DIP

LECTURE No. ( 1 )

# Digital Image Processing

10/14/2019

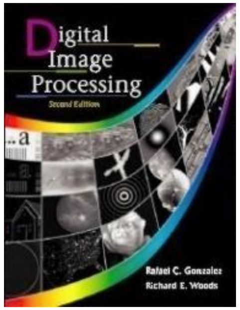
Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

# Introduction

*“One picture is worth more than ten thousand words”*

Anonymous

# References




“Digital Image Processing”, Rafael C. Gonzalez & Richard E. Woods, Addison-Wesley, 2002

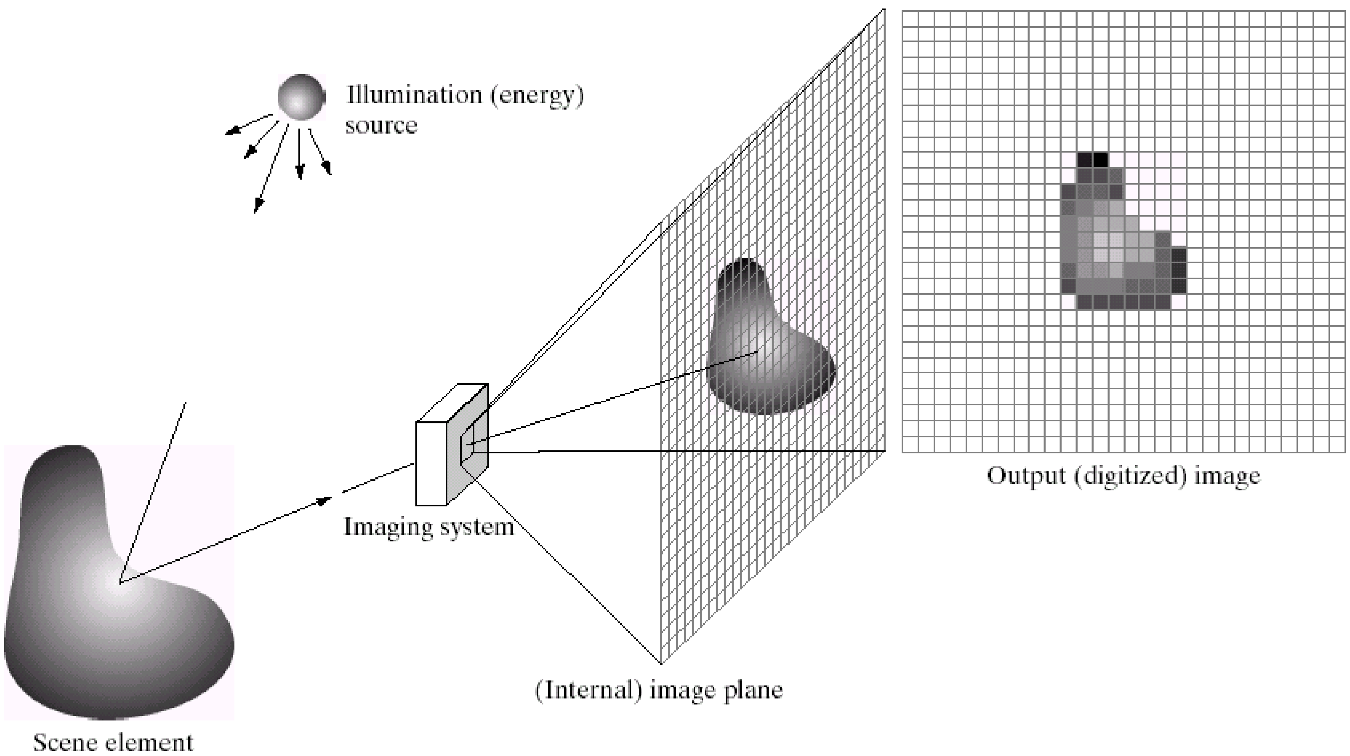
# Contents

This lecture will cover:

- What is a digital image?
- What is digital image processing?
- History of digital image processing
- State of the art examples of digital image processing
- Key stages in digital image processing

# What is a Digital Image?

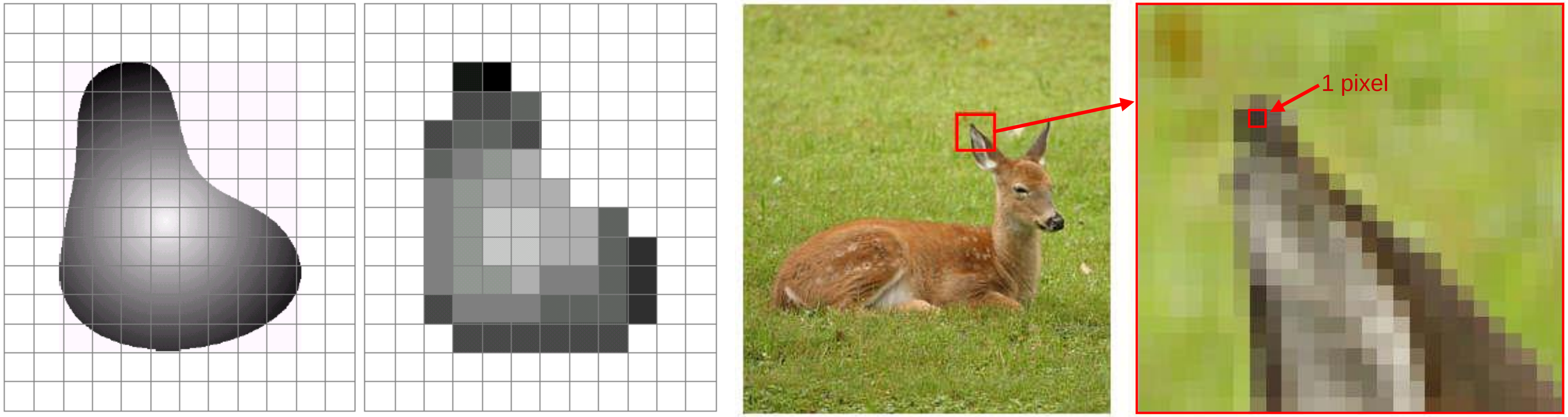
A **digital image** is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels



# What is a Digital Image? (cont…)

Pixel values typically represent gray levels, colours, heights … etc

**Remember** *digitization* implies that a digital image is an *approximation* of a real scene



# What is a Digital Image? (cont…)

Common image formats include:

- 1 sample per point (B&W or Grayscale)
- 3 samples per point (Red, Green, and Blue)
- 4 samples per point (Red, Green, Blue, and “Alpha”, a.k.a. Opacity)

For most of this course we will focus on grey-scale images

# What is Digital Image Processing?

Digital image processing focuses on two major tasks

- Improvement of pictorial information for human interpretation
- Processing of image data for storage, transmission and representation for autonomous machine perception

Some argument about where image processing ends and fields such as image analysis and computer vision start

# What is DIP? (cont…)

The continuum from image processing to computer vision can be broken up into low-, mid- and high-level processes

| Low Level Process | Mid Level Process | High Level Process |
|---|---|---|
| **Input:** Image<br>**Output:** Image<br>**Examples:** Noise removal, image sharpening | **Input:** Image<br>**Output:** Attributes<br>**Examples:** Object recognition, segmentation | **Input:** Attributes<br>**Output:** Understanding<br>**Examples:** Scene understanding, autonomous navigation |

In this course we will
stop here

# History of Digital Image Processing

**Early 1920s:** One of the first applications of digital imaging was in the news-paper industry

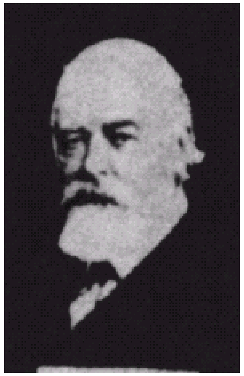
Early digital image

- The Bartlane cable picture transmission service
- Images were transferred by submarine cable between London and New York
- Pictures were coded for cable transfer and reconstructed at the receiving end on a telegraph printer

# History of DIP (cont…)

**Mid to late 1920s:** Improvements to the Bartlane system resulted in higher quality images

- New reproduction processes based on photographic techniques
- Increased number of tones in reproduced images



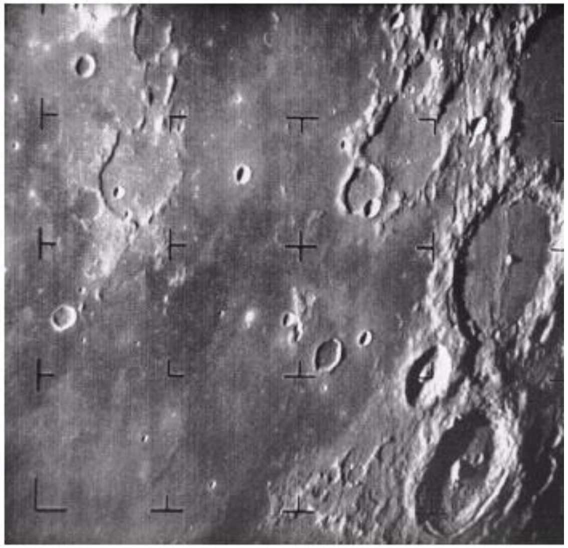
Improved digital image

Early 15 tone digital image

# History of DIP (cont…)

**1960s:** Improvements in computing technology and the onset of the space race led to a surge of work in digital image processing

– **1964:** Computers used to improve the quality of images of the moon taken by the *Ranger 7* probe
– Such techniques were used in other space missions including the Apollo landings

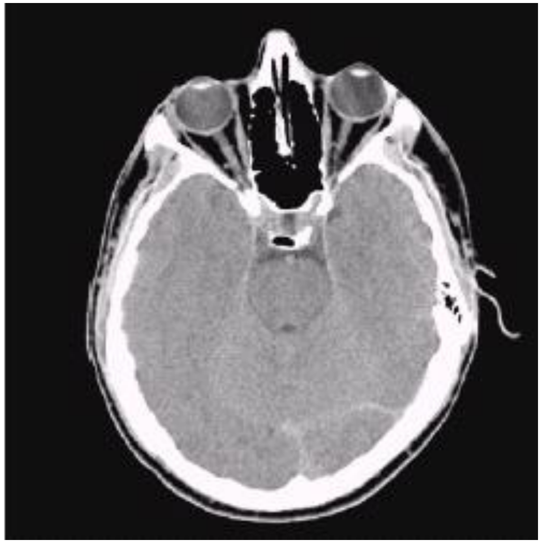


A picture of the moon taken by the Ranger 7 probe minutes before landing

# History of DIP (cont…)

**1970s:** Digital image processing begins to be used in medical applications

- **1979:** Sir Godfrey N. Hounsfield & Prof. Allan M. Cormack share the Nobel Prize in medicine for the invention of tomography, the technology behind Computerised Axial Tomography (CAT) scans



Typical head slice CAT image

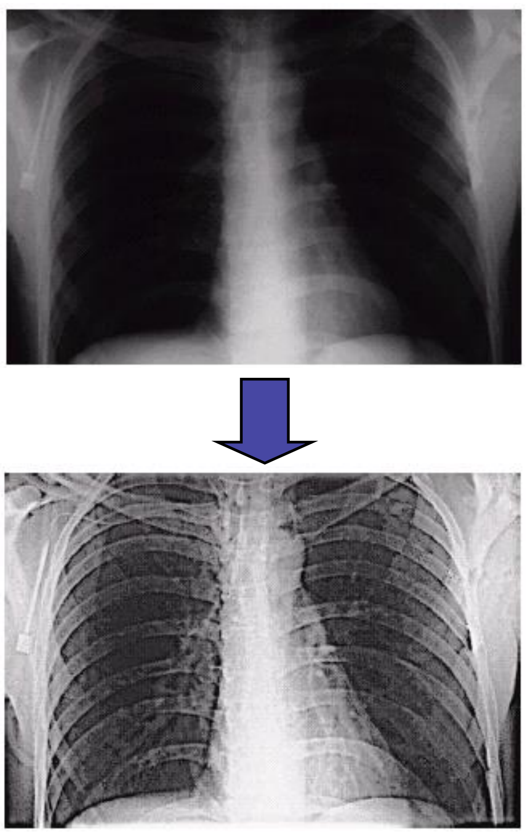
# History of DIP (cont…)

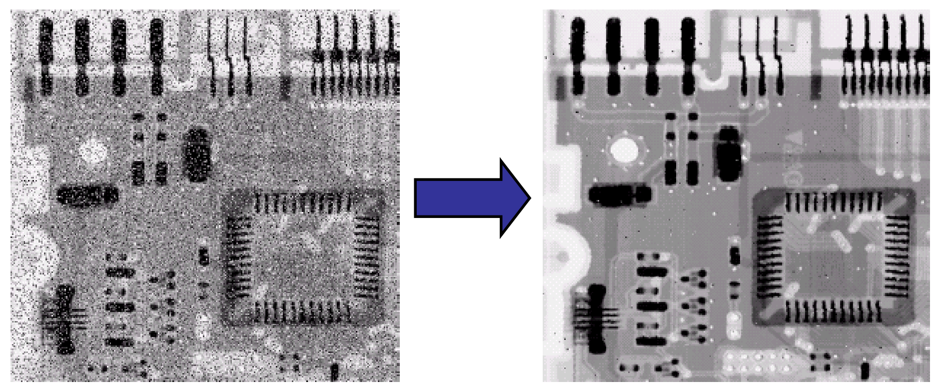
**1980s - Today:** The use of digital image processing techniques has exploded and they are now used for all kinds of tasks in all kinds of areas

- Image enhancement/restoration
- Artistic effects
- Medical visualisation
- Industrial inspection
- Law enforcement
- Human computer interfaces

# Examples: Image Enhancement

One of the most common uses of DIP techniques: improve quality, remove noise etc





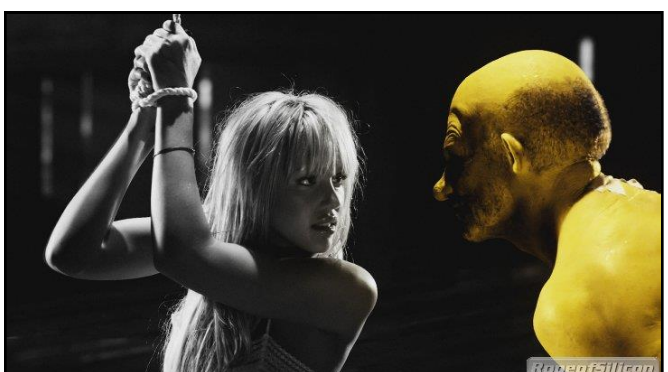
# Examples: The Hubble Telescope

- Launched in 1990 the Hubble telescope can take images of very distant objects
- However, an incorrect mirror made many of Hubble's images useless
- Image processing techniques were used to fix this

Wide Field Planetary Camera 1 Wide Field Planetary Camera 2

# Examples: Artistic Effects

Artistic effects are used to make images more visually appealing, to add special effects and to make composite images

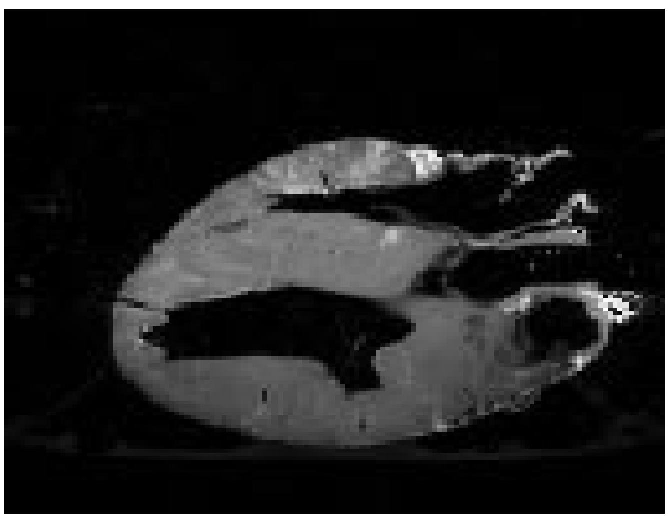


# Examples: Medicine

Take slice from MRI scan of canine heart, and find boundaries between types of tissue

– Image with gray levels representing tissue density

– Use a suitable filter to highlight edges



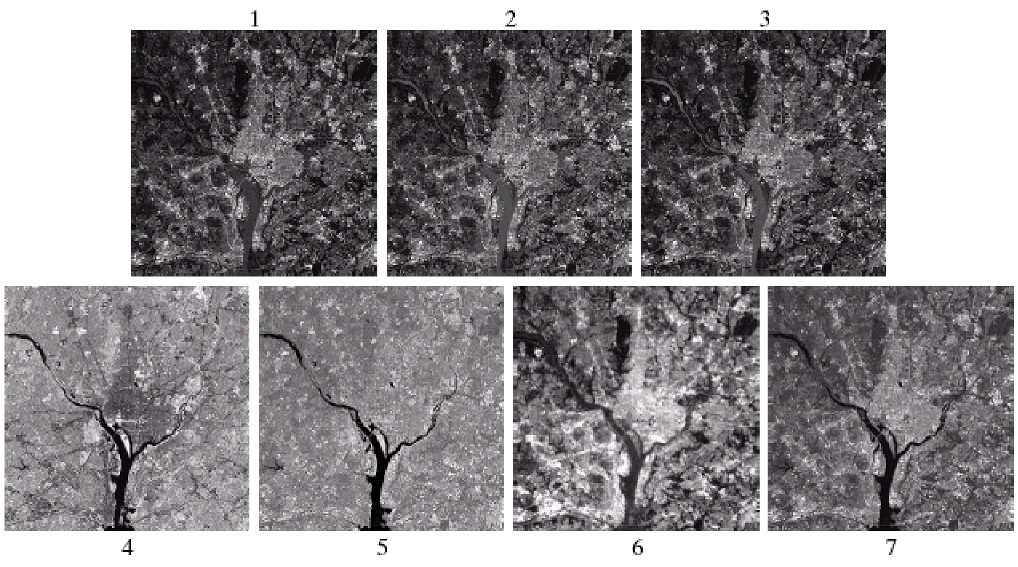
Original MRI Image of a Dog Heart

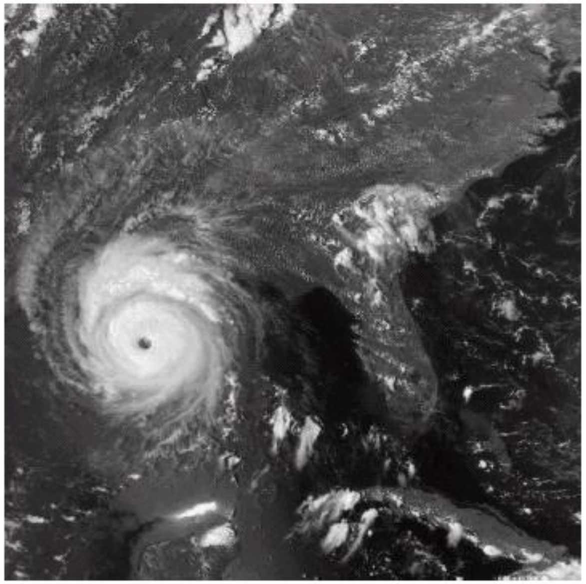
Edge Detection Image

# Examples: GIS

Geographic Information Systems

- Digital image processing techniques are used extensively to manipulate satellite imagery
- Terrain classification
- Meteorology

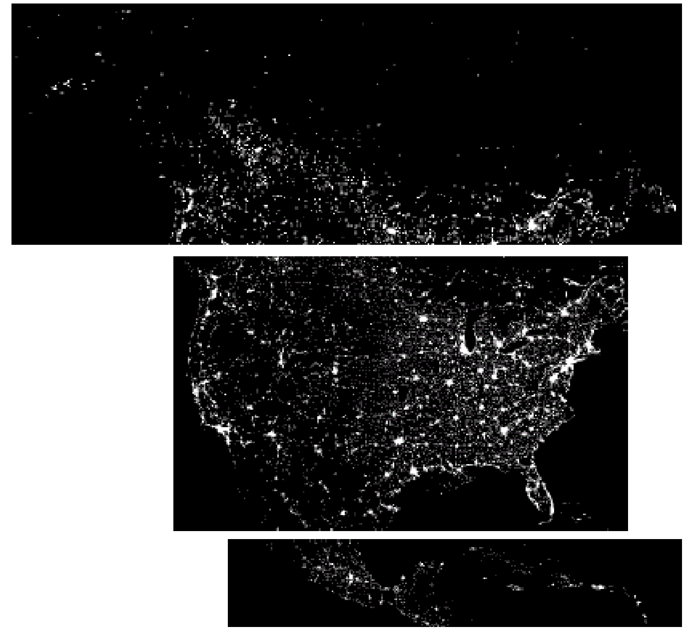


# Examples: GIS (cont…)

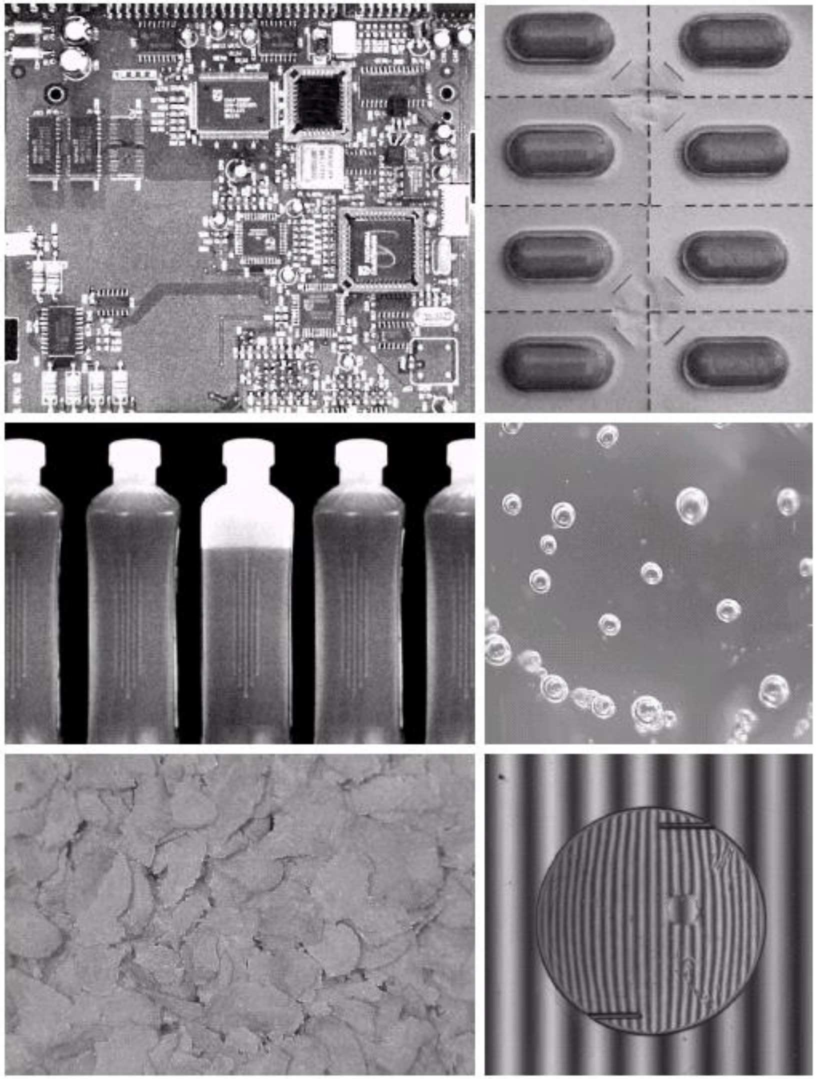
*Night-Time Lights of the World* data set

- Global inventory of human settlement
- Not hard to imagine the kind of analysis that might be done using this data



# Examples: Industrial Inspection

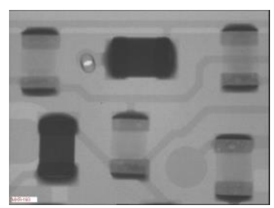
- Human operators are expensive, slow and unreliable
- Make machines do the job instead
- Industrial vision systems are used in all kinds of industries
- Can we trust them?

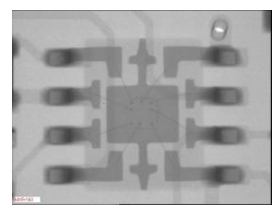


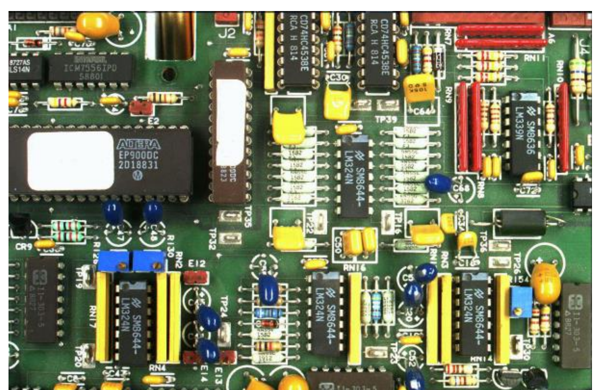
# Examples: PCB Inspection

Printed Circuit Board (PCB) inspection

- Machine inspection is used to determine that all components are present and that all solder joints are acceptable
- Both conventional imaging and x-ray imaging are used





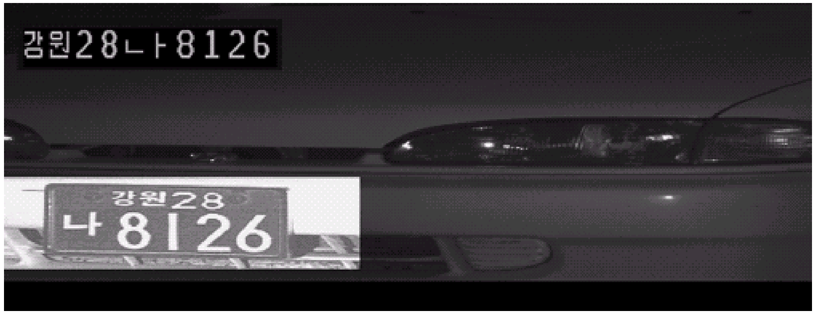
# Examples: Law Enforcement

Image processing techniques are used extensively by law enforcers

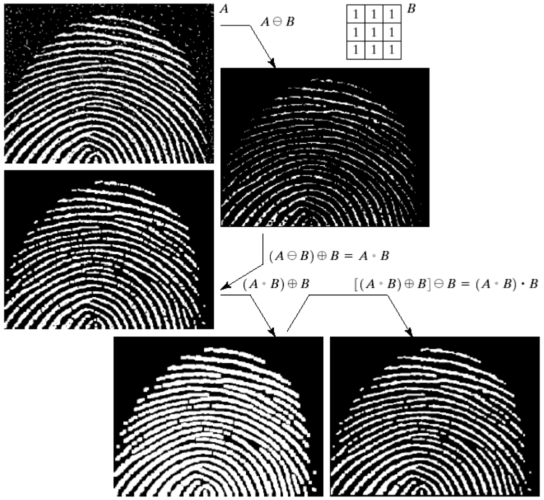
– Number plate recognition for speed cameras/automated toll systems

– Fingerprint recognition

– Enhancement of CCTV images

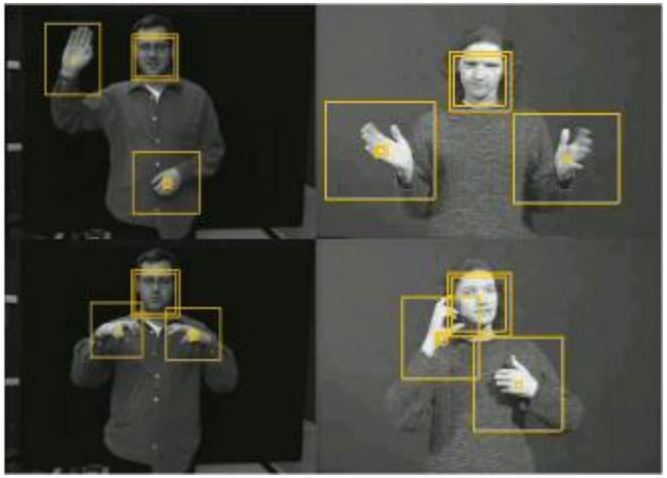


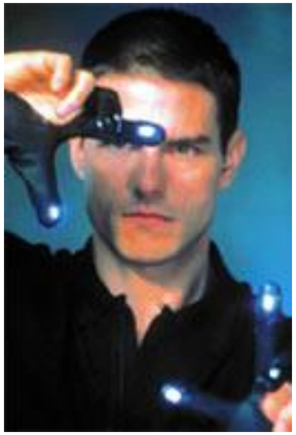


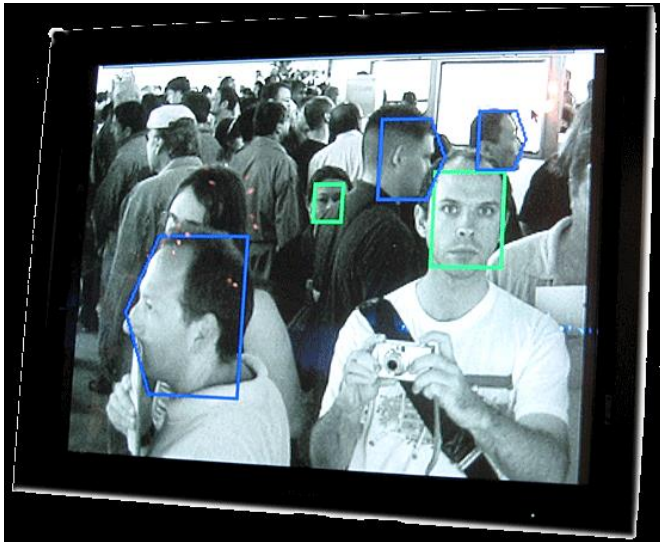


# Examples: HCI

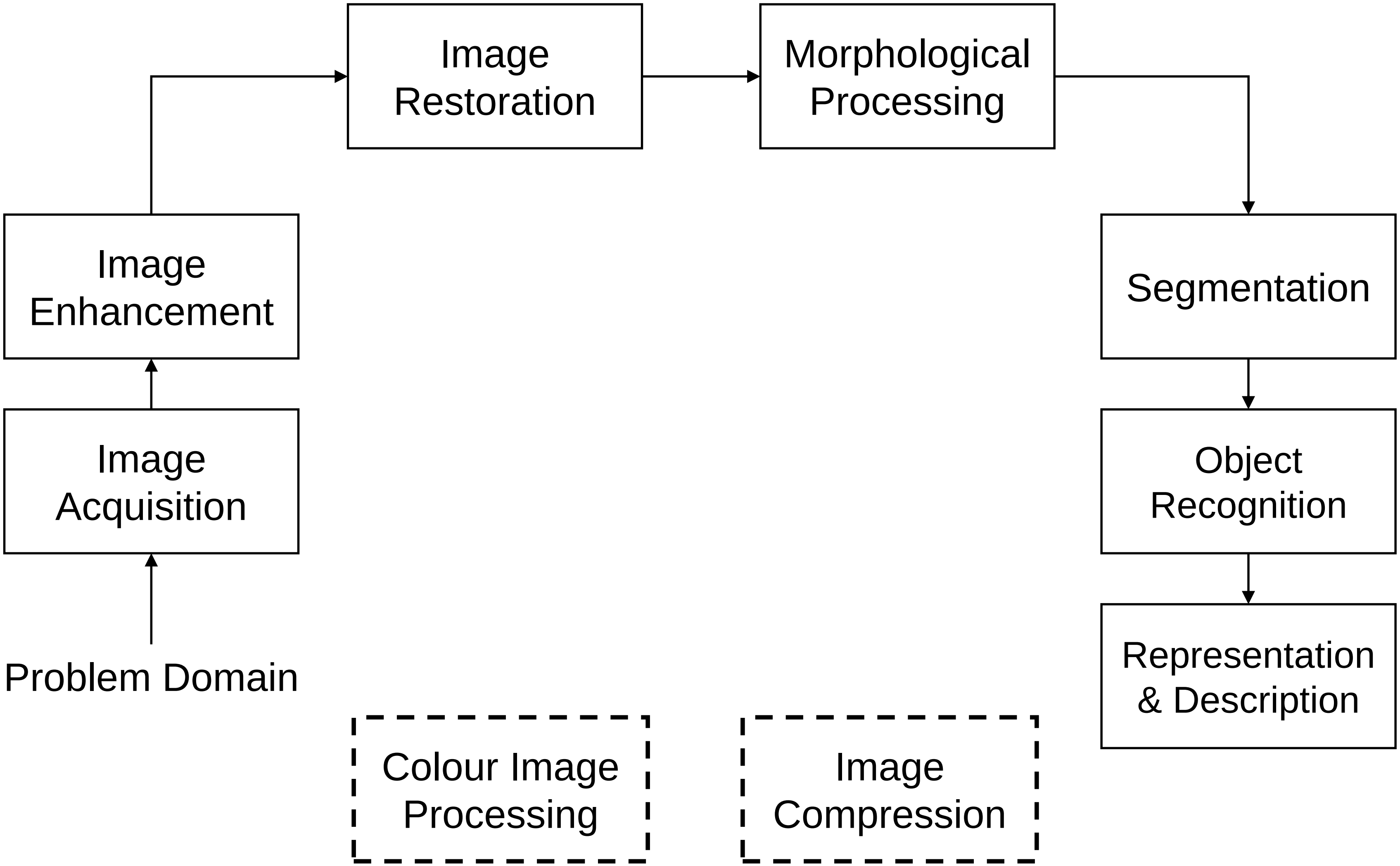
- Try to make human computer interfaces more natural
  - – Face recognition
  - – Gesture recognition
- Does anyone remember the user interface from “Minority Report”?
- These tasks can be extremely difficult







# Key Stages in Digital Image Processing



# Key Stages in Digital Image Processing: Image Aquisition

Image Restoration

Morphological Processing

Image Enhancement

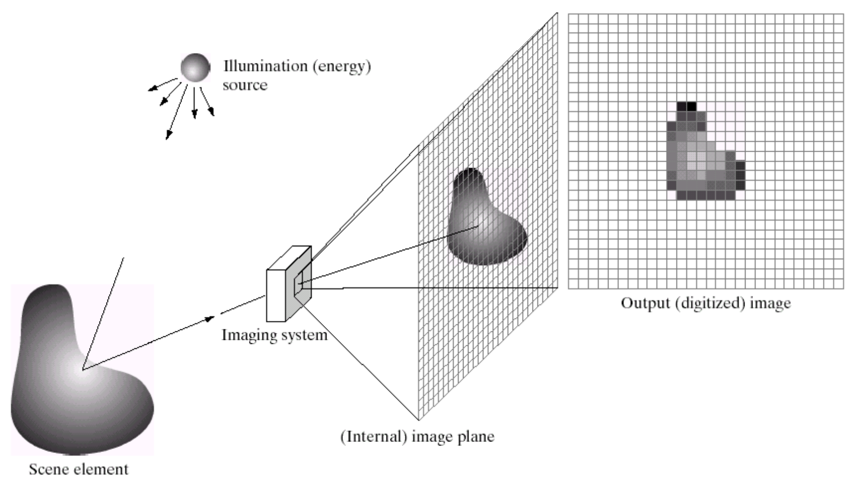
Segmentation

Image Acquisition

Object Recognition

Representation & Description

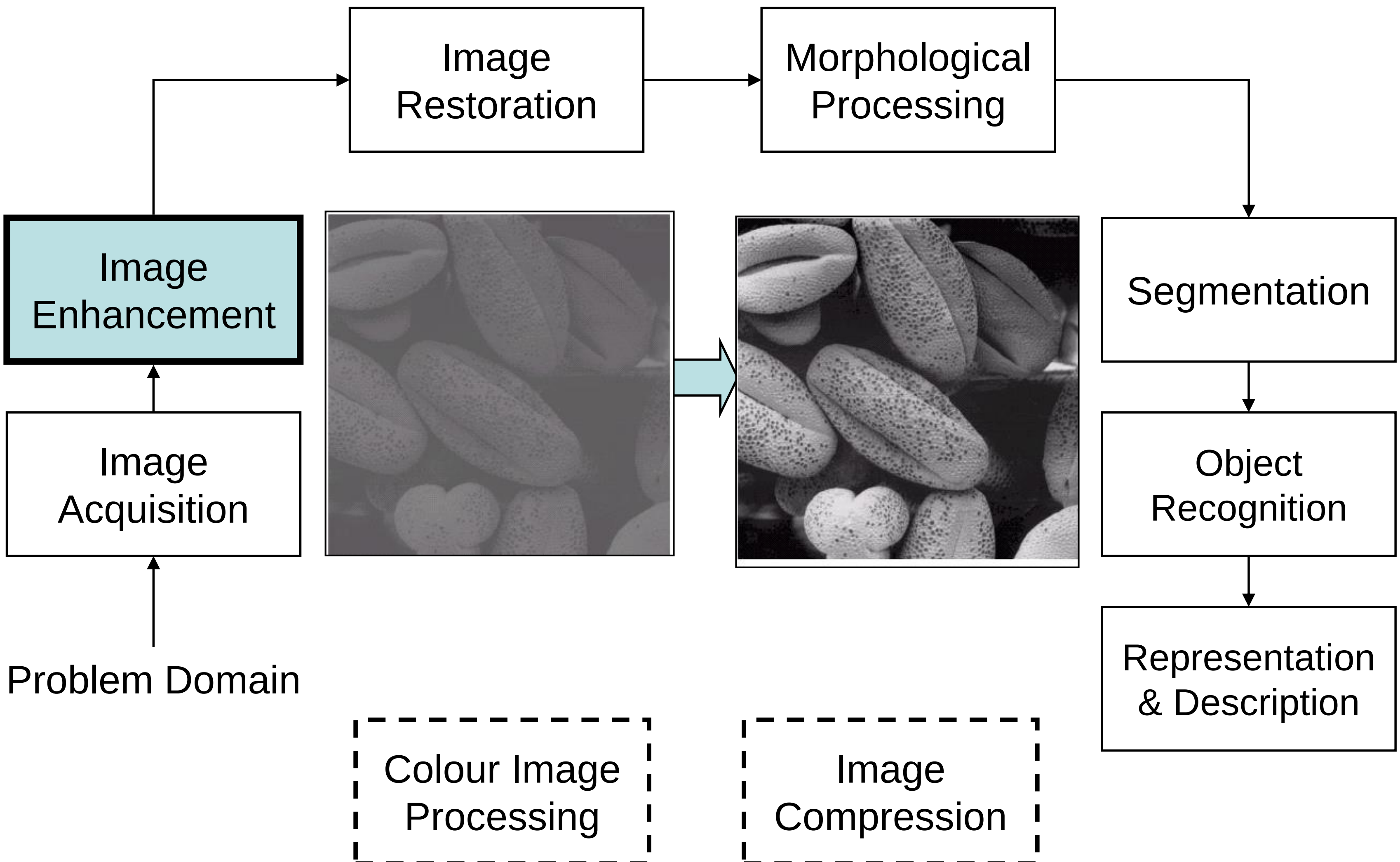
Problem Domain



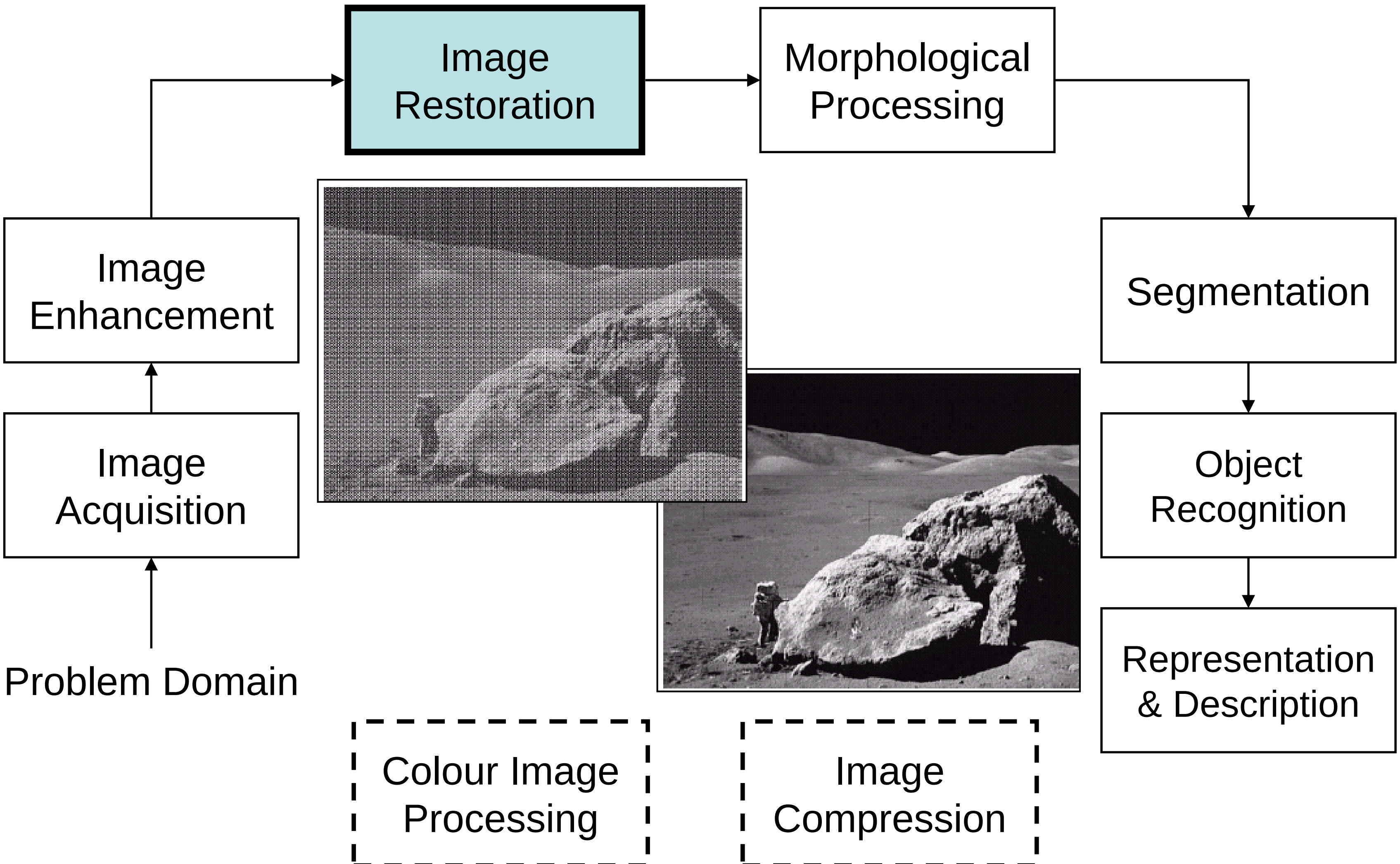

Colour Image Processing

Image Compression

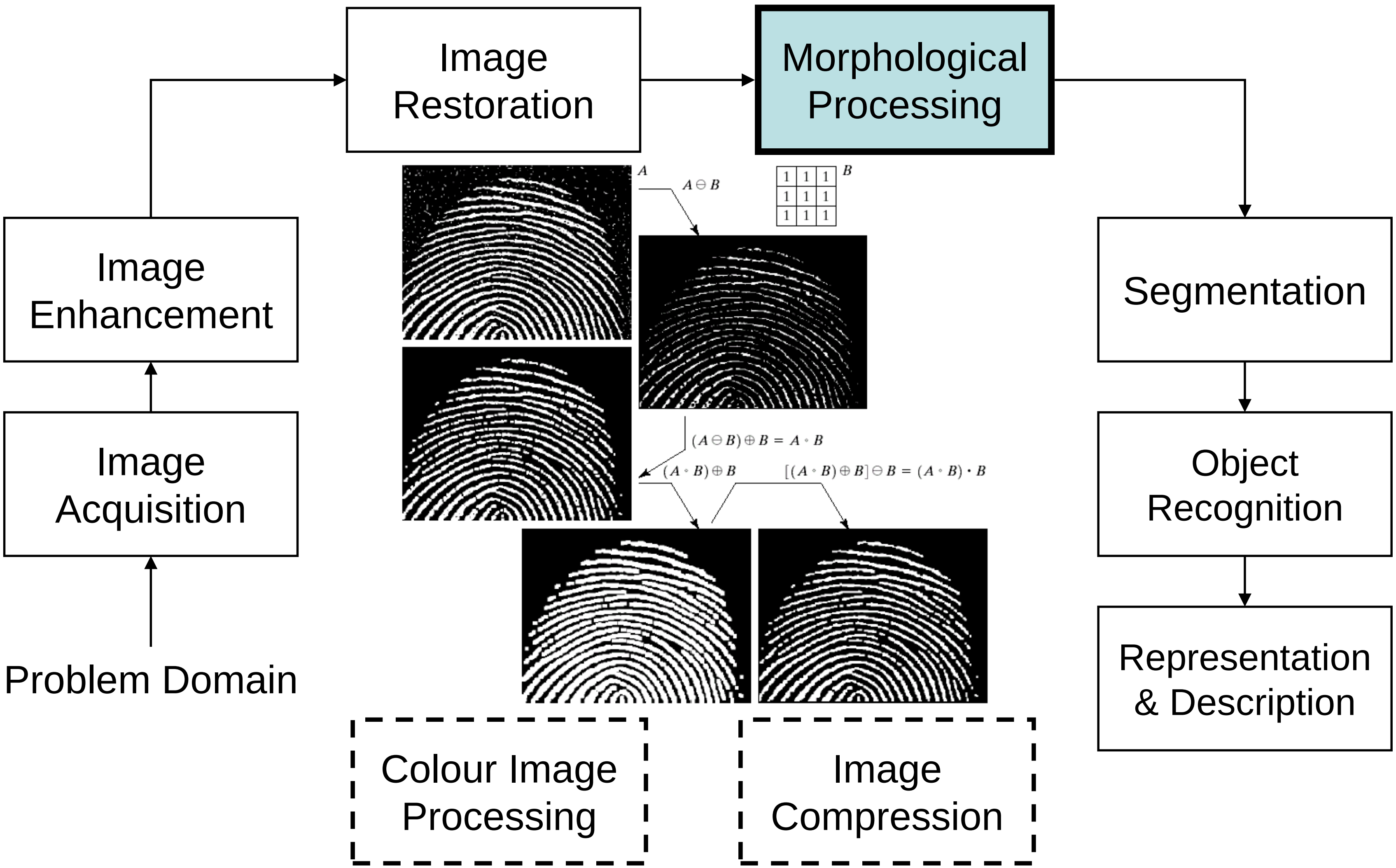
# Key Stages in Digital Image Processing: Image Enhancement



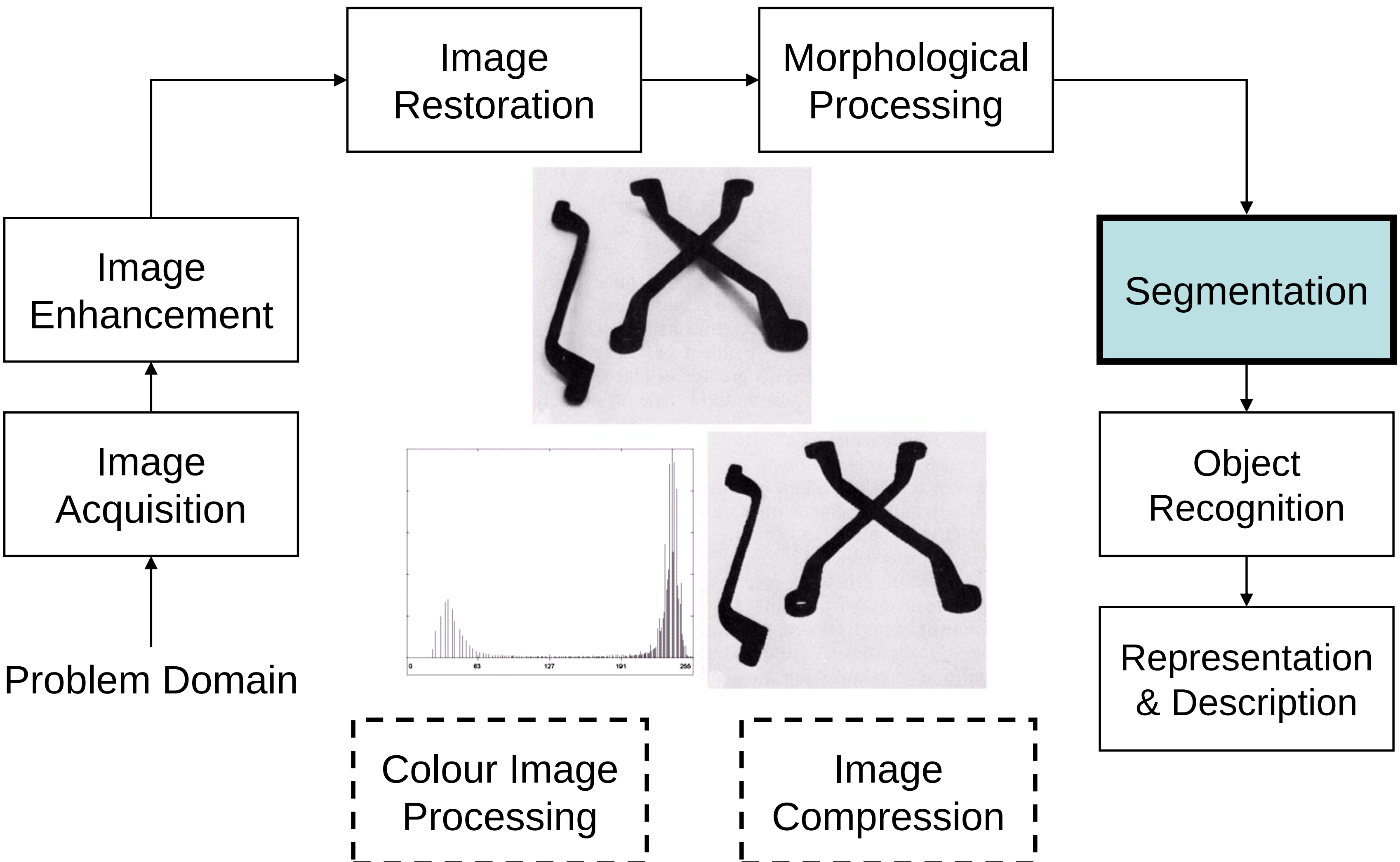
# Key Stages in Digital Image Processing: Image Restoration



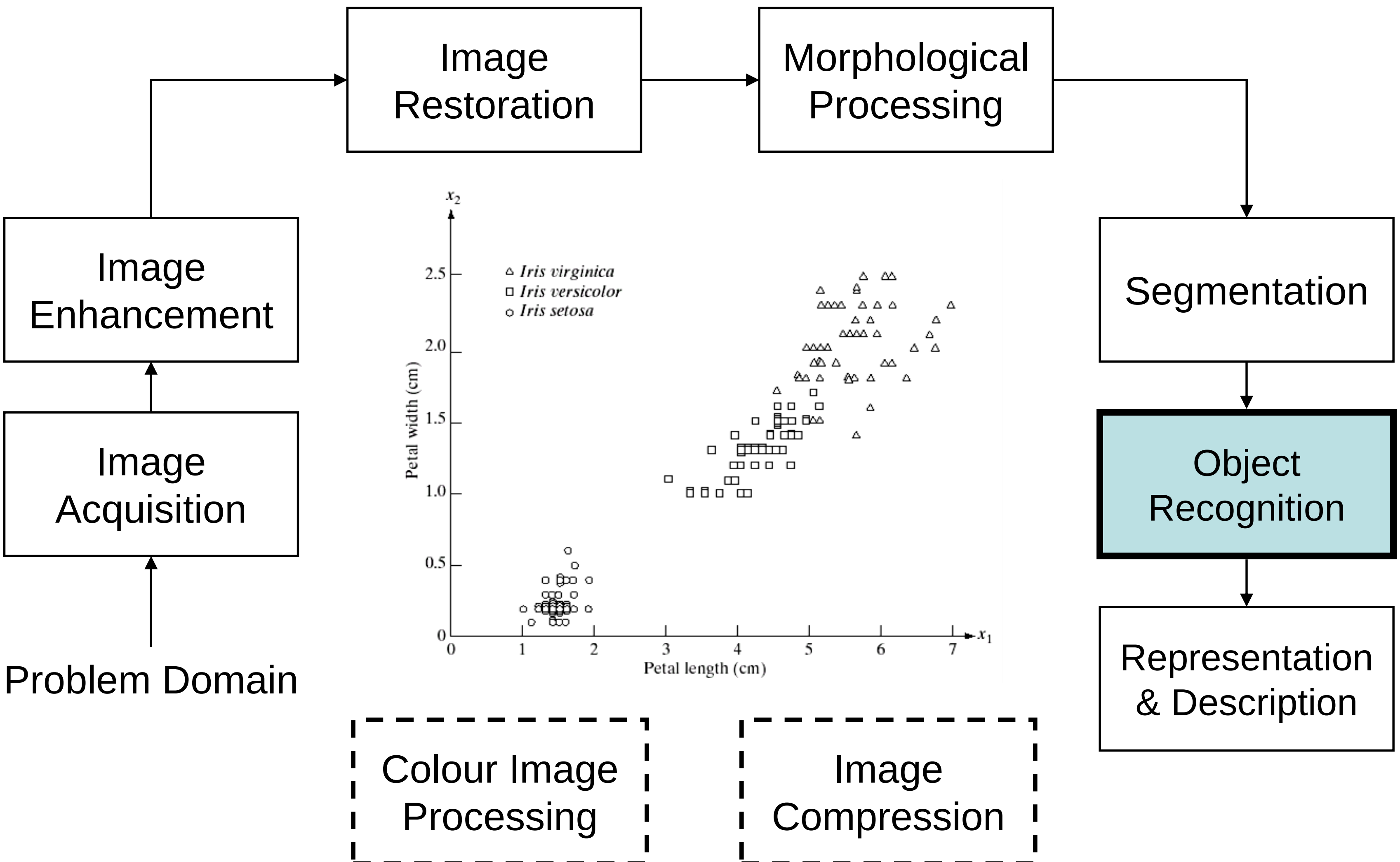
# Key Stages in Digital Image Processing: Morphological Processing



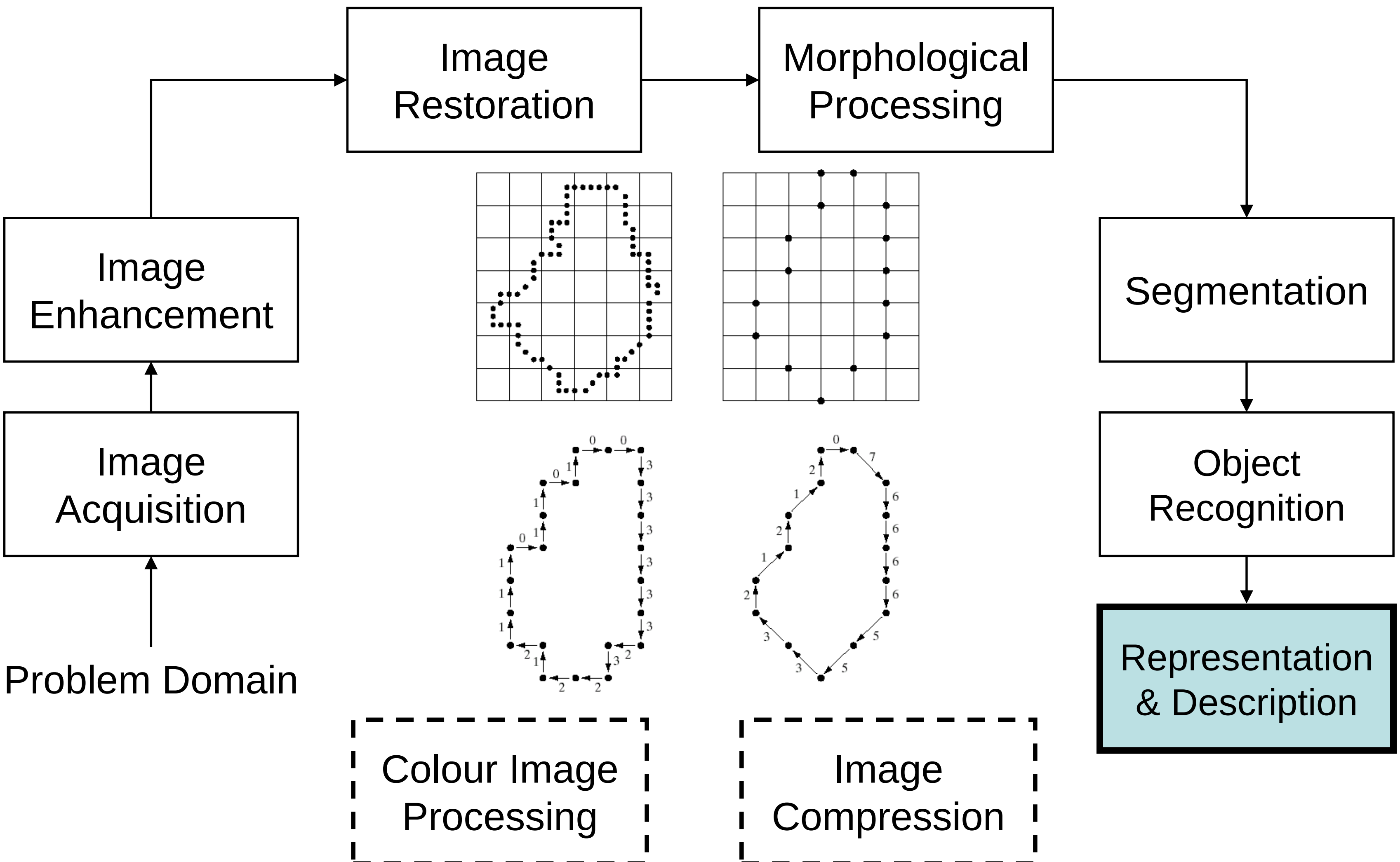
# Key Stages in Digital Image Processing: Segmentation



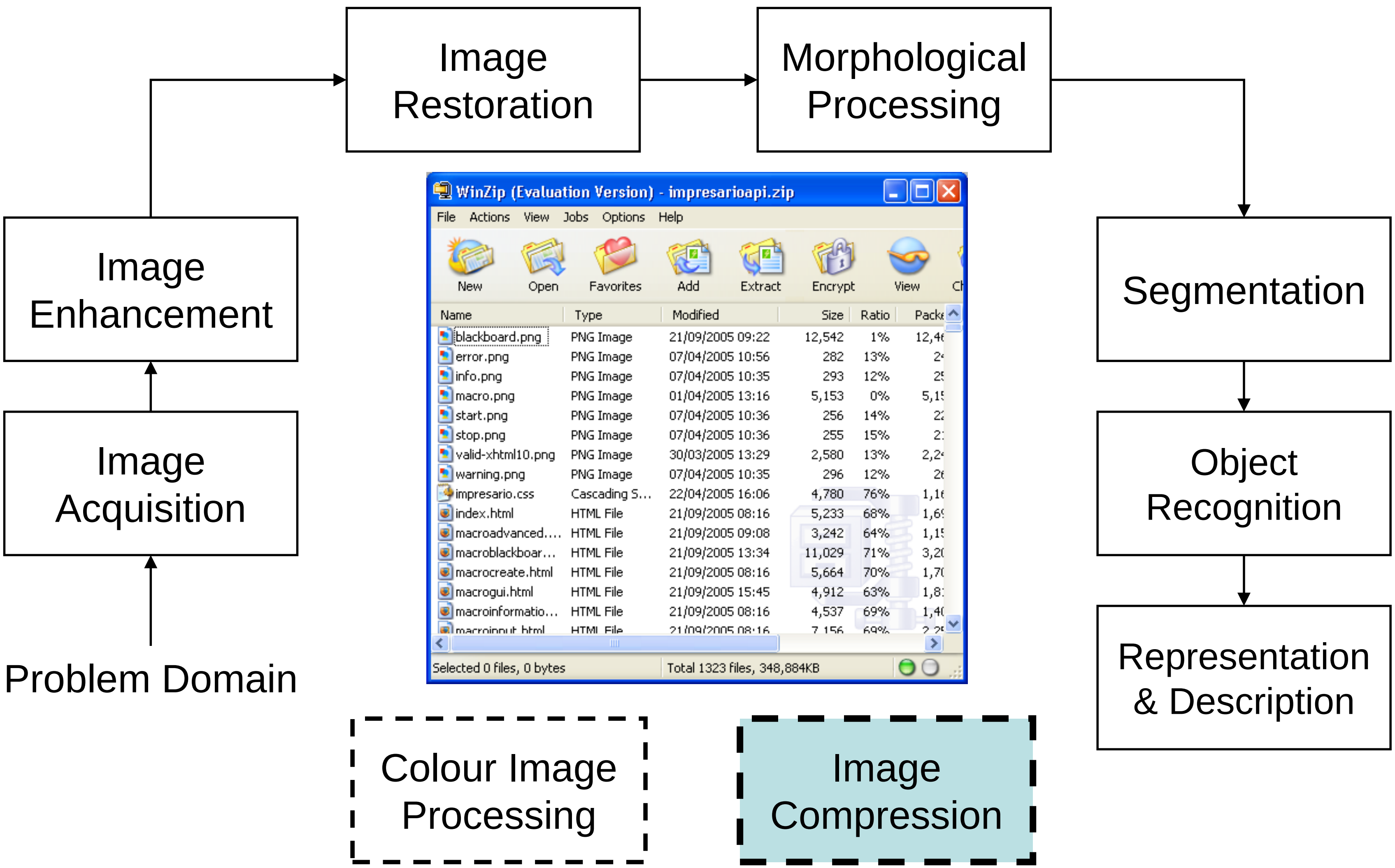
# Key Stages in Digital Image Processing: Object Recognition



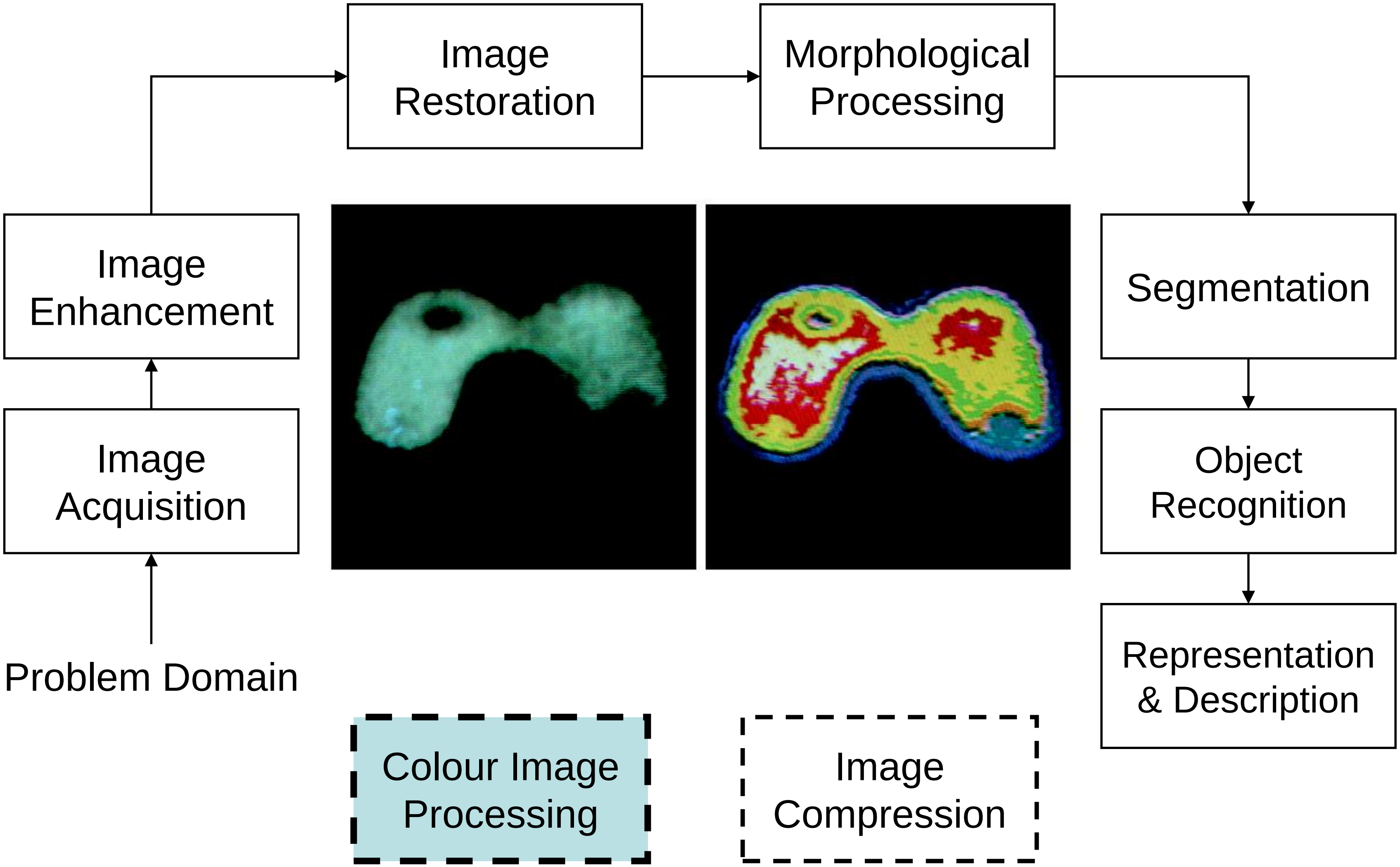
# Key Stages in Digital Image Processing: Representation & Description



# Key Stages in Digital Image Processing: Image Compression



# Key Stages in Digital Image Processing: Colour Image Processing



# Summary

We have looked at:

- What is a digital image?
- What is digital image processing?
- History of digital image processing
- State of the art examples of digital image processing
- Key stages in digital image processing

Next time we will start to see how it all works…

DIP

LECTURE No. ( 2 )

# *Digital Image Processing*

# *Digital Imaging Fundamentals*

10/14/2019

Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

# Contents

This lecture will cover:

– The human visual system
– Light and the electromagnetic spectrum
– Image representation
– Image sensing and acquisition
– Sampling, quantisation and resolution
– Basic Relation

# Human Visual System

The best vision model we have!

Knowledge of how images form in the eye can help us with processing digital images

We will take just a whirlwind tour of the human visual system

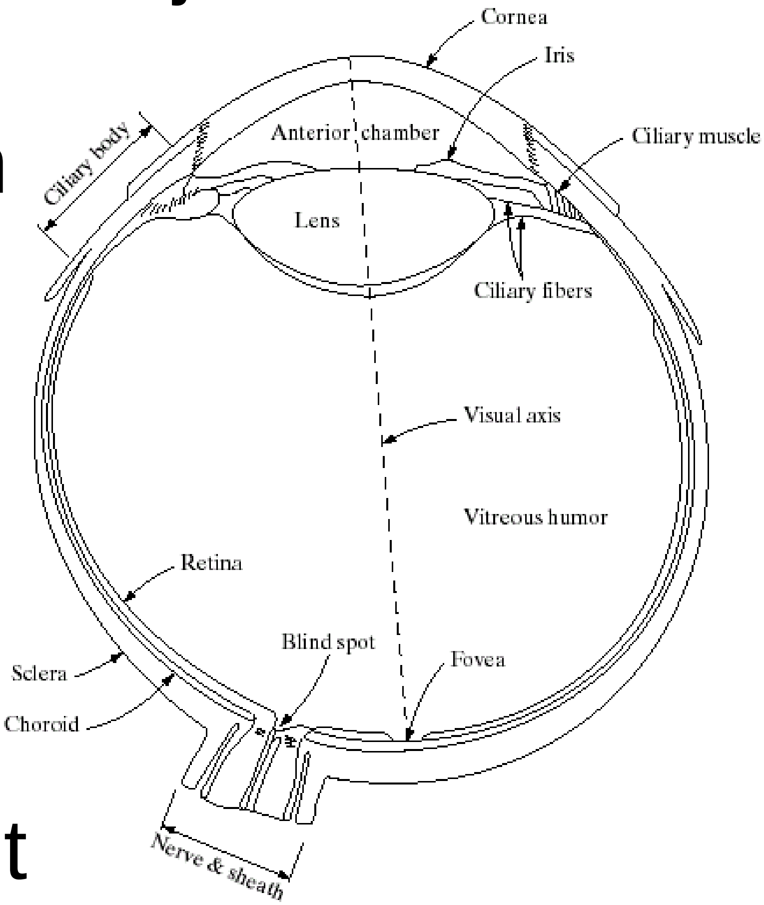
# Structure Of The Human Eye

The lens focuses light from objects onto the retina

The retina is covered with light receptors called *cones* (6-7 million) and *rods* (75-150 million)

Cones are concentrated around the fovea and are very sensitive to colour

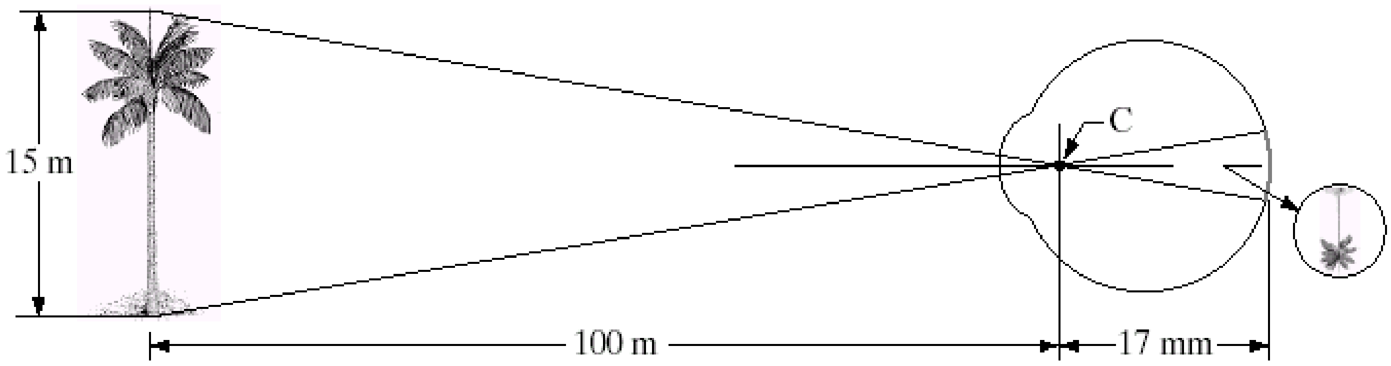
Rods are more spread out and are sensitive to low levels of illumination



# Image Formation In The Eye

Muscles within the eye can be used to change the shape of the lens allowing us focus on objects that are near or far away

An image is focused onto the retina causing rods and cones to become excited which ultimately send signals to the brain



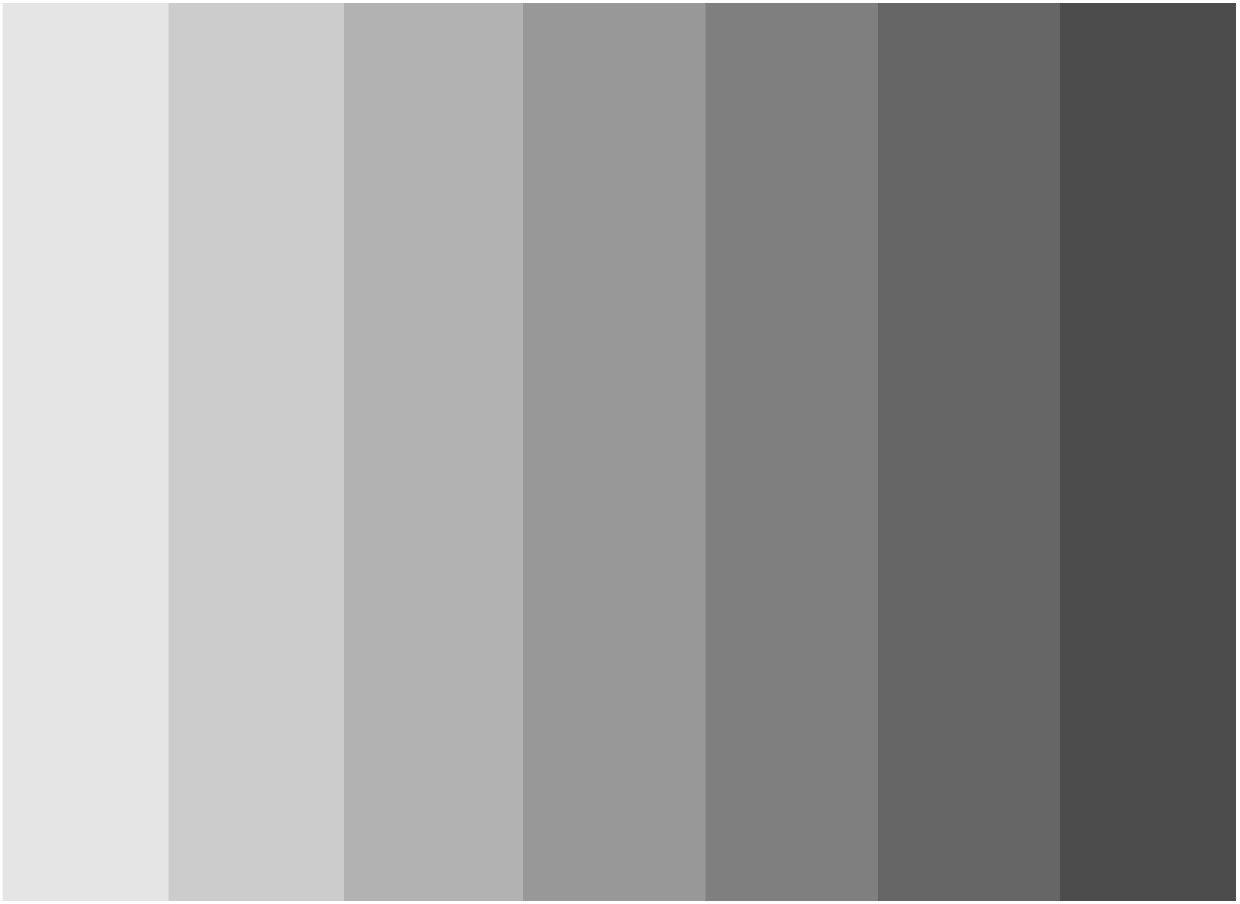
# Brightness Adaptation & Discrimination

The human visual system can perceive approximately $10^{10}$ different light intensity levels

However, at any one time we can only discriminate between a much smaller number – *brightness adaptation*

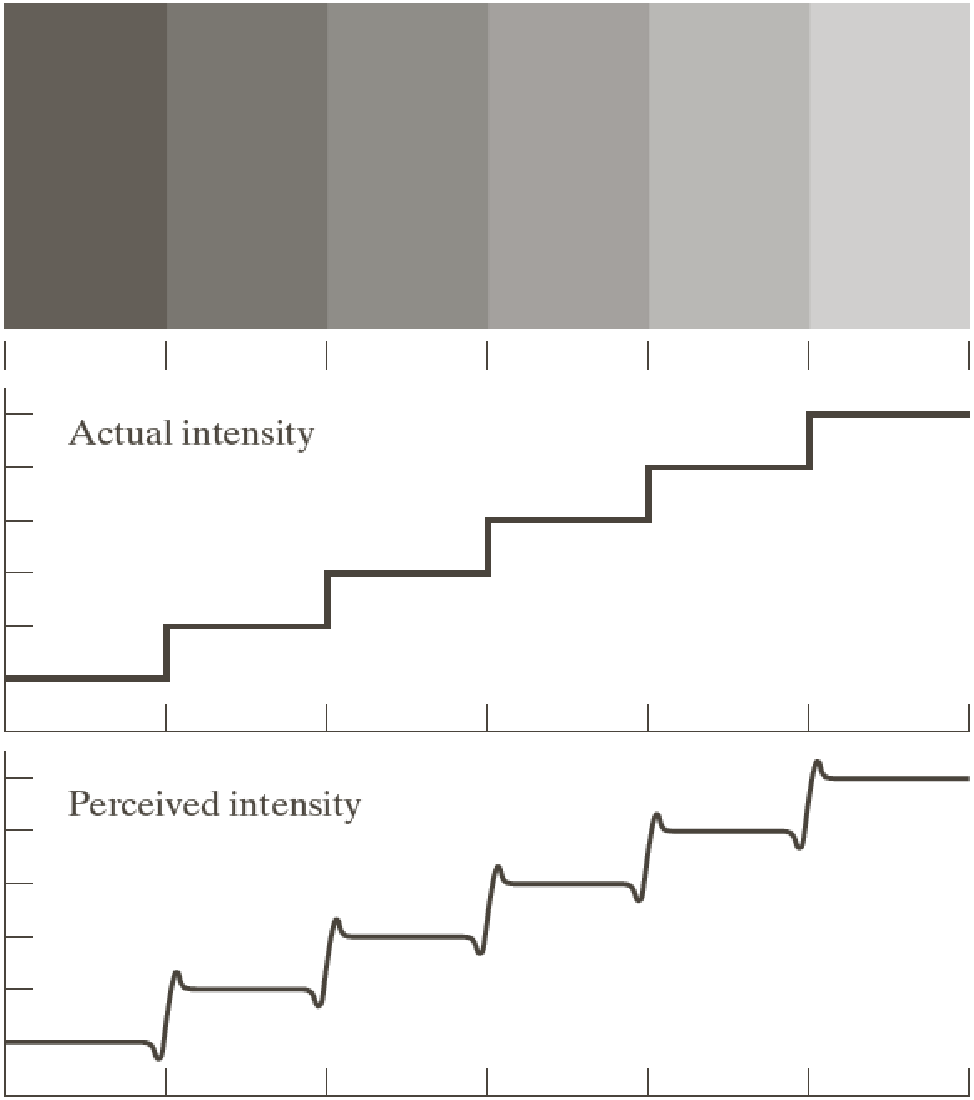
Similarly, the *perceived intensity* of a region is related to the light intensities of the regions surrounding it

# Brightness Adaptation & Discrimination (cont…)

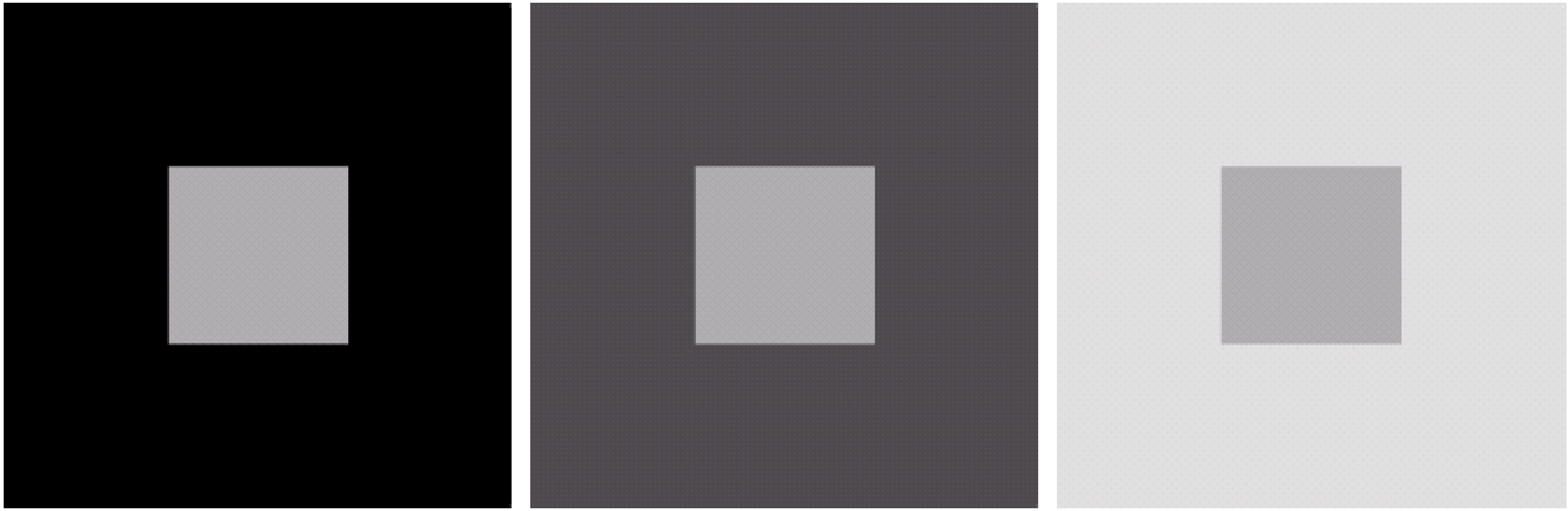


An example of Mach bands

# Brightness Adaptation & Discrimination (cont…)

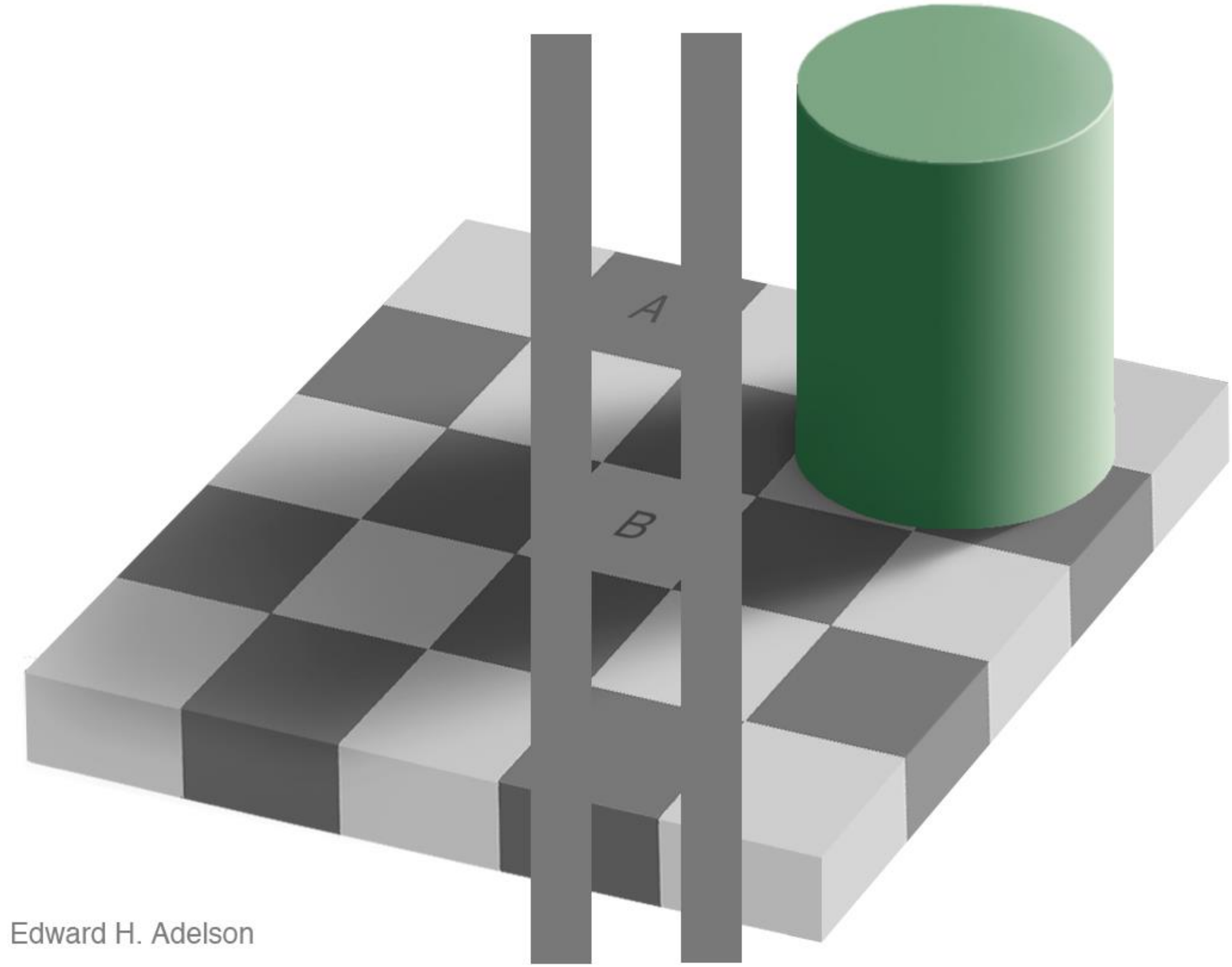


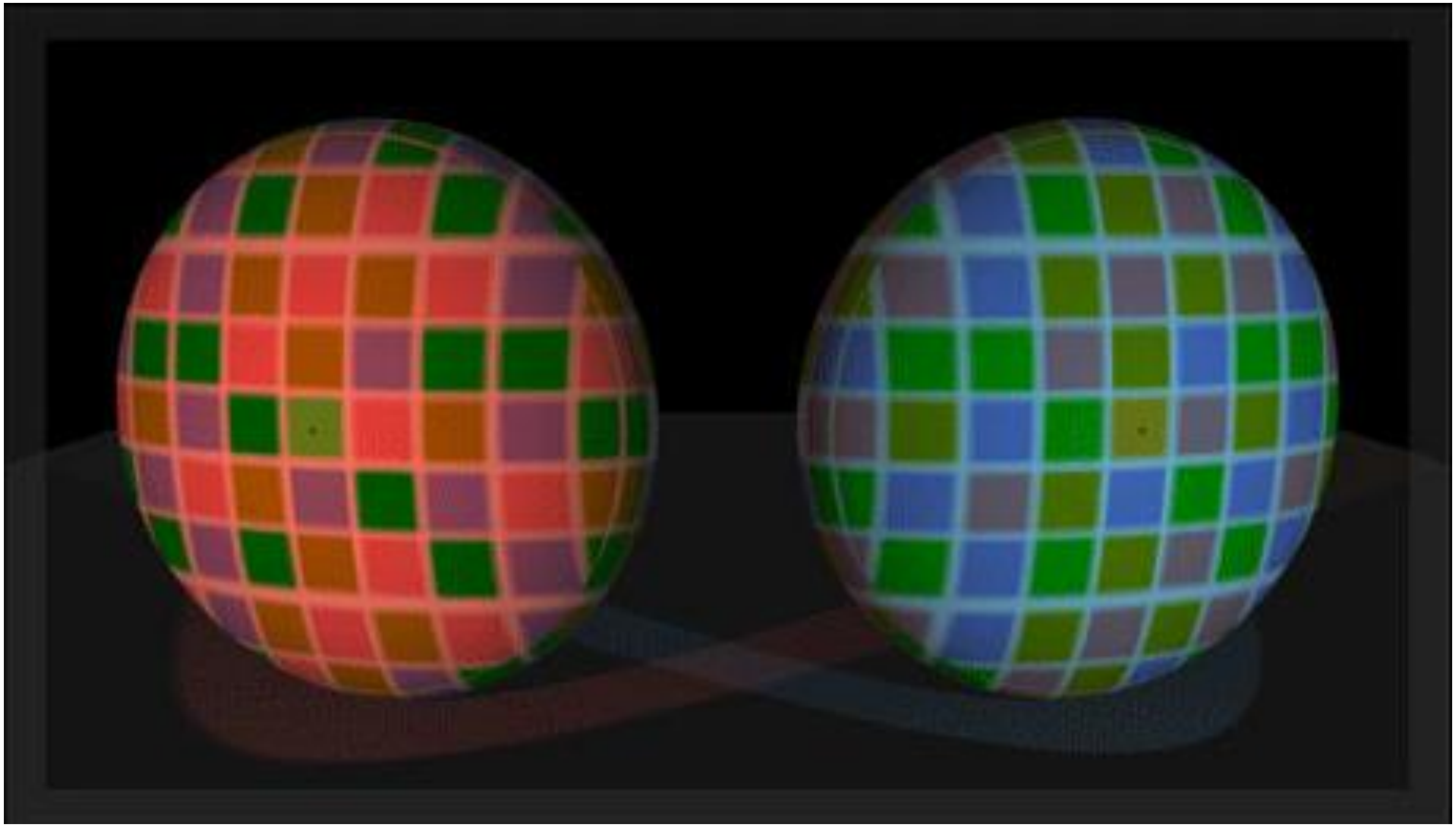
# Brightness Adaptation & Discrimination (cont…)



An example of *simultaneous contrast*

# Brightness Adaptation & Discrimination (cont…)

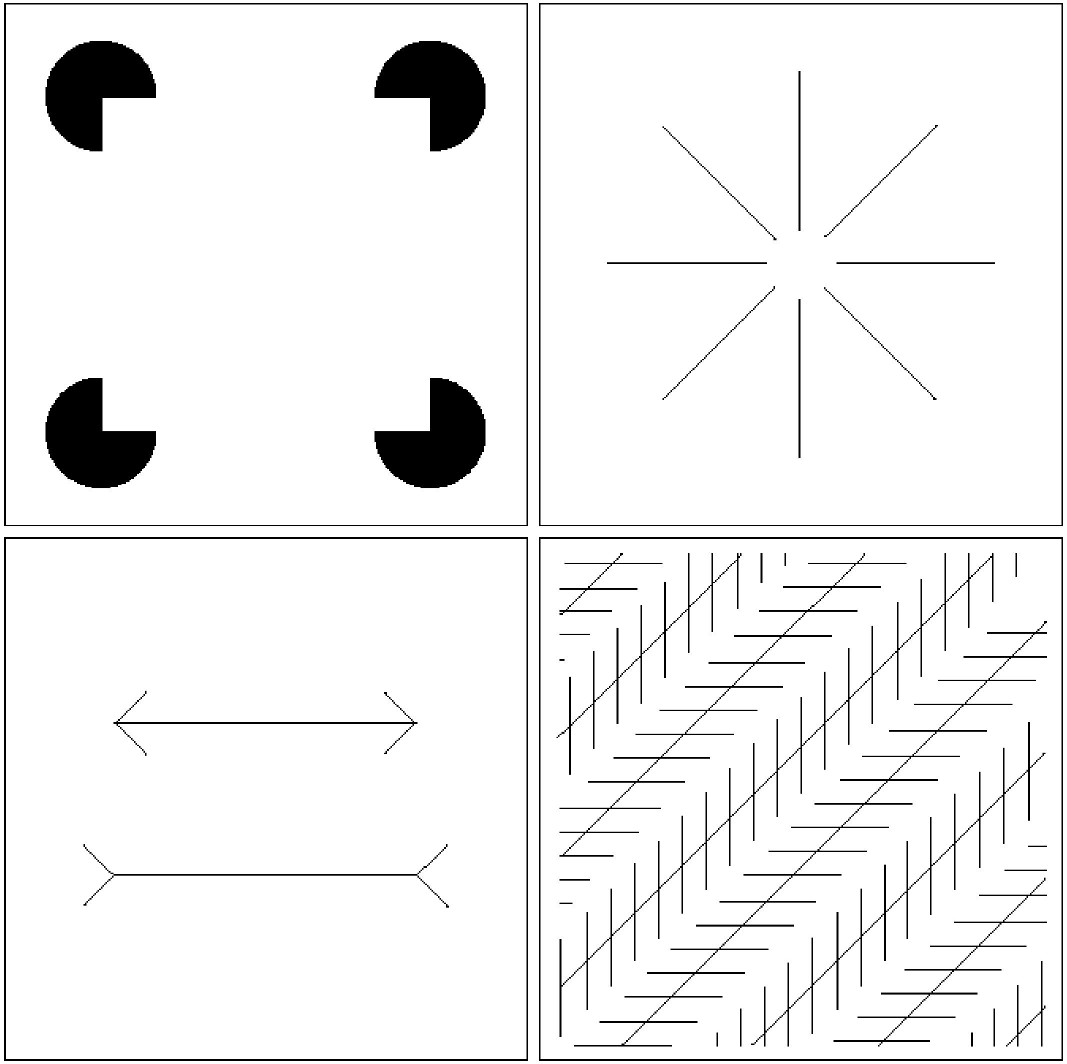




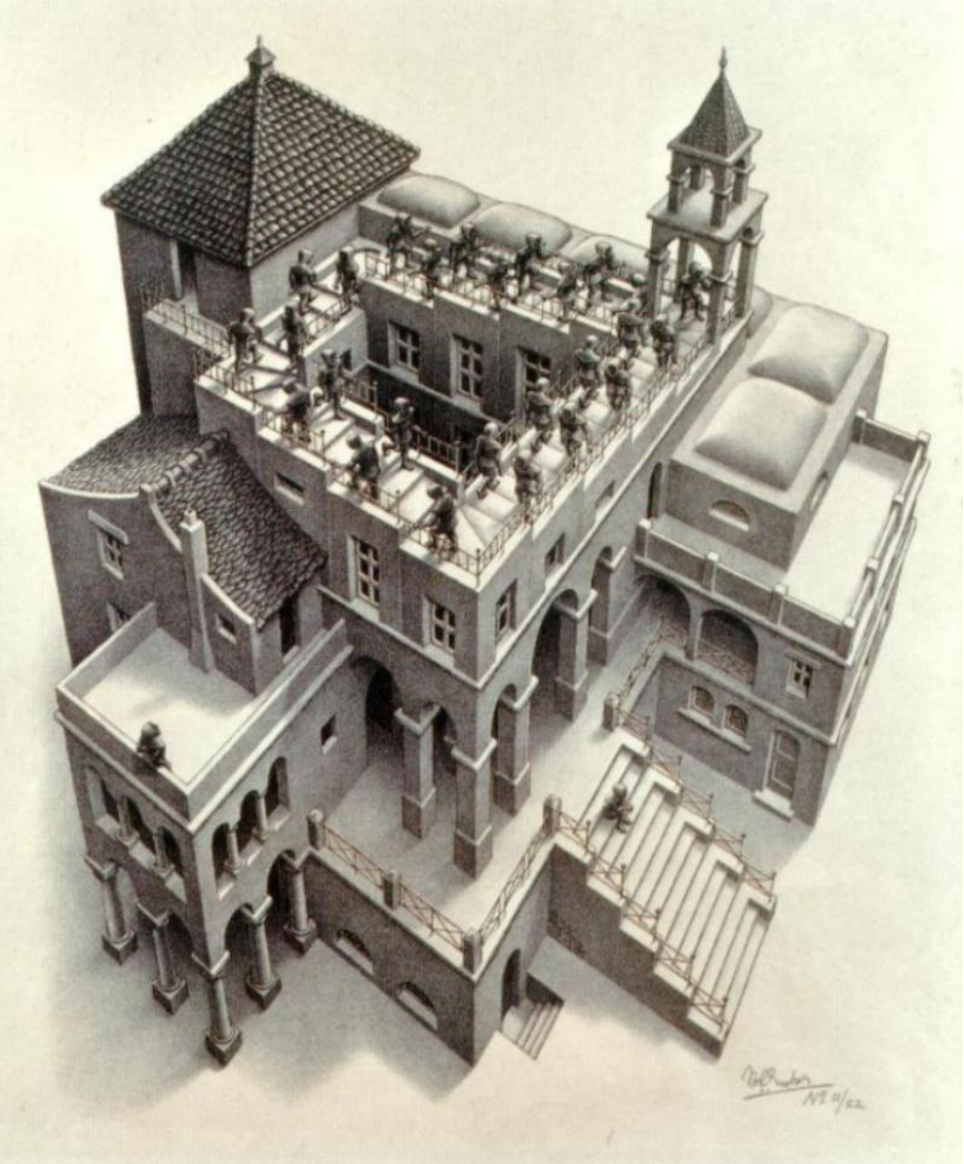
Available here: http://www.lottolab.org/Visual%20Demos/Demo%2015.html

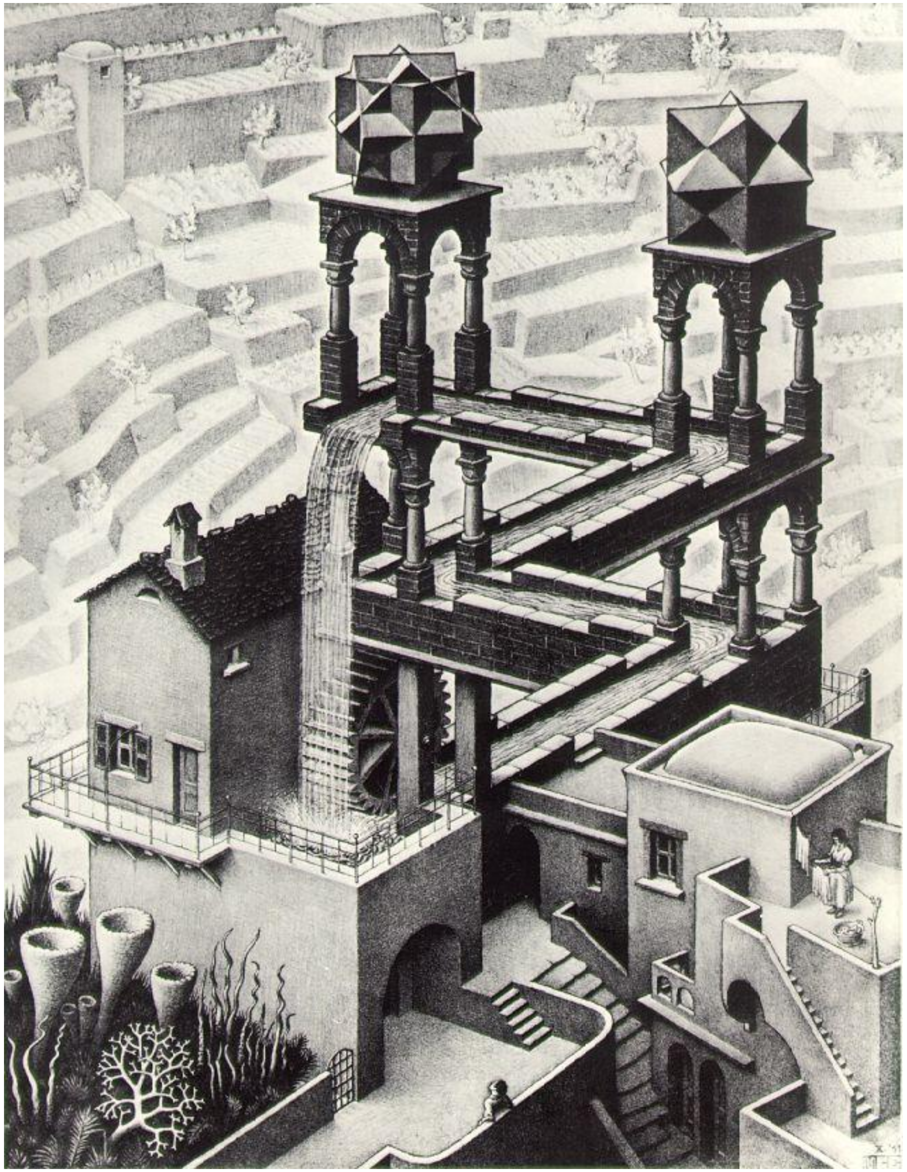
# Optical Illusions

Our visual systems play lots of interesting tricks on us

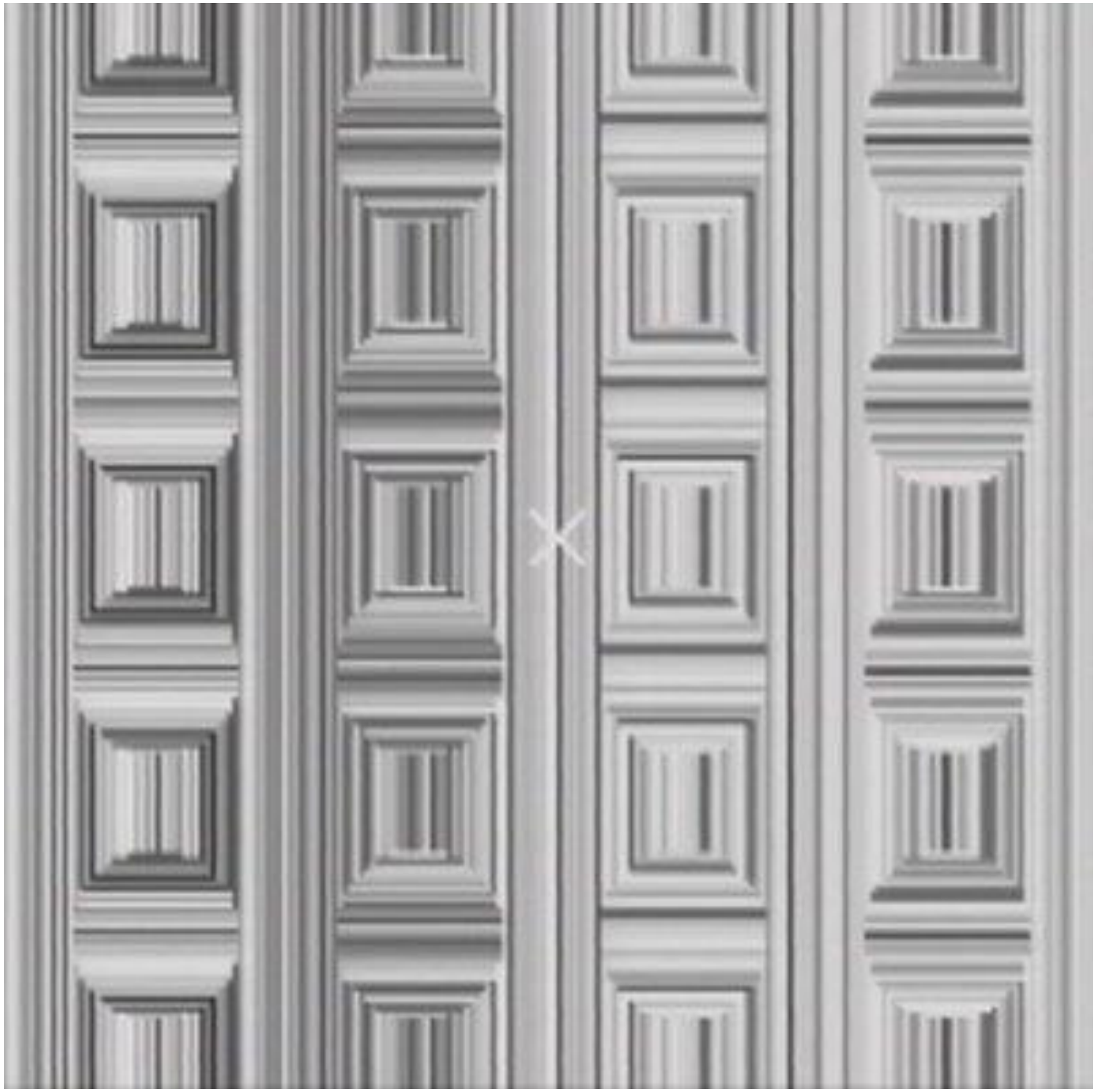


# Optical Illusions (cont…)





# Optical Illusions (cont…)



Stare at the cross in the middle of the image and think circles

# Mind Map Exercise: Mind Mapping For Note Taking


TED Ideas worth spreading

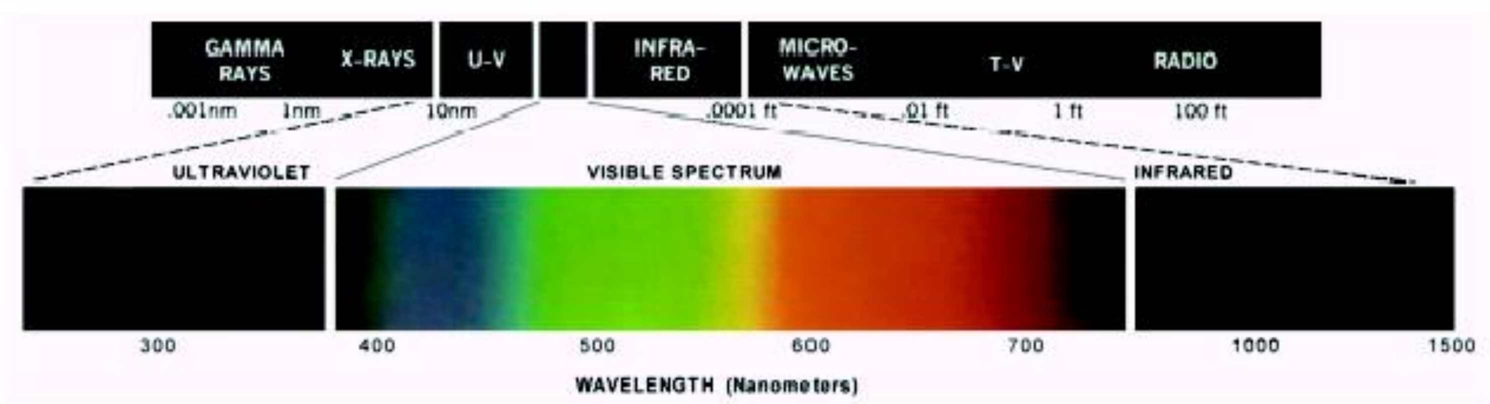
Beau Lotto: Optical Illusions Show How We See

http://www.ted.com/talks/lang/eng/beau_lotto_optical_illusions_show_how_we_see.html

# Light And The Electromagnetic Spectrum

Light is just a particular part of the electromagnetic spectrum that can be sensed by the human eye

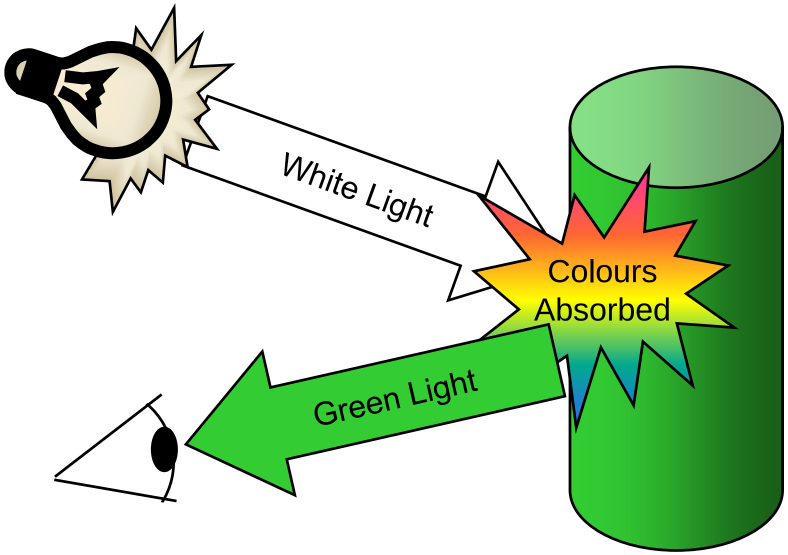
The electromagnetic spectrum is split up according to the wavelengths of different forms of energy



# Reflected Light

The colours that we perceive are determined by the nature of the light reflected from an object

For example, if white light is shone onto a green object most wavelengths are absorbed, while green light is reflected from the object



# Sampling, Quantisation And Resolution

In the following slides we will consider what is involved in capturing a digital image of a real-world scene

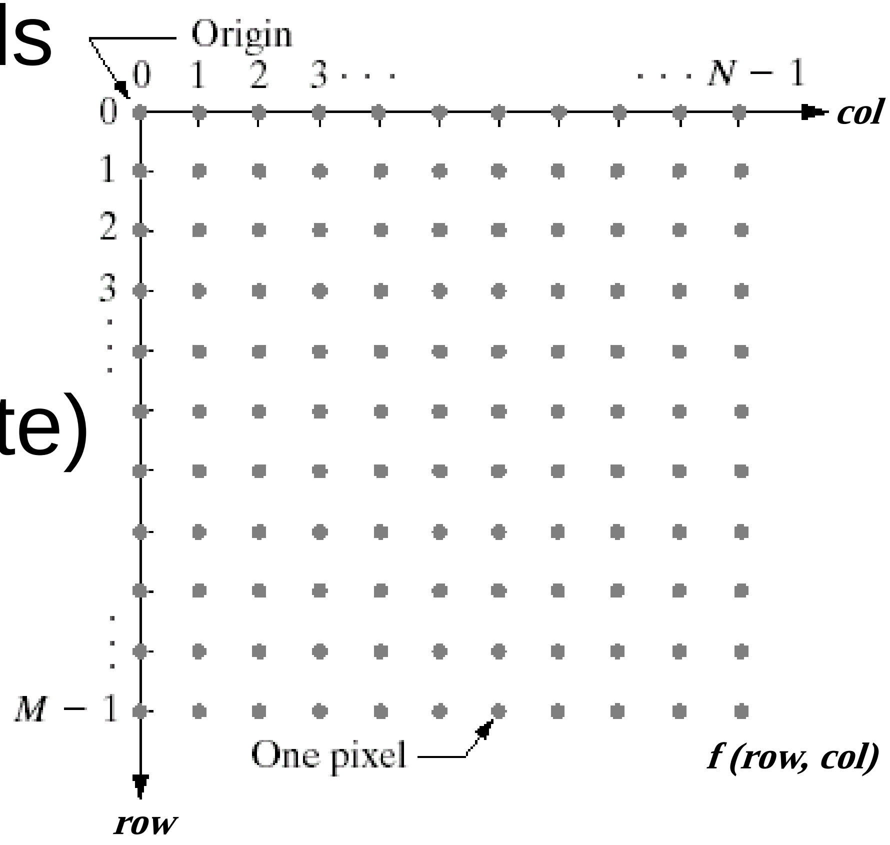
- Image sensing and representation
- Sampling and quantisation
- Resolution

# Image Representation

Before we discuss image acquisition recall that a digital image is composed of $M$ rows and $N$ columns of pixels each storing a value

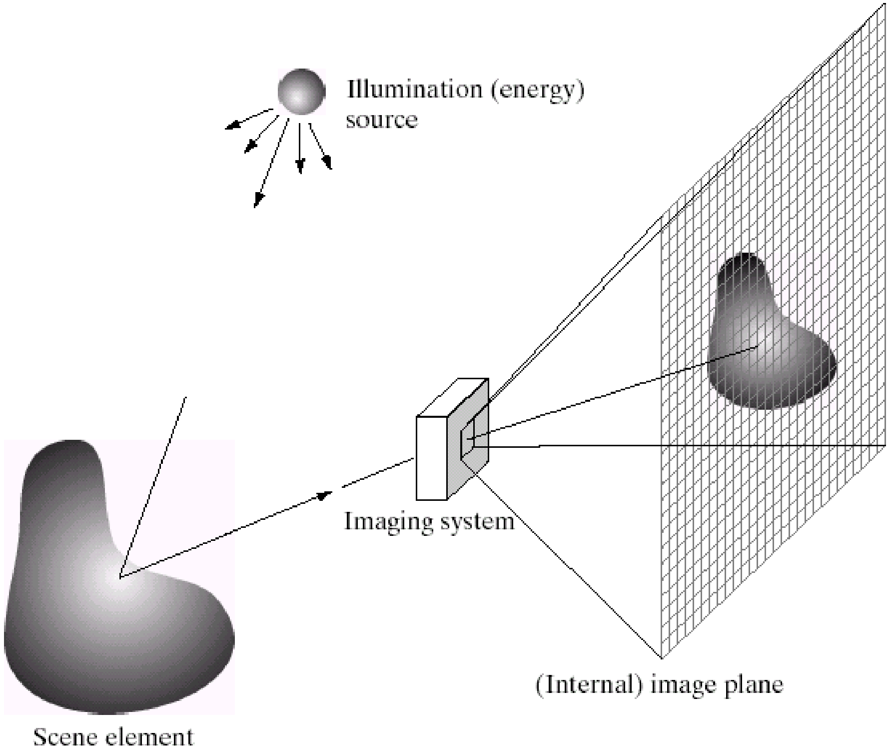
Pixel values are most often grey levels in the range 0-255(black-white)

We will see later on that images can easily be represented as matrices



# Image Acquisition

Images are typically generated by *illuminating* a *scene* and absorbing the energy reflected by the objects in that scene

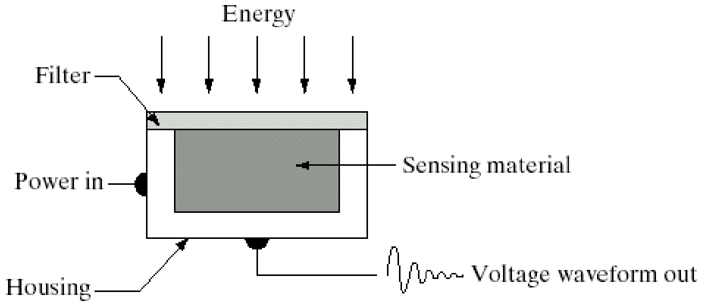



- Typical notions of illumination and scene can be way off:
  - X-rays of a skeleton
  - Ultrasound of an unborn baby
  - Electro-microscopic images of molecules

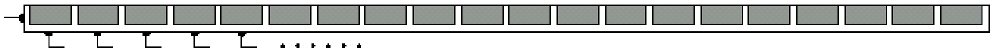
# Image Sensing

Incoming energy lands on a sensor material responsive to that type of energy and this generates a voltage

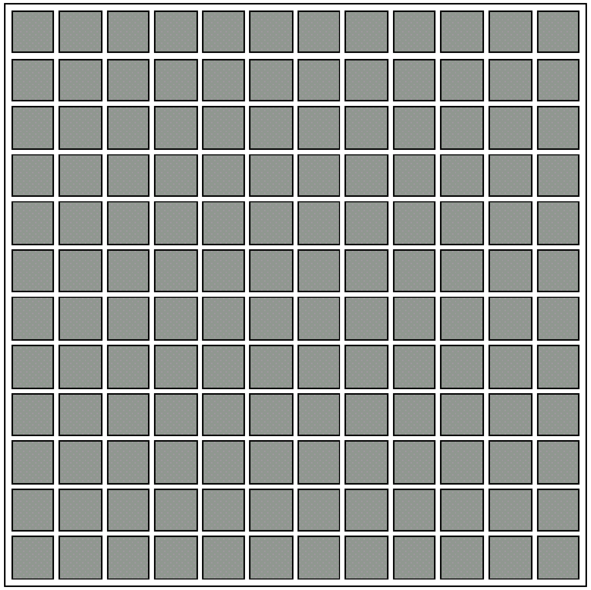
Collections of sensors are arranged to capture images




Imaging Sensor

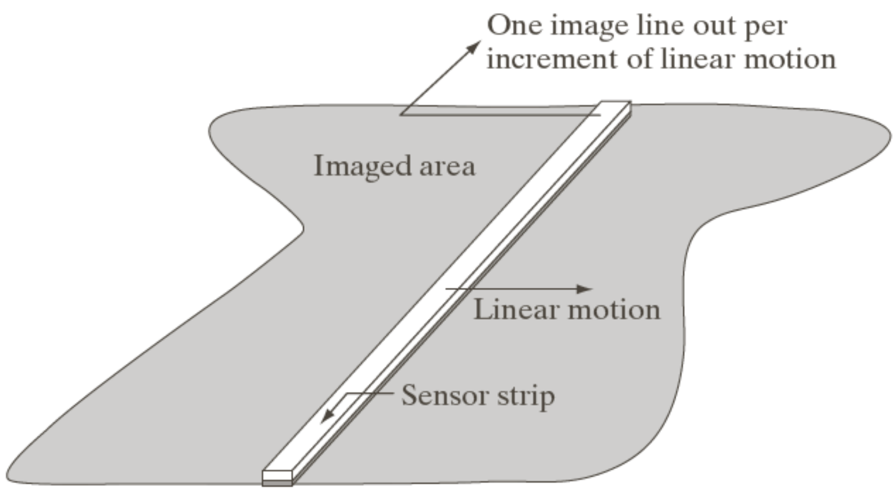


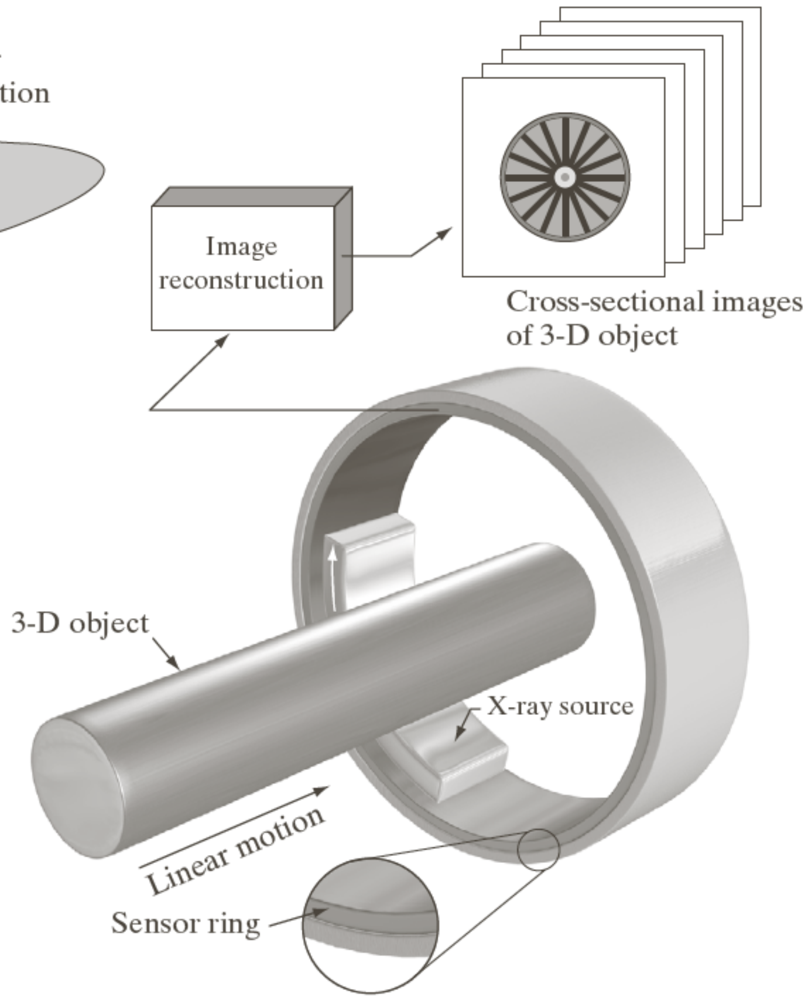
Line of Image Sensors



Array of Image Sensors

# Image Sensing



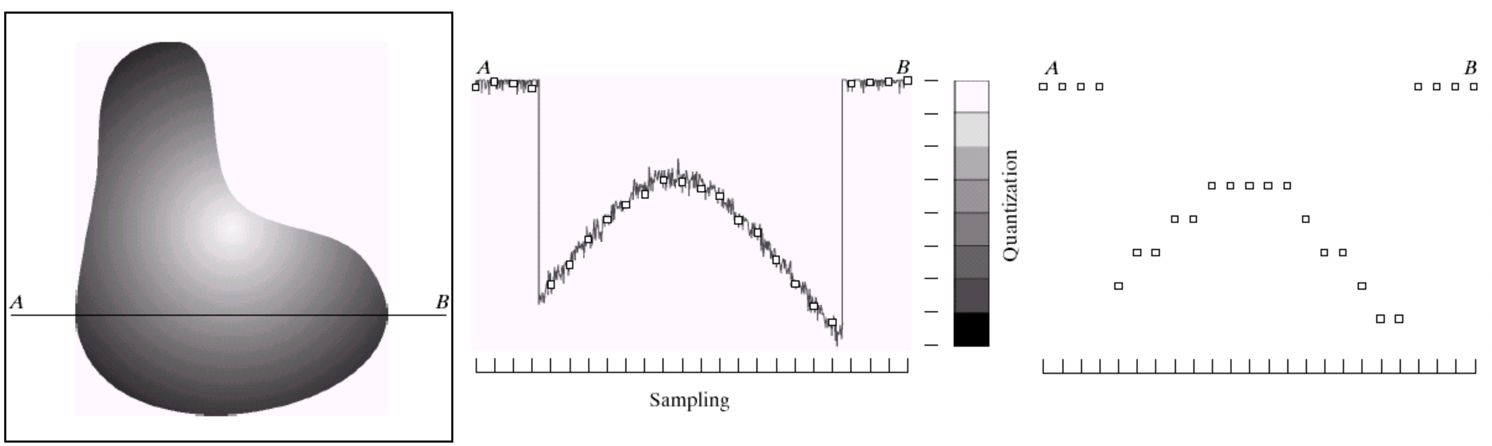




Using Sensor Strips and Rings

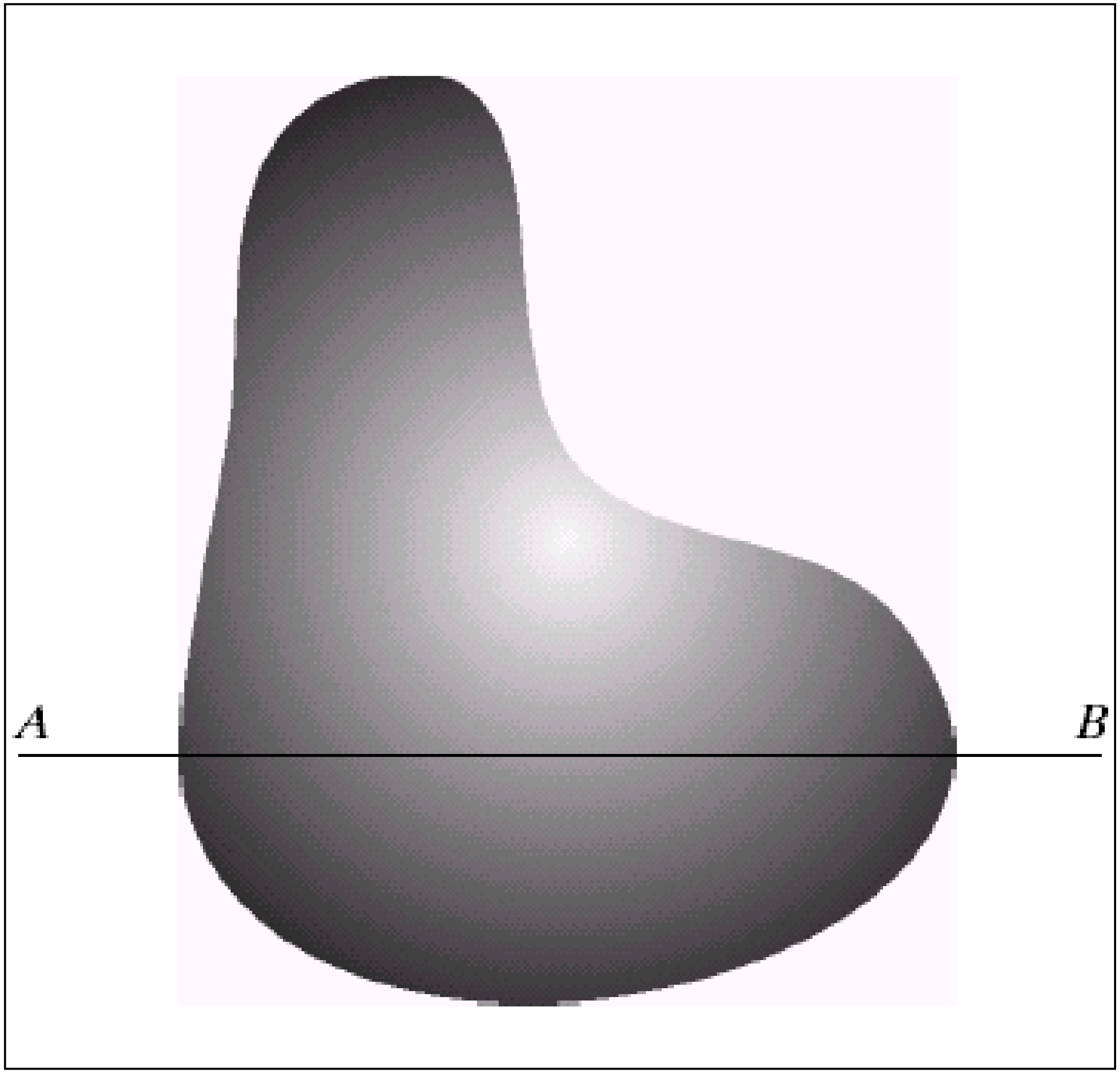
# Image Sampling And Quantisation

A digital sensor can only measure a limited number of **samples** at a **discrete** set of energy levels

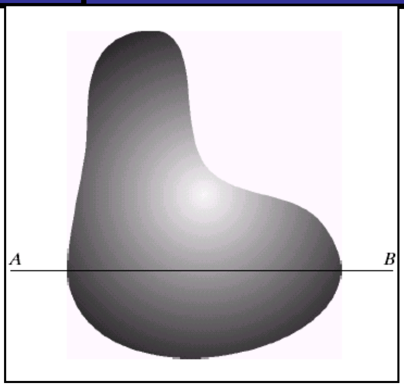
*Quantisation* is the process of converting a continuous **analogue** signal into a digital representation of this signal

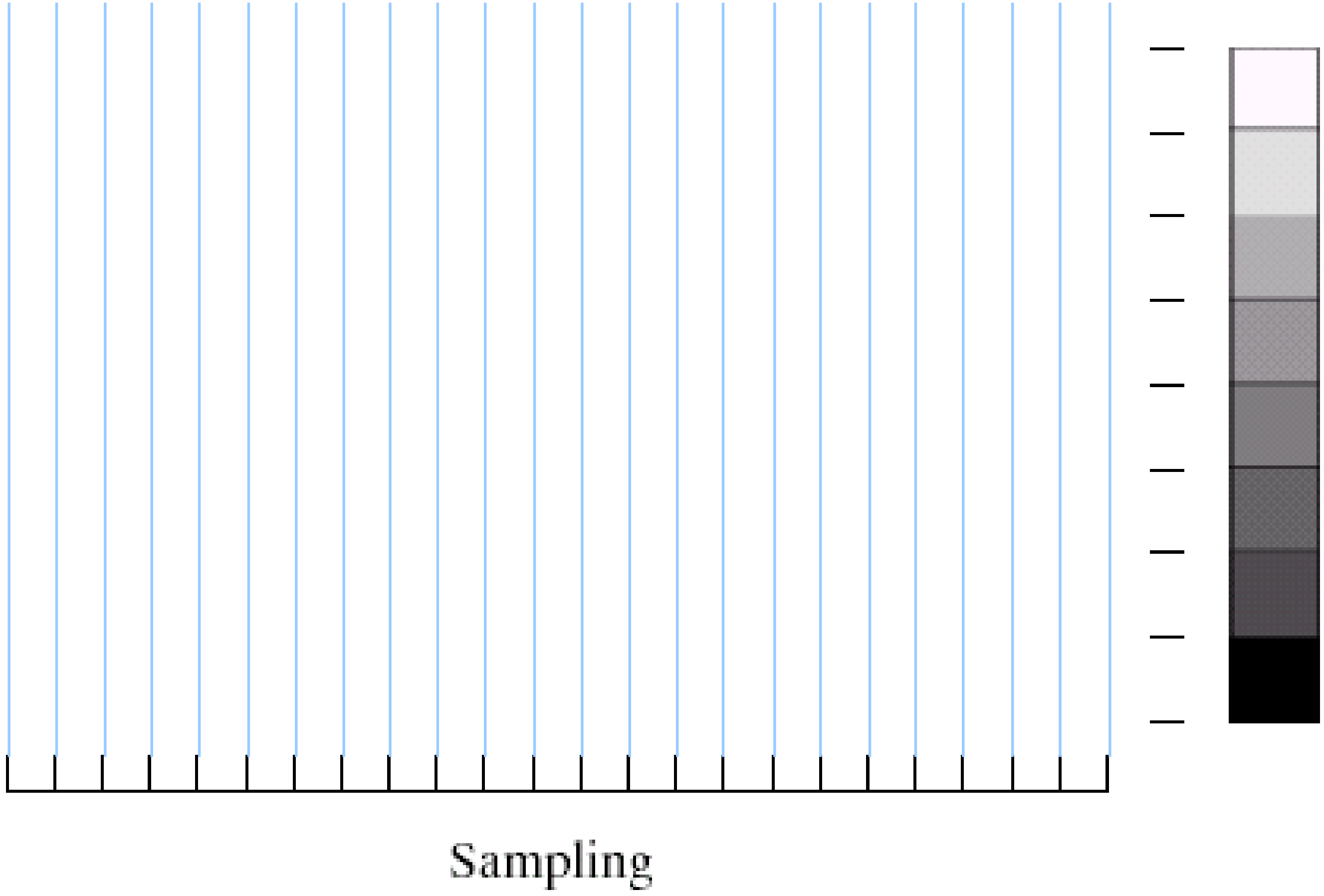


# Image Sampling And Quantisation



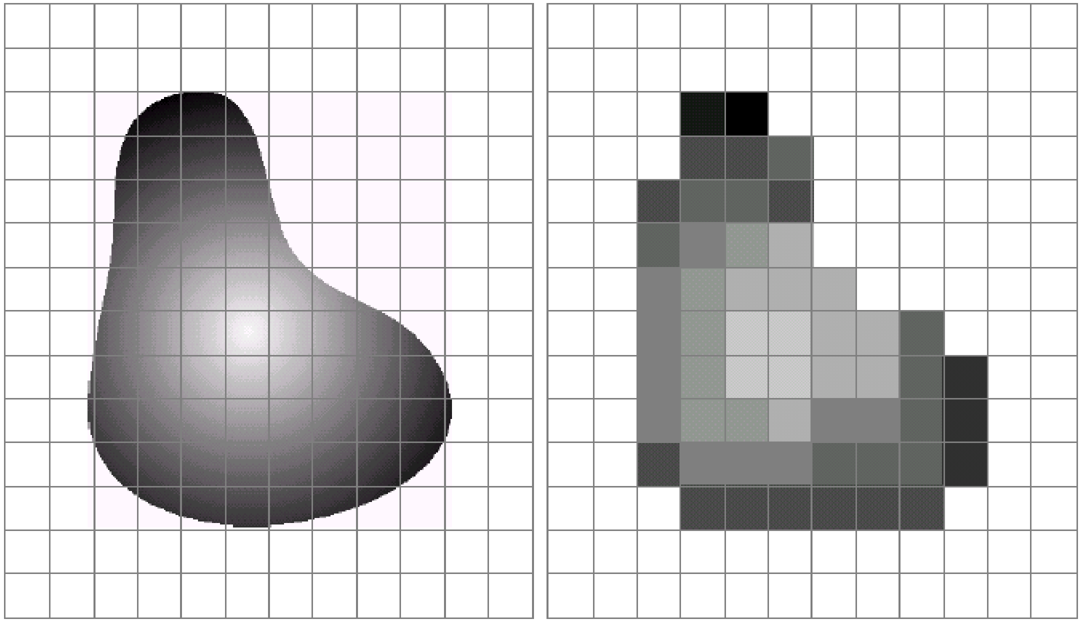
# Image Sampling And Quantisation



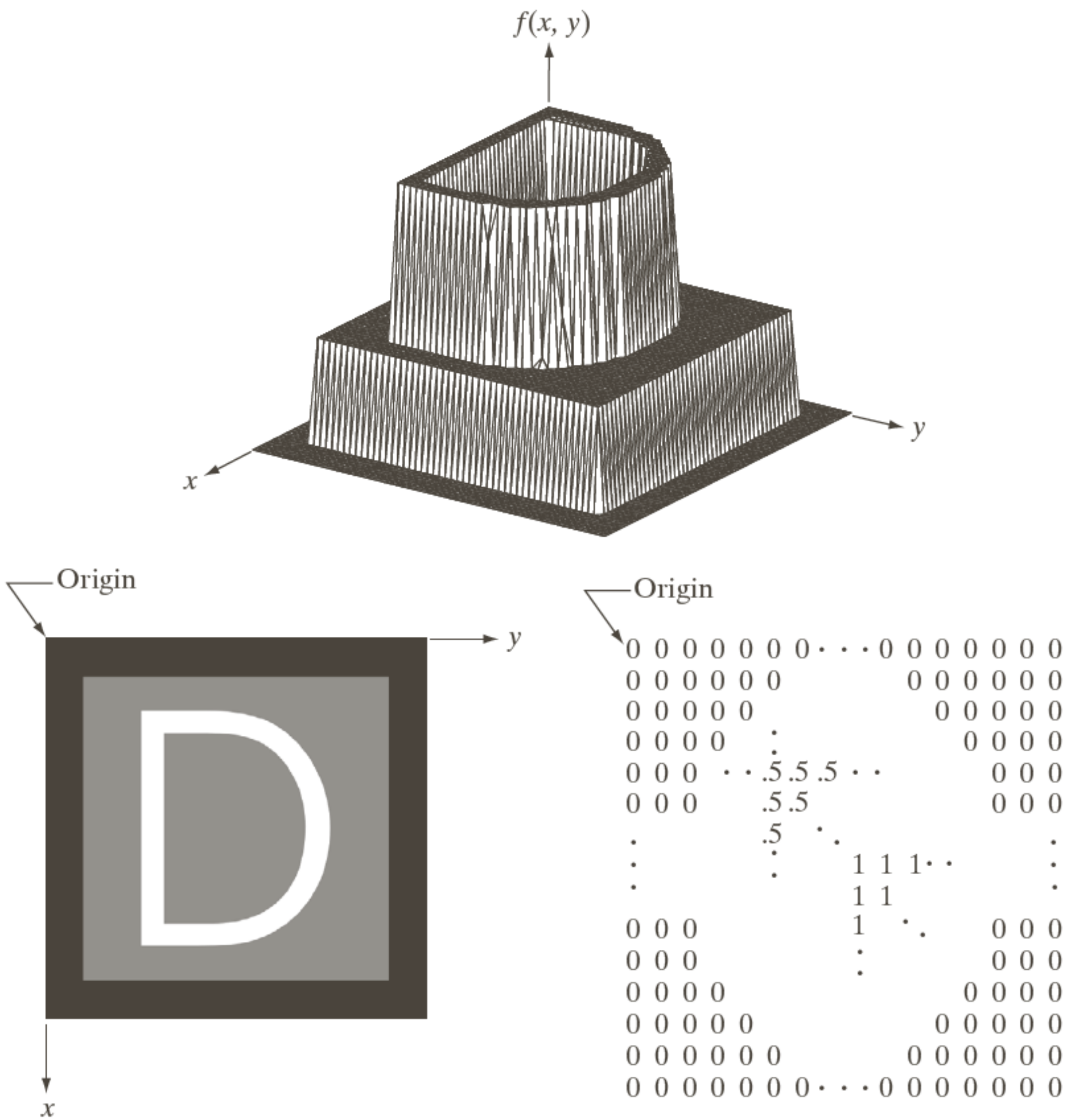



# Image Sampling And Quantisation (cont…)

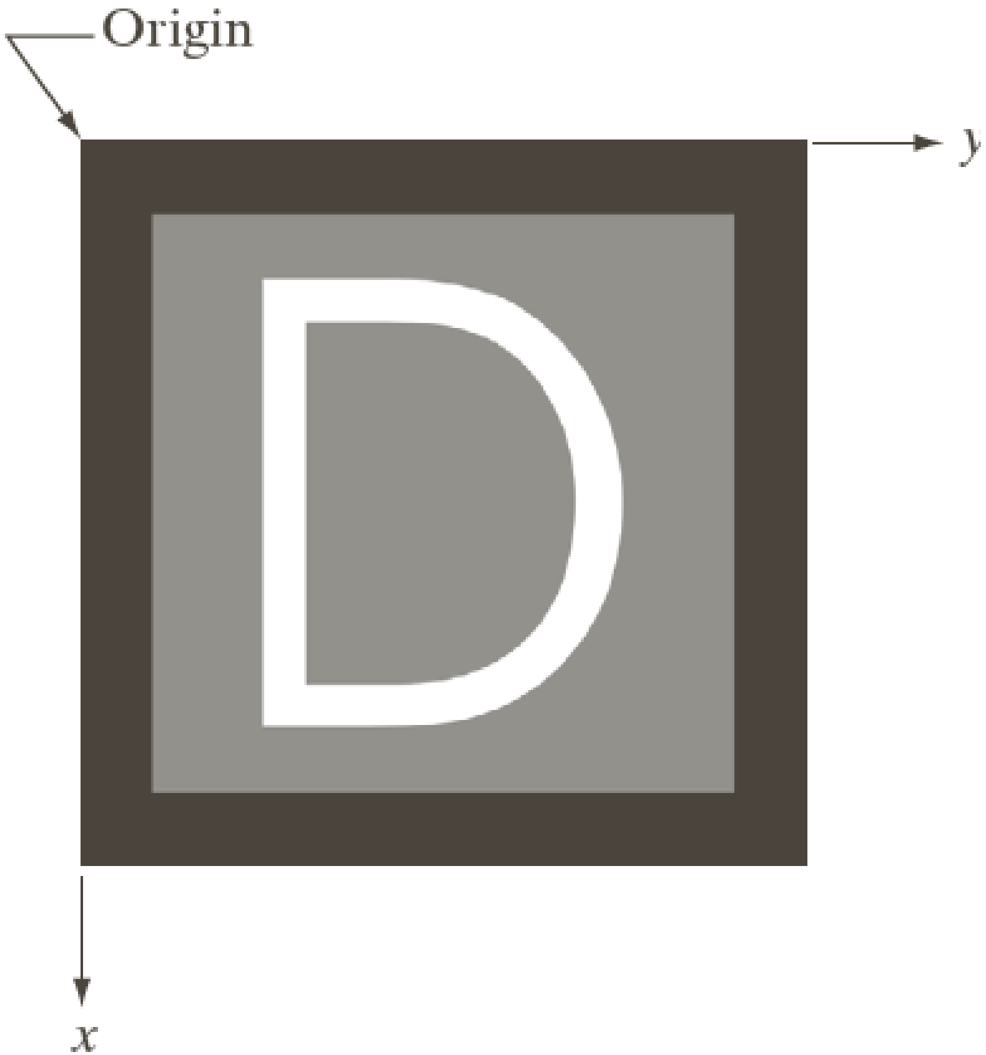
Remember that a digital image is always only an **approximation** of a real world scene



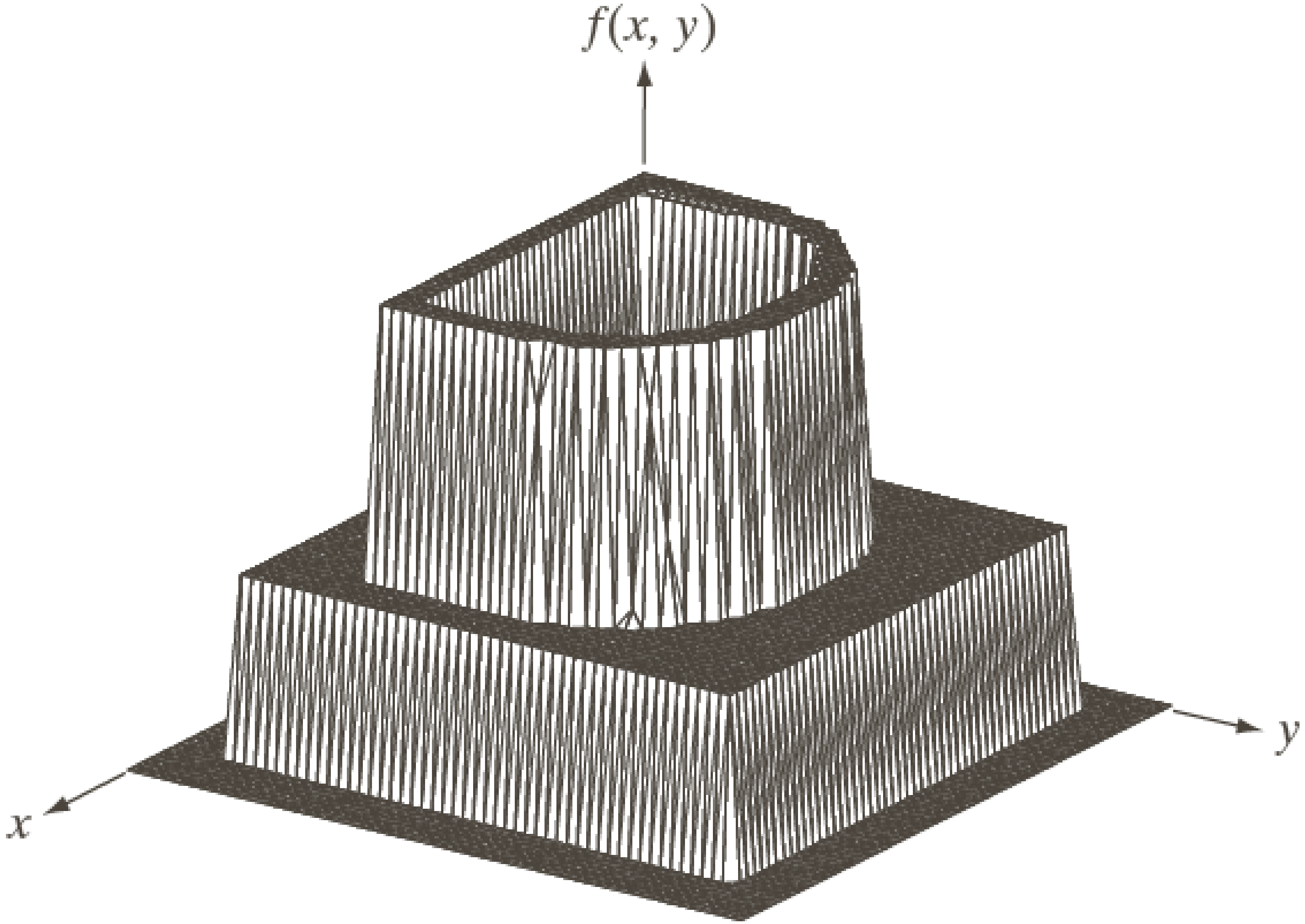
# Image Representation



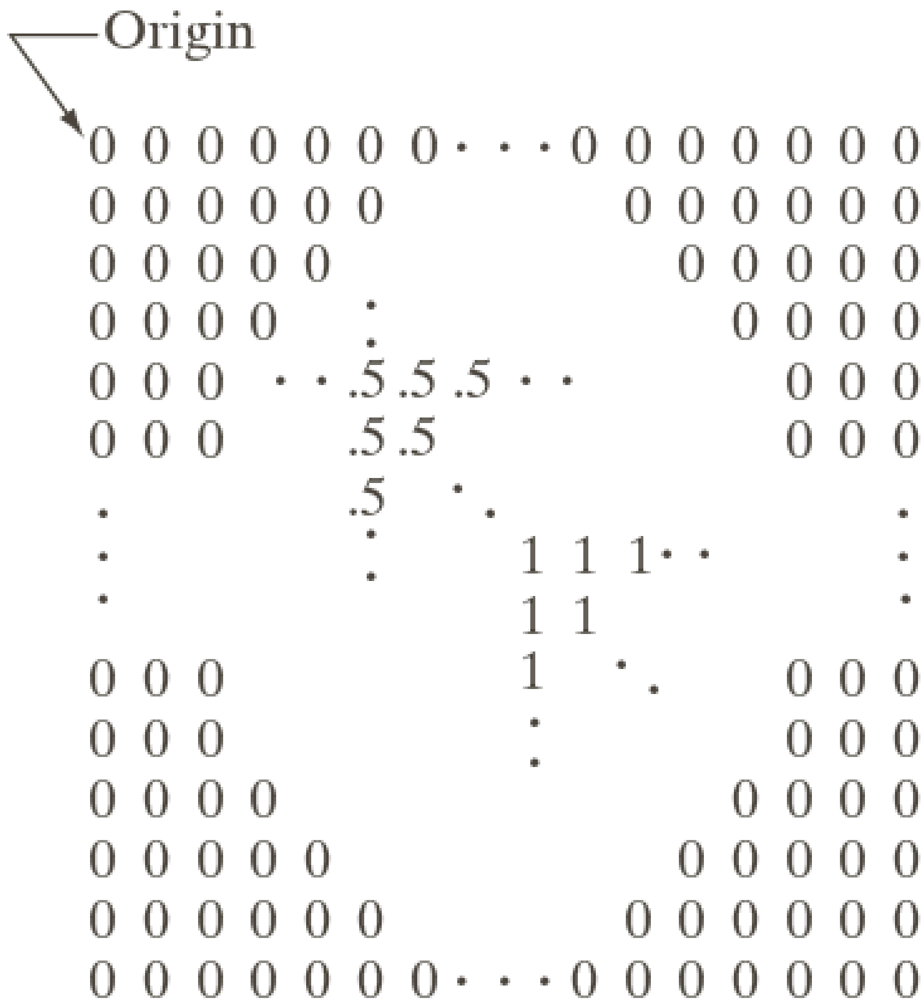
# Image Representation



# Image Representation



# Image Representation



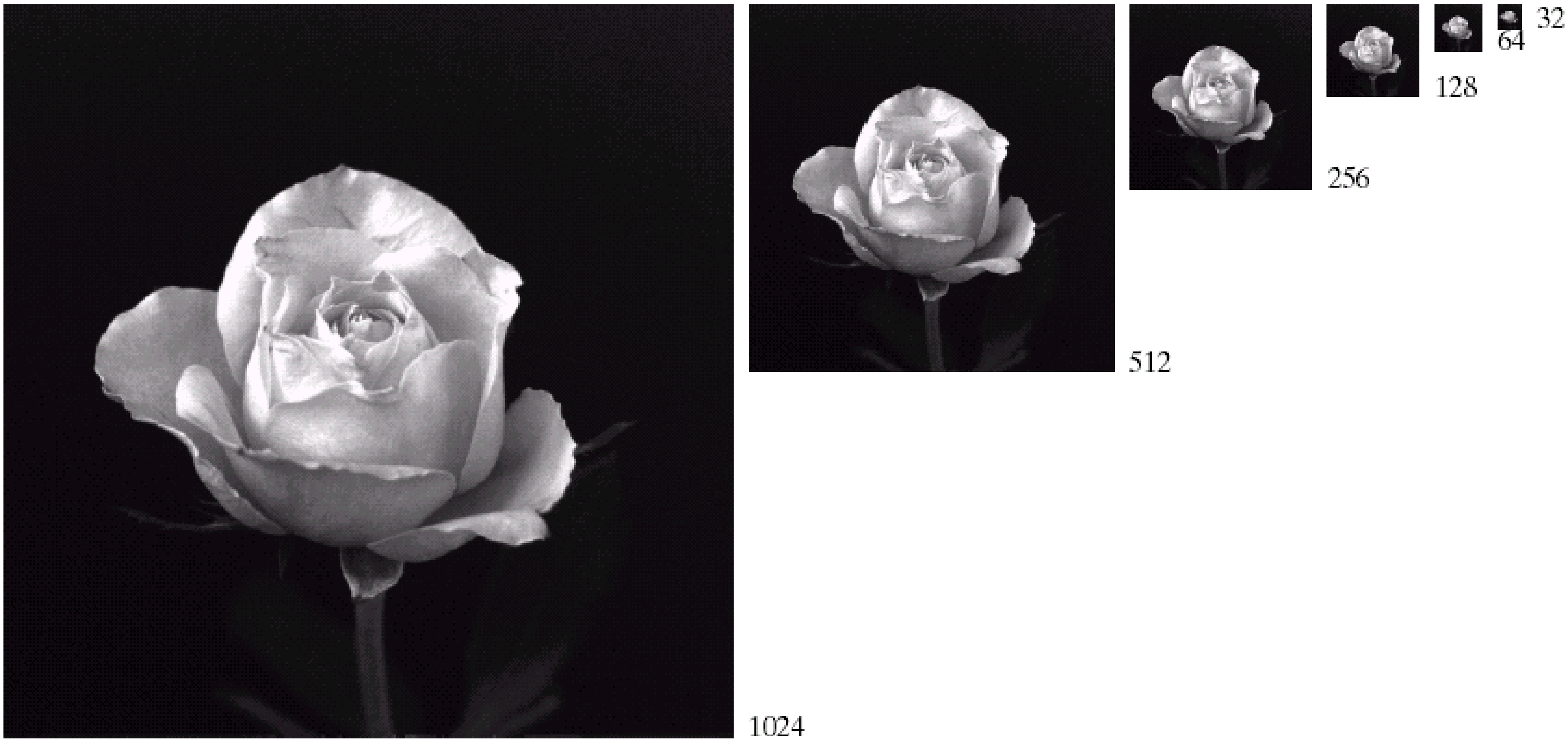
# Spatial Resolution

*The spatial resolution* of an image is determined by how sampling was carried out

Spatial resolution simply refers to the smallest discernable detail in an image

- Vision specialists will often talk about pixel size
- Graphic designers will talk about *dots per inch* (DPI)

# Spatial Resolution (cont…)



# Spatial Resolution (1024)

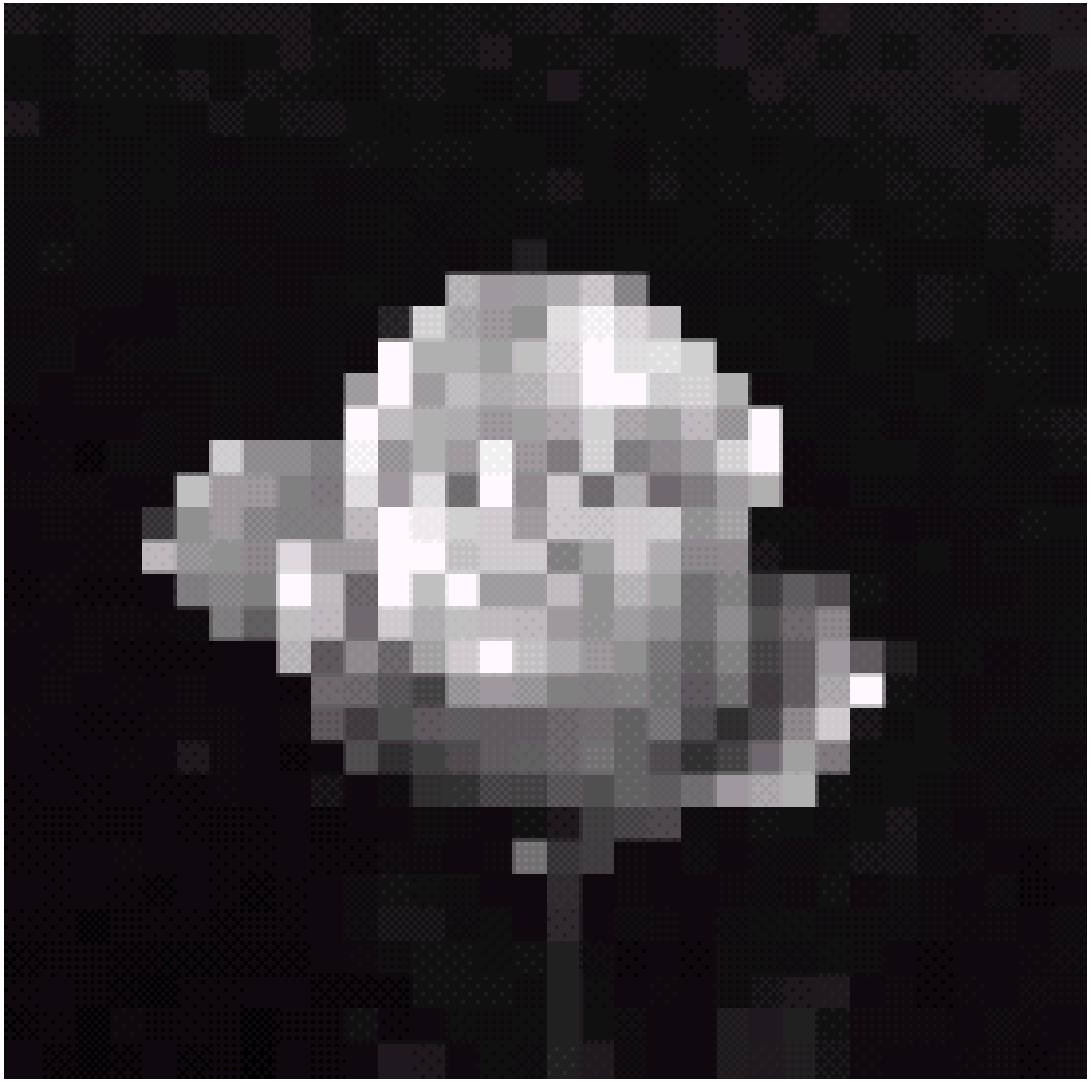
# Spatial Resolution (512)

# Spatial Resolution (256)

# Spatial Resolution (128)

# Spatial Resolution (64)

# Spatial Resolution (32)



# Intensity Level Resolution

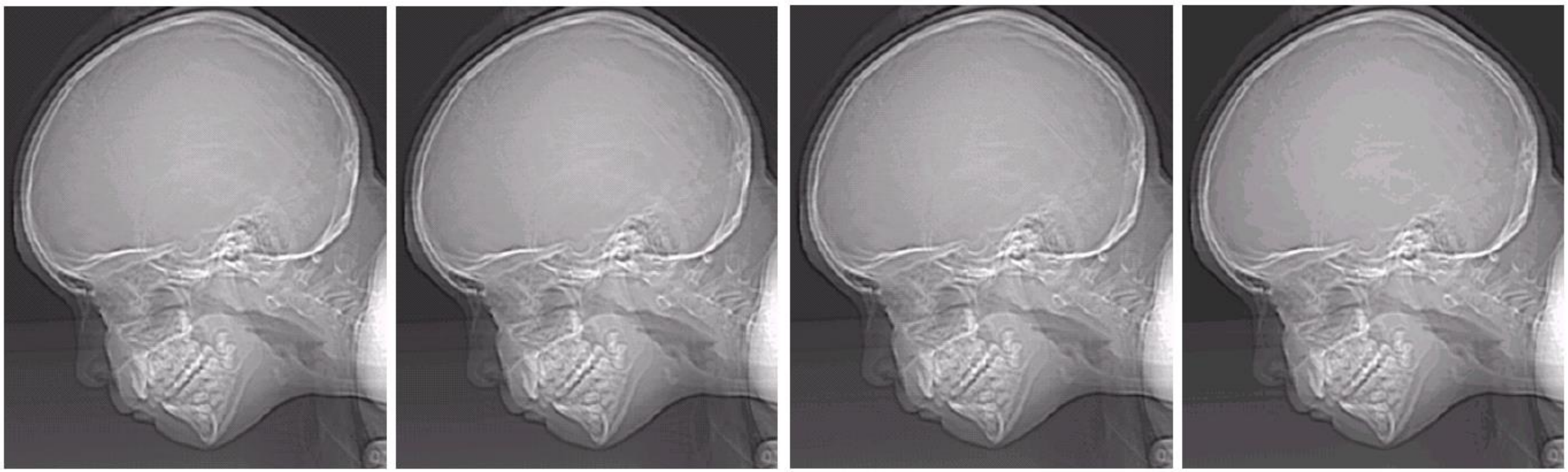
*Intensity level resolution* refers to the number of intensity levels used to represent the image

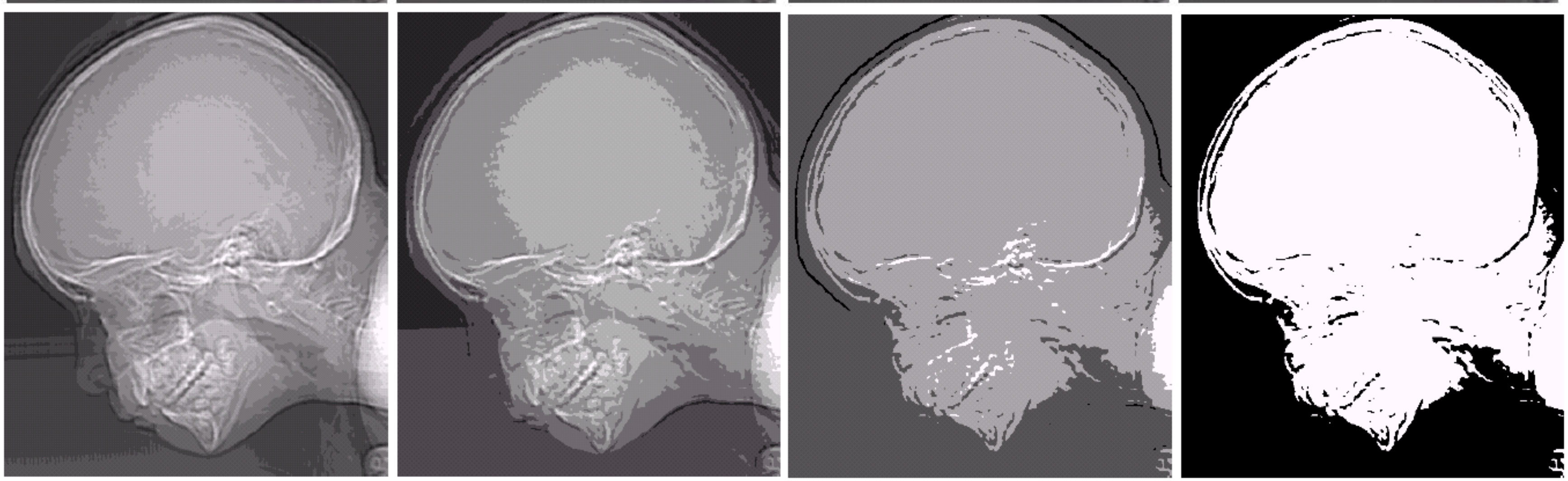
- The more intensity levels used, the finer the level of detail discernable in an image
- Intensity level resolution is usually given in terms of the number of bits used to store each intensity level

| Number of Bits | Number of Intensity Levels | Examples |
|---|---|---|
| 1 | 2 | 0, 1 |
| 2 | 4 | 00, 01, 10, 11 |
| 4 | 16 | 0000, 0101, 1111 |
| 8 | 256 | 00110011, 01010101 |
| 16 | 65,536 | 1010101010101010 |

# Intensity Level Resolution (cont…)

256 grey levels (8 bits per pixel) | 128 grey levels (7 bpp) | 64 grey levels (6 bpp) | 32 grey levels (5 bpp)





16 grey levels (4 bpp) | 8 grey levels (3 bpp) | 4 grey levels (2 bpp) | 2 grey levels (1 bpp)

# Intensity Level Resolution (8 Bit)

# Intensity Level Resolution (7 Bit)

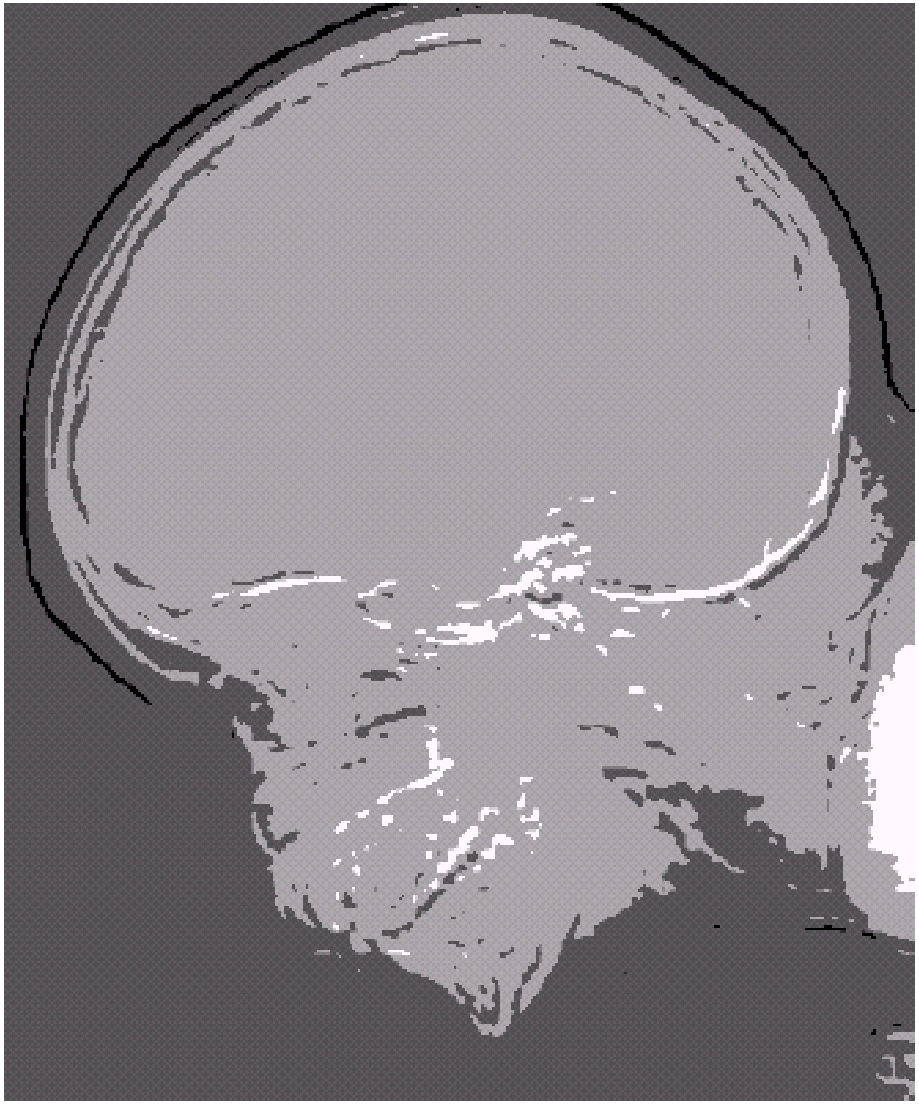
# Intensity Level Resolution (6 Bit)

# Intensity Level Resolution (5 Bit)

# Intensity Level Resolution (4 Bit)

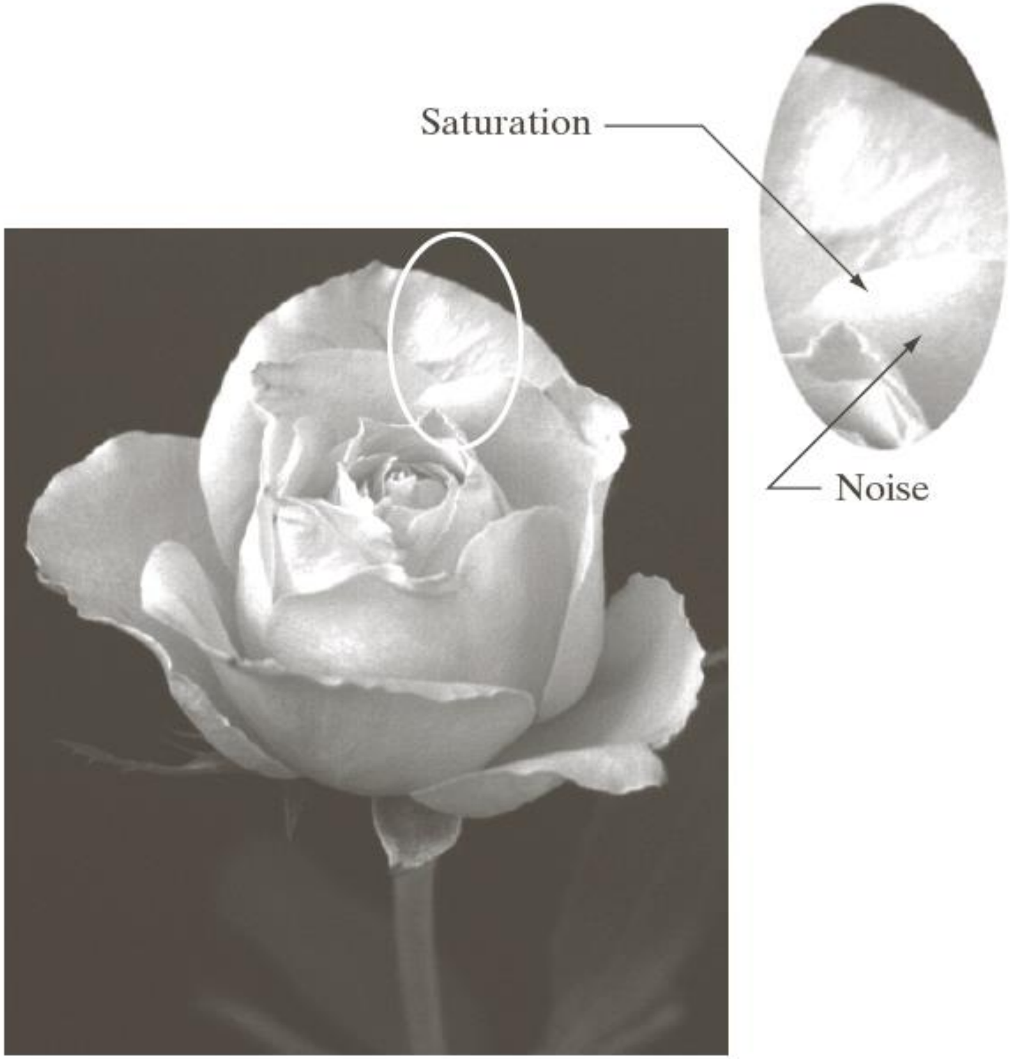
# Intensity Level Resolution (3 Bit)

# Intensity Level Resolution (2 Bit)



# Intensity Level Resolution (1 Bit)

# Saturation & Noise



# Resolution: How Much Is Enough?

The big question with resolution is always *how much is enough*?

- This all depends on what is in the image and what you would like to do with it
- Key questions include
  - Does the image look aesthetically pleasing?
  - Can you see what you need to see within the image?

# Resolution: How Much Is Enough? (cont…)


The picture on the right is fine for counting the number of cars, but not for reading the number plate

# Intensity Level Resolution (cont…)

Low Detail

Medium Detail

High Detail

# Intensity Level Resolution (cont…)

# Intensity Level Resolution (cont…)

# Intensity Level Resolution (cont…)

## *Neighbor of a pixel*

1. A pixel p at coordinates (x,y) has 4 neighbors $N_4(p)$

2. Four diagonal neighbors $N_D(p)$

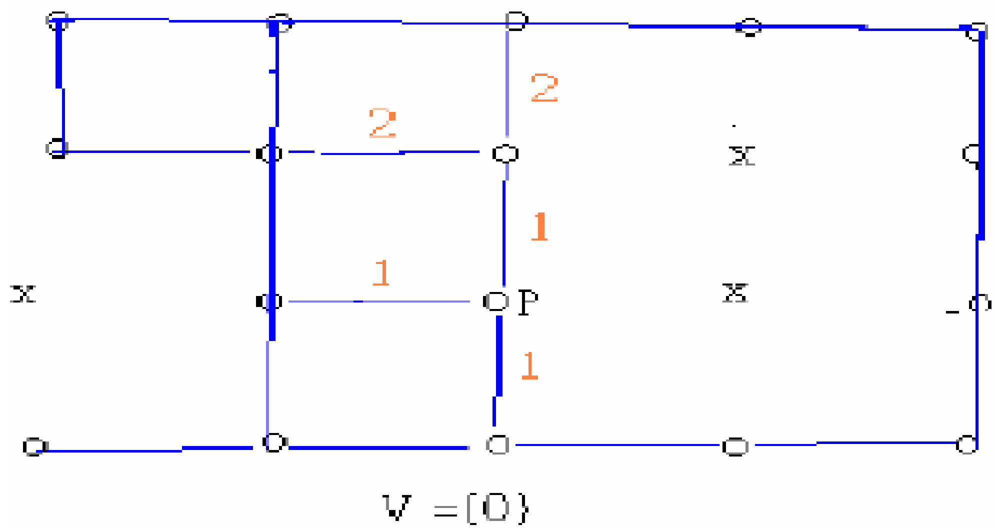
3. 8 neighbors $N_8(p)$

## *$C_4$ Connectivity*

Any (p and q ) pixels ∈ V , and if q ∈ $N_4$(p)

Then q is $C_4$ connected with p.

V is set of points to be connected.

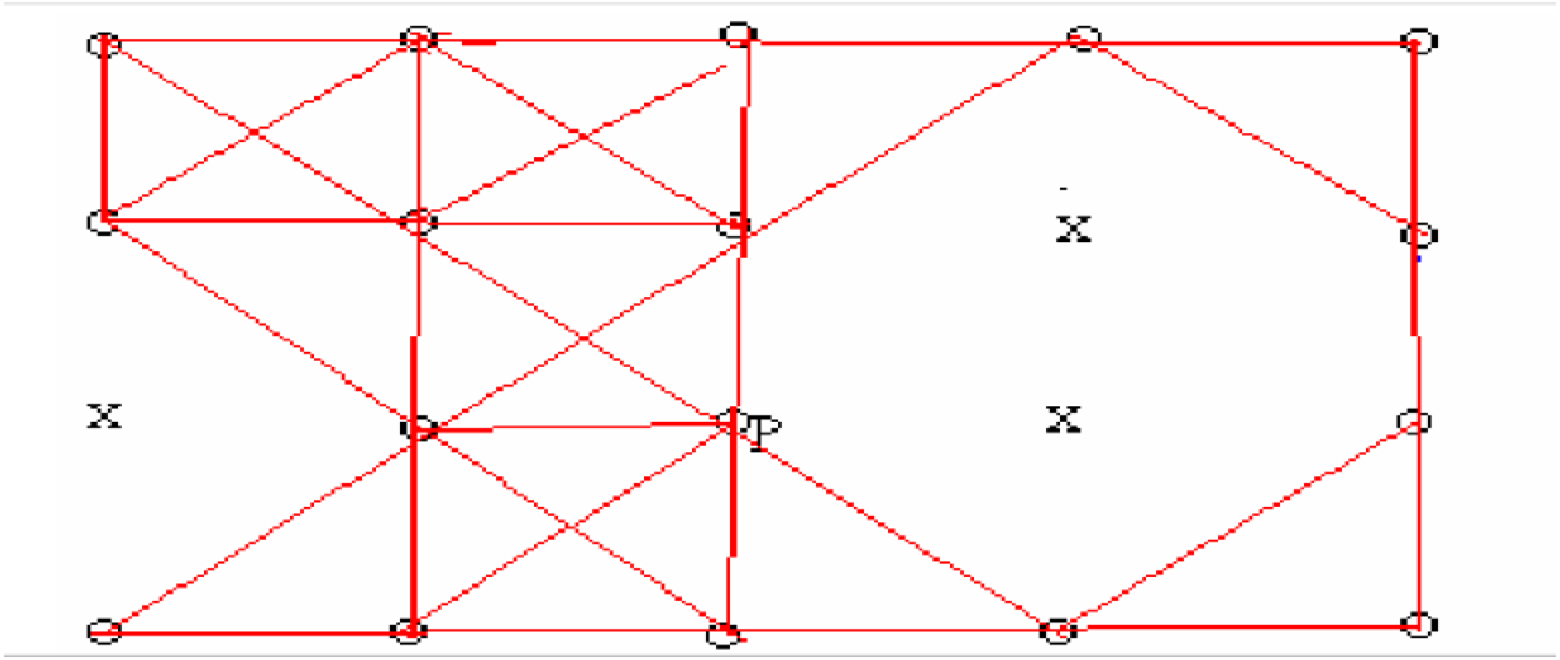


# Basic Relation
# 2. Connectivity

## *$C_8$ Connectivity*

Any (p and q ) pixels ∈ V , and if q ∈ $N_8$(p)

Then q is $C_8$ connected with p.



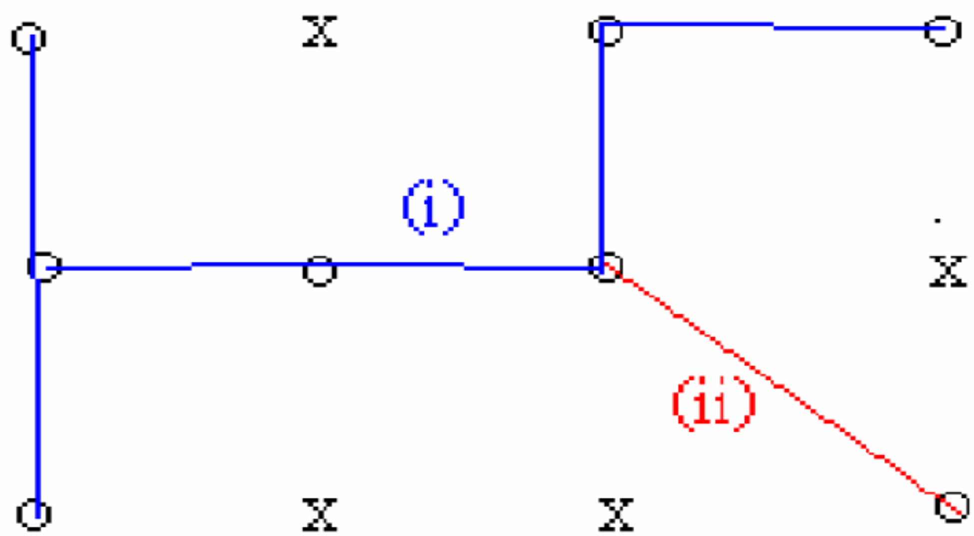
# Basic Relation
# 2. Connectivity

## *$C_m$ Connectivity*

Any (p and q ) pixels ∈ V

if q is $N_4(p)$ then q is Cm connected with p.

else if q is $N_D(p)$ and $(N_4(p) \cap N_4(q))=\emptyset$ Then q is $C_m$ connected with p.



# Basic Relation
## 3. Adjacent and Path

- *Adjacent :*

  p is adjacent to q if they are connected.

- *Path*:

  A path from $(X_0,Y_0)$ to $(X_n,Y_n)$ where $(X_0,Y_0)$ is adjacent to $(X_1,Y_1)$ and $(X_2,Y_2)$ is adjacent to $(X_2,Y_2)$ ….. $(X_n,Y_n)$.

# Basic Relation
# 4. Labeling

- ***Labeling of connected component***

**- scanning sequence**

"from left to right , from top to button "

**- procedure (eg. For 4-connected component)**

1.if $p,q \notin s$ , move on

2.if $p \in v$ ,examine r and t

(1) if both $\notin v$ , assign a new label to p

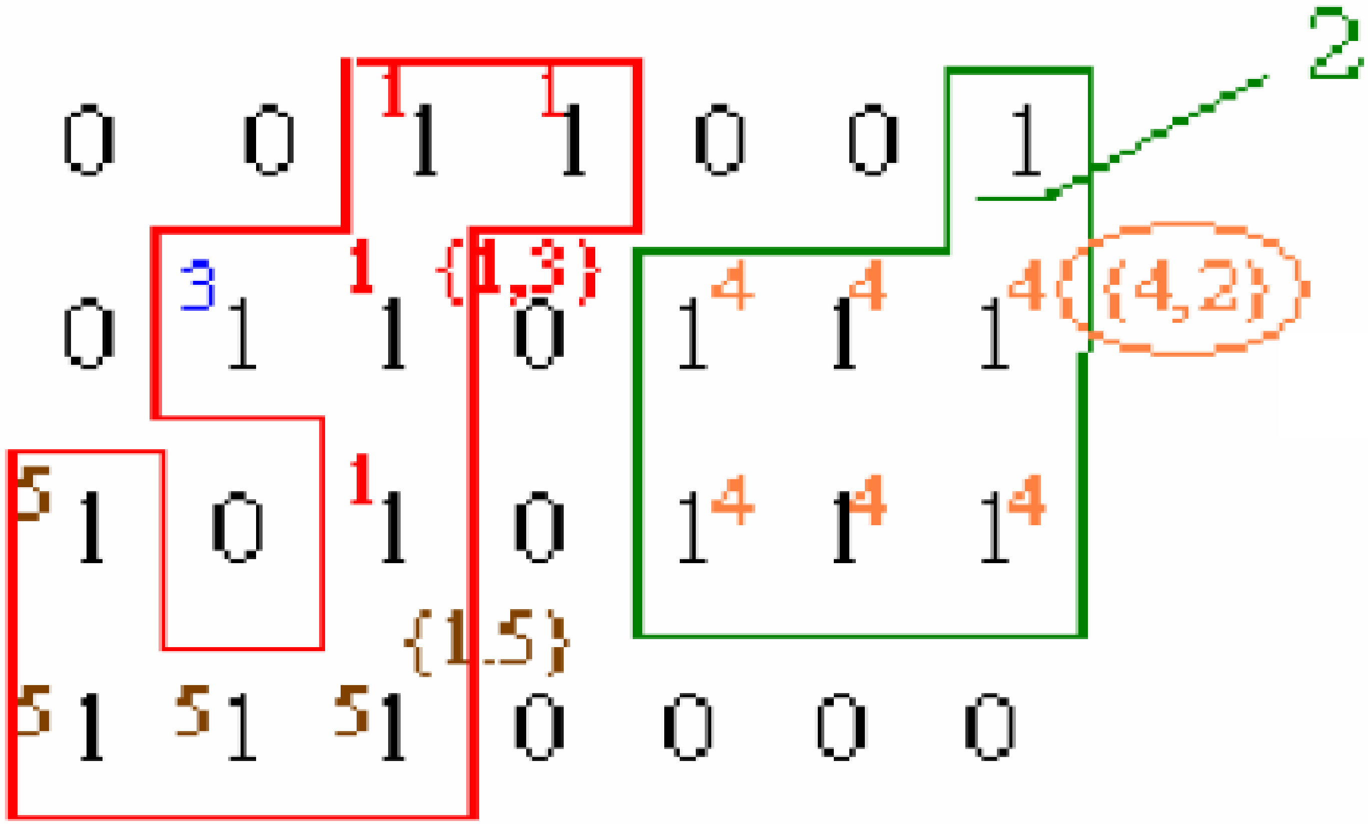
# Basic Relation
# 4. Labeling

(2) if only one of the two neighbors $\in v$ , assign
its label to p

(3) if both $\in v$ and have same label , assign
that to p

(4) if both $\in v$ , but have different lable ,
assign either one to p and make note that
these two labels one equivelent

# Example : V={1}



# Basic Relation
# 5. Distance Measue

- ***Distance Measure*** ( p(x,y) , q(s,t) )

  - **Euclidean distance**

$$De(p,q) = [(x-s)^2 + (y-t)^2]^{\frac{1}{2}}$$

  - **City-block distance**

$$D4(p,q) = |x-s| + |y-t|$$

  - **Chessboard distance**

$$D8(p,q) = \max(|x-s|, |y-t|)$$

# Summary

We have looked at:

- Human visual system
- Light and the electromagnetic spectrum
- Image representation
- Image sensing and acquisition
- Sampling, quantisation and resolution
- Basic Relation

Next time we start to look at techniques for image enhancement

LECTURE No. ( 3 )

# ***Image Enhancement***

# ***(Point Processing)***

10/14/2019

Dr. Majid D. Y.

dr.majid@uomosul.edu.iq

# Contents

Over the next few lectures we will look at image enhancement techniques working in the spatial domain:

– What is image enhancement?
– Different kinds of image enhancement
– Point processing
– Histogram processing
– Neighbourhood operations

# 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 in this lecture grey levels are assumed to be given in the range [0.0, 1.0]

# What Is Image Enhancement?

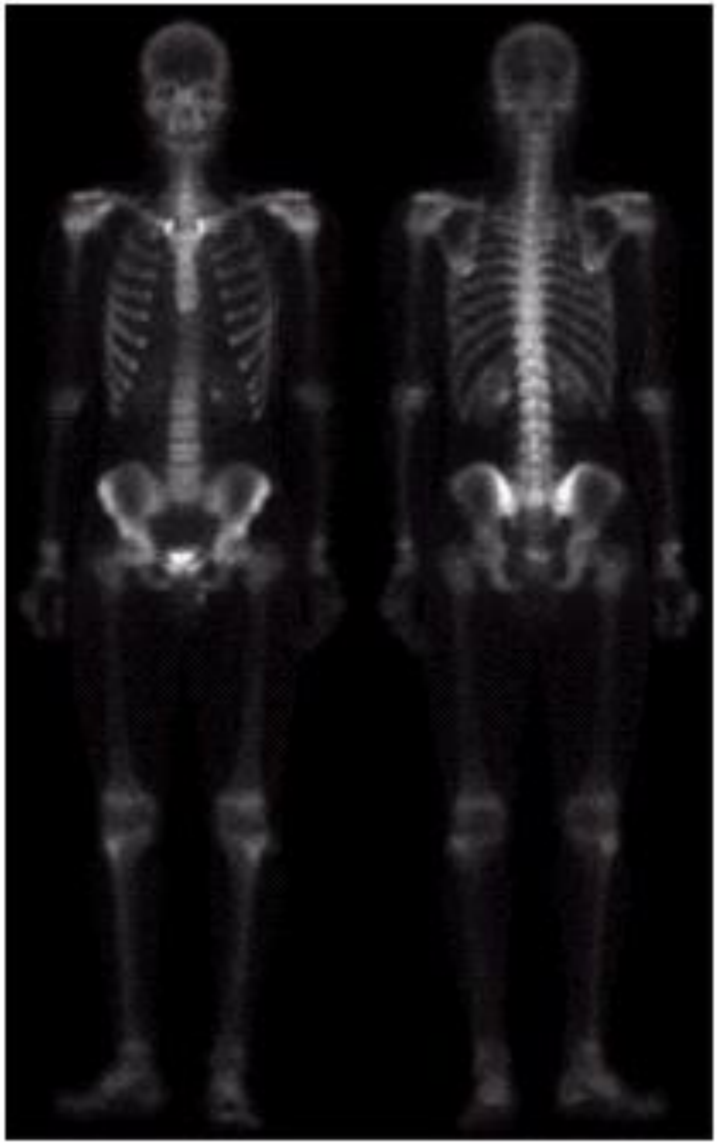
Image enhancement is the process of making images more useful

The reasons for doing this include:

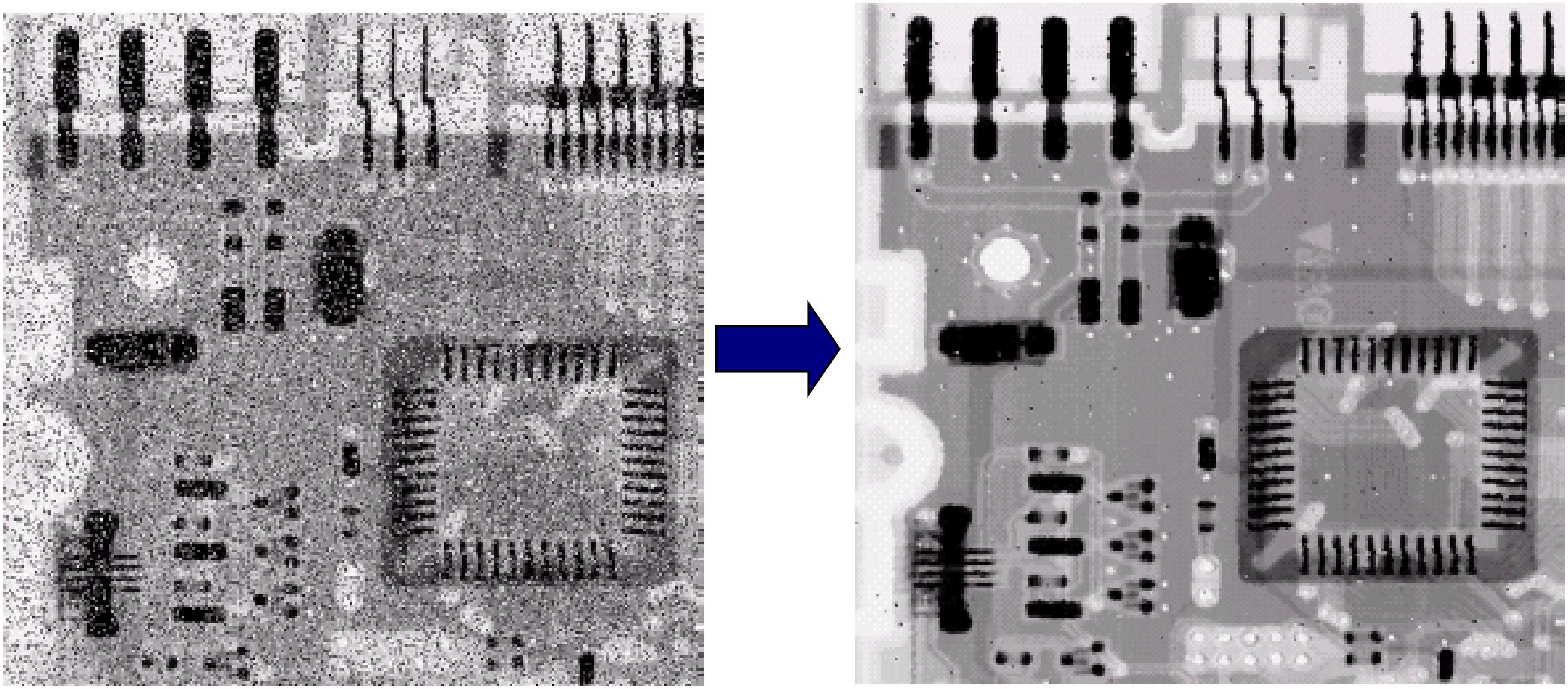
- Highlighting interesting detail in images
- Removing noise from images
- Making images more visually appealing

# Image Enhancement Examples

# Image Enhancement Examples (cont…)



# Image Enhancement Examples (cont…)



# 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

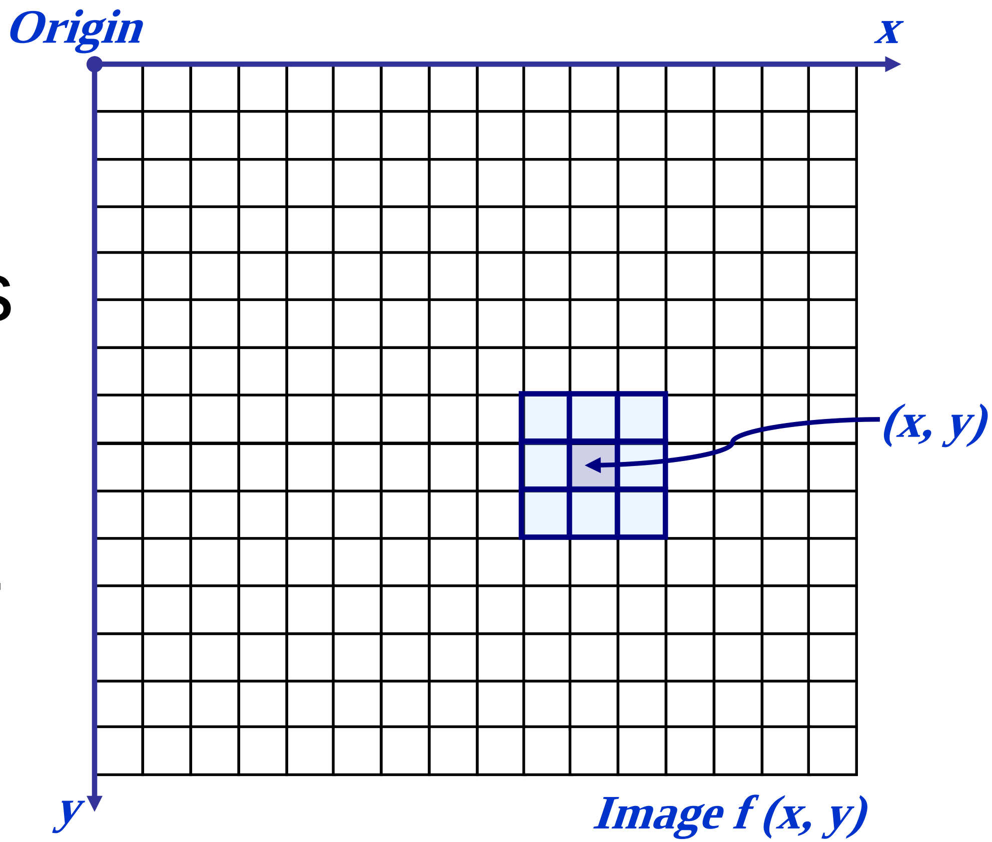
For the moment we will concentrate on techniques that operate in the spatial domain

# 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

– What is point processing?
– Negative images
– Thresholding
– Logarithmic transformation
– Power law transforms
– Piecewise Linear Transformation Functions
– Grey level slicing
– Bit plane slicing

# 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

$$s = T(r)$$

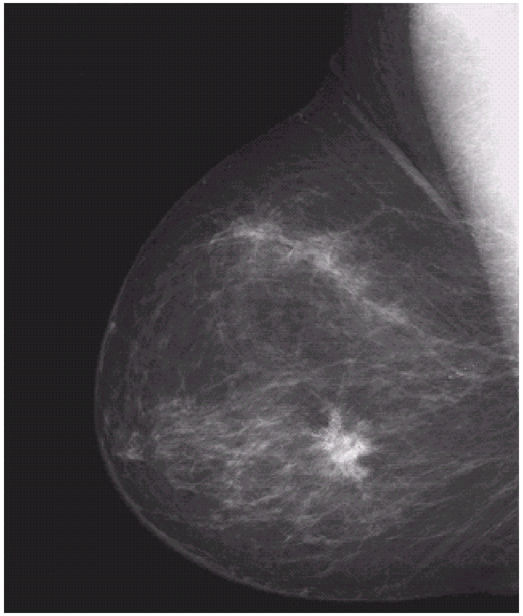
where $s$ refers to the processed image pixel value and $r$ refers to the original image pixel value

# 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

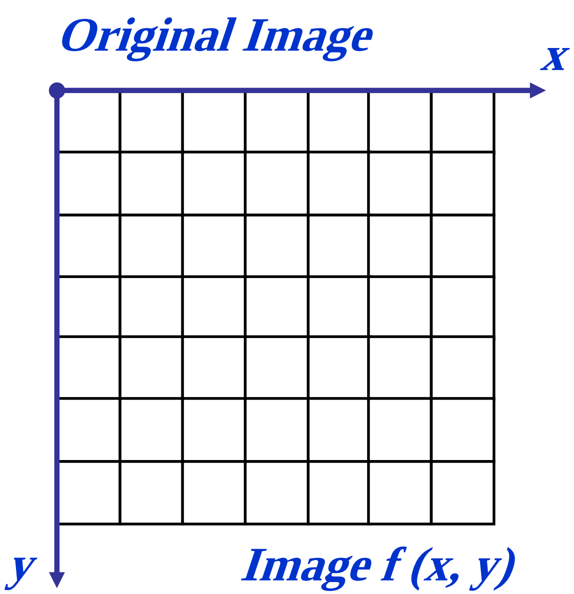
**Original Image**

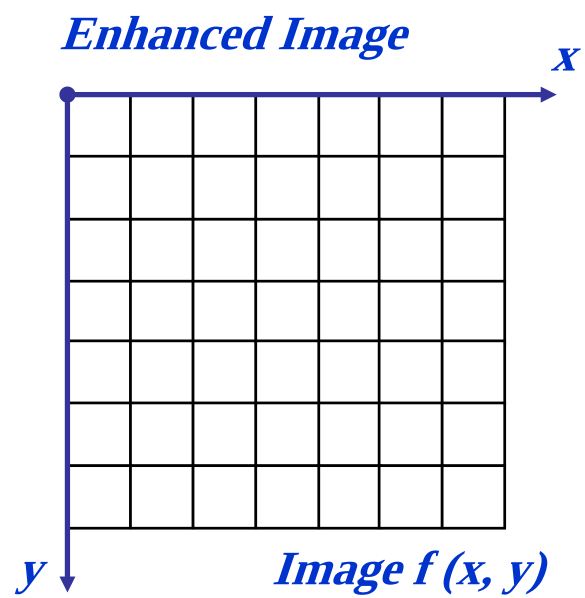


$s = 1.0 - r$

**Negative Image**

# Point Processing Example: Negative Images (cont…)



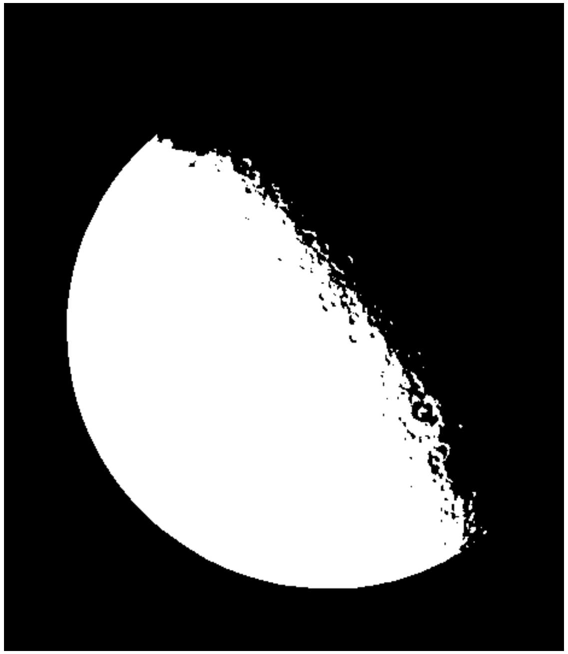




$$s = intensity_{max} - r$$

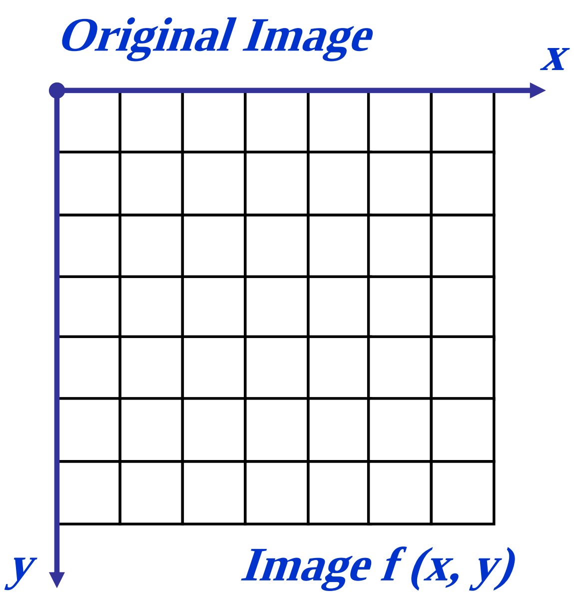
# Point Processing Example: Thresholding

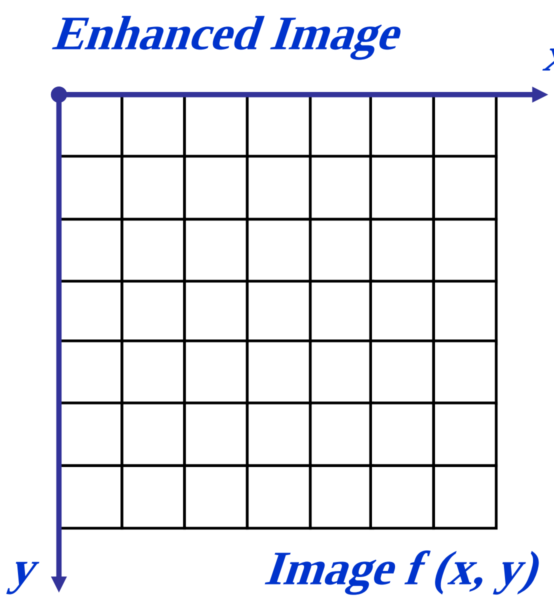
Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background

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



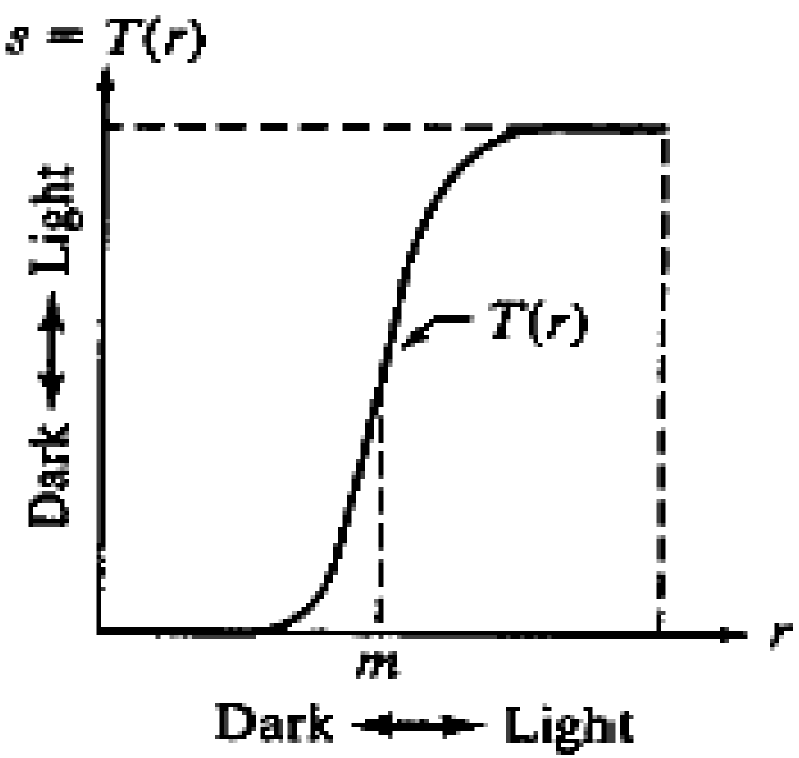
# Point Processing Example: Thresholding (cont…)

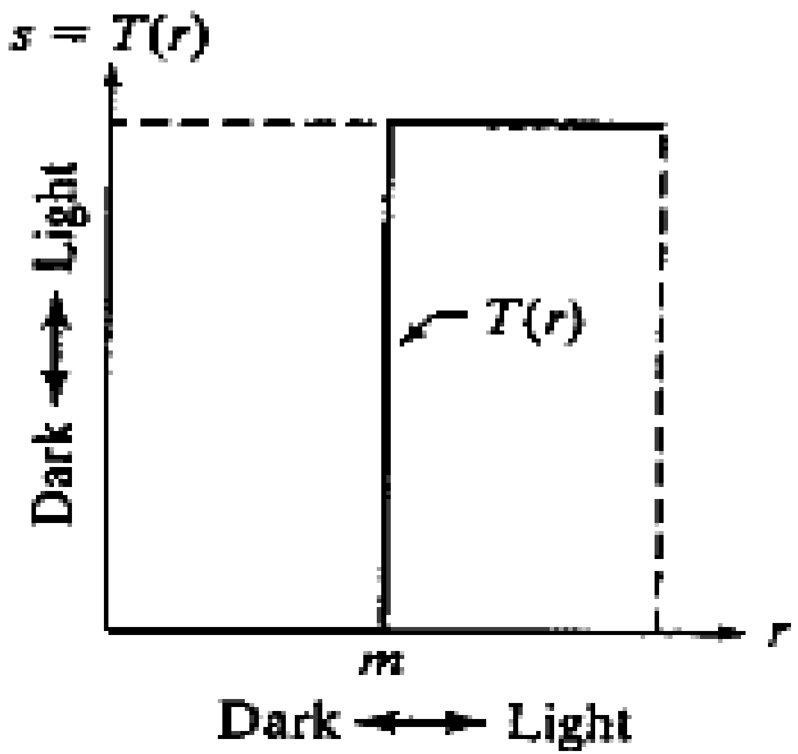



$$s = \begin{cases} 1.0 & r > threshold \\ 0.0 & r <= threshold \end{cases}$$

# thresholding function



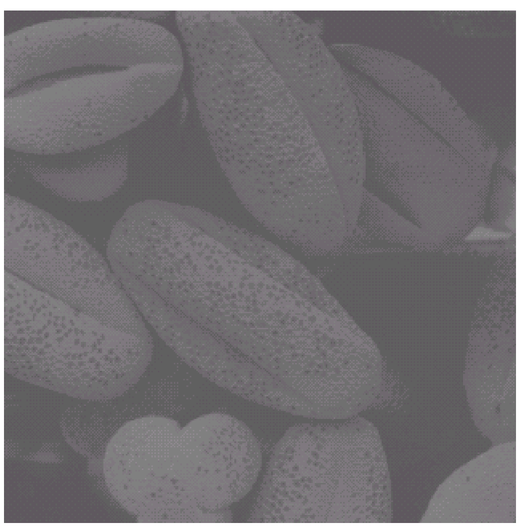



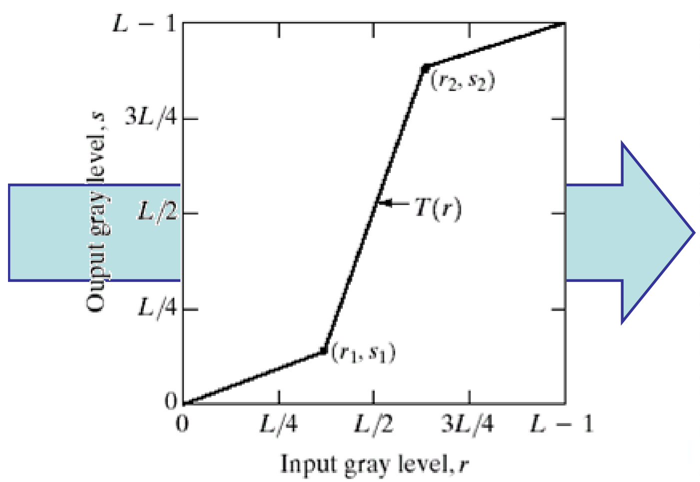

$$s = T(r) = \frac{1}{1 + (m/r)^E}$$

# Piecewise linear stretching function

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



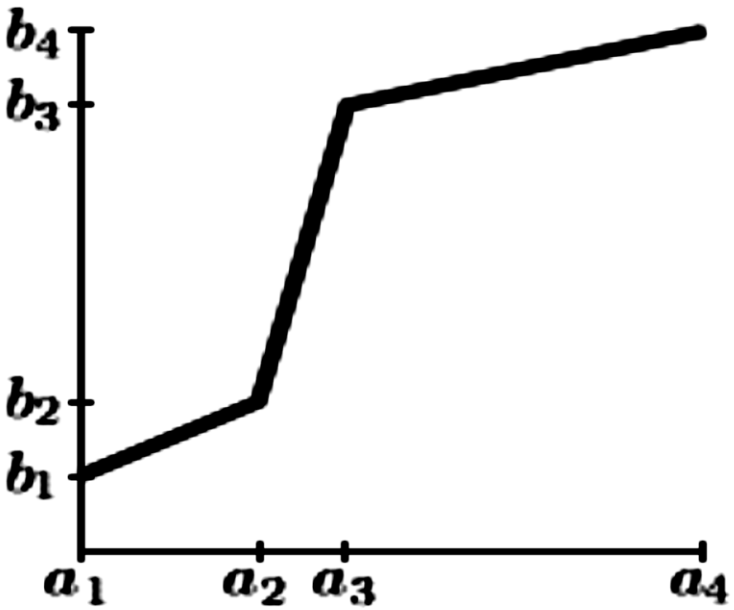



# Piecewise linear stretching function

We can easily write our own function to perform piecewise linear stretching as shown in Fig. To do this, we will make use of the **find** function, to find the pixel values in the image between ai and ai+1. Since the line between the coordinates (ai, bi) and (ai+1,bi+1) has the equation.

We can create a function to do the piecewise linear stretching

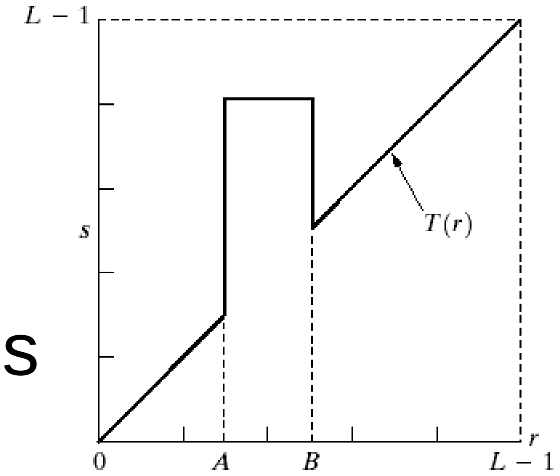
$$y = \frac{b_{i+1} - b_i}{a_{i+1} - a_i}(x - a_i) + b_i$$

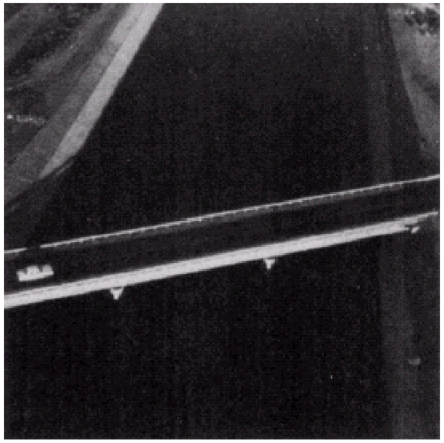


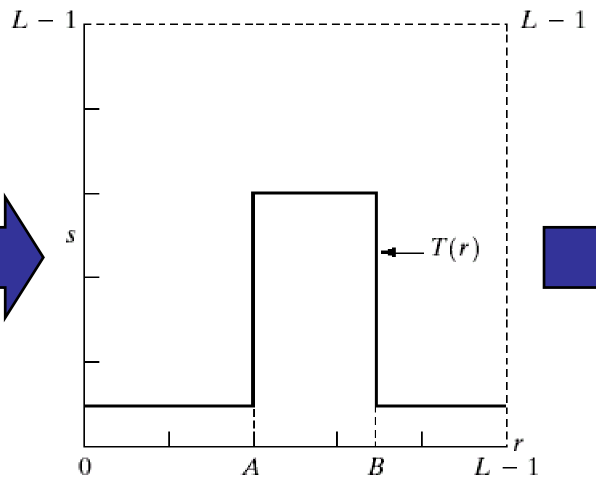
# Grey Level Slicing

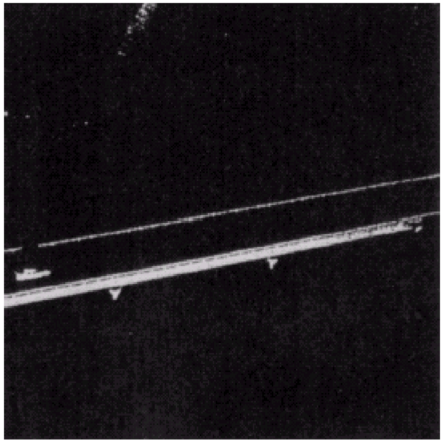
Highlights a specific range of grey levels

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





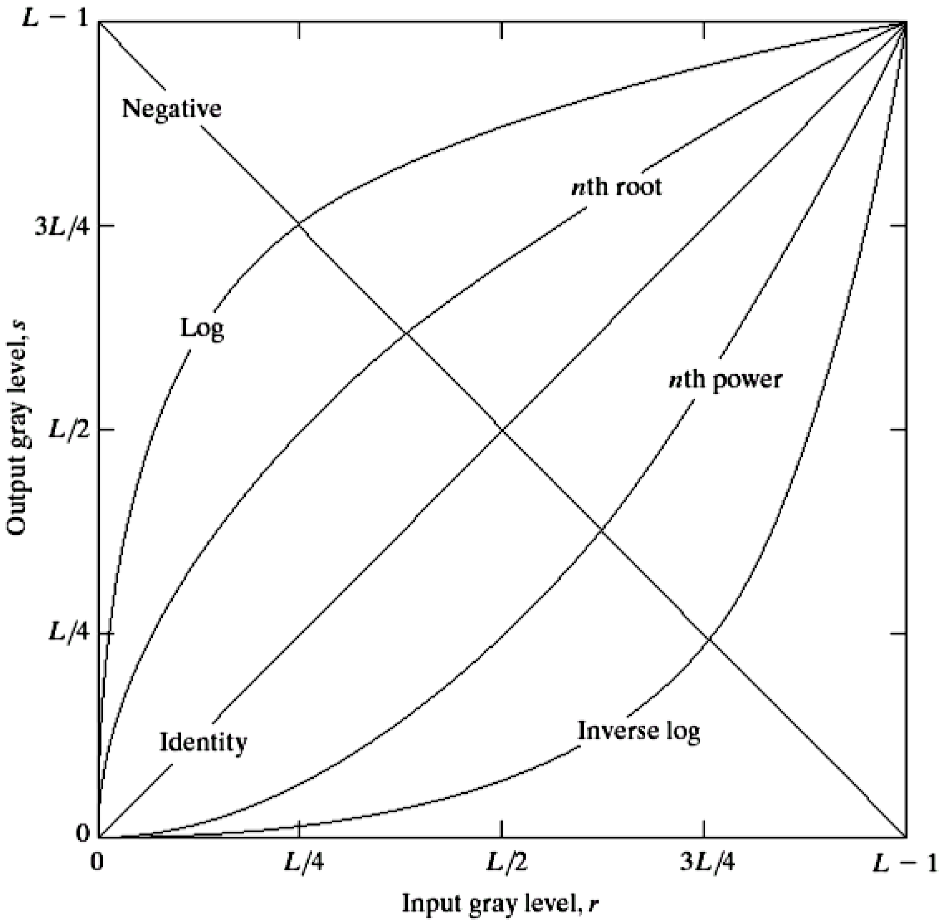



# Basic Grey Level Transformations

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

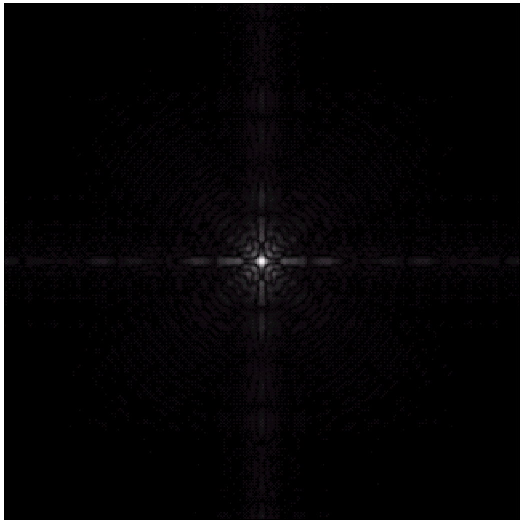
$$s = c * log(1 + r)$$

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

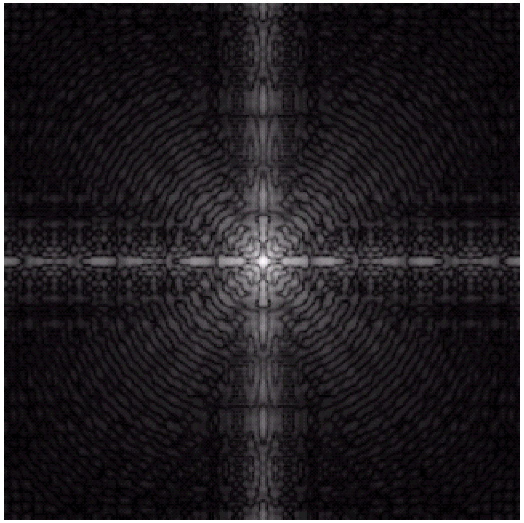
The inverse log transformation performs the opposite transformation

# Logarithmic Transformations (cont…)

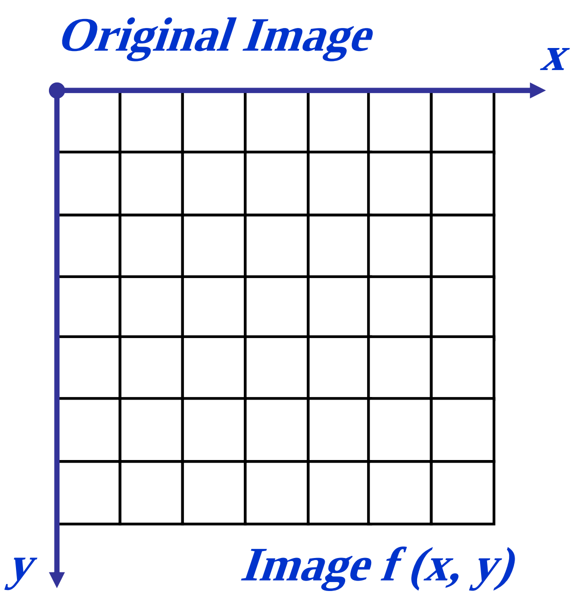
- Log functions are particularly useful when the input grey level values may have an extremely large range of values
- In the following example the Fourier transform of an image is put through a log transform to reveal more detail

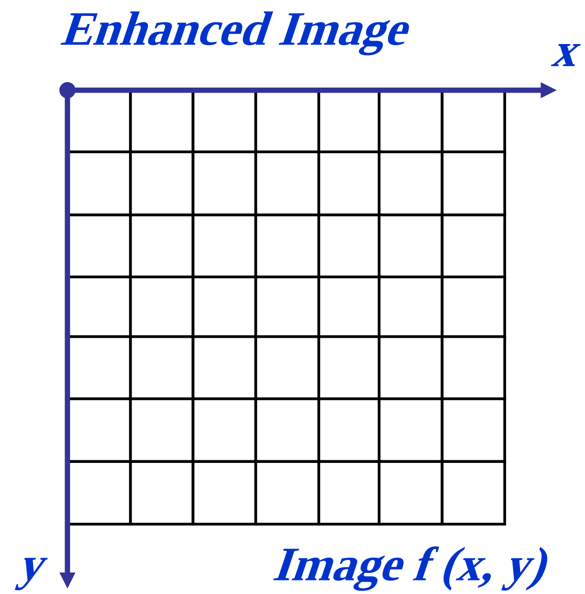


$s = \log(1 + r)$



# Logarithmic Transformations (cont…)




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

We usually set $c$ to 1

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

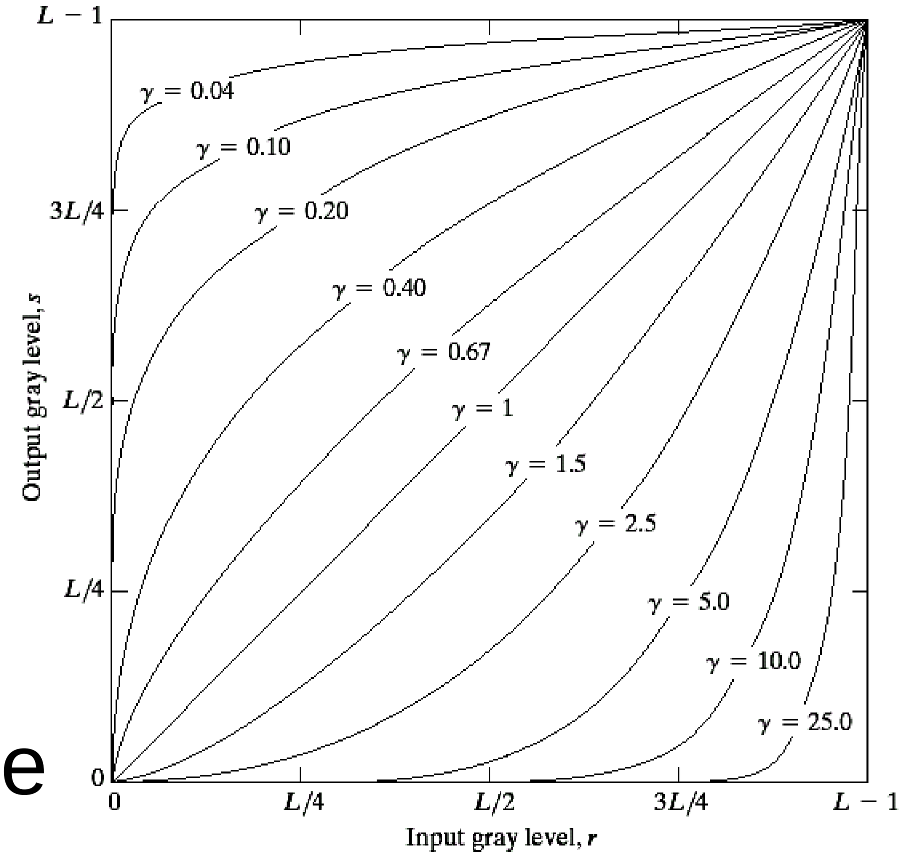
# Power Law Transformations

Power law transformations have the following form

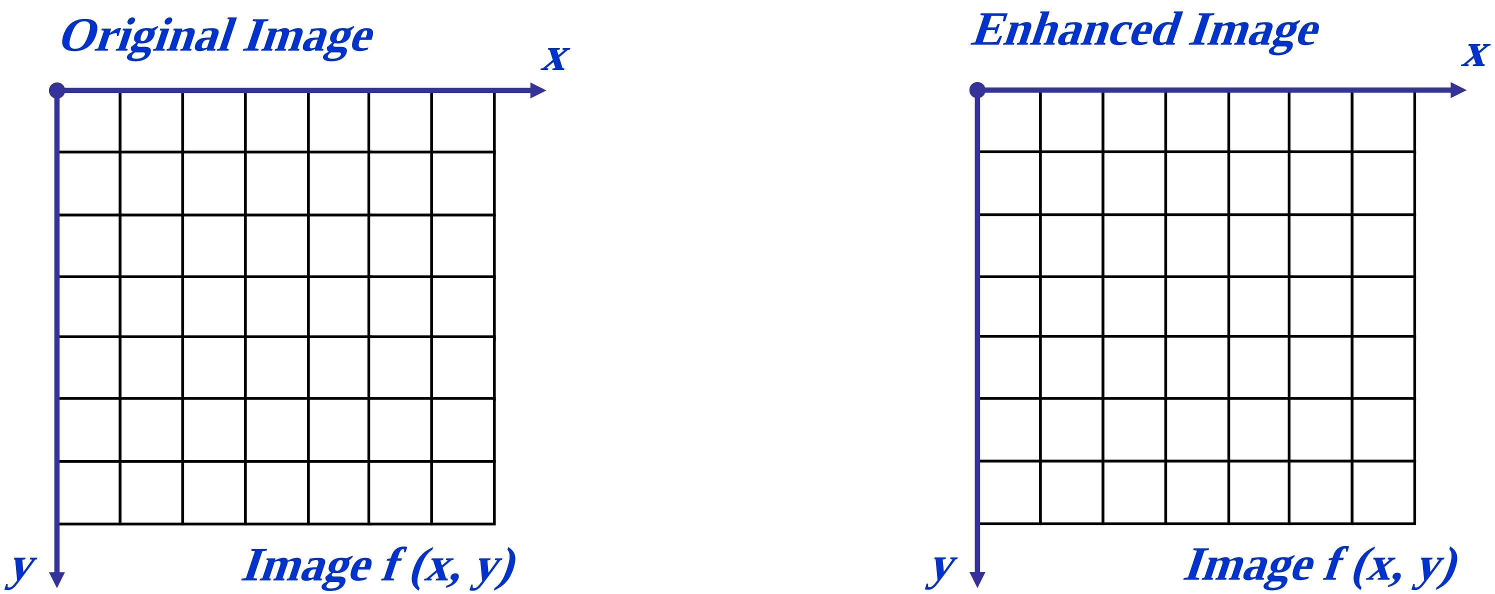
$$s = c * r^{\gamma}$$

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]

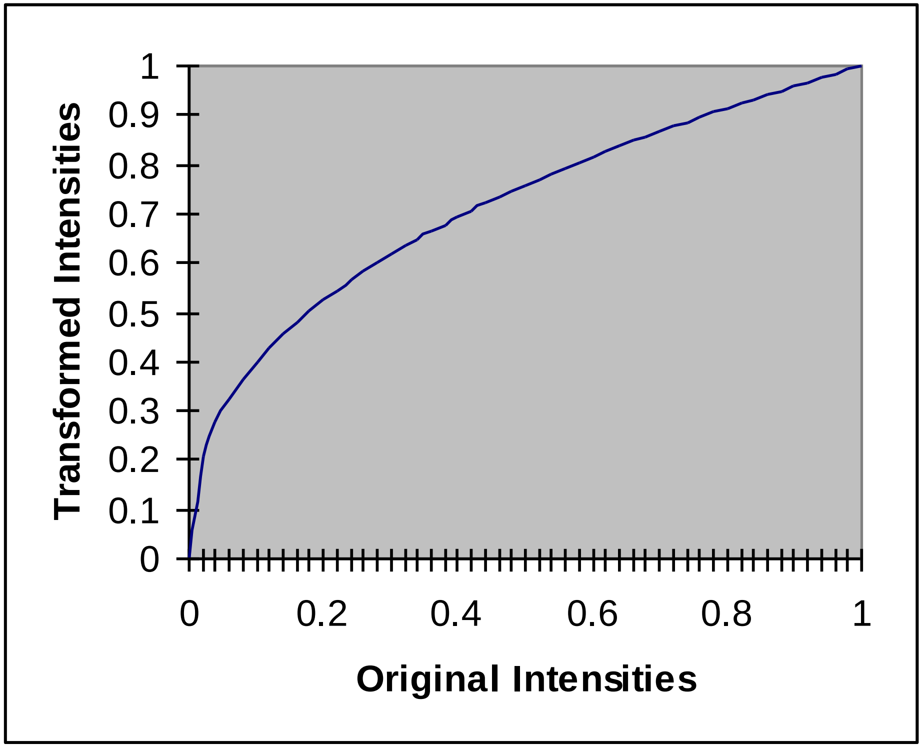
# Power Law Example

# Power Law Example (cont…)

$\gamma = 0.6$

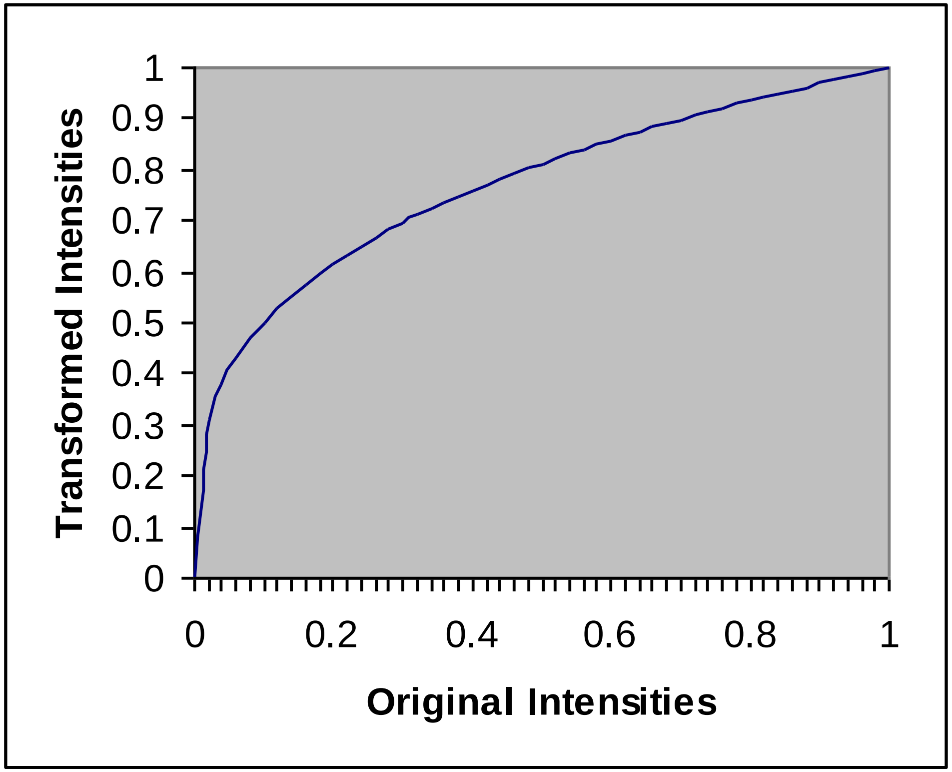

# Power Law Example (cont…)

$$\gamma = 0.4$$




# Power Law Example (cont…)

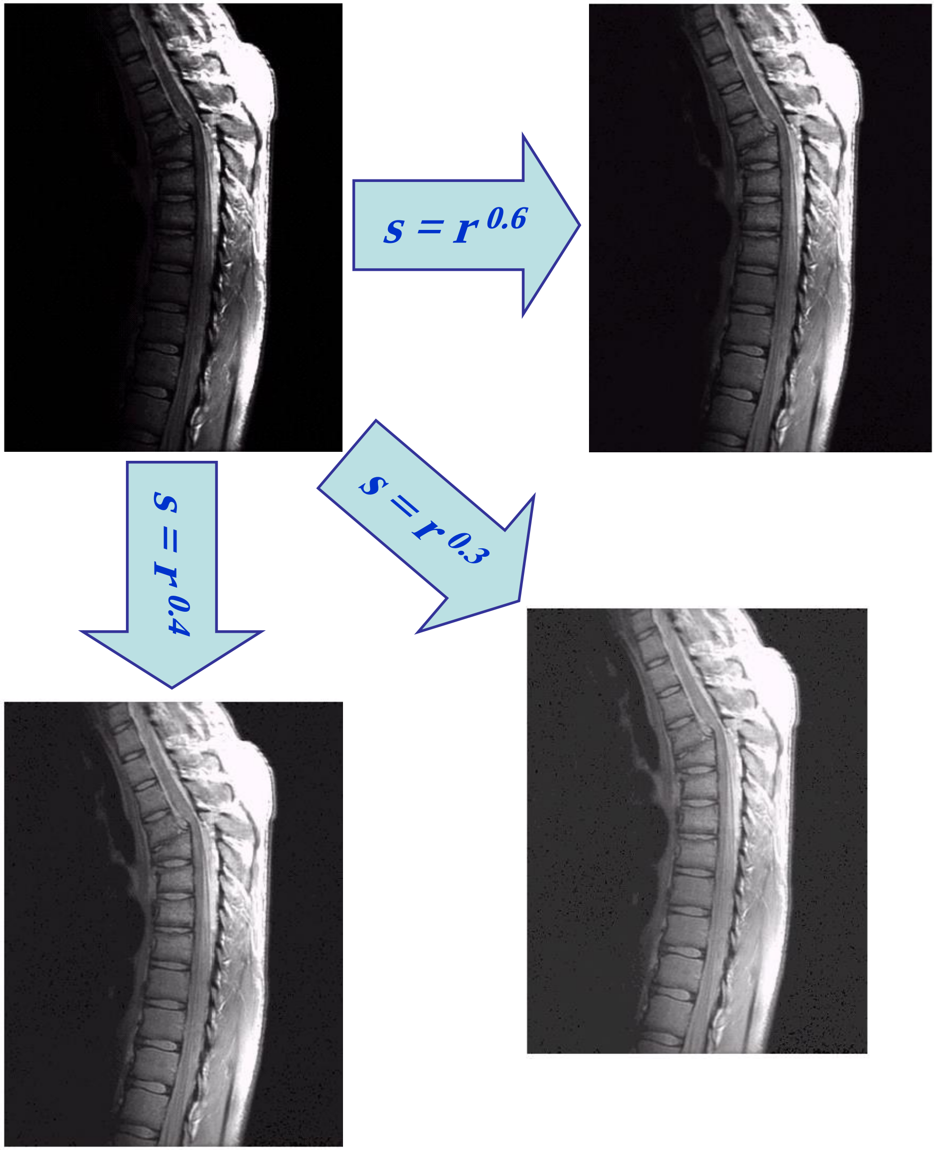
$\gamma = 0.3$




# Power Law Example (cont…)

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

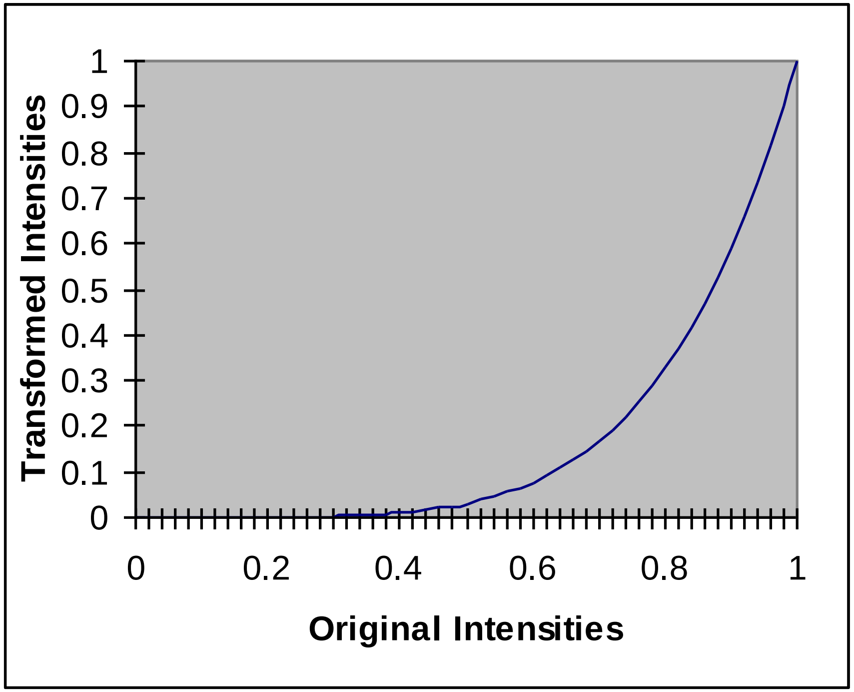
Different curves highlight different detail



# Power Law Example

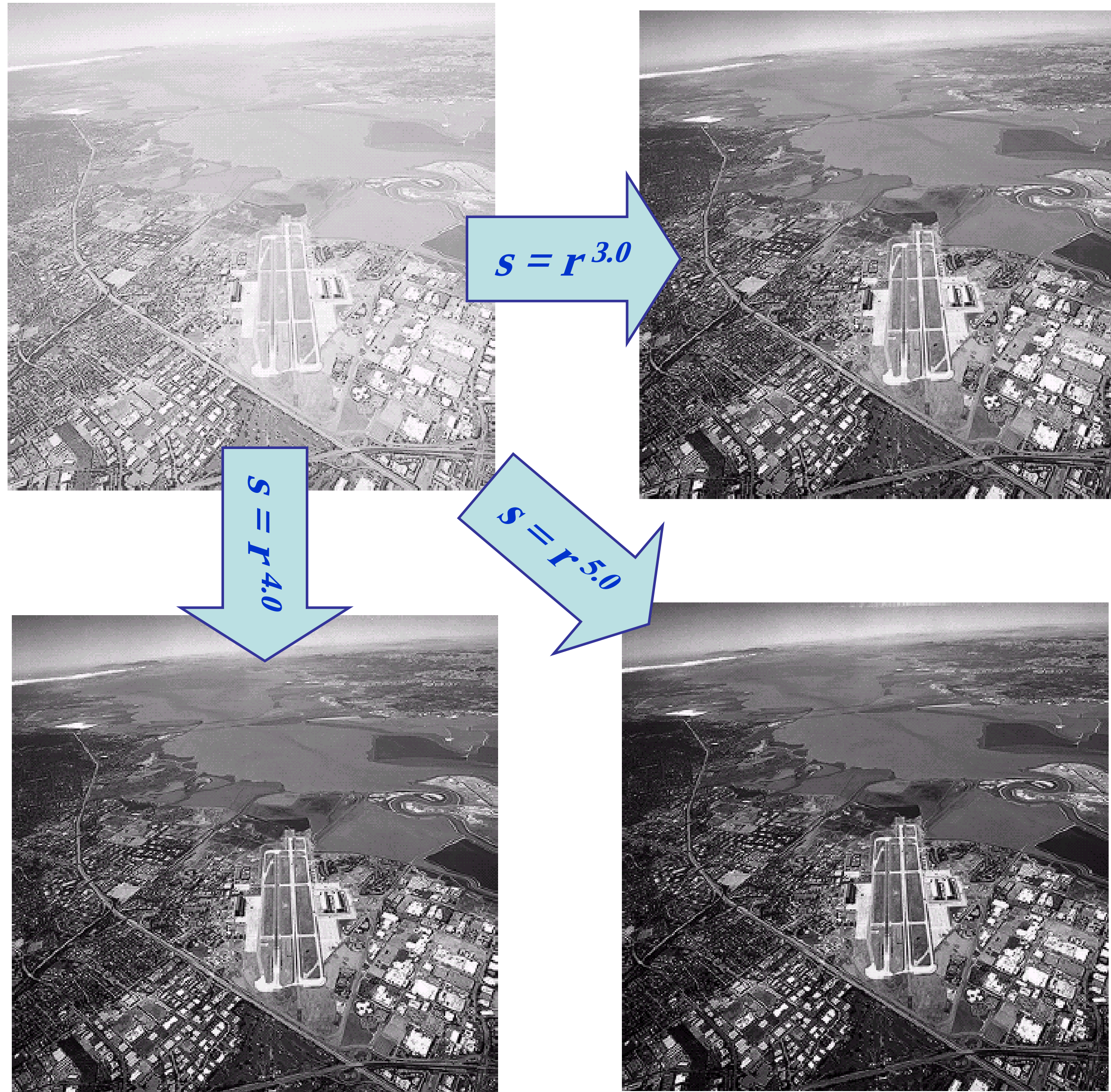
# Power Law Example (cont…)

$$\gamma = 5.0$$




# 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

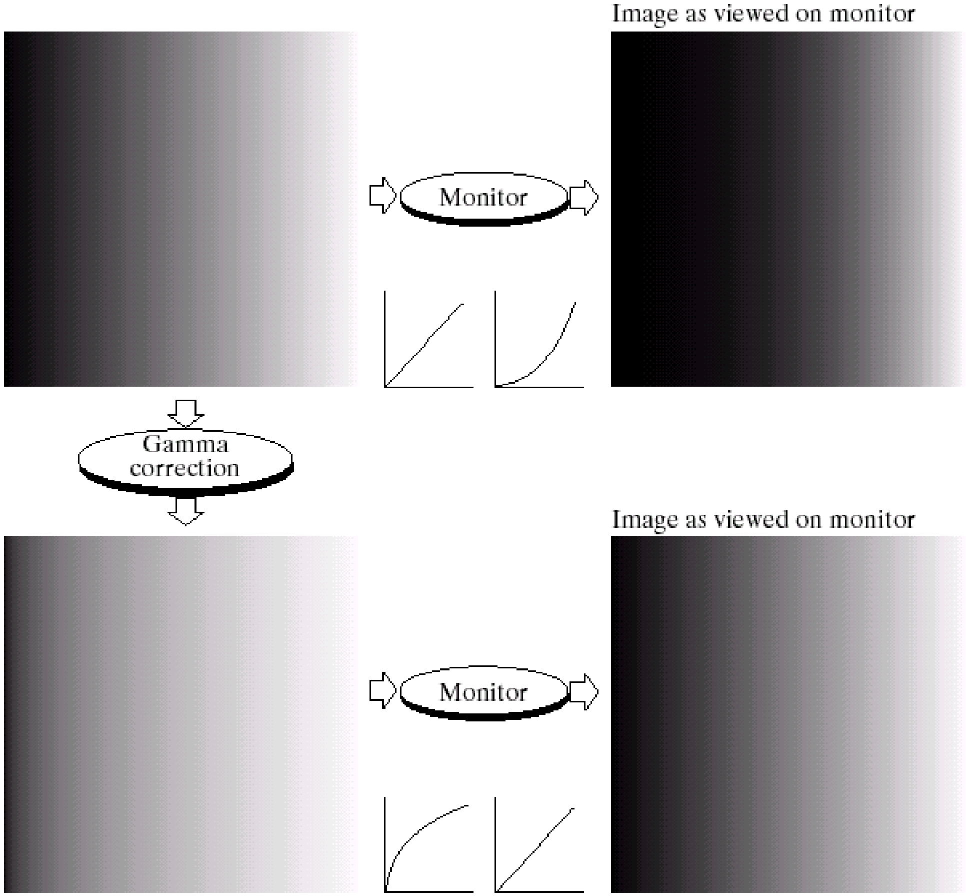


# Gamma Correction

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

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

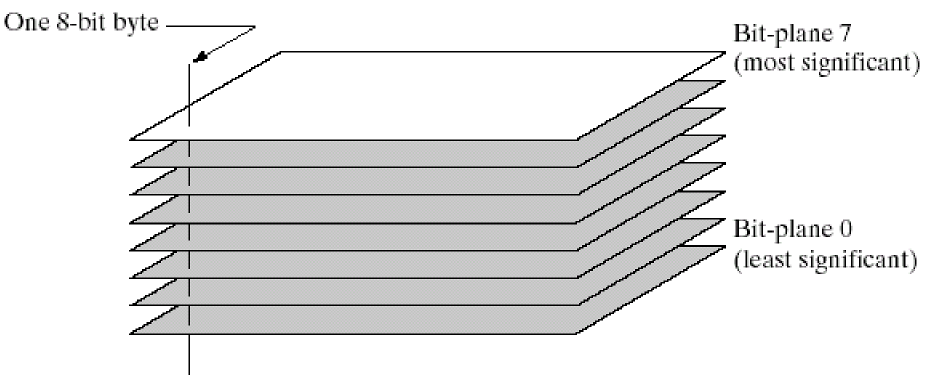
Can be corrected using a log transform

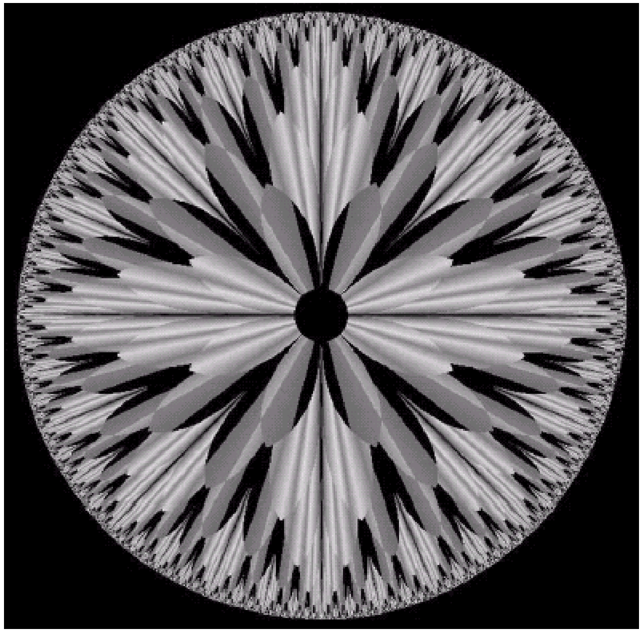


# Bit Plane Slicing

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

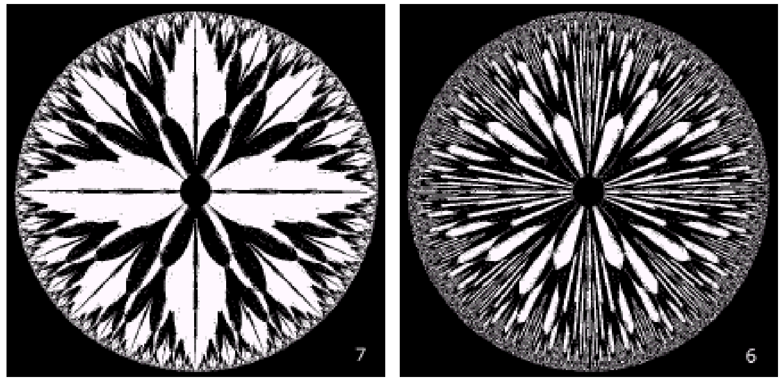
- Higher-order bits usually contain most of the significant visual information
- Lower-order bits contain subtle details



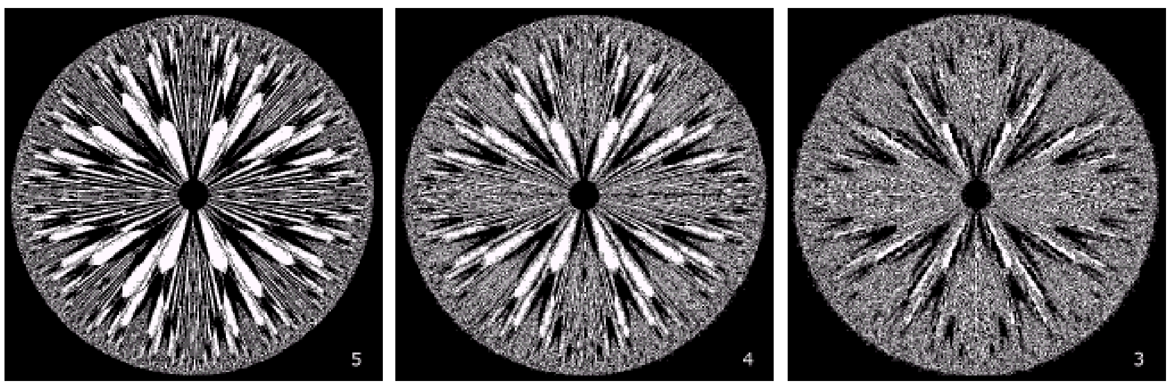
# Bit Plane Slicing (cont…)

[10000000]



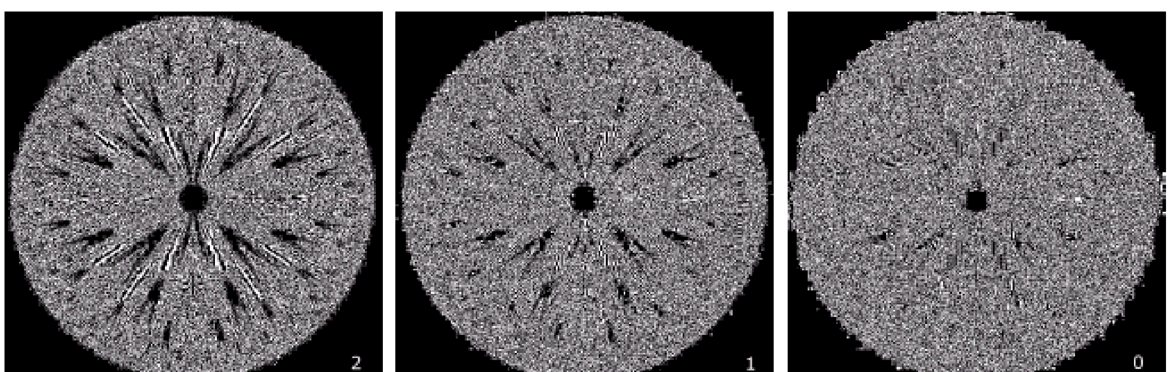

[01000000]

[00100000]




[00001000]

[00000100]




[00000001]

# Bit Plane Slicing (cont…)

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

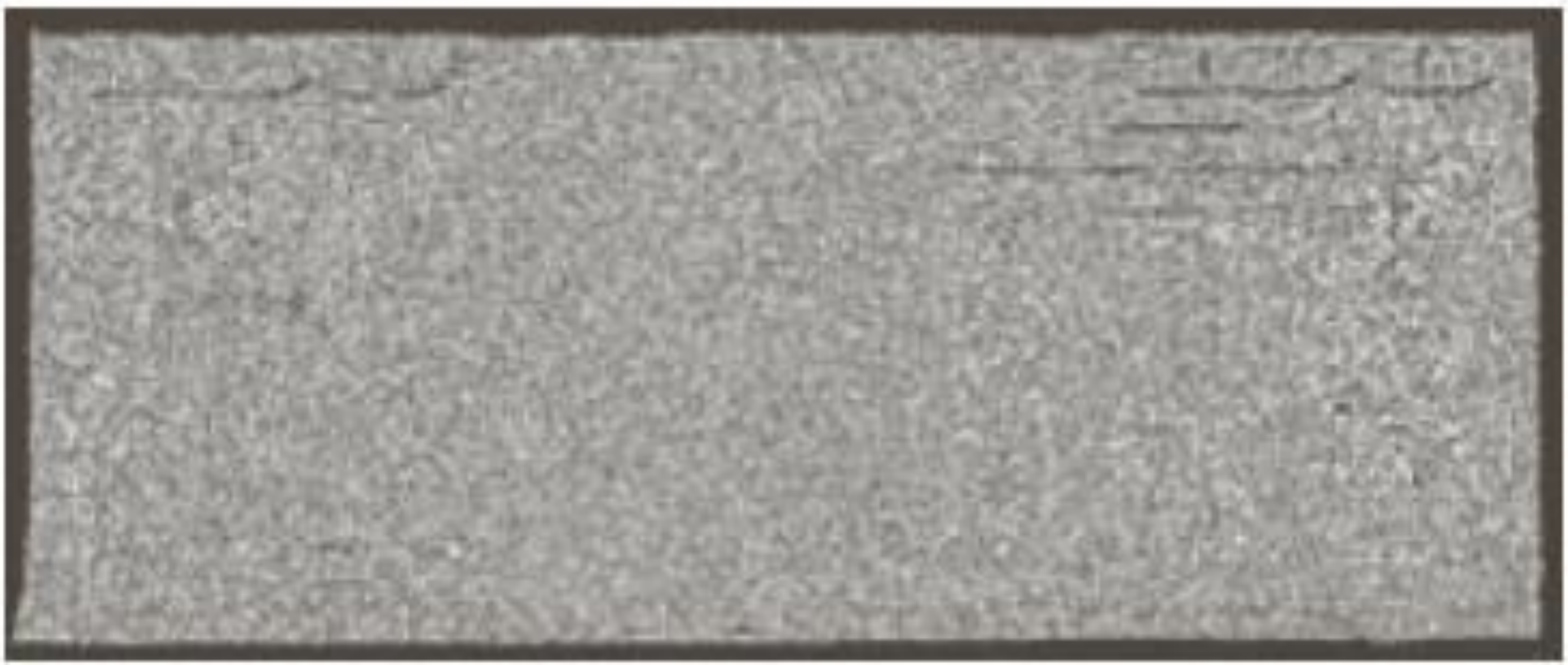
**FIGURE 3.14** (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

# Bit Plane Slicing (Bits 0…7)

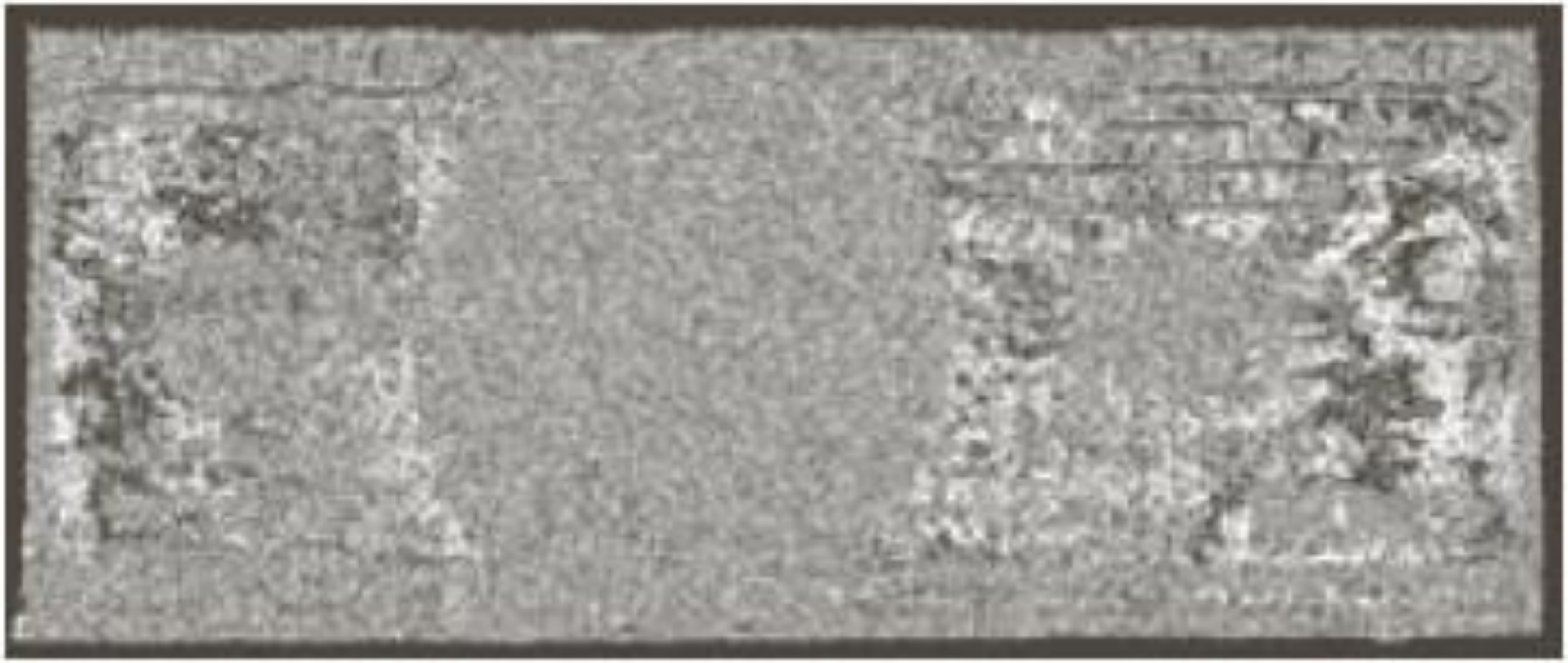
# Bit Plane Slicing (Bits 0)

# Bit Plane Slicing (Bits 1)

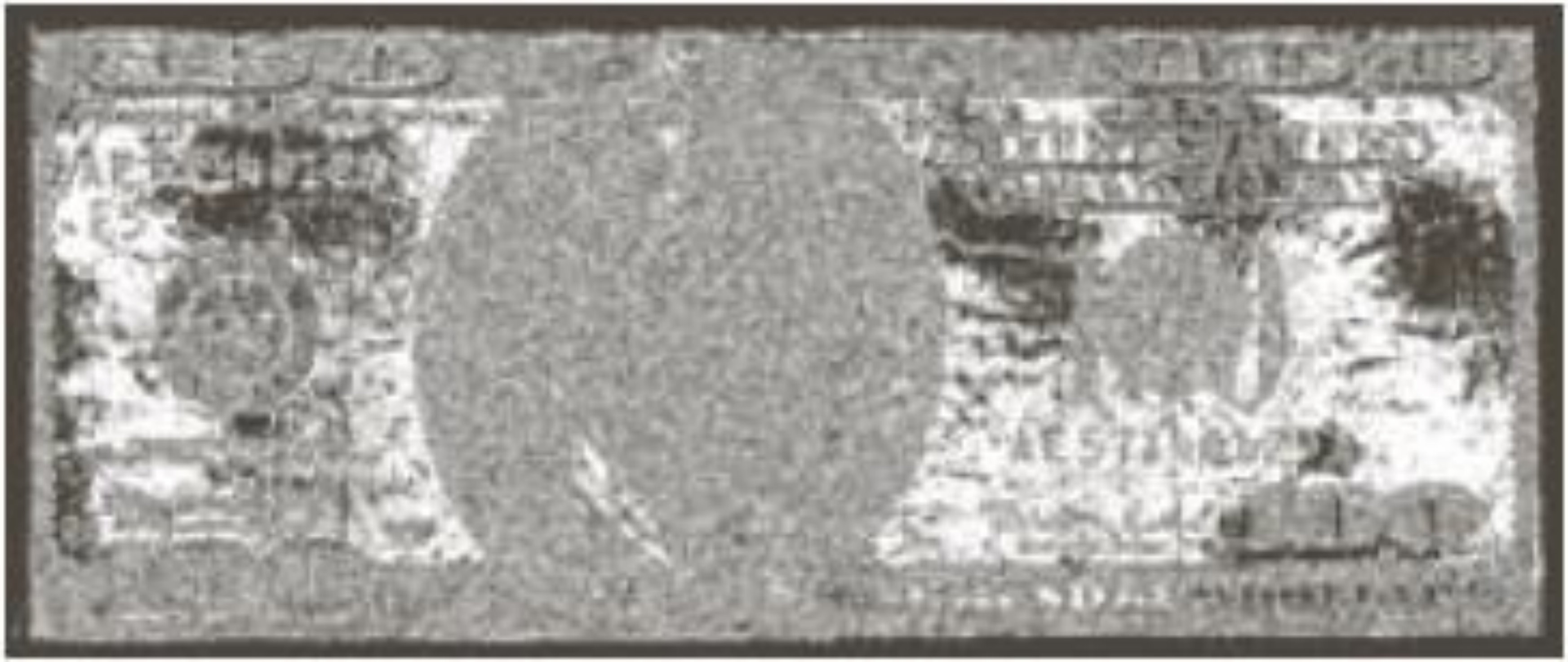
# Bit Plane Slicing (Bits 2)



# Bit Plane Slicing (Bits 3)



# Bit Plane Slicing (Bits 4)



# Bit Plane Slicing (Bits 5)

# Bit Plane Slicing (Bits 6)

# Bit Plane Slicing (Bits 7)

# 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

# Summary

We have looked at different kinds of point processing image enhancement

Next time we will start to look at image Histogram processing then neighbourhood operations – in particular *filtering* and *convolution*

1. *Wavelet and DIP*
2. *Image Segmentation(Thresholding)*
3. *Image Segmentation(Points,Lines)*
4. *Image Degradation models*
5. *Image Restoration*
6. *Graphics Processing Unit GPU*
7. *Computer vision*
8. *3d scanner`*
9. *Image Lossy Compression (*Predictive*)*
10. *Image Lossy Compression (Wavelet )*
11. *Image Lossy Compression (*Variable-Length*)*
12. *Image lossless Compression (*LZW*)*
13. *Image lossless Compression (*Predictive*)*
14. *Morphological Image Processing*
15. *Color models*
16. *Image file format*
17. *Image Processor*
18. *Dilation and erosion*
19. *Opening and closing*
20. *The Hough transform*
21. *The distance transform*
22. *The hit-or-miss transform*

DIP

LECTURE No. ( 4 )

# Image Enhancement (Histogram Processing)

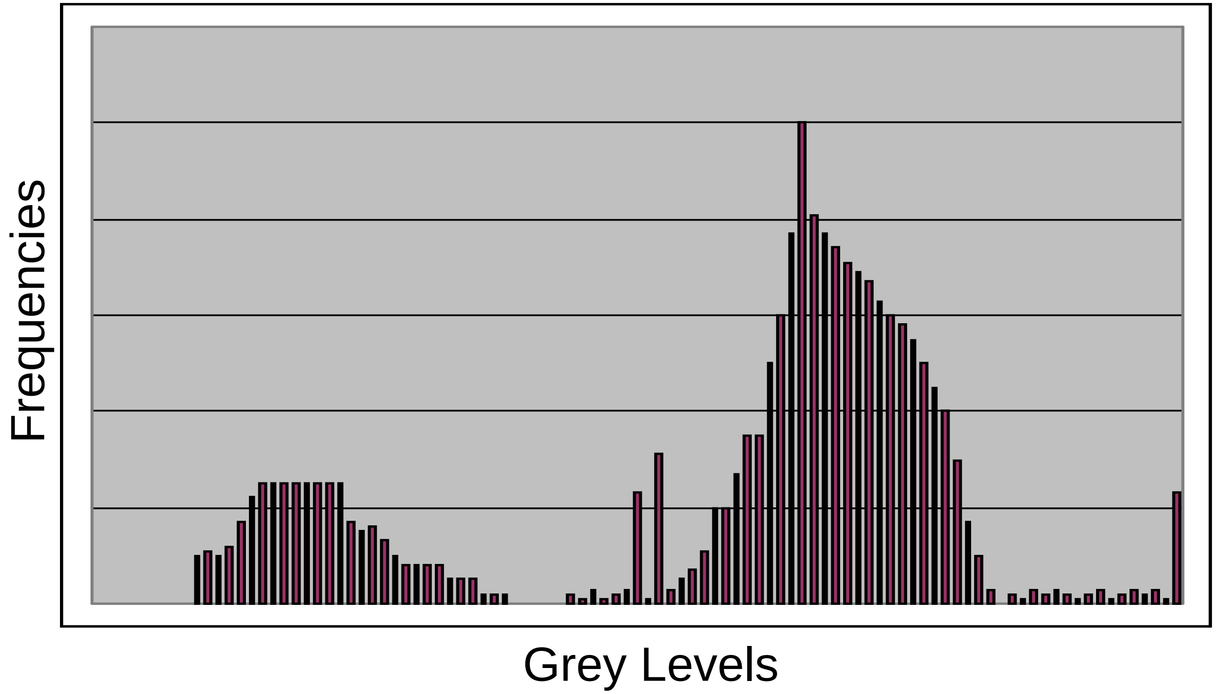
10/14/2019

Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

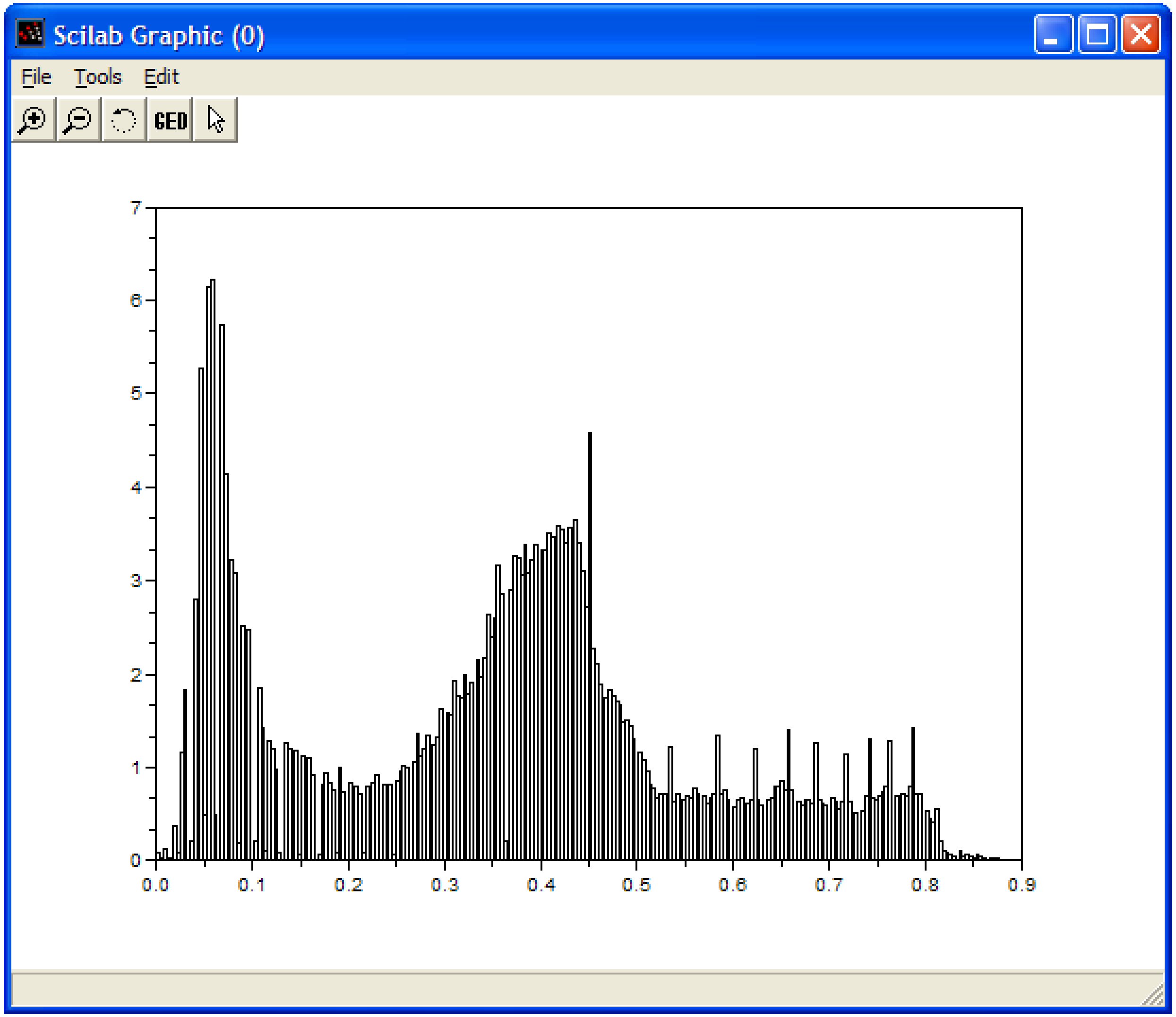
# Image Histograms

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

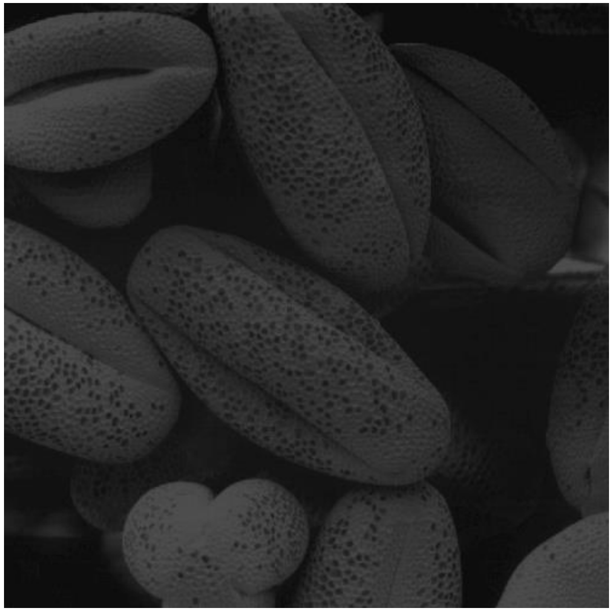
Massively useful in image processing, especially in segmentation

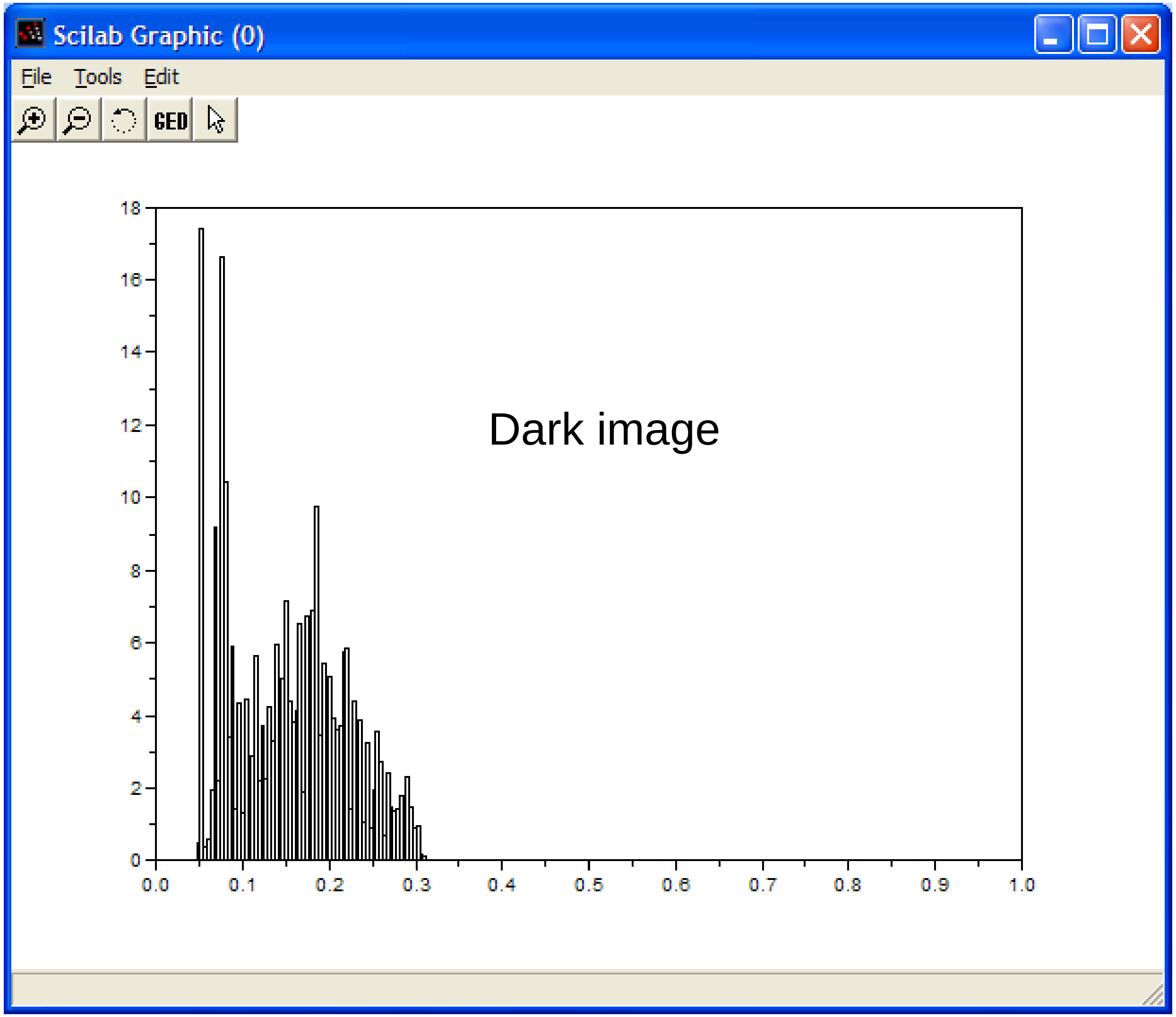


# Histogram Examples (cont…)

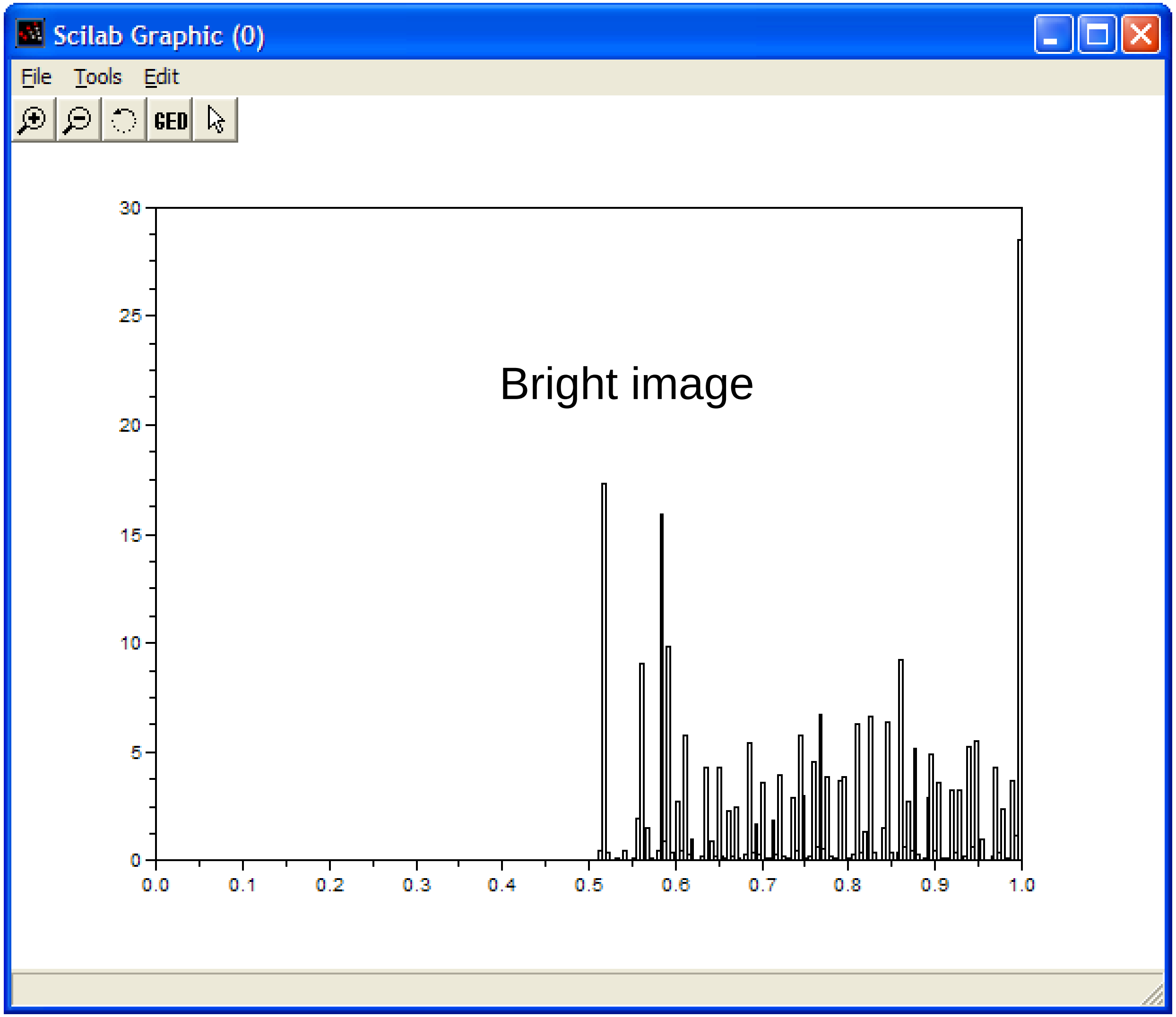


# Histogram Examples (cont…)

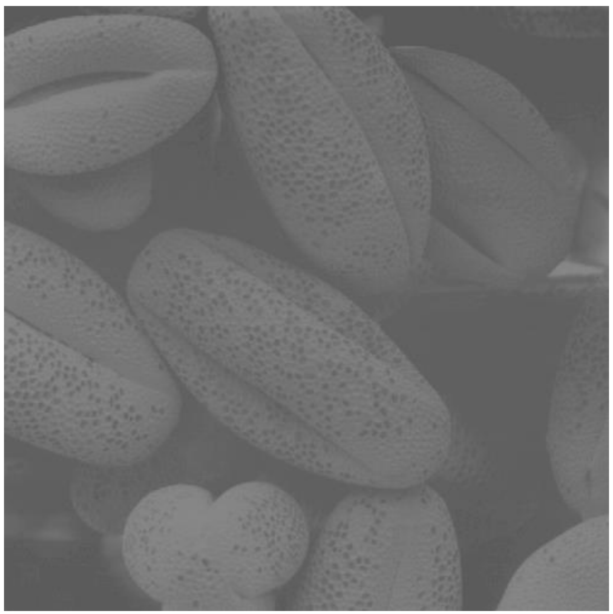


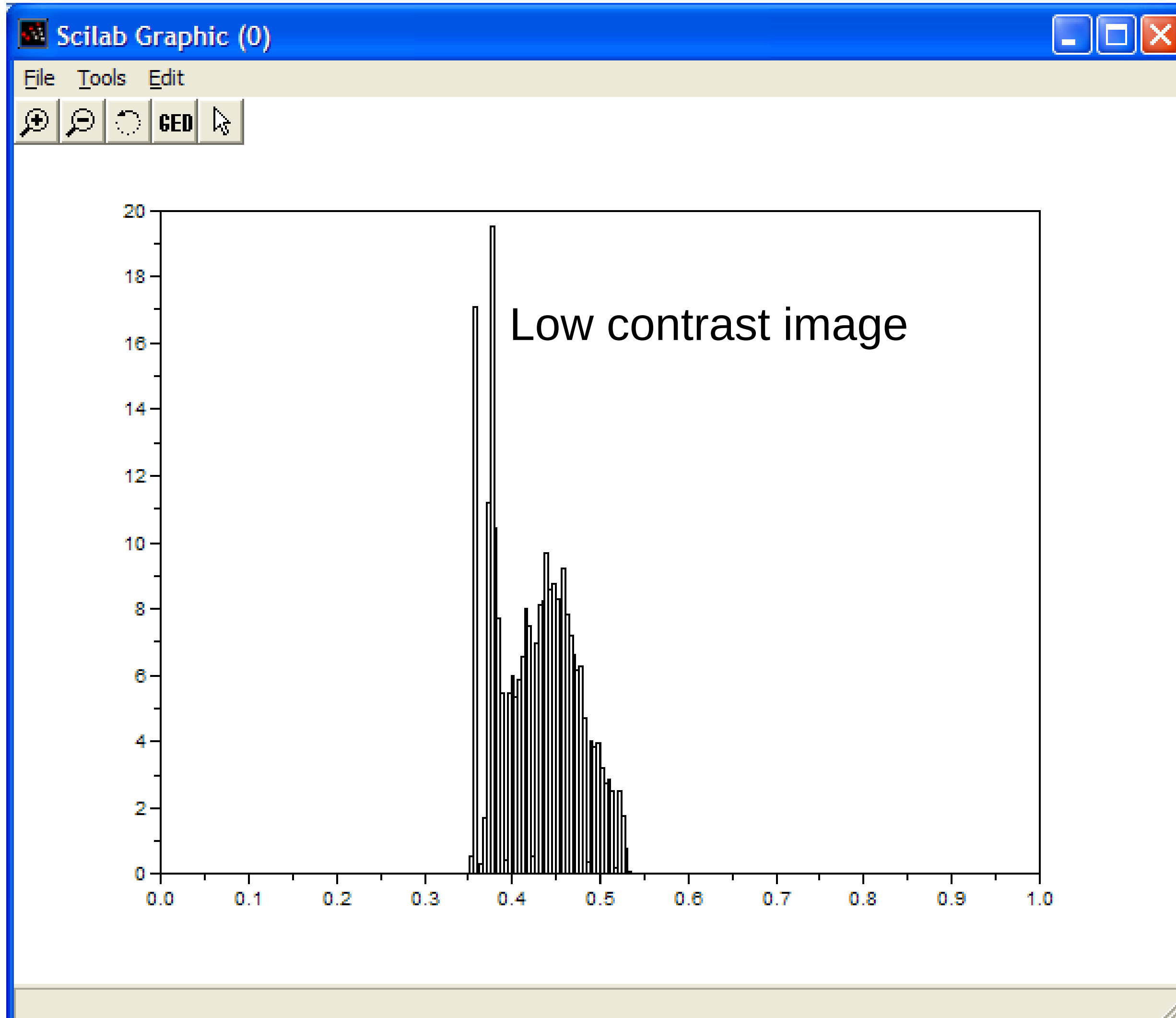


# Histogram Examples (cont…)

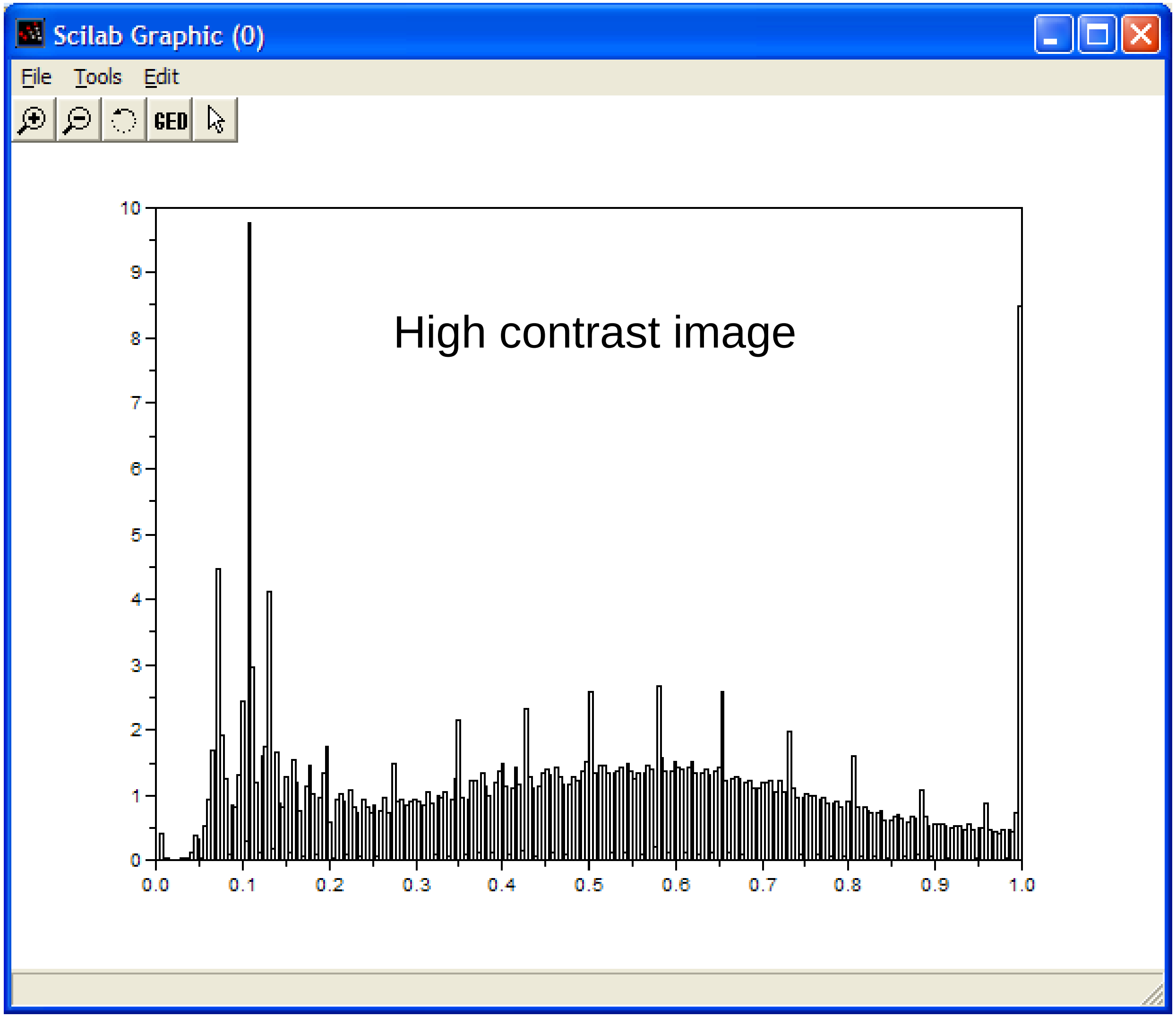


# Histogram Examples (cont…)





# Histogram Examples (cont…)

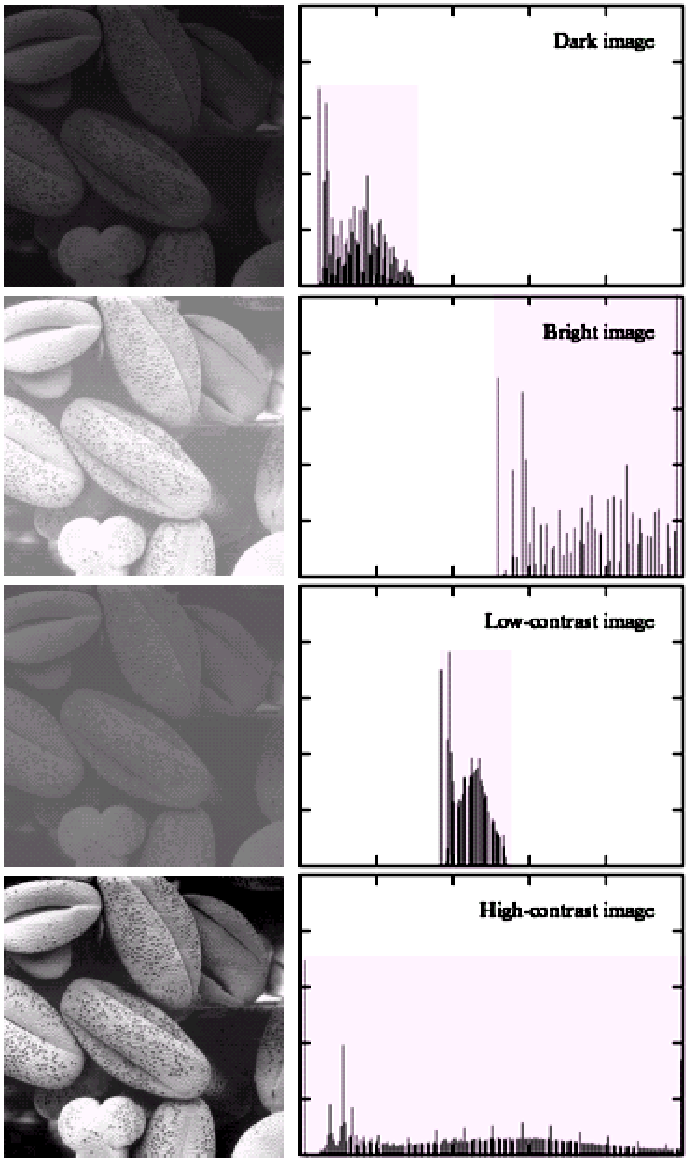


# Histogram Examples (cont…)

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

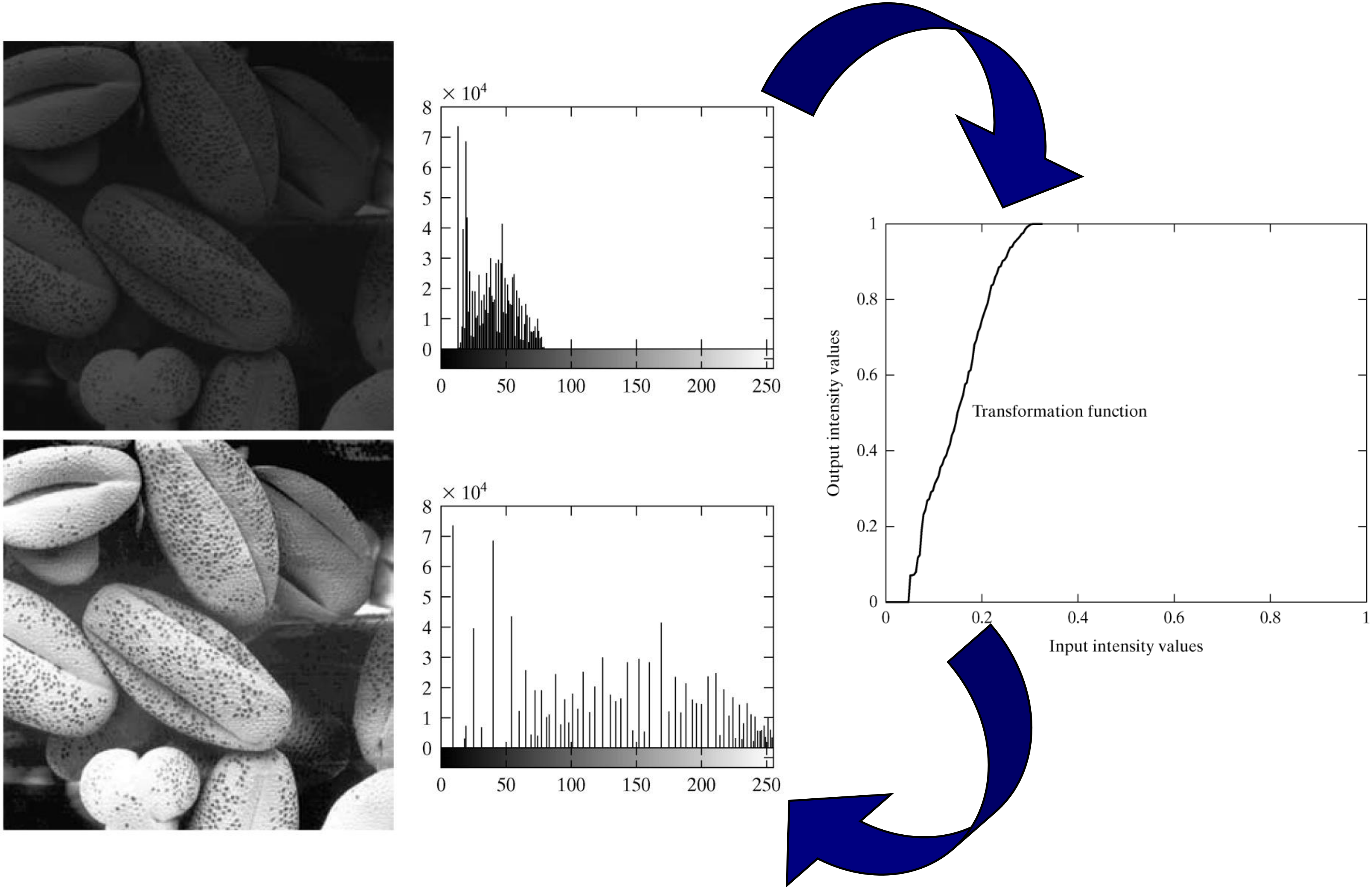
Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

The formula for histogram equalisation is given where

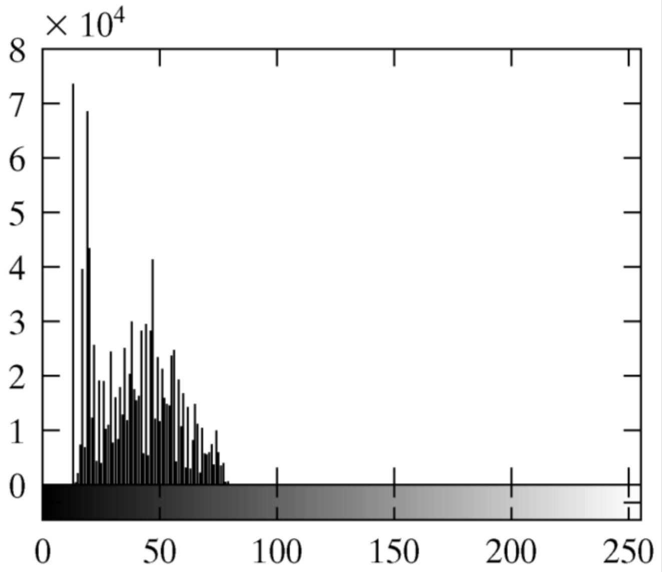
- $r_k$: input intensity
- $s_k$: processed intensity
- $k$: the intensity range (e.g 0.0 – 1.0)
- $n_j$: the frequency of intensity $j$
- $n$: the sum of all frequencies

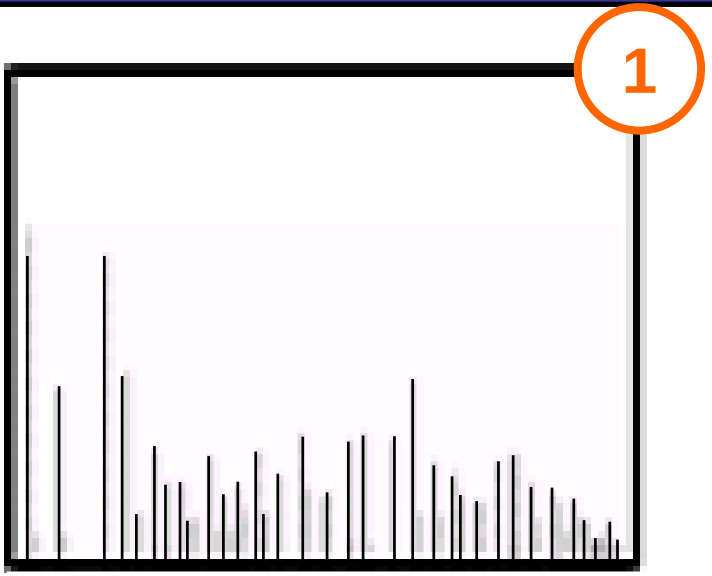
$$\begin{aligned} s_k &= T(r_k) \\ &= \sum_{j=1}^{k} p_r(r_j) \\ &= \sum_{j=1}^{k} \frac{n_j}{n} \end{aligned}$$

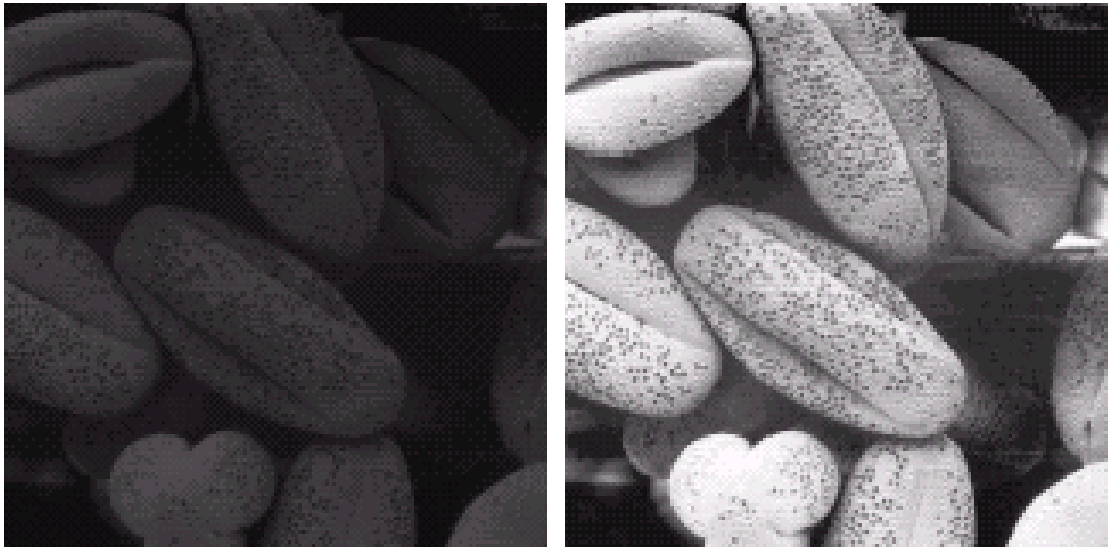
# Equalisation Transformation Function



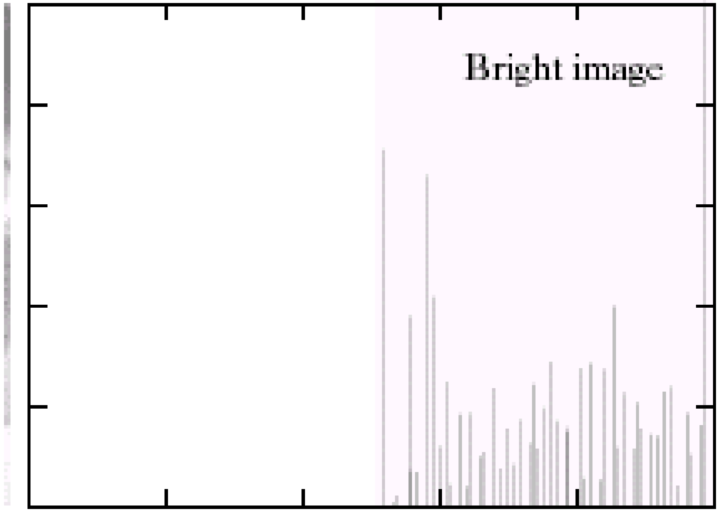
# Equalisation Examples

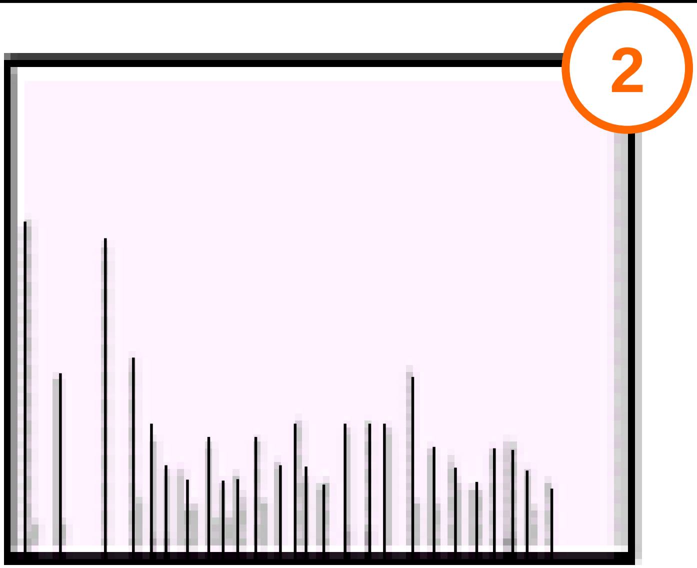


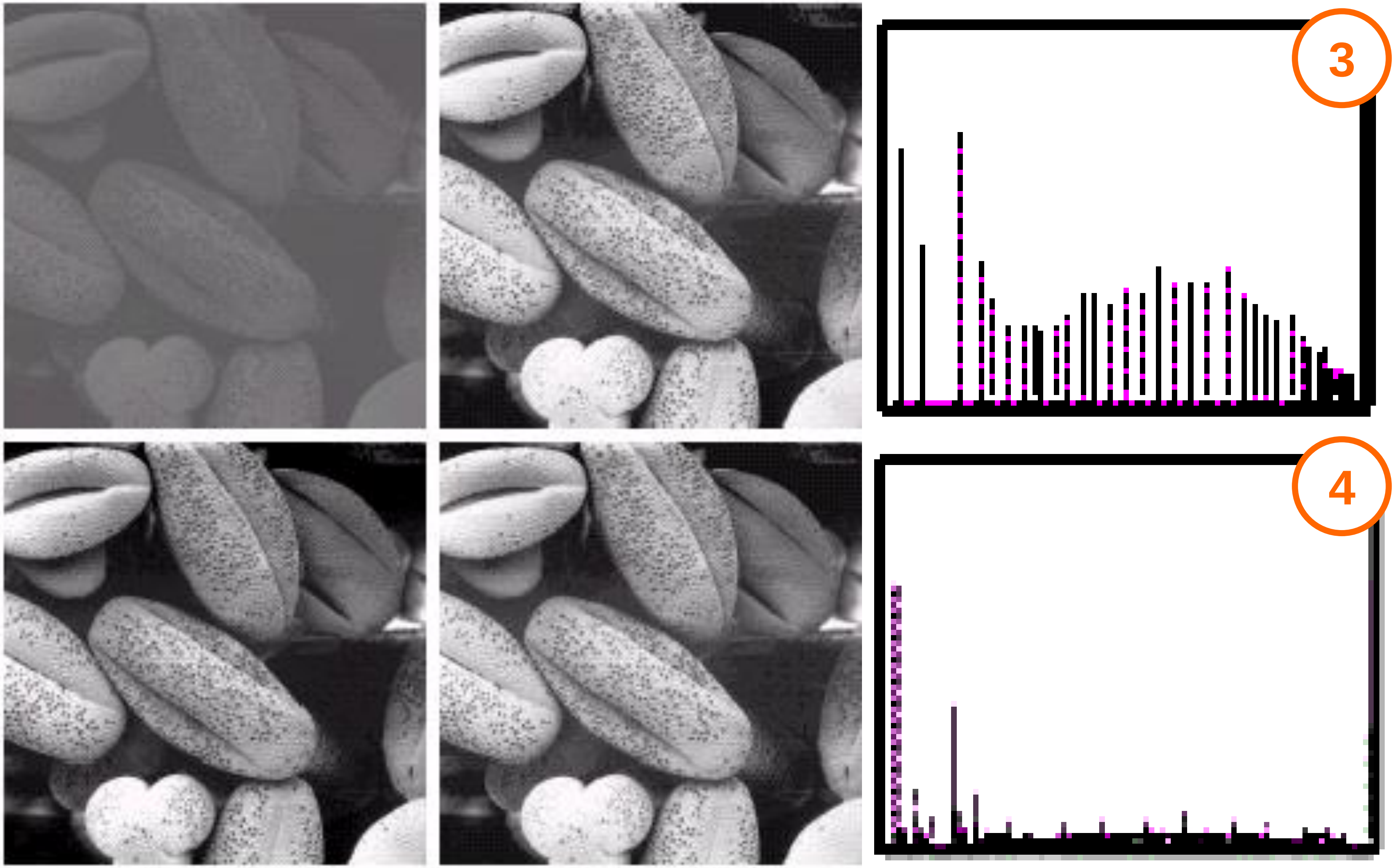



# Equalisation Examples

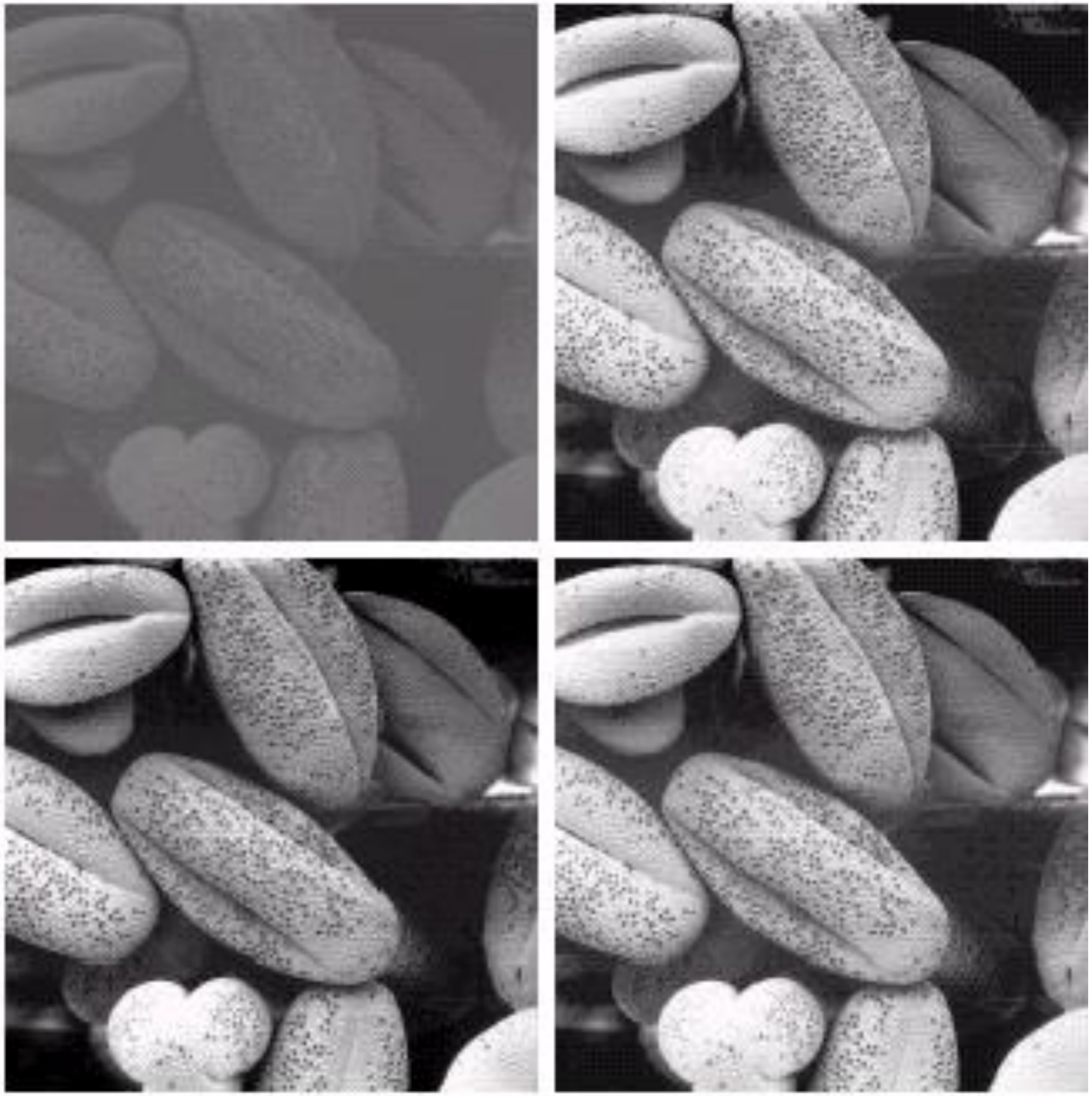


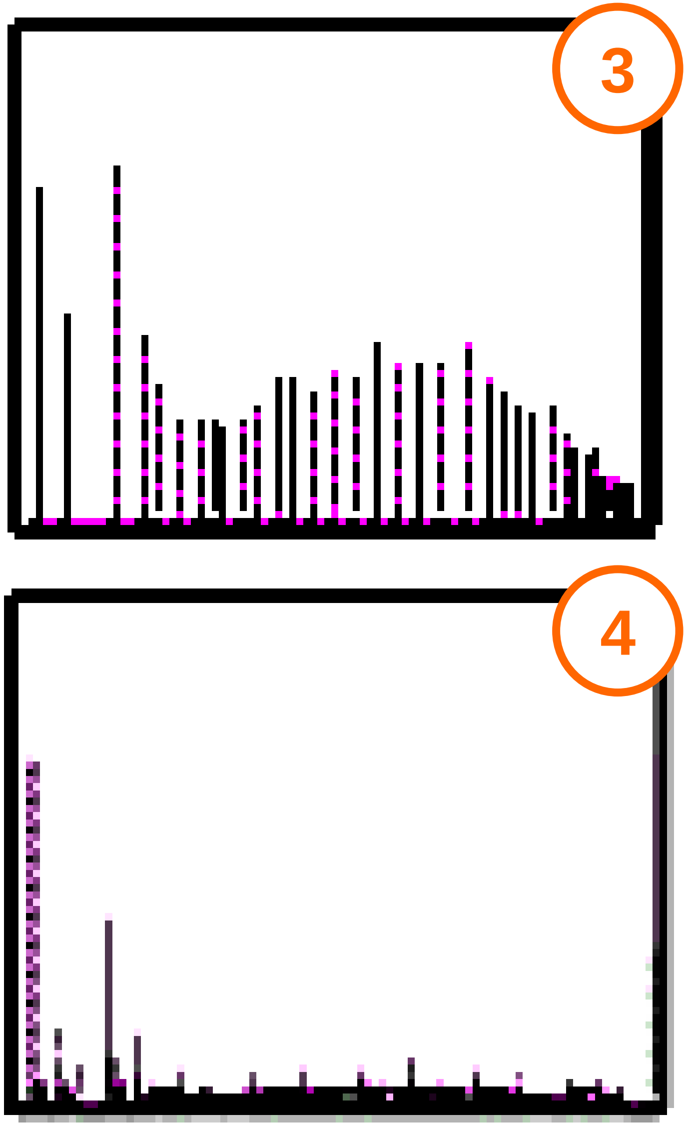




# Equalisation Examples (cont…)



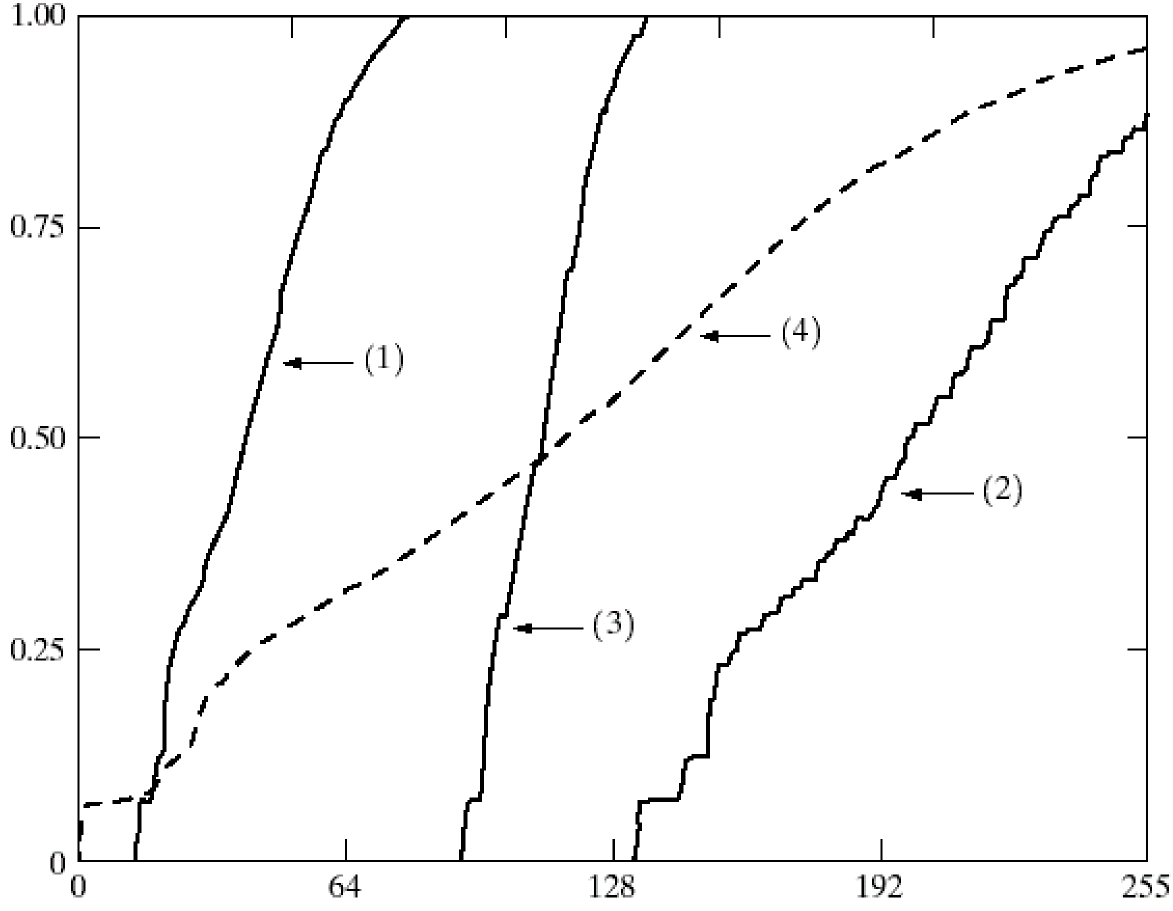
# Equalisation Examples (cont…)





# Equalisation Transformation Functions

The functions used to equalise the images in the previous examples

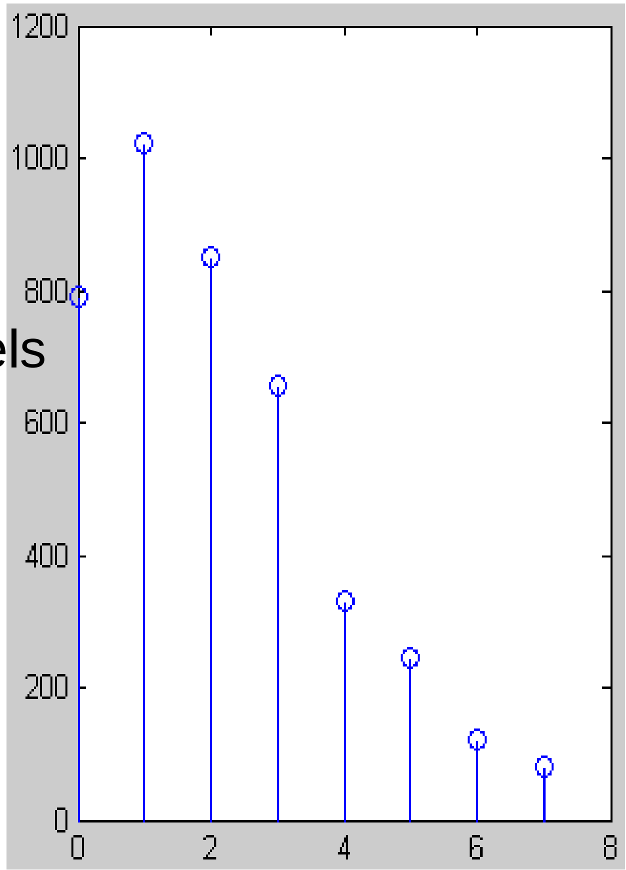


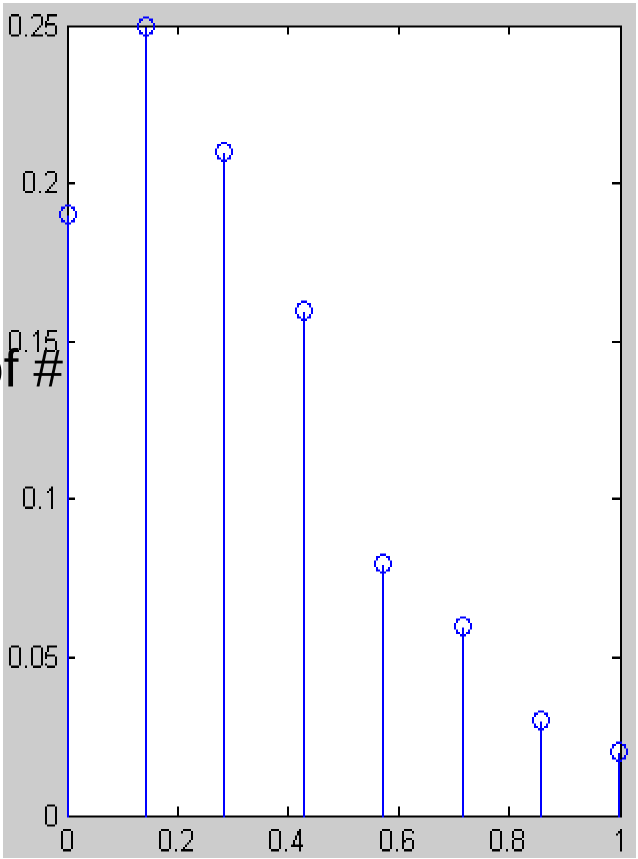
# Histogram Equalization

- Example:
- Consider an 8-level 64 x 64 image with gray values (0, 1, …,7). The normalized gray values are (0, 1/7, 2/7, …, 1). The normalized histogram is given below:
- The gray values in output are also (0, 1/7, 2/7, …, 1).

| $k$ | $r_k$ | $n_k$ | $p(r_k) = n_k/n$ |
|---|---|---|---|
| 0 | 0 | 790 | 0.19 |
| 1 | 1/7 | 1023 | 0.25 |
| 2 | 2/7 | 850 | 0.21 |
| 3 | 3/7 | 656 | 0.16 |
| 4 | 4/7 | 329 | 0.08 |
| 5 | 5/7 | 245 | 0.06 |
| 6 | 6/7 | 122 | 0.03 |
| 7 | 1 | 81 | 0.02 |

# Histogram Equalization



# Histogram Equalization

- Applying the transformation, $s_k = T(r_k) = \sum_{j=0}^{k} p_{in}(r_j)$ we have

$$s_0 = T(r_0) = \sum_{j=0} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) = 0.19 \rightarrow 1/7$$

$$s_1 = T(r_1) = \sum_{j=0}^{1} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) = 0.44 \rightarrow 3/7$$

$$s_2 = T(r_2) = \sum_{j=0}^{2} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + p_{\text{in}}(r_2) = 0.65 \rightarrow 5/7$$

$$s_3 = T(r_3) = \sum_{j=0}^{3} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_3) = 0.81 \rightarrow 6/7$$

$$s_4 = T(r_4) = \sum_{j=0}^{4} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_4) = 0.89 \rightarrow 6/7$$

$$s_5 = T(r_5) = \sum_{j=0}^{5} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_5) = 0.95 \rightarrow 1$$

$$s_6 = T(r_6) = \sum_{j=0}^{6} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_6) = 0.98 \rightarrow 1$$

$$s_7 = T(r_7) = \sum_{j=0}^{7} p_{\text{in}}(r_j) = p_{\text{in}}(r_0) + p_{\text{in}}(r_1) + \cdots + p_{\text{in}}(r_7) = 1.00 \rightarrow 1$$

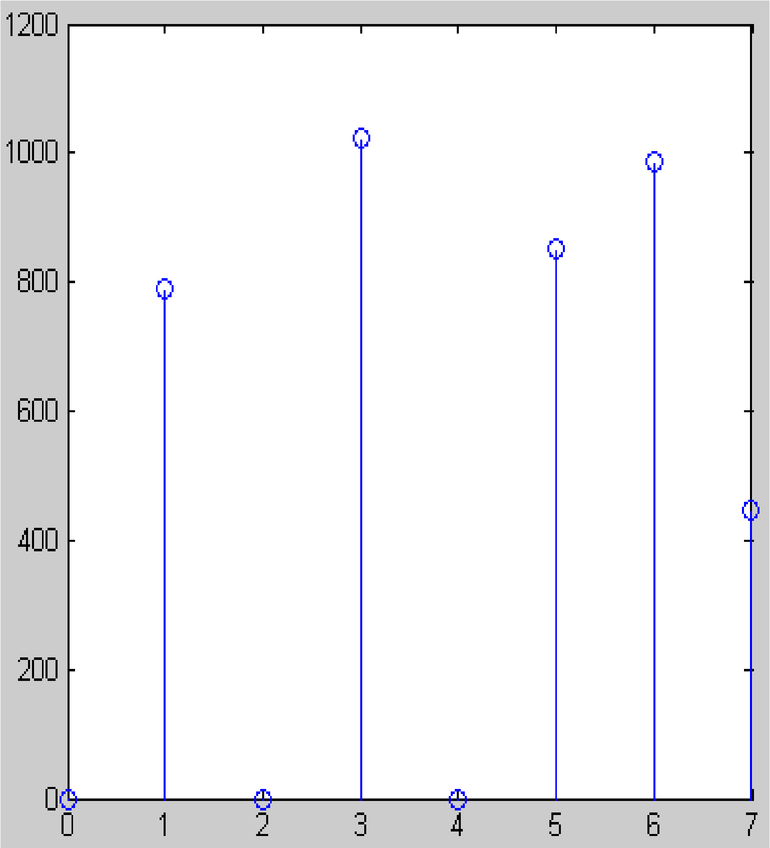
# Histogram Equalization

- Notice that there are only five distinct gray levels --- (1/7, 3/7, 5/7, 6/7, 1) in the output image. We will reliable them as ($s_0$, $s_1$, …, $s_4$ ).

- With this transformation, the output image will have histogram

| $k$ | $s_k$ | $n_k$ | $p(s_k) = n_k/n$ |
|---|---|---|---|
| 0 | 1/7 | 790 | 0.19 |
| 1 | 3/7 | 1023 | 0.25 |
| 2 | 5/7 | 850 | 0.21 |
| 3 | 6/7 | 985 | 0.24 |
| 4 | 1 | 448 | 0.11 |

# Histogram Equalization




Histogram of output image

# pixels

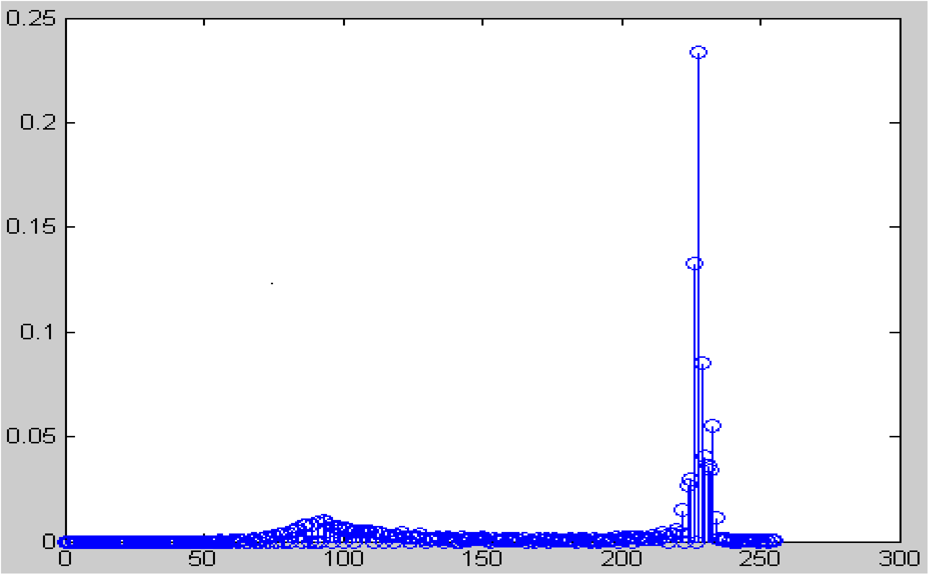
Gray values

• Note that the histogram of output image is only approximately, and not exactly, uniform. This should not be surprising, since there is no result that claims uniformity in the **discrete** case.

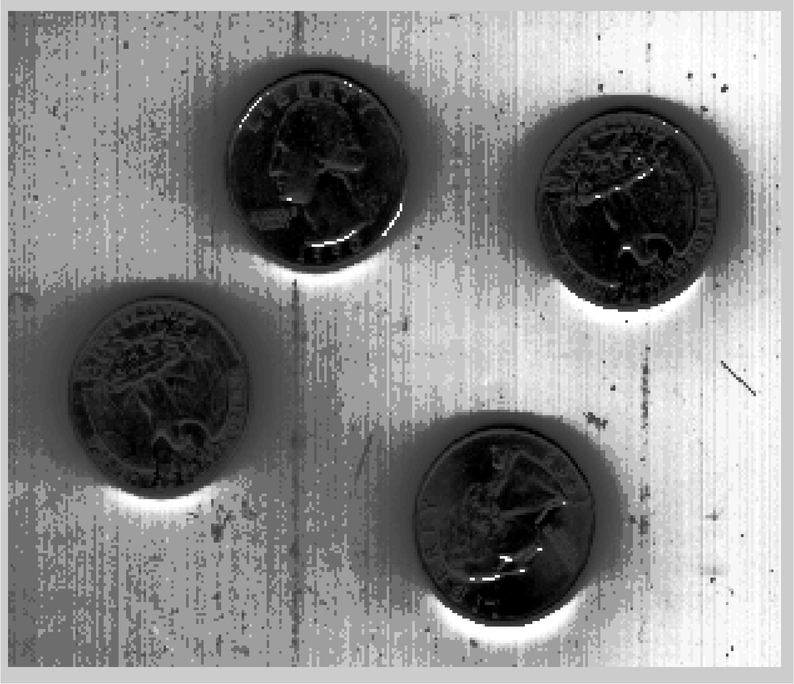
# Histogram Equalization

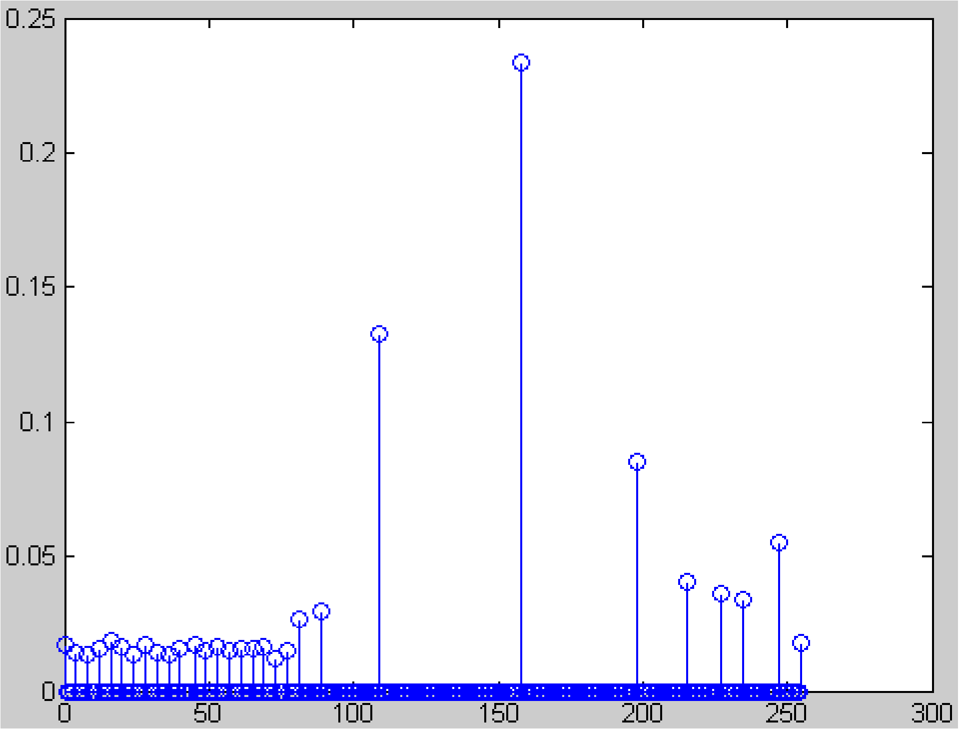
- Comments:

Histogram equalization may not always produce desirable results, particularly if the given histogram is very narrow. It can produce false edges and regions. It can also increase image “graininess” and “patchiness.”



# Histogram Equalization





# Histogram Matching

Histogram equalization yields an image whose pixels are (in theory) uniformly distributed among all gray levels.

Sometimes, this may not be desirable. Instead, we may want a transformation that yields an output image with a pre-specified histogram. This technique is called **histogram specification and/or Histogram Matching**.

# Histogram Matching

- Given Information

  (1) Input image from which we can compute its histogram .

  (2) Desired histogram.

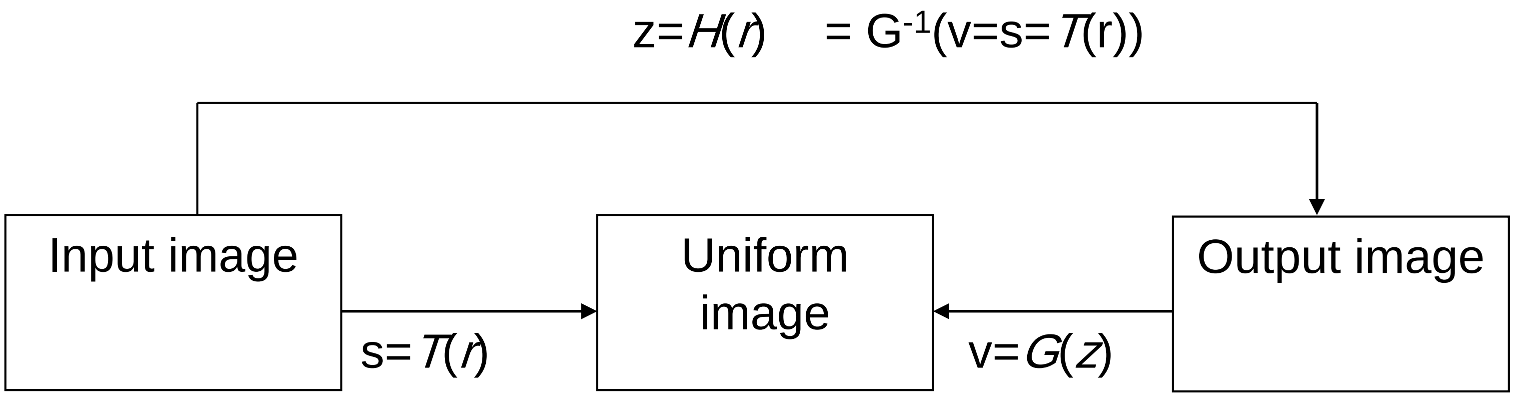
- Goal

Derive a point operation, $H(r)$, that maps the input image into an output image that has the user-specified histogram.

- Again, we will assume, for the moment, continuous-gray values.

# Histogram Matching

Approach of derivation



# Histogram Matching

- Suppose, the input image has probability density in $p(r)$ . We want to find a transformation $z = H(r)$ , such that the probability density of the new image obtained by this transformation is $p_{out}(z)$ , which is not necessarily uniform.

- First apply the transformation

$$s = T(r) = \int_0^r p_{in}(w)dw, \qquad 0 \le r \le 1 \qquad (*)$$

This gives an image with a uniform probability density.

- If the desired output image were available, then the following transformation would generate an image with uniform density:

$$V = G(z) = \int_0^z p_{out}(w)dw, \qquad 0 \le z \le 1 \qquad (**)$$

# Histogram Matching

- From the gray values ν we can obtain the gray values *z* by using the inverse transformation, $z = G^{-1}(v)$

- If instead of using the gray values ν obtained from (**), we use the gray values *s* obtained from (*) above (both are uniformly distributed ! ), then the point transformation

$$Z = H(r) = G^{-1}[\, v = s = T(r)]$$

will generate an image with the specified density out *p(z)* , from an input image with density in *p(r)* !

# Histogram Matching

- For discrete gray levels, we have

$$s_k = T(r_k) = \sum_{j=0}^{k} p_{in}(r_j) \qquad 0 \le k \le L-1$$

$$v_k = G(z_k) = \sum_{j=0}^{k} p_{out}(z_j) = s_k \qquad 0 \le k \le L-1$$

- If the transformation $z_k \rightarrow G(z_k)$ is one-to-one, the inverse transformation $s_k \rightarrow G^{-1}(s_k)$ , can be easily determined, since we are dealing with a small set of discrete gray values.

- In practice, this is not usually the case (i.e., ) $z_k \rightarrow G(z_k)$ is not one-to-one) and we assign gray values to match the given histogram, as closely as possible.

# Histogram Matching

Algorithm for histogram specification:

(1) Equalize input image to get an image with uniform gray values, using the discrete equation:

$$s_k = T(r_k) = \sum_{j=0}^{k} p_{in}(r_j) \qquad 0 \le k \le L-1$$

(2) Based on desired histogram to get an image with uniform gray values, using the discrete equation:

$$v_k = G(z_k) = \sum_{j=0}^{k} p_{out}(z_j) = s_k \qquad 0 \le k \le L-1$$

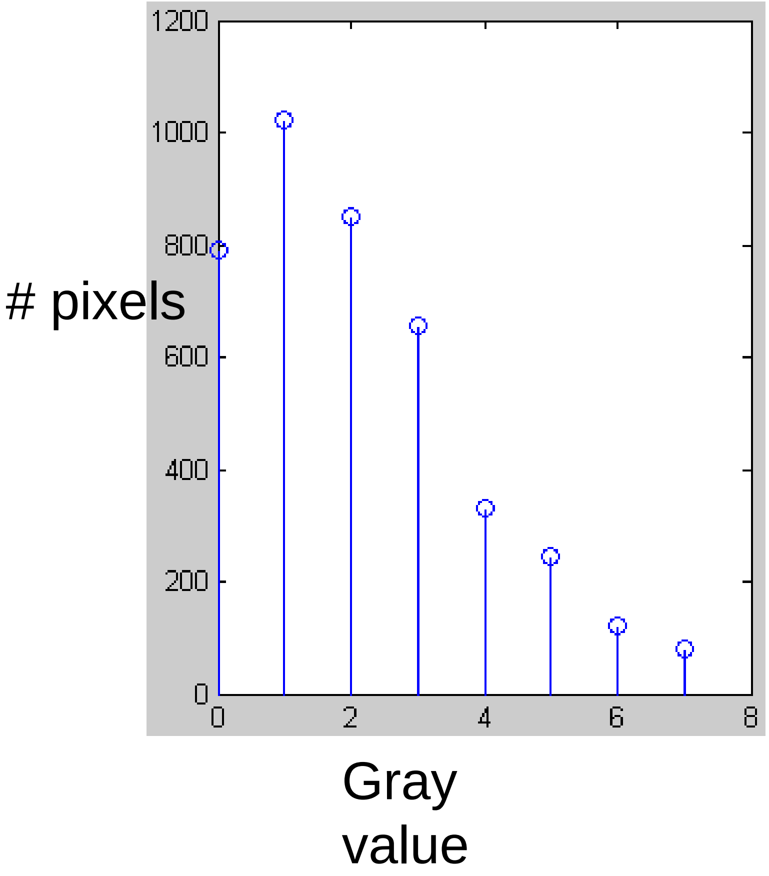
(3) $z = G^{-1}(v = s) \rightarrow z = G^{-1}[T(r)]$

# Histogram Matching

Example:

- Consider an 8-level 64 x 64 previous image.

| $k$ | $r_k$ | $n_k$ | $p(r_k) = n_k/n$ |
|---|---|---|---|
| 0 | 0 | 790 | 0.19 |
| 1 | 1/7 | 1023 | 0.25 |
| 2 | 2/7 | 850 | 0.21 |
| 3 | 3/7 | 656 | 0.16 |
| 4 | 4/7 | 329 | 0.08 |
| 5 | 5/7 | 245 | 0.06 |
| 6 | 6/7 | 122 | 0.03 |
| 7 | 1 | 81 | 0.02 |

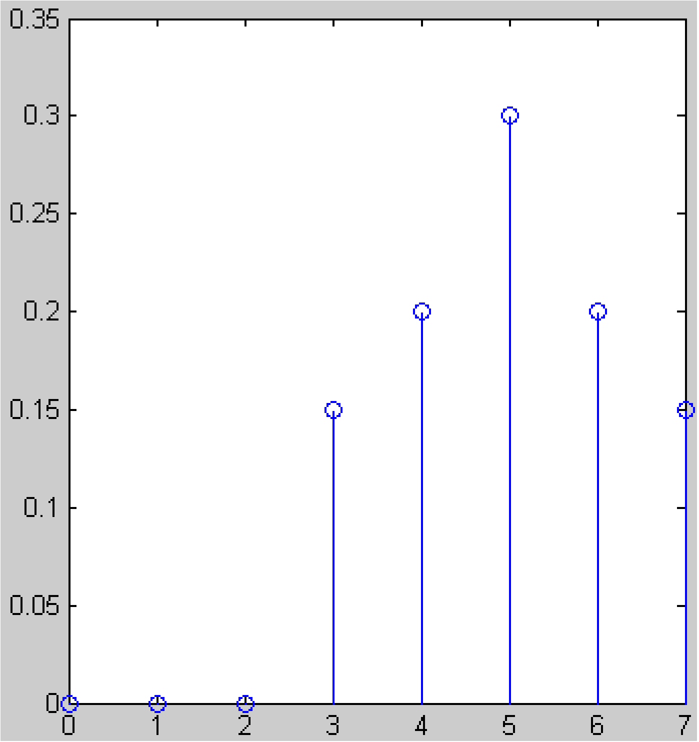


# Histogram Matching

- It is desired to transform this image into a new image, using a transformation $Z=H(r)=G^{-1}[T(r)]$, with histogram as specified below:

| $k$ | $z_k$ | $p_{\text{out}}(z_k)$ |
|---|---|---|
| 0 | 0 | 0.00 |
| 1 | 1/7 | 0.00 |
| 2 | 2/7 | 0.00 |
| 3 | 3/7 | 0.15 |
| 4 | 4/7 | 0.20 |
| 5 | 5/7 | 0.30 |
| 6 | 6/7 | 0.20 |
| 7 | 1 | 0.15 |

# pixels




Gray values

# Histogram Matching

- The transformation $T(r)$ was obtained earlier (reproduced below):

| $r_j \rightarrow s_k$ | $n_k$ | $p(s_k)$ |
|---|---|---|
| $r_0 \rightarrow s_0 = 1/7$ | 790 | 0.19 |
| $r_1 \rightarrow s_1 = 3/7$ | 1023 | 0.25 |
| $r_2 \rightarrow s_2 = 5/7$ | 850 | 0.21 |
| $r_3, r_4 \rightarrow s_3 = 6/7$ | 985 | 0.24 |
| $r_5, r_6, r_7 \rightarrow s_4 = 1$ | 448 | 0.11 |

- Now we compute the transformation G as before.

# Histogram Matching

$$v_0 = G(z_0) = \sum_{j=0}^{0} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) = 0.00 \rightarrow 0$$

$$v_1 = G(z_1) = \sum_{j=0}^{1} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) = 0.00 \rightarrow 0$$

$$v_2 = G(z_2) = \sum_{j=0}^{2} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + p_{\text{out}}(z_2) = 0.00 \rightarrow 0$$

$$v_3 = G(z_3) = \sum_{j=0}^{3} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_3) = 0.15 \rightarrow {}^{1}\!/_{7}$$

$$v_4 = G(z_4) = \sum_{j=0}^{4} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_4) = 0.35 \rightarrow {}^{2}\!/_{7}$$

$$v_5 = G(z_5) = \sum_{j=0}^{5} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_5) = 0.65 \rightarrow {}^{5}\!/_{7}$$

$$v_6 = G(z_6) = \sum_{j=0}^{6} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_6) = 0.85 \rightarrow {}^{6}\!/_{7}$$

$$v_7 = G(z_7) = \sum_{j=0}^{7} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_7) = 1.00 \rightarrow 1$$

# Histogram Matching

- Computer z=$G^{-1}$ (s), Notice that *G* is not invertible.

$G^{-1}(0) = ?$

$G^{-1}(1/7) = 3/7$

$G^{-1}(2/7) = 4/7$

$G^{-1}(3/7) = ?$

$G^{-1}(4/7) = ?$

$G^{-1}(5/7) = 5/7$

$G^{-1}(6/7) = 6/7$

$G^{-1}(1) = 1$

$$v_0 = G(z_0) = \sum_{j=0}^{0} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) = 0.00 \to 0$$

$$v_1 = G(z_1) = \sum_{j=0}^{1} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) = 0.00 \to 0$$

$$v_2 = G(z_2) = \sum_{j=0}^{2} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + p_{\text{out}}(z_2) = 0.00 \to 0$$

$$v_3 = G(z_3) = \sum_{j=0}^{3} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_3) = 0.15 \to {}^{1}\!/_{7}$$

$$v_4 = G(z_4) = \sum_{j=0}^{4} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_4) = 0.35 \to {}^{2}\!/_{7}$$

$$v_5 = G(z_5) = \sum_{j=0}^{5} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_5) = 0.65 \to {}^{5}\!/_{7}$$

$$v_6 = G(z_6) = \sum_{j=0}^{6} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_6) = 0.85 \to {}^{6}\!/_{7}$$

$$v_7 = G(z_7) = \sum_{j=0}^{7} p_{\text{out}}(z_j) = p_{\text{out}}(z_0) + p_{\text{out}}(z_1) + \cdots + p_{\text{out}}(z_7) = 1.00 \to 1$$

# Histogram Matching

- Combining the two transformation $T$ and $G^{-1}$ , compute $z=H(r)=G^{-1}[v=s=T(r)]$

| $r \rightarrow T(r)=s$ | $s \rightarrow G^{-1}(s)=z$ | $r \rightarrow z=H(r)=G^{-1}[T(r)]$ |
|---|---|---|
| $r_0=0 \rightarrow 1/7$ | $s_0=0 \rightarrow ?$ | $r_0=0 \rightarrow G^{-1}[1/7]=3/7$ |
| $r_1=1/7 \rightarrow 3/7$ | $s_1=1/7 \rightarrow 3/7$ | $r_1=1/7 \rightarrow G^{-1}[3/7]=?$ 4/7 |
| $r_2=2/7 \rightarrow 5/7$ | $s_2=2/7 \rightarrow 4/7$ | $r_2=2/7 \rightarrow G^{-1}[5/7]=5/7$ |
| $r_3=3/7 \rightarrow 6/7$ | $s_3=3/7 \rightarrow ?$ | $r_3=3/7 \rightarrow G^{-1}[6/7]=6/7$ |
| $r_4=4/7 \rightarrow 6/7$ | $s_4=4/7 \rightarrow ?$ | $r_4=4/7 \rightarrow G^{-1}[6/7]=6/7$ |
| $r_5=5/7 \rightarrow 1$ | $s_5=5/7 \rightarrow 5/7$ | $r_5=5/7 \rightarrow G^{-1}[1]=1$ |
| $r_6=6/7 \rightarrow 1$ | $s_6=6/7 \rightarrow 6/7$ | $r_6=6/7 \rightarrow G^{-1}[1]=1$ |
| $r_7=1 \rightarrow 1$ | $s_7=1 \rightarrow 1$ | $r_7=1 \rightarrow G^{-1}[1]=1$ |

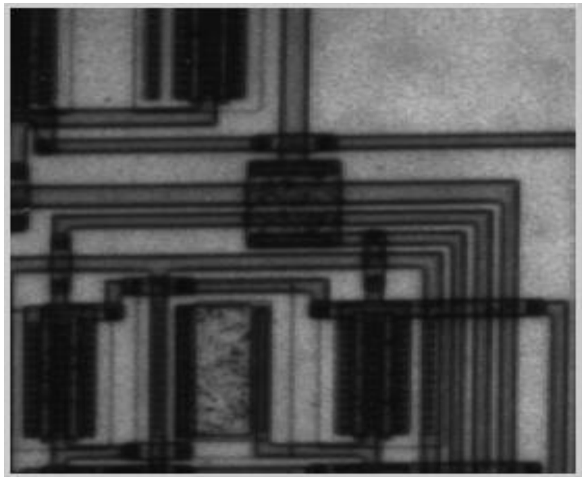
# Histogram Matching

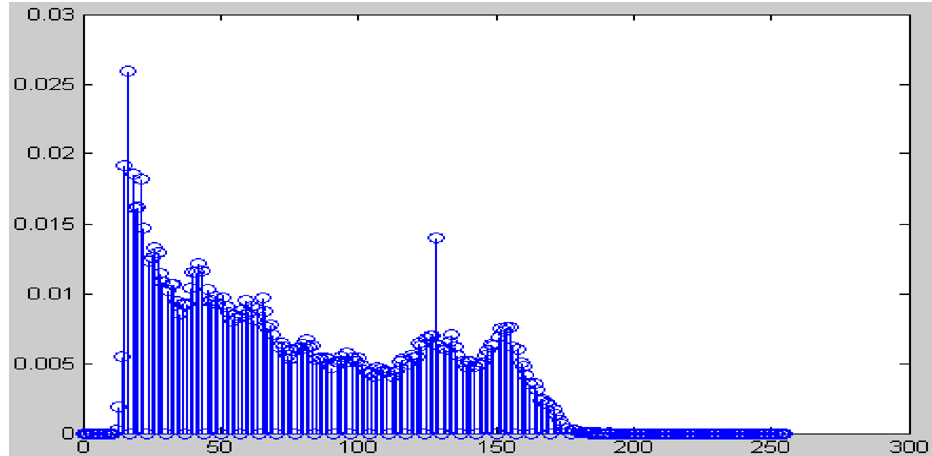
- Applying the transformation *H* to the original image yields an image with histogram as below:

| $k$ | $z_k$ | $n_k$ | $n_k/n$ (actual hist.) | $p_{out}(z_k)$ (specified hist.) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0.00 | 0.00 |
| 1 | 1/7 | 0 | 0.00 | 0.00 |
| 2 | 2/7 | 0 | 0.00 | 0.00 |
| 3 | 3/7 | 790 | 0.19 | 0.15 |
| 4 | 4/7 | 1023 | 0.25 | 0.20 |
| 5 | 5/7 | 850 | 0.21 | 0.30 |
| 6 | 6/7 | 985 | 0.24 | 0.20 |
| 7 | 1 | 448 | 0.11 | 0.15 |

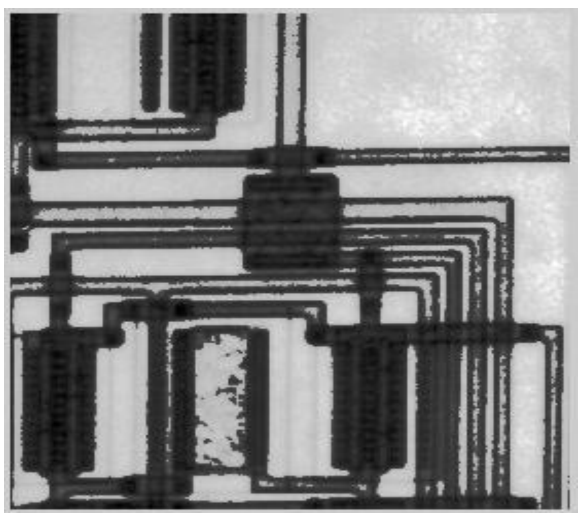
- Again, the actual histogram of the output image does not exactly but only approximately matches with the specified histogram. This is because we are dealing with discrete histograms.

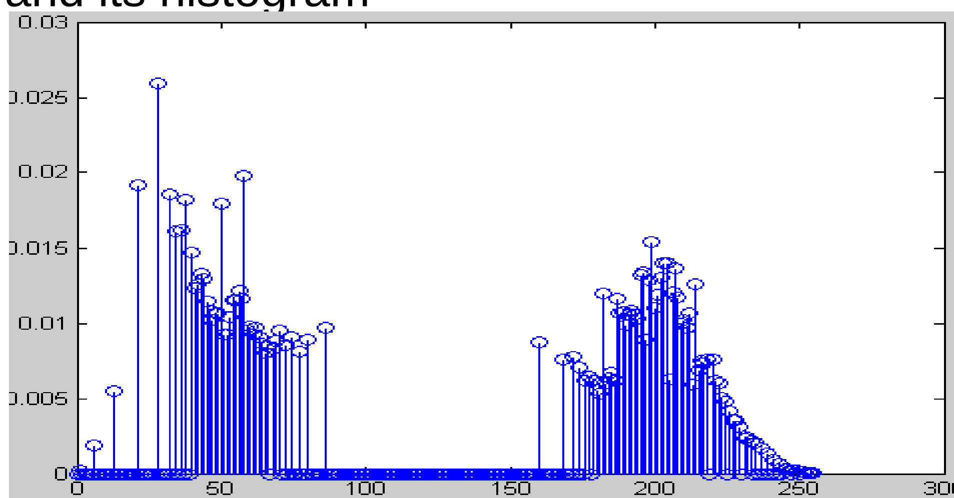
# Histogram Matching





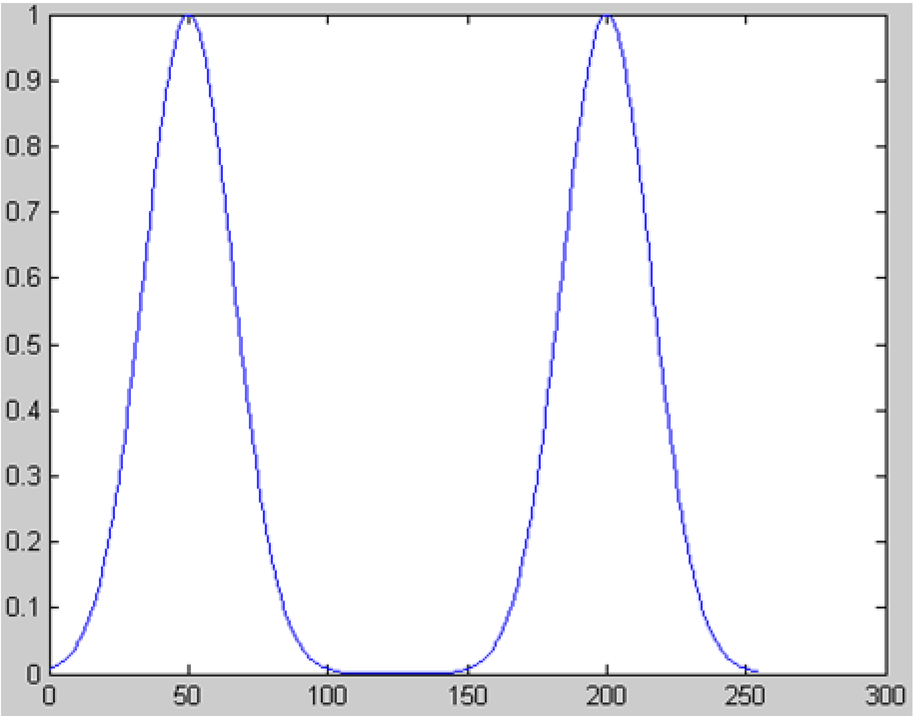

Original image and its histogram






Histogram specified image and its histogram

# Summary




We have looked at:

– Histogram equalisation

Next time we will start to look at neighbourhood operations

DIP

LECTURE No. ( 5 )

# *Image Enhancement*

# *(Spatial Filtering )*

10/14/2019

Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

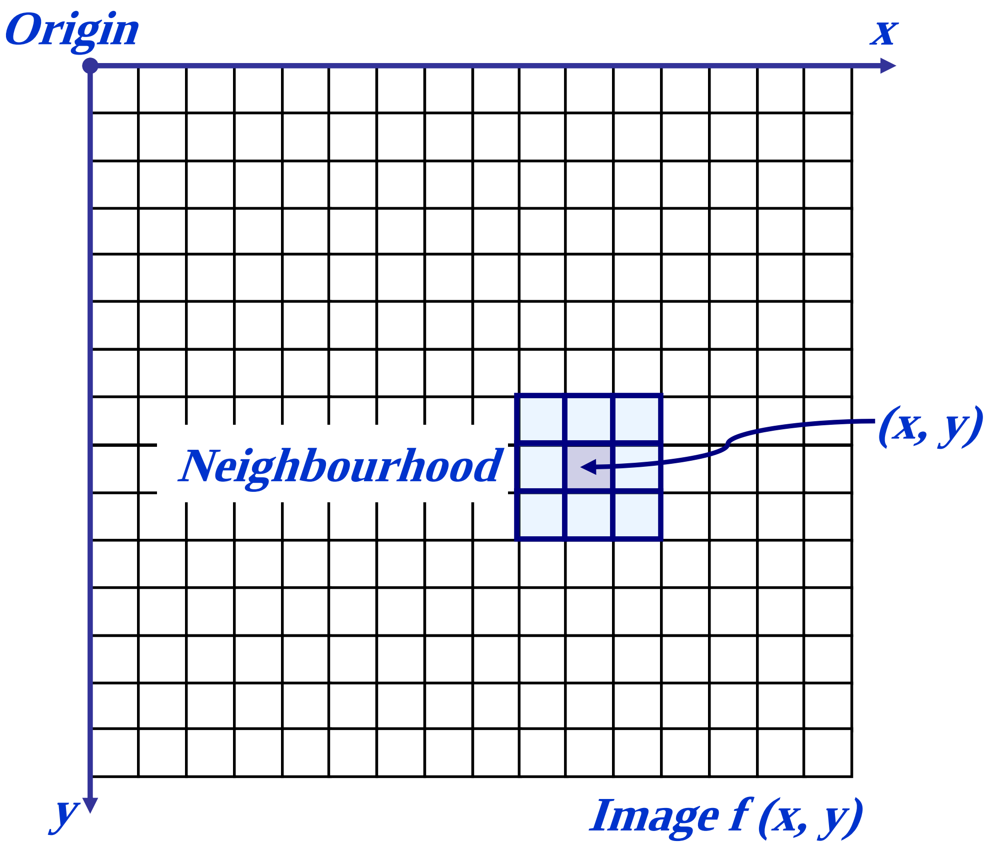
# Contents

In this lecture we will look at spatial filtering techniques:

- Neighbourhood operations
- What is spatial filtering?
- Smoothing operations
- What happens at the edges?
- Correlation and convolution
- Sharpening filters
  - 1st derivative filters
  - 2nd derivative filters
- Combining filtering techniques

# 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*** *x*

| | | | | | |
|---|---|---|---|---|---|
| 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 |

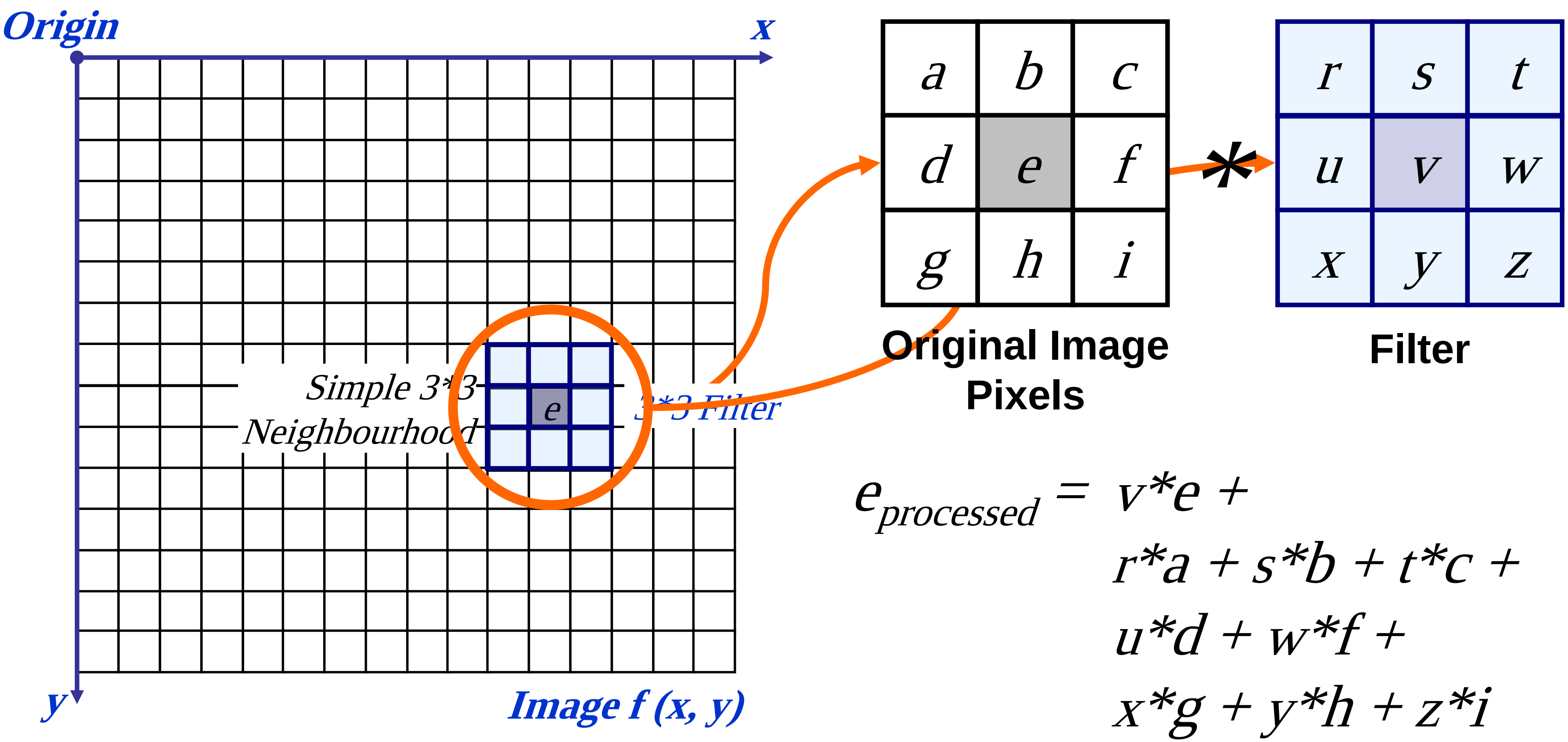
*y*

***Enhanced Image*** *x*

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

*y*

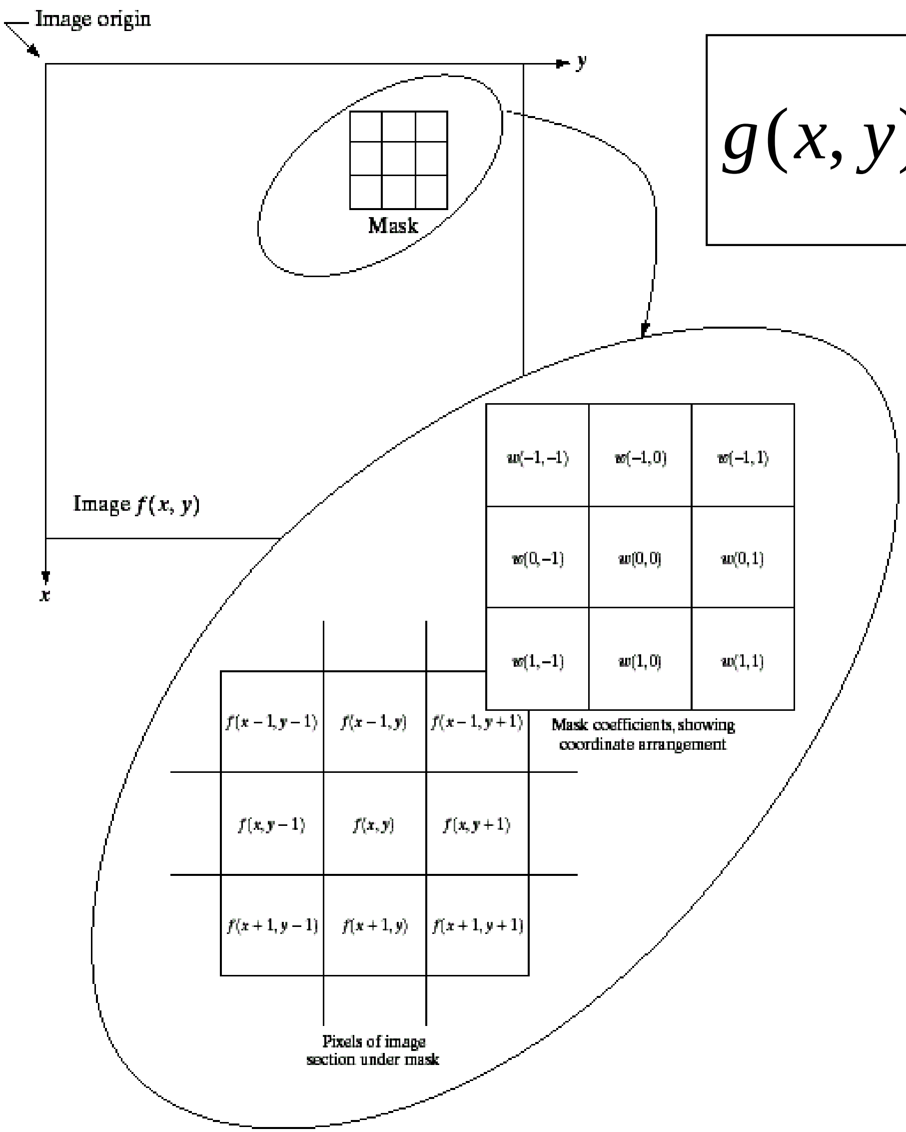
# The Spatial Filtering Process




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

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

# Spatial Filtering: Equation Form




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

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

# Smoothing Spatial Filters

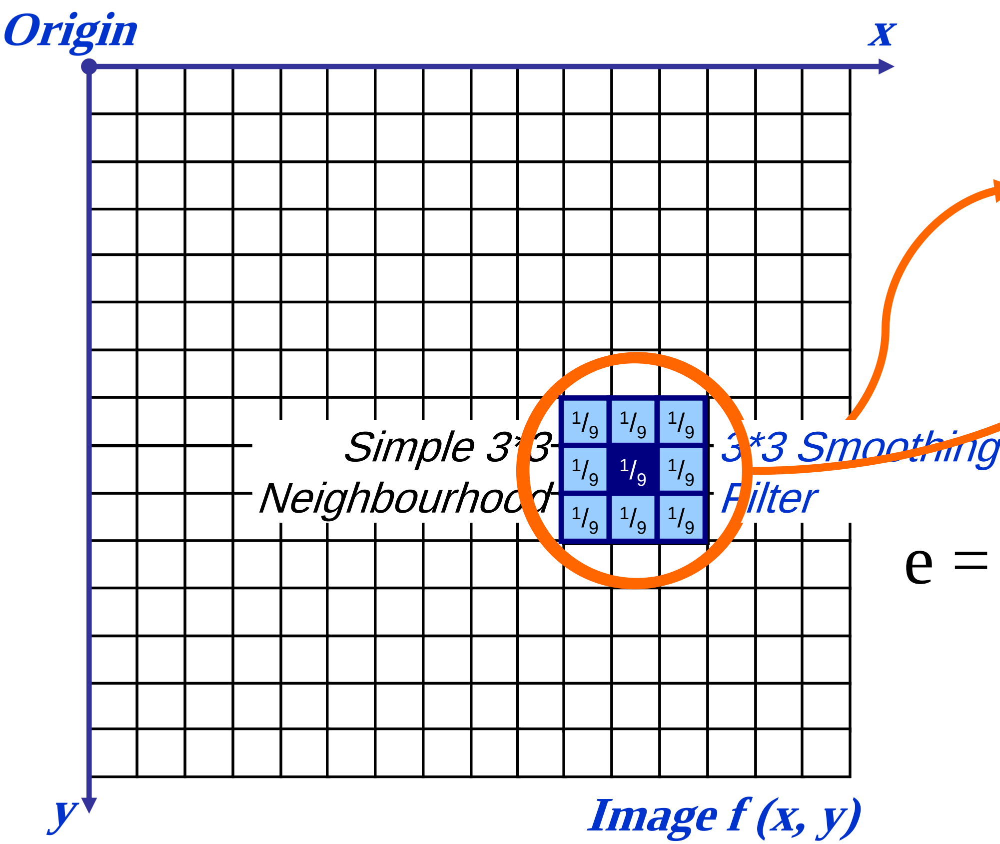
One of the simplest spatial filtering operations we can perform is a smoothing operation

- 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**

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

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

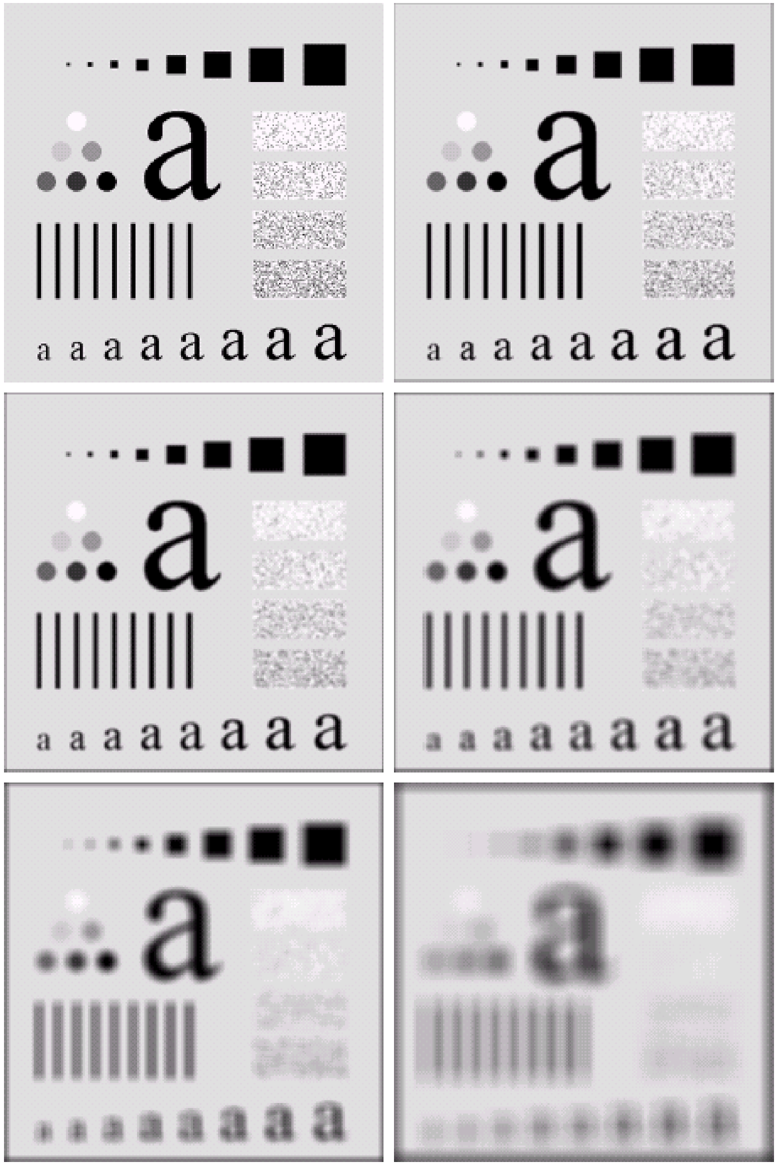
# Image Smoothing Example

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

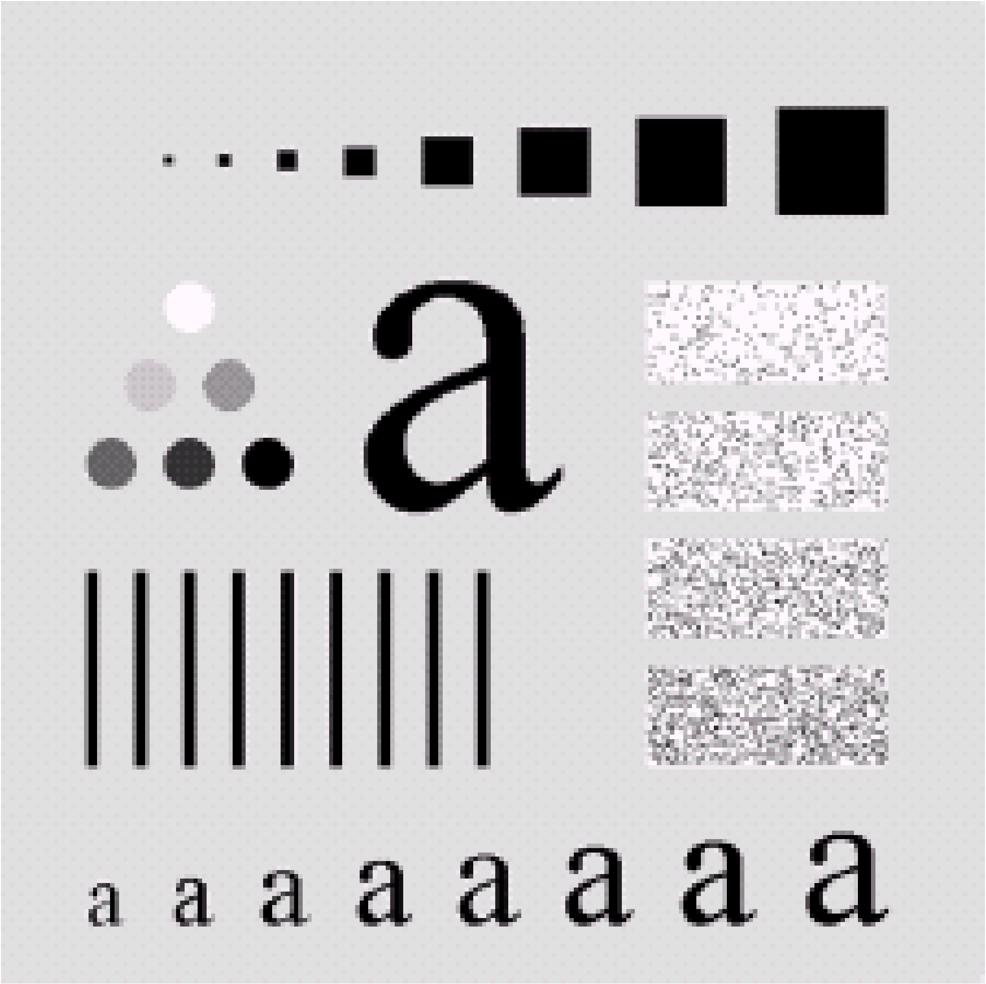
The subsequent images show the image after filtering with an averaging filter of increasing sizes

- 3, 5, 9, 15 and 35

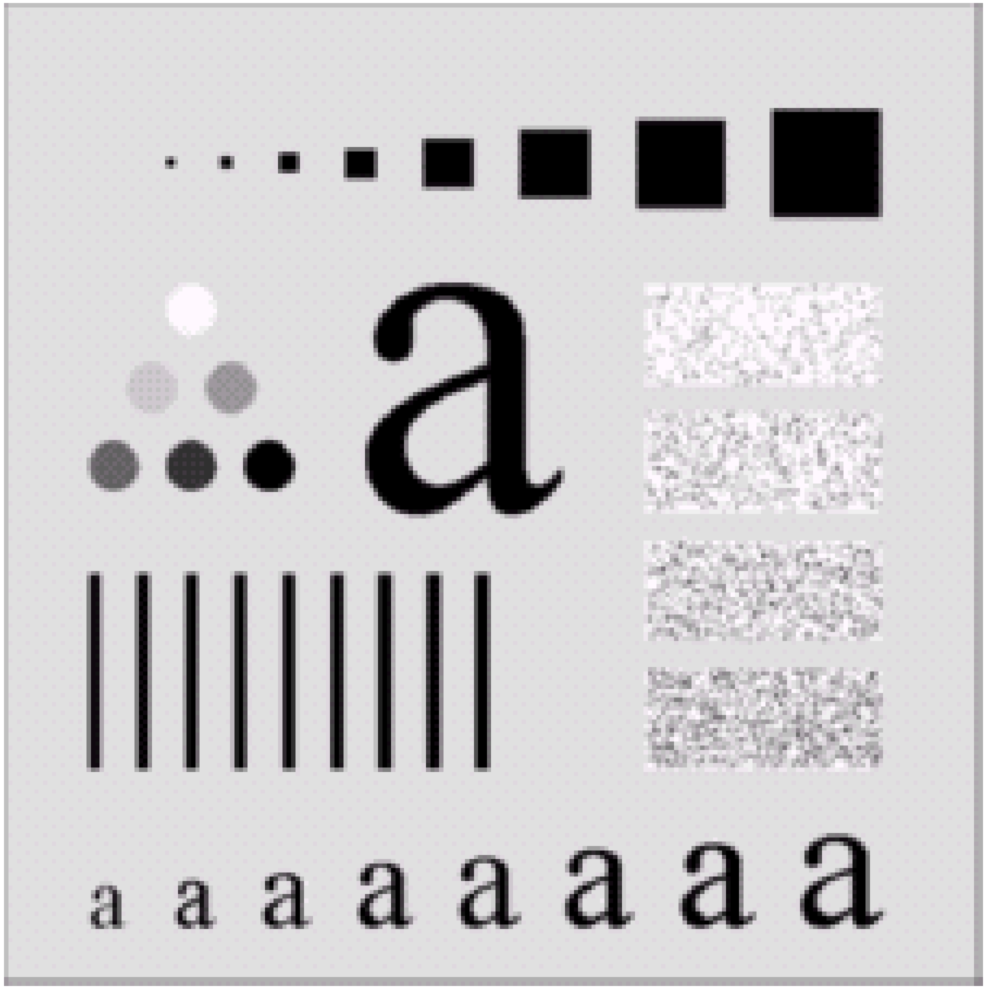
Notice how detail begins to disappear



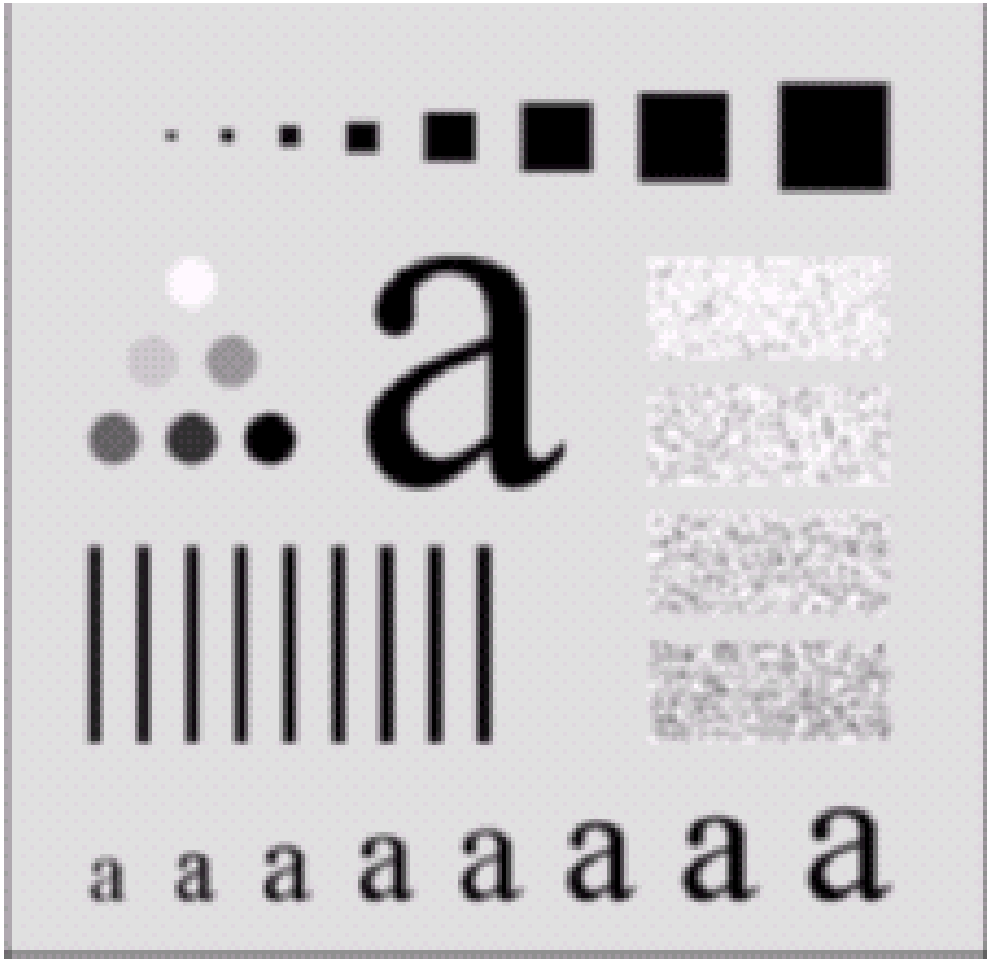
# Image Smoothing Example



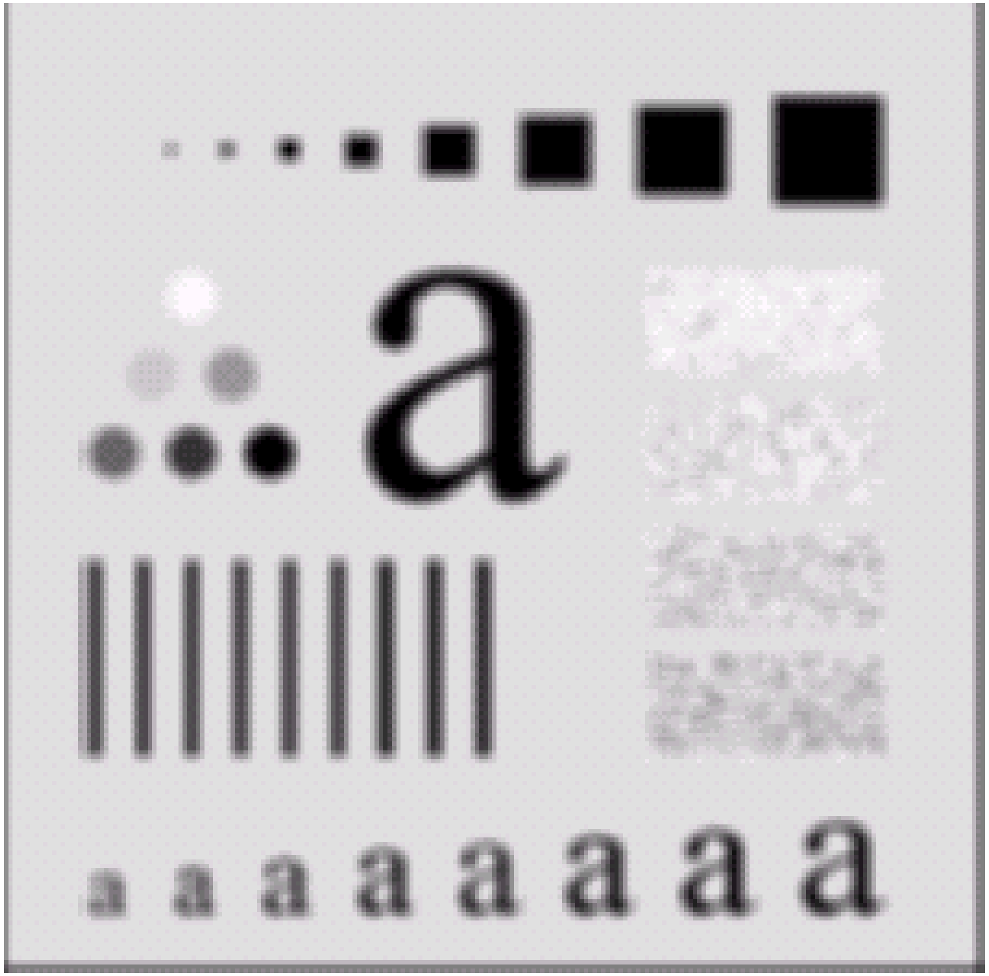
# Image Smoothing Example 3



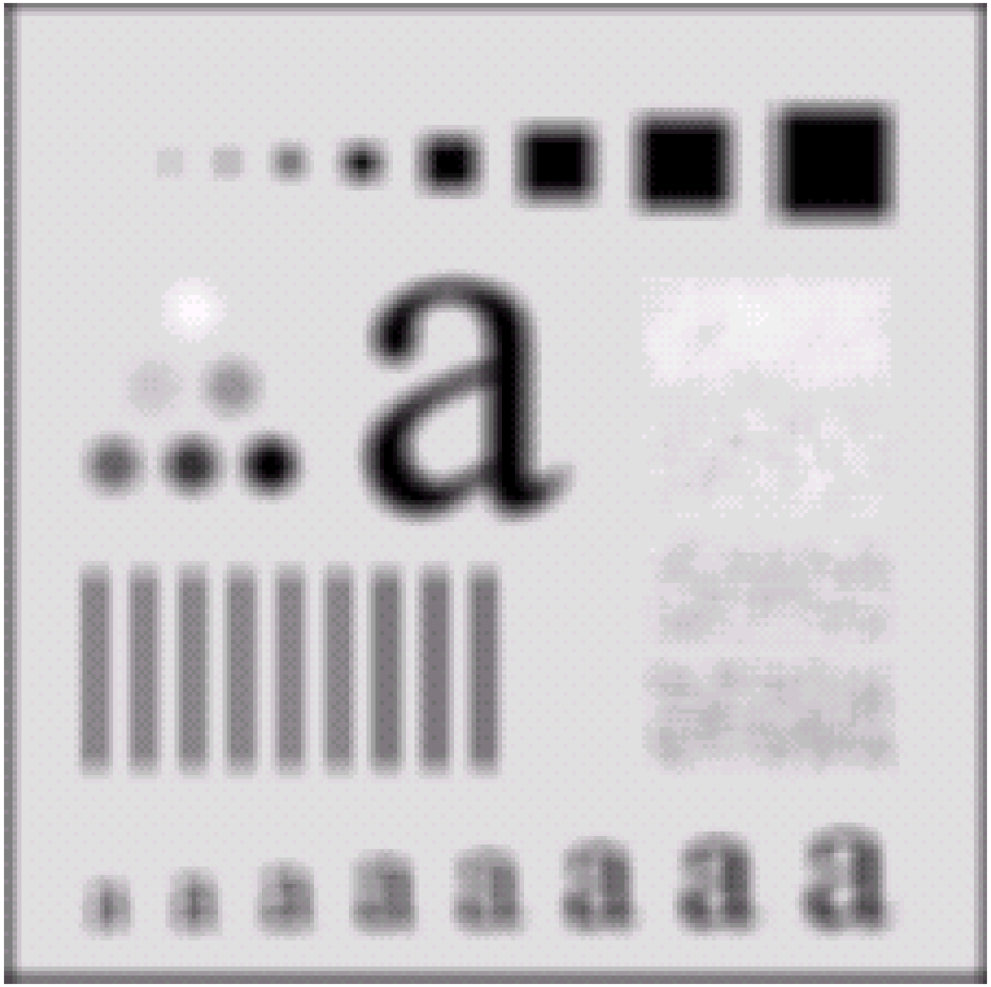
# Image Smoothing Example 5



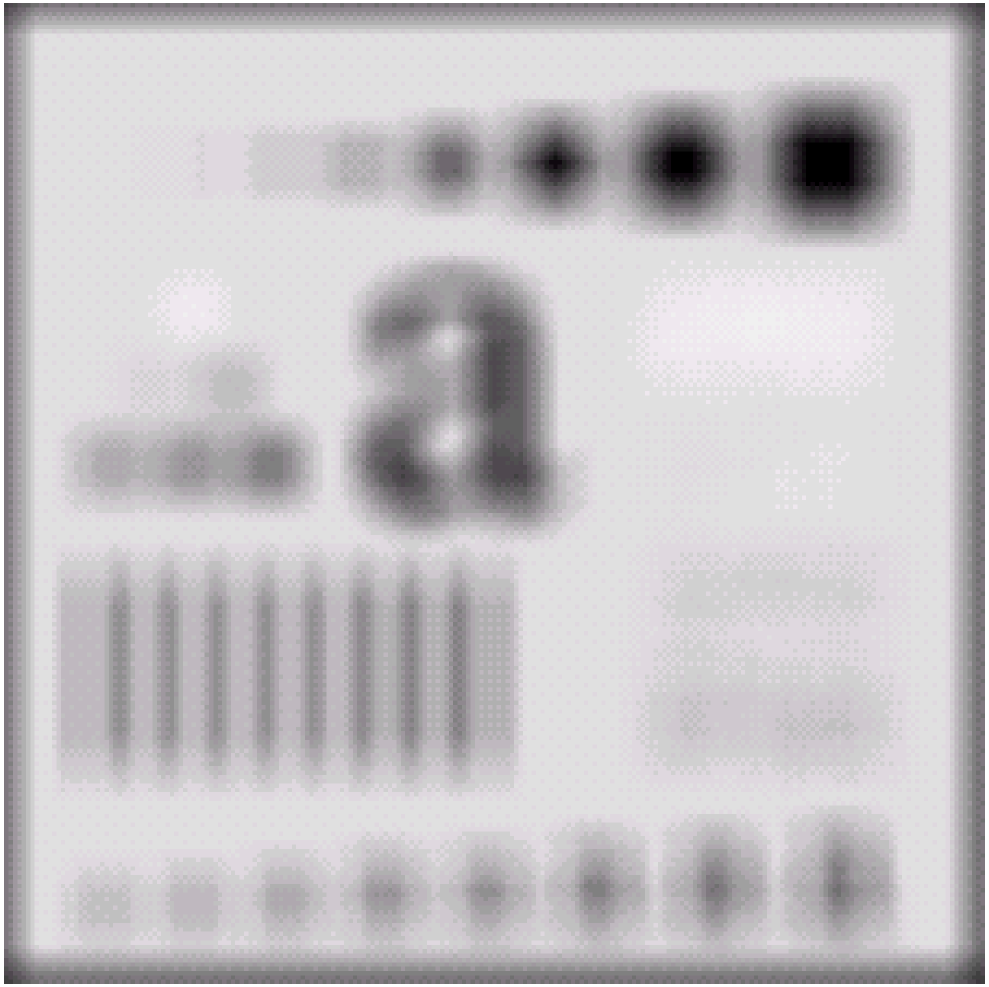
# Image Smoothing Example 9



# Image Smoothing Example 15



# Image Smoothing Example 35



# Weighted Smoothing Filters

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

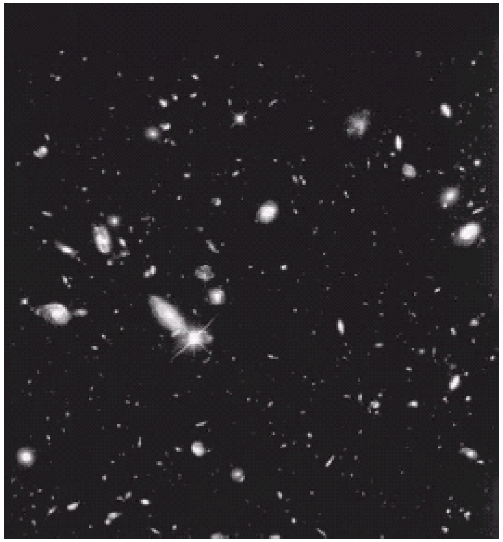
- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

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

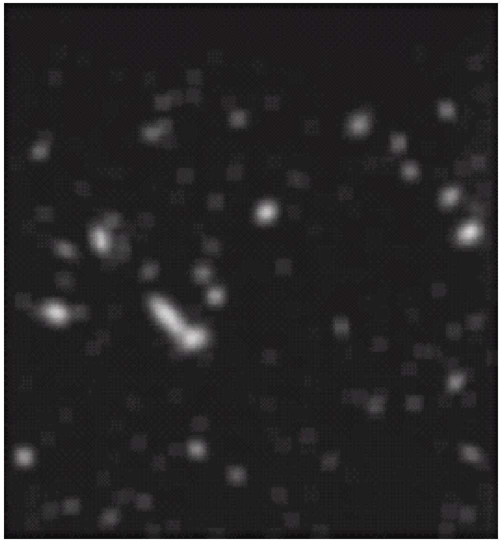
Weighted averaging filter

# Another Smoothing Example

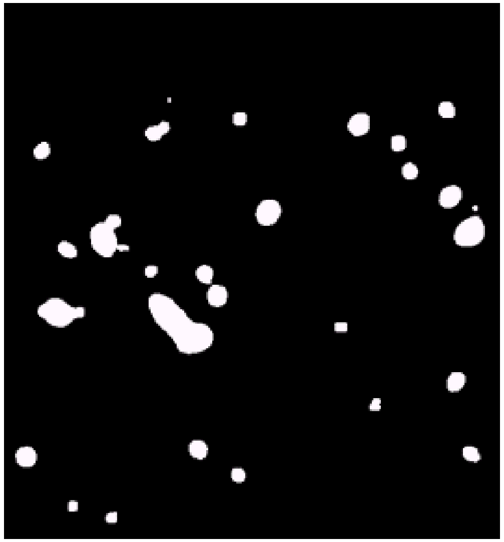
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



**Original Image**

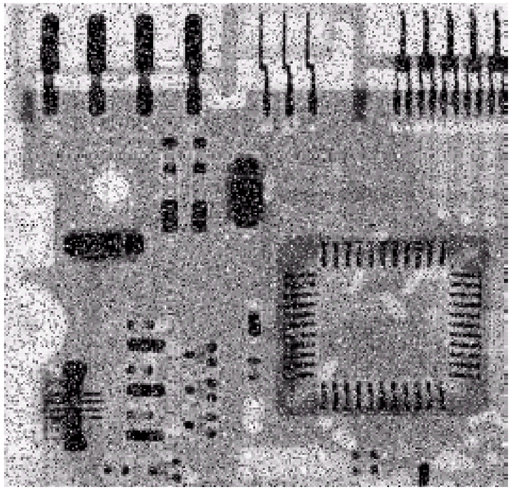


**Smoothed Image**



**Thresholded Image**

# Averaging Filter Vs. Median Filter Example



**Original Image With Noise**

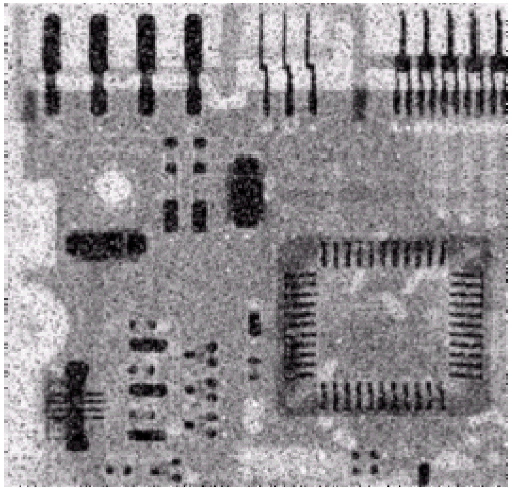


**Image After Averaging Filter**

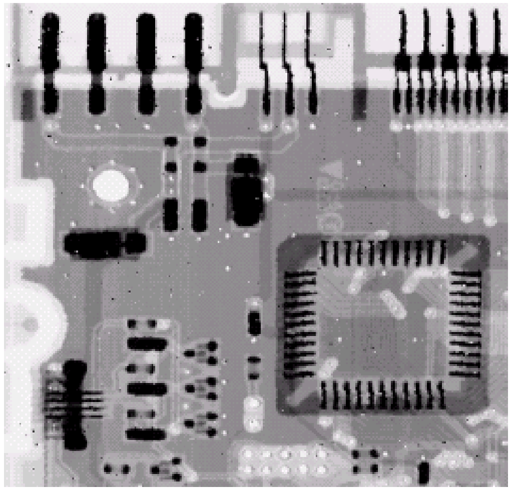
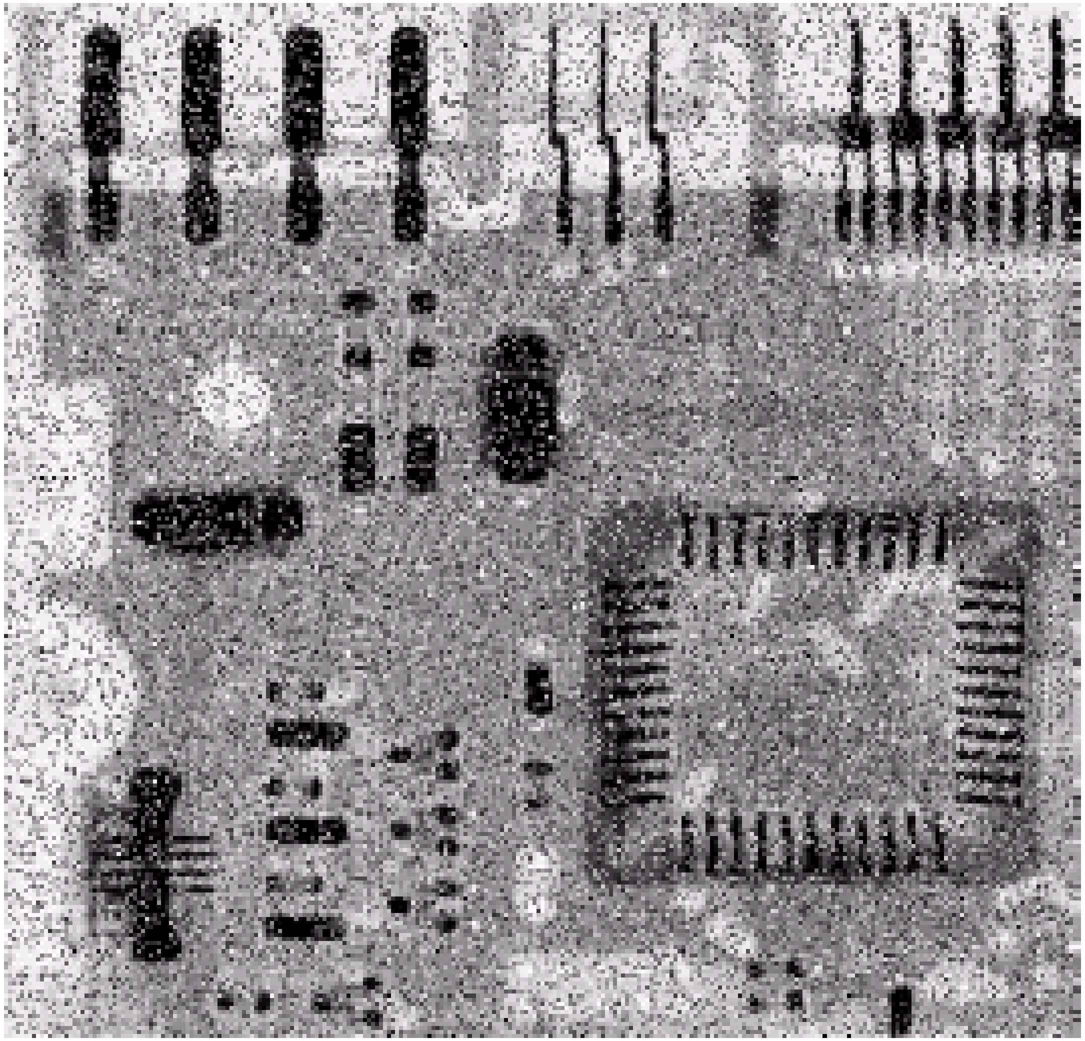


**Image After Median Filter**

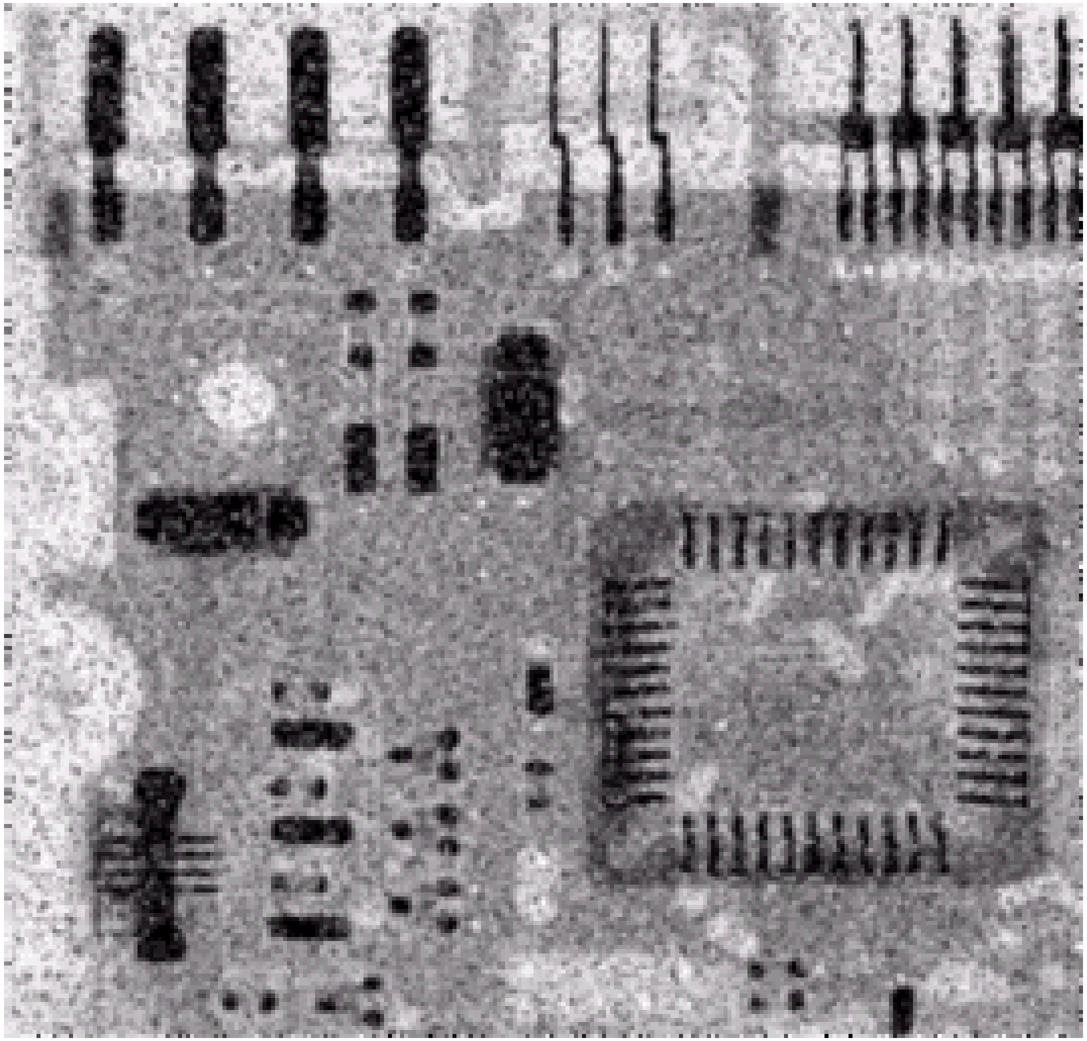
Filtering is often used to remove noise from images

Sometimes a median filter works better than an averaging filter

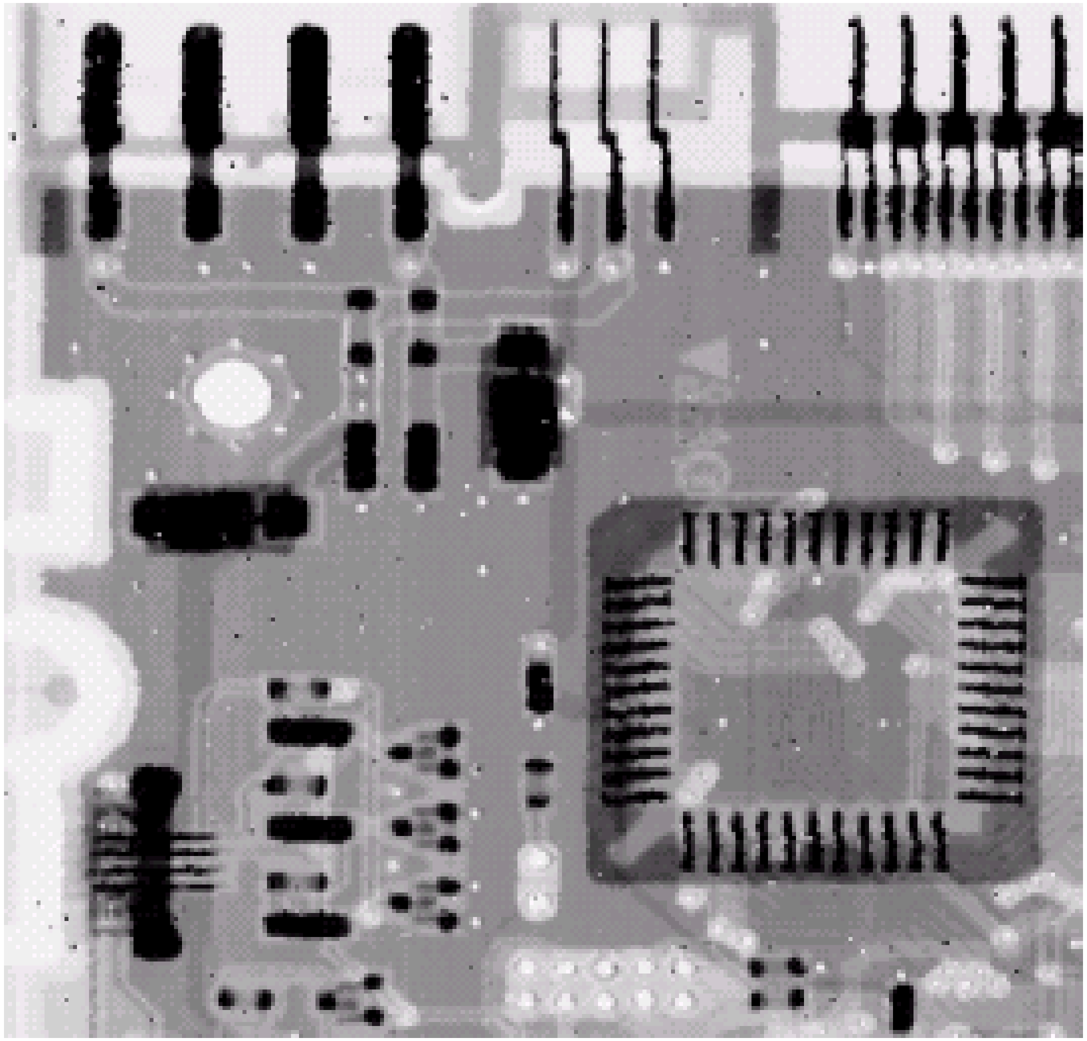
# Example Original Image With Noise



# Example Averaging Filter



# Example Median Filter



# Simple Neighbourhood Operations Example

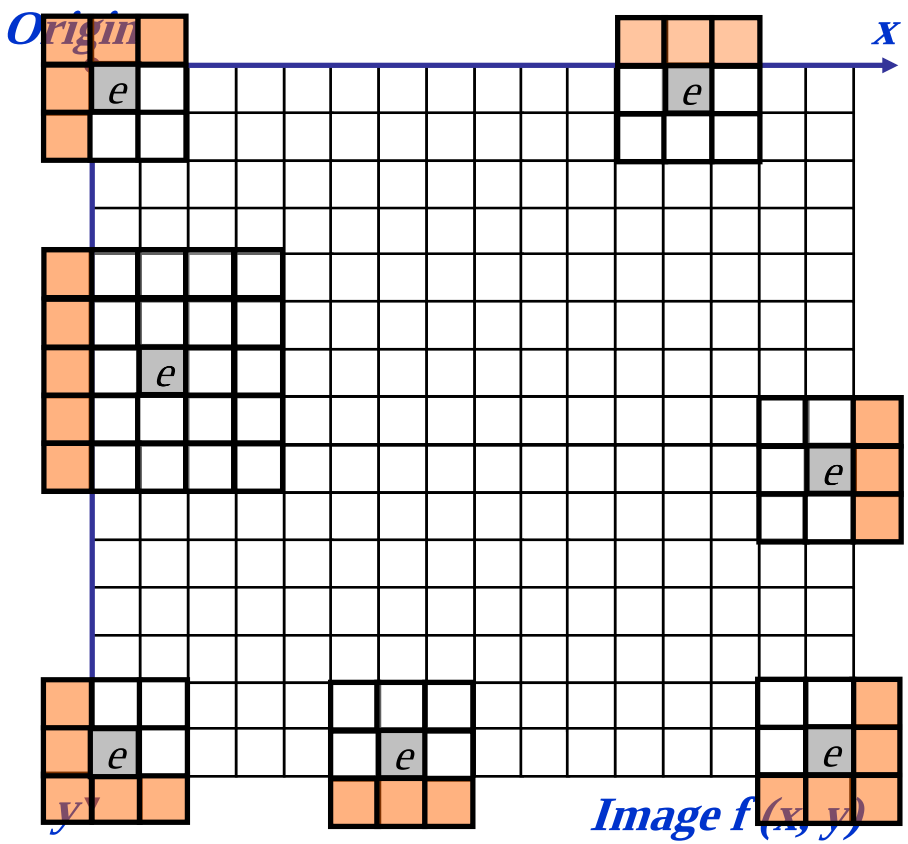
$x$

| | | | | | |
|---|---|---|---|---|---|
| 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 |

$y$

# 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

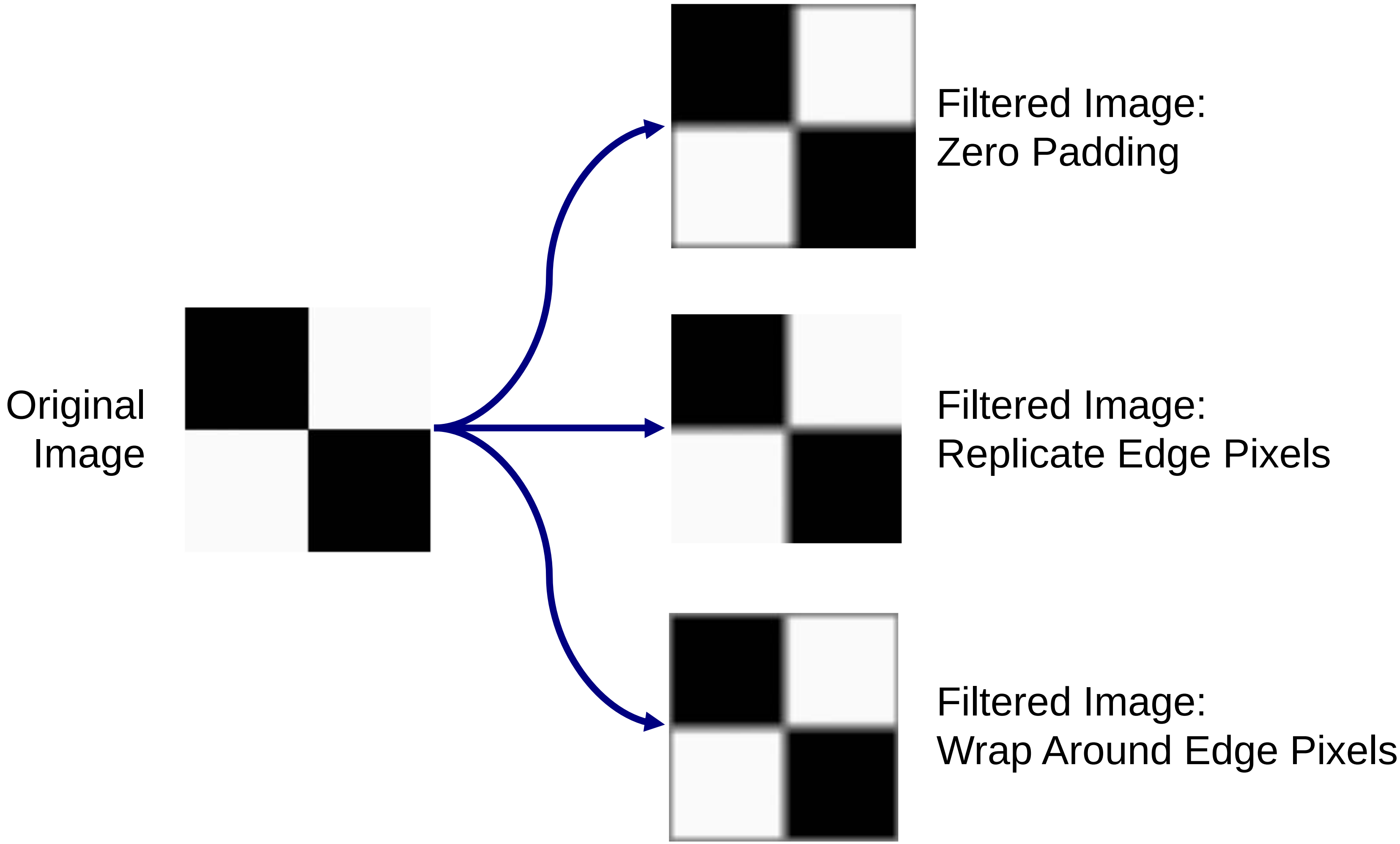
# Simple Neighbourhood Operations Example

x

| | | | | | |
|---|---|---|---|---|---|
| 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 |

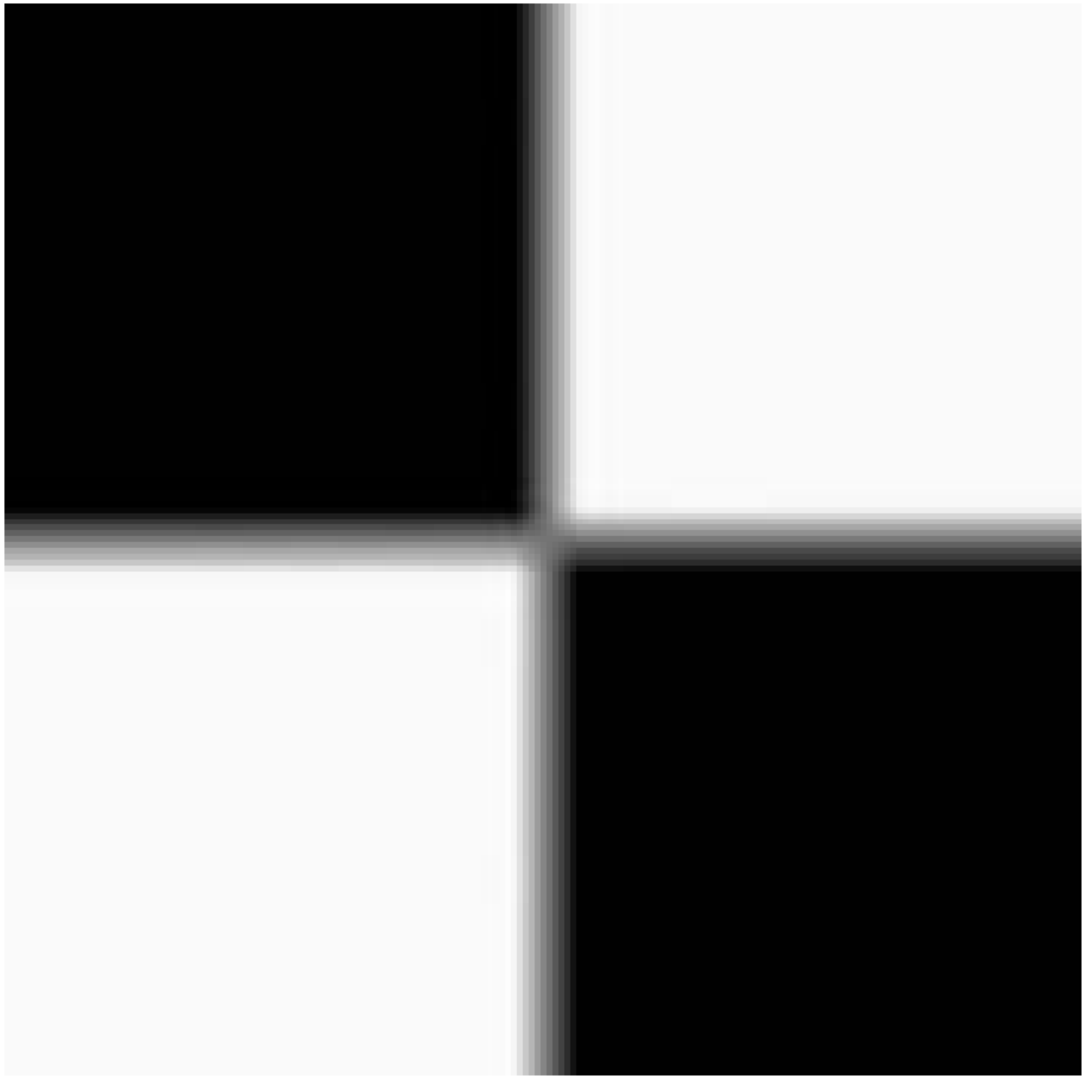
y

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



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

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



# 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

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

**Original Image Pixels**

$*$

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

**Filter**

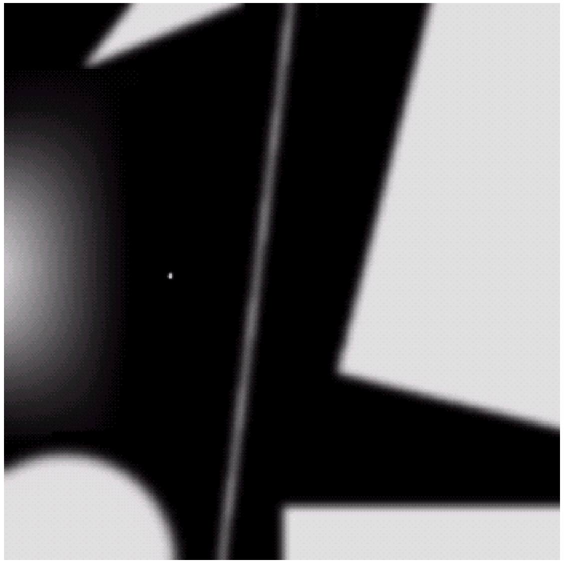
$$
\begin{aligned}
e_{processed} = \; & v*e + \\
& z*a + y*b + x*c + \\
& w*d + u*f + \\
& t*f + s*g + r*h
\end{aligned}
$$

For symmetric filters it makes no difference

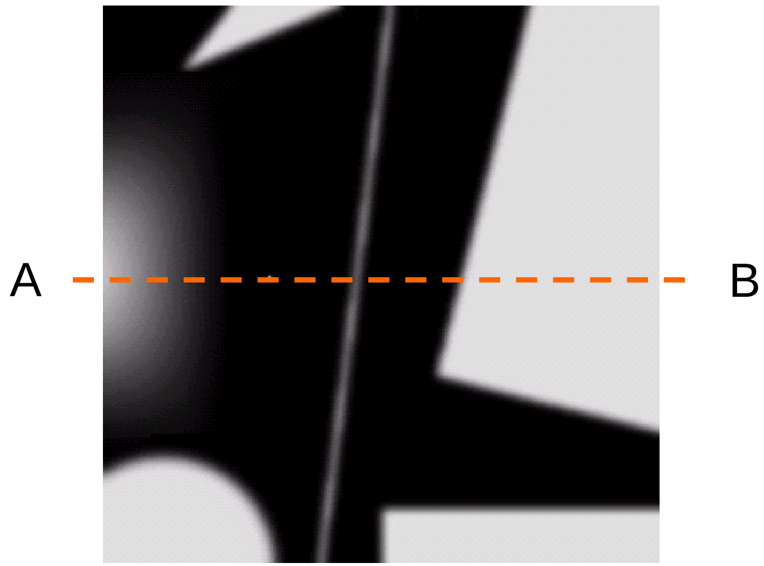
# Spatial Differentiation

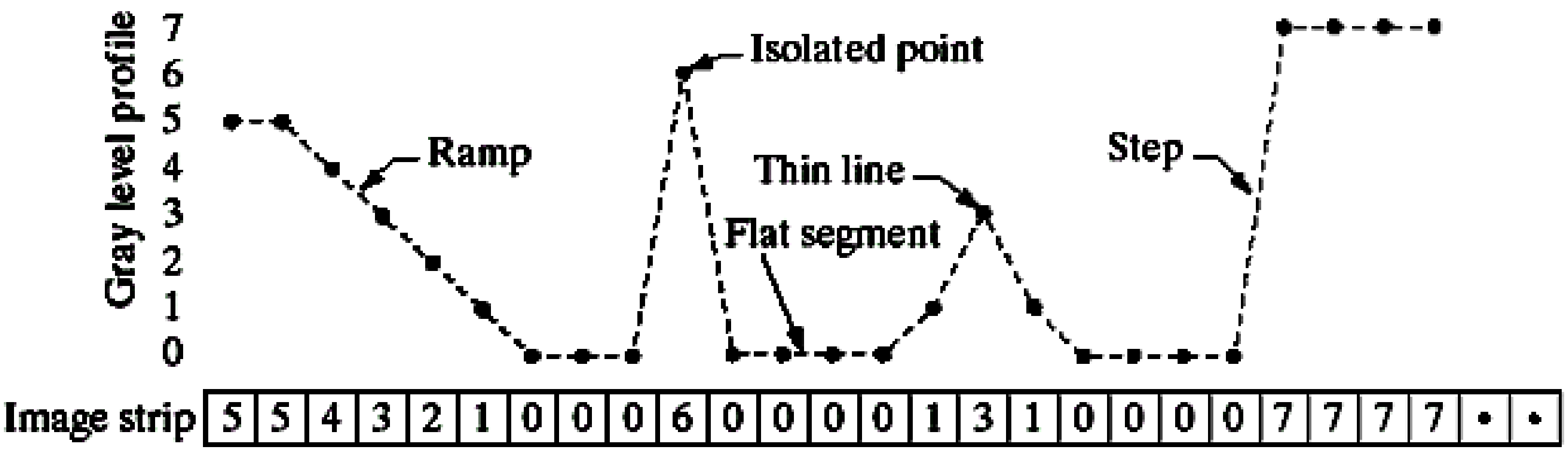
Differentiation measures the *rate of change* of a function

Let's consider a simple 1 dimensional example



# Spatial Differentiation



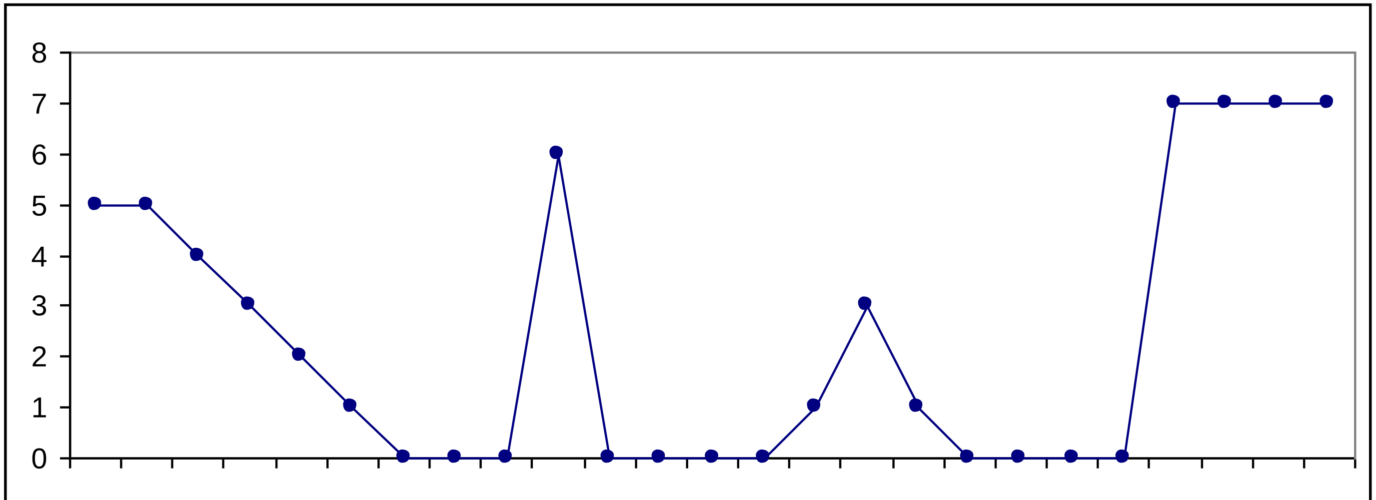
# 1st Derivative

The formula for the 1st derivative of a function is as follows:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

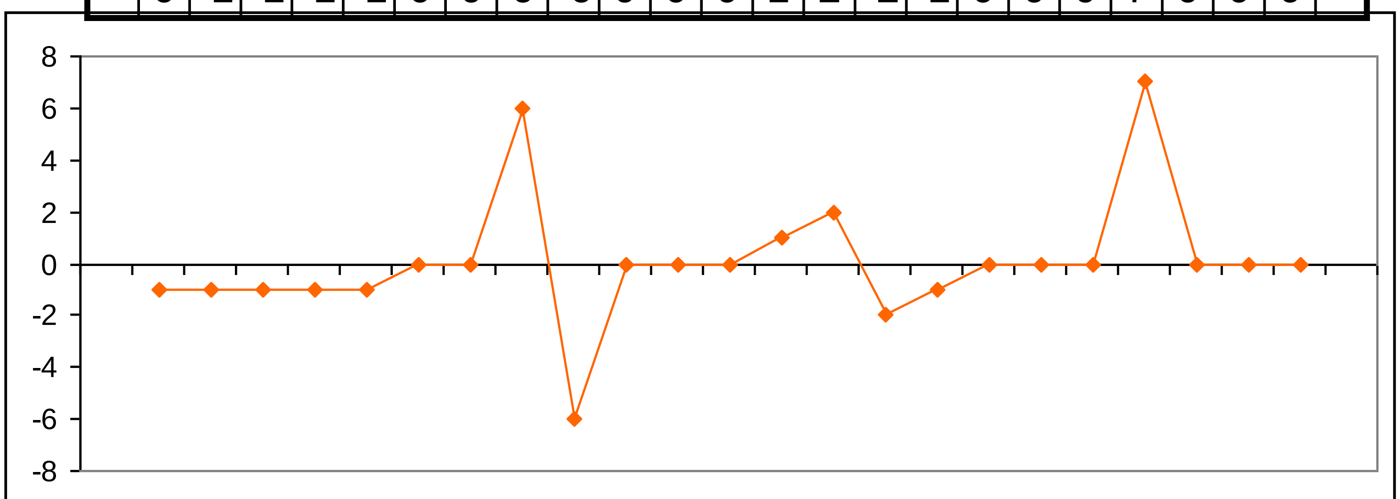
It's just the difference between subsequent values and measures the rate of change of the function

# $1^{st}$ Derivative (cont…)




| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | -1 | -1 | -1 | -1 | 0 | 0 | 6 | -6 | 0 | 0 | 0 | 1 | 2 | -2 | -1 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | |



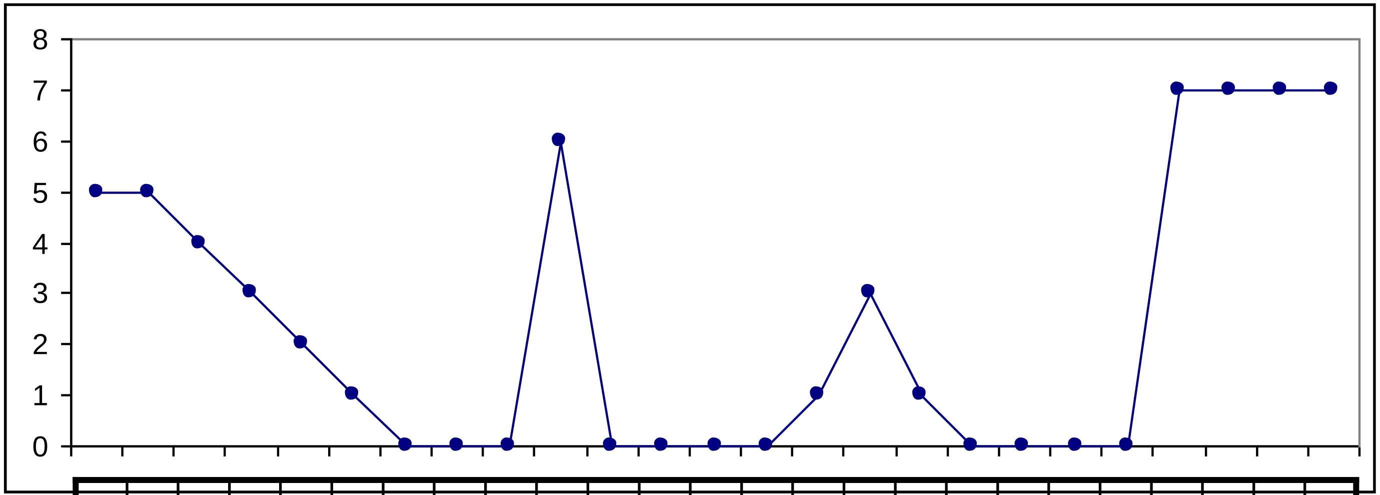
# 2$^{nd}$ Derivative

The formula for the 2$^{nd}$ derivative of a function is as follows:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

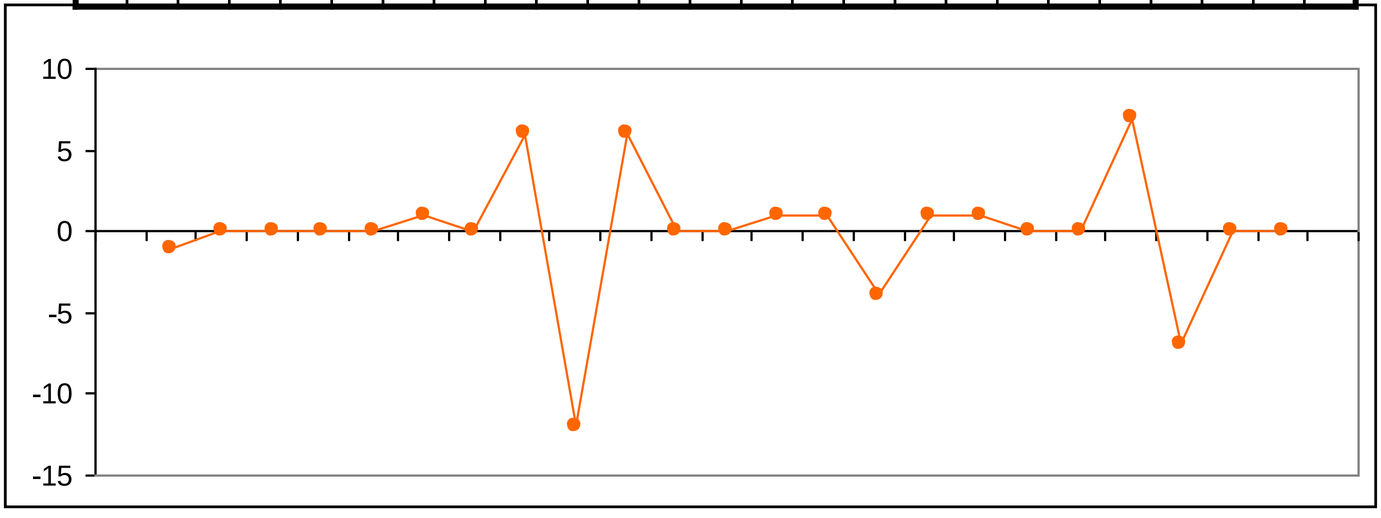
Simply takes into account the values both before and after the current value

# 2nd Derivative (cont…)




| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 4 | 3 | 2 | 1 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 |

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | -1 | 0 | 0 | 0 | 0 | 1 | 0 | 6 | -12 | 6 | 0 | 0 | 1 | 1 | -4 | 1 | 1 | 0 | 0 | 7 | -7 | 0 | 0 | |



# Using Second Derivatives For Image Enhancement

- The $2^{nd}$ derivative is more useful for image enhancement than the $1^{st}$ derivative
  - Stronger response to fine detail
  - Simpler implementation
  - We will come back to the $1^{st}$ order derivative later on
- The first sharpening filter we will look at is the *Laplacian*
  - Isotropic
  - One of the simplest sharpening filters
  - We will look at a digital implementation

# The Laplacian

The Laplacian is defined as follows:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

where the partial 1st order derivative in the $x$ direction is defined as follows:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

and in the $y$ direction as follows:

$$\frac{\partial^2 f}{\partial^2 y} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

# The Laplacian (cont…)

So, the Laplacian can be given as follows:

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$$

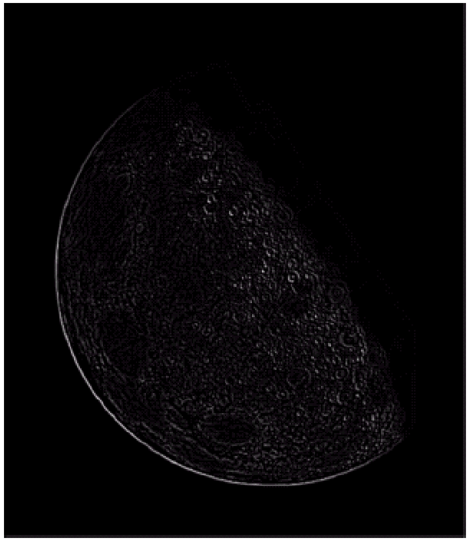
We can easily build a filter based on this

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

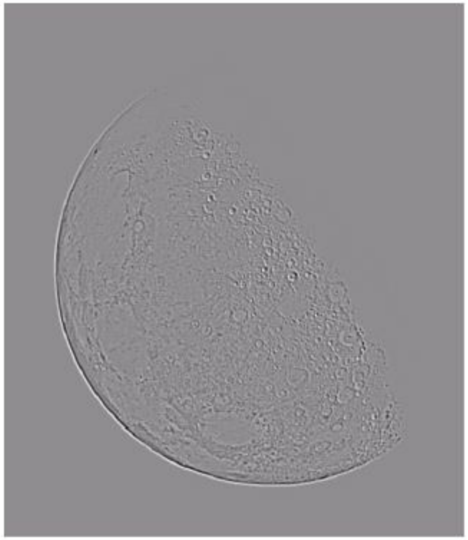
# The Laplacian (cont…)

Applying the Laplacian to an image we get a new image that highlights edges and other discontinuities

Original
Image



Laplacian
Filtered Image



Laplacian
Filtered Image
Scaled for Display

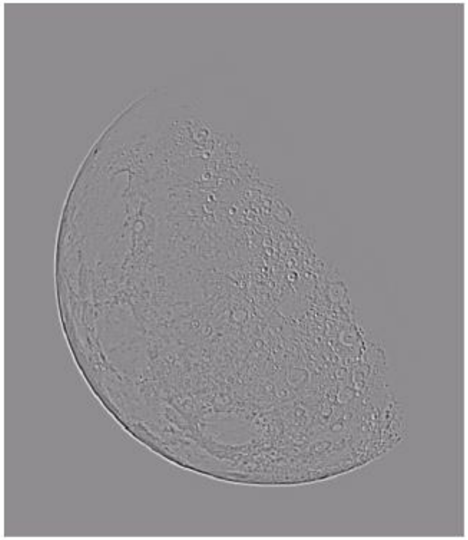
# But That Is Not Very Enhanced!

The result of a Laplacian filtering is not an enhanced image

We have to do more work in order to get our final image

Subtract the Laplacian result from the original image to generate our final sharpened enhanced image

$$g(x, y) = f(x, y) - \nabla^2 f$$



Laplacian
Filtered Image
Scaled for Display

# Laplacian Image Enhancement

Original
Image

-

Laplacian
Filtered Image

=

Sharpened
Image

In the final sharpened image edges and fine detail are much more obvious

# Laplacian Image Enhancement

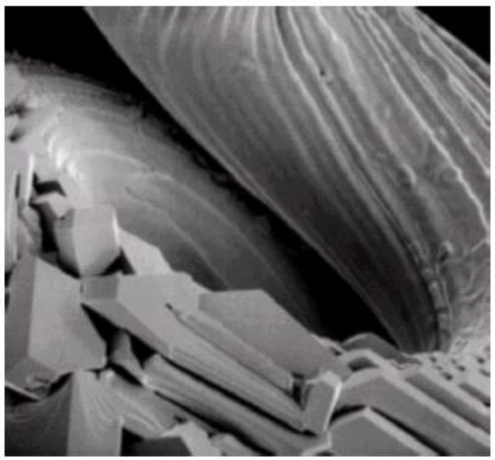
# Simplified Image Enhancement

The entire enhancement can be combined into a single filtering operation

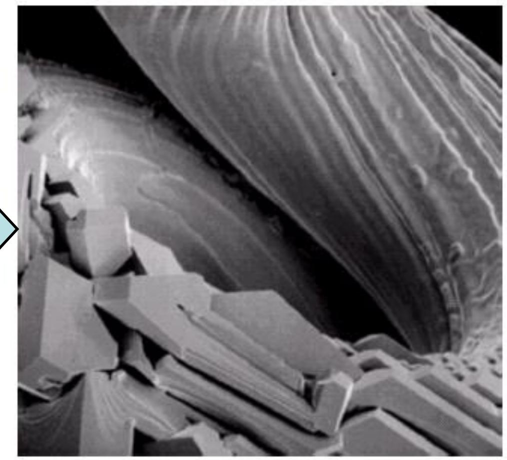
$$
\begin{aligned}
g(x, y) = f(x, y) - \nabla^2 f \\
&= f(x, y) - [f(x+1, y) + f(x-1, y) \\
&\quad + f(x, y+1) + f(x, y-1) \\
&\quad - 4f(x, y)] \\
&= 5f(x, y) - f(x+1, y) - f(x-1, y) \\
&\quad - f(x, y+1) - f(x, y-1)
\end{aligned}
$$

# Simplified Image Enhancement (cont…)

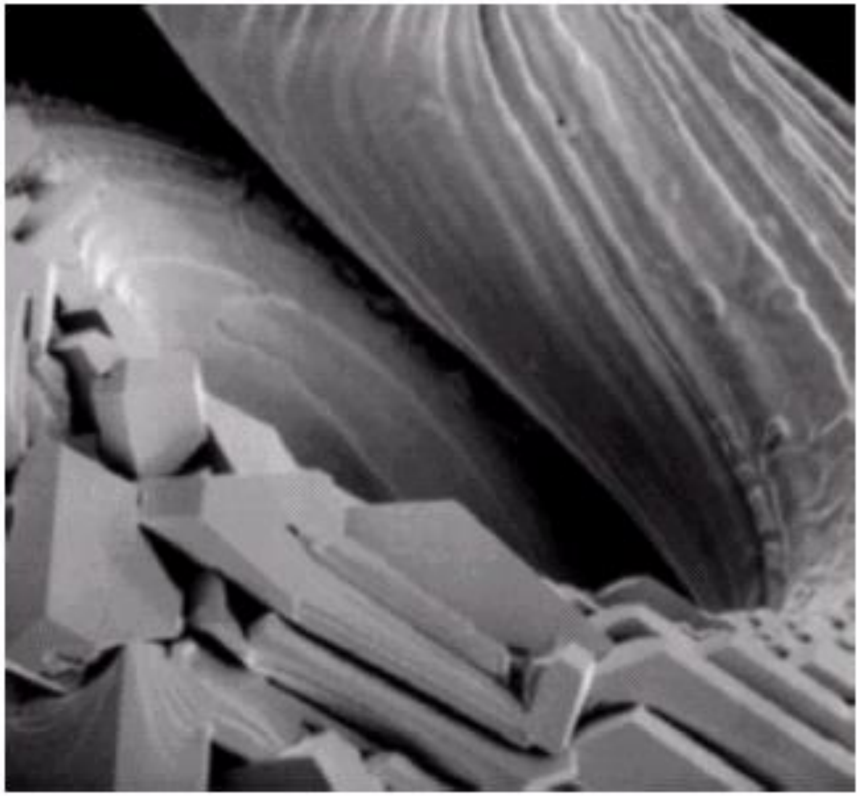
This gives us a new filter which does the whole job for us in one step

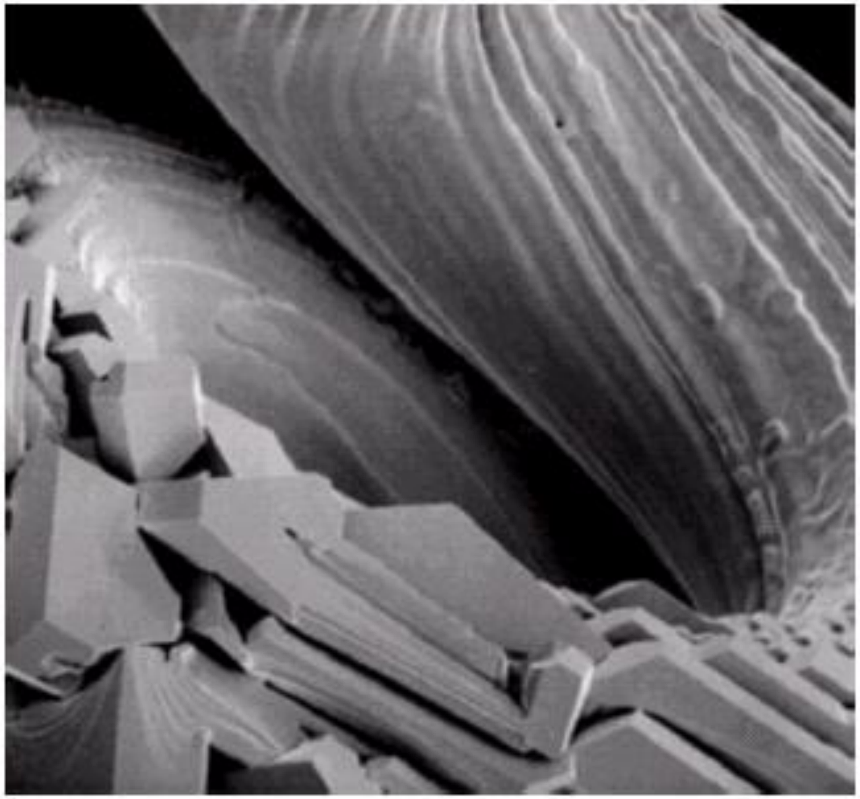


| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |



# Simplified Image Enhancement (cont…)





# Variants On The Simple Laplacian

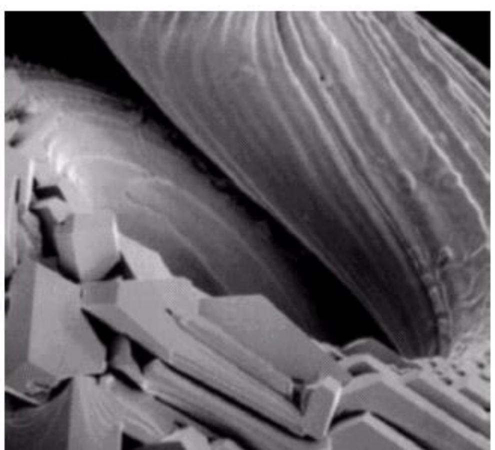
There are lots of slightly different versions of the Laplacian that can be used:

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

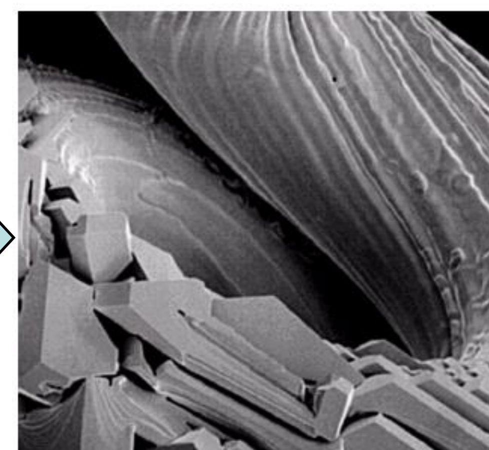
Simple Laplacian

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

Variant of Laplacian



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



# $1^{st}$ Derivative Filtering

Implementing $1^{st}$ derivative filters is difficult in practice

For a function $f(x, y)$ the gradient of $f$ at coordinates $(x, y)$ is given as the column vector:

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

# $1^{st}$ Derivative Filtering (cont…)

The magnitude of this vector is given by:

$$\begin{aligned}\nabla f &= mag(\nabla \mathbf{f}) \\ &= \left[G_x^2 + G_y^2\right]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}\end{aligned}$$

For practical reasons this can be simplified as:

$$\nabla f \approx \left|G_x\right| + \left|G_y\right|$$

# 1st Derivative Filtering (cont…)

There is some debate as to how best to calculate these gradients but we will use:

$$\nabla f \approx \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

which is based on these coordinates

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

# Sobel Operators

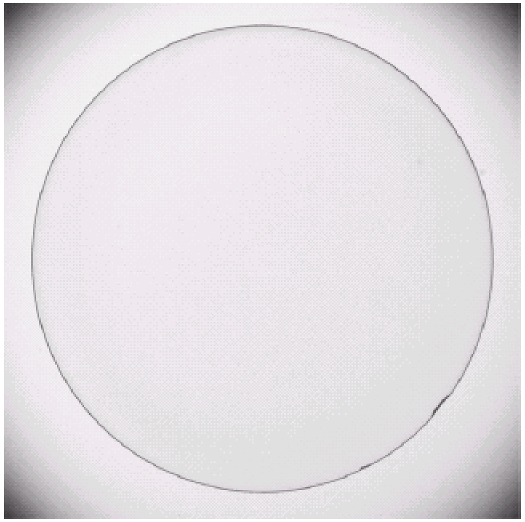
Based on the previous equations we can derive the *Sobel Operators*

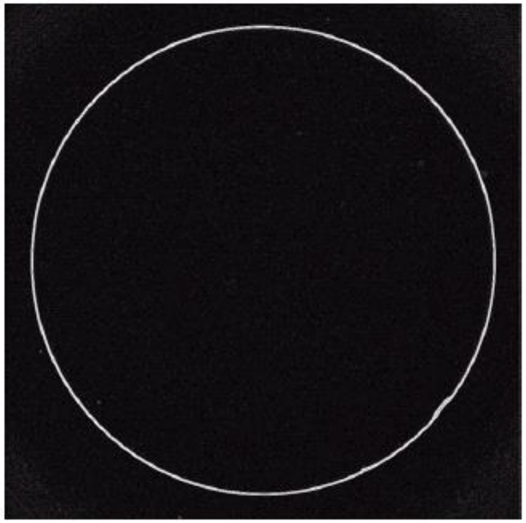
| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

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

To filter an image it is filtered using both operators the results of which are added together

# Sobel Example





An image of a contact lens which is enhanced in order to make defects (at four and five o'clock in the image) more obvious

Sobel filters are typically used for edge detection

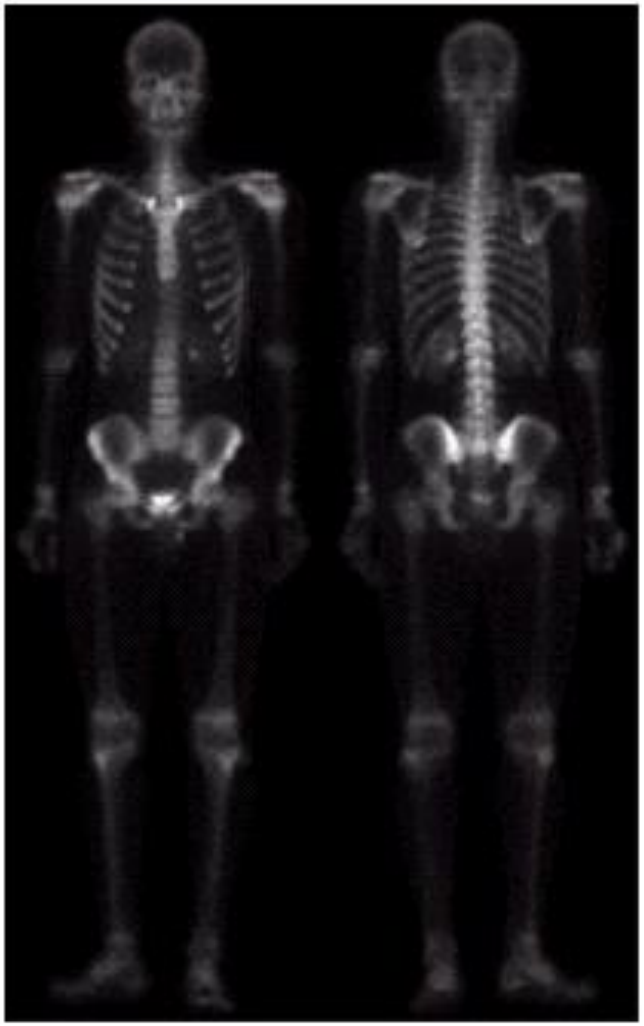
# $1^{st}$ & $2^{nd}$ Derivatives

Comparing the $1^{st}$ and $2^{nd}$ derivatives we can conclude the following:

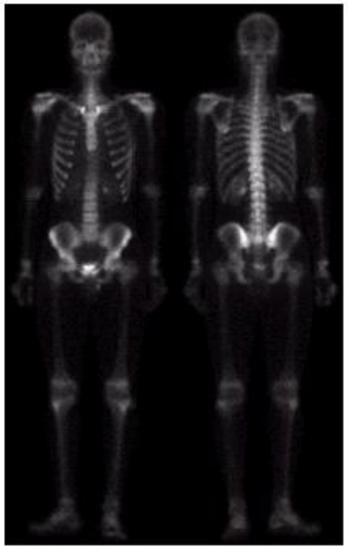
- $1^{st}$ order derivatives generally produce thicker edges
- $2^{nd}$ order derivatives have a stronger response to fine detail e.g. thin lines
- $1^{st}$ order derivatives have stronger response to grey level step
- $2^{nd}$ order derivatives produce a double response at step changes in grey level

# Combining Spatial Enhancement Methods

- Successful image enhancement is typically not achieved using a single operation
- Rather we combine a range of techniques in order to achieve a final result
- This example will focus on enhancing the bone scan to the right

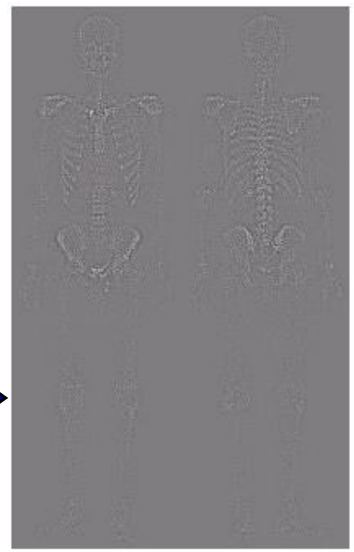


# Combining Spatial Enhancement Methods (cont…)



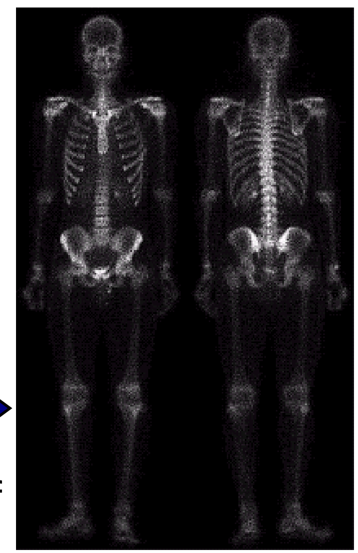
**(a)**

Laplacian filter of bone scan (a)



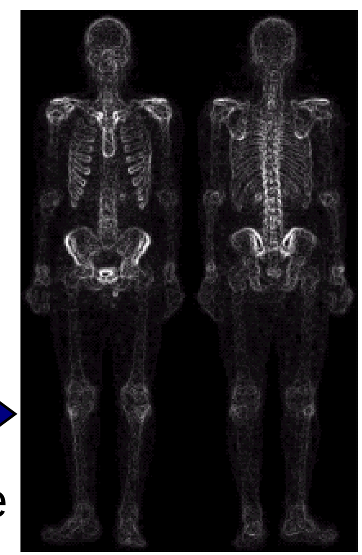
**(b)**

Sharpened version of bone scan achieved by subtracting (a) and (b)



**(c)**

Sobel filter of bone scan (a)



**(d)**

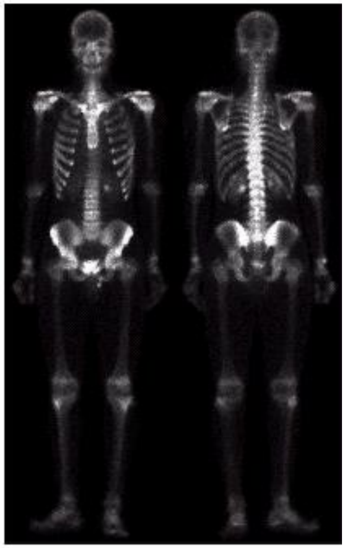
# Combining Spatial Enhancement Methods (cont…)

Result of applying a power-law trans. to (g)

**(h)**

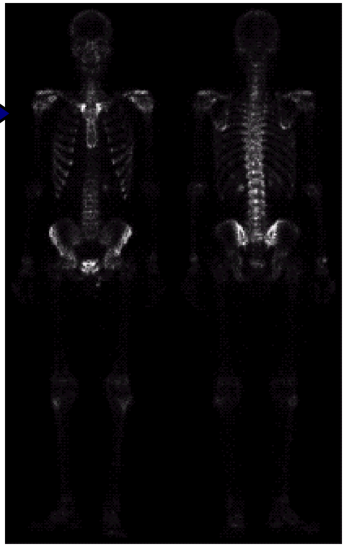
Sharpened image which is sum of (a) and (f)

**(g)**



The product of (c) and (e) which will be used as a mask

**(f)**



**(e)**

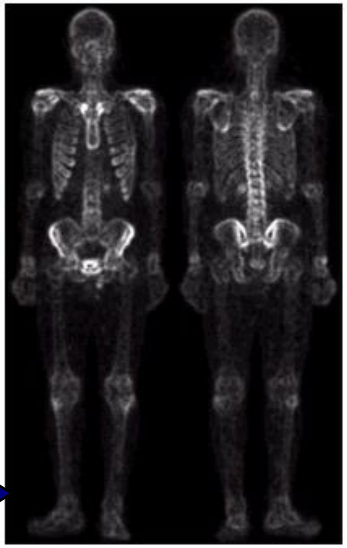
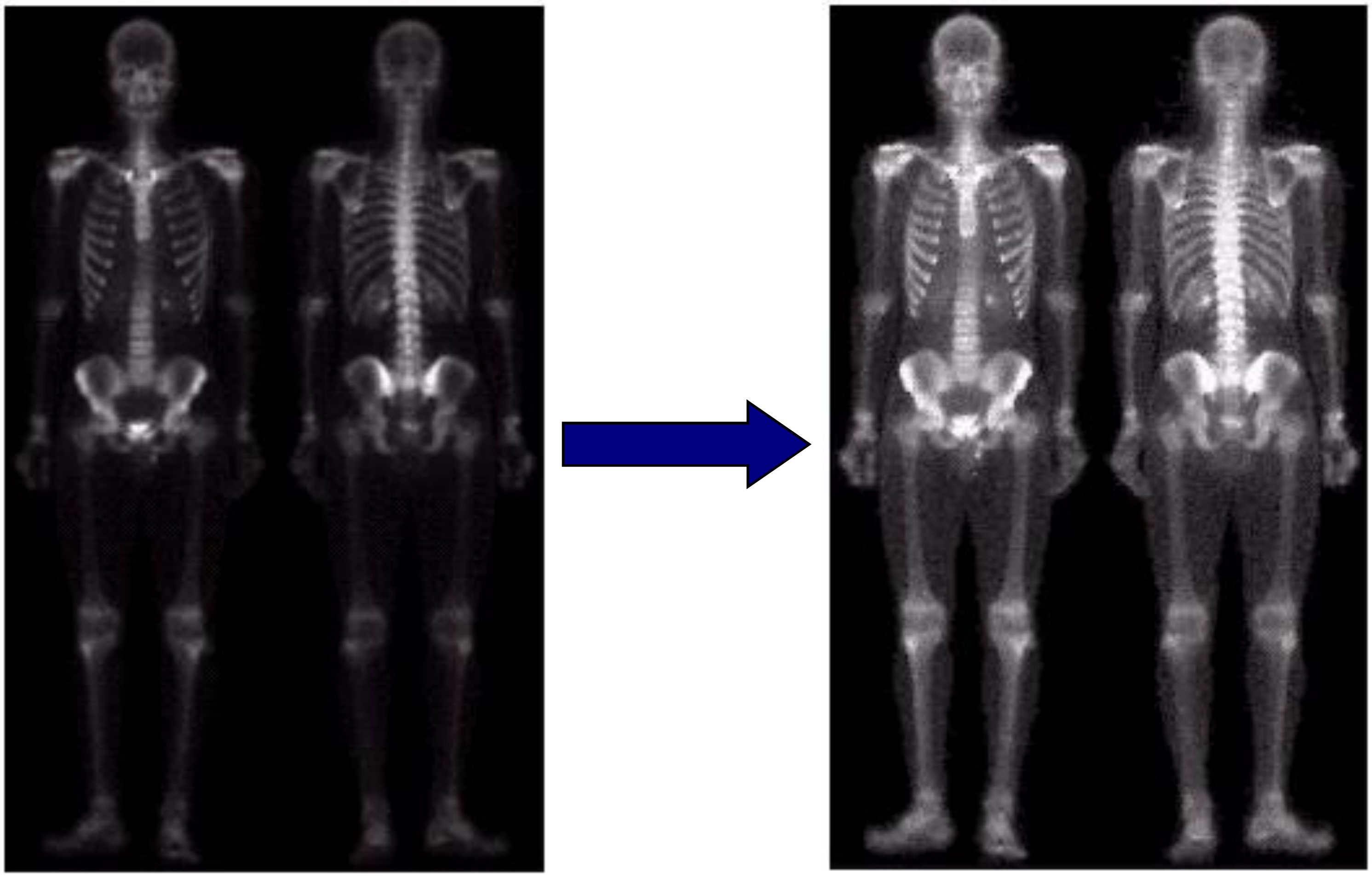


Image (d) smoothed with a 5*5 averaging filter

# Combining Spatial Enhancement Methods (cont…)

Compare the original and final images



# Summary

In this lecture we looked at:

- Sharpening filters
  - $1^{st}$ derivative filters
  - $2^{nd}$ derivative filters
- Combining filtering techniques
- Neighbourhood operations
- The filtering process
- Smoothing filters
- Dealing with problems at image edges when using filtering
- Correlation and convolution

DIP

LECTURE No. ( 7 )

# *Image Enhancement:*

# *Filtering in the Frequency Domain*

10/14/2019

Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

# Contents

In this lecture we will look at image enhancement in the frequency domain

- Jean Baptiste Joseph Fourier
- The Fourier series & the Fourier transform
- Image Processing in the frequency domain
  - Image smoothing
  - Image sharpening
- Fast Fourier Transform

# Jean Baptiste Joseph Fourier

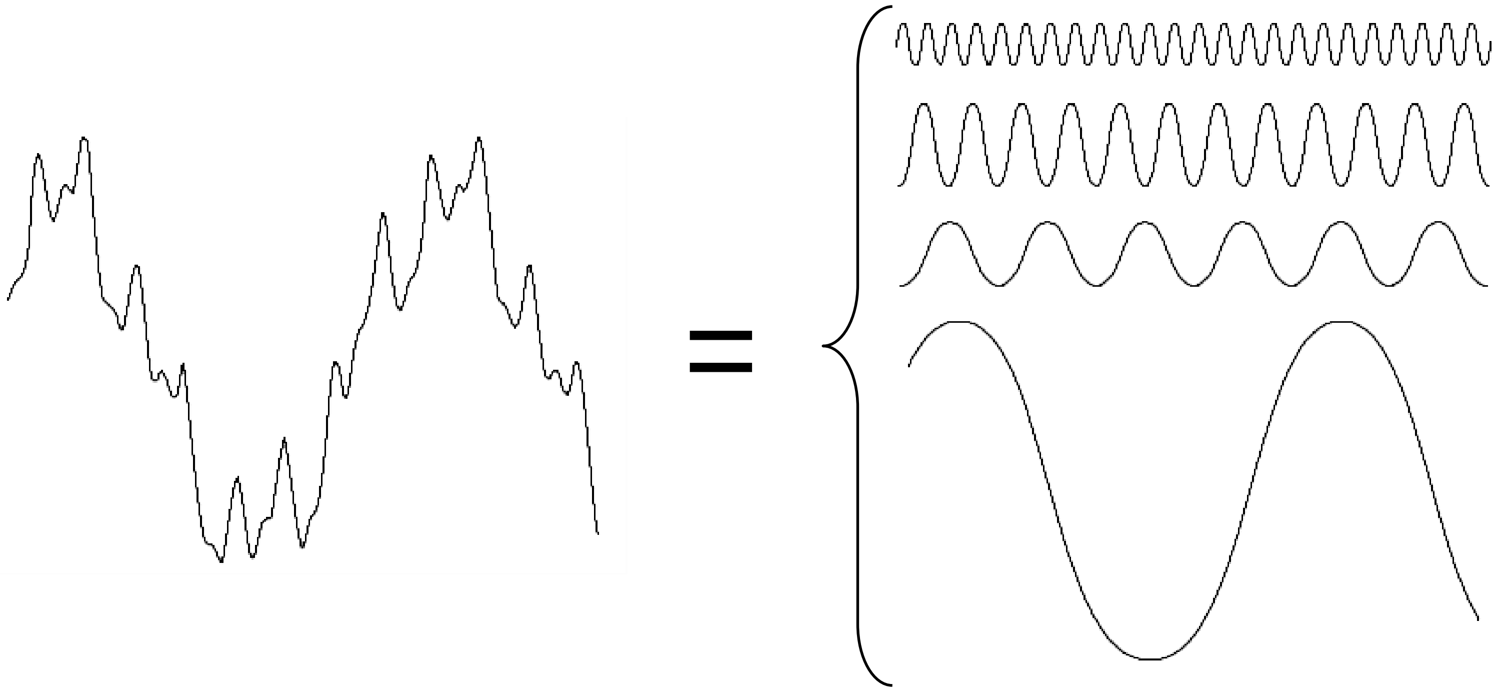
Fourier was born in Auxerre, France in 1768

- Most famous for his work "*La Théorie Analitique de la Chaleur*" published in 1822
- Translated into English in 1878: "*The Analytic Theory of Heat*"

Nobody paid much attention when the work was first published

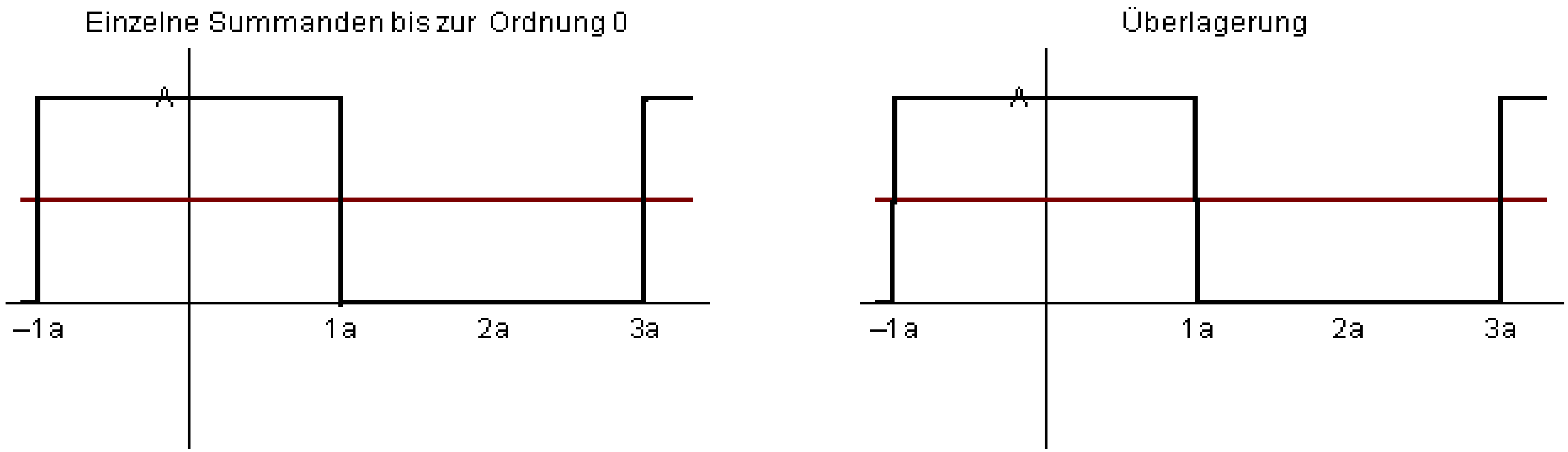
One of the most important mathematical theories in modern engineering

# The Big Idea



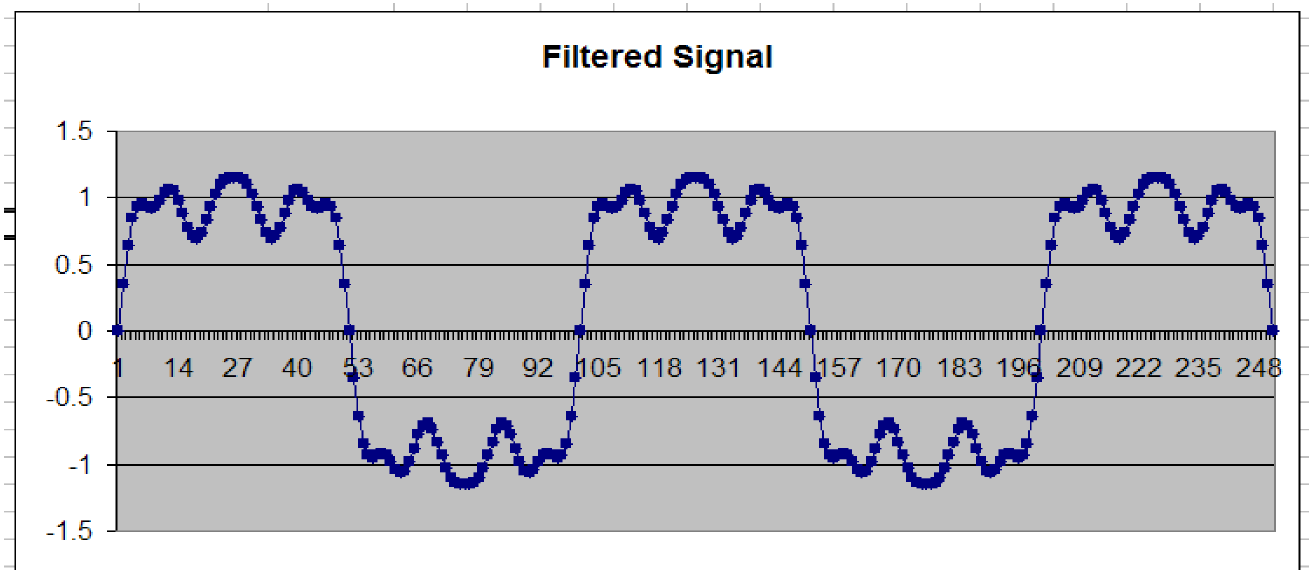
Any function that periodically repeats itself can be expressed as a sum of sines and cosines of different frequencies each multiplied by a different coefficient – a *Fourier series*

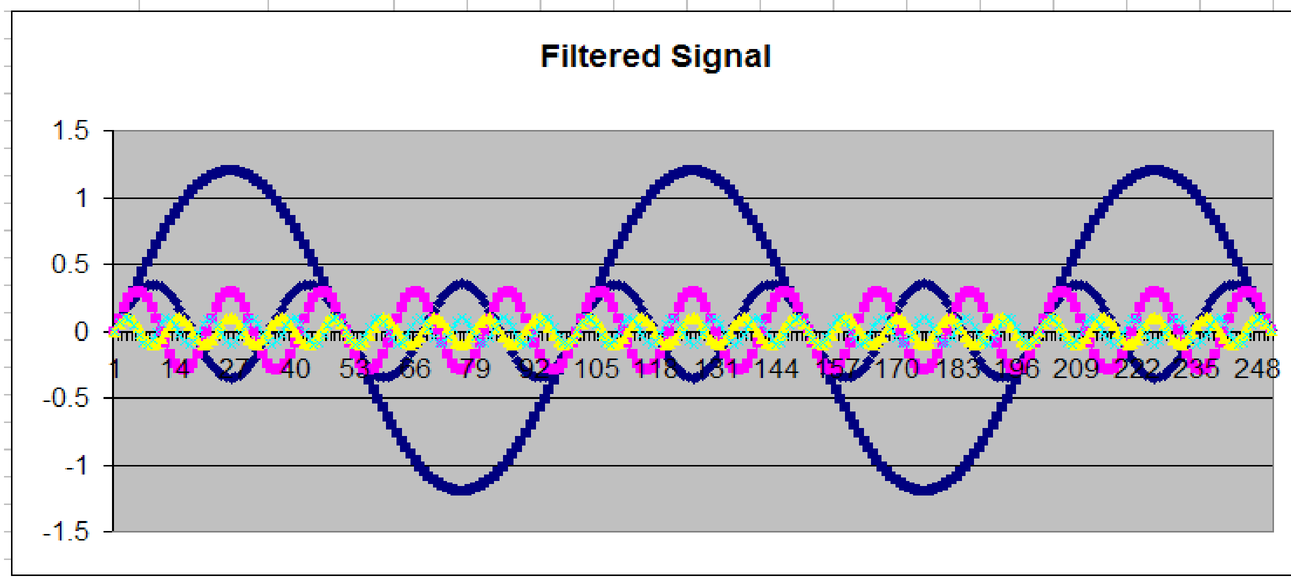
# The Big Idea (cont…)




Notice how we get closer and closer to the original function as we add more and more frequencies

# The Big Idea (cont…)




Frequency domain signal processing example in Excel

# The Discrete Fourier Transform (DFT)

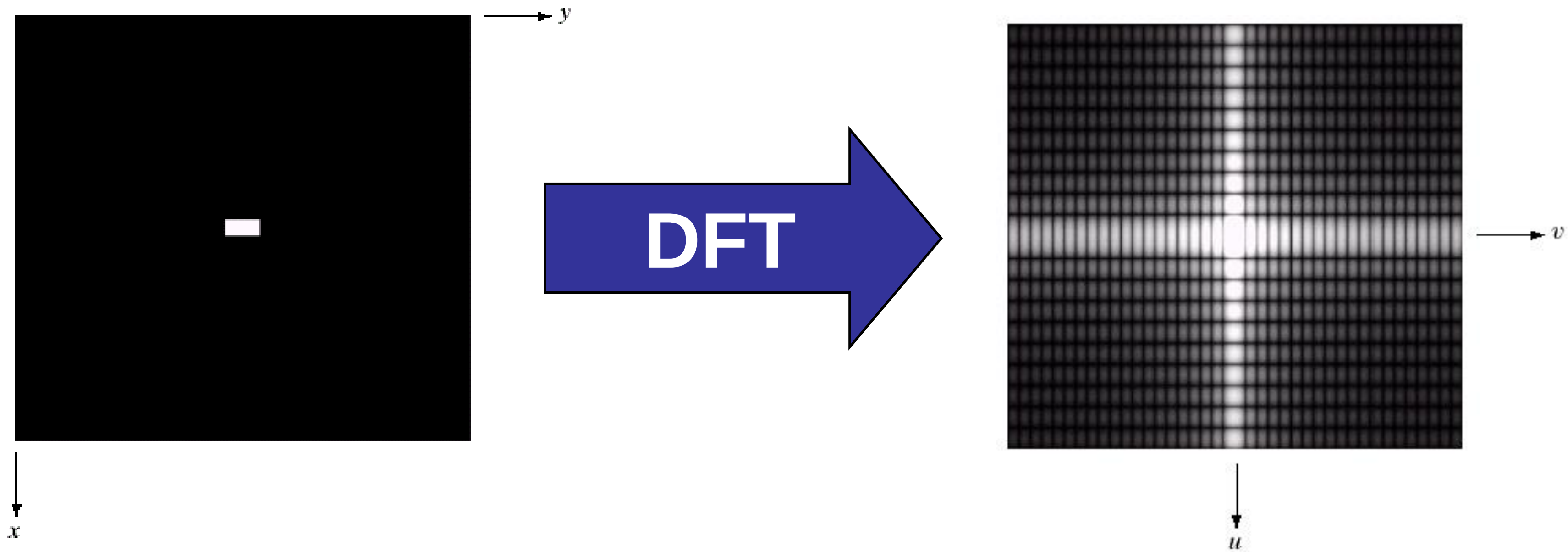
The *Discrete Fourier Transform* of $f(x, y)$, for $x$ = 0, 1, 2…M-1 and $y$ = 0,1,2…N-1, denoted by $F(u, v)$, is given by the equation:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)}$$

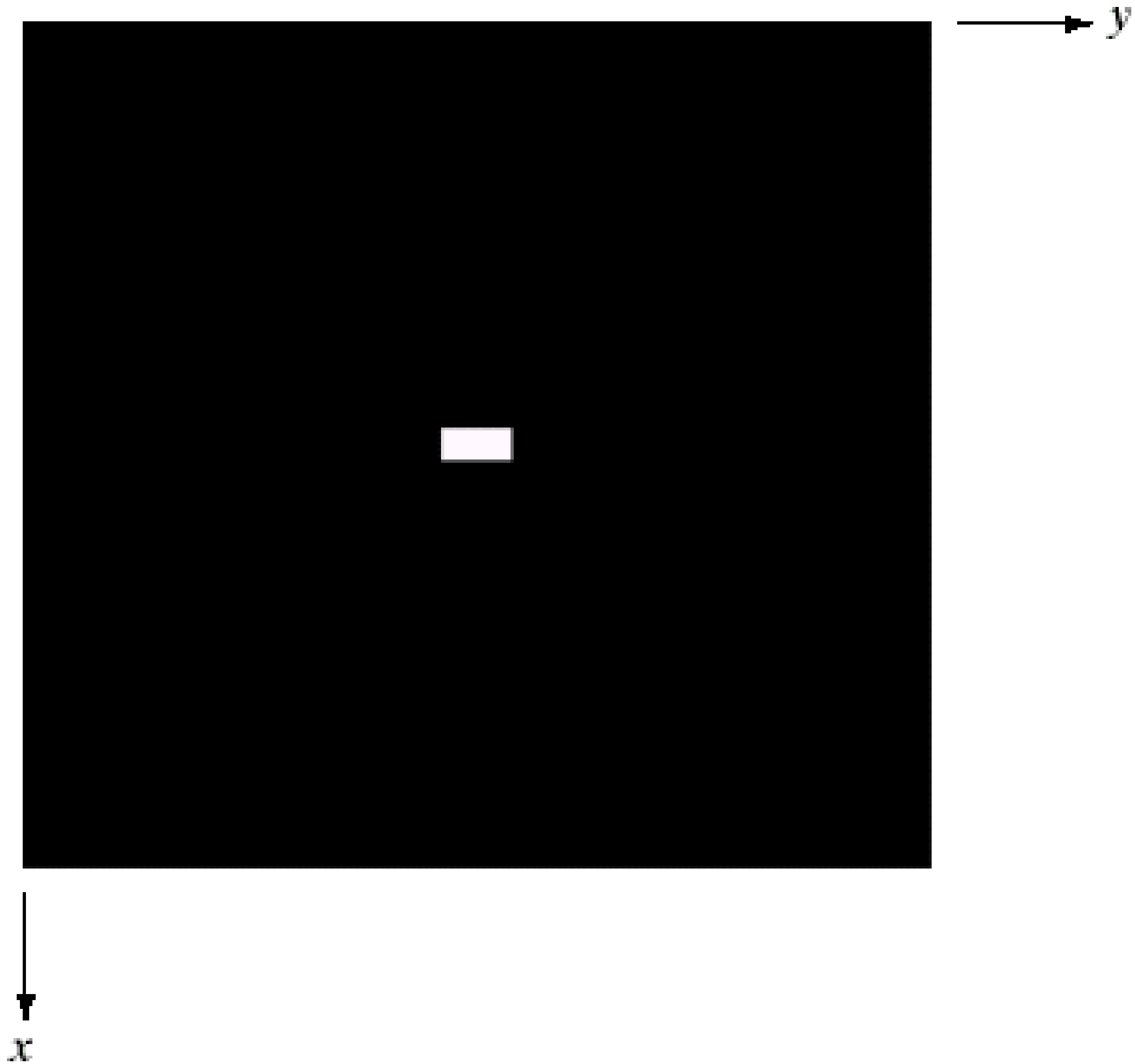
for $u$ = 0, 1, 2…M-1 and $v$ = 0, 1, 2…N-1.

# DFT & Images

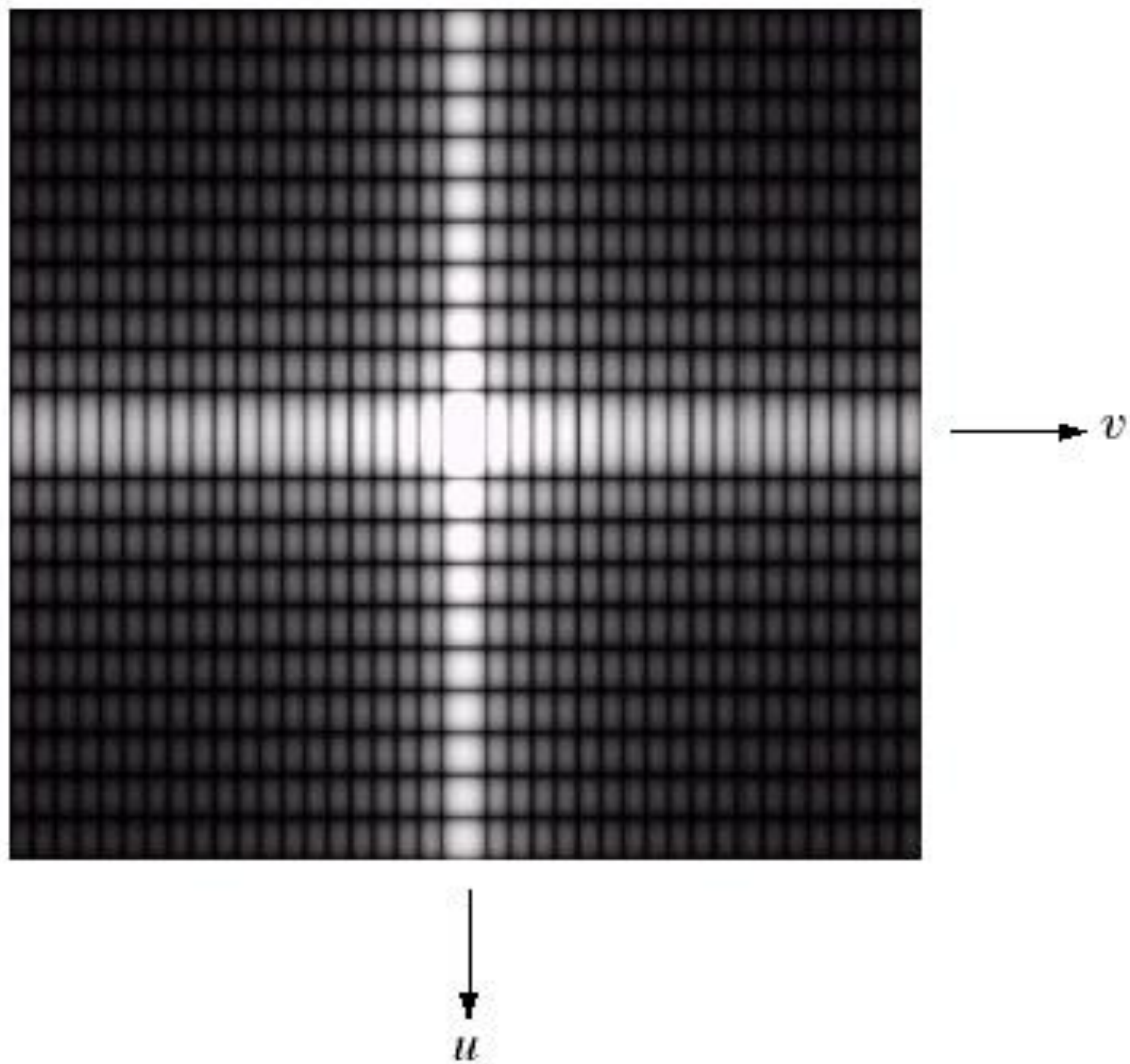
The DFT of a two dimensional image can be visualised by showing the spectrum of the images component frequencies



# DFT & Images



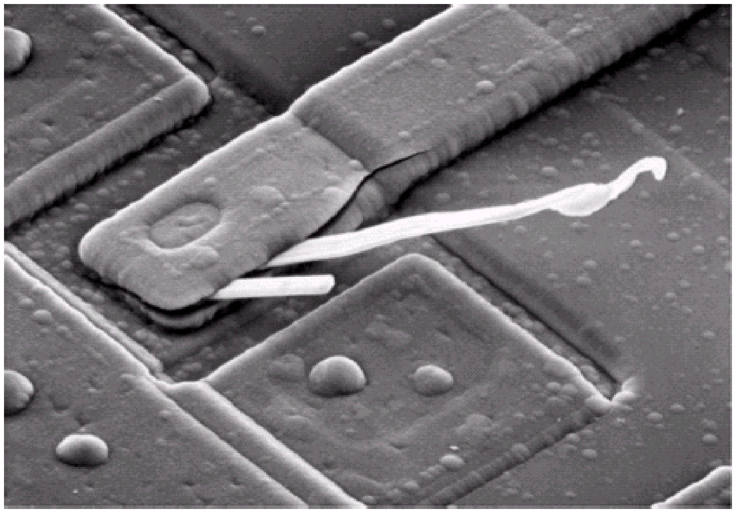
# DFT & Images



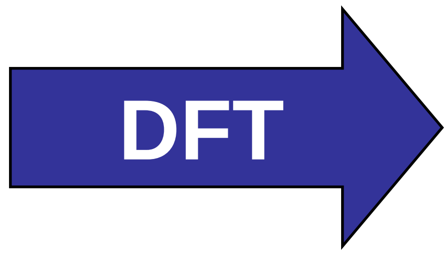
# Properties in the frequency domain

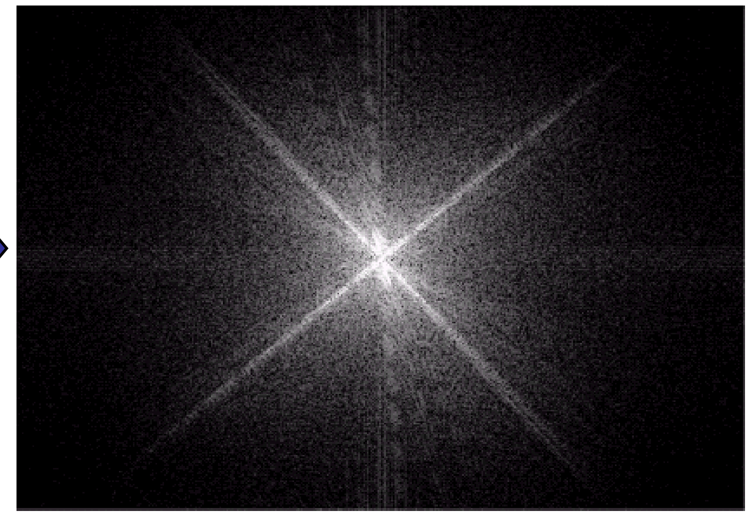
- Fourier transform works globally
  - No direct relationship between a specific components in an image and frequencies
- Intuition about frequency
  - Frequency content
  - Rate of change of gray levels in an image

# DFT & Images (cont…)



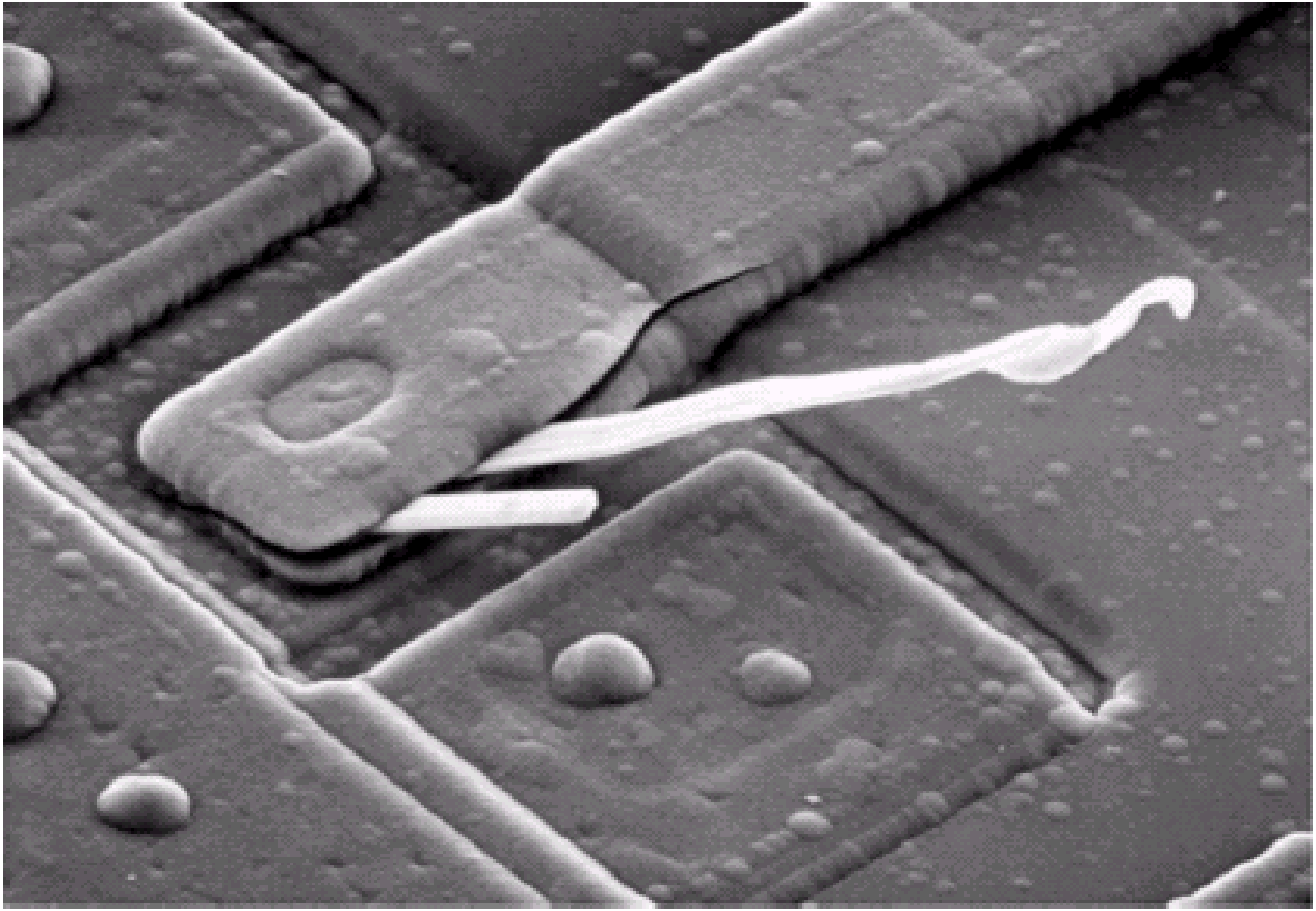
Scanning electron microscope image of an integrated circuit magnified ~2500 times



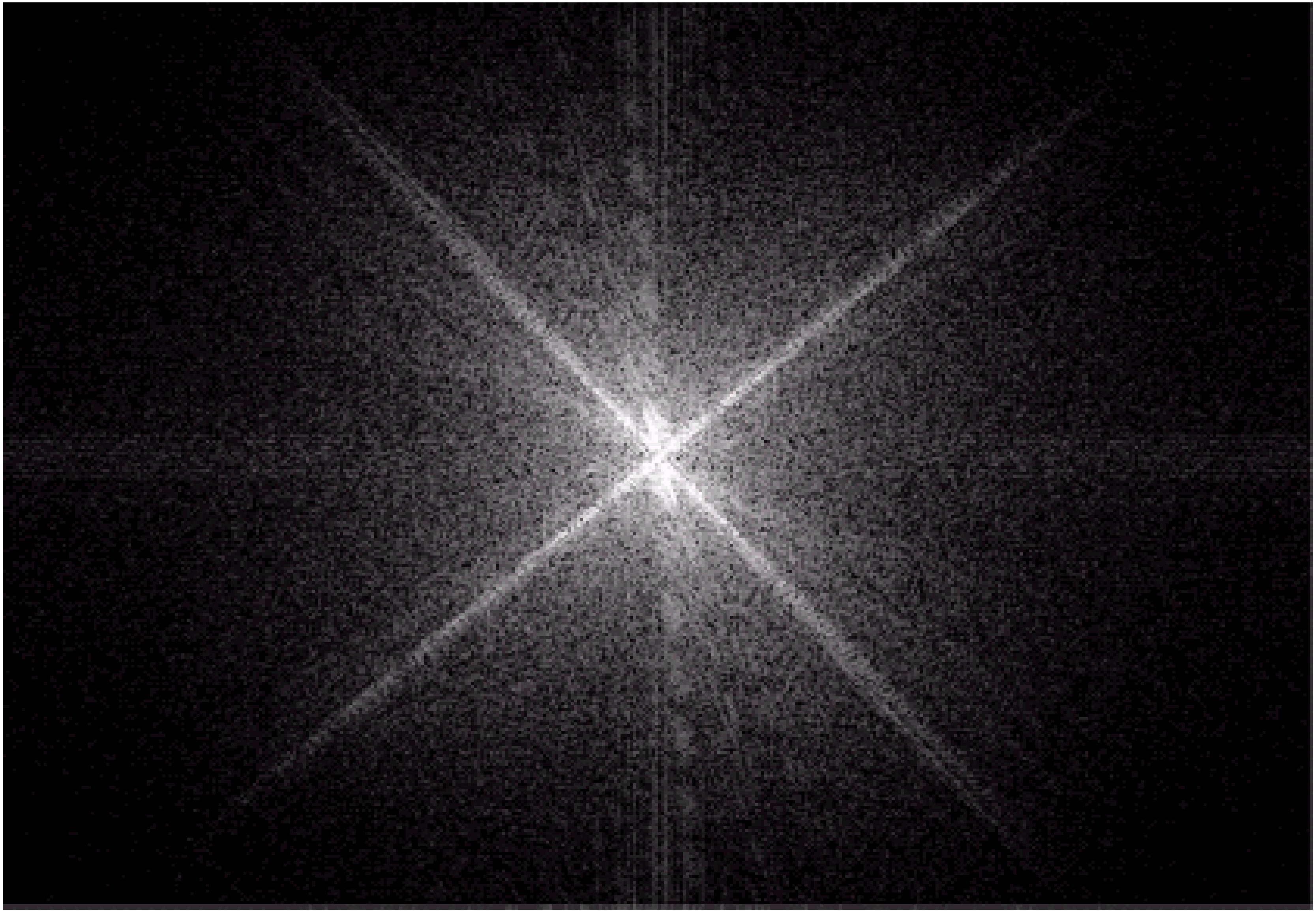



Fourier spectrum of the image

# DFT & Images (cont…)



# DFT & Images (cont…)



# The Inverse DFT

It is really important to note that the Fourier transform is completely **reversible**

The inverse DFT is given by:

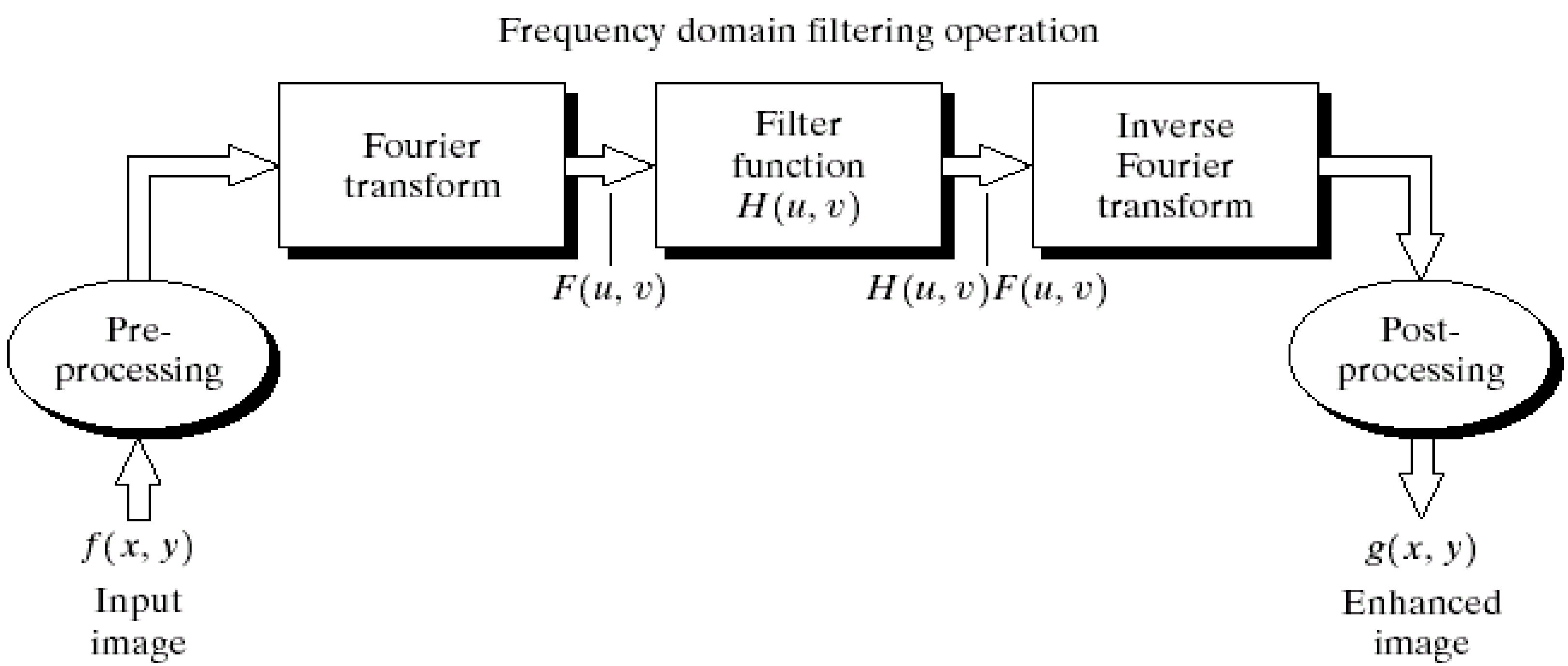
$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{j2\pi(ux/M+vy/N)}$$

for $x$ = 0, 1, 2…M-1 and $y$ = 0, 1, 2…N-1

# The DFT and Image Processing

To filter an image in the frequency domain:

1. Compute *F(u,v)* the DFT of the image
2. Multiply *F(u,v)* by a filter function *H(u,v)*
3. Compute the inverse DFT of the result

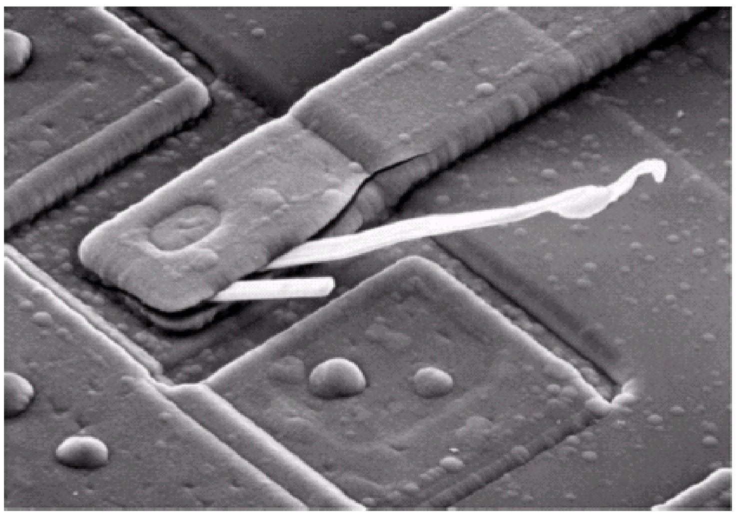


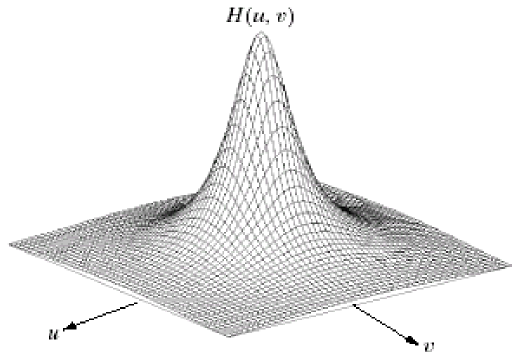
# Basics of filtering in the frequency domain

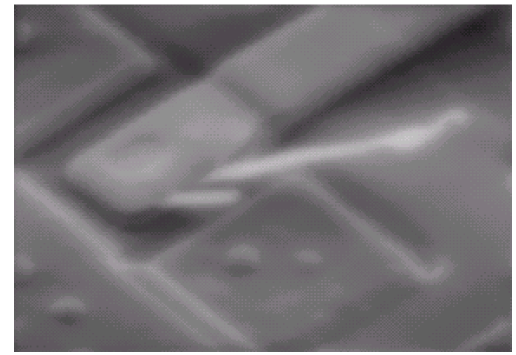
- multiply the input image by $(-1)^{x+y}$ to center the transform to u = M/2 and v = N/2 (if M and N are even numbers, then the shifted coordinates will be integers)
- computer F(u,v), the DFT
- multiply F(u,v) by a filter function H(u,v)
- compute the inverse DFT
- obtain the real part
- multiply the result in  by $(-1)^{x+y}$ to cancel the multiplication of the input image.

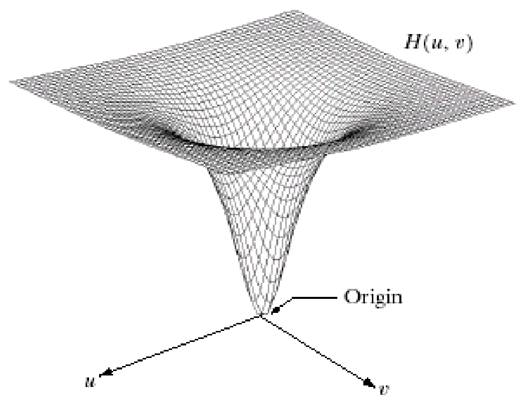
# Some Basic Frequency Domain Filters

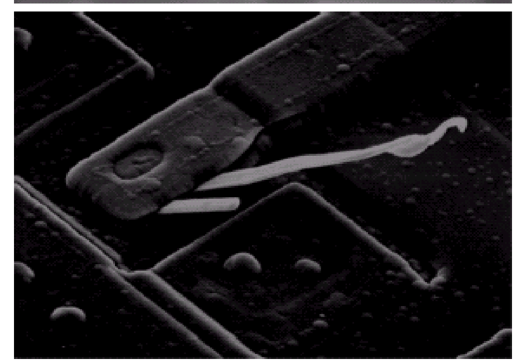
Low Pass Filter





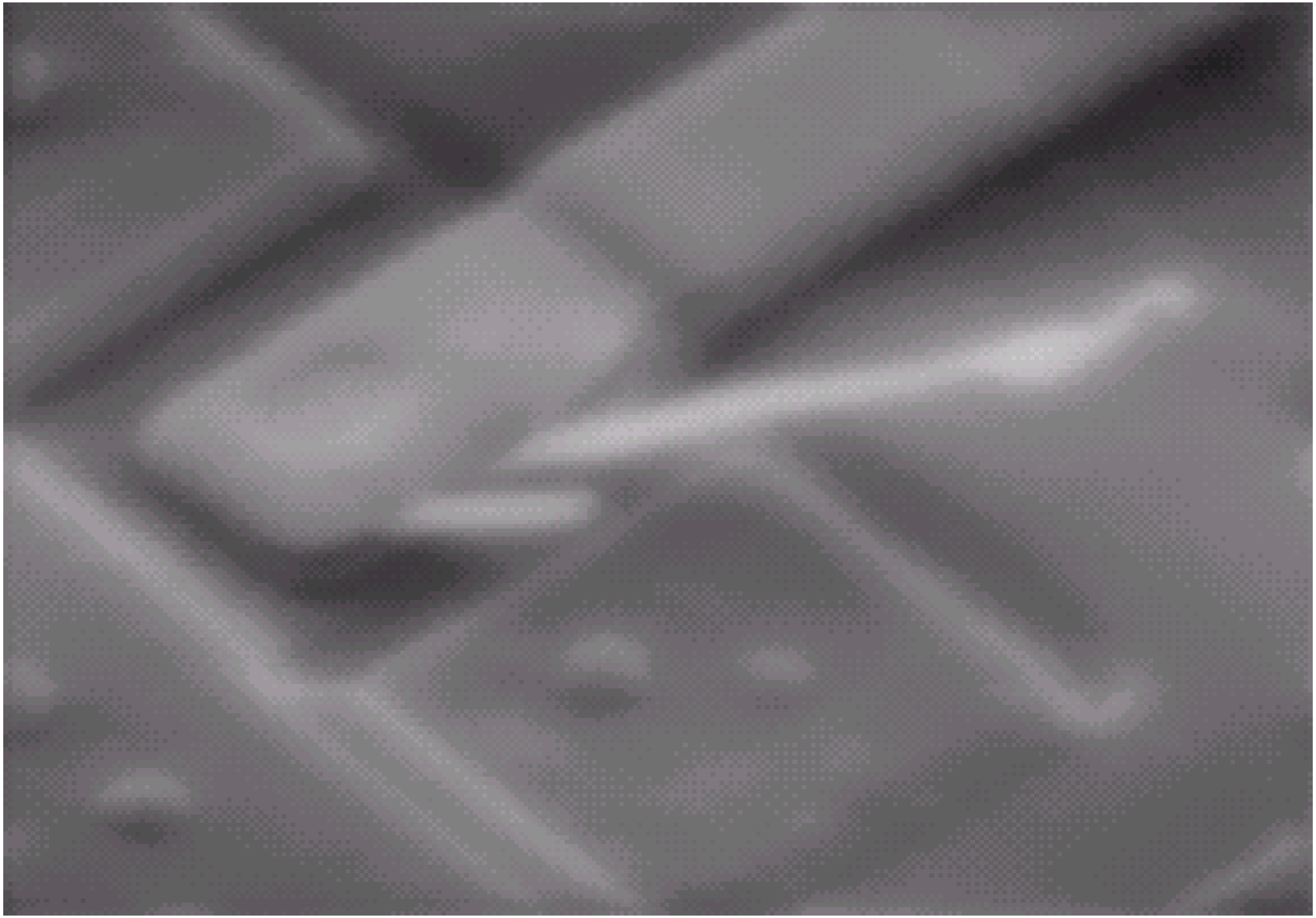





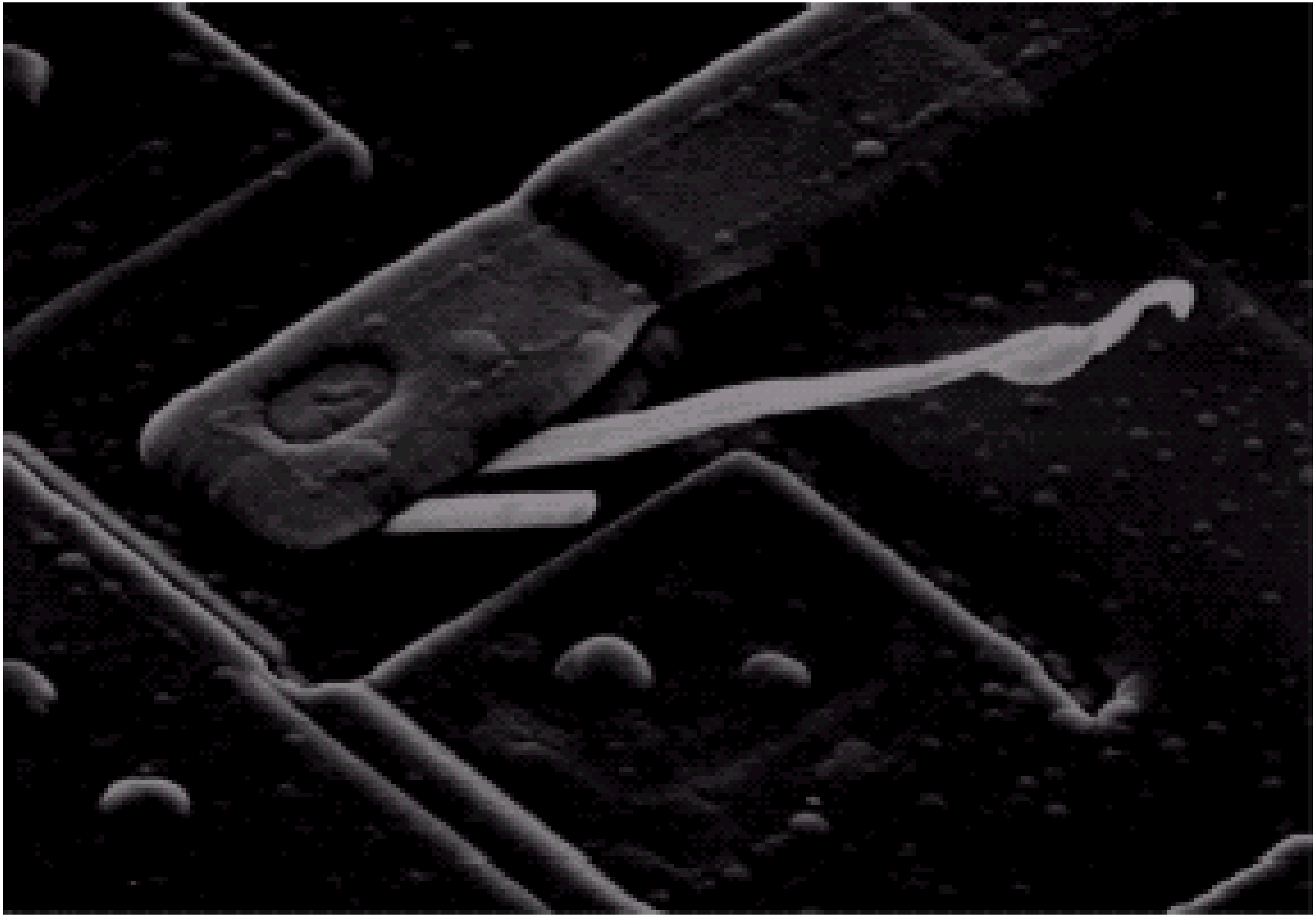



High Pass Filter

# Some Basic Frequency Domain Filters



# Some Basic Frequency Domain Filters



# Smoothing Frequency Domain Filters

Smoothing is achieved in the frequency domain by dropping out the high frequency components

The basic model for filtering is:

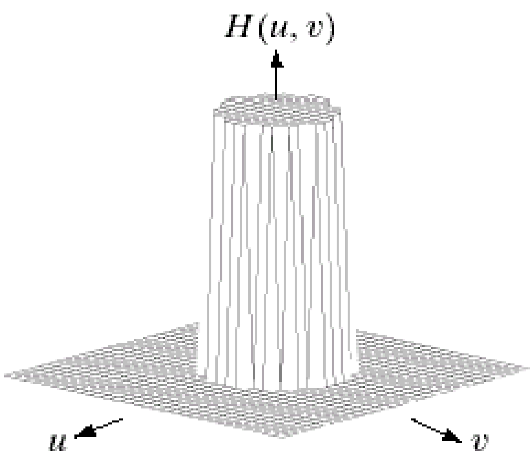
$$G(u,v) = H(u,v)F(u,v)$$

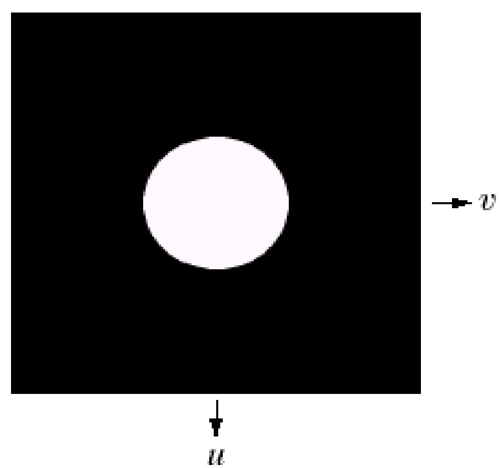
where $F(u,v)$ is the Fourier transform of the image being filtered and $H(u,v)$ is the filter transform function

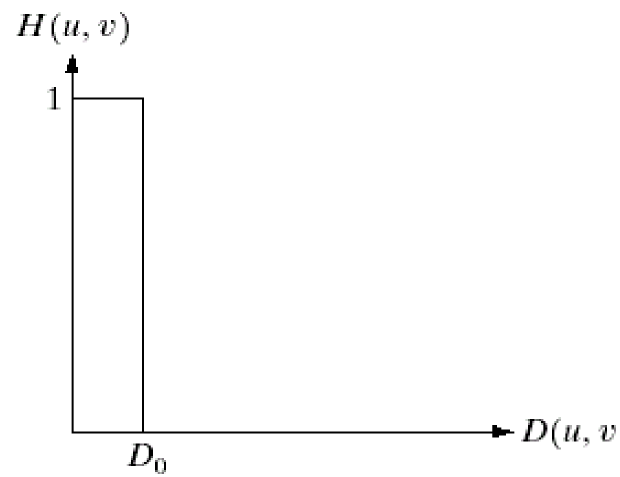
*Low pass filters* – only pass the low frequencies, drop the high ones

# Ideal Low Pass Filter

Simply cut off all high frequency components that are a specified distance $D_0$ from the origin of the transform




changing the distance changes the behaviour of the filter

# Ideal Low Pass Filter (cont…)

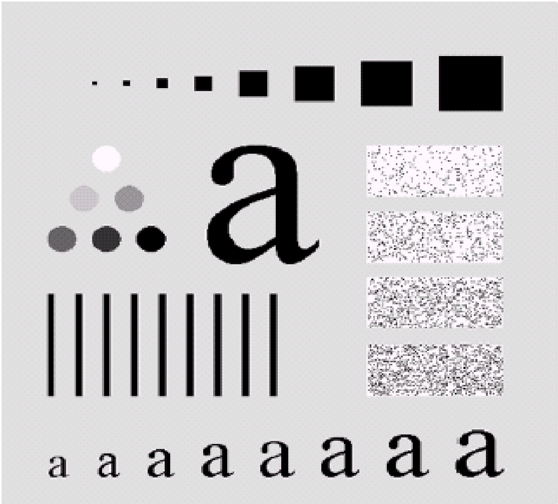
The transfer function for the ideal low pass filter can be given as:

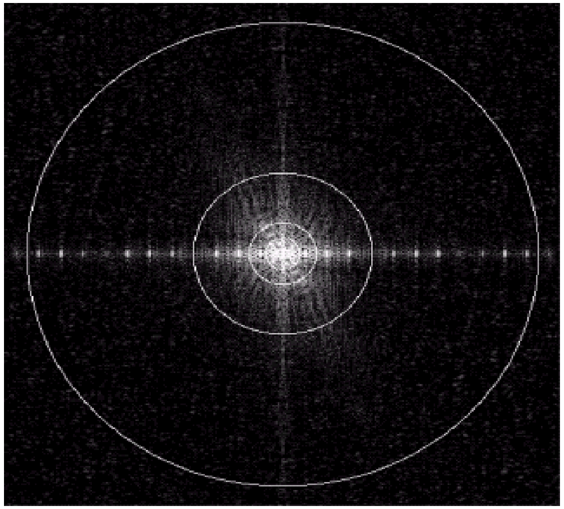
$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

where $D(u,v)$ is given as:

$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$$

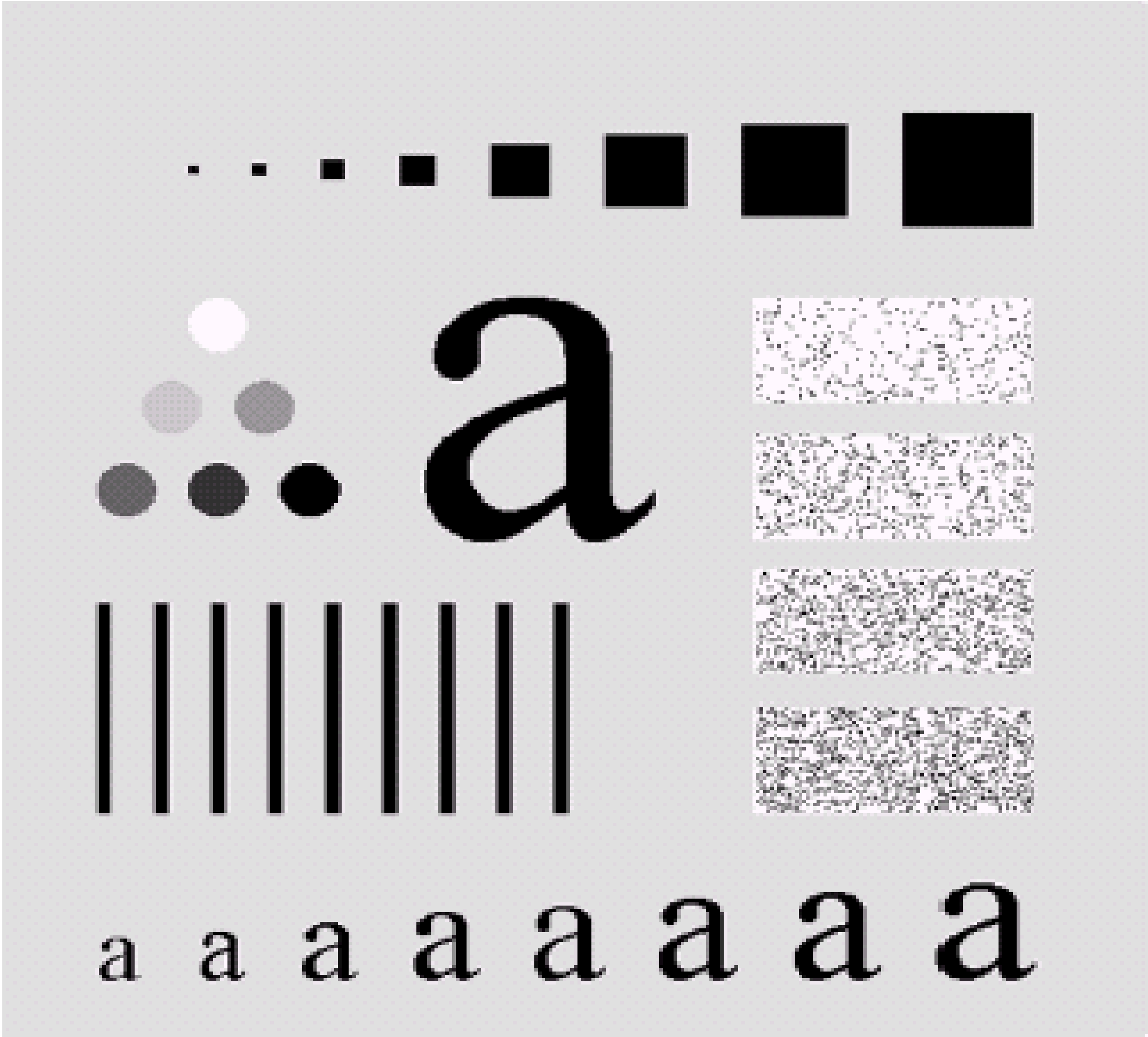
# Ideal Low Pass Filter (cont…)



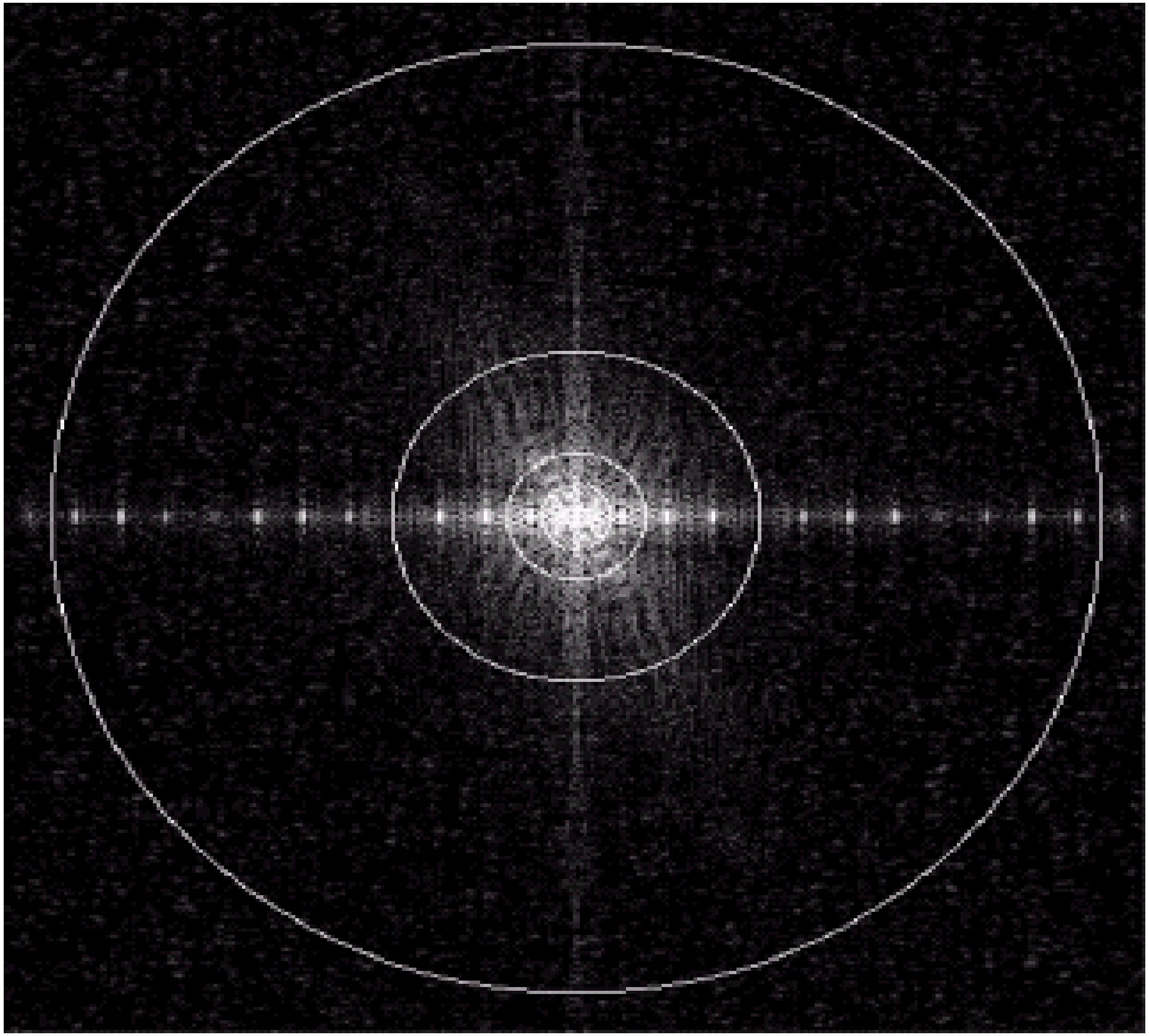



Above we show an image, it's Fourier spectrum and a series of ideal low pass filters of radius 5, 15, 30, 80 and 230 superimposed on top of it

# Ideal Low Pass Filter (cont…)

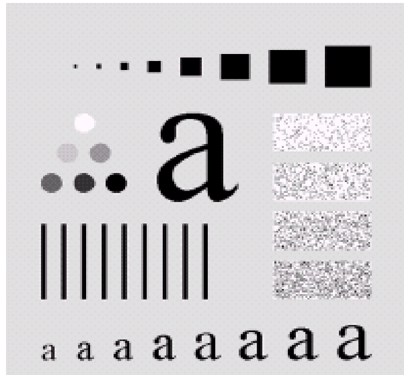


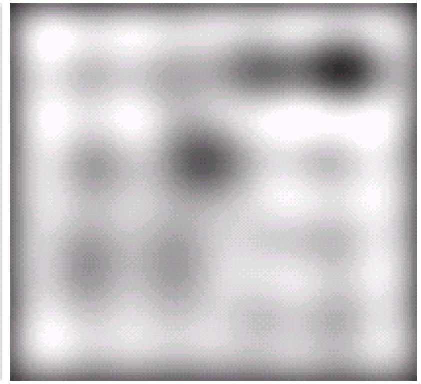
# Ideal Low Pass Filter (cont…)



# Ideal Low Pass Filter (cont…)

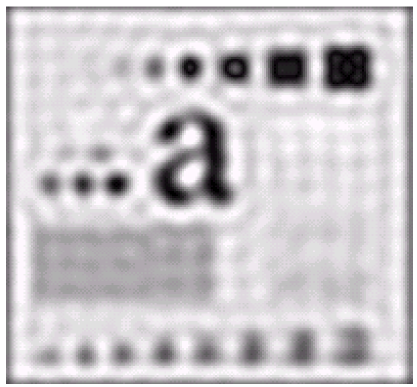
Original image

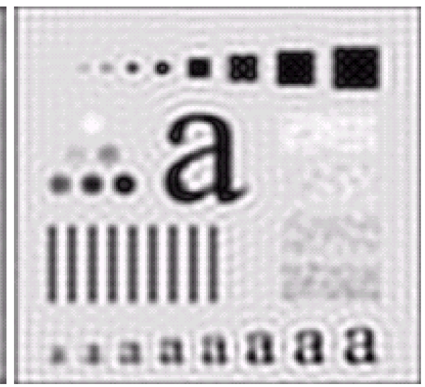




Result of filtering with ideal low pass filter of radius 5

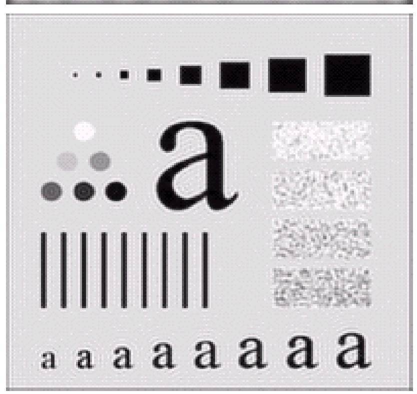
Result of filtering with ideal low pass filter of radius 15

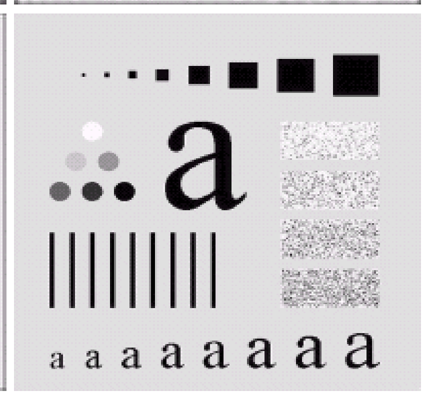




Result of filtering with ideal low pass filter of radius 30

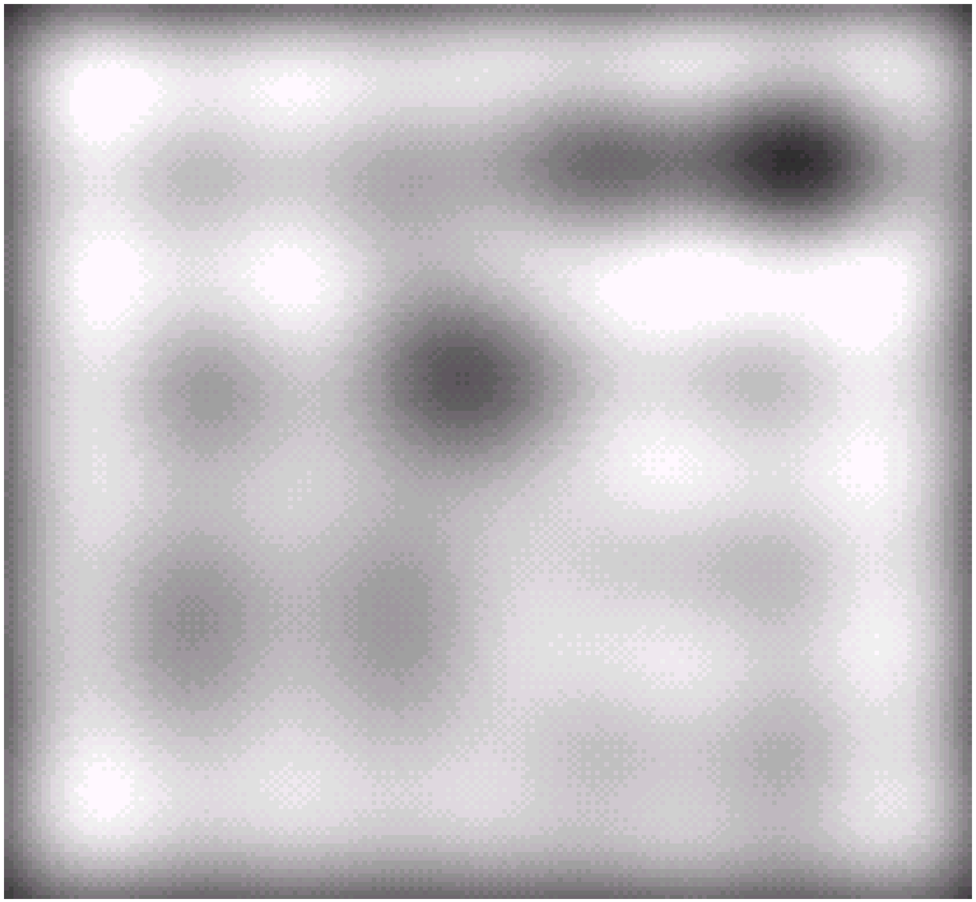
Result of filtering with ideal low pass filter of radius 80





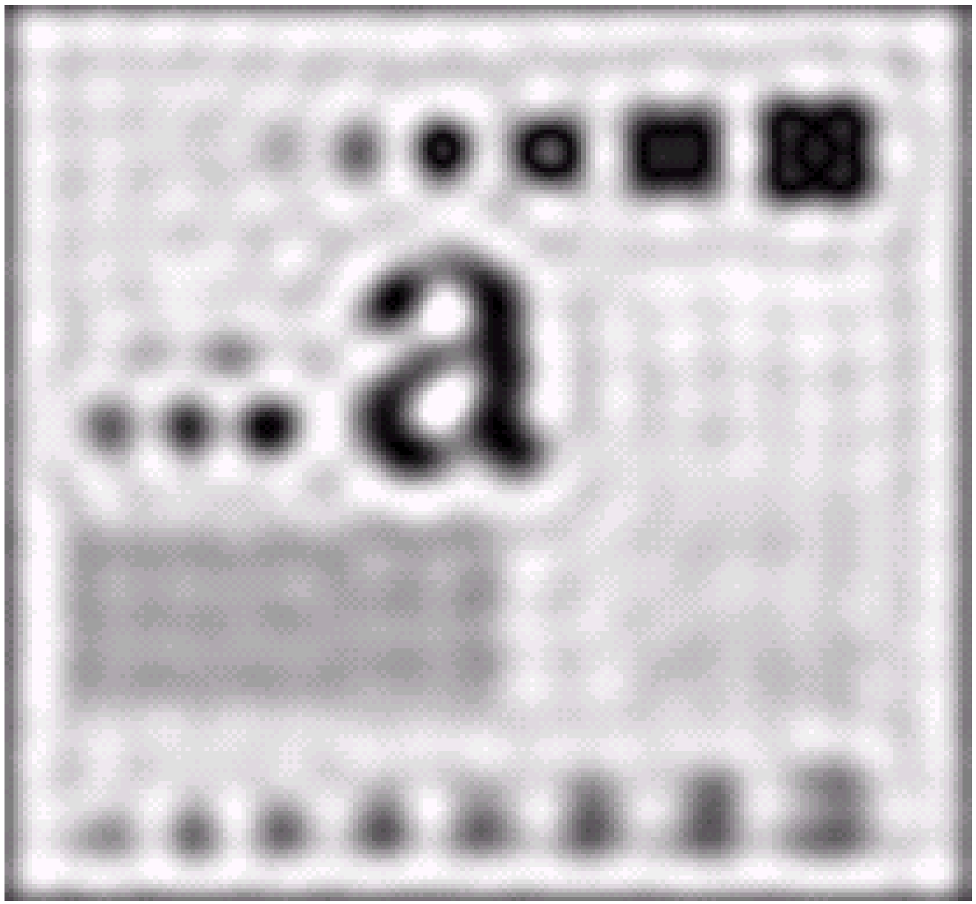
Result of filtering with ideal low pass filter of radius 230

# Ideal Low Pass Filter (cont…)



Result of filtering with ideal low pass filter of radius 5

# Ideal Low Pass Filter (cont…)

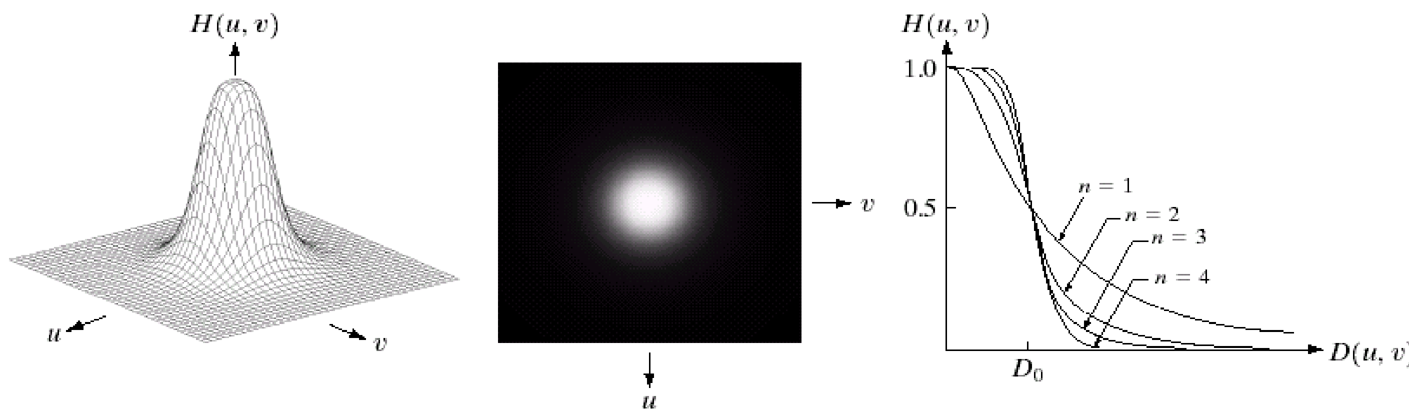



Result of filtering with ideal low pass filter of radius 15

# Butterworth Lowpass Filters

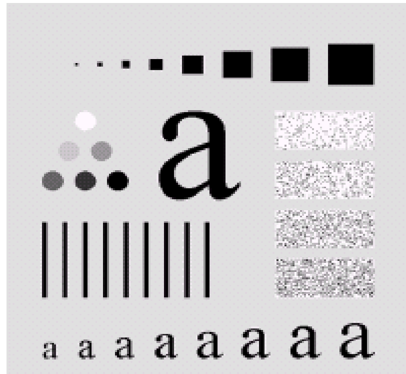
The transfer function of a Butterworth lowpass filter of order $n$ with cutoff frequency at distance $D_0$ from the origin is defined as:

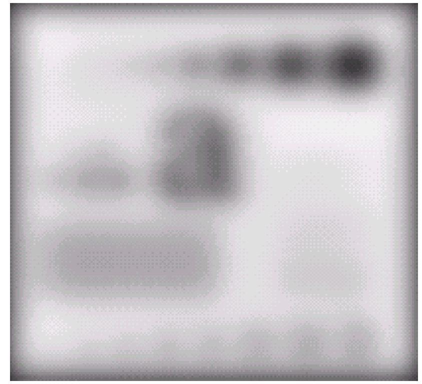
$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$



# Butterworth Lowpass Filter (cont…)

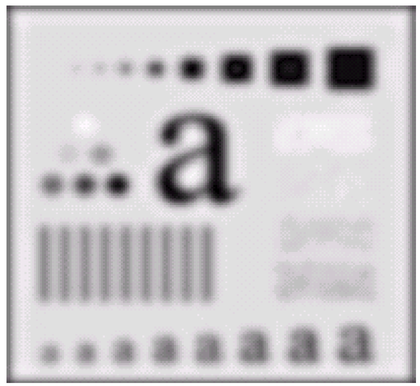
Original image

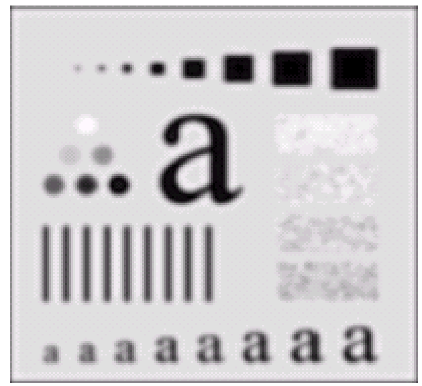




Result of filtering with Butterworth filter of order 2 and cutoff radius 5

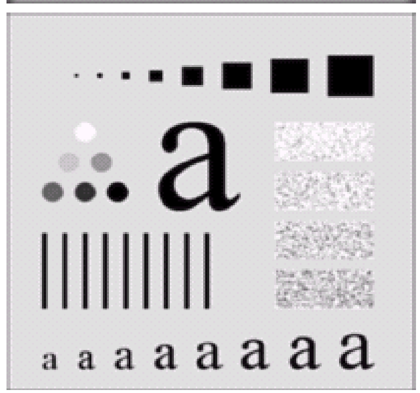
Result of filtering with Butterworth filter of order 2 and cutoff radius 15

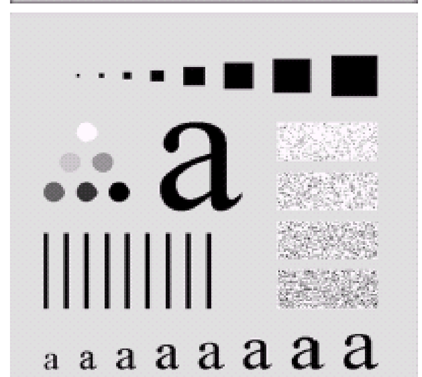




Result of filtering with Butterworth filter of order 2 and cutoff radius 30

Result of filtering with Butterworth filter of order 2 and cutoff radius 80



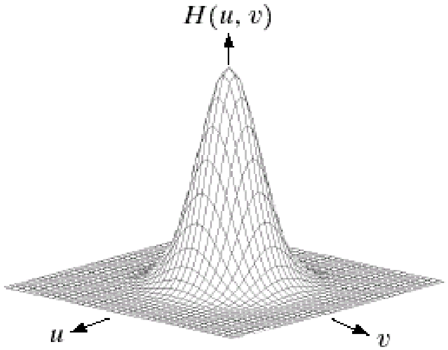


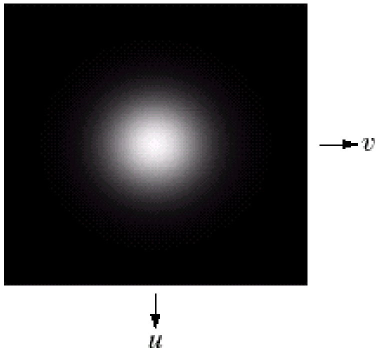
Result of filtering with Butterworth filter of order 2 and cutoff radius 230

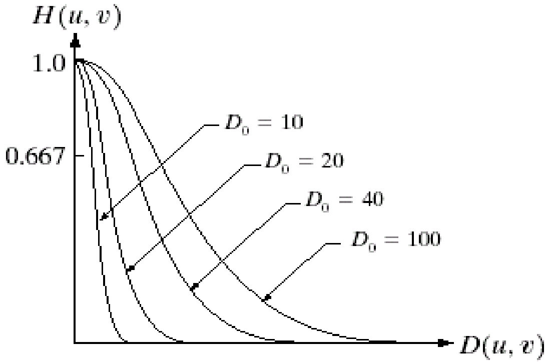
# Gaussian Lowpass Filters

The transfer function of a Gaussian lowpass filter is defined as:

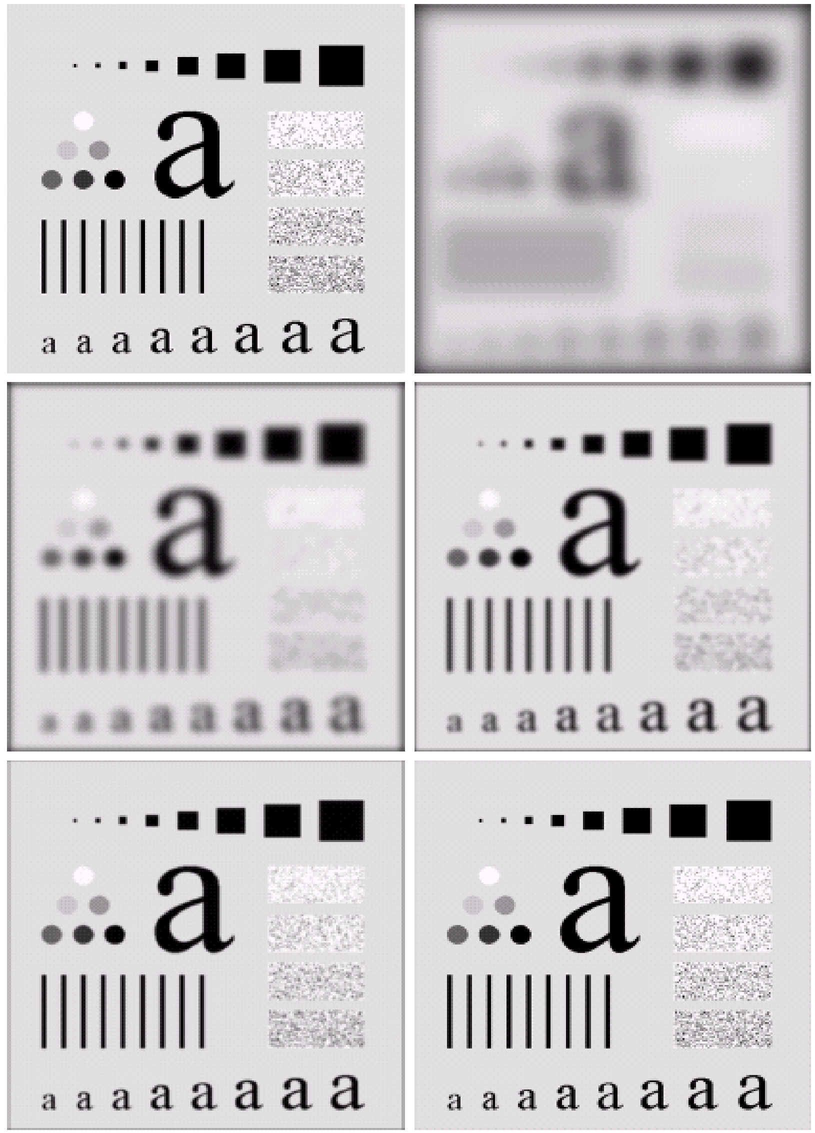
$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$



# Gaussian Lowpass Filters (cont…)



Original image

Result of filtering with Gaussian filter with cutoff radius 5

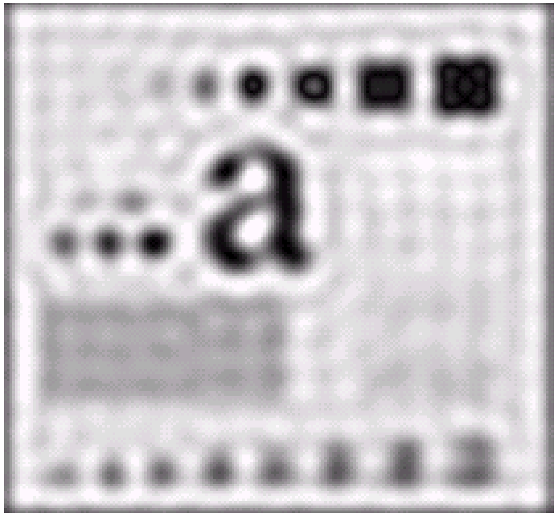
Result of filtering with Gaussian filter with cutoff radius 15

Result of filtering with Gaussian filter with cutoff radius 30

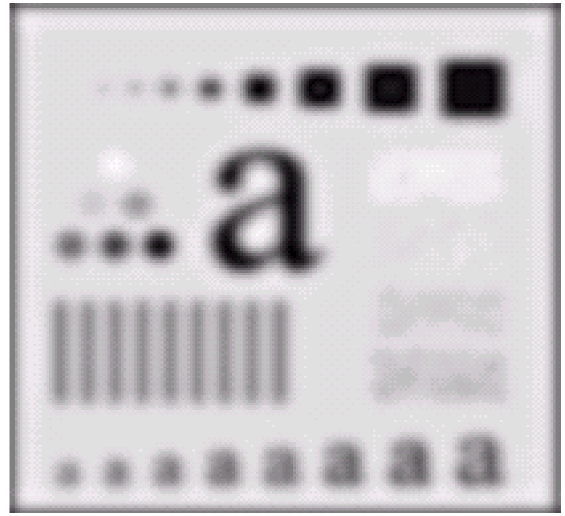
Result of filtering with Gaussian filter with cutoff radius 85

Result of filtering with Gaussian filter with cutoff radius 230

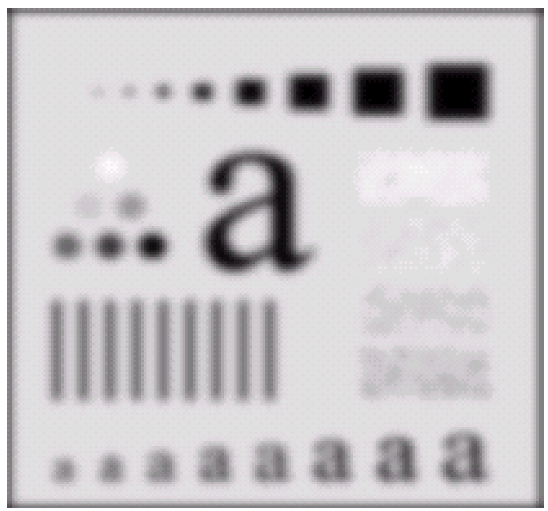
# Lowpass Filters Compared



Result of filtering with ideal low pass filter of radius 15



Result of filtering with Butterworth filter of order 2 and cutoff radius 15



Result of filtering with Gaussian filter with cutoff radius 15

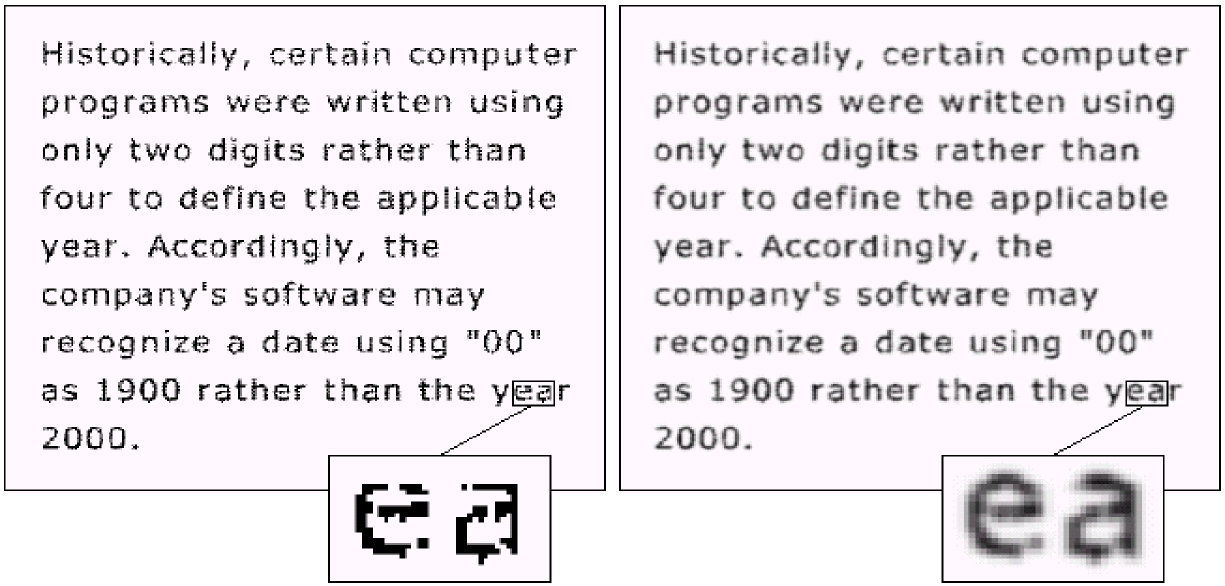
# Butterworth ←→ Gaussian

A Butterworth filter of order 1 has no ringing. Ringing generally is imperceptible in filters of order 2, but can become a significant factor in filters of higher order.

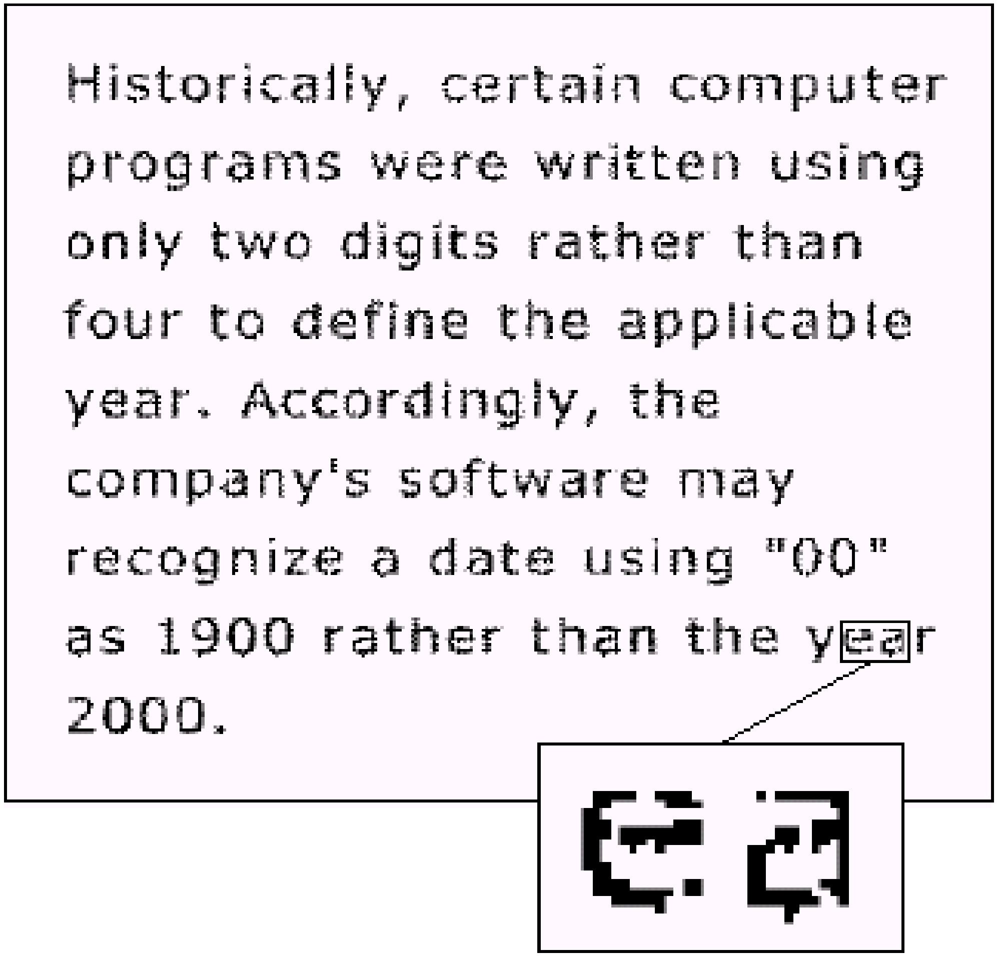
No ringing for all orders. Does not achieve as much smoothing as the Butterworth filter of order 2. This is an important characteristic in practice, especially in situations where any type of artifact is not acceptable.

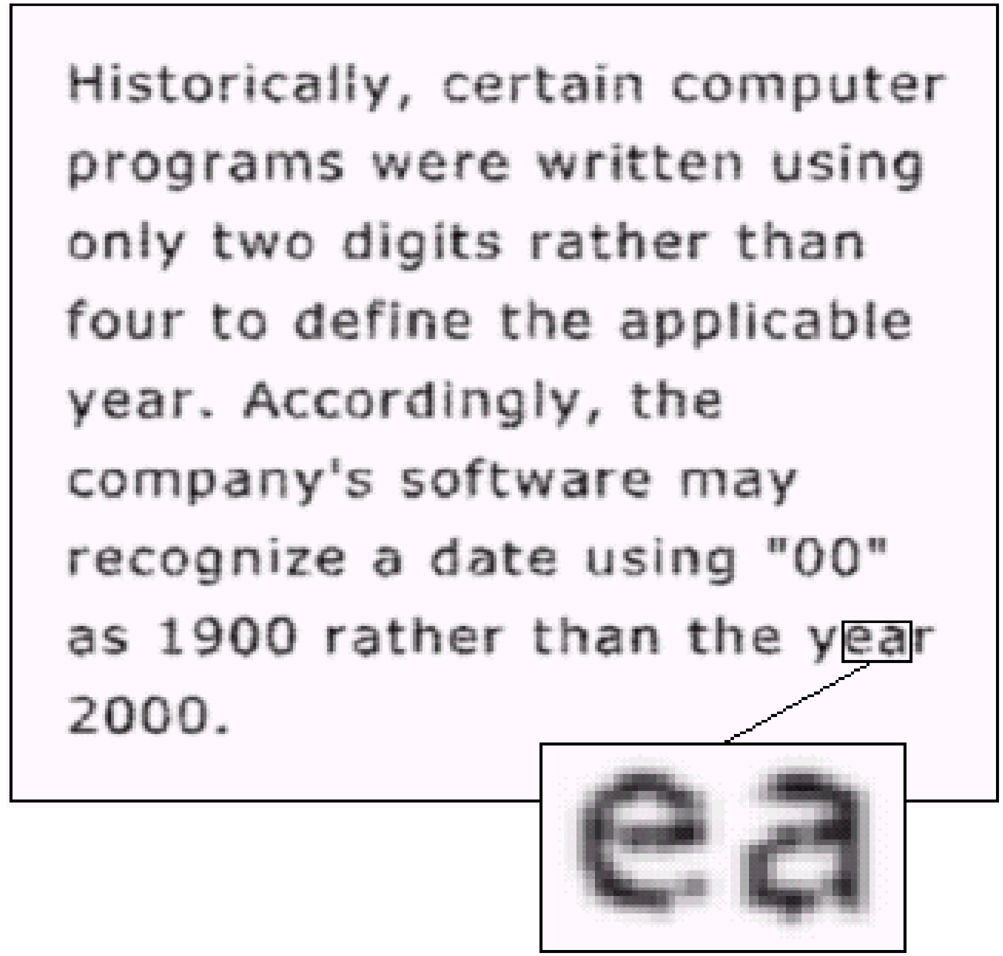
# Lowpass Filtering Examples

A low pass Gaussian filter is used to connect broken text



# Lowpass Filtering Examples



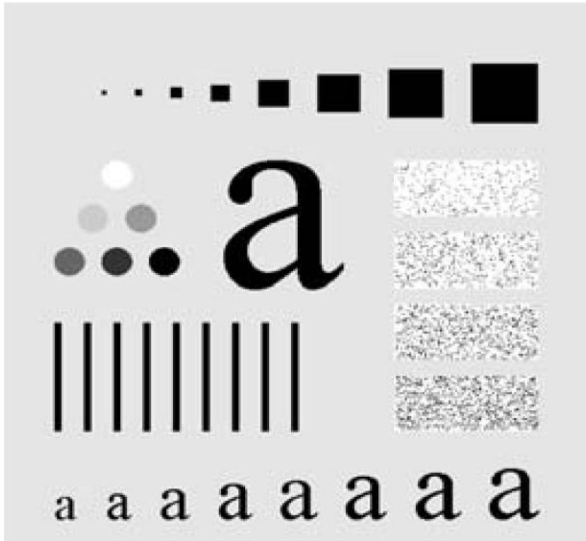
# Lowpass Filtering Examples (cont…)

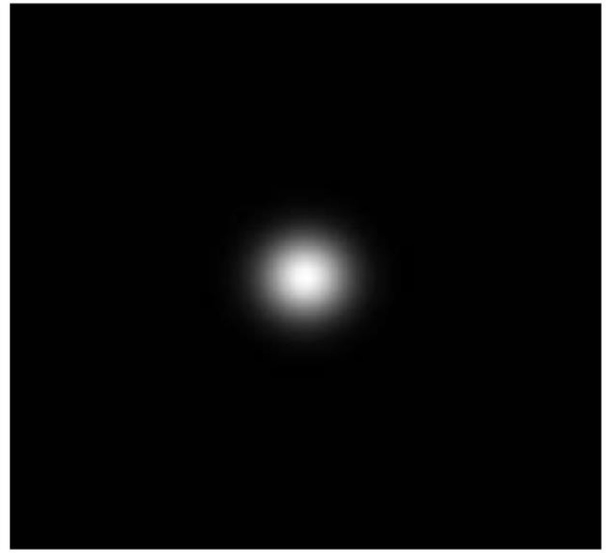
Different lowpass Gaussian filters used to remove blemishes in a photograph

# Lowpass Filtering Examples (cont…)

# Lowpass Filtering Examples (cont…)

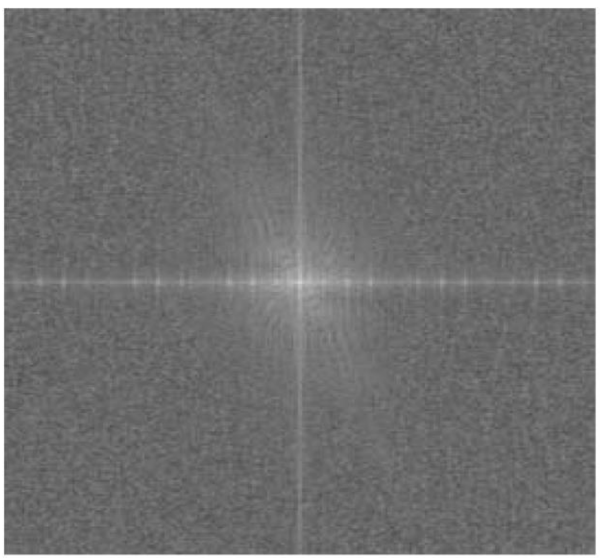
Original image





Gaussian lowpass filter

Spectrum of original image



Processed image

# Sharpening in the Frequency Domain

Edges and fine detail in images are associated with high frequency components

*High pass filters* – only pass the high frequencies, drop the low ones

High pass frequencies are precisely the reverse of low pass filters, so:

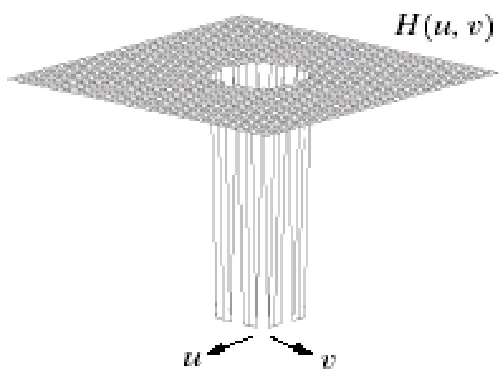
$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

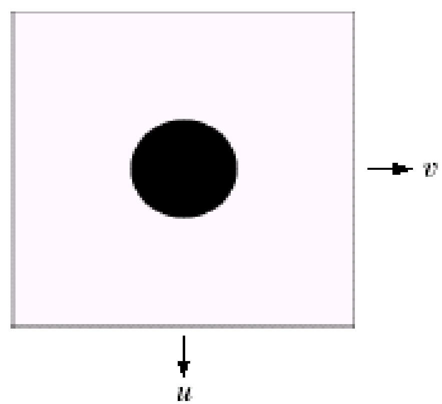
# Ideal High Pass Filters

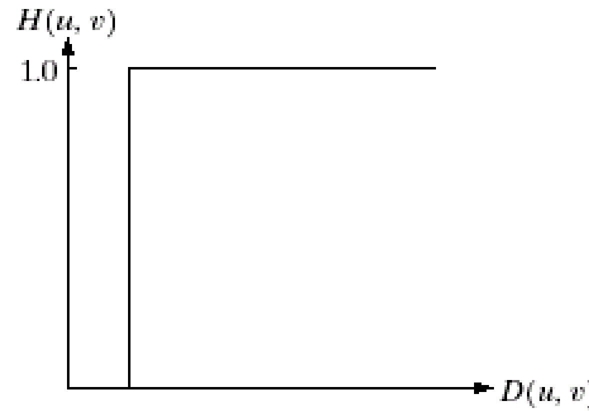
The ideal high pass filter is given as:

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

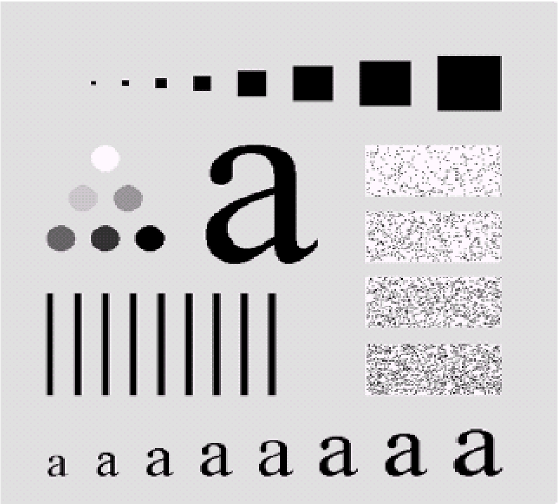
where $D_0$ is the cut off distance as before

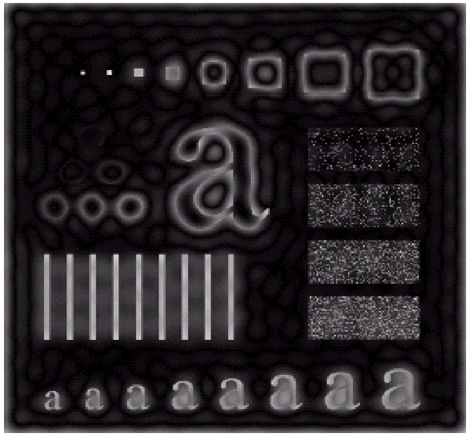


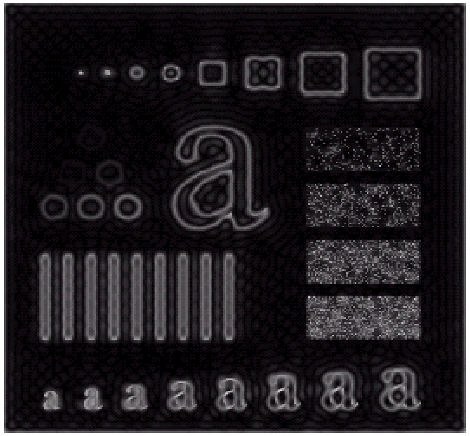



# Ideal High Pass Filters (cont…)

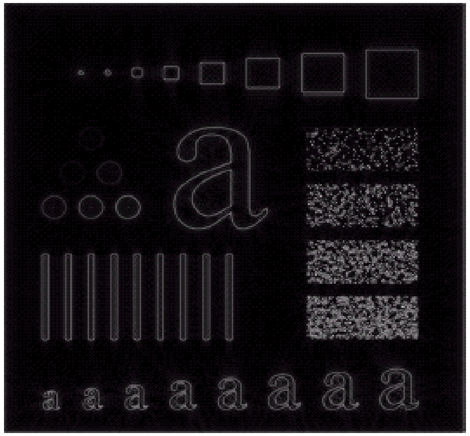




Results of ideal high pass filtering with $D_0 = 15$



Results of ideal high pass filtering with $D_0 = 30$



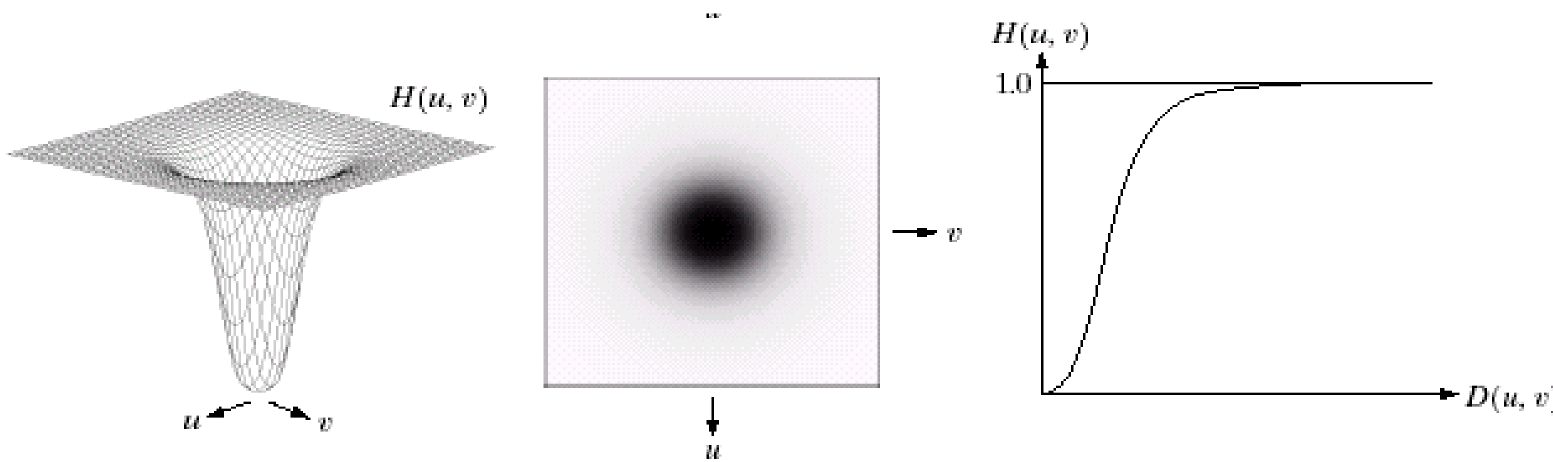
Results of ideal high pass filtering with $D_0 = 80$

# Butterworth High Pass Filters

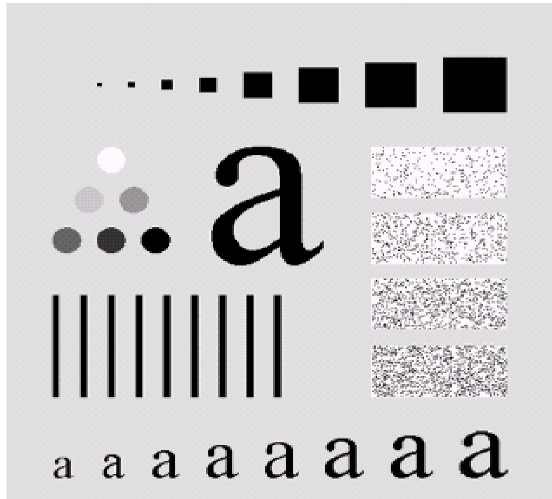
The Butterworth high pass filter is given as:

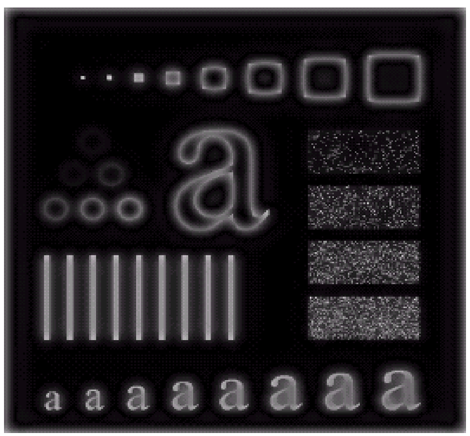
$$H(u,v) = \frac{1}{1 + [D_0 / D(u,v)]^{2n}}$$

where $n$ is the order and $D_0$ is the cut off distance as before

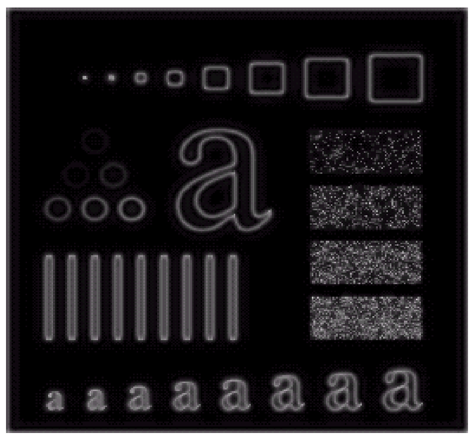


# Butterworth High Pass Filters (cont…)

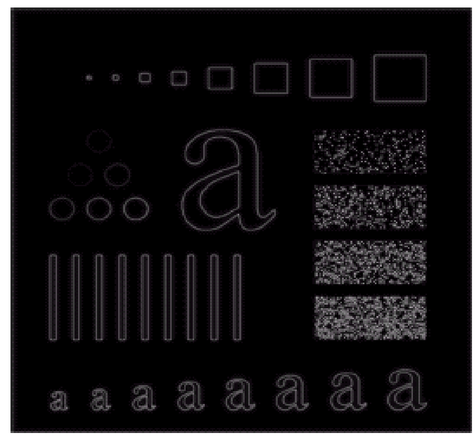




Results of Butterworth high pass filtering of order 2 with $D_0 = 15$



Results of Butterworth high pass filtering of order 2 with $D_0 = 30$



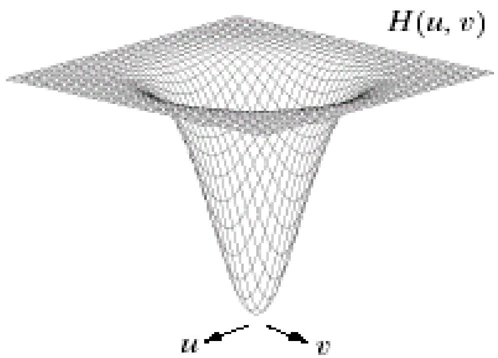
Results of Butterworth high pass filtering of order 2 with $D_0 = 80$

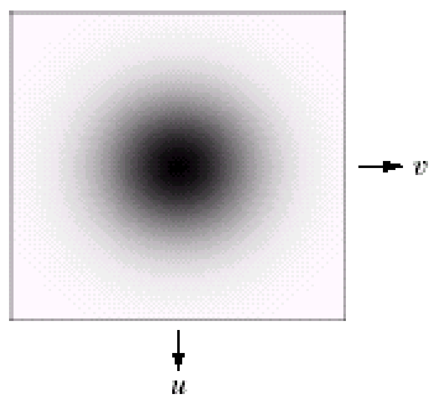
# Gaussian High Pass Filters

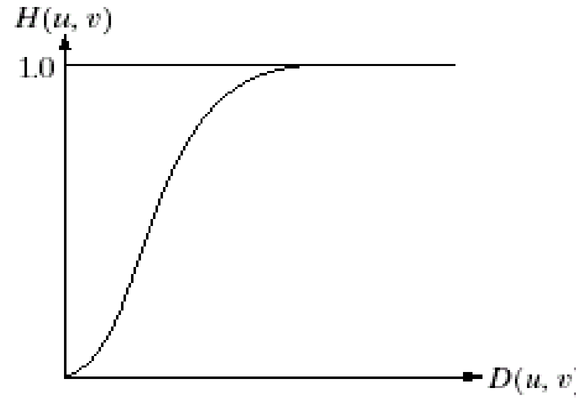
The Gaussian high pass filter is given as:

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

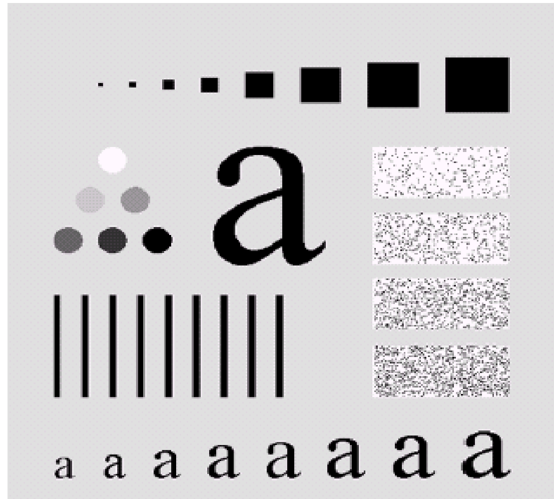
where $D_0$ is the cut off distance as before



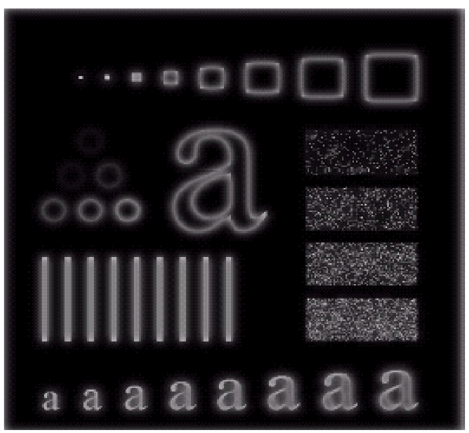



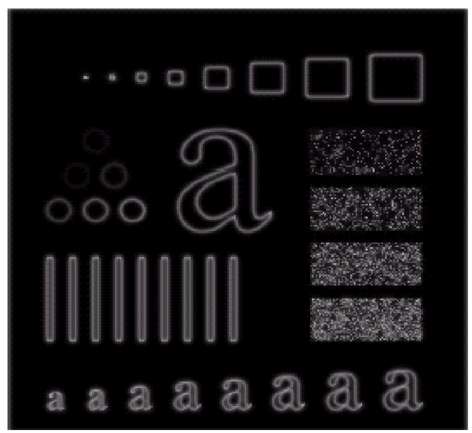



# Gaussian High Pass Filters (cont…)



Results of Gaussian high pass filtering with $D_0 = 15$

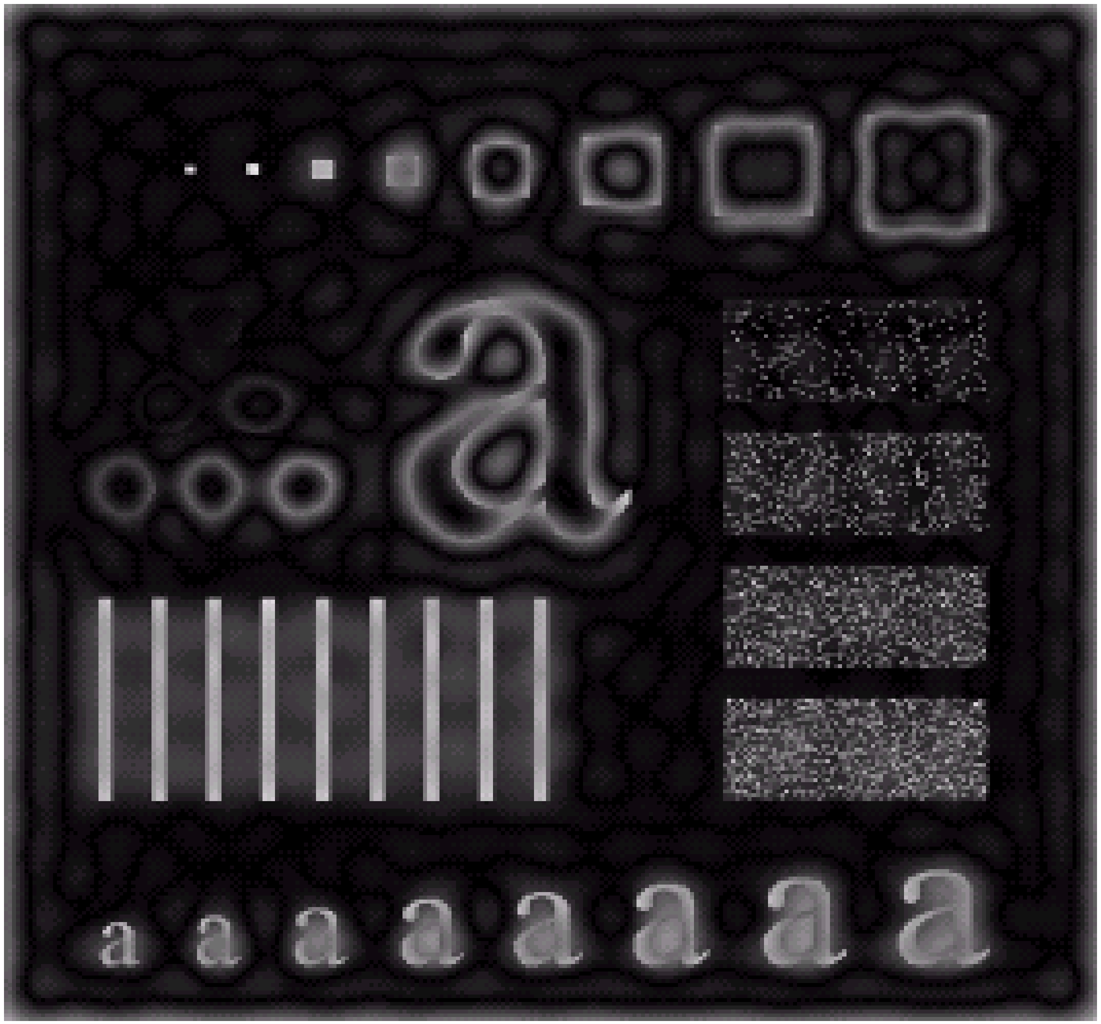




Results of Gaussian high pass filtering with $D_0 = 30$

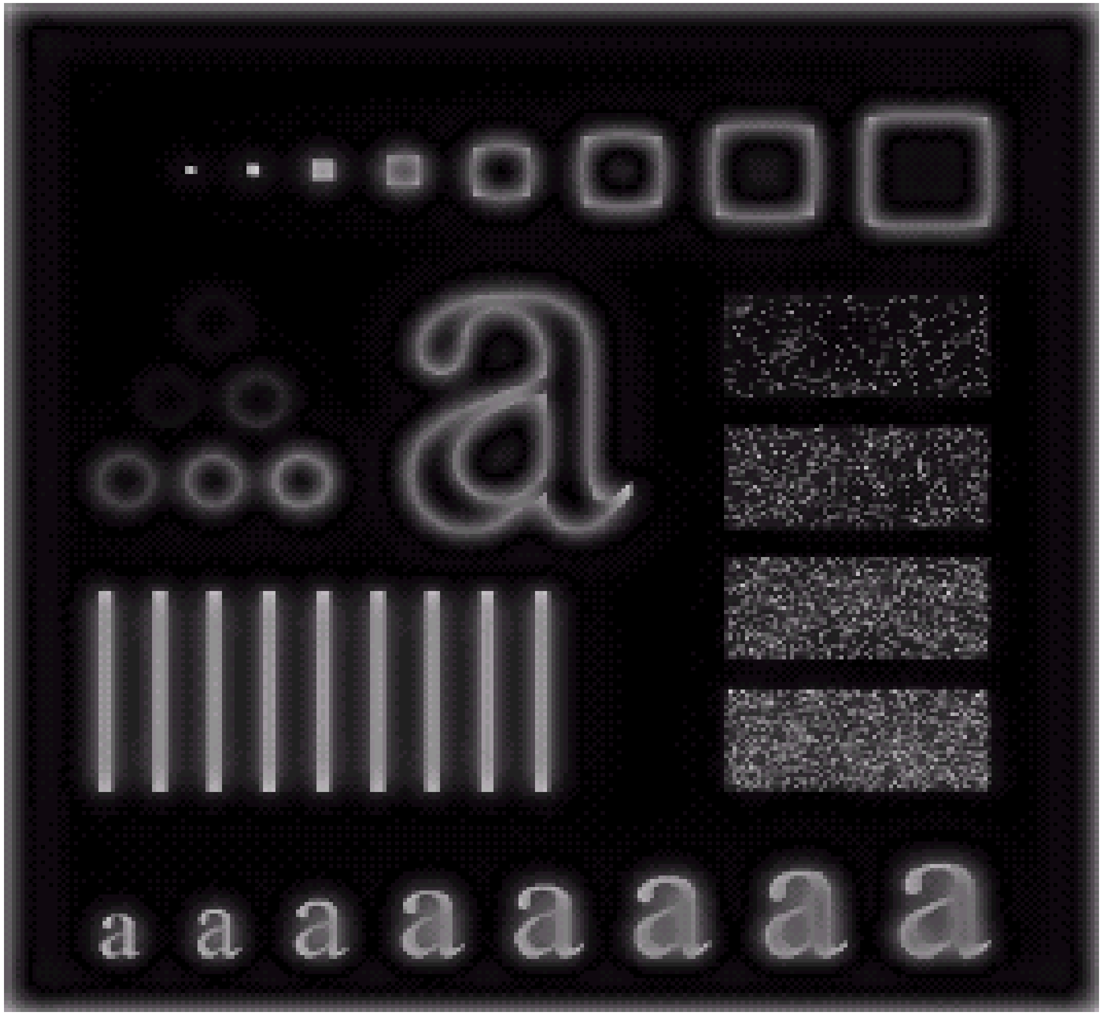
Results of Gaussian high pass filtering with $D_0 = 80$

# Highpass Filter Comparison



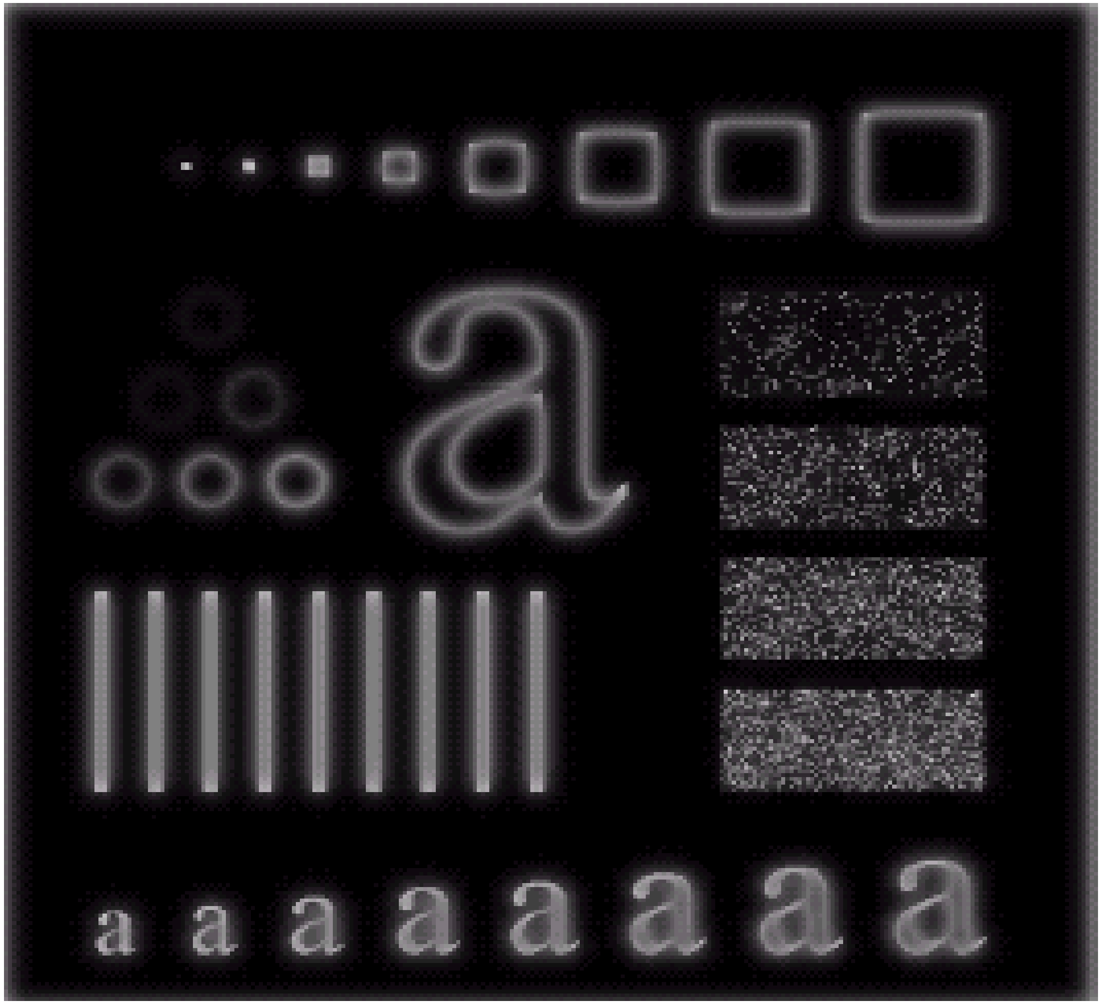
Results of ideal high pass filtering with $D_0 = 15$

# Highpass Filter Comparison



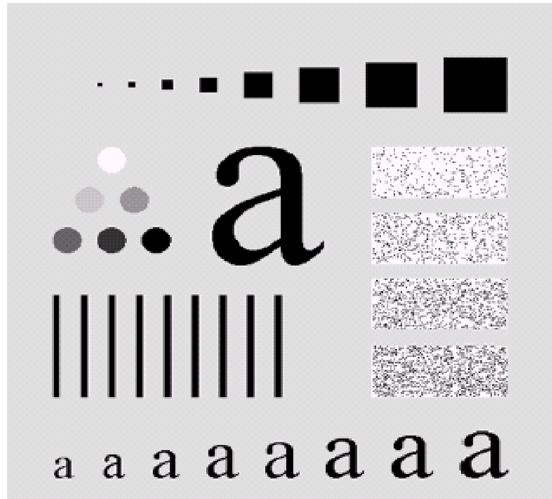
Results of Butterworth high pass filtering of order 2 with $D_0 = 15$

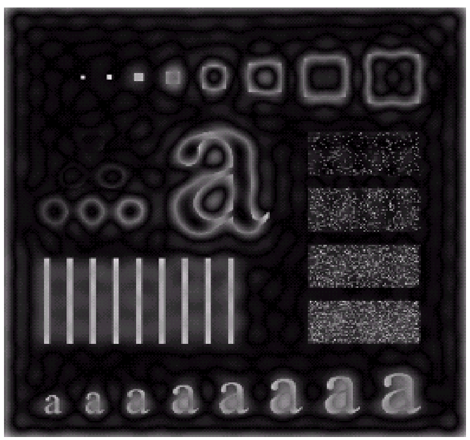
# Highpass Filter Comparison



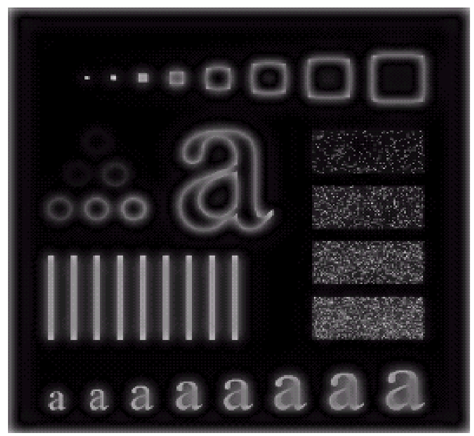
Results of Gaussian high pass filtering with $D_0 = 15$

# Highpass Filter Comparison

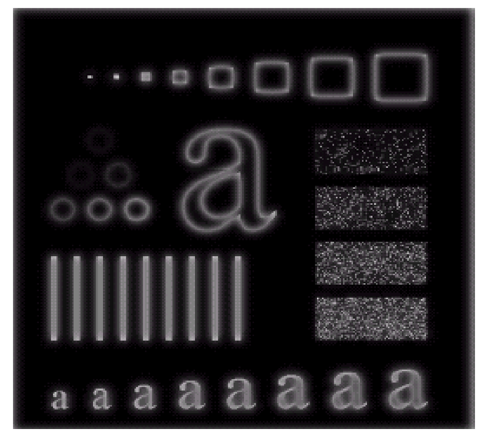




Results of ideal high pass filtering with $D_0 = 15$

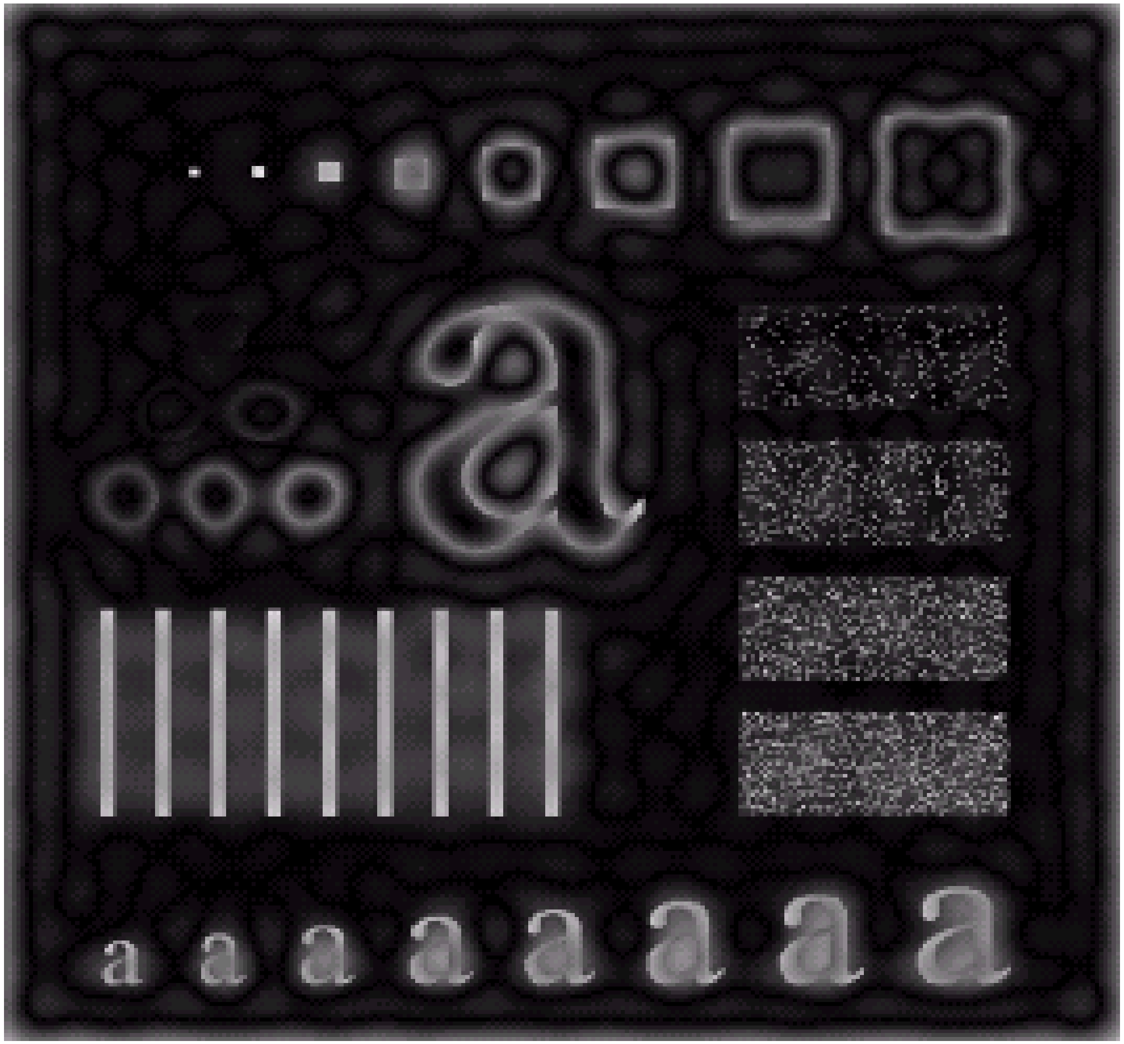


Results of Butterworth high pass filtering of order 2 with $D_0 = 15$



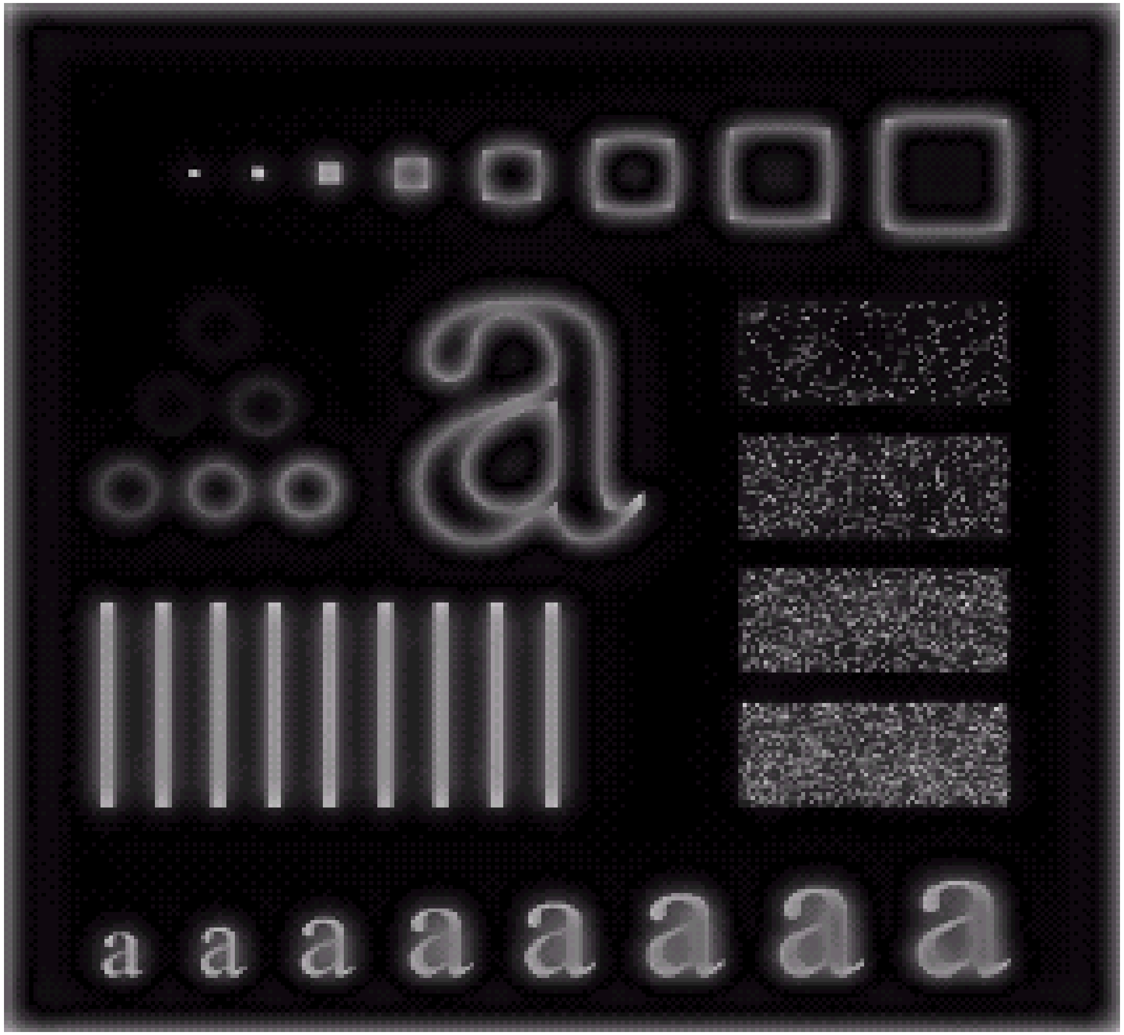
Results of Gaussian high pass filtering with $D_0 = 15$

# Highpass Filter Comparison



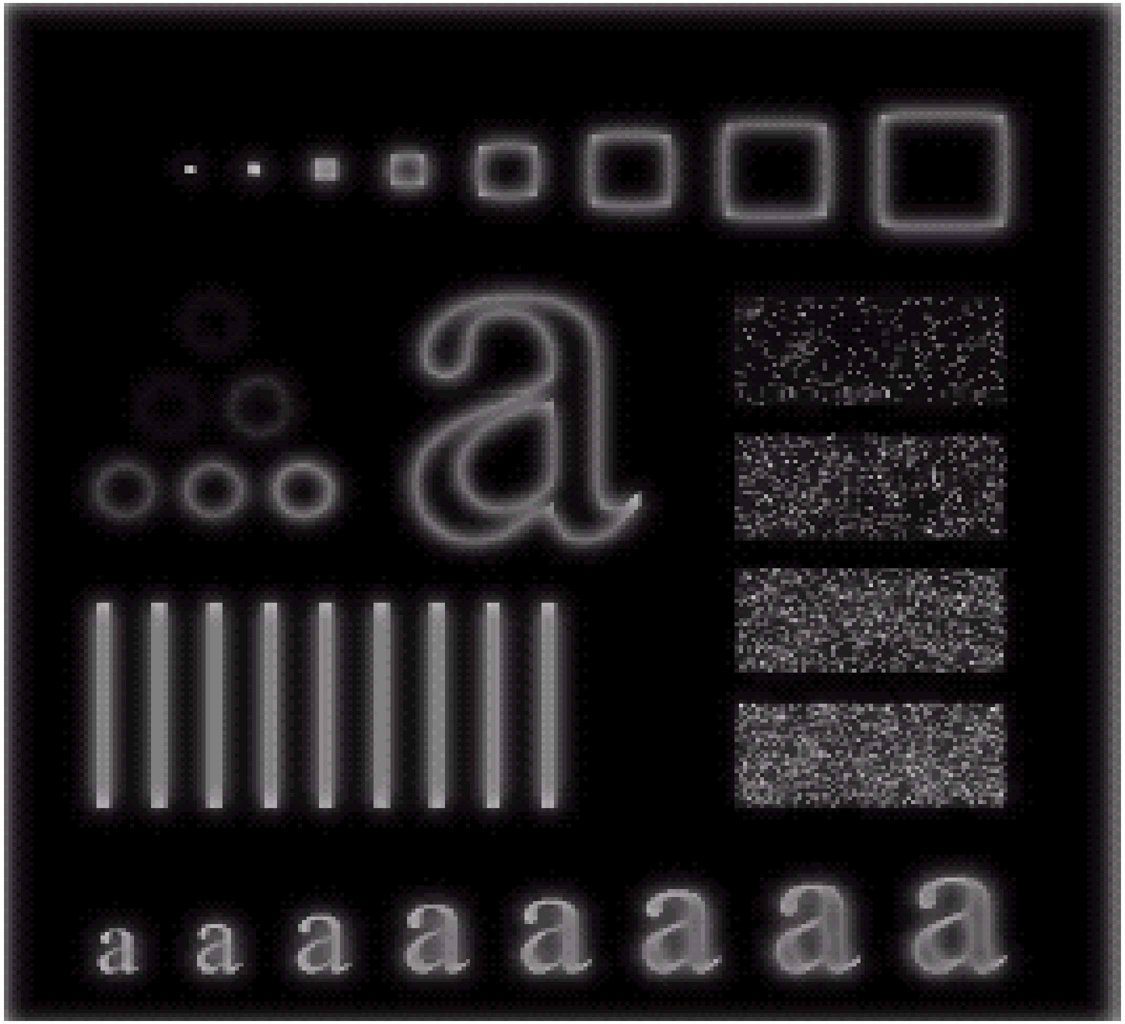
Results of ideal high pass filtering with $D_0 = 15$

# Highpass Filter Comparison



Results of Butterworth high pass filtering of order 2 with $D_0 = 15$

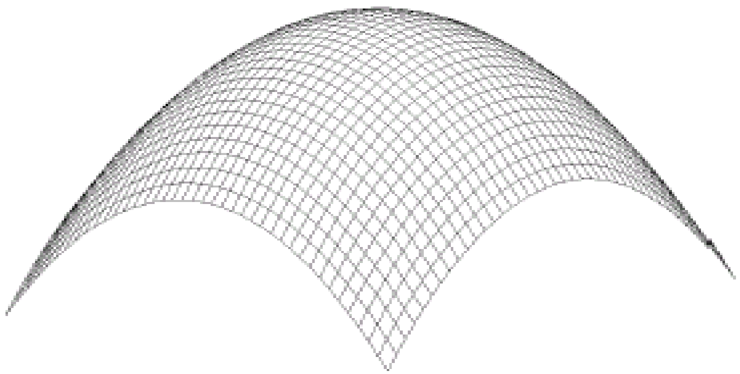
# Highpass Filter Comparison

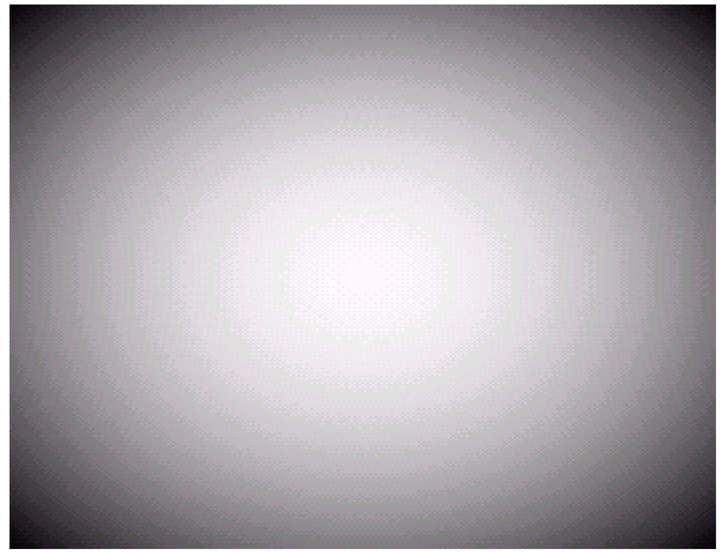


Results of Gaussian high pass filtering with $D_0 = 15$

# Laplacian In The Frequency Domain

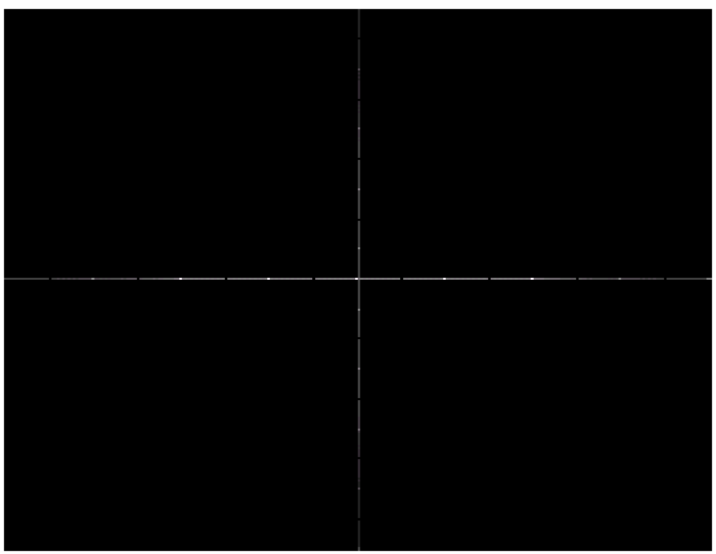
Laplacian in the frequency domain

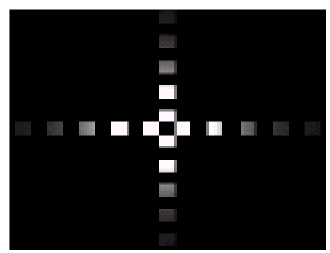


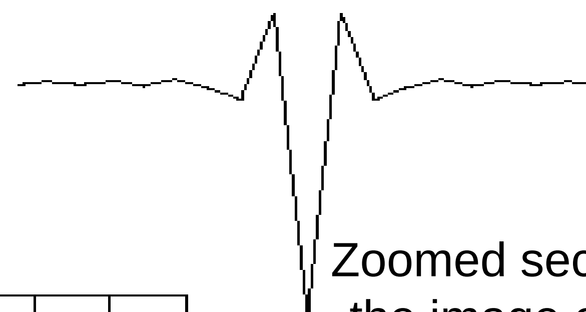


2-D image of Laplacian in the frequency domain

Inverse DFT of Laplacian in the frequency domain







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

Zoomed section of the image on the left compared to spatial filter

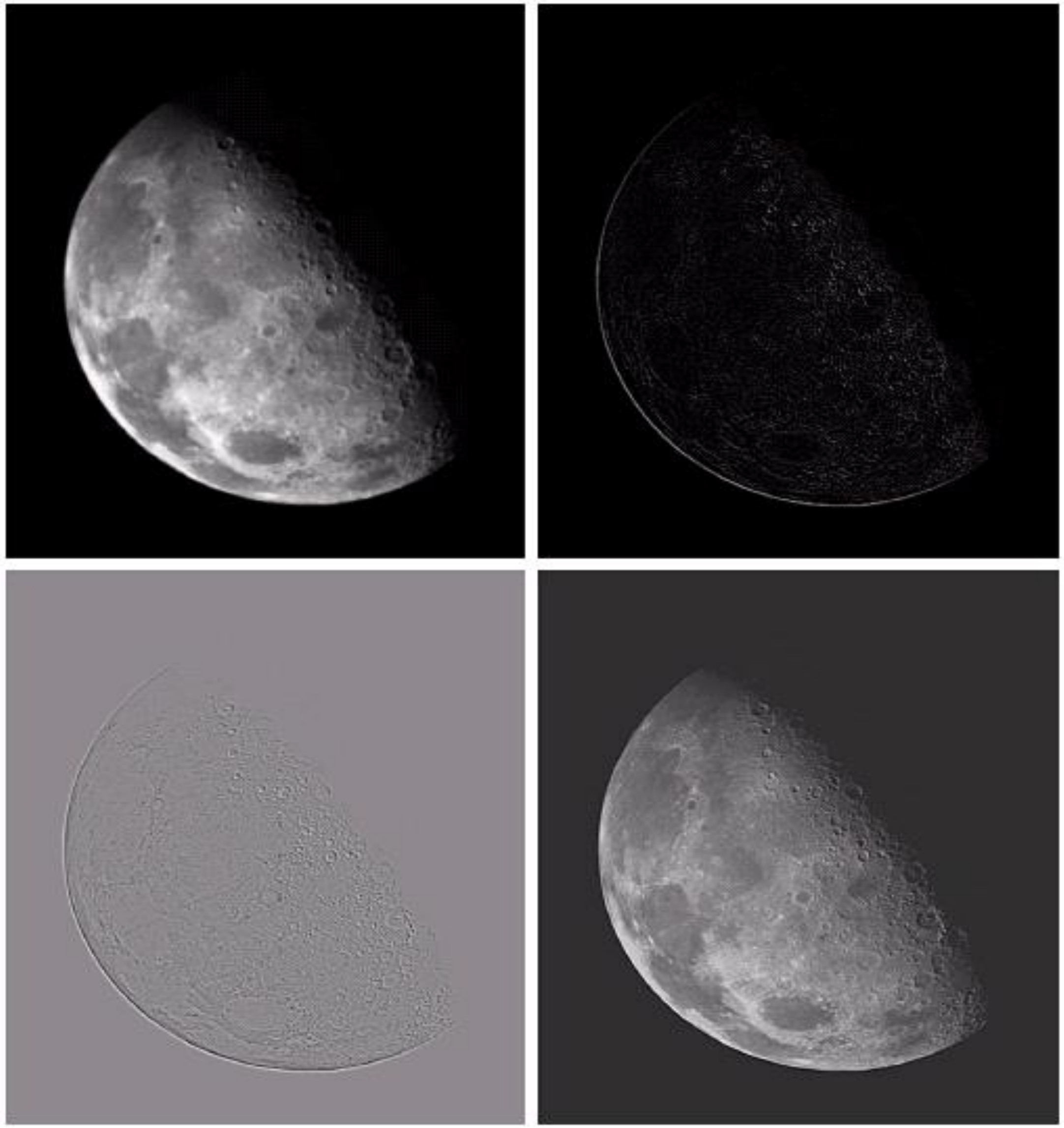
# Frequency Domain Laplacian Example

Original image

Laplacian filtered image

Laplacian image scaled

Enhanced image



# Fast Fourier Transform

The reason that Fourier based techniques have become so popular is the development of the *Fast Fourier Transform (FFT)* algorithm

Allows the Fourier transform to be carried out in a reasonable amount of time

Reduces the amount of time required to perform a Fourier transform by a factor of 100 – 600 times!

# Frequency Domain Filtering & Spatial Domain Filtering

Similar jobs can be done in the spatial and frequency domains

Filtering in the spatial domain can be easier to understand

Filtering in the frequency domain can be much faster – especially for large images

# Summary

In this lecture we examined image enhancement in the frequency domain

- The Fourier series & the Fourier transform
- Image Processing in the frequency domain
  - Image smoothing
  - Image sharpening
- Fast Fourier Transform

Next time we will begin to examine image restoration using the spatial and frequency based techniques we have been looking at

DIP

LECTURE No. ( 8)

# *Image Restoration:*

# *Noise Removal*

10/14/2019

Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

# Contents

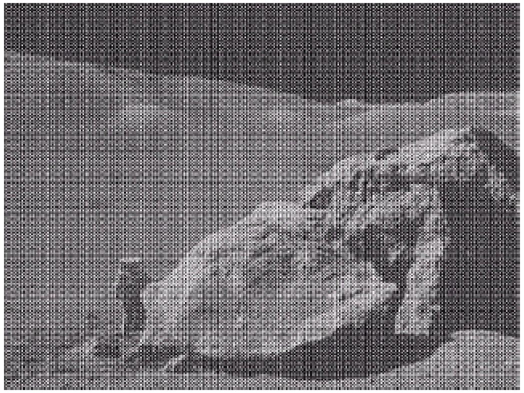
In this lecture we will look at image restoration techniques used for noise removal

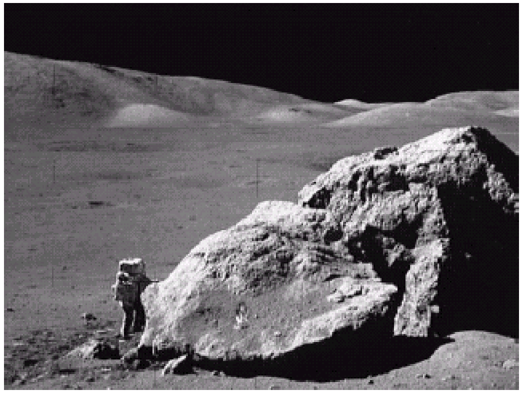
- What is image restoration?
- Noise and images
- Noise models
- Noise removal using spatial domain filtering
- Periodic noise
- Noise removal using frequency domain filtering

# What is Image Restoration?

Image restoration attempts to restore images that have been degraded

- Identify the degradation process and attempt to reverse it
- Similar to image enhancement, but more objective





# Noise and Images

The sources of noise in digital images arise during image acquisition (digitization) and transmission

– Imaging sensors can be affected by ambient conditions

– Interference can be added to an image during transmission

# Noise Model

We can consider a noisy image to be modelled as follows:

$$g(x, y) = f(x, y) + \eta(x, y)$$

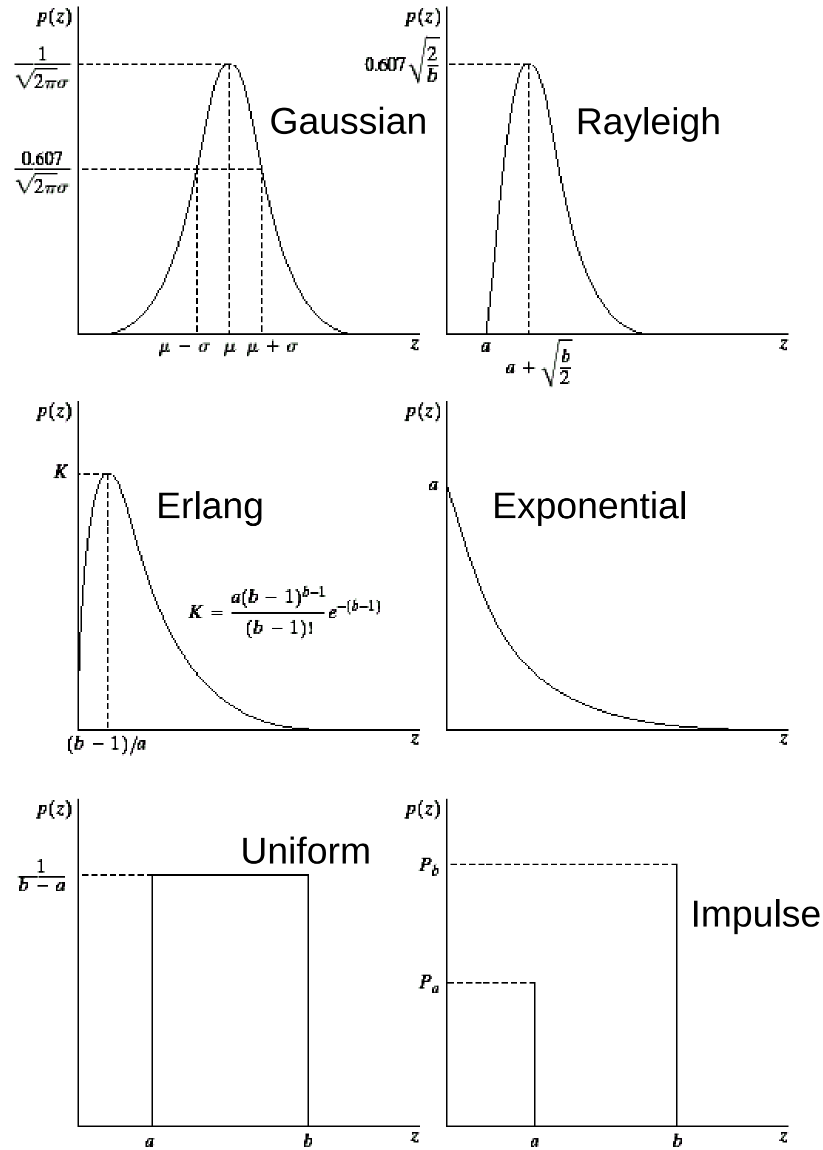
where $f(x, y)$ is the original image pixel, $\eta(x, y)$ is the noise term and $g(x, y)$ is the resulting noisy pixel

If we can estimate the model the noise in an image is based on this will help us to figure out how to restore the image

# Noise Models

There are many different models for the image noise term $\eta(x, y)$:

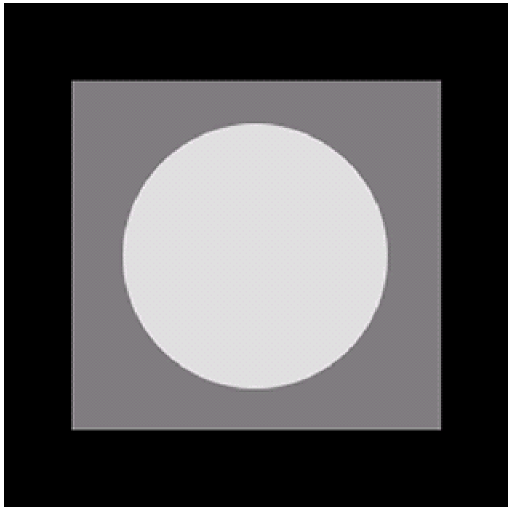
- Gaussian
  - Most common model
- Rayleigh
- Erlang
- Exponential
- Uniform
- Impulse
  - *Salt and pepper* noise



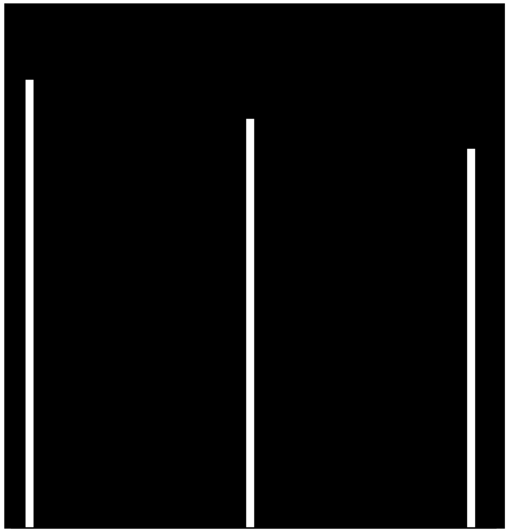
# Noise Example

The test pattern to the right is ideal for demonstrating the addition of noise

The following slides will show the result of adding noise based on various models to this image

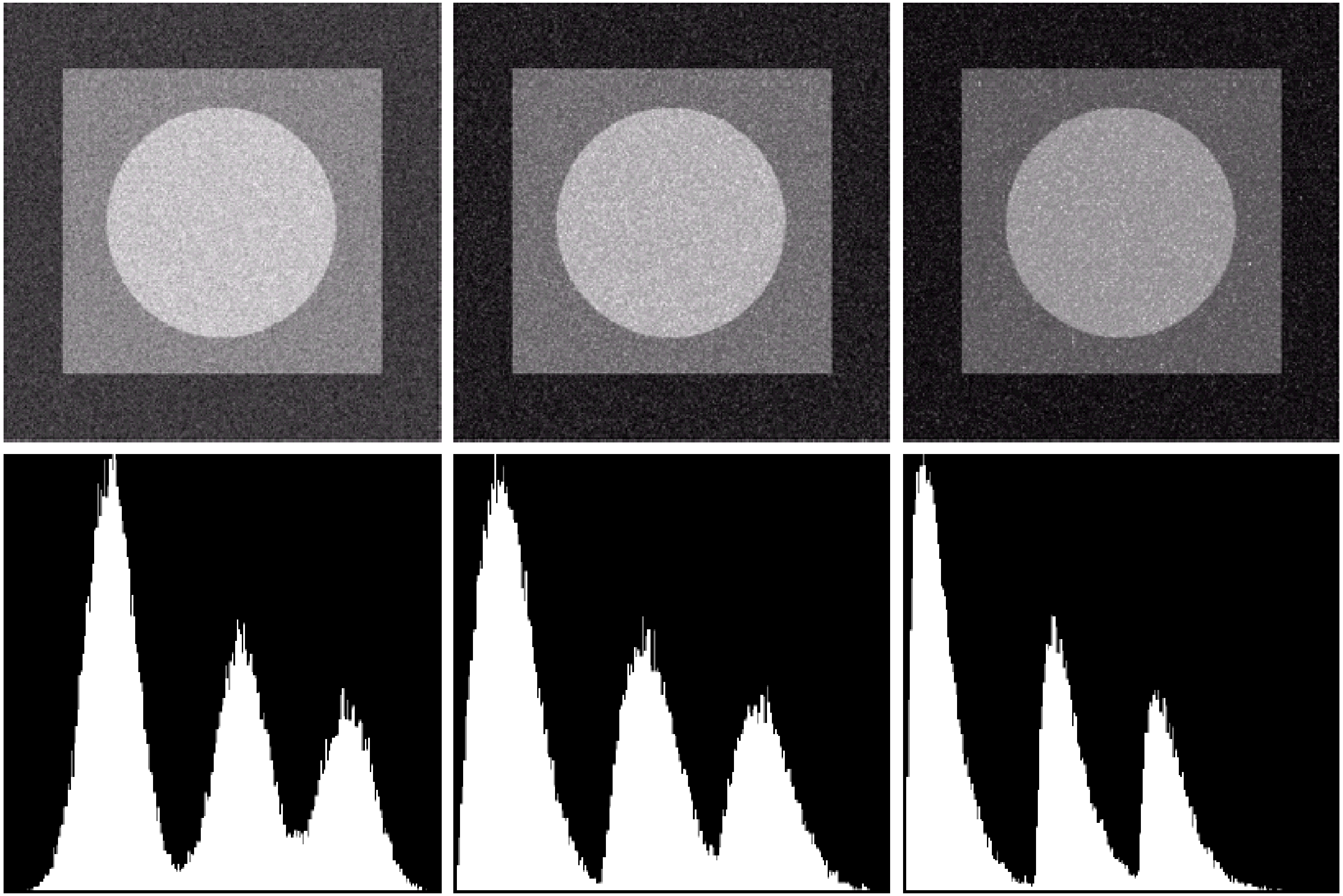


Image



Histogram

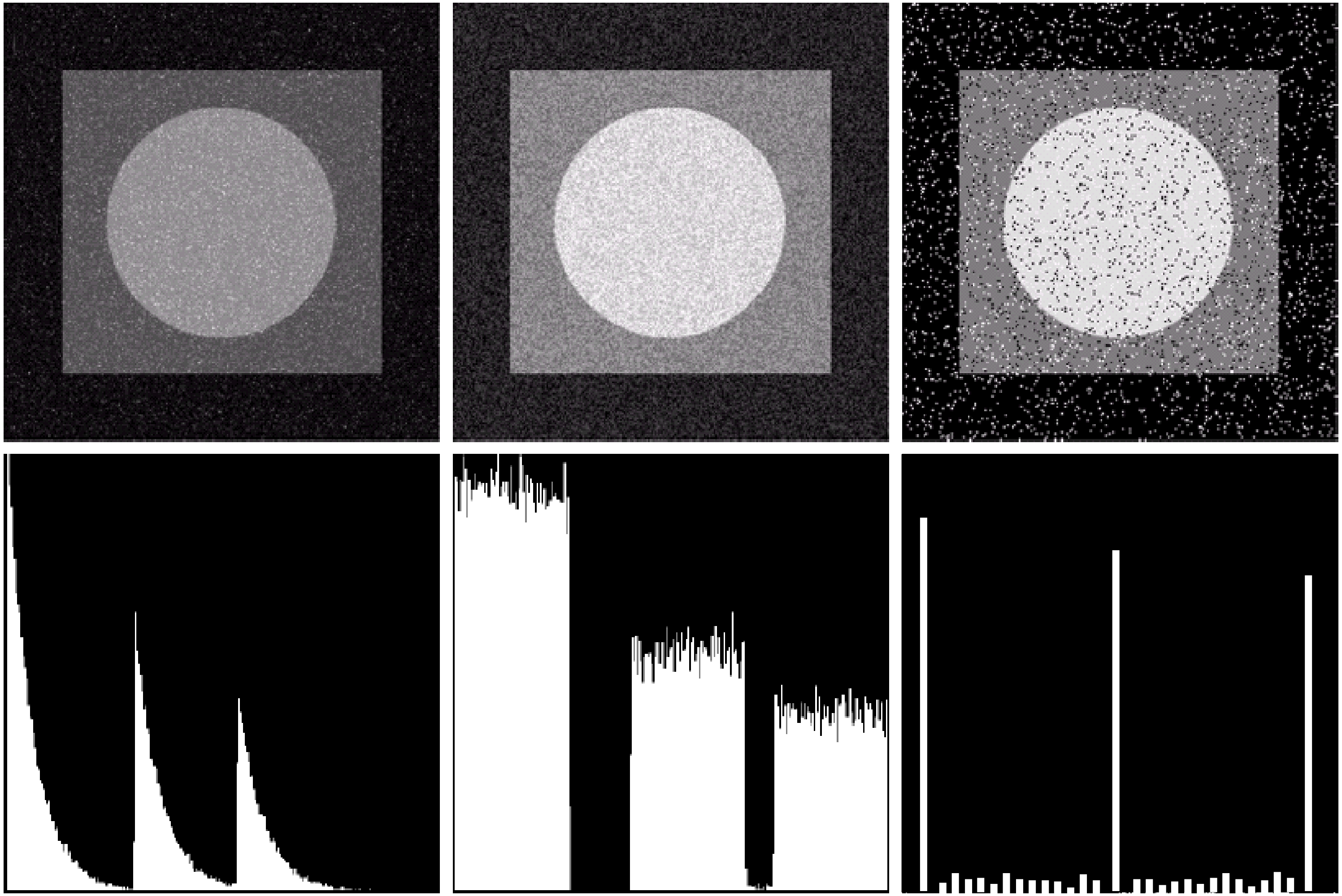
# Noise Example (cont…)



Gaussian Rayleigh Erlang

# Noise Example (cont…)





Exponential    Uniform    Impulse

# Filtering to Remove Noise

We can use spatial filters of different kinds to remove different kinds of noise

The *arithmetic mean* filter is a very simple one and is calculated as follows:

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

| | | |
|---|---|---|
| $^1/_9$ | $^1/_9$ | $^1/_9$ |
| $^1/_9$ | $^1/_9$ | $^1/_9$ |
| $^1/_9$ | $^1/_9$ | $^1/_9$ |

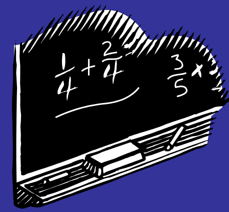
This is implemented as the simple smoothing filter

Blurs the image to remove noise

# Other Means

There are different kinds of mean filters all of which exhibit slightly different behaviour:

- Geometric Mean
- Harmonic Mean
- Contraharmonic Mean




# Other Means (cont…)

There are other variants on the mean which can give different performance

**Geometric Mean:**

$$\hat{f}(x, y) = \left[ \prod_{(s,t) \in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

Achieves similar smoothing to the arithmetic mean, but tends to lose less image detail

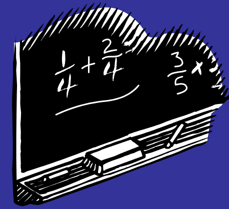
# Other Means (cont…)

**Harmonic Mean:**

$$\hat{f}(x,y) = \frac{mn}{\displaystyle\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

Works well for salt noise, but fails for pepper noise

Also does well for other kinds of noise such as Gaussian noise



# Other Means (cont…)

**Contraharmonic Mean:**

$$\hat{f}(x, y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

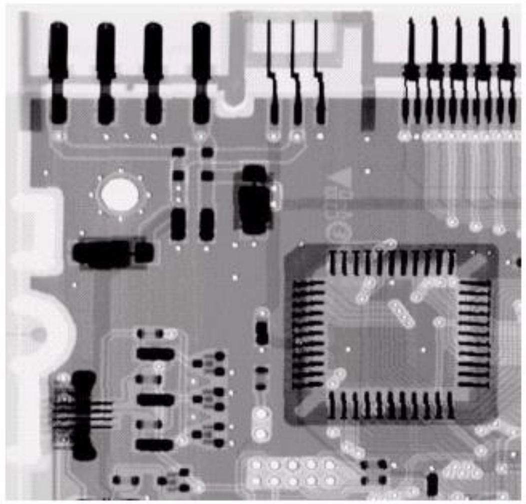
$Q$ is the *order* of the filter and adjusting its value changes the filter's behaviour

Positive values of $Q$ eliminate pepper noise

Negative values of $Q$ eliminate salt noise

# Noise Removal Examples

Original Image



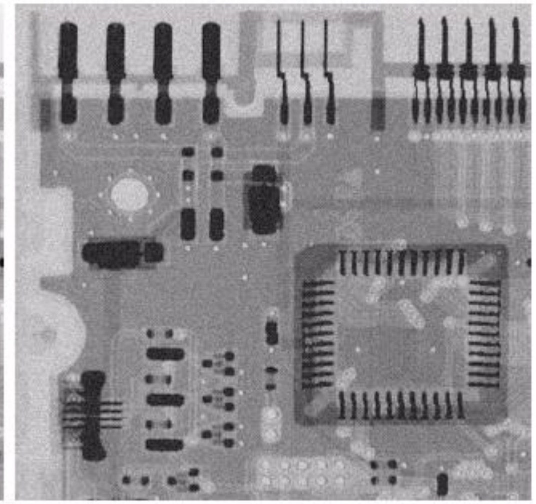
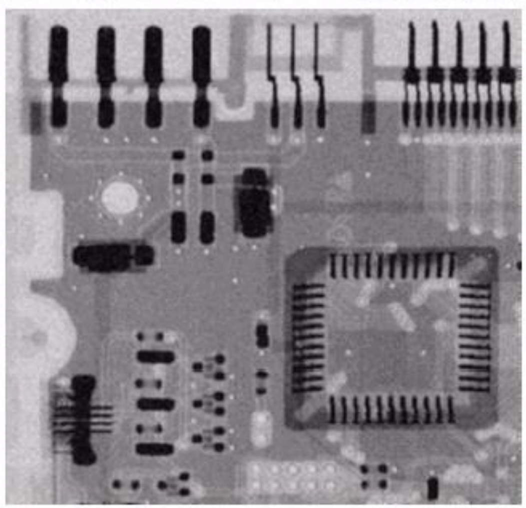
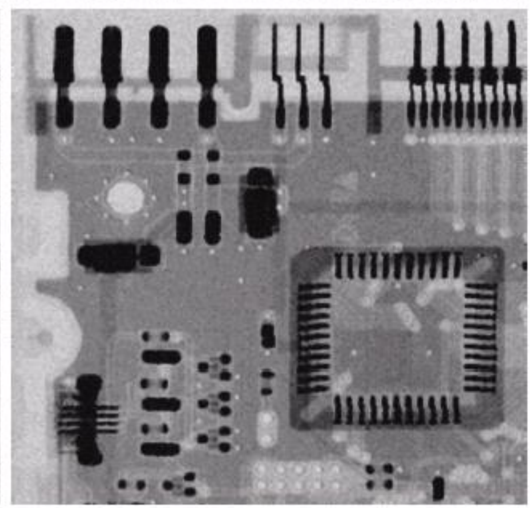


Image Corrupted By Gaussian Noise

After A 3*3 Arithmetic Mean Filter

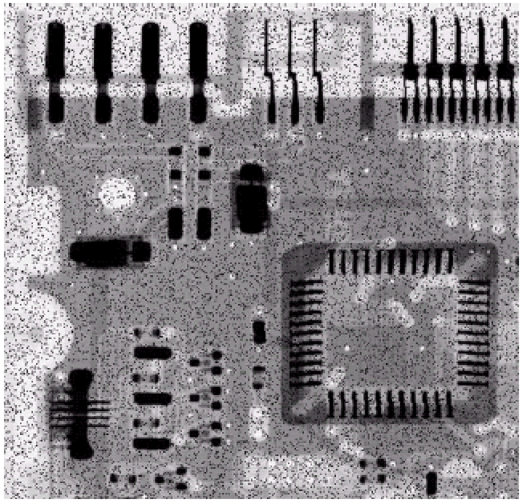




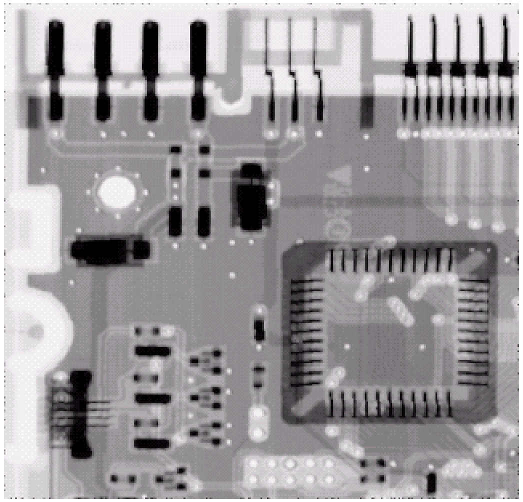
After A 3*3 Geometric Mean Filter

# Noise Removal Examples (cont…)

Image Corrupted By Pepper Noise



Result of Filtering Above With 3*3 Contraharmonic Q=1.5



# Noise Removal Examples (cont…)

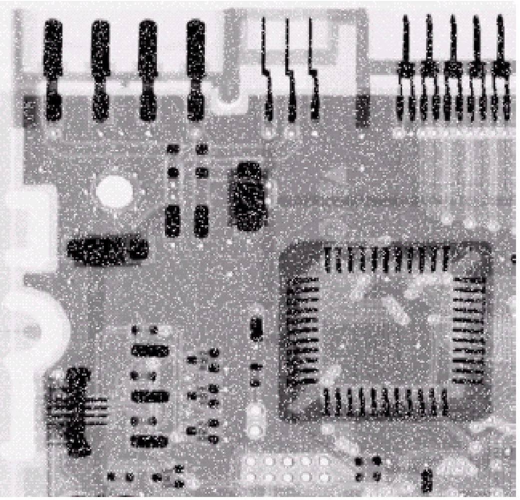
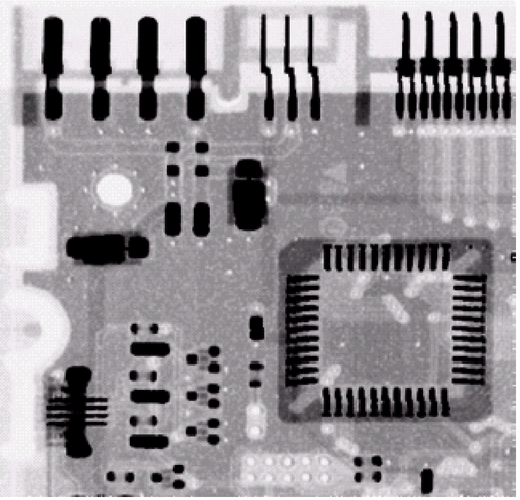


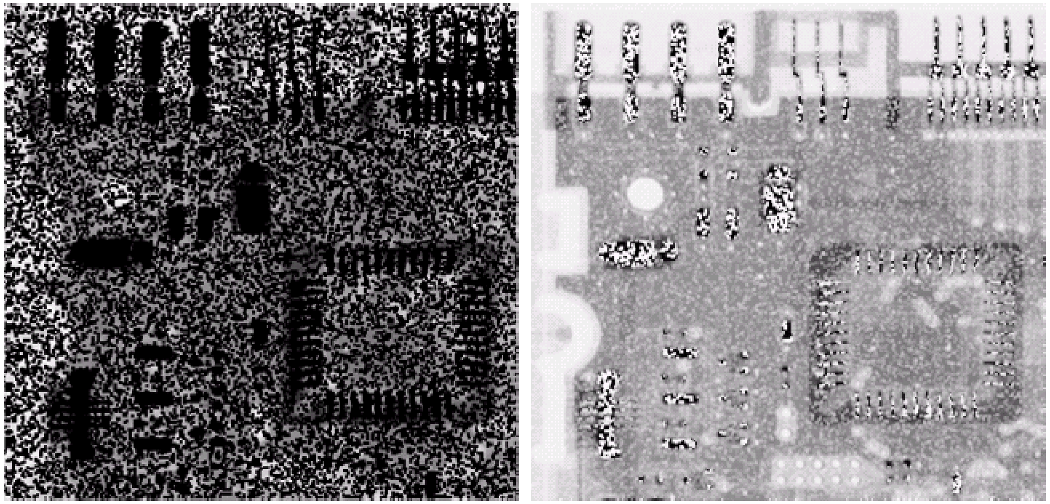
Image Corrupted By Salt Noise



Result of Filtering Above With 3*3 Contraharmonic Q=-1.5

# Contraharmonic Filter: Here Be Dragons

Choosing the wrong value for Q when using the contraharmonic filter can have drastic results



# Order Statistics Filters

Spatial filters that are based on ordering the pixel values that make up the neighbourhood operated on by the filter

Useful spatial filters include

- Median filter
- Max and min filter
- Midpoint filter
- Alpha trimmed mean filter

# Median Filter

**Median Filter:**

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{median}\{g(s,t)\}$$

Excellent at noise removal, without the smoothing effects that can occur with other smoothing filters

Particularly good when salt and pepper noise is present

# Max and Min Filter

**Max Filter:**

$$\hat{f}(x, y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$$

**Min Filter:**

$$\hat{f}(x, y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$$

Max filter is good for pepper noise and min is good for salt noise

# Midpoint Filter

**Midpoint Filter:**

$$\hat{f}(x, y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\} + \min_{(s,t)\in S_{xy}}\{g(s,t)\}\right]$$

Good for random Gaussian and uniform noise

# Alpha-Trimmed Mean Filter

**Alpha-Trimmed Mean Filter:**

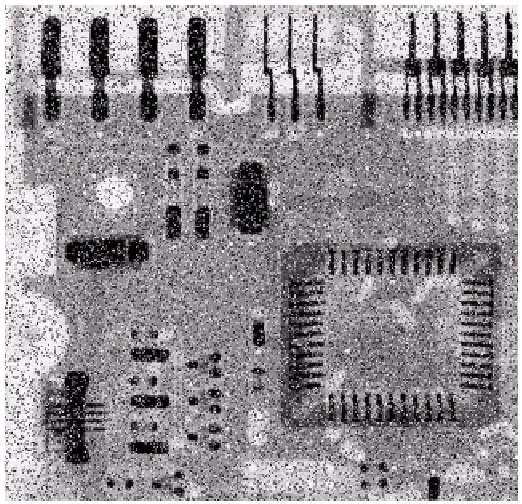
$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s,t) \in S_{xy}} g_r(s, t)$$

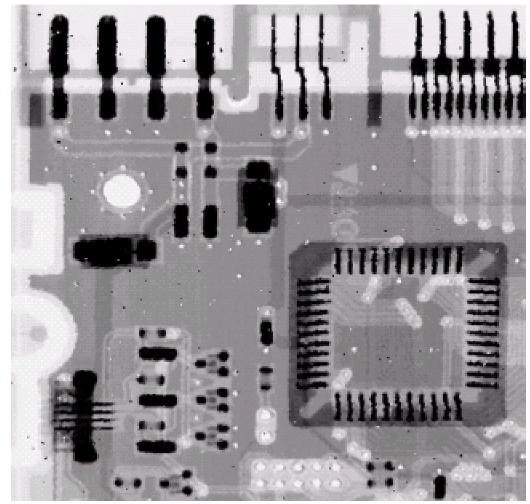
We can delete the $d/2$ lowest and $d/2$ highest grey levels

So $g_r(s, t)$ represents the remaining $mn - d$ pixels

# Noise Removal Examples

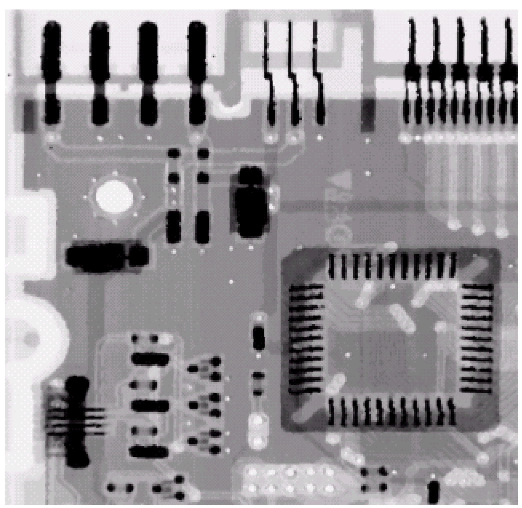
Image Corrupted By Salt And Pepper Noise

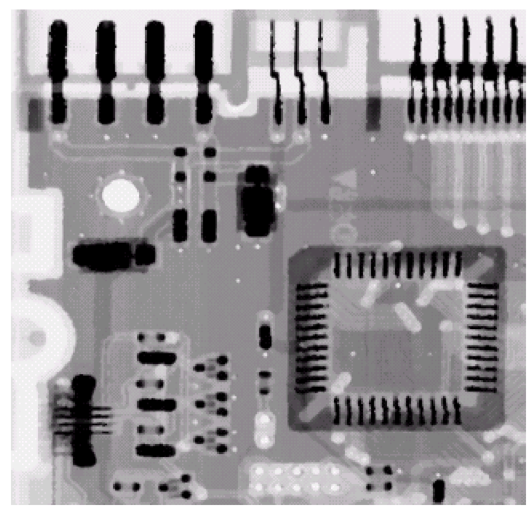




Result of 1 Pass With A 3*3 Median Filter

Result of 2 Passes With A 3*3 Median Filter

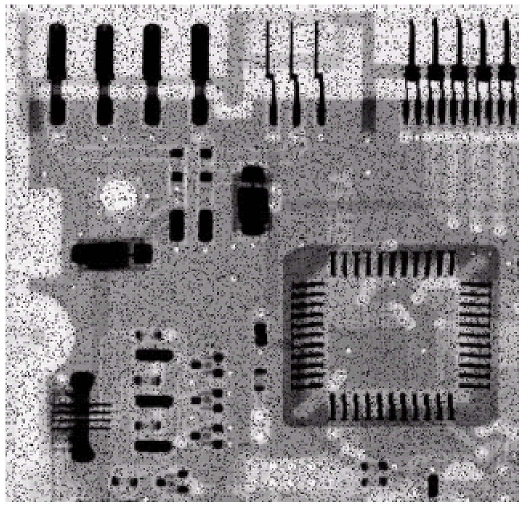




Result of 3 Passes With A 3*3 Median Filter

# Noise Removal Examples (cont…)

Image Corrupted By Pepper Noise



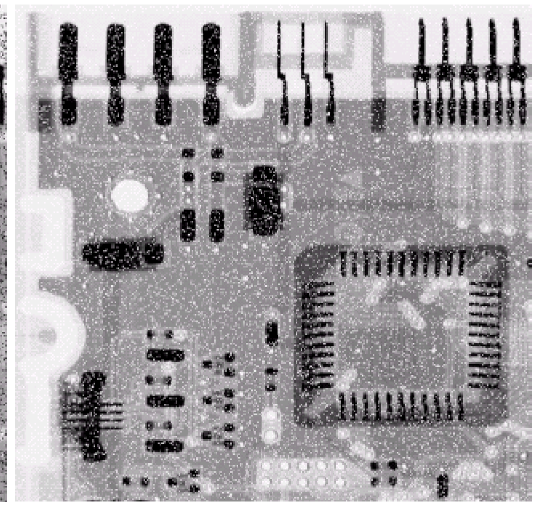
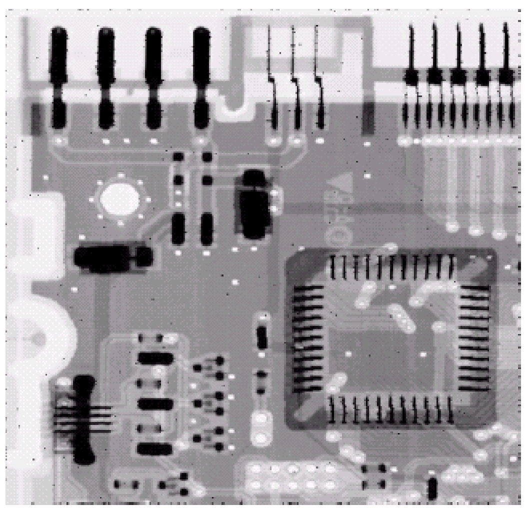
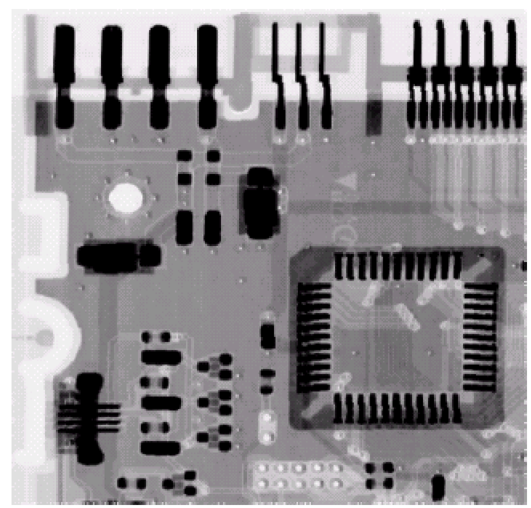


Image Corrupted By Salt Noise

Result Of Filtering Above With A 3*3 Max Filter





Result Of Filtering Above With A 3*3 Min Filter

# Noise Removal Examples (cont…)

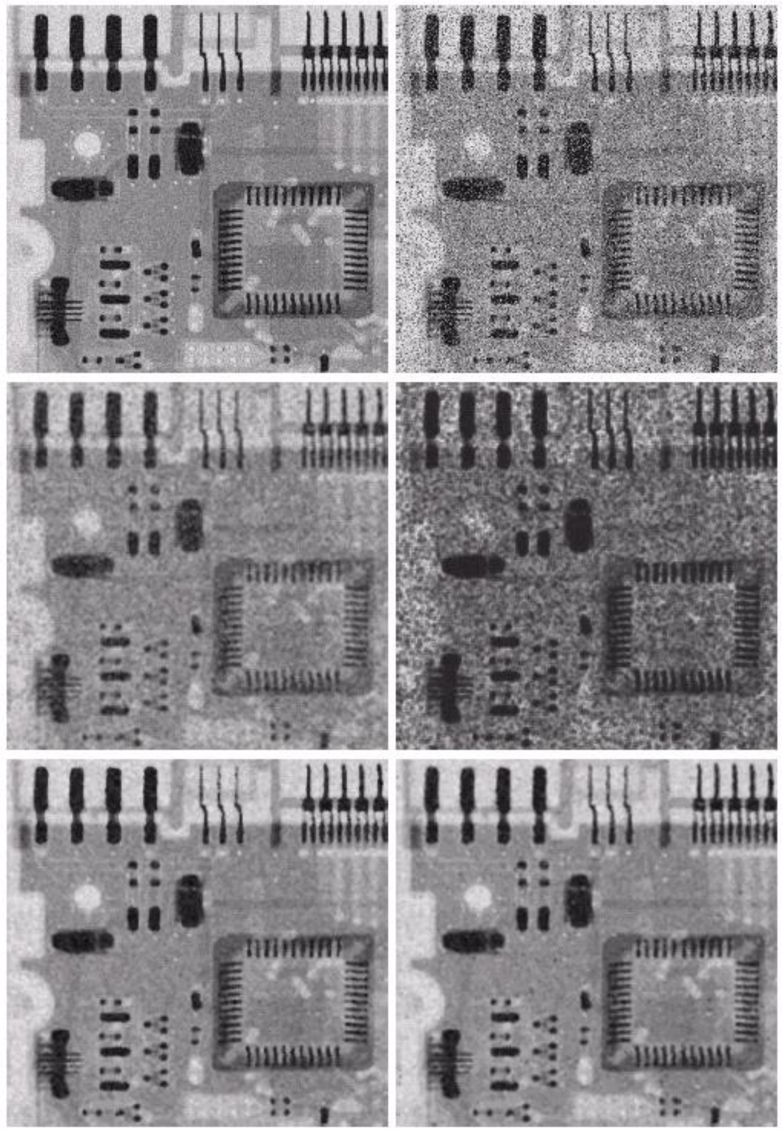


Image Corrupted By Uniform Noise

Image Further Corrupted By Salt and Pepper Noise

Filtered By 5*5 Arithmetic Mean Filter

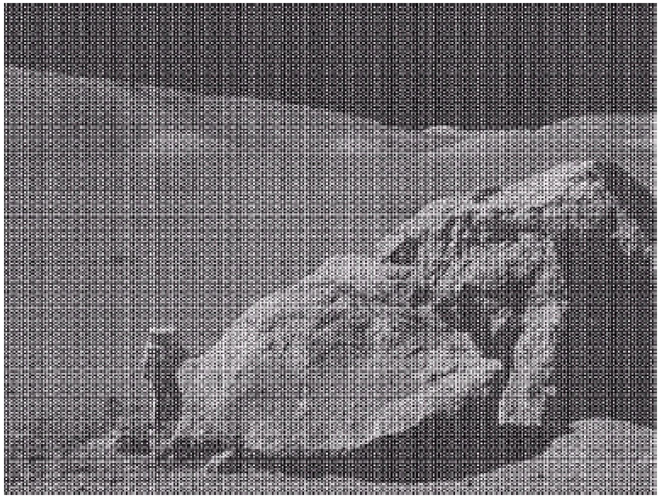
Filtered By 5*5 Geometric Mean Filter

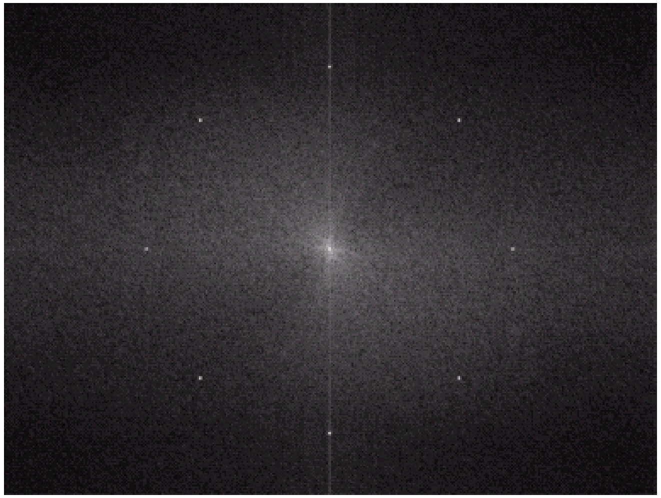
Filtered By 5*5 Median Filter

Filtered By 5*5 Alpha-Trimmed Mean Filter

# Periodic Noise

- Typically arises due to electrical or electromagnetic interference
- Gives rise to regular noise patterns in an image
- Frequency domain techniques in the Fourier domain are most effective at removing periodic noise





# Band Reject Filters

Removing periodic noise form an image involves removing a particular range of frequencies from that image

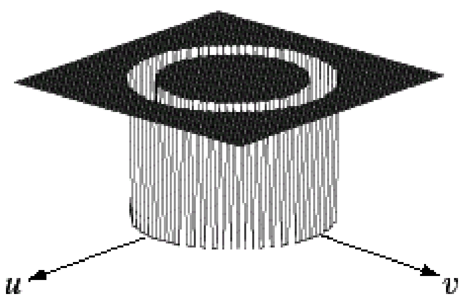
*Band reject* filters can be used for this purpose

An ideal band reject filter is given as follows:

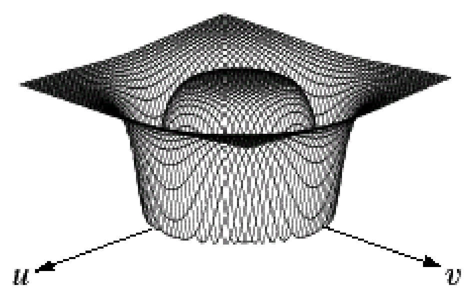
$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) < D_0 - \dfrac{W}{2} \\ 0 & \text{if } D_0 - \dfrac{W}{2} \le D(u,v) \le D_0 + \dfrac{W}{2} \\ 1 & \text{if } D(u,v) > D_0 + \dfrac{W}{2} \end{cases}$$

# Band Reject Filters (cont…)

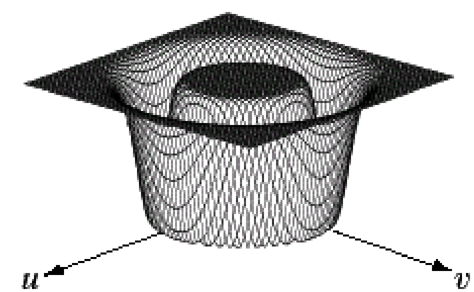
The ideal band reject filter is shown below, along with Butterworth and Gaussian versions of the filter




Ideal Band Reject Filter




Butterworth Band Reject Filter (of order 1)

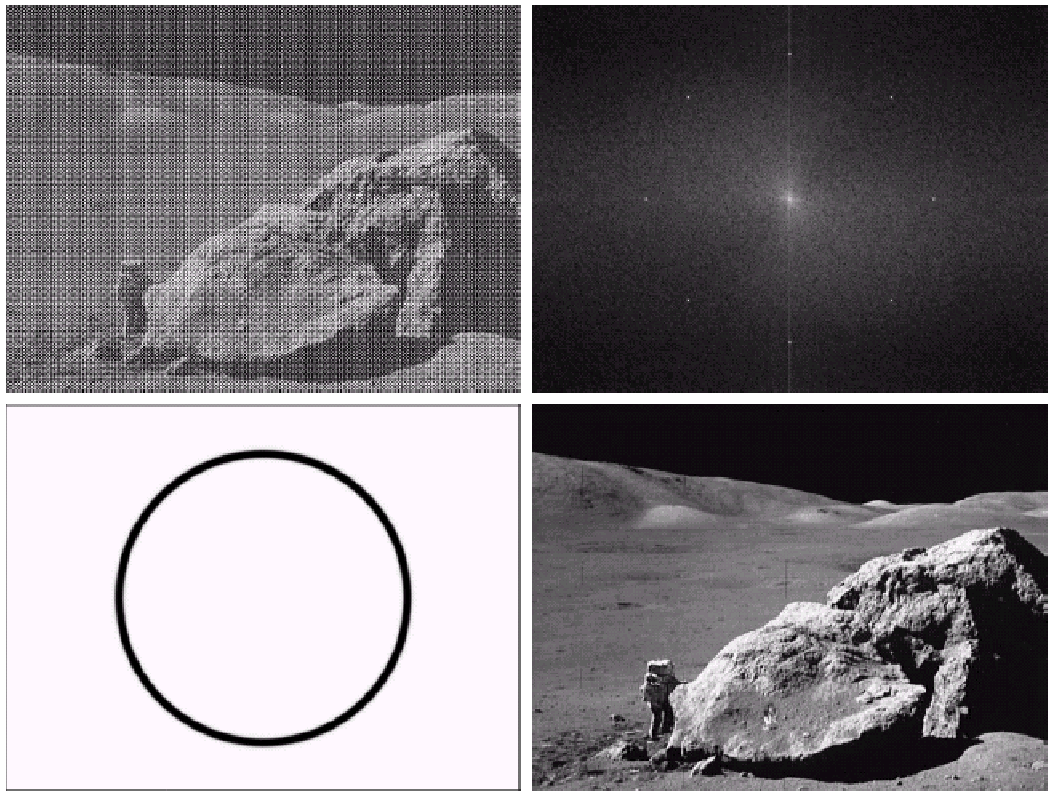



Gaussian Band Reject Filter

# Band Reject Filter Example

Image corrupted by sinusoidal noise

Fourier spectrum of corrupted image



Butterworth band reject filter

Filtered image

# Summary

In this lecture we will look at image restoration for noise removal

Restoration is slightly more objective than enhancement

Spatial domain techniques are particularly useful for removing random noise

Frequency domain techniques are particularly useful for removing periodic noise

# Adaptive Filters

The filters discussed so far are applied to an entire image without any regard for how image characteristics vary from one point to another

The behaviour of **adaptive filters** changes depending on the characteristics of the image inside the filter region

We will take a look at the **adaptive median filter**

# Adaptive Median Filtering

The median filter performs relatively well on impulse noise as long as the spatial density of the impulse noise is not large

The adaptive median filter can handle much more spatially dense impulse noise, and also performs some smoothing for non-impulse noise

The key insight in the adaptive median filter is that the filter size changes depending on the characteristics of the image

# Adaptive Median Filtering (cont…)

Remember that filtering looks at each original pixel image in turn and generates a new filtered pixel

First examine the following notation:

- $z_{min}$ = minimum grey level in $S_{xy}$
- $z_{max}$ = maximum grey level in $S_{xy}$
- $z_{med}$ = median of grey levels in $S_{xy}$
- $z_{xy}$ = grey level at coordinates $(x, y)$
- $S_{max}$ =maximum allowed size of $S_{xy}$

# Adaptive Median Filtering (cont…)

Level A: $A1 = z_{med} - z_{min}$

$A2 = z_{med} - z_{max}$

If $A1 > 0$ and $A2 < 0$, Go to level B

Else increase the window size

If window size $\leq$ repeat $S_{max}$ level A

Else output $z_{med}$

Level B: $B1 = z_{xy} - z_{min}$

$B2 = z_{xy} - z_{max}$

If $B1 > 0$ and $B2 < 0$, output $z_{xy}$

Else output $z_{med}$

# Adaptive Median Filtering (cont…)

The key to understanding the algorithm is to remember that the adaptive median filter has three purposes:

– Remove impulse noise

– Provide smoothing of other noise

– Reduce distortion

# Adaptive Filtering Example

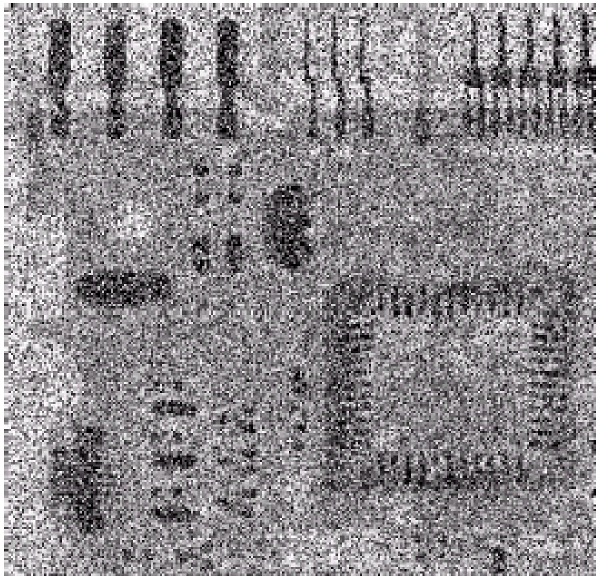
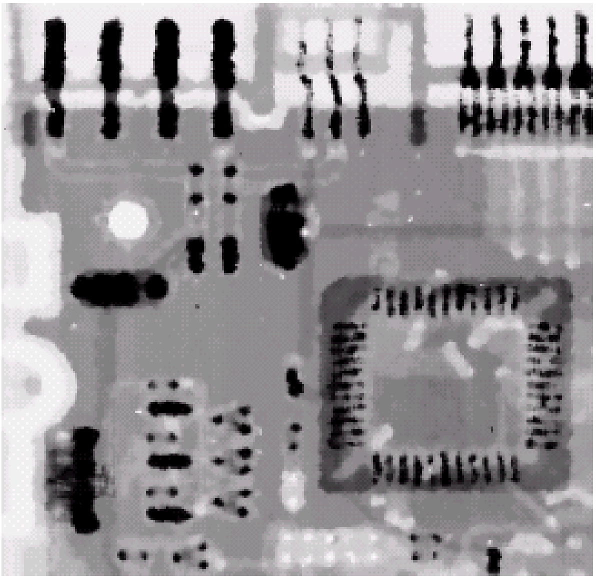
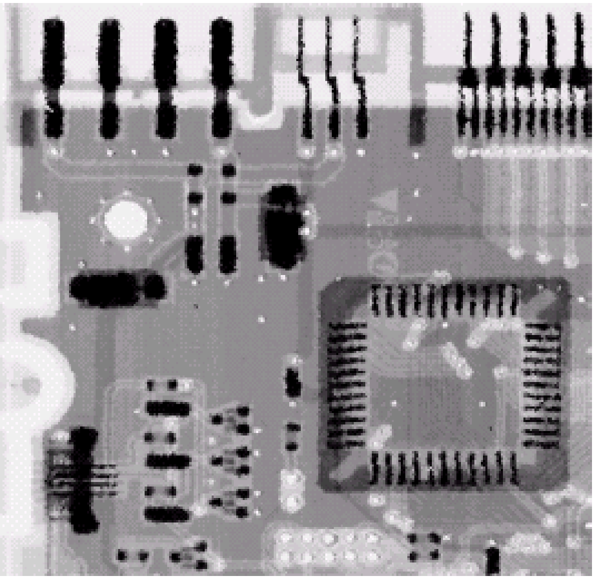


Image corrupted by salt and pepper noise with probabilities $P_a = P_b = 0.25$



Result of filtering with a 7 * 7 median filter



Result of adaptive median filtering with i = 7

DIP

LECTURE No. ( 9 )

# *Image Segmentation: Thresholding*

10/14/2019

Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

# Contents

So far we have been considering image processing techniques used to transform images for human interpretation

Today we will begin looking at automated image analysis by examining the thorny issue of image segmentation:

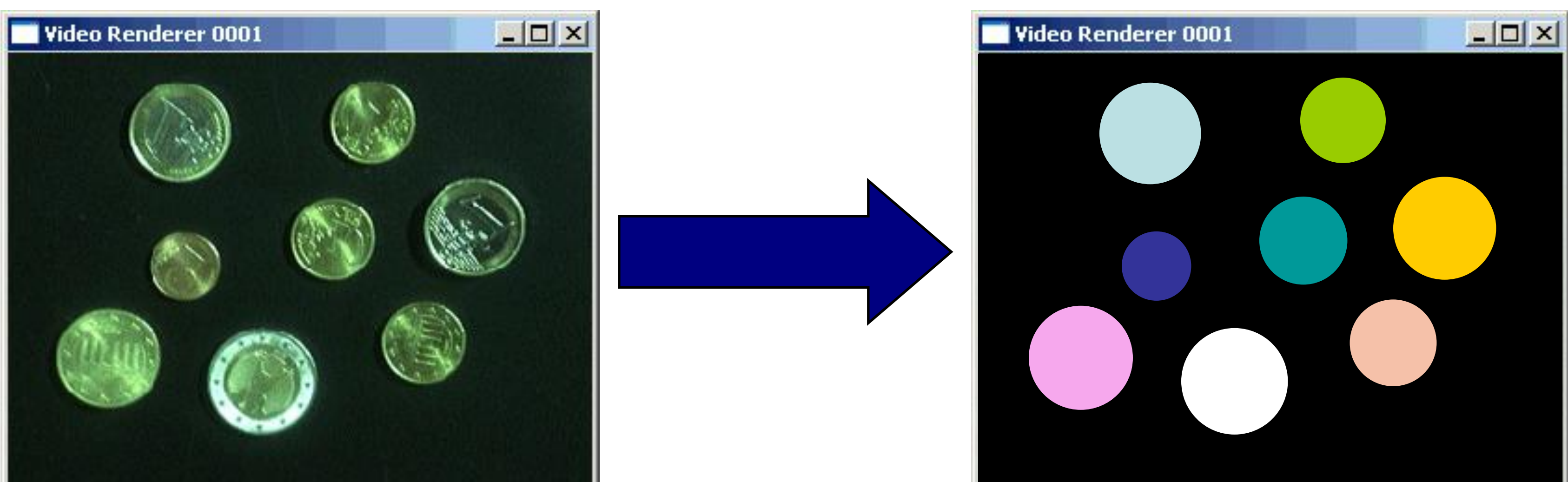
– The segmentation problem

– Finding points, lines and edges

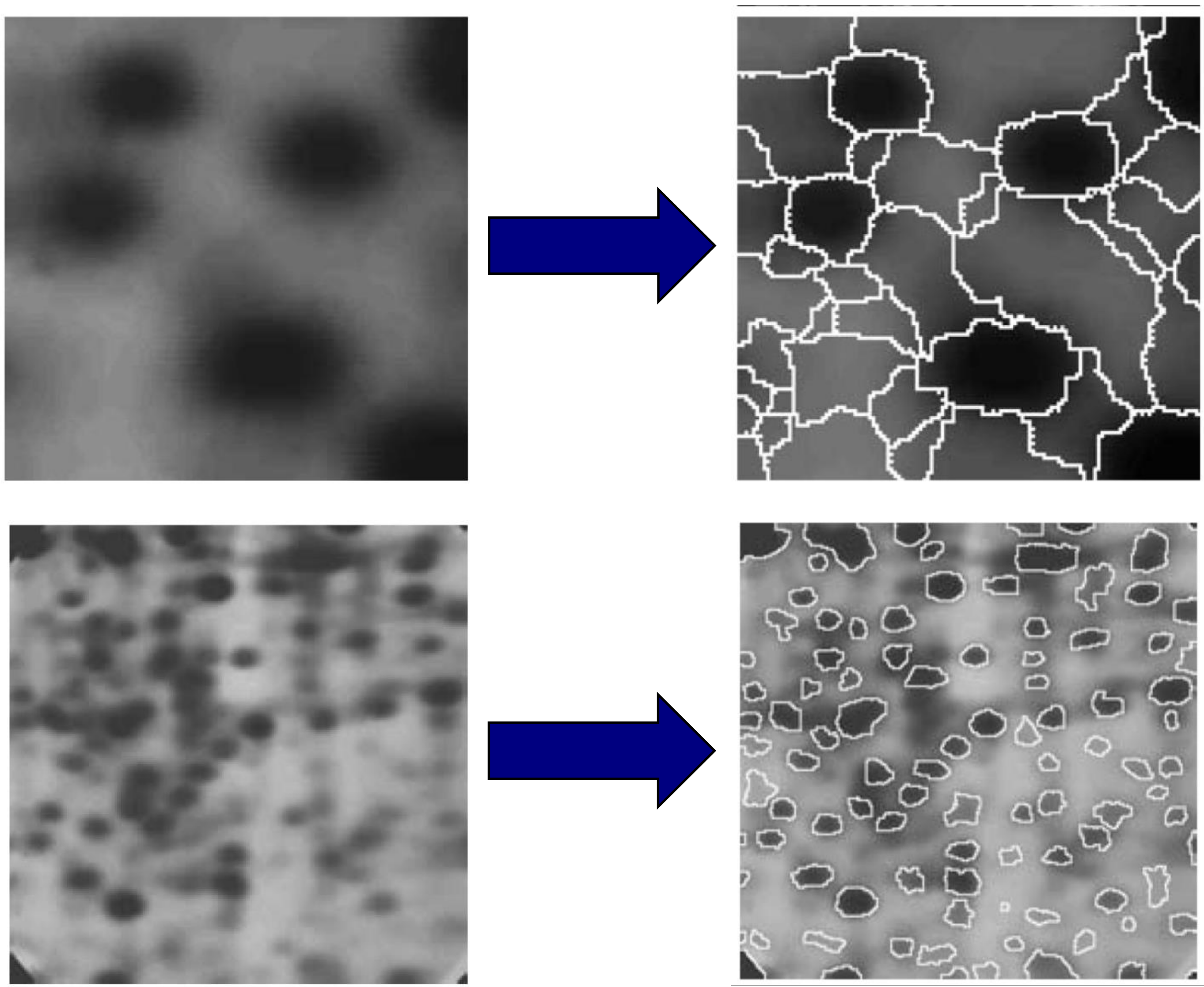
# The Segmentation Problem

Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image

Typically the first step in any automated computer vision application



# Segmentation Examples



# Detection Of Discontinuities

There are three basic types of grey level discontinuities that we tend to look for in digital images:

- Points
- Lines
- Edges

We typically find discontinuities using masks and correlation

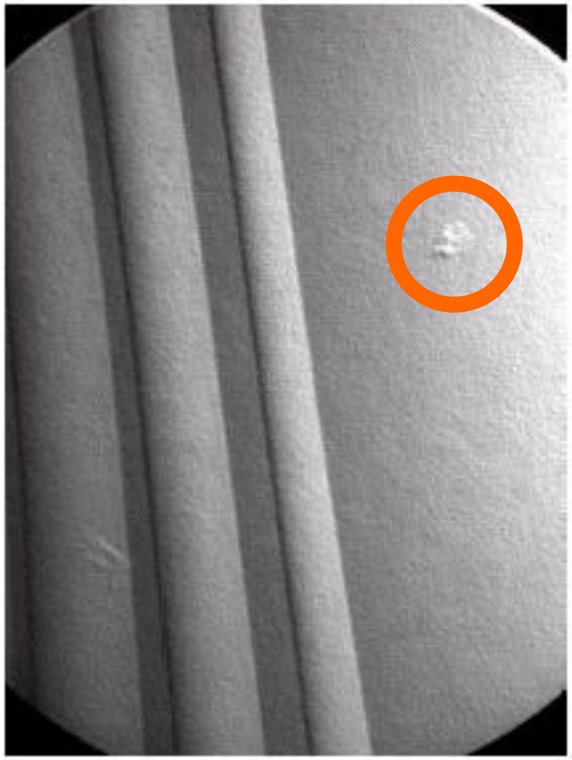
# Point Detection

Point detection can be achieved simply using the mask below:

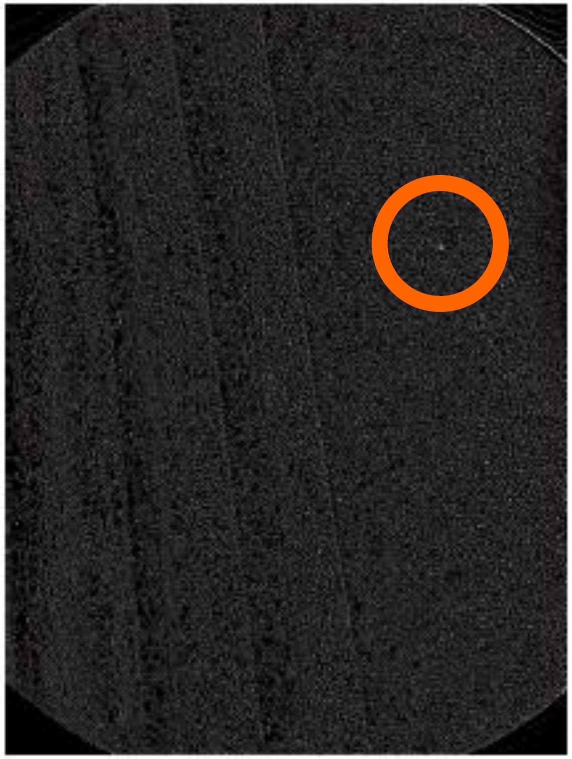
| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

Points are detected at those pixels in the subsequent filtered image that are above a set threshold

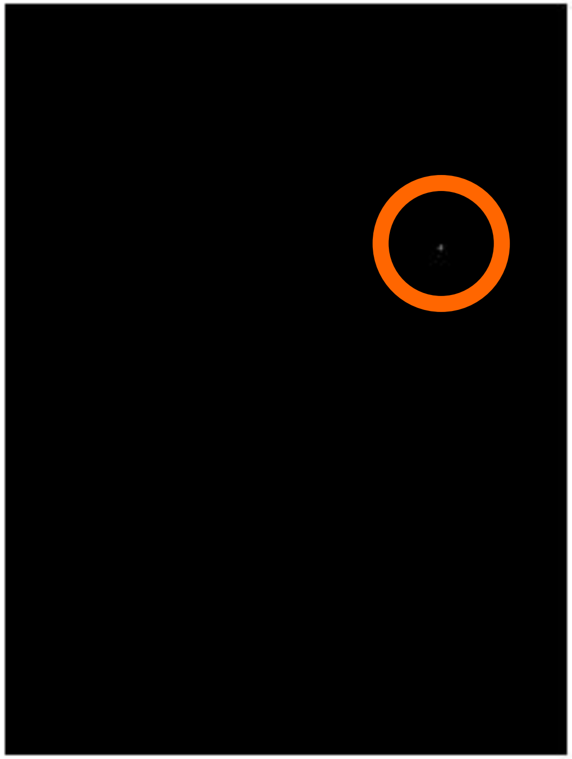
# Point Detection (cont…)



X-ray image of
a turbine blade



Result of point
detection



Result of
thresholding

# Line Detection

The next level of complexity is to try to detect lines

The masks below will extract lines that are one pixel thick and running in a particular direction

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 2 | 2 | 2 |
| −1 | −1 | −1 |

Horizontal

| | | |
|---|---|---|
| −1 | −1 | 2 |
| −1 | 2 | −1 |
| 2 | −1 | −1 |

+45°

| | | |
|---|---|---|
| −1 | 2 | −1 |
| −1 | 2 | −1 |
| −1 | 2 | −1 |

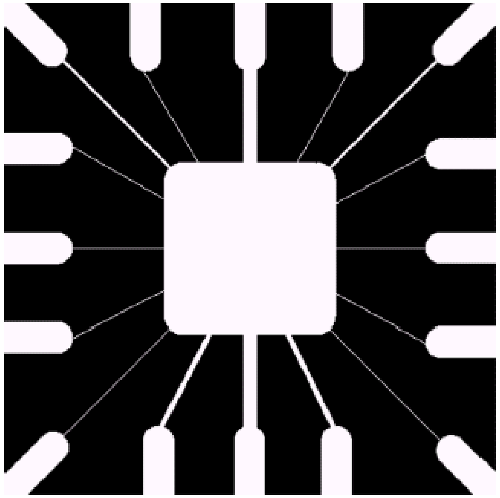
Vertical

| | | |
|---|---|---|
| 2 | −1 | −1 |
| −1 | 2 | −1 |
| −1 | −1 | 2 |

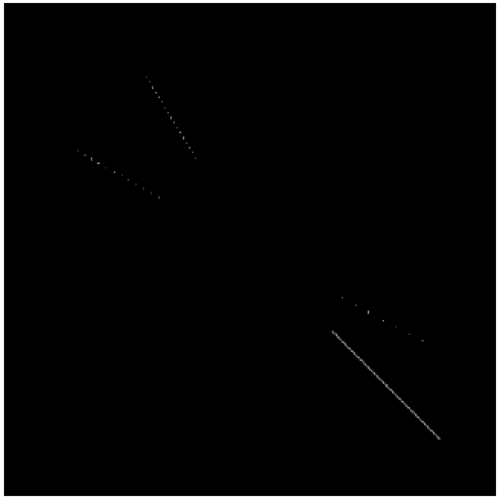
−45°

# Line Detection (cont…)

Binary image of a wire bond mask



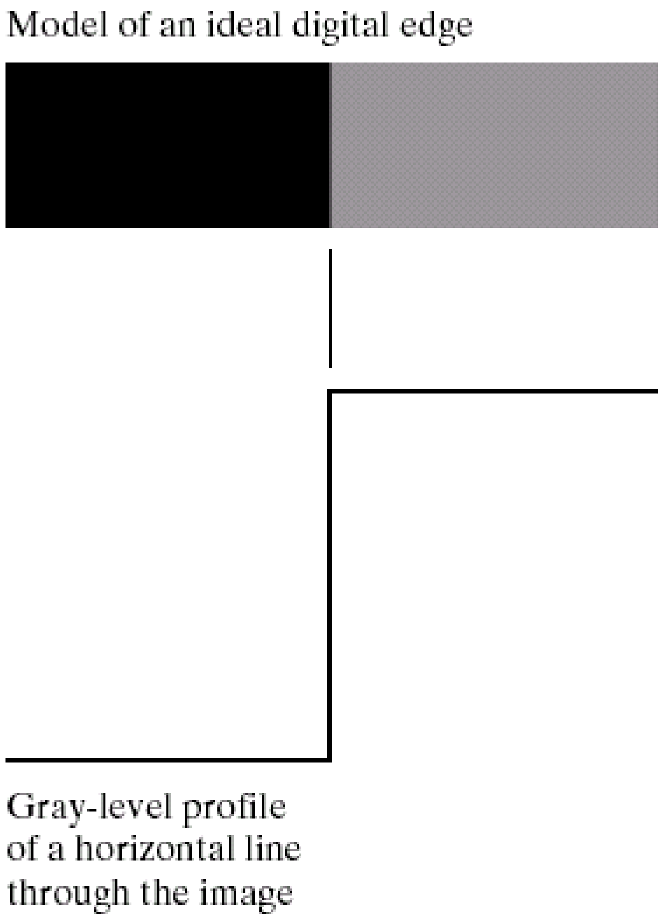
After processing with -45° line detector

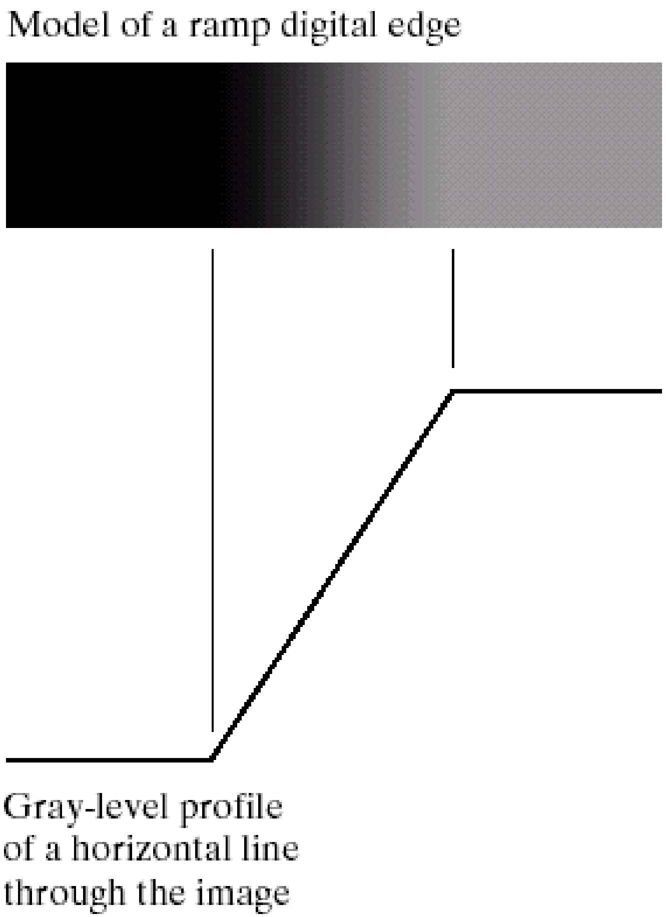


Result of thresholding filtering result

# Edge Detection

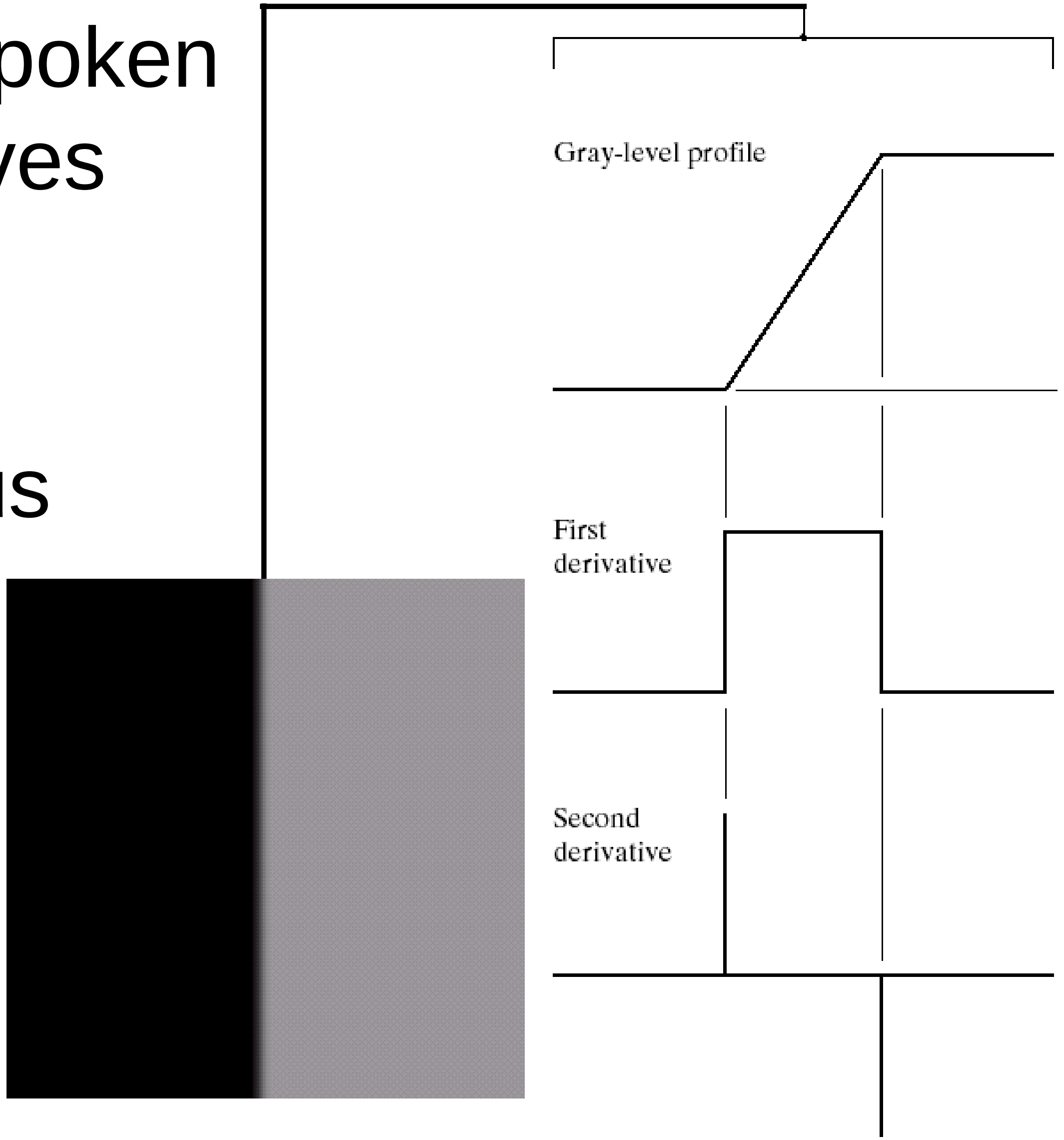
An edge is a set of connected pixels that lie on the boundary between two regions



# Edges & Derivatives

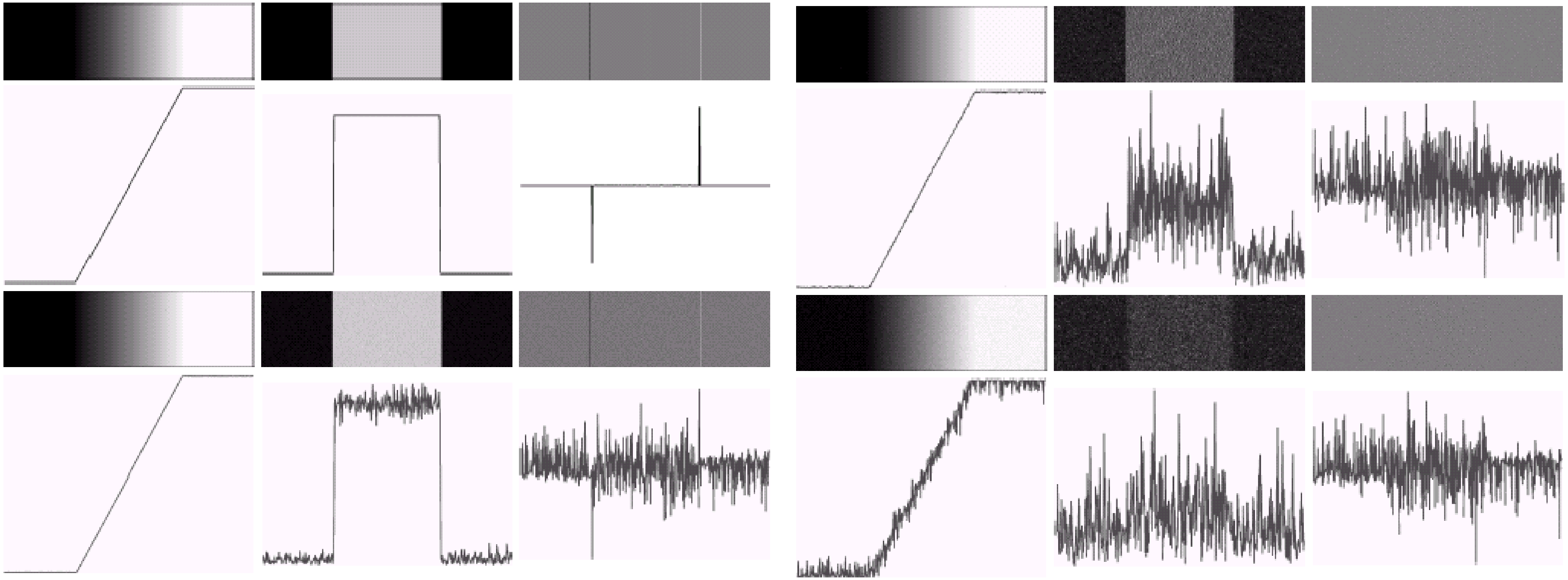
- We have already spoken about how derivatives are used to find discontinuities
- 1st derivative tells us where an edge is
- 2nd derivative can be used to show edge direction



# Derivatives & Noise

Derivative based edge detectors are extremely sensitive to noise

We need to keep this in mind



# Common Edge Detectors

Given a 3*3 region of an image the following edge detection filters can be used

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

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

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

Prewitt

| | |
|---|---|
| −1 | 0 |
| 0 | 1 |

| | |
|---|---|
| 0 | −1 |
| 1 | 0 |

Roberts

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

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

Sobel

# Edge Detection Example

Original Image

Horizontal Gradient Component

Vertical Gradient Component

Combined Edge Image

# Edge Detection Example

# Edge Detection Example

# Edge Detection Example

# Edge Detection Example

# Edge Detection Problems

Often, problems arise in edge detection in that there are is too much detail

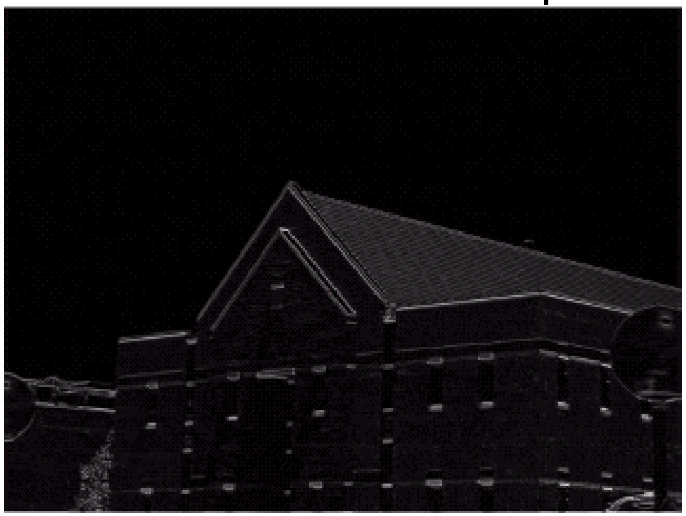
For example, the brickwork in the previous example

One way to overcome this is to smooth images prior to edge detection

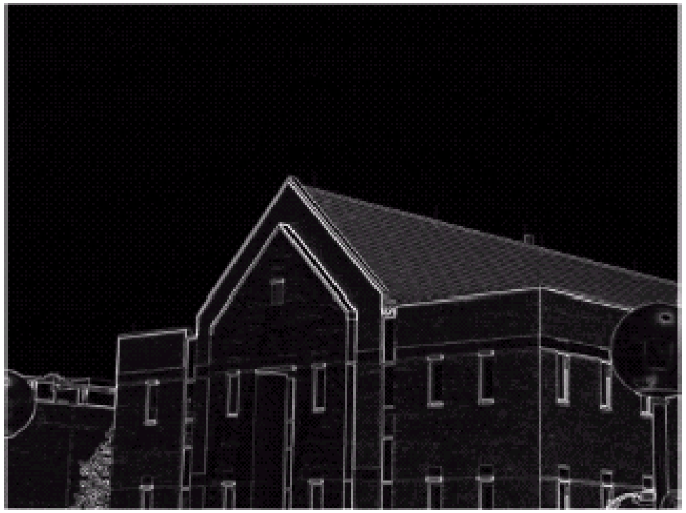
# Edge Detection Example With Smoothing

Original Image

Horizontal Gradient Component



Vertical Gradient Component



Combined Edge Image

# Laplacian Edge Detection

We encountered the 2nd-order derivative based Laplacian filter already

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

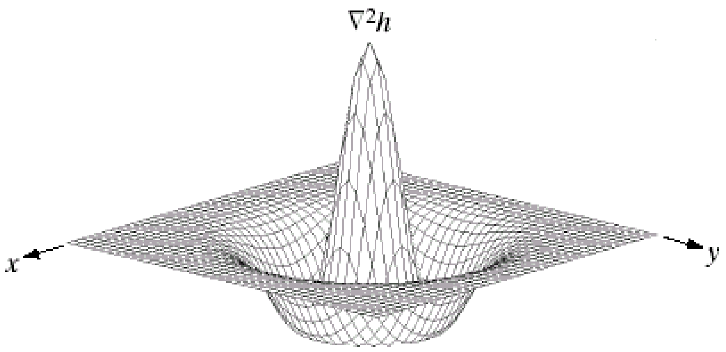
| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

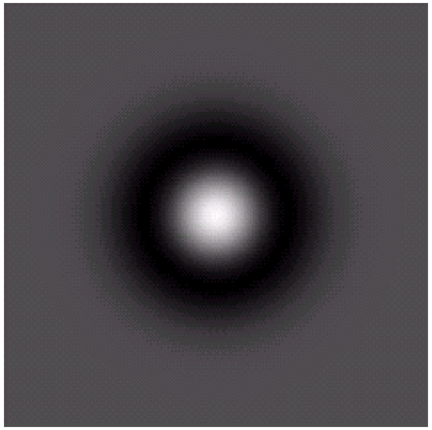
The Laplacian is typically not used by itself as it is too sensitive to noise

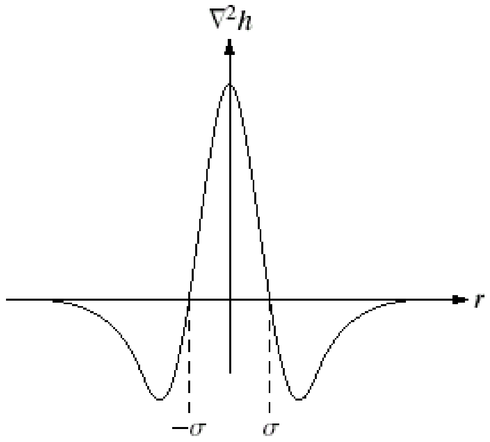
Usually hen used for edge detection the Laplacian is combined with a smoothing Gaussian filter

# Laplacian Of Gaussian

The Laplacian of Gaussian (or Mexican hat) filter uses the Gaussian for noise removal and the Laplacian for edge detection

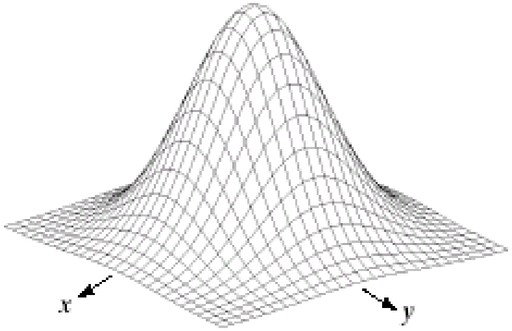


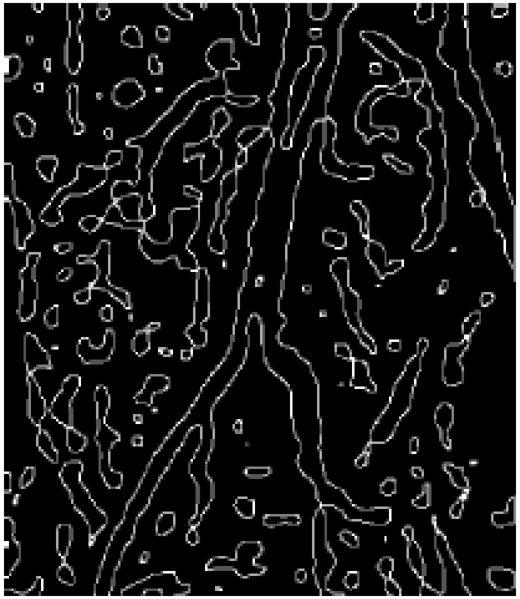



| | | | | |
|---|---|---|---|---|
| 0 | 0 | −1 | 0 | 0 |
| 0 | −1 | −2 | −1 | 0 |
| −1 | −2 | 16 | −2 | −1 |
| 0 | −1 | −2 | −1 | 0 |
| 0 | 0 | −1 | 0 | 0 |

# Laplacian Of Gaussian Example




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



# Summary

In this lecture we have begun looking at segmentation, and in particular edge detection

Edge detection is massively important as it is in many cases the first step to object recognition

DIP

LECTURE No. ( 10 )

# *Image Segmentation:*

# *Thresholding*

10/14/2019

Dr. Majid D. Y.
dr.majid@uomosul.edu.iq

# Contents

Today we will continue to look at the problem of segmentation, this time though in terms of thresholding

In particular we will look at:

- What is thresholding?
- Simple thresholding
- Adaptive thresholding

# Thresholding

Thresholding is usually the first step in any segmentation approach

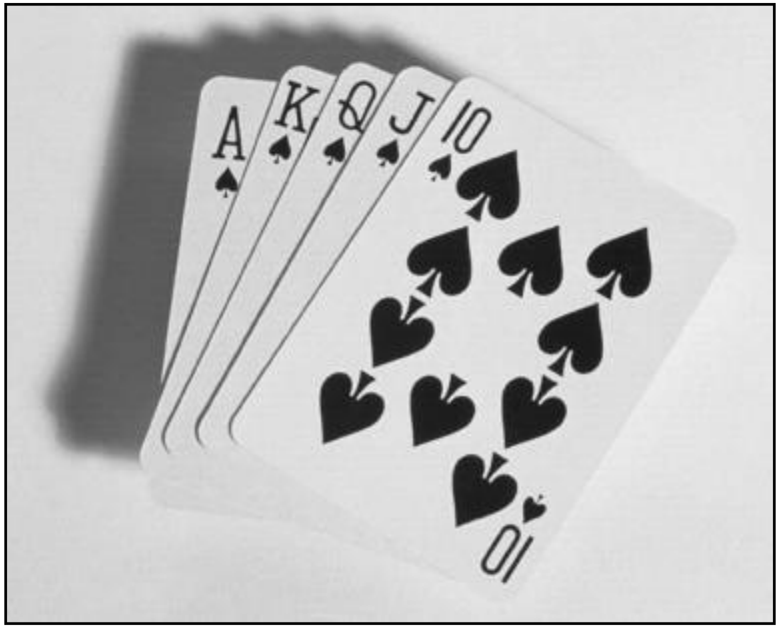
We have talked about simple single value thresholding already

Single value thresholding can be given mathematically as follows:

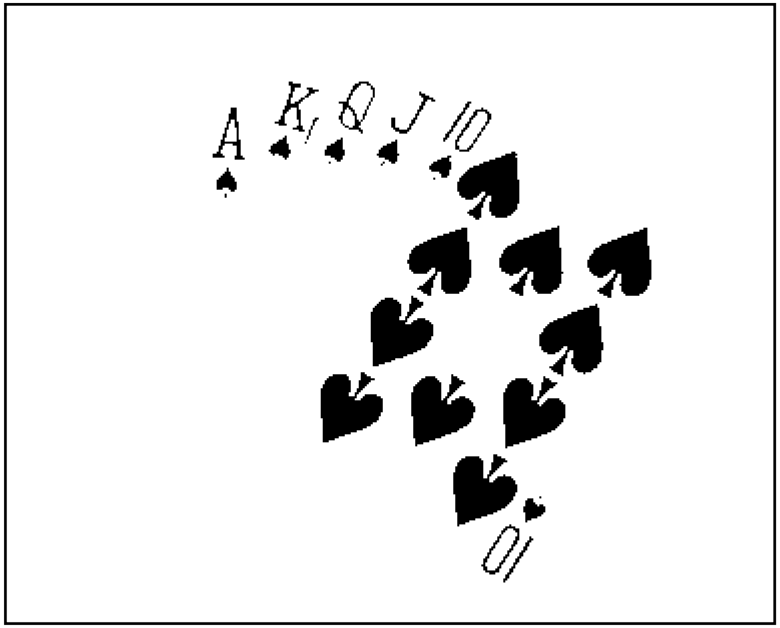
$$g(x,y) = \begin{cases} 1\, if \;\; f(x,y) > T \\ 0\, if \;\; f(x,y) \le T \end{cases}$$

# Thresholding Example

Imagine a poker playing robot that needs to visually interpret the cards in its hand



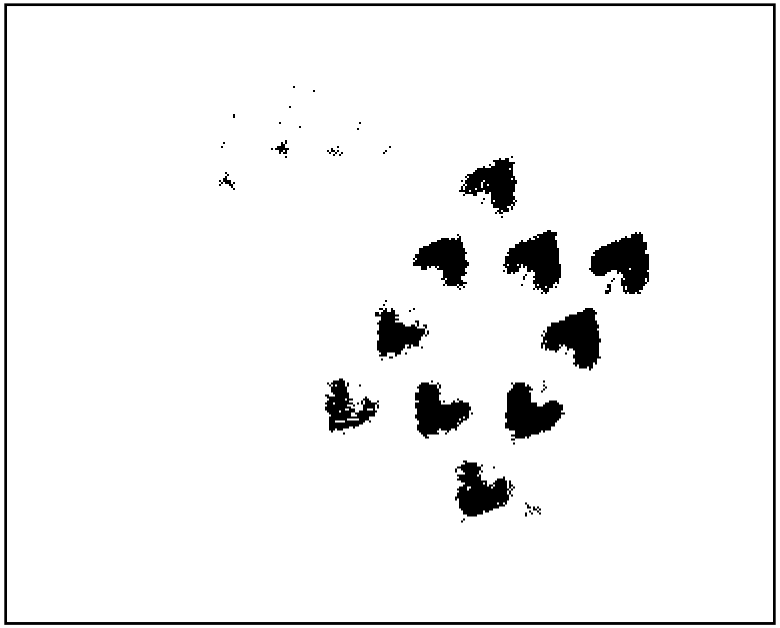
Original Image



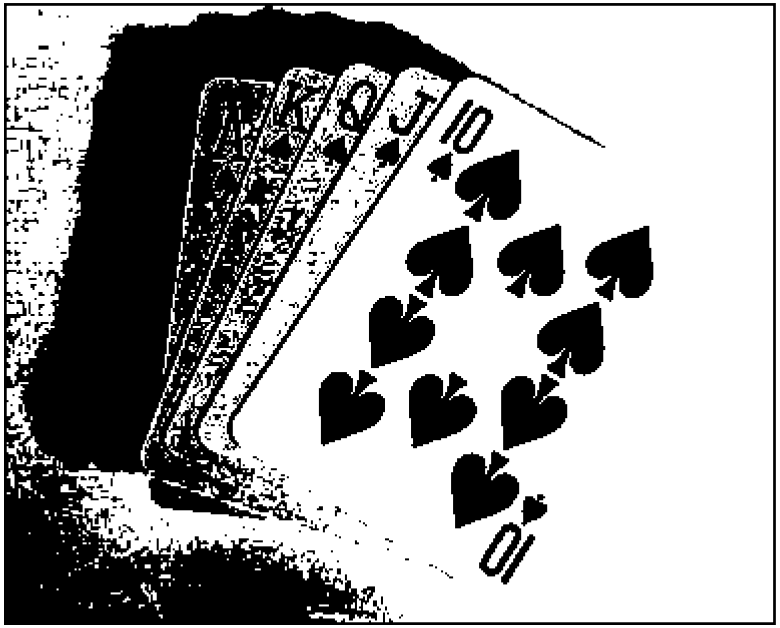
Thresholded Image

# But Be Careful

If you get the threshold wrong the results can be disastrous



Threshold Too Low



Threshold Too High

# Basic Global Thresholding

Based on the histogram of an image

Partition the image histogram using a single global threshold

The success of this technique very strongly depends on how well the histogram can be partitioned

# Basic Global Thresholding Algorithm

The basic global threshold, T, is calculated as follows:

1. Select an initial estimate for T (typically the average grey level in the image)
2. Segment the image using T to produce two groups of pixels: $G_1$ consisting of pixels with grey levels >T and $G_2$ consisting pixels with grey levels ≤ T
3. Compute the average grey levels of pixels in $G_1$ to give $\mu_1$ and $G_2$ to give $\mu_2$

# Basic Global Thresholding Algorithm

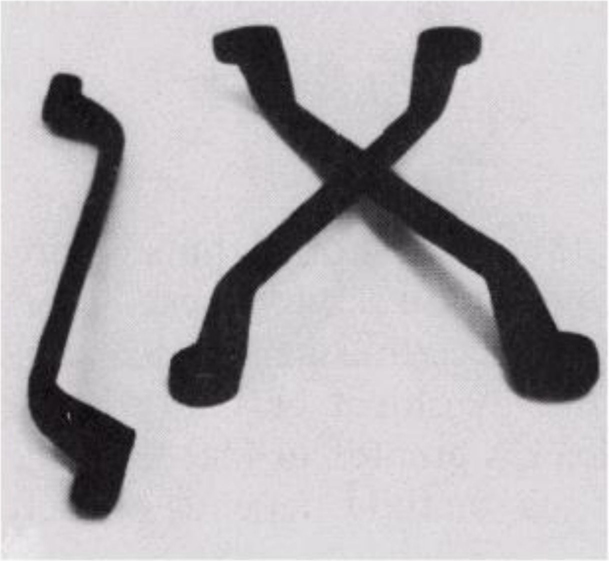
4. Compute a new threshold value:

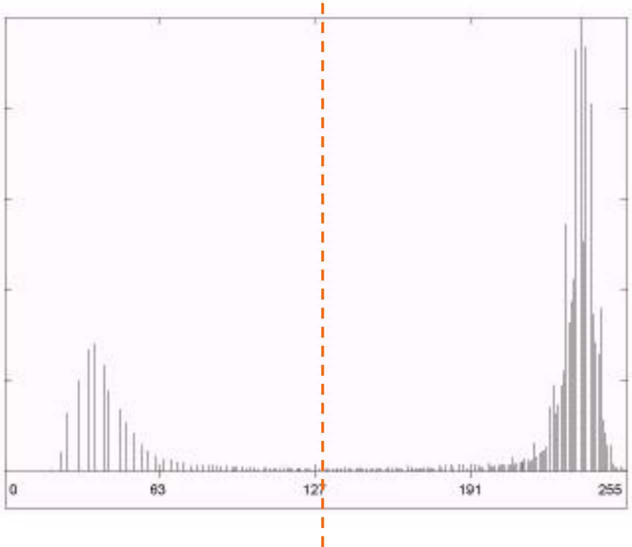
$$T = \frac{\mu_1 + \mu_2}{2}$$

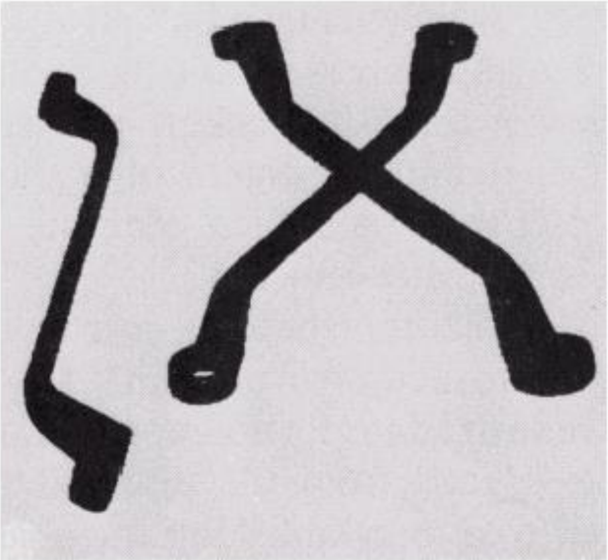
5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit $T_\infty$

This algorithm works very well for finding thresholds when the histogram is suitable

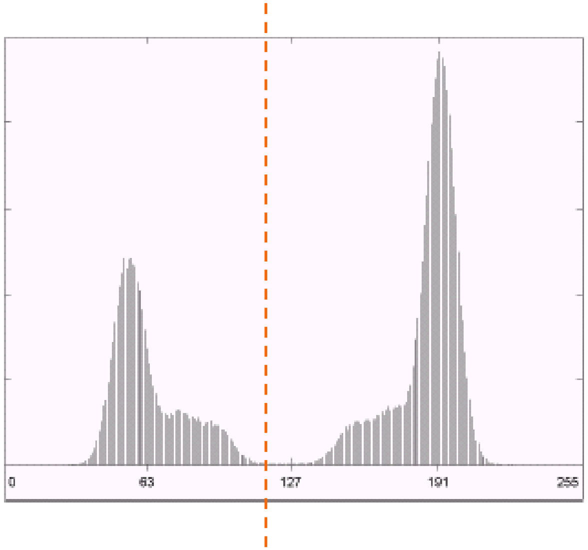
# Thresholding Example 1





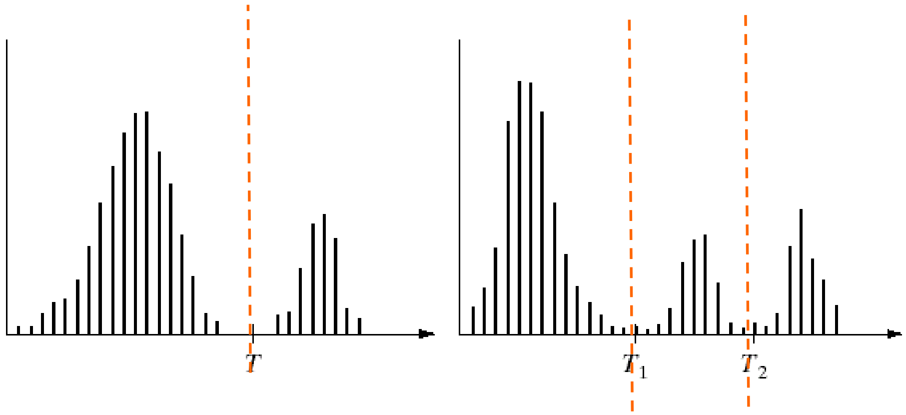
# Thresholding Example 2




# Problems With Single Value Thresholding

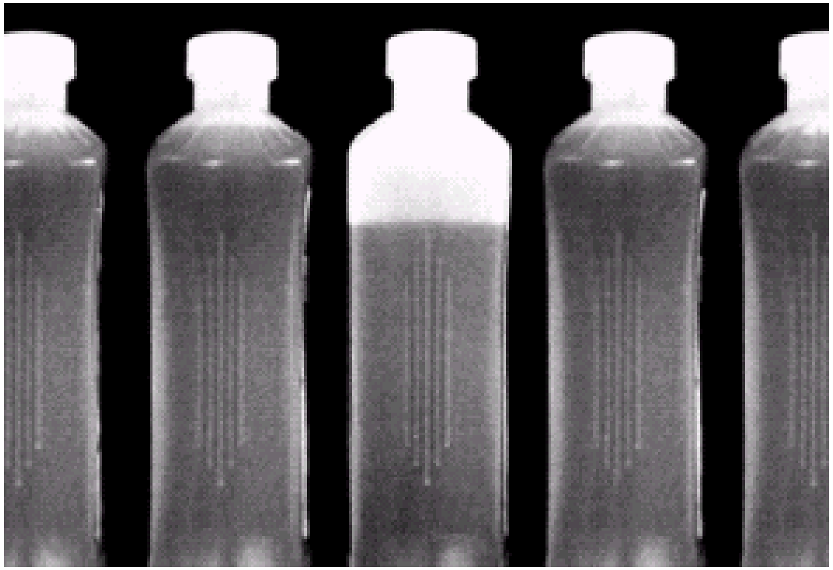
Single value thresholding only works for bimodal histograms

Images with other kinds of histograms need more than a single threshold



# Problems With Single Value Thresholding (cont…)

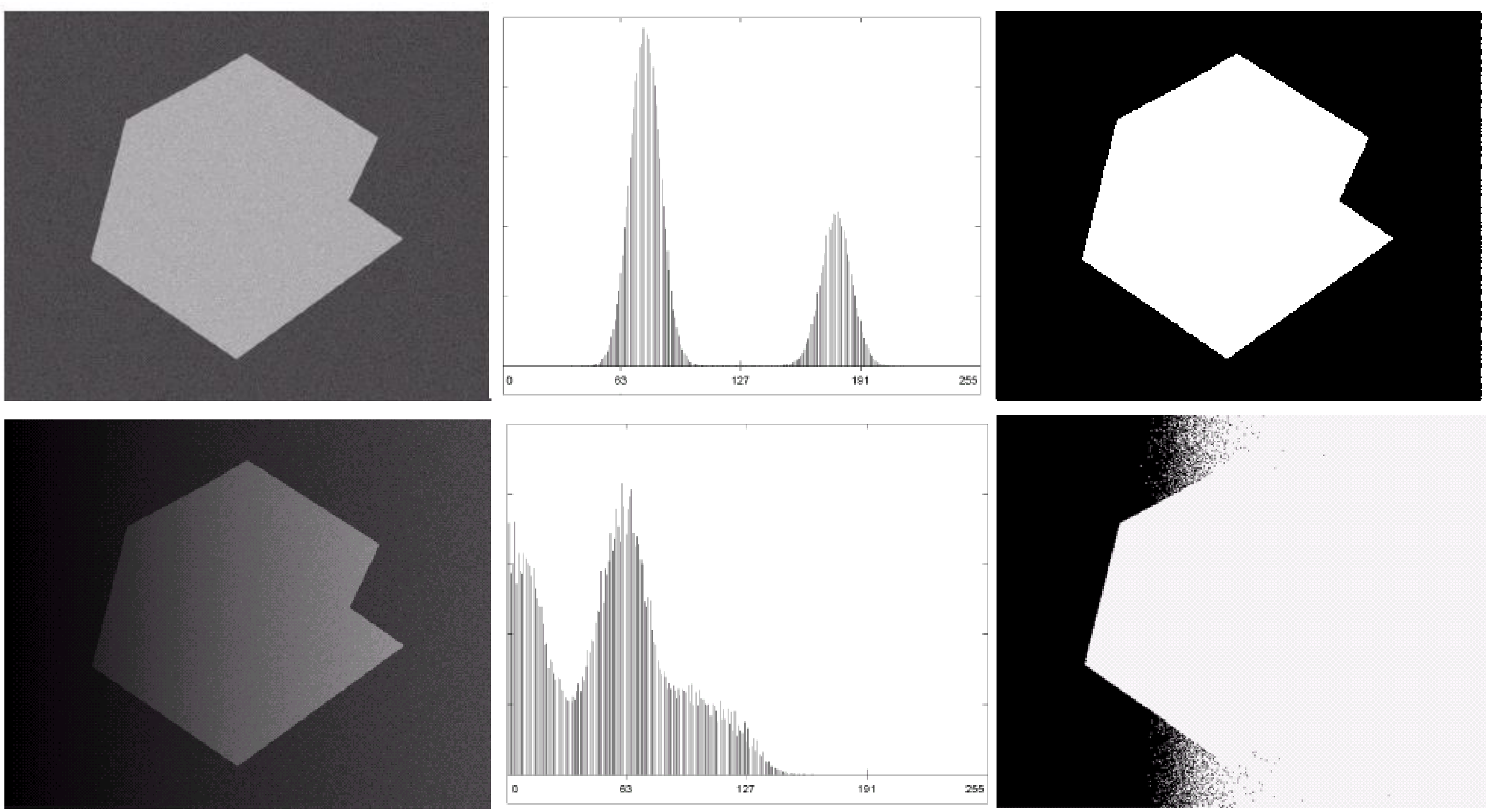
Let's say we want to isolate the contents of the bottles



Think about what the histogram for this image would look like

What would happen if we used a single threshold value?

# Single Value Thresholding and Illumination




Uneven illumination can really upset a single valued thresholding scheme

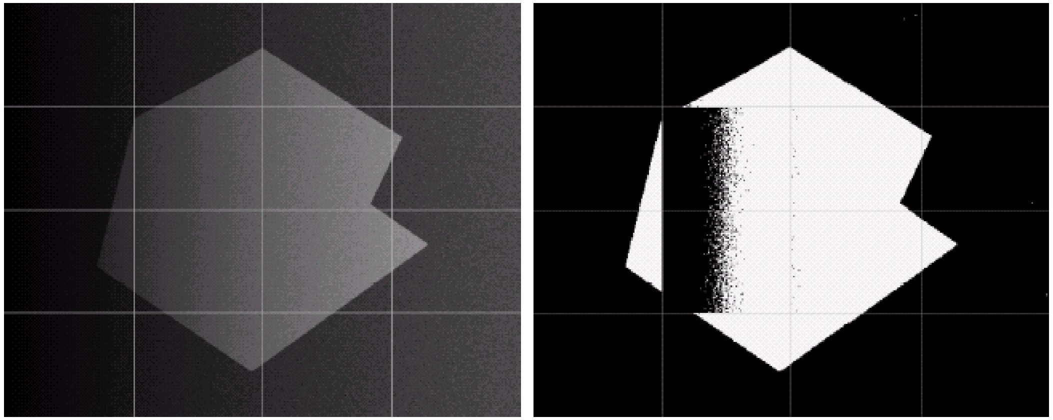
# Basic Adaptive Thresholding

An approach to handling situations in which single value thresholding will not work is to divide an image into sub images and threshold these individually

Since the threshold for each pixel depends on its location within an image this technique is said to *adaptive*

# Basic Adaptive Thresholding Example

The image below shows an example of using adaptive thresholding with the image shown previously



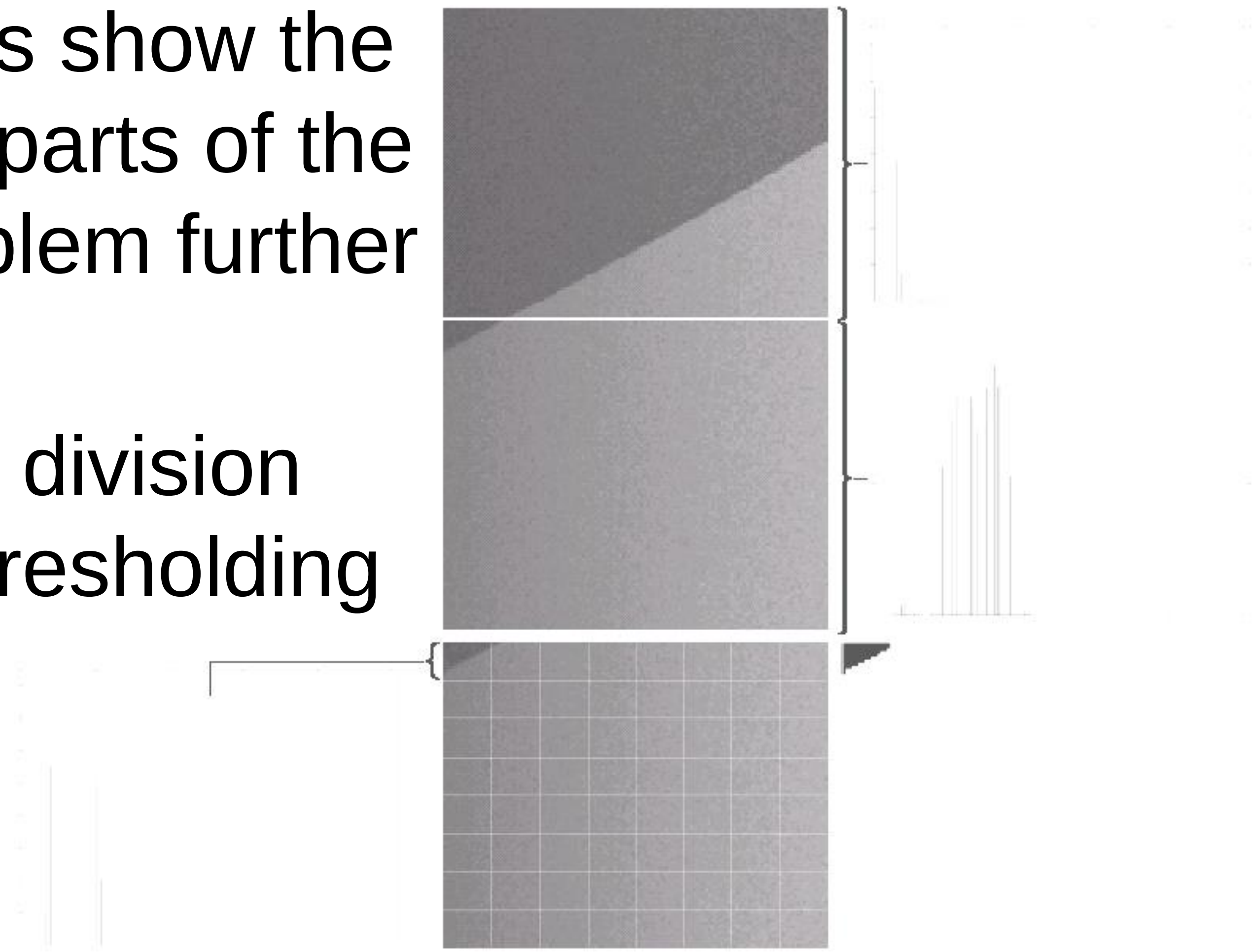
As can be seen success is mixed

But, we can further subdivide the troublesome sub images for more success

# Basic Adaptive Thresholding Example (cont…)

These images show the troublesome parts of the previous problem further subdivided

After this sub division successful thresholding can be achieved



# Summary

In this lecture we have begun looking at segmentation, and in particular thresholding

We saw the basic global thresholding algorithm and its shortcomings

We also saw a simple way to overcome some of these limitations using adaptive thresholding