Digital Image Processing

Introduction

Lecture 1 - Introduction

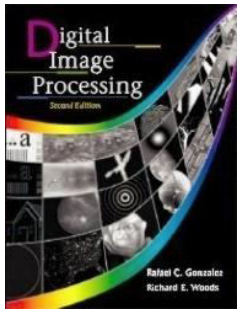
Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Introduction

“One picture is worth more than ten thousand words”

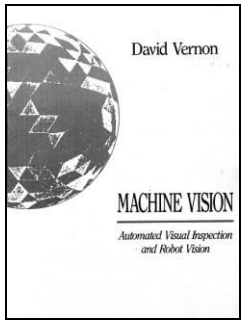
Anonymous

References



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

– Much of the material that follows is taken from this book



"Machine Vision: Automated Visual Inspection and Robot Vision", David Vernon, Prentice Hall, 1991

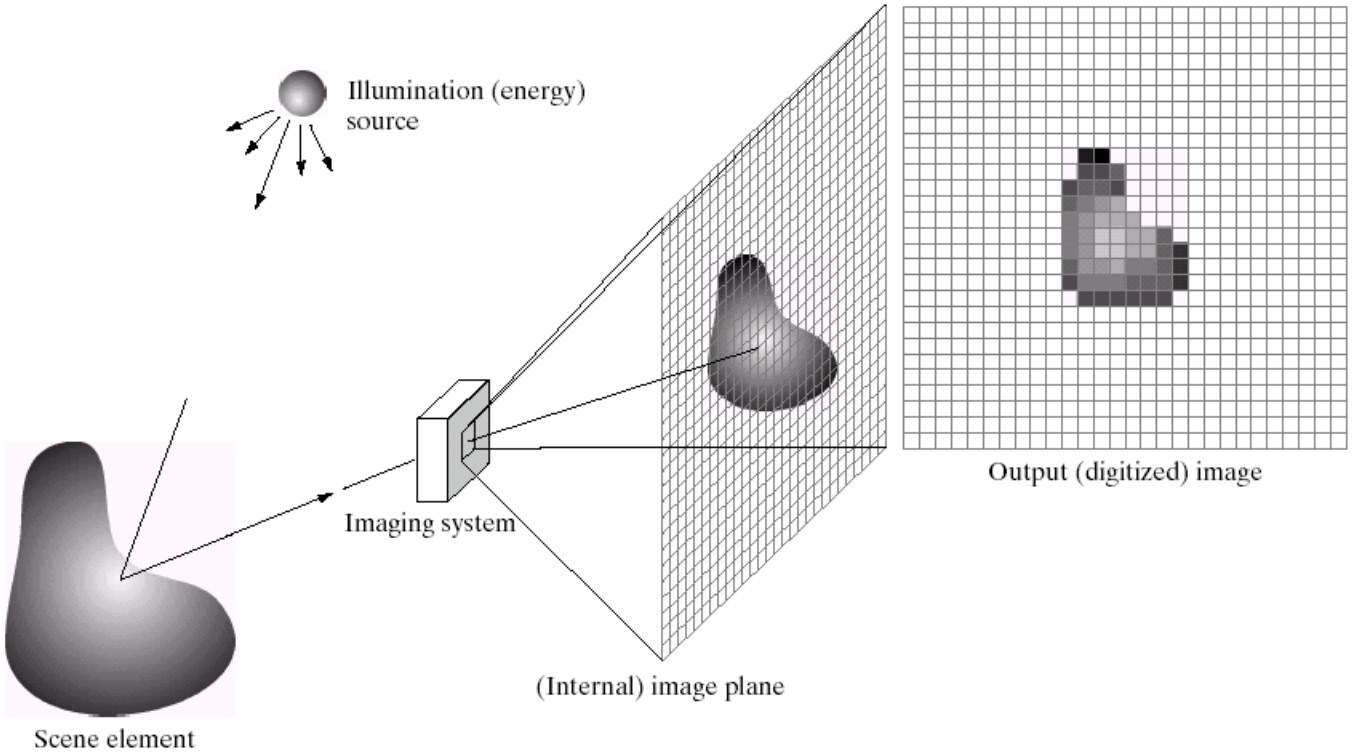
Contents

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?

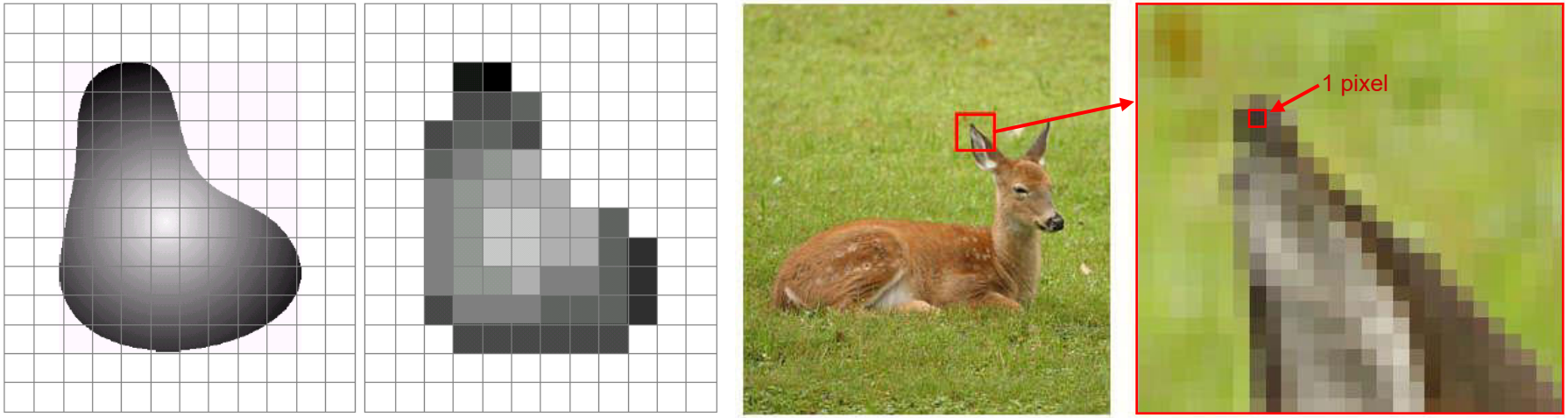
A **digital image** is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels



What is a Digital Image? (cont…)

Pixel values typically represent gray levels, colours, heights, opacities 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 **Output:** Image	**Input:** Image **Output:** Attributes	**Input:** Attributes **Output:** Understanding
Examples: Noise removal, image sharpening	**Examples:** Object recognition, segmentation	**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

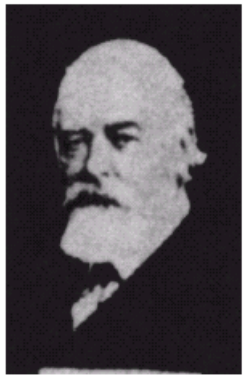
Early digital image

- 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



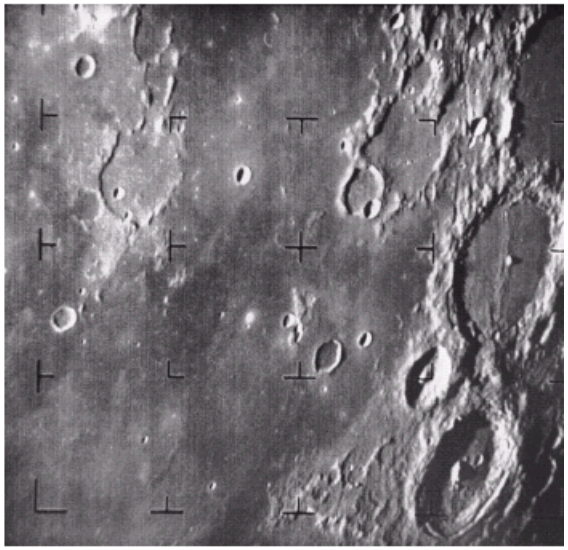
Improved digital image

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

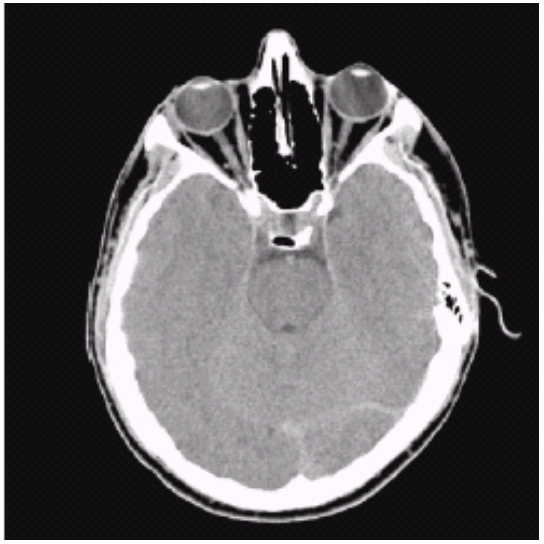


A picture of the moon taken by the Ranger 7 probe minutes before landing

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

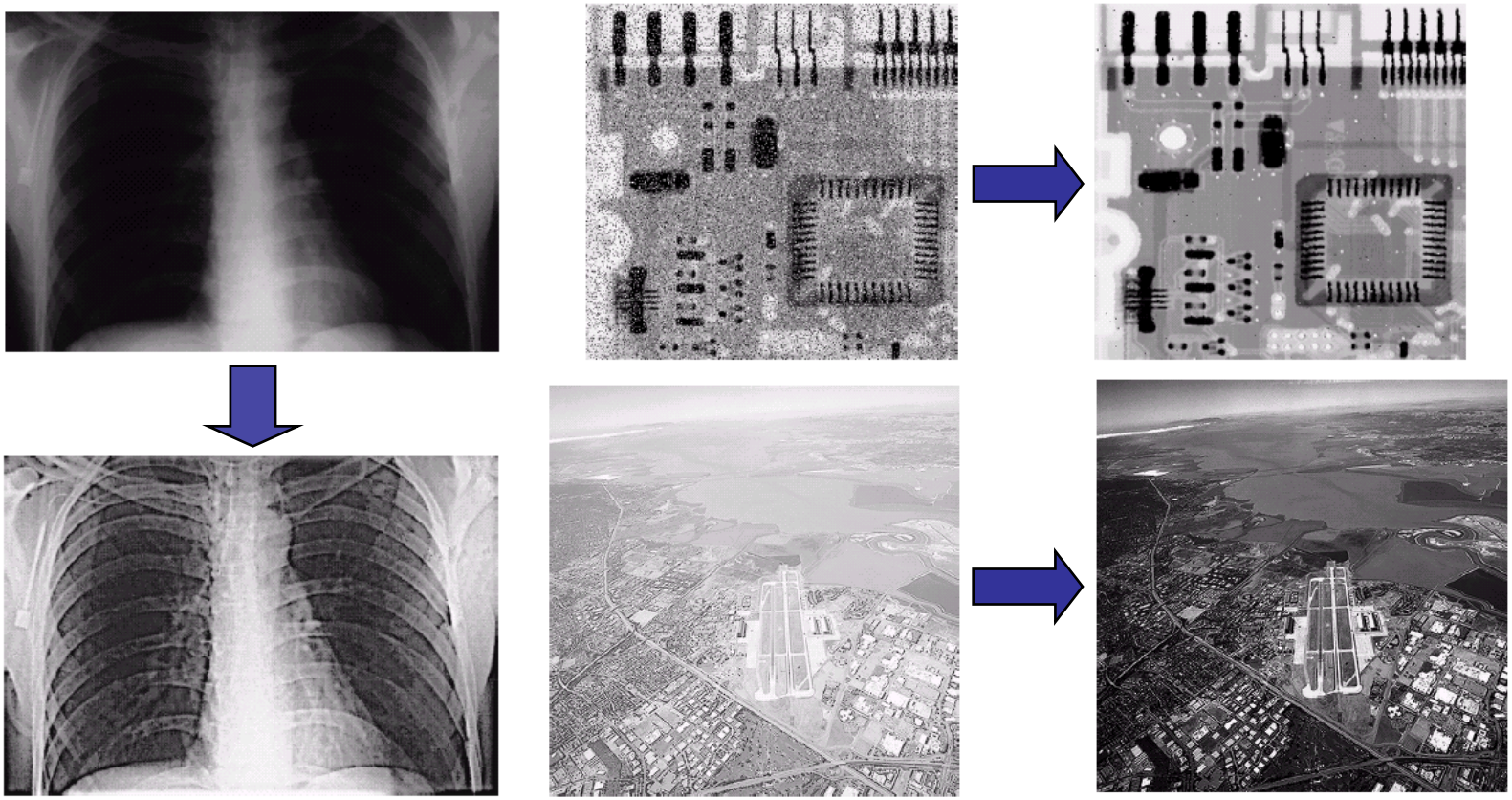
History of DIP (cont…)

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

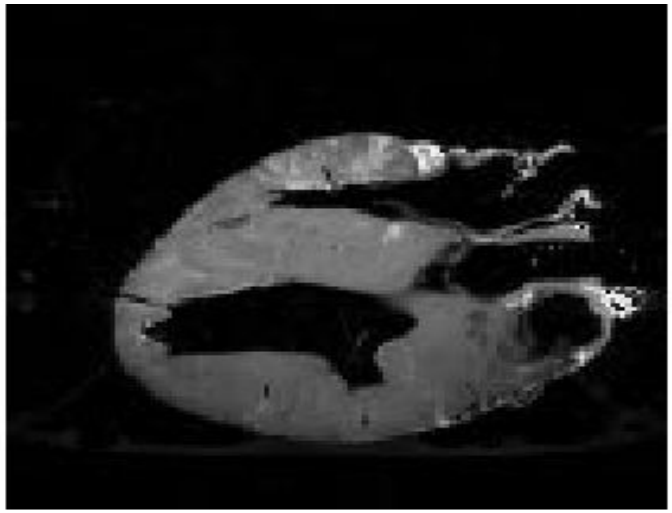
Examples: Artistic Effects

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

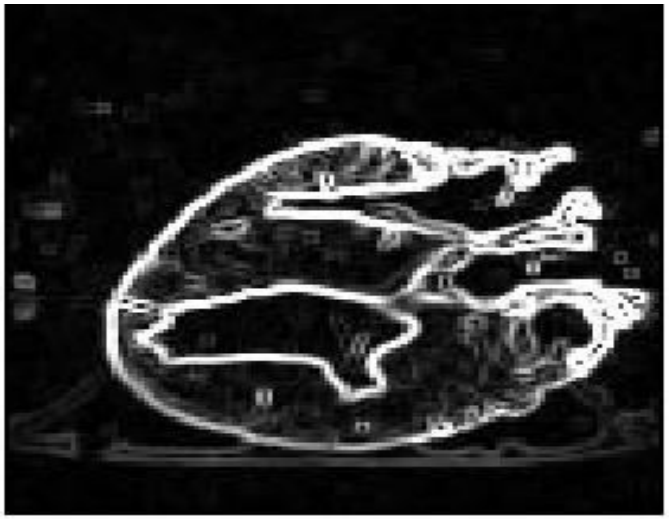
Examples: Medicine

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

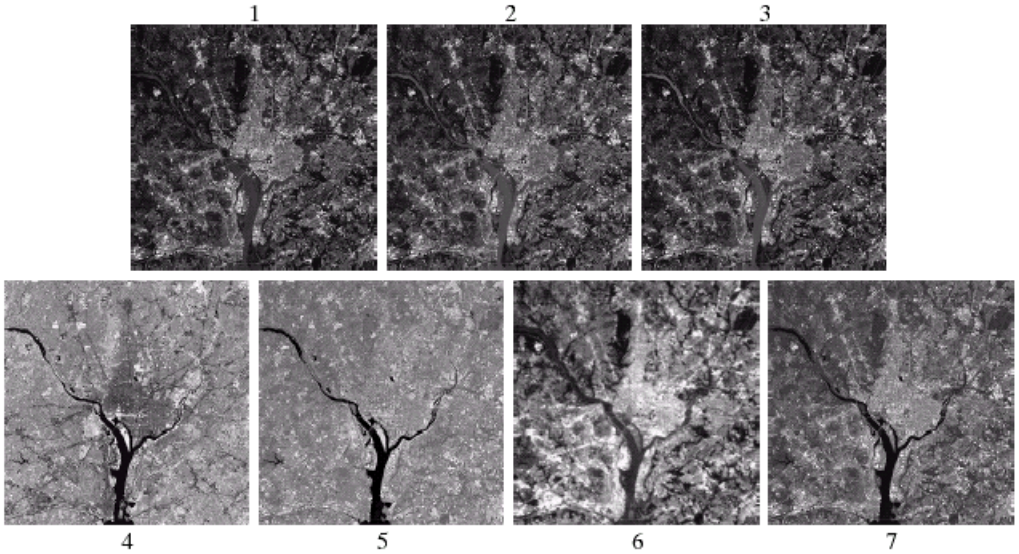


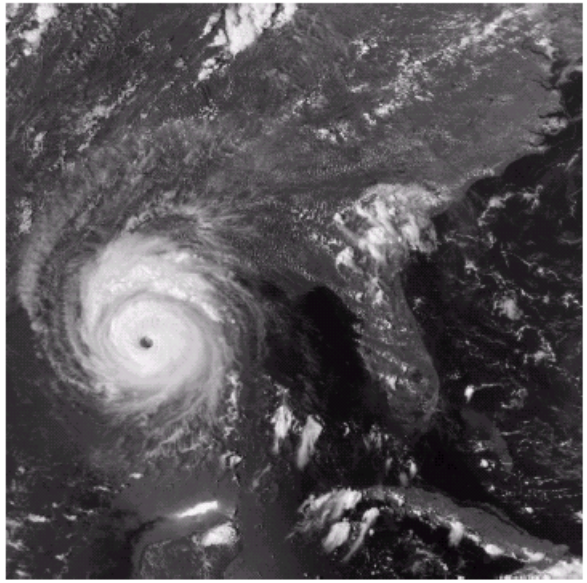
Edge Detection Image

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

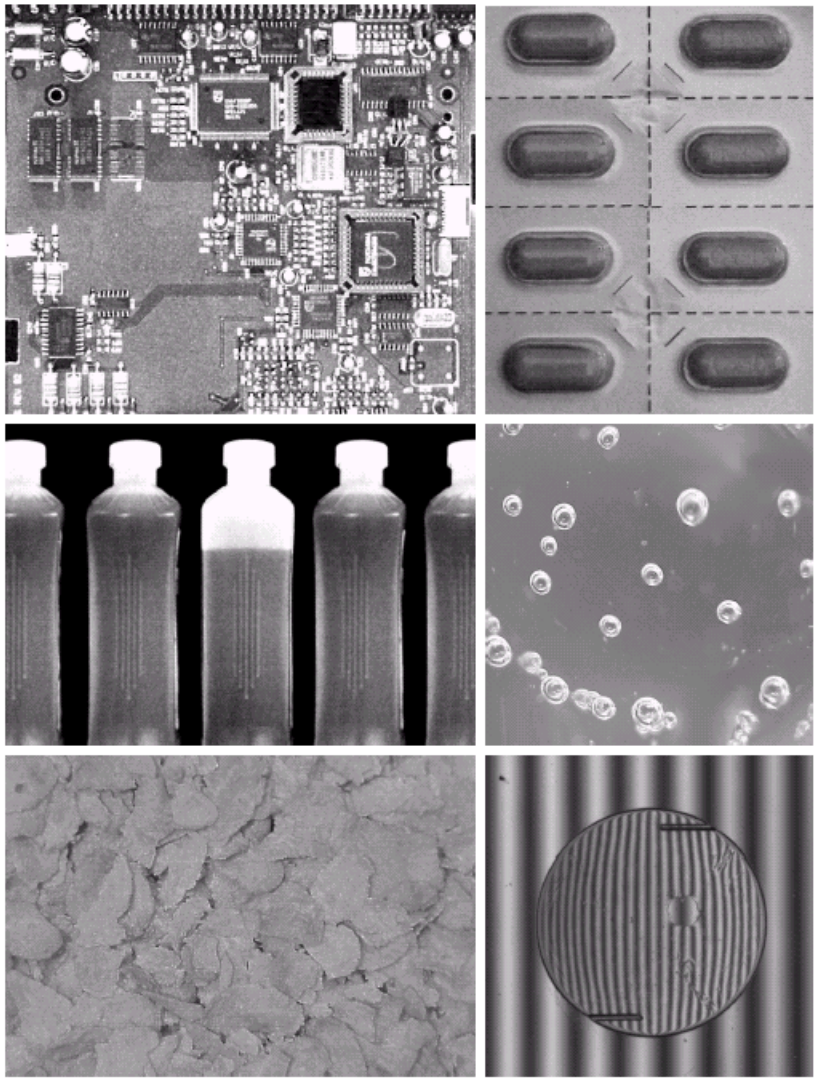
Examples: Industrial Inspection

Human operators are expensive, slow and unreliable

Make machines do the job instead

Industrial vision systems are used in all kinds of industries

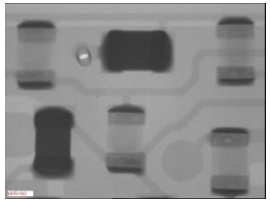
Can we trust them?

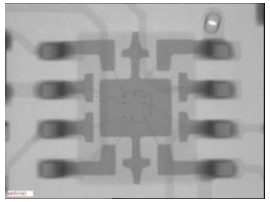


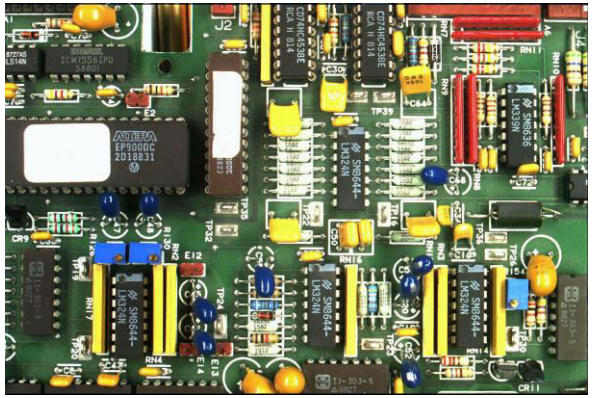
Examples: PCB Inspection

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



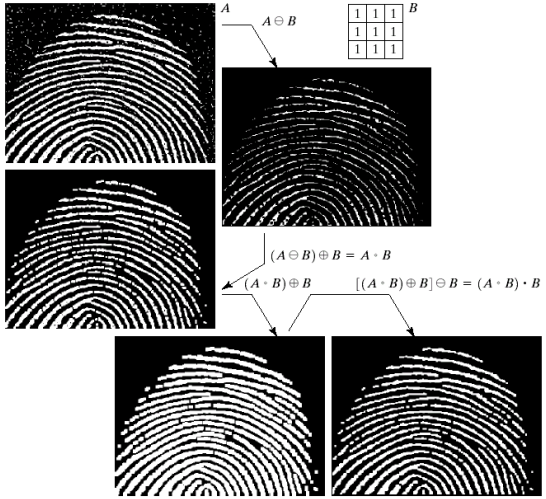




Examples: Law Enforcement

Image processing techniques are used extensively by law enforcers

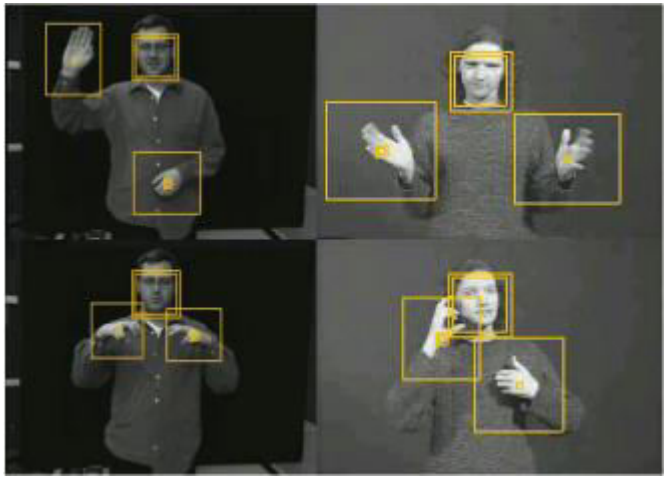
- Number plate recognition for speed cameras/automated toll systems
- Fingerprint recognition
- Enhancement of CCTV images



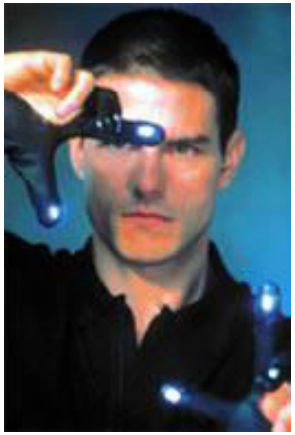
Examples: HCI

Try to make human computer interfaces more natural

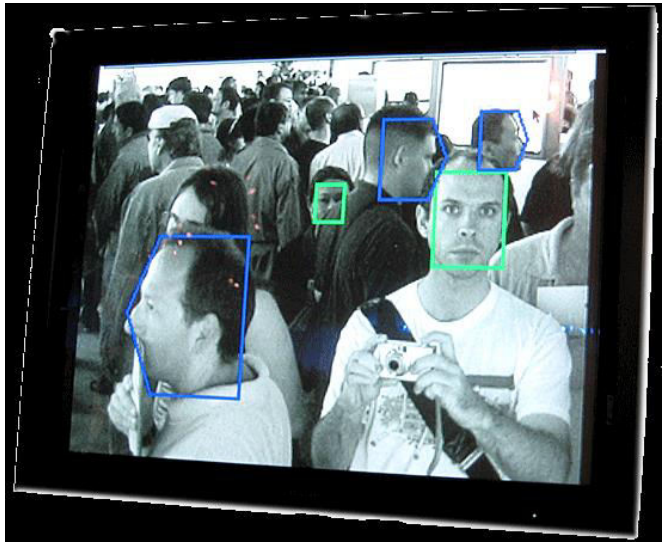
- Face recognition
- Gesture recognition



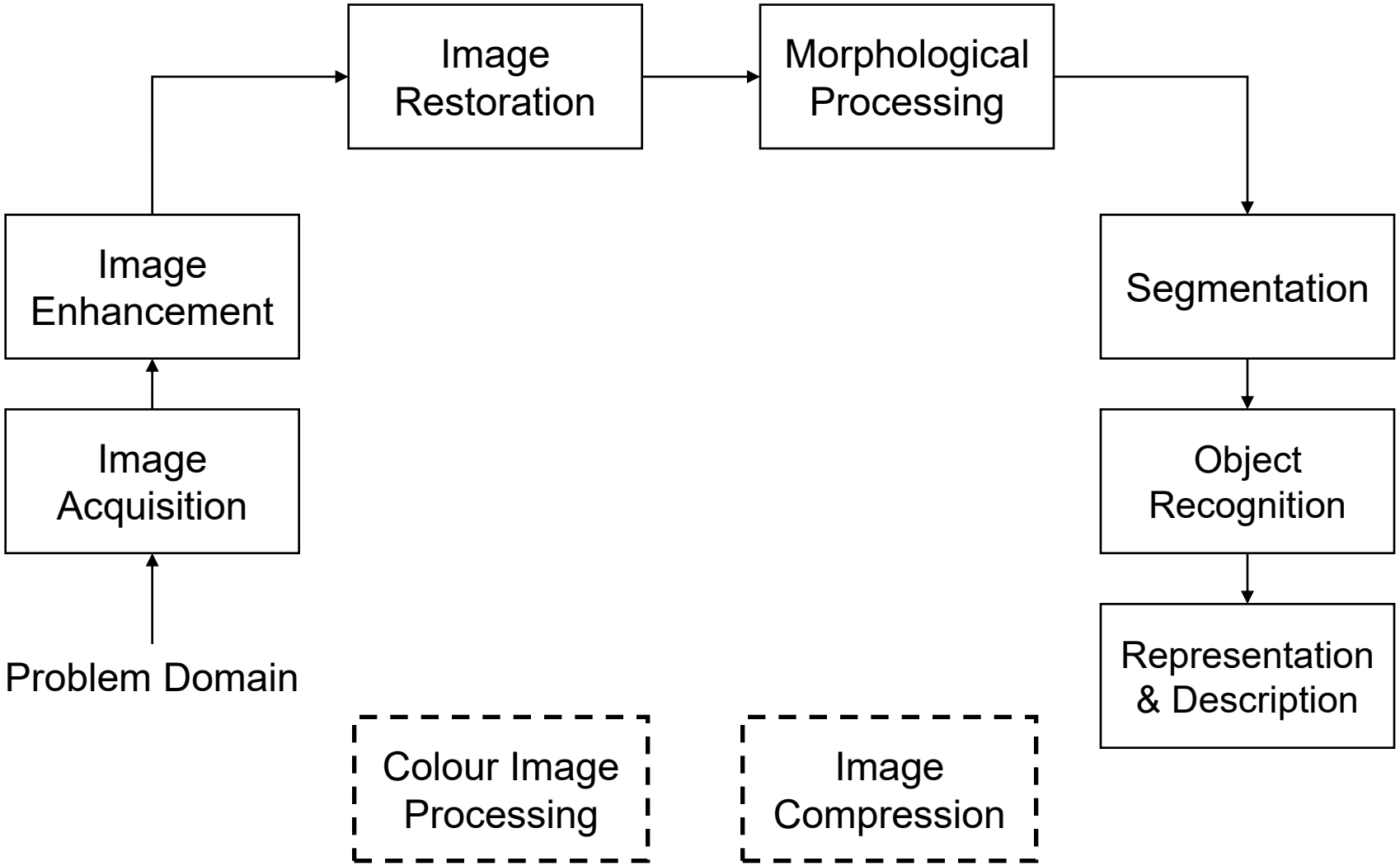
Does anyone remember the user interface from “Minority Report”?



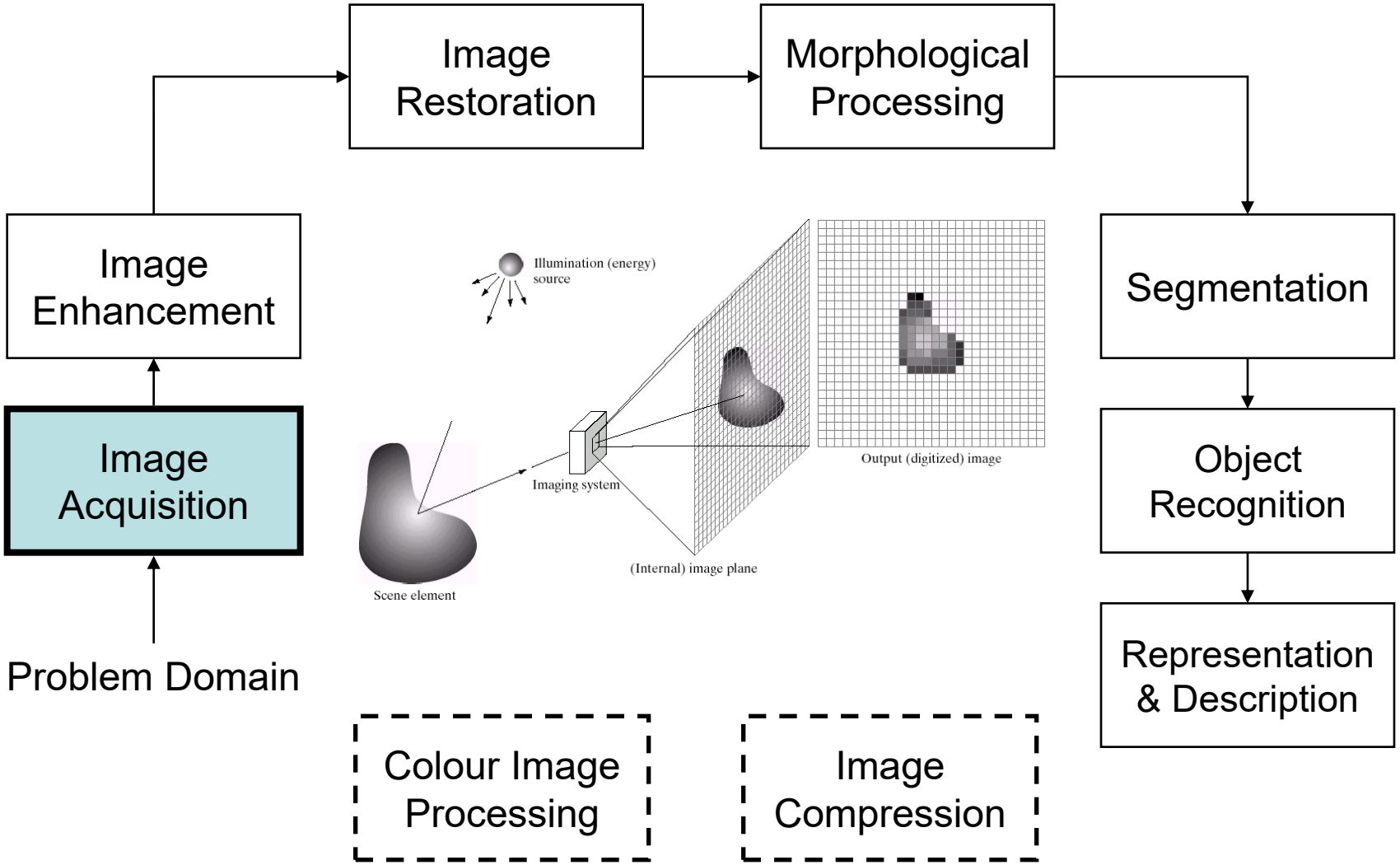
These tasks can be extremely difficult



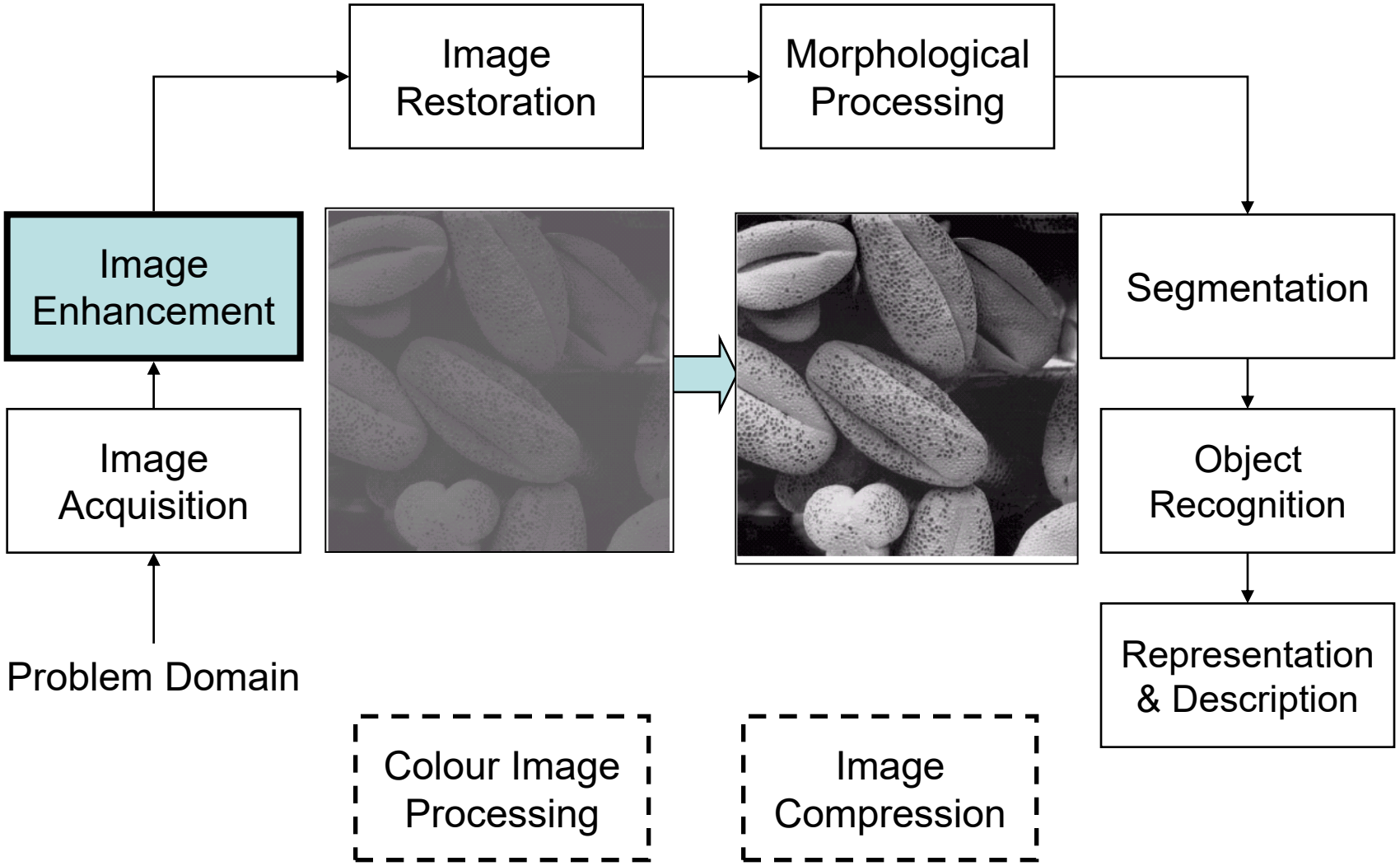
Key Stages in Digital Image Processing



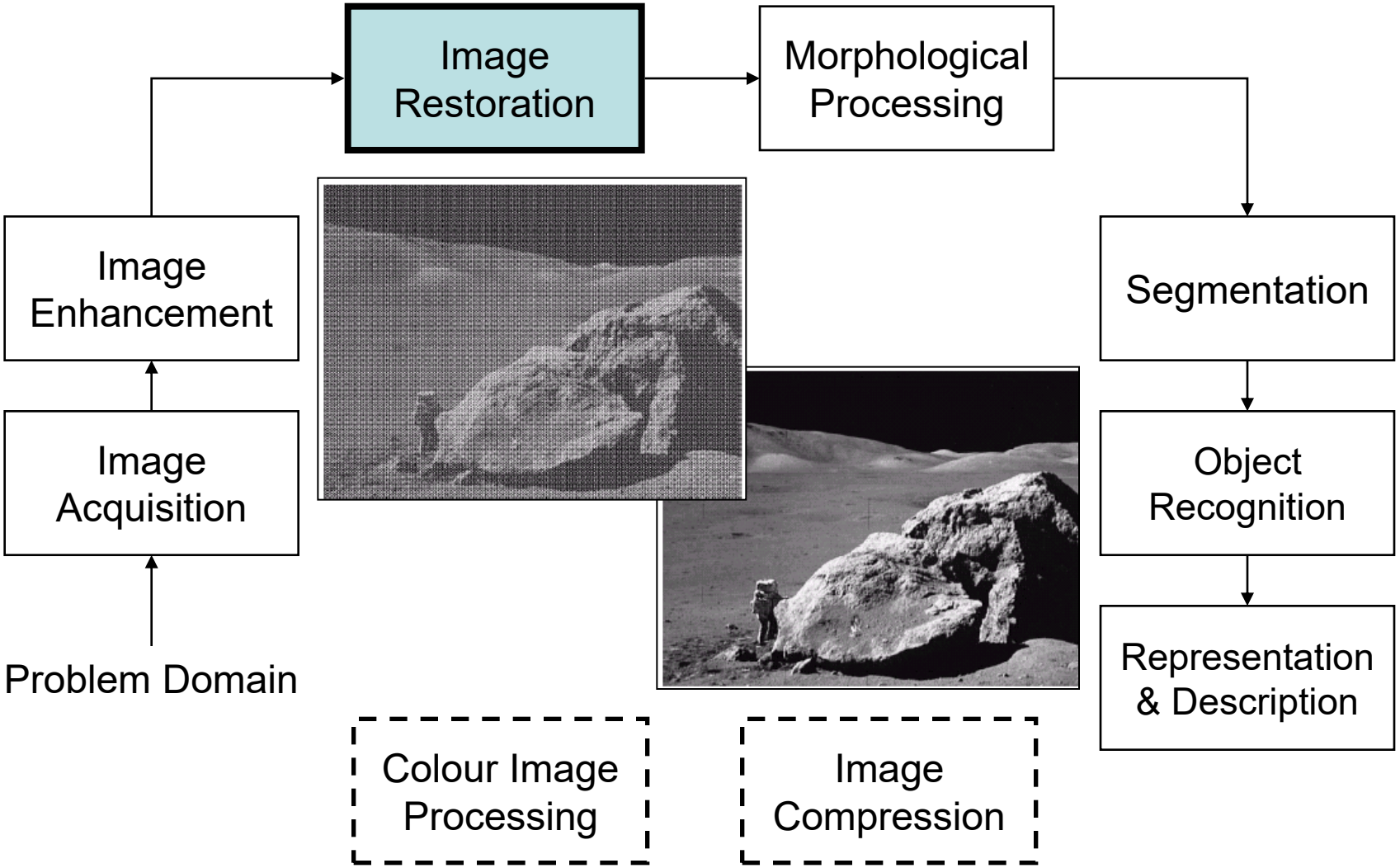
Key Stages in Digital Image Processing: Image Acquisition



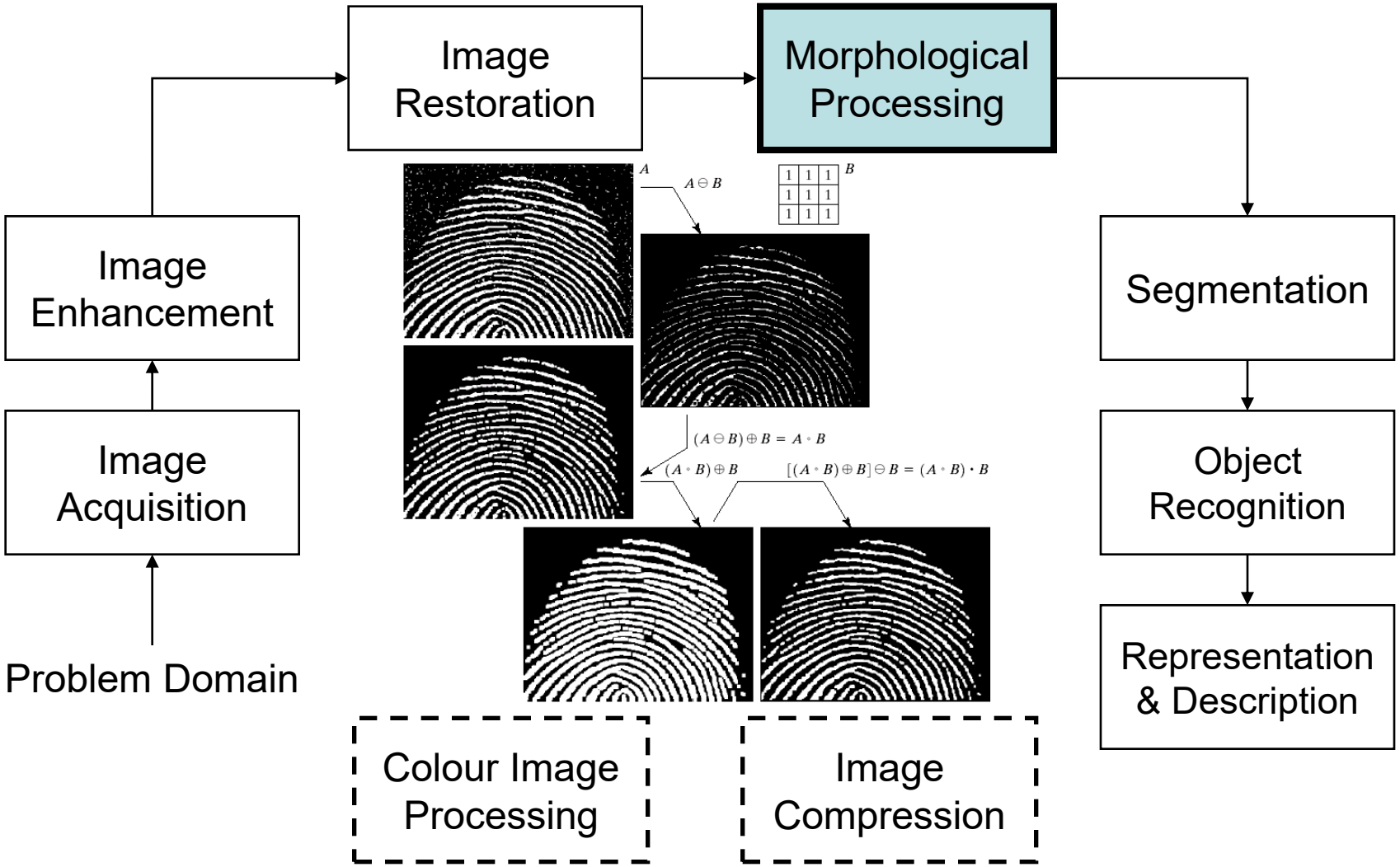
Key Stages in Digital Image Processing: Image Enhancement



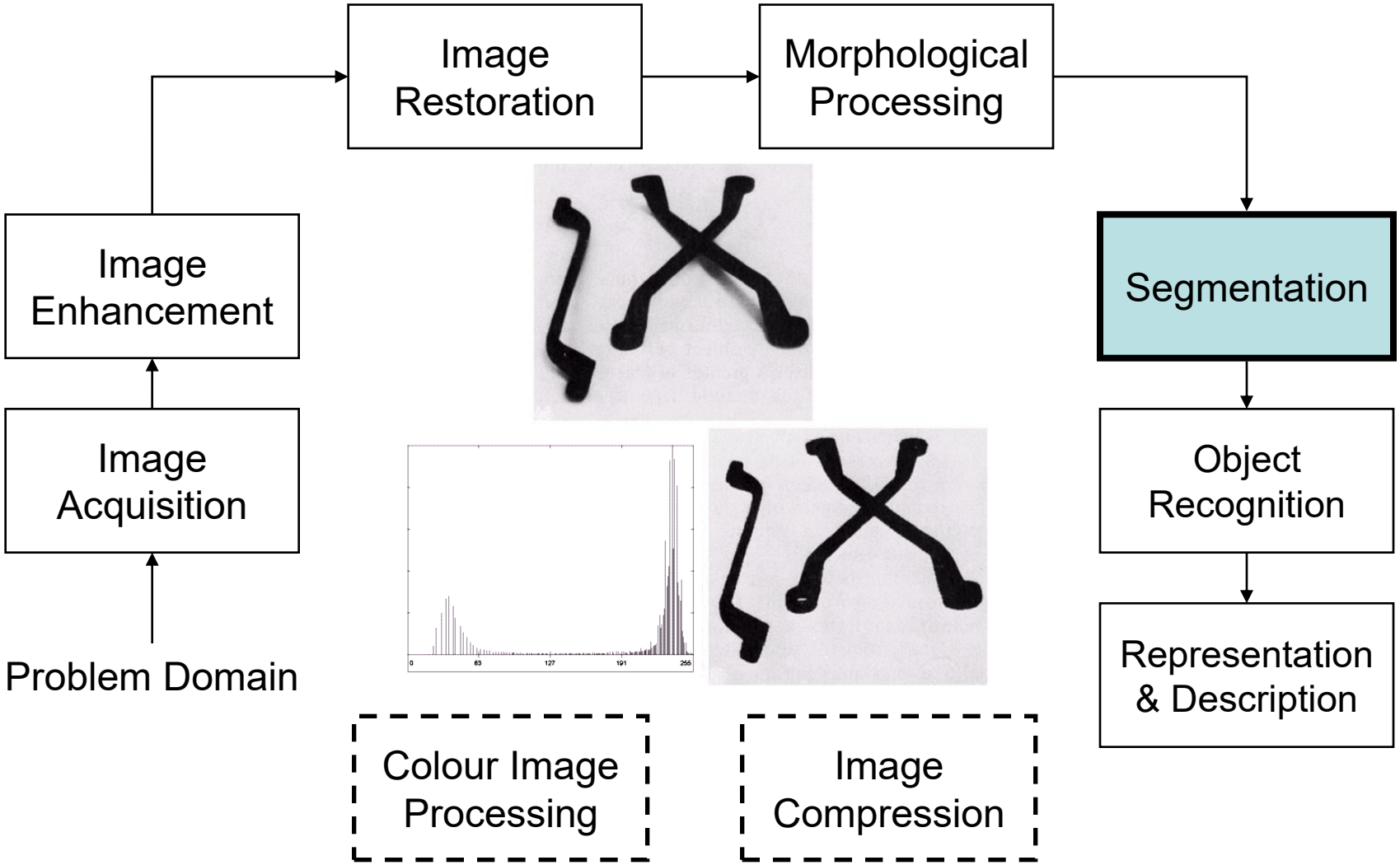
Key Stages in Digital Image Processing: Image Restoration



Key Stages in Digital Image Processing: Morphological Processing



Key Stages in Digital Image Processing: Segmentation



Key Stages in Digital Image Processing: Object Recognition

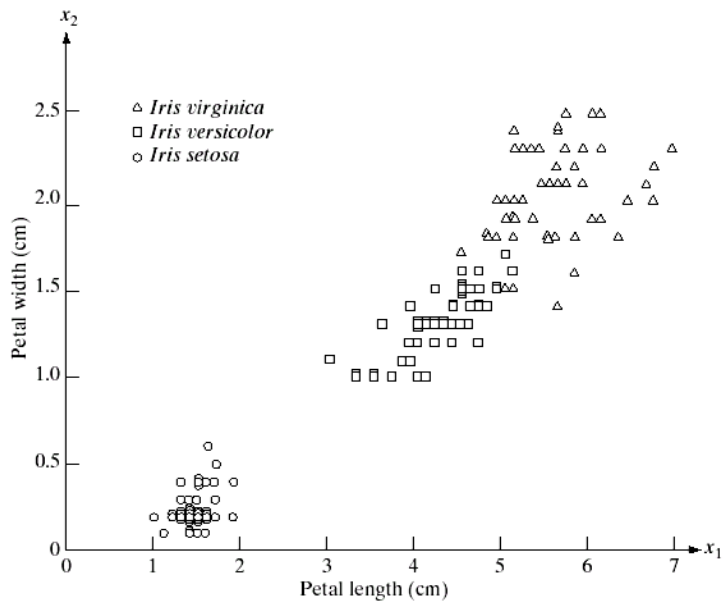
Image Restoration

Morphological Processing

Image Enhancement

Image Acquisition

Problem Domain



Segmentation

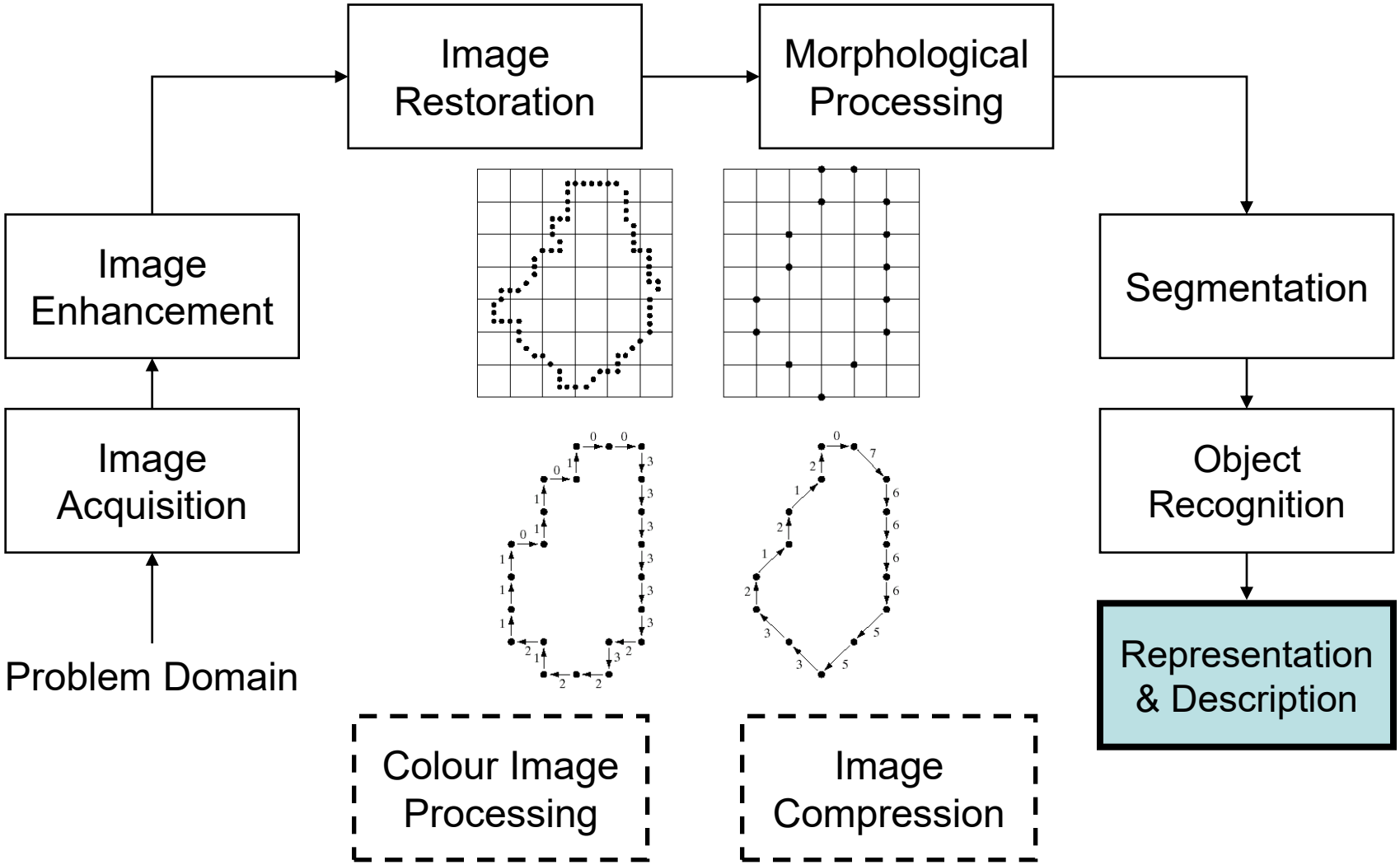
Object Recognition

Representation & Description

Colour Image Processing

Image Compression

Key Stages in Digital Image Processing: Representation & Description



Key Stages in Digital Image Processing: Image Compression

Image Restoration

Morphological Processing

Image Enhancement

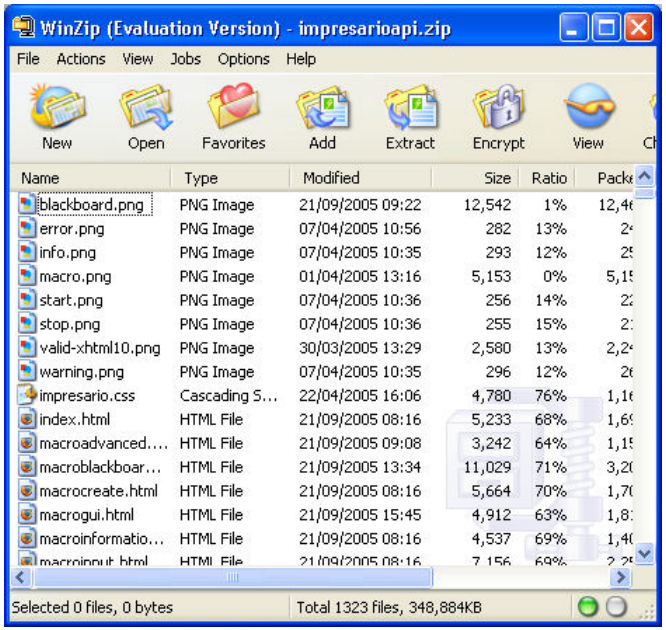
Segmentation

Image Acquisition

Object Recognition

Problem Domain

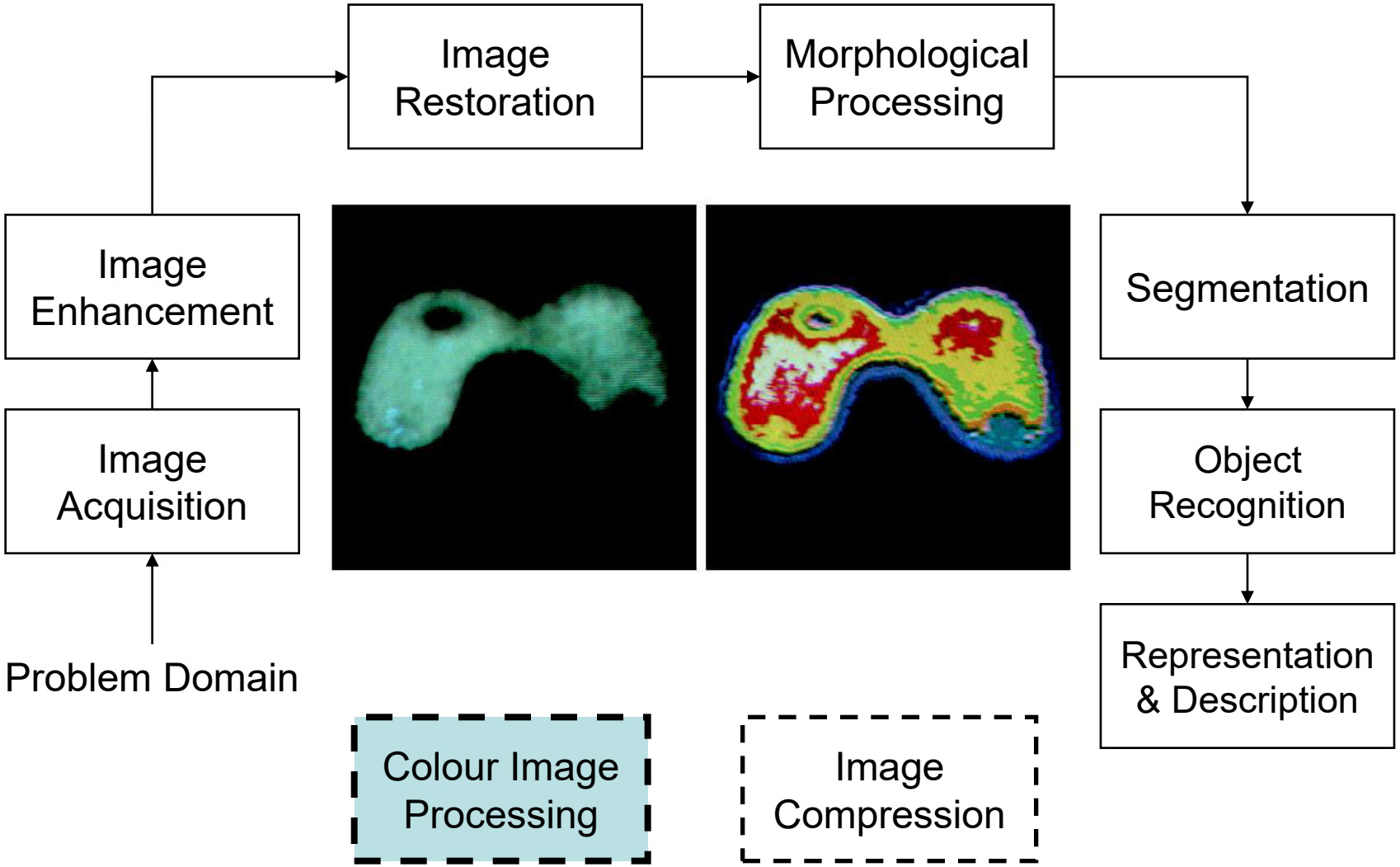
Representation & Description



Colour 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…

Digital Image Processing

Image Representation

Lecture-2

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Sampling, Quantisation And Resolution

- 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

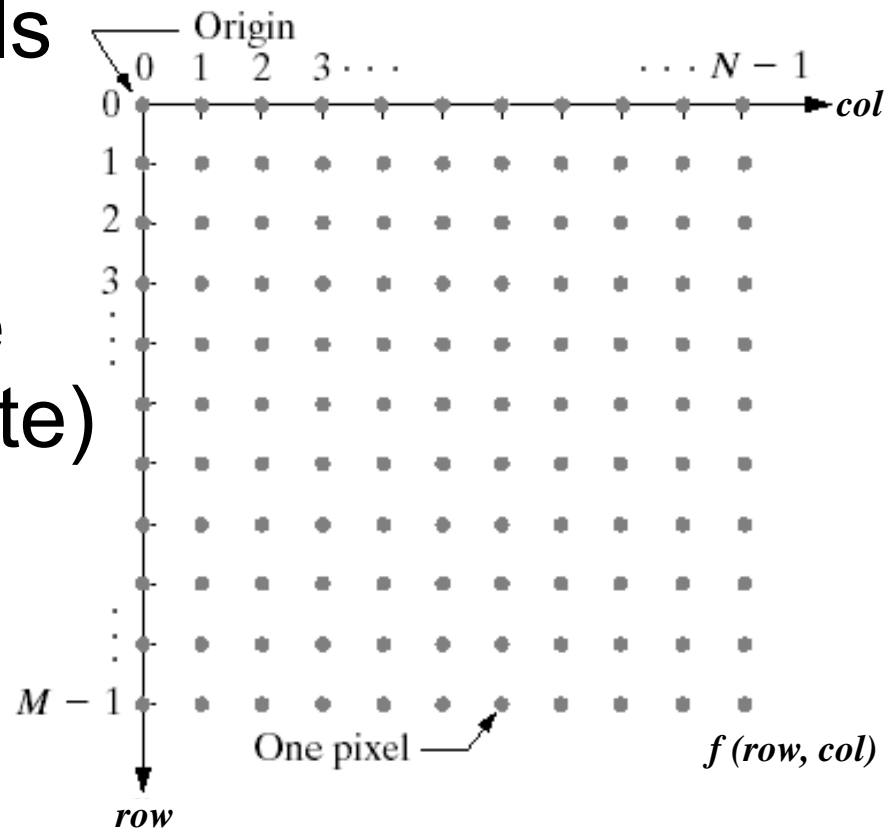
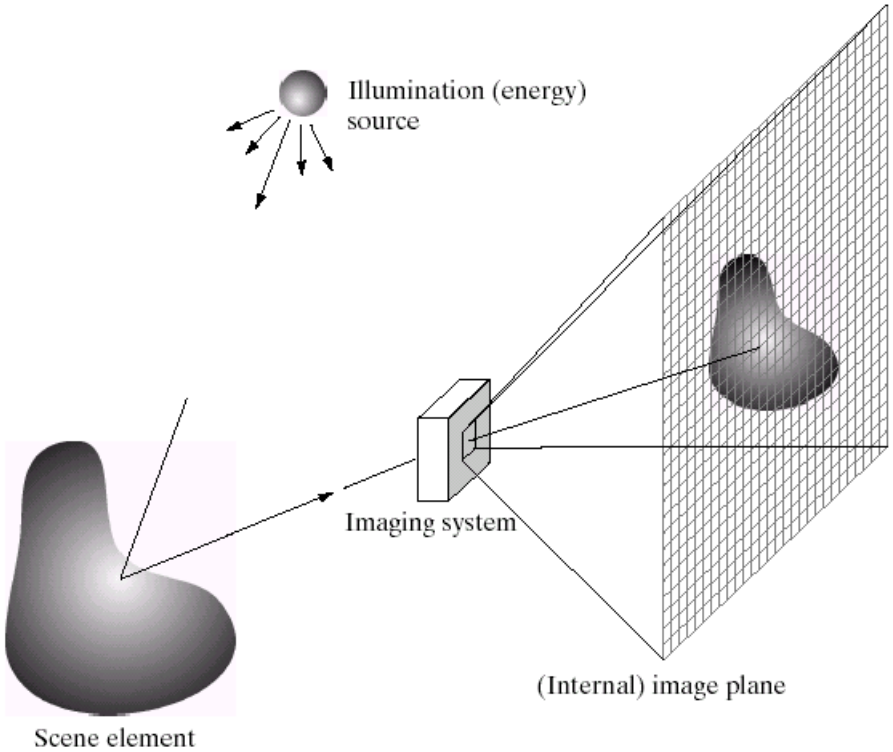


Image Acquisition

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

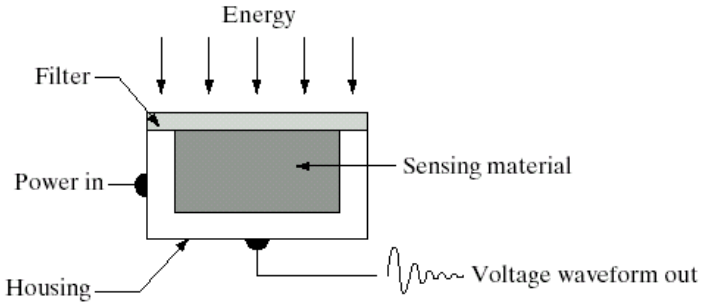


– Typical notions of illumination and scene can be way off:
  - X-rays of a skeleton
  - Ultrasound of an unborn baby
  - Electro-microscopic images of molecules

Image Sensing

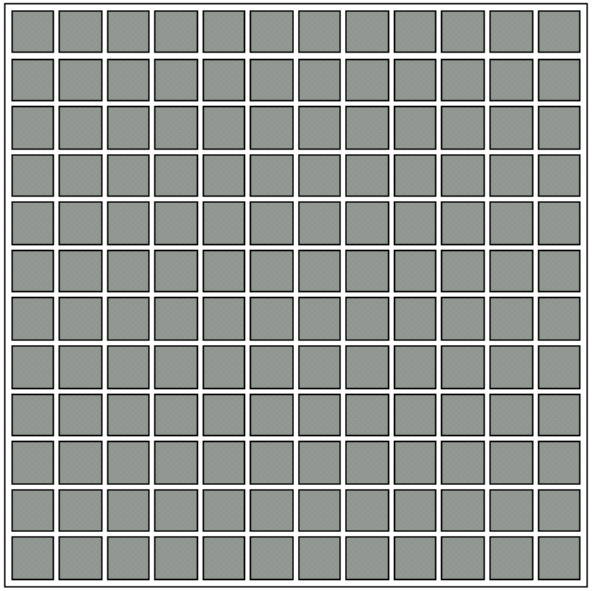
Incoming energy lands on a sensor material responsive to that type of energy and this generates a voltage

Collections of sensors are arranged to capture images



Imaging Sensor

Line of Image Sensors

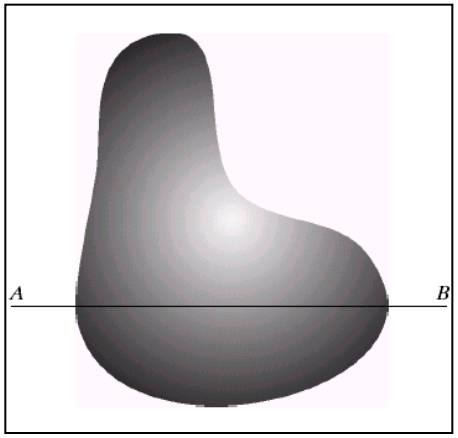


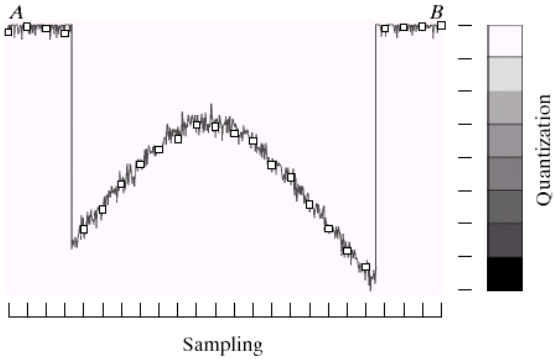
Array of Image Sensors

Image 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





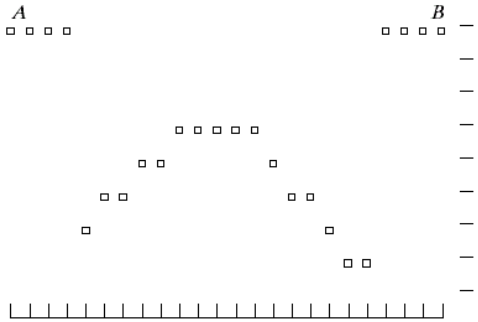


Image Sampling And Quantisation

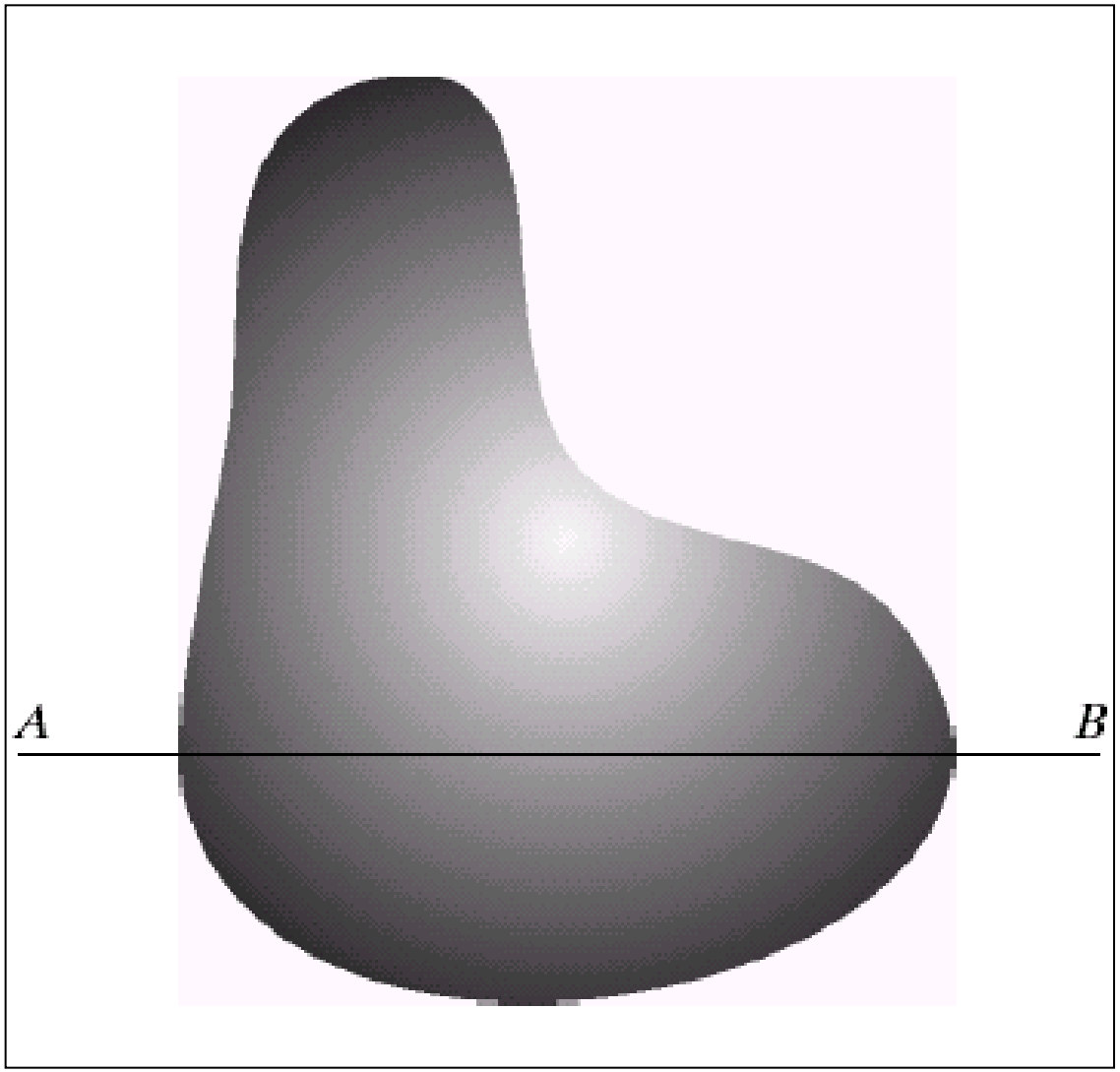
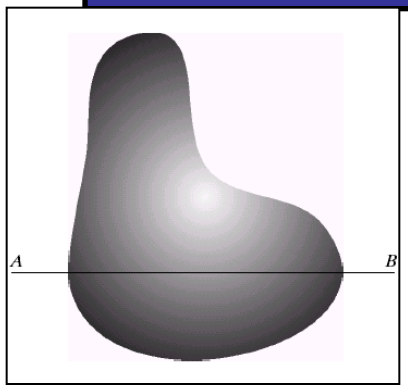
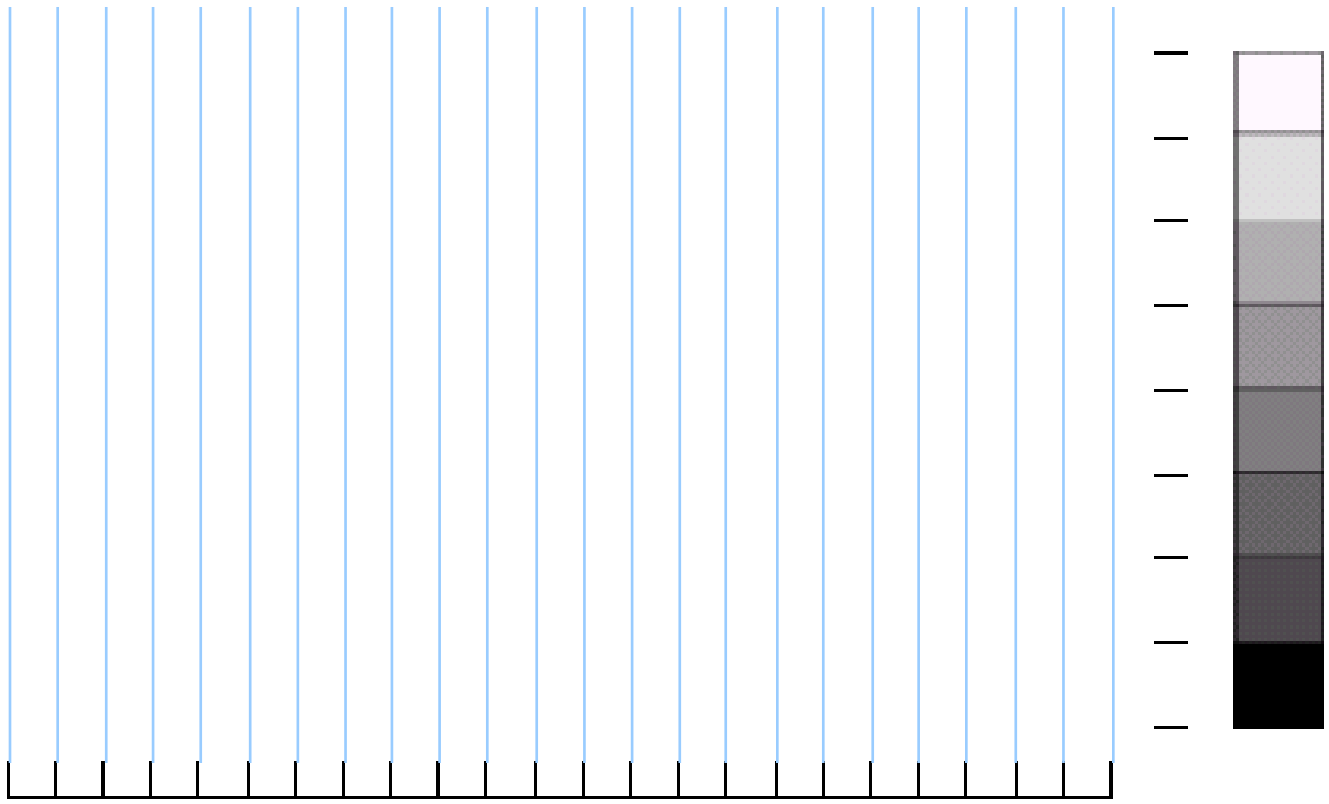


Image Sampling And Quantisation





Sampling

Image Sampling And Quantisation (cont…)

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

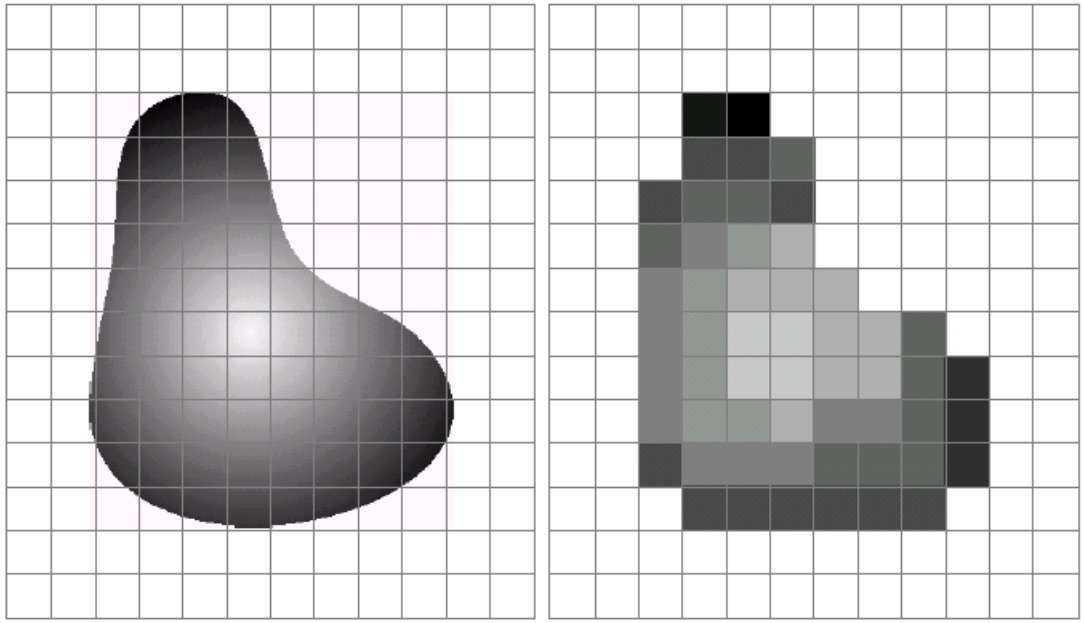
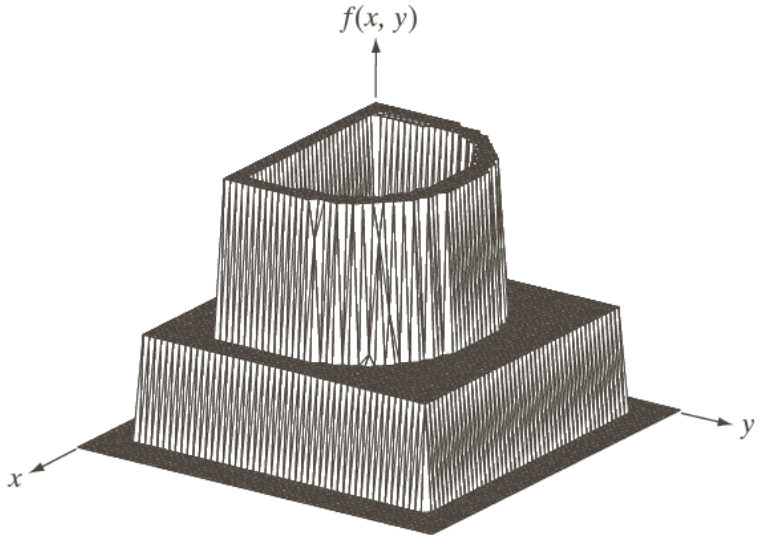
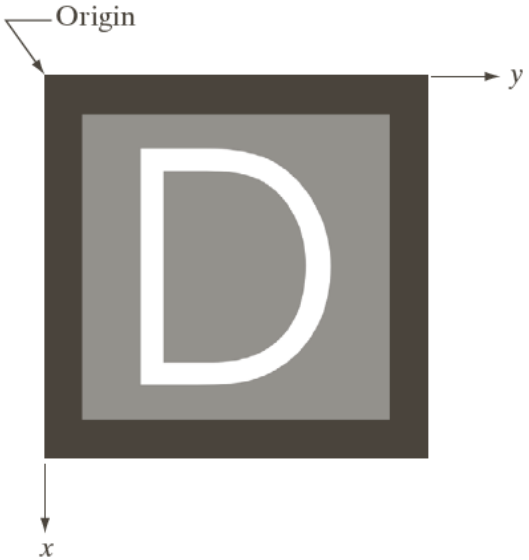


Image Representation





Origin

$$
\begin{array}{ccccccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & & & & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & & & & & & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & & \vdots & & & & & & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \cdots & .5 & .5 & .5 & \cdots & & & & & 0 & 0 & 0 \\
0 & 0 & 0 & & .5 & .5 & & & & & & & 0 & 0 & 0 \\
\vdots & & & & .5 & & \ddots & & & & & & & & \vdots \\
\vdots & & & & \vdots & & & 1 & 1 & 1 & \cdots & & & & \vdots \\
 & & & & & & & 1 & 1 & & & & & & \\
0 & 0 & 0 & & & & & 1 & & \ddots & & & 0 & 0 & 0 \\
0 & 0 & 0 & & & & & \vdots & & & & & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & & & & & & & & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & & & & & & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & & & & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Image Representation

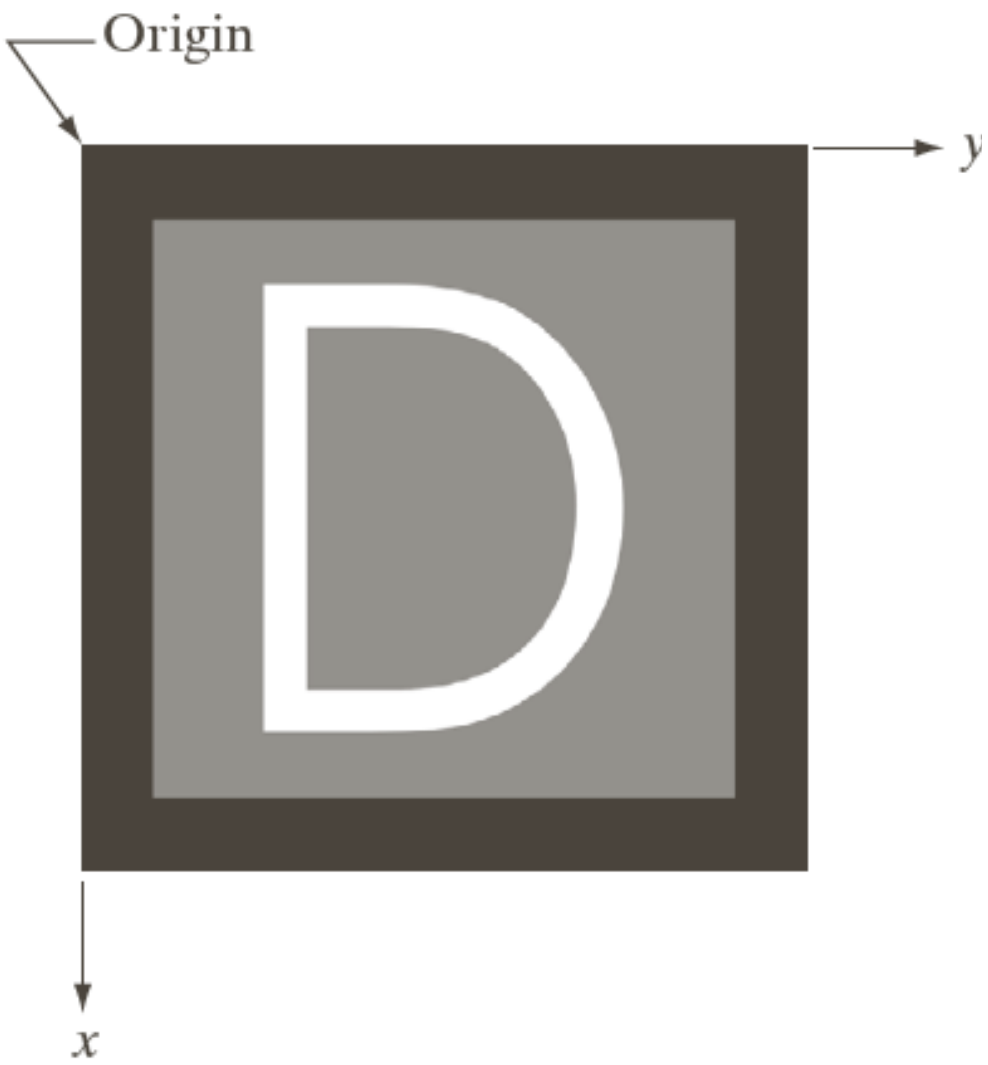


Image Representation

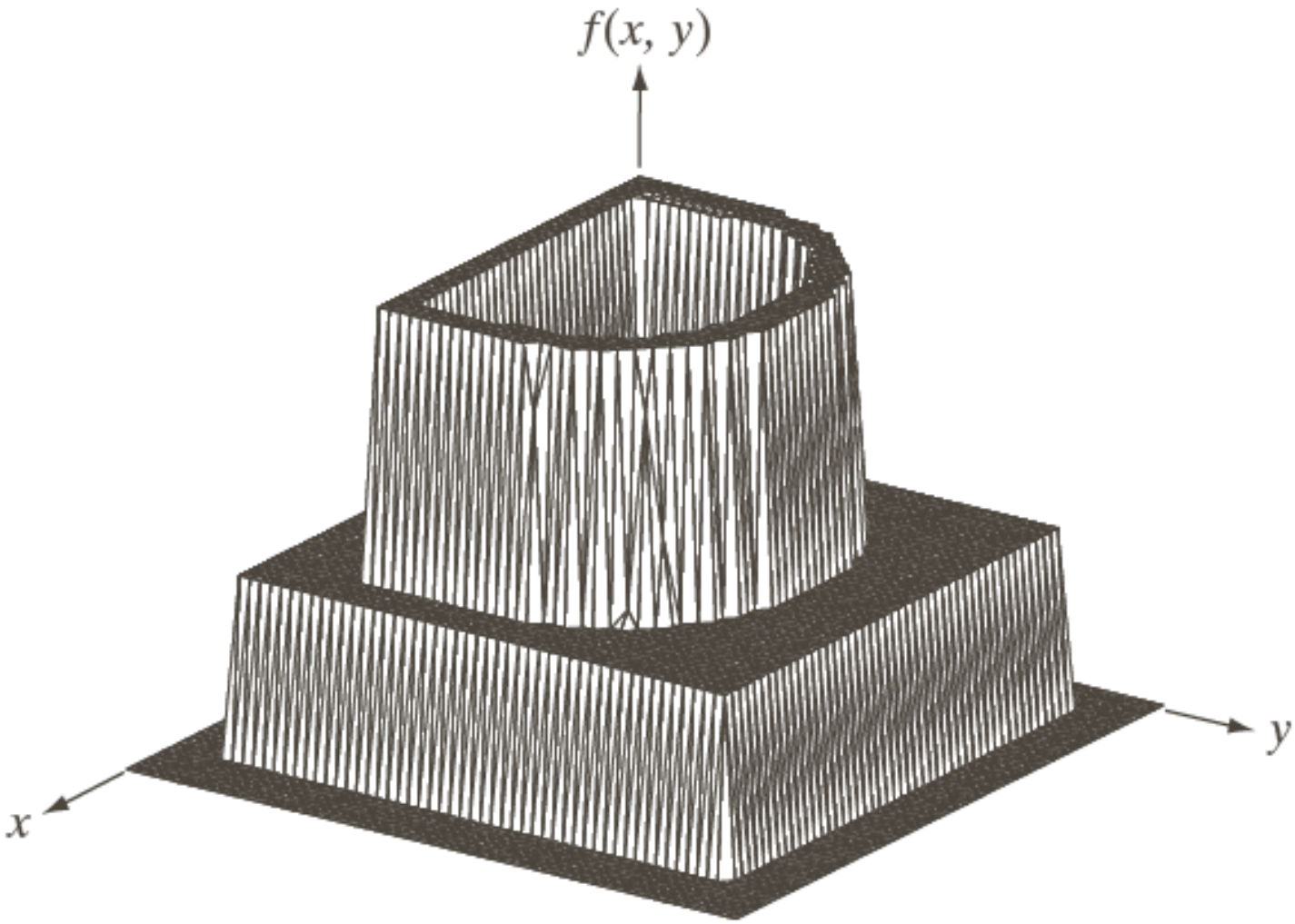
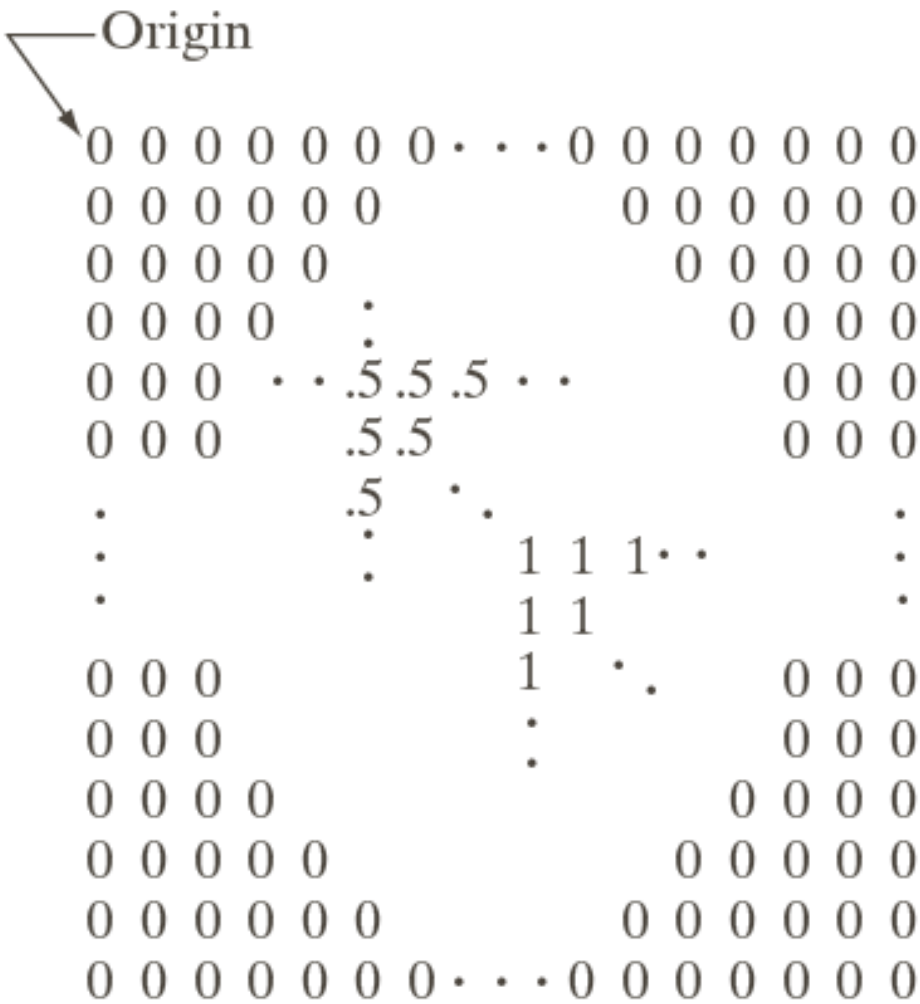


Image Representation



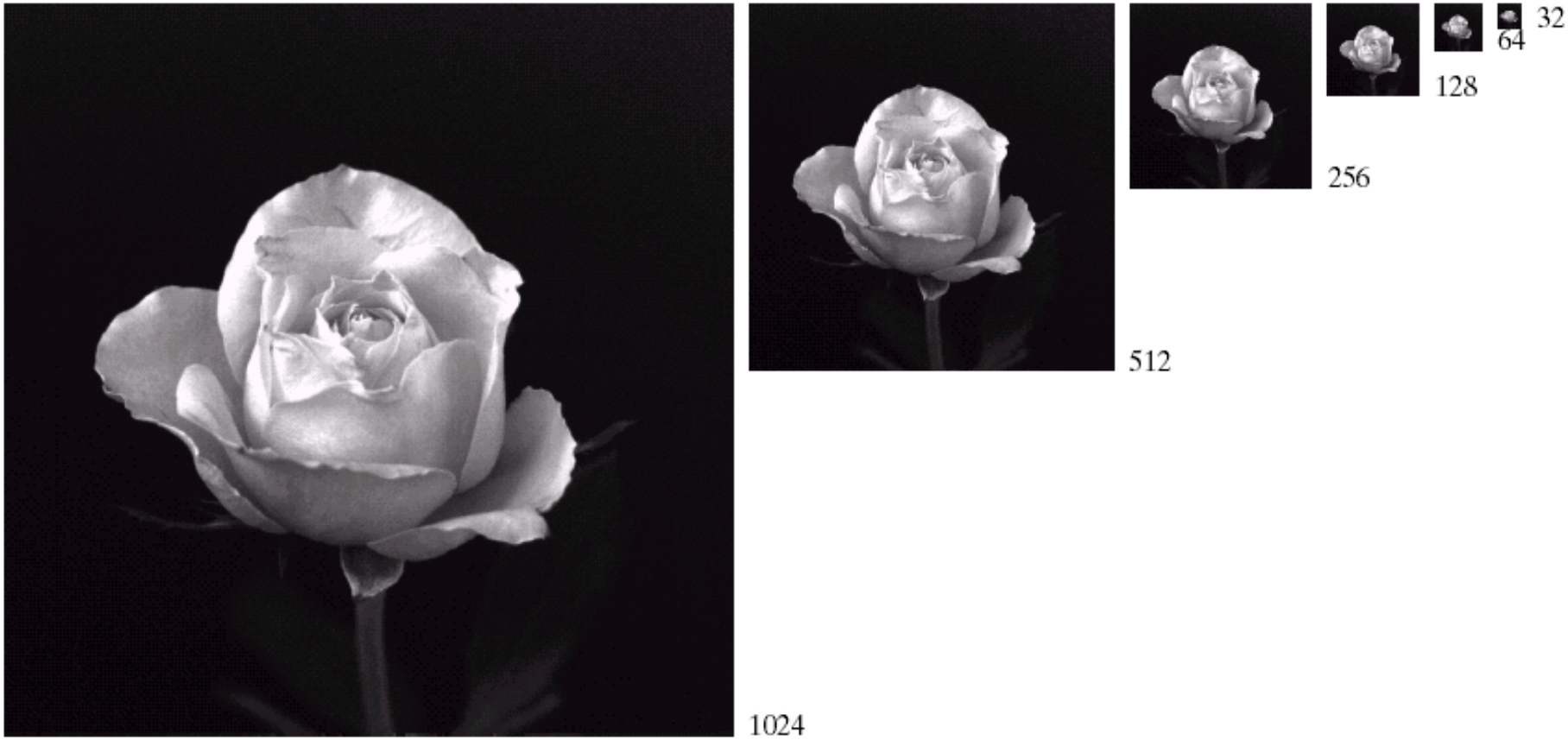
Spatial Resolution

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 (cont…)

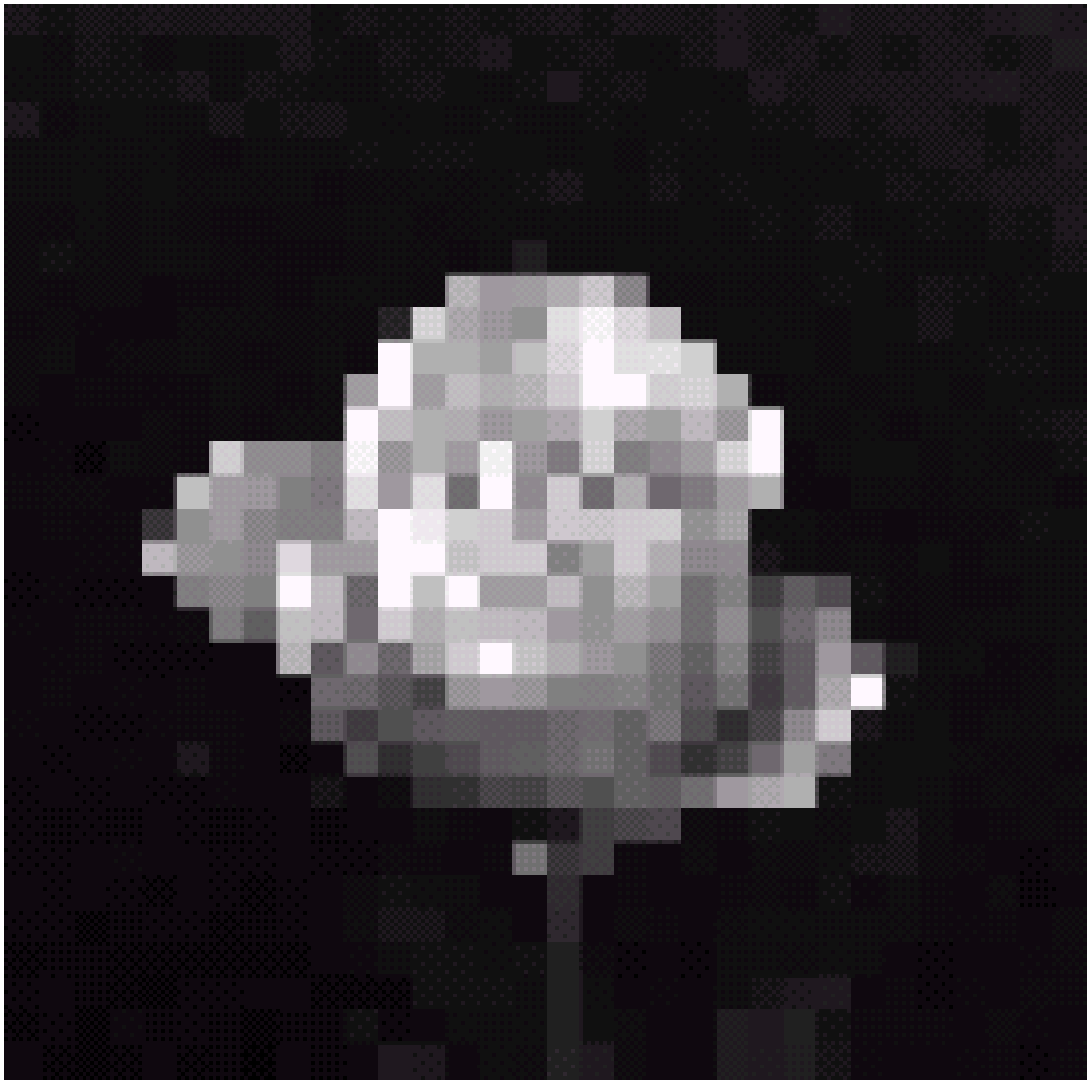
Spatial Resolution (cont…)

Spatial Resolution (cont…)

Spatial Resolution (cont…)

Spatial Resolution (cont…)

Spatial Resolution (cont…)



Intensity Level Resolution

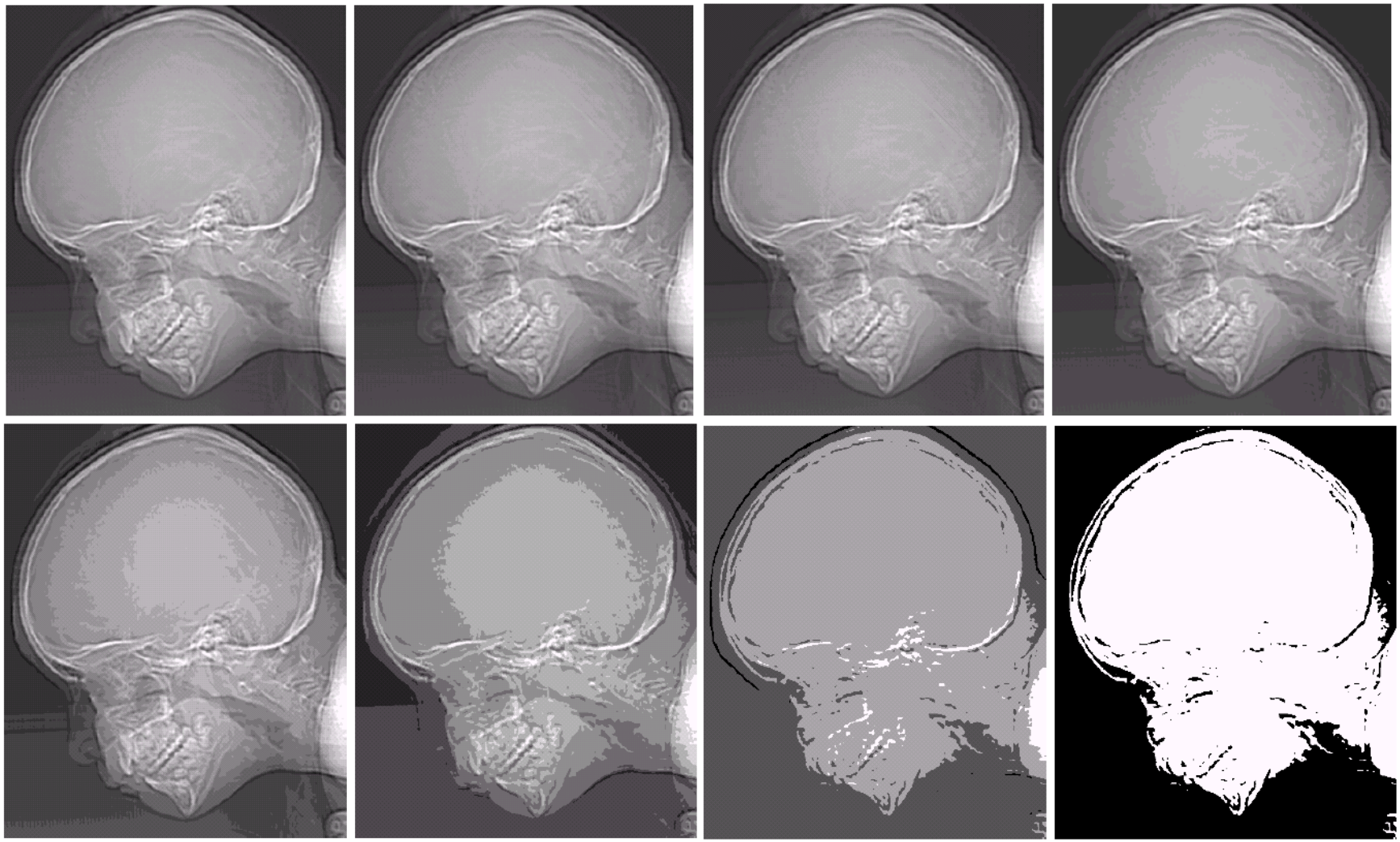
Intensity level resolution refers to the number of intensity levels used to represent the image

- 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 (cont…)

Intensity Level Resolution (cont…)

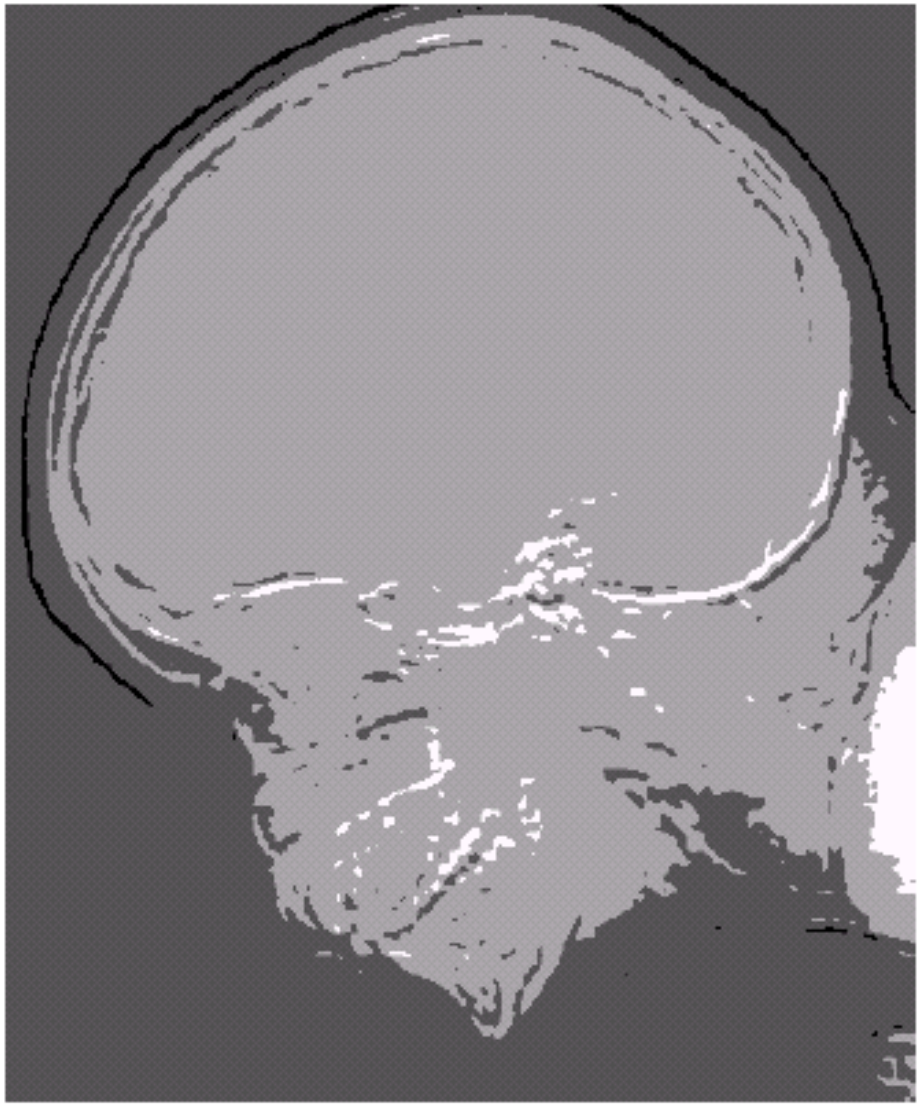
Intensity Level Resolution (cont…)

Intensity Level Resolution (cont…)

Intensity Level Resolution (cont…)

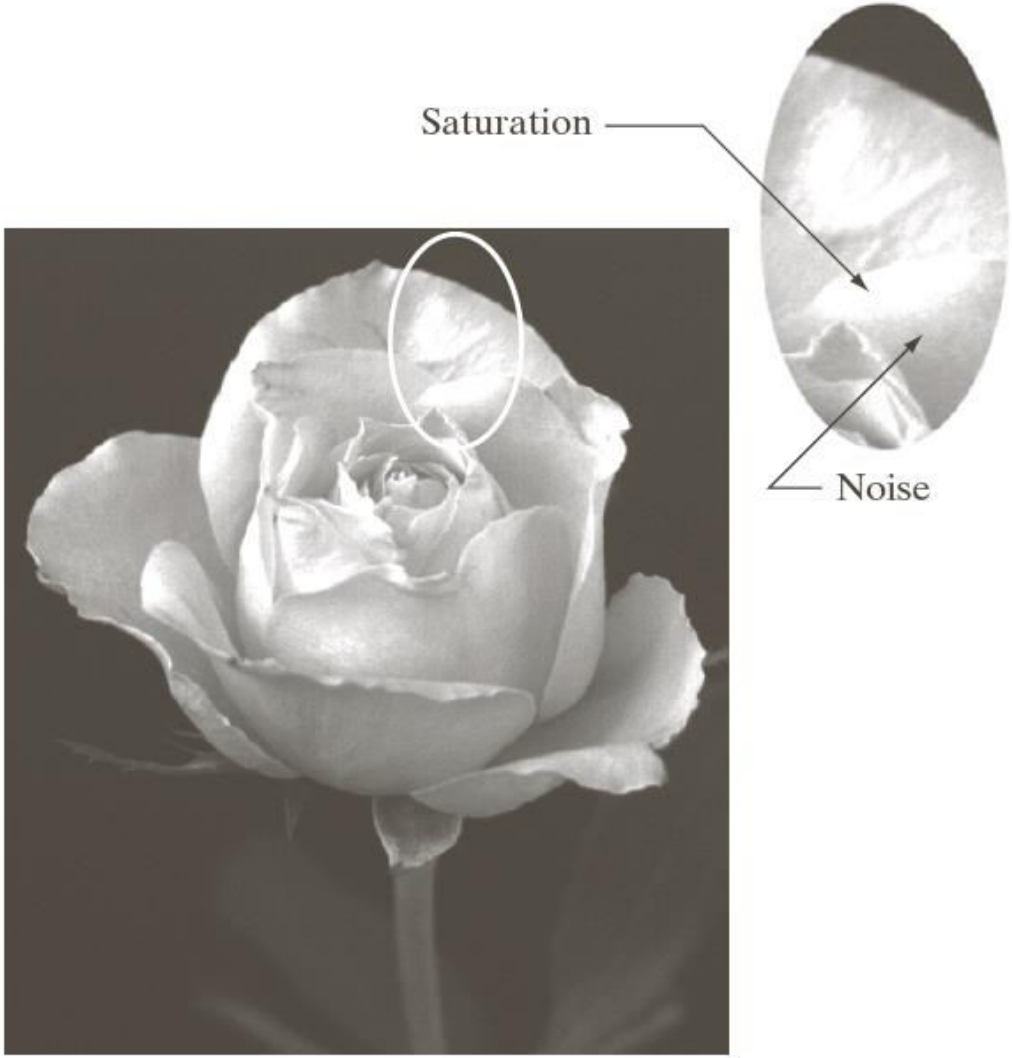
Intensity Level Resolution (cont…)

Intensity Level Resolution (cont…)



Intensity Level Resolution (cont…)

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…)

Summary

We have looked at:

- Image representation
- Image sensing and acquisition
- Sampling, quantisation and resolution

Next time we start to look at techniques for image enhancement

Digital Image Processing

Image Enhancement

(Histogram Processing)

Lecture - 3

•Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

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
- Histogram processing
- Point 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…)

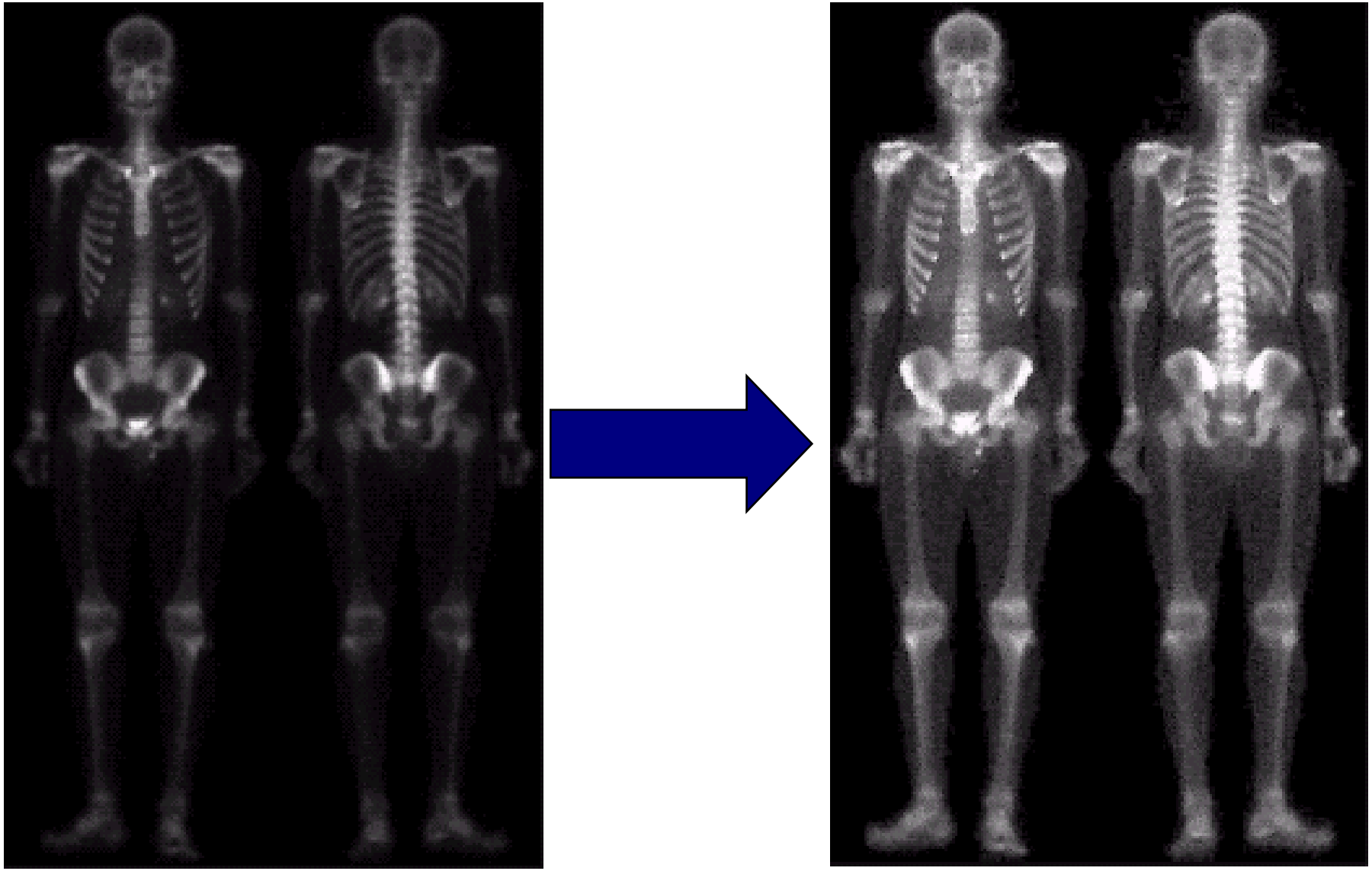
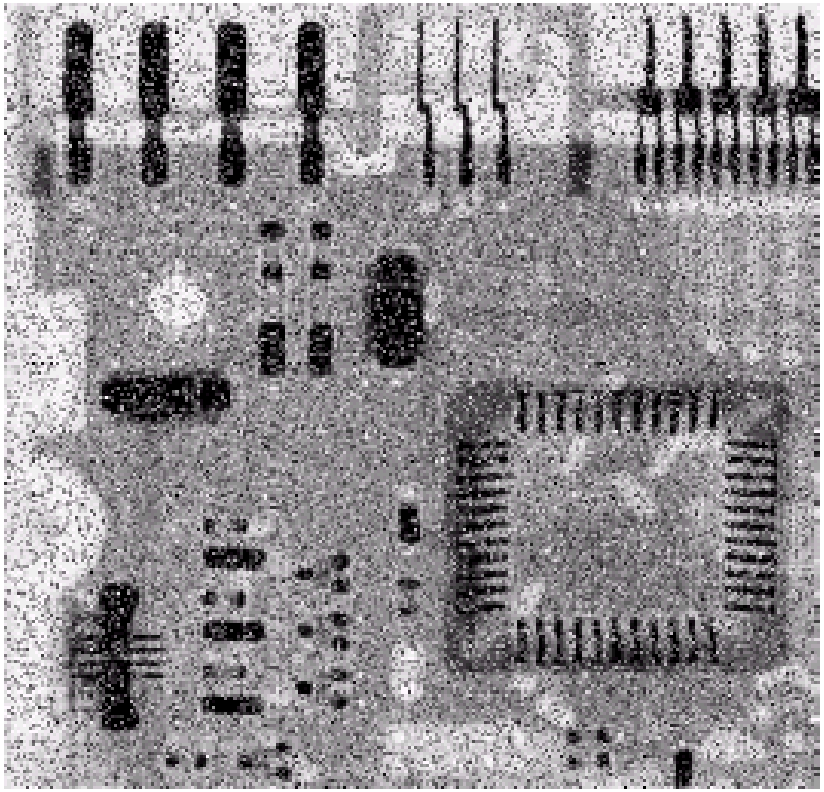


Image Enhancement Examples (cont…)



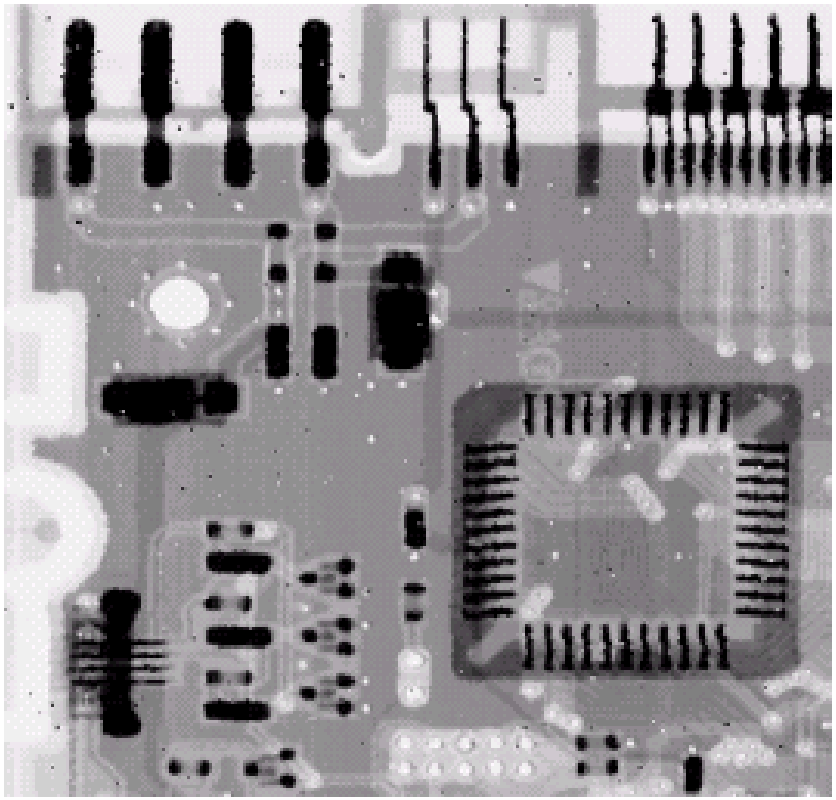


Image Enhancement Examples (cont…)

Spatial & Frequency Domains

There are two broad categories of image enhancement techniques

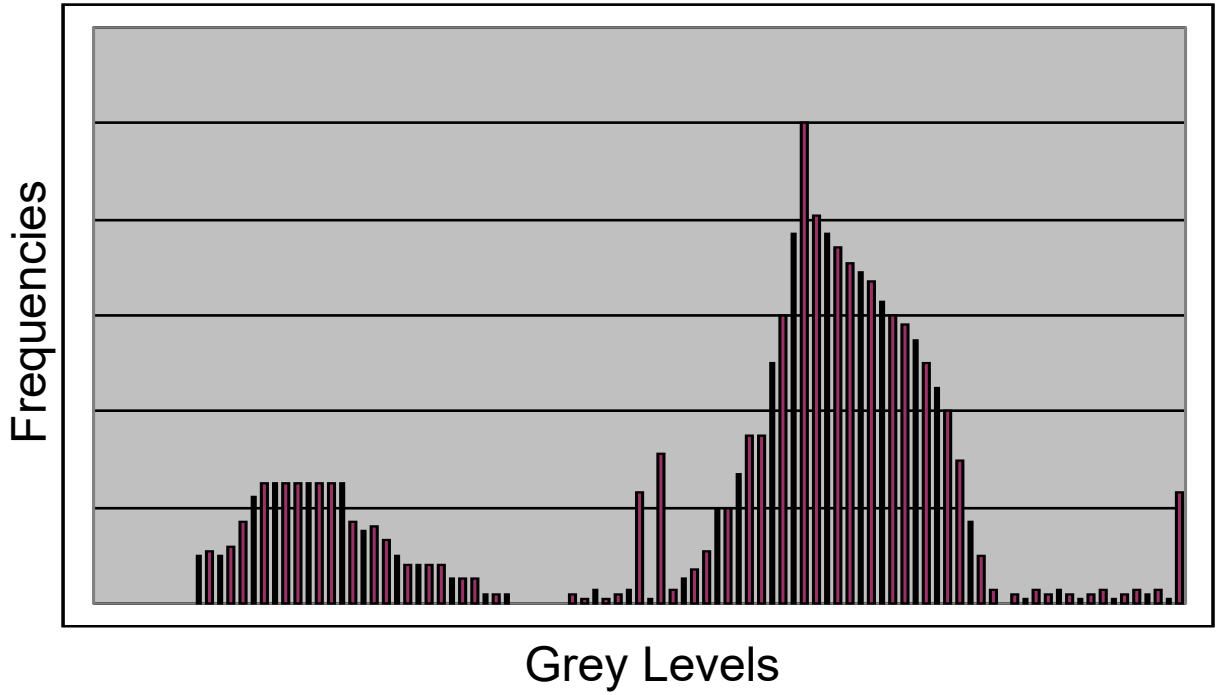
- Spatial domain techniques
  - Direct manipulation of image pixels
- Frequency domain techniques
  - Manipulation of Fourier transform

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

Image Histograms

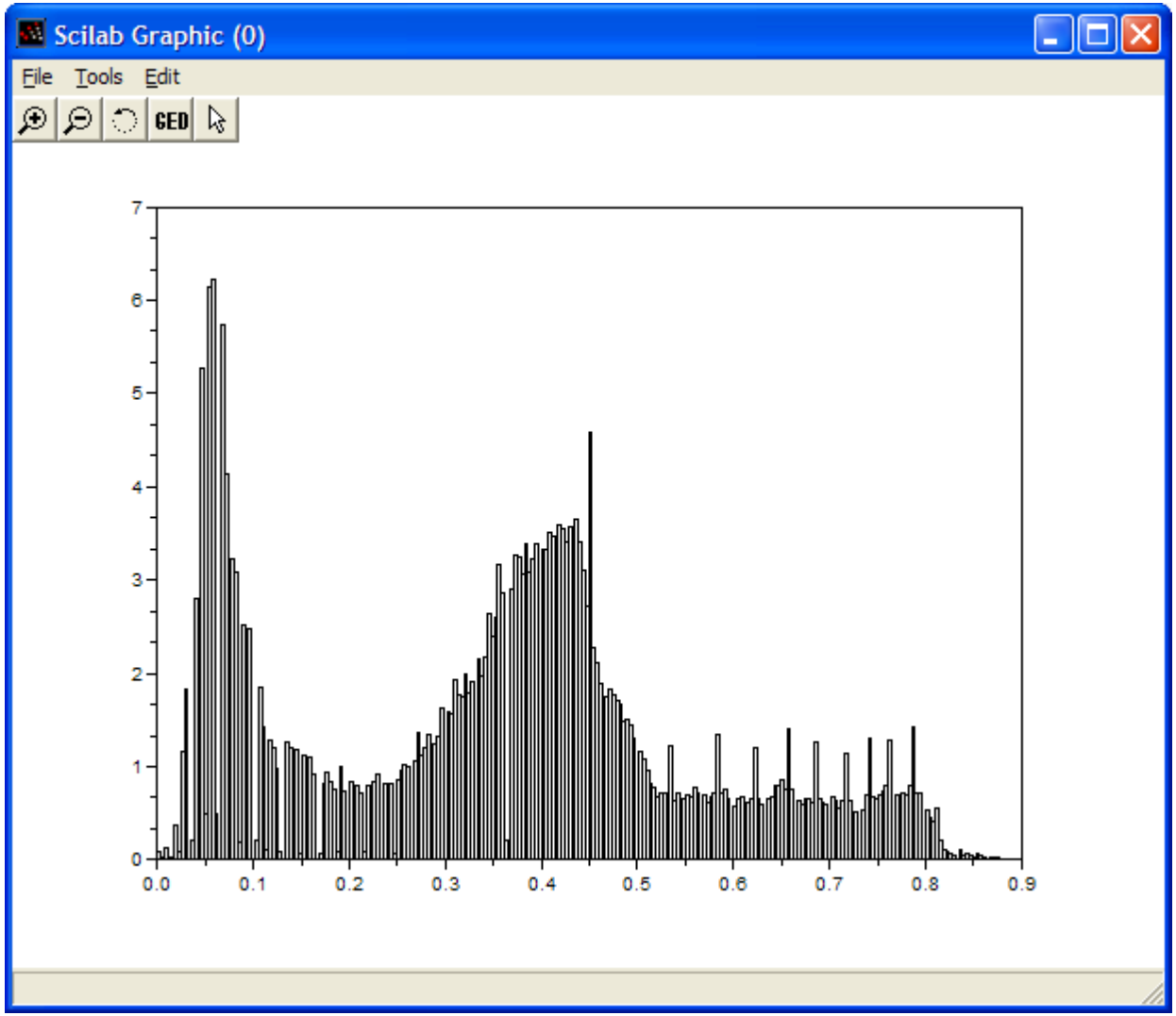
The histogram of an image shows us the distribution of grey levels in the image

Massively useful in image processing, especially in segmentation

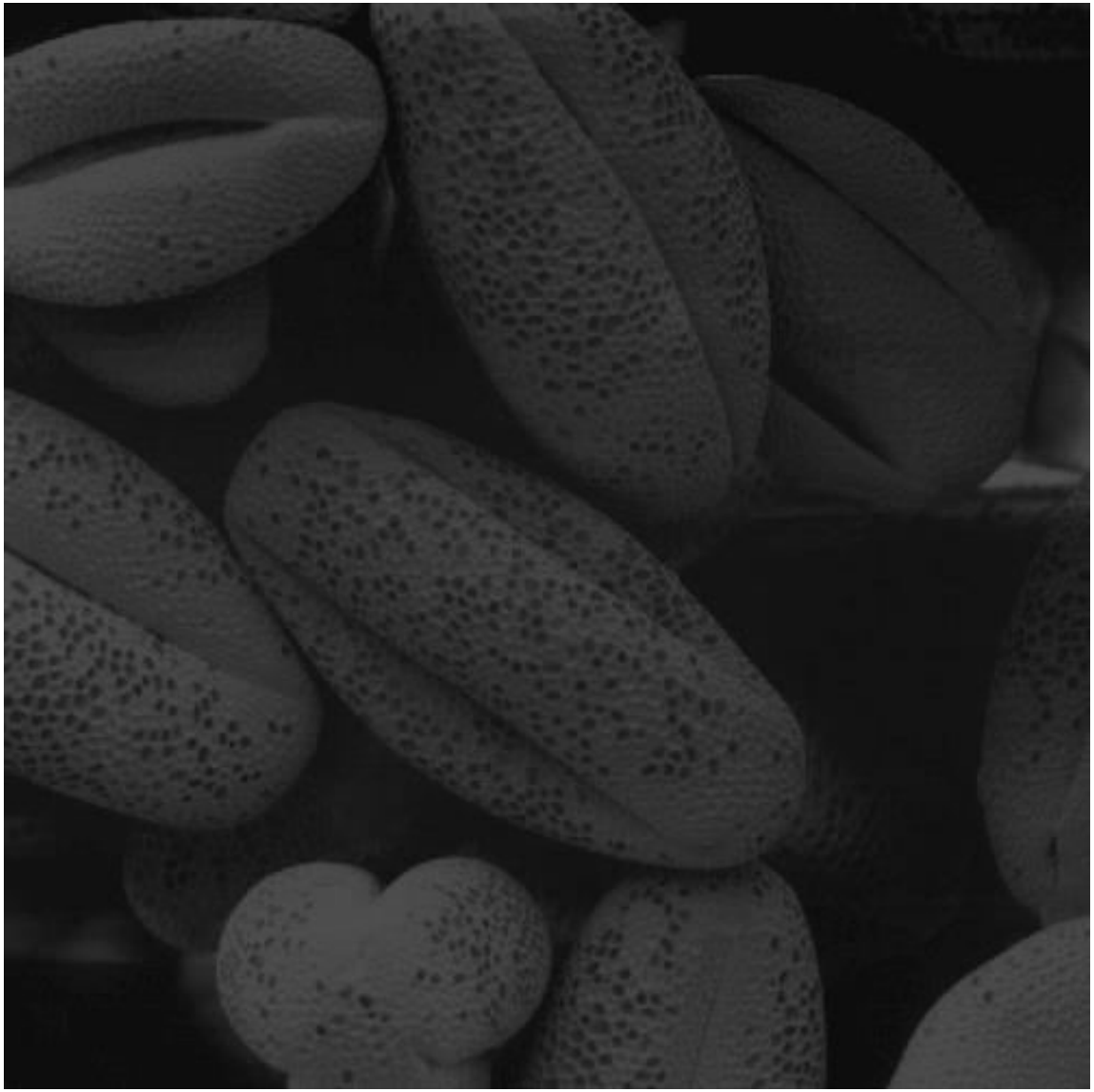


Histogram Examples

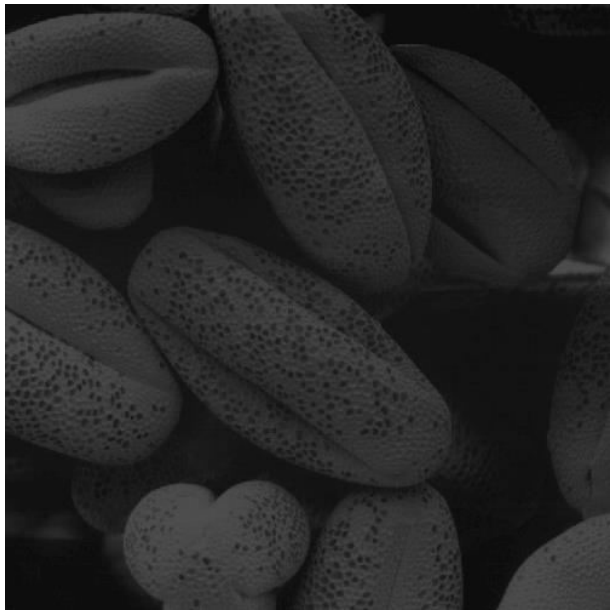
Histogram Examples (cont…)

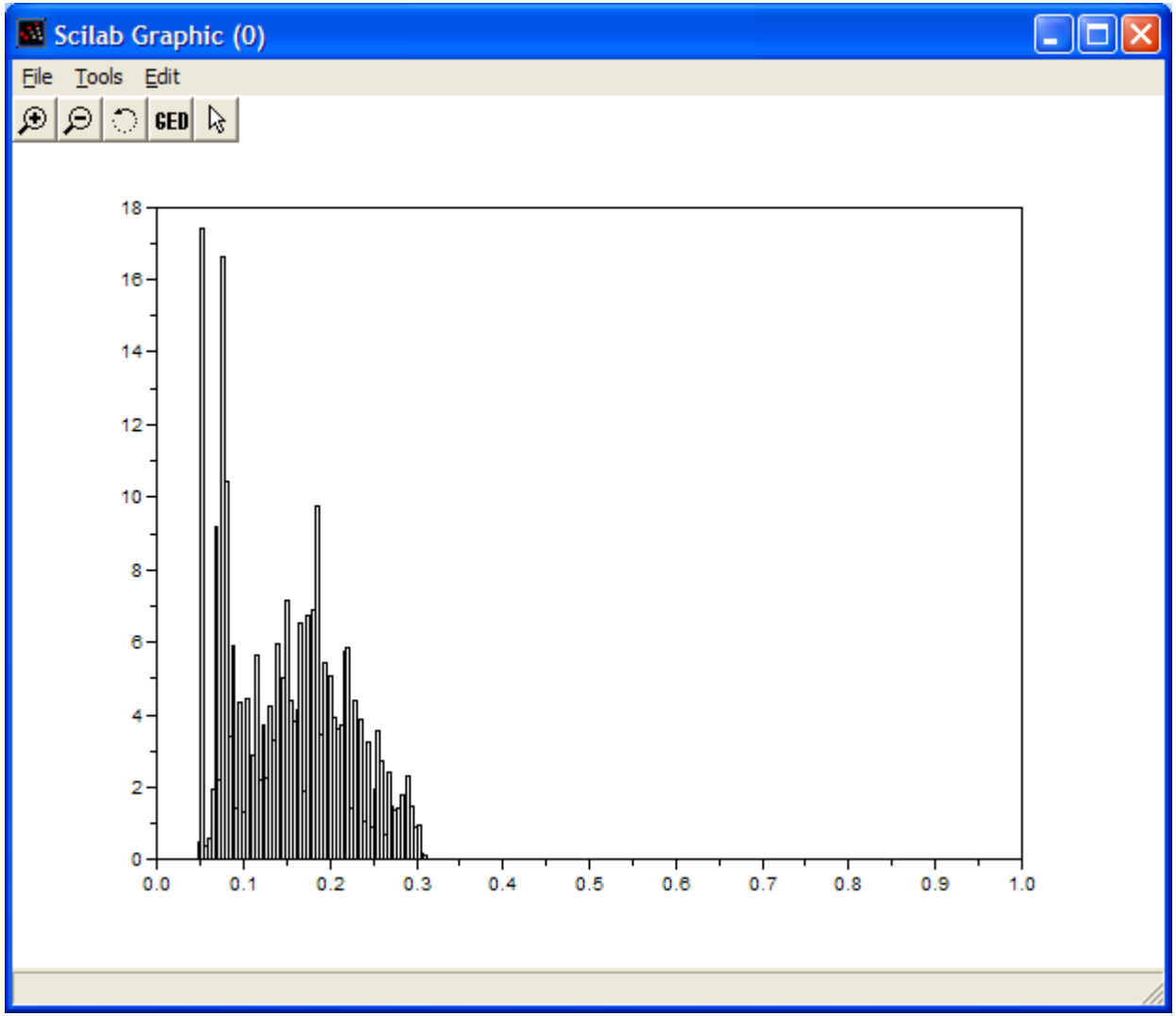


Histogram Examples (cont…)



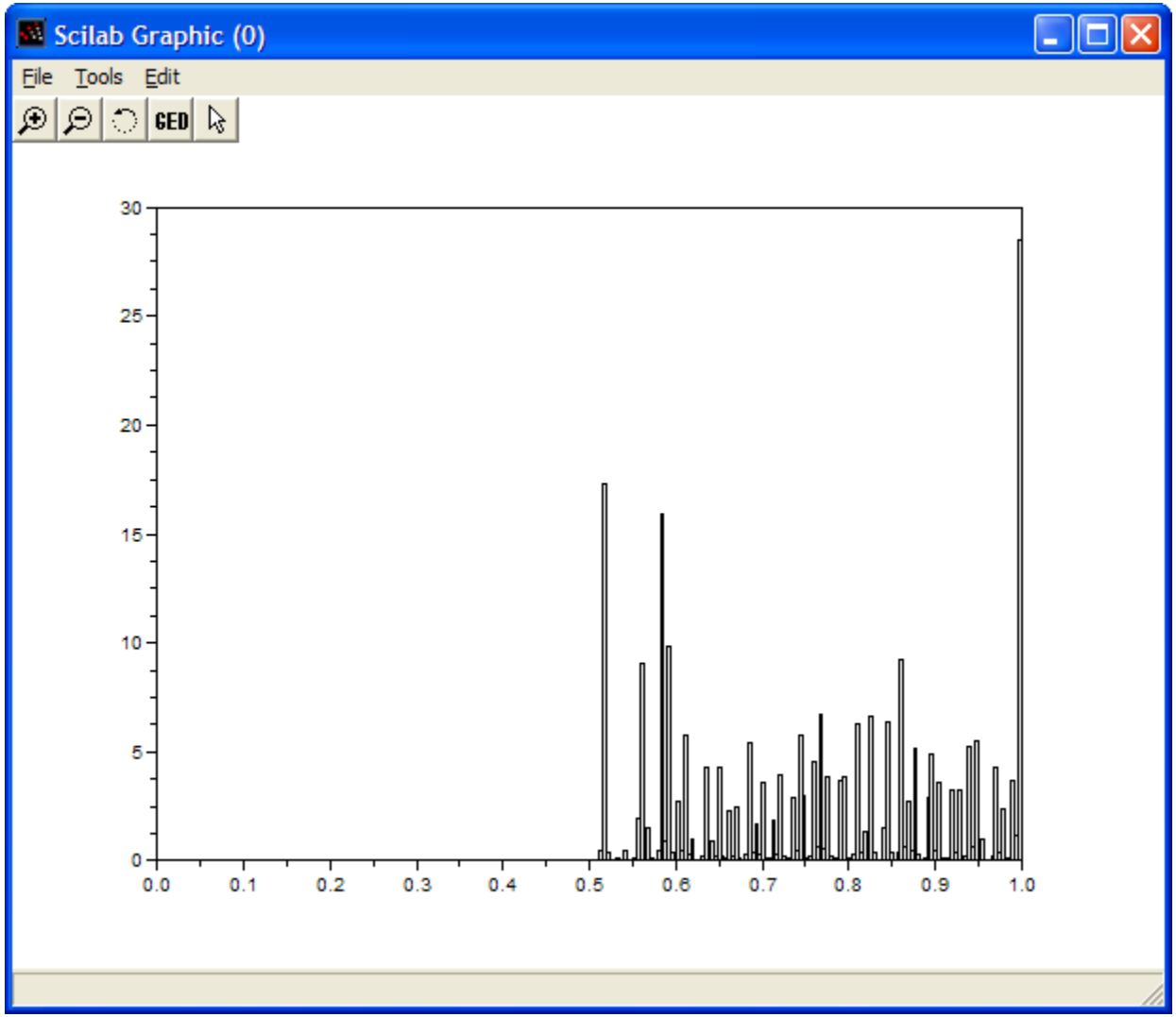
Histogram Examples (cont…)



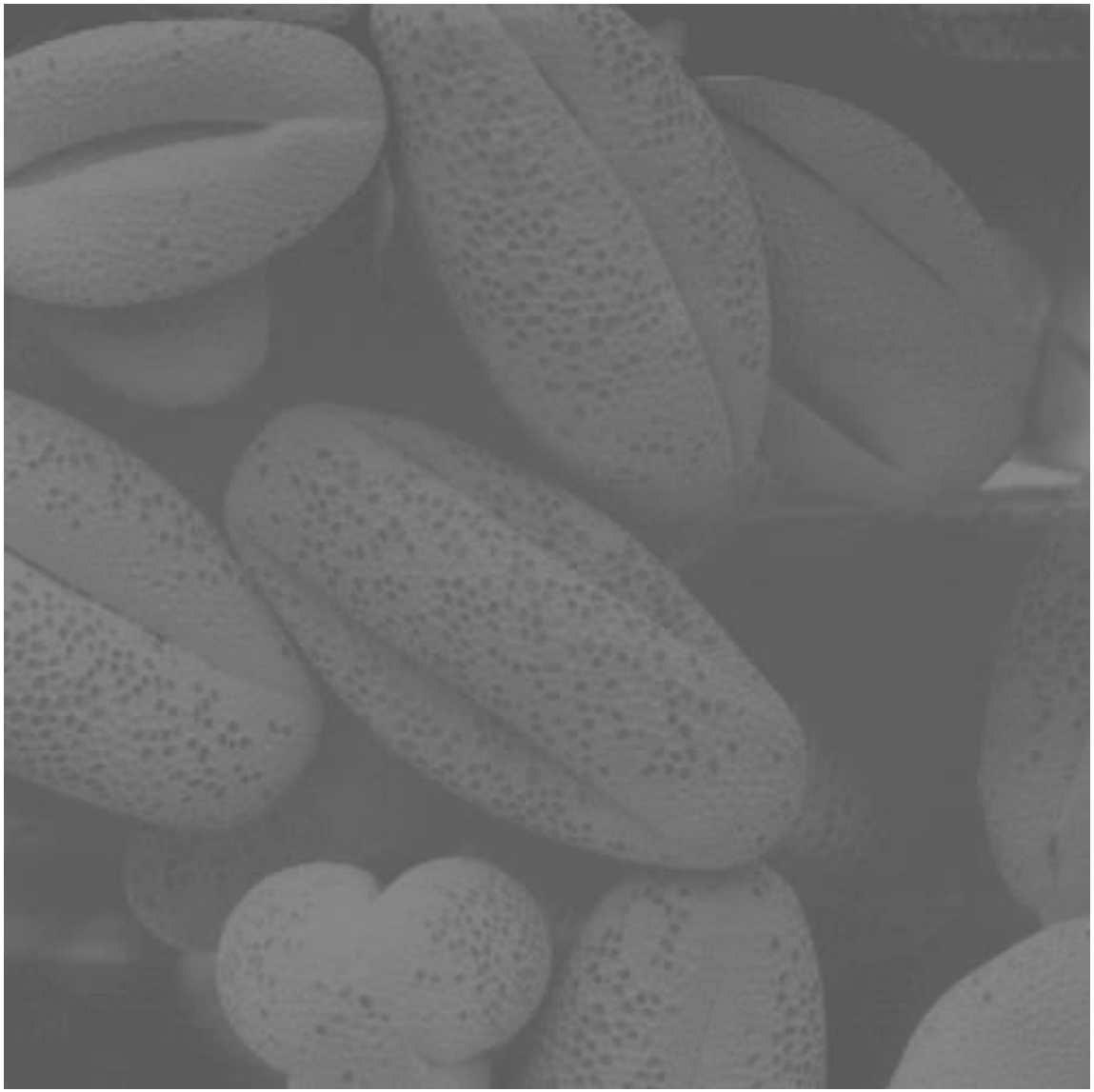


Histogram Examples (cont…)

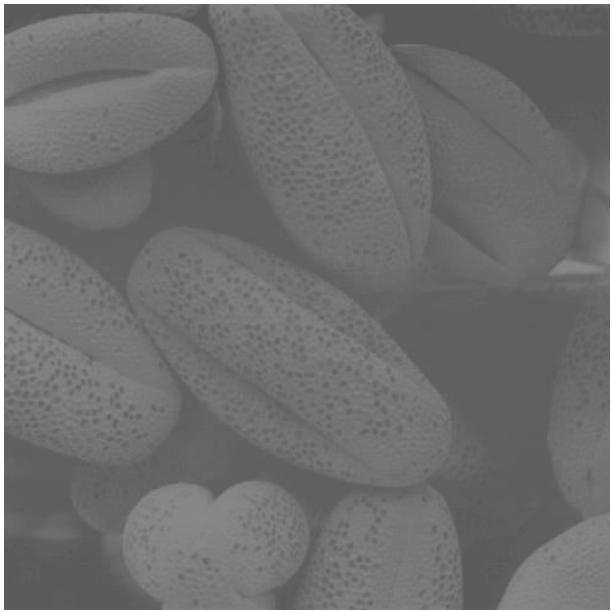
Histogram Examples (cont…)

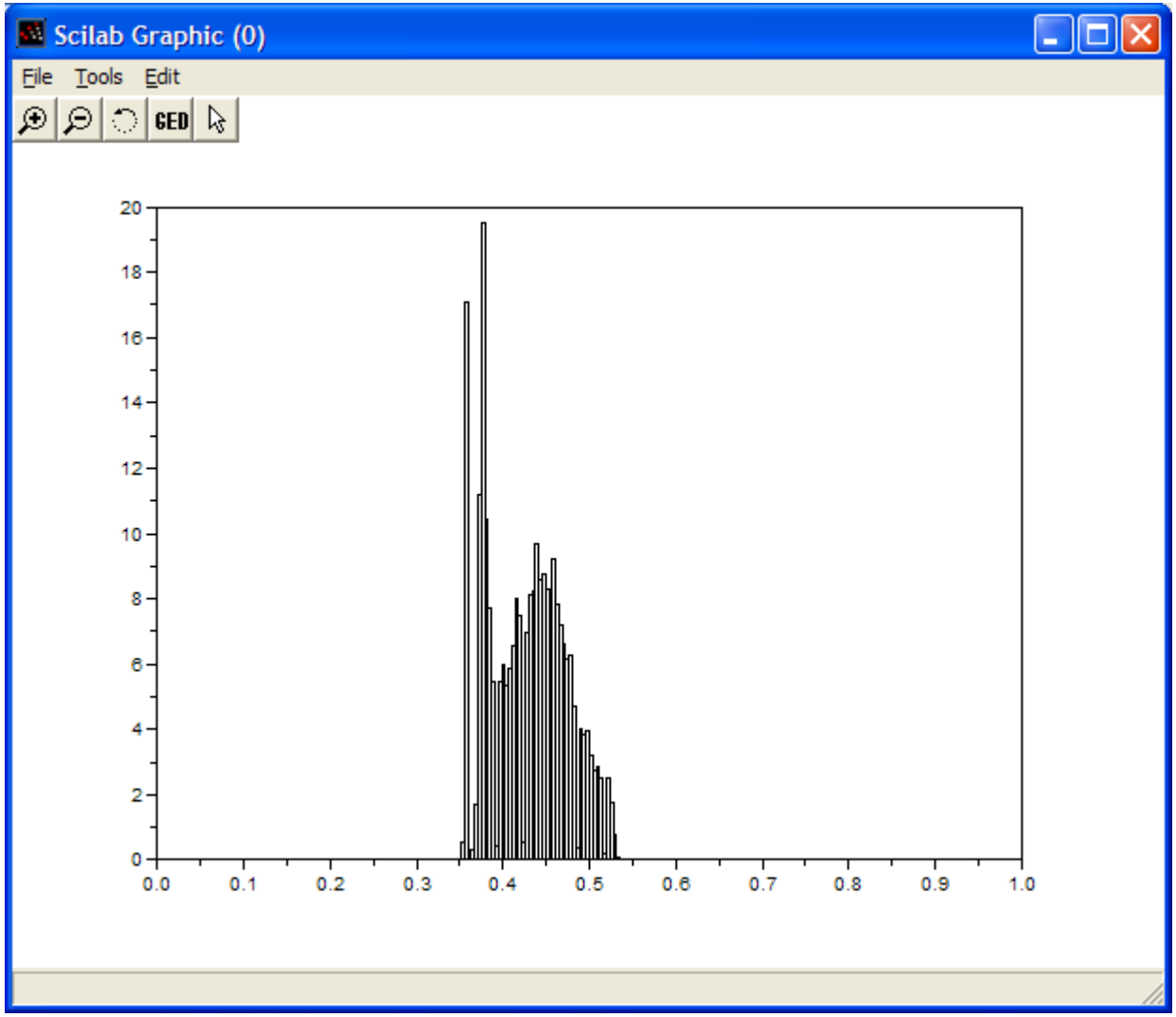


Histogram Examples (cont…)



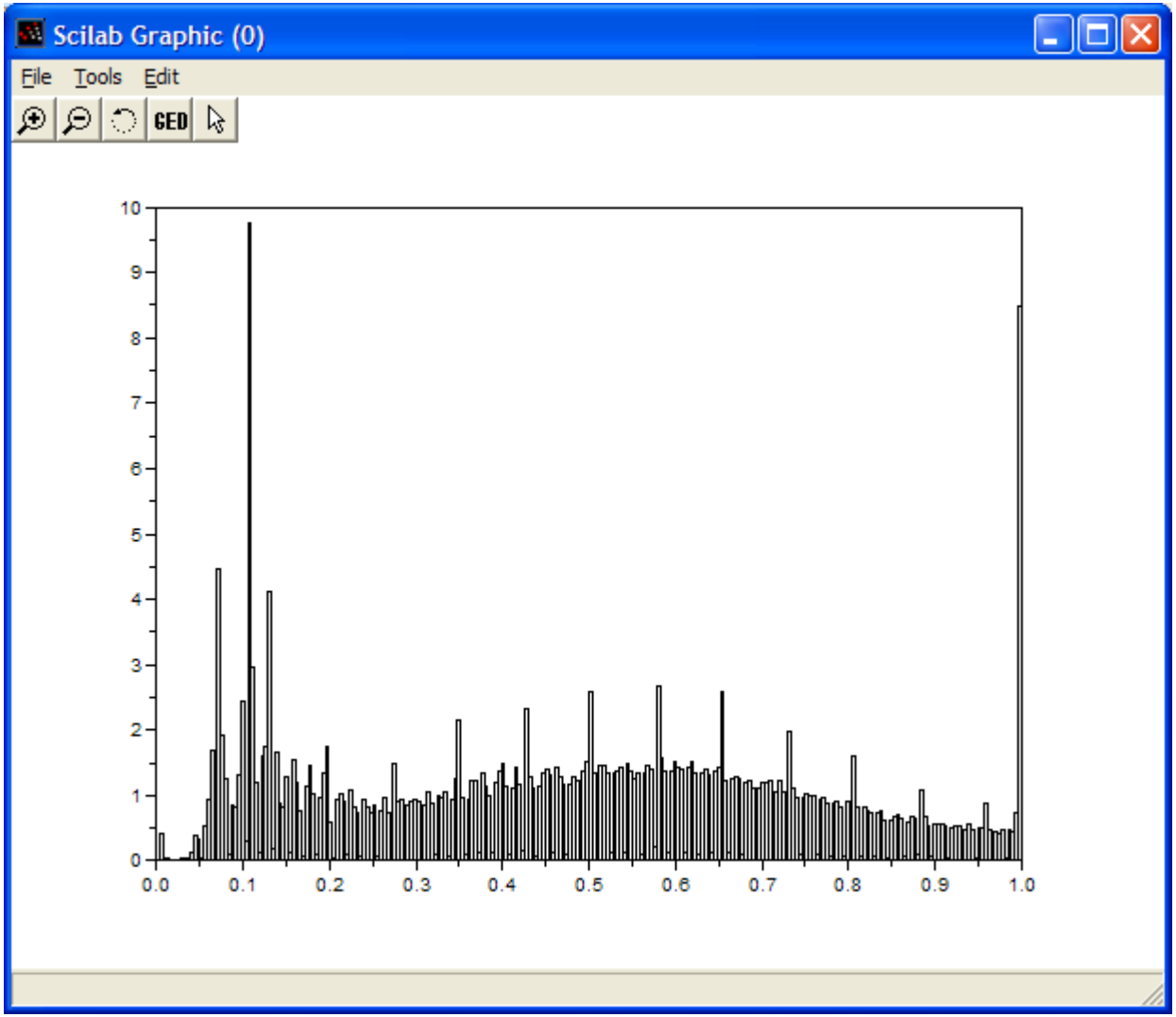
Histogram Examples (cont…)





Histogram Examples (cont…)

Histogram Examples (cont…)

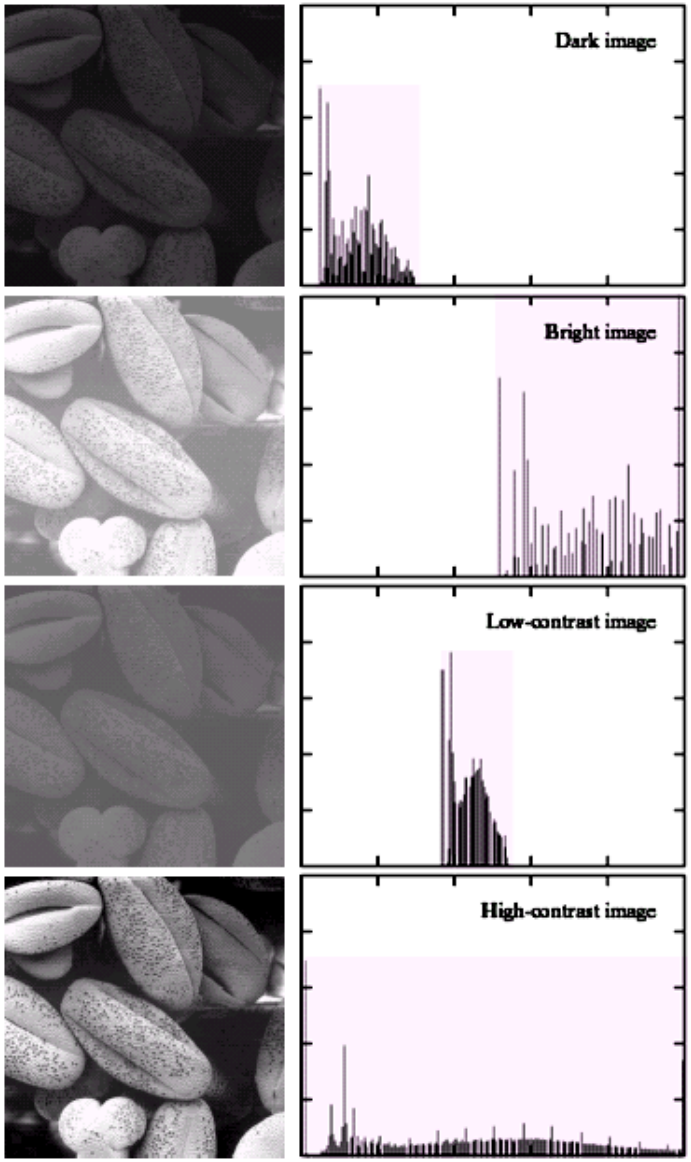


Histogram Examples (cont…)

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

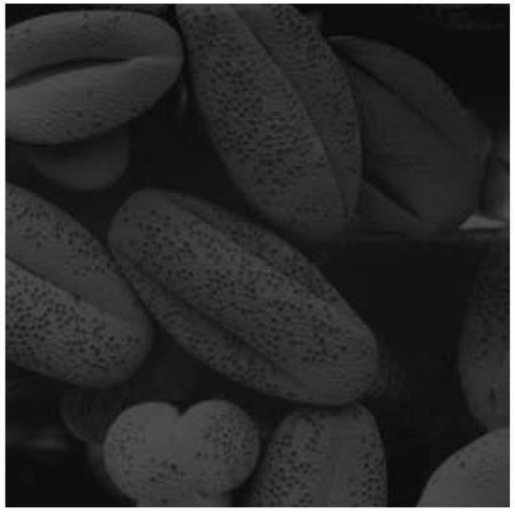
Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

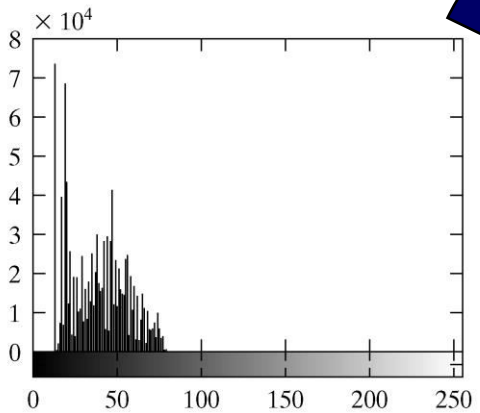
The formula for histogram equalisation is given where

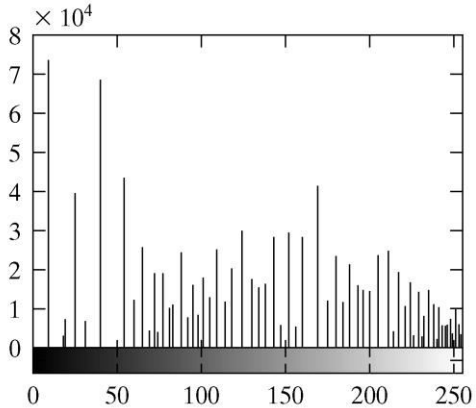
- r_k: input intensity
- s_k: processed intensity
- k: the intensity range (e.g 0.0 – 1.0)
- n_j: the frequency of intensity j
- n: the sum of all frequencies

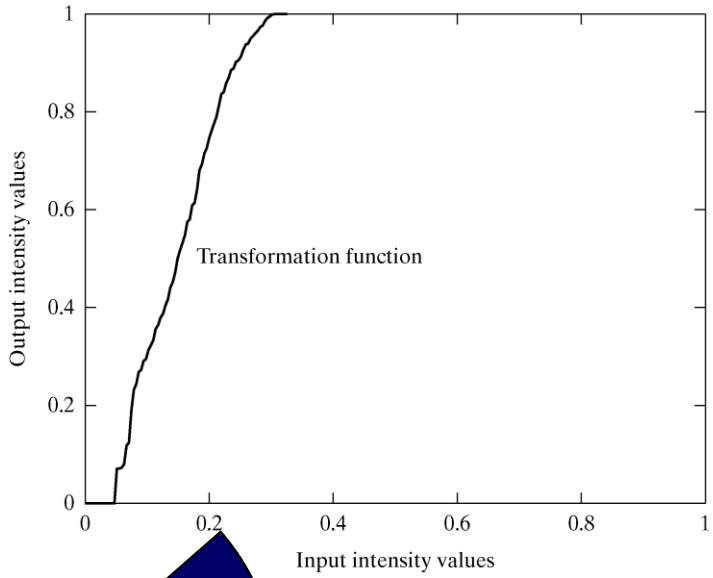
$$\boxed{\begin{aligned} s_k &= T(r_k) \\ &= \sum_{j=1}^{k} p_r(r_j) \\ &= \sum_{j=1}^{k} \frac{n_j}{n} \end{aligned}}$$

Equalisation Transformation Function

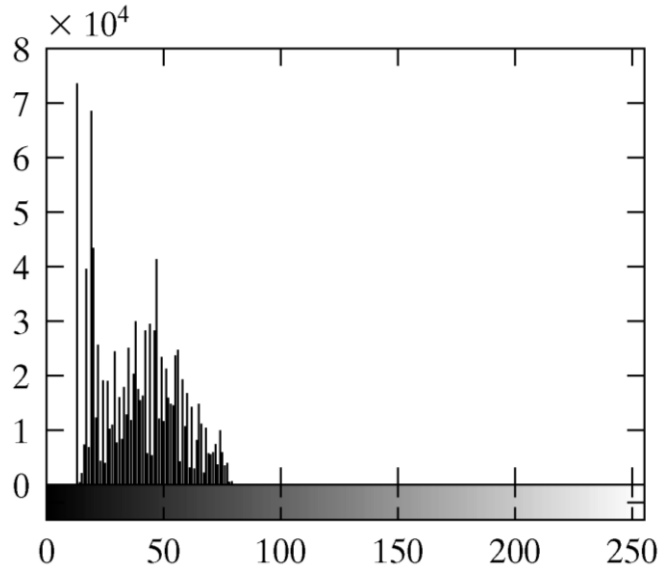


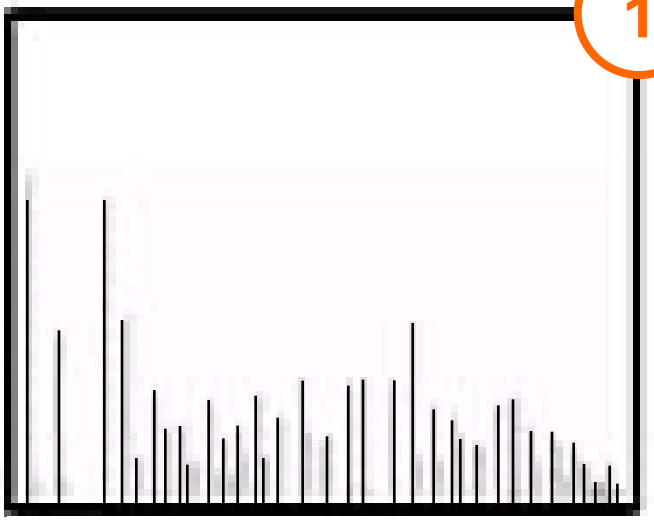


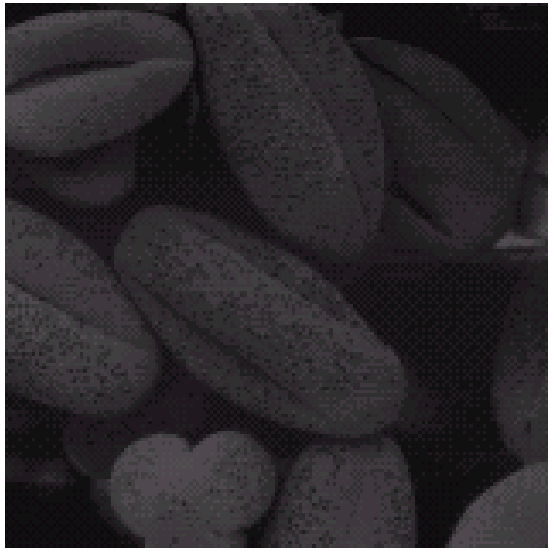




Equalisation Examples

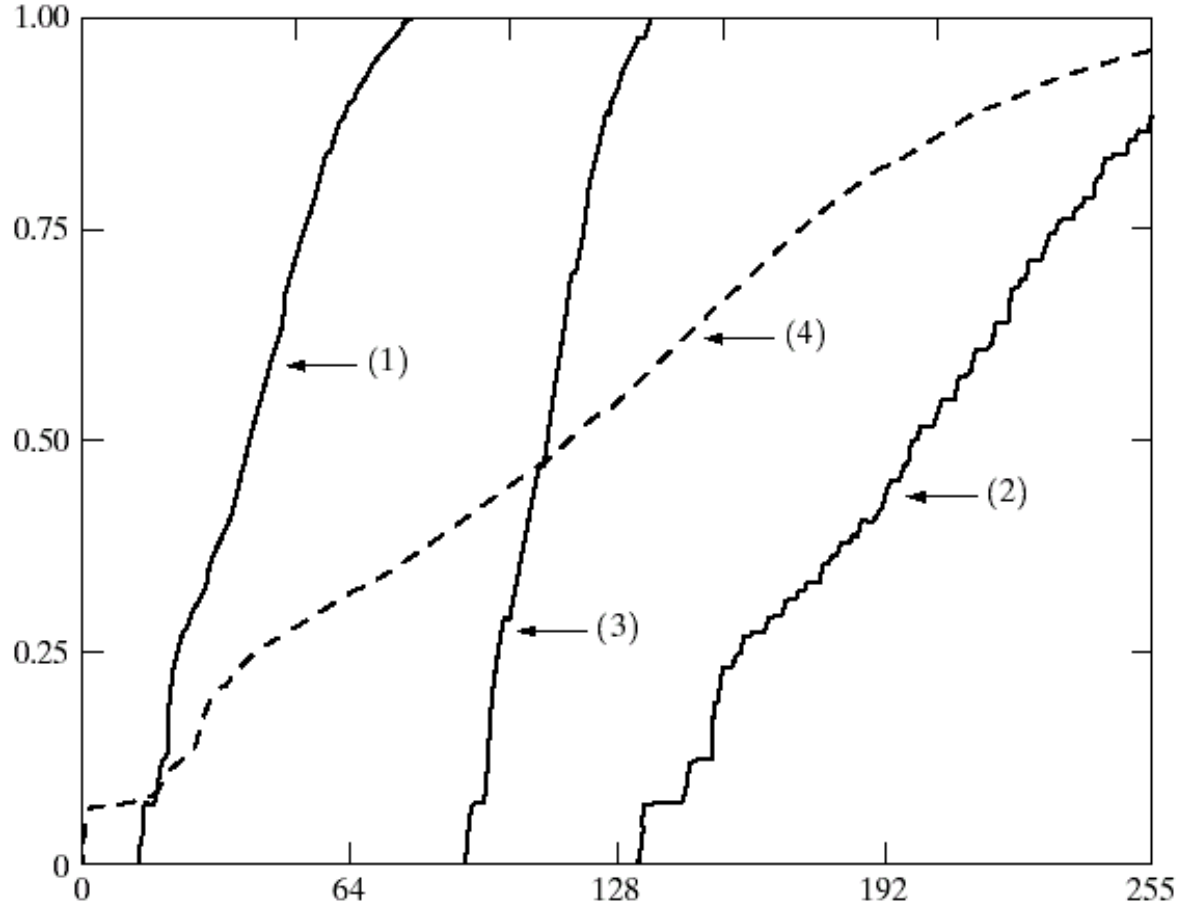




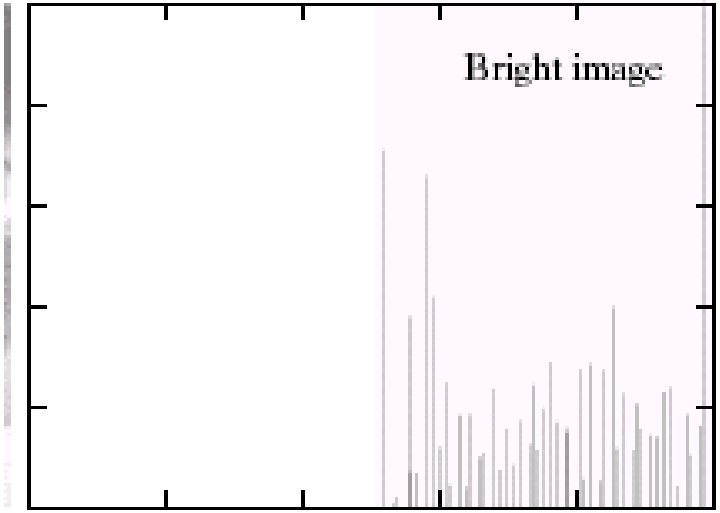


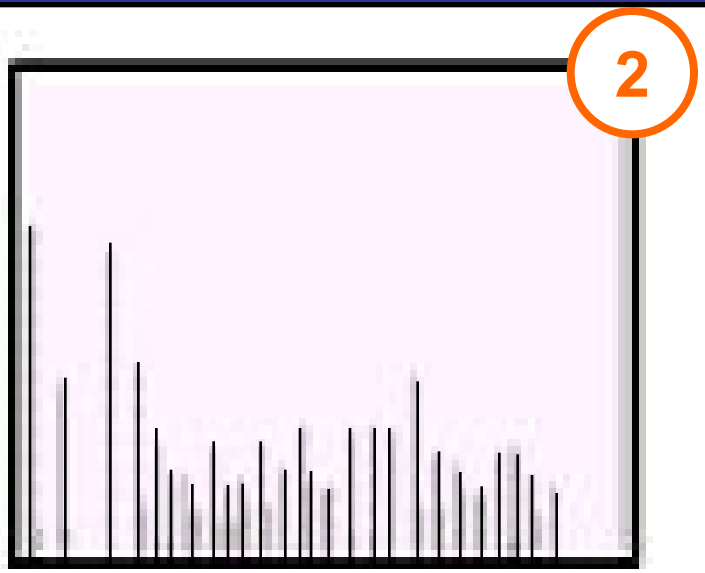
Equalisation Transformation Functions

The functions used to equalise the images in the previous example



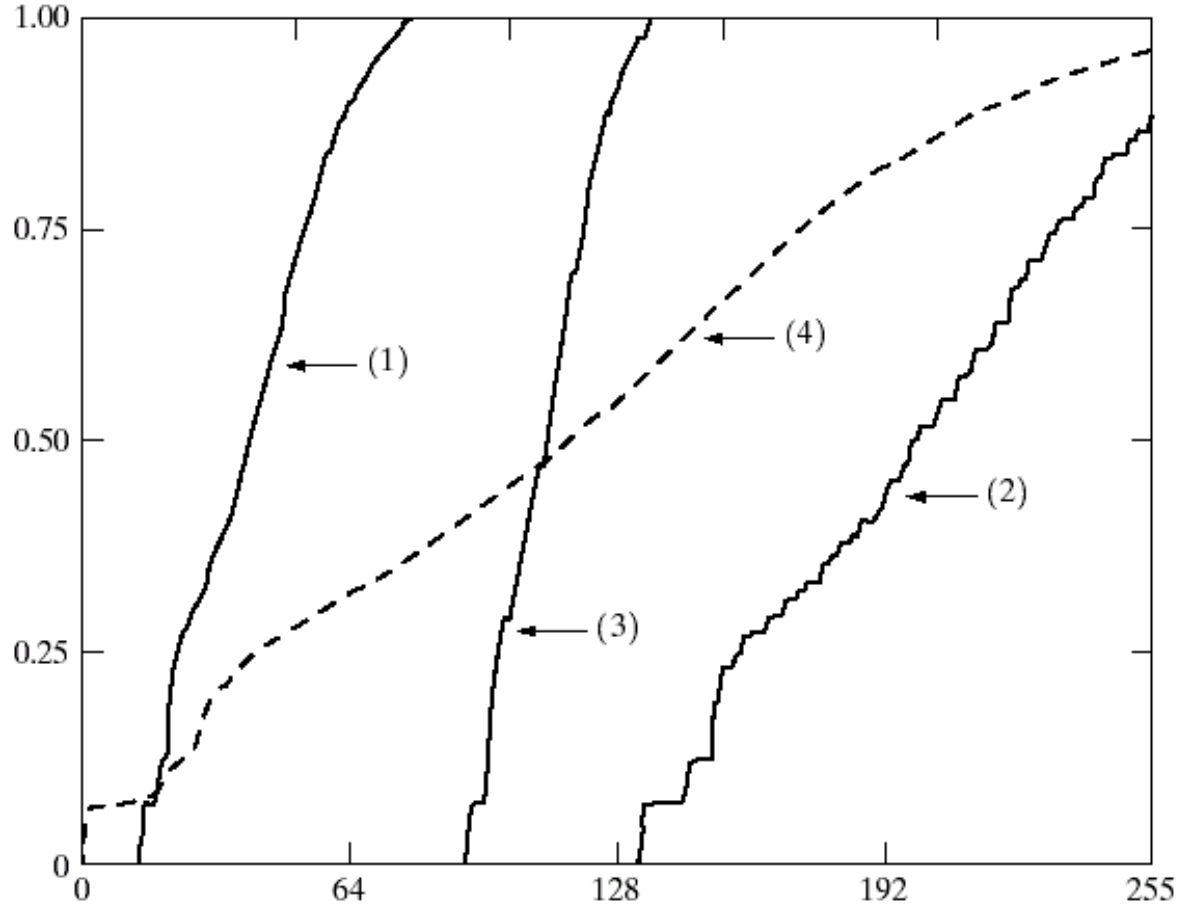
Equalisation Examples



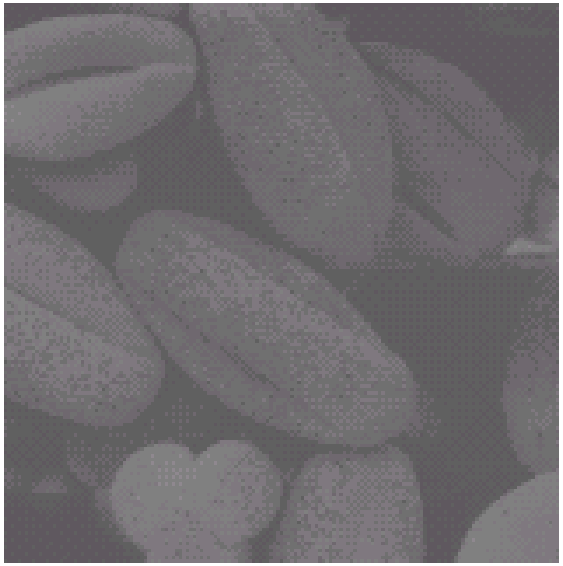


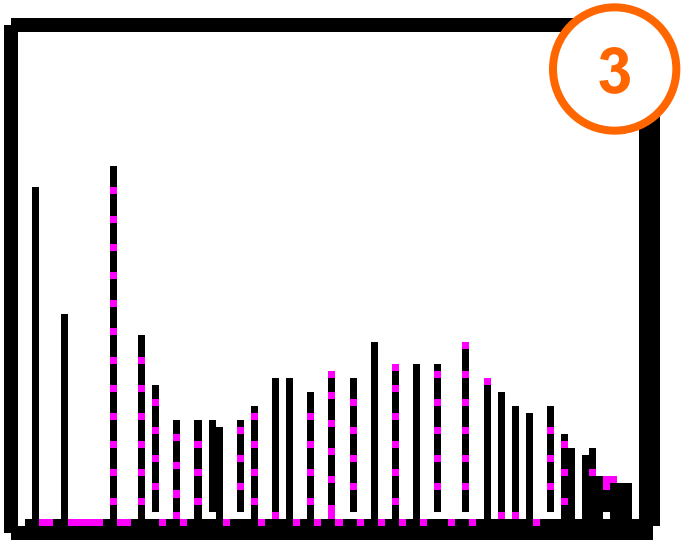
Equalisation Transformation Functions

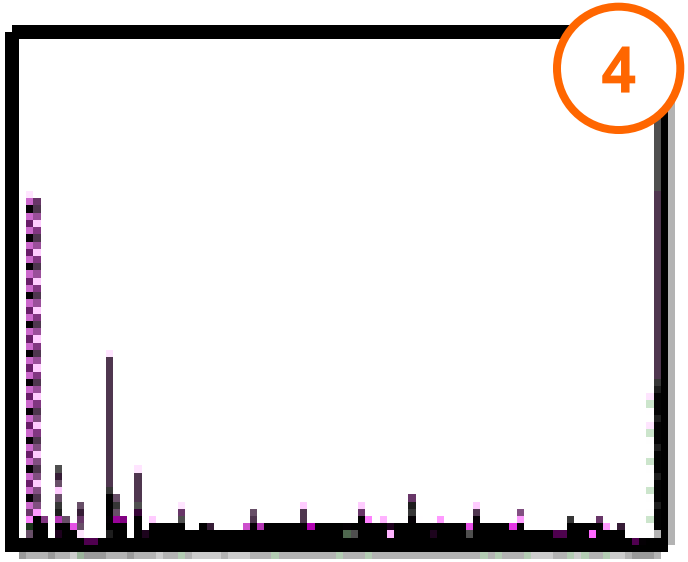
The functions used to equalise the images in the previous example



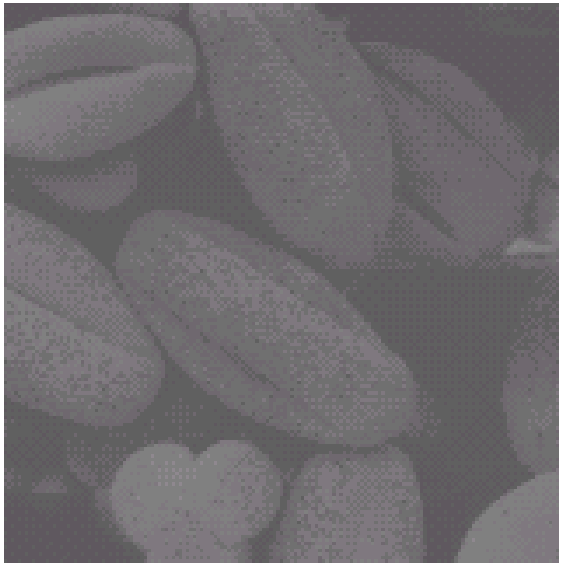
Equalisation Examples (cont…)

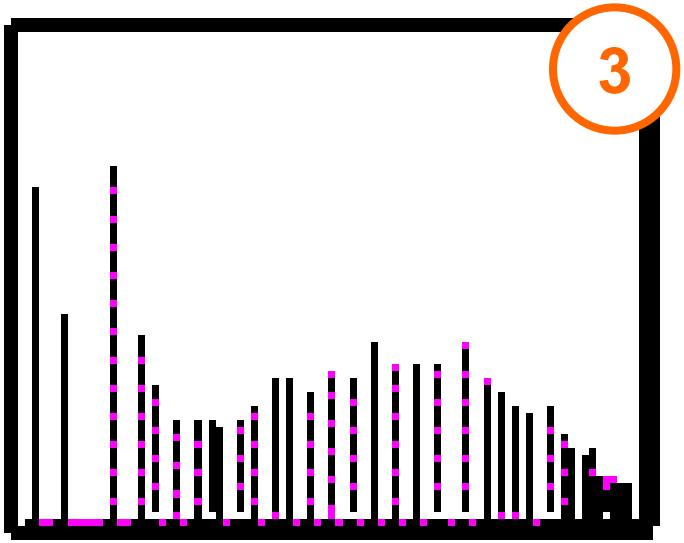


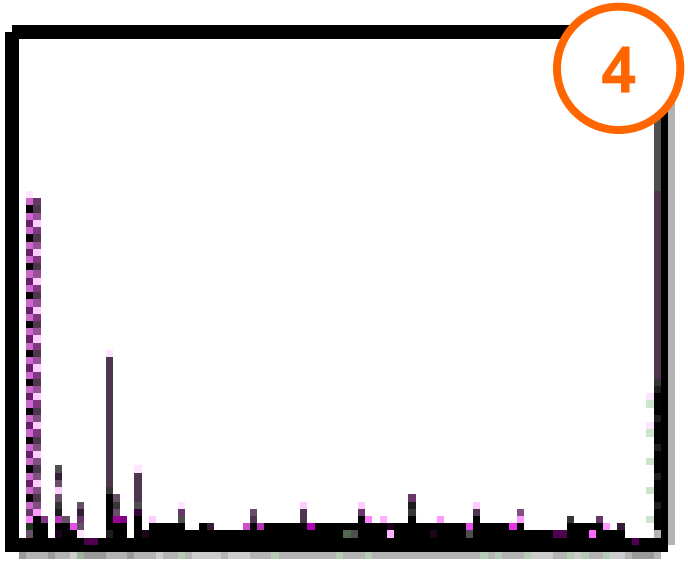




Equalisation Examples (cont…)

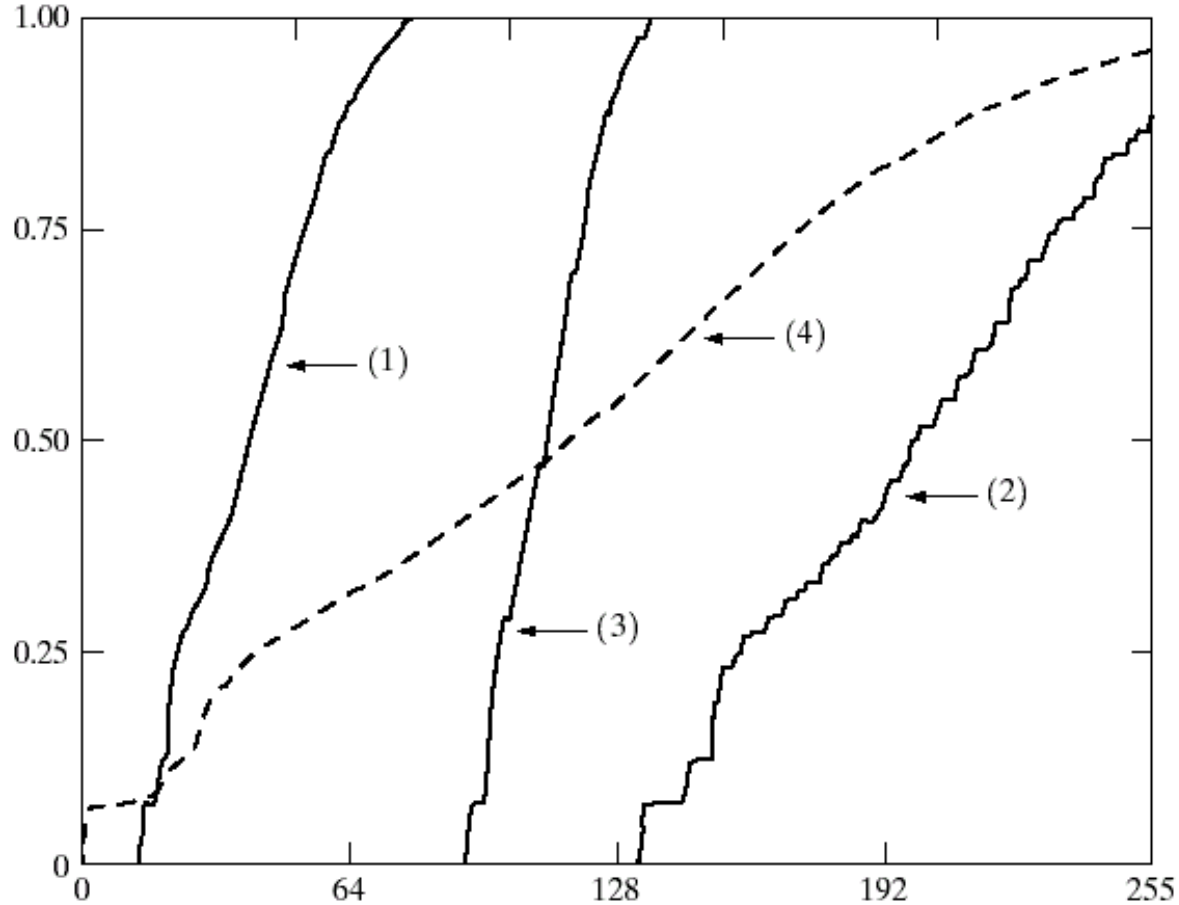






Equalisation Transformation Functions

The functions used to equalise the images in the previous examples



Summary

We have looked at:

- Different kinds of image enhancement
- Histograms
- Histogram equalisation

Next time we will start to look at point processing and some neighbourhood operations

Digital Image Processing

Lecture - 4

Image Enhancement

(Point Processing)

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Contents

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

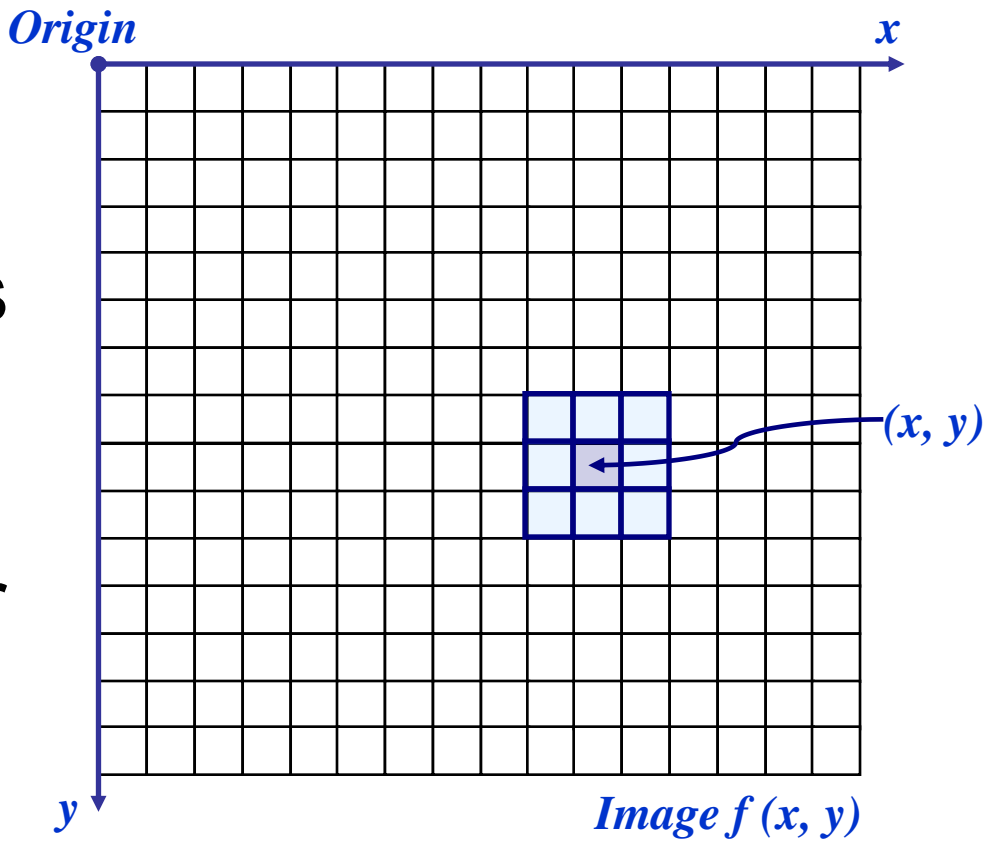
- What is point processing?
- Negative images
- Thresholding
- Logarithmic transformation
- Power law transforms
- Grey level slicing
- Bit plane slicing

Basic Spatial Domain Image Enhancement

Most spatial domain enhancement operations can be reduced to the form

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

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



Point Processing

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

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

Point processing operations take the form

$$s = T(r)$$

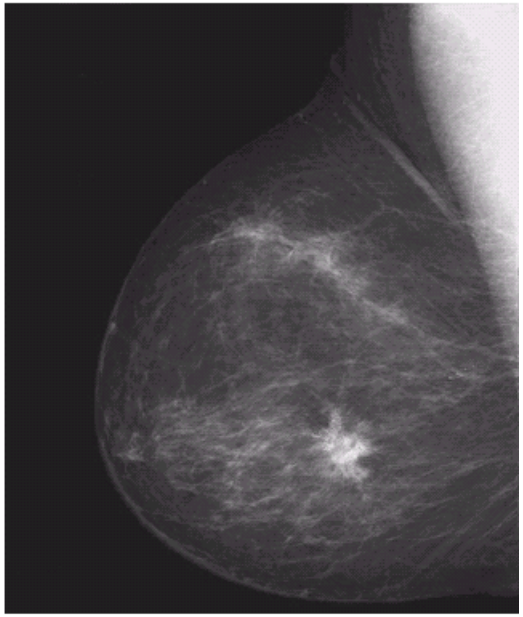
where s refers to the processed image pixel value and r refers to the original image pixel value

Point Processing Example: Negative Images

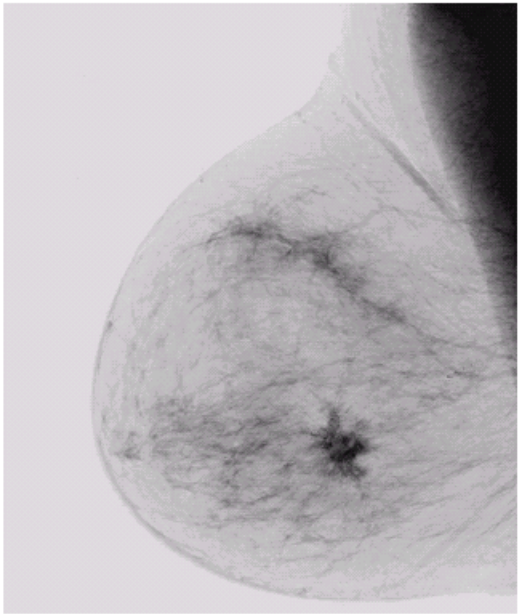
Negative images are useful for enhancing white or grey detail embedded in dark regions of an image

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

Original Image

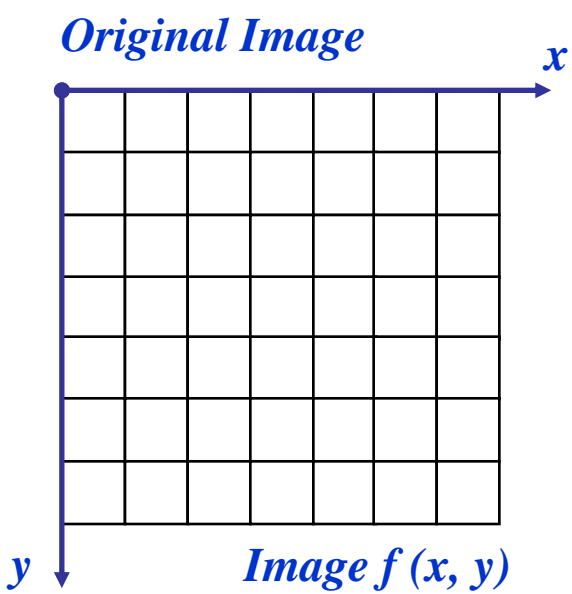


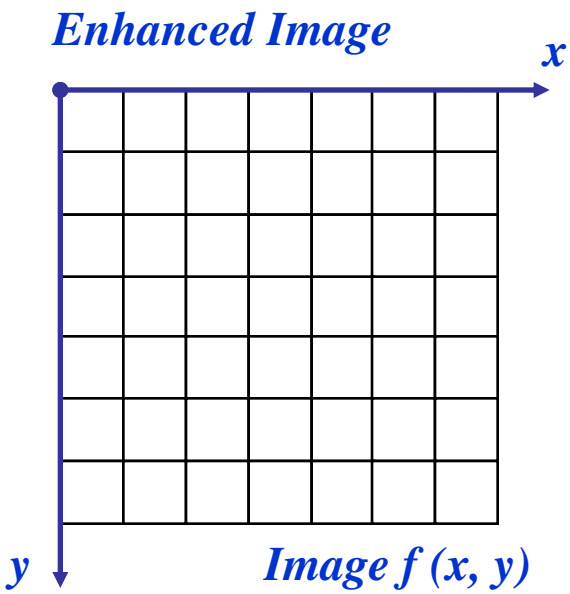
$s = 1.0 - r$



Negative Image

Point Processing Example: Negative Images (cont…)



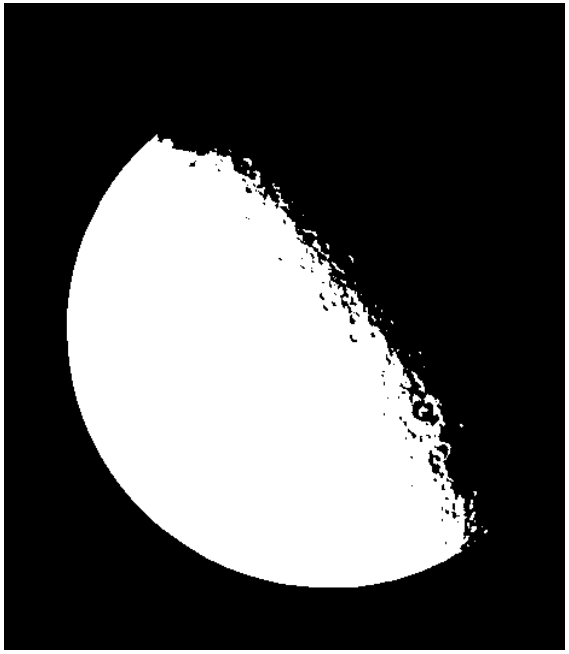


$$s = intensity_{max} - r$$

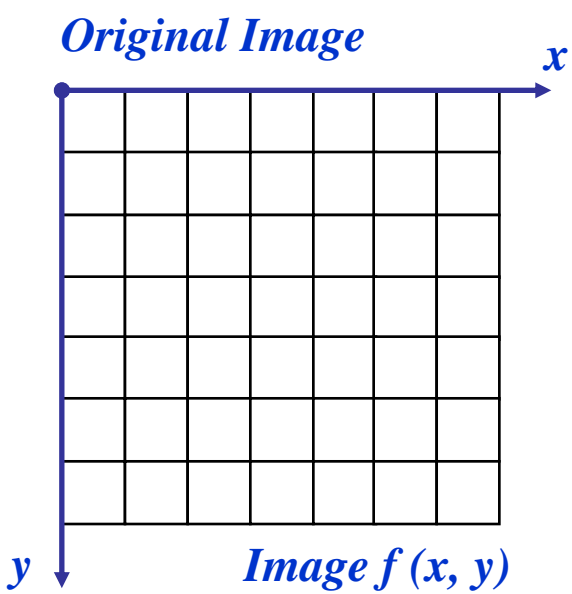
Point Processing Example: Thresholding

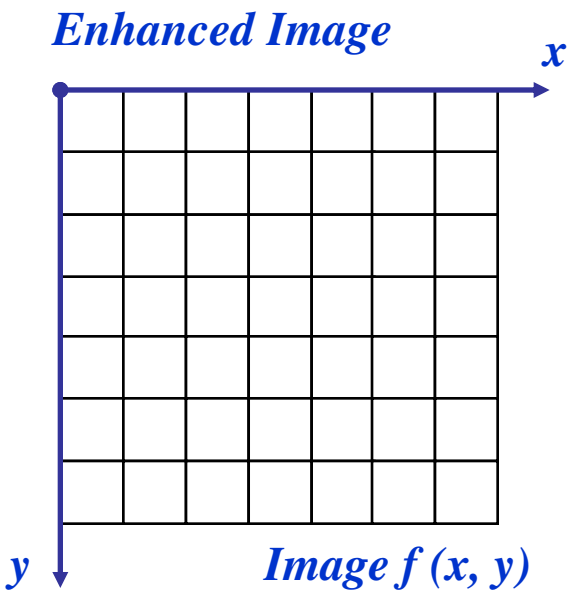
Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background

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



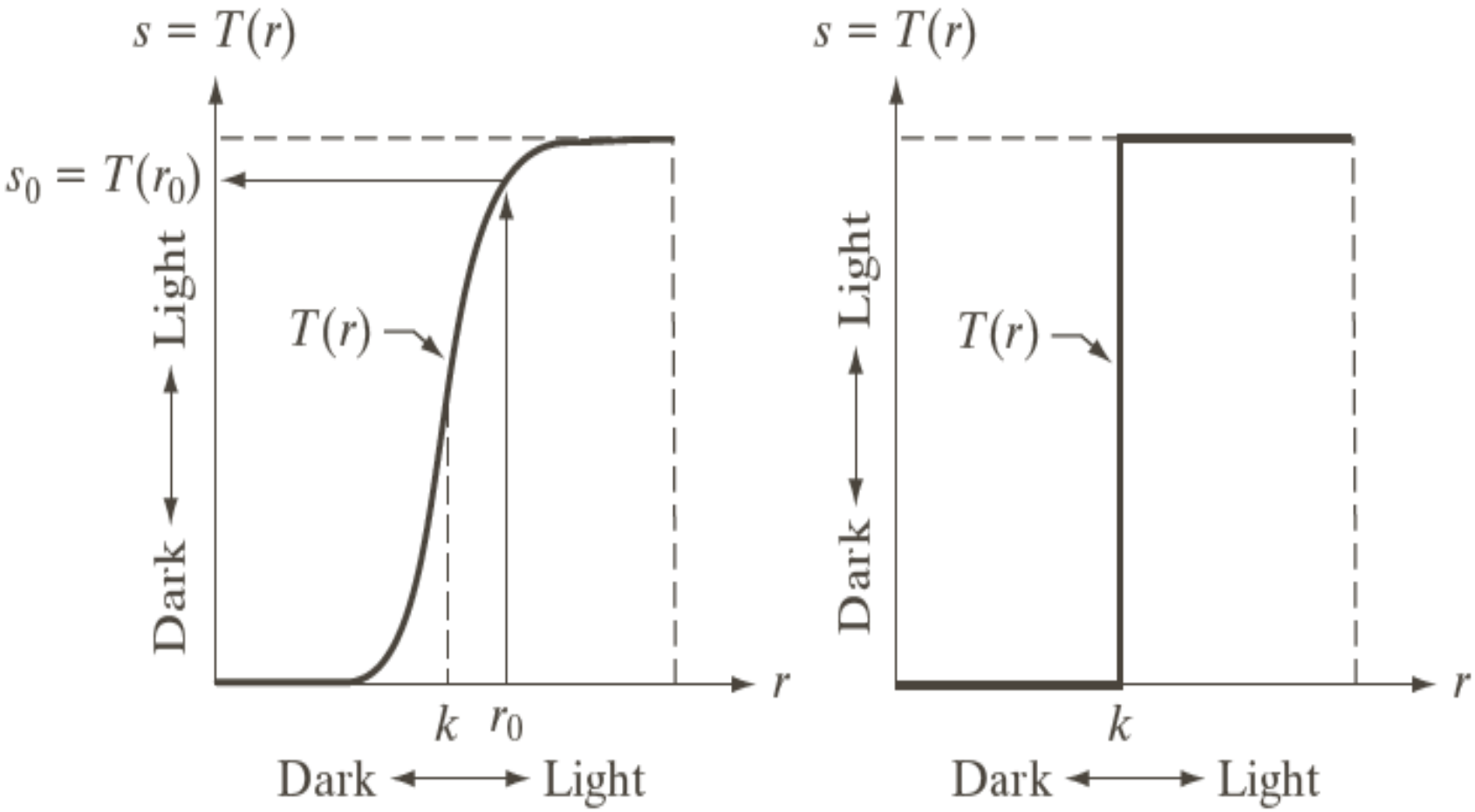
Point Processing Example: Thresholding (cont…)





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

Intensity Transformations

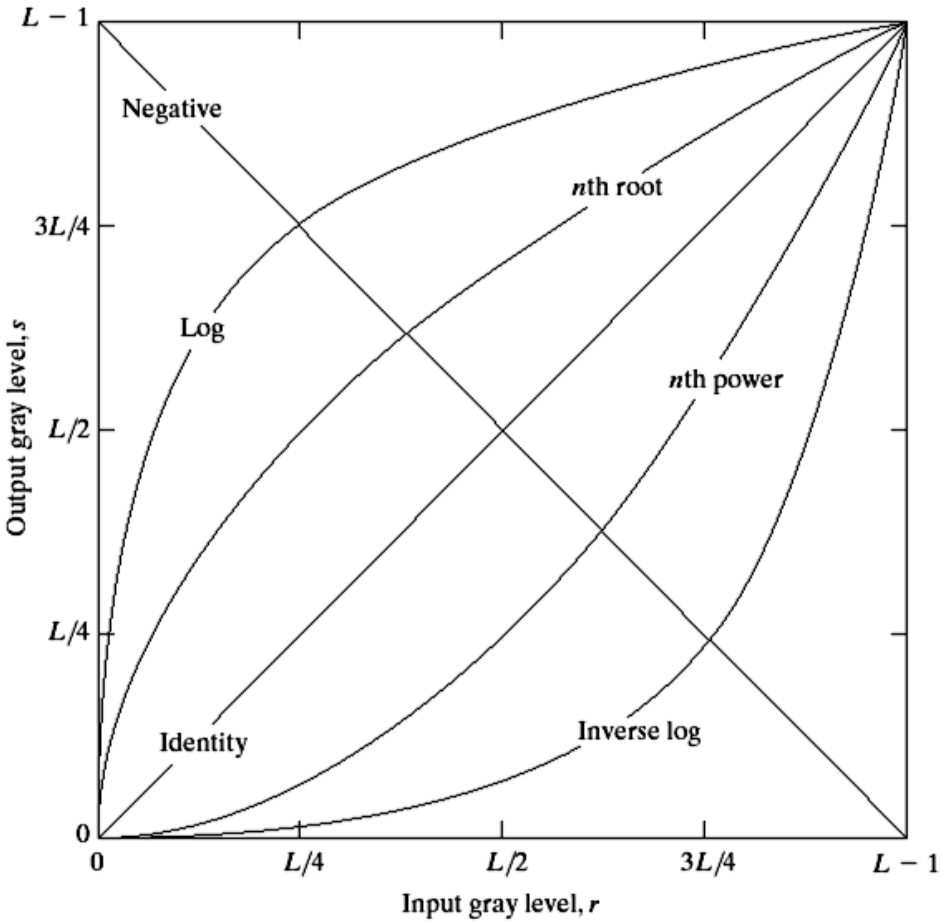


Basic Grey Level Transformations

There are many different kinds of grey level transformations

Three of the most common are shown here

- Linear
  - Negative/Identity
- Logarithmic
  - Log/Inverse log
- Power law
  - n^{th} power/n^{th} root



Logarithmic Transformations

The general form of the log transformation is

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

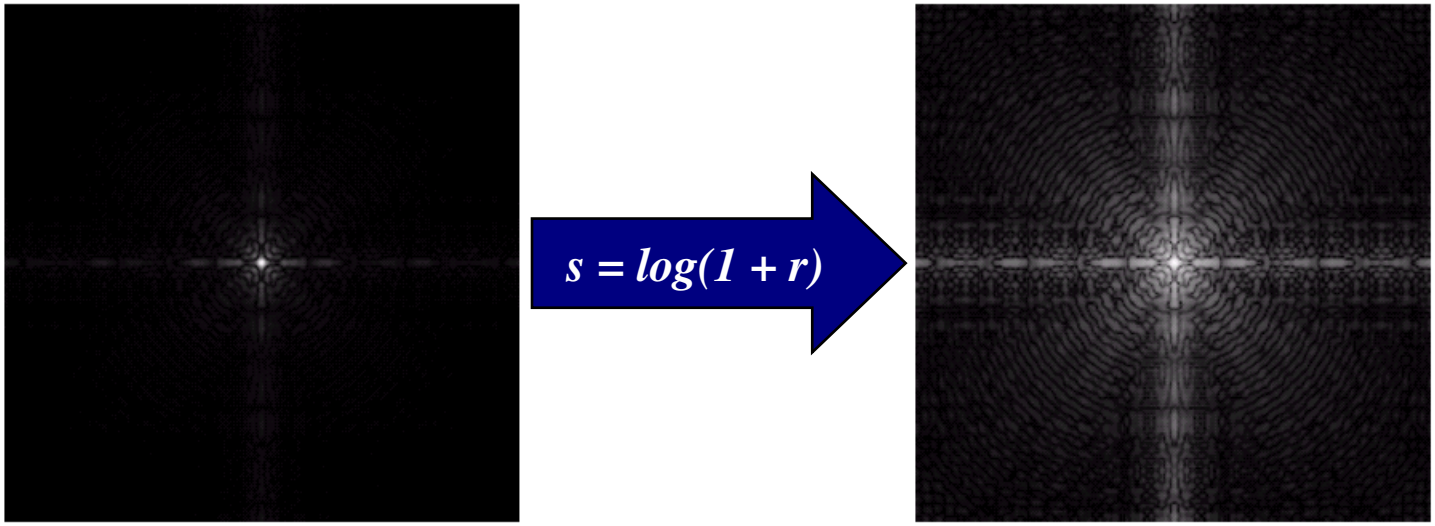
The log transformation maps a narrow range of low input grey level values into a wider range of output values

The inverse log transformation performs the opposite transformation

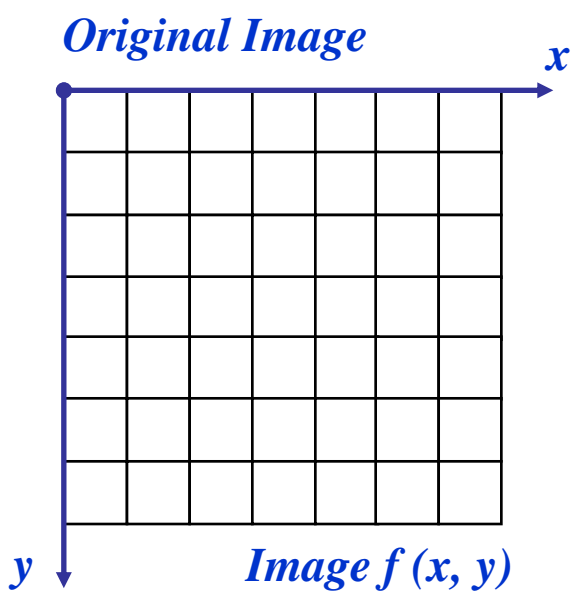
Logarithmic Transformations (cont…)

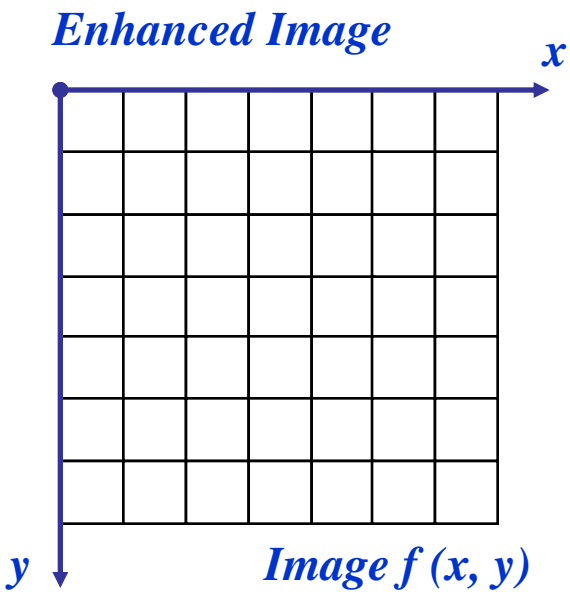
Log functions are particularly useful when the input grey level values may have an extremely large range of values

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



Logarithmic Transformations (cont…)





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

We usually set c to 1

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

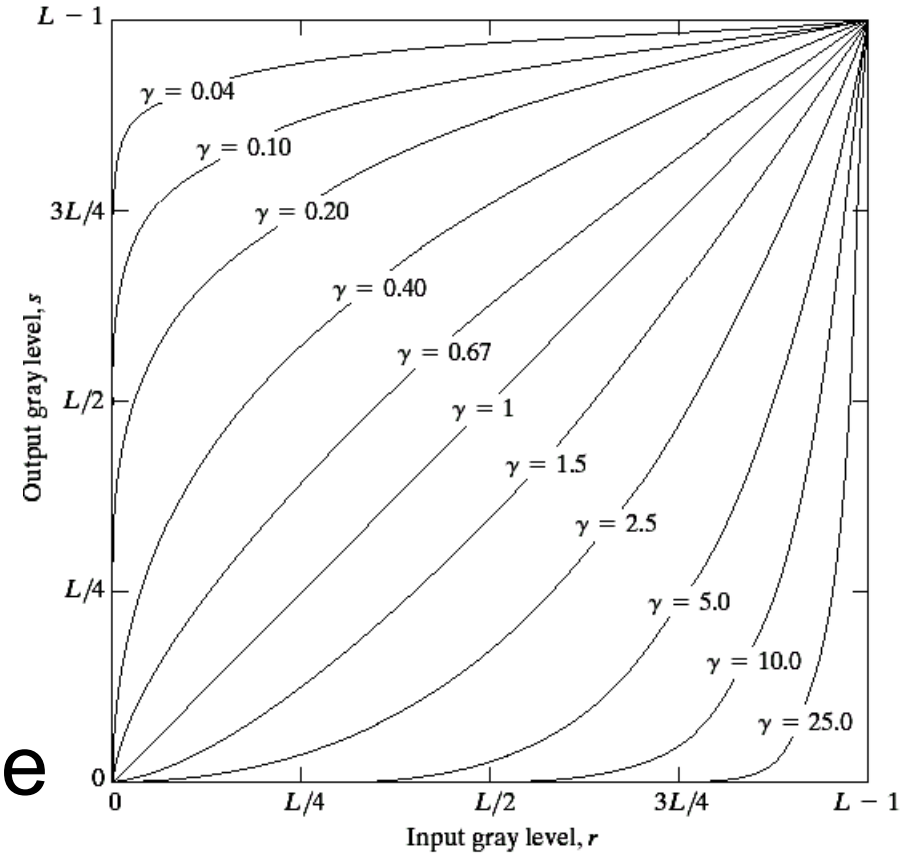
Power Law Transformations

Power law transformations have the following form

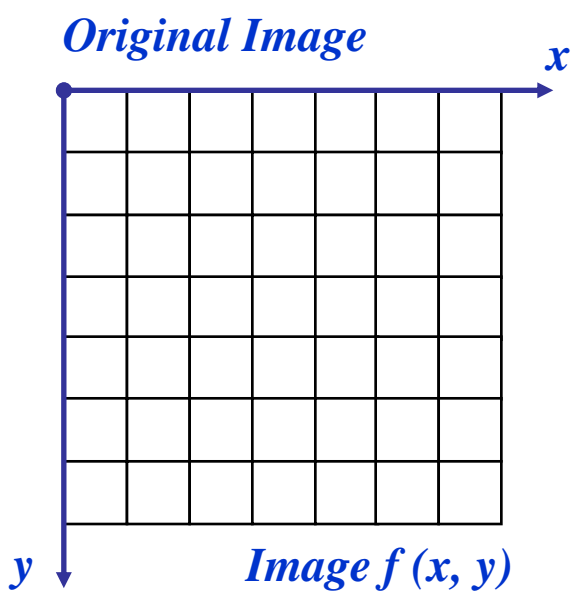
$$s = c * r^{\gamma}$$

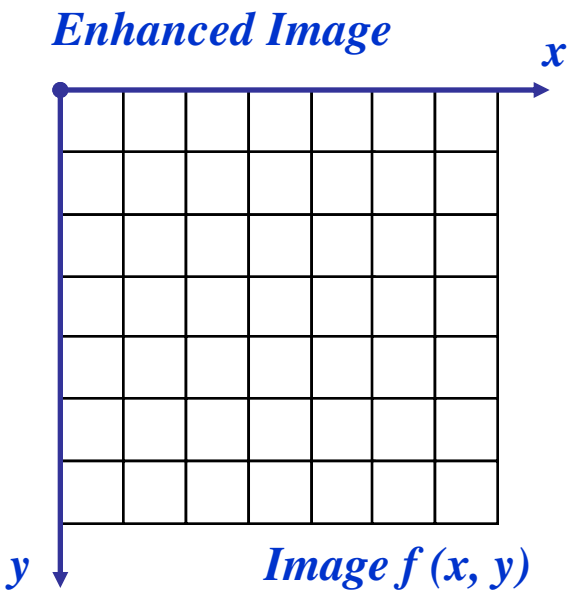
Map a narrow range of dark input values into a wider range of output values or vice versa

Varying 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]

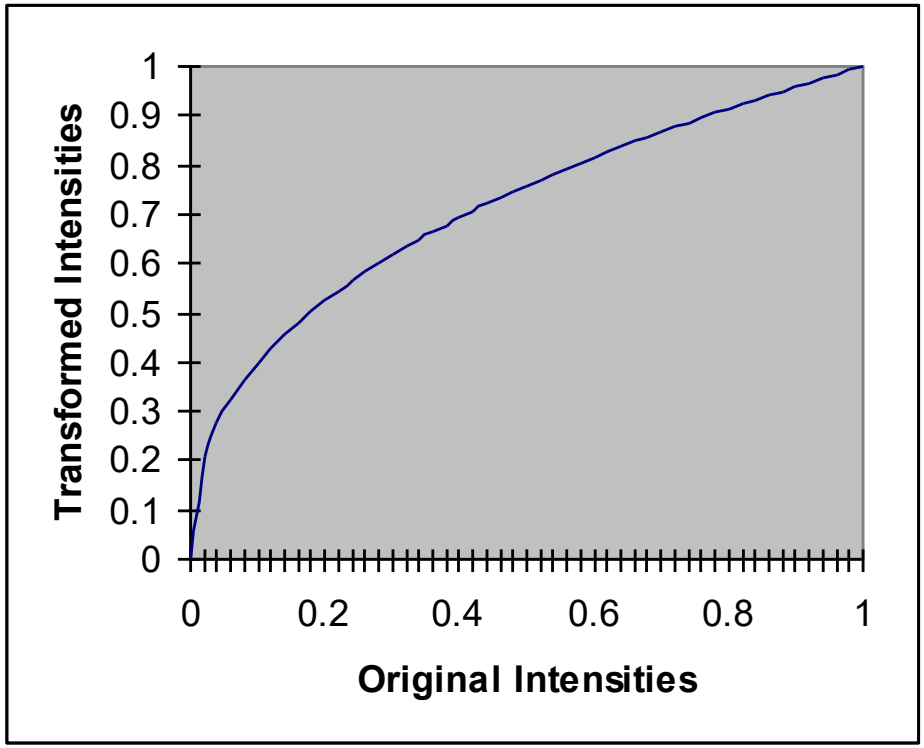
Power Law Example

Power Law Example (cont…)

$$\gamma = 0.6$$

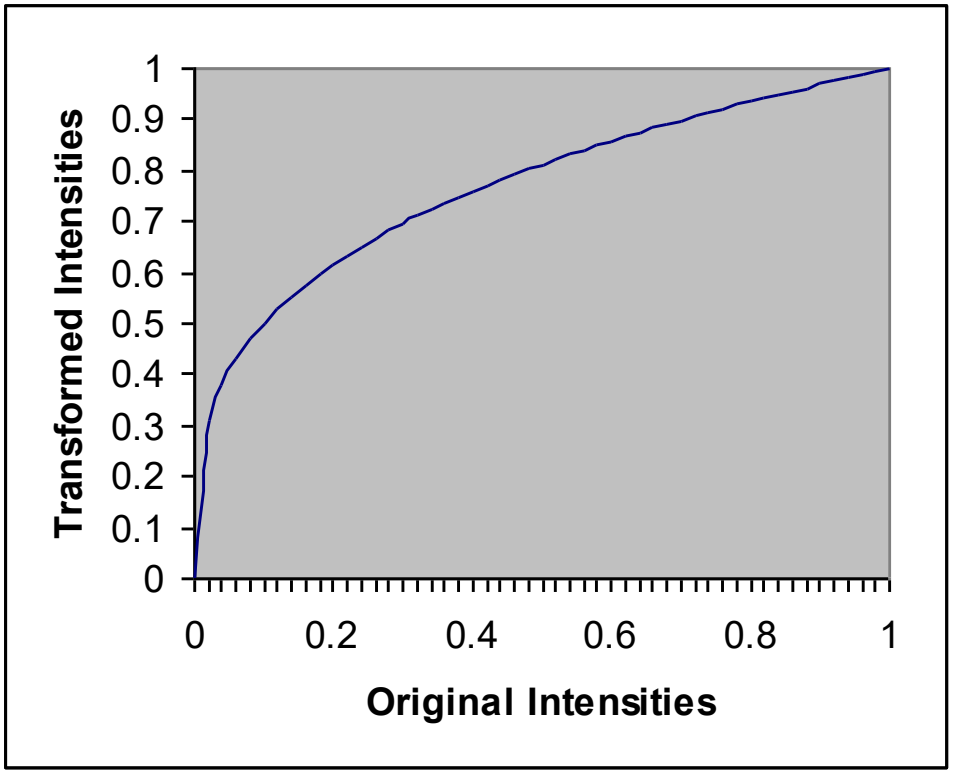
Power Law Example (cont…)

$$\gamma = 0.4$$



Power Law Example (cont…)

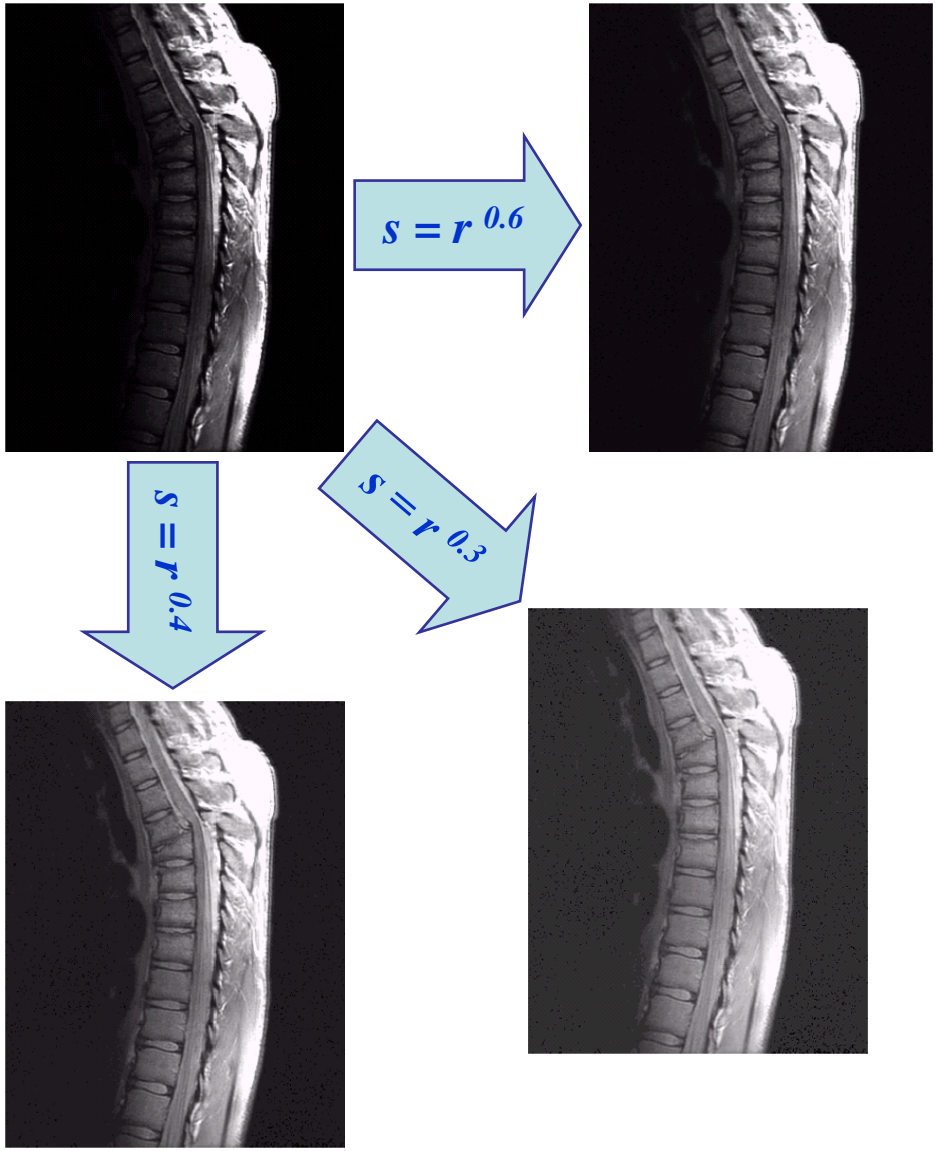
$$\gamma = 0.3$$



Power Law Example (cont…)

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

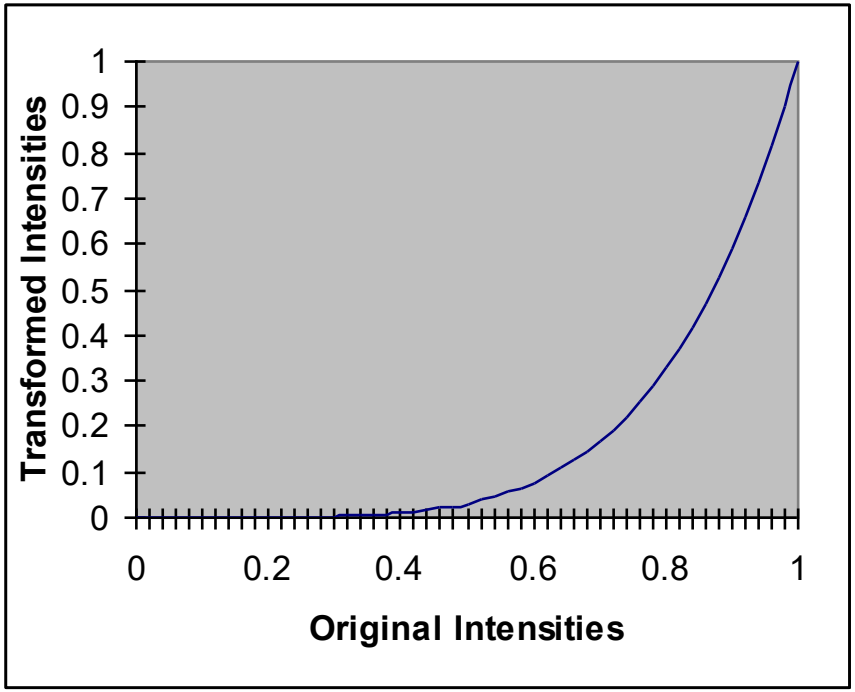
Different curves highlight different detail



Power Law Example

Power Law Example (cont…)

$$\gamma = 5.0$$

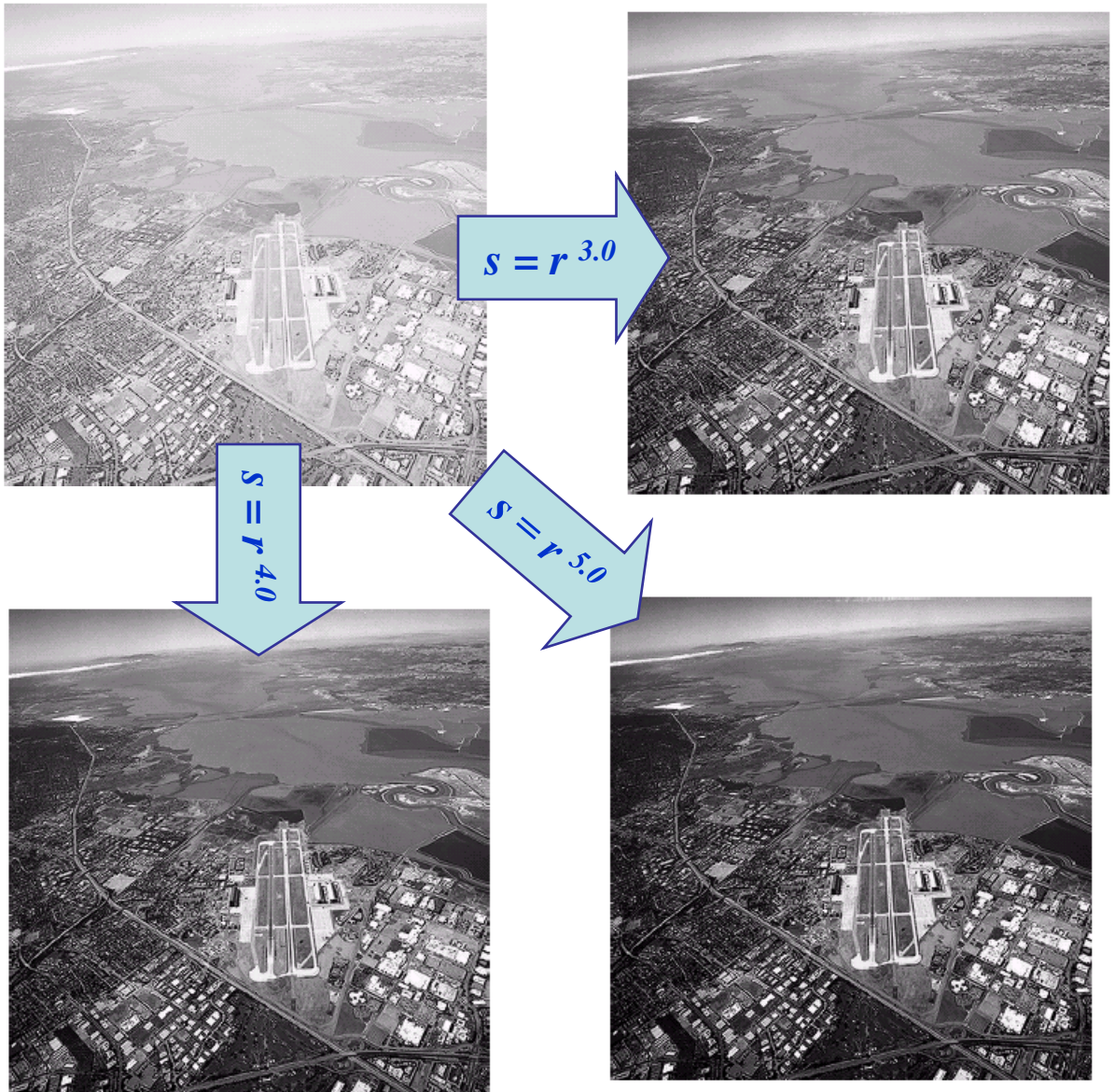


Power Law Transformations (cont…)

An aerial photo of a runway is shown

This time power law transforms are used to darken the image

Different curves highlight different detail

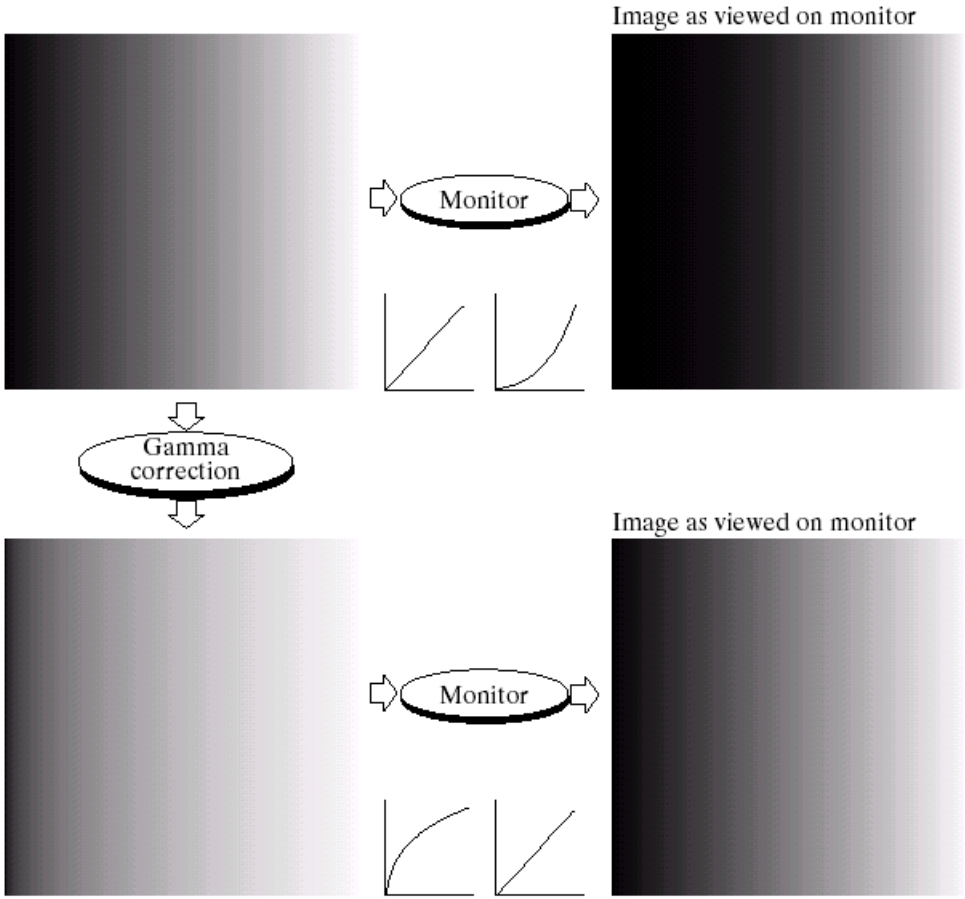


Gamma Correction

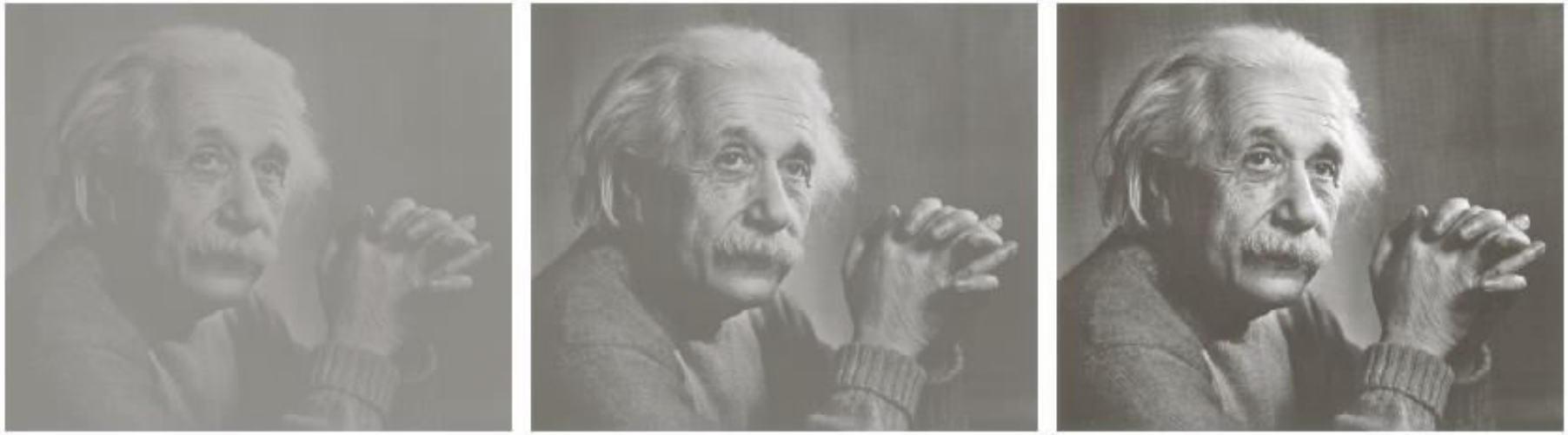
Many of you might be familiar with gamma correction of computer monitors

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

Can be corrected using a log transform



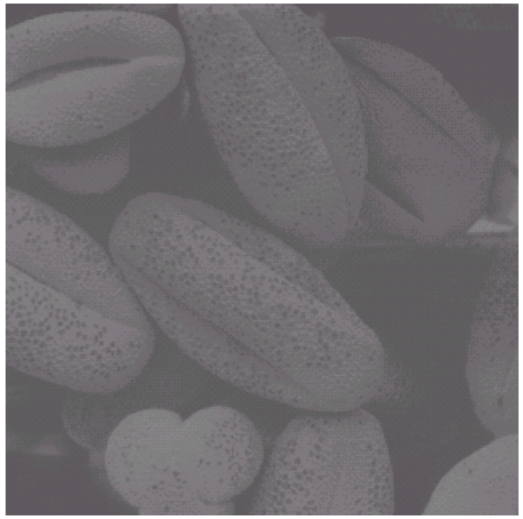
More Contrast Issues

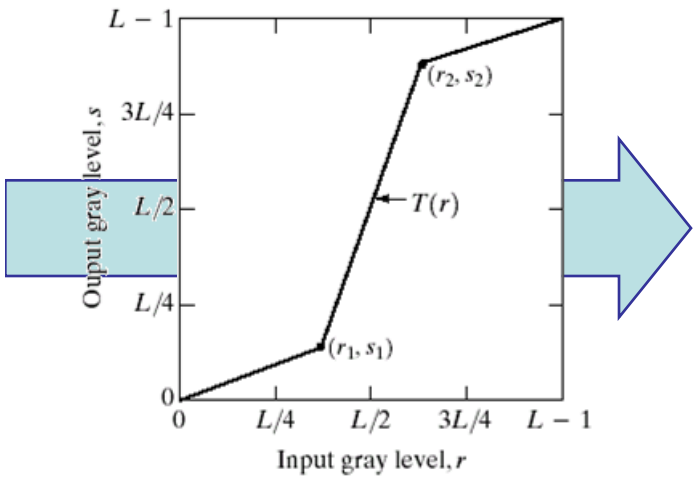


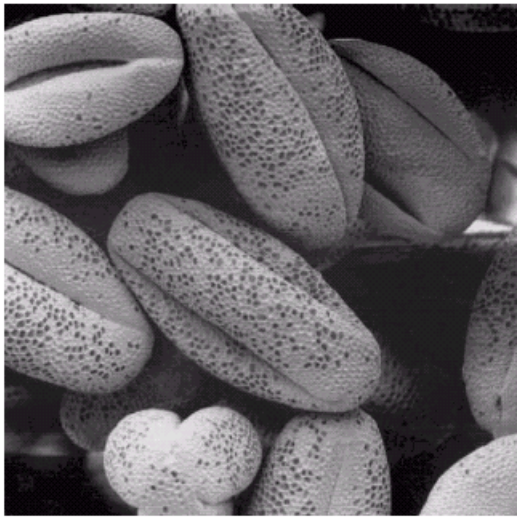
Piecewise Linear Transformation Functions

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

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



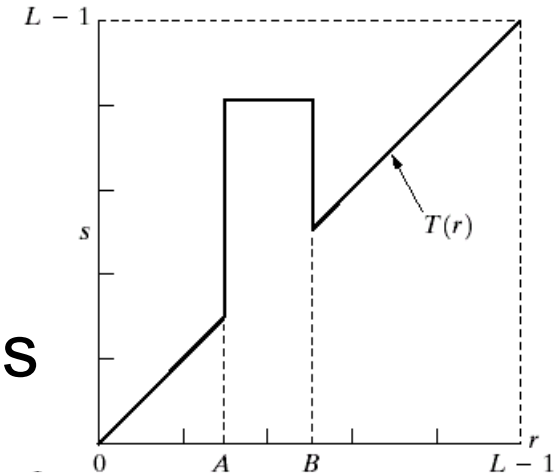


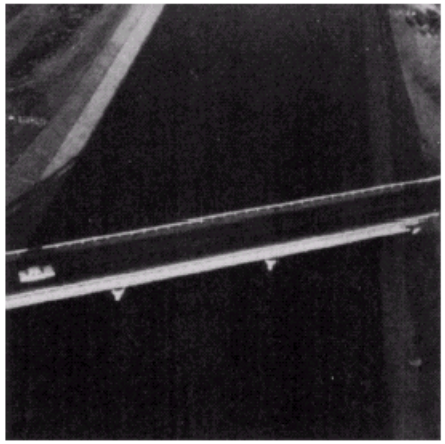


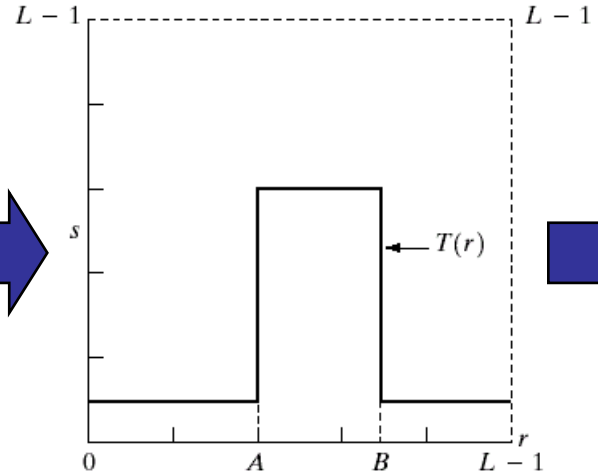
Gray Level Slicing

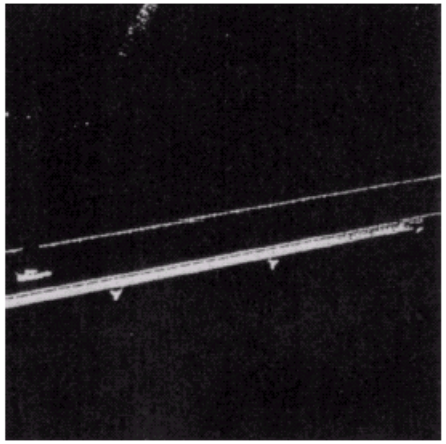
Highlights a specific range of grey levels

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





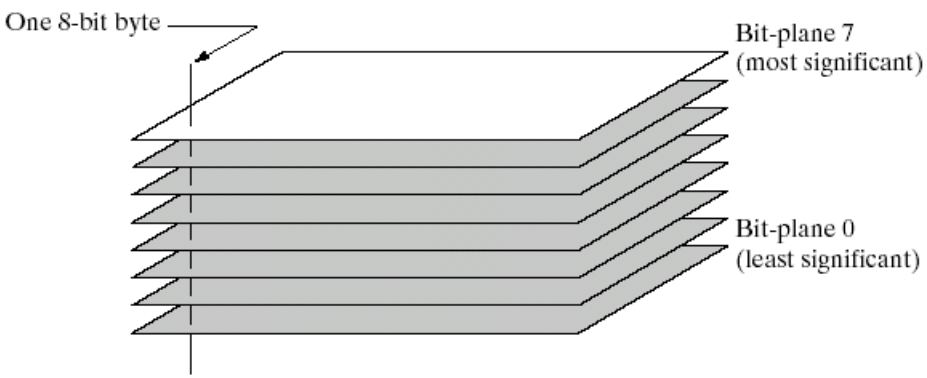


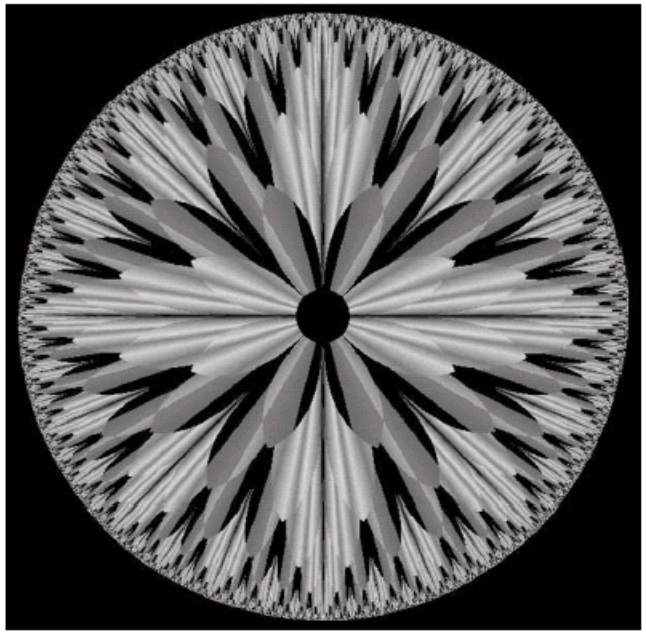


Bit Plane Slicing

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

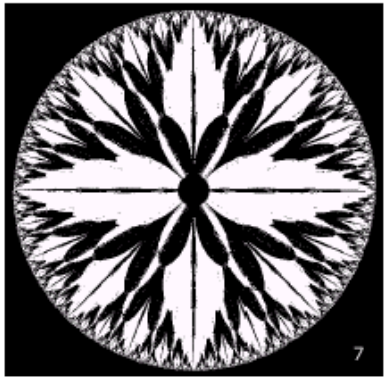
- Higher-order bits usually contain most of the significant visual information
- Lower-order bits contain subtle details

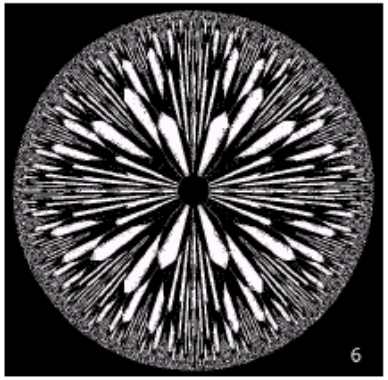




Bit Plane Slicing (cont…)

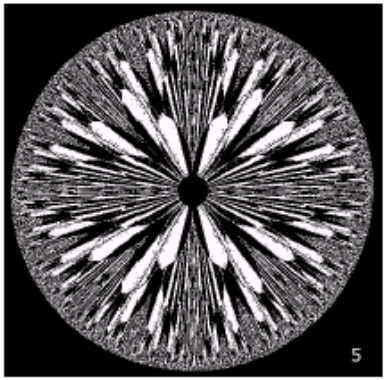
[10000000]

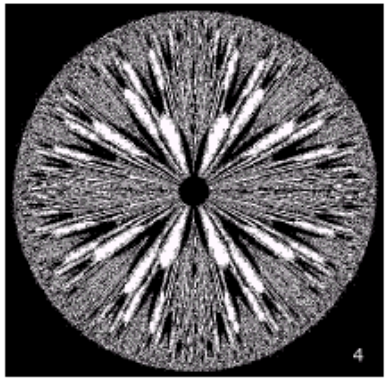


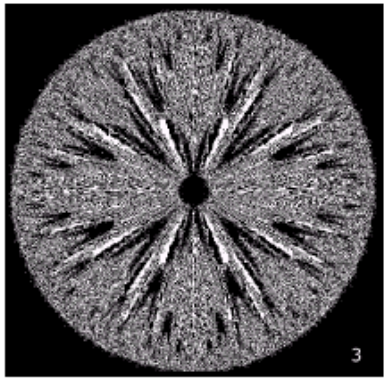


[01000000]

[00100000]

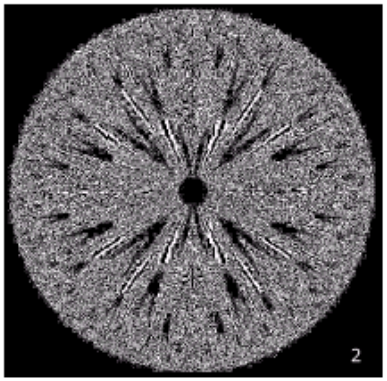


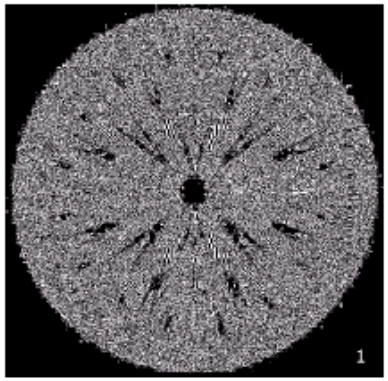


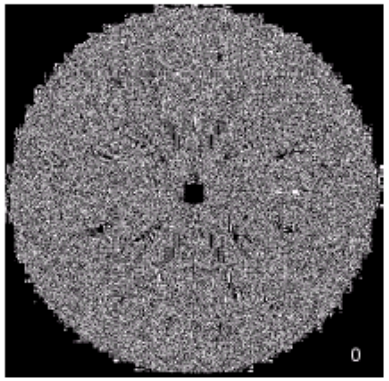


[00001000]

[00000100]



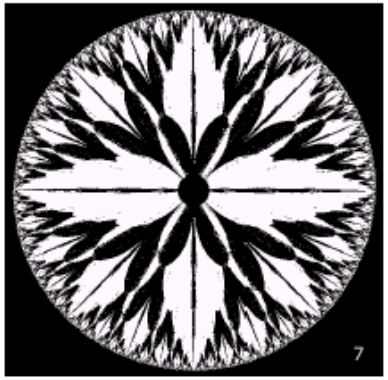


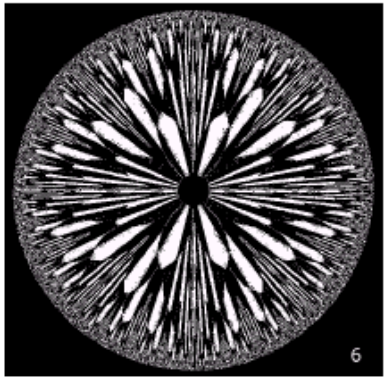


[00000001]

Bit Plane Slicing (cont…)

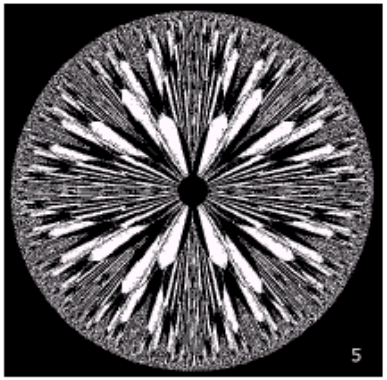
[10000000]

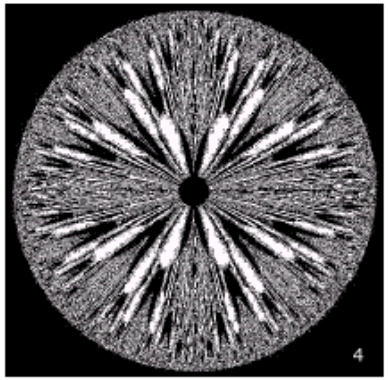


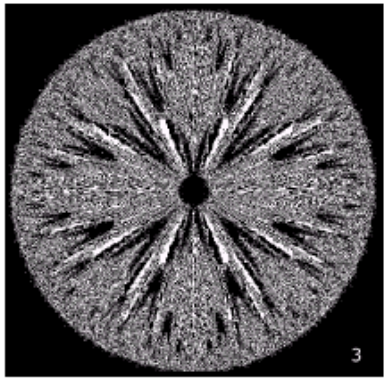


[01000000]

[00100000]

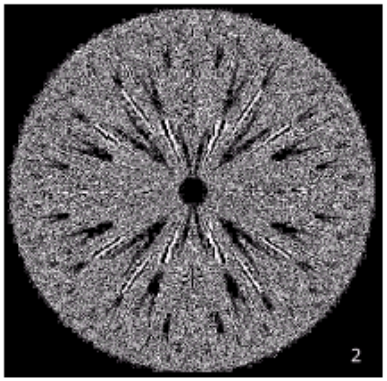


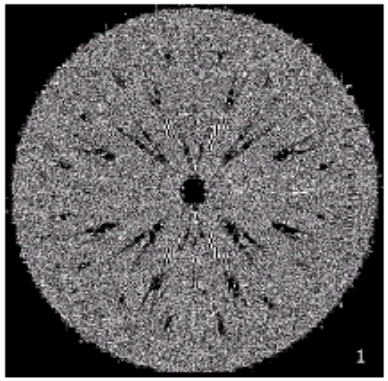


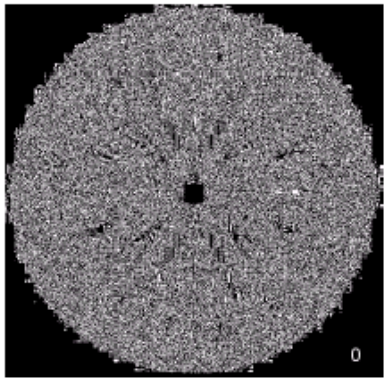


[00001000]

[00000100]







[00000001]

Bit Plane Slicing (cont…)

a b c
d e f
g h i

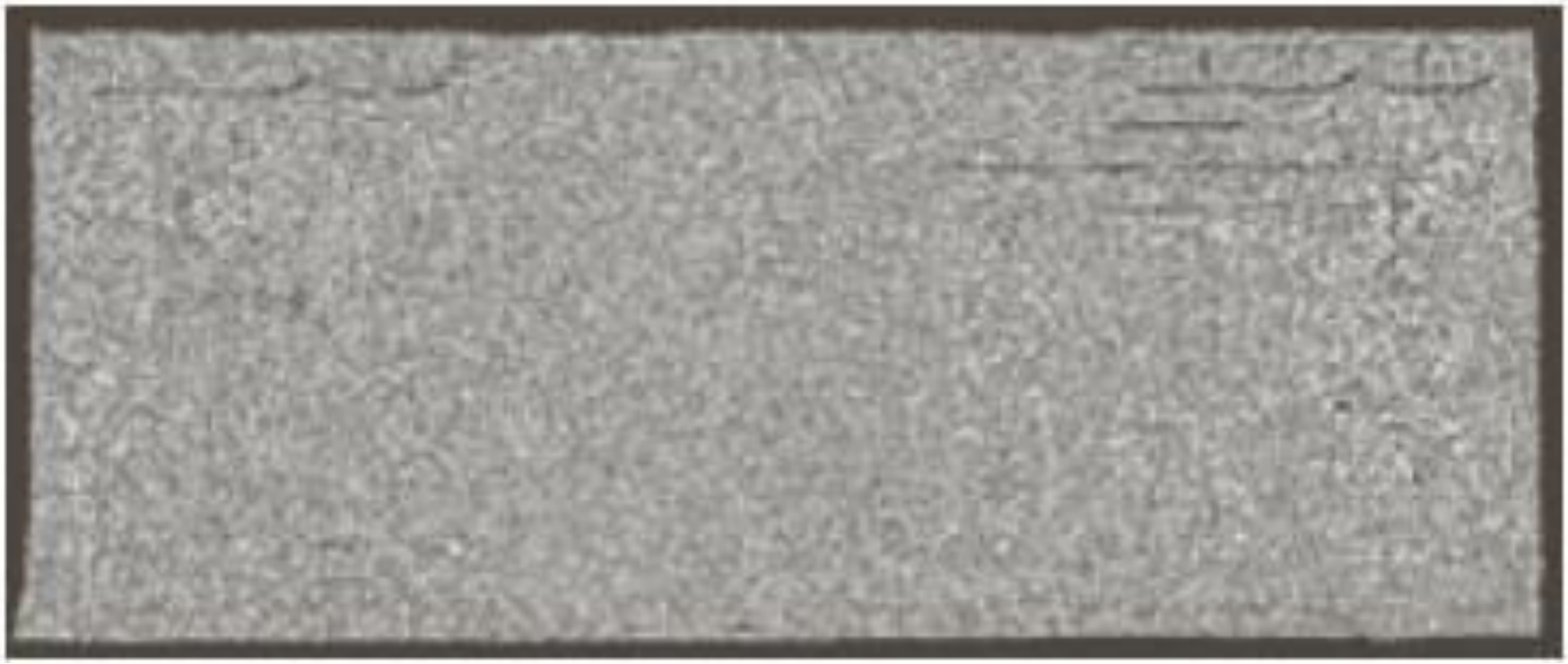
FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

Bit Plane Slicing (cont…)

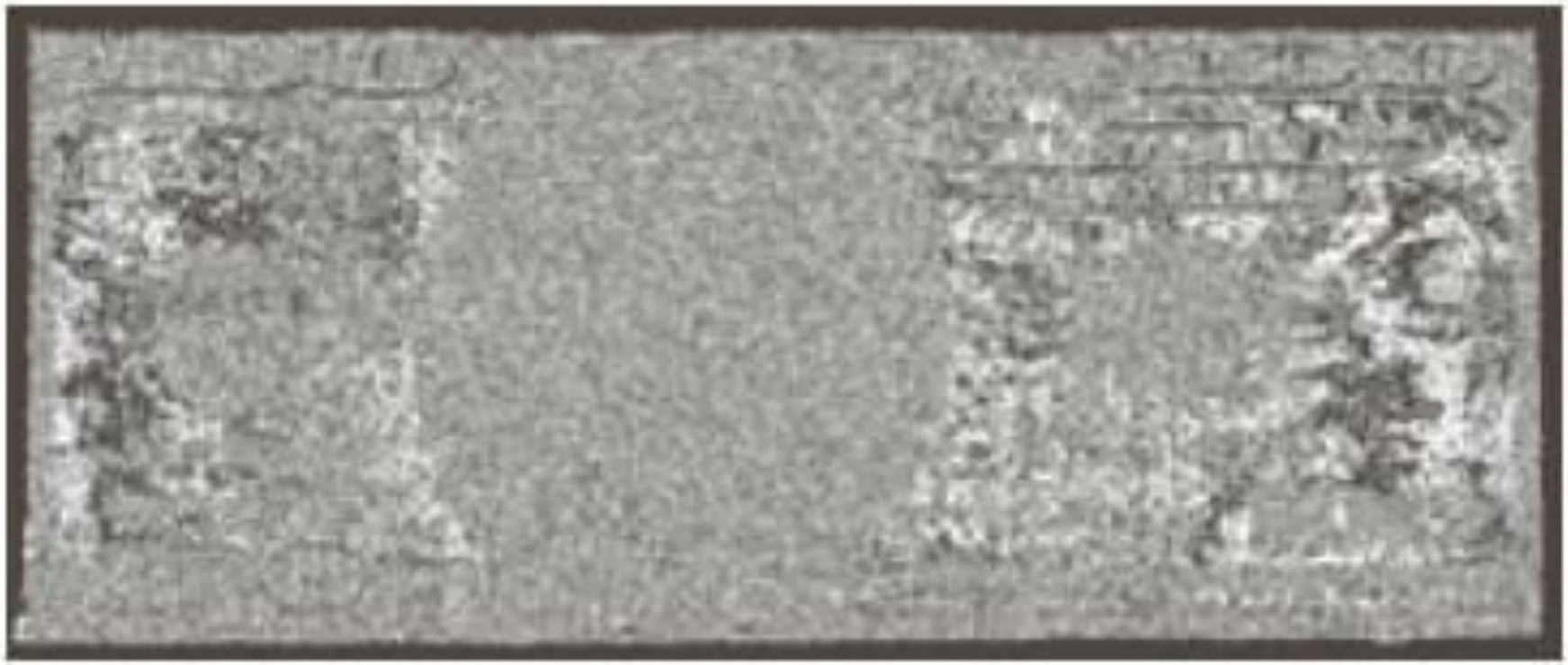
Bit Plane Slicing (cont…)

Bit Plane Slicing (cont…)

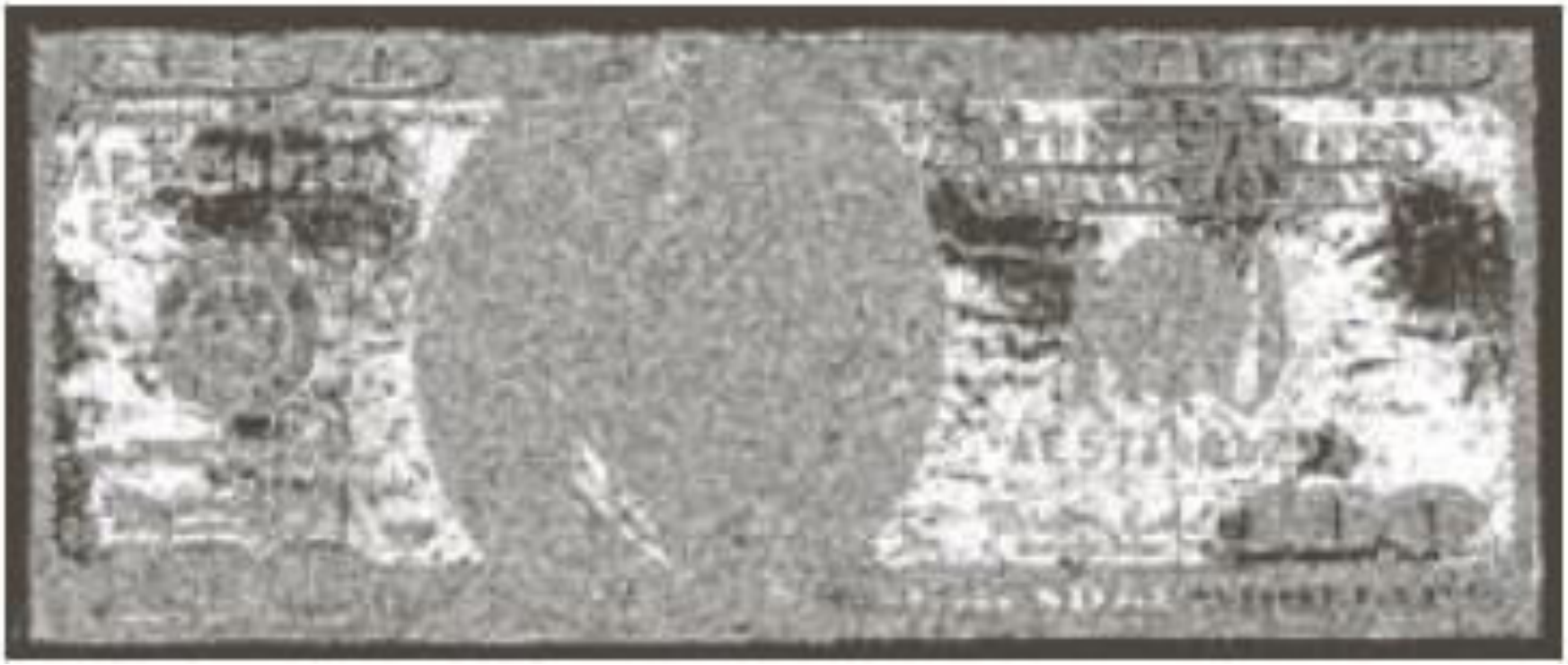
Bit Plane Slicing (cont…)



Bit Plane Slicing (cont…)



Bit Plane Slicing (cont…)



Bit Plane Slicing (cont…)

Bit Plane Slicing (cont…)

Bit Plane Slicing (cont…)

Bit Plane Slicing (cont…)

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 neighbourhood operations – in particular *filtering* and *convolution*

Digital Image Processing

Lecture - 5

Image Enhancement
(Spatial Filtering 1)

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Contents

In this lecture we will look at spatial filtering techniques:

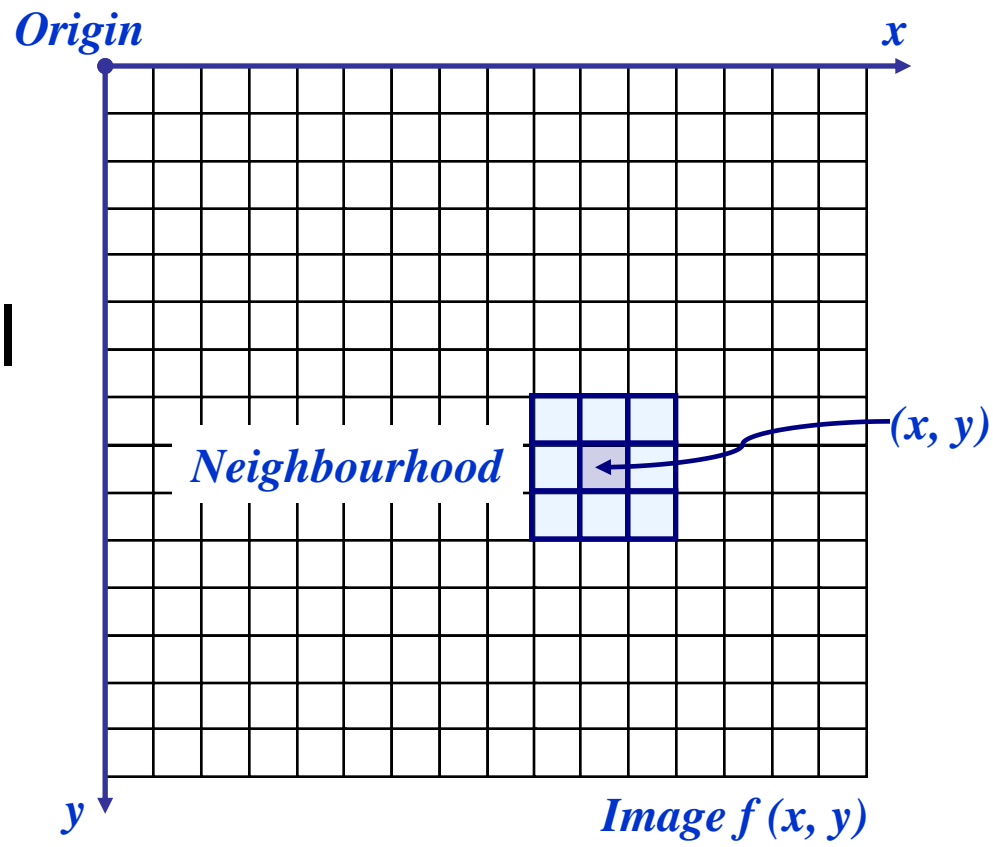
- Neighbourhood operations
- What is spatial filtering?
- Smoothing operations
- What happens at the edges?
- Correlation and convolution

Neighbourhood Operations

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

Neighbourhoods are mostly a rectangle around a central pixel

Any size rectangle and any shape filter are possible



Simple Neighbourhood Operations

Some simple neighbourhood operations include:

- **Min:** Set the pixel value to the minimum in the neighbourhood
- **Max:** Set the pixel value to the maximum in the neighbourhood
- **Median:** The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median). Sometimes the median works better than the average

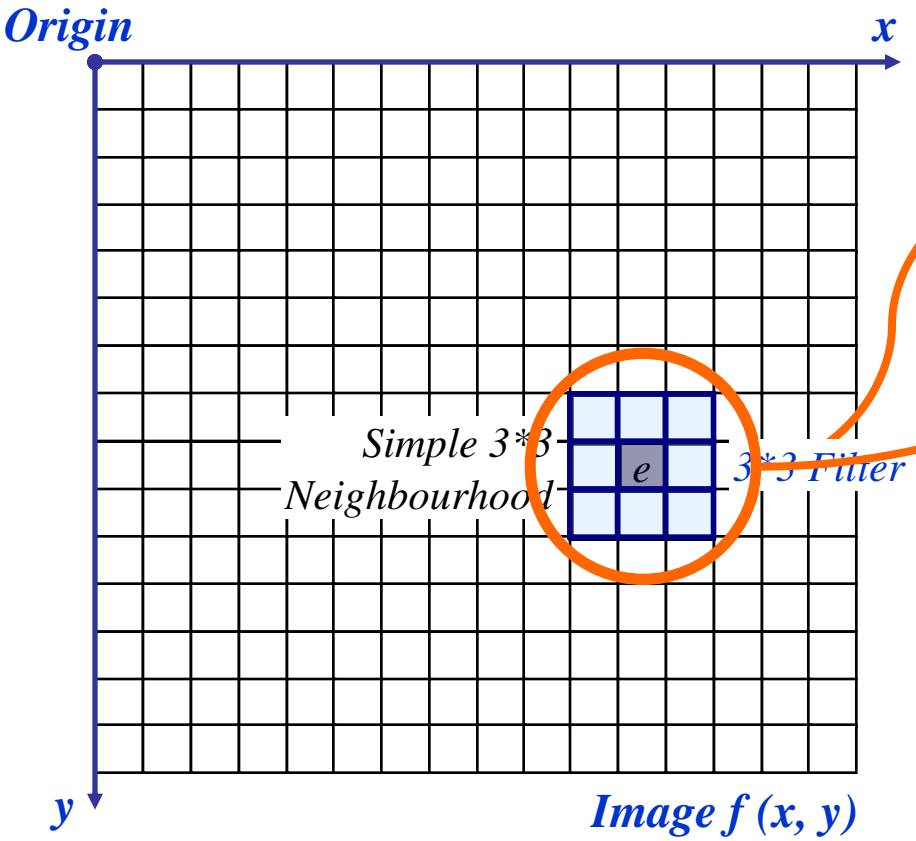
Simple Neighbourhood Operations Example

Original Image

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

Enhanced Image

The Spatial Filtering Process



a	b	c
d	e	f
g	h	i

Original Image Pixels

*

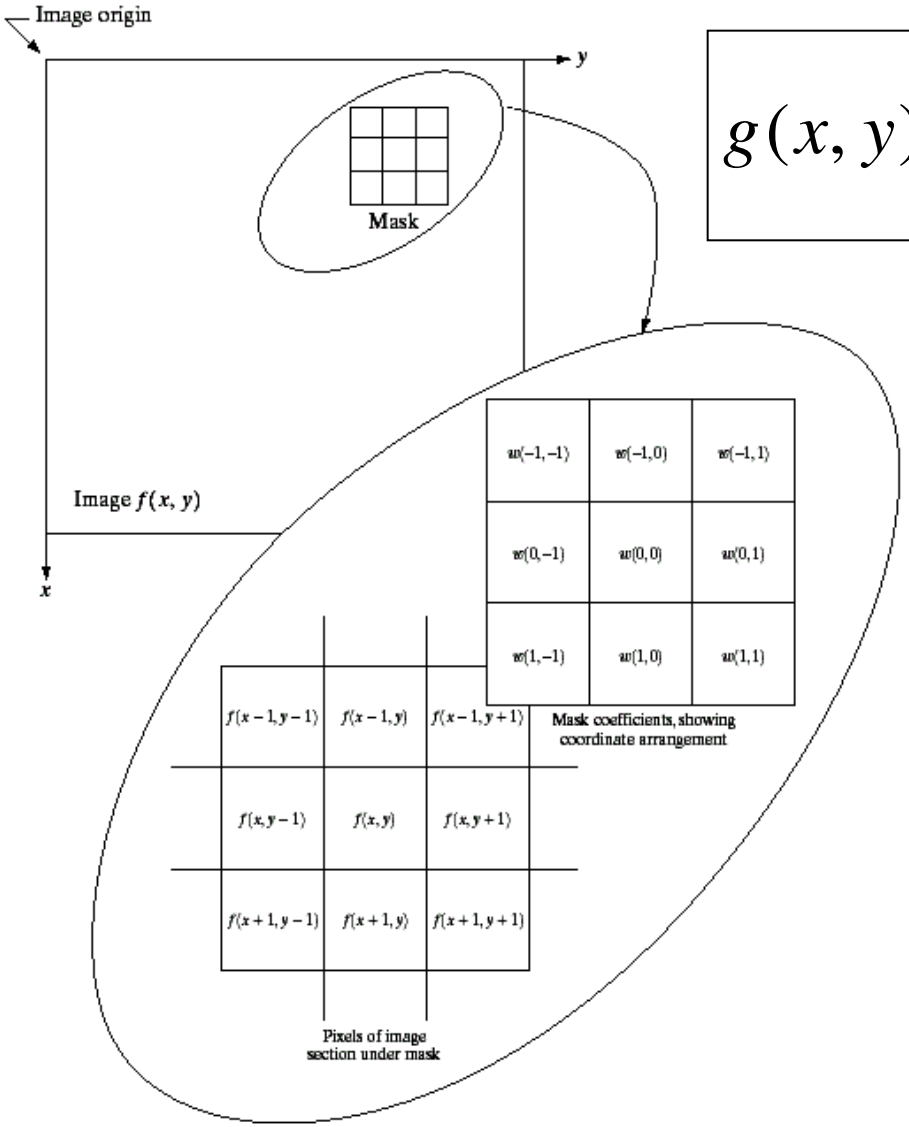
r	s	t
u	v	w
x	y	z

Filter

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

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

Spatial Filtering: Equation Form



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

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

Smoothing Spatial Filters

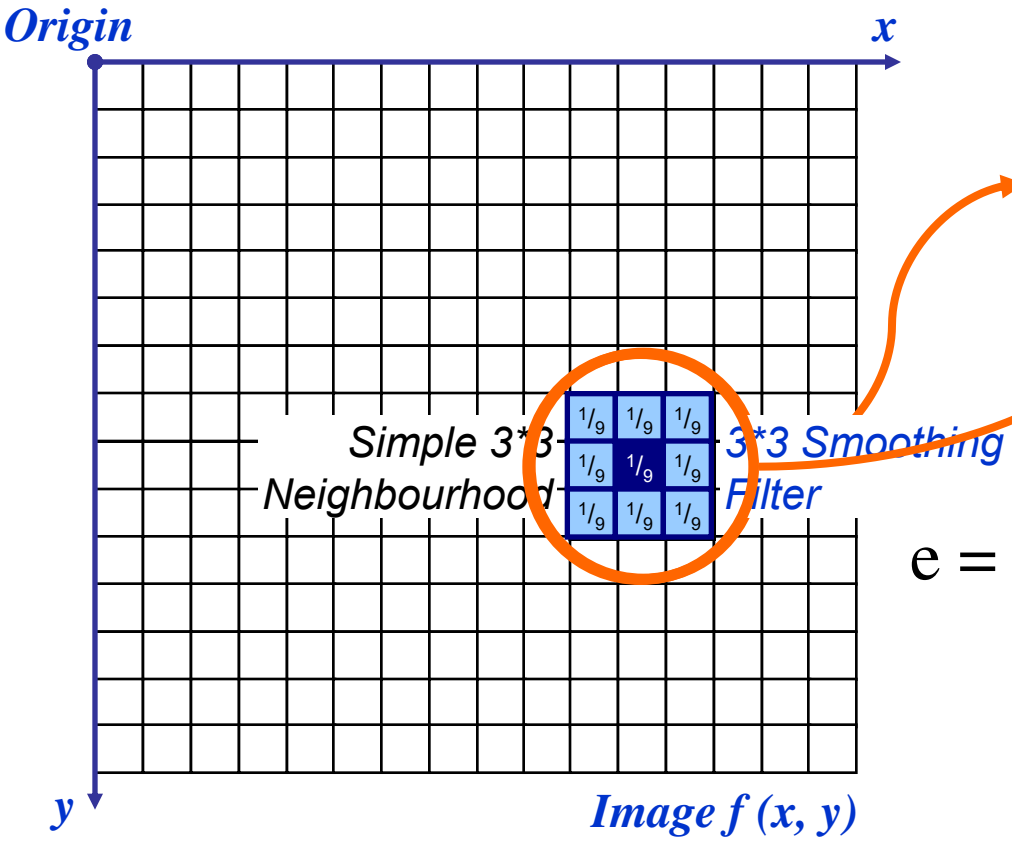
One of the simplest spatial filtering operations we can perform is a smoothing operation

- 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

$$
\begin{aligned}
e = {} & \tfrac{1}{9}*106 + \\
& \tfrac{1}{9}*104 + \tfrac{1}{9}*100 + \tfrac{1}{9}*108 + \\
& \tfrac{1}{9}*99 + \tfrac{1}{9}*98 + \\
& \tfrac{1}{9}*95 + \tfrac{1}{9}*90 + \tfrac{1}{9}*85 \\
= {} & 98.3333
\end{aligned}
$$

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

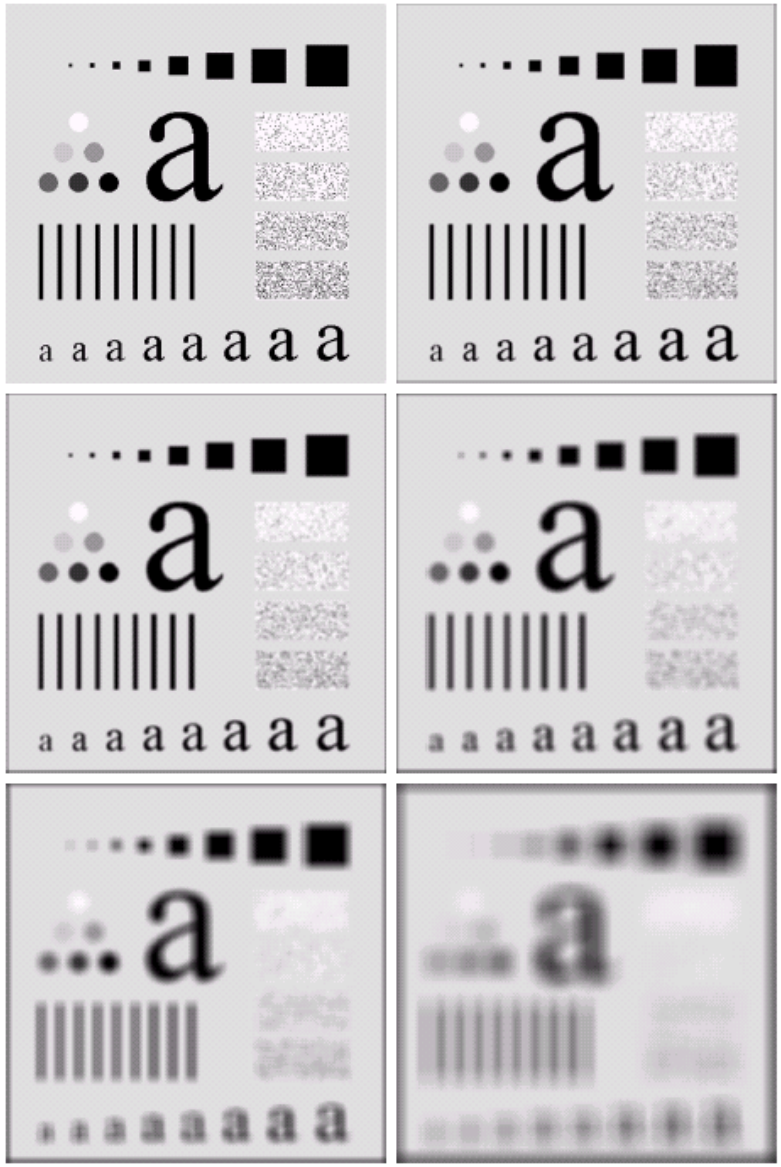


Image Smoothing Example

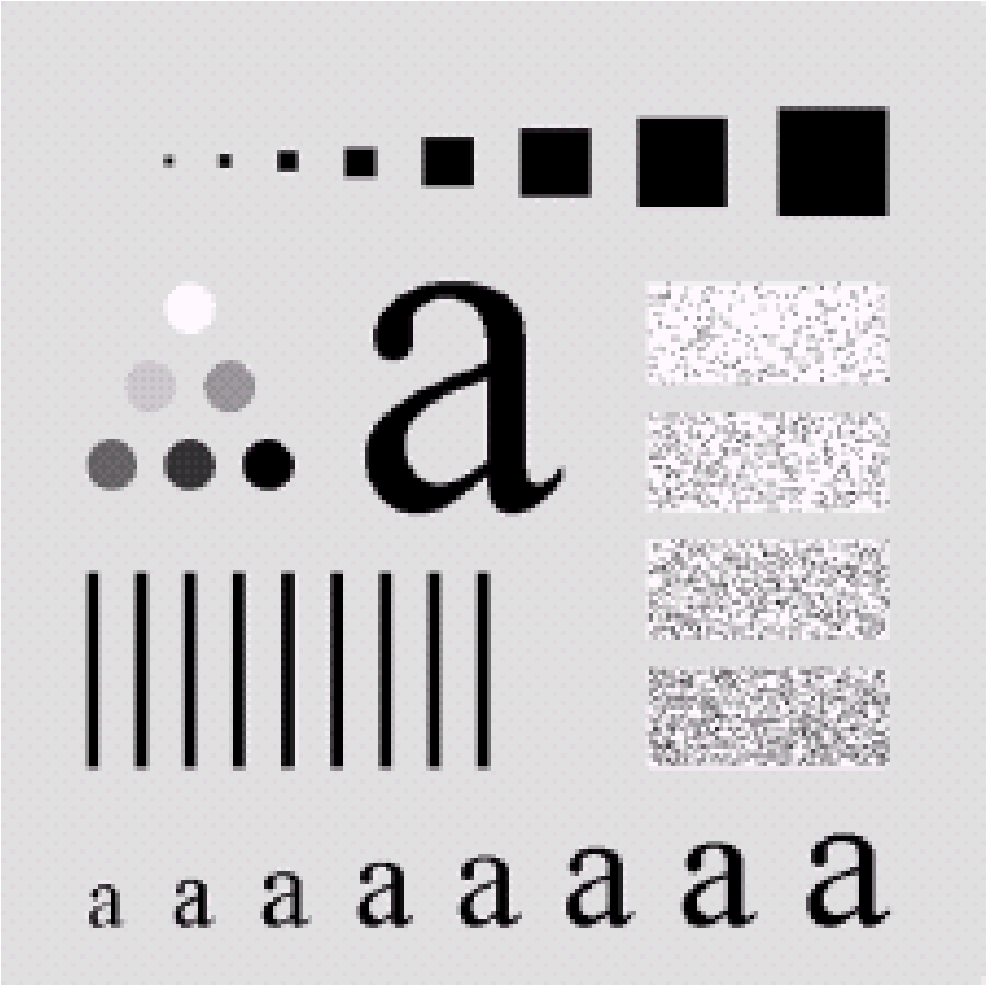


Image Smoothing Example

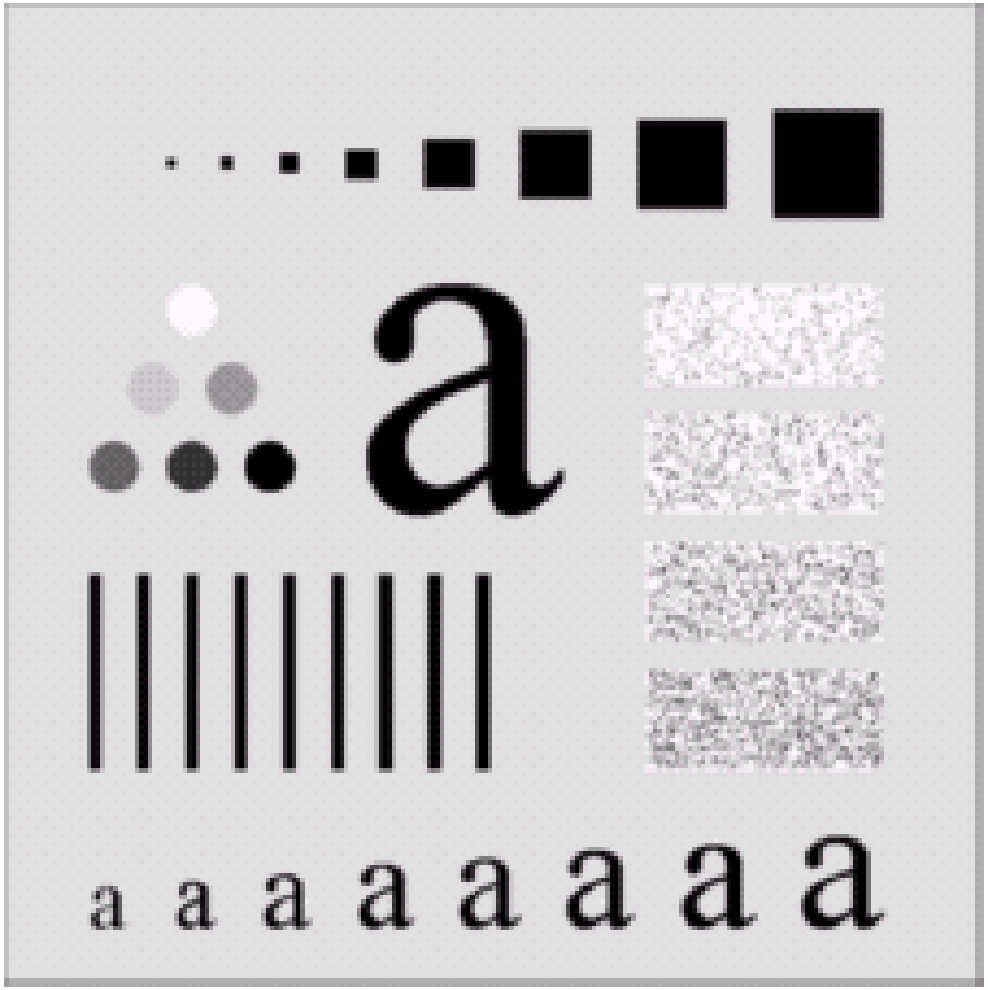


Image Smoothing Example

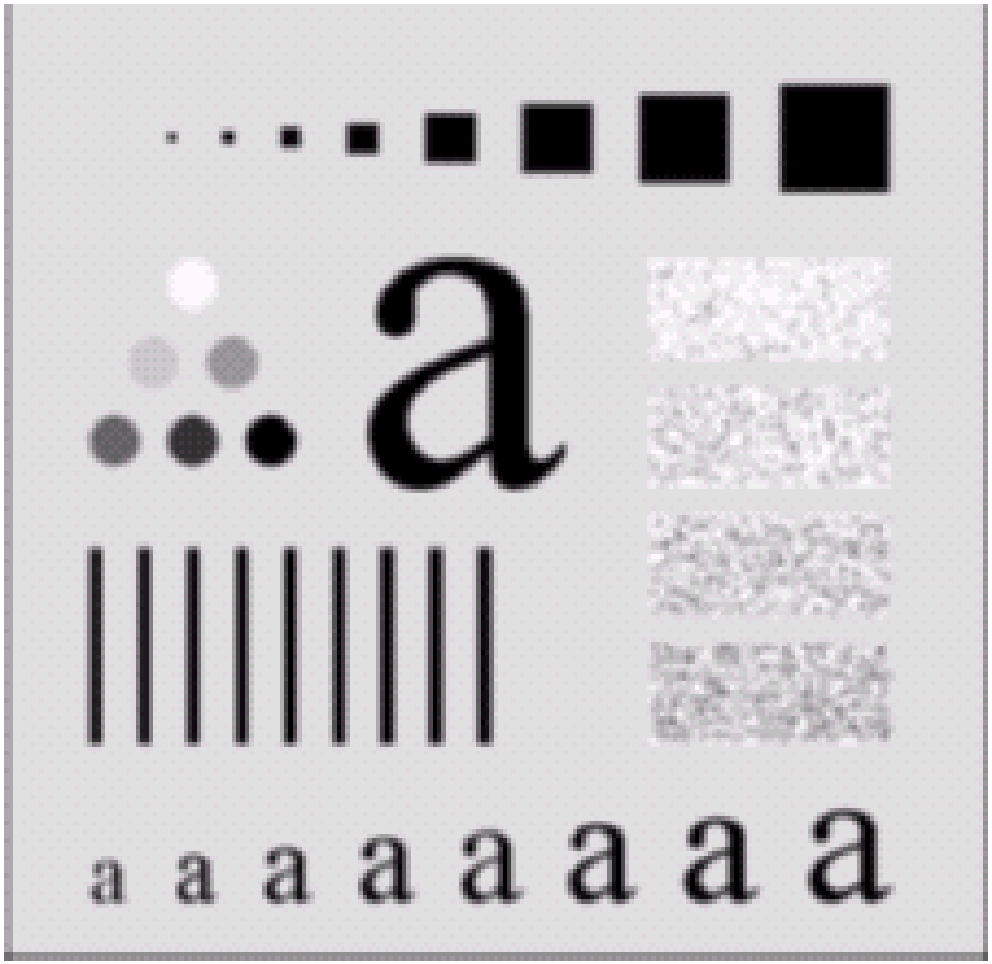


Image Smoothing Example

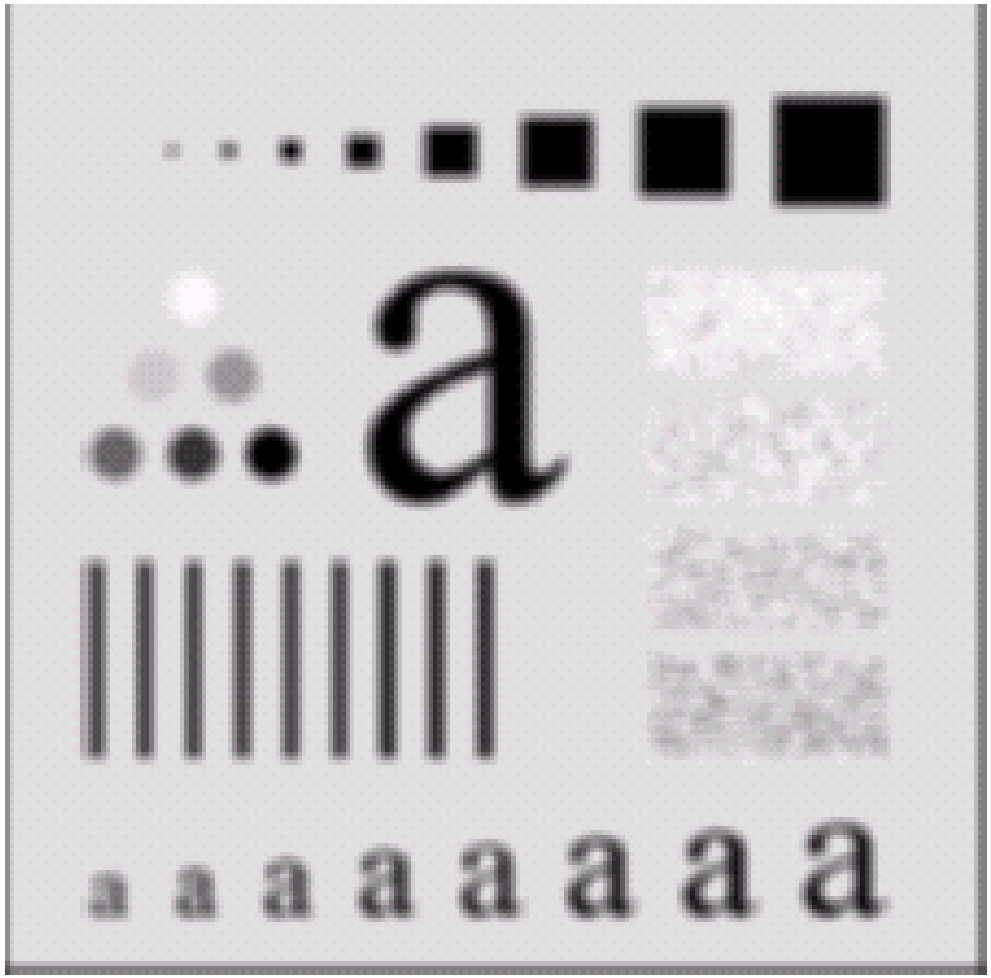


Image Smoothing Example

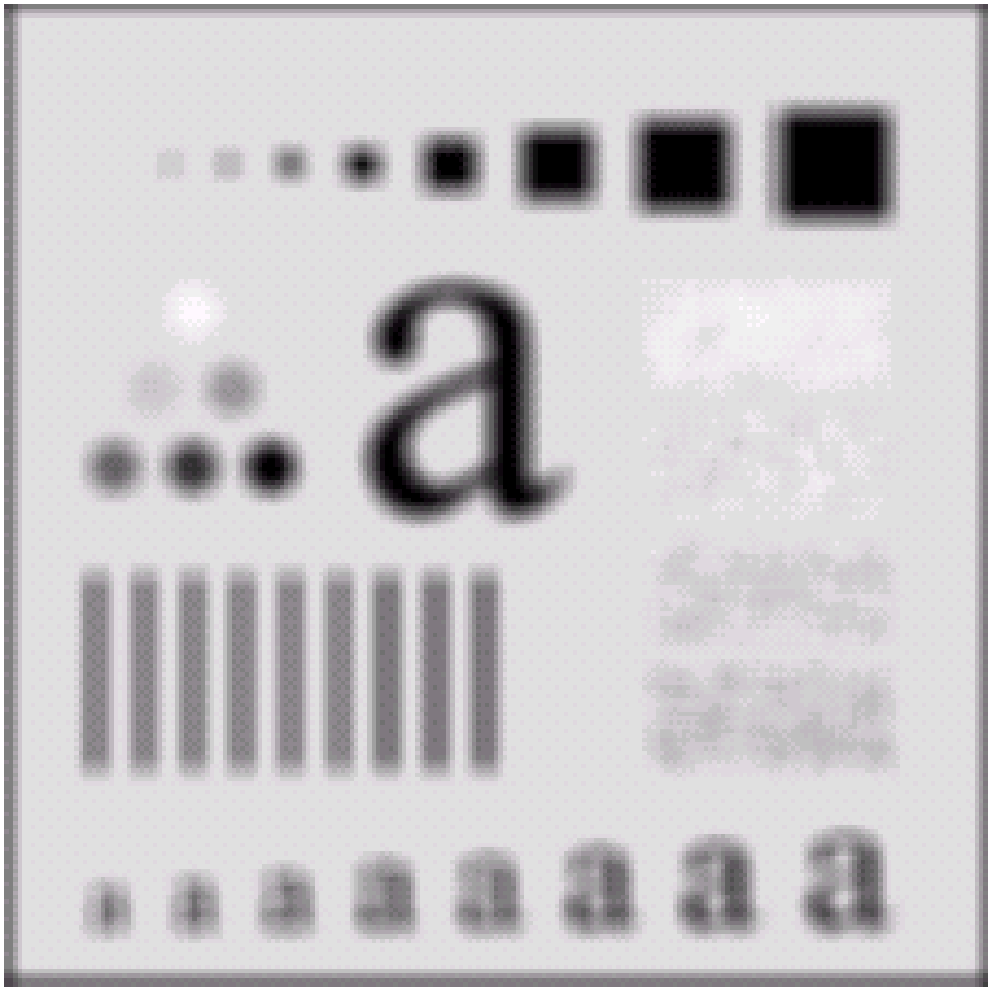
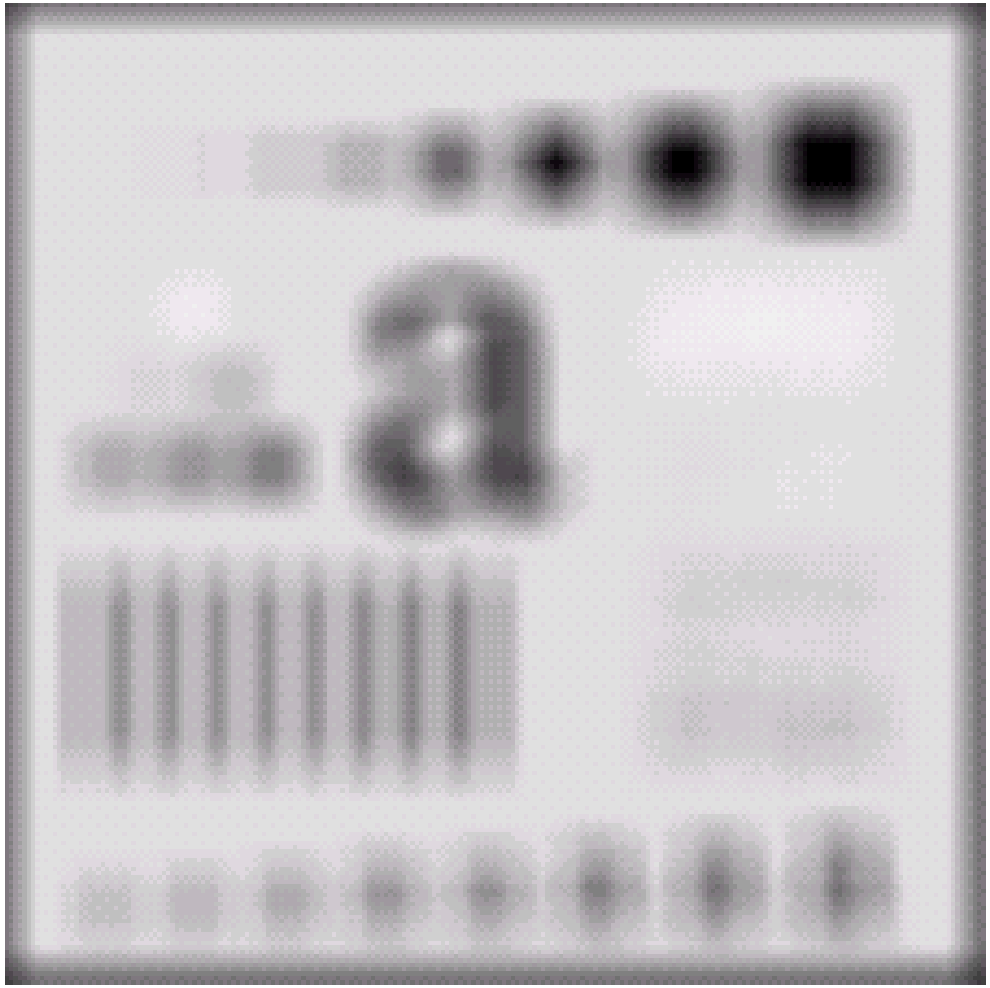


Image Smoothing Example



Weighted Smoothing Filters

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

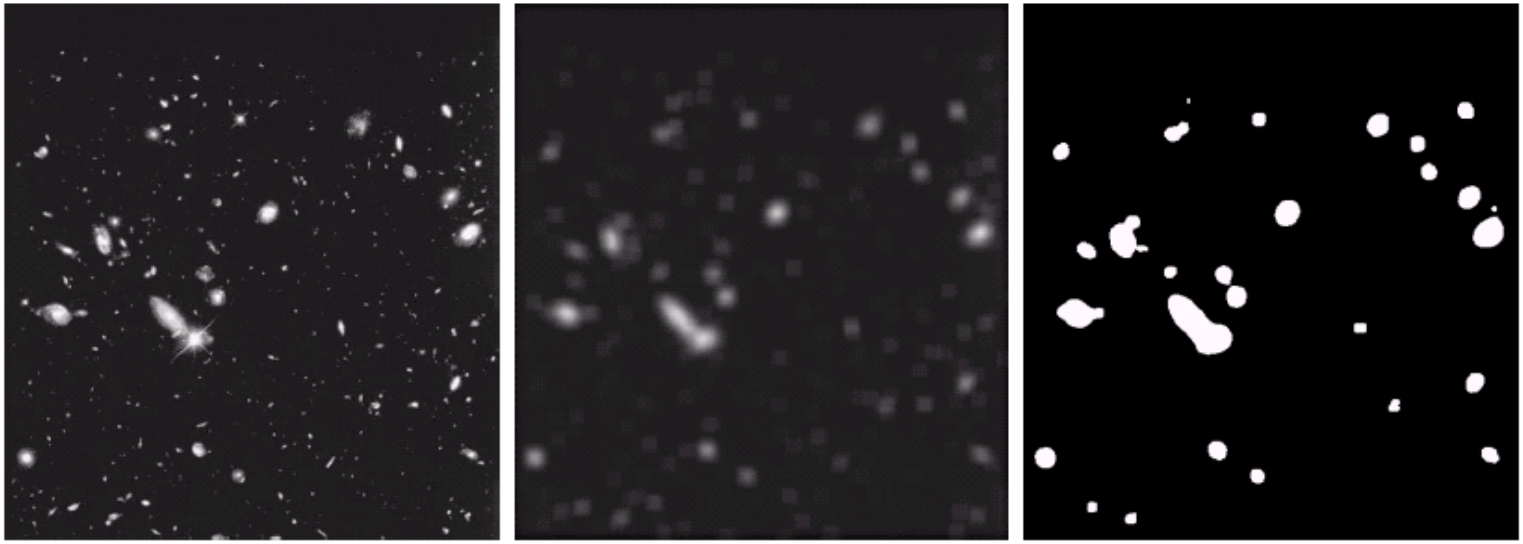
- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

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

Weighted averaging filter

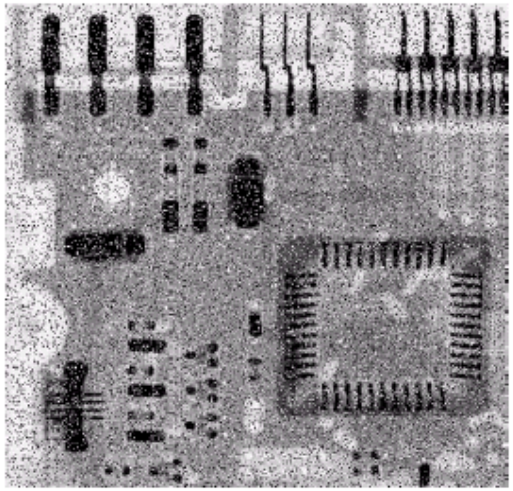
Another Smoothing Example

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

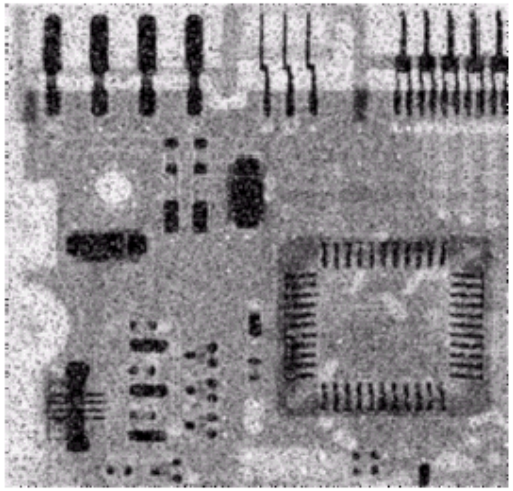


Image After Averaging Filter

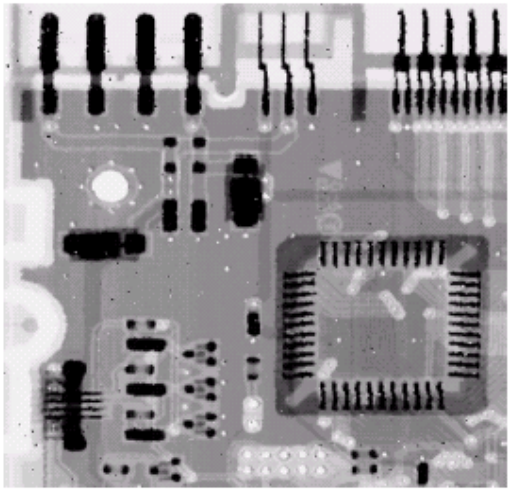
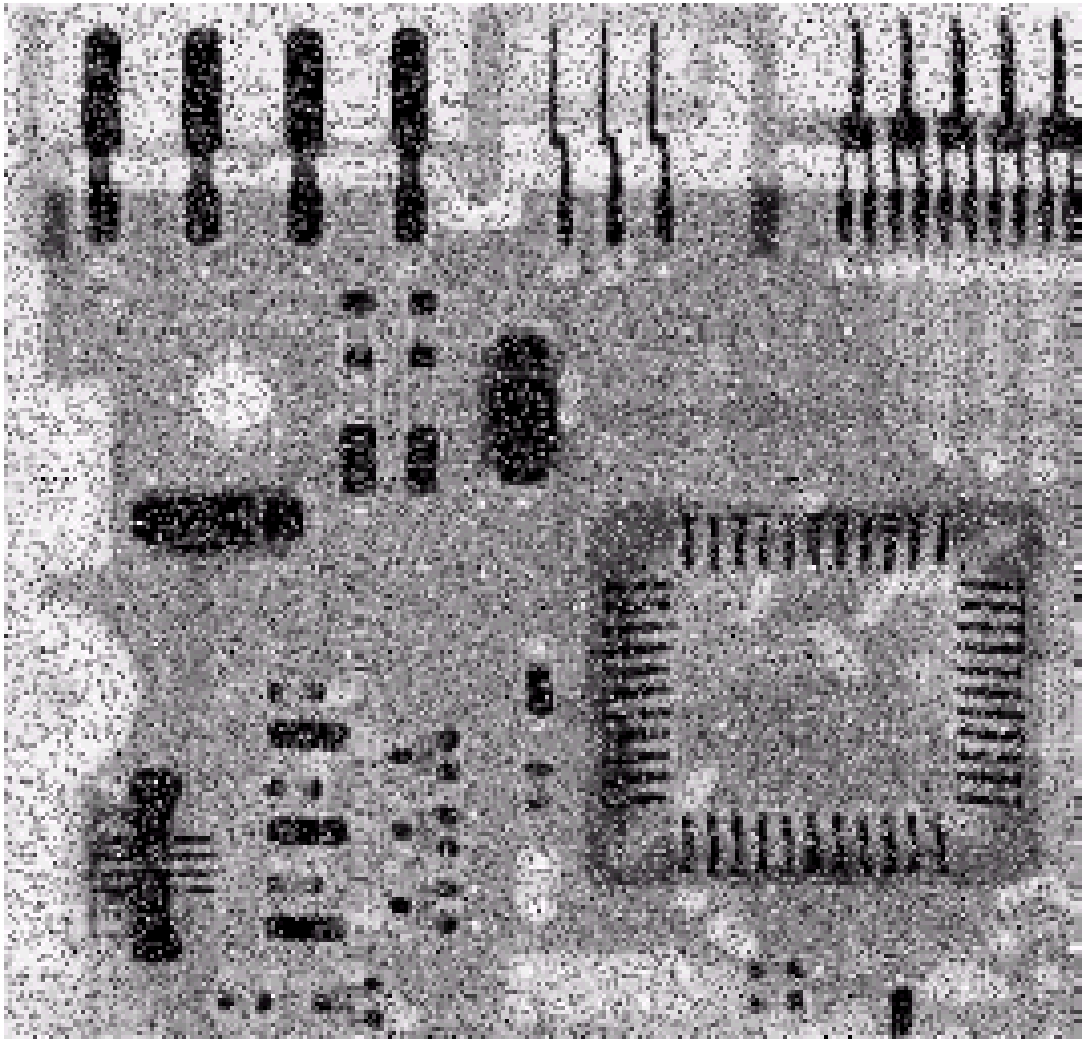


Image After Median Filter

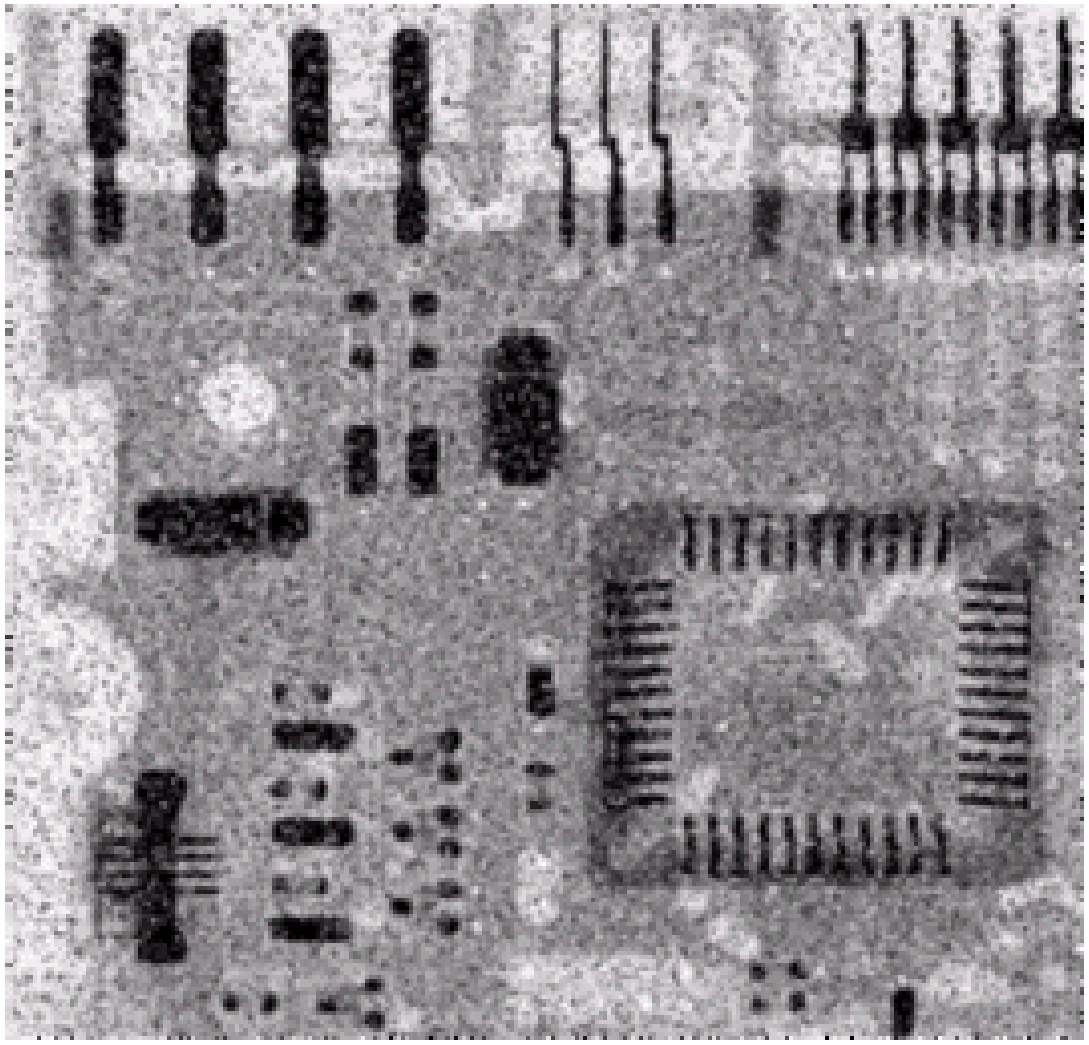
Filtering is often used to remove noise from images

Sometimes a median filter works better than an averaging filter

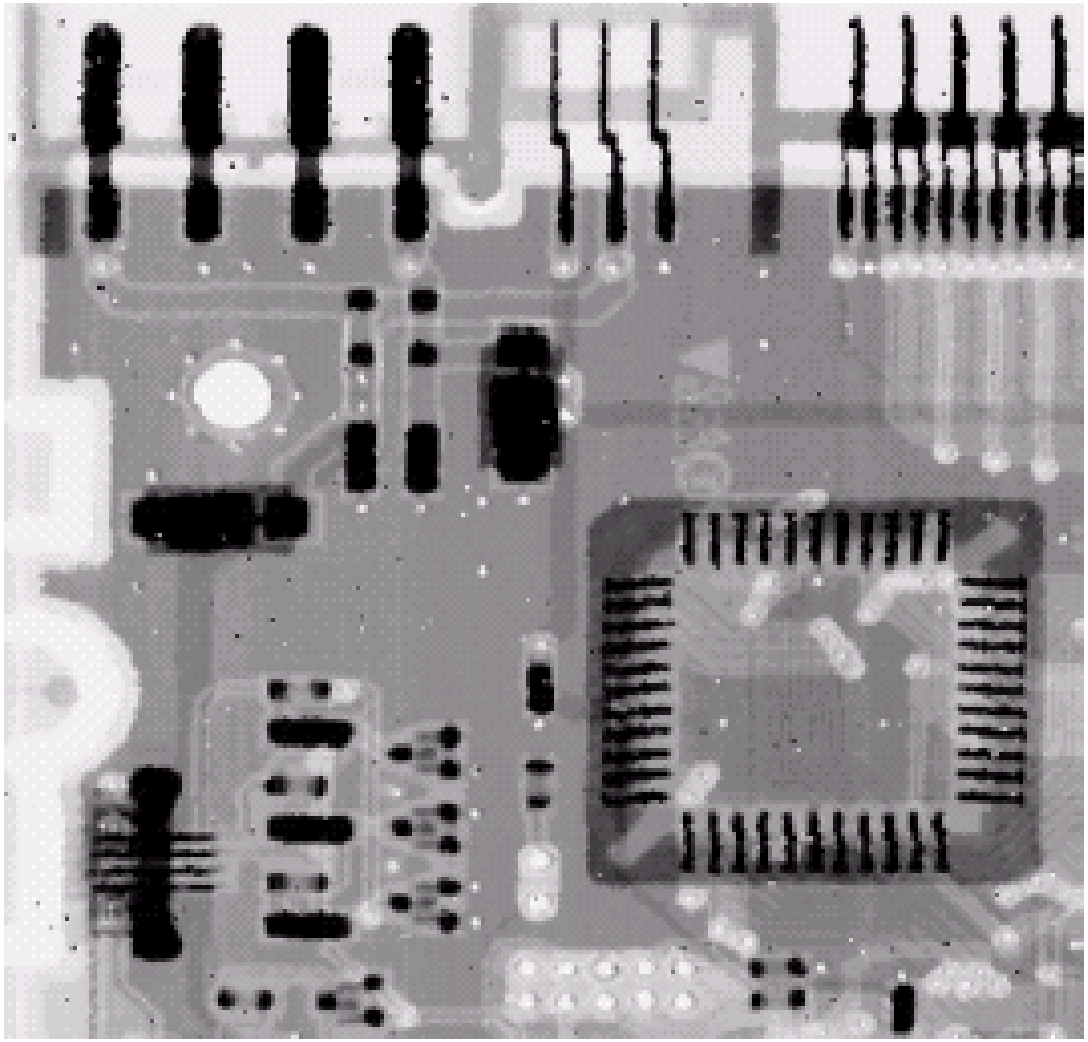
Averaging Filter Vs. Median Filter Example



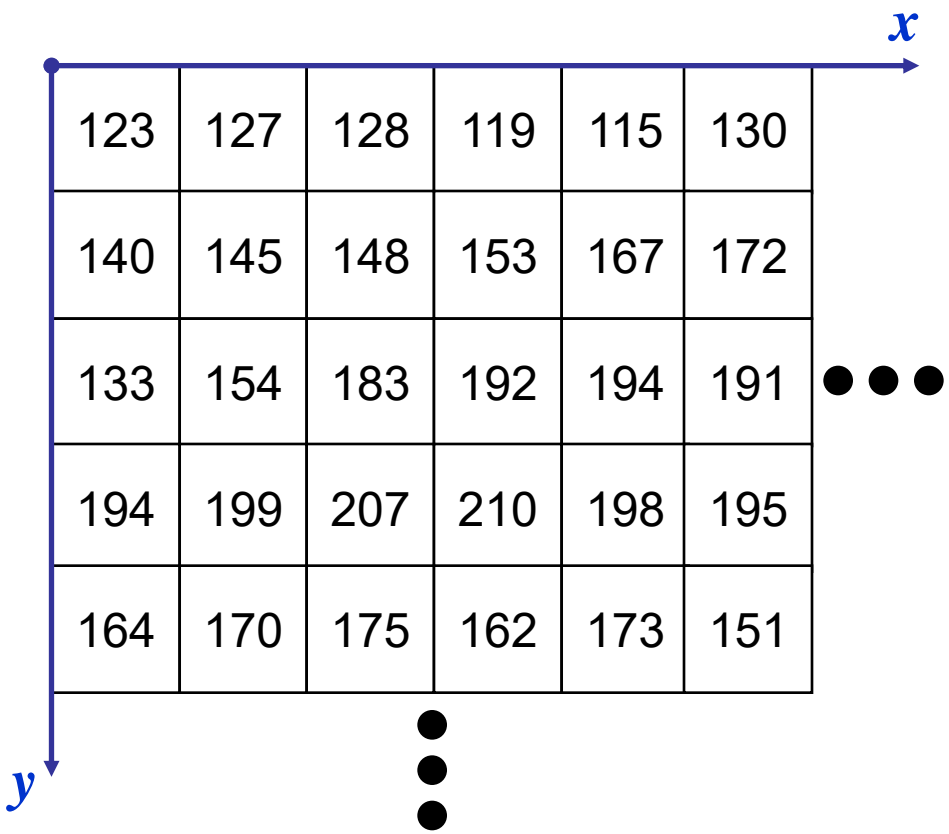
Averaging Filter Vs. Median Filter Example



Averaging Filter Vs. Median Filter Example

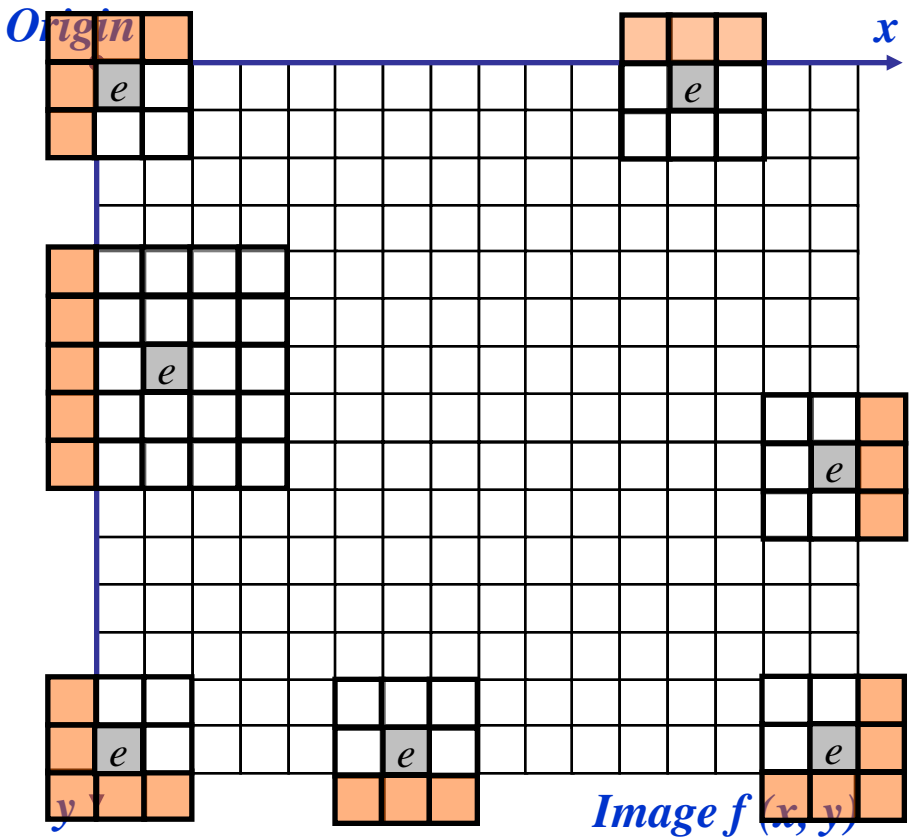


Simple Neighbourhood Operations Example



Strange Things Happen At The Edges!

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

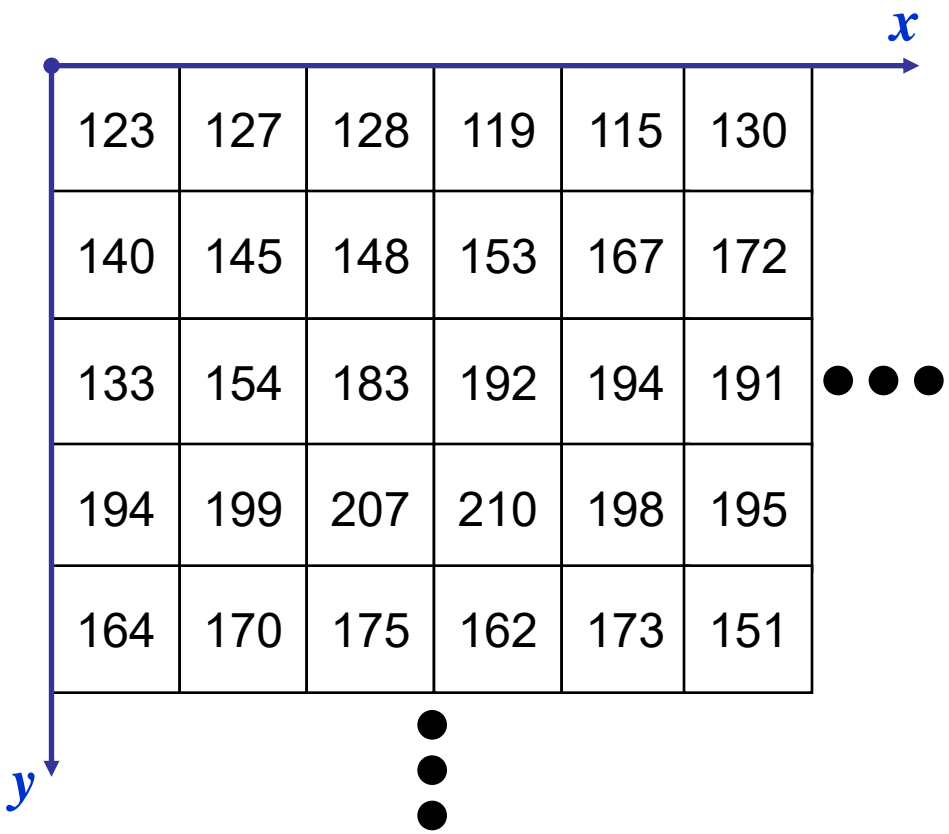


Strange Things Happen At The Edges! (cont…)

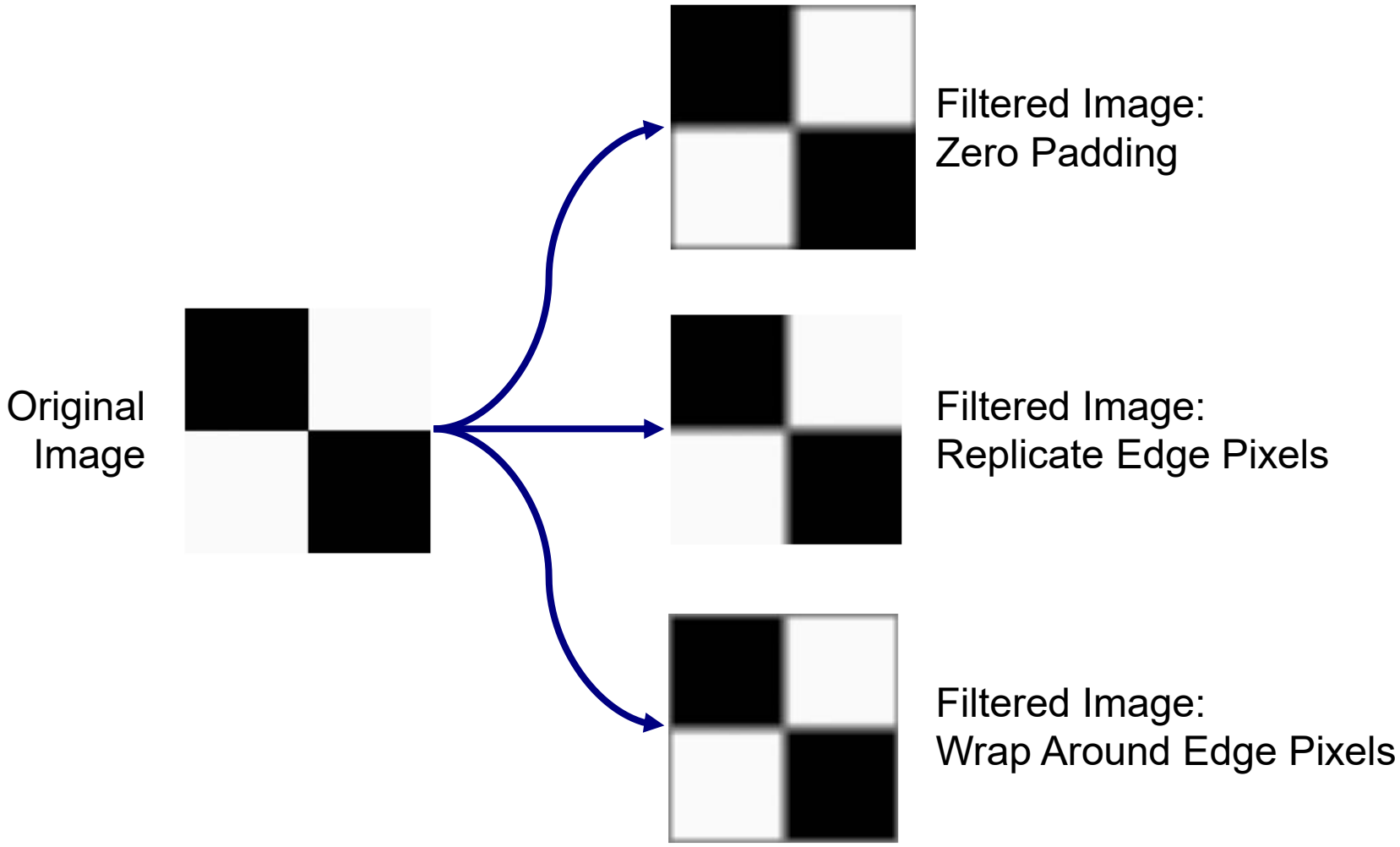
There are a few approaches to dealing with missing edge pixels:

- Omit missing pixels
  - Only works with some filters
  - Can add extra code and slow down processing
- Pad the image
  - Typically with either all white or all black pixels
- Replicate border pixels
- Allow pixels *wrap around* the image
  - Can cause some strange image artefacts

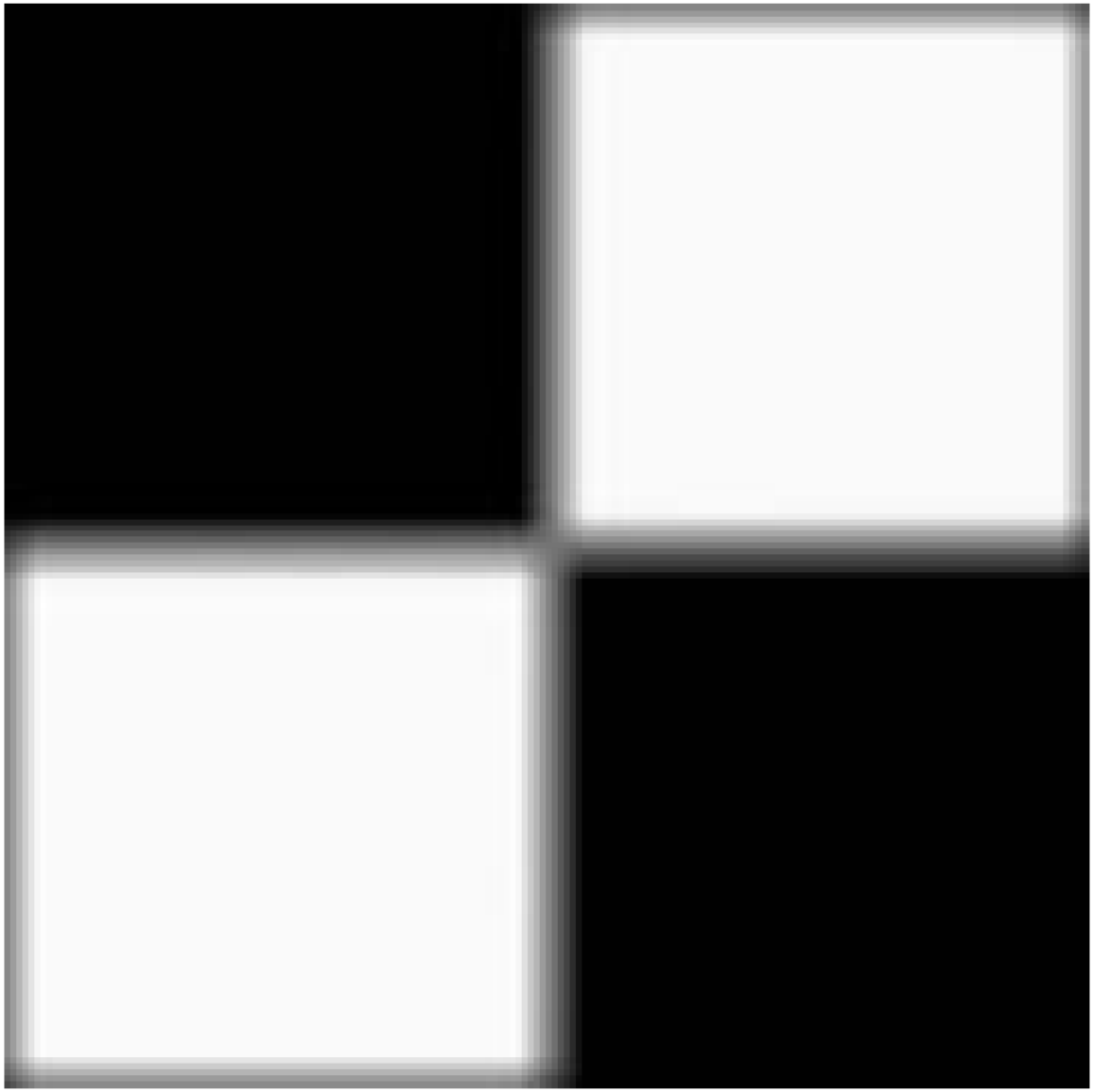
Simple Neighbourhood Operations Example



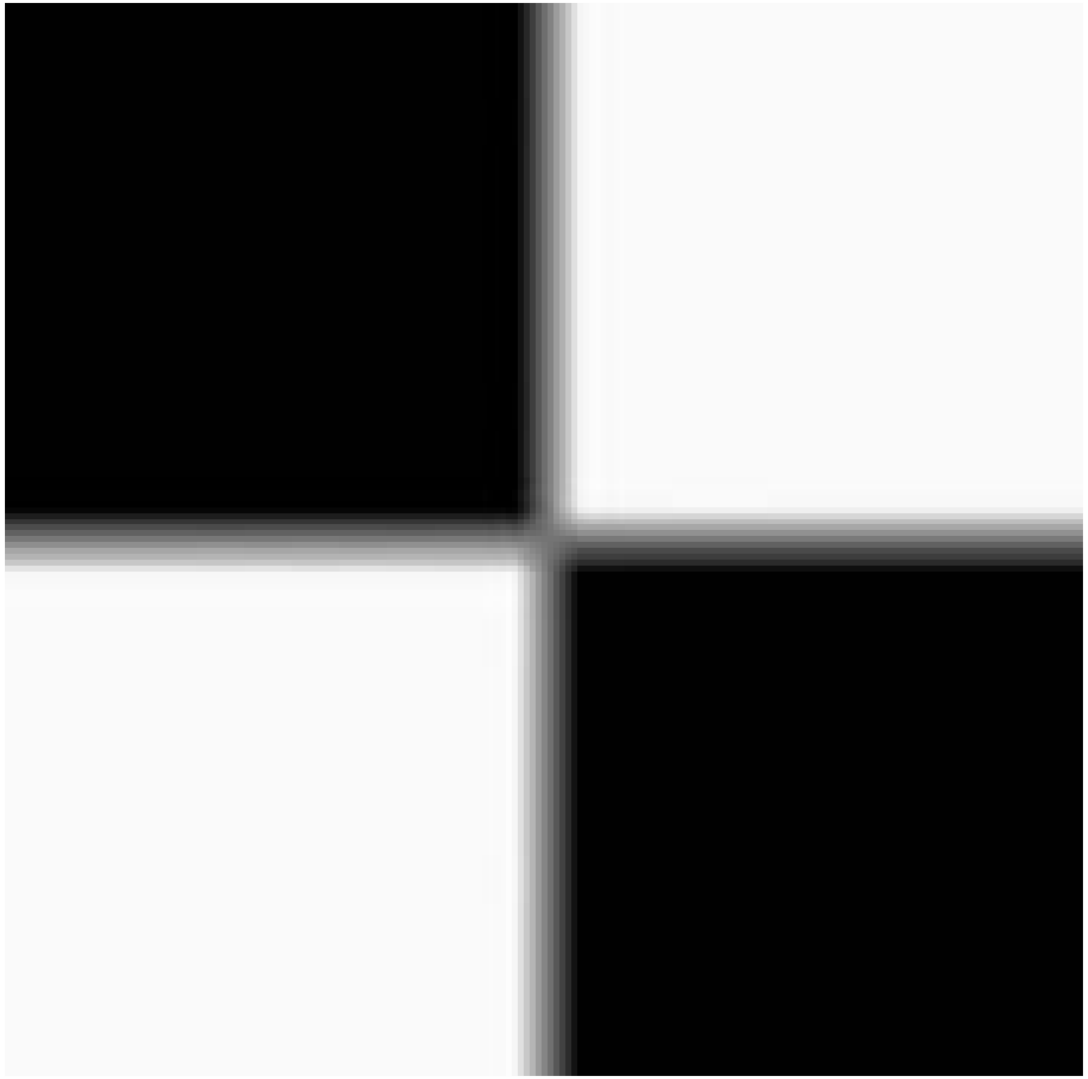
Strange Things Happen At The Edges! (cont…)



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	e
f	g	h

Original Image Pixels

$*$

r	s	t
u	v	w
x	y	z

Filter

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

For symmetric filters it makes no difference

Summary

In this lecture we have looked at the idea of spatial filtering and in particular:

- Neighbourhood operations
- The filtering process
- Smoothing filters
- Dealing with problems at image edges when using filtering
- Correlation and convolution

Next time we will looking at sharpening filters and more on filtering and image enhancement

Digital Image Processing

Lecture - 6

Image Enhancement
(Spatial Filtering 2)

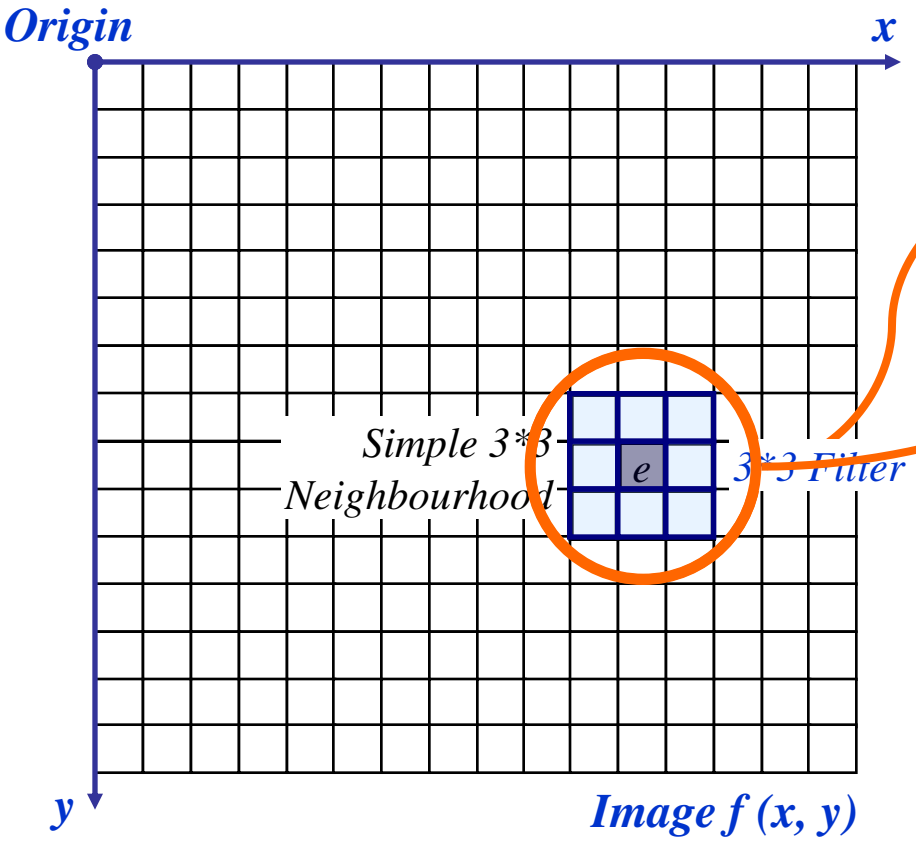
Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Contents

In this lecture we will look at more spatial filtering techniques

- Spatial filtering refresher
- Sharpening filters
  - 1st derivative filters
  - 2nd derivative filters
- Combining filtering techniques

Spatial Filtering Refresher



a	b	c
d	e	f
g	h	i

Original Image Pixels

*

r	s	t
u	v	w
x	y	z

Filter

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

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

Sharpening Spatial Filters

Previously we have looked at smoothing filters which remove fine detail

Sharpening spatial filters seek to highlight fine detail

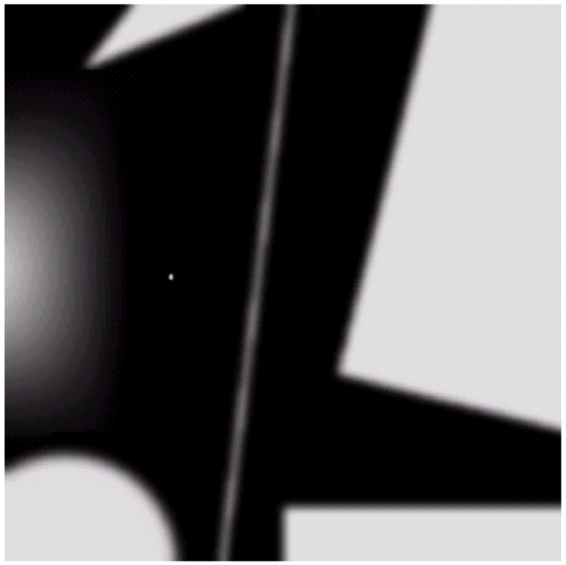
- Remove blurring from images
- Highlight edges

Sharpening filters are based on *spatial differentiation*

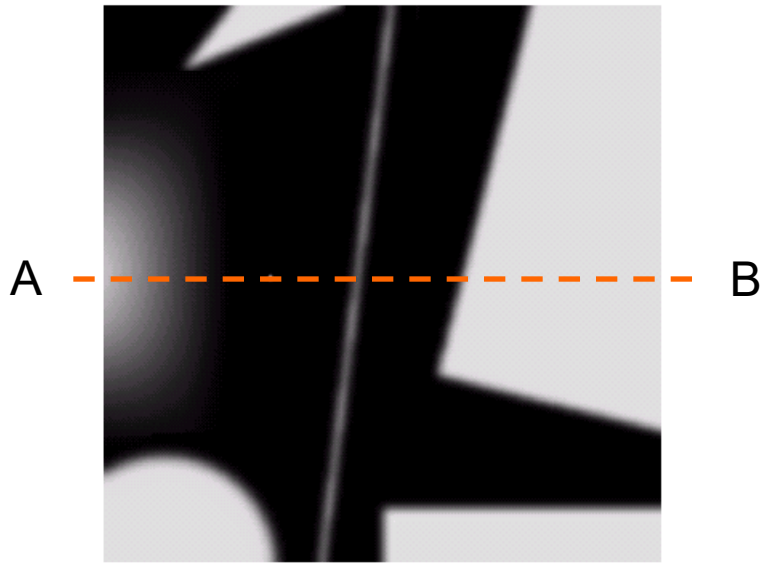
Spatial Differentiation

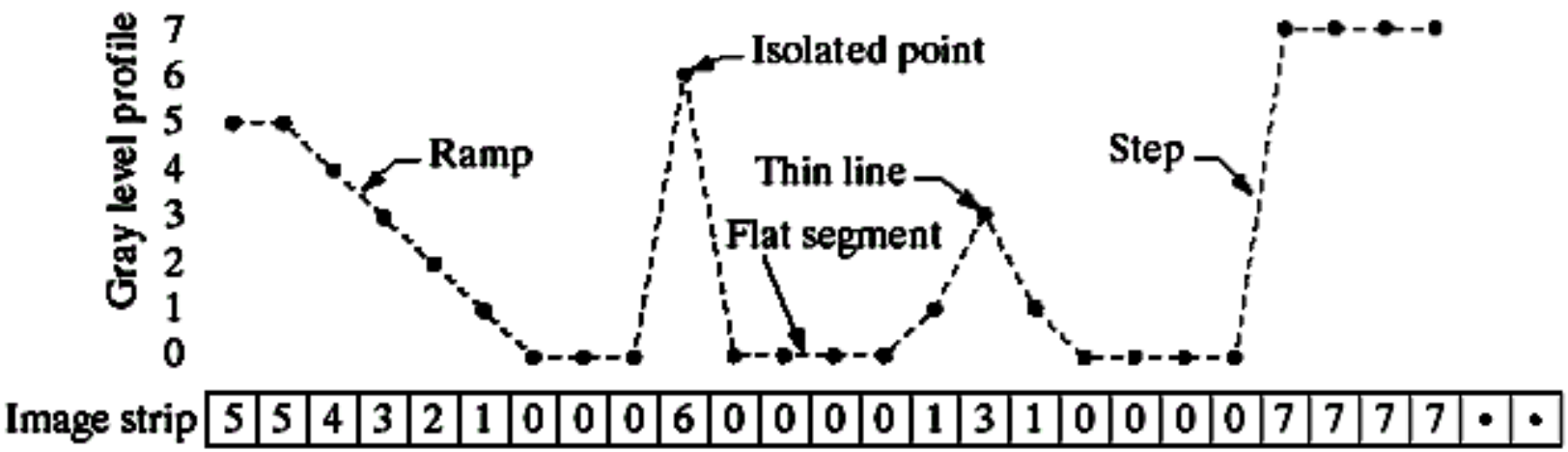
Differentiation measures the *rate of change* of a function

Let's consider a simple 1 dimensional example



Spatial Differentiation





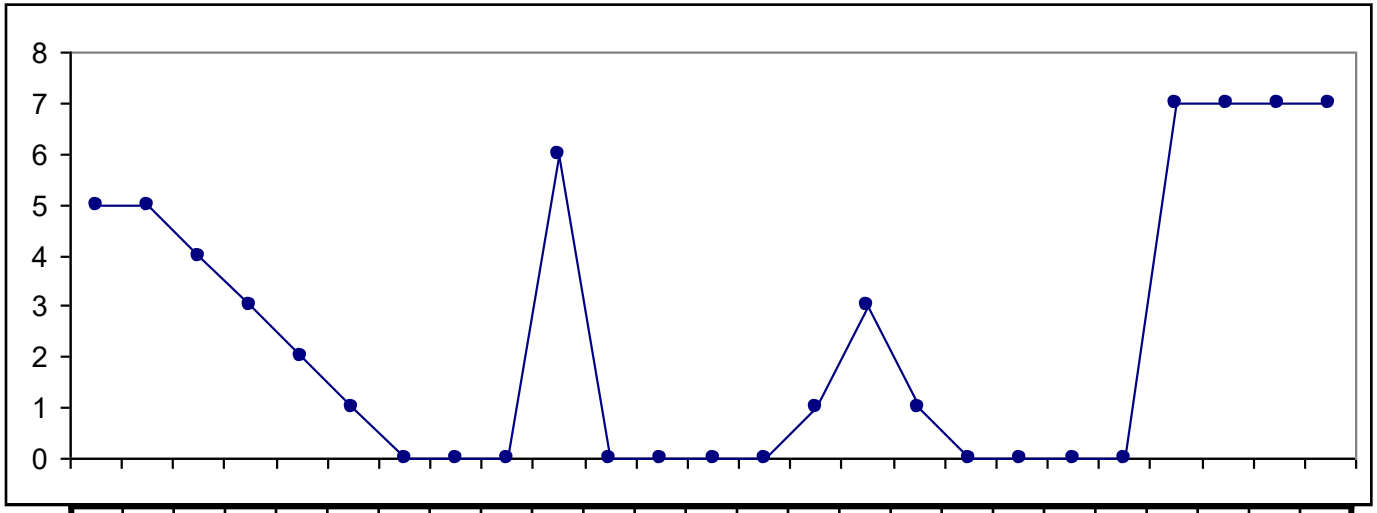
1st Derivative

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

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

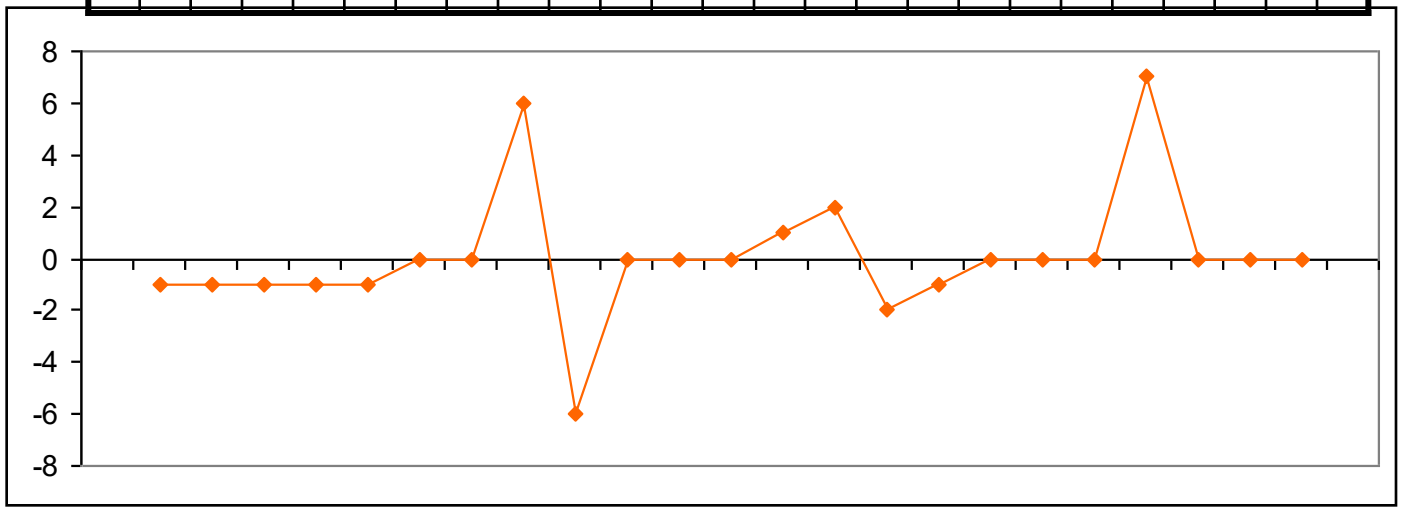
It's just the difference between subsequent values and measures the rate of change of the function

1st 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	



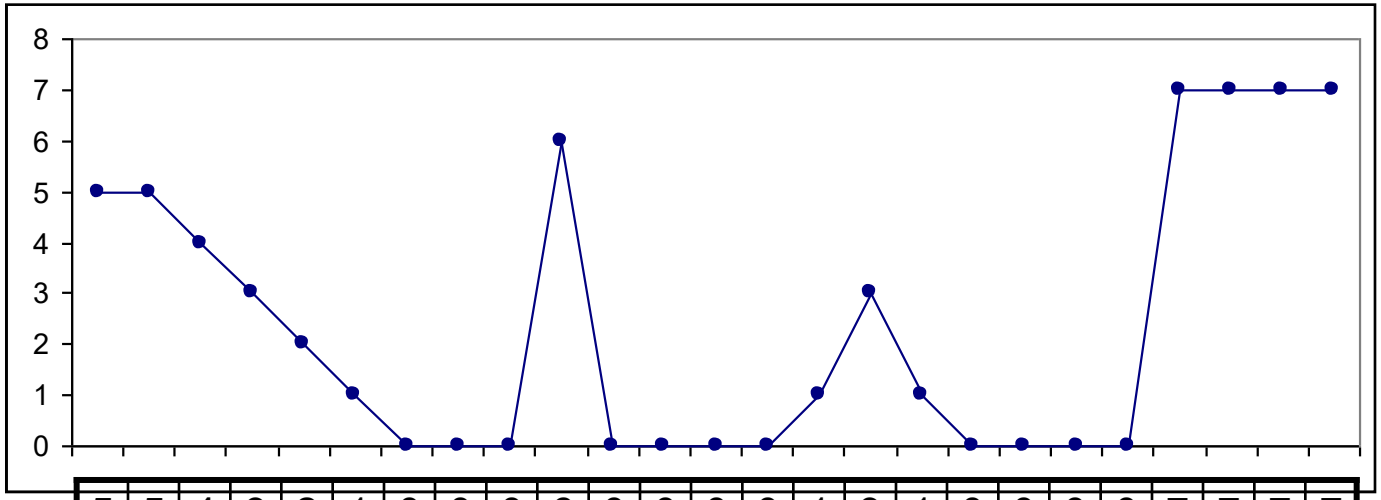
2nd Derivative

The formula for the 2nd derivative of a function is as follows:

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

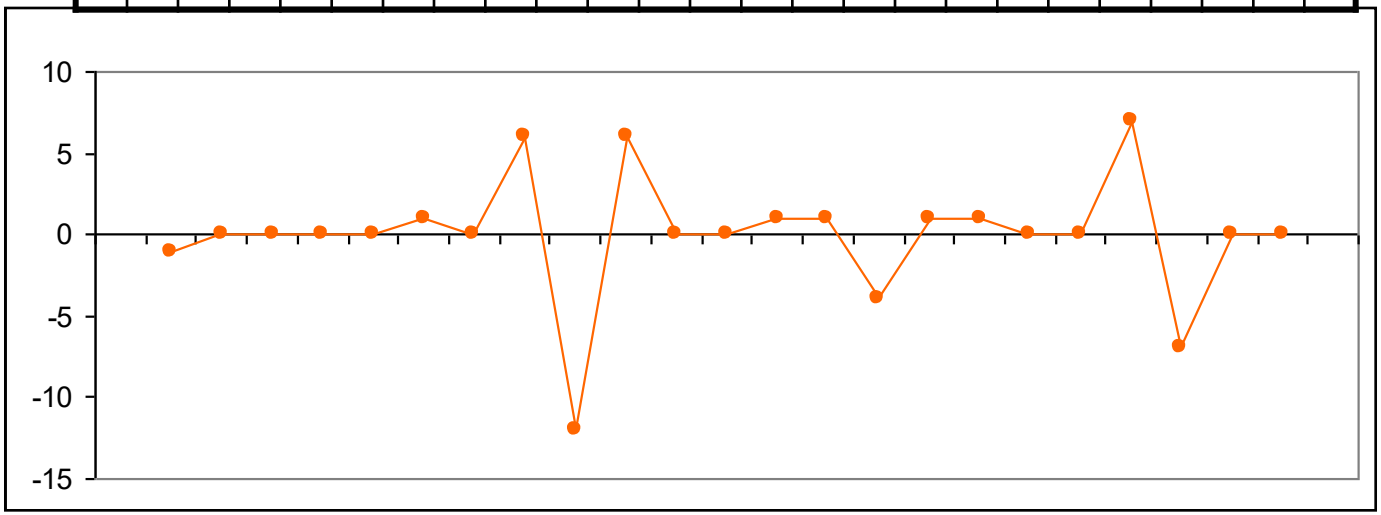
Simply takes into account the values both before and after the current value

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 2nd derivative is more useful for image enhancement than the 1st derivative

- Stronger response to fine detail
- Simpler implementation
- We will come back to the 1st order derivative later on

The first sharpening filter we will look at is the *Laplacian*

- 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:

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

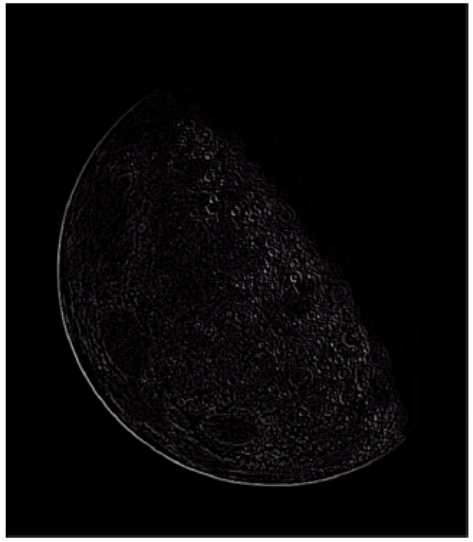
We can easily build a filter based on this

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

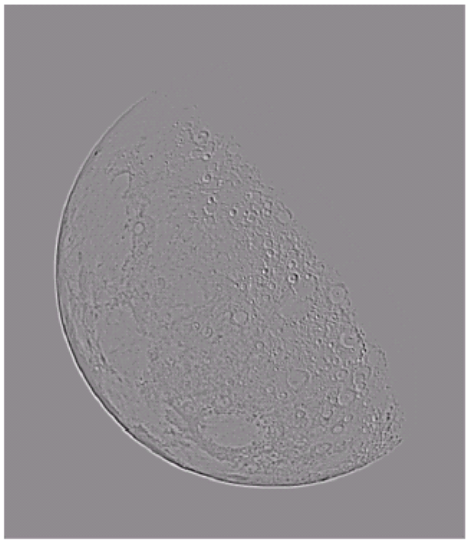
But That Is Not Very Enhanced!

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

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

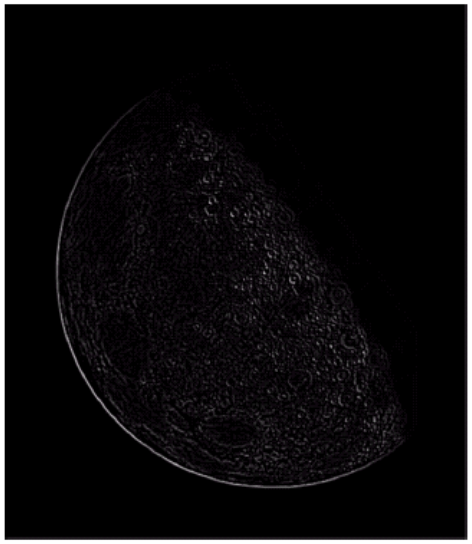


Laplacian
Filtered Image
Scaled for Display

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

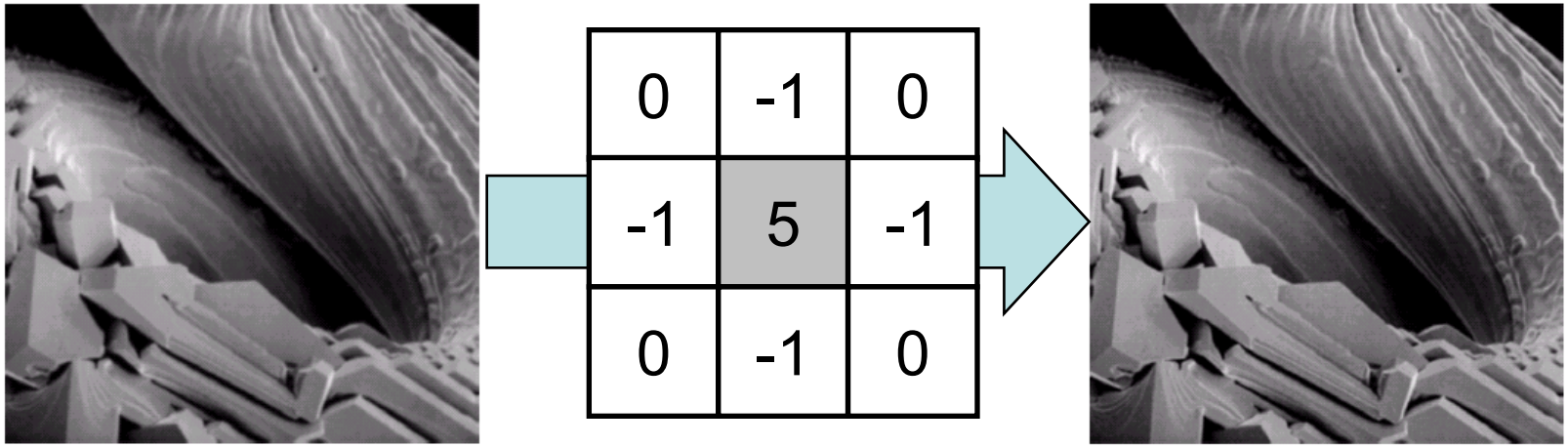
Simplified Image Enhancement

The entire enhancement can be combined into a single filtering operation

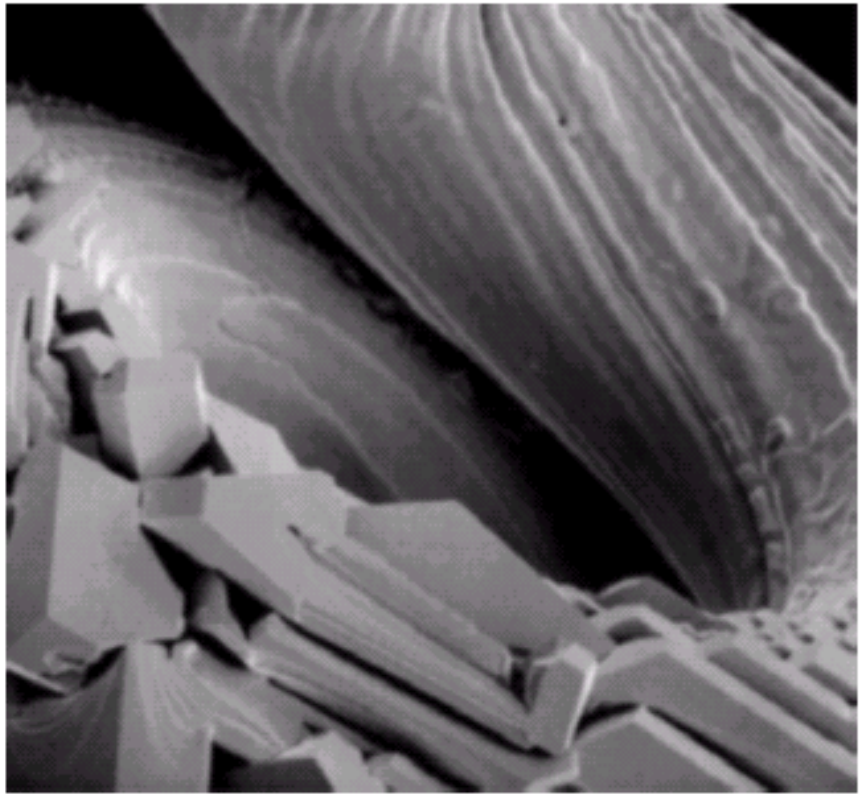
$$
\begin{aligned}
g(x,y) &= f(x,y) - \nabla^2 f \\
&= f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\qquad + f(x,y+1) + f(x,y-1) \\
&\qquad - 4f(x,y)] \\
&= 5f(x,y) - f(x+1,y) - f(x-1,y) \\
&\qquad - f(x,y+1) - f(x,y-1)
\end{aligned}
$$

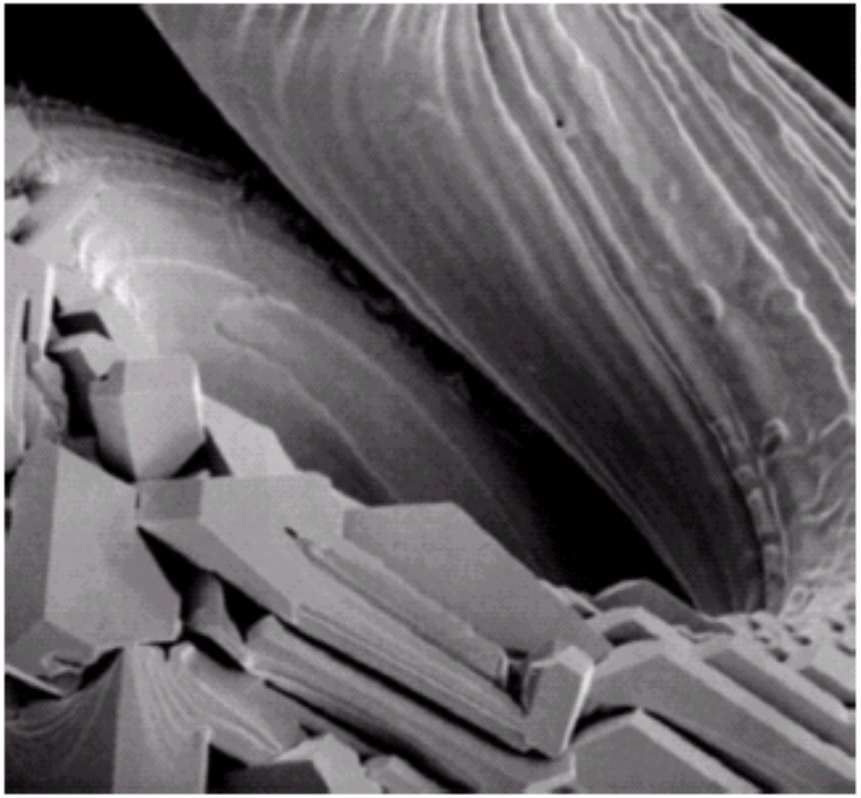
Simplified Image Enhancement (cont…)

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



Simplified Image Enhancement (cont…)

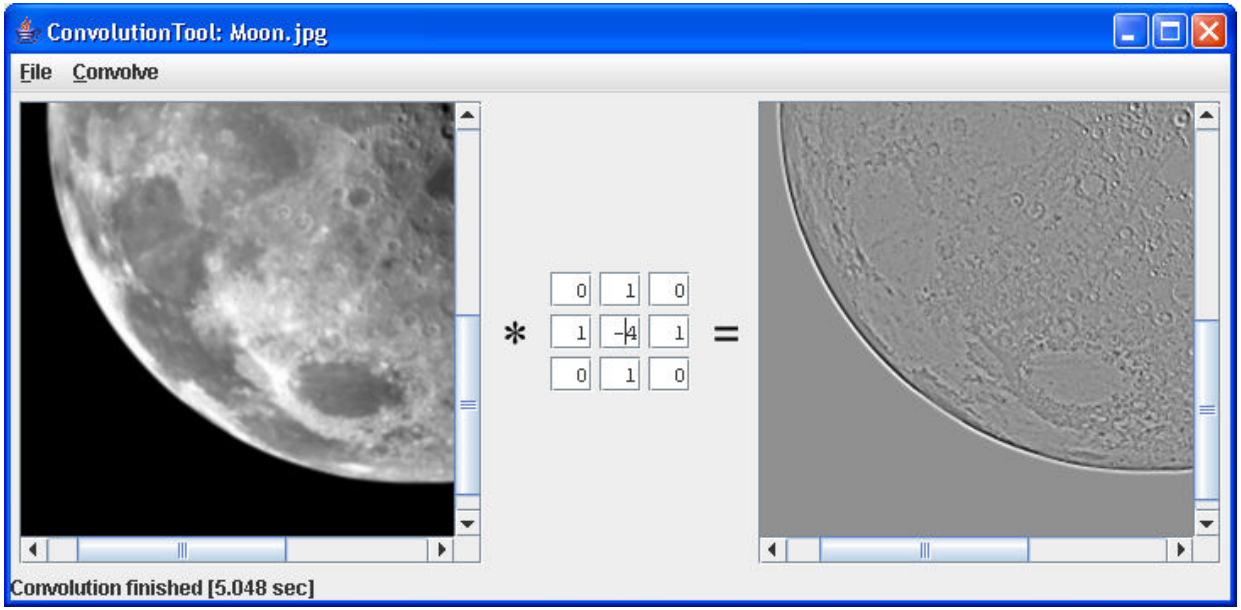




Simple Convolution Tool In Java

A great tool for testing out different filters

- From the book “Image Processing tools in Java”
- Available from webCT later on today
- To launch: `java ConvolutionTool Moon.jpg`



1st Derivative Filtering

Implementing 1st 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:

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

1st Derivative Filtering (cont…)

The magnitude of this vector is given by:

$$\nabla f = mag(\nabla \mathrm{f})$$

$$= \left[G_x^2 + G_y^2\right]^{1/2}$$

$$= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$

For practical reasons this can be simplified as:

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

1st Derivative Filtering (cont…)

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

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

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

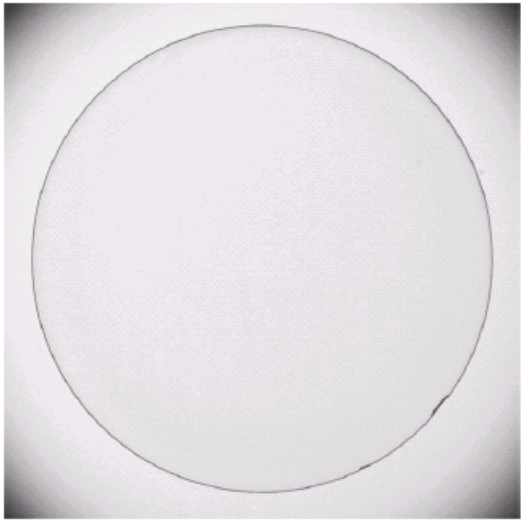
Based on the previous equations we can derive the *Sobel Operators*

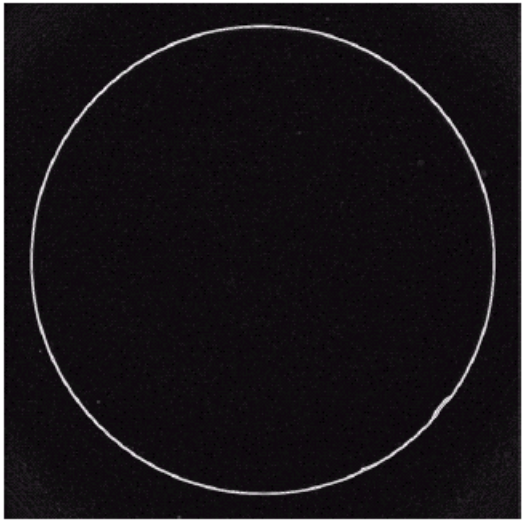
-1	-2	-1
0	0	0
1	2	1

-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

1st & 2nd Derivatives

Comparing the 1st and 2nd derivatives we can conclude the following:

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

Summary

In this lecture we looked at:

- Sharpening filters
  - 1st derivative filters
  - 2nd derivative filters
- Combining filtering techniques

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)

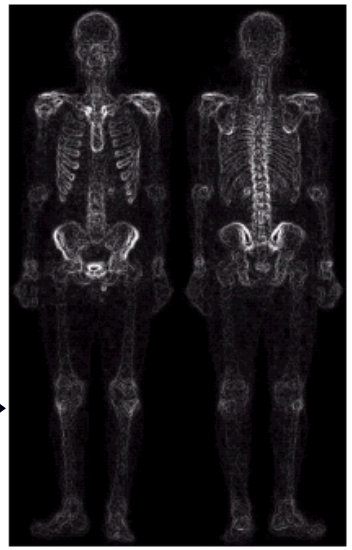
Laplacian filter of bone scan (a)

(b)

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

(c)

Sobel filter of bone scan (a)



(d)

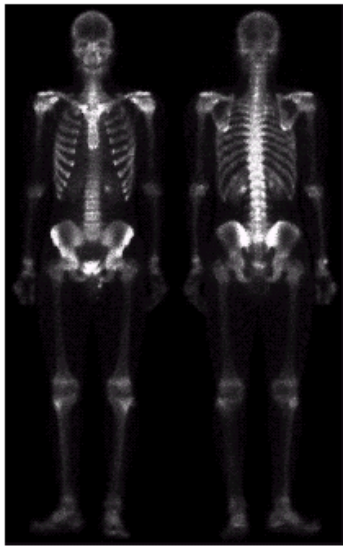
Combining Spatial Enhancement Methods (cont…)

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

(h)

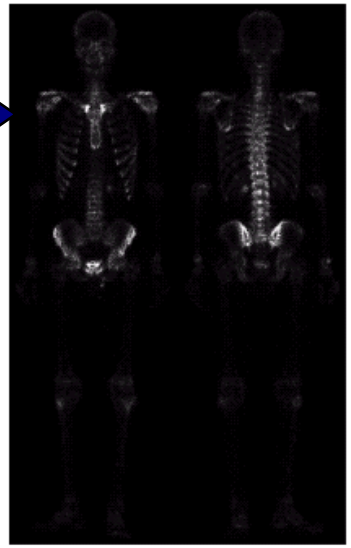
Sharpened image which is sum of (a) and (f)

(g)



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

(f)



(e)

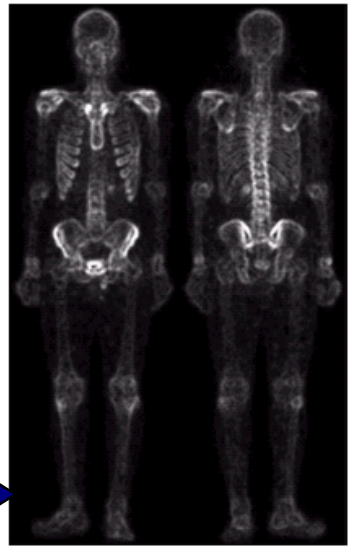


Image (d) smoothed with a 5*5 averaging filter

Combining Spatial Enhancement Methods (cont…)

Compare the original and final images

Digital Image Processing

Lecture - 7

Image Enhancement:
Filtering in the Frequency Domain

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

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

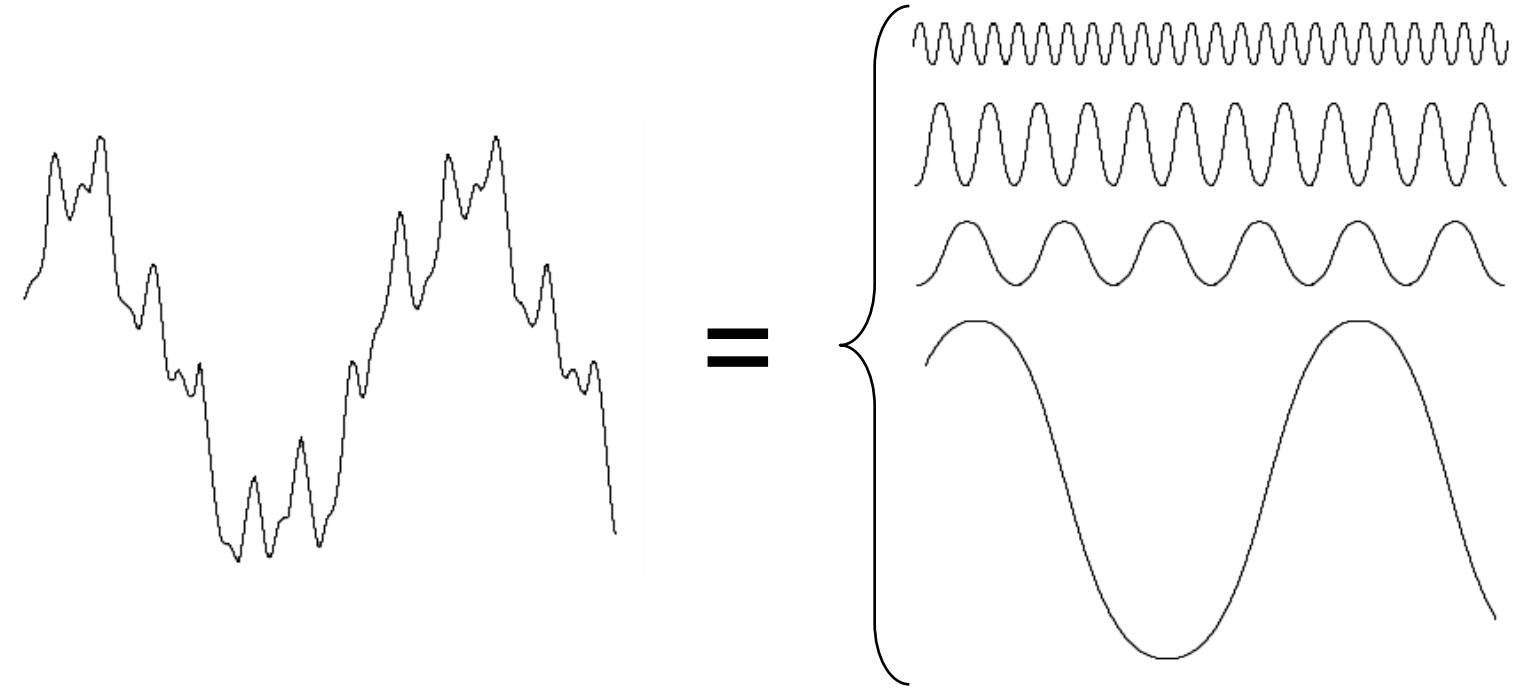
Fourier was born in Auxerre, France in 1768

- 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

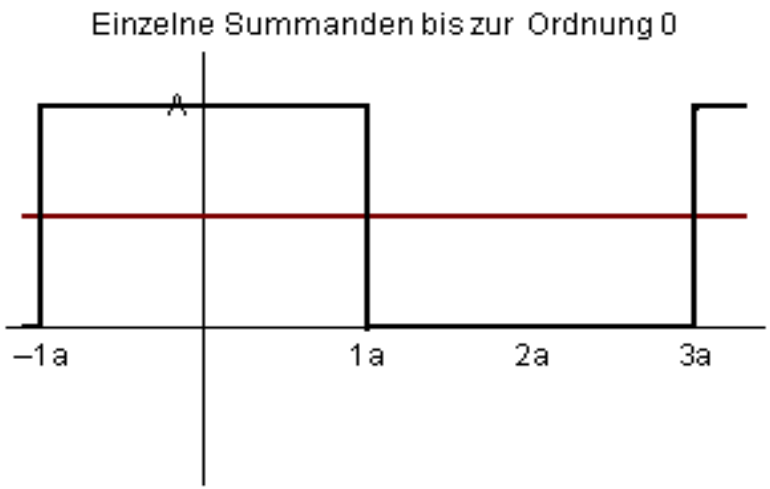
One of the most important mathematical theories in modern engineering

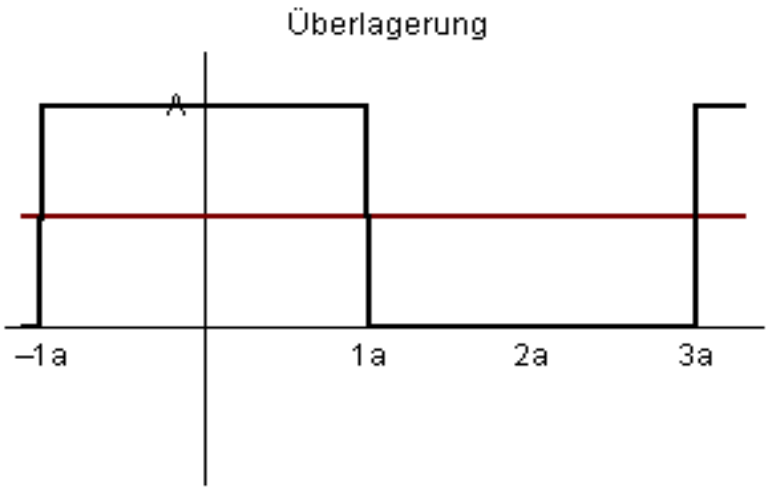
The Big Idea



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*

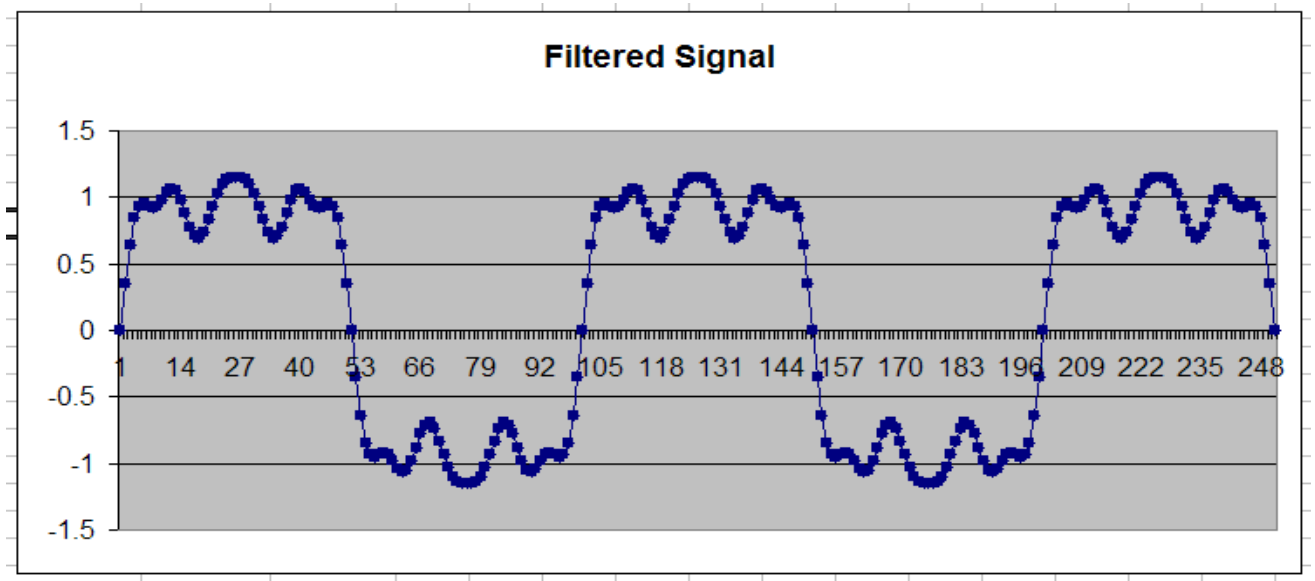
The Big Idea (cont…)

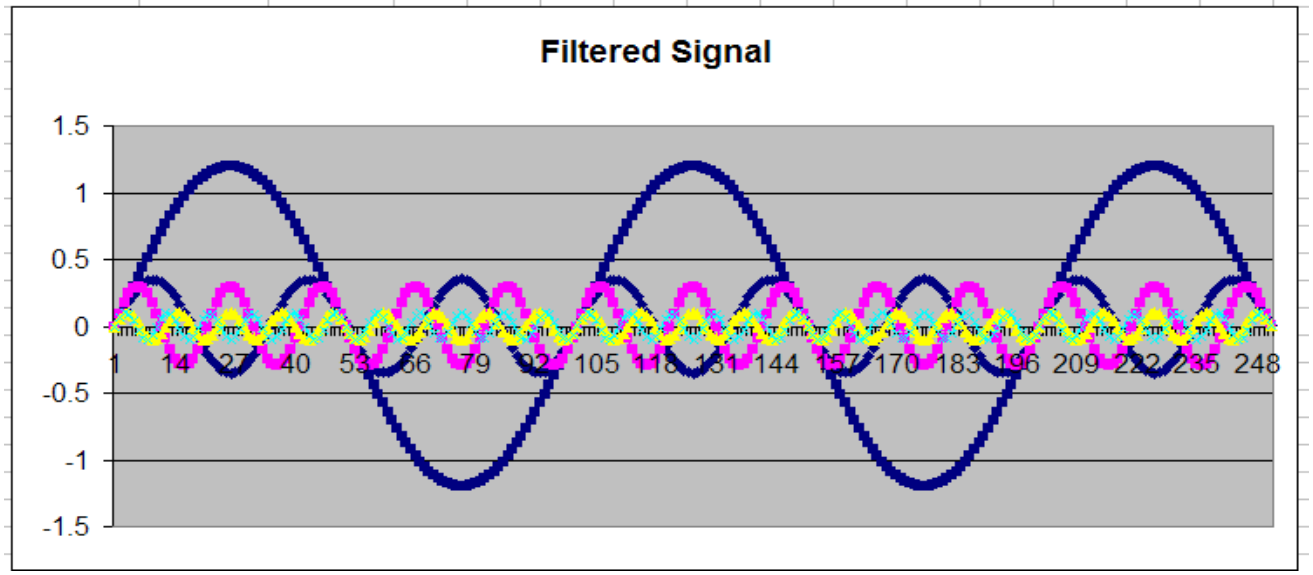




Notice how we get closer and closer to the original function as we add more and more frequencies

The Big Idea (cont…)





Frequency domain signal processing example in Excel

The Discrete Fourier Transform (DFT)

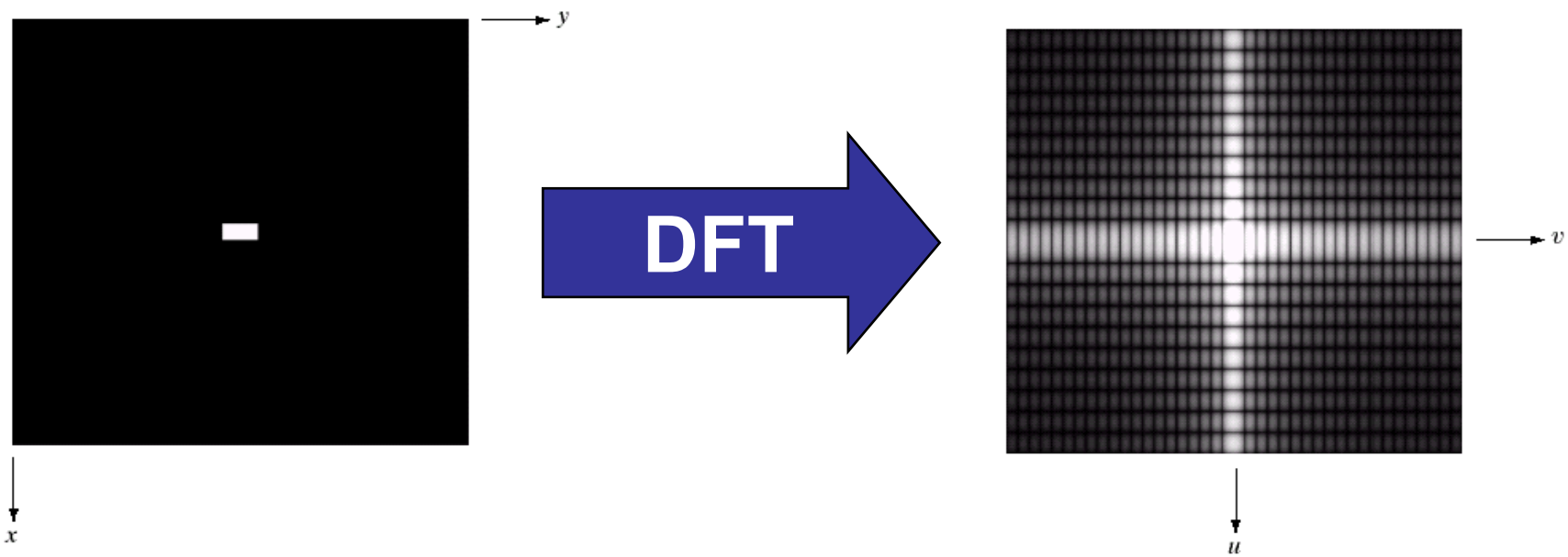
The *Discrete Fourier Transform* of $f(x, y)$, for x = 0, 1, 2…M-1 and y = 0,1,2…N-1, denoted by $F(u, v)$, is given by the equation:

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

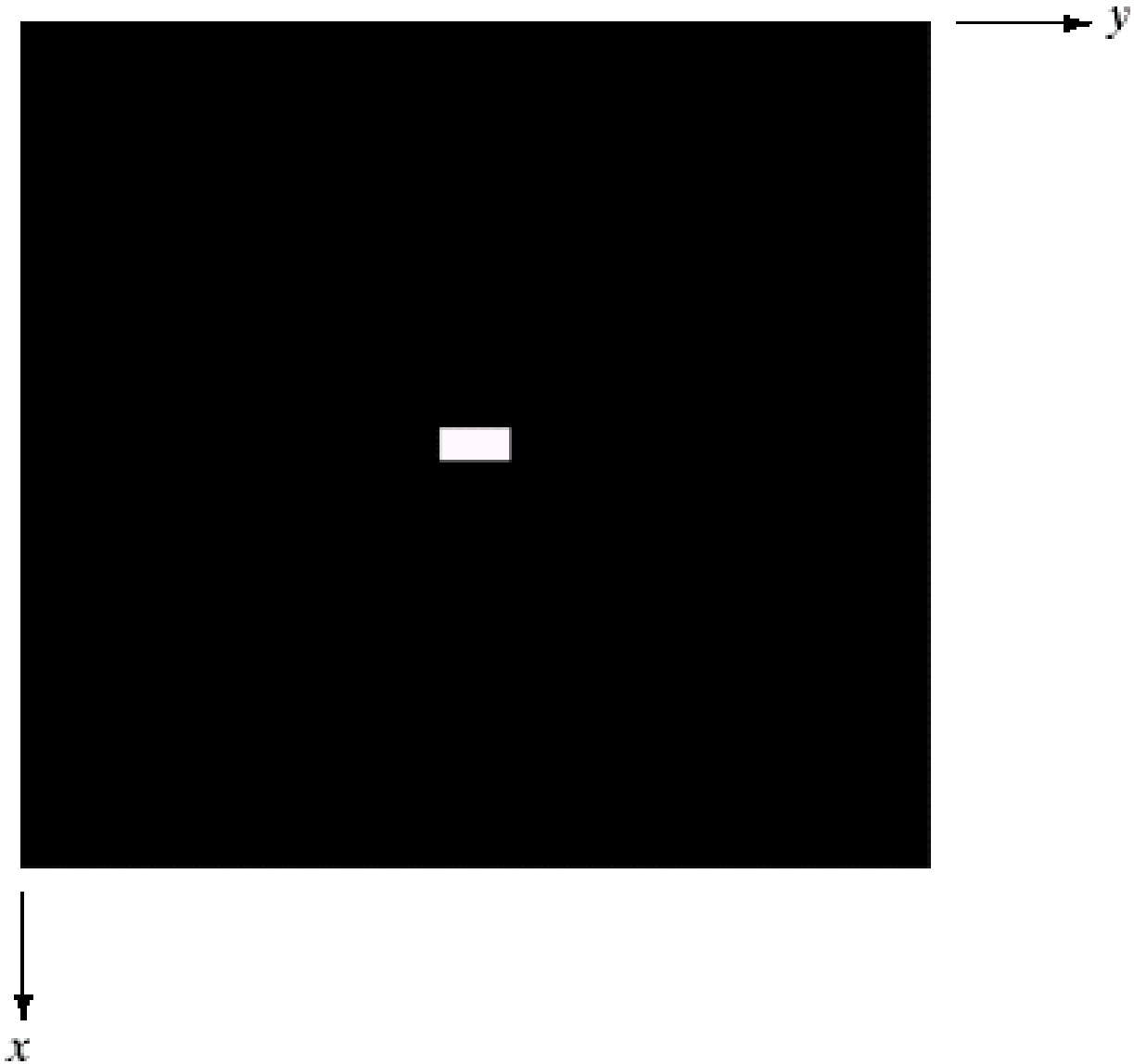
for u = 0, 1, 2…M-1 and v = 0, 1, 2…N-1.

DFT & Images

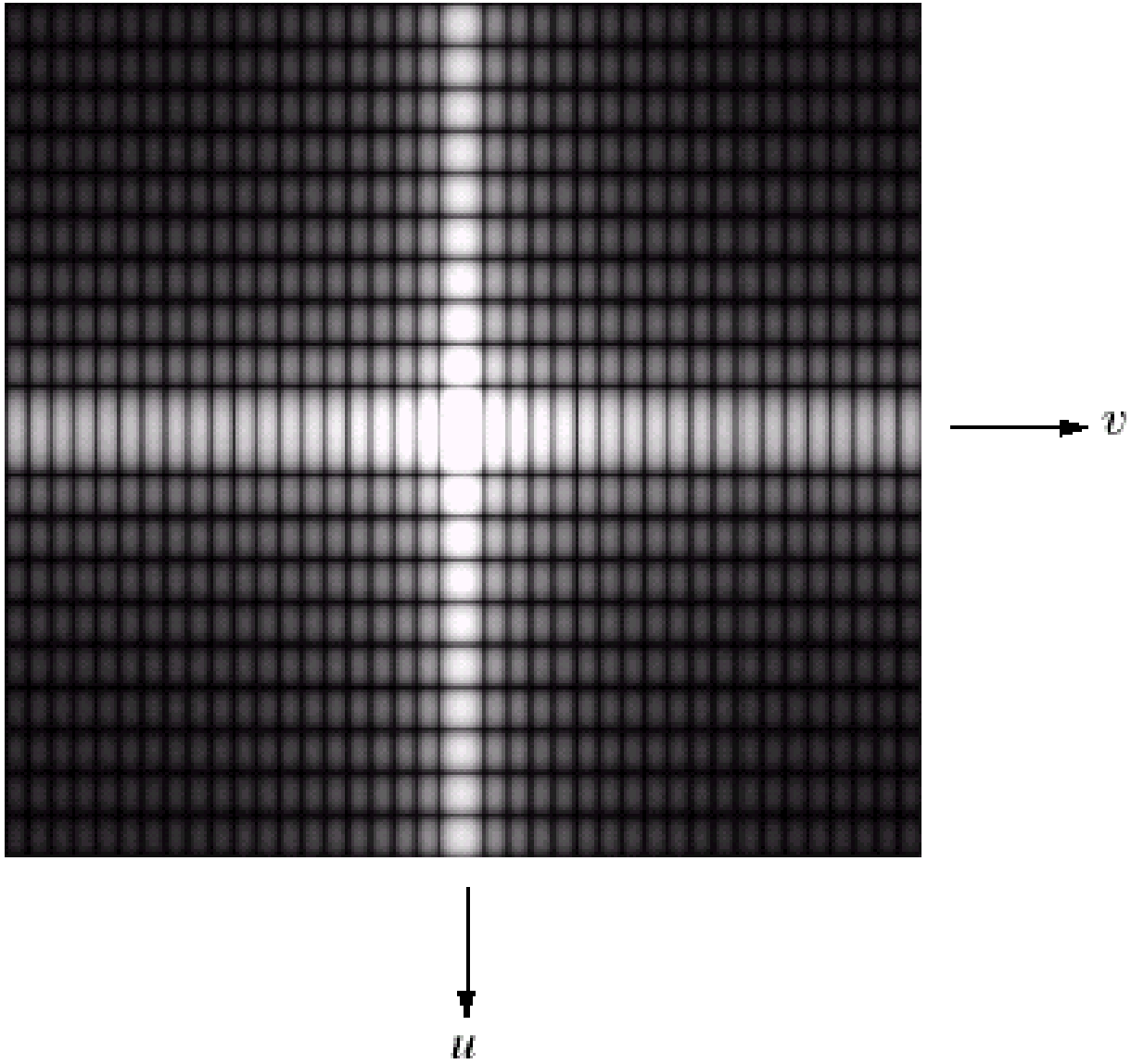
The DFT of a two dimensional image can be visualised by showing the spectrum of the images component frequencies



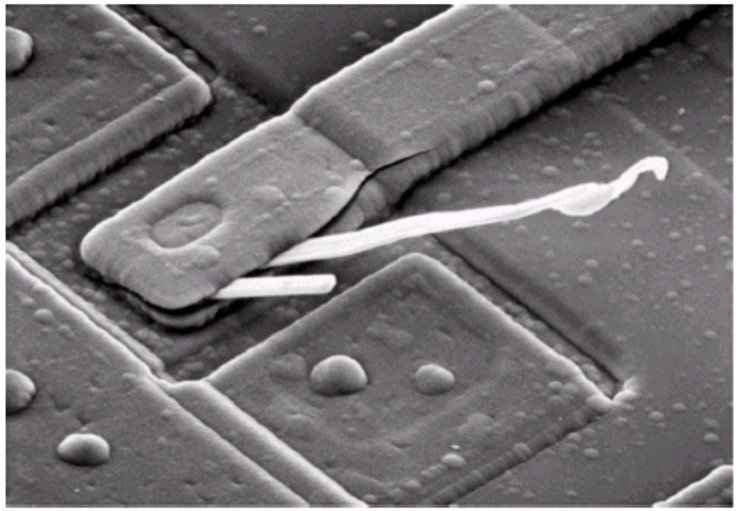
DFT & Images



DFT & Images

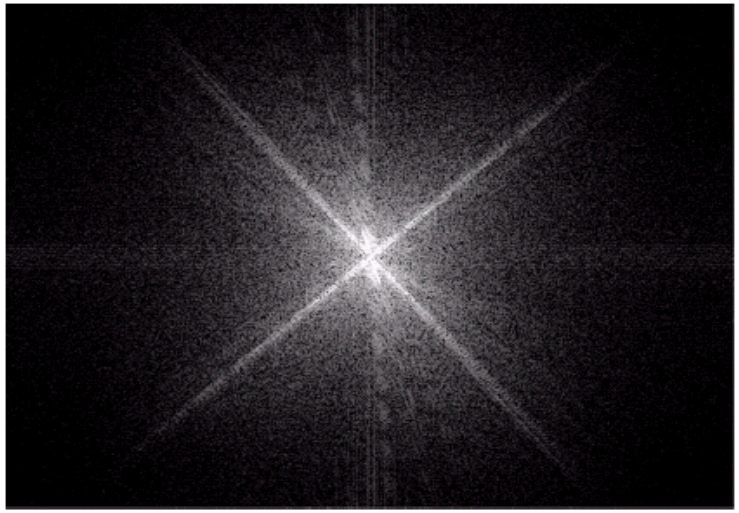


DFT & Images (cont…)



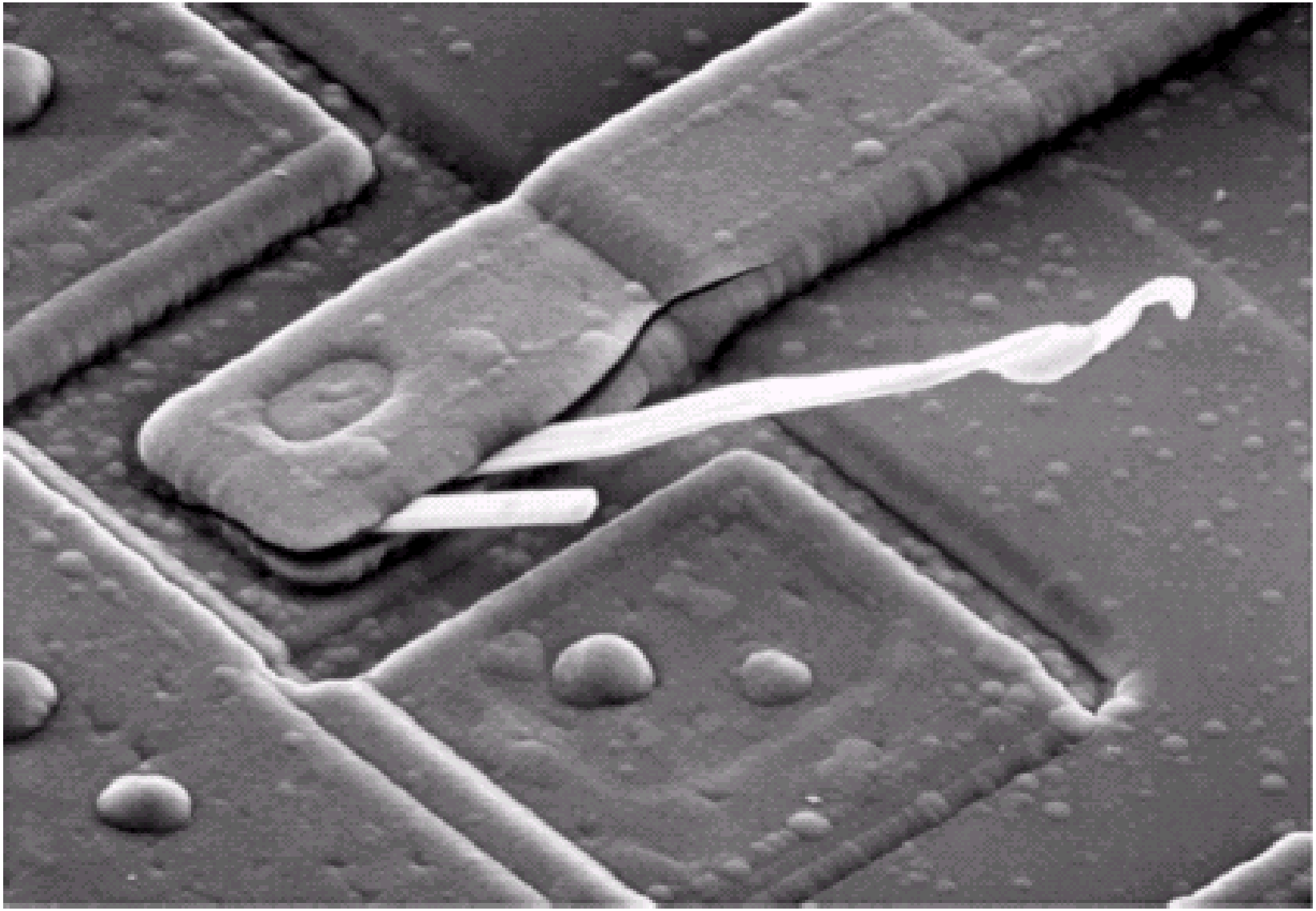
Scanning electron microscope image of an integrated circuit magnified ~2500 times

DFT

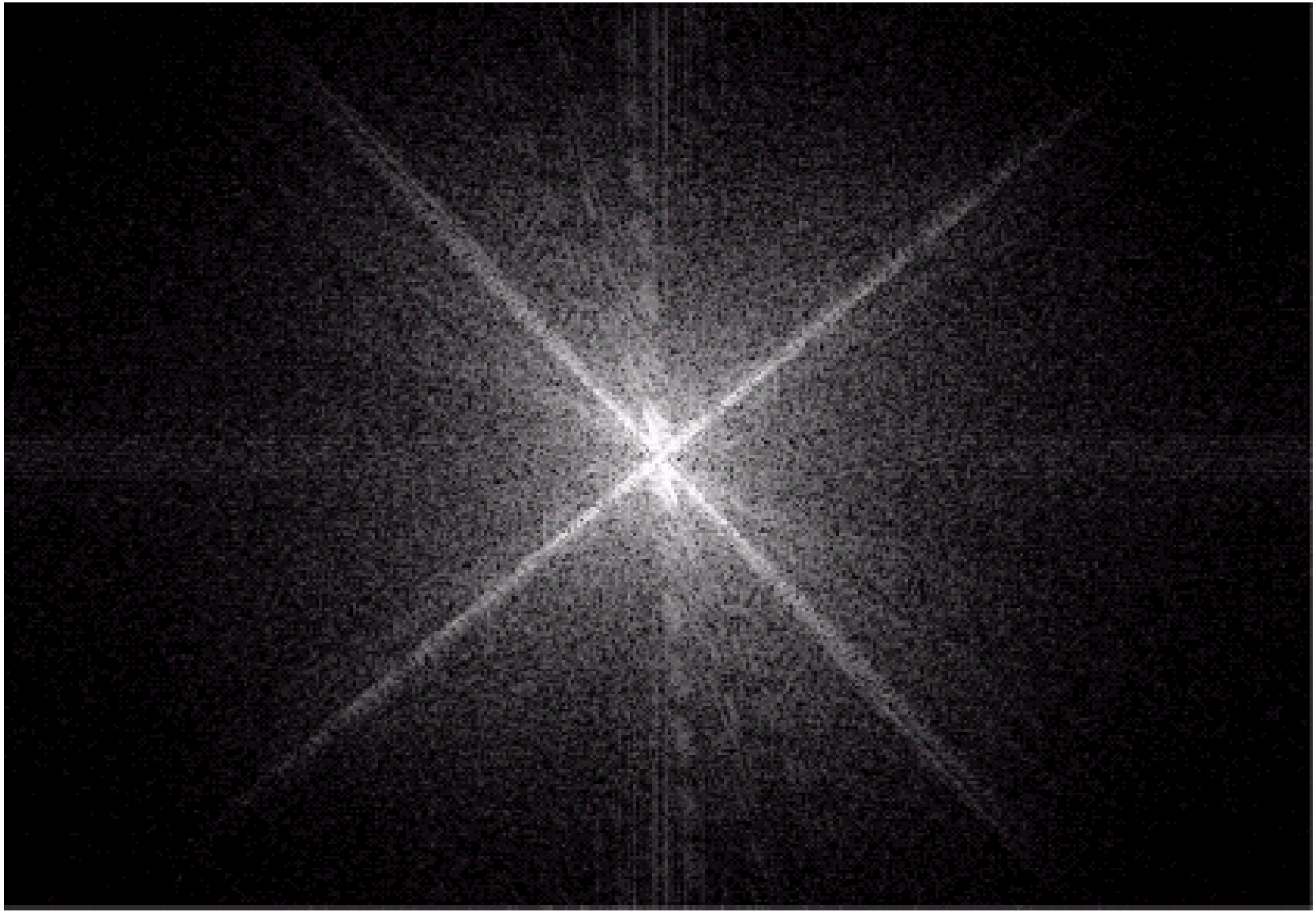


Fourier spectrum of the image

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:

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

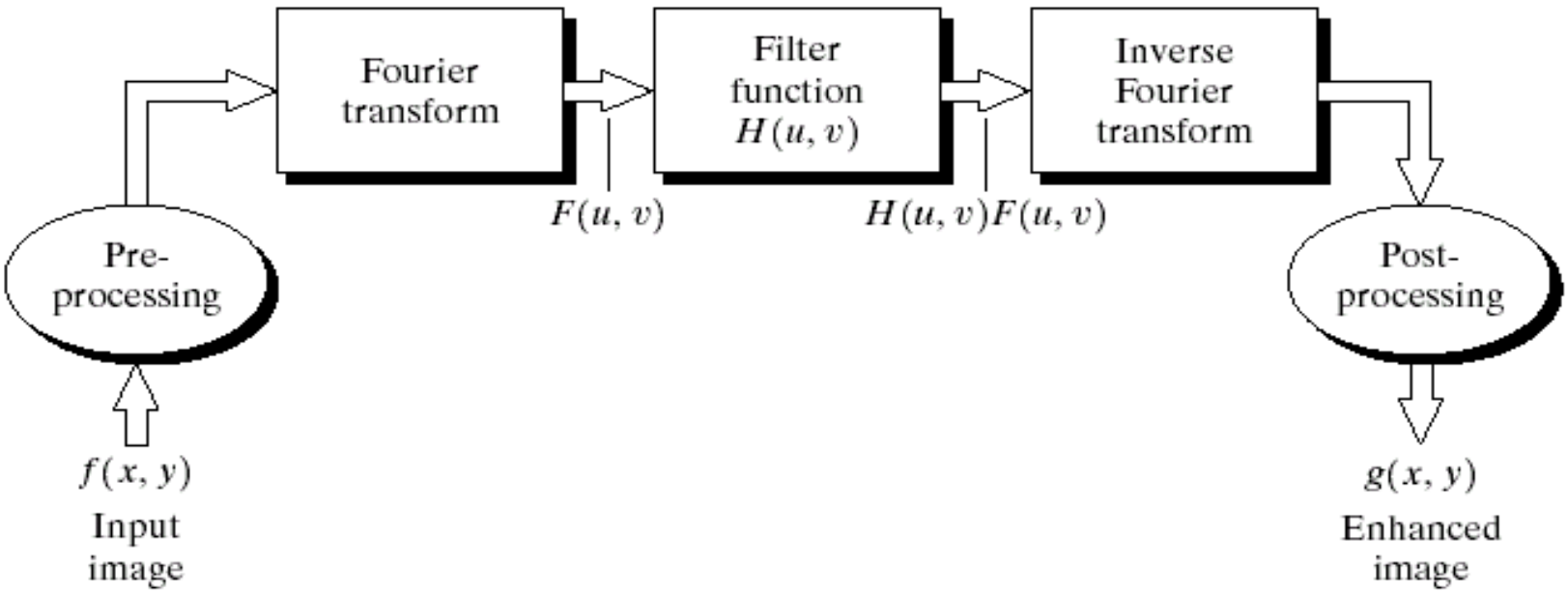
for x = 0, 1, 2…M-1 and y = 0, 1, 2…N-1

The DFT and Image Processing

To filter an image in the frequency domain:

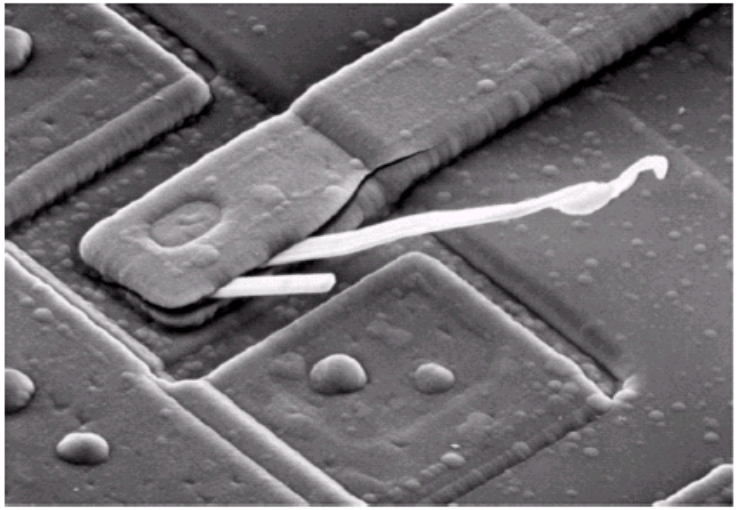
1. Compute $F(u,v)$ the DFT of the image
2. Multiply $F(u,v)$ by a filter function $H(u,v)$
3. Compute the inverse DFT of the result

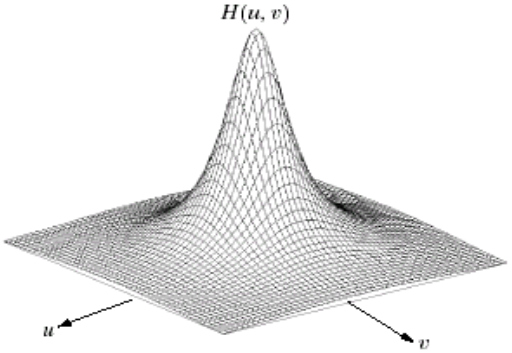
Frequency domain filtering operation

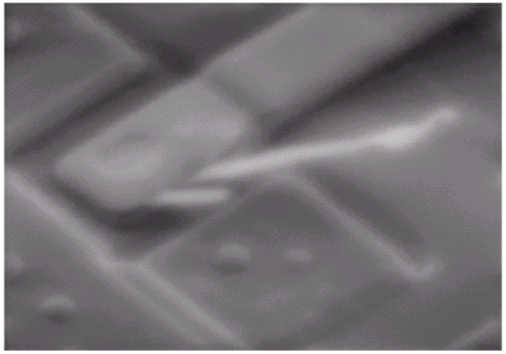


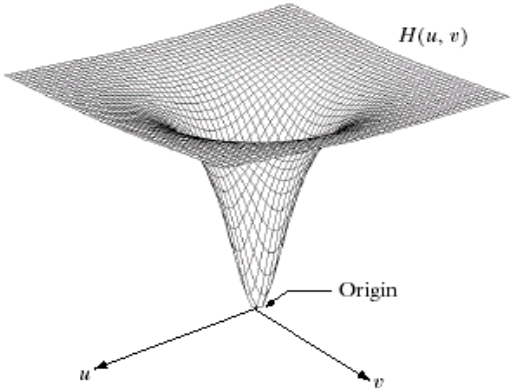
Some Basic Frequency Domain Filters

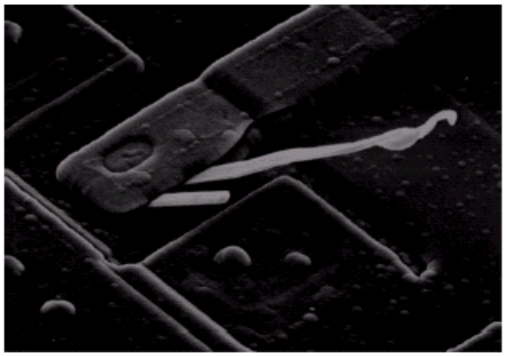
Low Pass Filter





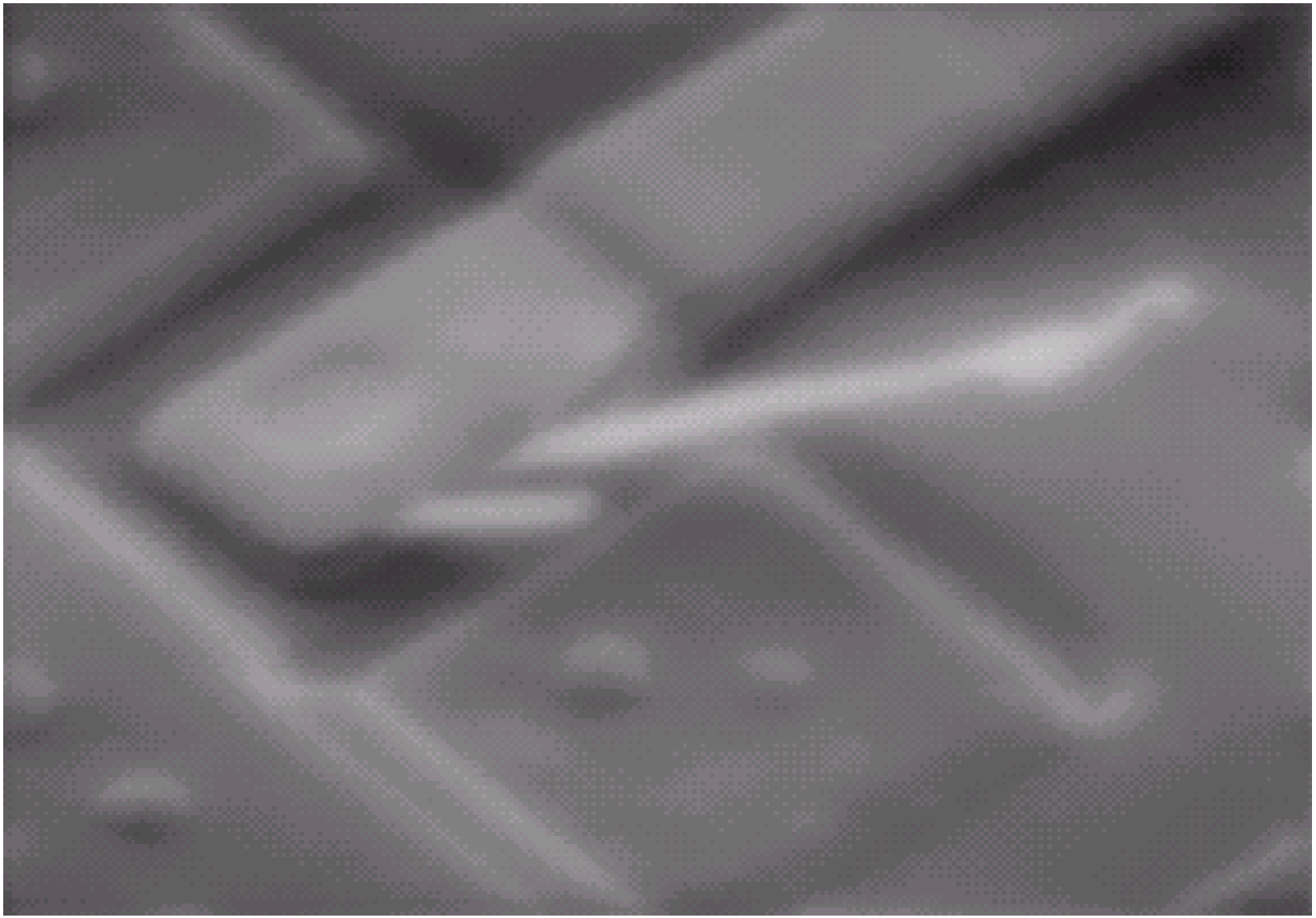




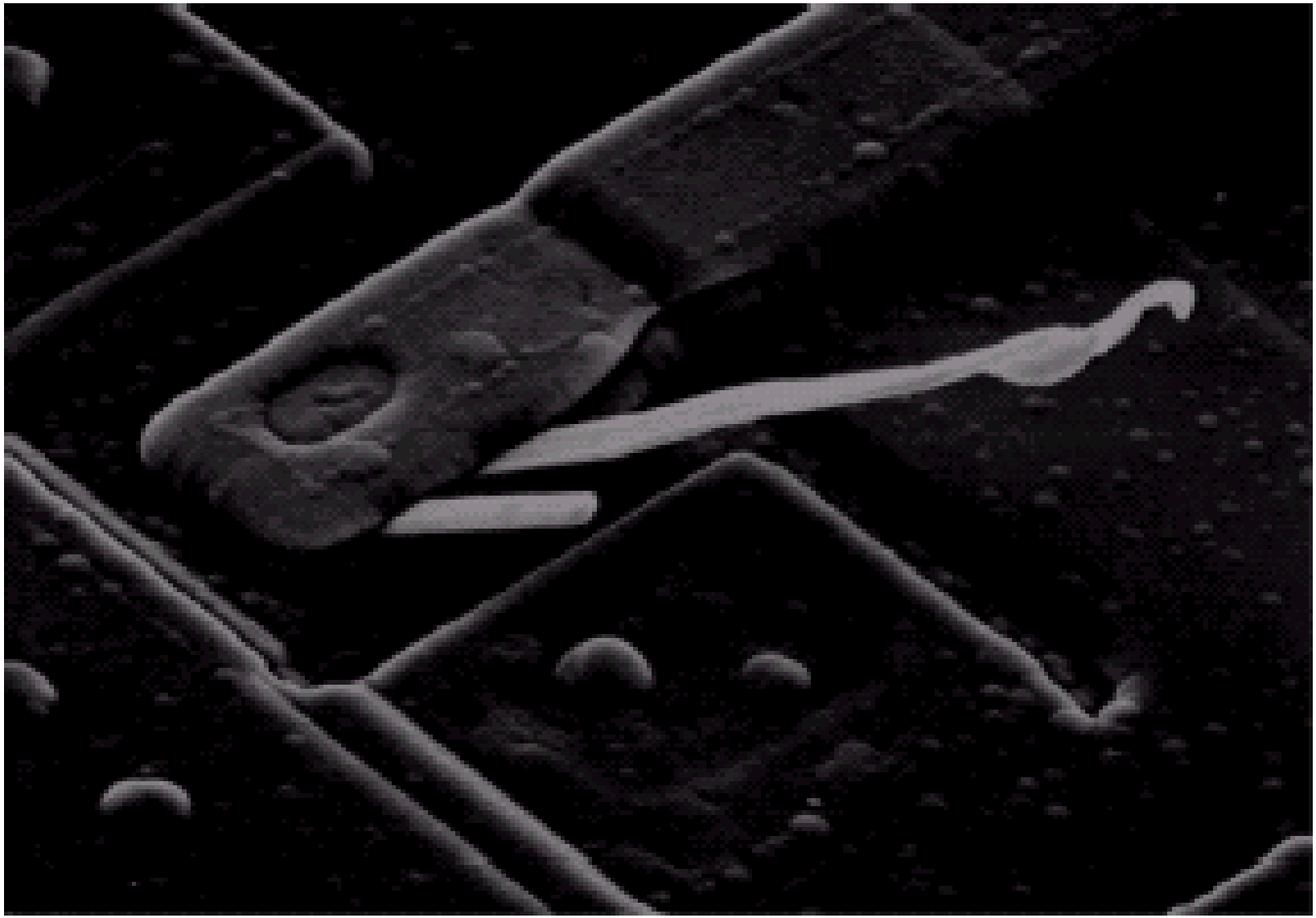


High Pass Filter

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:

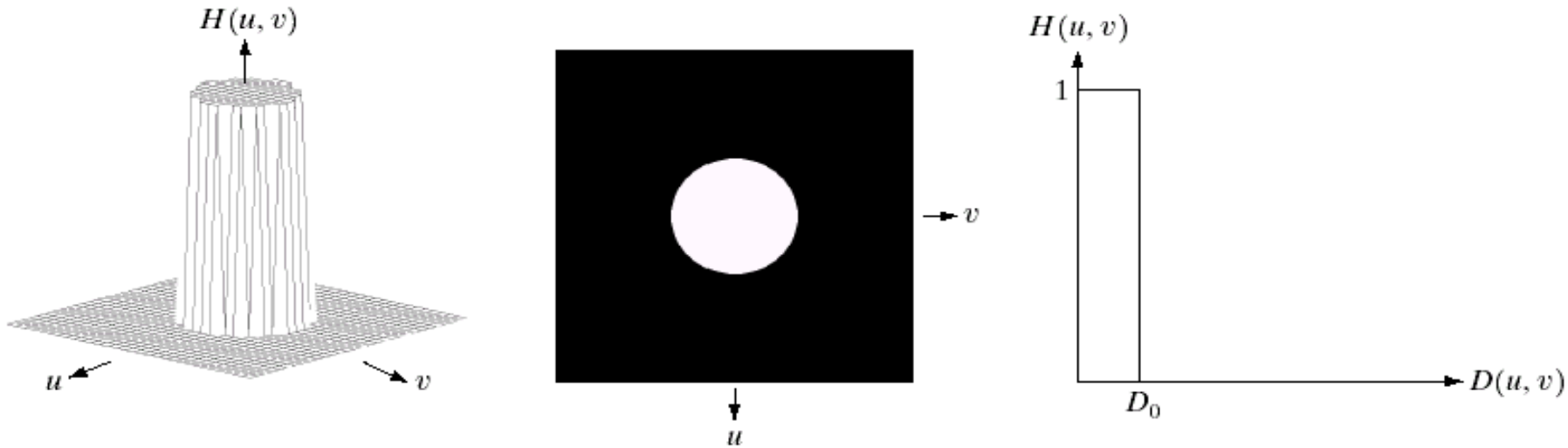
$$G(u,v) = H(u,v)F(u,v)$$

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…)

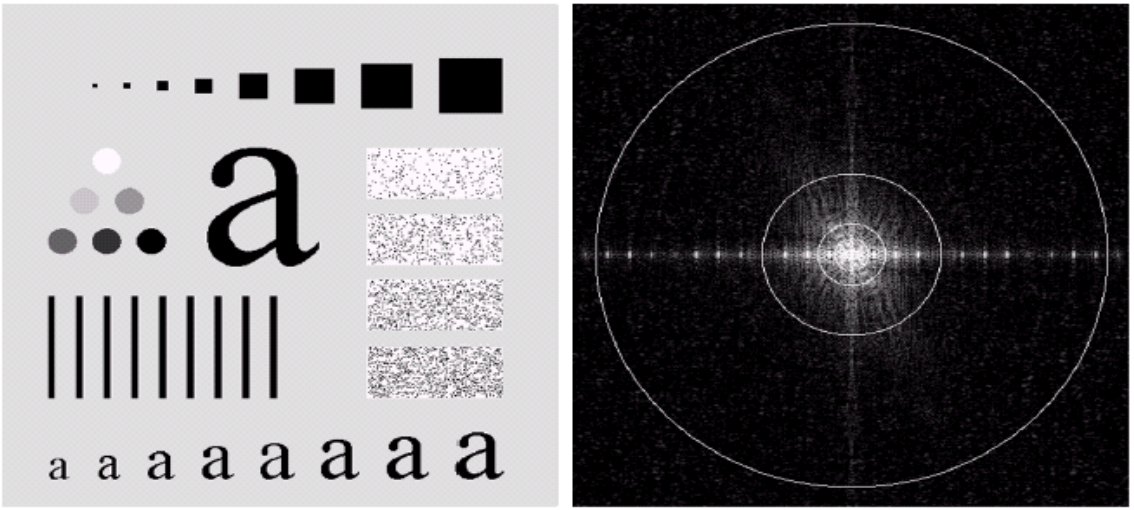
The transfer function for the ideal low pass filter can be given as:

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

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

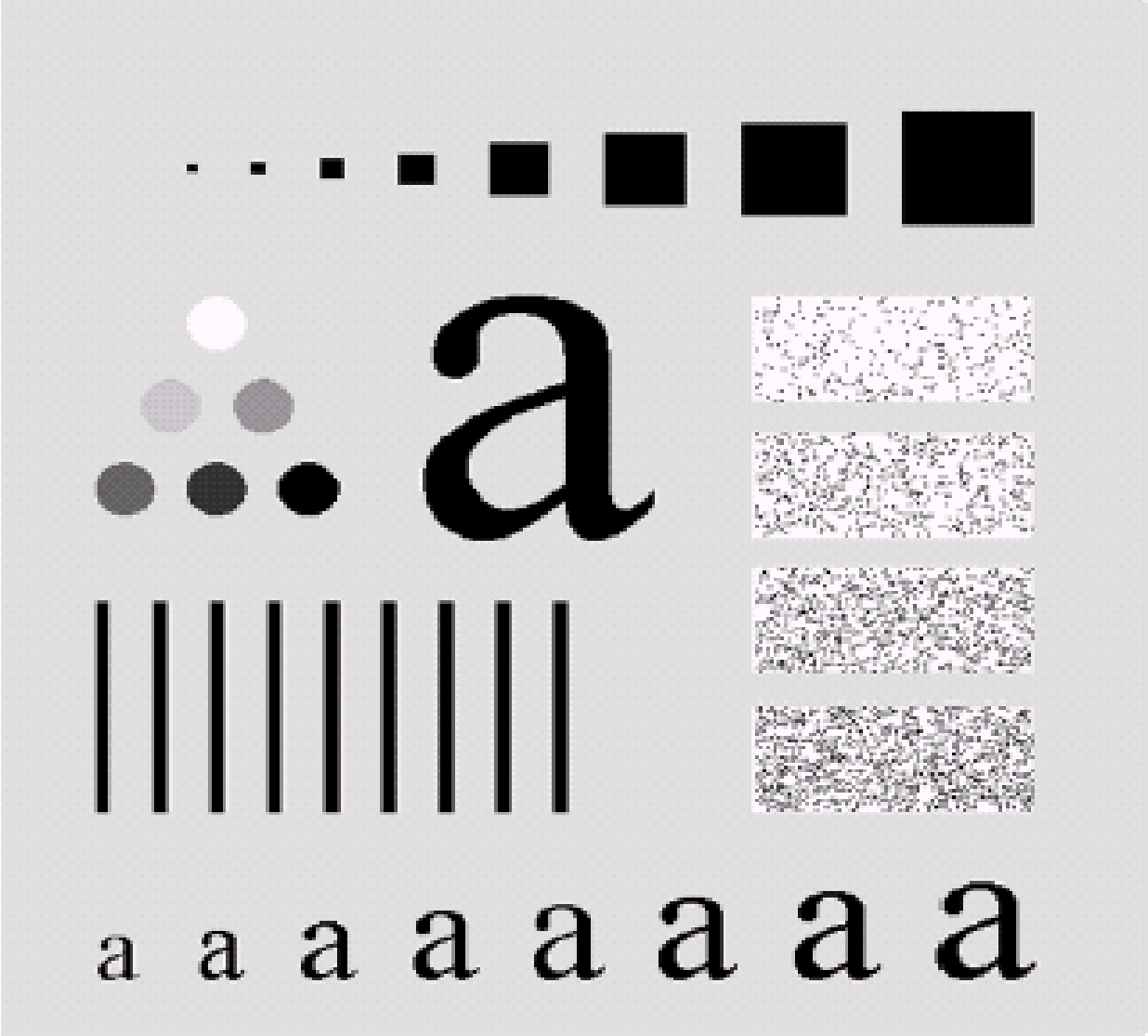
$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$$

Ideal Low Pass Filter (cont…)

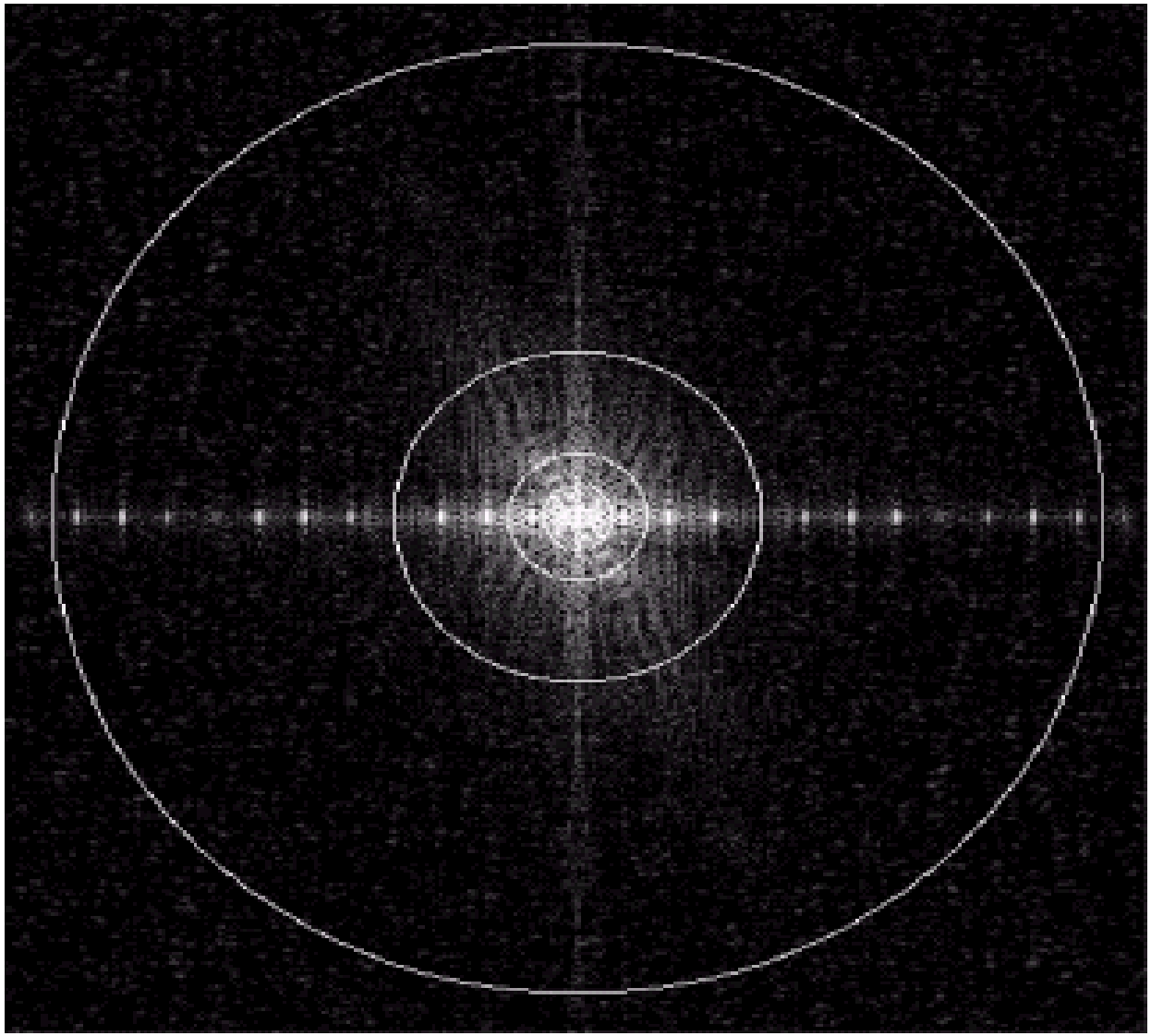


Above we show an image, it's Fourier spectrum and a series of ideal low pass filters of radius 5, 15, 30, 80 and 230 superimposed on top of it

Ideal Low Pass Filter (cont…)

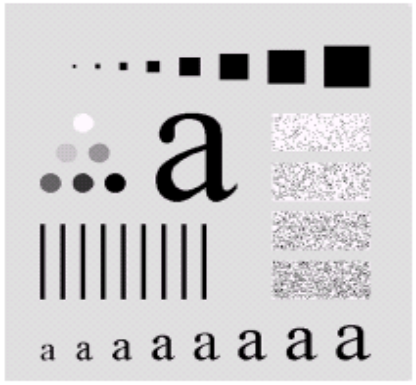


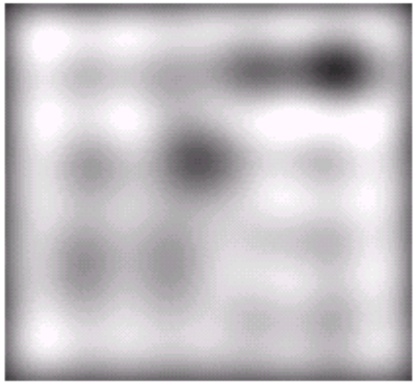
Ideal Low Pass Filter (cont…)



Ideal Low Pass Filter (cont…)

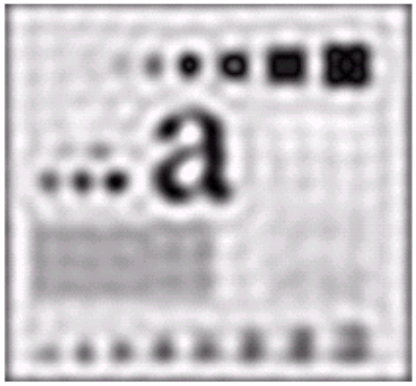
Original image

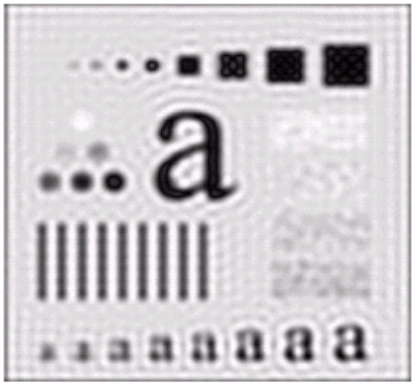




Result of filtering with ideal low pass filter of radius 5

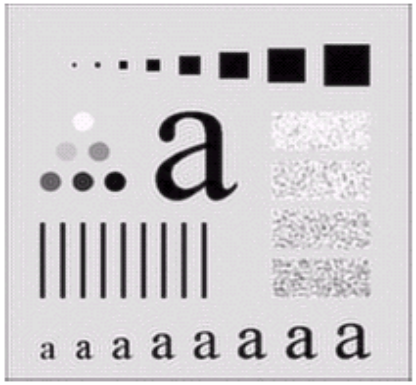
Result of filtering with ideal low pass filter of radius 15

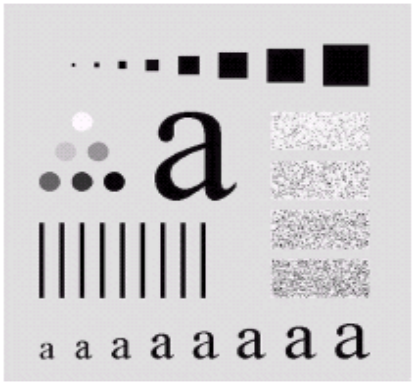




Result of filtering with ideal low pass filter of radius 30

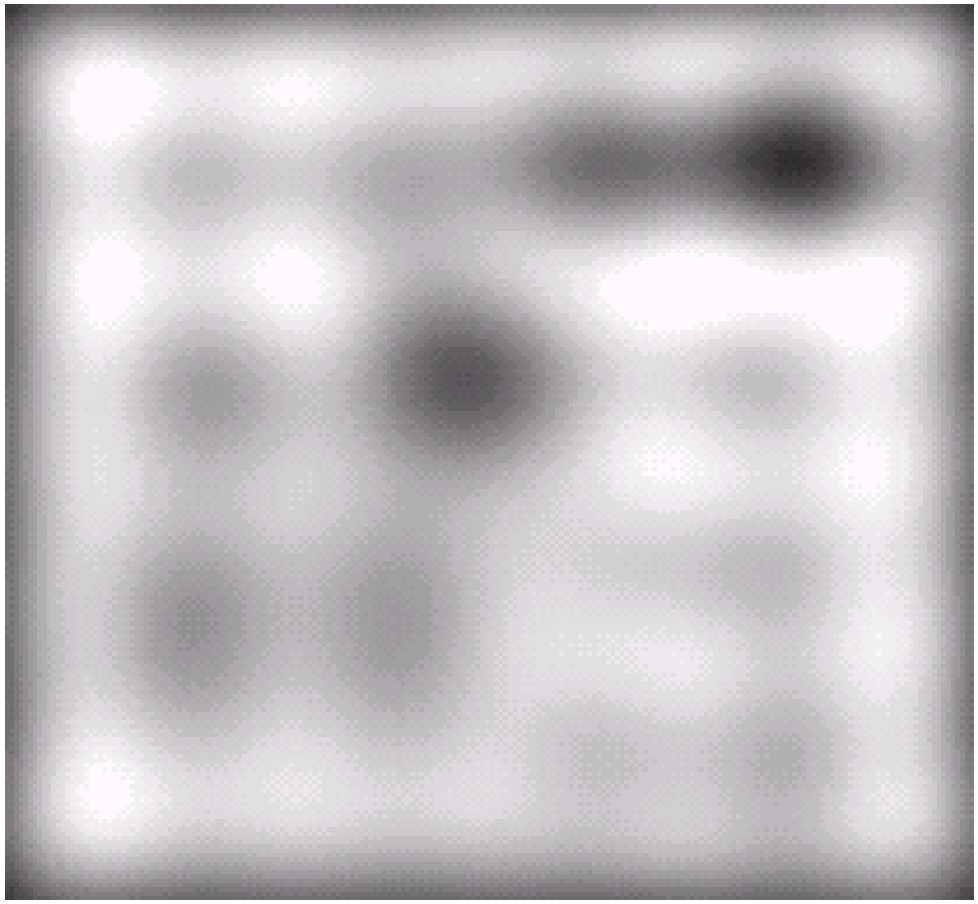
Result of filtering with ideal low pass filter of radius 80





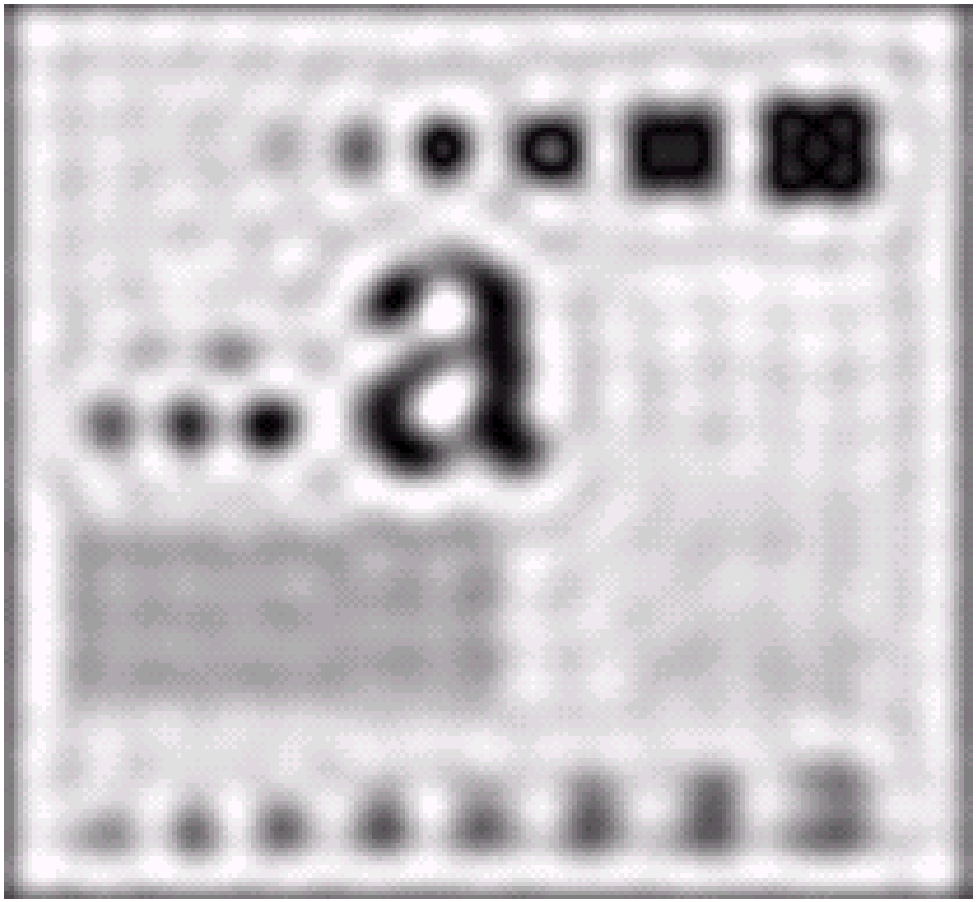
Result of filtering with ideal low pass filter of radius 230

Ideal Low Pass Filter (cont…)



Result of filtering with ideal low pass filter of radius 5

Ideal Low Pass Filter (cont…)

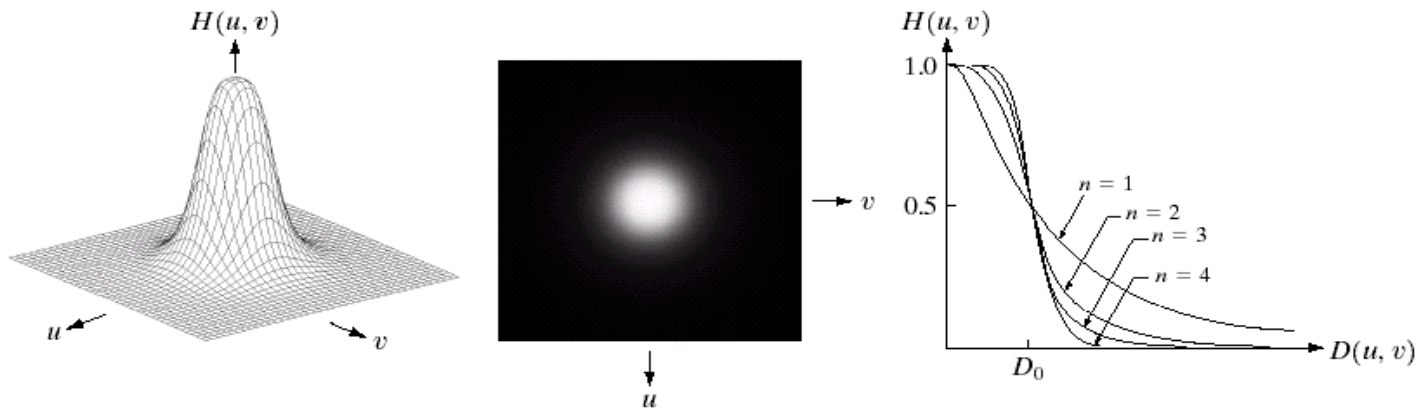


Result of filtering with ideal low pass filter of radius 15

Butterworth Lowpass Filters

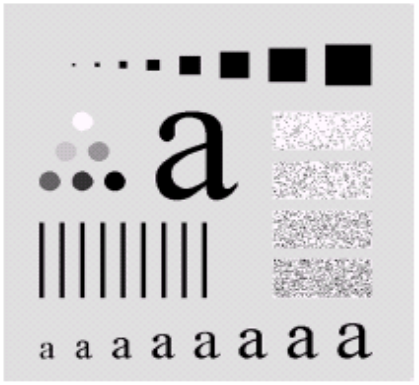
The transfer function of a Butterworth lowpass filter of order n with cutoff frequency at distance D_0 from the origin is defined as:

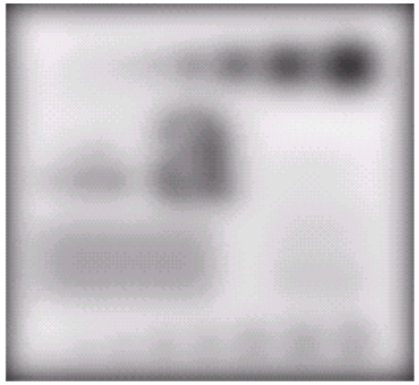
$$H(u,v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$



Butterworth Lowpass Filter (cont…)

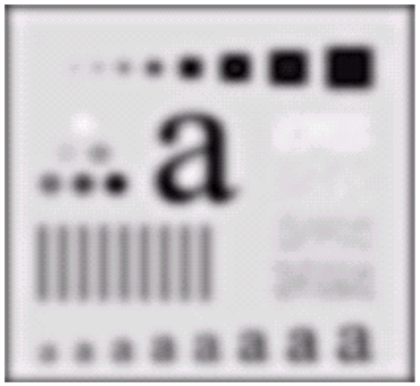
Original image

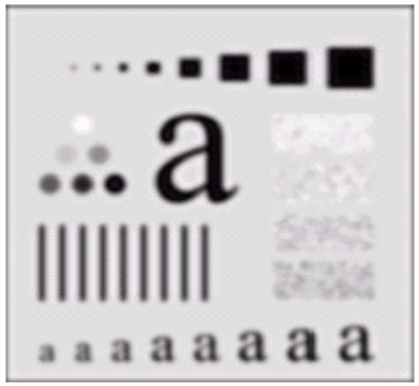




Result of filtering with Butterworth filter of order 2 and cutoff radius 5

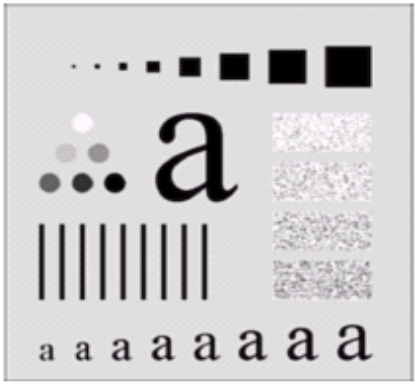
Result of filtering with Butterworth filter of order 2 and cutoff radius 15

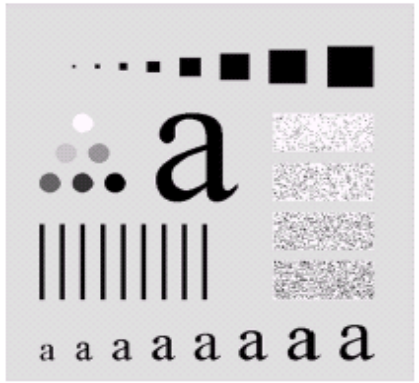




Result of filtering with Butterworth filter of order 2 and cutoff radius 30

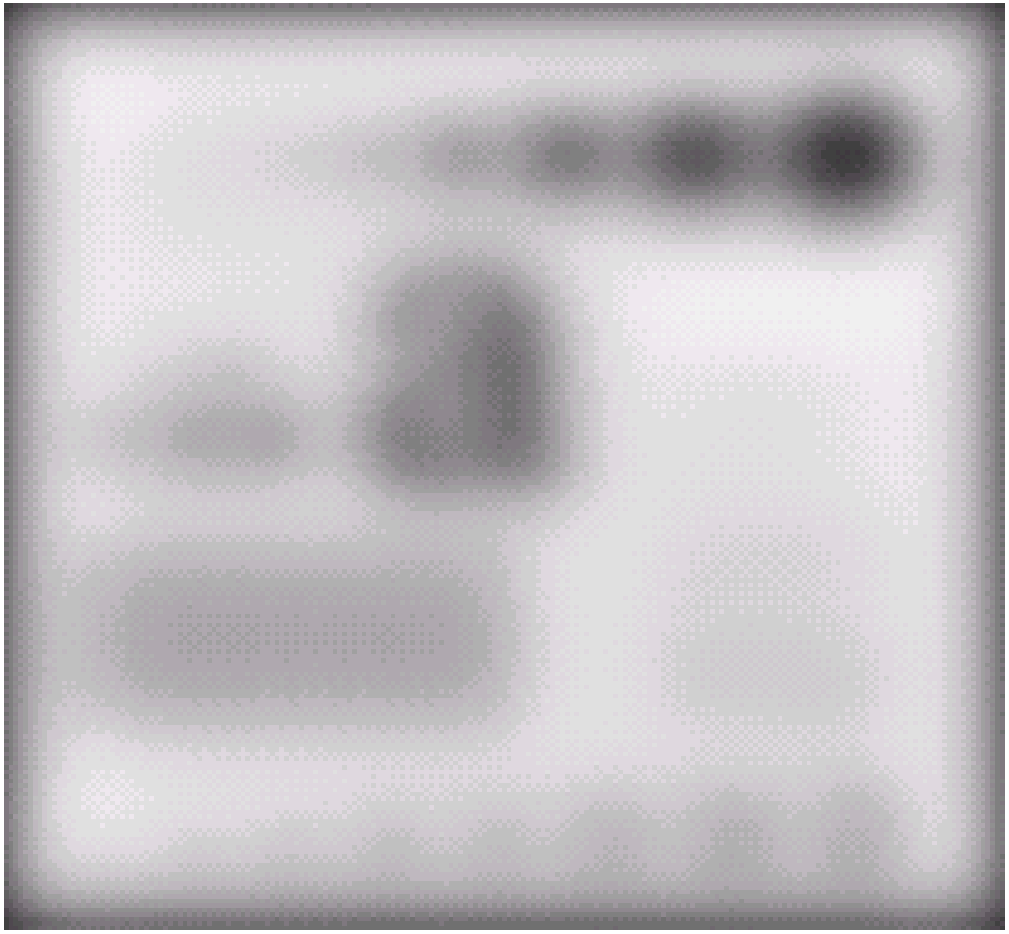
Result of filtering with Butterworth filter of order 2 and cutoff radius 80





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

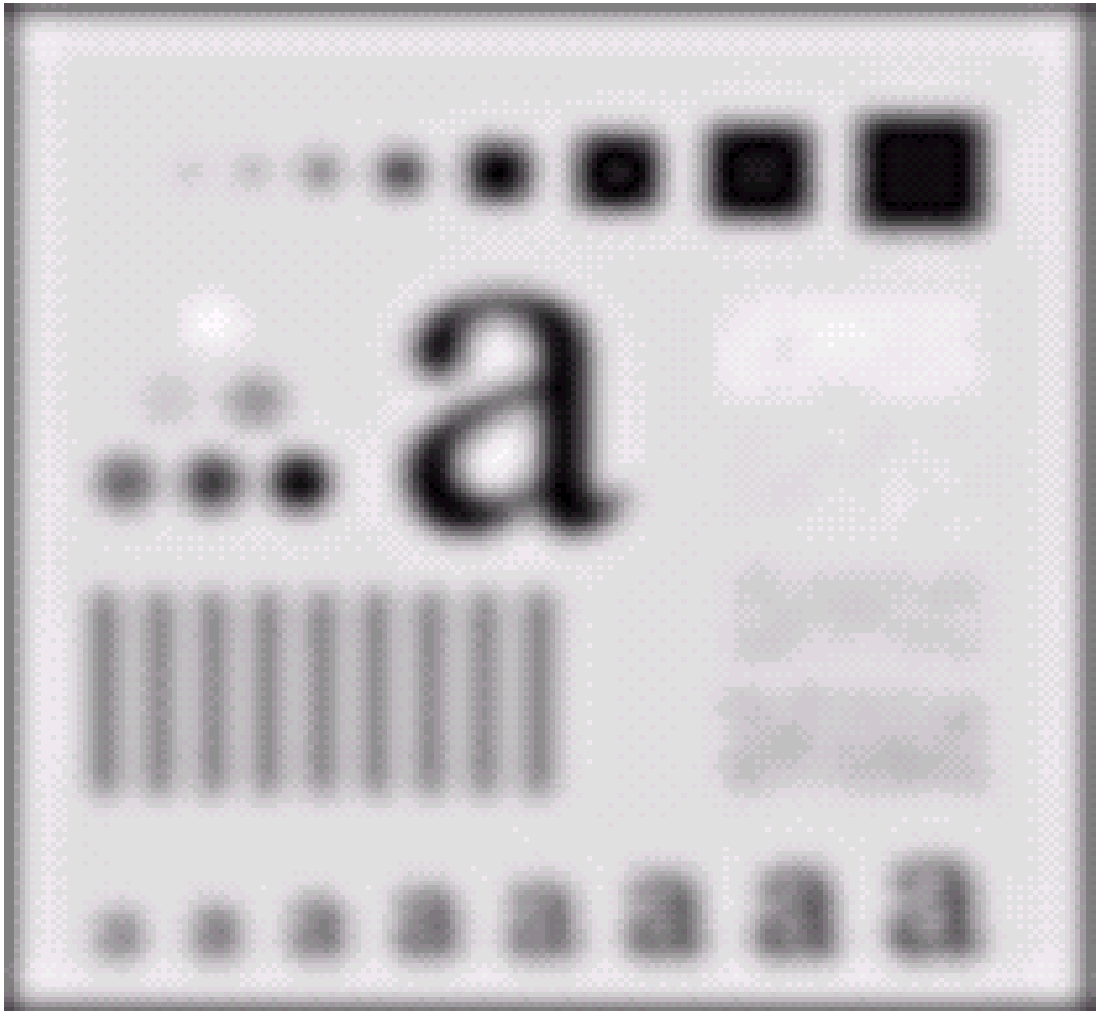
Butterworth Lowpass Filter (cont…)



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

Butterworth Lowpass Filter (cont…)

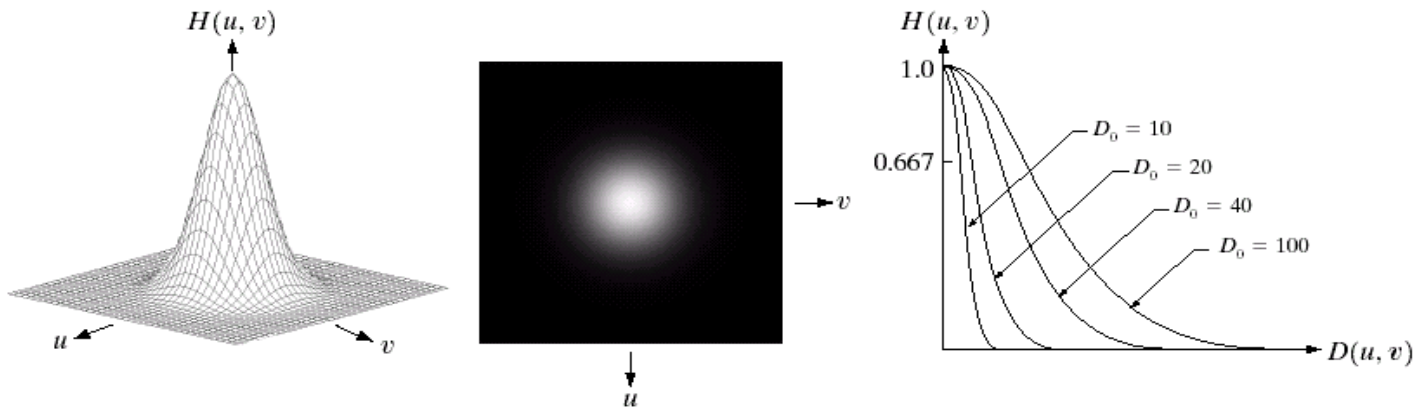
Result of filtering with Butterworth filter of order 2 and cutoff radius 15



Gaussian Lowpass Filters

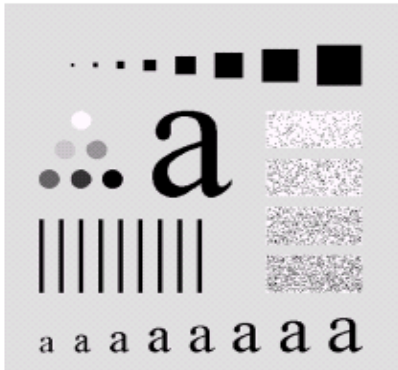
The transfer function of a Gaussian lowpass filter is defined as:

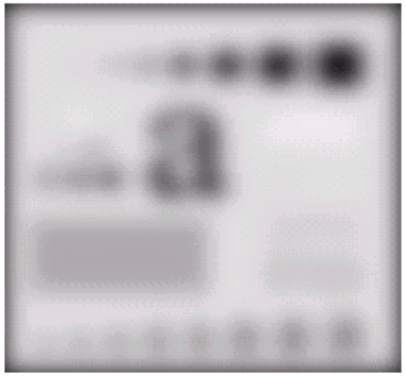
$$H(u,v) = e^{-D^2(u,v)/2D_0{}^2}$$



Gaussian Lowpass Filters (cont…)

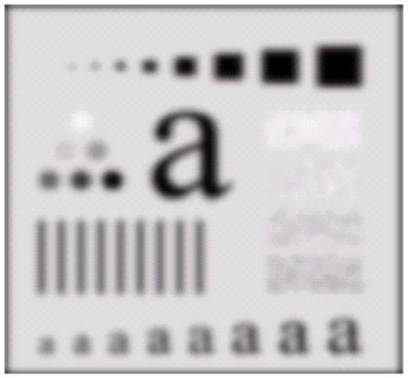
Original image

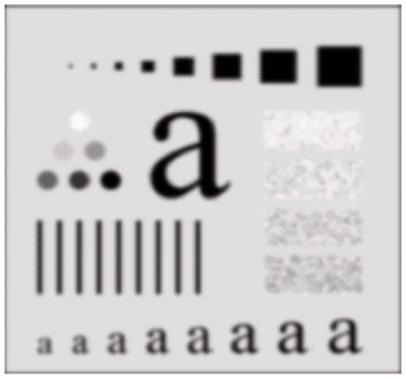




Result of filtering with Gaussian filter with cutoff radius 5

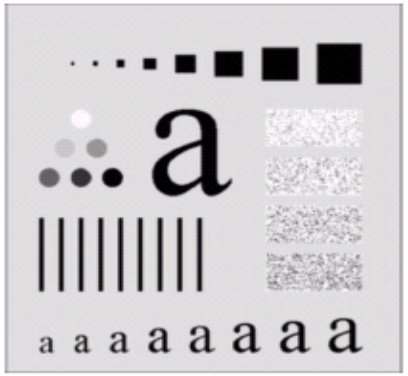
Result of filtering with Gaussian filter with cutoff radius 15

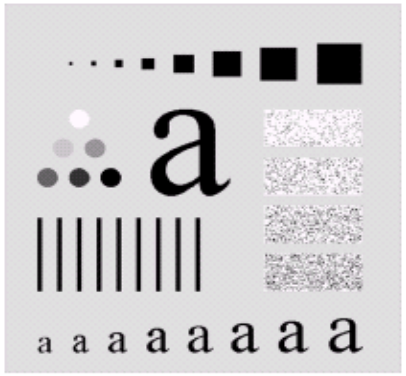




Result of filtering with Gaussian filter with cutoff radius 30

Result of filtering with Gaussian filter with cutoff radius 85

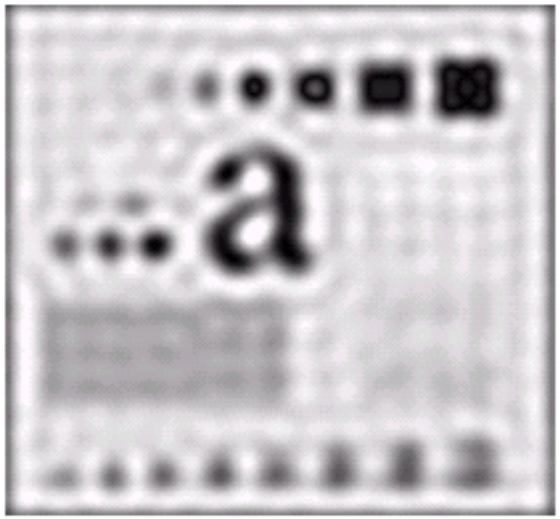


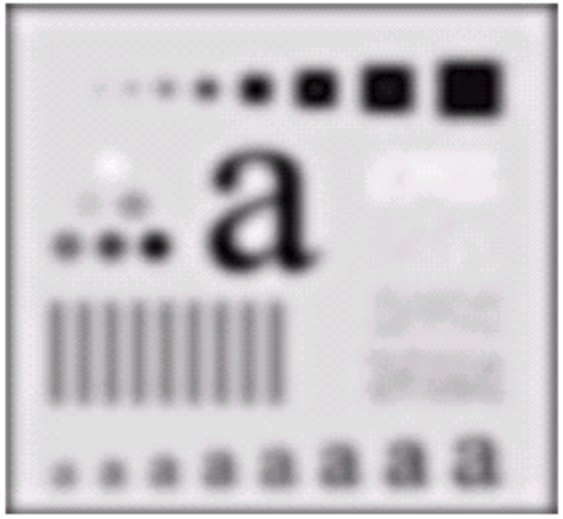


Result of filtering with Gaussian filter with cutoff radius 230

Lowpass Filters Compared

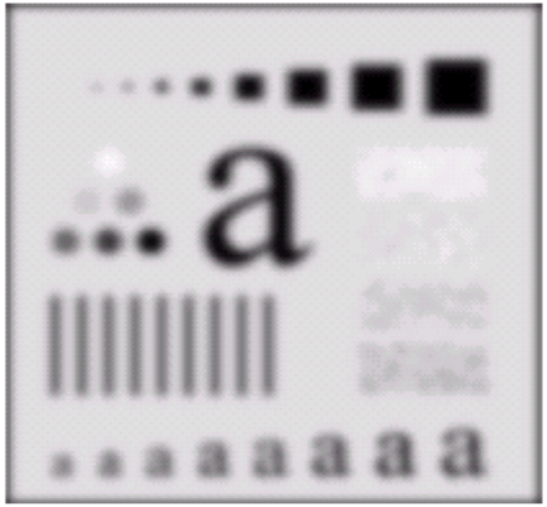
Result of filtering with ideal low pass filter of radius 15





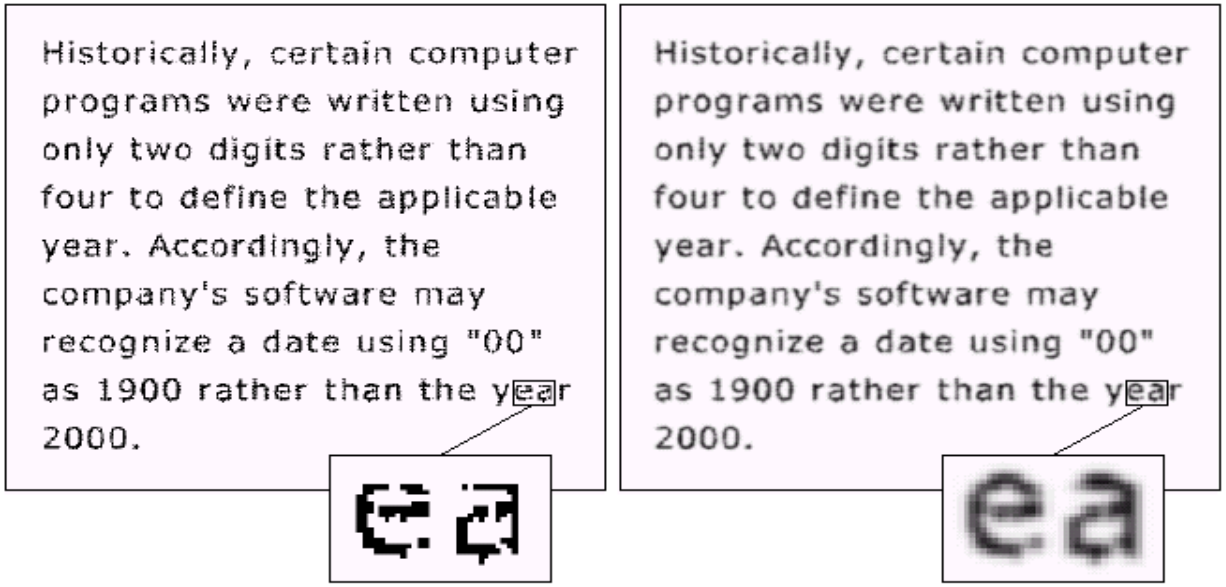
Result of filtering with Butterworth filter of order 2 and cutoff radius 15

Result of filtering with Gaussian filter with cutoff radius 15



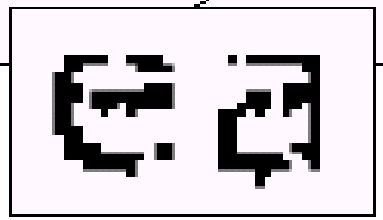
Lowpass Filtering Examples

A low pass Gaussian filter is used to connect broken text

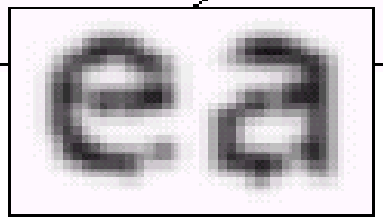


Lowpass Filtering Examples

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



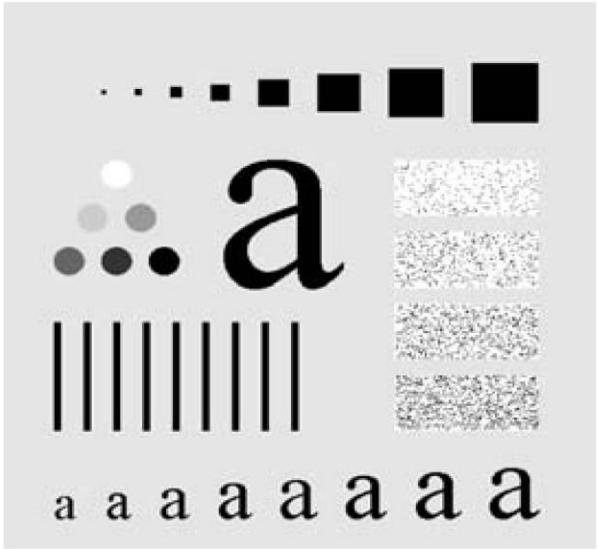
Lowpass Filtering Examples (cont…)

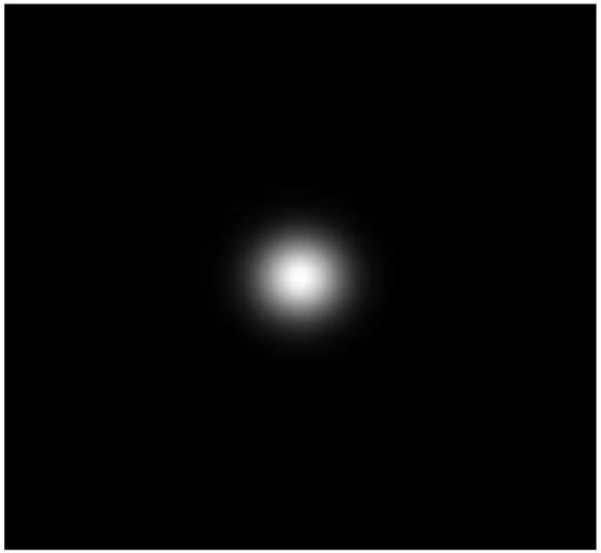
Different lowpass Gaussian filters used to remove blemishes in a photograph

Lowpass Filtering Examples (cont…)

Lowpass Filtering Examples (cont…)

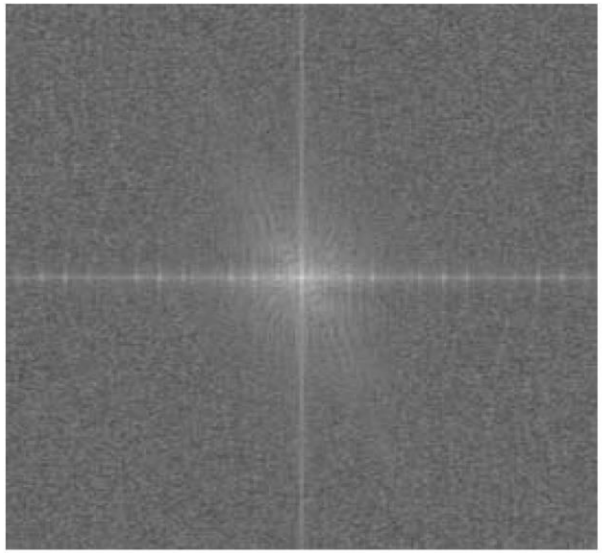
Original image





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:

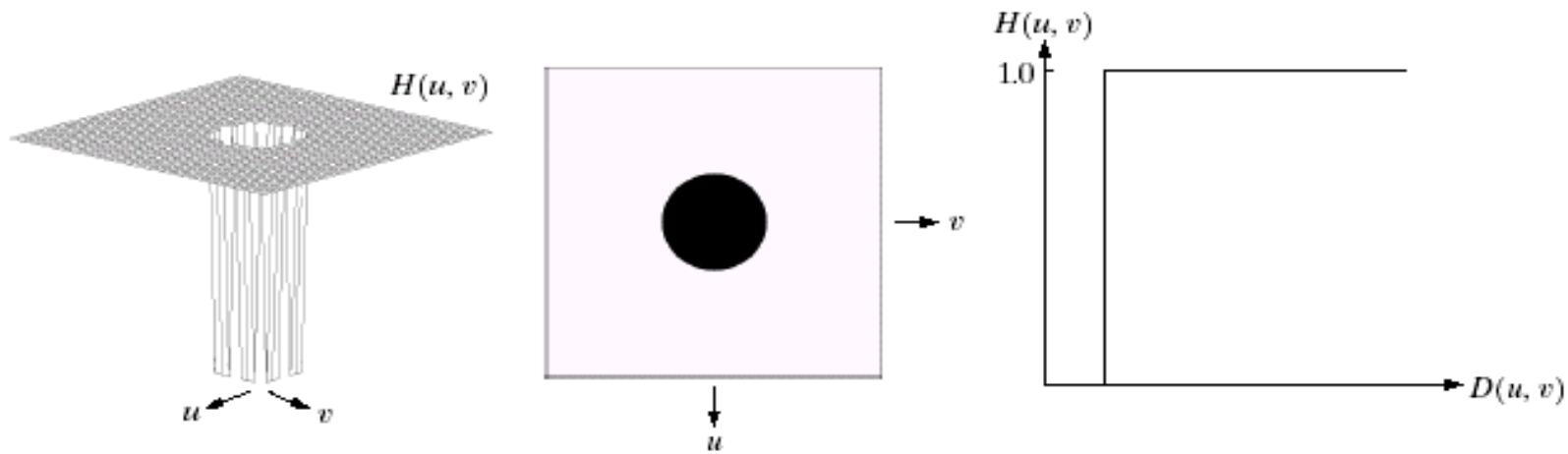
$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

Ideal High Pass Filters

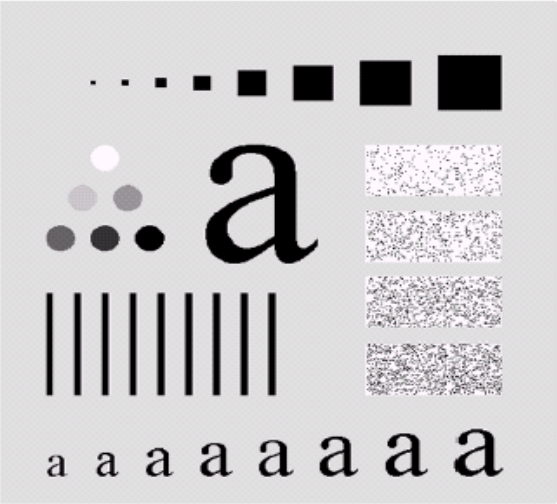
The ideal high pass filter is given as:

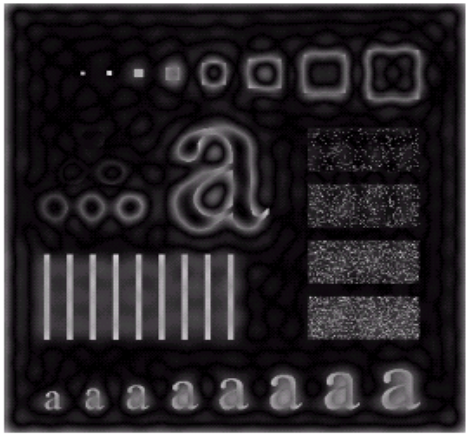
$$H(u,v) = \begin{cases} 0 \text{ if } D(u,v) \le D_0 \\ 1 \text{ if } D(u,v) > D_0 \end{cases}$$

where D_0 is the cut off distance as before

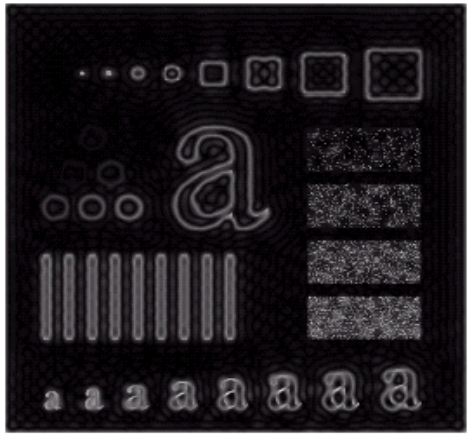


Ideal High Pass Filters (cont…)

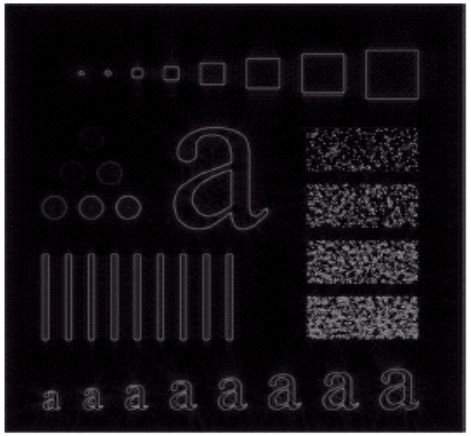




Results of ideal high pass filtering with D_0 = 15



Results of ideal high pass filtering with D_0 = 30



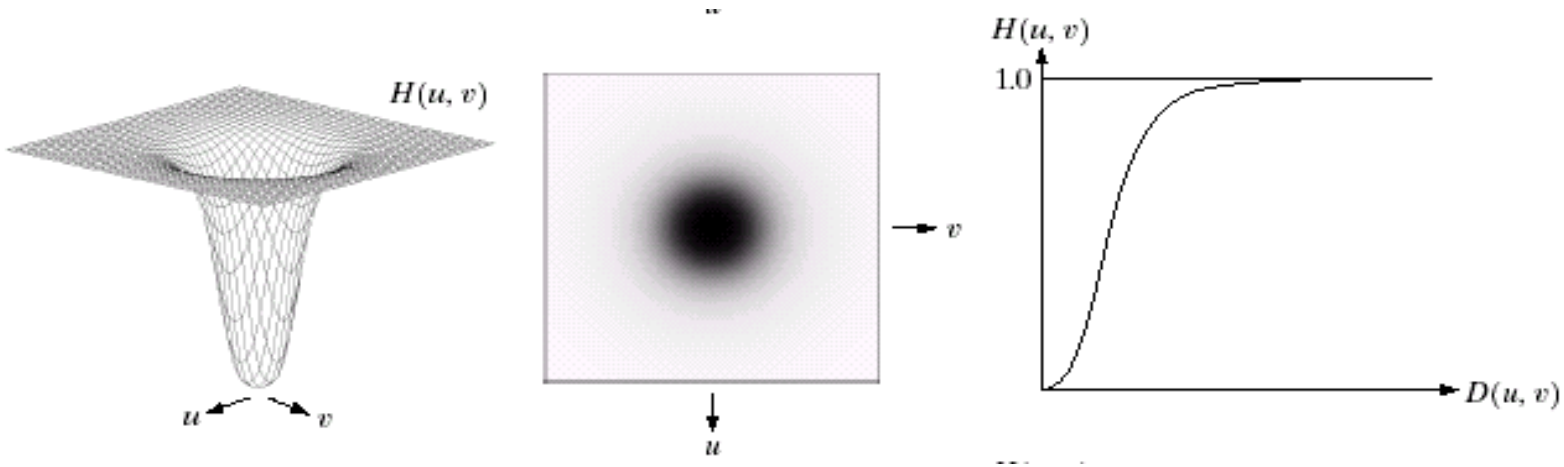
Results of ideal high pass filtering with D_0 = 80

Butterworth High Pass Filters

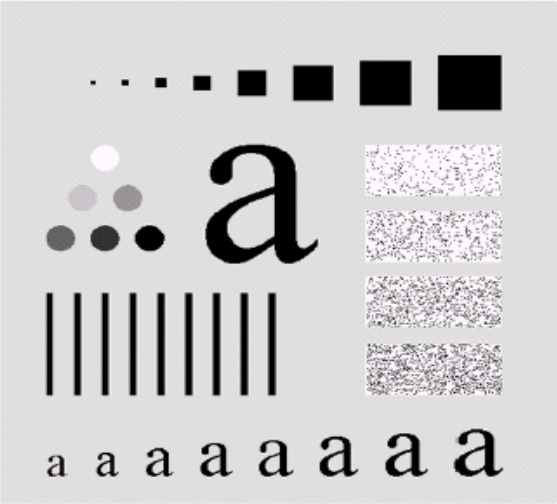
The Butterworth high pass filter is given as:

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

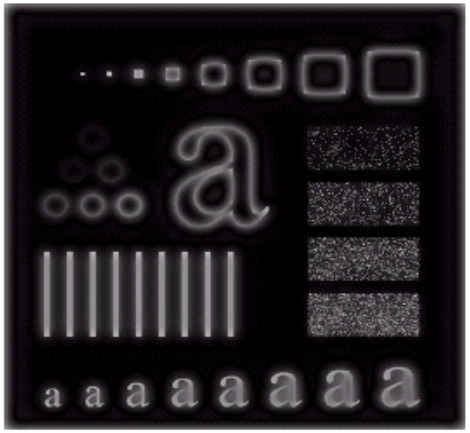
where n is the order and D_0 is the cut off distance as before

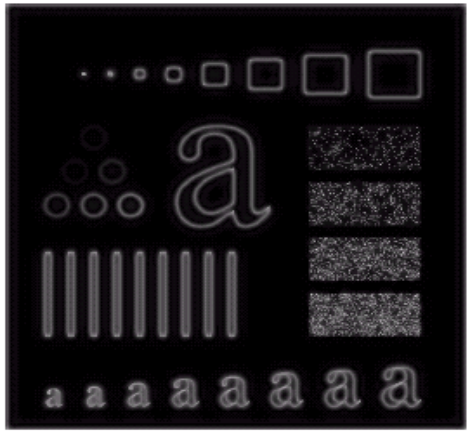


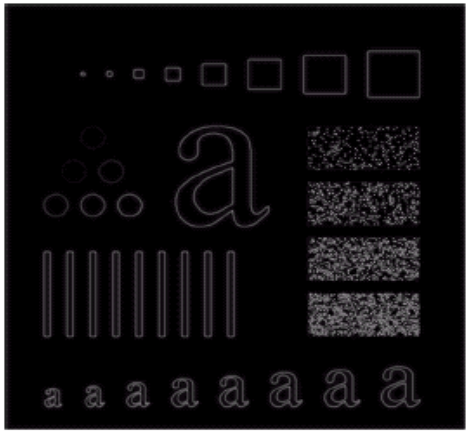
Butterworth High Pass Filters (cont…)



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 = 80

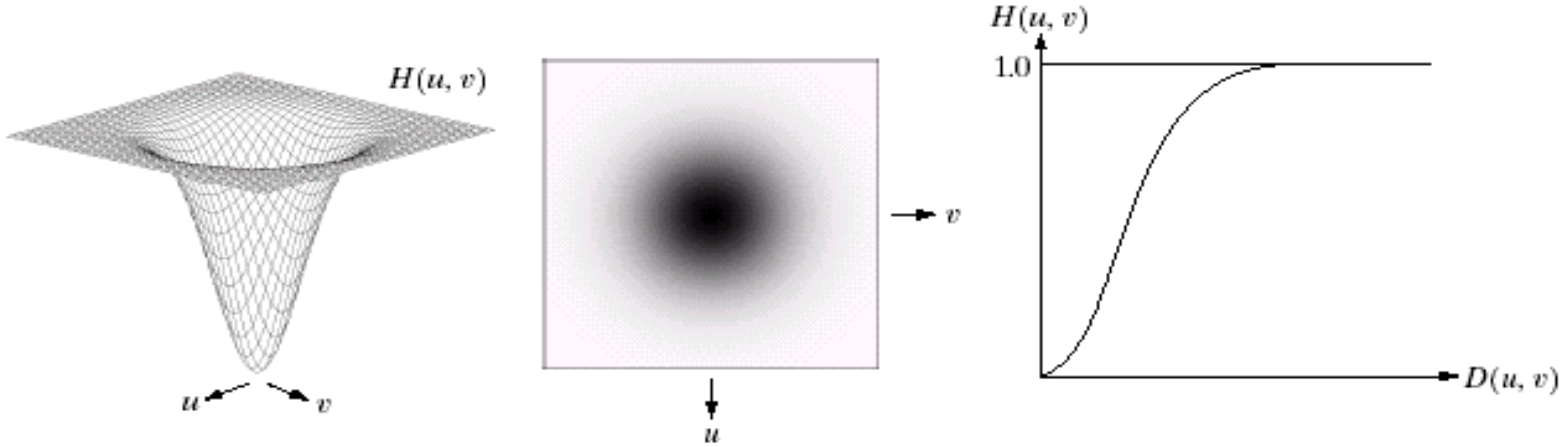
Results of Butterworth high pass filtering of order 2 with D_0 = 30

Gaussian High Pass Filters

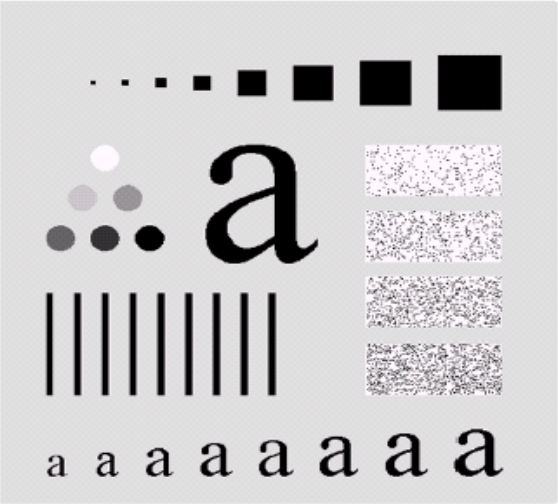
The Gaussian high pass filter is given as:

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

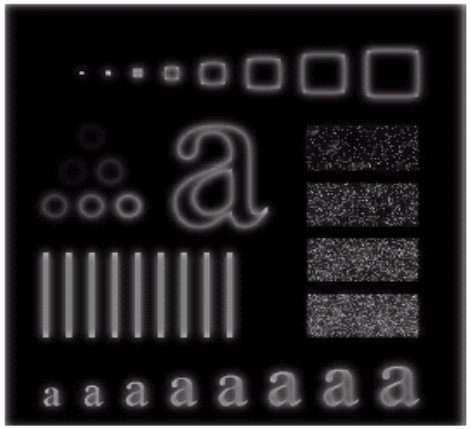
where D_0 is the cut off distance as before

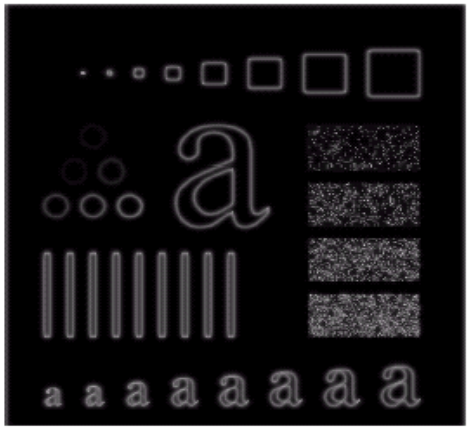


Gaussian High Pass Filters (cont…)



Results of Gaussian high pass filtering with D_0 = 15

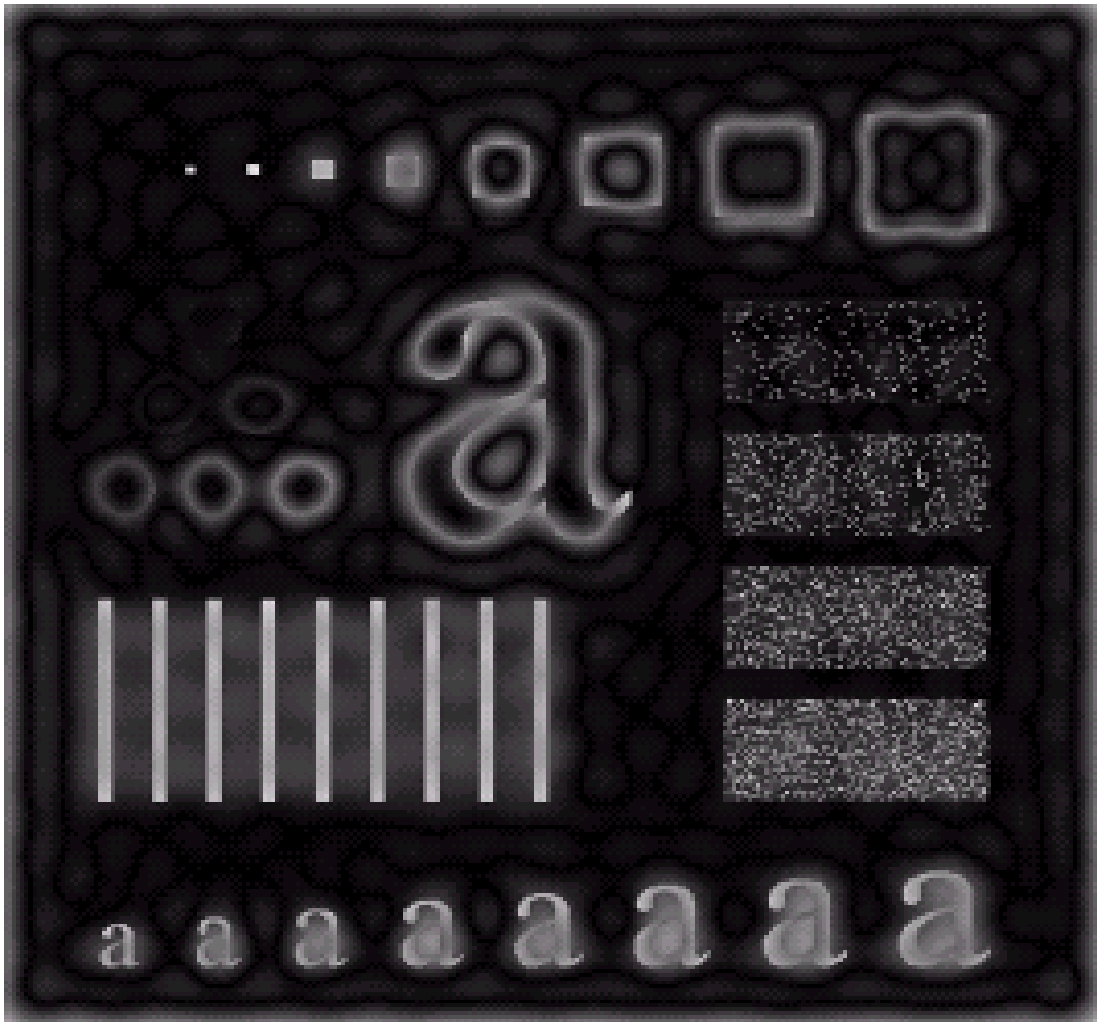




Results of Gaussian high pass filtering with D_0 = 80

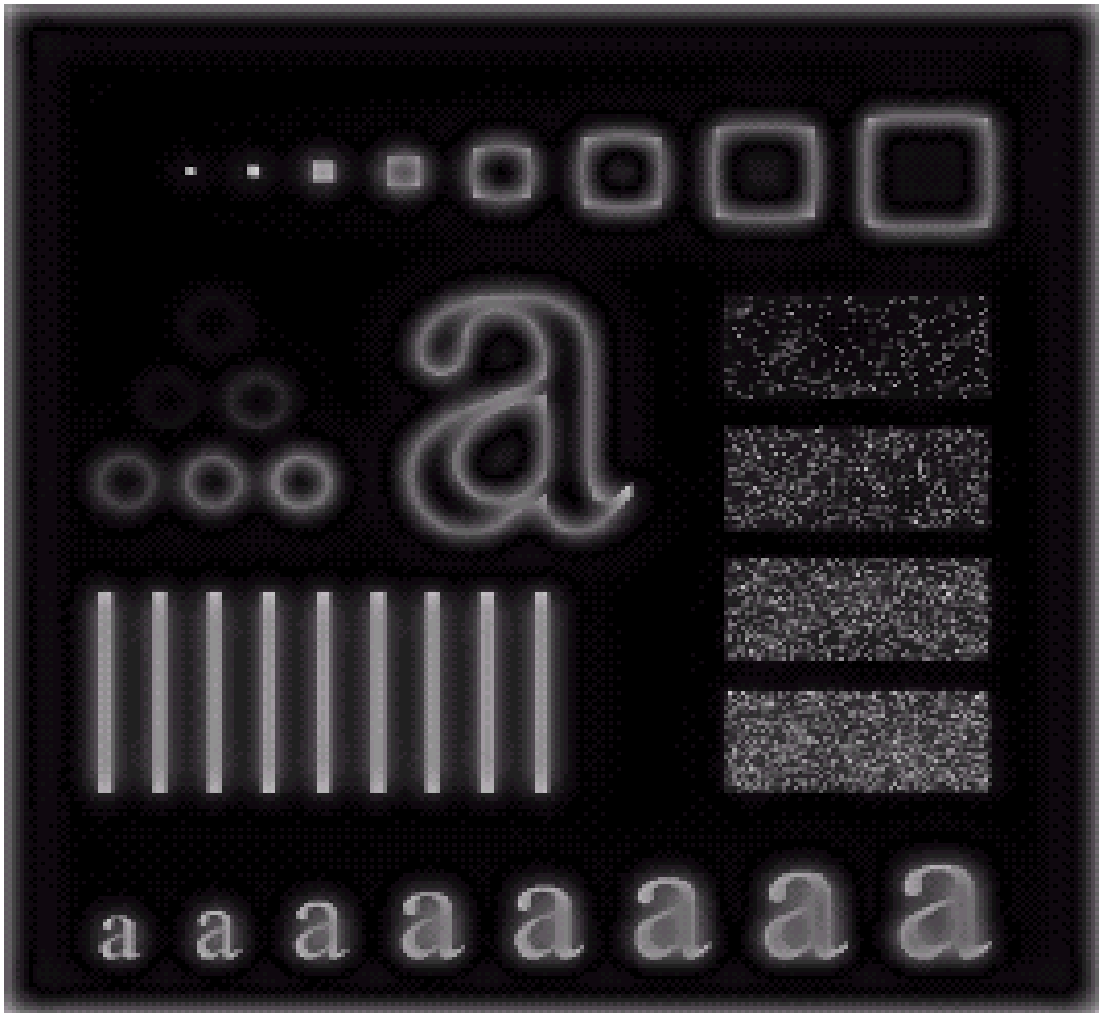
Results of Gaussian high pass filtering with D_0 = 30

Highpass Filter Comparison



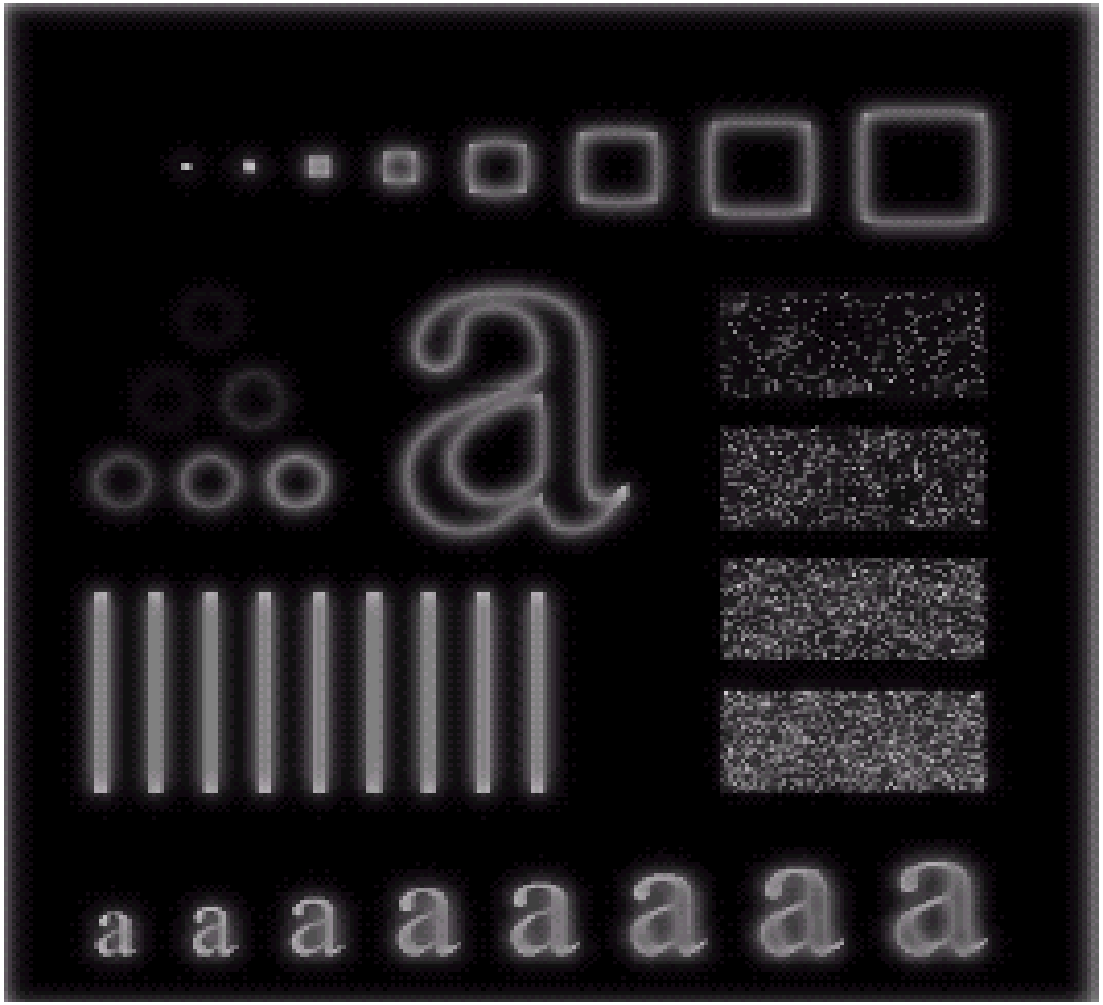
Results of ideal high pass filtering with D_0 = 15

Highpass Filter Comparison



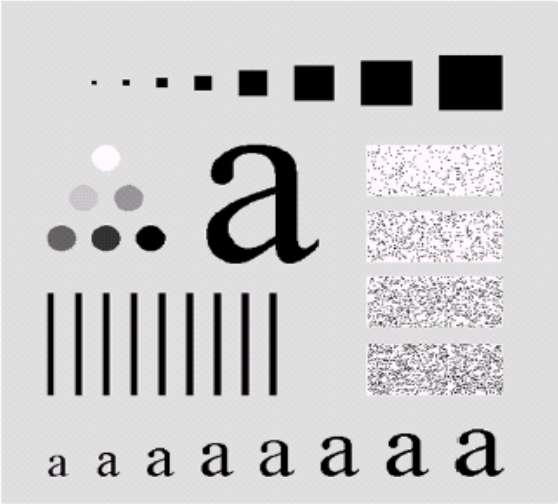
Results of Butterworth high pass filtering of order 2 with D_0 = 15

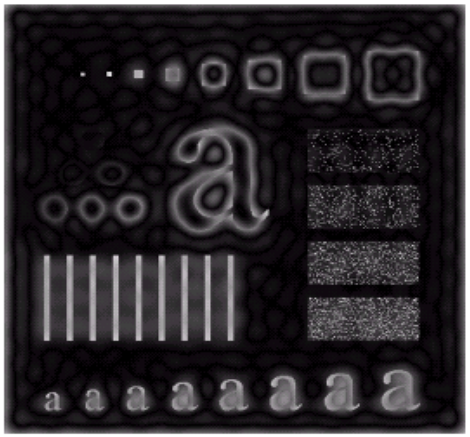
Highpass Filter Comparison



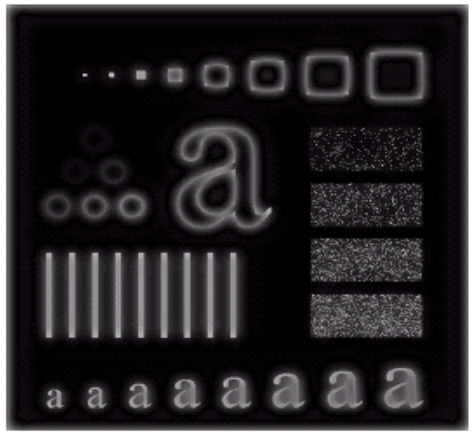
Results of Gaussian high pass filtering with D_0 = 15

Highpass Filter Comparison

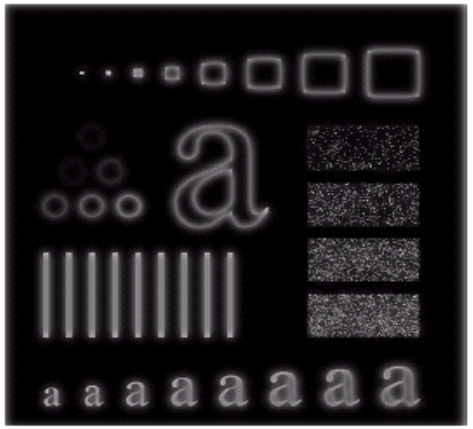




Results of ideal high pass filtering with D_0 = 15

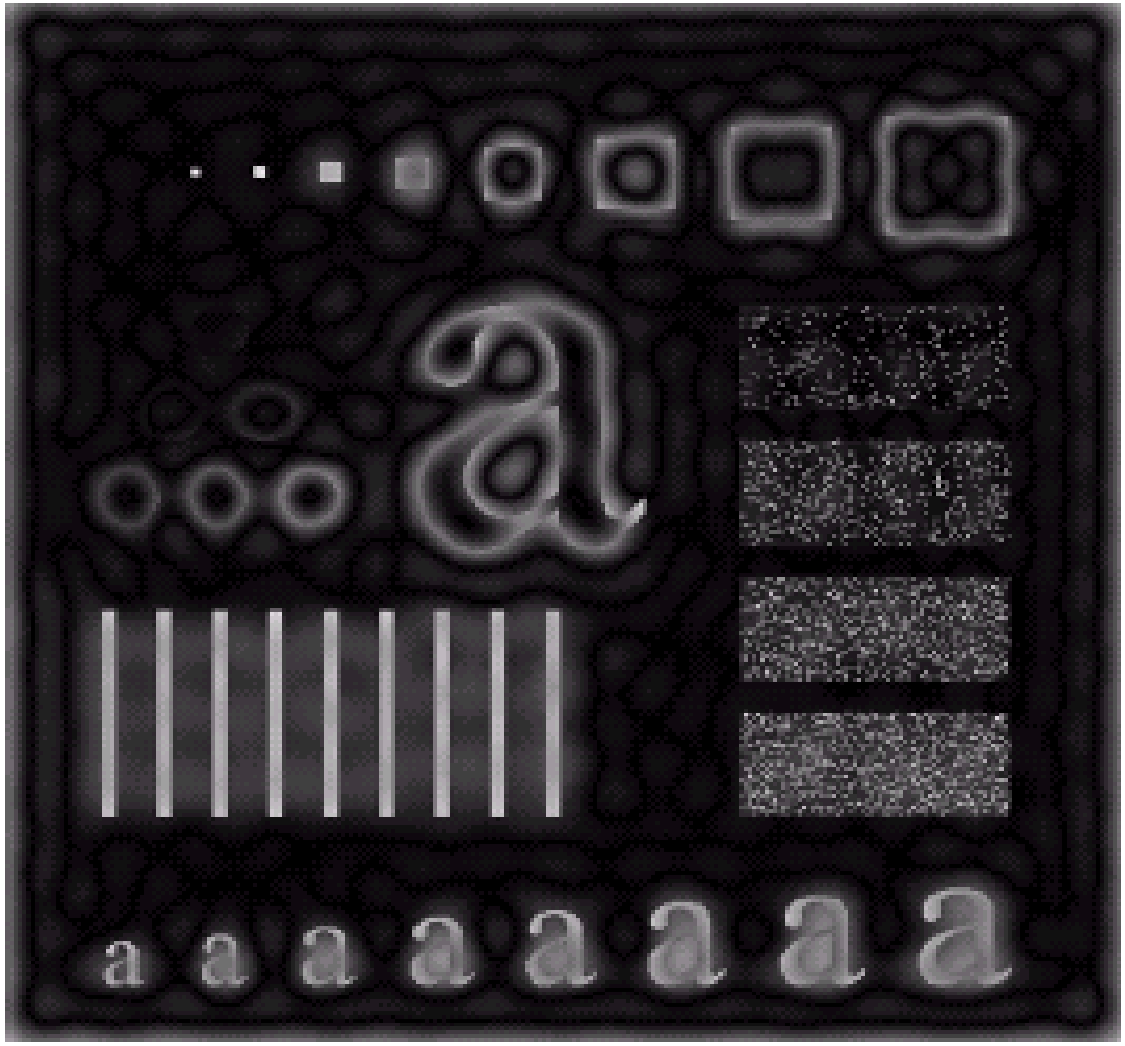


Results of Butterworth high pass filtering of order 2 with D_0 = 15



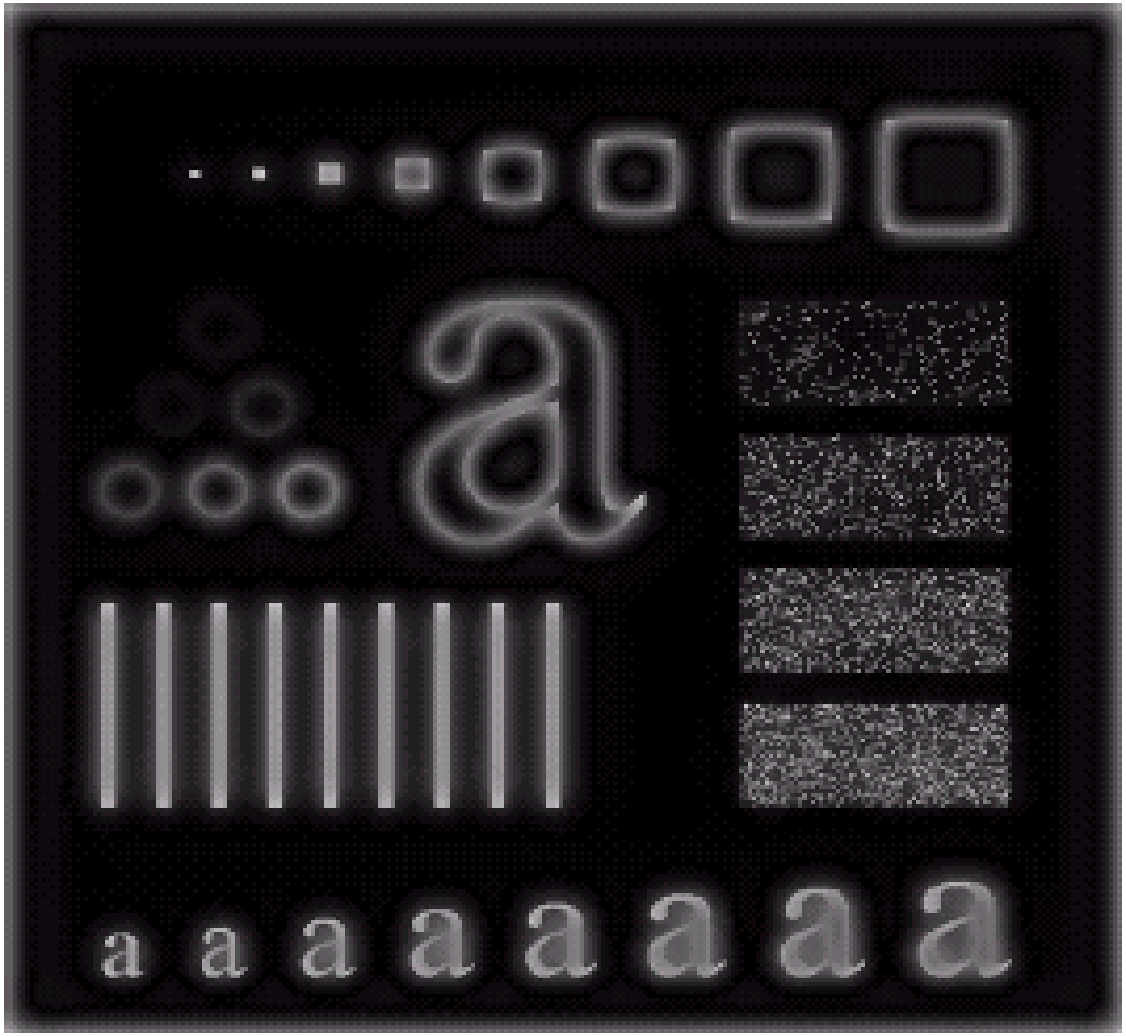
Results of Gaussian high pass filtering with D_0 = 15

Highpass Filter Comparison



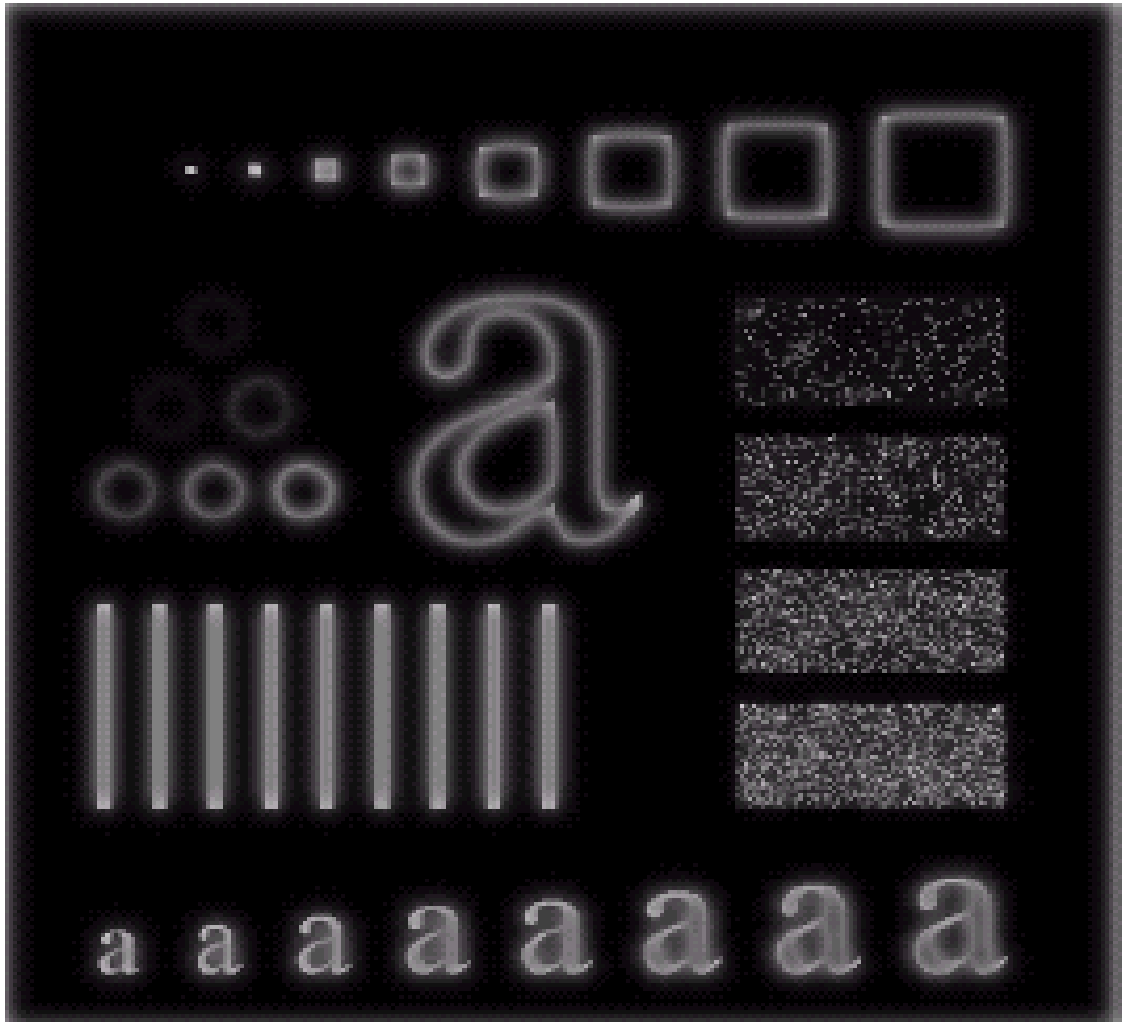
Results of ideal high pass filtering with D_0 = 15

Highpass Filter Comparison



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

Highpass Filter Comparison

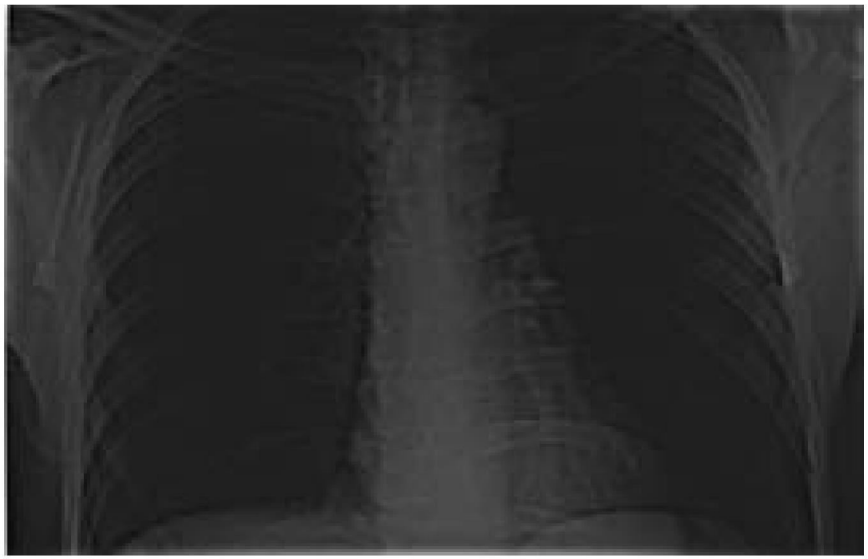


Results of Gaussian high pass filtering with D_0 = 15

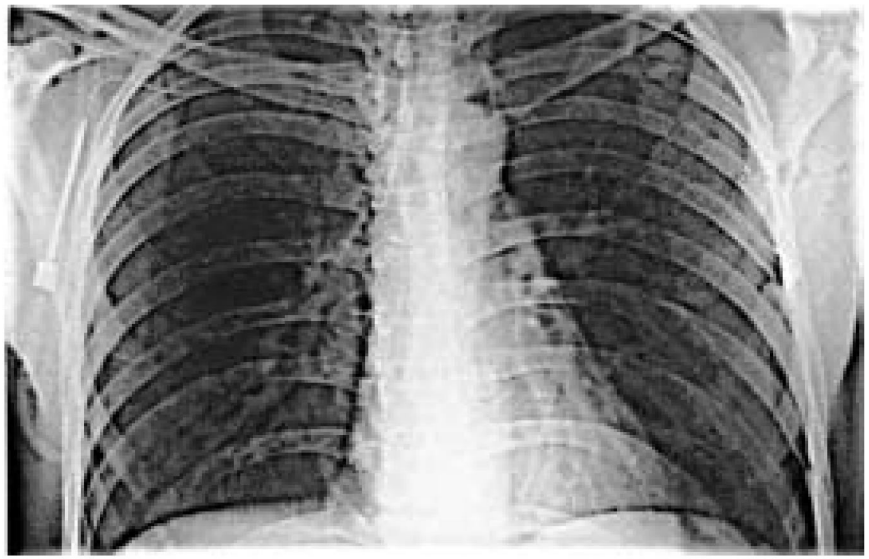
Highpass Filtering Example

Original image

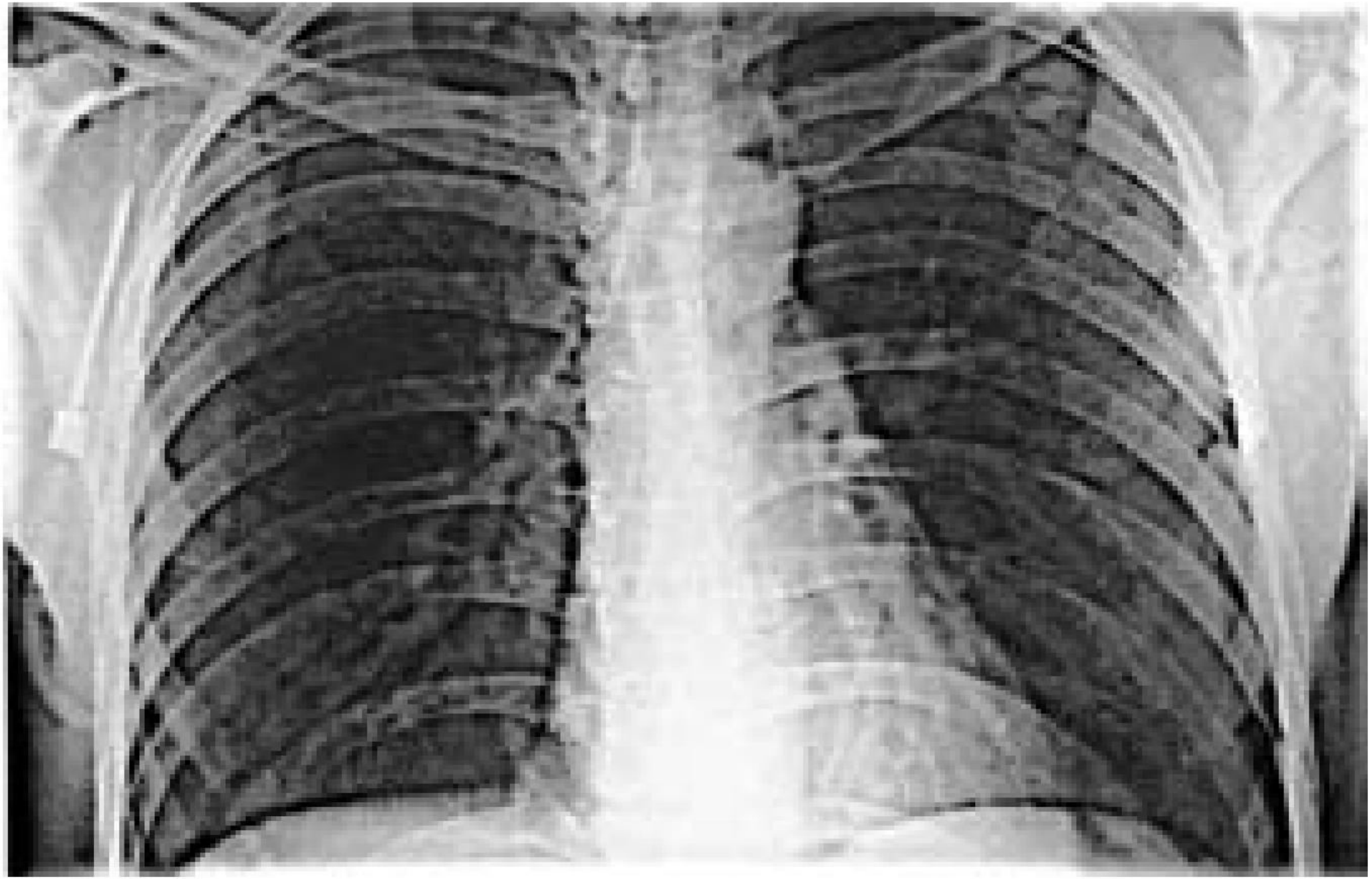
Highpass filtering result



High frequency emphasis result

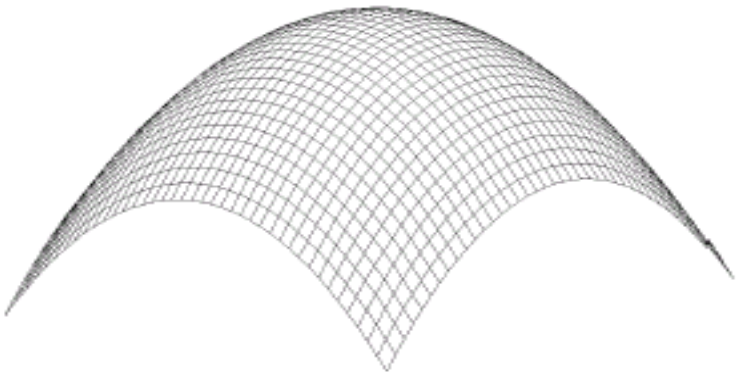


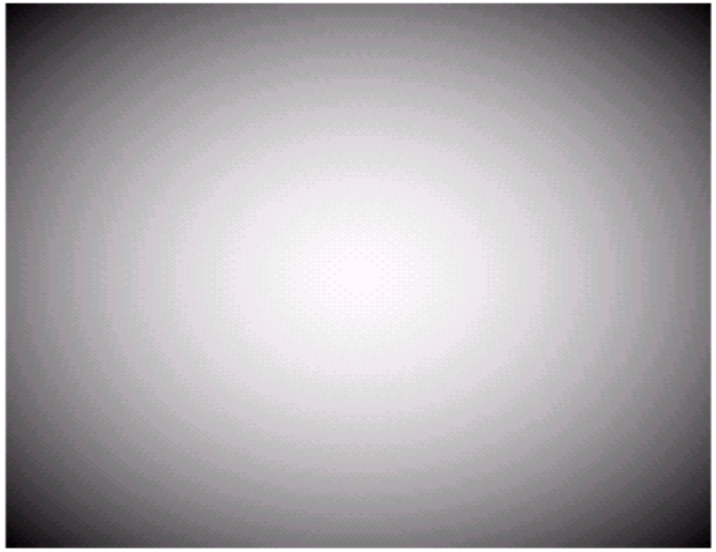
After histogram equalisation



Laplacian In The Frequency Domain

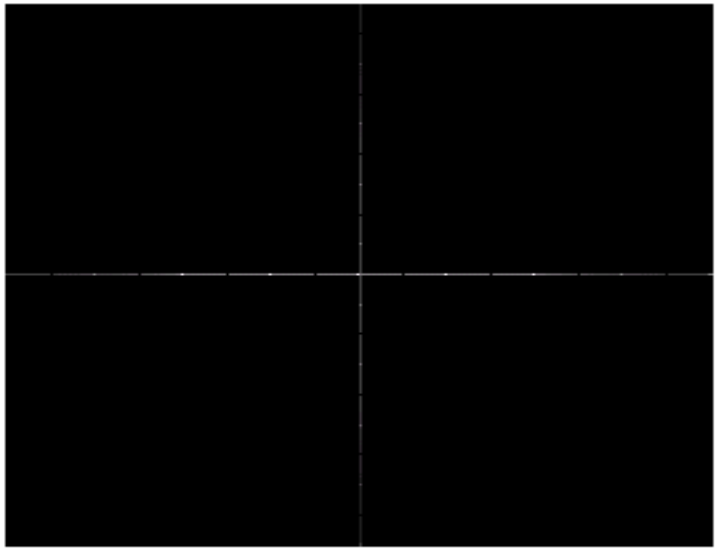
Laplacian in the frequency domain

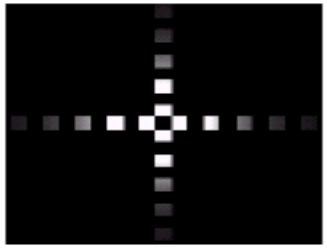


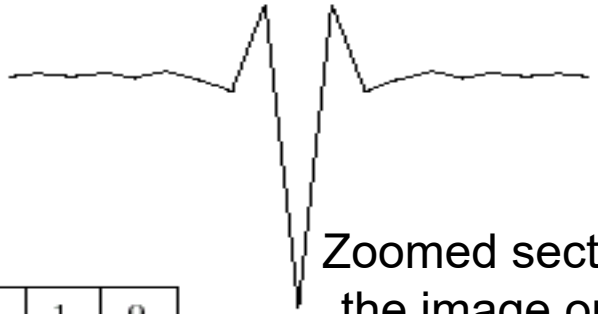


2-D image of Laplacian in the frequency domain

Inverse DFT of Laplacian in the frequency domain







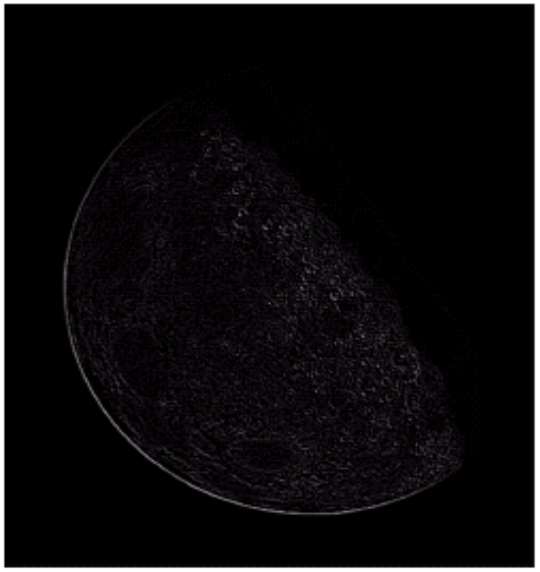
0	1	0
1	−4	1
0	1	0

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

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

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

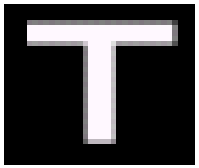
Summary

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

Imteresting Application Of Frequency Domain Filtering



Imteresting Application Of Frequency Domain Filtering

Questions?

Digital Image Processing

Lecture - 8

Image Restoration:

Noise Removal

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Contents

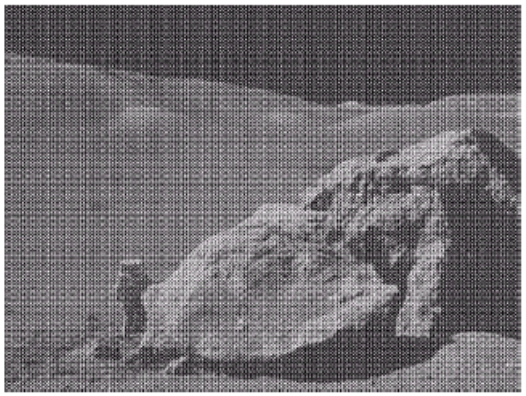
In this lecture we will look at image restoration techniques used for noise removal

- 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 Corruption Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	51	52	52	56	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
148	154	157	160	163	167	170
151	155	159	162	165	169	172

Image f (x, y)

y

Noisy Image

x

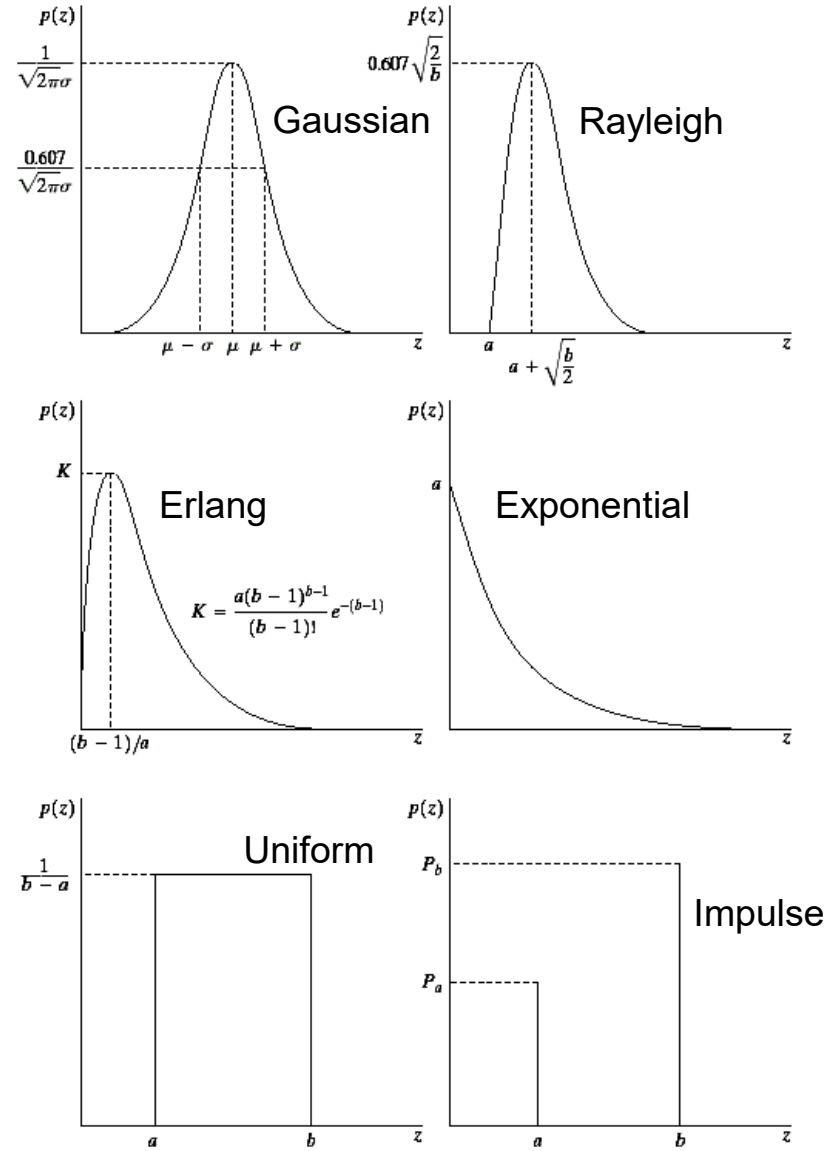
Image f (x, y)

y

Noise Models

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

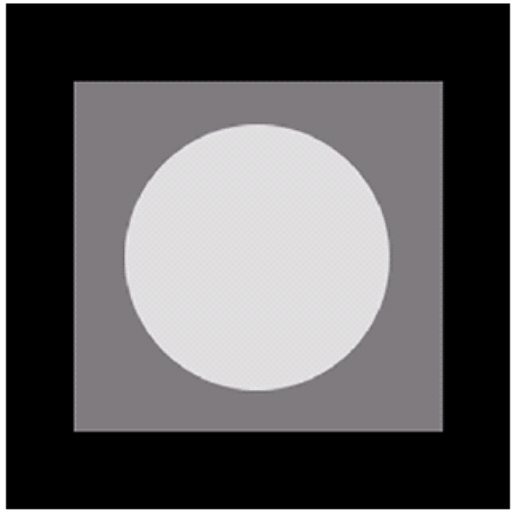
- Gaussian
  - Most common model
- Rayleigh
- Erlang
- Exponential
- Uniform
- Impulse
  - *Salt and pepper* noise



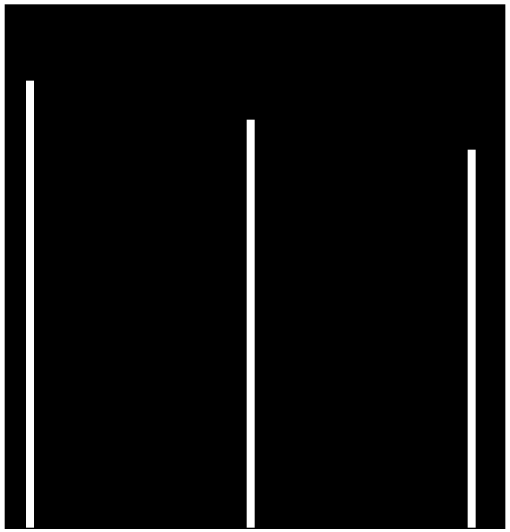
Noise Example

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

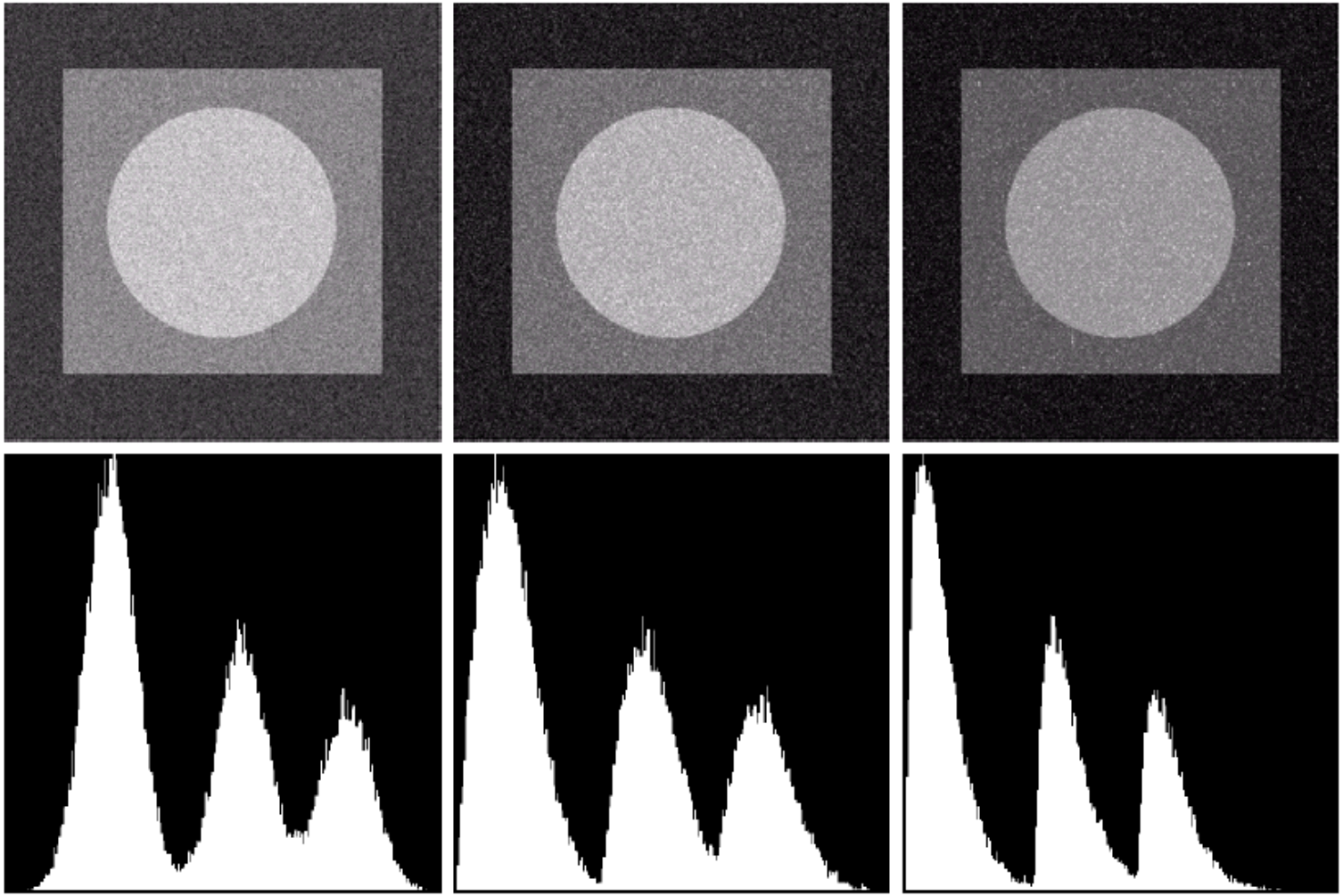


Image



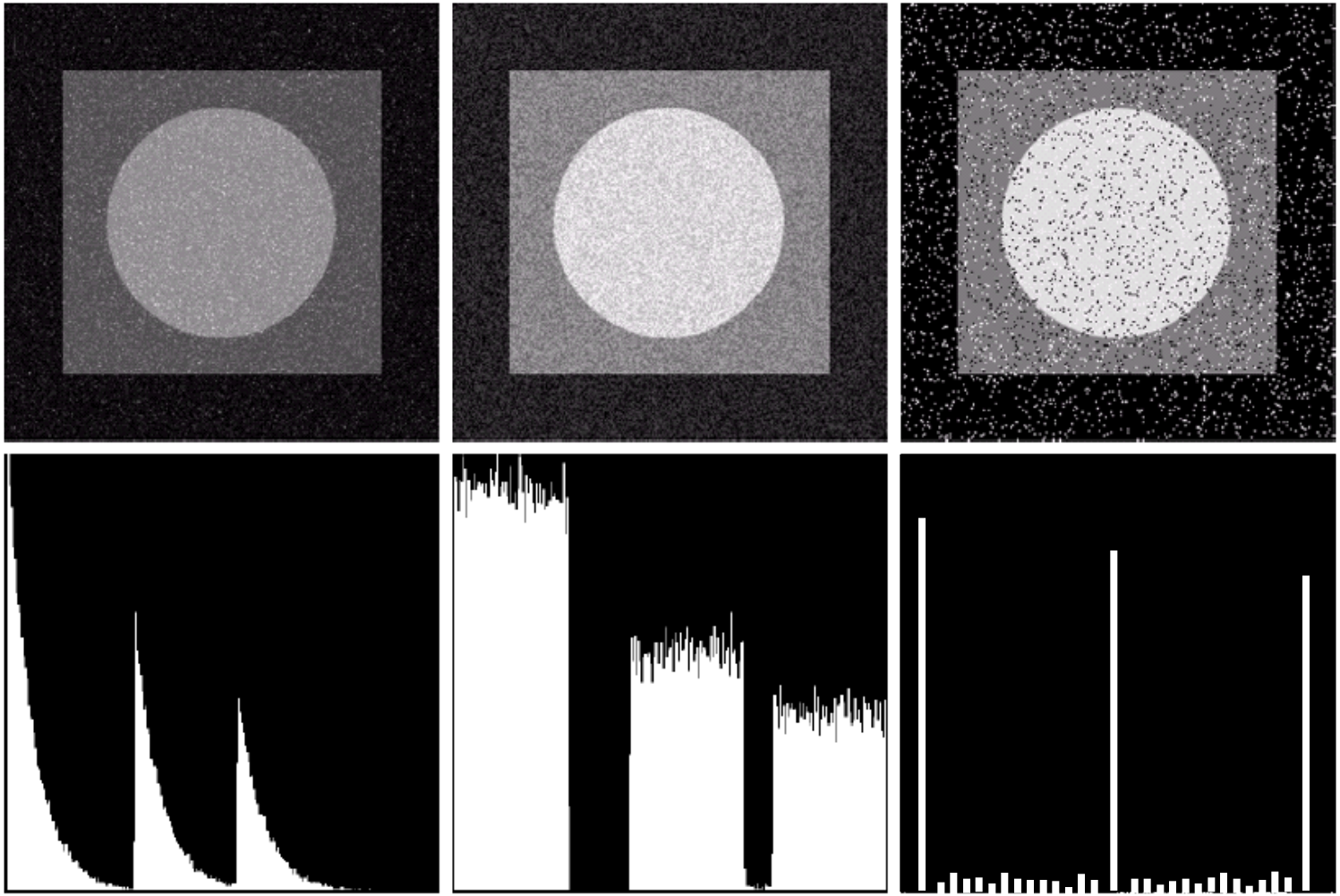
Histogram

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

Noise Removal Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
148	154	157	160	163	167	170
151	155	159	162	165	169	172

Image f (x, y)

y

Filtered Image

x

Image f (x, y)

y

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

Noise Removal Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
148	154	157	160	163	167	170
151	155	159	162	165	169	172

Image f (x, y)

y

***Filtered** Image*

x

Image f (x, y)

y

Other Means (cont…)

Harmonic Mean:

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

Works well for salt noise, but fails for pepper noise

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

Noise Corruption Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
50	54	57	60	63	67	70
51	55	59	62	65	69	72

Image f (x, y)

y

Filtered Image

x

Image f (x, y)

y

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 Corruption Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
50	54	57	60	63	67	70
51	55	59	62	65	69	72

Image f (x, y)

y

Filtered Image

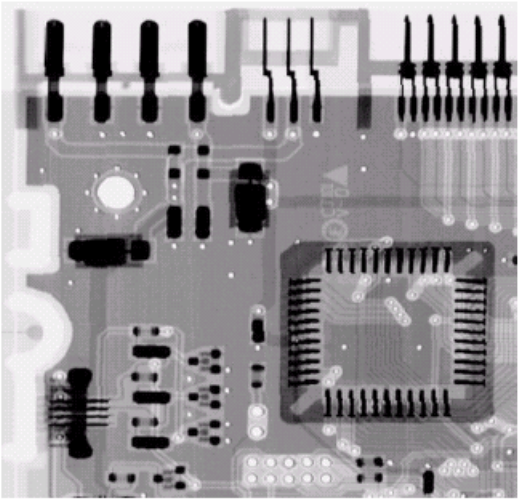
x

Image f (x, y)

y

Noise Removal Examples

Original Image



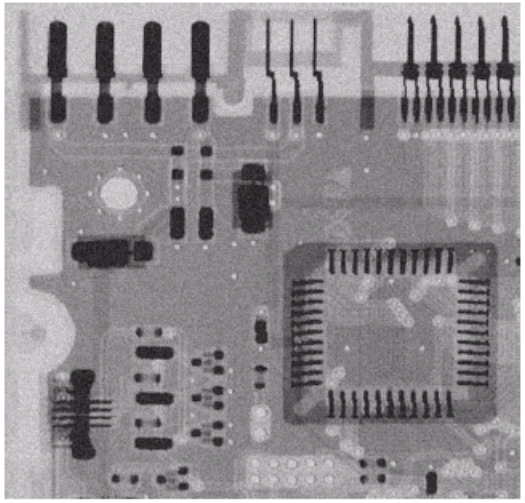
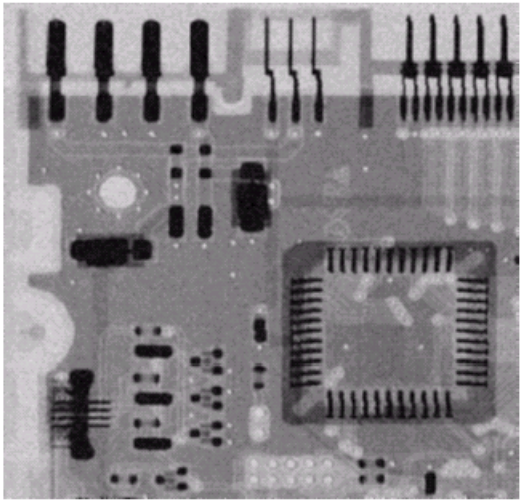
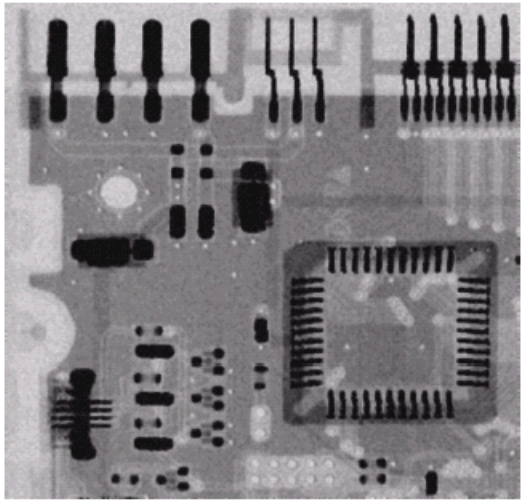


Image Corrupted By Gaussian Noise

After A 3*3 Arithmetic Mean Filter

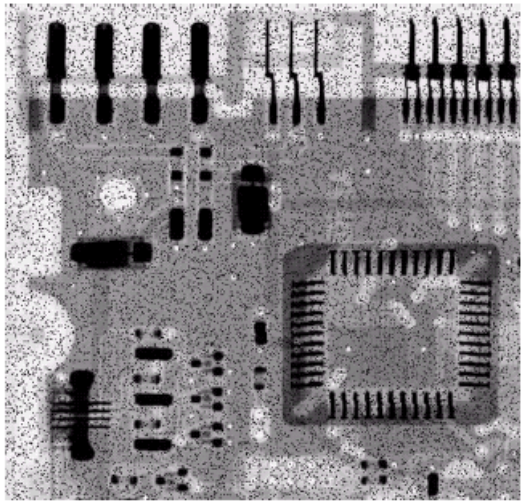




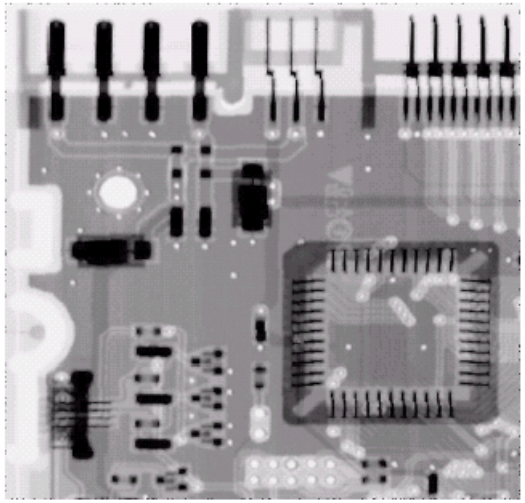
After A 3*3 Geometric Mean Filter

Noise Removal Examples (cont…)

Image Corrupted By Pepper Noise



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



Noise Removal Examples (cont…)

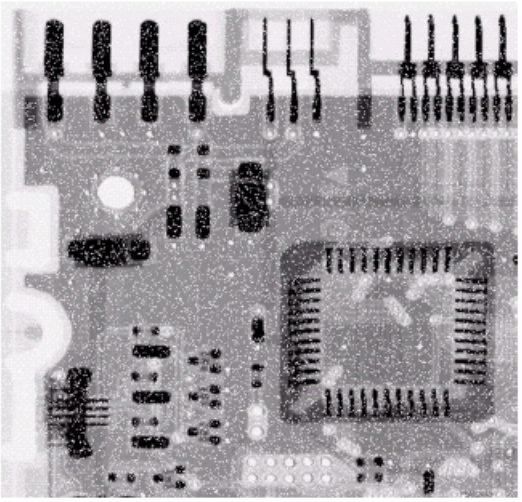
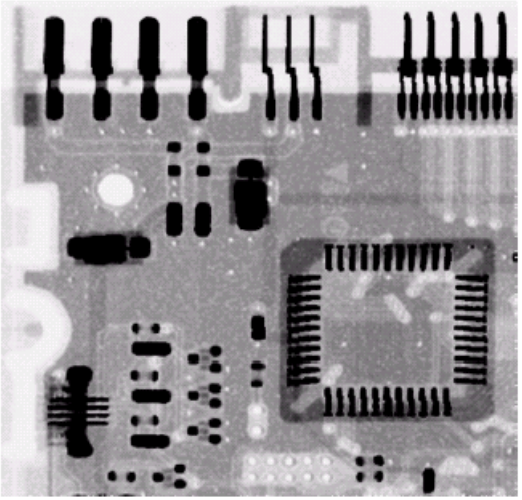


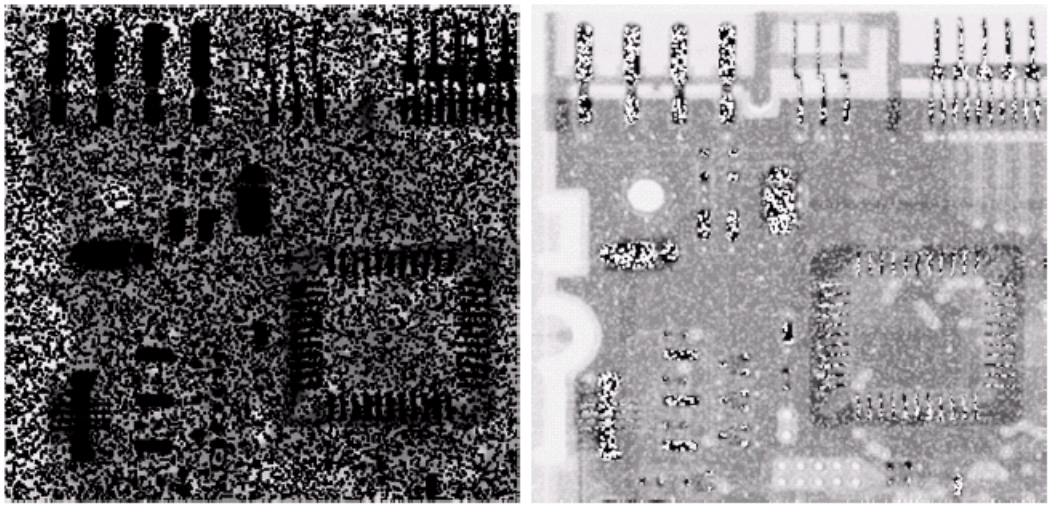
Image Corrupted By Salt Noise



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

Noise Corruption Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
50	54	57	60	63	67	70
51	55	59	62	65	69	72

Image f (x, y)

y

Filtered Image

x

Image f (x, y)

y

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

Noise Corruption Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
50	54	57	60	63	67	70
51	55	59	62	65	69	72

Image f (x, y)

y

Filtered Image

x

Image f (x, y)

y

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

Noise Corruption Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
50	54	57	60	63	67	70
51	55	59	62	65	69	72

Image f (x, y)

y

Filtered Image

x

Image f (x, y)

y

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 Corruption Example

Original Image

x

54	52	57	55	56	52	51
50	49	51	50	52	53	58
51	204	52	52	0	57	60
48	50	51	49	53	59	63
49	51	52	55	58	64	67
50	54	57	60	63	67	70
51	55	59	62	65	69	72

Image f (x, y)

y

Filtered Image

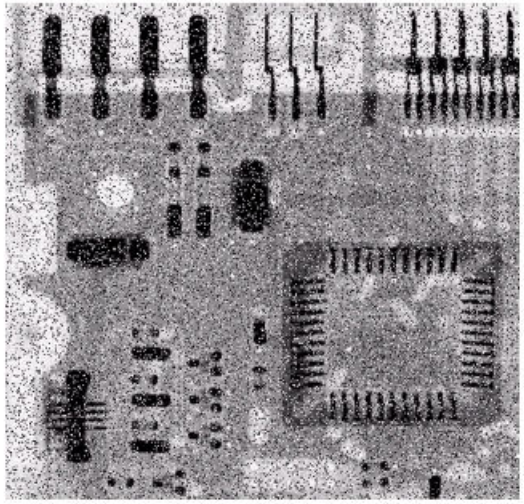
x

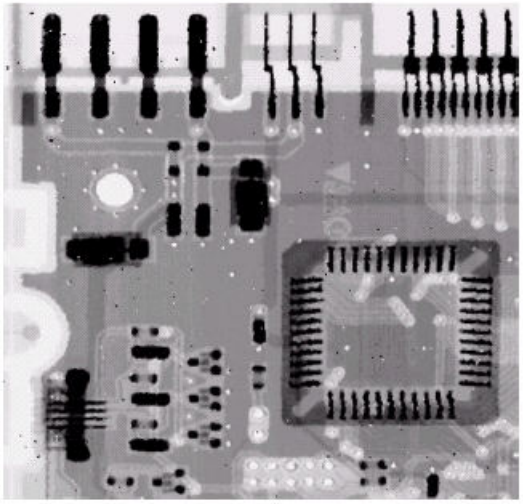
Image f (x, y)

y

Noise Removal Examples

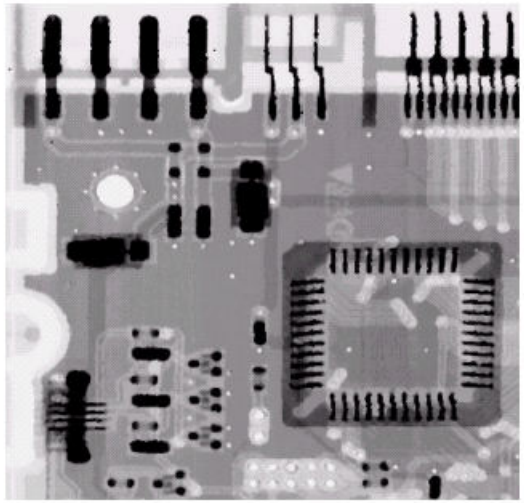
Image Corrupted By Salt And Pepper Noise

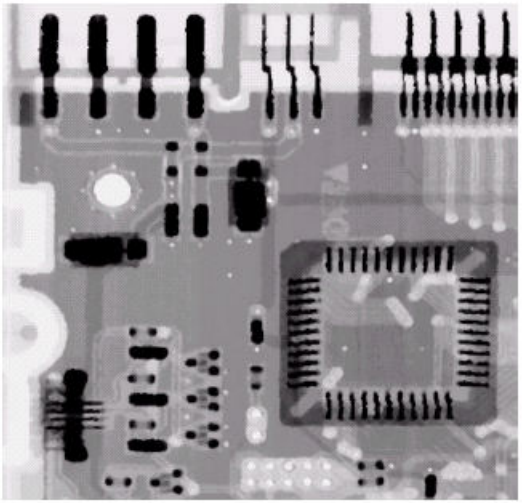




Result of 1 Pass With A 3*3 Median Filter

Result of 2 Passes With A 3*3 Median Filter

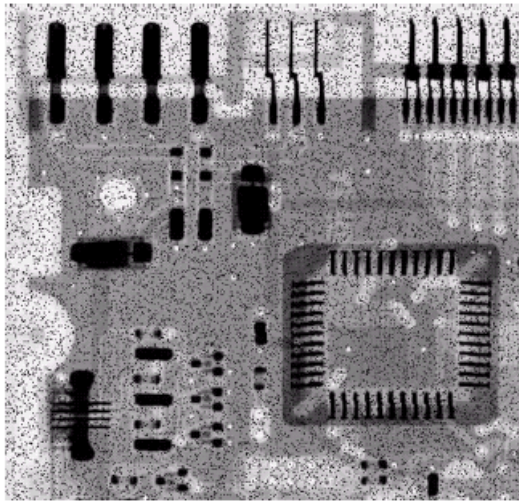




Result of 3 Passes With A 3*3 Median Filter

Noise Removal Examples (cont…)

Image Corrupted By Pepper Noise



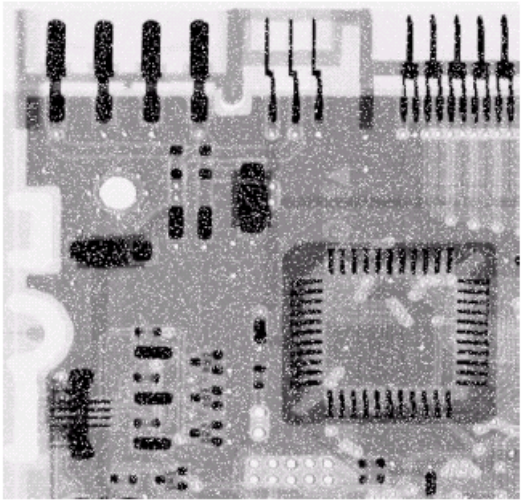
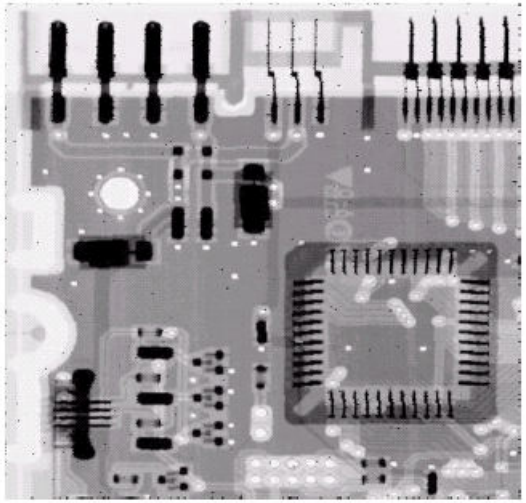
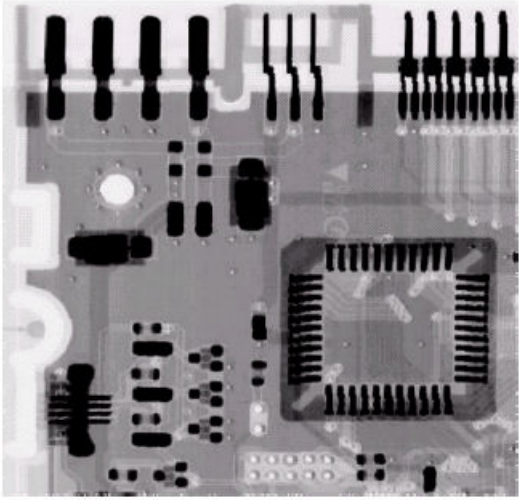


Image Corrupted By Salt Noise

Result Of Filtering Above With A 3*3 Max Filter

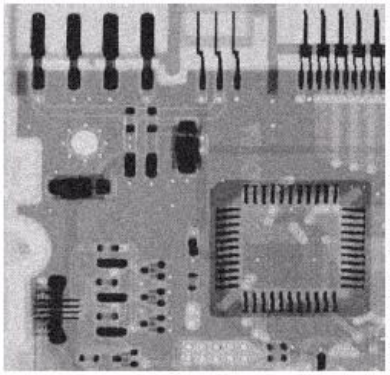




Result Of Filtering Above With A 3*3 Min Filter

Noise Removal Examples (cont…)

Image Corrupted By Uniform Noise



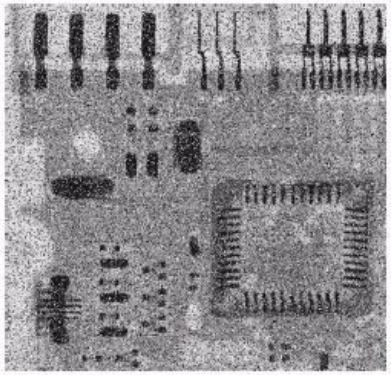
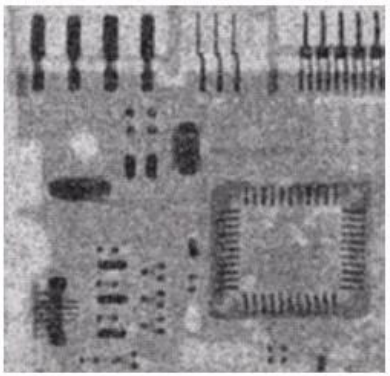
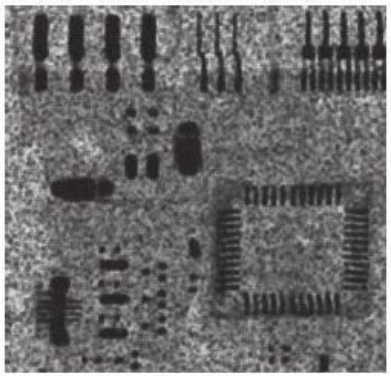


Image Further Corrupted By Salt and Pepper Noise

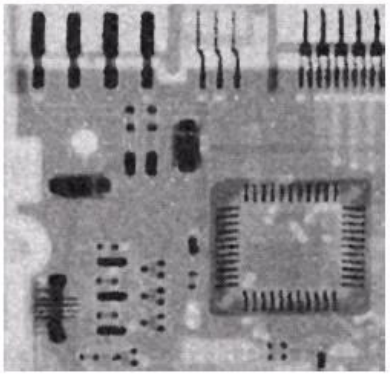
Filtered By 5*5 Arithmetic Mean Filter

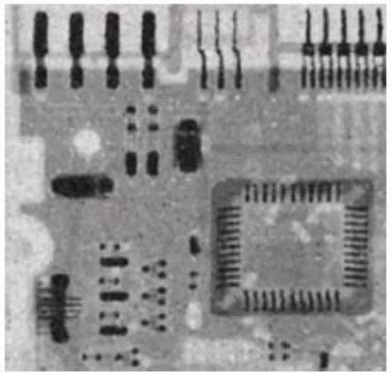




Filtered By 5*5 Geometric Mean Filter

Filtered By 5*5 Median Filter





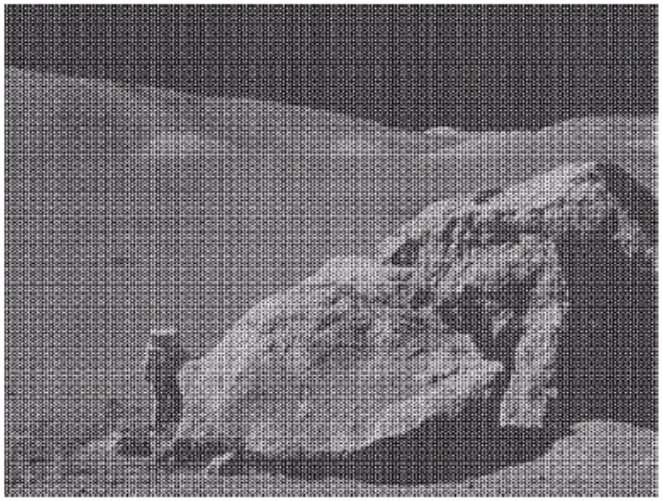
Filtered By 5*5 Alpha-Trimmed Mean Filter

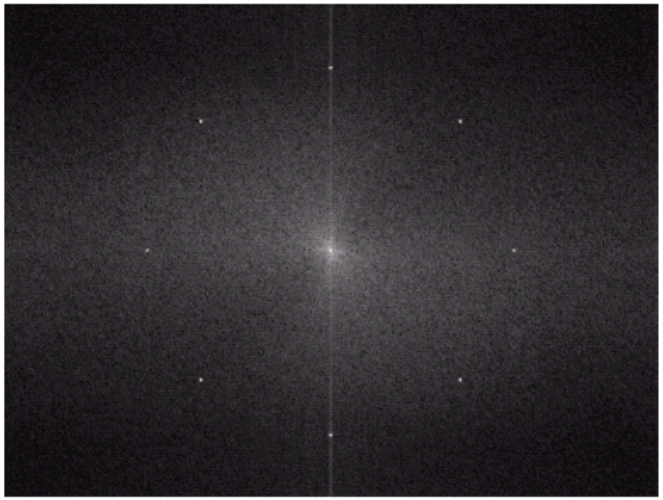
Periodic Noise

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

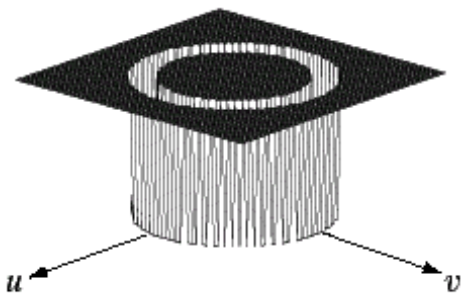
Band reject filters can be used for this purpose

An ideal band reject filter is given as follows:

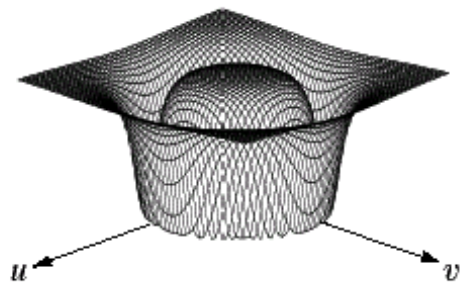
$$H(u,v) = \begin{cases} 1 & if\ D(u,v) < D_0 - \dfrac{W}{2} \\ 0 & if\ D_0 - \dfrac{W}{2} \le D(u,v) \le D_0 + \dfrac{W}{2} \\ 1 & if\ D(u,v) > D_0 + \dfrac{W}{2} \end{cases}$$

Band Reject Filters (cont…)

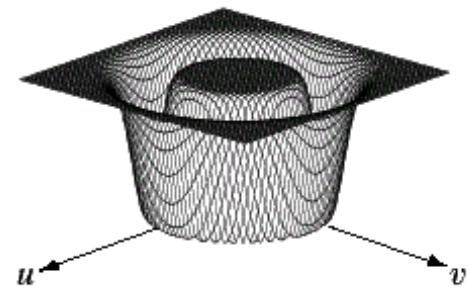
The ideal band reject filter is shown below, along with Butterworth and Gaussian versions of the filter



Ideal Band Reject Filter



Butterworth Band Reject Filter (of order 1)

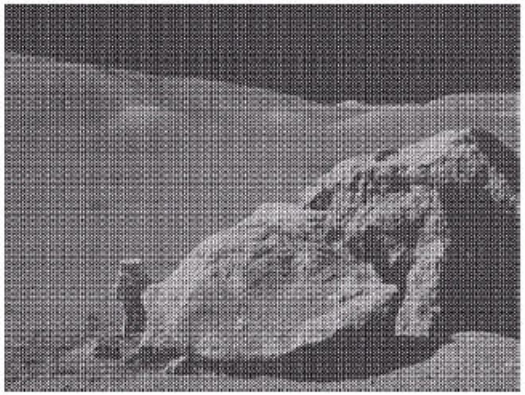


Gaussian Band Reject Filter

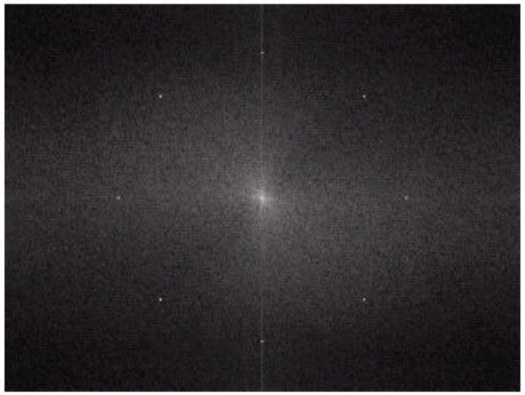
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

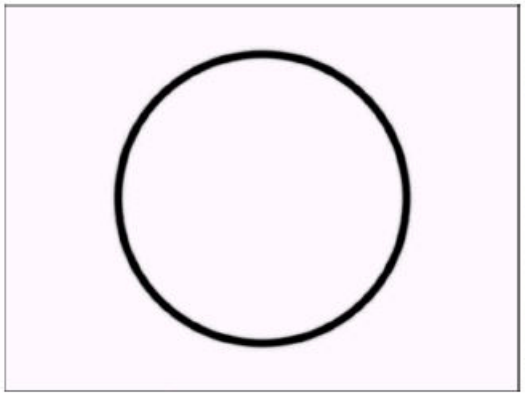
Band Reject Filter Example

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

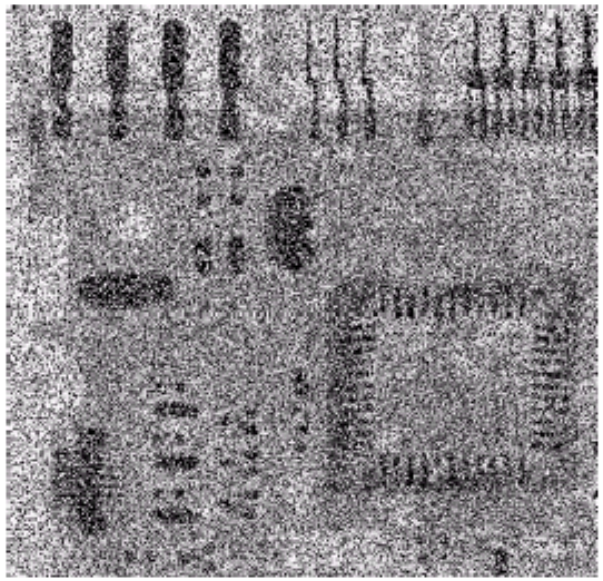
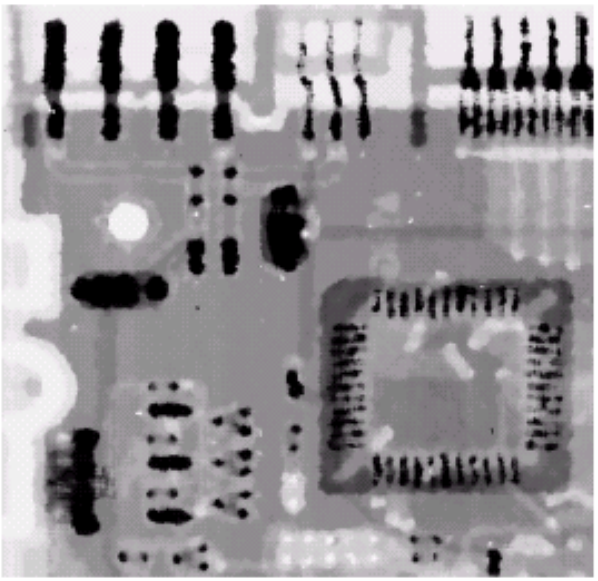
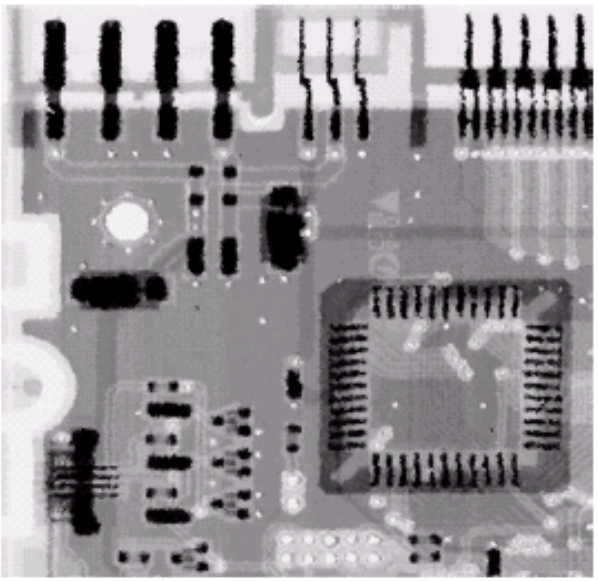


Image corrupted by salt and pepper noise with probabilities $P_a = P_b = 0.25$



Result of filtering with a 7 * 7 median filter



Result of adaptive median filtering with i = 7

Digital Image Processing

Lecture - 9

Image Segmentation: Thresholding

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Contents

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

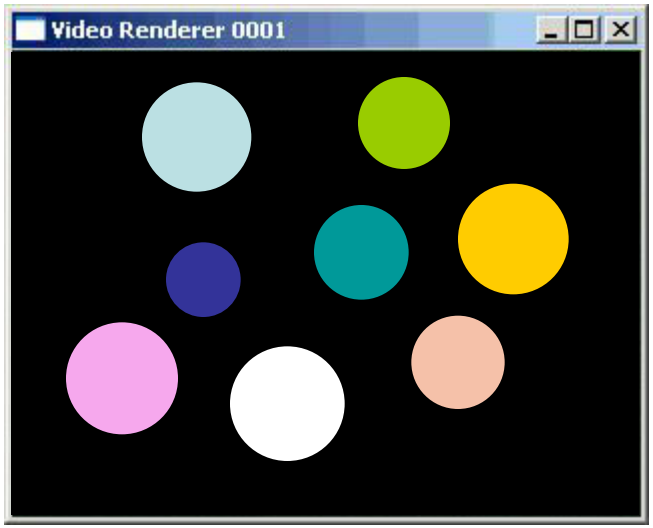
Today we will begin looking at automated image analysis by examining the thorny issue of image segmentation:

- The segmentation problem
- Finding points, lines and edges

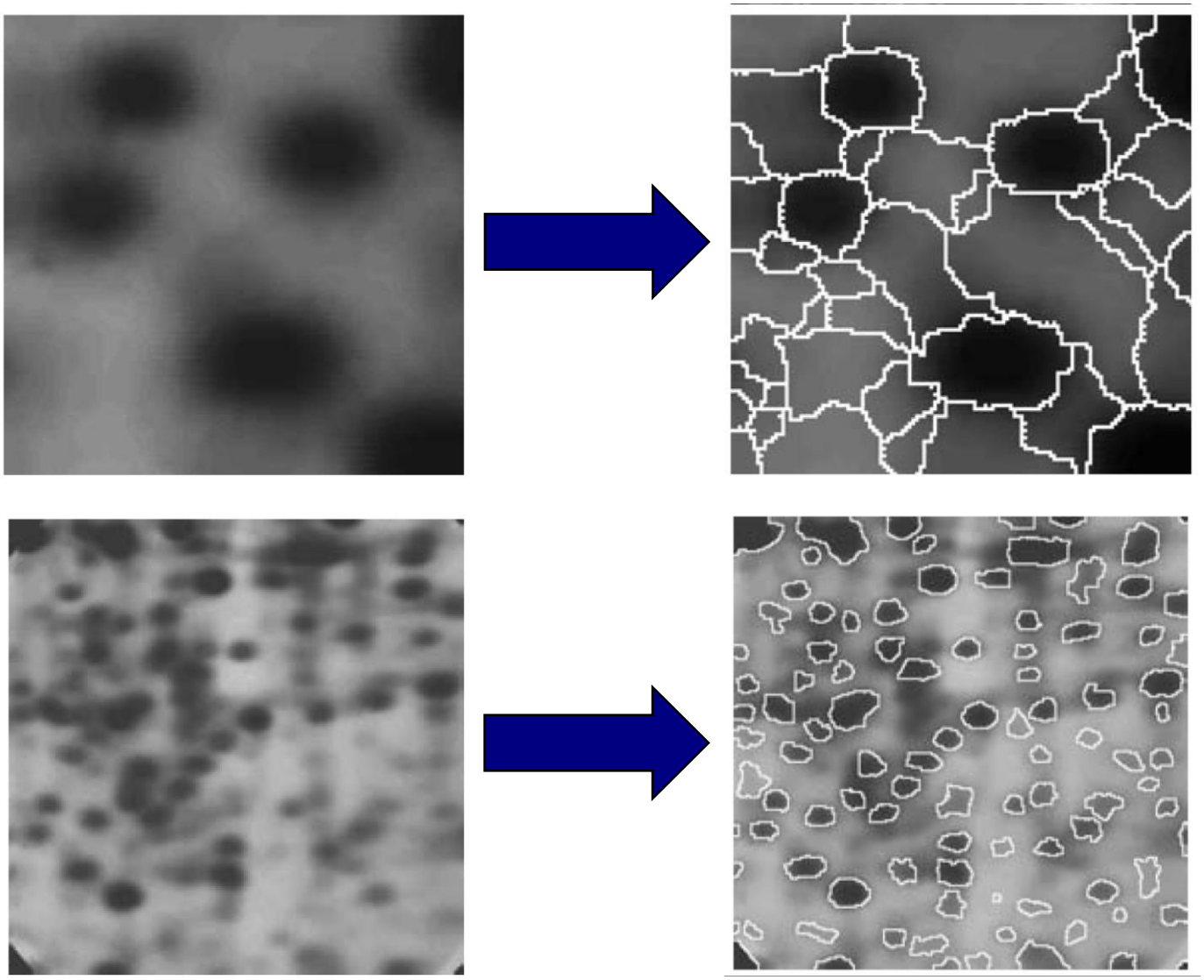
The Segmentation Problem

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



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

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

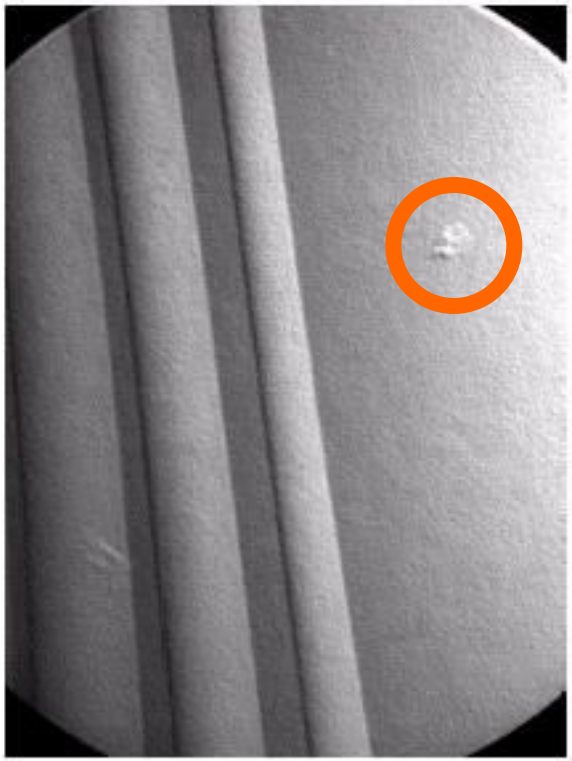
Point Detection

Point detection can be achieved simply using the mask below:

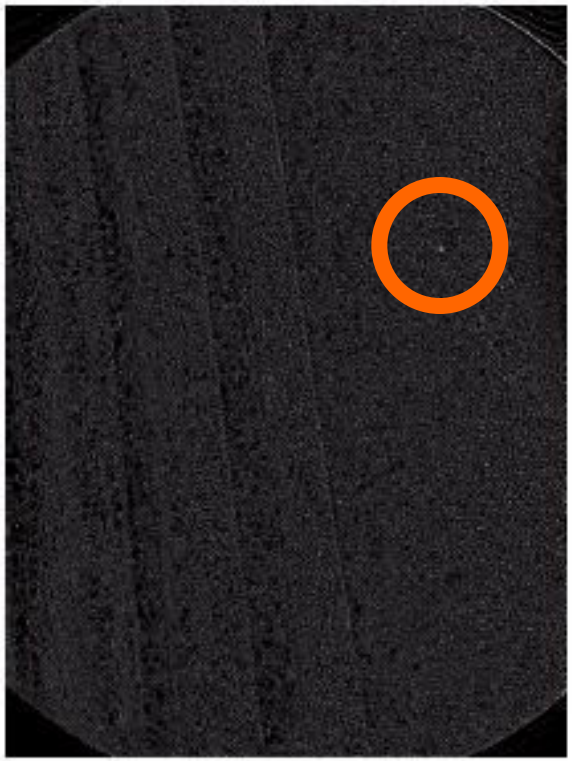
−1	−1	−1
−1	8	−1
−1	−1	−1

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

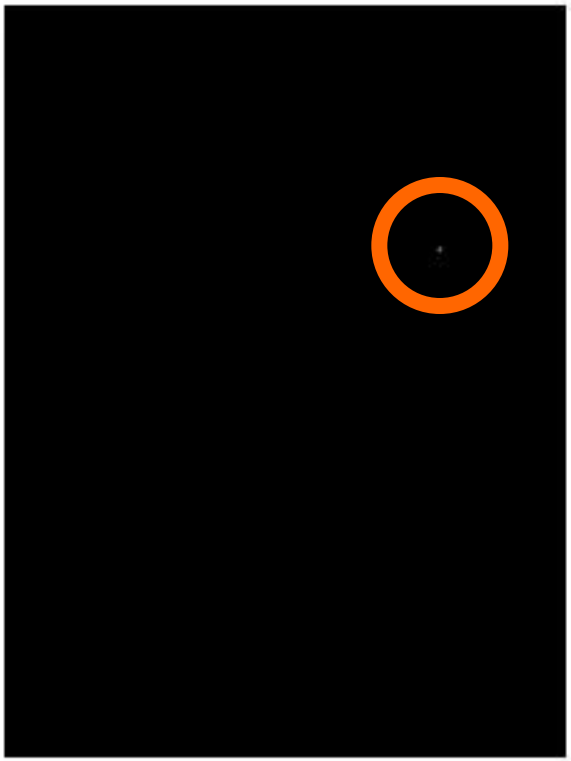
Point Detection (cont…)



X-ray image of a turbine blade



Result of point detection



Result of thresholding

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

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

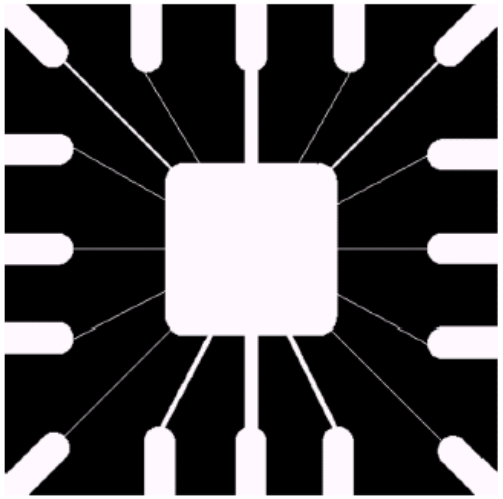
Vertical

2	−1	−1
−1	2	−1
−1	−1	2

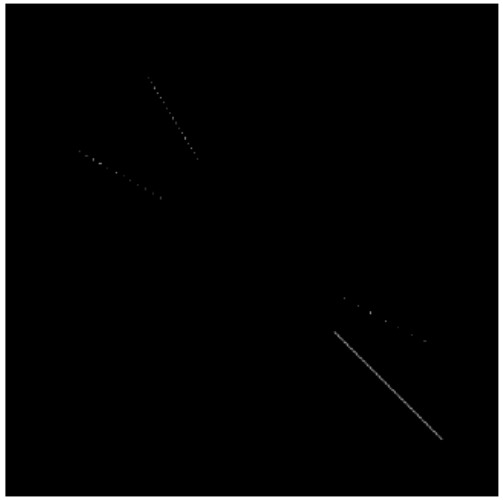
−45°

Line Detection (cont…)

Binary image of a wire bond mask



After processing with -45° line detector



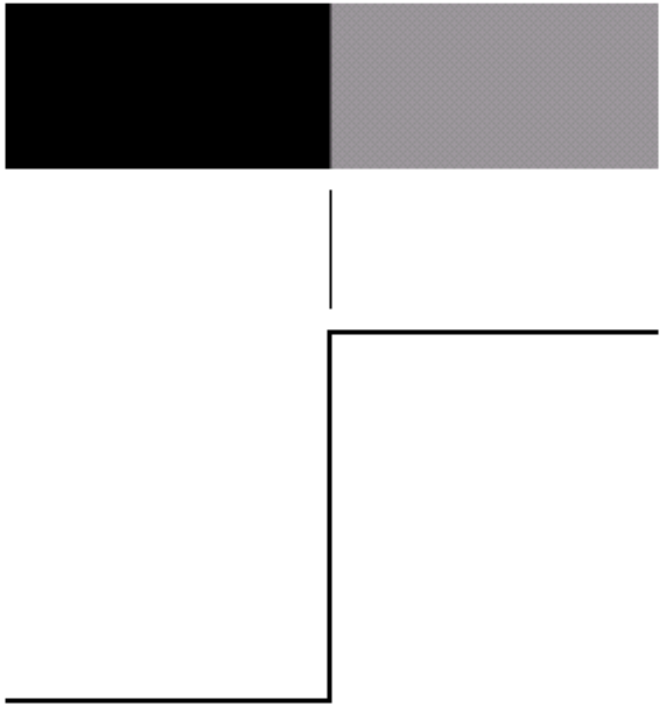
Result of thresholding filtering result

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

Edge Detection

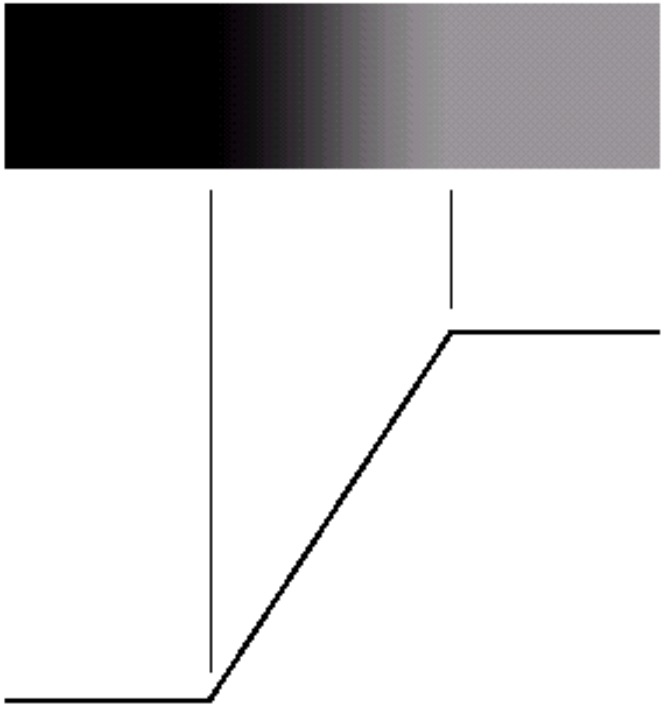
An edge is a set of connected pixels that lie on the boundary between two regions

Model of an ideal digital edge



Gray-level profile of a horizontal line through the image

Model of a ramp digital edge



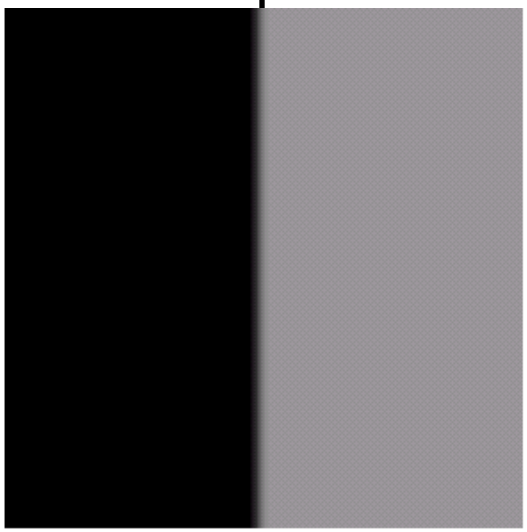
Gray-level profile of a horizontal line through the image

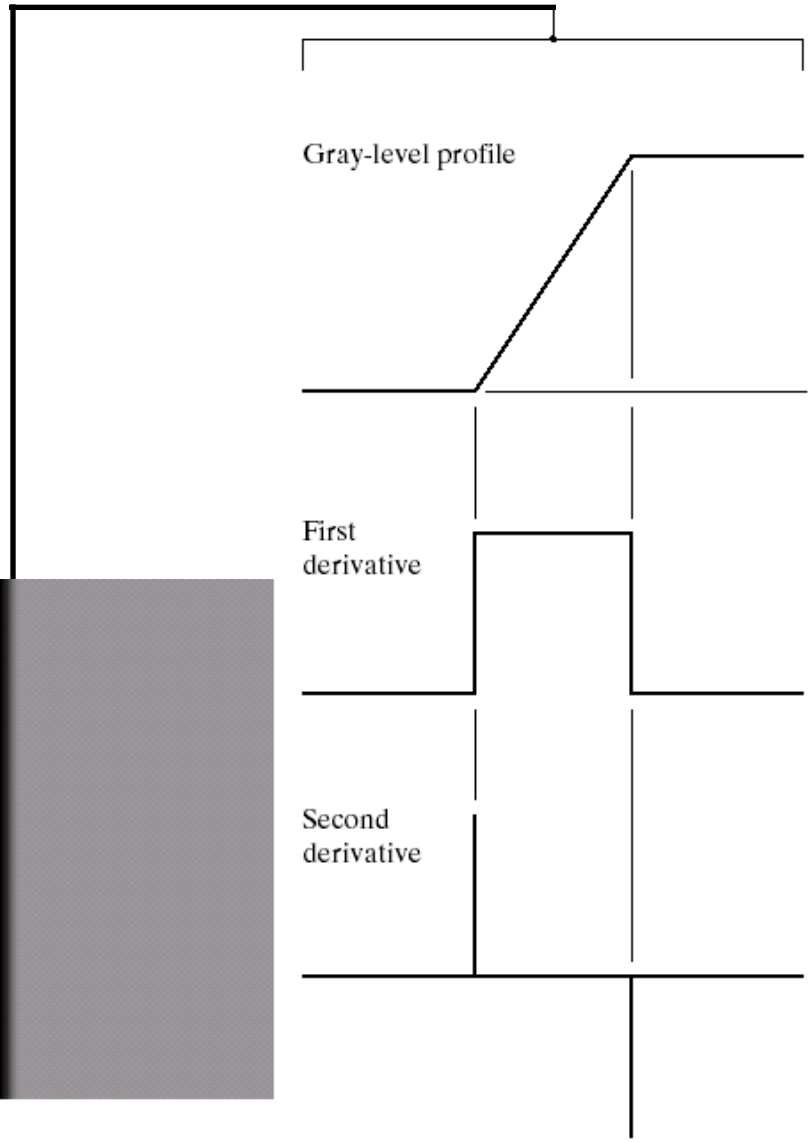
Edges & Derivatives

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



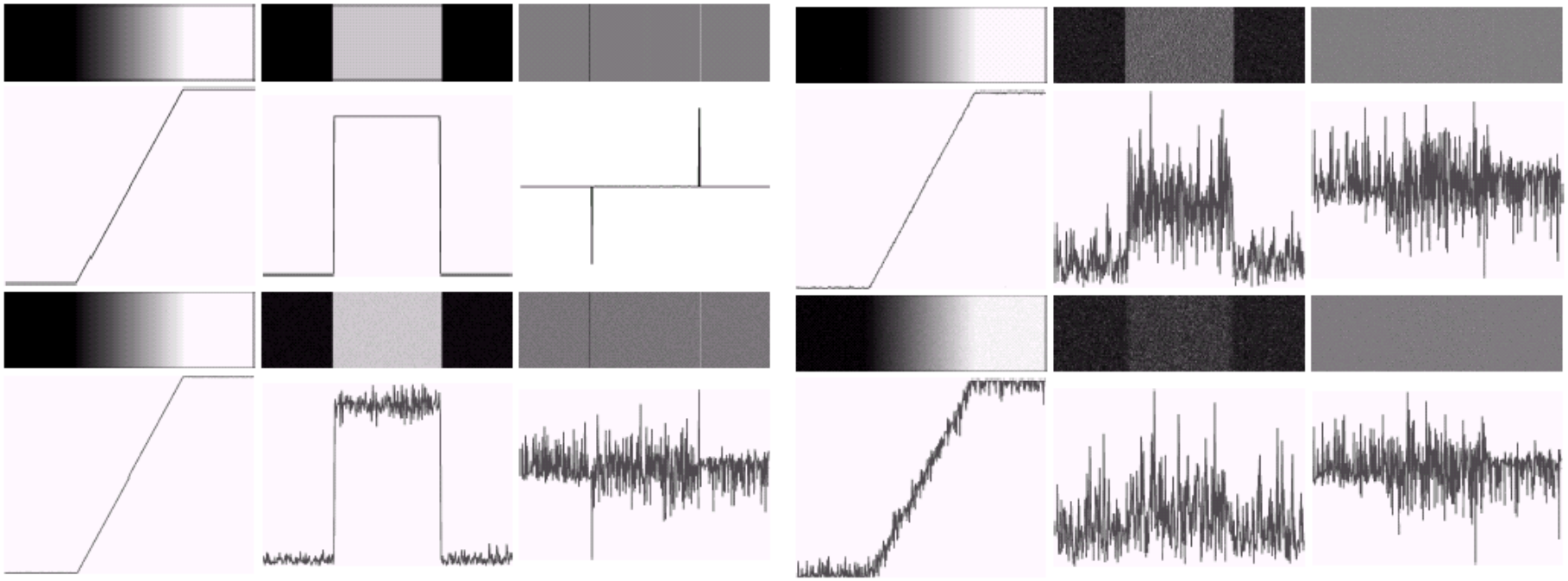


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

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

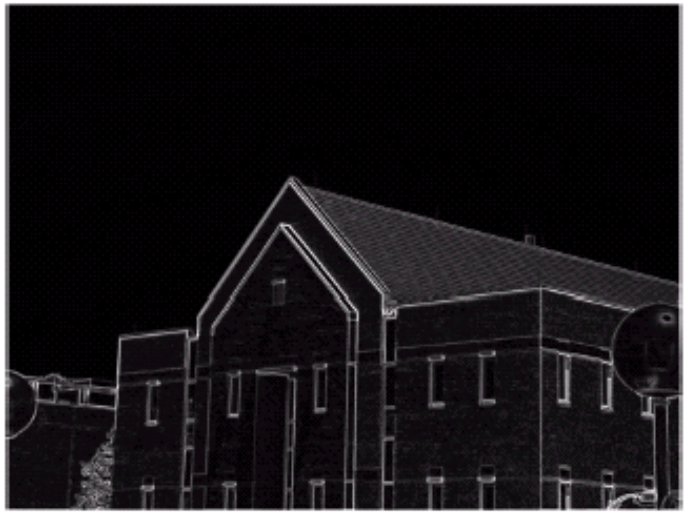
One way to overcome this is to smooth images prior to edge detection

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

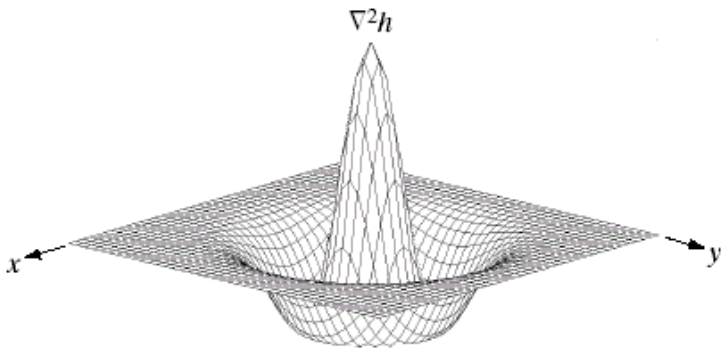
−1	−1	−1
−1	8	−1
−1	−1	−1

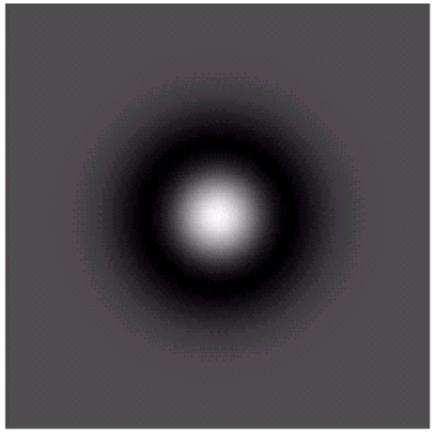
The Laplacian is typically not used by itself as it is too sensitive to noise

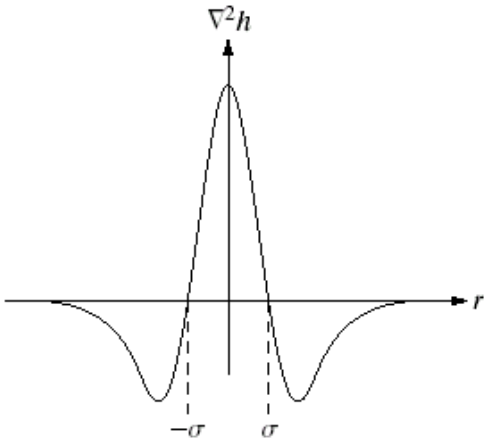
Usually hen used for edge detection the Laplacian is combined with a smoothing Gaussian filter

Laplacian Of Gaussian

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

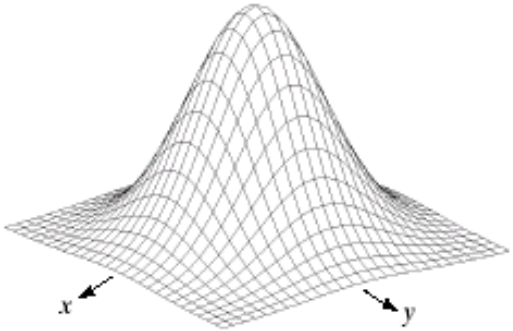






0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

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

Digital Image Processing

Lecture - 10

Image Segmentation: Thresholding

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

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

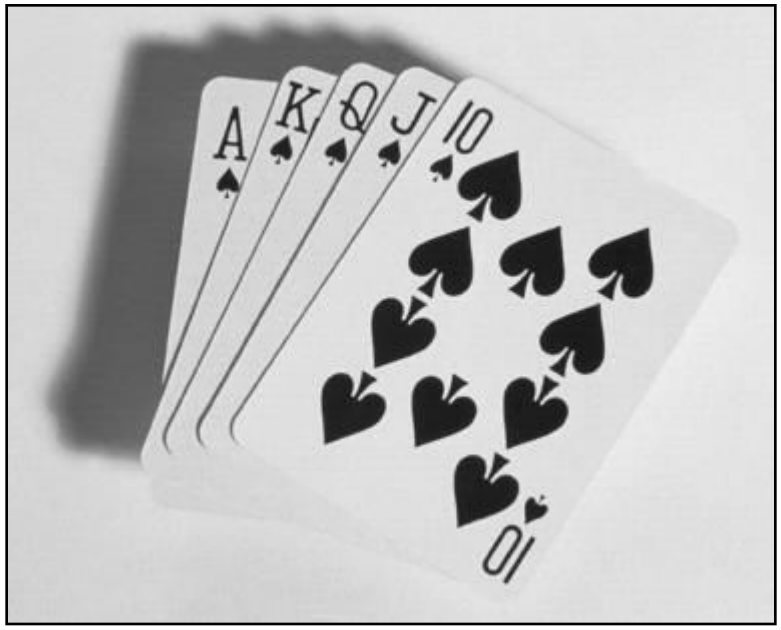
We have talked about simple single value thresholding already

Single value thresholding can be given mathematically as follows:

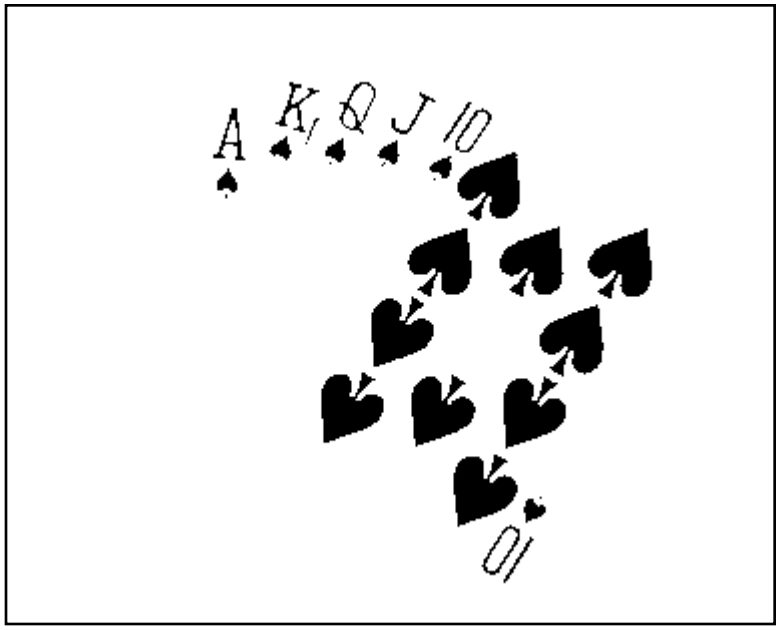
$$g(x,y) = \begin{cases} 1 \; if \;\; f(x,y) > T \\ 0 \; if \;\; f(x,y) \leq T \end{cases}$$

Thresholding Example

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



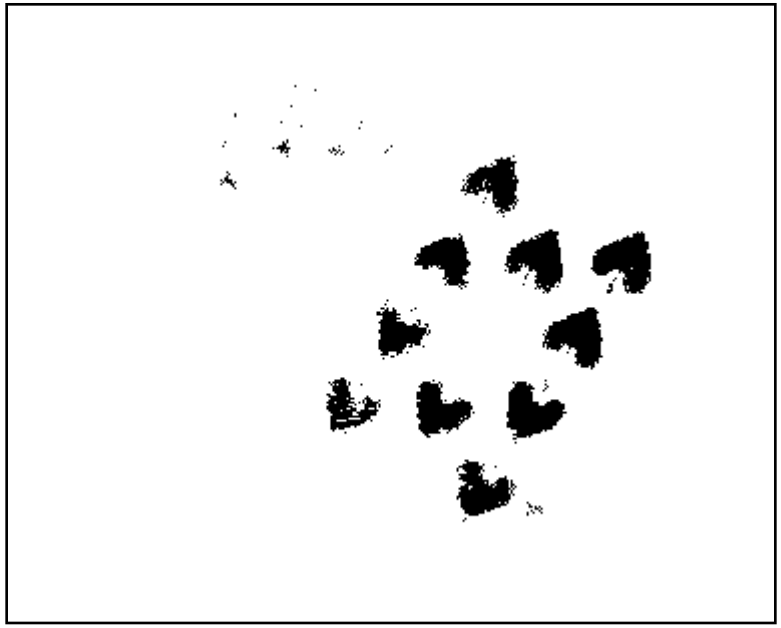
Original Image



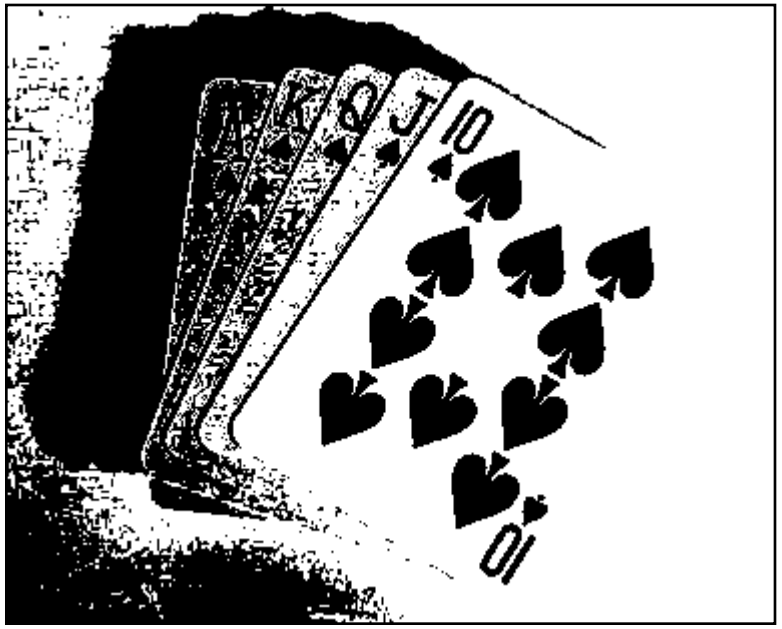
Thresholded Image

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 μ_1 and G_2 to give μ_2

Basic Global Thresholding Algorithm

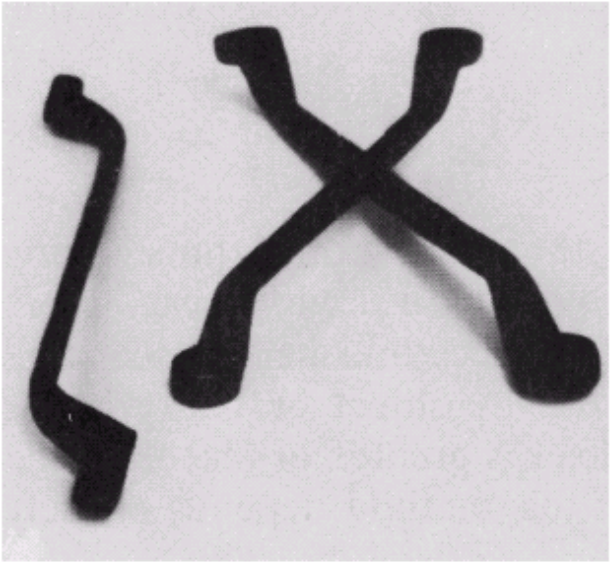
4. Compute a new threshold value:

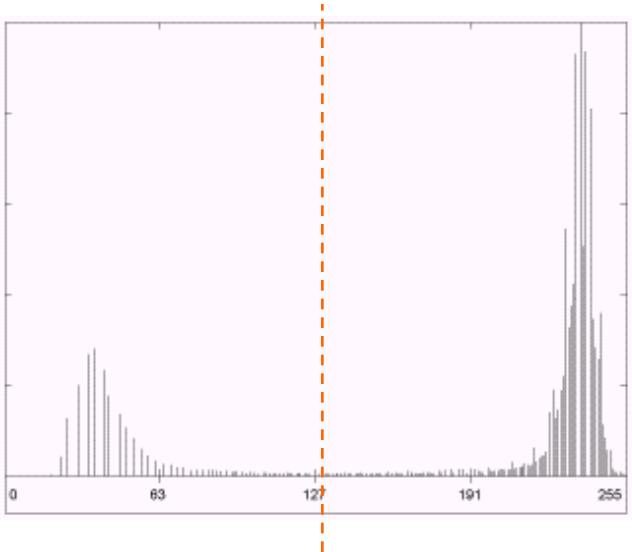
$$T = \frac{\mu_1 + \mu_2}{2}$$

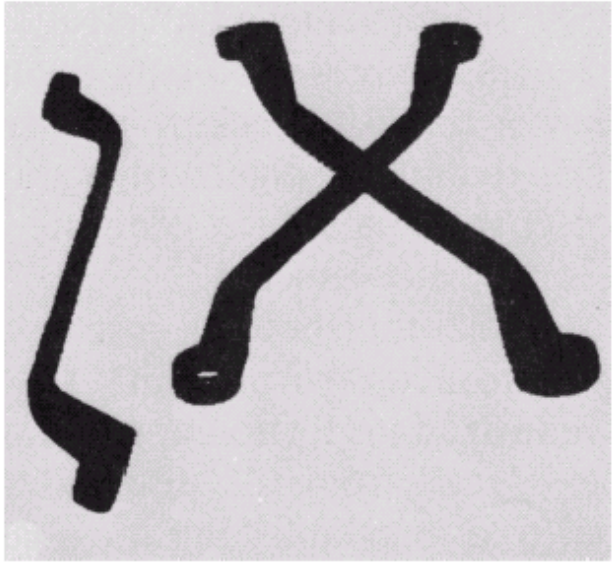
5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit T_∞

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

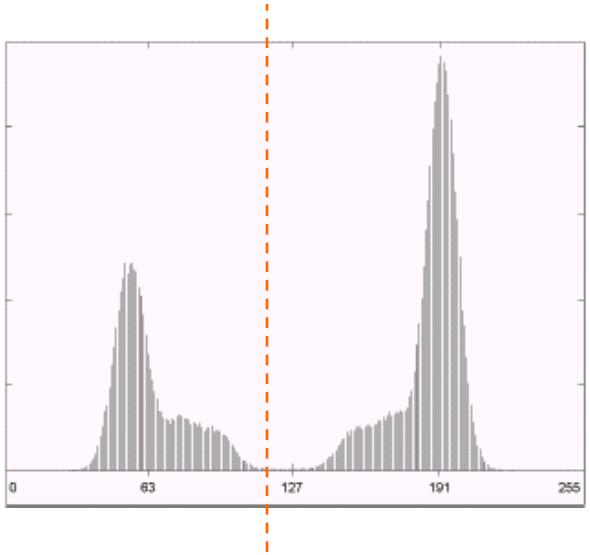
Thresholding Example 1







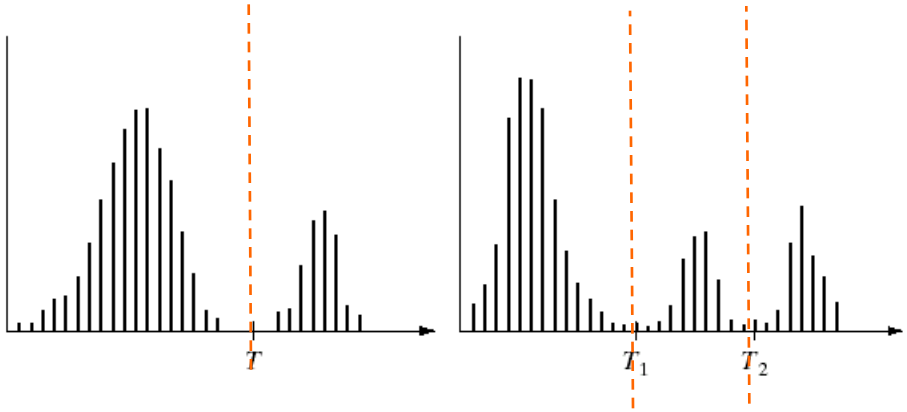
Thresholding Example 2



Problems With Single Value Thresholding

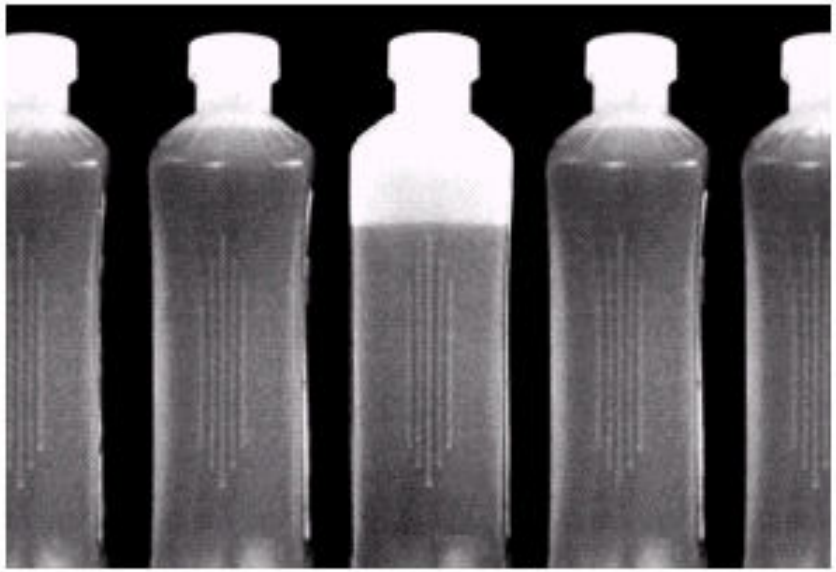
Single value thresholding only works for bimodal histograms

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



Problems With Single Value Thresholding (cont…)

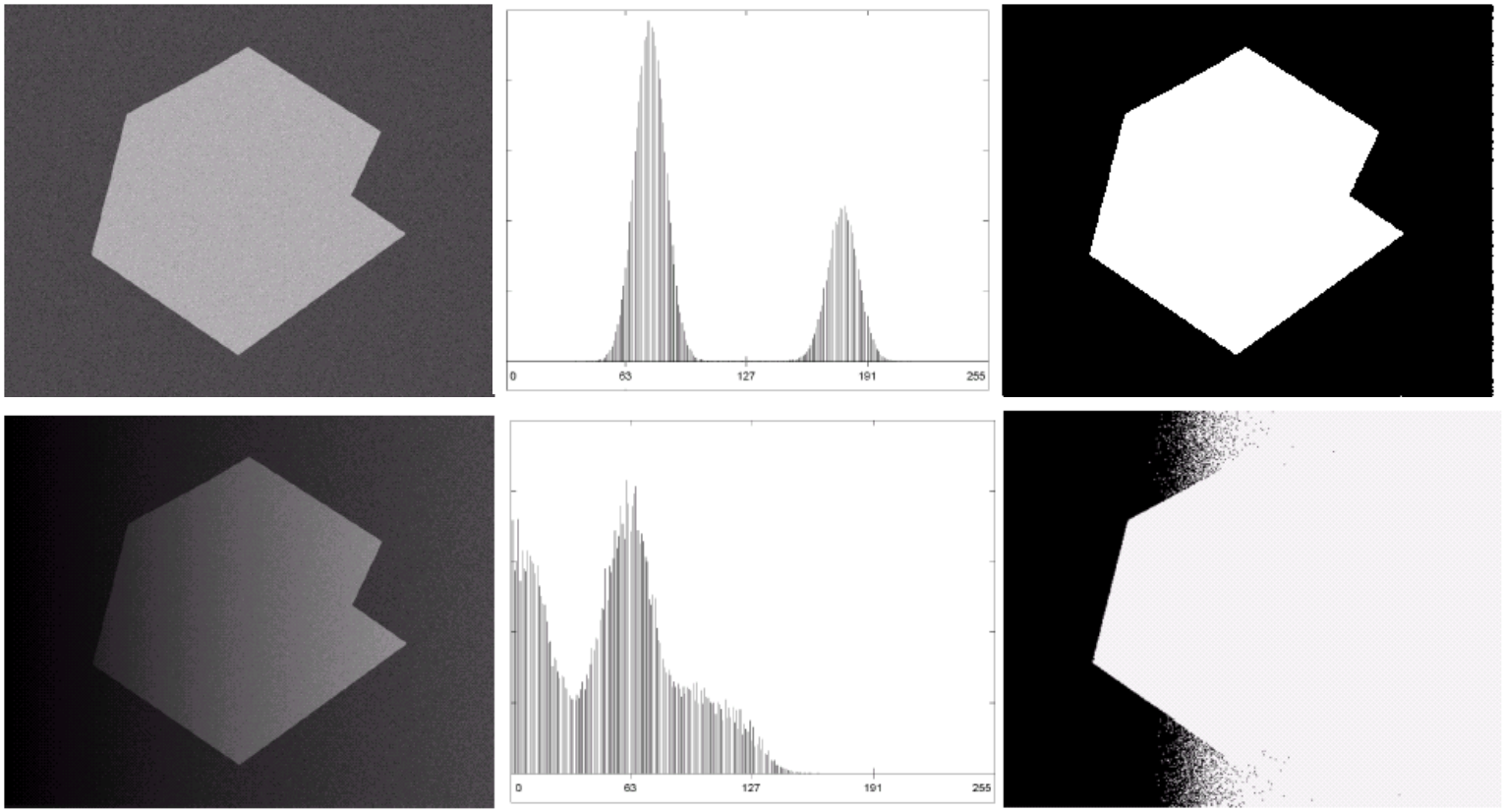
Let's say we want to isolate the contents of the bottles



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

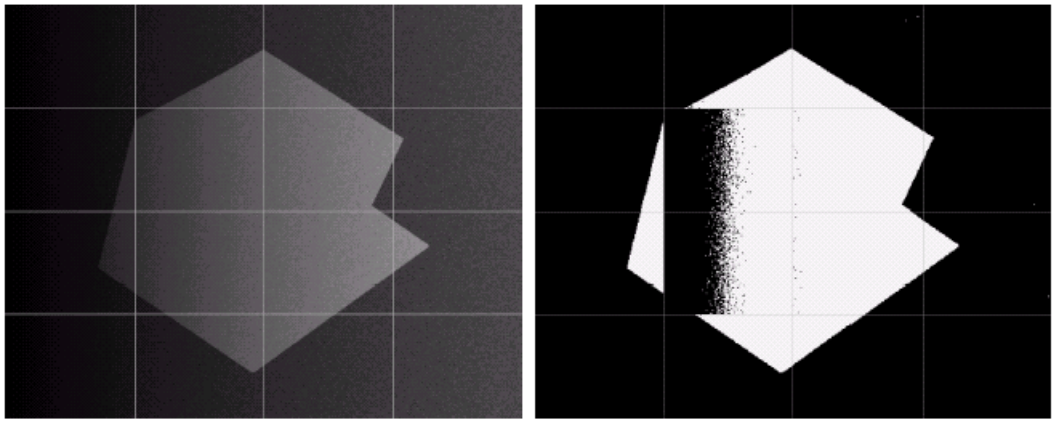
Basic Adaptive Thresholding

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



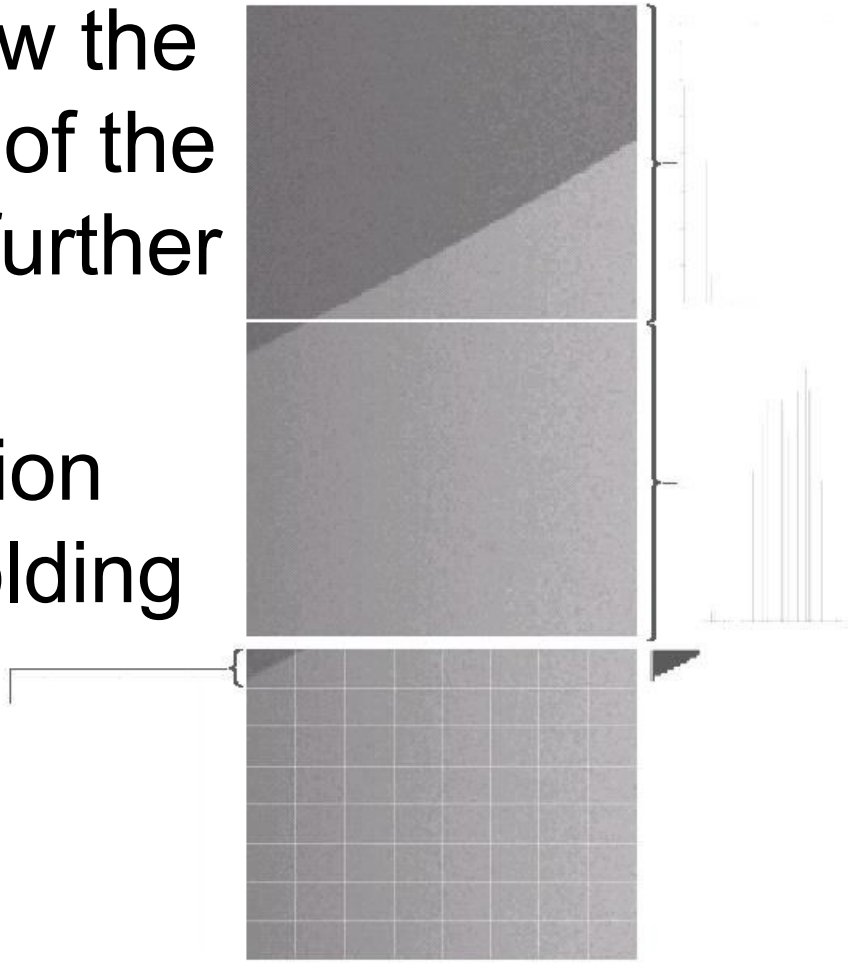
As can be seen success is mixed

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

Digital Image Processing

Lecture - 11

Morphological Image Processing

Md. Khademul Islam Molla, PhD
Professor of CSE
University of Rajshahi
khademul.cse@ru.ac.bd

Contents

Once segmentation is complete, morphological operations can be used to remove imperfections in the segmented image and provide information on the form and structure of the image

In this lecture we will consider

- What is morphology?
- Simple morphological operations
- Compound operations
- Morphological algorithms

1, 0, Black, White?

Throughout all of the following slides whether 0 and 1 refer to white or black is a little interchangeable

All of the discussion that follows assumes segmentation has already taken place and that images are made up of 0s for background pixels and 1s for object pixels

After this it doesn’t matter if 0 is black, white, yellow, green…….

What Is Morphology?

Morphological image processing (or *morphology*) describes a range of image processing techniques that deal with the shape (or morphology) of features in an image

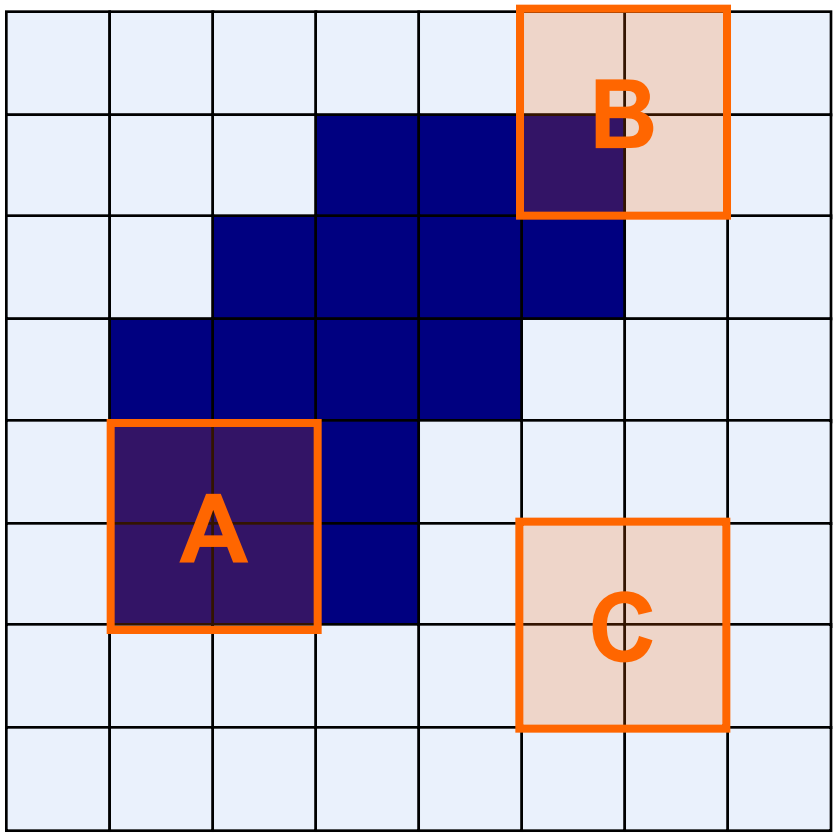
Morphological operations are typically applied to remove imperfections introduced during segmentation, and so typically operate on bi-level images

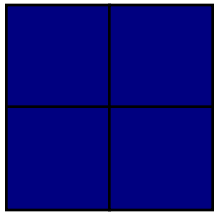
Quick Example

Image after segmentation

Image after segmentation and morphological processing

Structuring Elements, Hits & Fits





Structuring Element

Fit: All *on pixels* in the structuring element cover *on pixels* in the image

Hit: Any *on pixel* in the structuring element covers an *on pixel* in the image

All morphological processing operations are based on these simple ideas

Structuring Elements

Structuring elements can be any size and make any shape

However, for simplicity we will use rectangular structuring elements with their origin at the middle pixel

1	1	1
1	**1**	1
1	1	1

0	1	0
1	**1**	1
0	1	0

0	0	1	0	0
0	1	1	1	0
1	1	**1**	1	1
0	1	1	1	0
0	0	1	0	0

Fitting & Hitting

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	B	1	1	1	0	C	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	A	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

Structuring Element 1

0	1	0
1	1	1
0	1	0

Structuring Element 2

Fundamental Operations

Fundamentally morphological image processing is very like spatial filtering

The structuring element is moved across every pixel in the original image to give a pixel in a new processed image

The value of this new pixel depends on the operation performed

There are two basic morphological operations: **erosion** and **dilation**

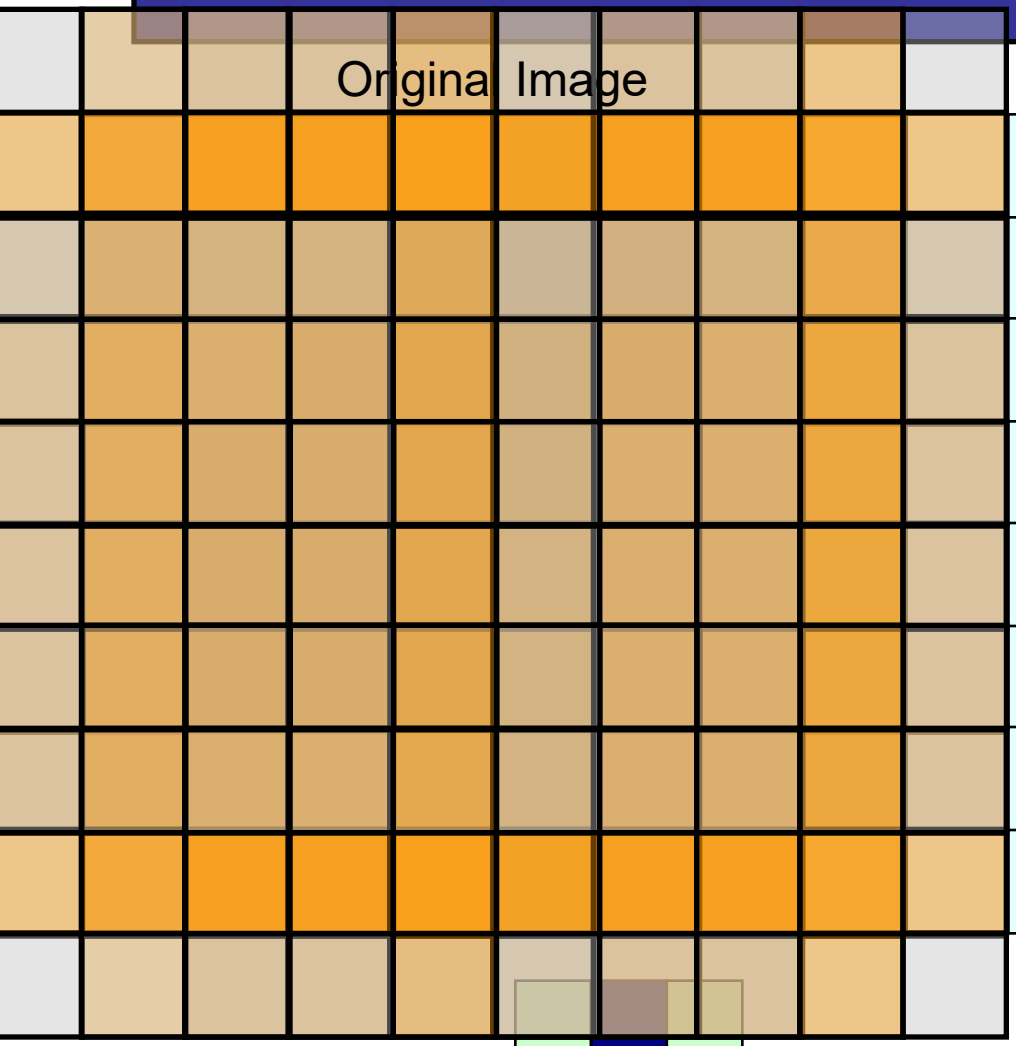
Erosion

Erosion of image f by structuring element s is given by $f \ominus s$

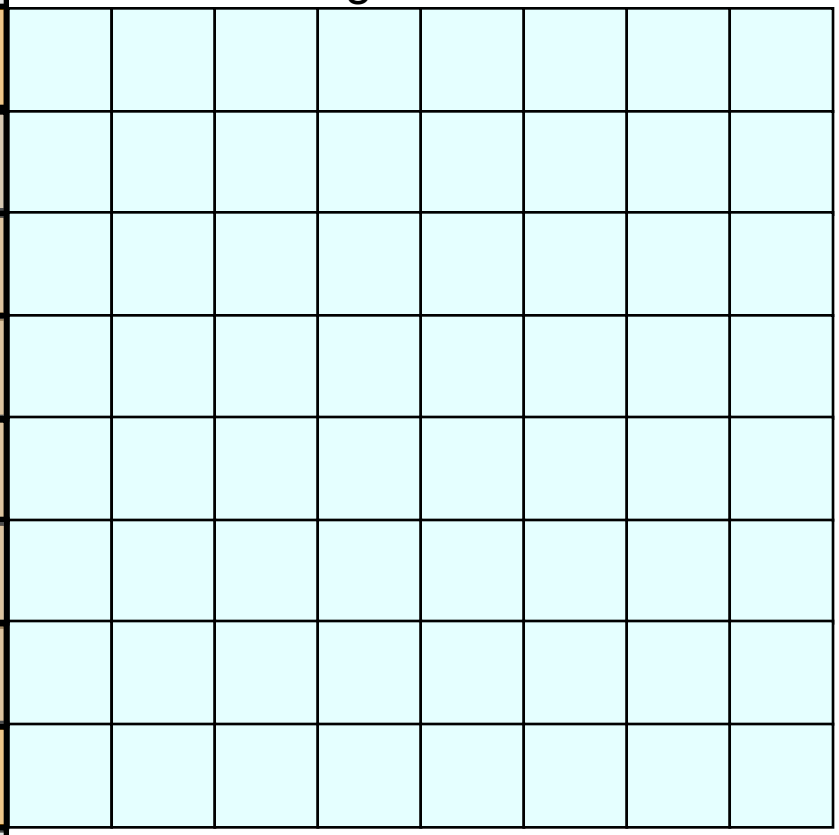
The structuring element s is positioned with its origin at (x, y) and the new pixel value is determined using the rule:

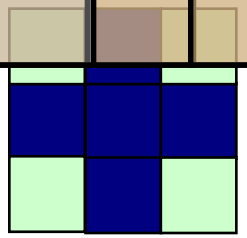
$$g(x, y) = \begin{cases} 1 \text{ if } \ s \text{ fits } \ f \\ 0 \text{ otherwise} \end{cases}$$

Erosion Example



Processed Image With Eroded Pixels

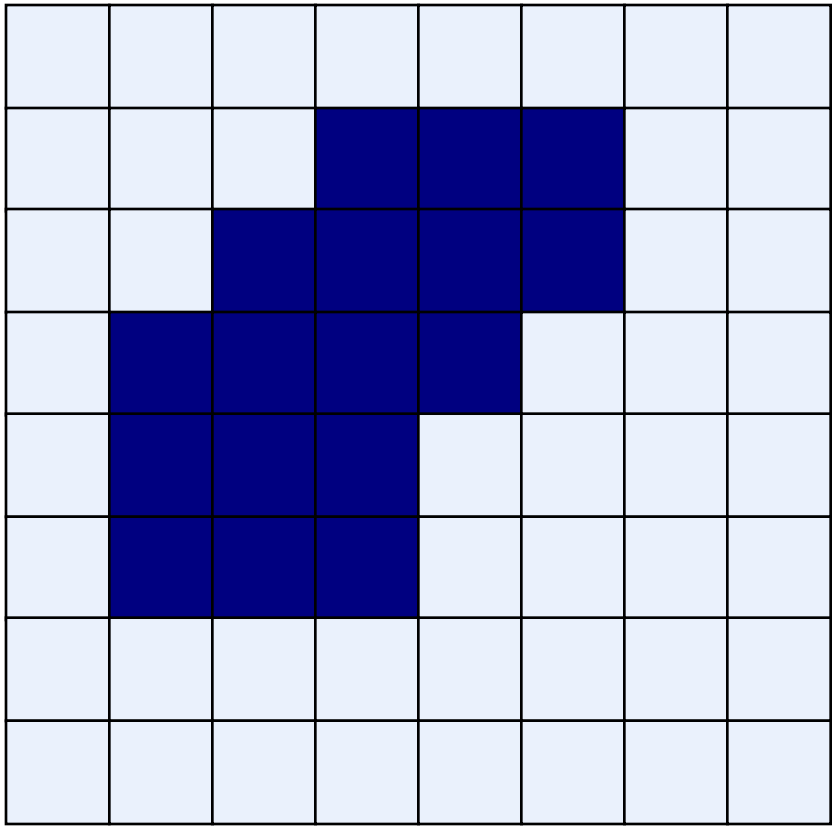




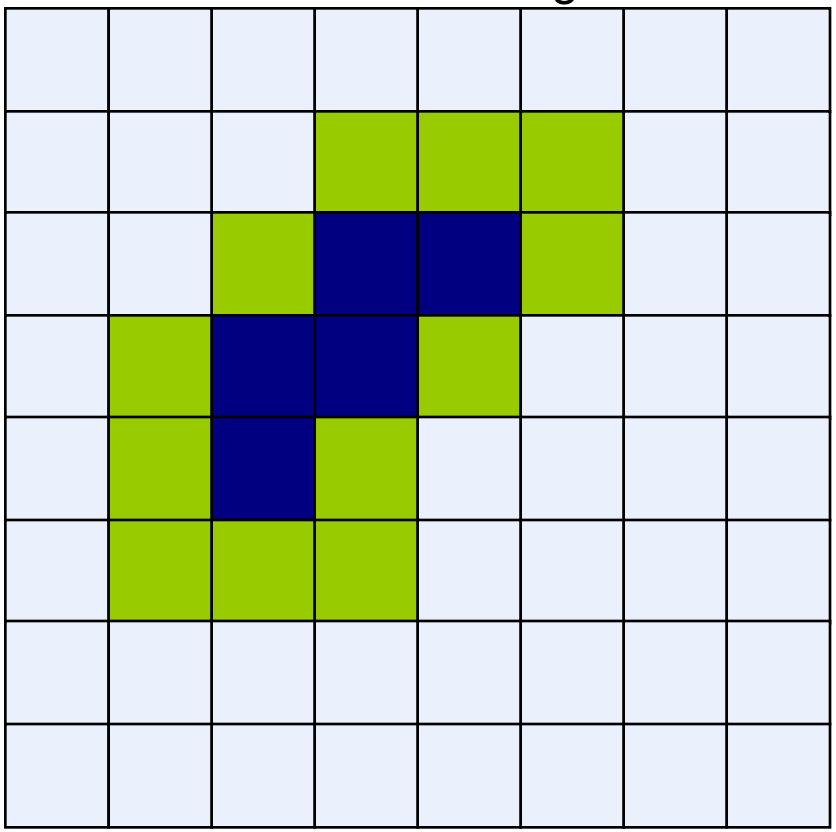
Structuring Element

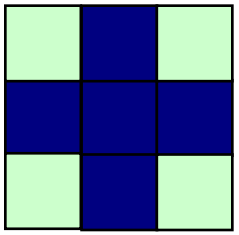
Erosion Example

Original Image



Processed Image

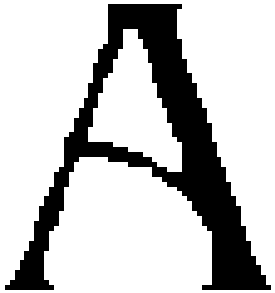




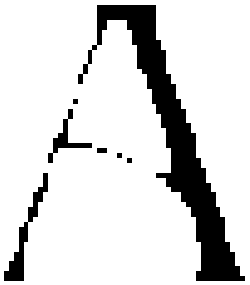
Structuring Element

Erosion Example 1

Original image



Erosion by 3*3 square structuring element

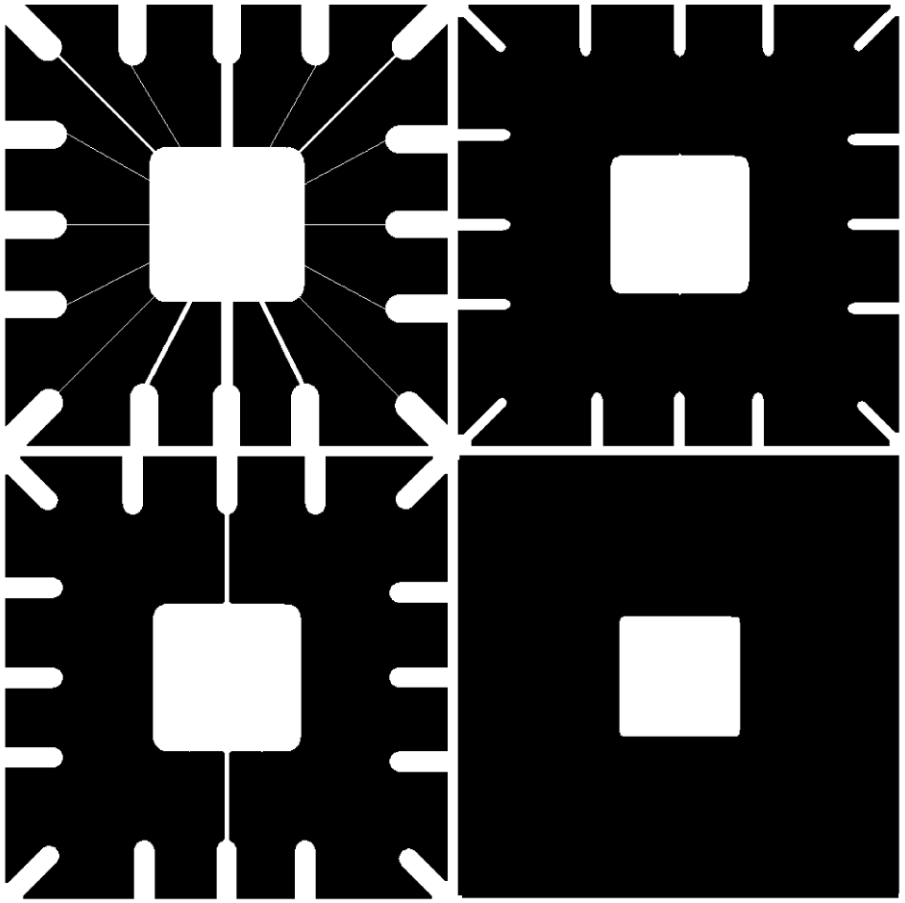


Erosion by 5*5 square structuring element

Watch out: In these examples a 1 refers to a black pixel!

Erosion Example 2

Original image



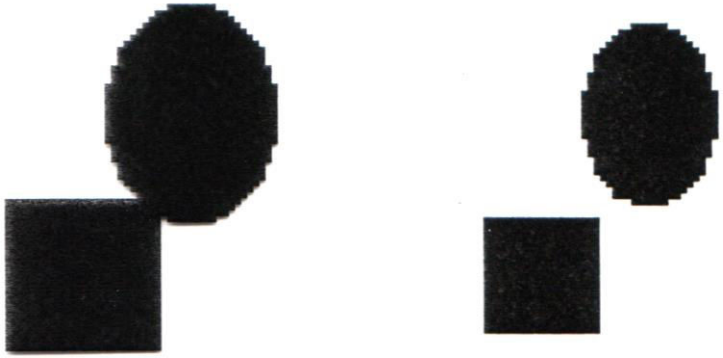
After erosion with a disc of radius 10

After erosion with a disc of radius 5

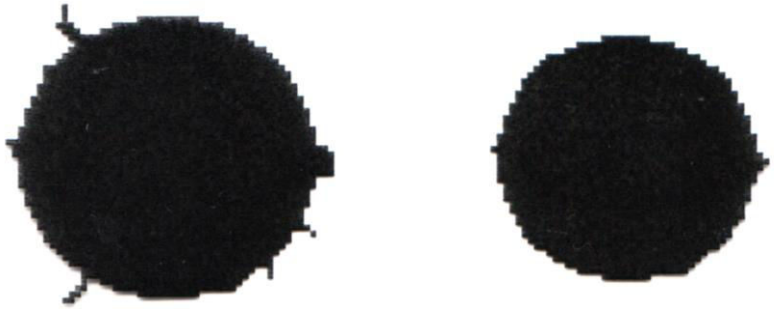
After erosion with a disc of radius 20

What Is Erosion For?

Erosion can split apart joined objects



Erosion can strip away extrusions



Watch out: Erosion shrinks objects

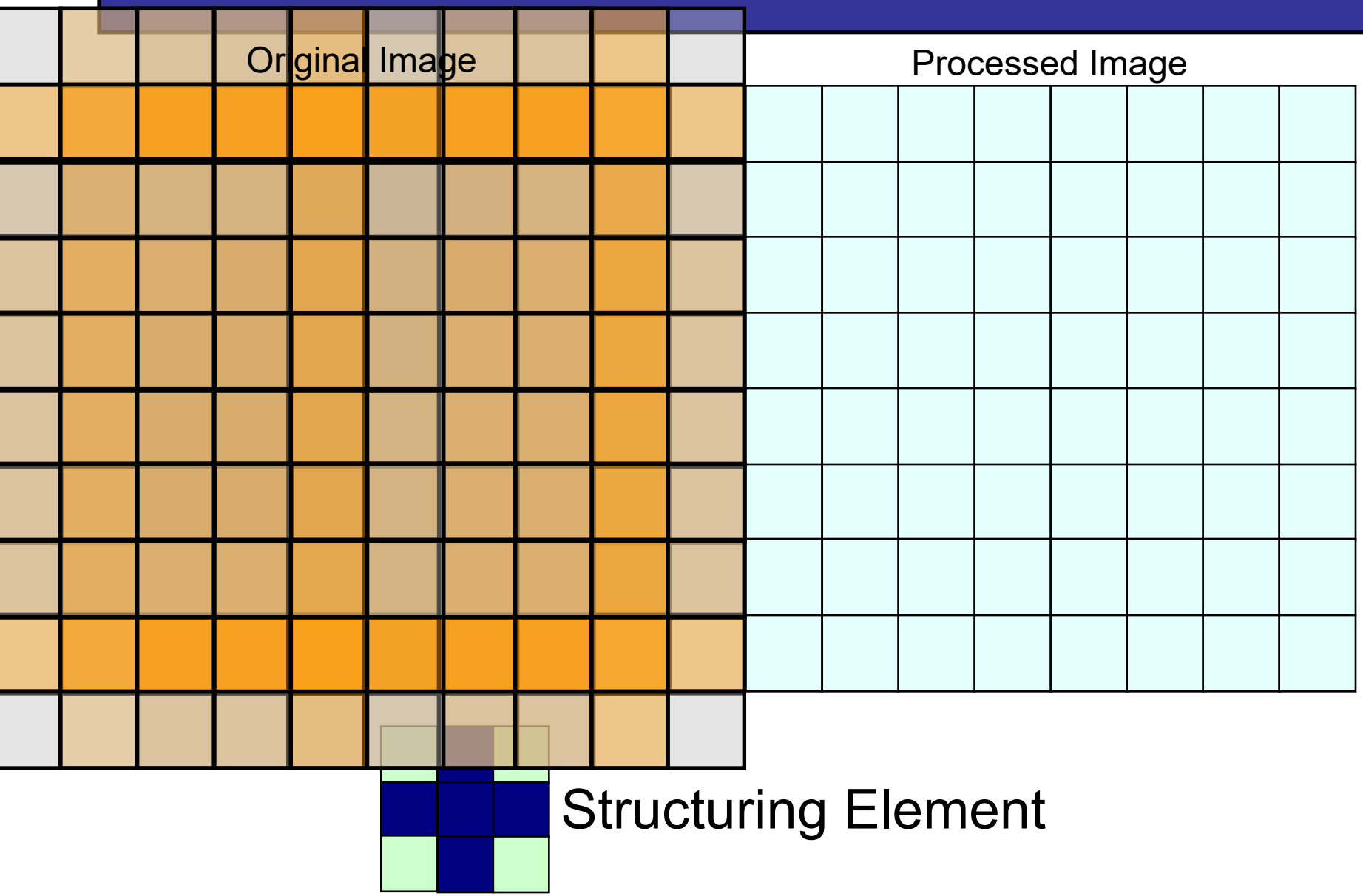
Dilation

Dilation of image f by structuring element s is given by $f \oplus s$

The structuring element s is positioned with its origin at (x, y) and the new pixel value is determined using the rule:

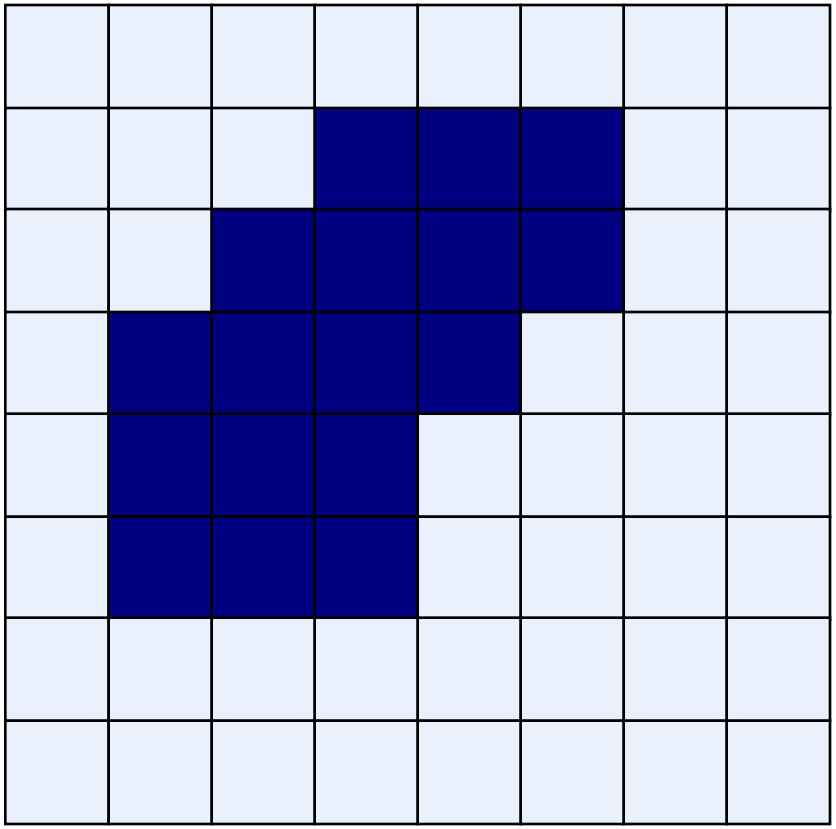
$$g(x, y) = \begin{cases} 1 \text{ if } \ s \text{ hits } \ f \\ 0 \text{ otherwise} \end{cases}$$

Dilation Example

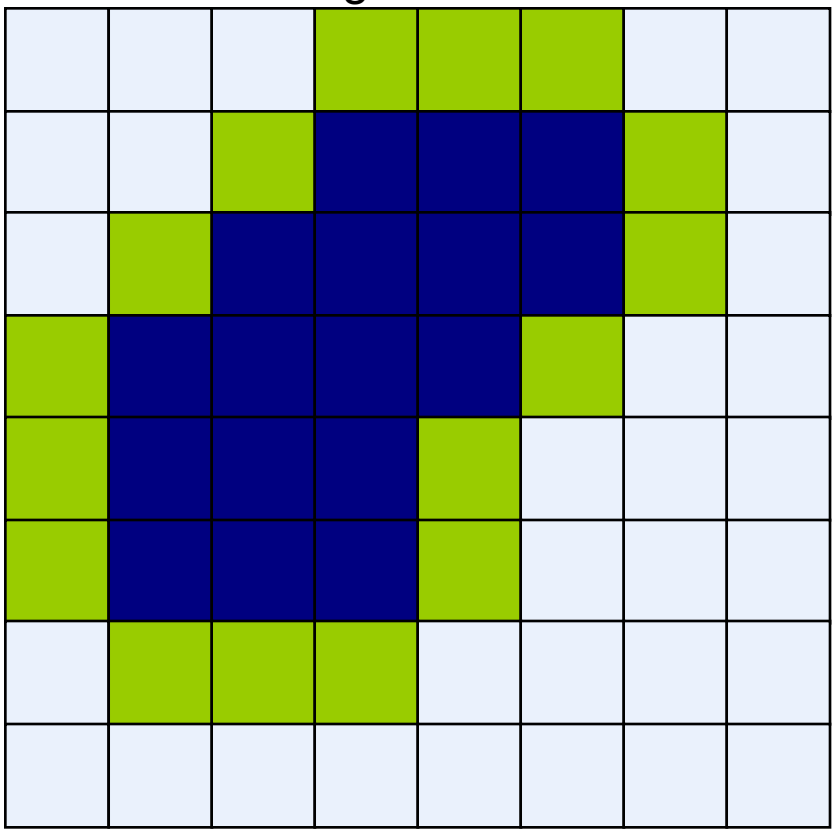


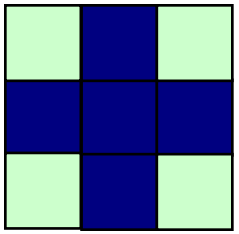
Dilation Example

Original Image



Processed Image With Dilated Pixels





Structuring Element

Dilation Example 1

Original image

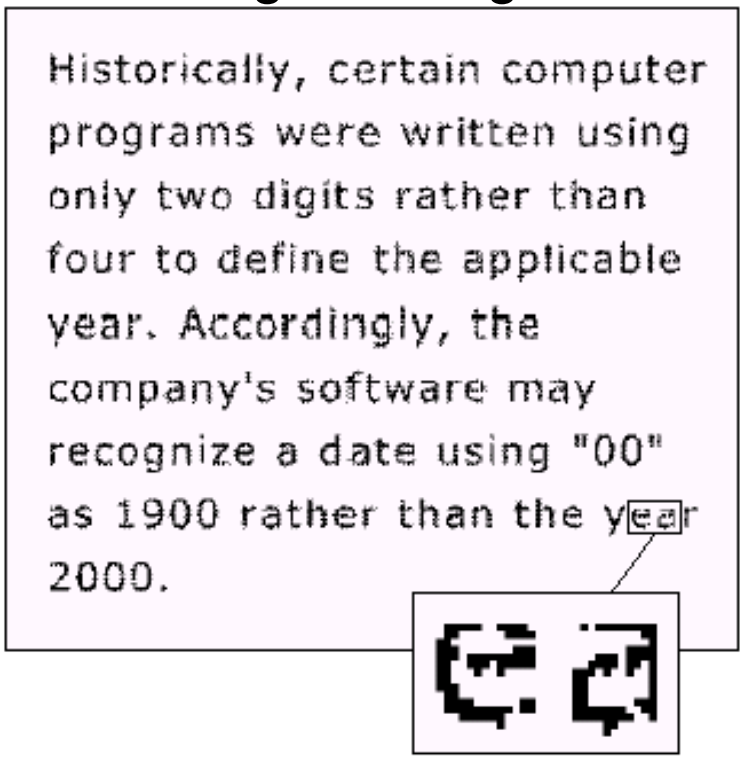
Dilation by 3*3 square structuring element

Dilation by 5*5 square structuring element

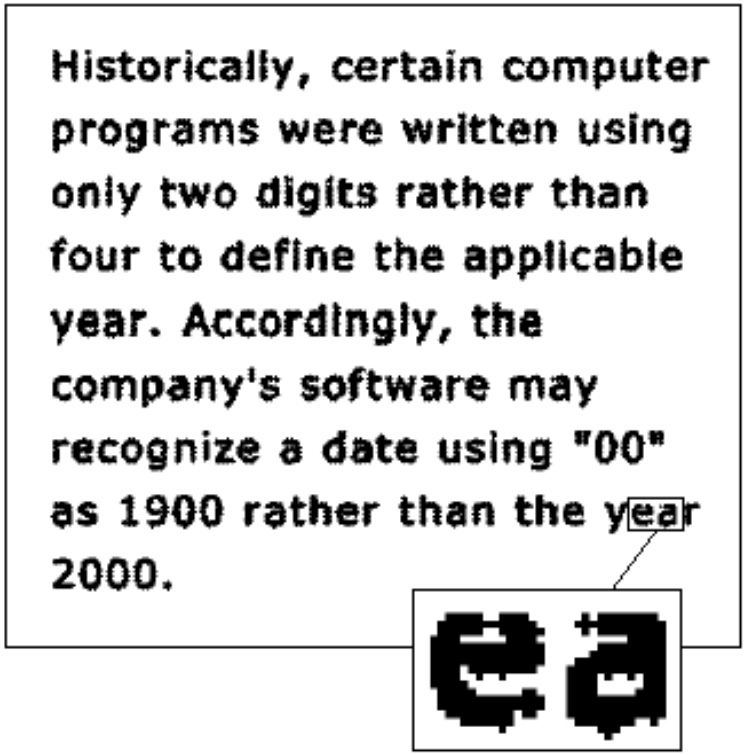
Watch out: In these examples a 1 refers to a black pixel!

Dilation Example 2

Original image



After dilation

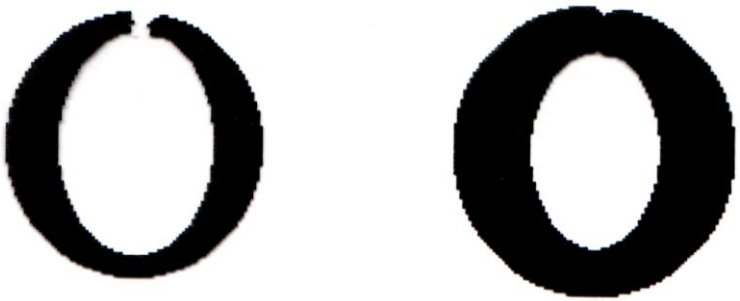


0	1	0
1	1	1
0	1	0

Structuring element

What Is Dilation For?

Dilation can repair breaks



Dilation can repair intrusions

Watch out: Dilation enlarges objects

Compound Operations

More interesting morphological operations can be performed by performing combinations of erosions and dilations

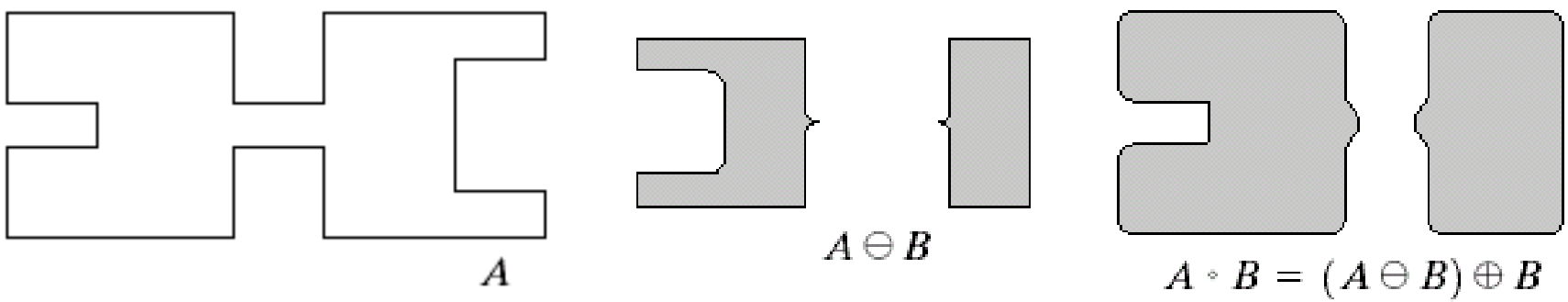
The most widely used of these *compound operations* are:

- Opening
- Closing

Opening

The opening of image f by structuring element s, denoted $f \circ s$ is simply an erosion followed by a dilation

$$f \circ s = (f \ominus s) \oplus s$$



Original shape

After erosion

After dilation (opening)

Note a disc shaped structuring element is used

Opening Example

Original Image

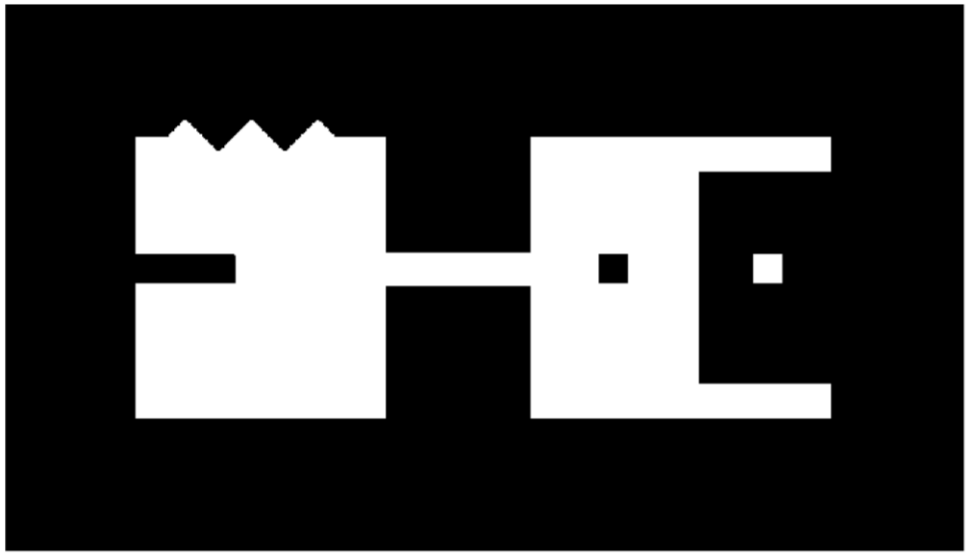
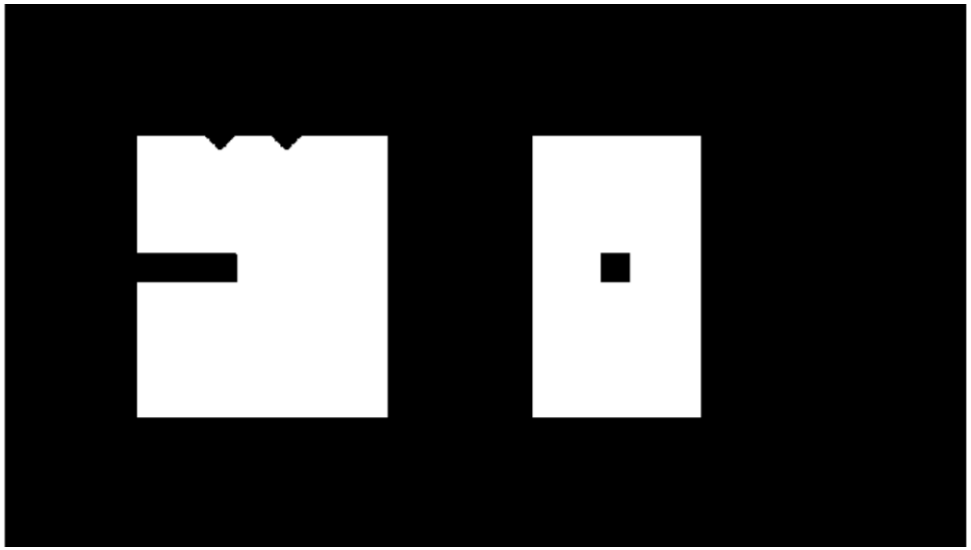
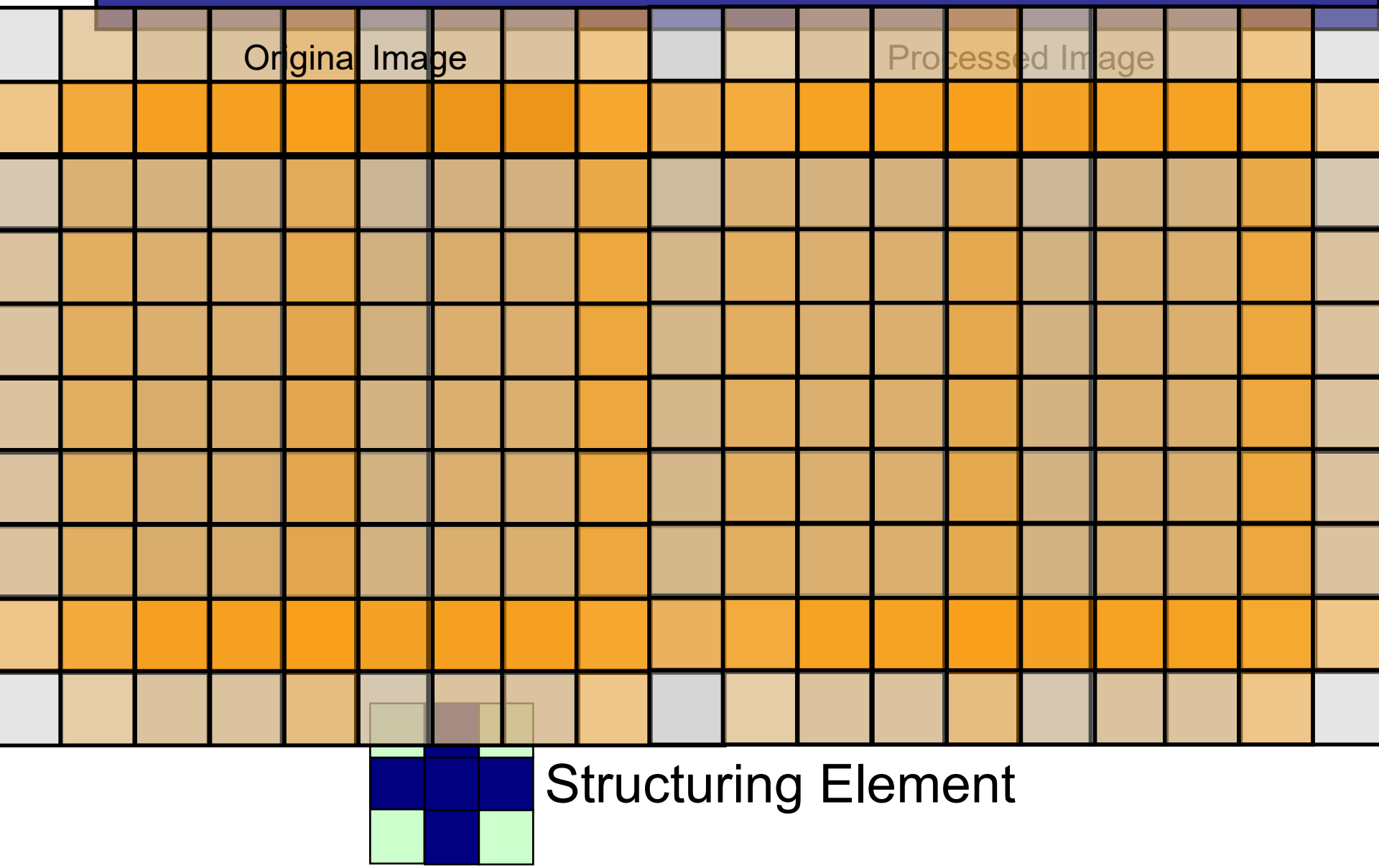


Image After Opening

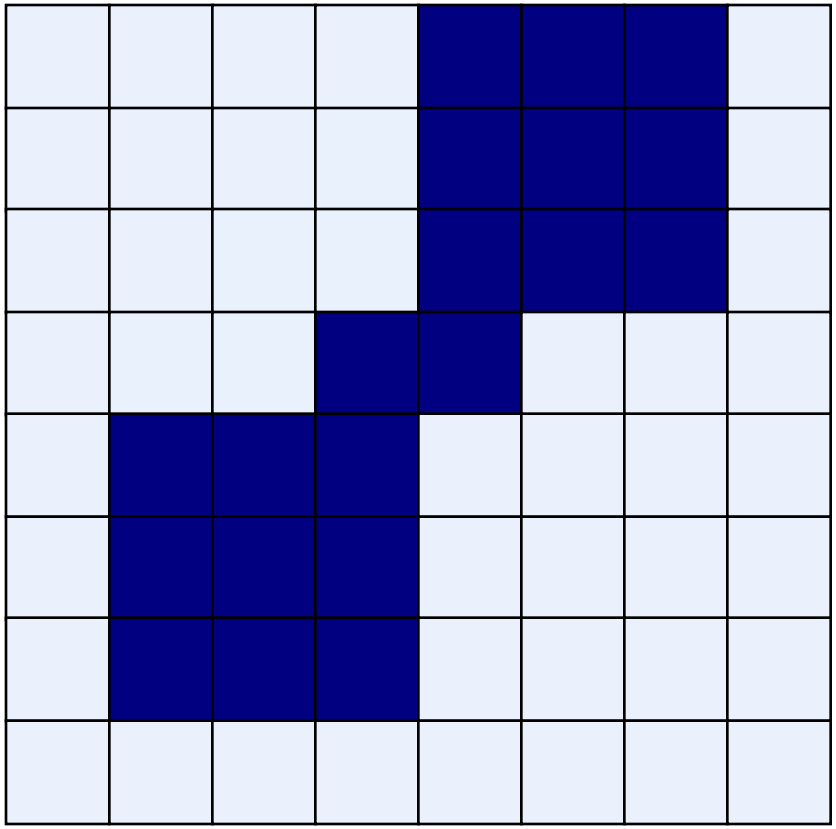


Opening Example

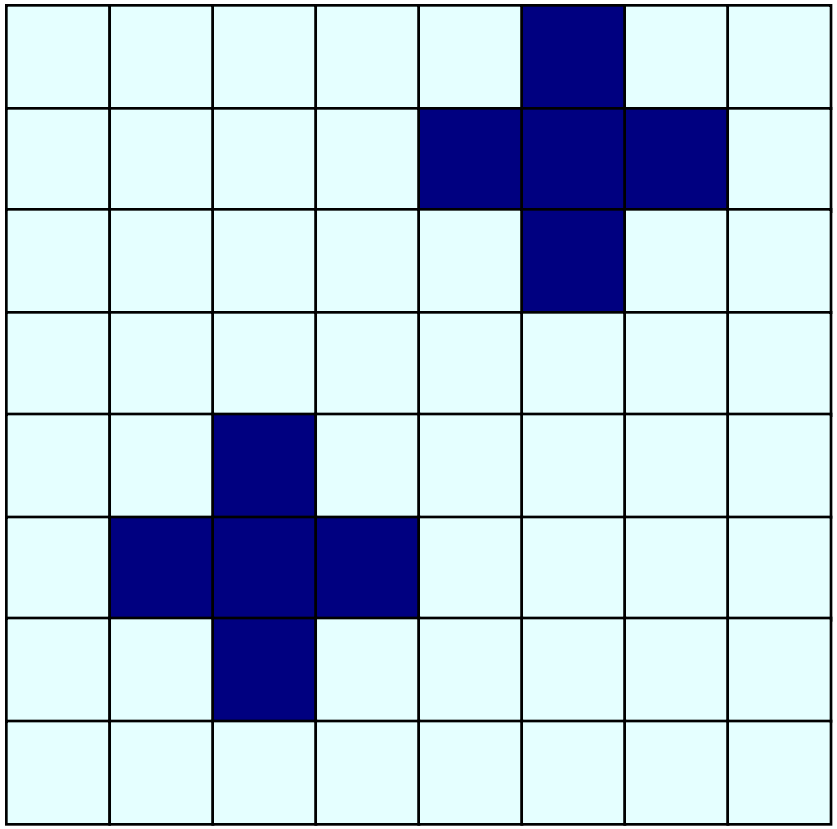


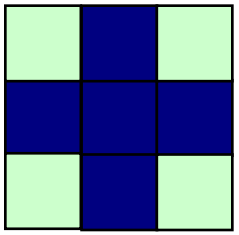
Opening Example

Original Image



Processed Image



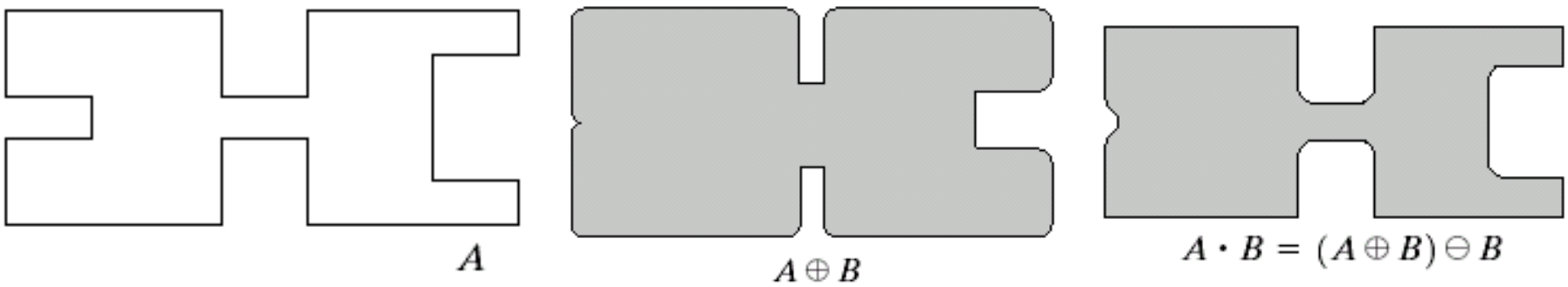


Structuring Element

Closing

The closing of image f by structuring element s, denoted $f \bullet s$ is simply a dilation followed by an erosion

$$f \bullet s = (f \oplus s) \ominus s$$



Original shape

After dilation

After erosion (closing)

Note a disc shaped structuring element is used

Closing Example

Original Image

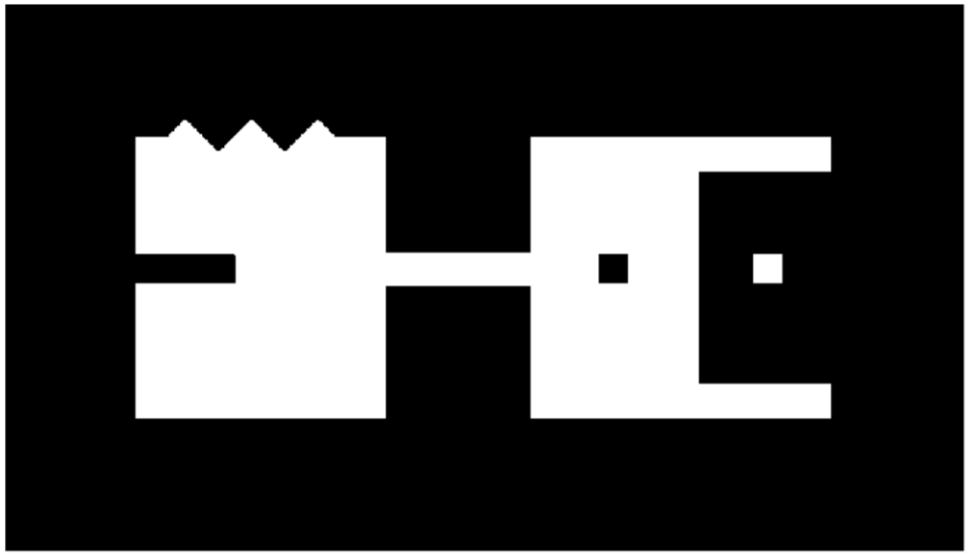
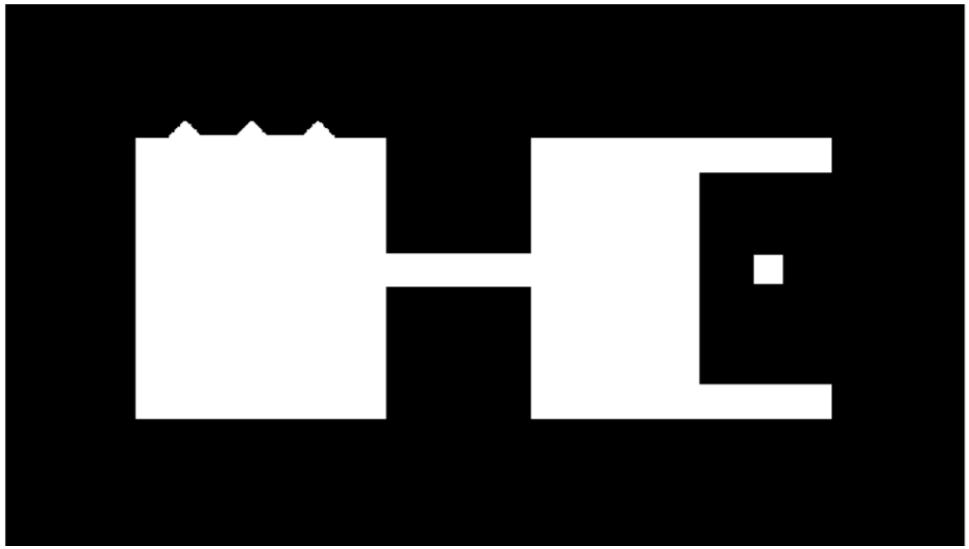
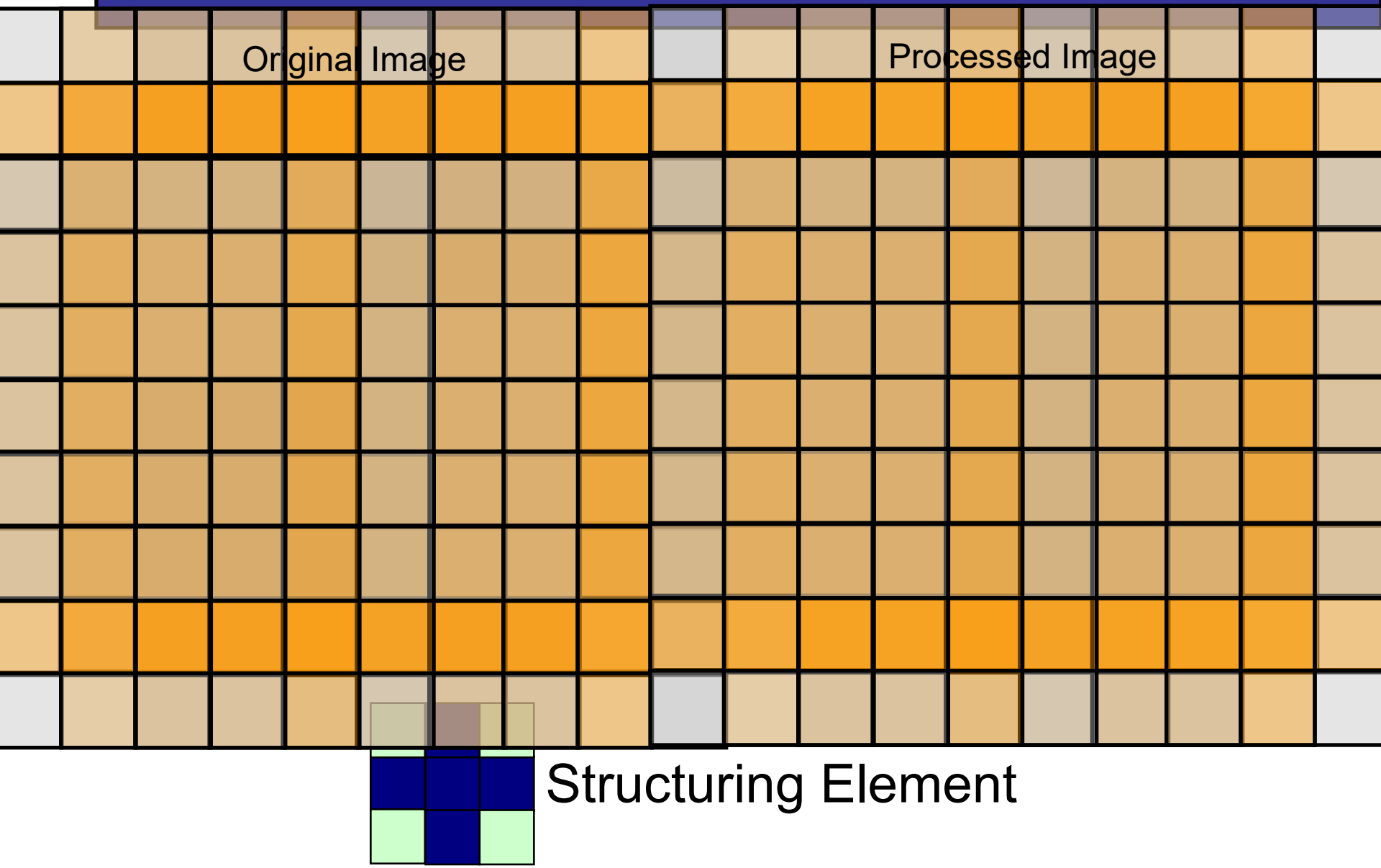


Image After Closing

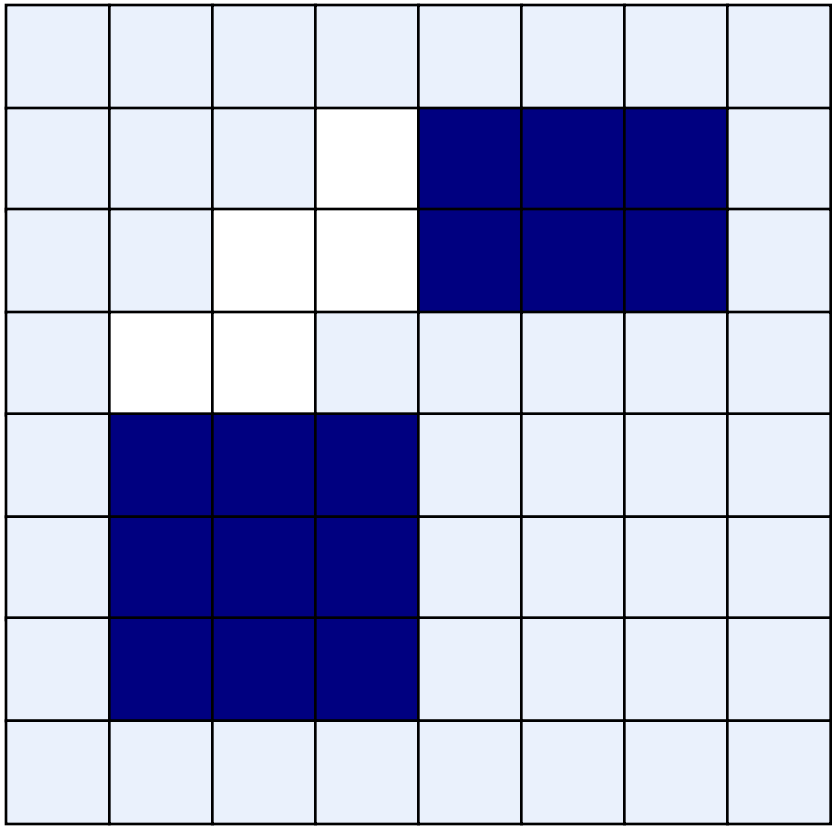


Closing Example

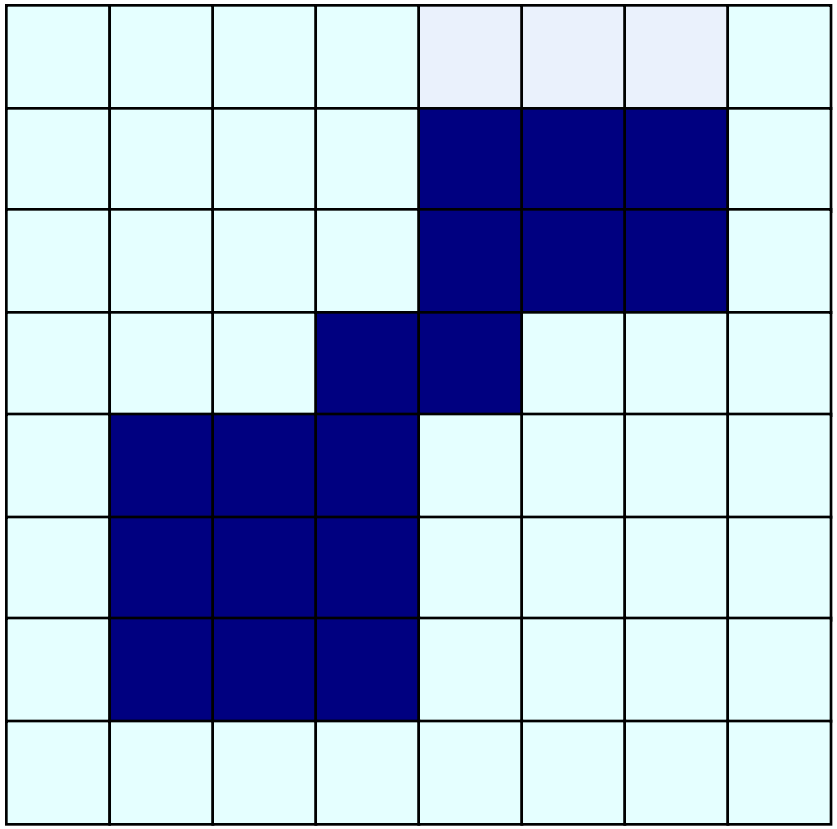


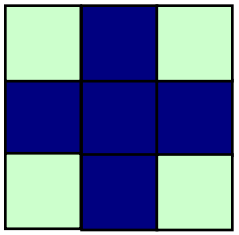
Closing Example

Original Image



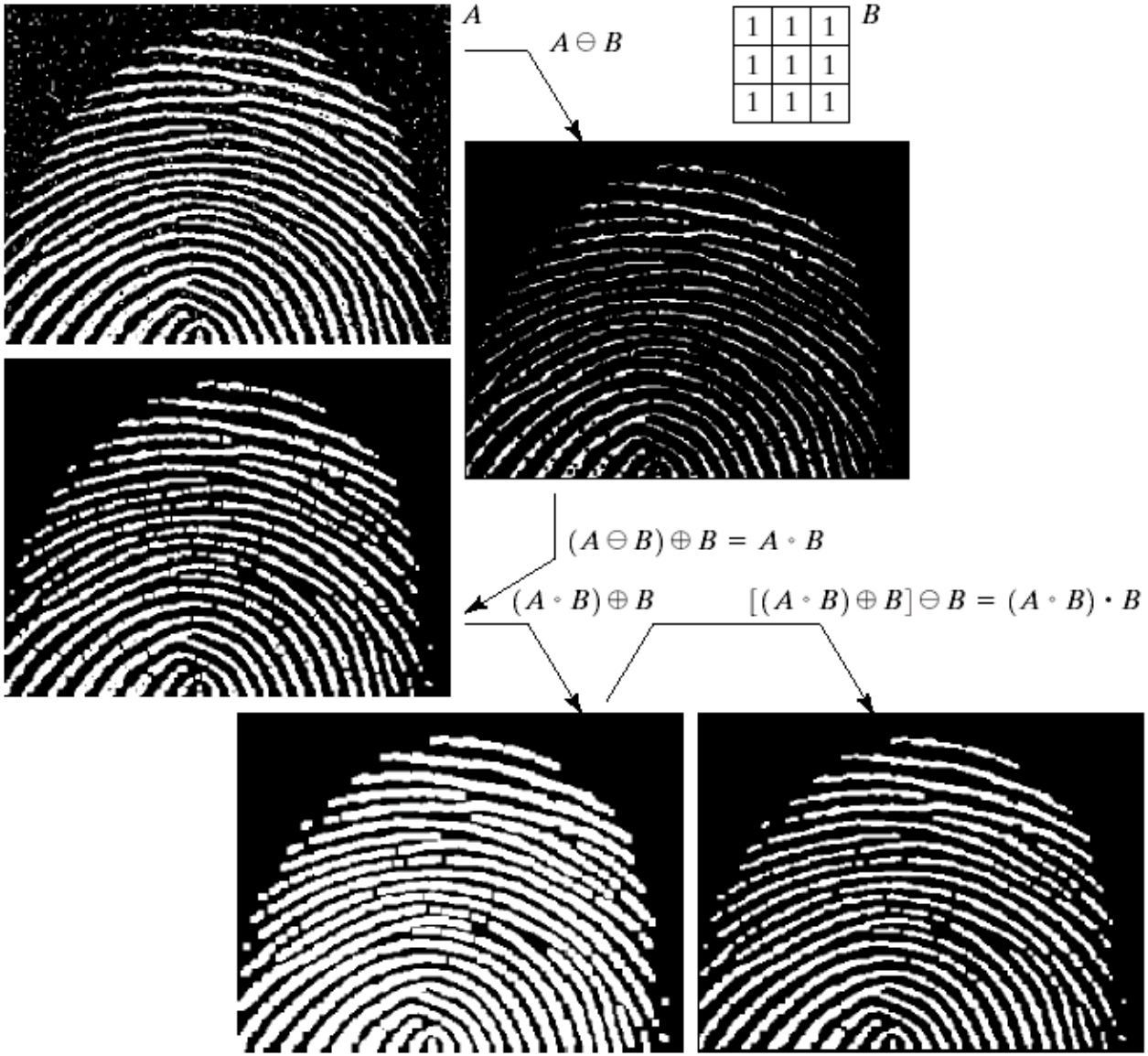
Processed Image





Structuring Element

Morphological Processing Example



Morphological Algorithms

Using the simple technique we have looked at so far we can begin to consider some more interesting morphological algorithms

We will look at:

- Boundary extraction
- Region filling

There are lots of others as well though:

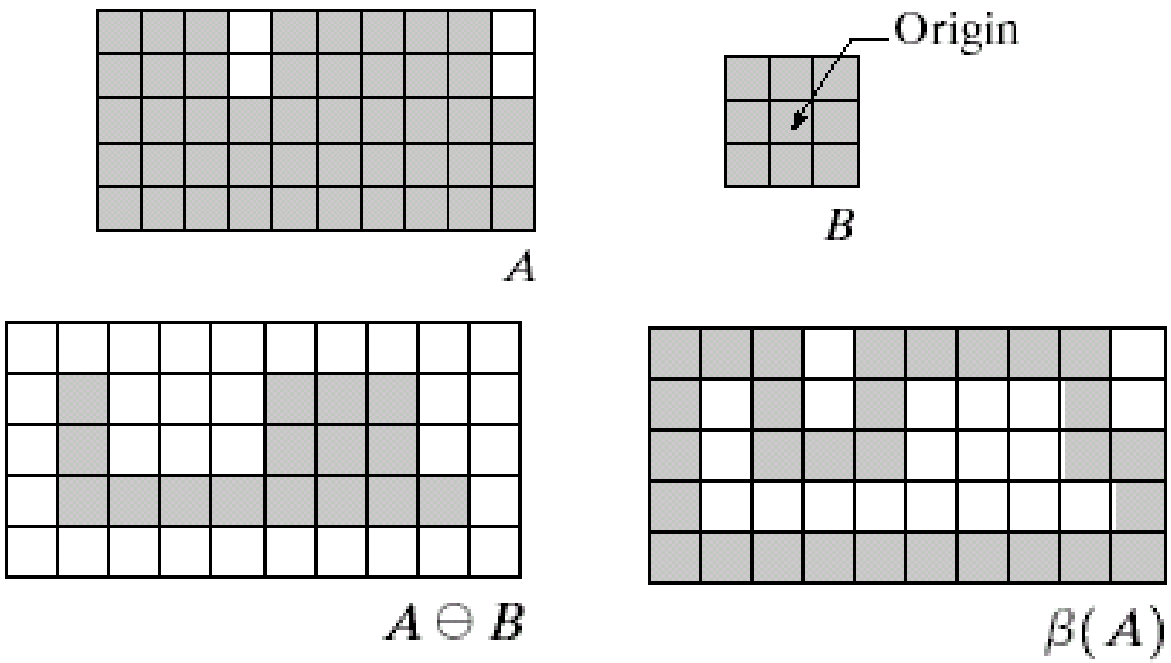
- Extraction of connected components
- Thinning/thickening
- Skeletonisation

Boundary Extraction

Extracting the boundary (or outline) of an object is often extremely useful

The boundary can be given simply as

$$\beta(A) = A - (A \ominus B)$$



Boundary Extraction Example

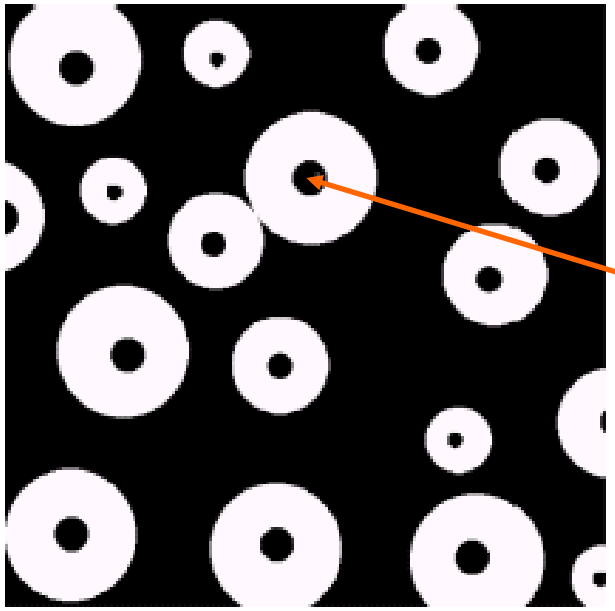
A simple image and the result of performing boundary extraction using a square 3*3 structuring element

Original Image

Extracted Boundary

Region Filling

Given a pixel inside a boundary, *region filling* attempts to fill that boundary with object pixels (1s)



Given a point inside here, can we fill the whole circle?

Region Filling (cont…)

The key equation for region filling is

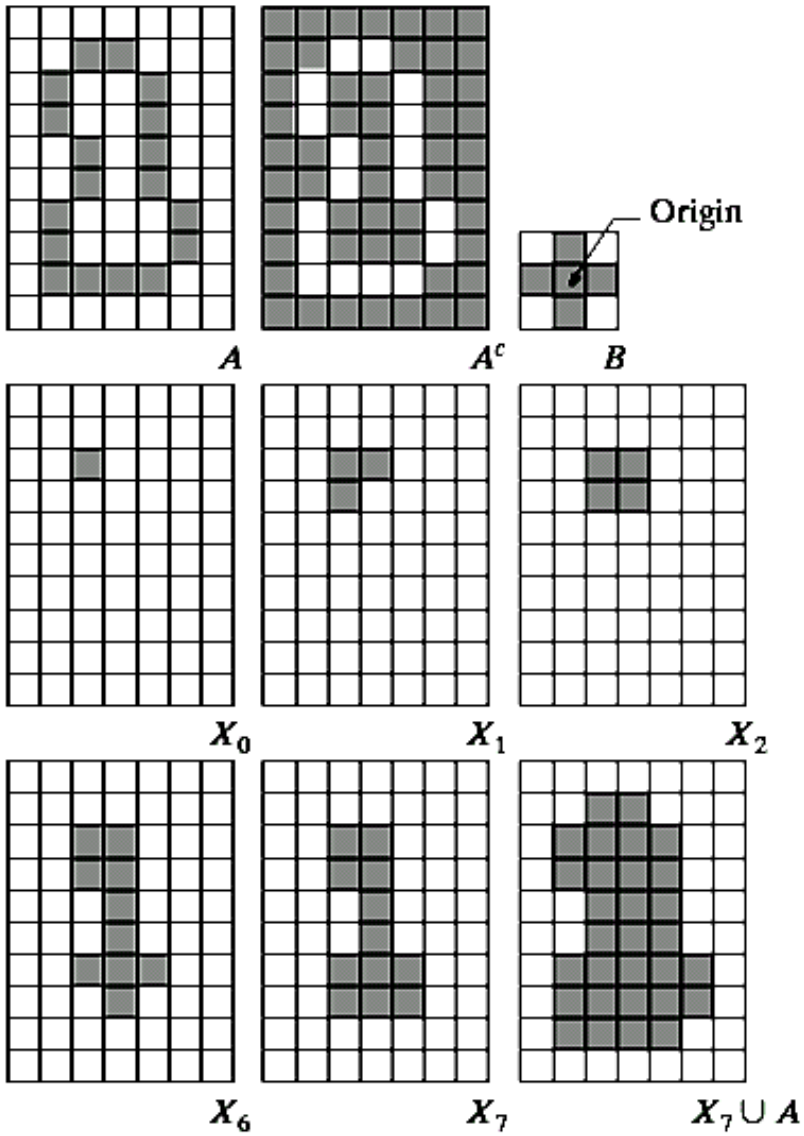
$$X_k = (X_{k-1} \oplus B) \cap A^c \qquad k = 1,2,3.....$$

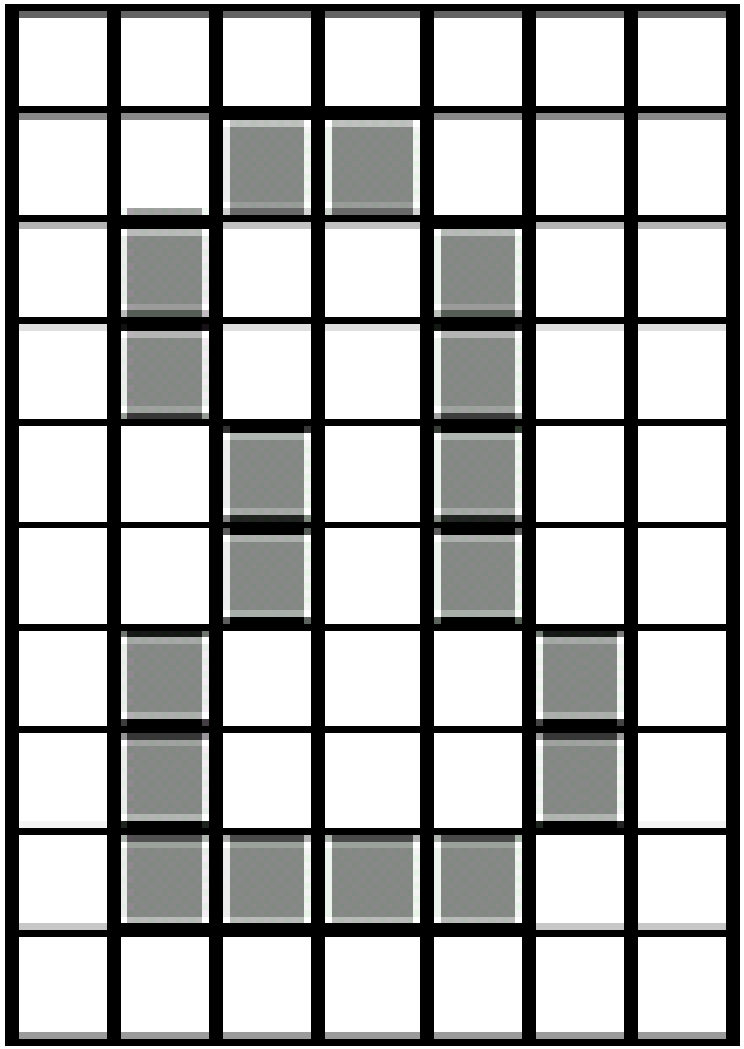
Where X_0 is simply the starting point inside the boundary, B is a simple structuring element and A^c is the complement of A

This equation is applied repeatedly until X_k is equal to X_{k-1}

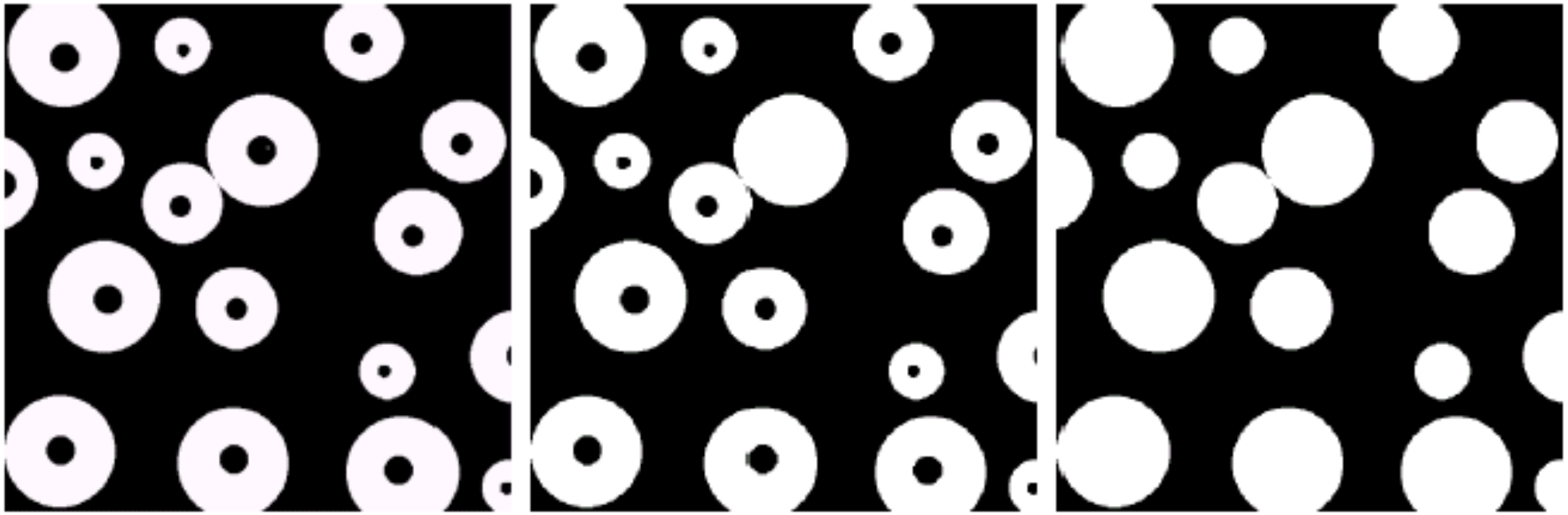
Finally the result is unioned with the original boundary

Region Filling Step By Step





Region Filling Example



Original Image | One Region Filled | All Regions Filled

Summary

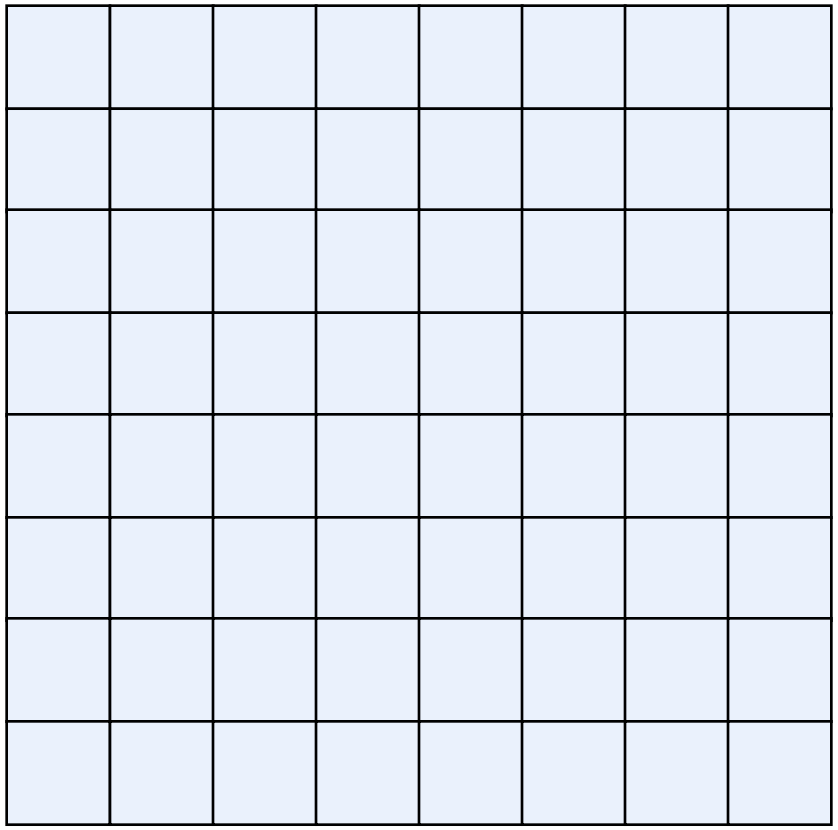
The purpose of morphological processing is primarily to remove imperfections added during segmentation

The basic operations are *erosion* and *dilation*

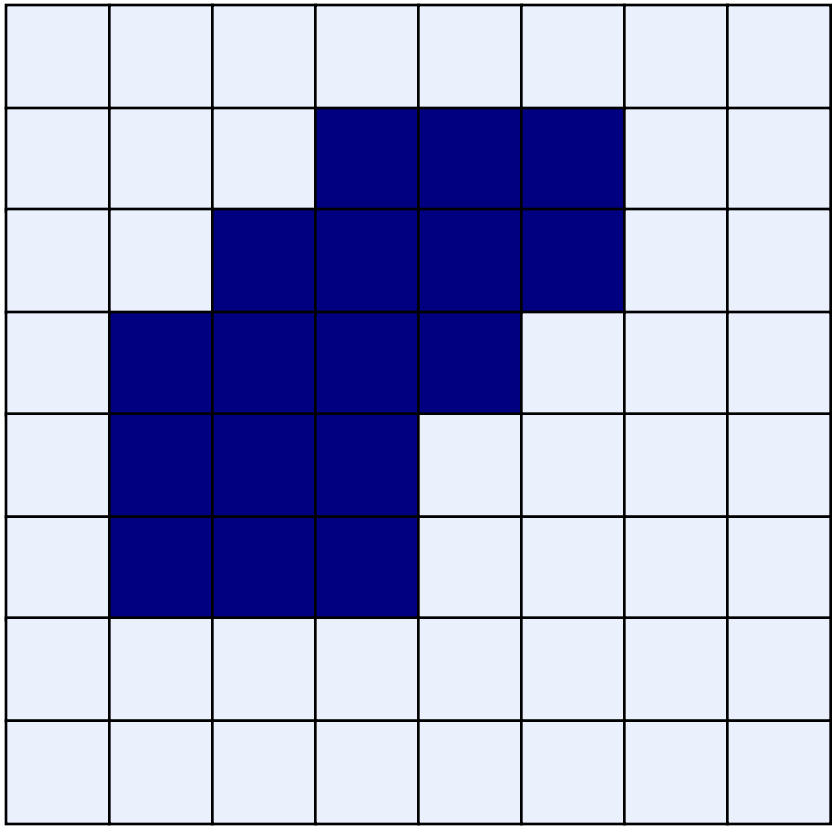
Using the basic operations we can perform *opening* and *closing*

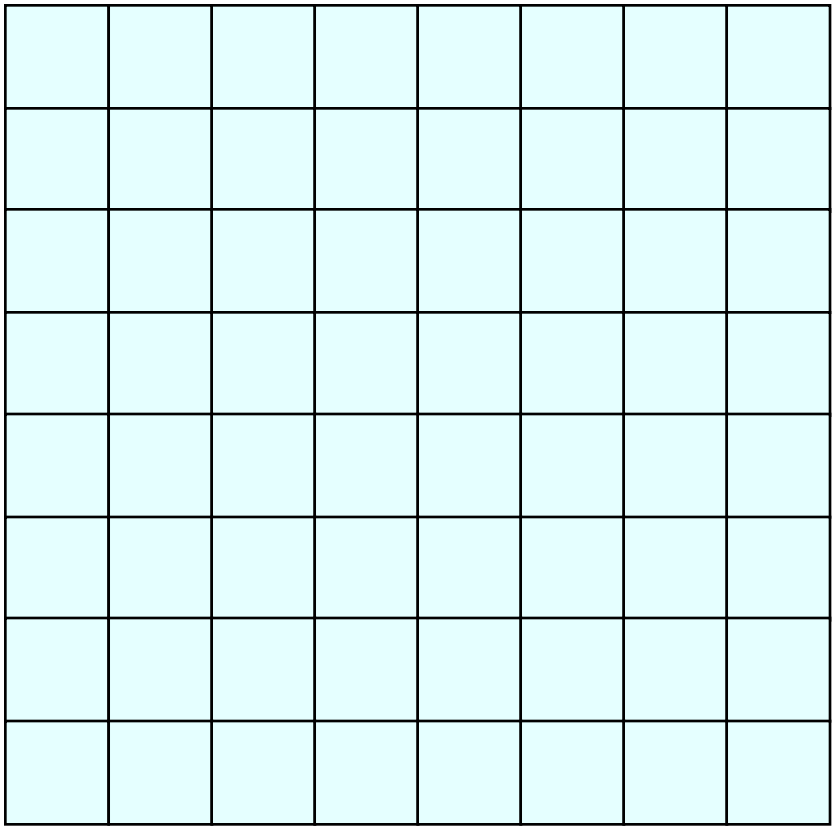
More advanced morphological operation can then be implemented using combinations of all of these

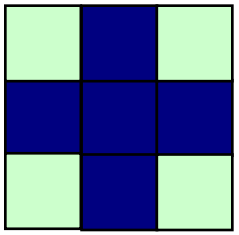
Structuring Elements, Hits & Fits



Erosion Example

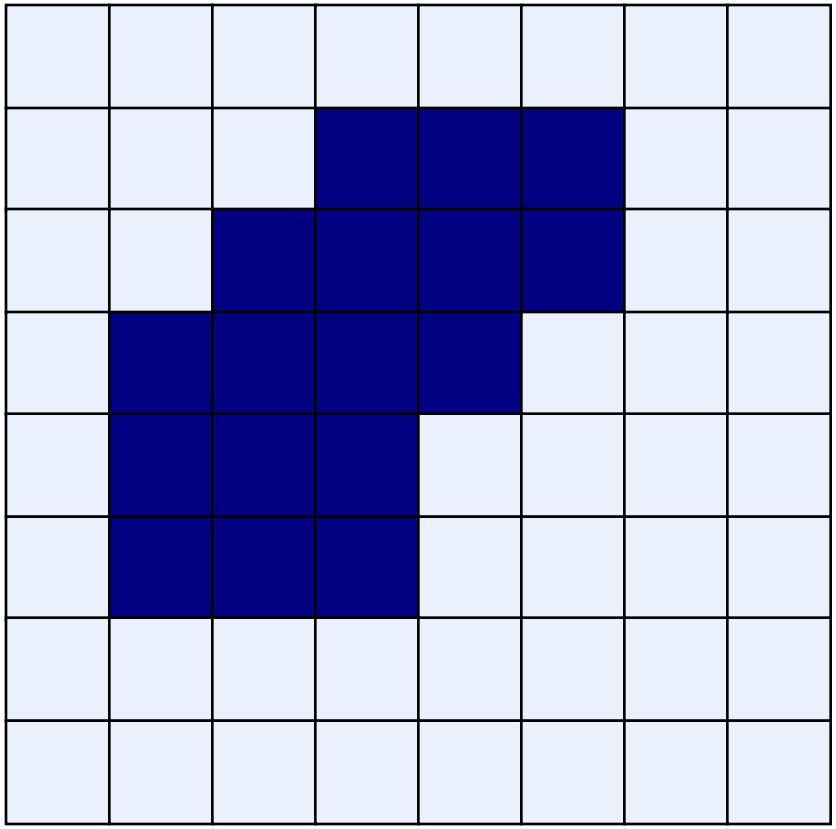


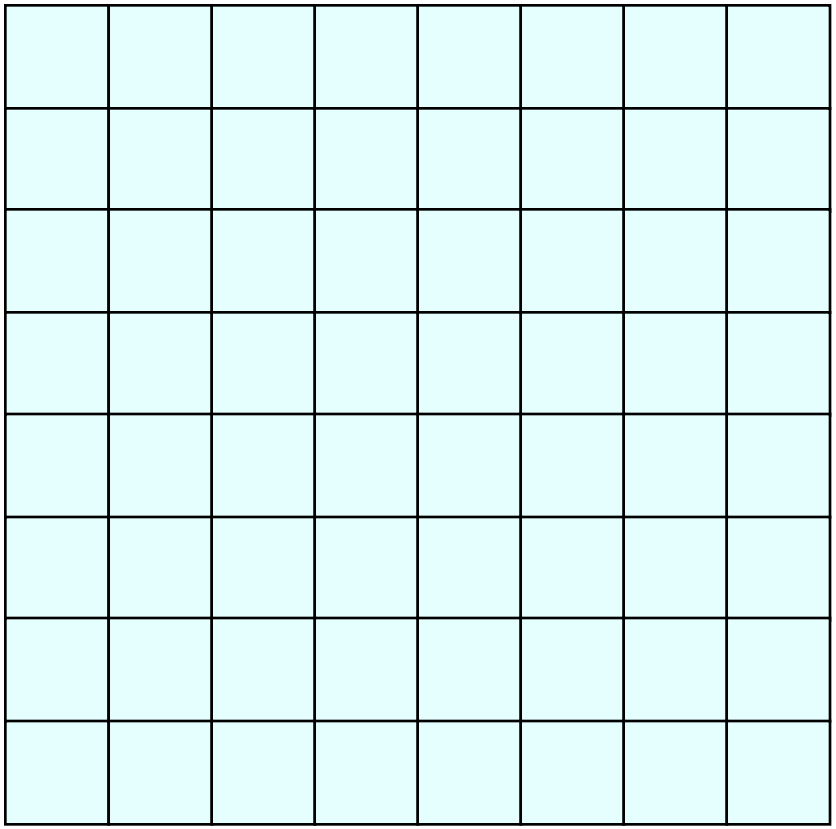


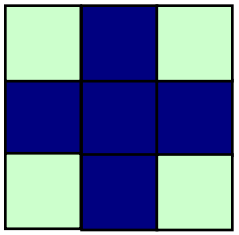


Structuring Element

Dilation Example



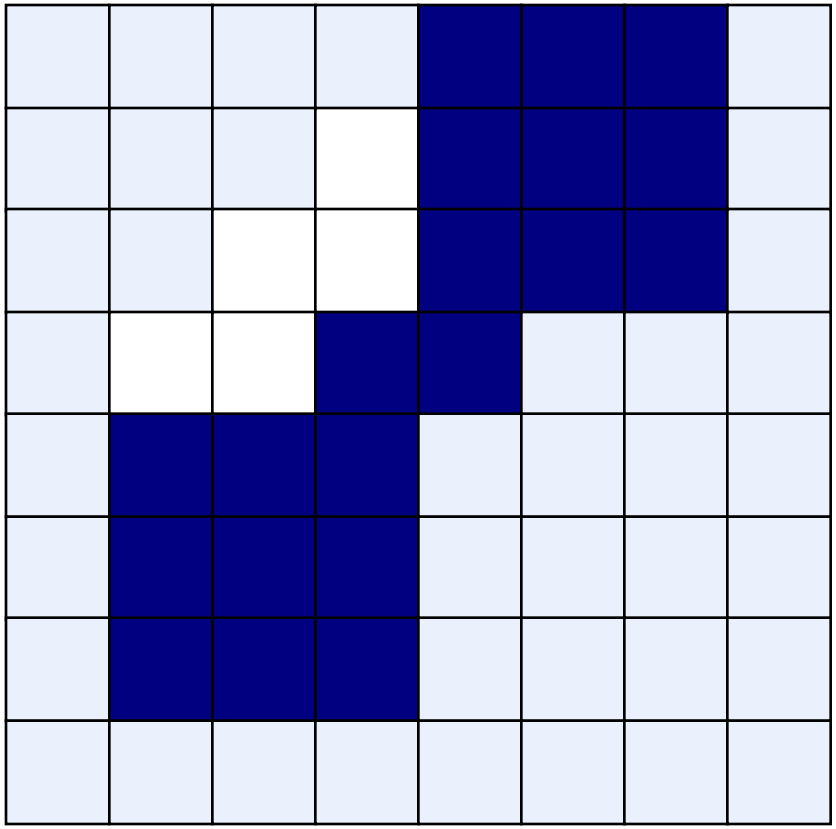




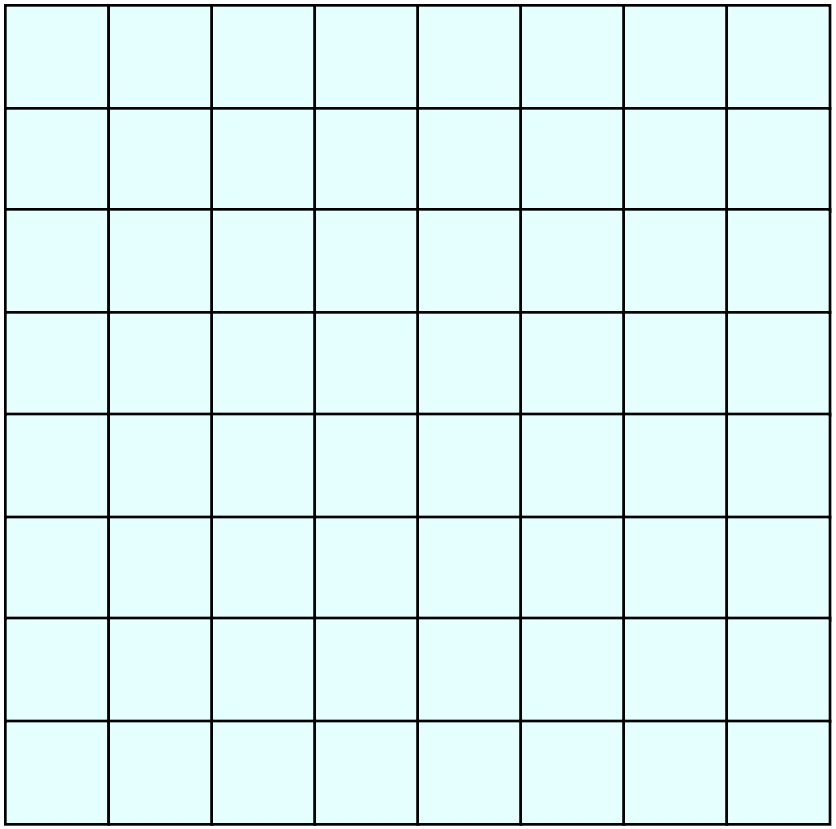
Structuring Element

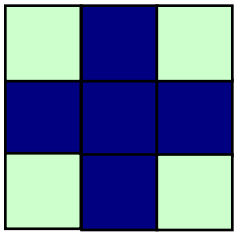
Opening Example

Original Image



Processed Image

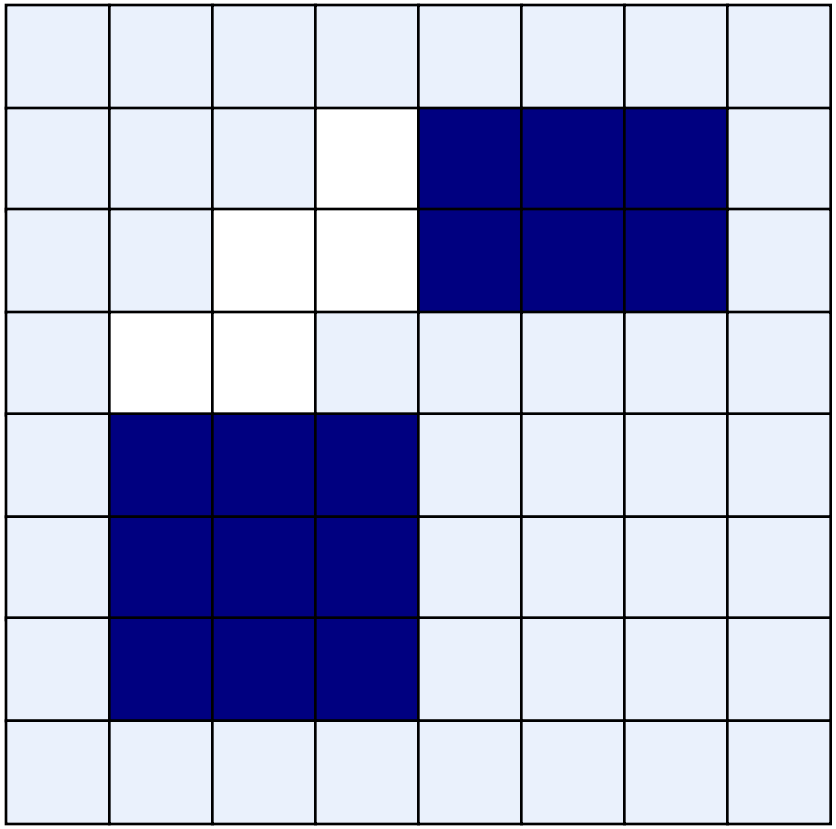




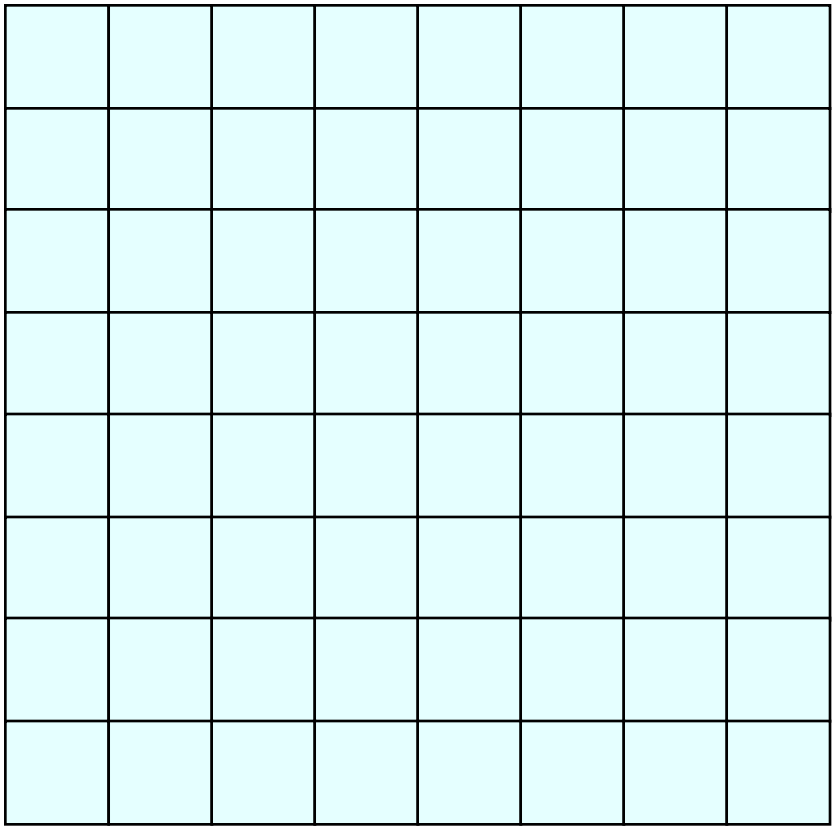
Structuring Element

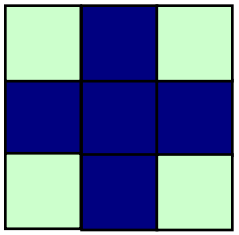
Closing Example

Original Image



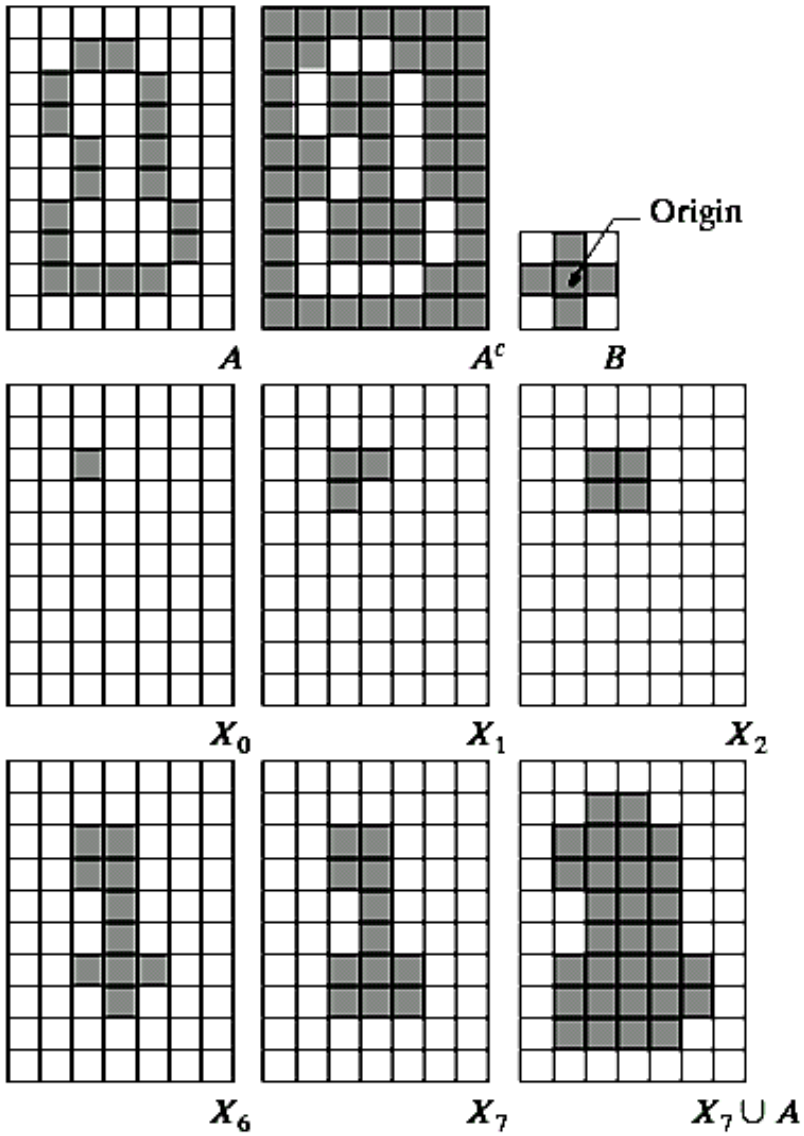
Processed Image

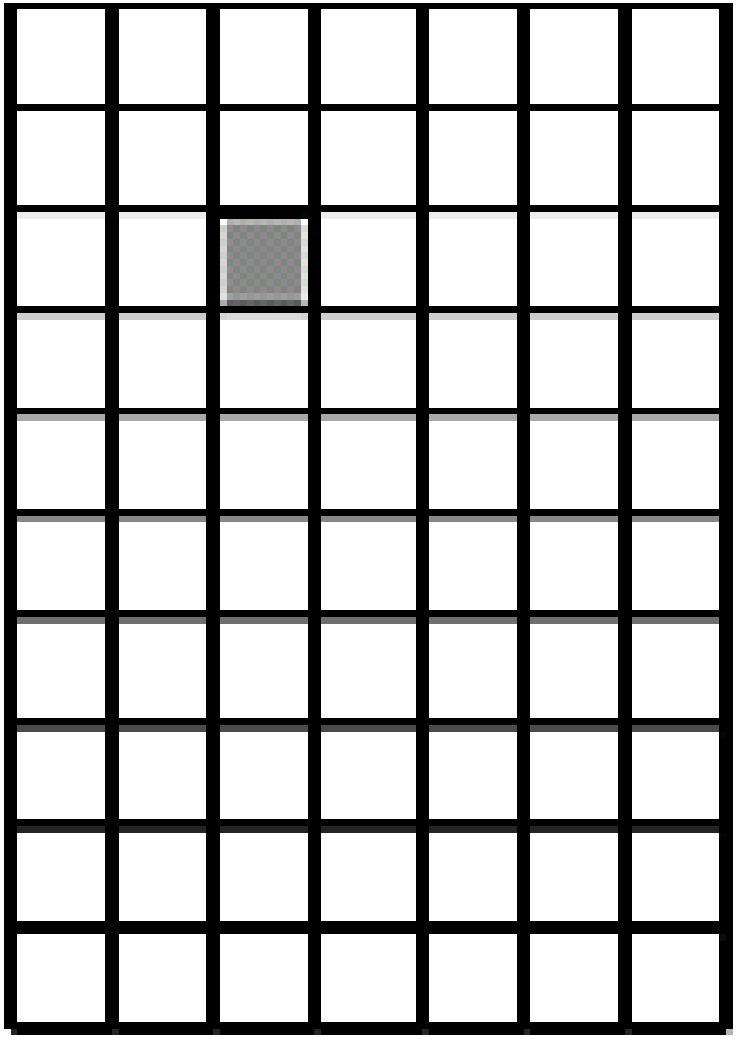




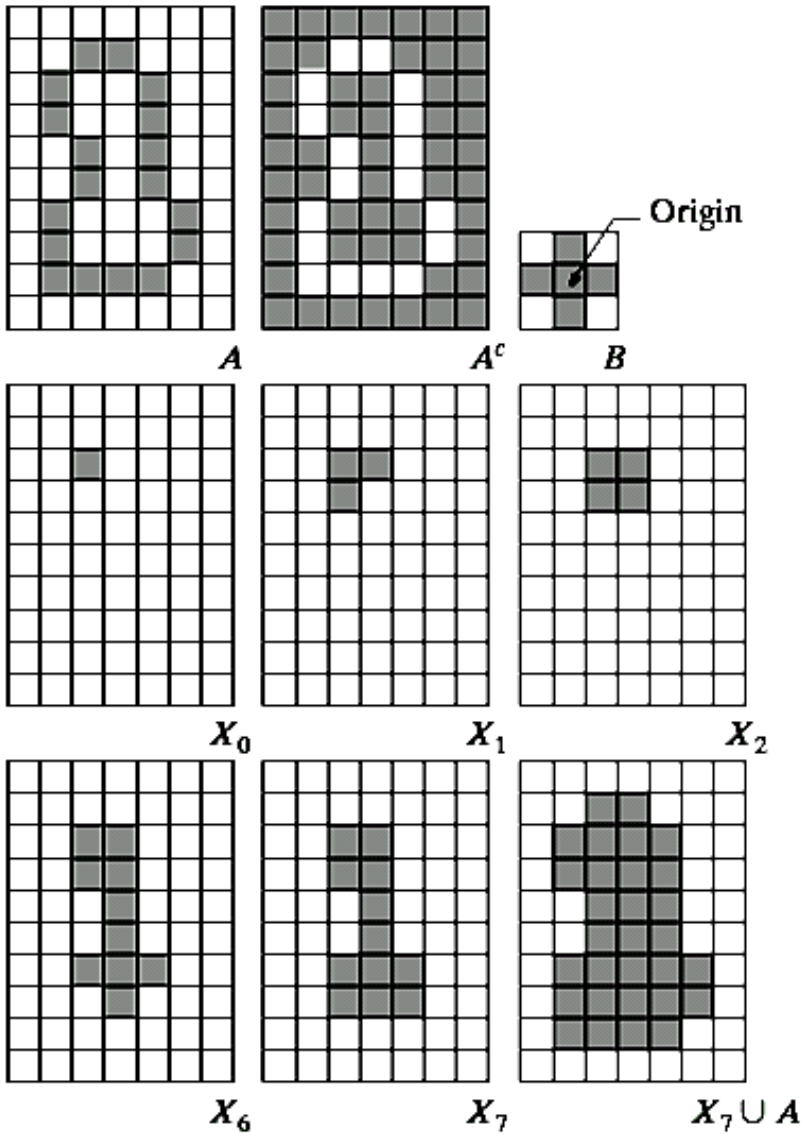
Structuring Element

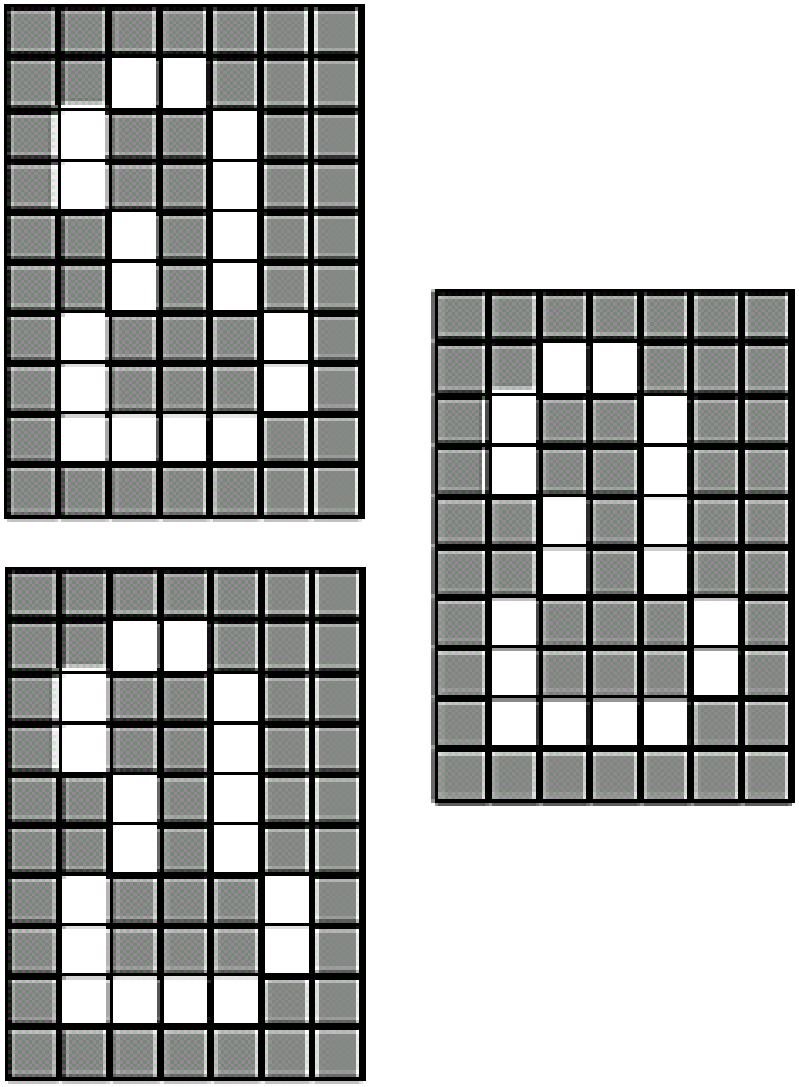
Region Filling Step By Step





Region Filling Step By Step





Lecture - 13

Image Compression

Image Compression

- The goal of image compression is to reduce the amount of data required to represent a digital image.

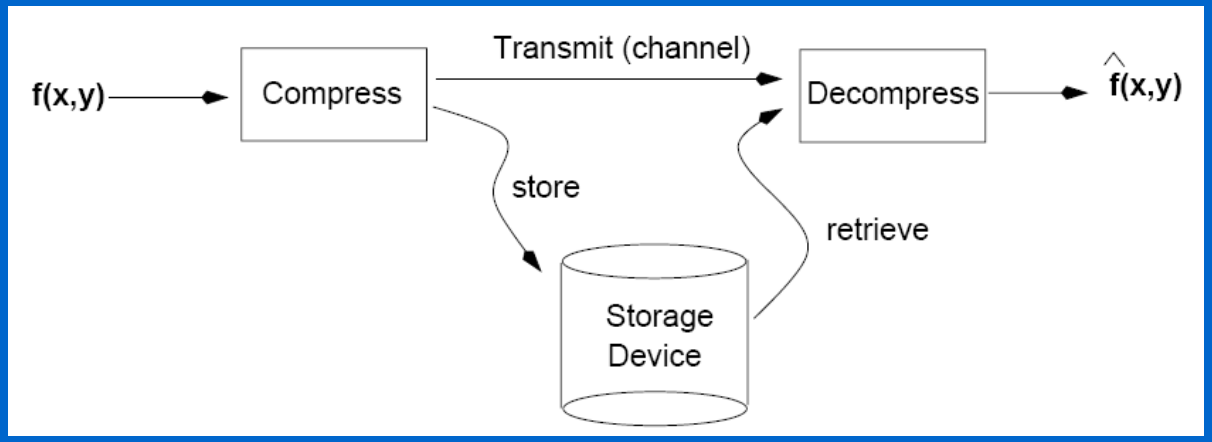


Image Compression (cont'd)

- Lossless
  - Information preserving
  - Low compression ratios
- Lossy
  - Not information preserving
  - High compression ratios

<u>Trade-off:</u> information loss **vs** compression ratio

Data ≠ Information

- Data and information are not synonymous terms!
- Data is the means by which information is conveyed.
- Data compression aims to reduce the amount of data while preserving as much information as possible.

Data vs Information (cont'd)

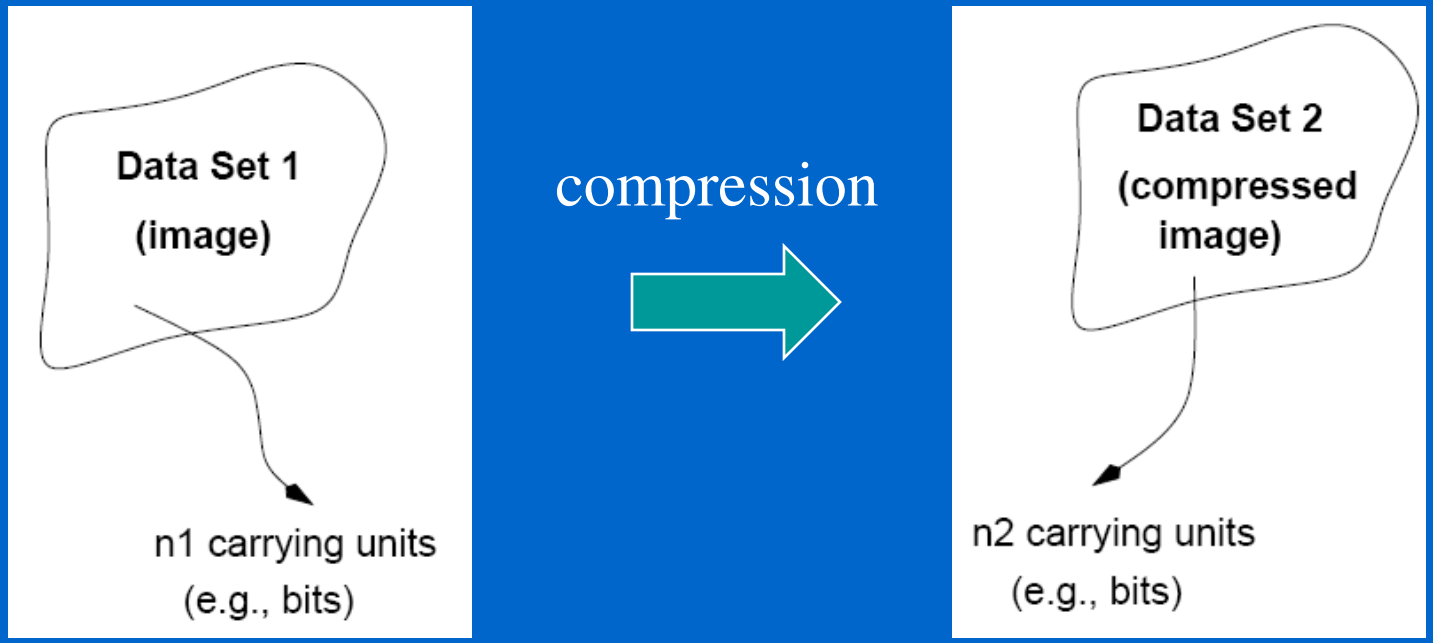
- The same information can be represented by different amount of data – for example:

Ex1: *Your wife, Helen, will meet you at Shah Mukhdam Airport in Rajshahi at 5 minutes past 6:00 pm tomorrow night*

Ex2: *Your wife will meet you at Shah Mukhdam Airport at 5 minutes past 6:00 pm tomorrow night*

Ex3: *Helen will meet you at Shah Mukhdam at 6:00 pm tomorrow night*

Compression Ratio



Compression ratio: $C_R = \frac{n_1}{n_2}$

Relevant Data Redundancy

$$R_D = 1 - \frac{1}{C_R}$$

Example:

If $C_R = \frac{10}{1}$, then $R_D = 1 - \frac{1}{10} = 0.9$

(90% of the data in dataset 1 is redundant)

if $n_2 = n_1$, then C_R=1, R_D=0

if $n_2 \ll n_1$, then $C_R \to \infty$, $R_D \to 1$

Types of Data Redundancy

(1) Coding Redundancy

(2) Interpixel Redundancy

(3) Psychovisual Redundancy

- Data compression attempts to reduce one or more of these redundancy types.

Coding - Definitions

- Code: a list of symbols (letters, numbers, bits etc.)
- Code word: a sequence of symbols used to represent some information (e.g., gray levels).
- Code word length: number of symbols in a code word.

Example: (binary code, symbols: 0,1, length: 3)

0: 000	4: 100
1: 001	5: 101
2: 010	6: 110
3: 011	7: 111

Coding - Definitions (cont'd)

N x M image

$\mathbf{r_k}$: k-th gray level

$\mathbf{l(r_k)}$: # of bits for r_k

$\mathbf{P(r_k)}$: probability of r_k

Average # of bits: $L_{avg} = E(l(r_k)) = \sum_{k=0}^{L-1} l(r_k)P(r_k)$

Total # of bits: NML_{avg}

Expected value: $E(X) = \sum_{x} xP(X = x)$

Coding Redundancy

- Case 1: $l(r_k)$ = constant length

Example:

r_k	$p_r(r_k)$	*Code 1*	$l_1(r_k)$
$r_0 = 0$	0.19	000	3
$r_1 = 1/7$	0.25	001	3
$r_2 = 2/7$	0.21	010	3
$r_3 = 3/7$	0.16	011	3
$r_4 = 4/7$	0.08	100	3
$r_5 = 5/7$	0.06	101	3
$r_6 = 6/7$	0.03	110	3
$r_7 = 1$	0.02	111	3

Assume an image with $L = 8$

Assume $l(r_k) = 3$, $L_{avg} = \sum_{k=0}^{7} 3P(r_k) = 3\sum_{k=0}^{7} P(r_k) = 3$ bits

Total number of bits: $3NM$

Coding Redundancy (cont'd)

- <u>Case 2:</u> $l(r_k)$ = variable length

Table 6.1 Variable-Length Coding Example variable length

r_k	$p_r(r_k)$	*Code 1*	$l_1(r_k)$	*Code 2*	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

$$L_{avg} = \sum_{k=0}^{7} l(r_k)P(r_k) = 2.7 \text{ bits}$$

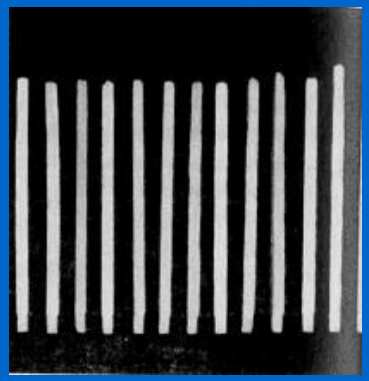
$$C_R = \frac{3}{2.7} = 1.11 \text{ (about 10\%)}$$

$$R_D = 1 - \frac{1}{1.11} = 0.099$$

Total number of bits: 2.7NM

Interpixel redundancy

- Interpixel redundancy implies that pixel values are correlated (i.e., a pixel value can be reasonably predicted by its neighbors).



$$f(x)\,o\,g(x) = \int_{-\infty}^{\infty} f(x)g(x+a)da$$

auto-correlation: f(x)=g(x)

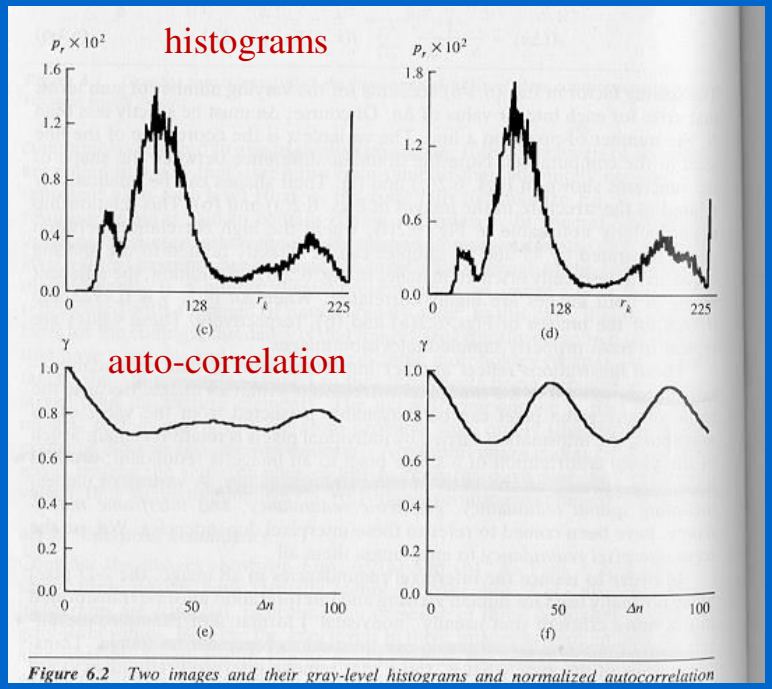


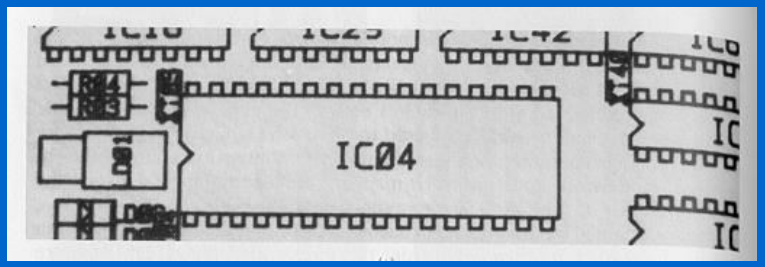
Figure 6.2 Two images and their gray-level histograms and normalized autocorrelation

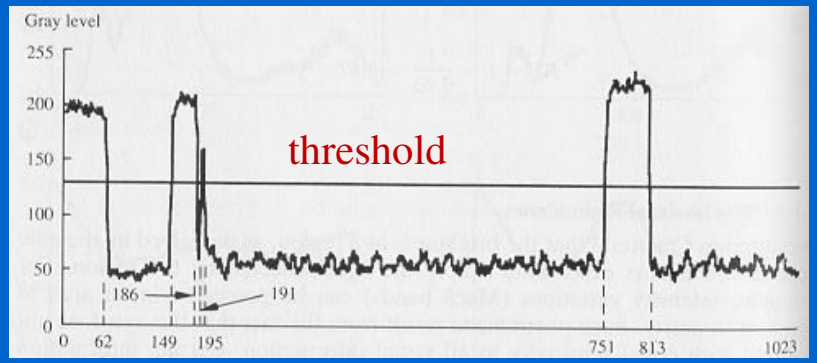
Interpixel redundancy (cont'd)

- To reduce interpixel redundancy, some kind of transformation must be applied on the data (e.g., thresholding, DFT, DWT)

Example:

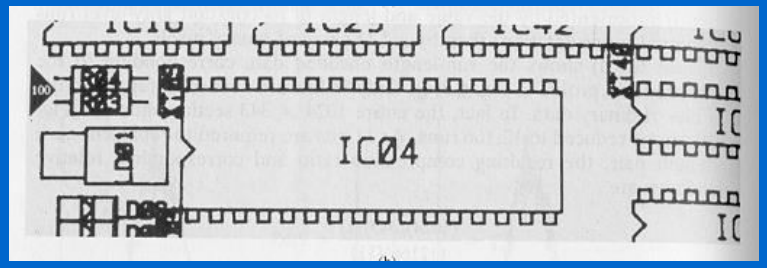
original





11 ……0000……..11…..000…..

thresholded



Run-length encoding:

(1,63) (0,87) (1,37) (0,5) (1,4) (0, 556) (1,62) (0,210)

Using 11 bits/pair: (1+10) bits/pair

88 bits are required (compared to 1024 !!)

Psychovisual redundancy

- The human eye is more sensitive to the lower frequencies than to the higher frequencies in the visual spectrum.

- Idea: discard data that is perceptually insignificant!

Psychovisual redundancy (cont'd)

Example: quantization

256 gray levels

16 gray levels

16 gray levels + random noise

C=8/4 = 2:1

add a small pseudo-random number to each pixel prior to quantization

Measuring Information

- A key question in image compression is:

  "What is the minimum amount of data that is sufficient to describe completely an image without loss of information?"

- How do we measure the information content of an image?

Measuring Information (cont'd)

- We assume that information generation is a probabilistic process.

- Idea: associate information with probability!

A random event E with probability $P(E)$ contains:

$$I(E) = log(\frac{1}{P(E)}) = -log(P(E)) \text{ units of information}$$

Note: I(E)=0 when P(E)=1

How much information does a pixel contain?

- Suppose that gray level values are generated by a random process, then r_k contains:

$$I(r_k) = -\log(P(r_k))$$

units of information!

(assume statistically independent random events)

How much information does an image contain?

- Average information content of an image:

$$E = \sum_{k=0}^{L-1} I(r_k)\mathrm{P}(r_k)$$

using $$I(r_k) = -\log(P(r_k))$$

Entropy: $$H = -\sum_{k=0}^{L-1} P(r_k)\log(P(r_k))$$ units/pixel (e.g., bits/pixel)

Redundancy - revisited

- **Redundancy:** $$R = L_{avg} - H$$

where: $$L_{avg} = E(l(r_k)) = \sum_{k=0}^{L-1} l(r_k)P(r_k)$$

Note: if $L_{avg} = H$, then $R=0$ (no redundancy)

Entropy Estimation

- It is not easy to estimate H reliably!

image

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Gray Level	*Count*	*Probability*
21	12	3/8
95	4	1/8
169	4	1/8
243	12	3/8

Entropy Estimation (cont'd)

- First order estimate of H:

$$H = -\sum_{k=0}^{3} P(r_k) log(P(r_k)) = 1.81 \text{ bits/pixel}$$

Total bits: 4 x 8 x 1.81 = 58 bits

L_{avg} = 8 bits/pixel R= L_{avg}-H

The first-order estimate provides only a <u>lower-bound</u> on the compression that can be achieved.

Estimating Entropy (cont'd)

- Second order estimate of H:
  - Use relative frequencies of pixel blocks :

image

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Gray Level Pair	*Count*	*Probability*
(21, 21)	8	1/4
(21, 95)	4	1/8
(95, 169)	4	1/8
(169, 243)	4	1/8
(243, 243)	8	1/4
(243, 21)	4	1/8

$$H = 2.5/2 = 1.25 \text{ bits/pixel}$$

Differences in Entropy Estimates

- Differences between higher-order estimates of entropy and the first-order estimate indicate the presence of interpixel redundancy!
- Need to apply some transformation to deal with interpixel redundancy!

Differences in Entropy Estimates (cont'd)

- For example, consider pixel differences:

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0
21	0	0	74	74	74	0	0

Gray Level or Difference	*Count*	*Probability*
0	16	1/2
21	4	1/8
74	12	3/8

Differences in Entropy Estimates (cont'd)

- What is the entropy of the <u>difference</u> image?

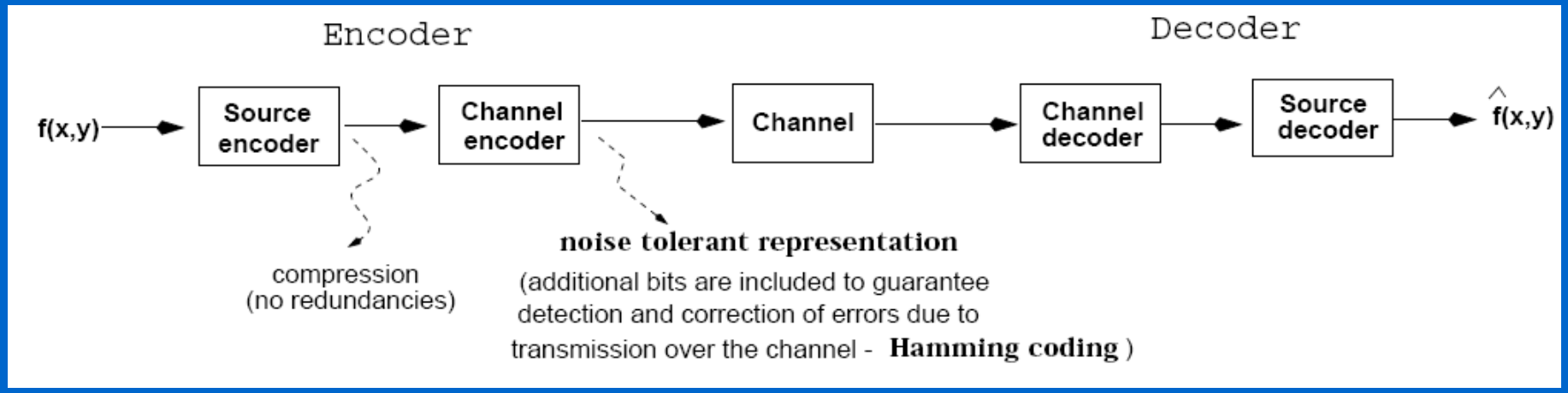
$$H = -\sum_{k=0}^{2} P(r_k) log(P(r_k)) = 1.41 \text{ bits/pixel}$$

(better than the entropy of the original image H=1.81)

- An even better transformation is possible since the second order entropy estimate is lower:

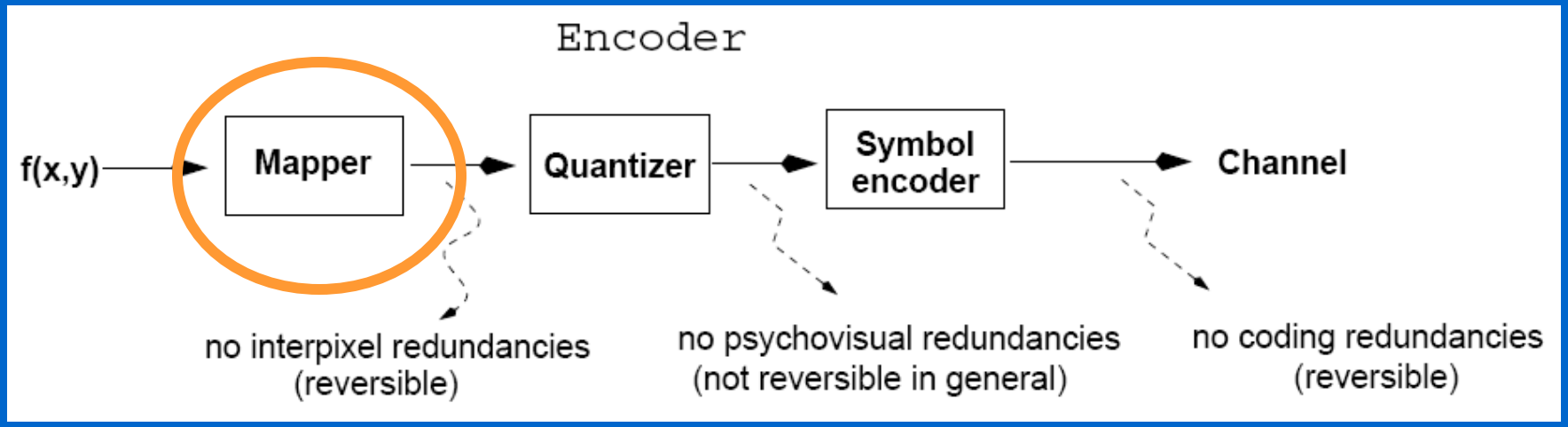
1.41 bits/pixel > 1.25 bits/pixel

Image Compression Model



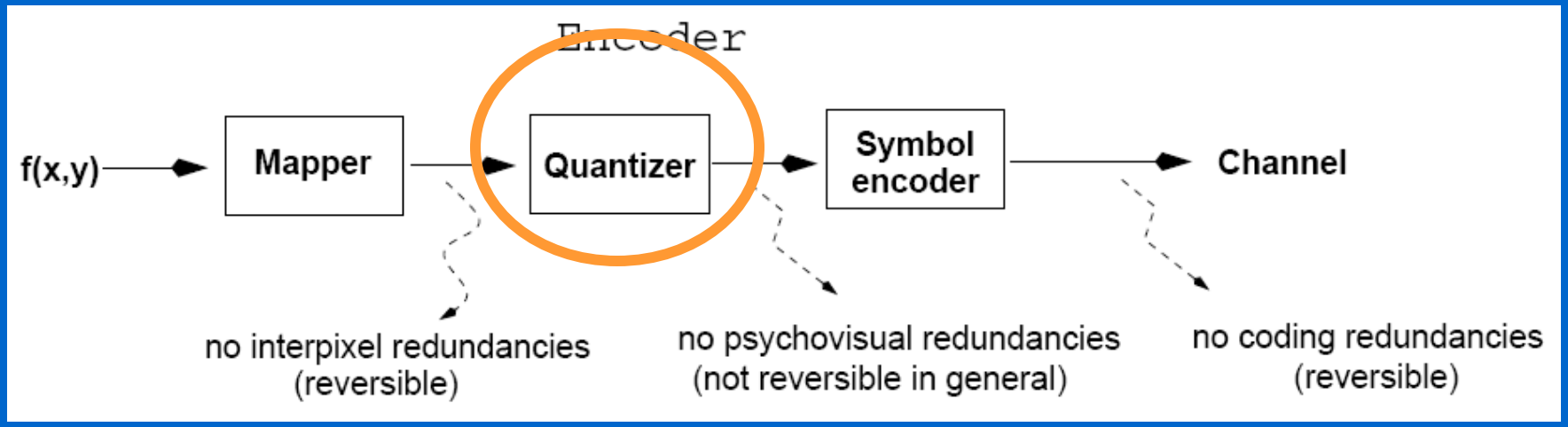
We will focus on the Source Encoder/Decoder only.

Image Compression Model (cont'd)



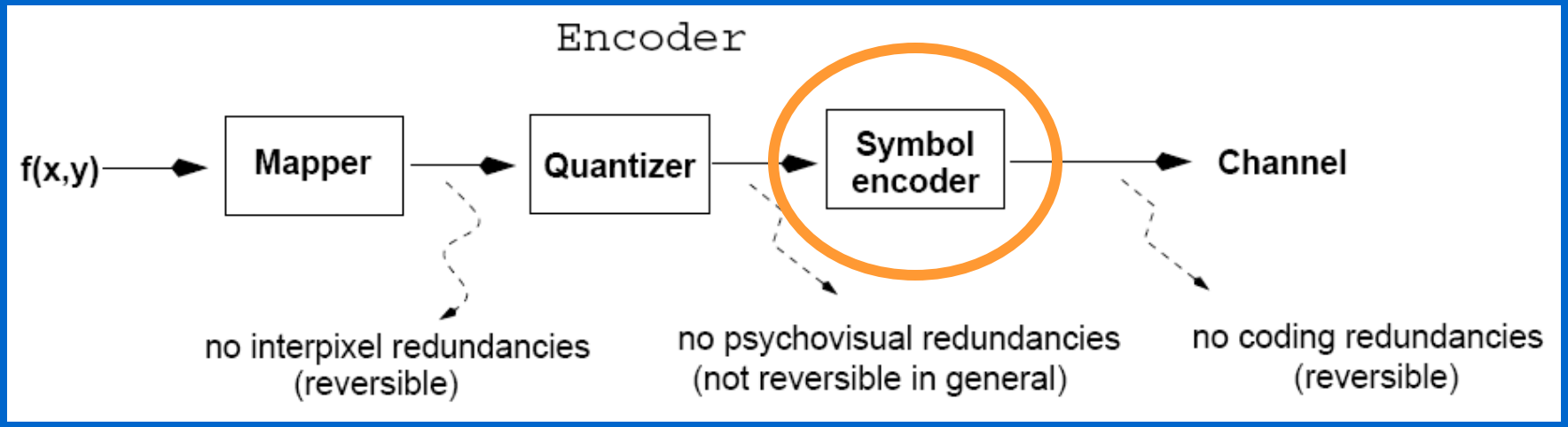
- **Mapper:** transforms data to account for interpixel redundancies.

Image Compression Model (cont'd)



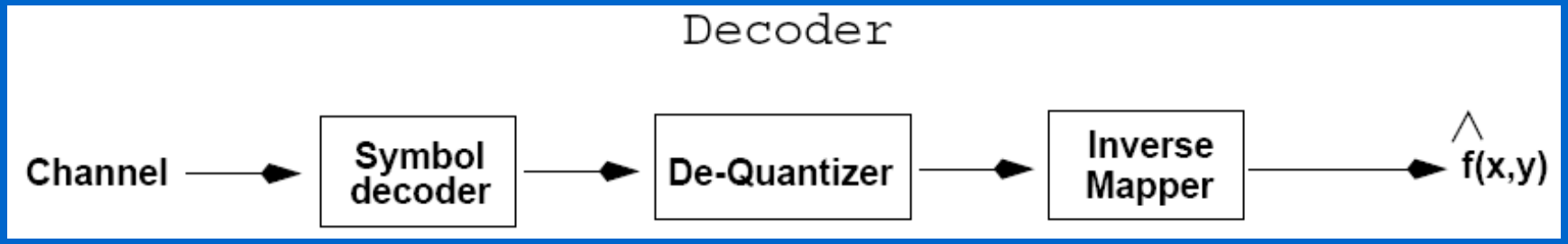
- **Quantizer:** quantizes the data to account for psychovisual redundancies.

Image Compression Model (cont'd)



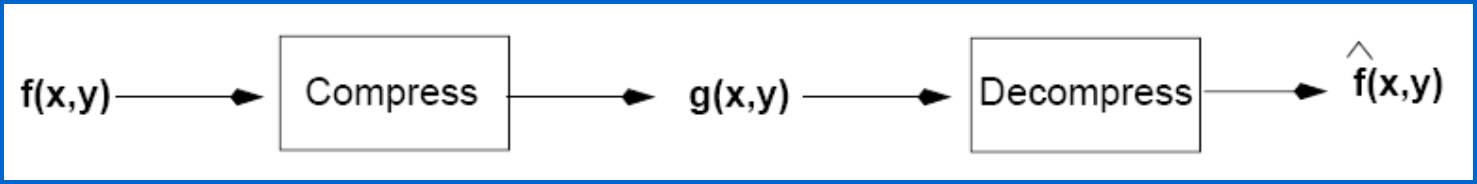
- **Symbol encoder:** encodes the data to account for coding redundancies.

Image Compression Models (cont'd)



- The decoder applies the inverse steps.
- Note that quantization is irreversible in general.

Fidelity Criteria



- How close is $f(x,y)$ to $\hat{f}(x,y)$?
- Criteria
  - Subjective: based on human observers
  - Objective: mathematically defined criteria

Subjective Fidelity Criteria

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

Objective Fidelity Criteria

- Root mean square error (RMS)

$$e_{rms} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (\hat{f}(x,y) - f(x,y))^2}$$

- Mean-square signal-to-noise ratio (SNR)

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (\hat{f}(x,y))^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (\hat{f}(x,y) - f(x,y))^2}$$

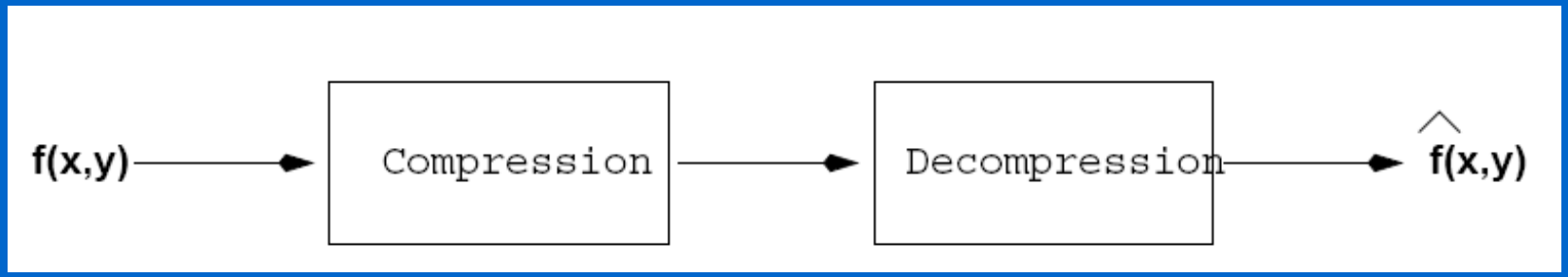
Subjective vs Objective Fidelity Criteria

RMSE = 5.17

RMSE = 15.67

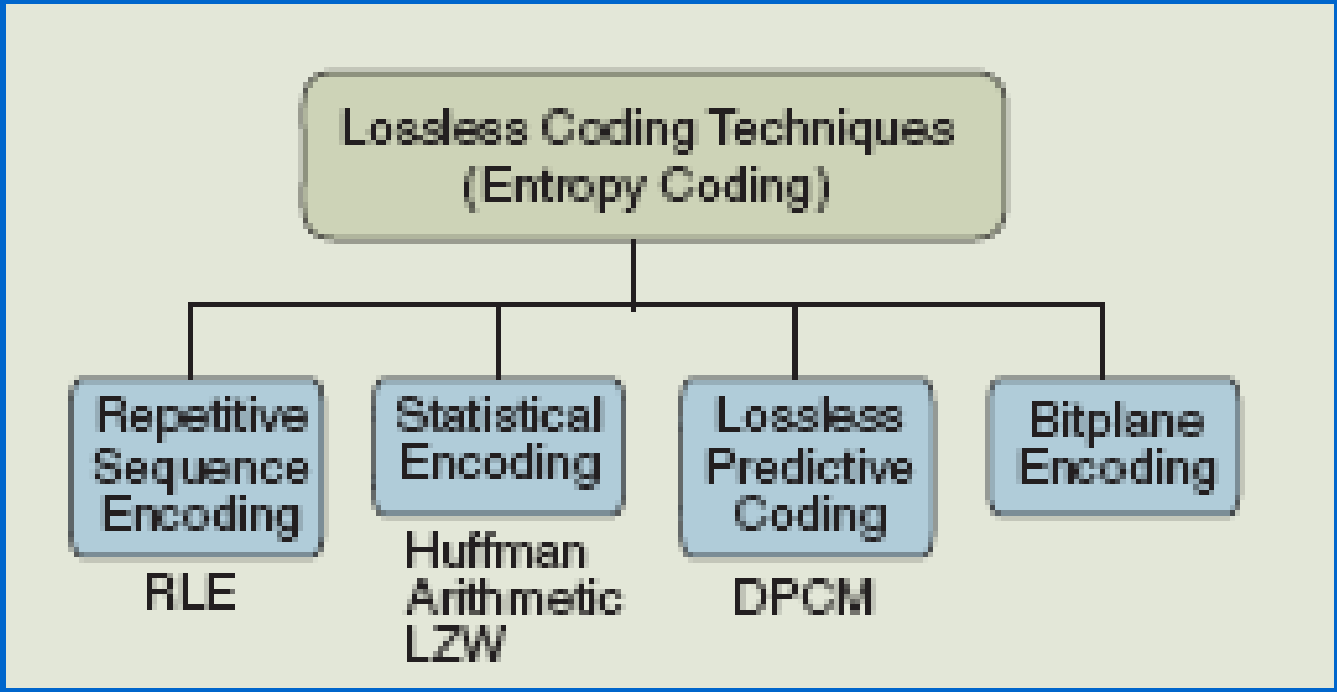
RMSE = 14.17

Lossless Compression

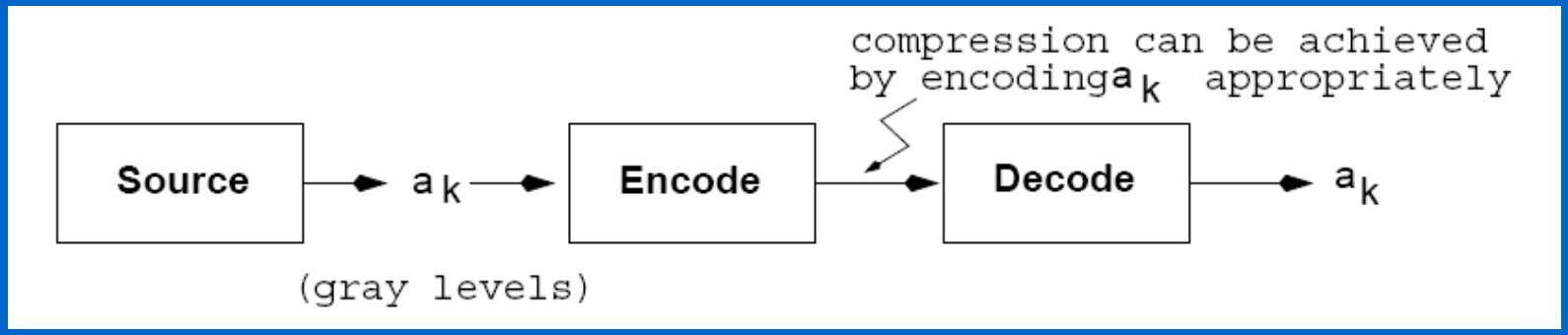


$$e(x, y) = \hat{f}(x, y) - f(x, y) = 0$$

Taxonomy of Lossless Methods



Huffman Coding
(addresses coding redundancy)



- A variable-length coding technique.
- Source symbols are encoded one at a time!
  - There is a one-to-one correspondence between source symbols and code words.
- Optimal code - minimizes code word length per source symbol.

Huffman Coding (cont'd)

- *Forward Pass*
  1. Sort probabilities per symbol
  2. Combine the lowest two probabilities
  3. Repeat *Step2* until only two probabilities remain.

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Huffman Coding (cont'd)

- *Backward Pass*

  Assign code symbols going backwards

Original source			Source reduction							
Sym.	Prob.	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

Huffman Coding (cont'd)

- L_{avg} assuming Huffman coding:

$$L_{avg} = E(l(a_k)) = \sum_{k=1}^{6} l(a_k)P(a_k) =$$

3x0.1 + 1x0.4 + 5x0.06 + 4x0.1 + 5x0.04 + 2x0.3=2.2 bits/symbol

- L_{avg} assuming binary coding:

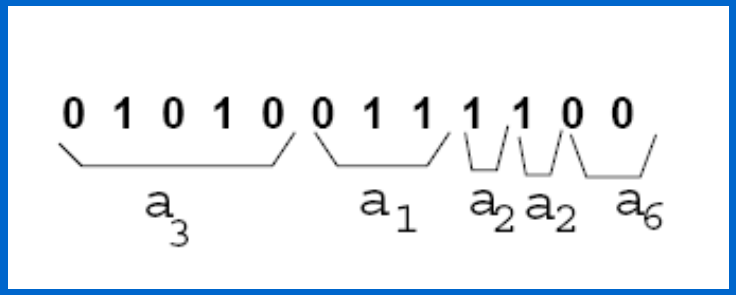
6 symbols, we need a 3-bit code

$(a_1: 000,\ a_2: 001,\ a_3: 010,\ a_4: 011,\ a_5: 100,\ a_6: 101)$

$$L_{avg} = \sum_{k=1}^{6} l(a_k)P(a_k) = \sum_{k=1}^{6} 3P(a_k) = 3\sum_{k=1}^{6} P(a_k) = 3 \text{ bits/symbol}$$

Huffman Coding/Decoding

- Coding/Decoding can be implemented using a look-up table.
- Decoding can be done unambiguously.



Original source

Sym.	Prob.	Code
a_2	0.4	1
a_6	0.3	00
a_1	0.1	011
a_4	0.1	0100
a_3	0.06	01010
a_5	0.04	01011

Arithmetic (or Range) Coding (addresses coding redundancy)

- The main weakness of Huffman coding is that it encodes source symbols one at a time.
- Arithmetic coding encodes sequences of source symbols together.
  - There is no one-to-one correspondence between source symbols and code words.
- Slower than Huffman coding but can achieve better compression.

Arithmetic Coding (cont'd)

- A sequence of source symbols is assigned to a sub-interval in [0,1) which can be represented by an arithmetic code, e.g.:

		sub-interval		arithmetic code
$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$	⇨	[0.06752, 0.0688)	⇨	0.068

- Start with the interval [0, 1) ; a sub-interval is chosen to represent the message which becomes smaller and smaller as the number of symbols in the message increases.

Arithmetic Coding (cont'd)

Encode message: $\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$

Source Symbol	Probability
a_1	0.2
a_2	0.2
a_3	0.4
a_4	0.2

1) Start with interval [0, 1)

0 1

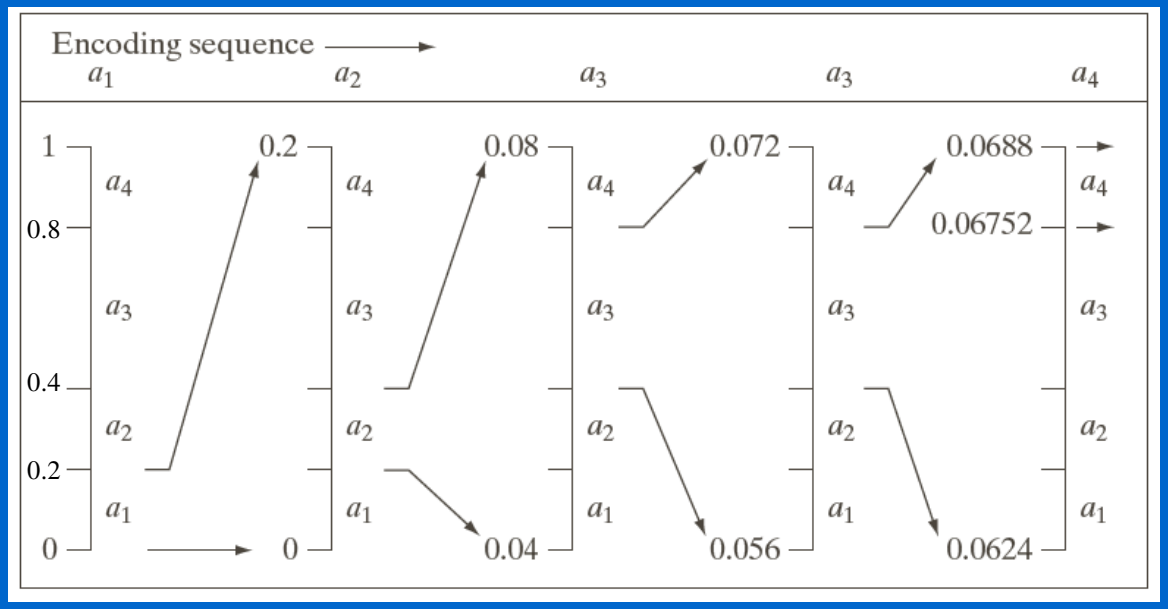
2) Subdivide [0, 1) based on the probabilities of α_i

Initial Subinterval
[0.0, 0.2)
[0.2, 0.4)
[0.4, 0.8)
[0.8, 1.0)

3) Update interval by processing source symbols

Example

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)



Encode

$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$

[0.06752, 0.0688)

or

0.068

(must be inside sub-interval)

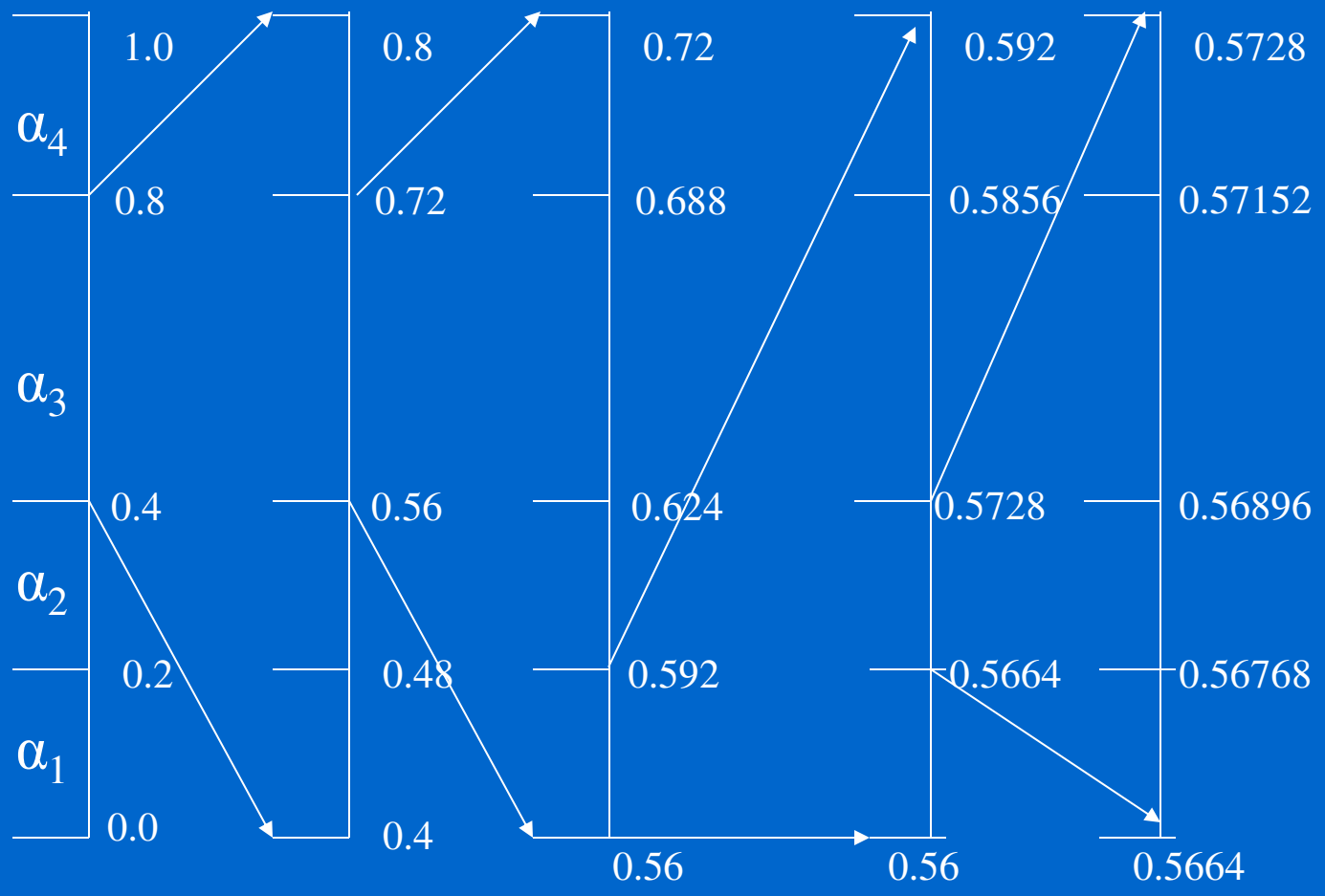
Example (cont'd)

- The message $\alpha_1\ \alpha_2\ \alpha_3\ \alpha_3\ \alpha_4$ is encoded using 3 decimal digits or 3/5 = 0.6 decimal digits per source symbol.
- The entropy of this message is: $H = -\sum_{k=0}^{3} P(r_k) log(P(r_k))$

$$-(3 \times 0.2\log_{10}(0.2)+0.4\log_{10}(0.4))=0.5786 \text{ digits/symbol}$$

<u>Note:</u> finite precision arithmetic might cause problems due to truncations!

Arithmetic Decoding



Decode 0.572

(code length=4)

$\alpha_3\ \alpha_3\ \alpha_1\ \alpha_2\ \alpha_4$

LZW Coding
(addresses interpixel redundancy)

- Requires no prior knowledge of symbol probabilities.
- Assigns fixed length code words to variable length symbol sequences.
  - There is no one-to-one correspondence between source symbols and code words.
- Included in GIF, TIFF and PDF file formats

LZW Coding

- A codebook (or dictionary) needs to be constructed.
- Initially, the first 256 entries of the dictionary are assigned to the gray levels 0,1,2,..,255 (i.e., assuming 8 bits/pixel)

Consider a 4x4, 8 bit image

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Initial Dictionary

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	-
511	-

LZW Coding (cont'd)

As the encoder examines image pixels, gray level sequences (i.e., blocks) that are not in the dictionary are assigned to a new entry.

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Dictionary Location	Entry
0	0
1	1
.	.
255	255
256	**39-39**
511	-

- Is 39 in the dictionary……..Yes
- What about 39-39……No
  * Add 39-39 at location 256

Example

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

CR = empty
repeat
P=next pixel
CS=CR + P

If CS is found:
(1) No Output
(2) CR=CS

else:
(1) Output D(CR)
(2) Add CS to D
(3) CR=P

Concatenated Sequence: CS = CR + P

(CR) (P)

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

Decoding LZW

- Use the dictionary for decoding the “encoded output” sequence.
- The dictionary need not be sent with the encoded output.
- Can be built on the “fly” by the decoder as it reads the received code words.

Run-length coding (RLC)
(addresses interpixel redundancy)

- Reduce the size of a repeating string of symbols (i.e., runs):

1 1 1 1 1 0 0 0 0 0 0 1 → (1,5) (0, 6) (1, 1)

a a a b b b b b b c c → (a,3) (b, 6) (c, 2)

- Encodes a run of symbols into two bytes: **(symbol, count)**
- Can compress any type of data but cannot achieve high compression ratios compared to other compression methods.

Combining Huffman Coding with Run-length Coding

- Assuming that a message has been encoded using Huffman coding, additional compression can be achieved using run-length coding.

0 1 0 1 0 0 1 1 1 1 0 0

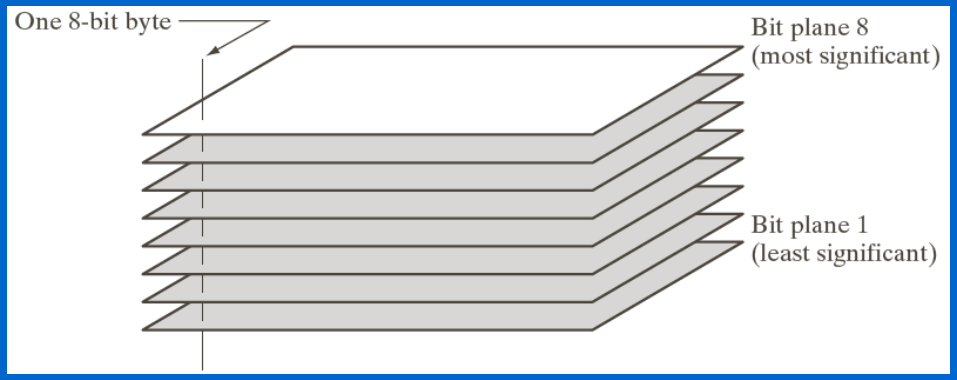
e.g., (0,1)(1,1)(0,1)(1,0)(0,2)(1,4)(0,2)

Bit-plane coding
(addresses interpixel redundancy)

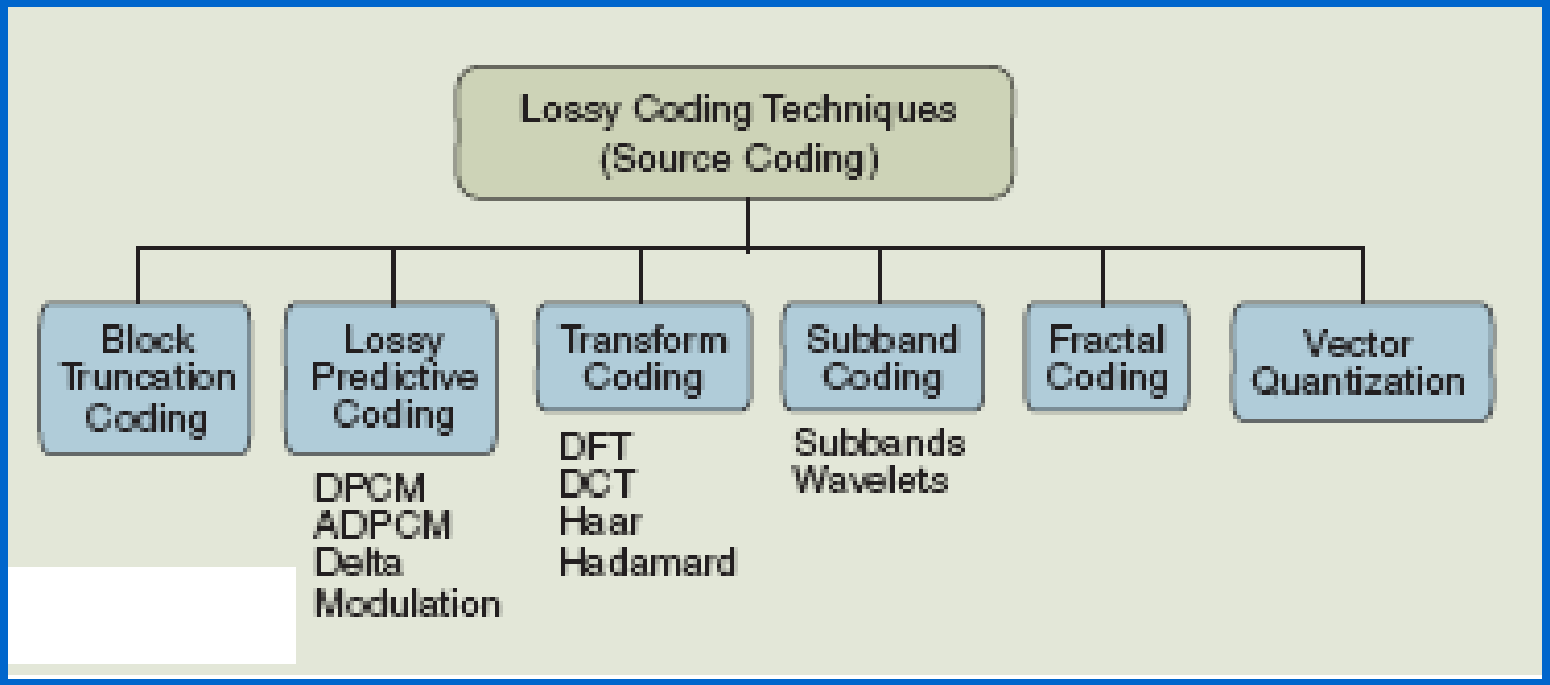
- Process each bit plane individually.

(1) Decompose an image into a series of binary images.

(2) Compress each binary image (e.g., using run-length coding)



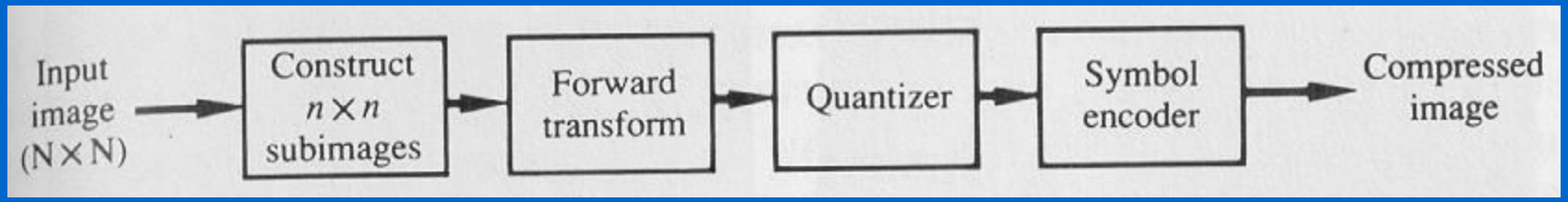
Lossy Methods - Taxonomy



Lossy Compression

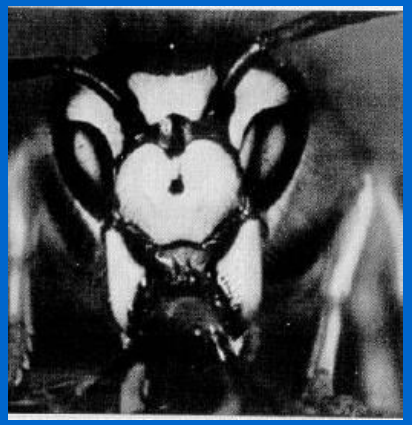
- Transform the image into some other domain to reduce interpixel redundancy.

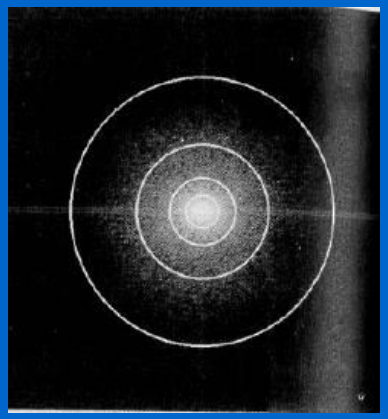
~ $(N/n)^2$ subimages



Example: Fourier Transform

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





Note that the magnitude of the FT decreases, as *u*, *v* increase!

K << N

$$\hat{f}(x,y) = \frac{1}{N}\sum_{u=0}^{K-1}\sum_{v=0}^{K-1} F(u,v)e^{\frac{j2\pi(ux+vy)}{N}}, \quad x,y=0,1,\ldots,N-1$$

$\sum_{x,y}(\hat{f}(x,y) - f(x,y))^2$ is very small !!

Transform Selection

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v) h(x, y, u, v)$$

- T(u,v) can be computed using various transformations, for example:
  - DFT
  - DCT (Discrete Cosine Transform)
  - KLT (Karhunen-Loeve Transformation) or Principal Component Analysis (PCA)

- JPEG using DCT for handling interplixel redundancy.

DCT (Discrete Cosine Transform)

Forward:

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)cos(\frac{(2x+1)u\pi}{2N})cos(\frac{(2y+1)v\pi}{2N}),$$

u, v=0,1,...,N-1

Inverse:

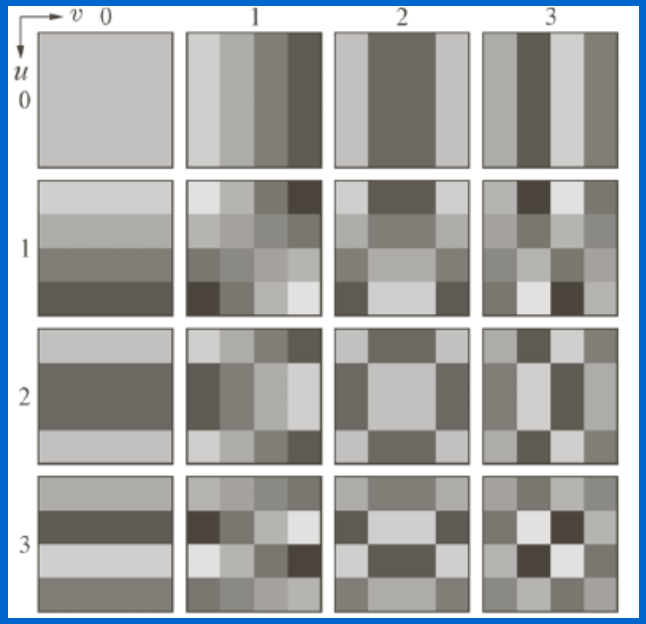
$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)cos(\frac{(2x+1)u\pi}{2N})cos(\frac{(2y+1)v\pi}{2N}),$$

x, y=0,1,...,N-1

$$\alpha(u) = \begin{cases} \sqrt{1/N} & \text{if u=0} \\ \sqrt{2/N} & \text{if u>0} \end{cases} \qquad \alpha(v) = \begin{cases} \sqrt{1/N} & \text{if v=0} \\ \sqrt{2/N} & \text{if v>0} \end{cases}$$

DCT (cont'd)

- Basis functions for a 4x4 image (i.e., cosines of different frequencies).



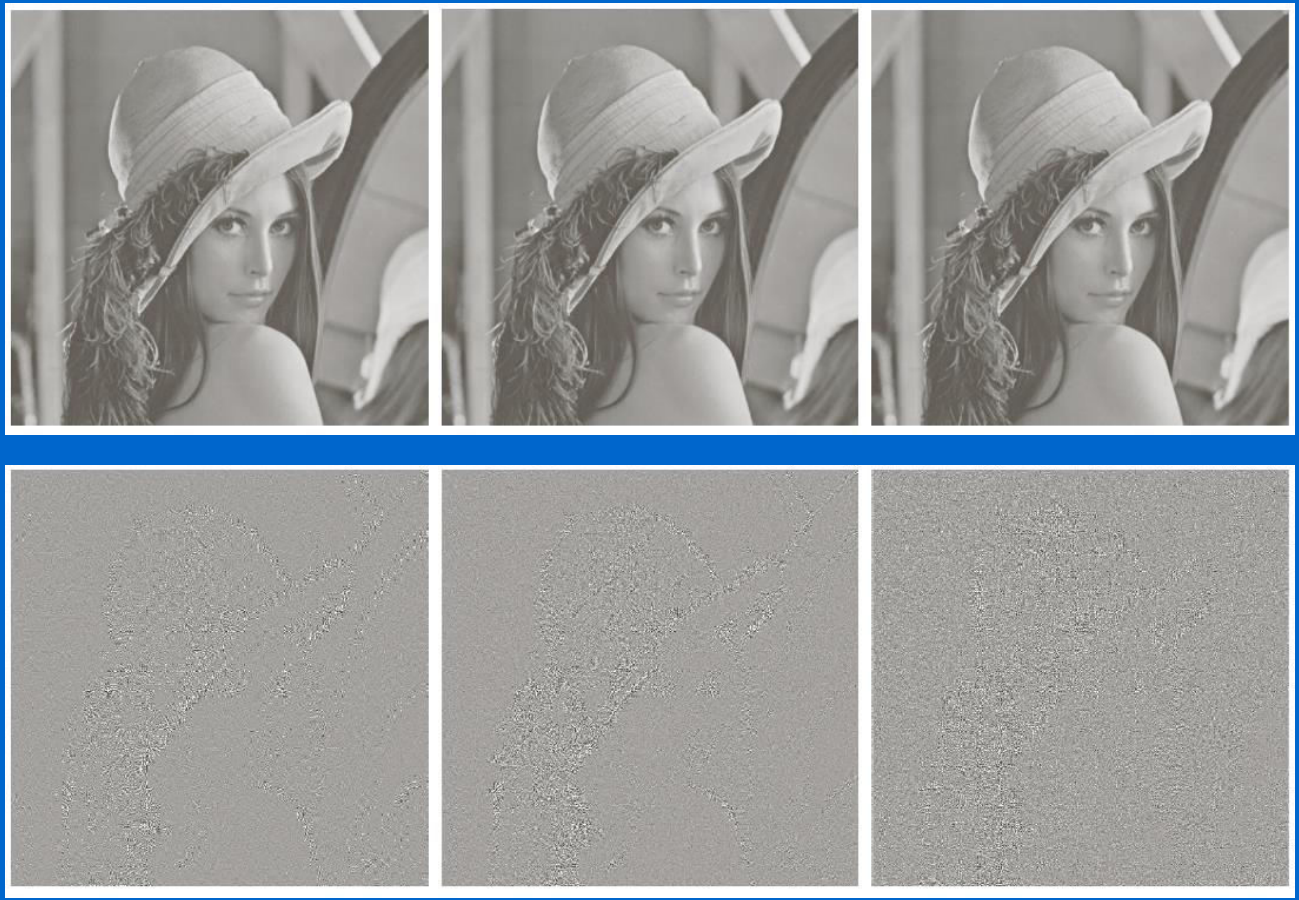
DCT (cont'd)

Using
8 x 8 sub-images
yields 64 coefficients
per sub-image.

Reconstructed images
by truncating
50% of the
coefficients

More compact
transformation

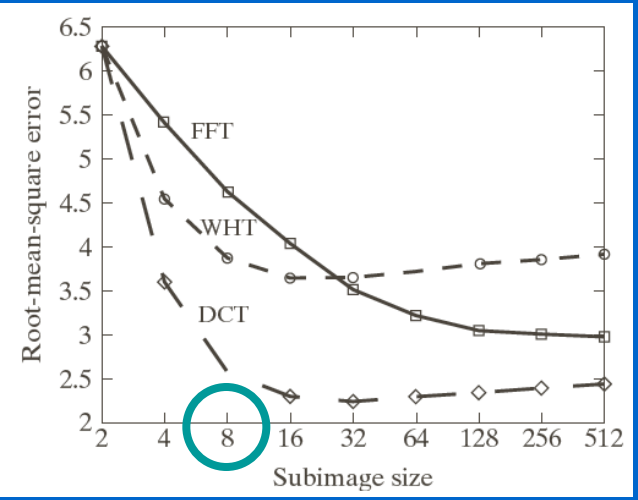
DFT	WHT	DCT



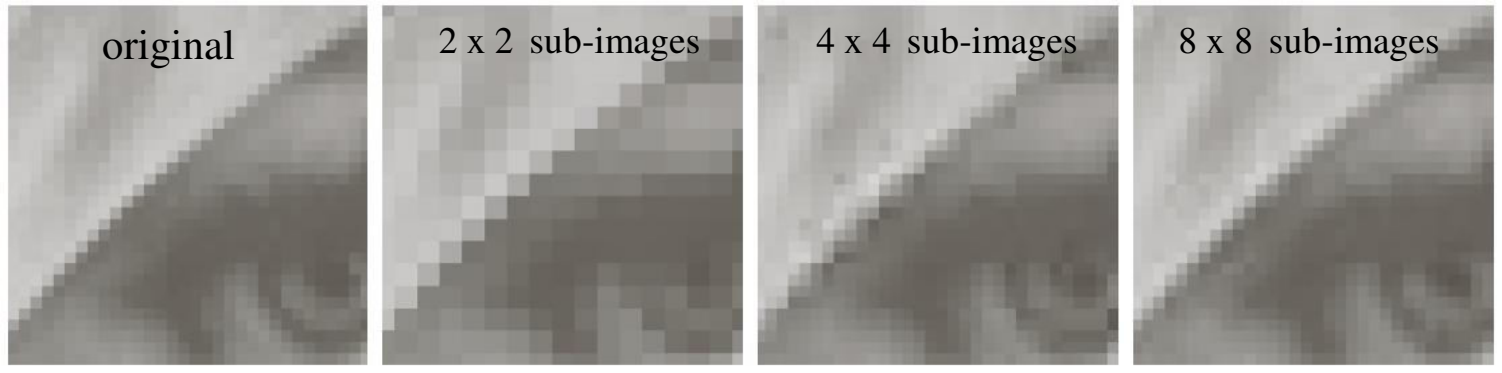
RMS error:	2.32	1.78	1.13

DCT (cont'd)

- Sub-image size selection:



Reconstructions

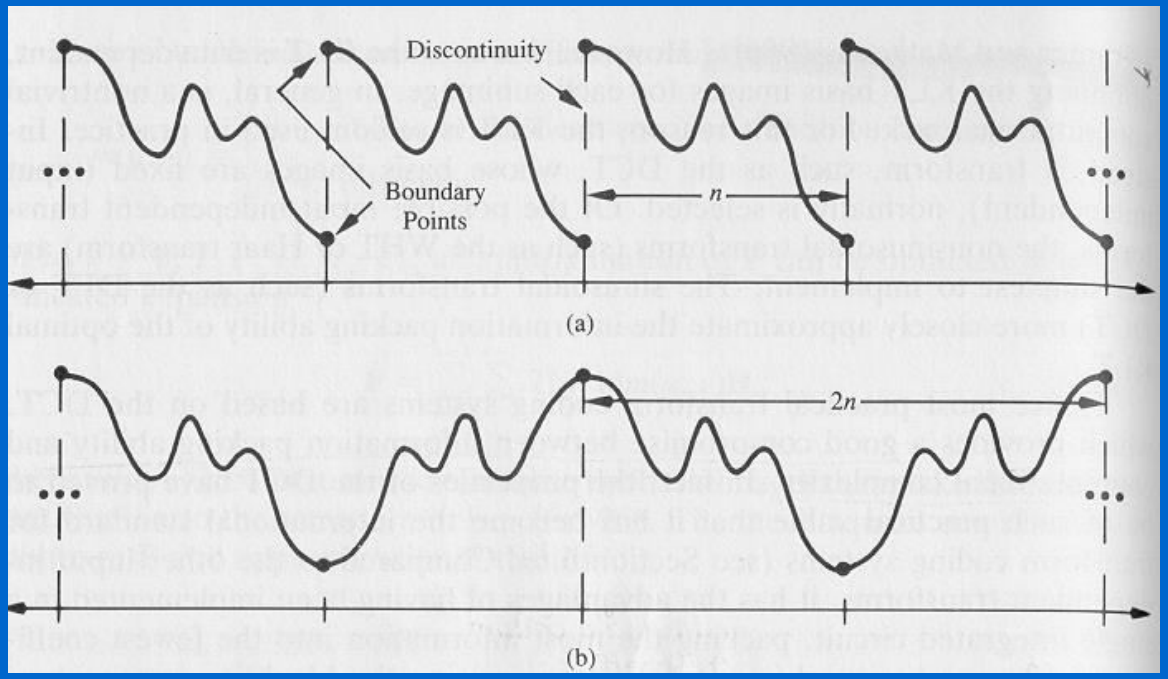


DCT (cont'd)

- DCT minimizes "blocking artifacts" (i.e., boundaries between subimages do not become very visible).

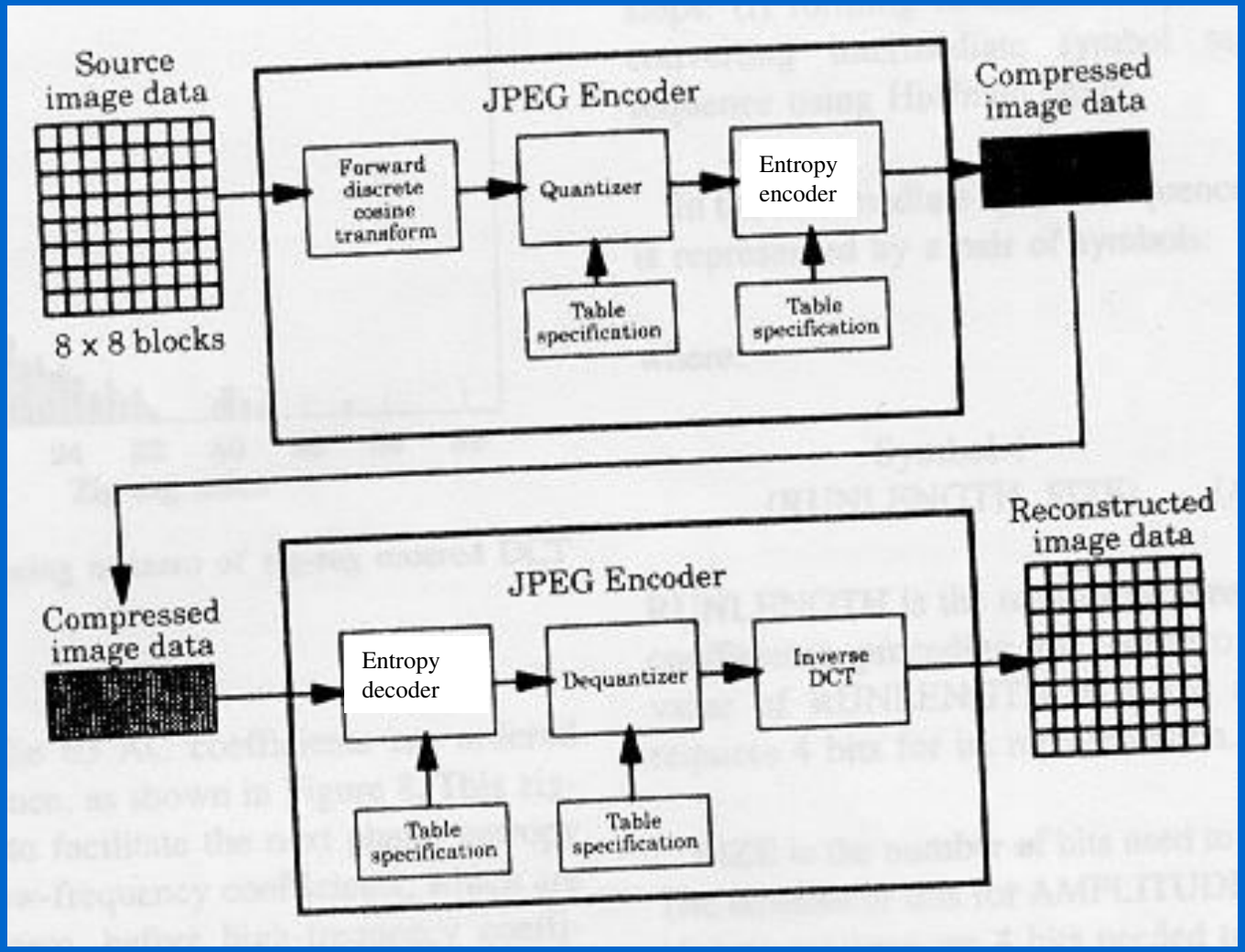
DFT
has n-point periodicity

DCT
has 2n-point periodicity



JPEG Compression

Accepted as an internatio nal image compressi on standard in 1992.



JPEG - Steps

1. Divide image into 8x8 subimages.

For each subimage do:

2. Shift the gray-levels in the range [-128, 127]

3. Apply DCT → 64 coefficients

   1 DC coefficient: F(0,0)

   63 AC coefficients: F(u,v)

Example

(a) Original 8 8 block

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

(b) Shifted block

12	16	19	12	11	27	51	47
16	24	12	19	12	20	39	51
24	27	8	39	35	34	24	44
40	17	28	32	24	27	8	32
34	20	28	20	12	8	19	34
19	39	12	27	27	12	8	34
8	28	-5	39	34	16	12	19
20	27	8	27	24	19	19	8

(c) Block after FDCT Eqn. (5)

185	-17	14	-8	23	-9	-13	-18
20	-34	26	-9	-10	10	13	6
-10	-23	-1	6	-18	3	-20	0
-8	-5	14	-14	-8	-2	-3	8
-3	9	7	1	-11	17	18	15
3	-2	-18	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-1
0	-7	-2	1	1	4	-6	0

[-128, 127]

(non-centered spectrum)

JPEG Steps

4. Quantize the coefficients (i.e., reduce the amplitude of coefficients that do not contribute a lot).

$$C_q(u,v) = Round[\frac{C(u,v)}{Q(u,v)}]$$

Q(u,v): quantization table

Example

- Quantiza

(d) Quantization table (quality = 2)

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

(best - low compression)

(worst - high compression)

Example (cont'd)

(c) Block after FDCT Eqn. (5)

185	-17	14	-8	23	-9	-13	-18
20	-34	26	-9	-10	10	13	6
-10	-23	-1	6	-18	3	-20	0
-8	-5	14	-14	-8	-2	-3	8
-3	9	7	1	-11	17	18	15
3	-2	-18	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-1
0	-7	-2	1	1	4	-6	0

(d) Quantization table (quality = 2)

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Quantization

(e) Block after quantization Eqn. (6)

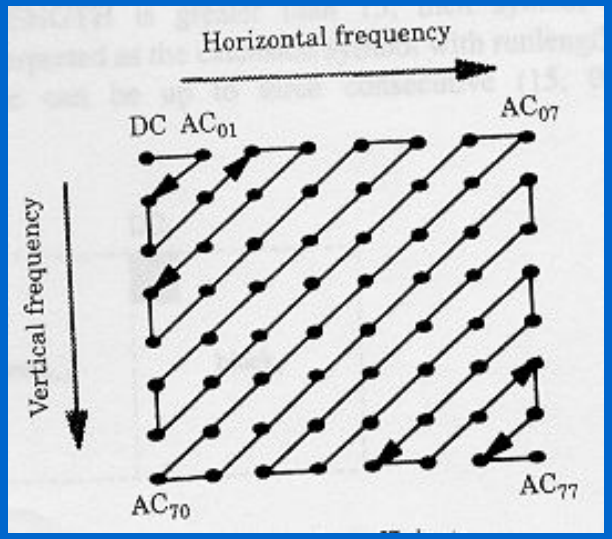
61	-3	2	0	2	0	0	-1
4	-4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

JPEG Steps (cont'd)

5. Order the coefficients using **zig-zag** ordering
   - Creates long runs of zeros (i.e., ideal for run-length encoding)

(e) Block after quantization Eqn. (6)

61	-3	2	0	2	0	0	-1
4	-4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,-1,0,0,-1,0,0,
0,0,-1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

JPEG Steps (cont'd)

6. Encode coefficients:

6.1 Form "intermediate" symbol sequence.

6.2 Encode "intermediate" symbol sequence into a binary sequence.

Intermediate Symbol Sequence – DC coeff

(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,-1,0,0,-1,0,0,
0,0,-1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

(g) Intermediate synbol sequence

(6)(61),(0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2),
[redacted](5,2)(2),(3,1)(1),(6,1)(-1),(2,1)(-1),(4,1)(-1),(7,1)(-1),(0,0)

	symbol_1 (SIZE)	symbol_2 (AMPLITUDE)
DC	(6)	(61)

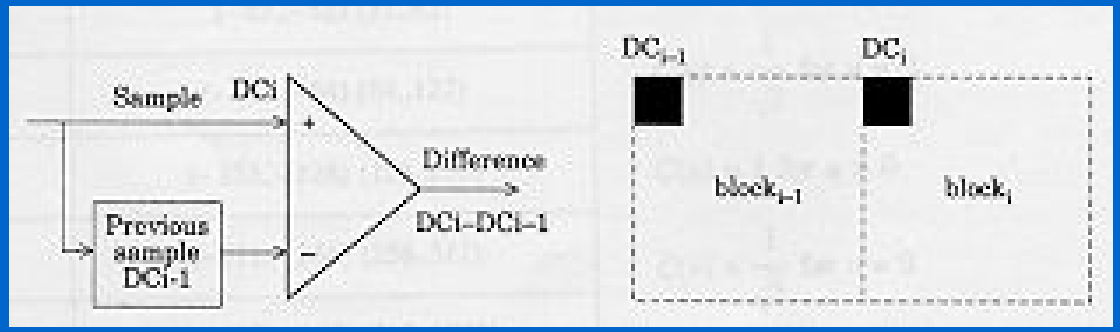
SIZE: # bits need to encode the coefficient

DC Coefficient Encoding

symbol_1 (SIZE)

symbol_2 (AMPLITUDE)

predictive coding:



Intermediate Symbol Sequence – AC coeff

(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,-1,0,0,-1,0,0,
0,0,-1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

(g) Intermediate synbol sequence

(6)(61),(0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2),
[redacted](5,2)(2),(3,1)(1),(6,1)(-1),(2,1)(-1),(4,1)(-1),(7,1)(-1),(0,0)

	symbol_1 (RUN-LENGTH, SIZE)	symbol_2 (AMPLITUDE)	end of block
AC	(0, 2)	(-3)	

RUN-LENGTH: run of zeros preceding coefficient
SIZE: # bits for encoding the amplitude of coefficient

<u>Note</u>: If RUN-LENGTH > 15, use symbol (15,0) ,

Example: AC Coefficients Encoding

Symbol_1

(Variable Length Code (VLC))

(Runlength, size)	Code word
(0,,0) EOB	1010
(0,1)	00
(0,2)	01
(0,3)	100
(1,2)	11011
(2,1)	11100
(3,1)	111010
(4,1)	111011
(5,2)	11111110111
(6,1)	1111011
(7,1)	11111010

Symbol_2

(Variable Length Integer (VLI))

# bits	Amplitude range
1	(−1, 1)
2	(−3, −2) (2,3)
3	(−7..−4) (4..7)
4	(−15..−8) (8..15)
5	(−31..−16) (16..31)
6	(−63..−32) (32..63)
7	(−127..−64) (64..127)
8	(−255..−128) (128..255)
9	(−511..−256) (256..511)
10	(−1023..−512) (512..1023)

(1,4) (12) → (111110110 1100)

VLC VLI

Final Symbol Sequence

(g) Intermediate synbol sequence

(6)(61),(0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2),
[redacted](5,2)(2),(3,1)(1),(6,1)(-1),(2,1)(-1),(4,1)(-1),(7,1)(-1),(0,0)

(e) Encoded bit sequence (total 98 bits)

11101111010010010000010001101101101110010111111011
11011101011111011011100011101101111101001010

What is the effect of the "Quality" parameter?

(58k bytes)

(21k bytes)

(8k bytes)

lower compression → higher compression

$$1 \leq quality \leq 25$$

What is the effect of the "Quality" parameter? (cont'd)

Table 6. Results of JPEG Compression for Grayscale Image 'Lisa' (320 ×240 pixels)

Quality factors	Original number of bits	Compressed number of bits	Compression ratio (Cr)	Bits/pixel (Nb)	RMS error
1	512,000	48,021	10·66	0·75	2·25
2	512,000	30,490	16·79	0·48	2·75
4	512,000	20,264	25·27	0·32	3·43
8	512,000	14,162	36·14	0·22	4·24
15	512,000	10,479	48·85	0·16	5.36
25	512,000	9,034	56·64	0·14	6·40

Effect of Quantization: homogeneous 8 x 8 block

An 8 × 8 block from the Y image of 'Lena'

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

$f(i,j)$

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

$F(u,v)$

Fig. 9.2: JPEG compression for a smooth image block.

Effect of Quantization: homogeneous 8 x 8 block (cont'd)

Quantized

32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\hat{F}(u,v)$

De-quantized

512	66	-10	0	0	0	0	0
-12	0	0	0	0	0	0	0
-14	0	16	0	0	0	0	0
-14	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\tilde{F}(u,v)$

Effect of Quantization: homogeneous 8 x 8 block (cont'd)

Reconstructed

199	196	191	186	182	178	177	176
201	199	196	192	188	183	180	178
203	203	202	200	195	189	183	180
202	203	204	203	198	191	183	179
200	201	202	201	196	189	182	177
200	200	199	197	192	186	181	177
204	202	199	195	190	186	183	181
207	204	200	194	190	187	185	184

Original

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

Error is low!

1	6	-2	2	7	-3	-2	-1
-1	4	2	-4	1	-1	-2	-3
0	-3	-2	-5	5	-2	2	-5
-2	-3	-4	-3	-1	-4	4	8
0	4	-2	-1	-1	-1	5	-2
0	0	1	3	8	4	6	-2
1	-2	0	5	1	1	4	-6
3	-4	0	6	-2	-2	2	2

Effect of Quantization: non-homogeneous 8 x 8 block

Another 8×8 block from the Y image of 'Lena'

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

$f(i,j)$

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

$F(u,v)$

Effect of Quantization: non-homogeneous 8 x 8 block (cont'd)

Quantized

-5	-4	9	-5	2	1	1	0
-11	-5	-2	0	1	0	0	-1
3	-6	4	0	-3	-1	0	1
0	1	-1	0	1	0	0	0
0	0	-1	0	0	1	0	0
0	-1	1	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\hat{F}(u,v)$

De-quantized

-80	-44	90	-80	48	40	51	0
-132	-60	-28	0	26	0	0	-55
42	-78	64	0	-120	-57	0	56
0	17	-22	0	51	0	0	0
0	0	-37	0	0	109	0	0
0	-35	55	-64	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\tilde{F}(u,v)$

Effect of Quantization: non-homogeneous 8 x 8 block (cont'd)

Reconstructed

70	60	106	94	62	103	146	176
85	101	85	75	102	127	93	144
98	99	92	102	74	98	89	167
132	53	111	180	55	70	106	145
173	57	114	207	111	89	84	90
164	123	131	135	133	92	85	162
141	159	169	73	106	101	149	224
150	141	195	79	107	147	210	153

Original:

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

Error is high!

0	10	-6	-24	25	-16	4	11
0	-1	11	4	-15	27	-6	-31
2	-14	24	-23	-4	-11	-3	29
4	16	-24	20	24	1	11	-49
-12	13	-27	-7	-8	-18	12	23
-3	0	16	-2	-20	21	0	-1
5	-12	6	27	-3	2	14	-37
6	5	-6	-9	6	14	-47	44

Case Study: Fingerprint Compression

- FBI is digitizing fingerprints at 500 dots per inch with 8 bits of grayscale resolution.
- A single fingerprint card turns into about 10 MB of data!

A sample fingerprint image

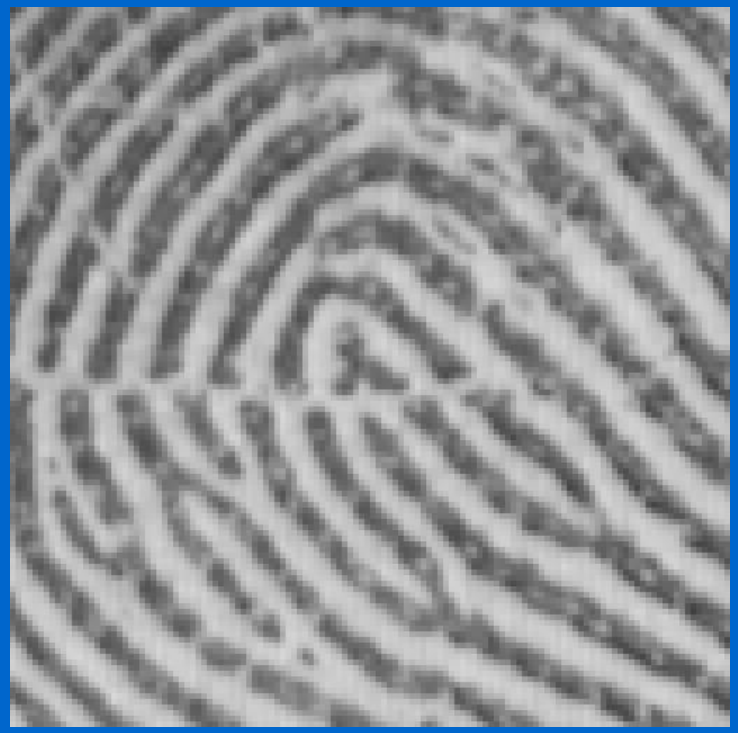
768 x 768 pixels =589,824 bytes

WSQ Fingerprint Compression

- An image coding standard for digitized fingerprints employing the Discrete Wavelet Transform (*Wavelet/Scalar Quantization or WSQ*).

- Developed and maintained by:
  - FBI
  - Los Alamos National Lab (LANL)
  - National Institute for Standards and Technology (NIST)

Need to Preserve Fingerprint Details



The "white" spots in the middle of the black ridges are *sweat pores* and they are admissible points of identification in court.

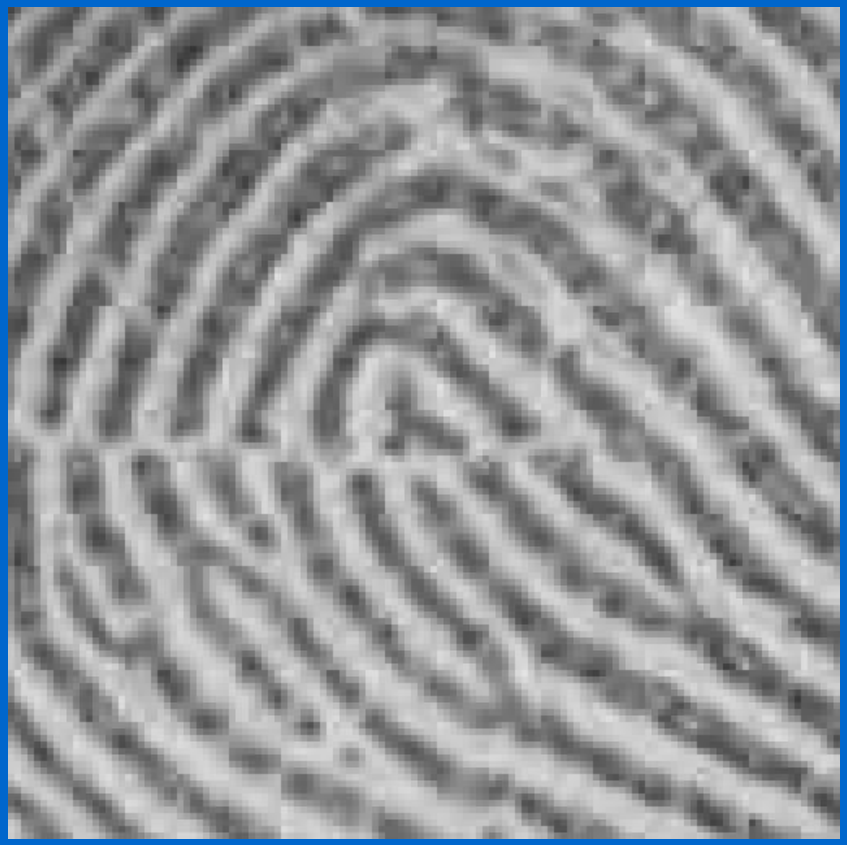
<u>These details are just a couple pixels wide!</u>

What compression scheme should be used?

- Lossless or lossy compression?
- In practice lossless compression methods haven't done better than 2:1 on fingerprints!
- Does JPEG work well for fingerprint compression?

Results using JPEG compression

file size 45853 bytes
compression ratio: 12.9



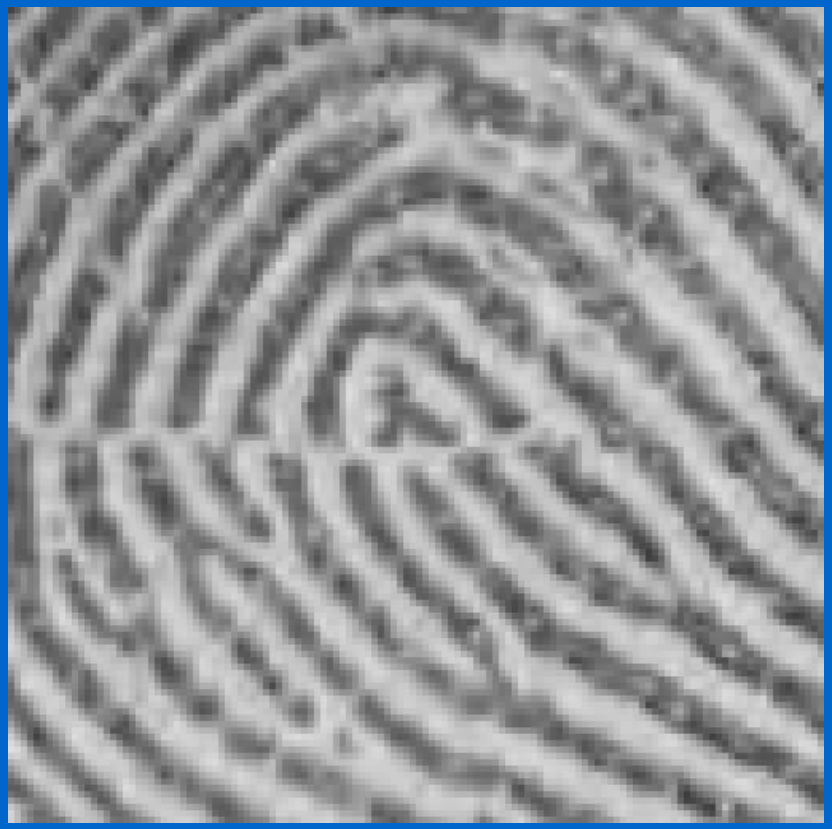
Fine details have been lost.

Image has an artificial "blocky" pattern superimposed on it.

Artifacts will affect the performance of fingerprint recognition.

Results using WSQ compression

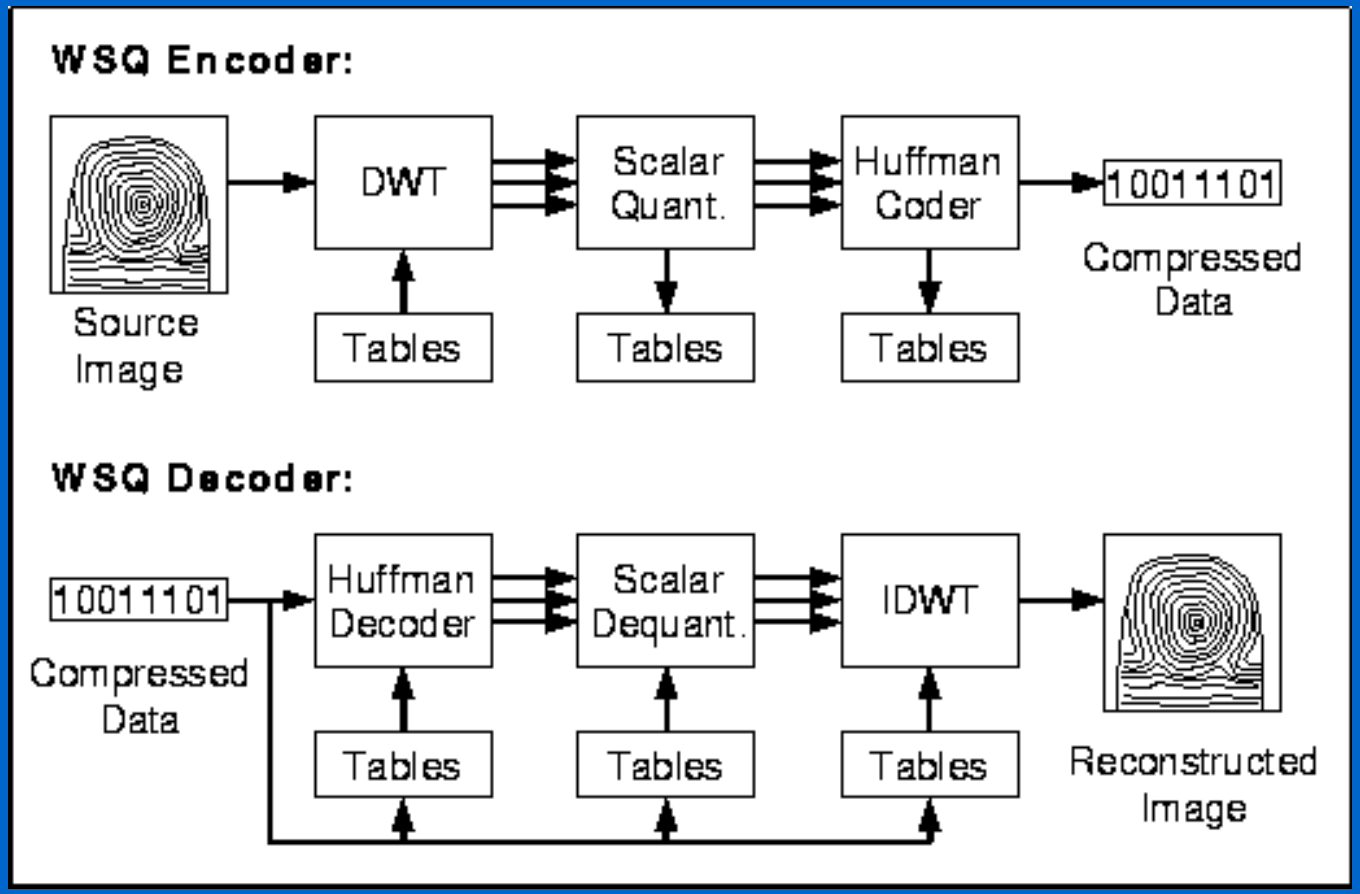
file size 45621 bytes
compression ratio: 12.9



Fine details are better preserved.

No “blocky” artifacts.

WSQ Algorithm



Compression ratio

- FBI’s target bit rate is around 0.75 bits per pixel (bpp)
- This corresponds to a compression ratio of 8/0.75=10.7
- Target bit rate can set via a parameter, similar to the "quality" parameter in JPEG.

Varying compression ratio (cont'd)

Original image 768 x 768 pixels (589824 bytes)

Varying compression ratio (cont'd)
0.9 bpp compression

WSQ image, file size 47619 bytes, compression ratio 12.4

JPEG image, file size 49658 bytes, compression ratio 11.9

Varying compression ratio (cont'd)
0.75 bpp compression

WSQ image, file size 39270 bytes
compression ratio 15.0

JPEG image, file size 40780 bytes,
compression ratio 14.5

Varying compression ratio (cont'd)
0.6 bpp compression

WSQ image, file size 30987 bytes, compression ratio 19.0

JPEG image, file size 30081 bytes, compression ratio 19.6

JPEG Modes

- JPEG supports several different modes
  - Sequential Mode
  - Progressive Mode
  - Hierarchical Mode
  - Lossless Mode
- The default mode is “sequential”
  - Image is encoded in a single scan (left-to-right, top-to-bottom).

Progressive JPEG

- Image is encoded in multiple scans, in order to produce a quick, rough decoded image when transmission time is long.

Sequential

Progressive

Progressive JPEG (cont'd)

- Each scan encodes a subset of DCT coefficients.

- We'll examine the following algorithms:
  (1) Progressive spectral selection algorithm
  (2) Progressive successive approximation algorithm
  (3) Combined progressive algorithm

Progressive JPEG (cont'd)

(1) Progressive spectral selection algorithm

- Group DCT coefficients into several spectral bands
- Send low-frequency DCT coefficients first
- Send higher-frequency DCT coefficients next

Band 1: DC coefficient only
Band 2: AC_1 and AC_2 coefficients
Band 3: AC_3, AC_4, AC_5, AC_6, coefficients
Band 4: AC_7....AC_{63}, coefficients

Example

Table 8. Progressive spectral selection JPEG. (Image 'Cheetah': 320 × 240 pixels –>512,000 bits)

Scan number	Bits transmitted	Compression ratio	Bits/pixel	RMS error
1	29,005	17·65	0·45	19·97
2	37,237	7·73	1·04	13·67
3	71,259	3·72	2·15	7·90
4	32,489	3·01	2·66	4·59
Sequential JPEG	172,117	2·97	2·69	4·59

Progressive JPEG (cont'd)

(2) Progressive successive approximation algorithm

- Send all DCT coefficients but with lower precision.
- Refine DCT coefficients in later scans.

Band 1: All DCT coefficients (divided by four)
Band 2: All DCT coefficients (divided by two)
Band 3: All DCT coefficients (full resolution)

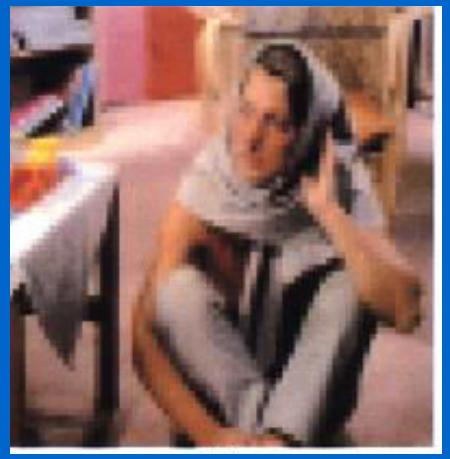
Example

Table 9. Progressive successive approximation JPEG. (Image 'Cheetah': 320 × 240 pixels –> 512,000 bits)

Scan number	Bits transmitted	Compression ratio	Bits/pixel	RMS error
1	26,215	19·53	0·41	22·48
2	34,506	8·43	0·95	12·75
3	63,792	4·11	1.95	7·56
4	95,267	2·33	2·43	4·59
Sequential JPEG	172,117	2·97	2·69	4·59

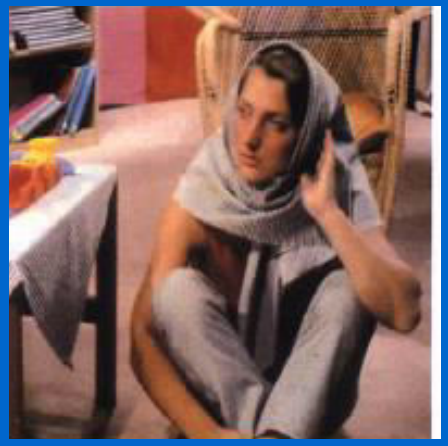
Example

after 0.9s

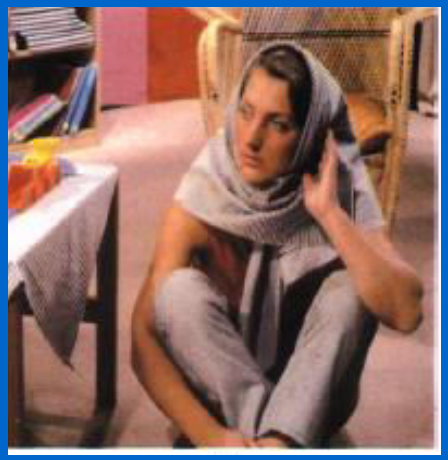


after 1.6s

after 3.6s



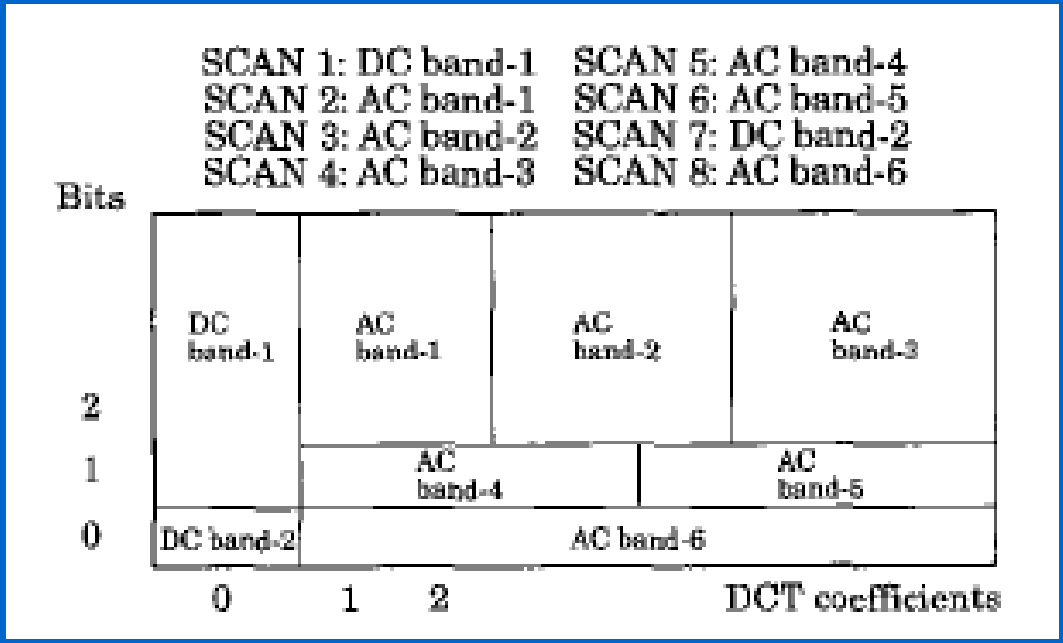
after 7.0s



Progressive JPEG (cont'd)

(3) Combined progressive algorithm

– Combines spectral selection and successive approximation.



Hierarchical JPEG

- Hierarchical mode encodes the image at different resolutions.
- Image is transmitted in multiple passes with increased resolution at each pass.

Hierarchical JPEG (cont'd)

N/4 x N/4

N/2 x N/2

N x N

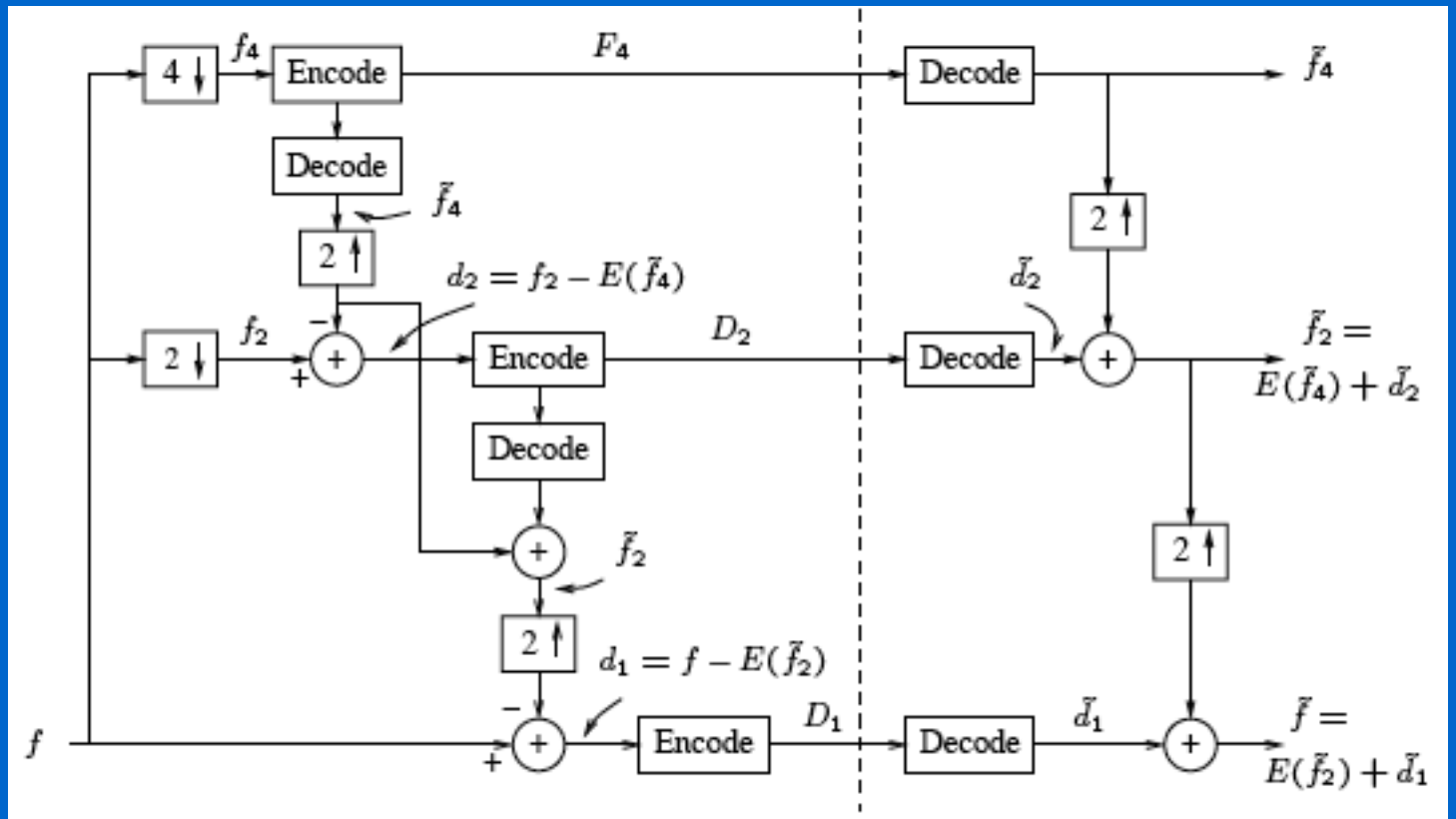


Fig. 9.5: Block diagram for Hierarchical JPEG.

More Methods …

- See “Image Compression Techniques”, IEEE Potentials, February/March 2001