced video sequence typically contains a series of fields, each representing half of the information in a complete video frame



2. Video Formats and Quality

- Frames and fields

- The advantage of this sampling method is that it is possible to send twice as many fields per second as the number of frames in an equivalent progressive sequence.
- For example, a PAL video sequence consists of 50 fields/s.
 - When played back, motion **appears smoother** than in an equivalent **progressive** video sequence containing **25 frames** per second.

2. Video Formats and Quality

- Color spaces

- RGB

- ✓ A pixel is represented with three numbers that indicate the amount of Red, Green and Blue which define its color.



2. Video Formats and Quality

- Color spaces

- YCrCb

- ✓ The human visual system (HVS) is less sensitive to colour than to luminance.
 - ✓ In the RGB colour space the three colours are equally important and so are usually all stored at the same resolution.
 - ✓ It is possible to represent a colour image more efficiently by separating the luminance from the colour information and representing **luma** with a higher resolution than the chrominance or **chroma**.

2. Video Formats and Quality

- Color spaces

➤ YCrCb

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = 0.564(B - Y)$$

$$Cr = 0.713(R - Y)$$

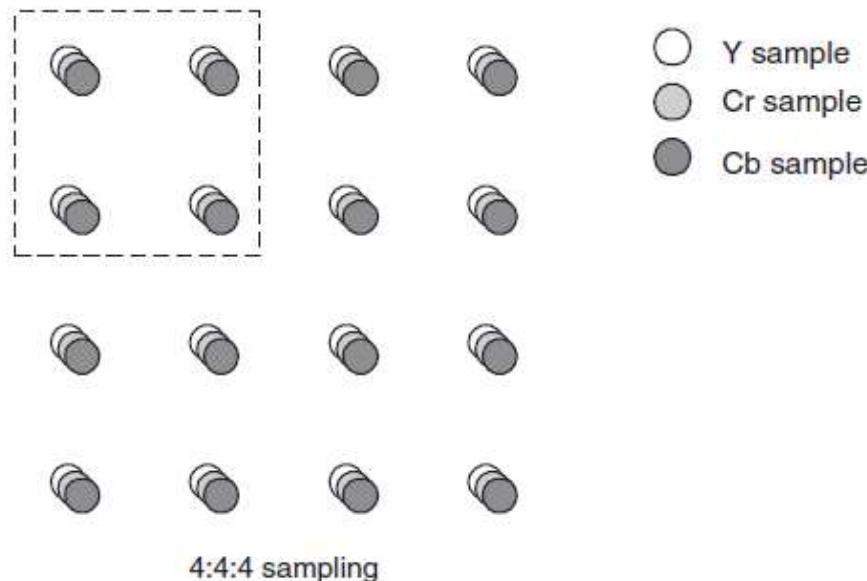
$$R = Y + 1.402Cr$$

$$G = Y - 0.344Cb - 0.714Cr$$

$$B = Y + 1.772Cb$$

2. Video Formats and Quality

- Color spaces
 - Examples of YCrCb sampling formats: 4:4:4
 - ✓ The three components have the same resolution.

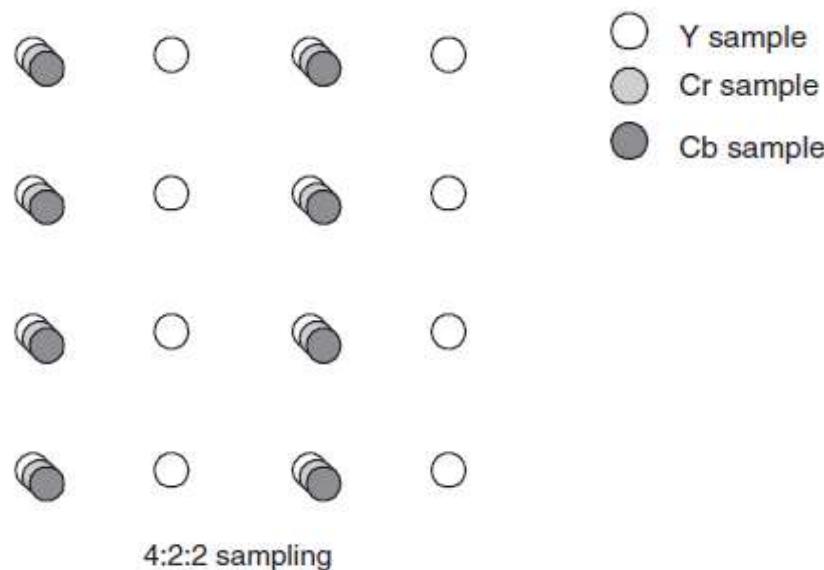


2. Video Formats and Quality

- Color spaces

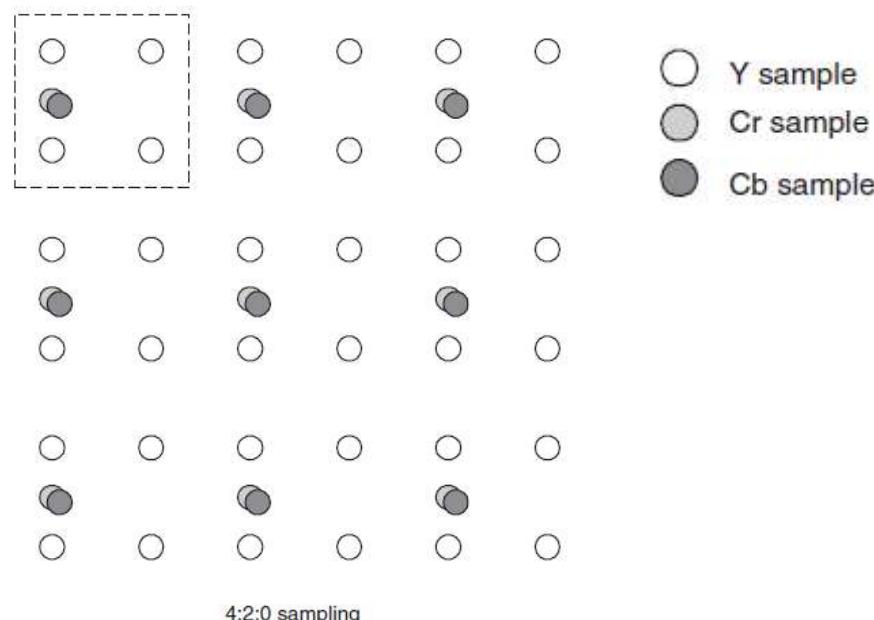
- Examples of YCrCb sampling formats: 4:2:2

- ✓ The chrominance components have the same vertical resolution as the luma but half the horizontal resolution.



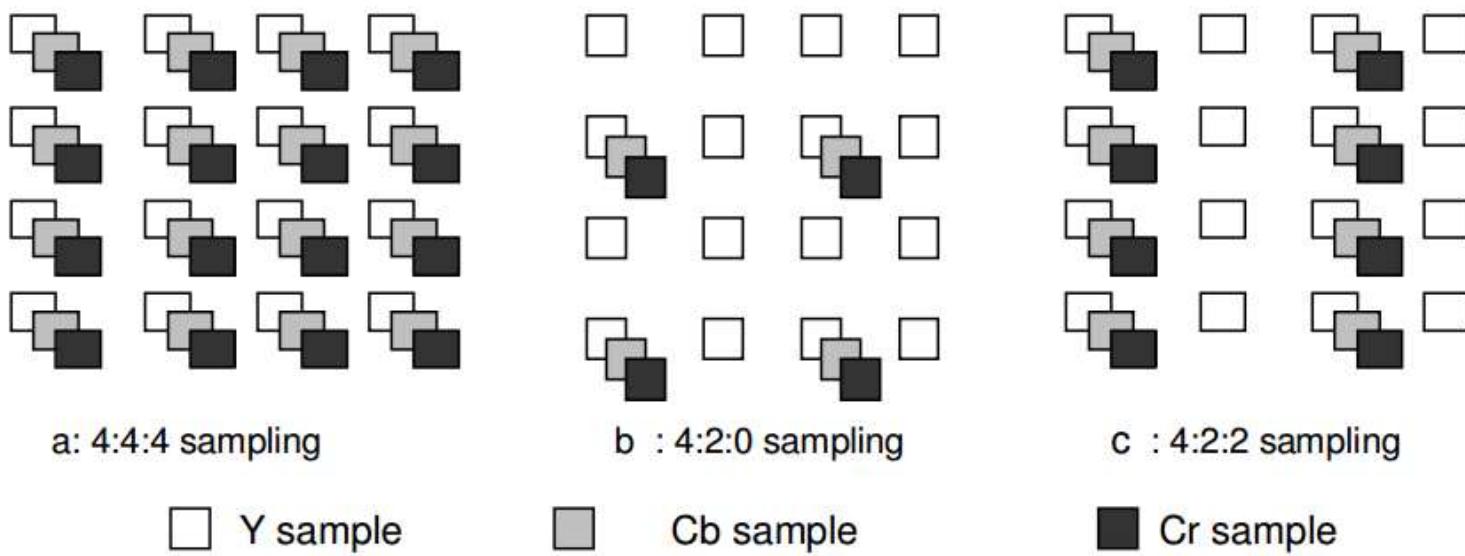
2. Video Formats and Quality

- Color spaces
 - Examples of YCrCb sampling formats: 4:2:0
 - C_b and C_r each have half the horizontal and vertical resolution.



2. Video Formats and Quality

- Color spaces
 - Summary: 4:4:4 | 4:2:2 |4:2:0

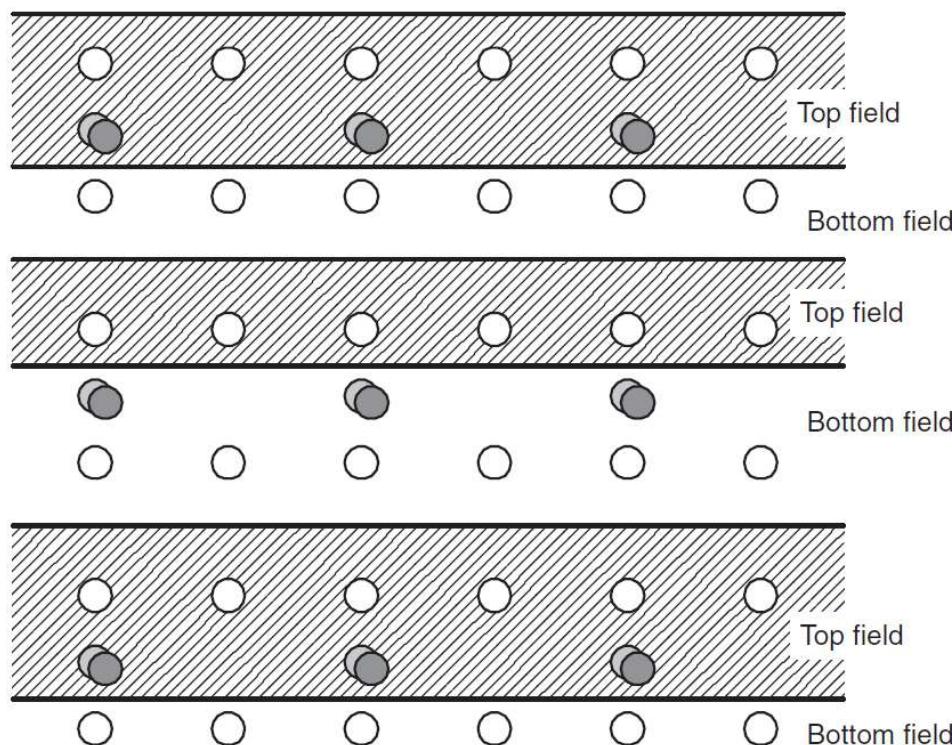


<http://www.diva-portal.org/smash/get/diva2:831349/FULLTEXT01.pdf>

2. Video Formats and Quality

- Formats

- Allocation of 4:2:0 samples to top and bottom fields.



2. Video Formats and Quality

- Formats

- Intermediate formats

- The Common Intermediate Format (CIF) is the basis for a popular set of formats:

Format	Luminance resolution (horiz. × vert.)	Bits per frame (4:2:0, 8 bits per sample)
Sub-QCIF	128 × 96	147456
Quarter CIF (QCIF)	176 × 144	304128
CIF	352 × 288	1216512
4CIF	704 × 576	4866048

2. Video Formats and Quality

- Formats
 - Intermediate formats



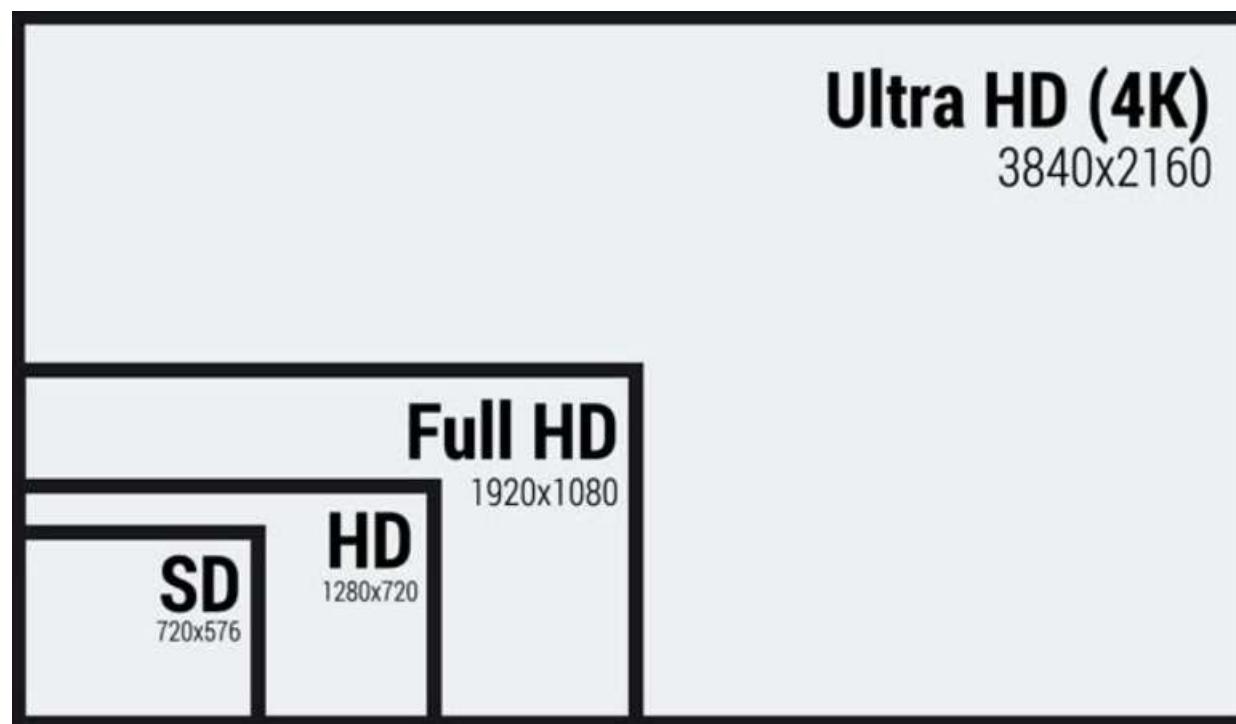
2. Video Formats and Quality

- Formats
 - Standard and High Definition

Format	Progressive or Interlaced	Horizontal pixels	Vertical pixels	Frames or fields per second
720p	Progressive	1280	720	25 frames
1080i	Interlaced	1920	1080	50 fields
1080p	Progressive	1920	1080	25 frames

2. Video Formats and Quality

- Formats
 - Standard and High Definition



2. Video Formats and Quality

- Quality
 - Subjective quality measurement
 - ✓ The perception of visual quality is influenced by:
 - **spatial fidelity**, i.e. how clearly parts of the scene can be seen, whether there is any obvious distortion; and
 - **temporal fidelity**, i.e. whether motion appears natural and 'smooth'.

2. Video Formats and Quality

- Quality
 - Subjective quality measurement
 - ✓ However, a viewer's opinion of 'quality' is also affected by other factors such as:
 - the viewing environment;
 - the observer's state of mind; and
 - the extent to which the observer interacts with the visual scene.

2. Video Formats and Quality

- Quality

- Subjective quality measurement

- ✓ A widely-used quality test procedure is the Double Stimulus Continuous Quality Scale (DSCQS) method.
 - ✓ An assessor is presented with a pair of images or short video sequences A and B, one after the other, and is asked to give A and B a 'quality score' by marking on a continuous line with five intervals ranging from 'Excellent' to 'Bad'.
 - ✓ Within each pair of sequences, one is an unimpaired 'reference' sequence and the other is the same sequence, modified by a system or process under test.

2. Video Formats and Quality

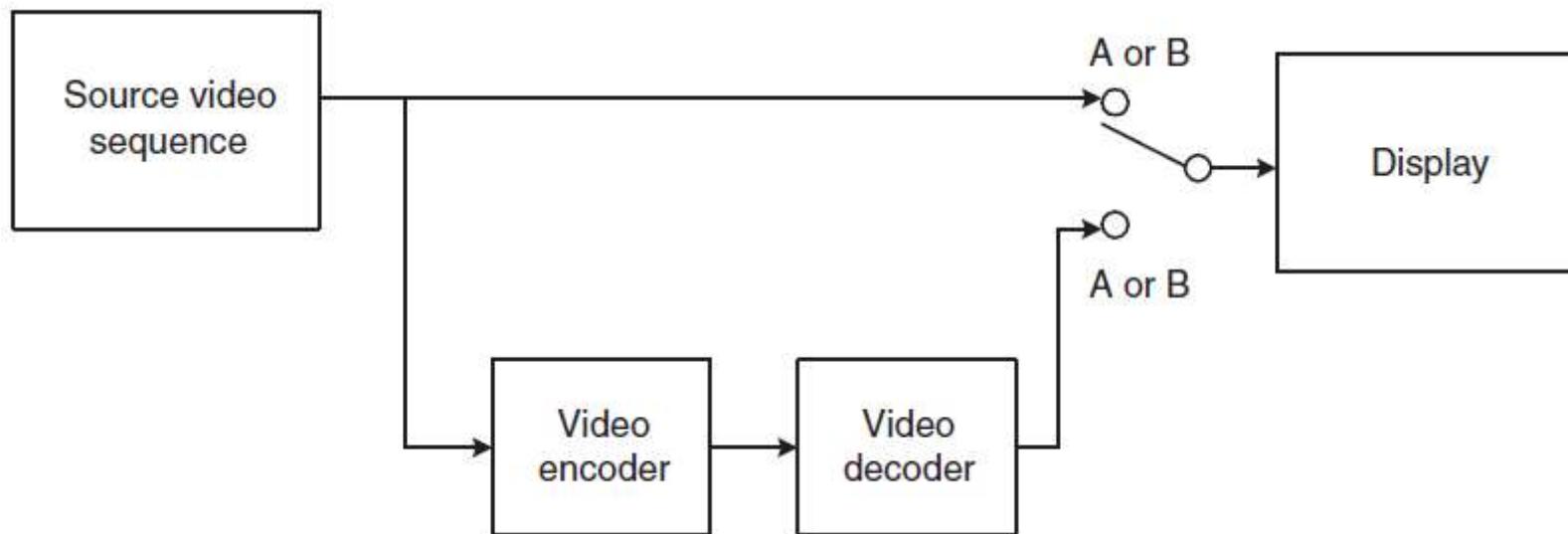
- Quality
 - Subjective quality measurement
 - The order of the two sequences, original and 'impaired', is randomized during the test session so that the assessor does not know which is the original and which is the impaired sequence.
 - At the end of the session, the scores are converted to a normalized range and the end result is a score, sometimes described as a 'mean opinion score' (MOS) that indicates the relative quality of the impaired and reference sequences.
 - Tests such as DSCQS are accepted as realistic measures of subjective visual quality.

2. Video Formats and Quality

- Quality
 - Subjective quality measurement
 - An 'expert' assessor who is familiar with the nature of video compression distortions or 'artefacts' may give a biased score and it is recommended to use 'non-expert' assessors.
 - This means that a large pool of assessors is required because a non-expert assessor will quickly learn to recognize characteristic artefacts in the video sequences and so will become 'expert'.
 - These factors make it expensive and time consuming to carry out the DSCQS tests thoroughly.

2. Video Formats and Quality

- Quality
 - Subjective quality measurement



2. Video Formats and Quality

- Quality
 - Subjective quality measurement



2. Video Formats and Quality

- Quality
 - Objective quality measurement
 - ✓ **PSNR** is a very popular quality measure, widely used to compare the 'quality' of compressed and decompressed video images.

$$PSNR_{dB} = 10 \log_{10} \frac{(2^n - 1)^2}{MSE}$$

2. Video Formats and Quality

- Quality
 - Objective quality measurement

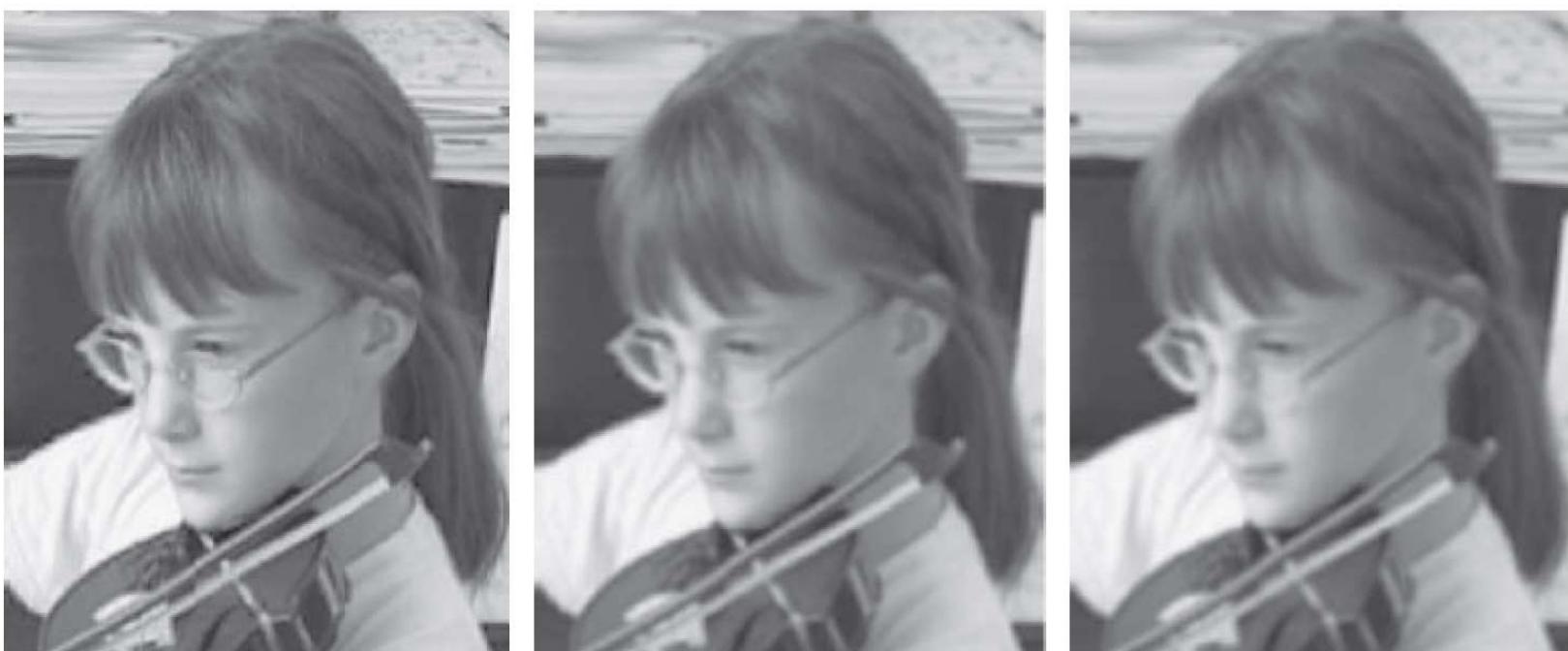


Figure 2.16 PSNR examples: (a) Original; (b) 30.6dB; (c) 28.3dB

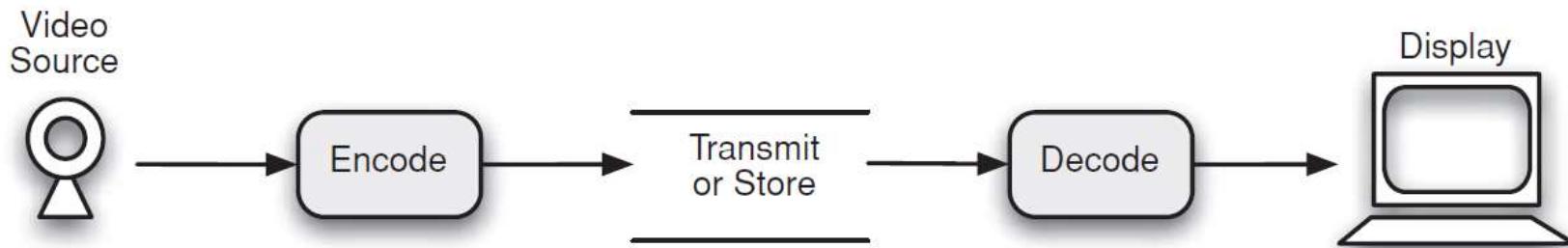
3. Video coding concepts

- Introduction
 - Compression involves a complementary pair of systems, a compressor (**encoder**) and a decompressor (**decoder**).
 - The **encoder** converts the **source** data into a compressed form occupying a **reduced** number of **bits**.
 - The **decoder** converts the **compressed form** back into a representation of the **original** video data.
 - The encoder/decoder pair is often described as a CODEC (enCOder/DECoder).

3. Video coding concepts

- Introduction

- Compression involves a complementary pair of systems, a compressor (**encoder**) and a decompressor (**decoder**).



3. Video coding concepts

- Introduction
 - Data **compression** is achieved by **removing redundancy**: components that are not necessary for faithful reproduction of the data.
 - Many types of data contain **statistical redundancy** and can be effectively compressed using **lossless** compression.
 - The best that can be achieved with **lossless** image **compression** standards is a compression ratio of around **3–4 times**.

3. Video coding concepts

- Introduction
 - **Lossy** compression is necessary to achieve higher compression.
 - Lossy video compression systems are based on the principle of removing **subjective** redundancy.
 - Most video coding methods exploit both **temporal** and **spatial** redundancy to achieve compression.
 - In the **temporal** domain, there is usually a **high correlation** or similarity between frames of video that were **captured at around the same time**.
 - In the **spatial** domain, there is usually a **high correlation** between pixels that are **close to each other**.

3. Video coding concepts

- Introduction

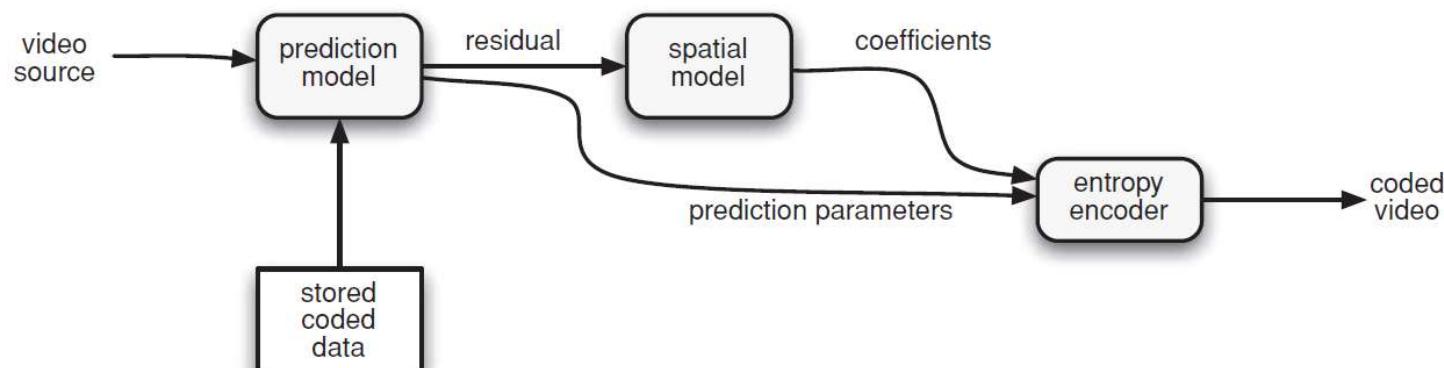


3. Video coding concepts

- Video CODEC

➤ A video encoder consists of three main functional units:

- ✓ a prediction model
- ✓ a spatial model; and
- ✓ an entropy encoder.



3. Video coding concepts

- Video CODEC

- Prediction model

- ✓ Attempts to **reduce redundancy** by exploiting the similarities between neighbouring video frames and/or neighbouring image samples.
 - ✓ The prediction is formed from data in **current**, **previous** and/or **future** frames.
 - ✓ Created by **spatial extrapolation** from neighbouring image samples, **intra prediction**.
 - ✓ Or by compensating for differences between the frames, **inter/motion compensated prediction**.

3. Video coding concepts

- Video CODEC
 - Prediction model
 - ✓ The output of the prediction model is a **residual frame**, created by
 - **subtracting** the **prediction** from the **actual** current frame; and
 - a set of **model parameters** indicating
 - the **intra prediction** type; or
 - Describing how the **motion** was **compensated**

3. Video coding concepts

- Video CODEC
 - Prediction model
 - ✓ The output of the prediction model is a **residual frame**, created by
 - **subtracting** the **prediction** from the **actual** current frame; and
 - a set of **model parameters** indicating
 - the **intra prediction** type; or
 - Describing how the **motion** was **compensated**

3. Video coding concepts

- Video CODEC

- Spatial model

- ✓ The **residual frame** forms the **input** to the **spatial model** which makes use of similarities between local samples in the residual frame to reduce **spatial redundancy**.
 - ✓ Carried out by applying a **transform to the residual samples** and **quantizing** the results.
 - ✓ The output of the spatial model is a set of **quantized transform coefficients**.

3. Video coding concepts

- Video CODEC

- Entropy encoder

- ✓ The parameters of the prediction model, i.e. **intra** and **inter** prediction mode(s), **motion vectors**, and the **coefficients**, are compressed by the **entropy encoder**.
 - ✓ It removes **statistical redundancy** in the data.
 - ✓ Produces a **compressed bit stream** or a file that may be transmitted and/or stored.
 - ✓ A **compressed sequence** consists of coded prediction parameters, coded residual coefficients and header information

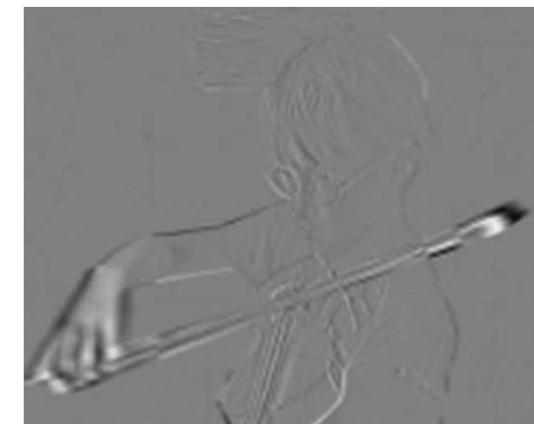
3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ The predicted frame is created from one or more past or future frames known as **reference frames**.
 - ✓ The simplest method of temporal prediction is to use the **previous frame** as the **predictor** for the **current frame**.

3. Video coding concepts

- Prediction Model

- Temporal prediction (inter prediction)
 - ✓ Residual formed by subtracting the predictor (frame 1) from the current frame (frame 2).



- ✓ A lot of energy remains in the residual frame

3. Video coding concepts

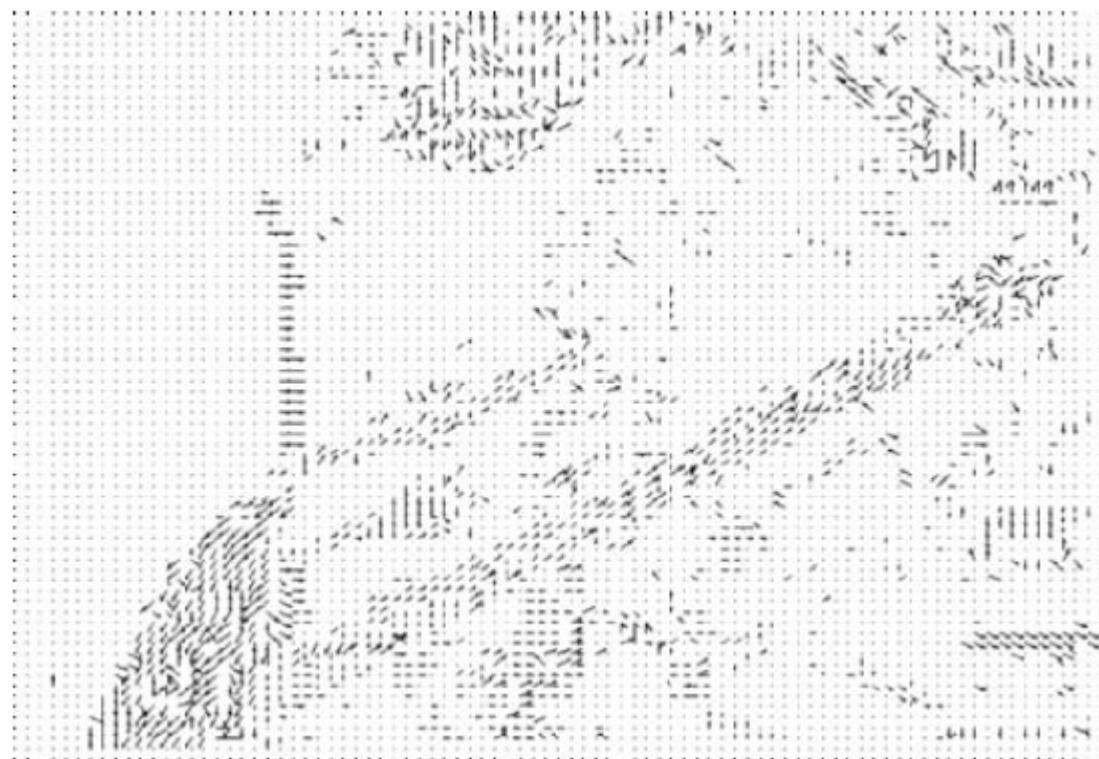
- Prediction Model

- Temporal prediction (inter prediction)

- ✓ Causes of **changes** between video frames include **motion**, **uncovered regions** and **lighting** changes.
 - ✓ It is possible to estimate the **trajectory** of each pixel between successive video frames, producing a field of trajectories (**optical flow**).
 - ✓ this is not a **practical** method of motion **compensation** for several reasons.
 - An accurate calculation of optical flow is very computationally intensive

3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)



3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ A practical and widely used method of motion compensation is to compensate for **movement** of '**blocks**' of the current frame.
 - ✓ The following procedure is carried out for each block of MxN samples in the current frame.
 1. Search an area in the reference frame to find a similar MxN-sample region. This process of finding the best match is known as **motion estimation**

3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - 2. Search an area in the reference frame to find a similar MxN-sample region. This process is known as **motion estimation**.
 - 3. The chosen candidate region becomes the predictor for the current MxN block and is subtracted from the current block to form a **residual (motion compensation)**.
 - 4. The **residual block** and the offset between the current block and the position of the candidate region (**motion vector**) are also encoded.

3. Video coding concepts

- Prediction Model

- Temporal prediction (inter prediction)

- ✓ The decoder uses the received motion vector to re-create the predictor region.
 - ✓ It decodes the residual block, adds it to the predictor and reconstructs a version of the original block.
 - ✓ **Block-based** motion compensation **fits well** with rectangular video frames and with block-based image transforms such as the **Discrete Cosine Transform**.

3. Video coding concepts

- Prediction Model

- Temporal prediction (inter prediction)

- ✓ The **macroblock**, corresponding to a **16 × 16-pixel region** of a frame, is the basic unit for motion compensated prediction in a number of important visual coding standards including **MPEG-2** and **H.264/AVC**.

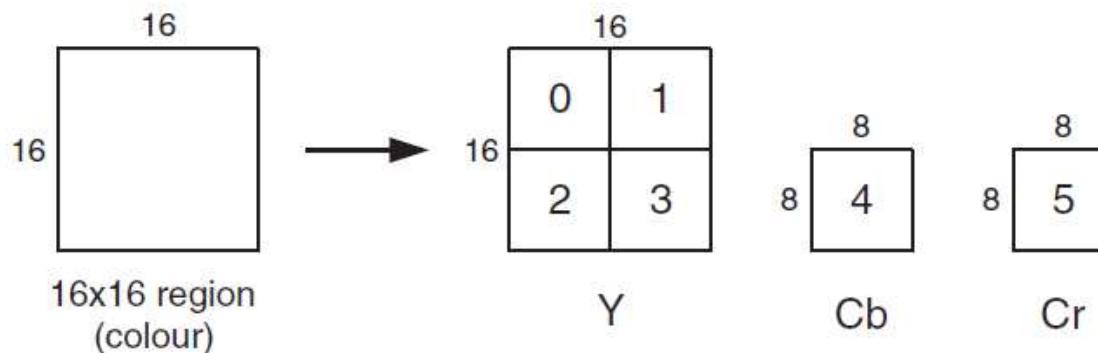


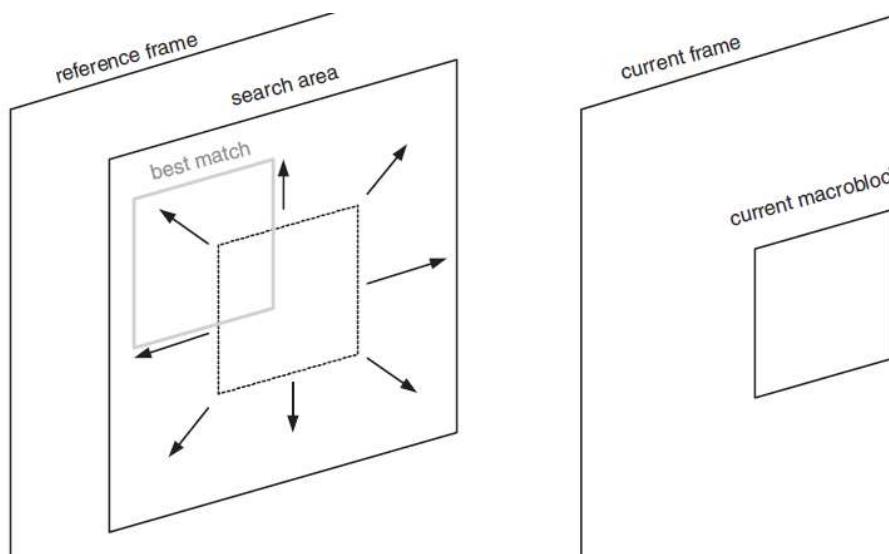
Figure 3.8 Macroblock (4:2:0)

3. Video coding concepts

- Prediction Model

- Temporal prediction (inter prediction)

- ✓ Motion estimation of a macroblock involves a 16 region in a **reference frame** that closely matches the **current macroblock**.



3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ Motion compensation (and block size):



3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ Motion compensation: no compensation



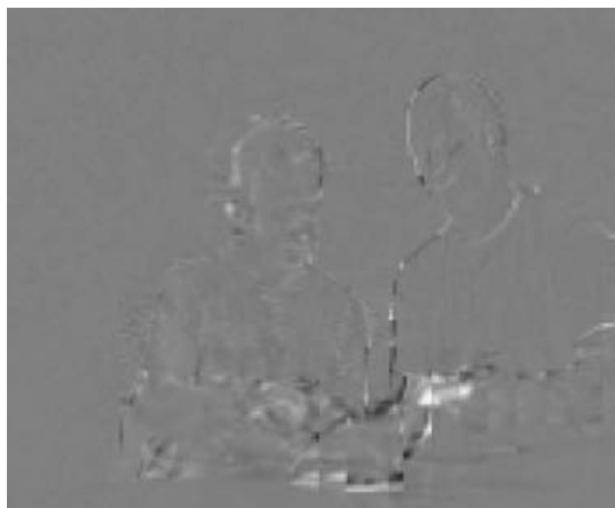
3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ Motion compensation: 16×16 pixels



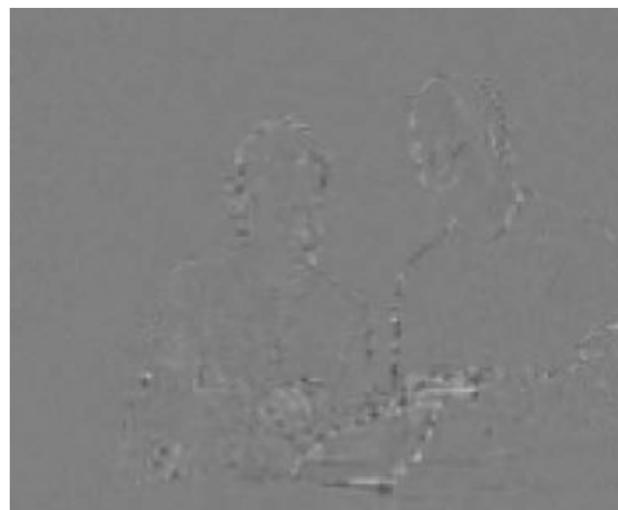
3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ Motion compensation: 8×8 pixels



3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ Motion compensation: 4×4 pixels

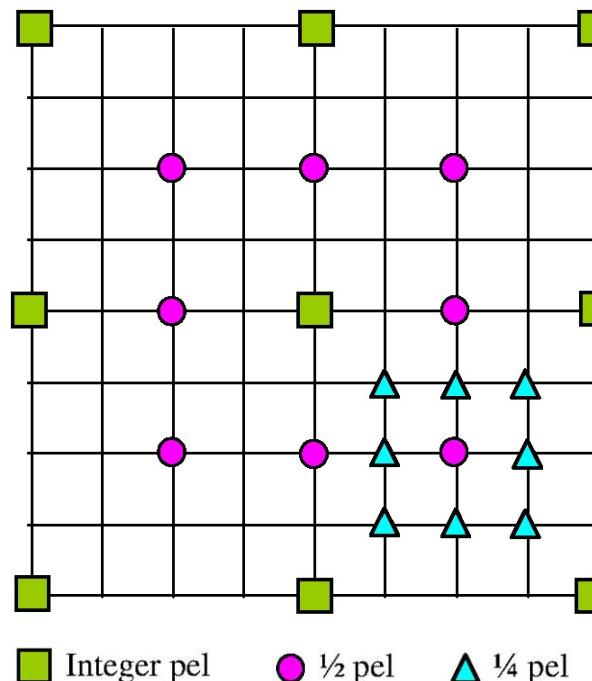


3. Video coding concepts

- Prediction Model

 - Temporal prediction (inter prediction)

 - ✓ Sub-pixel motion compensation

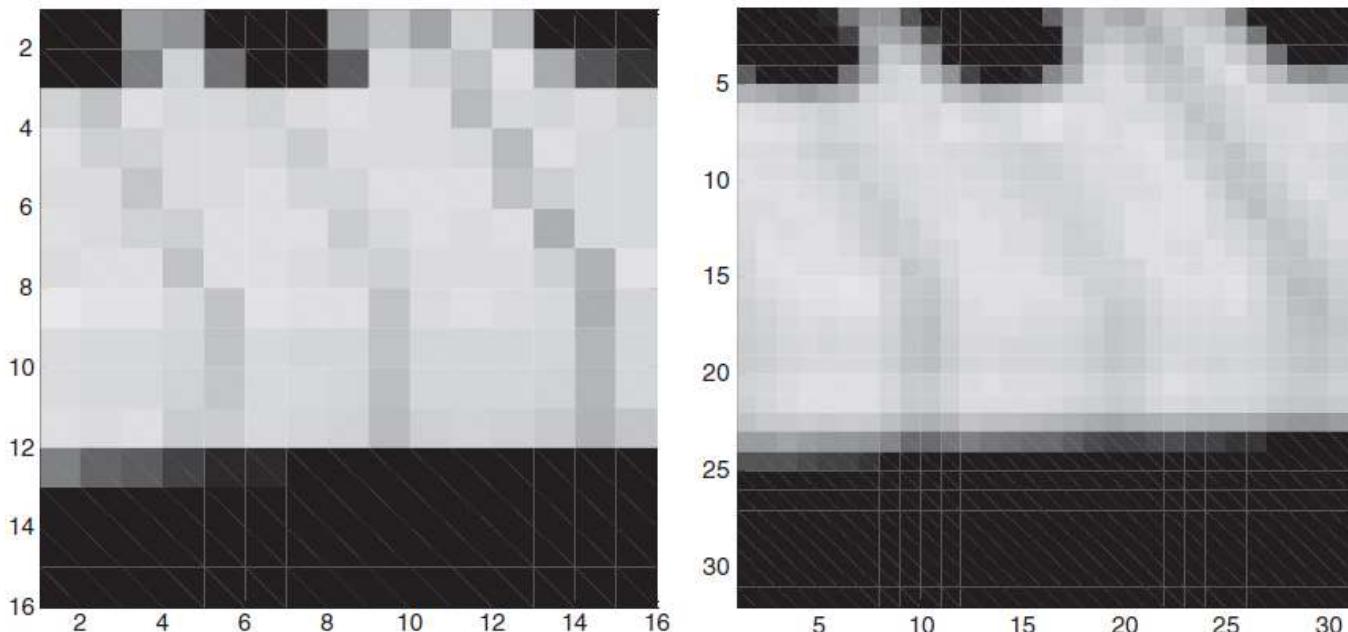


3. Video coding concepts

- Prediction Model

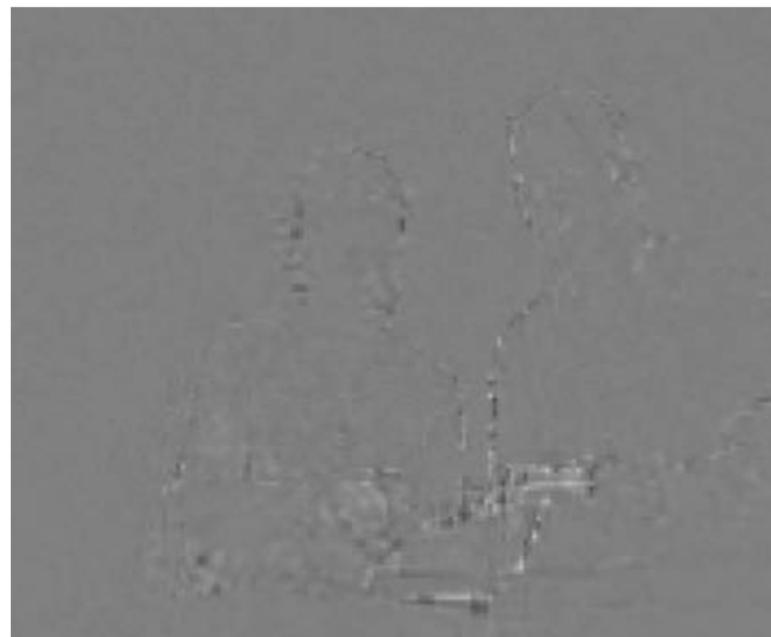
 - Temporal prediction (inter prediction)

 - ✓ Sub-pixel motion compensation: $\frac{1}{2}$ -pixel



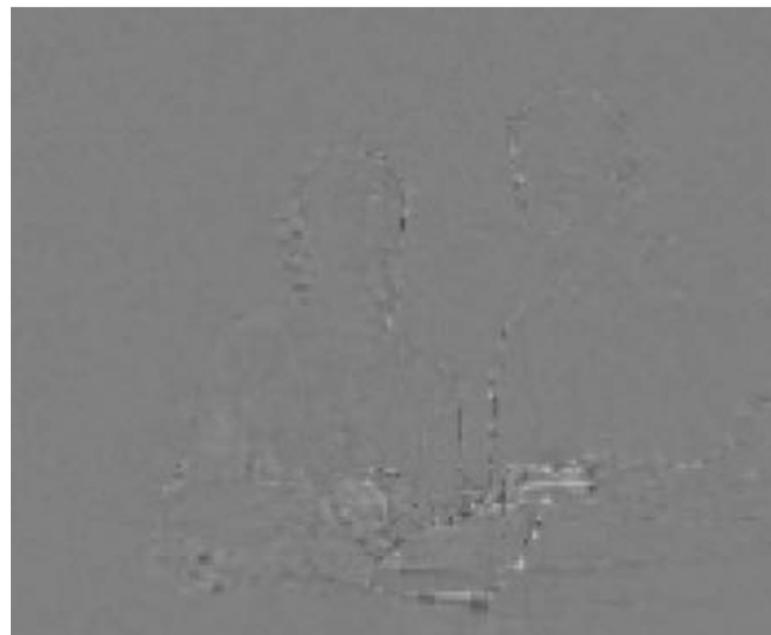
3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ Sub-pixel motion compensation: 4×4 , $\frac{1}{2}$ -pixel



3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ Sub-pixel motion compensation: 4×4 , $\frac{1}{4}$ -pixel



3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ In addition to the extra complexity, sub-pixel motion compensation implies coding penalty.



3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ In addition to the extra complexity, sub-pixel motion compensation implies coding penalty.

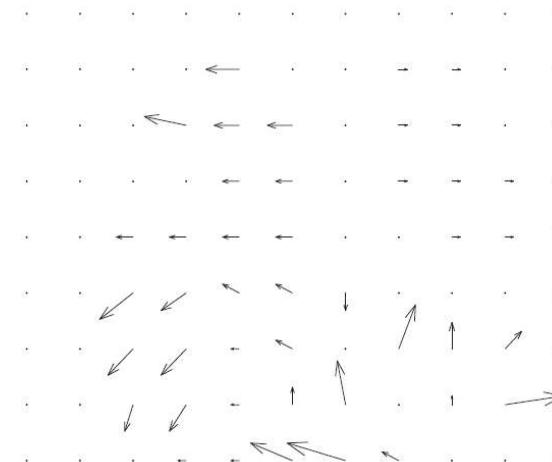


Figure 3.21 Motion vector map : 16×16 blocks, integer vectors

3. Video coding concepts

- Prediction Model
 - Temporal prediction (inter prediction)
 - ✓ In addition to the extra complexity, sub-pixel motion compensation implies coding penalty.

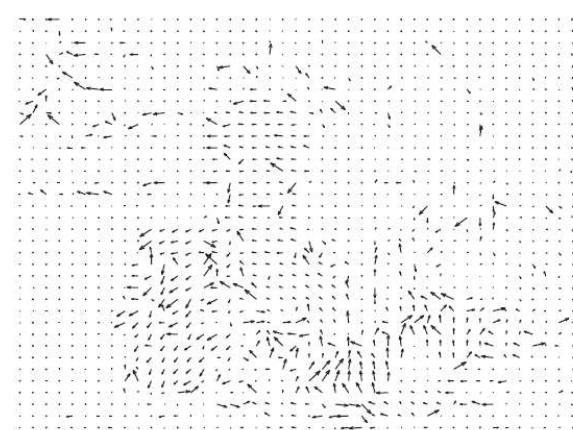
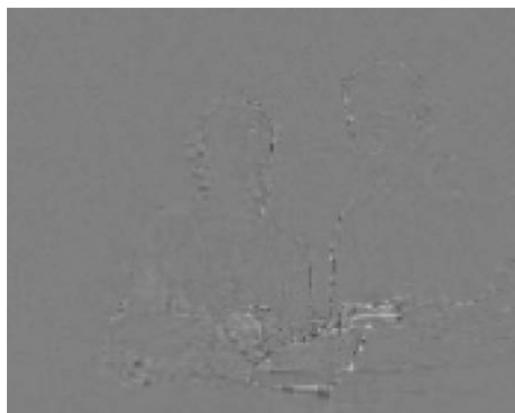


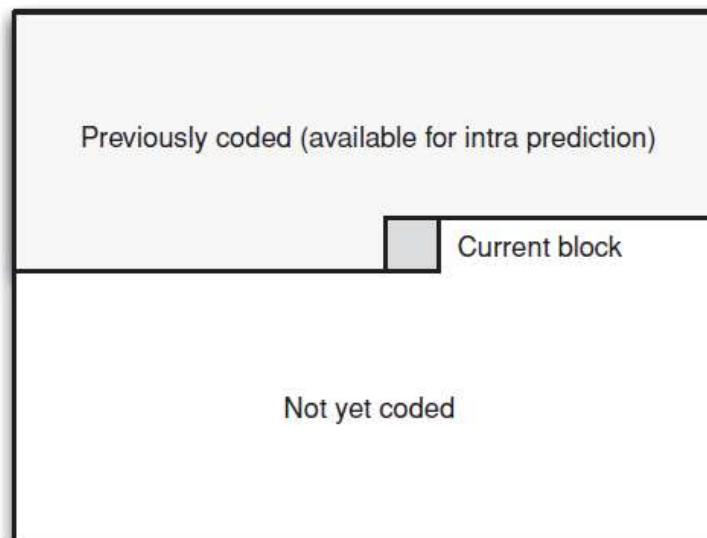
Figure 3.22 Motion vector map : 4×4 blocks, 1/4-pixel vectors

3. Video coding concepts

- Prediction Model

- Spatial prediction (intra prediction)

- ✓ The prediction for the current block of image samples is created from previously-coded samples in the same frame.

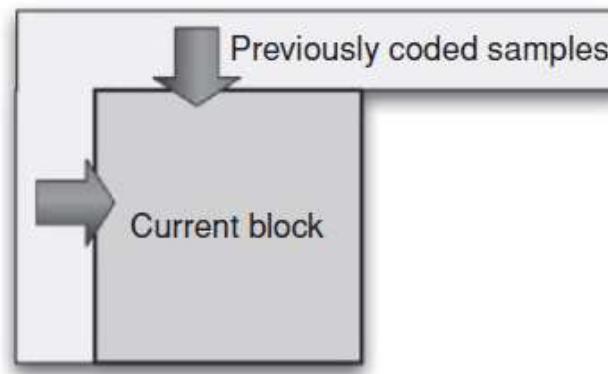


3. Video coding concepts

- Prediction Model

- Spatial prediction (intra prediction)

- ✓ Many different approaches to intra prediction have been proposed. H.264/AVC uses spatial extrapolation to create an intra prediction for a block or macroblock.



3. Video coding concepts

- Prediction Model

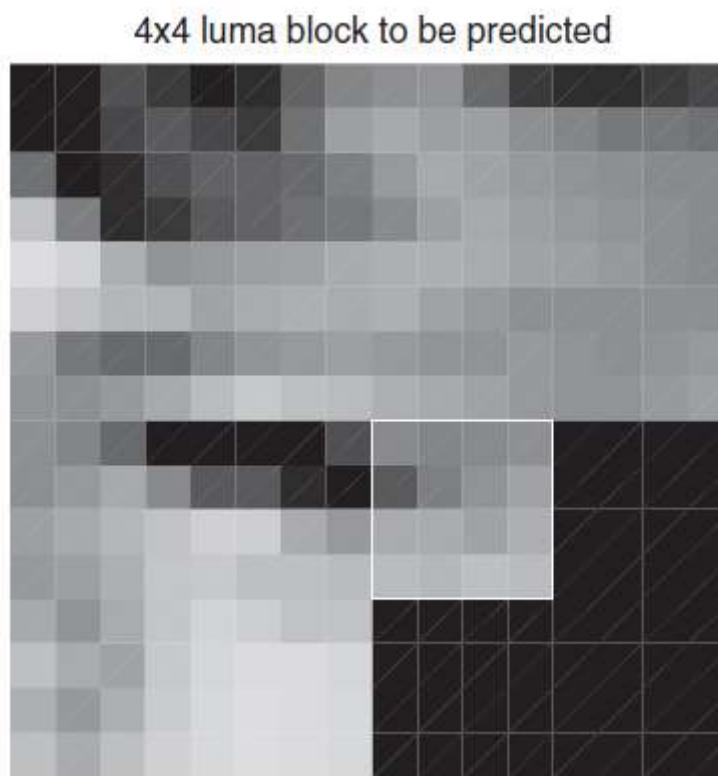
- Spatial prediction (intra prediction)

- ✓ One or more prediction(s) are formed by **extrapolating** samples from the top and/or left sides of the current block.
 - ✓ In general, the **nearest samples** are most likely to be **highly correlated** with the samples in the current block and so only the pixels along the top and/or left edges are used to create the prediction block.
 - ✓ Once the **prediction** has been generated, it is **subtracted** from the **current block** to form a residual in a similar way to inter prediction.

3. Video coding concepts

- Prediction Model

- Spatial prediction (intra prediction)



3. Video coding concepts

- Prediction Model

- Spatial prediction (intra prediction)

- ✓ Example: H.264 4x4 luma prediction modes

M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

Mode 0 (Vertical)

The upper samples A,B,C,D are extrapolated vertically.

Mode 1 (Horizontal)

The left samples I,J,K,L are extrapolated horizontally.

Mode 2 (DC)

All samples in P are predicted by the mean of samples A..D and I..L.

Mode 3 (Diagonal Down-Left)

The samples are interpolated at a 45° angle between lower-left and upper-right.

Mode 4 (Diagonal Down-Right)

The samples are extrapolated at a 45° angle down and to the right.

Mode 5 (Vertical-Left)

Extrapolation at an angle of approximately 26.6° to the left of vertical, i.e. width/height = $\frac{1}{2}$.

Mode 6 (Horizontal-Down)

Extrapolation at an angle of approximately 26.6° below horizontal.

Mode 7 (Vertical-Right)

Extrapolation or interpolation at an angle of approximately 26.6° to the right of vertical.

Mode 8 (Horizontal-Up)

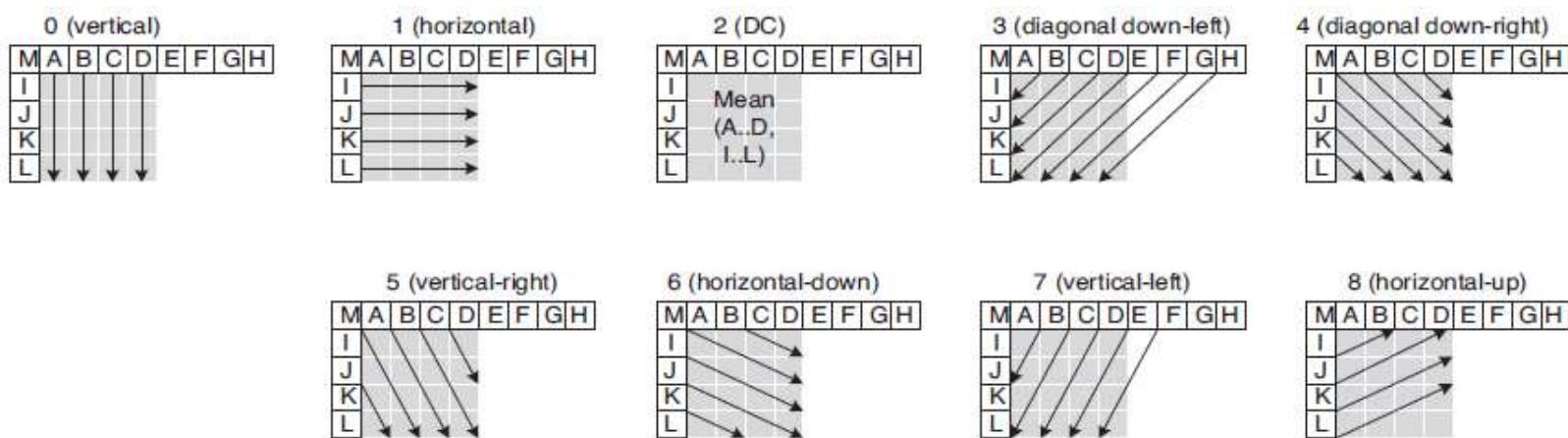
Interpolation at an angle of approximately 26.6° above horizontal.

3. Video coding concepts

- Prediction Model

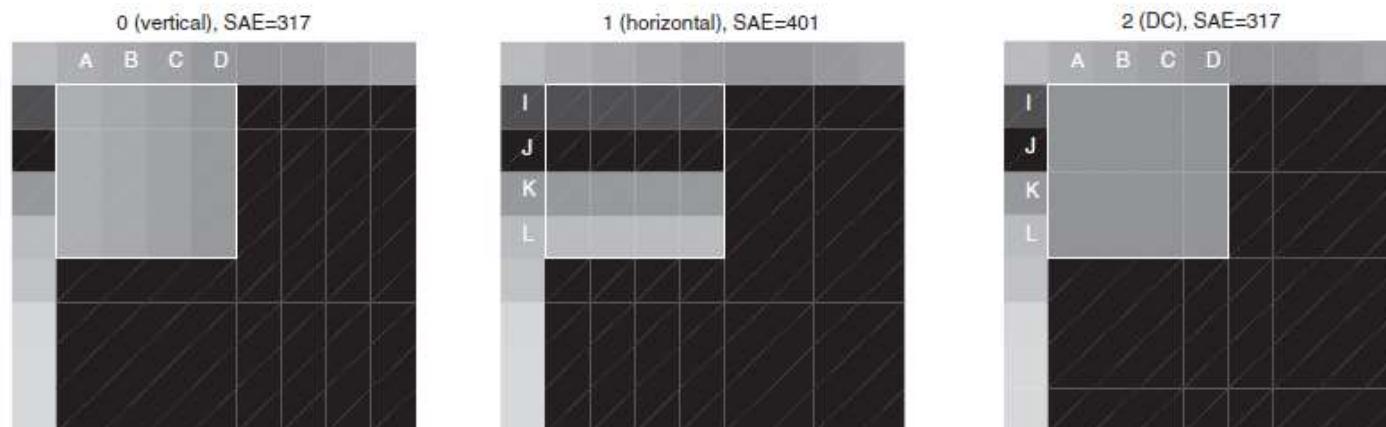
 - Spatial prediction (intra prediction)

 - ✓ Example: H.264 4x4 luma prediction modes



3. Video coding concepts

- Prediction Model
 - Spatial prediction (intra prediction)
 - ✓ Example: H.264 4x4 luma prediction modes

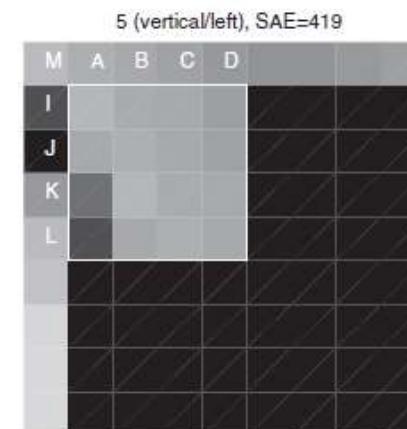
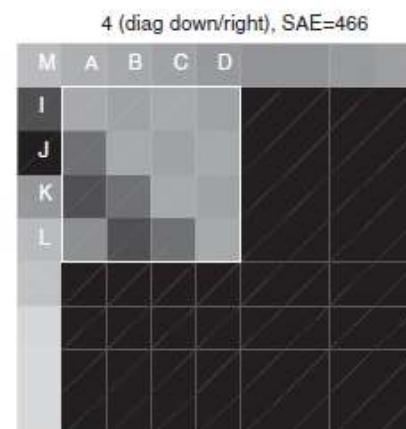
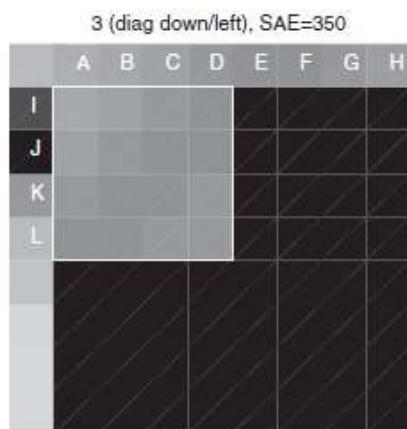


3. Video coding concepts

- Prediction Model

 - Spatial prediction (intra prediction)

 - ✓ Example: H.264 4x4 luma prediction modes



3. Video coding concepts

- Prediction Model
 - Spatial prediction (intra prediction)
 - ✓ Example: H.264 4x4 luma prediction modes

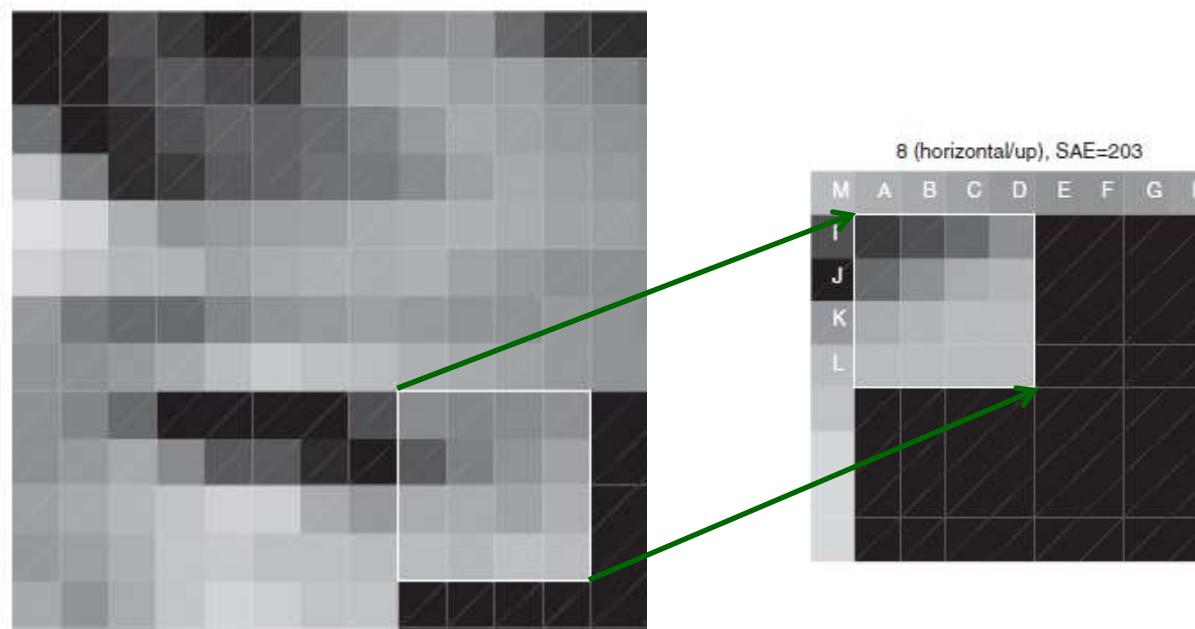
6 (horizontal/down, SAE=530)								
M	A	B	C	D	E	F	G	H
I								
J								
K								
L								

7 (vertical/right), SAE=351								
M	A	B	C	D	E	F	G	H
I								
J								
K								
L								

8 (horizontal/up), SAE=203								
M	A	B	C	D	E	F	G	H
I								
J								
K								
L								

3. Video coding concepts

- Prediction Model
 - Spatial prediction (intra prediction)
 - ✓ Example: H.264 4x4 luma prediction modes



3. Video coding concepts

- Spatial model

➤ The function of the spatial model is to **further decorrelate** image or residual data and to convert it into a form that can be efficiently compressed using an **entropy coder**.

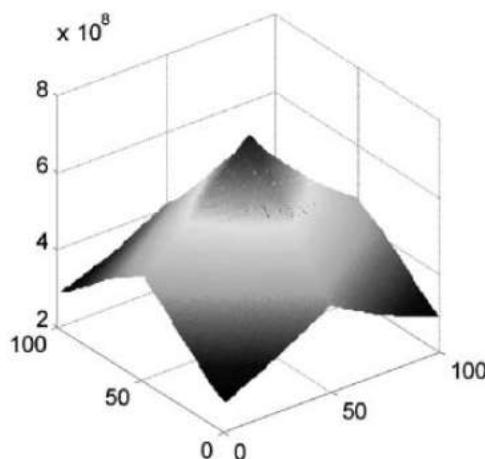


Figure 3.25 2D autocorrelation function of image

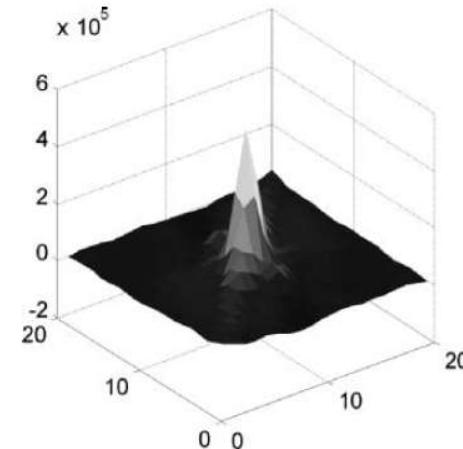


Figure 3.26 2D autocorrelation function of residual

3. Video coding concepts

- Spatial model
 - If the **prediction** is **successful**, the energy in the residual is **lower** than in the original frame and the residual can be represented with **fewer** bits.
 - The function of the **spatial model** is to:
 - ✓ **Further decorrelate** image or residual data; and
 - ✓ Convert it into a form that can be efficiently **compressed** using an **entropy coder**.

3. Video coding concepts

- Spatial model
 - Practical image models typically have three main components:
 - ✓ **Transformation** to **decorrelate** and compact the data;
 - ✓ **Quantization** to **reduce** the **precision** of the transformed data; and
 - ✓ **Reordering** to arrange the data to **group** together **significant values**.

3. Video coding concepts

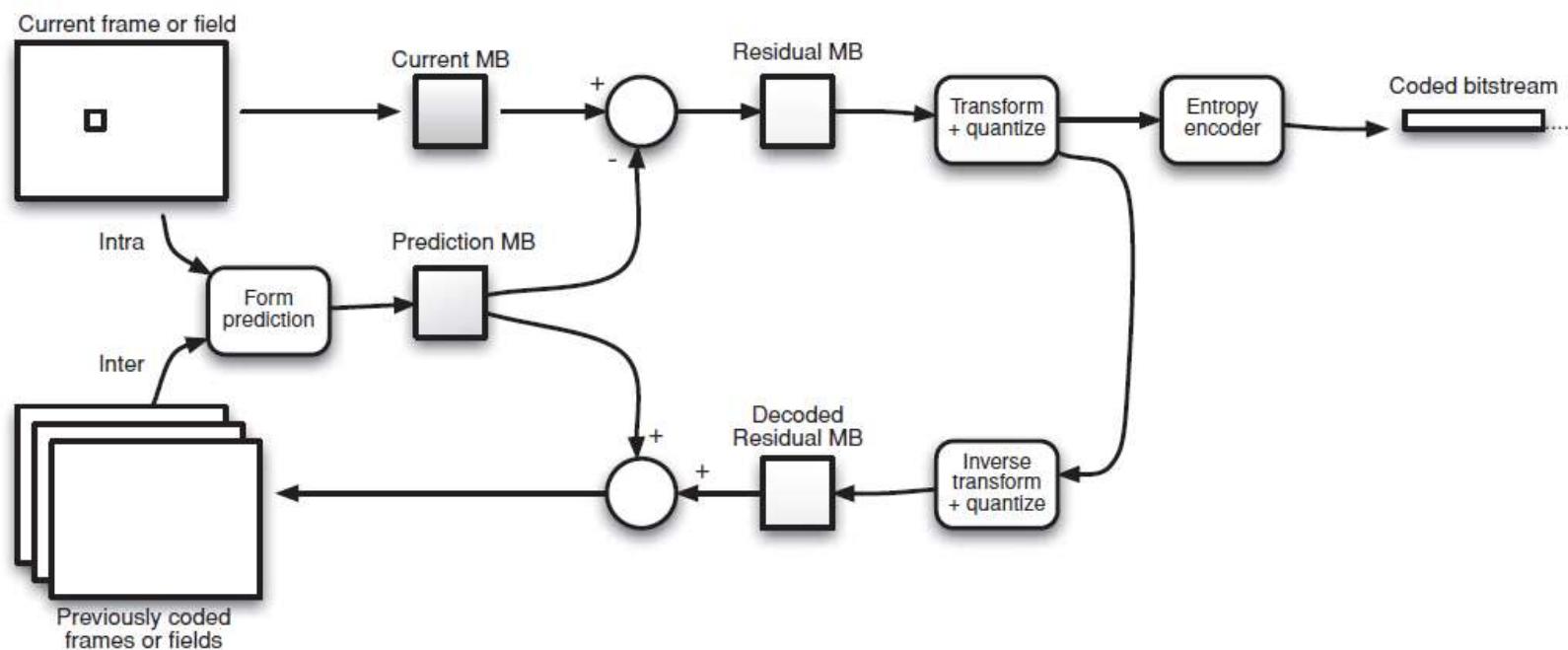
- Entropy coder
 - The **entropy encoder** converts a series of symbols representing elements of the video sequence into a **compressed bitstream** suitable for transmission or storage.

3. Video coding concepts

- Hybrid DPCM/DCT video CODEC model

➤ Example

✓ H.264/AVC encoder

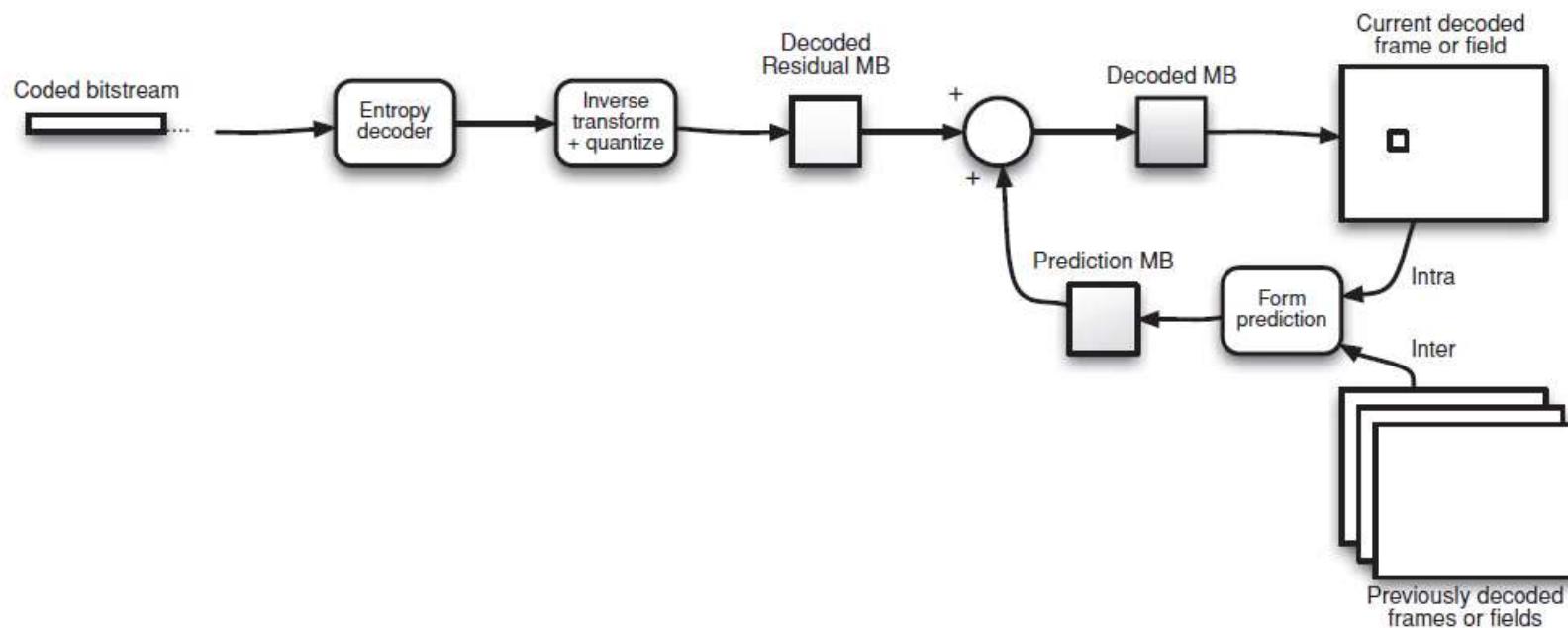


3. Video coding concepts

- Hybrid DPCM/DCT video CODEC model

➤ Example

- ✓ H.264/AVC decoder

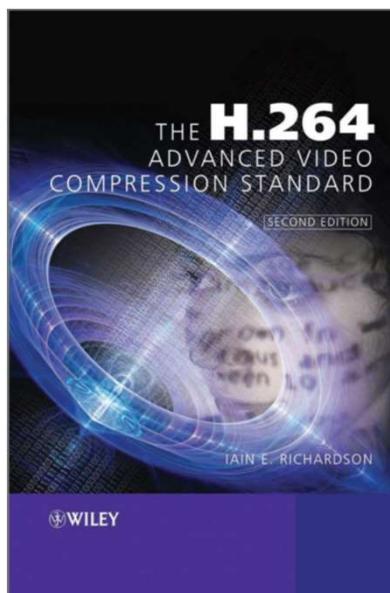
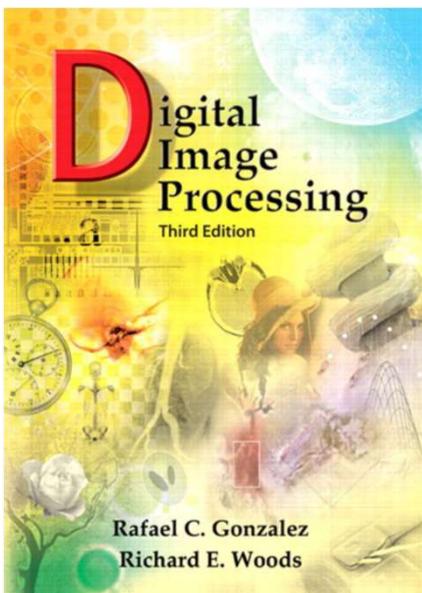


3. Video coding concepts

- The video coding tools described, namely:
 - motion compensated prediction;
 - intra-frame prediction;
 - transform coding;
 - quantization; and
 - entropy coding

form the basis of the reliable and effective coding model that has dominated the field of video compression for over 20 years.

4. Further Reading



IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 22, NO. 12, DECEMBER 2012
1649

Overview of the High Efficiency Video Coding (HEVC) Standard

Gary J. Sullivan, Fellow, IEEE; Jens-Rainer Ohm, Member, IEEE; Woo-Jin Han, Member, IEEE, and Thomas Wiegand, Fellow, IEEE

Abstract—High Efficiency Video Coding (HEVC) is currently being proposed as the newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group (MPEG). HEVC is intended to achieve a significant improvement in compression performance compared to its predecessor, the H.264/MPEG-4 Advanced Video Coding (AVC) [7] standard. This paper provides an overview of the technical features and characteristics of the HEVC standard.

Index Terms—Advanced video coding (AVC), H.264, High Efficiency Video Coding (HEVC), Joint Video Team on Video Coding (JVT), ITU-T, Moving Picture Experts Group (MPEG), MPEG-4, standards, Video Coding Experts Group (VCEG), video compression.

I. INTRODUCTION

The High Efficiency Video Coding (HEVC) standard is the most recent joint video effort of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations, working together in a partnership known as the Joint Video Team (JVT) [1]. The first version of the HEVC standard is expected to be finalized in January 2013, resulting in an annex text that will be published by the ITU-T and ISO/IEC. Additionally, the VCEG is extending the standard to support several additional application scenarios, including extended-range video with enhanced precision and quality, multi-view video, and 3-D stereoscopic 3-D-Derived-view video coding. In ISO/IEC, the HEVC standard will become MPEG-H Part 2 (ISO/IEC 23002-2) and in ITU-T it is likely to become ITU Recommendation H.265.

Manuscript received May 25, 2012; revised August 22, 2012; accepted August 28, 2012. Date of publication October 2, 2012; date of current version December 10, 2012. This work was supported in part by grants from H. Ohm (Corresponding author; e-mail: wjhan@knu.ac.kr); G. J. Sullivan (e-mail: gjsullivan@intel.com); and T. Wiegand (e-mail: thomas.wiegand@frin.de).

Lei Zhou received the Ph.D. degree in Institute of Communication Engineering, RWTH Aachen University, Aachen 52056, Germany (email: zhoul@iis.fraunhofer.de).

Woo-Jin Han is with the Department of Software Design and Management, Korea Maritime University, Pusan 699-791, Korea (e-mail: hanwj@kmu.ac.kr).

T. Wiegand is with the Fraunhofer Institute for Telecommunications, Heinrich Heine University Duesseldorf, Düsseldorf 40235, Germany (e-mail: thomas.wiegand@frin.de).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCSVT.2012.222191

1054-150X/13/\$31.00 © 2012 IEEE

Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 [2] and H.263 [3], ISO/IEC produced MPEG-1 [4] and MPEG-4 Visual [5], and the two standards were jointly developed. The ITU-T also produced H.264/MPEG-4 Advanced Video Coding (AVC) [7] standards. The two standards that were jointly produced have had a significant impact on the evolution of video coding. There is a variety of products that are increasingly prevalent in our daily lives. Throughout this evolution, continued efforts have been made to improve the efficiency of video coding. The new characteristics such as data loss robustness, while considering the computational resources that were practical for use in products such as mobile phones.

The major video coding standard directly preceding the HEVC project was H.264/MPEG-4 AVC, which was initially developed by the ITU-T and ISO/IEC in 2003 [6]. The standard was extended in several important ways from 2003–2009. H.264/MPEG-4 AVC has been an enabling technology for digital television, video distribution over the Internet, high-definition (HD) TV signals over satellite, cable, and terrestrial transmission systems; video content acquisition and editing systems; video content delivery systems; video storage; portable network videos; Blu-ray Discs; and real-time conversational applications such as video chat, video conferencing, and instant messaging.

However, an increasing diversity of services, the growing popularity of HD video, and the ever-increasing resolution (e.g., 4K, 8K, and beyond) are creating even stronger needs for coding efficiency superior to H.264/MPEG-4 AVC's capabilities. The need is even stronger when higher resolution is accompanied by stereo or multi-view video, which is becoming the norm in consumer video applications targeting mobile devices and tablet PCs, as well as the transmission needs for video content delivery. These are the severe challenges of today's networks. An increasing desire for higher quality and resolutions is also arising in medical applications.

HEVC has been designed to address essentially all existing applications of H.264/MPEG-4 AVC and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures. The syntax of HEVC