Image Processing

Chapter(3)

Part 2:Intensity Transformation and spatial filters

Prepared by: Hanan Hardan

Image Enhancement?

☞ **Enhancement تحسين الصورة:** is to process an image so that the result is more suitable than the original image for a *specific* application.

- Enhancement techniques are so varied, and use so many different image processing approaches
- 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?

- If we used 256 intensities of grayscale then:

- In dark image the most of pixels value <128

- In bright image the most of pixels value>128

Let x:old image

S:new image

S=x+c where c is constant value

>> S=imadd(x,50); ----------brighter image

S=x-c where c is constant value

>> S=imsubtract(x,50); ----------darker image

- We can add 2 image :

S=x+y (where x and y 2 image have same size)

Image Enhancement Methods

- **Spatial Domain Methods (Image Plane)**

  Techniques are based on direct manipulation of pixels in an image

- **Frequency Domain Methods**

  Techniques are based on modifying the Fourier transform of the image.

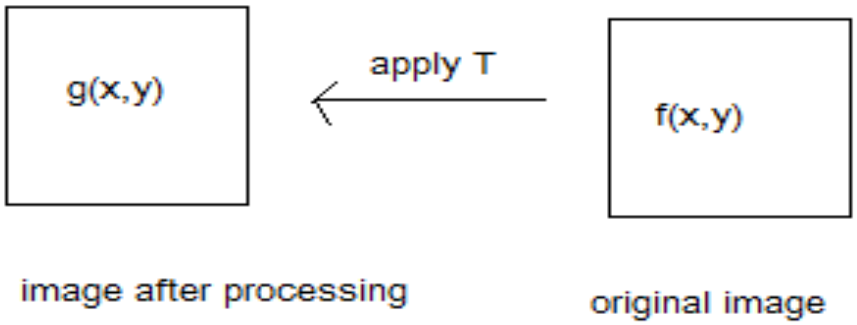
- **Combination Methods**

  There are some enhancement techniques based on various combinations of methods from the first two categories

☞ *In this chapter, we are going to discuss spatial domain techniques*

Spatial domain

- Spatial domain processes:



1. **intensity transformation**(Point operation)
   - g depends only on the value of f at(x,y)
1. **spatial filter** (or mask ,kernel, template or window)

Intensity (Gray-level)transformations functions

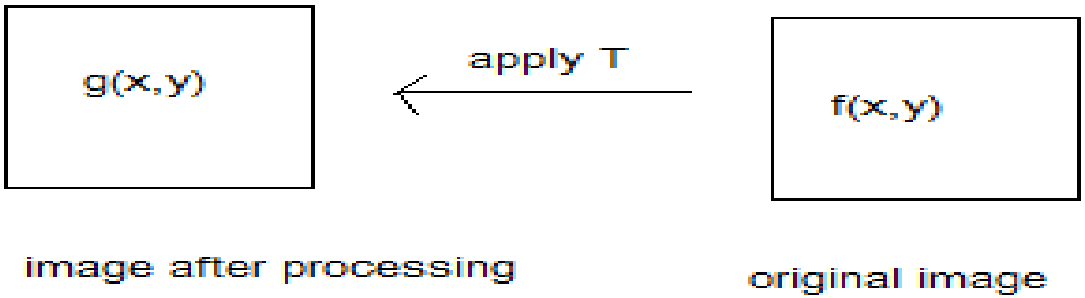
- Here, T is called intensity transformation function or (mapping, gray level function)

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

s= T(r)

s,r : denote the intensity of g and f at any point (x,y) .

In addition, ***T*** can operate on a set of input images



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

Intensity transformations functions

2) piecewise Linear transformation functions.

a) Contrast stretching, thresholding

b) Gray-level slicing

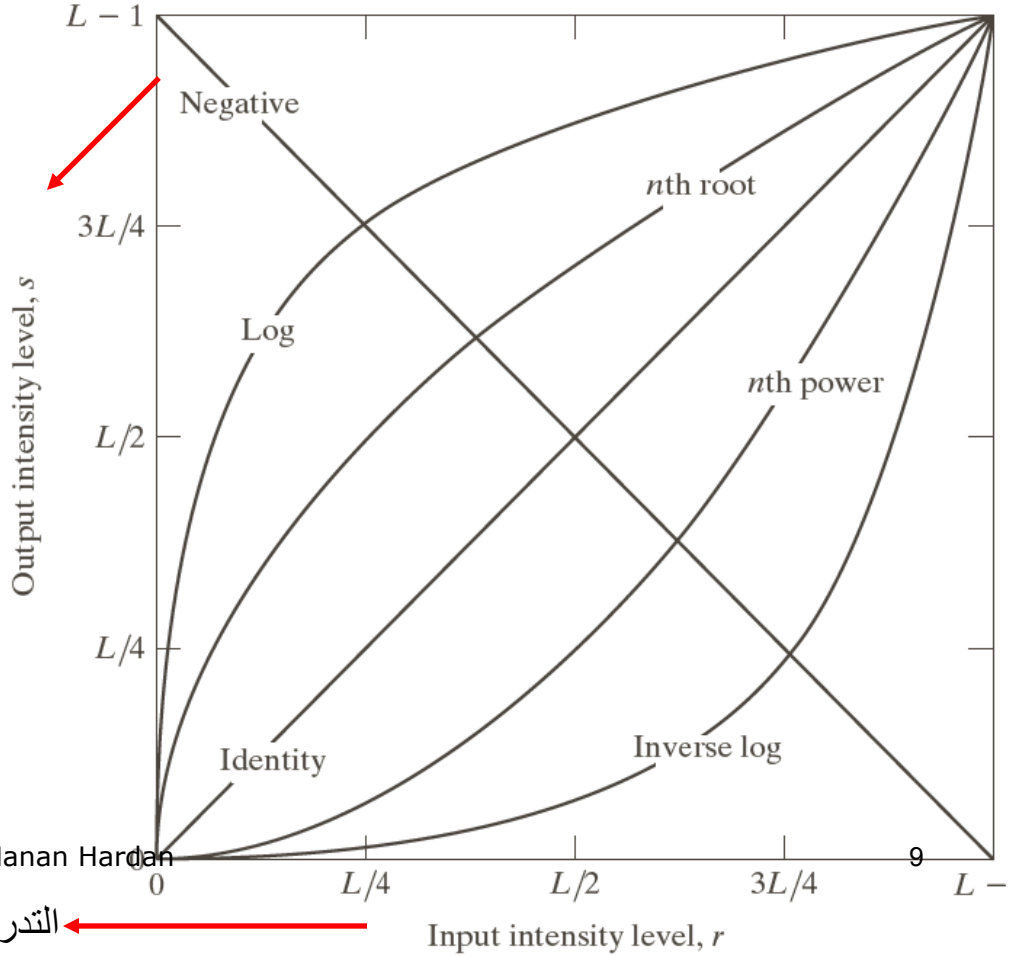
c) Bit-plane slicing

Basic intensity transformations

a) Linear (negative and identity).
b) logarithmic (Log and Inverse Log) .
c) Power(nth power and nth root

التدرجات اللونية gray levels بالصورة بعد التعديل

فمثلا, ان كان في البكسل 8 bit سيكون هناك 256 تدرج (L=256)



التدرجات اللونية gray levels بالصورة قبل التعديل

Identity Function

- 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

Image Negatives (Negative Transformation)

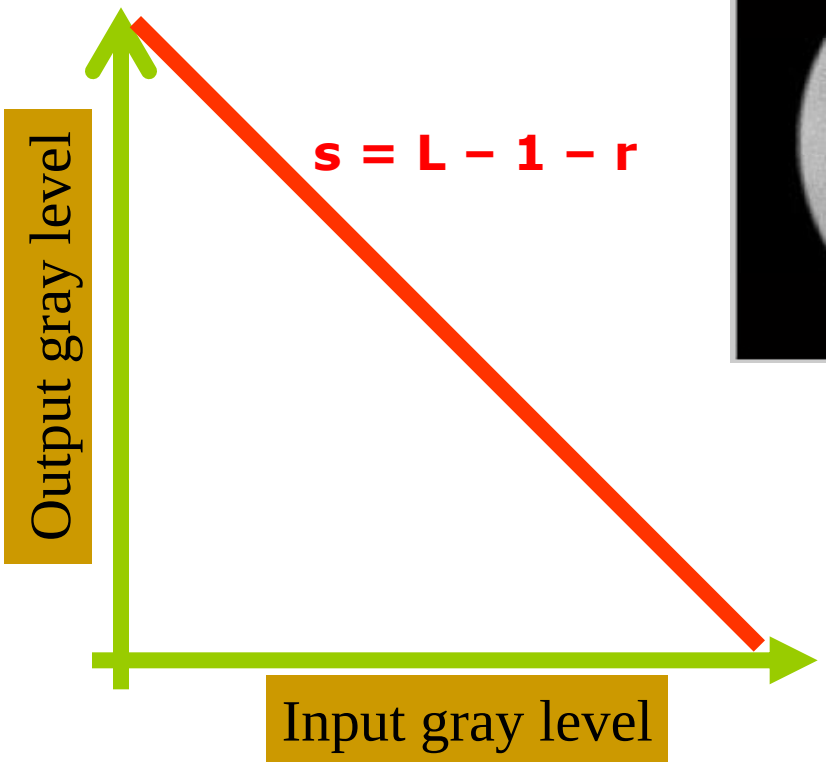
The negative of an image with gray level in the range [0, L-1], where L = Largest value in an image, is obtained by using the negative transformation's expression:

$$s = L - 1 - r$$

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

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

Image Negatives (Negative Transformation)



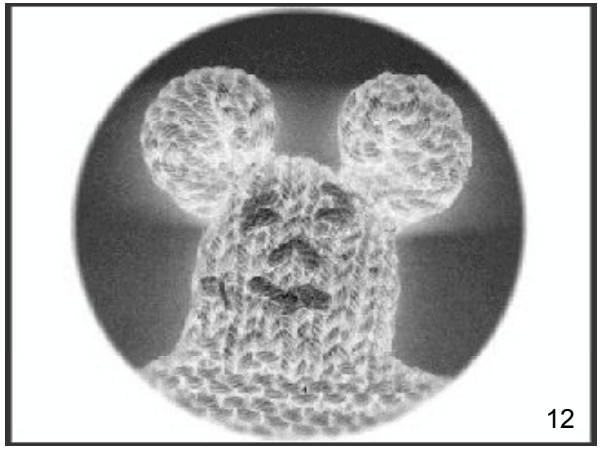


Image Negatives (Negative Transformation)

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.

Image Negatives (Negative Transformation)

Note how much clearer the tissue is in the negative image

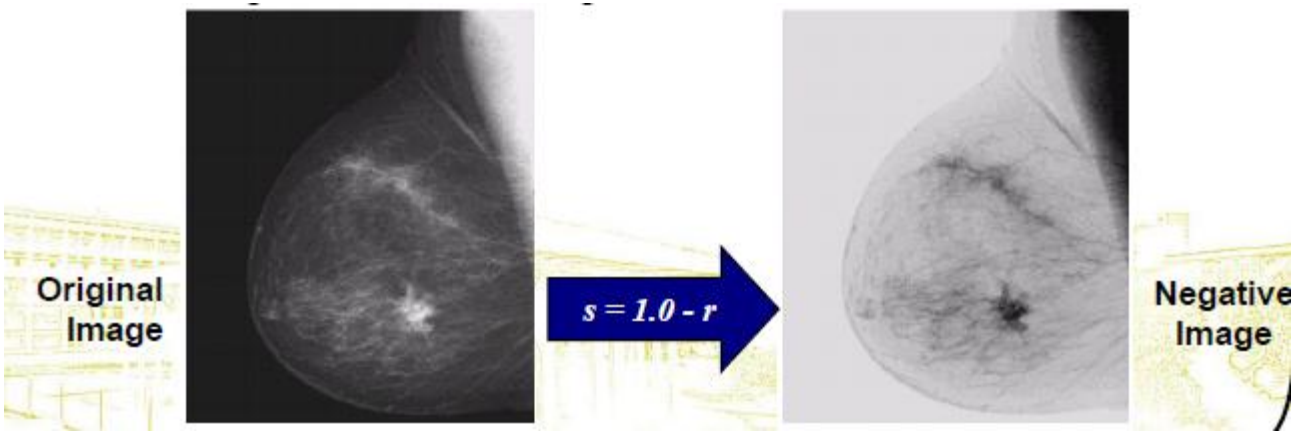


Image Negatives (Negative Transformation)

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)

Image Negatives (Negative Transformation)

The negative of an image can be obtained also with IPT function imcomplement:

g = imcomplement (f);

Log Transformations function

The general form of the log transformation:

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

Where c is a constant, and $r \geq 0$

- Log curve maps a narrow range of low gray-level values in the input image into a wider range of the output levels.
- Used to expand the values of dark pixels in an image while compressing the higher-level values.
- 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 Transformations function

- Logarithmic transformations are implemented using the expression:

  g = c * log (1 + double (f))

Log Transformations function

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

gs = im2uint8 (mat2gray(g));

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

Log Transformations function

- **Example:**

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

f

g

gs

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.

Logarithmic Transformations

InvLog

Log

Power-Law Transformations

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

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

  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

Power-Law Transformations

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

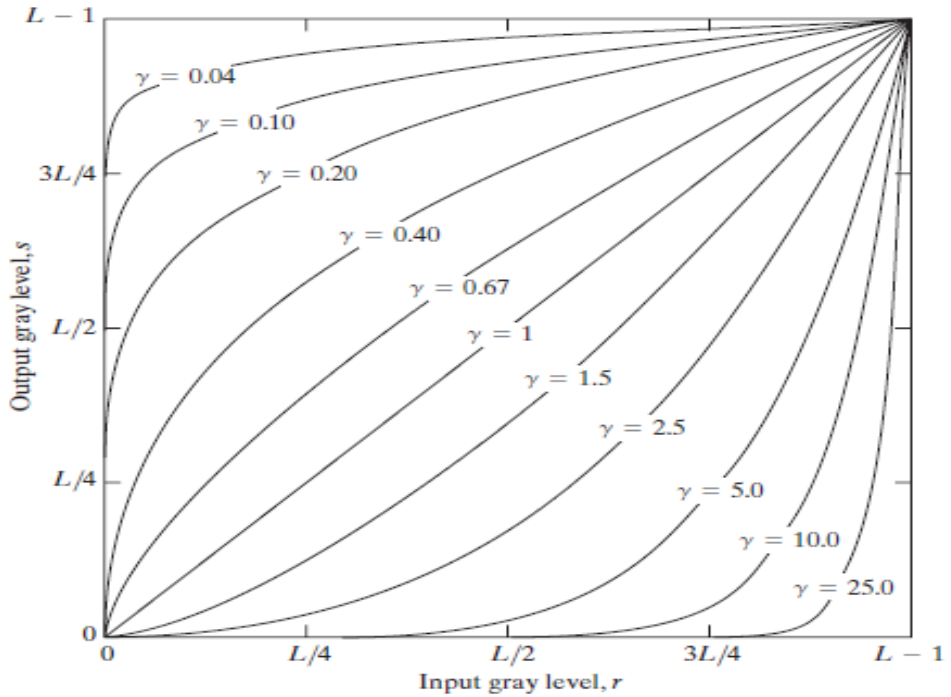


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

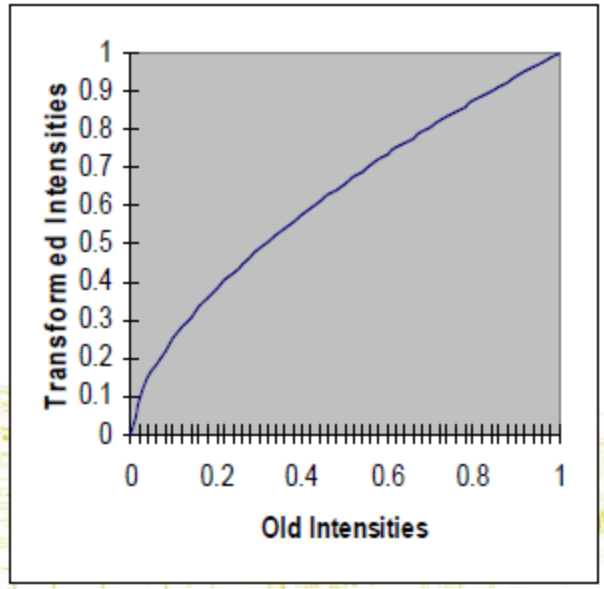
Power-Law Transformations

- 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.

Power Law Example

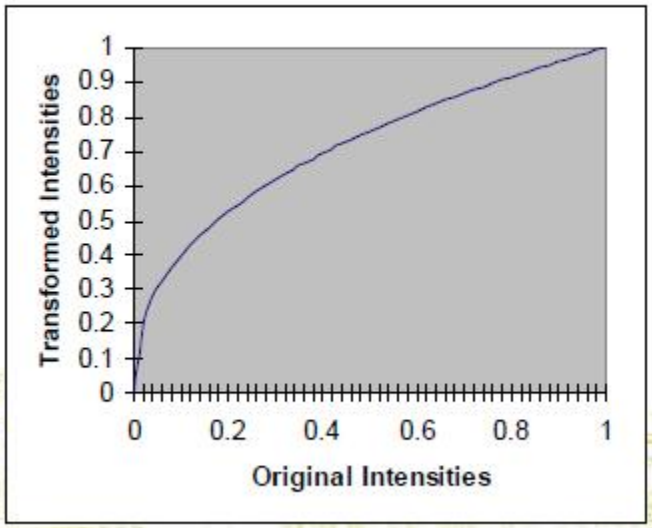
Power Law Example

$\gamma = 0.6$

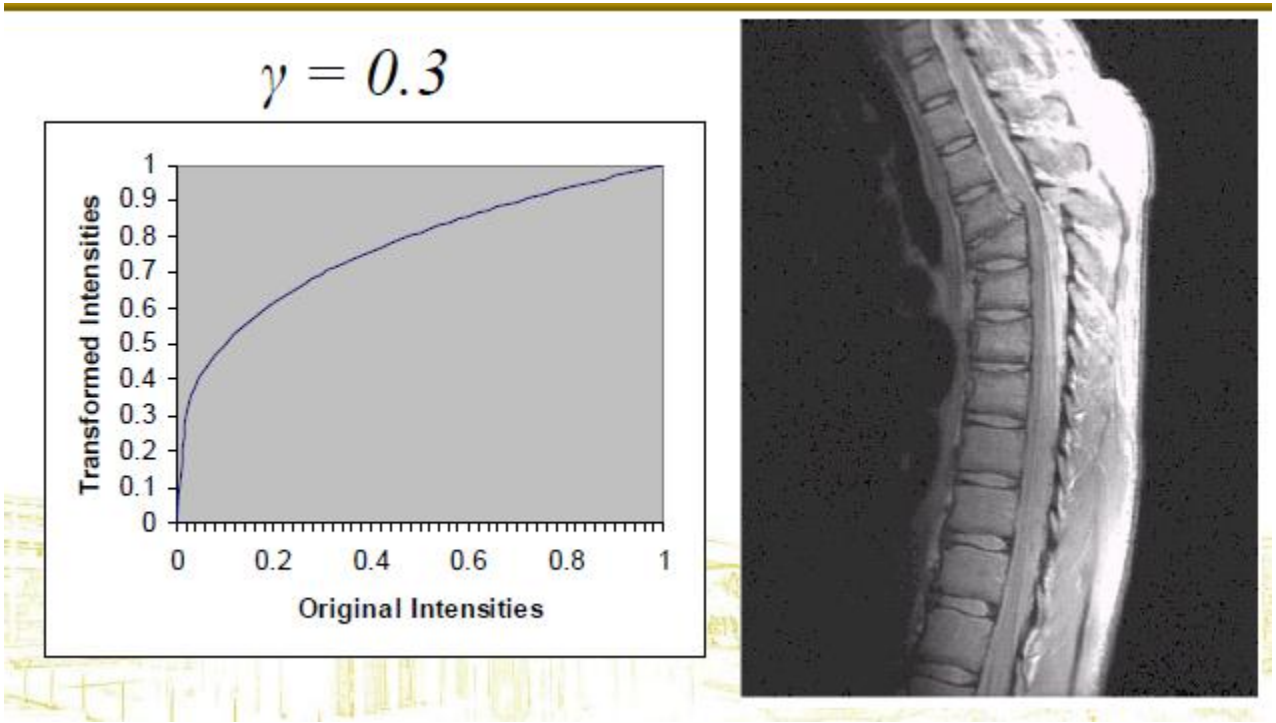


Power Law Example

$\gamma = 0.4$

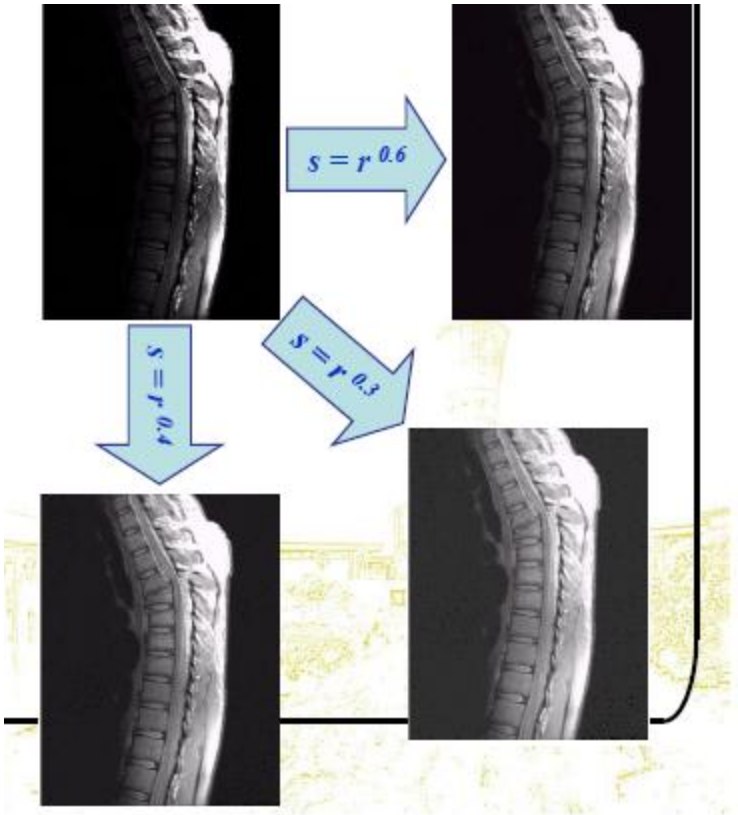


Power Law Example



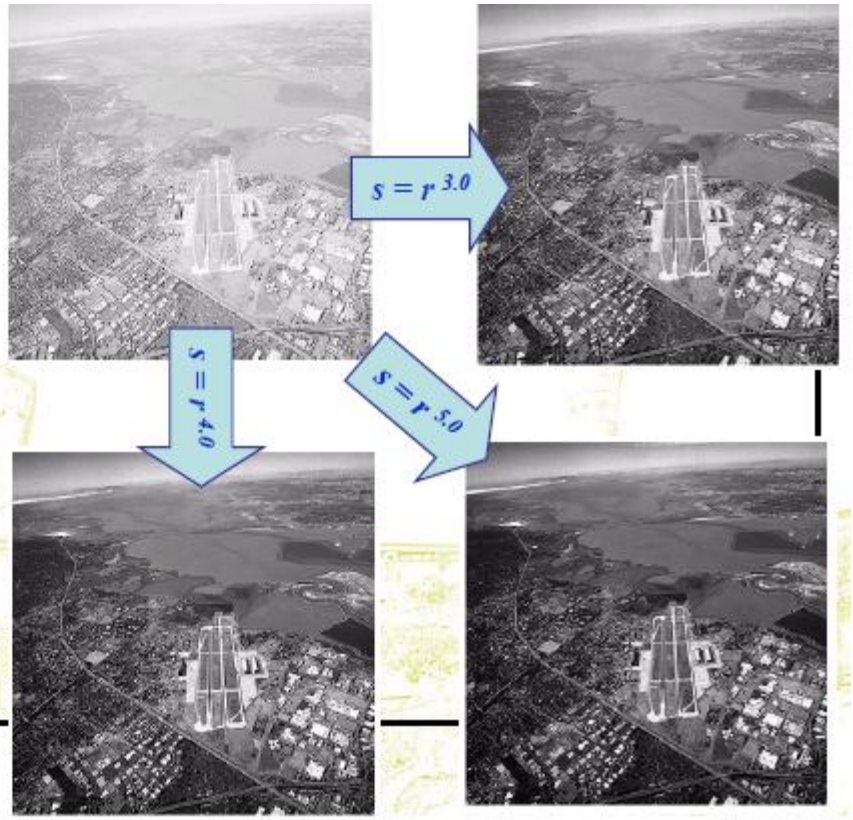
Power Law Example

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



Power Law Example

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

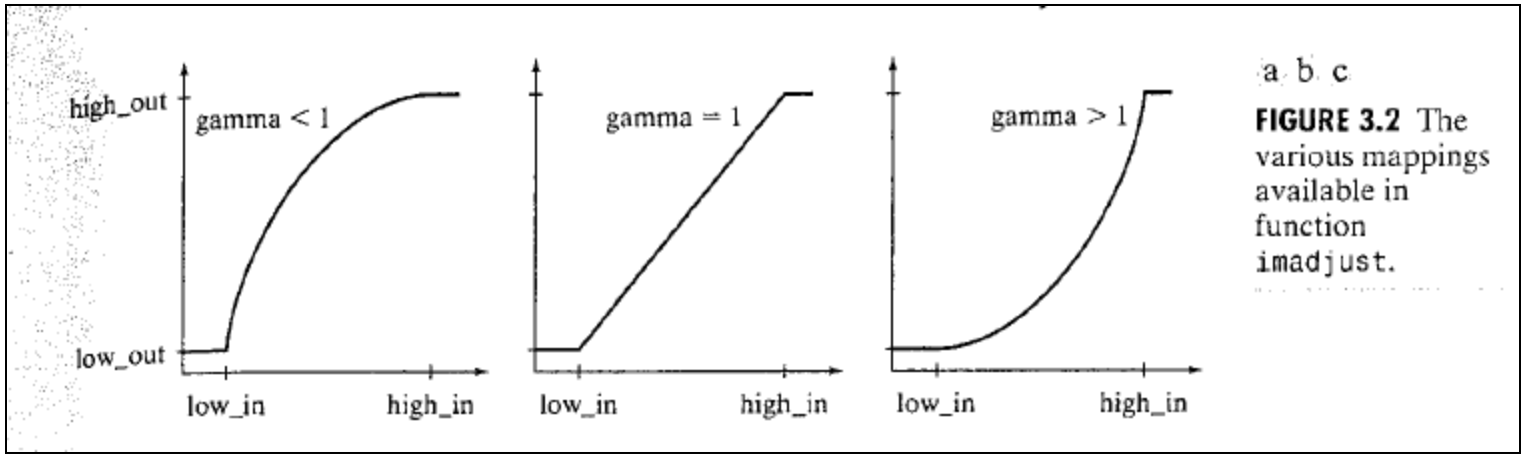


Function imadjust

- 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)

Function imadjust



a b c

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

- As illustrated in figure 3.2 (above), 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**.

Function imadjust

- The input image can be of class uint8, uint16, or double, and the output image has the same class as the input.

- All inputs to function **imadjust**, other than **f**, are specified as values between 0 and 1, regardless of the class of **f**. If **f** is of class uint8, imadjust multiplies the value supplied by 255 to determine the actual values to use; if **f** is of class uint16, the values are multiplied by 65535.

- Using the empty matrix ([ ]) for [**low_in high_in**] of for [**low_out high_out**] results in the default values [0 1].

- If **high_out** is less than **low_out**, the output intensity is reversed.

Function imadjust

- Parameter **gamma** specifies the shape of the curve that maps the intensity values of **f** to create **g**. If **gamma** is less than 1, the mapping is weighted toward higher (brighter) output values, as fig 3.2 (a) shows. If **gamma** is greater than 1, the mapping is weighted toward lower (darker) output values. If it is omitted from the function arguments, **gamma** defaults to 1 (linear mapping).

Function imadjust

- **Example1:**

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

This Obtaining the negative image

f

g

Function imadjust

- **Example2:**

```
>> g = imadjust (f, [0.5 0.75], [0 1], .5);
>> imshow(f), figure, imshow (g);
```

f

g

Function imadjust

- **Example3:**

```
>> g = imadjust (f, [0.5 0.75], [0.6 1], 0.5);
>> imshow(f), figure, imshow (g);
```

f

g

Function imadjust

- **Example4:**

```
>> g = imadjust (f, [ ], [ ], 2);
>> imshow(f), figure, imshow (g);
```

f

g

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 that previous transformations
- **Types of Piecewise transformations are:**
  - Contrast Stretching
  - 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.

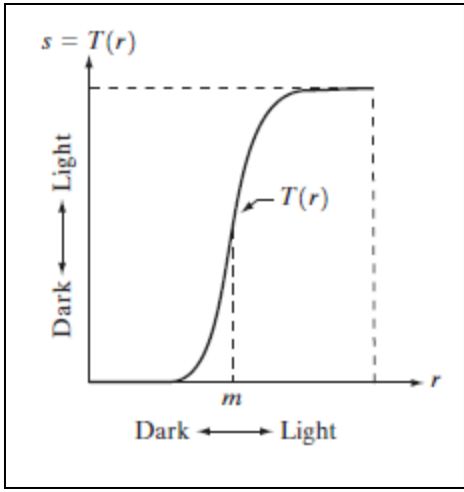
Contrast Stretching

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:

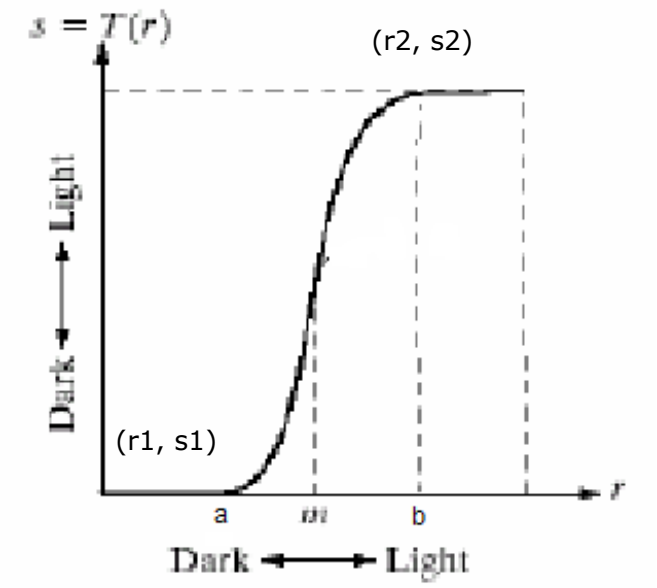
Produce higher contrast than the original image, by:

- 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



Contrast stretching

Assume that
a: rmin,
b:rmax,

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

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

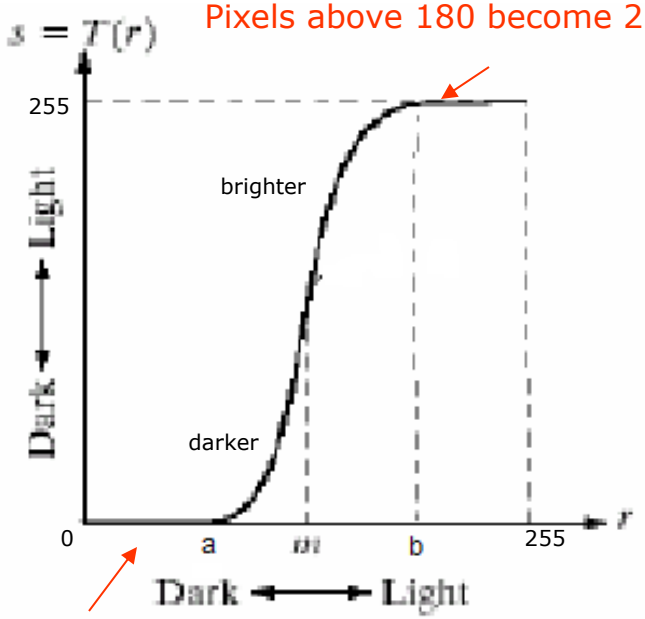
✓if r is above 180 ,it becomes 255 in s.

✓If r is below 90 , it becomes 0,

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

when r < 100 , s closes تقترب 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 Transformation

- The function takes the form of:

$$s = T(r) = \frac{1}{1 + (m/r)^E}$$

Where *r* represents the intensities of the input image, *s* the corresponding intensity values in the output image, and *E* controls the slope of the function.

Contrast Stretching

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 Transformation

- 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 Transformation

- **Example1:**

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

Contrast-Stretching Transformation

- **Example2:**

```
>> g = 1 ./ (1+ (50 ./(double(f) + eps)) .^ 20);
>> imshow(f), figure, imshow(g);
```

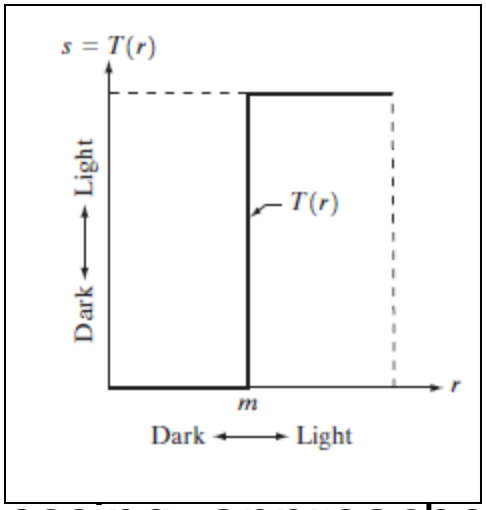
Contrast-Stretching Transformation

- **Example3:**

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

Thresholding

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

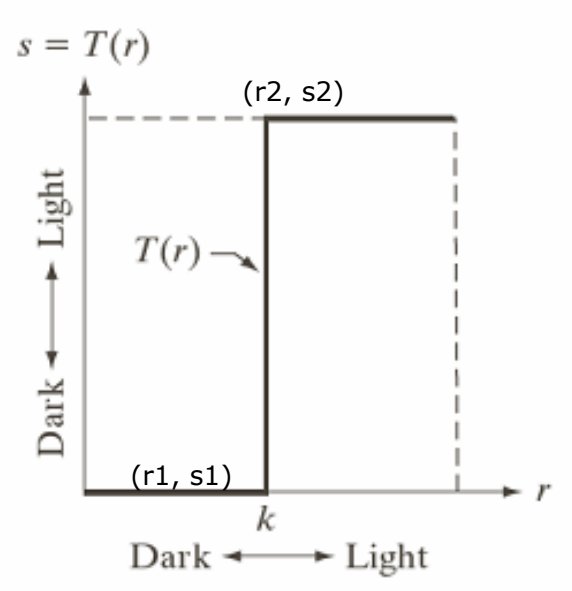


Some fairly simple, yet powerful, processing approaches can be formulated with grey-level transformations. Because enhancement at any point in an image depends only on the gray level at that point, techniques in this category often are referred to as *point processing*.

Thresholding

Assume that
a: rmin,
b:rmax,
k : intensity

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



Thresholding:

Thresholding

Remember that:

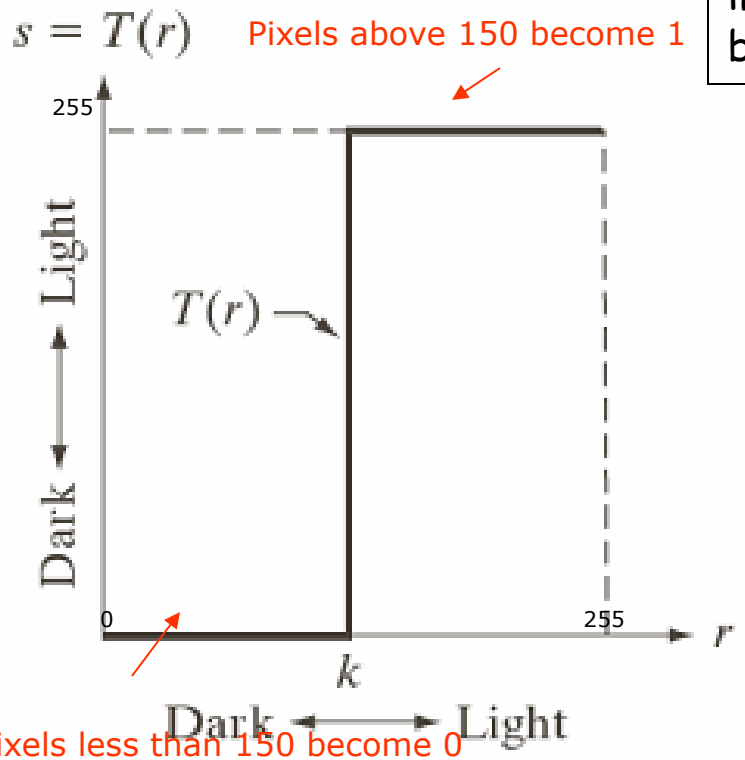
$g(x,y) = T[f(x,y)]$

Or

$s = T(r)$

Example: suppose m= 150 (called threshold),

if r (or pixel intensity in image f الصورة الاصلية) is above this threshold it becomes 1 in s (or pixel intensity in image g الصورة بعد التعديل), 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!

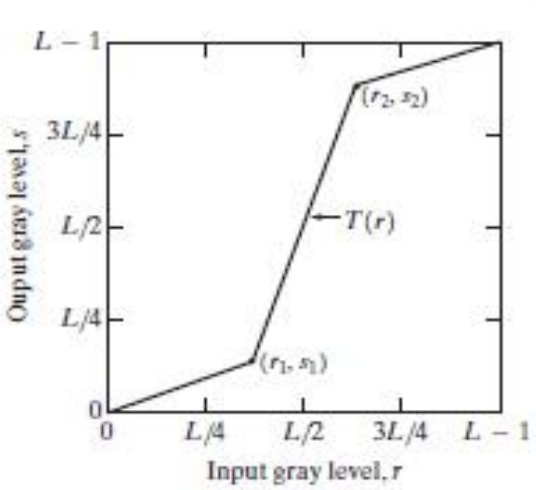
Thresholding

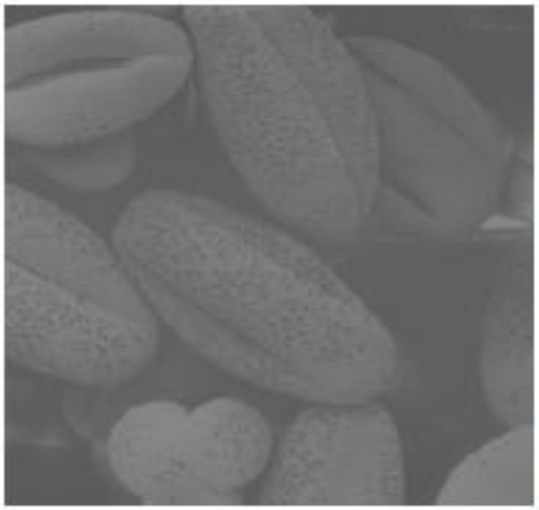
Image (r)

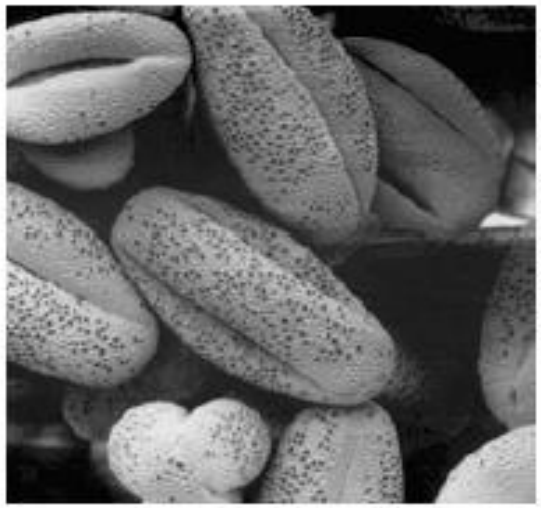
Image (s) after applying T (Thresholding)

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.

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.)

Application on Contrast stretching and thresholding

8-bit image with low contrast

After contrast stretching

(r1,s1)=(r_{min},0) , (r2,s2)=(r_{max},L-1)

Thresholding function

(r1,s1)=(m,0) , (r2,s2)=(m,L-1)

m : mean intensity level in the image

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.

piecewise Linear transformation functions.

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.

Image (r)

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

solution:

Image (s)

solution in matlab:

```
function a2(x,s)

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);
```