DSIP – Lecturer 06

Image Enhancement in the Spatial Domain

Basic Relationships Between Pixels

- Neighborhood
- Adjacency
- Connectivity
- Paths
- Regions and boundaries

Contents

- Relationship between Pixels
- Distance Measures Operations
- Practice Examples

Image Enhancement :

- **Image Enhancement:** is the process that improves the quality of the image for a specific application

- Image enhancement:

1. Improving the interpretability or perception of information in images for human viewers
2. Providing `better' input for other automated image processing techniques.

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

Spatial Domain Methods

- The term *spatial domain* refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels.
- Spatial domain processes will be denoted by the expression:

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

  Where f(x,y) in the input image, g(x,y) is the processed image and ***T*** is as operator on f, defined over some neighborhood of (x,y)

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

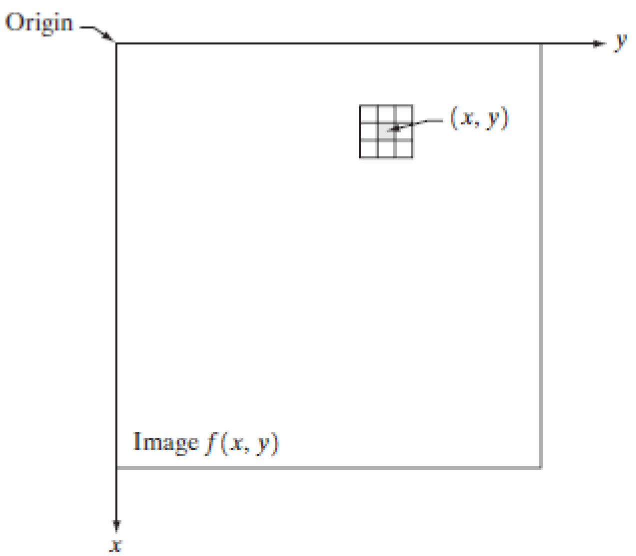


FIGURE 3.1 A 3 × 3 neighborhood about a point (x, y) in an image.

- The simplest form of ***T***, is when the neighborhood of size 1X1 (that is a single pixel).
- In this case, g depends only on the value of f at (x,y), and ***T*** becomes a *grey-level* (also called *intensity* or *mapping*) *transformation function* of the form:

$$s = T\,(r)$$

Where, r and s are variables denoting the grey levels of f(x,y) and g(x,y) at any point (x,y).

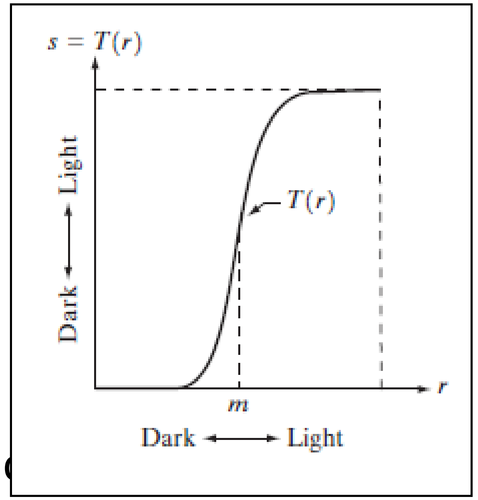
Examples of Enhancement Techniques :

- **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

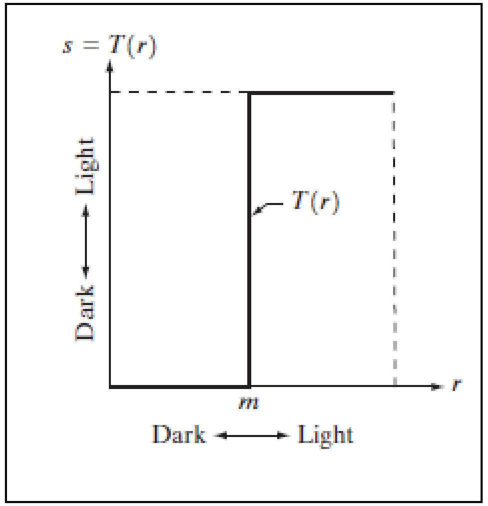
  in an image by 'stretching' the range of intensity values it
  desired range of values.

Typically, it uses a linear function

Examples of Enhancement Techniques

- **Thresholding**

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



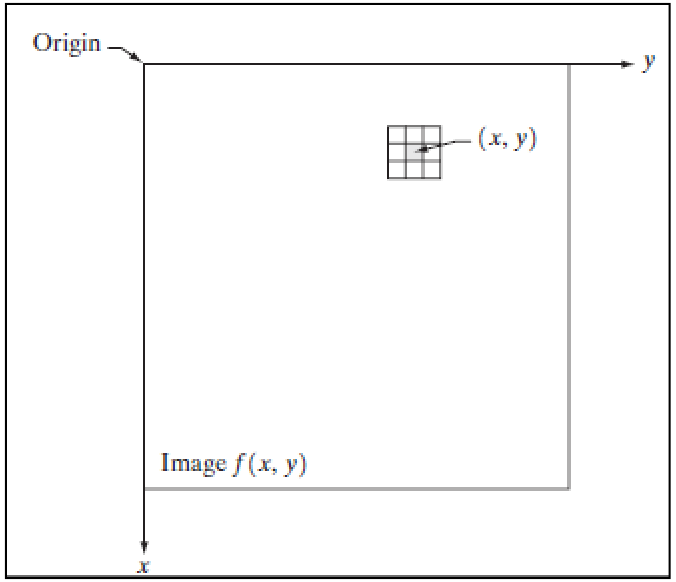
Some fairly simple, yet powerful, processing approaches can be formulated with grey-level transformations. Because enhancement at any point in an image depends only on the gray level at that point, techniques in this category often are referred to as *point processing*.

Examples of Enhancement Techniques

Larger neighborhoods allow considerable more flexibility. The general approach is to use a function of the values of *f* in a predefined neighborhood of (x,y) to determine the value of *g* at (x,y).

One of the principal approaches in this formulation is based on the use of so-called *masks* (also referred to as *filters*)

So, a mask/filter: is a small (say 3X3) 2-D array, such as the one shown in the figure, in which the values of the mask coefficients determine the nature of the process, such as *image sharpening*. Enhancement techniques based on this type of approach often are referred to as *mask processing* or *filtering*.



Some Basic Intensity (Gray-level) Transformation Functions

- Grey-level transformation functions (also called, intensity functions), are considered the simplest of all image enhancement techniques.
- The value of pixels, before an after processing, will be denoted by *r* and *s*, respectively. These values are related by the expression of the form:

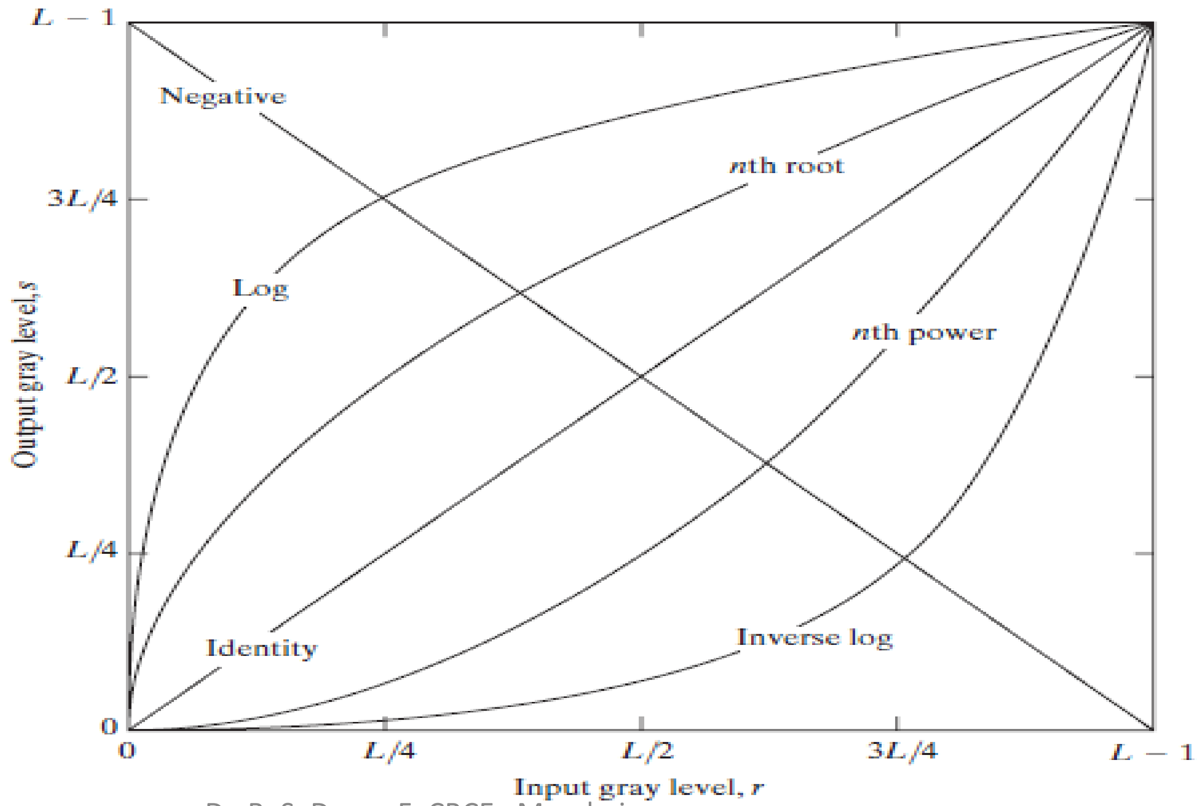
$$s = T\,(r)$$

where T is a transformation that maps a pixel value r into a pixel value s.

Some Basic Intensity (Gray-level) Transformation Functions

Consider the following figure, which shows three basic types of functions used frequently for image enhancement:

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



Some Basic Intensity (Gray-level) Transformation Functions

- The three basic types of functions used frequently for image enhancement:
  - Linear Functions:
    - Negative Transformation
    - Identity Transformation
  - Logarithmic Functions:
    - Log Transformation
    - Inverse-log Transformation
  - Power-Law Functions:
    - n^{th} power transformation
    - n^{th} root transformation

Linear Functions

- **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:

    $$s = r$$

Linear Functions

- **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

Logarithmic Transformations

- **Log Transformation**

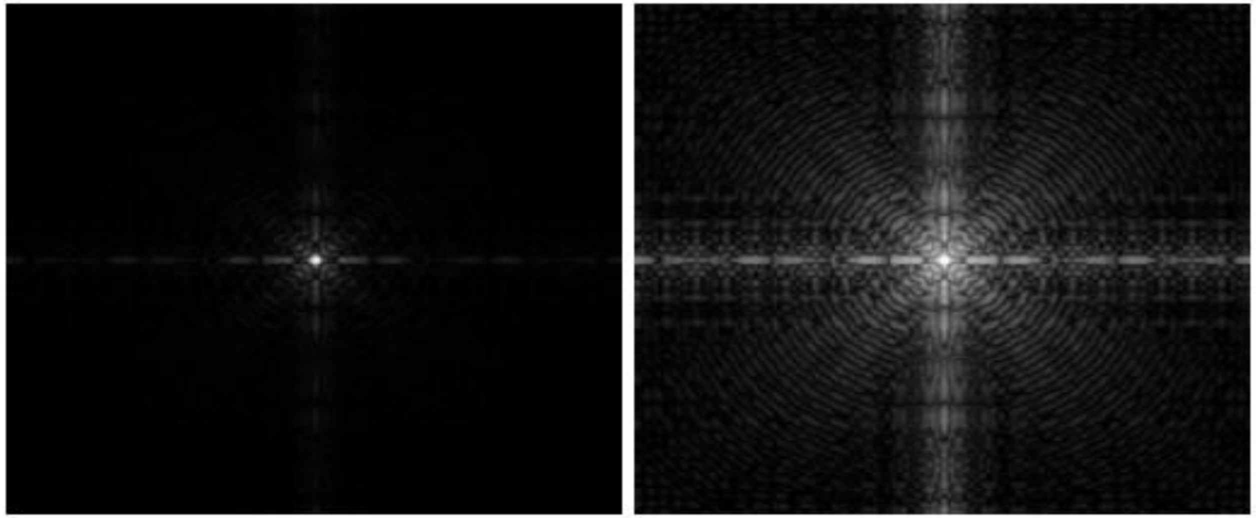
  The general form of the log transformation:

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

  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.

Logarithmic Transformations



a b

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

Logarithmic Transformations

- **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 Transformations

- Power-law transformations have the basic form of:

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

  Where c and γ are positive constants

Power-Law Transformations

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

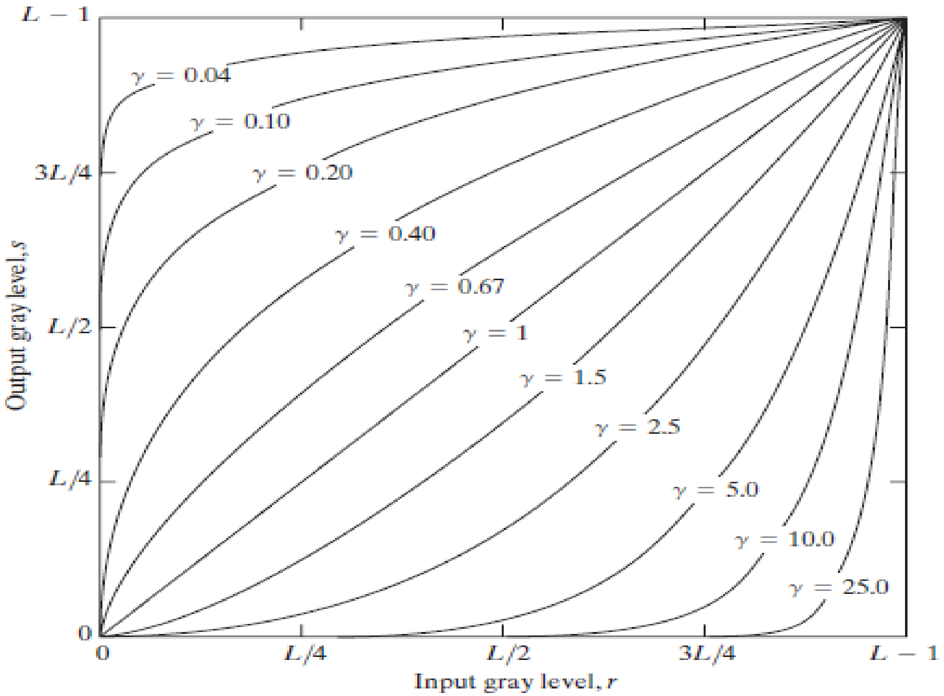


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

Power-Law Transformations

- Variety of devices used for image capture, printing and display respond according to a power law. The process used to correct this power-law response phenomena is called ***gamma correction***.

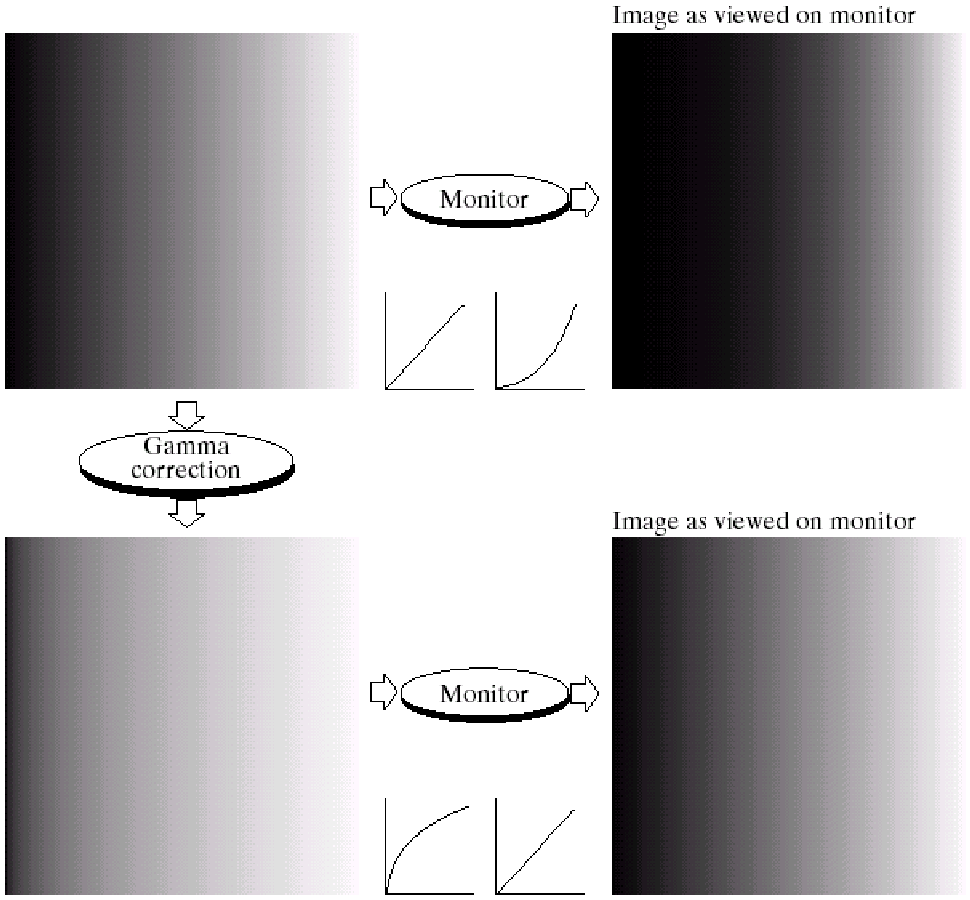
Why power laws are popular?

- A cathode ray tube (CRT), for example, converts a video signal to light in a nonlinear way. The light intensity *I* is proportional to a power (γ) of the source voltage *VS*
- For a computer CRT, γ is about 2.2
- Viewing images properly on monitors requires γ-correction

Gamma Correction

a b
c d

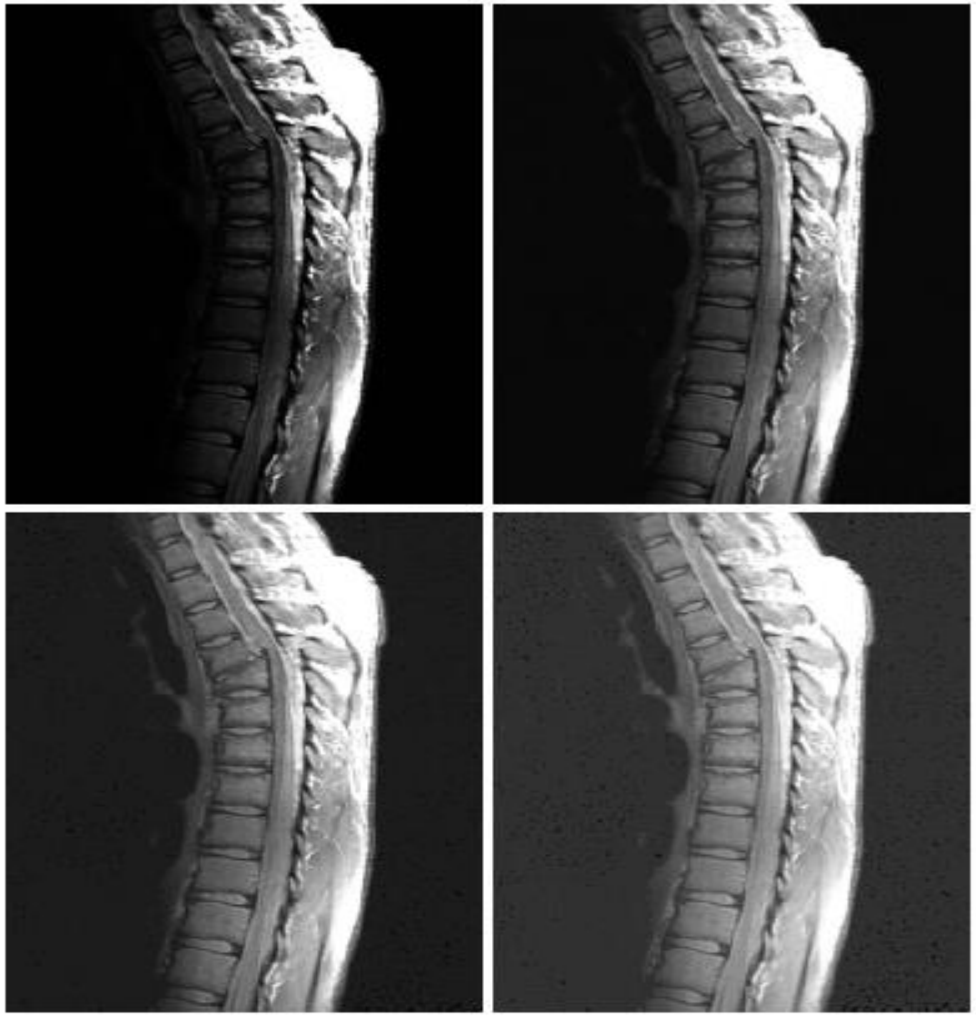
FIGURE 3.7 (a) Linear-wedge gray-scale image. (b) Response of monitor to linear wedge. (c) Gamma-corrected wedge. (d) Output of monitor.



Gamma Measuring Applet:
http://www.cs.cmu.edu/~efros/java/gamma/gamma.html

Power-Law Transformation

- In addition to gamma correction, power-law transformations are useful for general-purpose contrast manipulation.



a b
c d

FIGURE 3.8 (a) Magnetic resonance (MR) image of a fractured human spine. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4,$ and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Power-Law Transformation

- Another illustration of Power-law transformation

a b
c d

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

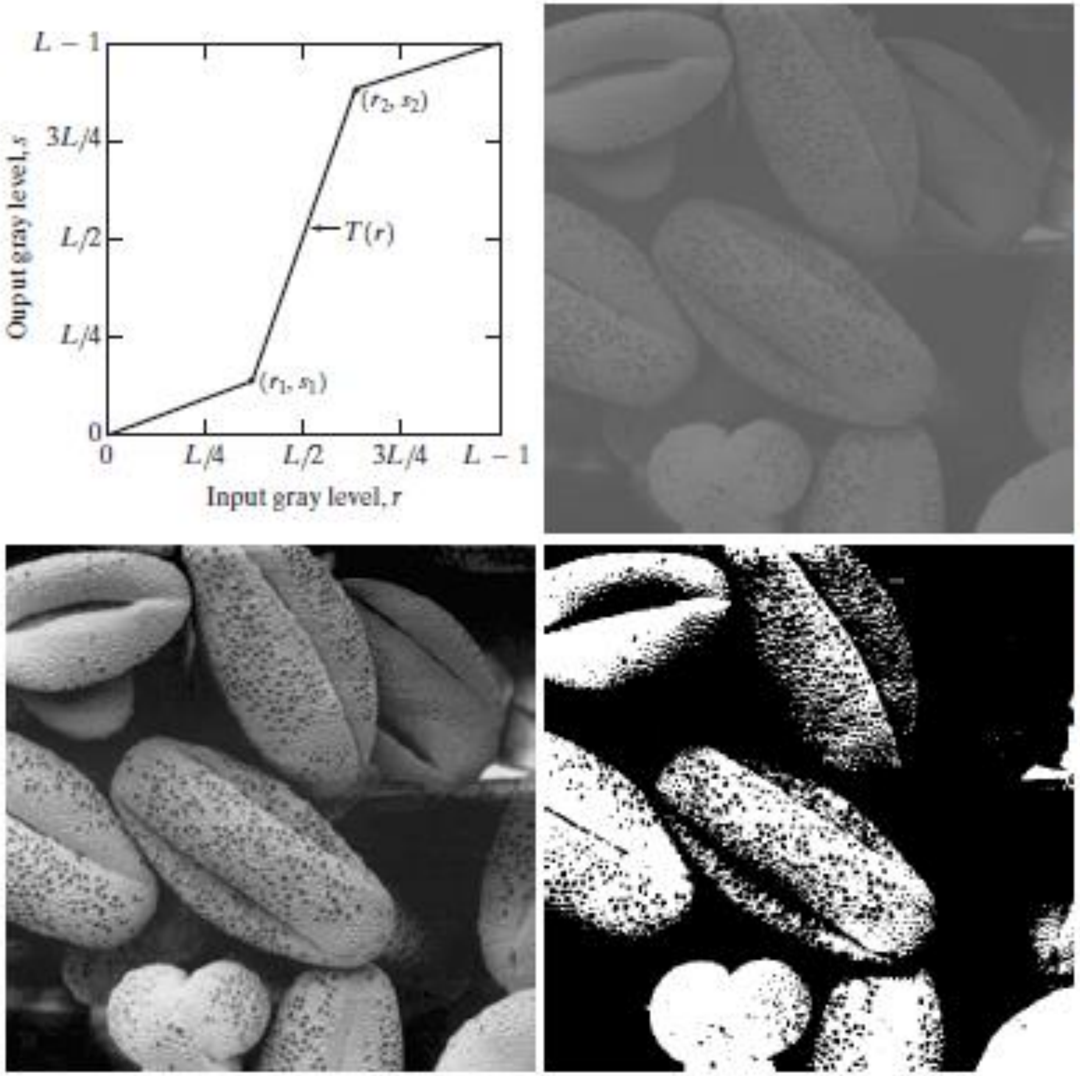
Piecewise-Linear Transformation Functions

- **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



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.)

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. As shown previously.
- Intermediate values of (r1, s1) 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.

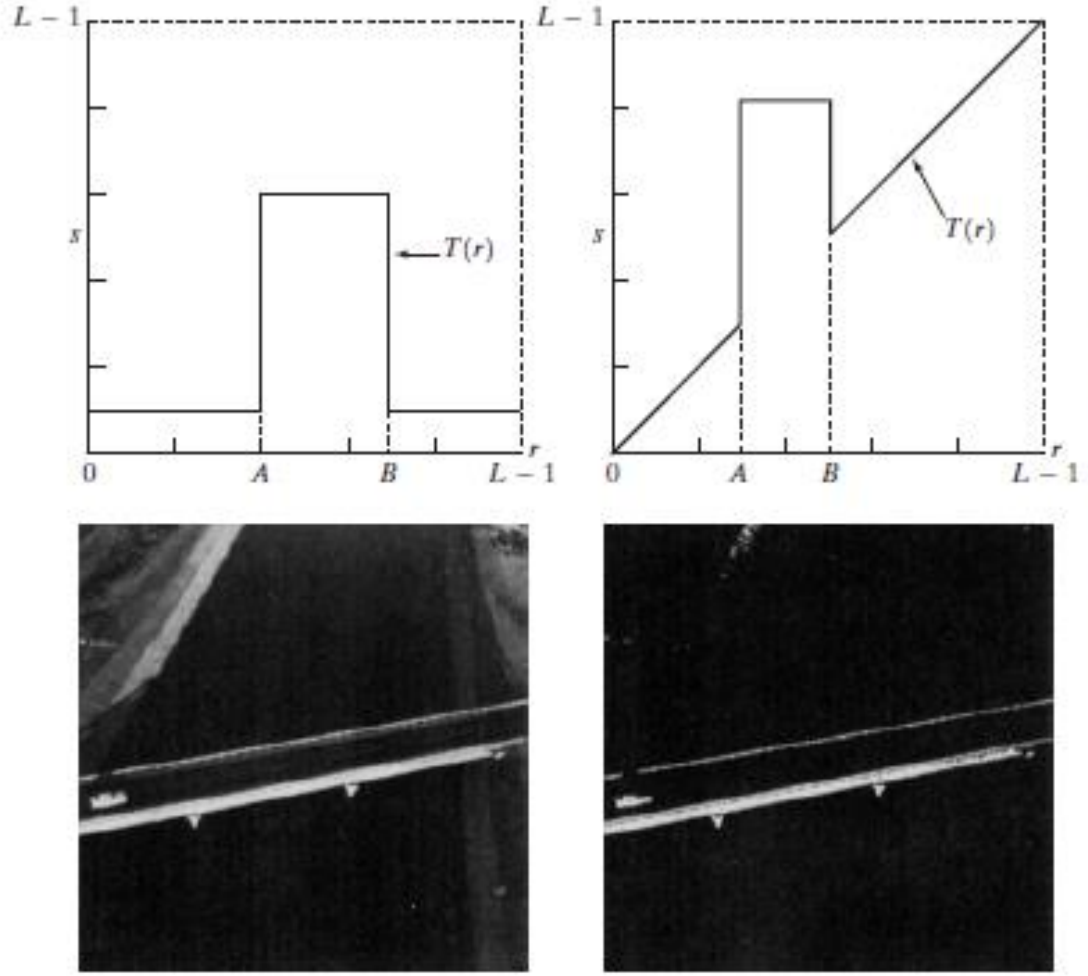
- Fig. 3.10(c) shows the result of contrast stretching, obtained by setting (r1, s1) = (r_{min}, 0) and (r2, s2) = (r_{max},L-1) where r_{min} and r_{max} denote the minimum and maximum gray levels in the image, respectively. Thus, the transformation function stretched the levels linearly from their 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.

Gray-level Slicing

- This technique is used to highlight a specific range of gray levels in a given image. It can be implemented in several ways, but the two basic themes 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. This transformation, shown in Fig 3.11 (a), produces a binary image.
  - The second approach, based on the transformation shown in Fig 3.11 (b), brightens the desired range of gray levels but preserves gray levels unchanged.
  - Fig 3.11 (c) shows a gray scale image, and fig 3.11 (d) shows the result of using the transformation in Fig 3.11 (a).

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).

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.

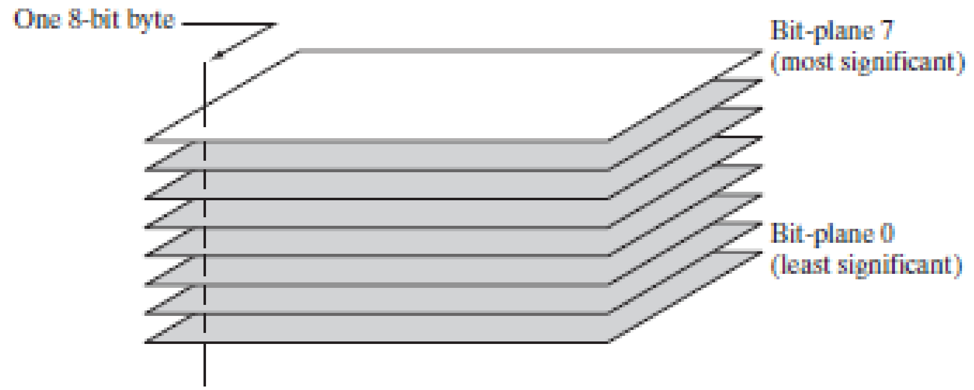


FIGURE 3.12 Bit-plane representation of an 8-bit image.

- Most significant bits contain the majority of visually significant data.

HW :

Assignment

- Write an M-function, that
  - inputs an image, a low value, a high value and a range of gray levels of interest

    (f, low, high, from, to)
  - Outputs a two images (g, h)
  - Applies the two approaches of Gray-level slicing on image f, and produces images g and h respectively.