CSE4227 Digital Image Processing

Lecture 04 – **Chapter 3: Intensity Transformations**

Dr. Kazi A Kalpoma

Professor, Department of CSE

Ahsanullah University of Science & Technology (AUST)

Contact: kalpoma@aust.edu

Google Class code: bux3jc2

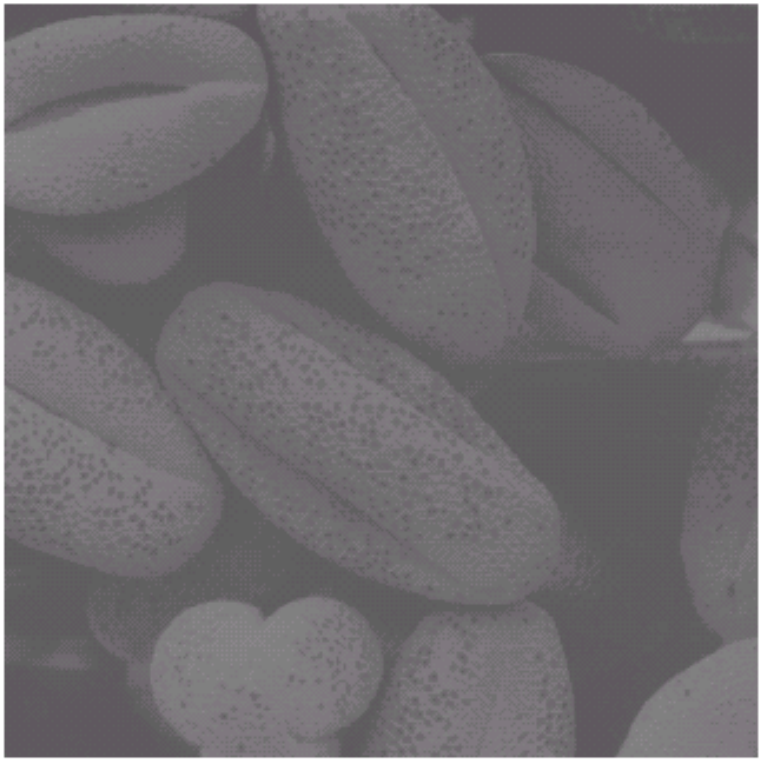
Today's Contents

- Enhancement
- Intensity transformation functions
  - Linear
  - Logarithmic
  - Power law
- Piecewise-Linear Transformation function
  - Contrast stretching
  - Intensity-level slicing
  - Bit-plane slicing

Chapter 3 from R.C. Gonzalez and R.E. Woods, Digital Image Processing (3rd Edition), Prentice Hall, 2008 [**Section 3.2**] [**Exercise Problems 3.1, 3.4, 3.5**]

Image Enhancement

Image Enhancement



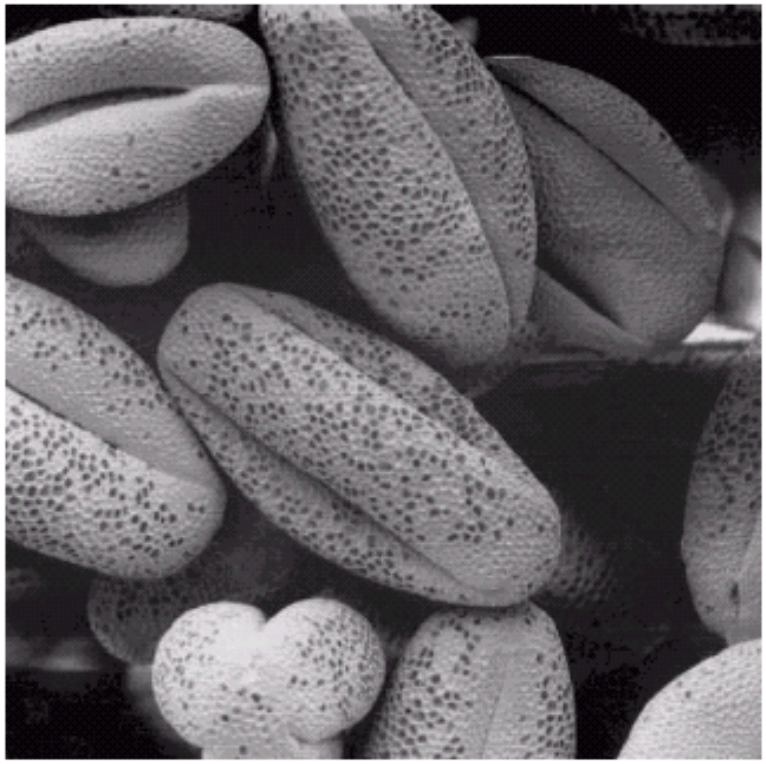


Image Enhancement

Process/manipulate an image so that the result is more suitable than the original image for a specific application.

**Problem oriented

- **Image Enhancement Methods**
  - Spatial Domain: Direct manipulation of pixels in an image
  - Frequency Domain: Process the image by modifying the Fourier transform of an image
  - Combination Methods:

This Chapter – Spatial Domain

>>>http://www.cs.uregina.ca/Links/class-info/425/index.html

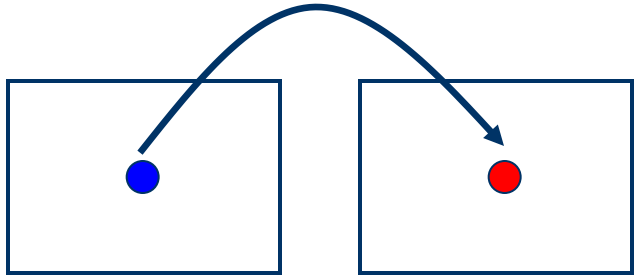
Image Enhancement Purposes

- The reasons for doing this include:
  - Highlighting interesting detail in images
  - Removing noise from images
  - Making images more visually appealing

Types of image enhancement operations

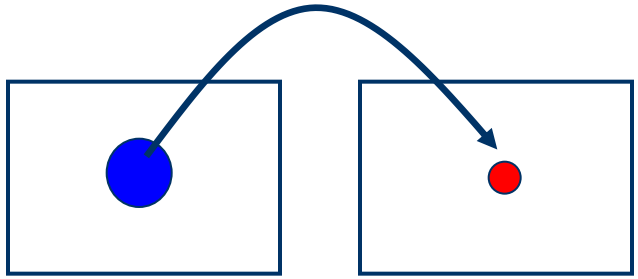
- Point/Pixel operations

  Output value at specific coordinates *(x,y)* is dependent only on the input value at *(x,y)*



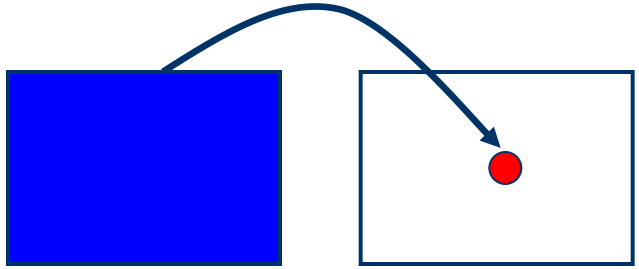
- Local operations

  The output value at *(x,y)* is dependent on the input values in the neighborhood of *(x,y)*



- Global operations

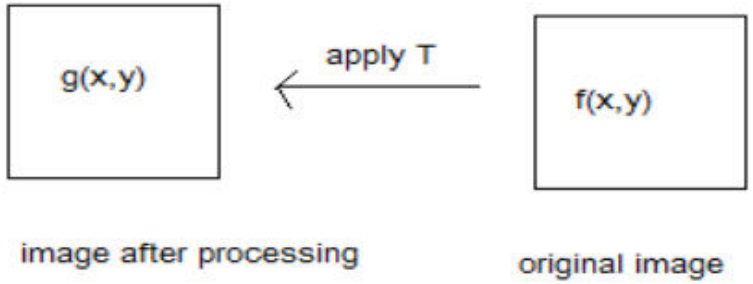
  The output value at *(x,y)* is dependent on all the values in the input image



Basic Concepts

- Spatial Domain Process

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



$f(x, y)$ = the input image

$g(x, y)$ = the processed/output image

T = some operator defined over some neighbourhood of (x, y)

- Intensity Transformation: point operation
- Spatial Filter: by mask, kernel, template or window.

Grey level/Intensity Transformation Function

$$\underbrace{g(x,y)}_{s} = T[\underbrace{f(x,y)}_{r}]$$

- Neighborhood of size 1x1:
- *g* depends only on *f* at *(x,y)*
- *T*: Gray-level/intensity transformation/mapping function

$$s = T(r)$$

  - *r* = Grey level of *f* at *(x,y)*
  - *s* = Grey level of *g* at *(x,y)*
  - *T* = *is a function that maps r to s*

Intensity transformations functions

Intensity transformation functions fall into 2 approaches:

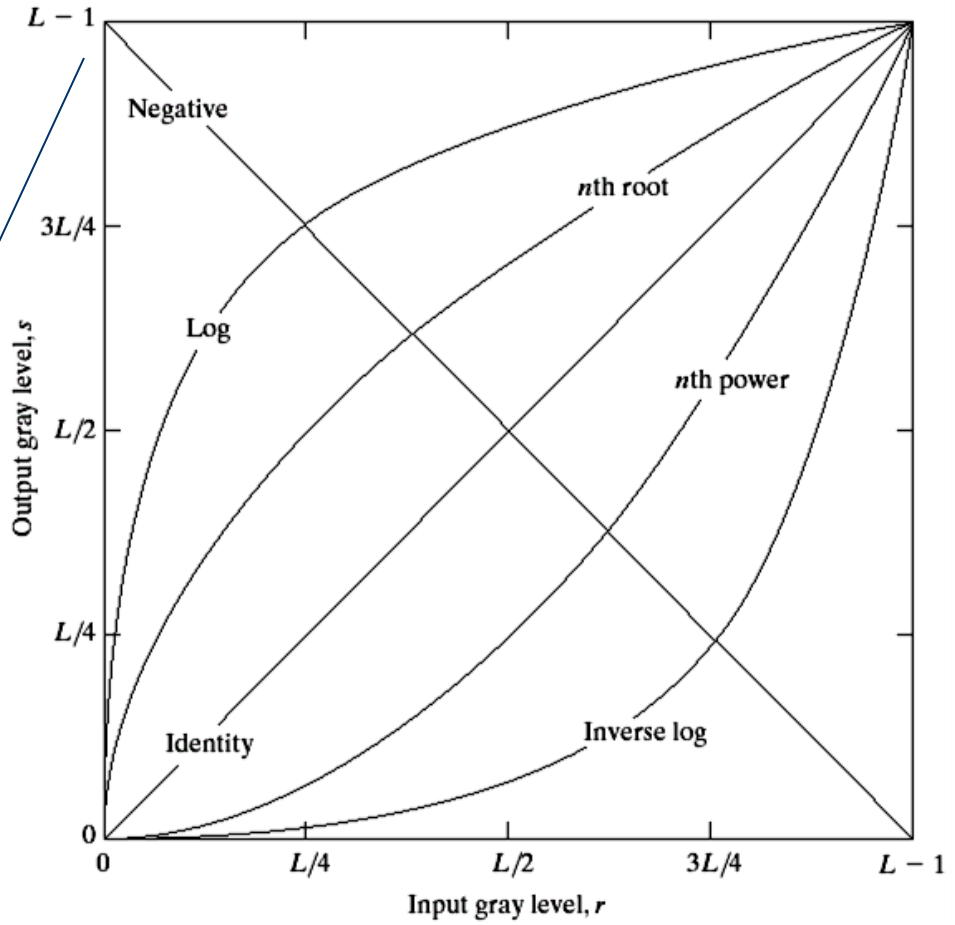
1) **Basic intensity transformations**
   - •Linear Functions:
     - Negative Transformation
     - Identity Transformation
   - •Logarithmic Functions:
     - Log Transformation
     - Inverse-log Transformation
   - •Power-Law Functions:
     - n^{th} power transformation
     - n^{th} root transformation

2) **piecewise Linear transformation functions.**
   a) Contrast stretching, thresholding
   b) Gray-level slicing
   c) Bit-plane slicing

Some Basic Intensity Transformation Functions

- Linear
  - Negative/Identity
- Logarithmic
  - Log/Inverse log
- Power law
  - n^{th} power/n^{th} root

Gray levels in the image after modification.
For example, if the pixel is 8 bit there will be 256, (L=256)

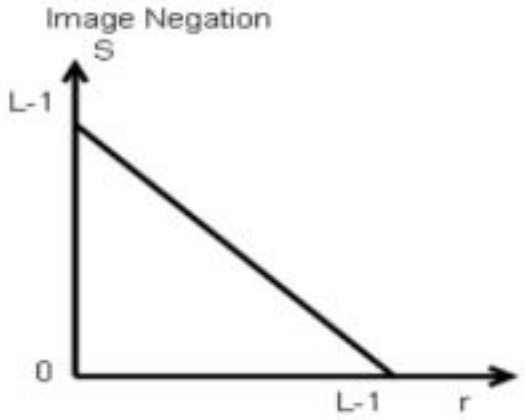


Gray levels in the image before modification

Linear Function: Negative Transformation

- Reverses the Grey level order
- For L Grey levels, the transformation (Image Negation) function has the form:

$$s = (L-1) - r$$



- Negative images are useful for enhancing white or grey detail embedded in dark regions of an image, especially when the black area are dominant in size.

Image Negatives (Negative Transformation)

Image (r)

Image (s) after applying T (negative)

Advantages of negative :

- ✓ Produces an equivalent of a photographic negative.
- ✓ Enhances white or gray detail embedded in dark regions.

Example: Negative Images

Input image (X-ray image)

Output image (negative)

Image Negatives (Negative Transformation)

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.

solution:
$L = 2^8 = 256$

$s = L-1-r$
$s = 255-r$

Apply this transform to each pixel to find the negative

Image (r)

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

Image (s)

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

Image Negatives (Negative Transformation)

Exercise:

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:

Image (s)

Linear Function: Identity Transformation

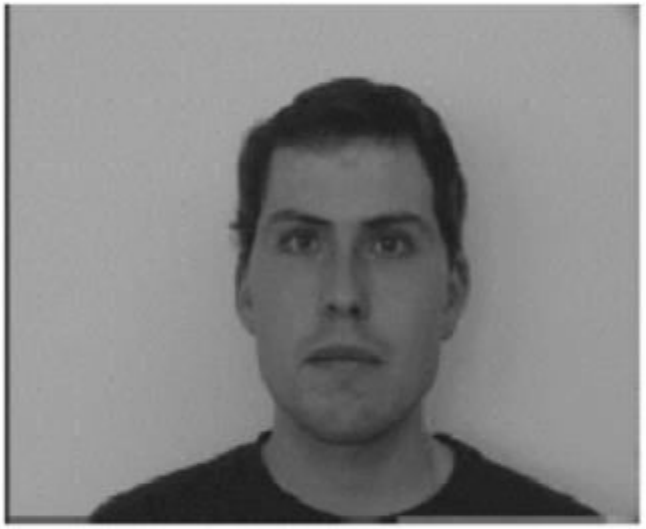
- Output intensities are identical to input intensities
- This function doesn't have an effect on an image, it was included in the graph only for completeness.
- Its expression:

$$s = r$$

Linear Function: Intensity Scaling

$$s = T(r) = a.r$$

Original image



$f(x, y)$

Scaled image

$a \cdot f(x, y)$

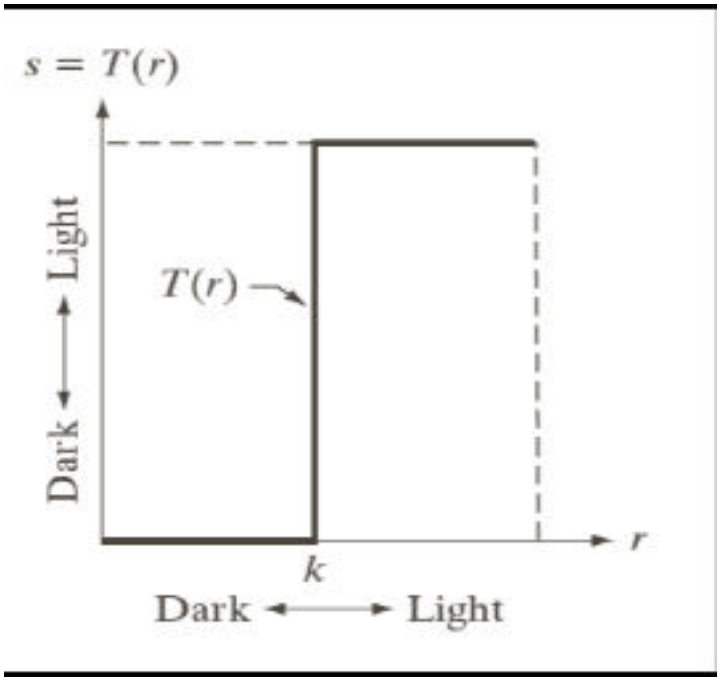
Linear Function: Thresholding

- **Contrast enhancement:**
  - **Darkens level below k**
  - **Brightens levels above k**

$$s = T(r)$$

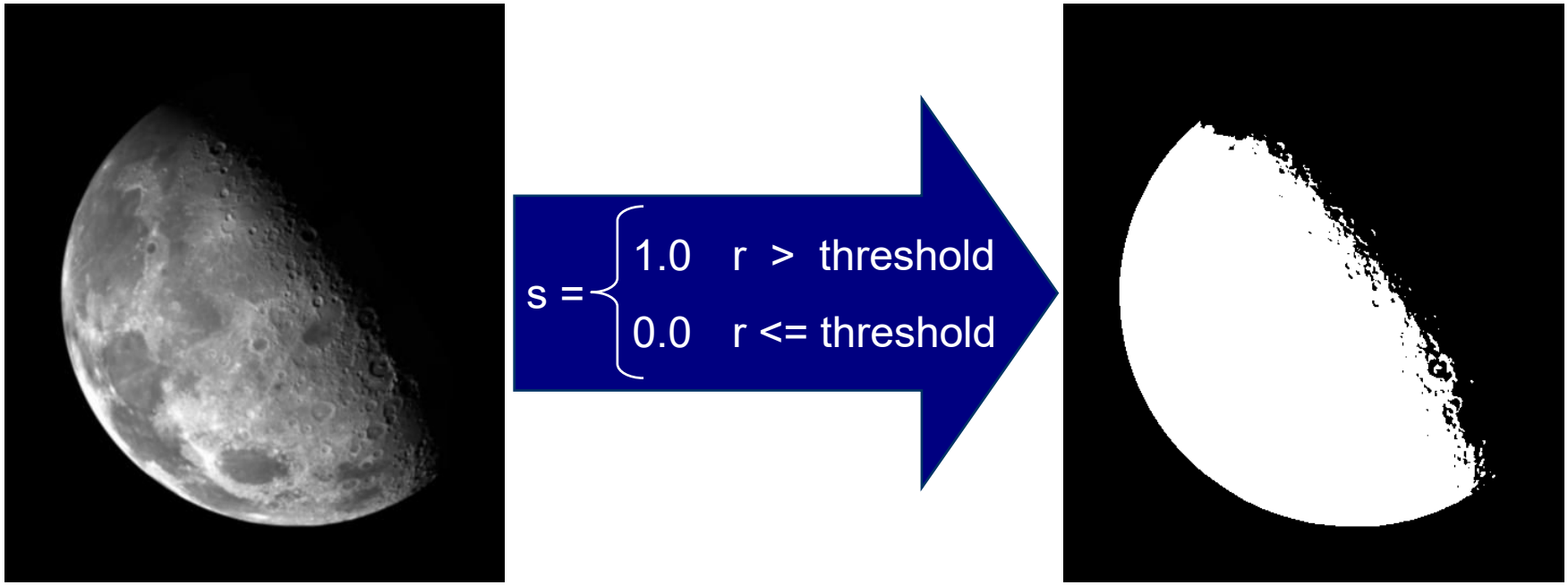
- **Replace values below k to black (0.0)**
- **Replace values above k to white (1.0)**

$$s = \begin{cases} 1.0 & r > \text{threshold} \\ 0.0 & r <= \text{threshold} \end{cases}$$



Example: Thresholding

- Segmentation of an object of interest from a background



Logarithmic Function: Log Transformations

- The general form of the log transformation is

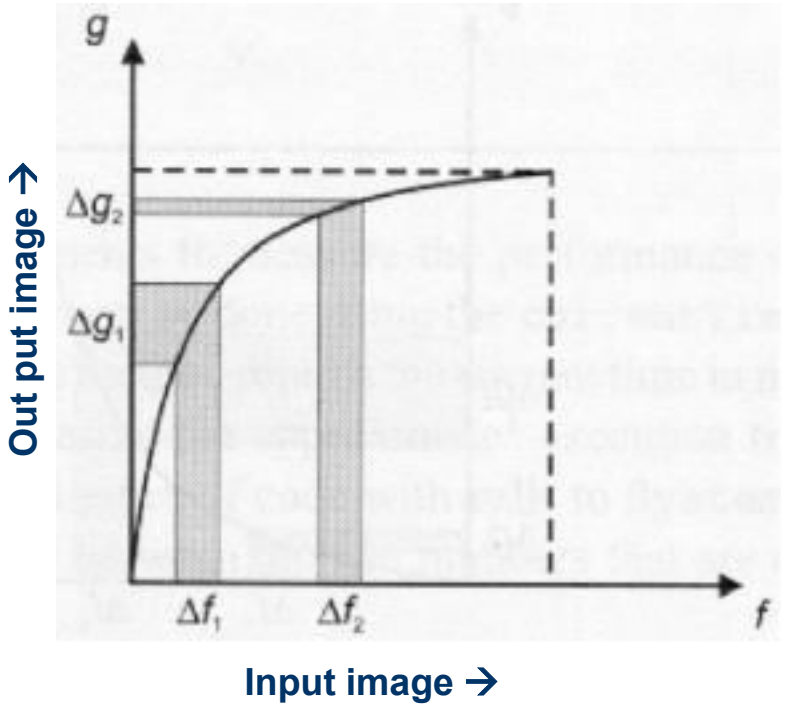
$$s = c \times \log(1 + r)$$

- The log transformation maps a **narrow range** of low input Grey level values into a **wider range** of output values

- The **inverse log** transformation performs the **opposite transformation**

Logarithmic Function: Log Transformations

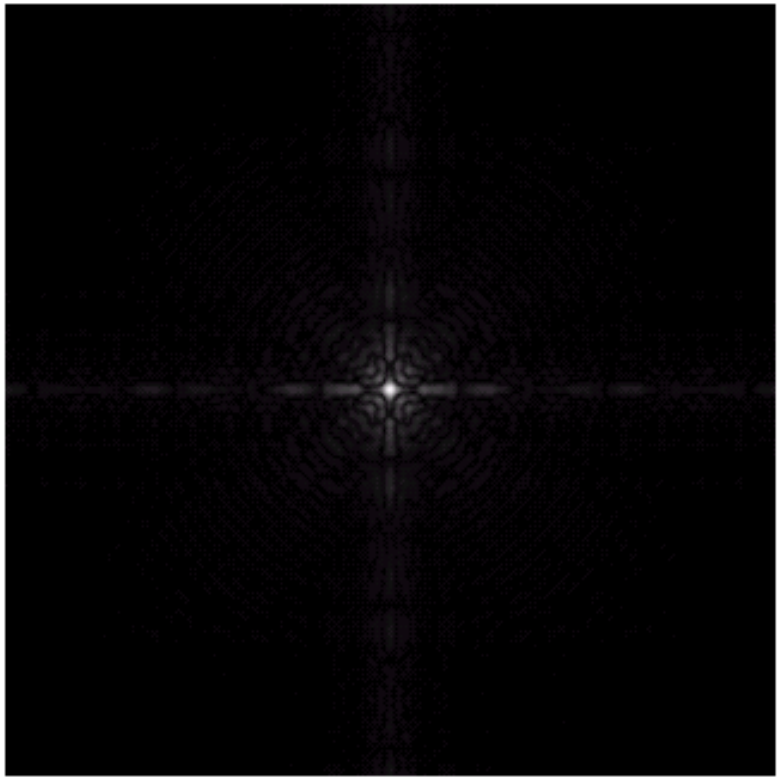
- **Properties**
  - For lower amplitudes of input image the range of Grey levels is **expanded**
  - For higher amplitudes of input image the range of Grey levels is **compressed**



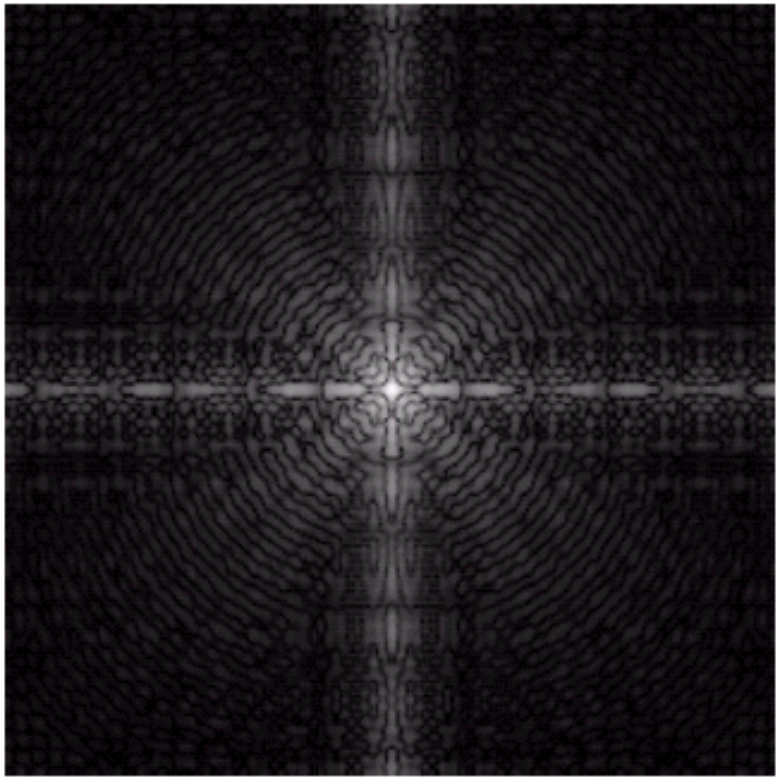
Logarithmic Function: Log Transformations

- **Application**
  - Log functions are particularly useful when the input grey level values may have an extremely large range of values
  - This transformation is suitable for the case when the dynamic range of a processed image far exceeds the capability of the display device (e.g. display of the Fourier spectrum of an image)
  - Also called "dynamic-range compression / expansion"

Example: Log Transformations



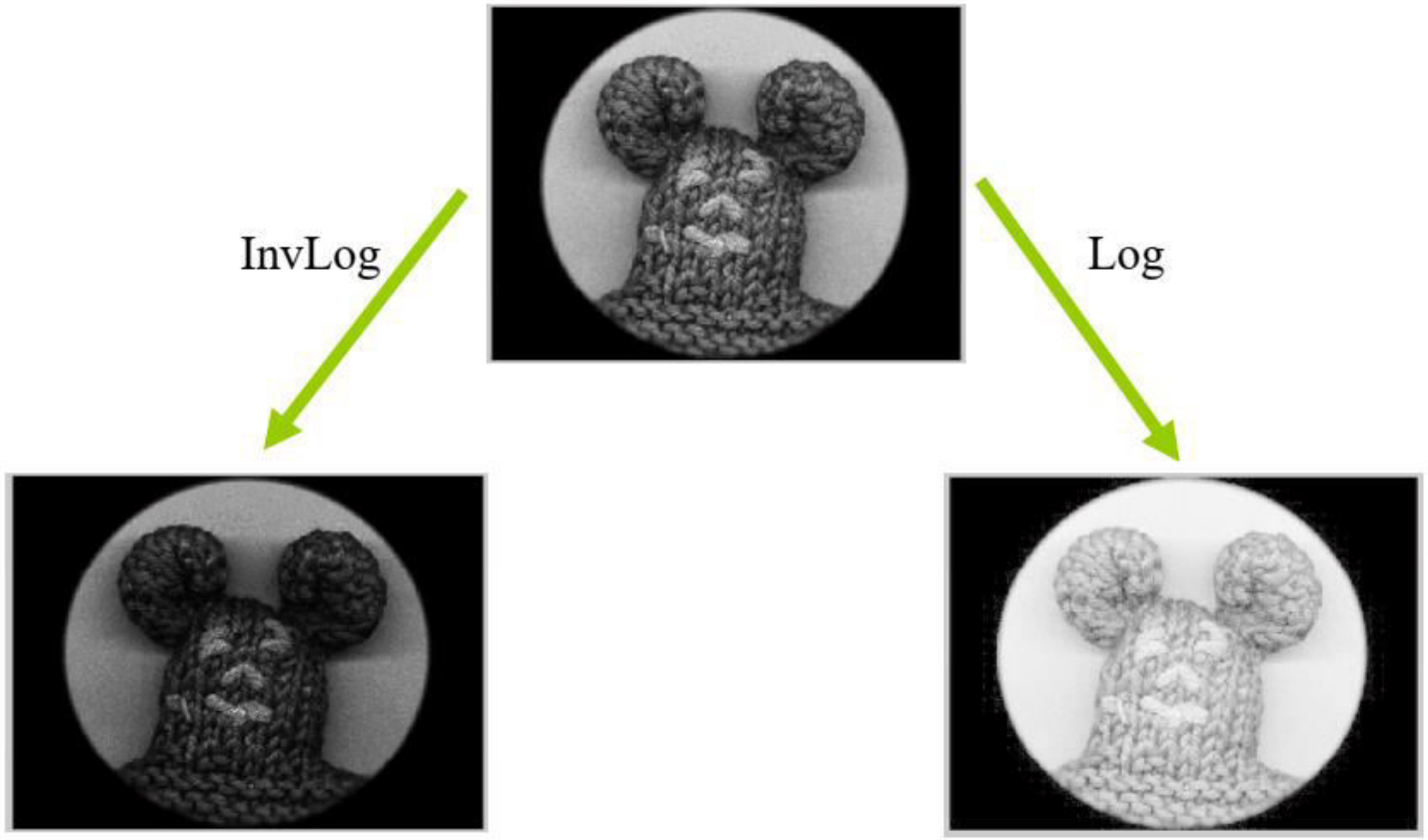
Fourier spectrum: image values ranging from 0 to 1.5×10^6



The result of log transformation with c = 1

Inverse Logarithm Transformation

- Do opposite to the log transformations.
- Used to expand the values of high pixels in an image while compressing the darker-level values



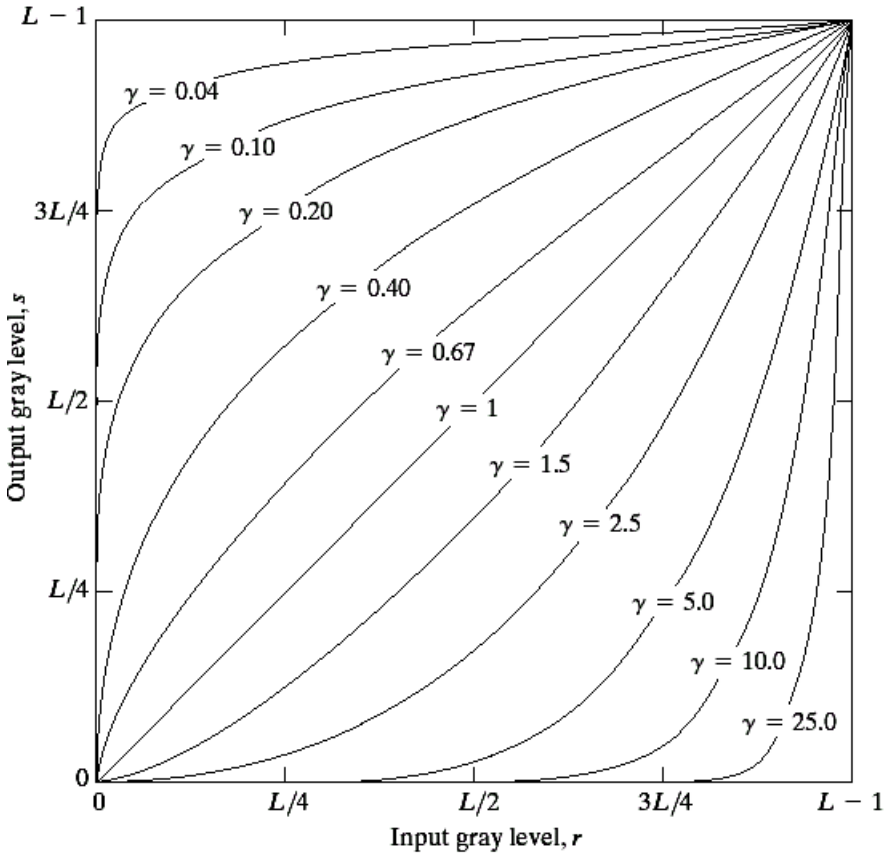
Power Law Function: Gamma Transformations

- Power law (Gamma) transformations have the following form

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

Where c and γ are positive constants.

- Map a narrow range of dark input values into a wider range of output values or vice versa.
- **Varying *γ* gives a whole family of curves.**



Different transformation curves are obtained by varying γ (gamma)

Power Law Function: Gamma Transformations

- 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.

- A variety of devices (image capture, printing, display) respond according to a power law and need to be corrected
- **Gamma (γ) correction**

  The process used to correct the power-law response phenomena

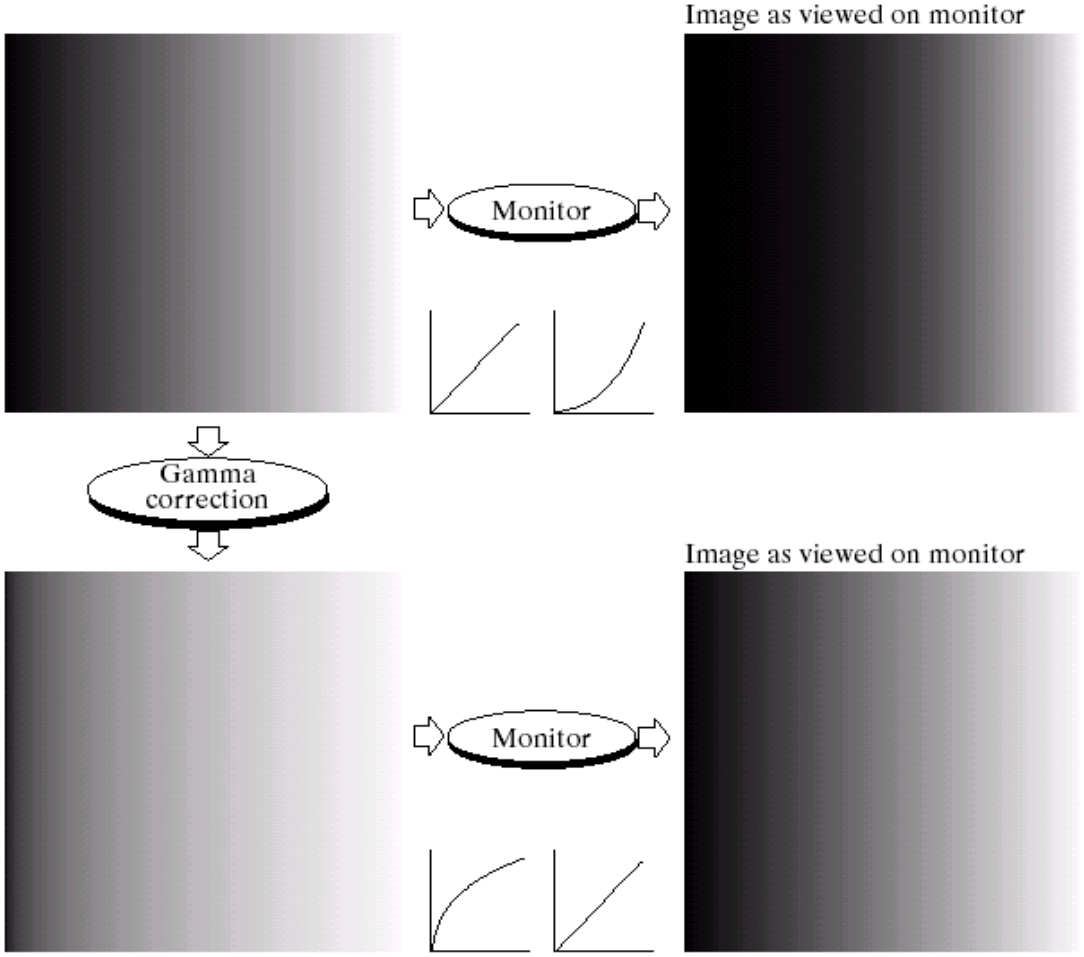
Gamma (γ) Correction

- Gamma correction is important to display an image accurately on a screen.
- Without any correction images can look either bleached out, or more likely too dark.
- Trying to reproduce colors accurately also requires gamma correction.
- It is also useful for general purpose contrast manipulation.

Power Law Transformations: Gamma Correction

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.



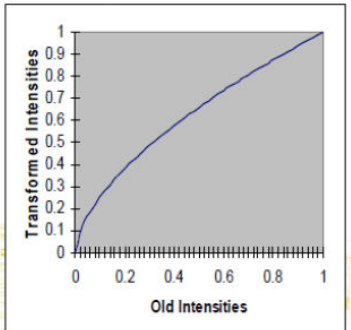
Power Law Transformations Contrast Enhancement

The image to the right show a magnetic resonance (MR) image of a fractured human spine with dislocation and spinal cord impingement

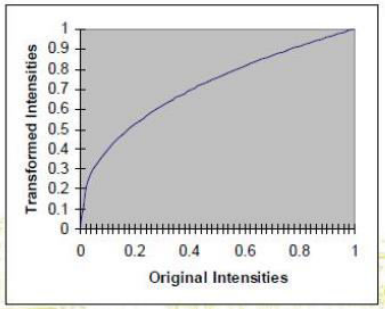
Power Law Transformations
Contrast Enhancement

MR image of fractured human spine With dislocation

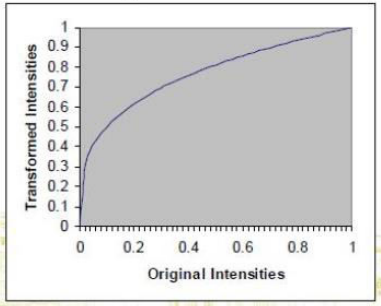
$\gamma = 0.6$



$\gamma = 0.4$



$\gamma = 0.3$



Power Law Transformations
Contrast Enhancement

MR image of fractured human spine With dislocation

Result after

Power law transformation

c = 1, γ = 0.6

Result after

Power law transformation

c = 1, γ = 0.4

Result after

Power law transformation

c = 1, γ = 0.3

Different curves highlights different detail

Image Enhancement

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

Result of Power law transformation c = 1, γ = 3.0 (suitable)

Result of Power law transformation c = 1, γ = 4.0 (suitable)

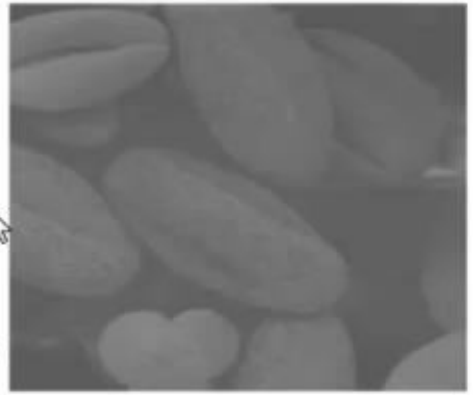
Result of Power law transformation c = 1, γ = 5.0 (high contrast, some regions are too dark)

Piecewise Linear Transformation Functions

- **Principle Advantage:** Some important transformations can be formulated only as a piecewise function.
- **Principle Disadvantage:** Their specification requires more user input than previous transformations.
- **Types of Piecewise transformations are:**
  - Contrast stretching
  - Grey/Intensity level slicing
  - Bit-Plane slicing

Contrast Stretching

- One of the simplest **piecewise linear** functions is a contrast-stretching transformation.
- **Low-contrast** images can result from
  - poor illumination,
  - lack of dynamic range in the imaging sensor,
  - or even wrong setting of a lens aperture during image acquisition.



- The idea behind contrast stretching is to **increase** the **dynamic range** of the gray levels in the image being processed.

Contrast Stretching

Contrast can be simply explained as the difference between maximum and minimum pixel intensity in an image.

The following example shows three 8-bit images with different contrast level

100	100	100	100
100	100	100	100
100	100	100	100
100	100	100	100

Contrast level
100 -100 = **0**
(No contrast)

0	10	20	30
40	60	80	90
100	120	150	180
200	220	240	255

Contrast level
255 - 0 = **255**
(High-contrast)

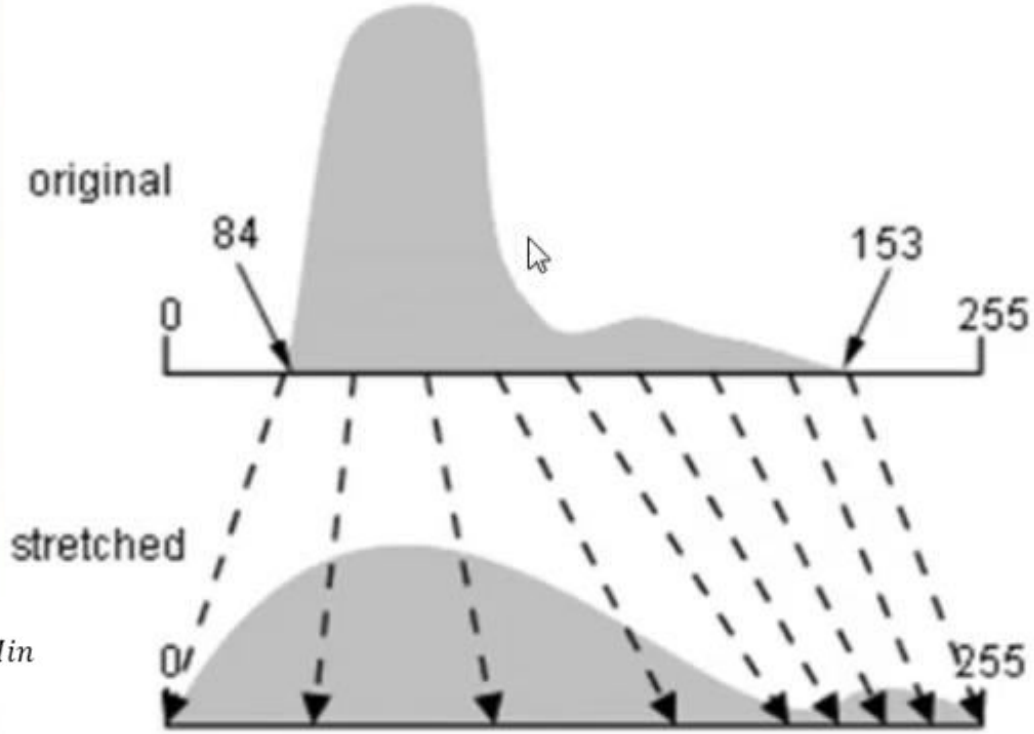
100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	150

Contrast level
150 - 100 = **50**
Low-contrast

Contrast Stretching

Contrast stretching is also known as normalization. The quality of image is enhanced by stretching the range of intensity values.

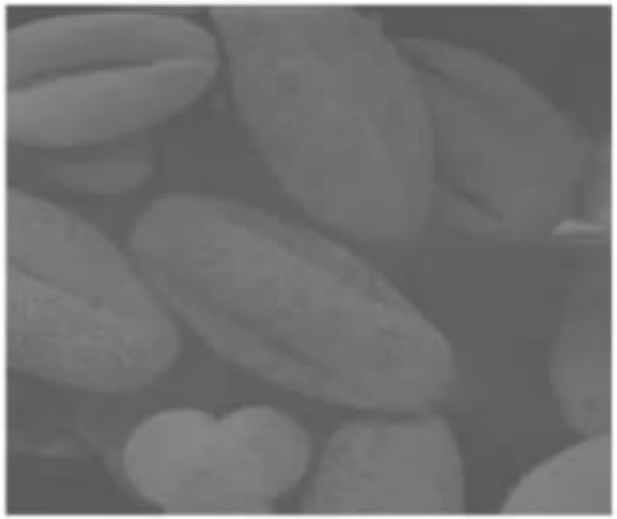
- Is a process that expands the range of intensity levels in an image so that it spans the full intensity range of the display device.



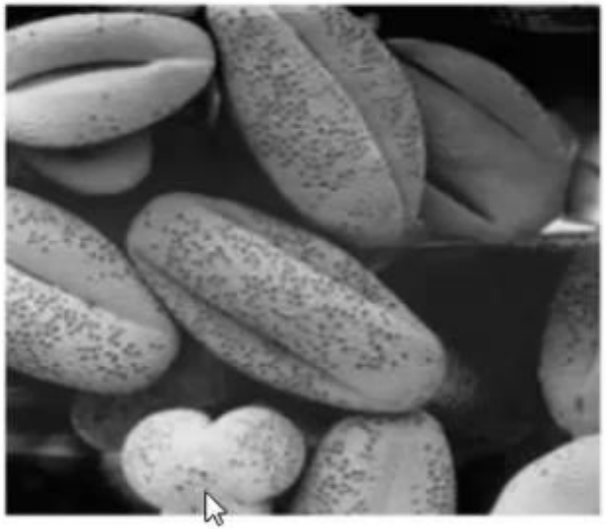
$$I_{new} = (I - Min)\frac{NewMax - NewMin}{Max - Min} + NewMin$$

Contrast Stretching

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.



Original Image



Contrast Enhanced Image

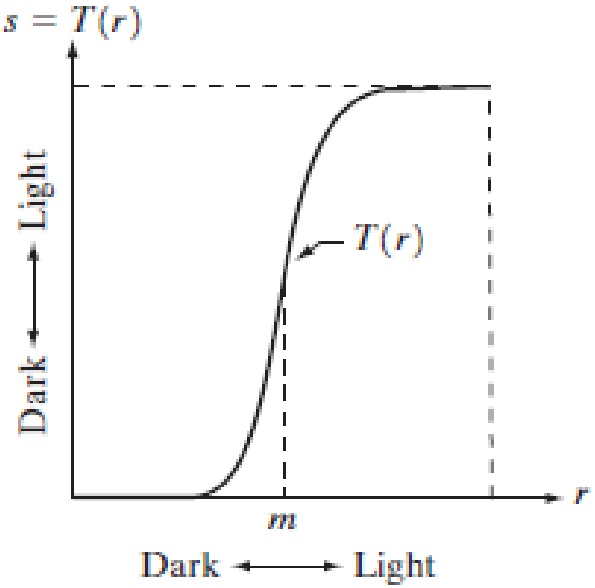
Contrast Stretching

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:

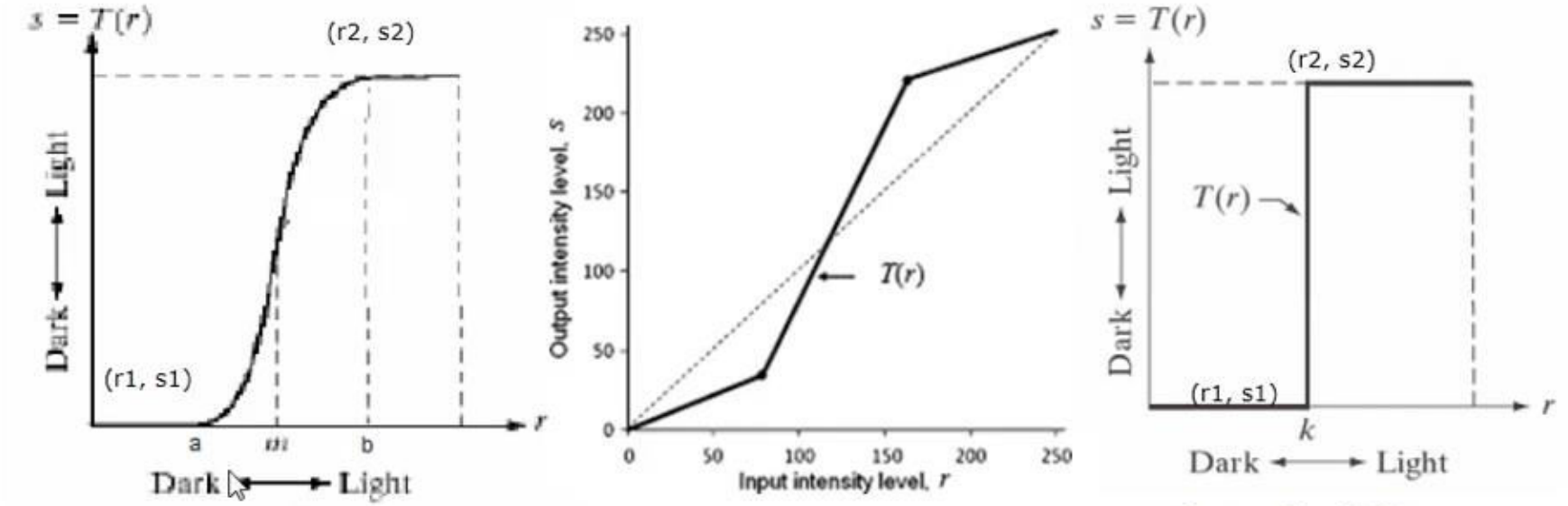
- Darkening the levels **below m** in the original image
- Brightening the levels **above m** in the original image

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 to span a desired range of values.**



Contrast Stretching

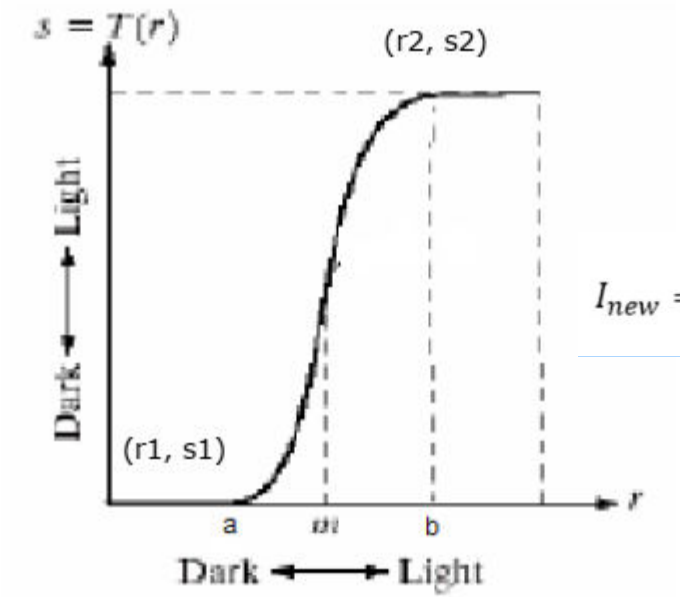
- To increase the dynamic range of the Grey levels in the image, Piecewise linear function a(r1, s1) & b(r2, s2) control the shape of the mapping, where r1<r2 & s1<s2.



Contrast stretching

Thresholding:

Contrast Stretching



Contrast stretching

$$I_{new} = (I - Min)\frac{NewMax - NewMin}{Max - Min} + NewMin$$

Assume that

$a=r_{min}$ (minimum input intensity)

$b=r_{max}$ (maximum input intensity)

where r1<r2 & s1<s2

Contrast stretching: (r1,s1)=(rmin,0) , (r2,s2)=(rmax,L-1)

where r_{min} and r_{max} denote the ***minimum*** and the ***maximum*** intensity level in the image. So the transformation function stretches the levels linearly from their original range to the full range **[0, L-1]**.

Contrast Stretching

- Example:

Input Image: 8-bit image have range [80 – 150]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min)\frac{NewMax - NewMin}{Max - Min} + NewMin$$

Where *I* is the input intensity, I_{new} is the output intensity after contrast stretching, *NewMax* and *NewMin* are the new intensity range (0 – 255), *Max* and *Min* are the input intensity range (80 – 150)

Contrast Stretching

- Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min)\frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast stretching

Min=100
Max=151

0			

Output Image

I = 100 : (100-100) * [(255-0)/(151-100)] +0
I = 100 → I_{new} = **0**

Contrast Stretching

- Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min)\frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast stretching

Min=100
Max=151

0	**25**		

Output Image

I = 105 : (105-100) * 5 +0
I = 105 → I_{new} = 5 * 5 = **25**

Contrast Stretching

- Example:

Input Image: 8-bit image have range [100 – 151]

After Contrast Stretching: the new range becomes [0 – 255] using the following equation:

$$I_{new} = (I - Min)\frac{NewMax - NewMin}{Max - Min} + NewMin$$

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast stretching

Min=100
Max=151

0	25	35	0
100	125	100	150
75	85	60	90
150	200	225	255

Output Image

NewMin=0
NewMax=255

MidPoint = m =125

Contrast Stretching

100	105	107	100
120	125	120	130
115	117	112	118
130	140	145	151

Input Image

After contrast stretching

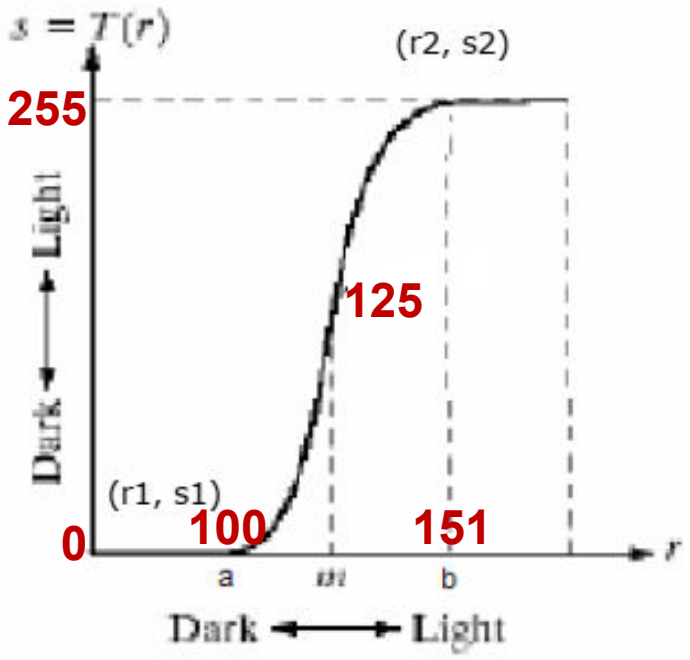
Min=100
Max=151

0	25	35	0
100	125	100	150
75	85	60	90
150	200	225	255

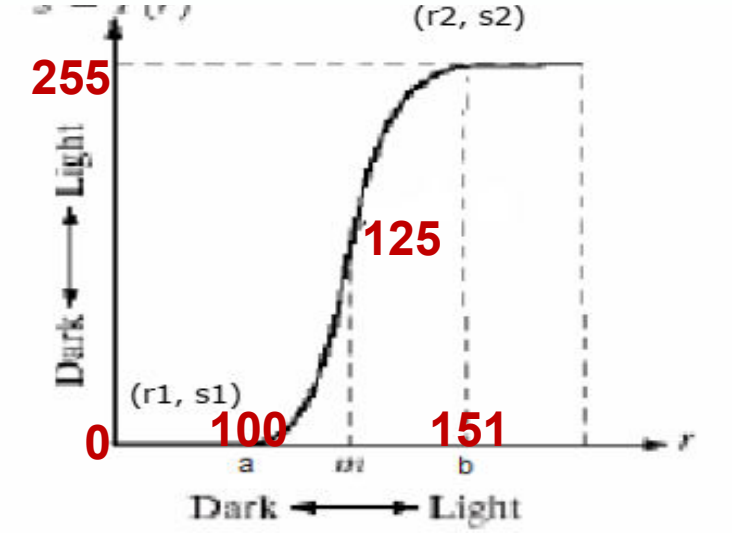
Output Image

NewMin=0
NewMax=255

MidPoint = m =125



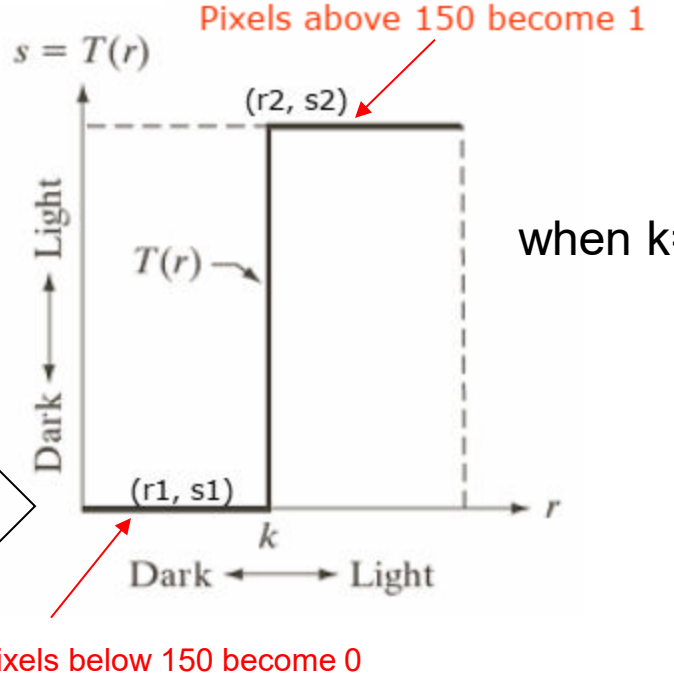
Contrast Stretching



What if r1=s1 & r2=s2?
OR
r1=r2, s1=0 & s2=L-1?

If $r_1 = s_1$ & $r_2 = s_2$,
no change in Grey levels.
Linear Transformation Function

If r1=r2, s1=0 & s2=L-1
This is called Thresholding,
and it produces a binary image!



when k=150

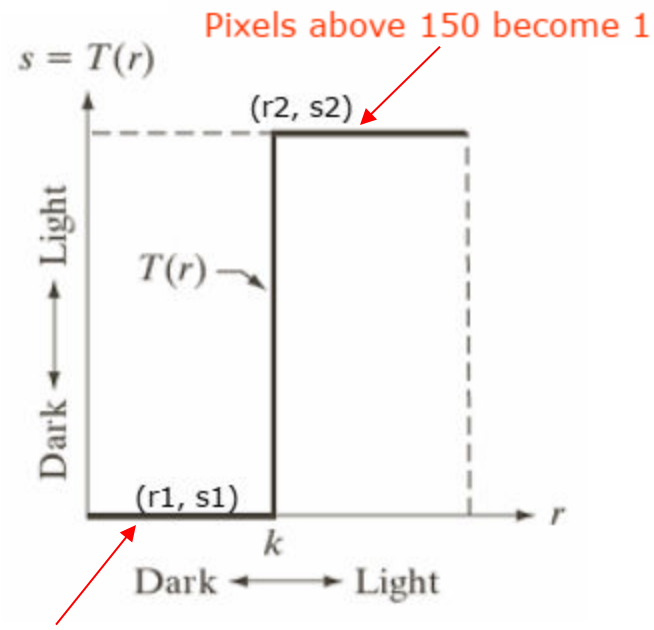
Contrast Stretching

Remember that:

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

Or

$$s = T(r)$$



Example: suppose k= 150 (called threshold),

if r (pixel intensity in original image f) is above this threshold it becomes 1 in s (pixel intensity in transferred image g), 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!

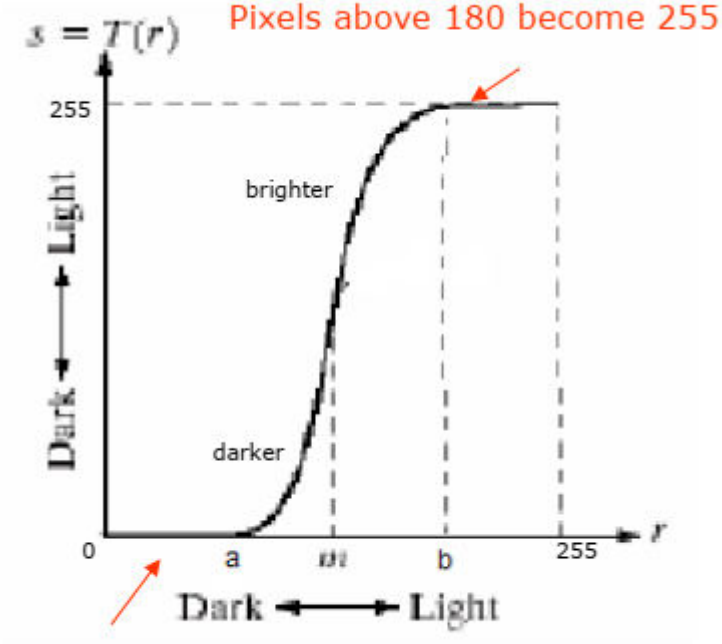
Contrast Stretching

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:

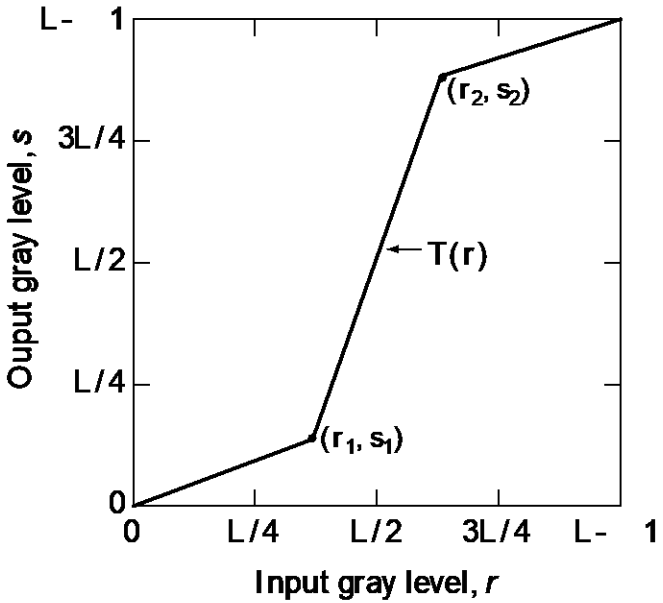
when r < 100 , s closes zero (darker)
when r>100 , s closes to 255 (brighter)

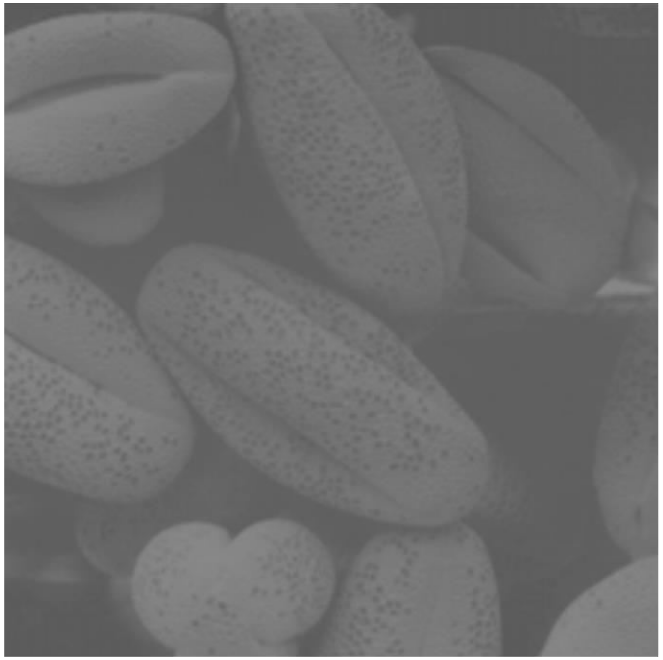
$$T = \begin{cases} \text{If } r > 180;\ s = 255 \\ \text{If } r < 180 \text{ and } r > 90;\ s = T(r) \\ \text{If } r < 90;\ s = 0 \end{cases}$$

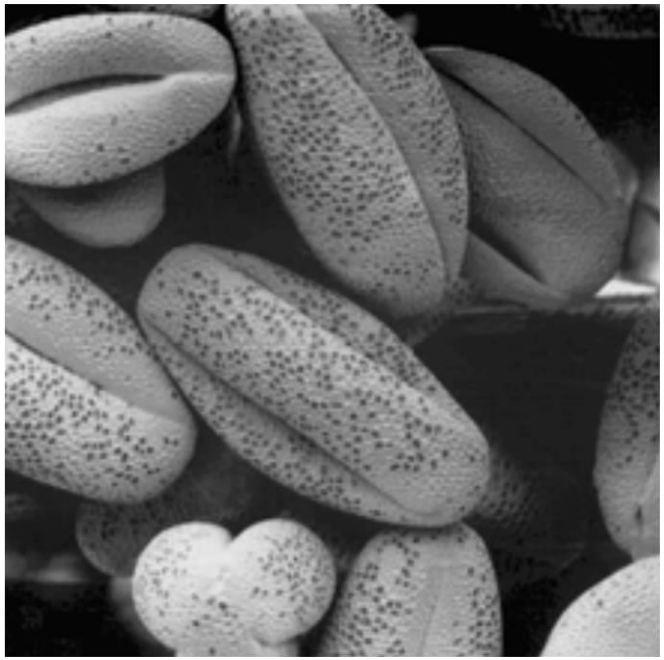
This is called contrast stretching, which means that the bright pixels in the image will become brighter and the dark pixels will become darker, this means : higher contrast image.

Contrast Stretching

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>

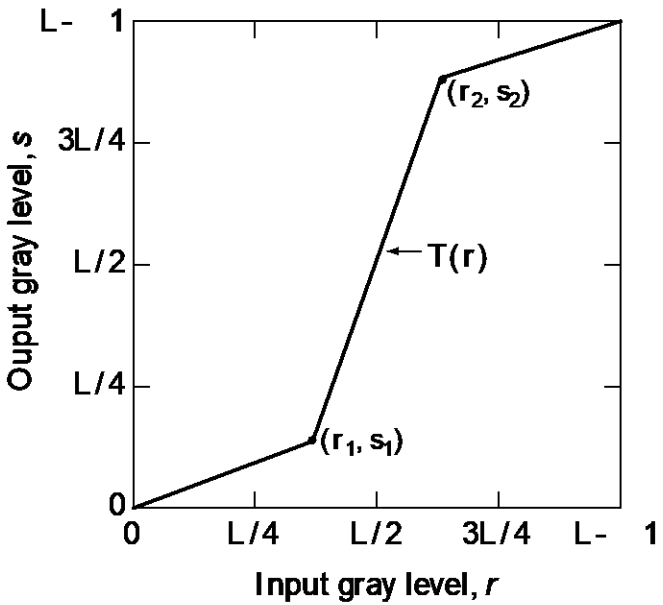


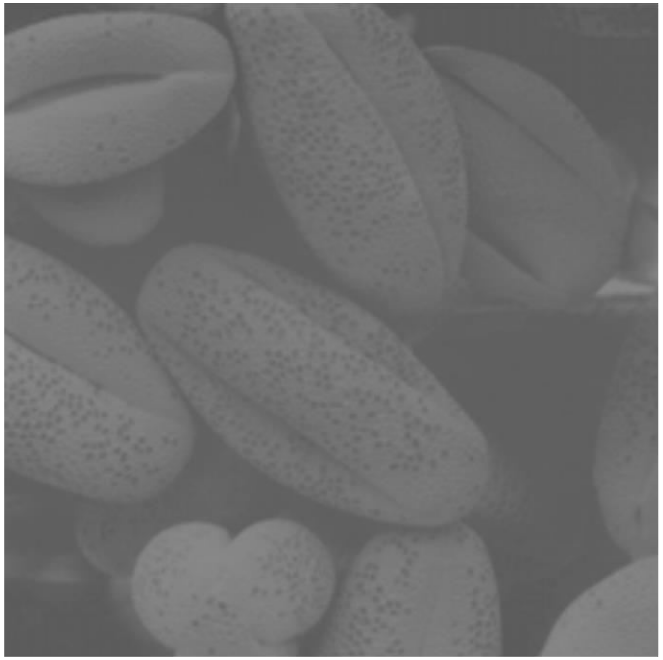




a b
c d

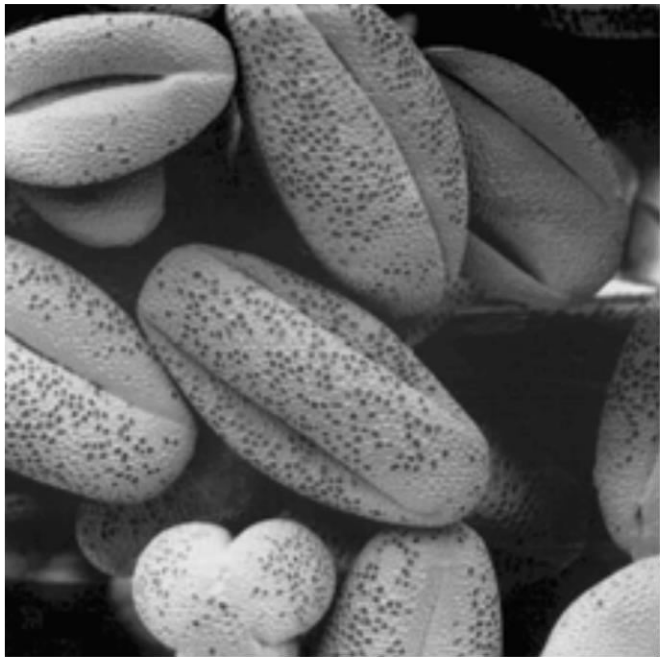
FIGURE 3.10
Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)





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.)



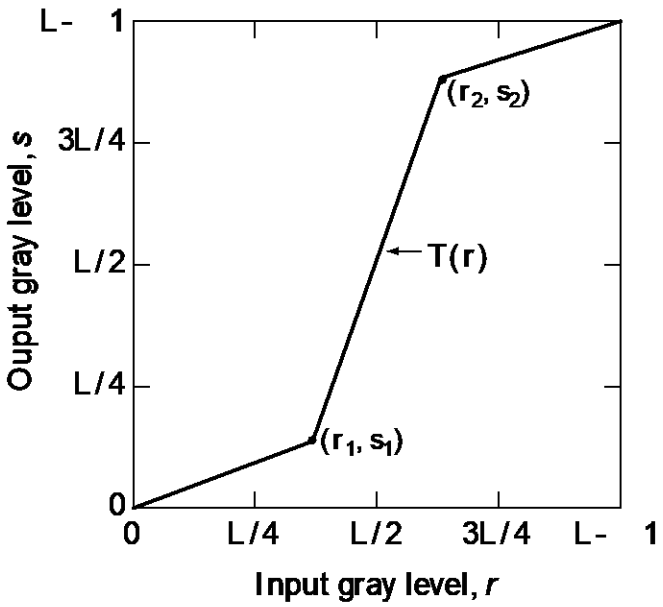
$r_1 = r_{min}$ & $s_1 = 0$

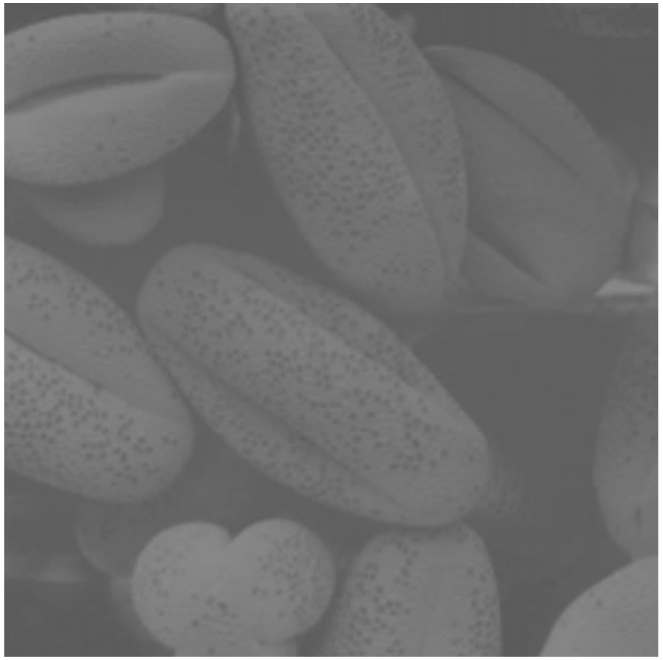
$r_2 = r_{max}$ & $s_2 = L\text{-}1$

$$I_N = (I - \text{Min})\frac{\text{newMax} - \text{newMin}}{\text{Max} - \text{Min}} + \text{newMin}$$

(as in fig 3.10 c)

- Min-Max Stretching





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.)

$r_1 = m \;\&\; s_1 = 0$

$r_2 = m \;\&\; s_2 = L\text{-}1$

Where, m is the mean intensity

(as in fig 3.10 c)

- Percentile Stretching

Contrast Stretching

- Figure 3.10(a) shows a typical transformation used for contrast stretching. The locations of points (r1, s1) and (r2, s2) control the shape of the transformation function.
- If r1 = s1 and r2 = s2, the transformation is a linear function that produces no changes in gray levels.
- If r1 = r2, s1 = 0 and s2 = L-1, the transformation becomes a *thresholding function* that creates a binary image.
- *Intermediate values of (r1, s*1) and (r2, s2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.
- In general, r1 ≤ r2 and s1 ≤ s2 is assumed, so the function is always increasing.

Contrast Stretching

- Figure 3.10(b) shows an 8-bit image with low contrast.
- Fig. 3.10(c) shows the result of contrast stretching, obtained by setting (r1, s1) = (r_{min}, 0) and (r2, s2) = (r_{max},L-1) where r_{min} and r_{max} denote the minimum and maximum gray levels in the image, respectively. Thus, the transformation function stretched the levels linearly from their original range to the full range [0, L-1].
- Finally, Fig. 3.10(d) shows the result of using the *thresholding function* defined previously, with r1=r2=m, the mean gray level in the image.

Contrast Stretching

Exercise:

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.

solution:

Image (r)

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

Image (s)

Gray/Intensity Level Slicing

This technique is useful when different features of an **image** are contained in different **Grey levels.**

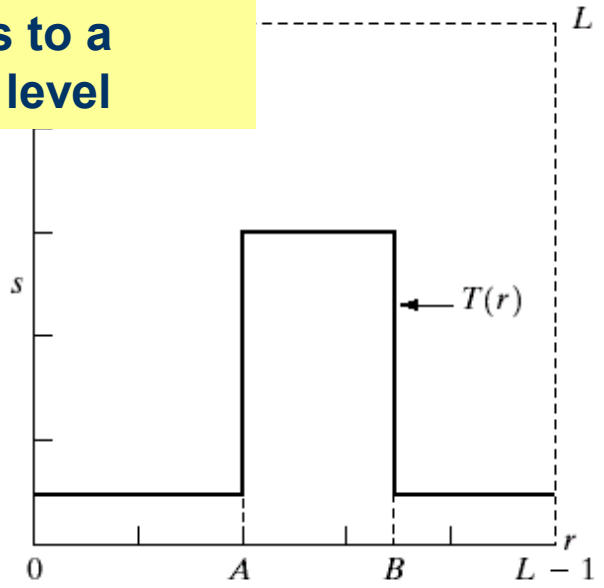
- Highlights a specific range of Grey levels in an image
- Similar to thresholding
- Other levels can be suppressed or maintained
- Useful for highlighting features in an image

Grey Level Slicing

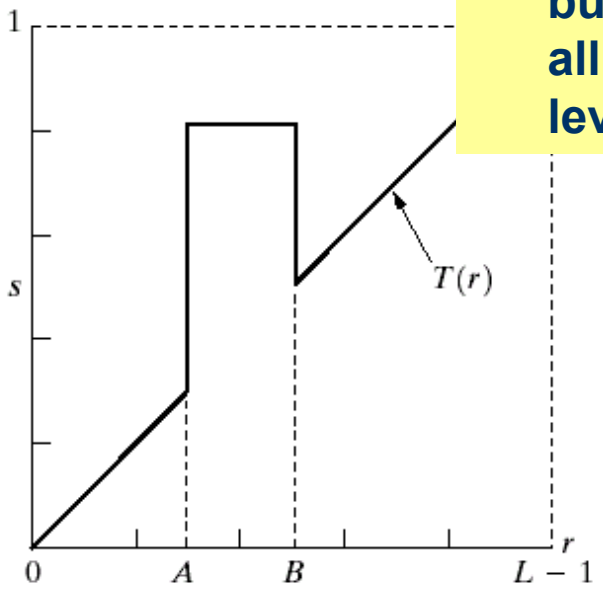
Highlights a specific range of Intensities levels in an image.

Two basic approaches

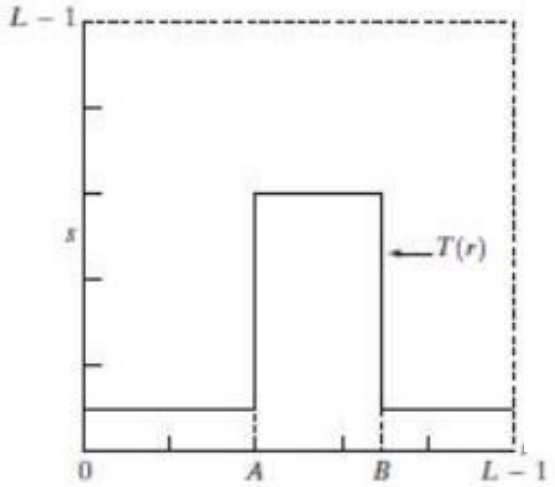
Highlights range [A,B] of grey levels and reduces all others to a lower level

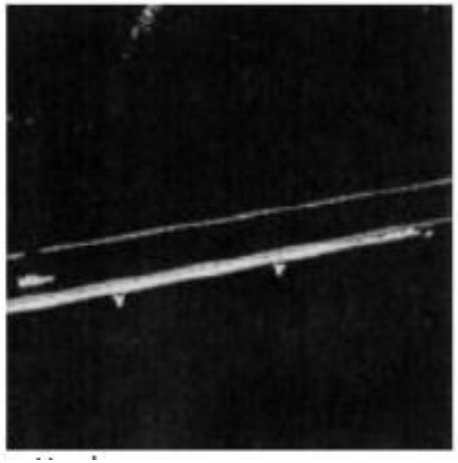


Highlights range [A,B] but preserves all other grey levels



Gray-level Slicing





a b
c d

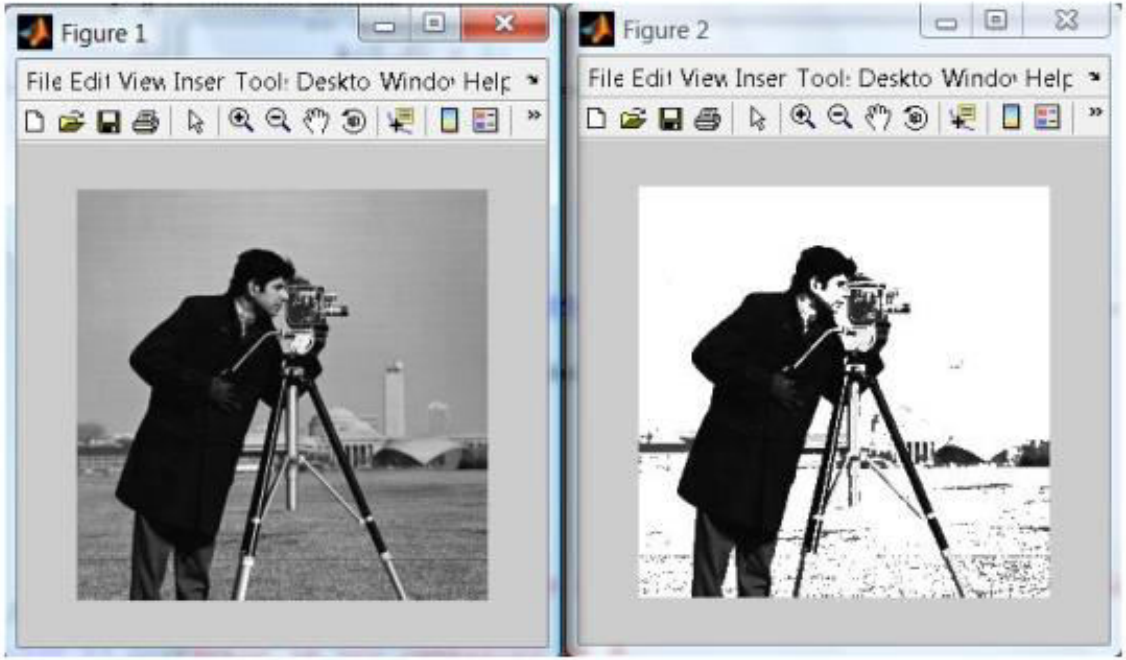
FIGURE 3.11
(a) This transformation highlights range $[A, B]$ of gray levels and reduces all others to a constant level. (b) This transformation highlights range $[A, B]$ but preserves all other levels. (c) An image. (d) Result of using the transformation in (a).

Gray-level Slicing: approach 1

example: 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).

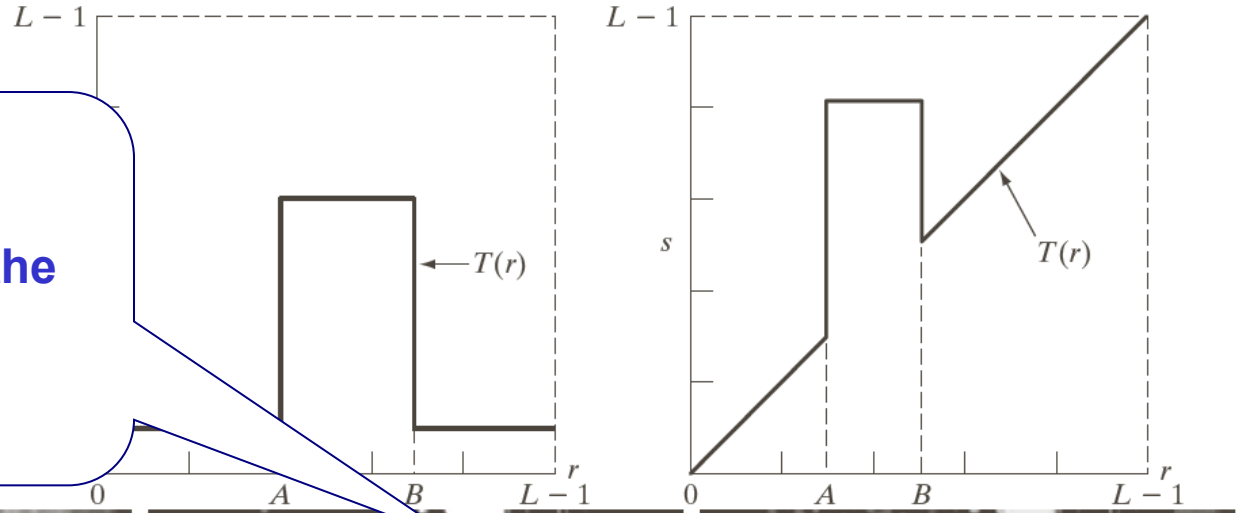
Gray-level Slicing: approach 2

example: 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.

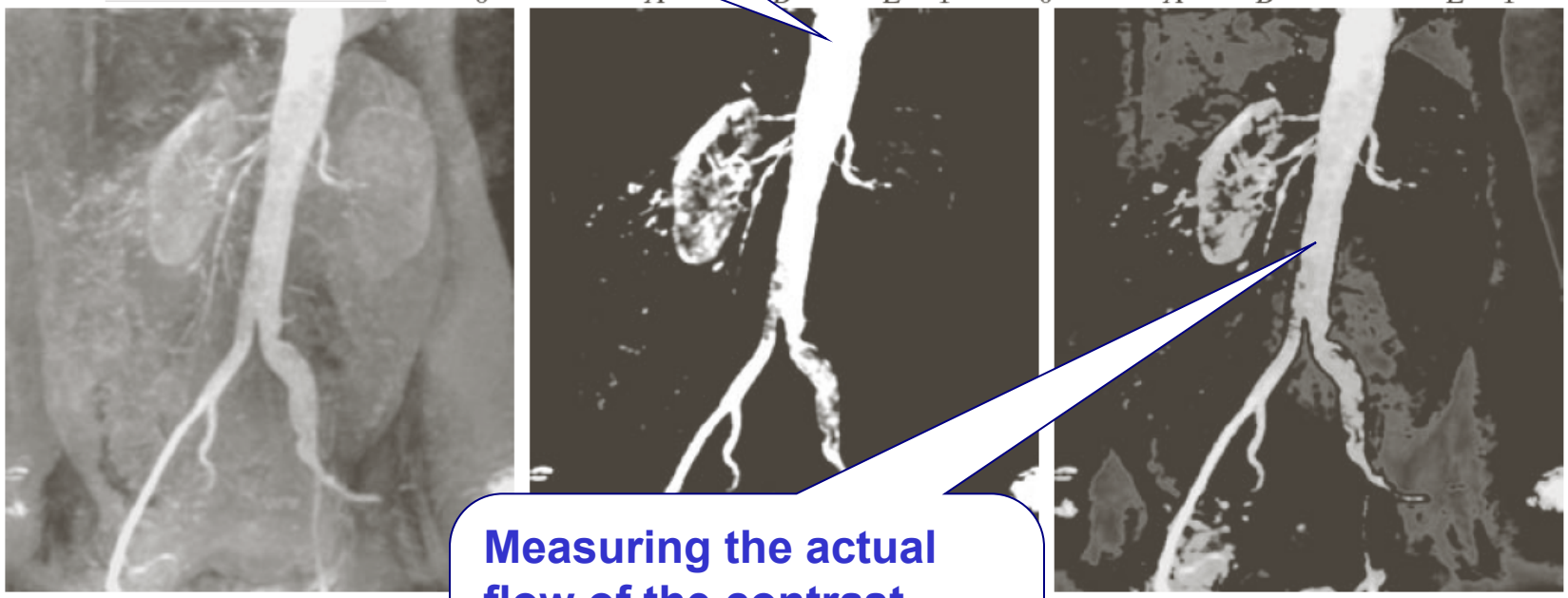


a b

FIGURE 3.11 (a) This



Highlight the major blood vessels and study the shape of the flow of the contrast medium (to detect blockages, etc.)



Measuring the actual flow of the contrast medium as a function of time in a series of images

a b c

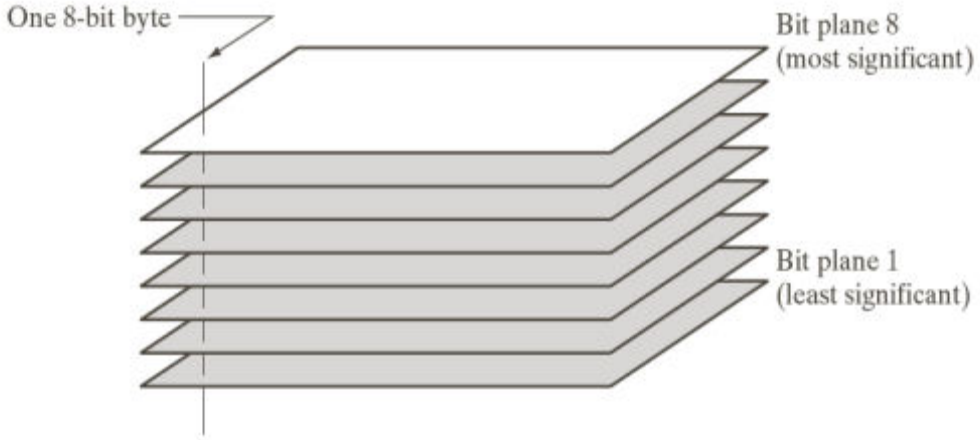
FIGURE 3.12 (a) Aortic angiog mation of the type illustrated in Fig. 3.11(a), with the range of inte end of the gray scale. (c) Result of using the transformation in Fig ack, so that grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)

Bit-plane Slicing

- Digitally, an image is represented in terms of pixels.
- Pixels are digital numbers, in terms of bits.
- The binary format for the pixel value, 0 is 00000000, 255 is 11111111.
- Similarly, for 167 is 10100111 and 144 it is 10010000.
- The bit in the far left side is referred as the most significant bit (MSB).
- The bit in the far right is referred as the least significant bit (LSB).

Bit-plane Slicing

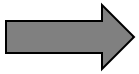
- The contribution made by each bit can be highlighted.
- This 8-bit image is composed of eight 1-bit planes.
- The bit plane representation of an eight-bit digital image is given by:



- Plane 1 contains the lowest order LSB of all the pixels in the image.
- And plane 8 contains the highest order MSB of all the pixels in the image

Bit-plane Slicing

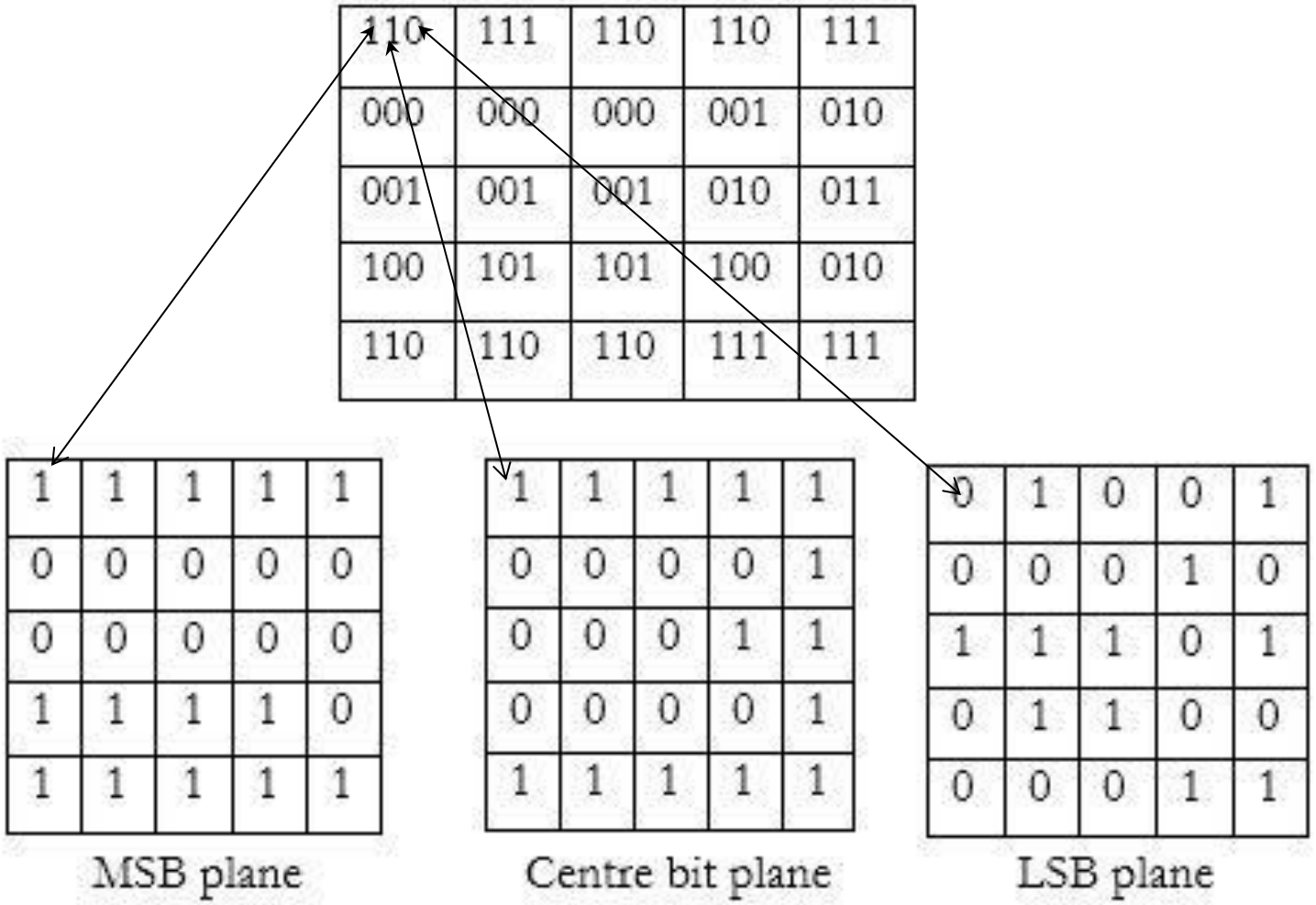
6	7	6	6	7
0	0	0	1	2
1	1	1	2	3
4	5	5	4	2
6	6	6	7	7



110	111	110	110	111
000	000	000	001	010
001	001	001	010	011
100	101	101	100	010
110	110	110	111	111

The given image is a **3-bit image**. We convert the image to binary

Bit-plane Slicing



110	111	110	110	111
000	000	000	001	010
001	001	001	010	011
100	101	101	100	010
110	110	110	111	111

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

MSB plane

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

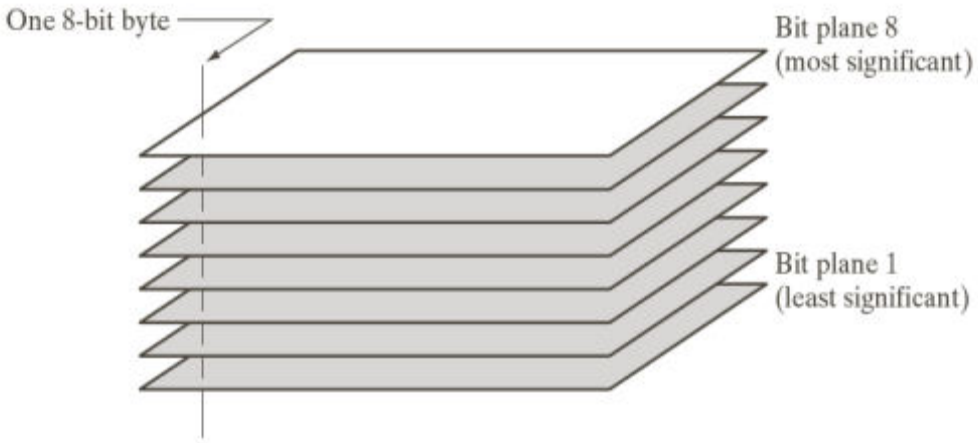
Centre bit plane

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

LSB plane

Separating the bit planes, we obtain 3 planes

Bit-plane Slicing



- Higher-order bits usually contain most of the significant visual information.

- Lower order bits contain subtle/slight details.

Bit-plane Slicing

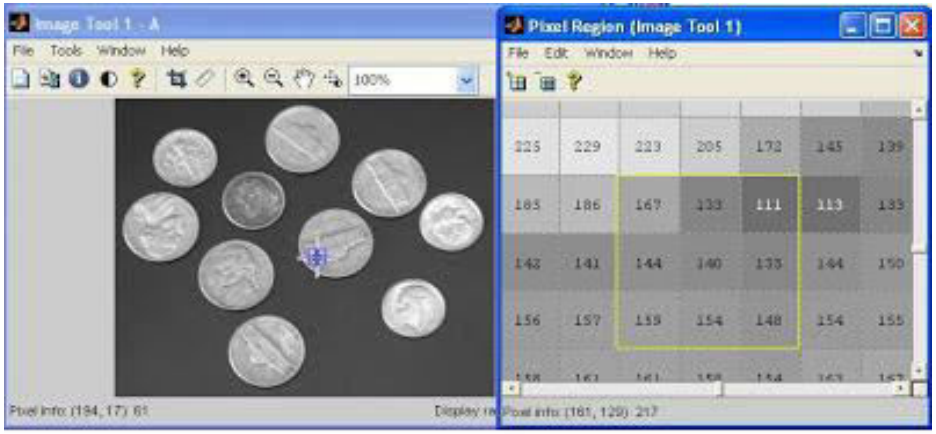
- One can use only MSB to represent the pixel, which reduces the original Grey level to a binary image.
- A change in MSB would significantly change the value encoded by the byte.
- But a change in LSB does not change the encoded Grey value much.

The three main goals of bit plane slicing is:

- Converting a Grey level image to a binary image.
- Representing an image with fewer bits and corresponding the image to a smaller size, i.e. compression.
- Enhancing the image by focusing.

Bit-plane Slicing Example

Consider the image 'coins.png' and the pixel representation of the image.



Consider the pixels that are bounded within the yellow line. The binary formats for those values are (8-bit representation)

10100111	10000101	01101111
10010000	10001100	10000111
10011111	10011010	10010100

The binary format for the pixel value 167 is 10100111 Similarly, for 144 it is 10010000

Bit-plane Slicing Example

This 8-bit image is composed of eight 1-bit planes.
Plane 1 contains the lowest order bit of all the pixels in the image.

10100111	10000101	01101111
10010000	10001100	10000111
10011111	10011010	10010100

1	1	1
0	0	1
1	0	0

And plane 8 contains the highest order bit of all the pixels in the image.

10100111	10000101	01101111
10010000	10001100	10000111
10011111	10011010	10010100

1	1	0
1	1	1
1	1	1

Bit-plane Slicing(example)

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

0 1 1 0 0 1 0 0

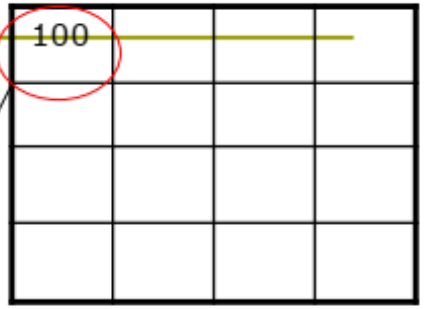


Image of bit1:
0000000**0**

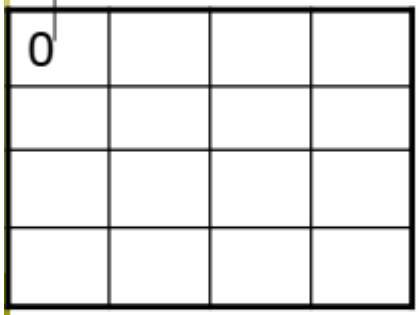


Image of bit2:
0000000**0**

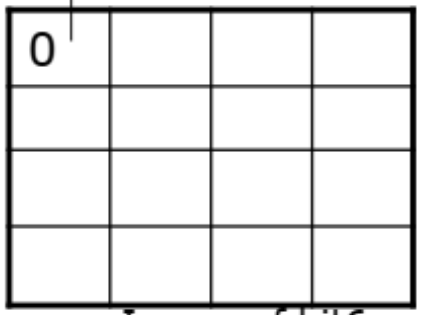


Image of bit3:
0000**1**00

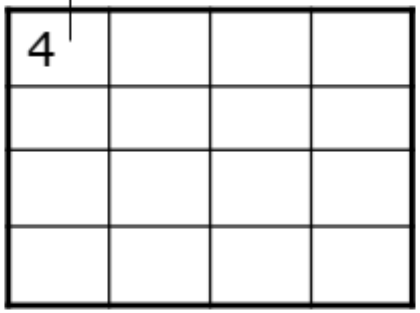


Image of bit4:
0000**0**000

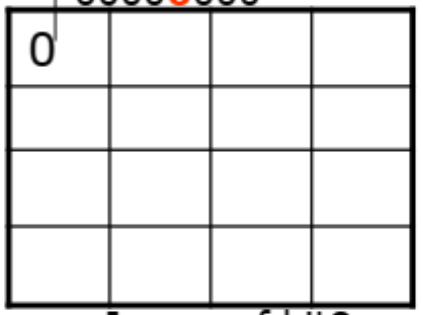


Image of bit5:
0000**0**000

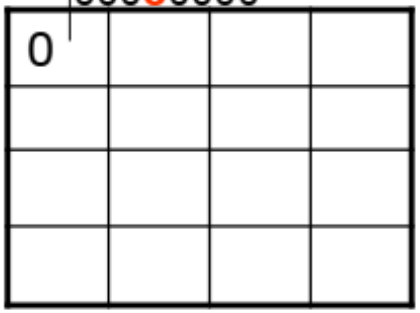


Image of bit6:
00**1**00000

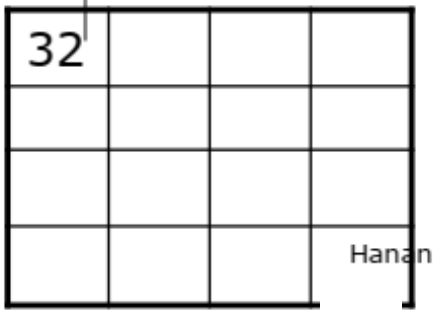


Image of bit7:
0**1**000000

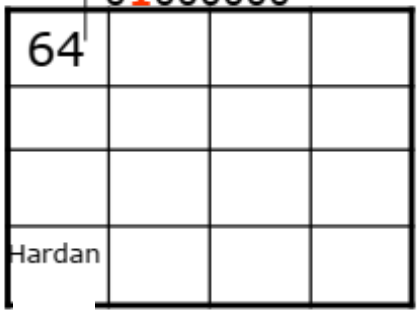
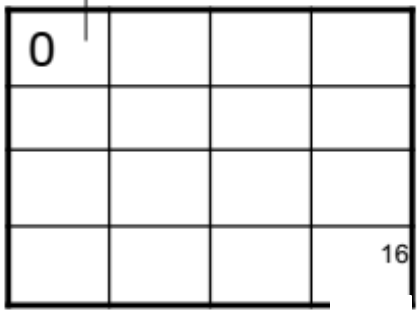


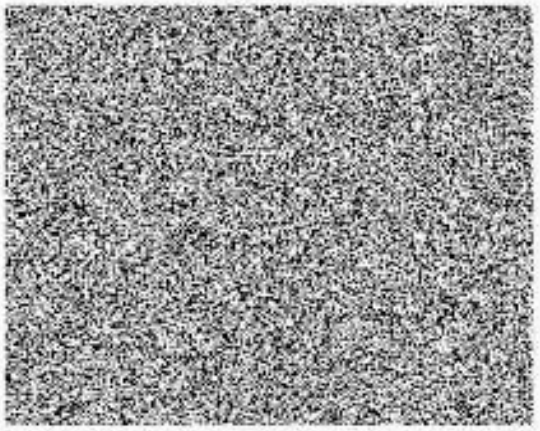
Image of bit8:
00000000



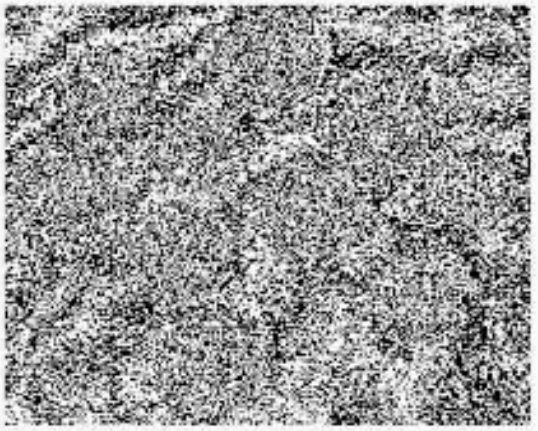
The n^th^ plane in the pixels are multiplied by the constant 2^n-1

Bit-plane Slicing Example

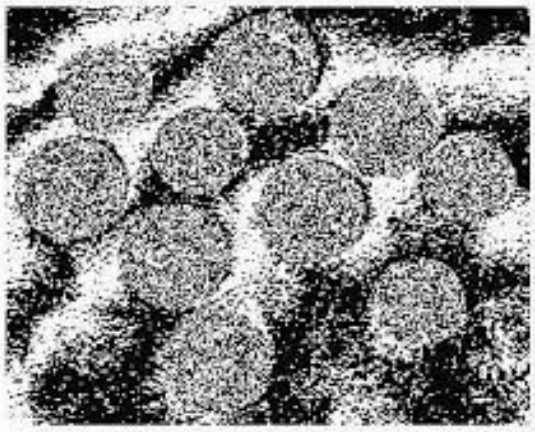
Bit plane 1



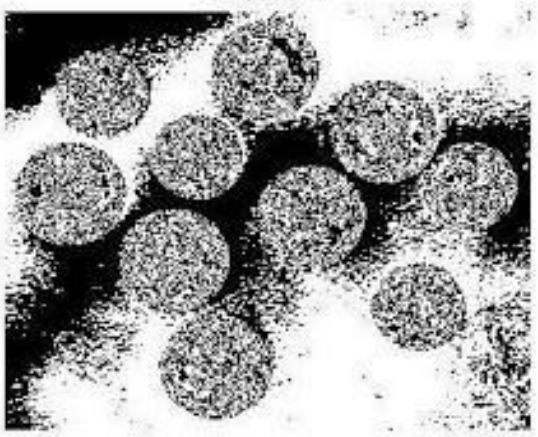
Bit plane 2



Bit plane 3

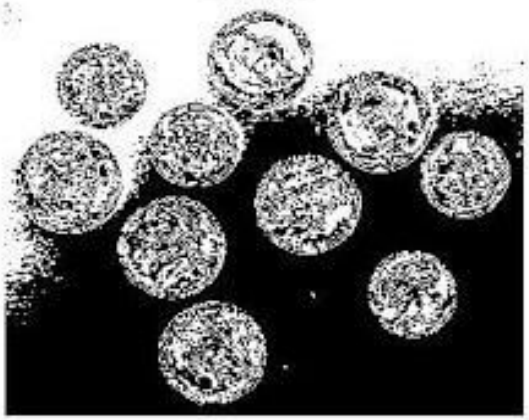


Bit plane 4

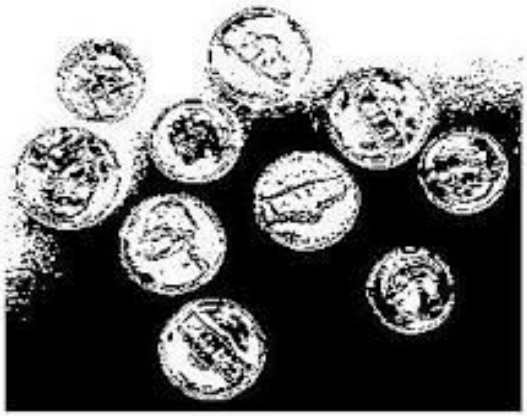


Bit-plane Slicing Example

Bit plane 5



Bit plane 6



Bit plane 7

Bit plane 8

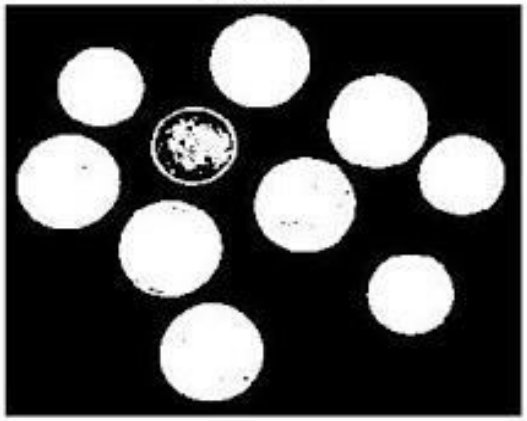


Image reconstruction using n bit planes.

1. The n^{th} plane in the pixels are multiplied by the constant 2^n-1
2. For instance, consider the matrix A below:

   A=[167 133 111
   144 140 135
   159 154 148]

and the respective bit format

10100111	10000101	01101111
10010000	10001100	10000111
10011111	10011010	10010100

3. Combine the 8 bit plane and 7 bit plane.

For 10100111, multiply the 8 bit plane with (2^7=)128 and 7 bit plane with (2^6=) 64.
(1x128)+(0x64)+(1x0)+(0x0)+(0x0)+(1x0)+(1x0)+(1x0)=128

4. Repeat this process for all the values in the matrix and the final result will be

   [128 128 64
   128 128 128
   128 128 128]

Image reconstruction using 7 and 8 bit planes.

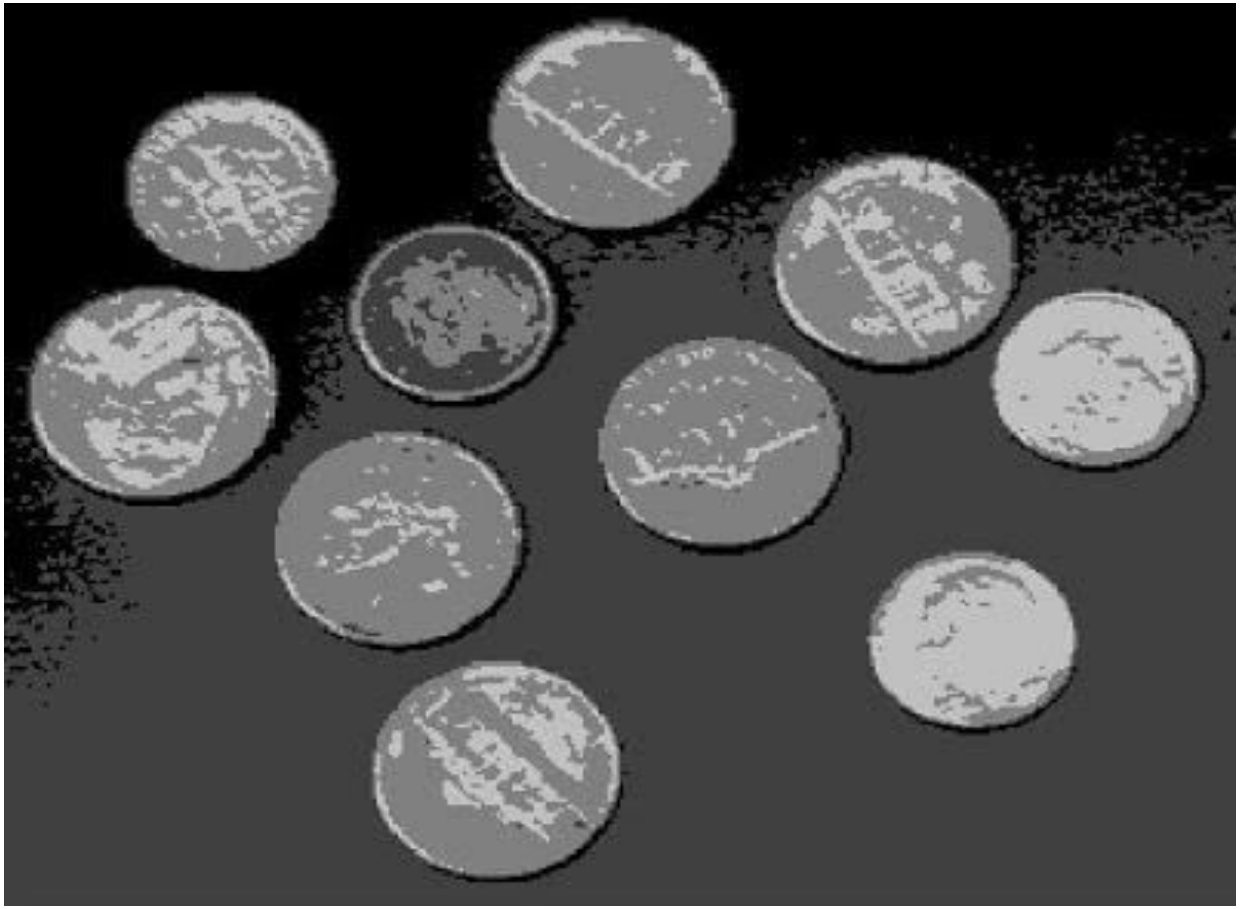


Image reconstruction using 5, 6, 7 and 8 bit planes.

Example

10100111	10000101	01101111
10010000	10001100	10000111
10011111	10011010	10010100

- Reconstruct an image using only 5^{th} and 4^{th} bit planes.
- Reconstruct an image using only 7^{th} and 6^{th} bit planes.

10100111	10000101	01101111
10010000	10001100	10000111
10011111	10011010	10010100

128	**128**	**64**
128	128	128
128	128	128

1	**1**	**1**
0	0	1
1	0	0

0	**0**	**1**
0	1	0
1	1	0

0	**0**	**1**
0	0	0
0	0	0

1	**0**	**1**
0	0	1
1	1	0

0	**0**	**0**
1	0	0
1	1	1

1	**1**	**0**
1	1	1
1	1	1

1	**1**	**1**
0	1	1
1	0	1

1	**0**	**1**
0	0	0
0	0	0

Class Work

Apply the following Image Enhancement techniques for the given 3-bits per pixels image segment.

- Digital Negative
- Bit planes
- Thresholding with T=3
- Intensity slicing with both approaches,
  when a =3 & b =5
  up the level to 5.

I=

2	1	2	1	0
7	1	4	3	2
2	4	1	3	7
1	3	4	6	3
1	4	1	3	4

Show your enhanced images.