

# INTENSITY TRANSFORMATIONS

- What is image enhancement?
- Different kinds of image enhancement
  - Point processing
  - Histogram processing
  - Spatial filtering

# What Is Image Enhancement?

Image enhancement is the process of making images more useful

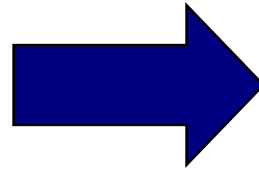
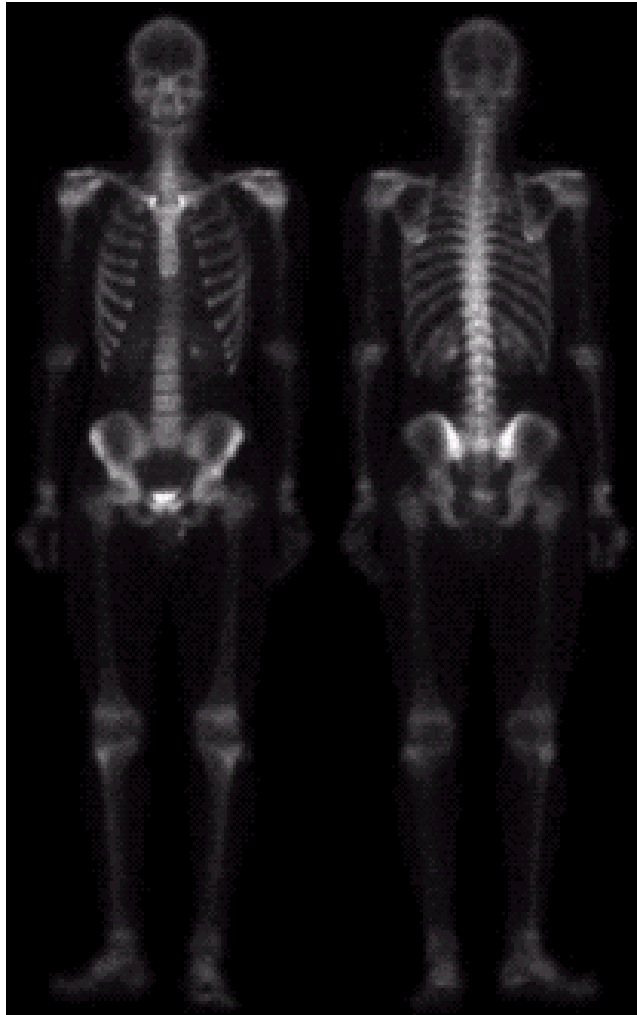
The reasons for doing this include:

- Highlighting interesting detail in images
- Removing noise from images
- Making images more visually appealing

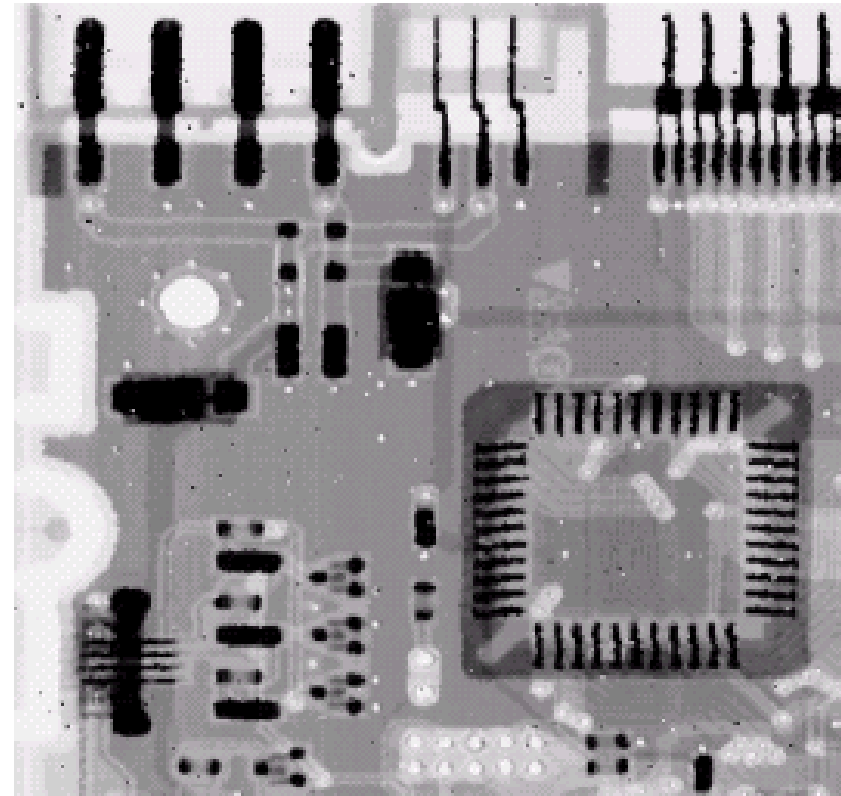
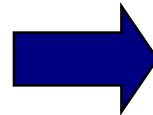
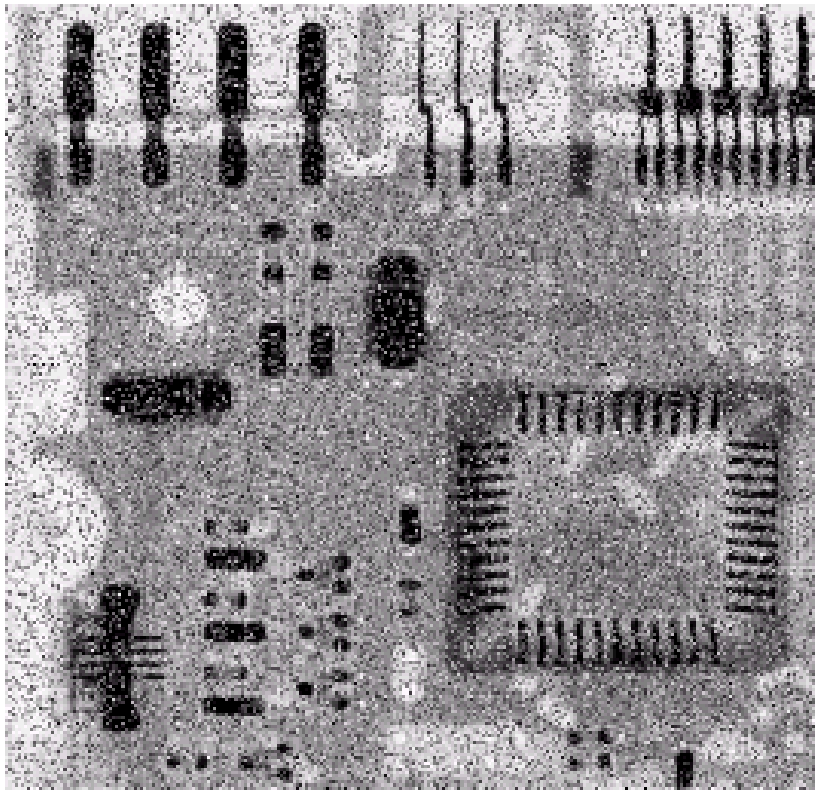
# Image Enhancement Examples



# Image Enhancement Examples (cont...)



# Image Enhancement Examples (cont...)



There are two broad categories of image enhancement techniques

- Spatial domain techniques
  - Direct manipulation of image pixels
- Frequency domain techniques
  - Manipulation through Fourier transform or wavelet transform of an image

For the moment we will concentrate on techniques that operate in the spatial domain

In this lecture we will look at image enhancement point processing techniques:

- What is point processing?
- Negative images
- Thresholding
- Logarithmic transformation
- Power law transforms
- Grey level slicing
- Bit plane slicing



# A Note About Grey Levels

So far when we have spoken about image grey level values we have said they are in the range  $[0, 255]$

- Where 0 is black and 255 is white

There is no reason why we have to use this range

- The range  $[0, 255]$  stems from display technologies

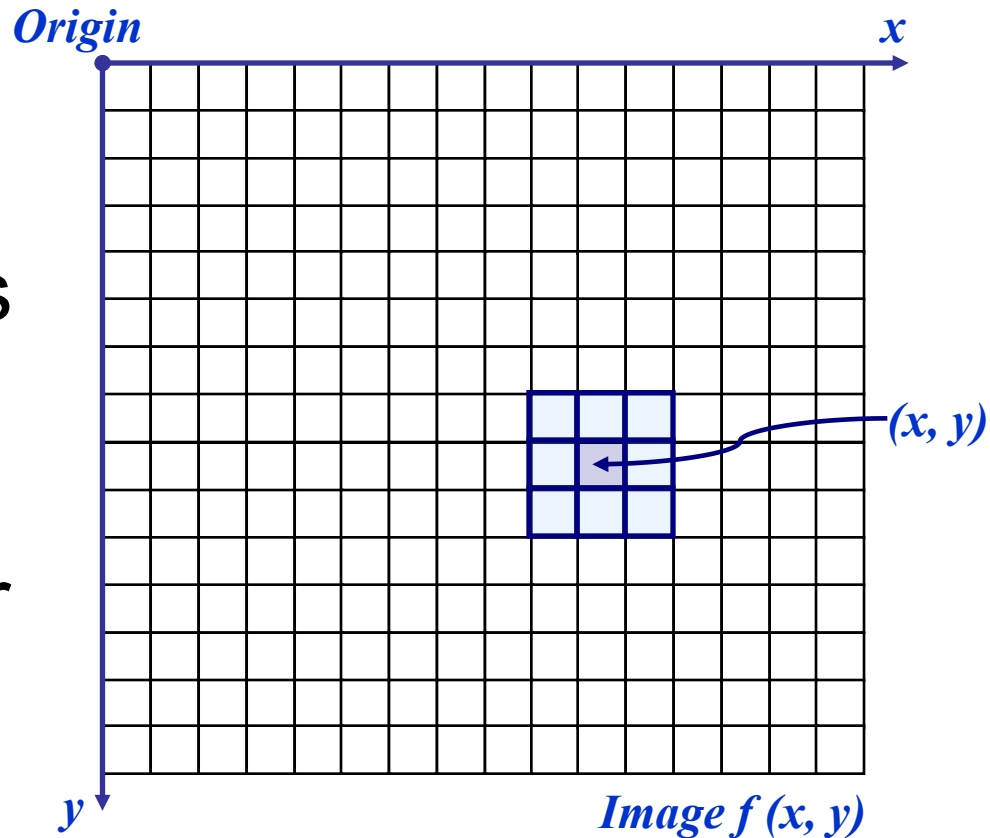
For many of the image processing operations in this lecture grey levels are assumed to be given in the range  $[0.0, 1.0]$

# Basic Spatial Domain Image Enhancement

Most spatial domain enhancement operations can be reduced to the form

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

where  $f(x, y)$  is the input image,  $g(x, y)$  is the processed image and  $T$  is some operator defined over some neighbourhood of  $(x, y)$



The simplest spatial domain operations occur when the neighbourhood is simply the pixel itself

In this case  $T$  is referred to as a *grey level transformation function* or a *point processing operation*

Point processing operations take the form

$$s = T ( r )$$

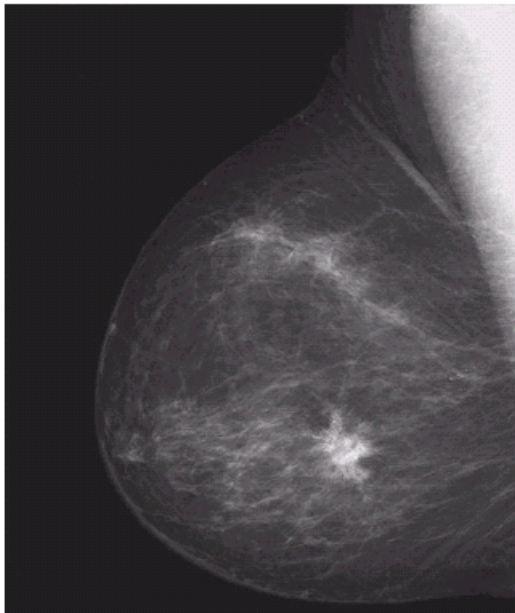
where  $s$  refers to the processed image pixel value and  $r$  refers to the original image pixel value.

# Point Processing Example: Negative Images

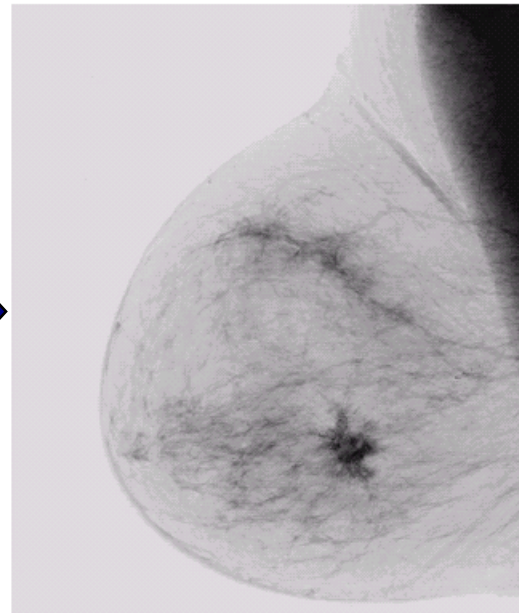
Negative images are useful for enhancing white or grey detail embedded in dark regions of an image

- Note how much clearer the tissue is in the negative image of the mammogram below

**Original  
Image**



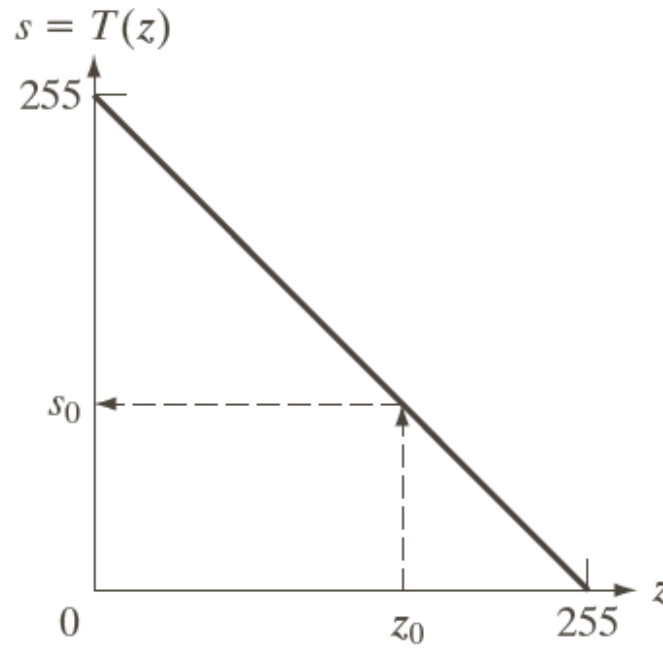
$$s = 1.0 - r$$



**Negative  
Image**

# Point Processing Example: Negative Images (cont...)

$$s = \textit{intensity}_{max} - r$$

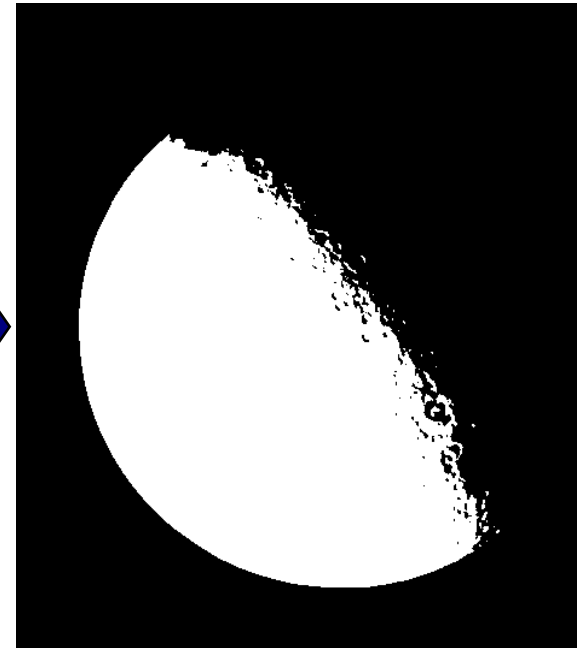


# Point Processing Example: Thresholding

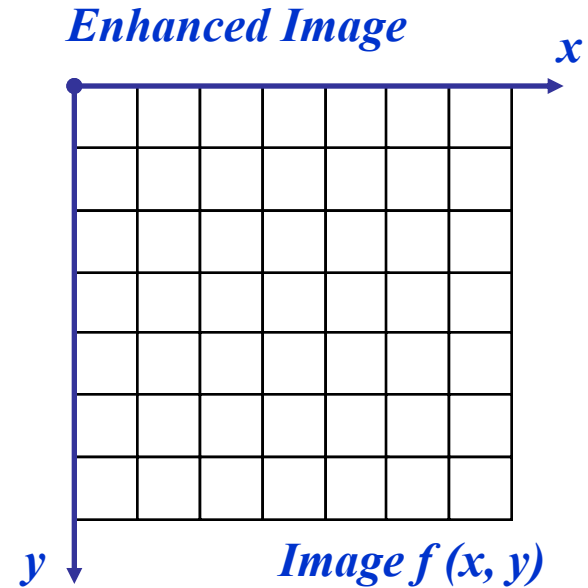
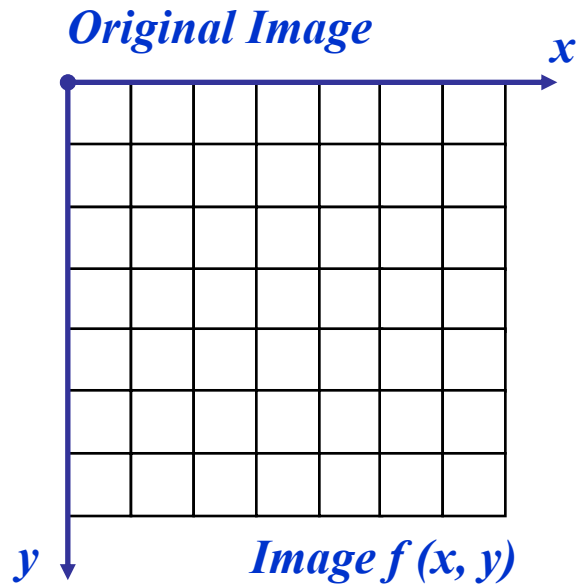
Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background



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

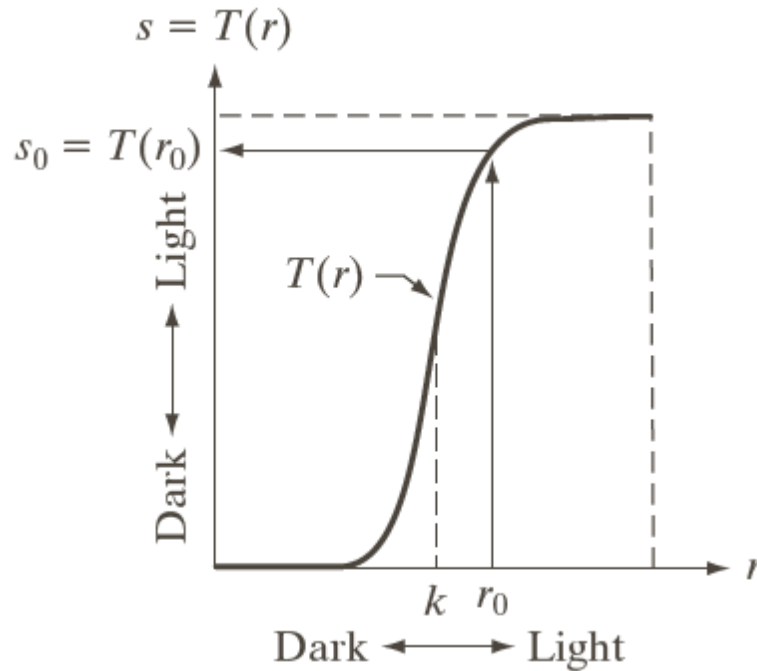


# Point Processing Example: Thresholding (cont...)

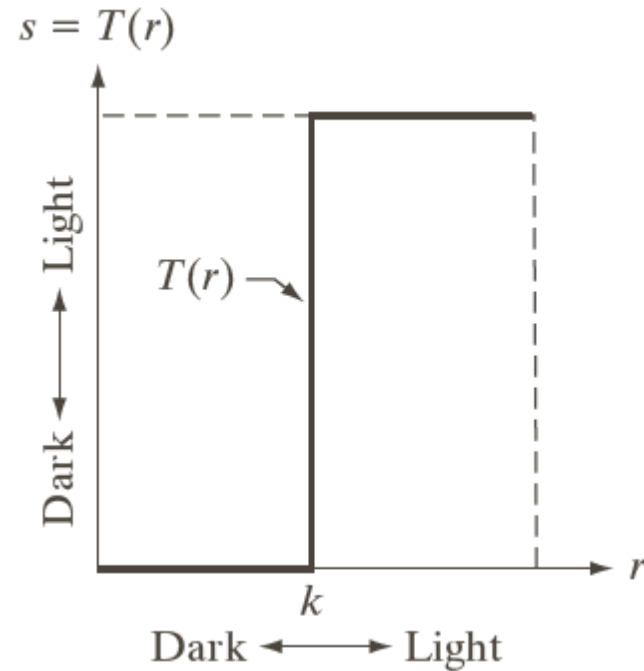


$$s = \begin{cases} 1.0 & r > \text{threshold} \\ 0.0 & r \leq \text{threshold} \end{cases}$$

# Intensity Transformations



Contrast stretching



Thresholding

Darkening of intensity levels below  $k$  and brightening the intensity level above  $k$

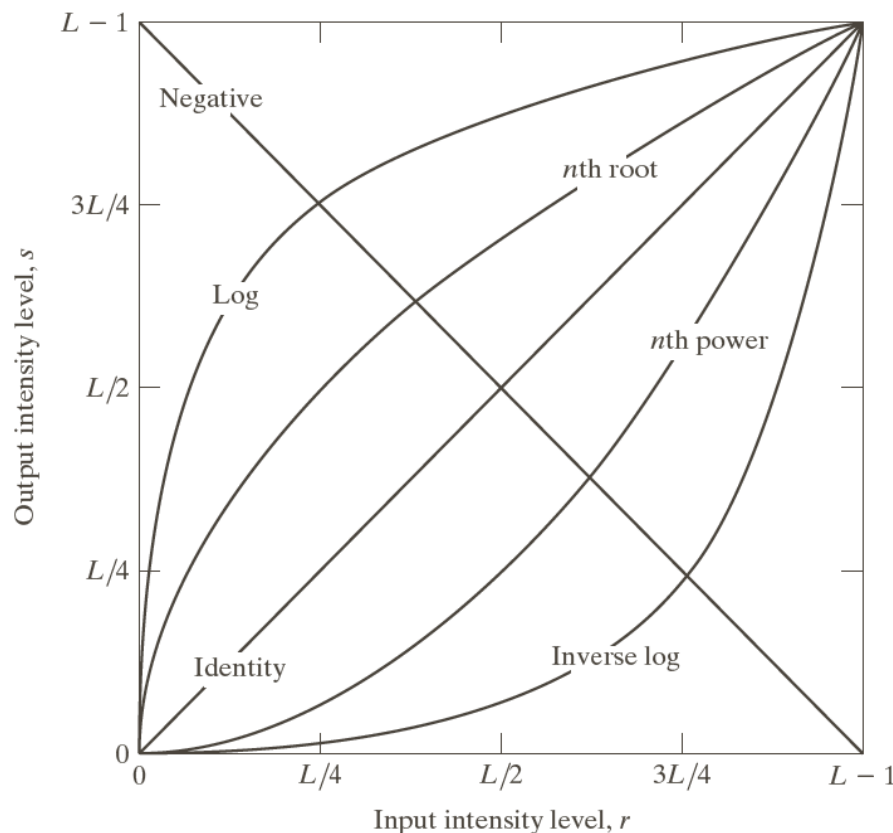


# Basic Grey Level Transformations

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
  - $n^{\text{th}}$  power/ $n^{\text{th}}$  root



# Logarithmic Transformations

The general form of the log transformation is

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

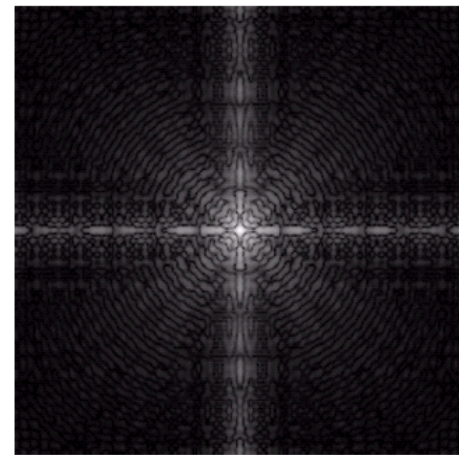
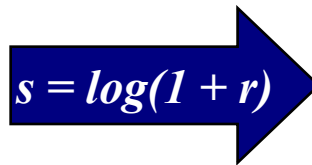
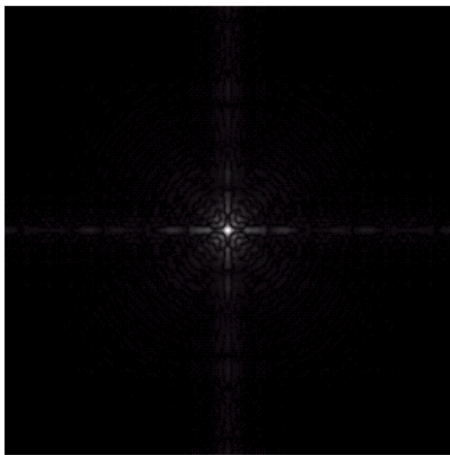
The log transformation maps a narrow range of low input grey level values into a wider range of output values

The inverse log transformation performs the opposite transformation

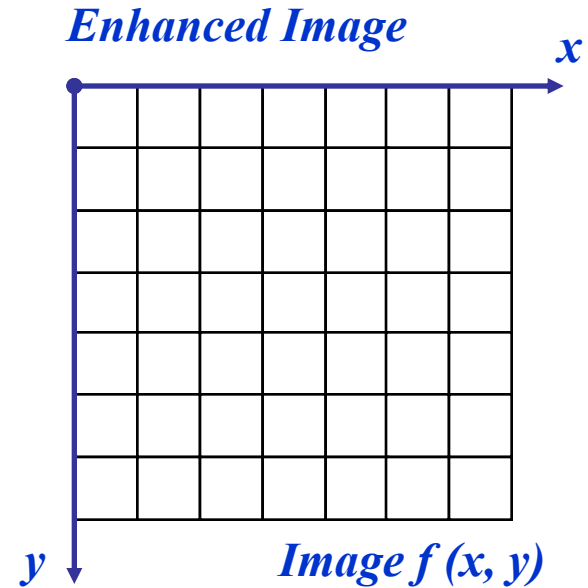
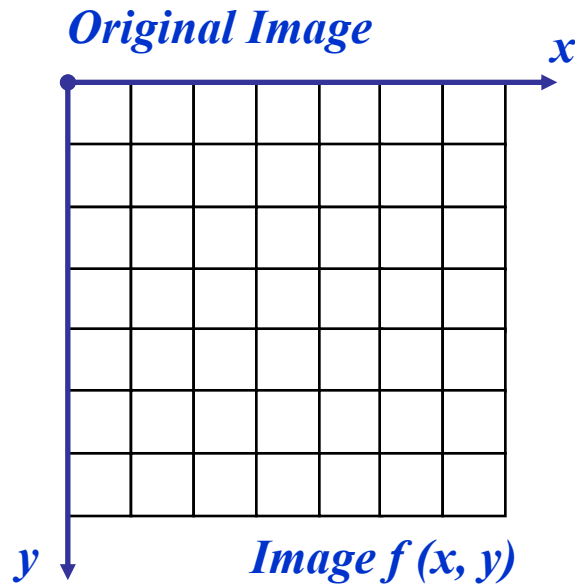
# Logarithmic Transformations (cont...)

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 (cont...)



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

We usually set  $c$  to 1.

Grey levels must be in the range  $[0.0, 1.0]$

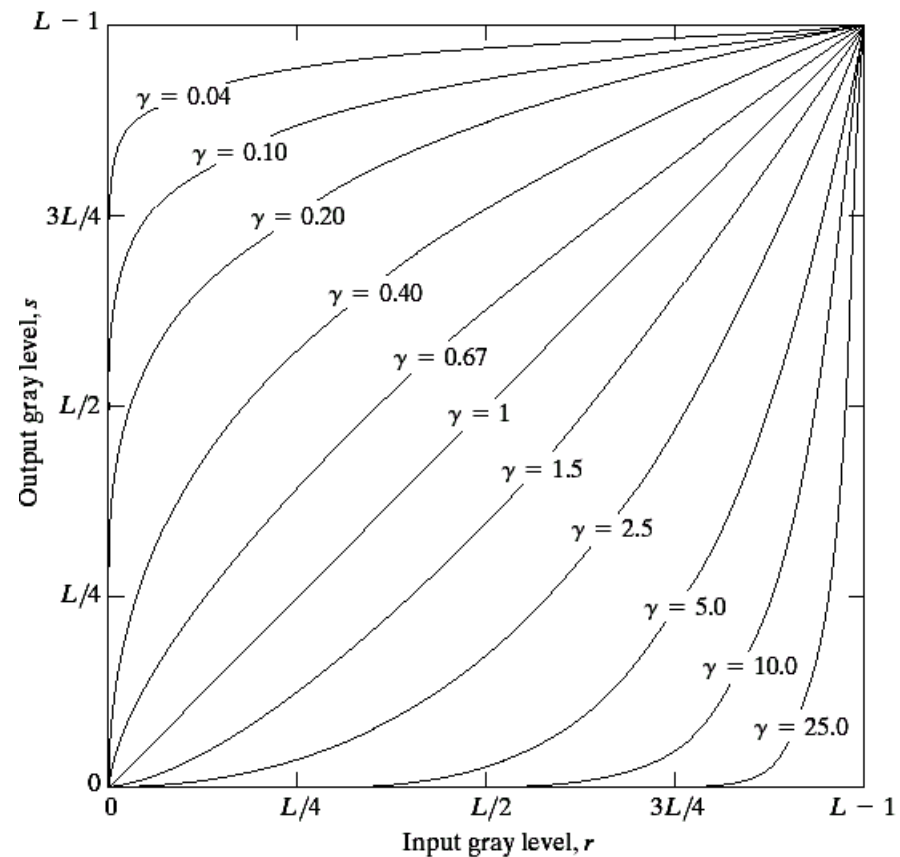
# Power Law Transformations

Power law transformations have the following form

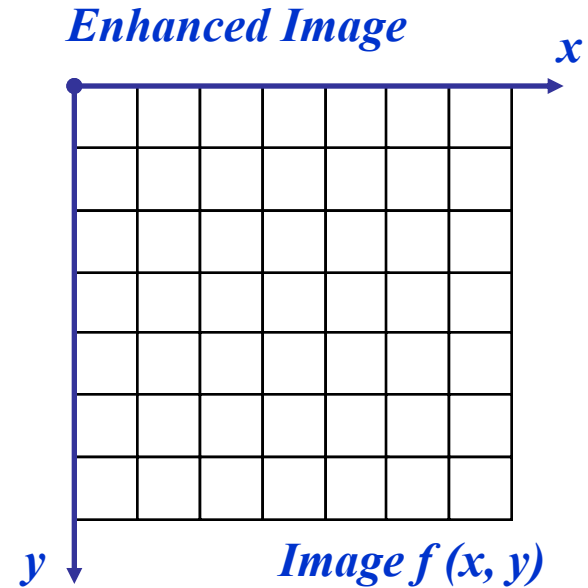
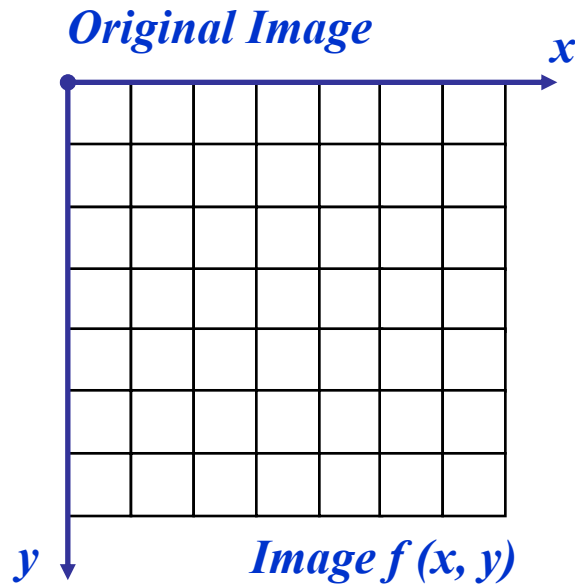
$$s = c * r^\gamma$$

Map a narrow range of dark input values into a wider range of output values or vice versa

Varying  $\gamma$  gives a whole family of curves



# Power Law Transformations (cont...)



$$s = r^\gamma$$

We usually set  $c$  to 1.

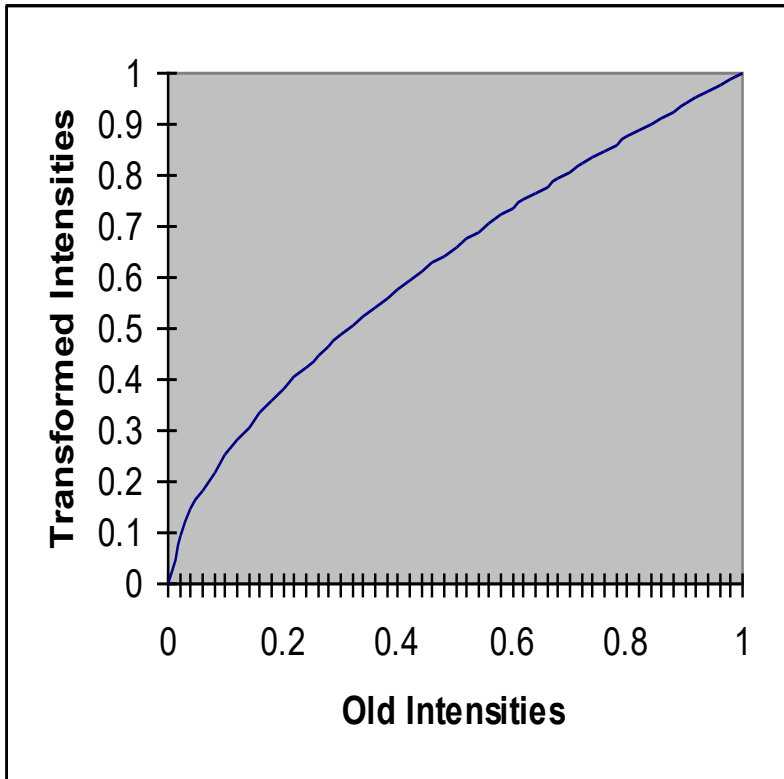
Grey levels must be in the range  $[0.0, 1.0]$

# Power Law Example



# Power Law Example (cont...)

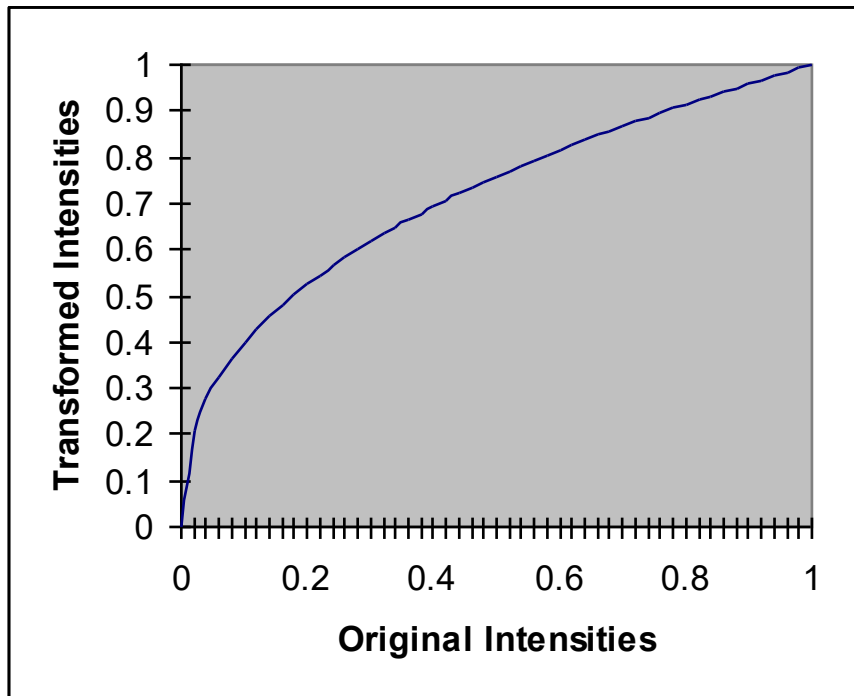
$$\gamma = 0.6$$





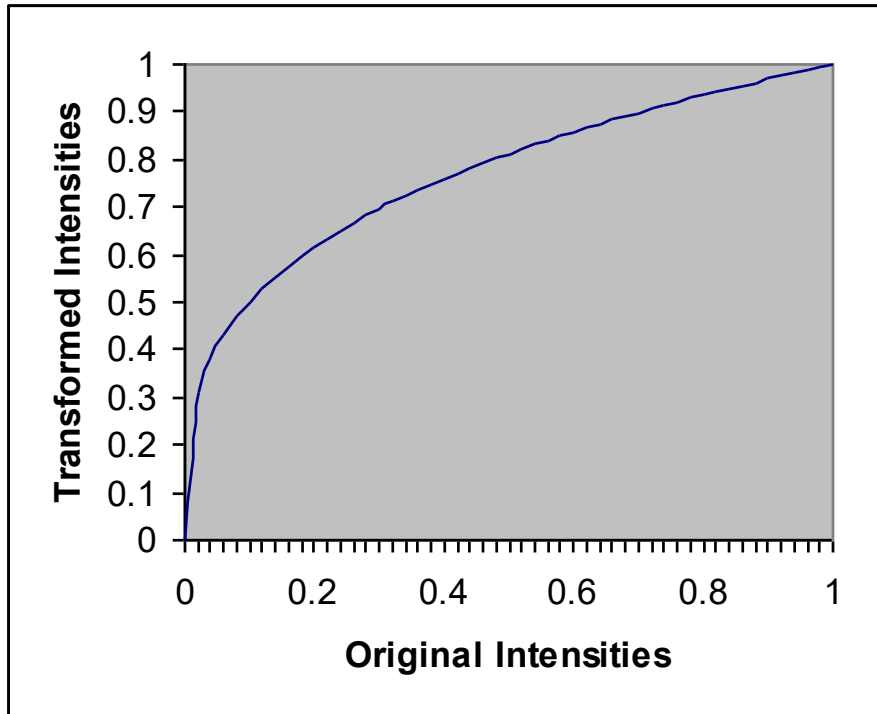
# Power Law Example (cont...)

$$\gamma = 0.4$$



# 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



$$s = r^{0.6}$$



$$s = r^{0.4}$$



$$s = r^{0.3}$$

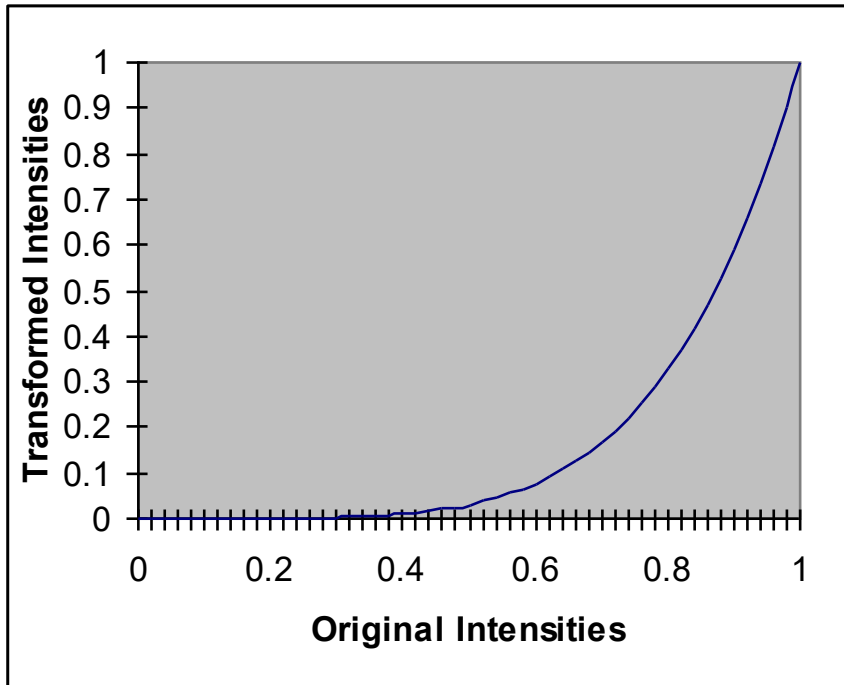


# Power Law Example



# 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



$$s = r^{3.0}$$



$$s = r^{4.0}$$



$$s = r^{5.0}$$

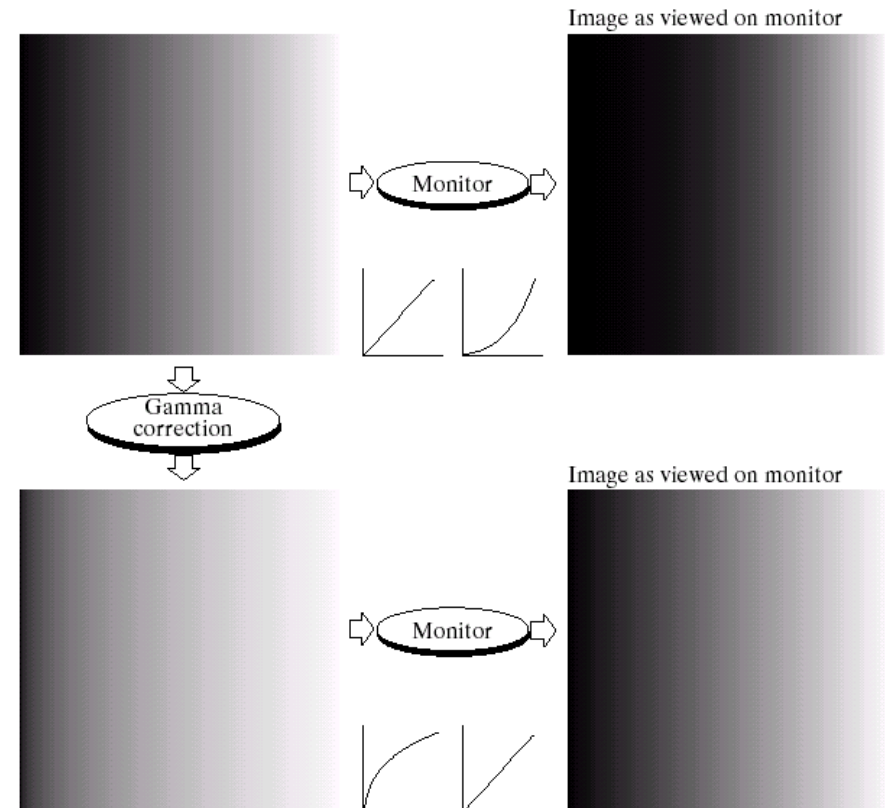


# Gamma Correction

Many of you might be familiar with gamma correction of computer monitors

Problem is that display devices do not respond linearly to different intensities

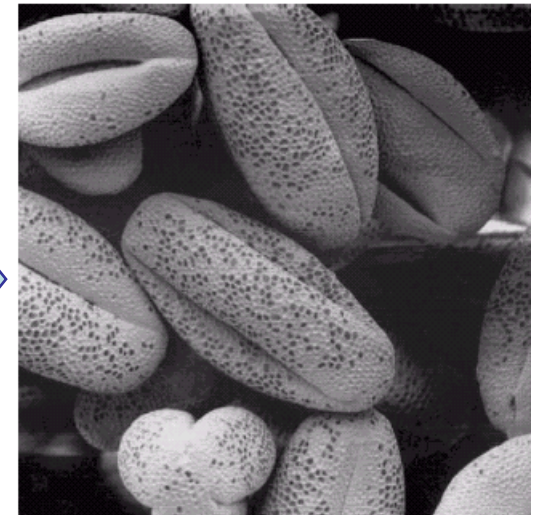
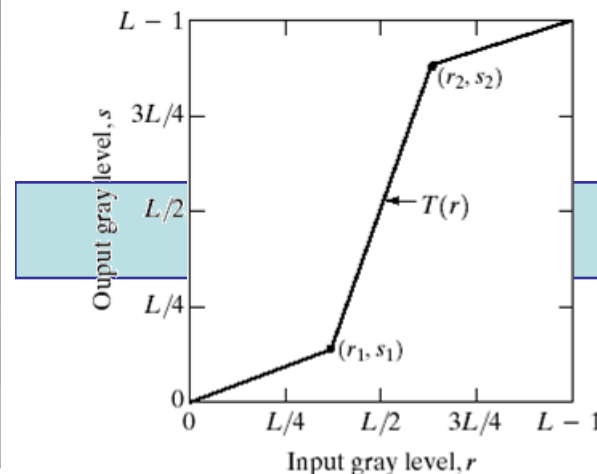
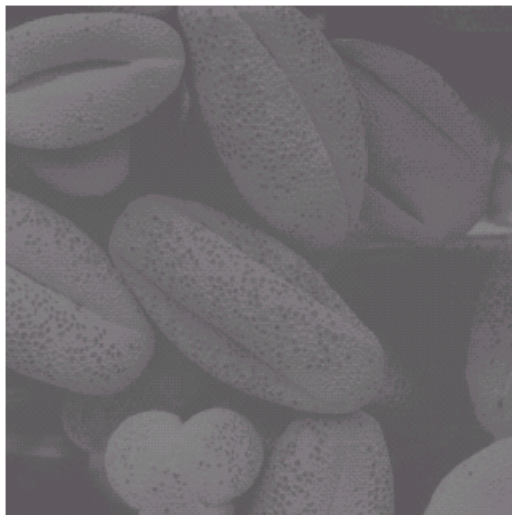
Can be corrected using a log transform



# Piecewise Linear Transformation Functions

Rather than using a well defined mathematical function we can use arbitrary user-defined transforms

The images below show a contrast stretching linear transform to add contrast to a poor quality image

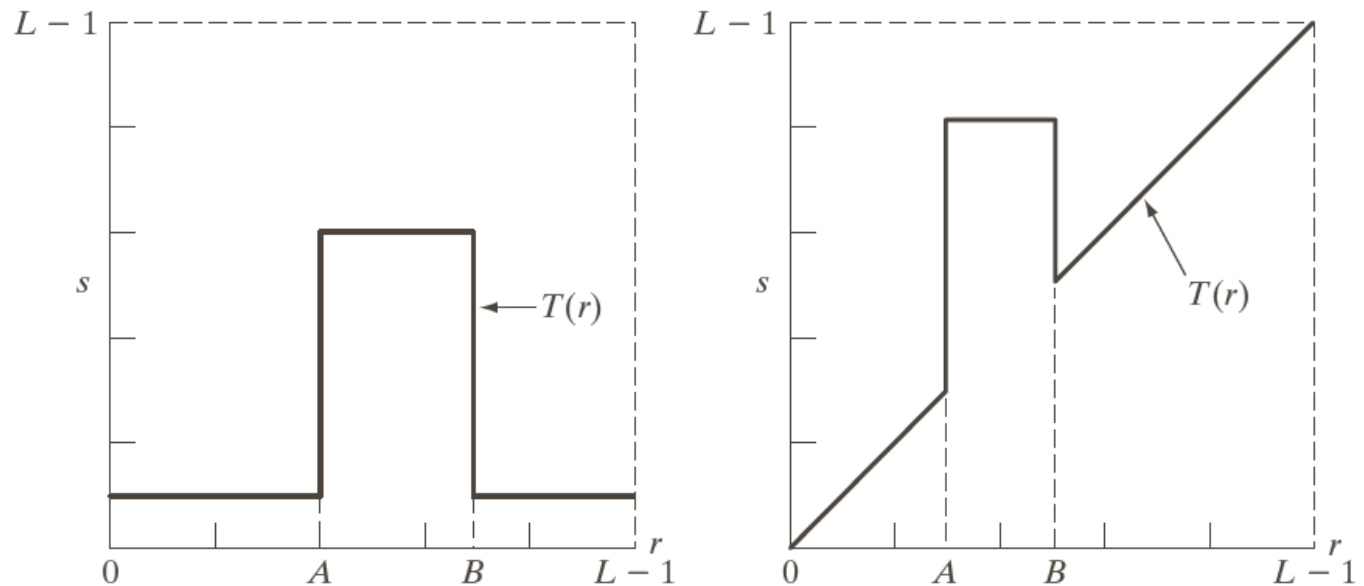




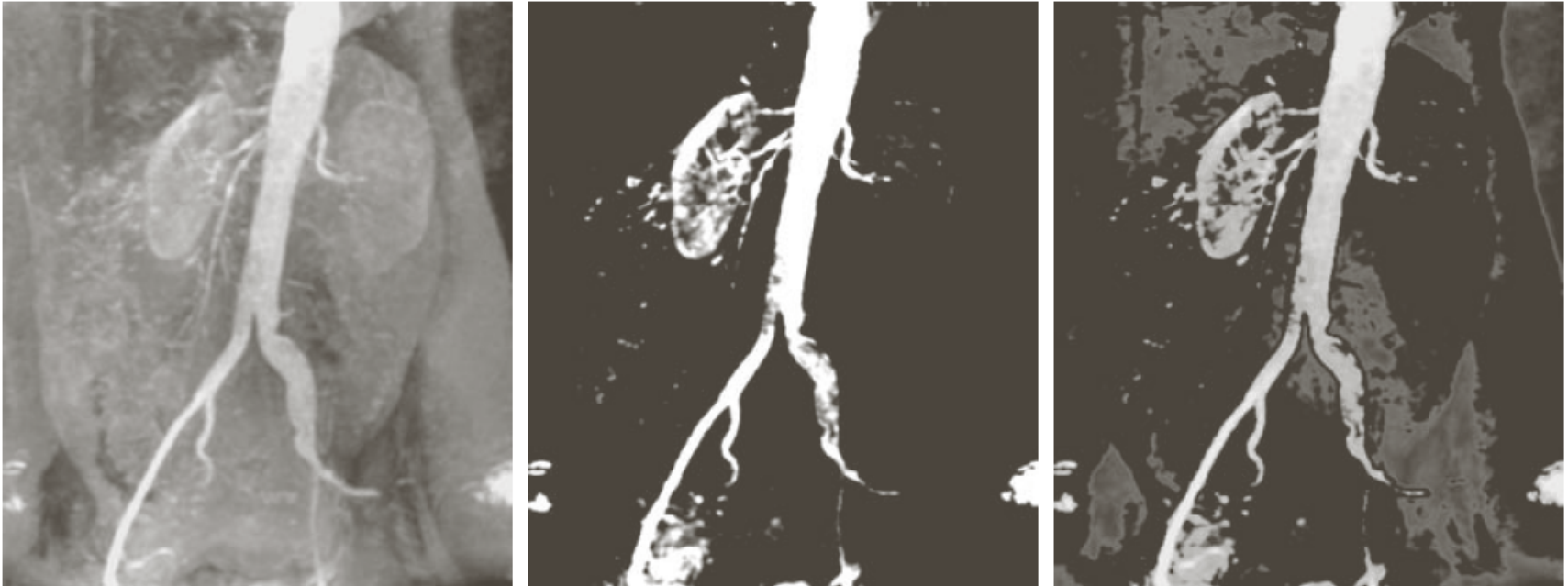
# Piecewise-Linear Transformation (cont...)

a b

**FIGURE 3.11** (a) This transformation highlights intensity range  $[A, B]$  and reduces all other intensities to a lower level. (b) This transformation highlights range  $[A, B]$  and preserves all other intensity levels.



# Piecewise-Linear Transformation (cont...)

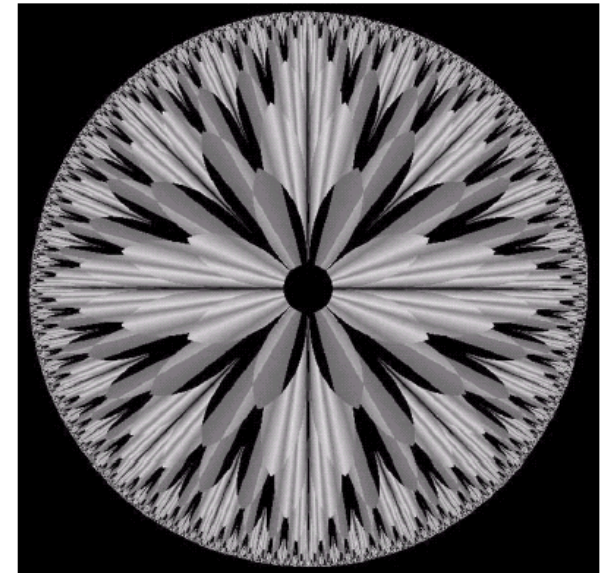
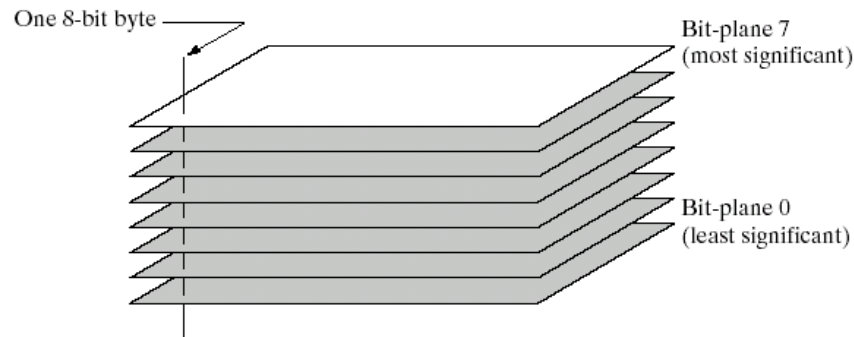


a b c

**FIGURE 3.12** (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected area set to black, 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.)

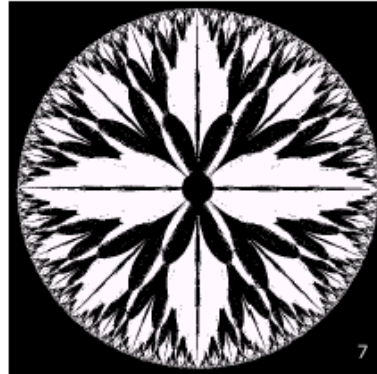
Often by isolating particular bits of the pixel values in an image we can highlight interesting aspects of that image

- Higher-order bits usually contain most of the significant visual information
- Lower-order bits contain subtle details

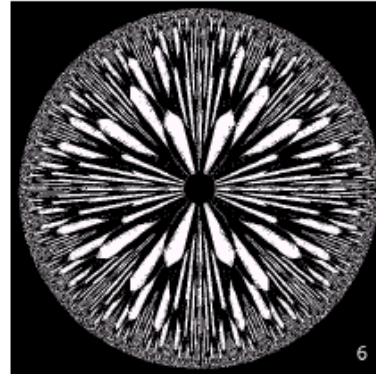


# Bit Plane Slicing (cont...)

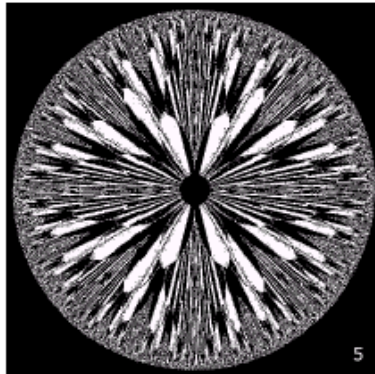
[10000000]



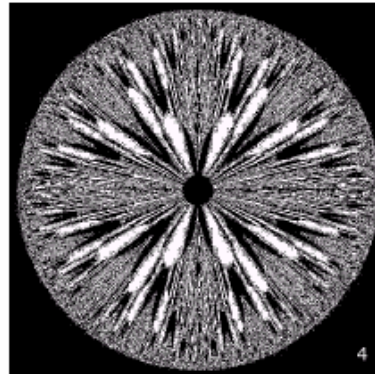
[01000000]



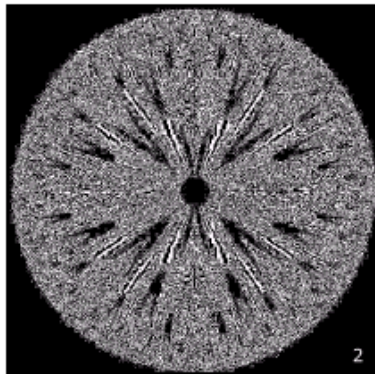
[00100000]



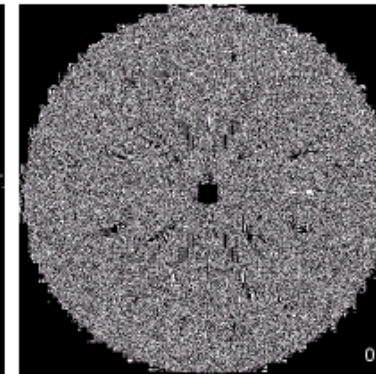
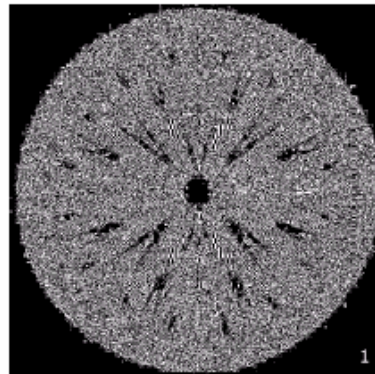
[00001000]



[00000100]



[00000001]





# Bit-Plane Slicing (cont...)



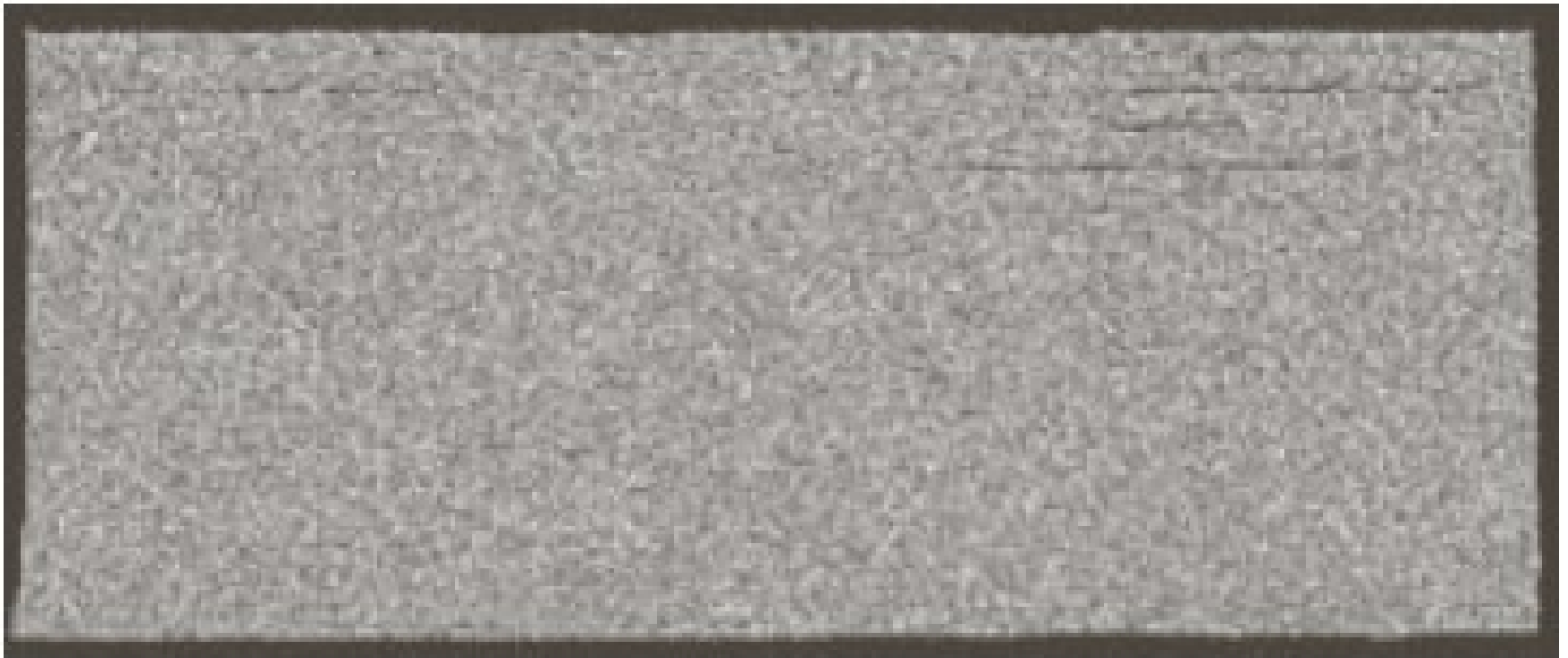
a	b	c
d	e	f
g	h	i

**FIGURE 3.14** (a) An 8-bit gray-scale image of size  $500 \times 1192$  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.

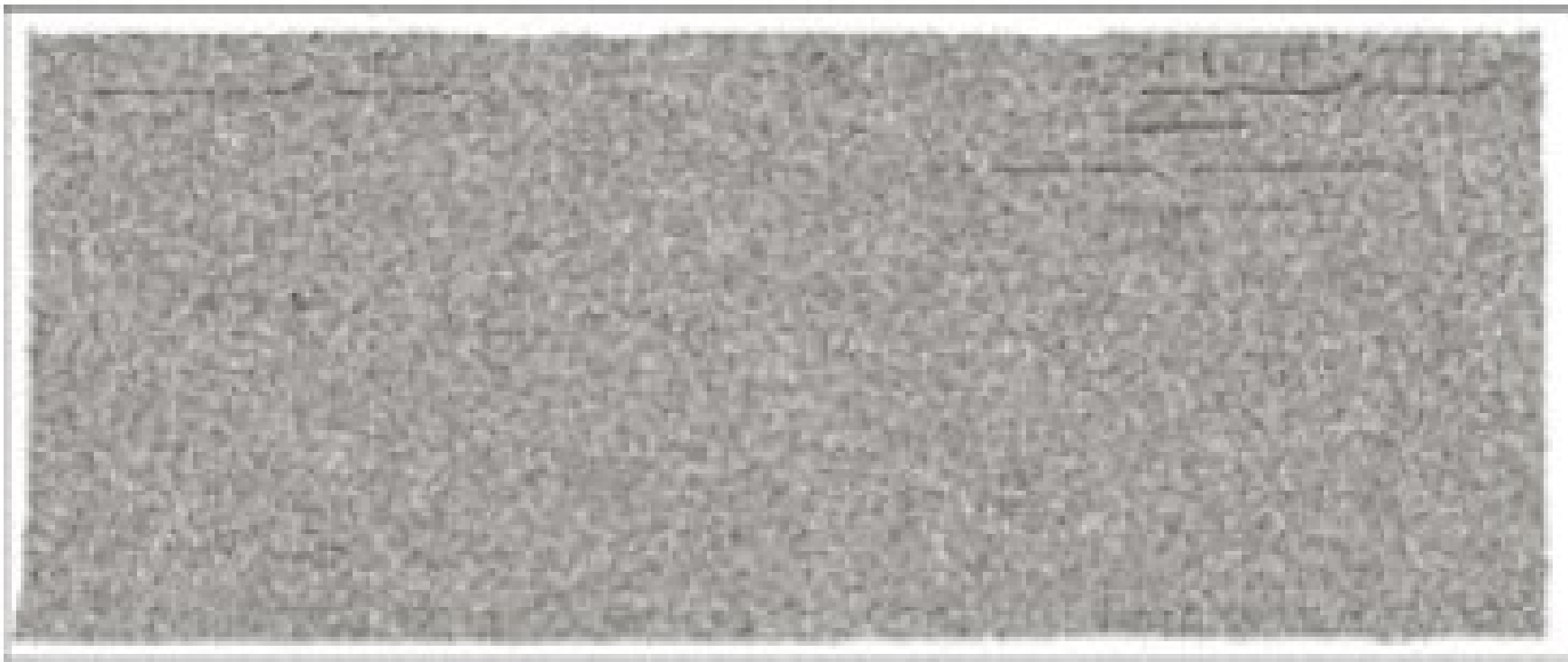
# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)

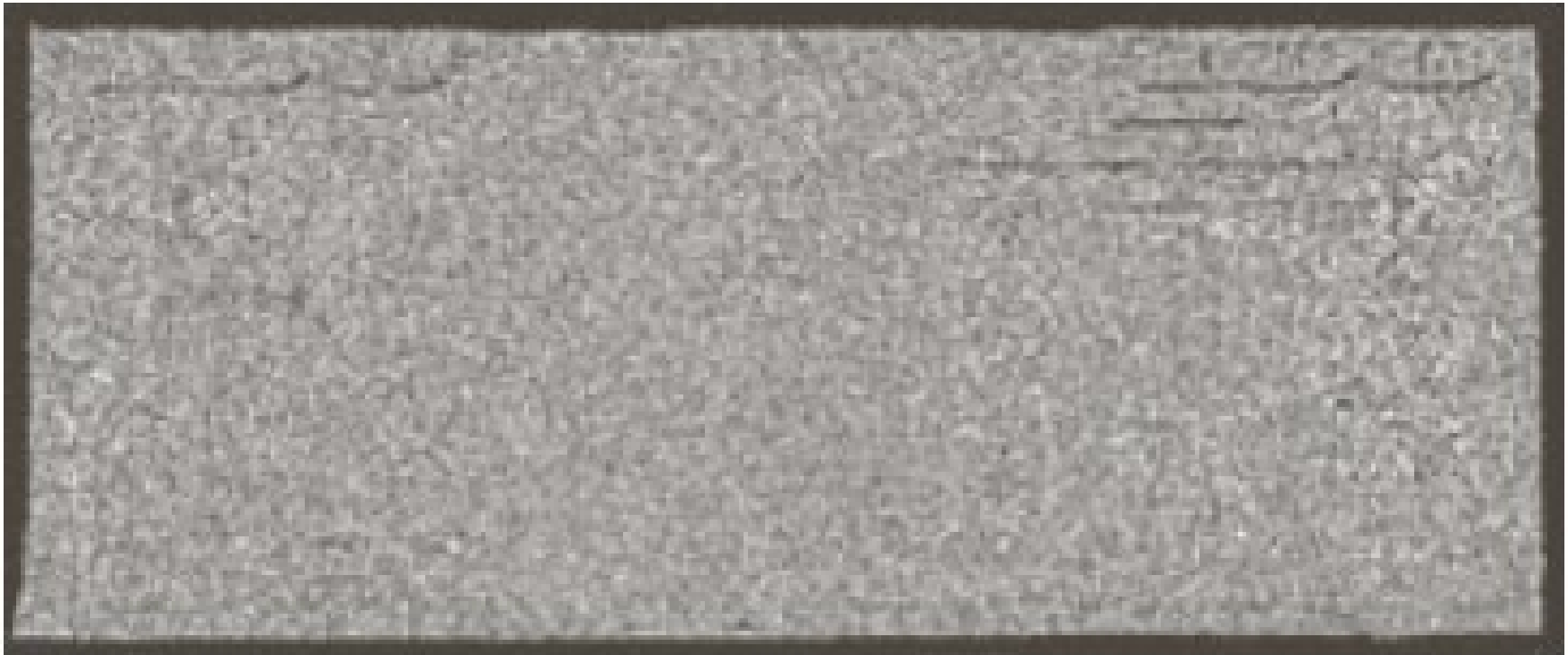


# Bit Plane Slicing (cont...)

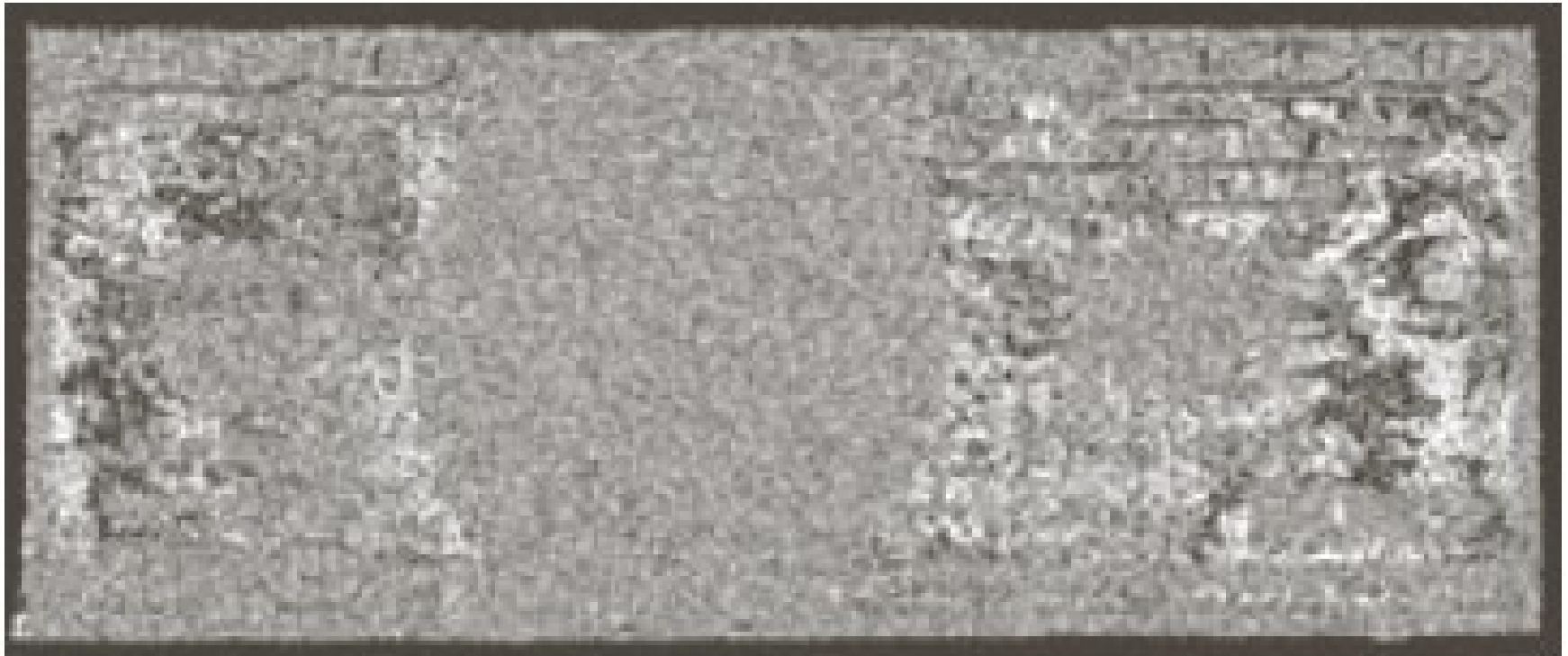




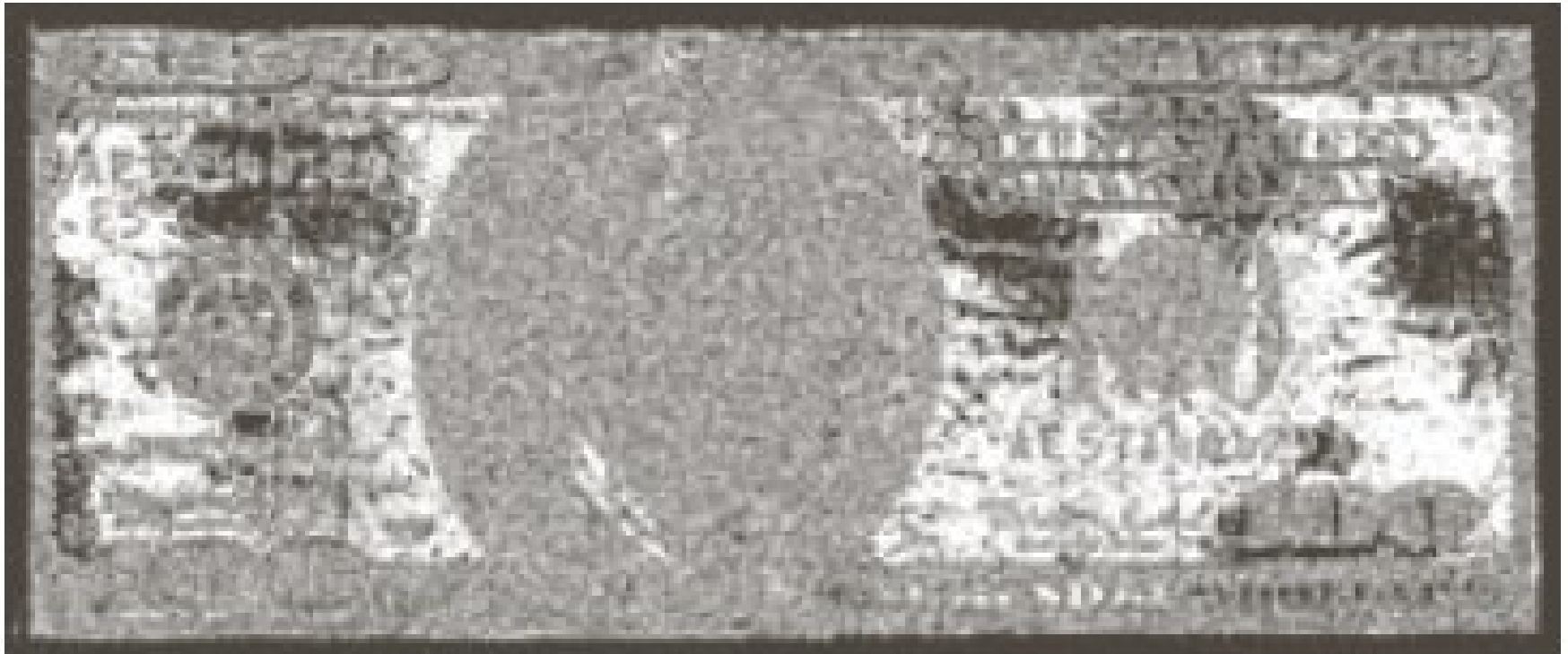
# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit Plane Slicing (cont...)



# Bit-Plane Slicing (cont...)



a b c

**FIGURE 3.15** Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5. Compare (c) with Fig. 3.14(a).

Useful for compression.

Reconstruction is obtained by:

$$I(i, j) = \sum_{n=1}^N 2^{n-1} I_n(i, j)$$

We have looked at different kinds of point processing image enhancement

Next time we will start to look at histogram processing methods.