

Digital Image Processing

Digital Image Processing Using Matlab

Digital Image Processing

Prepared by:

Dr. Ali J. Abboud

University of Diyala

2012-2013



Key Features of Chapter 5:

- **A Model of Image Degredation & Restoration.**
- **Noise Models .**
- **Statistical Models of Spatial Noise .**
- **Noise Models – Probability Density Functions.**
- **Image restoration in the presesncce of Noise Only .**
- **Adaptive filters .**

A Model of Image Degredation & Restoration

- ✓ *Image Degredation is a process that operates on an “input” image $f(x,y)$ during the process of acquisition and/or transmission.*
- ✓ *It is assumed that it is the result of a **degredation function** $H(u,v)$ plus an additive noise $N(u,v)$ acting on the frequency domain:*

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

- ✓ *The **observed** degraded image $g(x,y)$, can be modelled as*

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

where $h(x,y)$, $\eta(x,y)$, and $g(x,y)$ are the inverse DFT of $H(u,v)$, $N(u,v)$, and $G(u,v)$, respectively.

- ✓ *The process of restoration aims to recover $f(x,y)$ from the observed degraded image $g(x,y)$. Restoration requires a realistic model of noise to be removed first, and then filtering out $H(u,v)$.*

Noise Models

- ✓ *Noise in digital images arise during*
 - ✓ *Aquuasition: **environmental conditions (light level & sensor temperature)***
 - ✓ *and/or transmission – interference in the transmtion channel.*
- ✓ *To remove noise we need to understand the spatial Characteristics of noise and its frequency characteristics (Fourier spectrum).*
- ✓ *Generally, noise level is assumed to be independent of position in image and uncorrelated to pixel values, i.e. we cannot say that noise effects some positions or pixels values more or less than other even if this is so visibly.*
- ✓ *White noise refers to noise function with constant Fourier spectrum*
- ✓ *Spatial noise is described by the statistical behavior of the gray-level values in the noise component of the degraded image.*

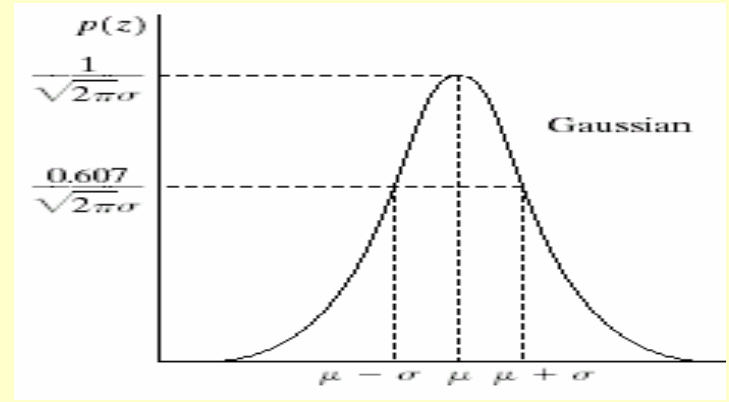
Statistical Models of Spatial Noise

- ✓ *Spatial noise can be modelled in terms of the statistical behavior of the gray-level values in the noise component of the degraded image, i.e. a random variable with a specific probability distribution.*
- ✓ *Important examples of noise models include:*
 1. *Gaussian Noise.*
 2. *Rayleigh Noise.*
 3. *Gamma Noise.*
 4. *Exponential Noise.*
 5. *Uniform Noise.*
 6. *Impulse Noise (Salt & Pepper).*

Noise Models – Probability Density Functions & graphs

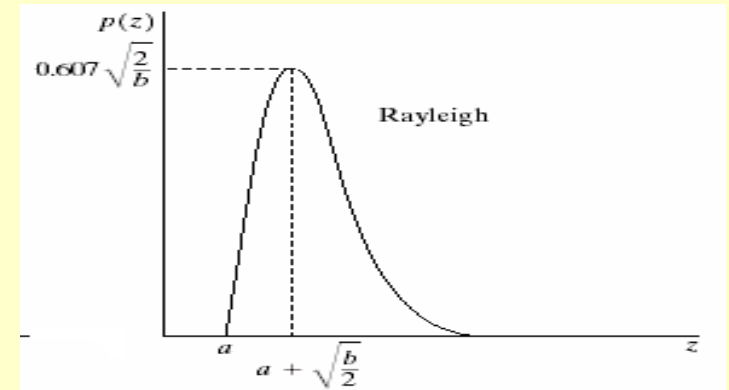
Gaussian :
$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2},$$

where z is the gray - value , μ is the mean and σ is the standard deviation.



Rayleigh :

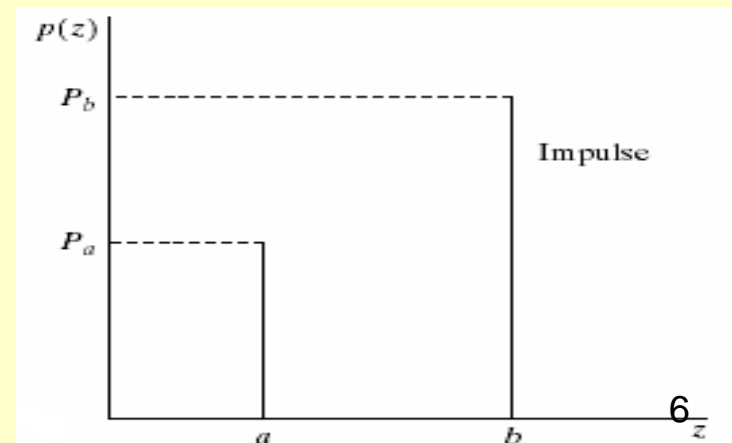
$$p(z) = \begin{cases} (2/b)(z-a)e^{-(z-a)^2/b} & \text{for } z \geq a \\ 0 & \text{for } z < a. \end{cases}$$



Impulse(Salt and Pepper)

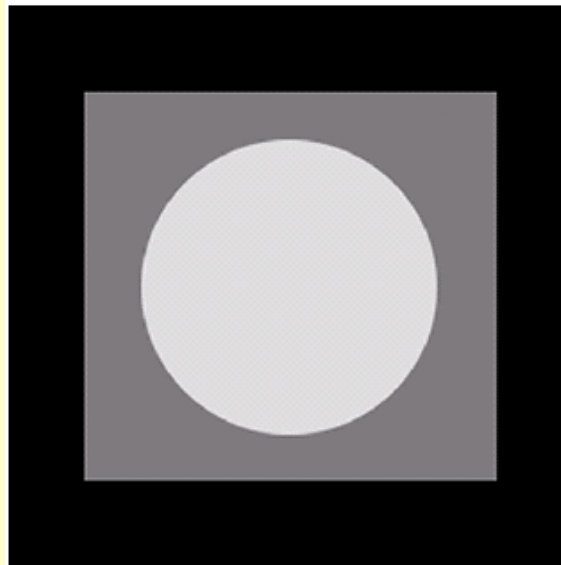
$$p(z) = \begin{cases} P_a & \text{if } z = a \\ P_b & \text{if } z = b \\ 0 & \text{otherwise} \end{cases}.$$

If $b > a$, then gray - level b appears as a light dot (salt), otherwise gray - level a appears as a dark dot (Pepper).

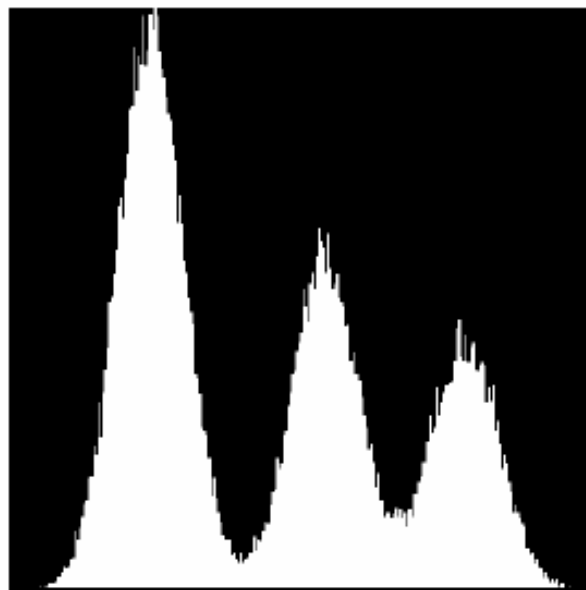
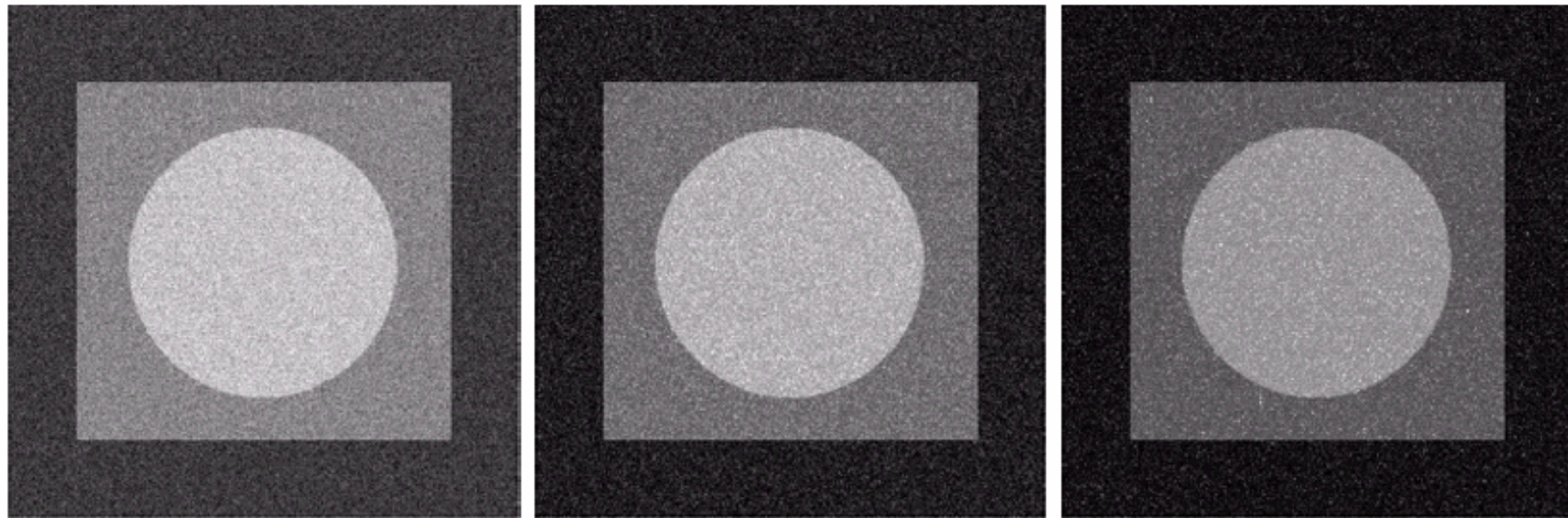


Determining noise models

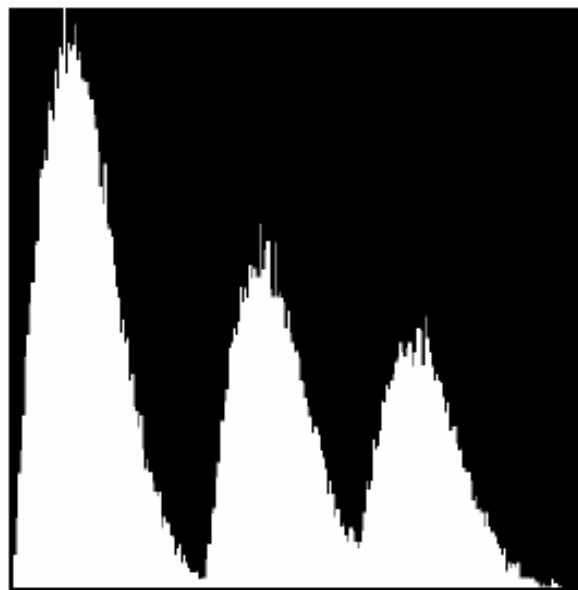
- ✓ *To determine the noise model in a noisy image, one may select a relatively small rectangular sub-image of relatively smooth region. The histogram of the sub-image approximates the density probability distribution of the corrupting model of noise.*
- ✓ *The simple image below is well-suited test pattern for illustrating the effect of adding noise of the various models.*



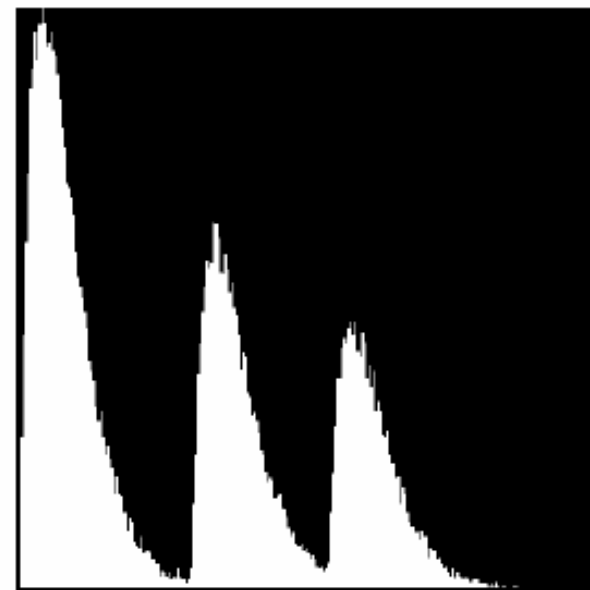
Addition of noise - Examples



Gaussian

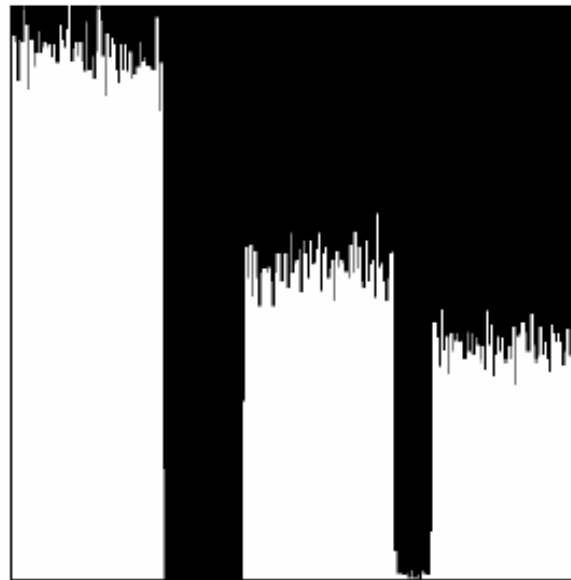
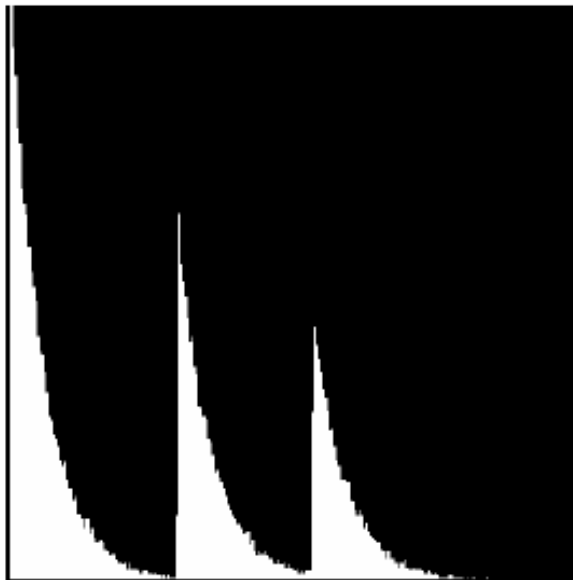
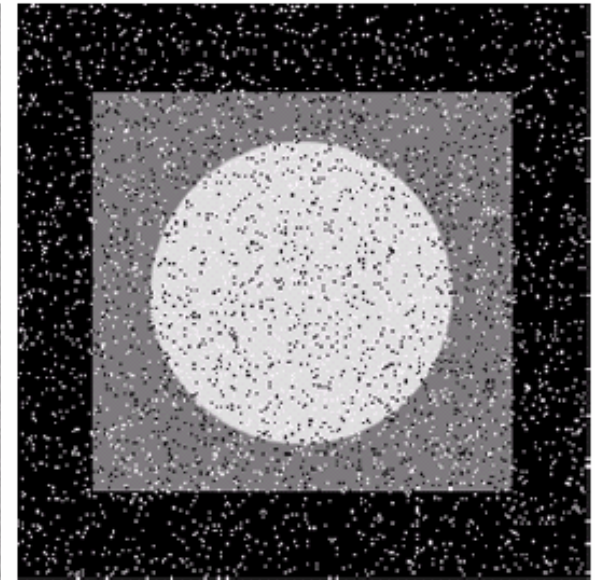
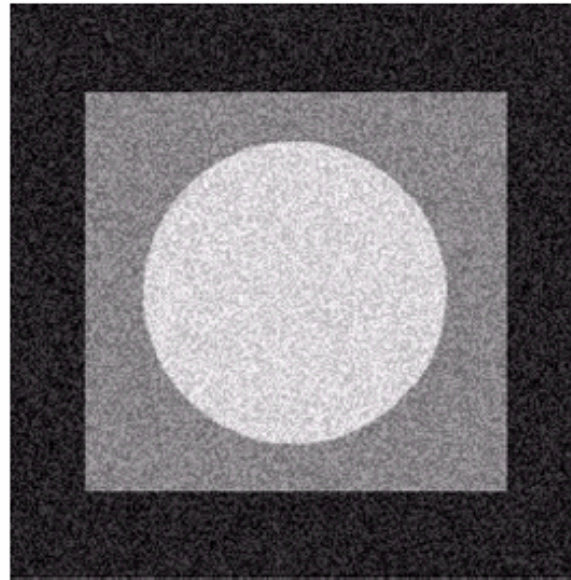
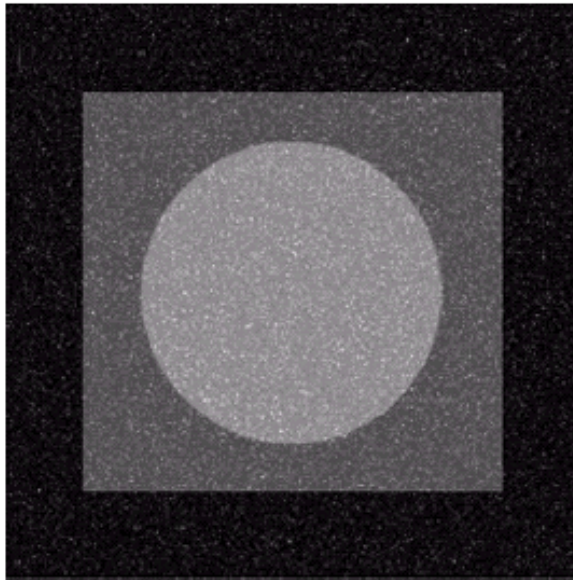


Rayleigh



Gamma

Addition of noise - Examples



Exponential

Uniform

Salt & Pepper

Adding Noise with MATLAB

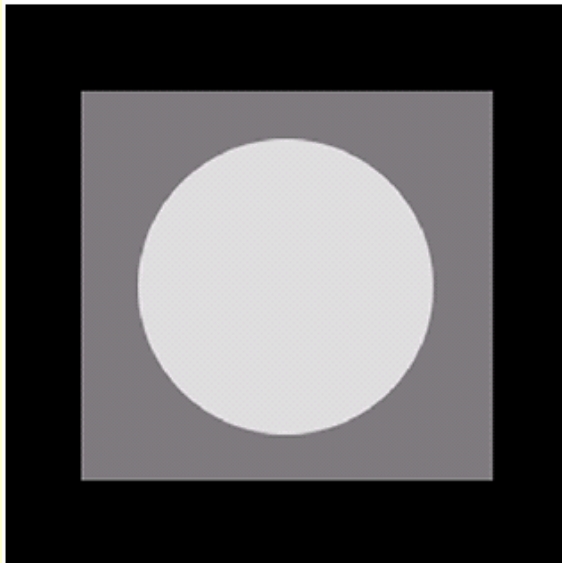
- ✓ *MATLAB provides special instructions to add noise to an image*

imnoise (f, type, parameters)

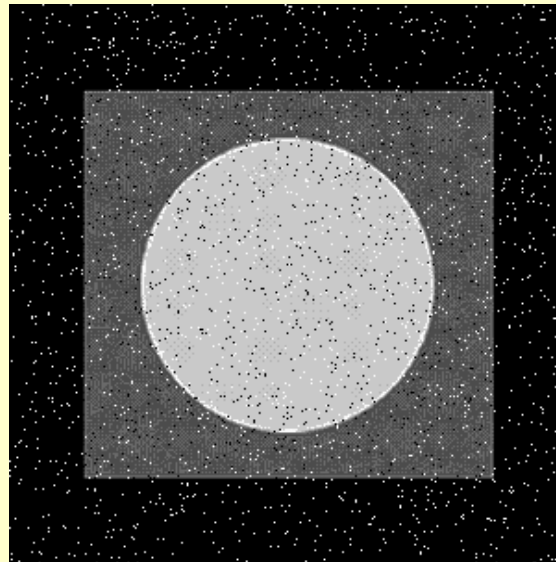
e.g. imnoise (f, 'gaussian', m, var), and

imnoise (f, 'salt & pepper', d)

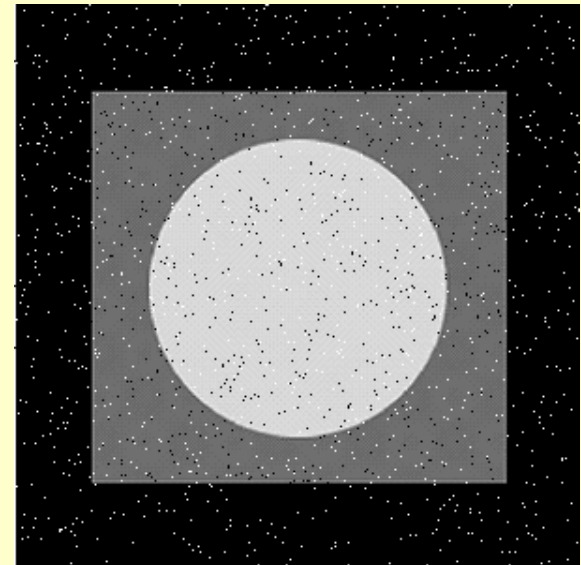
- ✓ *The left-most image below has been corrupted with “Salt & Pepper” noise with different densities.*



Original Clean image



Salt&pepper, d=0.05



Salt&pepper, d=0.025 ¹⁰

Image restoration in the presencce of Noise Only

- ✓ *Here we assume that $H(u,v)=1$, i.e. the degradation is modelled as:*

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

- ✓ *Spatial filters have been designed to remove noise include:*

1. Mean filters – Arithmetic, Geometric, Harmonic, & Contraharmonic

2. Order Statistics Filter – Median, Max, min, Midpoint

And

3. Adaptive Filters.

- ✓ *These filters are applied in the same way as before using a square array neighbourhood of size 3×3 , 5×5 , ...etc.*
- ✓ *The resulting image approximates the perceived clean image $f(x,y)$.*

Types of Mean Filters

1. Arithmetic Mean :

$$\hat{f}(x, y) = \frac{1}{mn} \sum g(x, y).$$

The sum is taken over all pixels in the mask.

2. Geometric Mean :

$$\hat{f}(x, y) = \left[\prod g(s, t) \right]^{1/mn}$$

The product is taken over all pixels in the mask.

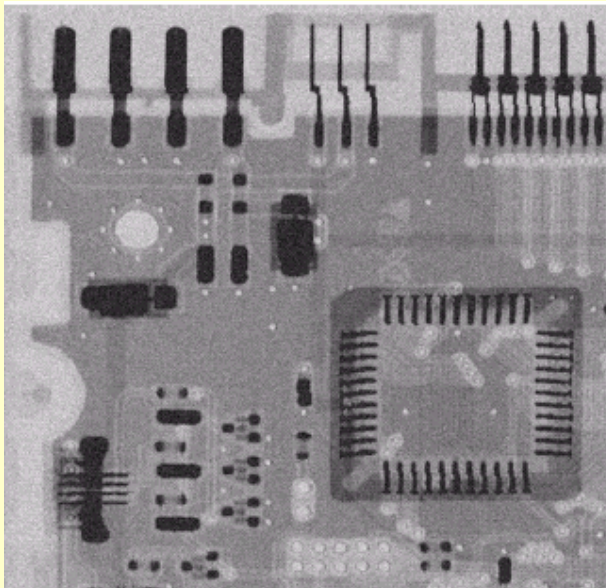


Image corrupted by adding Gaussian noise

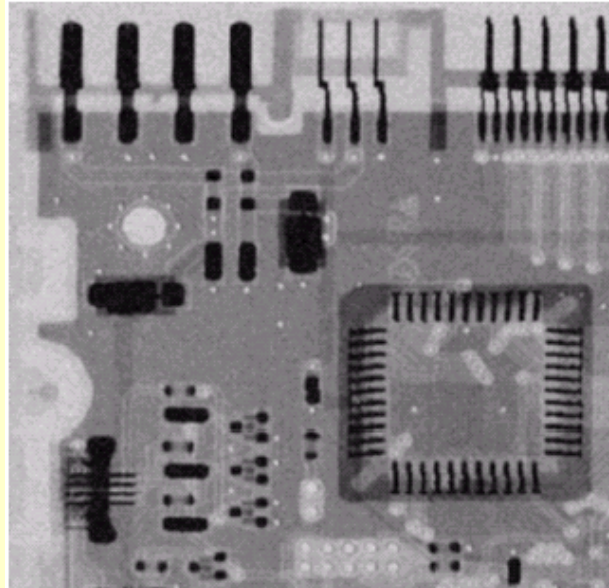


Image restored with arithmetic mean filter

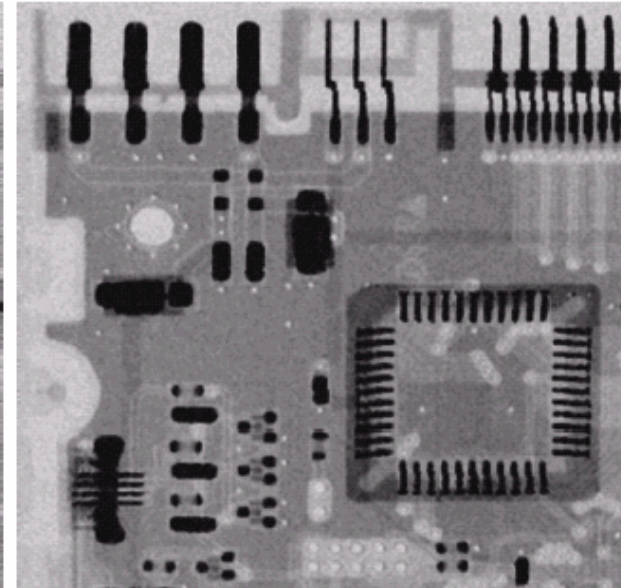


Image restored with Geometric mean filter

Types of Mean Filters - continued

3. Harmonic Mean :

$$\hat{f}(x, y) = \frac{mn}{\sum \frac{1}{g(x, y)}}.$$

The sum is taken over all pixels in the mask.

Works well for salt noise removal

4. Contraharmonic Mean :

$$\hat{f}(x, y) = \frac{\sum g(x, y)^{Q+1}}{\sum g(x, y)^Q}.$$

Q is the order. Positive Q eliminates pepper, while negative Q eliminate salt.

Note that the harmonic filter is a special case of contraharmonic with Q=-1, and contraharmonic with Q=0 is the arithmetic mean filter.

Effect of Mean Filters

Image (a)
corrupted by
pepper with
probability 0.1

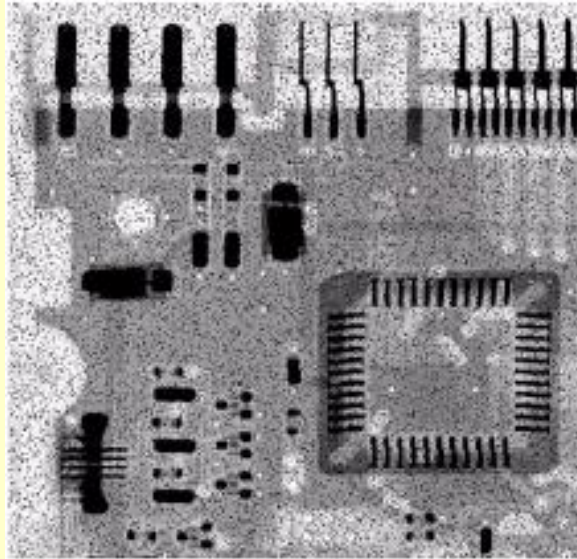


Image (b)
corrupted by
Salt with
probability 0.1

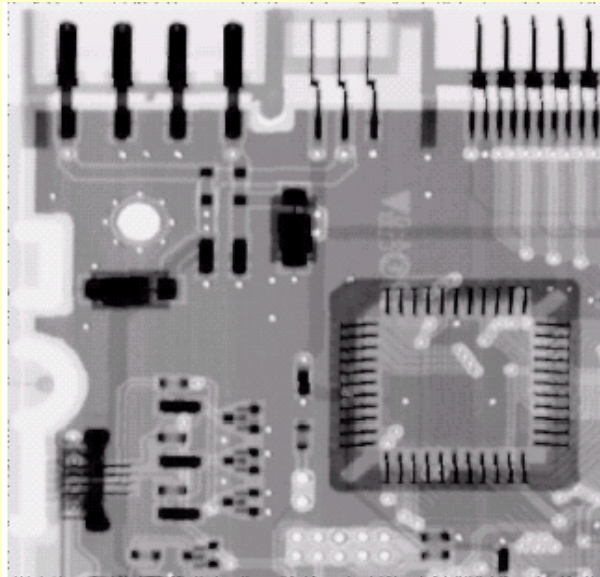
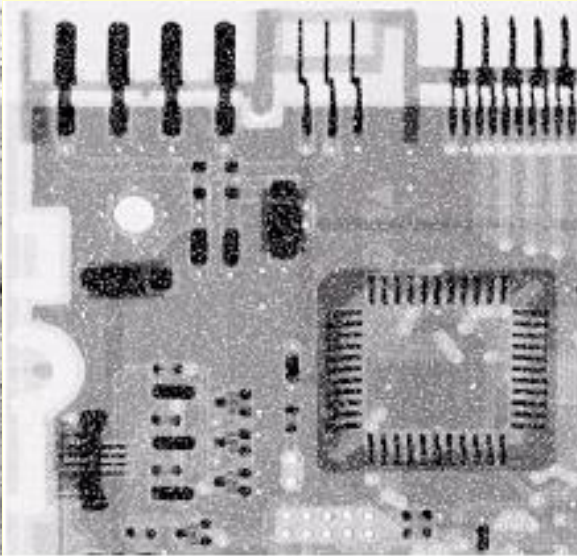


Image (a) restored with
Contraharmonic filter, $Q=1.5$

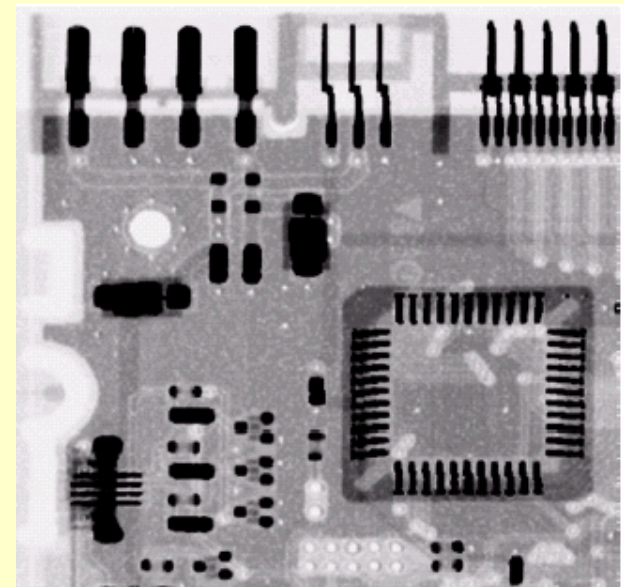


Image (b) restored with
Contraharmonic filter, $Q=-1.5$

Effect of Contraharmonic Filters with wrong sign

Image (a)
corrupted by
pepper with
probability 0.1

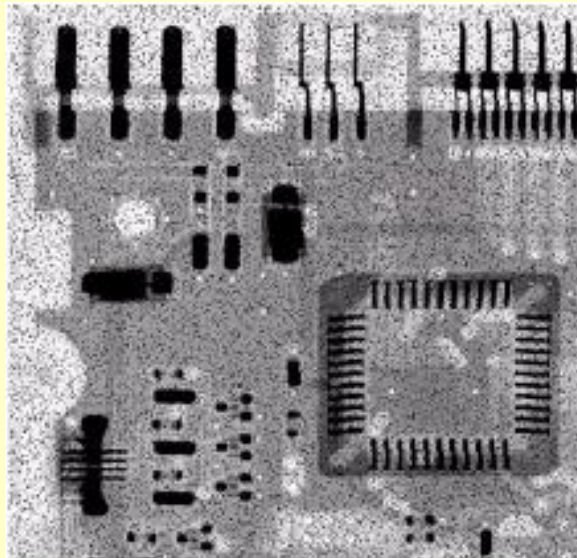


Image (b)
corrupted by
Salt with
probability 0.1

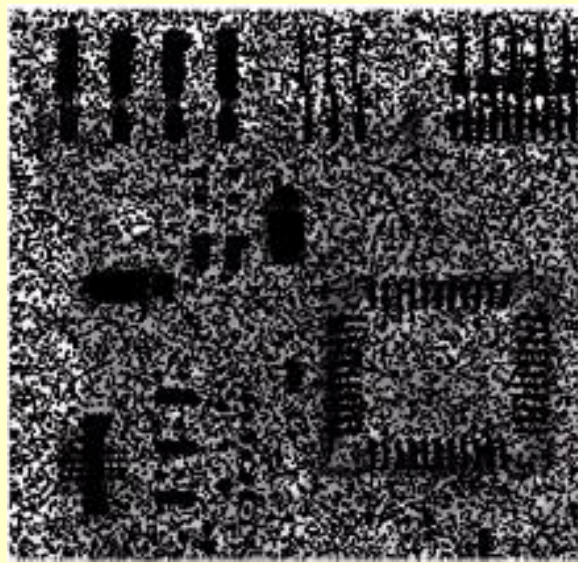
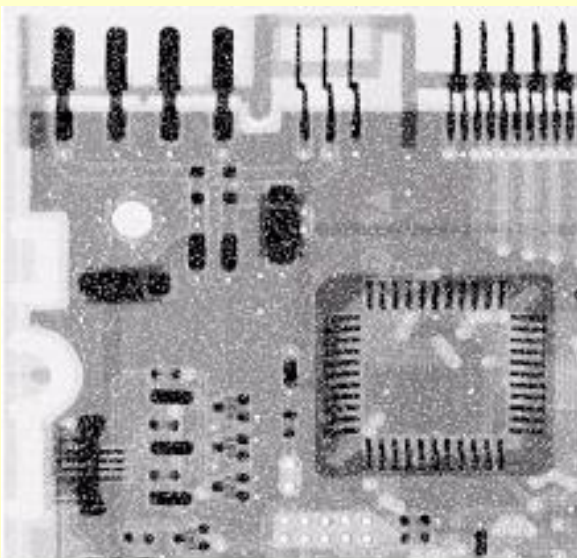


Image (a) with application of
Contraharmonic filter, $Q = -1.5$

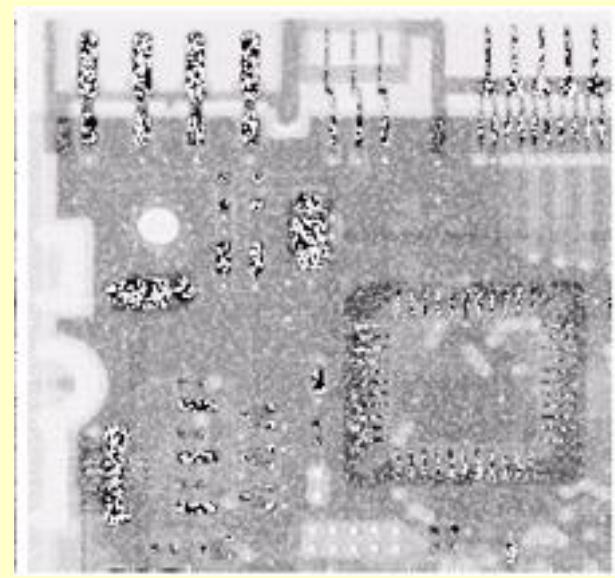


Image (b) with application of
Contraharmonic filter, $Q = 1.5$

Salt noise and Pepper noise Removal

Image corrupted by pepper noise with probability 0.1

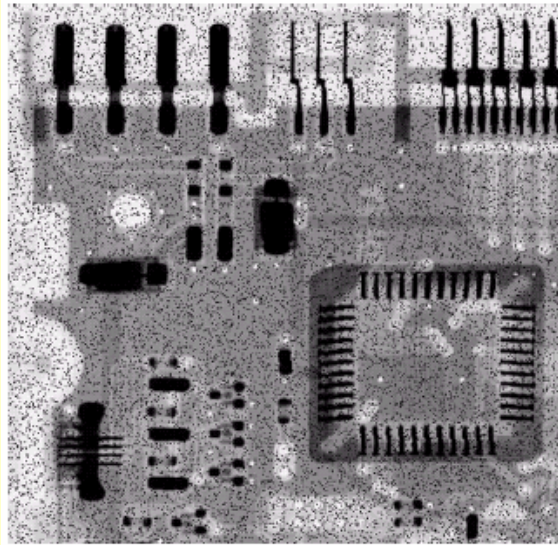
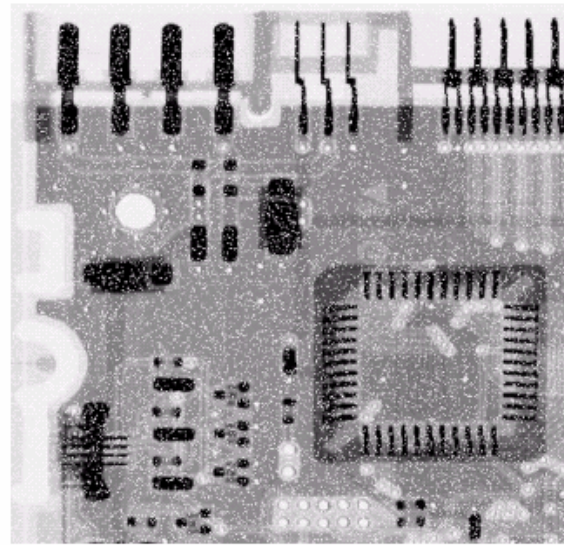
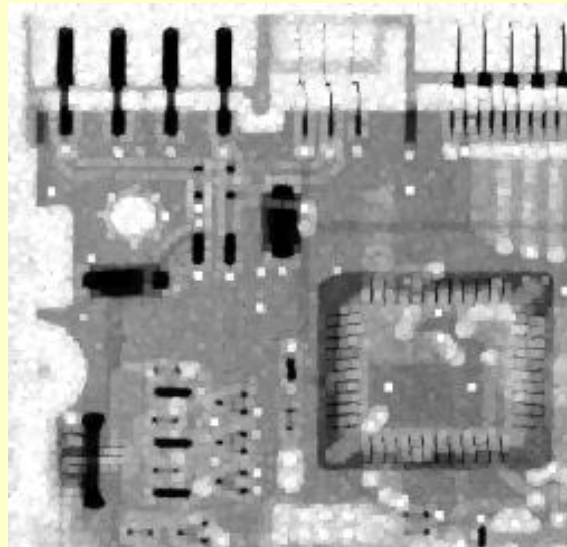


Image corrupted by salt noise with probability 0.1

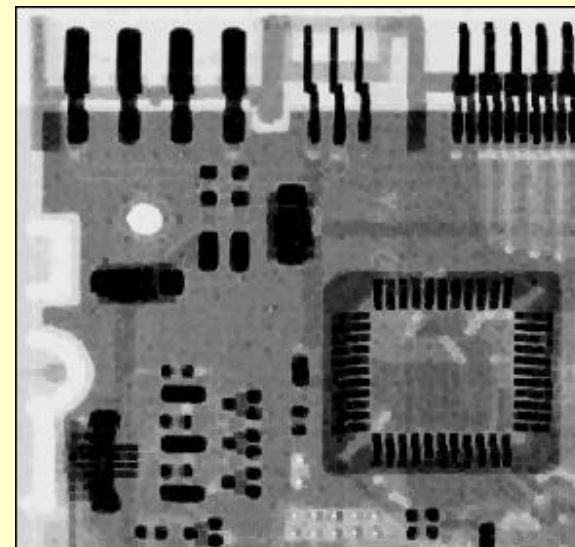


Min and Max order statistics filters may be useful in removing salt noise and pepper noise, respectively, but the result depends on the noise density.

Removing Pepper noise with Max filter



Removing Salt noise with Min filter



Spatial Noise removing filters in MATLAB

- ✓ *The mean as well as order statistics spatial filters can be applied using the following MATLAB instruction:*

F=spfilt(c, Type, m, n, parameters),

Where Type refer to one of the filters defined earlier, and m and n represent the filter size.

Examples

F=spfilt(c, 'amean', m, n) - for arithmetic's mean

F=spfilt(c, 'gmean', m, n) - for geometric's mean

F=spfilt(c, 'hmean', m, n) - for Harmonic mean

F=spfilt(c, 'Chmean', m, n, Q) - for contrahamonic mean

F=spfilt(c, 'max', m, n) - for max mxn mean filter

F=spfilt(c, 'min', m, n) - for the min mxn filter

F=spfilt(c, 'midpoint', m, n) - for the midpoint mxn filter

F=spfilt(c, 'median', m, n) - for the median mxn filter

Adaptive filters

- ✓ *The previous filters are applied regardless of local image variation.*
- ✓ *Adapted filters change their behaviour using local statistical parameters in the mask region. Consequently, adaptive filters outperform the non-adaptive ones.*
- ✓ *The following formula defines a simple adaptive filter:*

$$f(x,y) = c(x,y) - \sigma_i^2 \times (c(x,y) - m_L) / \sigma_L^2 ,$$

where m_L and σ_L stand for the mean and standard deviation of pixel values in the mask region, and σ_i is the image standard deviation.

- ✓ *An adaptive median filter can be defined which aims to replace $f(x,y)$ with the median of a neighbourhood up to a specified size as long as the **median** is different from the **max** and **min** values but $f(x,y)=\min$ or $f(x,y)=\max$. Otherwise, $f(x,y)$ is not changed. This filter is implemented using the MATLAB instruction:*

`adpmedian(f, Smax)`

where $Smax$ is the maximum allowed filter size.

Example

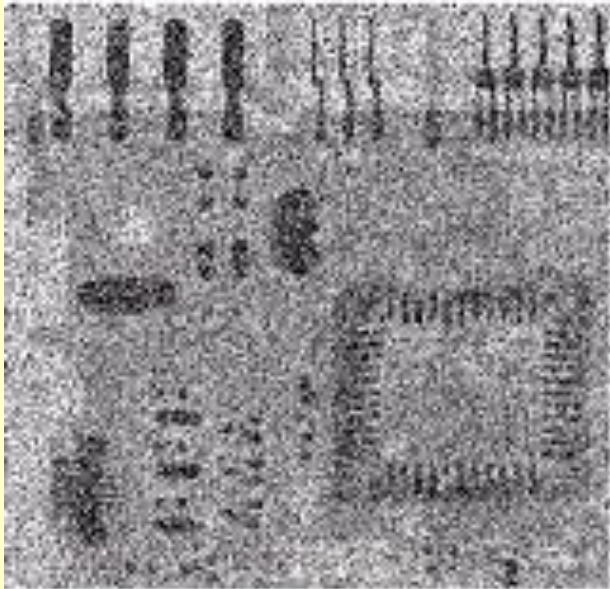
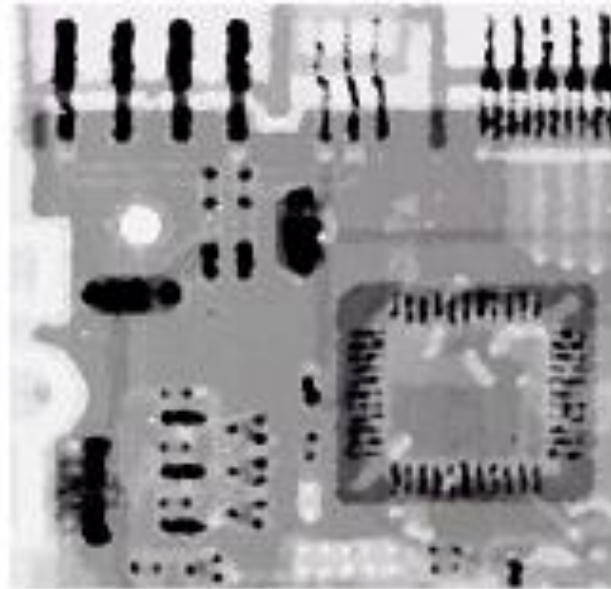
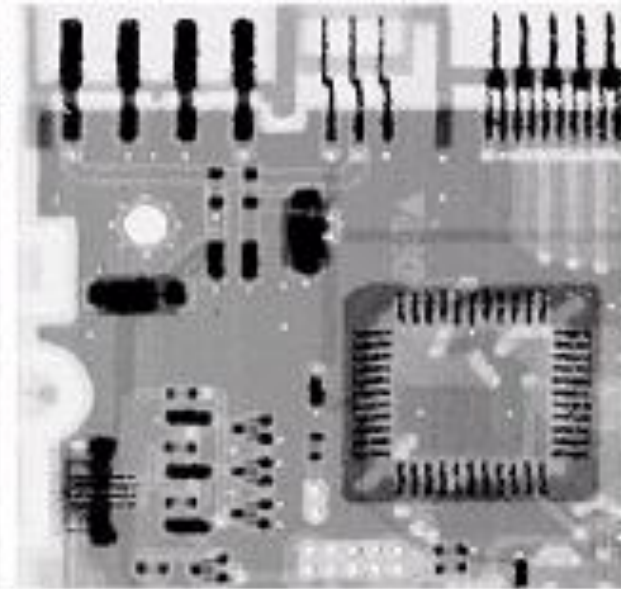


Image heavily Corrupted with Salt & Pepper



Result of filtering with 7x7 median filter.

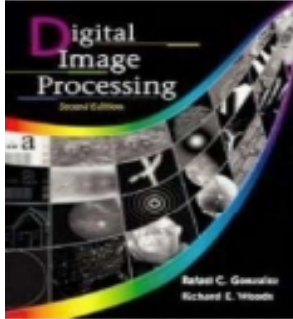


Result of filtering with adaptive 7x7 median filter.

Summary

- ✓ *There are different noise models.*
- ✓ *For good strategy in removing noise and restoring image quality one needs to determine noise distribution. (Check the histogram in a reasonable size smooth region with visibly small variation in values)*
- ✓ *Once the noise model is estimated use an appropriate filter. Histogram in the same region indicates level of success.*
- ✓ *Denoising (i.e. removing noise) often introduce other side effects.*
- ✓ *Advanced de-noising filters are based on adaptive strategy, i.e. the procedure tries to adapt the application of the filter as it progresses (see the main reference book on these types of filters).*
- ✓ *Frequency domain filters provide powerful de-noising methods.*
- ✓ *Noise in Colour images may have different characteristics in different colour channels, but removing noise uses the same strategy.*

END of Chapter 6



Digital Image Processing
Digital Image Processing Using Matlab

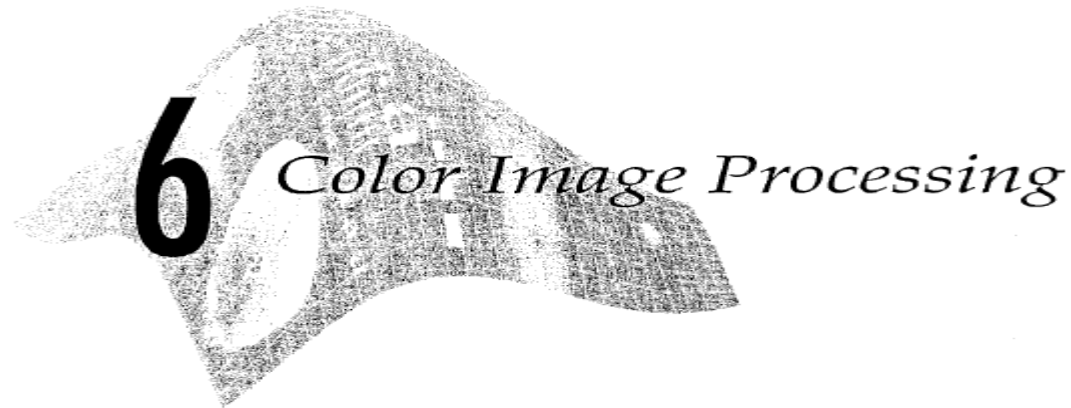
Digital Image Processing

Prepared by:

Dr. Ali J. Abboud

University of Diyala

2012-2013



6 *Color Image Processing*

Key Features of Chapter 6:

- **Colour Fundamentals.**
- **Colour Models .**
- **Pseudocolour image processing .**
- **Full Colour Image Processing.**
- **Colour Transformations.**
- **Sharpening and Smoothing .**

Introduction

Motivation to use colour:

- Powerful descriptor that often simplifies object identification and extraction from a scene
- Humans can discern thousands of colour shades and intensities, compared to about only two dozen shades of gray

Two major areas:

- *Full-colour processing*: e.g. images acquired by colour TV camera or colour scanner
- *Pseudo-colour processing*: assigning a colour to a particular monochrome intensity or range of intensities

Some of the gray-scale methods are directly applicable to colour images

Others require reformulation

Colour Fundamentals

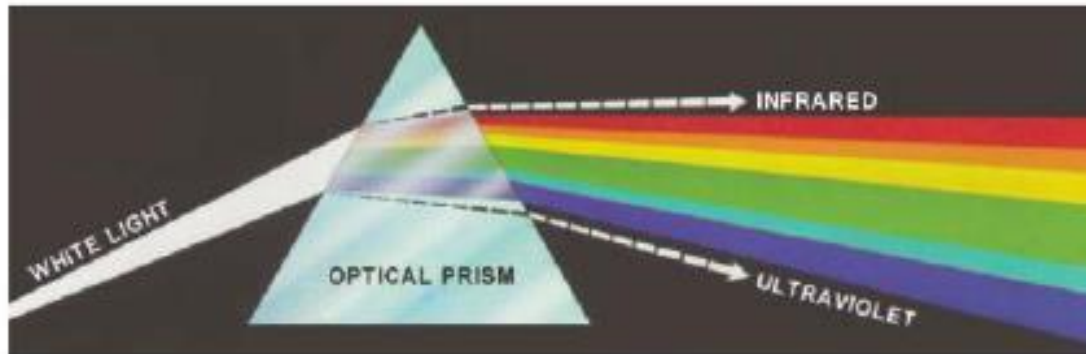


FIGURE 6.1 Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lamp Business Division.)

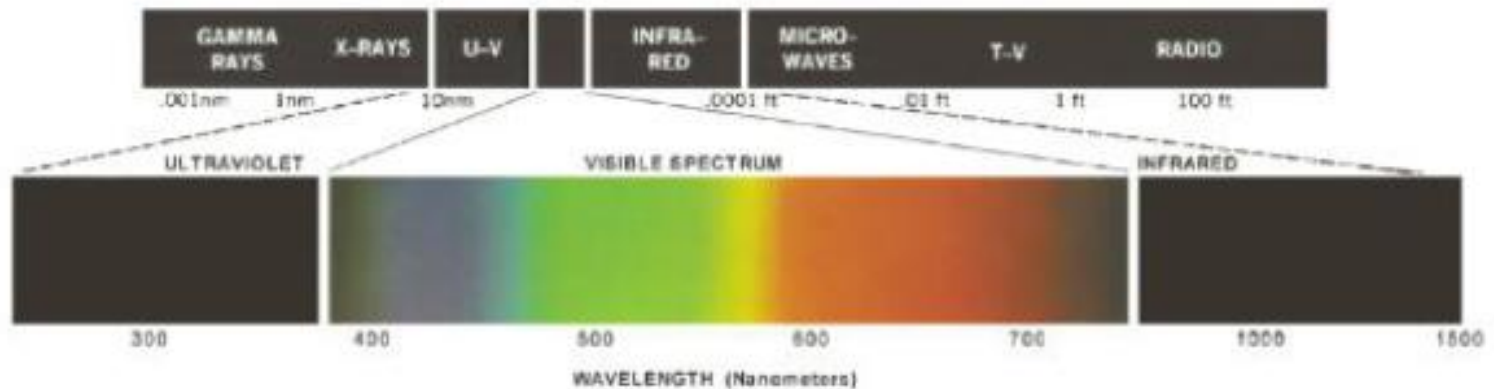


FIGURE 6.2 Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lamp Business Division.)

Colour Fundamentals

Perception of colours by the human eye

Cones can be divided into 3 principal sensing categories: (roughly) red, green and blue
~65% are sensitive to red light, ~33% to green light and ~2% to blue (but most sensitive)
⇒ Colours are seen as variable combinations of the *primary* colours: *Red, Green, Blue*
From CIE* (1931), wavelengths: blue = 435.8nm, green = 546.1nm, red = 700nm

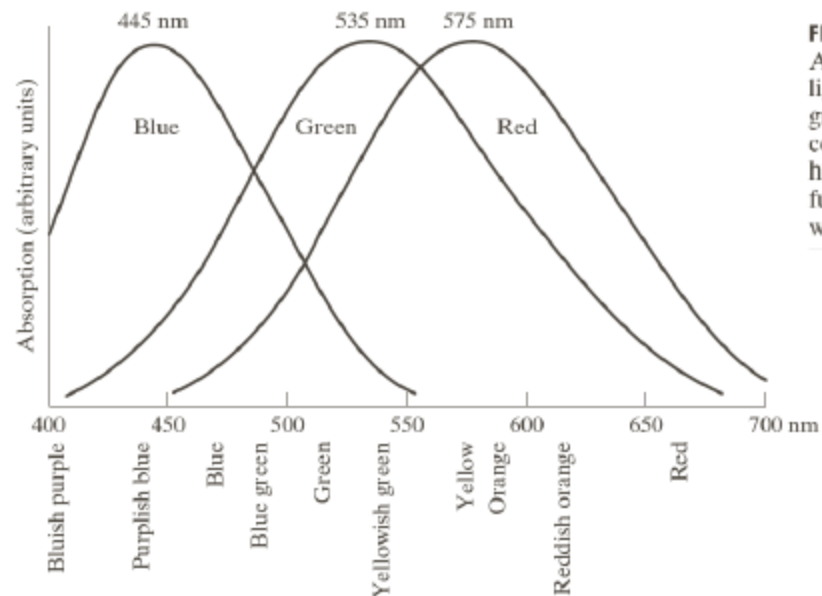


FIGURE 6.3
Absorption of light by the red, green, and blue cones in the human eye as a function of wavelength.

* CIE = Commission Internationale de l'Eclairage
(the International Commission on Illumination)

Colour Fundamentals

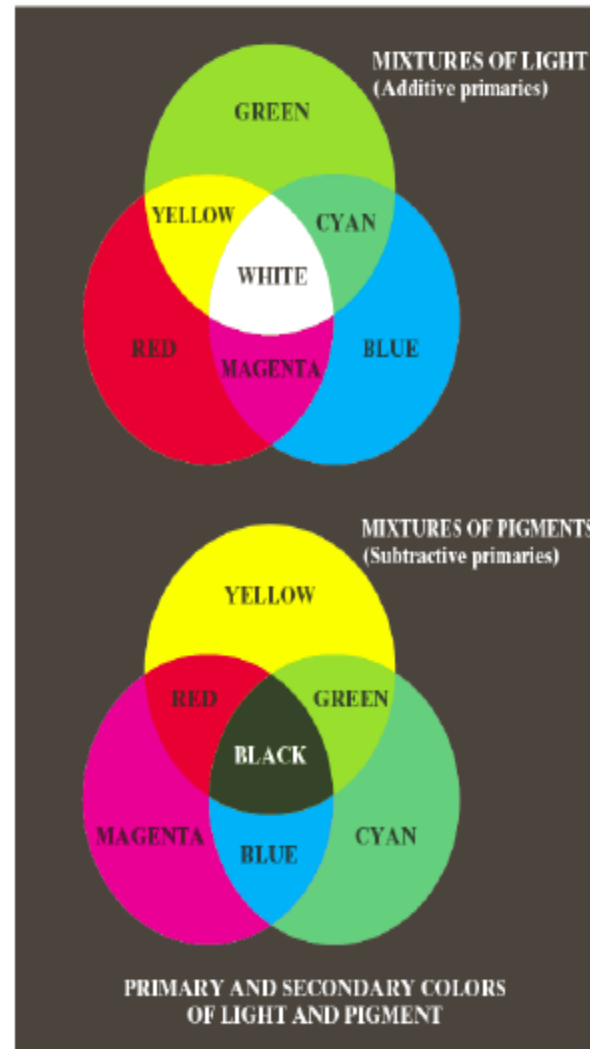
Primary colours can be added to produce the *secondary* colours of light:

- Magenta (red plus blue)
- Cyan (green plus blue)
- Yellow (red plus green)

Mixing the three primaries in the right intensities produce white light

Primary colours of pigment: absorb a primary colour of light and reflects or transmits the other two

→ magenta, cyan and yellow



a
b

FIGURE 6.4
Primary and secondary colors of light and pigments. (Courtesy of the General Electric Co., Lamp Business Division.)

Colour Fundamentals

Characteristics of a colour:

- **Brightness:** embodies the achromatic notion of intensity
- **Hue:** attribute associated with the dominant wavelength in a mixture of light waves
- **Saturation:** refers to the relative purity or the amount of white light mixed with a hue
(The pure spectrum colours are fully saturated; e.g. Pink (red and white) is less saturated, degree of saturation being inversely proportional to the amount of white light added)

Hue and Saturation together = *chromaticity*

⇒ Colour may be characterized by its brightness and chromaticity

Colour Fundamentals

Tristimulus values = amounts of red (X), green (Y) and blue (Z) needed to form a particular colour. A colour can be specified by its trichromatic coefficients:

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

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

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

$$\text{NB: } x + y + z = 1$$

Colour Fundamentals

Another approach for specifying colours:

The CIE *chromaticity diagram*:

Shows colour composition as a function of x (*red*) and y (*green*)

For any value of x and y : z (*blue*) is obtained by:

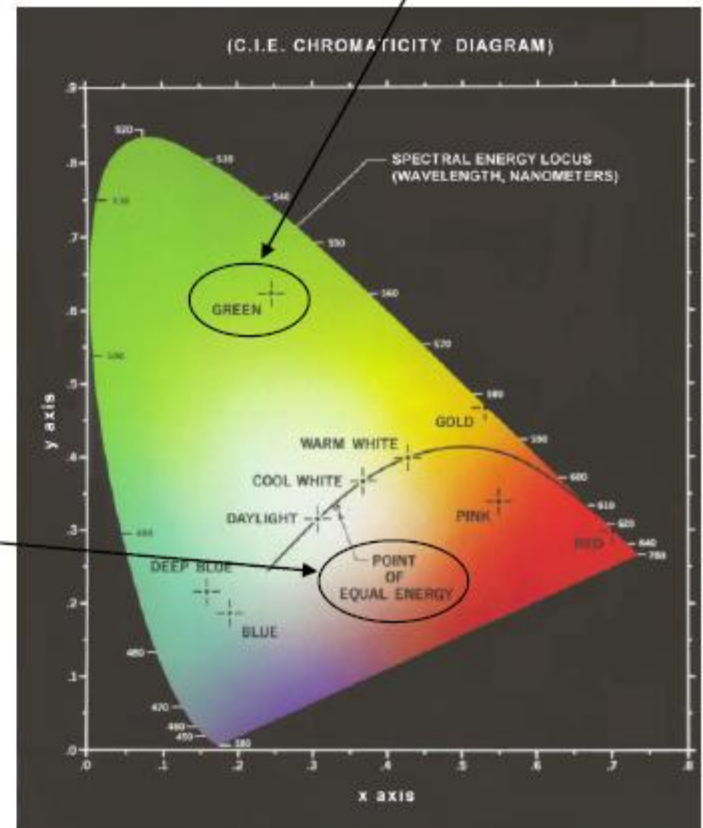
$$z = 1 - (x + y)$$

Pure colours of the spectrum (fully saturated):
boundary

Equal fractions of the 3 primary colours (CIE standard for white light)

FIGURE 6.5
Chromaticity diagram.
(Courtesy of the General Electric Co., Lamp Business Division.)

Content:
62% green
25% red
13% blue



Colour Fundamentals

Typical range of colours (*colour gamut*) produced by RGB monitors:

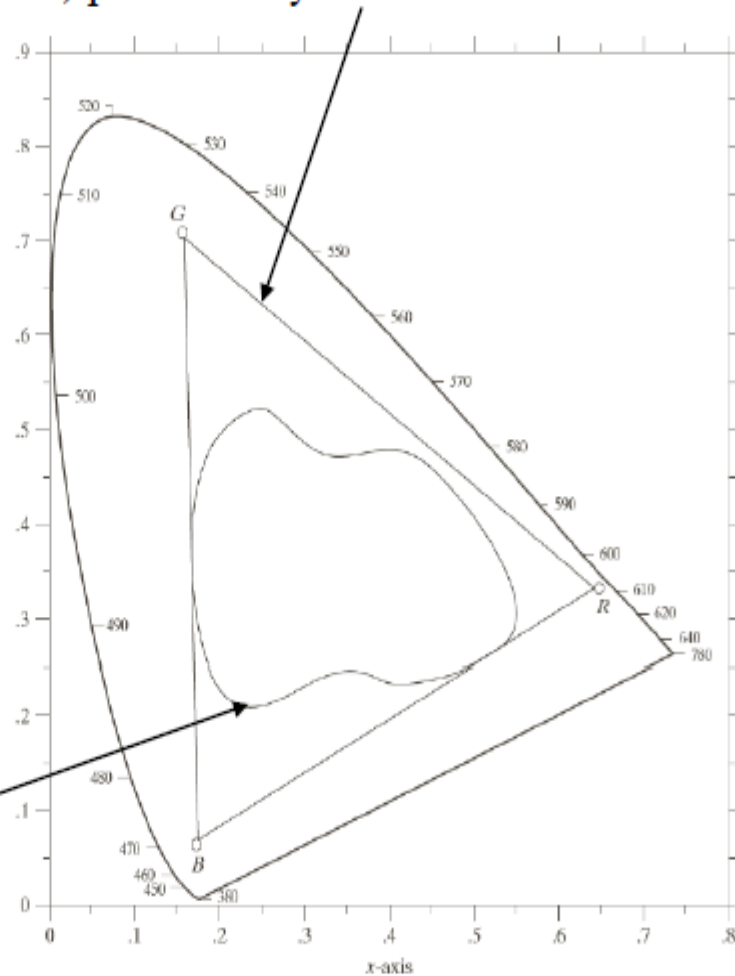


FIGURE 6.6
Typical color gamut of color monitors (triangle) and color printing devices (irregular region).

Colour gamut of today's high-quality colour printing devices

Colour Models

- Also called: *colour spaces* or *colour systems*
- Purpose: facilitate the specification of colours in some “standard” way
- Colour model = specification of a coordinate system and a subspace within it where each colour is represented by a single point

Most commonly used hardware-oriented models:

- **RGB** (Red, Green, Blue), for colour monitors and video cameras
- **CMY** (Cyan, Magenta, Yellow) and **CMYK** (CMY+Black) for colour printing
- **HSI** (Hue, Saturation, Intensity)

Colour Models

The RGB Colour Model

- Each colour appears in its primary spectral components of Red, Green and Blue
- Model based on a Cartesian coordinate System
- Colour subspace = cube
- RGB primary values: at 3 opposite corners (+ secondary values at 3 others)
- Black at the origin, White at the opposite corner

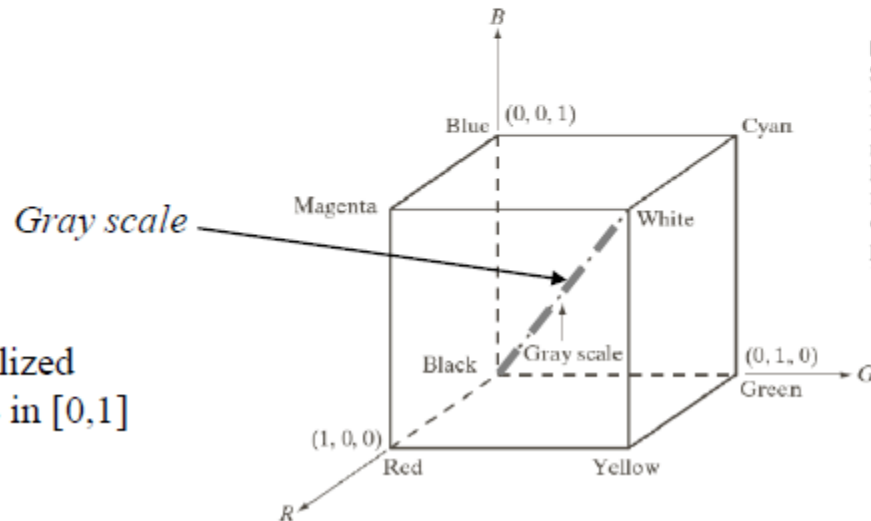


FIGURE 6.7
Schematic of the RGB color cube. Points along the main diagonal have gray values, from black at the origin to white at point (1, 1, 1).

Convention: all colour values normalized
=> unit cube and all values of R,G,B in [0,1]

Colour Models

Number of bits used to represent each pixel in the RGB space = *pixel depth*

Example: RGB image in which each of the red, green and blue images is a 8-bit image

⇒ Each RGB colour pixel (i.e. triplet of values (R,G,B)) is said to have a depth of 24 bits (*full-colour* image)

Total number of colours in a 24-bit RGB image is: $(2^8)^3 = 16,777,276$



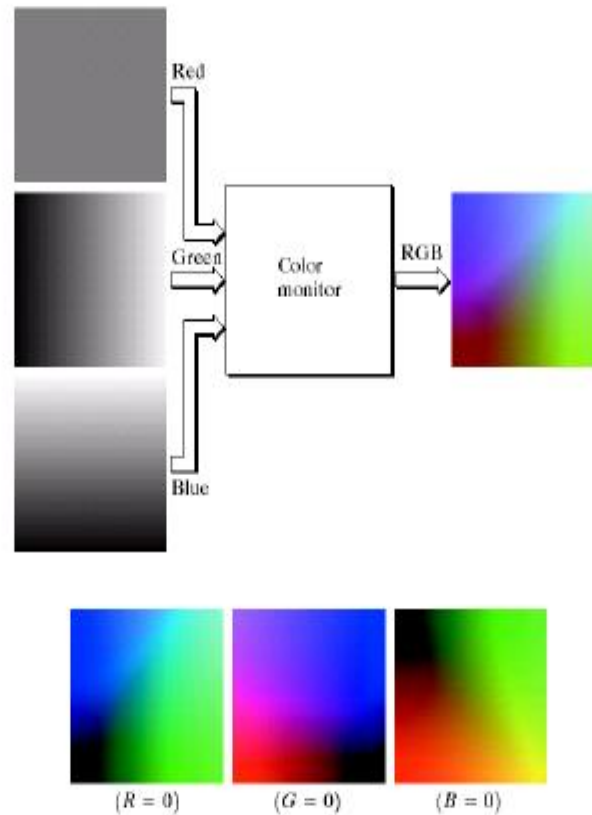
FIGURE 6.8 RGB
24-bit color cube.

Colour Models

a
b

FIGURE 6.9

(a) Generating the RGB image of the cross-sectional color plane ($127, G, B$).
(b) The three hidden surface planes in the color cube of Fig. 6.8.



NB: acquiring an image = reversed process:

Using 3 filters sensitive to red, green and blue, respectively (e.g. Tri-CCD sensor)

Colour Models

2.2 XYZ (CIE)

- Official definition of the CIE XYZ standard (normalised matrix):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Commonly used form: w/o leading fraction \Rightarrow RGB=(1,1,1) \rightarrow Y=1

Colour Models

2.2 The CMY and CMYK Colour Models

CMY: Cyan, Magenta, Yellow (secondary colours of light, or primary colours of pigments)

- CMY data input needed by most devices that deposit coloured pigments on paper, such as colour printers and copiers
- or RGB to CMY conversion:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(assuming normalized colour values)

Equal amounts of cyan, magenta and yellow => black, but muddy-looking in practice
=> To produce true black (predominant colour in printing) a 4th colour, black, is added
=> CMYK model (CMY + Black)

Colour Models

2.3 The HSI Colour Model

RGB and CMY models: straightforward + ideally suited for hardware implementations
+ RGB system matches nicely the human eye perceptive abilities

But, RGB and CMY not well suited for *describing* colours in terms practical for human interpretation

Human view of a colour object described by Hue, Saturation and Brightness (or Intensity)

- *Hue*: describes a pure colour (pure yellow, orange or red)
- *Saturation*: gives a measure of the degree to which a pure colour is diluted by white light
- *Brightness*: subjective descriptor practically impossible to measure. Embodies the achromatic notion of intensity => intensity (gray level), measurable

=> **HSI (Hue, Saturation, Intensity)** colour model

(or HSL: Lightness, HSB: Brightness, HSV: Value)

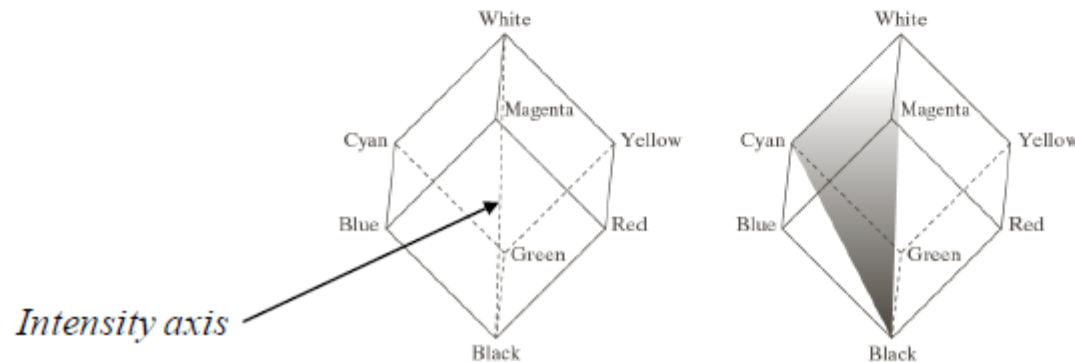
Colour Models

2.3 The HSI Colour Model

Intensity is along the line joining white and black in the RGB cube

⇒ To determine the intensity component of any colour point: pass a plane perpendicular to the intensity axis and containing the colour point. Intersection of the plane with the axis is the normalized intensity value

⇒ Saturation (purity) of a colour increases as a function of distance from the intensity axis (on the axis, saturation = 0, gray points)

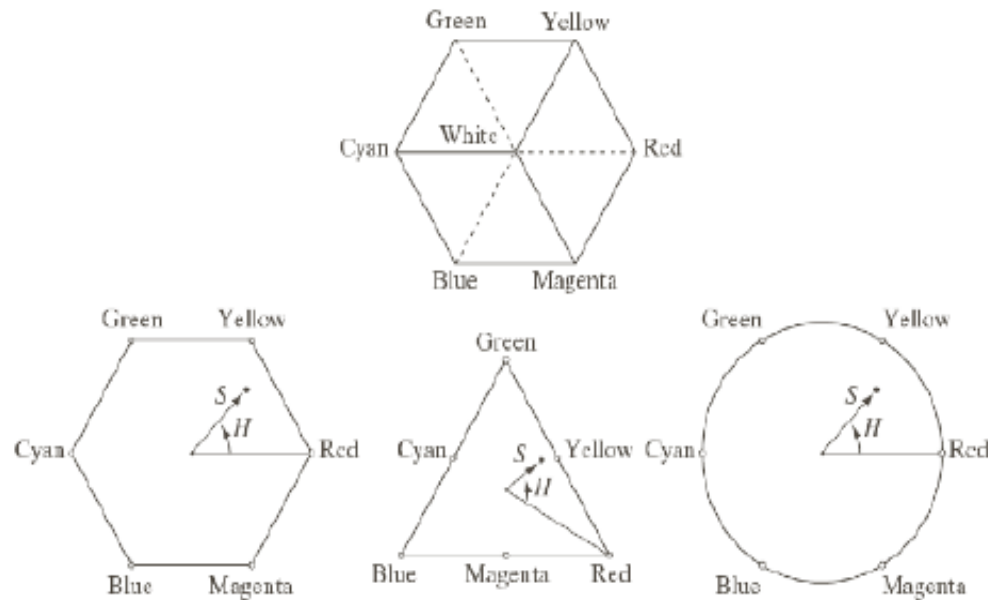


a b

FIGURE 6.12
Conceptual
relationships
between the RGB
and HSI color
models.

Colour Models

Colour planes, perpendicular to the intensity axis:



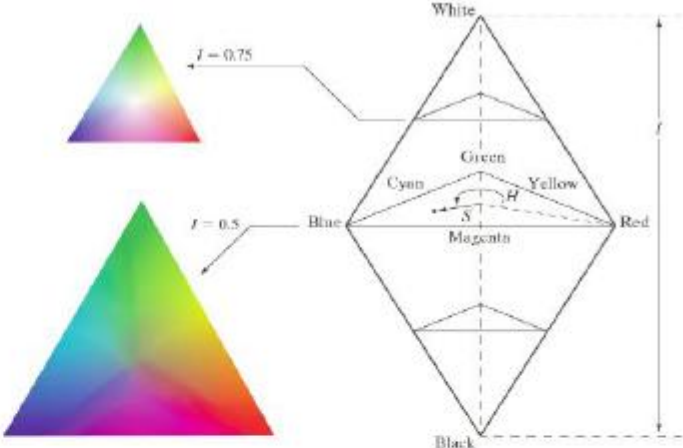
a
b c d

FIGURE 6.13 Hue and saturation in the HSI color model. The dot is an arbitrary color point. The angle from the red axis gives the hue, and the length of the vector is the saturation. The intensity of all colors in any of these planes is given by the position of the plane on the vertical intensity axis.

Colour Models

The HSI Colour Models based on:

Triangular colour planes



Circular colour planes

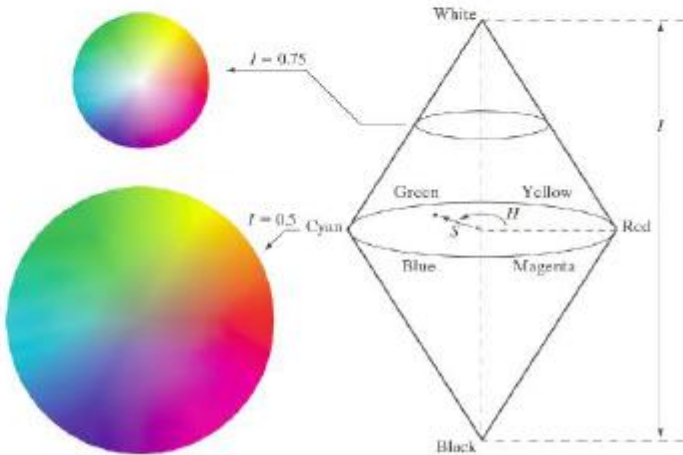


FIGURE 6.14 The HSI color model based on (a) triangular and (b) circular color planes. The triangles and circles are perpendicular to the vertical intensity axis.

Colour Models

Conversion from RGB to HSI

Given an RGB pixel: $H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases}$ then normalise H

with $\theta = \cos^{-1} \left\{ \frac{\frac{1}{2} [(R - G) + (R - B)]}{[(R - G)^2 + (R - B)(G - B)]^{1/2}} \right\}$

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

Intensity: $I = \frac{1}{3}(R + G + B)$ NB: RGB values **normalised to [0,1]**,
Theta measured w.r.t. red axis of the HSI space

Colour Models

Conversion from HSI to RGB

Three sectors of interest:

- RG sector ($0^\circ \leq H \leq 120^\circ$): $B = I(1 - S)$

$$R = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$G = 3I - (R + B)$$

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

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

$$B = 3I - (R + G)$$

Colour Models

Conversion from HSI to RGB

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

$$G = I(1 - S)$$

$$B = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$R = 3I - (G + B)$$

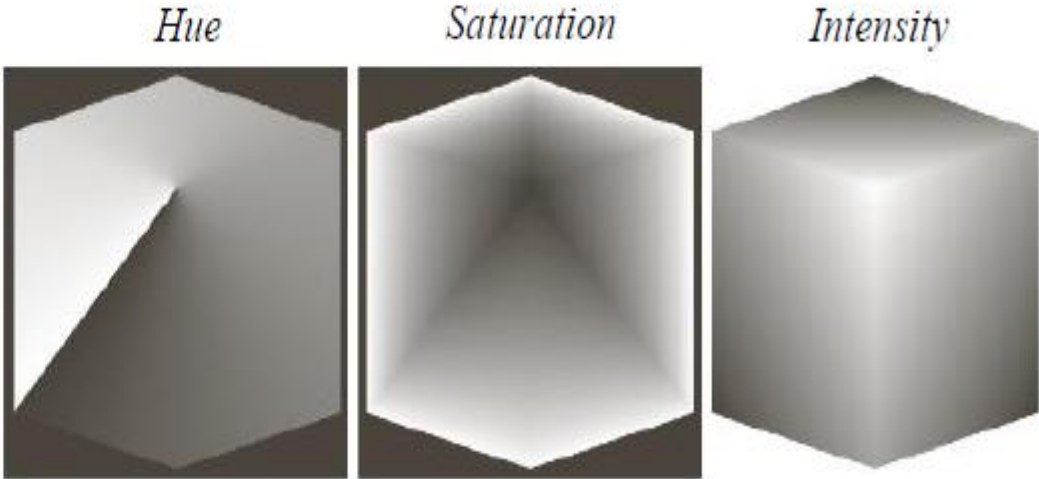
- Then normalise H

Colour Models

Conversion from HSI to RGB



RGB 24-bit colour cube



a b c

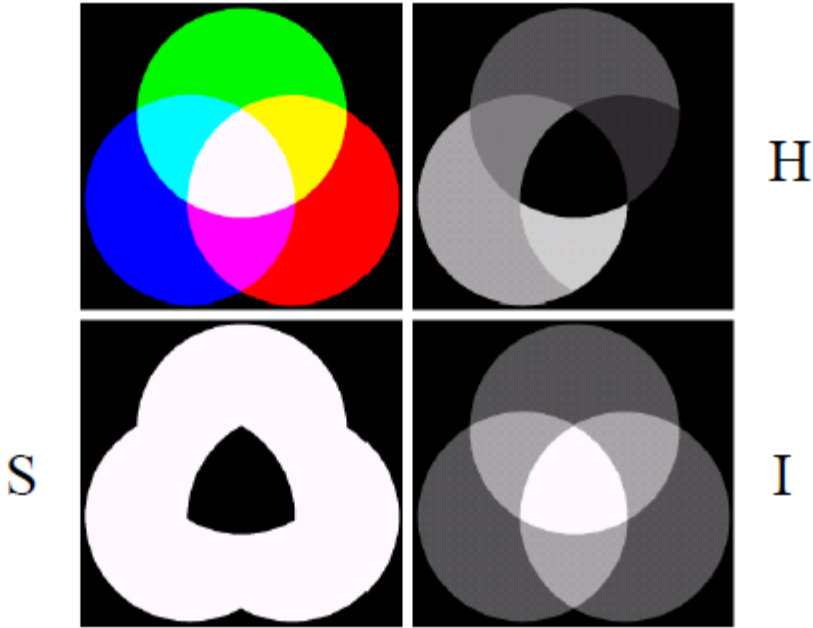
FIGURE 6.15 HSI components of the image in Fig. 6.8. (a) Hue, (b) saturation, and (c) intensity images.

Corresponding HSI values

Colour Models

Manipulation of HSI images:

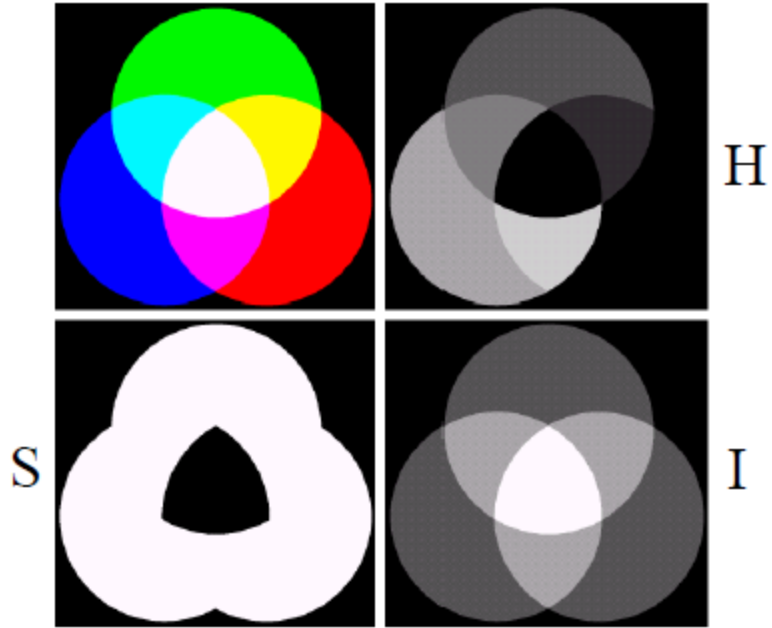
Primary and secondary
RGB colours



a b
c d

FIGURE 6.16 (a) RGB image and the components of its corresponding HSI image: (b) hue, (c) saturation, and (d) intensity.

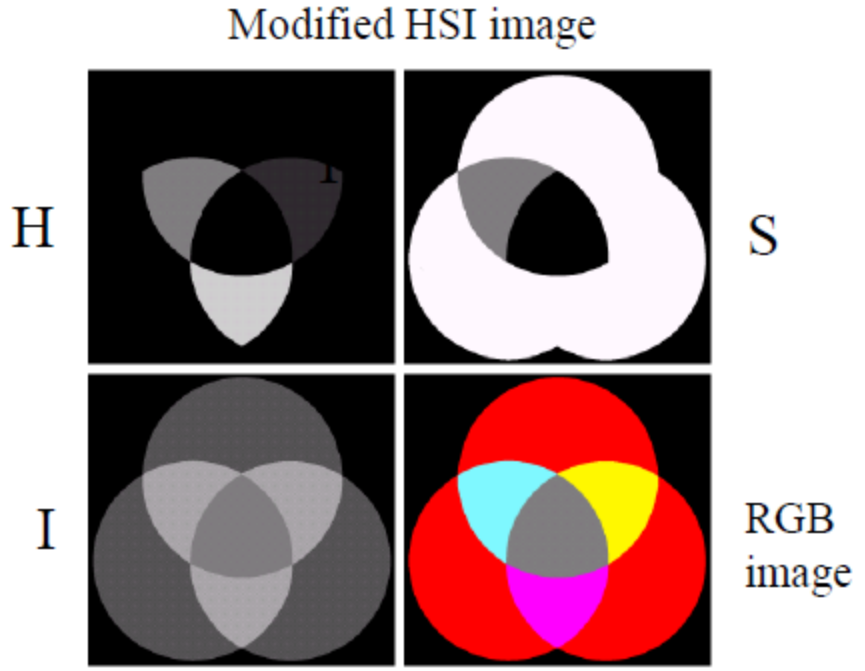
Colour Models



a b
c d

FIGURE 6.16 (a) RGB image and the components of its corresponding HSI image: (b) hue, (c) saturation, and (d) intensity.

Original image



a b
c d

FIGURE 6.17 (a)–(c) Modified HSI component images. (d) Resulting RGB image. (See Fig. 6.16 for the original HSI images.)

Colour Models

2.4 The L*a*b* model

Example of Colour Management System (CMS):

CIE L*a*b* model, or CIELAB:

$$L^* = 116 h\left(\frac{Y}{Y_W}\right) - 16$$

$$a^* = 500 \left[h\left(\frac{X}{X_W}\right) - h\left(\frac{Y}{Y_W}\right) \right]$$

$$b^* = 200 \left[h\left(\frac{Y}{Y_W}\right) - h\left(\frac{Z}{Z_W}\right) \right]$$

Where:
$$h(q) = \begin{cases} \sqrt[3]{q} & q > 0.008856 \\ 7.787q + 16/116 & q \leq 0.008856 \end{cases}$$

X_W , Y_W and Z_W are *reference white tristimulus*

Colour Models

The $L^*a^*b^*$ colour space is:

- *Colorimetric* (colours perceived as matching are encoded identically)
- *Perceptually uniform* (colour differences among various hues are perceived uniformly)
- *Device independent*

Other characteristics:

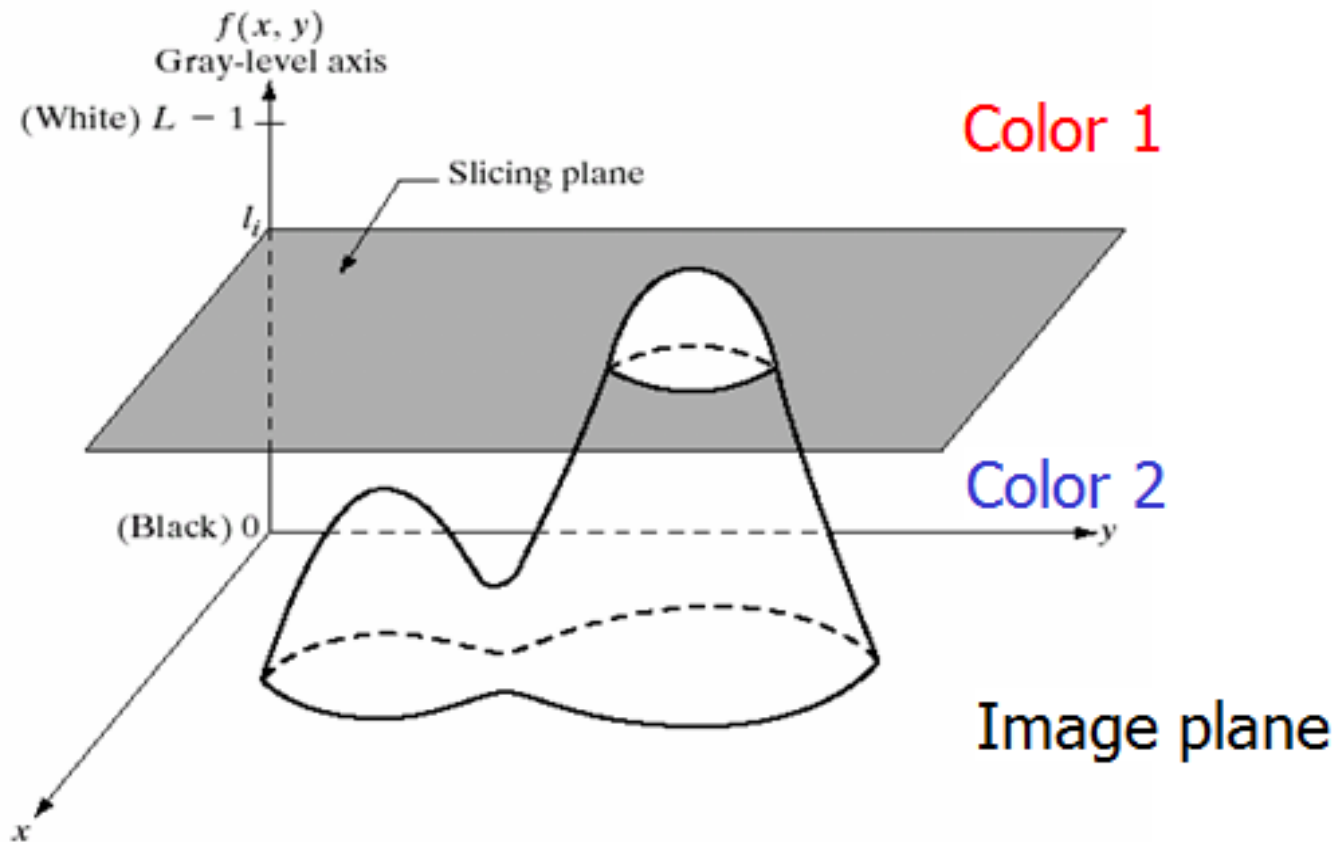
- Not a directly displayable format
- Its gamut encompasses the entire visible spectrum
- Can represent accurately the colours of any display, print, or input device
- Like HSI, excellent decoupler of intensity (represented by lightness L^*) and colour (a^* for red minus green, b^* for green minus blue)

Pseudo-Color Image Processing

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

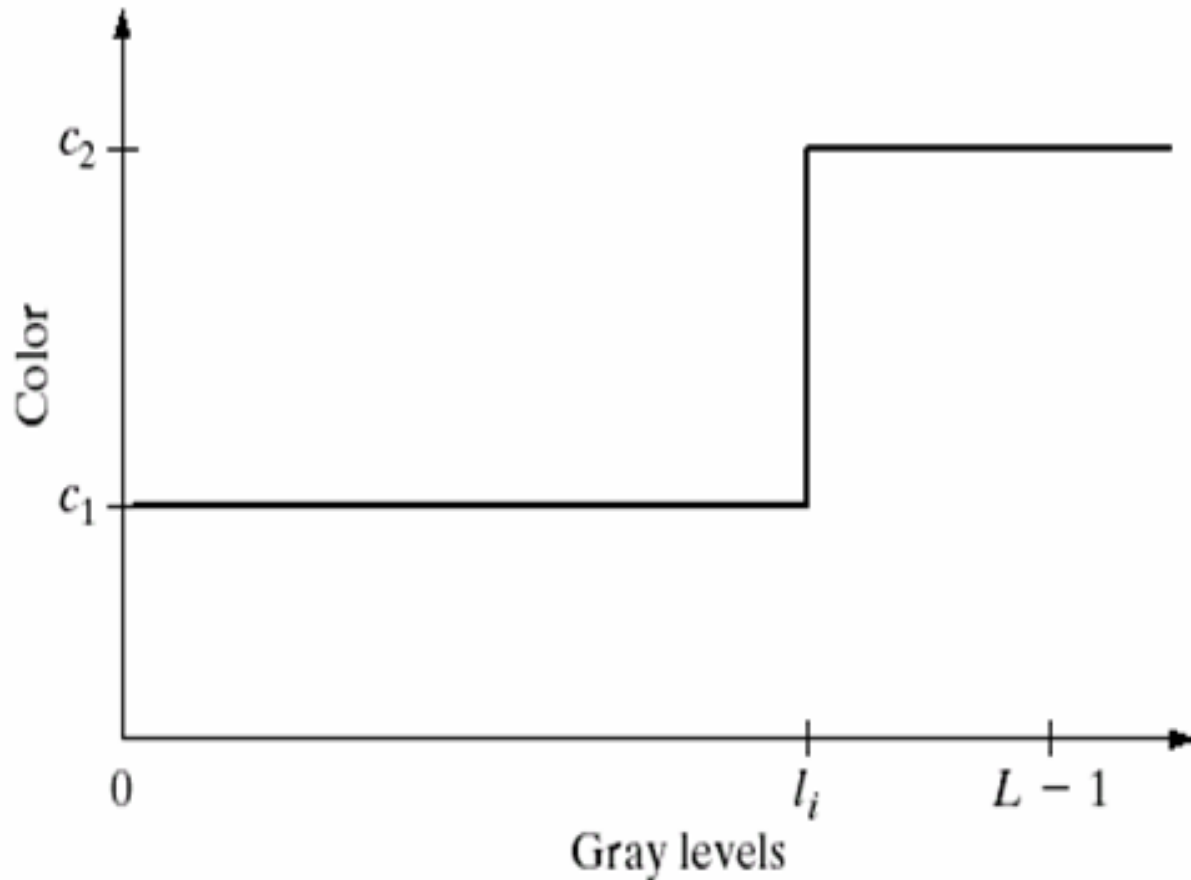
Pseudo-Color Image Processing: Intensity Slicing

■ 3-D view of intensity image



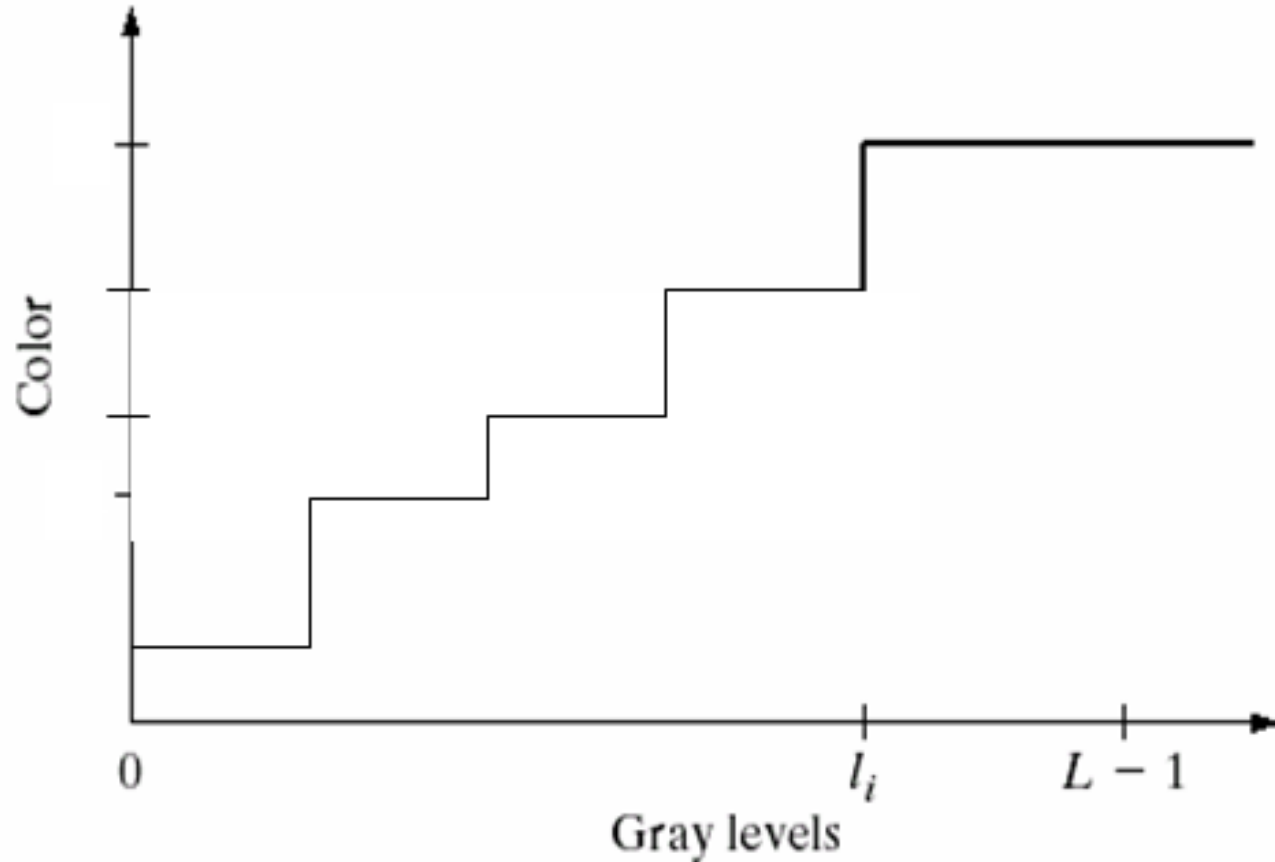
Pseudo-Color Image Processing: Intensity Slicing

- Alternative representation of intensity slicing

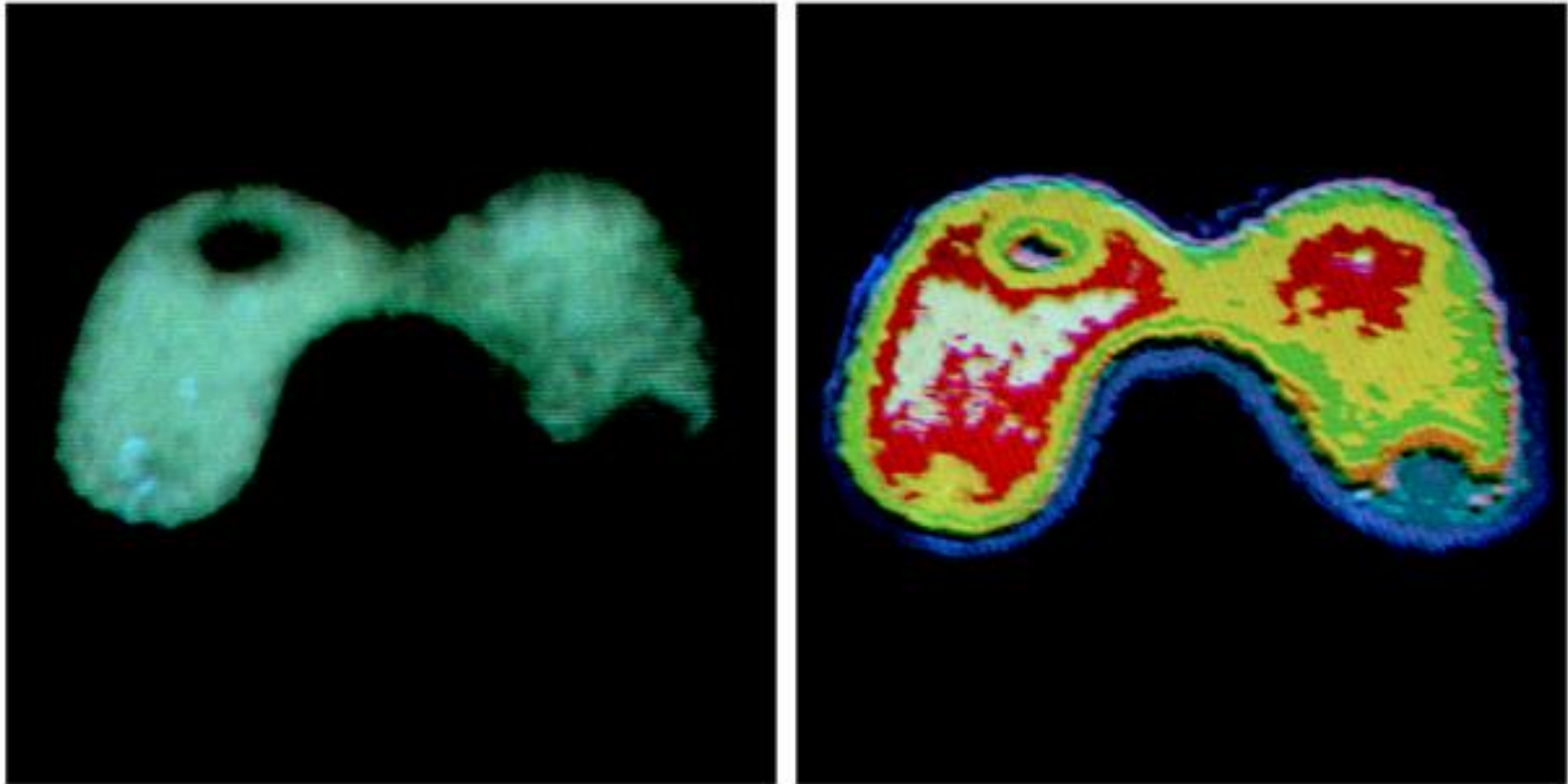


Pseudo-Color Image Processing: Intensity Slicing

- More slicing plane, more colors



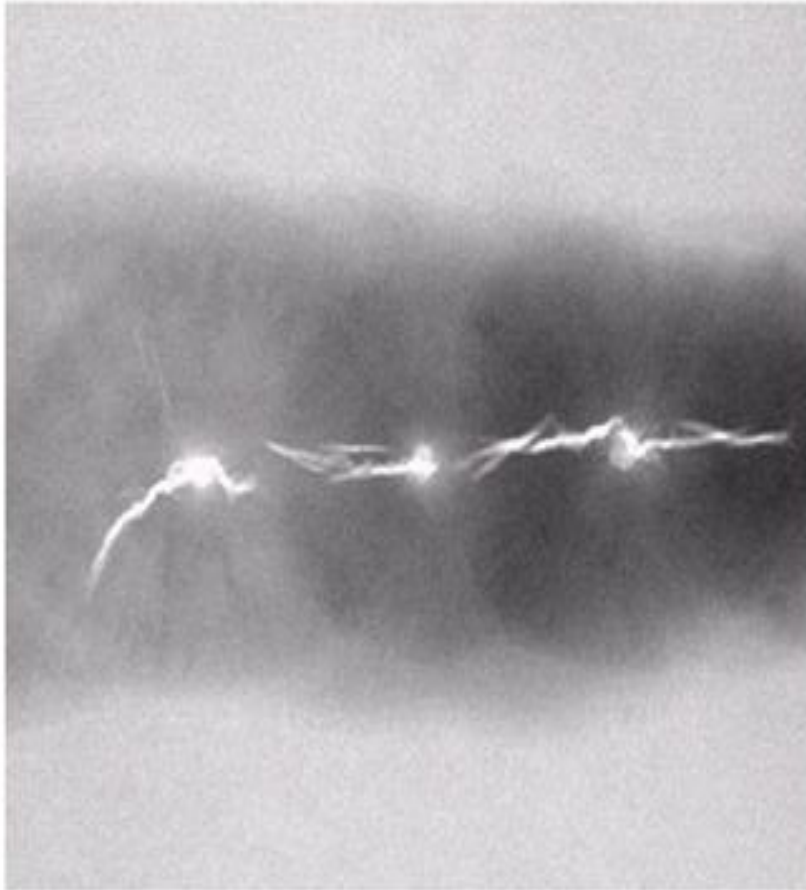
Pseudo-Color Image Processing: Application 1



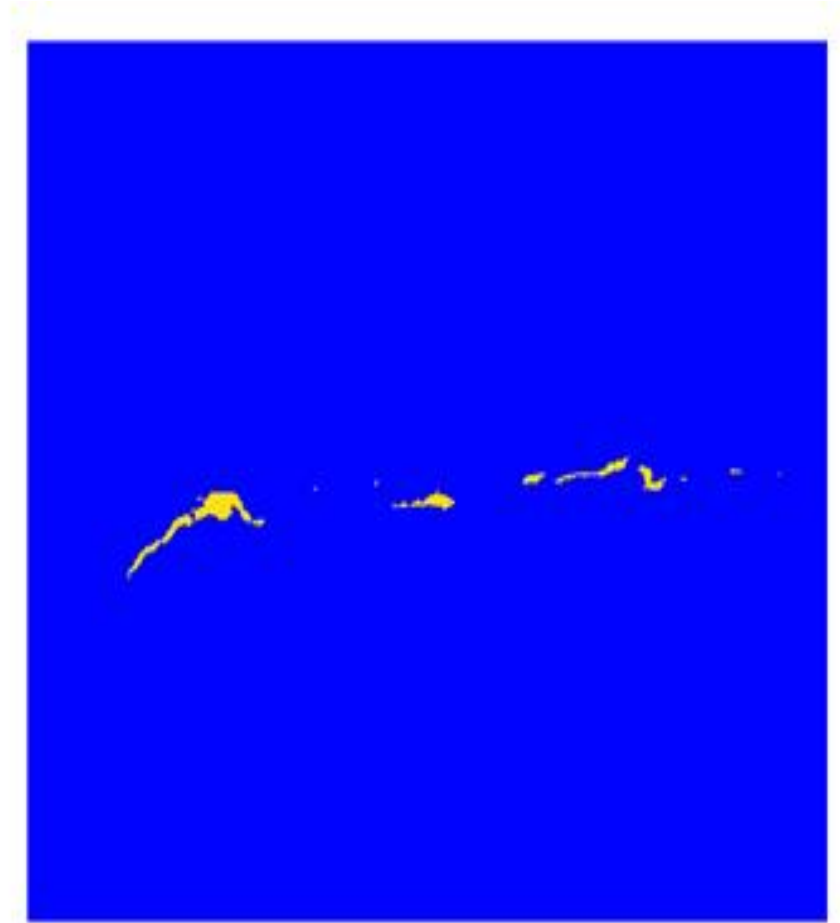
Radiation test pattern → 8 color regions

* See the gradual gray-level changes

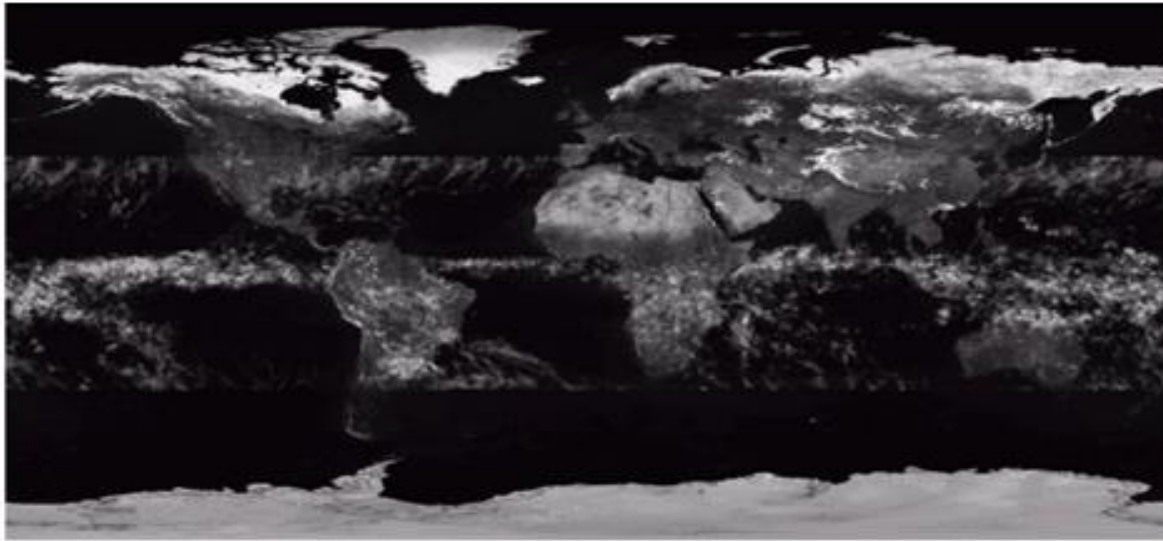
Pseudo-Color Image Processing: Application2



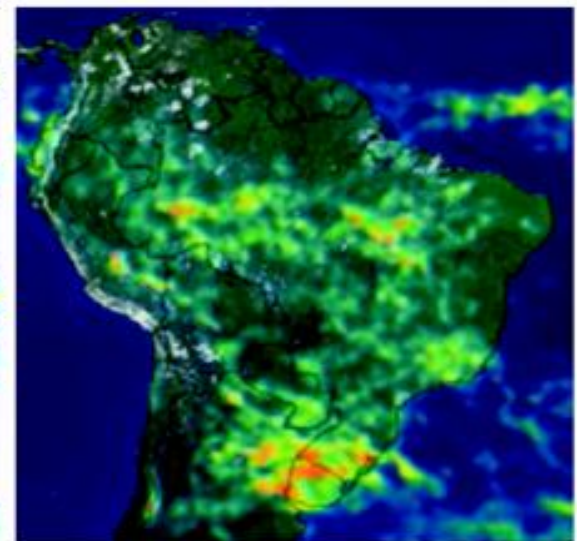
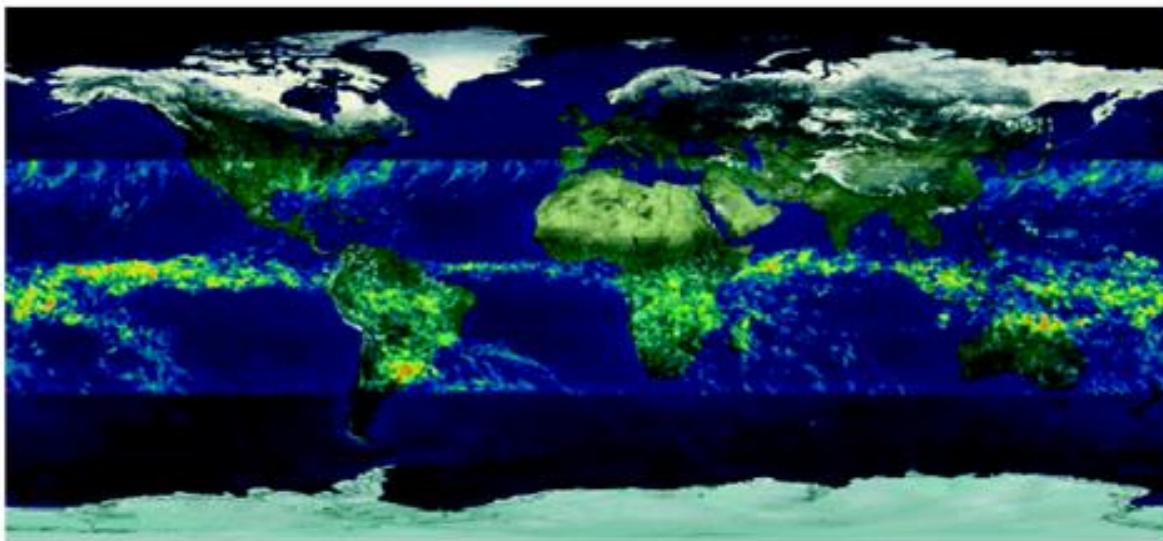
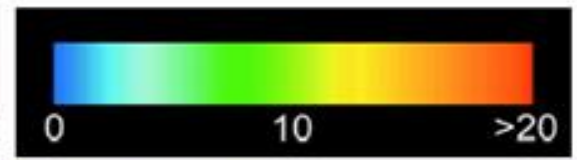
X-ray image of a weld



Pseudo-Color Image Processing: Application 3

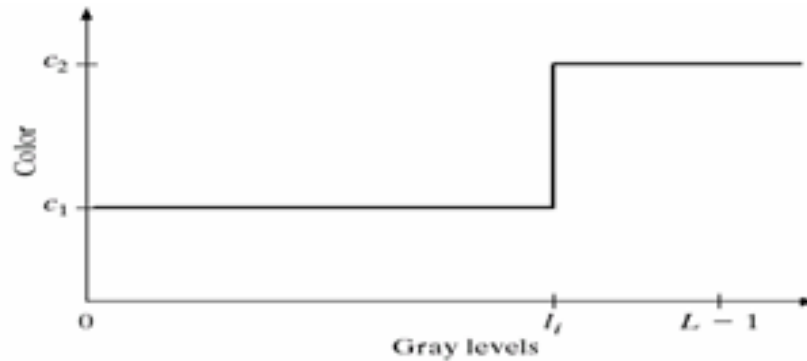


} Rainfall statistics

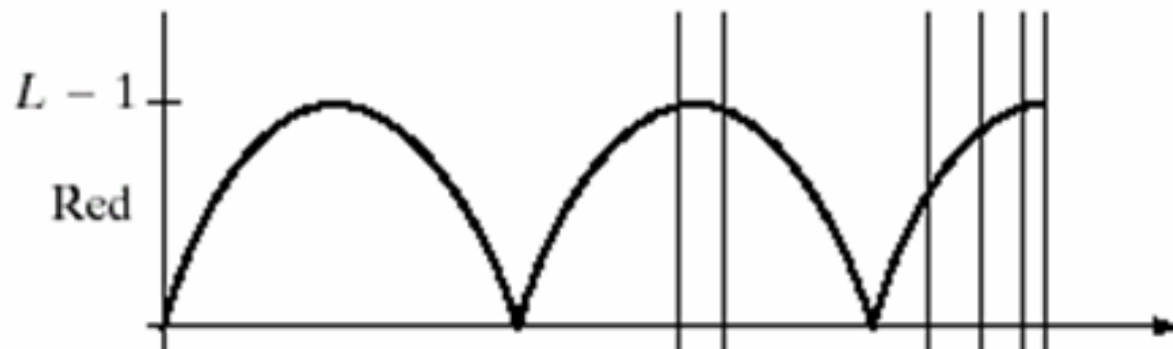


Gray Level to Color Transformation

- Intensity slicing: piecewise linear transformation



- General Gray level to color transformation



Gray Level to Color Transformation

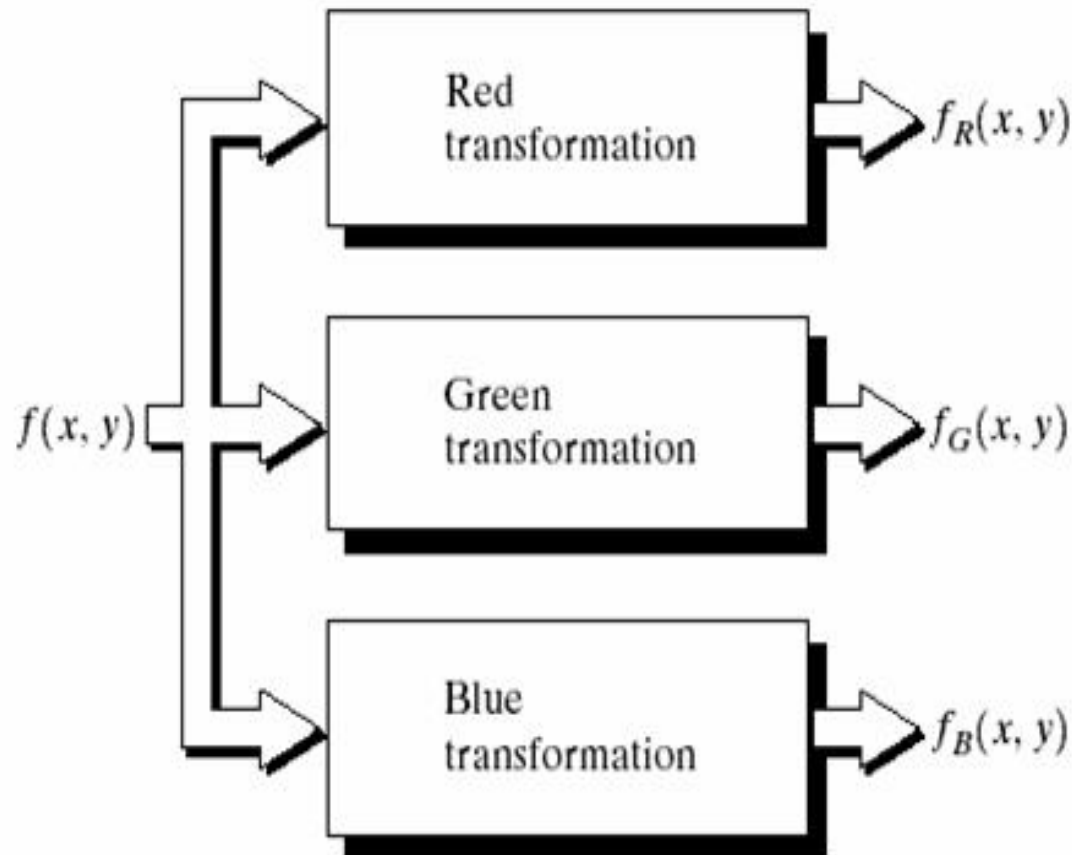
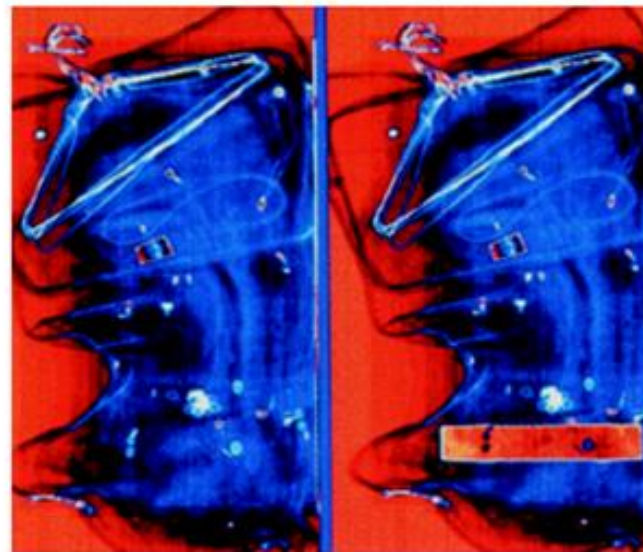
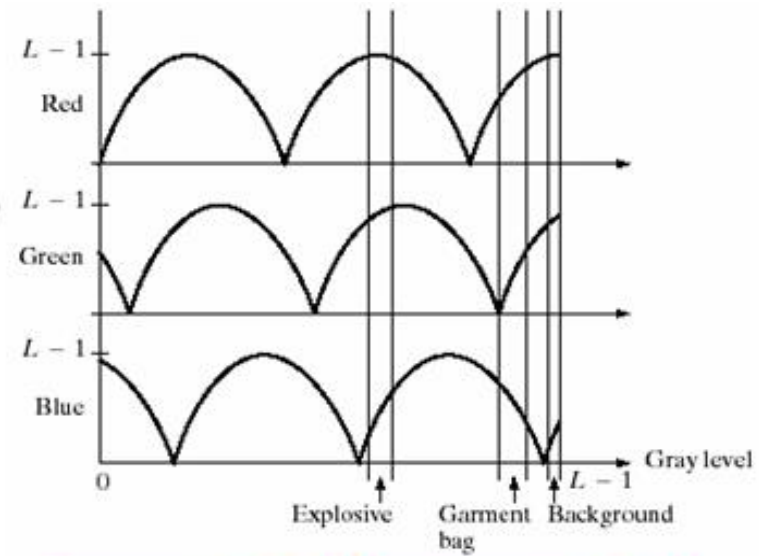
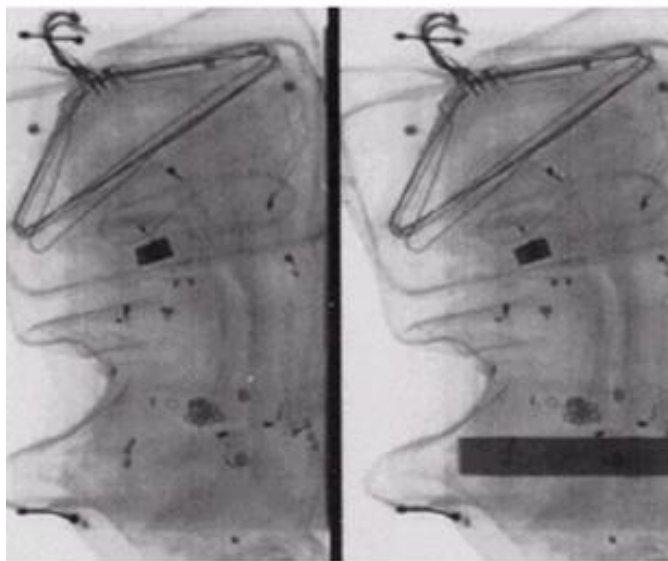


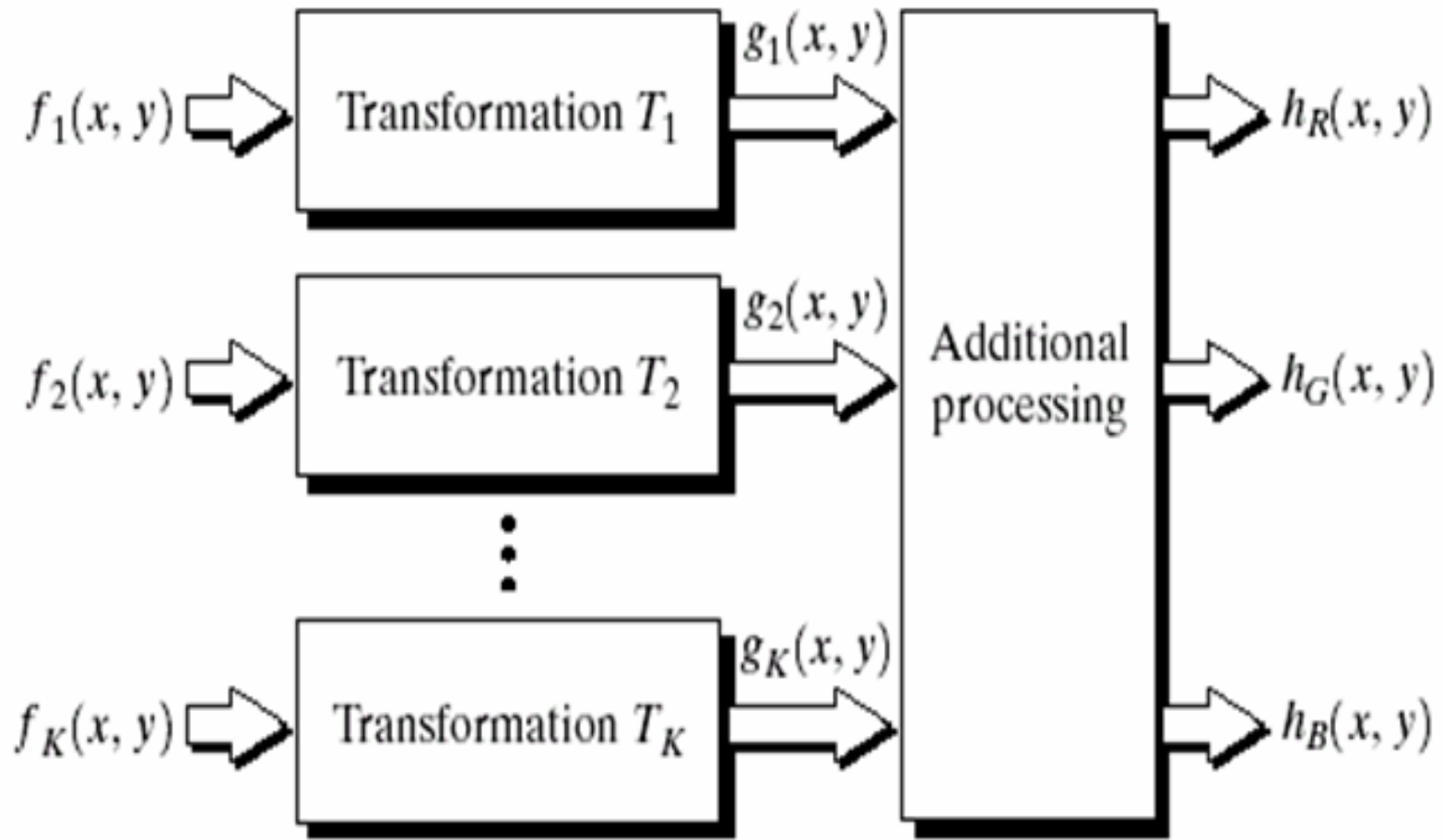
FIGURE 6.23 Functional block diagram for pseudocolor image processing. f_R , f_G , and f_B are fed into the corresponding red, green, and blue inputs of an RGB color monitor.

Gray Level to Color Transformation

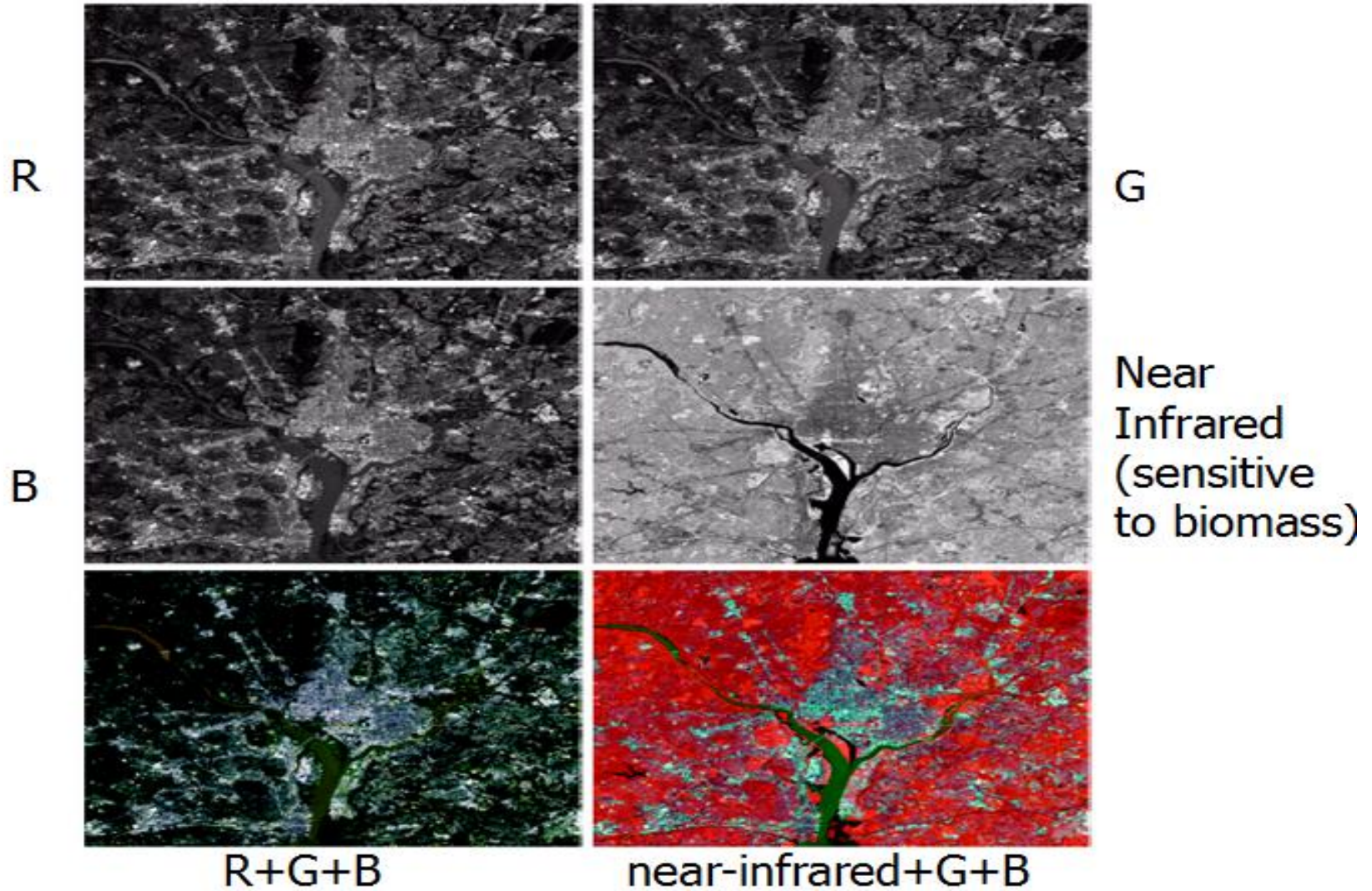


Combine Several Monochrome Images

Example: multi-spectral images



Combine Several Monochrome Images



Basics of Full-Color Image Processing

2 major categories:

- Processing of each component image individually
=> composite processed colour image
- Work with colour pixels (vectors) directly

Vector in RGB colour space:

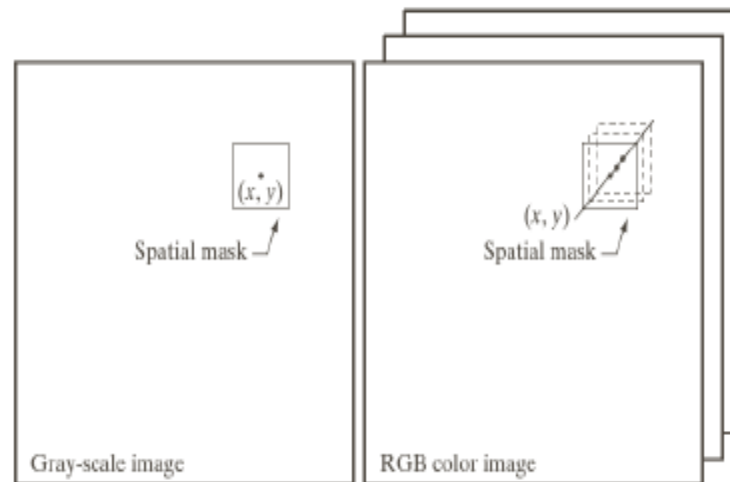
$$\mathbf{c} = \begin{bmatrix} c_R \\ c_G \\ c_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

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

Basics of Full-Color Image Processing

Per-colour-component and *vector-based* processing equivalent iff:

1. The process is applicable to both vectors and scalars
2. The operation on each component of a vector is independent of the other components



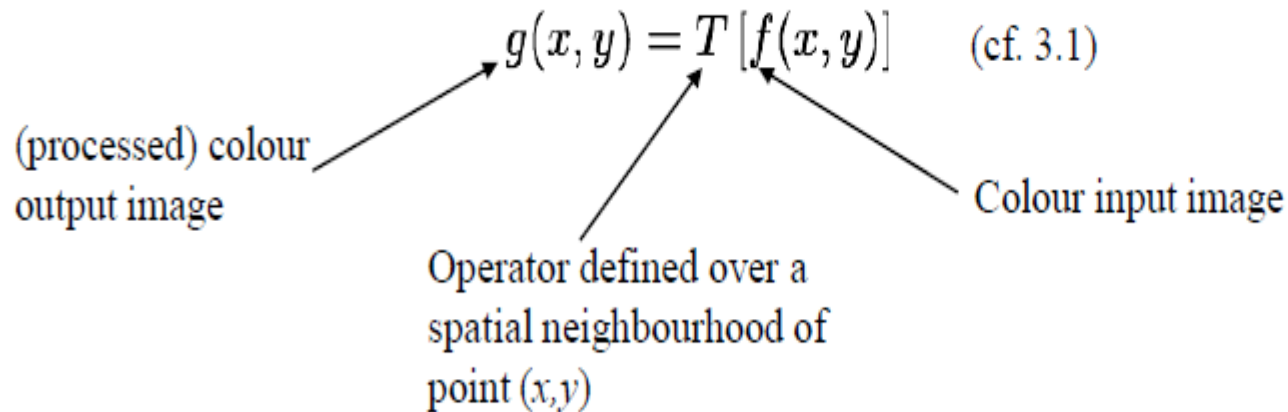
a b

FIGURE 6.29
Spatial masks for
gray-scale and
RGB color
images.

Basics of Full-Color Image Processing

NB: Context of a *single* colour model (no conversion between models)

5.1 Formulation



Set of *transformation of colour mapping functions*

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

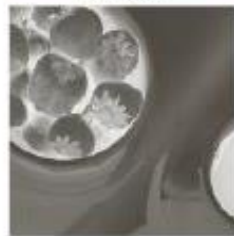
e.g. RGB or HSI: $n=3$. CMYK: $n=4$

Basics of Full-Color Image Processing



Full color

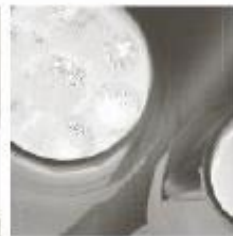
FIGURE 6.30 A full-color image and its various color-space components. (Interactive.)



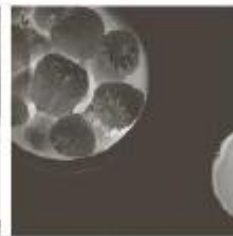
Cyan



Magenta



Yellow



Black



Red



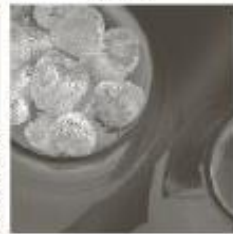
Green



Blue



Hue



Saturation



Intensity

Basics of Full-Color Image Processing

Example: modify the intensity of the full-colour image using: $g(x, y) = kf(x, y)$

In HSI: $s_3 = k r_3$ $s_1 = r_1$ and $s_2 = r_2$

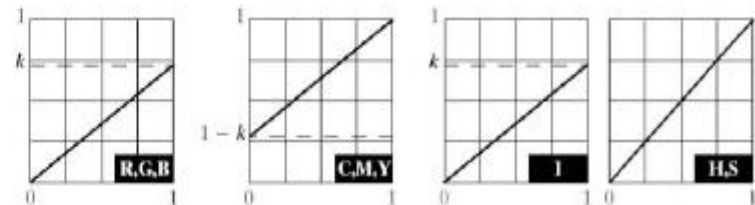
$$0 < k < 1$$

In RGB: $s_i = k r_i$ $i = 1, 2, 3$

In CMY: $s_i = k r_i + (1 - k)$ $i = 1, 2, 3$

a b
c d e

FIGURE 6.31
Adjusting the intensity of an image using color transformations. (a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting $k = 0.7$). (c)–(e) The required RGB, CMY, and HSI transformation functions. (Original image courtesy of MedData Interactive.)



Basics of Full-Color Image Processing

Colour complements

Hues directly opposite one another on the *colour circle* = *complements*

Complements are analogous to gray-scale negatives (section 3.2.1)
Useful for enhancing detail embedded in dark regions

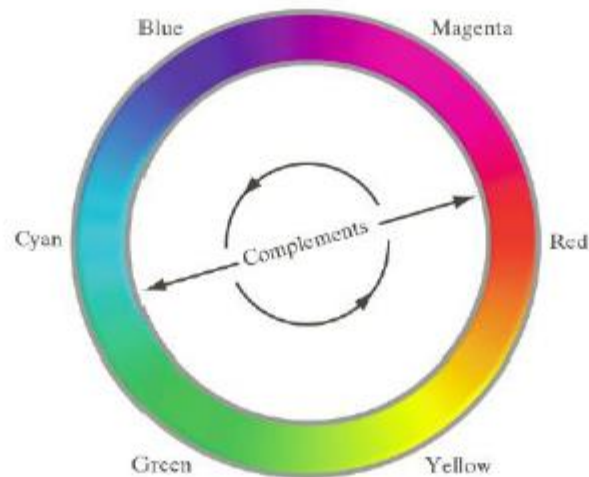
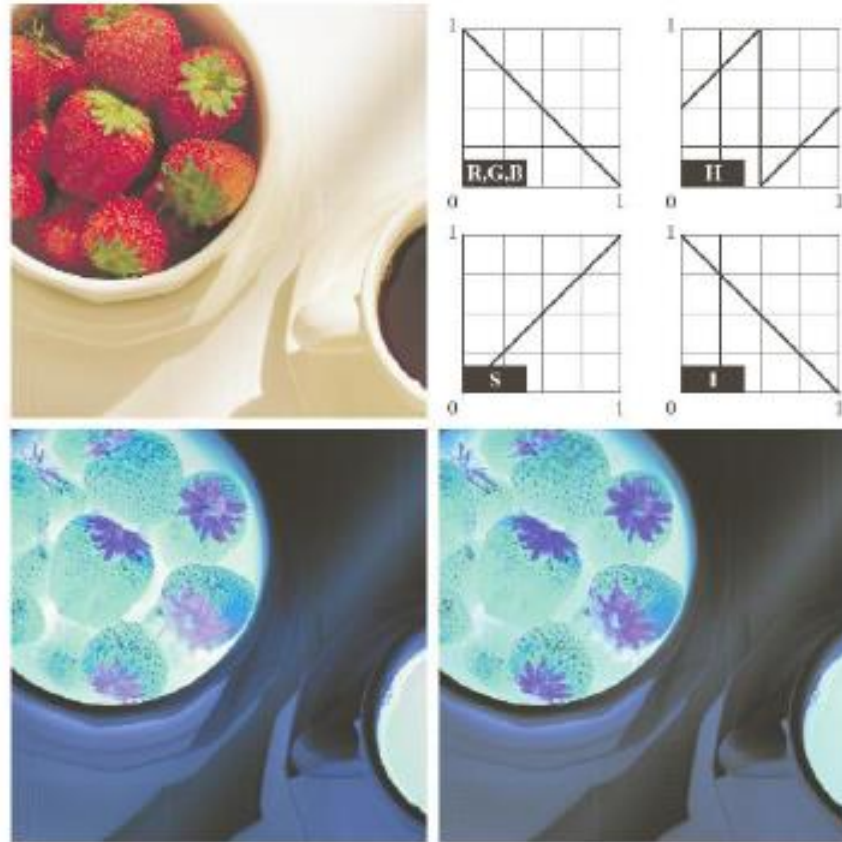


FIGURE 6.32
Complements on
the color circle.

Basics of Full-Color Image Processing

Colour complements



a b
c d

FIGURE 6.33

Color complement transformations.

(a) Original image.

(b) Complement transformation functions.

(c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

Basics of Full-Color Image Processing

Colour slicing

Highlighting a specific range of colours to separate objects from surroundings

- Display the colour of interest, or:
- Use the region defined by colours as a mask

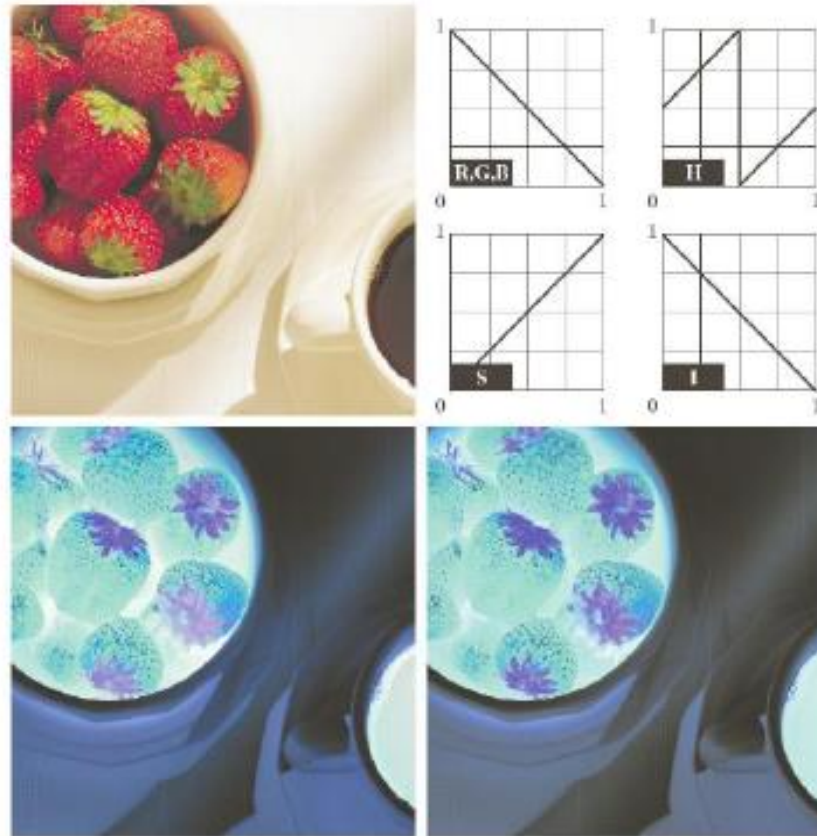
If colours of interest enclosed by a cube (or hypercube) of width W and centered at a prototypical (e.g. average) colours with components (a_1, a_2, \dots, a_n) , then:

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

=> Highlight the colours around the prototype by forcing all other colours to the midpoint of the reference colour space (e.g. middle gray in RGB: (0.5,0.5,0.5))

Basics of Full-Color Image Processing

Colour complements



a b
c d
FIGURE 6.33
Color complement transformations. (a) Original image. (b) Complement transformation functions. (c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

Basics of Full-Color Image Processing

Colour slicing

Highlighting a specific range of colours to separate objects from surroundings

- Display the colour of interest, or:
- Use the region defined by colours as a mask

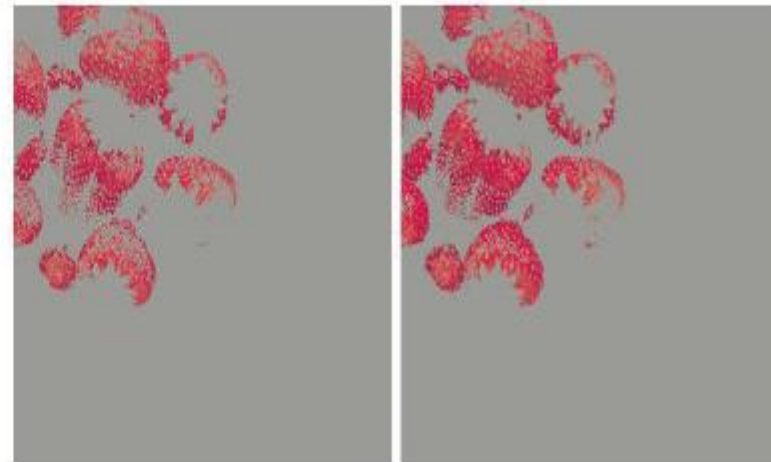
If colours of interest enclosed by a cube (or hypercube) of width W and centered at a prototypical (e.g. average) colours with components (a_1, a_2, \dots, a_n) , then:

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

=> Highlight the colours around the prototype by forcing all other colours to the midpoint of the reference colour space (e.g. middle gray in RGB: (0.5,0.5,0.5))

Basics of Full-Color Image Processing

Colour slicing



a b

FIGURE 6.34 Color-slicing transformations that detect (a) reds within an RGB cube of width $W = 0.2549$ centered at $(0.6863, 0.1608, 0.1922)$, and (b) reds within an RGB sphere of radius 0.1765 centered at the same point. Pixels outside the cube and sphere were replaced by color $(0.5, 0.5, 0.5)$.

Basics of Full-Color Image Processing

Tone and Colour Corrections

Effectiveness of these transformations judged ultimately in print

But developed, refined and evaluated on monitors

⇒ Need to maintain a high degree of colour consistency between monitors used and eventual output devices

⇒ *Device-independent* colour model, relating the colour gamuts of the monitors and output devices

Basics of Full-Color Image Processing

1. Tonal transformations

Typical transformations for correcting three common tonal imbalances:

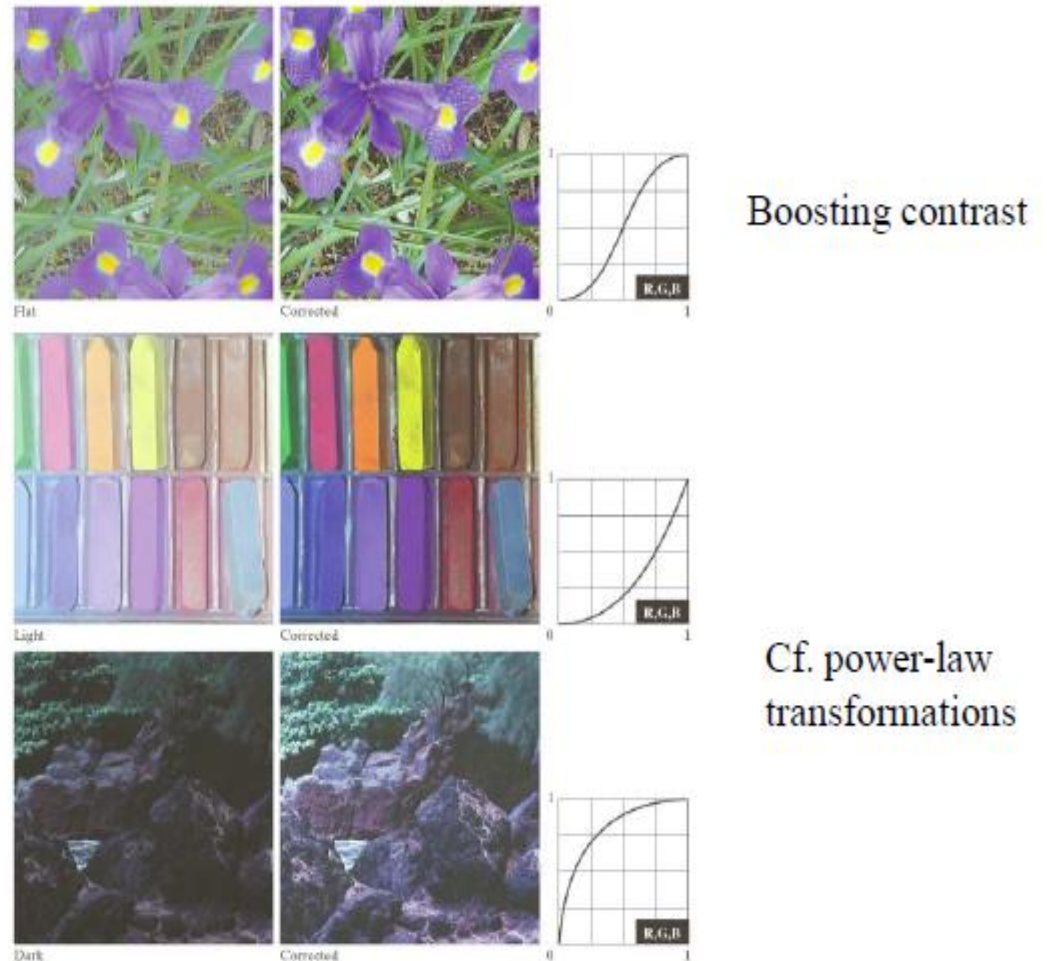


FIGURE 6.35 Tonal corrections for flat, light (high key), and dark (low key) color images. Adjusting the red, green, and blue components equally does not always alter the image hues significantly.

Basics of Full-Color Image Processing

2. Colour balancing

- Goal: move the white point of a given image closer to pure white (R=G=B)
- Example of strongly coloured illuminant: incandescent indoor lighting (=> yellow or orange hue)
- NB: using white may not always be a good idea...

Basics of Full-Color Image Processing

2. Colour balancing

Analyze (spectrometer) a known colour in an image
colour in an image

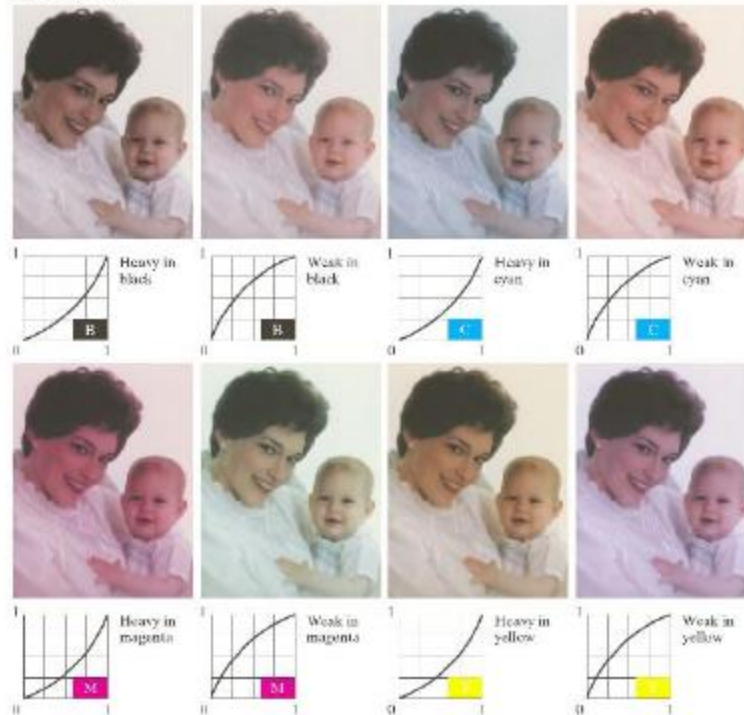
When white areas: accurate visual
assessments are possible

Other example: skin tones



Original/Corrected

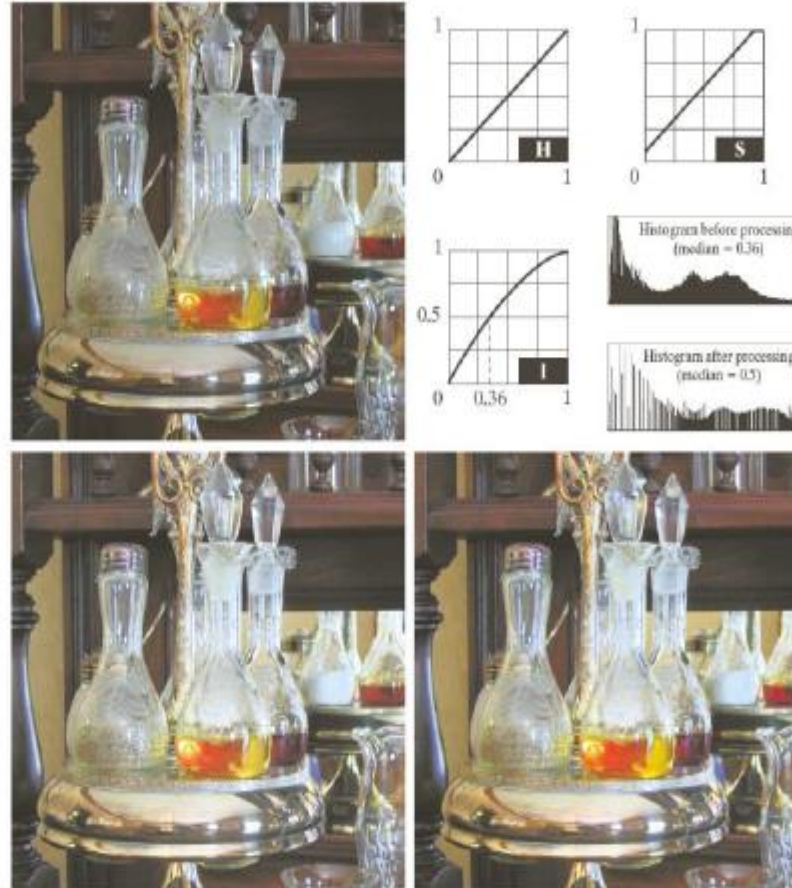
FIGURE 6.36 Color balancing corrections for CMYK color images.



Basics of Full-Color Image Processing

Histogram Processing

Example: Histogram Equalisation in the HSI colour space



a b
c d

FIGURE 6.37 Histogram equalization (followed by saturation adjustment) in the HSI color space.

Basics of Full-Color Image Processing

Smoothing and Sharpening

Colour Image Smoothing

S_{xy} : Set of coordinates of a neighbourhood centered at (x,y) in an RGB image

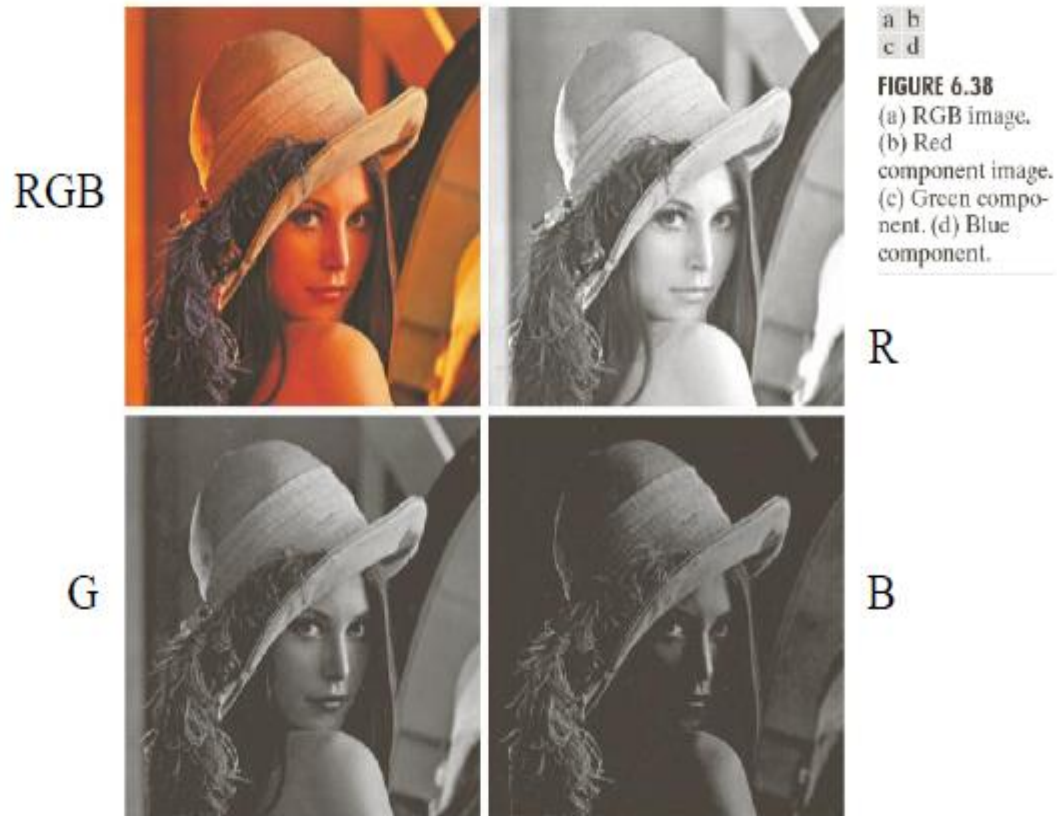
Average of the RGB component vectors in this neighbourhood:

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

Can be carried out on a per-colour-plane basis (same as averaging using RGB vectors)

Basics of Full-Color Image Processing

Colour Image Smoothing: Example



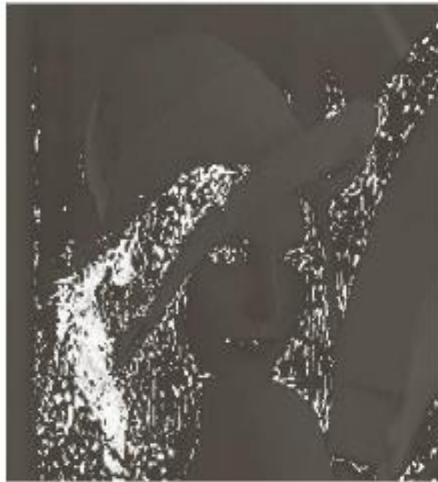
Basics of Full-Color Image Processing

Colour Image Smoothing: Example

RGB



H



S



I



a b c

FIGURE 6.39 HSI components of the RGB color image in Fig. 6.38(a). (a) Hue. (b) Saturation. (c) Intensity.

Basics of Full-Color Image Processing

Colour Image Smoothing: Example



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

Basics of Full-Color Image Processing

Colour Image Smoothing: Example



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

Basics of Full-Color Image Processing

Colour Image Sharpening

In RGB, the Laplacian of vector \mathbf{c} is:
$$\nabla^2 [\mathbf{c}(x, y)] = \begin{bmatrix} \nabla^2 R(x, y) \\ \nabla^2 G(x, y) \\ \nabla^2 B(x, y) \end{bmatrix}$$

=> Can be computed on each component image separately



a b c

FIGURE 6.41 Image sharpening with the Laplacian. (a) Result of processing each RGB channel. (b) Result of processing the HSI intensity component and converting to RGB. (c) Difference between the two results.

Basics of Full-Color Image Processing

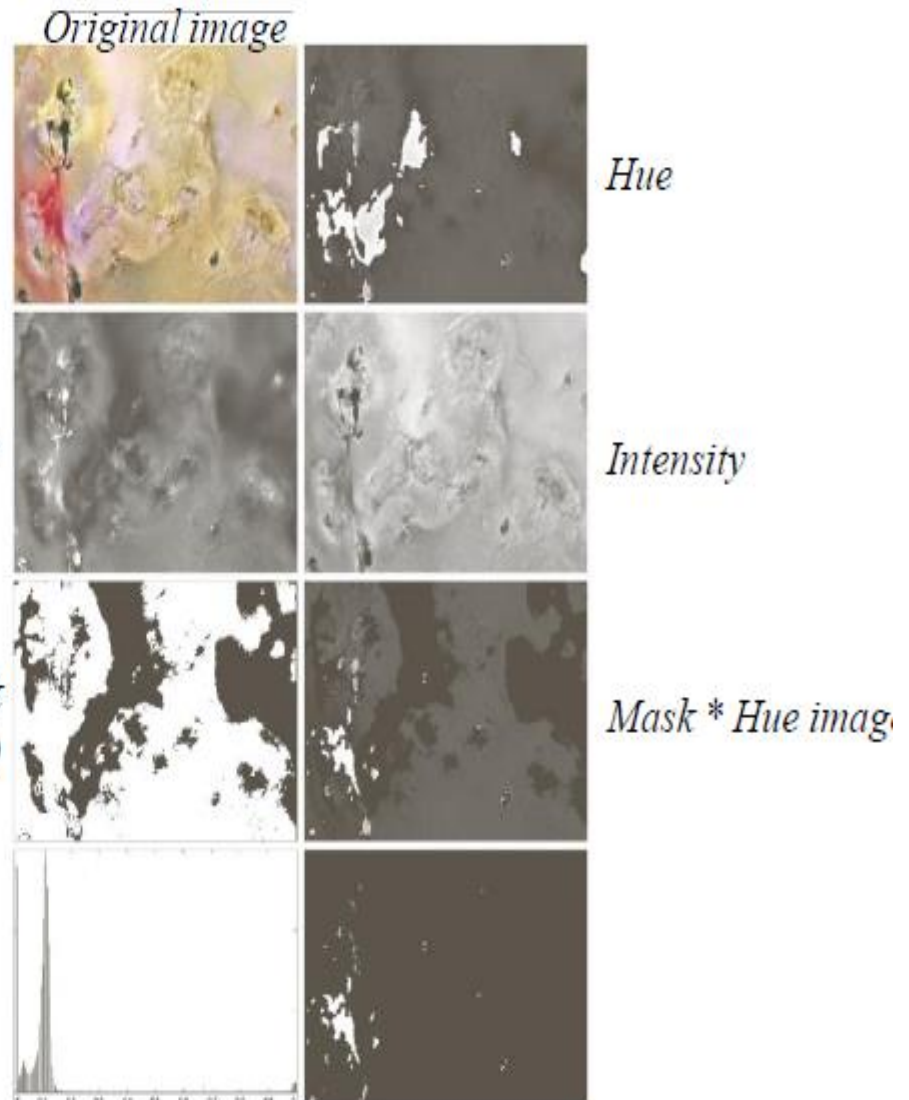
Image Segmentation Based on Colour

Segmentation in HSI Colour Space

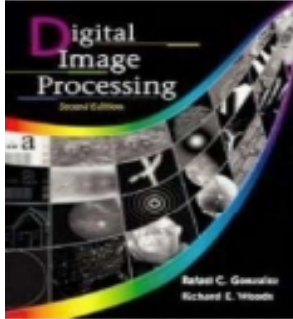
Typically: segmentation on
Hue image

Example:

*Binary saturation mask
(threshold=10% of max value)*



END of Chapter 6



Digital Image Processing
Digital Image Processing Using Matlab

Digital Image Processing

Chapter 7: Wavelets

Prepared by:

Dr. Ali J. Abboud

University of Diyala

2012-2013

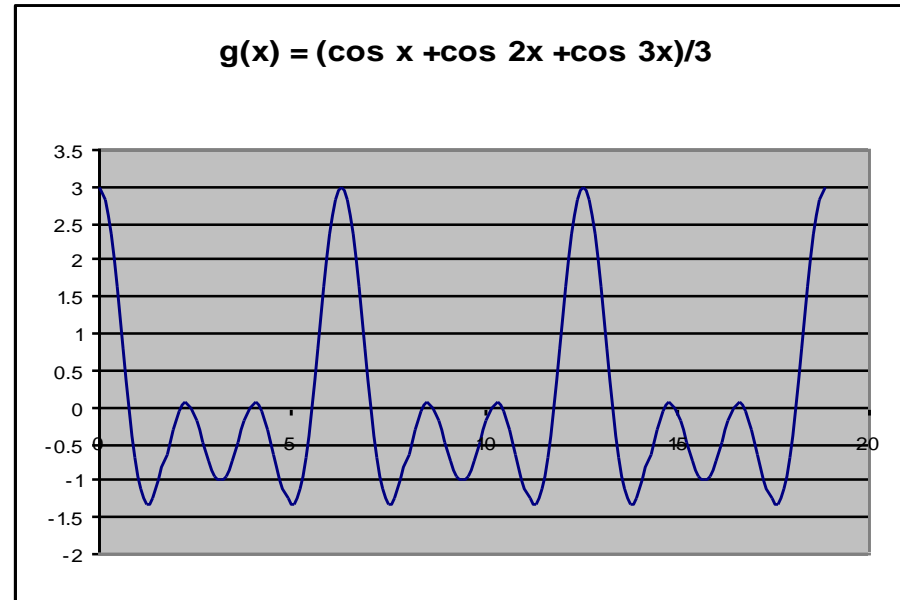
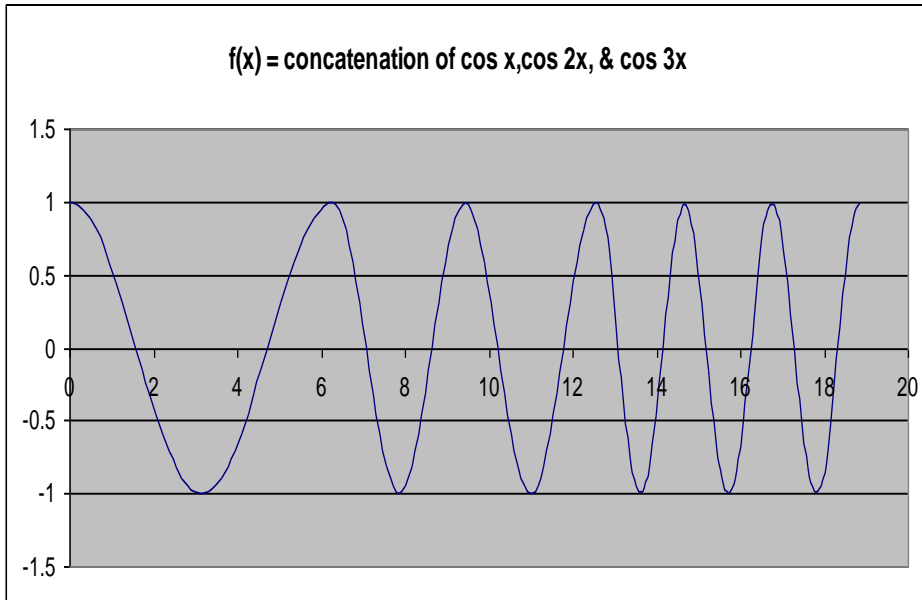


Key Features of Chapter 7:

- **Fourier Analysis – Shortcomings.**
- **Wavelet Transforms.**
- **CWT and DWT.**
- **One Dimension (1D) DWT.**
- **Multi-Resolution 2D Wavelet Transforms.**
- **Different Decomposition Schemes.**
- **Statistical Properties of Wavelet subbands.**
- **Applications of Wavelet Transforms.**
- **DWT Filters Types in Matlab.**
- **DWT Functions in Matlab.**

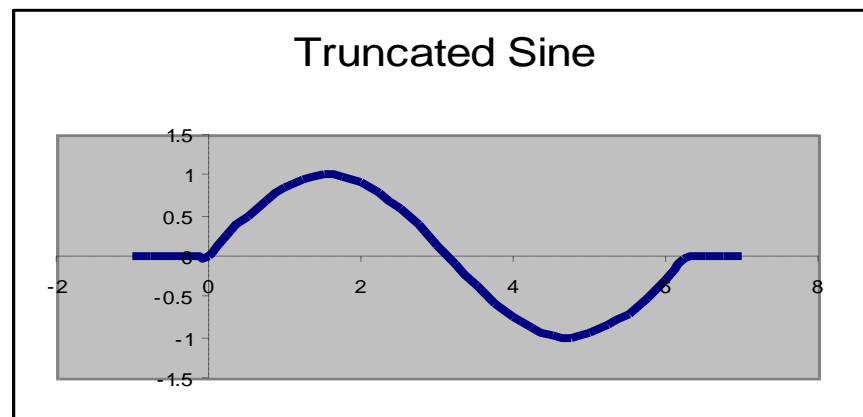
7.1 Fourier Analysis – Shortcomings

- Fourier analysis of the two signals, below, give the same answer
- Thus FT does not provide spatial support, i.e. cannot provide frequency and time/space information simultaneously



7.2 Short-Time Fourier Transform (STFT)

To overcome the above problem, STFT uses the Fourier transform on small window.



- It maps a signal into a two-dimensional function of time and frequency, i.e. provides some information about both when and at what frequencies a signal event occurs.
- The drawback is that once you choose a particular size for the time window, it remains the same for all frequencies. Many signals require a more flexible approach -- one where we can vary the window size.
- Wavelet Transforms represents the next logical step: a windowing technique with variable-sized regions.

7.3 Wavelet Transforms

- Wavelet analysis allows the use of long time intervals where we want more precise low-frequency information, and shorter regions where we want high-frequency information.
- A **wavelet** (i.e. small wave) is a mathematical function used to analyze a continuous-time signal into different frequency components and study each component with a resolution that matches its scale.
- A wavelet transform is the representation of a function by wavelets. The wavelets are **scaled** and **translated** copies of a finite-length or fast-decaying oscillating waveform $\psi(t)$, known as the *mother wavelet*.
- There are many wavelet filters to choose from.

7.3 Wavelet Transforms

- WT provides powerful insight into an image's spatial and frequency characteristics. However, The FT exposes only an image's frequency attributes.
- WT became the preferred image transform for a various reasons:
 - Localization.
 - Lossless Transform.
 - Multi-resolution Characteristics.
- **Some WT applications for a variety of image processing/analysis:**
 - Feature preserving of image/video quantization for compression.
 - Content based video retrieval.
 - Feature extraction for face detection.
 - Image watermarking and steganography (i.e. information security).
 - Object authentication\recognition.

7.4 CWT and DWT

- The continuous wavelet transform (CWT) of $f(t)$ is defined as the sum over all time multiplied by a scaled, shifted versions of the wavelet function $\psi(t)$:

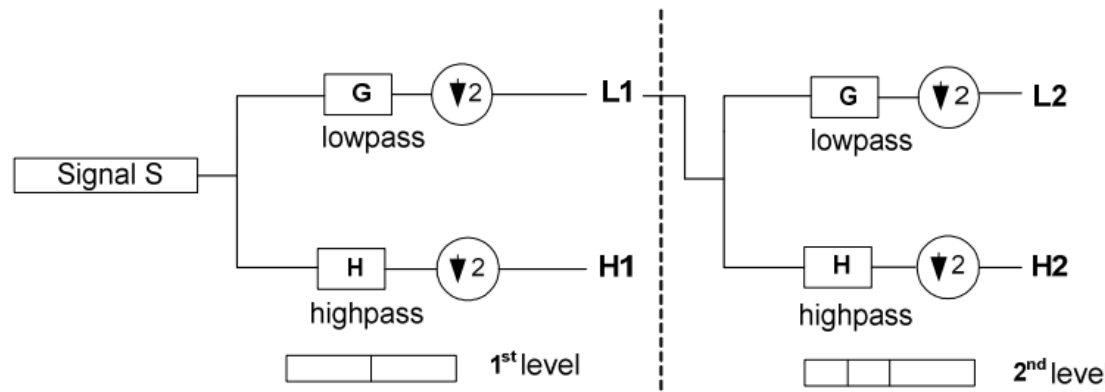
$$C(\text{scale}, \text{position}) = \int f(x) \psi(\mathbf{s}, \mathbf{p}, \mathbf{t}) dt.$$

where $\psi(\mathbf{s}, \mathbf{p}, \mathbf{t})$ is the scaled and mother wavelet. The best choice for the scale and position is to use multiple powers of 2.

- Computing CWT is rather inefficient. The Discrete wavelet transform (DWT) however provides a compact representation of a signal's frequency components with strong spatial support

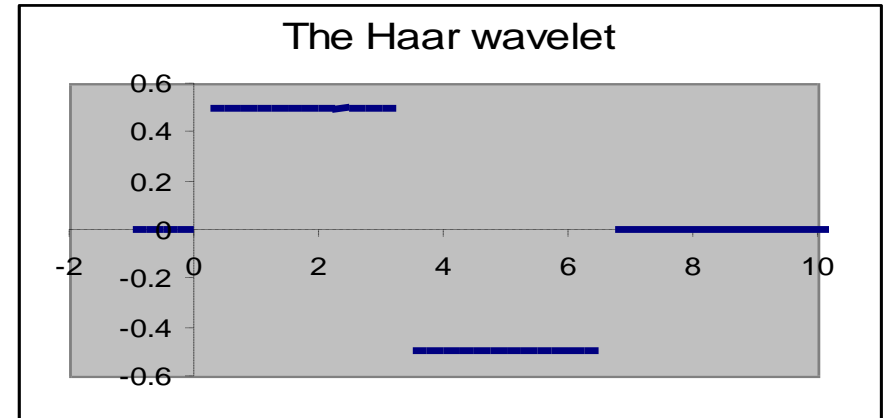
7.5 One Dimension (1D) DWT

- The Wavelet transform is a short time analysis tool of finite energy quasi-stationary signals at multi-resolutions.
- The Discrete wavelet transform (DWT) provide a compact representation of a signal's frequency components with strong spatial support.
- DWT decomposes a signal into frequency subbands at different scales from which it can be perfectly reconstructed.
- For example one diemension wavelet transform is shown below



7.5.1 Example: The Haar Wavelet Filter

- The *Haar wavelet* is a discontinuous, and resembles a step function.
- It is a crude version of the Truncated cosine.
- It can be implemented using a simple filter:



If $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ is a time-signal of length 8, then the Haar wavelet decomposes X into an *approximation* subband containing the **Low** frequencies and a *detail* subband containing the **high** frequencies:

$$\text{Low} = \{x_2 + x_1, x_4 + x_3, x_6 + x_5, x_8 + x_7\} / \sqrt{2}$$

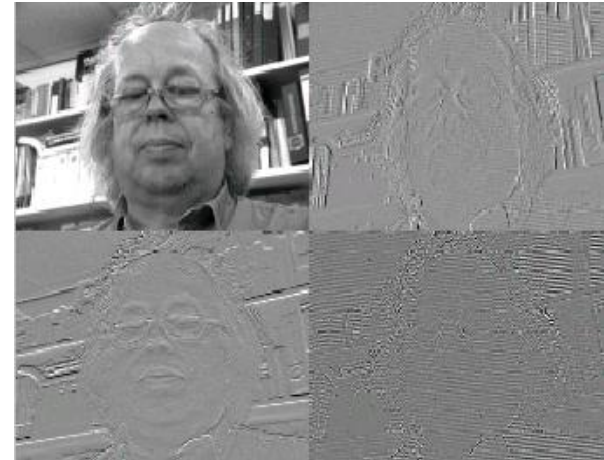
$$\text{High} = \{x_2 - x_1, x_4 - x_3, x_6 - x_5, x_8 - x_7\} / \sqrt{2}$$

7.6 Multi-Resolution 2D Wavelet Transforms

A *Haar* wavelet decompose images first on the rows and then on the columns resulting in 4 subbands, the LL-subband which an approximation of the original image while the other subbands contain the missing details

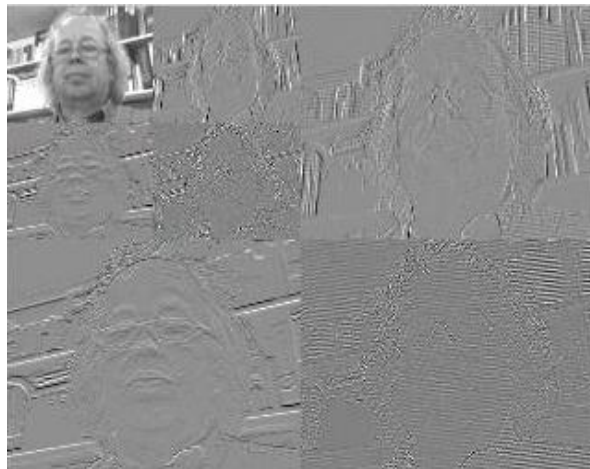
The LL-subband output from any stage can be decomposed further.

**Original
Image**



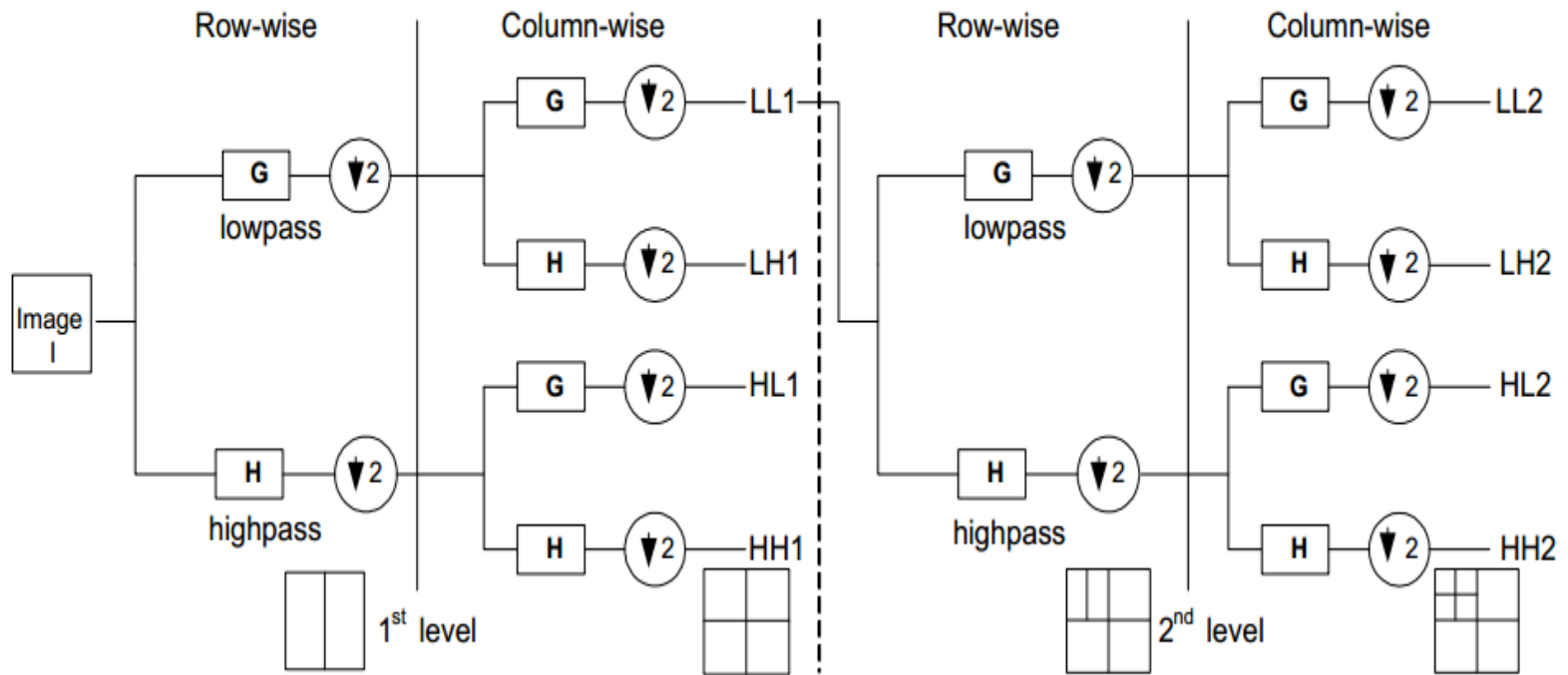
1 stage Transformation

After 2 stages



...

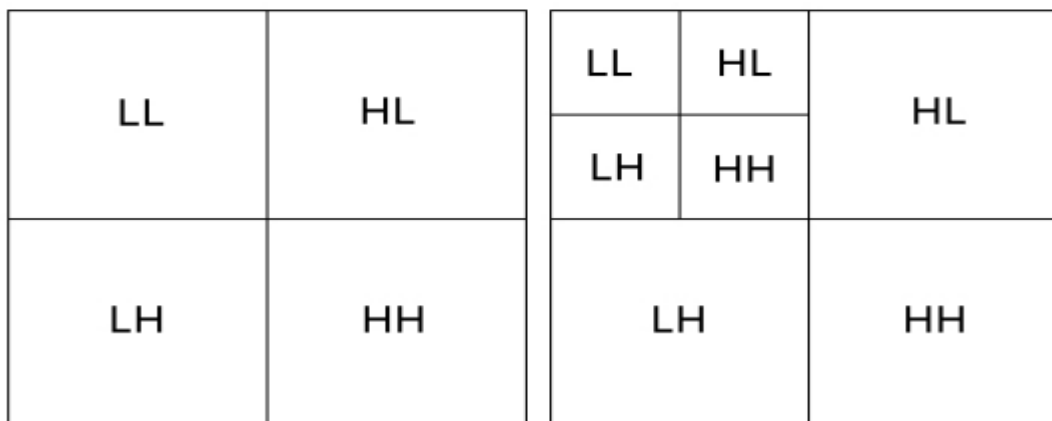
7.6 Multi-Resolution 2D Wavelet Transforms



7.6 Multi-Resolution 2D Wavelet Transforms

- LL sub-band: Representing the low frequencies in both horizontal and vertical directions. This sub-band is also called the approximation or the scaled sub-band.
- LH sub-band: Representing the high frequencies in the horizontal direction and the low frequencies in the vertical one. Hence, this sub-band holds information about horizontal features (e.g. edges) in the image.
- HL sub-band: Representing the high frequencies in the vertical direction and the low frequencies in the horizontal one. It holds information about vertical features in the image.
- HH sub-band: Representing the highest frequencies in the image, and holds information about diagonal features in the image.

7.6. Multi-Resolution Wavelet Transforms - Example 1



(a) Decomposition stage 1

(b) Decomposition stage 2



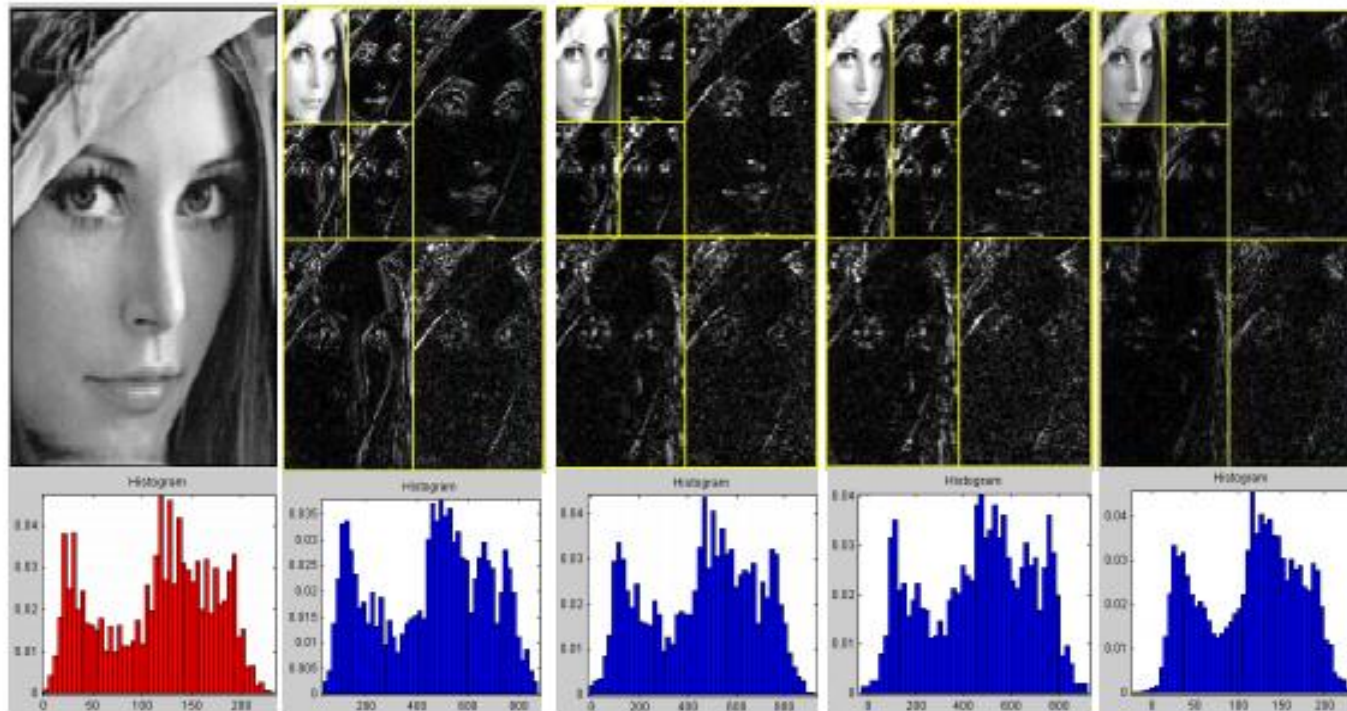
(a) Original

(b) Stage 1

(c) Stage 2

(d) Stage 3

7.6. Multi-Resolution Wavelet Transforms – Example2



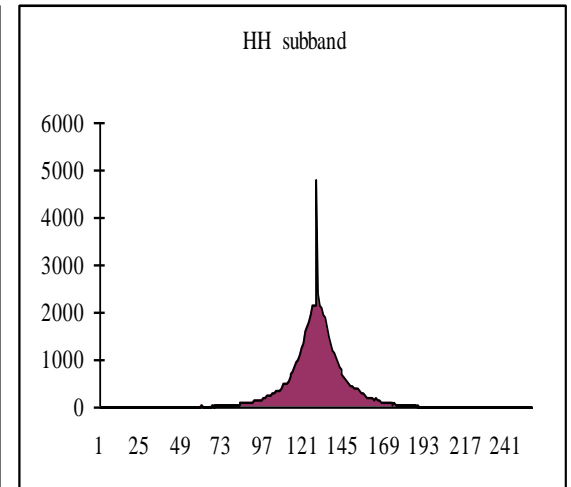
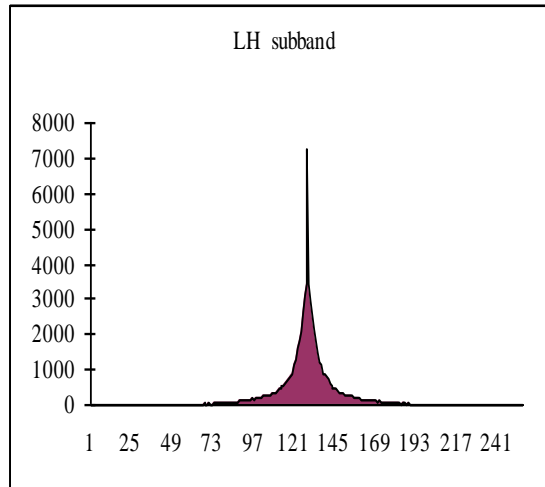
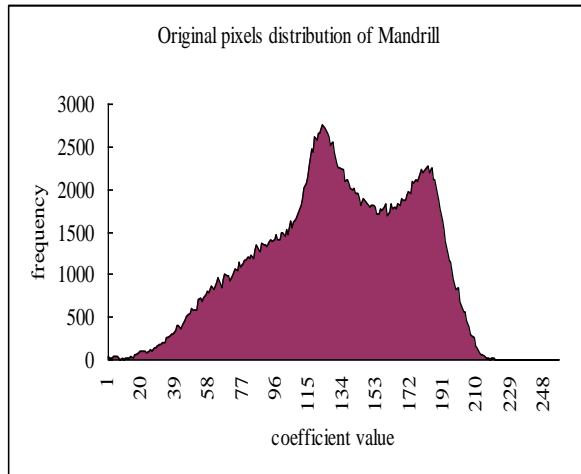
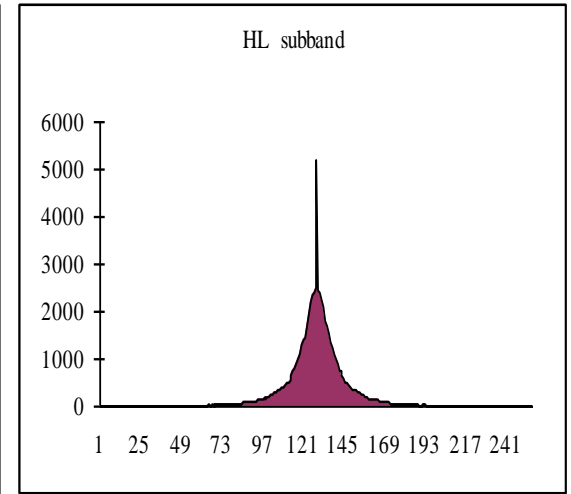
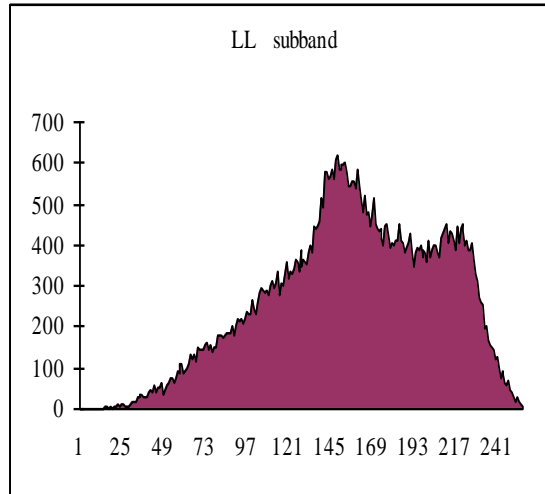
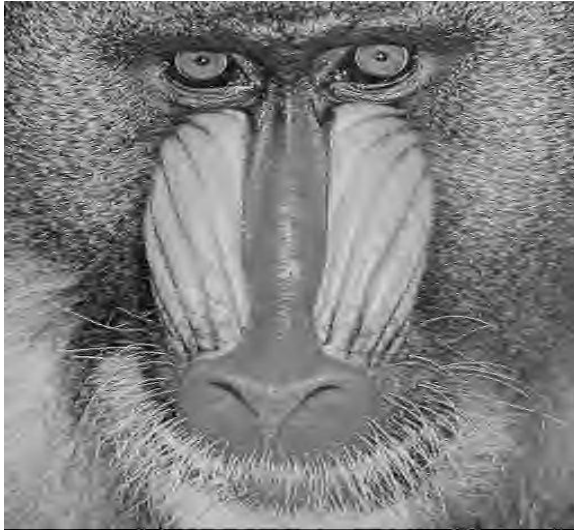
A	B	C	D	E
F	G	H	I	J

Figure 3.7 Example of applying different wavelet filters (db1, db2, db3 and Meyer) for B, C, D and E respectively, where (G, H, I and J shows the histogram for the corresponding LL2 sub-bands).

7.7 Different Decomposition Schemes

- The previous 2 decomposition scheme is known as the Pyramid scheme, whereby at successive stages only the LL subband is wavelet transformed.
- Other decomposition schemes include:
 - The *standard* scheme – At every stage all the image is wavelet transformed
 - The *wavelet packet* – After stage 1, a non-LL subband is transformed only if it satisfied certain condition.

7.8. Statistical Properties of Wavelet subbands



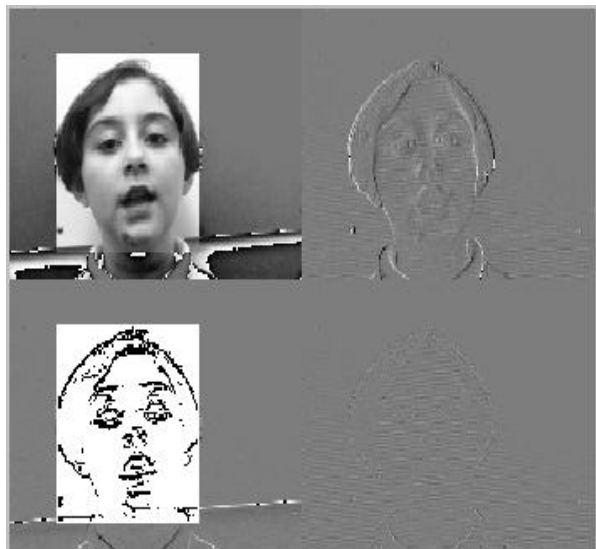
The distribution of the LL-subband approximate that of the original but all non-LL subbands have a Laplacian distribution. This remains valid at all depths₆

7.9. Applications of Wavelet Transforms

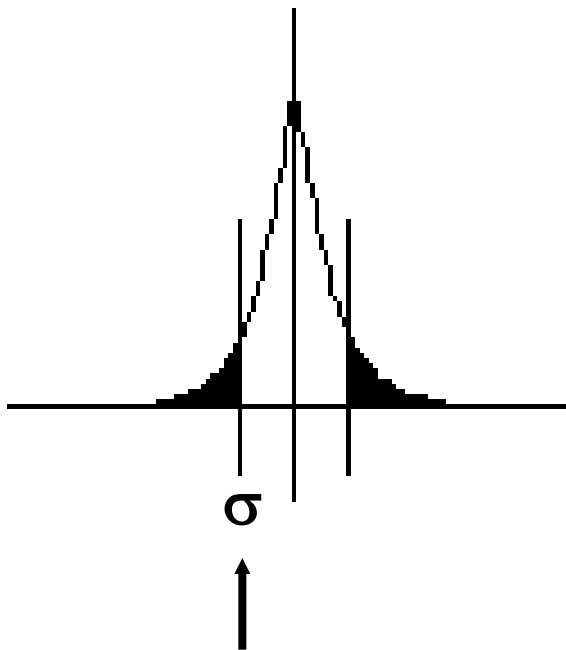
- The list of applications is growing fast. These include:
 - Image and video Compression
 - Feature detection and recognition
 - Image denoising
 - Face Recognition
 - Signal interpolation
- Most applications benefit from the statistical property of the non-LL subbands (*The laplacian distribution of the wavelet coefficients in these subbands*).

7.9.1 Wavelet-based Feature Detection

- Non-LL subbands of a wavelet decomposed image contains high frequencies (i.e. image features) which are highlighted. These significant coefficients are the furthest away from the mean.
- Thresholding reveals the main features.



Horizontal features



Vertical features

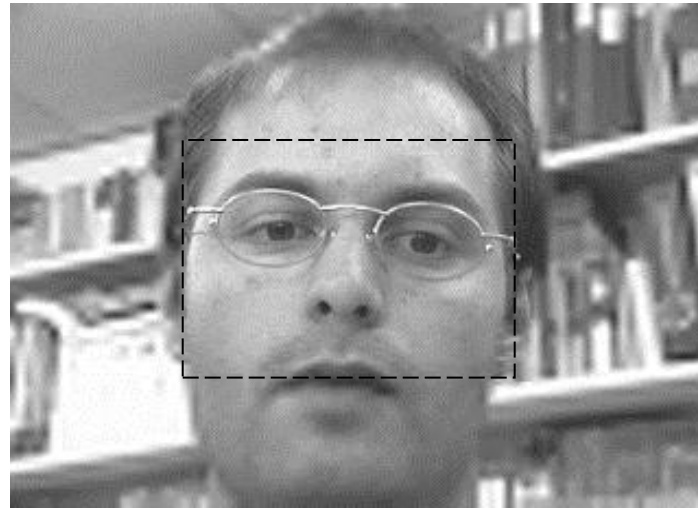
7.9.2 Image and Video Compression

- Compression is done in 4 steps:
 1. Transform into frequency domain (Fourier or Wavelet) the image to the required depth
 2. Quantise each subband according to the required ratio
 3. Create a code Entropy-based code book - the more frequent quantised coefficients have shorter representation than the others.
 4. Encode the coefficients
- Fourier based compression suffer from blocking effect at high compression rate.
- JPEQ2000 uses wavelet transforms

7.9.3. Wavelet-based Video compression schemes



Original



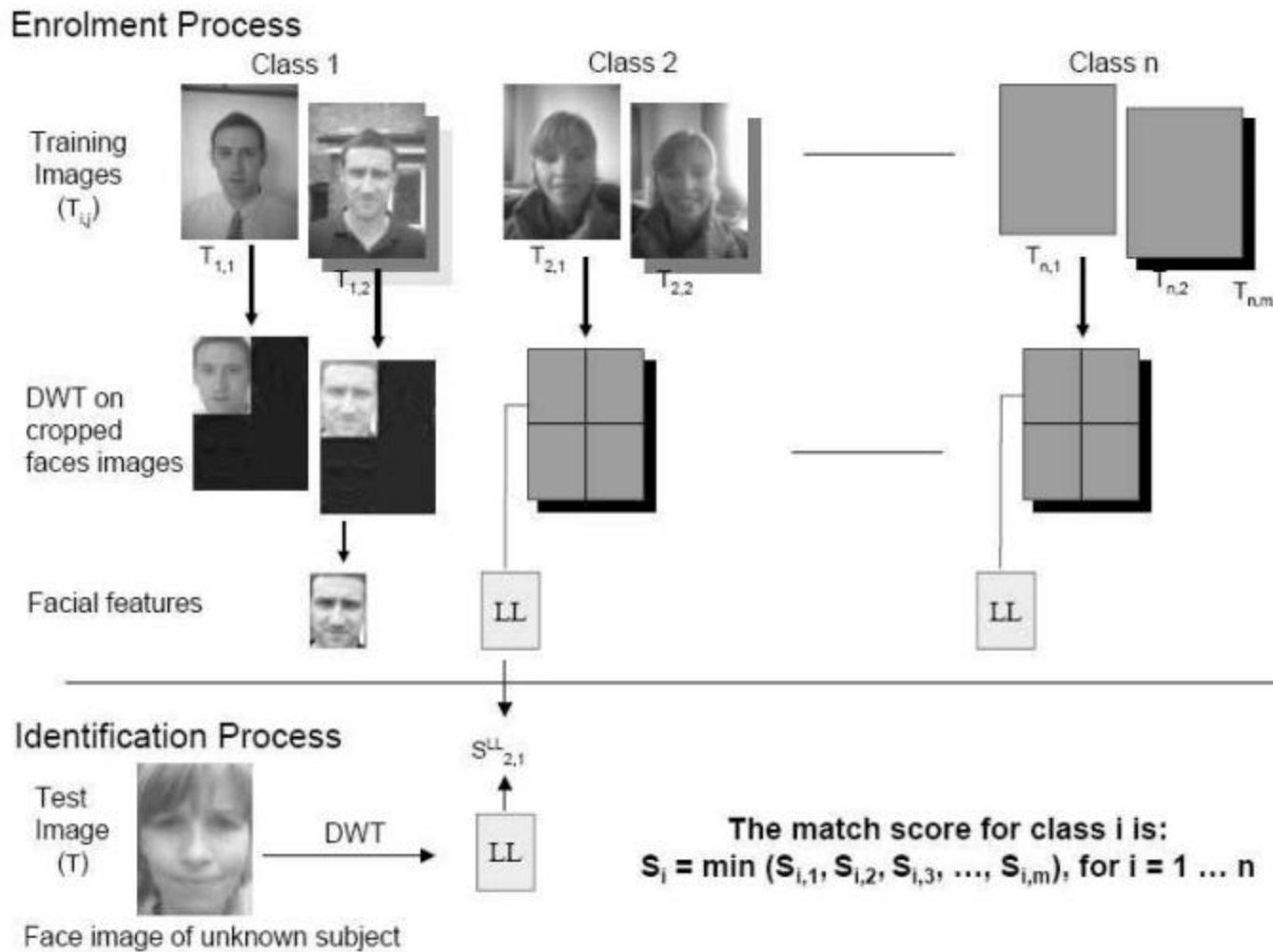
A Region of Interest (ROI) scheme



A feature-preserving (FP) scheme

In the ROI scheme, the non-LL coefficients are not calculated outside the region. For FP, significant LL-coefficients are quantised finely at the expense of others.

7.9.4. Wavelet-based Face Identification



7.10 Haar Wavelet – in MATLAB

```
File Edit Text Cell Tools Debug Desktop Window Help
[Icons] [B...] >> [Icons]
1 - clear all
2 - c=imread('ImageA.bmp');
3 - imshow(c);
4 - f=c;
5 - [m n]=size(c);
6 - k=n/2; %Wavelet tranform (rows)
7 - for i=1:1:m
8 -     for j=1:1:k
9 -         f(i,j)=c(i,2*j)+c(i,2*j-1);
10 -        f(i,j+k)=c(i,2*j)-c(i,2*j-1);
11 -     end
12 - end
13 - c=f;
14 - k=m/2; %Wavelet tranform (columns)
15 - for j=1:1:n
16 -     for i=1:1:k
17 -         c(i,j)=uint8((f(2*i,j)+f(2*i-1,j))/2);
18 -         c(i+k,j)=uint8((f(2*i,j)-f(2*i-1,j))/2);
19 -     end
20 - end
21 - figure; imshow(c)
```

7.11 DWT Types in Matlab

Wavelet	wfamily	wname
Haar	'haar'	'haar'
Daubechies	'db'	'db2', 'db3', ..., 'db45'
Coiflets	'coif'	'coif1', 'coif2', ..., 'coif5'
Symlets	'sym'	'sym2', 'sym3', ..., 'sym45'
Discrete Meyer	'dmey'	'dmey'
Biorthogonal	'bior'	'bior1.1', 'bior1.3', 'bior1.5', 'bior2.2', 'bior2.4', 'bior2.6', 'bior2.8', 'bior3.1', 'bior3.3', 'bior3.5', 'bior3.7', 'bior3.9', 'bior4.4', 'bior5.5', 'bior6.8'
Reverse Biorthogonal	'rbio'	'rbio1.1', 'rbio1.3', 'rbio1.5', 'rbio2.2', 'rbio2.4', 'rbio2.6', 'rbio2.8', 'rbio3.1', 'rbio3.3', 'rbio3.5', 'rbio3.7', 'rbio3.9', 'rbio4.4', 'rbio5.5', 'rbio6.8'

7.11 DWT Functions in Matlab

Analysis-Decomposition Functions

Function Name	Purpose
dwt2	Single-level decomposition
wavedec2	Decomposition
wmaxlev	Maximum wavelet decomposition level

Synthesis-Reconstruction Functions

Function Name	Purpose
idwt2	Single-level reconstruction
waverec2	Full reconstruction
wrcoef2	Selective reconstruction
upcoef2	Single reconstruction

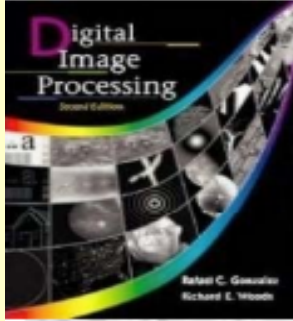
Decomposition Structure Utilities

Function Name	Purpose
detcoef2	Extraction of detail coefficients
appcoef2	Extraction of approximation coefficients
upwlev2	Recomposition of decomposition structure

De-Noising and Compression

Function Name	Purpose
ddencmp	Provide default values for de-noising and compression
wbpn	Penalized threshold for wavelet 1-D or 2-D de-noising
wcbm2	Thresholds for wavelet 2-D using Birgé-Massart strategy
wdencmp	Wavelet de-noising and compression
wthrmngr	Threshold settings manager

END of Chapter 7



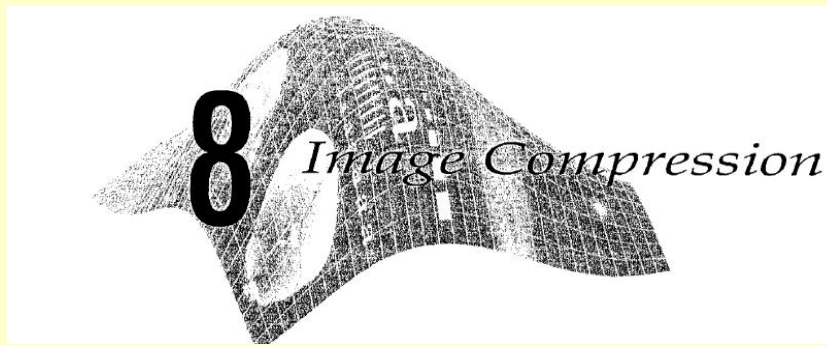
Digital Image Processing

Digital Image Processing Using Matlab

Chapter 8

Image Compression: Part1

Prepared by:
Dr. Ali J. Abboud



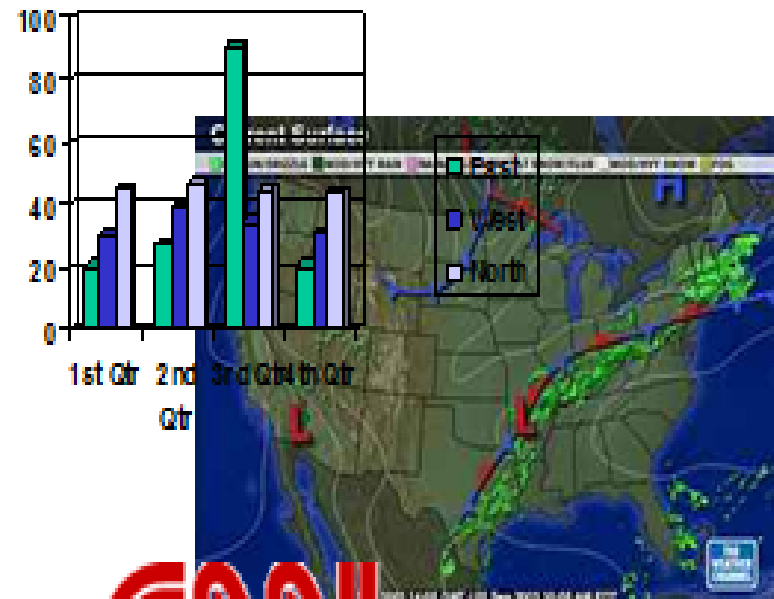
Key Features of Chapter 8:

- **Redundancy in Images.**
 1. **Coding Redundancy.**
 2. **Spatial and Temporal Redundancy.**
 3. **Irrelevant Information.**
- **Measuring Image Information.**
- **Fidelity Criteria.**
- **Image Compression Models.**
- **Image Formats and Compression Standards**
- **Compression Methods: Huffman Coding.**

Introduction

- Everyday an enormous amount of information is stored, processed, and transmitted

- Financial data
- Reports
- Inventory
- Cable TV
- Online Ordering and tracking



Introduction

- Because much of this information is graphical or pictorial in nature, the storage and communications requirements are immense.
- Image compression addresses the problem of reducing the amount of data requirements to represent a digital image.
- Image Compression is becoming an enabling technology: HDTV.
- Also it plays an important role in Video Conferencing, remote sensing, satellite TV, FAX, document and medical imaging.

Introduction

The size of typical still image (1200x1600)

$$1200 \times 1600 \times 3 \text{ byte} = 5760000 \text{ byte}$$
$$= 5,760 \text{ Kbyte} = 5.76 \text{ Mbyte}$$

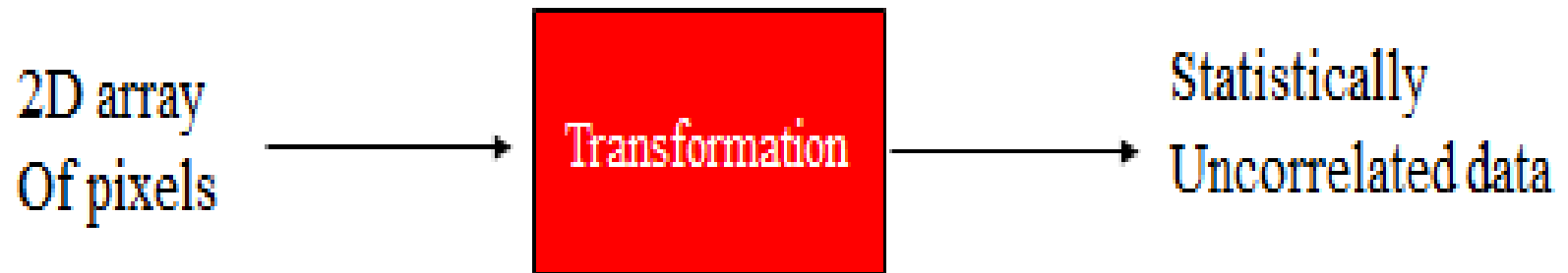
The size of two hours standard television (720x480) movies

$$30 \frac{\text{frame}}{\text{sec}} \times (760 \times 480) \frac{\text{pixels}}{\text{frame}} \times 3 \frac{\text{bytes}}{\text{pixel}} = 31,104,000 \text{ bytes / sec}$$

$$31,104,000 \times \frac{\text{bytes}}{\text{sec}} \times (60 \times 60) \frac{\text{sec}}{\text{hour}} \times 2 \text{ hours} = 2.24 \times 10^{11} \text{ bytes}$$
$$= 224 \text{ GByte.}$$

Introduction

- We want to remove redundancy from the data
- Mathematically



Introduction

- The term data compression refers to the process of reducing the amount of data required to represent a given quantity of information
- Data \neq Information
- Various amount of data can be used to represent the same information
- Data might contain elements that provide no relevant information :
data redundancy
- Data redundancy is a central issue in image compression. It is not an abstract concept but mathematically quantifiable entity

Redundancy in Images

Data, Information, and Redundancy

- **Information**
- **Data** is used to represent information
- **Redundancy** in data representation of an information provides no relevant information or repeats a stated information
- Let n_1 , and n_2 are data represents the same information. Then, the relative data redundancy R of the n_1 is defined as
$$R = 1 - 1/C \quad \text{where } C = n_1/n_2$$

Redundancy in Images

- Let n_1 and n_2 denote the number of information carrying units in two data sets that represent the same information
- The relative redundancy R_D is define as :

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

where C_R commonly called the compression ratio, is

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

Redundancy in Images

- If $n_1 = n_2$, $C_R = 1$ and $R_D = 0$ \longrightarrow *no redundancy*
- If $n_1 \gg n_2$, $C_R \rightarrow \infty$ and $R_D \rightarrow 1$ \longrightarrow *high redundancy*
- If $n_1 \ll n_2$, $C_R \rightarrow 0$ and $R_D \rightarrow -\infty$ \longrightarrow *undesirable*

- A compression ratio of 10 (10:1) means that the first data set has 10 information carrying units (say, bits) for every 1 unit in the second (compressed) data set.

- In Image compression, 3 basic redundancy can be identified
 - » *Coding Redundancy*
 - » *Interpixel Redundancy*
 - » *Psychovisual Redundancy*

Redundancy in Images

- Redundancy in Digital Images
 - Coding redundancy
usually appear as results of the uniform representation of each pixel
 - Spatial/Temopral redundancy
because the adjacent pixels tend to have similarity in practical.
 - Irrelevant Information
Image contain information which are ignored by the human visual system.

Redundancy in Images

- Redundancy in Digital Images
 - Coding redundancy
 - usually appear as results of the uniform representation of each pixel
 - Spatial/Temopral redundancy
 - because the adjacent pixels tend to have similarity in practical.
 - Irrelevant Information
 - Image contain information which are ignored by the human visual system.

Redundancy in Images



Coding Redundancy



Spatial Redundancy



Irrelevant Information

Coding Redundancy

- Assume the discrete random variable for r_k in the interval $[0,1]$ that represent the gray levels. Each r_k occurs with probability p_k
- If the number of bits used to represent each value of r_k by $l(r_k)$ then

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p(r_k)$$

- The average code bits assigned to the gray level values.
- The length of the code should be inverse proportional to its probability (occurrence).

Coding Redundancy

- Assume the discrete random variable for r_k in the interval $[0,1]$ that represent the gray levels. Each r_k occurs with probability p_k
- If the number of bits used to represent each value of r_k by $l(r_k)$ then

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p(r_k)$$

- The average code bits assigned to the gray level values.
- The length of the code should be inverse proportional to its probability (occurrence).

Coding Redundancy

Examples of variable length encoding

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

Coding Redundancy

- Recall from the histogram calculations

$$p(r_k) = \frac{h(r_k)}{n} = \frac{n_k}{n}$$

where $p(r_k)$ is the probability of a pixel to have a certain value r_k

If the number of bits used to represent r_k is $l(r_k)$, then

$$L_{av} = \sum_{k=0}^{L-1} l(r_k)(p(r_k))$$

Coding Redundancy

- Example:

$$L_{av} = \sum_{k=0}^7 l(r_k)(p(r_k))$$

$$= 2(0.19) + 2(0.25) + 3(0.16) + \dots + 6(0.02)$$

$$= 2.7 \text{ bits}$$

$$C_R = \frac{3}{2} = 1.11$$

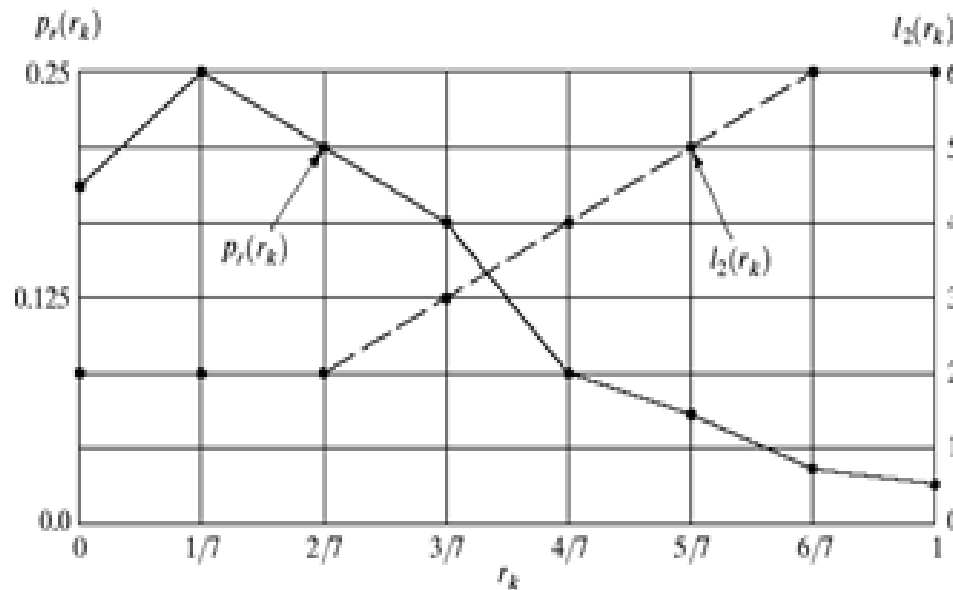
$$R_D = 1 - \frac{1}{1.11} = 0.099$$

r_k	$p(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

Coding Redundancy

r_k	$p_i(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

Variable-Length Coding



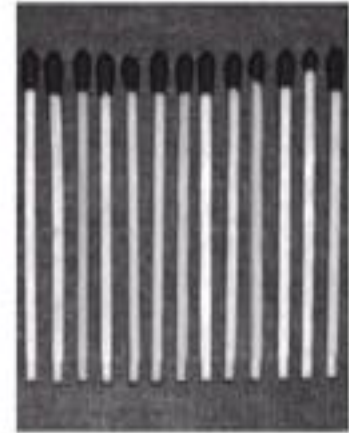
Graphic representation of the fundamental basis of data compression through variable-length coding.

Spatial/ Temporal redundancy

- Internal Correlation between the pixel result from
 - Respective Autocorrelation
 - Structural Relationship
 - Geometric Relationship
- The value of a pixel can be reasonably predicted from the values of its neighbors.
- To reduce the inter-pixel redundancies in an image the 2D array is transformed (*mapped*) into more efficient format (Frequency Domain etc.)

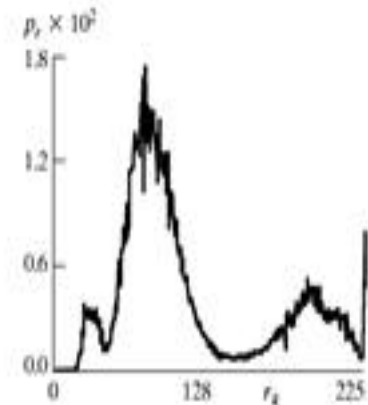
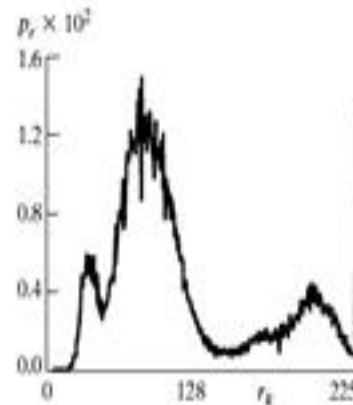
Spatial/ Temporal redundancy

Here the two pictures have
Approximately the same
Histogram.



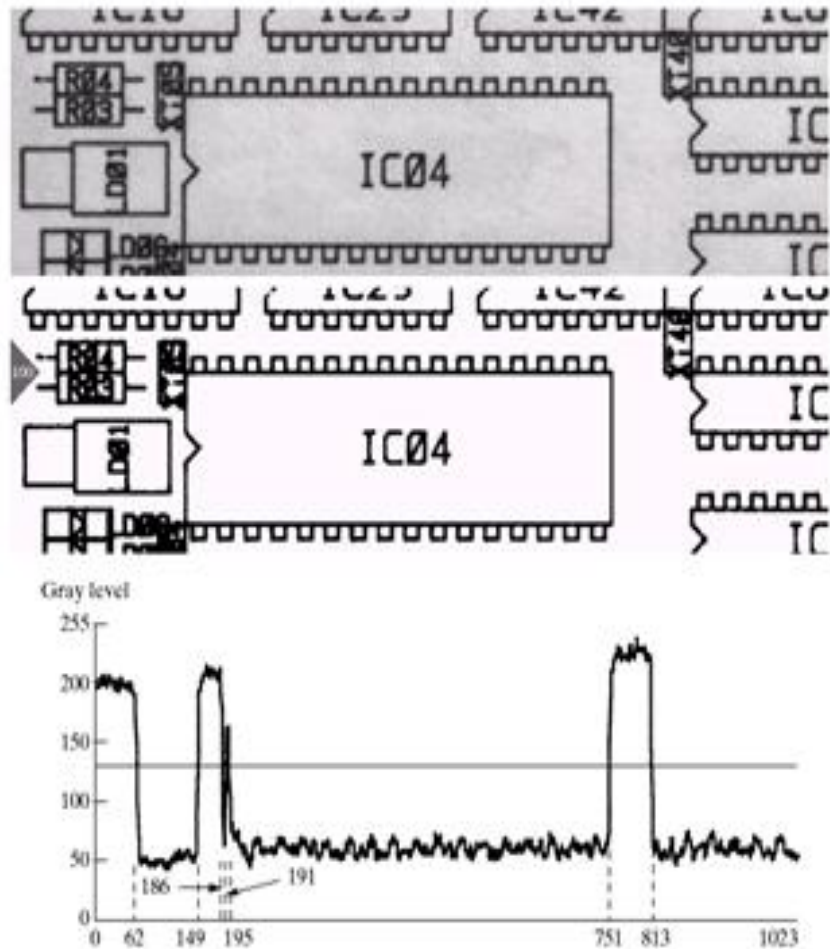
We must exploit Pixel
Dependencies.

Each pixel can be estimated
From its neighbors.



Spatial/ Temporal redundancy

Example of Inter-pixel
Redundancy removal



Psycho-visual Redundancy

Irrelevant information and Psycho-Visual Redundancy

- The brightness of a region depend on other factors that the light reflection
- The perceived intensity of the eye is limited an non linear
- Certain information has less relative importance that other information in normal visual processing
- In general, observer searches for distinguishing features such as edges and textural regions.

Psycho-visual Redundancy

The human visual system is more sensitive to edges

Middle Picture:

Uniform quantization from 256 to 16 gray levels

$$C_R = 2$$

Right picture:

Improved gray level quantization (IGS)

$$C_R = 2$$



Measuring Information

- A random event E that occurs with probability $P(E)$ is said to contain $I(E)$ information where $I(E)$ is defined as $I(E) = \log(1/P(E)) = -\log(P(E))$
- $P(E) = 1$ contain no information
- $P(E) = 1/2$ requires one bit of information.

Measuring Information

- For a source of events $a_0, a_1, a_2, \dots, a_k$ with associated probability $P(a_0), P(a_1), P(a_2), \dots, P(a_k)$.
- The average information per source (entropy) is

$$H = -\sum_{j=0}^k P(a_j) \log(P(a_j))$$

For image, we use the normalized histogram to generate the source probability, which leads to the entropy

$$\tilde{H} = -\sum_{i=0}^{L-1} p_r(r_i) \log(p_r(r_i))$$

Fidelity Criteria

The error between two functions is given by:

$$e(x, y) = \hat{f}(x, y) - f(x, y)$$

So, the total error between the two images is

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

The root-mean-square error averaged over the whole image is

$$e_{rms} = \frac{1}{MN} \sqrt{[\hat{f}(x, y) - f(x, y)]^2}$$

Fidelity Criteria

- A closely related objective fidelity criterion is the mean square signal to noise ratio of the compressed-decompressed image

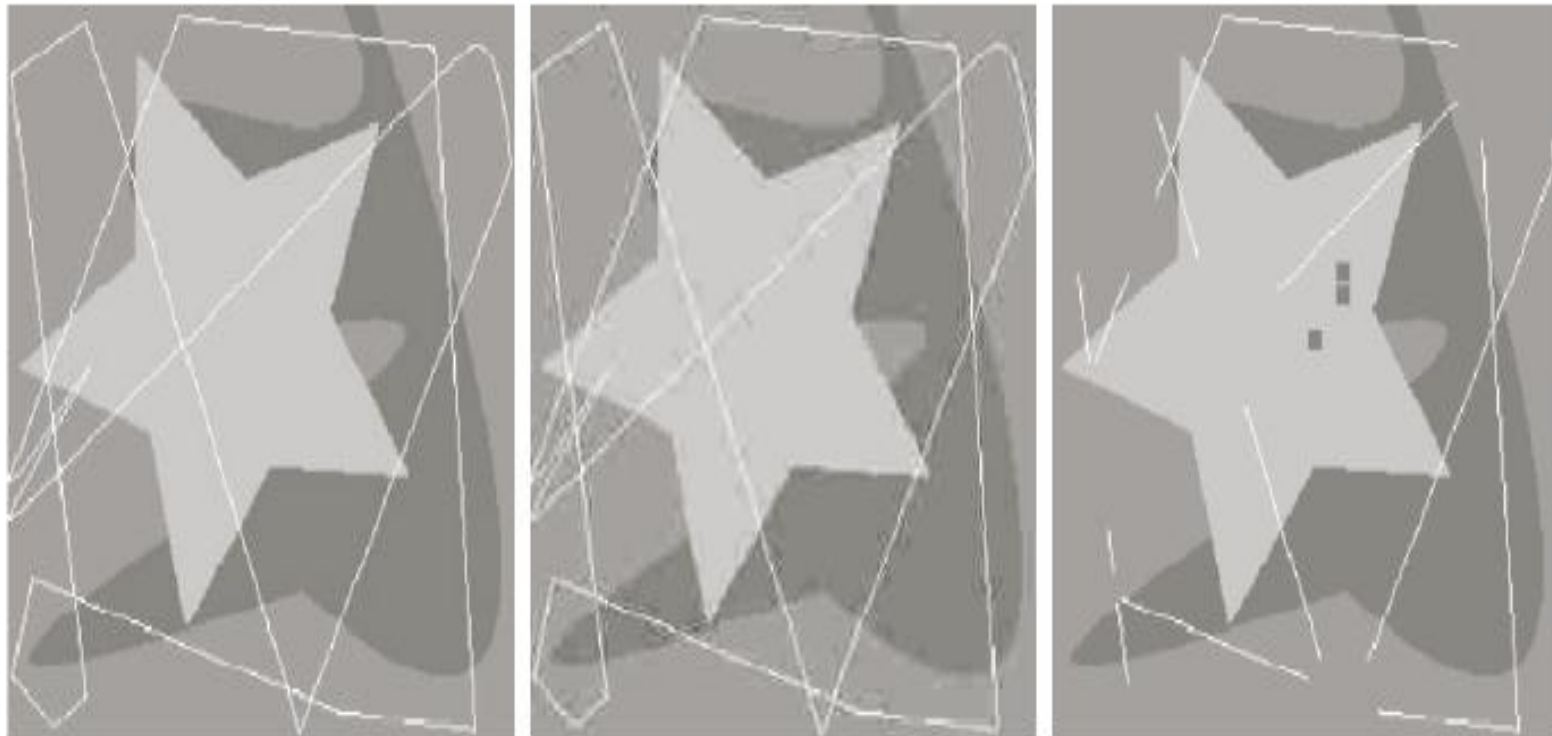
$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x, y) - f(x, y)]^2}$$

Fidelity Criteria

Rating scale of the
Television
Allocations Study
Organization.
(Frendendall and
Behrend.)

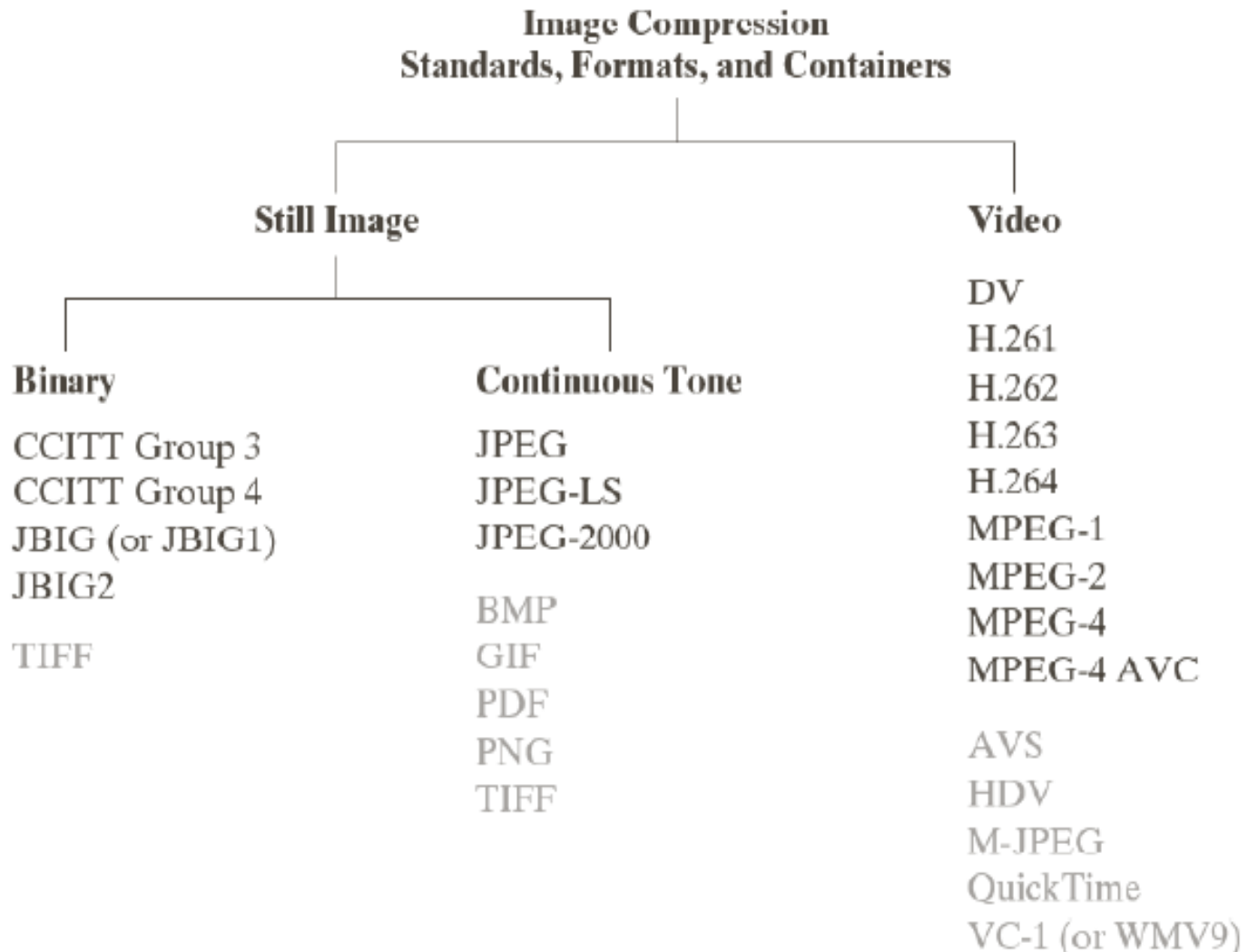
Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor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

Fidelity Criteria

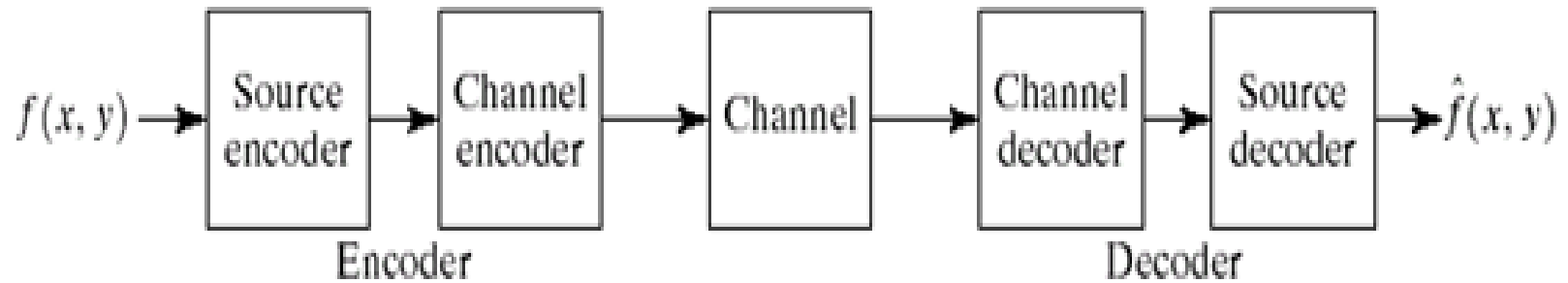


Three approximations of the same image

Compression Standards



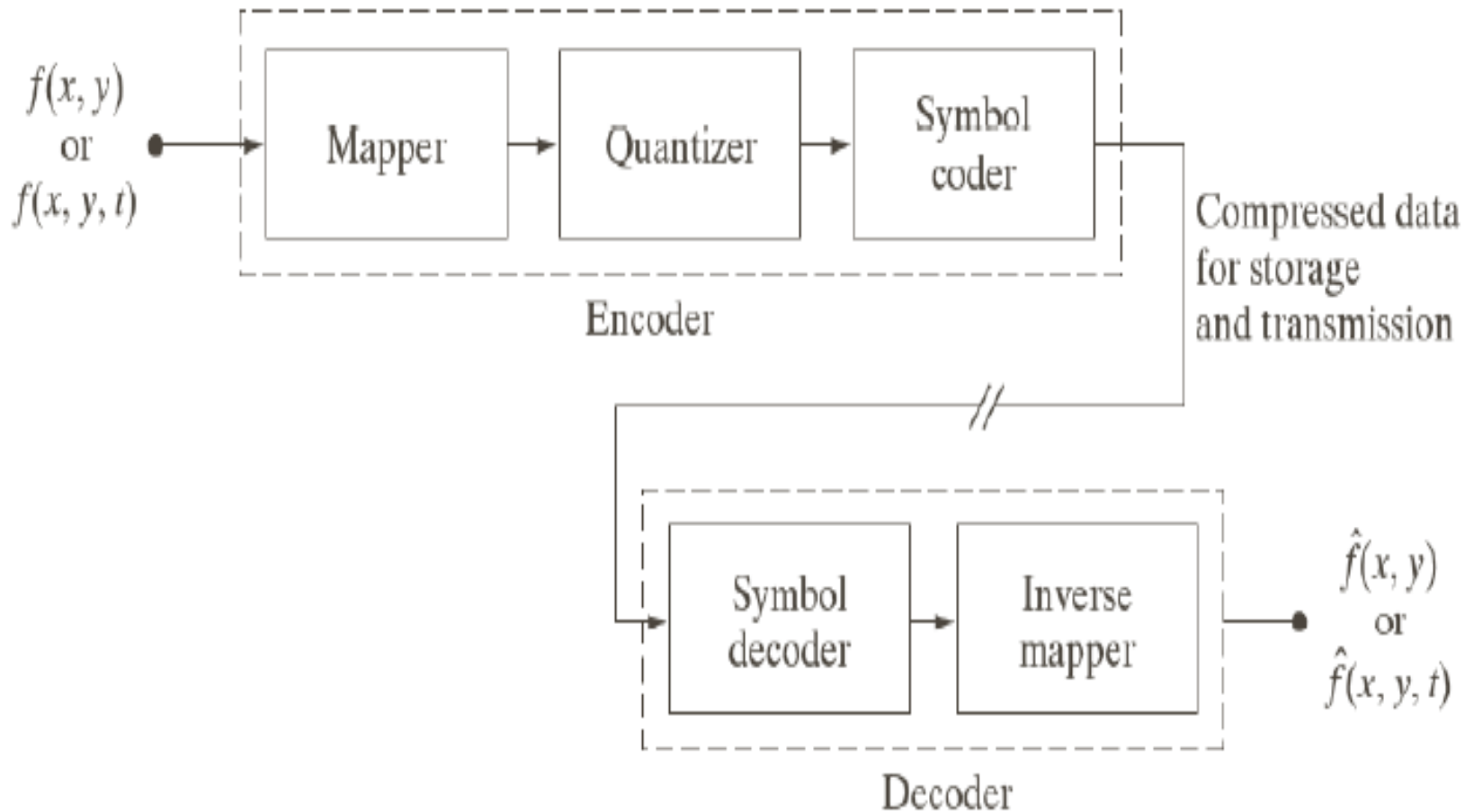
Compression Model



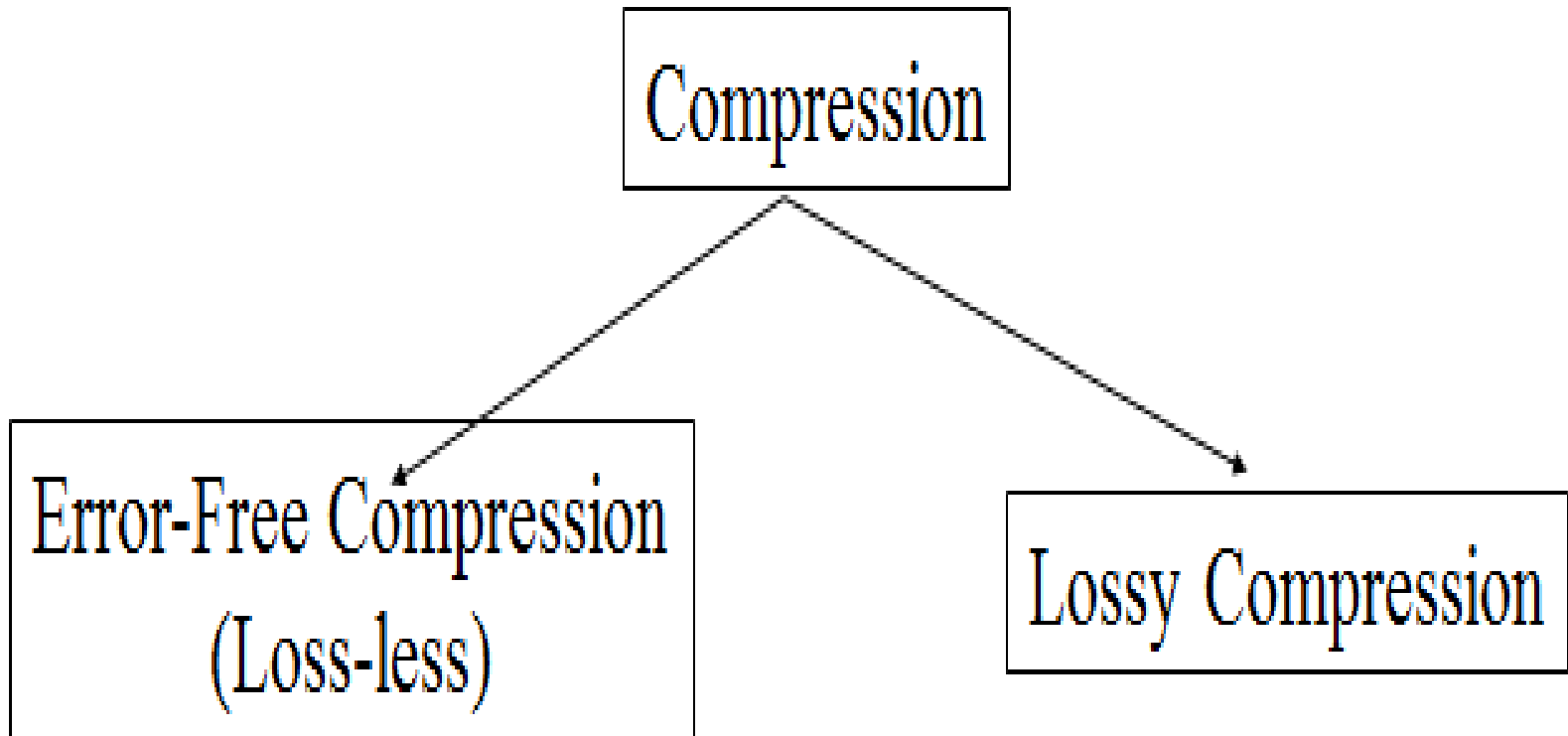
The source encoder is responsible for removing redundancy (coding, inter-pixel, psycho-visual)

The channel encoder ensures robustness against channel noise.

Compression Model



Compression Types

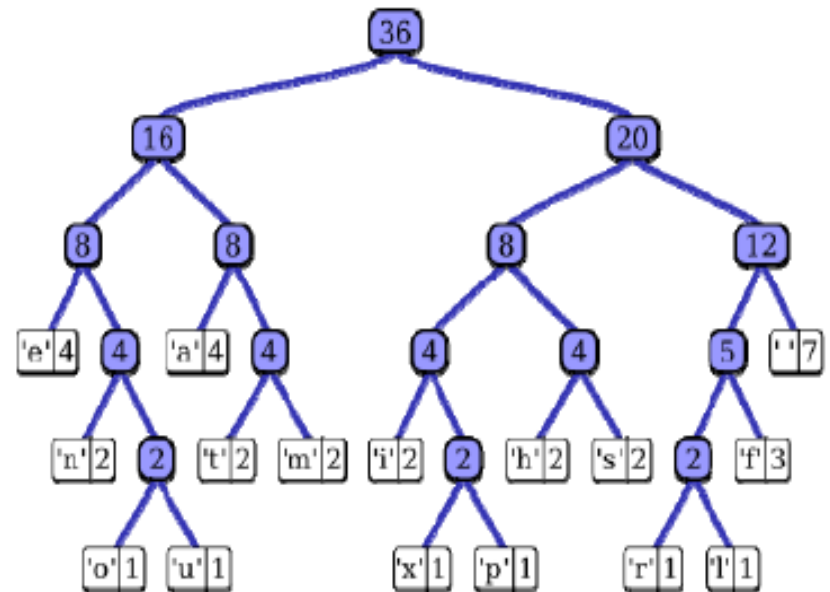


Error-Free Compression

- Some applications require no error in compression (medical, business documents, etc..)
- $C_R=2$ to 10 can be expected.
- Make use of coding redundancy and inter-pixel redundancy.
- Ex: Huffman codes, LZW, Arithmetic coding, 1D and 2D run-length encoding, Loss-less Predictive Coding, and Bit-Plane Coding.

Method1: Huffman Coding

Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.



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

Method1: Huffman Coding

- The most popular technique for removing coding redundancy is due to Huffman (1952)
- Huffman Coding yields the smallest number of code symbols per source symbol
- The resulting code is *optimal*

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

Method1: Huffman Codes

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

$$L_{avg} = 1(0.4) + 2(0.3) + 3(0.1) + 4(0.1) + 5(0.06) + 5(0.04)$$

$$= 2.2 \text{ bits}$$

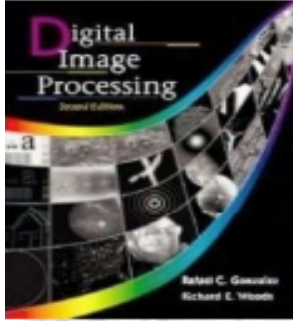
Method1: Huffman Codes

Huffman coding

Assignment procedure

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

END of Chapter 8 : Part1

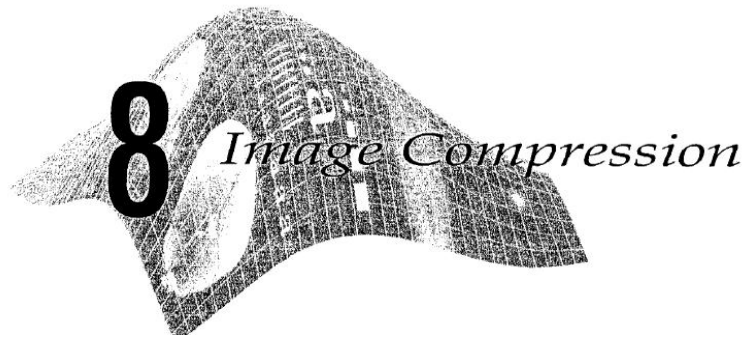


Digital Image Processing
Digital Image Processing Using Matlab

Chapter 8

Image Compression: Part2

Prepared by:
Dr. Ali J. Abboud



Key Features of Chapter 8

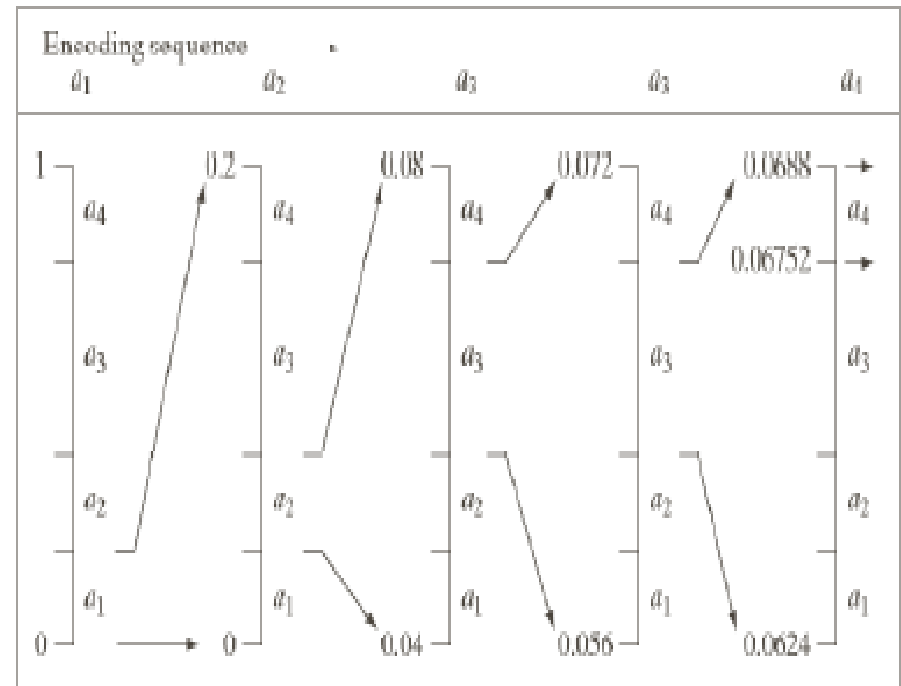
- **Compression Methods**
 1. **Huffman Coding.**
 2. **Arithmetic Coding.**
 3. **LZW Coding.**
 4. **Bit-plane Coding.**
 5. **Run Length Coding.**
 6. **Symbol-based Coding.**
- **Tutorials**

2. Arithmetic Coding

Arithmetic coding is a form of variable-length entropy encoding.

A string is converted to arithmetic encoding, usually characters are stored with fewer bits

Arithmetic coding encodes the entire message into a single number, a fraction n where $(0.0 \leq n < 1.0)$.



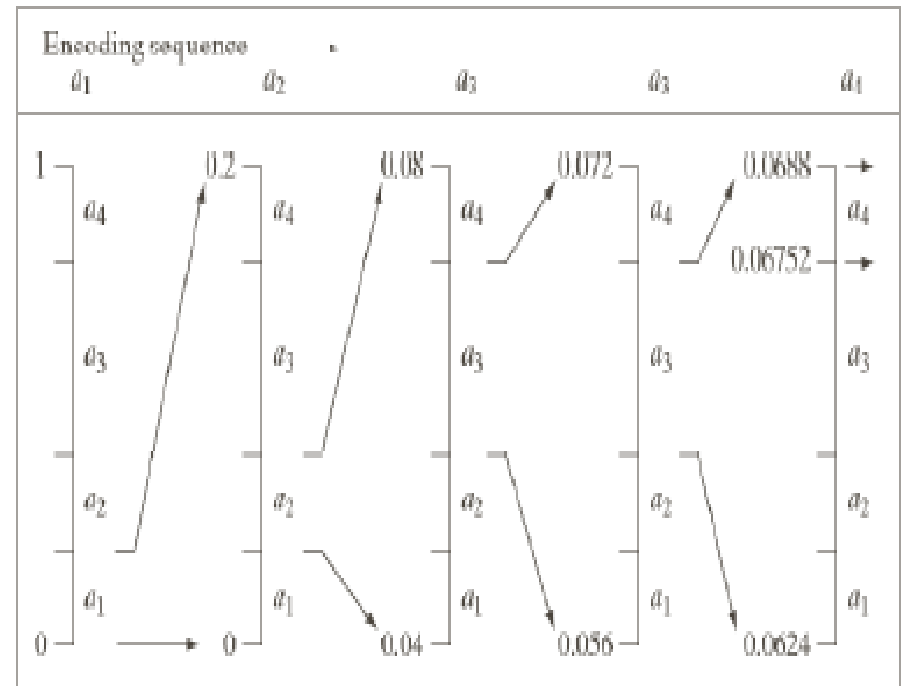
Source Symbol	Probability	Initial Subinterval
a_1	0.2	$[0.0, 0.2)$
a_2	0.2	$[0.2, 0.4)$
a_3	0.4	$[0.4, 0.8)$
a_4	0.2	$[0.8, 1.0)$

2. Arithmetic Coding

Arithmetic coding is a form of variable-length entropy encoding.

A string is converted to arithmetic encoding, usually characters are stored with fewer bits

Arithmetic coding encodes the entire message into a single number, a fraction n where $(0.0 \leq n < 1.0)$.



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)

3. LZW Coding

- Lempel-Ziv-Welch (LZW) coding assigns fixed-length code words to variable length sequences of source symbols but requires no a priori knowledge of the probability of occurrence of the symbols to be encoded.
- LZW compression has been integrated into a variety of mainstream imaging file formats, including the *graphic interchange format* (GIF), *tagged image file format* (TIFF), and the *portable document format* (PDF)
- At the onset of the coding process, a codebook or “dictionary” containing the source symbols to be coded is constructed.
- For 8-bit monochrome images, the first 256 words of the dictionary are assigned to the gray value 0,1,2,...,255.

3. LZW Coding

- As the encoder sequentially examines the image's pixels, gray-level sequences that are not in the dictionary are placed in algorithmically determined (e.g., the next unused) locations.
- If the first two pixels of the image are white, for instance, sequence "255-255" might be assigned to location 256, the address following the locations reserved for gray levels 0 through 255.
- The next time that two consecutive white pixels are encountered, code word 256, the address of the location containing sequence 255-255, is used to represent them.
- If a 9-bit, 512-word dictionary is employed in the coding process, the original (8+8) bits that were used to represent the two pixels are replaced by a single 9-bit code word.

3. LZW Coding

- The size of the dictionary is an important system parameter.
- If it is too small, the detection of matching gray-level sequences will be less likely.
- If it is too large, the size of the code words will adversely affect compression performance.
- If a 9-bit, 512-word dictionary is employed in the coding process, the original (8+8) bits that were used to represent the two pixels are replaced by a single 9-bit code word.

Ex. A 9-bit, 512-word dictionary is employed in the coding process.

Consider 4x4, 8-bit image of a vertical edge.

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

3. LZW Coding

The starting content of 512-word dictionary is:

Dictionary Location	Entry
0	0
1	1
...	...
255	255
256	-
...	...
511	-

Locations 256 through 511 are initially unused.

3. LZW Coding

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

TABLE 8.7

LZW coding example.

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

- At the conclusion of coding, the dictionary contains 265 code words and the LZW algorithm has successfully identified several repeating gray-level sequences – leveraging them to reduce the original 128-bit image to 90 bits (i.e., 10 9-bit codes).
- The image is encoded by processing its pixels in a left-to-right, top-to-bottom manner.
- Nine additional code words are defined.

3. LZW Coding

- The resulting compression ration is 1.42:1
- Most practical applications require a strategy for handling “dictionary overflow”.
 - simple solution is to flush or reinitialize the dictionary when it becomes full and continue coding with a new initialized dictionary.
 - a more complex option is to monitor compression performance and flush the dictionary when it becomes poor or unacceptable.
 - alternately, the least used dictionary entries can be tracked and replaced when necessary.

4. Bit-Plane Coding

Another effective technique for reducing an image's *interpixel redundancies* is to process the image's bit planes individually.

Bit-plane decomposition

The gray levels of an m -bit gray-scale image can be represented in the form of the base 2 polynomial

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \dots + a_12^1 + a_02^0$$

m 1-bit *bit planes*.

The inherent **disadvantage** of this approach is that small changes in gray level can have a significant impact on the complexity of the bit planes.

Ex. 127 (01111111) and 128 (10000000)

→ every bit plane contain a corresponding 0 to 1 (or 1 to 0) transition

4. Bit-Plane Coding

An alternative decomposition approach (which reduces the effect of small gray-level variations) is to first represent the image by an *m*-bit Gray code.

The *m*-bit Gray code $g_{m-1} \cdots g_2 g_1 g_0$

$$g_i = a_i \oplus a_{i+1} \quad 0 \leq i \leq m-2$$

$$g_{m-1} = a_{m-1}$$

Gray codes that correspond to 127 and 128 are 01000000 and 11000000, respectively.

4. Bit-Plane Coding



his indenture made two run
his year of our Lord one thousand
and ninety six between Stanley
of Ky. And Bluff of Tennessee
Induced Jackson of the Count
Latt. Aforesaid of the other part
said Stanley Donelson for a
of the sum of two thousand
hand paid the receipt where
with And by these presents
sell alien enoff And Confor
Jackson his heirs And a
certain tracts or parcels of La
sand acres one thousand
more or less being well his

a b

FIGURE 8.14 A

1024 × 1024

(a) 8-bit
monochrome
image and

(b) binary image.

These two images are used to illustrate the compression techniques.

4. Bit-Plane Coding

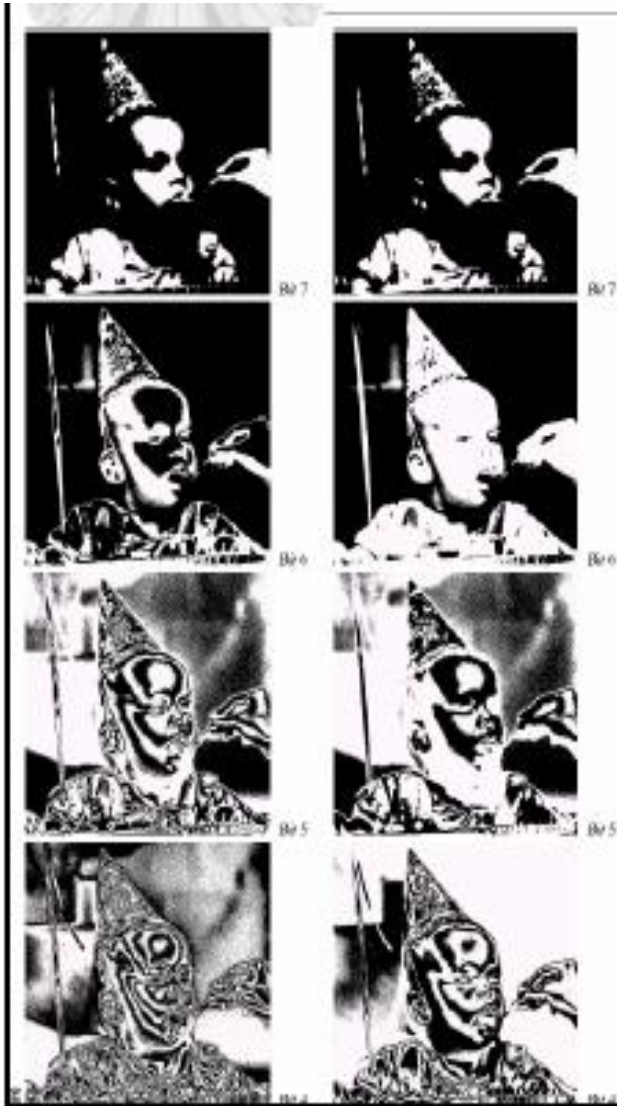


FIGURE 8.15 The four most significant binary (left column) and Gray-coded (right column) bit planes of the image in Fig. 8.14(a).

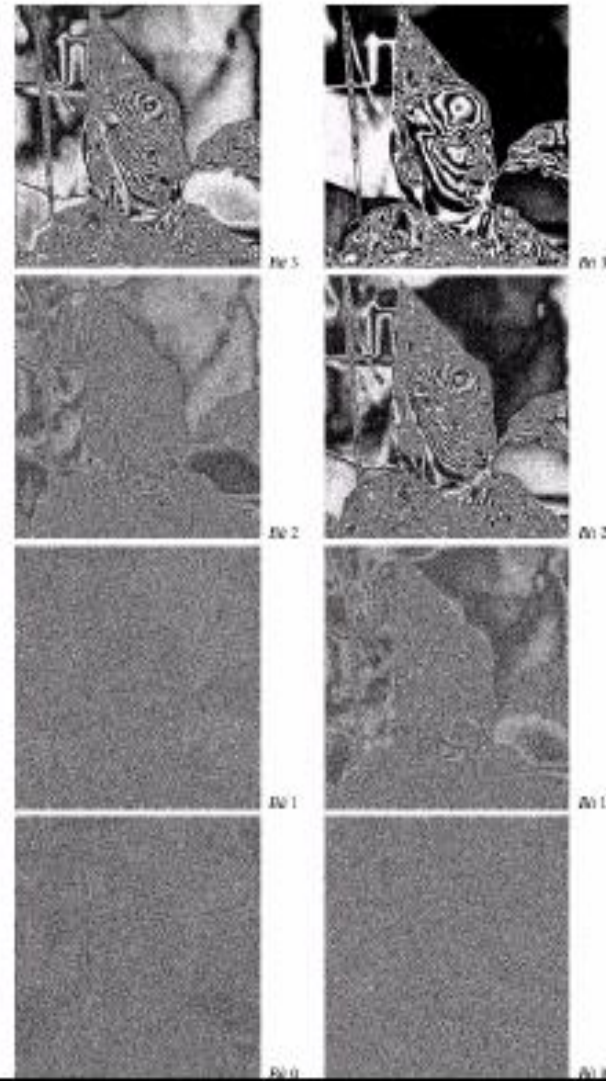


FIGURE 8.16 The four least significant binary (left column) and Gray-coded (right column) bit planes of the image in Fig. 8.14(a).

kiat Wangsiripitak

5. Run Length Coding

- 1-D run-length coding
 - RLC+VLC according to run-lengths statistics
- 2-D run-length coding
 - used for FAX image compression
 - Relative address coding (RAC)
 - based on the principle of tracking the binary transitions that begin and end each black and white run
 - combined with VLC

5. Run Length Coding

One-dimensional run-length coding

represent each row of an image or bit plane by a sequence of lengths that describe successive runs of black and white pixels.

→ *run-length coding*

The basic concept is to code each contiguous group of 0's or 1's encountered in a left to right scan of a row by its length and to establish *a convention for determining the value of the run.*

- (1) to specify the value of the first run of each row, or
- (2) to assume that each row begins with a white run, whose run length may in fact be zero

5. Run Length Coding

- The black and white run lengths may be coded separately using variable-length codes that are specifically tailored to their own statistics.

H_0 : an estimate of the entropy of the black run-length source

H_1 : an estimate of the entropy of the white run-length source

L_0 : the average value of the black run lengths

L_1 : the average value of the white run-lengths

The approximate *run-length entropy* of the image is

$$H_{RL} = \frac{H_0 + H_1}{L_0 + L_1} \quad (8.4-4)$$

Eq. (8.4-4) provides an estimate of the average number of bits per pixel required to code the run lengths in a binary image using a variable-length code.

6. Symbol-based Coding

Symbol compression

This approaches determine a set of symbols that constitute the image, and take advantage of their multiple appearance. It convert each symbol into token, generate a token table and represent the compressed image as a list of tokens.

This approach is good for document images.

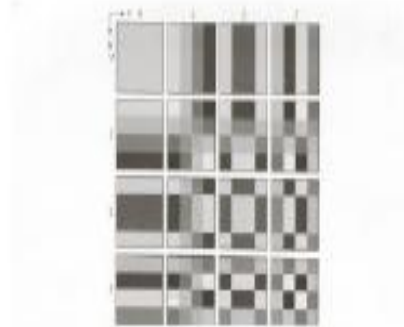


FIGURE 8.30 Discrete cosine basis functions for $N = 4$. The origin of each block is at the top left.

where

$$a(n) = \begin{cases} \frac{1}{\sqrt{N}} & \text{for } n = 0 \\ \frac{2}{\sqrt{N}} \cos\left(\frac{n\pi}{N}\right) & \text{for } n = 1, 2, \dots, N-1 \end{cases} \quad (8.133)$$

FIGURE 8.31 (a) (1) Figure 8.31 shows $g(x, y, n, r)$ for the case $N = 4$. The computation follows the same format as explained for Fig. 8.29, with the difference that the values of g are not integers. In Fig. 8.30, the lighter gray levels correspond to large values of g .

(1) Figures 8.21(a), (c), and (e) show three approximations of the 512×512 monochrome image in Fig. 8.23. These pictures were obtained by dividing the original image into subimages of size 8×8 , representing each subimage using one of the transforms are described (i.e., the DFT, WHT, or DCT transform), truncating 50% of the resulting coefficients, and taking the inverse transform of the truncated coefficient array.

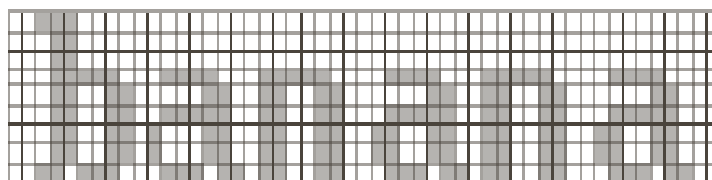
In each case, the 32 retained coefficients were selected on the basis of maximum magnitude. When we disregard any quantization or coding issues, this process amounts to compressing the original image by a factor of 2. Note that in all cases, the 32 discarded coefficients had little visual impact on reconstruction of the image quality. Their elimination, however, was accompanied by some mean square error, which can be seen in the shaded error images of Fig. 8.21(b), (d), and (f). The actual mean errors were 1.20, 0.6, and 0.69 gray levels, respectively.



$$r N = 4. T1$$

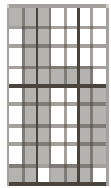
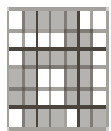
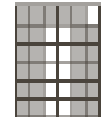
EXAMPLE 8.10
Function coding
with the DFT,
WHT, and DCT

6. Symbol-based Coding



a b c

FIGURE 8.17
 (a) A bi-level document,
 (b) symbol dictionary, and
 (c) the triplets used to locate the symbols in the document.

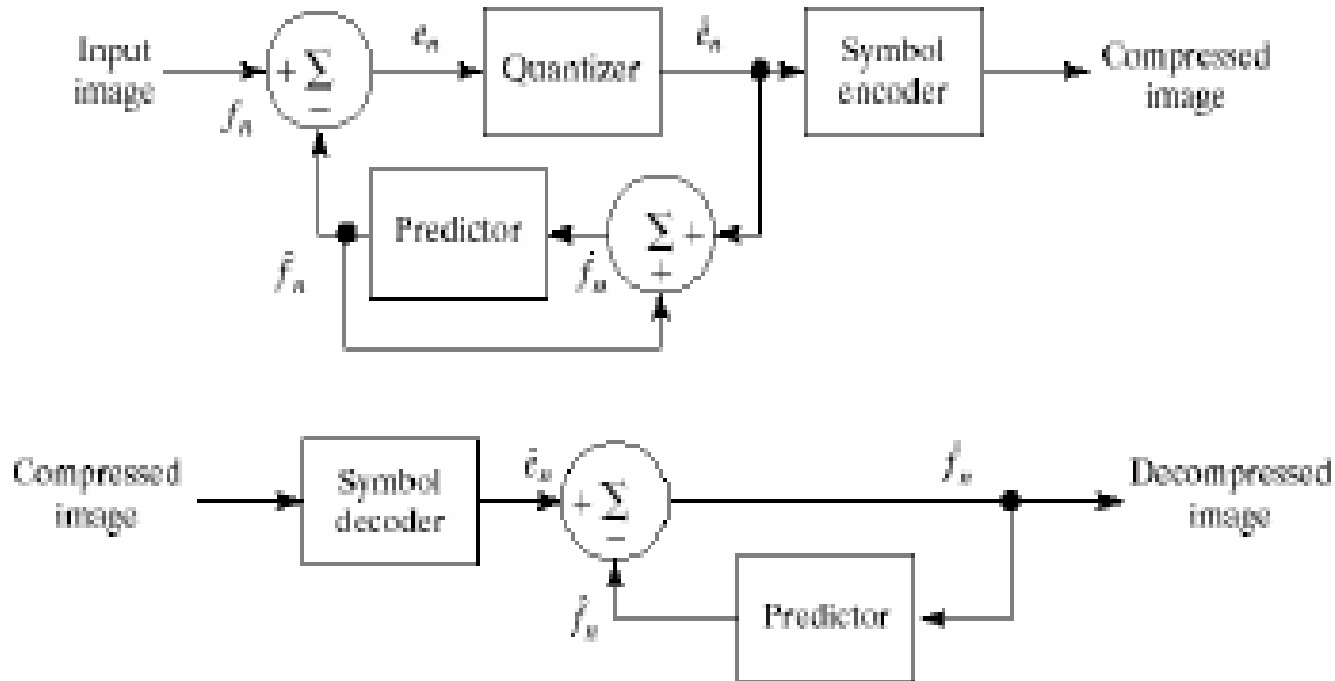
Token	Symbol	Triplet
0		(0, 2, 0) (3, 10, 1) (3, 18, 2)
1		(3, 26, 1) (3, 34, 2) (3, 42, 1)
2		

7. Lossy Compression

- Lossy encoding is based on the concept of *compromising the accuracy* of the reconstructed image in exchange for increased compression.
- If the resulting distortion (which may or may not be visually apparent) can be tolerated, the increase in compression can be significant.

10:1 to 50:1 \rightarrow more than 100:1

8. Lossy Predictive Coding



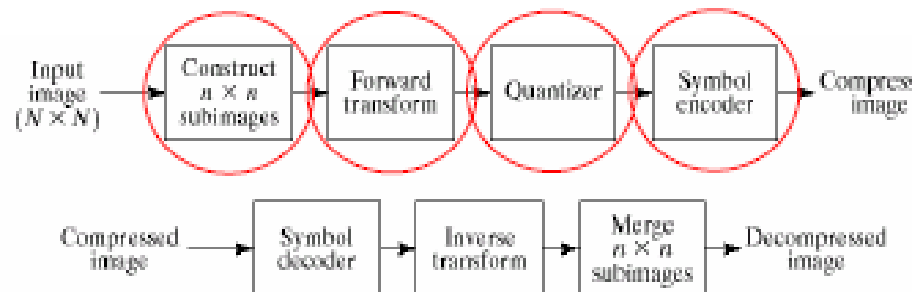
a
b
FIGURE 8.21 A lossy predictive coding model: (a) encoder and (b) decoder.

9. Lossy Transform Coding

- The predictive coding techniques operate directly on the pixels of an image and thus are spatial domain methods.
- In this section, we consider compression techniques that are based on *modifying the transform of an image*.
- In *transform coding*, a reversible, linear transform (such as Fourier transform) is used to map the image into a set of transform coefficients, which are then quantized and coded.
- For most natural images, a significant number of the coefficients have small magnitudes and can be coarsely quantized (or discarded entirely) with little image distortion.

9. Lossy Transform Coding

- The goal of the transformation process is to decorrelate the pixels of each subimage, or to pack as much information as possible into the smallest number of transform coefficients.



a
b

FIGURE 8.28 A transform coding system: (a) encoder; (b) decoder.

- The quantization stage then selectively eliminates or more coarsely quantizes the coefficients that carry the least information.
- The encoding process terminates by coding (normally using a variable length code) the quantized coefficients.
- Any or all of the transform encoding steps can be adapted to local image content, called *adaptive transform coding*, or fixed for all subimages, called *nonadaptive transform coding*.

9. Lossy Transform Coding

Transform selection

Walsh-Hadamard transform (WHT)

Discrete cosine transform (DCT)

One of the most frequently used transformation for image compression.

Wavelet Selection

- The most widely used expansion functions for wavelet-based compression are the *Daubechies wavelets* and *biorthogonal wavelets*.

9. Lossy Transform Coding



Three approximations of the 512 x 512 monochrome image in Fig.8.23.

These pictures were obtained by

1. Dividing the original image into subimages of size 8 x 8,
2. Transforms *rms* error
 - DFT 1.28
 - WHT 0.86
 - DCT 0.68
3. truncating 50% of the resulting coefficients (minimum magnitude).
4. inverse transform

a b
c d
e f

FIGURE 8.31 Approximations of Fig. 8.23 using the (a) Fourier, (c) Hadamard, and (e) cosine transforms, together with the corresponding scaled error images.

9. Lossy Transform Coding



Compression ratio

34 : 1

67 : 1

(the average compression ratio obtained by using all the error-free methods was only 2.62 : 1)

rms error

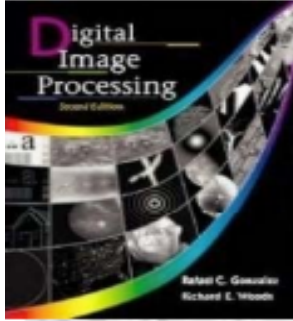
3.42

6.33 gray levels

a b
c d
e f

FIGURE 8.38 Left column: Approximations of Fig. 8.23 using the DCT and normalization array of Fig. 8.37(b). Right column: Similar results for 4Z.

END of Chapter 8 : Part2



Digital Image Processing
Digital Image Processing Using Matlab

Chapter 9

Image Segmentation: Part1

Prepared by:
Dr. Ali J. Abboud

Key Features of Chapter 9

- Detection of gray level discontinuities
 - Point detection
 - Line detection
 - Edge detection
 - Gradient operators
 - LoG : Laplacian of Gaussian
- Edge linking and boundary detection
 - Hough transform
- Thresholding
- Region-based segmentation
- Segmentation by Morphological watersheds
- The use of motion in segmentation

2. Preview

- Segmentation is to subdivide an image into its component regions or objects.
- Segmentation should stop when the objects of interest in an application have been isolated.



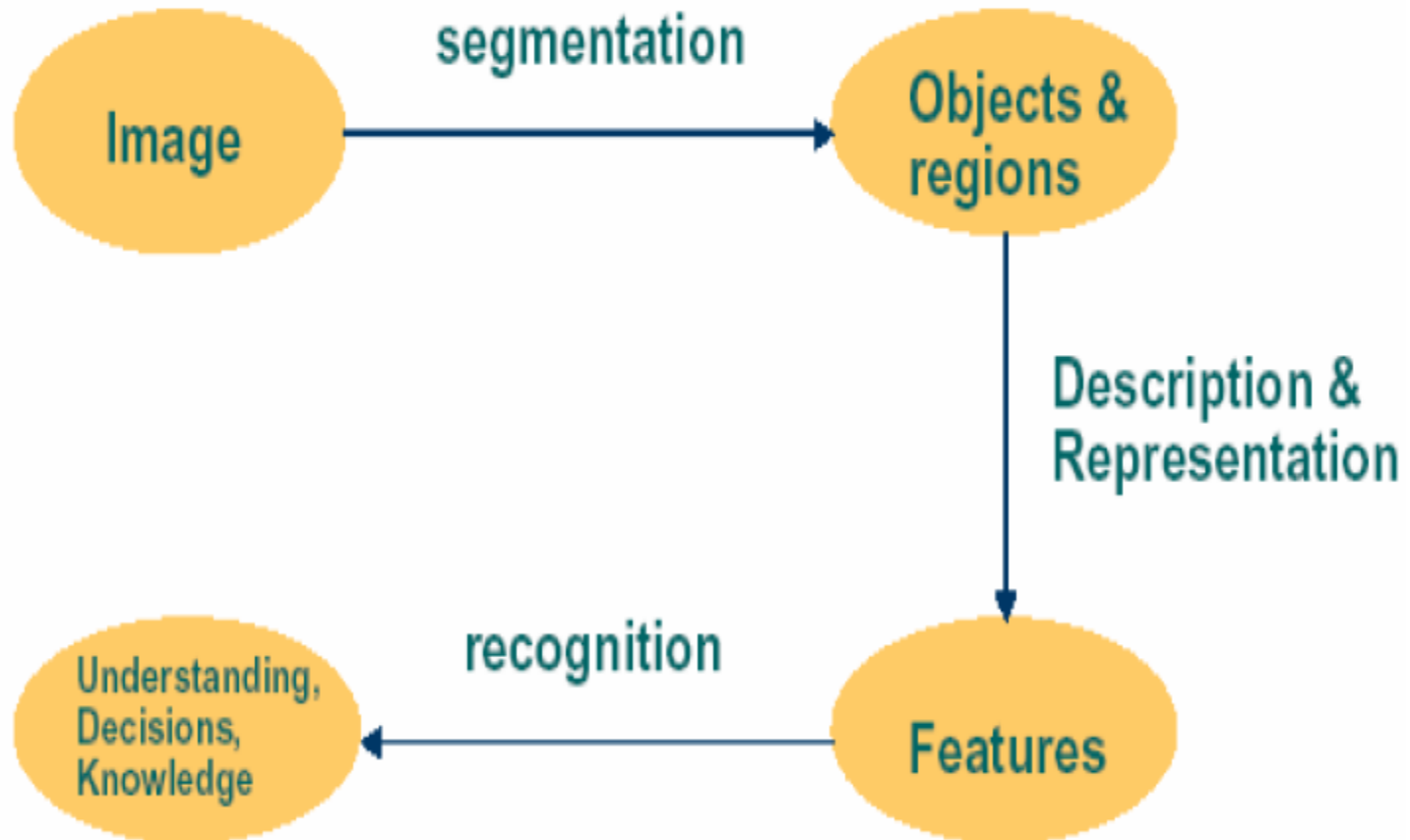
3. Principal approaches

Goal of Image Analysis : Extracting information from an image

- **Step 1 : segment the image ---> objects or regions**
- **Step 2 : describe and represent the segmented regions in a form suitable for computer processing**
- **Step 3 : image recognition and interpretation**

- **Segmentation algorithms generally are based on one of 2 basis properties of intensity values**
 - **discontinuity : to partition an image based on sharp changes in intensity (such as edges)**
 - **similarity : to partition an image into regions that are similar according to a set of predefined criteria.**

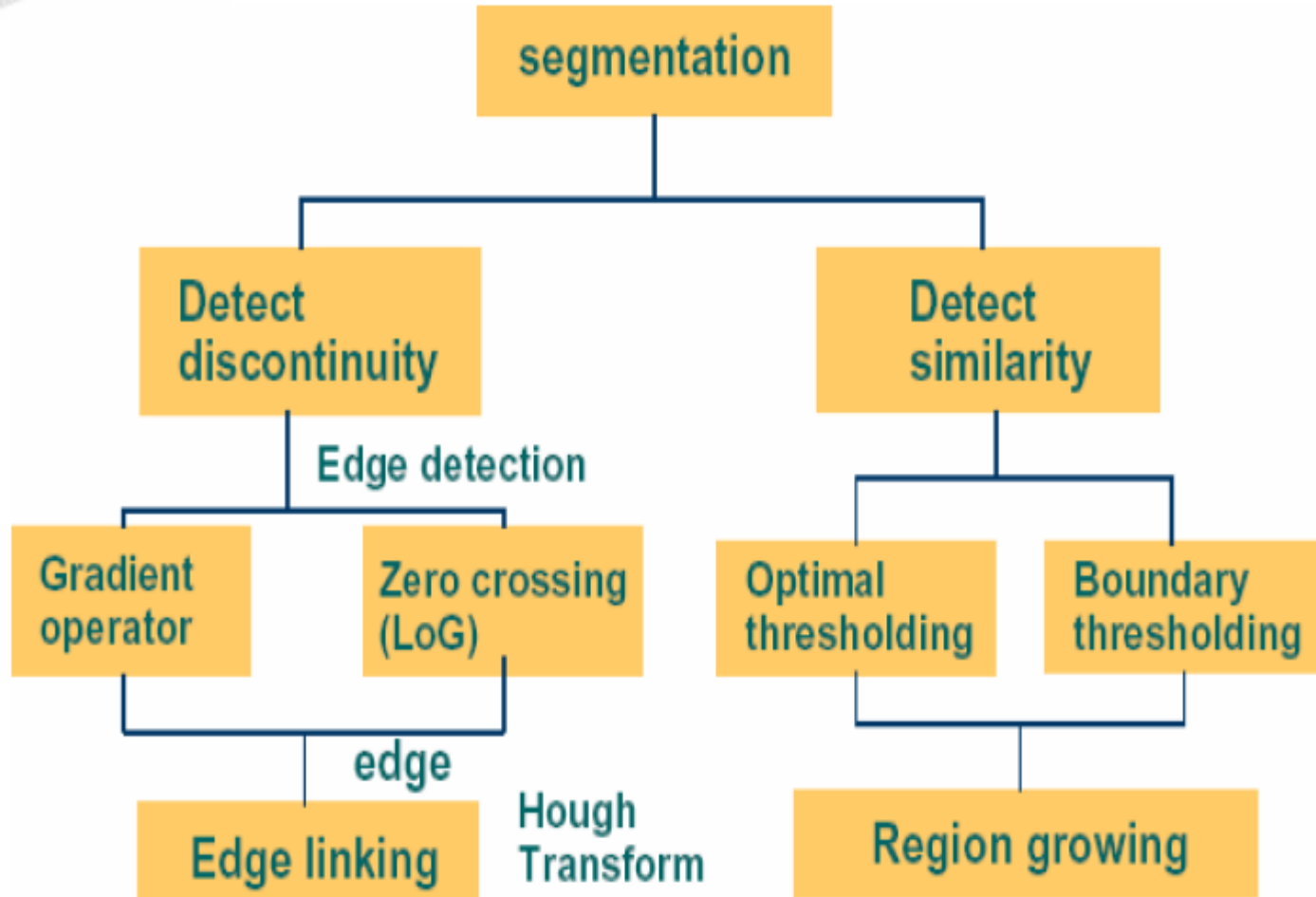
4. Principle Approaches



5. What is segmentation?

- Subdivides an image into its constituent regions or objects
- Separate an image into regions which are called as objects and background
- Represent the result images with binary images, label the objects as “1” and the background as “0” commonly.
- Heavily rely on one of two properties of intensity values:
 - Discontinuity ---- Partition based on abrupt changes in intensity, e.g. edges in an image
 - point / line / edge / corner detection
 - Similarity ---- Partition based on intensity similarity, e.g. thresholding
 - thresholding
 - region growing / splitting / merging

6. Segmentation



7. What should Good Segmentation Algorithm be?

- Region interiors
 - Simple
 - Without many small holes
- Adjacent regions
 - Should have significantly different values
- Boundaries
 - Simple
 - Not ragged
 - Spatially accurate

8. Detection of Discontinuity

- detect the three basic types of gray-level discontinuities
 - points , lines , edges
- the common way is to run a mask through the image

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

8. Detection of Discontinuity

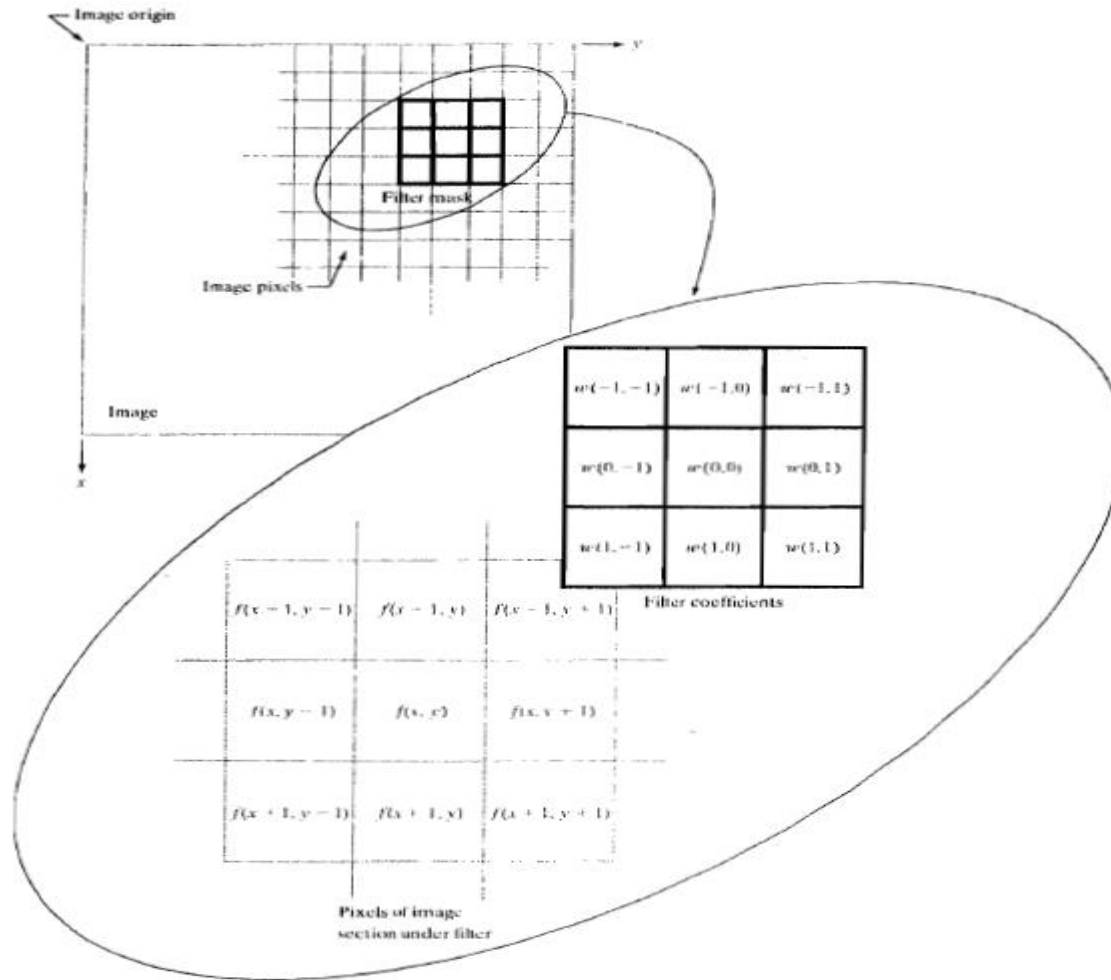
Correlation

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

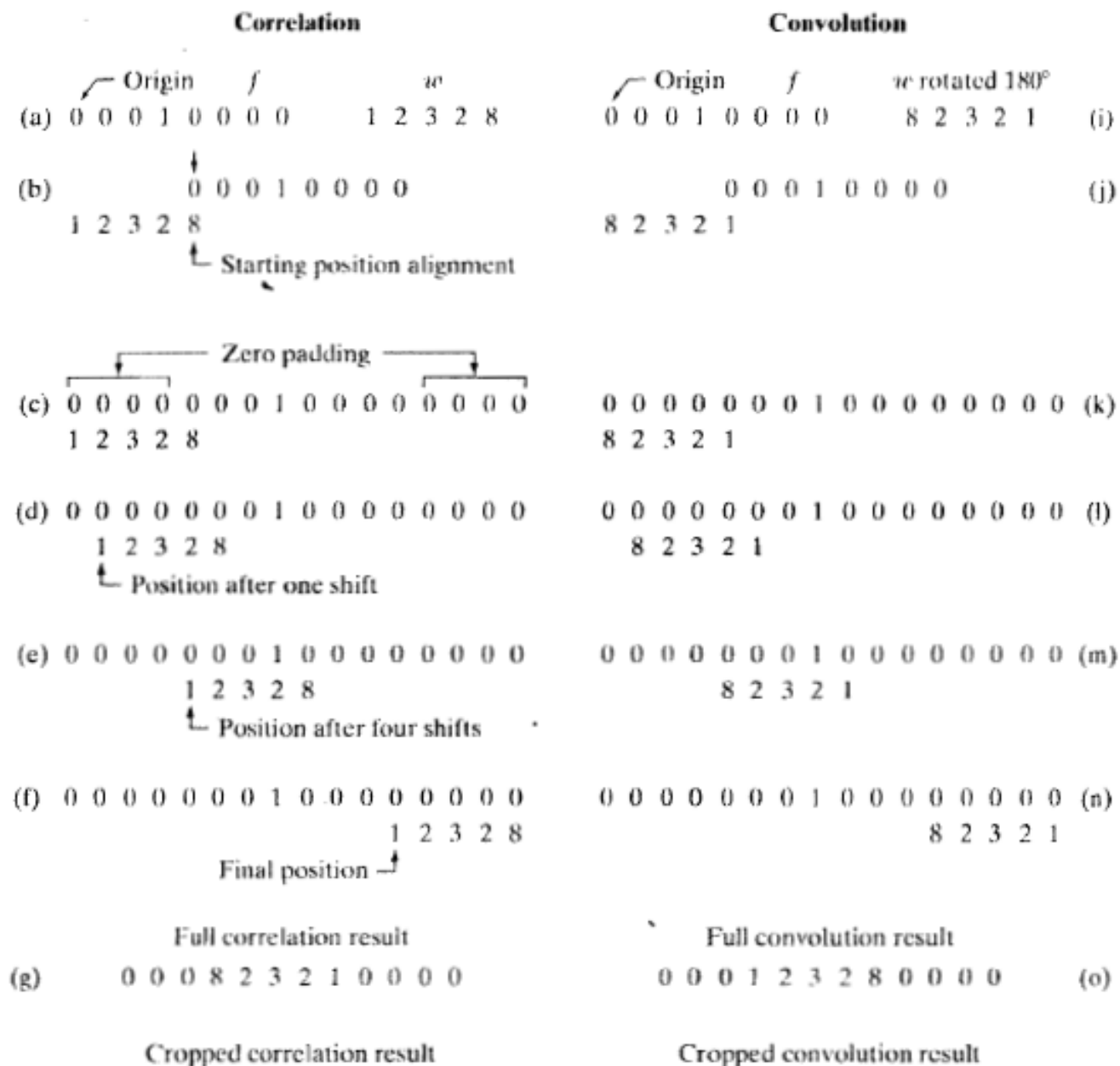
Grayscale image

Mask coefficient

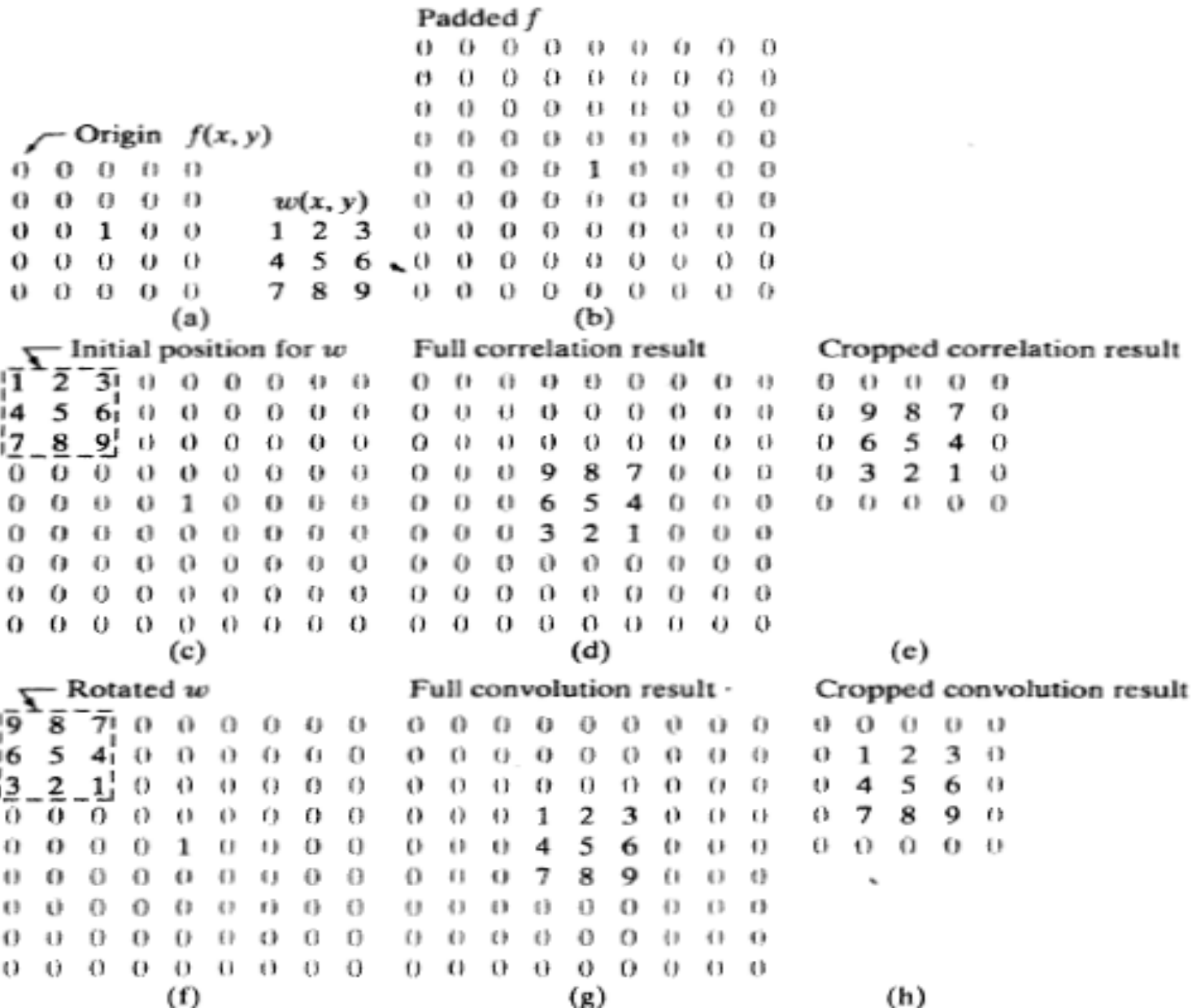
9. Convolution Vs. Correlation



9. Convolution Vs. Correlation



9. Convolution Vs. Correlation



9. Convolution Vs. Correlation

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

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

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

$$= \sum_{k=1}^{mn} w_k z_k$$

$$= \mathbf{w}^T \mathbf{z}$$

11. First and Second Derivatives

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

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

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

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

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

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

12. Isolated Point Detection

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

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

$$|R| \geq T$$

- where
 - T is a nonnegative threshold
 - R is the sum of products of the coefficients with the gray levels contained in the region encompassed by the mark.

12. Isolated Point Detection

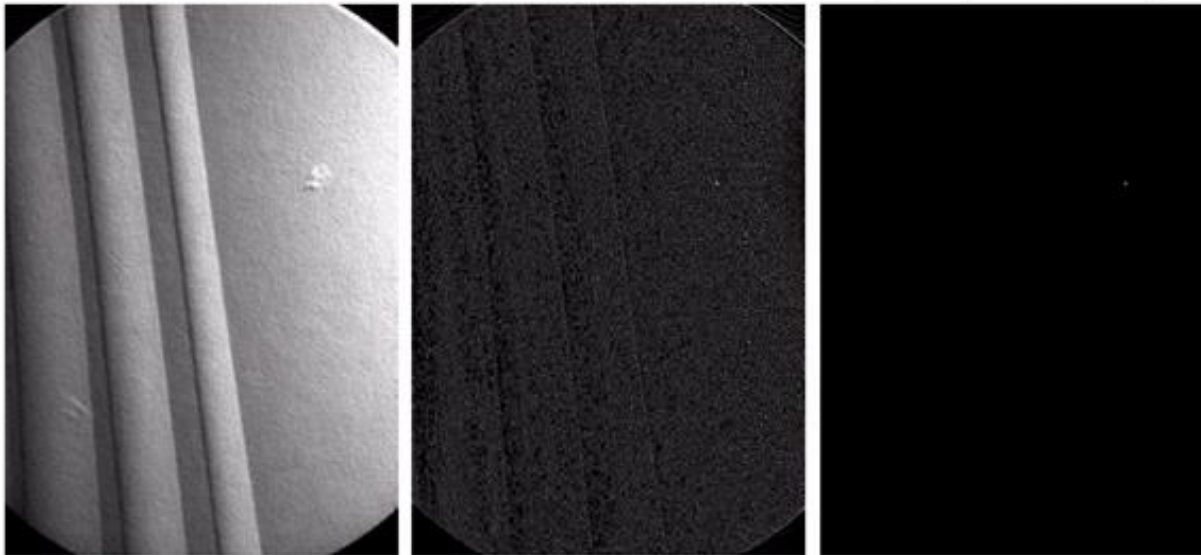
- Note that the mask is the same as the mask of Laplacian Operation (in chapter 3)
- The only differences that are considered of interest are those large enough (as determined by T) to be considered isolated points.

$$|R| \geq T$$

12. Isolated Point Detection

Example

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



a
b c d

FIGURE 10.2
(a) Point detection mask.
(b) X-ray image of a turbine blade with a porosity.
(c) Result of point detection.
(d) Result of using Eq. (10.1-2).
(Original image courtesy of X-TEK Systems Ltd.)

13. Line Detection



Line Detection

-1	-1	-1	-1	-1	2	-1	2	-1	2	-1	-1
2	2	2	-1	2	-1	-1	2	-1	-1	2	-1
-1	-1	-1	2	-1	-1	-1	2	-1	-1	-1	2
Horizontal			+45°			Vertical			-45°		

- Horizontal mask will result with max response when a line passed through the middle row of the mask with a constant background.
- the similar idea is used with other masks.
- note: the preferred direction of each mask is weighted with a larger coefficient (i.e., 2) than other possible directions.

13. Line Detection



Line Detection

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

13. Line Detection

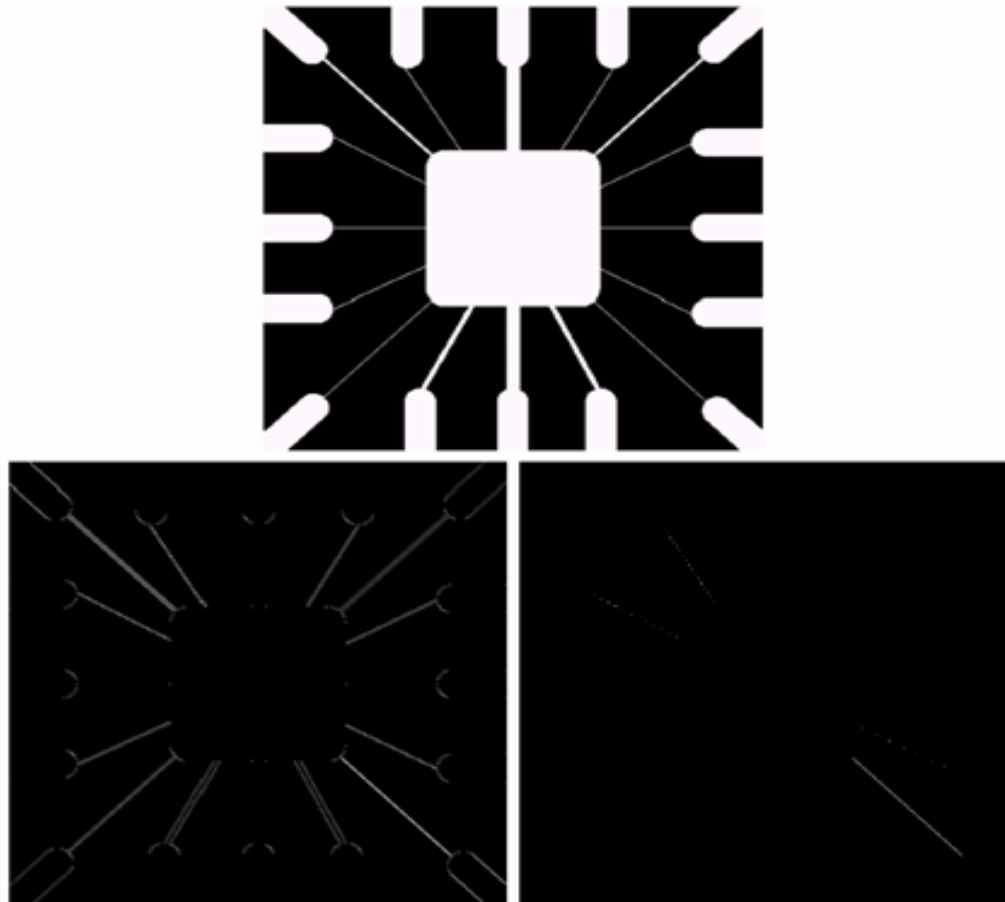


Line Detection

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

13. Line Detection

Example



a
b c

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

14. Edge Detection

- we discussed approaches for implementing
 - first-order derivative (Gradient operator)
 - second-order derivative (Laplacian operator)
- Here, we will talk only about their properties for edge detection.
- we have introduced both derivatives in chapter 3

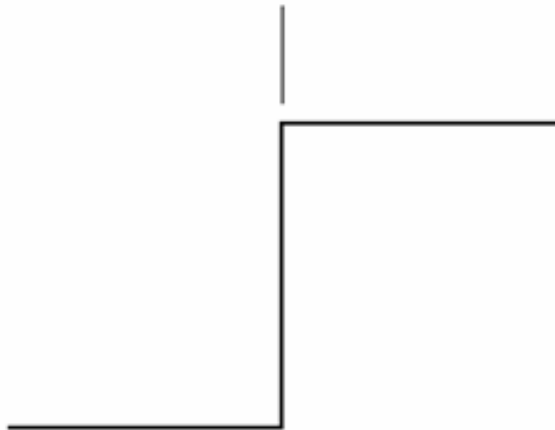
14. Edge Detection

Ideal and Ramp Edges

Model of an ideal digital edge



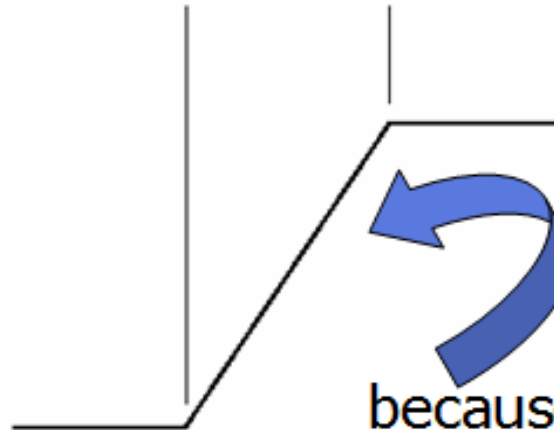
Gray-level profile
of a horizontal line
through the image



Model of a ramp digital edge



Gray-level profile
of a horizontal line
through the image



a b

FIGURE 10.5

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

because of optics,
sampling, image
acquisition
imperfection

14. Edge Detection



Thick edge

- The slope of the ramp is inversely proportional to the degree of blurring in the edge.
- We no longer have a thin (one pixel thick) path.
- Instead, an edge point now is any point contained in the ramp, and an edge would then be a set of such points that are connected.
- The thickness is determined by the length of the ramp.
- The length is determined by the slope, which is in turn determined by the degree of blurring.
- Blurred edges tend to be thick and sharp edges tend to be thin

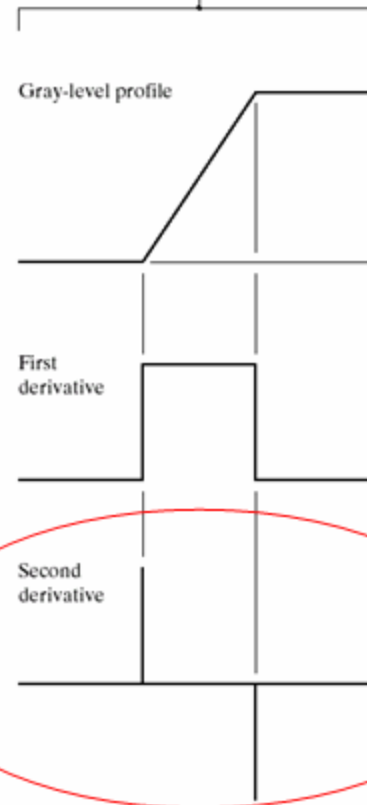
14. Edge Detection

First and Second derivatives

a b

FIGURE 10.6

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



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


14. Edge Detection



Second derivatives

- produces 2 values for every edge in an image (an undesirable feature)
- an imaginary straight line joining the extreme positive and negative values of the second derivative would cross zero near the midpoint of the edge. (**zero-crossing property**)

14. Edge Detection



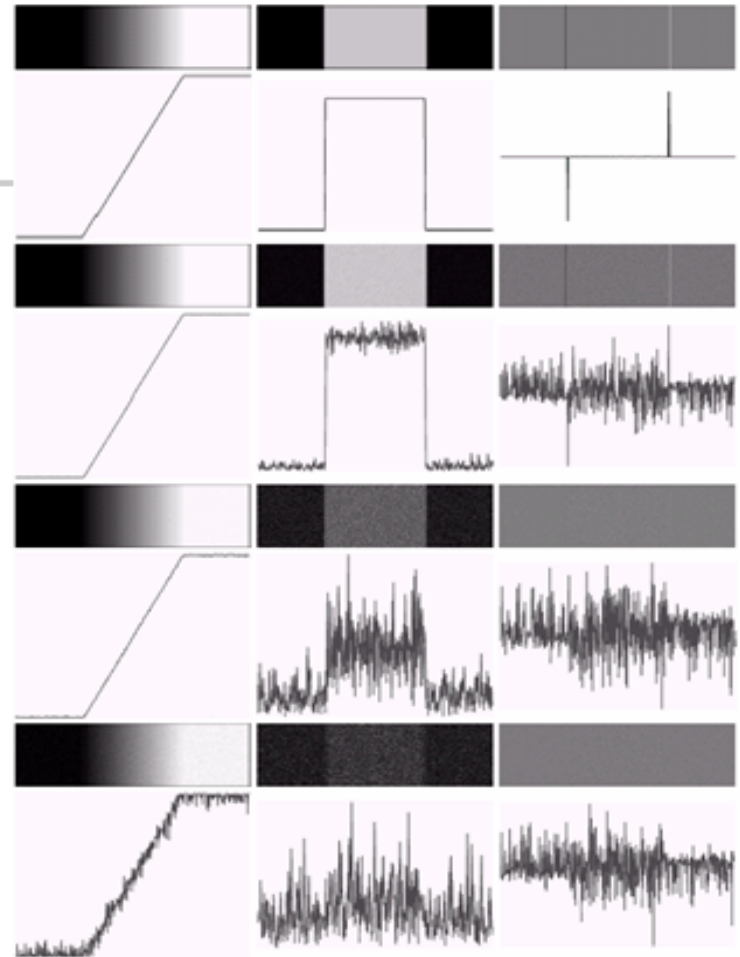
Zero-crossing

- quite useful for locating the centers of thick edges
- we will talk about it again later

14. Edge Detection

Noise Images

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



14. Edge Detection



Keep in mind

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

14. Edge Detection



Edge point

- to determine a point as an edge point
 - the transition in grey level associated with the point has to be **significantly stronger** than the background at that point.
 - use **threshold** to determine whether a value is “significant” or not.
 - the point’s two-dimensional first-order derivative must be greater than a specified threshold.

14. Edge Detection

Gradient Operator

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

- first derivatives are implemented using the **magnitude of the gradient**.

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

commonly approx.

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

the magnitude becomes nonlinear

14. Edge Detection

Gradient Masks

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

Prewitt

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

Sobel

14. Edge Detection

Diagonal edges with Prewitt and Sobel masks

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

Prewitt

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

Sobel

14. Edge Detection

Example

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, $|G_x| + |G_y|$.



14. Edge Detection

Example



a	b
c	d

FIGURE 10.11
Same sequence as in Fig. 10.10, but with the original image smoothed with a 5×5 averaging filter.

14. Edge Detection

Example



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).

END of Chapter 9 : Part1