Image Enhancement

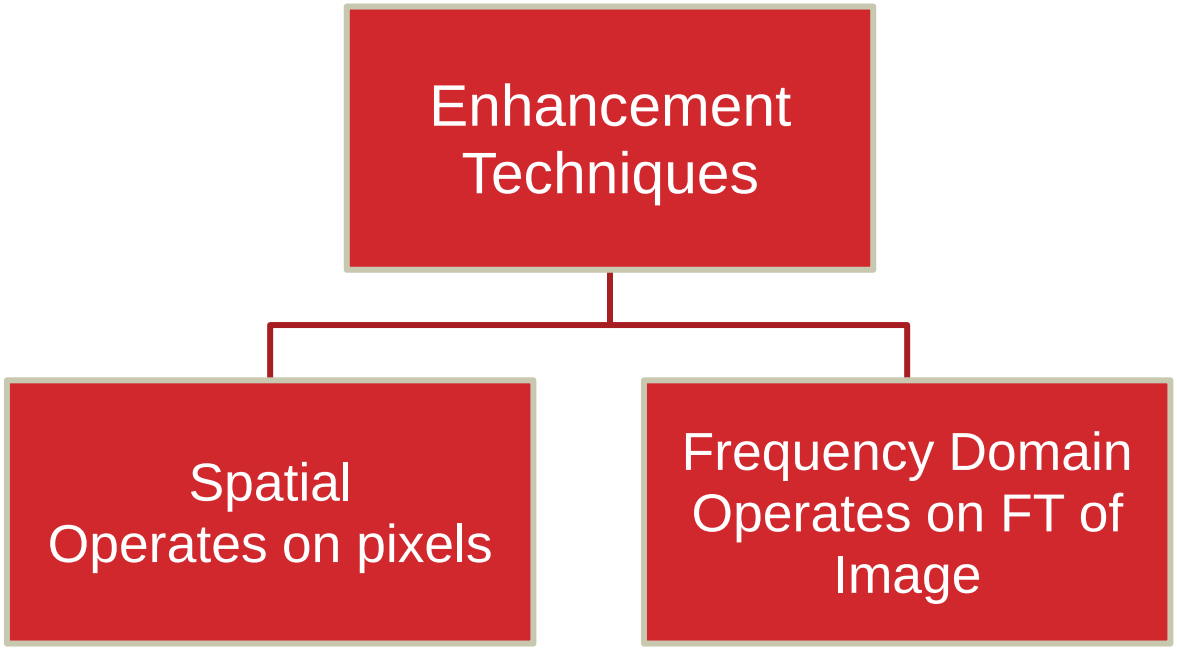


Image Enhancement in the Spatial Domain

Objective of Enhancement :

- To “improve” the usefulness of an image by using some transformation on the image.
- Often the improvement is to make the image “better” looking, such as increasing the intensity or contrast.

The spatial domain: Spatial means the plane of image itself. So in spatial domain is based on direct manipulation of pixels of an image. Term spatial refers to the aggregate of pixel composing an image.

The frequency domain : Frequency domain processing techniques are based on manipulation in Fourier transform of image.

- A (2-dimensional) discrete Fourier transform of the spatial domain.

Combination Methods

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

Mathematically Analysis of Enhancement on Spatial Domain

A mathematical representation of spatial domain enhancement:

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

where *f*(*x*, *y*): the input image

***g*(*x*, *y*): the processed image**

***T*: an operator on *f*, defined over some neighborhood of (*x*, *y*)**

Basic intensity transformation functions

- **Intensity transformations are among the simplest of all image processing technique.**
- **The values of pixels before and after processing will be denoted by r and s respectively**

$$s = T(r)$$

- **When T is a transformation that maps a pixel value r into a pixel value s.**
- **Basic types of functions for image enhancement are**

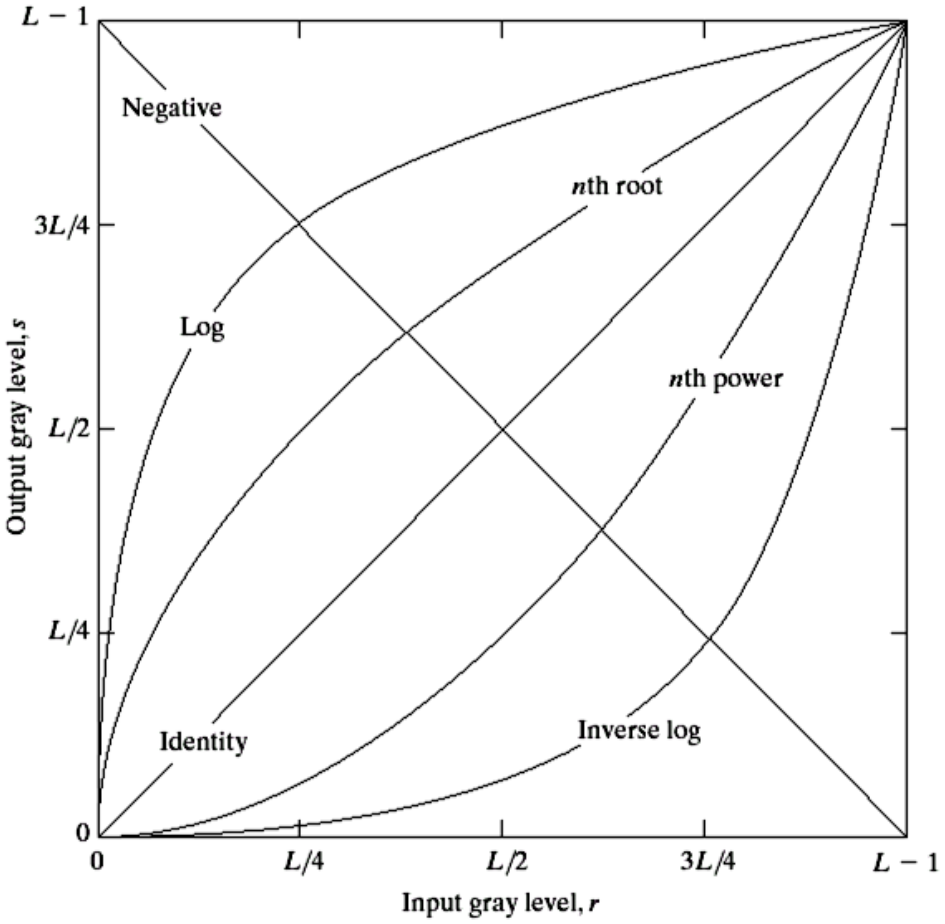
1. Linear { negative and identity transformation}
2. Logarithmic (log and inverse log transformation}
3. Power law (nth power and nth root transformation}
4. Piecewise Linear Transformation Functions

BASIC GREY LEVEL TRANSFORMATIONS

There are many different kinds of grey level transformations

Three of the most common are shown here

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



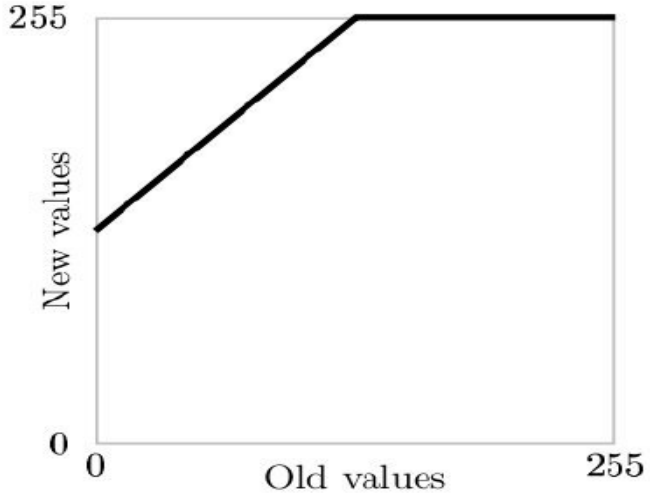
SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

Arithmetic Operations

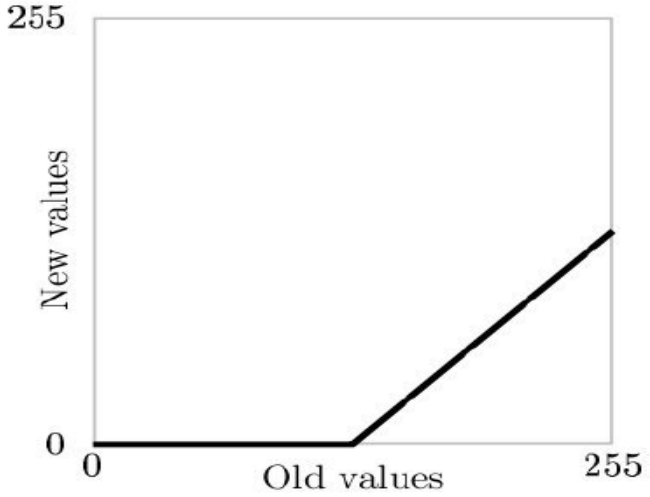
- These operations act by applying a simple function y=f(x) to each gray value in the image.
- Simple functions include **adding** or **subtract** a constant value to each pixel: $y = x \pm C$ (imadd, imsubtract)
- **Multiplying** each pixel by a constant: $y = C \cdot x$ (immultiply, imdivide)
- **Complement**: For a grayscale image is its photographic negative.

SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

Addition - Subtraction



Adding 128 to each pixel



Subtracting 128 from each pixel

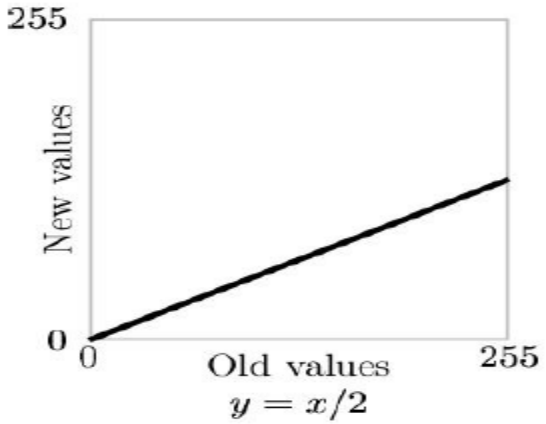
Image: J

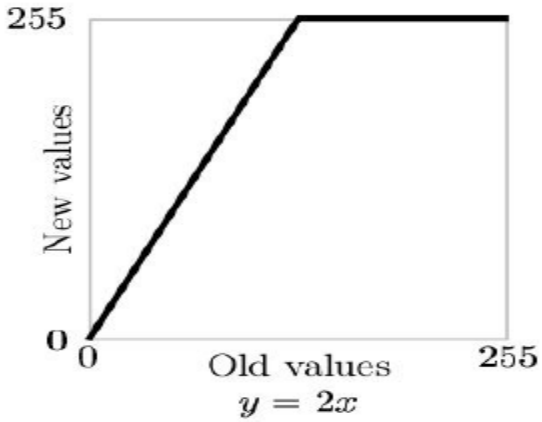
Image: J+20

Image: J-50

SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

Multiplication-Division





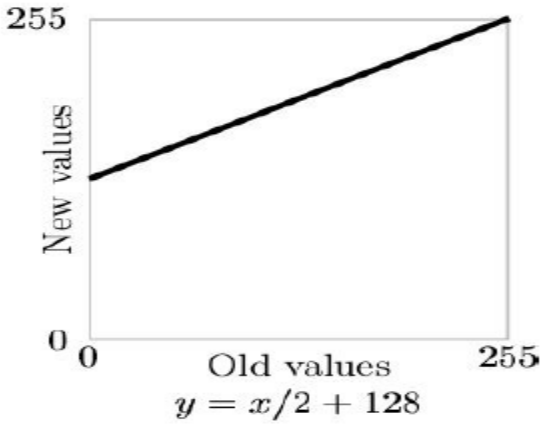


Image: J

Image: J*3

Image: J/2

LINEAR TRANSFORMATIONS

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 Negative:

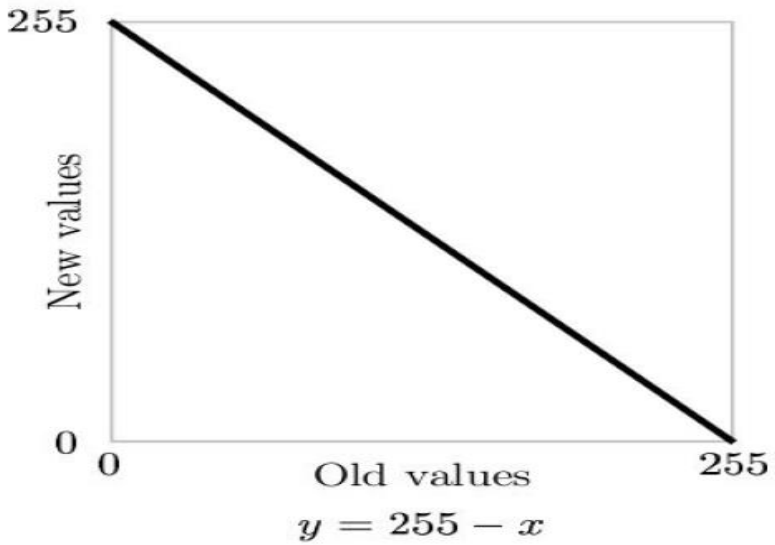
The negative of an image with intensity levels in the range [0, L-1] is obtained by using the negative transformation, by the expression

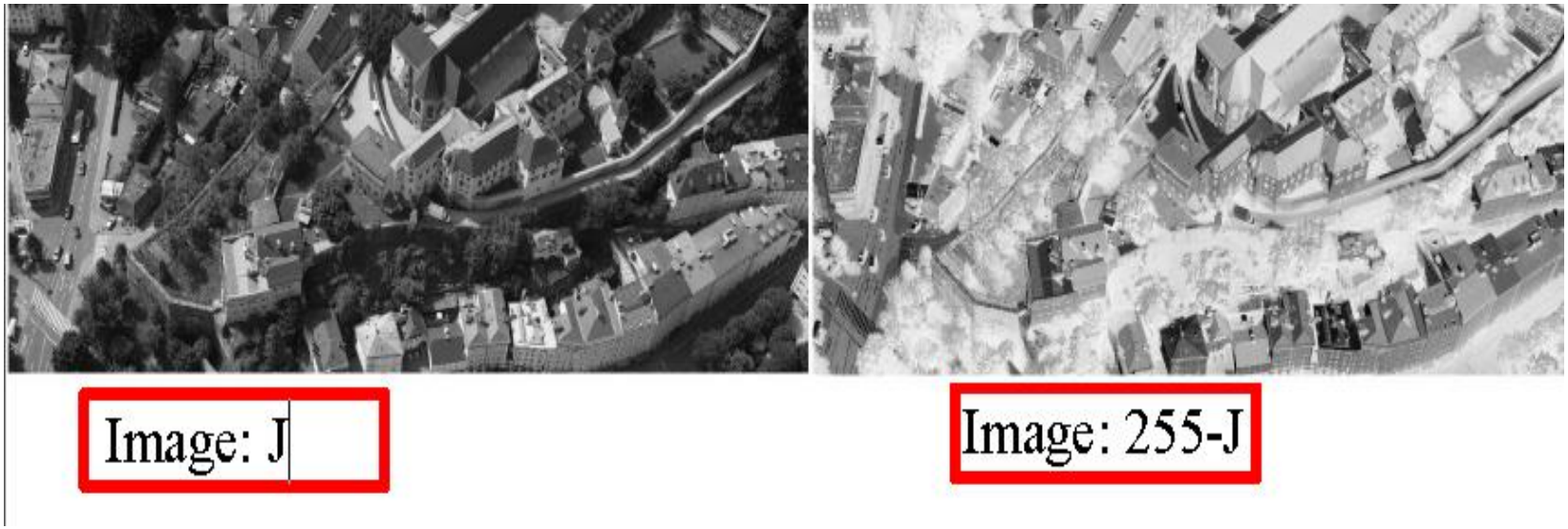
$$S = L - 1 - r$$

By reversing the intensity levels of an image in this manner produces the equivalent of a photographic negative

NEGATIVE TRANSFORMATION FUNCTIONS

Complement





This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image.

LOGARITHMIC TRANSFORMATIONS

Log Transformation

The general form of the log transformation:

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

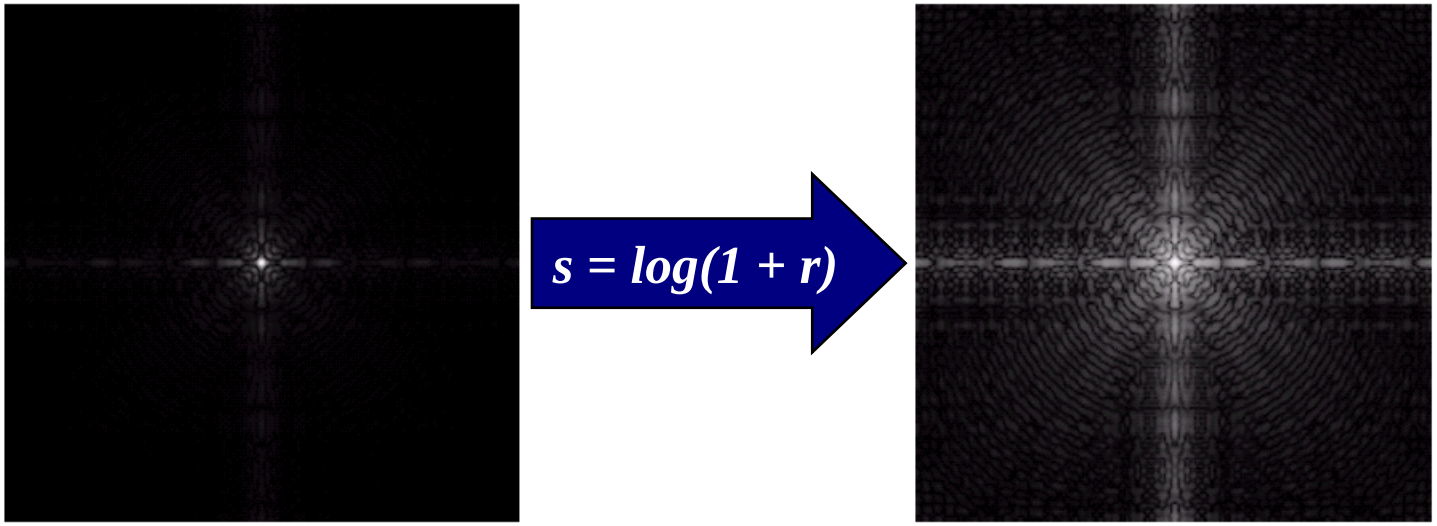
Where c is a constant, and r ≥ 0

- 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.

LOGARITHMIC TRANSFORMATIONS (CONT…)

Log functions are particularly useful when the input grey level values may have an extremely large range of values

In the following example the Fourier transform of an image is put through a log transform to reveal more detail



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.

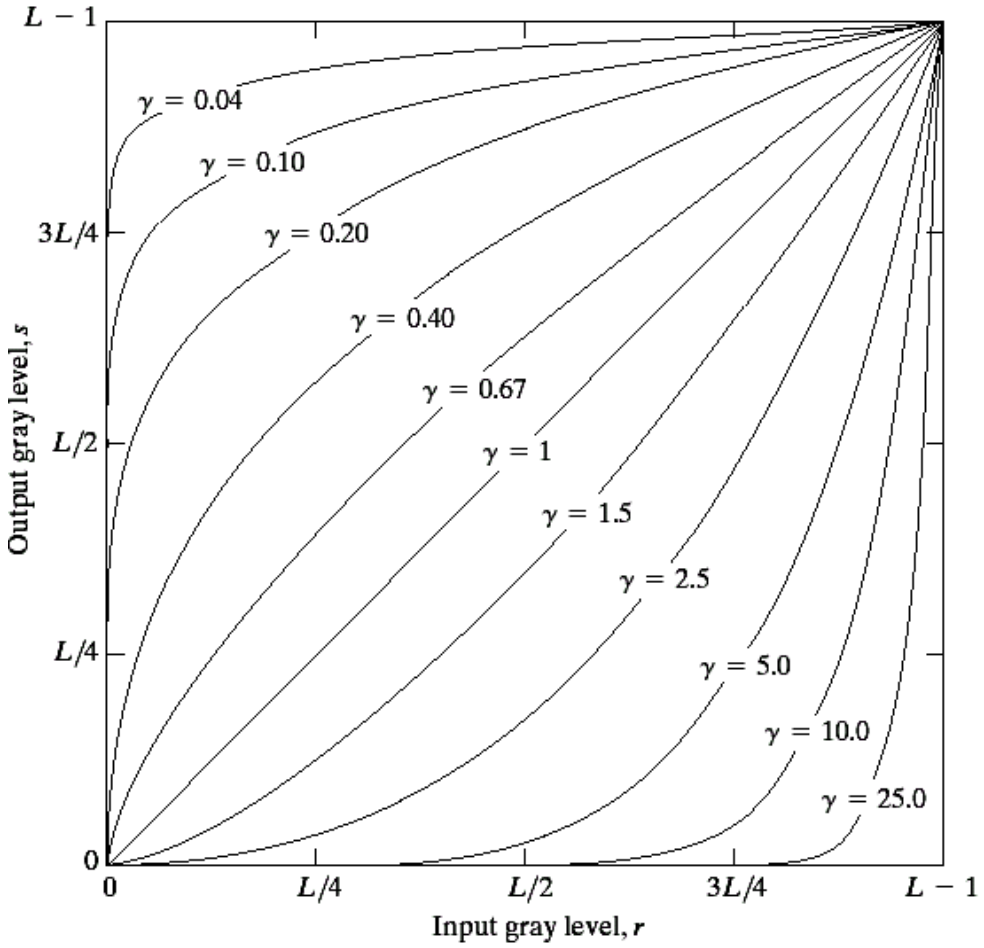
POWER LAW TRANSFORMATIONS

Power law transformations have the following form

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

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



POWER-LAW TRANSFORMATIONS

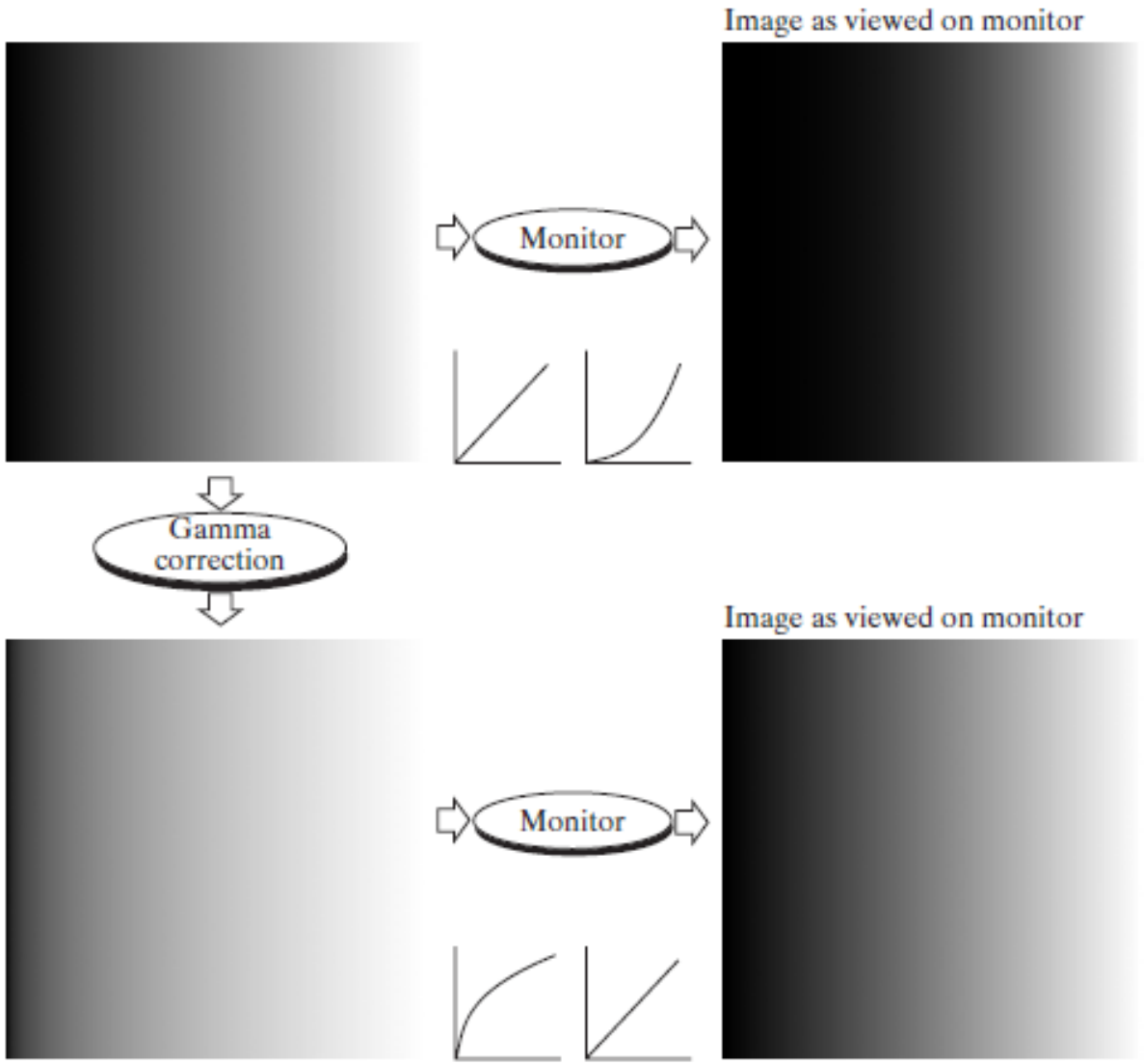
Variety of devices used for image capture, printing and display respond according to a power law.

The process used to correct this power-law response phenomena is called *gamma correction*.

Gamma correction in this case is straightforward. All we need to do is preprocess the input image before inputting it into the monitor by performing the transformation.

The result is shown in Fig. 3.7(c). When input into the same monitor, this gamma-corrected input produces an output that is close in appearance to the original image, as shown in Fig. 3.7(d).

POWER-LAW TRANSFORMATION



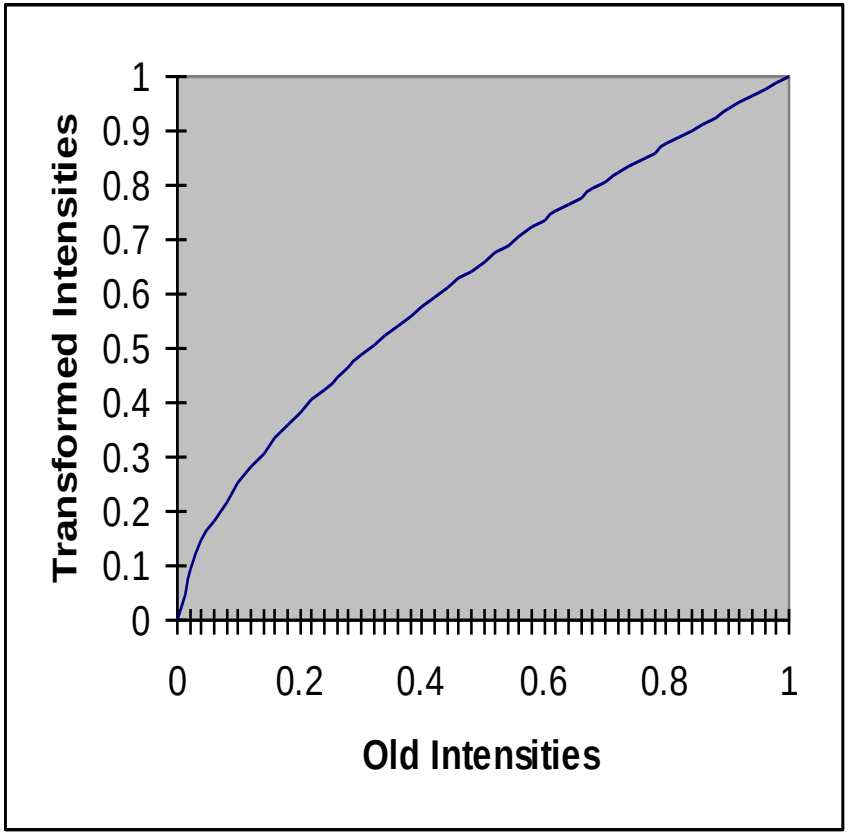
a b
c d

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 EXAMPLE

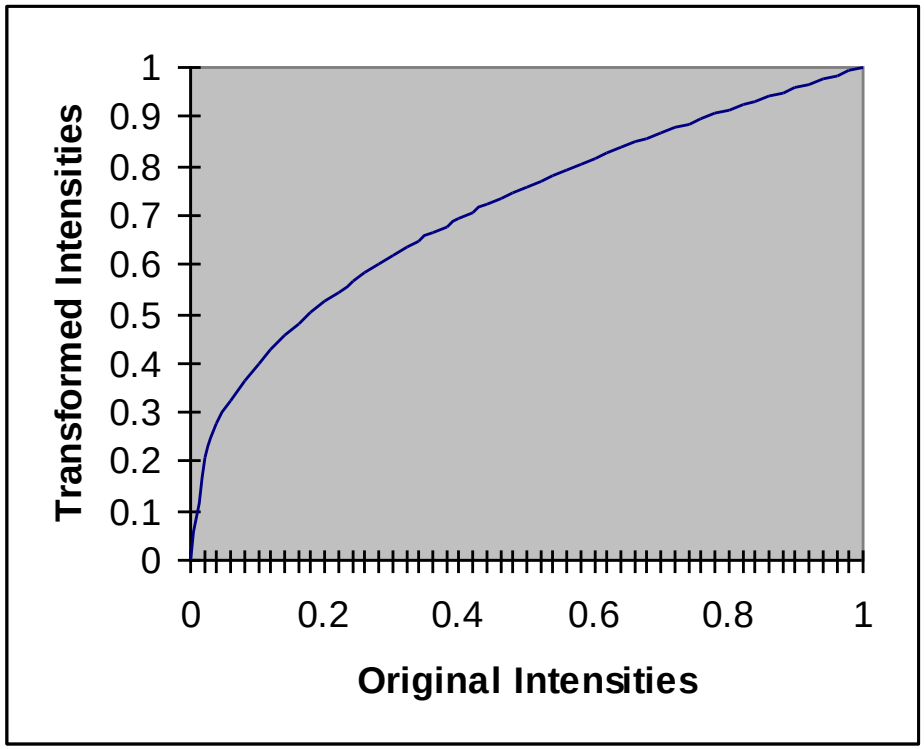
POWER LAW EXAMPLE (CONT…)

$$\gamma = 0.6$$



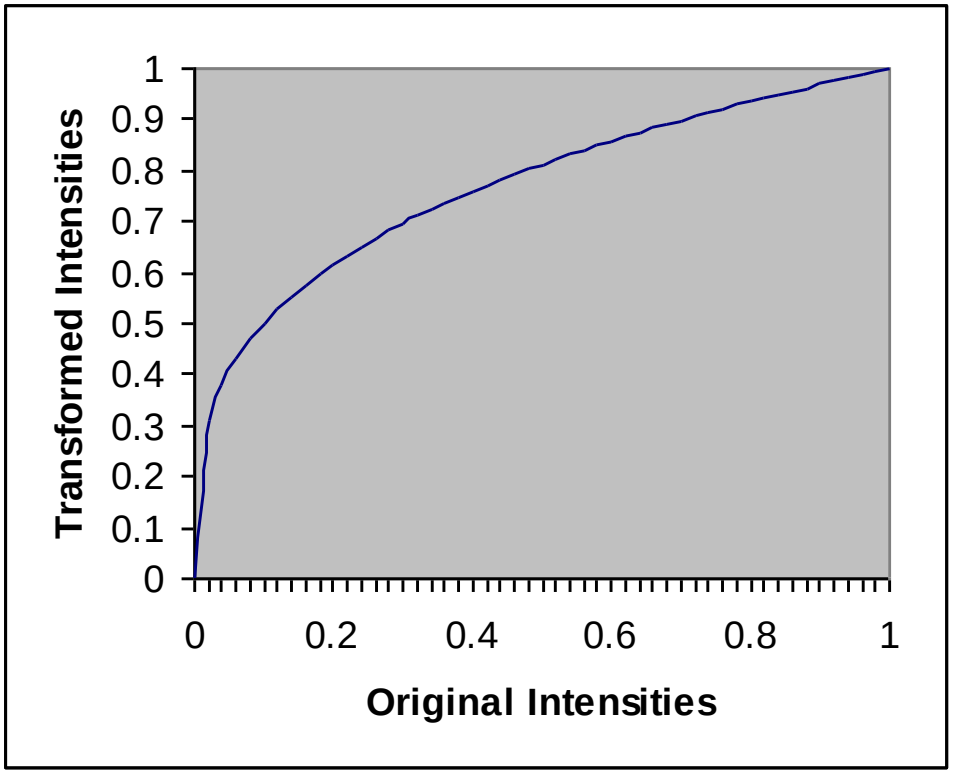
POWER LAW EXAMPLE (CONT…)

$$\gamma = 0.4$$



POWER LAW EXAMPLE (CONT…)

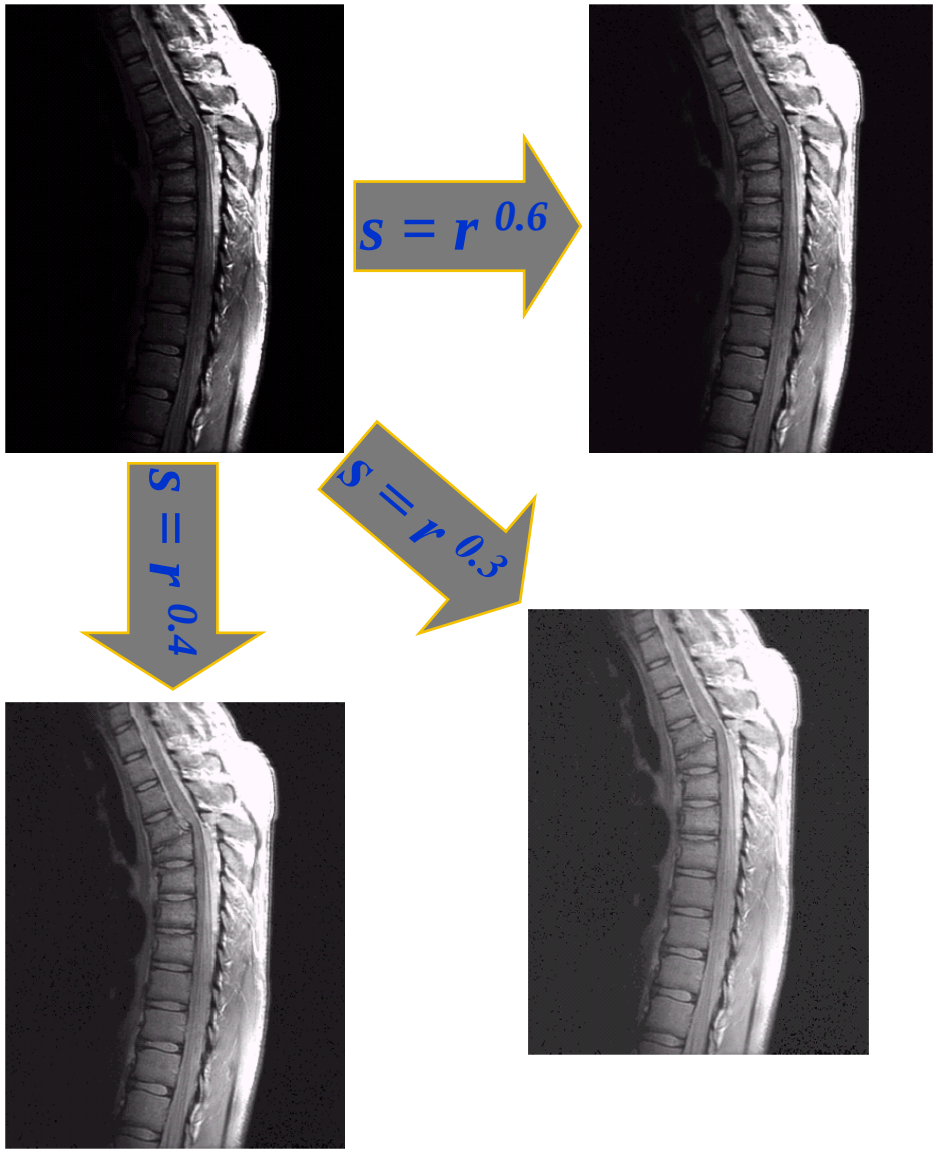
$$\gamma = 0.3$$



POWER LAW EXAMPLE (CONT…)

The images to the right show a magnetic resonance (MR) image of a fractured human spine.

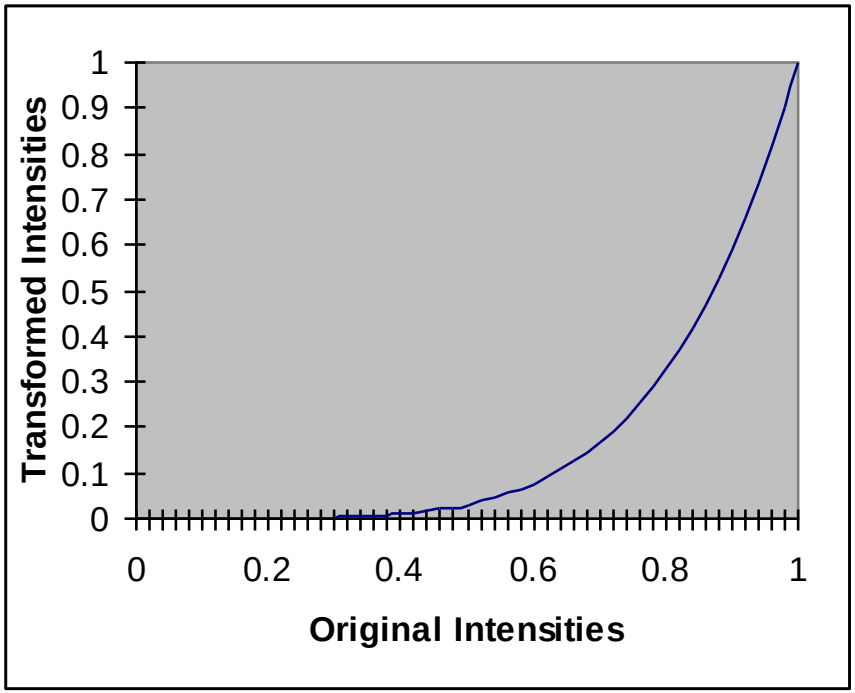
Different curves highlight different detail



POWER LAW EXAMPLE

POWER LAW EXAMPLE (CONT…)

$$\gamma = 5.0$$

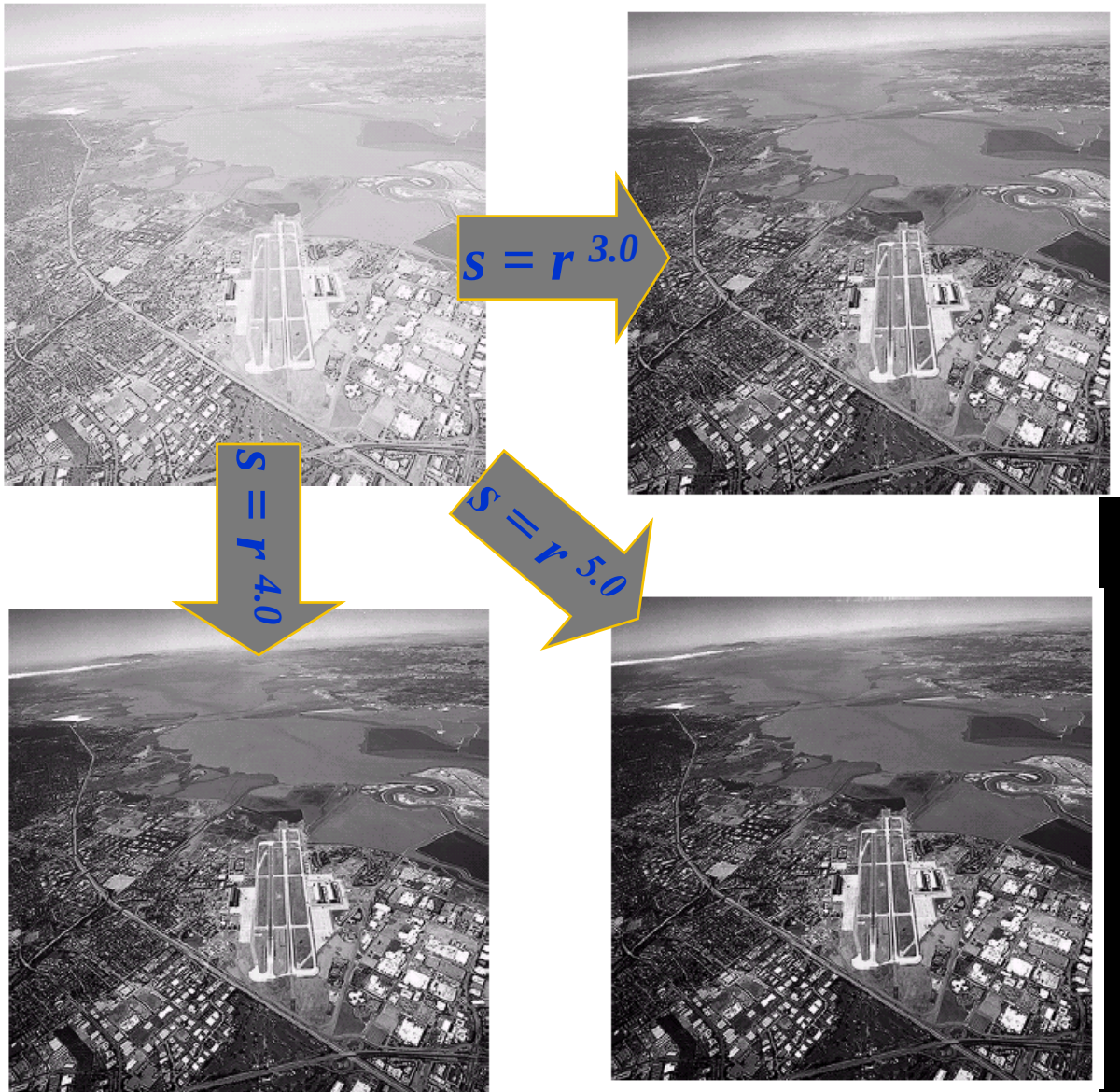


POWER LAW TRANSFORMATIONS (CONT…)

An aerial photo of a runway is shown

This time power law transforms are used to darken the image

Different curves highlight different detail



SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

General Commands

- **imread**: Read an image
- **figure**: creates a figure on the screen.
- **imshow(g)**: which displays the matrix g as an image.
- **pixval on**: turns on the pixel values in our figure.
- **impixel(i,j)**: the command returns the value of the pixel (i,j)
- **iminfo**: Information about the image.

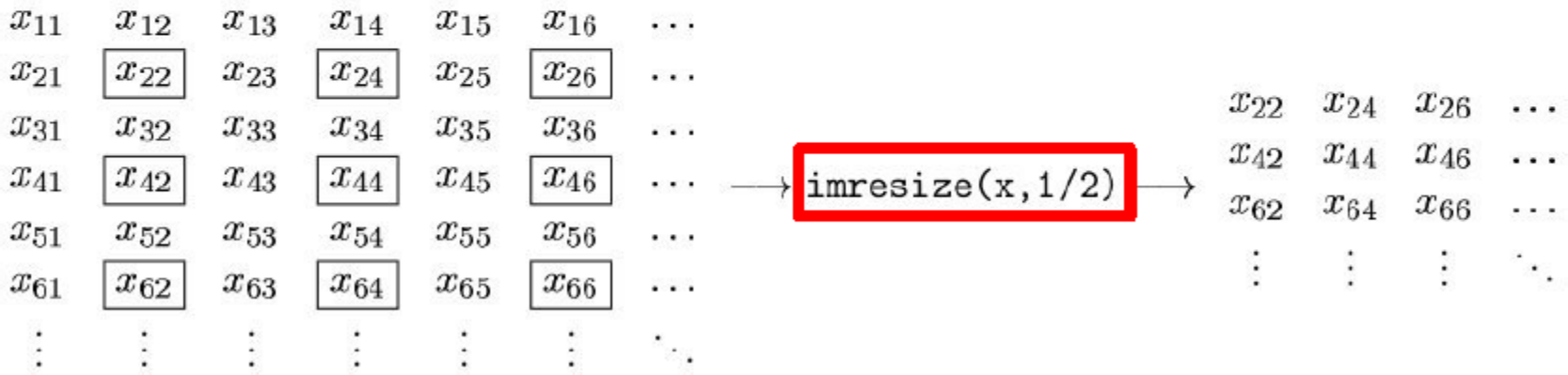
SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

Data Types

Data type	Description	Range
`int8`	8-bit integer	−128 — 127
`uint8`	8-bit unsigned integer	0 — 255
`int16`	16-bit integer	−32768 — 32767
`uint16`	16-bit unsigned integer	0 — 65535
`double`	Double precision real number	Machine specific

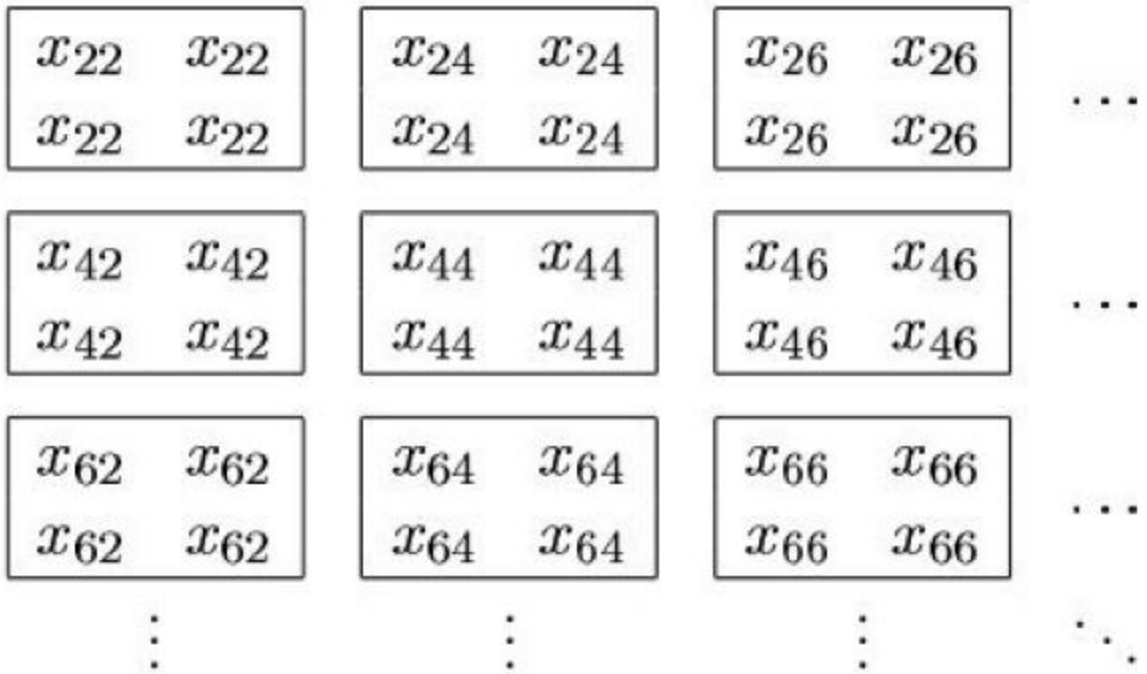
SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

Interpolation



SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

Extrapolation



PIECEWISE LINEAR TRANSFORMATION FUNCTIONS

- **A practical implementation of some important transformations can be formulated as piecewise functions.**
- **The disadvantage of piecewise functions is that their specification requires considerably more user input.**

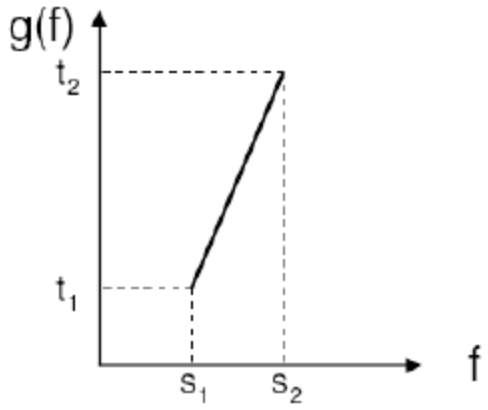
- **Principle Advantage: Some important transformations can be formulated only as a piecewise function.**

- **Types of Piecewise transformations are:**
  - Contrast Stretching
  - Gray-level Slicing
  - Bit-plane slicing

PIECEWISE-LINEAR TRANSFORMATION FUNCTIONS

Linear Stretching

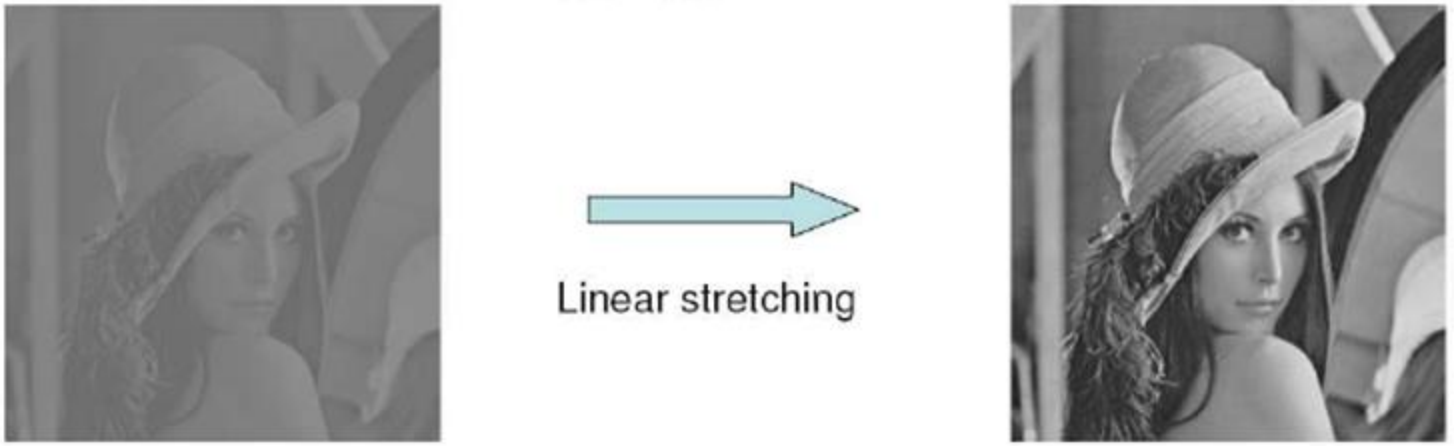
- Enhance the dynamic range by linear stretching the original gray levels to the range of 0 to 255
- Example
  - The original gray levels are [100, 150]
  - The target gray levels are [0, 255]
  - The transformation function



$$g(f) = (f - s_1)/(s_2 - s_1)*(t_2 - t_1) + t_1$$

$$g(f) = ((f - 100)/50)*255 + 0, \qquad for\ 100 \le f \le 150$$

PIECEWISE-LINEAR TRANSFORMATION FUNCTIONS

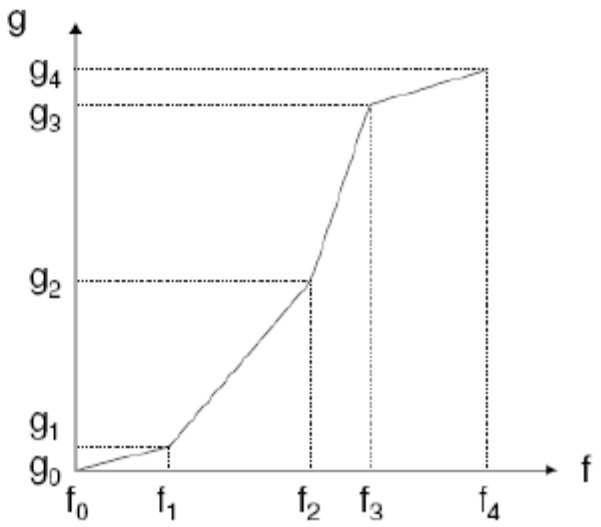


PIECEWISE-LINEAR TRANSFORMATION FUNCTIONS

Piecewise Linear Stretching

- K segments
  - Starting position of input $\{f_k, k = 0, \ldots K\text{-}1\}$
  - Starting position of output $\{g_k, k = 0, \ldots K\text{-}1\}$
  - Transform function

$$g(f) = \frac{f - f_k}{f_{k+1} - f_k} * (g_{k+1} - g_k) + g_k;$$
$$for\ f_k < f \le f_{k+1}, \quad k = 0,1,\ldots,K-1.$$

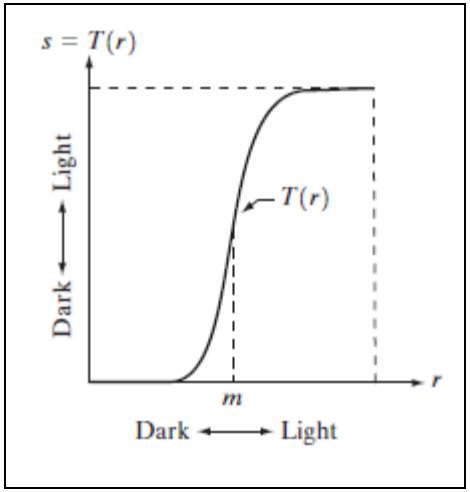


Contrast Stretching:

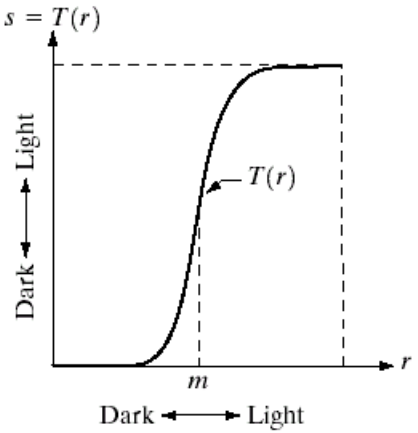
The effect of applying the transformation to every pixel of f to generate the corresponding pixels in g would:

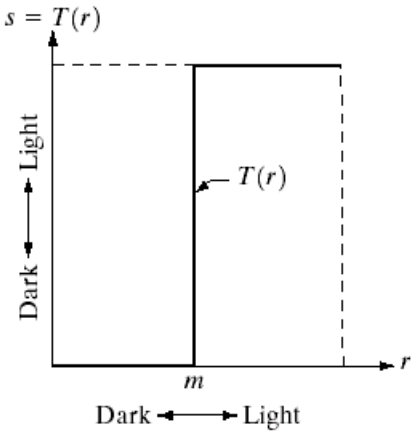
Produce higher contrast than the original image, by:

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



Gray-level Transformation





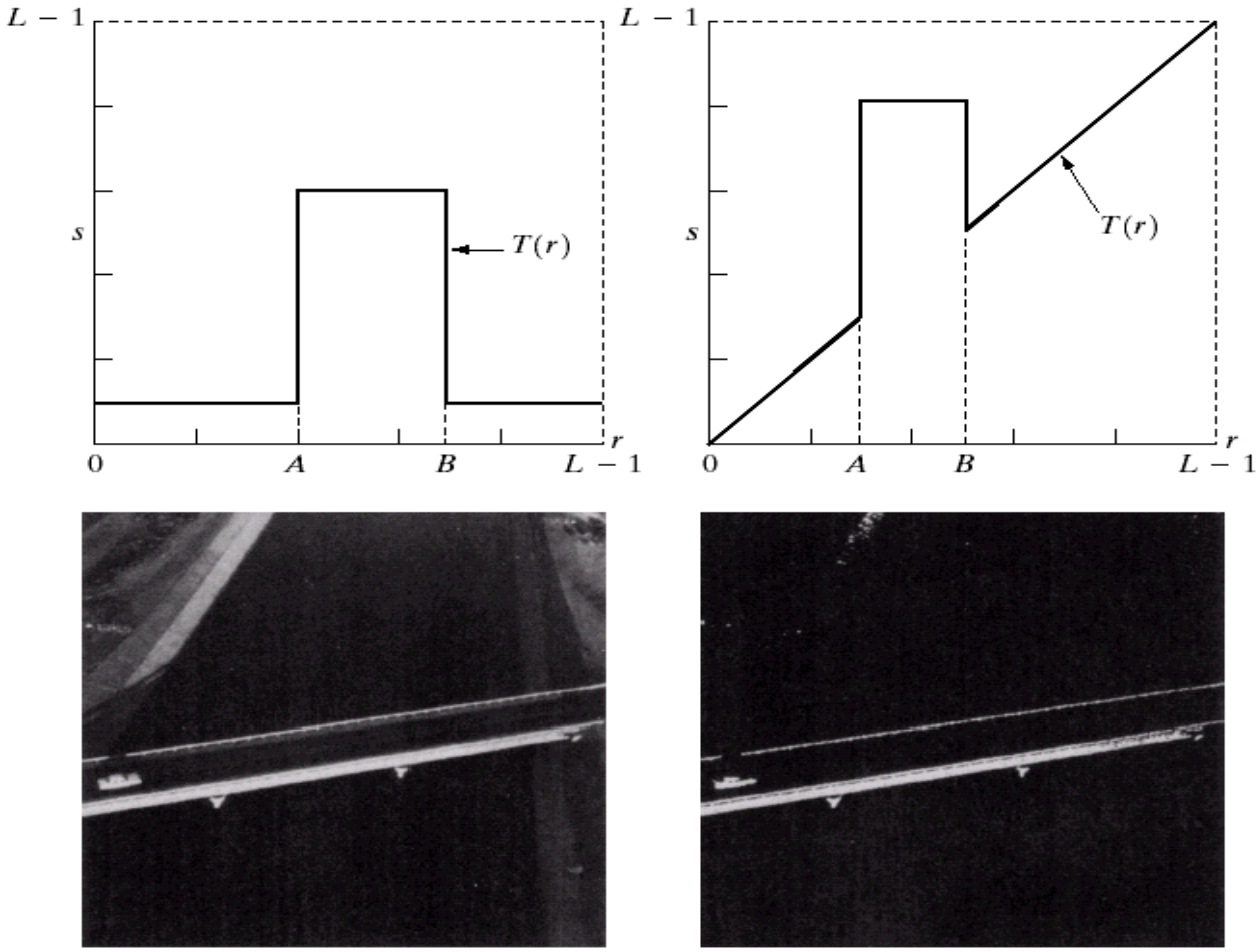
a b

FIGURE 3.2 Gray-level transformation functions for contrast enhancement.

Intensity-level Slicing:

- Highlighting a specific range of intensities in an image.
- Applications include enhancing features such as water in satellite imagery and enhancing flaws in X-ray images.

Piecewise-Linear Transformation Functions
Case 2: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).

Fig. 3.11 (a) produces a binary image

Fig. 3.11 (b) brightens / darkens the desired range of intensities but leaves all other intensity levels in the image unchanged.

Highlights a specific range of grey levels

- Similar to thresholding
- Other levels can be suppressed or maintained
- Useful for highlighting features in an image

Piecewise-Linear Transformation Functions
Case 3:Bit-plane Slicing

Bit-plane slicing:

- **Pixels are digital numbers composed of bits.**
- **256 gray level image composed of 8 bits**
- **It can highlight the contribution made to total image appearance by specific bits.**
- **Each pixel in an image represented by 8 bits.**
- **Image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit plane 7 for the most significant bit.**

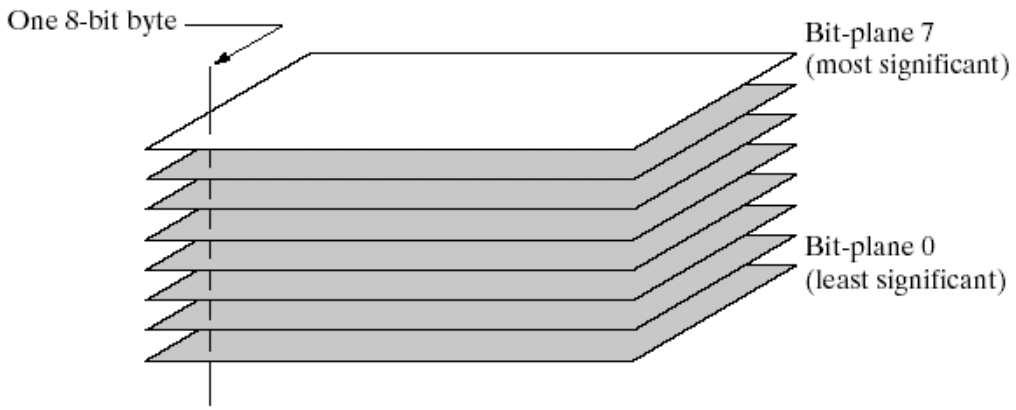


FIGURE 3.12 Bit-plane representation of an 8-bit image.

Piecewise-Linear Transformation Functions
Bit-plane Slicing: A Fractal Image

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

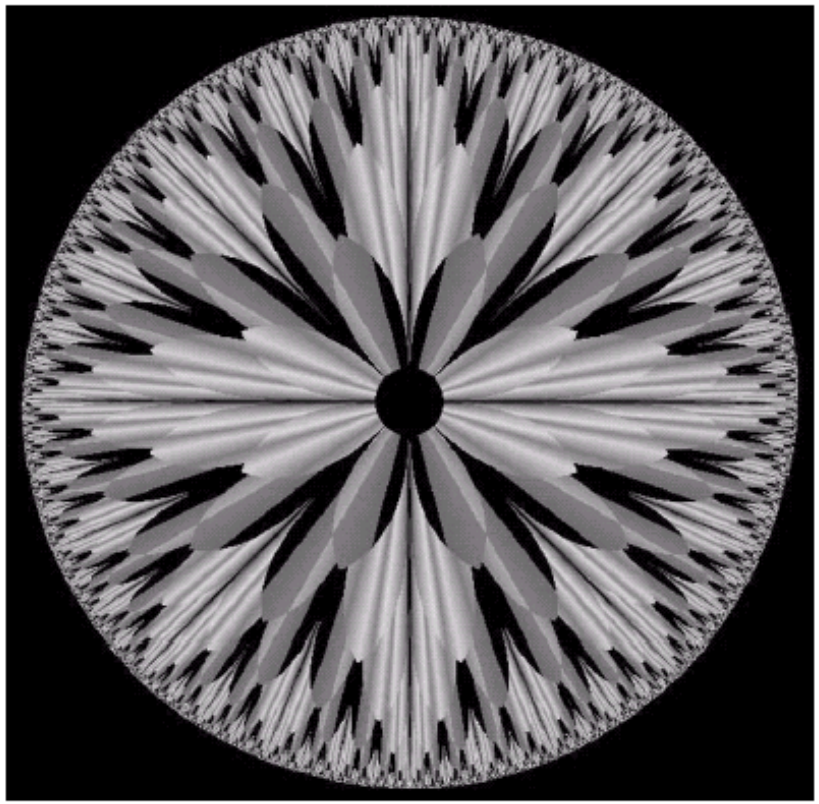
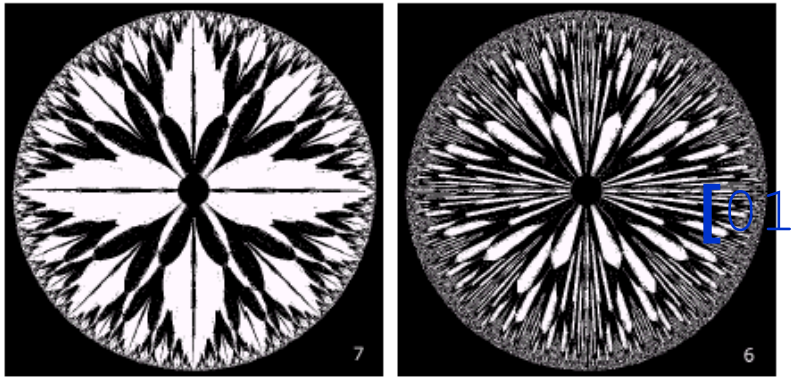


FIGURE 3.13 An 8-bit fractal image. (A fractal is an image generated from mathematical expressions). (Courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA.)

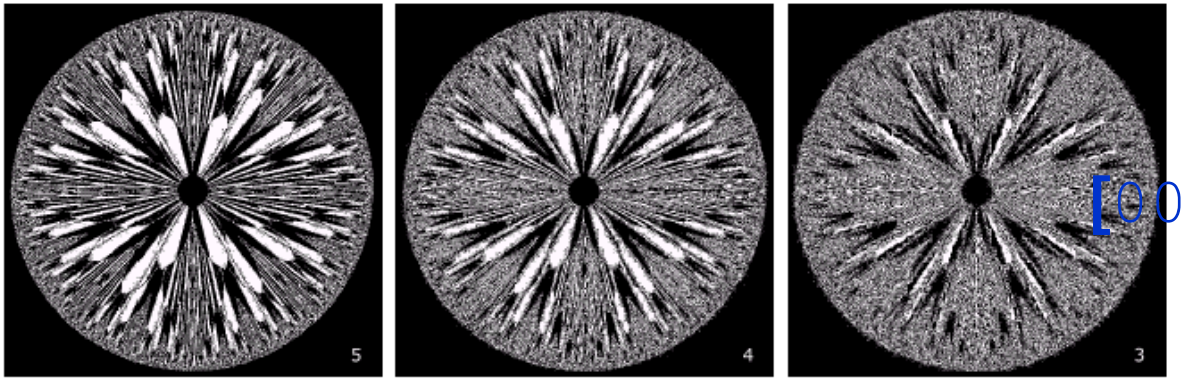
Piecewise-Linear Transformation Functions
Bit-plane Slicing: A Fractal Image

[10000000]



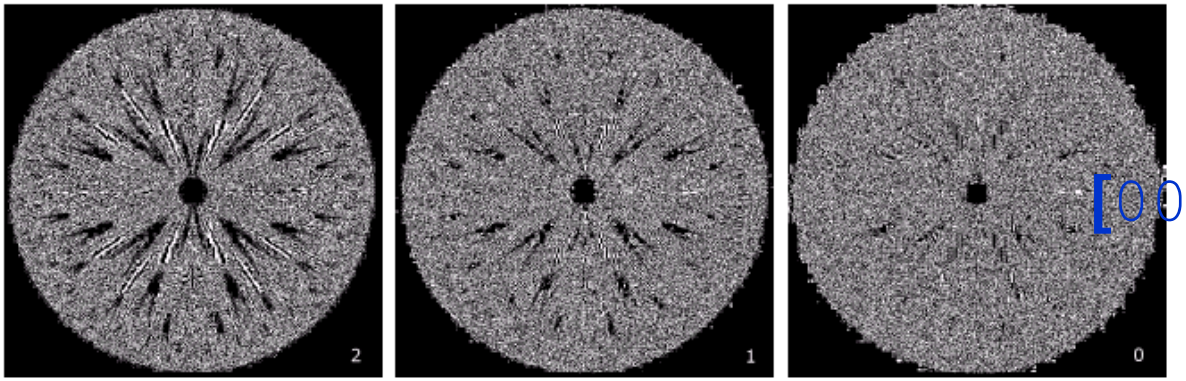
[01000000]

[00100000]



[00001000]

[00000100]



[00000001]

BIT PLANE SLICING (CONT...)

a b c
d e f
g h i

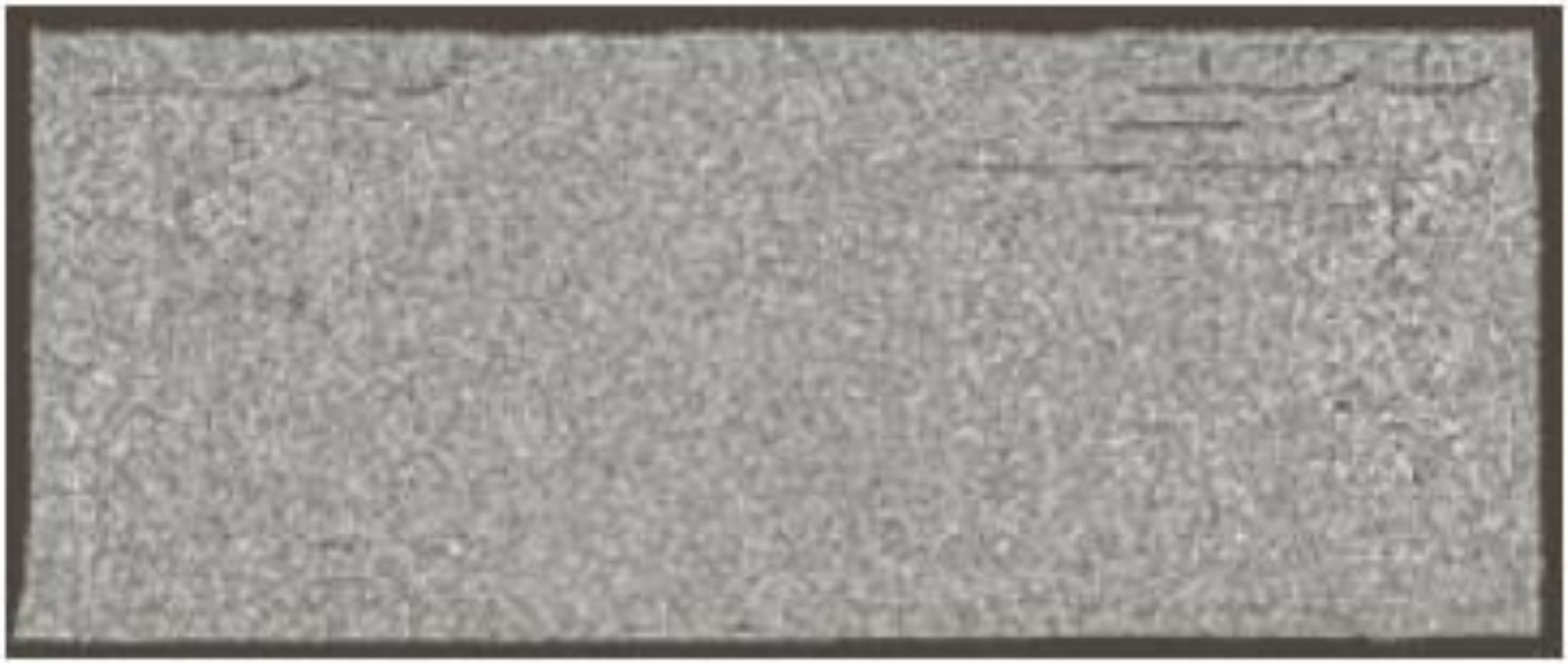
FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

BIT PLANE SLICING (CONT…)

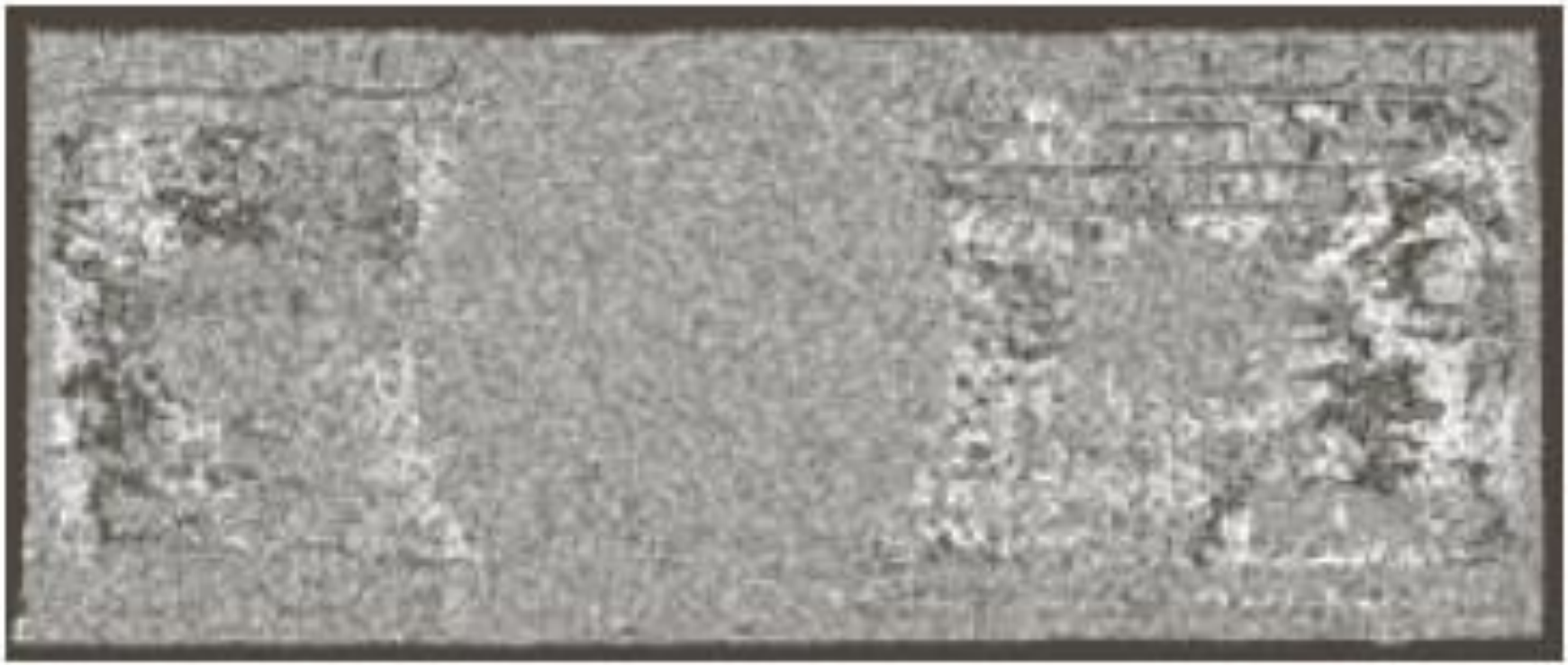
BIT PLANE SLICING (CONT…)

BIT PLANE SLICING (CONT…)

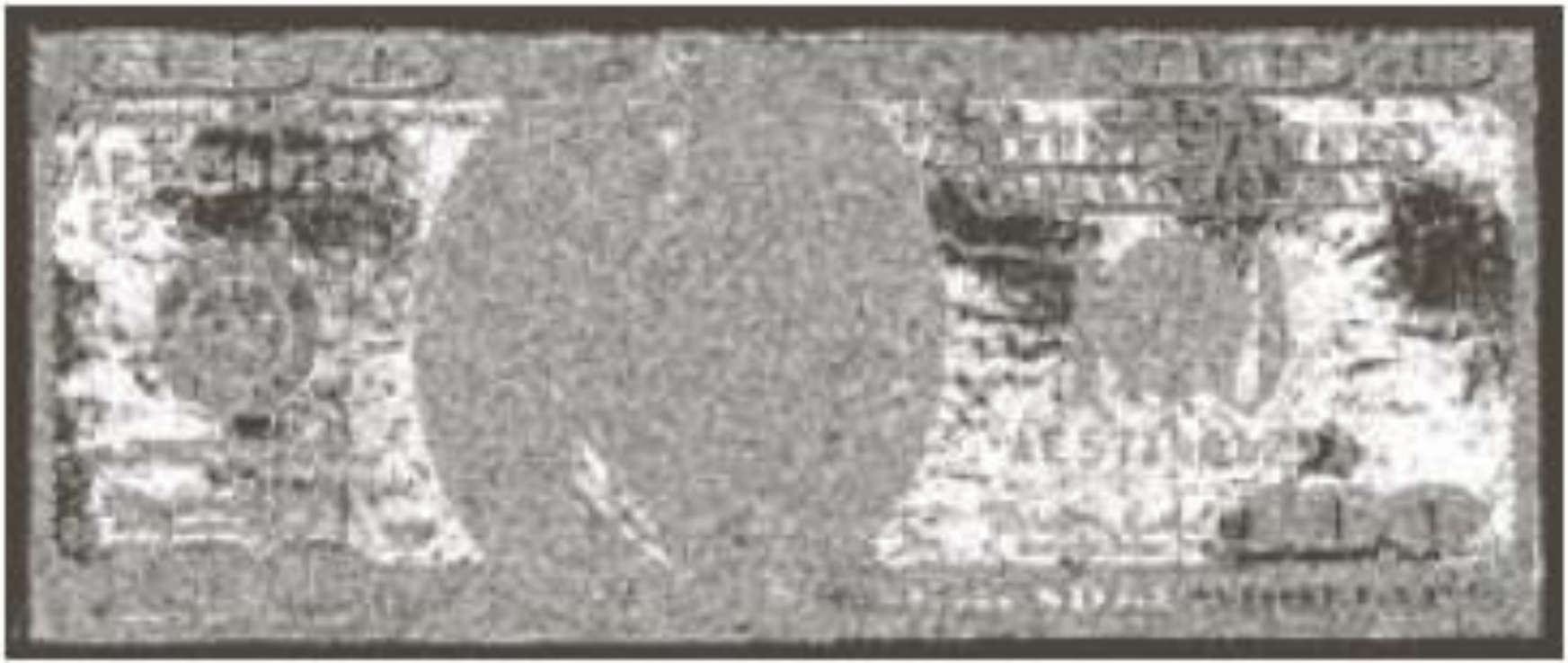
BIT PLANE SLICING (CONT…)



BIT PLANE SLICING (CONT…)



BIT PLANE SLICING (CONT…)



BIT PLANE SLICING (CONT…)

BIT PLANE SLICING (CONT…)

BIT PLANE SLICING (CONT…)

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

Piecewise Linear Transformations

Exercise:

How does the transformation function look for bitplanes 0,1,… ?

What is the easiest way to filter a single bitplane (e.g. in MATLAB) ?

```
A=imread('boy.tif');
B=bitget(img,1); figure, imshow(logical(B));title('Bit plane 1');
B=bitget(A,2); figure, imshow(logical(B));title('Bit plane 2');
.
.
B=bitget(A,8); figure, imshow(logical(B));title('Bit plane 8');
```

SUMMARY

We have looked at different kinds of point processing image enhancement