SE3IA11/SEMIP12
Image Analysis

# Image Enhancement in the Spatial Domain

Lecturer:
Prof. James Ferryman, Computational Vision Group
Email: j.m.ferryman@reading.ac.uk

## Introduction

- The purpose of image enhancement is to emphasise some selected (desirable) image features and suppress others (undesirable features)
- Precursor to information extraction
- The desirability of image features is application/task dependent
  - E.g. for human observers: to improve the interpretability or perception of detail
  - E.g. for automated image image processing: to provide “enhanced” input
- Hence, there exists no formal definition of image enhancement
- Process is usually interactive and iterative

## Introduction

- Image enhancement approaches fall into 2 broad categories
  - Spatial domain methods
    - Refers to the image plane itself, and operate directly on the image pixels
  - Frequency domain methods
    - Operate on the Fourier transform of an image

- In this lecture, we will concentrate on the spatial domain, with frequency domain methods covered in the next lecture

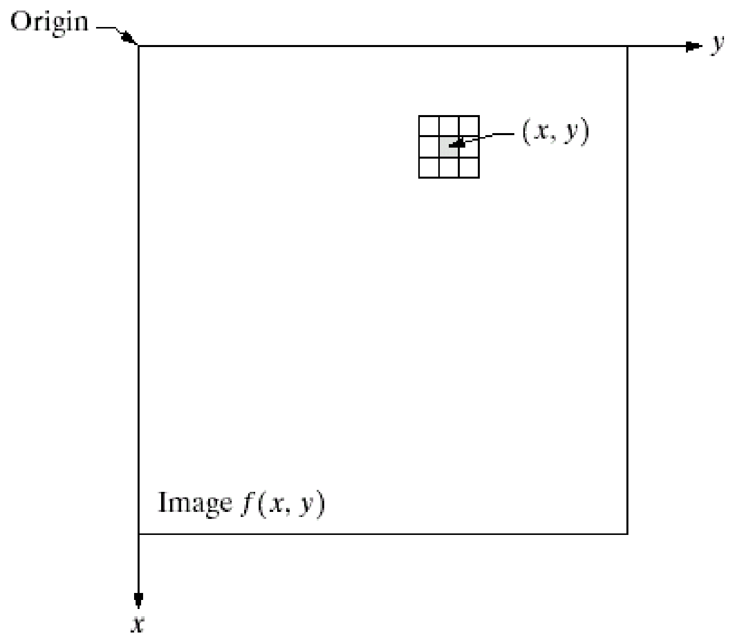
## Preliminaries

- The term *spatial domain* refers to an aggregate of pixels composing an image
- Spatial domain methods operate directly on these pixels
  - Denoted by $g(x,y) = T[f(x,y)]$
    where *f(x,y)* is the input image, *g(x,y)* is the processed image, and *T* is an operator defined some neighbourhood of (x,y)

    *T* can also operate over a set of input images (e.g. pixel by pixel sum of K images for noise reduction)

## Preliminaries

- The main approach to defining a neighbourhood about a pixel *(x,y)* is to use a square (or rectangular) subimage centered at *(x,y)*



## Preliminaries

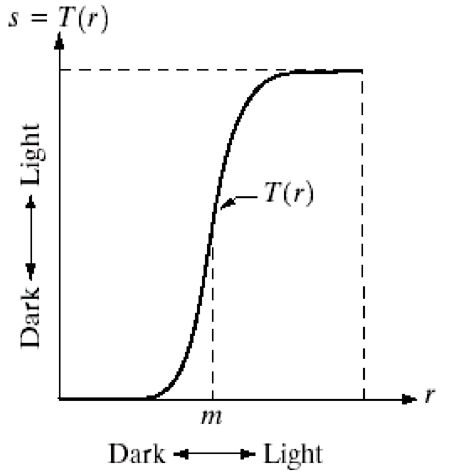
- The centre of the subimage is moved from pixel to pixel starting, e.g. at top left hand corner
- Operator *T* is applied at each location *(x,y)* y to obtain the output, g, at that location
- Process uses only the pixels in the area of the image spanned by the neighbourhood
- Square and rectangular arrays are easiest to implement

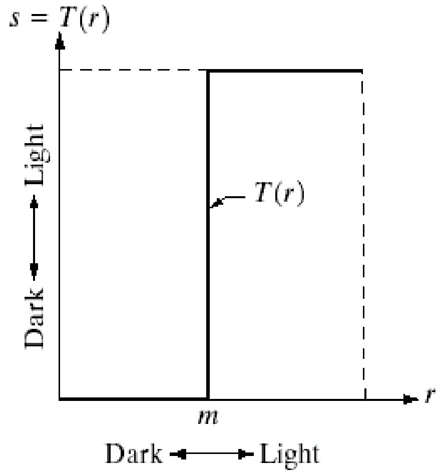
## Introduction to Grey Level Transforms

- Simplest form of *T* is when neighbourhood is 1x1 (i.e. single pixel): *T* becomes a grey level transformation function of the form

$$s = T(r)$$

  - where *r* and *s* are respectively the grey level of *f(x,y)* and *g(x,y)* at any point *(x,y)*



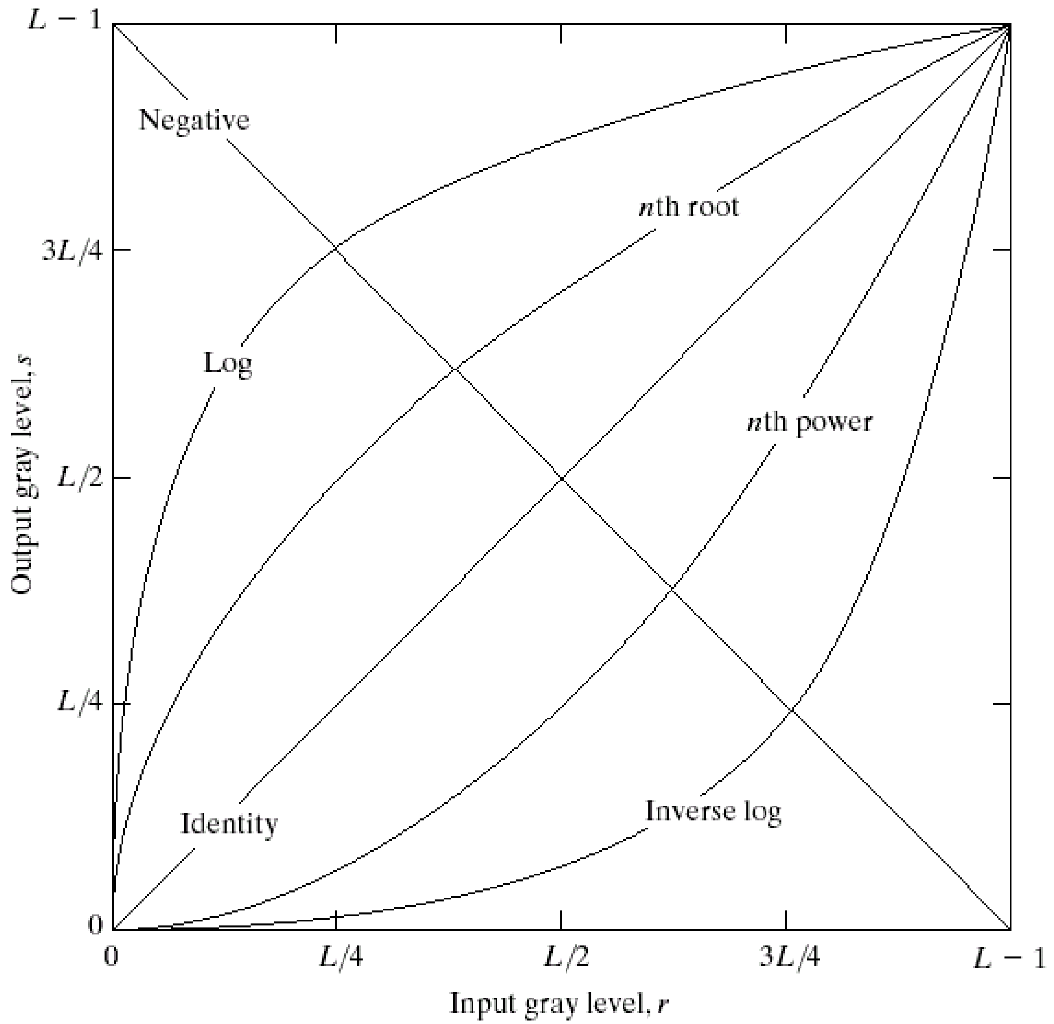


## Grey Level Transformations

- Among the simplest of all enhancement techniques
- Assume $s = T(r)$, where $r$ and $s$ are value of pixels before and after processing
- Values of transformation function $T(r)$ are typically stored in 1D array, and mappings from $r \rightarrow s$ are implemented via table lookups
  - For an 8-bit image, lookup table contains 256 entries
- 3 basic types of functions used frequently for enhancement are:
  - Linear (negative and identity transformations)
  - Logarithmic (log and inverse-log transformations)
  - Power-Law (*n*th power and *n*th root transformations)

Identity function is simplest: output intensities identical to input

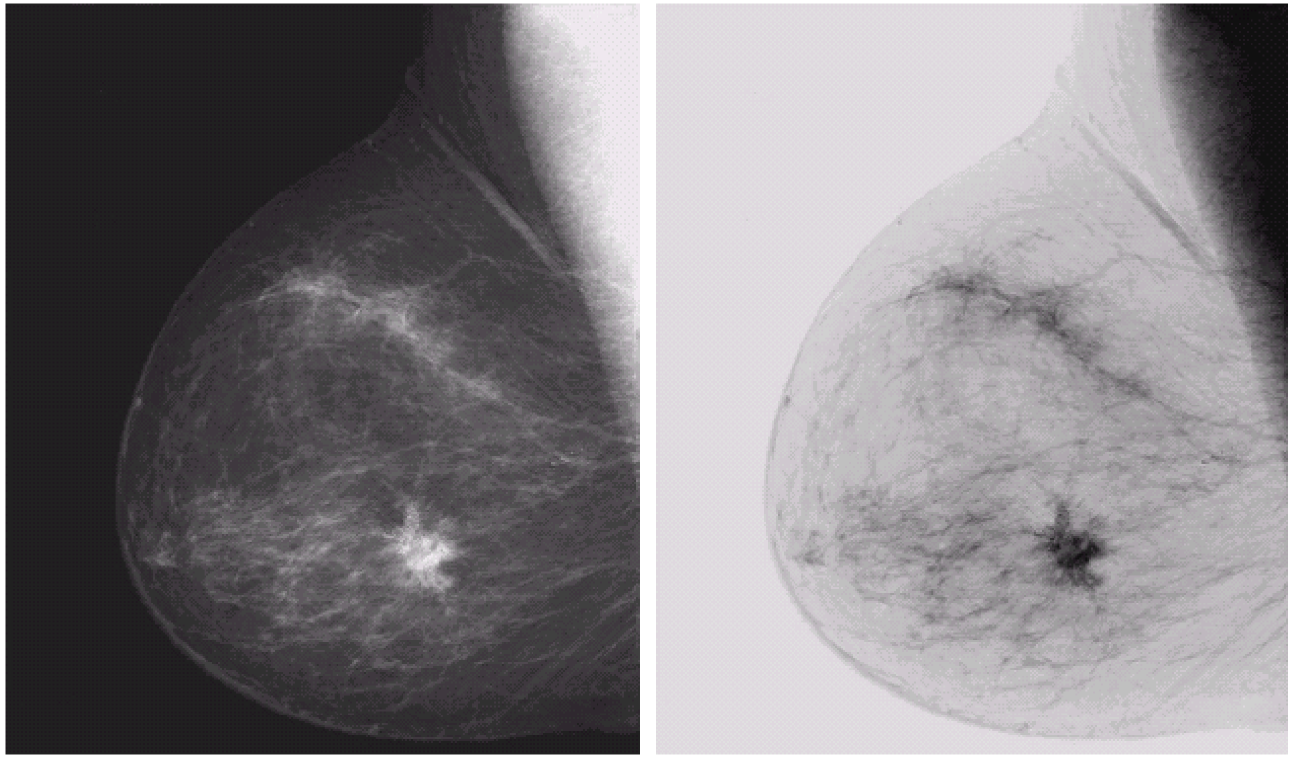
# Grey Level Transformations



## Image Negatives

- The negative of an image with grey levels in the range [0, $L$-1] s obtained by using negative transformation (see previous slide) and given by the expression $s = L - 1 - r$
- Reversing the intensity levels produces equivalent of photographic negative
- Processing is especially suited for enhancing white or grey level detail embedded in dark image regions

## Image Negatives



- Original image is digital  mammogram showing small legion
- Note that while visual content is same in both images, it is easier to analyse the breast tissue in the negative image (in this case)

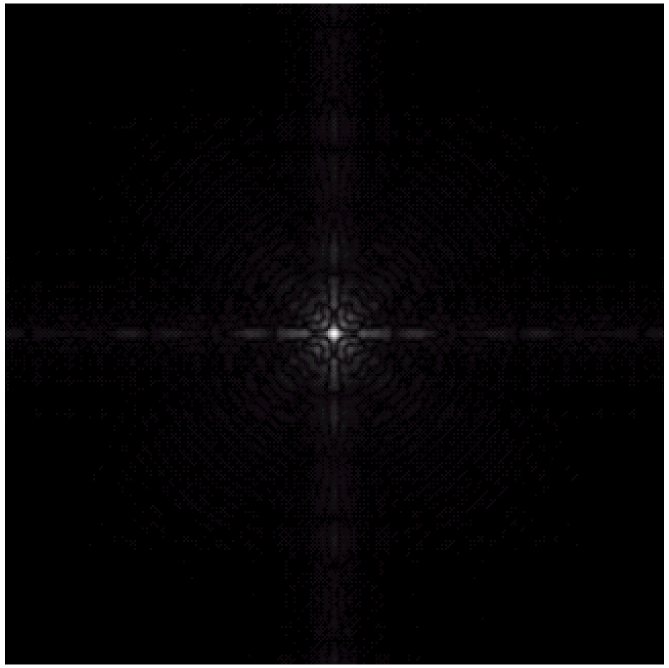
# Log Transformations

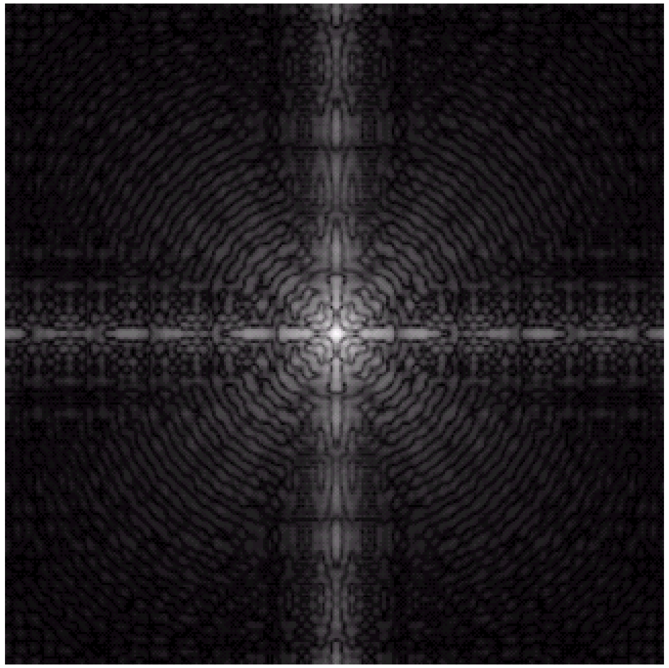
- The general form of log transformation is

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

  where *c* is a constant, and it is assumed that $r \geq 0$
- The transformation maps a narrow range of low greylevel values in input image into a wider range of output levels; opposite true of higher values of input levels
- Transformation is used to expand values of dark pixels, while compressing higher-level values
  - Compresses dynamic range of images with large variations in pixel values
- Inverse log transformation performs the opposite

## Log Transformations



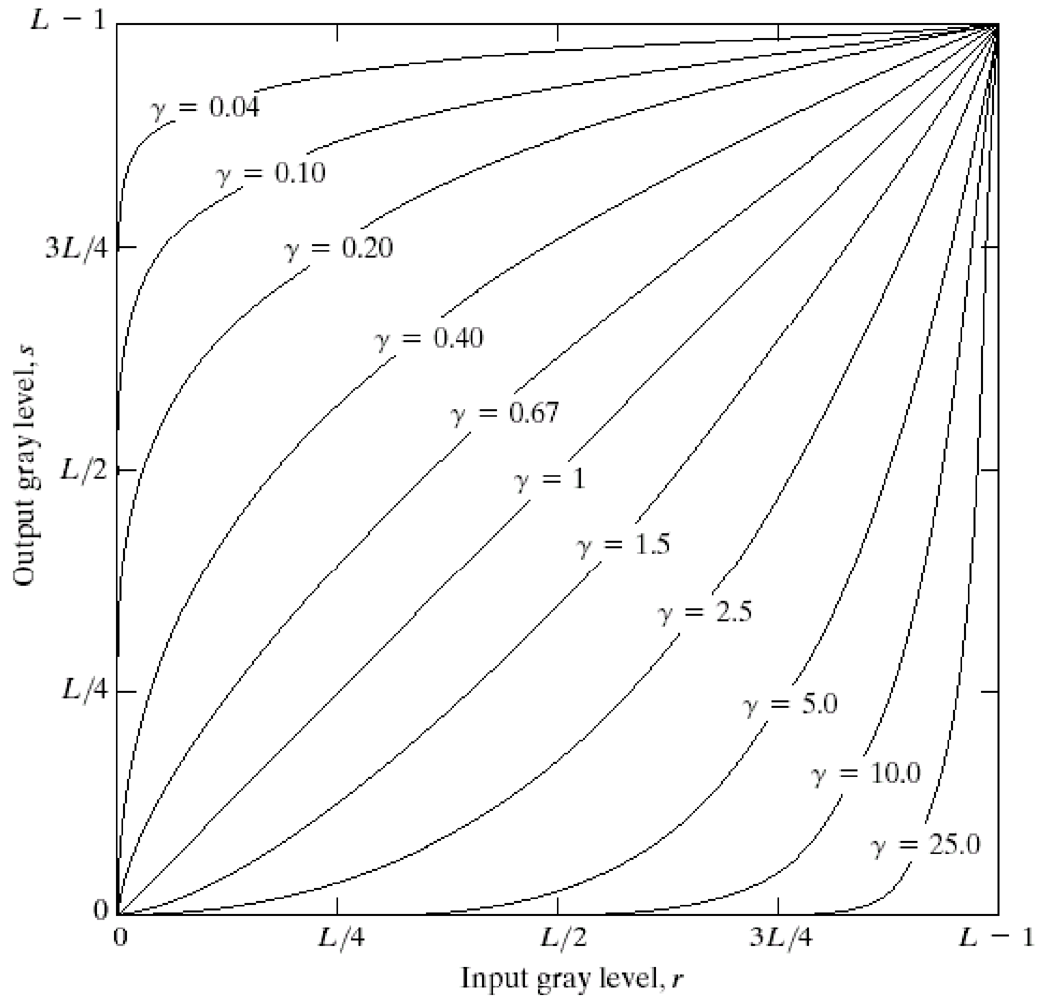


- Figure (left) shows a Fourier spectrum (next lecture) with values in range 0 to 1.5 x $10^6$ : brightest pixels dominate
- Post linear scaling ($c = 1$) (right image) visible detail is enhanced
- Most Fourier spectra seen in image processing are scaled in this way

## Power-Law Transformations

- Power-law transformations have basic form $s = cr^{y}$
  where $c$ and $y$ are positive constants
- As for log transformation, power-law curves with fractional values $y$ map a narrow range of dark input values into a wider range of output values
- Unlike log function, a *family* of transformation curves can be produced, by varying $y$
- Many devices used for capture, printing, display of images respond according to power law
  - Exponent in power-law equation is referred to as *gamma*
  - Process used for correction is called gamma correction

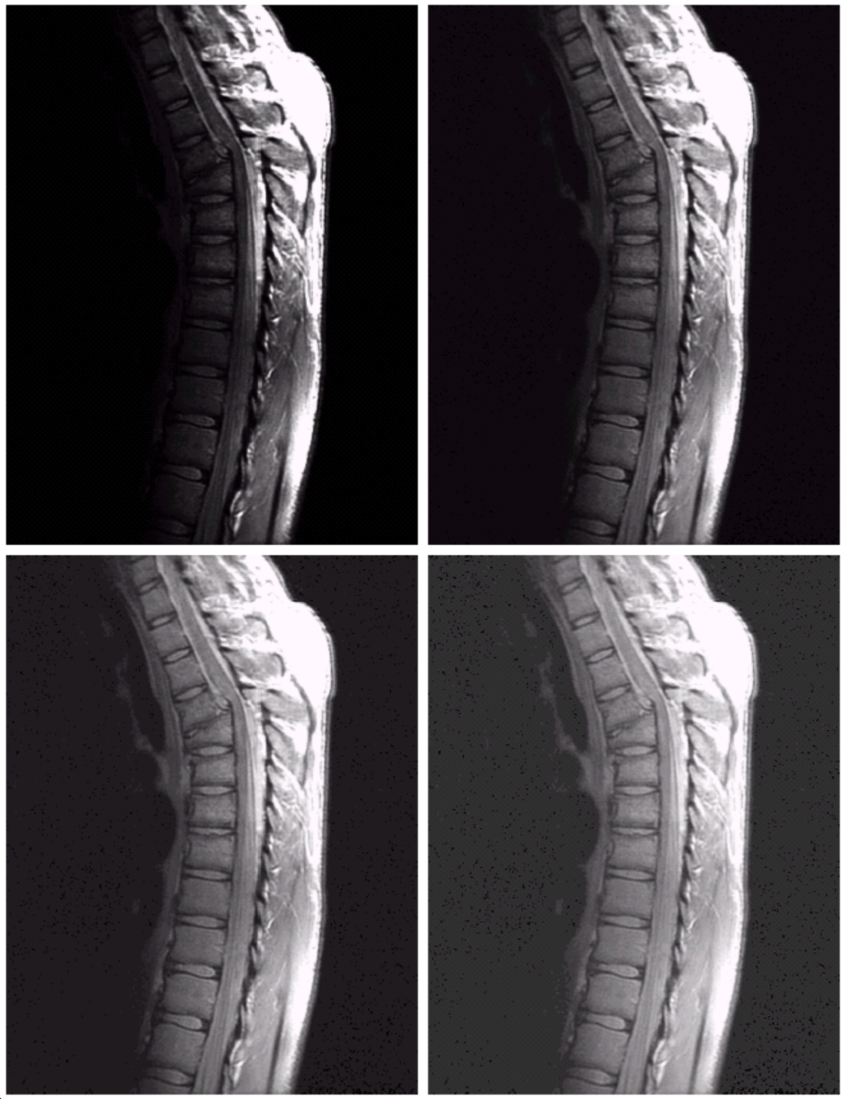
## Power-Law Transformations



## Power-Law Transformations

- Power-law functions are useful for general-purpose contrast manipulation
- Following figure is a magnetic resonance (MR) image of an upper thoracic human spine with fracture dislocation
- Fracture is visible near vertical centre of spine
- As image is dark, expansion of grey levels is desirable
- Accomplished with power-law transformation
- $c = 1$ fixed; $y$ varies from 0.6, 0.4, 0.3

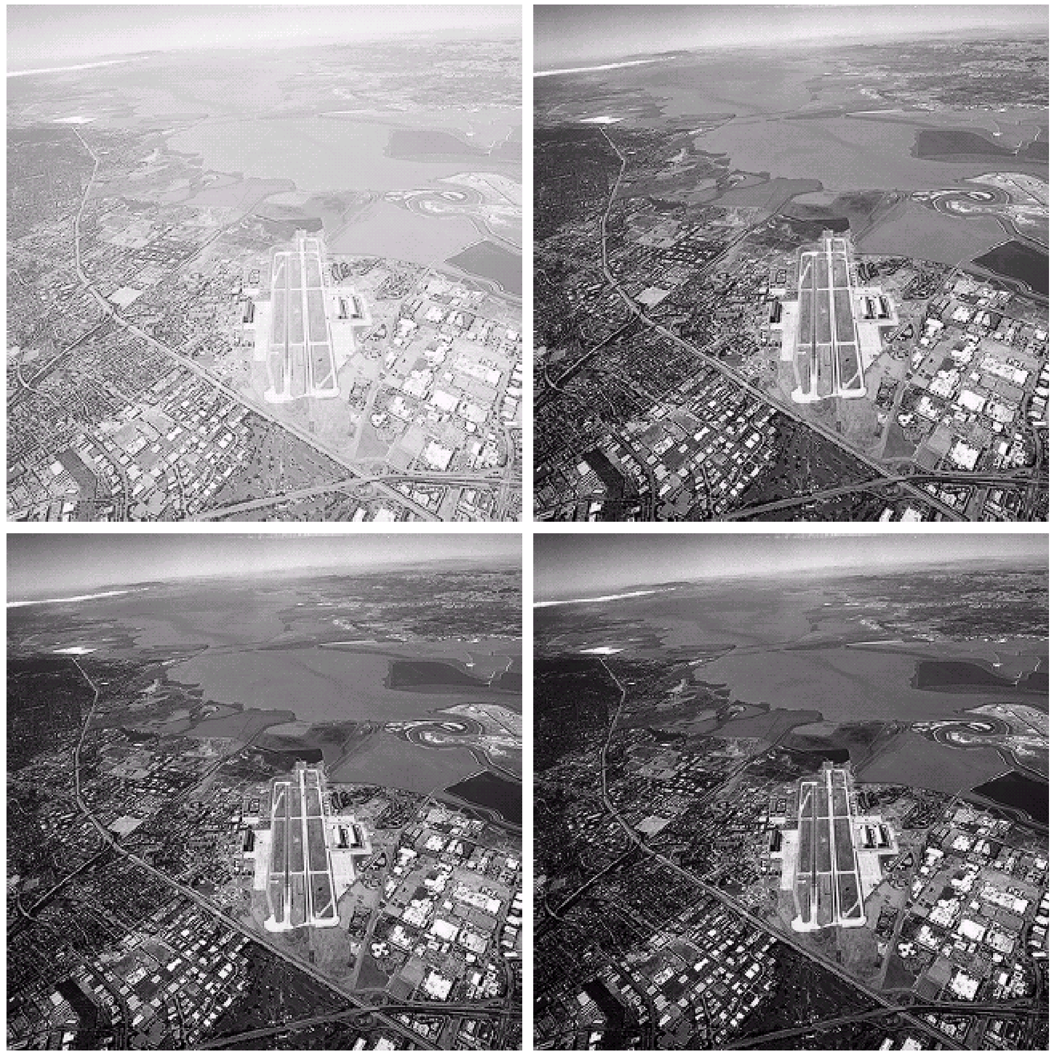
# Power-Law Transformations - Example



## Power-Law Transformations - Example

- The next figure illustrates the opposite problem to the previous example
- The image has a washed-out appearance
  - Indicates that a compression of grey levels is desirable
  - Can be achieved using power-law transform with $\gamma > 1$
- Images show processing with $y = 3.0, 4.0, 5.0$

## Power-Law Transformations - Example



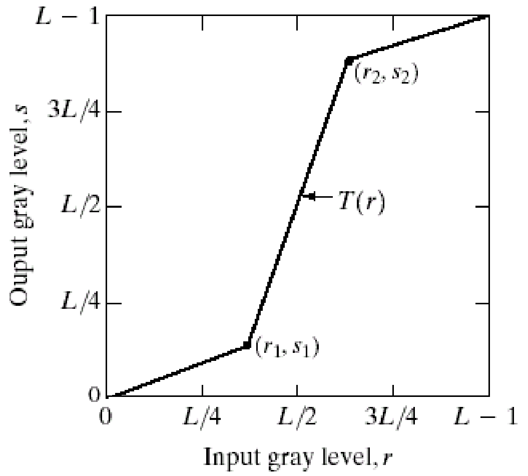
## Piecewise-Linear Transformations

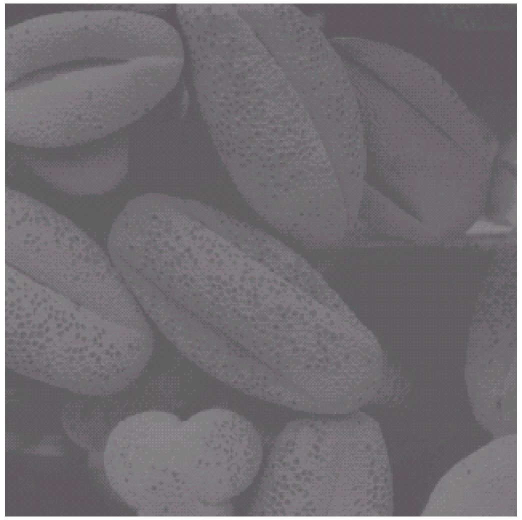
- Complementary approach to previous methods
- Main advantage is that form of piecewise functions can be arbitrarily complex
- Main disadvantage is that their specification requires more user input

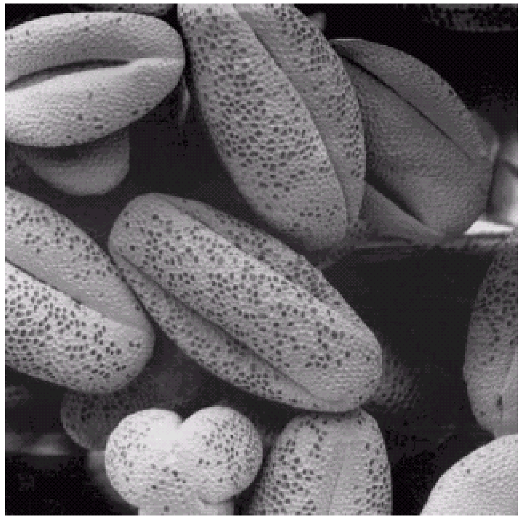
## Contrast Stretching

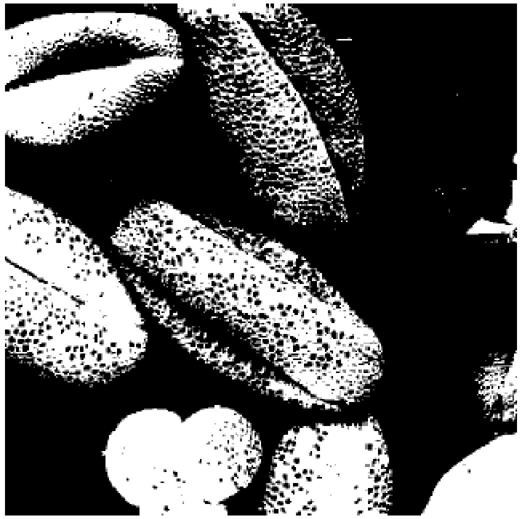
- One of simplest piecewise linear functions
- Idea is to increase dynamic range of image grey levels
- Main application is low contrast images
  - Due to poor illumination, lack of dynamic range in sensor, wrong settings during image capture, …
- Example shows typical transformation used
- Locations of points $(r_1,s_1)$ and $(r_2,s_2)$ control shape of transformation

# Contrast Stretching - Example









$(r_1,s_1) = (r_{min},0)$ & $(r_2,s_2) = (r_{max}, L\text{-}1)$, where $r_{min}$ & $r_{max}$ = min/max image greylevels

$r_1 = r_2 = m$ (mean image greylevel)

University of Reading

## Contrast Stretching

- If $r_1 = s_1$ and $r_2 = s_2$, the transform is a linear function, producing no change
- If $r_1 = r_2$, $s_1 = 0$, and $s_2 = L - 1$, transform is a thresholding function
- Intermediate values of $(r_1, s_1)$ and $(r_2, s_2)$ produces varying degrees of spread in grey levels of output image, i.e. change in contrast
- In general, $r_1 \le r_2$ and $s_1 \le s_2$ is assumed so function is single valued and monotonically increasing
  - Preserves order of greylevels and prevents artifacts in output image

## Greylevel & Bitplane Slicing

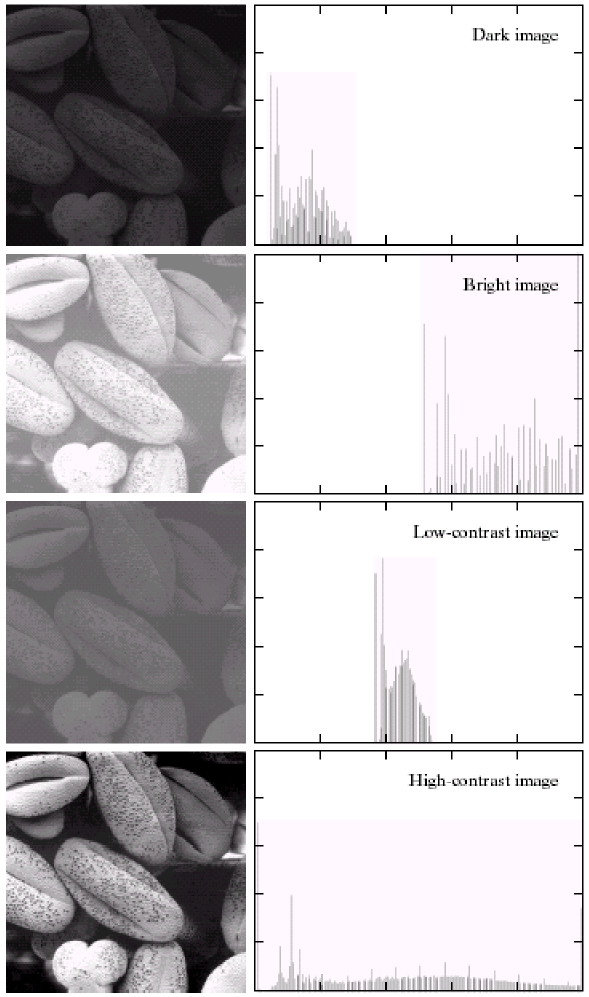
- Greylevel Slicing
  - Highlighting specific range of greylevels in image is often desired
  - One approach is to display high value for all greylevels in range of interest, and low value for all others
  - Another approach brightens desired range of grey levels but preserves background and tones in image

- Bitplane slicing
  - Instead of emphasising ranges, contribution made by specific bits to total image appearance, may be desired
  - Separating image into bit planes is useful for analysis of relative importance of each bit to an image, providing information on adequacy of number of bits used to quantize each pixel

## Histogram Processing

- The histogram of an image with grey levels in range [0, $L$-1] is a discrete function $h(r_k) = n_k$, where $r_k$ is the $k_{th}$ grey level and $n_k$ is the number of image pixels having greylevel $r_k$
- It is usual practice to normalise a histogram by dividing each of the values by total number, $N$, of image pixels
  - Normalised histogram given by $p(r_k) = \frac{n_k}{N}$ for $k = 0, 1, \ldots, L-1$
- $p(r_k)$ provides an estimate of probability of occurrence of greylevel $r_k$

## Histogram Processing - Example



## Histogram Equalisation

- Powerful enhancement method that seeks to optimise image contrast
  - Flattens (equalises) image histogram – redistribute greylevels uniformly
  - Non-linear
- Especially useful in images with large regions of similar appearance (e.g. light background, dark foreground)
- Can be used to expose hidden detail

## (Discrete) Histogram Equalisation

- Uses the cumulative distribution function (CDF) as a lookup table
- CDF essentially answers: "what percentage of samples in image are equal to or less than value k?
- The normalized CDF is defined as: $C_k = \sum_{i=0}^{k} p(r_i)$ where $p(r_k) = \frac{n_k}{N}$
- CDF is monotonically increasing
  - Derivative (slope) is steep where source image has much information, flat conversely
  - CDF of perfect equalised image is straight line, with slope = 1
- The equalised histogram can then be computed as:

$$s_k = (L-1)\sum_{i=0}^{k} p_r(r_i) \qquad 0 \le s_k \le 1$$

The values of $s_k$ will have to be scaled up by 255 and rounded to the nearest integer so that the output values of this transformation will lie in the range [0, 255]. Thus, the discretisation and rounding of $s_k$ to the nearest integer will mean that the transformed image will not have a perfectly uniform histogram

## Histogram Equalisation - Example

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

| Value | Count | Value | Count | Value | Count | Value | Count | Value | Count |
|---|---|---|---|---|---|---|---|---|---|
| 52 | 1 | 64 | 2 | 72 | 1 | 85 | 2 | 113 | 1 |
| 55 | 3 | 65 | 3 | 73 | 2 | 87 | 1 | 122 | 1 |
| 58 | 2 | 66 | 2 | 75 | 1 | 88 | 1 | 126 | 1 |
| 59 | 3 | 67 | 1 | 76 | 1 | 90 | 1 | 144 | 1 |
| 60 | 1 | 68 | 5 | 77 | 1 | 94 | 1 | 154 | 1 |
| 61 | 4 | 69 | 3 | 78 | 1 | 104 | 2 | | |
| 62 | 1 | 70 | 4 | 79 | 2 | 106 | 1 | | |
| 63 | 2 | 71 | 2 | 83 | 1 | 109 | 1 | | |

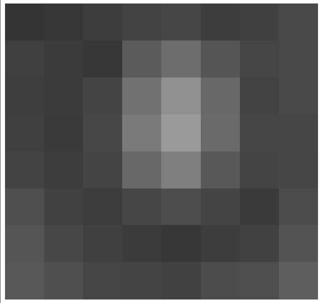
| Value | cdf | Value | cdf | Value | cdf | Value | cdf | Value | cdf |
|---|---|---|---|---|---|---|---|---|---|
| 52 | 1 | 64 | 19 | 72 | 40 | 85 | 51 | 113 | 60 |
| 55 | 4 | 65 | 22 | 73 | 42 | 87 | 52 | 122 | 61 |
| 58 | 6 | 66 | 24 | 75 | 43 | 88 | 53 | 126 | 62 |
| 59 | 9 | 67 | 25 | 76 | 44 | 90 | 54 | 144 | 63 |
| 60 | 10 | 68 | 30 | 77 | 45 | 94 | 55 | 154 | 64 |
| 61 | 14 | 69 | 33 | 78 | 46 | 104 | 57 | | |
| 62 | 15 | 70 | 37 | 79 | 48 | 106 | 58 | | |
| 63 | 17 | 71 | 39 | 83 | 49 | 109 | 59 | | |

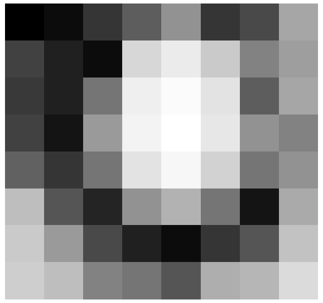
$$s_k = round\left[(L-1)\times\frac{cdf(k)-cdf_{\min}}{N-cdf_{\min}}\right]$$

$$s_{78} = round\left[255\times\frac{cdf(k)-1}{63}\right]$$

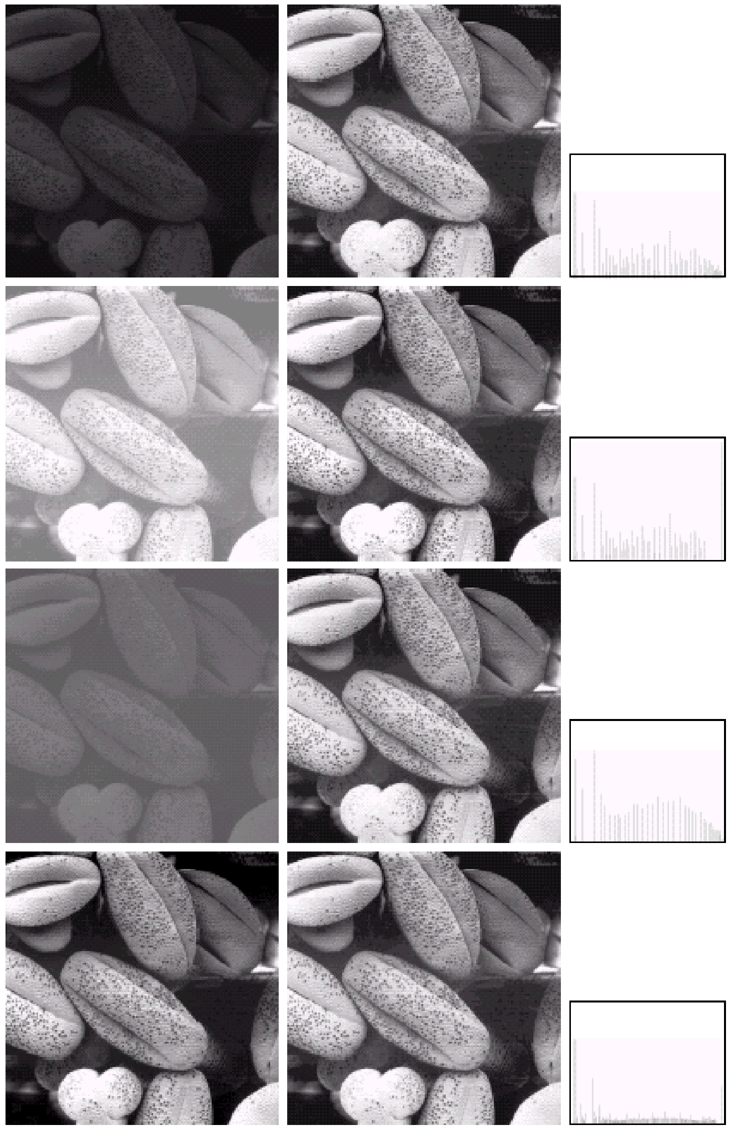
$$s_{78} = round(0.714286\times 255) = 182$$

$$\begin{bmatrix} 0 & 12 & 53 & 93 & 146 & 53 & 73 & 166 \\ 65 & 32 & 12 & 215 & 235 & 202 & 130 & 158 \\ 57 & 32 & 117 & 239 & 251 & 227 & 93 & 166 \\ 65 & 20 & 154 & 243 & 255 & 231 & 146 & 130 \\ 97 & 53 & 117 & 227 & 247 & 210 & 117 & 146 \\ 190 & 85 & 36 & 146 & 178 & 117 & 20 & 170 \\ 202 & 154 & 73 & 32 & 12 & 53 & 85 & 194 \\ 206 & 190 & 130 & 117 & 85 & 174 & 182 & 219 \end{bmatrix}$$


Original


Equalized

## Histogram Equalisation



## Equalisation: Summary

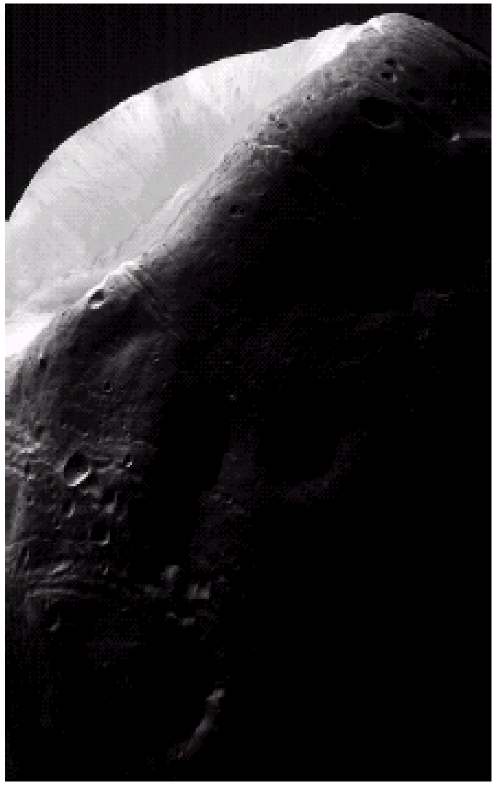
- Determines a transformation function that seeks to produce an output image with uniform histogram
- May be performed automatically
- Results are predictable
- Method is simple to implement

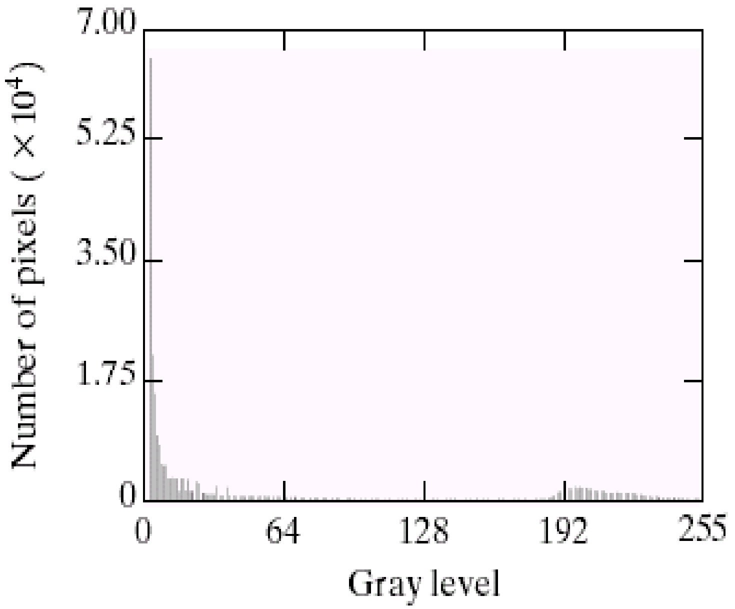
- However....

  there exist applications in which enhancement based on a *uniform* histogram is not optimal; it is better to specify the *shape* of the histogram that the processed image should possess

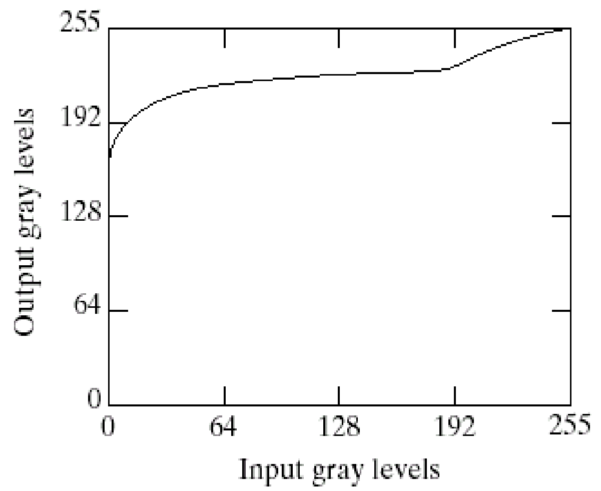
- Method used to undertake this is called *histogram matching* (specification)

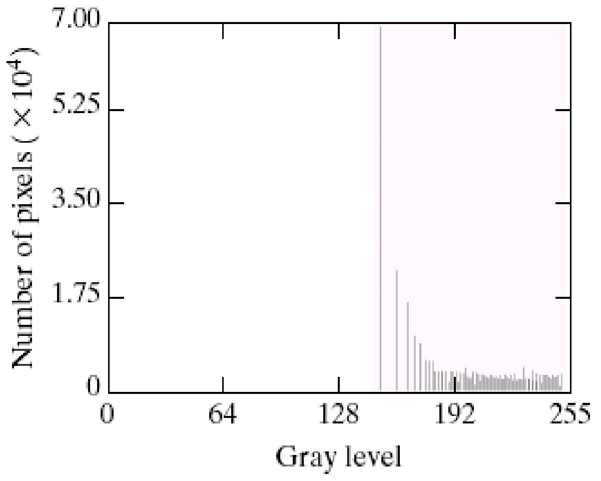
# Histogram Matching

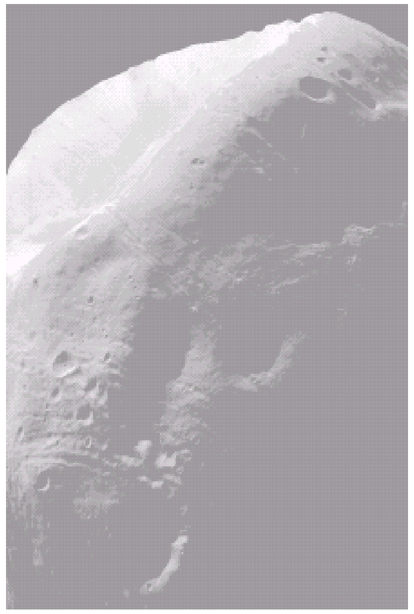




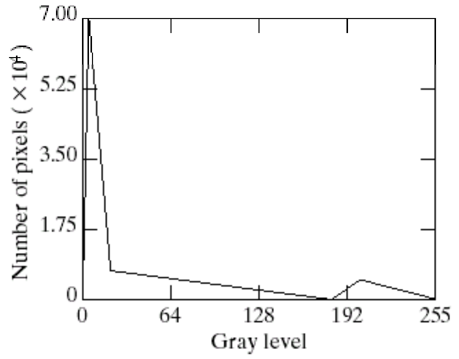
## Histogram Matching

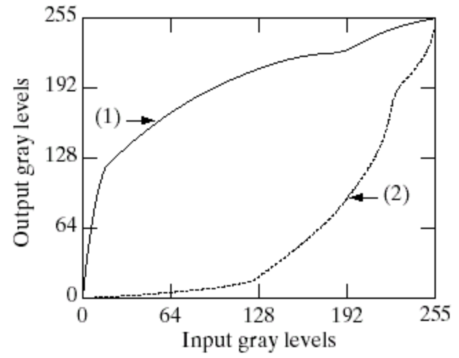


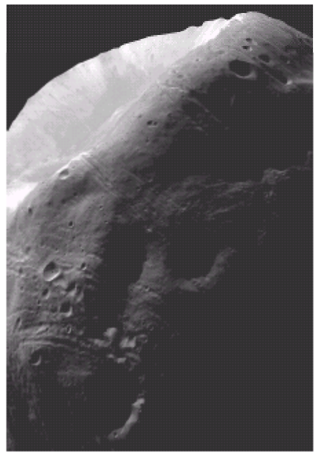


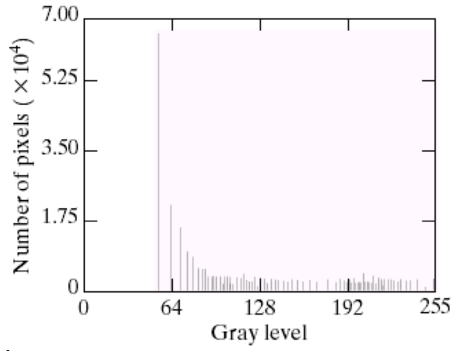


# Histogram Matching









## Histogram Matching

- For most part, a trial-and-error process
- Guidelines may be available for certain types of data
- There may exist cases in which an “average” histogram should look like and use this as specified histogram
- In general, however, there are no rules for specifying histograms
- One must resort to case-by-case analysis for enhancement

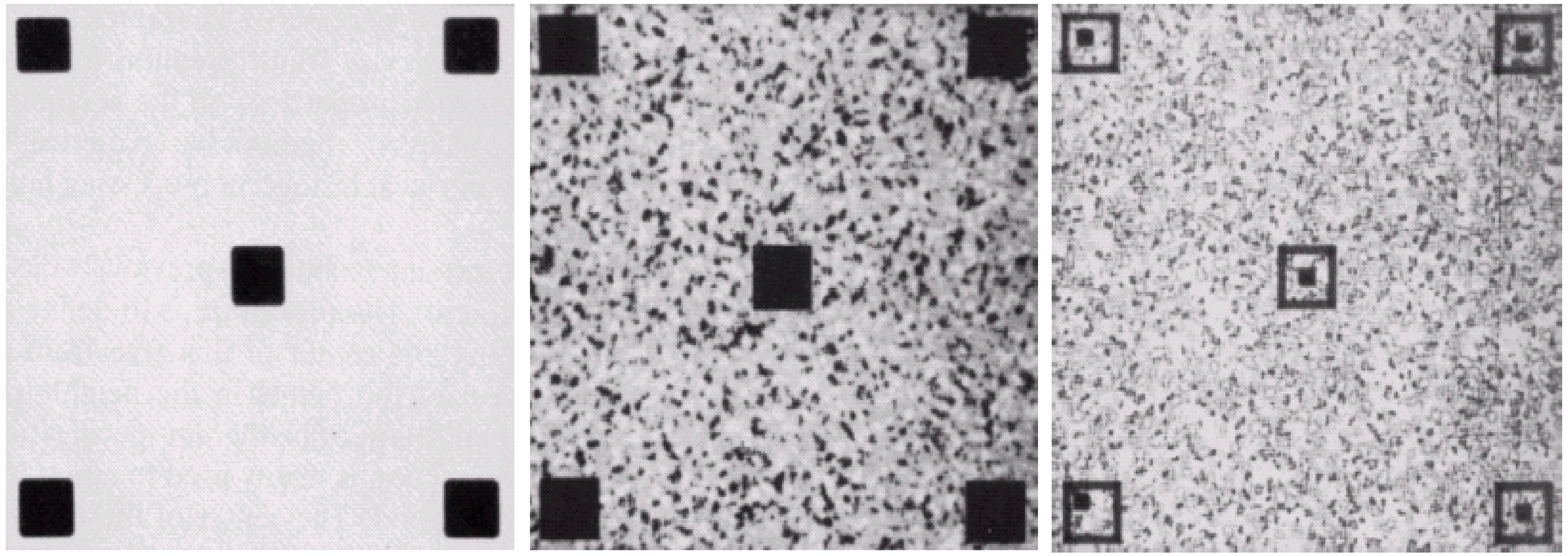
## Local Enhancement

- Previous histogram methods are global
  - Pixels are modified by transformation function based on greylevel content of entire image
- Sometimes necessary to enhance detail over small image areas
- Solution is to devise transformation functions based on the greylevel distribution in the neighborhood of every pixel

## Local Enhancement

- The procedure is as follows:
  - Define a square (or rectangular) neighborhood and move the centre of this area from pixel to pixel
  - At each location, the histogram of the points in the neighborhood is computed and either a histogram equalisation or histogram specification transformation function is obtained
  - This function is finally used to map the greylevel of the pixel centered in the neighborhood
  - The centre is then moved to an adjacent pixel location and the procedure repeated

## Local Enhancement: Example



## Arithmetic/Logic Operations

- Performed on a pixel-by-pixel basis between 2 or more images (except NOT on single image)
- Operations include:
  - Subtraction, Addition, Multiplication, Division
  - NOT
    - Used for negative transformation
  - AND, OR
    - Used for masking (selecting subimage/ROI within an image)

## Image Subtraction

- The difference between two images *f(x,y)* and *h(x,y)*, expressed as $g(x,y) = f(x,y) - h(x,y)$

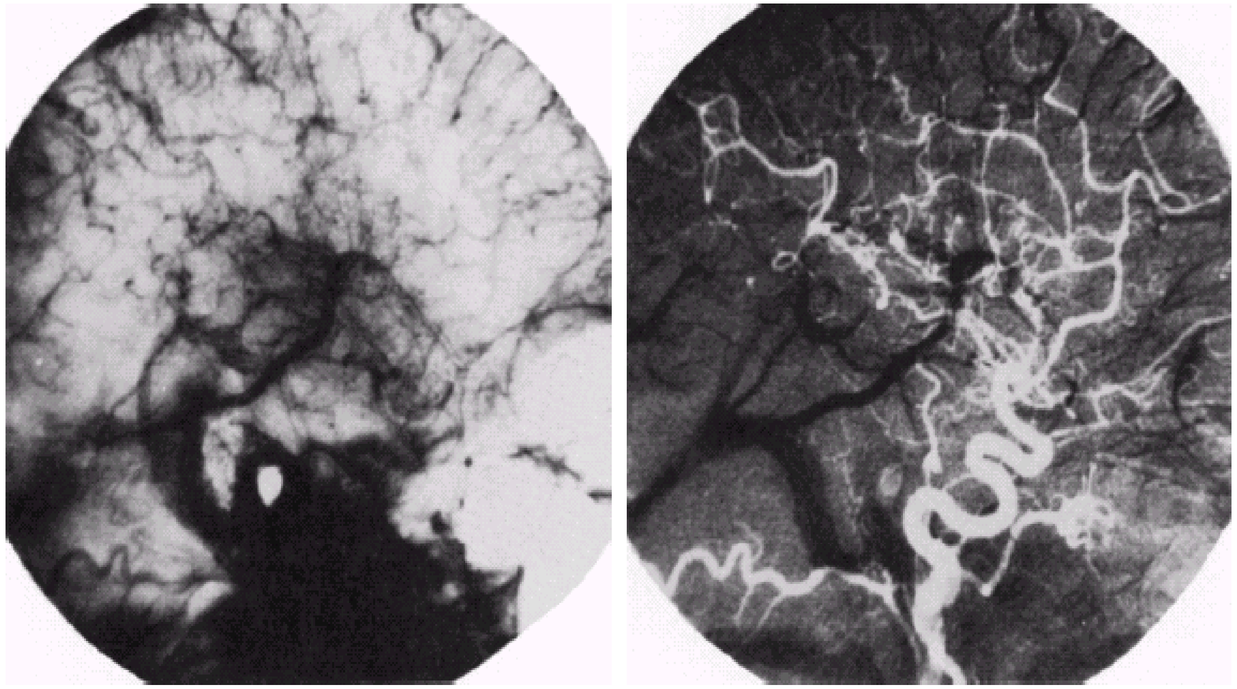
  is obtained by computing the difference between all pairs of corresponding pixels from *f* and *h*

- Provides enhancement of *differences* between images

## Image Subtraction: Radiography

- Major application of image subtraction in medical imaging, specifically *mask mode radiography*
- In this case, *h(x,y)*, the mask, is an X-ray image of region of patient's body captured by intensified camera located opposite X-ray source
- Procedure involves injecting *contrast medium* into patient's bloodstream
- Series of images of same anatomical region as *h(x,y)* captured
- Mask is subtracted from series of incoming images
- Areas different between *f(x,y)* and *h(x,y)* provide detail
- Overall effect is video showing how medium propagates through arteries in area under observation

# Image Subtraction: Radiography



## Image Averaging

- Consider a noisy image *g(x,y)* formed by the addition of noise *n(x,y)* to an original image *f(x,y)*

$$g(x, y) = f(x, y) + n(x, y)$$

  where the assumption is the noise is uncorrelated and has zero average value
- Objective is to reduce noise content by adding a set of noisy images $\{g_i(x,y)\}$
- Averaging *M* different noisy images

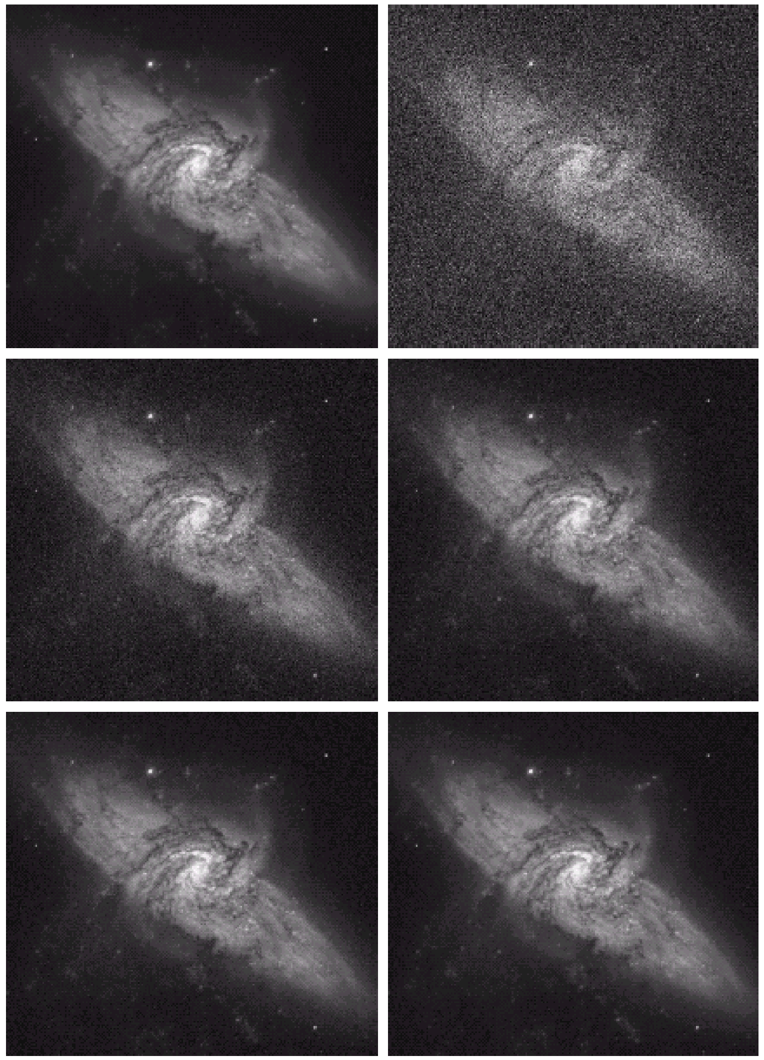
$$\bar{g}(x, y) = \frac{1}{M}\sum_{i=1}^{M} g_i(x, y)$$

## Image Averaging

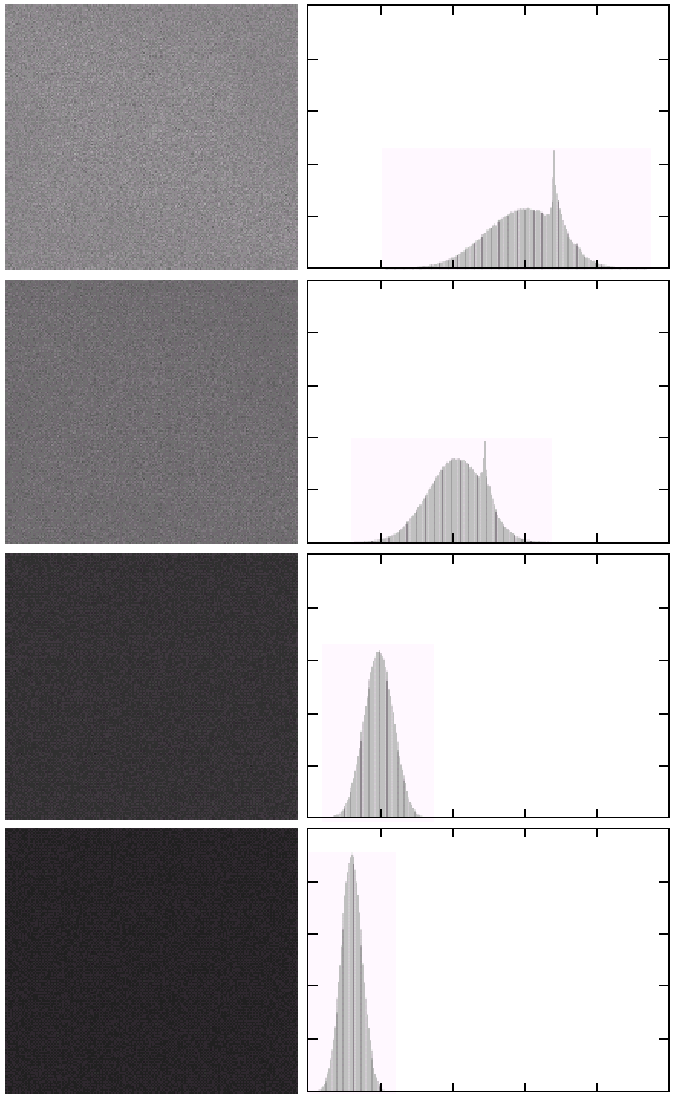
- As M increases, the variability of the pixel values at each location decreases
  - This means that *g(x,y)* approaches *f(x,y)* as the number of noisy images used in the averaging process increases

- Registering (alignment) of the images is necessary to avoid blurring in the output image

## Image Averaging



## Image Averaging



## Introduction to Spatial Filtering

- Use of spatial masks (kernels) for image processing (spatial filters)
- Linear and nonlinear filters
- Low-pass filters eliminate or attenuate high frequency components in the frequency domain (sharp image details), and result in image blurring
- High-pass filters attenuate or eliminate low-frequency components (resulting in sharpening edges and other sharp details)
- Band-pass filters remove selected frequency regions between low and high frequencies (for image restoration, not enhancement)

## Introduction to Spatial Filtering

- In general, linear filtering of an image *f* of size *M* x *N* with a filter mask of size *m* x *n* is given by the expression

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

where *a=(m-1)/2* and *b=(n-1)/2; m* x *n* (odd numbers)

for *x*=0,1,…,M-1 and *y*=0,1,…,N-1

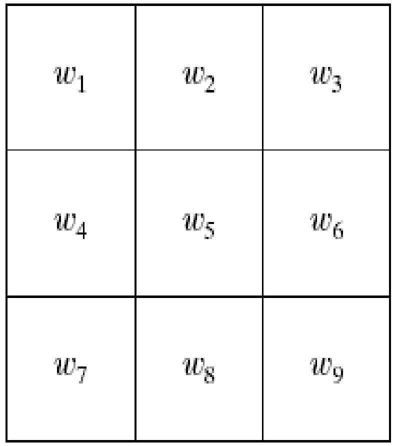
- The process is also called *convolution* (used primarily in the frequency domain)

## Spatial Filtering

- The basic approach is to sum products between the mask coefficients and the intensities of the pixels under the mask at a specific location in the image:

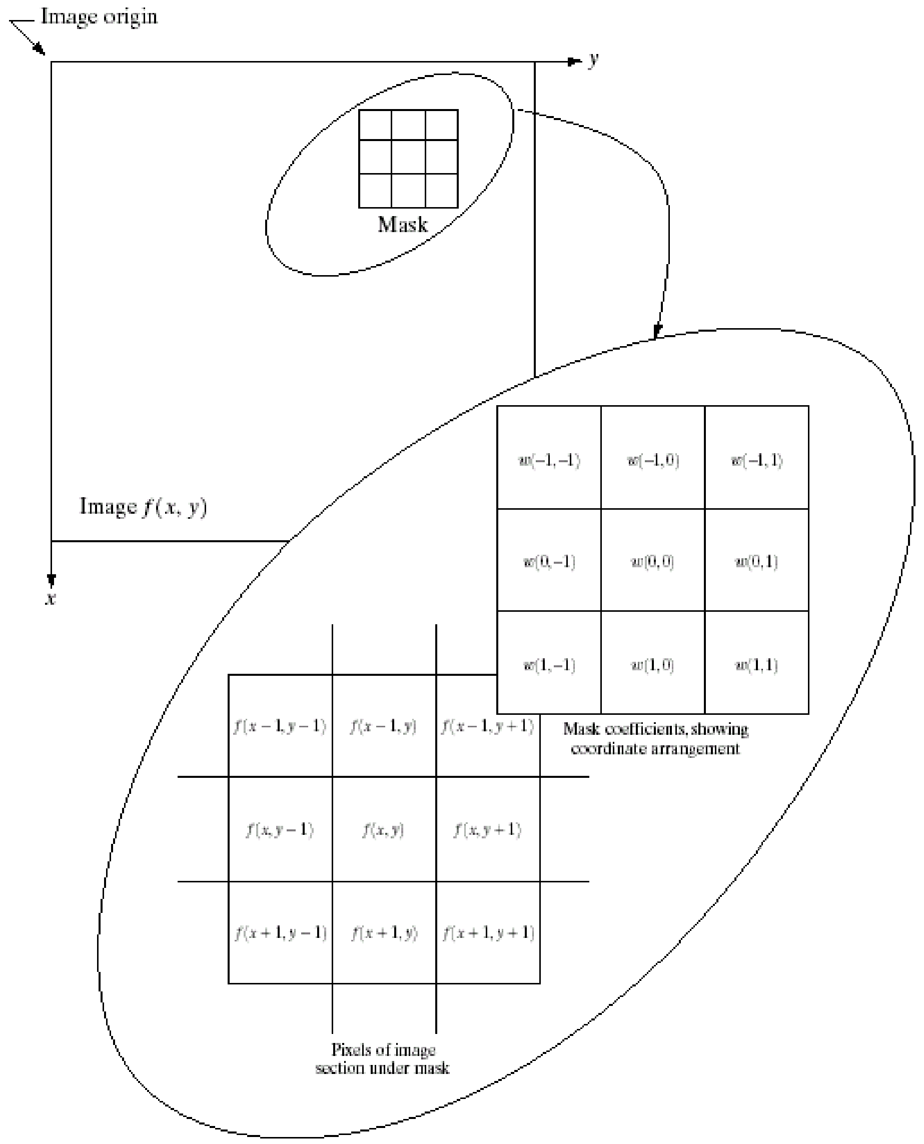
$$R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9$$

for a 3x3 image



| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

# Spatial Filtering



## Spatial Filter: Image Smoothing

- Aim of image smoothing is to diminish effects of camera noise, spurious pixel values, missing pixel values, etc.
- There exist many different techniques for image smoothing
- We will consider *neighbourhood averaging* and *edge-preserving* smoothing

University of Reading

## Smoothing: Local Neighbourhood

- Each point in the smoothed image $F_s(x,y)$ is obtained from average pixel value in neighbourhood $(x,y)$ in input image
- For example, for a 3x3 neighbour hood around each pixel, the following mask would be used:

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

- Each pixel value is multiplied by 1/9, summed, and the result placed in output image
- The mask is successively moved across the image until every pixel has been covered
- In other words, the image has been *convolved* with smoothing mask

University of Reading

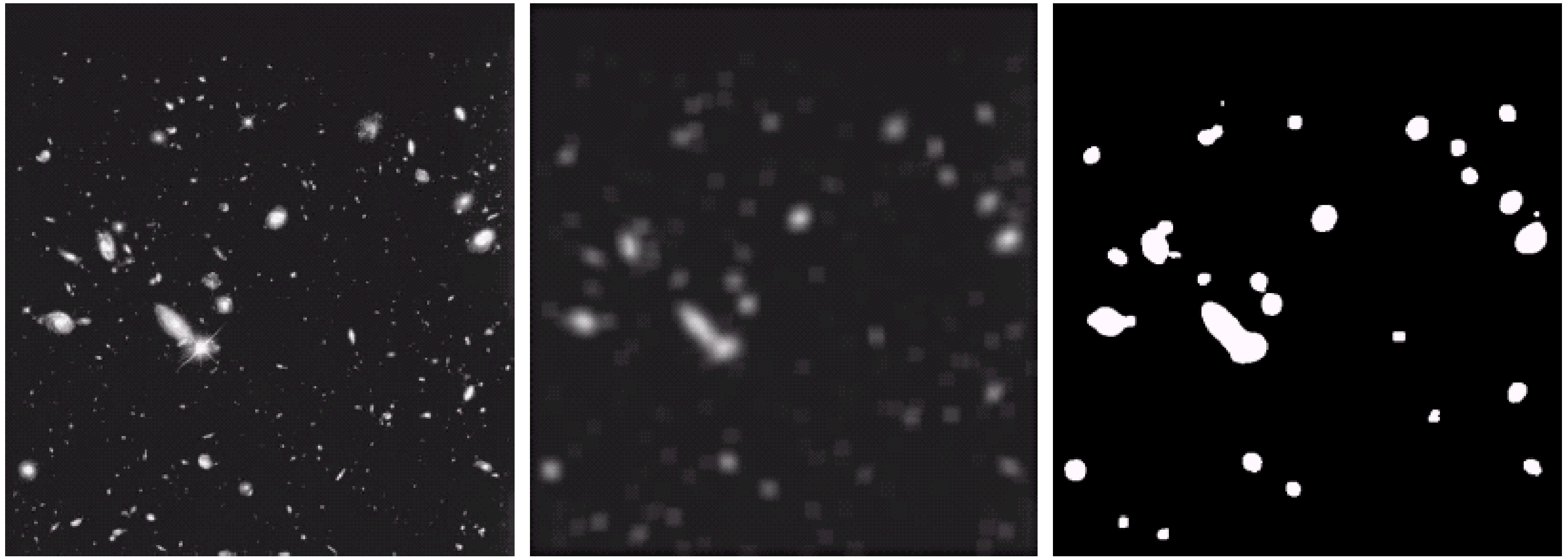
## Smoothing: Local Neighbourhood

- However, one usually expects value of pixel to be more closely related to values of pixels close to it than those further away
  - Most image points are spatially coherent with their neighbours
  - Therefore usual to weight pixels near centre of mask more strongly

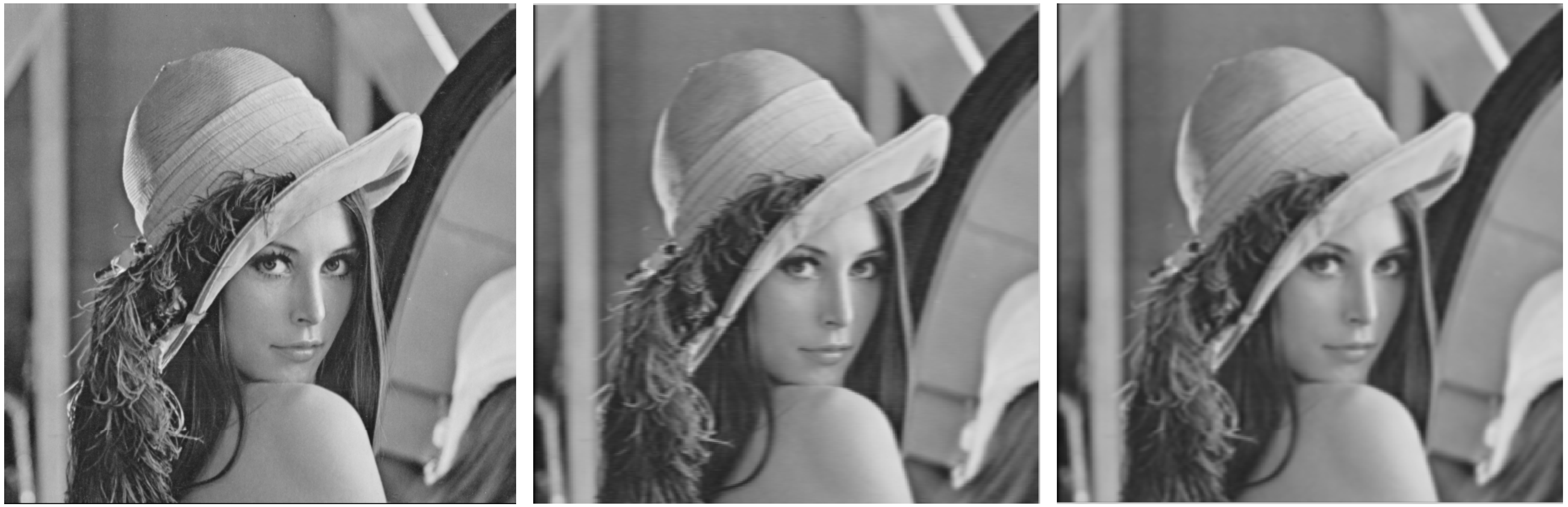
$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

- Common weighting functions include
  - Rectangular, triangular, Gaussian
- In practice, Gaussian smoothing is most commonly used
  - Frequency components are modified in smooth manner

## Averaging Mask: Example



# Averaging Mask: Example



## Smoothing: Edge-Preserving

- Neighbourhood averaging or Gaussian smoothing tends to blur edges, due to attenuation of high frequencies in image
- An alternative approach is to use *median filtering*
- The greylevel of each pixel is replaced by the median of the greylevels in the neighborhood of that pixel (instead of by the average as before)
- Pixels with outlying values are forced to become their neighbours, and edges are preserved
- Non-linear filter
- Used primarily for noise reduction (eliminates isolated spikes)

## Spatial Filter: Image Sharpening

- Main aim is to highlight fine image detail, or to enhance detail that has been blurred (e.g. due to motion, noise)
- Require to enhance high-frequency components
  - Implies filter shape with high positive component at centre
- Masks that achieve image sharpening include use of two 2x2 masks as follows:

| | |
|---|---|
| 1 | 0 |
| 0 | -1 |

| | |
|---|---|
| 0 | 1 |
| -1 | 0 |

- Computes sum of the squares of the differences between diagonally adjacent pixels
- Known at the Roberts Operator
- Each pixel in output image computed as

  tmp1 = absolute_value(input_image(x,y) - input_image(x+1,y+1))
  tmp2 = absolute_value(input_image(x+1,y) - input_image(x,y+1))
  output_image(x,y) = tmp1 + tmp2
- Note that all mask coefficients sum to 0, indicating a response of 0 in constant areas, as expected of a derivative operator

# Spatial Filter: Image Sharpening

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

- Prewitt Operator

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

- Sobel Operator

## Edge Filter: General Sobel

- The Sobel algorithm operates on the full array and evaluates

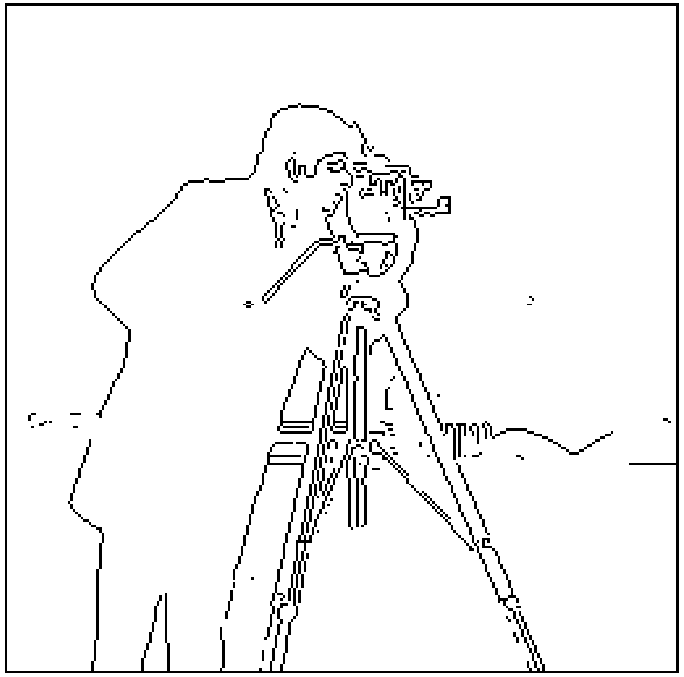
$$S(e) = 1/8\,[\,|\,(a+2b+c)-(g+2h+i)\,|\,]$$
$$+\,|\,(a+2d+g)-(c+2f+i)\,|\,]$$

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

Basic 3x3 mask

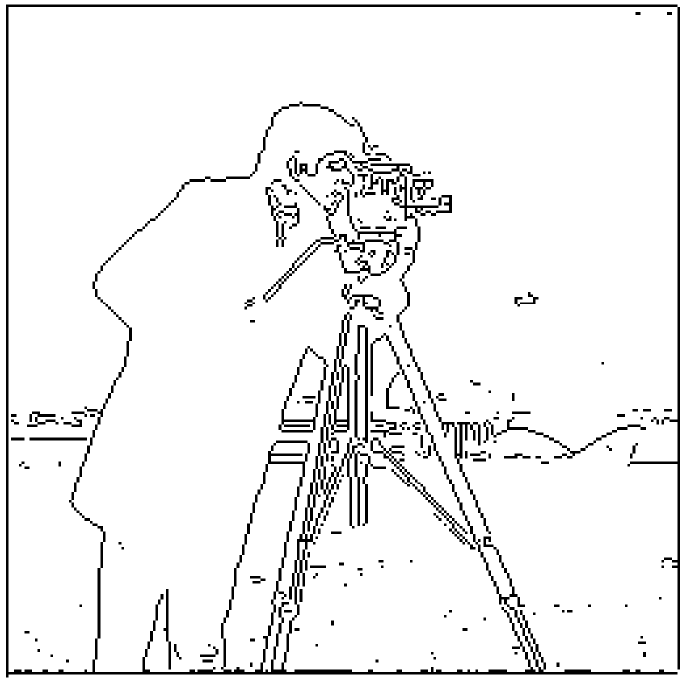
for each pixel. This output is a measure of the edge components passing through the kernel and is independent of both the polarity of the edge and, to a large extent, its orientation

## Edge Filter:  General Sobel



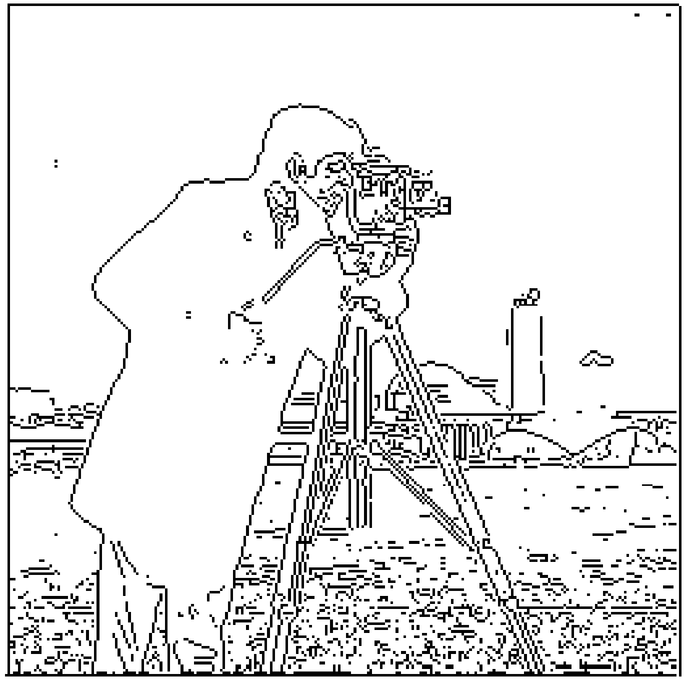
Threshold 1

## Edge Filter:  General Sobel



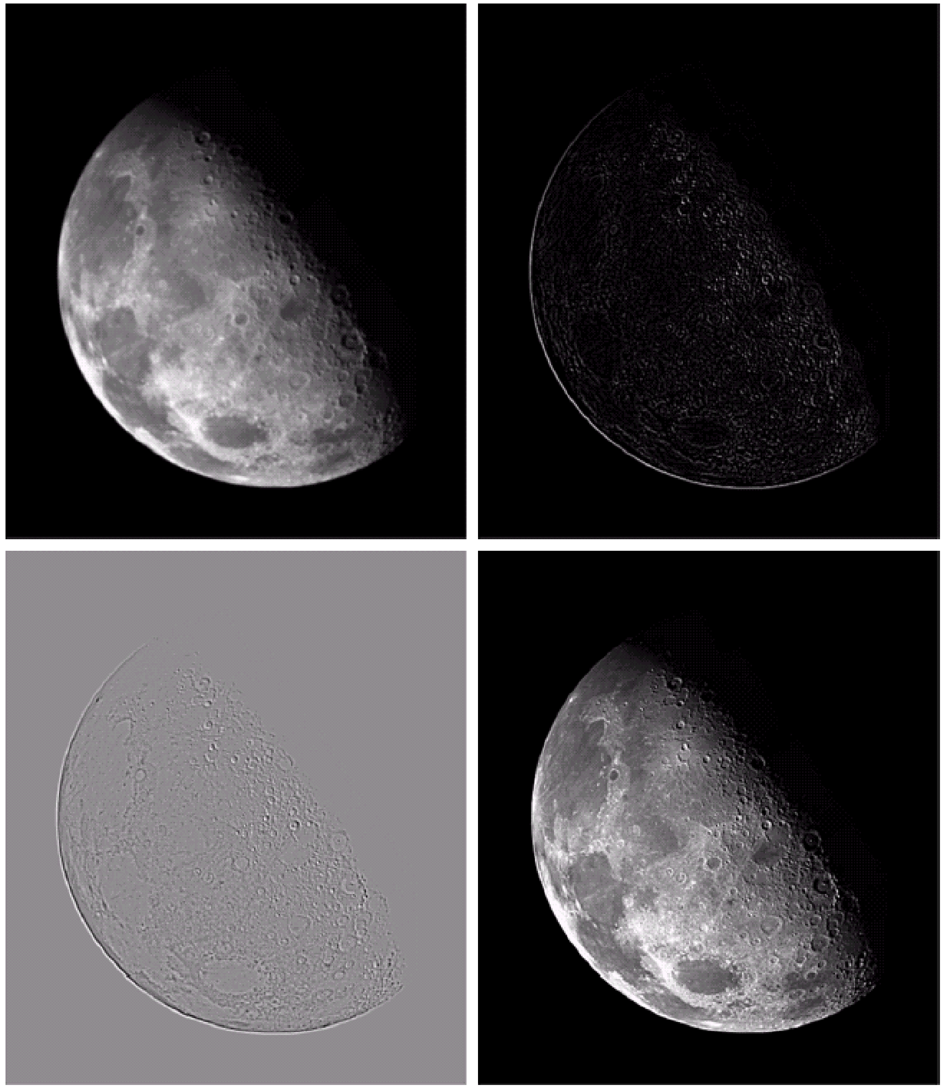
Threshold 2

## Edge Filter:  General Sobel



Threshold 3

# Edge Filter:  Laplacian



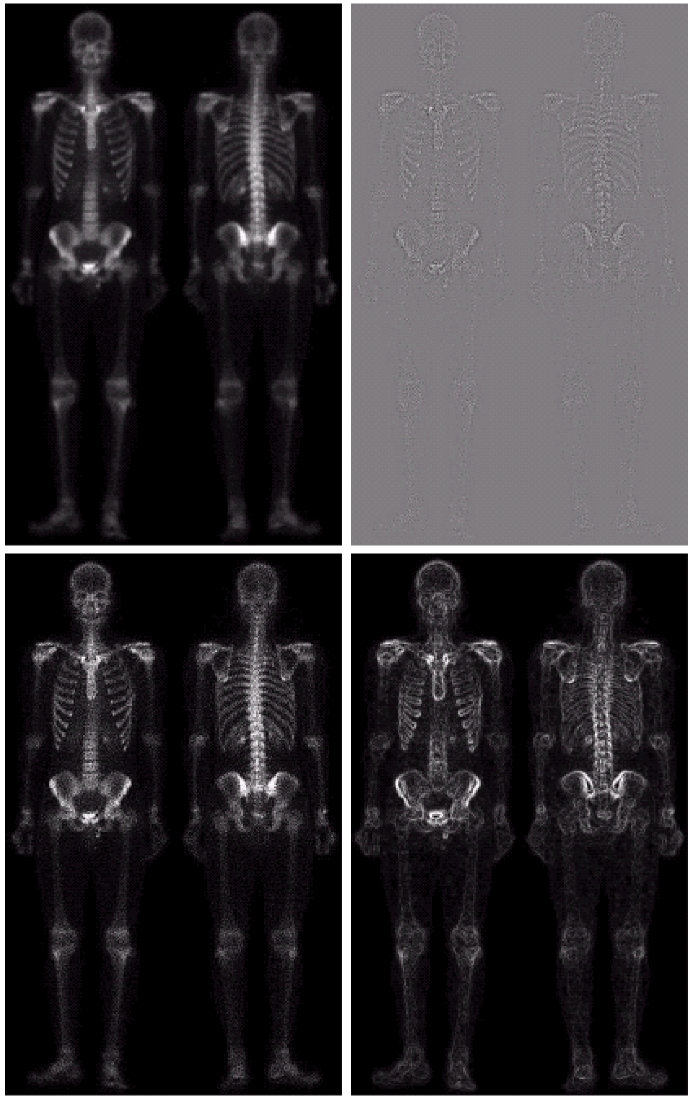
## Edge Filter: Laplacian

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |



## Combining Enhancement Methods

- Attention has been paid to individual approaches
- Frequently, given enhancement requires application of several complementary techniques to achieve acceptable result

## Summary

- Overview provided of common spatial enhancement techniques