# Image Processing
# (Lecture 3)

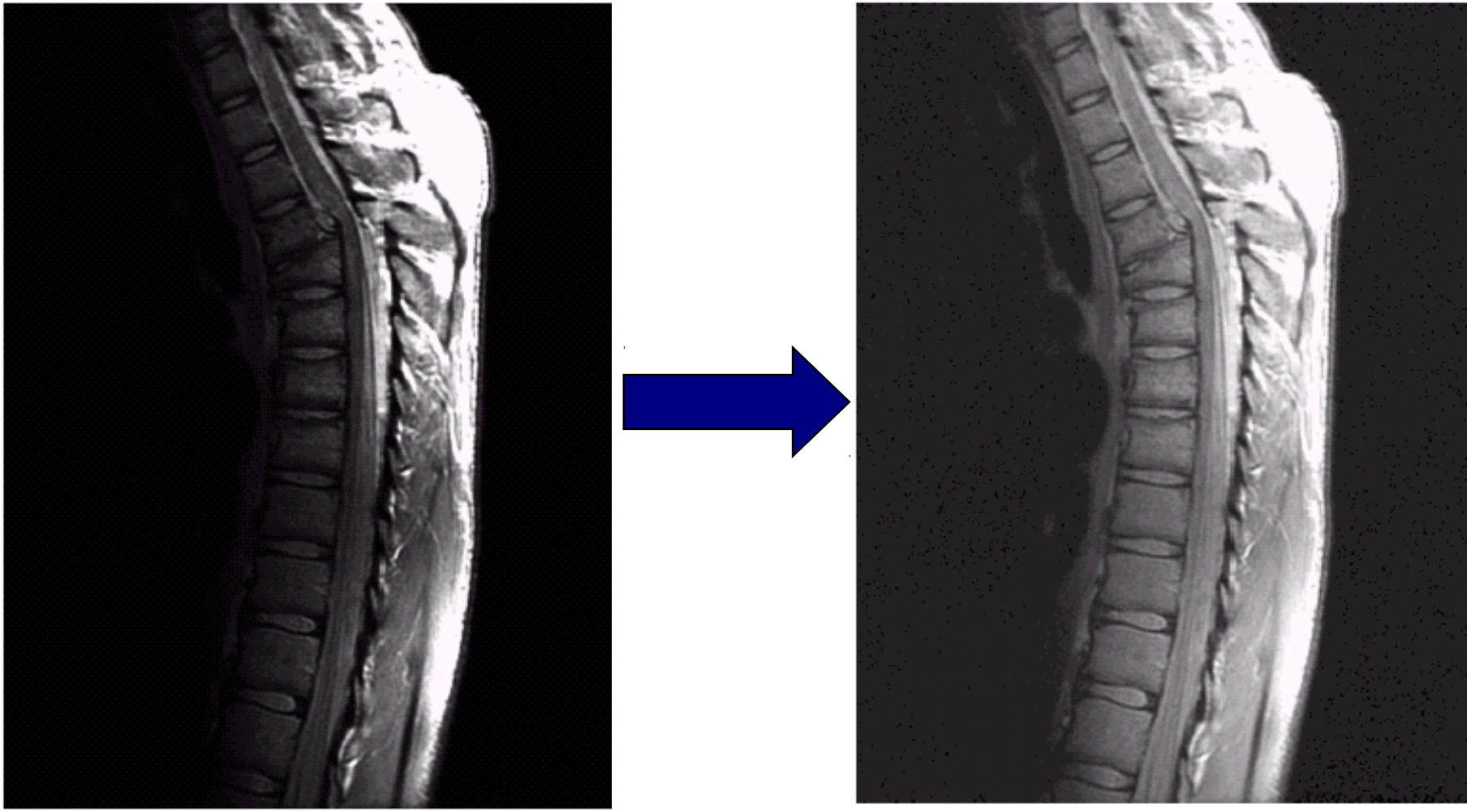
# What Is Image Enhancement?

Image enhancement is the process of making images more useful

The reasons for doing this include:

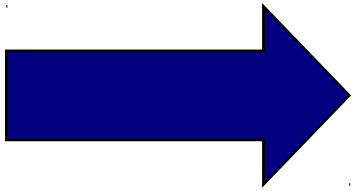
- Highlighting interesting detail in images
- Removing noise from images
- Making images more visually appealing

# Image Enhancement Examples



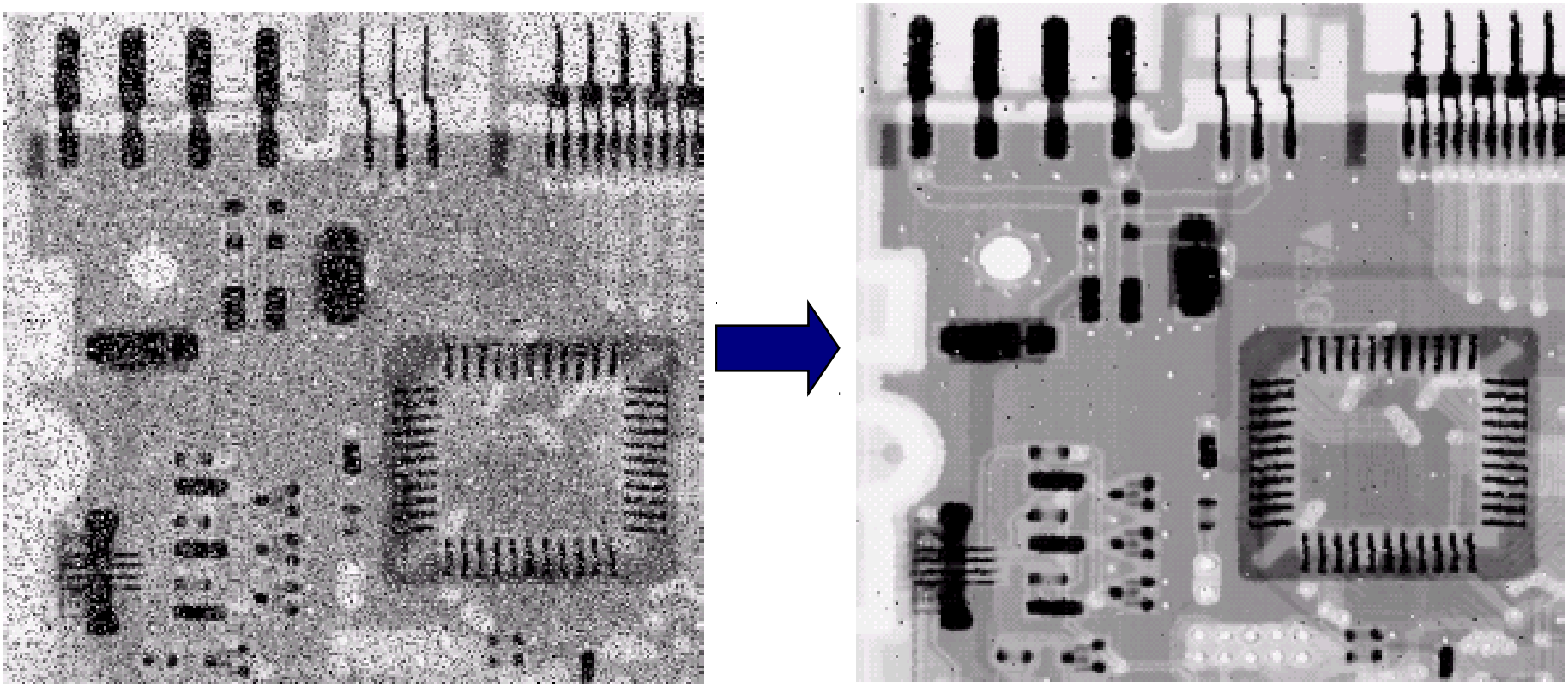
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Enhancement Examples (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Enhancement Examples (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Enhancement Examples (cont…)

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Spatial & Frequency Domains

There are two broad categories of image enhancement techniques

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

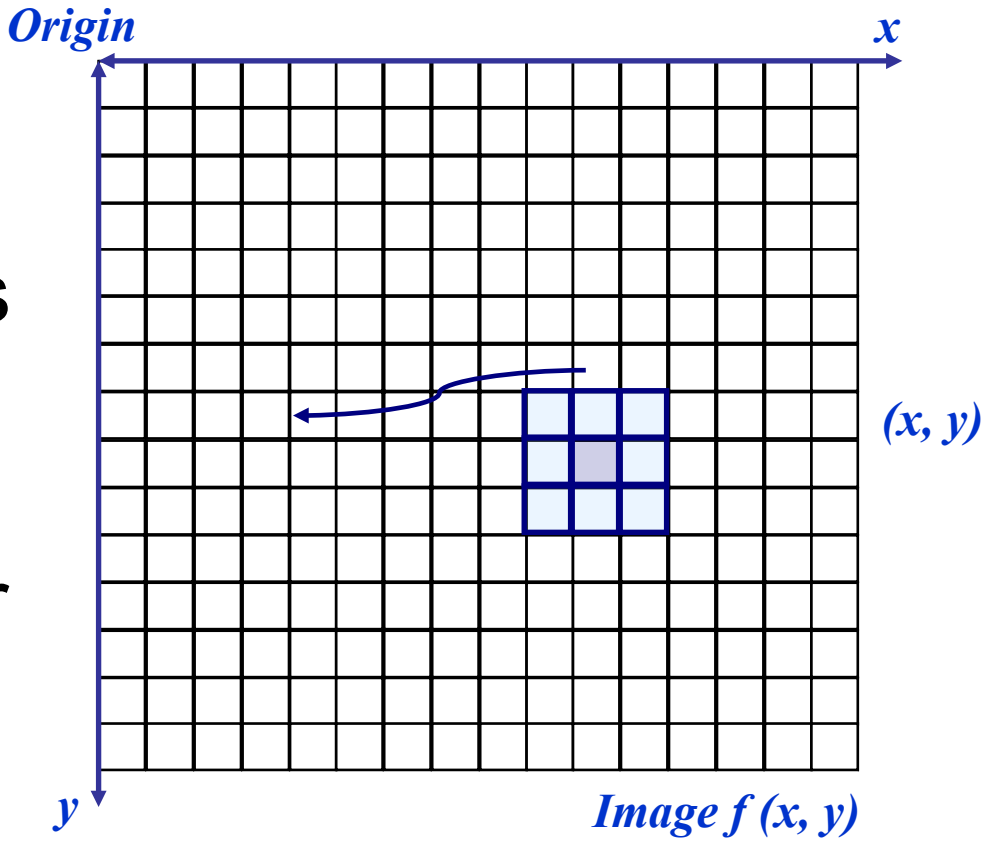
For the moment we will concentrate on techniques that operate in the spatial domain

# 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 $(x, y)$



# Point Processing

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

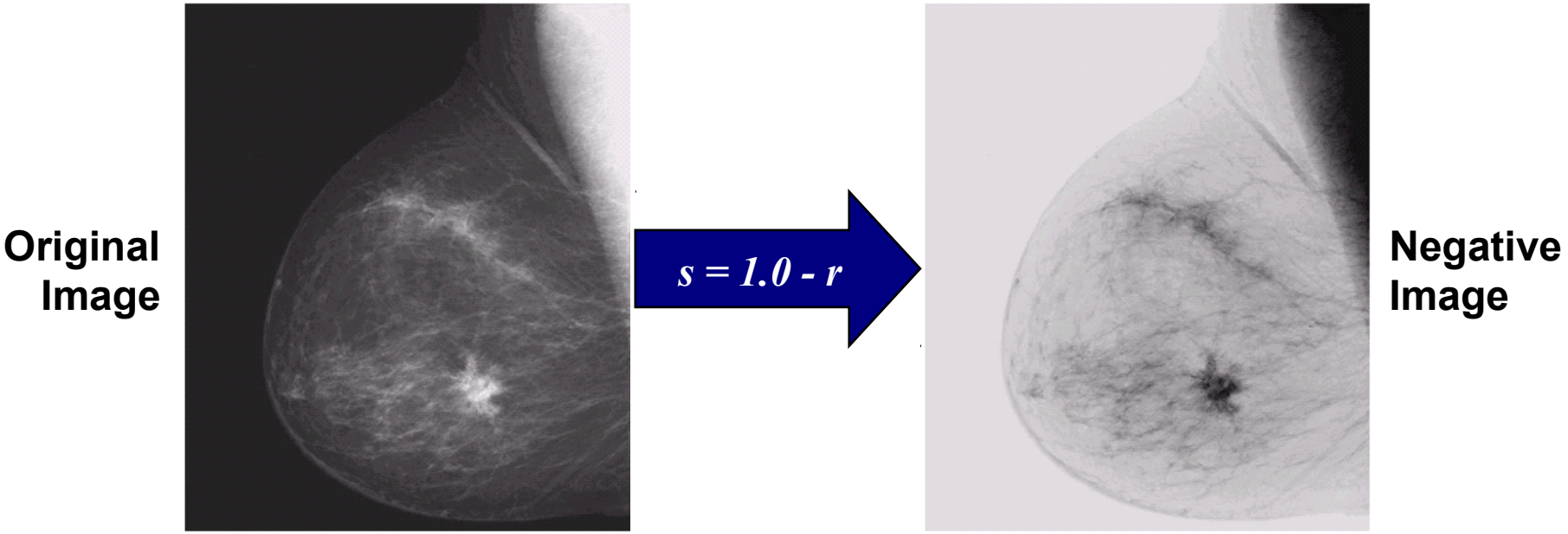
$$s = T(r)$$

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

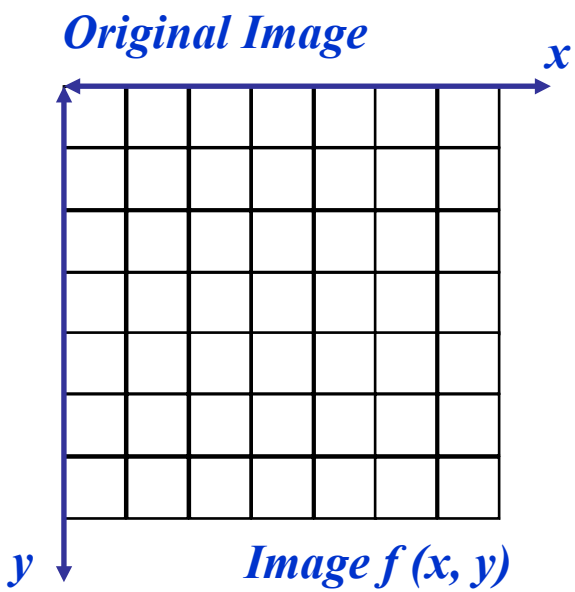
# Point Processing Example: Negative Images

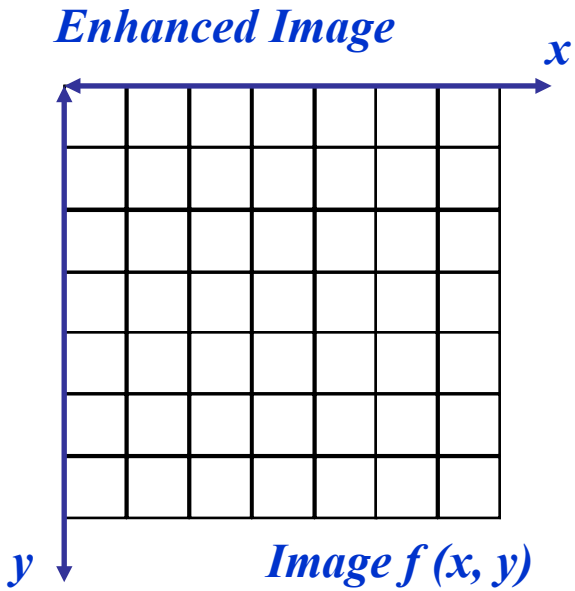
Negative images are useful for enhancing white or grey detail embedded in dark regions of an image

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



# Point Processing Example: Negative Images (cont…)

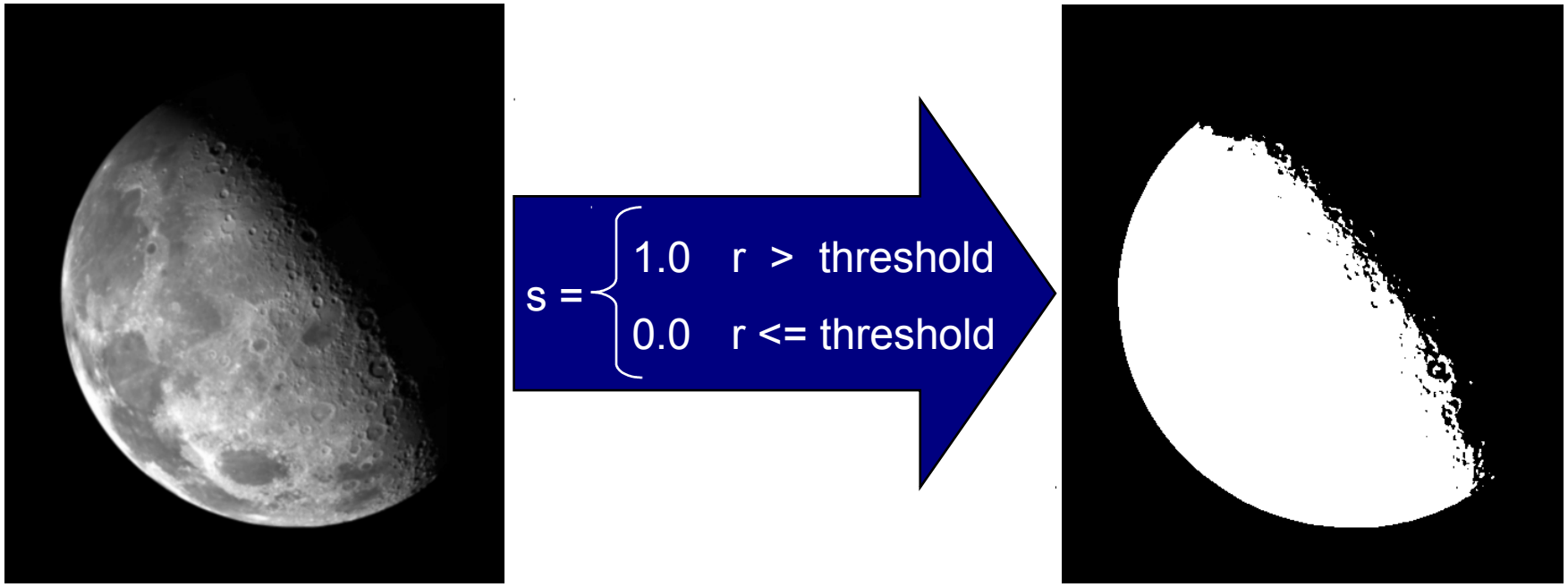


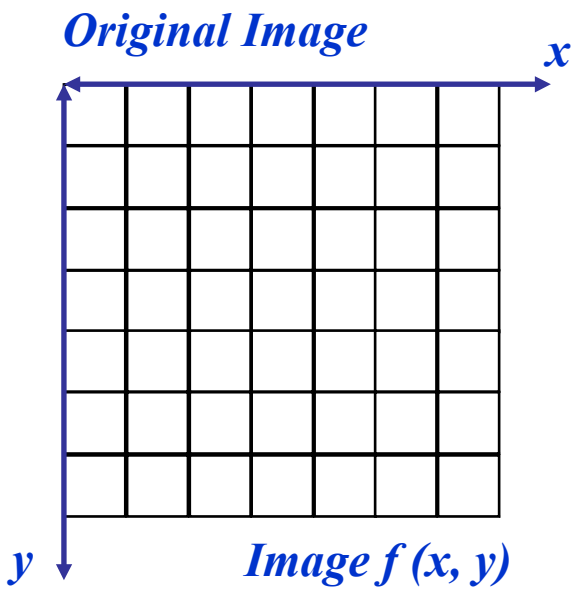

$$s = intensity_{max} - r$$

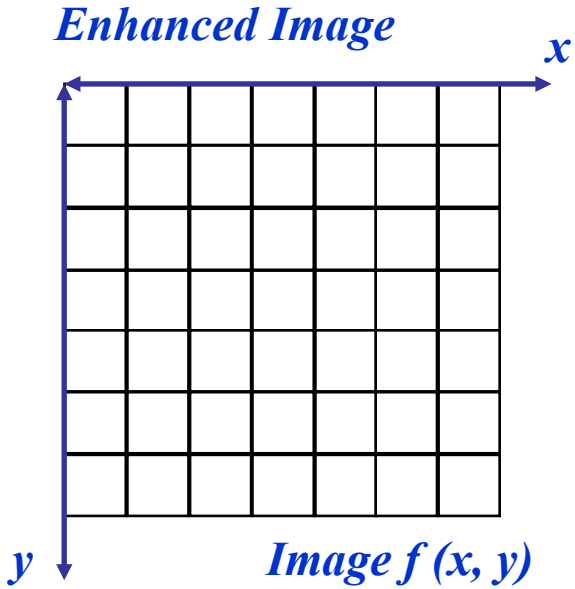
# Point Processing Example: Thresholding

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



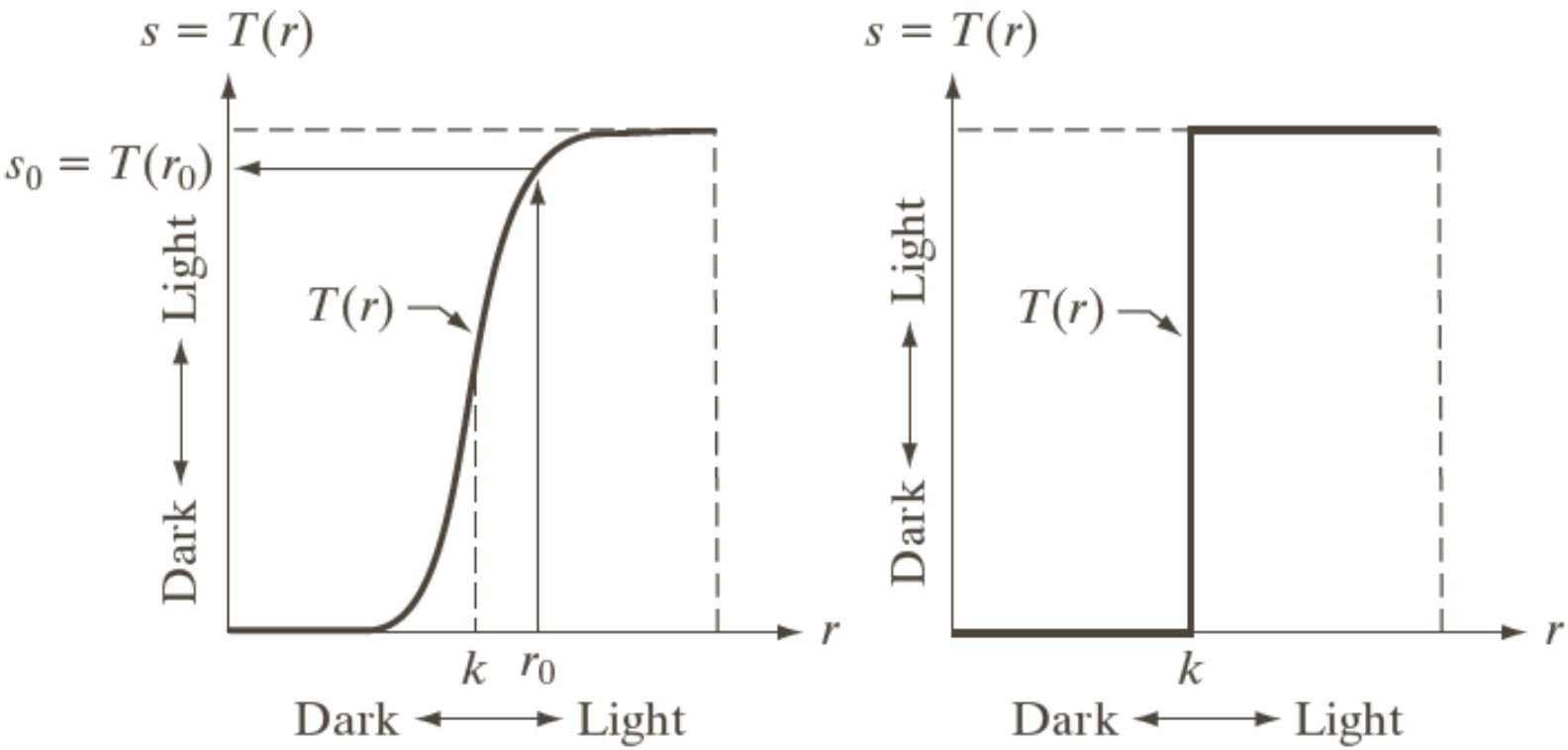
# Point Processing Example: Thresholding (cont…)




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

# Intensity Transformations

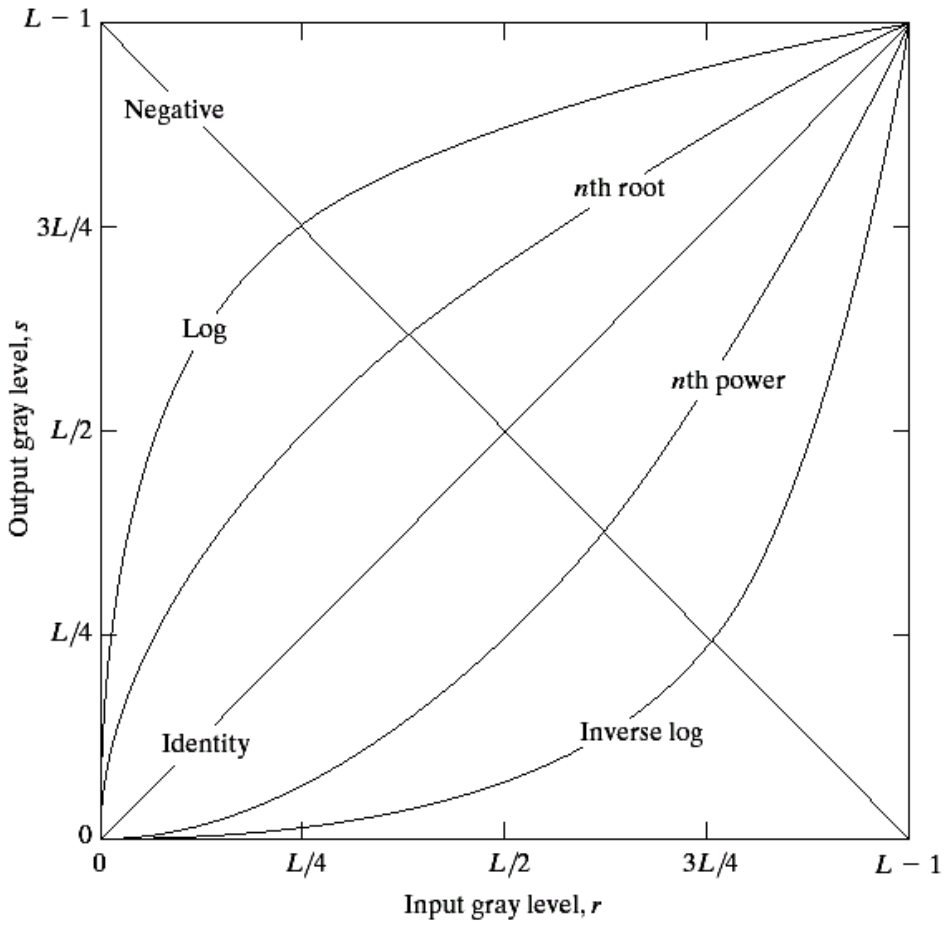


# Basic Grey Level Transformations

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



# 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

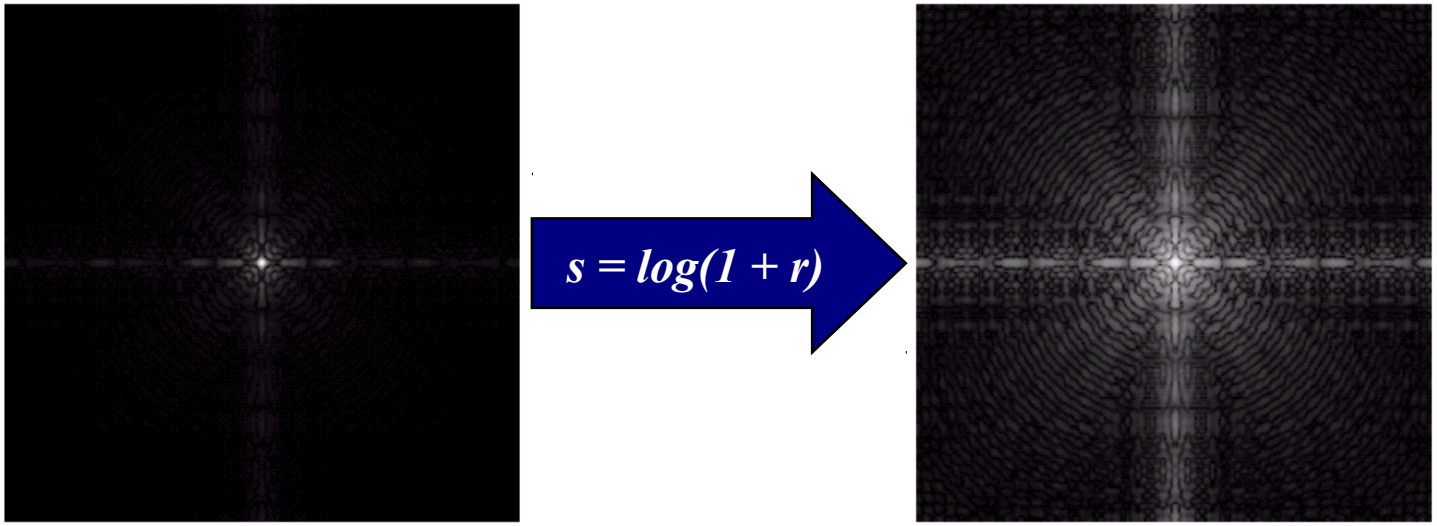
The inverse log transformation performs the opposite transformation

Compresses the dynamic range of images with large variations in pixel values

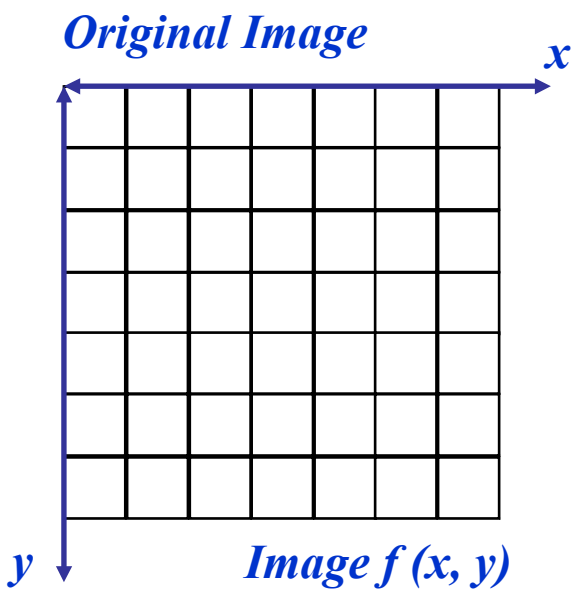
# Logarithmic Transformations (cont…)

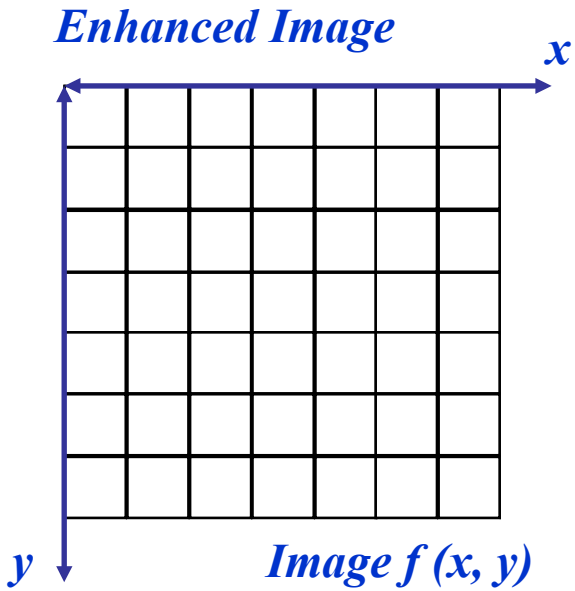
Log functions are particularly useful when the input grey level values may have an extremely large range of values

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



# Logarithmic Transformations (cont…)




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

We usually set $c$ to 1

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

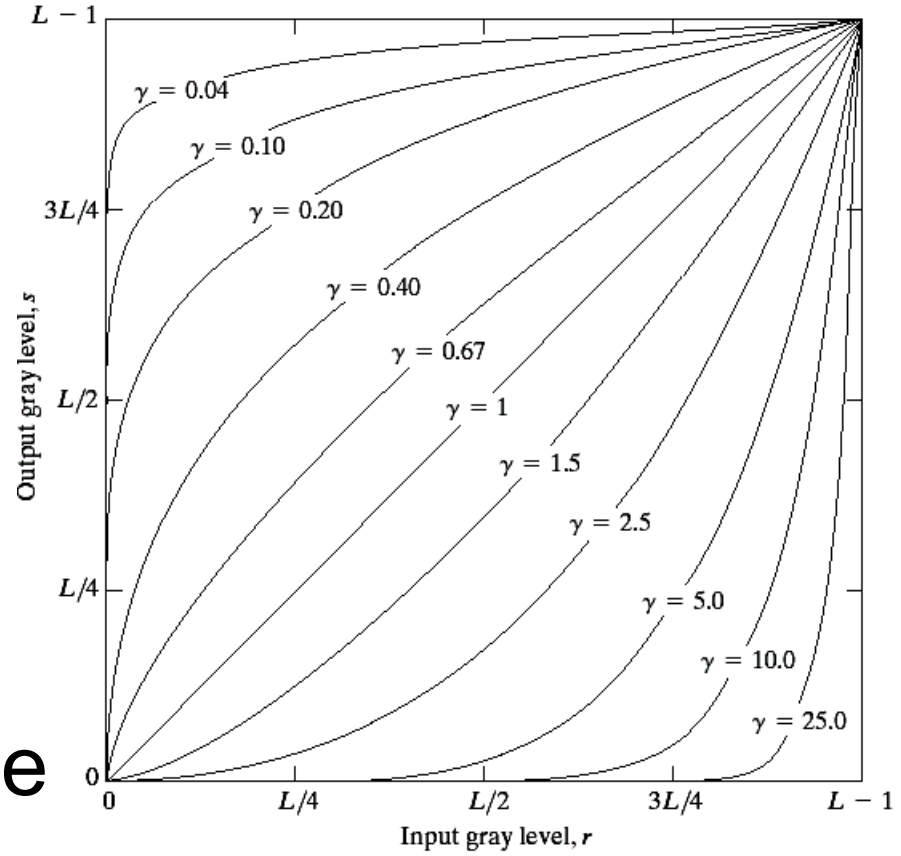
# Power Law Transformations

Power law transformations have the following form

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

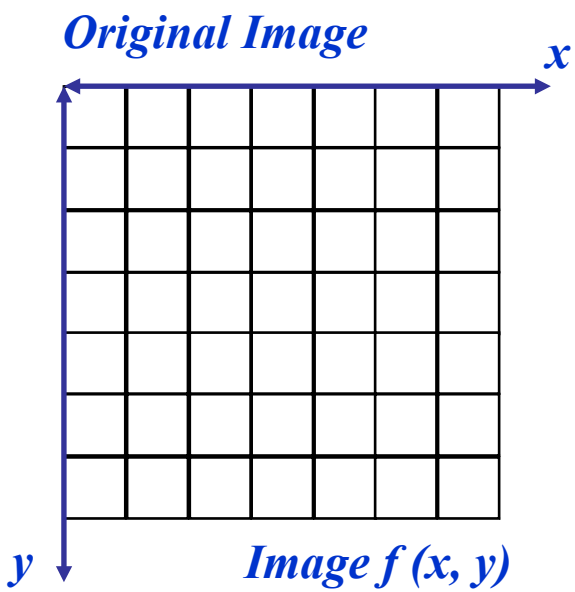
Map a narrow range of dark input values into a wider range of output values or vice versa

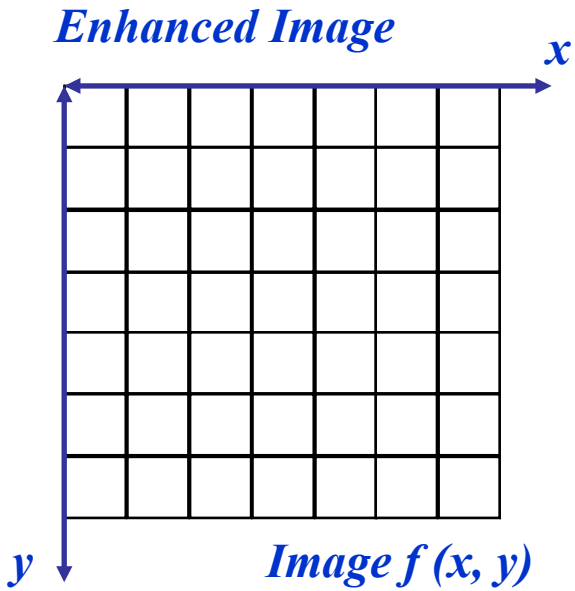
Varying □ gives a whole family of curves




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Power Law Transformations (cont…)




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

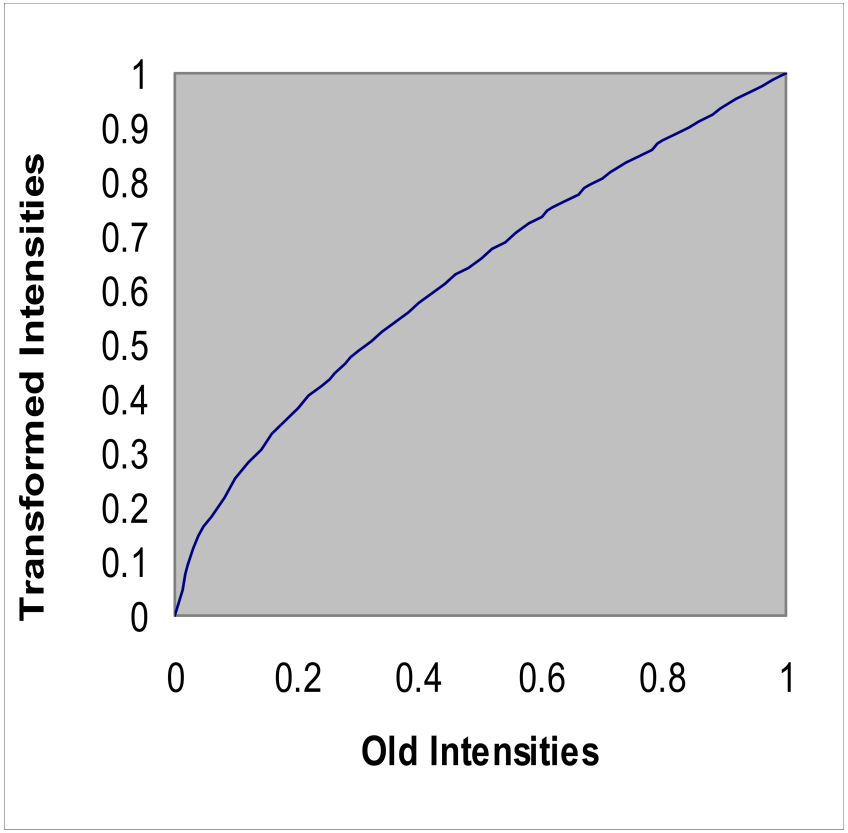
We usually set $c$ to 1

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

# Power Law Example

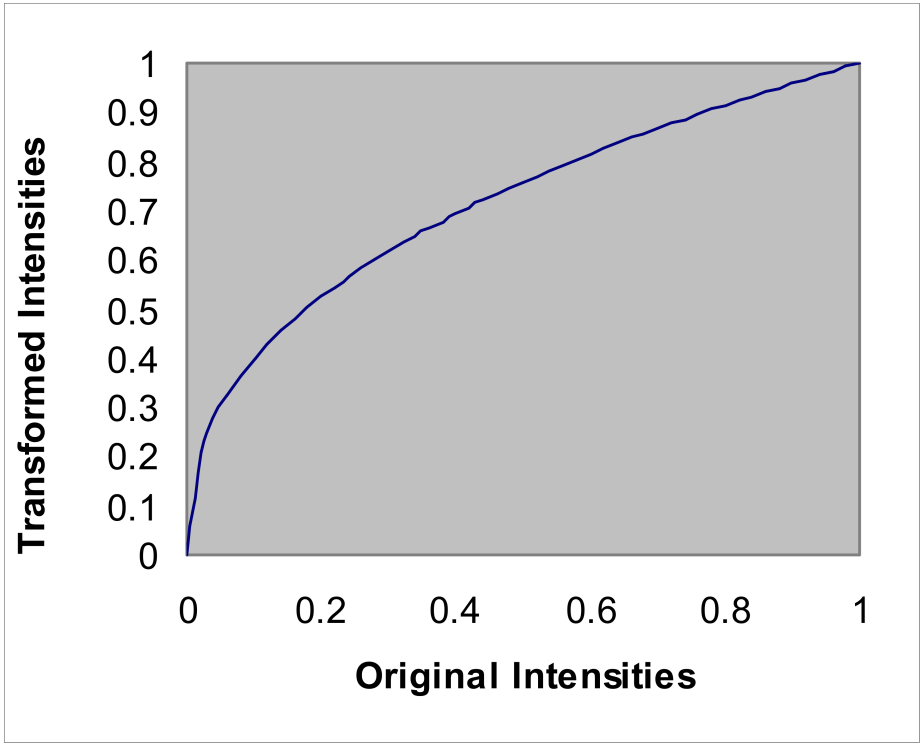
# Power Law Example (cont…)

$$\gamma = 0.6$$



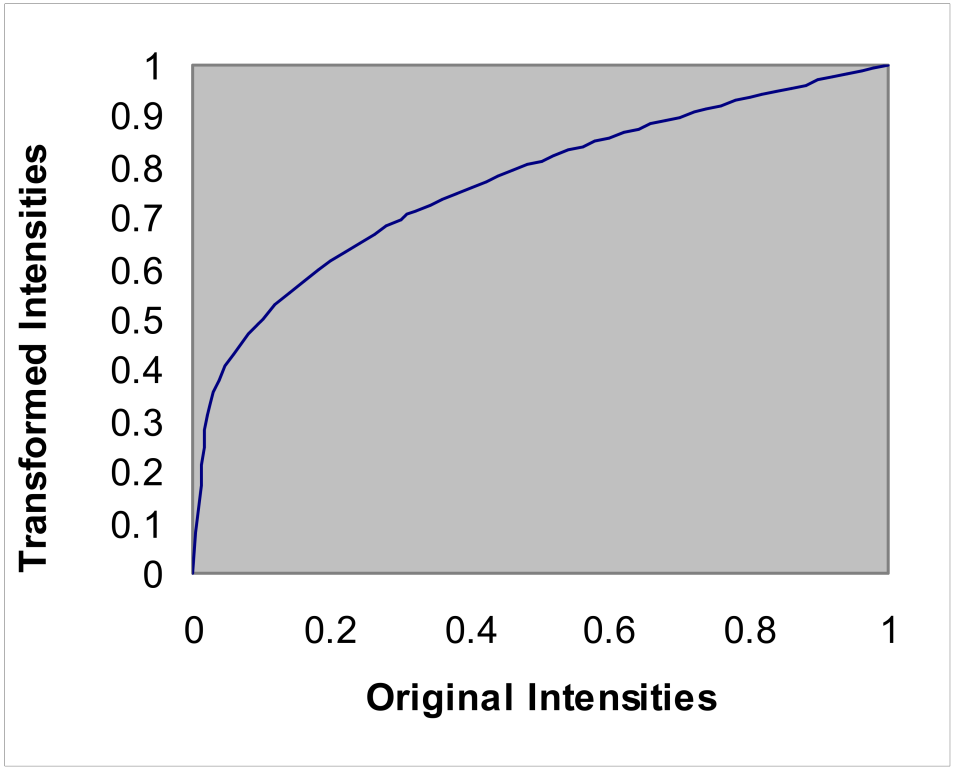

# Power Law Example (cont…)

$$\gamma = 0.4$$




# Power Law Example (cont…)

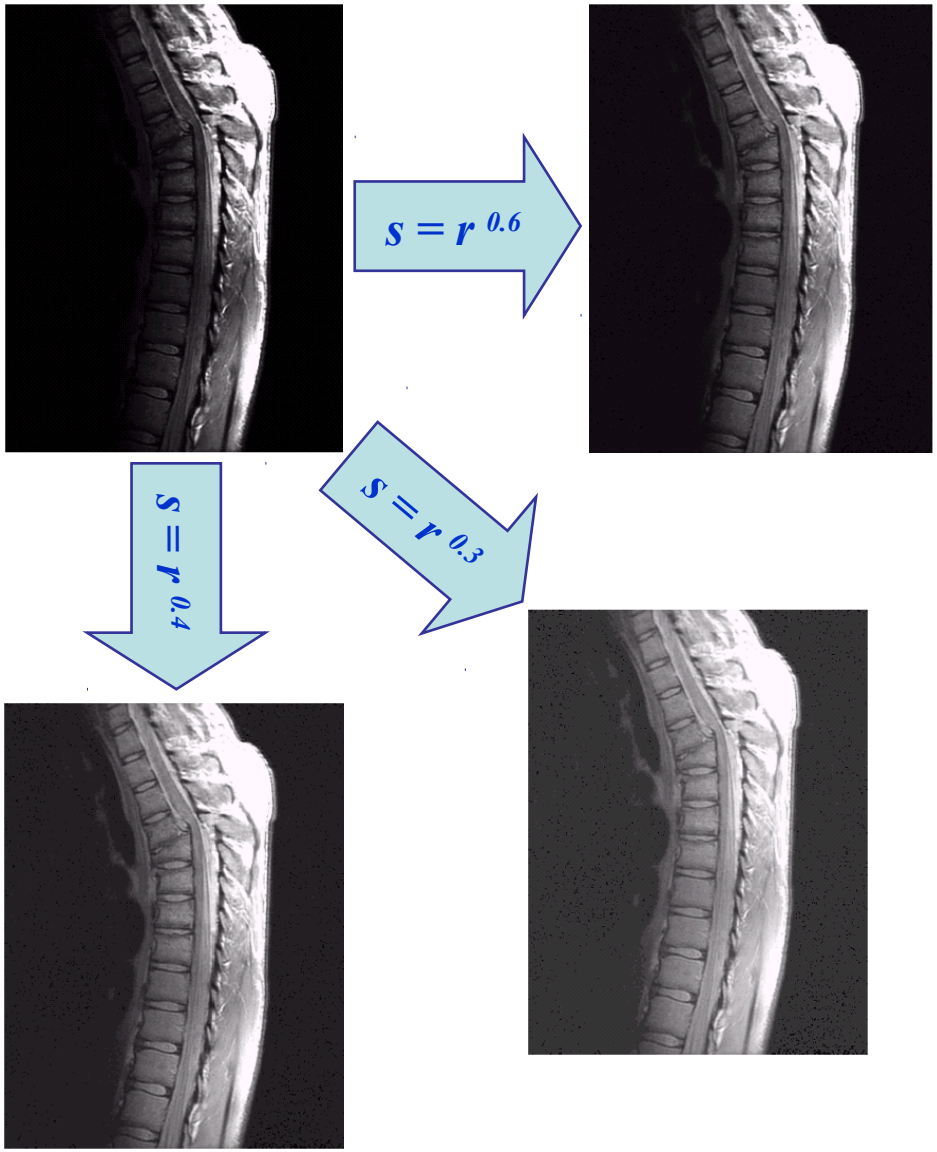
$$\gamma = 0.3$$




# 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

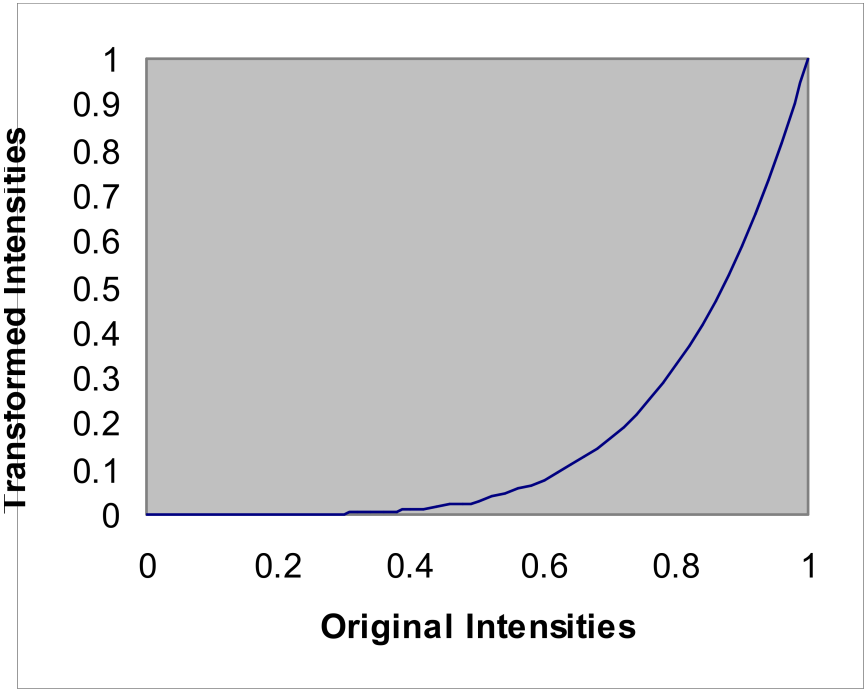



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Power Law Example

# Power Law Example (cont…)

$$\gamma = 5.0$$

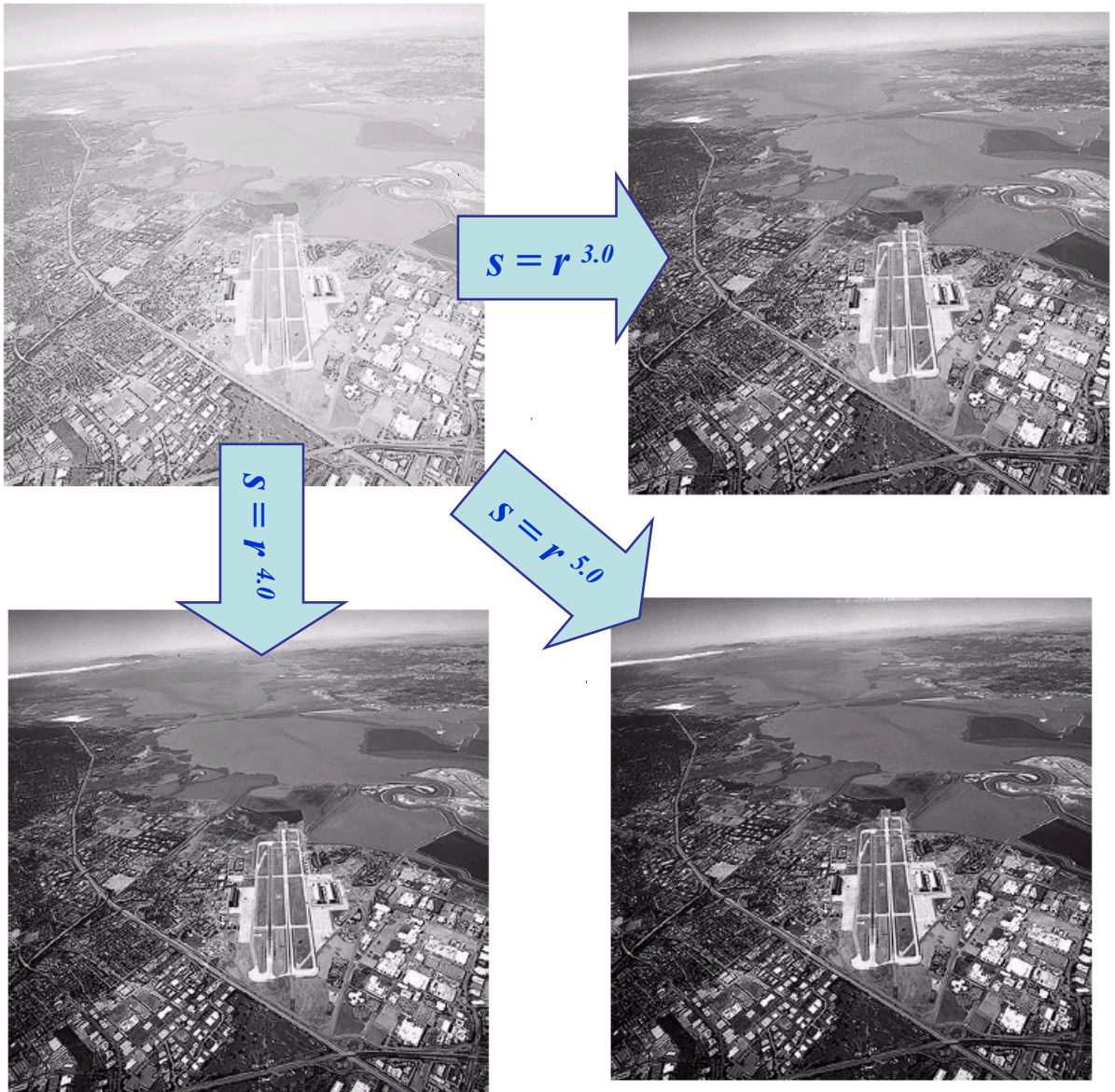



# Power Law Transformations (cont…)

An aerial photo of a runway is shown

This time power law transforms are used to darken the image

Different curves highlight different detail



# Gamma Correction

Many devices used for image capture, display and printing respond according to a power law

• The exponent in the power-law equation is referred to as gamma

• The process of correcting for the power-law response is referred to as gamma correction

• Example: – CRT devices have an intensity-to-voltage response that is

a power function (exponents typically range from 1.8-2.5)

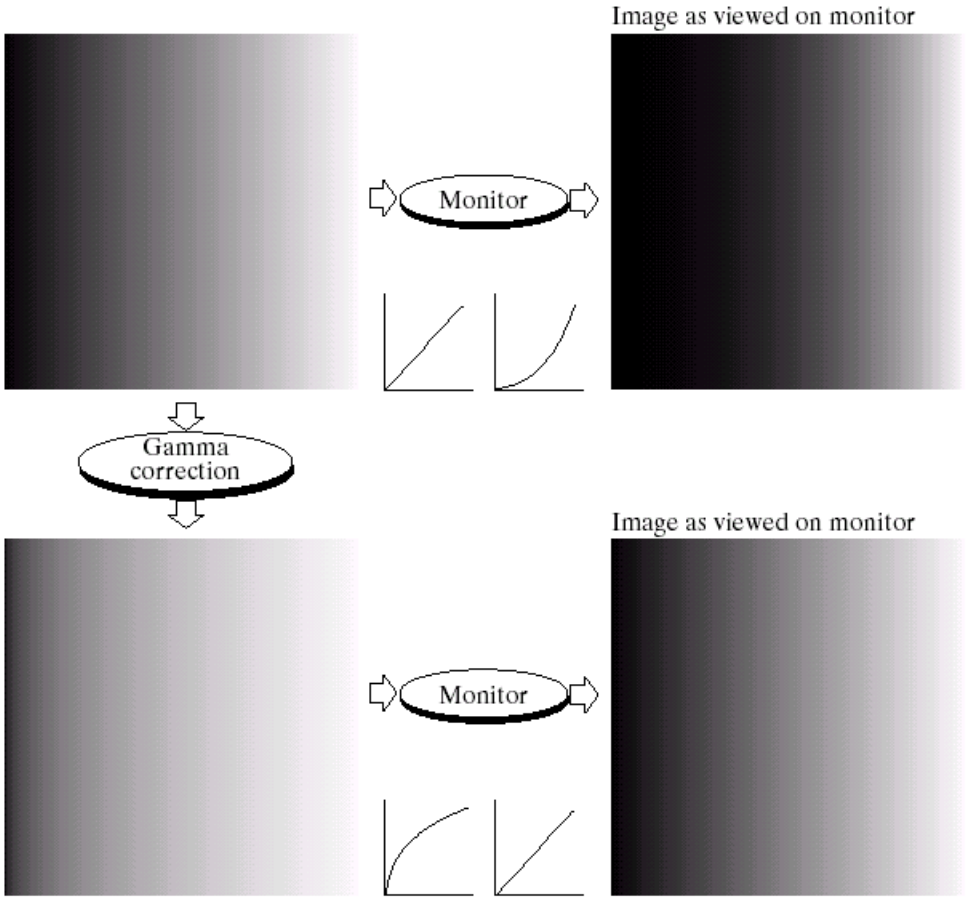
– Gamma correction in this case could be achieved by applying the transformation $s=r_1/2.5=r^{0.4}$

# Gamma Correction

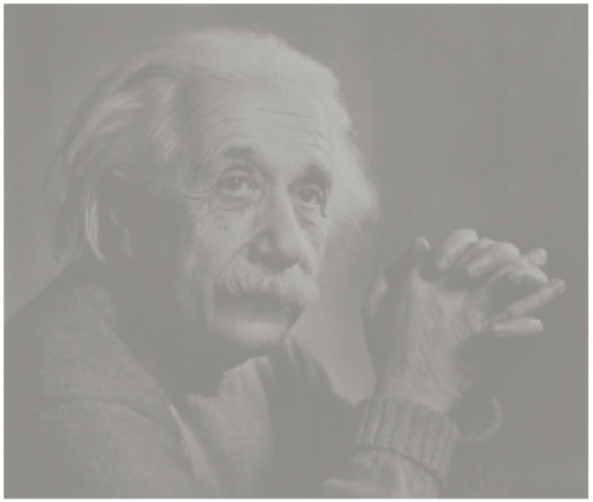
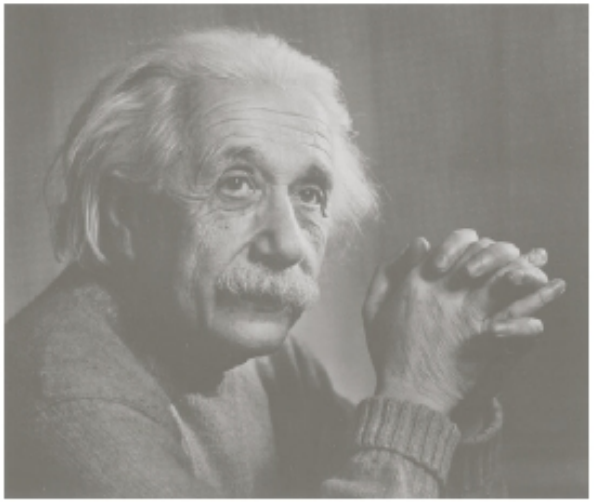
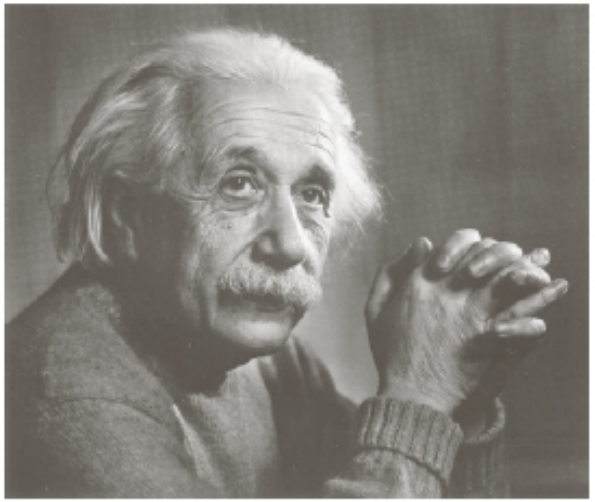
Many of you might be familiar with gamma correction of computer monitors

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

Can be corrected using a log transform



# More Contrast Issues

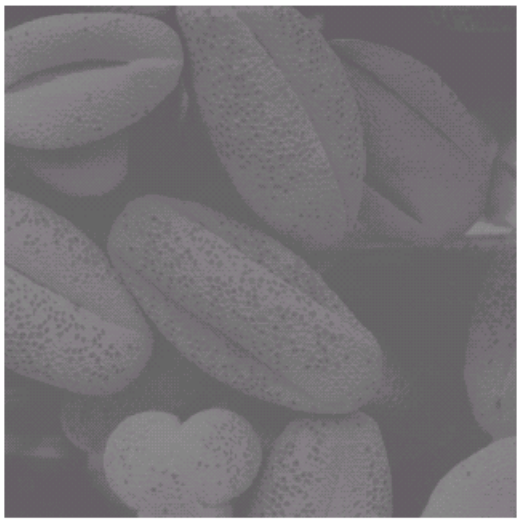




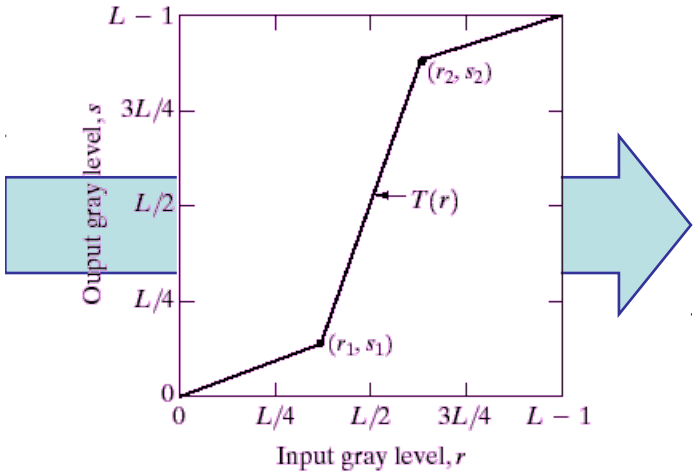
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

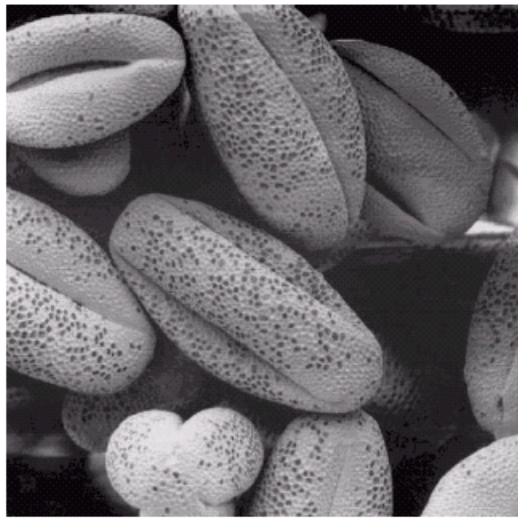
# Piecewise Linear Transformation Functions

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 Functions

- Rather than using a well defined mathematical function we can use arbitrary user-defined transforms
- Contrast stretching expands the range of intensity levels in an image so it spans a given (full) intensity range
- Control points (r1,s1) and (r2,s2) control the shape of the transform T(r)
- $r_1=r_2$, $s_1=0$ and $s_2=L-1$ yields a thresholding function

The contrast stretched image shown in the previous slide is obtained using the transformation obtained from the equation of the line having following points

- $(r_1,s_1)=(r_{min},0)$ and $(r_2, s_2)=(r_{max},L-1)$

# Gray Level Slicing

• Used to highlight a specific range of intensities in an image that might be of interest

•Two common approaches

– Set all pixel values within a range of interest to one value (white) and all others to another value (black)

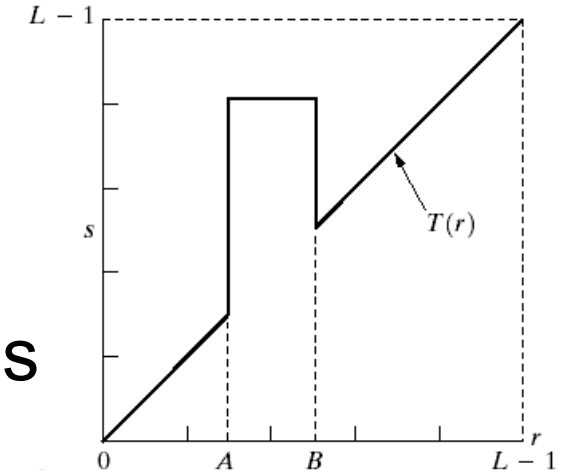
Produces a binary image

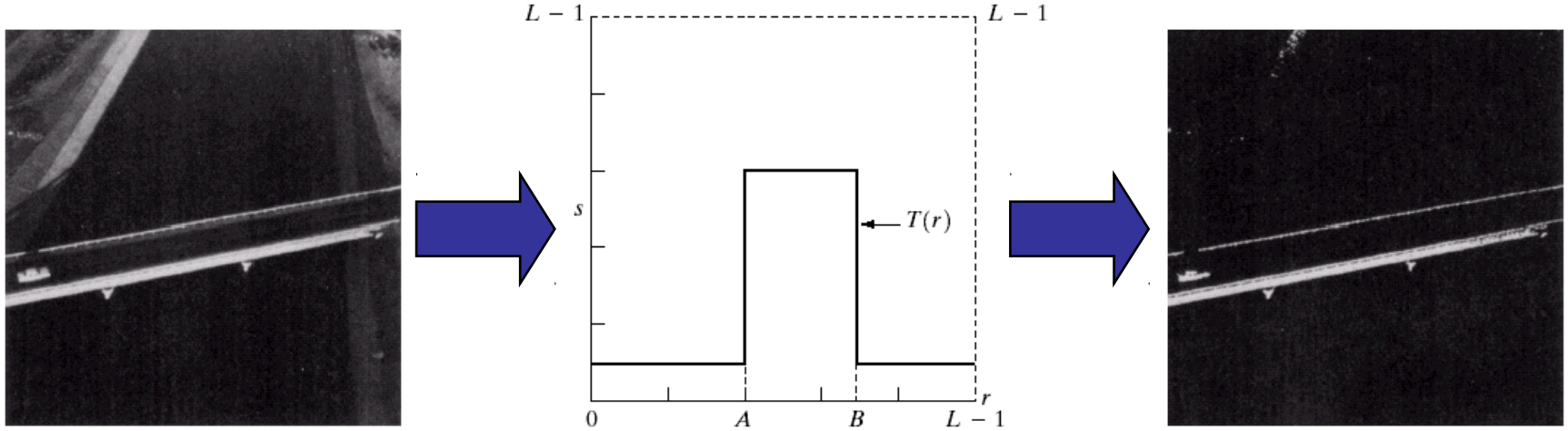
– Brighten (or darken) pixel values in a range of interest and leave all others unchanged

# Gray Level Slicing

Highlights a specific range of grey levels

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



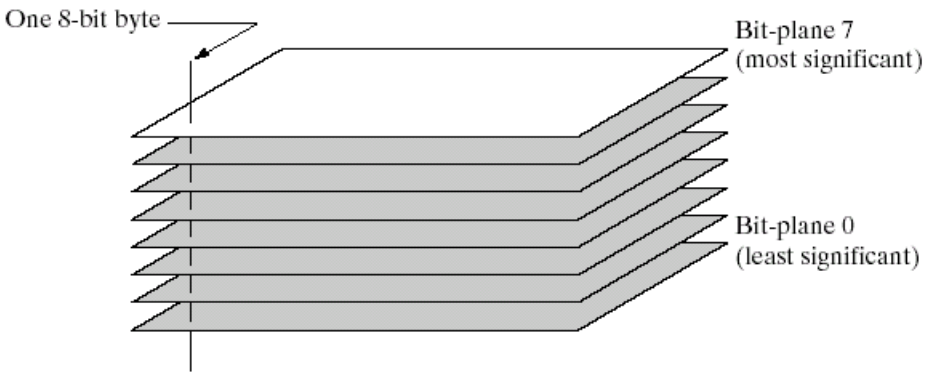



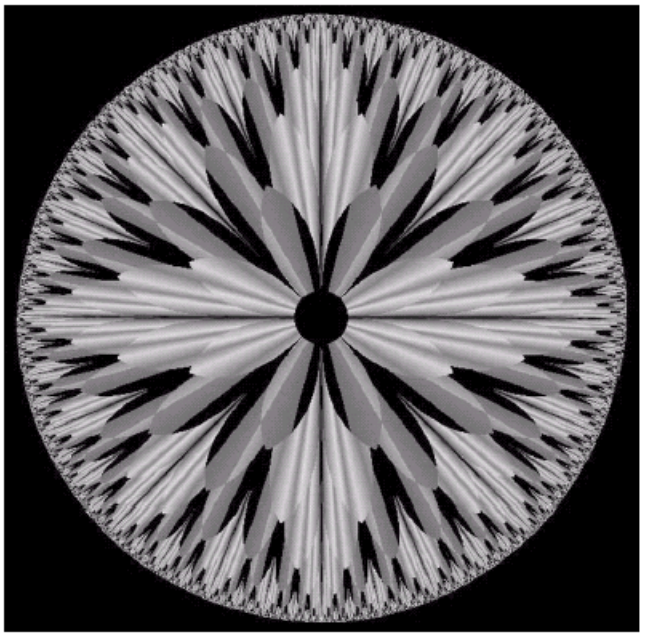

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing

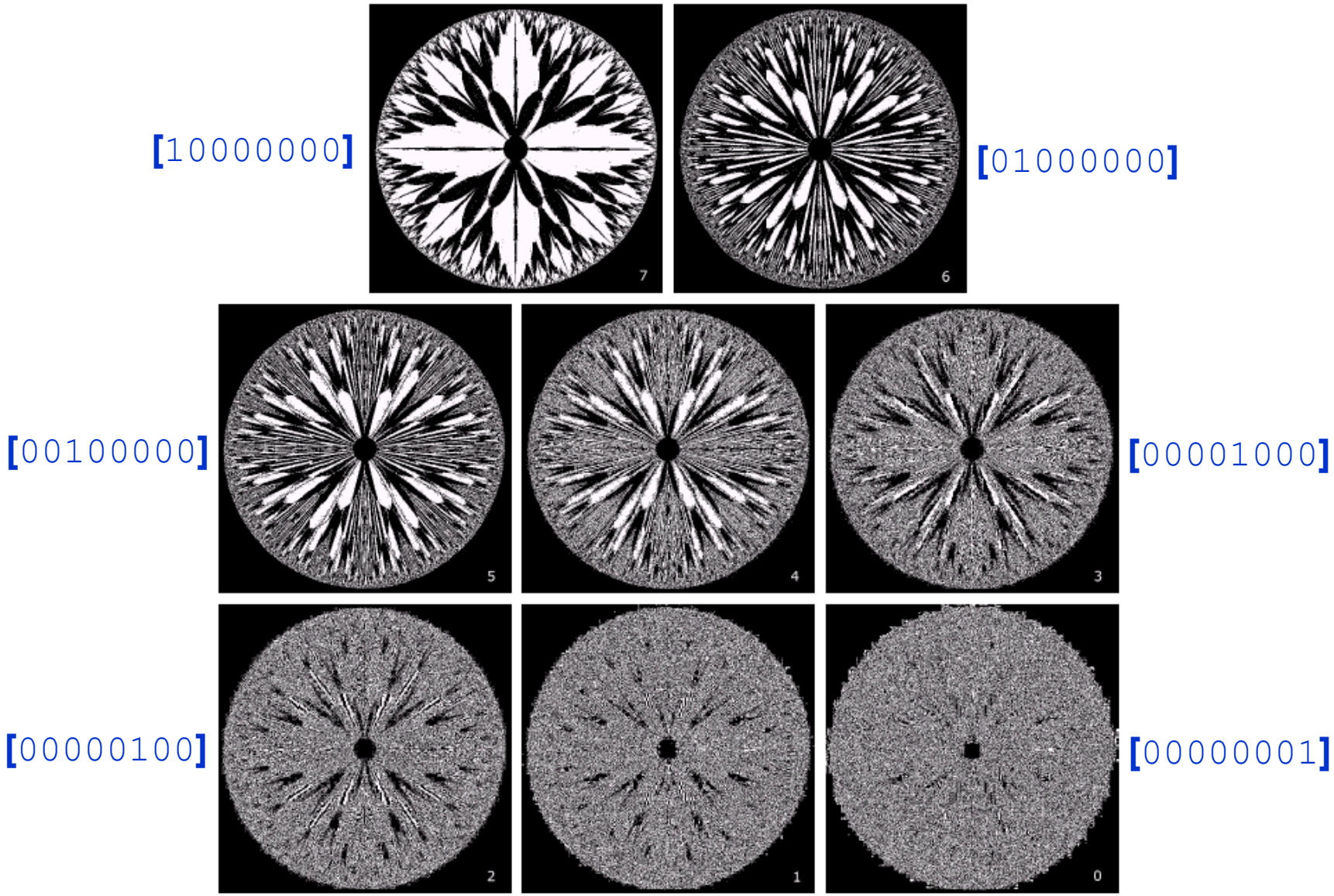
Often by isolating particular bits of the pixel values in an image we can highlight interesting aspects of that image

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



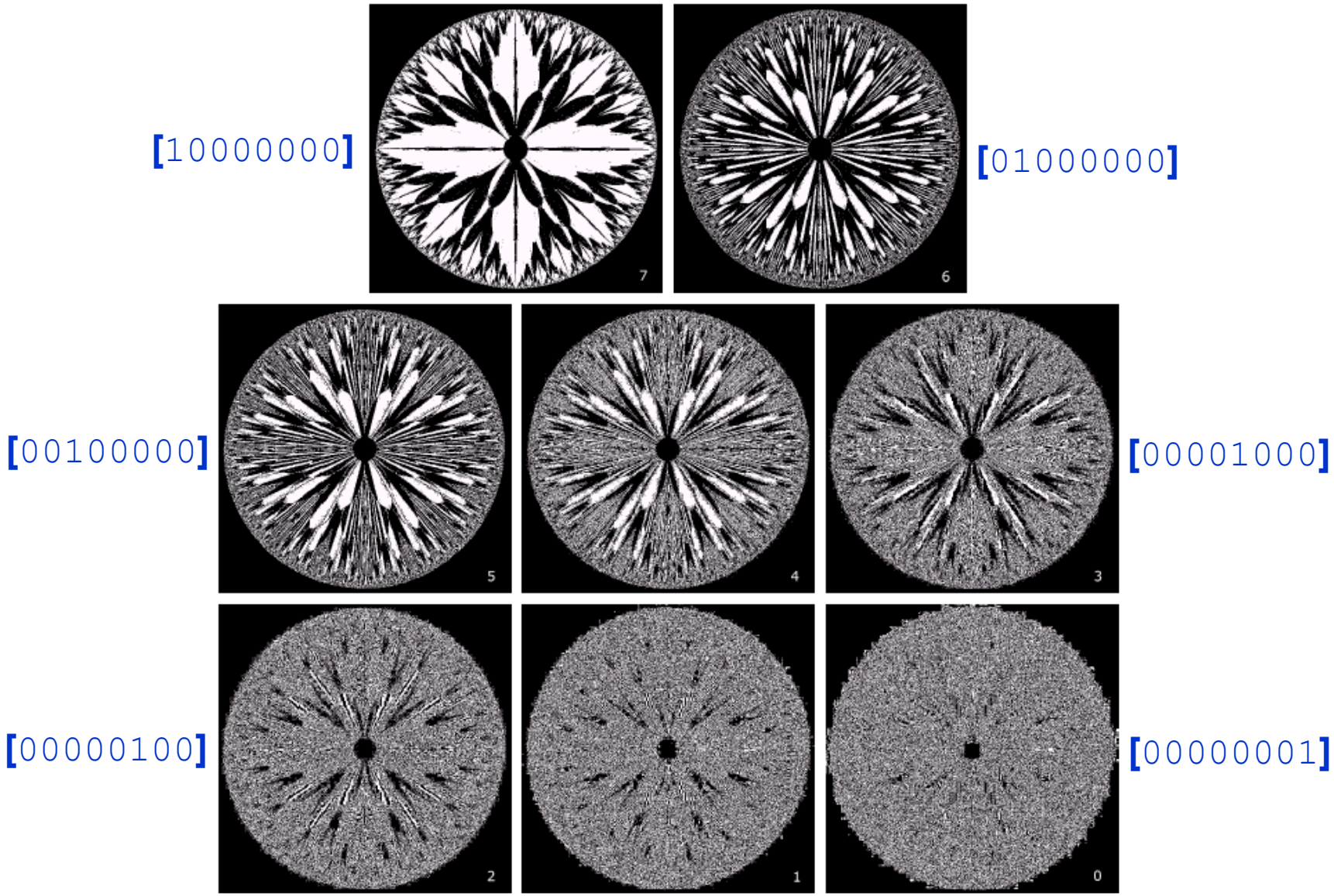



# Bit Plane Slicing (cont…)




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)

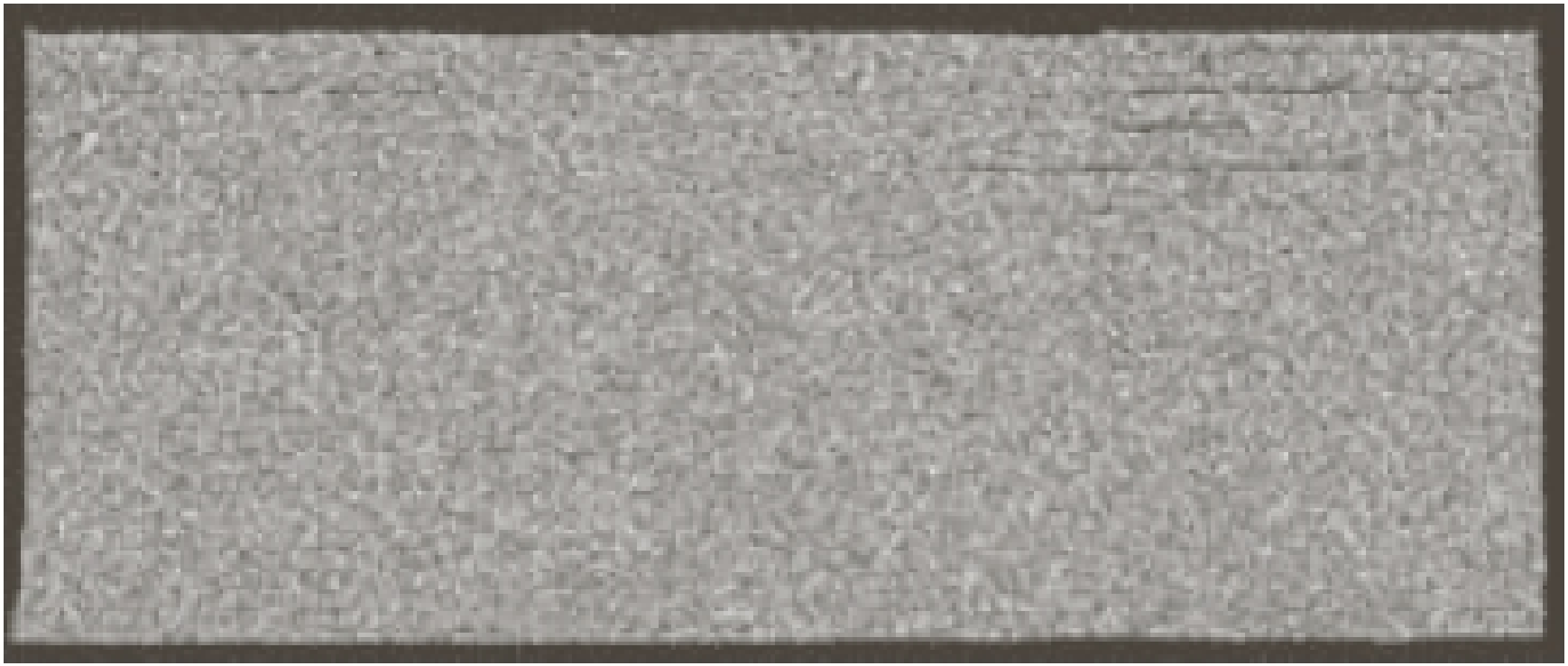
a b c
d e f
g h i

**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.

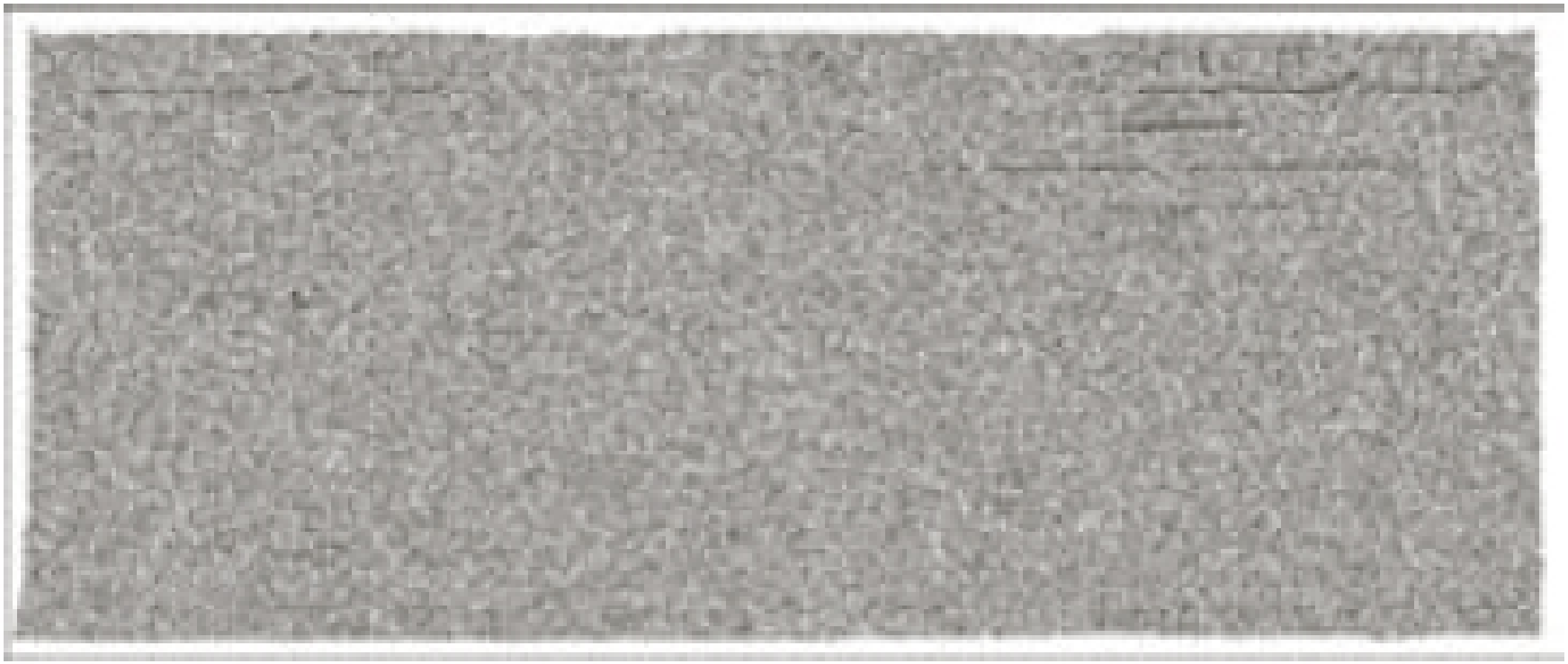
# Bit Plane Slicing (cont…)


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)

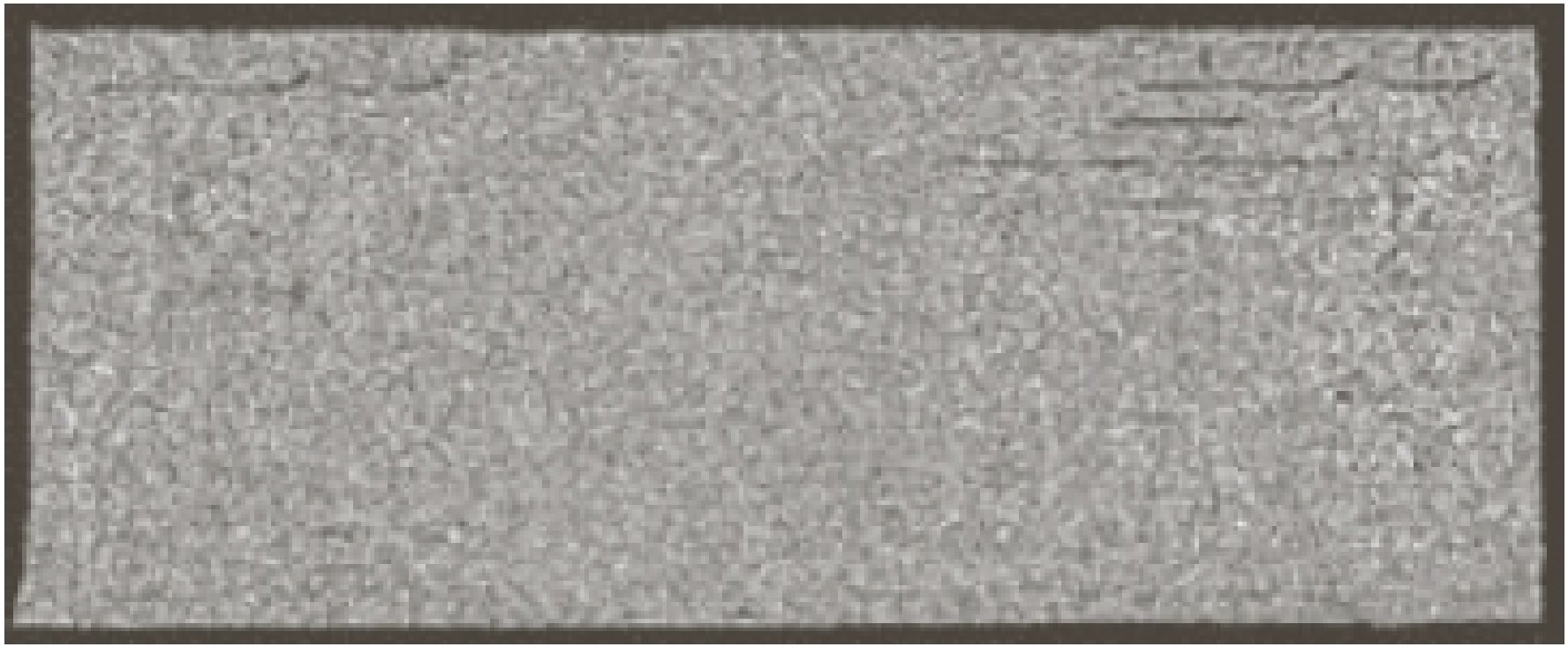


# Bit Plane Slicing (cont…)



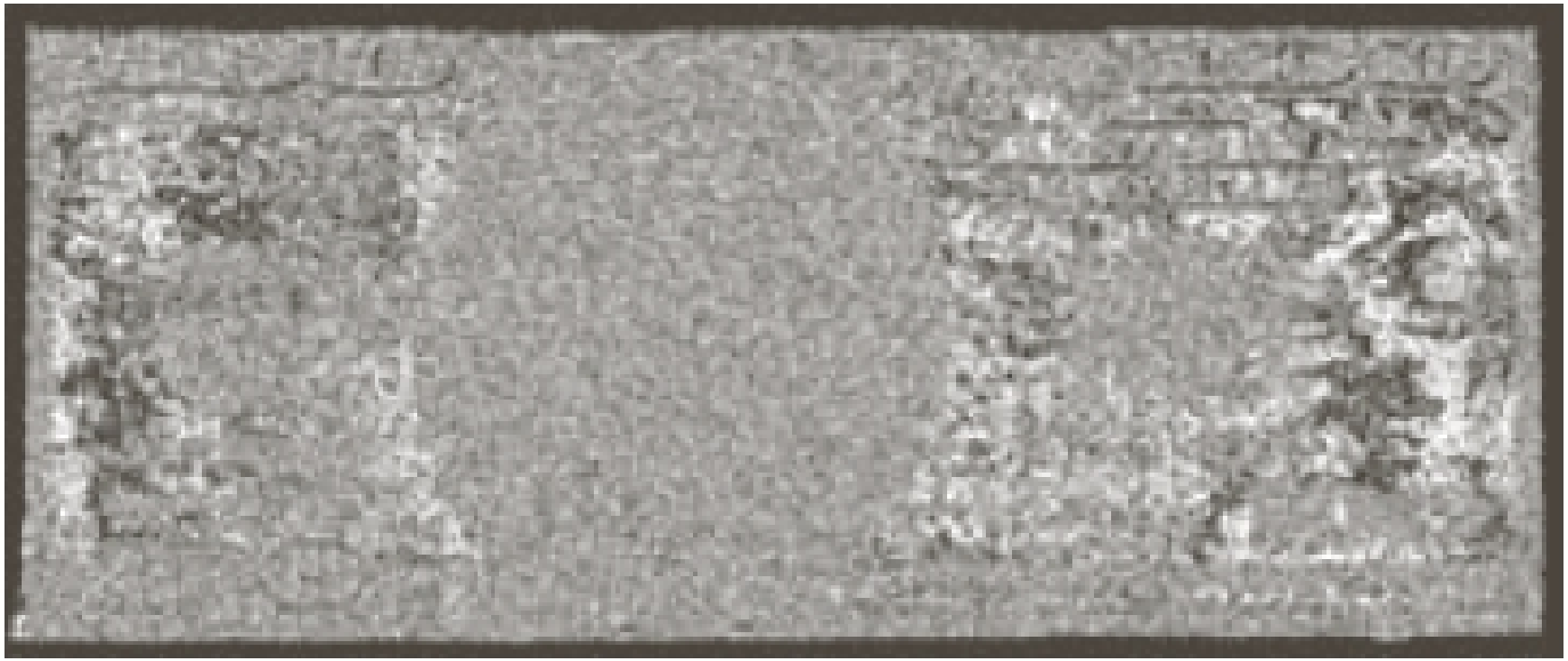
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)



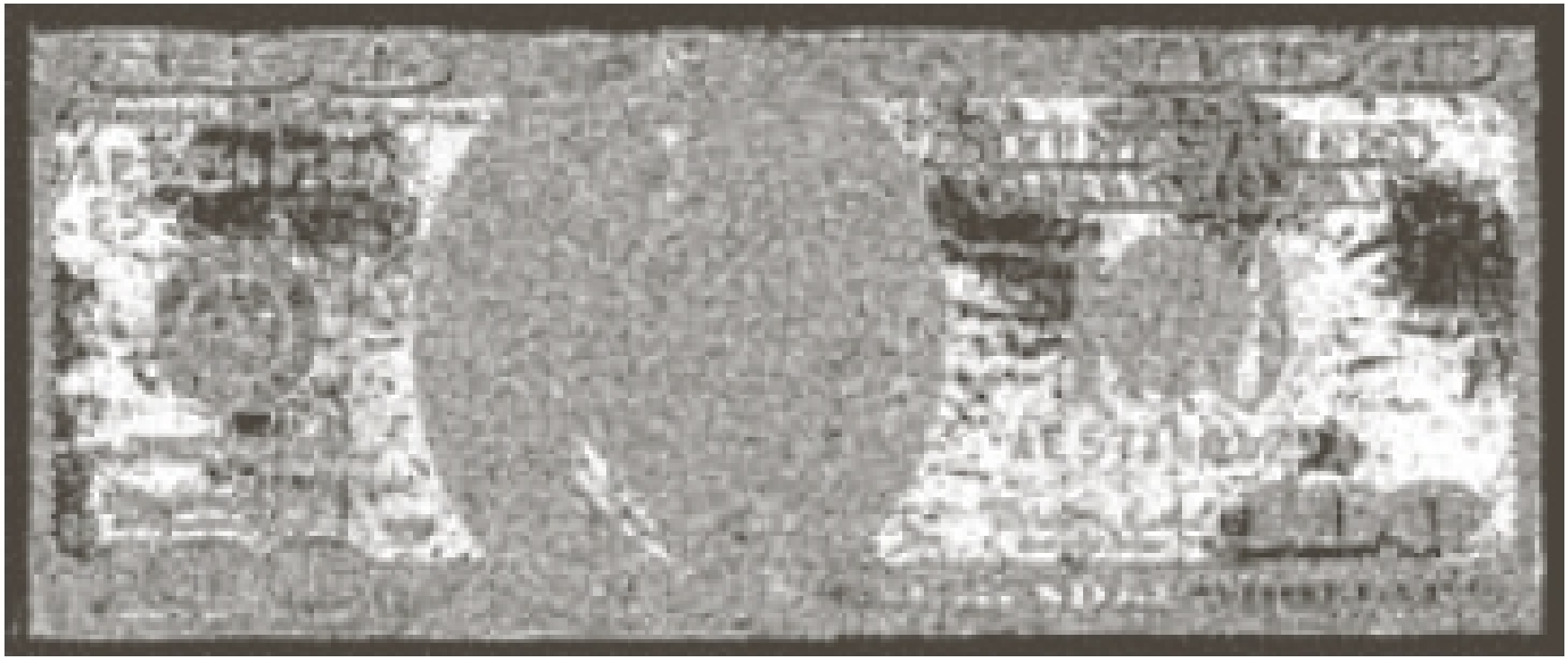
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Bit Plane Slicing (cont…)

Reconstructed image using only bit planes 8 and 7

Reconstructed image using only bit planes 8, 7 and 6

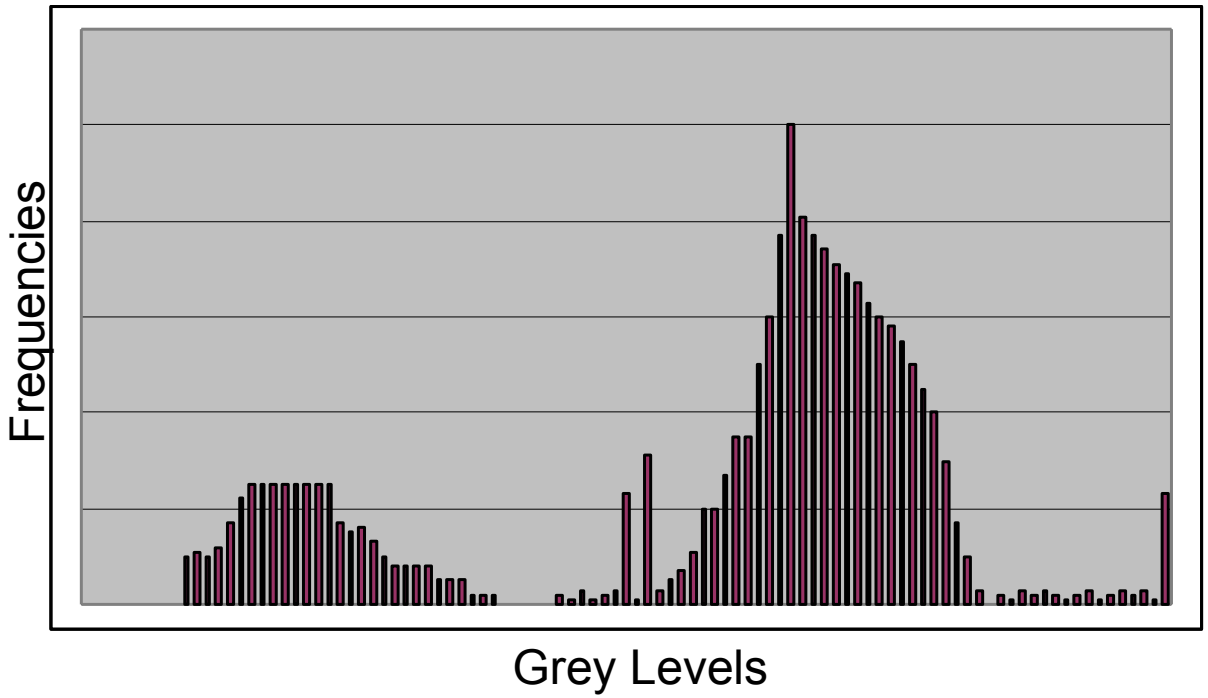
Reconstructed image using only bit planes 7, 6 and 5

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Histograms

The histogram of an image shows us the distribution of grey levels in the image

Massively useful in image processing, especially in segmentation



# Image Histograms

- The histogram of a digital image, f, (with intensities [0,L-1]) is a discrete function

$h(r_k) = n_k$

- Where $r_k$ is the kth intensity value and $n_k$ is the number of pixels in f with intensity $r_k$
- Normalizing the histogram is common practice – Divide the components by the total number of pixels in the image – Assuming an MxN image, this yields

$p(r_k) = n_k/MN$ for k=0,1,2,....,L-1 – $p(r_k)$ is, basically, an estimate of the probability of occurrence of intensity level $r_k$ in an image

$\Sigma\ p(r_k) = 1$

# Uses for Histogram Processing

- Image enhancements
- Image statistics
- Image compression
- Image segmentation
- Simple to calculate in software
- Economic hardware implementations
  - Popular tool in real-time image processing
- A plot of this function for all values of k provides a global description of the appearance of the image (gives useful information for contrast enhancement)
- Histograms commonly viewed in plots as

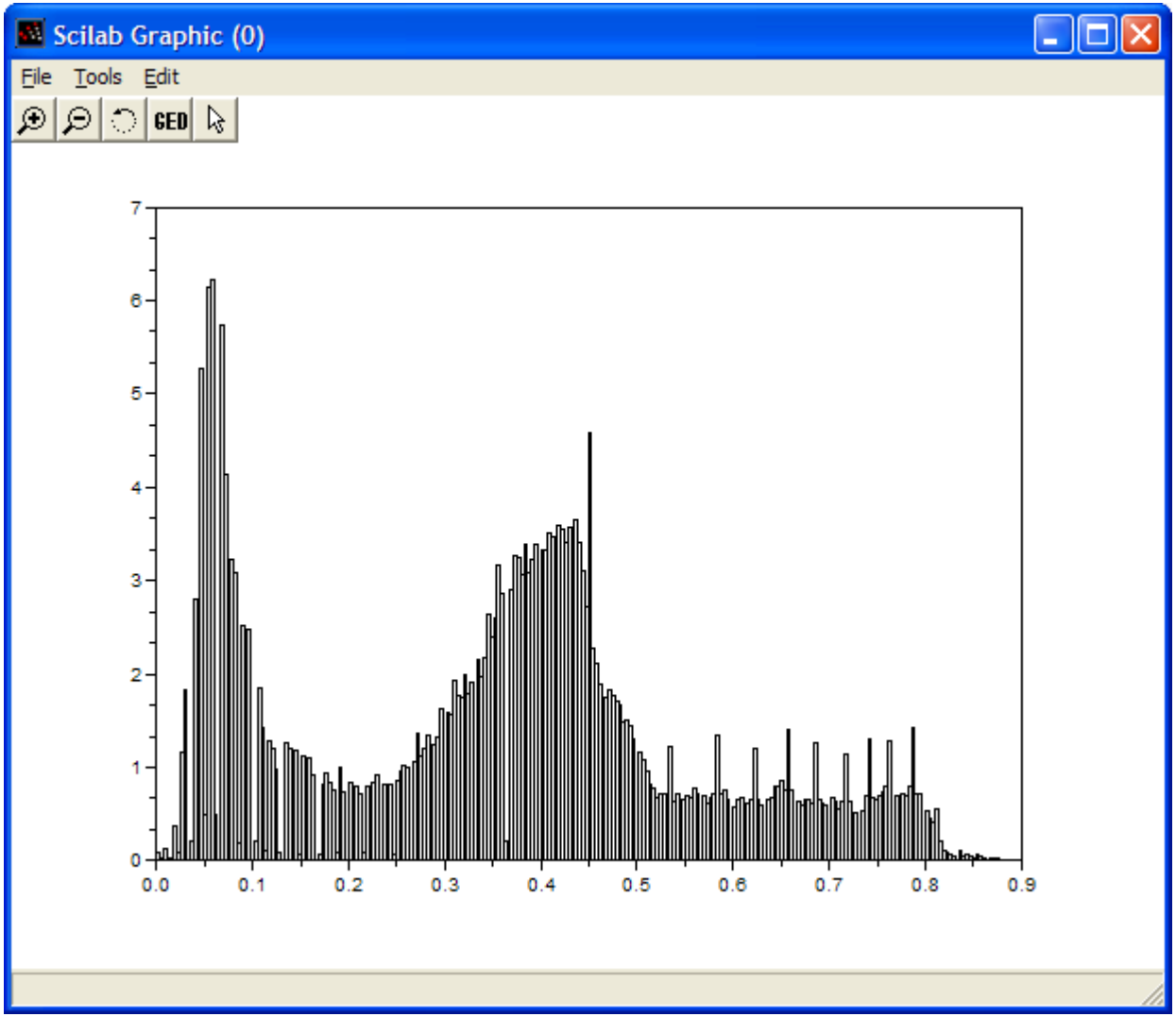
$h(r_k) = n_k$ versus $r_k$

$p(r_k) = n_k / MN$ versus $r_k$

# Histogram Examples

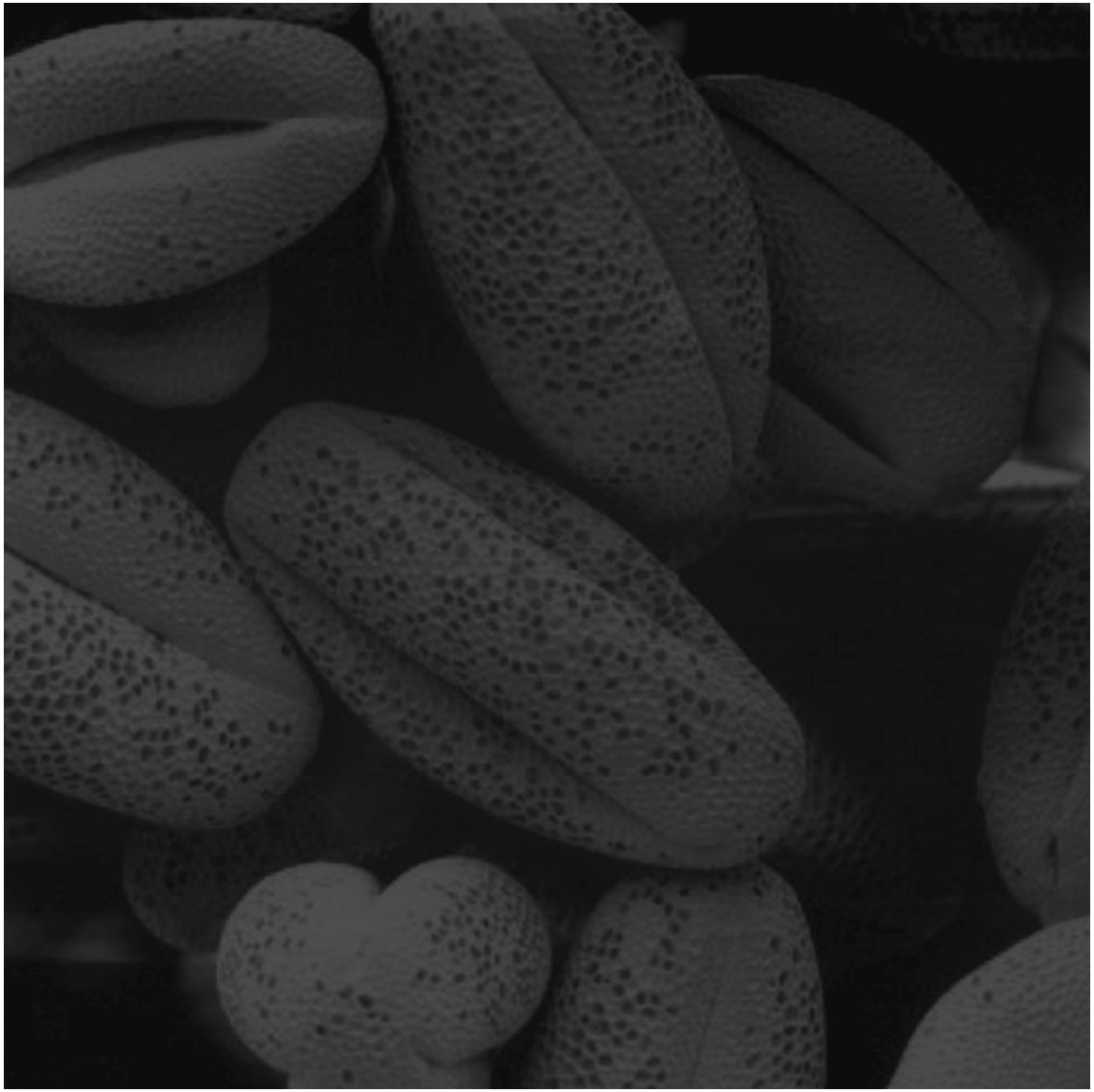
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Histogram Examples (cont…)

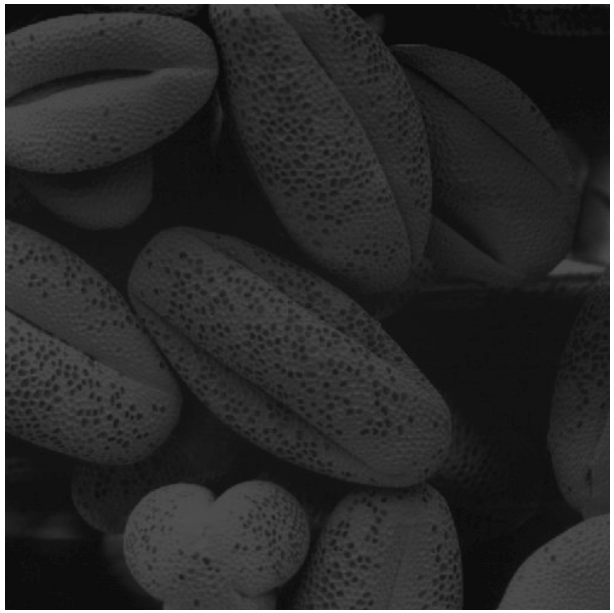


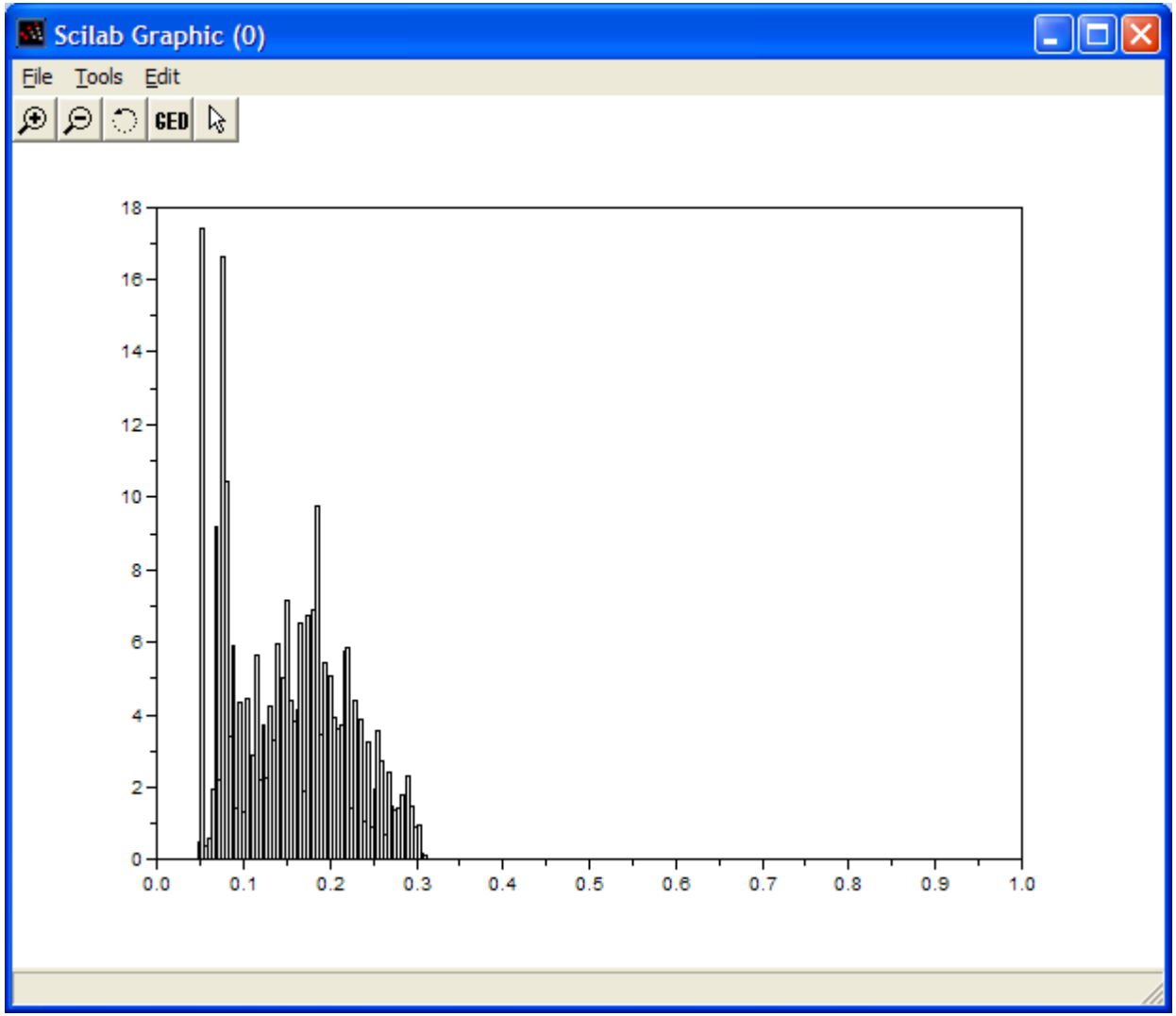

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Histogram Examples (cont…)



# Histogram Examples (cont…)

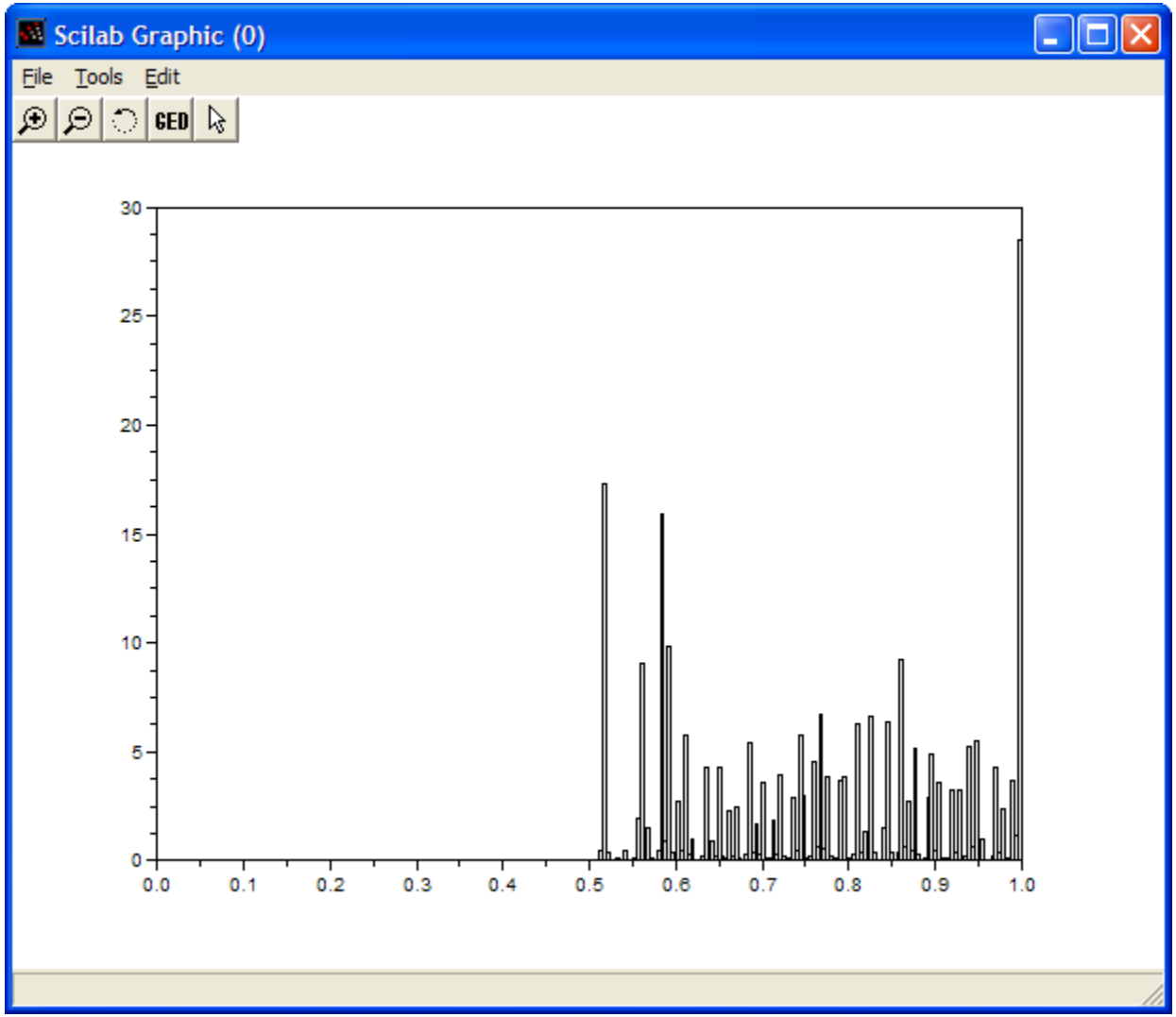





Images taken from Gonzalez & Woods, Digital Image Processing (2002)

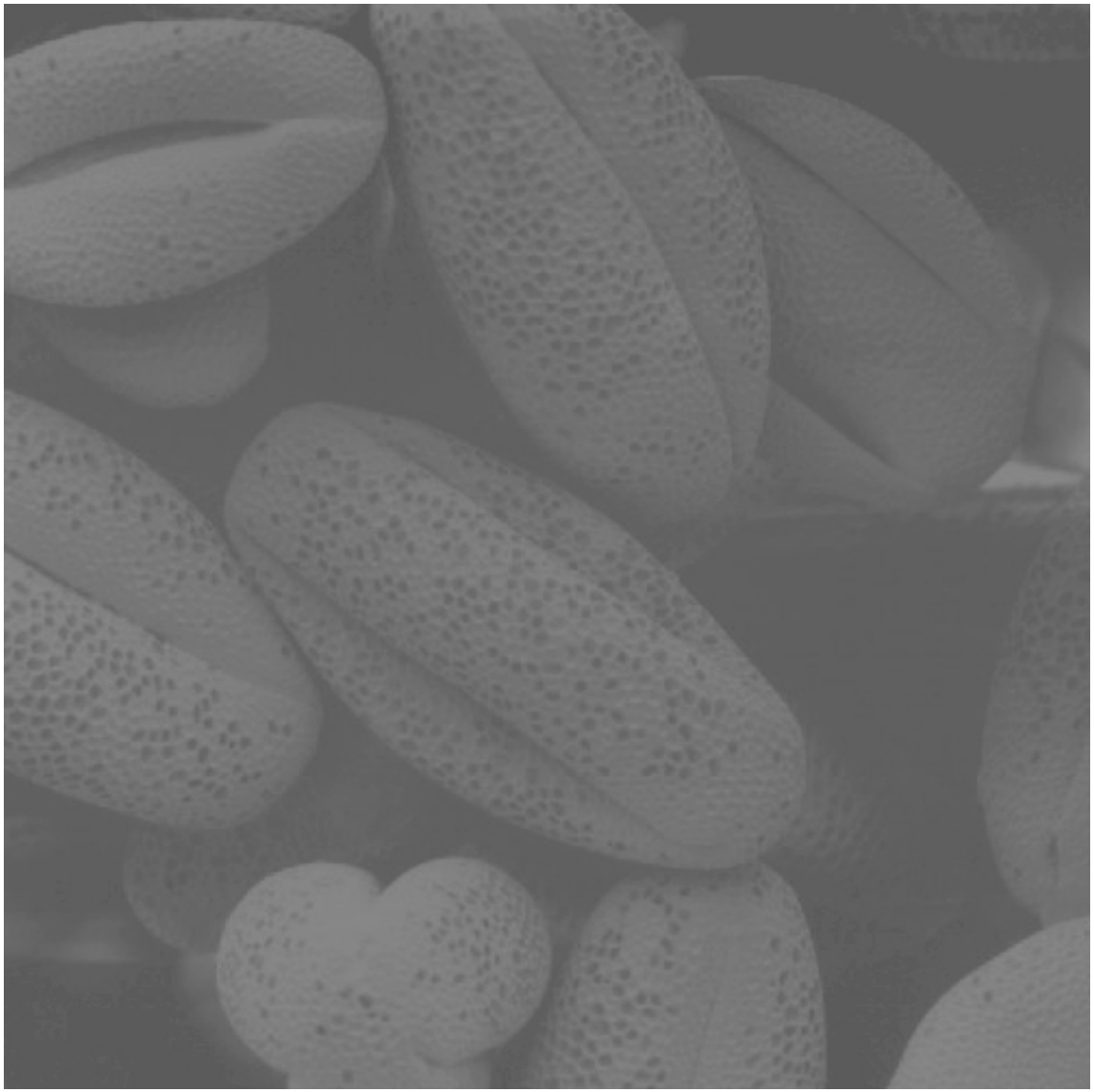
# Histogram Examples (cont…)

# Histogram Examples (cont…)



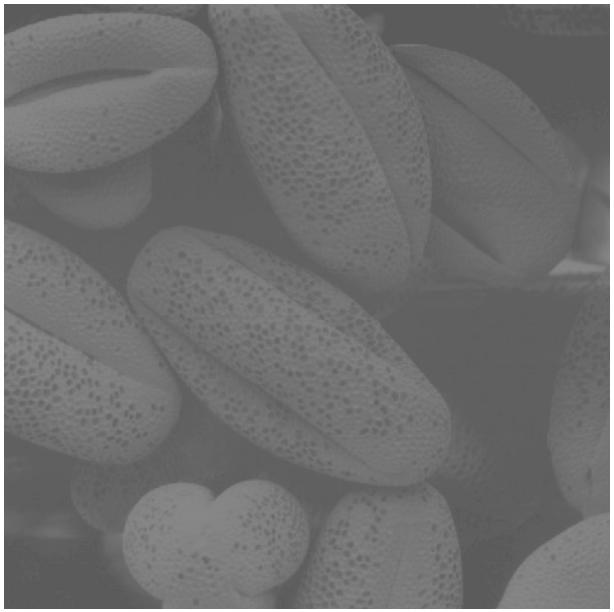

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

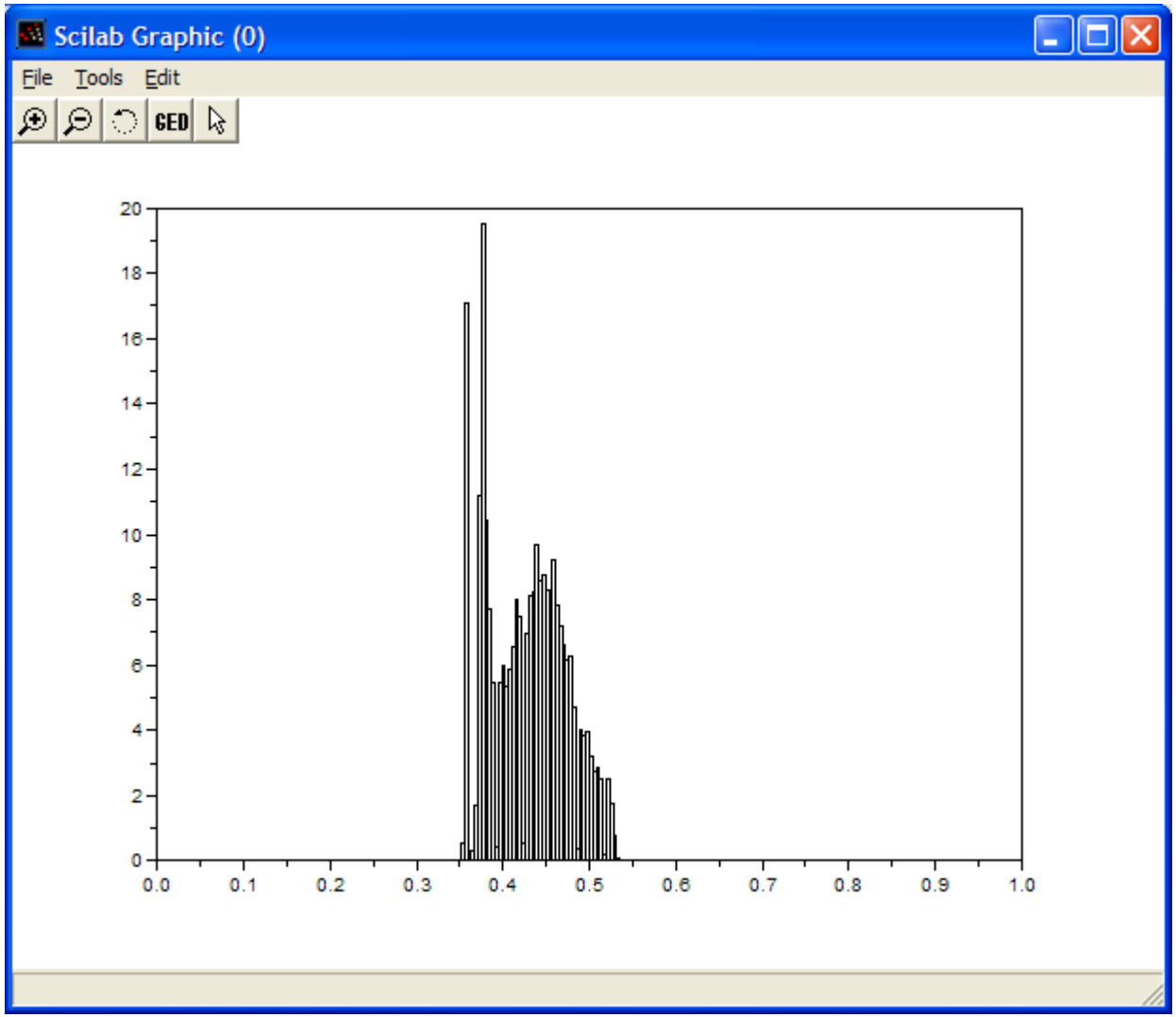
# Histogram Examples (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Histogram Examples (cont…)

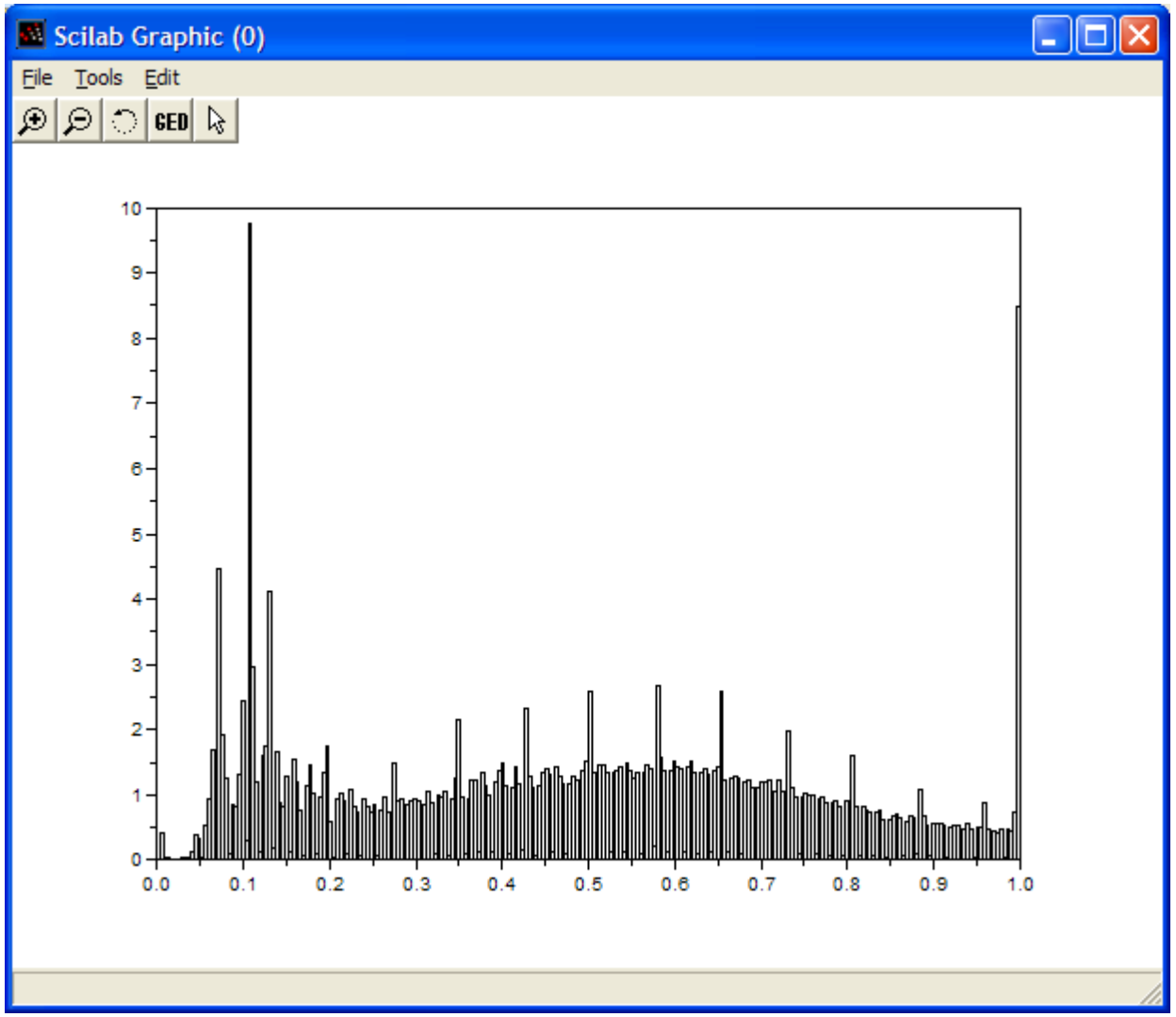





Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Histogram Examples (cont…)

# Histogram Examples (cont…)



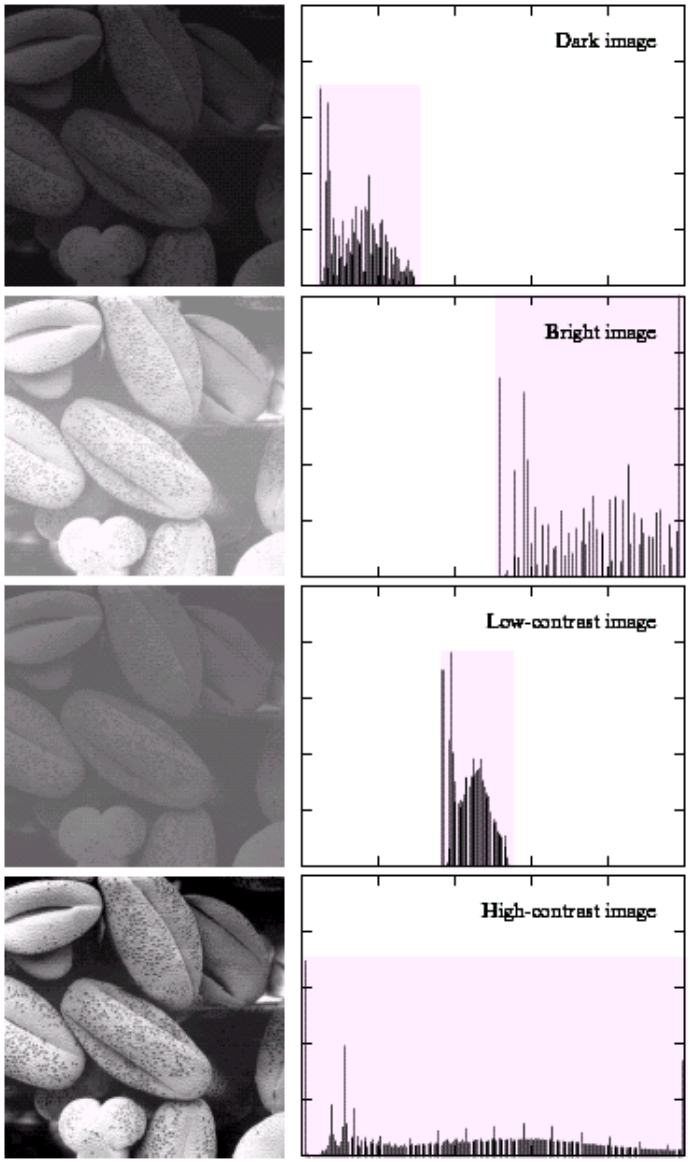

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Histogram Examples (cont…)

A selection of images and their histograms

Notice the relationships between the images and their histograms

Note that the high contrast image has the most evenly spaced histogram




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Contrast Stretching

We can fix images that have poor contrast by applying a pretty simple contrast specification

The interesting part is how do we decide on this transformation function?

# Histogram Equalisation

- Histogram equalization is a process for increasing the contrast in an image by spreading the histogram out to be approximately uniformly distributed
- The gray levels of an image that has been subjected to histogram equalization are spread out and always reach white

– The increase of dynamic range produces an increase in contrast

- For images with low contrast, histogram equalization has the adverse effect of increasing visual graininess

# Histogram Equalisation

- The intensity transformation function we are constructing is of the form

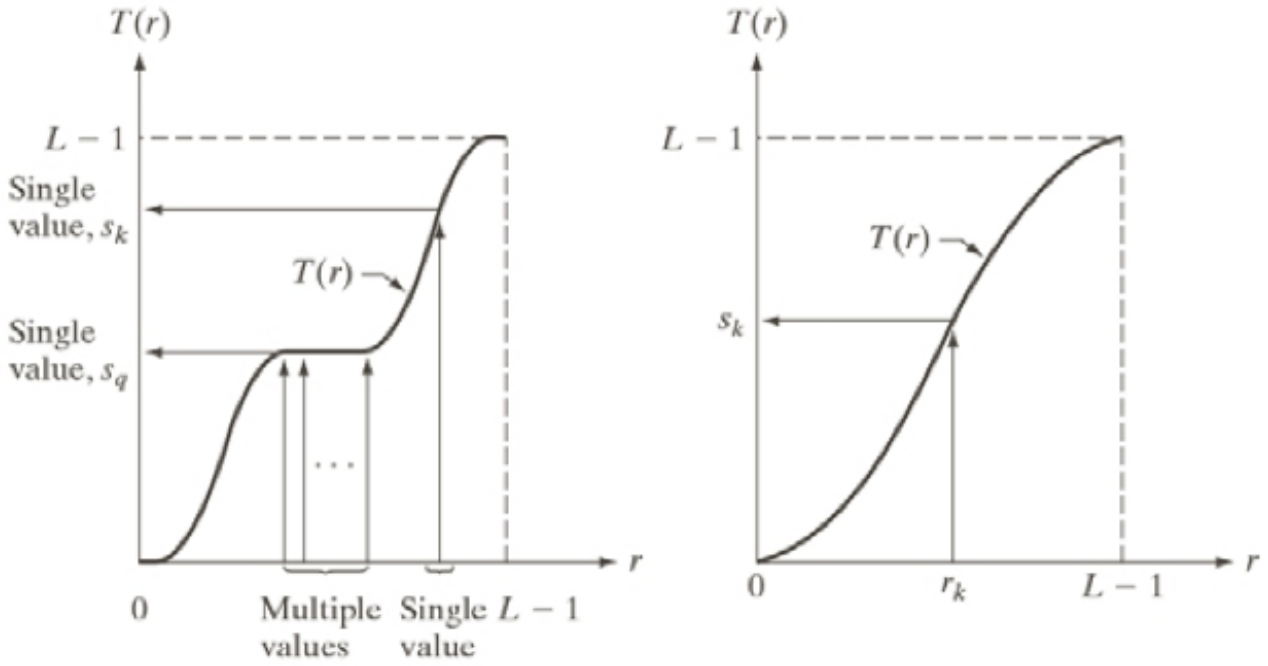
$$s=T(r) \qquad 0 \le r \le L-1$$

- An output intensity level $s$ is produced for every pixel in the input image having intensity $r$
- We assume
  - $T(r)$ is monotonically increasing in the interval $0 \le r \le L-1$
    - $0 \le T(r) \le L-1$ for $0 \le r \le L-1$
- If we define the inverse

$$r=T^{-1}(s) \qquad 0 \le s \le L-1$$

- Then $T(r)$ should be strictly monotonically increasing

# Histogram Equalisation




a b

**FIGURE 3.17** (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

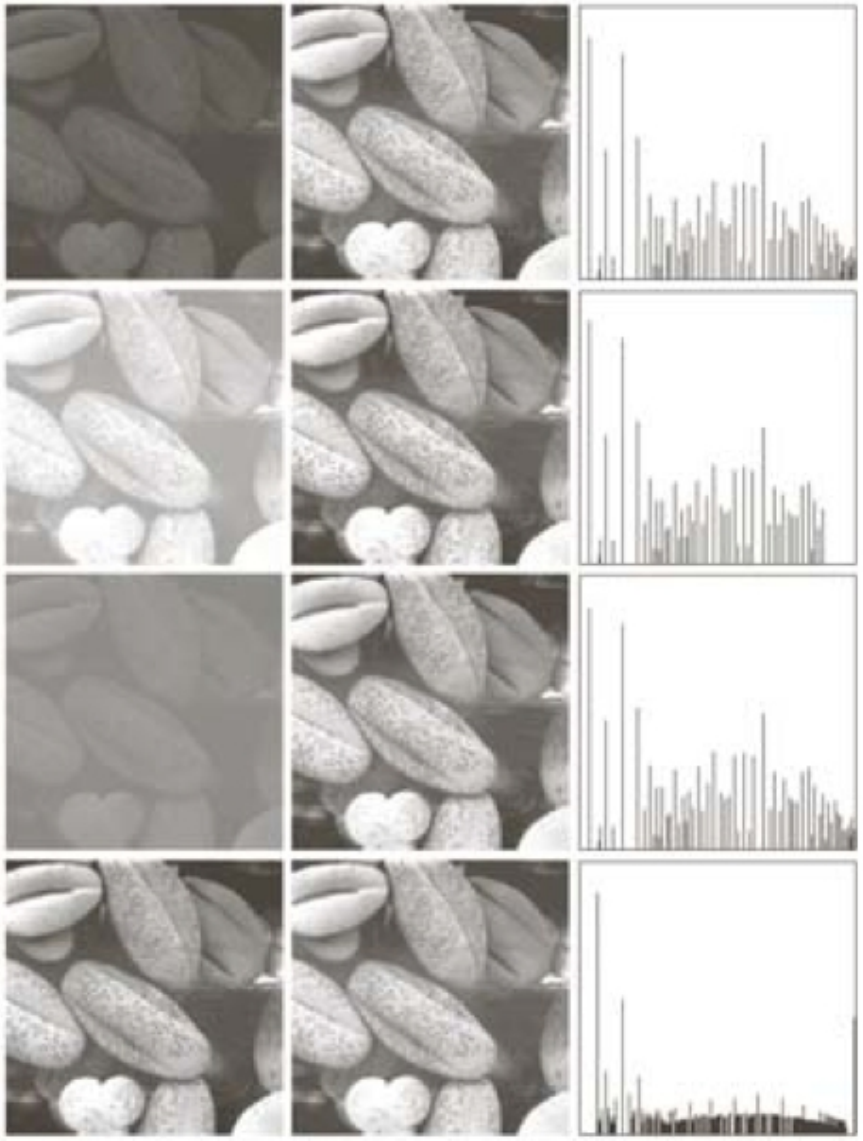
# Histogram Equalisation

- Histogram equalization requires construction of a transformation function $s_k$

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{M \times N} \qquad s_k = T(r_k) = \frac{(L-1)}{M \times N} \sum_{j=0}^{k} n_j$$

- where $r_k$ is the kth gray level, $n_k$ is the number of pixels with that gray level, MxN is the number of pixels in the image, and k=0,1,...,L-1
- This yields an s with as many elements as the original image's histogram (normally 256 for our test images)
- The values of s will be in the range [0,1]. For constructing a new image, s would be scaled to the range [1,256]

# Histogram Equalisation



# Histogram Equalisation

Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

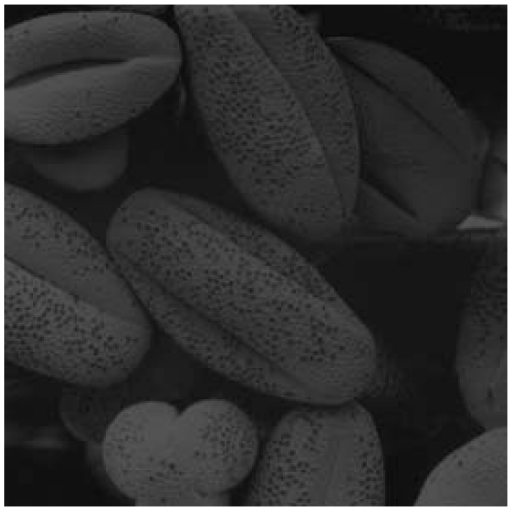
The formula for histogram equalisation is given where

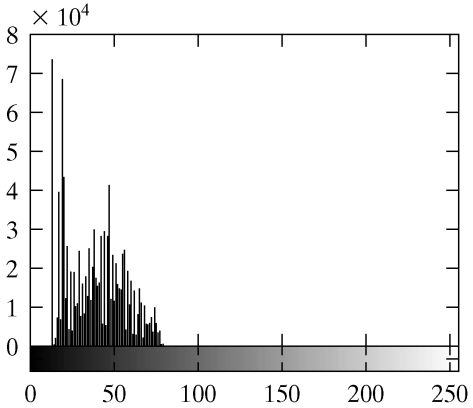
- $r_k$: input intensity
- $s_k$: processed intensity
- $k$: the intensity range (e.g 0.0 – 1.0)
- $n_j$: the frequency of intensity $j$
- $n$: the sum of all frequencies

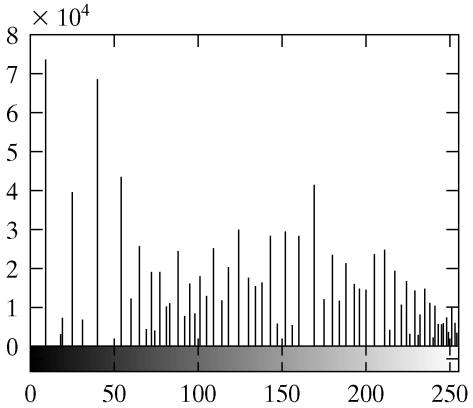
$$s_k = T(r_k)$$
$$= \sum_{j=1}^{k} p_r(r_j)$$
$$= \sum_{j=1}^{k} \frac{n_j}{n}$$

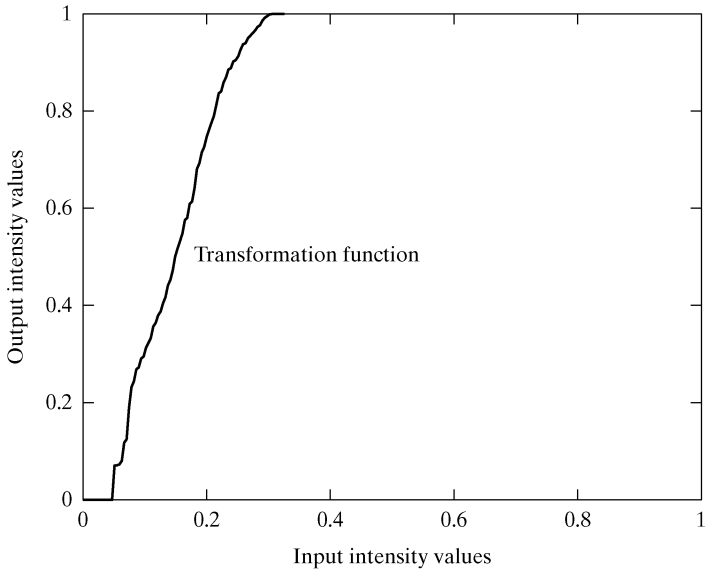
The transformation function in the next slide is Obtained by applying the RHS formula to the Original image in the next side.

# Equalisation Transformation Function



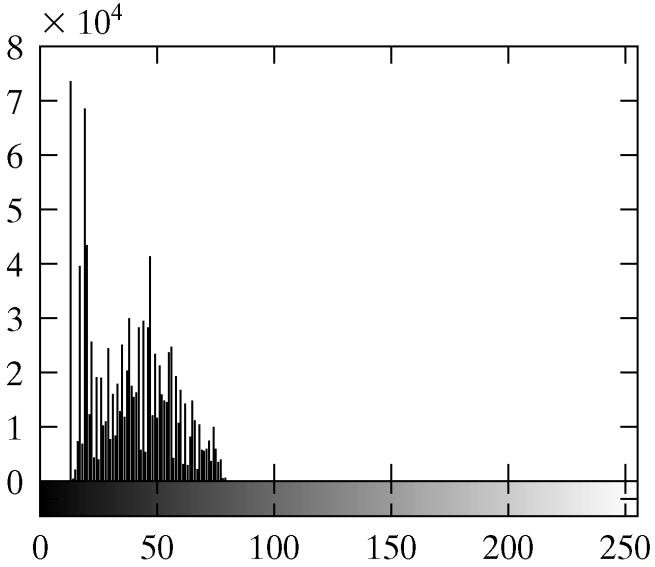


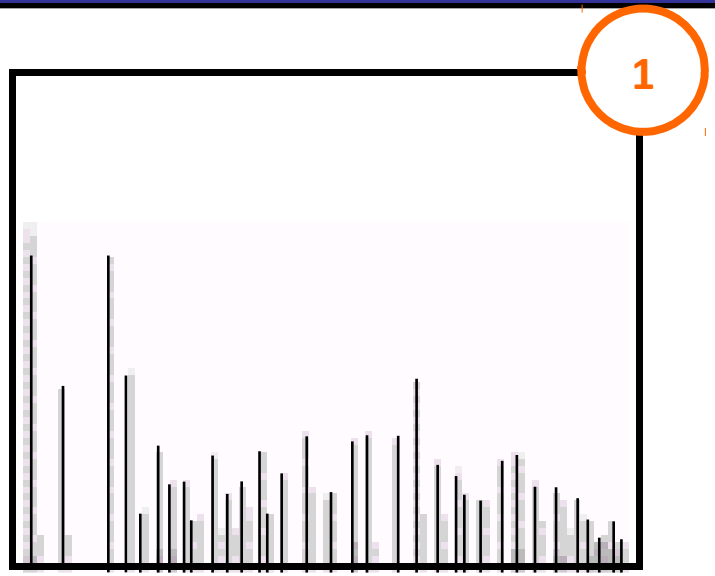



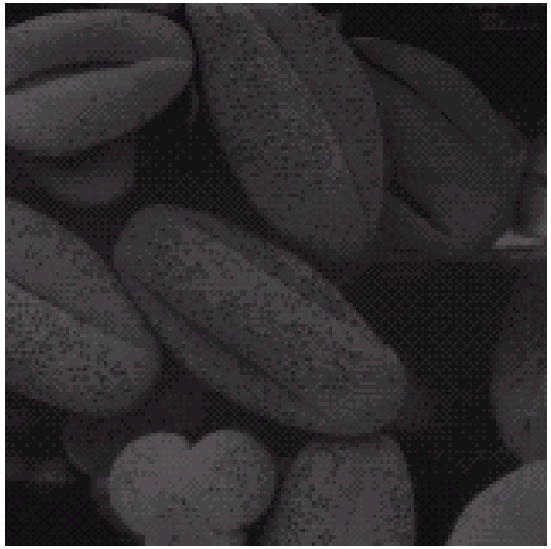




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Equalisation Examples



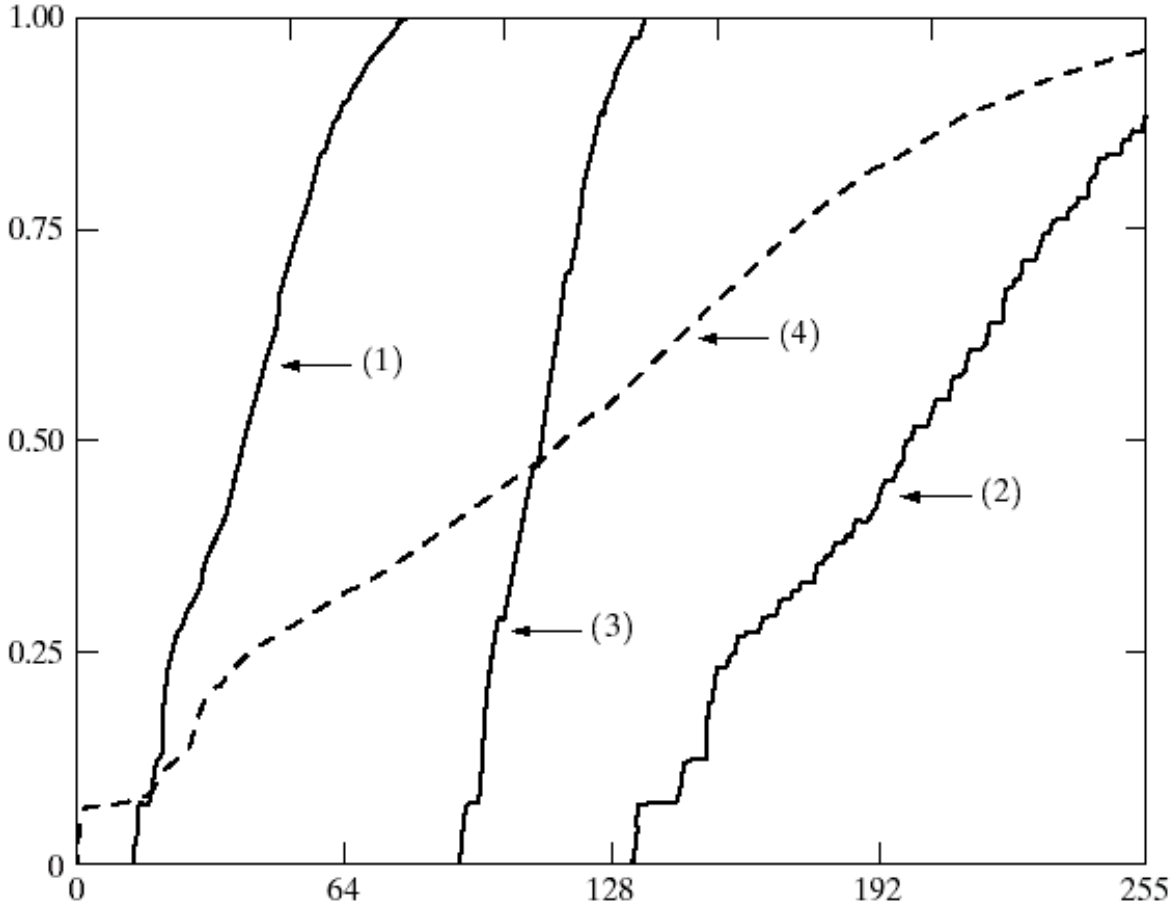



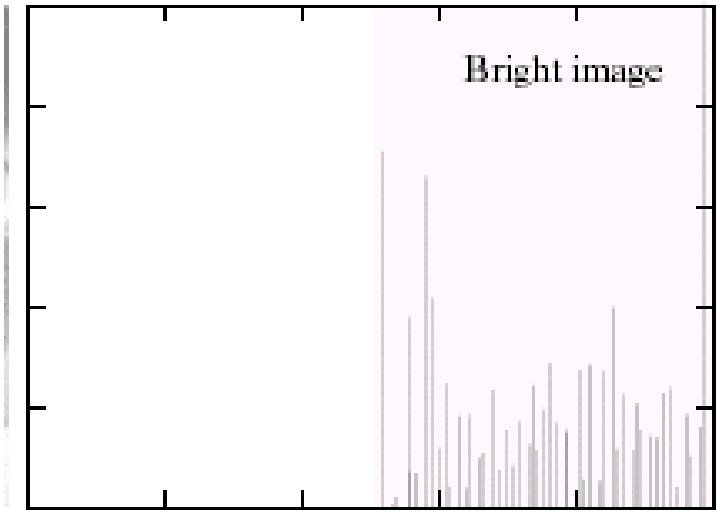
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

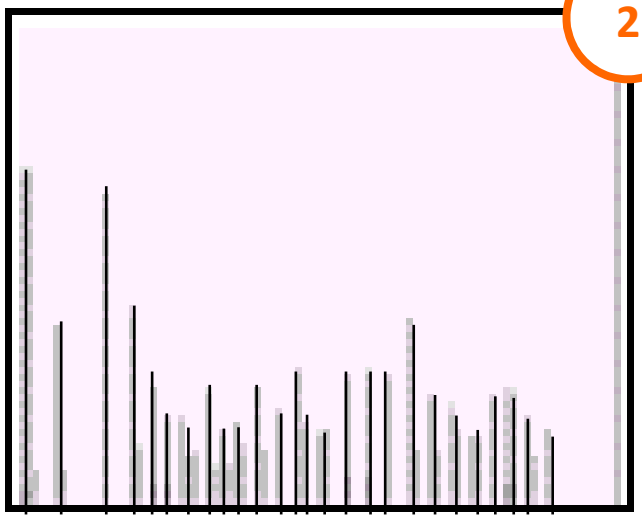
# Equalisation Transformation Functions

The functions used to equalise the images in the previous example



# Equalisation Examples

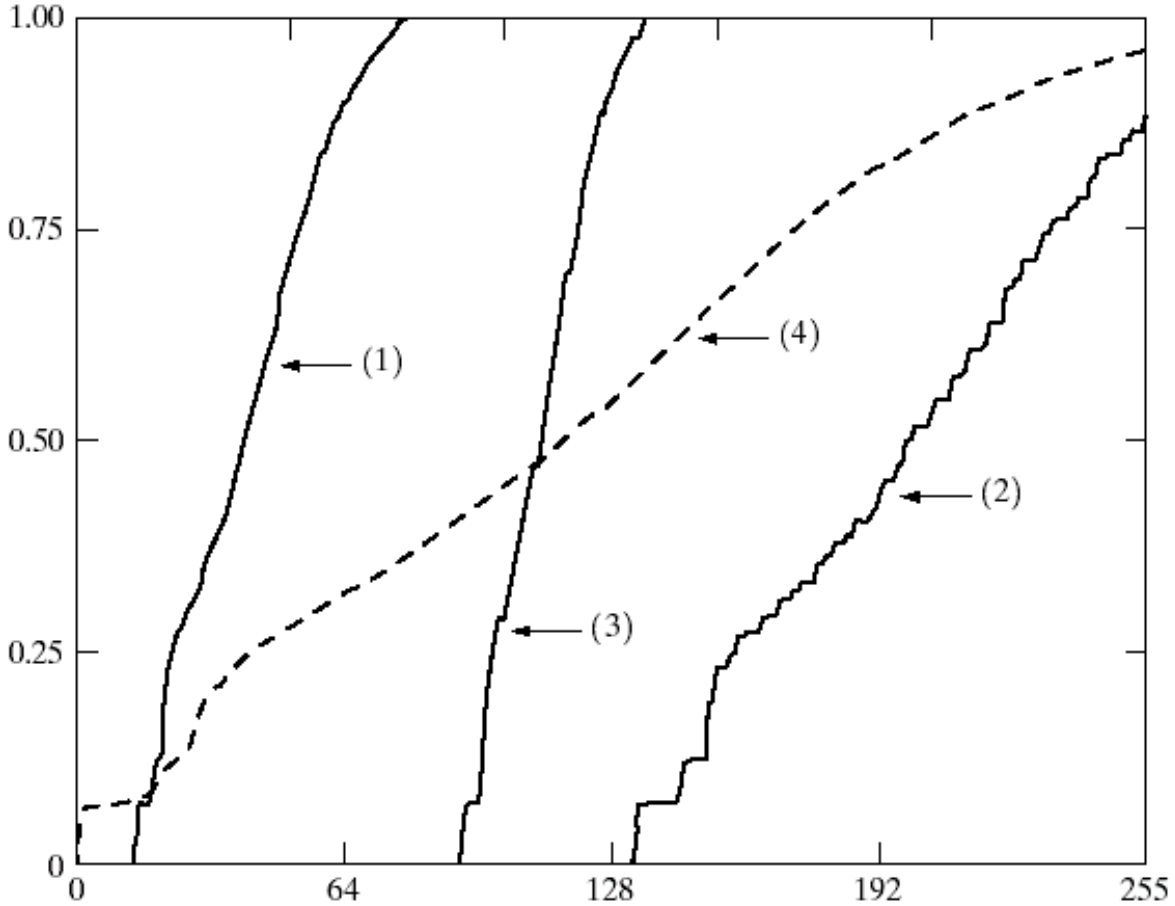


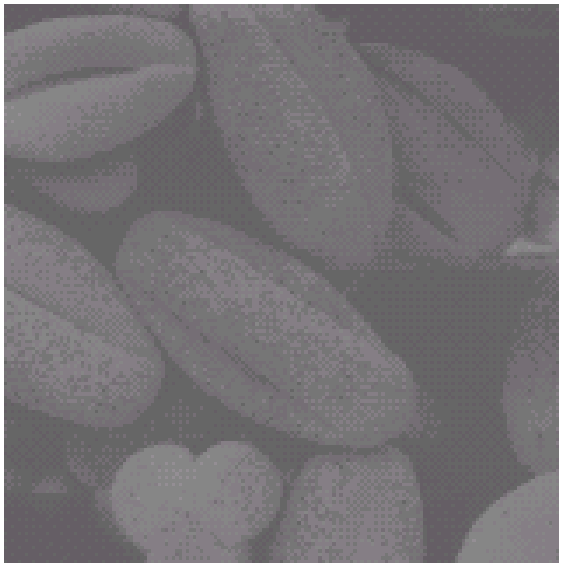


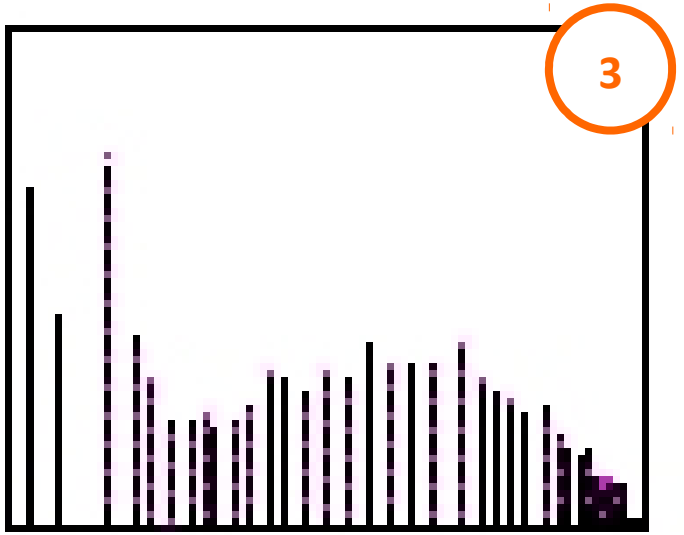
# Equalisation Transformation Functions

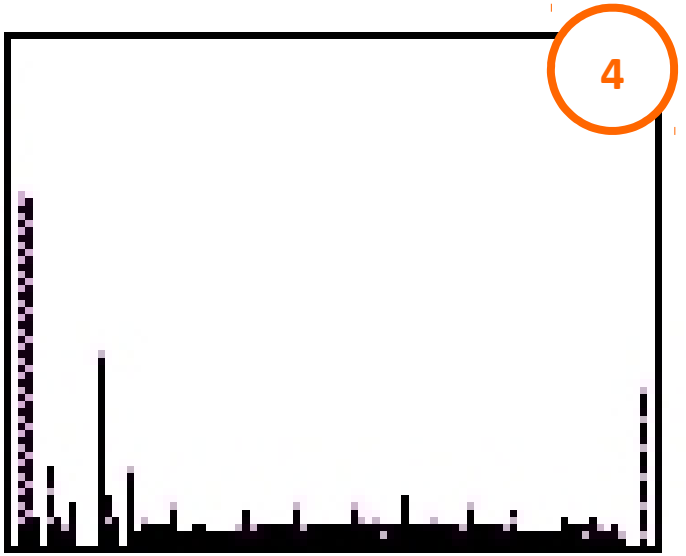
The functions used to equalise the images in the previous example



# Equalisation Examples (cont…)

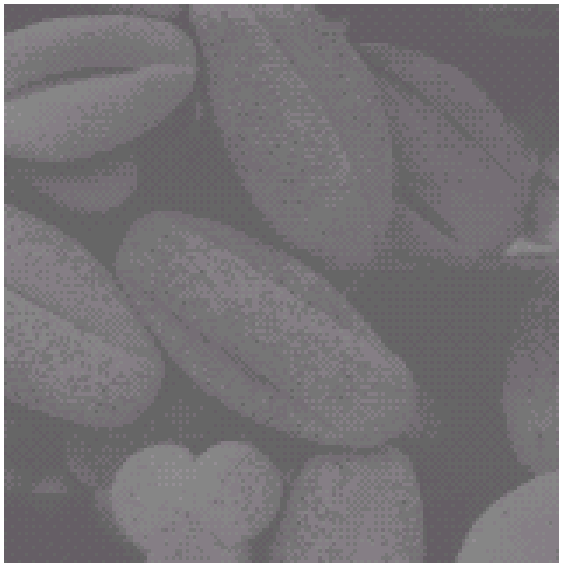


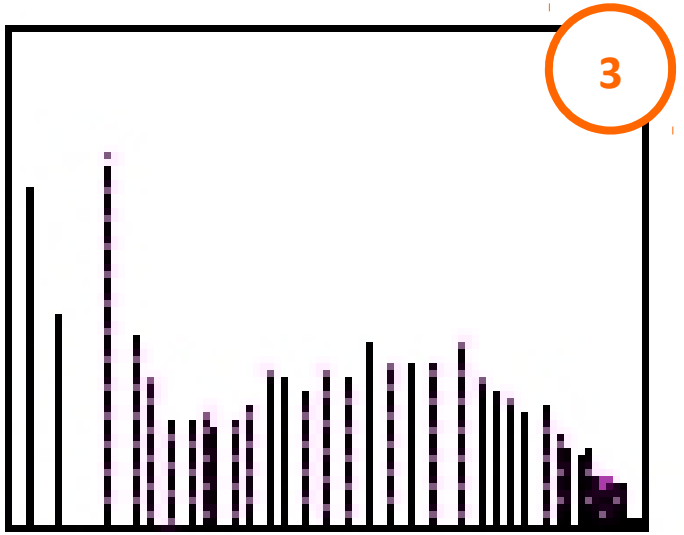


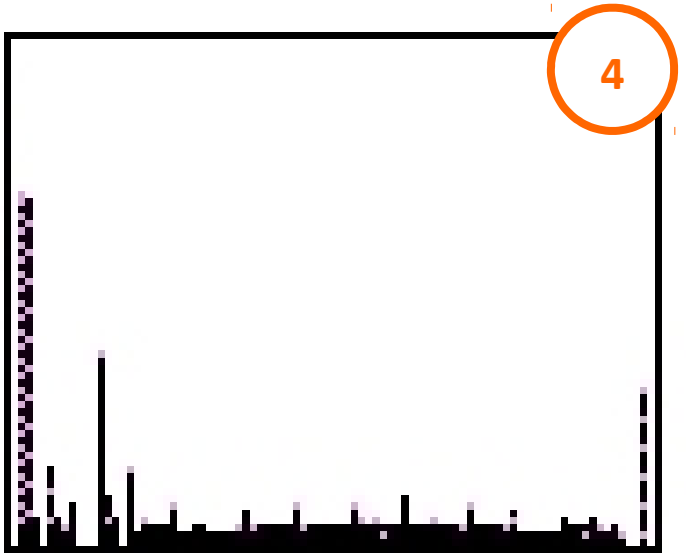




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Equalisation Examples (cont…)



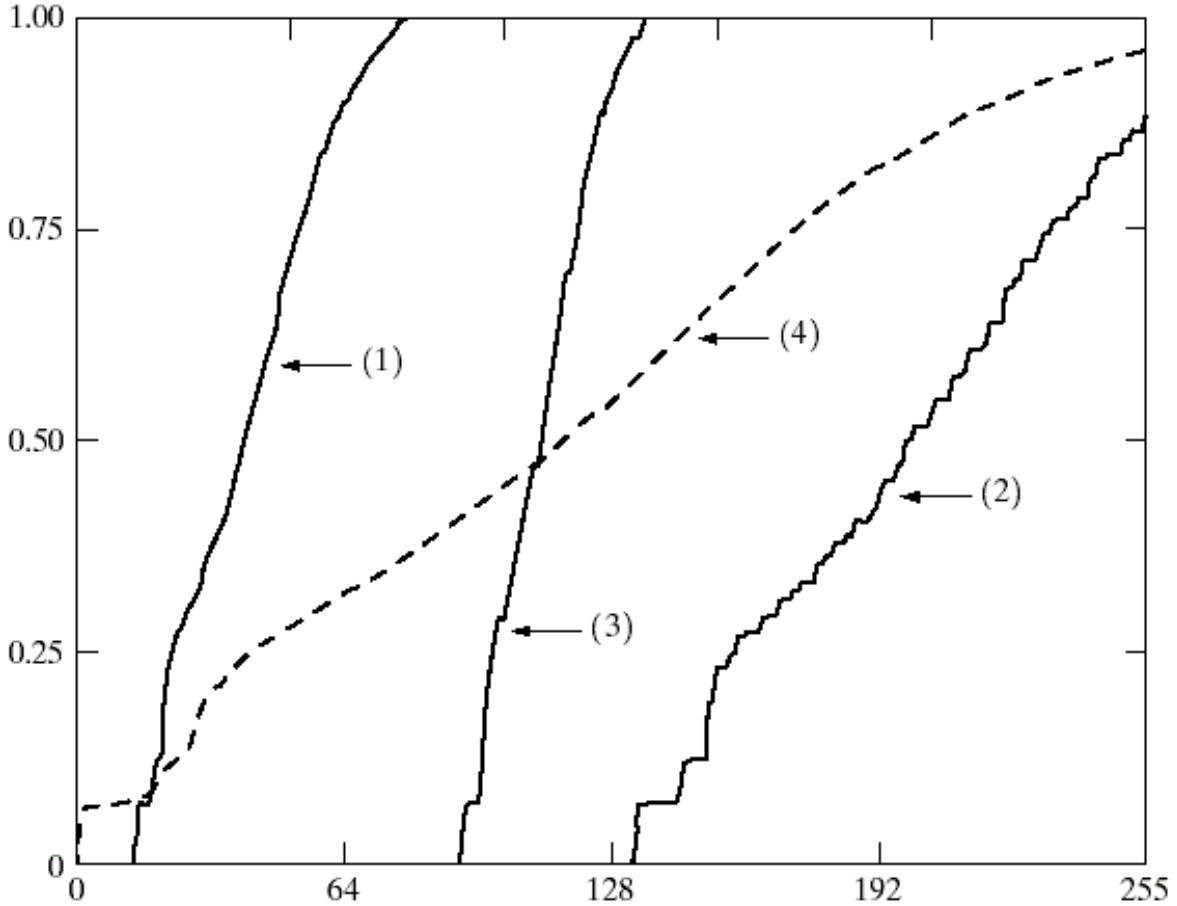



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Equalisation Transformation Functions

The functions used to equalise the images in the previous examples

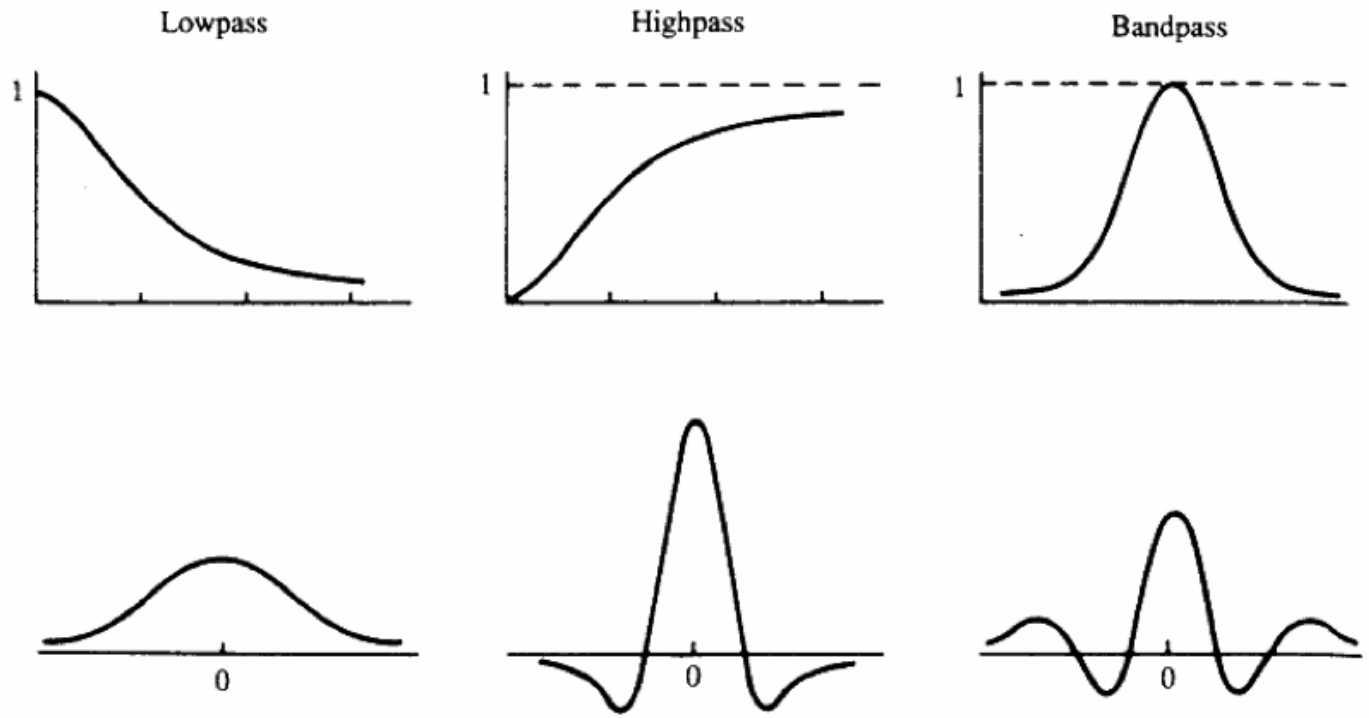


# Spatial Filtering

- Use of spatial masks for filtering is called spatial filtering
  - May be linear or nonlinear
- Linear filters
  - Lowpass: attenuate (or eliminate) high frequency components such as characterized by edges and sharp details in an image
    - Net effect is image blurring
  - Highpass: attenuate (or eliminate) low frequency components such as slowly varying characteristics
    - Net effect is a sharpening of edges and other details
  - Bandpass: attenuate (or eliminate) a given frequency range
    - Used primarily for image restoration(are of little interest for image enhancement)

# Spatial Filtering

- Filters in the frequency domain and corresponding spatial filters
- Basic approach is to sum products between mask coefficients and pixel values
  - $R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9$



# Order-Statistic nonlinear spatial filters

- Nonlinear spatial filters also operate on neighborhoods
- Their operation is based directly on pixel values in the neighborhood under consideration
  - They do not explicitly use coefficient values as in the linear spatial filters
- Example nonlinear spatial filters
  - Median filter: Computes the median gray-level value of the neighborhood. Used for noise reduction.
  - Max filter: Used to find the brightest points in an image

$$R=\max\{z_k \mid k=1,2,...,9\}$$

  - Min filter: Used to find the dimmest points in an image

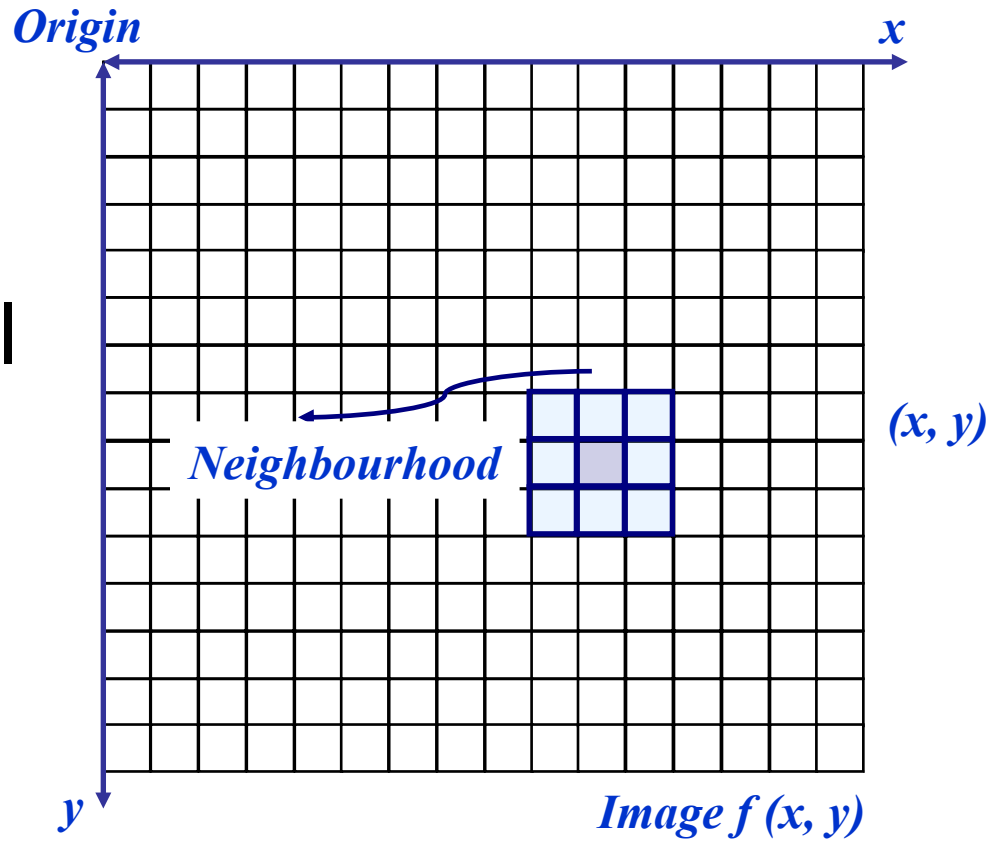
$$R=\min\{z_k \mid k=1,2,...,9\}$$

# Neighbourhood Operations

Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations

Neighbourhoods are mostly a rectangle around a central pixel

Any size rectangle and any shape filter are possible



# Simple Neighbourhood Operations

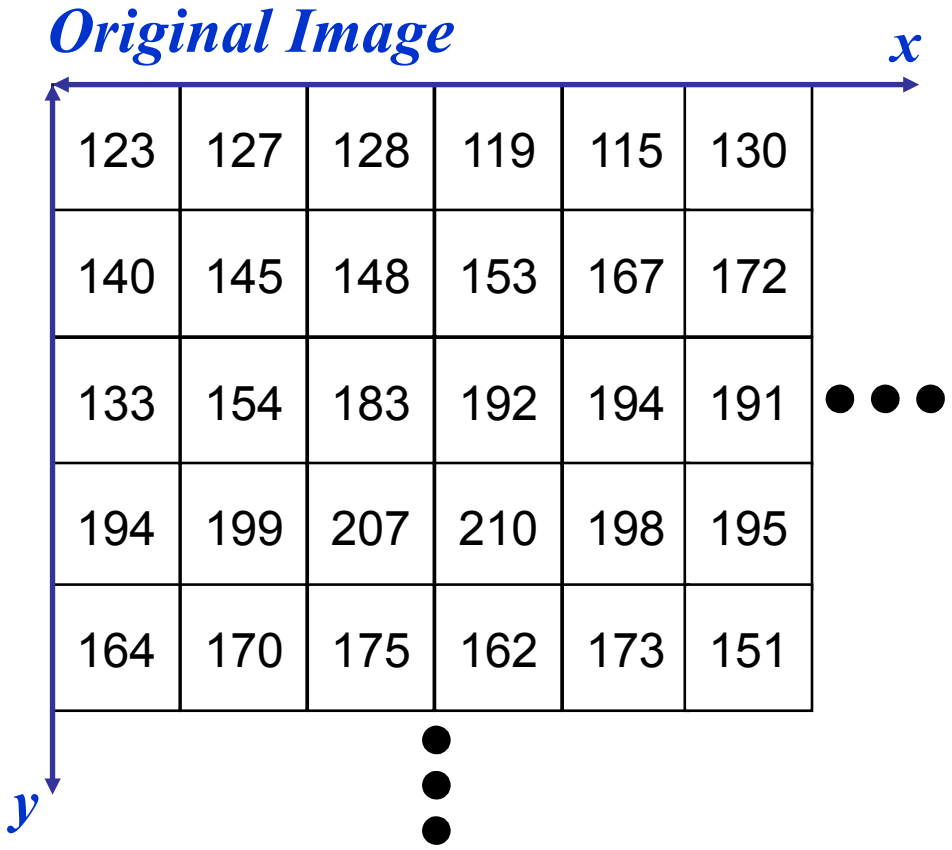
Some simple neighbourhood operations include:

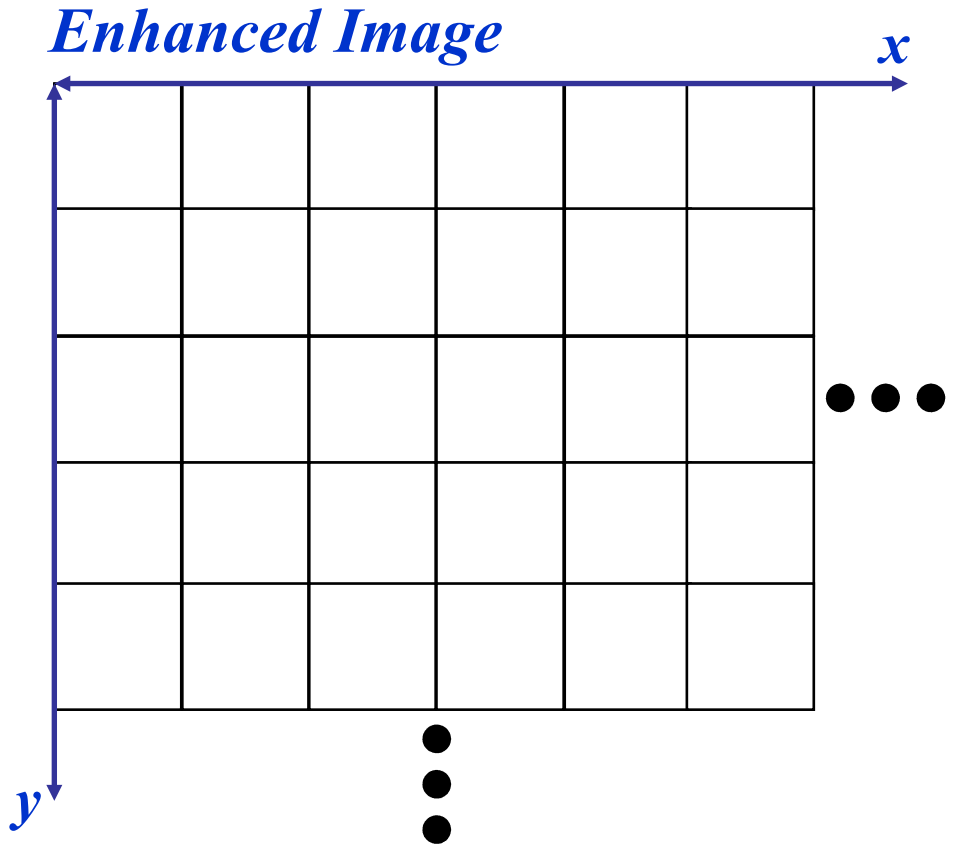
**Min:** Set the pixel value to the minimum in the neighbourhood

**Max:** Set the pixel value to the maximum in the neighbourhood

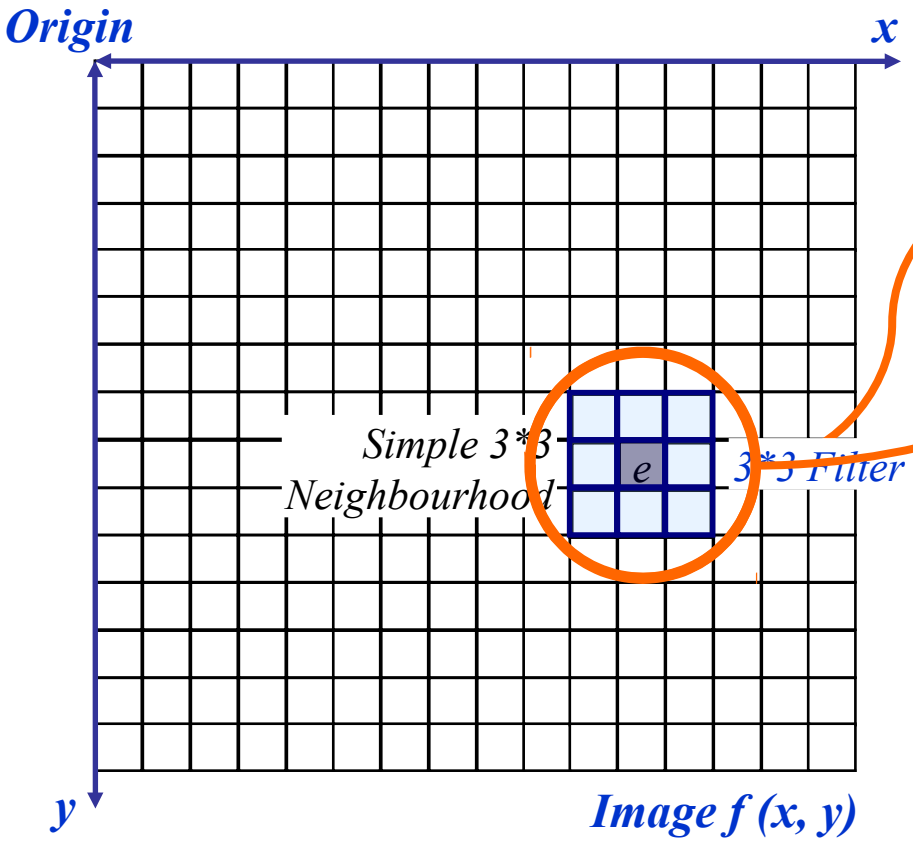
**Median:** The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median). Sometimes the median works better than the average

# Simple Neighbourhood Operations Example



# The Spatial Filtering Process




| | | |
|---|---|---|
| *a* | *b* | *c* |
| *d* | *e* | *f* |
| *g* | *h* | *i* |

**Original Image Pixels**

*

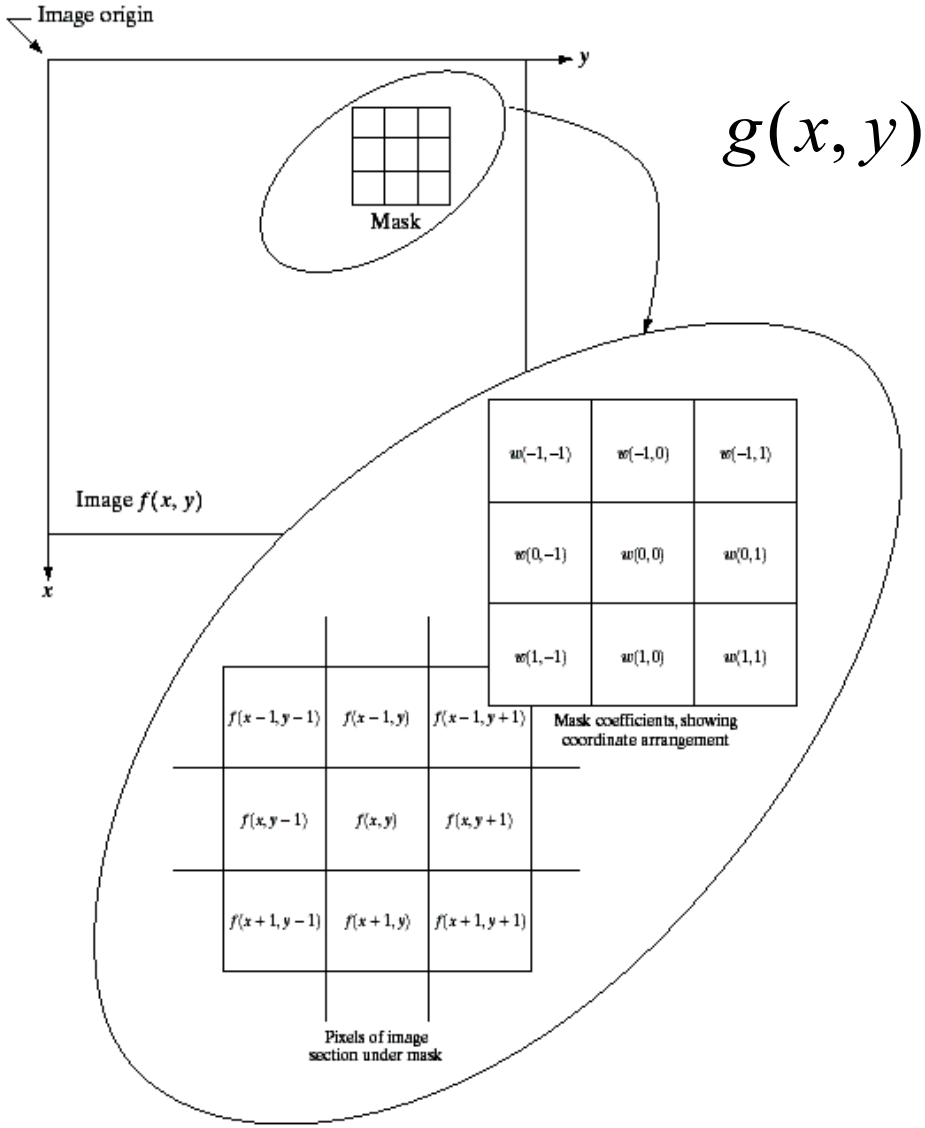
| | | |
|---|---|---|
| *r* | *s* | *t* |
| *u* | *v* | *w* |
| *x* | *y* | *z* |

**Filter**

$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

The above is repeated for every pixel in the original image to generate the filtered image

# Spatial Filtering: Equation Form




$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Smoothing Spatial Filters

- The shape of the impulse response needed to implement a lowpass (smoothing) filter indicates the filter should have all positive coefficients
- For a 3x3 mask, the simplest arrangement is to have all the coefficient values equal to one (neighborhood averaging)
  - The response would be the sum of all gray levels for the nine pixels in the mask
  - This could cause the value of R to be out of the valid gray-level range
- The solution is to scale the result by dividing by 9

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# Smoothing Spatial Filters

One of the simplest spatial filtering operations we can perform is a smoothing operation

Simply average all of the pixels in a neighbourhood around a central value

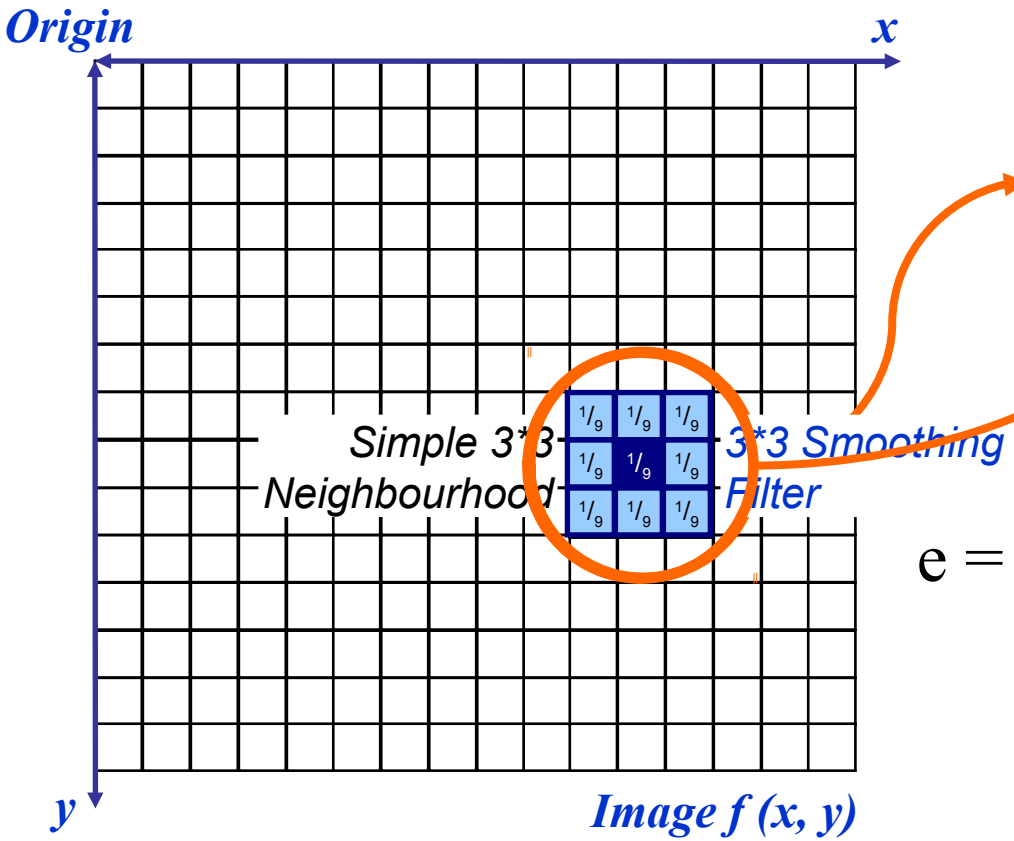
Especially useful in removing noise from images

Also useful for highlighting gross detail

| | | |
|---|---|---|
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |
| $1/9$ | $1/9$ | $1/9$ |

Simple averaging filter

# Smoothing Spatial Filtering




| | | |
|---|---|---|
| 104 | 100 | 108 |
| 99 | 106 | 98 |
| 95 | 90 | 85 |

**Original Image Pixels**

*

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

**Filter**

$$e = \tfrac{1}{9}*106 + \tfrac{1}{9}*104 + \tfrac{1}{9}*100 + \tfrac{1}{9}*108 + \tfrac{1}{9}*99 + \tfrac{1}{9}*98 + \tfrac{1}{9}*95 + \tfrac{1}{9}*90 + \tfrac{1}{9}*85$$

$$= 98.3333$$

The above is repeated for every pixel in the original image to generate the smoothed image

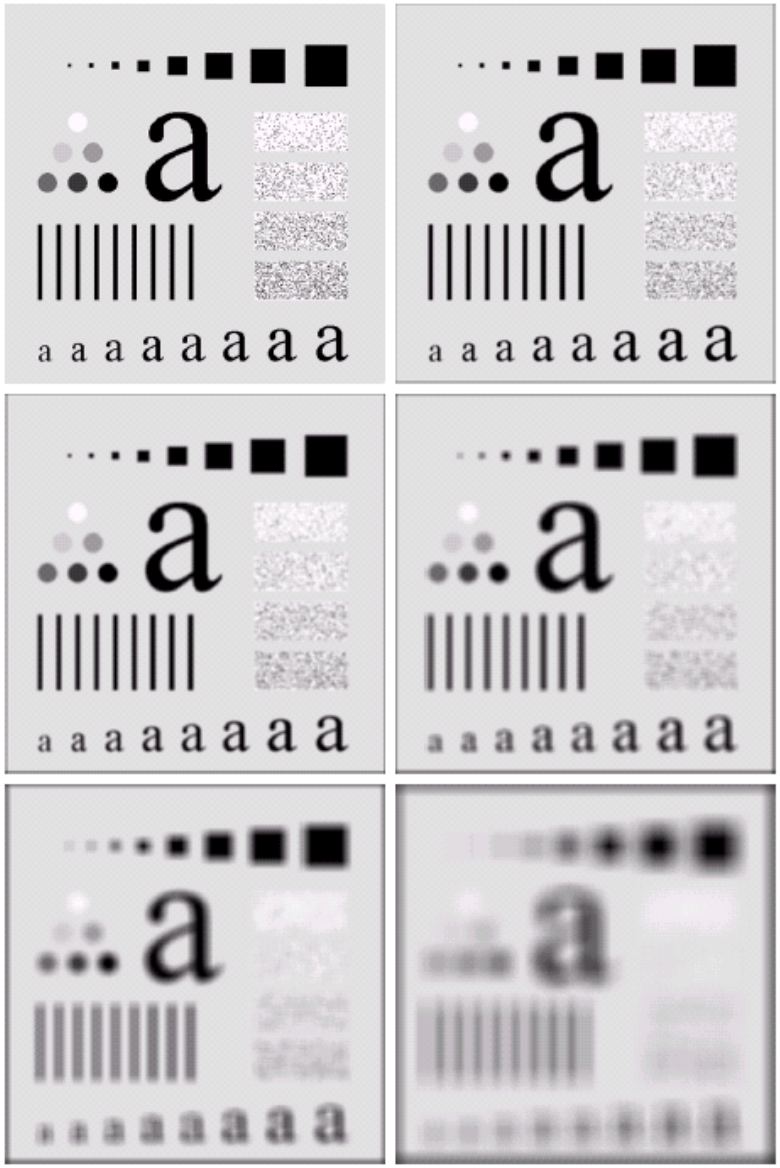
# Image Smoothing Example

The image at the top left is an original image of size 500*500 pixels

The subsequent images show the image after filtering with an averaging filter of increasing sizes

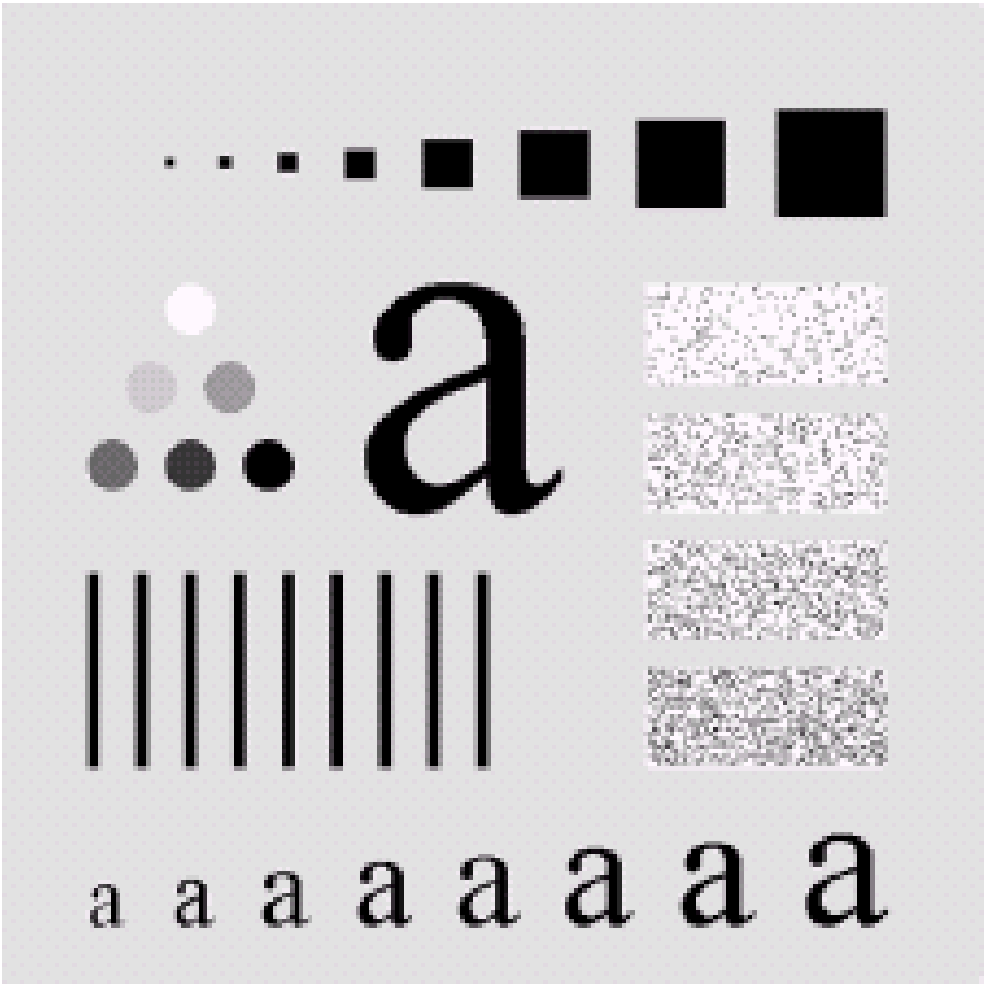
3, 5, 9, 15 and 35

Notice how detail begins to disappear



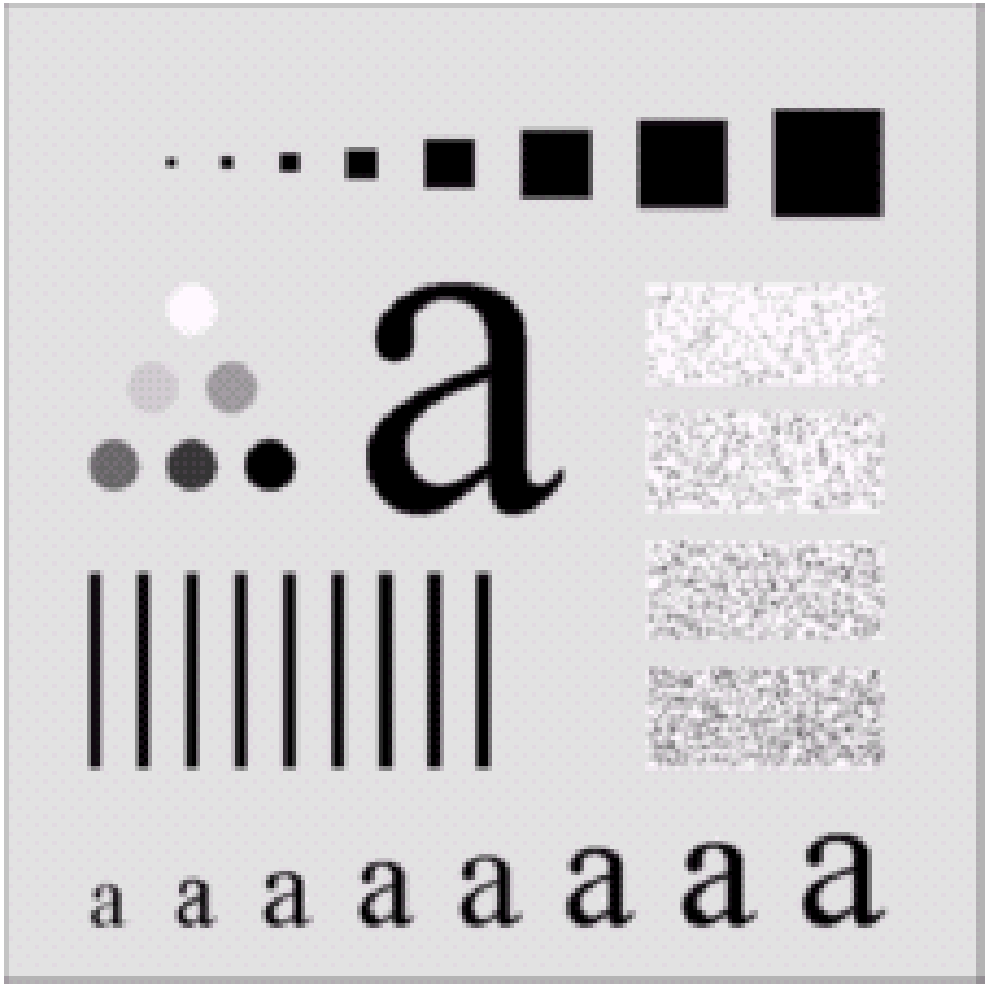
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Smoothing Example



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

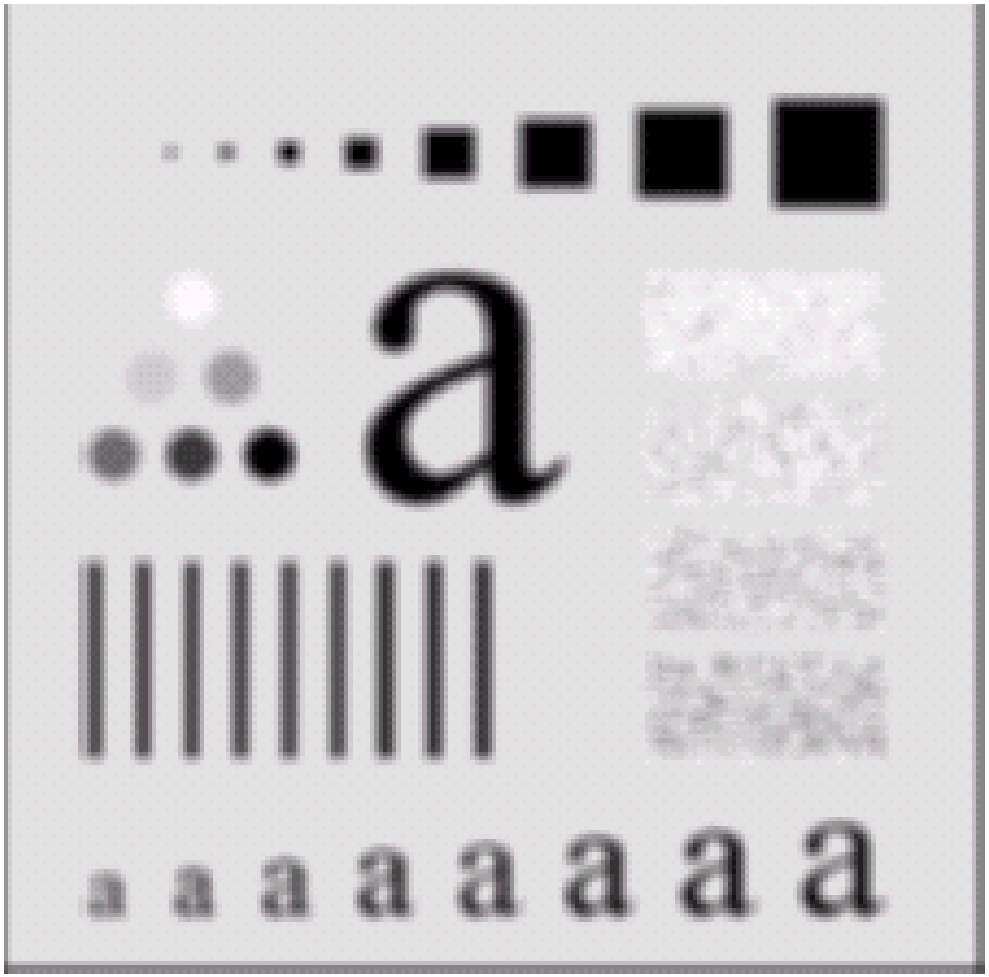
# Image Smoothing Example



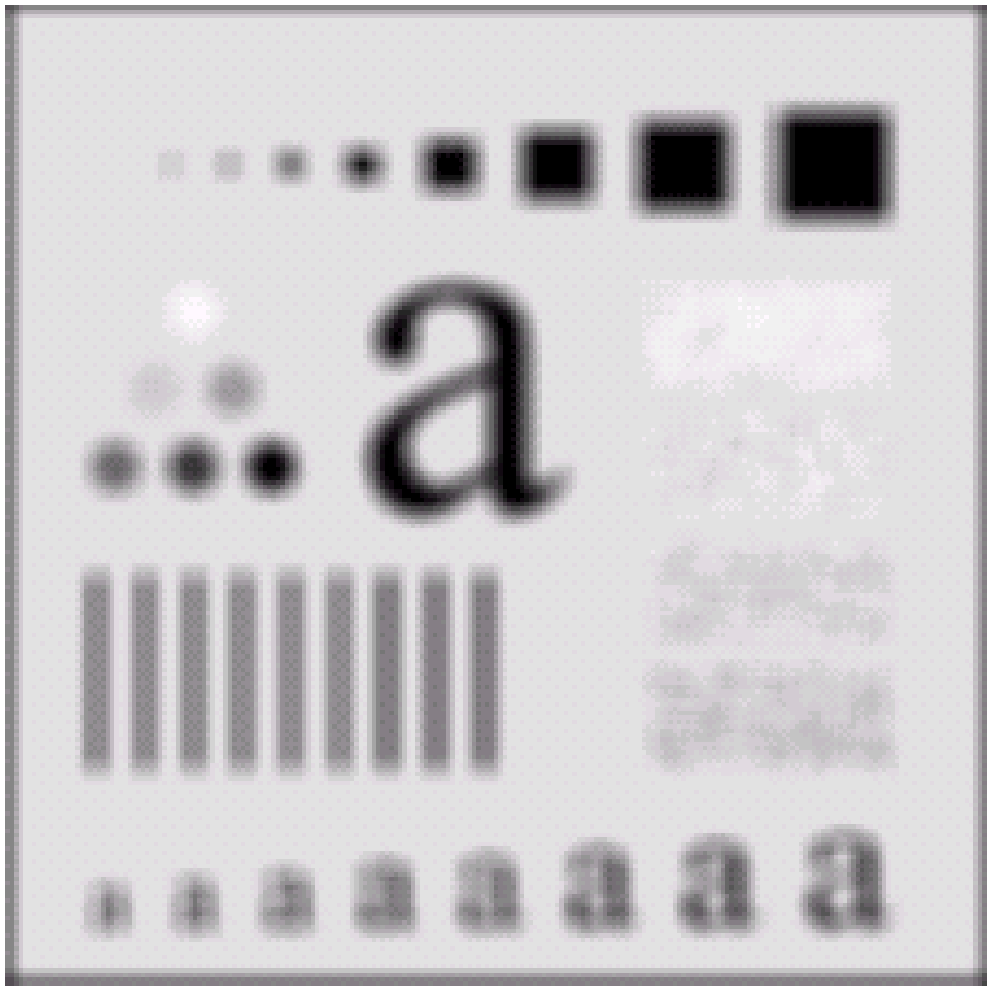
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Smoothing Example

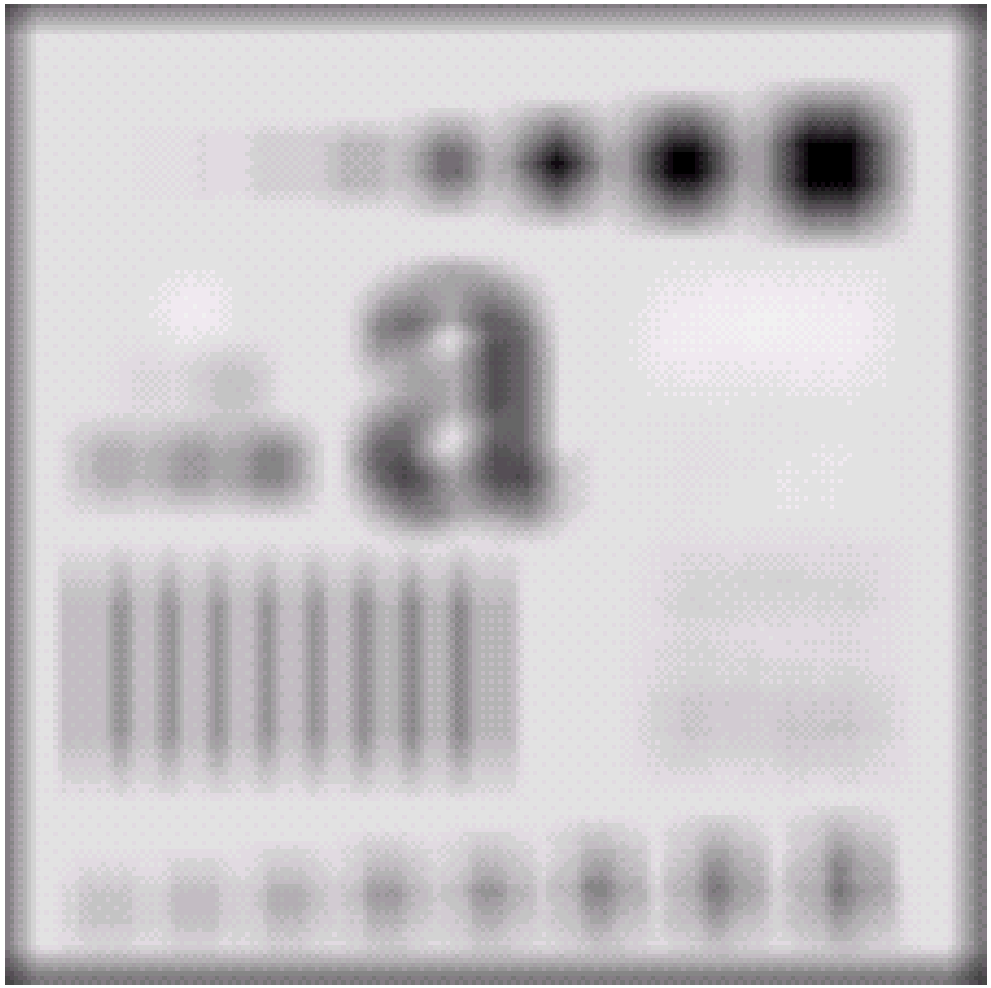
# Image Smoothing Example



# Image Smoothing Example



# Image Smoothing Example



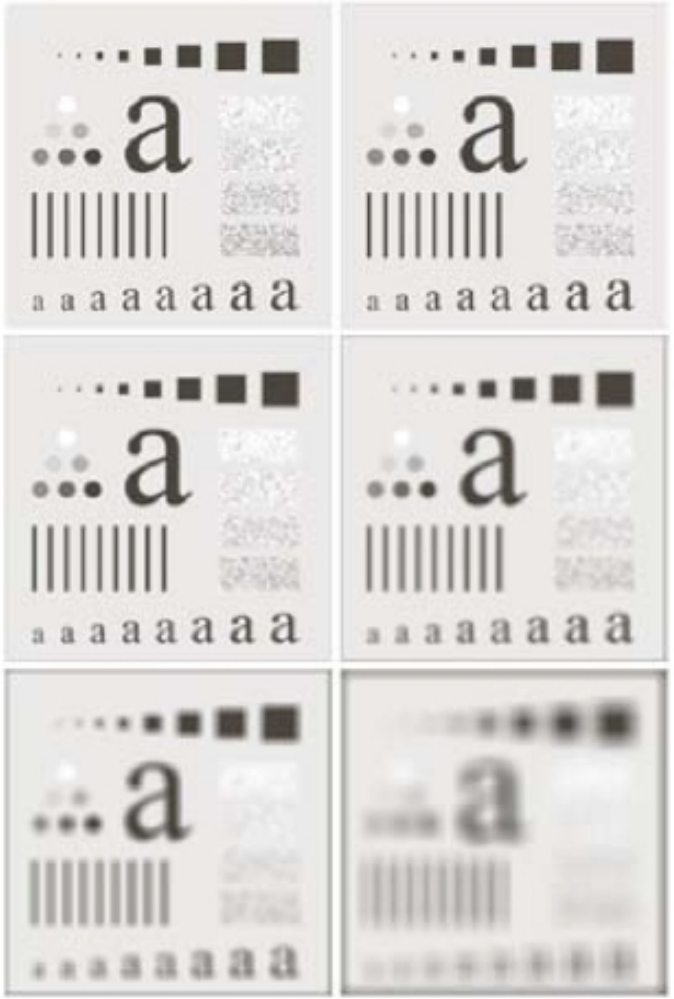
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Smoothing Example



**FIGURE 3.33** (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

a b
c d
e f



# Weighted Smoothing Filters

More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function

Pixels closer to the central pixel are more important

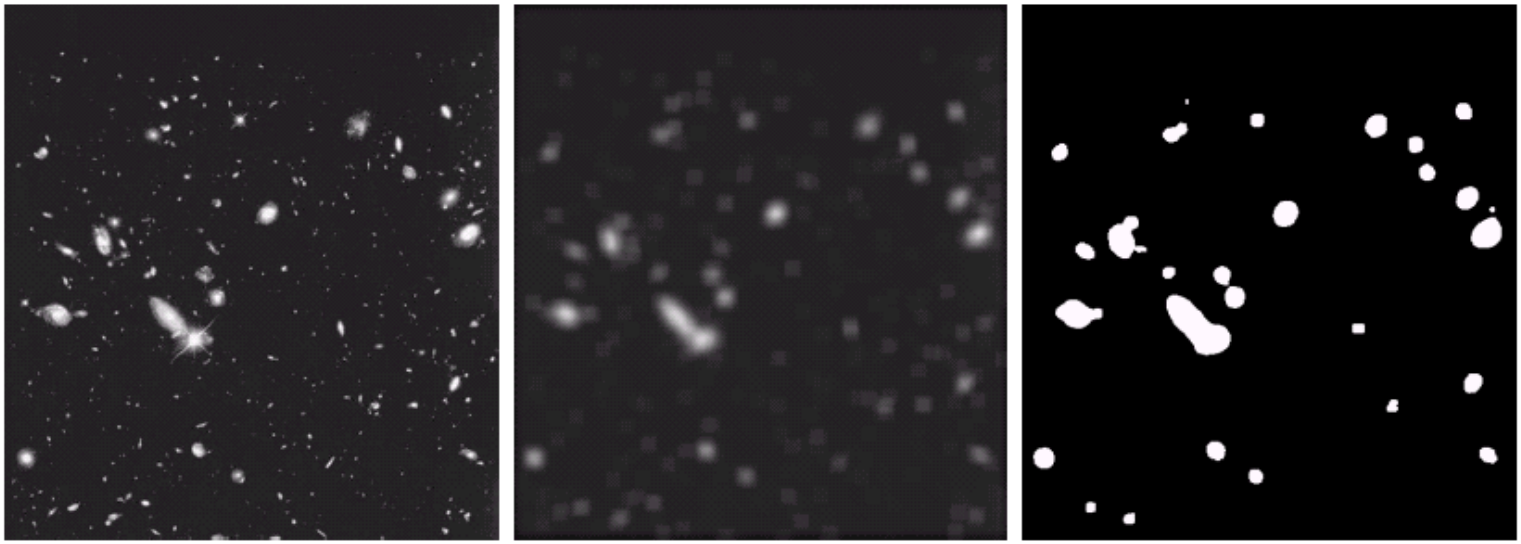
Often referred to as a *weighted averaging*

| | | |
|---|---|---|
| $^1/_{16}$ | $^2/_{16}$ | $^1/_{16}$ |
| $^2/_{16}$ | $^4/_{16}$ | $^2/_{16}$ |
| $^1/_{16}$ | $^2/_{16}$ | $^1/_{16}$ |

Weighted averaging filter

# Another Smoothing Example

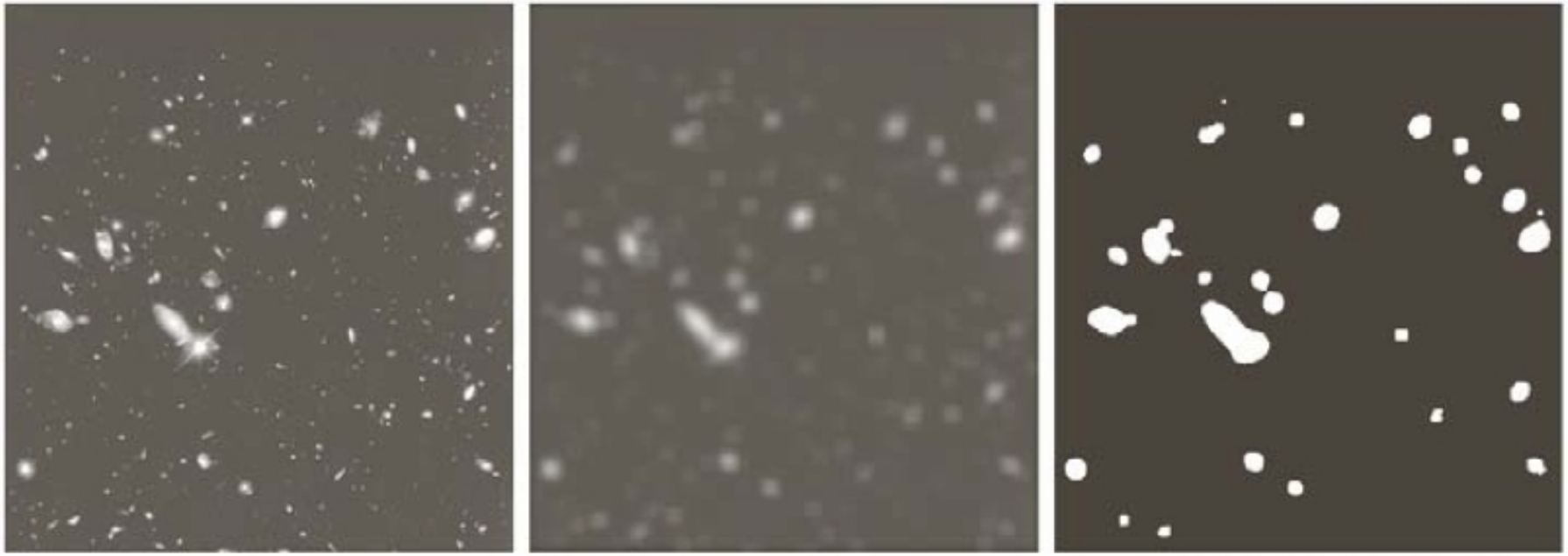
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



**Original Image** **Smoothed Image** **Thresholded Image**

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Another Smoothing Example



a b c

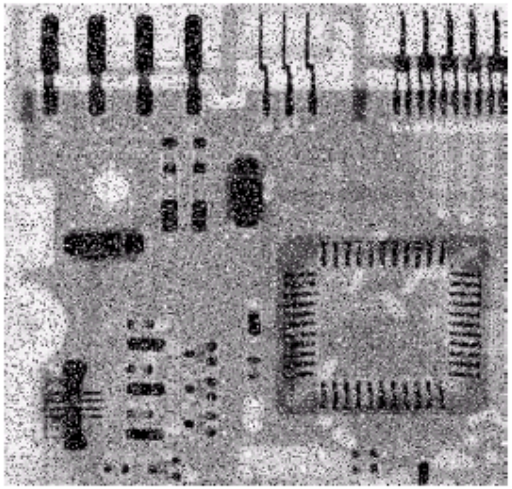
**FIGURE 3.34** (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Smoothing Filters

- One problem with the lowpass filter is it blurs edges and other sharp details
- If the intent is to achieve noise reduction, one approach can be to use median filtering
  - The value of each pixel is replaced by the median pixel value in the neighborhood (as opposed to the average)
  - Particularly effective when the noise consists of strong, spike like components and edge sharpness is to be preserved
- The median m of a set of values is such that half of the values are greater than m and half are less than m
- To implement, sort the pixel values in the neighborhood, choose the median and assign this value to the pixel of interest
- Forces pixels with distinct intensities to be more like their neighbors

# Averaging Filter Vs. Median Filter Example



**Original Image With Noise**

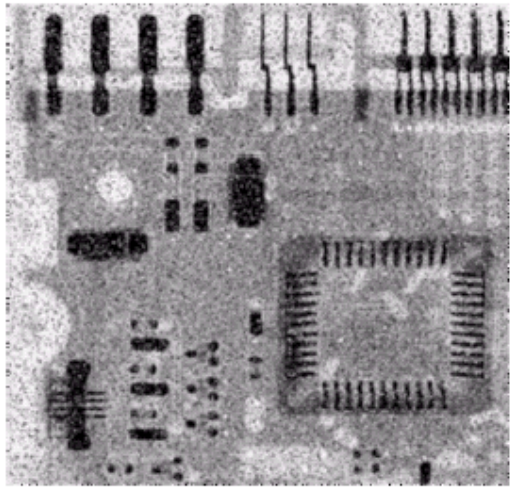


**Image After Averaging Filter**

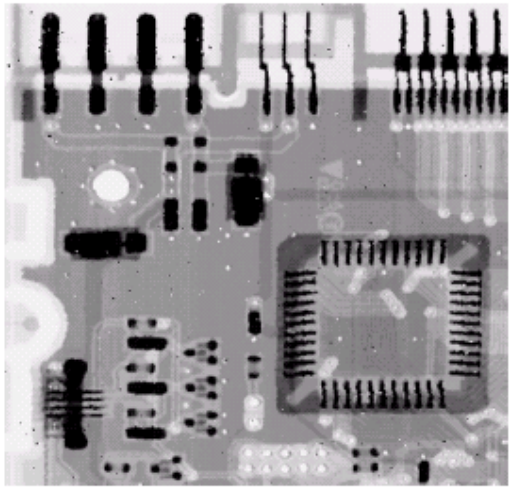
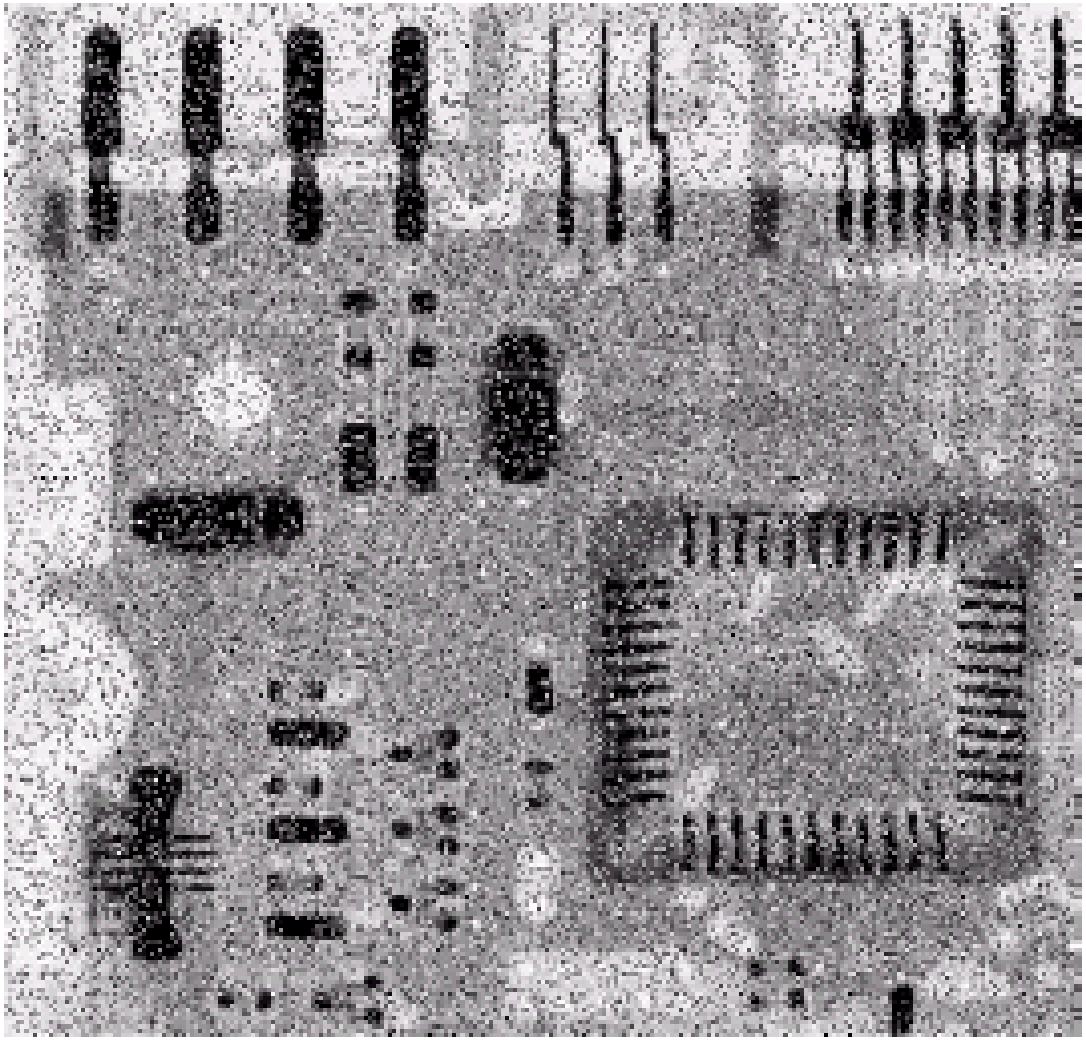


**Image After Median Filter**

Filtering is often used to remove noise from images

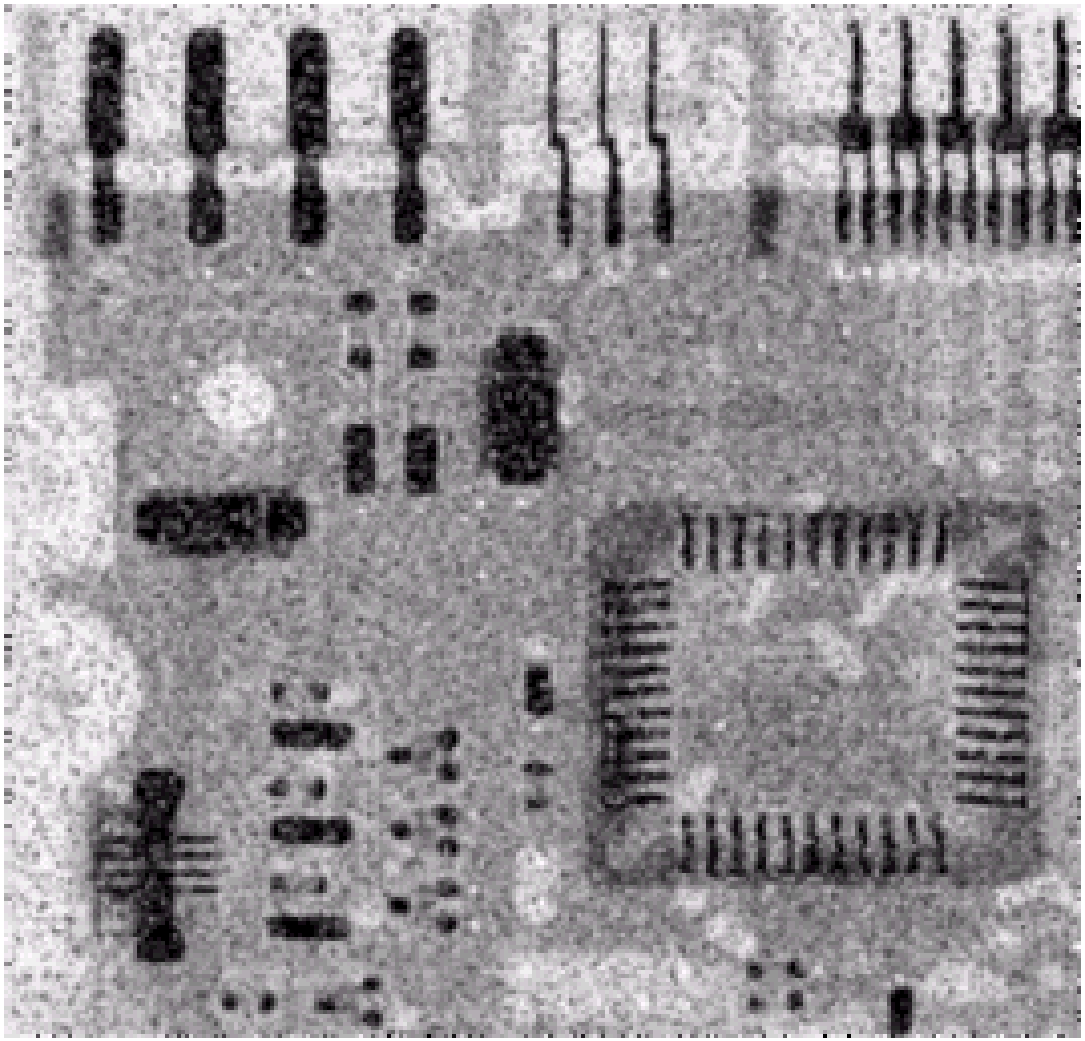
Sometimes a median filter works better than an averaging filter

# Averaging Filter Vs. Median Filter Example



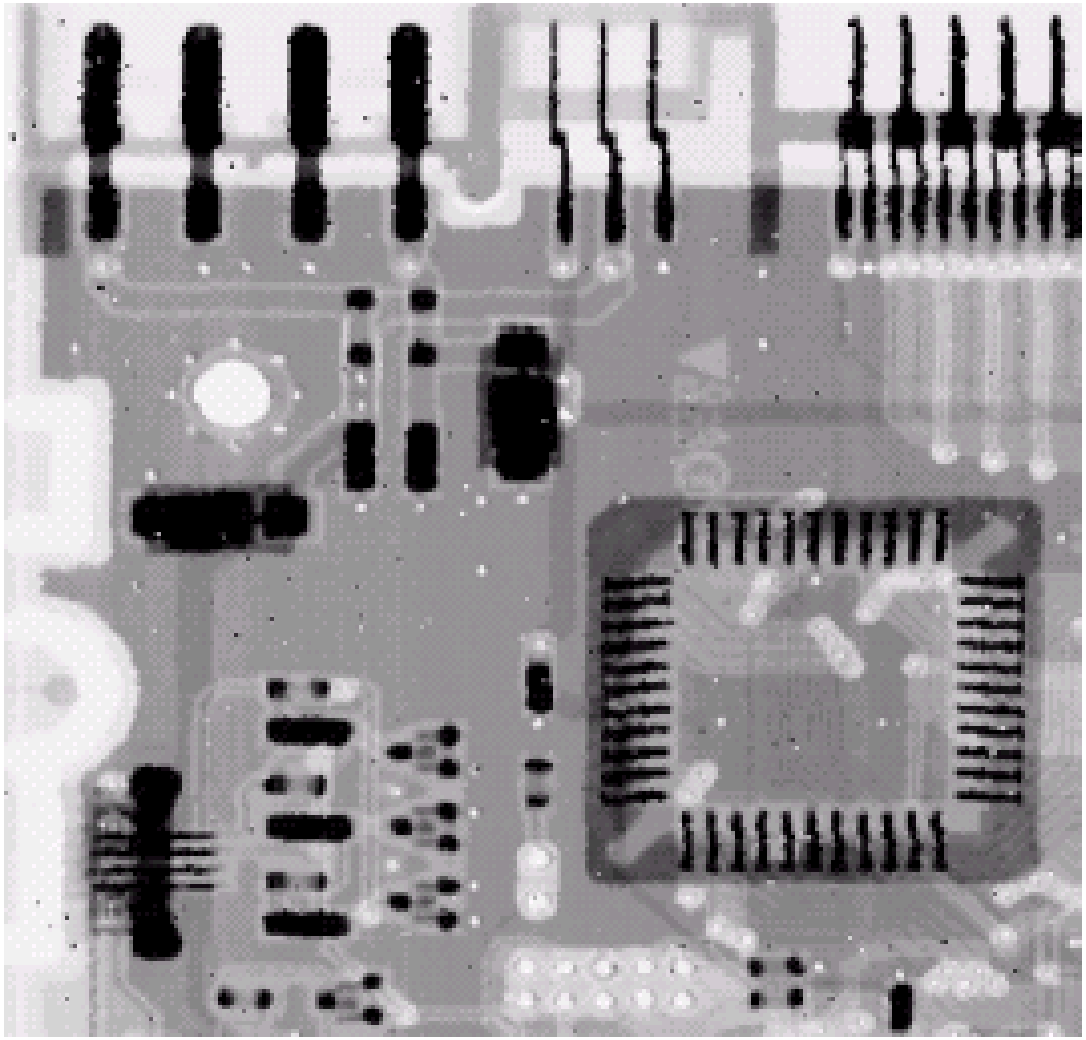
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Averaging Filter Vs. Median Filter Example



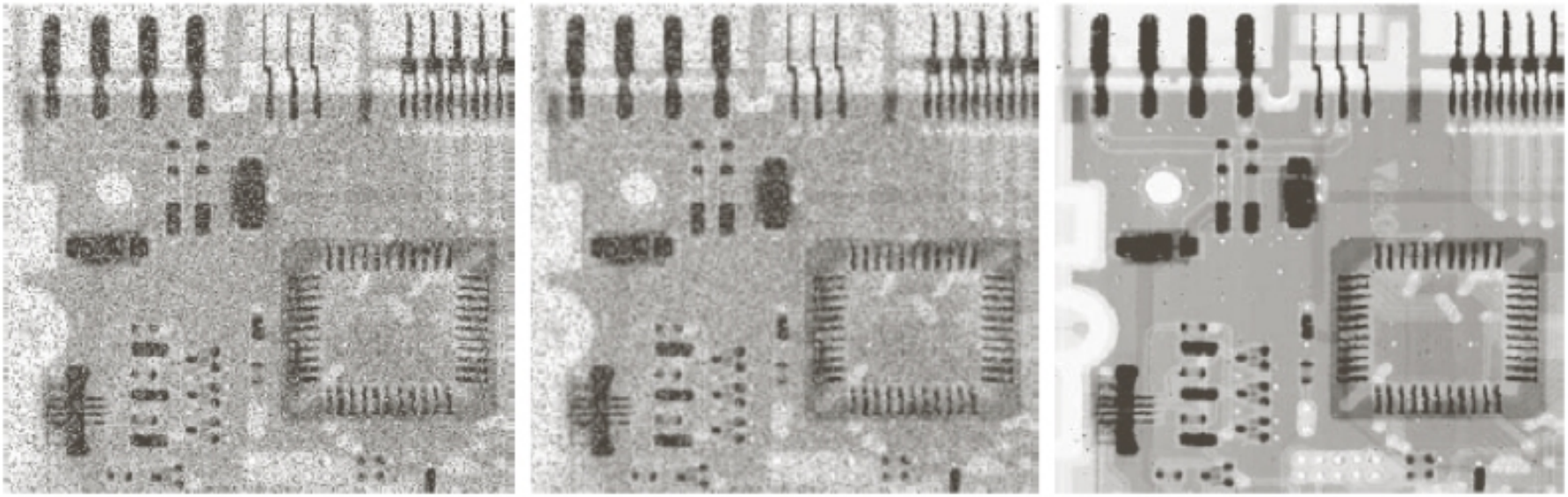
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Averaging Filter Vs. Median Filter Example



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Averaging Filter Vs. Median Filter Example



a b c

**FIGURE 3.35** (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a $3 \times 3$ averaging mask. (c) Noise reduction with a $3 \times 3$ median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

# Sharpening Filters (High Pass)

- The shape of the impulse response needed to implement a high-pass (sharpening) filter indicates the filter should have positive coefficients near its center and negative coefficients in the outer periphery
- For a 3x3 mask, the simplest arrangement is to have the center coefficient positive and all others negative

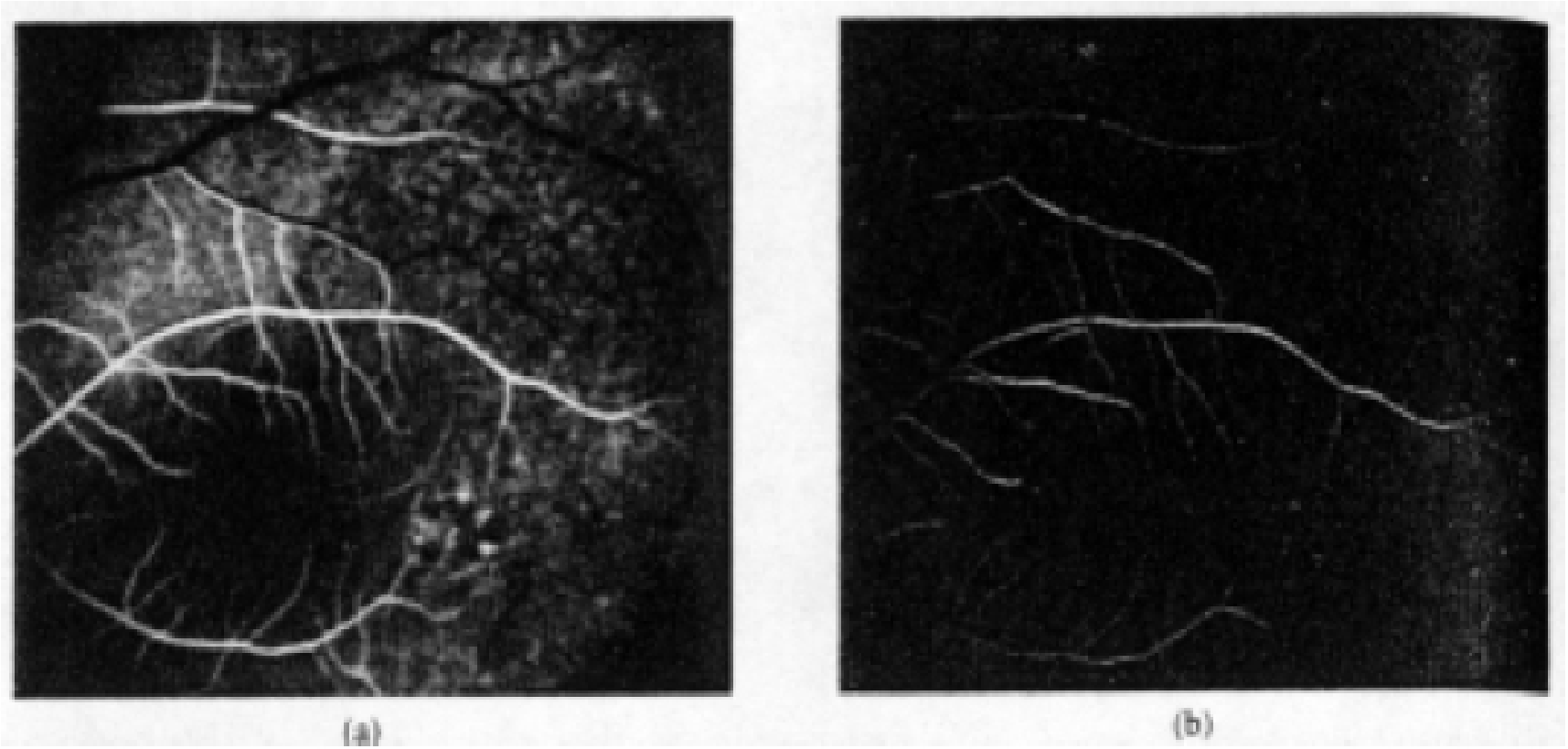
$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

# Sharpening Filters (High Pass)

- Note the sum of the coefficients is zero
  - When the mask is over a constant or slowly varying region the output is zero or very small
    - This filter eliminates the zero-frequency term
    - Eliminating this term reduces the average gray-level value in the image to zero (will reduce the global contrast of the image)
    - Result will be a some what edge-enhanced image over a dark background
- Reducing the average gray-level value to zero implies some negative gray levels
  - The output should be scaled back into an appropriate range

    [0, L-1]

# Sharpening Filters (High Pass)



# Sharpening Filters (High-boost)

- A high-pass filter may be computed as:

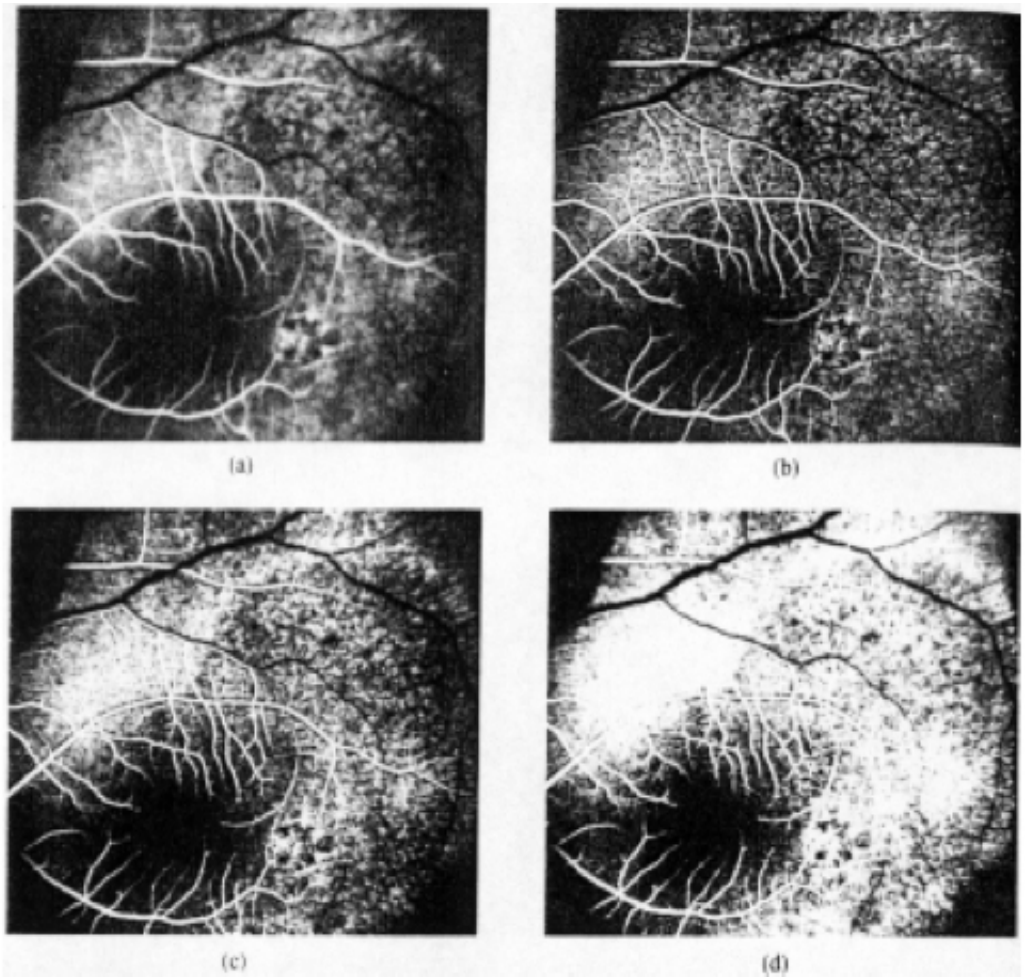
$$\text{High-pass} = \text{Original} - \text{Lowpass}$$

- Multiplying the original by an amplification factor yields a high-boost or high-frequency-emphasis filter

$$\begin{aligned}\text{High-boost} &= A(\text{Original}) - \text{Lowpass} \\ &= (A-1)(\text{Original}) + \text{Original} - \text{Lowpass} \\ &= (A-1)(\text{Original}) + \text{High-pass}\end{aligned}$$

  - If A>1, part of the original image is added to the high-pass result (partially restoring low frequency components)
  - Result looks more like the original image with a relative degree of edge enhancement that depends on the value of A
  - May be implemented with the center coefficient value $w=9A-1$ $(A \geq 1)$

# Sharpening Filters (High-boost)

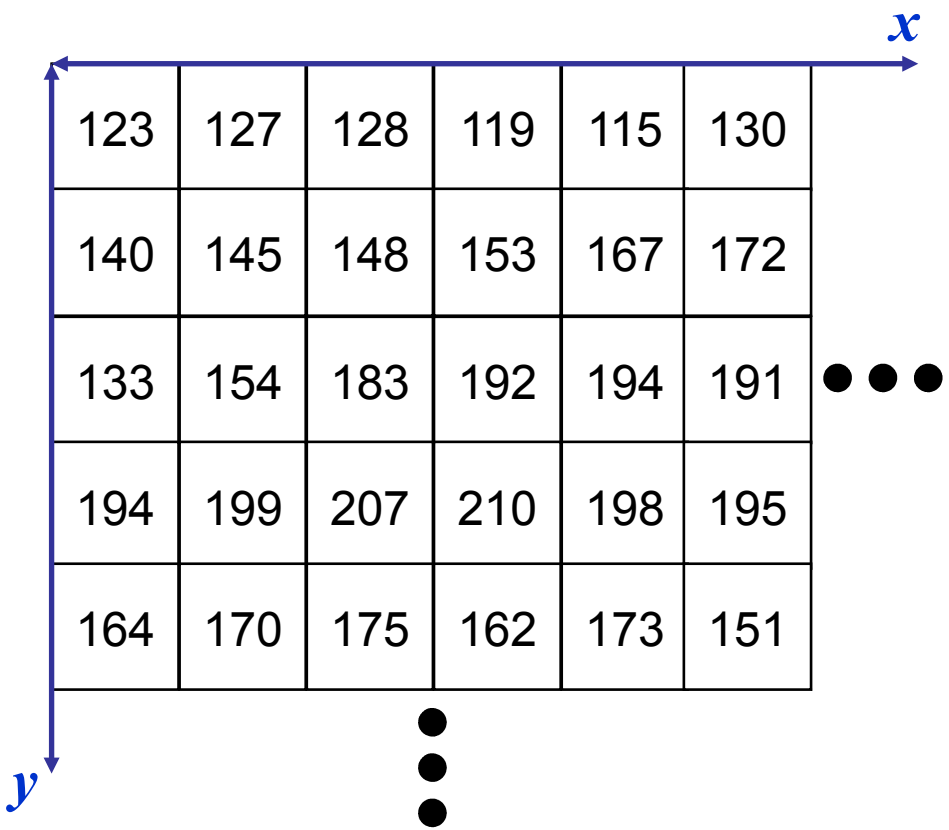



A=1.1

A=1.15

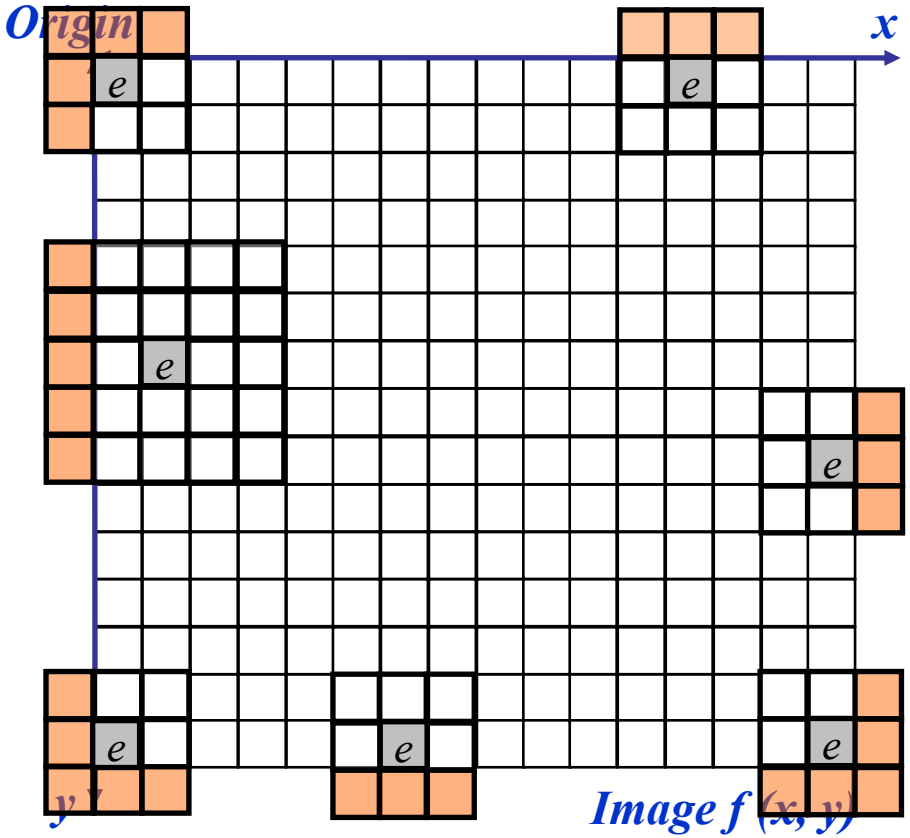
A=1.2

# Simple Neighbourhood Operations Example



# Strange Things Happen At The Edges!

At the edges of an image we are missing pixels to form a neighbourhood



# Strange Things Happen At The Edges! (cont…)

There are a few approaches to dealing with missing edge pixels:

Omit missing pixels

- Only works with some filters
- Can add extra code and slow down processing

Pad the image

- Typically with either all white or all black pixels

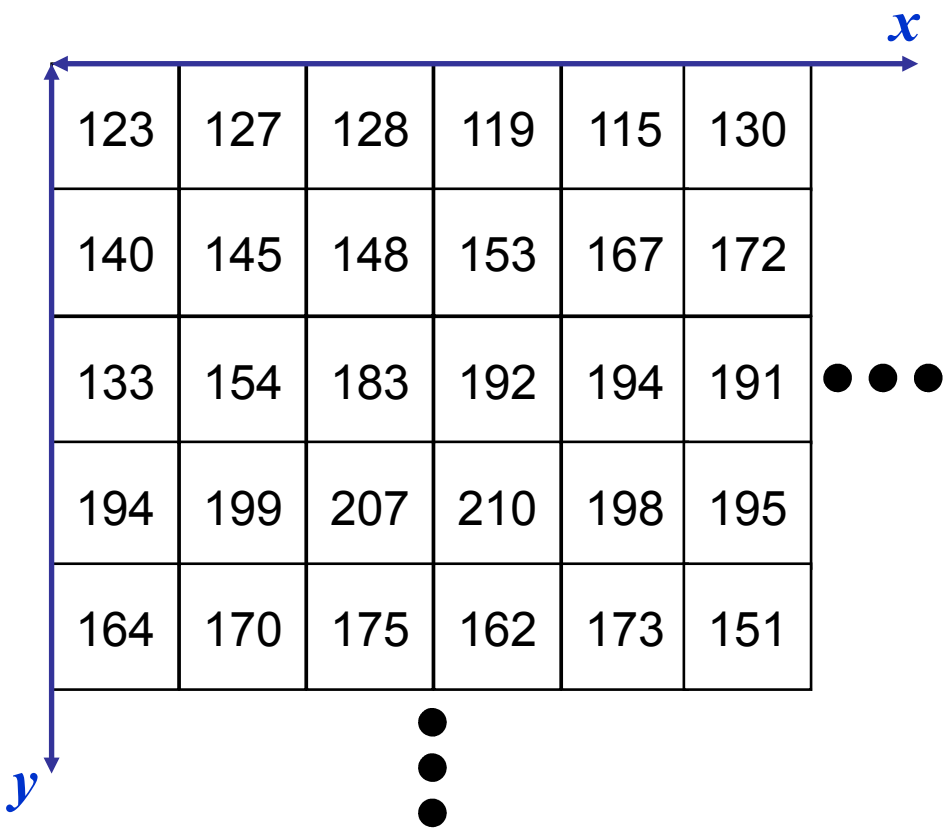
Replicate border pixels

Truncate the image

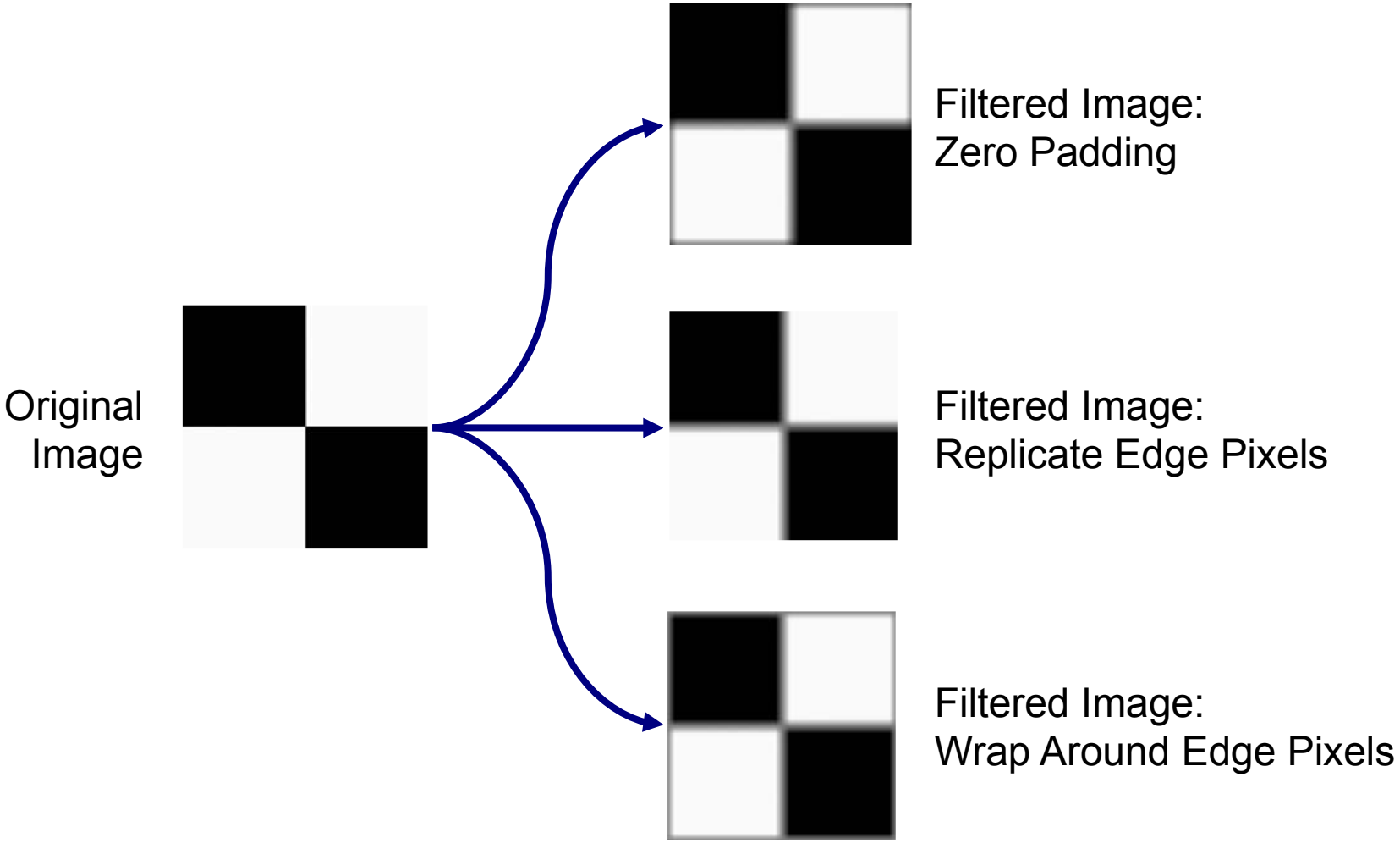
Allow pixels *wrap around* the image

- Can cause some strange image artefacts

# Simple Neighbourhood Operations Example

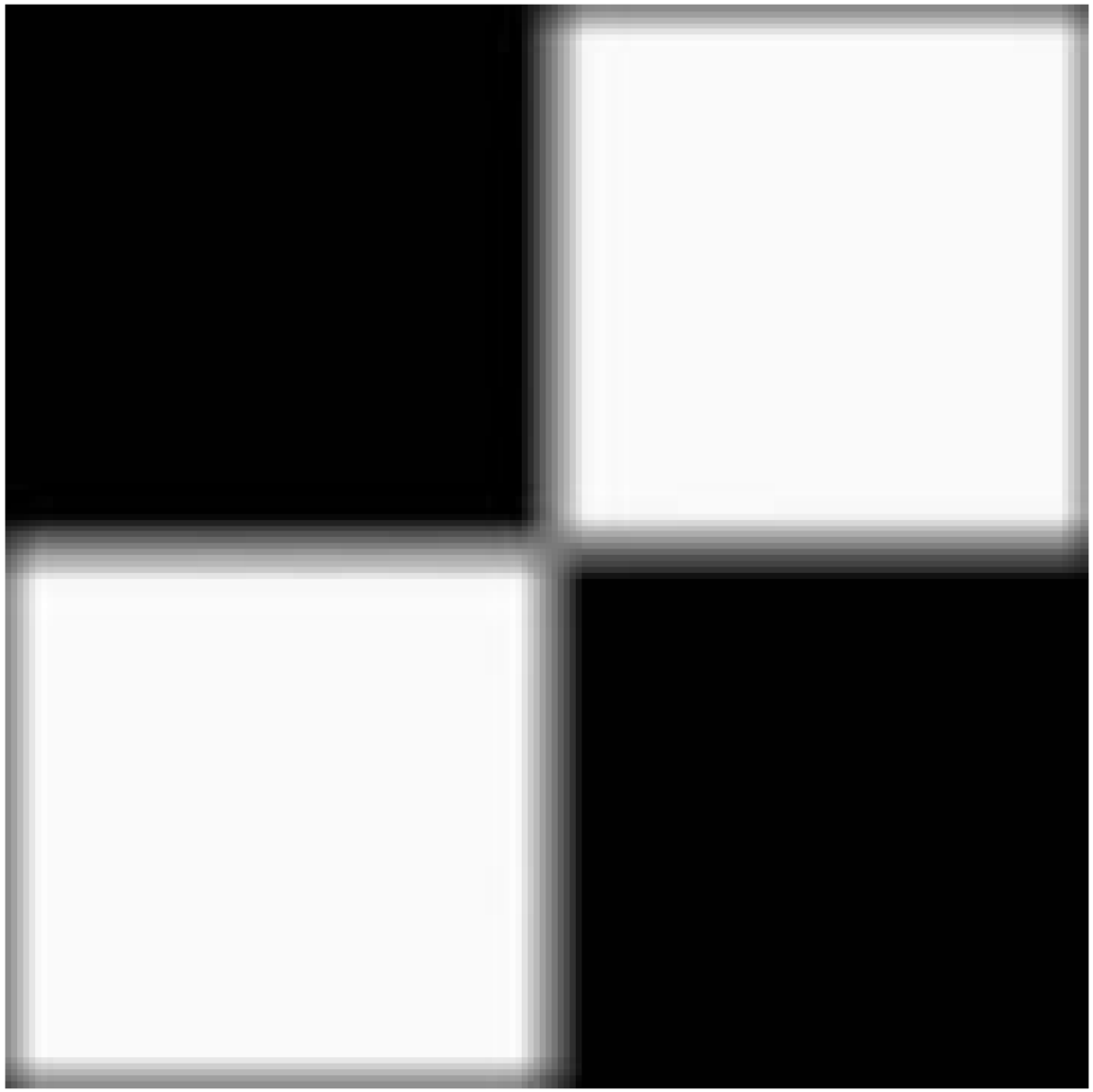


# Strange Things Happen At The Edges! (cont…)



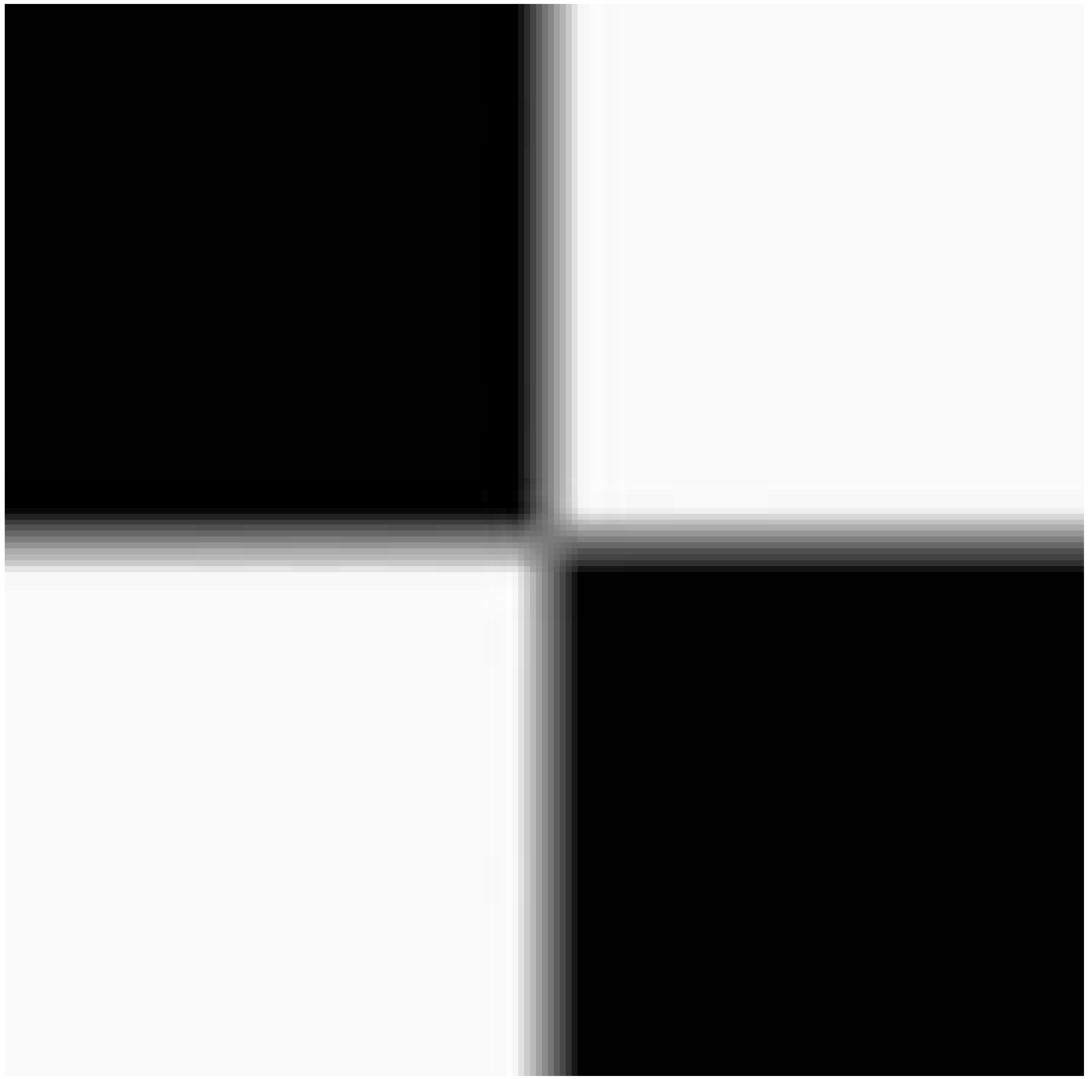

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Strange Things Happen At The Edges! (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Strange Things Happen At The Edges! (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Strange Things Happen At The Edges! (cont…)

# Correlation & Convolution

The filtering we have been talking about so far is referred to as *correlation* with the filter itself referred to as the *correlation kernel*

*Convolution* is a similar operation, with just one subtle difference

| | | |
|---|---|---|
| *a* | *b* | *c* |
| *d* | *e* | *e* |
| *f* | *g* | *h* |

**Original Image Pixels**

$*$

| | | |
|---|---|---|
| *r* | *s* | *t* |
| *u* | *v* | *w* |
| *x* | *y* | *z* |

**Filter**

$$e_{processed} = v*e + z*a + y*b + x*c + w*d + u*e + t*f + s*g + r*h$$

For symmetric filters it makes no difference