Marwadi
University

IMAGE PROCESSING
01CE0507

Unit - 3
Image Enhancement

Prof. Urvi Y. Bhatt
Department of Computer Engineering

Outline

- What is Image Enhancement?
- Image Enhancement Techniques
  - Spatial Domain Methods
    - Intensity (Gray-level) transformations functions
    - Histogram Processing
    - Spatial Filtering
  - Frequency Domain Methods

Image Enhancement

- Image enhancement refers to the process of highlighting certain information of an image, as well as weakening or removing any unnecessary information according to specific needs.
- For example, eliminating noise, revealing blurred details, and adjusting levels to highlight features of an image.

Image Enhancement (Cont.)

- It is to process an image so that the result is more suitable than the original image for a specific application.
- The idea behind enhancement techniques is to **bring out details that are hidden, or simple to highlight certain features of interest** in an image.

Image Enhancement Techniques

- Image enhancement techniques can be divided into two broad categories:
  - **Spatial Domain**
  - **Frequency Domain**

Spatial Domain

- It enhancement of the image space that divides an image into uniform pixels according to the spatial coordinates with a particular resolution.
- The spatial domain methods perform operations on pixels directly.

Frequency Domain

- It enhancement obtained by applying the Fourier Transform to the spatial domain.
- In the frequency domain, pixels are operated in groups as well as indirectly.

Spatial Domain Technique

- **Intensity Transformation Techniques / Point Operation**
  - Point operations refer to running the same conversion operation for each pixel in a grayscale image.
  - The transformation is based on the original pixel and is independent of its location or neighboring pixels.
- **Spatial Filtering**
  - The output value depends on the values of f(x,y) and its neighborhood.

Intensity Transformation Techniques / Point Operation

- Point operations are often used to change the grayscale range and distribution.
- The concept of point operation is to map every pixel onto a new image with a predefined transformation function.

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

Where,

- $g(x, y)$ is the output image
- T is an operator of intensity transformation
- $f(x, y)$ is the input image

Intensity Transformation Techniques

Intensity Transformation Functions Fall Into 2 Approaches:

- Basic Intensity Transformation / Grey Level Transformation
- Piecewise Linear Transformation

Basic Intensity Transformations

- The simplest image enhancement method is to use a 1 x 1 neighborhood size. It is a point operation.
- In this case, the output pixel ('s') only depends on the input pixel ('r'), and the point operation function can be simplified as follows:

$$s = T(r)$$

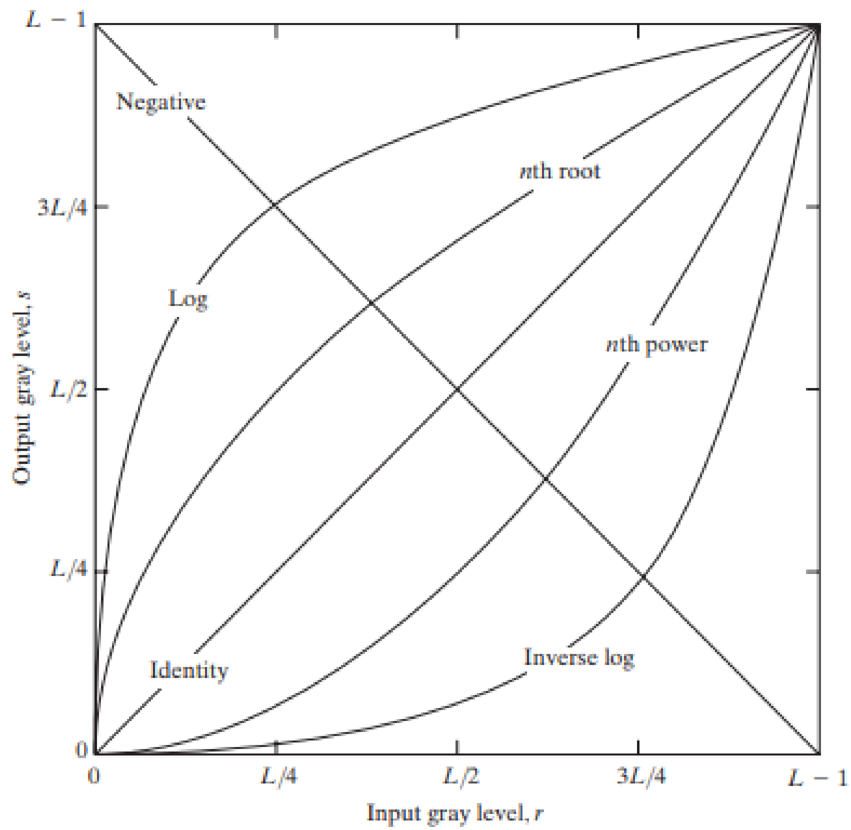
- Where
  - T is the point operator of a certain gray-level mapping relationship between the original image and the output image.
  - s,r: denote the gray level of the input pixel and the output pixel.

Basic Intensity Transformations

- Linear Functions
  - Identity Transformation
  - Negative Transformation
- Logarithmic Functions
  - Log Transformation
  - Inverse-log Transformation
- Power-law / Gamma / Exponential Functions
  - Nth Power Transformation
  - Nth Root Transformation

Basic Intensity Transformations

- Different transformation functions work for different scenarios.



Identity Transformation

- Output intensities are identical to input intensities
- This function doesn't have an effect on an image, it was included in the graph only for completeness
- In identity transformation, the input image is the same as the output image.

$$s = r$$

Negative Transformation

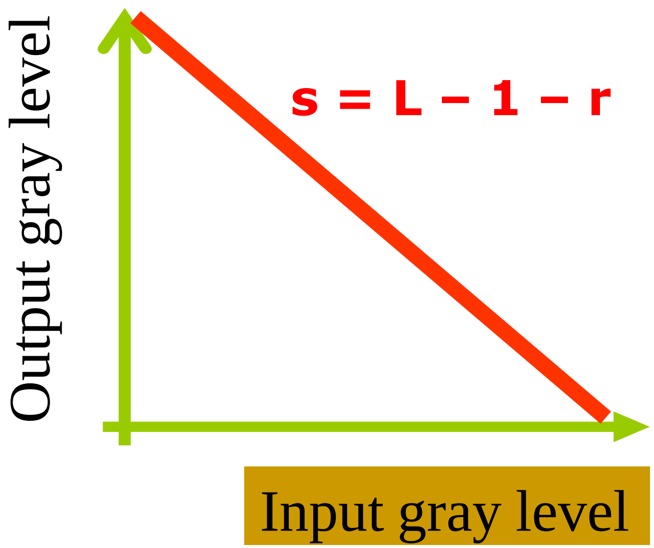
- The negative of an image with gray level in the range [0, L-1], where L = Largest value in an image, is obtained by using the negative transformation's expression:

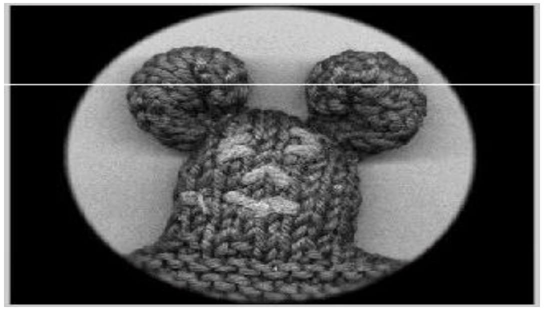
$$s = L - 1 - r$$

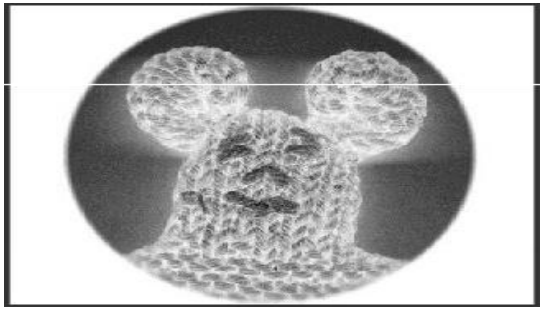
- Negative transformation reverses the intensity levels of an input image , in this manner produces the equivalent of a photographic negative.

Negative Transformation (Cont.)

- The negative transformation is suitable for enhancing white or gray detail embedded in dark regions of an image, especially when the black area are dominant in size

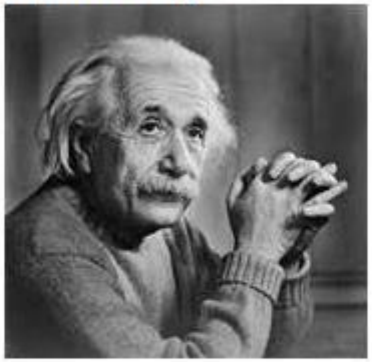






Negative Transformation (Cont.)

Input Image



Output Image

Advantages of negative :

✓ Produces an equivalent of a photographic negative.

✓ Enhances white or gray detail embedded in dark regions.

Negative Transformation (Cont.)

- Example 1: The following matrix represents the pixels values of an 8-bit image (r) , apply negative transform and find the resulting image pixel values.

Image (r)

100	110	90	95
98	140	145	135
89	90	88	85
102	105	99	115

Solution:

$L = 2^8 = 256$

$s = L - 1 - r$

$s = 255 - r$

Apply this transform to each pixel to find the negative

Image (s)

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

Negative Transformation (Cont.)

- Example 2: the following matrix represents the pixels values of a 5-bit image (r) , apply negative transform and find the resulting image pixel values.

Image (r)

21	26	29	30
19	21	20	30
16	16	26	31
19	18	27	23

Solution:

$L= 2^5 = 32$

$s=L-1-r$

$s =31-r$

Apply this transform to each pixel to find the negative

Image (s)

10	5	2	1
12	10	11	1
15	15	5	0
12	13	4	9

Negative Transformation (Cont.)

- The negative of an image can be obtained also with Image Processing Toolkit function imcomplement:

g = imcomplement (f);

Log Transformation

- The equation of general log transformation is:

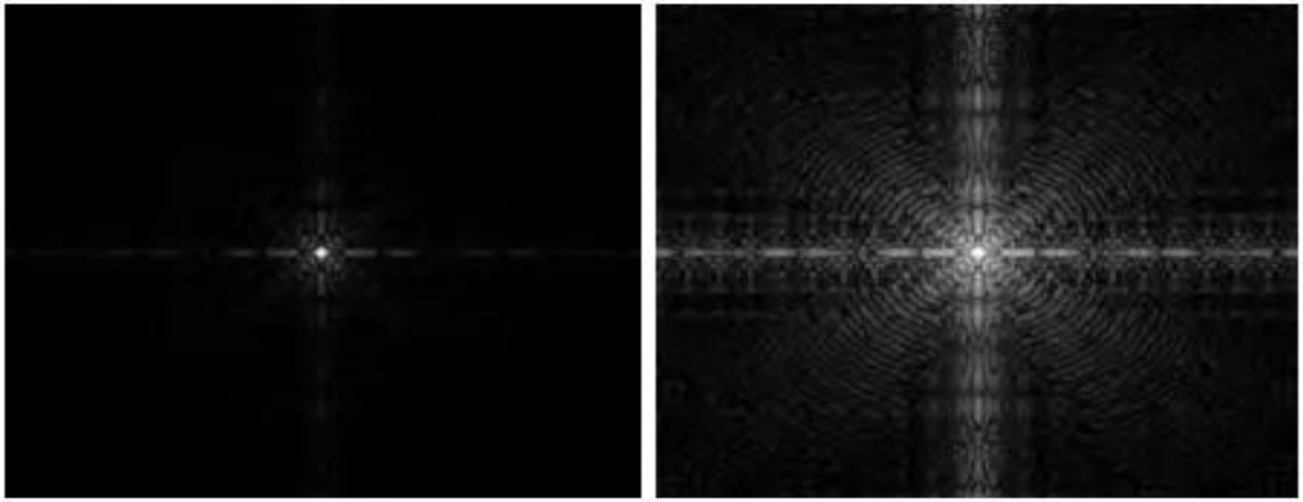
$$s = c * \log(1 + r)$$

- Note:
  - s,r: denote the gray level of the input pixel and the output pixel.
  - ‘c’ is a constant; to map from [0,255] to [0,255], c = 255/LOG(256)
  - the base of a common logarithm is 10

Log Transformation (Cont.)

- In the log transformation, the low-intensity values are mapped into higher intensity values.
- It maps a narrow range of low gray levels to a much wider range.
- The log transformation works the best for dark images.
- It compresses the dynamic range of images with large variations in pixel values.
- Log functions are particularly useful when the input grey level values may have an extremely large range of values

Log Transformation (Cont.)



Log Transformation (Cont.)

- Example 1: The following matrix represents the pixels values of an 8-bit image (r), apply Log transform and find the resulting image pixel values.

(i) C = 1;

(ii) C = L / Log(1+L)

110	120	90
91	94	98
90	91	99

Log Transformation (Cont.)

- Logarithmic transformations are implemented using expression:

$$g = c * \log(1 + \text{double}(f))$$

- But this function changes the data class of the image to double, so another sentence to return it back to uint8 should be done:

- implemented expression:

$$gs = \text{im2uint8}(\text{mat2gray}(g));$$

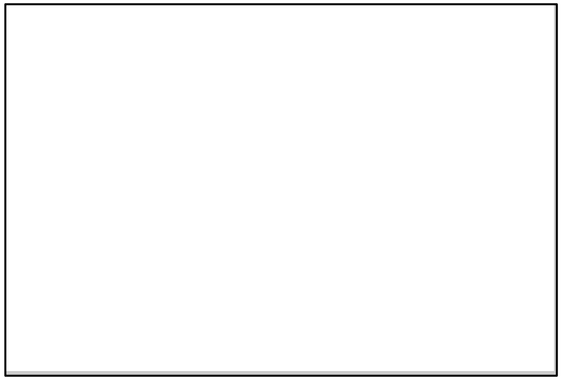
- Use of mat2gray brings the values to the range [0 1] and im2uint8 brings them to the range [0 255]

Marwadi University

Log Transformation (Cont.)

Example: 1

```
g = log(1 + double(f));
gs = im2uint8(mat2gray(g));
imshow(f)
figure
imshow (g)
Figure
imshow(gs);
```



g

f

gs

Inverse-log Transformation

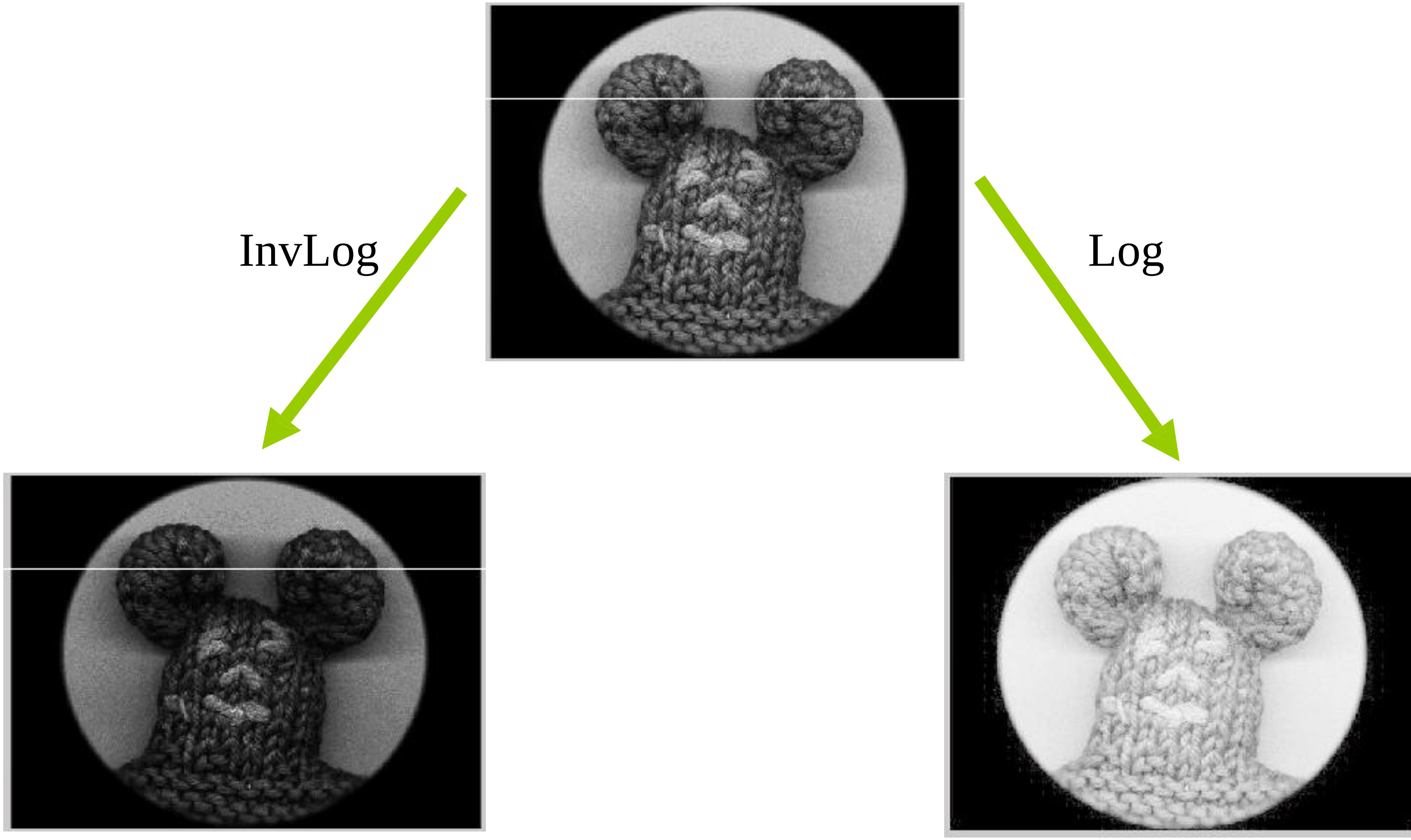
- The inverse log transform is opposite to log transform.
- The inverse log transform expands the values of light-level pixels while compressing the darker-level values.

$$s = power(10, r * c)-1$$

- Note:
  - s,r: denote the gray level of the input pixel and the output pixel.
  - ‘c’ is a constant; to map from [0,255] to [0,255], c =LOG(256)/255

Inverse-log Transformation (Cont.)

- Used to expand the values of high pixels in an image while compressing the darker-level values.
- It maps a narrow range of high gray levels to a much wider range.


InvLog
Log

Power-law / Gamma / Exponential Transformation

- Power-law(Gamma) transformations have the basic form of:

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

Where c and $^{\gamma}$ are positive constants

- Variation in the value of γ varies the enhancement of the images. Different display devices / monitors have their own gamma correction, that's why they display their image at different intensity.

Power-law / Gamma / Exponential Transformation (Cont.)

Marwadi University

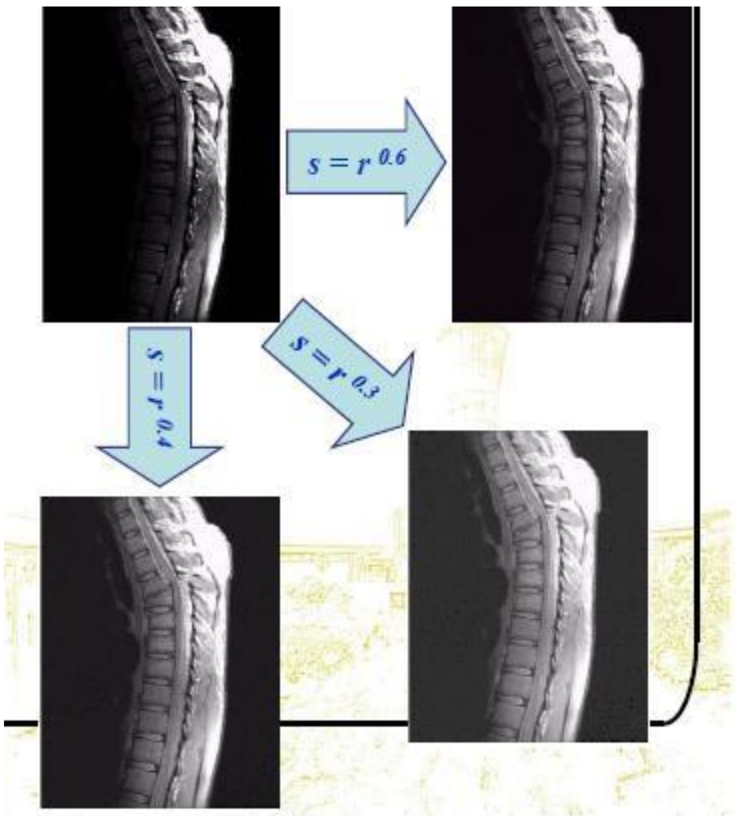
- This type of transformation is used for enhancing images for different type of display devices.
- Map a narrow range of dark input values into a wider range of output values or vice versa
- The gamma of different display devices is different.
- For example Gamma of CRT lies in between of 1.8 to 2.5, that means the image displayed on CRT is dark.

Power-law / Gamma / Exponential Transformation (Cont.)

- This type of transformation is used for enhancing images for different type of display devices.
- The gamma of different display devices is different.
- For example Gamma of CRT lies in between of 1.8 to 2.5, that means the image displayed on CRT is dark.

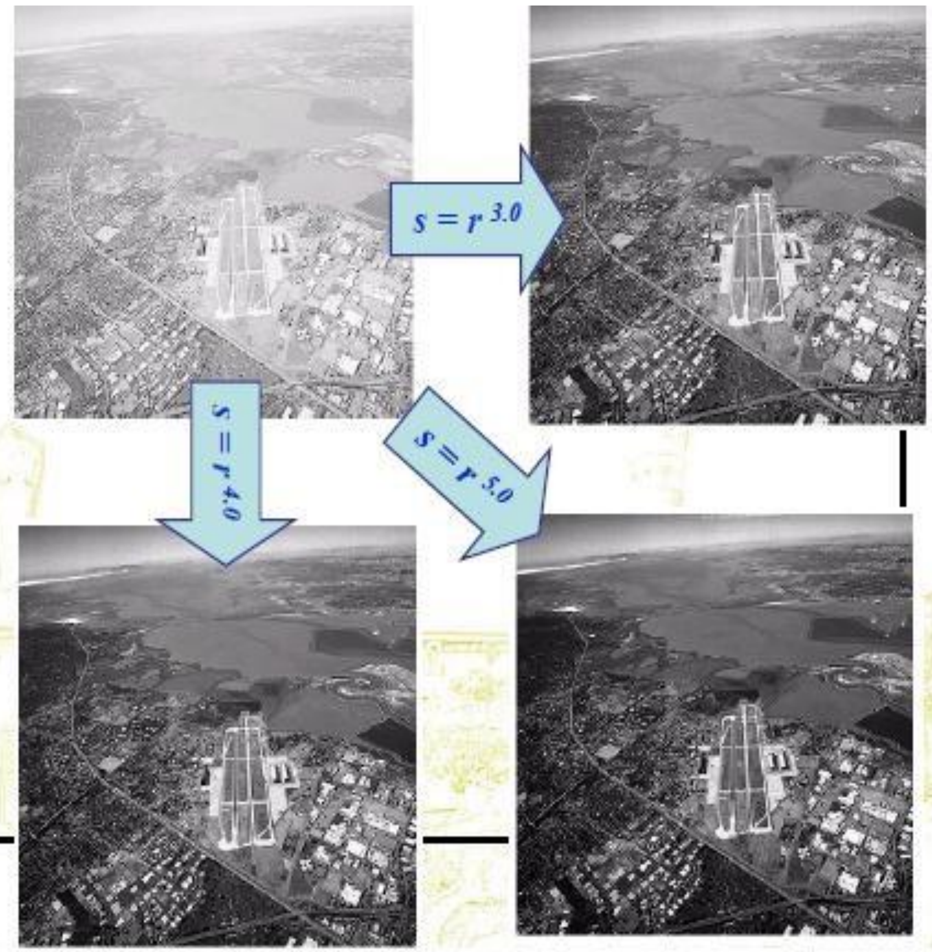
Power-law / Gamma / Exponential Transformation (Cont.)

- The images to the right show a magnetic resonance (MR) image of a fractured human spine
- Different curves highlight different detail



Power-law / Gamma / Exponential Transformation (Cont.)

- An aerial photo of a runway is shown
- This time power law transforms are used to darken the image
- Different curves highlight different detail



Power-law / Gamma / Exponential Transformation (Cont.)

Gamma = 10

Gamma = 8

Gamma = 6

Power-law / Gamma / Exponential Transformation (Cont.)

- Different transformation curves are obtained by varying ᵞ (gamma)

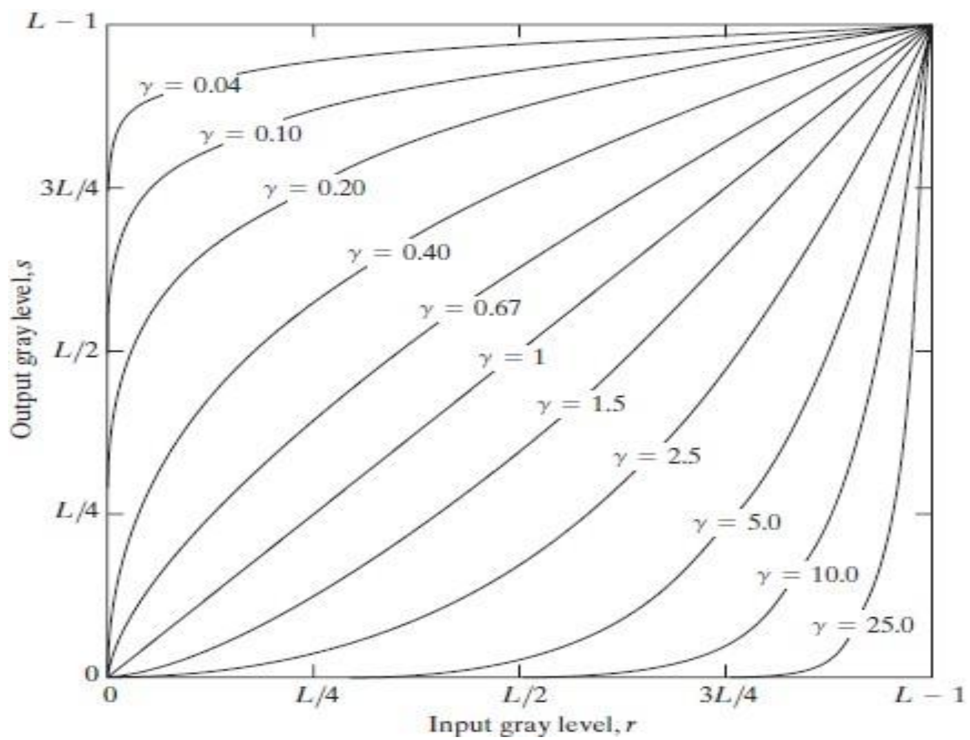


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

Power-law / Gamma / Exponential Transformation (Cont.)

- If gamma <1 :the mapping is weighted toward brighter output values.
- If gamma =1 (default):the mapping is linear.
- If gamma >1 :the mapping is weighted toward darker output values.

Power-law / Gamma / Exponential Transformation (Cont.)

- Example 1: The following matrix represents the pixels values of an 8-bit image (r) , apply Power – Law transform and find the resulting image pixel values.

(i) C = 1; Gamma = 0.2

110	120	90
91	94	98
90	91	99

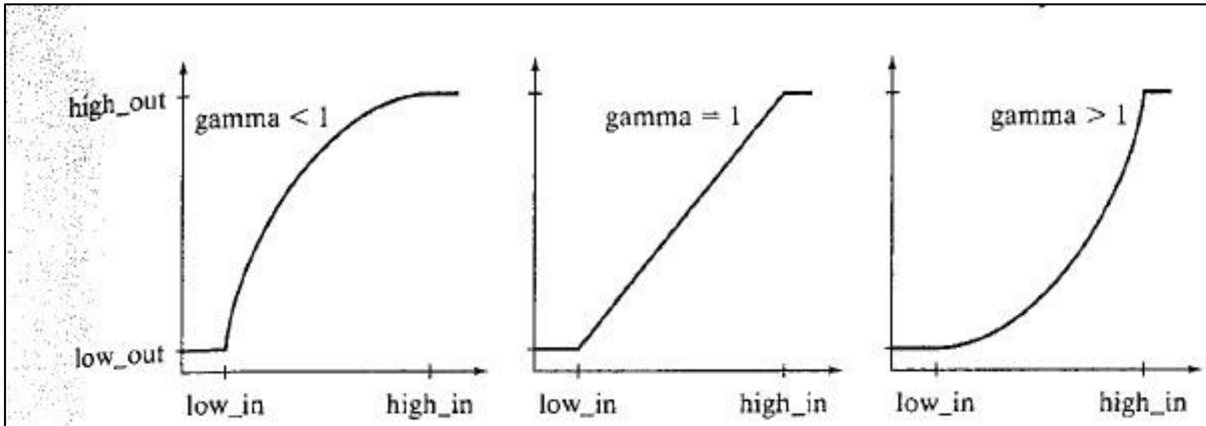
Power-law / Gamma / Exponential Transformation (Cont.)

- Function **imadjust** is the basic IPT tool for intensity transformations of gray-scale images. It has the syntax:

g = imadjust (f, [low_in high_in], [low_out high_out], gamma)

Power-law / Gamma / Exponential Transformation (Cont.)

- As illustrated in figure, this function maps the intensity values in image f to new values in g, such that values between low_in and high_in map to values between low_out and high_out.
- Values below low_in and above high_in are clipped; that is values below low_in map to low_out, and those above high_in map to high_out.



a b c

FIGURE 3.2 The various mappings available in function `imadjust`.

Power-law / Gamma / Exponential Transformation (Cont.)

Example: 1

- f = imread ('baby-BW.jpg');
- g = imadjust (f, [0 1], [1 0]);
- imshow(f), figure, imshow (g);

f

g

Power-law / Gamma / Exponential Transformation (Cont.)

Example: 2

- f = imread ('baby-BW.jpg');
- g = imadjust (f, [0.5 0.75], [0 1], 0.5);
- imshow(f), figure, imshow (g);

f

g

Power-law / Gamma / Exponential Transformation (Cont.)

Example: 3

- f = imread ('baby-BW.jpg');
- g = imadjust (f, [0.5 0.75], [0.6 1], 0.5);
- imshow(f), figure, imshow (g);

f

g

Power-law / Gamma / Exponential Transformation (Cont.)

Example: 4

- f = imread ('baby-BW.jpg');
- g = imadjust (f, [ ], [ ], 2);
- imshow(f), figure, imshow (g);

f

g

Piecewise Linear Transformation

- Principle Advantage: Some important transformations can be formulated only as a piecewise function.
- Principle Disadvantage: Their specification requires more user input that previous transformations
- In mathematics, a piecewise-defined function is a function defined by multiple sub-functions, where each sub-function applies to a different interval in the domain.

Piecewise Linear Transformation (Cont.)

Types of Piecewise Linear Transformation Function:

- Contrast Stretching
- Thresholding / Grayscale Threshold Transform / Binarization
- Gray-level Slicing
- Bit-plane Slicing

Contrast Stretching

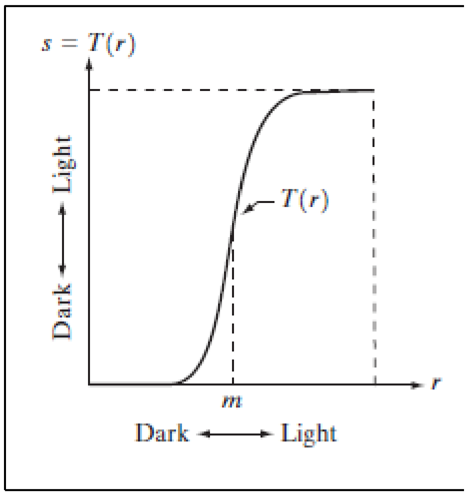
- One of the simplest piecewise linear functions is a contrast-stretching transformation, which is used to enhance the low contrast images.
- Low contrast images may result from:
  - Poor illumination
  - Wrong setting of lens aperture during image acquisition.

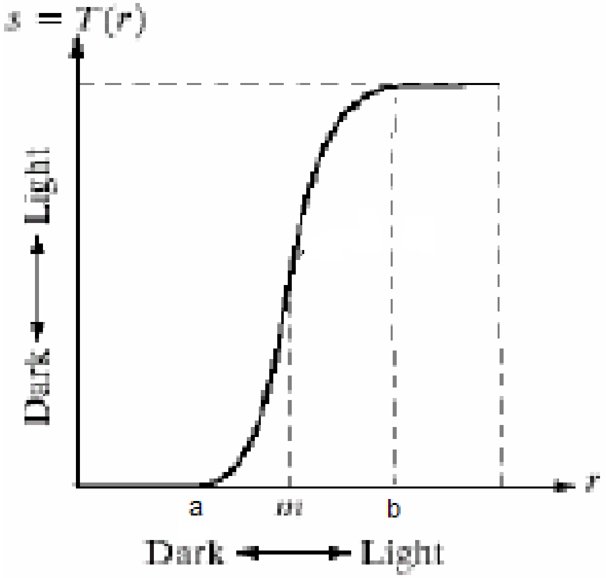
Contrast Stretching (Cont.)

- If T(r) has the form as shown in the figure below, the effect of applying the transformation to every pixel of f to generate the corresponding pixels in g would:
- Produce higher contrast than the original image, by:
  - – Darkening the levels below m in the original image
  - – Brightening the levels above m in the original image

Contrast Stretching (Cont.)

- So, Contrast Stretching: is a simple image enhancement technique that improves the contrast in an image by ‘stretching’ the range of intensity values it contains desired range of values.





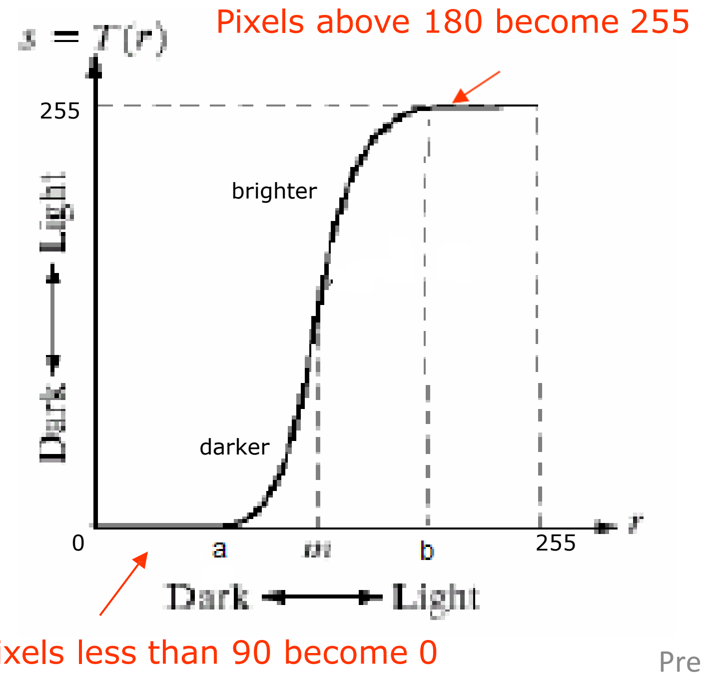
Contrast Stretching (Cont.)

Remember that:

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

Or

$$s = T(r)$$



Example: in the graph, suppose we have the following intensities : a=90, b=180, m=100

✓if r is above 180 ,it becomes 255 in s.

✓If r is below 90 , it becomes 0,

✓If r is between 90, 180 , T applies as follows:

when r < 100 , s closes to zero (darker)

when r>100 , s closes to 255 (brighter)

$$T = \begin{cases} \text{If } r > 180;\ s = 255 \\ \text{If } r < 180 \text{ and } r < 90;\ s = T(r) \\ \text{If } r < 90;\ s = 0 \end{cases}$$

This is called contrast stretching, which means that the bright pixels in the image will become brighter and the dark pixels will become darker, this means : Higher Contrast Image

Contrast Stretching (Cont.)

Image (r(

Image (s (after applying T (contrast stretching)

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

Contrast Stretching (Cont.)

- This equation is implemented in MATLAB for the entire image as

```
g = 1./(1 + (m./(double(f) + eps)).^E)
```

- Note the use of eps to prevent overflow if f has any 0 values.

Contrast Stretching (Cont.)

- Example1:
- >>g = 1 ./ (1+ (100 ./(double(f) + eps)) .^ 20);
- >> imshow(f), figure, imshow(g);

Contrast Stretching (Cont.)

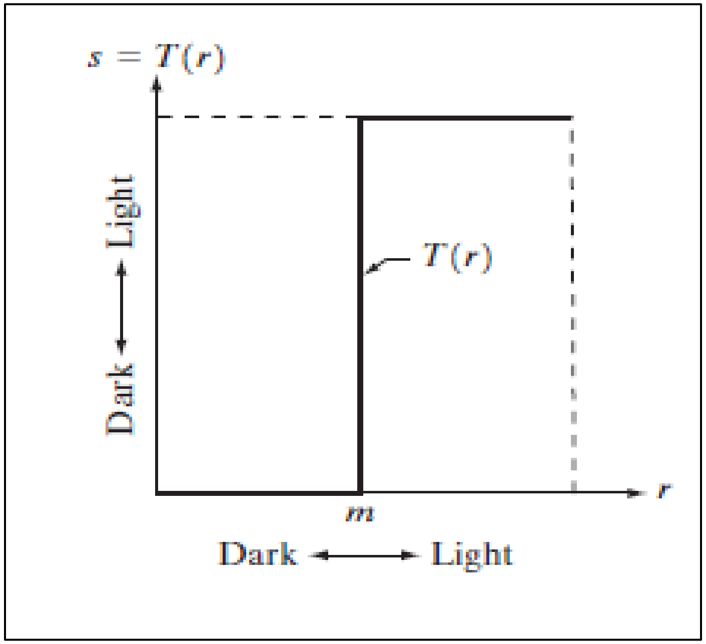
- Example 2:
- >>g = 1 ./ (1+ (50 ./(double(f) + eps)) .^ 20);
- >> imshow(f), figure, imshow(g);

Contrast Stretching (Cont.)

- Example 3:
- `>>g = 1 ./ (1+ (150 ./(double(f) + eps)) .^ 20);`
- `>> imshow(f), figure, imshow(g);`

Thresholding / Grayscale Threshold Transform / Binarization

- Is a limited case of contrast stretching, it produces a two- level (binary) image.



Thresholding / Grayscale Threshold Transform / Binarization (Cont.)

- Thresholding Transform converts a grayscale image into a black and white binary image.
- The user specifies a value that acts as a dividing line.
- If the gray value of a pixel is smaller than the dividing, the intensity of the pixel is set to 0, otherwise it's set to 255.
- The value of the dividing line is called the threshold.
- The grayscale threshold transform is often referred to as thresholding, or binarization.

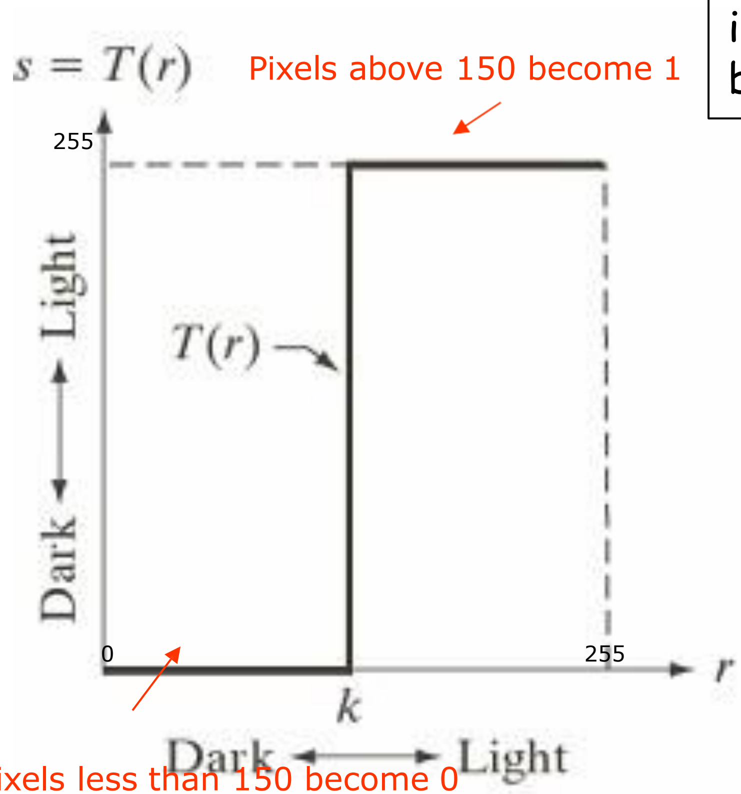
Thresholding / Grayscale Threshold Transform / Binarization (Cont.)

Remember that:

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

Or

$$s = T(r)$$



Example: suppose m= 150 (called threshold),

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

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

Or simply…

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

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

Thresholding / Grayscale Threshold Transform / Binarization (Cont.)

Image (r (

Image (s (after applying T (Thresholding)

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

Thresholding / Grayscale Threshold Transform / Binarization (Cont.)

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

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

Thresholding / Grayscale Threshold Transform / Binarization (Cont.)

Logical Code: Or im2bw() function

```
S=95
y=x;
[m n]=size(x);
for i=1:m
        for j=1:n
                if x(i,j)>= s
                        y(i,j)=255;
                else
                        y(i,j)=0;
                end
        end
end
figure, imshow(x);  figure, imshow(y);
```

Gray-level Slicing

- This technique is used to highlight a specific range of gray levels in a given image.
- Similar to thresholding Other levels can be suppressed or maintained
- Useful for highlighting features in an image
- It can be implemented in several ways, but the

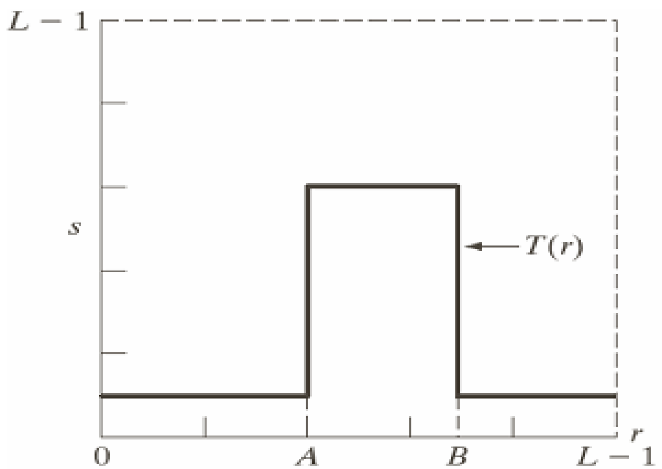
Gray-level Slicing (Cont.)

- Two basic Approach are:
  - One approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels.
  - The second approach, based on the transformation brightens the desired range of gray

Gray-level Slicing (Cont.)

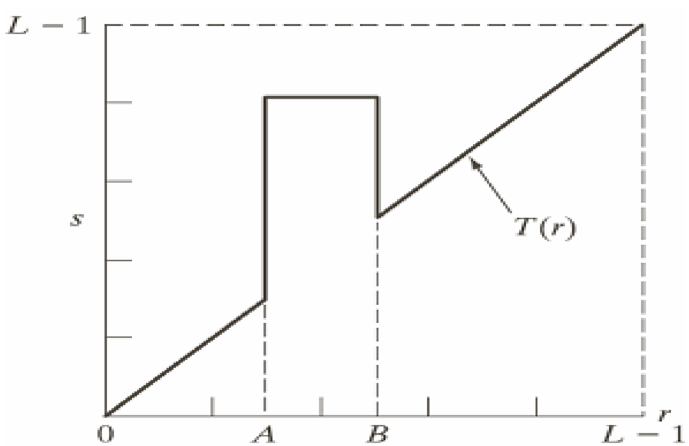
Highlighting a specific range of intensities in an image.

Approach 1



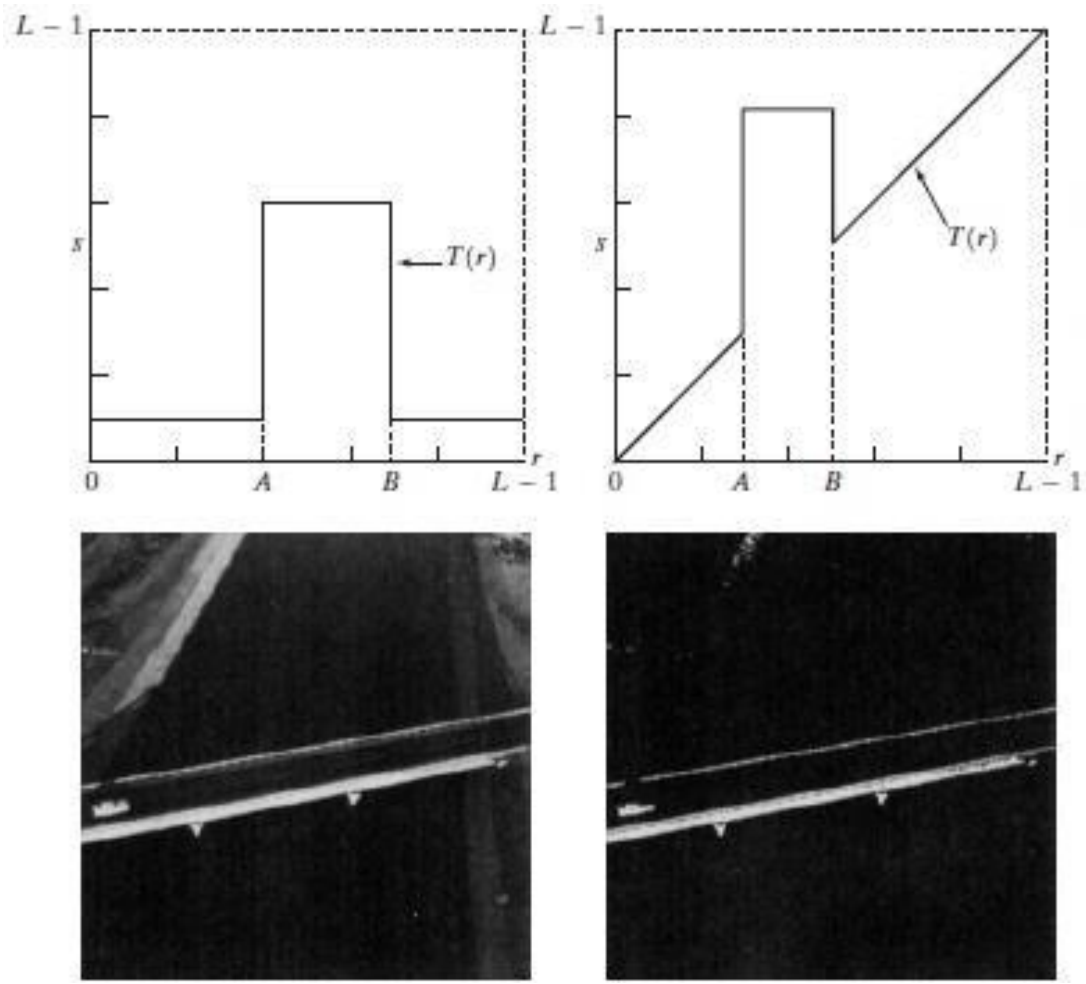
display in one value(e.g white) all the values in the range of interest , and in another (e.g black) all other intensities

Approach 2



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

Gray-level Slicing (Cont.)



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).

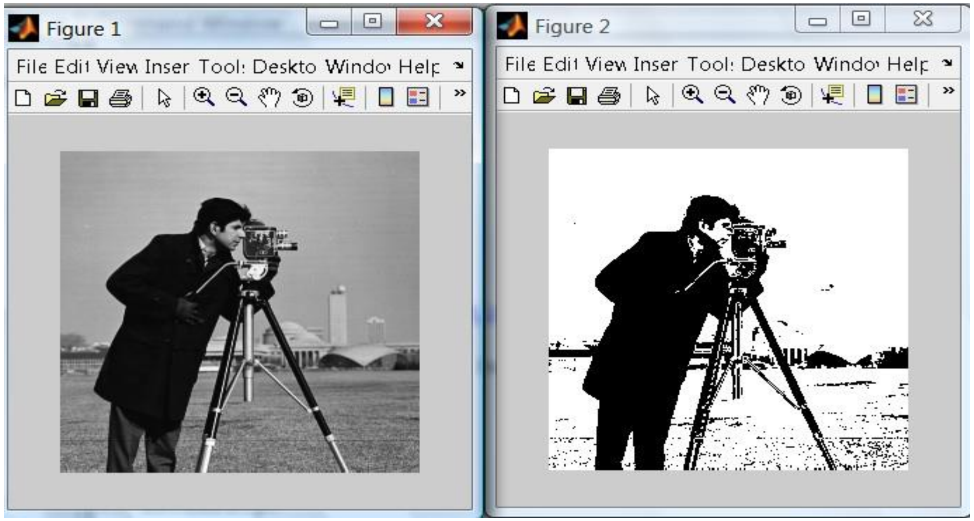
Gray-level Slicing (Cont.)

- Example 1 : Apply intensity level slicing in below image ,
  - Approach 1: then If the pixel intensity in the old image is between (100 - 200) convert it in the new image into 255 (white). Otherwise convert it to 0 (black).
  - Approach 2: then If the pixel intensity in the old image is between (100 - 200) convert it in the new image into 255 (white). Otherwise it leaves it the same.

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

Gray-level Slicing (Cont.)

- Example for Approach 1: apply intensity level slicing in Matlab to read cameraman image , then If the pixel intensity in the old image is between (100 - 200) convert it in the new image into 255 (white). Otherwise convert it to 0 (black).



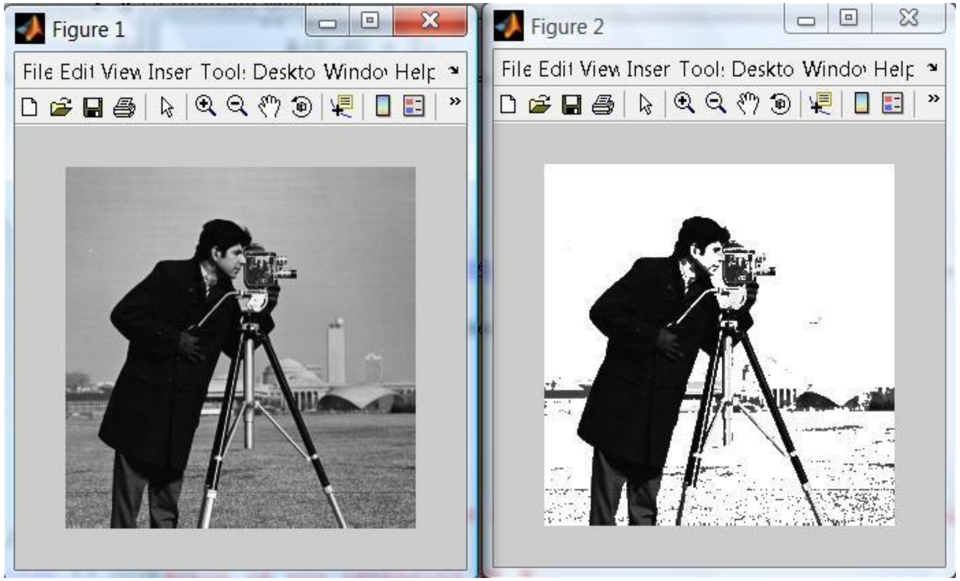
Gray-level Slicing (Cont.)

Solution:

```
x=imread('cameraman.tif');  y=x;
[w h]=size(x);
for i=1:w
        for j=1:h
                if x(i,j)>=100 && x(i,j)<=200
                        y(i,j)=255;
                else
                        y(i,j)=0;
                end
        end
end
figure, imshow(x);  figure, imshow(y);
```

Gray-level Slicing (Cont.)

- Example for Approach 2 : apply intensity level slicing in Matlab to read cameraman image , then If the pixel intensity in the old image is between (100 - 200) convert it in the new image into 255 (white). Otherwise it leaves it the same.



Gray-level Slicing (Cont.)

Solution:

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

Bit-Plane Slicing

- Pixels are digital numbers, each one composed of bits. Instead of highlighting gray-level range, we could highlight the contribution made by each bit.
- This method is useful and used in image compression.
- Most significant bits contain the majority of visually significant data

Bit-Plane Slicing (Cont.)

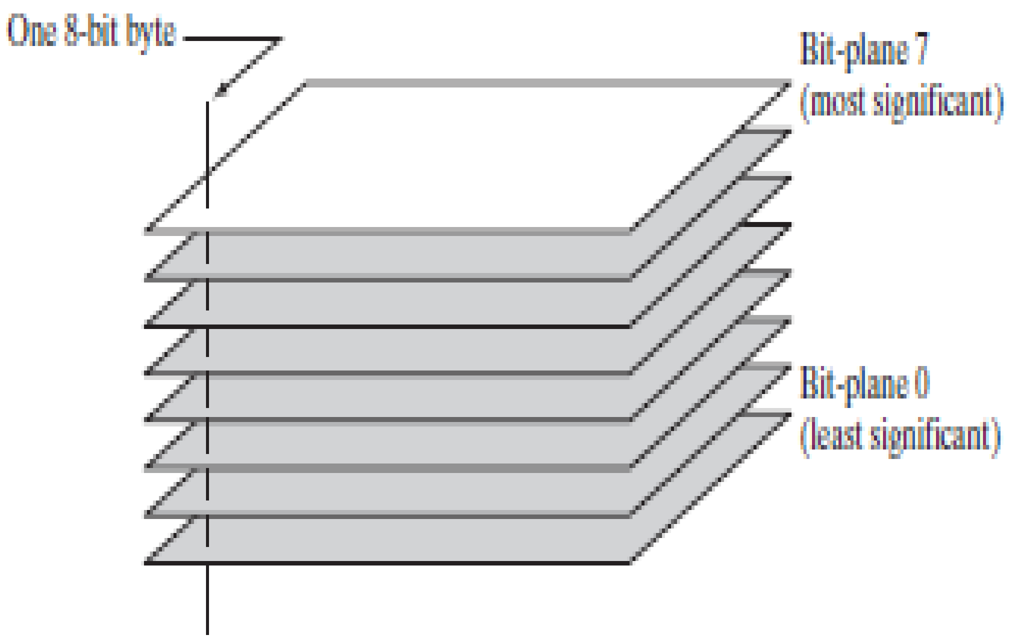


FIGURE 3.12 Bit-plane representation of an 8-bit image.

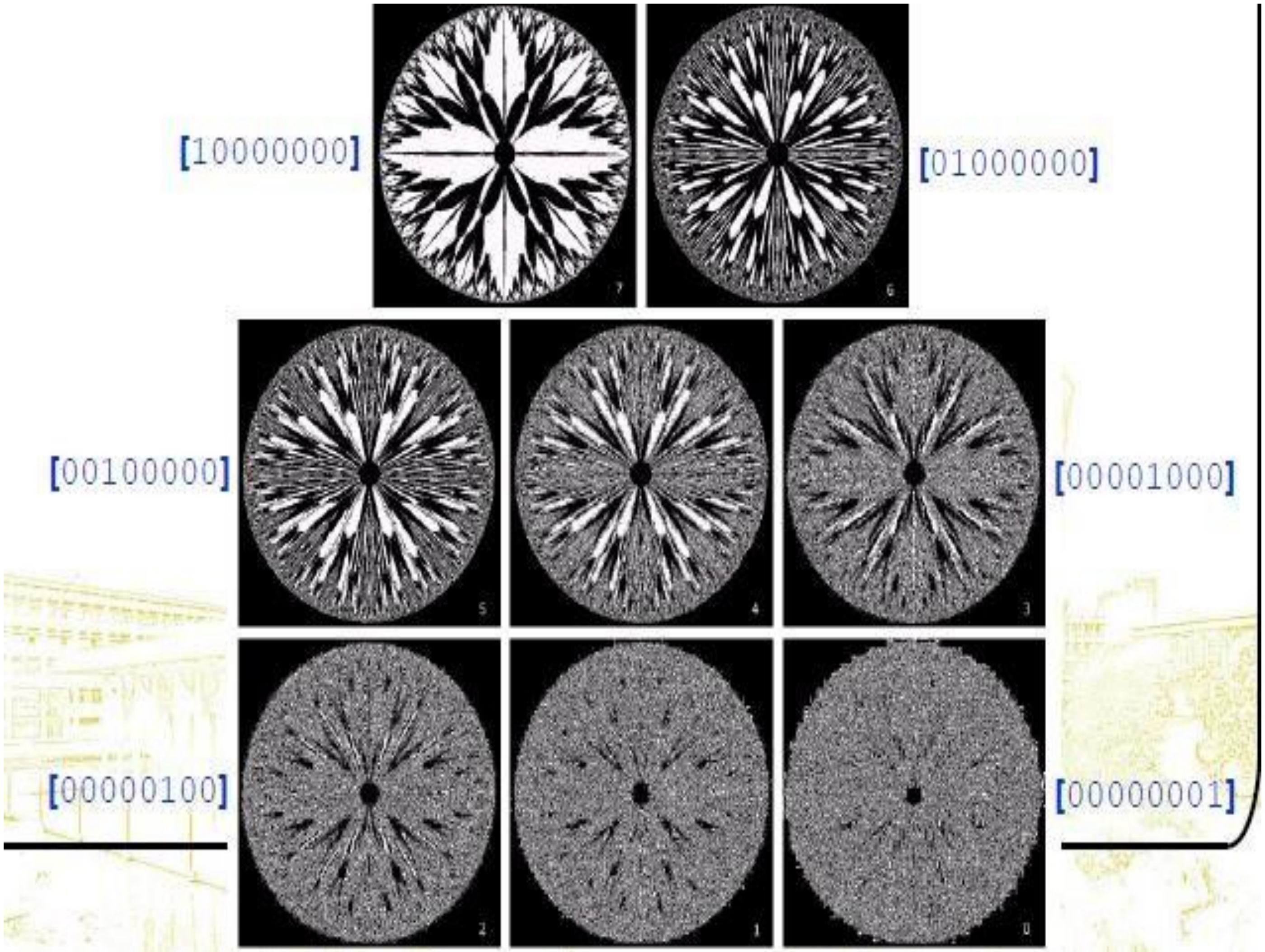
Remember that pixels are digital numbers composed of bits.

8-bit Image composed of 8 1-bit planes

Bit-Plane Slicing (Cont.)

- Often by isolating particular bits of the pixel values in an image we can highlight interesting aspects of that image
  - Higher-order bits usually contain most of the significant visual information
  - Lower-order bits contain subtle details

Bit-Plane Slicing (Cont.)



Bit-Plane Slicing (Cont.)

a	b	c
d	e	f
g	h	i

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

Bit-Plane Slicing (Cont.)

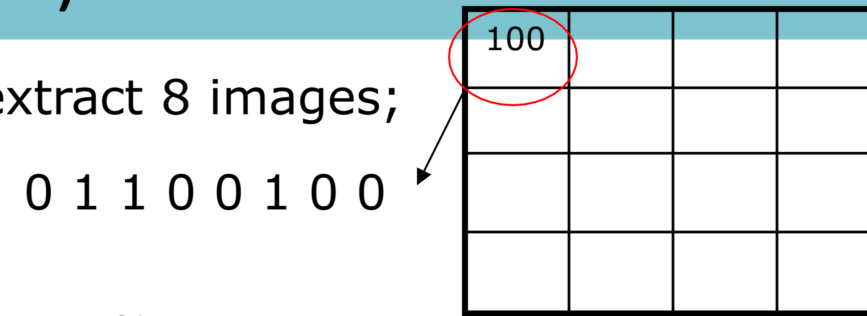
Reconstructed image using only bit planes 8 and 7

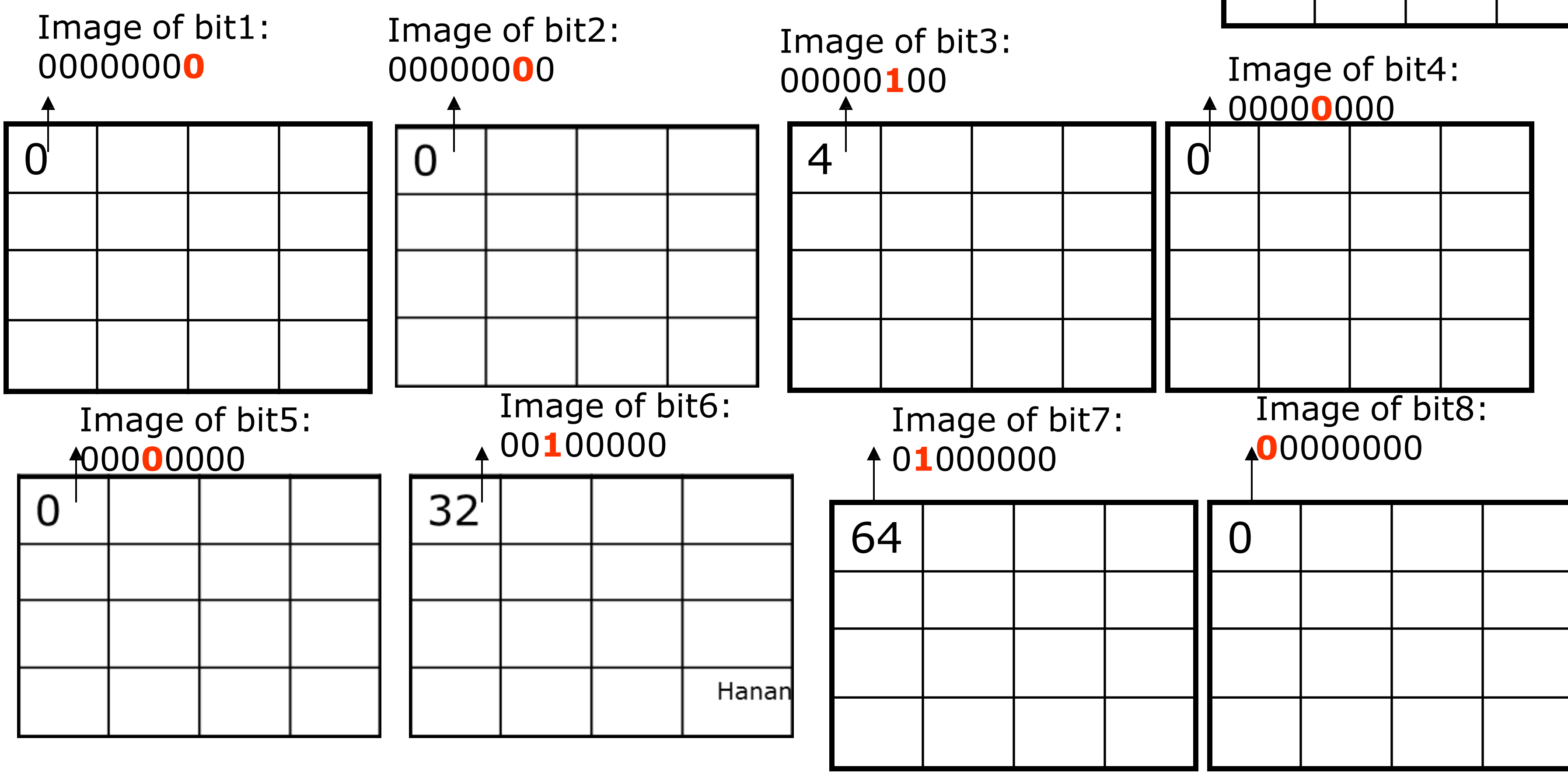
Reconstructed image using only bit planes 8, 7 and 6

Reconstructed image using only bit planes 7, 6 and 5

Bit-Plane Slicing (Cont.)

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





Bit-Plane Slicing (Cont.)

- Function to implement Bit-Plan Slicing:

```
b=bitget(x(i,j),6);
y(i,j)=bitset(y(i,j),6,b);
```

Bit-Plane Slicing (Cont.)

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

Solution:

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

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


Thank
you