



# Digital Image Processing

## References

---

1. Rafael C. Gonzalez and Richard E. Woods, “[Digital Image Processing](#)”, Pearson Education, Inc., Second Edition, 2004.
2. Anil K. Jain, “[Fundamentals of Digital Image Processing](#)”, Prentice Hall of India, 2002.
3. William K. Pratt, “[Digital Image Processing](#)”, 2<sup>nd</sup> Edition, Wiley & Sons, Inc., 1991.

## Course Outline

---

1. Introduction
2. Digital Image Fundamentals
3. Image Transforms
4. Image Enhancement and Restoration
5. Image Segmentation and Recognition
6. Image Compression



## UNIT I

# DIGITAL IMAGE FUNDAMENTALS

- What is an Image?

Picture, photograph

---

Visual data

Usually two or three dimensional

## What is a digital image?

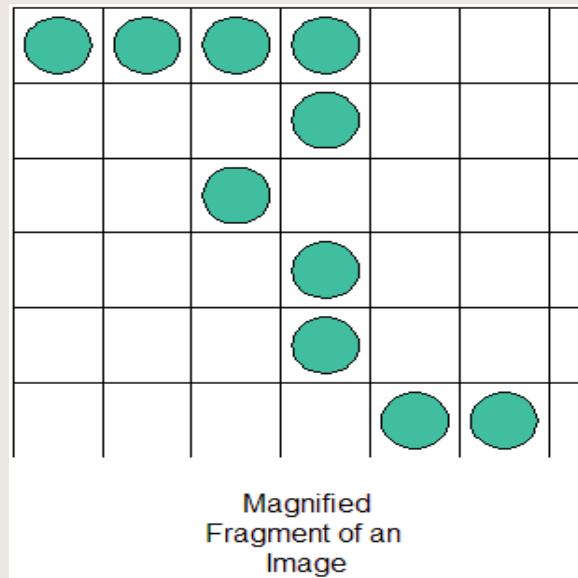
An image which is “discretized,”, i.e., defined on a discrete grid

Two-dimensional collection of light values (or gray values)

# Basic Terminology

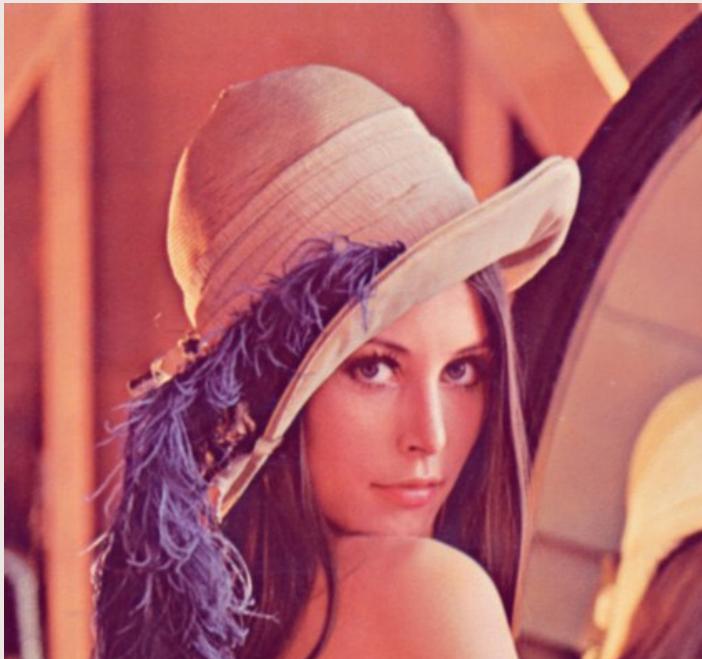
What is a digital image?

- A representation of the original image by a discrete set of data points
- Each of these data points is called a picture element, commonly referred to as a “pixel”



# Digital Image Generation

- Generating a digital image
- Place a regular 2D grid over an image and read the colour value at the intersections of the grid => set of data points for that image



# What is digital image processing?

Digital image processing is the study of representation and manipulation of pictorial information by a computer.

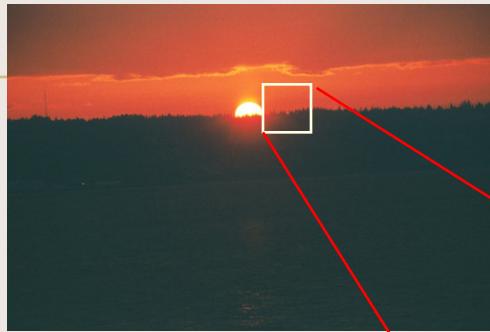
Improve pictorial information for better clarity  
(human interpretation)

Automatic machine processing of scene data  
(interpretation by a machine/non-human, storage, transmission)

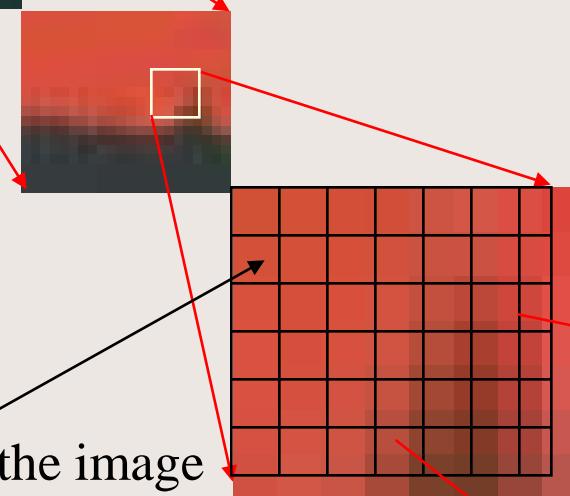
## What is image interpretation?

Assign meaning to an ensemble of recognized objects

## Digital Image



Digital image = a multidimensional array of numbers (such as intensity image) or vectors (such as color image)



Each component in the image called pixel associates with the pixel value (a single number in the case of intensity images or a vector in the case of color images).

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

# Important steps in a typical image processing system

## Step 1: Image acquisition

---

Capturing visual data by an image sensor

## Step 2: Discretization / digitalization; Quantization ; Compression

Convert data into discrete form; compress for efficient storage/transmission

## Step 3: Image enhancement and restoration

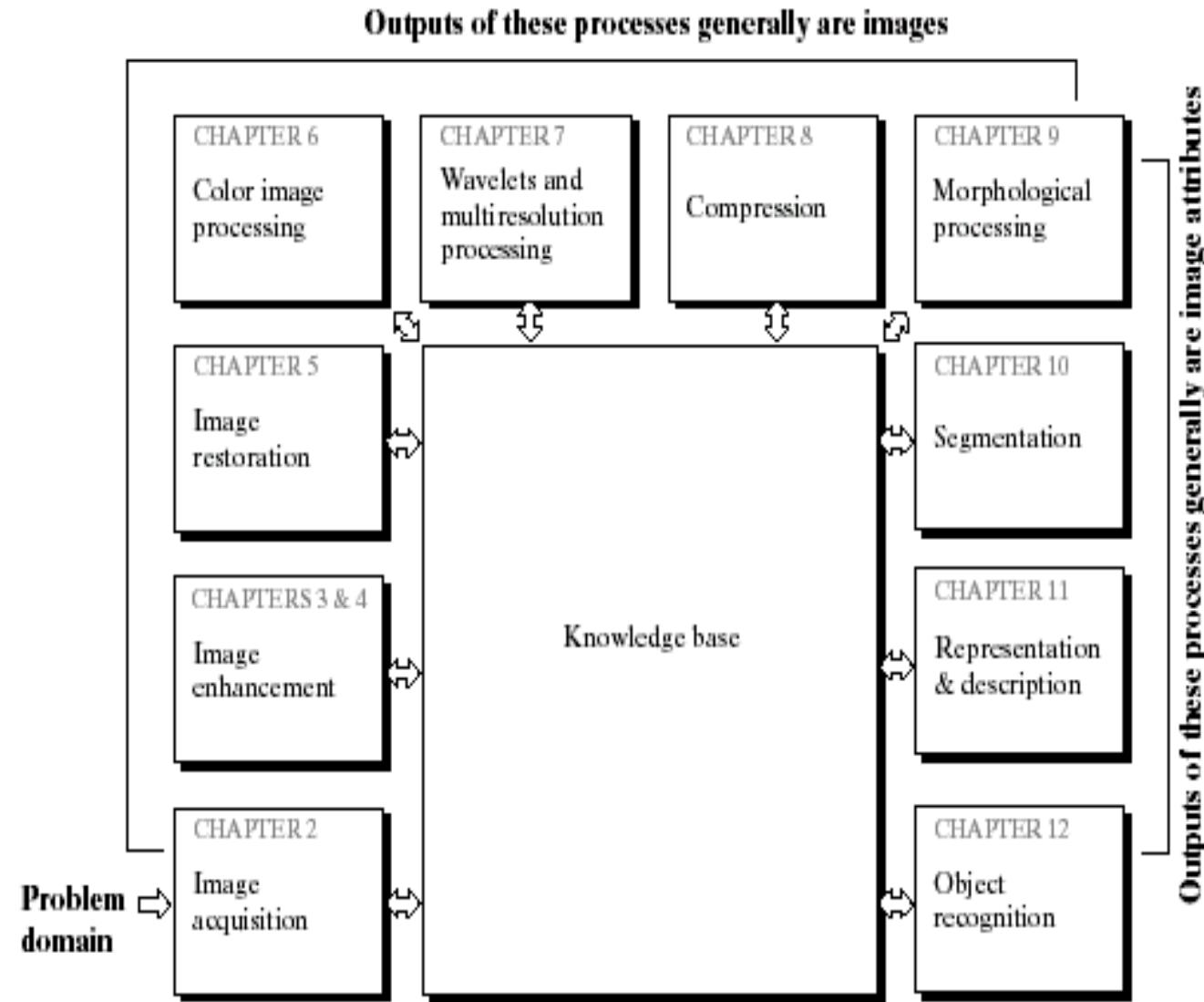
Improving image quality (low contrast, blur noise)

## Step 4: Image segmentation

Partition image into objects or constituent parts

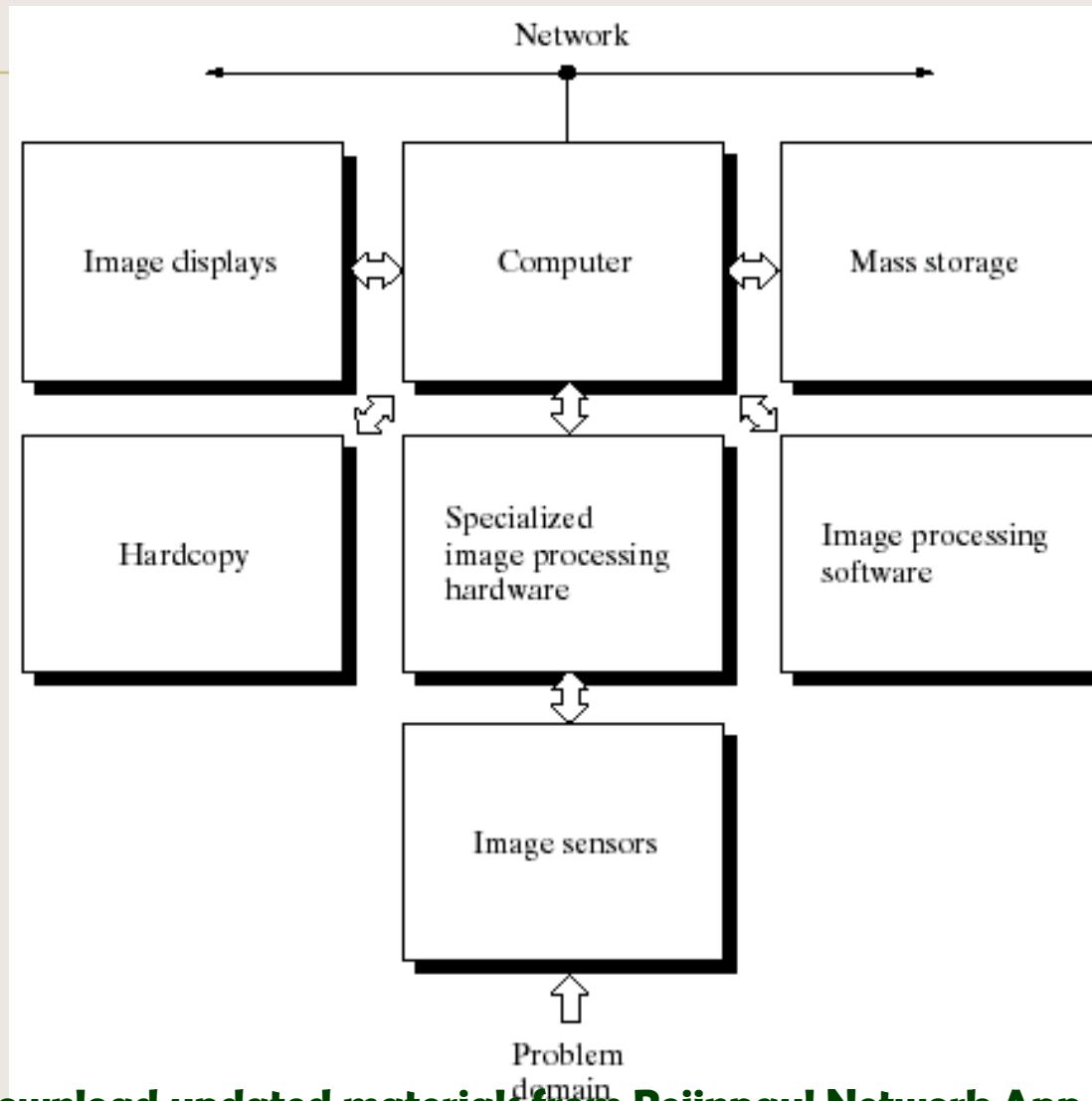
# Chapter 1: Introduction

**FIGURE 1.23**  
Fundamental  
steps in digital  
image processing.





# Chapter 1: Introduction



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

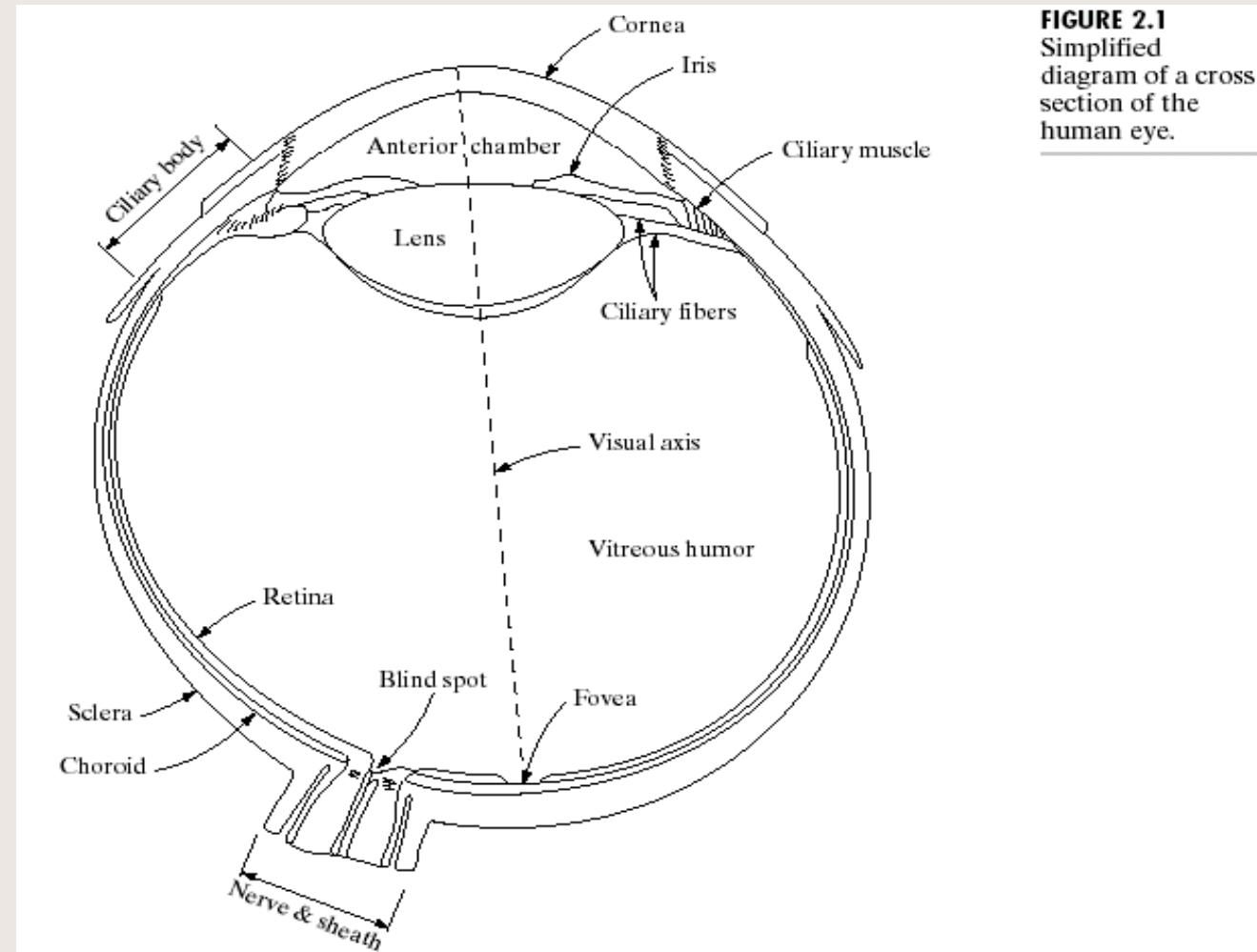
- 
- ✓ Image enhancement and restoration both involve techniques to make a better image
  
  - ✓ Image compression involves development of techniques to make smaller files, while still retaining high quality images



# ELEMENTS OF VISUAL PERCEPTION

## ELEMENTS OF VISUAL PERCEPTION

### Human Visual Perception



**FIGURE 2.1**  
Simplified  
diagram of a cross  
section of the  
human eye.

# The Human Eye

---

- Diameter: 20 mm
- 3 membranes enclose the eye
  - Cornea & sclera
  - Choroid
  - Retina

# The Choroid

---

- The choroid contains blood vessels for eye nutrition and is heavily pigmented to reduce extraneous light entrance and backscatter.
- It is divided into the ciliary body and the iris diaphragm, which controls the amount of light that enters the pupil (2 mm ~ 8 mm).

# The Lens

---

- The lens is made up of fibrous cells and is suspended by fibers that attach it to the ciliary body.
- It is slightly yellow and absorbs approx. 8% of the visible light spectrum.
- The *lens* contains 60-70% water, 6% of fat.

# The Retina

---

- The retina lines the entire posterior portion.
- Discrete light receptors are distributed over the surface of the retina:
  - cones (6-7 million per eye) and
  - rods (75-150 million per eye)

# Cones

---

- Cones are located in the fovea and are sensitive to color.
- Each one is connected to its own nerve end.
- Cone vision is called *photopic* (or bright-light vision or day vision).
- There are three types of cones: Red, Green, and Blue

# Rods

---

- ✓ Rods are giving a general, overall picture of the field of view
- ✓ They see only brightness (not color) and are not involved in color vision.
- Several rods are connected to a single nerve and are sensitive to low levels of illumination (*scotopic* or dim-light vision or night vision).

# MESOPIC VISION

---

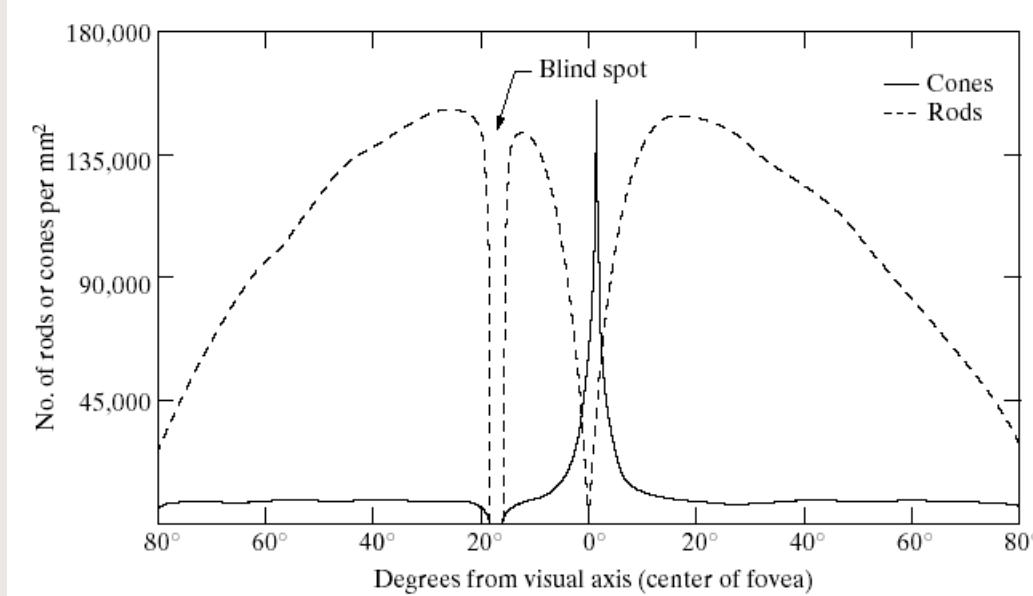
- Intermediate region of illumination – between dim and bright light
- Both rods and cones are active

# Receptor Distribution

---

- The distribution of receptors is radially symmetric about the fovea.
- Cones are most dense in the center of the fovea while rods increase in density from the center out to approximately 20% off axis and then decrease.

## Cones & Rods



**FIGURE 2.2**  
Distribution of rods and cones in the retina.

# The Fovea

---

- The fovea is circular (1.5 mm in diameter) but can be assumed to be a square sensor array (1.5 mm x 1.5 mm).
- The density of cones: 150,000 elements/mm<sup>2</sup> ~ 337,000 for the fovea.
- A CCD imaging chip of medium resolution needs 5 mm x 5 mm for this number of elements

# BLIND SPOT

---

- *Blind spot* is the region of emergence of the optic nerve from the eye
- Place on the retina where optic nerve connects, and which consists of no light sensors

# Image Formation in the Eye

---

- The eye lens (if compared to an optical lens) is flexible.
- It gets controlled by the fibers of the ciliary body and to focus on distant objects it gets flatter (and vice versa).

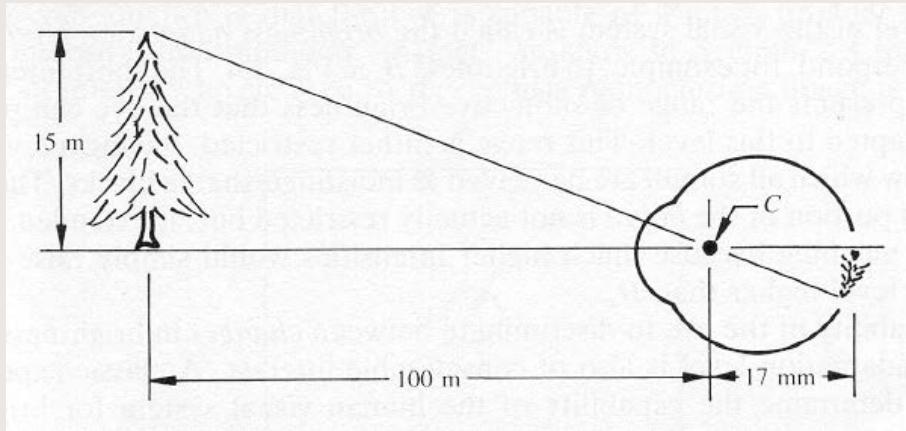
# Image Formation in the Eye

---

- Distance between the center of the lens and the retina (*focal length*):
  - varies from 17 mm to 14 mm (refractive power of lens goes from minimum to maximum).
- Objects farther than 3 m use minimum refractive lens powers (and vice versa).

# Image Formation in the Eye

- Example:
  - Calculation of retinal image of an object

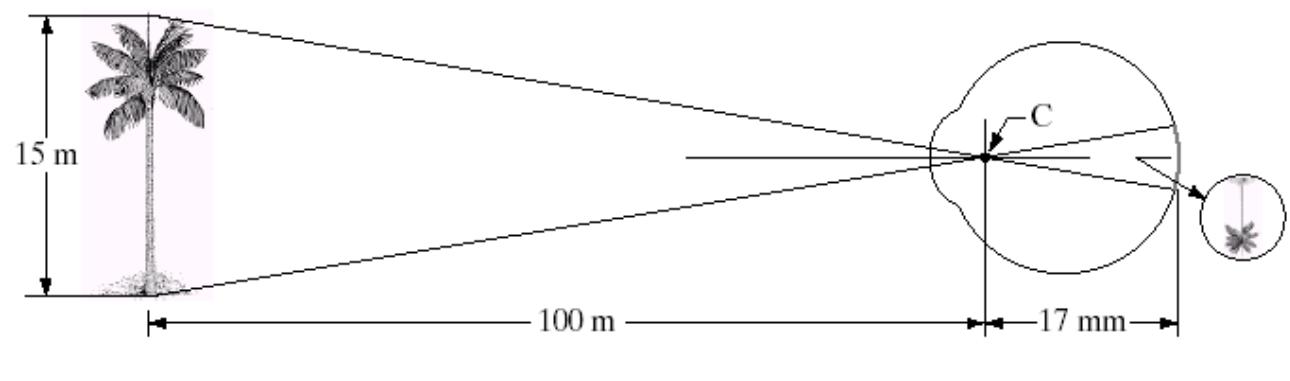


$$\frac{15}{100} = \frac{x}{17}$$

$$x = 2.55\text{mm}$$

# Image Formation in the Eye

**FIGURE 2.3**  
Graphical representation of the eye looking at a palm tree. Point C is the optical center of the lens.



# Image Formation in the Eye

---

- Perception takes place by the relative excitation of light receptors.
- These receptors transform radiant energy into electrical impulses that are ultimately decoded by the brain.

# LUMINANCE OR INTENSITY

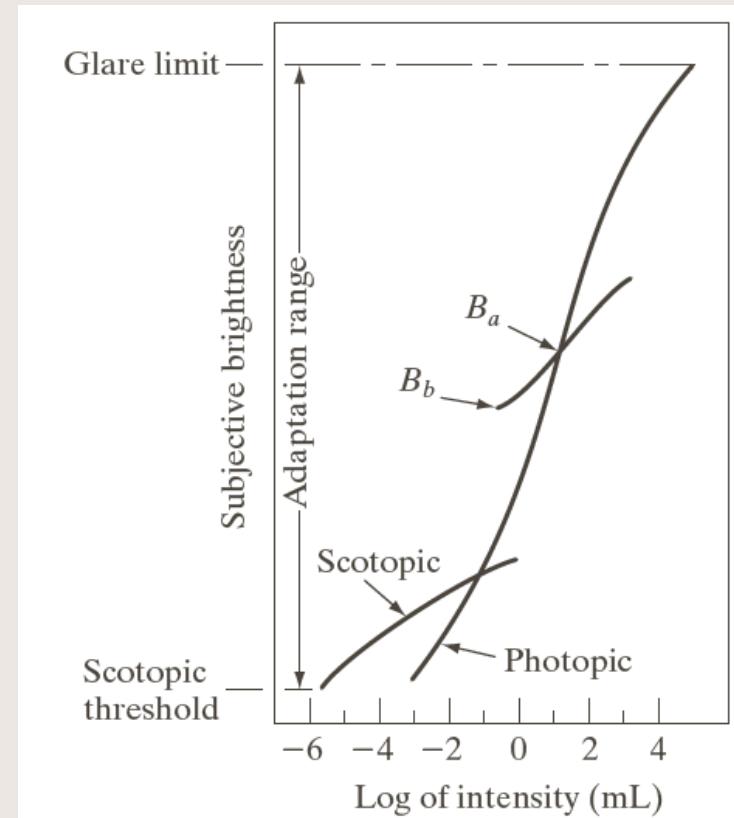
- Emitting or reflecting light from the object
- The luminance of an object is independent of the luminance of the surrounding objects

# BRIGHTNESS

---

- Brightness is the perceived luminance
- Depends on the luminance of the surround
- Two objects with different surroundings could have identical luminance but different brightness

- Range of subjective Brightness  
Sensations Showing a particular  
Adaptation level

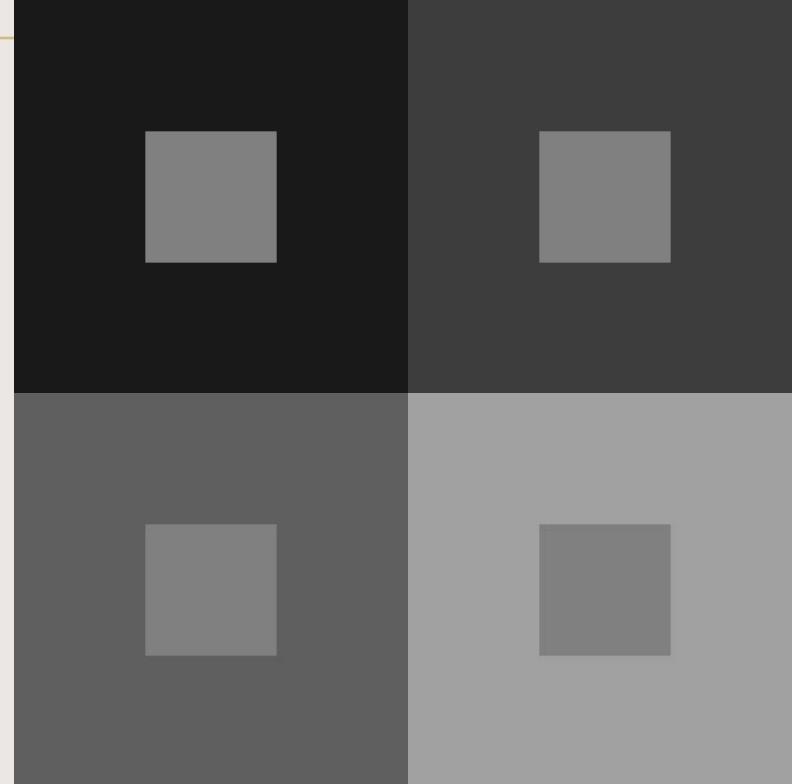


# CONTRAST

---

- **Contrast** is the difference in visual properties that makes an object (or its representation in an image) distinguishable from other objects and the background
- Contrast is determined by the difference in the color and brightness of the light reflected or emitted by an object and other objects within the same field of view .

## Brightness Adaptation of Human Eye (cont.)



***Simultaneous contrast.*** All small squares have exactly the same intensity but they appear progressively darker as background becomes lighter.

# HUE

---

- The hue of a color refers to its “redness”, “greenness” and so on.
- A **hue** refers to the gradation of color within the optical spectrum, or visible spectrum, of light.
- "Hue" may also refer to a particular color within this spectrum, as defined by its dominant wavelength,
- or the central tendency of its combined wavelengths. For example, a light wave with a central tendency within 565-590 nm will be yellow.

# SATURATION

---

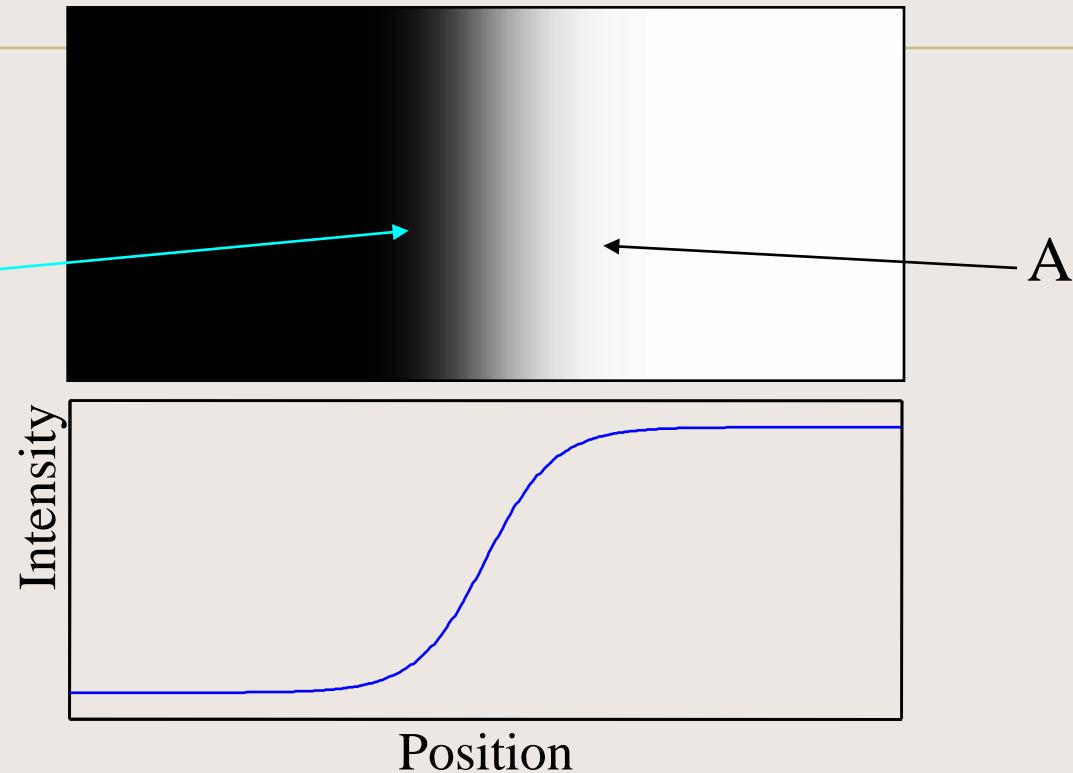
- **Saturation** refers to the intensity of a specific hue.
- In art or when working with colors, saturation is the amount of color a certain color has. For example, black and white have no saturation and bright red has 100% saturation

# MACH BAND EFFECT

---

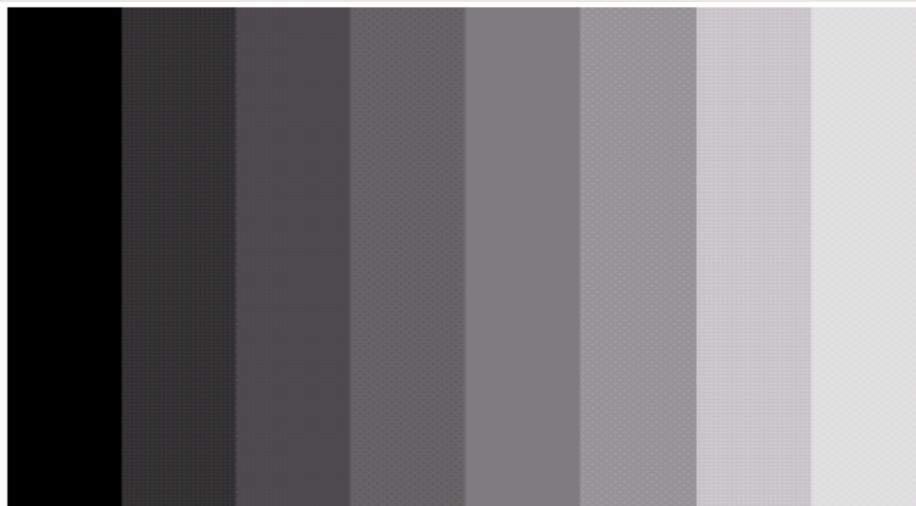
- The spatial interaction from an object and its surroundings creates a phenomenon called mach band effect.
- The visual system tends to undershoot or overshoot around the boundary of regions of different intensities

## Brightness Adaptation of Human Eye (cont.)



In area A, brightness perceived is darker while in area B is brighter. This phenomenon is called **Mach Band Effect**.

## MACH BAND EFFECT



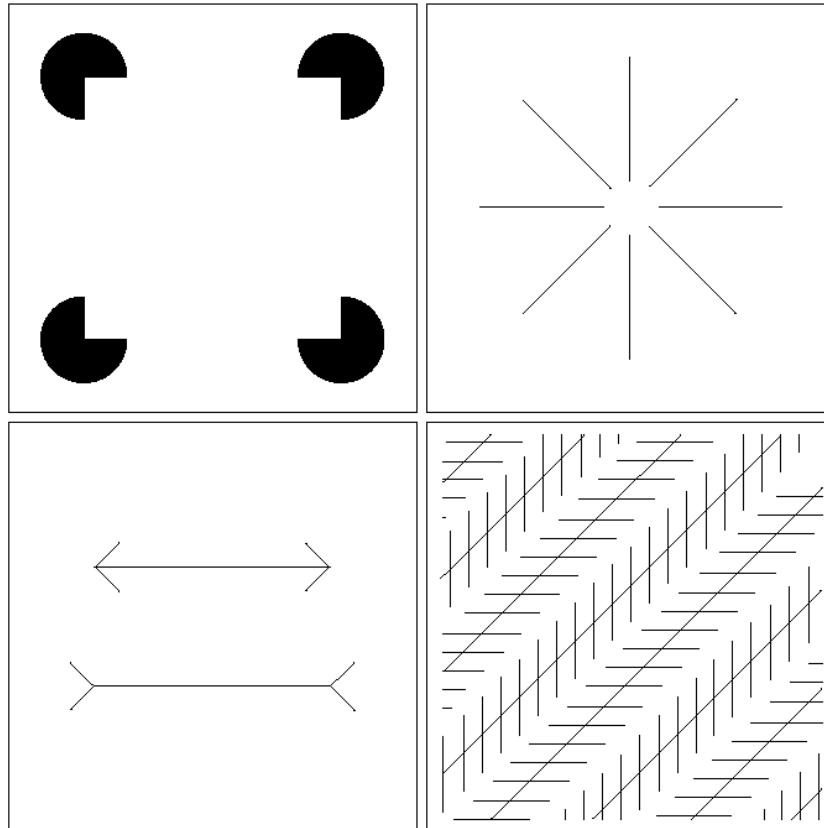
a  
b

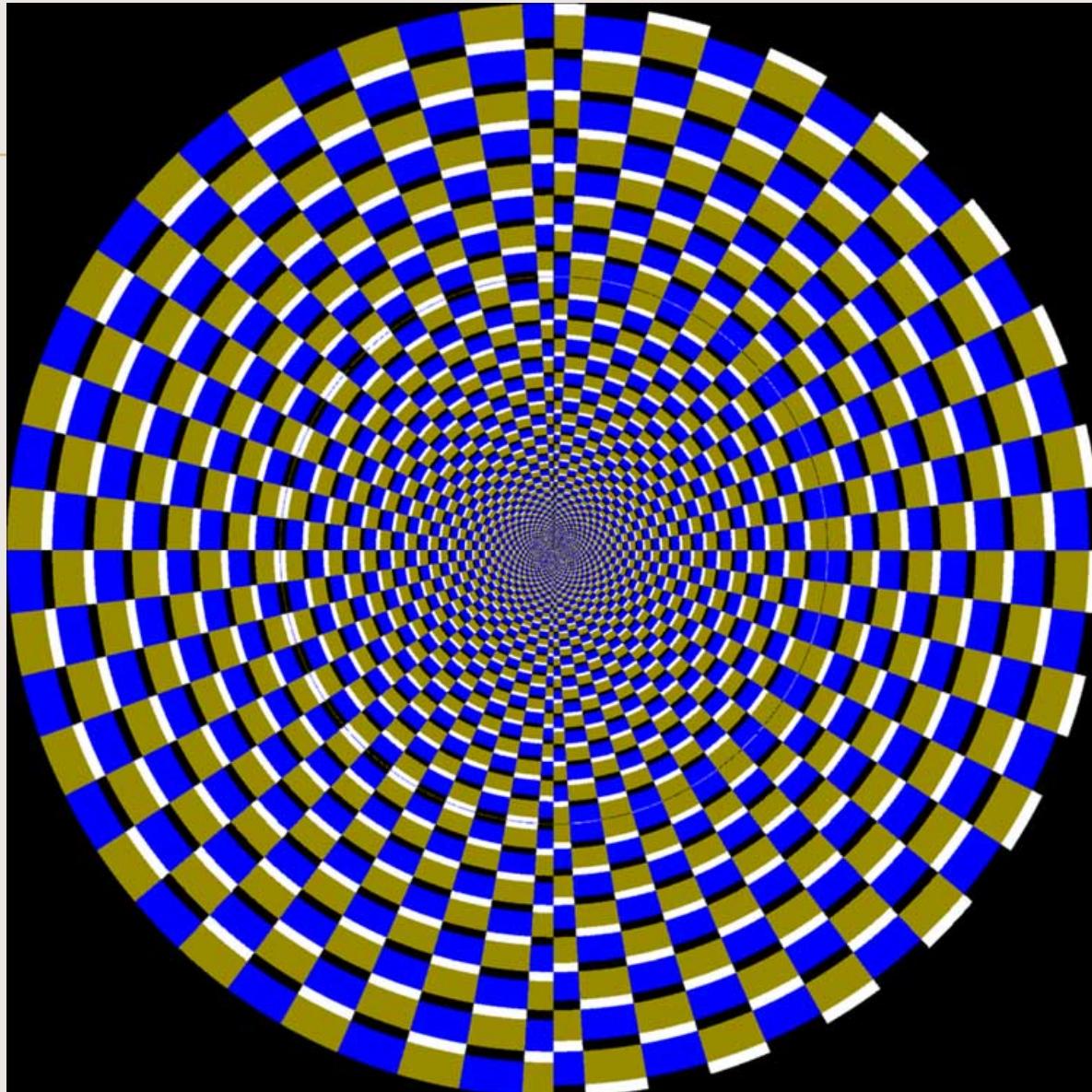
**FIGURE 2.7**

(a) An example showing that perceived brightness is not a simple function of intensity. The relative vertical positions between the two profiles in (b) have no special significance; they were chosen for clarity.

a  
b  
c  
d

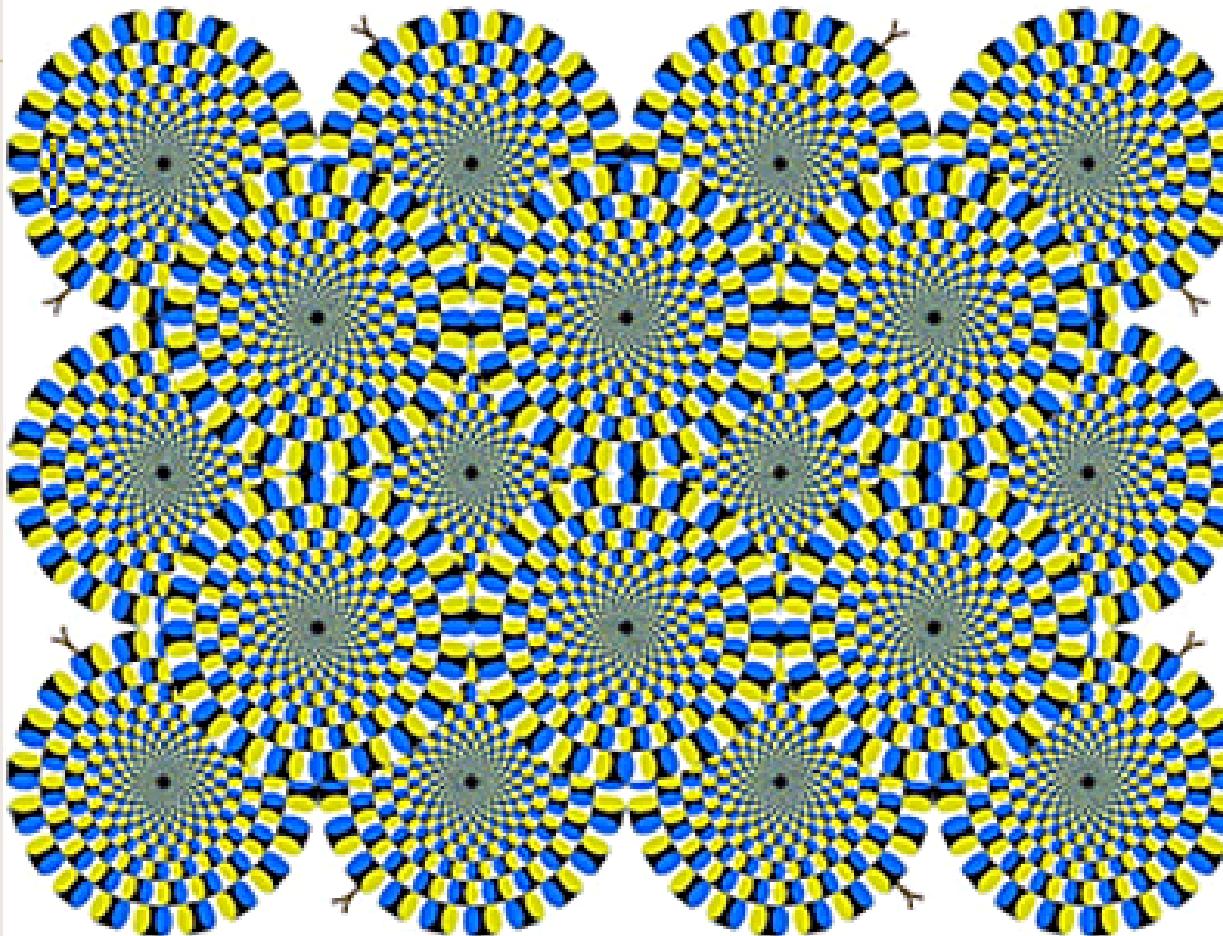
**FIGURE 2.9** Some well-known optical illusions.



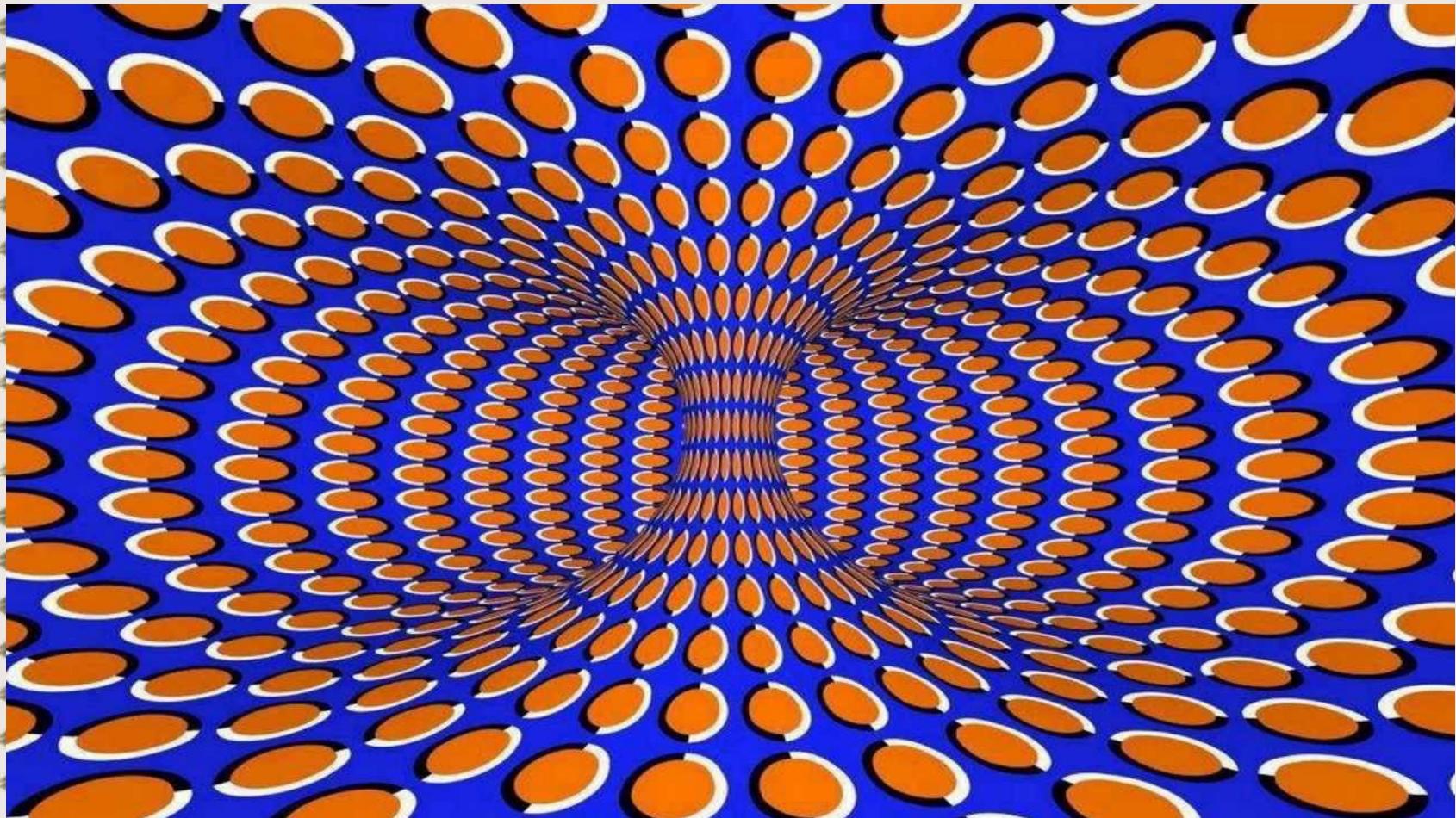


Download updated materials from Rejinpaul Network App

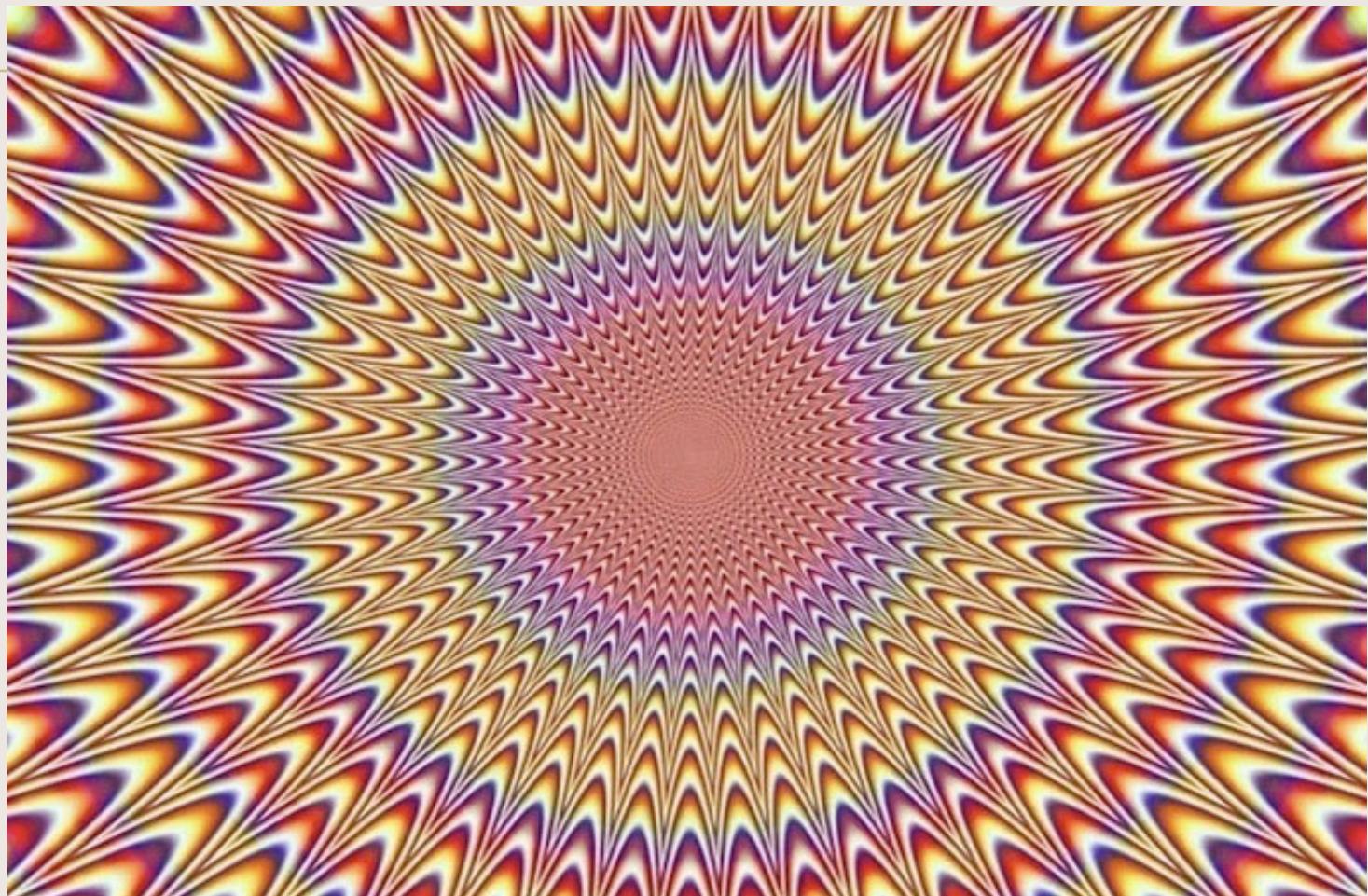
Are the circles moving in the following image?



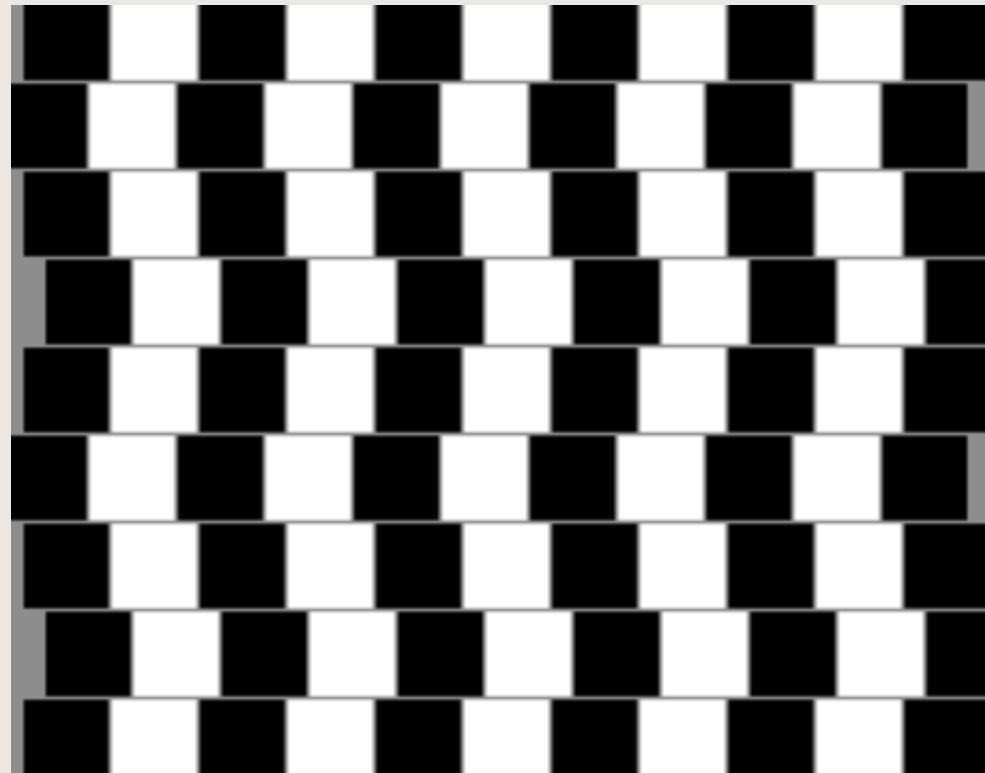
Answer: Look closely, they aren't moving.



Download updated materials from Rejinpaul Network App

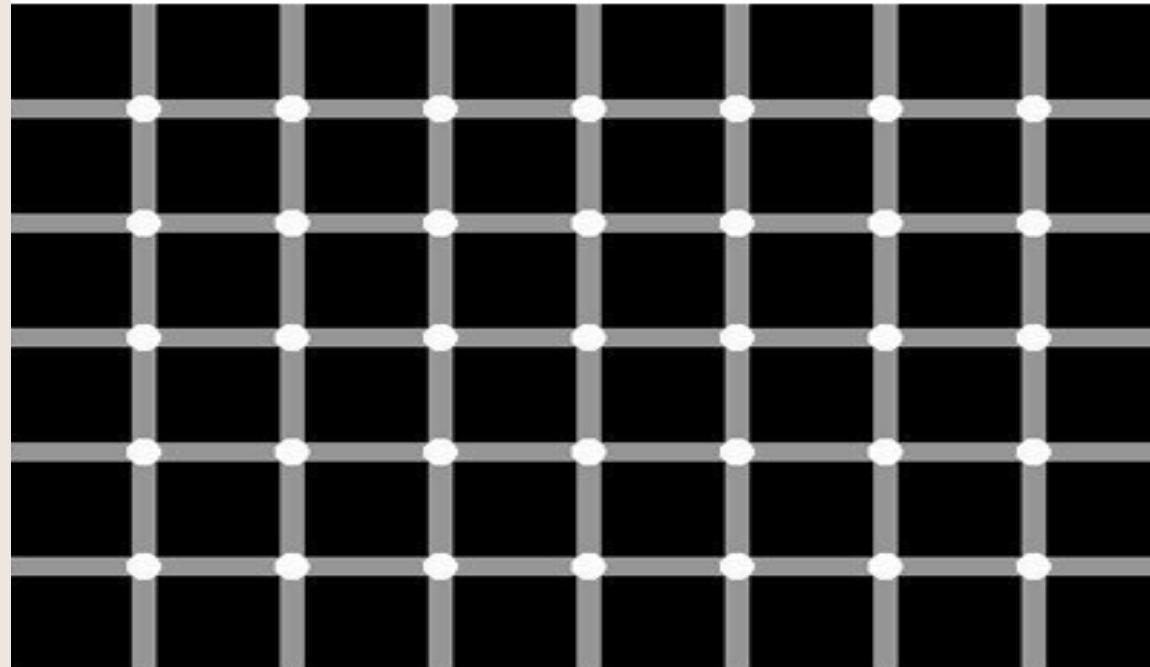


**Download updated materials from Rejinpaul Network App**



Download updated materials from Rejinpaul Network App

Count all the black dots you can see.



**Answer: There are no black dots**

**If you focus directly on each dot, you'll see that all of them are white.**

- 
- 
- Although the intensity of the stripes is constant, we actually perceive a brightness pattern that is strongly scalloped, especially near the boundaries
  - These scalloped bands are called mach bands

## RGB Model

- An **additive** colour model, based on three components: red(R), green (G) and Blue (B).
- 8 bits are usually used for each colour ( $\Rightarrow$  24 bits total)
- These can be stored (in memory) as 3 separate colour planes ....
- ... or in an ‘interlaced form’, with each pixel represented with 3 bytes (24bits) sequentially.

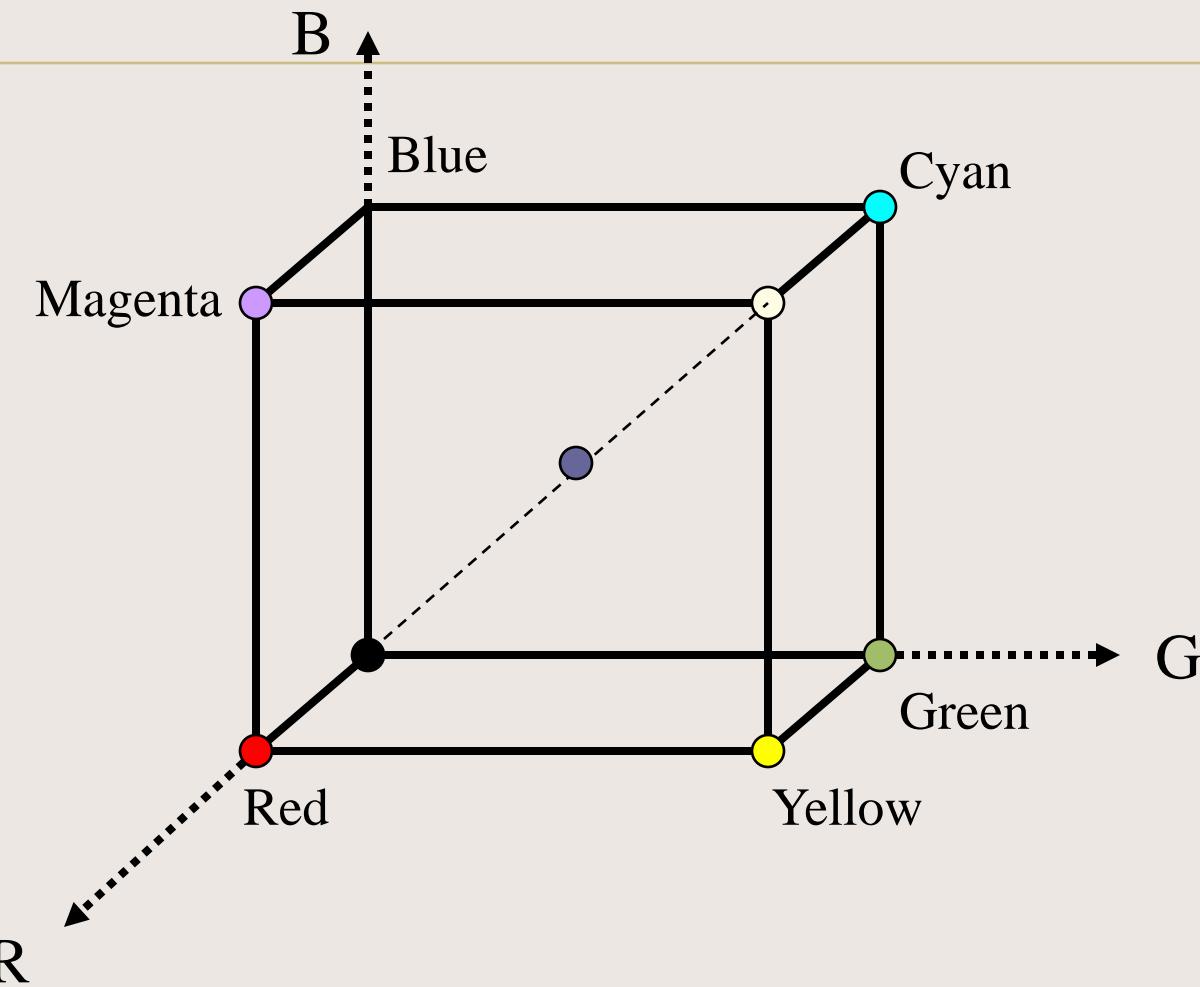
**But!!!!**

Not all colours can be represented by the RGB system

**However**

It is probably good enough to fool the human vision system

# The RGB Cube



# The HSI Model (an alternative to RGB)

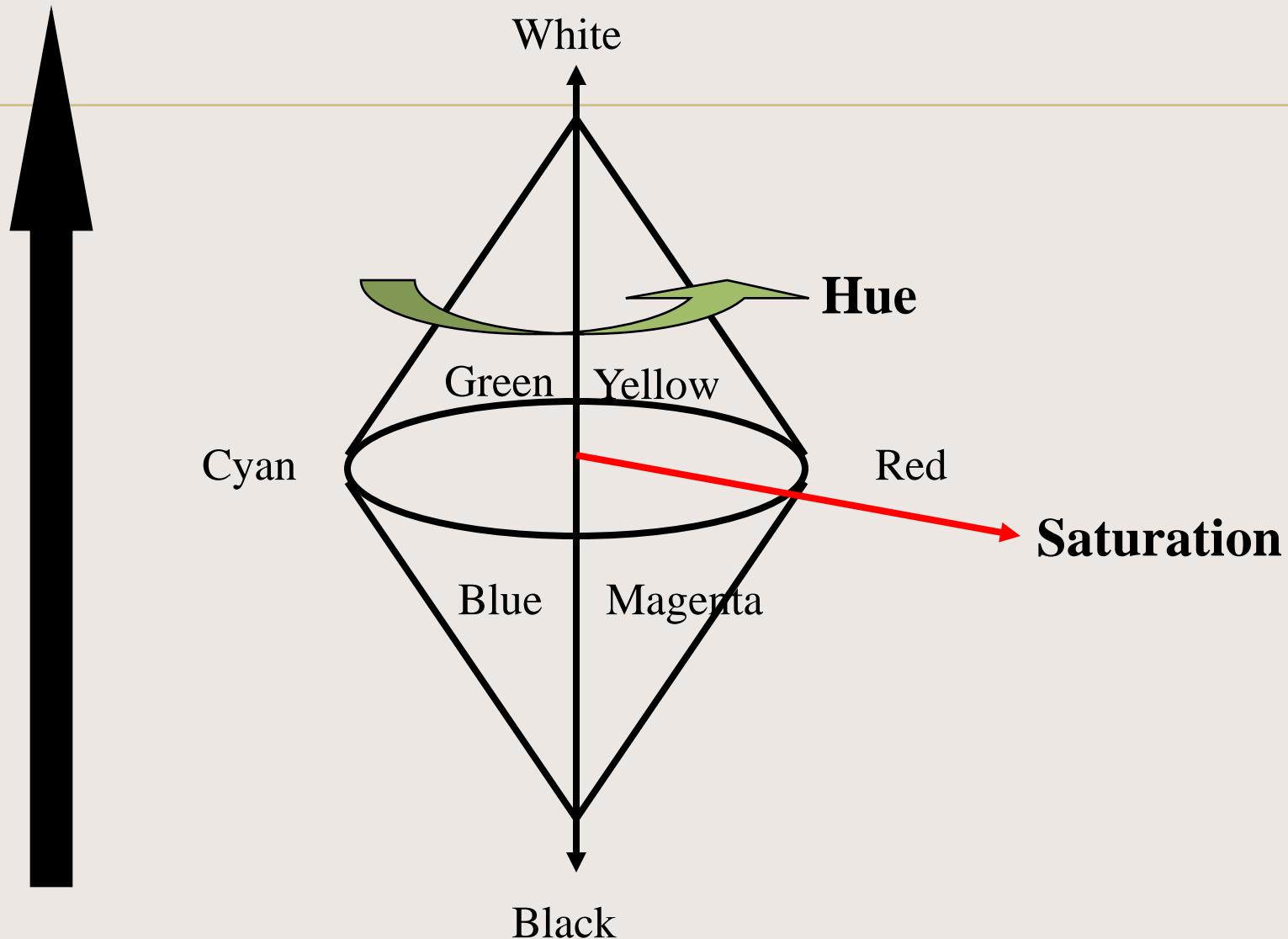
- **Hue**, H, specifies the dominant pure colour perceived by the observer (e.g. red, yellow, blue)  
[i.e. a representation of the frequency/wavelength]
- **Saturation**, S, specifies the degree to which a pure colour has been diluted by white light to produce observed colour.
- **Intensity**, I, is related to the perceived brightness of the colour.

N.B. Decoupling (separating) intensity from colour is very useful in image manipulation and analysis as increased/decreased lighting in a scene (more or less) only effect this parameter.

# The HSI Cone

[www.rejinpaul.com](http://www.rejinpaul.com)

Intensity



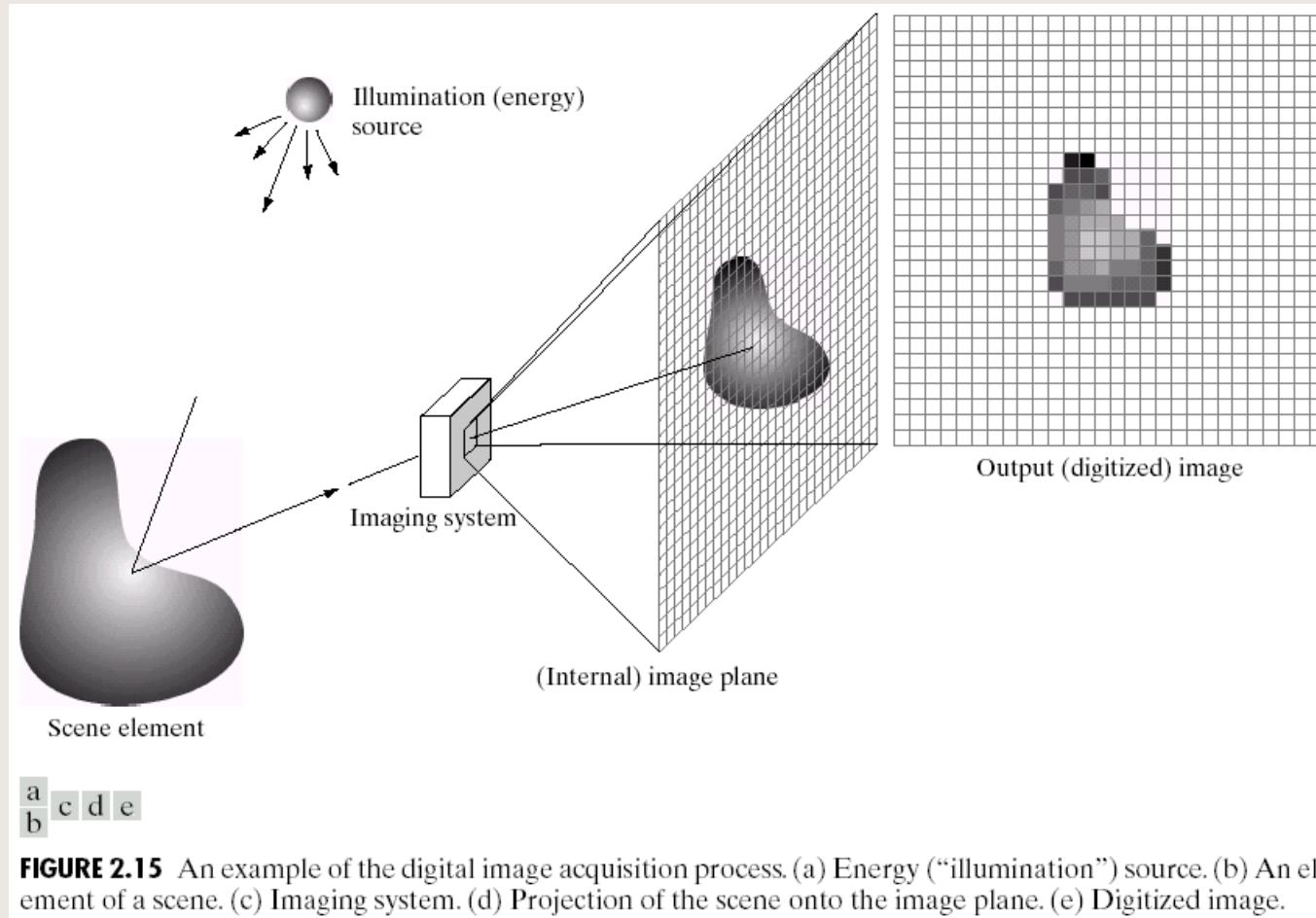
# A Simple Image Model

---

- Image: a 2-D light-intensity function  $f(x,y)$
- The value of  $f$  at  $(x,y) \rightarrow$  the intensity (brightness) of the image at that point
- $0 < f(x,y) < \infty$



## Digital Image Acquisition



# A Simple Image Model

---

- Nature of  $f(x,y)$ :
  - The amount of source light incident on the scene being viewed
  - The amount of light reflected by the objects in the scene

# A Simple Image Model

---

- Illumination & reflectance components:
  - Illumination:  $i(x,y)$
  - Reflectance:  $r(x,y)$
  - $f(x,y) = i(x,y) \cdot r(x,y)$
  - $0 < i(x,y) < \infty$   
and  $\leftarrow 0 < r(x,y) < 1 \rightarrow$   
(from total absorption to total reflectance)

# A Simple Image Model

---

- Sample values of  $r(x,y)$ :
  - 0.01: black velvet
  - 0.93: snow
- Sample values of  $i(x,y)$ :
  - 9000 foot-candles: sunny day
  - 1000 foot-candles: cloudy day
  - 0.01 foot-candles: full moon

# Create an image

---

To create a digital image, we need to convert the continuously sensed data into digital form.

This involves:

Sampling:

- Digitizing the coordinate values (resolution)

- Depends on density of sensor in an array

- Limited by optical resolution

Quantization (bits/pixel)

- Digitizing the amplitude values

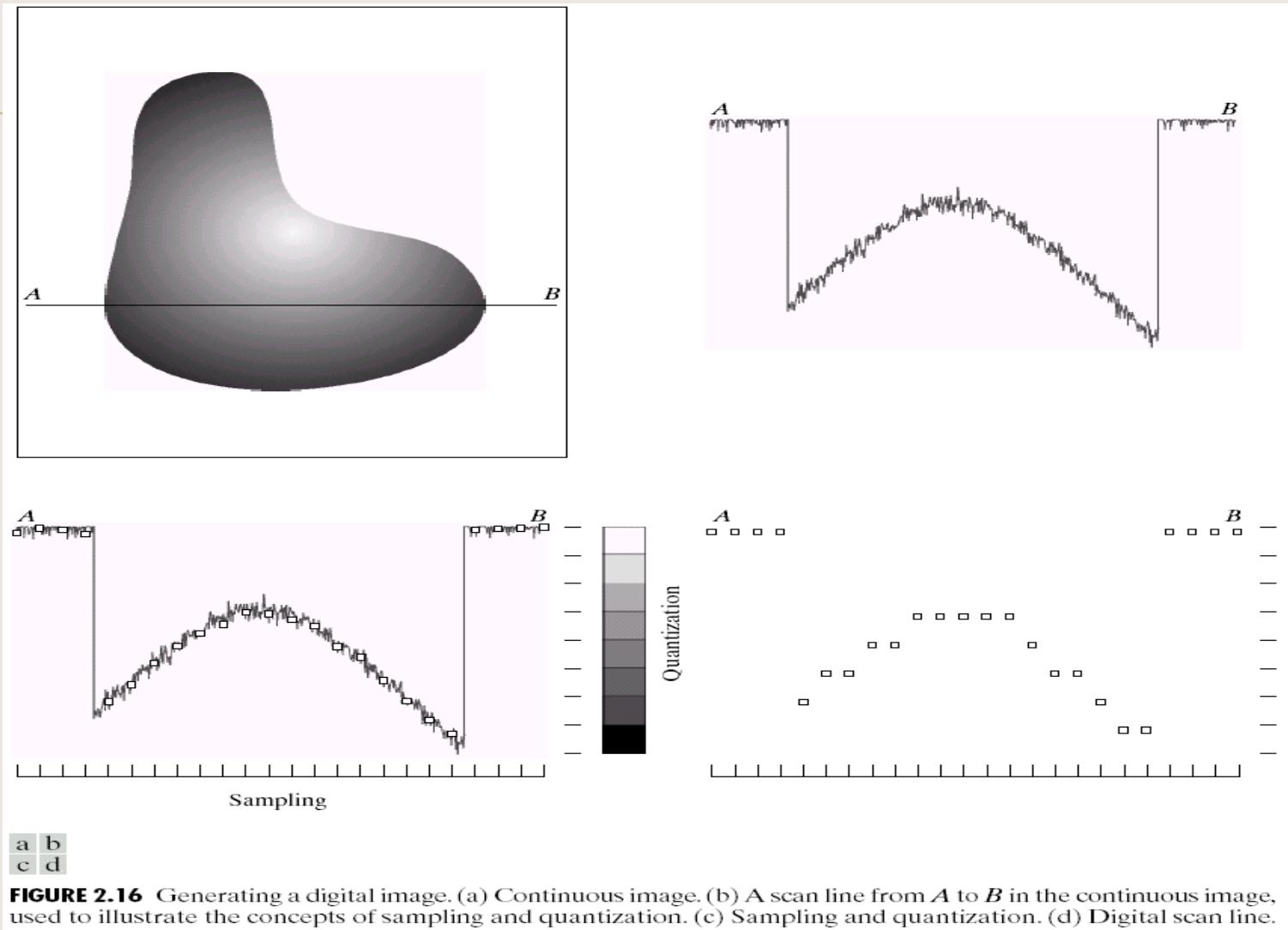
Pixel: short for picture element

# Sampling & Quantization

---

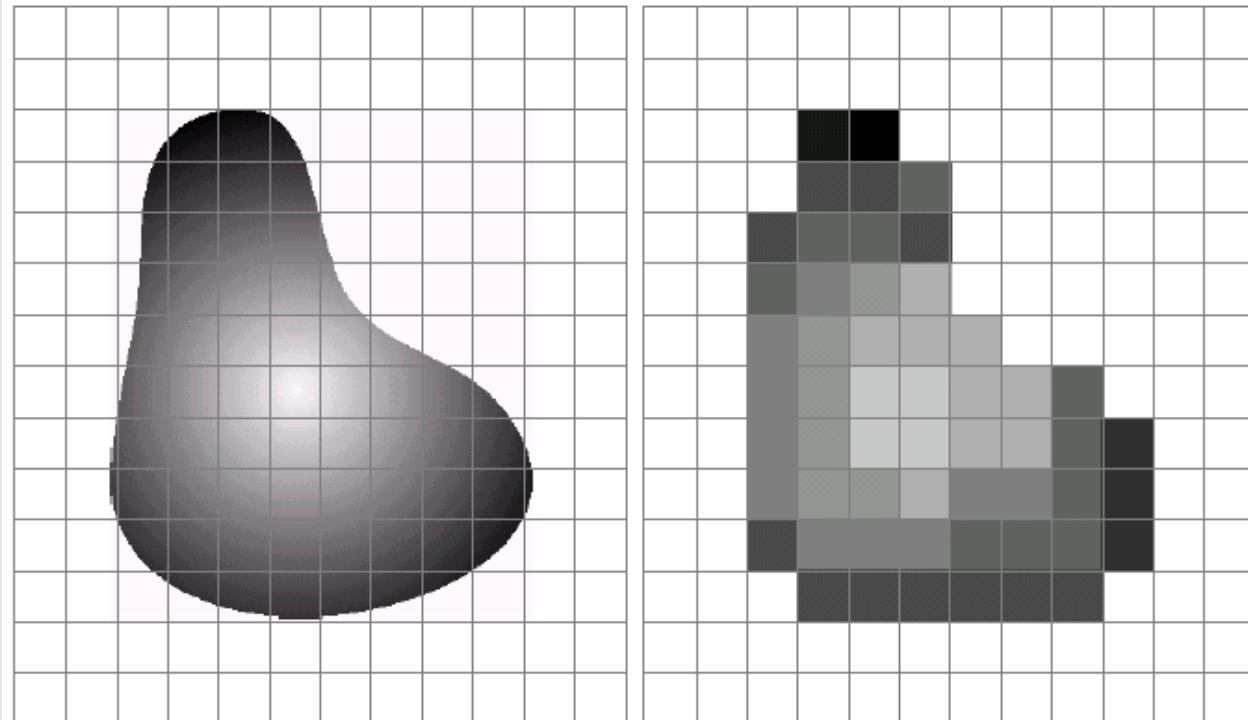
- The spatial and amplitude digitization of  $f(x,y)$  is called:
  - **image sampling** when it refers to spatial coordinates  $(x,y)$  and
  - **gray-level quantization** when it refers to the amplitude.

# Digital Image





## Sampling and Quantization

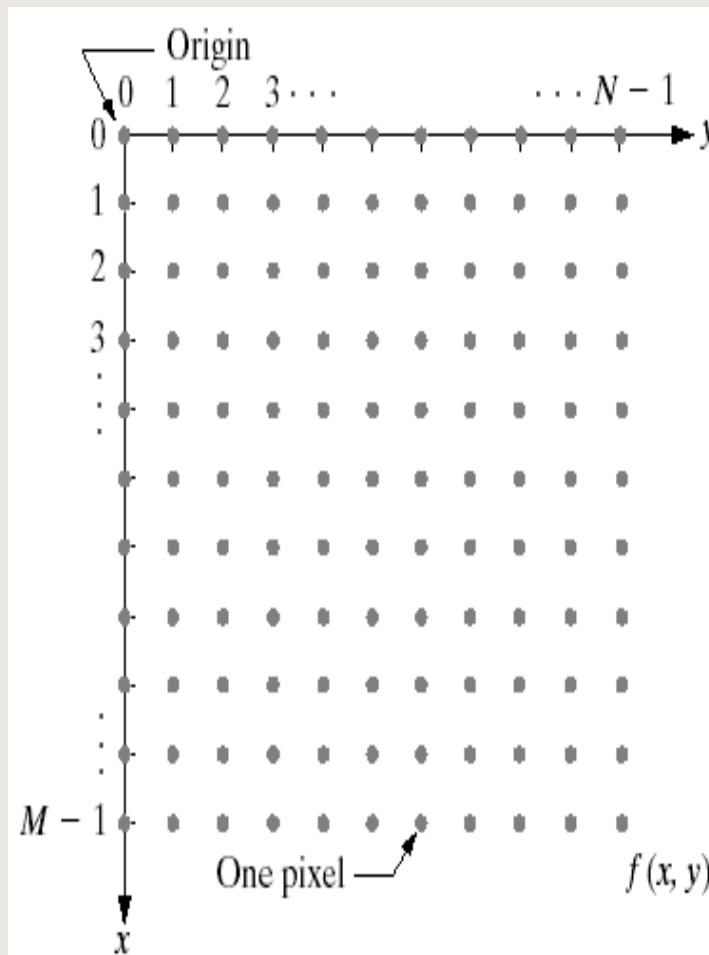


a b

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



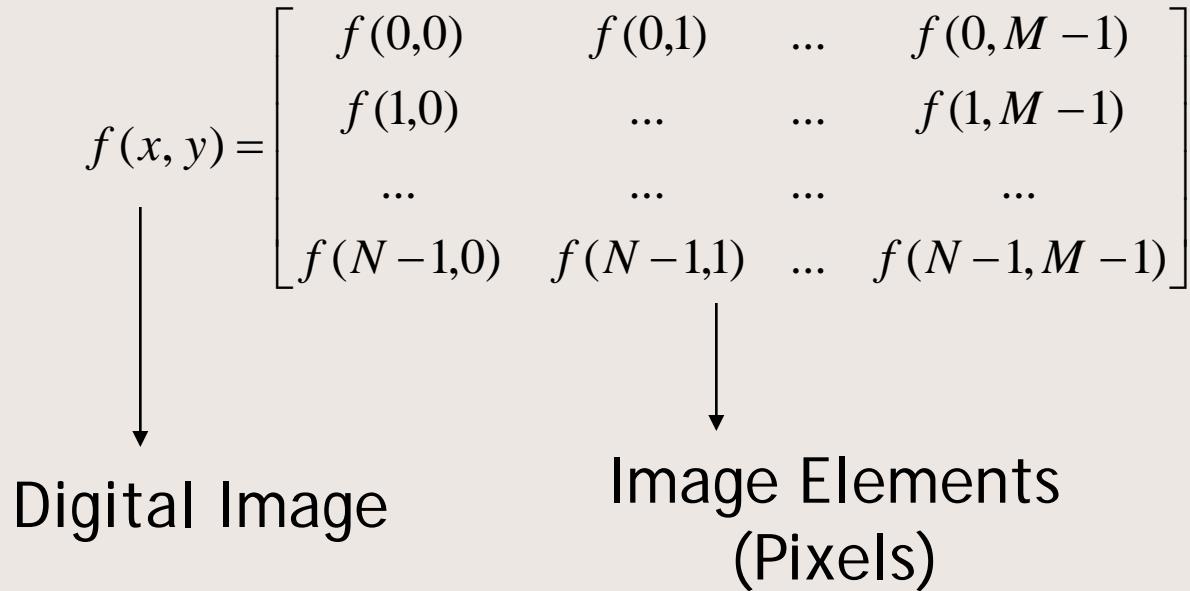
# A Digital Image



**FIGURE 2.18**

Coordinate convention used in this book to represent digital images.

# Sampling & Quantization



# Sampling & Quantization

---

- The digitization process requires decisions about:
  - values for  $N, M$  (where  $N \times M$ : the image array)
  - and
  - the **number** of discrete gray levels allowed for each pixel.

# Sampling & Quantization

---

- Usually, in DIP these quantities are integer powers of two:

$$N=2^n \quad M=2^m \quad \text{and} \quad G=2^k$$

number of gray levels

- Another assumption is that the discrete levels are equally spaced between 0 and  $L-1$  in the gray scale.

## Examples



1024



512



256



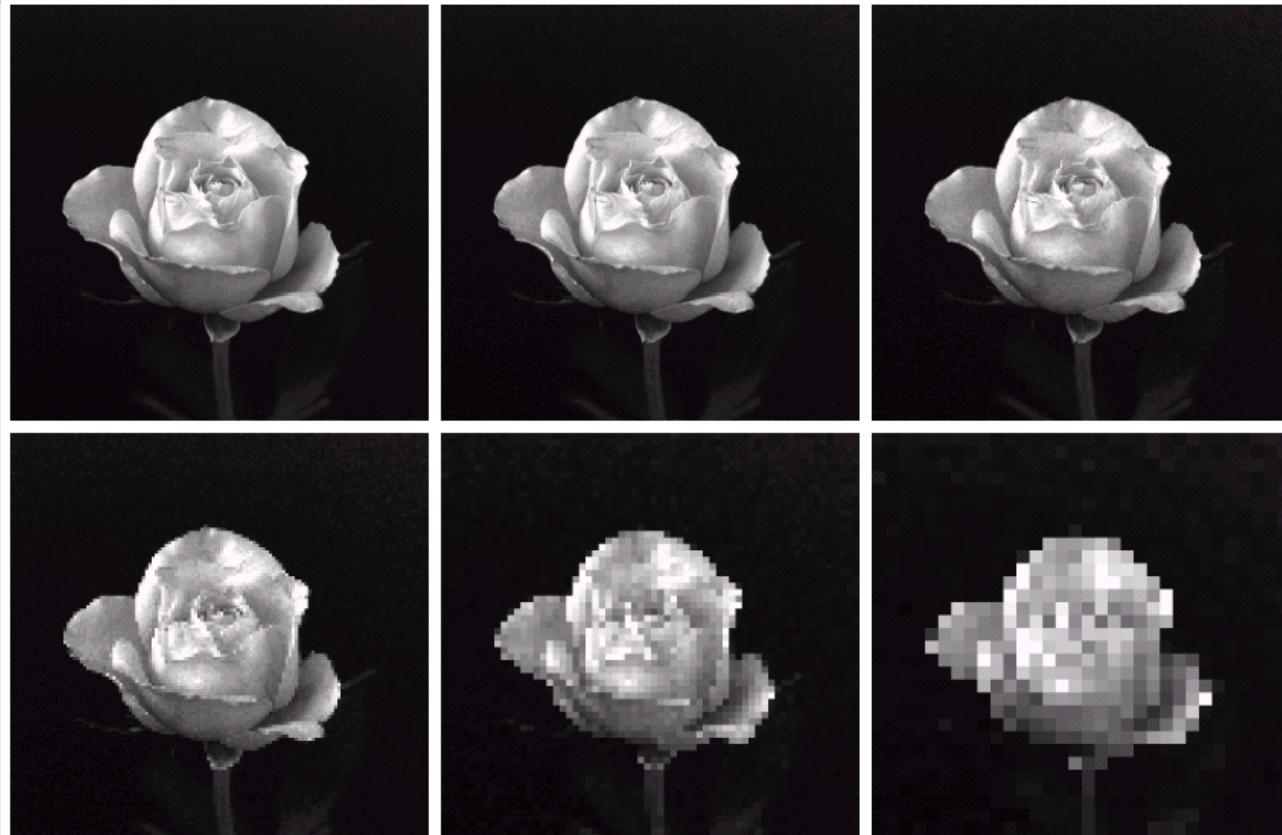
64



32

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

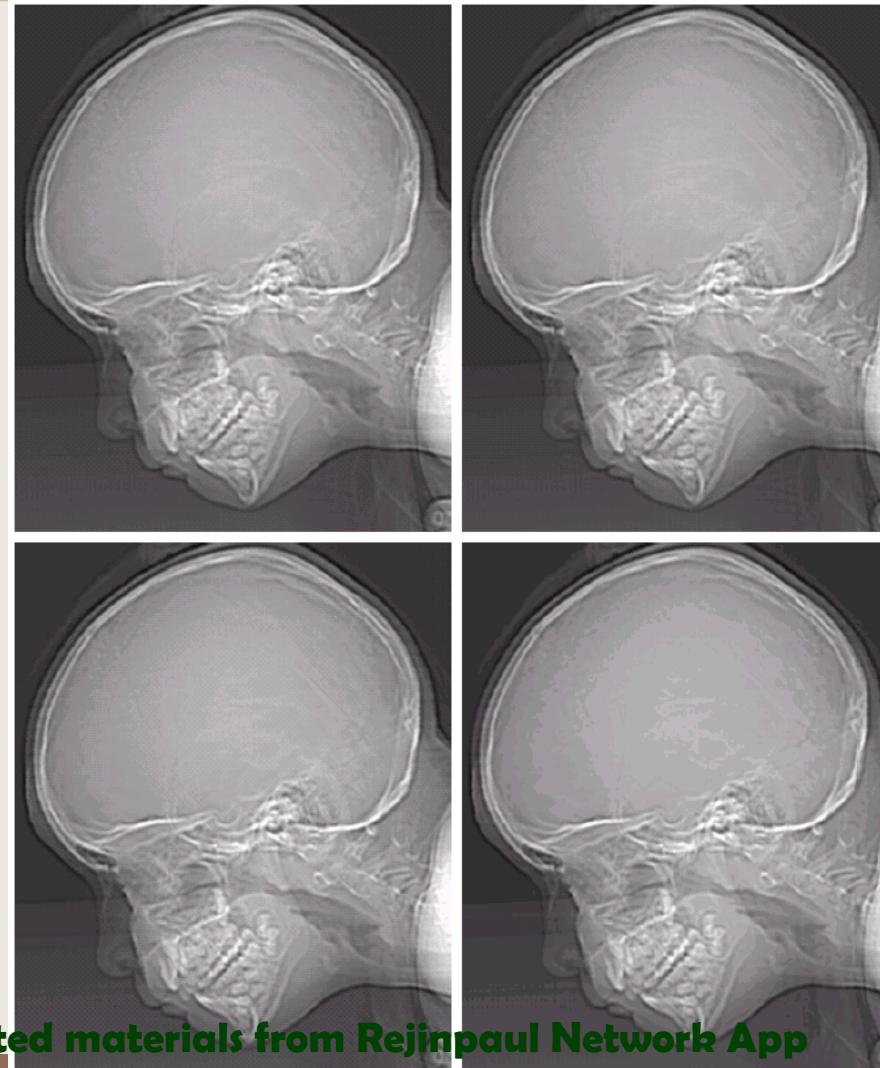
## Examples



a b c  
d e f

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

## Examples



a b  
c d

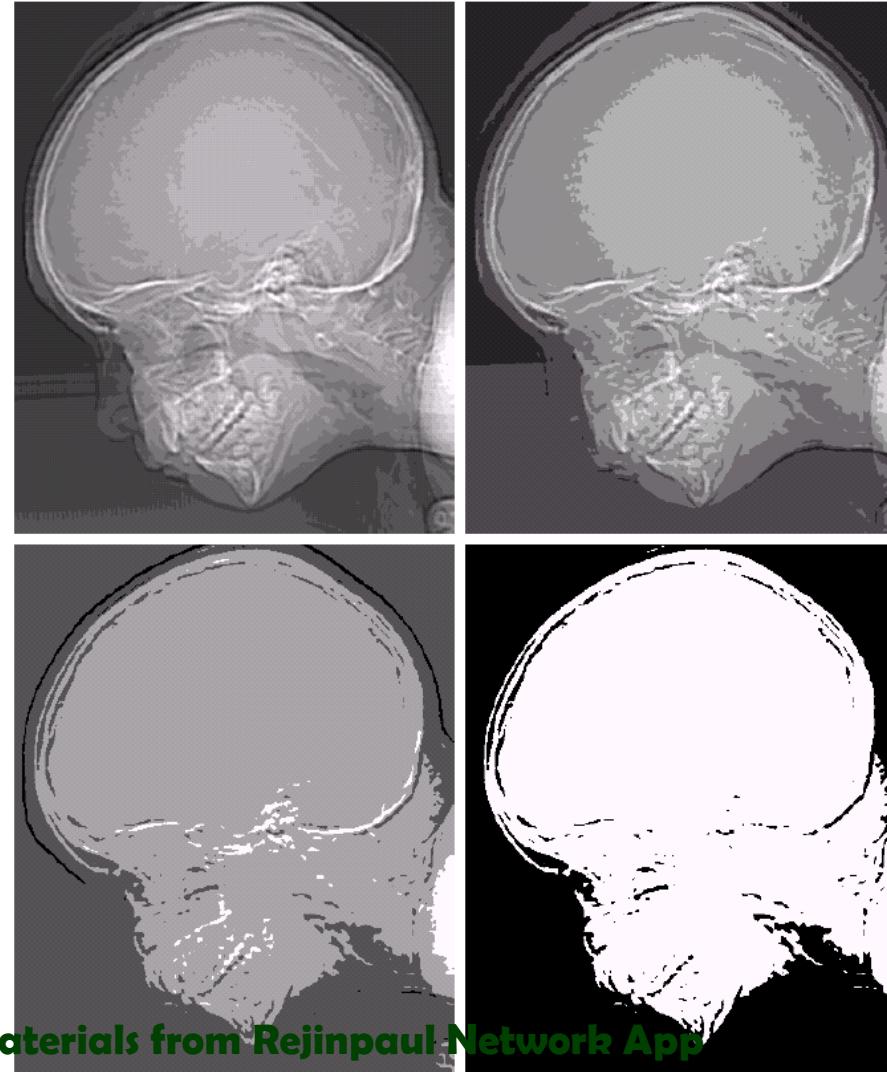
**FIGURE 2.21**  
(a) 452 × 374,  
256-level image.  
(b)–(d) Image  
displayed in 128,  
64, and 32 gray  
levels, while  
keeping the  
spatial resolution  
constant.



## Examples

e  
f  
g  
h

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



# Sampling & Quantization

---

- If  $b$  is the number of bits required to store a digitized image then:
  - $b = N \times M \times k$  (if  $M=N$ , then  $b=N^2k$ )

# Storage

**TABLE 2.1**

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

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

# Sampling & Quantization

---

- How many samples and gray levels are required for a good approximation?
  - Resolution (the degree of discernible detail) of an image depends on sample number and gray level number.
  - i.e. the more these parameters are increased, the closer the digitized array approximates the original image.

# Sampling & Quantization

---

- How many samples and gray levels are required for a good approximation? (cont.)
  - **But:** storage & processing requirements increase rapidly as a function of  $N$ ,  $M$ , and  $k$

# Sampling & Quantization

---

- Different versions (images) of the same object can be generated through:
  - Varying N, M numbers
  - Varying k (number of bits)
  - Varying both

# Sampling & Quantization

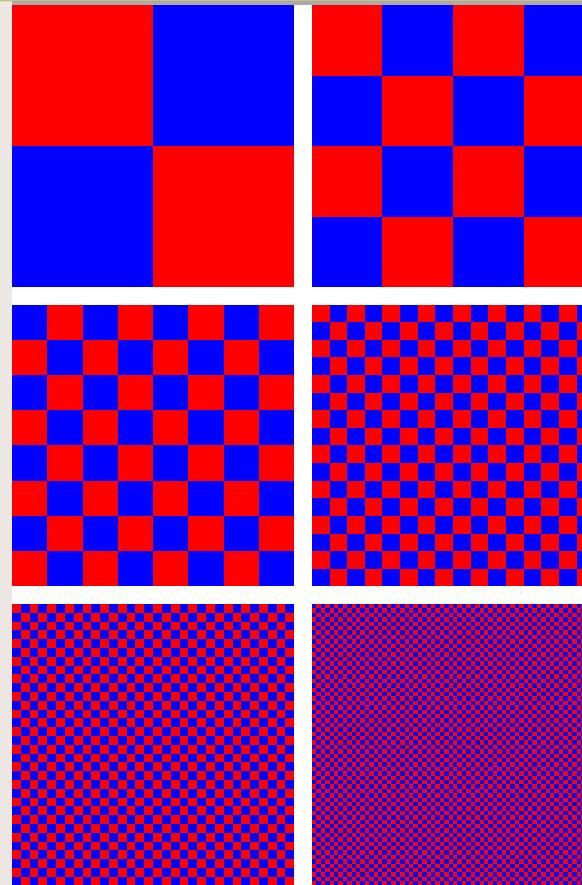
---

- Conclusions:
  - Quality of images increases as  $N$  &  $k$  increase
  - Sometimes, for fixed  $N$ , the quality improved by decreasing  $k$  (increased contrast)
  - For images with large amounts of detail, few gray levels are needed

# Dithering

---

- Full-color photographs may contain an almost infinite range of color values
- Dithering is the attempt by a computer program to approximate a color from a mixture of other colors when the required color is not available
- Dithering is the most common means of reducing the color range of images down to the 256 (or fewer) colors seen in 8-bit GIF images



---

Original full-color photograph

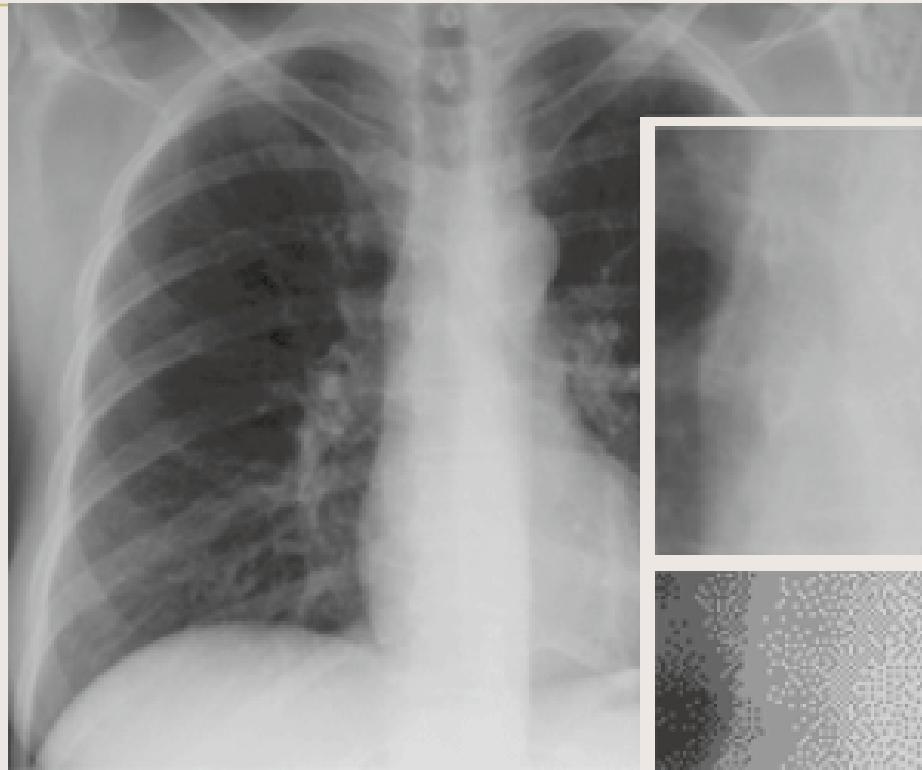


Dithered to 256 colors



- Most images are dithered in a diffusion or randomized pattern to diminish the harsh transition from one color to another
- But dithering also reduces the overall sharpness of an image, and it often introduces a noticeable grainy pattern in the image
- This loss of image detail is especially apparent when full-color photos are dithered down to the 216-color browser-safe palette.

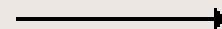
Original full-tone image



Detail of original image



Dithered image shows loss of tone and loss of image detail



# **TWO -DIMENSIONAL MATHEMATICAL PRELIMINARIES**

## Definitions (Con't)

---

A *column vector* is an  $m \times 1$  matrix:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

A *row vector* is a  $1 \times n$  matrix:

$$\mathbf{b} = [b_1, b_2, \dots, b_n]$$

A column vector can be expressed as a row vector by using the transpose:

$$\mathbf{a}^T = [a_1, a_2, \dots, a_m]$$



## Some Basic Matrix Operations

---

- The **sum** of two matrices  $\mathbf{A}$  and  $\mathbf{B}$  (of equal dimension), denoted  $\mathbf{A} + \mathbf{B}$ , is the matrix with elements  $a_{ij} + b_{ij}$ .
- The **difference** of two matrices,  $\mathbf{A} - \mathbf{B}$ , has elements  $a_{ij} - b_{ij}$ .
- The **product**,  $\mathbf{AB}$ , of  $m \times n$  matrix  $\mathbf{A}$  and  $p \times q$  matrix  $\mathbf{B}$ , is an  $m \times q$  matrix  $\mathbf{C}$  whose  $(i,j)$ -th element is formed by multiplying the entries across the  $i$ th row of  $\mathbf{A}$  times the entries down the  $j$ th column of  $\mathbf{B}$ ; that is,

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{pj}$$



## Some Basic Matrix Operations (Con't)

The *inner product* (also called *dot product*) of two vectors

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

is defined as

$$\begin{aligned}\mathbf{a}^T \mathbf{b} &= \mathbf{b}^T \mathbf{a} = a_1 b_1 + a_2 b_2 + \cdots + a_m b_m \\ &= \sum_{i=1}^m a_i b_i.\end{aligned}$$

Note that the inner product is a scalar.

---

## TOEPLITZ MATRIX, CIRCULANT MATRIX

---

- It has a constant elements along the main diagonal and the sub diagonals
- Circulant matrix – each of its rows (or columns) is a circular shift of the previous row(or column)

# ORTHOGONAL AND UNITARY MATRICES

---

- An orthogonal matrix is such that its inverse is equal to its transpose

$$A^{-1} = A^T$$

- A Matrix is called unitary if its inverse is equal to its conjugate transpose

$$A^{-1} = A^{*T}$$

# BLOCK MATRIX

---

- Any matrix whose elements are matrices themselves is called a block matrix.



# **TWO DIMENSIONAL DISCRETE FOURIER TRANSFORM ( 2D DFT)**

## **2-D DFT:**

- The discrete fourier transform of a function (image)  $f(x,y)$  of size  $M \times N$  is given by,

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

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

- Similarly, given  $F(u,v)$ , we obtain  $f(x,y)$  by the inverse fourier transform, given by,

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

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

- U and V are the transform or frequency variables and X and Y are the spatial or image variables.
-

$$v(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} u(m, n) W_N^{km} W_N^{ln},$$

$$0 \leq k, l \leq N - 1$$

inverse transform is

$$u(m, n) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} v(k, l) W_N^{-km} W_N^{-ln},$$

# Properties of the Two-Dimensional DFT

DFT AND UNITARY  
DFT ARE SYMMETRIC

$$\mathcal{F}^T = \mathcal{F}, \quad \mathcal{F}^{-1} = \mathcal{F}^*$$

PERIODIC

$$v(k + N, l + N) = v(k, l),$$

$$N^2 \log_2 N$$

FAST ALGORITHM

$$v(k, l) = v^*(N - k, N - l),$$

CONJUGATE  
SYMMETRIC ABOUT  
N/2

- CIRCULAR CONVOLUTION THEOREM
- The DFT of the circular convolution of the two sequences is equal to the product of their DFTs.



# Discrete Cosine Transform

# Introduction

---

- The discrete cosine transform (DCT) is a technique for converting a signal into elementary frequency components. It is widely used in image compression.
- Developed by Ahmed, Natarajan, and Rao [1974], the DCT is a close relative of the discrete Fourier transform (DFT). Its application to image compression was pioneered by Chen and Pratt [1984].

# Definitions

- The N X N Cosine Transform matrix  $C=\{c(k,n)\}$ ,also called as Discrete Cosine Transform (DCT) ,is defined as

$$c(k,n)=1/\sqrt{N} , \quad k=0, 0 \leq n \leq N-1$$

$$= \sqrt{2/N} \cos \pi(2n+1)k/2N , \quad 1 \leq k \leq N-1, 0 \leq n \leq N-1$$

The One Dimensional DCT of a sequence  $\{u(n), 0 \leq n \leq N-1\}$  is defined as

$$v(k)=\alpha(k) \sum_{n=0}^{N-1} u(n) \cos \pi(2n+1)k/2N , \quad 0 \leq k \leq N-1$$

$$\text{where } \alpha(0) \approx 1/\sqrt{N} , \alpha(k) = \sqrt{2/N} \text{ for } 1 \leq k \leq N-1$$

The 2-Dimensional DCT pair is given by

$$C(u,v)=\alpha(u) \alpha(v) \sum_{n=0}^{N-1} f(n,y) \cos[u \pi(2x+1)/2N] \cos[v \pi(2y+1)/2N] , \quad 0 \leq u,v \leq N-1$$

# Transformation

---

- An  $N \times N$  image is divided into a set of smaller  $m \times m$  sub images
- The DCT is computed for each images
- Within each sub image ,only the DCT components that are non-negligible are retained, providing a means of image compression. All other components are assumed to be zero
- The remaining non-negligible components are used to reconstruct the image

# Here is the original image and its discrete cosine transform



Particularly note the concentration of large DCT coefficients in the low-frequency zone. The DCT is known to have excellent energy compaction properties.

# Advantages

---

1. The DCT has the added advantage of mirror symmetry, making it suitable for block encoding of an image  
ie; processing an image via small overlapping sub images
2. Because of the symmetry property of the DCT, it produces less degradation at each of the sub image boundaries than the DFT.
3. It supports Data compression

# Properties of DCT

1. The Cosine Transform is real and orthogonal  
ie;  $C = C^*$   $\rightarrow C^{-1} = C^T$
2. The Cosine Transform of a sequence is related to the DFT of its symmetric extension
3. The Cosine Transform is a fast transform.  
The Cosine Transform of a vector of N elements can be calculated in  $N \log_2 N$  operations via an N-point FFT.
4. The Cosine Transform has excellent energy compaction for highly correlated data
5. The Basic Vectors of the DCT are the eigen vectors of the symmetric tridiagonal matrix
6. The NxN DCT is very close to first order KL Transform.

# Applications

---

- JPEG
- MJPEG
- MPEG
- SOLVING PARTIAL DIFFERENTIAL FAMILY OF EQUATIONS
- CCITT H.261 (also known as Px64), for compression of videotelephony and teleconferencing.



# **KARHUNEN AND LOEVE (KL) TRANSFORM and HOTELLING TRANSFORM**

- **Karhunen–Loève theorem** (named after Kari Karhunen and Michel Loèvre)
- Representation of a stochastic process as an infinite linear combination of orthogonal functions
- In contrast to a Fourier series where the coefficients are real numbers and the expansion basis consists of sinusoidal functions (that is, sine and cosine functions)
- The coefficients in the Karhunen–Loève theorem are random variables and the expansion basis depends on the process.

## 2-D KL Transform of Images

For a zero mean 2-D random process  $\{x(m, n)\}$ ,  $m, n \in [0, N - 1]$  using the same procedure adopted for DFT and DCT, the 2-D KL transform of image matrix  $x$  is

$$X = \Psi_1^{*t} X \Psi_2^*$$

where  $\Psi_1^{*t}$  and  $\Psi_2^*$  are 1-D KL matrices applied to columns and rows of the image, respectively. The inverse KT transform is

$$x = \Psi_1 X \Psi_2^t$$

- For a real  $M \times N$  image  $U$ , the basis vectors of the KL transform are given by the orthonormalized eigenvectors of its autocorrelation matrix  $R_U$ .
- 

- $U_{M \times N}$ : Image matrix
- $u_n$ :  $n$  th column vector of the image.
- $V_{M \times N}$ : Transformed image matrix.
- $v_n$ :  $n$  th column vector of the transformed image.
- $R_u$ : Autocorrelation matrix of the image.
- $R_v$ : Autocorrelation matrix of the transformed image.

The KL transform of U is defined as,

$$V_{M \times N} = \Phi^T U_{M \times N}$$

And for the inverse transform, matrix  $\Phi$  , the KL transform matrix is used.

- Here the matrix  $\Phi$  consists of eigen vectors of the matrix  $R_u$  which is autocorrelation matrix of the image.
- As known from matrix theory, the matrix  $R_v$ , autocorrelation matrix of the transformed image is a diagonal matrix that consists of eigen values of the autocorrelation matrix  $R_u$ .
- That means transformed image is definitely uncorrelated. In other word, the transformed image vectors are orthogonal, perpendicular as each other.

$$R_v = \mathcal{F} \cdot R_u \cdot \mathcal{F}$$

---

- There is no a particular KL transform matrix.
- Every image has its own KL transform matrix described above.

# KL transform example

small image matrix

$$U = \begin{bmatrix} 2 & 4 & 5 & 5 & 3 & 2 \\ 2 & 3 & 4 & 5 & 4 & 3 \end{bmatrix}$$

Its vertical mean vector is,

$$\bar{m} = (1/N) \sum_{i=0}^{N-1} u_i$$

$$\bar{m} = (1/6) \begin{bmatrix} 2 + 4 + 5 + 5 + 3 + 2 \\ 2 + 3 + 4 + 5 + 4 + 3 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 3.5 \end{bmatrix}$$

its covariance matrix

$$R_u = \frac{1}{(N-1)} \sum_{i=0}^{N-1} (u_i - \bar{m})(u_i - \bar{m})^T$$

$$R_u = \frac{1}{5} \left\{ \begin{bmatrix} 2.25 & 2.25 \\ 2.25 & 2.25 \end{bmatrix} + \begin{bmatrix} 0.25 & -0.25 \\ -0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 2.25 & 0.75 \\ 0.75 & 0.25 \end{bmatrix} + \begin{bmatrix} 2.25 & 2.25 \\ 2.25 & 2.25 \end{bmatrix} + \begin{bmatrix} 0.25 & -0.25 \\ -0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 2.25 & 0.75 \\ 0.75 & 0.25 \end{bmatrix} \right\}$$

$$R_u = \begin{bmatrix} 1.9 & 1.1 \\ 1.1 & 1.1 \end{bmatrix}$$

$$R_v = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Now, to form  $Rv$  as, orthogonal set of the eigen vectors for the  $Ru$  must be calculated.

$$\begin{vmatrix} 1.9 - \lambda & 1.1 \\ 1.1 & 1.1 - \lambda \end{vmatrix} = 0$$

$$\lambda^2 - 3\lambda + 0.88 = 0$$

$$\lambda_1 = 2.67$$

$$\lambda_2 = 0.33$$

$$[R_u - \lambda I] \phi = 0$$

$$\phi = \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} = \left\{ \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} \middle| \begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix} \right\}$$

Equations for the eigen vector  $\Phi_1$

$$\begin{aligned} -0.77 \phi_{11} + 1.1 \phi_{21} &= 0 \\ 1.1 \phi_{11} - 1.57 \phi_{21} &= 0 \end{aligned}$$

$$\phi_{11} = 1.43 \phi_{21}$$

---

## Equations for the eigen vector $\Phi_2$

$$1.57 \phi_2' + 1.1 \phi_2 = 0$$

$$1.1 \phi_2' + 0.77 \phi_2 = 0$$

$$\phi_2' = -0.7 \phi_2$$

In addition to Eq1 and Eq2, if  $\Phi_1$  and  $\Phi_2$  are orthogonal eigen vectors, following equations can be written,

$$\phi_1'^2 + \phi_1^2 = 1$$

$$\phi_2'^2 + \phi_2^2 = 1$$

# The solution of these four equations is,

$$\rho = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix} = \left\{ \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \right\} = \left\{ \begin{bmatrix} 0.82 & -0.57 \\ 0.57 & 0.82 \end{bmatrix} \right\}$$

Now, let's transform the image U,

$$V_n = \rho^T u_n$$

$$V = \begin{bmatrix} 0.82 & 0.57 \\ -0.57 & 0.82 \end{bmatrix} \begin{bmatrix} 2 & 4 & 5 & 5 & 3 & 2 \\ 2 & 3 & 4 & 5 & 4 & 3 \end{bmatrix}$$

$$V = \begin{bmatrix} 2.78 & 4.99 & 6.38 & 6.95 & 4.74 & 3.35 \\ 0.5 & 0.18 & 0.43 & 1.25 & 1.57 & 1.32 \end{bmatrix}$$

$$U' = \begin{bmatrix} 0.82 & -0.57 \\ 0.57 & 0.82 \end{bmatrix} \begin{bmatrix} 2.78 & 4.99 & 6.38 & 6.95 & 4.74 & 3.35 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$U' = \begin{bmatrix} 2.28 & 4.1 & 5.23 & 5.7 & 3.89 & 2.75 \\ 1.58 & 2.84 & 3.64 & 3.96 & 2.7 & 1.91 \end{bmatrix}$$

As seen above, inverse transform of the transformed image is very similar to original image that **means mean squared error reduced minimum.**

# Properties:

---

- The transformed image are uncorrelated
- orthogonal
- Perpendicular
- Optimality and Data Reduction
- Distribution of Variance

Among all unitary transforms KL packs the maximum average energy into  $m \times N$  elements of  $X$ .



# Singular Value Decomposition

# Singular Value Decomposition (SVD)

- Handy mathematical technique that has application to many problems
- Given any  $m \times n$  matrix  $\mathbf{A}$ , algorithm to find matrices  $\mathbf{U}$ ,  $\mathbf{V}$ , and  $\mathbf{W}$  such that

$$\mathbf{A} = \mathbf{U} \mathbf{W} \mathbf{V}^T$$

$\mathbf{U}$  is  $m \times n$  and orthonormal

$\mathbf{W}$  is  $n \times n$  and diagonal

$\mathbf{V}$  is  $n \times n$  and orthonormal

# SVD

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_n \end{pmatrix} \mathbf{V}^T$$

- Treat as black box: code widely available  
In Matlab: `[U,w,V]=svd(A,0)`

# SVD

---

- The  $w_i$  are called the singular values of  $\mathbf{A}$
- If  $\mathbf{A}$  is singular, some of the  $w_i$  will be 0
- In general  $\text{rank}(\mathbf{A}) = \text{number of nonzero } w_i$
- SVD is mostly unique (up to permutation of singular values, or if some  $w_i$  are equal)

# SVD and Inverses

---

- Why is SVD so useful?
- Application #1: inverses
- $\mathbf{A}^{-1} = (\mathbf{V}^T)^{-1} \mathbf{W}^{-1} \mathbf{U}^{-1} = \mathbf{V} \mathbf{W}^{-1} \mathbf{U}^T$ 
  - Using fact that inverse = transpose for orthogonal matrices
  - Since  $\mathbf{W}$  is diagonal,  $\mathbf{W}^{-1}$  also diagonal with reciprocals of entries of  $\mathbf{W}$

# UNIT 2

# IMAGE ENHANCEMENT

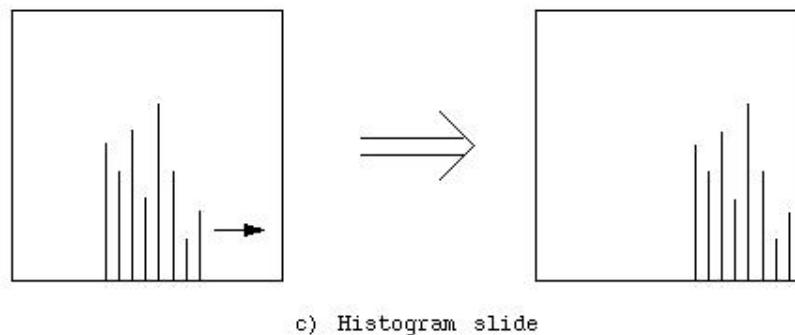
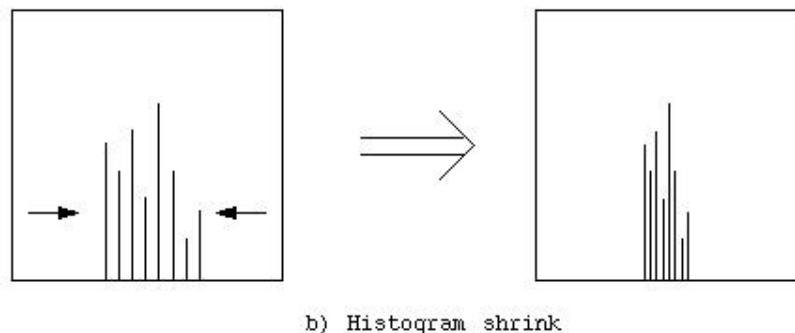
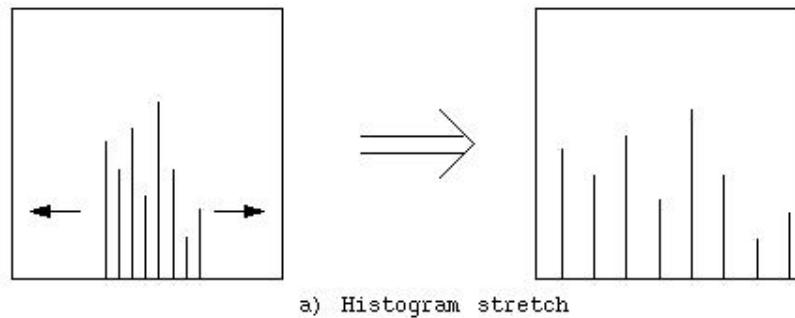
# ➤ Histogram Modification

- ✓ Histogram modification performs a function similar to gray level mapping, but works by considering histogram's shape and spread
- ✓ ***Gray level histogram*** of an image is the distribution of the gray levels in an image
- ✓ Examination of the histogram is one of the most useful tools for image enhancement, as it makes easy to see the modifications that may improve an image

- ✓ The histogram can be modified by a mapping function, which will **stretch**, **shrink** (compress), or **slide** the histogram
- ✓ Histogram stretching and histogram shrinking are forms of gray scale modification, sometimes referred to as ***histogram scaling***

Figure 8.2-8: Histogram Modification

[www.rejinpaul.com](http://www.rejinpaul.com)



## ✓ **Histogram stretch**

- The mapping function equation is as follows:

$$\text{Stretch}(I(r,c)) = \left[ \frac{I(r,c) - I(r,c)_{\text{MIN}}}{I(r,c)_{\text{MAX}} - I(r,c)_{\text{MIN}}} \right] [\text{MAX} - \text{MIN}] + \text{MIN}$$

w in

the image  $I(r,c)$ ,  $I(r,c)_{\text{MIN}}$  is the smallest gray level value in  $I(r,c)$  and

MAX and MIN correspond to the maximum and minimum gray level values possible (for an 8-bit image these are 0 and 255)

- This equation will take an image and stretch the histogram across the entire gray level range, which has the effect of increasing the contrast of a low contrast image
- If most of the pixel values in an image fall within a small range, it is useful to allow a small percentage of the pixel values to be clipped at the low and high end of the range (for an 8-bit image this means truncating at 0 and 255)

# Histogram Stretching

[www.rejinpaul.com](http://www.rejinpaul.com)



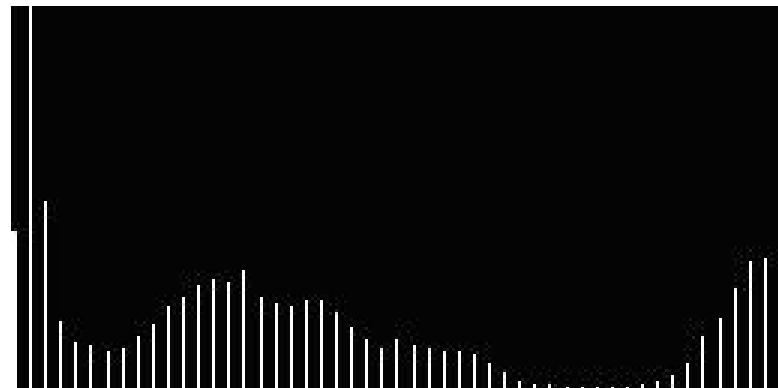
a) Low-contrast image



b) Histogram of image (a)



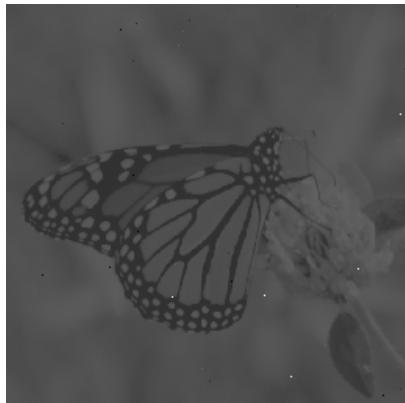
c) Image (a) after histogram stretch



d) Histogram of image after stretch

**Download updated materials from Rejinpaul Network App**

# Histogram Stretching with Clipping



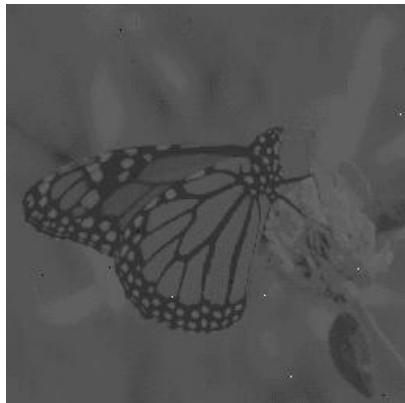
a) Original image



b) Histogram of original image



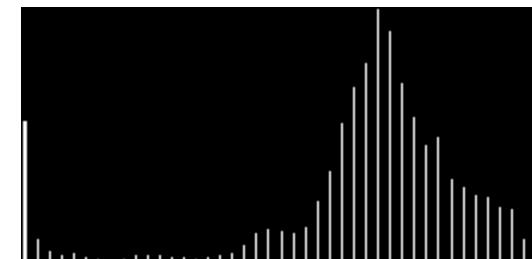
c) Image after histogram stretching with out clipping



d) Histogram of image (c)



e) Image after histogram stretching with clipping 1% of the values at the high and low ends



f) Histogram of image (e)

✓ **Histogram shrink**

- ✓ The mapping function equation is as follows:

$$\text{Shrink}(I(r,c)) = \left[ \frac{\text{Shrink}_{\text{MAX}} - \text{Shrink}_{\text{MIN}}}{I(r,c)_{\text{MAX}} - I(r,c)_{\text{MIN}}} \right] [ I(r,c) - I(r,c)_{\text{MIN}} ] + \text{Shrink}_{\text{MIN}}$$

the image  $I(r,c)$ ,  $I(r,c)_{\text{MIN}}$  is the smallest gray level value in  $I(r,c)$  and

$\text{Shrink}_{\text{MAX}}$  and  $\text{Shrink}_{\text{MIN}}$  correspond to the maximum and minimum desired in the compressed histogram

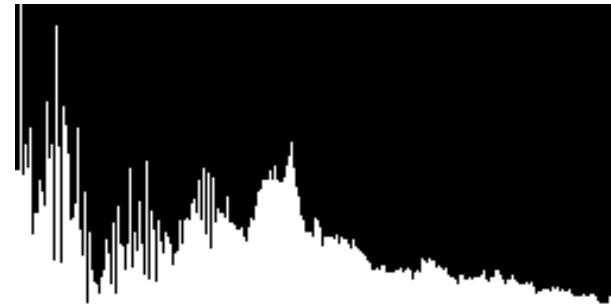
- Decreases image contrast by compressing the gray levels
- However this method may not be useful as an image enhancement tool, but it is used in an image sharpening algorithm (unsharp masking) as a part of an enhancement technique

# Histogram Shrinking

[www.rejinpaul.com](http://www.rejinpaul.com)



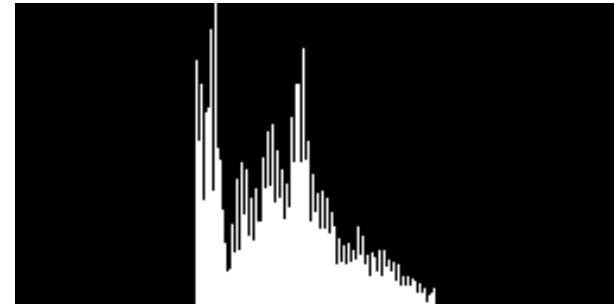
a) Original image



b) Histogram of image



c) Image after shrinking the histogram  
to the range [75,175]



d) Histogram of image (c)

**Download updated materials from Rejinpaul Network App**

## ✓ **Histogram slide**

- Used to make an image either darker or lighter, but retain the relationship between gray level values
- Accomplished by simply adding or subtracting a fixed number from all of the gray level values, as follows:

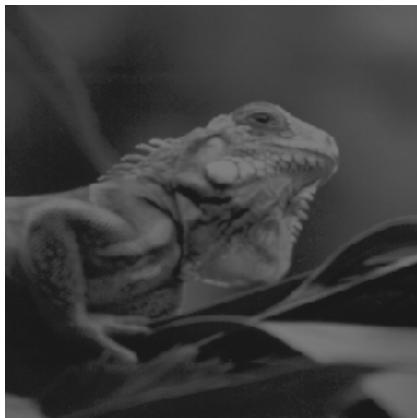
$$\text{Slide}(I(r, c)) = I(r, c) + \text{OFFSET}$$

where  
the histogram

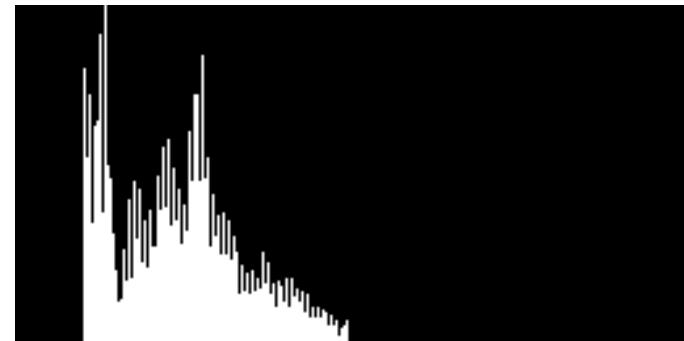
:lide

- In this equation we assume that any values slid past the minimum and maximum values will be clipped to the respective minimum or maximum
- A positive OFFSET value will increase the overall brightness, while a negative OFFSET will create a darker image

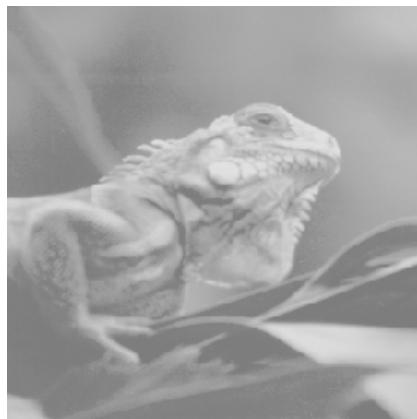
# Histogram Slide



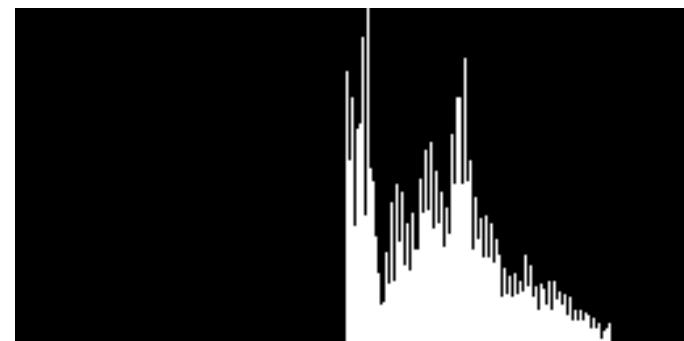
a) Resultant image from sliding the histogram down by 50



b) Histogram of image (a)



c) Resultant image from sliding the histogram up by 50  
**Download updated materials from Rejinpaul Network App**



d) Histogram of image (c)

✓ **Histogram equalization**

- A technique where the histogram of the resultant image is as flat as possible
- The theoretical basis for histogram equalization involves probability theory, where we treat the histogram as the probability distribution of the gray levels
- Its function is similar to that of a histogram stretch but often provides more visually pleasing results across a wider range of images

- Consists of four steps:
  1. Find the running sum of the histogram values
  2. Normalize the values from step (1) by dividing by the total number of pixels
  3. Multiply the values from step (2) by the maximum gray level value and round
  4. Map the gray level values to the results from step (3) using a one-to-one correspondence

## Example:

[www.rejinpaul.com](http://www.rejinpaul.com)

3-bits per pixel image – range is 0 to 7.

Given the following histogram:

<u>Gray Level Value</u>	Number of Pixels (Histogram values)
0	10
1	8
2	9
3	2
4	14
5	1
6	5
7	2

1) Create a running sum of the histogram values.

This means the first value is 10, the second is  $10+8=18$ , next  $10+8+9=27$ , and so on. Here we get 10, 18, 27, 29, 43, 44, 49, 51

2) Normalize by dividing by the total number of pixels. The total number of pixels is:

$10+8+9+2+14+1+5+0 = 51$  (note this is the last number from step 1), so we get:  $10/51, 18/51, 27/51, 29/51, 43/51, 44/51, 49/51, 51/51$

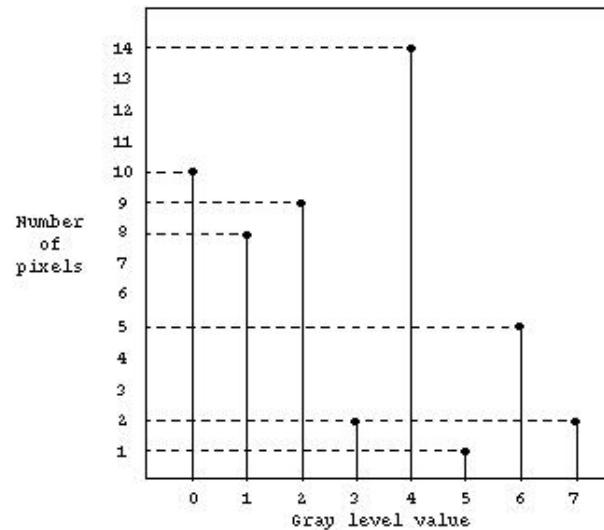
3) Multiply these values by the maximum gray level values, in this case 7, and then round the result to the closest integer. After this is done we obtain: 1, 2, 4, 4, 6, 6, 7, 7

4) Map the original values to the results from step 3 by a one-to-one correspondence. This is done as follows:

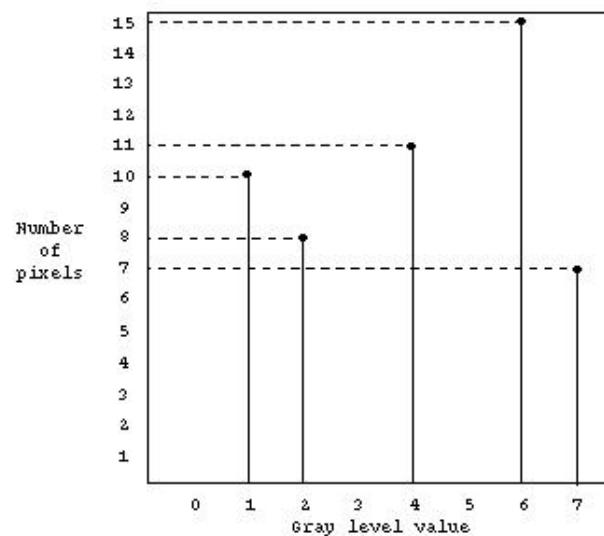
Original Gray Level Value	Histogram Equalized Values
0	1
1	2
2	4
3	4
4	6
5	6
6	7
7	7

- All pixels in the original image with gray level 0 are set to 1, values of 1 are set to 2, 2 set to 4, 3 set to 4, and so on. After the histogram equalization values are calculated and can be implemented efficiently with a look-up-table (LUT), as discussed in Chapter 2
- We can see the original histogram and the resulting histogram equalized histogram in Fig. 8.2.14. Although the result is not flat, it is closer to being flat than the original histogram

Figure 8.2-14: Histogram Equalization [www.rejinpaul.com](http://www.rejinpaul.com)



a) Original histogram



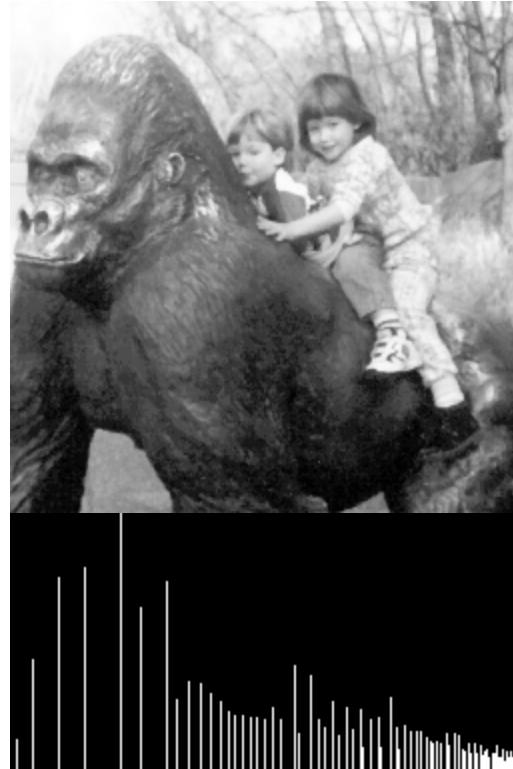
b) After histogram equalization

# Histogram Equalization Examples

1.



Input image



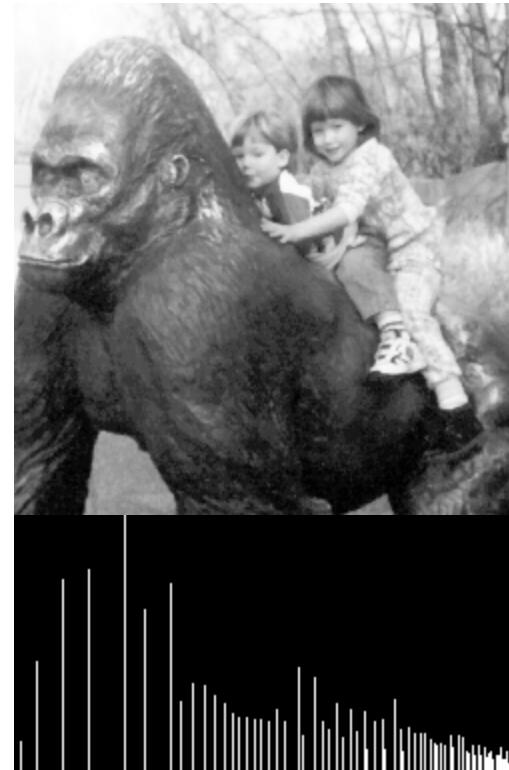
Resultant image after histogram equalization

# Histogram Equalization Examples (contd)

2.



Input image



Resultant image after histogram equalization

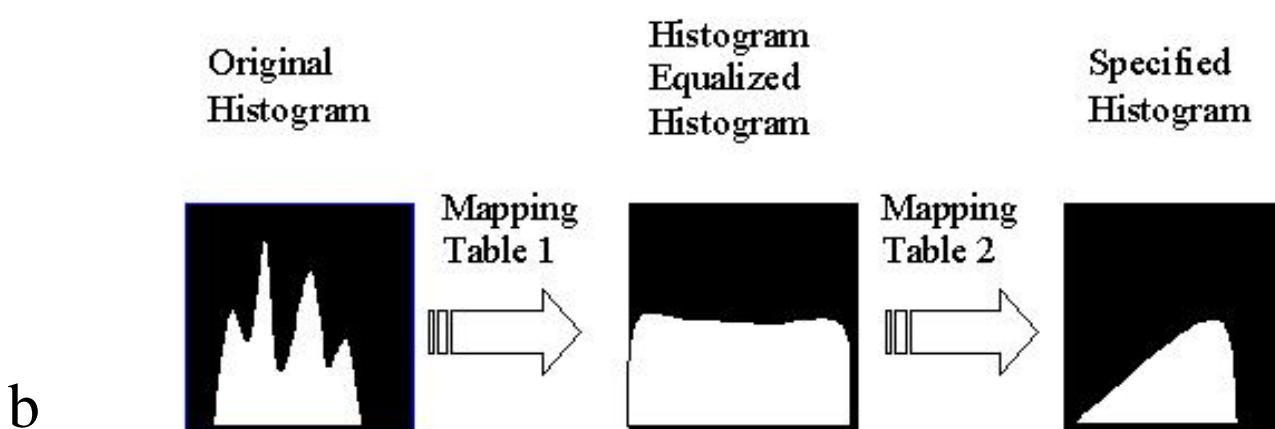
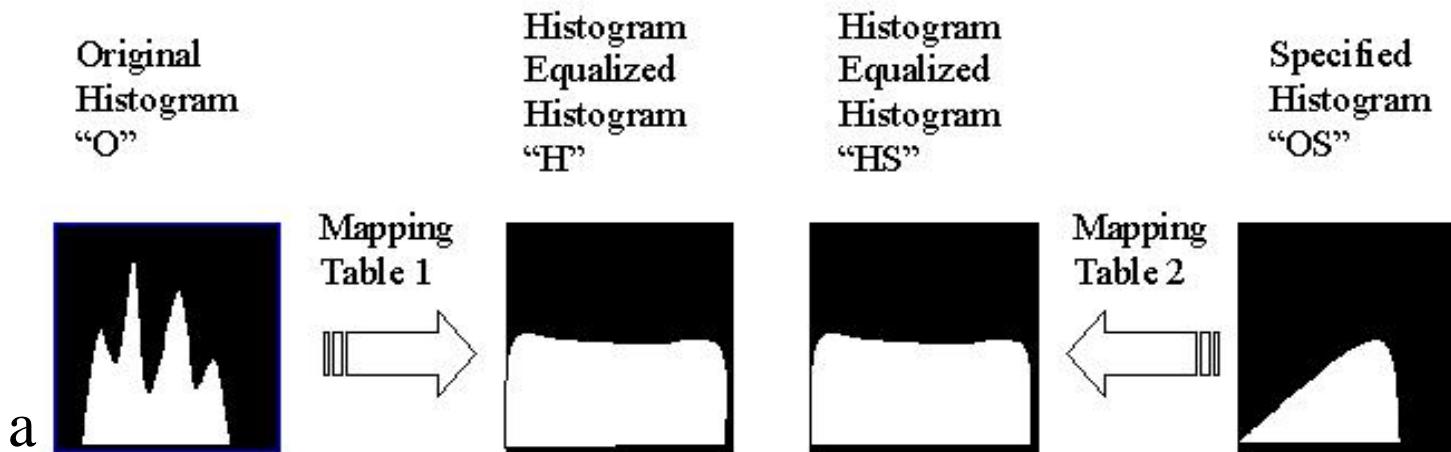
**Note:** As can be seen histogram equalization provides similar results regardless of the input image

- Histogram equalization of a digital image will not typically provide a histogram that is perfectly flat, but it will make it as flat as possible
- Histogram equalization may not always provide the desired effect, since its goal is fixed – to distribute the gray level values as evenly as possible. To allow for interactive histogram manipulation, the ability to specify the histogram is necessary

## ✓ **Histogram specification**

- Process of defining a histogram and modifying the histogram of the original image to match the histogram as specified
- Key concept is to picture the original image being histogram equalized, and the specified histogram being histogram equalized

**Histogram Specification.** This figure is a conceptual look at histogram specification. a) Here we depict the histogram equalized versions of the original image histogram and the specified histogram. Now we have a common histogram for both, the histogram equalized version should both be approximately flat. b) Now we can use the histogram equalization mapping tables to get from the original histogram to the specified histogram



- Histogram specification consists of following 5 steps:
  1. Specify the desired histogram
  2. Find the mapping table to histogram  
equalize the image, Mapping Table 1,
  3. Find the mapping table to histogram  
equalize the values of the specified  
histogram, Mapping Table 2

- Histogram specification steps (continued)
4. Use mapping Tables 1 & 2 to find the mapping table to map the original values to the histogram equalized values and then to the specified histogram values
  5. Use the table from step (4) to map the original values to the specified histogram values

## EXAMPLE:

1) Specify the desired histogram:

<u>Gray Level Value</u>	<u>Number of pixels in desired histogram</u>
0	1
1	5
2	10
3	15
4	20
5	0
6	0
7	0

2) For this we will use the image and mapping table from the previous example, where the histogram equalization mapping table (Mapping Table 1) is given by:

Original Gray Level Value level values-OS	Histogram Equalized equalized values-HS
0	1
1	2
2	4
3	4
4	6
5	6
6	7
7	7

3) Find the histogram equalization mapping table  
(Mapping Table 2) for the specified histogram:

Gray Level Value - OS	Histogram Equalized Values – HS .
0	$\text{round}(1/51)*7 = 0$
1	$\text{round}(6/51)*7 = 1$
2	$\text{round}(16/51)*7 = 2$
3	$\text{round}(31/51)*7 = 4$
4	$\text{round}(51/51)*7 = 7$
5	$\text{round}(51/51)*7 = 7$
6	$\text{round}(51/51)*7 = 7$
7	$\text{round}(51/51)*7 = 7$

- 4) Use Mapping Tables 1 and 2 to find the final mapping table by mapping the values first to the histogram equalized values and then to the specified histogram values. (*Mapping Table 2, columns switched to match Fig. 8.2.16 – slide 57*)

Mapping Table 1

O	H
0	1
1	2
2	4
3	4
4	6
5	6
6	7
7	7

Mapping Table 2

HS	OS	M
0	0	1
1	1	2
2	2	3
4	3	3
7	4	4
7	5	4
7	6	4
7	7	4

5) Use the table from STEP 4 to perform the histogram specification mapping. For this all we need are columns O (or OS) and M:

O	M
0	1
1	2
2	3
3	3
4	4
5	4
6	4
7	4

Now, all the 0's get mapped to 1's, the 1's to 2's, the 3's to 3's and so on

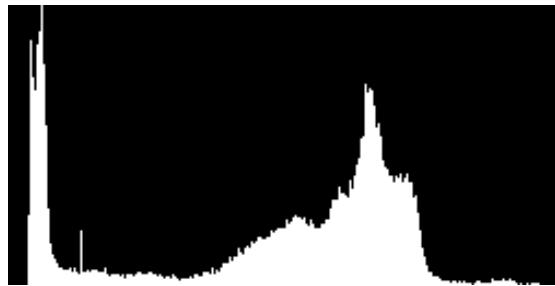
- In practice, the desired histogram is often specified by a continuous (possibly non-linear) function, for example a sine or a log function.
- To obtain the numbers for the specified histogram the function is sampled, the values are normalized to 1, and then multiplied by the total number of pixels in the image

# Histogram Specification Examples

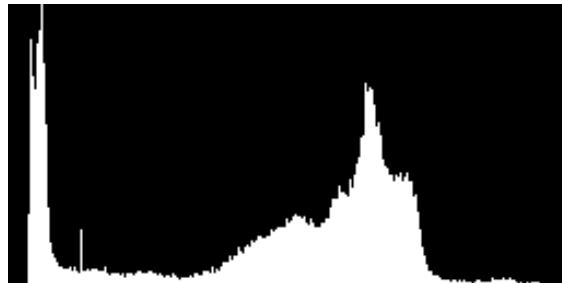
Original  
image



Histogram  
of  
original  
image



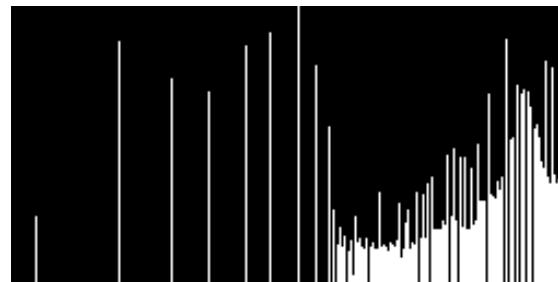
# Histogram Specification Examples (contd)



Original histogram

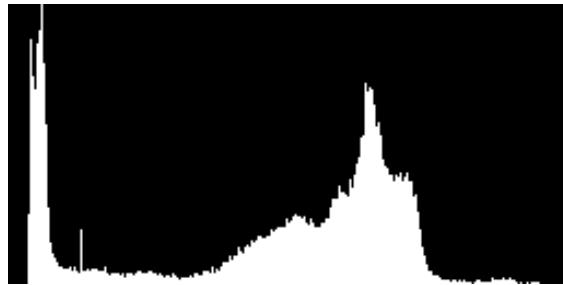


Specified histogram,  $\exp(0.015*x)$



Output image and its histogram

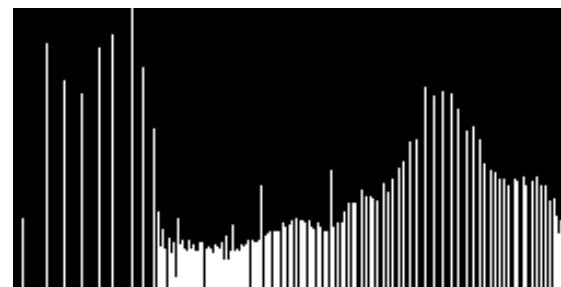
# Histogram Specification Examples (contd)



Original histogram



Specified histogram,  $\log(0.5*x+2)$



Output image and its histogram

# IMAGE ENHANCEMENT

## SPATIAL AVERAGING

Download updated materials from Rejinpaul Network App

# Image Averaging

- A noisy image:

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

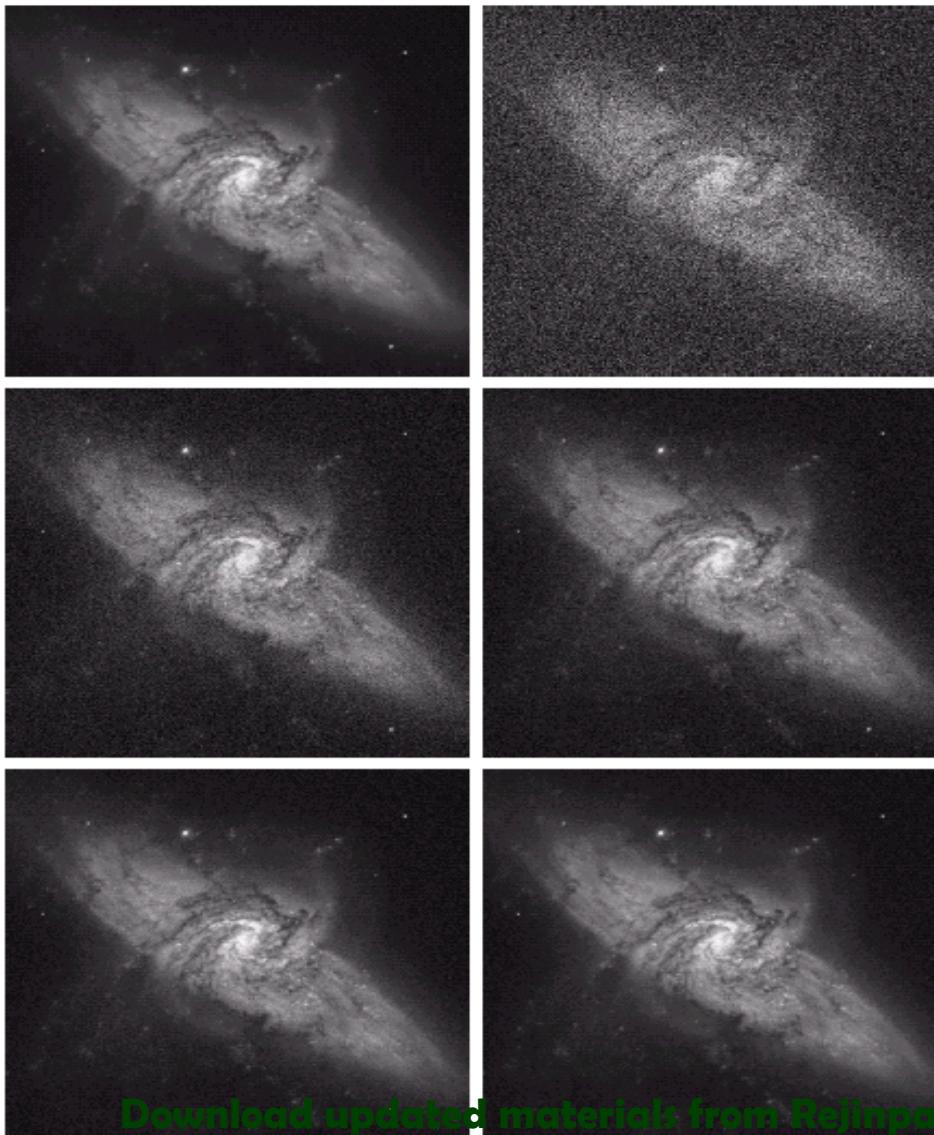
- Averaging K different noisy images:

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

# Image Averaging

- As K increases, the variability of the pixel values at each location decreases.
  - This means that  $\bar{g}(x,y)$  approaches  $f(x,y)$  as the number of noisy images used in the averaging process increases.
- Registering(aligned) of the images is necessary to avoid blurring in the output image.

# Image Enhancement in the Spatial Domain

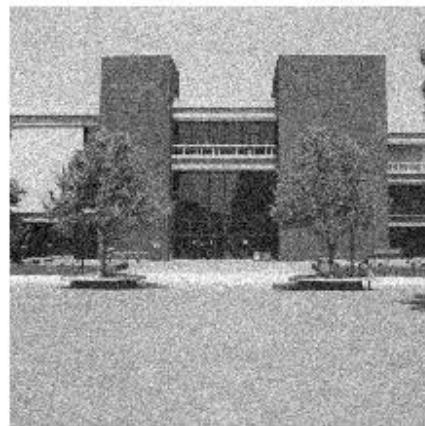


a b  
c d  
e f

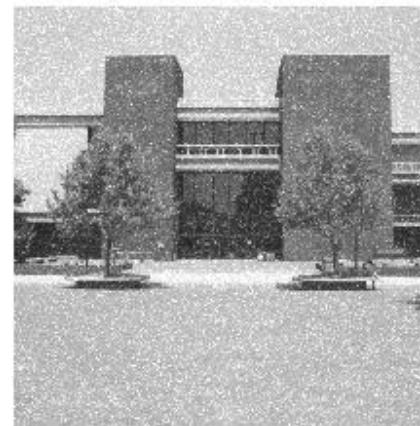
**FIGURE 3.30** (a) Image of Galaxy Pair NGC 3314, (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging 8, 16, 64, and 128 noisy images. (Original image courtesy of NASA.)

Figure 9.3-6: Arithmetic Mean Filter

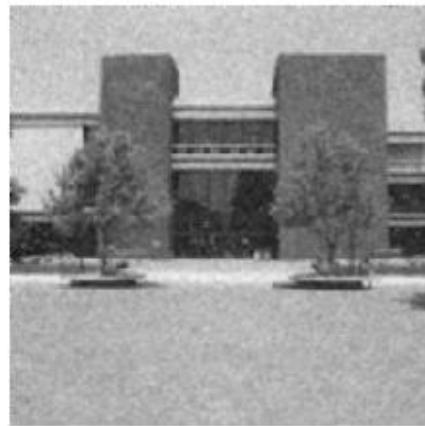
[www.rejinpaul.com](http://www.rejinpaul.com)



a) Image with gaussian noise  
variance = 300, mean = 0



b) Image with gamma noise  
variance = 300, alpha = 1



c) Result of arithmetic mean  
filter, mask size = 3, on  
image with gaussian noise



d) Result of arithmetic mean  
filter, mask size = 3, on  
image with gamma noise

Figure 9.3.6, continued



e) Result of arithmetic mean  
filter, mask size = 5, on  
image with gaussian noise

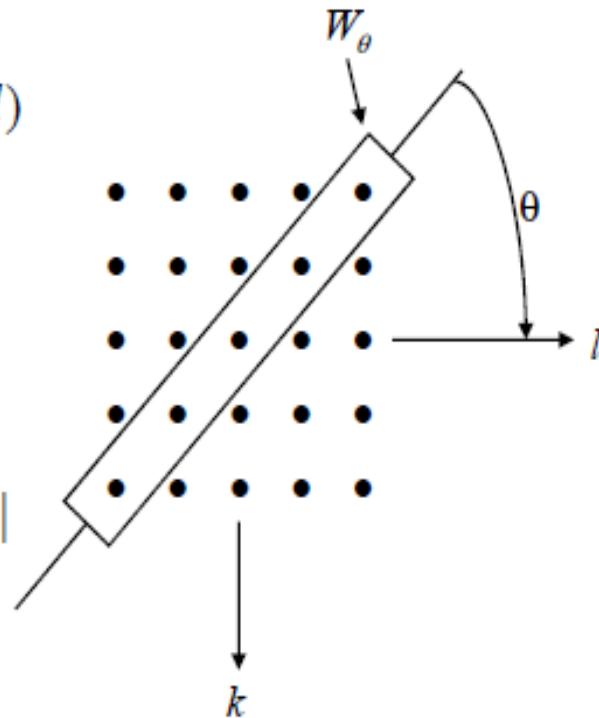


f) Result of arithmetic mean  
filter, mask size = 5, on  
image with gamma noise

- Directional Smoothing
  - to protect the edges from blurring while smoothing

$$v(m,n : \theta) = \frac{1}{N_\theta} \sum_{(k,l) \in W_\theta} y(m-k, n-l)$$

$$\Rightarrow v(m,n) = v(m,n : \theta^*)$$



Find out  $\theta^*$

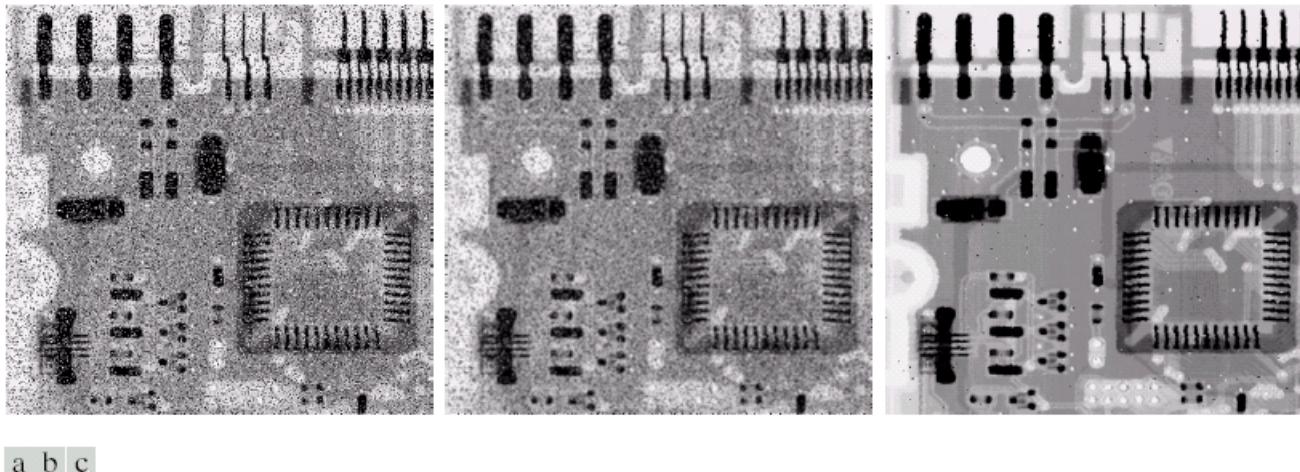
such that  $\min |y(m,n) - v(m,n : \theta^*)|$

# Smoothing Filters

- Median filtering (nonlinear)
  - Used primarily for noise reduction (eliminates isolated spikes)
  - The gray level of each pixel is replaced by the median of the gray levels in the neighborhood of that pixel (instead of by the average as before).

# Chapter 3

## Image Enhancement in the Spatial Domain



a b c

**FIGURE 3.37** (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a  $3 \times 3$  averaging mask. (c) Noise reduction with a  $3 \times 3$  median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

# Median Filter



a) Image with added salt-and-pepper noise,  
the probability for salt = probability  
for pepper = 0.10



b) After median filtering with a 3x3  
window, all the noise is not removed

## Median Filter



- c) After median filtering with a 5x5 window, all the noise is removed, but the image is blurry acquiring the “painted” effect

- The **contra-harmonic mean filter** works well for images containing salt OR pepper type noise, depending on the filter order,  $R$ :

$$\text{Contra - Harmonic Mean} = \frac{\sum_{(r,c) \in W} d(r, c)^{R+1}}{\sum_{(r,c) \in W} d(r, c)^R}$$

where  $W$  is the  $N \times N$  window under consideration

- For negative values of  $R$ , it eliminates salt-type noise, while for positive values of  $R$ , it eliminates pepper-type noise

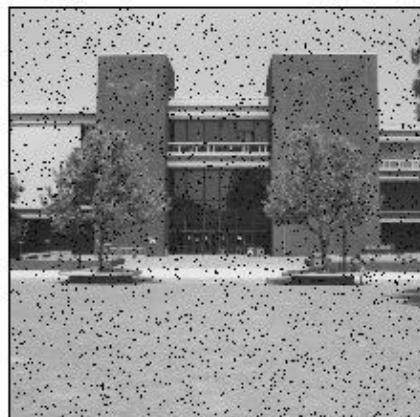
Figure 9.3.9: Contra-harmonic Mean Filter [www.rejinpaul.com](http://www.rejinpaul.com)



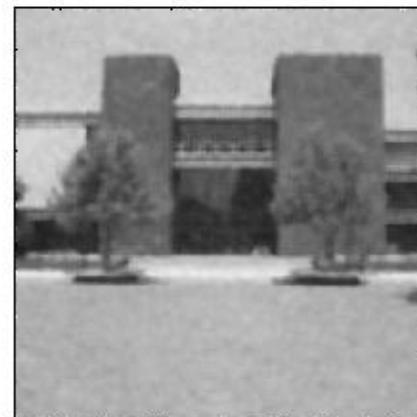
a) Image with salt noise,  
probability = .04



b) Result of contra-harmonic  
mean filter, mask size = 3,  
order = -3



c) Image with pepper noise,  
probability = .04



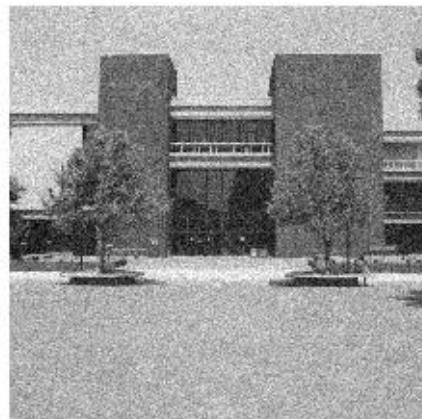
d) Result of contra-harmonic  
mean filter, mask size = 3,  
order = +3

- The **geometric mean filter** works best with Gaussian noise, and retains detail information better than an arithmetic mean filter
- It is defined as the product of the pixel values within the window, raised to the  $1/(N^*N)$  power:

$$\text{Geometric Mean} = \prod_{(r,c) \in W} [d(r,c)]^{\frac{1}{N}}$$

Figure 9.3-10: Geometric Mean Filter

[www.rejinpaul.com](http://www.rejinpaul.com)



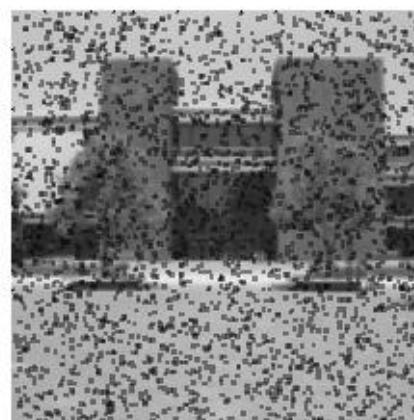
a) Image with gaussian noise,  
variance = 300, mean = 0



b) Result of geometric mean  
filter, mask size = 3, on  
image with gaussian noise



c) Image with pepper noise,  
probability = .04



d) Result of geometric mean  
filter, mask size = 3, on  
image with pepper noise

- The **harmonic mean filter** fails with pepper noise, but works well for salt noise
- It is defined as follows:

$$\text{Harmonic Mean} = \frac{N^2}{\sum_{(r,c) \in W} \frac{1}{d(r,c)}}$$

- This filter also works with Gaussian noise, retaining detail information better than the arithmetic mean filter

Figure 9.3-11: Harmonic Mean Filter



a) Image with gaussian noise,  
variance = 300, mean = 0



b) Result of harmonic mean  
filter, mask size = 3, on  
image with gaussian noise



c) Image with salt noise,  
probability = .04



d) Result of harmonic mean  
filter, mask size = 3, on  
image with salt noise

- The *Yp mean filter* is defined as follows:

$$Y_p \text{ Mean} = \left[ \sum_{(r,c) \in W} \frac{d(r, c)^P}{N^2} \right]^{\frac{1}{P}}$$

- This filter removes salt noise for negative values of  $P$ , and pepper noise for positive values of  $P$

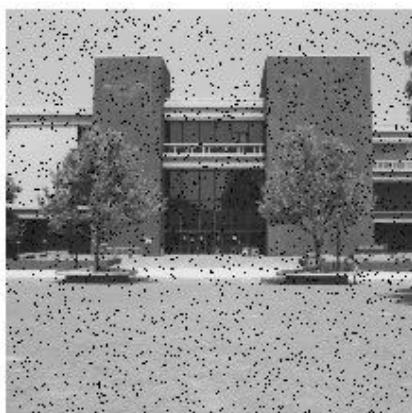
Figure 9.3-12: Yp Mean Filter



a) Image with salt noise,  
probability = .04



b) Result of Yp mean filter,  
mask size = 3, order = -3  
on image with salt noise



c) Image with pepper noise,  
probability = .04



d) Result of Yp mean filter,  
mask size = 3, order = +3  
on image with pepper noise

# HOMOMORPHIC FILTERING

Download updated materials from Rejinpaul Network App

- It simultaneously normalizes the brightness across an image and increases contrast.
- Filtering is used to remove multiplicative\_noise
- Illumination and reflectance are not separable
- Illumination and reflectance combine multiplicatively
- Components are made additive by taking the logarithm of the image intensity

- Multiplicative components of the image can be separated linearly in the frequency domain
- To make the illumination of an image more even, the high-frequency components are increased and low-frequency components are decreased
- High-frequency components are assumed to represent mostly the reflectance in the scene (the amount of light reflected off the object in the scene)
- Low-frequency components are assumed to represent mostly the illumination in the scene
- High-pass\_filtering is used to suppress low frequencies and amplify high frequencies, in the log-intensity domain

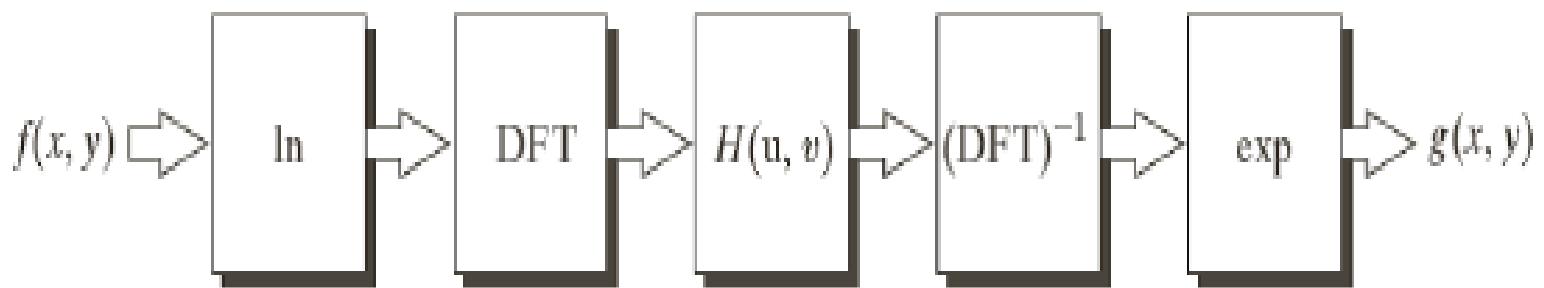
- The illumination component tends to vary slowly across the image.
- The reflectance tends to vary rapidly, particularly at junctions of dissimilar objects.
- Therefore, by applying a frequency domain filter of the form we can reduce intensity variation across the image while highlighting detail.

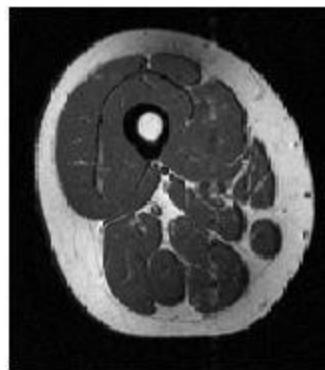
$$f(x, y) = i(x, y) \cdot r(x, y)$$

$$g = \ln f = \ln i + \ln r.$$

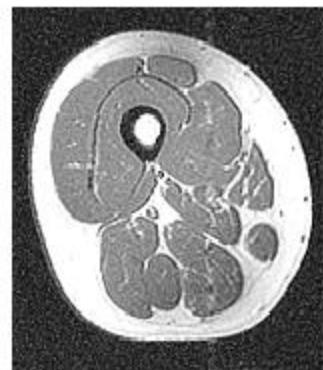
$$\mathcal{F}\{g(x, y)\} = \mathcal{F}\{\ln i(x, y)\} + \mathcal{F}\{\ln r(x, y)\}$$

$$G(u, v) = I_l(u, v) + R_l(u, v).$$



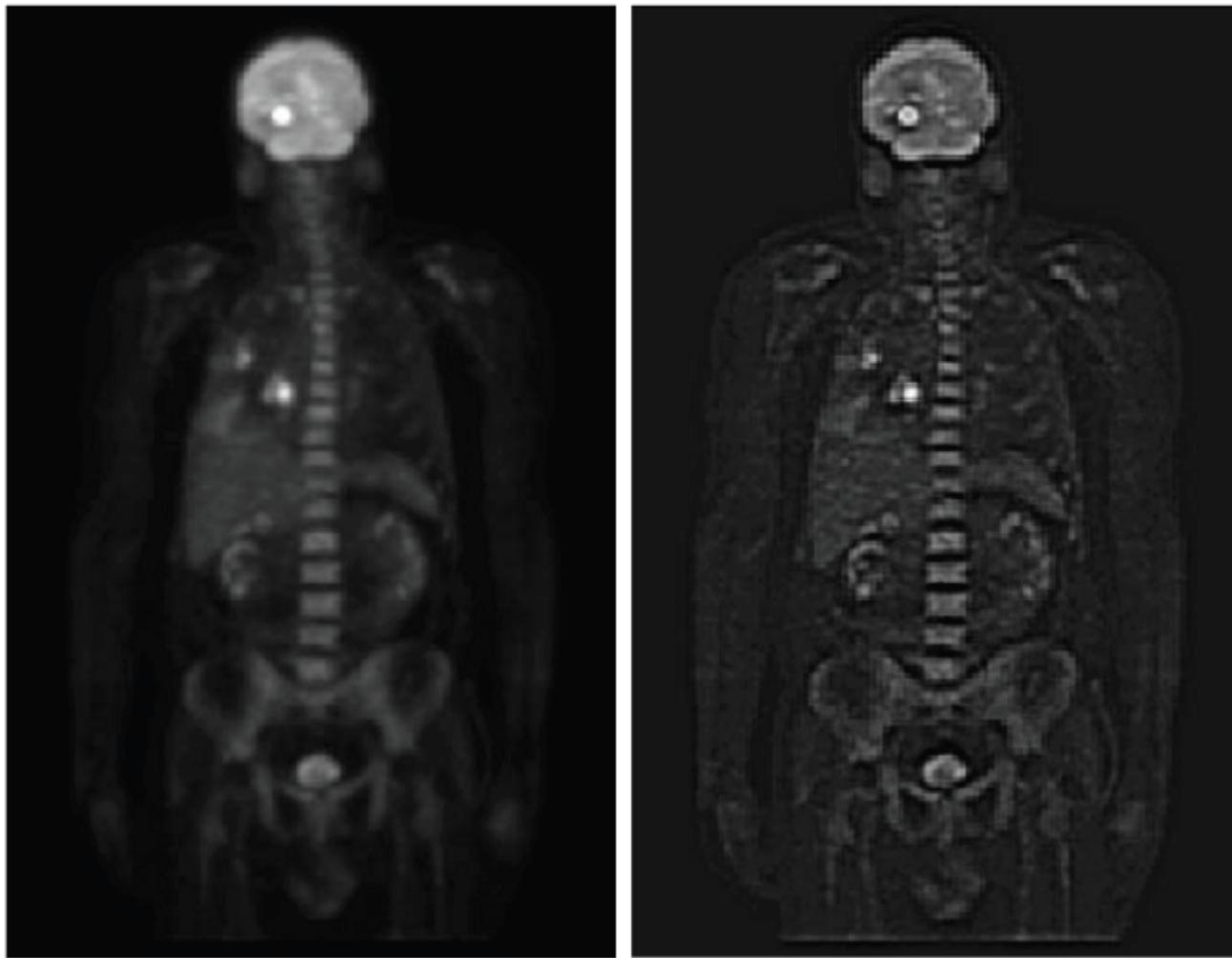


(a)



(b)

## Homomorphic filtering: PET example



# Color Image Enhancement

**Download updated materials from Rejinpaul Network App**

# Enhancement of Color Images

- Gray scale transforms and histogram modification techniques can be applied by treating a color image as three gray images
- Care must be taken in how this is done to avoid color shifts

- ✓ Histogram modification can be performed on color images, but doing it on each color band separately can create relative color changes
  
- ✓ The relative color can be retained by applying the gray scale modification technique to one of the color bands, and then using the ratios from the original image to find the other values

# Histogram Equalization of Color Images



a) Original poor contrast image



b) Histogram equalization based on  
the red color band

# Histogram Equalization of Color Images (contd)



c) Histogram equalization based on  
the green color band



d) Histogram equalization based on  
the blue color band

Note: In this case the red band gives the best results  
This will depend on the image and the desired result

- ✓ Typically the most important color band is selected, and this choice is very much application-specific and will not always provide us with the desired result
  
- ✓ Often, we really want to apply the gray scale modification method to the image brightness only, even with color images

- ✓ Histogram modification on color images can be performed in the following ways:
  - Retain the RGB ratios and perform the modification on one band only, then use the ratios to get the other two bands' values, or
  - Perform a color transform, such as HSL, do the modification on the lightness (brightness band), then do the inverse color transform

# UNIT III

# IMAGE RESTORATION

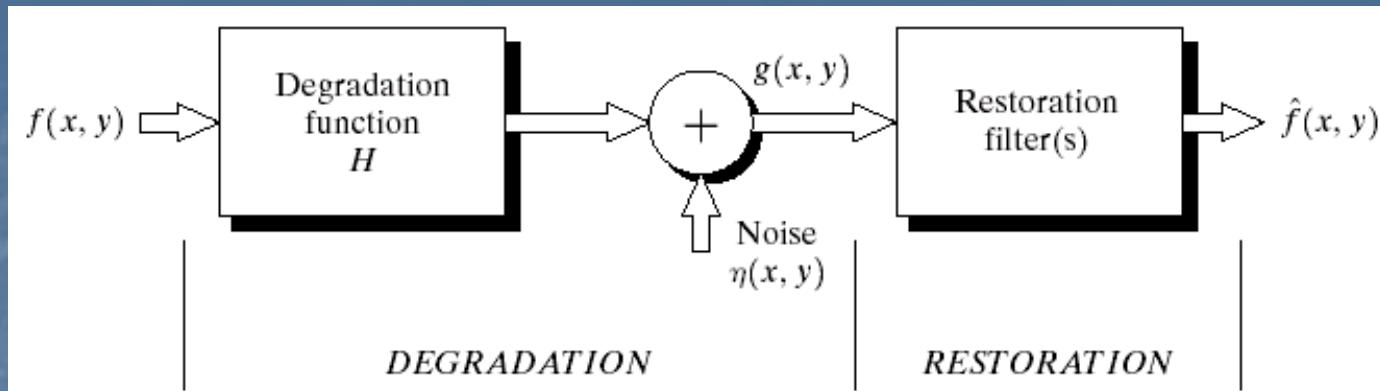
- As in enhancement, goal of restoration tech., is to improve an image in some predefined manner
- Enhancement – subjective process
- Restoration – Objective process
- It is a tech., to reconstruct or recover an image that has been degraded by using a priori knowledge of the degradation phenomenon
- Modeling the degradation and applying the inverse process in order to recover the original image

## ➤ The Degradation Function

- ✓ Degradation occurs in the form of blurring due the signal fluctuating during the measured time interval, imperfect lenses, motion of the object or imaging device, and spatial quantization
  
- ✓ The degradation is either spatially-invariant or spatially-variant

- ✓ *Spatially-invariant* degradation affects all pixels in the image the same
  
- ✓ Examples of spatially-invariant degradation includes poor lens focus and camera motion
  
- ✓ *Spatially-variant* degradations are dependent on spatial location and are more difficult to model

- ✓ Examples of spatially-variant degradations include imperfections in a lens or object motion
- ✓ Spatially-variant degradations can often be modeled as being spatially-invariant over small regions
- ✓ Image degradation functions can be considered to be linear or nonlinear, here assume linearity



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

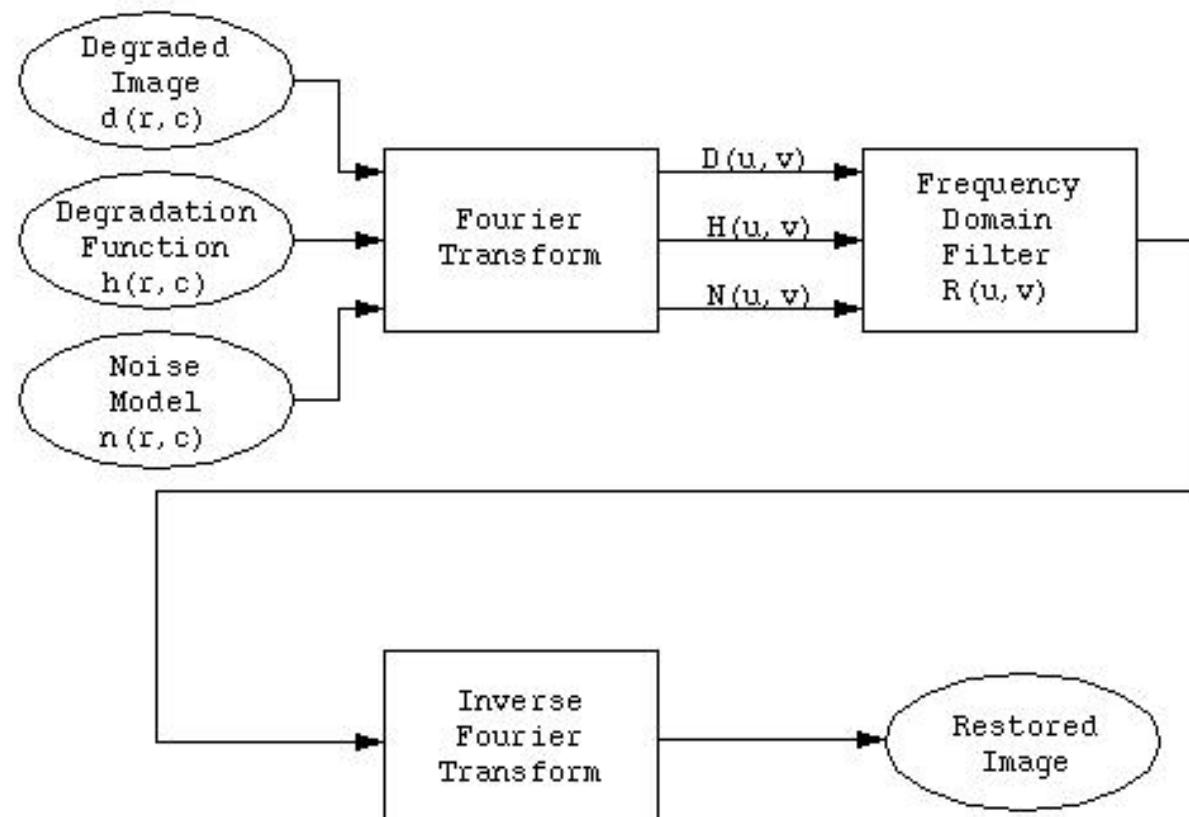
$$g(x, y) = h(x, y) * f(x, y) + \eta(x, y)$$

$$\text{or } G(u, v) = H(u, v) F(u, v) + N(u, v)$$

## ➤ Frequency Domain Filters

- ✓ Frequency domain filtering operates by using the Fourier transform representation of images
  
- ✓ This representation consists of information about the spatial frequency content of the image, also referred to as the *spectrum* of the image

Figure 9.5-1: Frequency Domain Filtering



- ✓ The Fourier transform is performed on three spatial domain functions:
  1. *The degraded image,  $d(r,c)$*
  2. *The degradation function,  $h(r,c)$*
  3. *The noise model,  $n(r,c)$*
- ✓ The frequency domain filter is applied to the Fourier transform outputs,  $N(u,v)$ ,  $D(u,v)$ , and  $H(u,v)$

- ✓ The output of the filter operation undergoes an inverse Fourier transform to give the restored image
- ✓ The frequency domain filters incorporate information regarding the noise and the PSF into their model, and are based on the mathematical model given as

$$D(u, v) = H(u, v)I(u, v) + N(u, v)$$

Where  $D(u, v)$ = Fourier transform of the degraded image

$H(u, v)$ = Fourier transform of the degradation function

$I(u, v)$ = Fourier transform of the original image

$N(u, v)$ = Fourier transform of the additive noise function

- ✓ In order to obtain the restored image, the general form is as follows:

$$\hat{I}(r, c) = F^{-1} [\hat{I}(u, v)] = F^{-1} [R_{\text{type}}(u, v) D(u, v)]$$

where  $\hat{I}(r, c)$  = the restored image, an approximation to  $I(r, c)$

$F^{-1} []$  = the inverse Fourier transform

$R_{\text{type}}(u, v)$  = the Restoration (frequency domain) filter, the subscript defines the type of filter

- ✓ Many of the filters to be discussed ahead assume that the image and noise functions are *stationary*, which means spatial frequency content is fairly constant across the entire image

- ✓ Observation of noise images shows they may be stationary – subimages tend to be self-similar
- ✓ Most real images are not stationary – some areas may be primarily low frequency, and others may have more high frequency energy
- ✓ Advanced adaptive filtering methods (discussed later) can help manage this problem

# Inverse Filter

## 5.7 Inverse Filtering

Degraded image is given by  $G(u,v) = \underbrace{H(u,v)}_{\text{degradation function}} F(u,v) + \underbrace{N(u,v)}_{\text{noise}}$

Estimate  $\tilde{F}(u,v)$  by simply dividing  $G(u,v)$  by  $H(u,v)$

$$\text{Then } \tilde{F}(u,v) = \frac{H(u,v) F(u,v) + N(u,v)}{H(u,v)}$$

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

can never recover  $F(u,v)$  exactly

1.  $N(u,v)$  is not known since  $n(x,y)$  is a random variable

2. If  $H(u,v) \rightarrow 0$  then  $\frac{N(u,v)}{H(u,v)}$  will dominate

This can be somewhat overcome  
by restricting the analysis to values  
near the origin.

This example shows the problems of small values of  $H(u,v)$  in the inversion process.

$$\text{For } H(u,v) = e^{-k\left[\left(u-\frac{M}{2}\right)^2 + \left(v-\frac{N}{2}\right)^2\right]^{\frac{5}{2}}}$$

This is never zero but can get small.

Using  $\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$  gives (a) above.

We can improve the result by cutting off values of  $\frac{G(u,v)}{H(u,v)}$  outside a radius  $D_0$ .

The cutoff shown above was done using a Butterworth low-pass filter of order 10.

a b  
c d

**FIGURE 5.27**

Restoring Fig. 5.25(b) with Eq. (5.7-1).  
(a) Result of using the full filter. (b) Result with  $H$  cut off outside a radius of 40; (c) outside a radius of 70; and (d) outside a radius of 85.



- inverse filtration gives poor results in pixels suffering from noise since the noise is not taken into account

# WIENER FILTERING

MINIMUM MEAN SQUARE ERROR  
FILTERING

To generate the best estimate  $\hat{f}$  of  $f$  we minimize

$$e^2 = \underset{\text{expected value}}{\mathbb{E}} \{ (f - \hat{f})^2 \}$$

Assumptions

1.  $f$  and  $n$  are uncorrelated
2.  $f$  and/or  $n$  is zero mean
3. gray levels in  $\hat{f}$  are a linear function of gray levels in  $f$

Then the best estimate  $\hat{F}(u,v)$  is given by

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

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

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

where  $H(u,v)$  = degradation function

$H^*(u,v)$  = complex conjugate of  $H(u,v)$

$$|H(u,v)|^2 = H^*(u,v) H(u,v)$$

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

In practice  $S_f(u,v) = |F(u,v)|^2$  of the undegraded image is not usually known.

So we simply replace  $\frac{S_n(u,v)}{S_f(u,v)}$  by a constant  $k$

$$\hat{F}(u,v) = \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)| + k} G(u,v)$$



a b c

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

# Geometric Transformations

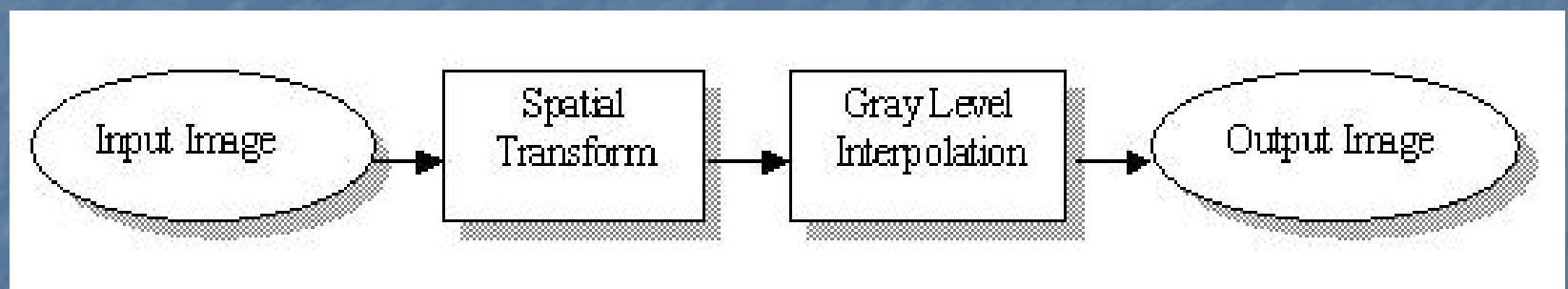
# ➤ Geometric Transforms

- ✓ Geometric transforms are by their very nature spatially-variant
- ✓ Used to modify the location of pixel values within an image, typically to correct images that have been spatially warped

- ✓ These methods are often referred to as *rubber-sheet transforms*, because the image is modeled as a sheet of rubber and stretched and shrunk, as required to correct for any spatial distortion
  
- ✓ This type of distortion can be caused by defective optics in an image acquisition system, distortion in image display devices, or 2-D imaging of 3-D surfaces

- ✓ The methods are used in map making, image registration, image morphing, and other applications requiring spatial modification
  
- ✓ Geometric transforms can also be used in image warping where the goal is to take a "good" image and distort it spatially
  
- ✓ The simplest geometric transforms are translate, rotate, zoom and shrink

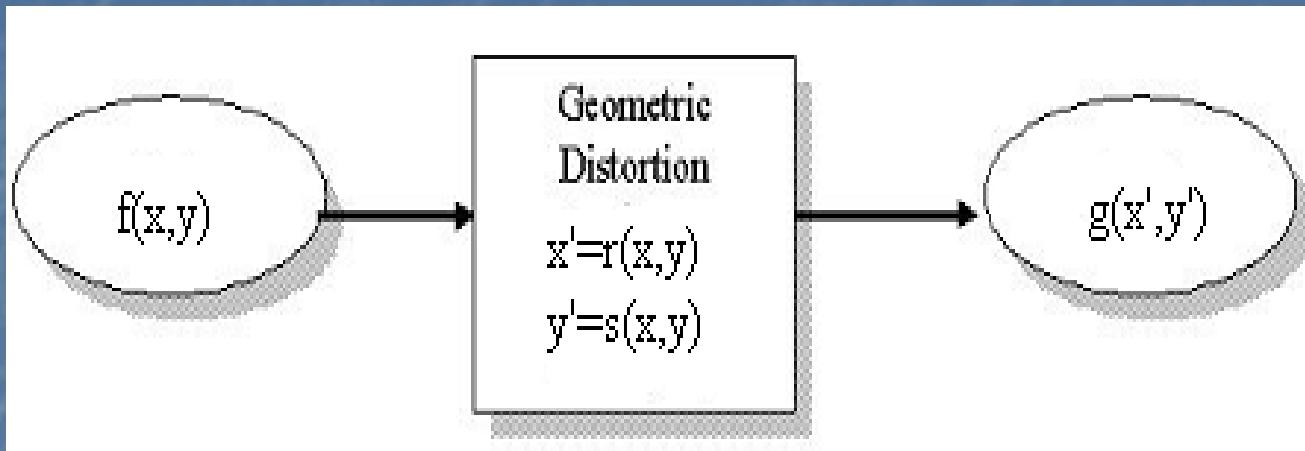
- ✓ The more sophisticated geometric transforms, require two steps:
  1. *Spatial transform*
  2. *Gray level interpolation*
  
- ✓ The model used for the *geometric transforms* is:



# ✓ Spatial Transforms

- *Spatial transforms* are used to map the input image location to a location in the output image
- It defines how the pixel values in the output image are to be arranged

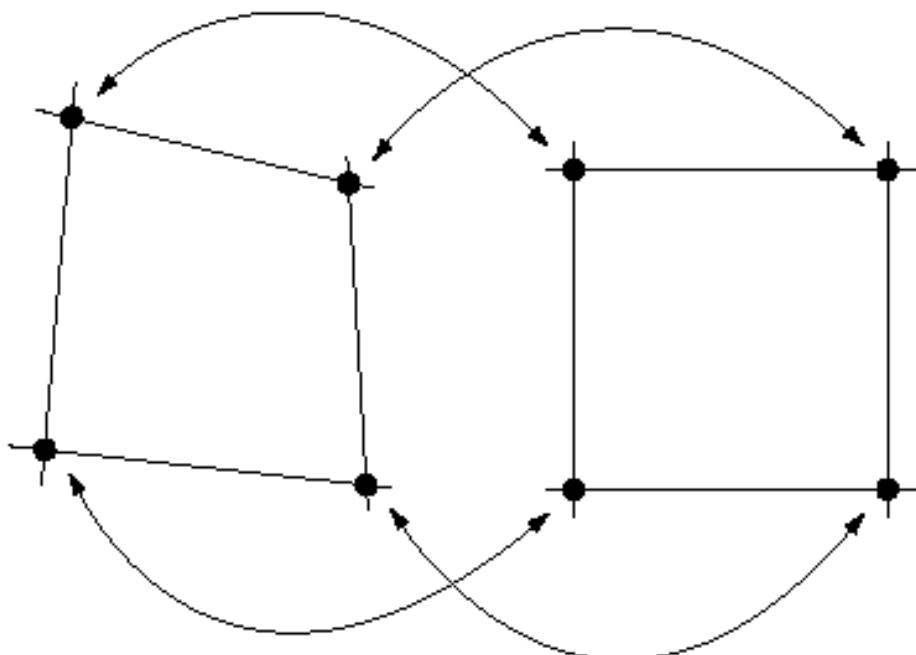
- *Spatial transforms* can be modeled as:



- Actual image  $f(x,y)$
- Geometric distorted image  $g(x',y')$
- $r(x,y)$ ,  $s(x,y)$  spatial transformations produced by  $g(x',y')$
- If  $r=x/2$  &  $s=y/2$  then the distortion is simply shrinking

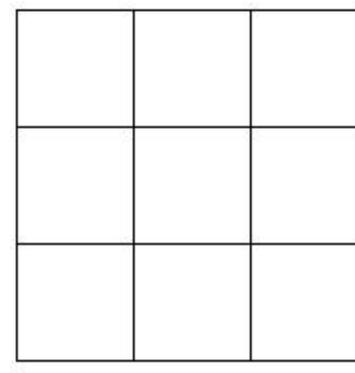
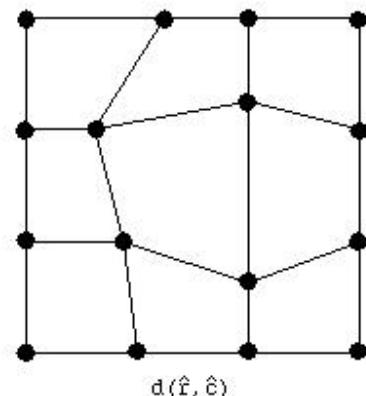
- To find these equations requires identifying a set of points in the original image that match points in the distorted image, called *tiepoints*
- The form of these equations is typically bilinear, although higher-order polynomials can be used

- A geometrically distorted image can be restored in the following way
1. Define quadrilaterals with known or best guessed tiepoints for the entire image
  2. Find the equations for each set of tiepoints
  3. Remap all the pixels within each quadrilateral subimage using the equations corresponding to those tiepoints

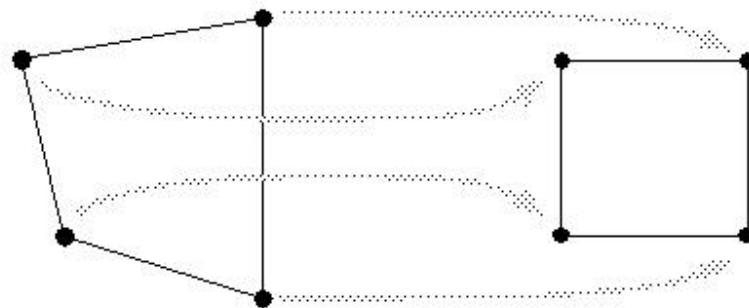


**FIGURE 5.32**  
Corresponding  
tiepoints in two  
image segments.

Figure 9.6-3: Restoring Geometric Distortion



a) Images divided into quadrilateral subimages



b) Center subimage showing corresponding points, also known as tie-points

- In step 2, using a bilinear model for the mapping equations, the four corner points are used to generate the equations:
- $r(x,y) = c_1x + c_2y + c_3xy + c_4$
- $s(x,y) = c_5x + c_6y + c_7xy + c_8$
- $x' = c_1x + c_2y + c_3xy + c_4$
- $y' = c_5x + c_6y + c_7xy + c_8$
- The  $c_i$  values are constants to be determined by solving the eight simultaneous equations

- Step 3 involves application of the mapping equations to all the  $(r, c)$  pairs in the corresponding quadrilateral in  $I(r, c)$

EXAMPLE 9.6.1:

Assume we have found the following mapping equations:

$$\hat{R}(r, c) = 5r + 3c + 3rc + 2 = \hat{r}$$

$$\hat{C}(r, c) = 1r + 1c + 2rc + 0 = \hat{c}$$

To find  $I(2, 3)$ , substitute  $(r, c) = (2, 3)$  into the above equations and we find:

$$\hat{R}(r, c) = 5(2) + 3(3) + 3(2)(3) + 2 = 39$$

$$\hat{C}(r, c) = 1(2) + 1(3) + 2(2)(3) + 0 = 17$$

Now, we let  $I(2, 3) = d(39, 17)$

**EXAMPLE 9.6.2:**

Assume we have found the following mapping equations:

$$\hat{R}(r,c) = 4.5r + 3c + 3.5rc + 2.4 = \hat{r}$$

$$\hat{C}(r,c) = 1.6r + 1c + 2.4rc + 0 = \hat{c}$$

To find  $I(2,3)$ , substitute  $(r,c) = (2,3)$  into the above equations and we find:

$$\hat{R}(r,c) = 4.5(2) + 3(3) + 3.5(2)(3) + 2.4 = 41.4$$

$$\hat{C}(r,c) = 1.6(2) + 1(3) + 2.4(2)(3) + 0 = 20.6$$

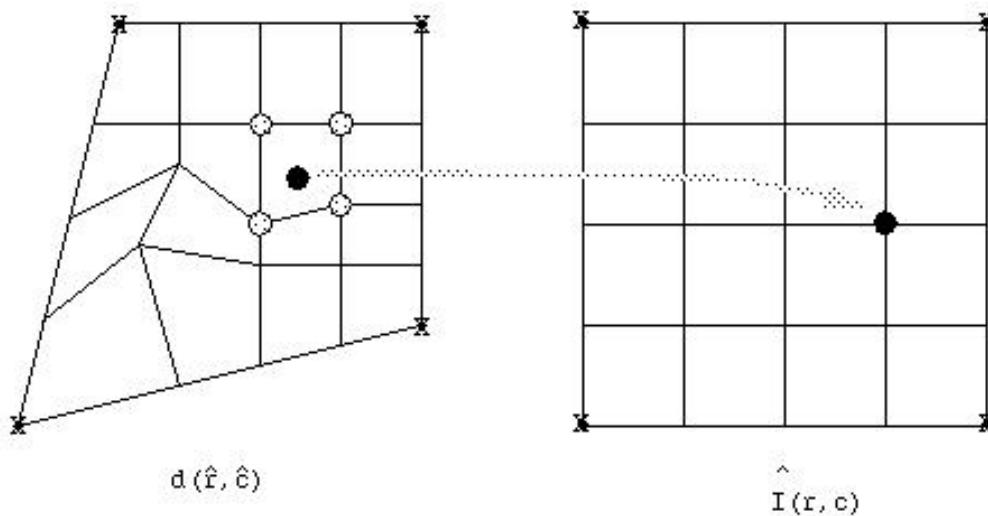
Now, we want to set  $I(2,3) = d(41.4, 20.6)$ .

- The difficulty in the above example arises when we try to determine the value of  $d(41.4,20.6)$
- Since the digital images are defined only at the integer values for  $(r,c)$ , gray interpolation must be performed
- In this case, we define  $\hat{I}(r,c)$  as an estimate to the original image  $I(r,c)$  to represent the restored image

## ✓ Gray Level Interpolation

- Gray Level Interpolation can be performed in three ways:
  1. *Nearest neighbor method*
  2. *Neighborhood average method*
  3. *Bilinear interpolation method*

Figure 9.6-4: Gray Level Interpolation



**x** = tiepoints

**○** = points used in determining gray level interpolation equations

**●** =  $d(\hat{f}, \hat{c})$  point we want to map to  $\hat{I}(\hat{r}, \hat{c})$

# 1. Nearest neighbor method

- Each pixel is assigned the value of the closest pixel in the distorted image
- It is similar to the zero-order hold
- It does not necessarily provide optimal results, but has the advantage of being *easy to implement* and *computationally fast*
- Object edges tend to appear *jagged* or *blocky*

## 2. Neighborhood average method

- Surrounding pixel values in the distorted image are used to estimate the desired value, and this estimated value is used in the restored image
- It can be performed in 1-D or 2-D
- It is of *medium complexity*, *reasonably fast*, and provides *smooth but blurred edges*

### 3. Bilinear interpolation method

- Bilinear interpolation is accomplished by the following equation:

$$g(\hat{r}, \hat{c}) = k_1 \hat{r} + k_2 \hat{c} + k_3 \hat{r}\hat{c} + k_4$$

where  $g(\hat{r}, \hat{c})$  = the gray level interpolating equation

- Bilinear interpolation is the *most complex, slowest*, but has the *best results*

- The constants,  $k_i$ , are different than the constants used in the spatial mapping equations
- The four unknown constants are found by using the four surrounding points shown in Figure 9.6-4
- The values for row and column, and the gray level values at each point are used

**EXAMPLE 9.6.3 :**

Suppose the four surrounding points are as follows:

$$d(\hat{r}, \hat{c}) \rightarrow d(1,2) = 50, d(1,3) = 55, d(2,2) = 44, d(2,3) = 48$$

Then we define the following four equations, and solve for the constants,  $k_i$ :

$$50 = k_1(1) + k_2(2) + k_3(1)(2) + k_4$$

$$55 = k_1(1) + k_2(3) + k_3(1)(3) + k_4$$

$$44 = k_1(2) + k_2(2) + k_3(2)(2) + k_4$$

$$48 = k_1(2) + k_2(3) + k_3(2)(3) + k_4$$

Solving these equations simultaneously gives us:

$$k_1 = -4, k_2 = 6, k_3 = -1, k_4 = 44$$

$$\therefore g(\hat{r}, \hat{c}) = -4\hat{r} + 6\hat{c} - \hat{r}\hat{c} + 44$$

After the equation,  $g(\hat{r}, \hat{c})$ , is found, the interpolated value can be determined. To do this we insert the noninteger values for row and column into the gray level interpolating equation, and the resulting  $g(\hat{r}, \hat{c})$  value is the interpolated gray level value.

#### EXAMPLE 9.6.4:

The preceding example assumes that the row and column coordinates are between rows 1 and 2, and column 2 and 3; for example  $\hat{r} = 1.3$  and  $\hat{c} = 2.6$ . Applying these values to the preceding gray level interpolating equation, we obtain:

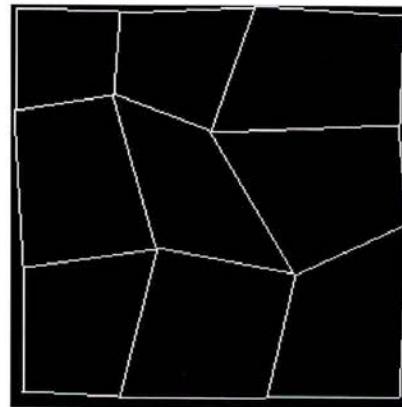
$$g(1.3, 2.6) = 4(1.3) + 6(2.6) - (1.3)(2.6) + 44 = 51.02 \approx 51$$

The gray level value of 51 (or 51.02 can be used if the image is of FLOAT data type) is then inserted into the restored image at the row and column location used to generate  $\hat{r} = 1.3$  and  $\hat{c} = 2.6$  from the mapping equations.

Figure 9.6-5: Geometric Restoration Example



a) Original image



b) A mesh defined by 16 tiepoints will be used to first distort and then restore the image.



c) The original image has been distorted using the bilinear interpolation method.



d) Restoration by the nearest neighbor method shows the blocky effect that occurs at edges.

Figure 9.6-5, continued



e) Restoration with neighborhood averaging interpolation provides smoother edges than with the nearest neighbor method, but also blurs the image.



f) Restoration by bilinear interpolation provides optimal results. Note that some distortion occurs at the boundaries of the mesh quadrilaterals.

- For applications requiring even higher quality results, such as medical imaging or computer-aided design (CAD) graphics, more mathematically complex methods can be used
- For example, *cubic convolution interpolation* will fit a smooth surface over a larger group of pixels to provide a reasonably optimal gray level value at any point on the surface

## ✓ The Geometric Restoration Procedure

- The complete procedure for restoring an image that has undergone geometric distortion is as follows:
  - 1) Find tiepoints throughout the image mapping the distorted image,  $d(\hat{r}, \hat{c})$  to the restored image,  $\hat{I}(r, c)$ .  $\hat{I}(r, c)$  is the estimate to the original, undistorted image  $I(r, c)$ .
  - 2) For each quadrilateral find the equations for  $\hat{R}(r, c)$  and  $\hat{C}(r, c)$

- 3) For each value of  $(r, c)$  in  $\hat{I}(r, c)$ , apply the equation pair,  $\hat{R}(r, c)$  and  $\hat{C}(r, c)$ , corresponding to the mapped quadrilateral to find  $(\hat{r}, \hat{c})$ .
- 4) Perform the selected method of gray level interpolation using the values for  $(\hat{r}, \hat{c})$  found in Step (3) to find the gray value for  $\hat{I}(r, c)$ .
- 5) Continue Step (3) and Step (4) until all values for  $\hat{I}(r, c)$  are found and we have our restored image.

- Many variations of this method are possible

# UNIT IV

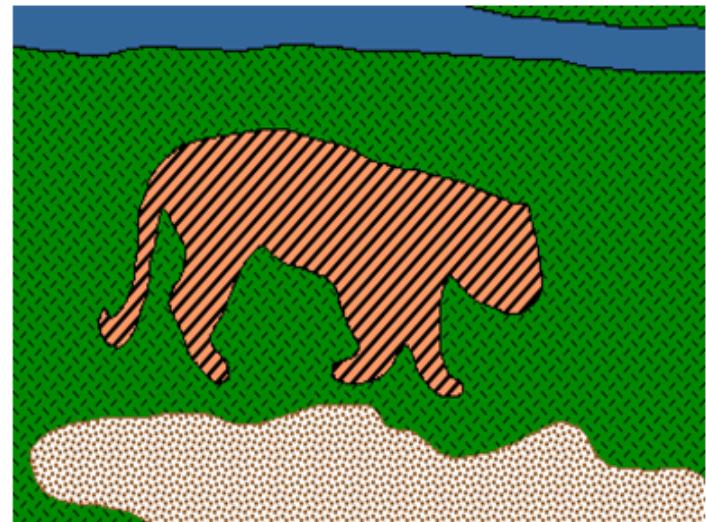
# IMAGE SEGMENTATION

# Image Segmentation

- Image analysis:
  - Extracting information from an image.
- First step:
  - To segment the image
  - i.e. to subdivide an image into its constituent regions or objects.

# Image Segmentation

- Dividing image into its constituent regions or objects



# Image Segmentation

- Edge Detection
- Edge Linking
- Boundary Detection
- Region growing
- Region splitting
- Region merging
- Split and Merge

# Image Segmentation

- Segmentation is based on two basic properties of gray-level values:
  - **Discontinuity**, i.e. to partition the image based on abrupt changes in intensity (gray levels), e.g. edges
  - **Similarity**, i.e. to partition the image into similar (according to predefined criteria) regions, e.g. thresholding, region growing, region splitting and merging

# Detection of Discontinuities

- 3 basic types of gray-level discontinuities:
  - Points
  - Lines
  - Edges
- Common method of detection: run a mask through the image.

# Image Segmentation

**FIGURE 10.1** A general  $3 \times 3$  mask.

---

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

# Point Detection

- T: nonnegative threshold:  $|R| \geq T$

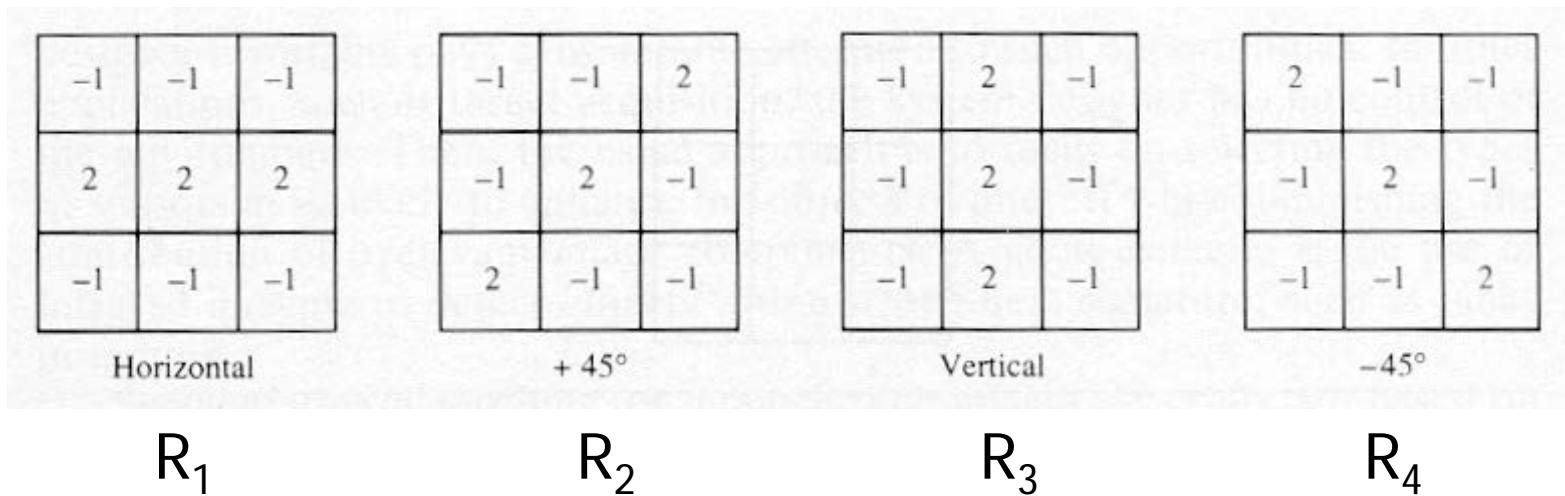
$$R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9 = \sum_{i=1}^9 w_i z_i$$

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

# Point Detection

- A point has been detected at the location on which the mask is centered if:  $|R| > T$
- The gray level of an isolated point will be quite different from the gray levels of its neighbors
  - measure the weighted differences between the center point and its neighbors

# Line Detection



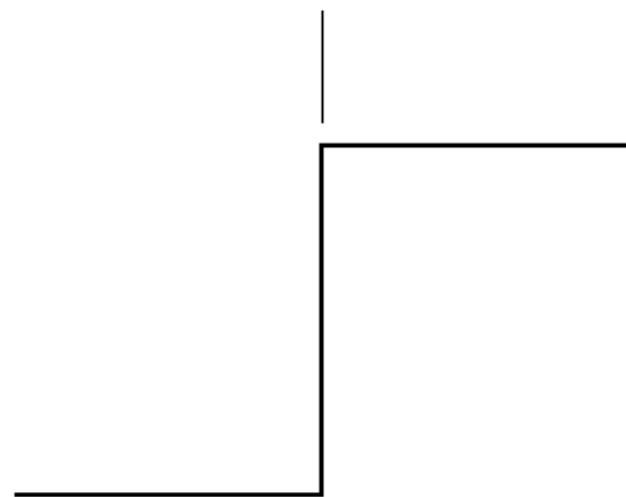
- If at a certain point  $|R_i| > |R_j|$ , this point is more likely associated with a line in the direction of mask i.

# Edge Detection

- Edge (a set of connected pixels):
  - the boundary between two regions with relatively distinct gray-level properties.
  - Note: edge vs. boundary
- Assumption:
  - the regions are sufficiently homogeneous, so that the transition between two regions can be determined on the basis of gray-level discontinuities alone.

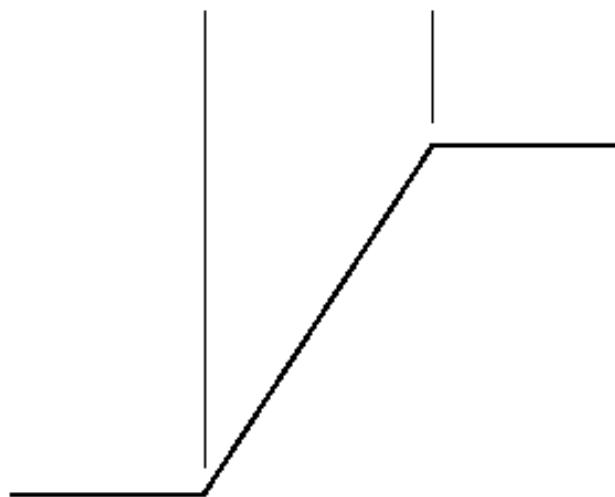
# Image Segmentation

Model of an ideal digital edge



Gray-level profile  
of a horizontal line  
through the image

Model of a ramp digital edge



Gray-level profile  
of a horizontal line  
through the image

a b

**FIGURE 10.5**

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

# Image Segmentation

# Edge Detection

- Basic Idea:
  - A profile is defined perpendicularly to the edge direction and the results are interpreted.
  - The magnitude of the first derivative is used to detect an edge (if a point is on a ramp)
  - The sign of the second derivative can determine whether an edge pixel is on the dark or light side of an edge.
- Remarks on second derivative:
  - It produces two responses for every edge
  - The line that can be formed joining its positive and negative values crosses zero at the mid point of the edge (zero-crossing)

# Edge Detection

- Computation of a local derivative operator
  - A profile is defined perpendicularly to the edge direction and the results are interpreted.
  - The **first derivative** is obtained by using the magnitude of the gradient at that point.
  - The **second derivative** is obtained by using the Laplacian.

# Gradient Operators

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

The gradient vector points in the direction of maximum rate of change of  $f$  at  $(x,y)$ .

# Gradient Operators

Gradient:  $\nabla f = \text{mag}(\nabla F) = [G_x^2 + G_y^2]^{1/2}$

(maximum rate of increase of  $f(x,y)$  per unit distance)

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

Direction angle of  $\nabla f$  at  $(x,y)$ :  $a(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$

# Image Segmentation

[www.rejinpaul.com](http://www.rejinpaul.com)

a
b c
d e
f g

**FIGURE 10.8**

A  $3 \times 3$  region of an image (the  $z$ 's are gray-level values) and various masks used to compute the gradient at point labeled  $z_5$ .

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

-1	0
0	1

0	-1
1	0

Roberts

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

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

Prewitt

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

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

# Image Segmentation

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 10.9** Prewitt and Sobel masks for detecting diagonal edges.

# Image Segmentation

[www.rejinpaul.com](http://www.rejinpaul.com)

a	b
c	d

**FIGURE 10.10**

- (a) Original image. (b)  $|G_x|$ , component of the gradient in the  $x$ -direction.  
(c)  $|G_y|$ , component in the  $y$ -direction.  
(d) Gradient image,  $|G_x| + |G_y|$ .



# Image Segmentation

[www.rejinpaul.com](http://www.rejinpaul.com)



a b  
c d

**FIGURE 10.11**

Same sequence as in Fig. 10.10, but with the original image smoothed with a  $5 \times 5$  averaging filter.



# Image Segmentation



a b

**FIGURE 10.12**  
Diagonal edge detection.  
(a) Result of using  
the mask in  
Fig. 10.9(c).  
(b) Result of using  
the mask in  
Fig. 10.9(d). The  
input in both cases  
was Fig. 10.11(a).

# Gradient Operators

- Computation of the gradient of an image:
  - Soebel operators provide both a differencing & a smoothing effect:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

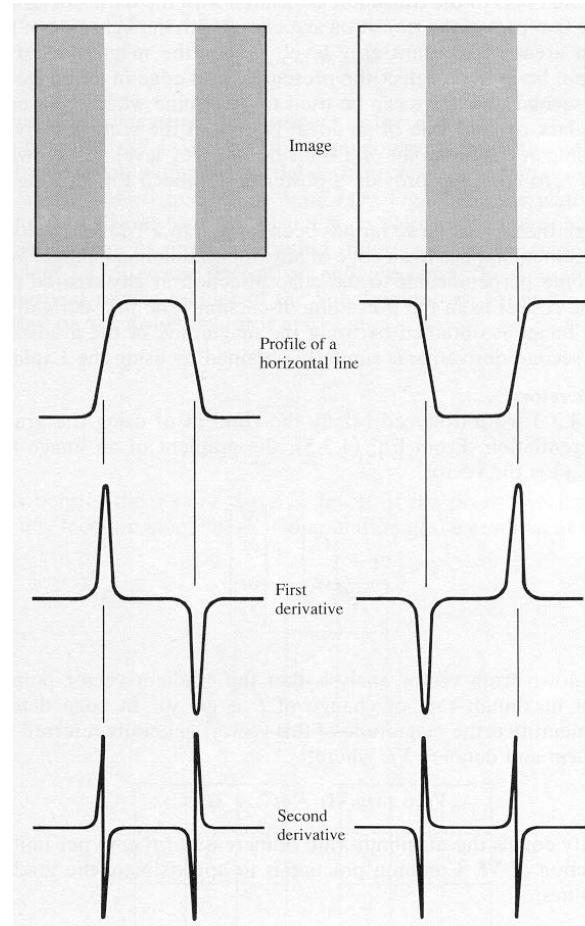
$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

# Summary: Gradient Operators

Smooth edges due to  
blurring (result of sampling) →

Positive: leading  
Negative: trailing  
Zero: in constant gray levels →

Positive: from dark side  
Negative: from light side  
Zero: in constant gray levels →



# Summary

- The magnitude of the first derivative detects the presence of an edge and the sign of the second detects whether the edge pixel lies on the dark or light side of an edge.
- The second derivative has a zero-crossing at the mid-point of a transition.

# Laplacian

- (of a 2-D function  $f(x,y)$ ):  $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- A  $3 \times 3$  discrete mask based on the above is:

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$

# Laplacian

- The idea:
  - Coefficient of center pixel should be positive
  - Coefficients of outer pixels should be negative
  - Sum of coefficients should be zero  
(the Laplacian is a derivative)

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

# Image Segmentation

**FIGURE 10.13**  
Laplacian masks  
used to  
implement  
Eqs. (10.1-14) and  
(10.1-15),  
respectively.

---

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

# Laplacian

- The Laplacian is seldom used in practice, because:
  - It is unacceptably sensitive to noise (as second-order derivative)
  - It produces double edges
  - It is unable to detect edge direction

# Laplacian

- An important use of the Laplacian:
  - To find the location of edges using its zero-crossings property.
- Plus, the Laplacian plays only the role of detector of whether a pixel is on the dark or light side of an edge.

# Laplacian

- Convolve an image with the Laplacian of a 2D Gaussian function of the form:

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

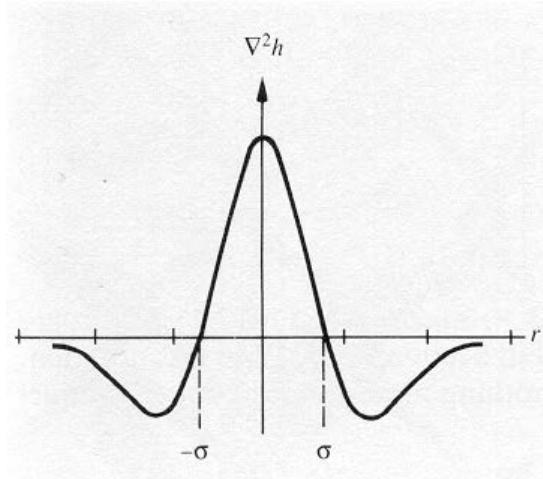
where  $\sigma$  is the standard deviation.

# Laplacian

- The Laplacian of the above Gaussian is:

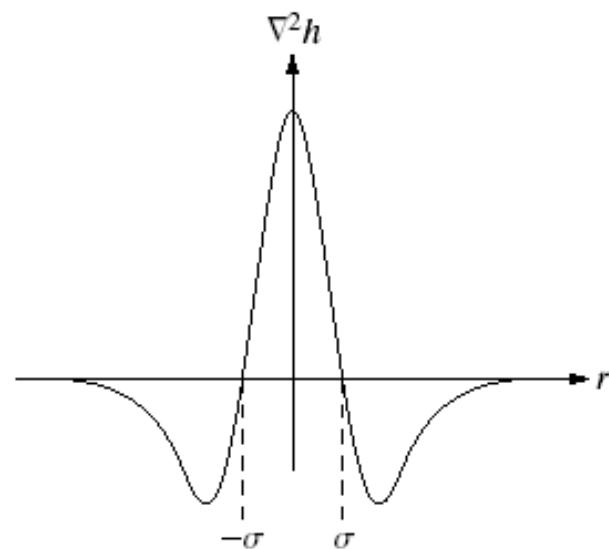
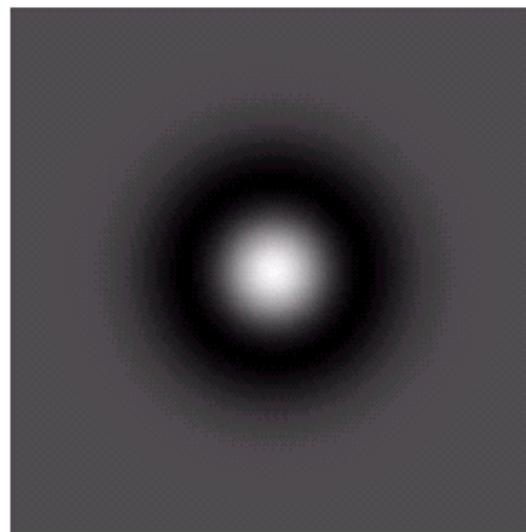
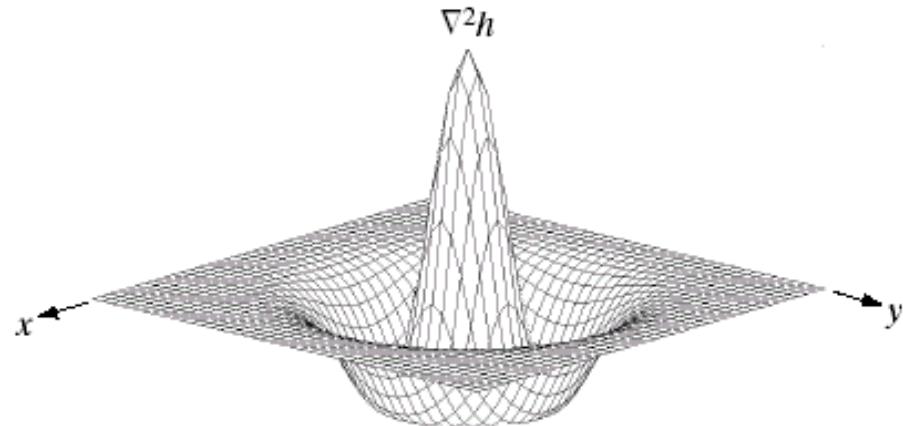
$$\nabla^2 h = -\left(\frac{r^2 - \sigma^2}{\sigma^4}\right) \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

where  $r^2 = x^2 + y^2$ .



$\sigma$  determines the degree of blurring that occurs.

# Image Segmentation



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

a  
b  
c  
d

**FIGURE 10.14**  
Laplacian of a Gaussian (LoG).  
(a) 3-D plot.  
(b) Image (black is negative, gray is the zero plane, and white is positive).  
(c) Cross section showing zero crossings.  
(d)  $5 \times 5$  mask approximation to the shape of (a).

# Image Segmentation

[www.rejinpaul.com](http://www.rejinpaul.com)

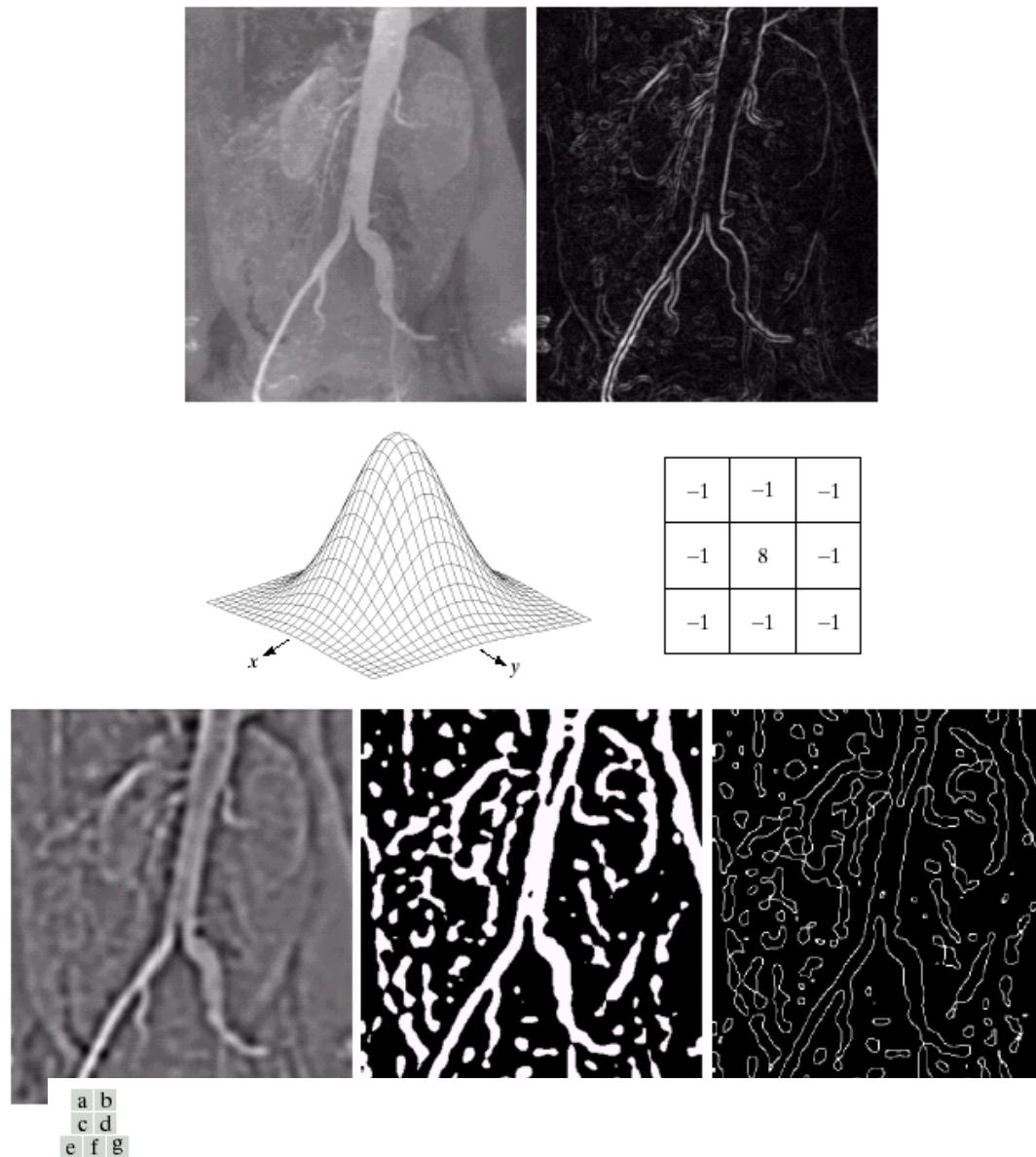


FIGURE 10.15 (a) Original image. (b) Sobel gradient (shown for comparison). (c) Spatial Gaussian smoothing function. (d) Laplacian mask. (e) Log-G. (f) Thresholded Log-G. (g) Zero crossings. (Original image courtesy of Dr. David R. Fickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

# Region-Oriented Segmentation

- Segmentation is a process that partitions  $R$  into  $n$  subregions  $R_1, R_2, \dots, R_n$  such that:

$$- \bigcup_{i=1}^n R_i = R$$

-  $R_i$  is a connected region,  $i = 1, 2, \dots, n$

-  $R_i \cap R_j = \emptyset$  for all  $i$  and  $j$ ,  $i \neq j$

-  $P(R_i) = \text{TRUE}$  for  $i = 1, 2, \dots, n$   
-  $P(R_i \cup R_j) = \text{FALSE}$  for  $i \neq j$

$\left. \begin{array}{l} P(R_i) : \text{logical predicate} \end{array} \right\}$

# Region Growing by Pixel Aggregation

- Start with a set of “seed” points and from these grow regions by appending to each seed point those neighboring pixels that have similar properties.

# Region Growing by Pixel Aggregation

- Problems:
  - Seed selection
  - Selection of suitable properties for including points in the various regions
- Descriptors
- Local vs. general criteria

# Region Splitting and Merging

- Subdivide an image initially into a set of arbitrary, disjointed regions and then merge and/or split the regions in an attempt to satisfy the conditions of region-oriented segmentation.
- Quadtree-based algorithm

# Region Splitting and Merging

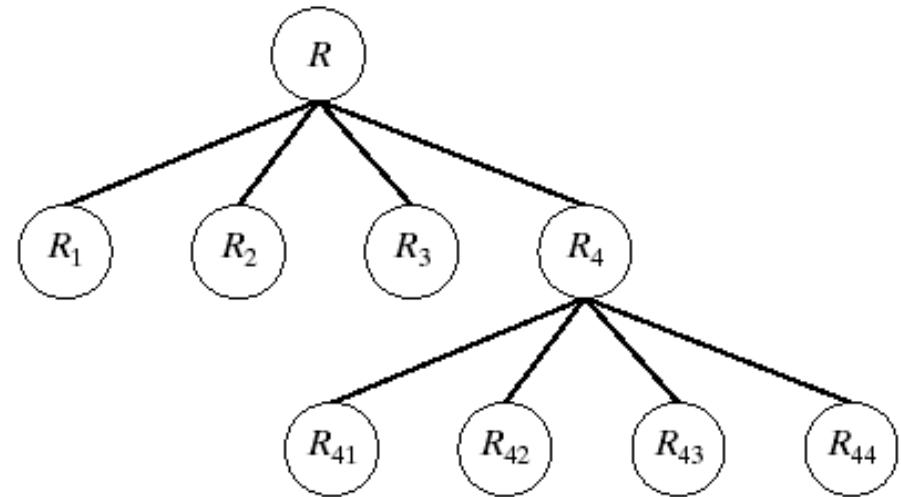
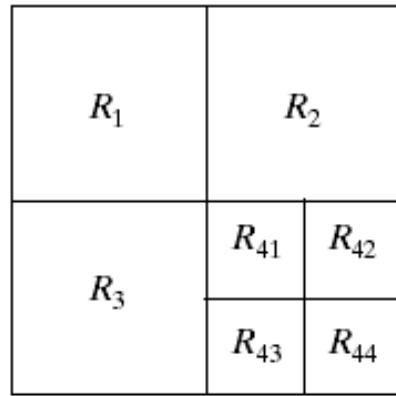
- Procedure:
  - Split into 4 disjointed quadrants any region  $R_i$  where  $P(R_i) = \text{FALSE}$
  - Merge any adjacent regions  $R_j$  and  $R_k$  for which  $P(R_j \cup R_k) = \text{TRUE}$
  - Stop when no further splitting or merging is possible.

# Image Segmentation

a b

**FIGURE 10.42**

- (a) Partitioned image.  
(b) Corresponding quadtree.



# IMAGE COMPRESSION

- **Definition:** Compression means storing data in a format that requires less space than usual.
- Data compression is particularly useful in communications because it enables devices to transmit the same amount of data in fewer bits.

- The bandwidth of a digital communication link can be effectively increased by compressing data at the sending end and decompressing data at the receiving end.
- There are a variety of data compression techniques, but only a few have been standardized.

# Types of Data Compression

- There are two main types of data compression :  
Lossy and Lossless.
- In Lossy data compression the message can never be recovered exactly as it was before it was compressed.

- In a Lossless data compression file the original message can be exactly decoded.
  
  
  
  
  
  
  
  
- Lossless compression is ideal for text.
  
  
  
  
  
  
  
  
- Huffman coding is type of lossless data compression.

# Compression Algorithms

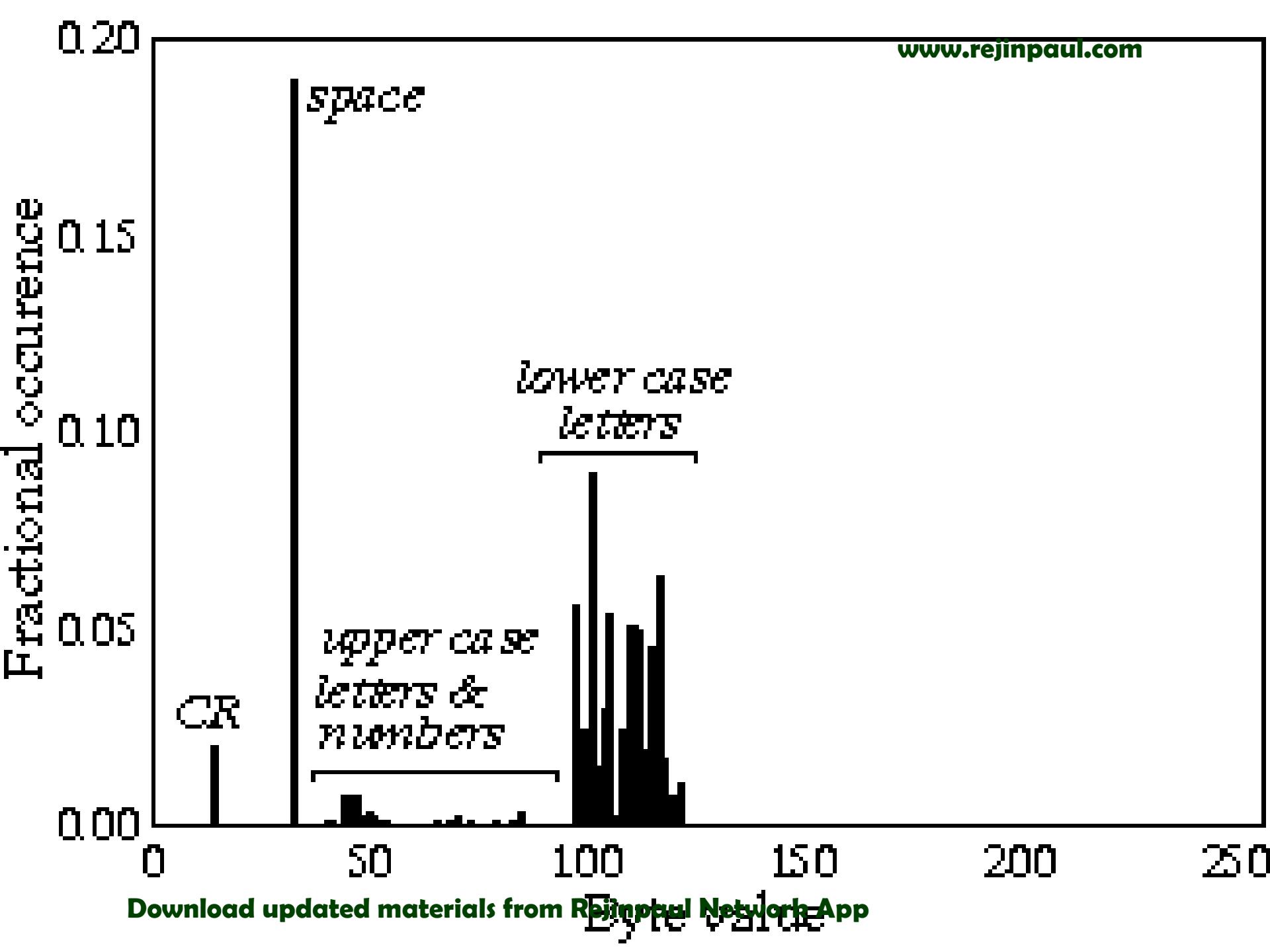
- Huffman Coding
- Run Length Encoding
- Shift Codes
- Arithmetic Codes
- Block Truncation Codes
- Transform codes
- Vector Quantization

# Huffman Coding

- Huffman coding is a popular compression technique that assigns variable length codes (VLC) to symbols, so that the most frequently occurring symbols have the shortest codes.
- On decompression the symbols are reassigned their original fixed length codes.

- The idea is to use short bit strings to represent the most frequently used characters
- and to use longer bit strings to represent less frequently used characters.

- That is, the most common characters, usually space, e, and t are assigned the shortest codes.
  
- In this way the total number of bits required to transmit the data can be considerably less than the number required if the fixed length ASCII representation is used.
  
- A Huffman code is a binary tree with branches assigned the value 0 or 1.



# Huffman Algorithm

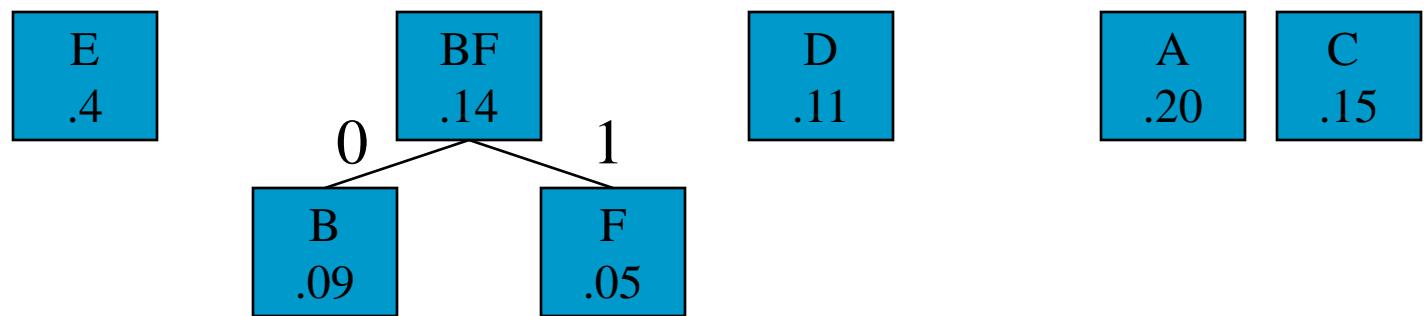
- To each character, associate a binary tree consisting of just one node.
- To each tree, assign the character's frequency, which is called the tree's weight.
- Look for the two lightest-weight trees. If there are more than two, choose among them randomly.

- Merge the two into a single tree with a new root node whose left and right sub trees are the two we chose.
  
- Assign the sum of weights of the merged trees as the weight of the new tree.
  
- Repeat the previous step until just one tree is left.

# Huffman Coding Example

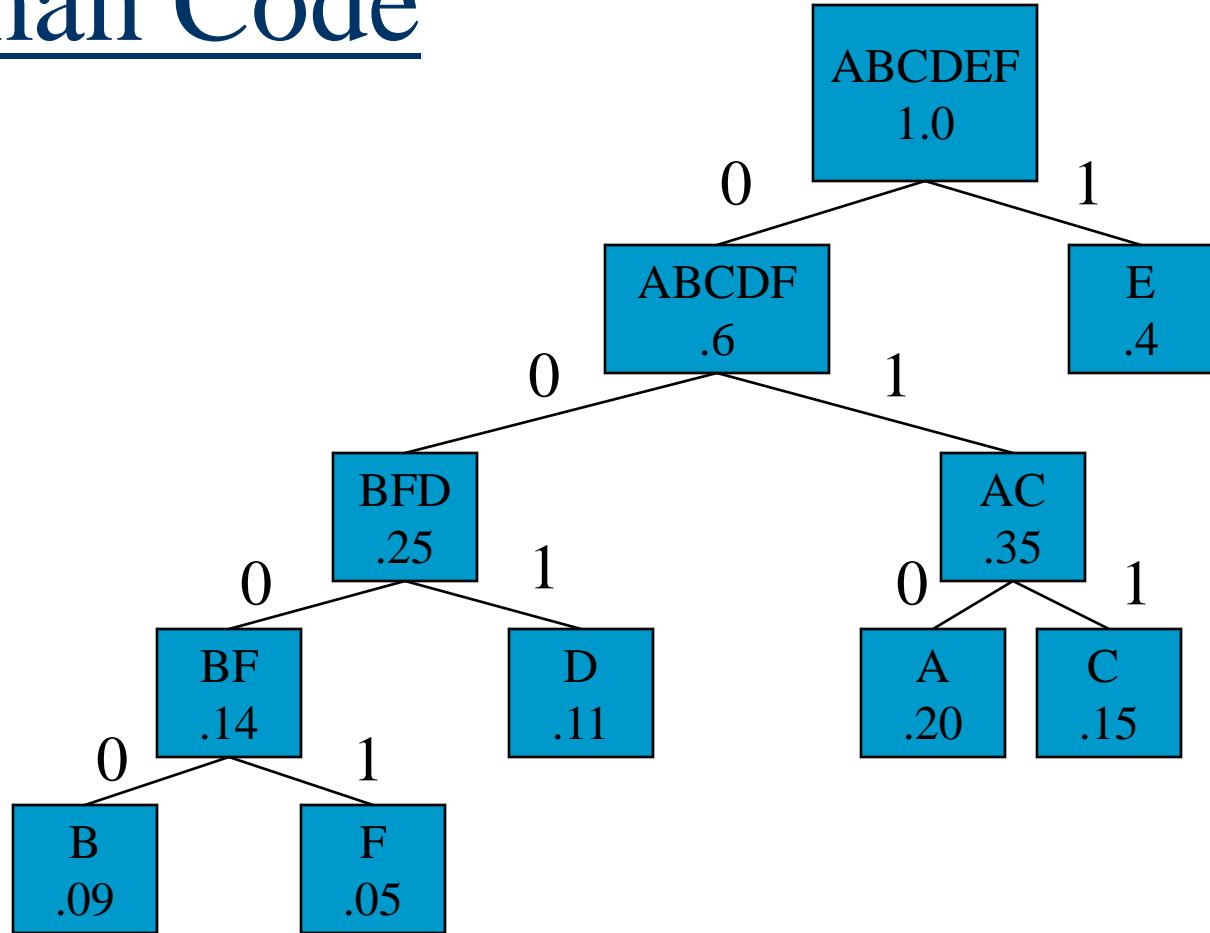
- **Character frequencies**
  - A: 20% (.20)
  - B: 9% (.09)
  - C: 15%
  - D: 11%
  - E: 40%
  - F: 5%
  
- **No other characters in the document**

# Huffman Code



# Huffman Code

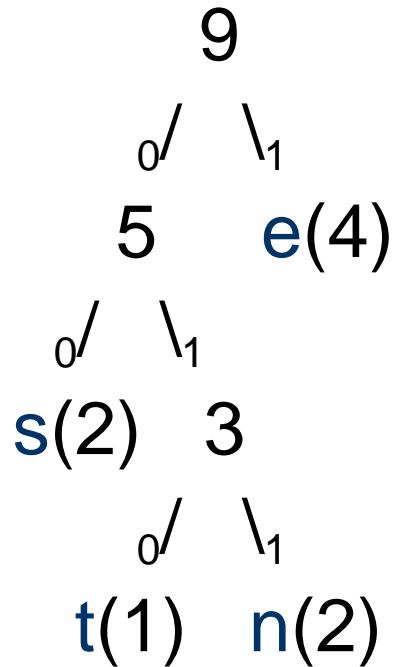
- Codes
  - A: 010
  - B: 0000
  - C: 011
  - D: 001
  - E: 1
  - F: 0001



- Note
  - None are prefixes of another

# Huffman Coding

- TENNESSEE



- ENCODING

- E : 1
- S : 00
- T : 010
- N : 011

$$\text{Average code length} = (1*4 + 2*2 + 3*2 + 3*1) / 9 = 1.89$$

# Average Code Length

Average code length =

$$\sum_{i=0,n} (\text{length} * \text{frequency}) / \sum_{i=0,n} \text{frequency}$$

$$= \{ 1(4) + 2(2) + 3(2) + 3(1) \} / (4+2+2+1)$$

$$= 17 / 9 = 1.89$$

# ENTROPY

[www.rejinpaul.com](http://www.rejinpaul.com)

Entropy is a measure of information content:  
the more probable the message, the lower its  
information content, the lower its entropy

$$\text{Entropy} = -\sum_{i=1,n} (p_i \log_2 p_i)$$

**( p - probability of the symbol)**

$$\begin{aligned} &= - ( 0.44 * \log_2 0.44 + 0.22 * \log_2 0.22 \\ &\quad + 0.22 * \log_2 0.22 + 0.11 * \log_2 0.11 ) \\ &= - ( 0.44 * \log 0.44 + 2(0.22 * \log 0.22 + 0.11 * \log 0.11) \\ &\quad / \log 2 \\ &= 1.8367 \end{aligned}$$

# Advantages & Disadvantages

- The problem with Huffman coding is that it uses an integral number of bits in each code.
  
- If the entropy of a given character is 2.5 bits, the Huffman code for that character must be either 2 or 3 bits , not 2.5.

- Though Huffman coding is inefficient due to using an integral number of bits per code , it is relatively easy to implement and very efficient for coding and decoding.
- It provides the best approximation for coding symbols when using fixed width codes.

# Run-length encoding

- Run-length encoding (RLE) is a very simple form of data compression encoding.
- RLE is a lossless type of compression
- It is based on simple principle of encoding data. This principle is to every stream which is formed of the same data values (repeating values is called a *run*) i.e sequence of repeated data values is replaced with count number and a single value.

- This intuitive principle works best on certain data types in which sequences of repeated data values can be noticed;
- RLE is usually applied to the files that contain large number of consecutive occurrences of the same byte pattern.

- RLE may be used on any kind of data regardless of its content, but data which is being compressed by RLE determines how good compression ratio will be achieved.
  
- RLE is used on text files which contains multiple spaces for indentation and formatting paragraphs, tables and charts.
  
- Digitized signals also consist of unchanged streams so such signals can also be compressed by RLE.
  
- A good example of such signal are monochrome images, and questionable compression would be probably achieved if such compression was used on continuous-tone (photographic) images.

- Fair compression ratio may be achieved if RLE is applied on computer generated color images.
  
- RLE is a lossless type of compression and cannot achieve great compression ratios,
  
- but a good point of that compression is that it can be easily implemented and quickly executed.

# Example1

- WWWWWWWWWWWWWWWBWWWWWWWWWWWWWWB  
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW  
BWWWWWWWWWWWWWWWW

- If we apply a simple run-length code to the above hypothetical scan line, we get the following:

12WB12W3B24WB14W

## **Shift code:**

**A shift code is generated by**

- Arranging the source symbols so that their probabilities are monotonically decreasing
- Dividing the total number of symbols into symbol blocks of equal size.
- Coding the individual elements within all blocks identically, and
- Adding special shift-up or shift-down symbols to identify each block. Each time a shift-up or shift-down symbol is recognized at the decoder, it moves one block up or down with respect to a pre-defined reference block.

# Arithmetic coding

- Unlike the variable-length codes described previously, *arithmetic coding*, generates non-block codes.
- In arithmetic coding, a one-to-one correspondence between source symbols and code words does not exist.
- Instead, an entire sequence of source symbols (or message) is assigned a single arithmetic code word.
- Arithmetic coding, is entropy coder widely used, the only problem is it's speed, but compression tends to be better than can achieve

- The code word itself defines an interval of real numbers between 0 and 1
- As the number of symbols in the message increases, the interval used to represent it becomes smaller and the number of information units (say, bits) required to represent the interval becomes larger
- Each symbol of the message reduces the size of the interval in accordance with the probability of occurrence.
- It is suppose to approach the limit set by entropy.

- The idea behind arithmetic coding is to have a probability line, 0-1
- assign to every symbol a range in this line based on its probability
- higher the probability, the higher range which assigns to it.
- Once we have defined the ranges and the probability line, start to encode symbols
- every symbol defines where the output floating point number lands

## Example

Symbol	Probability	Range
a	2	[0.0 , 0.5)
b	1	[0.5 , 0.75)
c	1	[0.75 , 1.0)

## *Algorithm to compute the output number*

- Low = 0
- High = 1
- Loop. For all the symbols.

Range = high - low

High = low + range \* high\_range of  
the symbol being  
coded

Low = low + range \* low\_range of the symbol  
being

Symbol	Range	Low value	High value
		0	1
b	1	0.5	0.75
a	0.25	0.5	0.625
c	0.125	0.59375	0.625
a	0.03125	0.59375	0.609375

The output number will be 0.59375

# Arithmetic coding

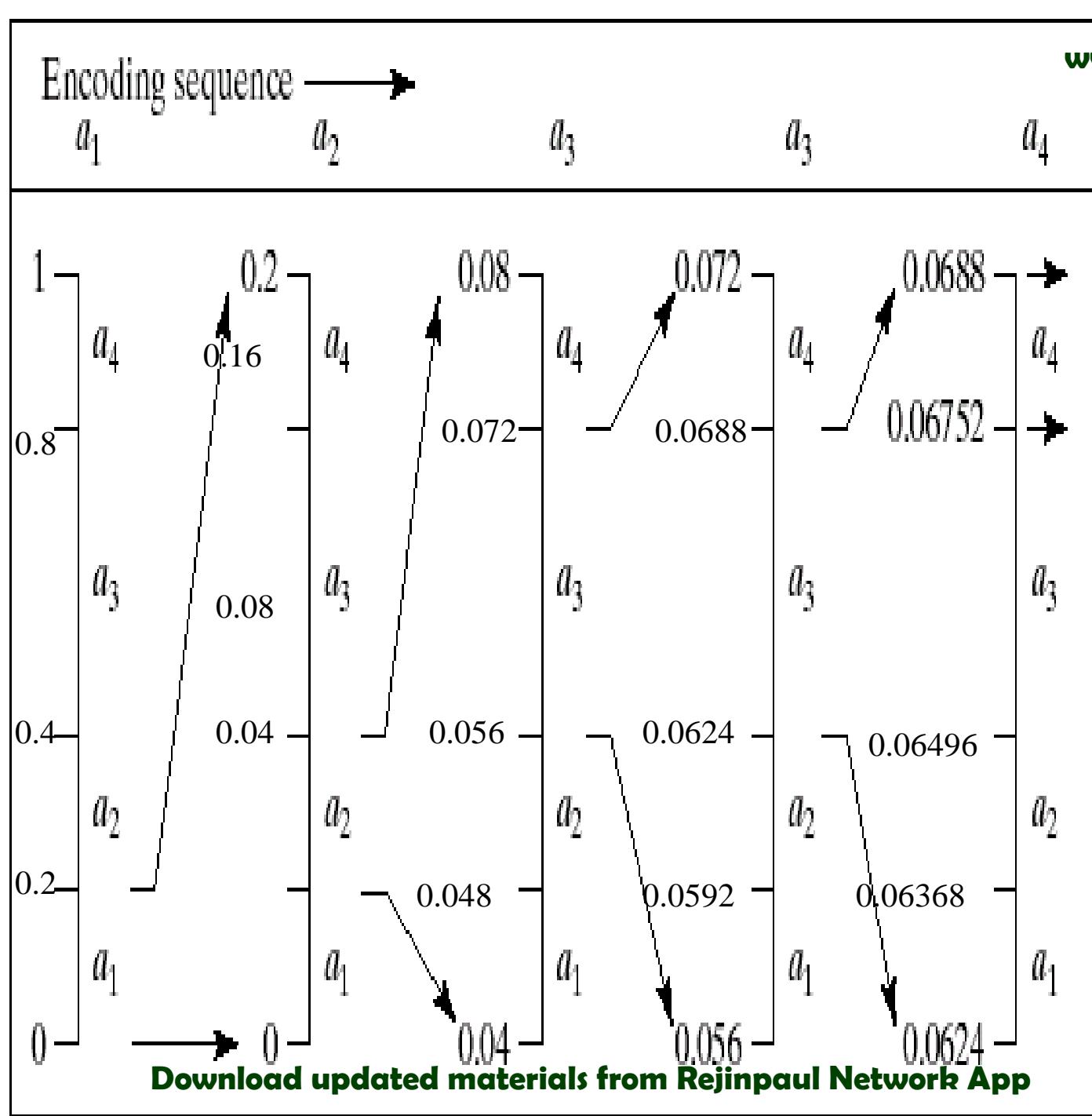
[www.rejinpaul.com](http://www.rejinpaul.com)

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

Let the message to be encoded be  $a_1a_2a_3a_3a_4$

Arithmetic coding procedure.



So, any number in the interval  $[0.06752, 0.0688)$ , for example 0.068 can be used to represent the message.

Decode 0.39.

Since  $0.8 > \text{code word} > 0.4$ , the first symbol should be  $a_3$ .

