Chapter 2

DR. BHAKTI PALKAR

Syllabus

Fundamentals of Digital Image and Spatial Domain Enhancement

2.1 Digital image Representation, Elements of digital image processing systems, sampling and quantization, basic relationships between pixels, mathematical operations on images.

2.2 Spatial domain enhancement techniques: Point processing, Neighborhood processing, spatial domain filtering, zooming.

2.3 Spatial enhancement: Global processing: Histogram Equalization.

Self-Learning Topic: Histogram specification

What is Digital Image Processing?

Digital image processing focuses on two major tasks

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

Key Stages in Digital Image Processing

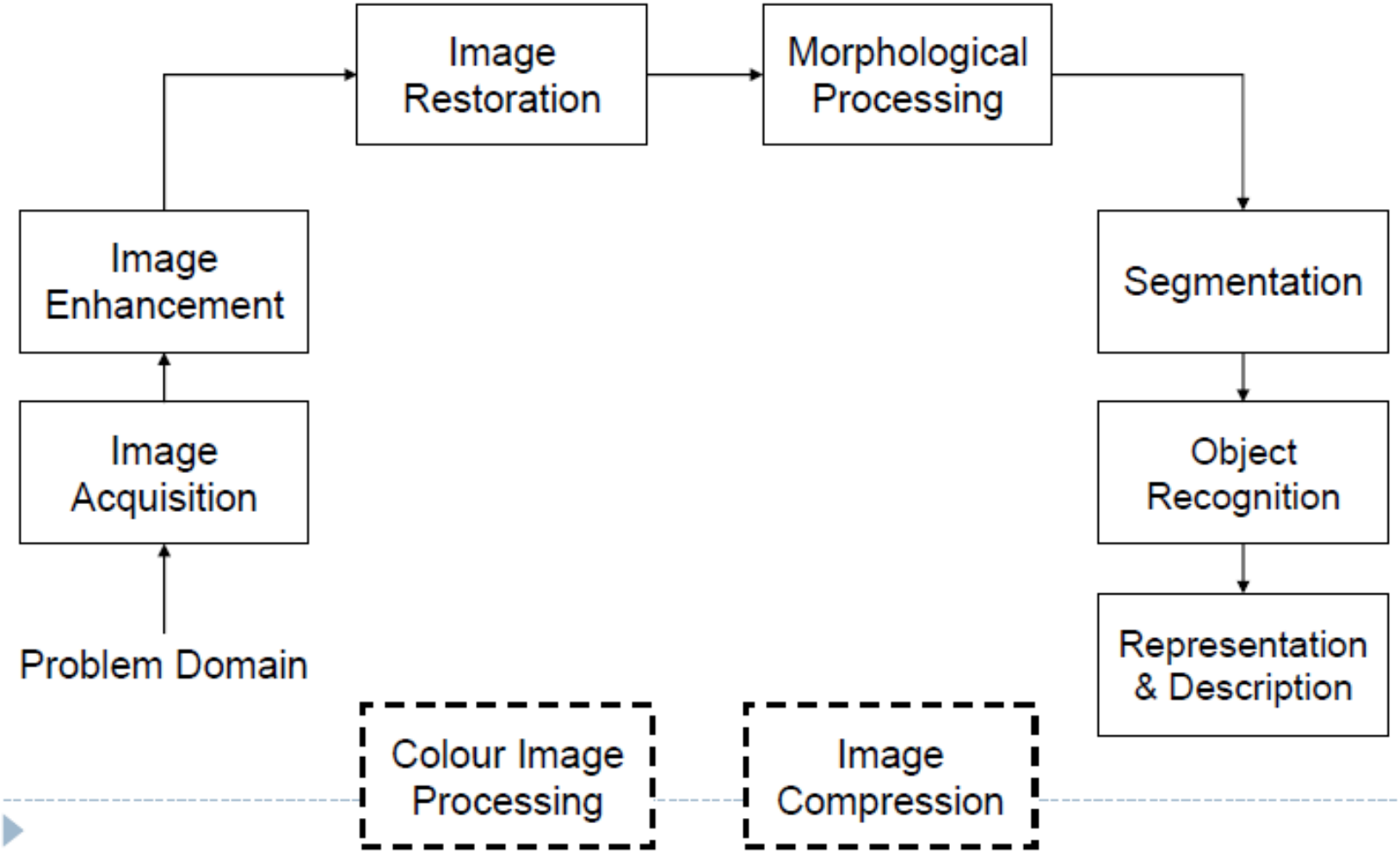
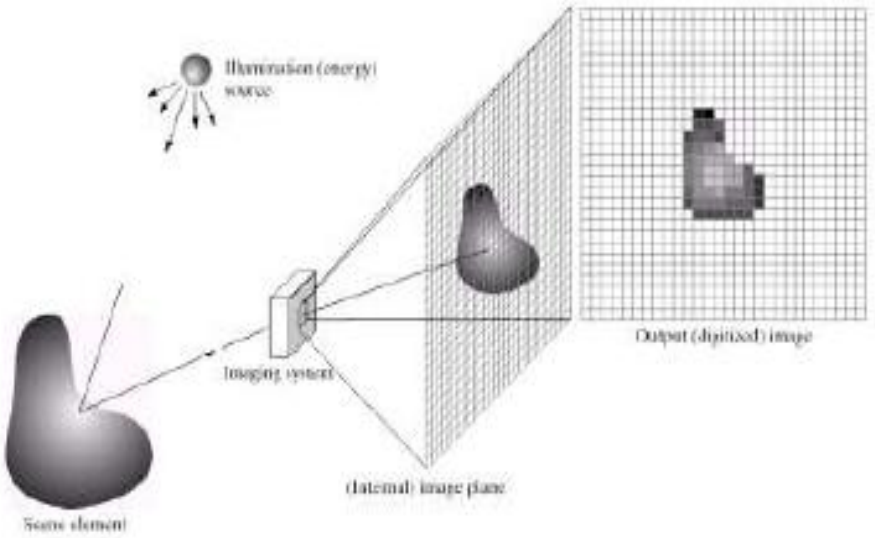


Image Restoration

Morphological Processing

Image Enhancement



Segmentation

Image Acquisition

Object Recognition

Problem Domain

Representation & Description

Colour Image Processing

Image Compression

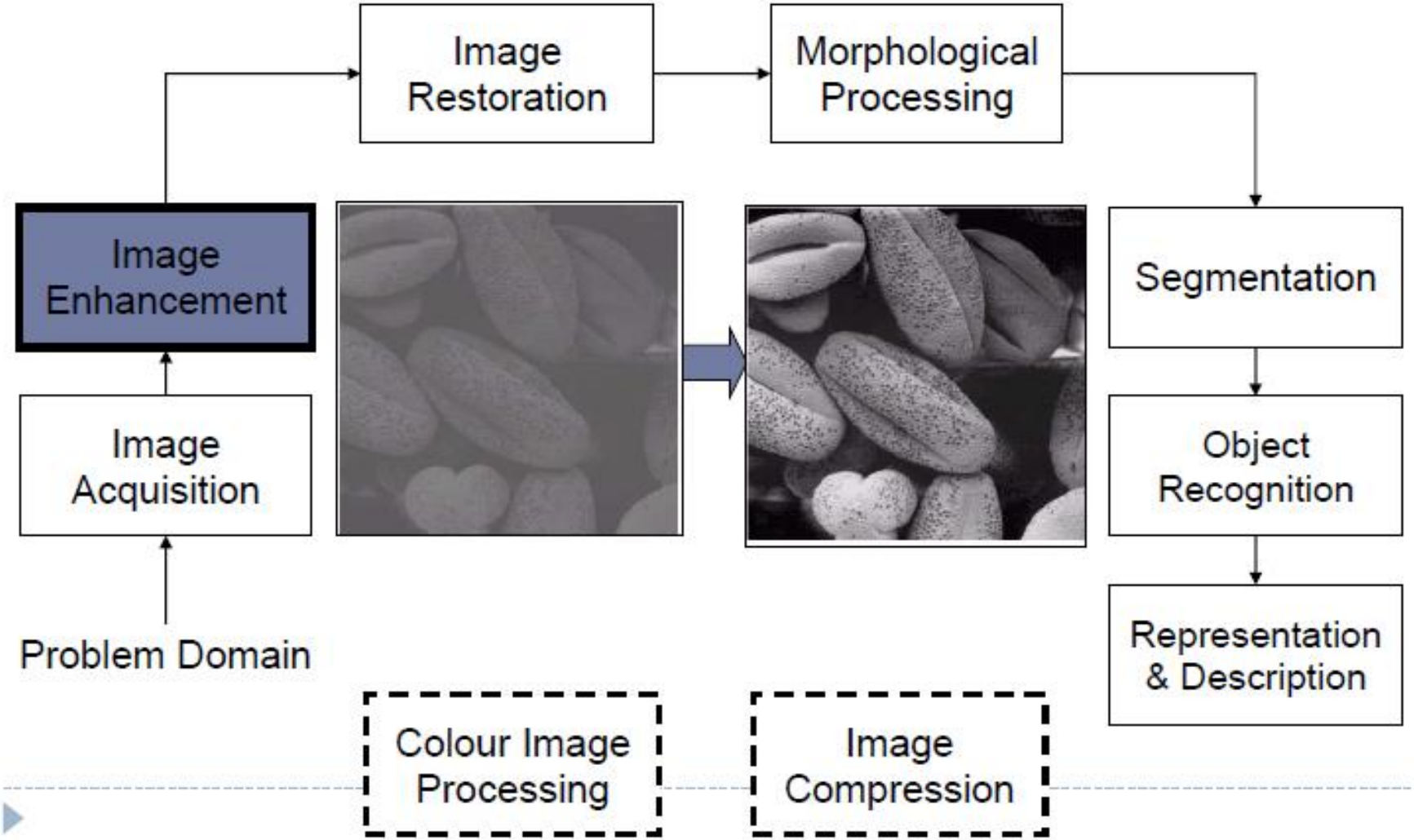
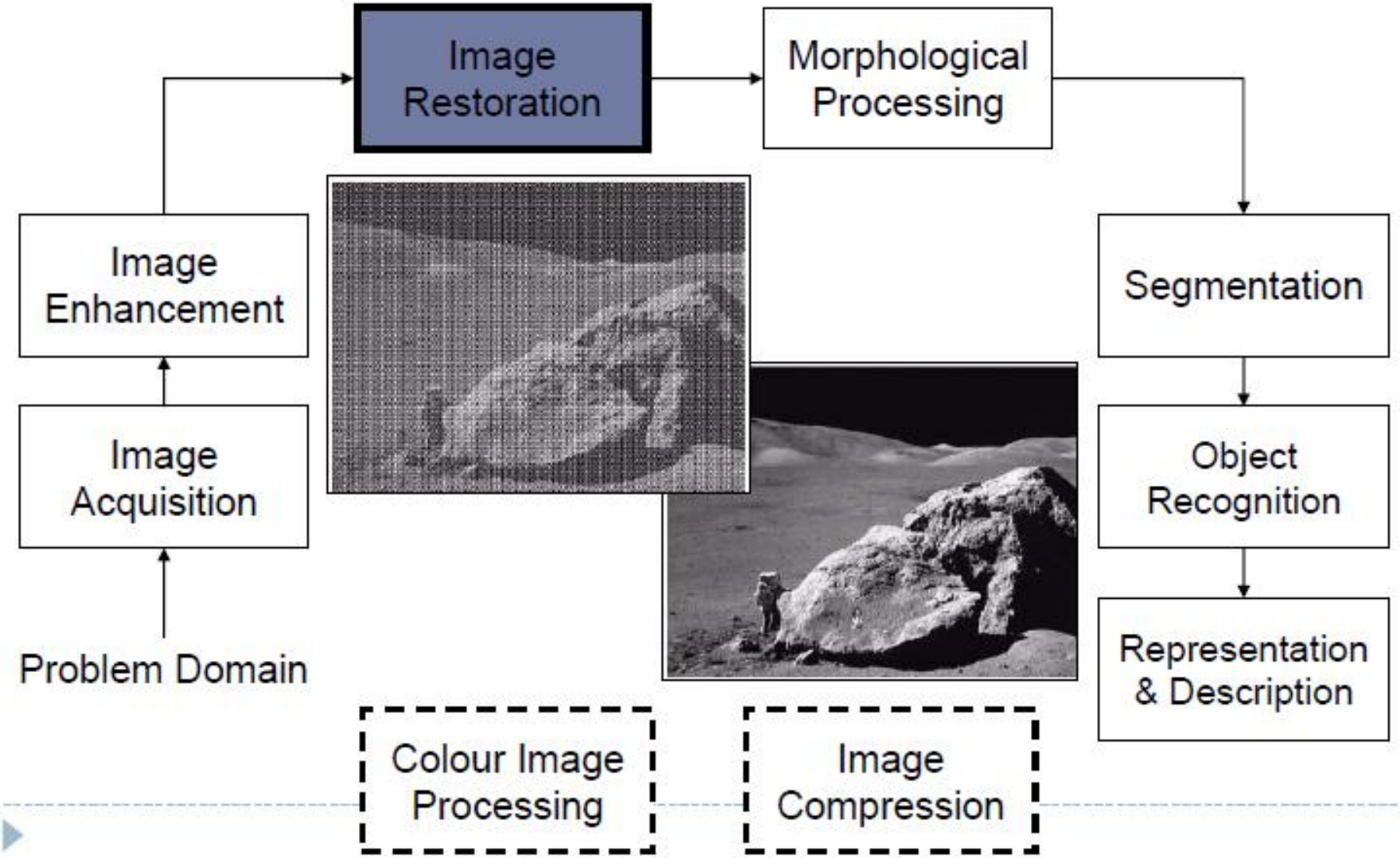
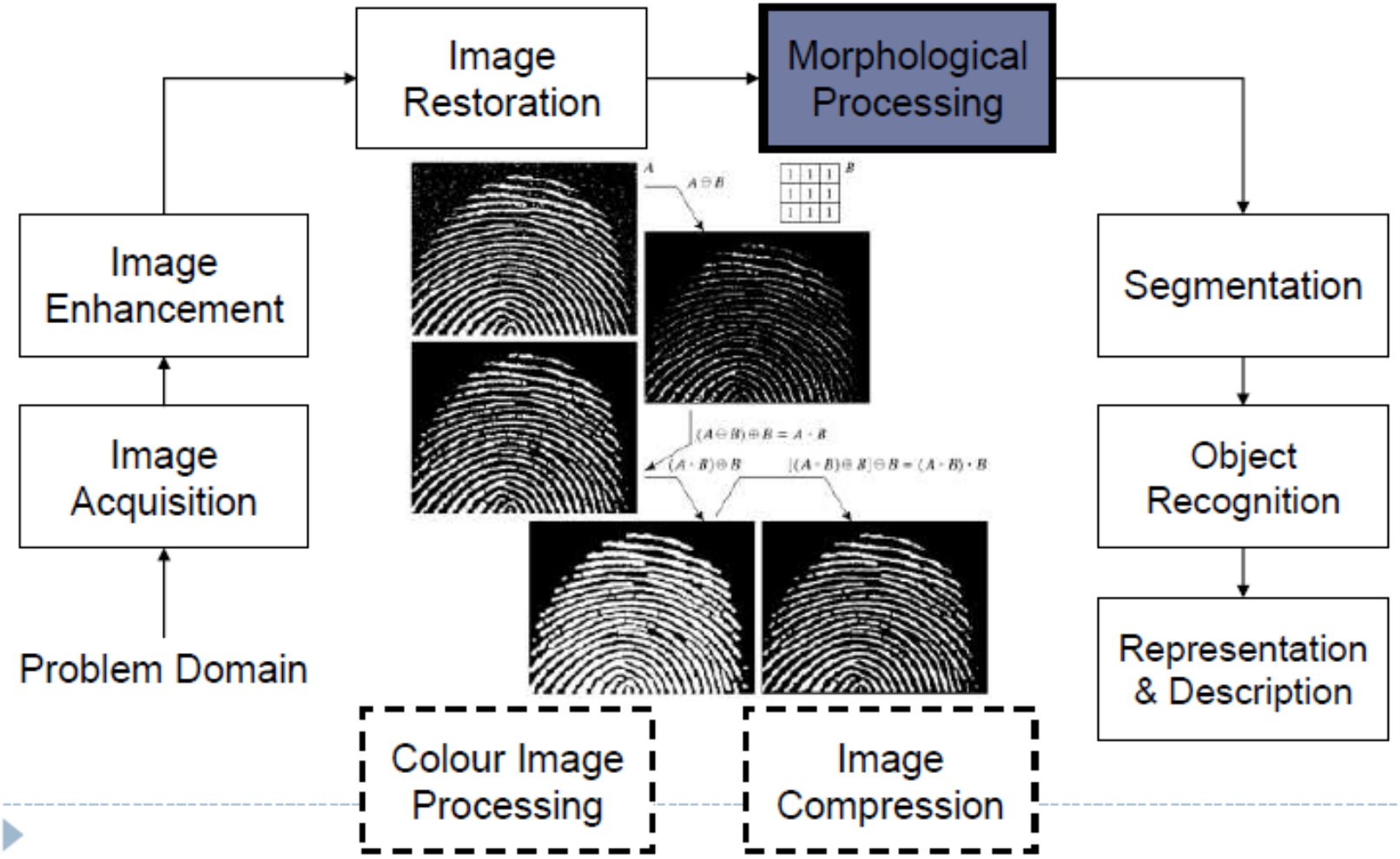


Image Restoration
Morphological Processing
Image Enhancement
Segmentation
Image Acquisition
Object Recognition
Problem Domain
Representation & Description
Colour Image Processing
Image Compression

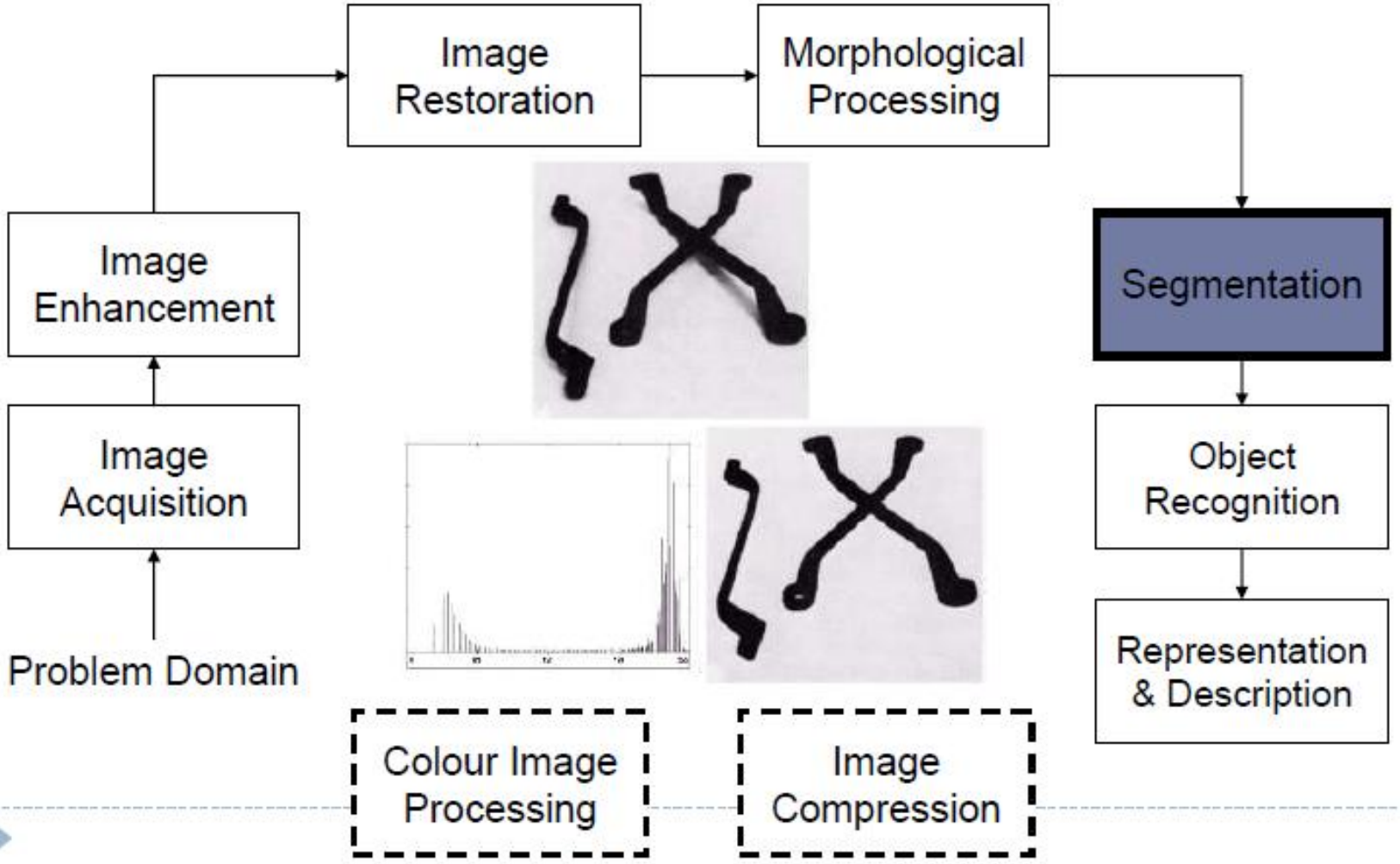
Image Restoration



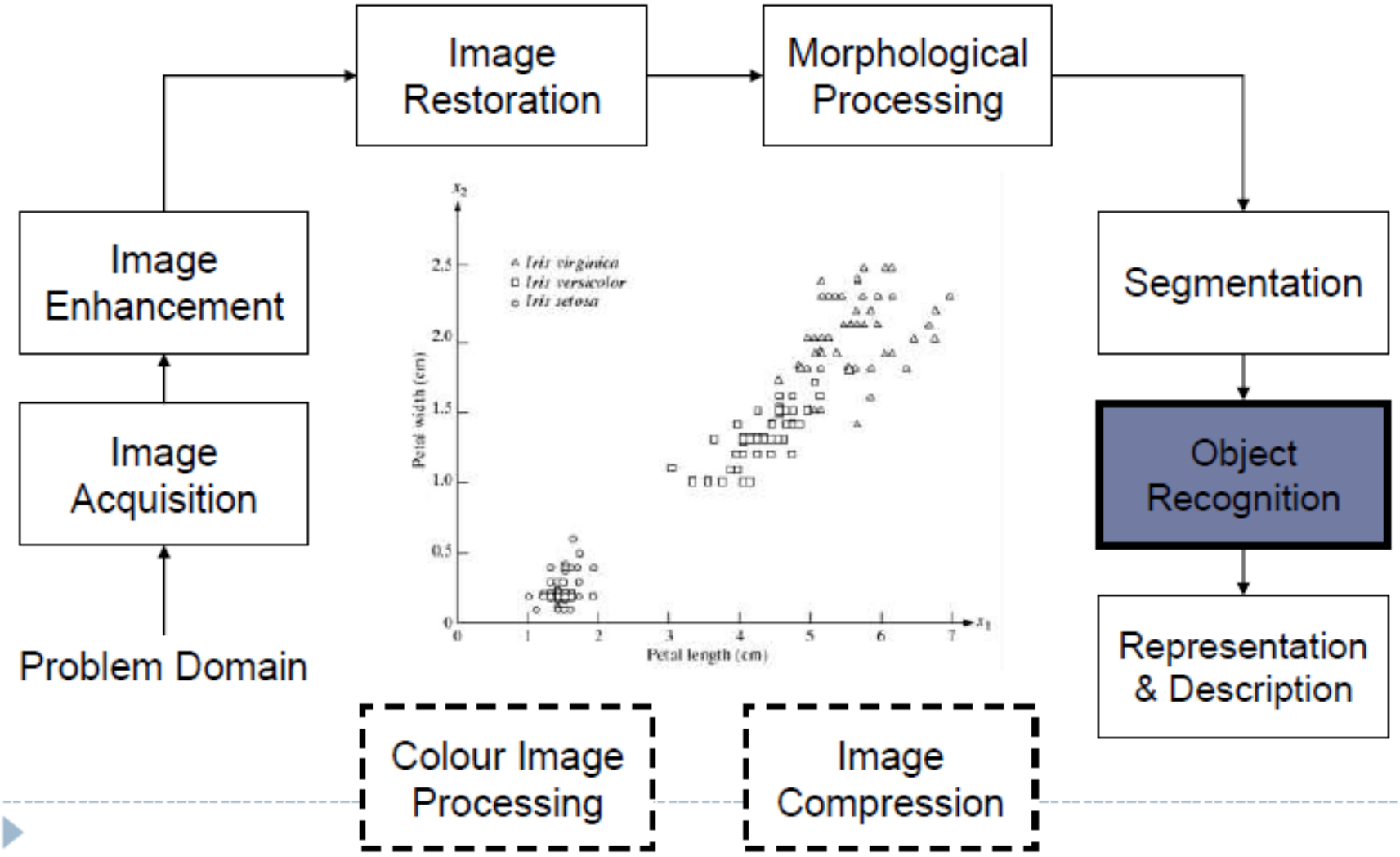
Key Stages in Digital Image Processing: Morphological Processing



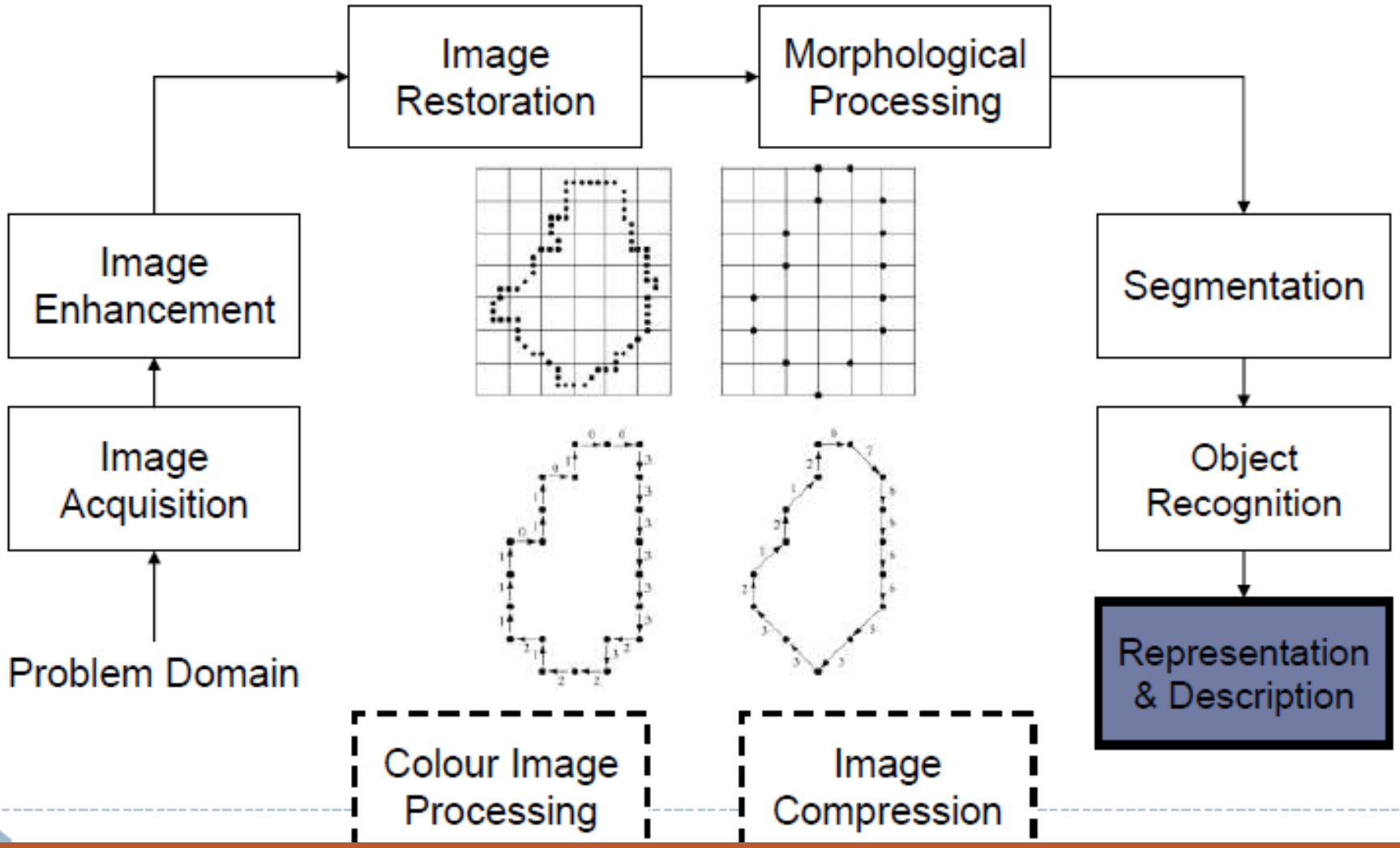
Key Stages in Digital Image Processing: Segmentation



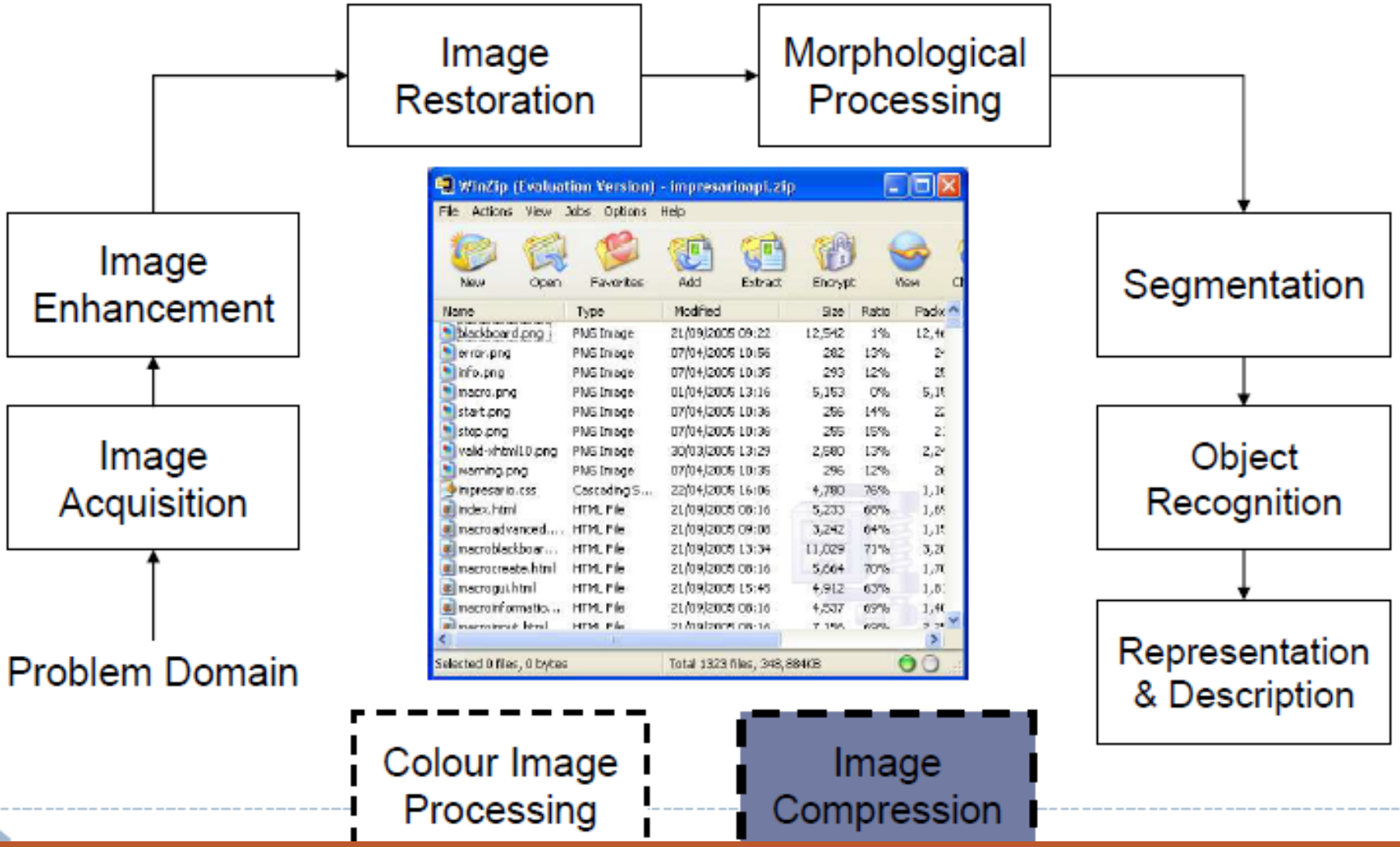
Key Stages in Digital Image Processing: Object Recognition



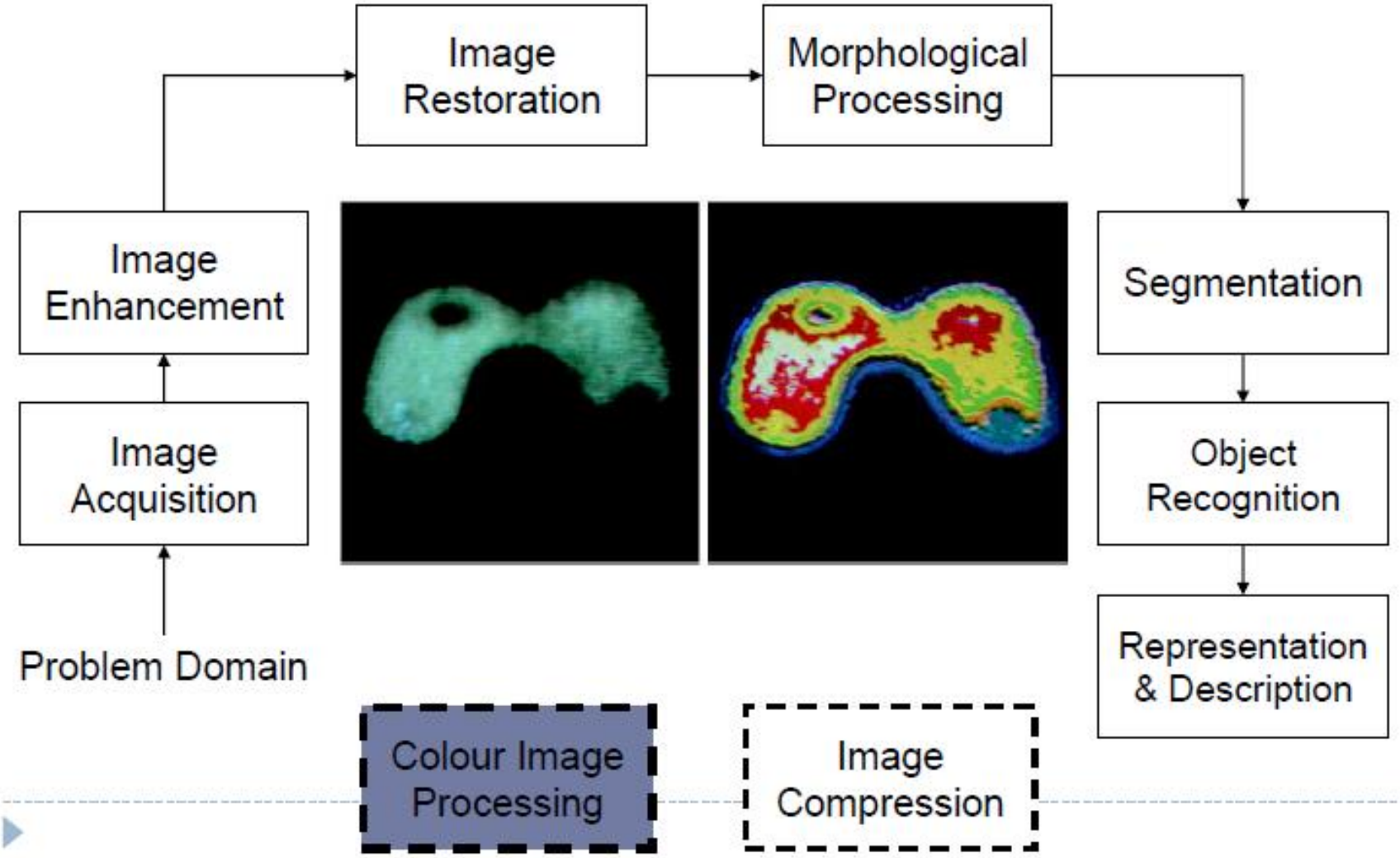
Key Stages in Digital Image Processing: Representation & Description



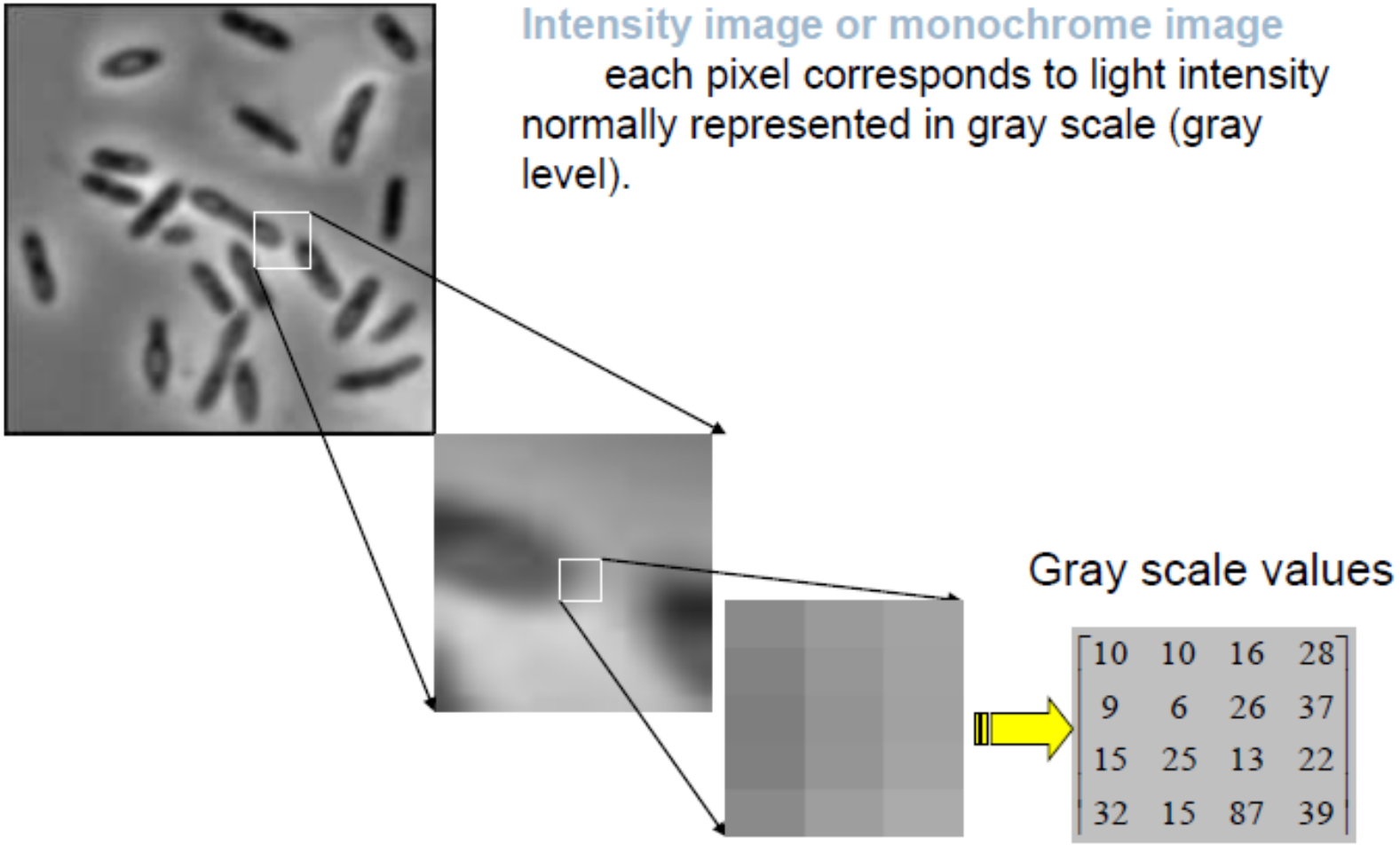
Key Stages in Digital Image Processing: Image Compression



Key Stages in Digital Image Processing: Colour Image Processing



Digital Image Types : Intensity Image



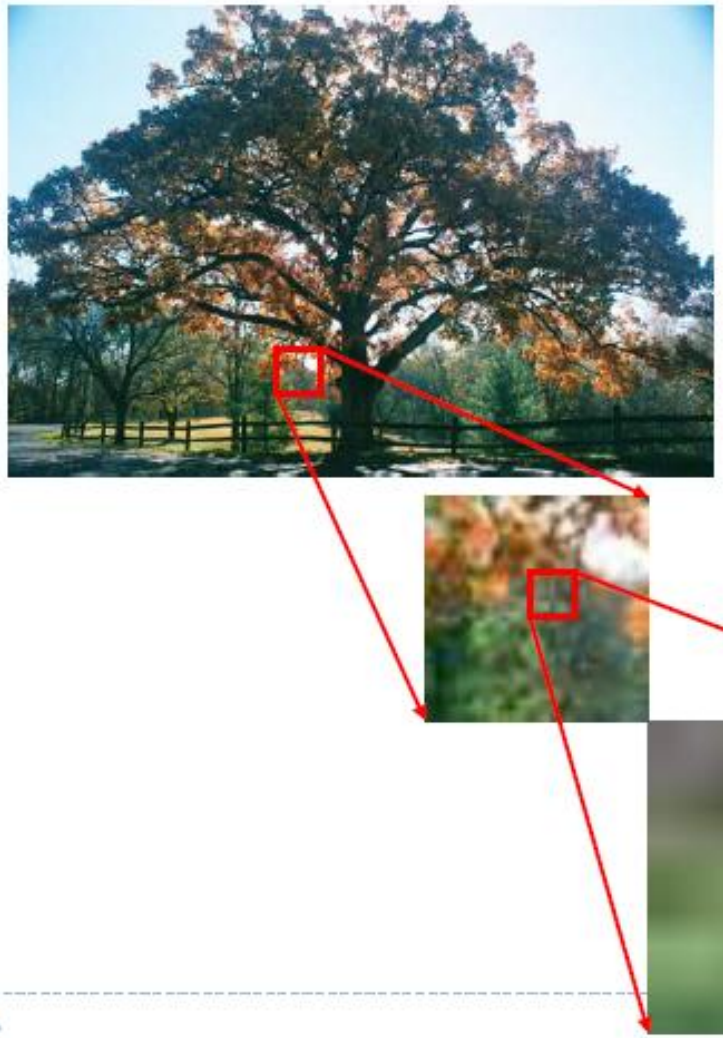
Intensity image or monochrome image

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

Gray scale values

$$\begin{bmatrix} 10 & 10 & 16 & 28 \\ 9 & 6 & 26 & 37 \\ 15 & 25 & 13 & 22 \\ 32 & 15 & 87 & 39 \end{bmatrix}$$

Digital Image Types : RGB Image

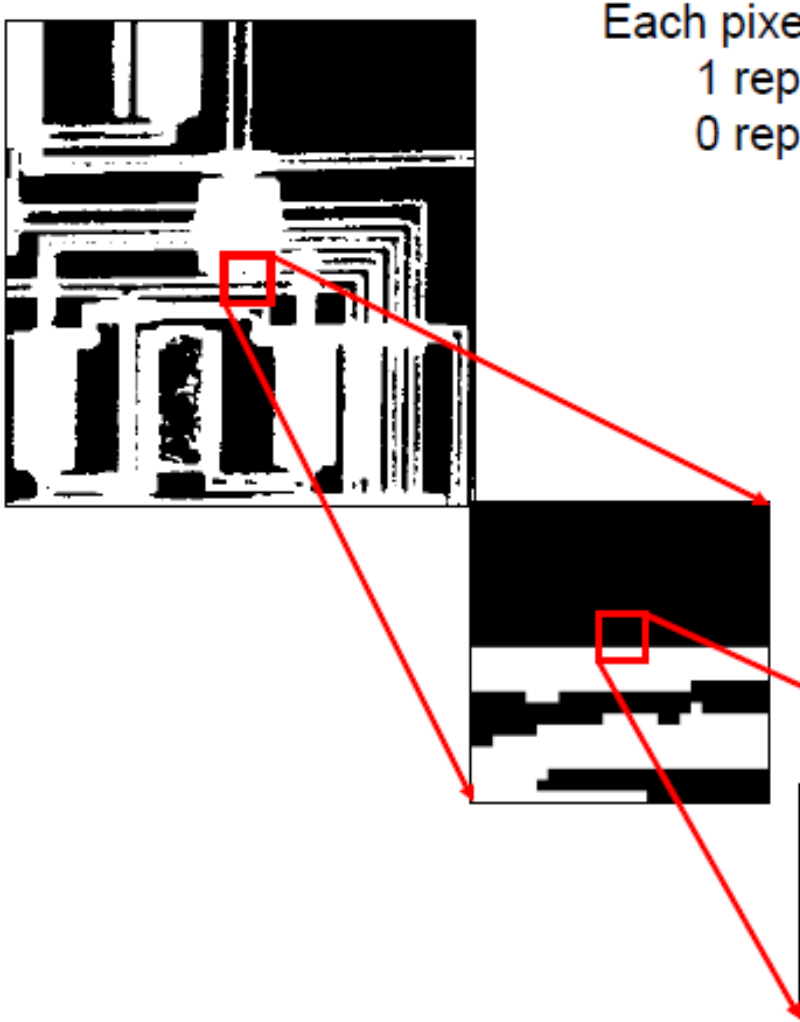


Color image or RGB image: each pixel contains a vector representing red, green and blue components.

RGB components

$$\begin{bmatrix} 10 & 10 & 16 & 28 \\ 9 & & & \\ 15 & & & \\ 32 & & & \end{bmatrix} \begin{bmatrix} 65 & 70 & 56 & 43 \\ 32 & & & \\ 21 & & & \\ 54 & & & \end{bmatrix} \begin{bmatrix} 99 & 70 & 56 & 78 \\ 60 & 90 & 96 & 67 \\ 85 & 85 & 43 & 92 \\ 32 & 65 & 87 & 99 \end{bmatrix}$$

Image Types : Binary Image



Binary image or black and white image

Each pixel contains one bit :

1 represent white

0 represents black

Binary data

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Effect of Quantization Levels or Intensity resolution

256 levels

128 levels

64 levels

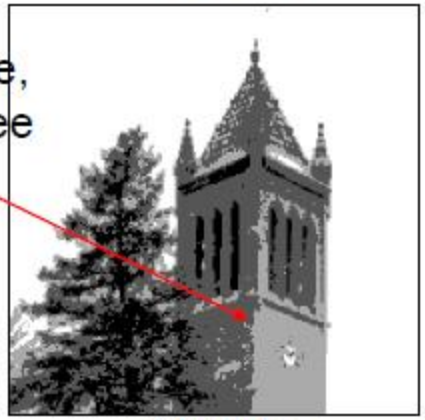
32 levels

Effect of Quantization Levels (continued) Intensity resolution

16 levels

8 levels

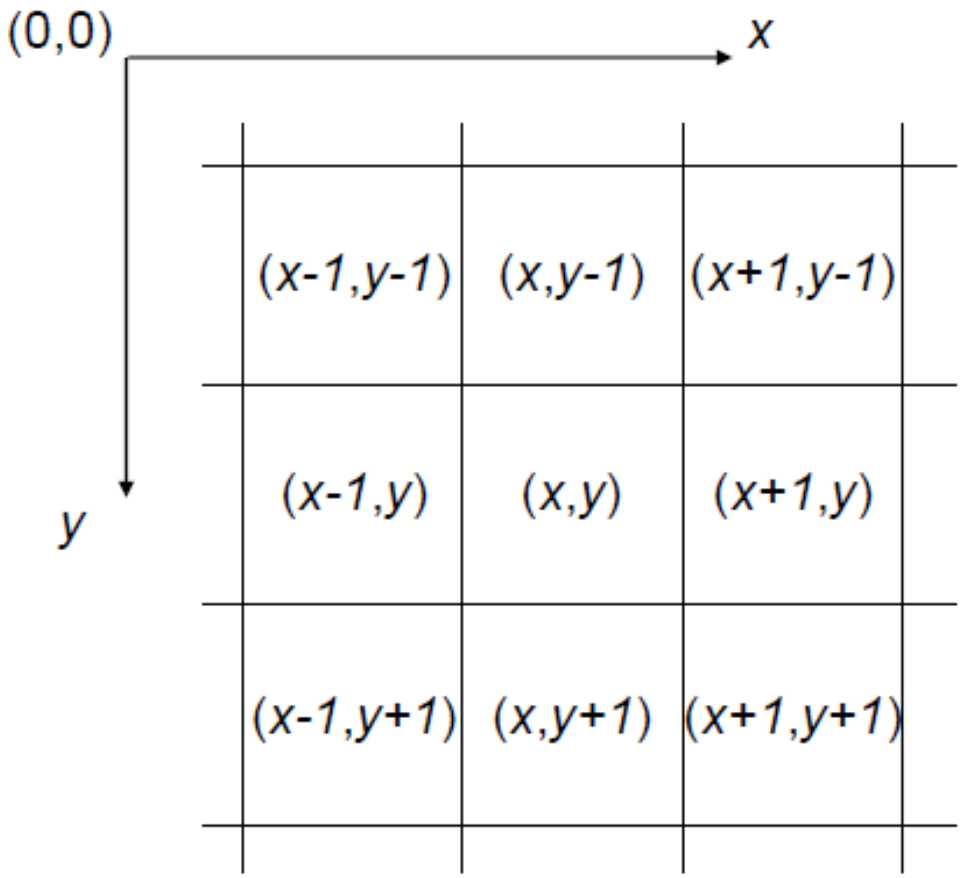
In this image, it is easy to see false contour.



4 levels

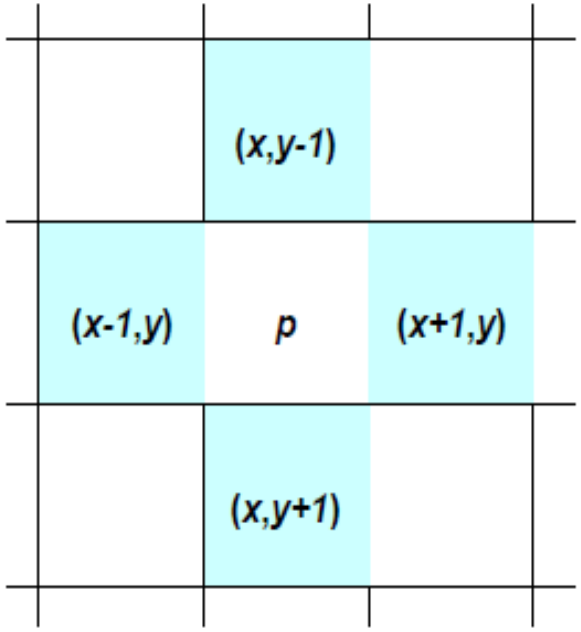
2 levels

Basic Relationship of Pixels



Conventional indexing method

Neighborhood relation is used to tell adjacent pixels. It is useful for analyzing regions.



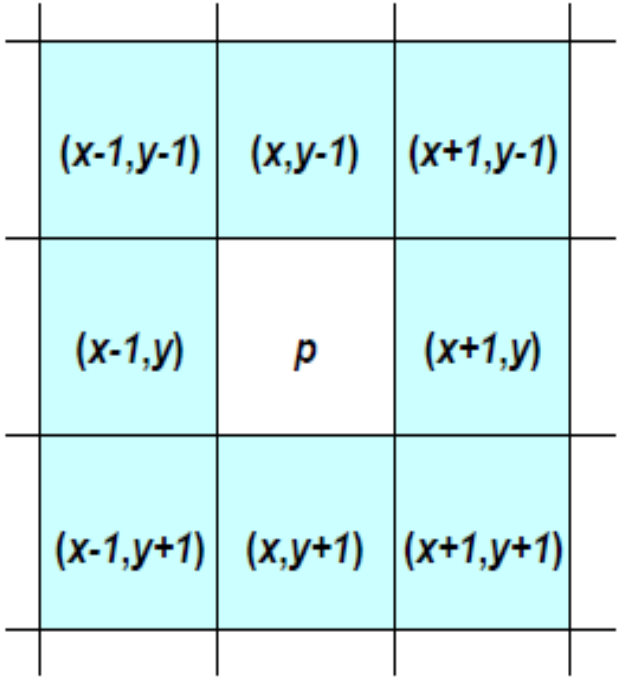
4-neighbors of *p*:

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

4-neighborhood relation considers only vertical and horizontal neighbors.

Note: $q \in N_4(p)$ implies $p \in N_4(q)$

Neighbors of a Pixel (cont.)

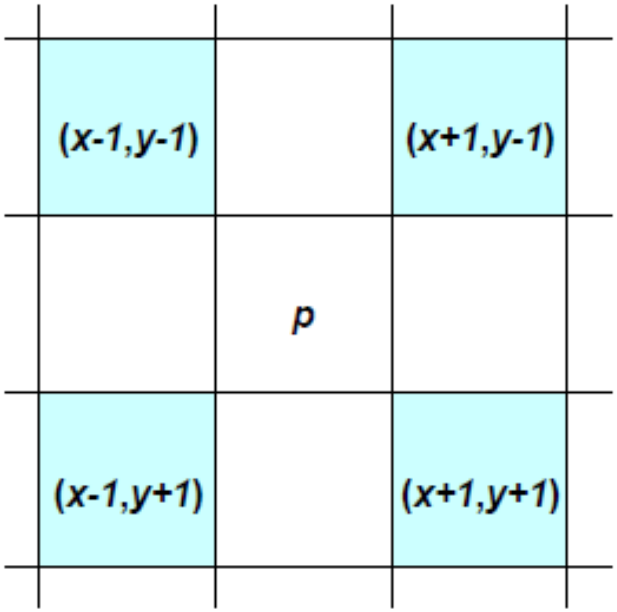


8-neighbors of *p*:

$$N_8(p) = \{ (x-1,y-1), (x,y-1), (x+1,y-1), (x-1,y), (x+1,y), (x-1,y+1), (x,y+1), (x+1,y+1) \}$$

8-neighborhood relation considers all neighbor pixels.

Neighbors of a Pixel (cont.)



Diagonal neighbors of *p*:

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

Diagonal -neighborhood relation considers only diagonal neighbor pixels.

Mathematical Operations on Images

Image Addition

```
a1 = imread('D:\Matlab\bin\cameraman.jpg'); % Read the image
a2 = imread('D:\Matlab\bin\lena.jpg'); % Read the image
a1=imresize(a1,[255,255]);
a2=imresize(a2,[255,255]);
g=imadd(a1,a2,'uint8');
figure(1)
        subplot(1,3,1),imshow(a1),title('Original Image');
subplot(1,3,2),imshow((a2)),title('Image 2');subplot(1,3,3),imshow(g);
```

Original Image

Image 2

Image Subtraction

```
a1 = imread('D:\Matlab\bin\cameraman.jpg'); % Read the image
a2 = imread('D:\Matlab\bin\lena.jpg'); % Read the image
a1=imresize(a1,[255,255]);
a2=imresize(a2,[255,255]);
g=imadd(a1,a2,'uint8');
figure(1)
        subplot(1,3,1),imshow(a1),title('Original Image');
subplot(1,3,2),imshow((a2)),title('Image 2');subplot(1,3,3),imshow(g);
a3 = imread('D:\Matlab\bin\cameraman.jpg'); % Read the image
a4 = imread('D:\Matlab\bin\lena.jpg'); % Read the image
a3=imresize(a3,[255,255]);
a4=imresize(a4,[255,255]);
figure(2)
        subplot(1,3,1),imshow(a3),title('Original Image');
subplot(1,3,2),imshow((a4)),title('Image 2');subplot(1,3,3),imshow(z);
```

Original Image

Image 2

Mathematical Operations on Images

Image Multiplication

$G(x,y)=f1(x,y)*f2(x,y)$

Image Division

$G(x,y)=f1(x,y)/f2(x,y)$

Image Enhancement

objective of image enhancement is to improve the interpretability of the information present in images for human viewers.

enhancement algorithm is one that yields a better-quality image for the purpose of some particular application which can be done by either suppressing the noise or increasing the image contrast.

Spatial Domain Methods

Image-enhancement techniques can be classified into two broad categories as

(1) spatial domain method

The spatial domain method operates directly on pixels or raw data.

(2) **transform domain method**

Transform domain method operates on the Fourier transform of an image and then transforms it back to the spatial domain.

Spatial Domain Methods

When the neighborhood is 1 x 1 then g depends only on the value of f at (x,y) and T becomes a gray-level transformation (or mapping) function:

$$s=T(r)$$

r,s: gray levels of f(x,y) and g(x,y) at (x,y)

- Point processing techniques (e.g. contrast stretching, thresholding)

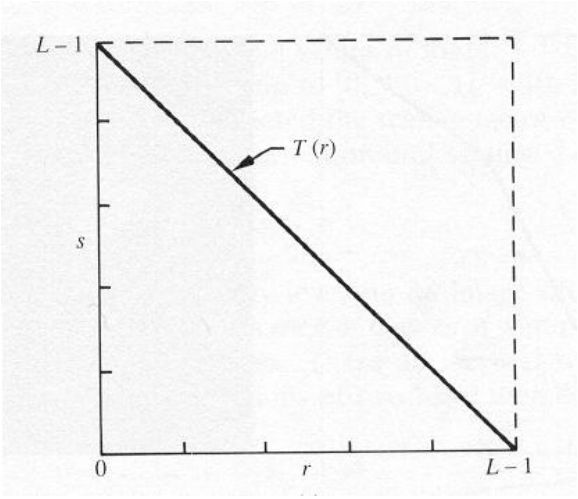
Enhancement by Point Processing

These are methods based only on the intensity of single pixels.

- r denotes the pixel intensity before processing.
- s denotes the pixel intensity after processing.

Image Negatives

Are obtained by using the transformation function s=T(r).



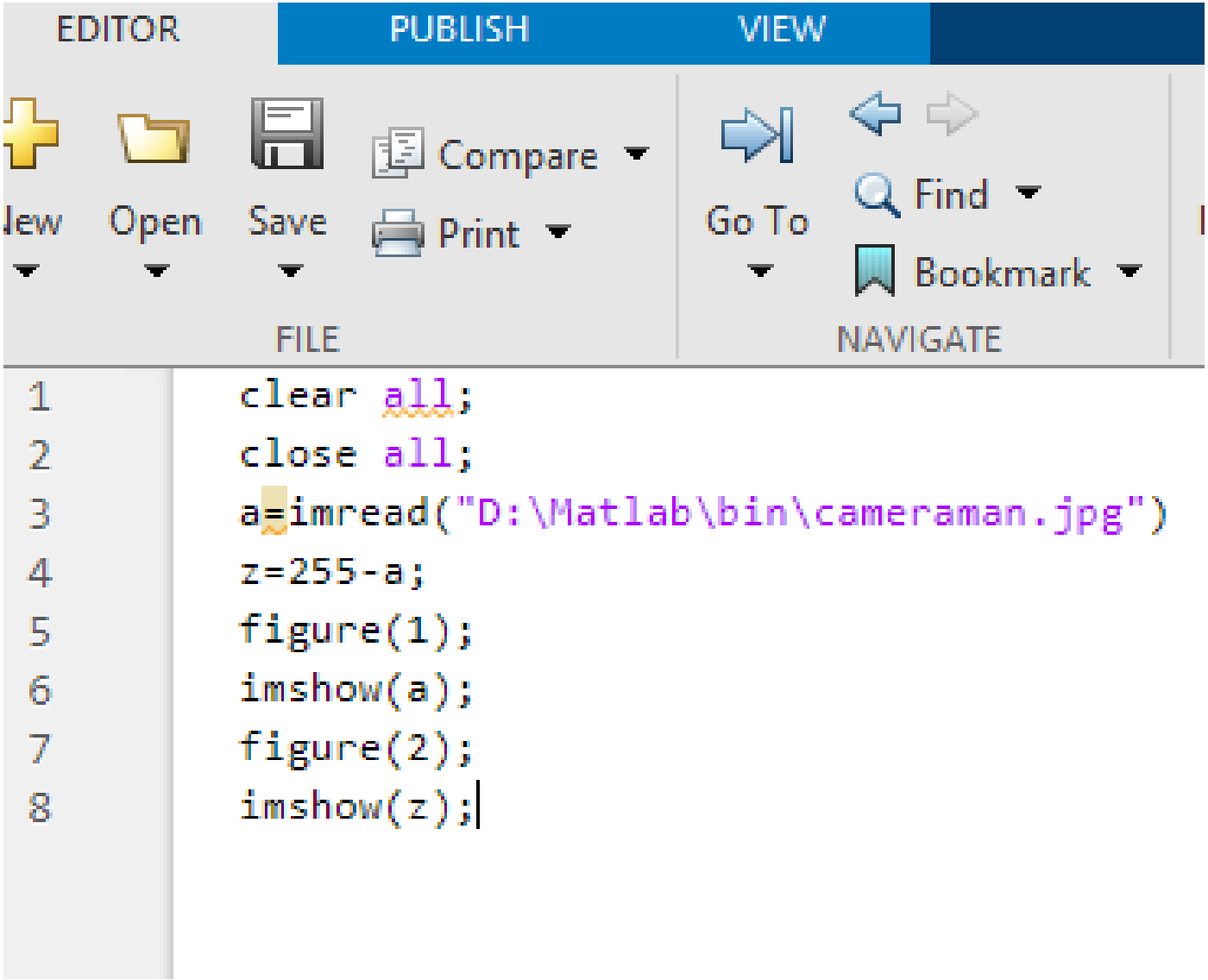
[0,L-1] the range of gray levels

$s = L-1-r$

Image Enhancement in the Spatial Domain

a b

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


EDITOR
PUBLISH
VIEW
New
Open
Save
Compare
Print
FILE
Go To
Find
Bookmark
NAVIGATE
1 clear all;
2 close all;
3 a=imread("D:\Matlab\bin\cameraman.jpg")
4 z=255-a;
5 figure(1);
6 imshow(a);
7 figure(2);
8 imshow(z);

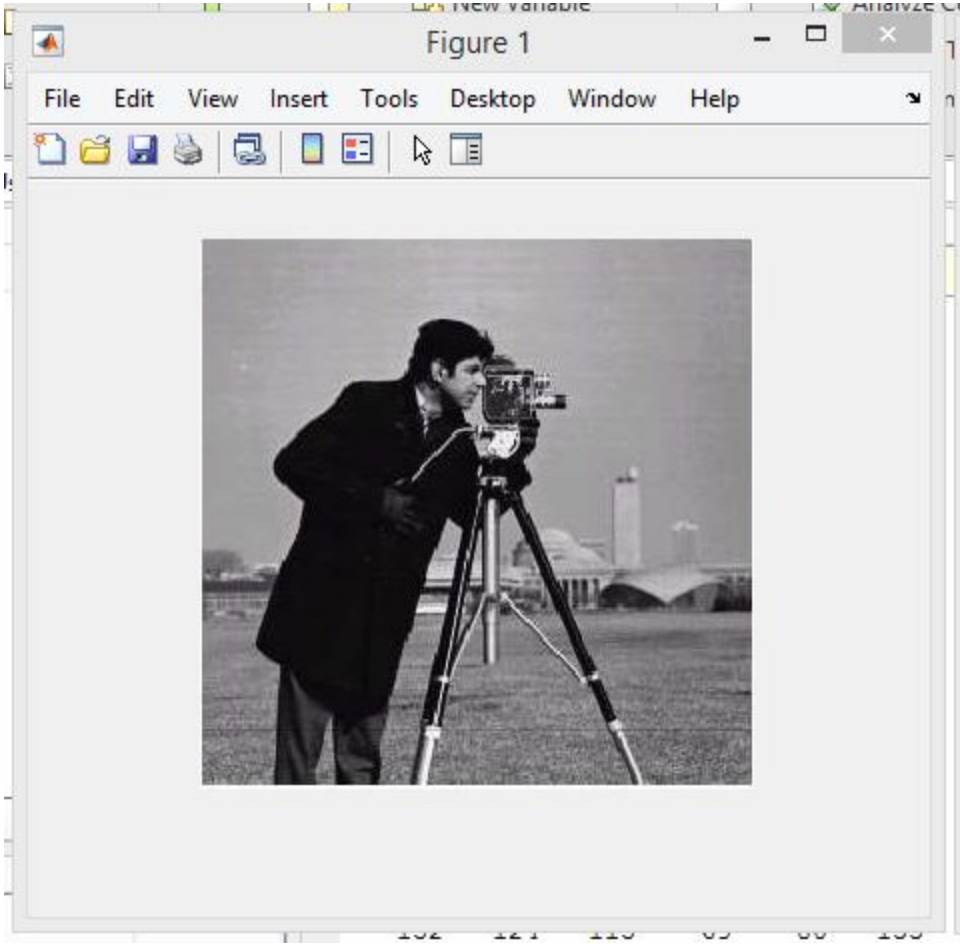


Figure 1
File Edit View Insert Tools Desktop Window Help

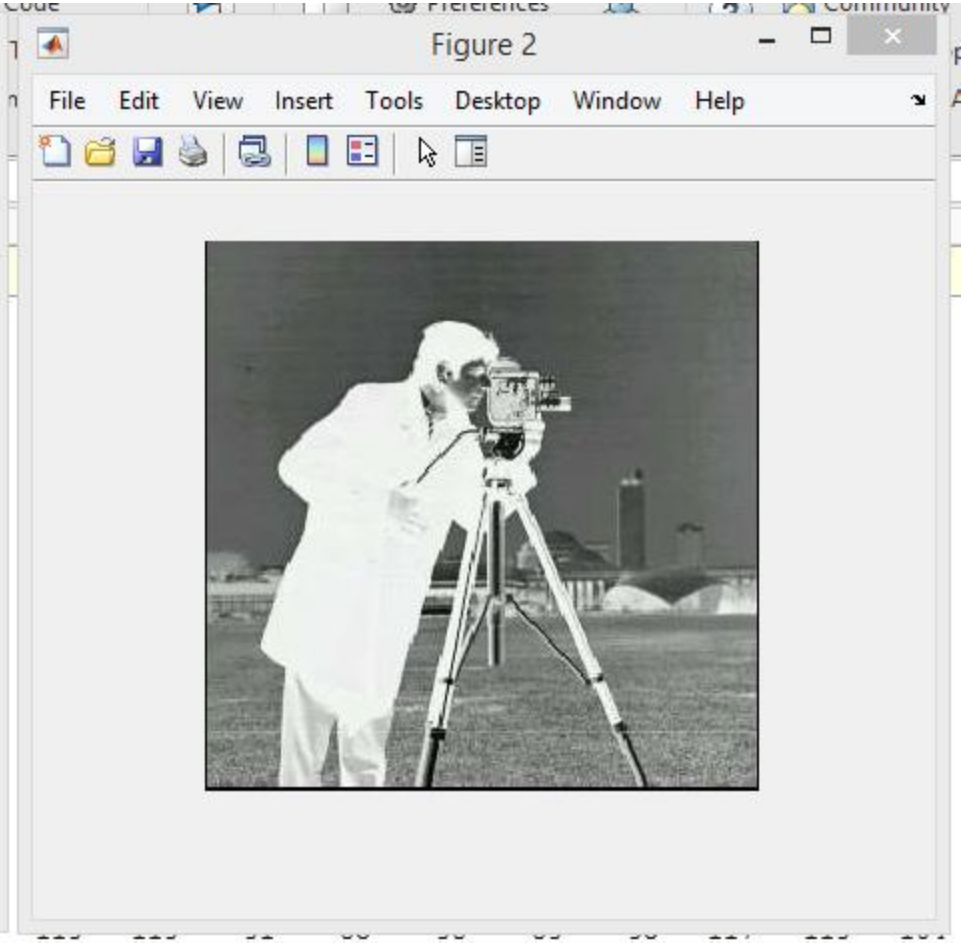
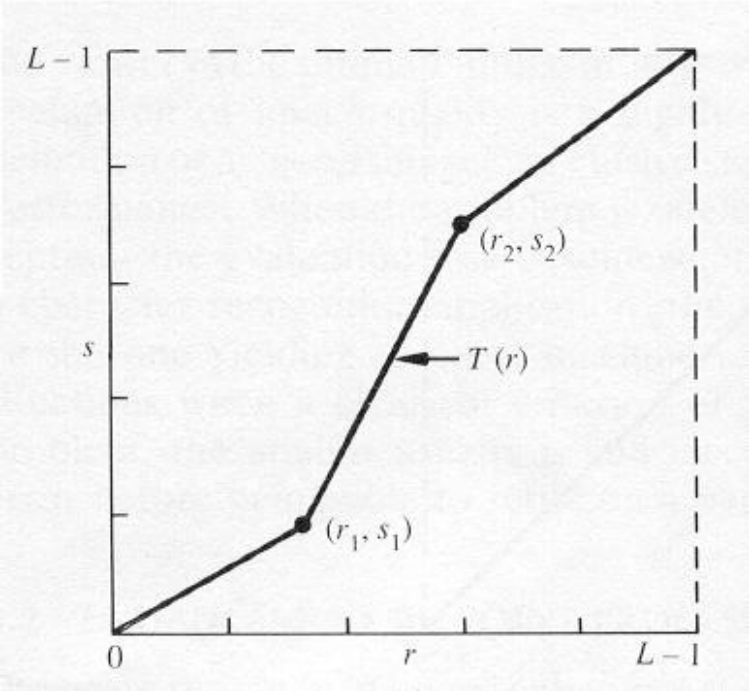


Figure 2
File Edit View Insert Tools Desktop Window Help

Piecewise-Linear Transformation Functions

Contrast Stretching

To increase the dynamic range of the gray levels in the image being processed.



Contrast Stretching

The locations of (r_1,s_1) and (r_2,s_2) control the shape of the transformation function.

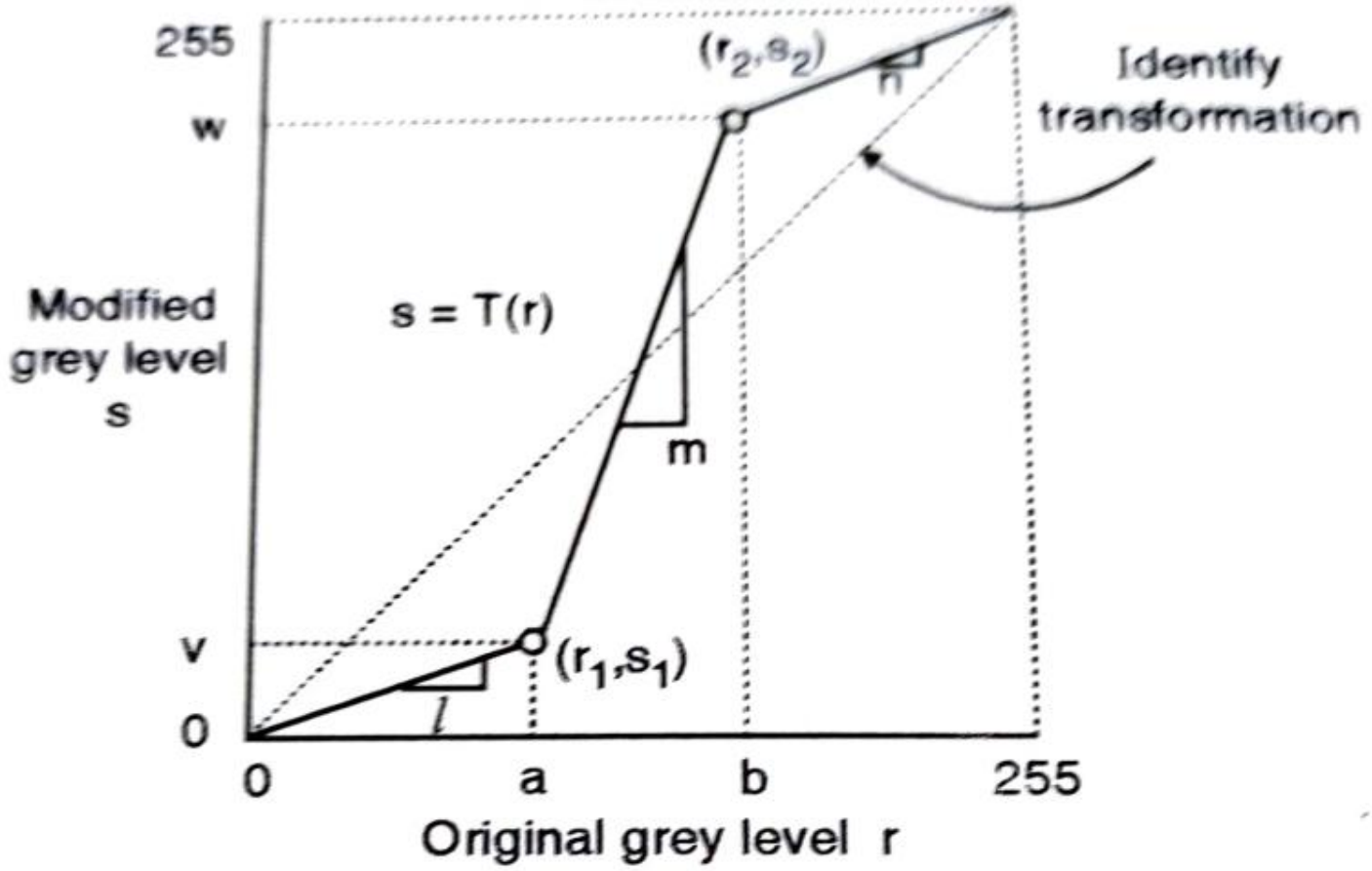
- If $r_1 = s_1$ and $r_2 = s_2$ the transformation is a linear function and produces no changes.
- If $r_1=r_2$, $s_1=0$ and $s_2=L-1$, the transformation becomes a thresholding function that creates a binary image.

Contrast Stretching

More on function shapes:

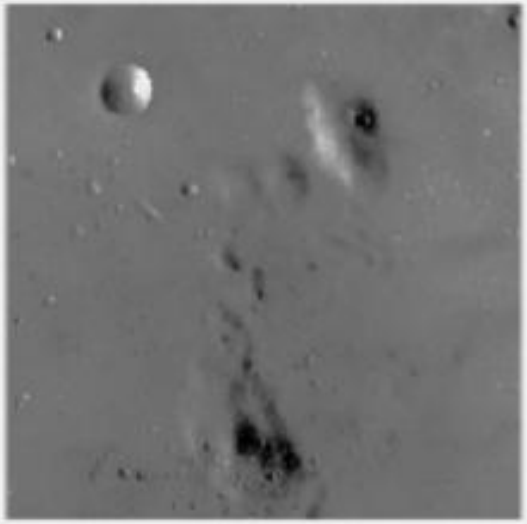
- Intermediate values of (r_1,s_1) and (r_2,s_2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.

- Generally, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed.

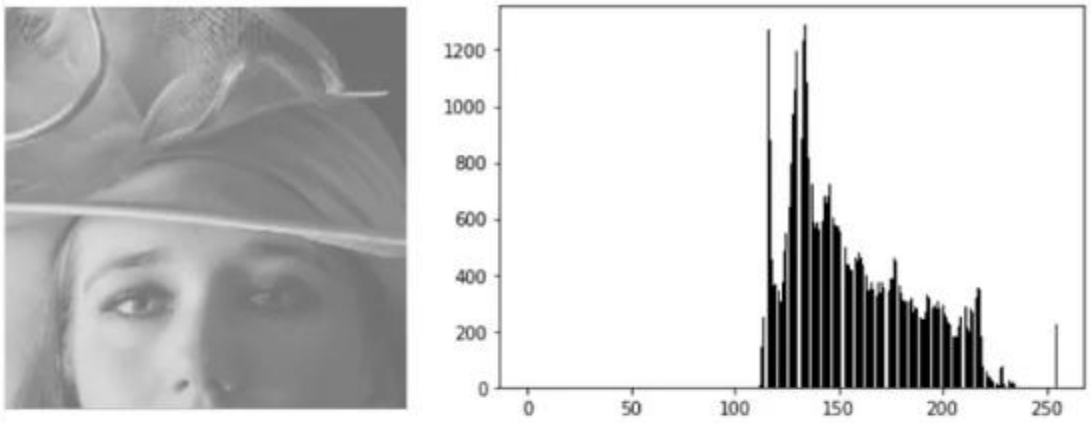


$$s = \begin{cases} l \cdot r & 0 \le r < a \\ m \cdot (r-a) + v & a \le r < b \\ n \cdot (r-b) + w & b \le r < L-1 \end{cases}$$

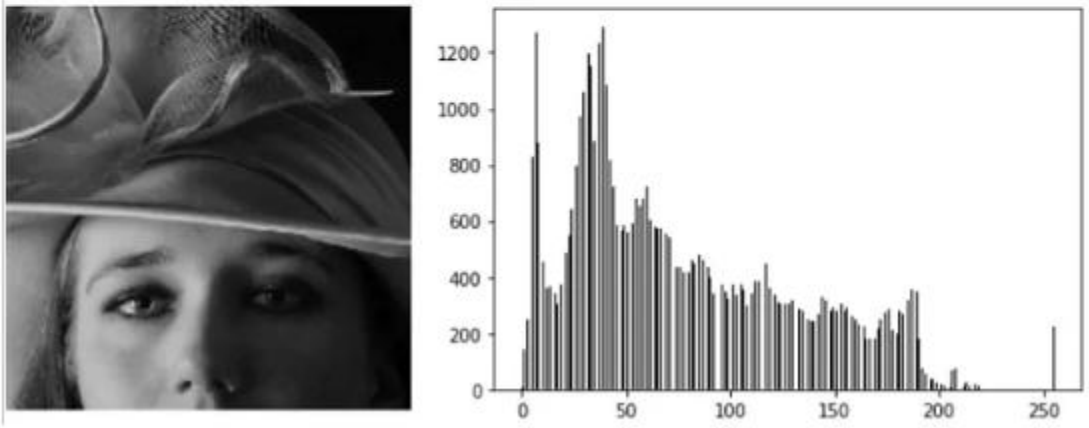
Original Image



Strethced Image



Input Image before Contrast Stretching with its histogram



Input Image after Contrast Stretching with its histogram

Spatial Domain - Point Processing

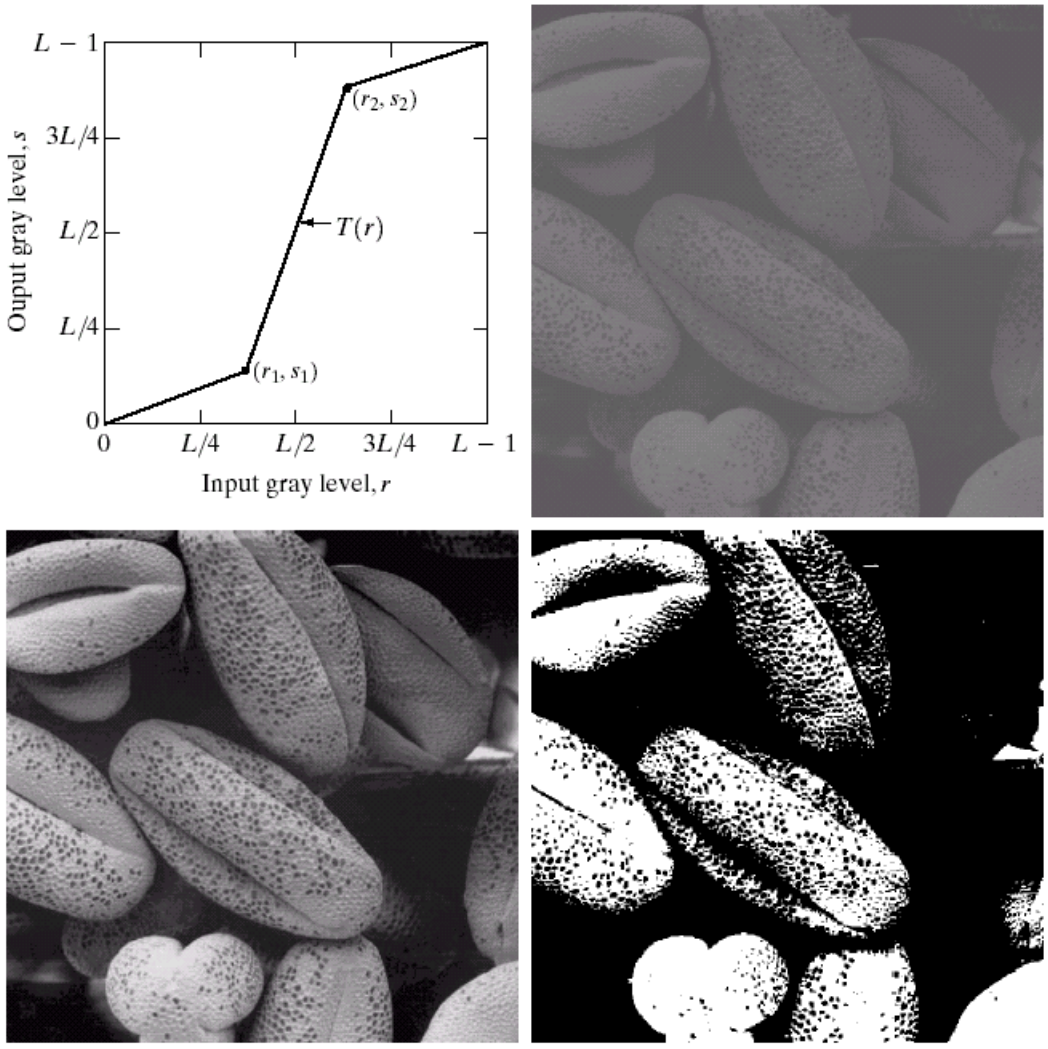
3) Thresholding

Extreme contrast stretching yields in thresholding.

S=255; if r(pixel intensity value) >=a (threshold value)

S=0 if r(pixel intensity value) < a(threshold value)

Image Enhancement in the Spatial Domain

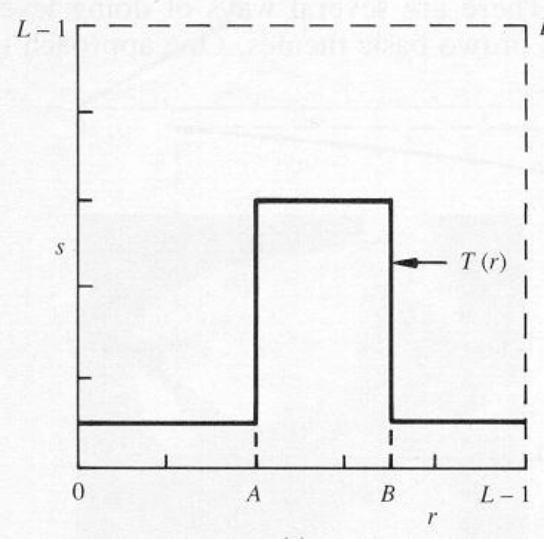


a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Gray-Level Slicing

To highlight a specific range of gray levels in an image (e.g. to enhance certain features).



One way is to display a high value for all gray levels in the range of interest and a low value for all other gray levels (binary image).

Gray-Level Slicing

- The second approach is to brighten the desired range of gray levels but preserve the background and gray-level tonalities in the image:

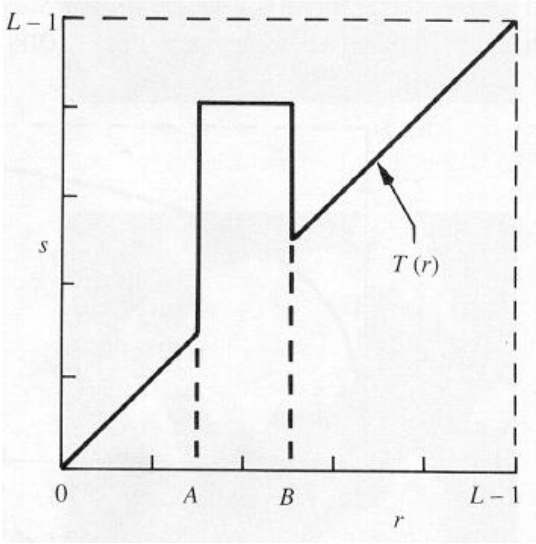
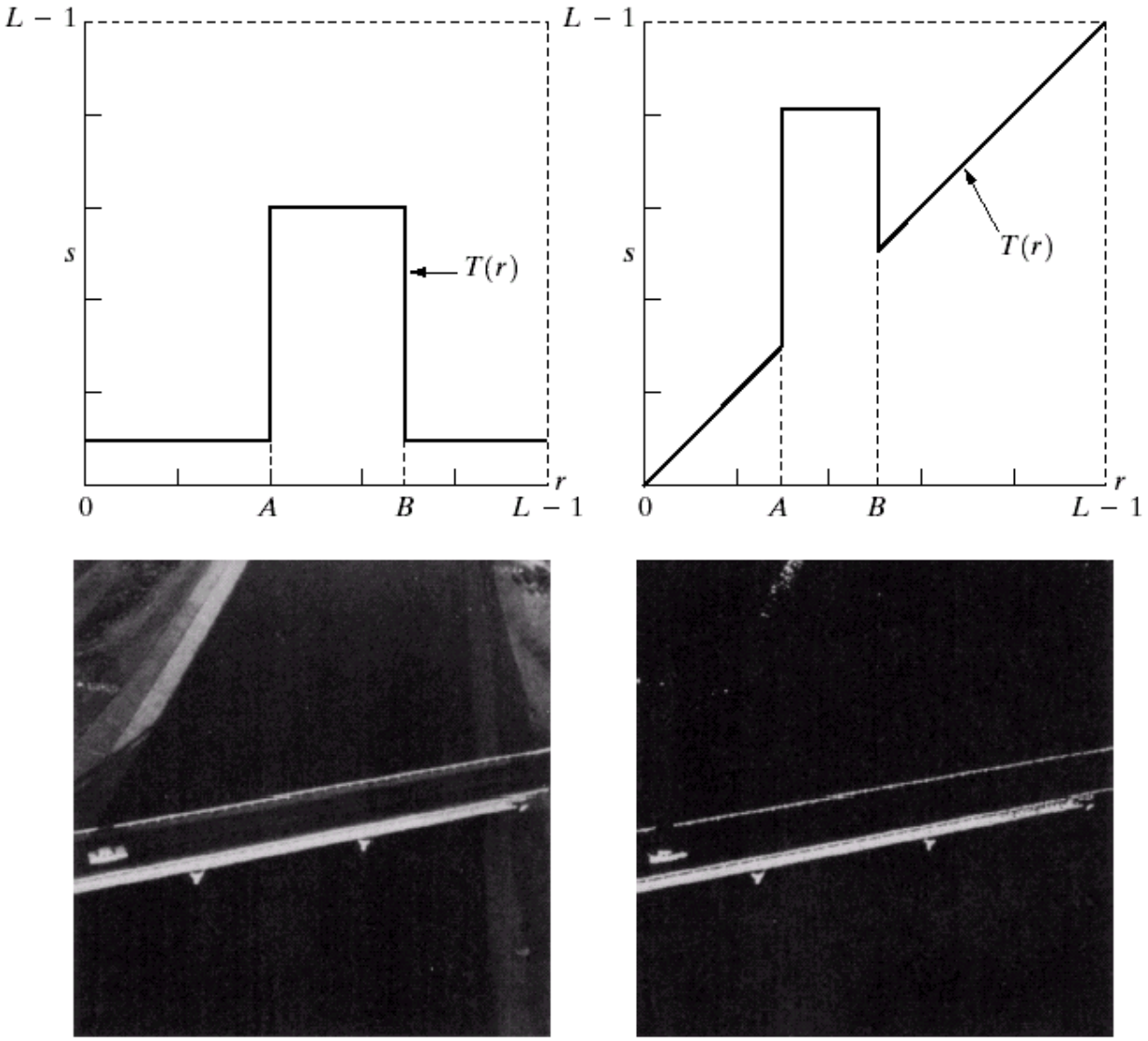


Image Enhancement in the Spatial Domain



a b
c d

FIGURE 3.11
(a) This transformation highlights range $[A, B]$ of gray levels and reduces all others to a constant level. (b) This transformation highlights range $[A, B]$ but preserves all other levels. (c) An image. (d) Result of using the transformation in (a).

Bit-Plane Slicing

To highlight the contribution made to the total image appearance by specific bits.

- i.e. Assuming that each pixel is represented by 8 bits, the image is composed of 8 1-bit planes.
- Plane 0 contains the least significant bit and plane 7 contains the most significant bit.

Bit-Plane Slicing

More on bit planes:

- Only the higher order bits (top four) contain visually significant data. The other bit planes contribute the more subtle details.
- Plane 7 corresponds exactly with an image thresholded at gray level 128.

Image Enhancement in the Spatial Domain

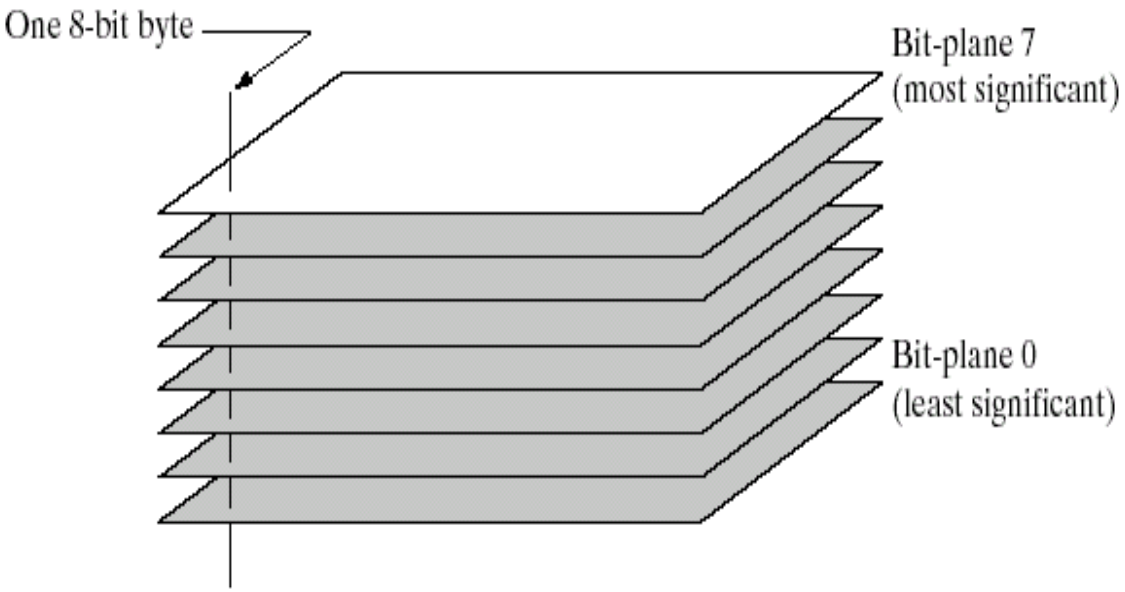


FIGURE 3.12
Bit-plane representation of an 8-bit image.

Image Enhancement in the Spatial Domain

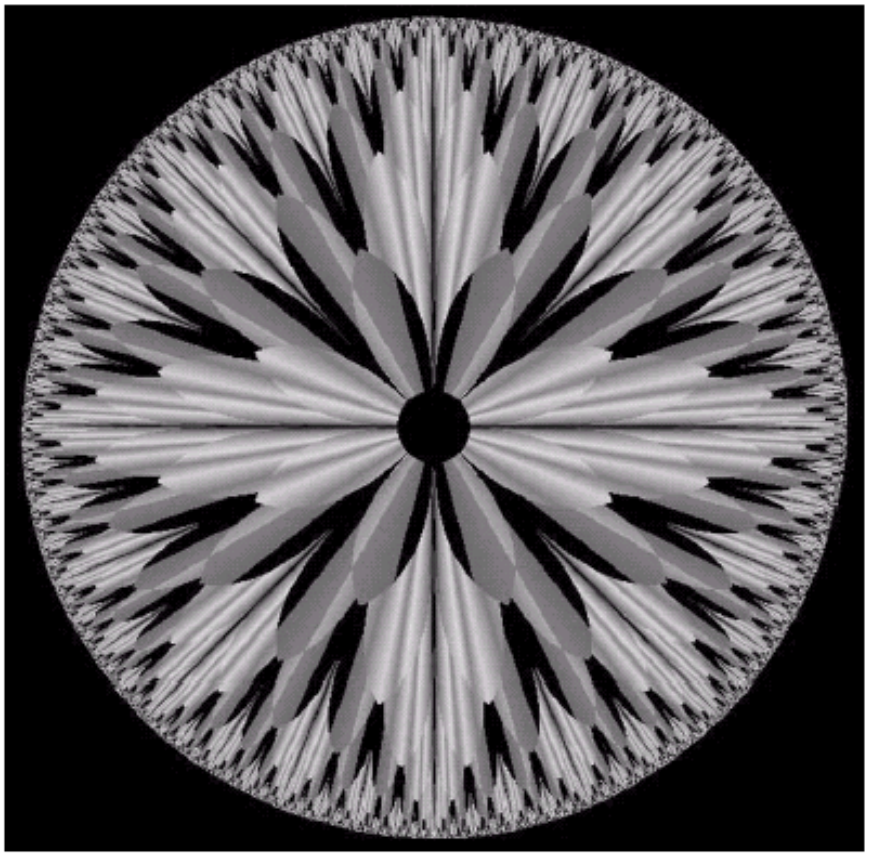


FIGURE 3.13 An 8-bit fractal image. (A fractal is an image generated from mathematical expressions). (Courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA.)

Image Enhancement in the Spatial Domain

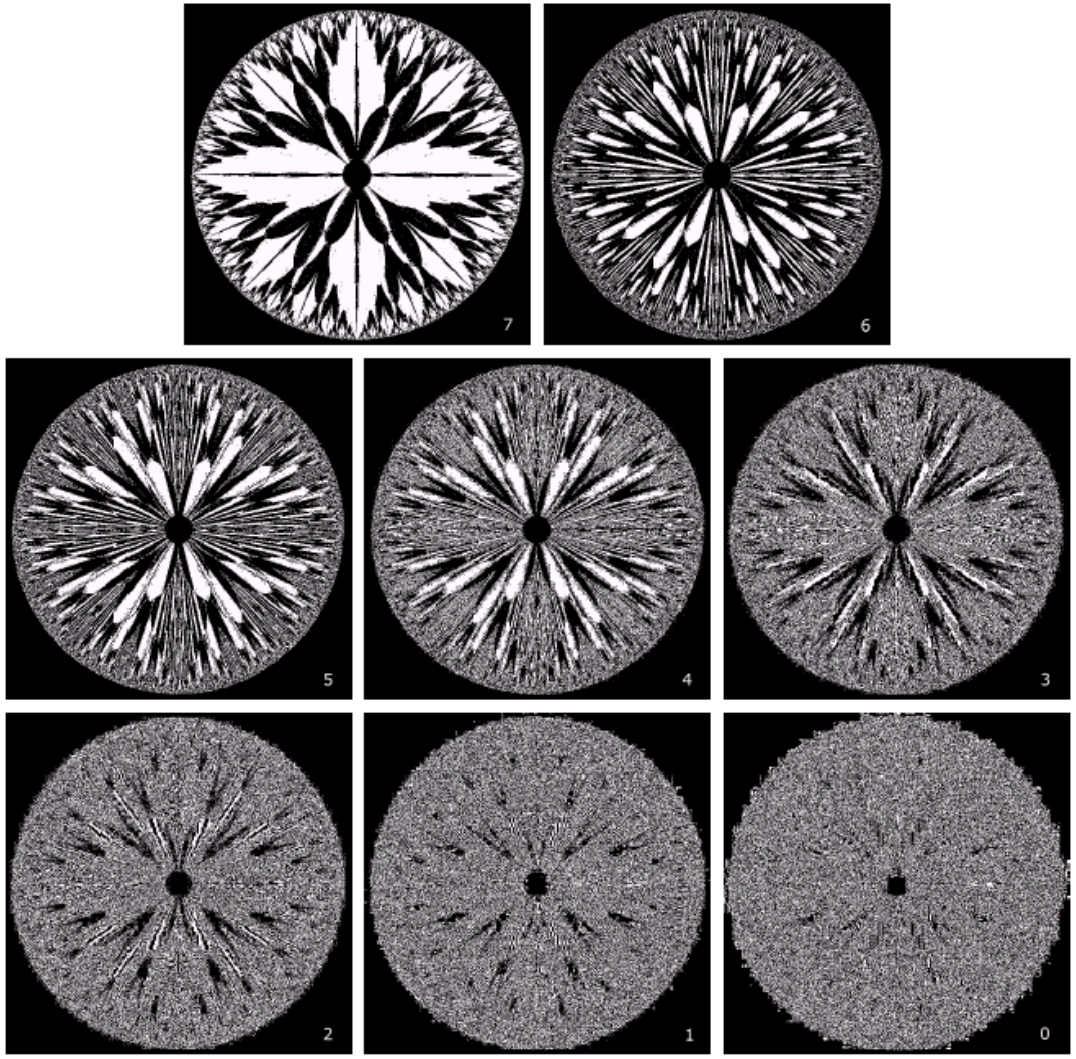


FIGURE 3.14 The eight bit planes of the image in Fig. 3.13. The number at the bottom, right of each image identifies the bit plane.

Spatial Domain - Point Processing

Ex. 7.1 : Given a 3 × 3 image, plot its bit planes.

1	2	0
4	3	2
7	5	2

Soln. :

Since 7 is the maximum grey level, we need only 3-bits to represent the grey levels.

Hence we will have 3-bit planes. Converting the image to binary we get,

001	010	000
100	011	010
111	101	010

Binary image

1	0	0
0	1	0
1	1	0

LSB plane

0	1	0
0	1	1
1	0	1

Middle bit plane

0	0	0
1	0	0
1	1	0

MSB plane

```
clear all;
close all;
clc;
a=imread('D:\Matlab\bin\lena.jpg');
b1=[];
b2=[];
b3=[];
b4=[];
b5=[];
b6=[];
b7=[];
b8=[];
for m=1:256
    for n=1:256
        t=de2bi(a(m,n),8,'left-msb');
        b1(m,n)=t(1,1);
        b2(m,n)=t(1,2);
        b3(m,n)=t(1,3);
        b4(m,n)=t(1,4);
        b5(m,n)=t(1,5);
        b6(m,n)=t(1,6);
        b7(m,n)=t(1,7);
        b8(m,n)=t(1,8);
    end
end

subplot(3,3,1);
imshow(a);
title('image of cameramen','color','r');
```

```
subplot(3,3,1);
imshow(a);
title('image of cameramen','color','r');

subplot(3,3,2);
imshow(b8);
title('image of bit-1','color','r');
subplot(3,3,3);
imshow(b7);
title('image of bit-2','color','r');
subplot(3,3,4);
imshow(b6);
title('image of bit-3','color','r');
subplot(3,3,5);
imshow(b5);
title('image of bit-4','color','r');
subplot(3,3,6);
imshow(b4);
title('image of bit-5','color','r');

subplot(3,3,7);
imshow(b3);
title('image of bit-6','color','r');

subplot(3,3,8);
imshow(b2);
title('image of bit-7','color','r');
```

Figure 1

File Edit View Insert Tools Desktop Window Help

image of cameramen

image of bit-1

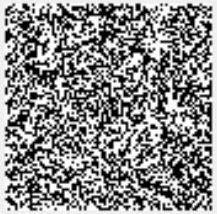


image of bit-2

image of bit-3

image of bit-4

image of bit-5

image of bit-6

image of bit-7

image of bit-8

Spatial Domain - Point Processing

6) Dynamic Range Compression (Log Transformation)

Dynamic range of the image exceeds the capability of the display device.

Some pixel values are so high that low pixel values get over shadowed.

To be able to see the small value pixels dynamic range compression is used.

Spatial Domain - Point Processing

Log operator being the excellent compressing function.

Dynamic range compression is achieved by using log operator.

Log Transformation = C*log(1+|a|)…(a=image)

Log Transformations

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

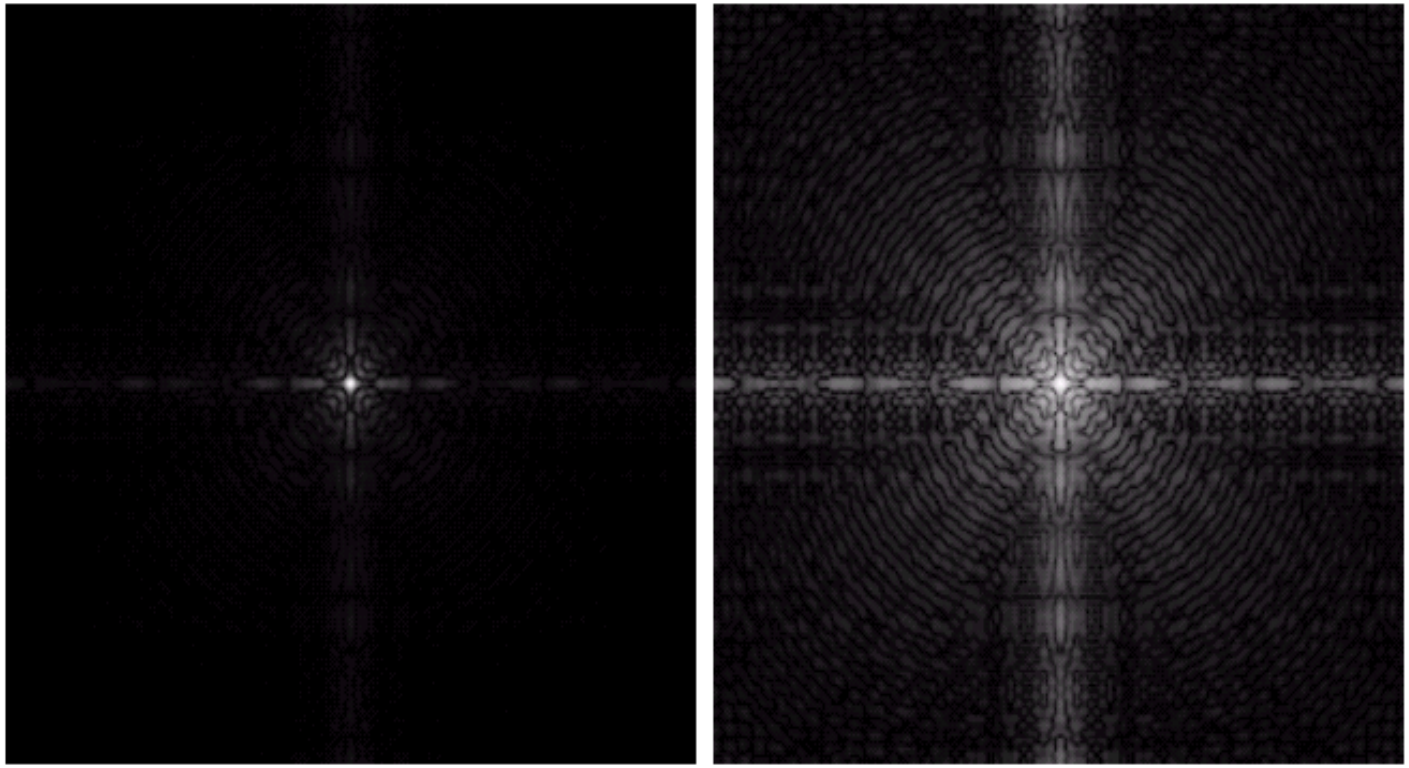
c: constant

Compresses the dynamic range of images with large variations in pixel values

Image Enhancement in the Spatial Domain

a b

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



Power-Law Transformations

$$s = cr^{\gamma}$$

c, γ : positive constants

Gamma correction

Image Enhancement in the Spatial Domain

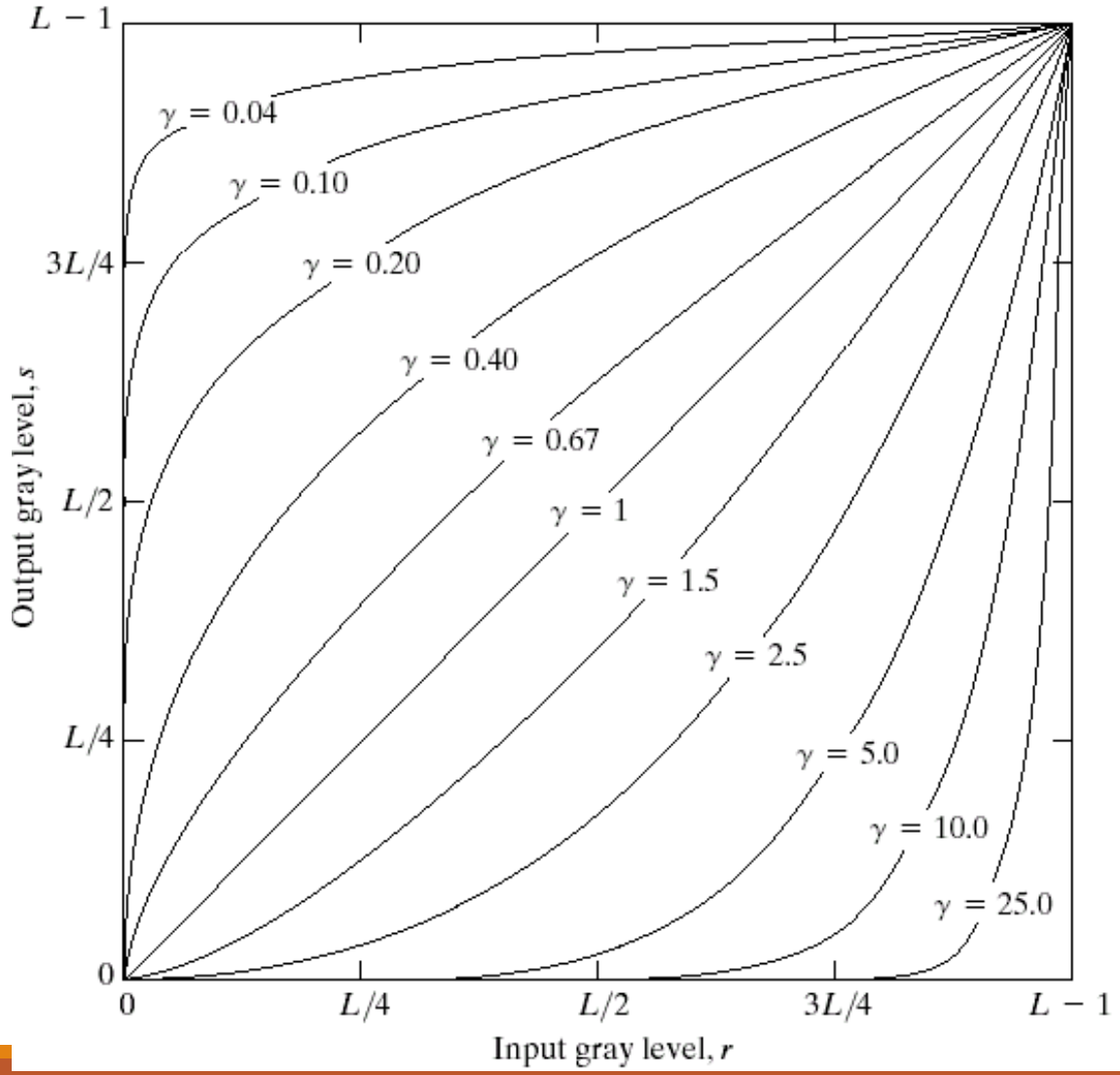


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

γ=c=1: identity

Image Enhancement in the Spatial Domain

a b
c d

FIGURE 3.7
(a) Linear-wedge gray-scale image. (b) Response of monitor to linear wedge. (c) Gamma-corrected wedge. (d) Output of monitor.

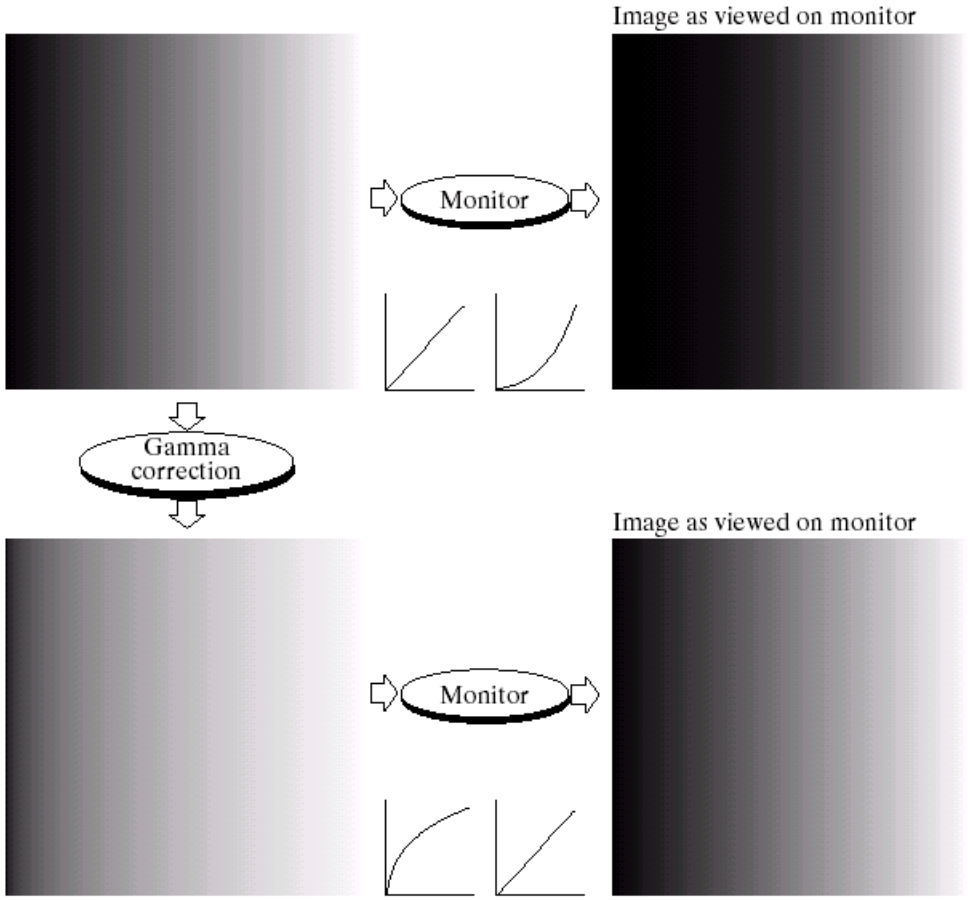
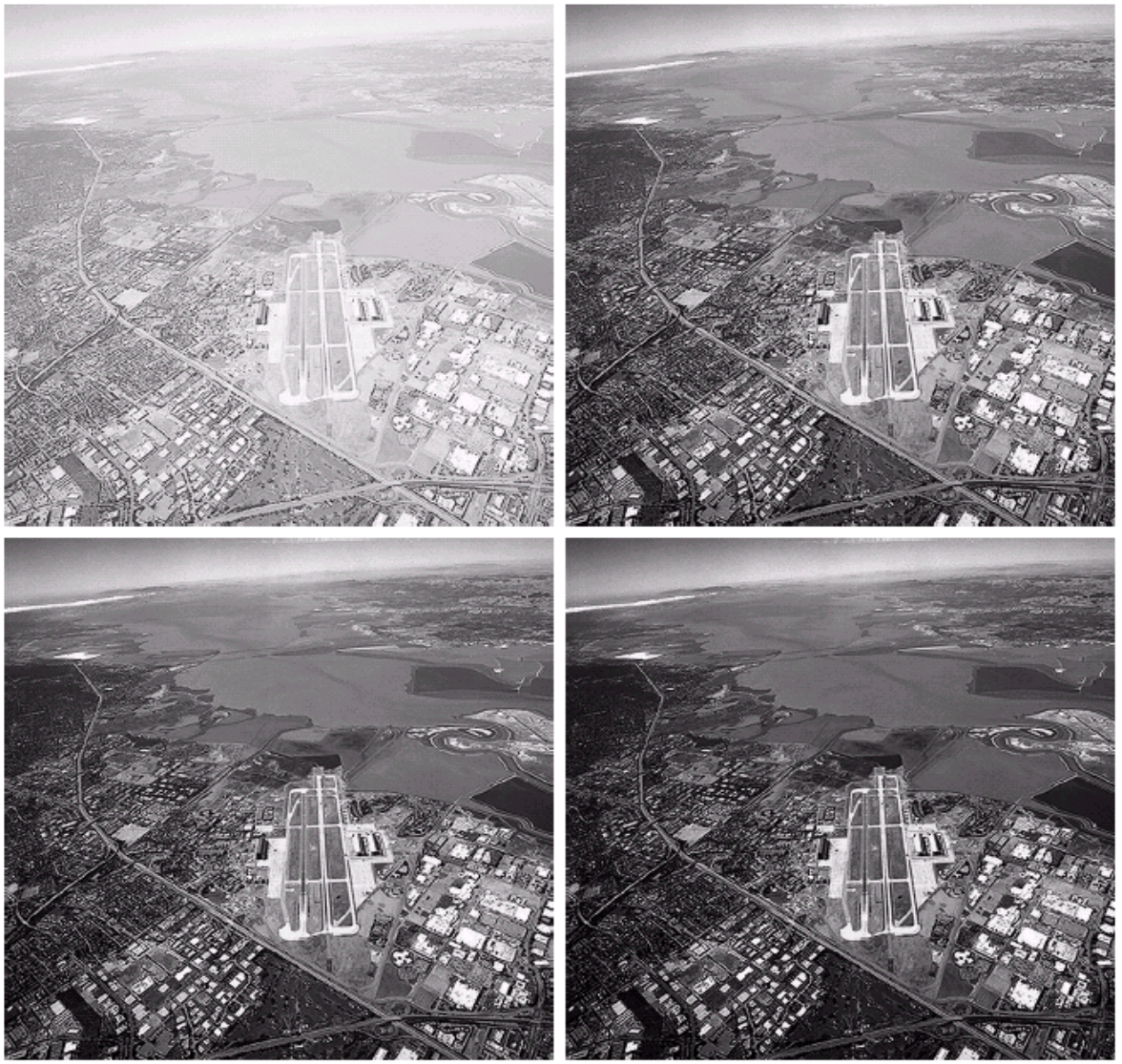


Image Enhancement in the Spatial Domain

a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)



FILE NAVIGATE CODE ANALYZ

```
1  a1 = imread('D:\Matlab\bin\saturn.jpg'); % Read the image
2  a = double(a1)/256; % Normalized Image
3  c = 2; % Constant
4  f = c*log(1 + (a)); % Log Transform
5  subplot(1,2,1),imshow(a1),title('Original Image');
6  subplot(1,2,2),imshow((f)),title('Log Transformation Image');
```

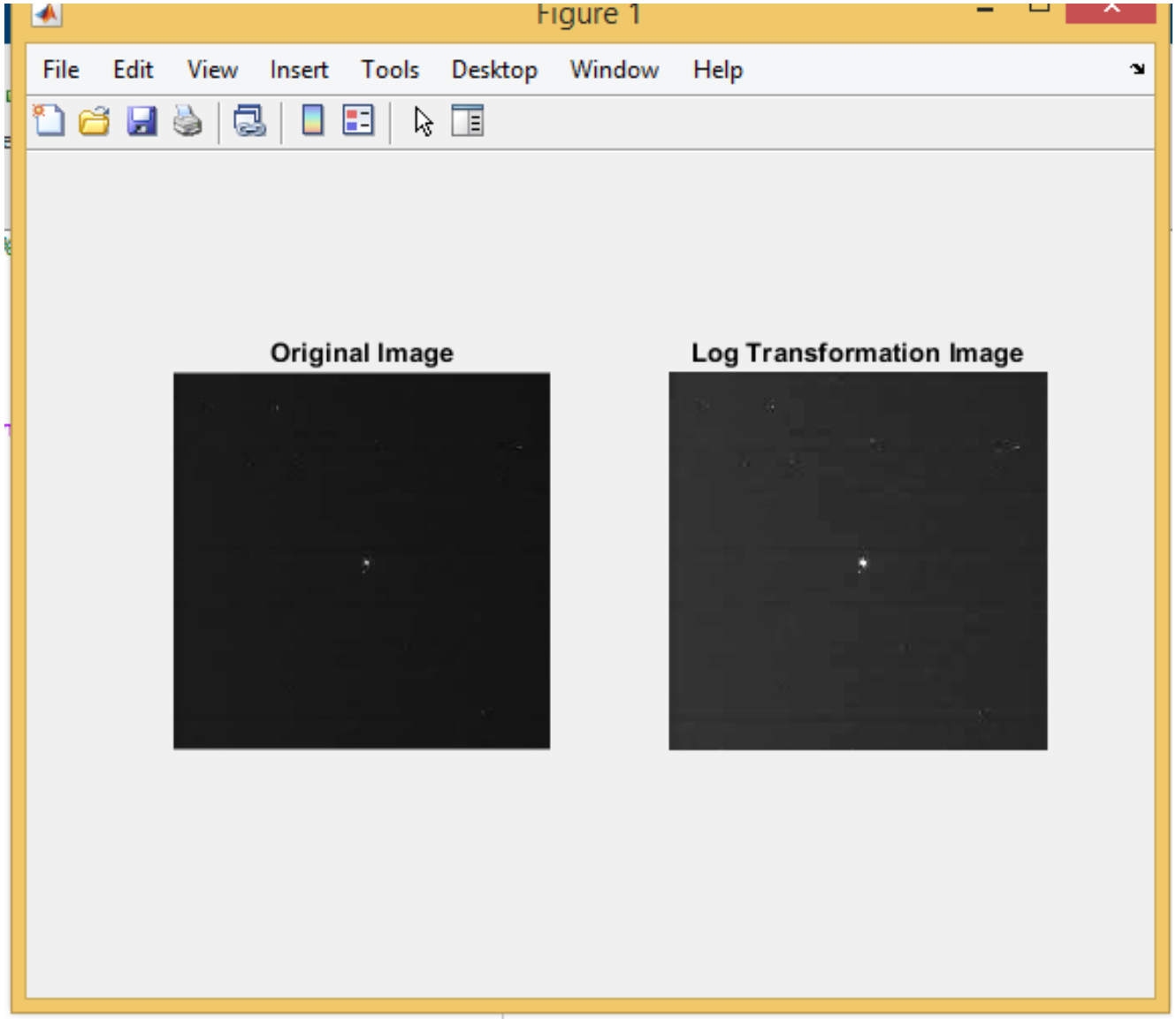


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Log Transformation Image

C:\Users\Admin\Documents\MATLAB\image9.m

TOR PUBLISH VIEW

Open Save Compare Print FILE

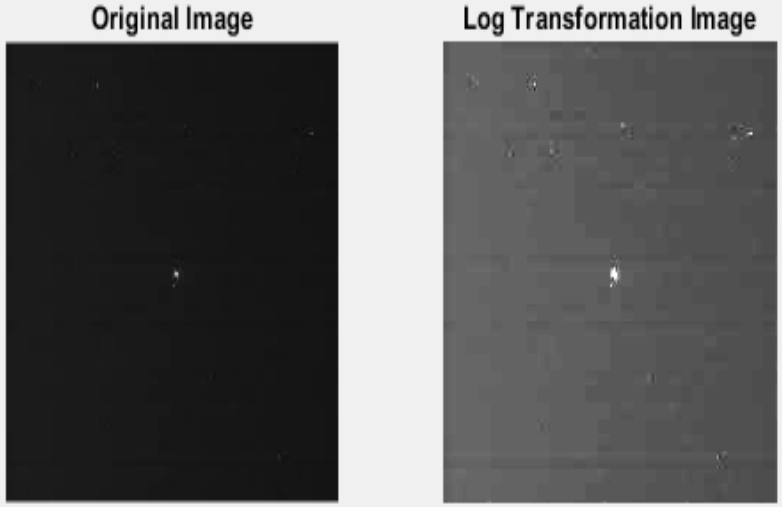
Go To Find Bookmark NAVIGATE

Refactor CODE

```
a1 = imread('D:\Matlab\bin\saturn.jpg'); % Read the image
a = double(a1)/256; % Normalized Image
c = 4; % Constant
f = c*log(1 + (a)); % Log Transform
subplot(1,2,1),imshow(a1),title('Original Image');
subplot(1,2,2),imshow((f)),title('Log Transformation Image');
```

Figure 1

File Edit View Insert Tools Desktop Window Help

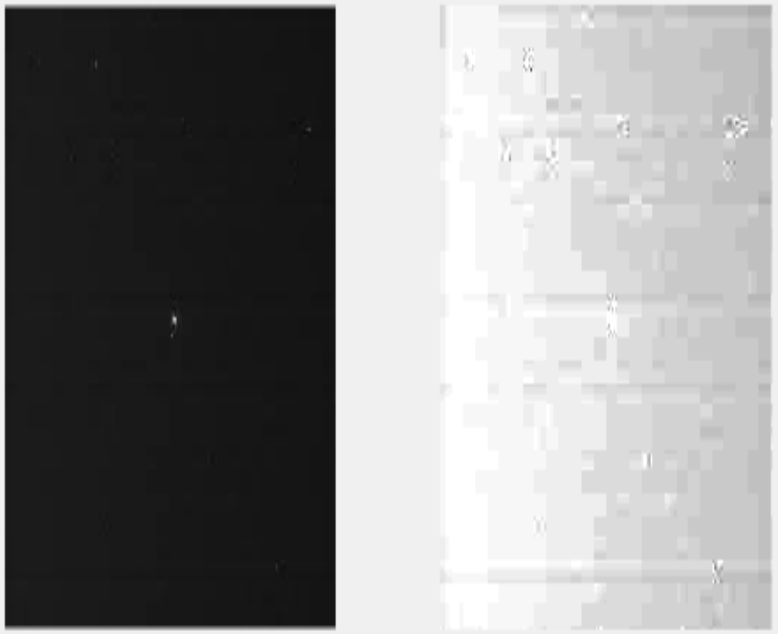


FILE NAVIGATE CODE

Bookmark

```
a1 = imread('D:\Matlab\bin\saturn.jpg'); % Read the image
a = double(a1)/256; % Normalized Image
c = 10; % Constant
f = c*log(1 + (a)); % Log Transform
subplot(1,2,1),imshow(a1),title('Original Image');
subplot(1,2,2),imshow((f)),title('Log Transformation Image');
```

File Edit View Insert Tools Desktop Window Help



Spatial Domain - Point Processing

7) Power Law Transformation

human perception of brightness - more sensitive to changes in dark as compared to bright.

display devices like computer screen have Intensity to voltage (non linear) - which is a power function with exponents(Gamma) varying from 1.8 to 2.5.

any input signal(say from a camera), the output will be transformed by gamma because of non-linear intensity to voltage relationship of the display screen - This results in images that are darker than intended.

Spatial Domain - Point Processing

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

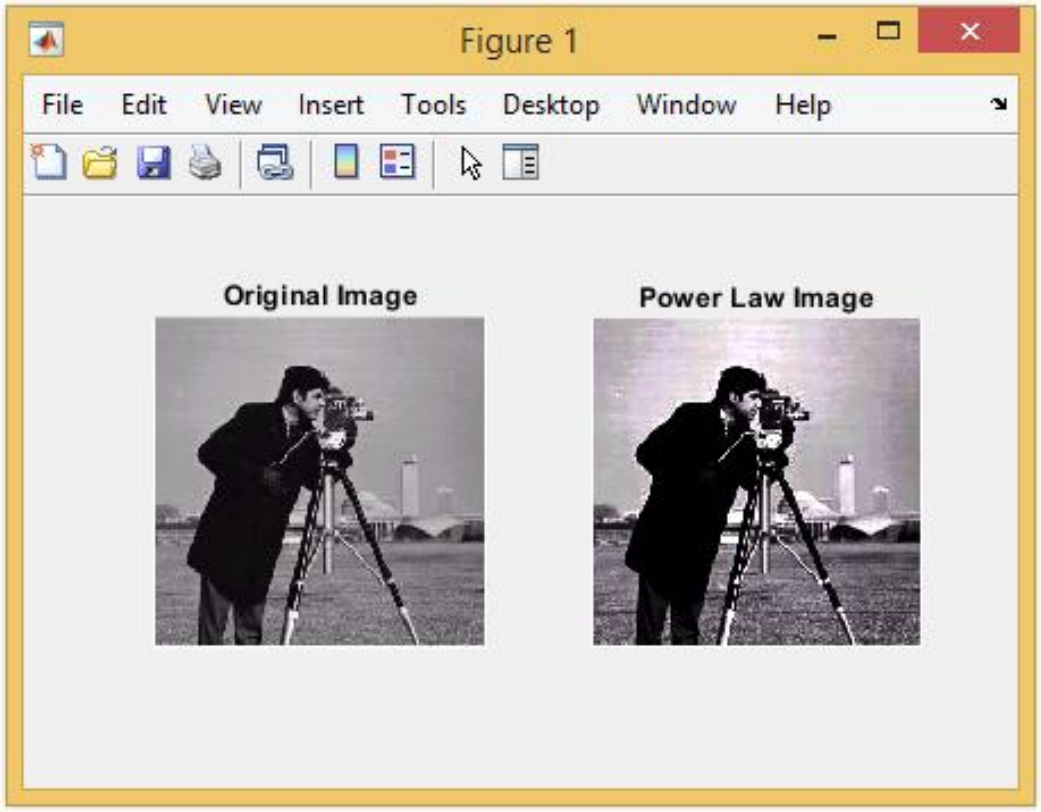
we apply gamma correction to the input signal.

This input cancels out the effects generated by the display and we see the image as it is.

```
a1 = imread('D:\Matlab\bin\cameraman.jpg'); % Read the image
a = double(a1)/256; % Normalized Image
gamma=input('Enter the correction factor gamma value:')
c=2;
f = c*a.^gamma;
subplot(1,2,1),imshow(a1),title('Original Image');
subplot(1,2,2),imshow((f)),title('Power Law Image');
```

```
>> image10
Enter the correction factor gamma value:0.4

gamma =

    0.4000

>>
```

```
>> image10
Enter the correction factor gamma value:2.2

gamma =

    2.2000

>>
```



Filtering in spatial domain

refers to image operators that change the gray value at any pixel (x,y) depending on the pixel values in a square neighborhood centered at (x,y) using a fixed integer matrix of the same size. The integer matrix is called a *filter*, *mask*, *kernel* or a *window*.

Neighbourhood Processing

In neighbourhood operation, the pixels in an image are modified based on some function of the pixels in their neighbourhood.

Instead of 3*3 neighbourhood , we could also use 5*5 or a 7*7 …….

Low Pass Averaging Filter (Box Filter)

If an image has Gaussian noise present in it a low pass averaging filter is used to eliminate it.

The mean filter replaces each pixel by the average of all the values in the local neighbourhood.

The size of the neighbourhood controls the amount of filtering.

Low Pass Averaging Filter

spatial averaging operation, each pixel is replaced by a weighted average of its neighbourhood pixels.

The low-pass filter preserves the smooth region in the image and it removes the sharp variations leading to blurring effect.

Low Pass Averaging Filter

$$\text{3 by 3 low-pass spatial mask} = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{Similarly, the 5 by 5 averaging mask} = \frac{1}{25} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Low Pass Averaging Filter

It is to be noted that the sum of the elements is equal to 1 in the case of a low-pass spatial mask.

The blurring effect will be more with the increase in the size of the mask.

the size of the mask will be odd so that the central pixel can be located exactly.

→ Low Pass Averaging Filter

Given Image :-

8×8 Image

$$\begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \end{bmatrix}$$

Mask that will be used is 3×3

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Iteration 1 :—

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \end{bmatrix}$$

After multiplication & average of each element we get center pixel as 10 so no change.

→ Similarly the mask will be shifted to the next position row-wise and column-wise

$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

$\Rightarrow 1\times10 + 1\times10 + 1\times10 + 1\times10 + 1\times10 + 1\times10 + 1\times50 + 1\times50 + 1\times50$

$\Rightarrow 23.33$

→ Similarly the other values can be calculated.

→ The result of convolving the image with the 3×3 averaging mask

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
23.33	23.33	23.33	23.33	23.33	23.33	23.33	23.33
36.6	36.6	36.6	36.6	36.6	36.6	36.6	36.6
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

Low Pass Averaging Filter

Low frequency regions remain unchanged while sharp edges become blur.

Grey level transition from 10 to 50 reduces to 10 -23.33 -36.6-50 blurring effect.

Low Pass Averaging Filter

Limitations of Averaging Filter

1. Averaging operation **leads to the blurring** of an image. Blurring affects feature localisation.

2. If the averaging operation is applied to an image corrupted by impulse noise then the impulse noise is attenuated and diffused but not removed.

3. A single pixel with unrepresentative value can affect the mean value of the centre pixel.

Low Pass Averaging Filter

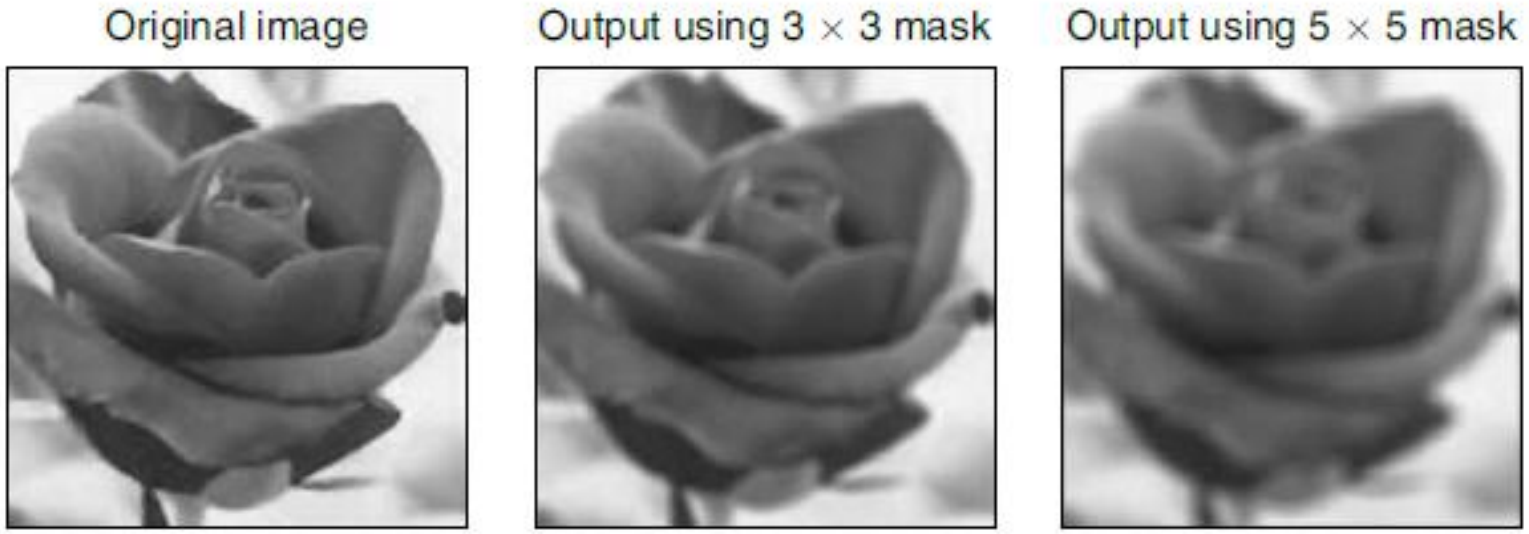


Fig. 5.33 *Results of spatial domain filters*

Weighted Average Filter

The mask of a weighted average filter is given by

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

the pixels nearest to the ıan the distant pixels

Weighted Average Filter

In the mask the pixel at the center of the mask is multiplied by a higher value than any other, thus giving this pixel more importance in the calculation of the average.

Median Filter

Median filters are statistical non linear filters in spatial domain.

Median filter smoothens the image by utilising the median of the neighbourhood.

Steps performed by median filter to find each pixel value in processed image:

1. All pixels in the neighbourhood of the pixel in the original image which are identified by the mask are sorted in the ascending (or) descending order.
2. The median of the sorted value is computed and is chosen as the pixel value for the processed image.

Median Filter

Example 5.5 *Compute the median value of the marked pixel shown in Fig. 5.40 using a 3 × 3 mask.*

$$\begin{bmatrix} 1 & 5 & 7 \\ 2 & (4) & 6 \\ 3 & 2 & 1 \end{bmatrix}$$

Fig. 5.40 *Data for Example 5.5*

Step 1 First, the pixel values are arranged in ascending order as follows:

1 1 2 2 3 4 5 6 7

Step 2 The median value of the ordered pixel is computed as follows:

1 1 2 2 3 *4 5 6 7*

The median value is computed to be 3. Then, the original pixel value of 4 will be replaced by the computed median value of 3.

$$\begin{bmatrix} 1 & 5 & 7 \\ 2 & (4) & 6 \\ 3 & 2 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 5 & 7 \\ 2 & \boxed{3} & 6 \\ 3 & 2 & 1 \end{bmatrix}$$

Original image data After median filtering

Example 5.6 *Compute the median value of the marked pixels shown in Fig. 5.41 using a 3 × 3 mask.*

$$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & (128) & (24) & (172) & (26) & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$$

Fig. 5.41 *Input image*

Step 1 **To compute the median value of the marked pixel 128**

$$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & (128) & 24 & 172 & 26 & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$$

Step 1a The eight neighbours of the marked pixel '128' are now arranged in ascending order as follows:

18 19 22 22 24 32 33 34 128

Step 1b The median value is computed as **24**.

~~18 19 22 22~~ 24 ~~32 33 34 128~~

Step 2 **To Compute the median value of the marked pixel 24**

Step 2a Arrange the pixels in the neighbourhood (shown by dotted lines) in the ascending order.

$$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & 128 & (24) & 172 & 26 & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$$

After arranging the pixels in the neighbourhood in the ascending order, the values are displayed as follows:

19 22 24 25 31 32 33 128 174

Step 2b The median value is computed to be **31**.

19 22 24 25 31 32 33 128 174

Step 3 **To compute the median value of the marked pixel 172**

Step 3a Arrange the pixels in the neighbourhood of 172 in the ascending order.

$$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & 128 & 24 & (172) & 26 & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$$

After arranging the pixel values in the neighbourhood in the ascending order, we have

24 25 26 28 31 32 32 33 172

Step 3b The median value is computed as **31**.

24 25 26 28 31 32 32 33 172

Step 4 **To compute the median value of the marked pixel 26**

Step 4a Arrange the pixels in the neighbourhood of 26 in the ascending order:

$$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & 128 & 24 & 172 & (26) & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$$

After arranging the pixel values in the neighbourhood in the ascending order, we have

23 24 25 26 26 28 31 32 172

Step 4b The median value is computed as 26.

~~23~~ ~~24~~ ~~25~~ ~~26~~ 26 ~~28~~ ~~31~~ ~~32~~ ~~172~~

The original pixel values and the values replaced by their median are shown side by side below:

$$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & (128) & (24) & (172) & (26) & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix} \longrightarrow \begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & (24) & (31) & (31) & (26) & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$$

Original pixel value **Median value**

Median Filter

When we take the median value, the pixel values which are very different from their neighbouring pixels are replaced by a value equal to the neighbouring pixel value.

median filter is capable of reducing salt-and-pepper noise.

A median filter is an effective tool to minimise salt-and-pepper noise.

Original image

Salt and pepper noise

3 × 3 smoothing

5 × 5 smoothing

3 × 3 median filter

5 × 5 median filter

Fig. 5.43 *Output of the box and median-filtered image*

Comparison Between Filters:

Parameters	Average Filter	Weighted Filter	Median Filter
Noise Reduction	Reduces Noise but it introduces blurring effect at edges.	Blurring effect is less as compared with Average filter	Blurring effect is less as compared with Average filter
Percentage of noise Reduction	100% noise Not Reduced	100% noise Not Reduced	Almost 100% noise reduced.
Size of Filter	As we increase the size of the filter mask, Noise reduces but blurring effect increases.	As we increase the size of the filter mask, Noise reduces but blurring effect increases.	As we increase the size of the filter mask, 100% of Noise reduces but blurring effect at edges increases.
Mask	1/9x[1,1,1;1,1,1;1,1,1]	1/16x[1,2,1;2,4,4;1,2,1]	Pixel value is replaced by median value of neighborhood.
MATLAB Function	filter2(mask,Noisy_img)	filter2(mask,Noisy_img)	medfilt2(Noisy_img,[3 3])

Highpass Filtering

High pass filtering eliminates the low frequency regions while retaining or enhancing the high frequency components.

High pass image will have no background as it’s a low frequency region.

The main aim of image sharpening is to highlight fine details in the image.

Image sharpening is used to enhance the high-frequency components.

Highpass Filtering

The spatial mask which performs image sharpening is given as $\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Highpass Filtering

Sum of coefficients of high pass mask has to be zero to eliminate the low frequency regions.

Example of High Pass Filtering

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	150	100

High pass Mask $\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-270	-270	-270	-270	-270	-270	-270	-270
270	270	270	270	270	270	270	270
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1) Pixel values cannot be negative. They always start with zero. Consider all negative values as zero.

2) The values of the original image at the edge are very large and there is a tendency of them going out of range due to center weight of +8. 1/9 is simply scaling factor.

final result using mask & negative values zero.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
30	30	30	30	30	30	30	30
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

High-boost Filtering

High-boost filter is also known as a high-frequency emphasis filter.

A high boost filter is an extension of the **high-pass filter**, but instead of completely removing the low-frequency components, it **retains some portion of the original image** while amplifying the high-frequency details.

In high-boost filtering, the input image f(m, n) is multiplied by an amplification factor A before subtracting the low-pass image.

$$\text{High-Boost Image} = A \cdot f(x, y) - \text{Low-Pass Filtered Image}$$

where:

- $f(x, y)$ is the original image
- A is the amplification factor (typically $A > 1$)
- **Low-Pass Filtered Image** is obtained using a smoothing filter like Gaussian or averaging filter.

High Boost Image = A(original image) – LPF image
= (A-1)original image + original image – LPF
= (A-1)original image + HPF image

Rewriting it in terms of the **High-Pass Filter (HPF)**:

$$\text{High-Boost Image} = (A - 1) \cdot f(x, y) + \text{High-Pass Filtered Image}$$

This means if **A = 1**, it behaves like a standard high-pass filter. If **A > 1**, the original image is included, making it a **high boost filter**.

A simple **high boost filter kernel** (for a 3×3 filter) can be:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & A & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

where **A > 1** controls the level of boosting.

Applications

1. **Image sharpening** – Enhances edges and details.
2. **Medical imaging** – Helps in better visualization of X-rays and MRIs.
3. **Remote sensing** – Enhances satellite images for better interpretation.
4. **Facial recognition and computer vision** – Improves feature detection.

Original Grayscale Image

High-Pass Filtered Image

High Boost Filtered Image

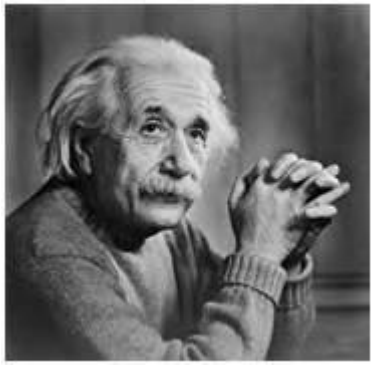
Adjusting the amplification factor **A** allows you to control the degree of sharpening applied to the image. Lower values result in gentler enhancement, while higher values can produce more pronounced sharpening effects.

High Boost Filtered Image (A=5)

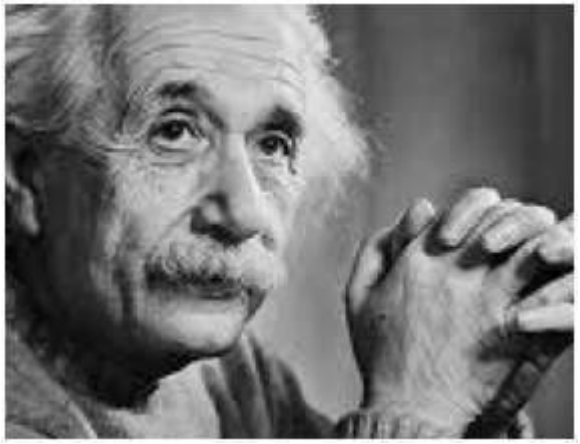
Zooming

Zooming simply means enlarging a picture in a sense that the details in the image became more visible and clear. Zooming an image has many wide applications ranging from zooming through a camera lens, to zoom an image on internet e.t.c.

For example



is zoomed into



Optical Zoom vs digital Zoom

Optical Zoom:

The optical zoom is achieved using the movement of the lens of your camera.

The result of the optical zoom is far better then that of digital zoom.

In optical zoom, an image is magnified by the lens in such a way that the objects in the image appear to be closer to the camera.

In optical zoom the lens is physically extended to zoom or magnify an object.

Optical Zoom vs digital Zoom

Digital Zoom:

Digital zoom is basically image processing within a camera.

During a digital zoom, the pixels got expand , due to which the quality of the image is compromised.

The same effect of digital zoom can be seen after the image is taken through your computer by using an image processing toolbox / software, such as Photoshop.

Image Zooming

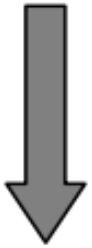
The zoom process can be done in numerous ways:

1. Zero Order Hold.
2. First Order Hold.
3. Convolution.

Zero-Order hold: is performed by repeating previous pixel values, thus creating a blocky effect.

Example: if we have an image of size(n*n), we can zooming it using zero order method with size (2n)*(2n)

$$\begin{bmatrix} f(x,y) & f(x,y+1) \\ f(x+1,y) & f(x+1,y+1) \end{bmatrix}_{2\times 2}$$



$$\begin{bmatrix} f(x,y) & f(x,y) & f(x,y+1) & f(x,y+1) \\ f(x,y) & f(x,y) & f(x,y+1) & f(x,y+1) \\ f(x+1,y) & f(x+1,y) & f(x+1,y+1) & f(x+1,y+1) \\ f(x+1,y) & f(x+1,y) & f(x+1,y+1) & f(x+1,y+1) \end{bmatrix}_{4\times 4}$$

a-Original image

b- cropped image

c- zoomed imag

First _Order Hold: is performed by finding linear interpolation between a adjacent pixels, i.e., finding the **average** value between two pixels and use that as the pixel value between those two, we can do this for the rows first as follows:

$$\begin{bmatrix} 8 & 4 & 8 \\ 4 & 8 & 4 \\ 8 & 2 & 8 \end{bmatrix} \longrightarrow \begin{bmatrix} 8 & 6 & 4 & 6 & 8 \\ 4 & 6 & 8 & 6 & 4 \\ 8 & 5 & 2 & 5 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 8 & 6 & 4 & 6 & 8 \\ 6 & 6 & 6 & 6 & 6 \\ 4 & 6 & 8 & 6 & 4 \\ 6 & 5.5 & 5 & 5.5 & 6 \\ 8 & 5 & 2 & 5 & 8 \end{bmatrix}$$

This method allows us to enlarge an N×N sized image to a size of (2N-1) ×(2N-1) and be repeated as desired.

Original image

cropped image

zooming image

The Convolution Process:

this method requires a mathematical process to enlarge an image. This method required **two** steps:

1. **Extend the image by adding rows and columns of zeros**

between the existing rows and columns. The image is extended as follows:

Original Image Array

$$\begin{pmatrix} 3 & 5 & 7 \\ 2 & 7 & 6 \\ 3 & 4 & 9 \end{pmatrix}$$

Image extended with zeros

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 7 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 4 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2. Perform the convolution.

Next, we use convolution mask, which is slide a cross the extended image, and perform simple arithmetic operation at each pixel location .

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

For example, if we put the mask over the upper-left corner of the image, we obtain (from right to left, and top to bottom):

$1/4(0) +1/2(0) +1/4(0) +1/2(0) +1(3) +1/2(0) + 1/4(0) +1/2(0) +1/4(0) =3$

$1/4(0) +1/2(0) +1/4(0) +1/2(3) +1(0) +1/2(5) + 1/4(0) +1/2(0) +1/4(0) =4$

Why we use this convolution method when it require, so many more calculation than the basic averaging of the neighbors method?

The answer :is that many computer boards can perform convolution in hardware, which is generally very fast, typically much faster than applying a faster algorithm in software.

Not only first-order hold be performed via convolution, but zeroorder hold can also achieved by extending the image with zeros and using the following convolution mask.

Zero-order hold convolution mask

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Note that for this mask we will need to put the result in the pixel location corresponding to the lower-right corner because there is no center pixel.

Note: These methods will only allows us to enlarge an image by a factor of (2N-1), but what if we want to enlarge an image by something other than a factor of (2N-1)?

To do this we need to apply a more general method. We take two adjacent values and linearly interpolate more than one value between them.

This is done by defining an enlargement number k and then following this process:

1. Get the difference between two adjacent values.
2. Divide the result by k.
3. Add the result to the smaller value, and keep adding the result from the second step in a running total until all (k-1) intermediate pixel locations are filled.

Example: we want to enlarge an image to three times its original size, and we have two adjacent pixel values 125 and 140.

1. Find the difference between the two values, 140-125 =15.
2. The desired enlargement is k=3, so we get 15/3=5.
3. next determine how many intermediate pixel values .we need :

   K-1=3-1=2. The two pixel values between the 125 and 140 are

   125+5=130 and 125+2*5 = 135.

- We do this for every pair of adjacent pixels .first along the rows and then along the columns. This will allows us to enlarge the image by any factor of K (N-1) +1 where K is an integer and N×N is the image size.

Histogram Modelling

Histogram of images provides global description of the appearance of an image.

Histogram of an image represents the relative frequency of occurrence of the various grey levels in an image.

Histogram Modelling

Two methods of plotting histogram.

Method 1 :

Grey level	Number of pixels (n_k)
0	40
1	20
2	10
3	15
4	10
5	3
6	2

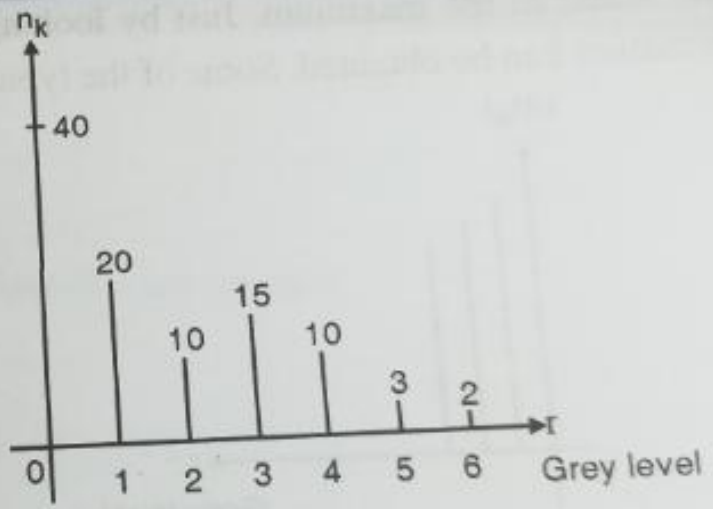


Fig. 8.1

Histogram Modelling

Grey level	Number of pixels (n_k)	$p(r_k) = \frac{n_k}{n}$
0	40	0.4
1	20	0.2
2	10	0.1
3	15	0.15
4	10	0.1
5	3	0.03
6	2	0.02
	n = 100	

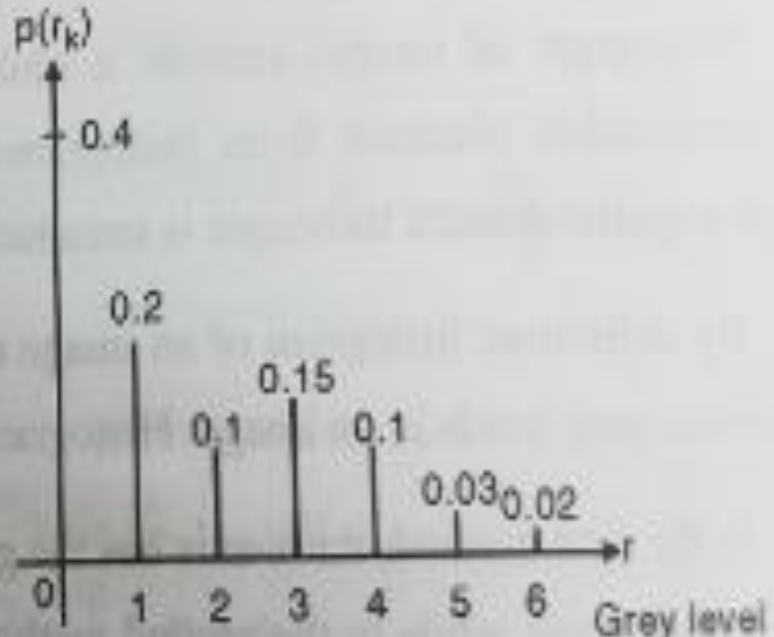


Fig. 8.2

Histogram Modelling

Histogram plotted in the second method is normalised histogram.

Maximum value to be plotted will be 1.

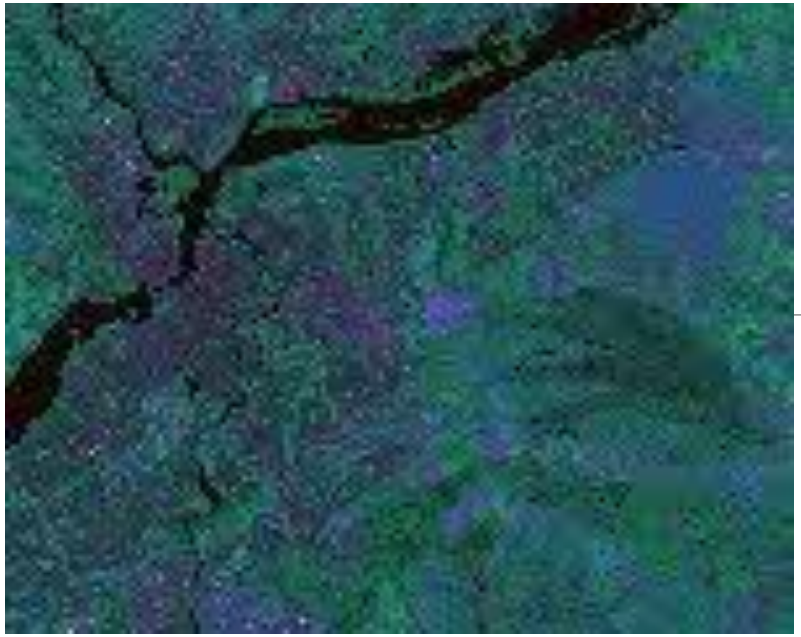
Normalised histogram provides clear information about the image.

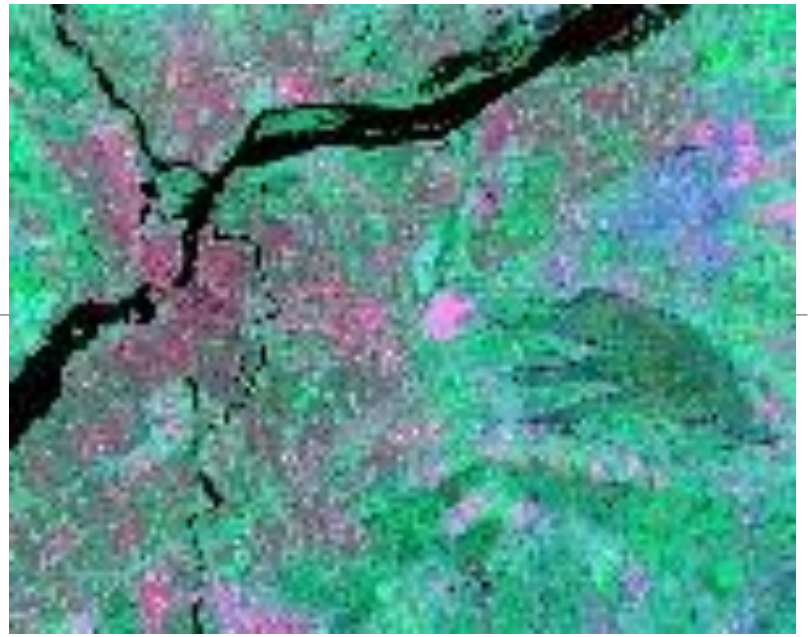
Linear Stretching

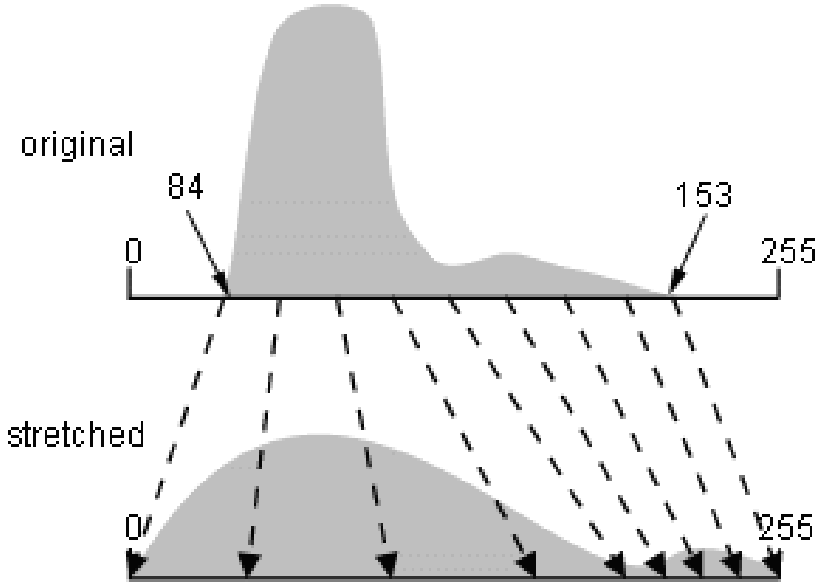
Histogram stretching is one way to increase dynamic range of the image.

Here the basic shape of the histogram is not altered , instead the histogram is spread to cover the entire dynamic range.

This enhances the contrast in the image with light toned areas appearing lighter and dark areas appearing darker, making visual interpretation much easier.







$$S = T(r) = \frac{s_{max} - s_{min}}{r_{max} - r_{min}}(r - r_{min}) + s_{min}$$

s_{max}= Maximum grey level of output image

s_{min}= Minimum grey level of output image

r_{max}= Maximum grey level of input image

r_{min}= Minimum grey level of input image

Histogram Equalization

Equalisation is a process that attempts to spread out the gray levels in an image so that they are evenly distributed across their range.

Histogram equalisation reassigns the brightness values of pixels based on the image histogram.

Histogram equalisation is a technique where the histogram of the resultant image is as flat as possible.

Histogram Equalization

Histogram equalisation provides more visually pleasing results across a wider range of images.

(c) *Procedure to Perform Histogram Equalisation*

Histogram equalisation is done by performing the following steps:

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

Example 5.1 *Perform histogram equalisation of the image*

$$\begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 4 & 5 & 4 & 3 \\ 3 & 5 & 5 & 5 & 3 \\ 3 & 4 & 5 & 4 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}.$$

Solution The maximum value is found to be 5. We need a minimum of 3 bits to represent the number. There are eight possible gray levels from 0 to 7. The histogram of the input image is given below:

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0

Step 1 Compute the running sum of histogram values.

The running sum of histogram values is otherwise known as cumulative frequency distribution.

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running sum	0	0	0	6	20	25	25	25

Step 2 Divide the running sum obtained in Step 1 by the total number of pixels. In this case, the total number of pixels is 25.

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running sum	0	0	0	6	20	25	25	25
Running Sum/Total number of pixels	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25

Step 3 Multiply the result obtained in Step 2 by the maximum gray-level value, which is 7 in this case.

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running Sum	0	0	0	6	20	25	25	25
Running sum / Total number of pixels	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25
Multiply the above result by maximum gray level	$\frac{0}{25}*7$	$\frac{0}{25}*7$	$\frac{0}{25}*7$	$\frac{6}{25}*7$	$\frac{20}{25}*7$	$\frac{25}{25}*7$	$\frac{25}{25}*7$	$\frac{25}{25}*7$

The result is then rounded to the closest integer to get the following table:

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running Sum	0	0	0	6	20	25	25	25
Running Sum/Total number of pixels	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25
Multiply the above result by maximum gray level	0	0	0	2	6	7	7	7

Step 4 Mapping of gray level by a one-to-one correspondence:

Original gray level	Histogram equalised values
0	0
1	0
2	0
3	2
4	6
5	7
6	7
7	7

The original image and the histogram equalised image are shown side by side.

$$\begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 4 & 5 & 4 & 3 \\ 3 & 5 & 5 & 5 & 3 \\ 3 & 4 & 5 & 4 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix} \xrightarrow{\text{Histogram Equalisation}} \begin{bmatrix} 6 & 6 & 6 & 6 & 6 \\ 2 & 6 & 7 & 6 & 2 \\ 2 & 7 & 7 & 7 & 2 \\ 2 & 6 & 7 & 6 & 2 \\ 6 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Original image **Histogram equalised image**

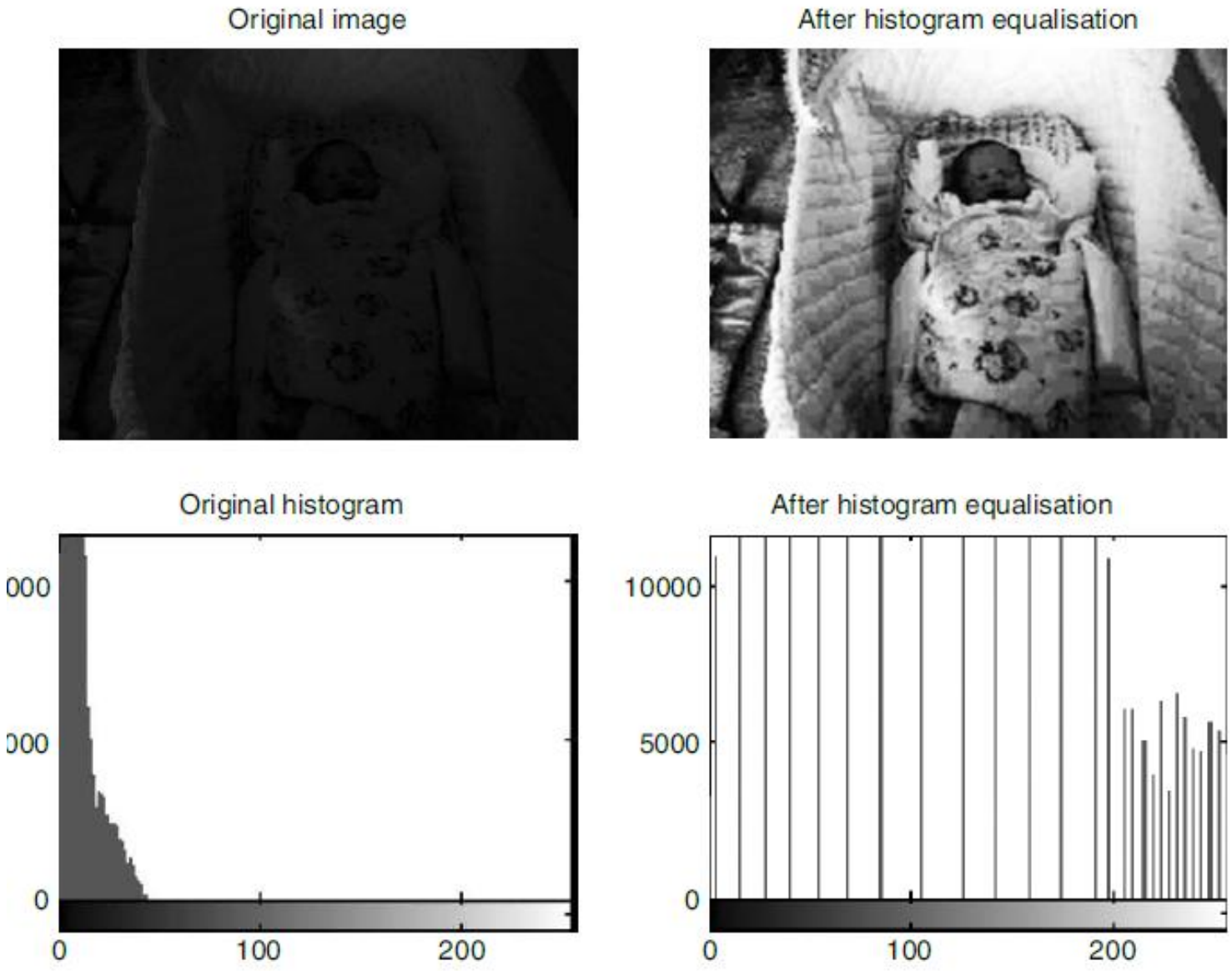


Fig. 5.11 *Results of histogram equalisation*