What Is Image Enhancement?

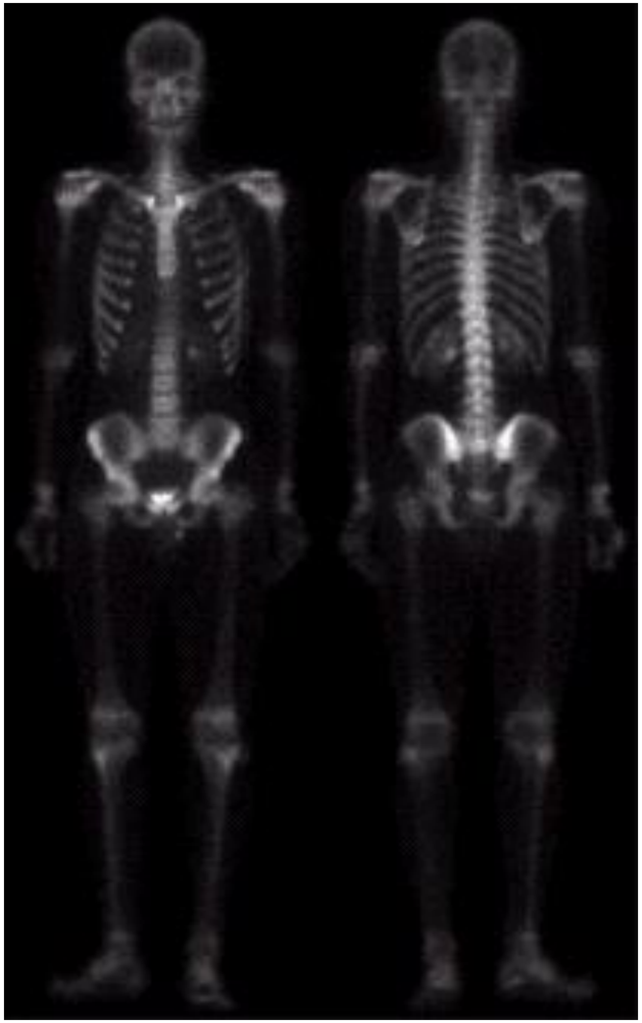
Image enhancement is the process of making images more useful

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…)



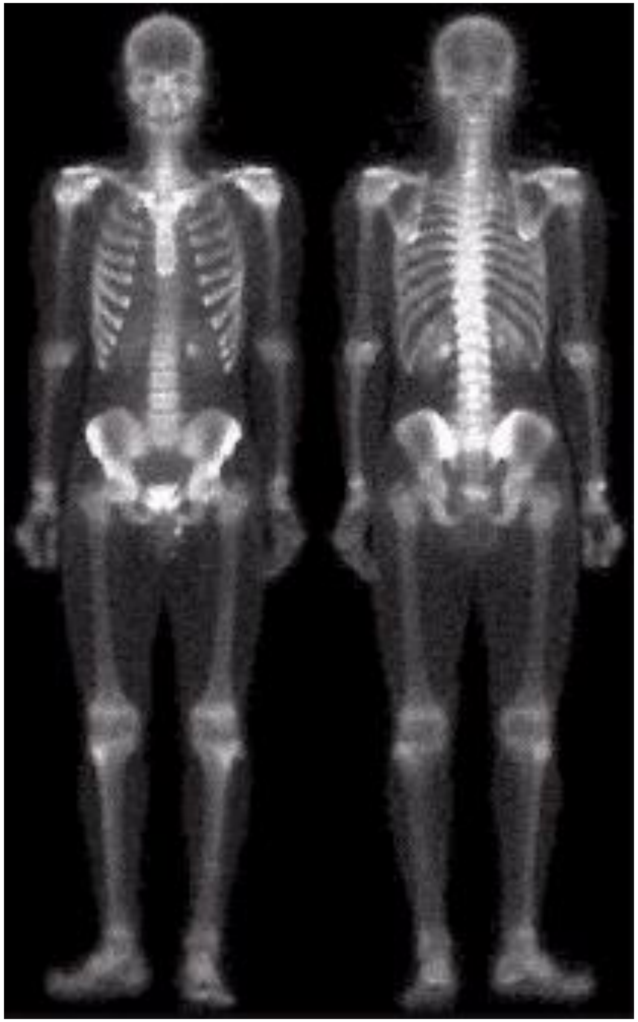
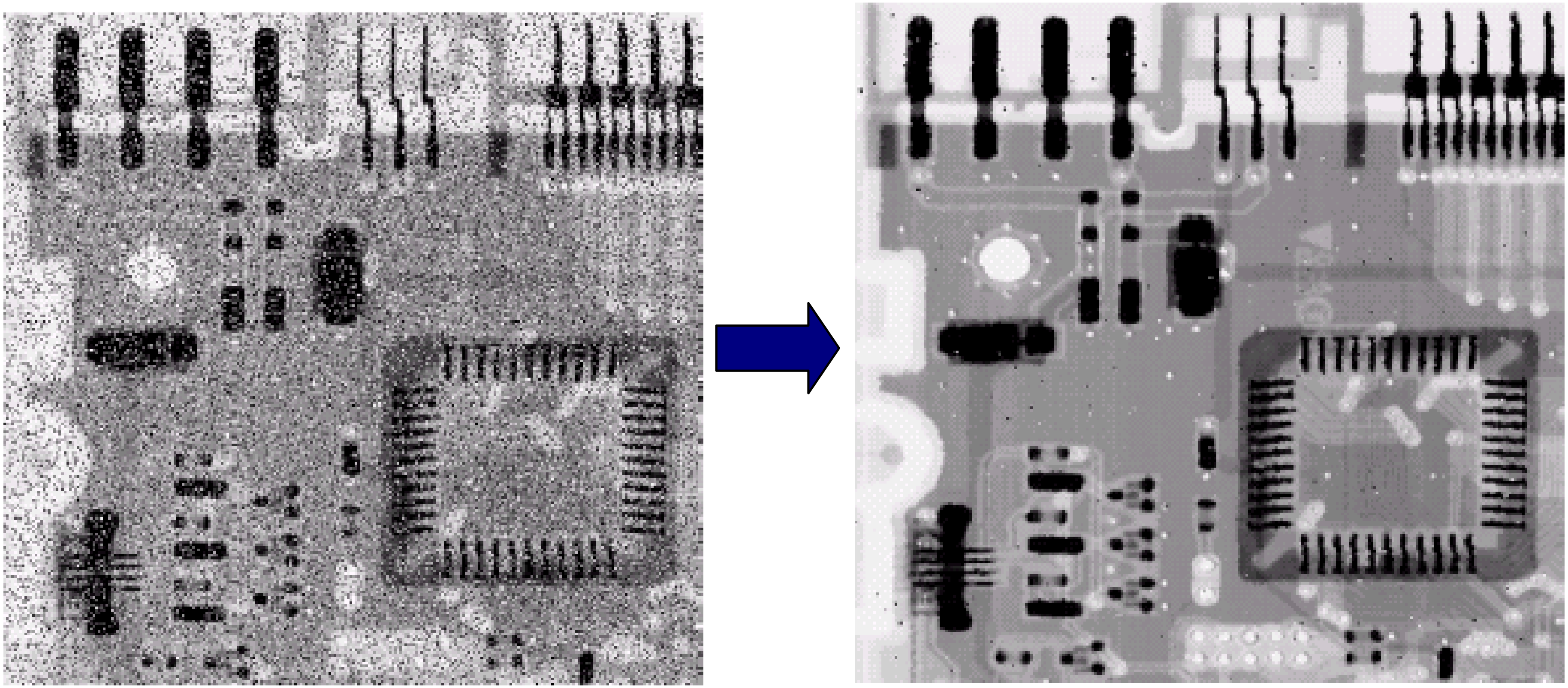


Image Enhancement Examples (cont…)



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

Contents

A Note About Grey Levels

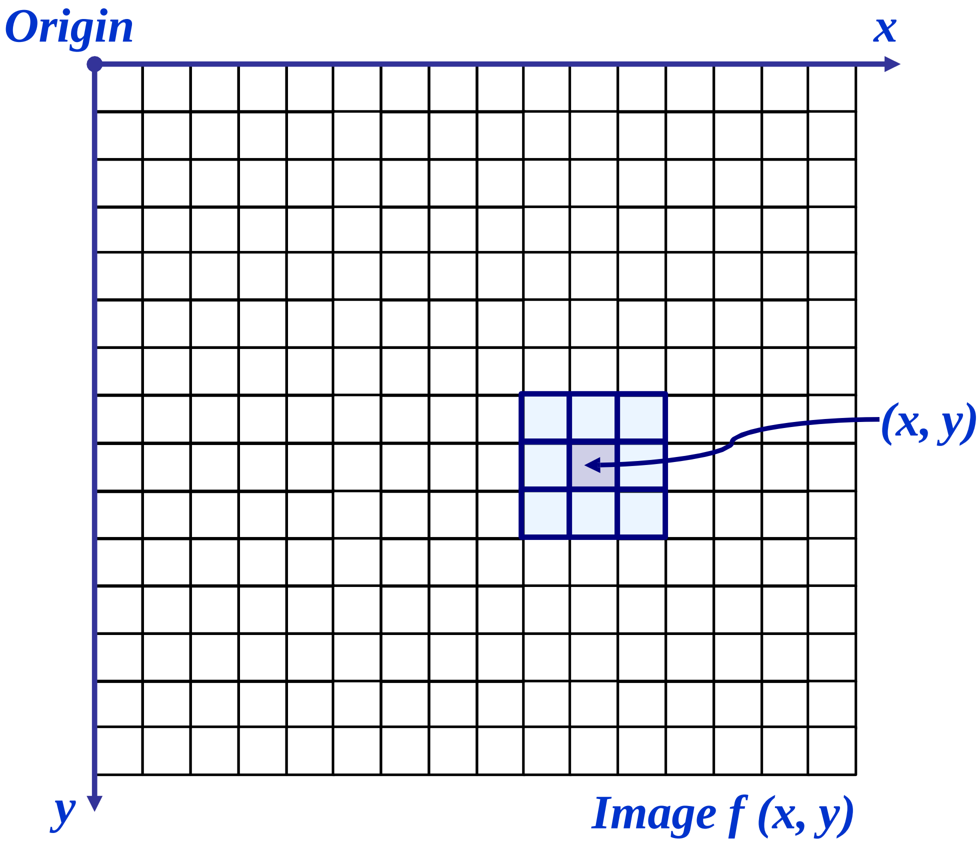
- So far when we have spoken about image grey level values we have said they are in the range [0, 255]
  - Where 0 is black and 255 is white
- There is no reason why we have to use this range
  - The range [0,255] stems from display technologes
- For many of the image processing operations, grey levels 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)



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.

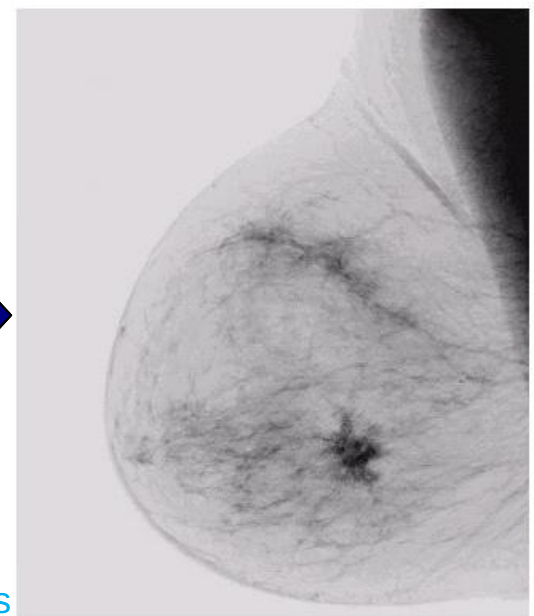
Point Processing Example: Negative Images

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

Original Image

$s = 1.0 - r$

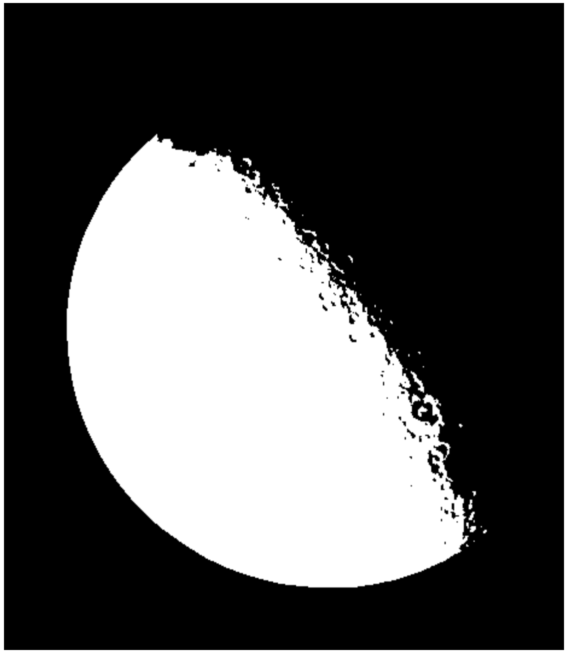


Negative Image

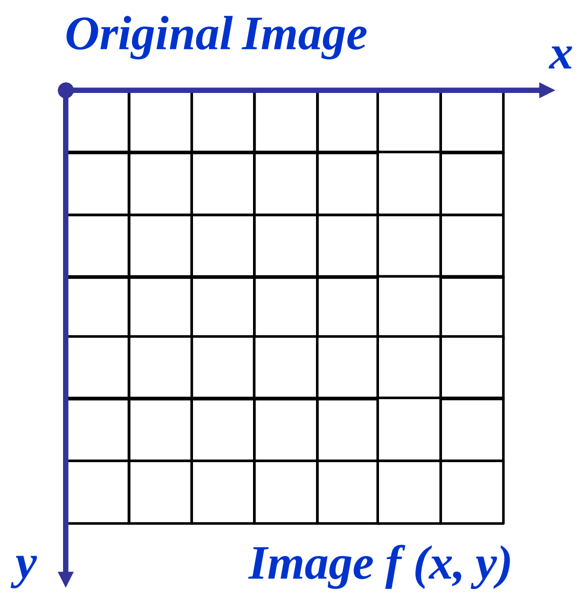
Point Processing Example: Thresholding

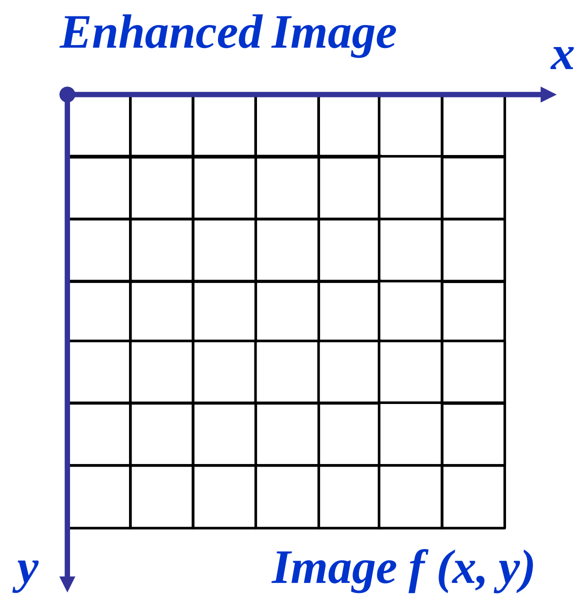
Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background

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



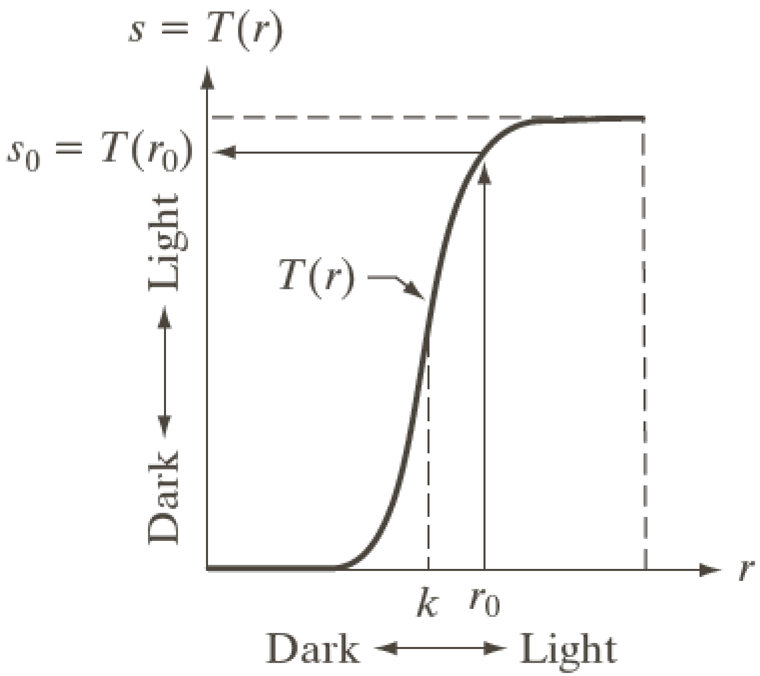
Point Processing Example: Thresholding (cont…)



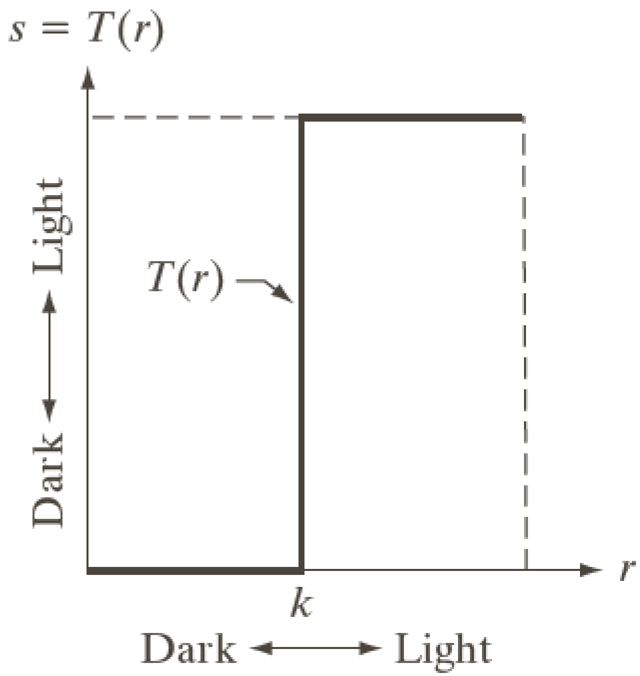


$$s = \begin{cases} 1.0 & r > threshold \\ 0.0 & r <= threshold \end{cases}$$

Intensity Transformations



Contrast stretching



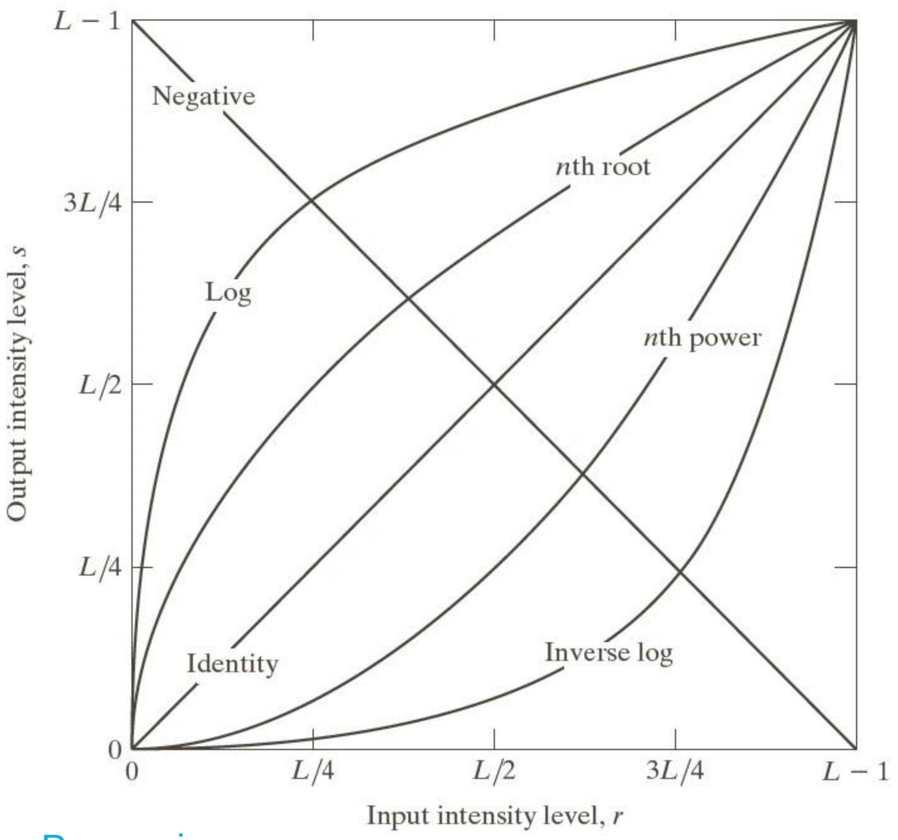
Thresholding

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

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

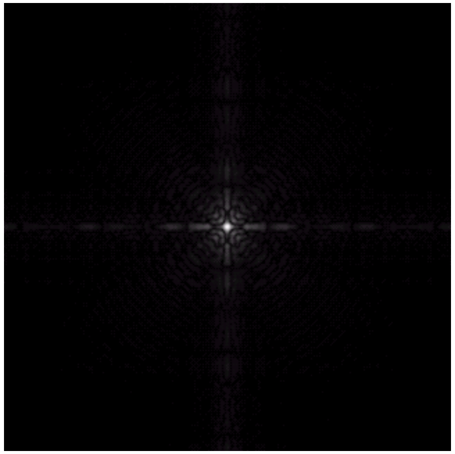
The log transformation maps a narrow range of low input grey level values into a wider range of output values

The inverse log transformation performs the opposite transformation

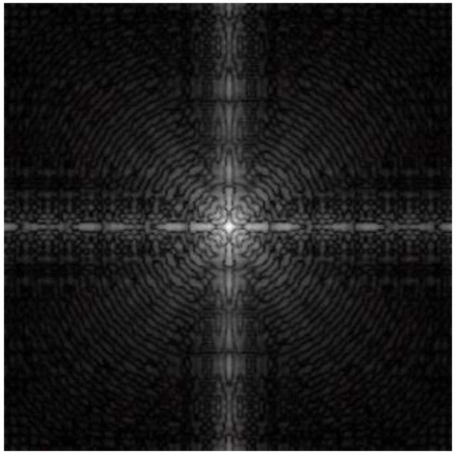
Logarithmic Transformations (cont…)

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

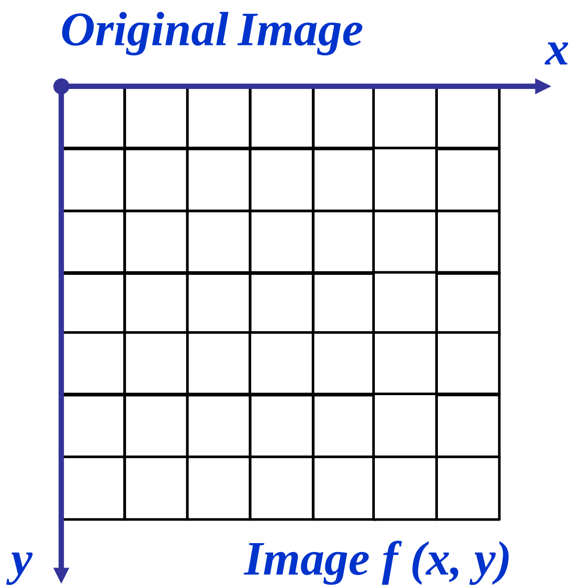
In the following example the Fourier transform of an image is put through a log transform to reveal more detail

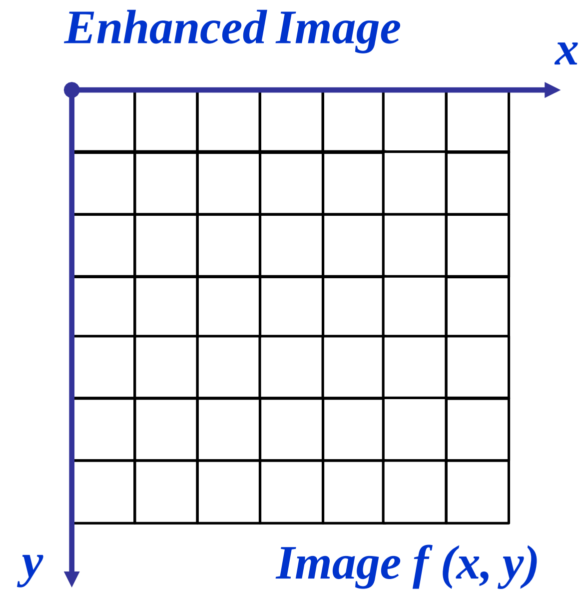


$s = \log(1 + r)$



Logarithmic Transformations (cont…)





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

We usually set c to 1.

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

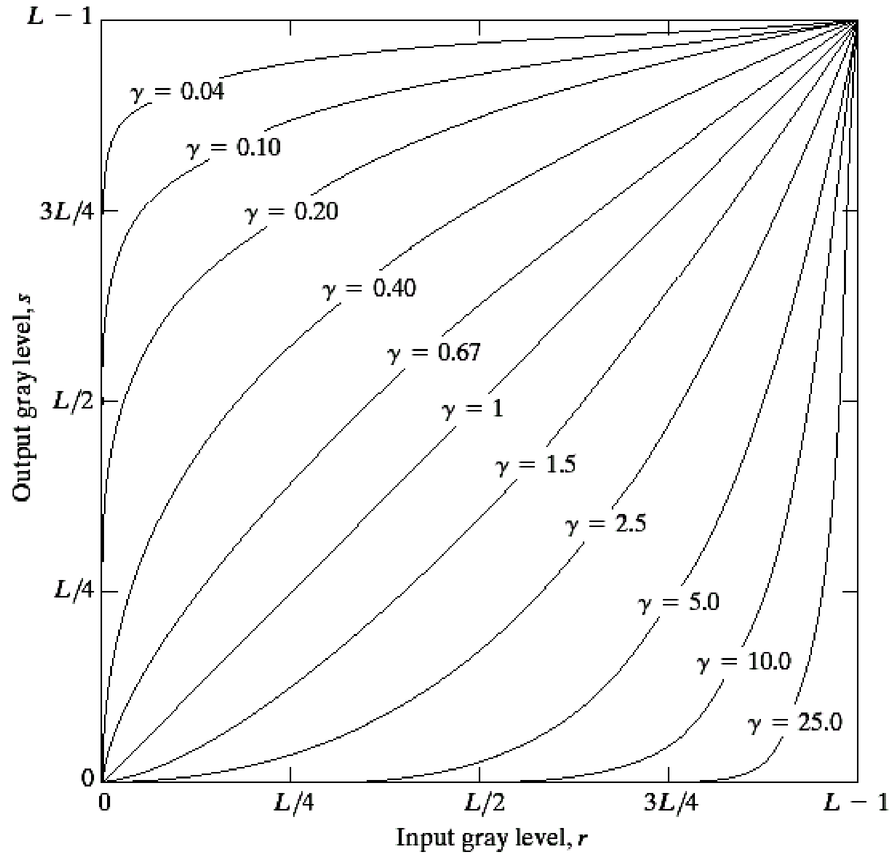
Power Law Transformations

Power law transformations have the following form

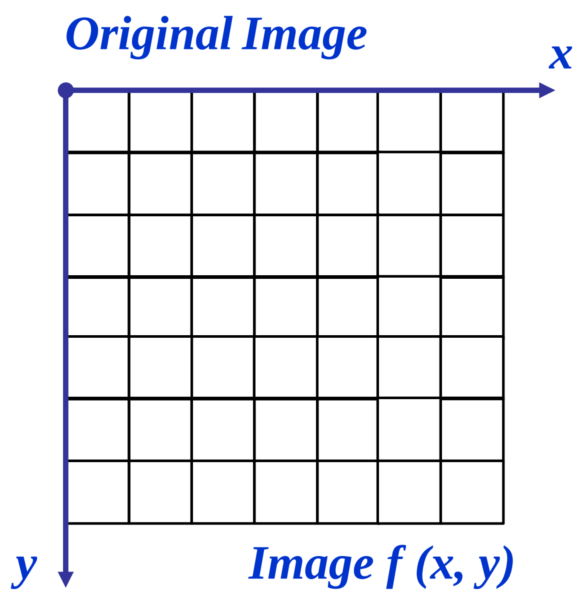
$$s = c * r^{\gamma}$$

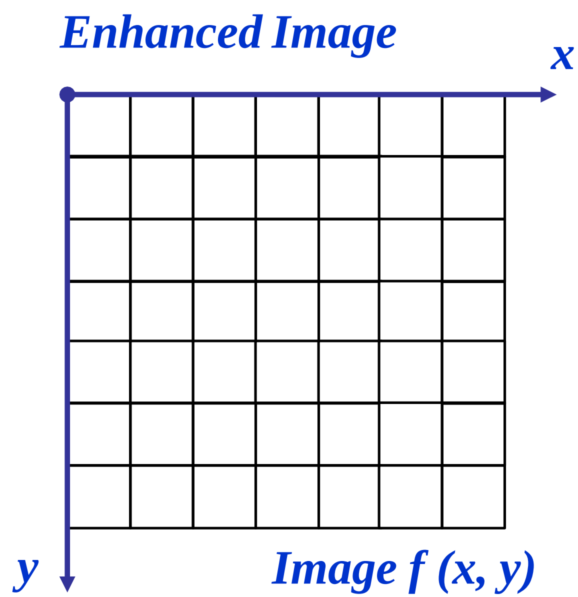
Map a narrow range of dark input values into a wider range of output values or vice versa

Varying γ gives a whole family of curves



Power Law Transformations (cont…)





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

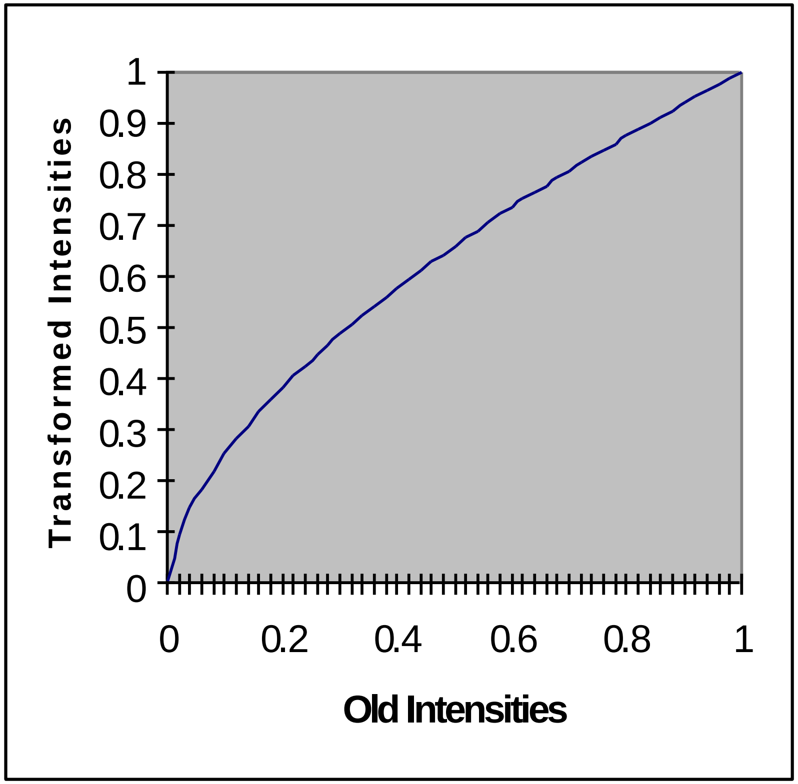
We usually set c to 1.

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

Power Law Example

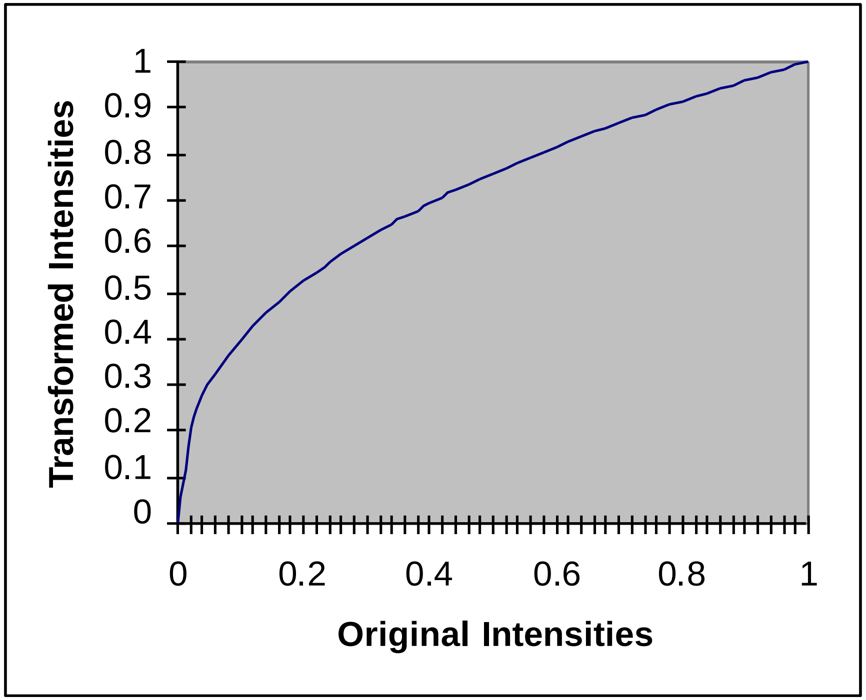
Power Law Example (cont…)

$$\gamma = 0.6$$



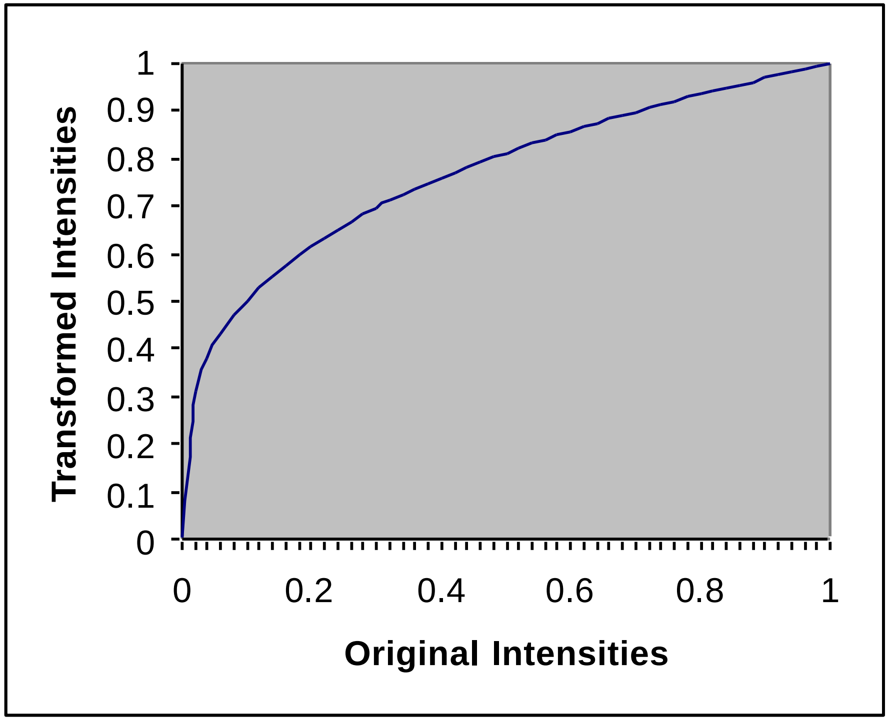
Power Law Example (cont…)

$$\gamma = 0.4$$



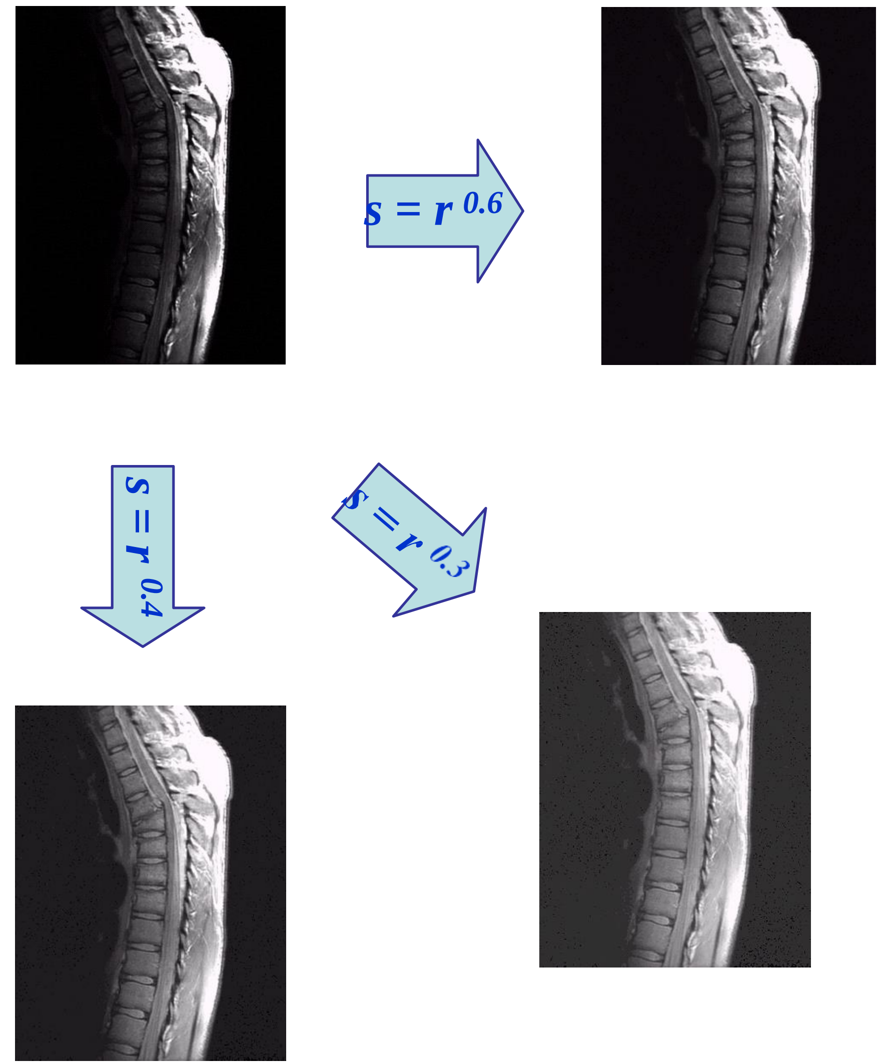
Power Law Example (cont…)

$$\gamma = 0.3$$



Power Law Example (cont…)

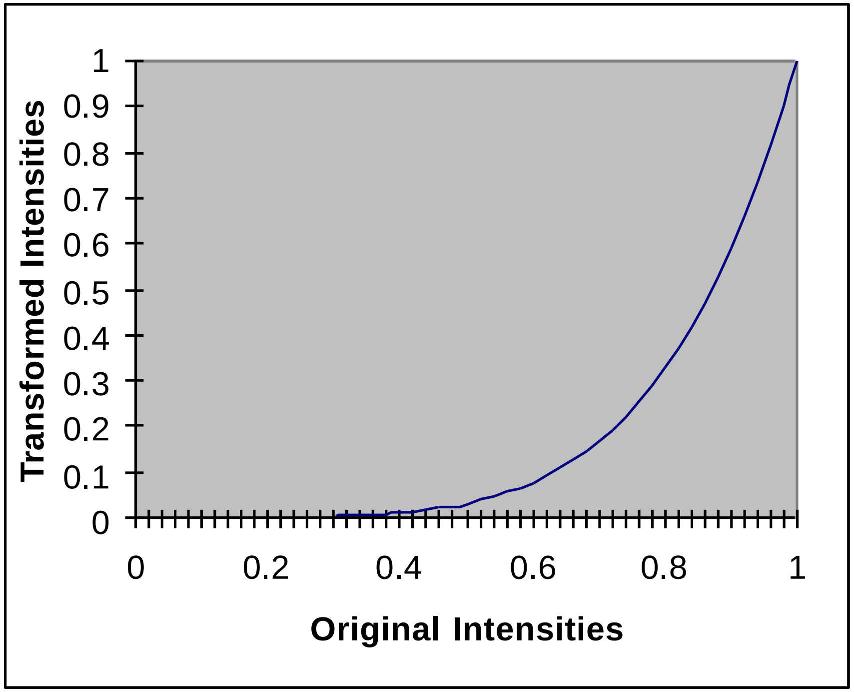
- The images to the right show a magnetic resonance (MR) image of a fractured human spine
- Different curves highlight different detail



Power Law Example

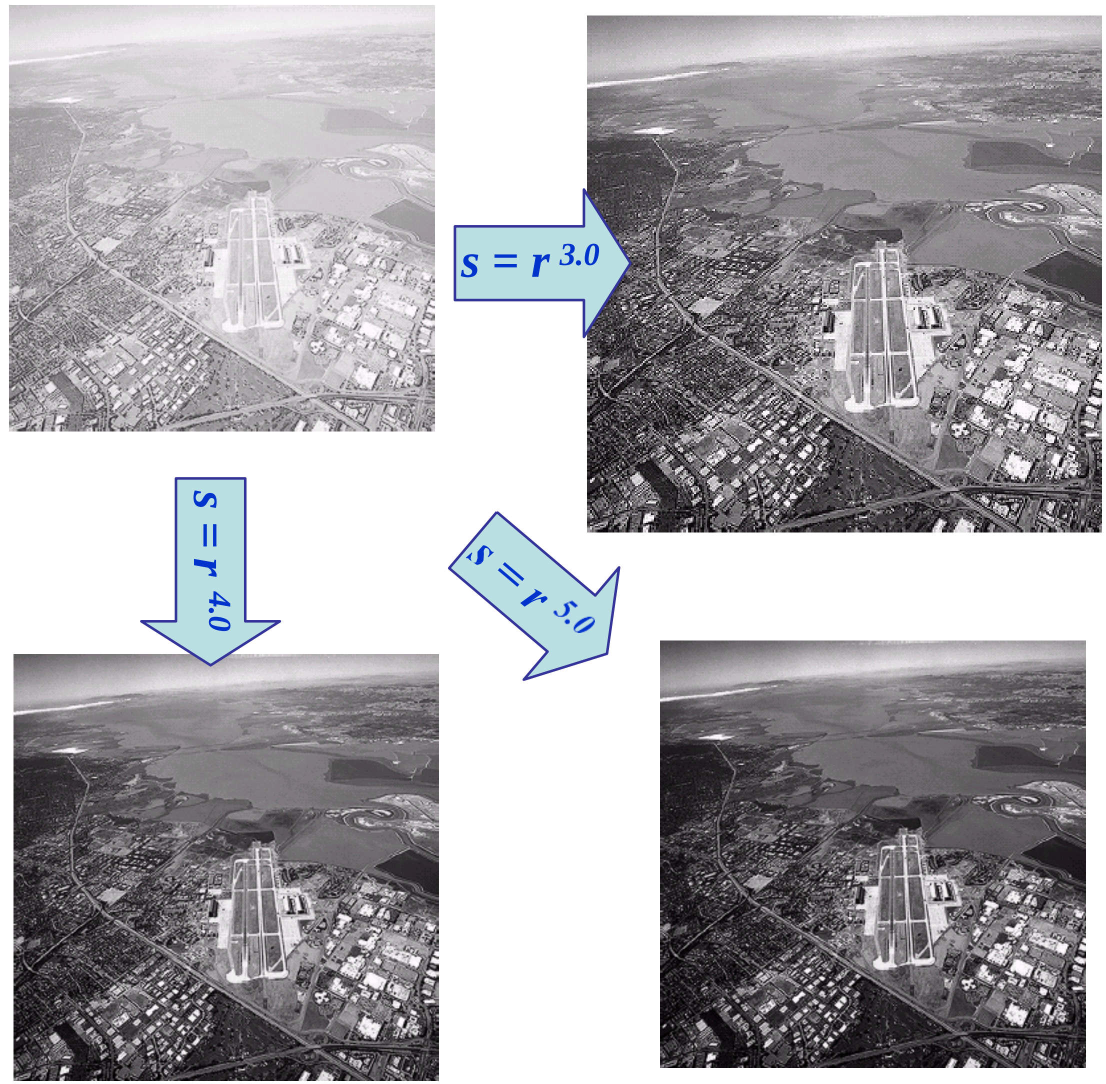
Power Law Example (cont…)

$$\gamma = 5.0$$



Power Law Transformations (cont…)

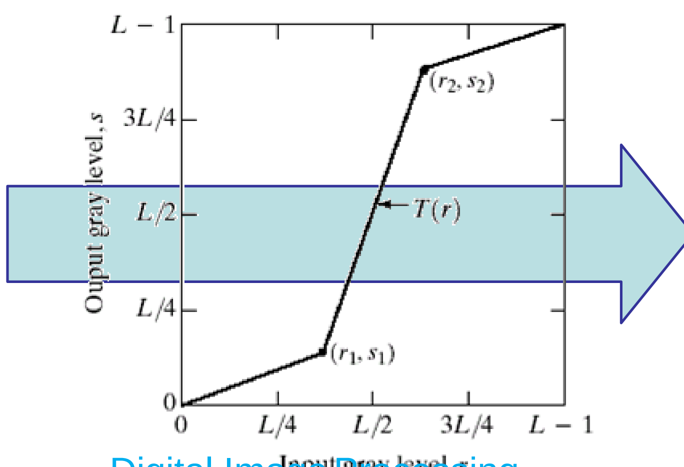
- An aerial photo of a runway is shown
- This time power law transforms are used to darken the image
- Different curves highlight different detail



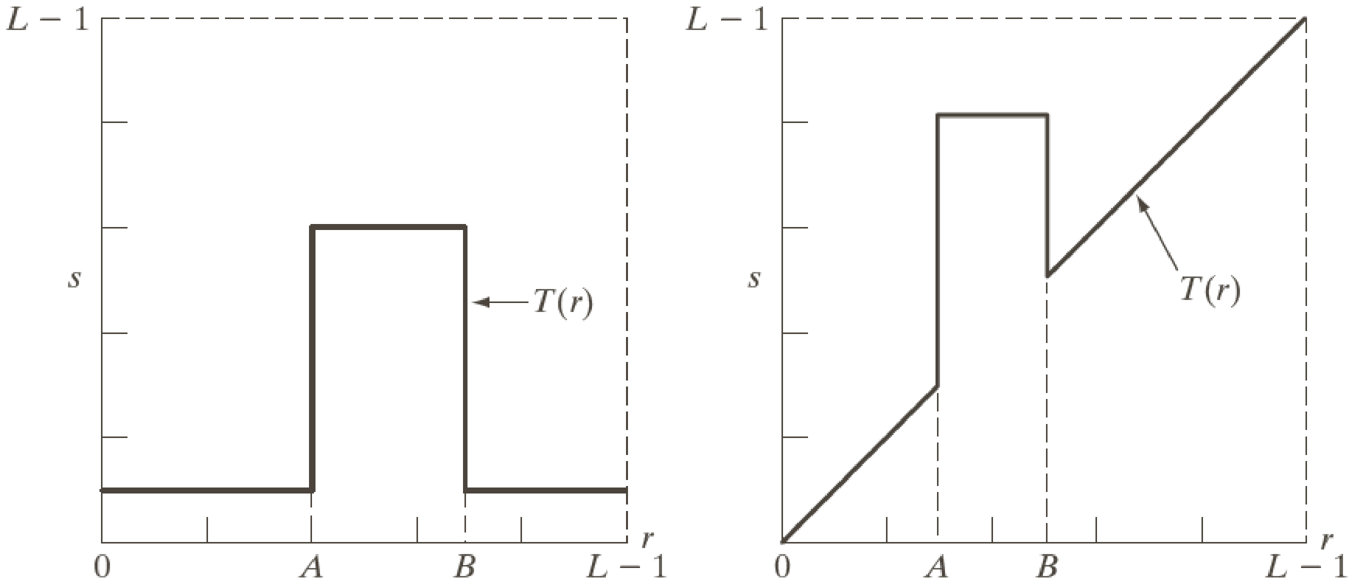
Piecewise Linear Transformation Functions

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…)



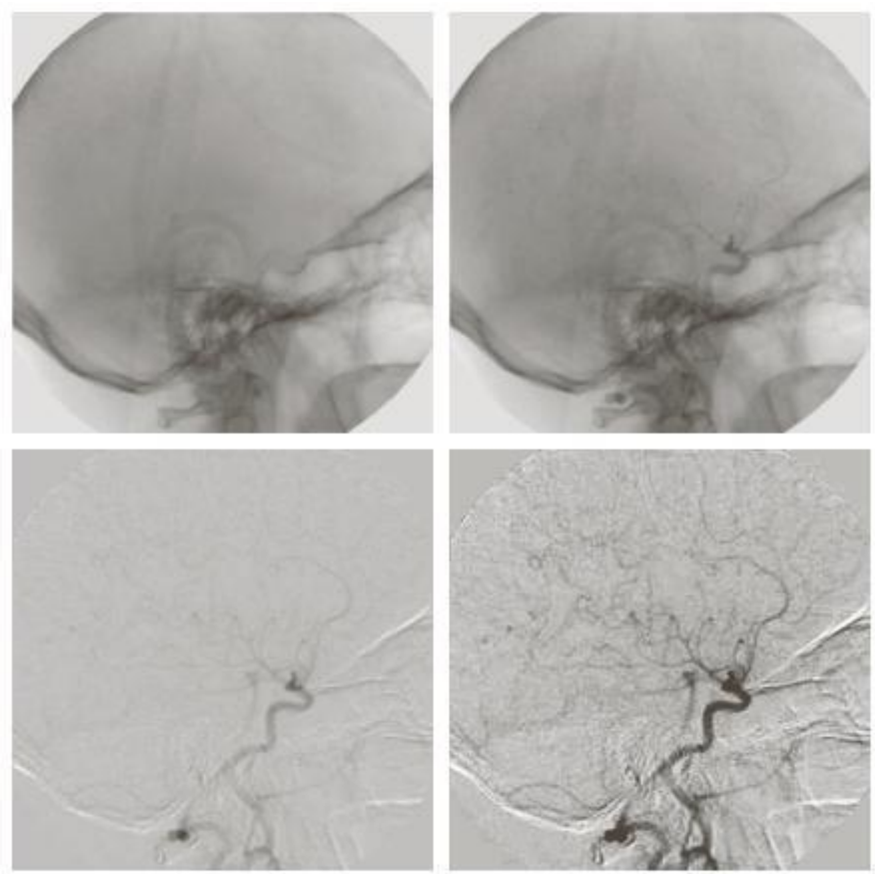
a b

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.

Image subtraction

- Medical application.

iodine medium injected into the bloodstream



a b
c d

FIGURE 2.28 Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of The Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

$$g(x, y) = f(x, y) - h(x, y)$$

Image subtraction

a b c

FIGURE 2.27 (a) Infrared image of the Washington, D.C. area. (b) Image obtained by setting to zero the least significant bit of every pixel in (a). (c) Difference of the two images, scaled to the range [0, 255] for clarity.

- Arithmetic operation can produce pixel values outside of the range [0 – 255].
- You should convert values back to the range [0 – 255] to ensure that the image is displayed properly.

Logical Operations

- Two images of the same size can be combined using operations of addition, subtraction, multiplication, division, logical AND, OR, XOR and NOT.
- Such operations are done on pairs of their corresponding pixels.
- Often only one of the images is a real picture while the other is a machine generated mask. The mask often is a binary image consisting only of pixel values 0 and 1.

Logical Operations

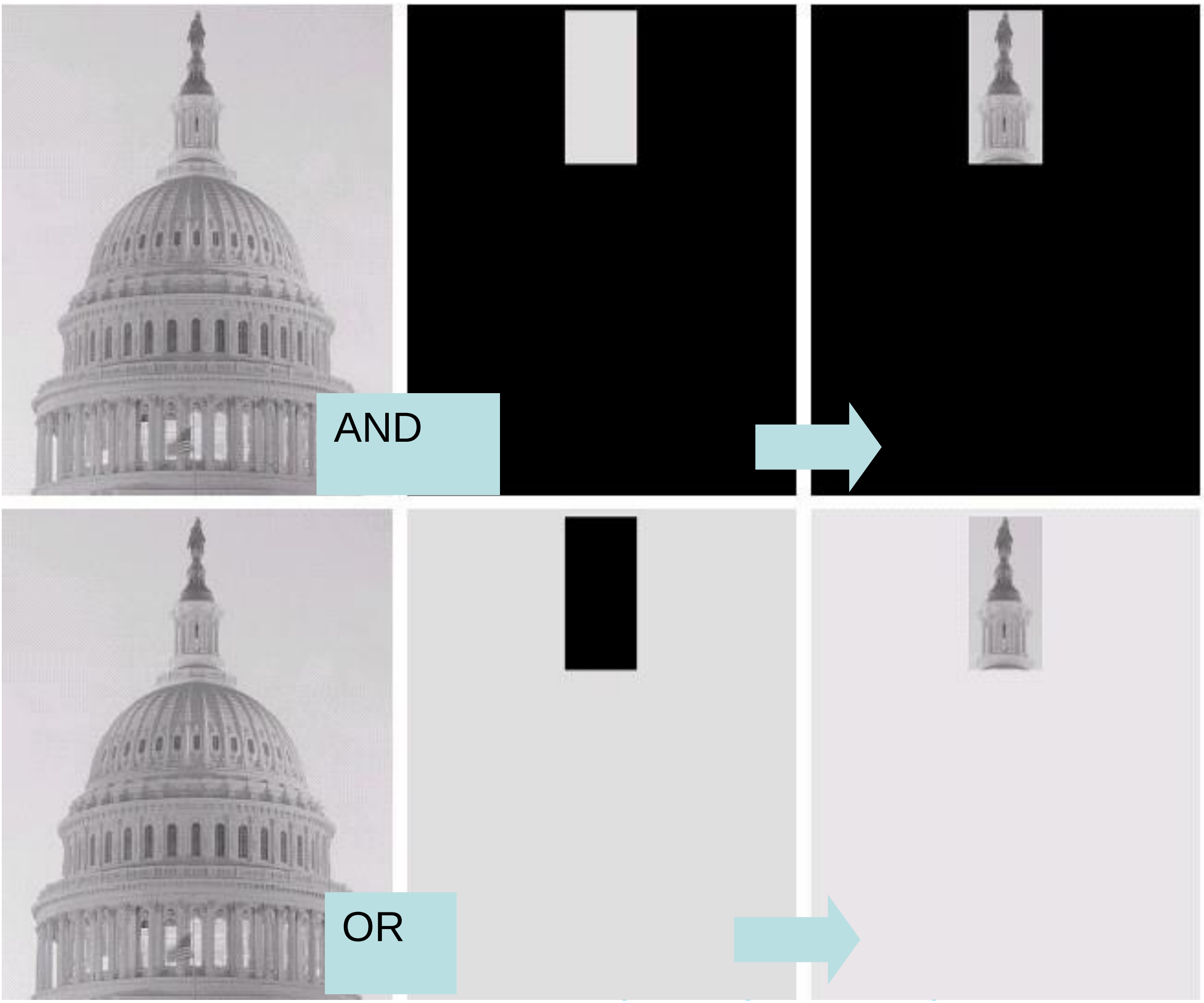


FIGURE 3.27 (a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask. (f) Result of operation OR on images (d) and (e).

Logical Operations

FIGURE 2.30 (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

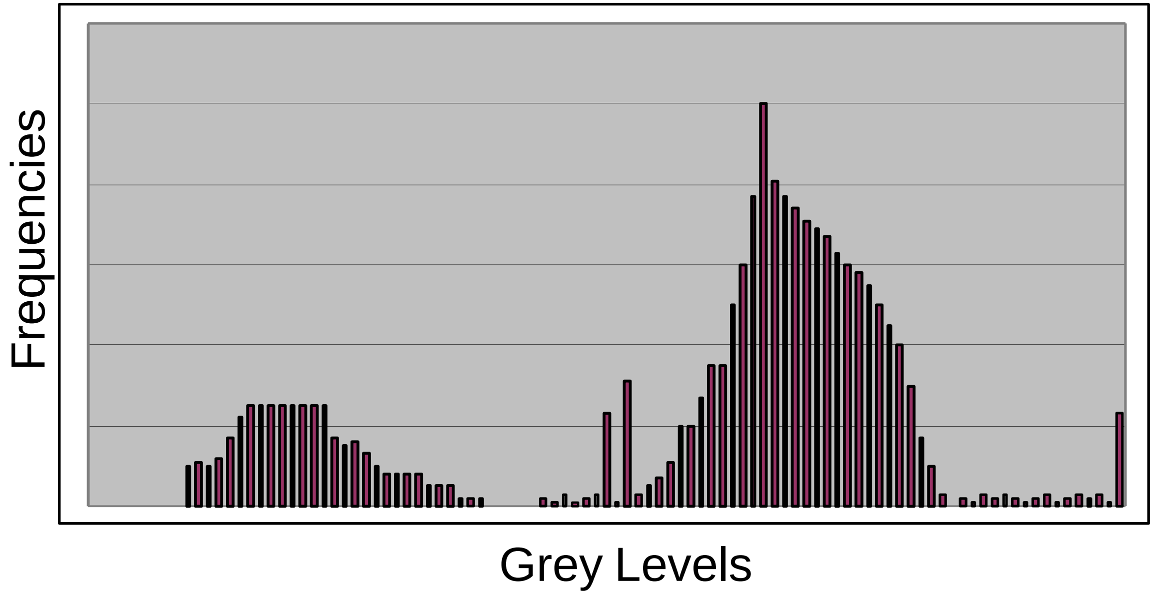
The AND operator is usually used to mask out part of an image.

Intensity Transformations (Histogram Processing)

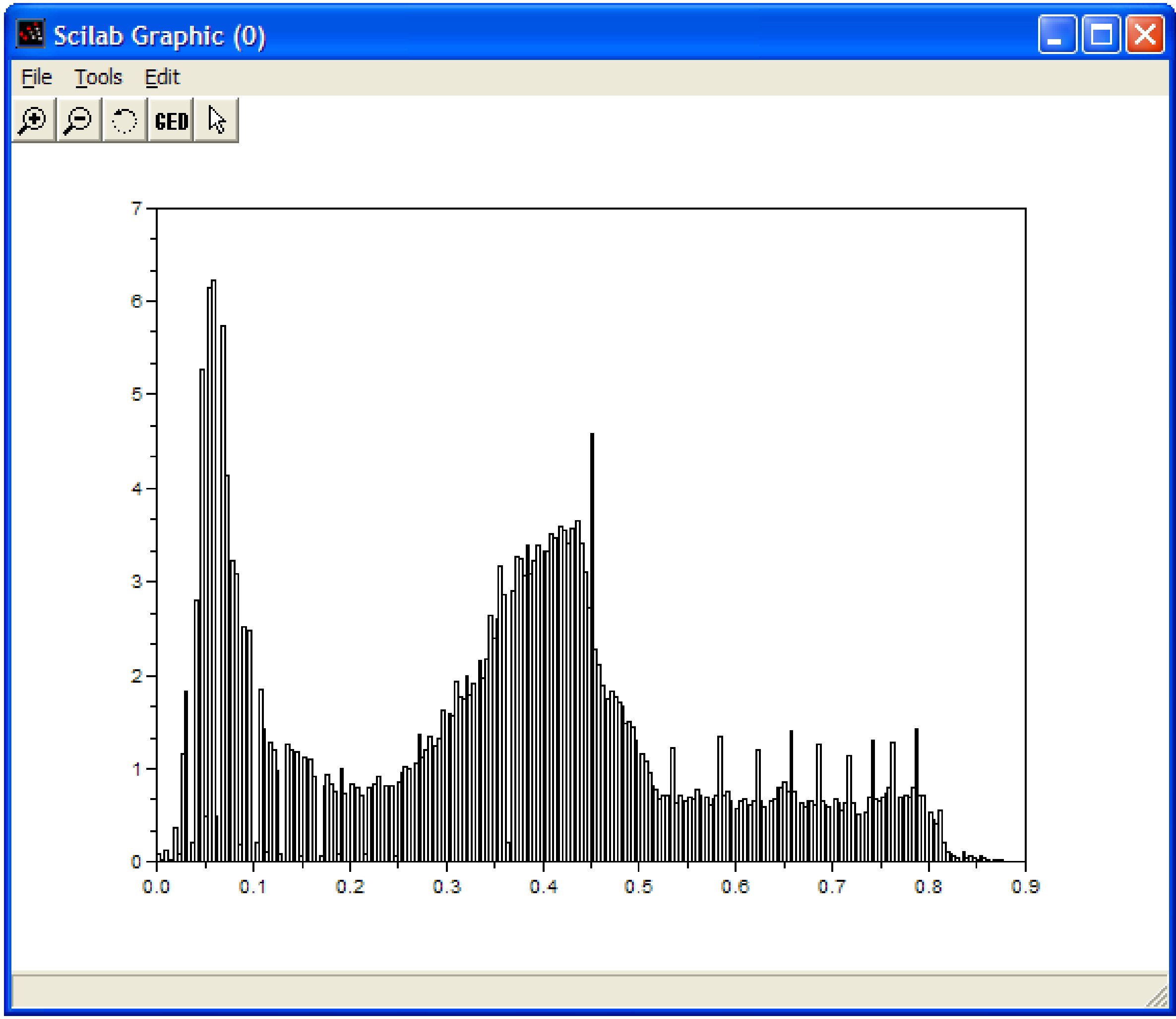
Image Histograms

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

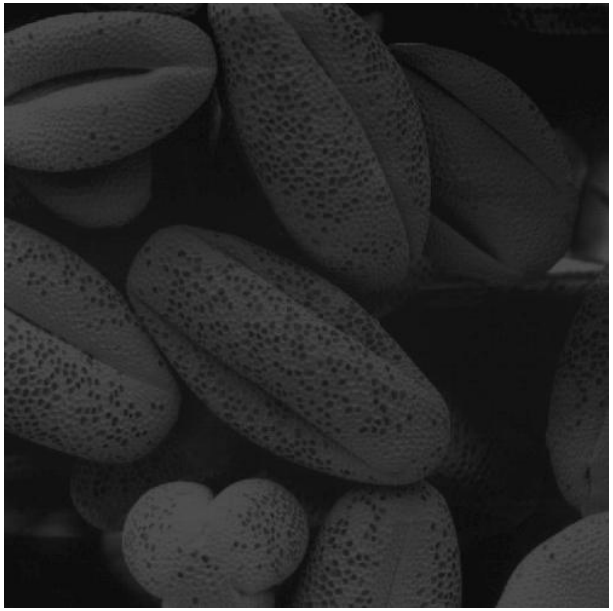
Massively useful in image processing, especially in segmentation

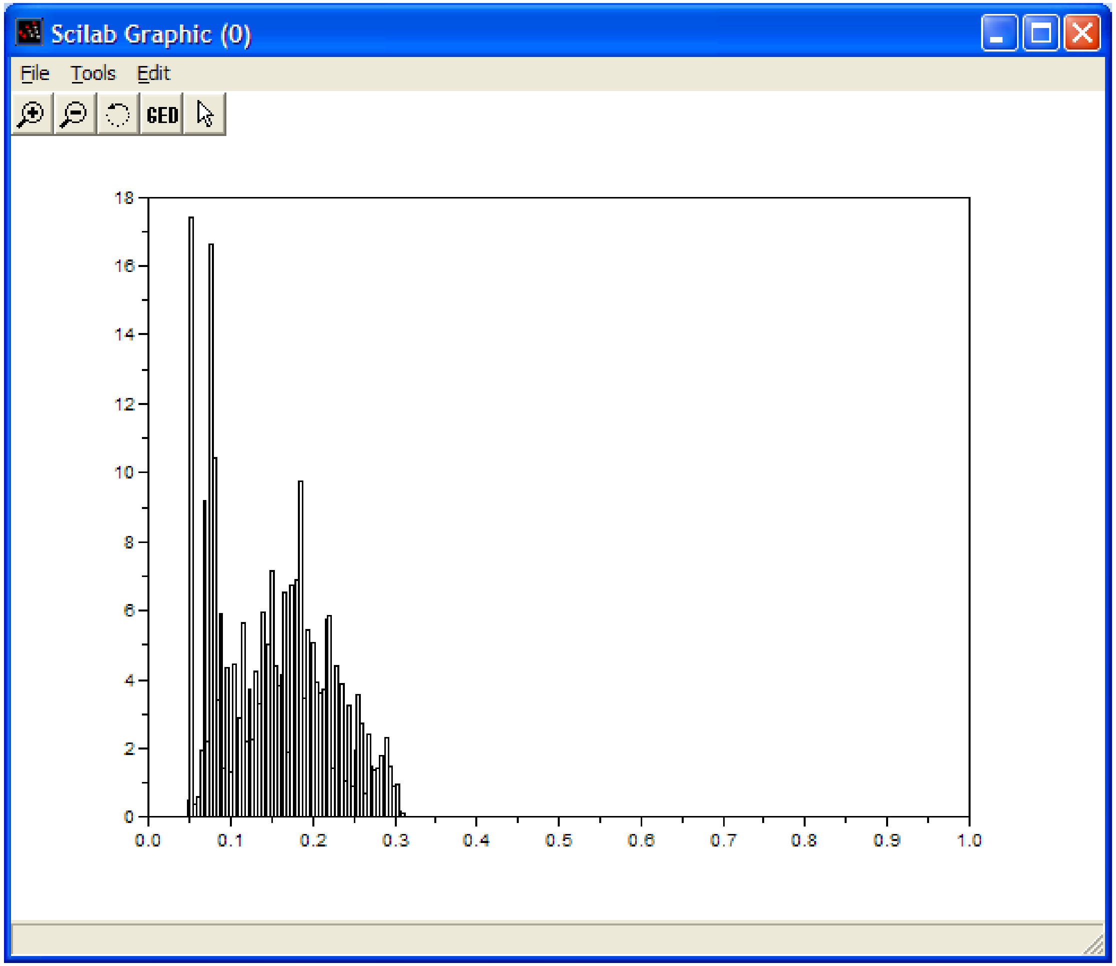


Histogram Examples (cont…)

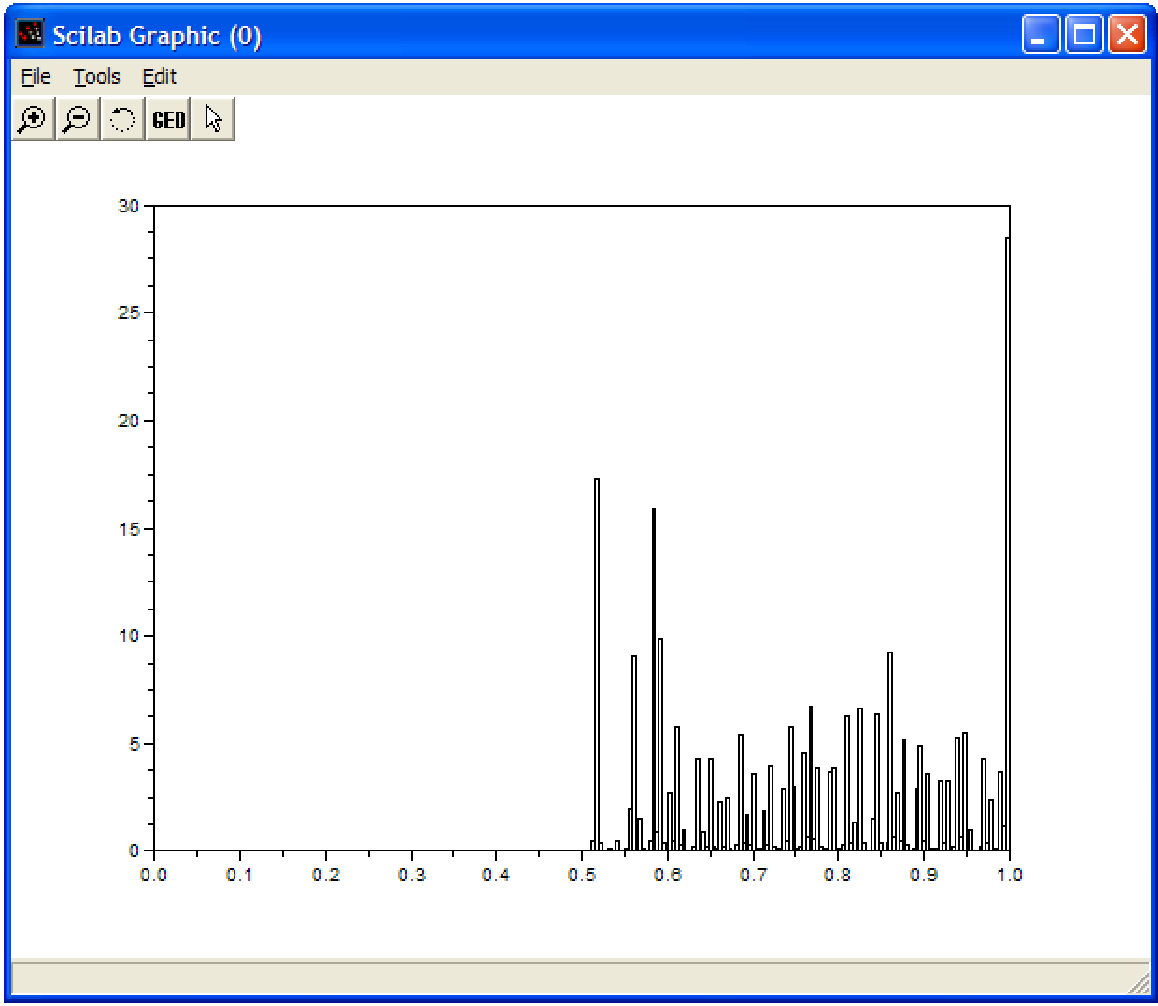


Histogram Examples (cont…)

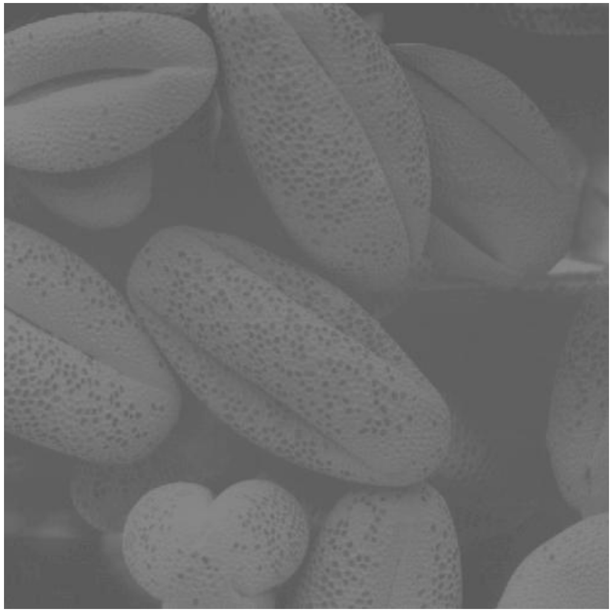


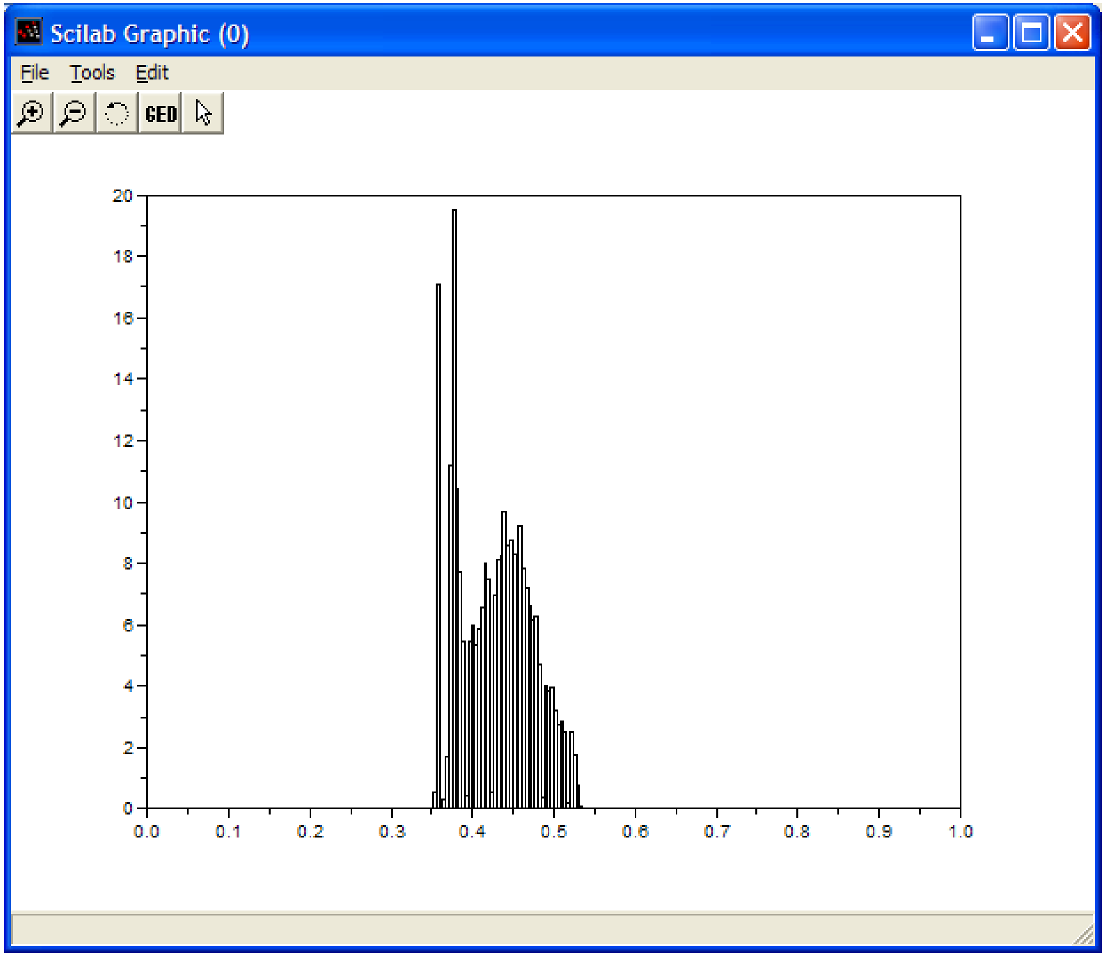


Histogram Examples (cont…)

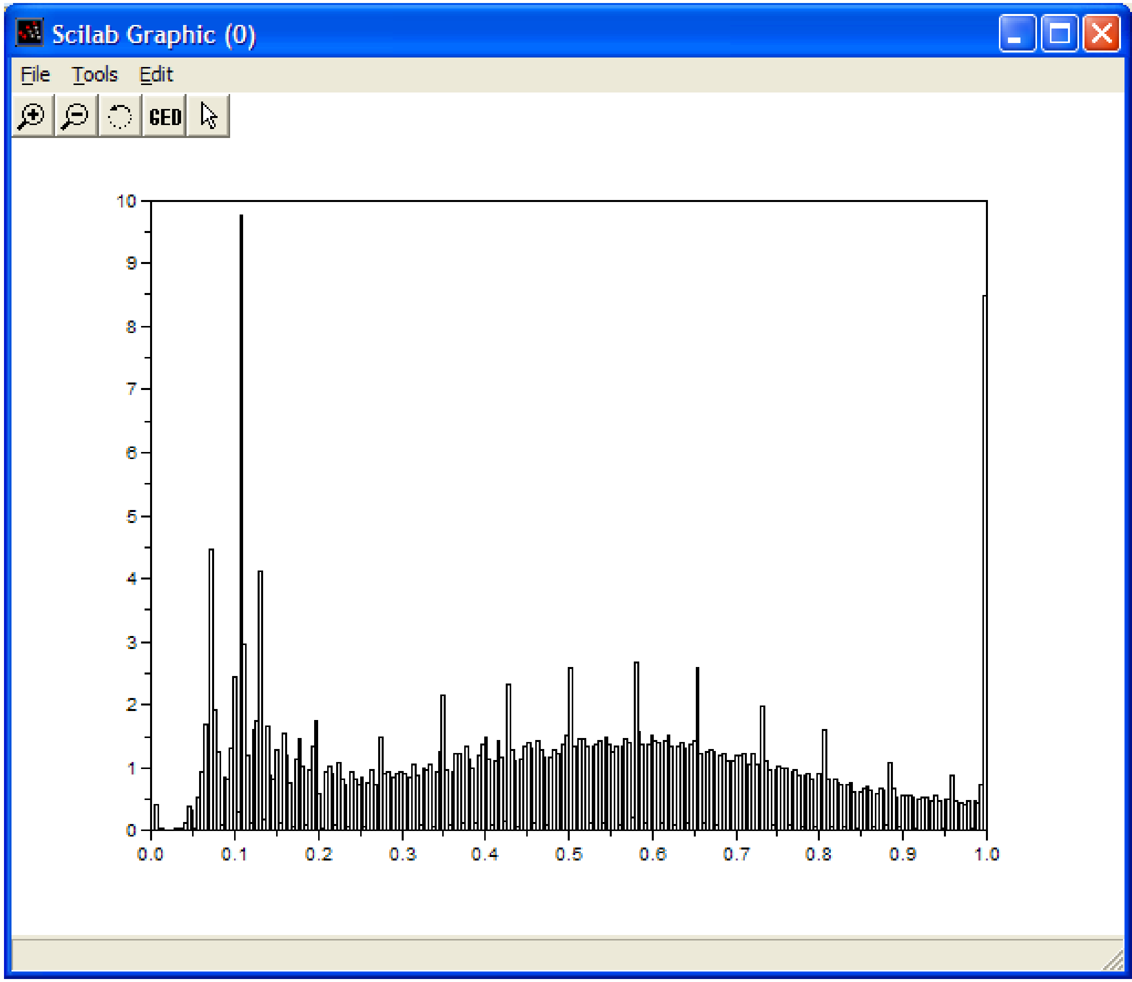


Histogram Examples (cont…)



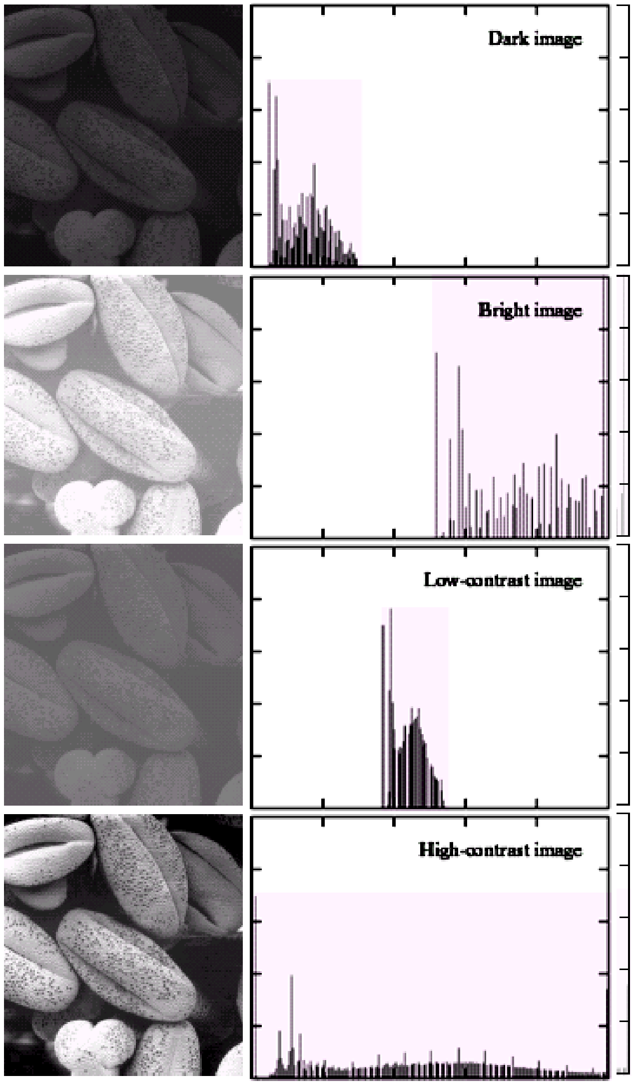


Histogram Examples (cont…)



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



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

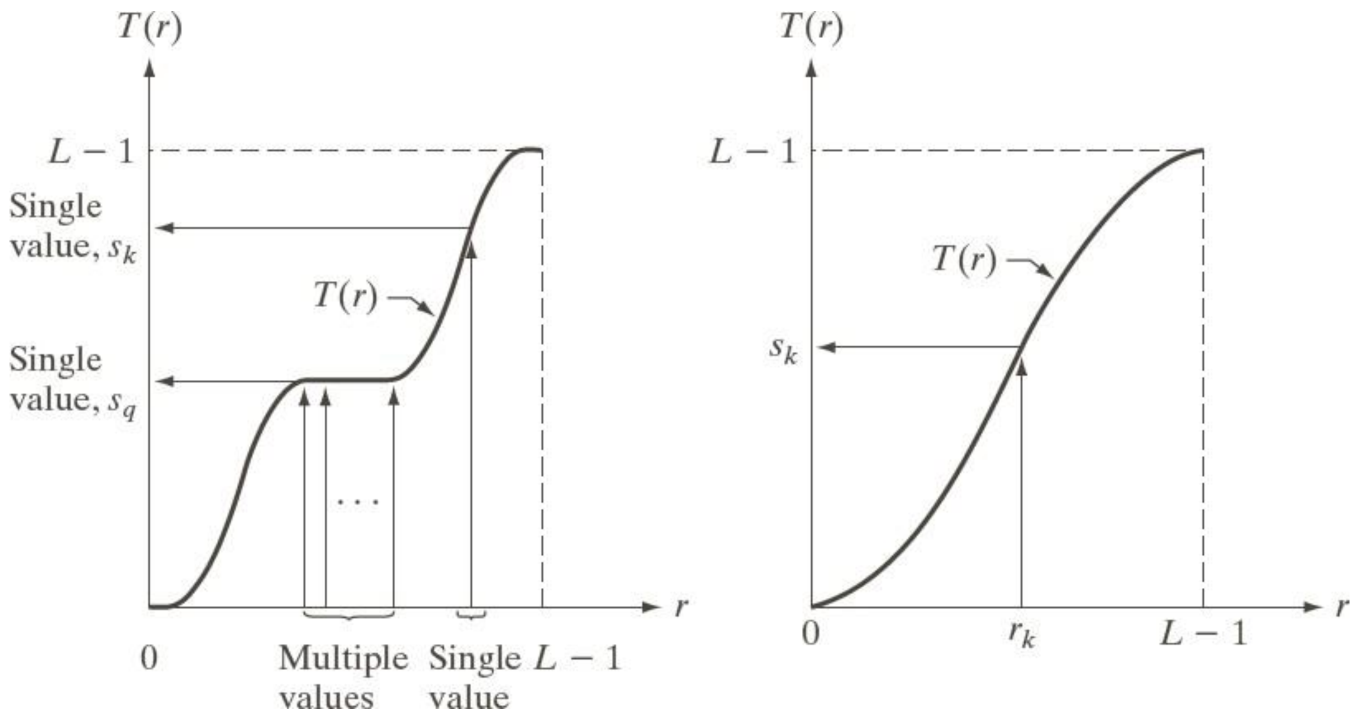
- Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images.
- At first, the continuous case will be studied:
  - r is the intensity of the image in $[0, L\text{-}1]$.
  - we focus on transformations $s=T(r)$:
    - $T(r)$ is strictly monotonically increasing.
    - $T(r)$ must satisfy:

$$0 \le T(r) \le L-1, \ \ for \ \ 0 \le r \le L-1$$

Histogram Equalisation (cont…)

- The condition for $T(r)$ to be monotonically increasing guarantees that ordering of the output intensity values will follow the ordering of the input intensity values (avoids reversal of intensities).
- If $T(r)$ is strictly monotonically increasing then the mapping from s back to r will be 1-1.
- The second condition ($T(r)$ in $[0,1]$) guarantees that the range of the output will be the same as the range of the input.

Histogram Equalisation (cont...)



a) We cannot perform inverse mapping (from s to r).
b) Inverse mapping is possible.

Histogram Equalisation (cont...)

- We can view intensities r and s as random variables and their histograms as probability density functions (pdf) $p_r(r)$ and $p_s(s)$.
- Fundamental result from probability theory:
  - If $p_r(r)$ and $T(r)$ are known and $T(r)$ is continuous and differentiable, then

$$p_s(s) = p_r(r)\frac{1}{\left|\frac{ds}{dr}\right|} = p_r(r)\left|\frac{dr}{ds}\right|$$

Histogram Equalisation (cont…)

- The pdf of the output is determined by the pdf of the input and the transformation.
- This means that we can determine the histogram of the output image.
- A transformation of particular importance in image processing is the cumulative distribution function (CDF) of a random variable:

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$

Histogram Equalisation (cont...)

- It satisfies the first condition as the area under the curve increases as r increases.
- It satisfies the second condition as for $r=L$-1 we have $s=L$-1.
- To find $p_s(s)$ we have to compute

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\frac{d}{dr}\int_0^r p_r(w)\,dw = (L-1)p_r(r)$$

Histogram Equalisation (cont...)

Substituting this result:

$$\frac{ds}{dr} = (L-1)p_r(r)$$

to

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

Uniform pdf

yields

$$p_s(s) = p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| = \frac{1}{L-1}, \quad 0 \le s \le L-1$$

Histogram Equalisation (cont...)

The formula for histogram equalisation in the discrete case is given

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$$

where

- r_k: input intensity
- s_k: processed intensity
- n_j: the frequency of intensity j
- MN: the number of image pixels.

Histogram Equalisation (cont...) Example

A 3-bit 64x64 image has the following intensities:

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$

Applying histogram equalization:

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7\,p_r(r_0) = 1.33$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7\,p_r(r_0) + 7\,p_r(r_1) = 3.08$$

Histogram Equalisation (cont…) Example

Rounding to the nearest integer:

$$s_0 = 1.33 \rightarrow 1 \quad s_1 = 3.08 \rightarrow 3 \quad s_2 = 4.55 \rightarrow 5 \quad s_3 = 5.67 \rightarrow 6$$

$$s_4 = 6.23 \rightarrow 6 \quad s_5 = 6.65 \rightarrow 7 \quad s_6 = 6.86 \rightarrow 7 \quad s_7 = 7.00 \rightarrow 7$$

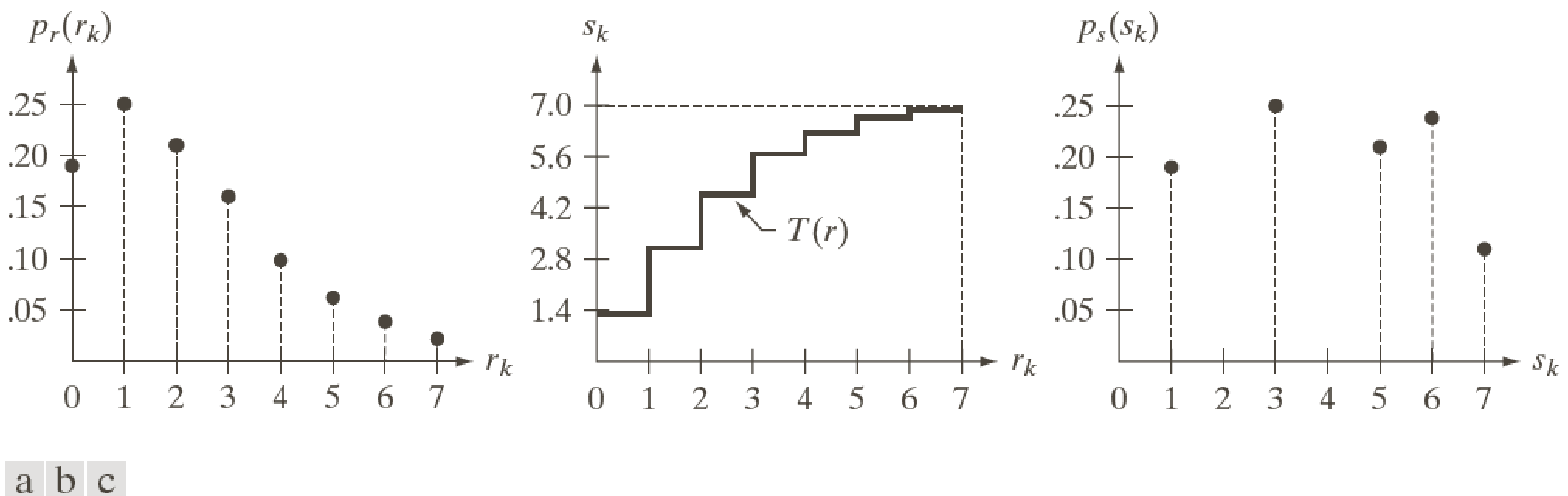


FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

Histogram Equalization (cont…) Example

Notice that due to discretization, the resulting histogram will rarely be perfectly flat. However, it will be extended.

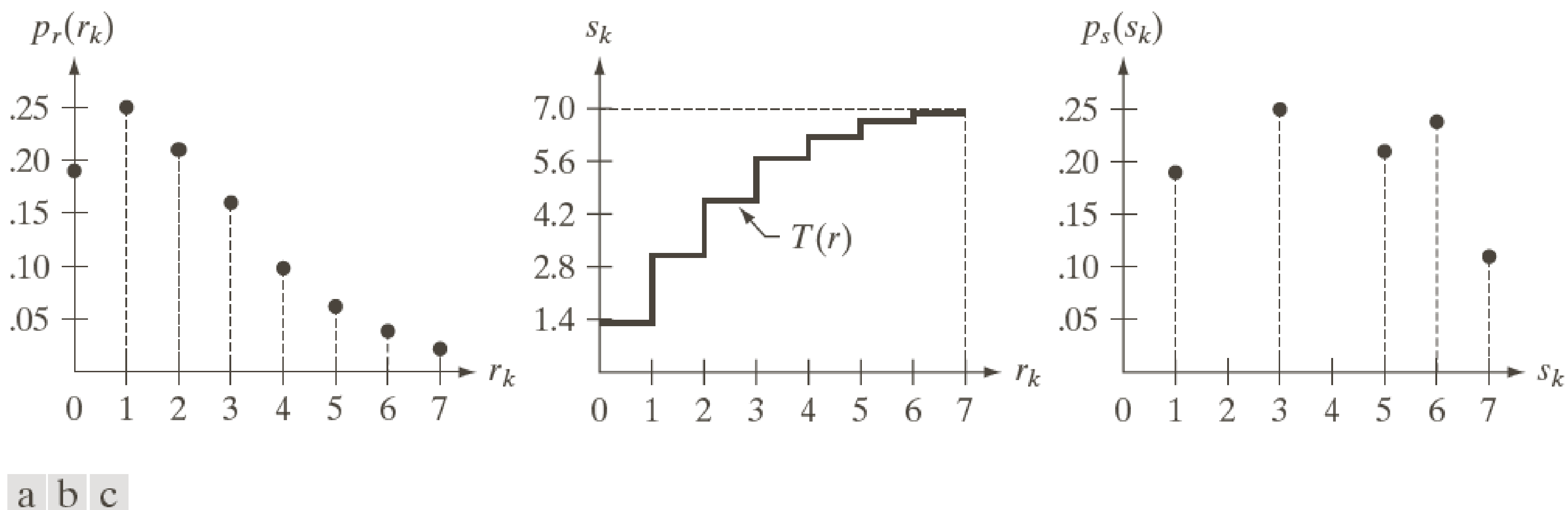
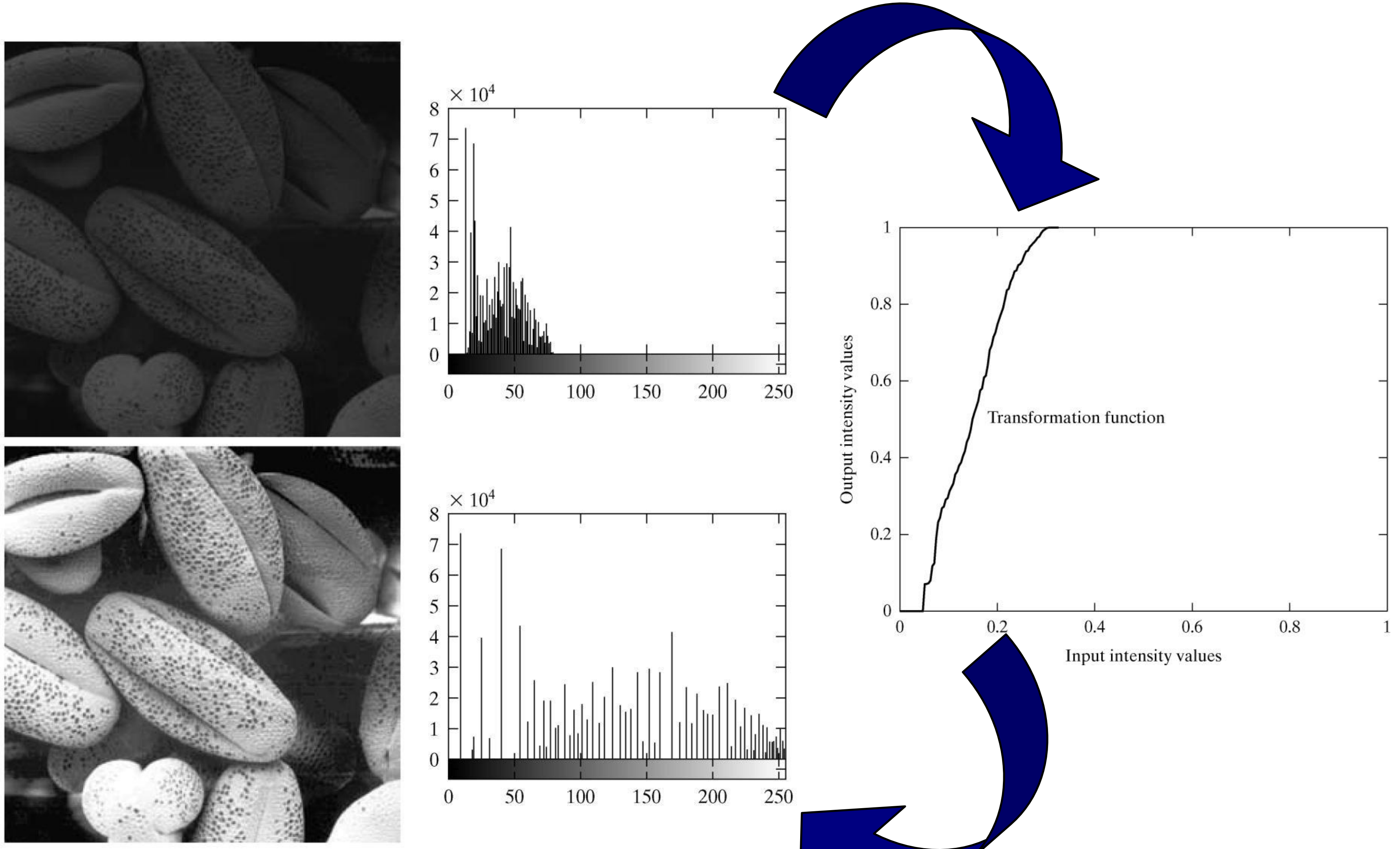
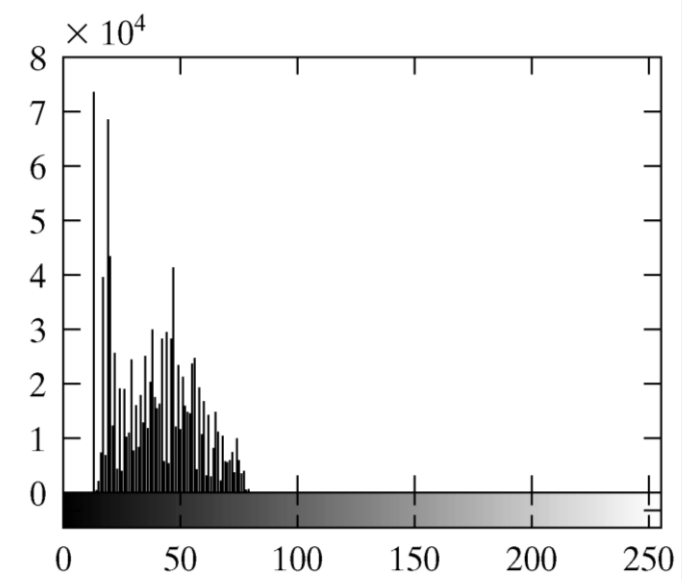


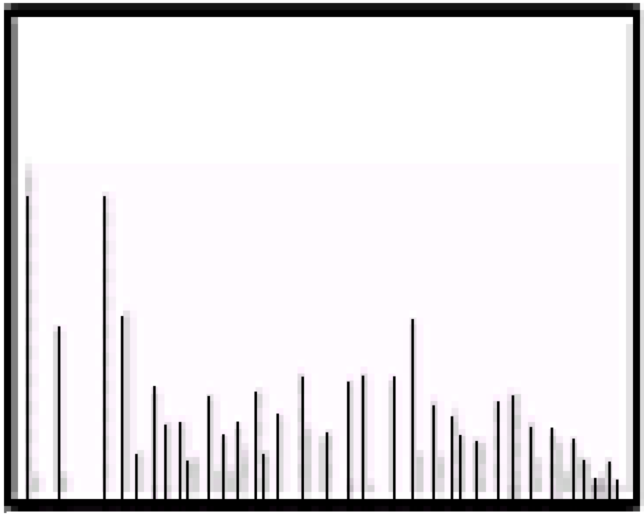
FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

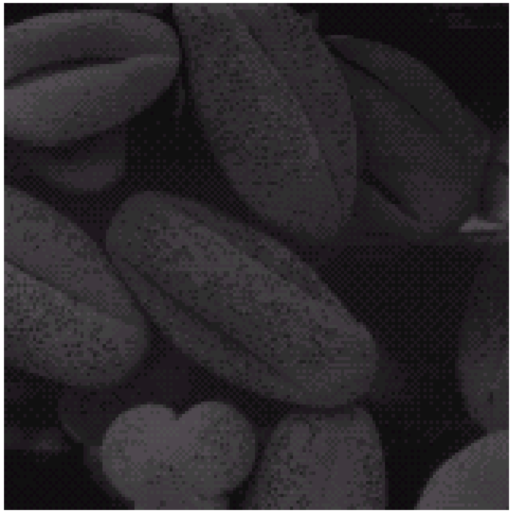
Equalisation Transformation Function



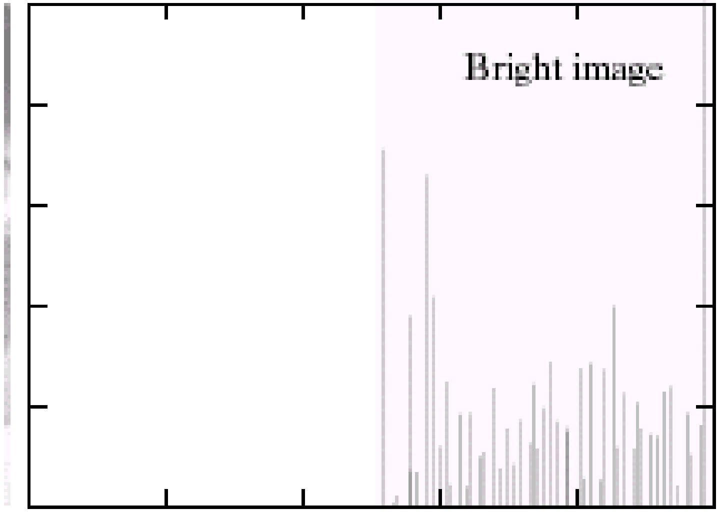
Equalisation Examples

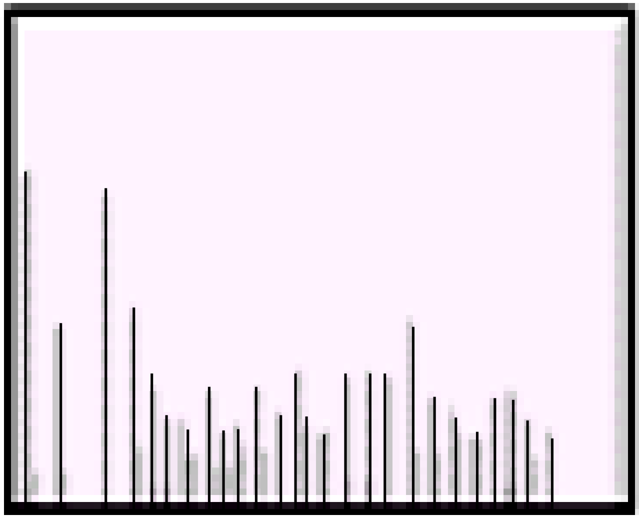






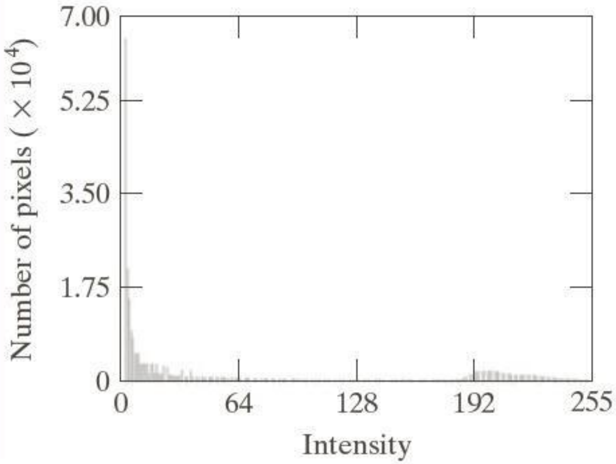
Equalisation Examples





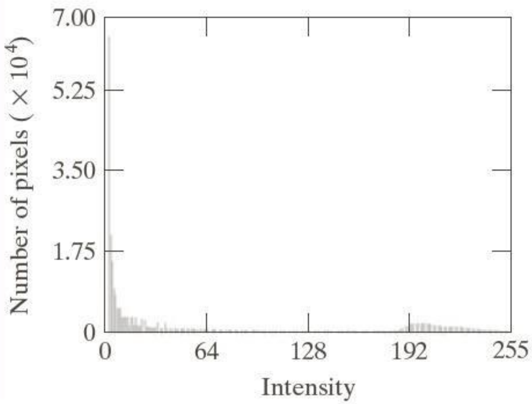
Histogram Specification

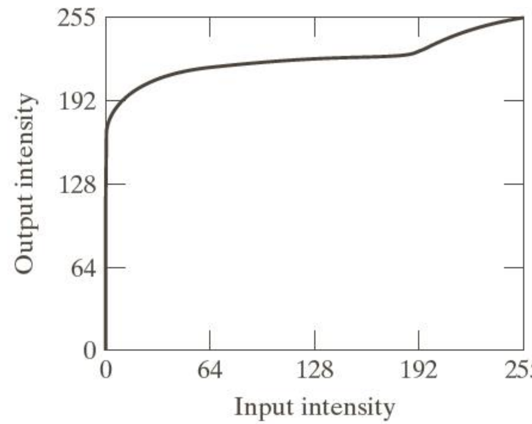
- Histogram equalization does not always provide the desirable results.

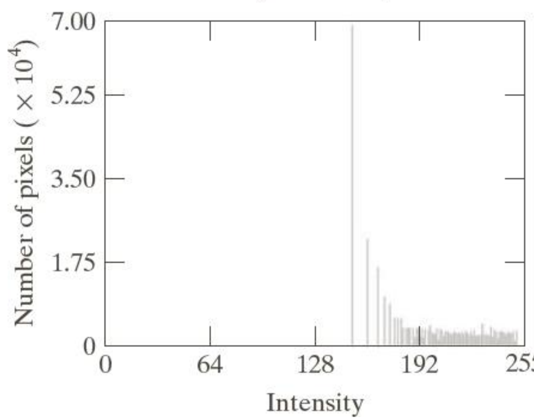


- Image of Phobos (Mars moon) and its histogram.
- Many values near zero in the initial histogram

Histogram Specification (cont...)



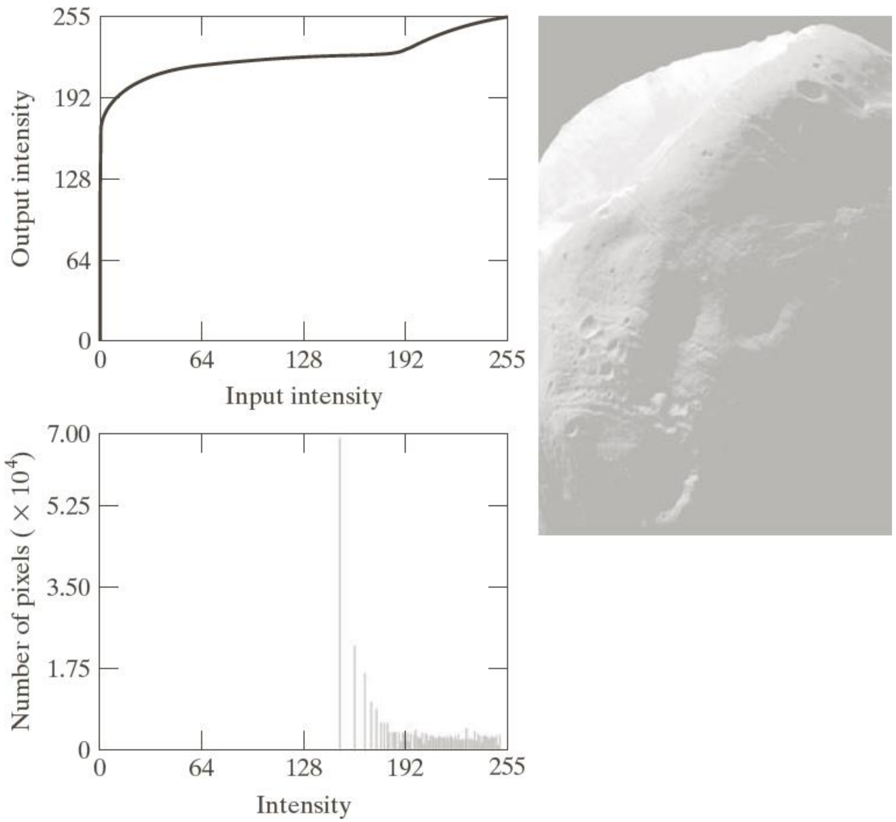




Original image

Histogram equalization

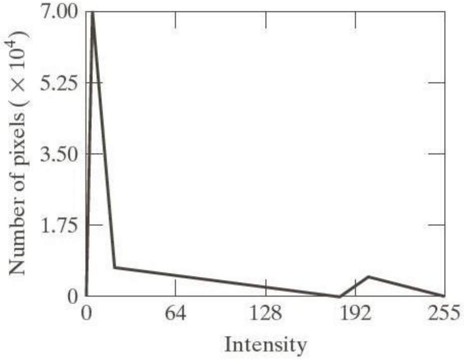
Histogram Specification (cont...)



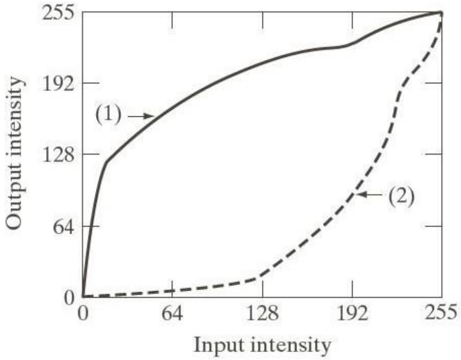
Histogram equalization

Histogram Specification (cont…)

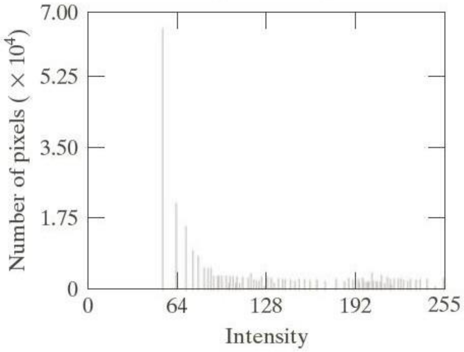
Specified histogram



Transformation function and its inverse



Resulting histogram



Local Histogram Processing

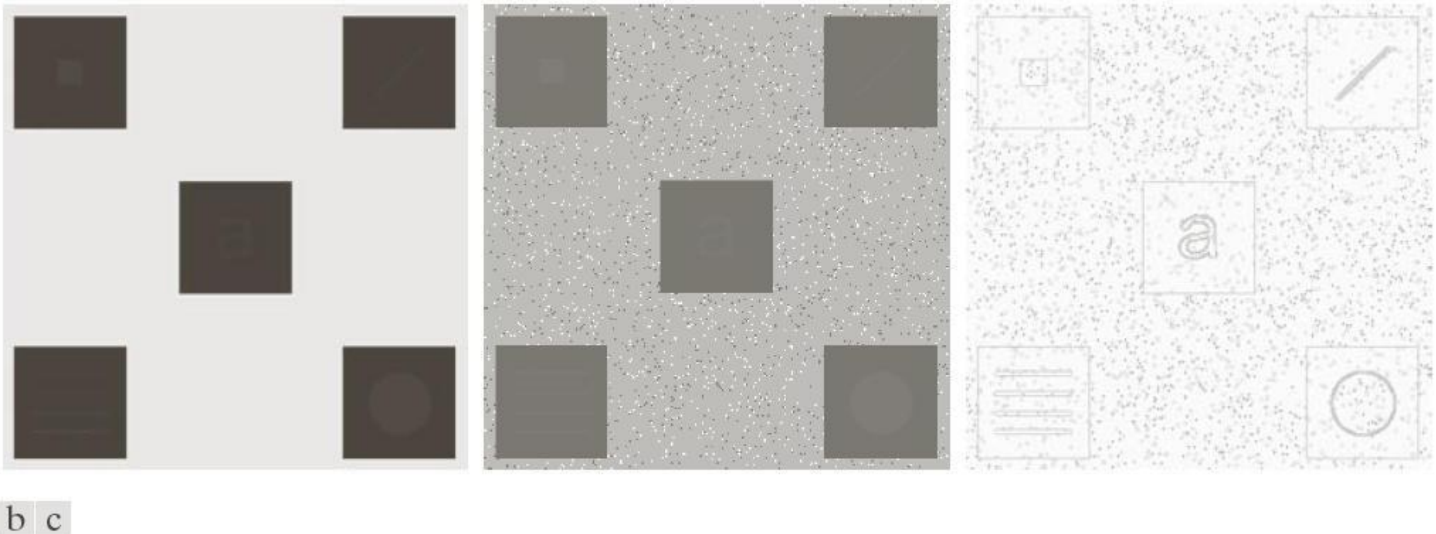


FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3 × 3.

- Image in (a) is slightly noisy but the noise is imperceptible.
- HE enhances the noise in smooth regions (b).
- Local HE reveals structures having values close to the values of the squares and small sizes to influence HE (c).

Histogram based statistics

- r denote a discrete random variable representing gray-levels in [0,L-1],

- $p(r_i)$ denote the normalized histogram component corresponding to the ith value of r, then the nth moment of r about its mean is defined as

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n \, p(r_i)$$

where m is the mean value of r

$$m = \sum_{i=0}^{L-1} r_i \, p(r_i)$$

- For example, the second moment (also the variance of r) is

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 \, p(r_i)$$

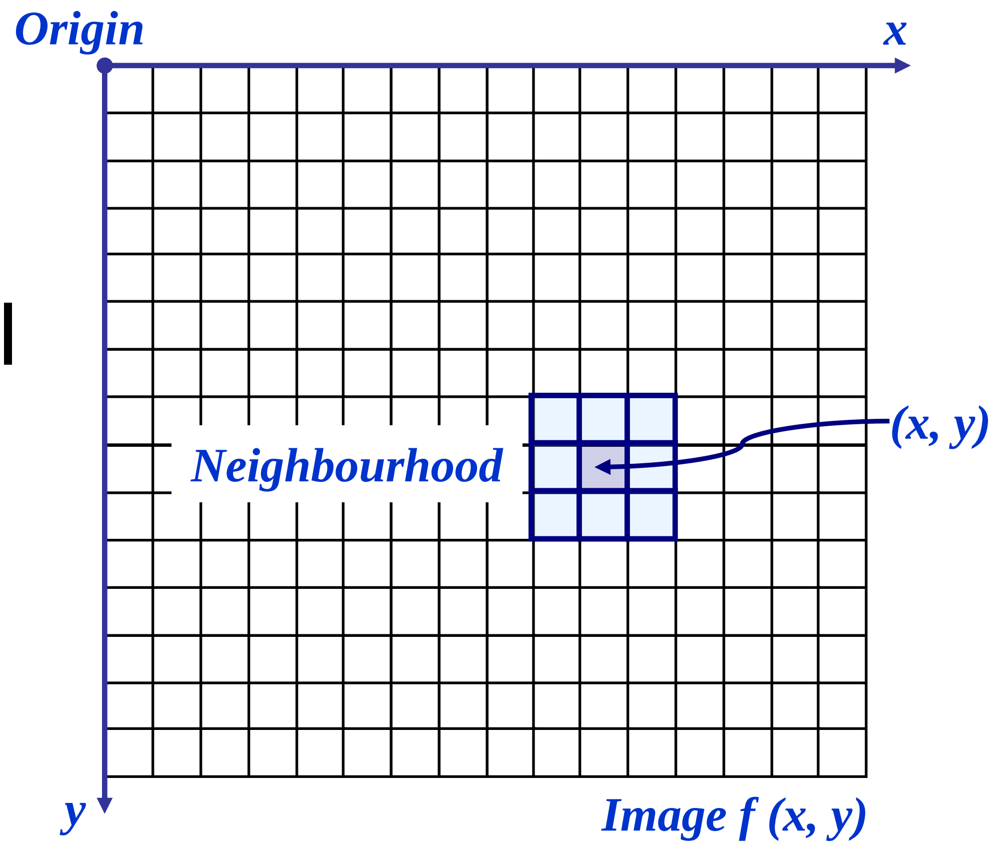
- Two uses of the mean and variance for enhancement purposes:

  - The global mean and variance (global means for the entire image) are useful for adjusting overall contrast and intensity.

  - The mean and standard deviation for a local region are useful for correcting for large-scale changes in intensity and contrast.

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

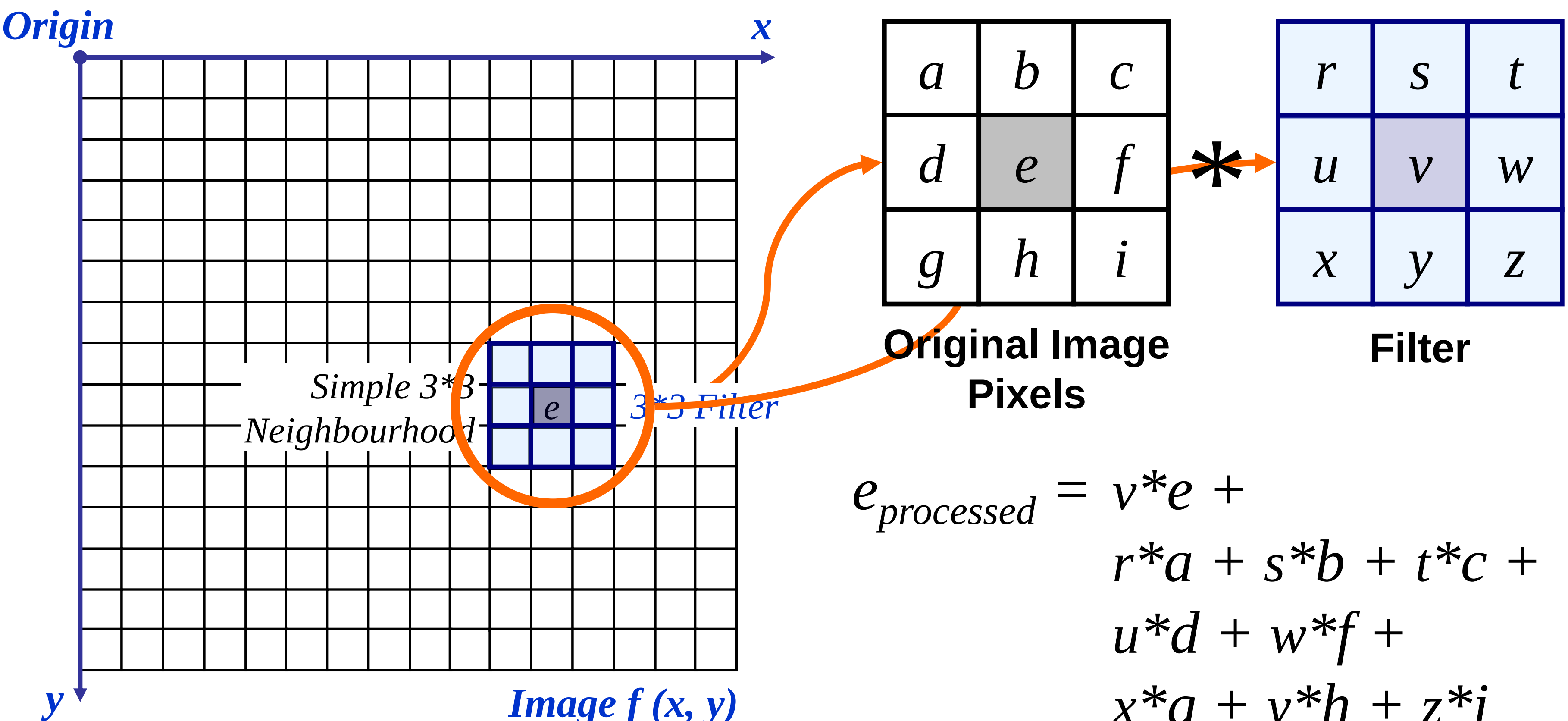
Simple Neighbourhood Operations Example

Original Image

123	127	128	119	115	130
140	145	148	153	167	172
133	154	183	192	194	191
194	199	207	210	198	195
164	170	175	162	173	151

Enhanced Image

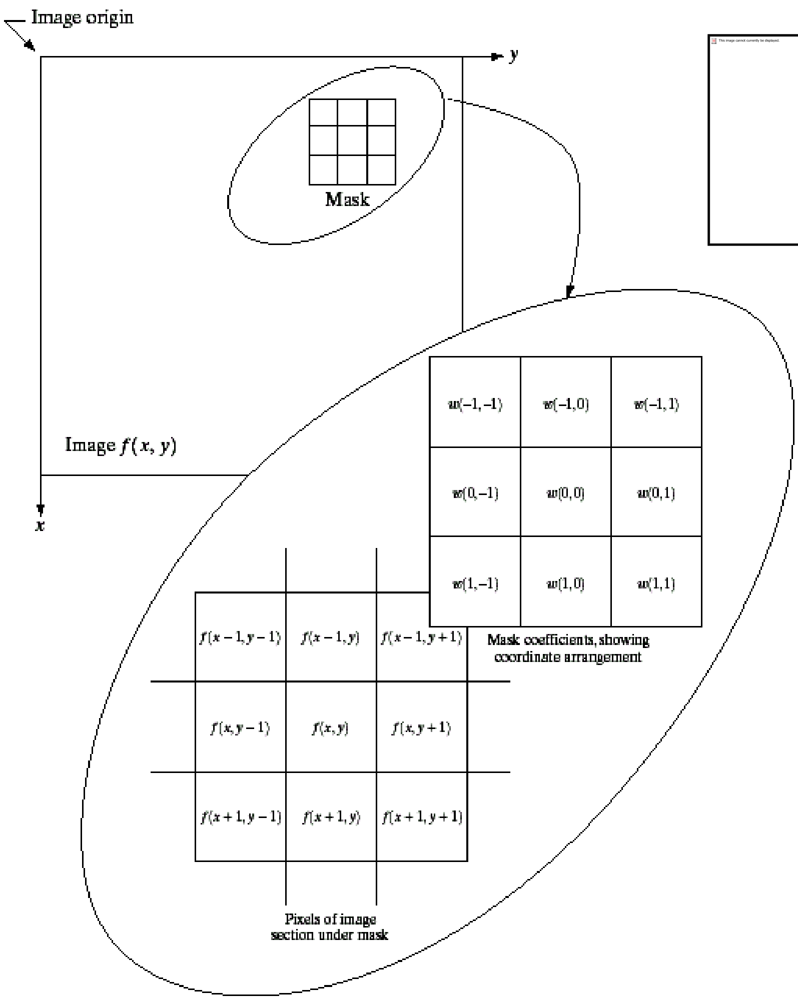
The Spatial Filtering Process



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

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

Spatial Filtering



Filtering can be given in equation form as shown above

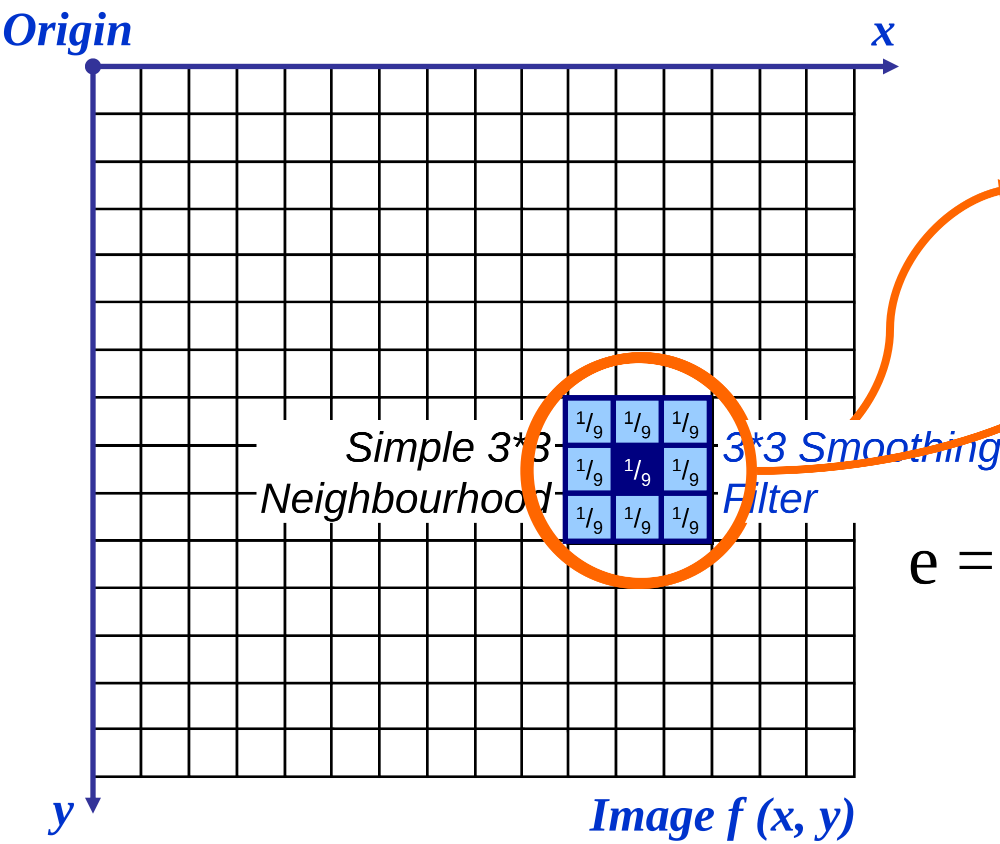
Smoothing Spatial Filters

- Average all of the pixels in a neighbourhood around a central value
- Useful in removing noise from images
- Also useful for highlighting gross detail
- Edge blurring

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

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

$$
\begin{aligned}
e = {} & {}^1/_9*106 + \\
& {}^1/_9*104 + {}^1/_9*100 + {}^1/_9*108 + \\
& {}^1/_9*99 + {}^1/_9*98 + \\
& {}^1/_9*95 + {}^1/_9*90 + {}^1/_9*85 \\
= {} & 98.3333
\end{aligned}
$$

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

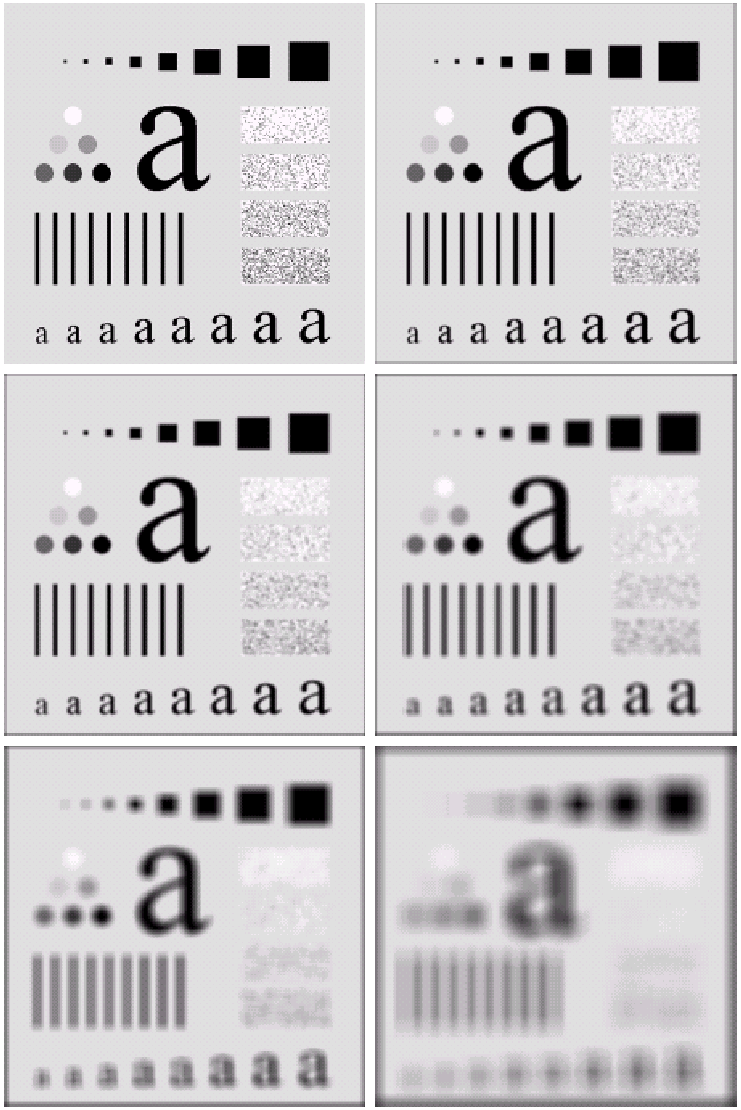
Image Smoothing Example

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.



Weighted Smoothing Filters

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

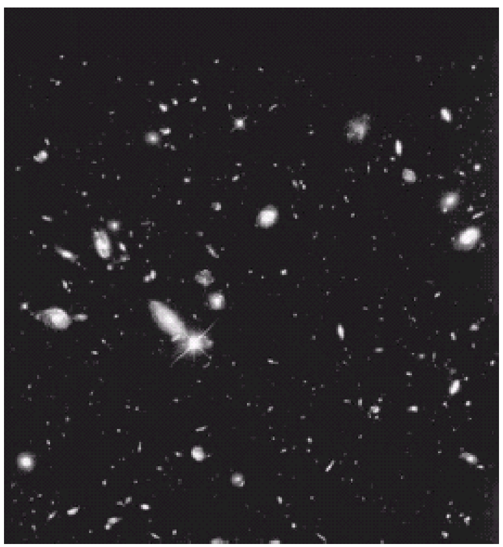
- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

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

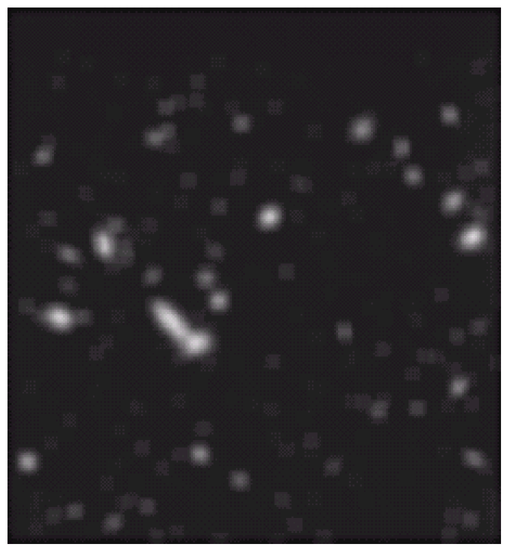
Weighted averaging filter

Another Smoothing Example

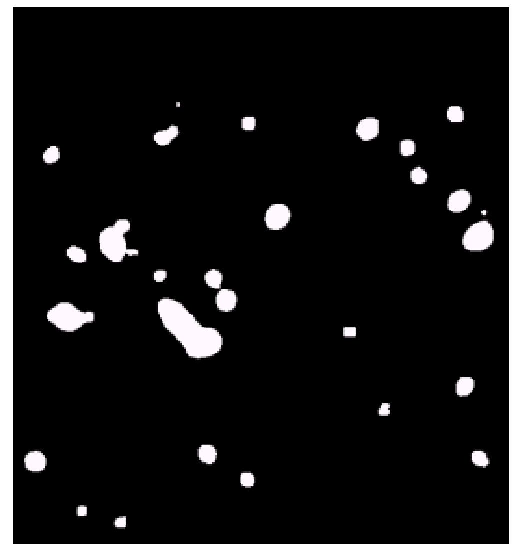
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



Original Image

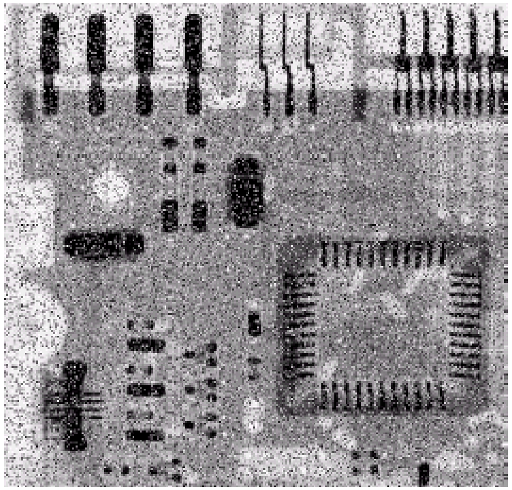


Smoothed Image



Thresholded Image

Averaging Filter Vs. Median Filter Example



Original Image With Noise

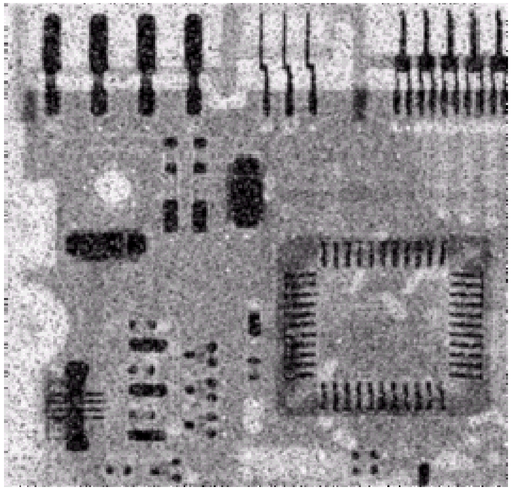


Image After Averaging Filter

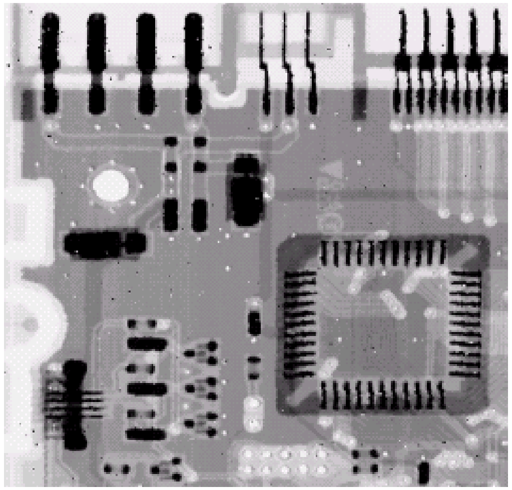


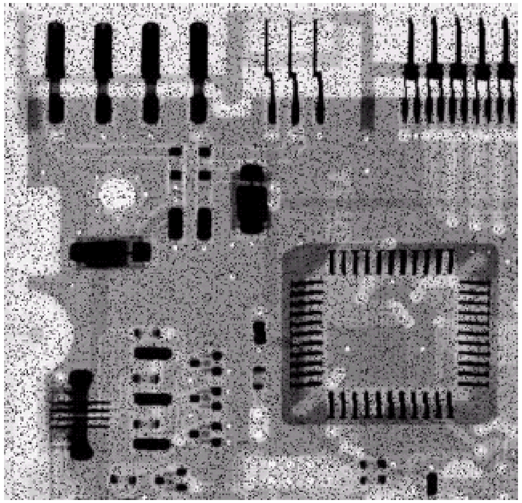
Image After Median Filter

Filtering is often used to remove noise from images

Sometimes a median filter works better than an averaging filter

Noise Removal Examples

Image Corrupted By Pepper Noise



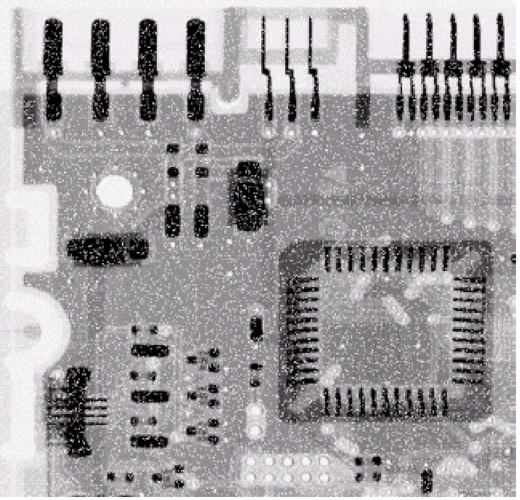
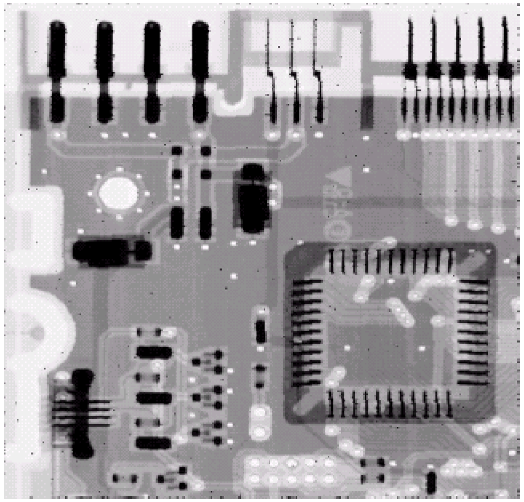
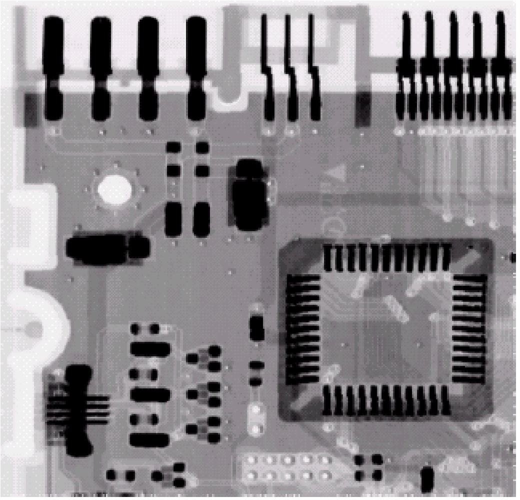


Image Corrupted By Salt Noise

Result Of Filtering Above With A 3*3 Max Filter





Result Of Filtering Above With A 3*3 Min Filter

Spatial smoothing and image approximation

Spatial smoothing may be viewed as a process for estimating the value of a pixel from its neighbours.

What is the value that “best” approximates the intensity of a given pixel given the intensities of its neighbours?

We have to define “best” by establishing a criterion.

Spatial smoothing and image approximation (cont...)

A standard criterion is the the sum of squares differences.

$$E = \sum_{i=1}^{N} \left[x(i) - m \right]^2 \quad \Leftrightarrow m = \arg\min_{m} \left\{ \sum_{i=1}^{N} \left[x(i) - m \right]^2 \right\}$$

$$\frac{\partial E}{\partial m} = 0 \quad \Leftrightarrow -2 \sum_{i=1}^{N} \left(x(i) - m \right) = 0 \quad \Leftrightarrow \sum_{i=1}^{N} x(i) = \sum_{i=1}^{N} m$$

$$\Leftrightarrow \sum_{i=1}^{N} x(i) = Nm \quad \Leftrightarrow m = \frac{1}{N} \sum_{i=1}^{N} x(i)$$

The average value

Spatial smoothing and image approximation (cont...)

Another criterion is the the sum of absolute differences.

$$E = \sum_{i=1}^{N} \left| x(i) - m \right| \quad \Leftrightarrow m = \arg\min_{m} \left\{ \sum_{i=1}^{N} \left| x(i) - m \right| \right\}$$

$$\frac{\partial E}{\partial m} = 0 \Leftrightarrow -\sum_{i=1}^{N} sgn\left(x(i) - m \right) = 0, \quad sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

There must be equal in quantity positive and negative values.

$$m = \mathrm{median}\{x(i)\}$$

Spatial smoothing and image approximation (cont...)

- The median filter is non linear:

$$\mathrm{median}\{x+y\} \neq \mathrm{median}\{x\} + \mathrm{median}\{y\}$$

- It works well for impulse noise (e.g. salt and pepper).
- It requires sorting of the image values.
- It preserves the edges better than an average filter in the case of impulse noise.

Spatial smoothing and image approximation (cont...)

Example

x[n]	**1**	**1**	**1**	**1**	**1**	**2**	**2**	**2**	**2**	**2**

edge

Impulse noise

x[n]	**1**	**3**	**1**	**1**	**1**	**2**	**3**	**2**	**2**	**3**

Median (N=3)

x[n]	**-**	**1**	**1**	**1**	**1**	**2**	**2**	**2**	**2**	**-**

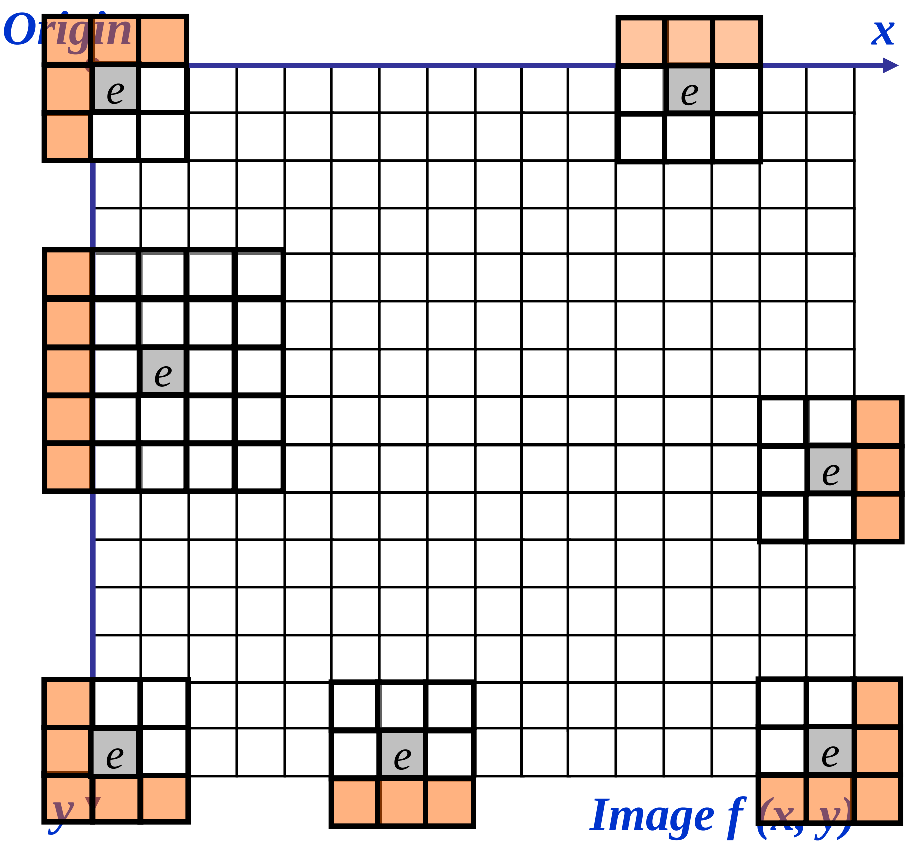
Average (N=3)

x[n]	**-**	**1.7**	**1.7**	**1**	**1.3**	**2**	**2.3**	**2.3**	**2.2**	**-**

The edge is smoothed

Strange Things Happen At The Borders!

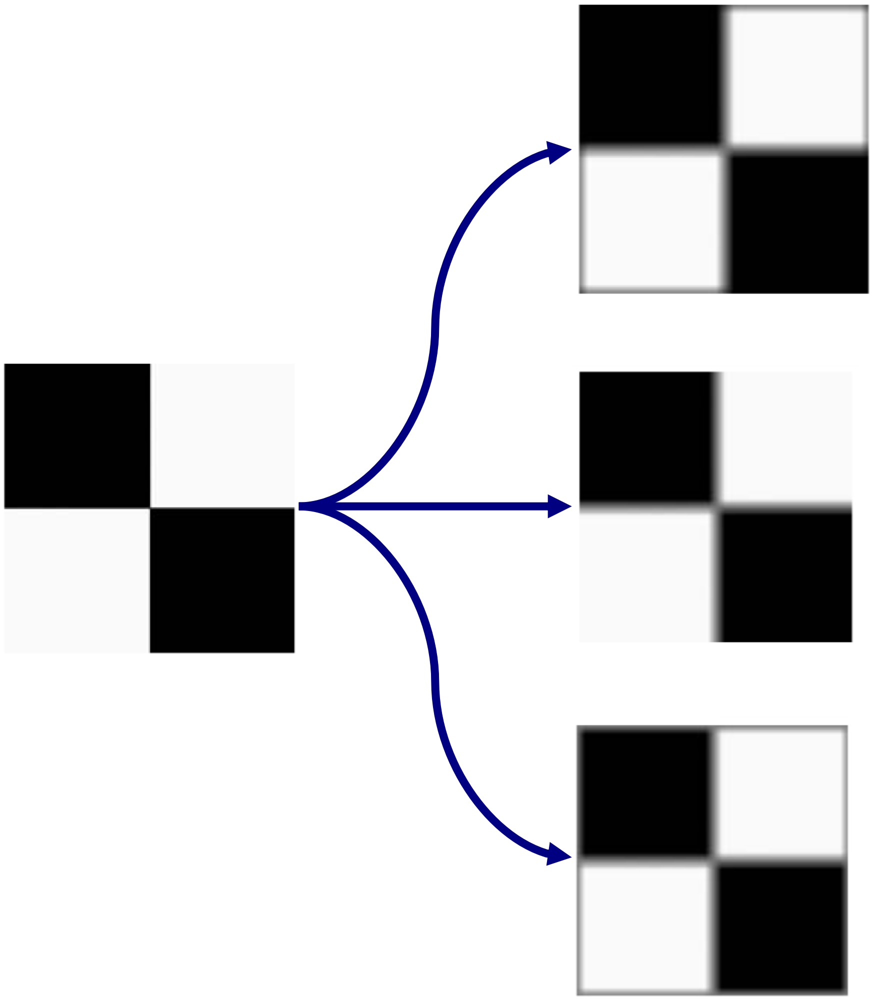
At the borders of an image we are missing pixels to form a neighbourhood



There are a few approaches to dealing with missing border 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

Original Image



Filtered Image: Zero Padding

Filtered Image: Replicate border Pixels

Filtered Image: Wrap Around border Pixels

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.