

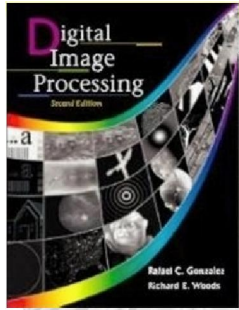
Digital Image Processing

2020-2021

المرحلة الثالثة / الفصل الدراسي الثاني

استاذ المادة:

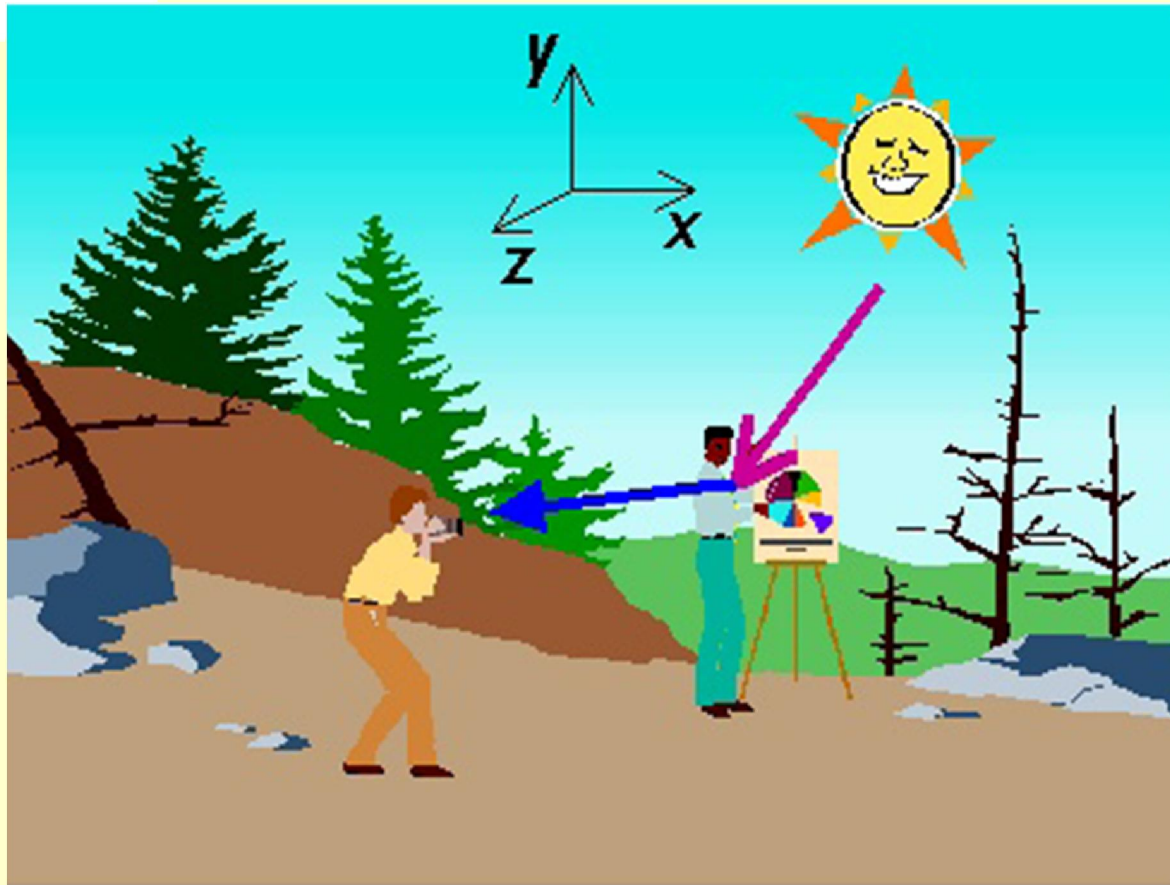
أ.م.د. ناصر حسين سلمان



Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

IMAGE FORMATION



The Electromagnetic Spectrum

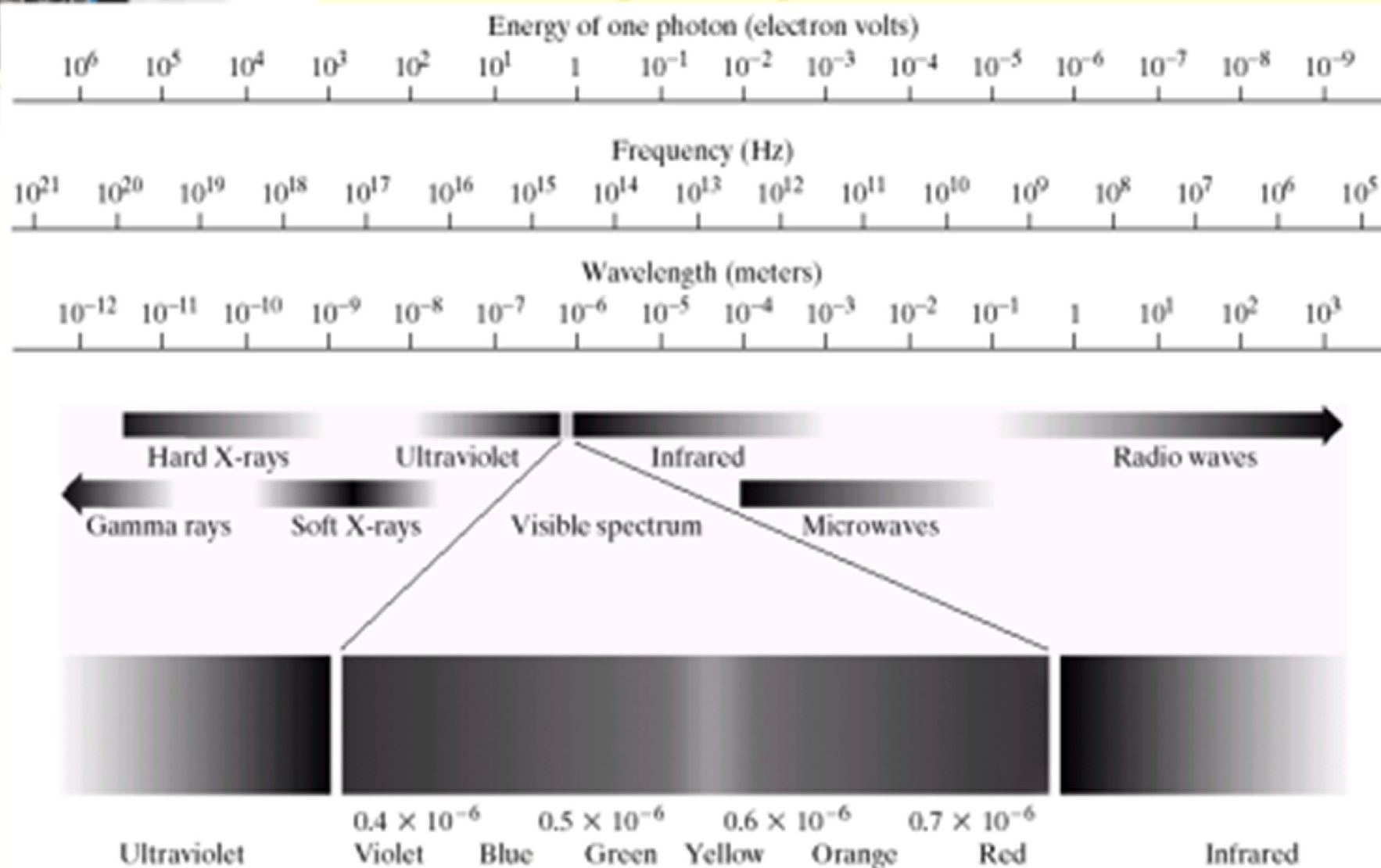
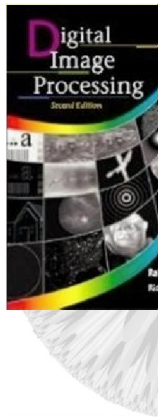
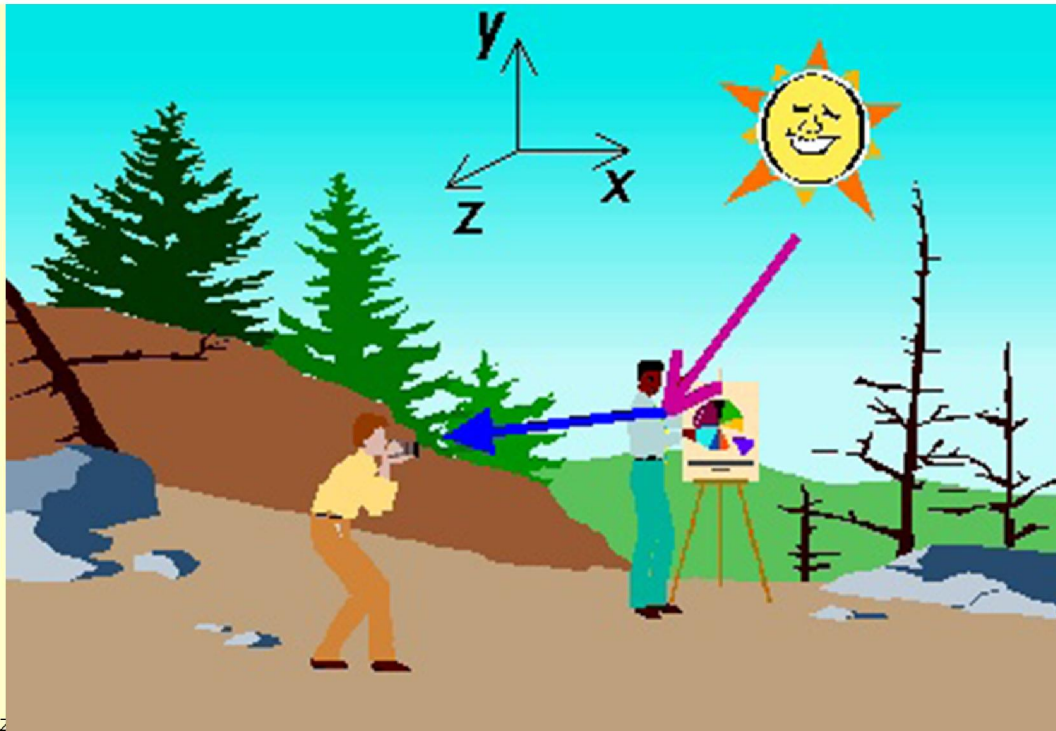
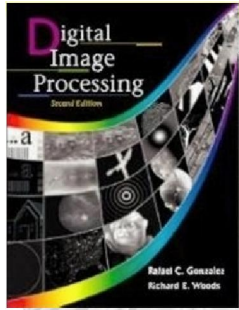


FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum.



- For natural images we need a light source (λ : wavelength of the source) ^(?).
 - $E(x, y, z, \lambda)$: incident light on a point (x, y, z world coordinates of the point)
- Each point in the scene has a reflectivity function.
 - $r(x, y, z, \lambda)$: reflectivity function
- Light reflects from a point and the reflected light is captured by an imaging device.
 - $c(x, y, z, \lambda) = E(x, y, z, \lambda) \times r(x, y, z, \lambda)$: reflected light.





Digital Image Processing, 2nd ed.

www.imageprocessingbook.com



→ $E(x, y, z, \lambda)$

→ $c(x, y, z, \lambda) = E(x, y, z, \lambda) \cdot r(x, y, z, \lambda)$

Camera($c(x, y, z, \lambda)$) =

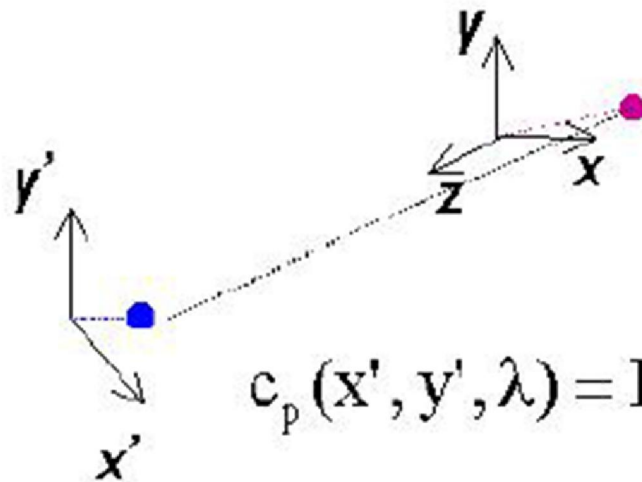


Inside the Camera - Projection

Camera($c(x, y, z, \lambda)$) =



- Projection (\mathcal{P}) from world coordinates (x, y, z) to camera or image coordinates (x', y') [$c_p(x', y', \lambda) = \mathcal{P}(c(x, y, z, \lambda))$].



$$c_p(x', y', \lambda) = \mathcal{P}(c(x, y, z, \lambda))$$

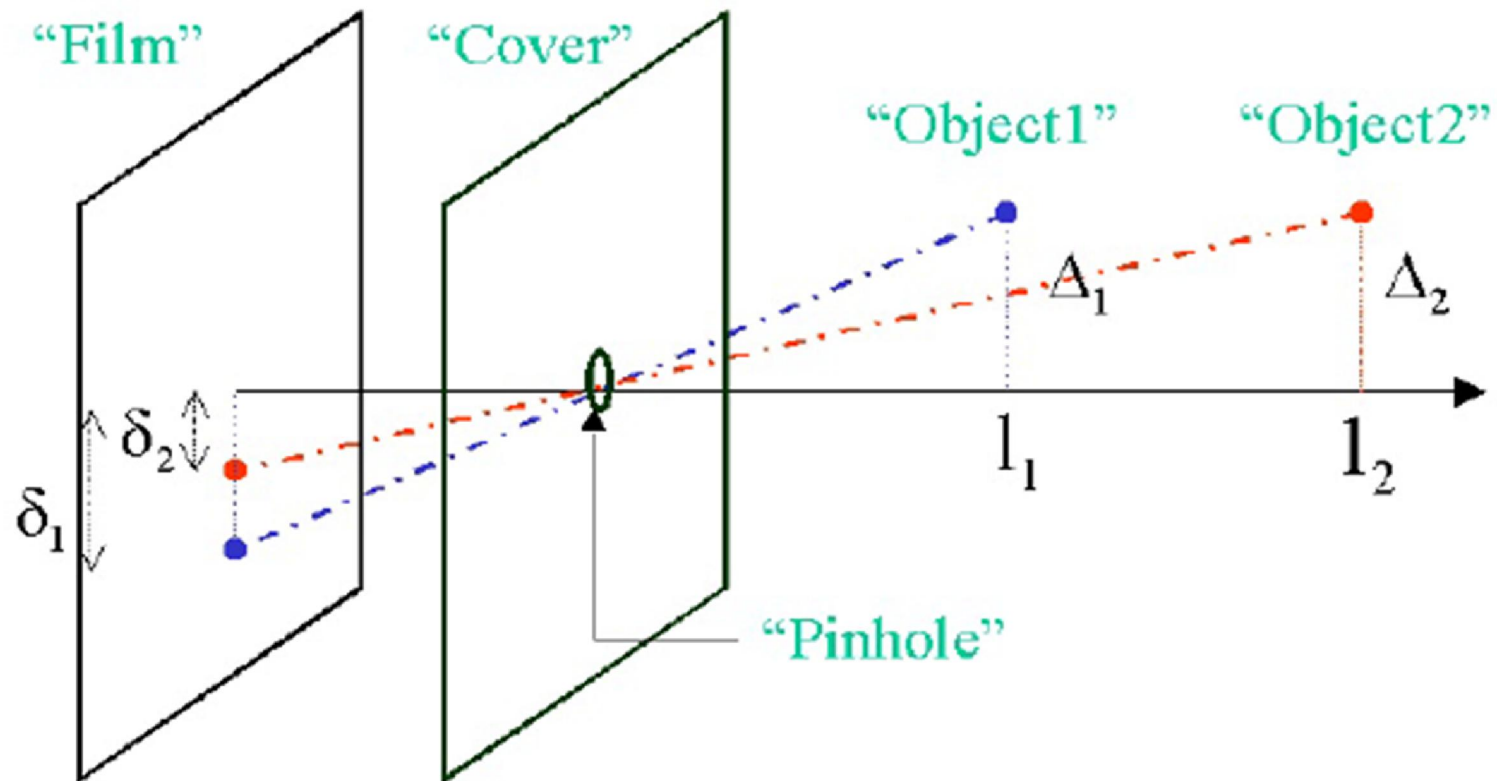


Projections

- There are two types of projections (\mathcal{P}) of interest to us:
 1. Perspective Projection
 - Objects closer to the capture device appear bigger. Most image formation situations can be considered to be under this category, including images taken by camera and the *human eye*.
 2. Ortographic Projection
 - This is “unnatural”. Objects appear the same size regardless of their distance to the “capture device”.
- Both types of projections can be represented via mathematical formulas. Ortographic projection is *easier* and is sometimes used as a mathematical convenience. For more details see [1].

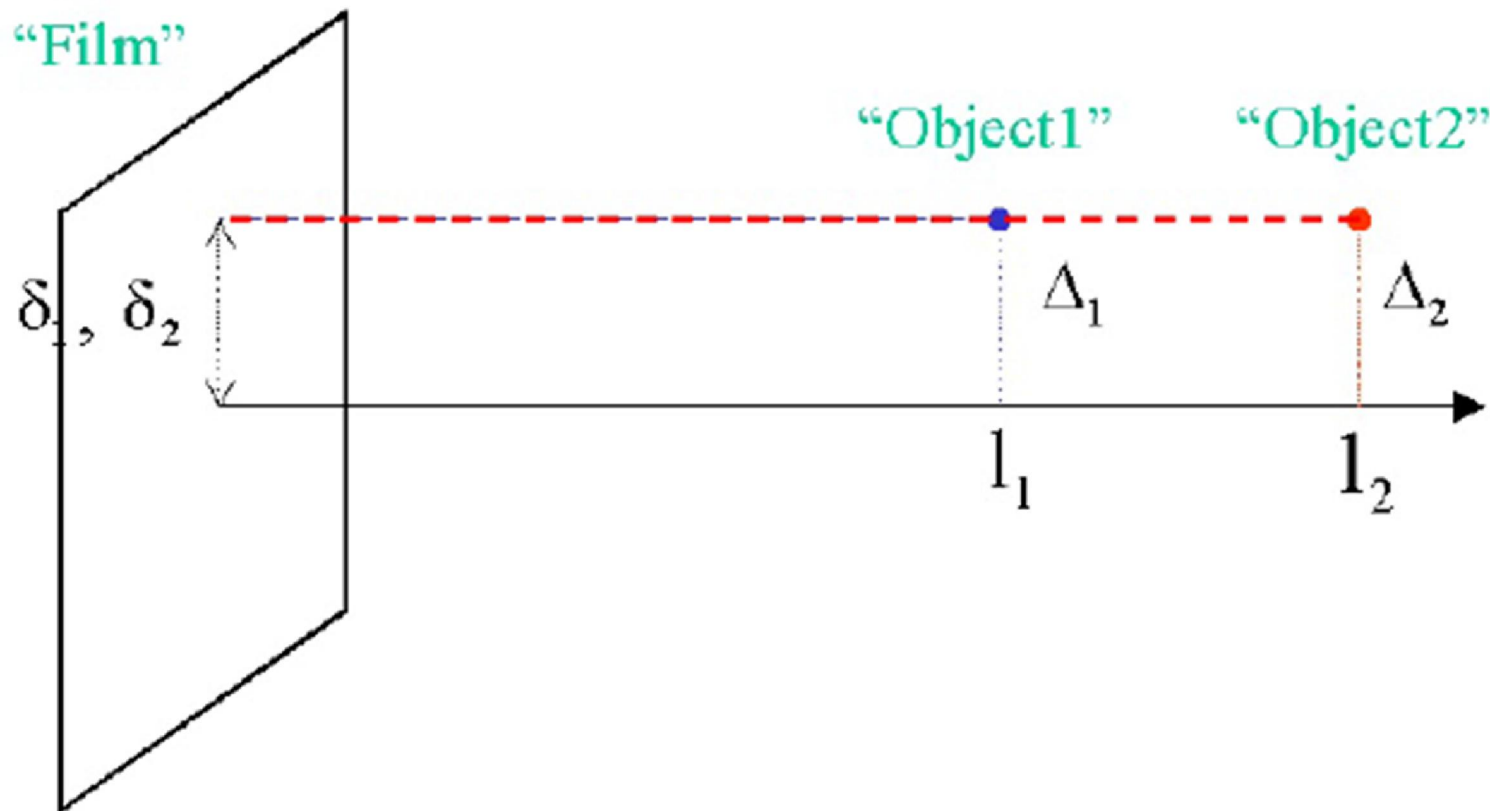


Example - Perspective



- Perspective Projection: $\Delta_1 = \Delta_2$, $l_1 < l_2 \rightarrow \delta_2 < \delta_1$.

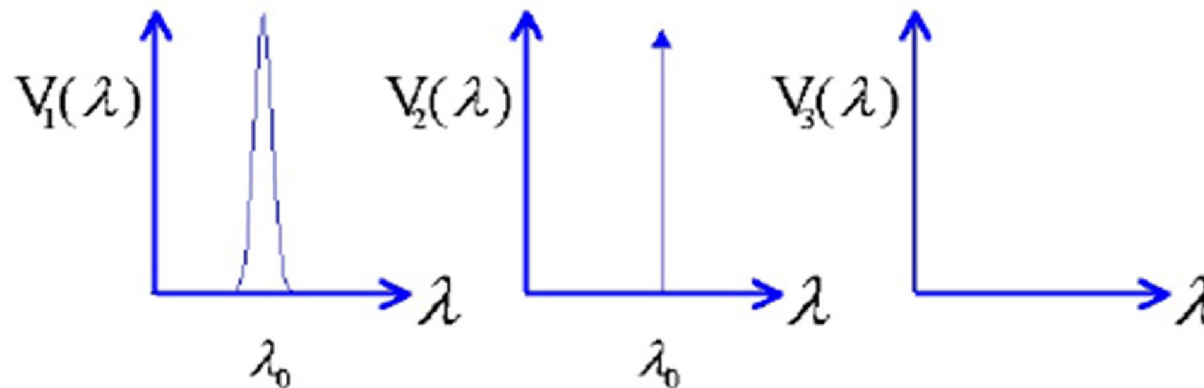
Example - Ortographic



- Ortographic Projection: $\Delta_1 = \Delta_2, l_1 < l_2 \rightarrow \delta_2 = \delta_1$.

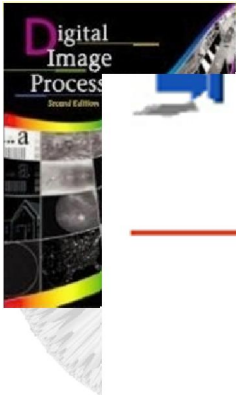
Inside the Camera - Sensitivity

- Once we have $c_p(x', y', \lambda)$ the characteristics of the capture device take over.
- $V(\lambda)$ is the *sensitivity function* of a capture device. Each capture device has such a function which determines how sensitive it is in capturing the range of *wavelengths* (λ) present in $c_p(x', y', \lambda)$.

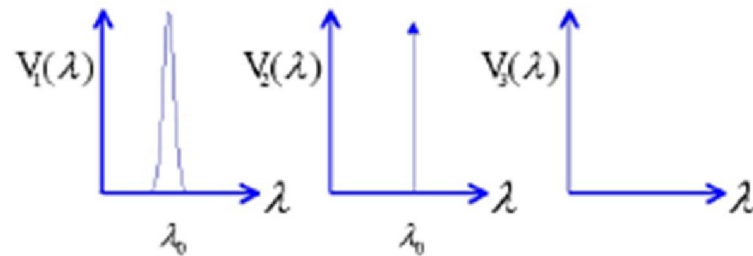


- The result is an “image function” which determines the amount of reflected light that is captured at the camera coordinates (x', y') .

$$f(x', y') = \int c_p(x', y', \lambda) V(\lambda) d\lambda \quad (1)$$



Example



Let us determine the image functions for the above sensitivity functions imaging the same scene:

1. This is the most realistic of the three. Sensitivity is concentrated in a band around λ_0 .

$$f_1(x', y') = \int c_p(x', y', \lambda) V_1(\lambda) d\lambda$$

2. This is an unrealistic capture device which has sensitivity only to a single wavelength λ_0 as determined by the delta function. However there are devices that get close to such “selective” behavior.

$$\begin{aligned} f_2(x', y') &= \int c_p(x', y', \lambda) V_2(\lambda) d\lambda = \int c_p(x', y', \lambda) \delta(\lambda - \lambda_0) d\lambda \quad (?) \\ &= c_p(x', y', \lambda_0) \end{aligned}$$

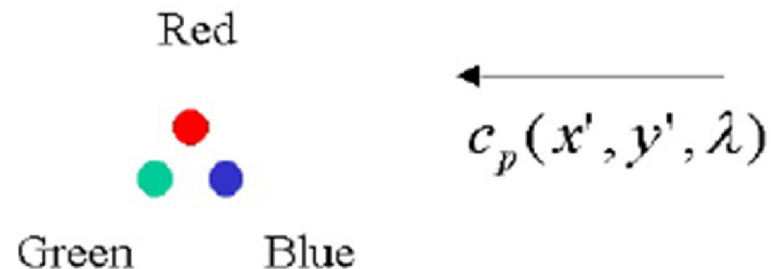
3. This is what happens if you take a picture without taking the cap off the lens of your camera.

$$\begin{aligned} f_3(x', y') &= \int c_p(x', y', \lambda) V_3(\lambda) d\lambda = \int c_p(x', y', \lambda) 0 d\lambda \\ &= 0 \end{aligned}$$



Sensitivity and Color

Camera Sensors



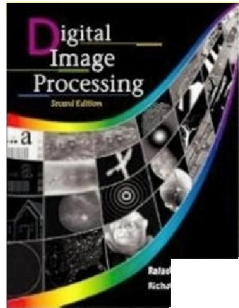
- For a camera that captures color images, imagine that it has *three sensors* at each (x', y') with sensitivity functions tuned to the colors or wavelengths **red**, **green** and **blue**, outputting *three* image functions:

$$f_{\text{R}}(x', y') = \int c_p(x', y', \lambda) V_{\text{R}}(\lambda) d\lambda$$

$$f_{\text{G}}(x', y') = \int c_p(x', y', \lambda) V_{\text{G}}(\lambda) d\lambda$$

$$f_{\text{B}}(x', y') = \int c_p(x', y', \lambda) V_{\text{B}}(\lambda) d\lambda$$

- These three image functions can be used by display devices (such as your monitor or your eye) to show a “color” image.



- The image function $f_c(x', y')$ ($C = R, G, B$) is formed as:

$$f_c(x', y') = \int c_p(x', y', \lambda) V_c(\lambda) d\lambda \quad (2)$$

- It is the result of:

1. Incident light $E(x, y, z, \lambda)$ at the point (x, y, z) in the scene,
2. The reflectivity function $r(x, y, z, \lambda)$ of this point,
3. The formation of the reflected light $c(x, y, z, \lambda) = E(x, y, z, \lambda) \times r(x, y, z, \lambda)$,
4. The **projection** of the reflected light $c(x, y, z, \lambda)$ from the *three* dimensional world coordinates to *two* dimensional camera coordinates which forms $c_p(x', y', \lambda)$,
5. The **sensitivity** function(s) of the camera $V(\lambda)$.

Digital Image Formation

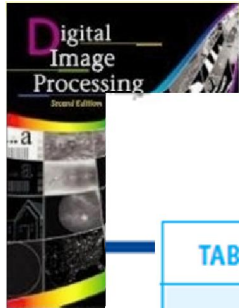
- The **image function** $f_c(x', y')$ is still a function of $x' \in [x'_{min}, x'_{max}]$ and $y' \in [y'_{min}, y'_{max}]$ which vary in a continuum given by the respective intervals.
- The values taken by the image function are real numbers which again vary in a continuum or interval $f_c(x', y') \in [f_{min}, f_{max}]$.
- Digital computers cannot process parameters/functions that vary in a continuum.

- We have to *discretize*:

1. $x', y' \Rightarrow x'_i, y'_j \quad (i = 0, \dots, N - 1, j = 0, \dots, M - 1)$:
2. $f_c(x'_i, y'_j) \Rightarrow \hat{f}_c(x'_i, y'_j)$: Quantization.

54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

When x, y , and the amplitude values of f are all **finite**,
discrete quantities, we call the image a **digital image**.



Digital Image Formation

TABLE 1.1 Types of intervals

	Notation	Set description	Type	Picture
Finite:	(a, b)	$\{x a < x < b\}$	Open	
	$[a, b]$	$\{x a \leq x \leq b\}$	Closed	
	$[a, b)$	$\{x a \leq x < b\}$	Half-open	
	$(a, b]$	$\{x a < x \leq b\}$	Half-open	
Infinite:	(a, ∞)	$\{x x > a\}$	Open	
	$[a, \infty)$	$\{x x \geq a\}$	Closed	
	$(-\infty, b)$	$\{x x < b\}$	Open	
	$(-\infty, b]$	$\{x x \leq b\}$	Closed	
	$(-\infty, \infty)$	\mathbb{R} (set of all real numbers)	Both open and closed	

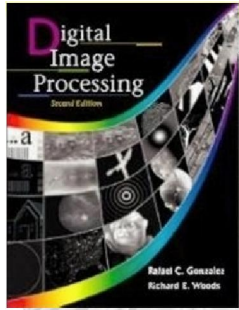
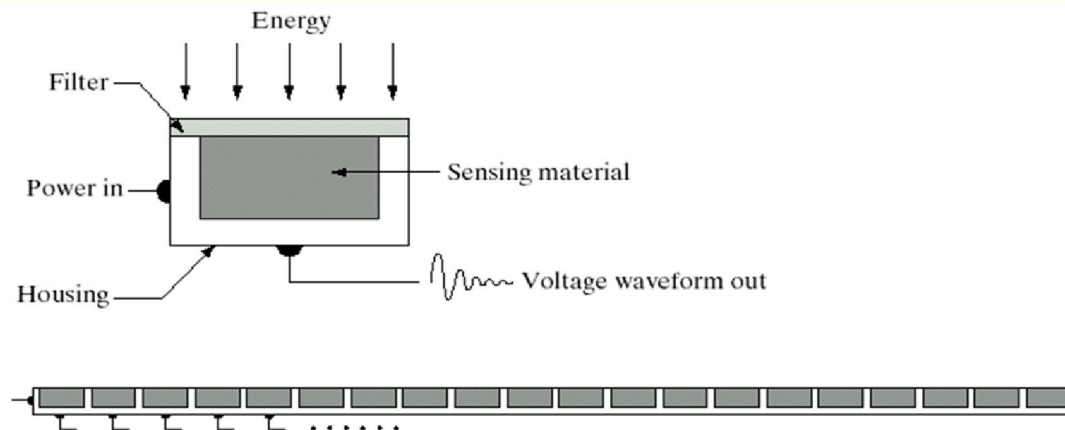


Image Sensing and Acquisition

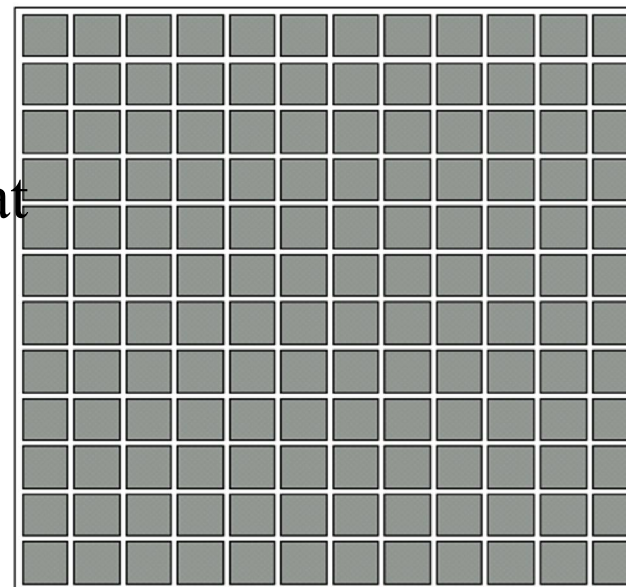
a
b
c

FIGURE 2.12

(a) Single imaging sensor.
(b) Line sensor.
(c) Array sensor.



A digital image is nothing more than data—numbers indicating variations of red, green, and blue at a particular location on a grid of pixels.



54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

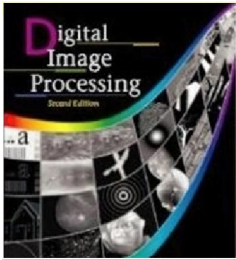


Image acquisition using a single sensor

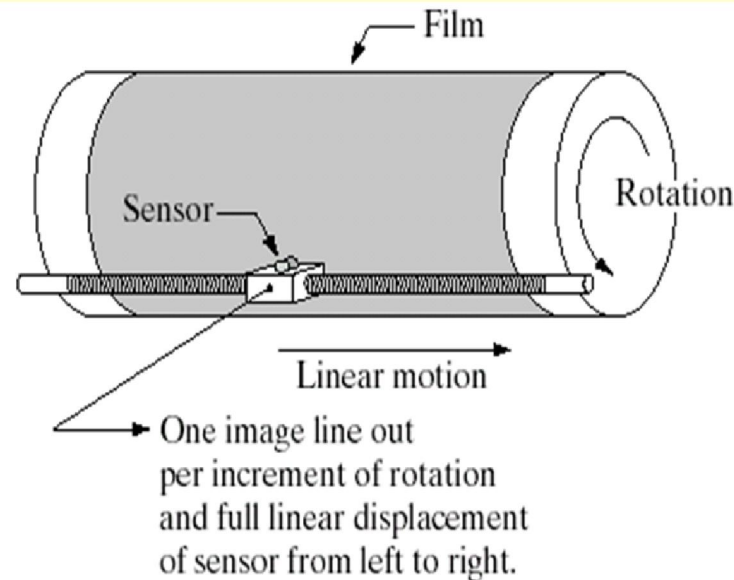
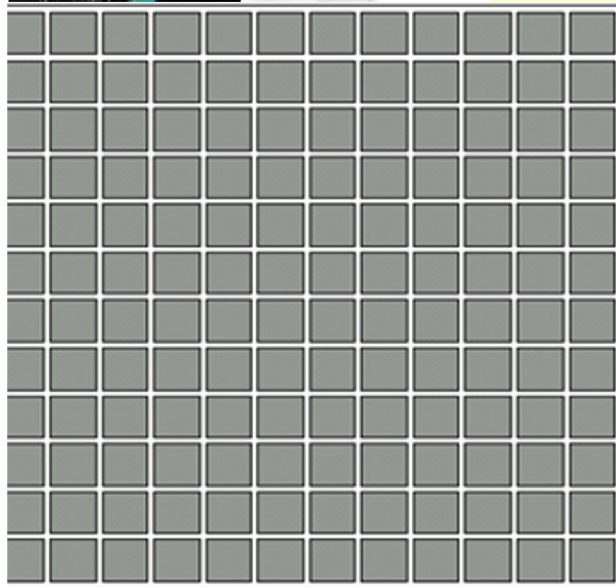


FIGURE 2.13 Combining a single sensor with motion to generate a 2-D image.

This is an arrangement used in **high precision scanning**, where a **film negative** is mounted onto a drum whose mechanical rotation provides displacement in one dimension. The single sensor is mounted on a lead screw that provides motion in perpendicular direction. In this case we generate a **2-D image using single sensor**, where there has a relative displacements in both the **x- and y directions** between the **sensor and the area to be imaged**.

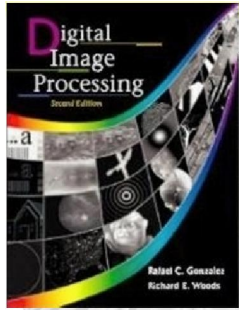
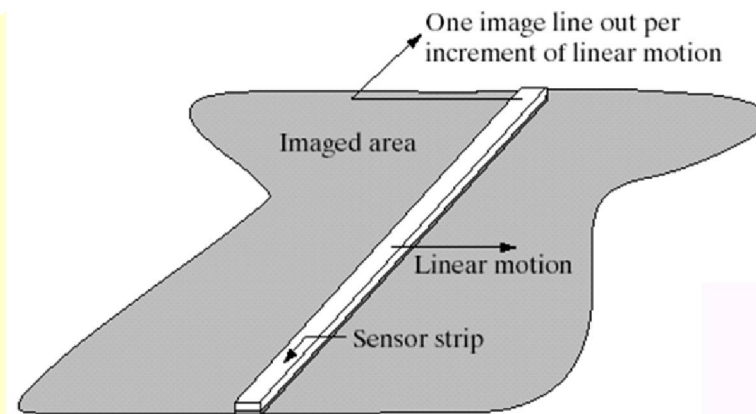


Image acquisition using line sensor or sensor strips

Medical and Industrial Computerized Axial Tomography (CAT)
Magnetic resonance images (MRI)
Positron Emission Tomography (PET)



In-line arrangement of sensors in the form of a **sensor strip**. The strip provides imaging elements in one direction. **Motion perpendicular to the strip provides** imaging in the other direction. This is the type of arrangement used in the most flat bed scanners. **4000 in-line sensor is possible**. The number of the sensor in the strip establishes the **sampling limitations in one image direction**. But in single sensor, sampling is accomplished by selecting the number of individual mechanical increments.

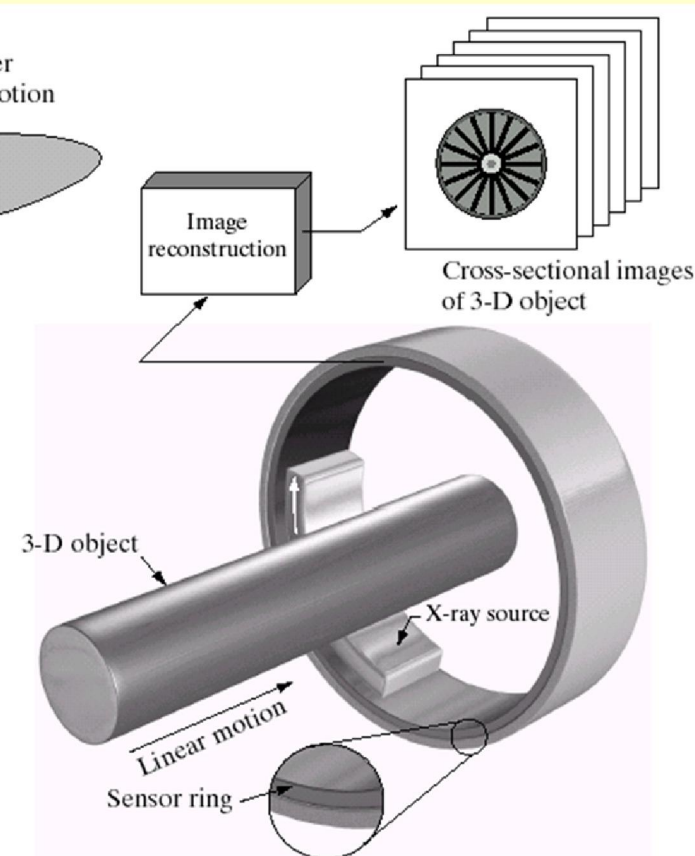


FIGURE 2.14 (a) Image acquisition using a linear sensor strip. (b) Image acquisition using a circular sensor strip.

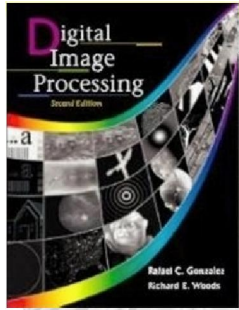


Image acquisition using sensor arrays

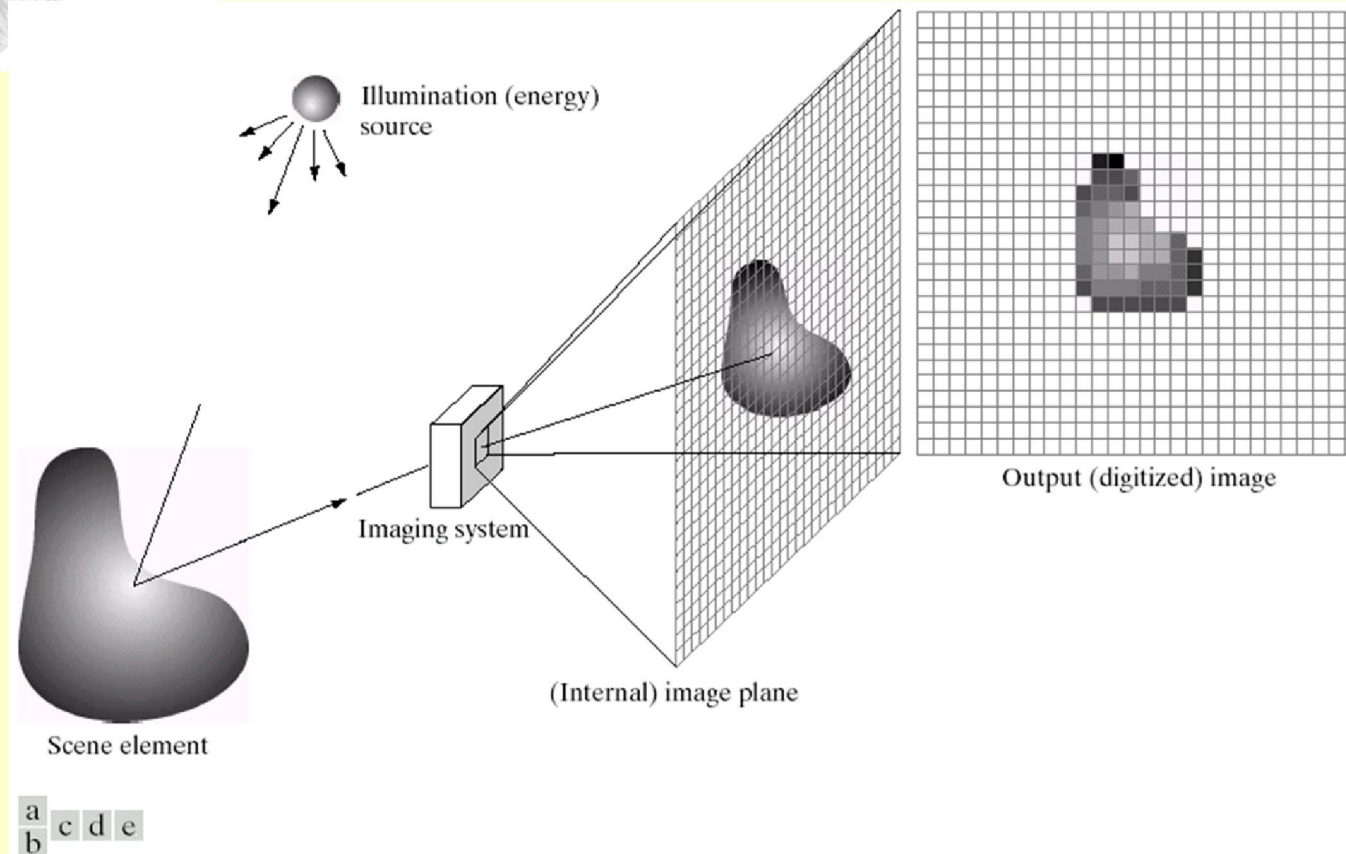
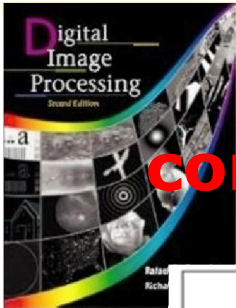
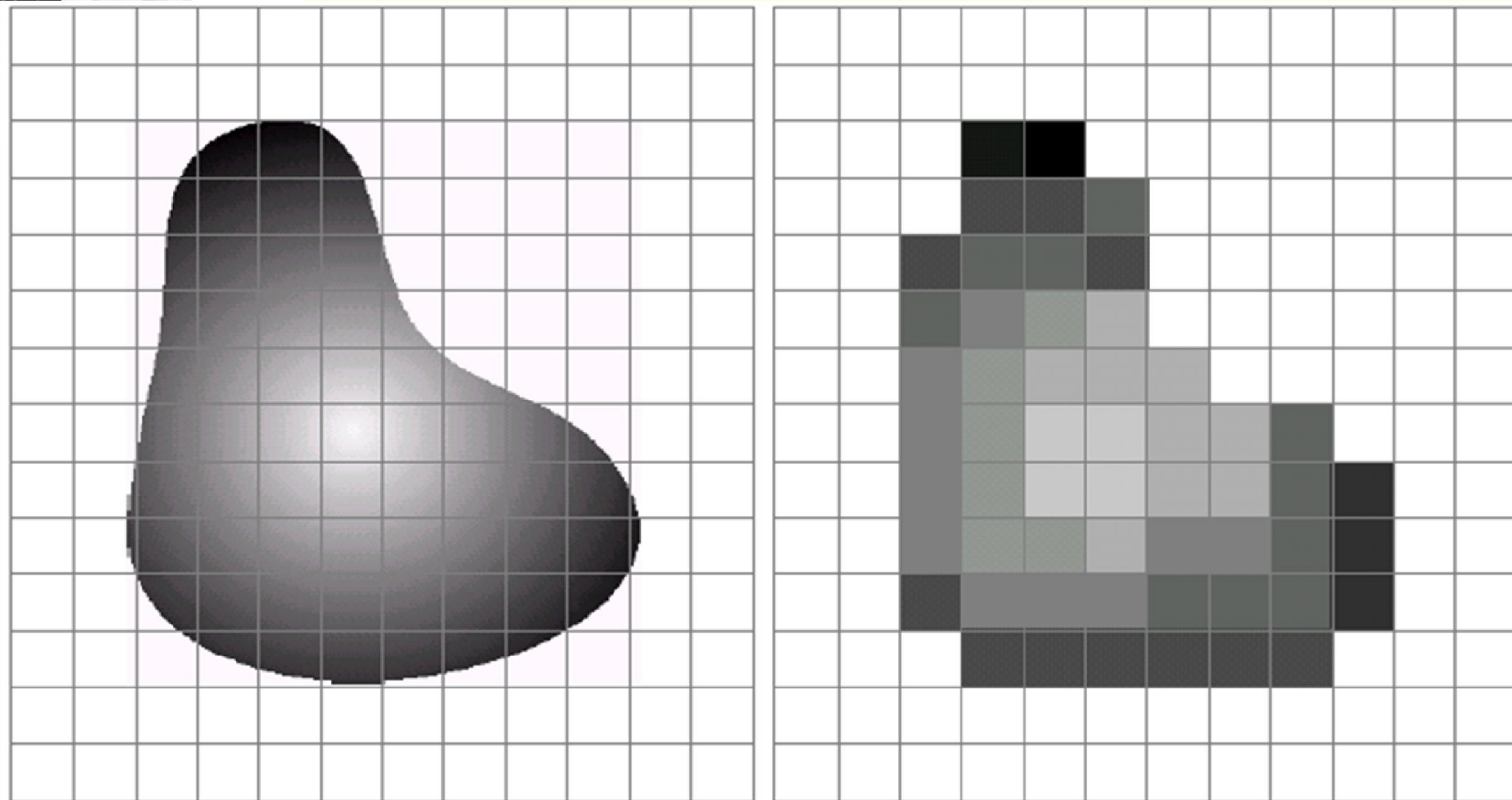


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.



continuous image projected onto a sensor array

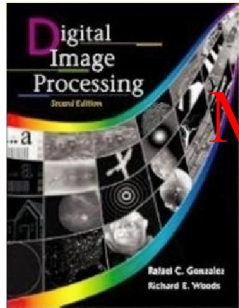


a b

FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

• Pixel Values

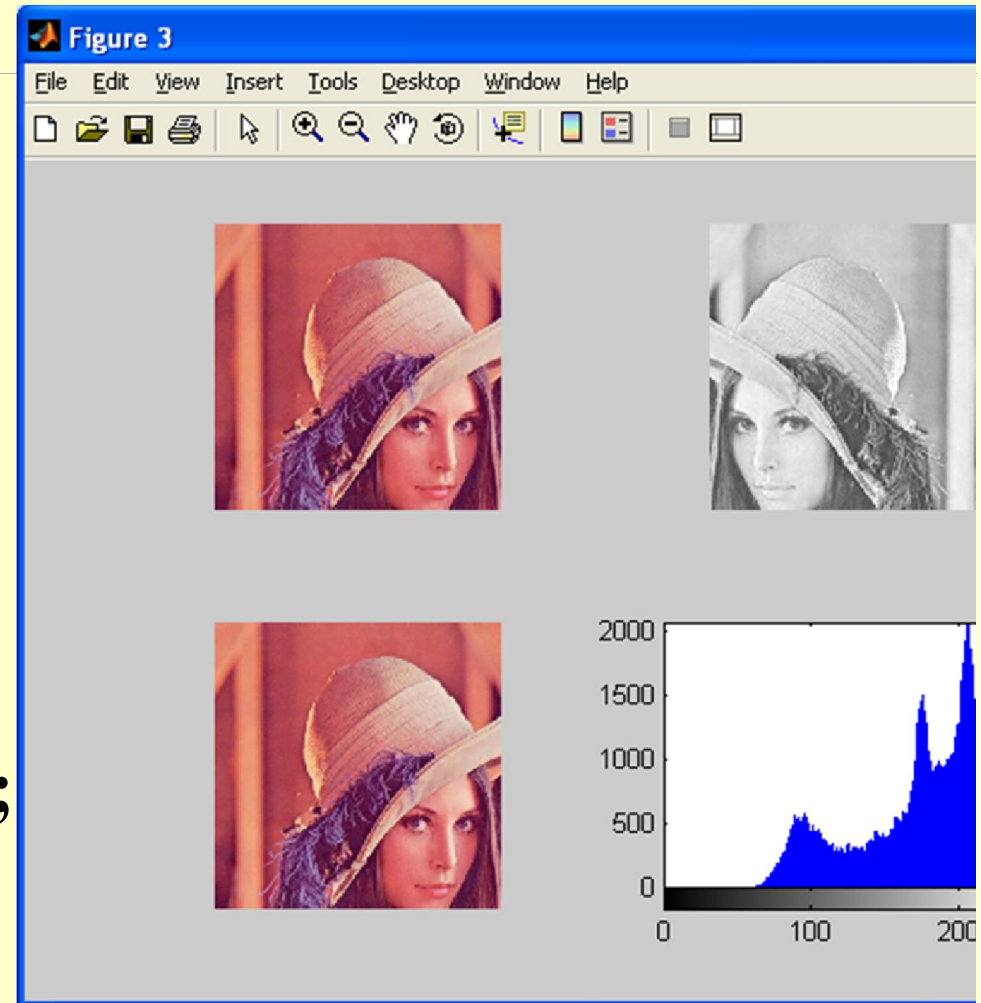
- Each of the pixels that represents an image stored inside a computer has a *pixel value* which describes how bright that pixel is, and/or what color it should be. In the simplest case of binary images, the pixel value is a 1-bit number indicating either foreground or background. For a grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common *pixel format* is the *byte image*, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of gray.
- To represent color images, separate red, green and blue components must be specified for each pixel (assuming an RGB colorspace), and so the pixel 'value' is actually a vector of three numbers. Often the three different components are stored as three separate 'grayscale' images known as *color planes* (one for each of red, green and blue), which have to be recombined when displaying or processing.
- Multi-spectral images can contain even more than three components for each pixel, and by extension these are stored in the same kind of way, as a vector pixel value, or as separate color planes. See satellite images 7 band in TM
- The actual grayscale or color component intensities for each pixel may not actually be stored explicitly. Often, all that is stored for each pixel is an index into a colormap in which the actual intensity or colors can be looked up.
- Although simple 8-bit integers or vectors of 8-bit integers are the most common sorts of pixel values used, some image formats support different types of value, for instance 32-bit signed integers or floating point values. Such values are extremely useful in image processing as they allow processing to be carried out on the image where the resulting pixel values are not necessarily 8-bit integers. If this approach is used then it is usually necessary to set up a colormap which relates particular ranges of pixel values to particular displayed colors.
-

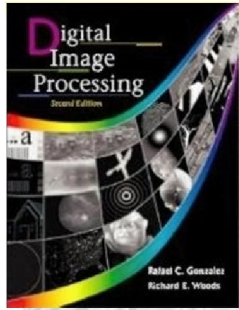


MATLAB CODE TO READ DISPLAY IMAGES

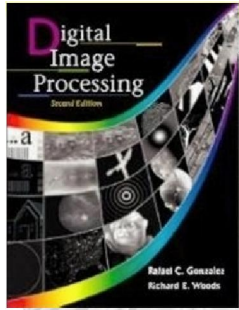
examples

- **`A=imread('c:\lena.jpg')`**
- **`figure`**
- **`imshow(A)`**
- **`iminfo('C:\lena.jpg')`**
- **`for i=1:380`**
- **`for j=1:380`**
- **`B(i,380+1-j)=A(i,j);`**
- **`end`**
- **`end`**





- **figure**
- **subplot(2,2,1)**
- **imshow(A)**
- **%figure**
- **subplot(2,2,2)**
- **imshow(B)**
- **subplot(2,2,3)**
- **%figure**
- **imshow(A)**
- **subplot(2,2,4)**
- **imhist(B)**
- **size (B)**



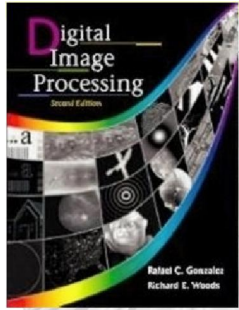
Representing Digital Images

TABLE 2.1

Number of storage bits for various values of N and k .

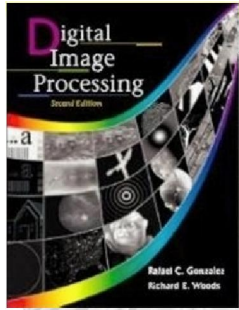
$$L = 2^k$$

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912



Spatial and Intensity Resolution

- Spatial resolution
 - A measure of the smallest discernible detail in an image
 - stated with *line pairs per unit distance*, *dots (pixels) per unit distance*, *dots per inch (dpi)*
- Intensity resolution
 - The smallest discernible change in intensity level
 - stated with *8 bits*, *12 bits*, *16 bits*, *etc.*



Spatial and Intensity Resolution

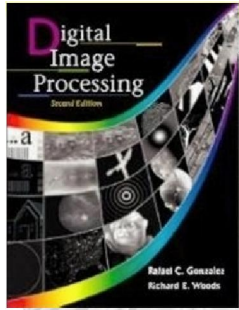
Digital Image Processing, 2nd ed.

www.imageprocessingbook.com



a b
c d

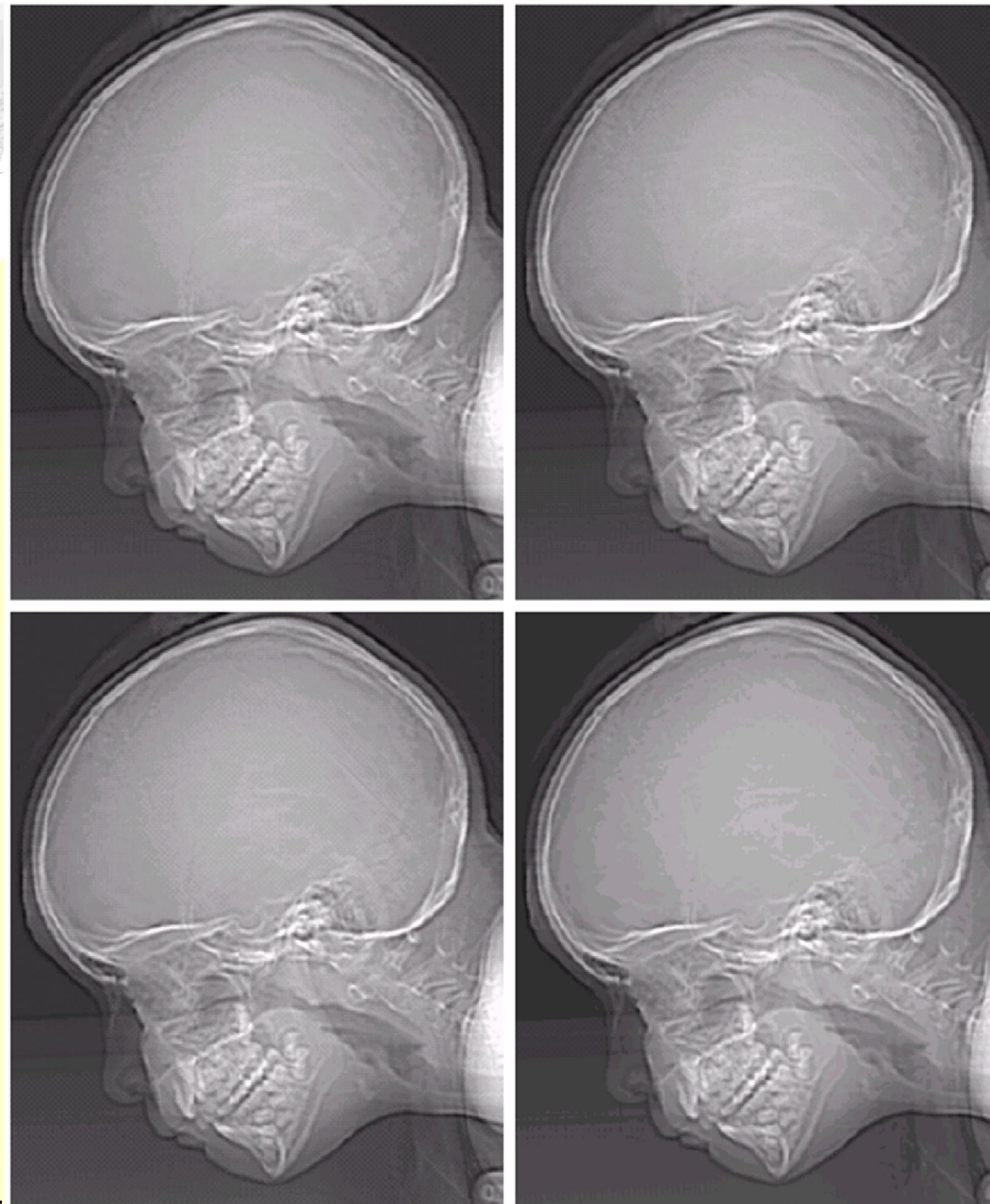
FIGURE 2.20 Typical effects of reducing spatial resolution. Images shown at: (a) 1250 dpi, (b) 300 dpi, (c) 150 dpi, and (d) 72 dpi. The thin black borders were added for clarity. They are not part of the data.



Spatial and Intensity Resolution

Digital Image Processing, 2nd ed.

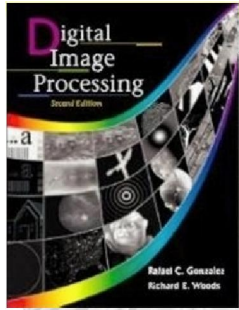
www.imageprocessingbook.com



a	b
c	d

FIGURE 2.21

(a) 452×374 , 256-level image. (b)–(d) Image displayed in 128, 64, and 32 gray levels, while keeping the spatial resolution constant.



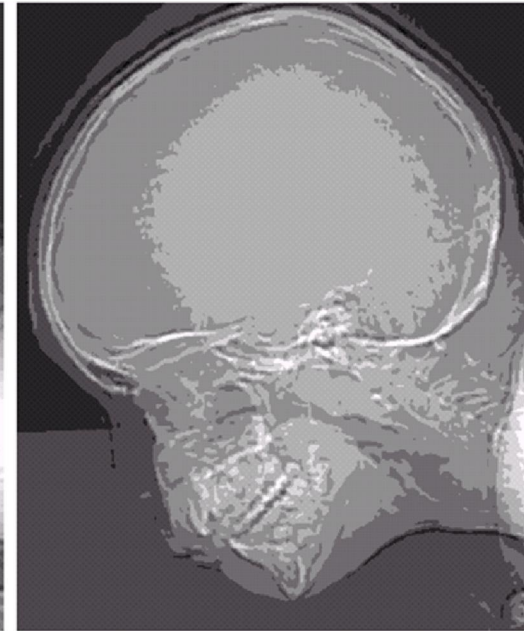
Spatial and Intensity Resolution

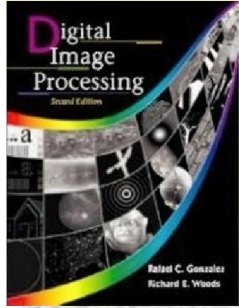
Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

e f
g h

FIGURE 2.21
(Continued)
(e)–(h) Image displayed in 16, 8, 4, and 2 gray levels. (Original courtesy of Dr. David R. Pickens, Department of Radiology & Radiological Sciences, Vanderbilt University Medical Center.)





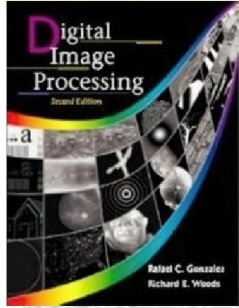
Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

Basic Relationships Between Pixels

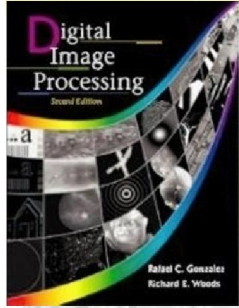
- Neighborhood
- Adjacency
- Connectivity
- Paths
- Regions and boundaries





Basic Relationships Between Pixels

- **Neighbors of a pixel p at coordinates (x,y)**
 - **4-neighbors of p** , denoted by $N_4(p)$:
 $(x-1, y)$, $(x+1, y)$, $(x, y-1)$, and $(x, y+1)$.
 - **4 diagonal neighbors of p** , denoted by $N_D(p)$:
 $(x-1, y-1)$, $(x+1, y+1)$, $(x+1, y-1)$, and $(x-1, y+1)$.
 - **8 neighbors of p** , denoted $N_8(p)$
 $N_8(p) = N_4(p) \cup N_D(p)$



Basic Relationships Between Pixels

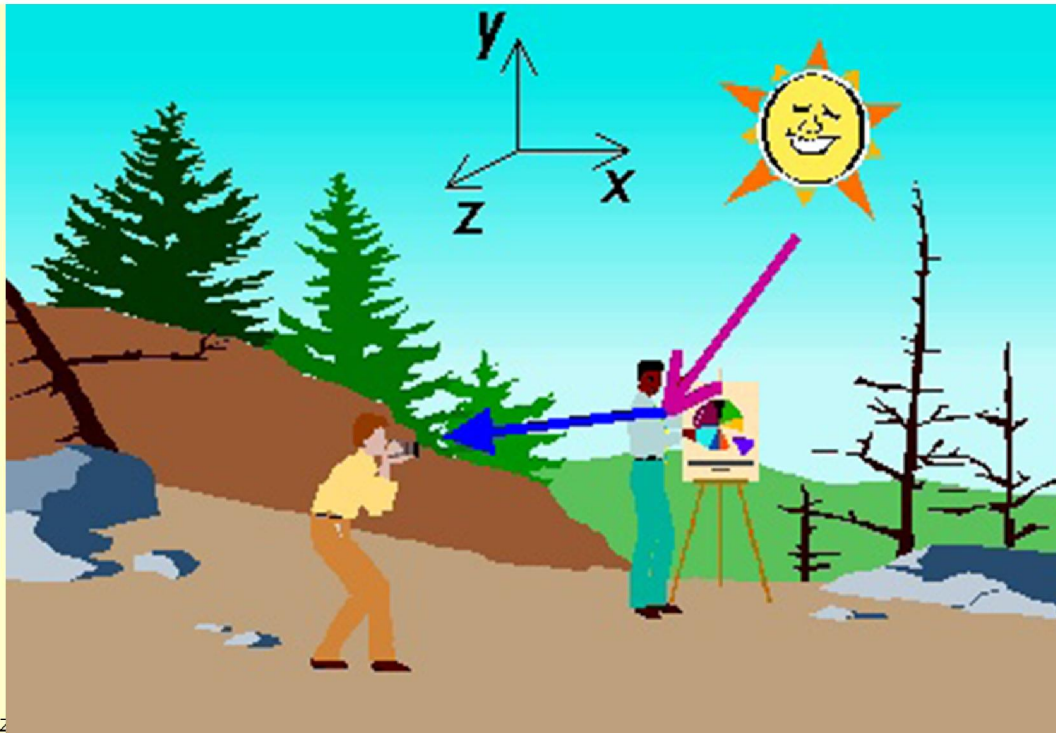
- **Adjacency**

Let V be the set of intensity values

- **4-adjacency:** Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.
- **8-adjacency:** Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.



- For natural images we need a light source (λ : wavelength of the source) ^(?).
 - $E(x, y, z, \lambda)$: incident light on a point (x, y, z world coordinates of the point)
- Each point in the scene has a reflectivity function.
 - $r(x, y, z, \lambda)$: reflectivity function
- Light reflects from a point and the reflected light is captured by an imaging device.
 - $c(x, y, z, \lambda) = E(x, y, z, \lambda) \times r(x, y, z, \lambda)$: reflected light.



- In lecture one

- Image Formation
- Sensors types
- Image types
- Image file size

The Electromagnetic Spectrum

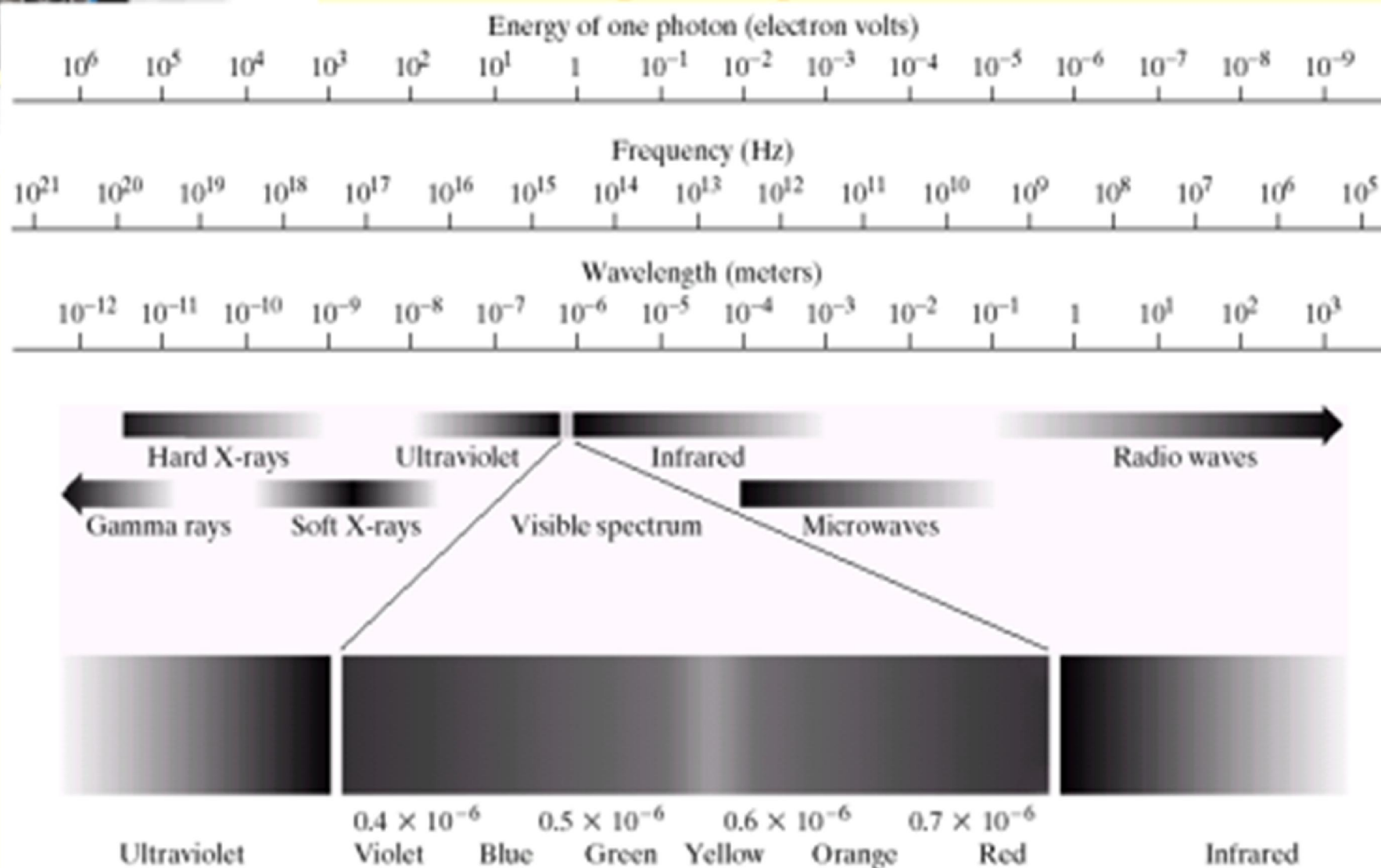


FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum.

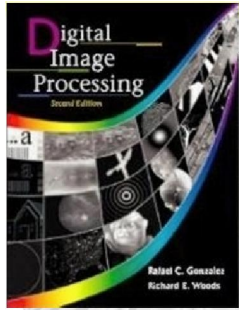
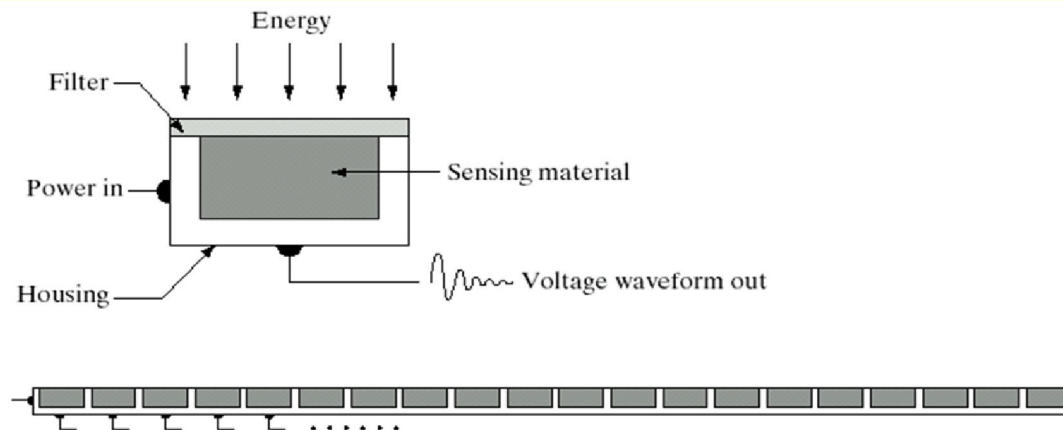


Image Sensing and Acquisition

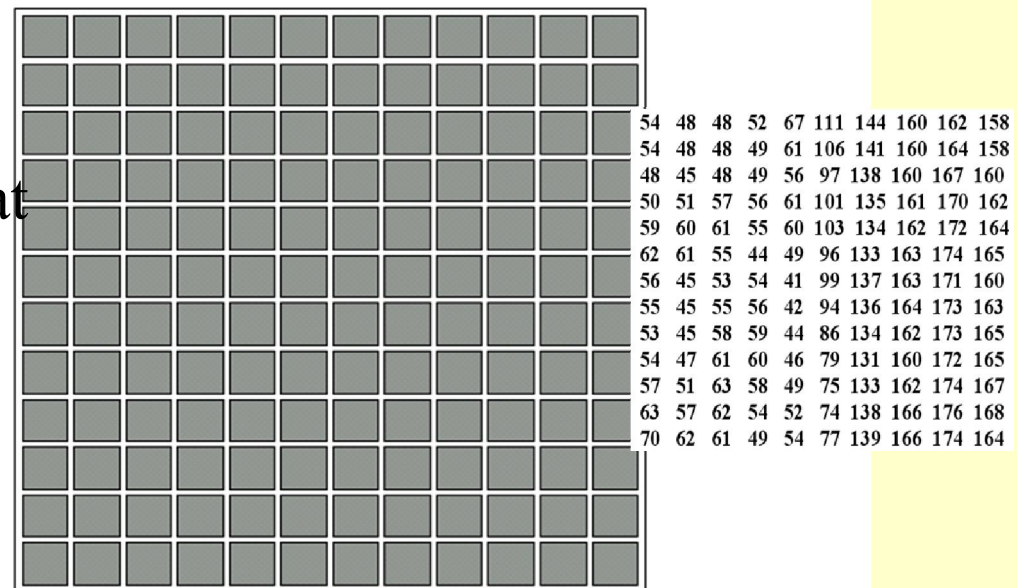
a
b
c

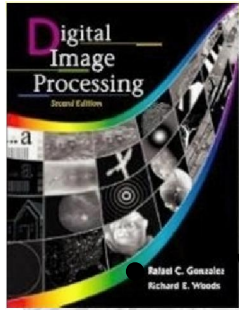
FIGURE 2.12

(a) Single imaging sensor.
(b) Line sensor.
(c) Array sensor.



A digital image is nothing more than data—numbers indicating variations of red, green, and blue at a particular location on a grid of pixels.





Introduction

What is Digital Image Processing?

Digital Image

— A two-dimensional function $f(x, y)$ x and y are spatial coordinates

The amplitude of f is called **intensity** or **gray level** at the point (x, y)

Digital Image Processing

— Process digital images by means of computer, it covers low-, mid-, and high-level processes

low-level: inputs and outputs are images

mid-level: outputs are attributes extracted from input images

high-level: an ensemble of recognition of individual objects

Pixel

— The elements of a digital image

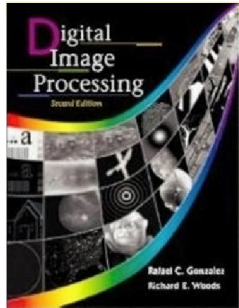
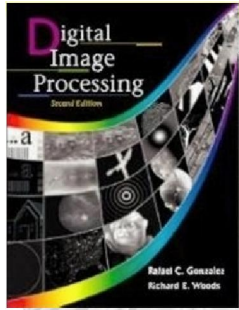


image definition

- An image may be defined as a two – dimensional function, $f(x, y)$, where x and y are spatial (plane) coordinates, and the amplitude of f at any pair coordinates (x, y) is called the intensity or gray level of the image at that point.
- When x , y , and the amplitude values of f are all finite, discrete quantities, we call the image a digital image.



Digital image representation

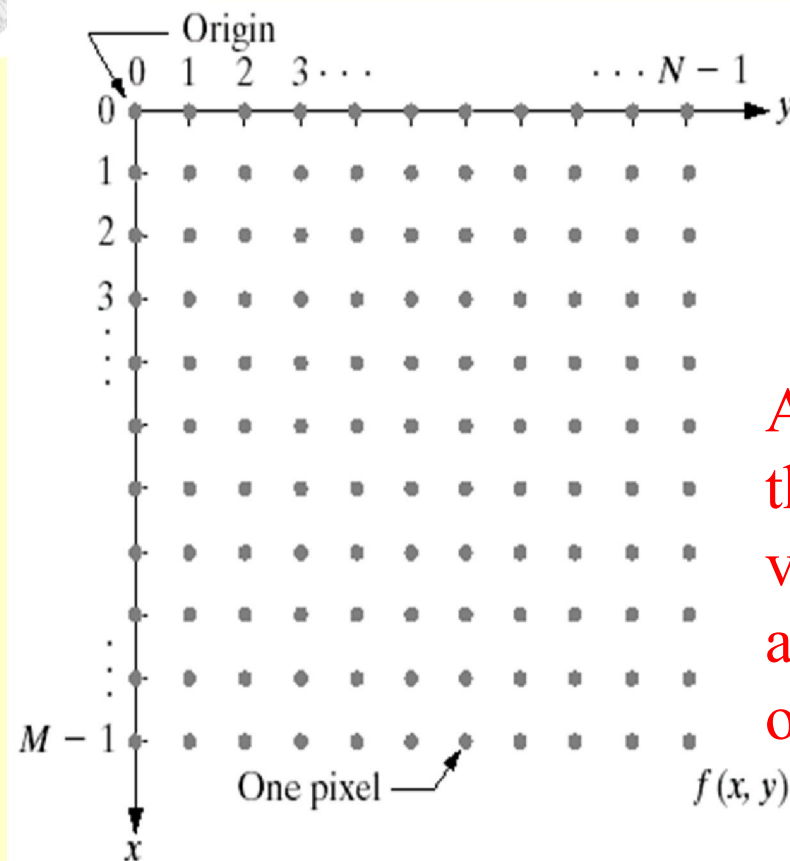
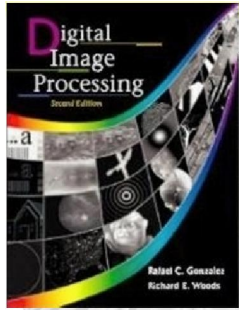


FIGURE 2.18

Coordinate convention used in this book to represent digital images.

A digital image is nothing more than data—numbers indicating variations of red, green, and blue at a particular location on a grid of pixels.



Digital image representation

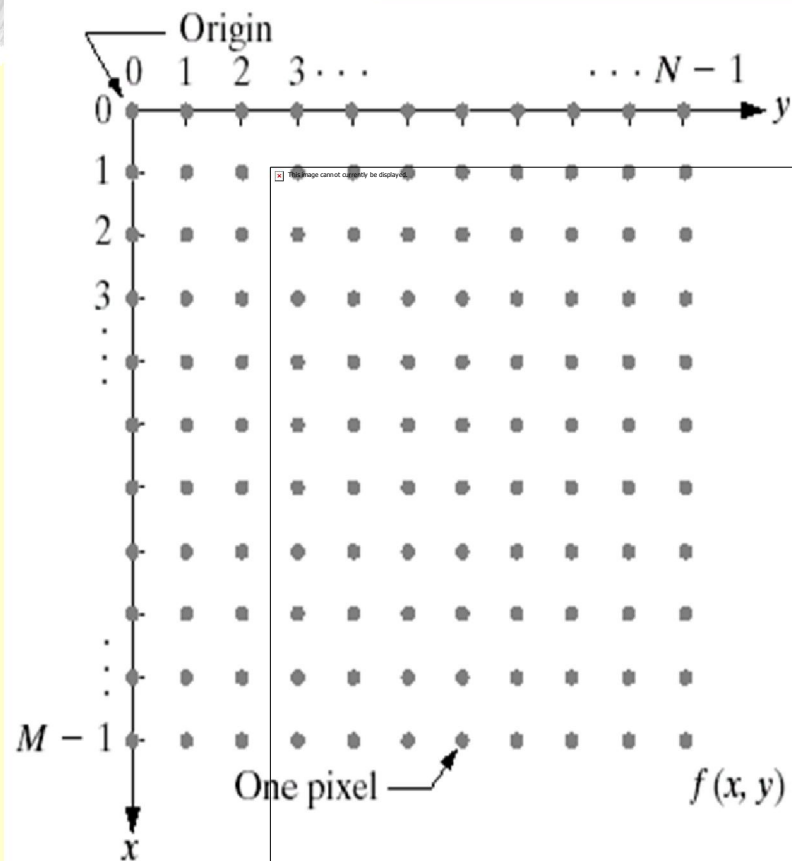
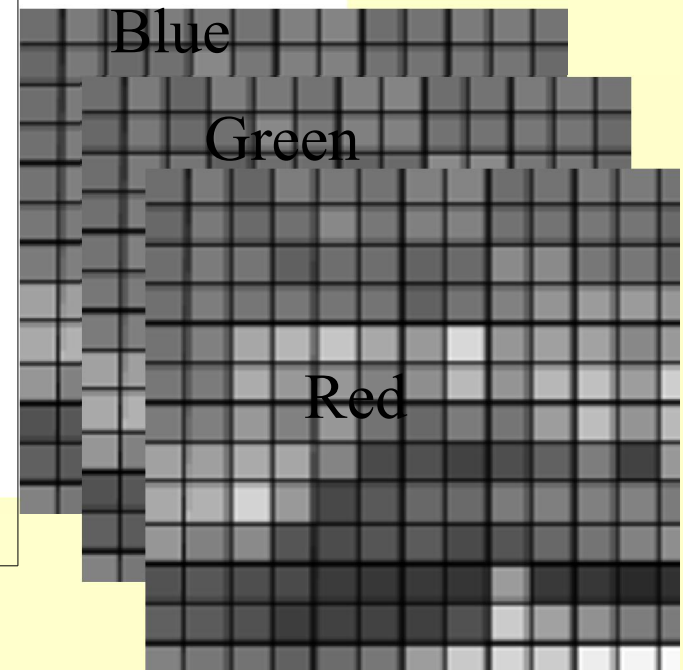
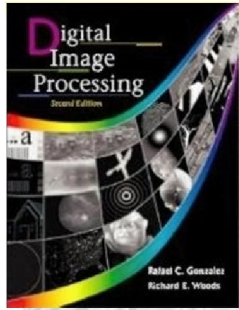


FIGURE 2.18
Coordinate convention used in this book to represent digital images.





Digital image Array representation

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots\dots\dots f(0, N-1) \\ f(1,0) & f(1,1) & \dots\dots\dots f(1, N-1) \\ \vdots & & \\ \vdots & & \\ f(M-1,0) & f(M-1,1) & \dots\dots f(M-1, N-1) \end{bmatrix}$$

$$f(x, y) = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots\dots\dots a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots\dots\dots a_{1,N-1} \\ \vdots & & \\ \vdots & & \\ a_{M-1,0} & a_{M-1,1} & \dots\dots\dots a_{M-1,N-1} \end{bmatrix}$$

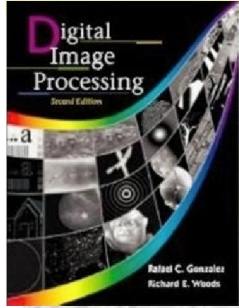


Images as Matrices

- An image matrix ($N \times M$):

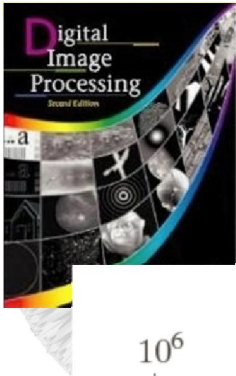
$$\mathbf{A} = \left[\begin{array}{cccc} A(0,0) & A(0,1) & A(0,2) & \dots A(0,M-1) \\ A(1,0) & A(1,1) & A(1,2) & \dots A(1,M-1) \\ \vdots & & & \\ A(N-1,0) & A(N-1,1) & A(N-1,2) & \dots A(N-1,M-1) \end{array} \right] \left. \vphantom{\begin{array}{c} A(0,0) \\ A(1,0) \\ \vdots \\ A(N-1,0) \end{array}} \right\} N \text{ rows } \textcircled{?}$$

- $A(i,j) \in \{0,1,\dots,255\}$.
- $A(i,j)$:
 - “Matrix case:” The matrix element (i,j) with value $A(i,j)$.
 - “Image case:” The pixel (i,j) with value $A(i,j)$.
 - Will use both terminologies.



Sources for Images

- Electromagnetic (EM) energy spectrum
 - Acoustic
 - Ultrasonic
 - Electronic
 - Synthetic images produced by computer



Electromagnetic (EM) energy spectrum

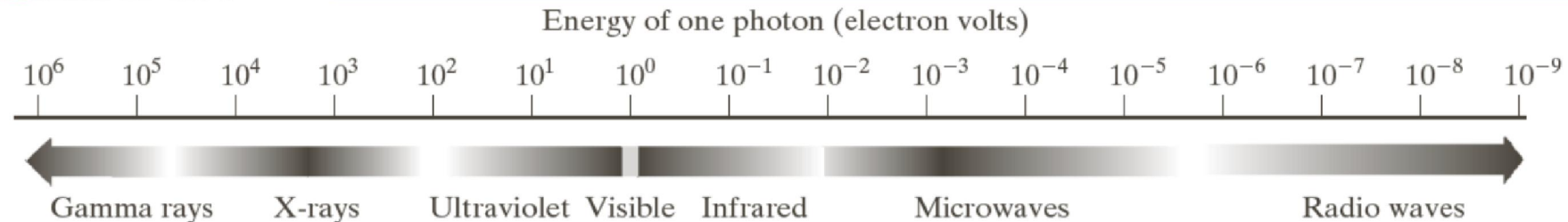


FIGURE 1.5 The electromagnetic spectrum arranged according to energy per photon.

Major fields in which digital image processing is widely used

Gamma-ray imaging: nuclear medicine and astronomical observations

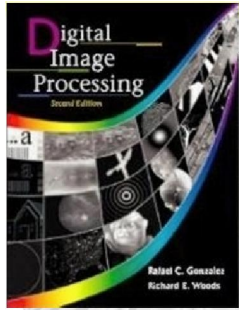
X-rays: medical diagnostics (**X-rays of body**), industry, and astronomy, etc.

Ultraviolet: lithography, industrial inspection, microscopy, lasers, biological imaging, and astronomical observations

Visible and infrared bands: light microscopy, astronomy, remote sensing, industry, and law enforcement

Microwave band: Radar imaging

Radio band: medicine (such as **MRI**) and astronomy



Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

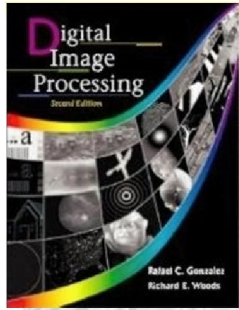
Pixels in image

How the pixels look:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

How the pixels are stored:

0	1	2	3	4	5	6	7	8	9	.	.	.		
---	---	---	---	---	---	---	---	---	---	---	---	---	--	--



Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

1. Assume a window or image with a given **WIDTH** and **HEIGHT**.
2. We then know the pixel array has a total number of elements equaling **WIDTH * HEIGHT**.
3. For any given X, Y point in the window, the location in our **1 dimensional** pixel array is: **LOCATION = X + Y*WIDTH**

x →

	0	1	2	3	4
y ↓	0	1	2	3	4
	5	6	7	8	9
	10	11	12	13	14
	15	16	17	18	19
	20	21	22	23	24

← width = 5 →

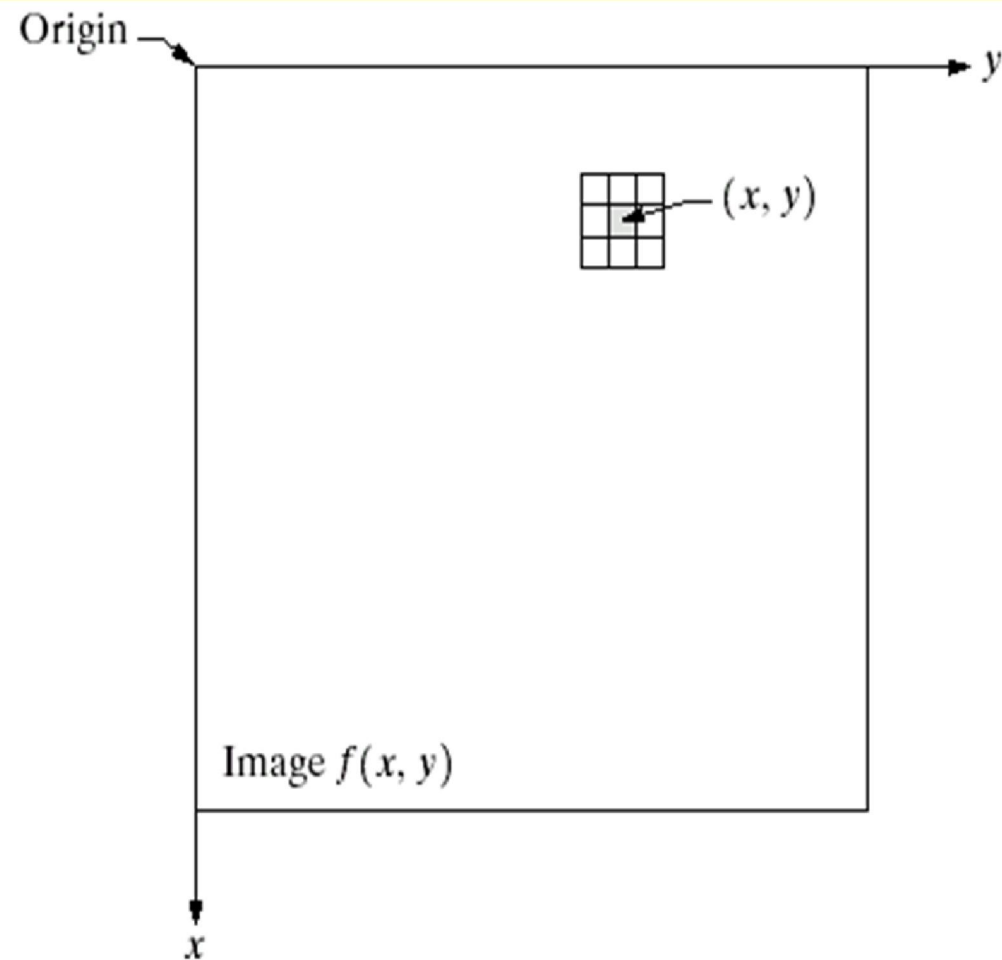
Pixel 13 has an x value of 3 and y value of 2.

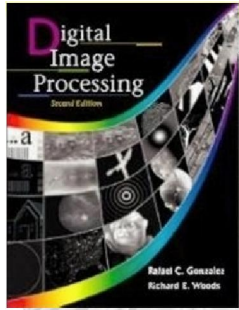
$$\begin{aligned}
 & x + (y * \text{width}) \\
 &= 3 + (2 * 5) \\
 &= 3 + 10 \\
 &= 13
 \end{aligned}$$

How the pixels are stored:

0	1	2	3	4	5	6	7	8	9	.	.	.		
---	---	---	---	---	---	---	---	---	---	---	---	---	--	--

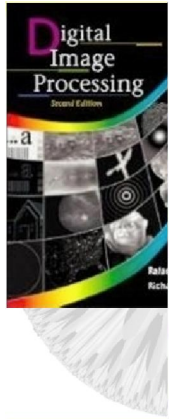
FIGURE 3.1 A 3×3 neighborhood about a point (x, y) in an image.



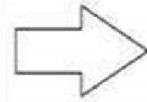


Digital image processing field

Digital image processing deals with manipulation of digital images through a digital computer. **It is a subfield of signals and systems but focus particularly on images.** DIP focuses on developing a computer system that is able to perform processing on an image. *The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output.* The most common example is **Adobe Photoshop**. It is one of the widely used application for processing digital images.



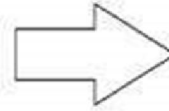
3d world around us



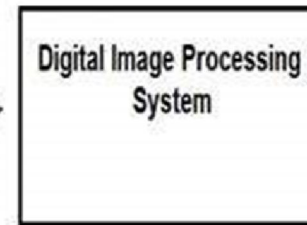
captured
by



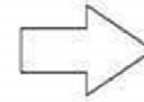
a camera



and sent to



a particular system to
focus on a water drop,



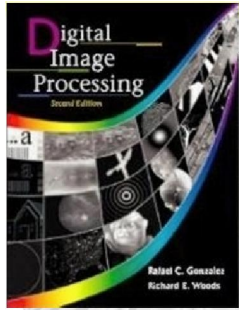
that's gives its
ouput as an



Processed image

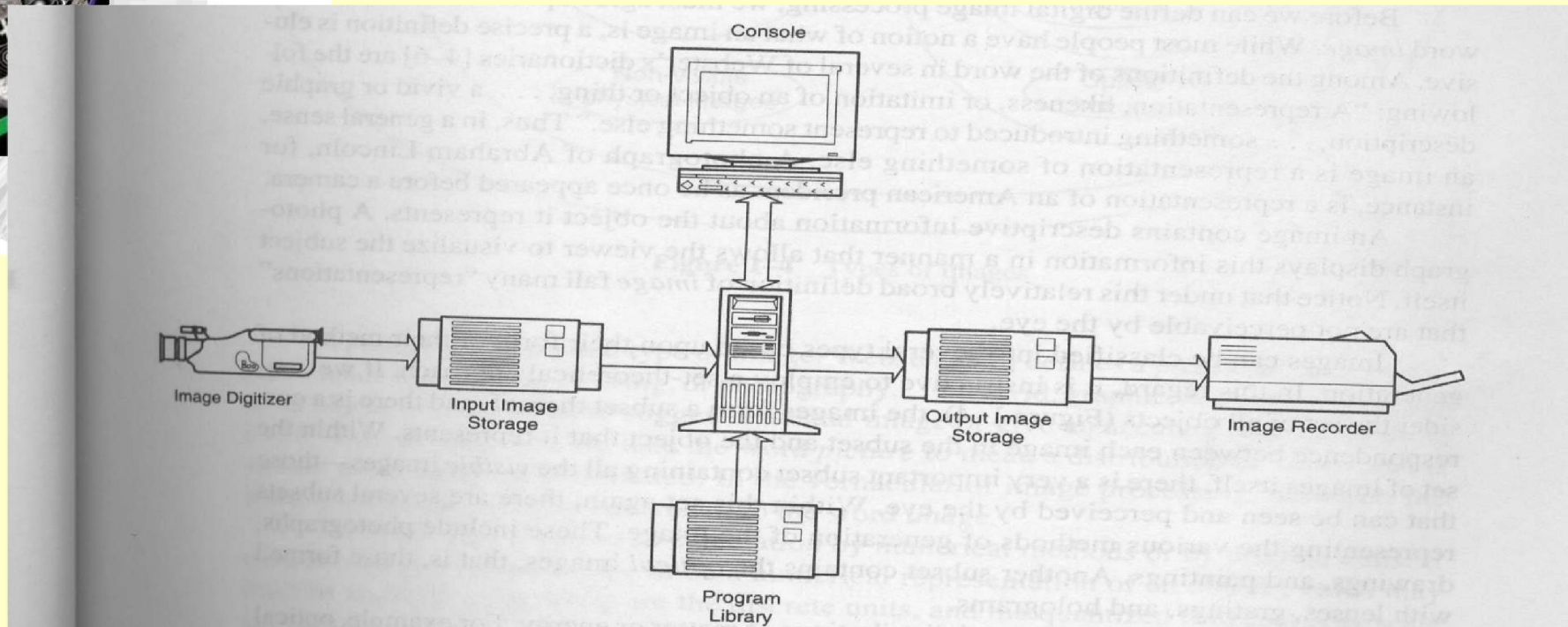
In the above figure, an image has been captured by a camera and has been sent to a digital system to remove all the other details, and just focus on the water drop by zooming it in such a way that the quality of the image remains the same.

<http://www.tutorialspoint.com/dip/>



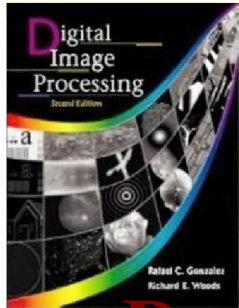
- Note that a **digital image is composed of a finite number of elements**, each of which has a particular **location and value**. These elements are referred to as *picture elements*, *image elements*, *peks*, and *pixels*. [1]
- ❖ Interest in digital image processing methods stems from **two principal application areas**:
 - 1- improvement of pictorial information for human interpretation; and
 - 2- processing of image data for storage, transmission, and representation for autonomous machine perception.

Complete system for Image processing



The digital image produced by the digitizer goes into temporary storage on a suitable device. In response to instructions from the operator, the computer calls up and executes image processing programs from library.

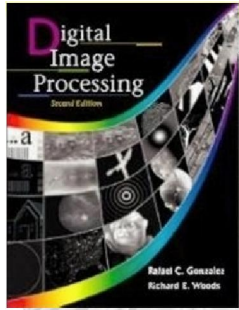
During the execution, The input image is read into the computer line by line .Operating upon one or several lines, the computer generates the output image, pixel by pixel, and store it on the output data storage device, line by line.



During the processing, the pixels may be modified. After processing, the final product is displayed by a process is the reversed of digitization:

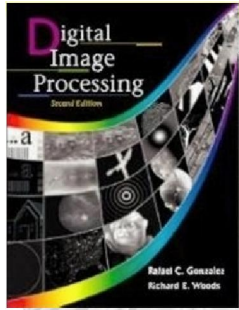
The gray level of each pixel is used to determine the brightness of the corresponding point on a display screen. The processed image is thereby made visible, and once again amenable to human interpretation.

the brightness of each pixel is represented by **a numeric value. Gray-scale images** typically contain values in the range from 0 to 255, with **0** representing **black**, **255** representing **white** and values in between representing shades of gray



Types of Computerized Processes

- 1-Low Level Process
- 2-Mid Level Process
- 3-High Level Process
- =====
- 1-low Level Process involves primitive operations, such as image processing to reduce noise, contrast, enhancement and image sharpening. In this level, both its input and output are digital images



Digital Image Processing, 2nd ed.

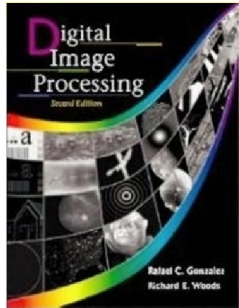
www.imageprocessingbook.com

Original



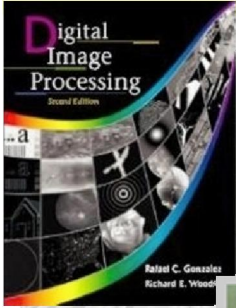
Imadjust





2-Mid Level Process

- Involves tasks such as ,**segmentation** (partitioning an image into regions or objects), description these objects to reduce them to a form suitable to computer, and **classification (recognition) of individual objects**. The inputs are digital images and the outputs are attributes extracted from those images (i.e, edges, contours, and the identity of individual objects).

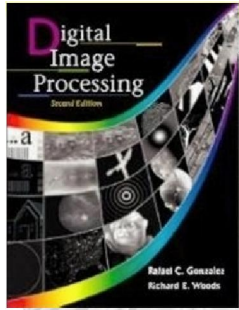


Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

PEARS COLOR IMAGE



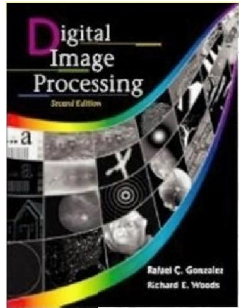


Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

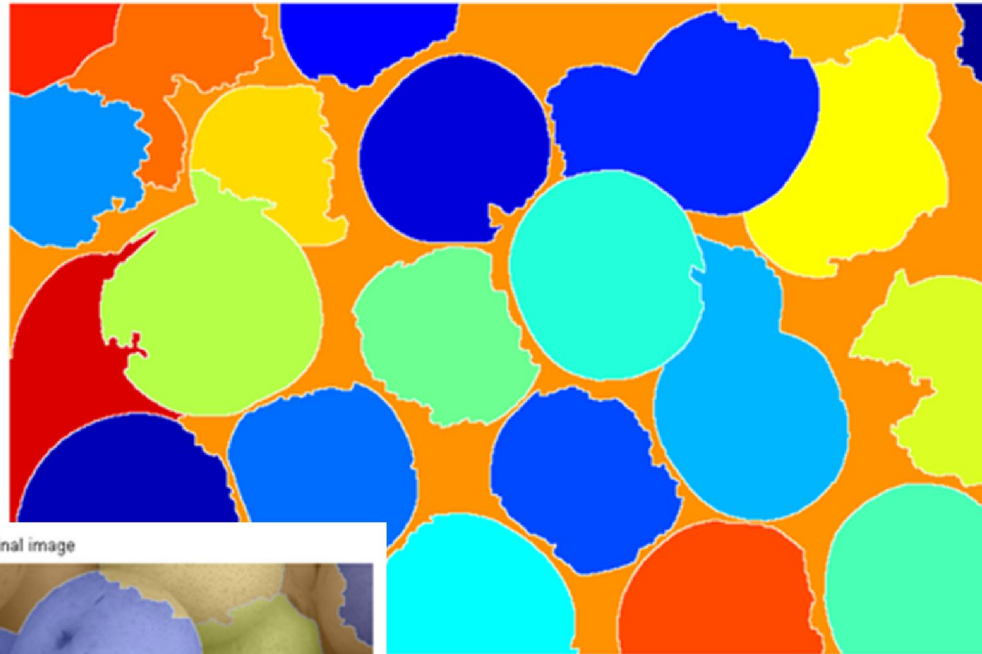
Pears GRAY IMAGE





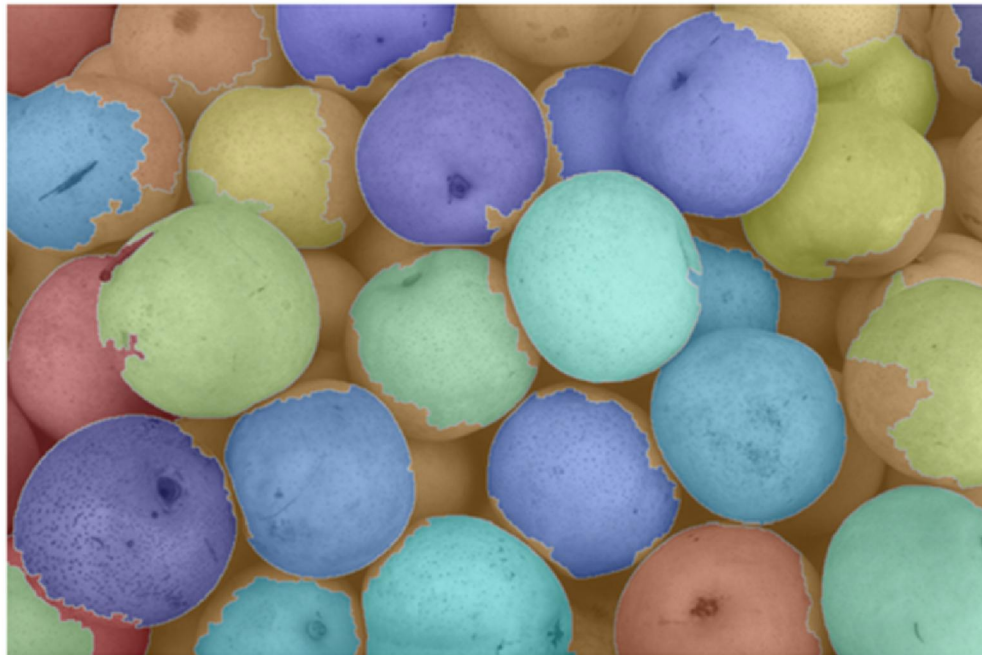
Digital Image Processing, Second Edition

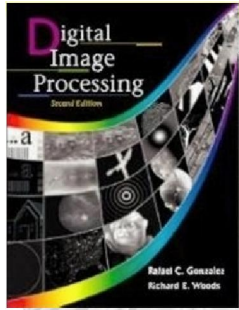
Colored watershed label matrix (Lrgb)



ingbook.com

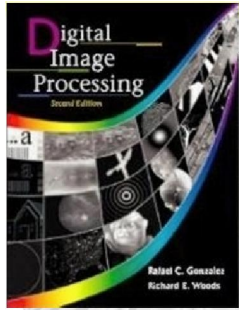
Lrgb superimposed transparently on original image





3-High Level Process

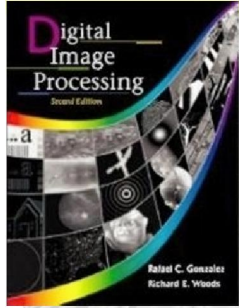
- Involves making sense of a recognized objects as in image analysis , for example : if a digital image contains a number of objects , a program may analyzed the image and extract the objects.
- So the digital image process encompasses processes whose inputs and outputs are images and in addition, encompasses processes that extract attributes from images up to and including recognition of individual objects.



Example

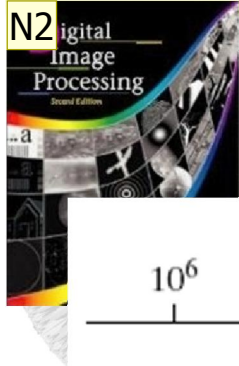
Consider the area of automated analysis of text:

- The processes of acquiring an image of the area containing the text.
- Preprocessing the image.
- Extracting (**segmentation**) the individual characters.
- Describing the characters in away **suitable for computer**.
- Recognizing these characters.



Where use images

- Medicine(diagnostics, brain images x-rays)
- Web sites
- Books and magazine
- TV, movies, graphics,
- Digital camera
- Official application forms
- ID
- Licence
- Satellite



The Electromagnetic Spectrum

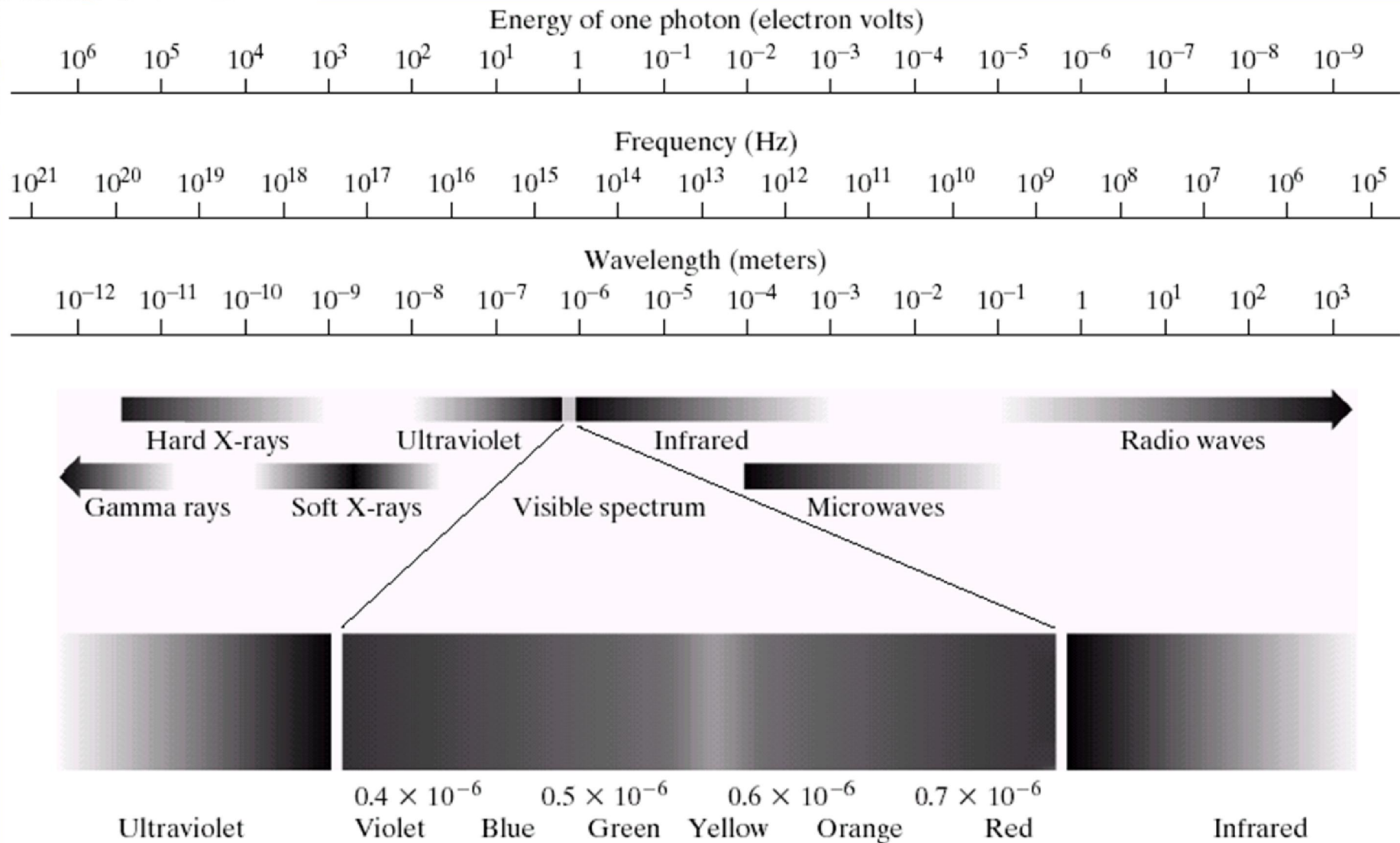
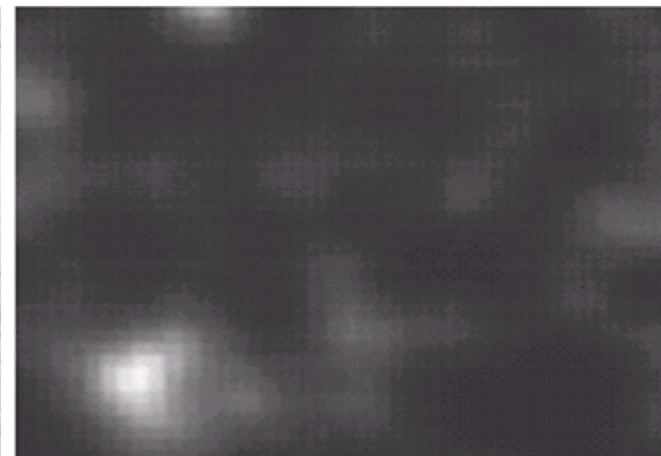
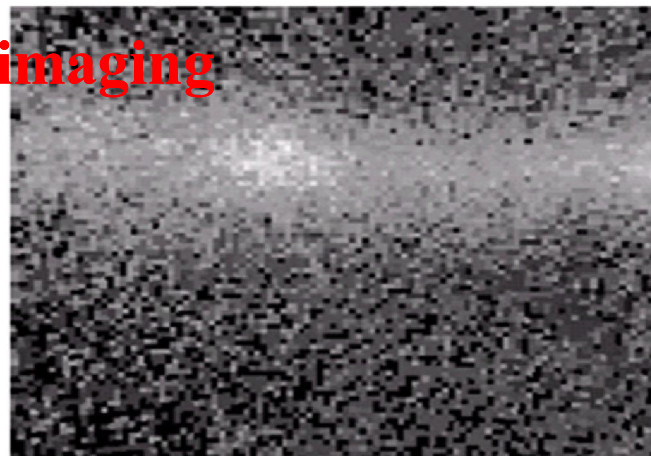
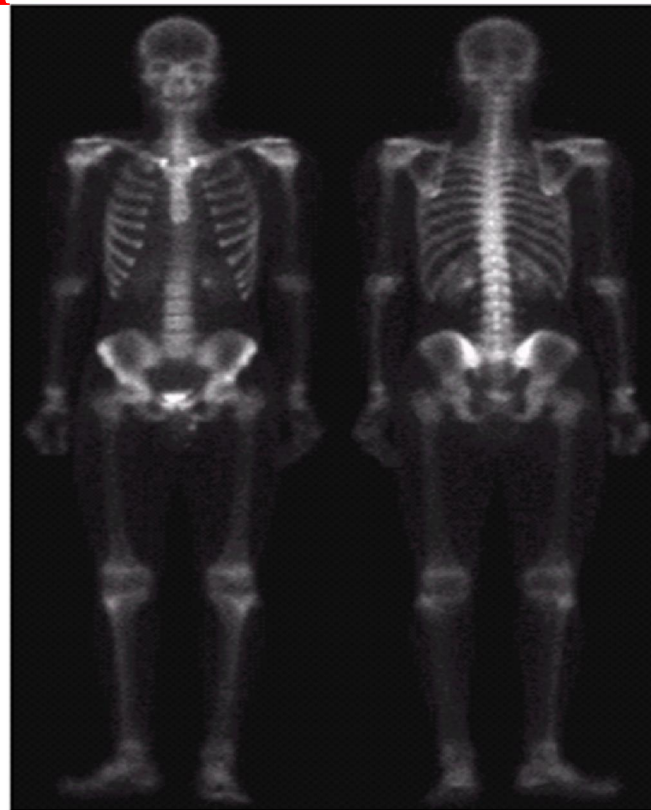


FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum.

Examples Of Fields That Use Digital Image Processing

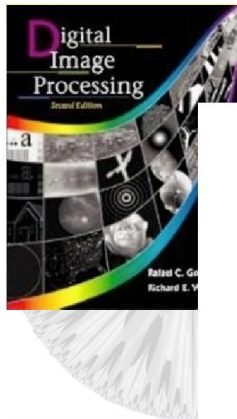
FIGURE 1.6

Examples of gamma-ray imaging. (a) Bone scan. (b) PET image. (c) Cygnus Loop. (d) Gamma radiation (bright spot) from a reactor valve. (Images courtesy of (a) G.E. Medical Systems, (b) Dr. Michael E. Casey, CTI PET Systems, (c) NASA, (d) Professors Zhong He and David K. Wehe, University of Michigan.)

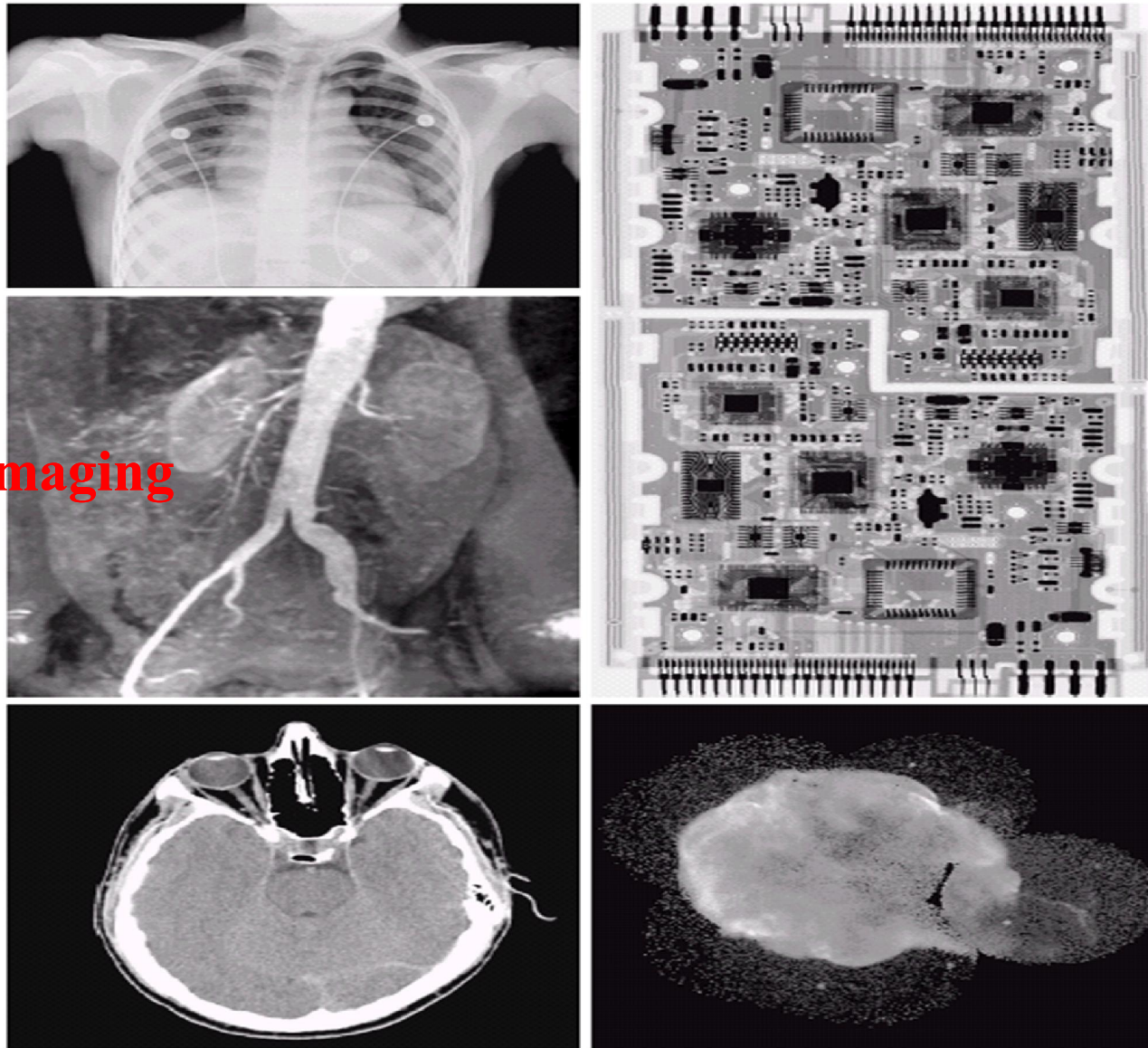


1-Gamma rays imaging



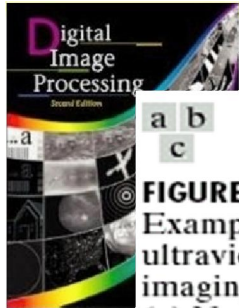


2-X-Rays imaging



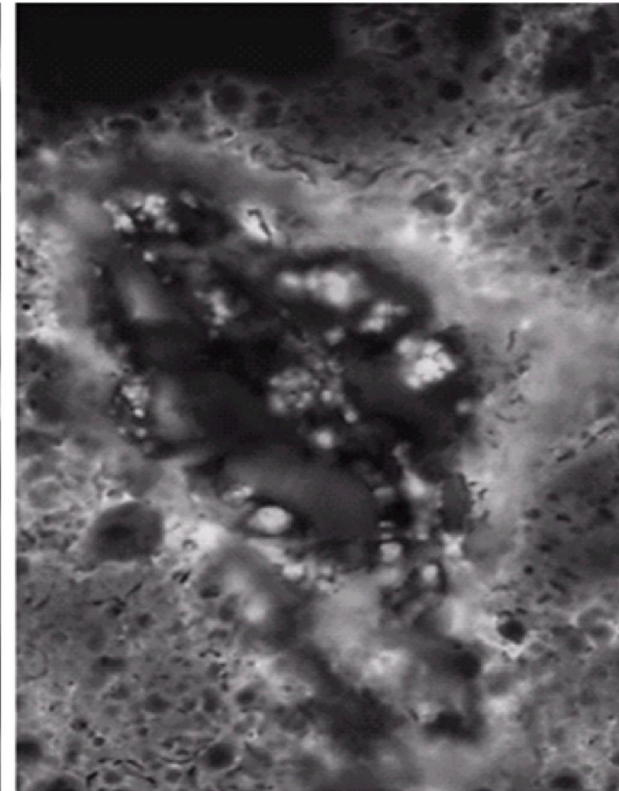
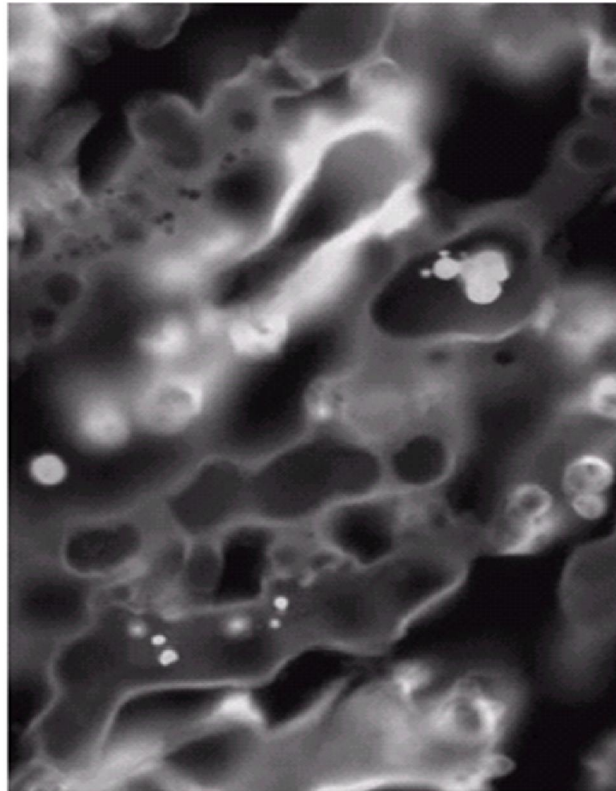
a	d
b	
c	e

FIGURE 1.7 Examples of X-ray imaging. (a) Chest X-ray. (b) Aortic angiogram. (c) Head CT. (d) Circuit boards. (e) Cygnus Loop. (Images courtesy of (a) and (c) Dr. David R. Pickens, Dept. of Radiology & Radiological Sciences, Vanderbilt University Medical Center, (b) Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, (d) Mr. Joseph E. Pascente, Lixi, Inc., and (e) NASA.)

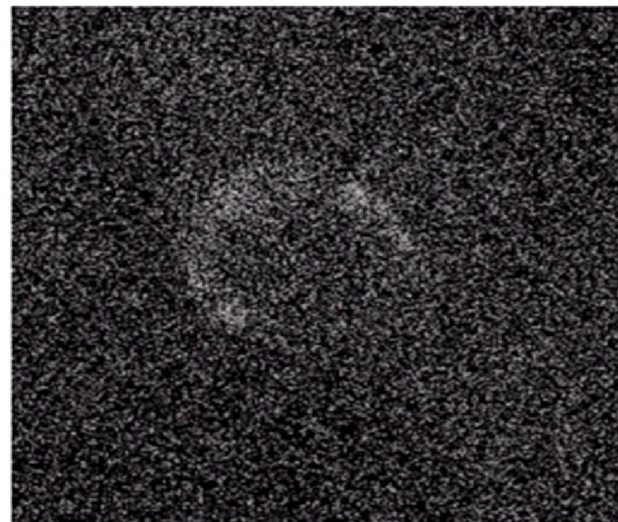


a b
c

FIGURE 1.8
Examples of
ultraviolet
imaging.
(a) Normal corn.
(b) Smut corn.
(c) Cygnus Loop.
(Images courtesy
of (a) and
(b) Dr. Michael
W. Davidson,
Florida State
University,
(c) NASA.)



3-Ultraviolet imaging



Satellite Imaging

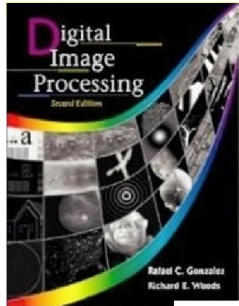
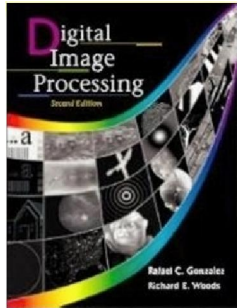


TABLE 1.1
Thematic bands
in NASA's
LANDSAT
satellite.

1-Spectral bands

Band No.	Name	Wavelength (μm)	Characteristics and Uses
1	Visible blue	0.45–0.52	Maximum water penetration
2	Visible green	0.52–0.60	Good for measuring plant vigor
3	Visible red	0.63–0.69	Vegetation discrimination
4	Near infrared	0.76–0.90	Biomass and shoreline mapping
5	Middle infrared	1.55–1.75	Moisture content of soil and vegetation
6	Thermal infrared	10.4–12.5	Soil moisture; thermal mapping
7	Middle infrared	2.08–2.35	Mineral mapping



TM Band Wavelength (μm)

6	10.4 - 12.5		Thermal Infrared
7	2.08 - 2.35		Shortwave Infrared
5	1.55 - 1.75		Shortwave Infrared
4	0.76 - 0.90		Near Infrared
3	0.63 - 0.69		Red
2	0.52 - 0.60		Green
1	0.45 - 0.52		Blue

Landsat satellite images in TM Bands

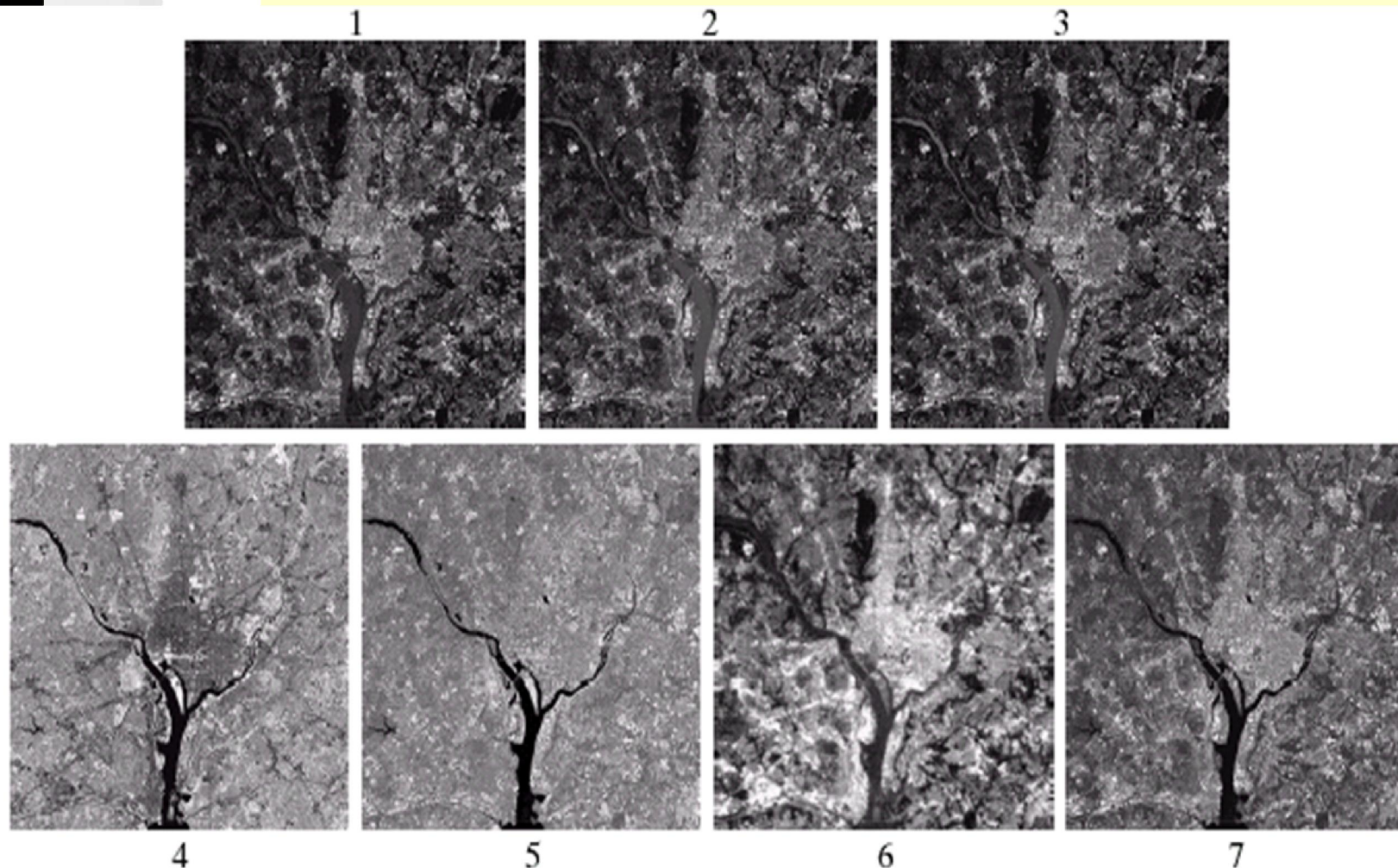


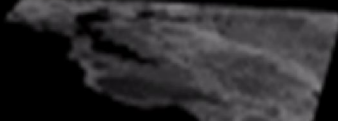




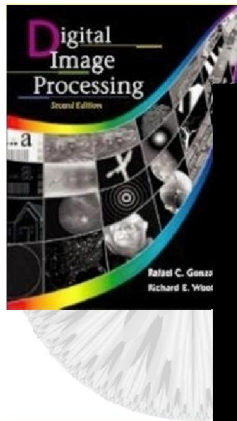



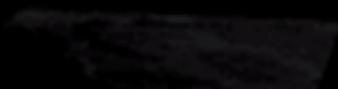

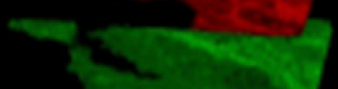



FIGURE 1.10 LANDSAT satellite images of the Washington, D.C. area. The numbers refer to the thematic bands in Table 1.1. (Images courtesy of NASA.)

TM Band Wavelength (μm)

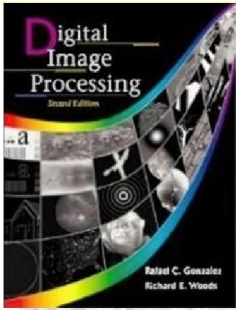
6	10.4 - 12.5		Thermal Infrared
7	2.08 - 2.35		Shortwave Infrared
5	1.55 - 1.75		Shortwave Infrared
4	0.76 - 0.90		Near Infrared
3	0.63 - 0.69		Red
2	0.52 - 0.60		Green
1	0.45 - 0.52		Blue



TM Band Wavelength (μm)

6	10.4 - 12.5		Thermal Infrared
7	2.08 - 2.35		Shortwave Infrared
5	1.55 - 1.75		Shortwave Infrared
4	0.76 - 0.90		Near Infrared
3	0.63 - 0.69		Red
2	0.52 - 0.60		Green
1	0.45 - 0.52		Blue

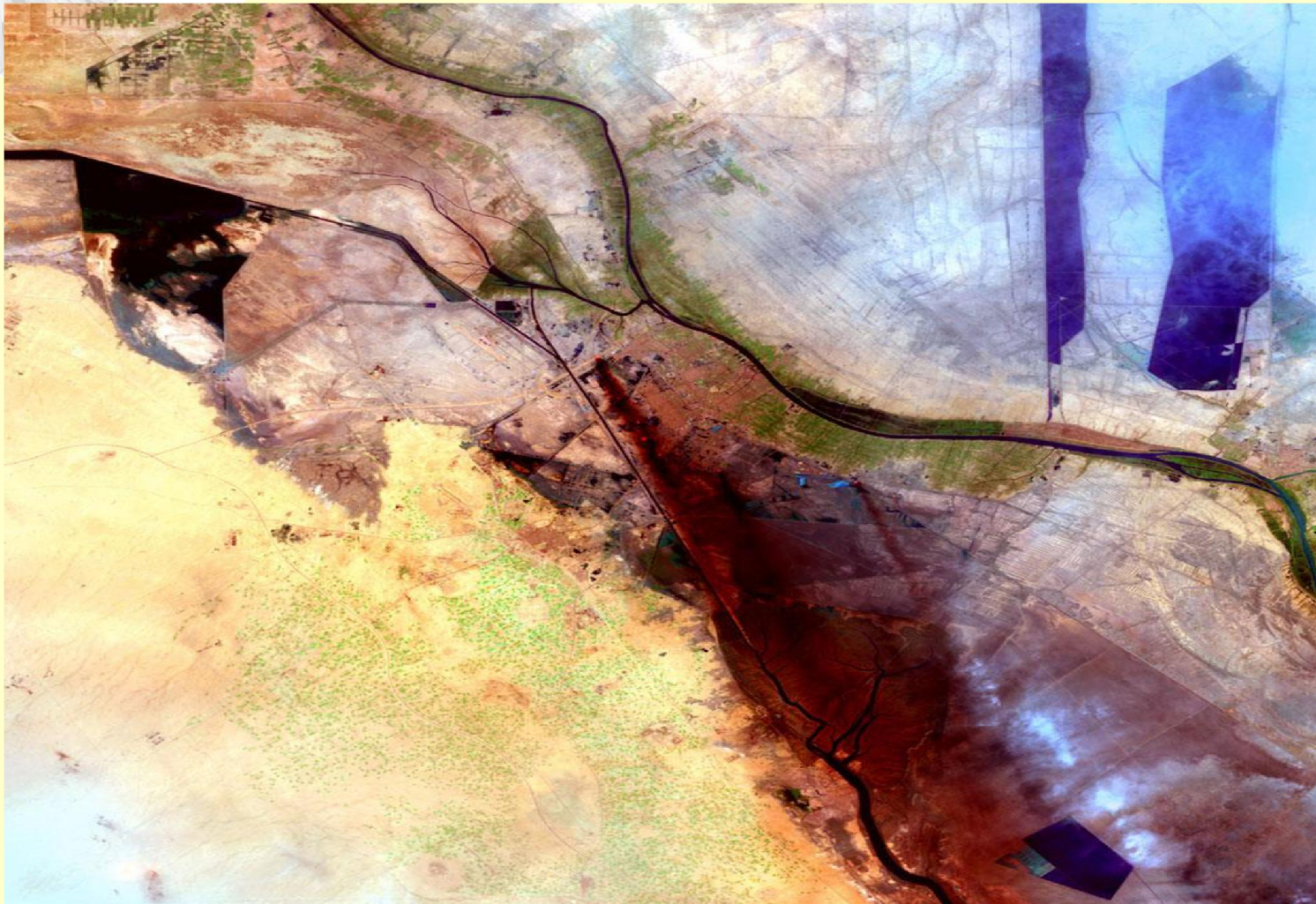
Combining TM bands 5, 4, & 2 to make an image

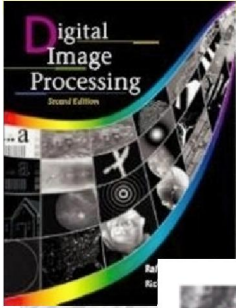


Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

Satellite Images Examples: Basra-IRAQ , April 4,2003 ,L7, 742





Images used In metrology

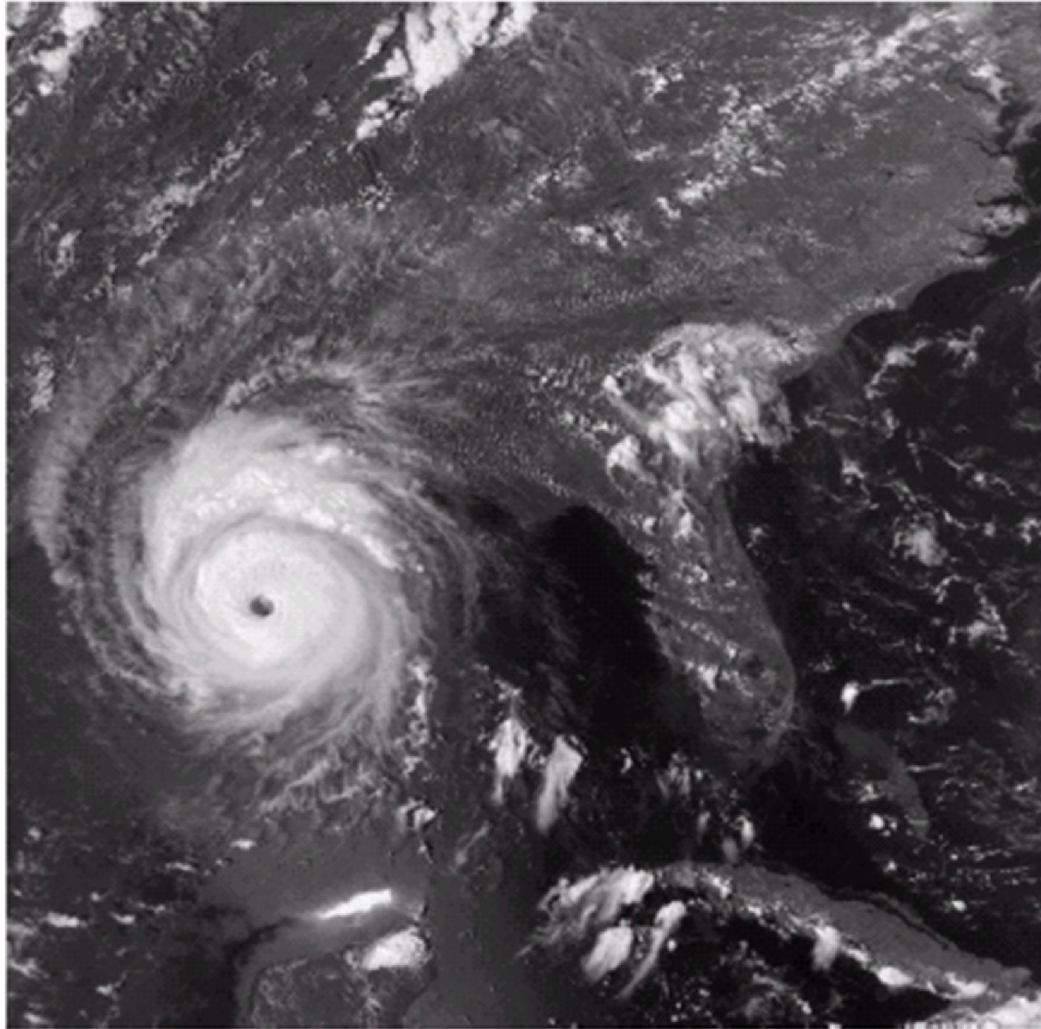
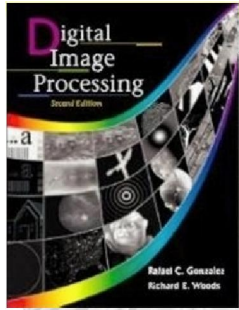
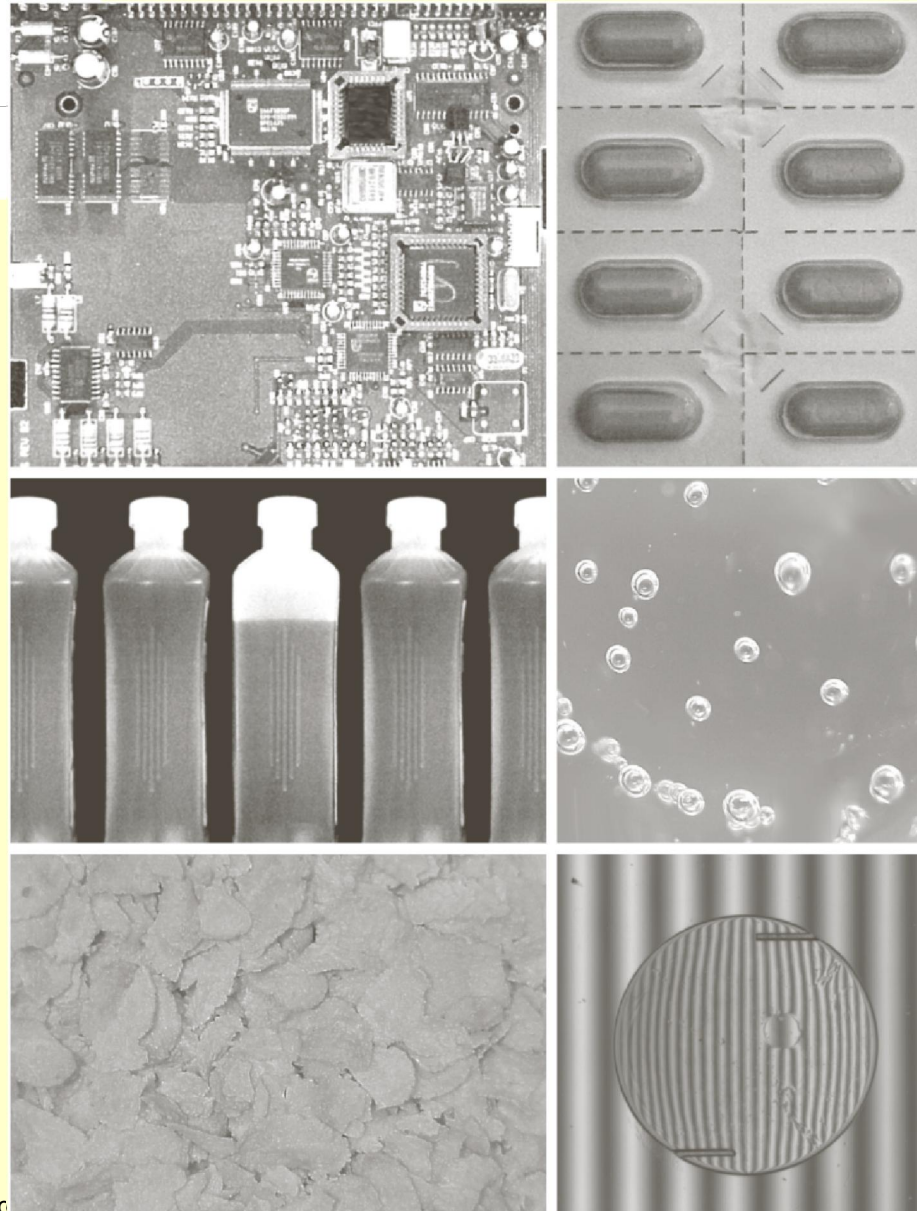


FIGURE 1.11
Multispectral
image of
Hurricane
Andrew taken by
NOAA GEOS
(Geostationary
Environmental
Operational
Satellite) sensors.
(Courtesy of
NOAA.)



Digital Image Processing, 2nd ed. **Examples: Automated Visual Inspection**

www.imageprocessingbook.com



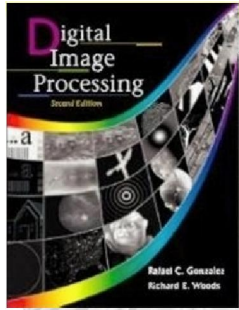
a	b
c	d
e	f

FIGURE 1.14

Some examples of manufactured goods often checked using digital image processing.

- (a) A circuit board controller.
- (b) Packaged pills.
- (c) Bottles.
- (d) Air bubbles in a clear-plastic product.
- (e) Cereal.
- (f) Image of intraocular implant.

(Fig. (f) courtesy of Mr. Pete Sites, Perceptics Corporation.)



Examples: Automated Visual Inspection



a b
c
d

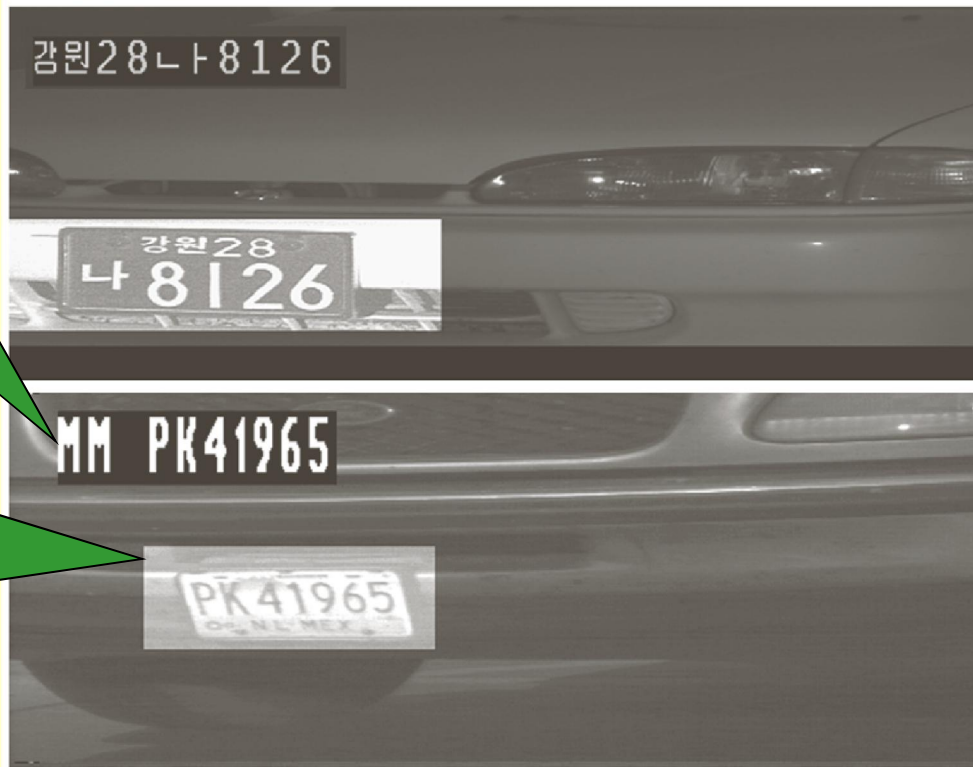
FIGURE 1.15

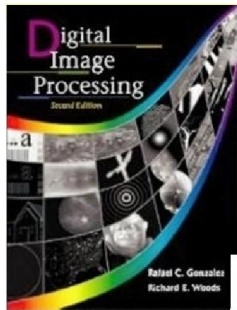
Some additional examples of imaging in the visual spectrum. (a) Thumb print. (b) Paper currency. (c) and (d) Automated license plate reading.

(Figure (a) courtesy of the National Institute of Standards and Technology. Figures (c) and (d) courtesy of Dr. Juan Herrera, Perceptics Corporation.)

Results of automated reading of the plate content by the system

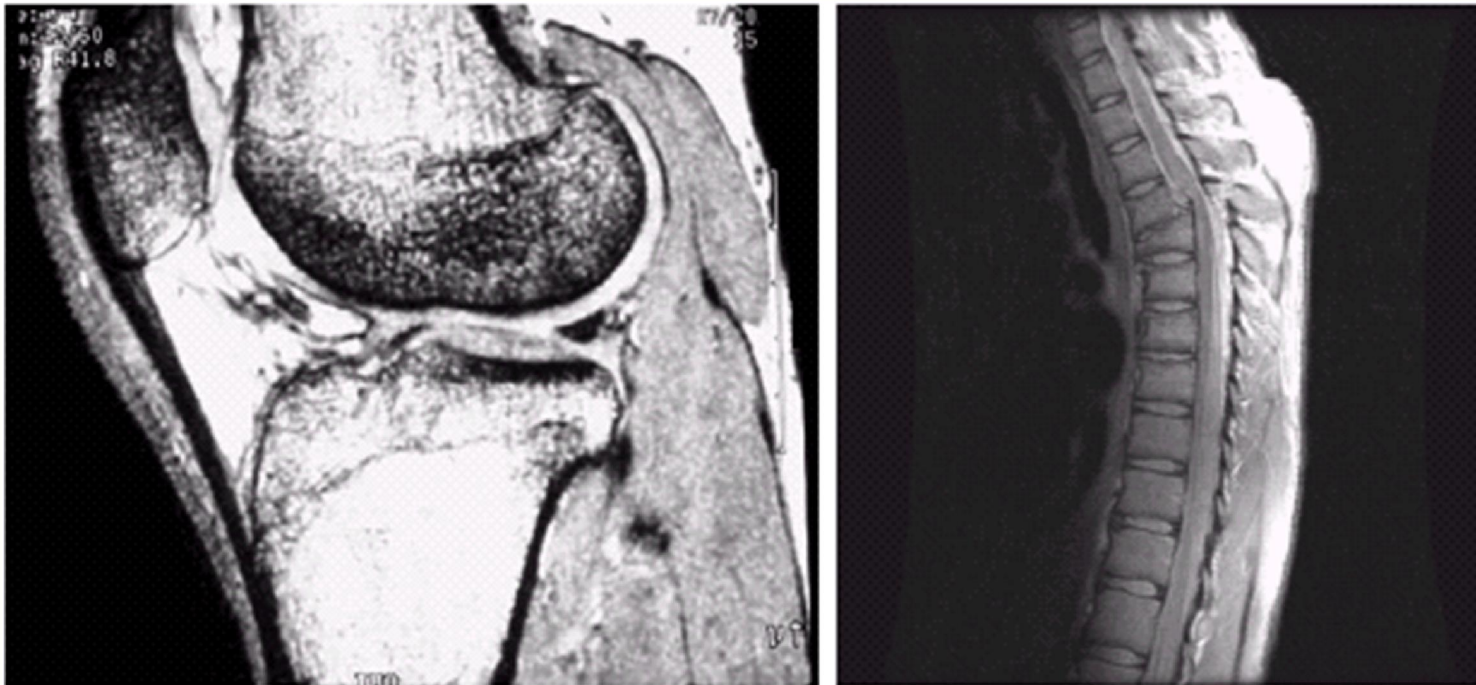
The area in which the imaging system detected the plate





Digital Image Processing, 2nd ed. Examples: MRI (Radio Band)

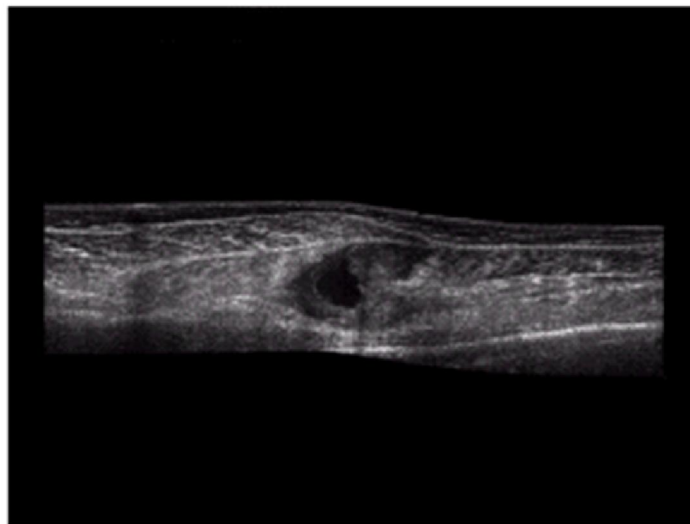
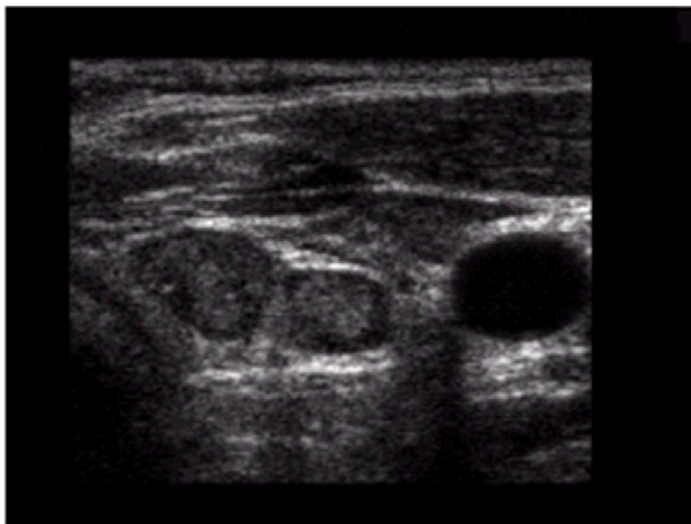
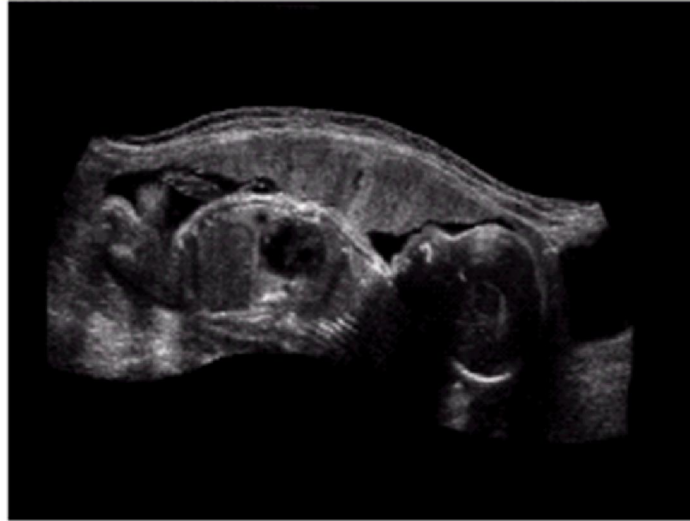
www.imageprocessingbook.com



a b

FIGURE 1.17 MRI images of a human (a) knee, and (b) spine. (Image (a) courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, and (b) Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Examples: Ultrasound Imaging



a	b
c	d

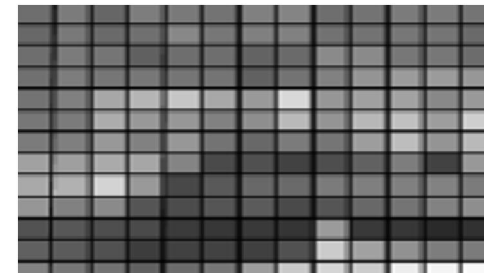
FIGURE 1.20
Examples of
ultrasound
imaging. (a) Baby.
(2) Another view
of baby.
(c) Thyroids.
(d) Muscle layers
showing lesion.
(Courtesy of
Siemens Medical
Systems, Inc.,
Ultrasound
Group.)

Lect 3 DIP steps

1-from lecture 2:→Digital Image Definition

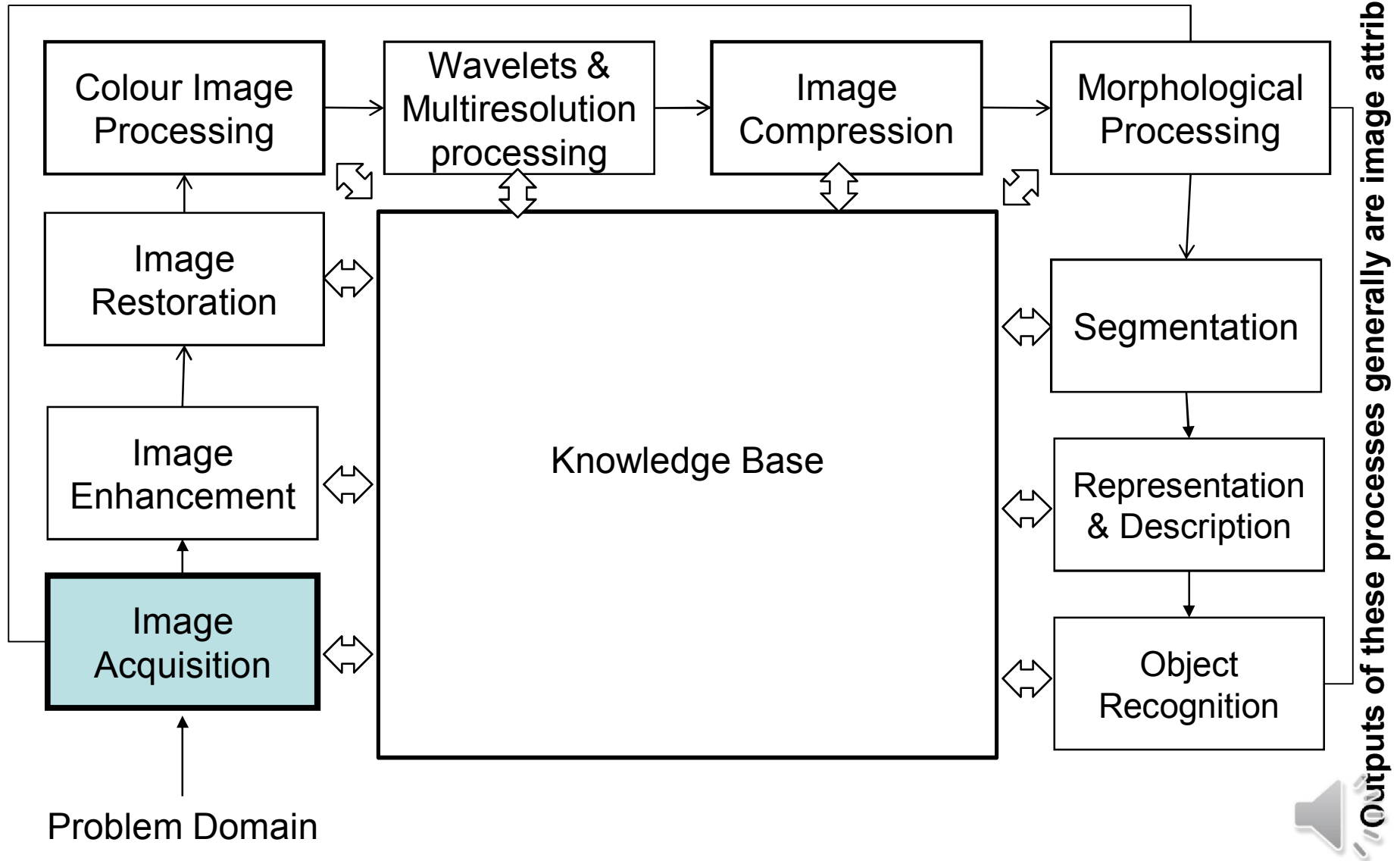
- An image can be defined as a two-dimensional function $f(x, y)$
- x, y : Spatial coordinate
- F : the amplitude of any pair of coordinate x, y , which is called the intensity or gray level of the image at that point.
- x, y and F , are all finite and discrete quantities.

54	48	48	52	67	111	144	160
54	48	48	49	61	106	141	160
48	45	48	49	56	97	138	160
50	51	57	56	61	101	135	161
59	60	61	55	60	103	134	162
62	61	55	44	49	96	133	163
56	45	53	54	41	99	137	163
55	45	55	56	42	94	136	164
53	45	58	59	44	86	134	162
54	47	61	60	46	79	131	160
57	51	63	58	49	75	133	162
63	57	62	54	52	74	138	166
70	62	61	49	54	77	139	166



Fundamental Steps in Digital Image Processing:

Outputs of these processes generally are images



Fundamental Steps in DIP

Step 1: Image Acquisition

The image is captured by a sensor (eg. **Camera**), and digitized if the output of the camera or sensor is not already in digital form, using **Analogue-to-Digital** convertor

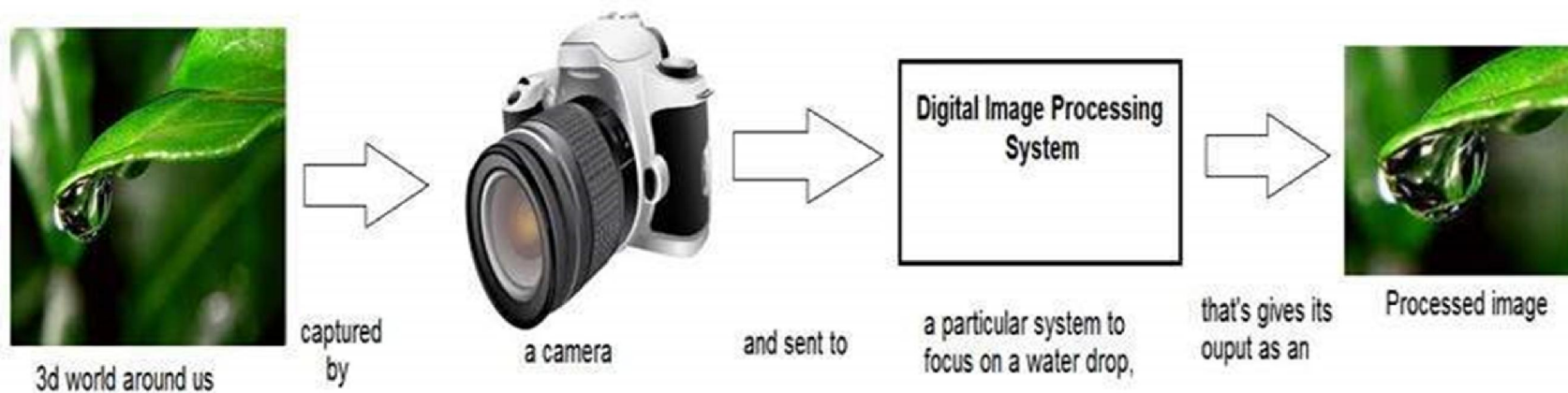
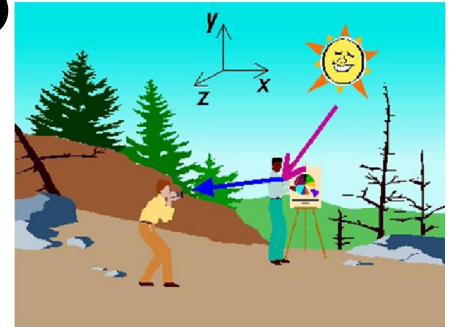


Image Acquisition equipment



Cont. Fundamental Steps in DIP:

Step 2: Image Enhancement

The process of manipulating an image so that the result is more suitable than the original for specific applications.

The idea behind enhancement techniques is to bring out details that are hidden, or simple to highlight certain features of interest in an image.

- Filtering with morphological operators.
- Histogram equalization.
- Noise removal using a Wiener filter.
- Linear contrast adjustment.
- Median filtering.
- Unsharp mask filtering.



Image Enhancement



Cont. Fundamental Steps in DIP:

Step 3: Image Restoration

- **Improving** the appearance of an image
- **Tend to be mathematical or probabilistic** models. Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result.



What is Image Restoration?

- Image restoration attempts to restore images that have been degraded
 - ✓ Identify the degradation process and attempt to reverse it.
 - ✓ Almost Similar to image enhancement, but more objective.

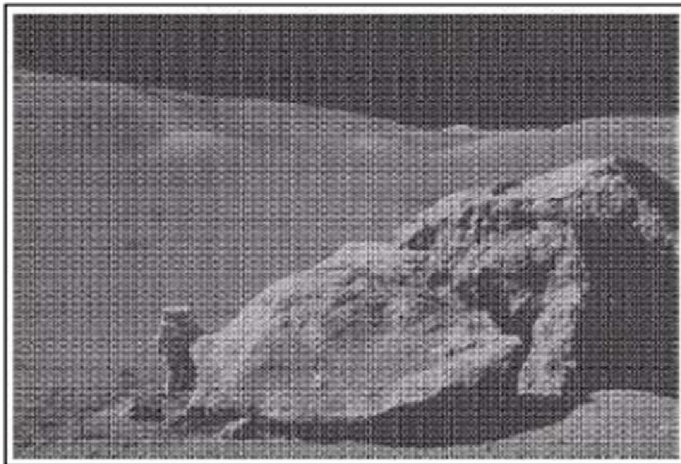


Fig: Degraded image

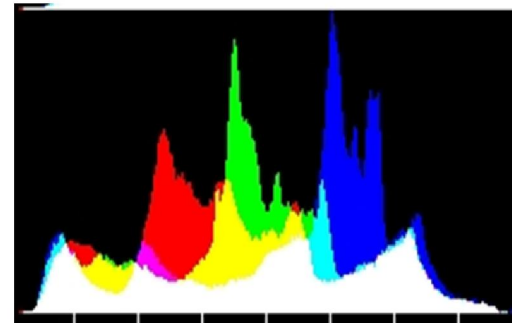


Fig: Restored image

Cont. Fundamental Steps in DIP:

Step 4: Colour Image Processing

Use the colour of the image to extract features of interest in an image



Color to grey and negative

origin



gray



negative

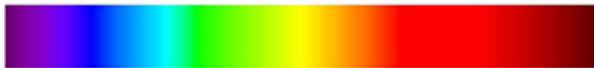


Color Issue of an Image

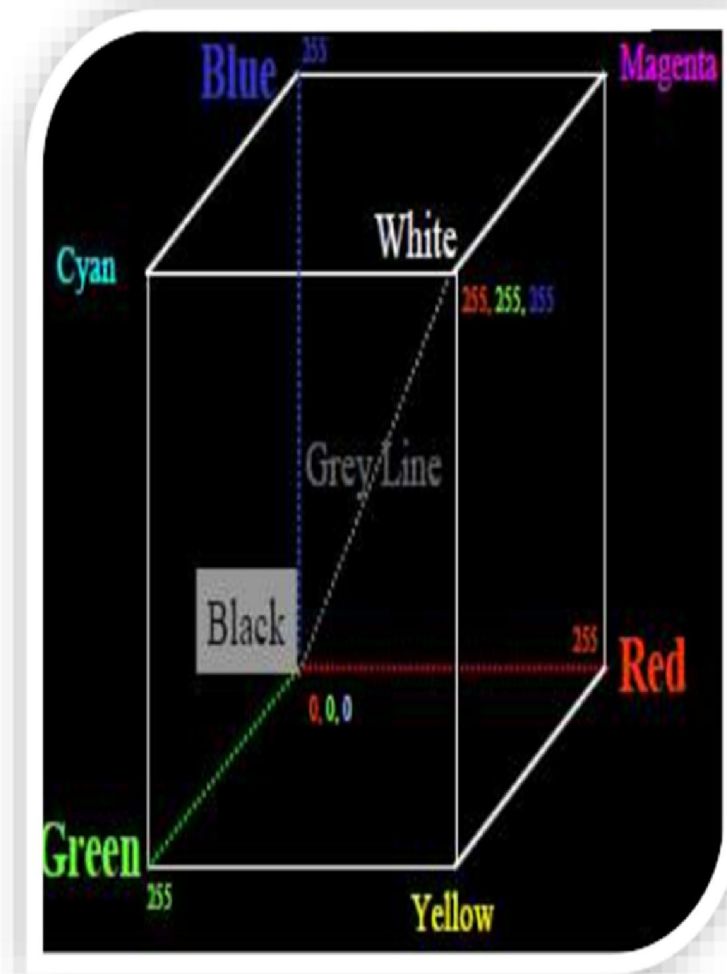
- Red, Green and Blue Color cube
- Consider Each element=8 bit
- R,G,B ~0 to 255
- Grey scale $f(x, y, L)$



- 256 Grey shades
- Color Scale $f(x, y, r, g, b)$ -24 bit



- $255 \times 255 \times 255 = 16777216$ colors



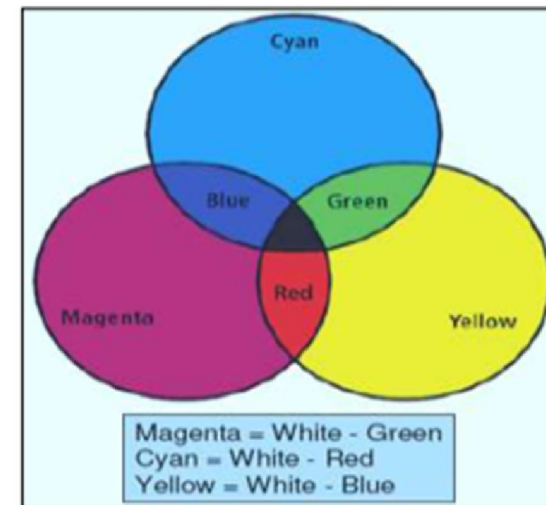
CMY and CMYK Color Model

- Cyan(C), Magenta(M) and Yellow(Y) are the secondary colors of light.
 - Or CMY are Primary colors of pigments.

- RGB to CMY

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Black=Cyan + Magenta + Yellow
- Printing Industry used to **four color Printing**.
- Cyan, Magenta, Yellow plus Black.



HIS Color Model

- RGB to HSI Conversion

$$I = \frac{1}{3}(R + G + B), \quad \text{where } 0 \leq I, R, G, B \leq 1$$

$$H = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}} \right\}, \quad \text{if } g_0 > b_0$$

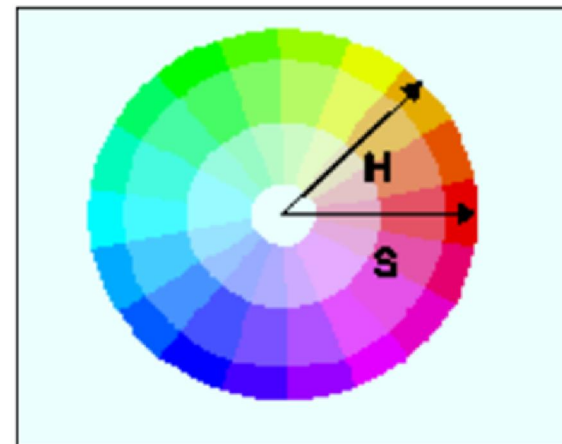
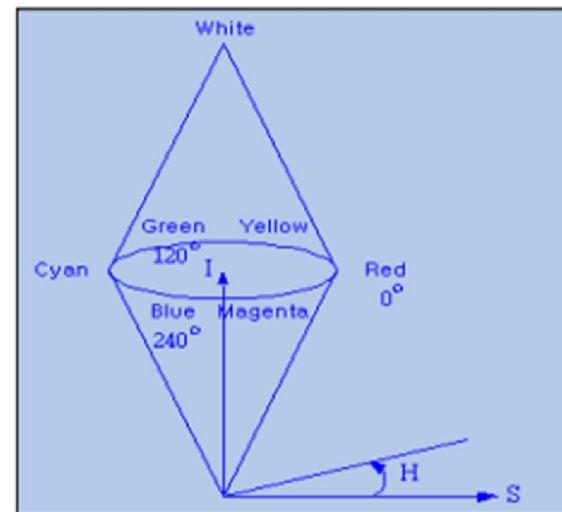
$$H = 360^\circ - H, \quad \text{if } g_0 < b_0 \quad \text{where } g_0 = G/I, \quad b_0 = B/I$$

$$S = 1 - \frac{3}{R + G + B} \times (\min\{R, G, B\})$$

-Hue : Range [0, 360]

-Saturation : Range[0, 1]

-Intensity : Range[0, 1]



Cont. Fundamental Steps in DIP:

Step 5: Wavelets

Are the foundation of representing images in various degrees of resolution. It is used for image data compression.

Fundamental Steps in DIP:

Step 6: Compression

Techniques for reducing the storage required to save an image or the bandwidth required to transmit it.

Image Compression?

- How we stored the image: Reduce the size for storage .
- How analog image world is relate to digital processing world.
- Compression-Remove redundancies.
- Transmission with minimum bandwidth.
- Lossy Compression=redundancy +some information, but still acceptable.



Original Image
Size-116 KB



Compressed Image
Size-12.9 KB, 11 %



Compressed Image
Size-1.95 KB, 16 %

Fundamental Steps in DIP:

Step 7: Morphological Processing

Tools for extracting image components that are useful in the representation and description of shape.

In this step, there would be a transition from processes that output images, to processes that output image attributes.

Morphological Image Processing

Extract image components that are useful in the representation and description of region shape, such as-

- Boundaries extraction
- Skeletons
- Convex hull
- Morphological filtering
- Thinning
- Pruning...many More



Fundamental Steps in DIP:

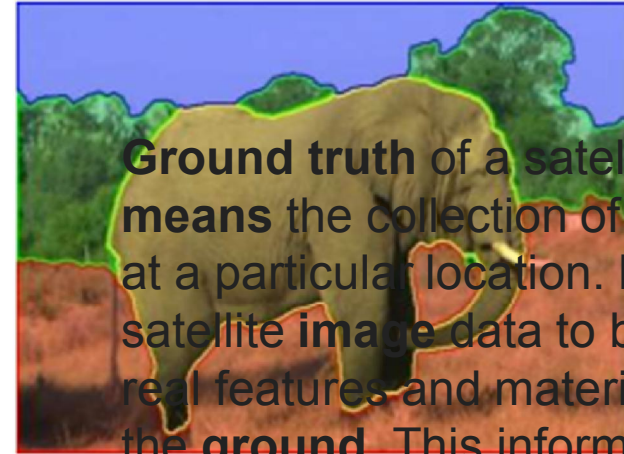
Step 8: Image Segmentation

Segmentation procedures partition an image into its constituent parts or objects.

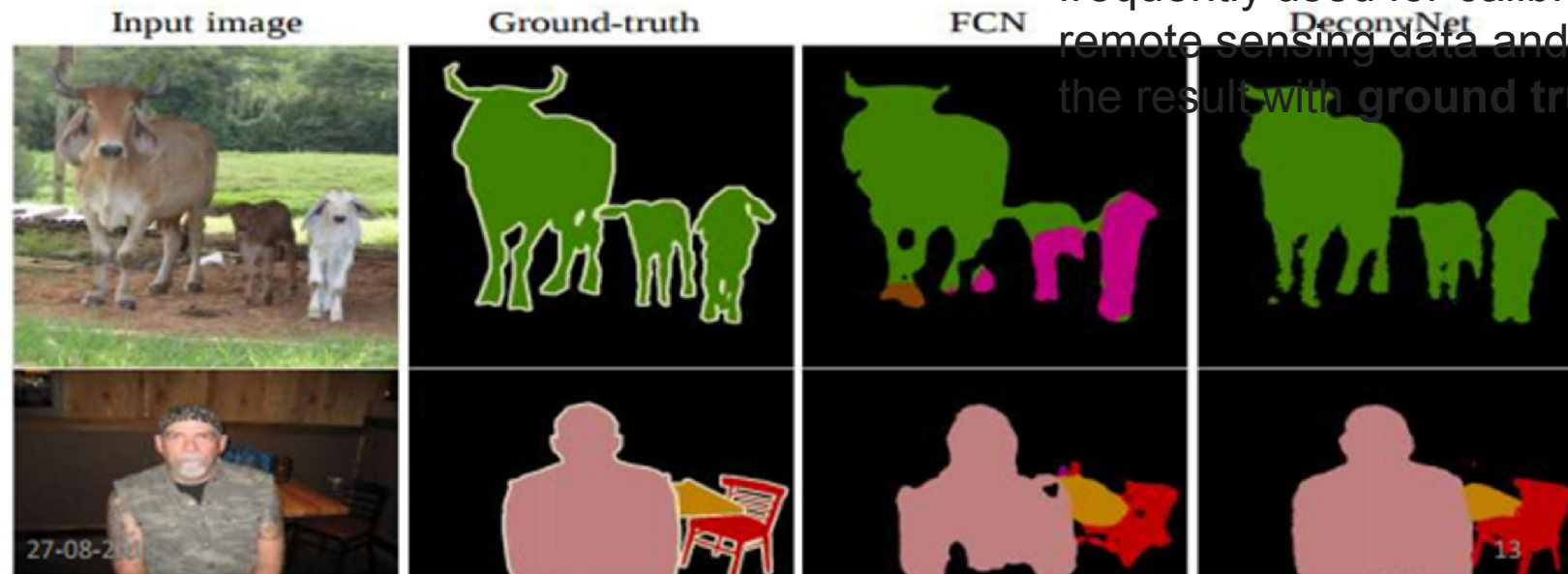
Important Tip: The more accurate the segmentation, the more likely recognition is to succeed.

Image Segmentation

In computer vision, **Image Segmentation** is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze



Ground truth of a satellite image means the collection of information at a particular location. It allows satellite **image** data to be related to real features and materials on the **ground**. This information is frequently used for calibration of remote sensing data and compares the result with **ground truth**



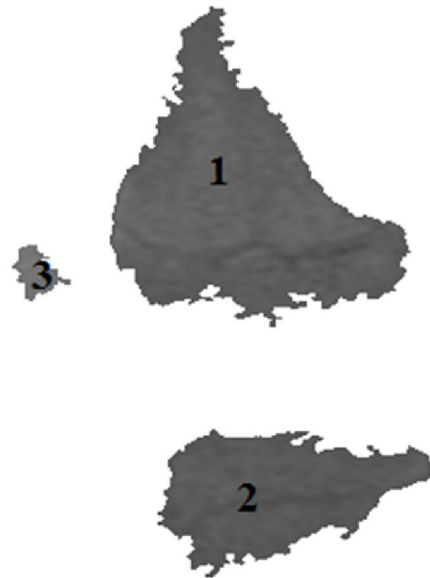
Fully convolutional network :Classify the object class for each pixel within an image. That means there is a label for each pixel.

Ground truths are “true and accurate” segmentations that are typically made by one or more human experts, so

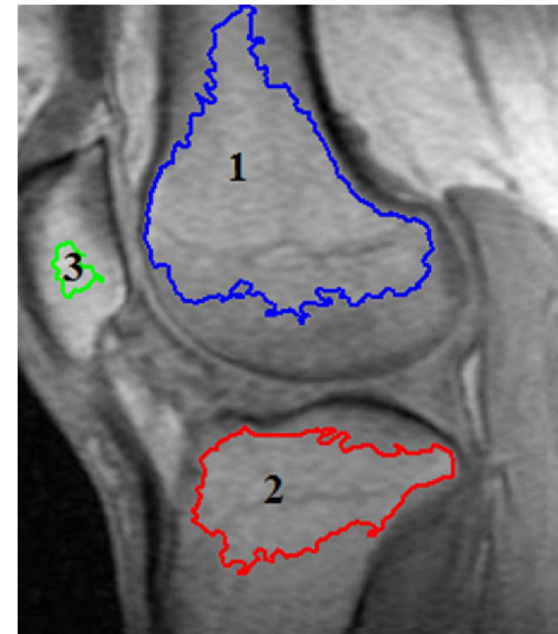
Fundamental Steps in DIP:

Step 9: Representation and Description

- **Representation:** Make a decision whether the data should be represented as a boundary or as a complete region. It almost always follows the output of a segmentation stage.
 - **Boundary Representation:** Focus on external shape characteristics, such as corners and inflections (انحناءات)
 - **Region Representation:** Focus on internal properties, such as texture or skeleton (هيكلية) shape



Extracted the three regions



Selected three regions on the original image



Calculated edge map of the segmented regions

Fundamental Steps in DIP:

Step 9: Representation and Description

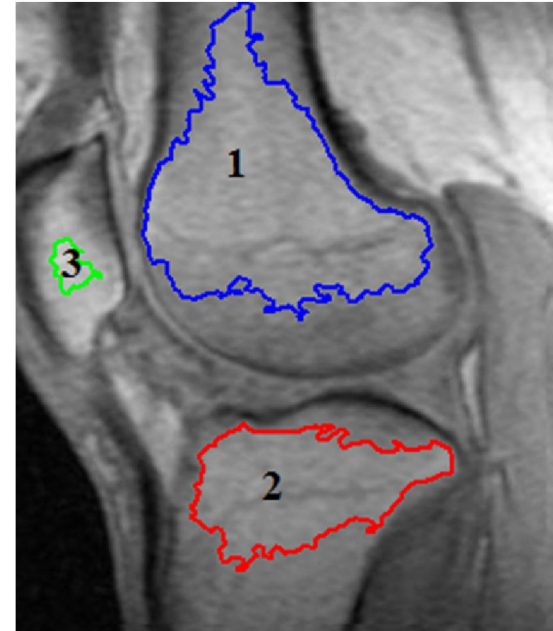
- Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing (mainly recognition)
- **Description:** also called, *feature selection*, deals with extracting attributes that result in some information of interest.

Fundamental Steps in DIP:

Step 9: Recognition and Interpretation

Recognition: the process that assigns label to an object based on the information provided by its description.

All the pixels in region 1 ,
Have label 1, and so on for
other regions, according
to some criteria



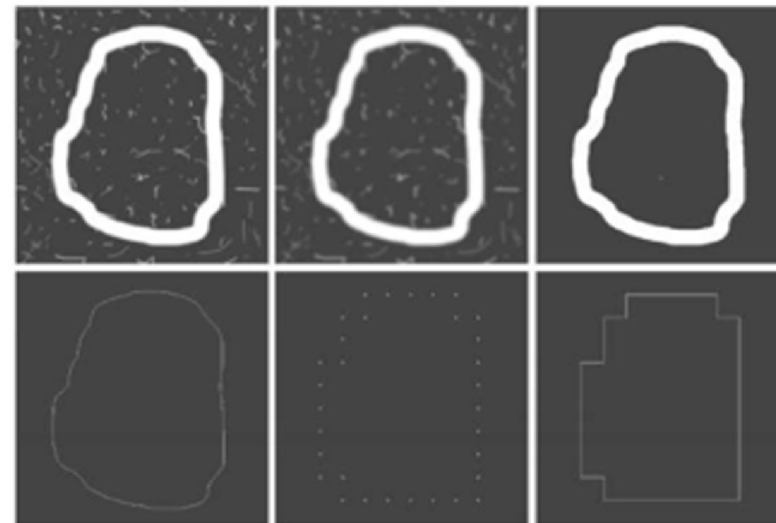
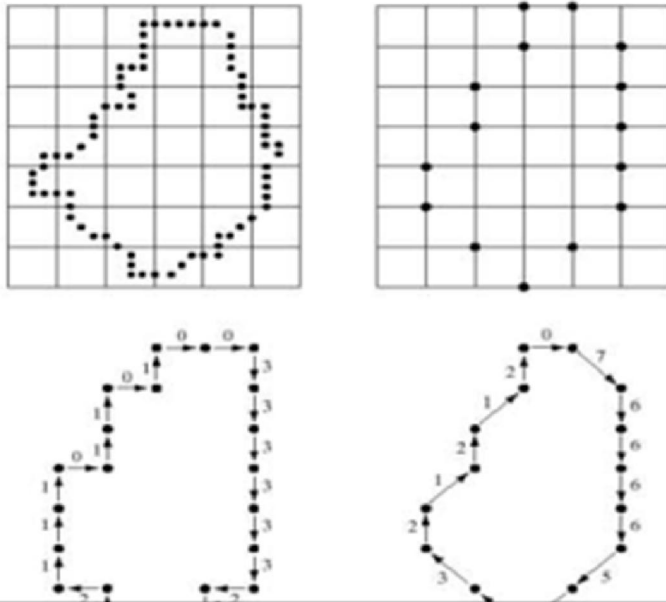
As a Conclusion:

Image Representation & Description

- Image representation & description: After an image is segmented into regions; the resulting aggregate of segmented pixels is represented & described for further computer processing.
- Representation and Description
 - Representing regions in 2 ways:
 - Based on their external characteristics (its boundary):
 - Shape characteristics
 - Based on their internal characteristics (its region):
 - Regional properties: color, texture, and ...
 - Both



Image Representation & Description

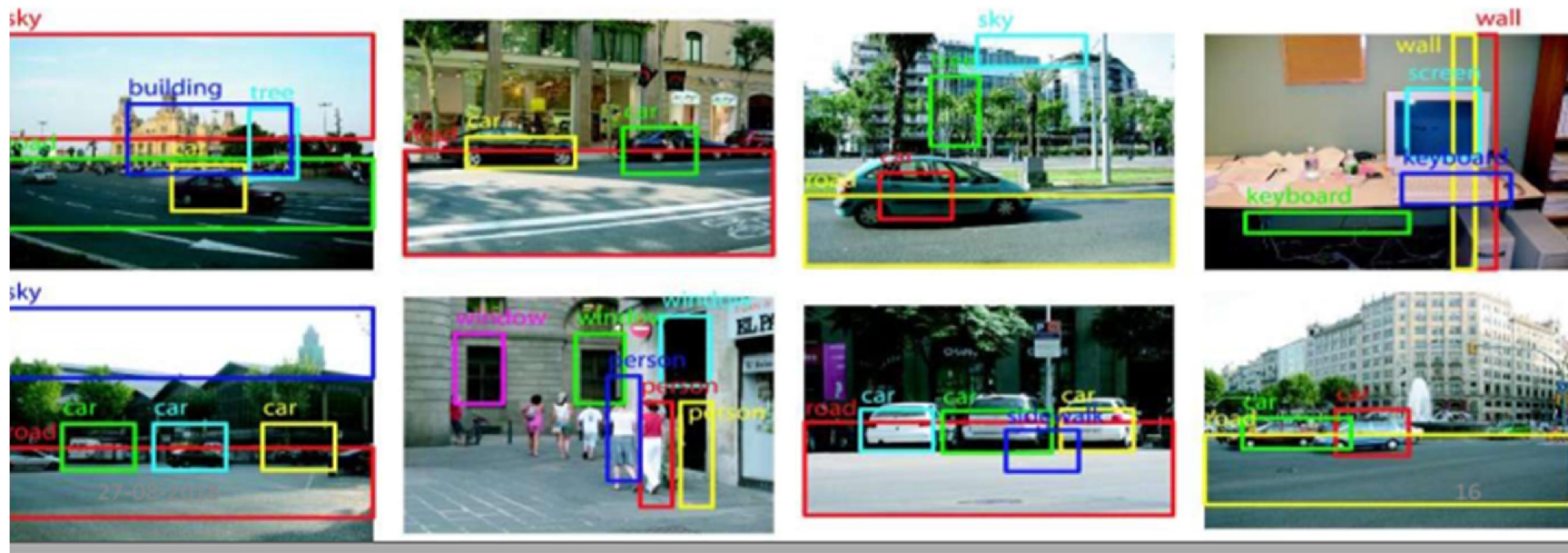


a b c
d e f

FIGURE 11.5 (a) Noisy image. (b) Image smoothed with a 9×9 averaging mask. (c) Smoothed image, thresholded using Otsu's method. (d) Longest outer boundary of (c). (e) Subsampled boundary (the points are shown enlarged for clarity). (f) Connected points from (e).

Object Recognition

- Object Detection is the process of finding instances of objects in images. This allows for multiple objects to be identified and located within the same image.
- Object recognition can be termed as identifying a specific object in a digital image or video. Object recognition has immense applications in the field of monitoring and surveillance, medical analysis, robot localization and navigation etc.



Fundamental Steps in DIP:

Step 10: Knowledge Base

Knowledge about a problem domain is coded into an image processing system in the form of a knowledge database.

The DIP steps can be summarized as:

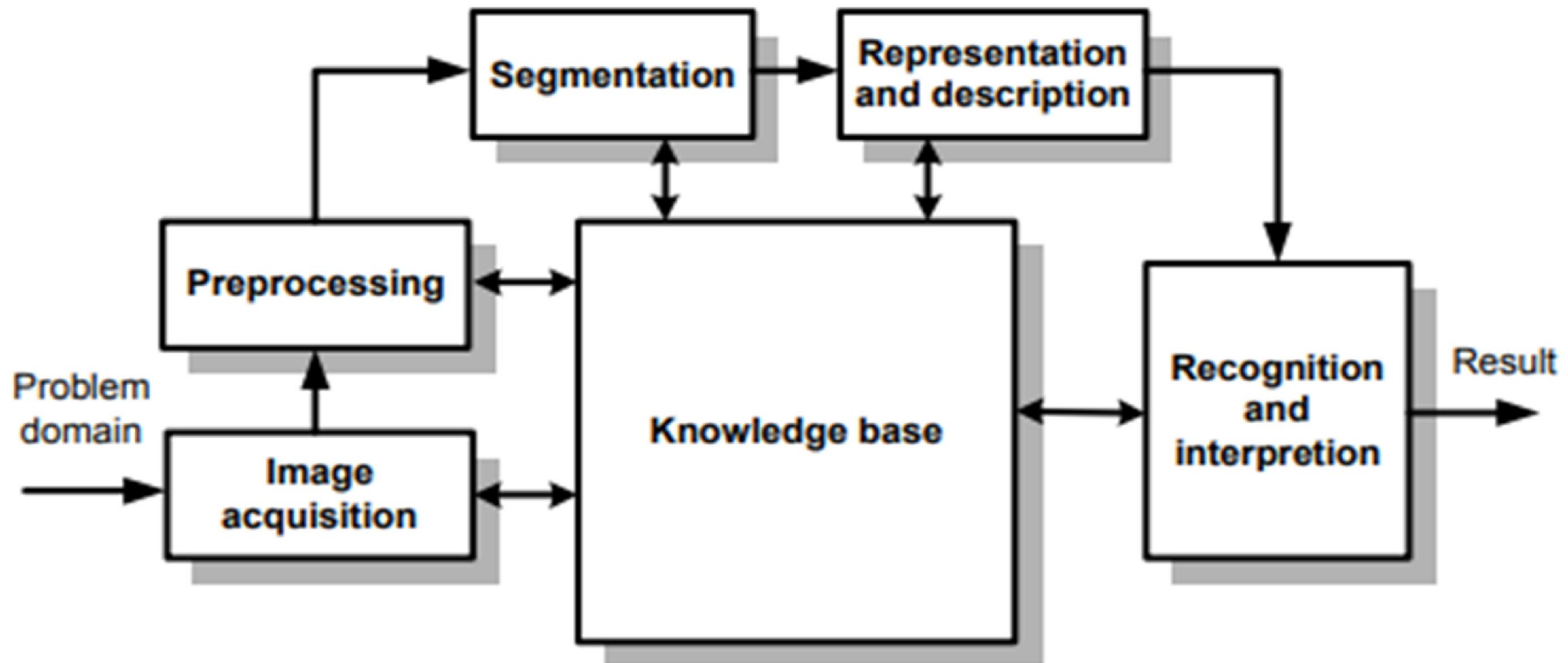


Fig 1. Fundamental steps in digital image processing



Elements of digital image processing systems:

- The basic operations performed in a digital image processing systems include (1) acquisition, (2) storage, (3) processing, (4) communication and (5) display.

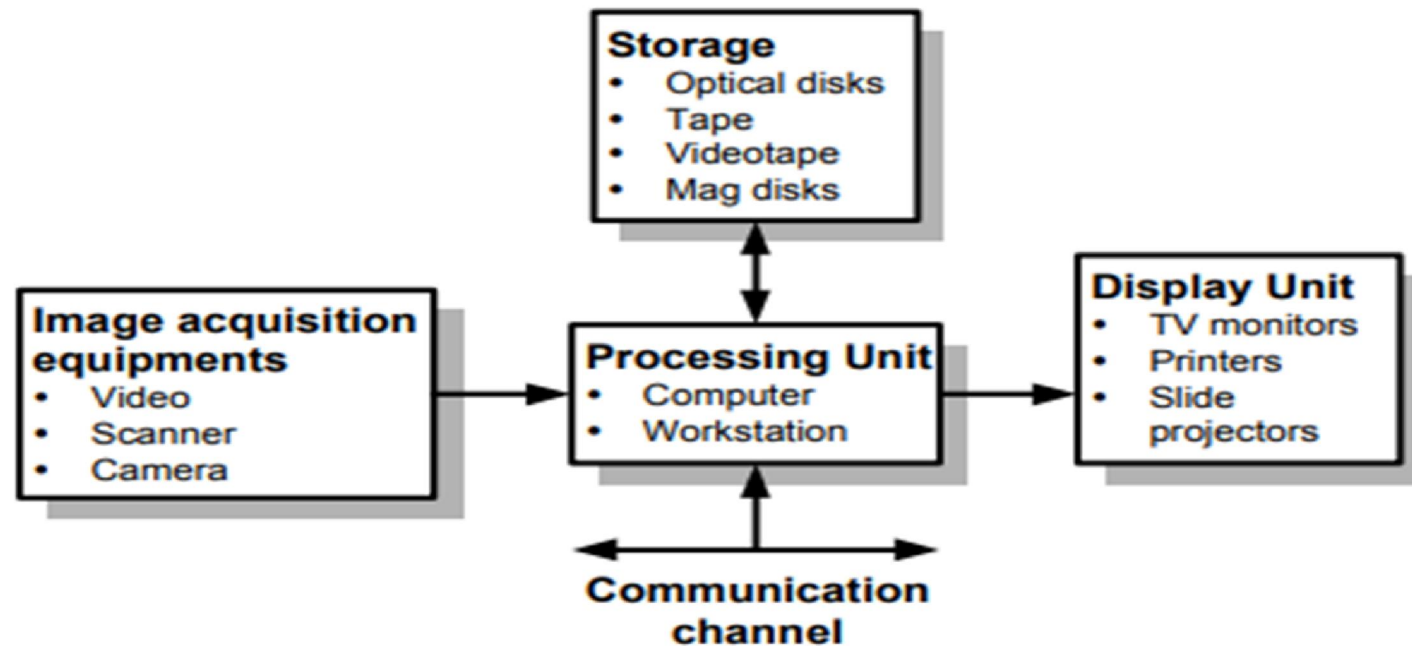
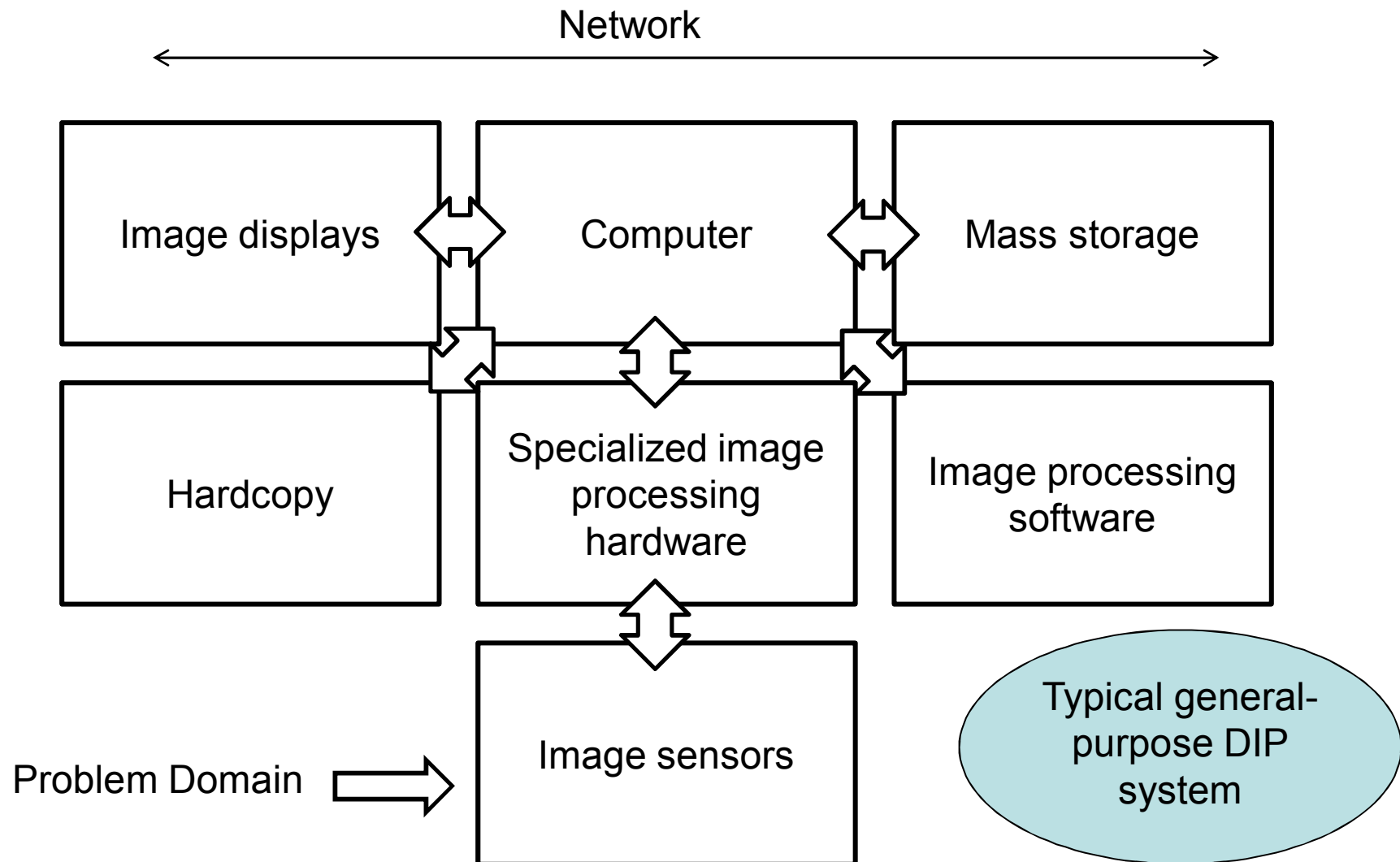


Fig 2. Basic fundamental elements of an image processing system

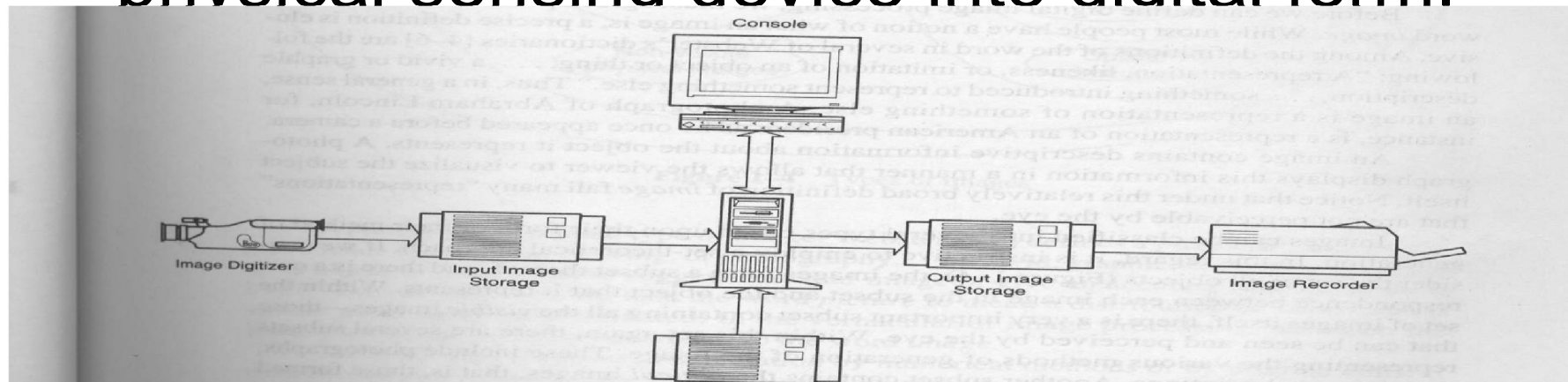
Components of an Image Processing System



Components of an Image Processing System

1. Image Sensors

Two elements are required to acquire digital images. The first is the physical device that is sensitive to the energy radiated by the object we wish to image (*Sensor*). The second, called a *digitizer*, is a device for converting the output of the physical sensing device into digital form.



Components of an Image Processing System

2. Specialized Image Processing Hardware

Usually consists of the digitizer, mentioned before, plus hardware that performs other primitive operations, such as an arithmetic logic unit (ALU), which performs arithmetic and logical operations in parallel on entire images.

This type of hardware sometimes is called a front-end subsystem, and its most distinguishing characteristic is speed. In other words, this unit performs functions that require fast data throughputs that the typical main computer cannot handle.

Components of an Image Processing System

3. Computer

The computer in an image processing system is a general-purpose computer and can range from a PC to a supercomputer. In dedicated applications, sometimes specially designed computers are used to achieve a required level of performance.

Components of an Image Processing System

4. Image Processing Software

Software for image processing consists of specialized modules that perform specific tasks. A well-designed package also includes the capability for the user to write code that, as a minimum, utilizes the specialized modules.

Components of an Image Processing System

5. Mass Storage Capability

Mass storage capability is a must in a image processing applications. And image of sized $1024 * 1024$ pixels requires one megabyte of storage space if the image is not compressed.

Digital storage for image processing applications falls into three principal categories:

1. Short-term storage for use during processing.
2. on line storage for relatively fast recall
3. Archival storage, characterized by infrequent access

Components of an Image Processing System

5. Mass Storage Capability

One method of providing short-term storage is computer memory. Another is by specialized boards, called frame buffers, that store one or more images and can be accessed rapidly.

The on-line storage method, allows virtually instantaneous image zoom, as well as scroll (vertical shifts) and pan (horizontal shifts). On-line storage generally takes the form of magnetic disks and optical-media storage. The key factor characterizing on-line storage is frequent access to the stored data.

Finally, archival storage is characterized by massive storage requirements but infrequent need for access.

Components of an Image Processing System

6. Image Displays

The displays in use today are mainly color (preferably flat screen) TV monitors. Monitors are driven by the outputs of the image and graphics display cards that are an integral part of a computer system.

Components of an Image Processing System

7. Hardcopy devices

Used for recording images, include laser printers, film cameras, heat-sensitive devices, inkjet units and digital units, such as optical and CD-Rom disks.

Components of an Image Processing System

8. Networking

Is almost a default function in any computer system, in use today. Because of the large amount of data inherent in image processing applications the key consideration in image transmission is bandwidth.

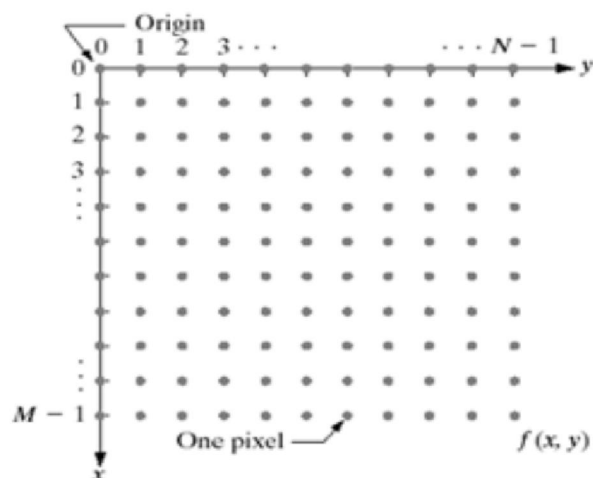
In dedicated networks, this typically is not a problem, but communications with remote sites via the internet are not always as efficient.



Images as Matrices

مراجعة المحاضرة ٣

- Recalling the **image formation operations** we have discussed, note that the image $\hat{f}_{gray}(i, j)$ is an $N \times M$ matrix with integer entries in the range $0, \dots, 255$.
- From now on suppress $(\hat{\cdot})_{gray}$ and denote an image as a matrix "A" (or B, ..., etc.) with elements $A(i, j) \in \{0, \dots, 255\}$ for $i = 0, \dots, N - 1$, $j = 0, \dots, M - 1$.
- So we will be processing matrices!

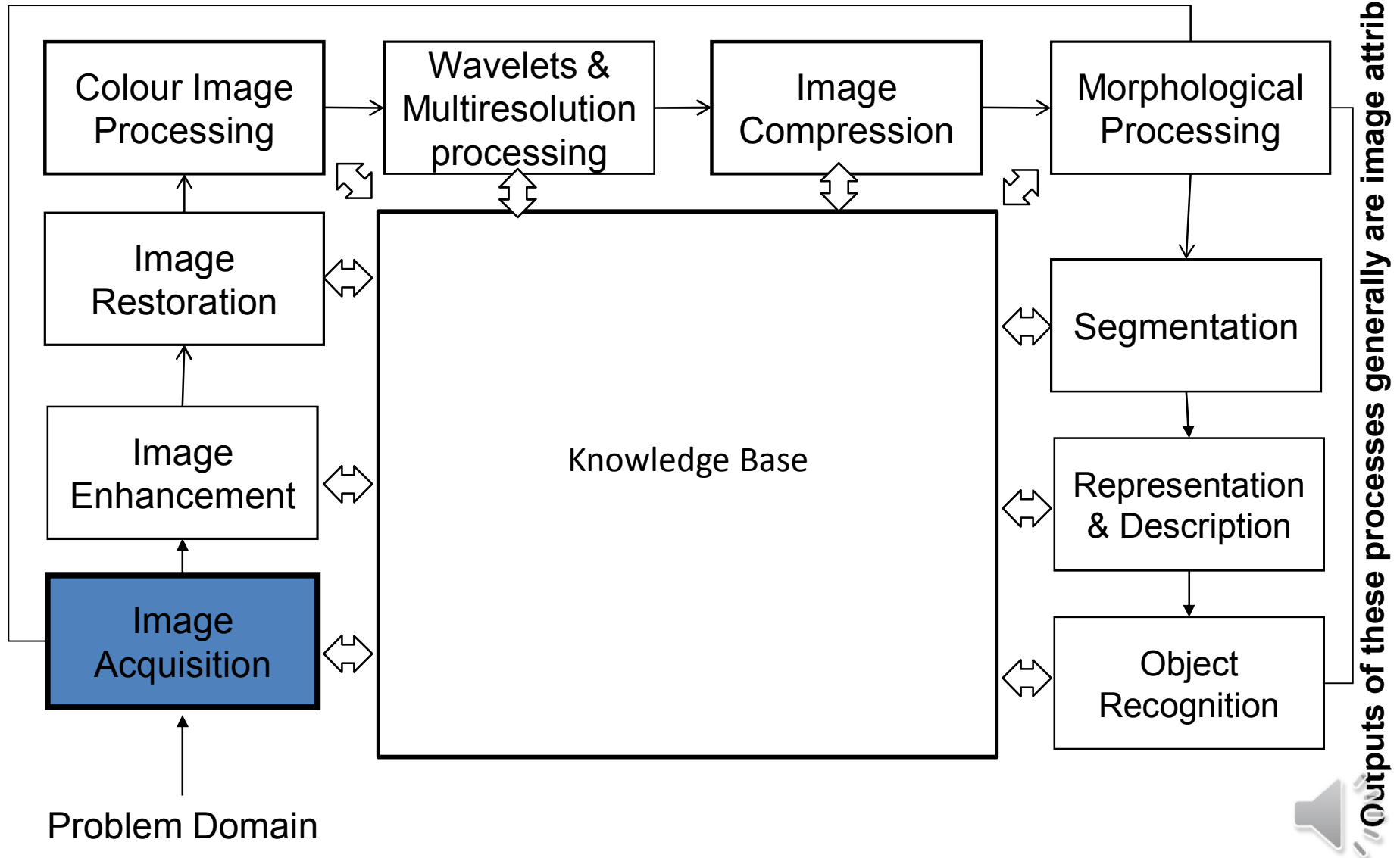


54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

Fundamental Steps in Digital Image Processing:

مراجعة المحاضرة ٣

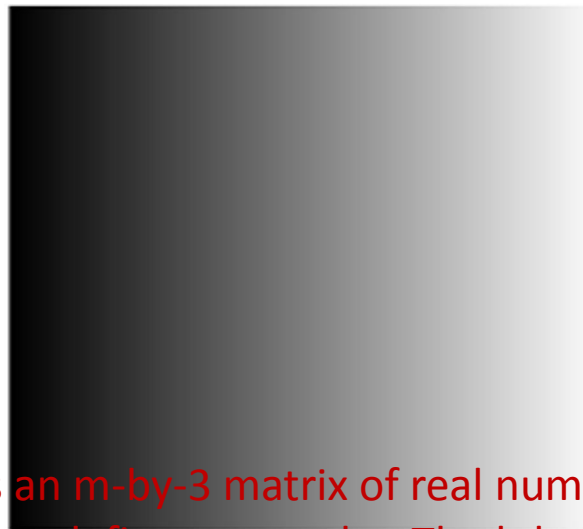
Outputs of these processes generally are images



Example - I

- The image of a ramp (256×256):

$$A = \left[\begin{array}{ccccc} 0 & 1 & 2 & \dots & 255 \\ 0 & 1 & 2 & \dots & 255 \\ \vdots & & & & \\ 0 & 1 & 2 & \dots & 255 \end{array} \right] \left. \vphantom{\begin{array}{ccccc} 0 & 1 & 2 & \dots & 255 \\ 0 & 1 & 2 & \dots & 255 \\ \vdots & & & & \\ 0 & 1 & 2 & \dots & 255 \end{array}} \right\} 256 \text{ rows } \textcircled{?}$$



```
>> for i = 1 : 256
    for j = 1 : 256
        A(i,j) = j - 1;
    end
end
>> image(A);
>> colormap(gray(256));
>> axis('image');
```

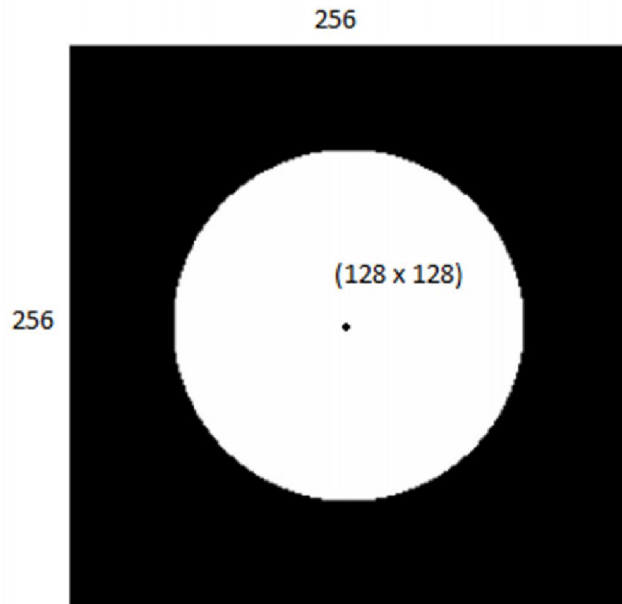
A colormap is an m -by-3 matrix of real numbers between 0.0 and 1.0. Each row is an RGB vector that defines one color. The k th row of the colormap defines the k th color, where $\text{map}(k,:) = [r(k) \ g(k) \ b(k)]$ specifies the intensity of red, green, and blue. `colormap(map)` sets the colormap to the matrix `map`. If any values in `map` are outside the interval $[0 \ 1]$, MATLAB returns the error Colormap must have values in $[0,1]$.

2nd example

- The image of a circle (256×256) of radius 80 pixels centered at (128, 128):



$$B(i, j) = \begin{cases} 255 & \text{if } \sqrt{(i - 128)^2 + (j - 128)^2} < 80 \\ 0 & \text{otherwise} \end{cases}$$

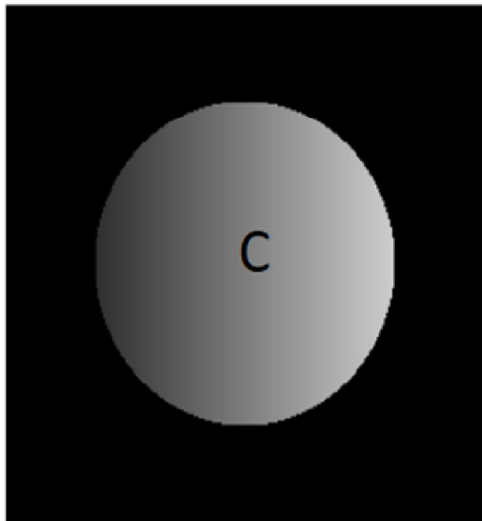


```
>> for i = 1 : 256
    for j = 1 : 256
        dist = ((i - 128)^2 + (j - 128)^2)^(.5);
        if (dist < 80)
            B(i, j) = 255;
        else
            B(i, j) = 0;
        end
    end
end
>> image(B);
>> colormap(gray(256));
>> axis('image');
```

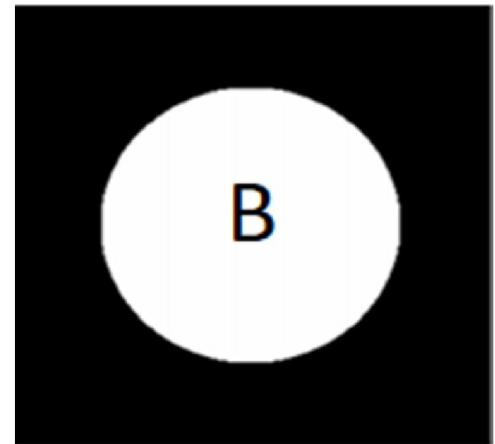
Example 3

- The image of a “graded” circle (256×256):

$$C(i, j) = A(i, j) \times B(i, j) / 255$$



```
>> for i = 1 : 256
    for j = 1 : 256
        C(i, j) = A(i, j) * B(i, j) / 255;
    end
end
>> image(C);
>> colormap(gray(256));
>> axis('image');
```





Images as Matrices

- An image matrix ($N \times M$):

$$\mathbf{A} = \left[\begin{array}{cccc} A(0,0) & A(0,1) & A(0,2) & \dots A(0,M-1) \\ A(1,0) & A(1,1) & A(1,2) & \dots A(1,M-1) \\ \vdots & & & \\ A(N-1,0) & A(N-1,1) & A(N-1,2) & \dots A(N-1,M-1) \end{array} \right] \left. \vphantom{\begin{array}{c} A(0,0) \\ A(1,0) \\ \vdots \\ A(N-1,0) \end{array}} \right\} N \text{ rows } \textcircled{?}$$

- $A(i,j) \in \{0,1,\dots,255\}$.
- $A(i,j)$:
 - “Matrix case:” The matrix element (i,j) with value $A(i,j)$.
 - “Image case:” The pixel (i,j) with value $A(i,j)$.
 - Will use both terminologies.

Image Statistics

54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

➤ **Arithmetic Mean,** المحاضرة ٤

➤ **Standard Deviation,** الخصائص الاحصائية للصورة الرقمية

➤ **and Variance**

- Useful statistical features of an image **are its arithmetic mean,**
- **standard deviation, and variance.** These are well known mathematical constructs that, when applied to a digital image, can reveal important information.
- **The arithmetic mean is the image's average value.**
- The standard deviation is a measure of the frequency distribution, or range of pixel values, of an image. If an image is supposed to be uniform throughout, the standard deviation should be small. *A small standard deviation indicates that the pixel intensities do not stray very far from the mean; a large value indicates a greater range.*
- The standard deviation is the square root of the variance.
- **The variance** is a measure of how spread out a distribution is. It is computed as the average squared deviation of each number from its mean

• The **sample mean** (m_A) of an image A ($N \times M$):

$$m_A = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} A(i,j)}{NM}$$

• The **sample variance** (σ_A^2) of A :

$$\sigma_A^2 = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A(i,j) - m_A)^2}{NM}$$



Simple Image Statistics - Sample Mean and Sample Variance

- The **sample mean** (m_A) of an image A ($N \times M$):

$$m_A = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} A(i, j)}{NM} \quad (1)$$

- The **sample variance** (σ_A^2) of A :

$$\sigma_A^2 = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A(i, j) - m_A)^2}{NM} \quad (2)$$

- The **sample standard deviation**, $\sigma_A = \sqrt{\sigma_A^2}$.

THE VARIANCE

- The variance is a measure of how spread out a distribution is. It is computed as the average squared deviation of each number from its mean. For example, for the numbers 1, 2, and 3, the mean is 2 and the variance is $(1+2+3)/3=2$

-

$$\sigma^2 = \frac{(1-2)^2 + (2-2)^2 + (3-2)^2}{3} = .667$$

The formula (in summation notation) for the variance in a population is

$$\sigma^2 = \frac{\sum (X - \mu)^2}{N}$$

where μ is the mean and N is the number of scores.

- The **sample variance** (σ_A^2) of A :

$$\sigma_A^2 = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A(i, j) - m_A)^2}{NM}$$



Simple Image Statistics - Histogram

Let S be a set and define $\#S$ to be the cardinality of this set, i.e., $\#S$ is the number of elements of S .

- The **histogram** $h_A(l)$ ($l = 0, \dots, 255$) of the image A is defined as:

$$h_A(l) = \#\{(i, j) \mid A(i, j) = l, i = 0, \dots, N - 1, j = 0, \dots, M - 1\} \quad (3)$$

- Note that:

$$\sum_{l=0}^{255} h_A(l) = \text{Number of pixels in } A \quad (4)$$



```
OR      >> h=zeros(256,1);
        >> for l = 0 : 255
            h(l+1)=sum(sum(A == l));
        end
        >> bar(0:255,h);
```

[illegible]

The vertical flipped image B ($N \times M$) of A ($N \times M$) can be obtained as $B(i, M + 1 - j) = A(i, j)$ ($i = 0, \dots, N - 1; j = 0, \dots, M - 1$).



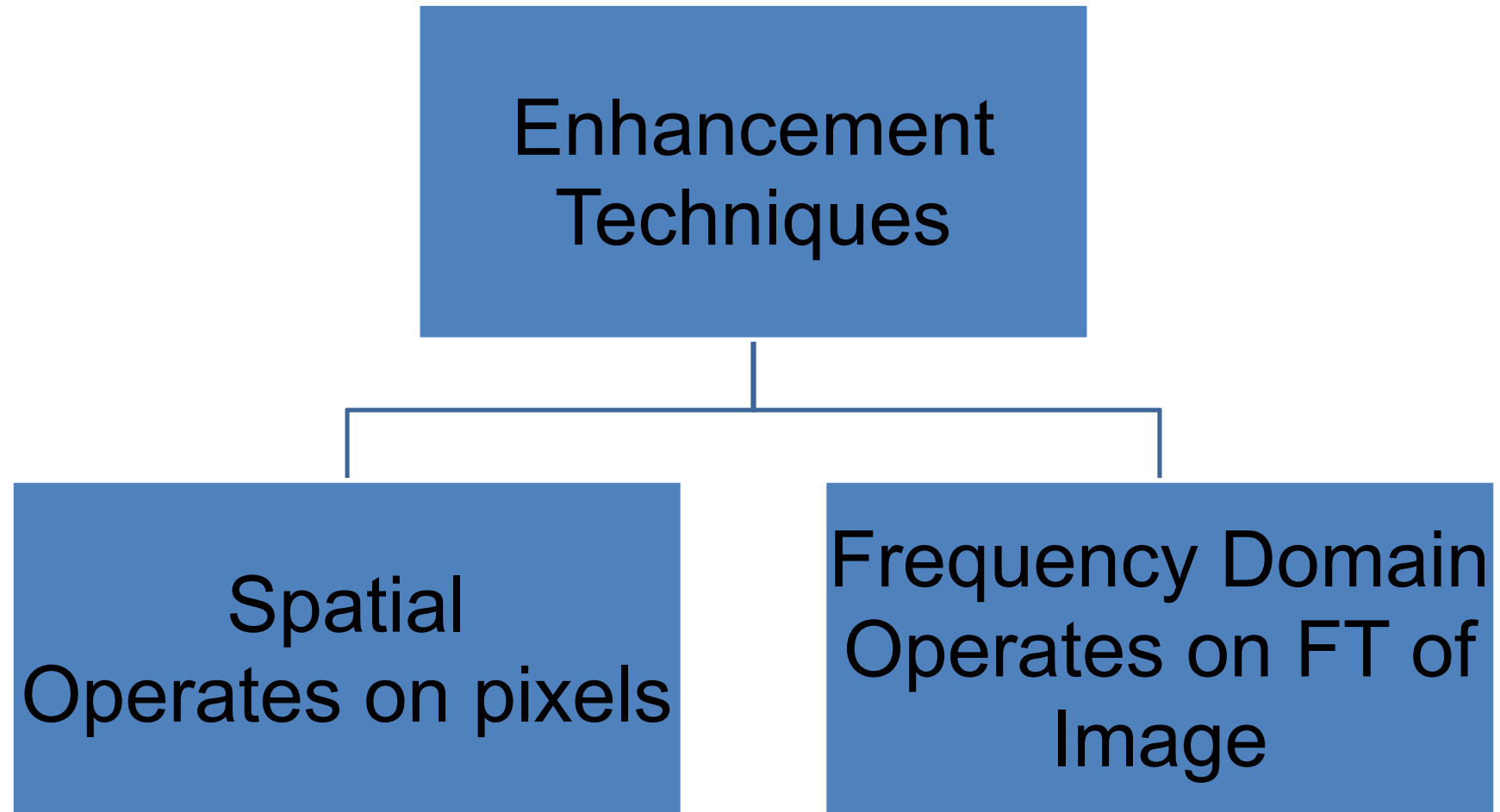
```
clear B;  
A=imread('c:\lena.jpg');  
for i = 1 : 380  
for j = 1 : 380  
B(i, 380 + 1 - j) = A(i, j);  
end  
  
figure  
subplot(1,2,1)  
imshow(A)  
subplot(1,2,2)  
imshow(B)
```

The cropped image B ($N1 \times N2$) of A ($N \times M$), starting from $(n1, n2)$, can be obtained as $B(k, l) = A(n1+k, n2+l)$ ($k = 0, \dots, N1-1; l = 0, \dots, N2-1$).



```
A=imread('c:\lena.jpg');  
for k = 1 : 64  
    for j = 1 : 128  
        B(k,j) = A(220+k,220+j);  
    end  
end  
figure  
subplot(1,2,1)  
imshow(A)  
subplot(1,2,2)  
imshow(B)
```

تقنيات تحسين الصورة الرقمية مع الامثلة



Intensity Transformations and Spatial Filtering:

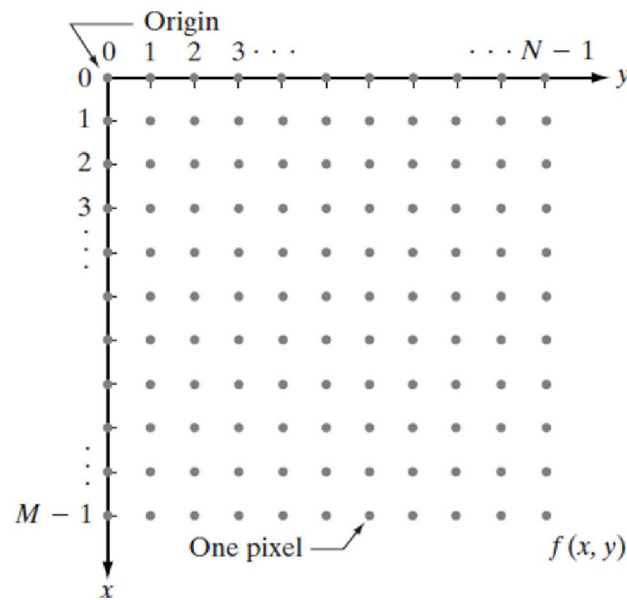
Intensity transformation Functions

54	48	48	52	67	111	144	160	1
54	48	48	49	61	106	141	160	1
48	45	48	49	56	97	138	160	1
50	51	57	56	61	101	135	161	1
59	60	61	55	60	103	134	162	1
62	61	55	44	49	96	133	163	1
56	45	53	54	41	99	137	163	1
55	45	55	56	42	94	136	164	1
53	45	58	59	44	86	134	162	1
54	47	61	60	46	79	131	160	1
57	51	63	58	49	75	133	162	1
63	57	62	54	52	74	138	166	1
70	62	61	49	54	77	139	166	1

Making changes in the intensity is done through *Intensity Transformation Functions*

- photographic negative (using `imcomplement`)
- gamma transformation (using `imadjust`)
- logarithmic transformations (using `c*log(1+f)`)
- contrast-stretching transformations
(using `1./(1+(m./(double(f)+eps)).^E)`)

Representing digital image



value $f(x, y)$ at each x, y is called ***intensity level*** or ***gray level***

Intensity Transformations and Filters

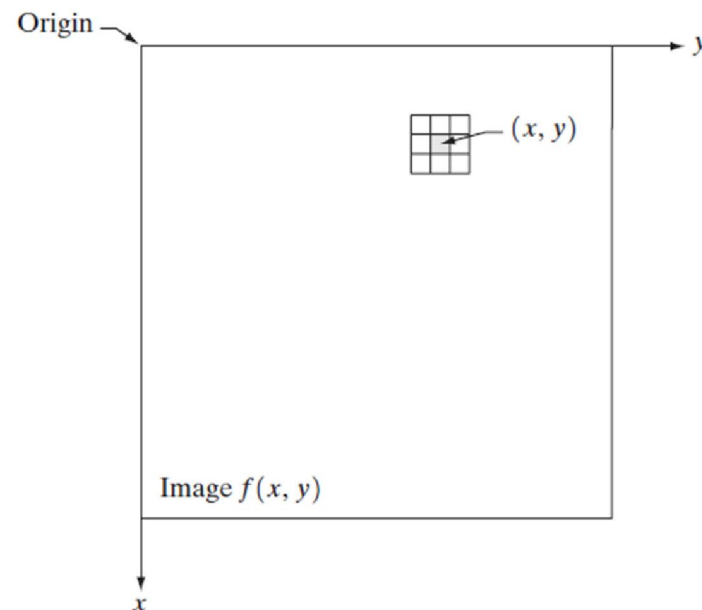
$$g(x,y)=T[f(x,y)]$$

$f(x,y)$ – input image,

$g(x,y)$ – output image

T is an operator on f defined over a neighborhood of point (x,y)

FIGURE 3.1 A
 3×3
neighborhood
about a point
 (x, y) in an image.



Intensity Transformation

- 1×1 is the smallest possible neighborhood.
- In this case g depends only on value of f at a single point (x,y)
and we call T an *intensity (gray-level mapping) transformation* and write

$$s = T(r)$$

where s and r denotes respectively the intensity of g and f at any point (x, y) .

Spatial Domain Methods

- In these methods a operation (**linear or non-linear**) is performed on the pixels in the neighborhood of coordinate **(x,y)** in the input image **F**, giving enhanced image **F'**
- Neighborhood can be any shape but generally it is rectangular (**3x3, 5x5, 9x9** etc)

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

Grey Scale Manipulation

- Simplest form of window (**1x1**)
- Assume input gray scale values are in **range [0, L-1]** (in 8 bit images L = 256)

- **nth root Transformation**

$$s = c (r)^n$$

- S is output image, r input im
- **A=imread('d:\flowers.jpg');**
- A=rgb2gray(A);
- C=1;
- n=.5;
- **B=C*double(A).^n;**
- figure
- subplot(1,2,1)
- imshow(A,[])
- subplot(1,2,2)
- imshow(B,[])



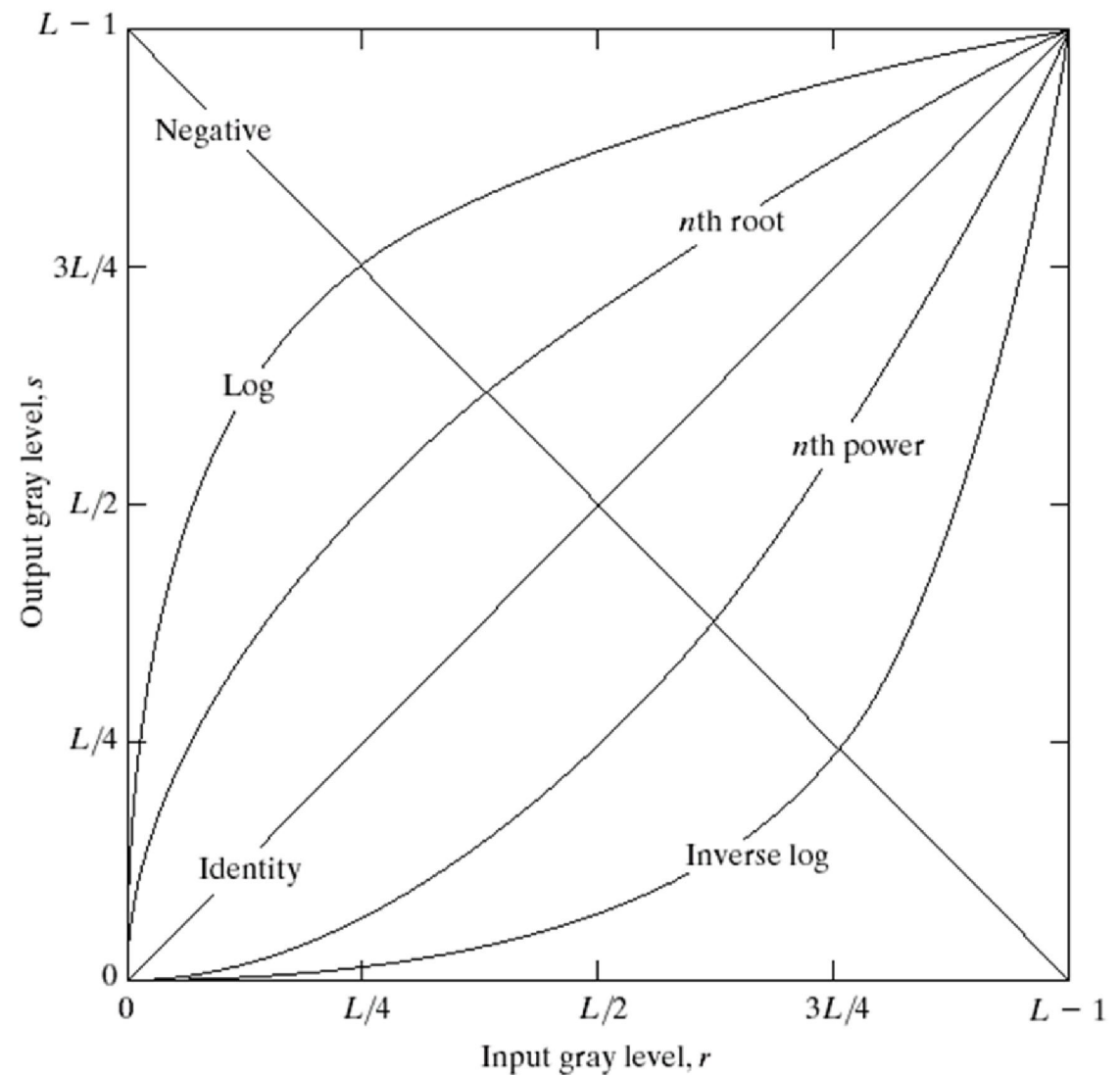
before

after

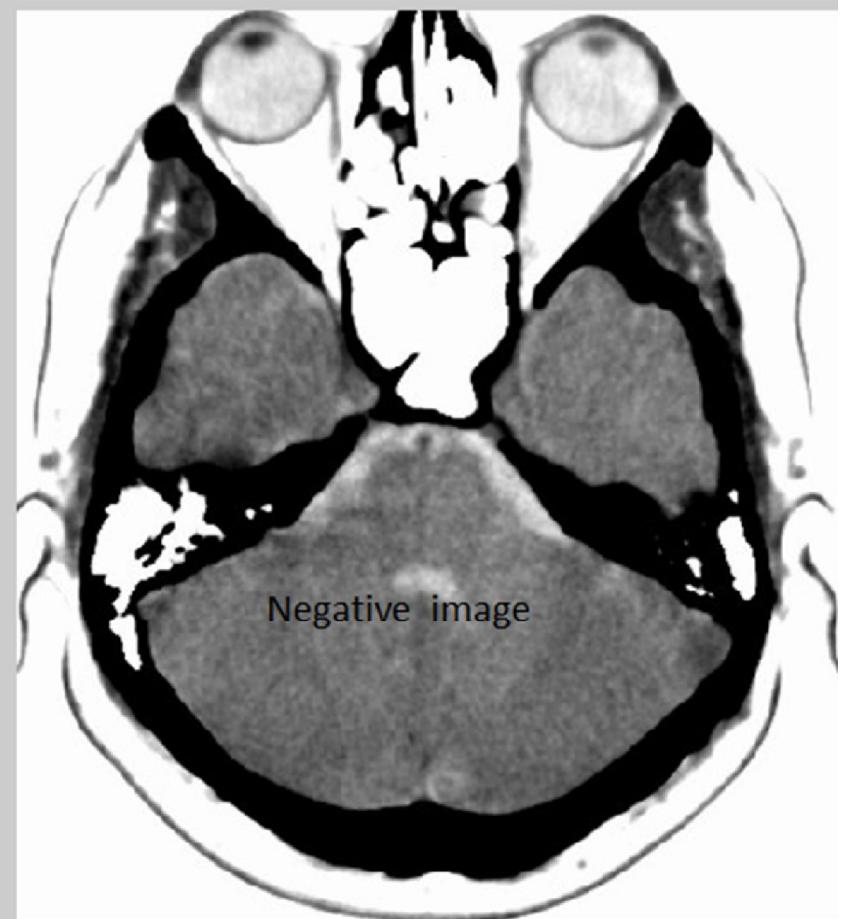
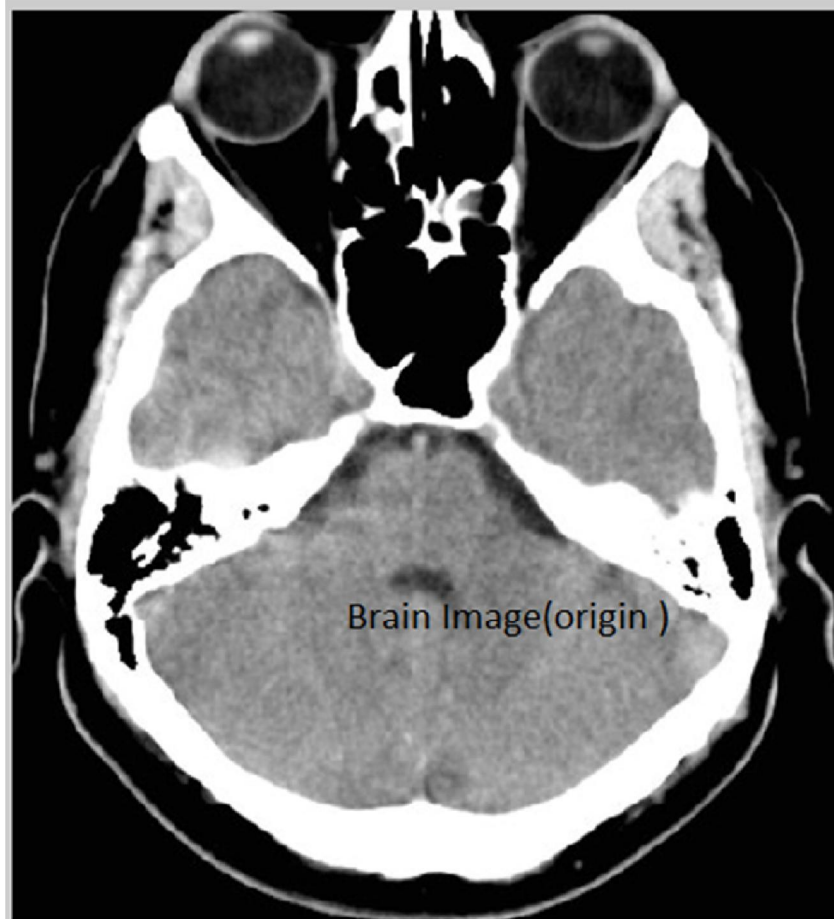
Some intensity transform functions

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.

- **Linear:** Negative, Identity
- **Logarithmic:** Log, Inverse Log
- **Power-Law:** n th power, n th root



Brain image and its Image Negatives



Power Law Transformation

- $s = cr^\gamma$
- C, γ : positive constants
- Gamma correction

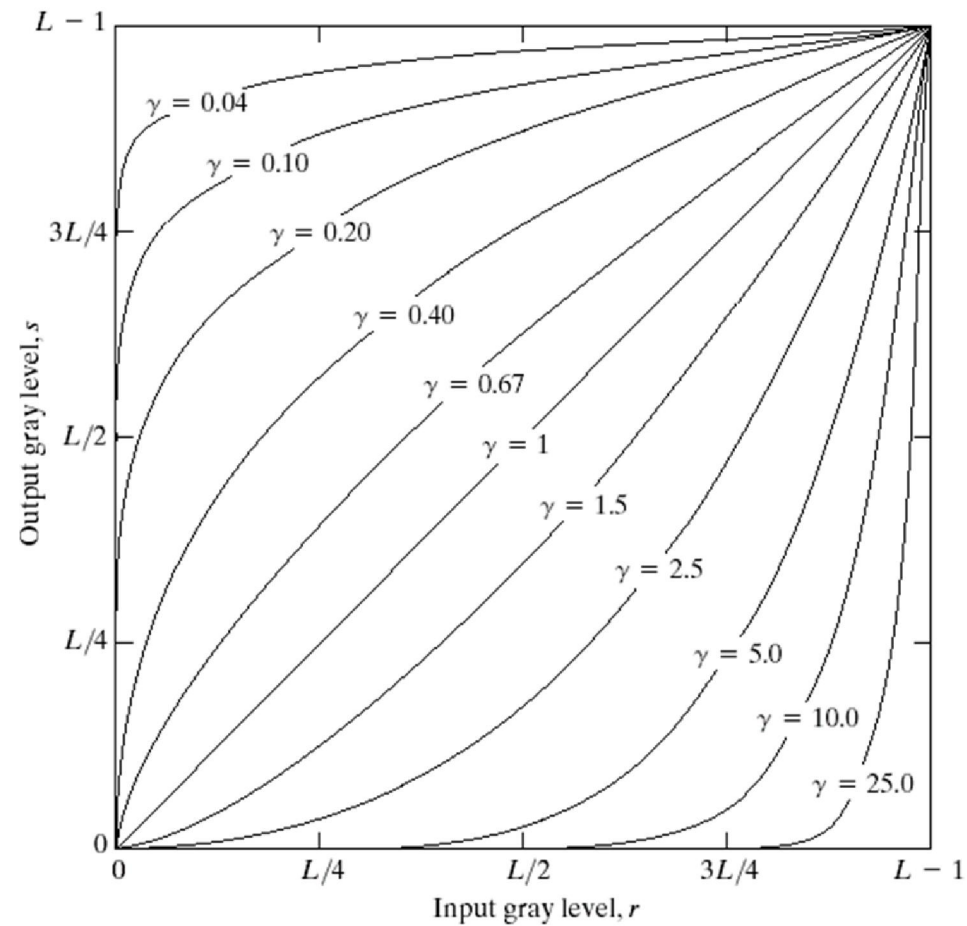


FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases).

- `A=imread('d:\flowers.jpg');`
- `A=rgb2gray(A);`
- `C=1;`
- `gamma1=0.6;`
- `B=C*double(A).^gamma1;`
- `figure`
- `subplot(1,2,1)`
- `imshow(A,[])`
- `subplot(1,2,2)`
- `imshow(B,[])`

before

using Gamma



In Matlab gamma transformation (using imadjust)

- `imadjust(f, [low_in high_in], [low_out high_out], gamma)`

With Gamma Transformations, you can curve the grayscale components either to brighten the intensity (when gamma is less than one) or darken the intensity (when gamma is greater than one).

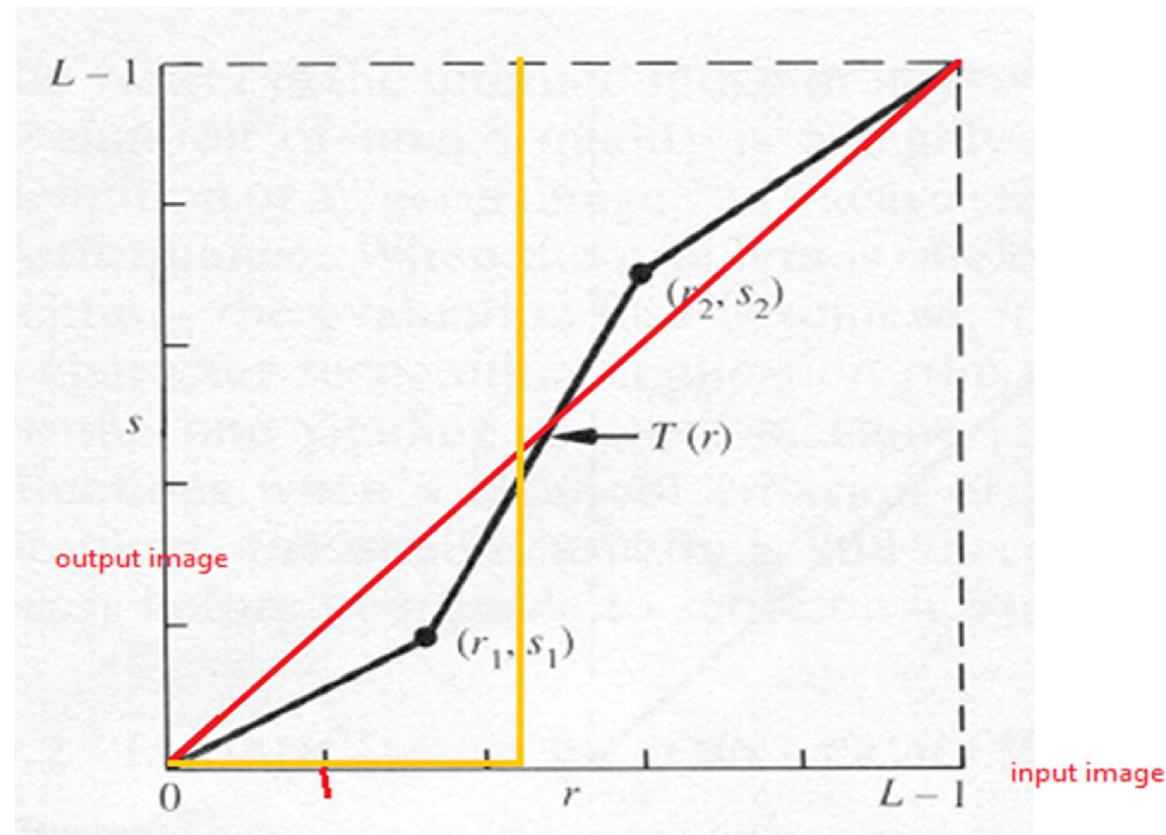
`imadjust(f, [low_in high_in], [low_out high_out], gamma)`

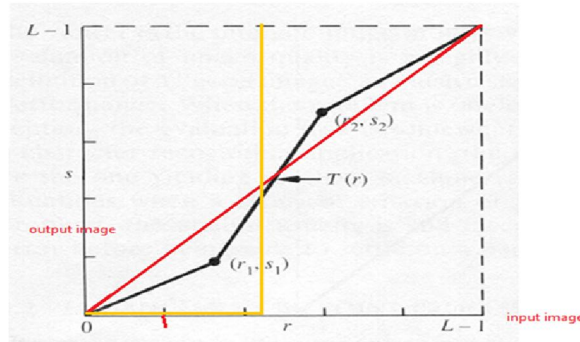
-

f is the input image, gamma controls the curve, and **[low_in high_in]** and **[low_out high_out]** are used for clipping. Values below low_in are clipped to low_out and values above high_in are clipped to high_out. For the purposes of this lab, we use [] for both [low_in high_in] and [low_out high_out]. This means that the full range of the input is mapped to the full range of the output

Contrast Stretching

- To increase the **dynamic range** of the gray levels in the image being processed.

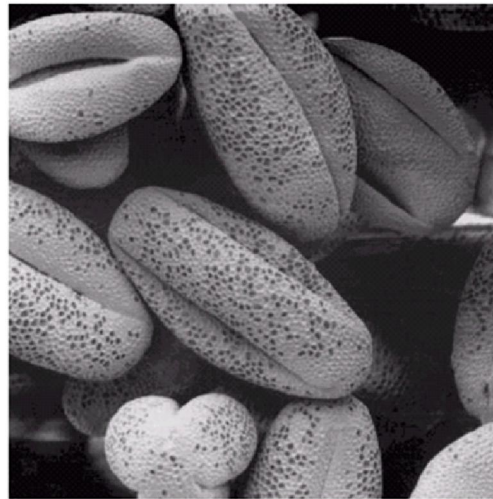
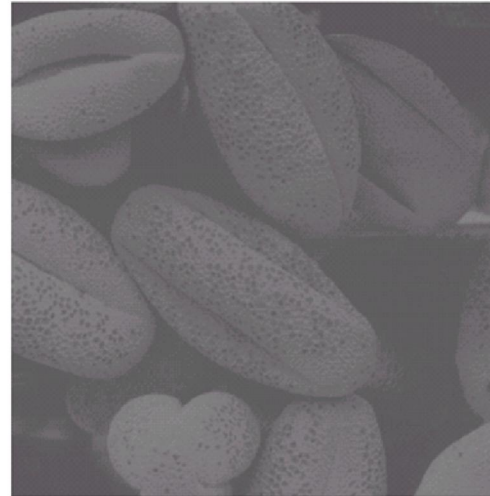
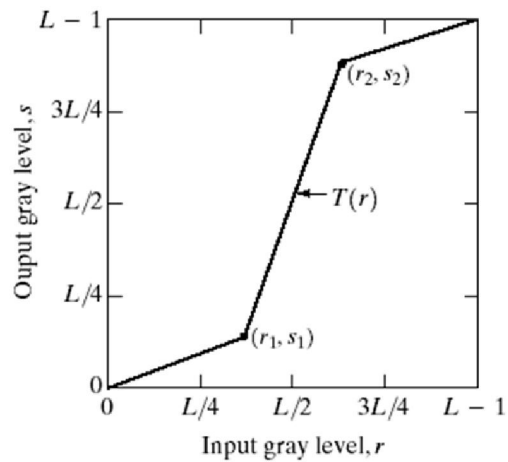




contd...

- The locations of (r_1, s_1) and (r_2, s_2) control the shape of the transformation function.
 - If $r_1 = s_1$ and $r_2 = s_2$ the transformation is a linear function and produces no changes.
 - If $r_1 = r_2$, $s_1 = 0$ and $s_2 = L-1$, the transformation becomes a thresholding function that creates a binary image.
 - Intermediate values of (r_1, s_1) and (r_2, s_2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.
 - Generally, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed.

Example



a b
c d

FIGURE 3.10

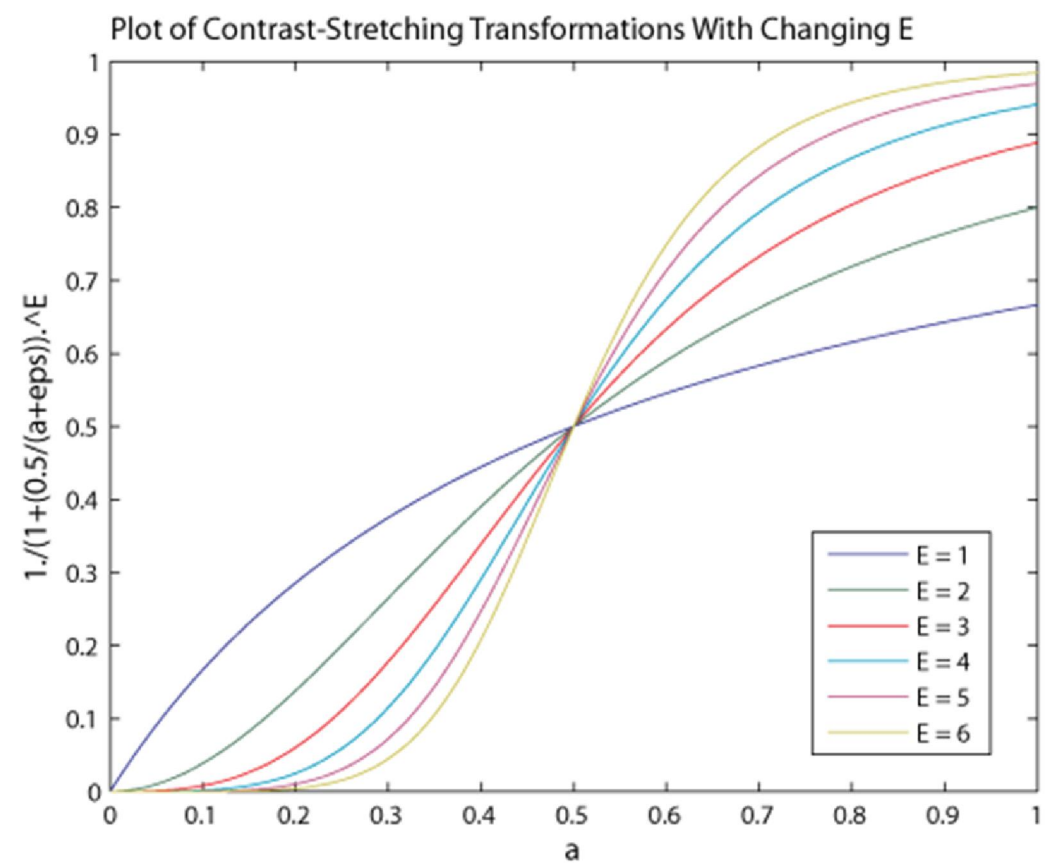
Contrast stretching.

(a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching.

(d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

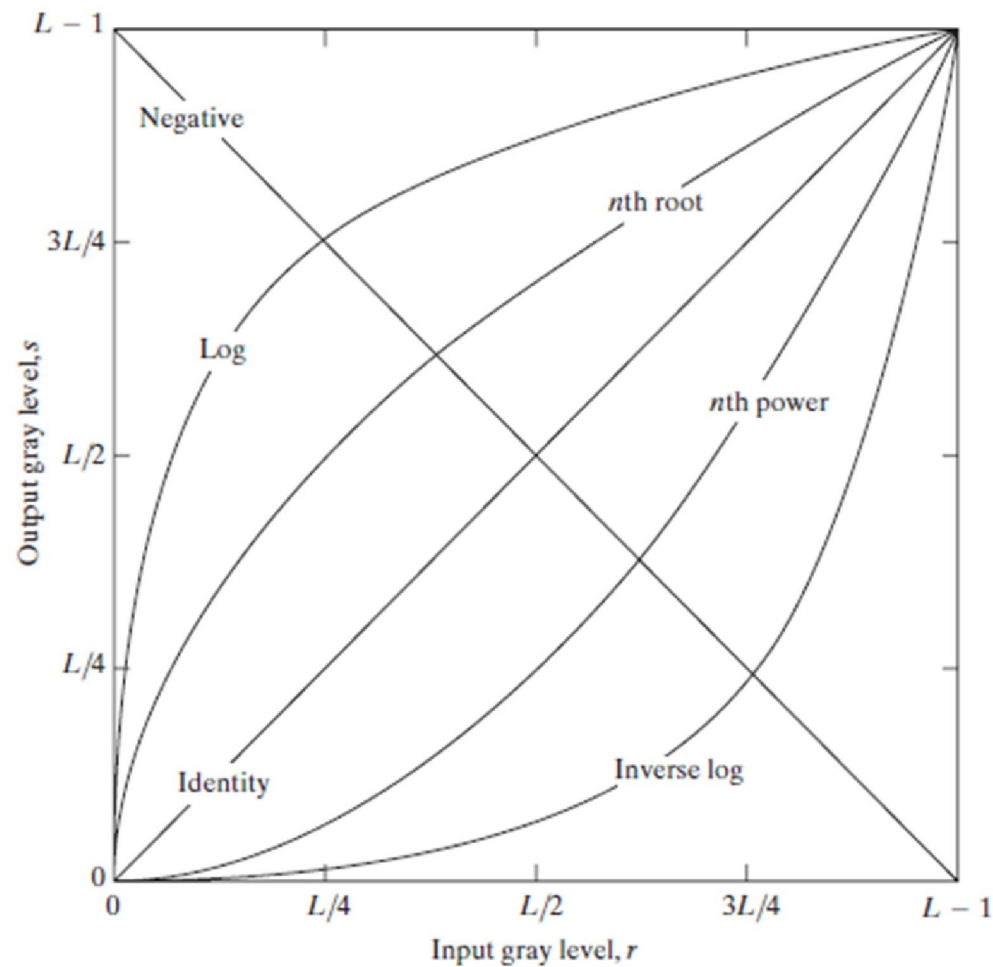
In MATLAB Contrast stretching transform

- $g = 1 ./ (1 + (m ./ (\text{double}(f) + \text{eps})) .^E)$
- `I=imread('tire.tif');`
- `I2=im2double(I);`
- `m=mean2(I2)`
- `contrast1=1./((1+(m./(I2+eps)).^4);`
- `contrast2=1./((1+(m./(I2+eps)).^5);`
- `contrast3=1./((1+(m./(I2+eps)).^10);`
- `imshow(I2)`
- `figure,imshow(contrast1)`
- `figure,imshow(contrast2)`
- `figure,imshow(contrast3)`



Again Some Intensity Transformation Functions

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



Power–Law (Gamma) transformation

$$s = cr^\gamma, \quad c, \gamma - \text{positive constants}$$

curve the grayscale components either to brighten the intensity (when $\gamma < 1$) or darken the intensity (when $\gamma > 1$).

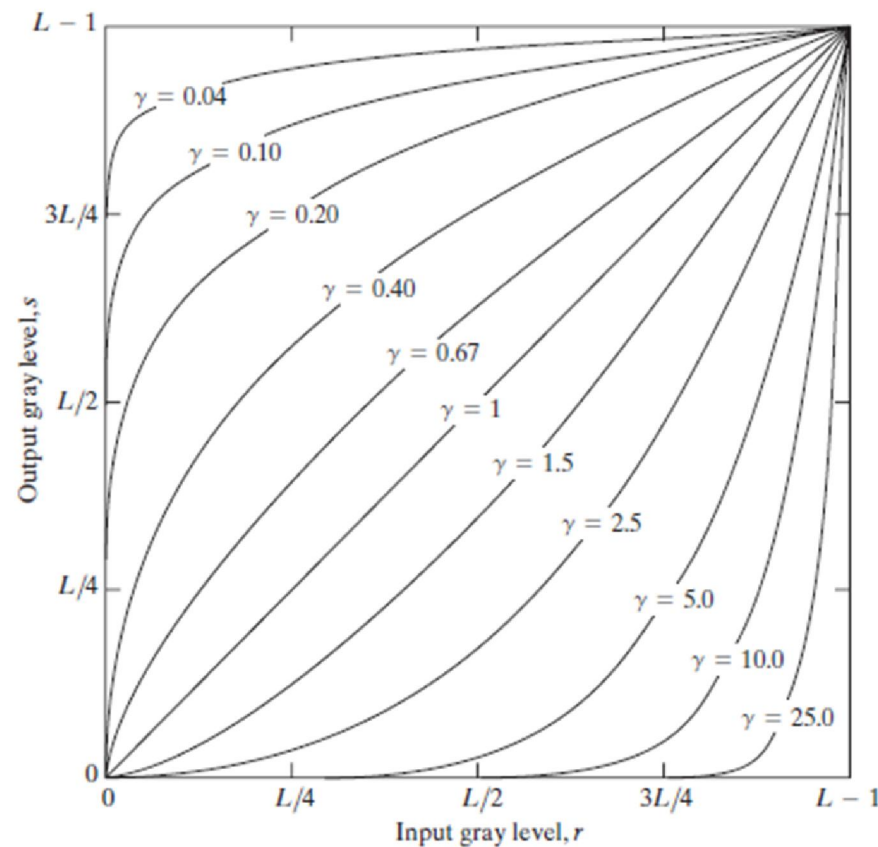
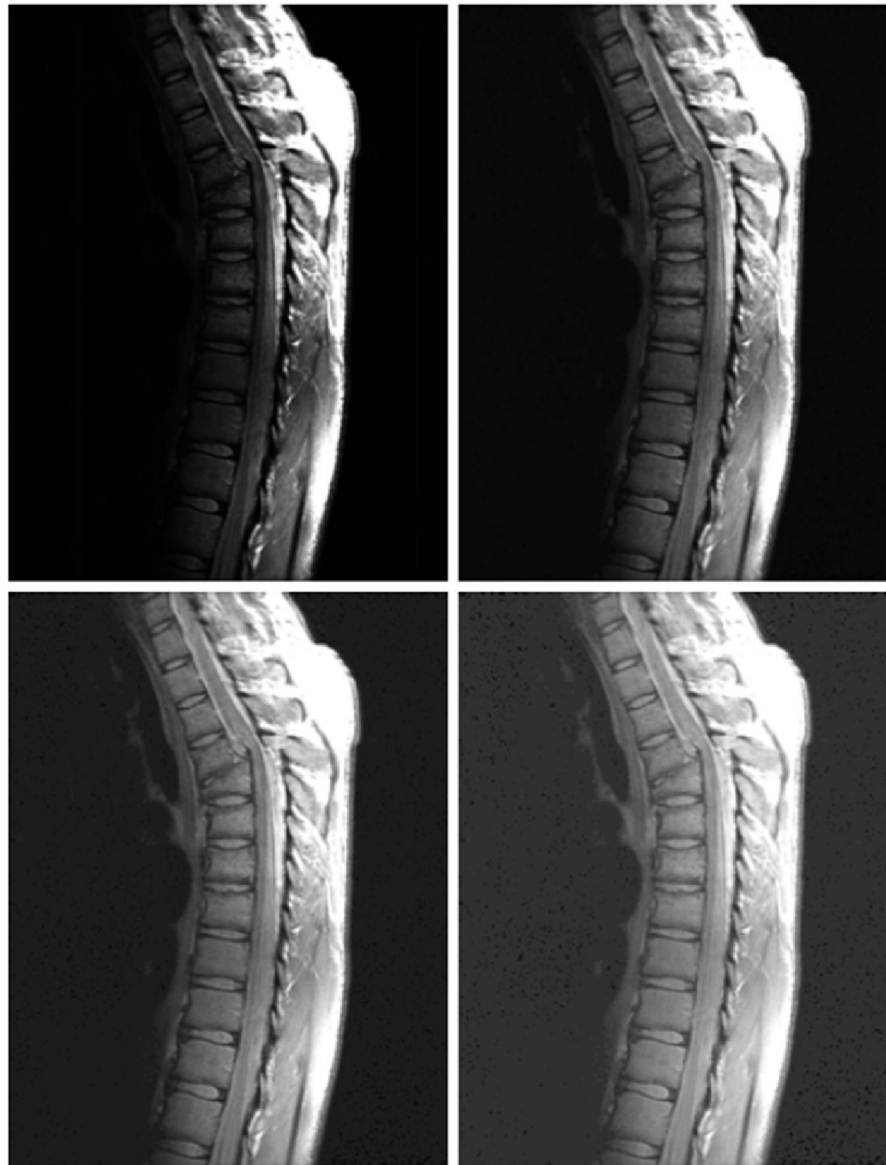


FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases).

Power –Law (Gamma) transformation



a b
c d

FIGURE 3.8

(a) Magnetic resonance (MR) image of a fractured human spine.

(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3 , respectively.

(Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Power –Law (Gamma) transformation

a b
c d

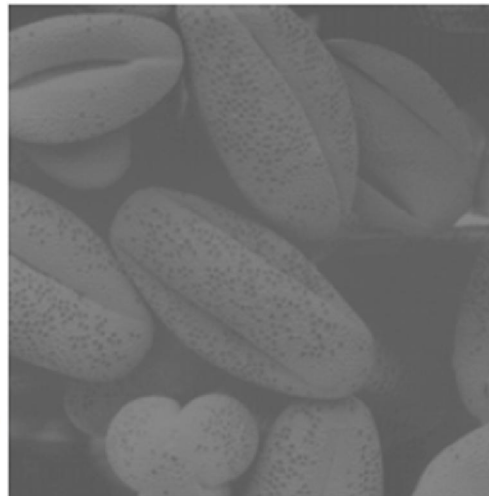
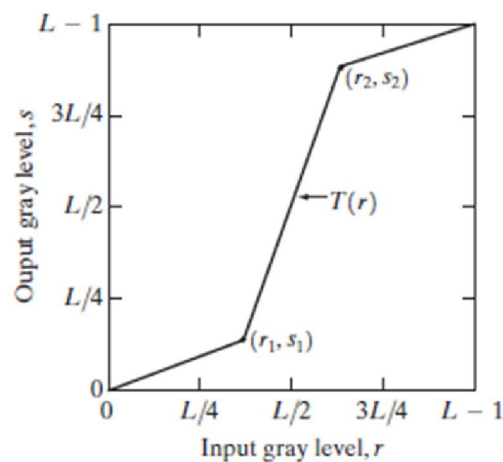
FIGURE 3.9
(a) Aerial image.
(b)–(d) Results of
applying the
transformation in
Eq. (3.2-3) with
 $c = 1$ and
 $\gamma = 3.0, 4.0,$ and
 5.0 , respectively.
(Original image
for this example
courtesy of
NASA.)



Contrast stretching

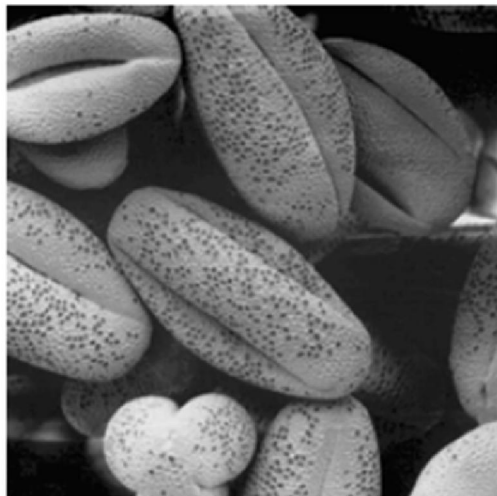
Contrast stretching is a process that expands the range of intensity levels in a image so that it spans the full intensity range of the recording medium or display device.

Contrast-stretching transformations increase the contrast between the darks and the lights

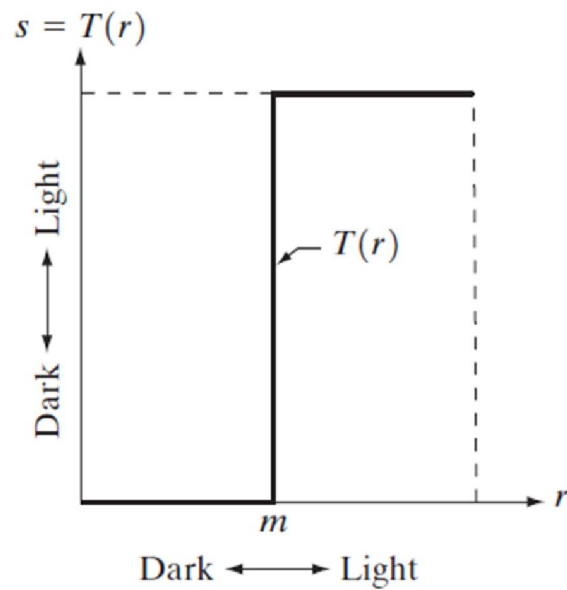


a b
c d

FIGURE 3.10
Contrast stretching.
(a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of



Thresholding function



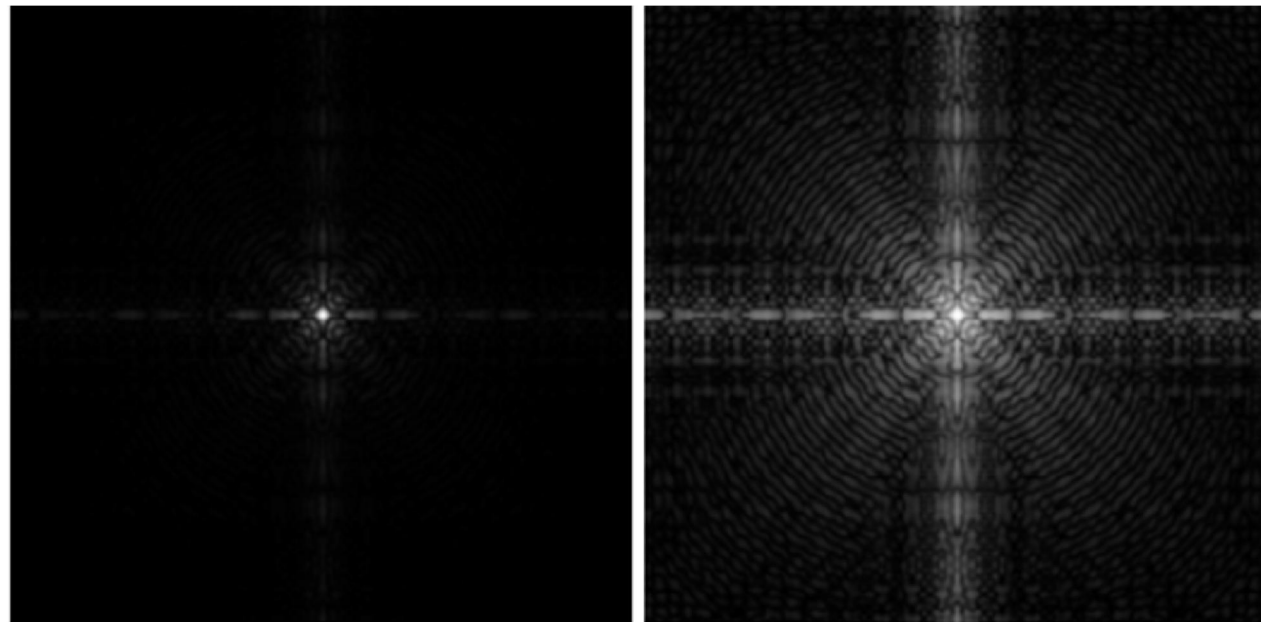
Log Transformations

$$s = c \log(1+r), \quad c = \text{const}, \quad r \geq 0$$

Maps a narrow range of low intensity values in the input into a wider range of output levels. The opposite is true for higher values of input levels.

a b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.



In MATLAB code Logarithmic transformation

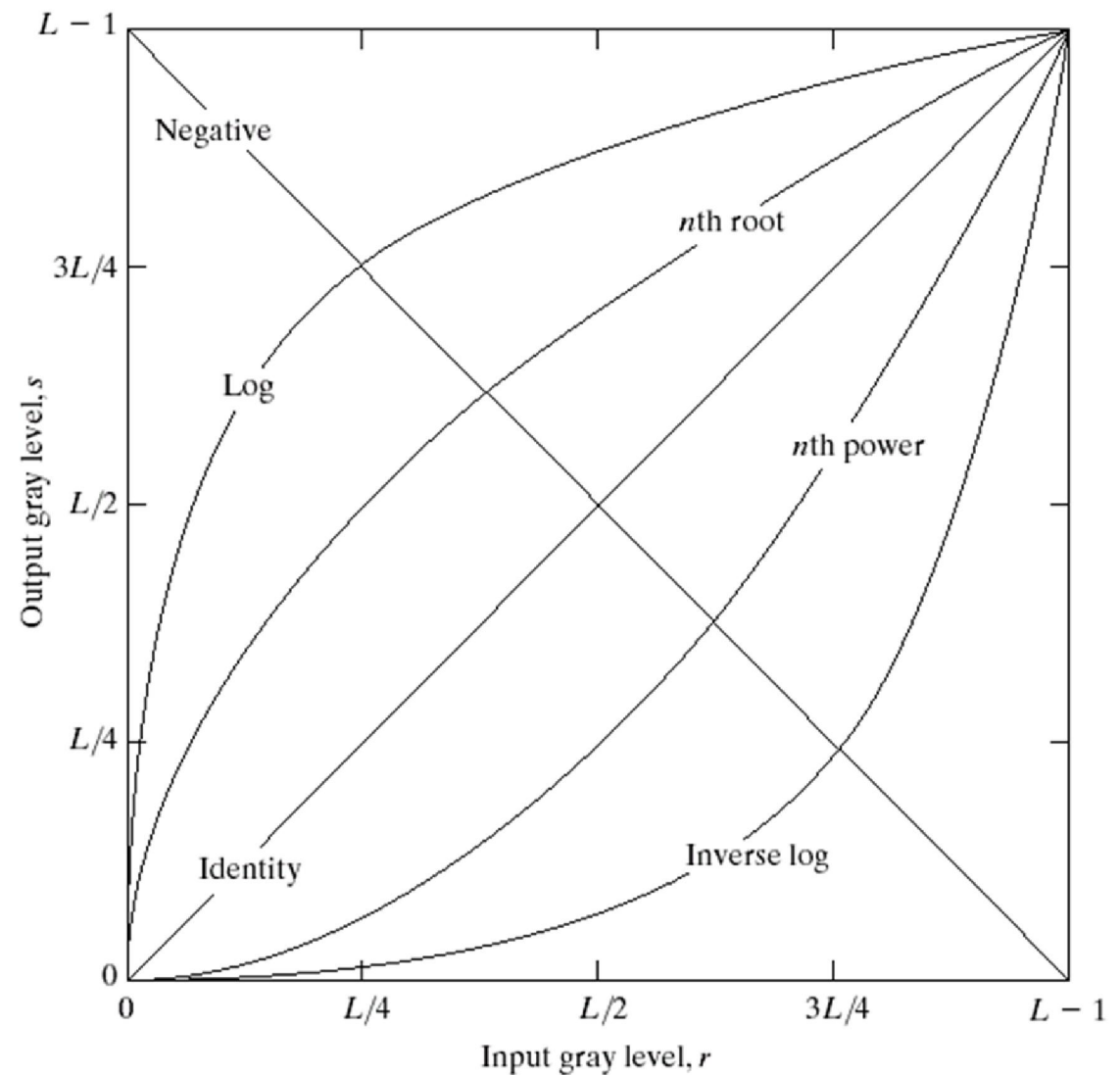
- `I=imread('tire.tif');`
- `imshow(I)`
- `I2=im2double(I);`
- `J=1*log(1+I2);`
- `J2=2*log(1+I2);`
- `J3=5*log(1+I2);`
- `figure, imshow(J)`
- `figure, imshow(J2)`
- `figure, imshow(J3)`

Some intensity transform functions

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.

- Linear: Negative, Identity
- Logarithmic: Log, Inverse Log
- Power-Law: n th power, n th root

From previous lecture



- `A=imread('d:\flowers.jpg');`
- `A=rgb2gray(A);`
- `C=1;`
- `gamma1=0.6;`
- `B=C*double(A).^gamma1;`
- `figure`
- `subplot(1,2,1)`
- `imshow(A,[])`
- `subplot(1,2,2)`
- `imshow(B,[])`

From previous lecture

before

using Gamma



Image processing course

Lecture 5

Color System , number system and
Filters

Dr. Nassir H. Salman



No. of colors and image file size

$$\text{colors number} = 2^{\text{color resolution}}$$

$$\text{image size} = \text{image resolution} \times \text{color resolution}$$

Color resolution = no. bits used to record each pixel , if it is 1 give us binary image, if it is 8 give gray image etc...

Image resolution= rows x columns= M x N =
no. of pixels in the image



النظام العشري	النظام الثنائي	السادس عشر
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7



تكملة النظام ١٦

8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15



تمثيل البكسل الملونة

11011001 00111011 00001111		
11011001	00111011	00001111
الأحمر	أخضر	أزرق



تمثيل البكسل الملونة بنظام الـ ٦ ١

11011001 00111011 00001111					
11011001		00111011		00001111	
1101	1001	0011	1011	0000	1111
D	9	3	B	0	F
D9		3B		0F	
#D93B0F					



Primary and Secondary Colors

- Colors are seen as variable combinations of the the so-called primary colors R, G and B.

Mixture of lights (additive primaries)

- The **primary** colors of light can be added to produce the **secondary** colors of light : magenta ($M=R+B$), cyan ($C=G+B$), and yellow ($Y=R+G$)
- Mixing the three primaries in the right intensities produces white light.

Mixture of Pigments (Subtractive primaries)

- **Primary** color of pigments is defined as one that subtracts or absorbs a primary color of light and reflects or transmits the other two.
- The **primary** colors of pigments are magenta, cyan, and yellow. And the **secondary** colors are R,G and B.
- The proper combinations of the three pigments primaries, produces black.



- Fig. 4.16: color combinations that result from combining primary colors available in the two situations, additive color and subtractive color.

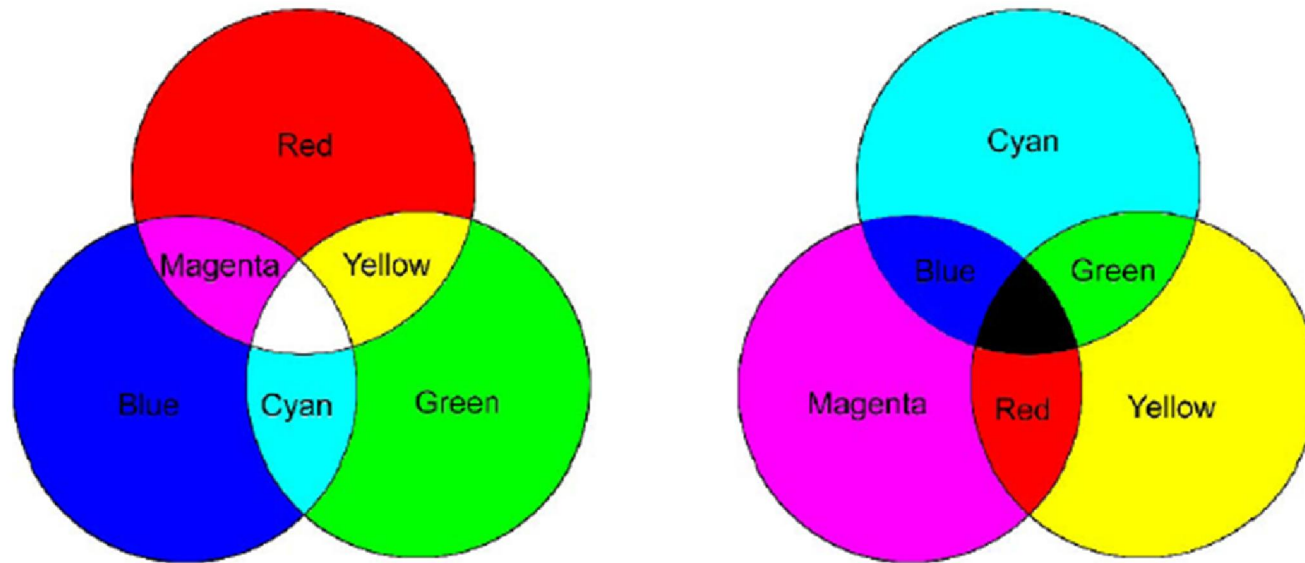


Fig. 4.16: Additive and subtractive color. (a): RGB is used to specify additive color. (b): CMY is used to specify subtractive color



الالوان الاساسية في نظام RGB بالاضافة الى الاسود والابيض

The color	Red	Green	Blue	
Red	255	0	0	
Green	0	255	0	
Blue	0	0	255	
White	255	255	255	
Black	0	0	0	
C	0	255	255	G+B
M	255	0	255	B+R
Y	255	255	0	R+G



CMY System

The color	Y	M	C	
Red	255	255	0	Absorb C and reflect Y,M
Green	255	0	255	
Blue	255	0	0	
White	0	0	0	
Black	255	255	255	المعادلات ادناه من مقارنة الجدولين
C	0	0	255	$C=255-R$
M	0	255	0	$M=255-G$
Y	255	0	0	$Y=255-B$



احتساب قيم CMYK من CMY حيث L تمثل اقل قيمة بين القيم CMY

$$C = \frac{C - L}{255 - L}$$

$$M = \frac{M - L}{255 - L}$$

$$Y = \frac{Y - L}{255 - L}$$

$$K = \frac{L}{255}$$



حول القيم (96,134,200) من النظام RGB الى النظام CMYK
اولا: نجري التحويل من النظام RGB الى النظام CMY ثم من CMY الى
النظام CMYK

$$C = \frac{C - L}{255 - L} = \frac{159 - 55}{255 - 55} = 0.52$$

$$M = \frac{M - L}{255 - L} = \frac{121 - 55}{255 - 55} = 0.33$$

$$Y = \frac{Y - L}{255 - L} = \frac{55 - 55}{255 - 55} = 0$$

$$K = \frac{L}{255} = \frac{55}{255} \approx 0.216$$

$$\therefore CMYK = (52\%, 33\%, 0\%, 21.6\%)$$

$$C = 255 - R = 255 - 96 = 159$$

$$M = 255 - G = 255 - 134 = 121$$

$$Y = 255 - B = 255 - 200 = 55$$

$$\therefore CMY = (159, 121, 55)$$



مثال: حول اللون #7AB50F الى الفضاء اللوني CMY
نقوم بتحويل التمثيل ١٦ لنظام RGB الى النظام العشري
ثم نقوم بالتحويل من RGB الى CMY

$$(7A)_{16} = 7 \times 16 + 10 = 122 \Rightarrow \text{red}$$

$$(B5)_{16} = 11 \times 16 + 5 = 181 \Rightarrow \text{green}$$

$$(0F)_{16} = 0 \times 16 + 15 = 15 \Rightarrow \text{blue}$$

$$C = 255 - R = 255 - 122 = 133$$

$$M = 255 - G = 255 - 181 = 74$$

$$Y = 255 - B = 255 - 15 = 240$$

$$\therefore CMY = (133, 74, 240)$$



filters

1	1	1
1	1	1
1	1	1

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

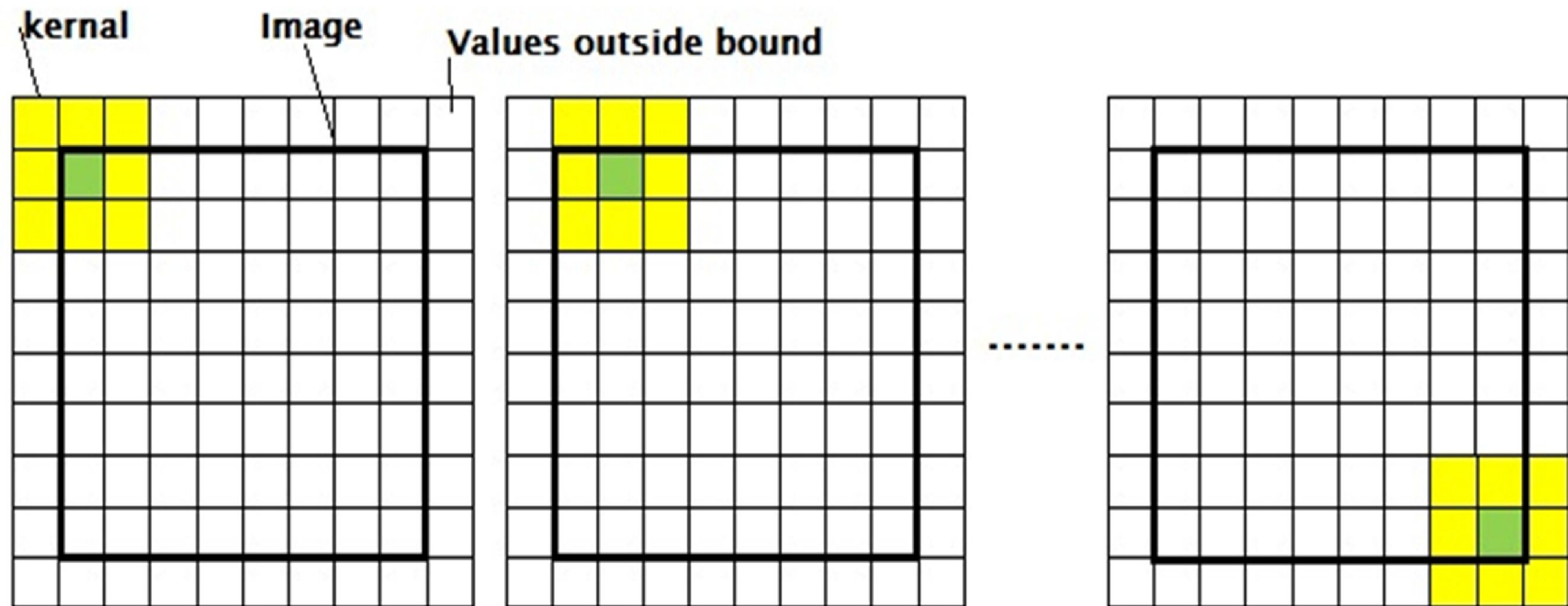
3 x 3

5 x 5

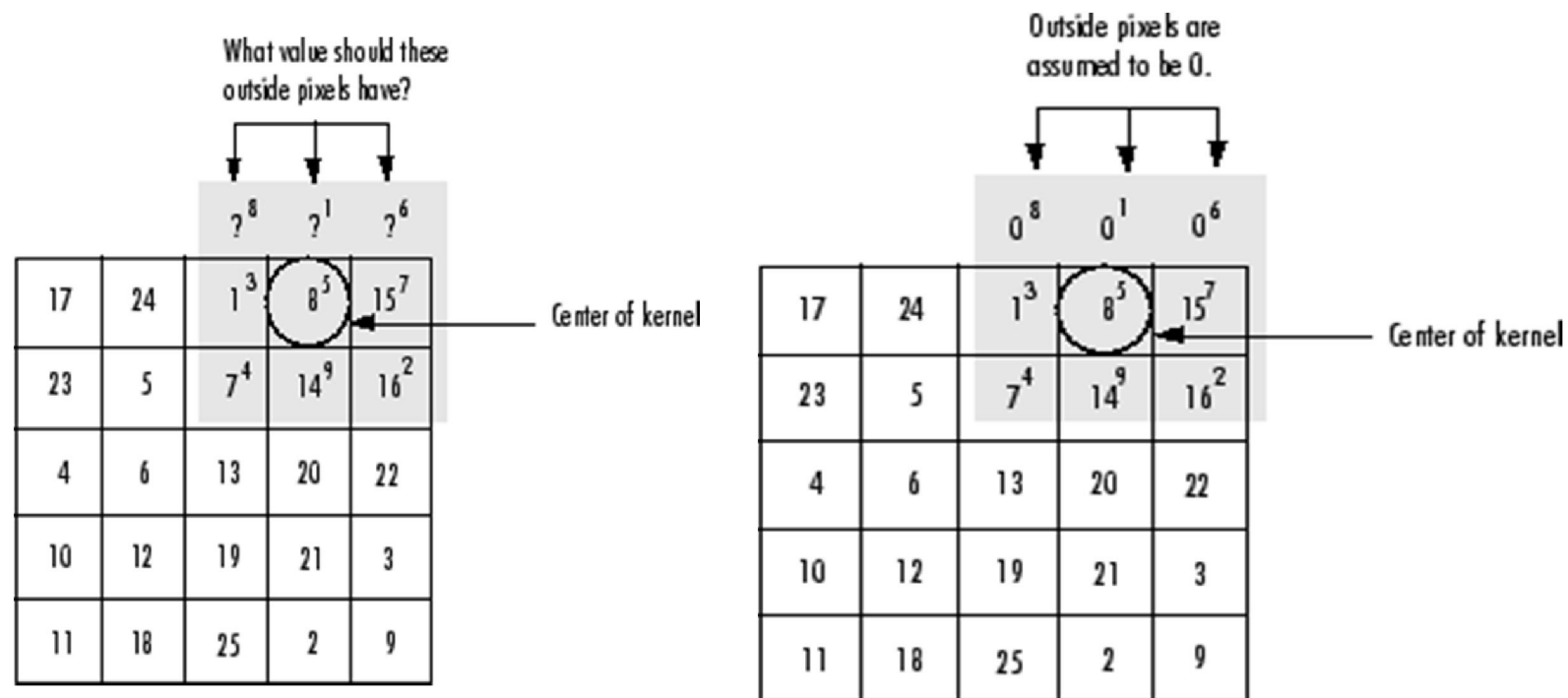
Masks , windows

7 x 7

Using filters



Filter and image



filters

From previous lecture 5

1	1	1
1	1	1
1	1	1

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

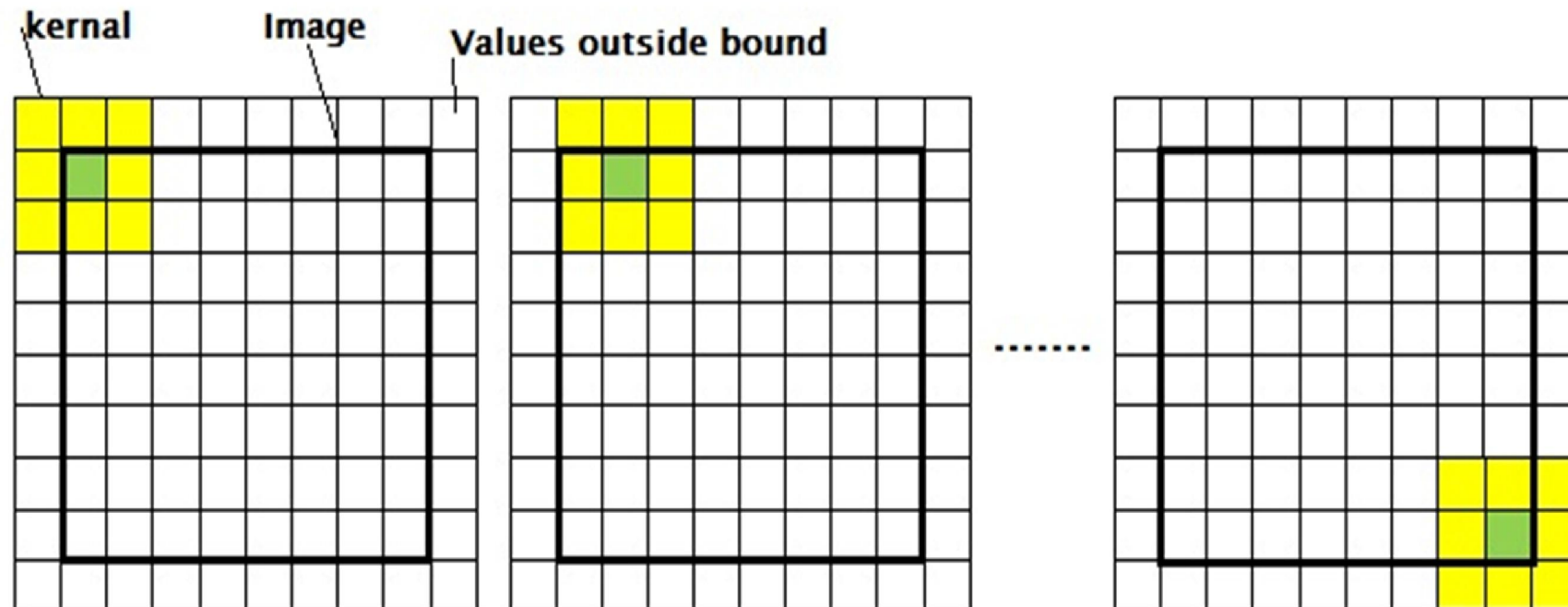
3 x 3

5 x 5

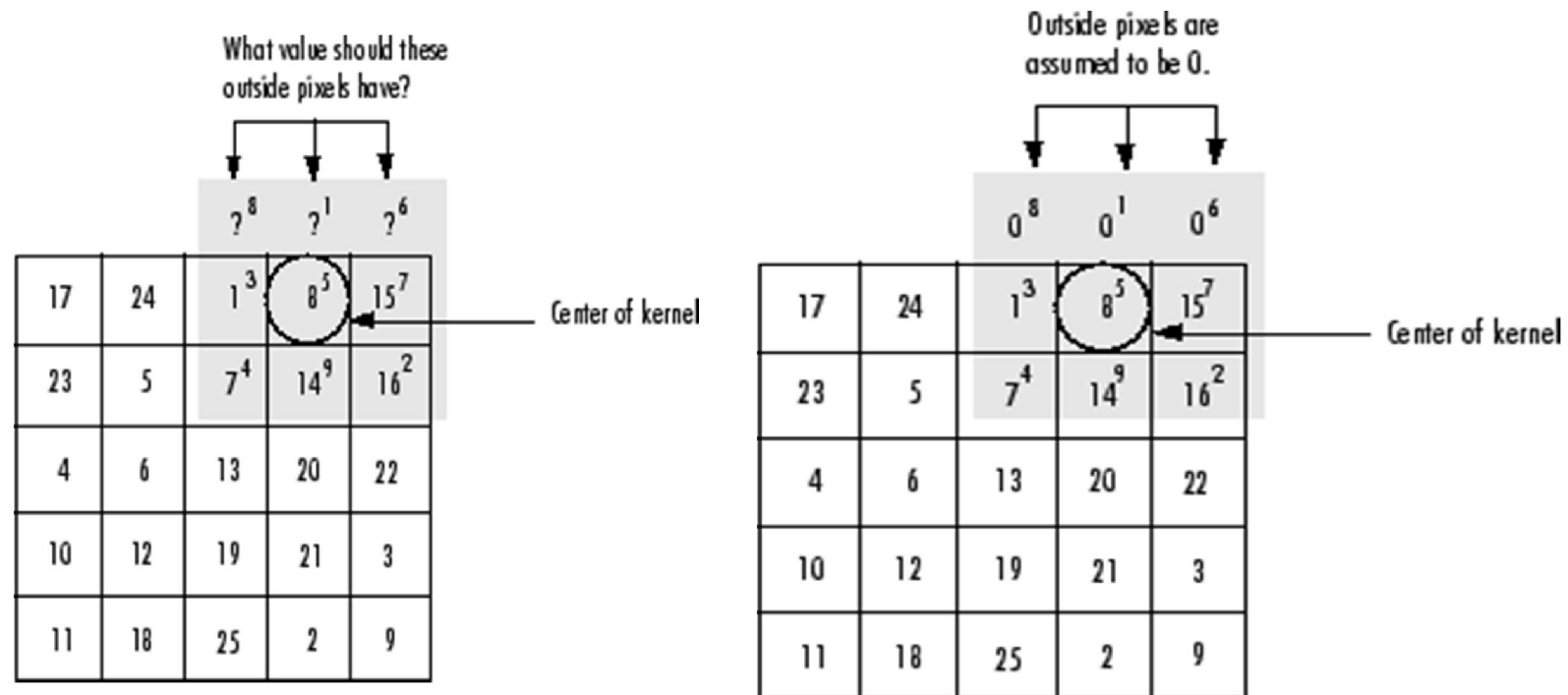
Masks , windows

7 x 7

Using filters



Filter and image



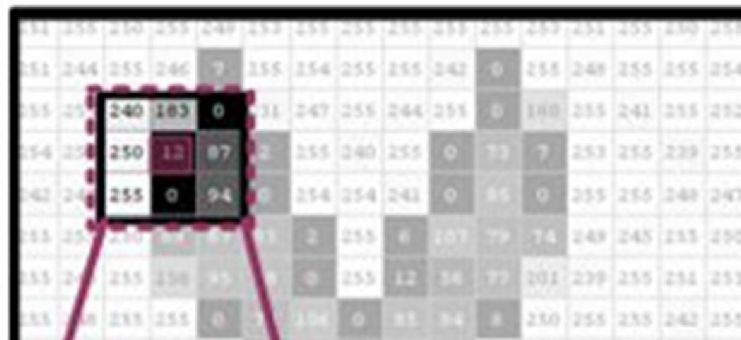
Lecture 6

Spatial Filtering process

For spatial domain filtering, we are performing filtering operations directly on the pixels of an image.

Spatial filtering is a technique that uses a pixel and its neighbors to select a new value for the pixel

Original Image



- There are two main types of filtering applied to images:
- **spatial domain filtering**
- **frequency domain filtering**

In a later lab we will talk about frequency domain filtering, which makes use of the Fourier Transform.

For spatial domain filtering, we are performing filtering operations directly on the pixels of an image.

- Spatial filtering is a technique that uses a pixel and its neighbors to select a new value for the pixel. The simplest type of spatial filtering is called linear filtering
- It attaches a weight to the pixels in the neighborhood of the pixel of interest, and these weights are used to blend those pixels together to provide a new value for the pixel of interest

- Linear filtering can be used to **smooth, blur, sharpen, or find the edges of an image**. The following four images are meant to demonstrate what spatial filtering can do. The **original image is shown in the upper left-hand corner**.



- Smooth blur sharpen find the edges

- Such non-linear filters are useful for smoothing only smooth areas, enhancing only strong edges or removing speckles from images.
- In **signal processing**, it is often desirable to be able to perform some kind of noise reduction on an image or signal. The median filter is a nonlinear digital filtering technique, often used to remove **noise**. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). **Median filtering** is very widely used in digital image processing because, under certain conditions, it preserves **edges while removing noise**

What are the mean and median filters?

- **The mean filter is a simple sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window. The window, or kernel, is usually square but can be any shape. An example of mean filtering of a single 3x3 window of values is shown below.**

Mean filter

unfiltered values		
5	3	6
2	1	9
8	4	7

$$5 + 3 + 6 + 2 + 1 + 9 + 8 + 4 + 7 = 45$$

$$45 / 9 = 5$$

mean filtered

*	*	*
*	5	*
*	*	*

Center value (previously 1) is replaced by the mean of all nine values (5).

median filter

- The median filter is also a sliding-window spatial filter, but it replaces the center value in the window with the median of all the pixel values in the window. As for the mean filter, the kernel is usually square but can be any shape. An example of median filtering of a single 3x3 window of values is shown below.

unfiltered values

6	2	0
3	97	4
19	3	10

in order:

0, 2, 3, 3, 4, 6, 10, 19, 97

median filtered

*	*	*
*	4	*
*	*	*

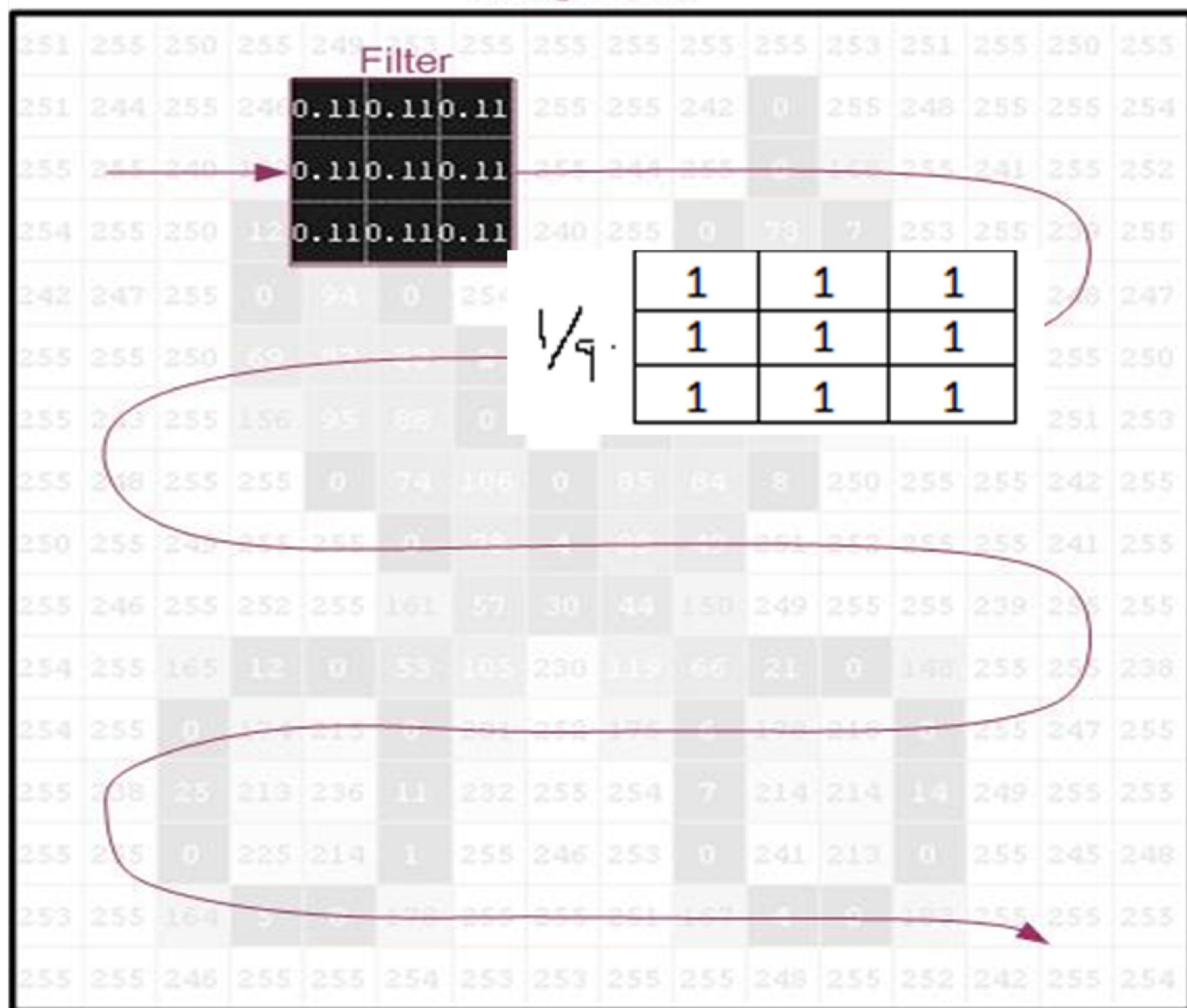
Center value (previously 97) is replaced by the median of all nine values (4).

- **This illustrates one of the celebrated features of the median filter: its ability to remove 'impulse' noise (outlying values, either high or low).**
- **The median filter is also widely claimed to be 'edge-preserving' since it theoretically preserves step edges without blurring.**
- **However, in the presence of noise it does blur edges in images slightly.**

Basic idea of Spatial Filtering

- **Spatial Filtering** is sometimes also known as **neighborhood processing**. Neighborhood processing is an appropriate name because you define a center point and perform an operation (or apply a filter) to only those pixels in predetermined neighborhood of that center point.
- **The result of the operation is one value, which becomes the value at the center point's location in the modified image.** Each point in the image is processed with its neighbors. The general idea is shown below as a "sliding filter" that moves throughout the image to calculate the value at the center location

Image Data

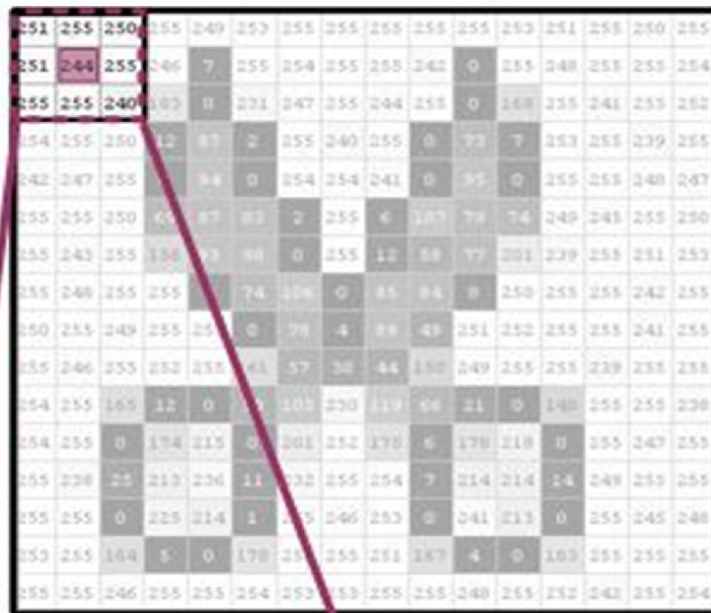


The following diagram is meant to illustrate in further details how the filter is applied. The filter (an averaging filter) is

applied to location 2,2.

Filter Applied at (2, 2)

Original Image



Filtered Image



Averaging Filter

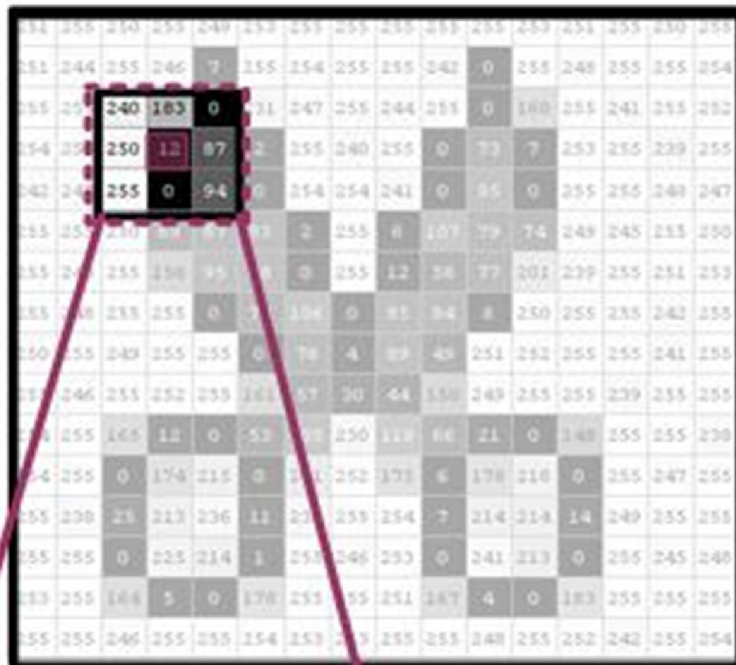
251	255	250	.	*	0.1111	0.1111	0.1111	=	27.8888	28.3333	27.7777	⇒	250.66
251	244	255			0.1111	0.1111	0.1111		27.8888	27.1111	28.3333		
255	255	240			0.1111	0.1111	0.1111		28.3333	28.3333	26.6666		

- Notice how the resulting value is placed at location 2,2 in the filtered image.
- The breakdown of how the resulting value of 251 (rounded up from 250.66) was calculated mathematically is:
 - $= 251 * 0.1111 + 255 * 0.1111 + 250 * 0.1111 + 251 * 0.1111 + 244 * 0.1111 + 255 * 0.1111 + 255 * 0.1111 + 255 * 0.1111 + 240 * 0.1111$
 - $= 27.88888 + 28.33333 + 27.77777 + 27.88888 + 27.11111 + 28.33333 + 28.33333 + 28.33333 + 26.66666$
 - $= 250.66$

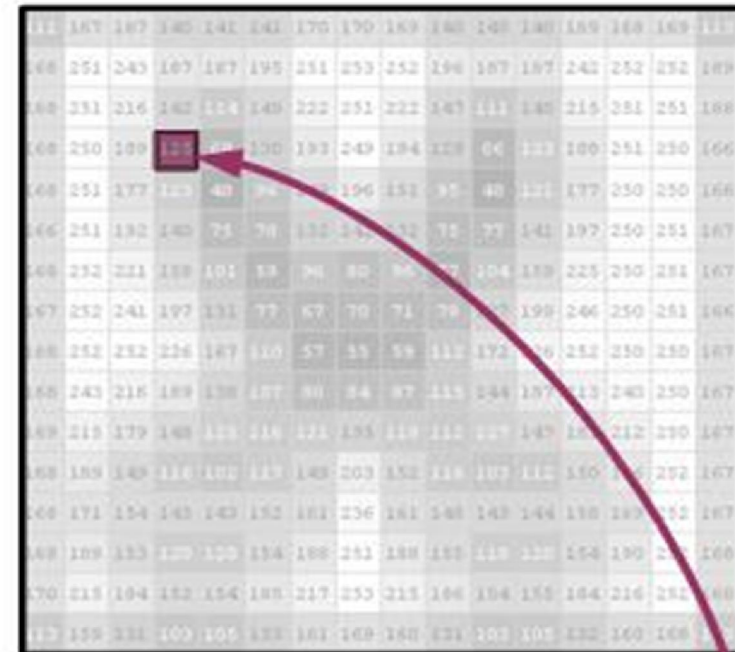
The following illustrates the averaging filter applied to location 4,4.

Filter Applied at (4, 4)

Original Image



Filtered Image



240	183	0
250	12	87
255	0	94

 \cdot

0.1111	0.1111	0.1111
0.1111	0.1111	0.1111
0.1111	0.1111	0.1111

 $=$

26.6666	20.3333	0
27.7777	1.3333	9.6666
28.3333	0	10.4444

 \Rightarrow

124.55

- Once again, the mathematical breakdown of how 125 (rounded up from 124.55) was calculated is below:
- $= 240 * 0.11111 + 183 * 0.11111 + 0 * 0.11111 + 250 * 0.11111 + 12 * 0.11111 + 87 * 0.11111 + 255 * 0.11111 + 0 * 0.11111 + 94 * 0.11111$
- $= 26.6666 + 20.3333 + 0 + 27.7777 + 1.3333 + 9.6666 + 28.3333 + 0 + 10.4444$
- $= 124.55$

Boundary Options

- See section 3.5. in your textbook.
- The example above deliberately applied the filter at location 2,2. This is because there is an inherent problem when you are working with the corners and edges. The problem is that some of the "neighbors" are missing. Consider location 1,1:

Filter Applied at 1,1

		251	255	250	255	249	253	255	255	255	255	253	251	255	250	255
		251	244	255	246	7	255	254	255	255	242	0	255	248	255	254
				255	240	183	0	231	247	255	244	255	0	168	255	252
				254	255	250	12	87	2	255	240	255	0	73	7	253
				242	247	255	0	94	0	254	254	241	0	95	0	255
				255	255	250	69	87	83	2	255	6	107	79	74	249
				255	243	255	156	95	88	0	255	12	58	77	201	239
				255	248	255	255	0	74	106	0	85	84	8	250	255
				250	255	249	255	255	0	78	4	89	49	251	252	255
				255	246	255	252	255	161	57	30	44	150	249	255	255

Original Image

251	255	250	255	249	253	255	255	255	255	253	251	255	250	255		
251	244	255	246	7	255	254	255	255	242	0	255	248	255	255	254	
255	255	240	183	0	231	247	255	244	255	0	168	255	241	255	252	
254	255	250	12	87	2	255	240	255	0	73	7	253	255	239	255	
242	247	255	0	94	0	254	254	241	0	95	0	255	255	248	247	
255	255	250	69	87	83	2	255	6	107	79	74	249	245	255	250	
255	243	255	156	95	88	0	255	12	58	77	201	239	255	251	253	
255	248	255	255	0	74	106	0	85	84	8	250	255	255	242	255	
250	255	249	255	255	0	78	4	89	49	251	252	255	255	241	255	
255	246	255	252	255	161	57	30	44	150	249	255	255	239	255	255	

- In this case, there are no upper neighbors or neighbors to the left. Two solutions, zero padding and replicating, are shown below. The pixels highlighted in blue have been added to the original image:

Zero Padding

[illegible]

Zero padding is the default. You can also specify a value other than zero to use as a padding value.

Another solution is replicating the pixel values along the edges:

Replicating

251	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	255
251	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	255
251	251	244	255	246	7	255	254	255	255	242	0	255	248	255	255	254	254
255	255	255	240	183	0	231	247	255	244	255	0	168	255	241	255	252	252
254	254	255	250	12	87	2	255	240	255	0	73	7	253	255	239	255	255
242	242	247	255	0	94	0	254	254	241	0	95	0	255	255	248	247	247
255	255	255	250	69	87	83	2	255	6	107	79	74	249	245	255	250	250
255	255	243	255	156	95	88	0	255	12	58	77	201	239	255	251	253	253
255	255	248	255	255	0	74	106	0	85	84	8	250	255	255	242	255	255
250	250	255	249	255	255	0	78	4	89	49	251	252	255	255	241	255	255
255	255	246	255	252	255	161	57	30	44	150	249	255	255	239	255	255	255
254	254	255	165	12	0	53	105	230	119	66	21	0	148	255	255	238	238
254	254	255	0	174	215	0	201	252	175	6	178	218	0	255	247	255	255
255	255	238	25	213	236	11	232	255	254	7	214	214	14	249	255	255	255
255	255	255	0	225	214	1	255	246	253	0	241	213	0	255	245	248	248
253	253	255	164	5	0	178	255	255	251	167	4	0	183	255	255	255	255
255	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	254
255	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	254

- As a note, if your filter were larger than 3x3, then the "border padding" would have to be extended. For a filter of size 3x3, 'replicate' and 'symmetric' yield the same results.
- The following images show the results of the four different boundary options. The filter used below is a 5x5 averaging filter that was created with the following syntax:
`h=fspecial('average',5)`

[illegible]

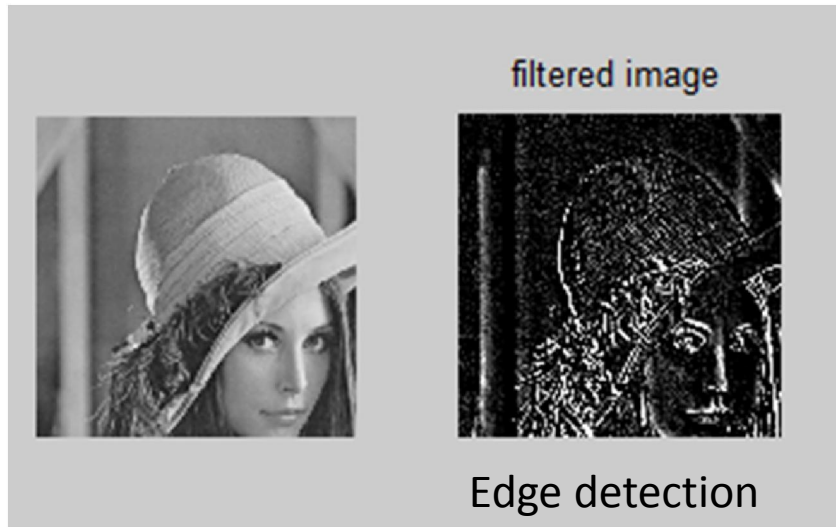
The following MATLAB function demonstrates how spatial filtering may be applied to an image

- `function img = myfilter(f, w)`
- `%MYFILTER Performs spatial correlation`
- `% I=MYFILTER(f, w) produces an image that has undergone correlation.`
- `% f is the original image`
- `% w is the filter (assumed to be 3x3)`
- `% The original image is padded with 0's`
- `%Author: Nova Scheidt`
-
- `% check that w is 3x3`
- `[m,n]=size(w);`
- `if m~=3 | n~=3`
- `error('Filter must be 3x3')`
- `end`

example

```
clear
%W= 1/9*[1 1 1; 1 1 1;1 1 1];
%W= [0 1 0; 1 -4 1;0 1 0];
%W= [1 1 1; 1 -8 1;1 1 1];
%W= [-1 2 -1; 2 -4 2;-1 2 -1];
%W= [1 0; 0 -1];
%W= 1/16*[1 2 1; 2 4 2;1 2 1];
%W=[-0.5 -0.5 -0.5 ;-0.5 5.5 -0.5;-0.5 -0.5 -0.5];
w=[-3 5 5 ;-3 0 5;-3 -3 -3];%kirsch detection
%W= [ 1 4 1; 4 -20 4; 1 4 1];%Laplacian aproximation more dense spatial kernel
%W= [ 1 -2 1; -2 4 -2; 1 -2 1];
%W=[0 0 0;0 1 0;0 0 0]
%W=[0 0 0;0 1 0;0 0 -1]
%W=[0 0 0 0 0;0 1 1 1 0;0 1 1 1 0;0 1 1 1 0; 0 0 0 0 0]
f=imread('D:\lena1.jpg');
f=rgb2gray(f);
[x,y]=size(f);
```


examples



Fourier Transforms

تحويلات فوريير الرياضية للصور الرقمية

The Fourier transform is a representation of an image as a sum of complex exponentials of varying magnitudes, frequencies, and phases.

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

- Fourier components: sinusoidal patterns

$F(u, v)$ Fourier Coefficients

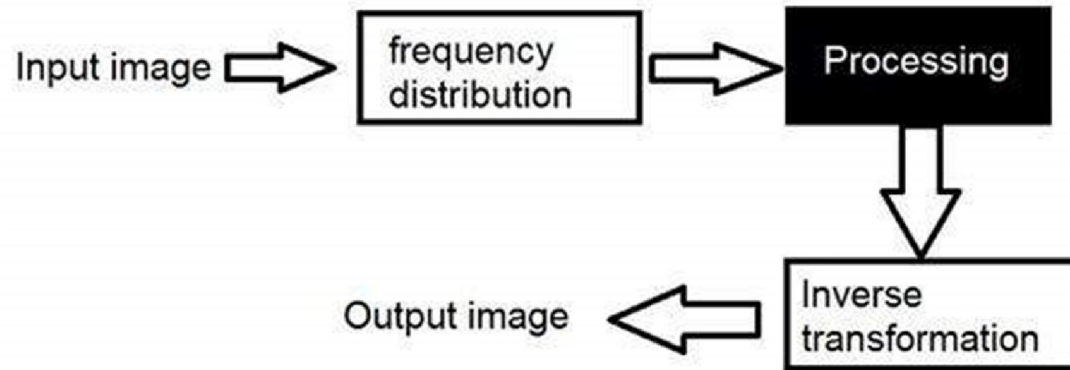
$f(x, y)$ is an image

- The **Fourier Transform** is an important **image** processing tool which is used to decompose an **image** into its sine and cosine components. The output of the **transformation** represents the **image** in the **Fourier** or frequency domain, while the input **image** is the spatial domain equivalent.
- The **DFT is used** to convert an **image** from the spatial domain into frequency domain, in other words it allows us to separate high frequency from low frequency coefficients and neglect or alter specific frequencies leading to an **image** with less information but still with a convenient level of quality .
- Fourier transform is a mathematical formula by which we can extract out the frequency domain components of a continuous time domain signal. Using fourier transform we can process time domain signal in frequency domain. We can use various Frequency domain filters to process the signal.
- If signal is discrete, Discrete Fourier Transform use to analyse discrete signal
- The **Fast Fourier Transform (FFT)** is commonly used to transform an **image** between the spatial and frequency domain. Unlike other domains such as Hough and Radon, the **FFT** method preserves all original data. Plus, **FFT** fully transforms **images** into the frequency domain, unlike time-frequency or wavelet transforms.

Difference between **spatial domain** and **frequency domain**

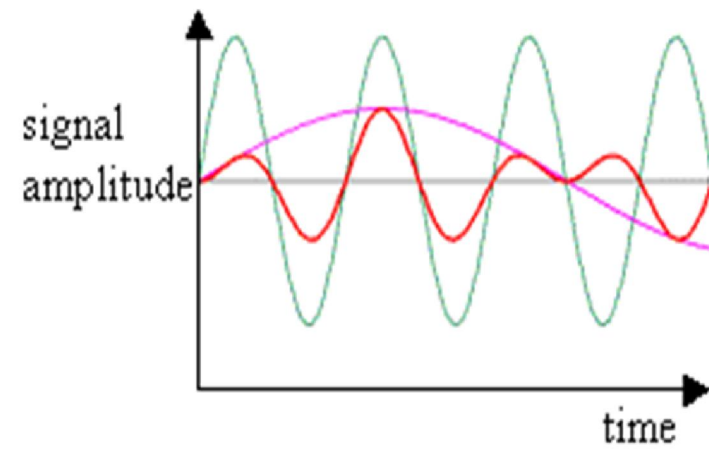


In spatial domain, we deal with images as it is. The value of the pixels of the image change with respect to scene. Whereas in frequency domain, we deal with the rate at which the pixel values are changing in spatial domain.

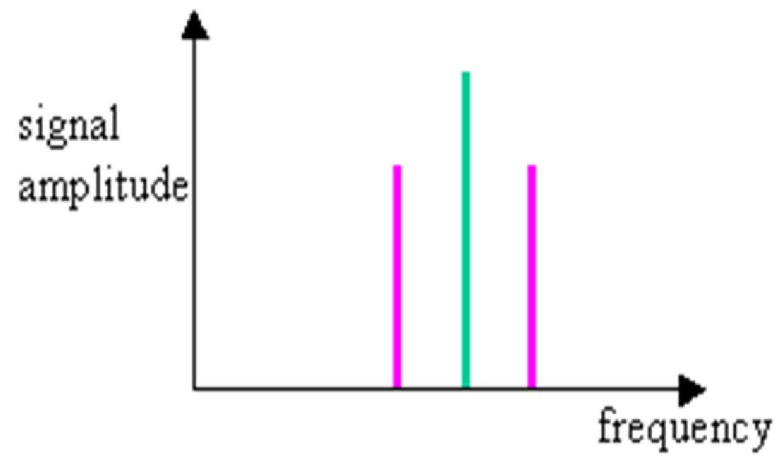


Frequency domain

We first transform the image to its frequency distribution. Then our black box system perform what ever processing it has to performed, and the output of the black box in this case is not an image, but a transformation. After performing inverse transformation, it is converted into an image which is then viewed in spatial domain.



Time domain

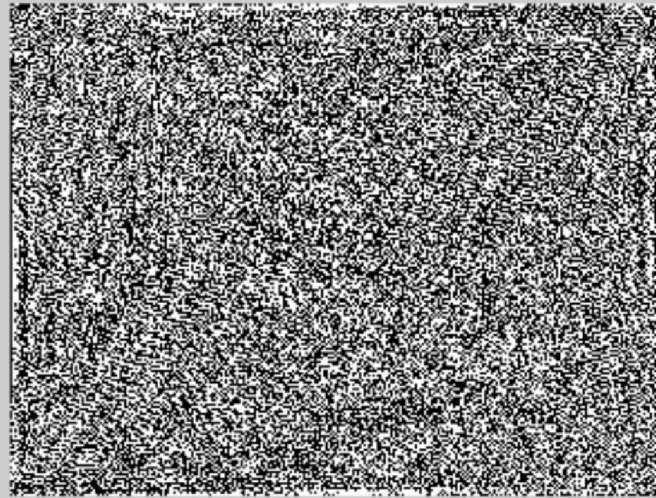


Frequency domain

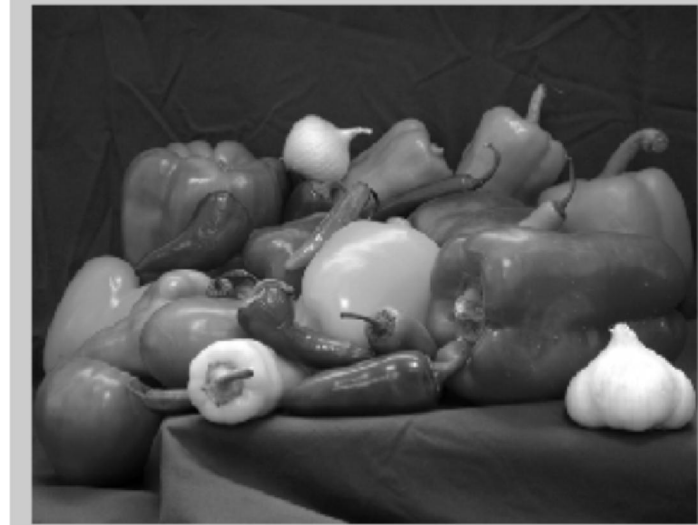
example



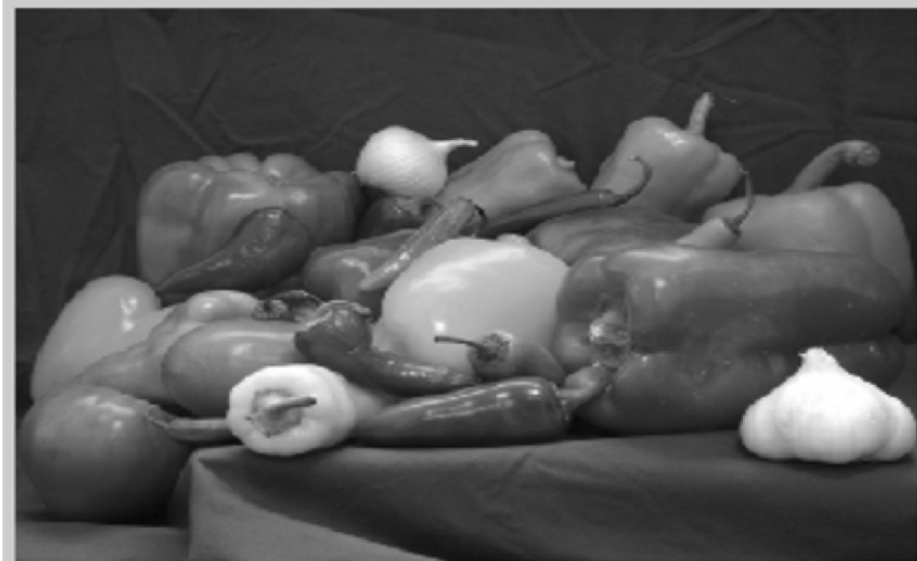
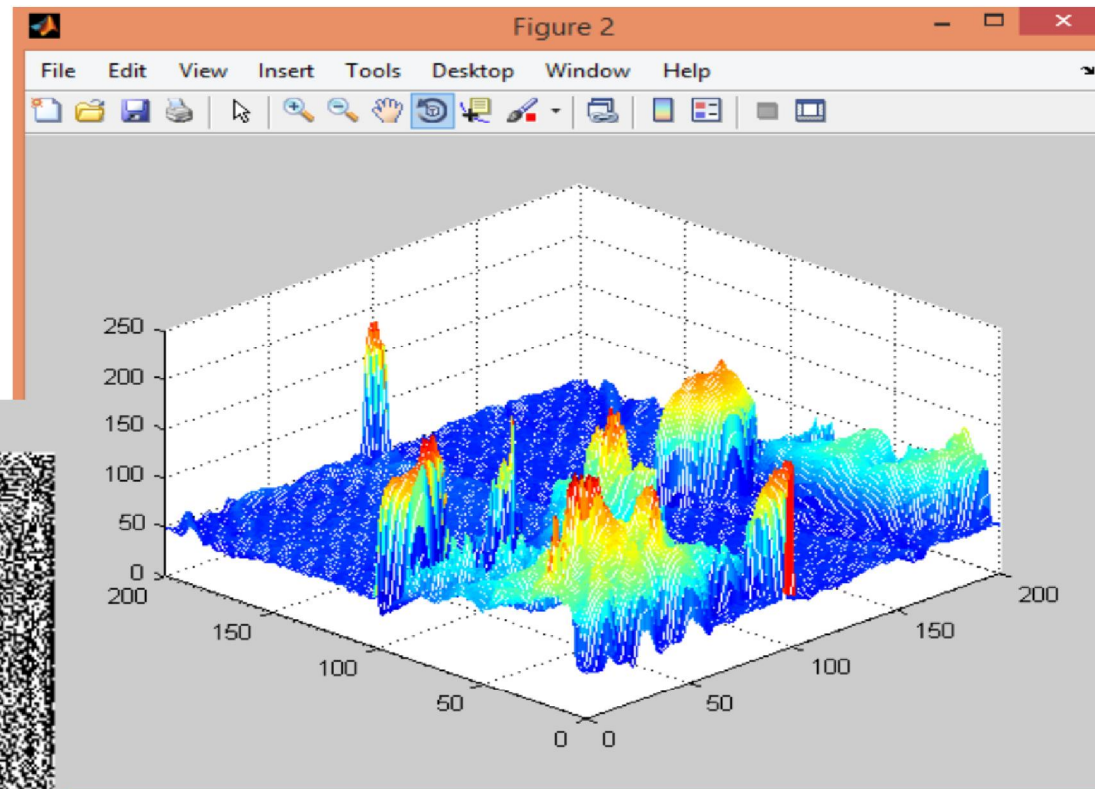
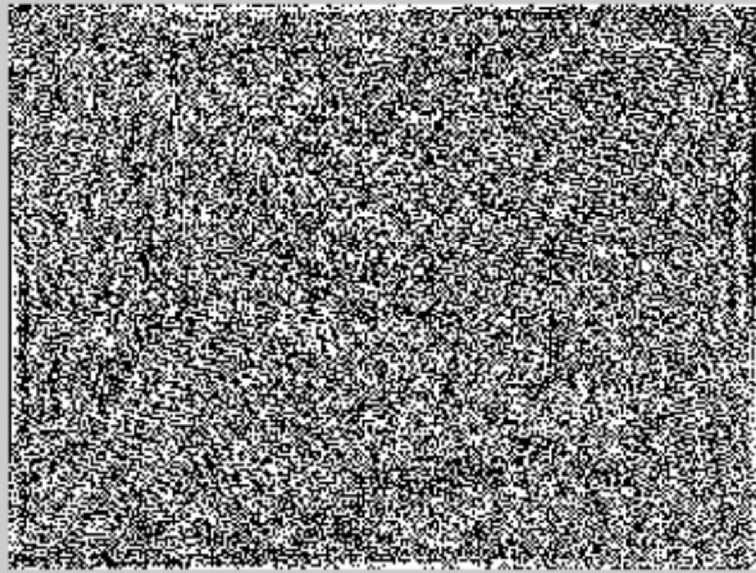
fourier transforms

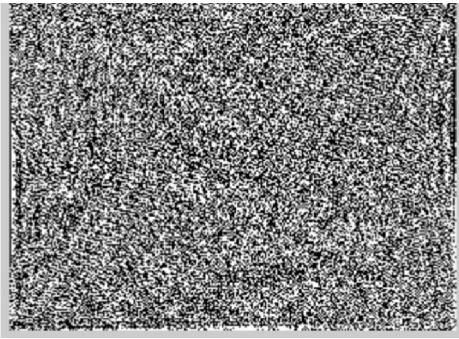


inverse fourier transforms



fourier transforms





Background

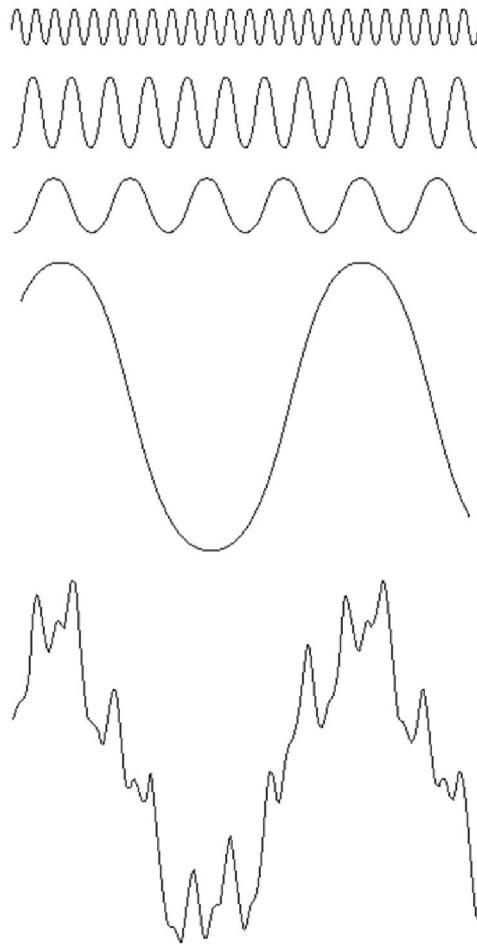
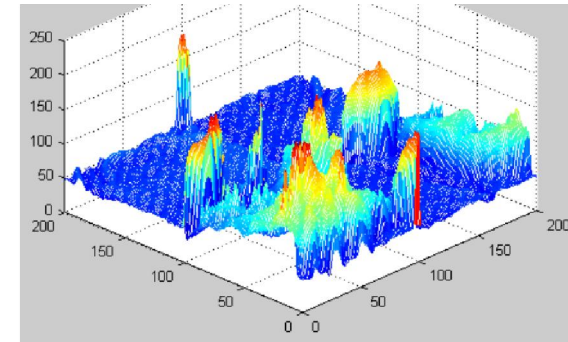
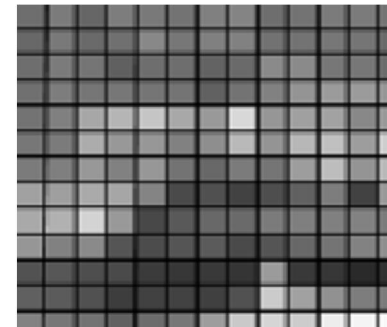


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

- The **frequency domain** refers to the plane of the two dimensional discrete Fourier transform of an image.
- The purpose of the Fourier transform is to represent a signal as a linear combination of sinusoidal signals of various frequencies.



54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

- The **one-dimensional** Fourier transform and its inverse

- Fourier transform (**continuous case**)

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx \quad \text{where } j = \sqrt{-1}$$

- Inverse Fourier transform:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

$$e^{j\theta} = \cos \theta + j \sin \theta$$

Euler formula

- The **two-dimensional** Fourier transform and its inverse

- Fourier transform (**continuous case**)

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy$$

- Inverse Fourier transform:

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)} du dv$$

2D Fourier transform for digital image

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)} \quad u = 0, 1, 2, \dots, M-1 \text{ and } v = 0, 1, 2, \dots, N-1$$

$$\tilde{X} = F_N X F_N$$

for image $f(x, y)$ of size $M \times N$

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & \dots & W_N^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

IDFT inverse discrete Fourier Transform

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{-j2\pi(ux/M + vy/N)} \quad x = 0, 1, 2, \dots, M-1 \text{ and } y = 0, 1, 2, \dots, N-1$$

$$X = \frac{1}{N^2} F_N^* \tilde{X} F_N^*$$

$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

where $\omega = e^{-\frac{2\pi j}{N}}$ is a primitive N th root of unity in which $j = \sqrt{-1}$.

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \dots & W_N^{1-N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{1-N} & W_N^{2(1-N)} & \dots & W_N^{-(N-1)} \end{bmatrix}$$

The Fourier transform

- The Fourier transform plays a critical role in a broad range of image processing applications, including enhancement, analysis, restoration, and compression.
- Example:
- If $f(m, n)$ is a function of two discrete spatial variables m and n , then the *two-dimensional Fourier transform* of $f(m, n)$ is defined by the relationship

$$F(\omega_1, \omega_2) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m, n) e^{-j\omega_1 m} e^{-j\omega_2 n}.$$

- The variables ω_1 and ω_2 are frequency variables; their units are radians per sample.
- $F(\omega_1, \omega_2)$ is often called the *frequency domain* representation of $f(m, n)$.
- $F(\omega_1, \omega_2)$ is a complex-valued function that is periodic both in ω_1 and ω_2 , with period 2π . Because of the periodicity, usually only the range $-\pi \leq \omega_1, \omega_2 \leq \pi$
- Note that $F(0,0)$ is the sum of all the values of $f(m,n)$. For this reason, $F(0,0)$ is often called the *constant component* or *DC component* of the Fourier transform.

$$F(\omega_1, \omega_2) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m,n) e^{-j\omega_1 m} e^{-j\omega_2 n}.$$

- The inverse of a transform is an operation that when performed on a transformed image produces the original image. The inverse two-dimensional Fourier transform is given by

$$f(m, n) = \frac{1}{4\pi^2} \int_{\omega_1=-\pi}^{\pi} \int_{\omega_2=-\pi}^{\pi} F(\omega_1, \omega_2) e^{j\omega_1 m} e^{j\omega_2 n} d\omega_1 d\omega_2.$$

- Roughly speaking, this equation means that $f(m, n)$ can be represented as a sum of an infinite number of complex exponentials (sinusoids) with different frequencies. The magnitude and phase of the contribution at the frequencies (ω_1, ω_2) are given by $F(\omega_1, \omega_2)$.

2D DFT and Inverse DFT (IDFT)

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

$f(x, y)$  $F(u, v)$

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

M, N : image size

x, y : image pixel position

u, v : spatial frequency

often used
short notation:

$$W_N = e^{-j2\pi/N}$$

The Meaning of DFT and Spatial Frequencies

- Important Concept

Any signal can be represented as a linear combination of a set of basic components

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

- Fourier components: sinusoidal patterns
 - Fourier coefficients: weighting factors assigned to the Fourier components
- Spatial frequency: The frequency of Fourier component
- Not to confused with electromagnetic frequencies (e.g., the frequencies associated with light colors)

Real Part, Imaginary Part, Magnitude, Phase, Spectrum

Real part: $R = \text{Real}(F)$

Imaginary part: $I = \text{Imag}(F)$

Magnitude-phase representation: $F(u, v) = |F(u, v)|e^{-j\phi(u, v)}$

Magnitude (spectrum): $|F(u, v)| = [R^2(u, v) + I^2(u, v)]^{1/2}$

Phase (spectrum): $\phi(u, v) = \tan^{-1} \left[\frac{I(u, v)}{R(u, v)} \right]$

Power Spectrum: $P(u, v) = |F(u, v)|^2$

2D DFT Properties

Spatial domain
differentiation:

$$\frac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v)$$

Frequency domain
differentiation:

$$(-jx)^n f(x, y) \Leftrightarrow \frac{\partial^n F(u, v)}{\partial u^n}$$

Distribution law:

$$\mathfrak{F}[f_1(x, y) + f_2(x, y)] = \mathfrak{F}[f_1(x, y)] + \mathfrak{F}[f_2(x, y)]$$

Laplacian:

$$\nabla^2 f(x, y) \Leftrightarrow -(u^2 + v^2)F(u, v)$$

Spatial domain

Periodicity: $f(x, y) = f(x + M, y) = f(x, y + N) = f(x + M, y + N)$

Frequency domain

periodicity: $F(u, v) = F(u + M, v) = F(u, v + N) = F(u + M, v + N)$

Computation of 2D-DFT

- To compute the 1D-DFT of a 1D signal \mathbf{x} (as a vector):

$$\tilde{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$$

To compute the inverse 1D-DFT:

$$\mathbf{x} = \frac{1}{N} \mathbf{F}_N^* \tilde{\mathbf{x}}$$

- To compute the 2D-DFT of an image \mathbf{X} (as a matrix):

$$\tilde{\mathbf{X}} = \mathbf{F}_N \mathbf{X} \mathbf{F}_N$$

To compute the inverse 2D-DFT:

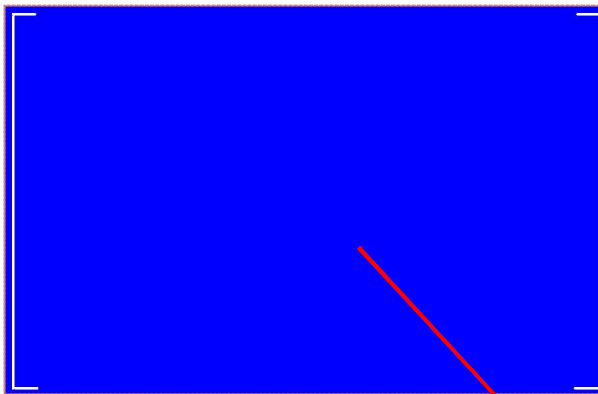
$$\mathbf{X} = \frac{1}{N^2} \mathbf{F}_N^* \tilde{\mathbf{X}} \mathbf{F}_N^*$$

Computation of 2D-DFT

remember Fourier transform matrices

$$W_N = e^{-j2\pi/N}, \text{ where, } N = 4$$

$$W_4 = e^{-j2\pi/4} = e^{-j\pi/2}$$

$$\mathbf{F}_N =$$


$$\mathbf{F}_N^* = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \dots & W_N^{1-N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{1-N} & W_N^{2(1-N)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

$$\tilde{\mathbf{X}} = \mathbf{F}_N \mathbf{X} \mathbf{F}_N$$

relationship: $\mathbf{F}_N^{-1} = \frac{1}{N} \mathbf{F}_N^*$ $\mathbf{X} = \frac{1}{N^2} \mathbf{F}_N^* \tilde{\mathbf{X}} \mathbf{F}_N$

In particular, for $N = 4$:

$$\mathbf{F}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$W_N = e^{-j2\pi/N}$$

$$e^{-j\phi} = \cos \phi - j \sin \phi$$

$$\mathbf{F}_4^* = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

An N -point DFT is expressed as an N -by- N matrix multiplication as $X = Wx$, where x is the original input signal, and X is the DFT of the signal.

The transformation W of size $N \times N$ can be defined as $W = \left(\frac{\omega^{jk}}{\sqrt{N}} \right)_{j,k=0,\dots,N-1}$, or equivalently:

$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix},$$

where $\omega = e^{-\frac{2\pi i}{N}}$ is a **primitive N th root of unity** in which $i = \sqrt{-1}$.

Calculate of Fourier matrices

$$W_N = e^{-j2\pi/N}, \text{ where, } N = 4$$

$$W_4 = e^{-j2\pi/4} = e^{-j\pi/2}$$

$$e^{-j\phi} = \cos \phi - j\sin \phi$$

$$\text{Where } \phi = \frac{\pi}{2}$$

$$e^{-j\phi} = \cos \frac{\pi}{2} - j\sin \frac{\pi}{2} = 0 - j = -j = W_N$$

$$e^{-jn\phi} = (\cos n\phi - j\sin n\phi)$$

$$e^{-j2\phi} = \cos 2\frac{\pi}{2} - j\sin 2\frac{\pi}{2} = -1 - 0 = -1$$
$$= W^2$$

$$\sin(3x) = \sin(2x)\cos(x) + \cos(2x)\sin(x)$$

$$\cos(3x) = \cos(2x + x) = \cos(2x)\cos(x) - \sin(2x)\sin(x)$$

Computation of 2D-DFT: Example

- A 4x4 image

$$\mathbf{X} = \begin{bmatrix} 1 & 3 & 6 & 8 \\ 9 & 8 & 8 & 2 \\ 5 & 4 & 2 & 3 \\ 6 & 6 & 3 & 3 \end{bmatrix}$$

- Compute its 2D-DFT:

$$\tilde{\mathbf{X}} = \mathbf{F}_4 \mathbf{X} \mathbf{F}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 & 3 & 6 & 8 \\ 9 & 8 & 8 & 2 \\ 5 & 4 & 2 & 3 \\ 6 & 6 & 3 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

MATLAB function: *fft2*

$$= \begin{bmatrix} 21 & 21 & 19 & 16 \\ -4-3j & -1-2j & 4-5j & 5+j \\ -9 & -7 & -3 & 6 \\ -4+3j & -1+2j & 4+5j & 5-j \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

lowest frequency
component

highest frequency
component

$$= \begin{bmatrix} 77 & 2-5j & 3 & 2+5j \\ 4-9j & -11+8j & -4-7j & -5-4j \\ -13 & -6+13j & -11 & -6-13j \\ 4+9j & -5+4j & -4+7j & -11-8j \end{bmatrix}$$

Computation of 2D-DFT: Example

$$\tilde{\mathbf{X}} = \begin{bmatrix} 77 & 2-5j & 3 & 2+5j \\ 4-9j & -11+8j & -4-7j & -5-4j \\ -13 & -6+13j & -11 & -6-13j \\ 4+9j & -5+4j & -4+7j & -11-8j \end{bmatrix}$$

Real part:

$$\tilde{\mathbf{X}}_{real} = \begin{bmatrix} 77 & 2 & 3 & 2 \\ 4 & -11 & -4 & -5 \\ -13 & -6 & -11 & -6 \\ 4 & -5 & -4 & -11 \end{bmatrix}$$

Imaginary part:

$$\tilde{\mathbf{X}}_{imag} = \begin{bmatrix} 0 & -5 & 0 & 5 \\ -9 & 8 & -7 & -4 \\ 0 & 13 & 0 & -13 \\ 9 & 4 & 7 & -8 \end{bmatrix}$$

Magnitude:

$$\tilde{\mathbf{X}}_{magnitude} = \begin{bmatrix} 77 & 5.39 & 3 & 5.39 \\ 9.85 & 13.60 & 8.06 & 6.4 \\ 13 & 14.32 & 11 & 14.32 \\ 9.85 & 6.40 & 8.06 & 13.60 \end{bmatrix}$$

Phase:

$$\tilde{\mathbf{X}}_{phase} = \begin{bmatrix} 0 & -1.19 & 0 & 1.19 \\ -1.15 & 2.51 & -2.09 & -2.47 \\ 3.14 & 2.00 & 3.14 & -2.00 \\ 1.15 & 2.47 & 2.09 & -2.51 \end{bmatrix}$$

Computation of 2D-DFT: Example

- Compute the inverse 2D-DFT:

$$\mathbf{F}_4^* \tilde{\mathbf{X}} \mathbf{F}_4^* = \frac{1}{4^2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} 77 & 2-5j & 3 & 2+5j \\ 4-9j & -11+8j & -4-7j & -5-4j \\ -13 & -6+13j & -11 & -6-13j \\ 4+9j & -5+4j & -4+7j & -11-8j \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

$$= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} 21 & 21 & 19 & 16 \\ -4-3j & -1-2j & 4-5j & 5+j \\ -9 & -7 & -3 & 6 \\ -4+3j & -1+2j & 4+5j & 5-j \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 3 & 6 & 8 \\ 9 & 8 & 8 & 2 \\ 5 & 4 & 2 & 3 \\ 6 & 6 & 3 & 3 \end{bmatrix} = \mathbf{X}$$

MATLAB function: *ifft2*

Fourier Transforms

تحويلات فوريير الرياضية للصور الرقمية

The Fourier transform is a representation of an image as a sum of complex exponentials of varying magnitudes, frequencies, and phases.

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

- Fourier components: sinusoidal patterns

$F(u, v)$ Fourier Coefficients

$f(x, y)$ is an image

The **Fourier Transform** is an important **image** processing tool which is used to decompose an **image** into its sine and cosine components. The output of the **transformation** represents the **image** in the **Fourier** or frequency domain, while the input **image** is the spatial domain equivalent.

The **DFT is used** to convert an **image** from the spatial domain into frequency domain, in other words it allows us to separate high frequency from low frequency coefficients and neglect or alter specific frequencies leading to an **image** with less information but still with a convenient level of quality .

Fourier transform is a mathematical formula by which we can extract out the frequency domain components of a continuous time domain signal. Using fourier transform we can process time domain signal in frequency domain. We can use various Frequency domain filters to process the signal.

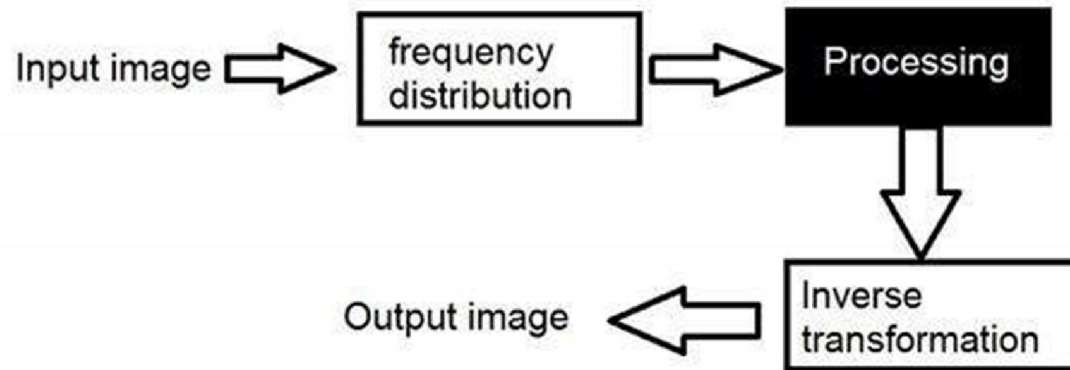
If signal is discrete, Discrete Fourier Transform use to analyse discrete signal

The **Fast Fourier Transform (FFT)** is commonly used to transform an **image** between the spatial and frequency domain. Unlike other domains such as Hough and Radon, the **FFT** method preserves all original data. Plus, **FFT** fully transforms **images** into the frequency domain, unlike time-frequency or wavelet transforms.

Difference between **spatial domain** and **frequency domain**

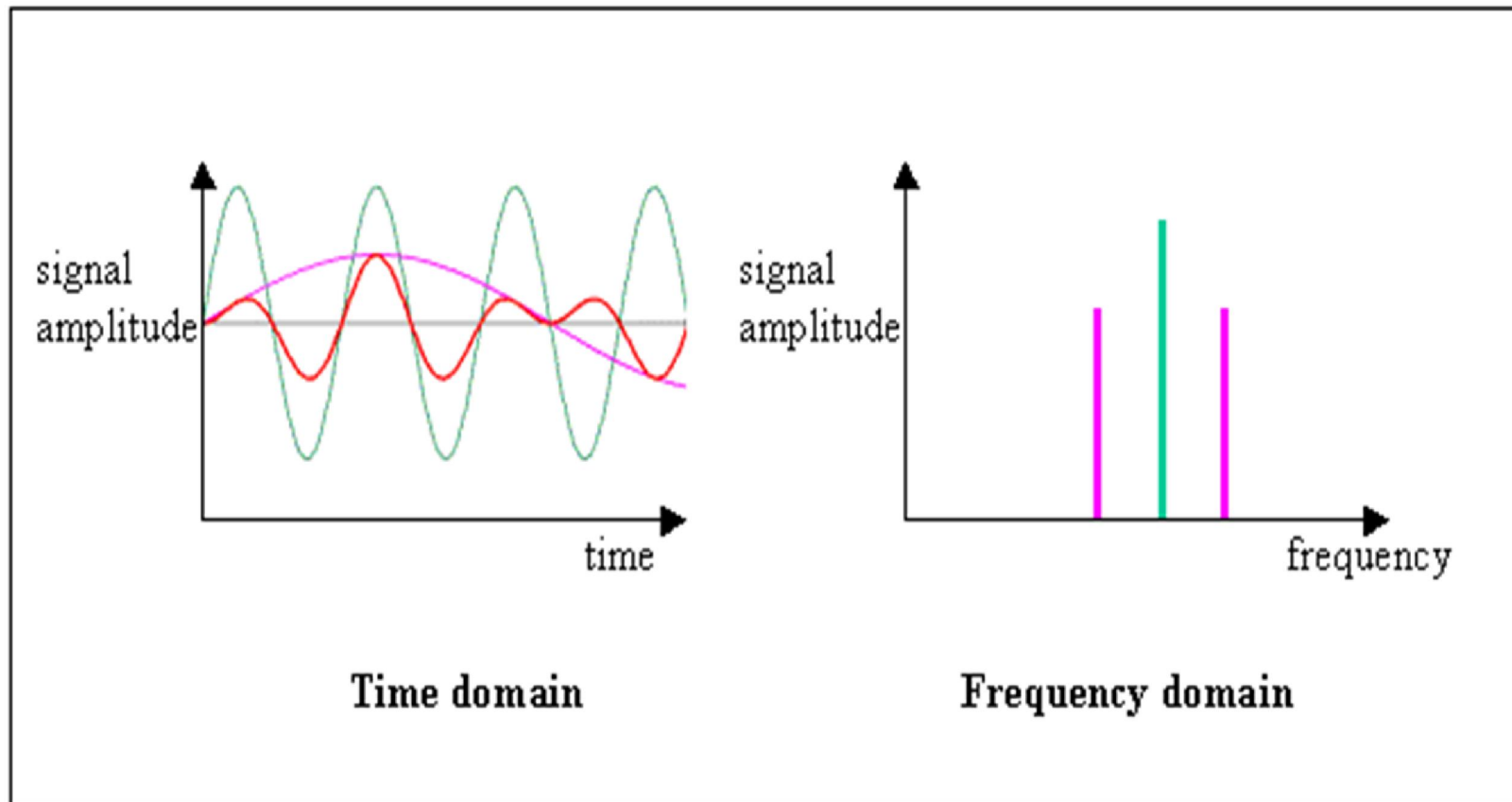


In spatial domain, we deal with images as it is. The value of the pixels of the image change with respect to scene. Whereas in frequency domain, we deal with the rate at which the pixel values are changing in spatial domain.



Frequency domain

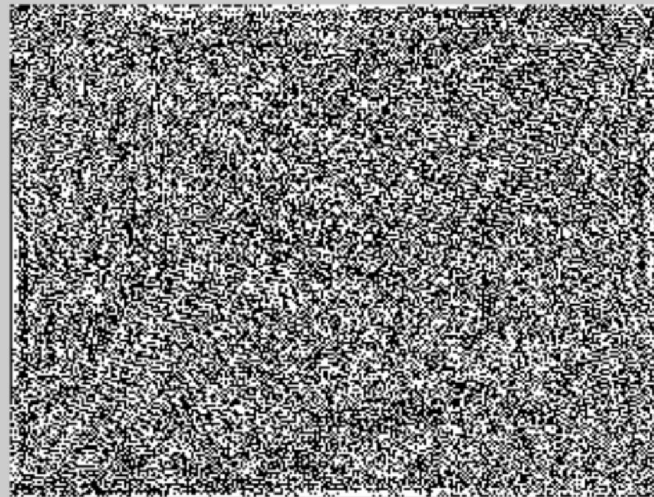
We first transform the image to its frequency distribution. Then our black box system perform what ever processing it has to performed, and the output of the black box in this case is not an image, but a transformation. After performing inverse transformation, it is converted into an image which is then viewed in spatial domain.



example



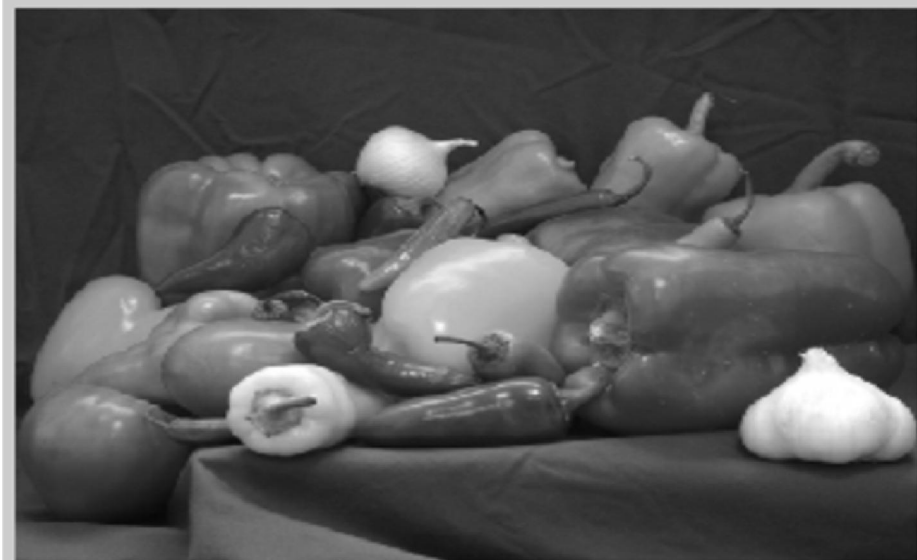
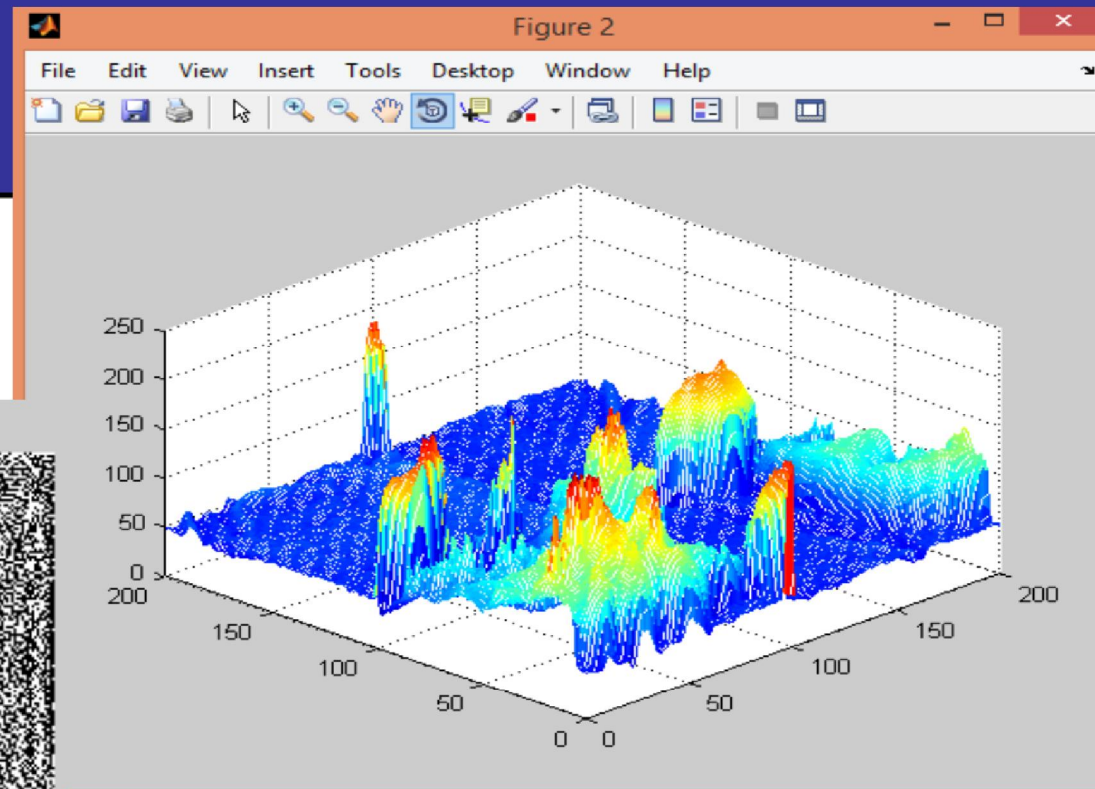
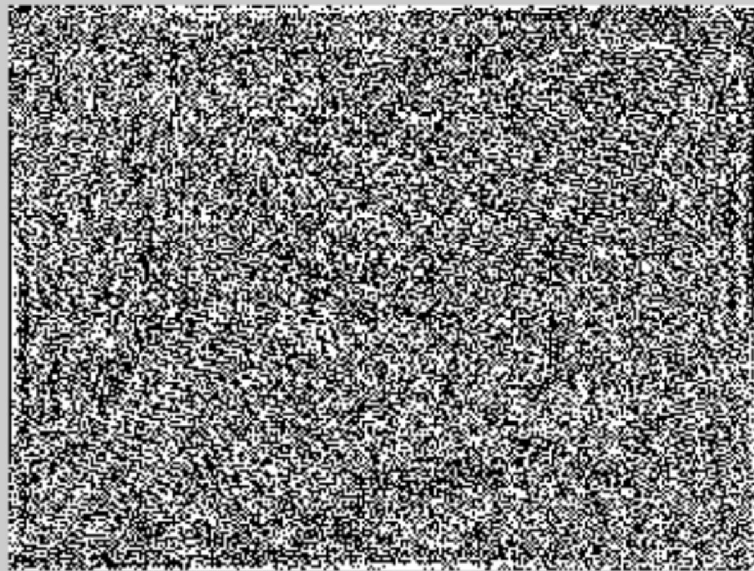
fourier transforms

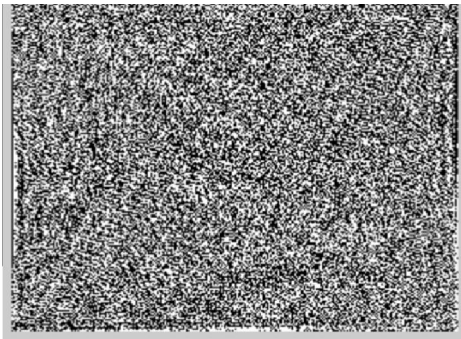


inverse fourier transforms



fourier transforms





Background

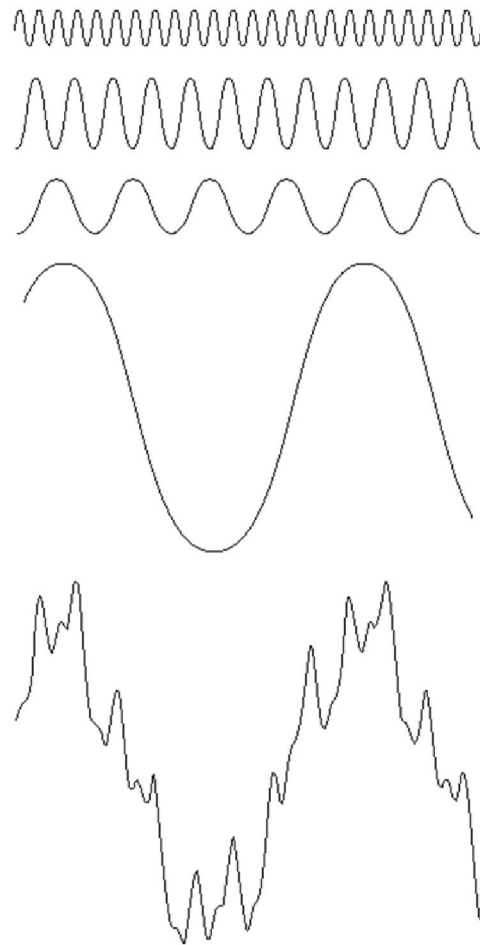
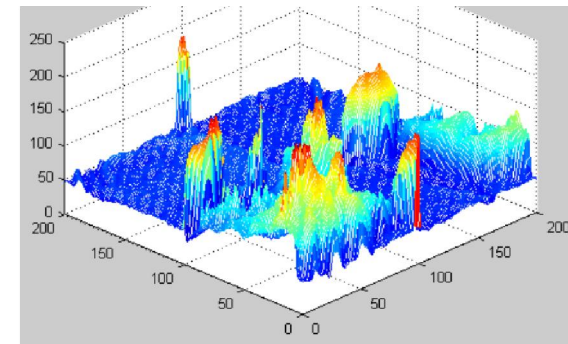
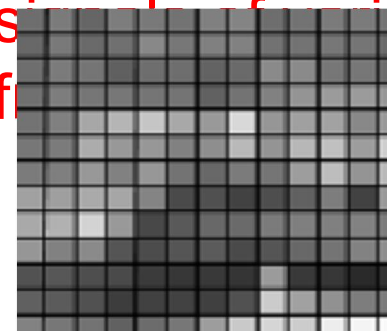


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

The **frequency domain** refers to the plane of the two dimensional discrete Fourier transform of an image.

The purpose of the Fourier transform is to represent a signal as a linear combination of sinusoidal functions.



54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

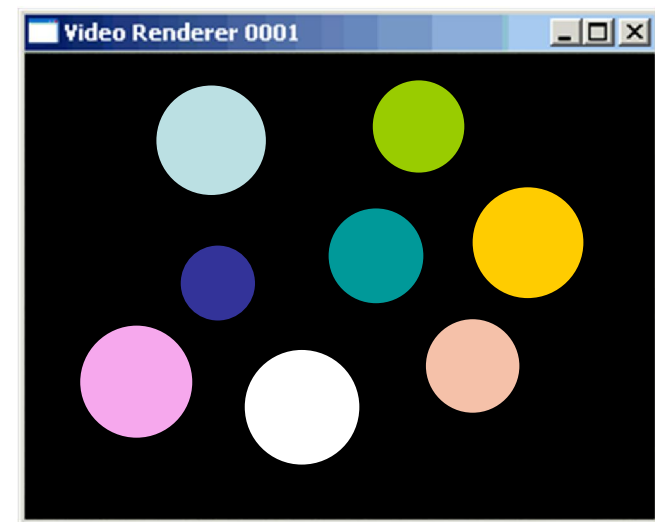
Lect 8. Image Segmentation issue:

- The segmentation problem
- Finding points, lines and edges

The Segmentation Problem

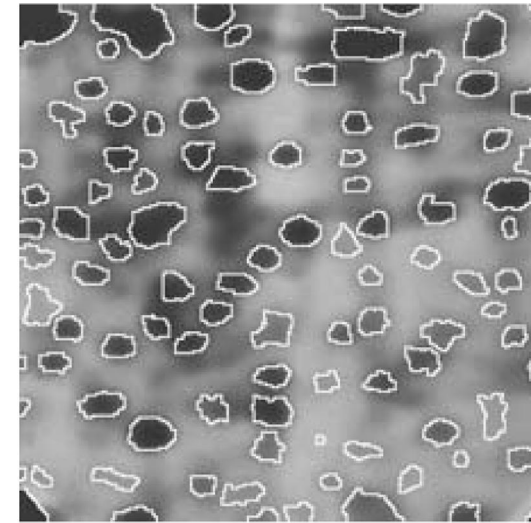
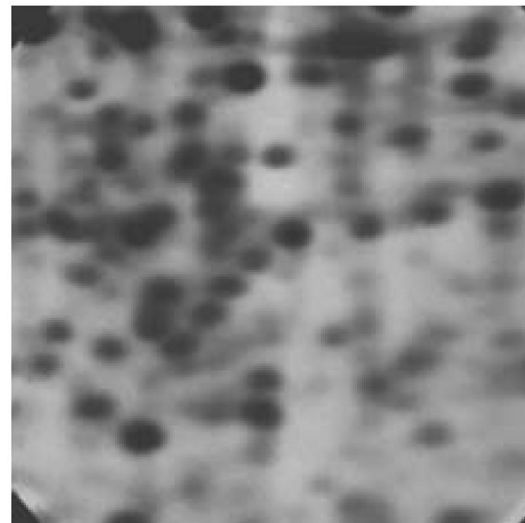
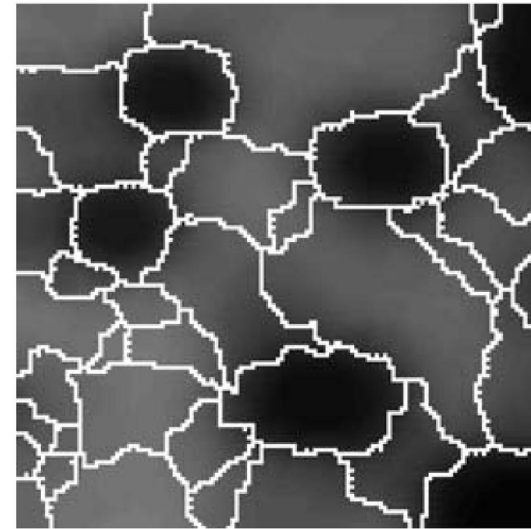
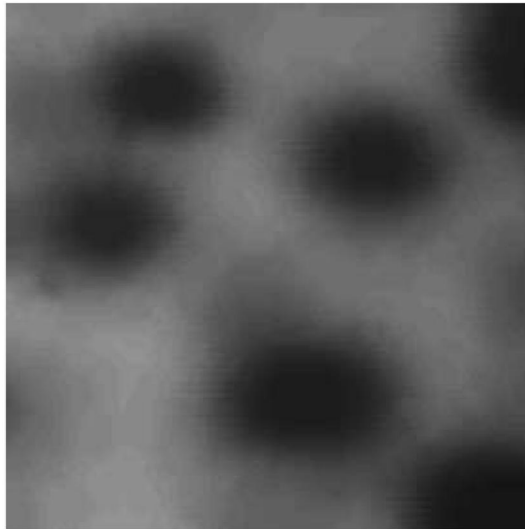
Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image

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

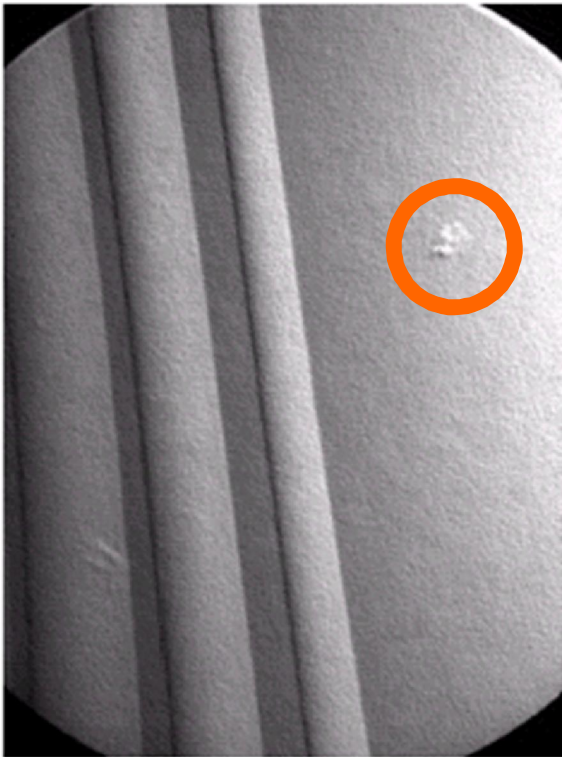
Point detection can be achieved simply using the mask below:

-1	-1	-1
-1	8	-1
-1	-1	-1

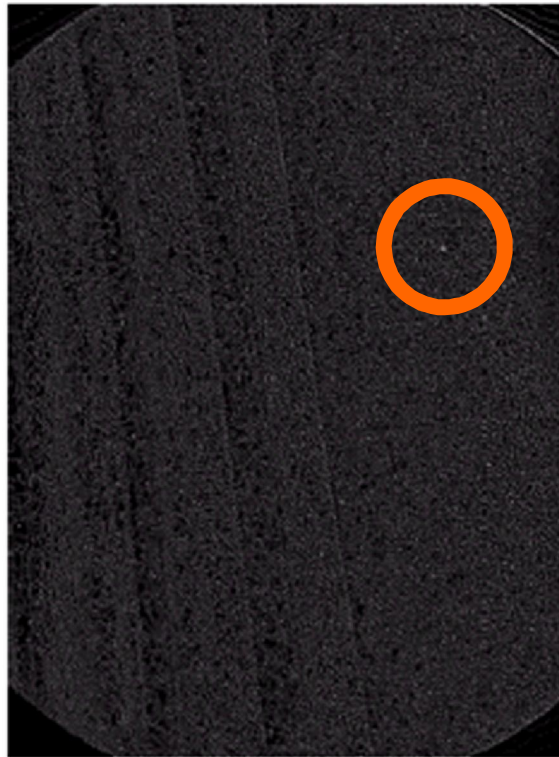
Points are detected at those pixels in the subsequent filtered image that are above a set threshold

Point Detection (cont...)

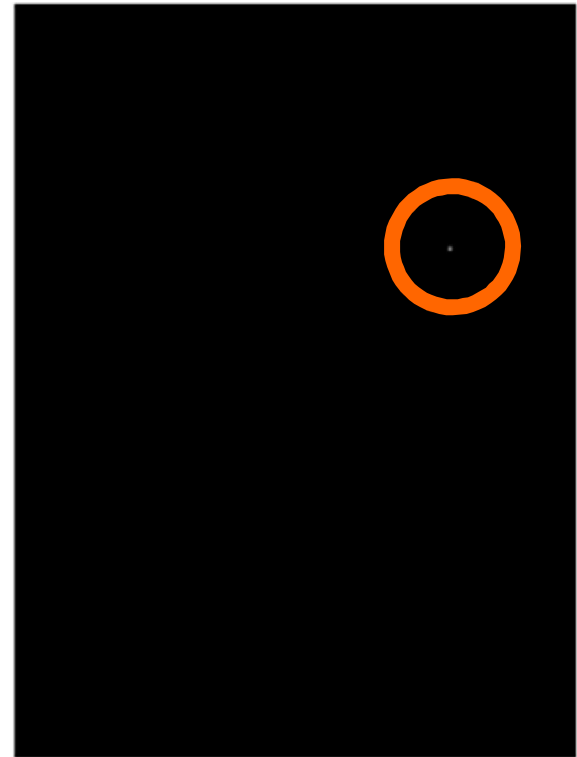
Images taken from Gonzalez & Woods, Digital Image Processing (2002)



X-ray image of
a turbine blade



Result of point
detection



Result of
thresholding

The next level of complexity is to try to detect lines

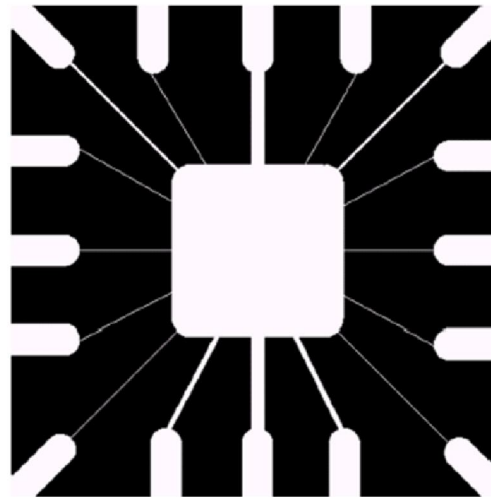
The masks below will extract lines that are one pixel thick and running in a particular direction

-1	-1	-1	-1	-1	2	-1	2	-1	2	-1	-1
2	2	2	-1	2	-1	-1	2	-1	-1	2	-1
-1	-1	-1	2	-1	-1	-1	2	-1	-1	-1	2
Horizontal			+45°			Vertical			-45°		



Line Detection (cont...)

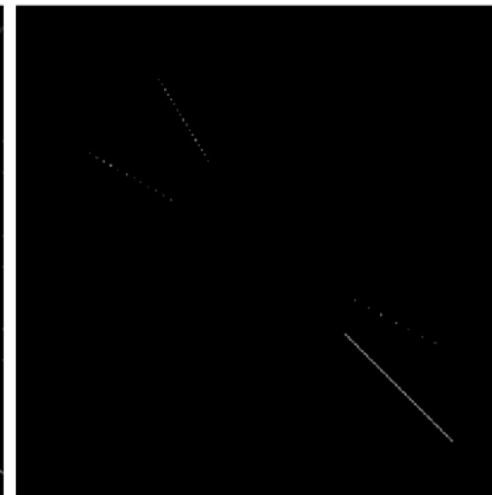
Binary image of a wire
bond mask



After
processing
with -45° line
detector

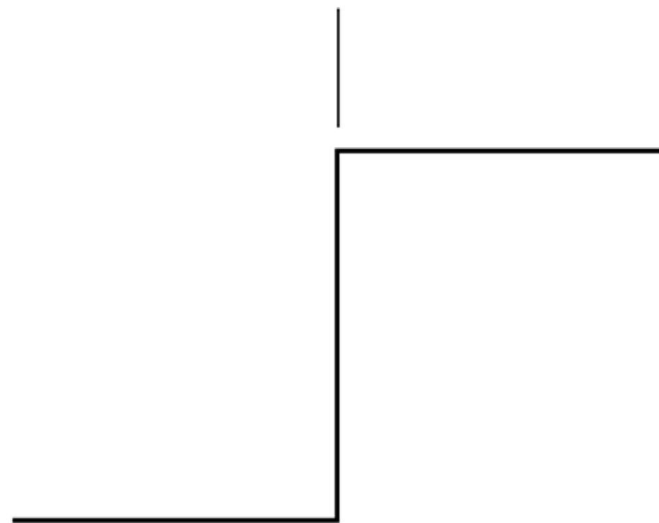


Result of
thresholding
filtering result



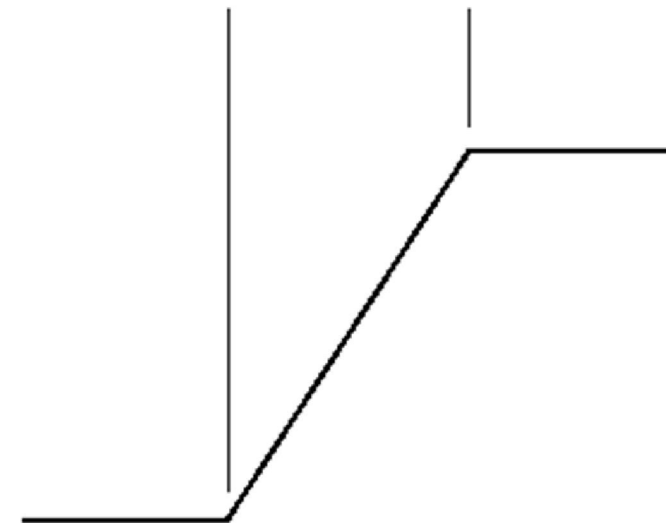
An edge is a set of connected pixels that lie on the boundary between two regions

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

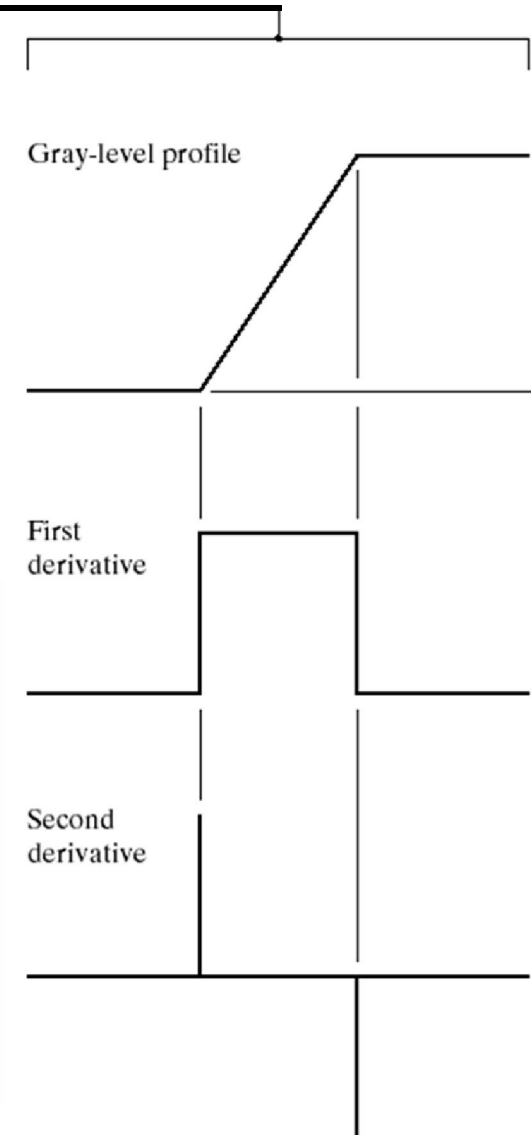


Edges & Derivatives

We have already spoken about how derivatives are used to find discontinuities

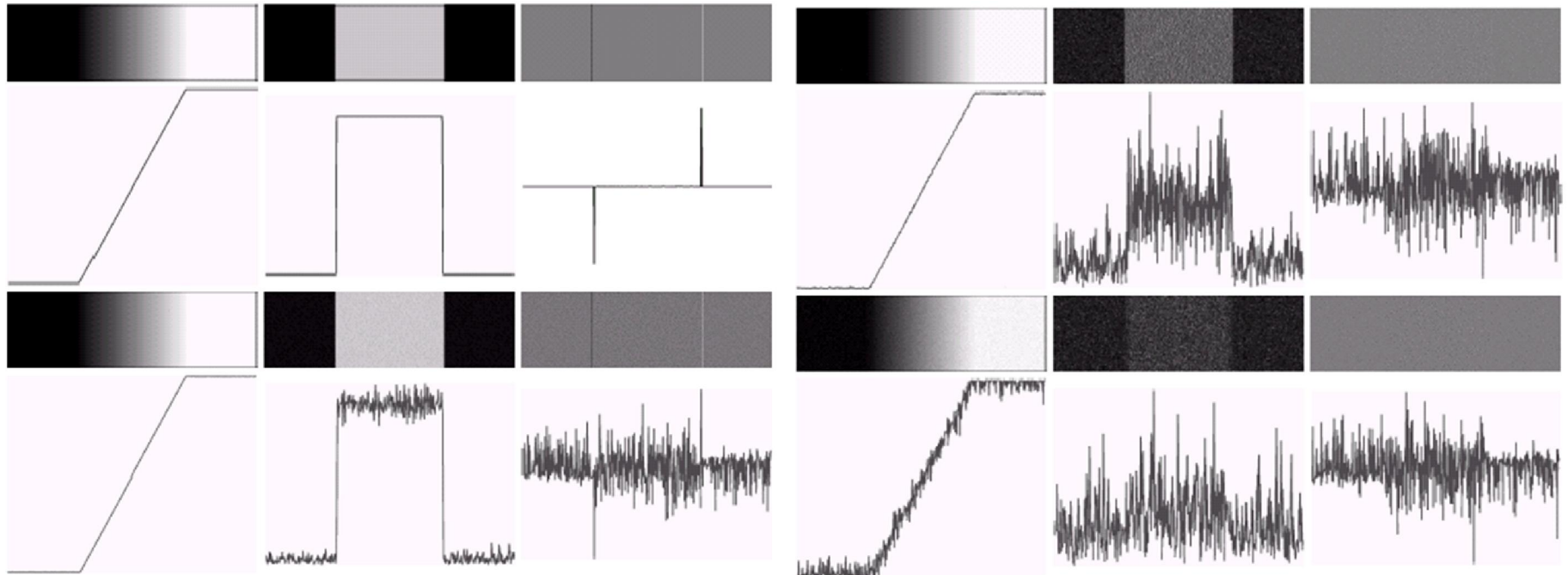
1st derivative tells us where an edge is

2nd derivative can be used to show edge direction



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	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

Prewitt

-1	0	0	-1
0	1	1	0

Roberts

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

Sobel



Edge Detection Example

Original Image



Horizontal Gradient Component



Vertical Gradient Component



Combined Edge Image



Edge Detection Example



Edge Detection Example

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



Edge Detection Example

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



Edge Detection Example

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



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

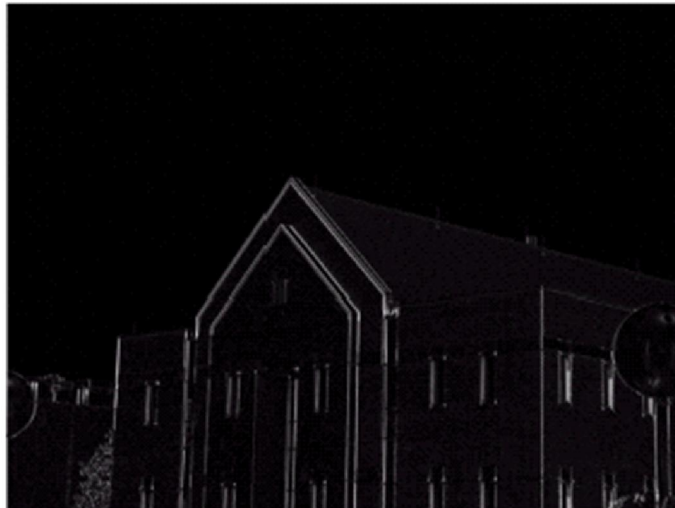
One way to overcome this is to smooth images prior to edge detection

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

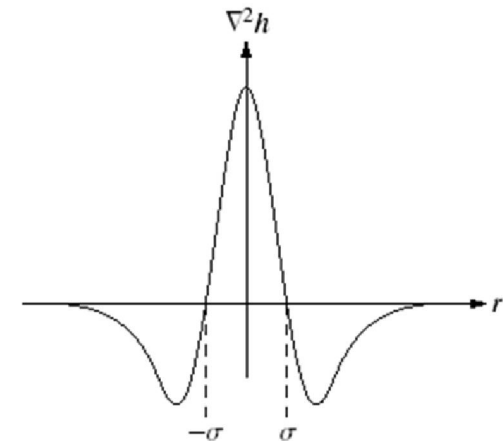
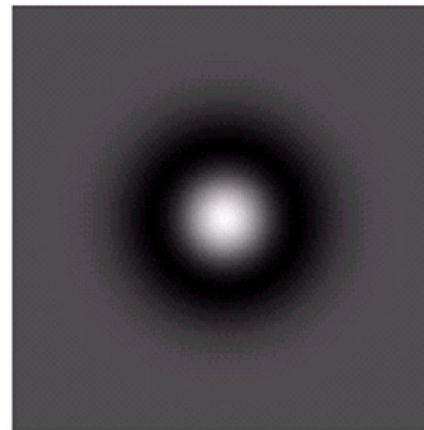
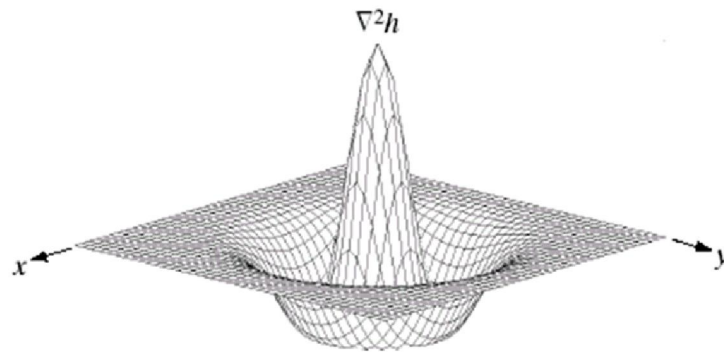
0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

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

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

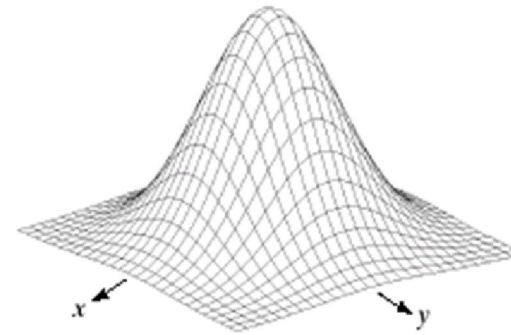
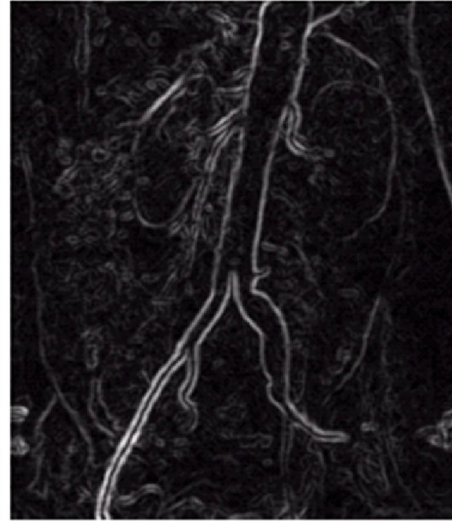
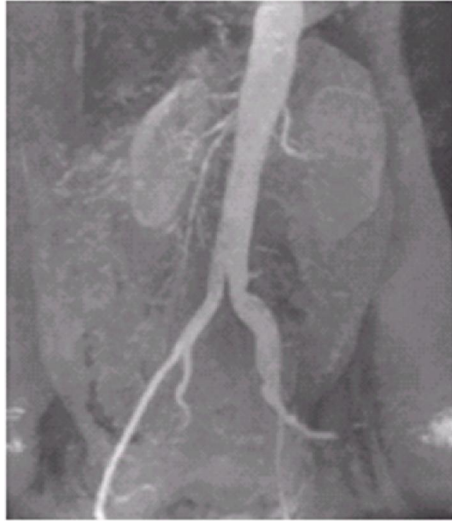
Laplacian Of Gaussian

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

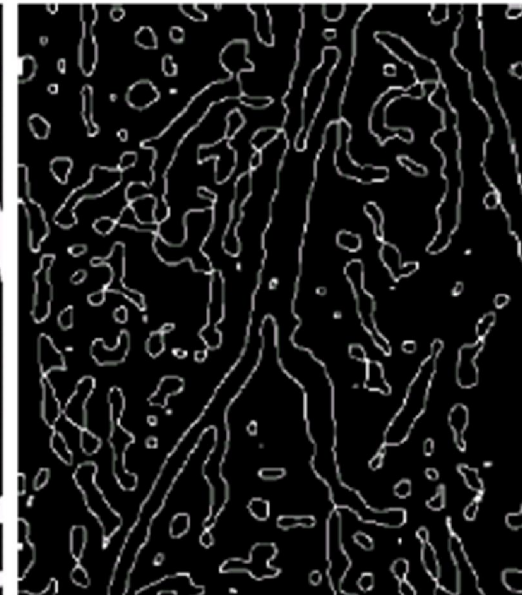


0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Laplacian Of Gaussian Example



-1	-1	-1
-1	8	-1
-1	-1	-1



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



Convolution and Correlation in Image Processing

Convolution and Correlation in Image Processing is quite similar except that the kernel are rotated 180 degree between these 2 operation.

For example, image A = [10 20 30 ; 40 50 60; 70 80 90] and the kernel h = [1 2 3; 4 5 6; 7 8 9], correlation of 2 matrices is sum of the elements multiplication of 2 matrices

10 * 1	20 * 2	30 * 3
40 * 4	50 * 5	60 * 6
70 * 7	80 * 8	90 * 9

Correlation result = (10 x 1) + (20 x 2) + (30 x 3) + (40 x 4) + (50 x 5) + (60 x 6) + (70 x 7) + (80 x 8) + (90 x 9) = 2850

1	2	3
4	5	6
7	8	9

7	4	1
8	5	2
9	6	3

9	8	7
6	5	4
3	2	1

For the convolution, the operation is almost the same, except that the kernel is rotated 180 degree.

10 * 9	20 * 8	30 * 7
40 * 6	50 * 5	60 * 4
70 * 3	80 * 2	90 * 1

Convolution result = (10 x 9) + (20 x 8) + (30 x 7) + (40 x 6) + (50 x 5) + (60 x 4) + (70 x 3) + (80 x 2) + (90 x 1) = 1650

In the real Image convolution and correlation, the kernel is sliding over every single pixel and performs above operation to form a new image.

Frequency Domain Versions of Spatial Filters

The following **convolution theorem** shows an interesting relationship between the spatial domain and frequency domain:

$$f(x,y) * h(x,y) \Leftrightarrow H(u,v) F(u,v)$$

and, conversely,

$$f(x,y) h(x,y) \Leftrightarrow H(u,v) * G(u,v)$$

where the symbol "*" indicates **convolution** of the two functions. The important thing to extract out of this is that **the multiplication of two Fourier transforms** corresponds to the convolution of the associated functions in the spatial domain. **This means we can perform linear spatial filters as a simple component-wise multiply in the frequency domain.**

Lecture 9

- Frequency Domain Filtering

Dr Nassir H. Salman

One-Dimensional Fourier transform and its inverse

Fourier transforms pair

- The Fourier transform , $F(u)$, of a single variable, continuous function , $f(x)$ is defined by :

$$1) \quad F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx \quad j = \sqrt{-1}$$

- where conversely , given $F(u)$, we can obtain $f(x)$ by means of the inverse Fourier transform:

$$2) \quad f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

They indicate the important fact that a function can be recovered from its transform

Fourier transform

- The two equations above can be extended to two variables , and : following Fourier transform and it's inverse of 2-D

$$1) \quad F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy$$

$$2) \quad f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)} du dv$$

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)} \quad u=0, 1, 2, \dots, M-1 \text{ and } v=0, 1, 2, \dots, N-1$$

for image $f(x, y)$ of size $M \times N$

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{-j2\pi(ux/M + vy/N)} \quad x=0, 1, 2, \dots, M-1 \text{ and } y=0, 1, 2, \dots, N-1$$

Fourier transform

- The Fourier transform of a discrete function **(DFT)** of one variable , $f(x)$, $x = 0, 1, \dots, M-1$, is given by the equation:

- 1)
$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j 2 \pi u x / M} \quad \text{for } u = 0, 1, 2, \dots, M-1.$$

- Inverse **(DFT)** , See $1/M$ Value

- 2)
$$f(x) = \sum_{u=0}^{M-1} F(u) e^{j 2 \pi u x / M} \quad \text{or } x = 0, 1, 2, \dots, M-1.$$

$\frac{1}{\sqrt{M}}$

How to Compute 1-D $F(u)$

- To compute $F(u)$ we start by substituting $u = 0$ in the exponential term and then summing for all values of x . we then substitute $u = 1$ in the exponential and repeat the summation over all values of x . we repeat this process for all M values of u in order to obtain the complete Fourier transform.

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M}$$

- It takes approximately M^2 summations and multiplications to compute discrete Fourier transforms. Like $f(x)$, the transform is a discrete quantity, and it has the same components as $f(x)$. Similar comments apply to the computation of the inverse Fourier transform

- An important property of the discrete transform pair is that :the discrete Fourier transform and its inverse always exists. This can be shown by substituting either of

$$1) \bullet F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux / M} \quad \text{or} \quad 2) f(x) = \sum_{u=0}^{M-1} F(u) e^{j2\pi ux / M}$$

into other and making orthogonality of the exponential ,we need the following orthogonality property

$$\sum_{x=0}^{M-1} e^{j2\pi rx / M} e^{-j2\pi ux / M} = \begin{cases} M & \text{if } r = u \\ 0 & \text{otherwise} \end{cases}$$

- The concept of the frequency domain follows from Euler's formula:

$$e^{j\theta} = \cos \theta + j \sin \theta$$

- Fourier transform becomes: After we substitute in 1-D Fourier T

$$1) \quad F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) [\cos 2\pi ux / M - j \sin 2\pi ux / M] \quad \text{for } u = 0, 1, 2, \dots, M-1.$$

- Thus we see that each term of Fourier transform [that is, the value of $F(u)$ for each value of u] composed of the sum of all values of the function $f(x)$. The values of $f(x)$, in turn, are multiplied by sines and cosines of various frequencies. The domain (values of u) over which the values of $F(u)$ range is appropriately called frequency domain.
- The Fourier transform may be viewed as a “mathematical prism” that separates a function into various components based on frequency contents.

- In general we see from equations:

$$1) \quad F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j 2 \pi u x / M}$$

$$1) \quad F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) [\cos 2 \pi u x / M - j \sin 2 \pi u x / M]$$

That the components of the Fourier transform are complex quantities. As in analysis of complex numbers, we find it is convenient sometimes to express $F(u)$ in polar coordinates:

$$1) \quad F(u) = |F(u)| e^{-j \phi(u)}$$

Where;

$$|F(u)| = [R^2(u) + I^2(u)]^{1/2}$$

- is called magnitude or spectrum of the Fourier transform , and

$$\phi(u) = \tan^{-1} \left[\frac{I(u)}{R(u)} \right]$$

is called the phase angle or phase spectrum of the transform

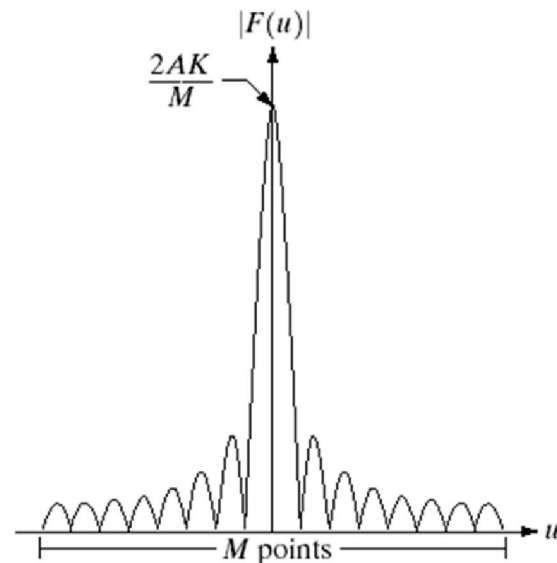
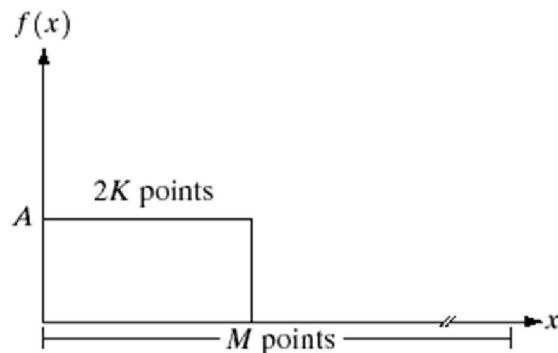
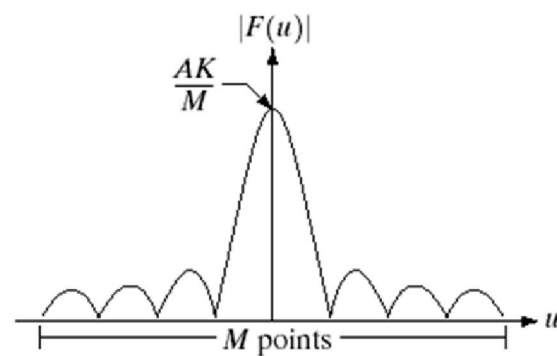
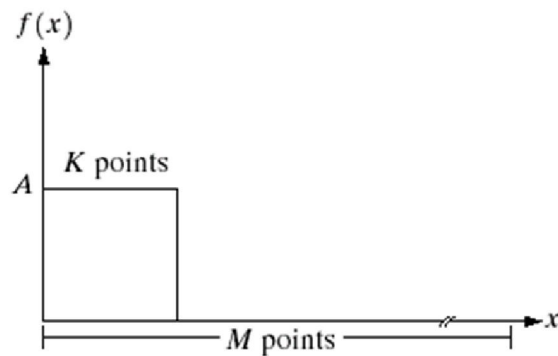
The power spectrum or spectral density, defined as the square of the Fourier spectrum:

$$P(u) = |F(u)|^2 = R^2(u) + I^2(u)$$

Lecture today 9:

Simple 1-D Example of the DFT

a) $M=1024$, $A=1$, $K=8$ non – zero POINTS



a	b
c	d

FIGURE 4.2 (a) A discrete function of M points, and (b) its Fourier spectrum. (c) A discrete function with twice the number of nonzero points, and (d) its Fourier spectrum.

Chapter 4

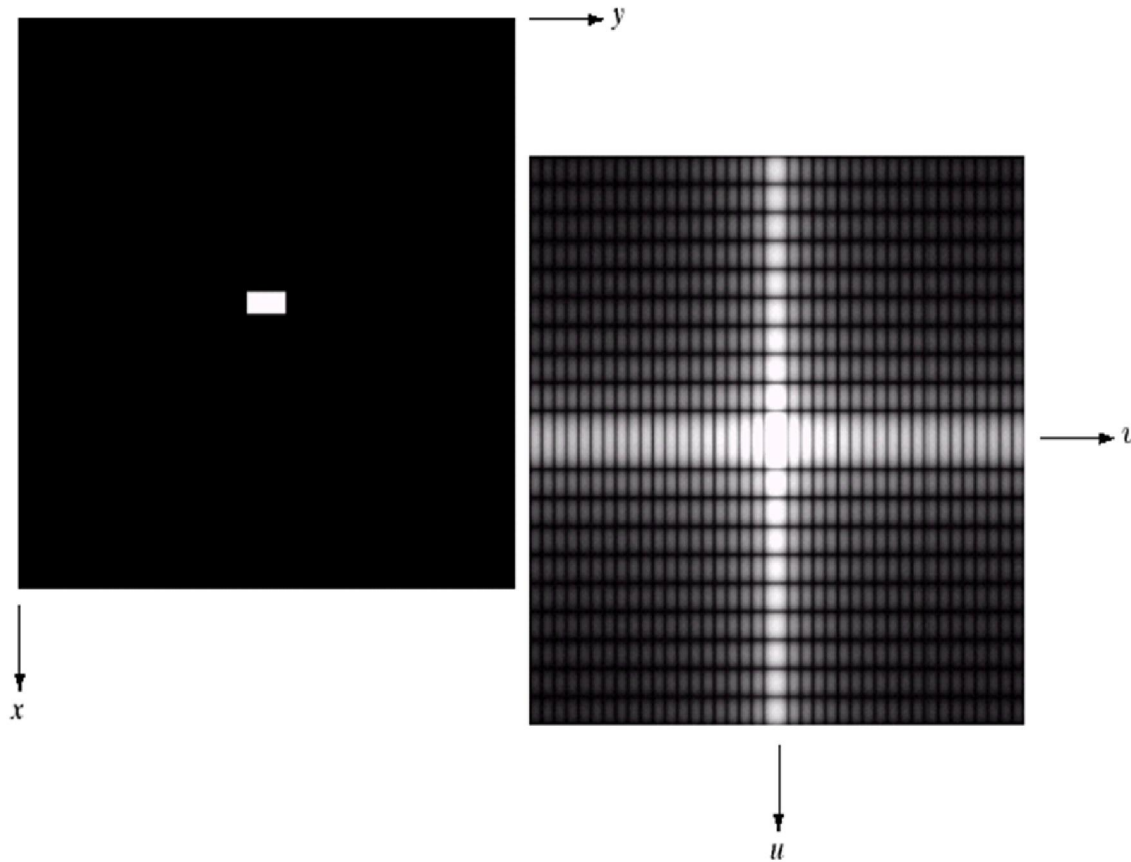
Image Enhancement in the Frequency Domain

a b

FIGURE 4.3

(a) Image of a 20×40 white rectangle on a black background of size 512×512 pixels.

(b) Centered Fourier spectrum shown after application of the log transformation given in Eq. (3.2-2). Compare with Fig. 4.2.



Chapter 4

Image Enhancement in the Frequency Domain

Frequency domain filtering operation

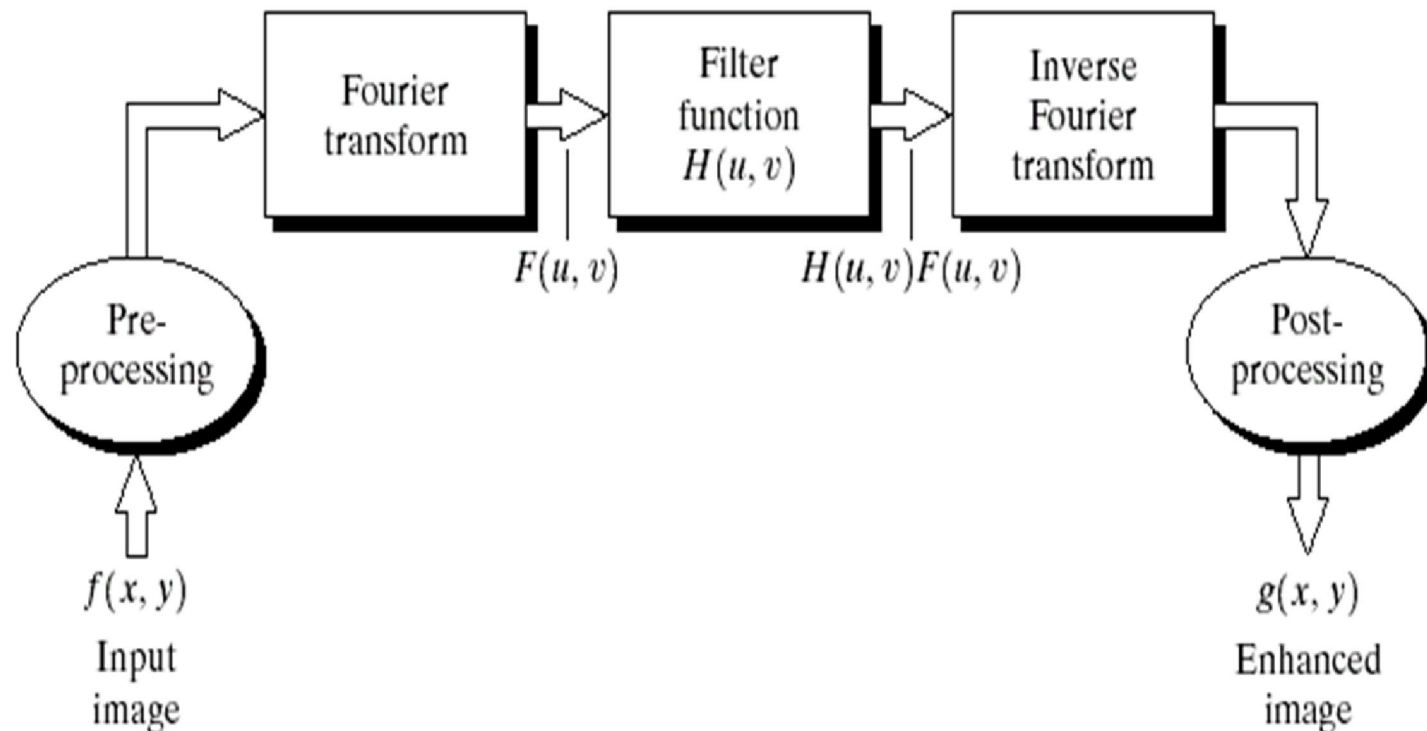
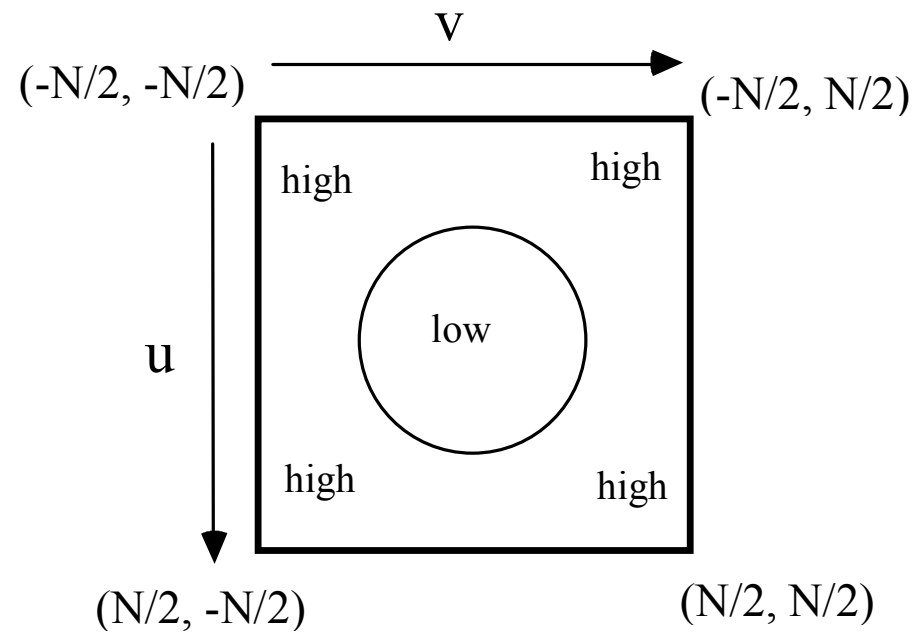


FIGURE 4.5 Basic steps for filtering in the frequency domain.

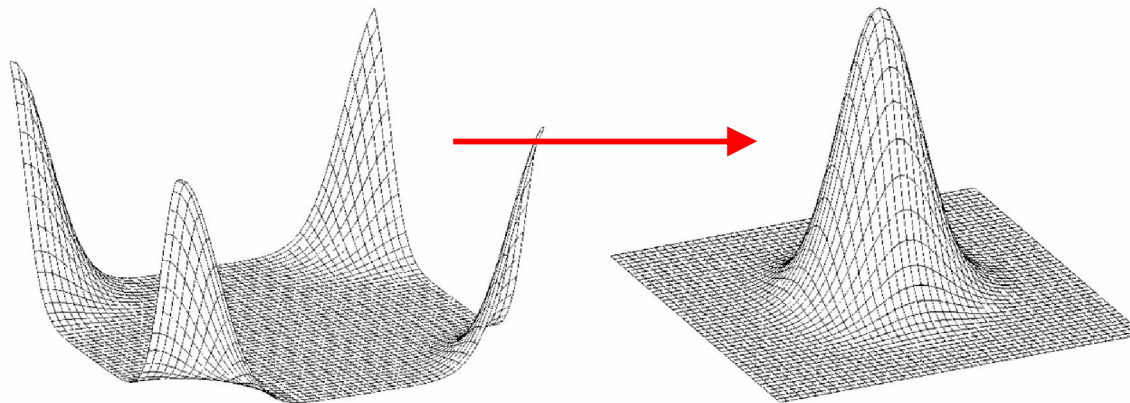
Centered Representation



MATLAB
function: *fftshift*

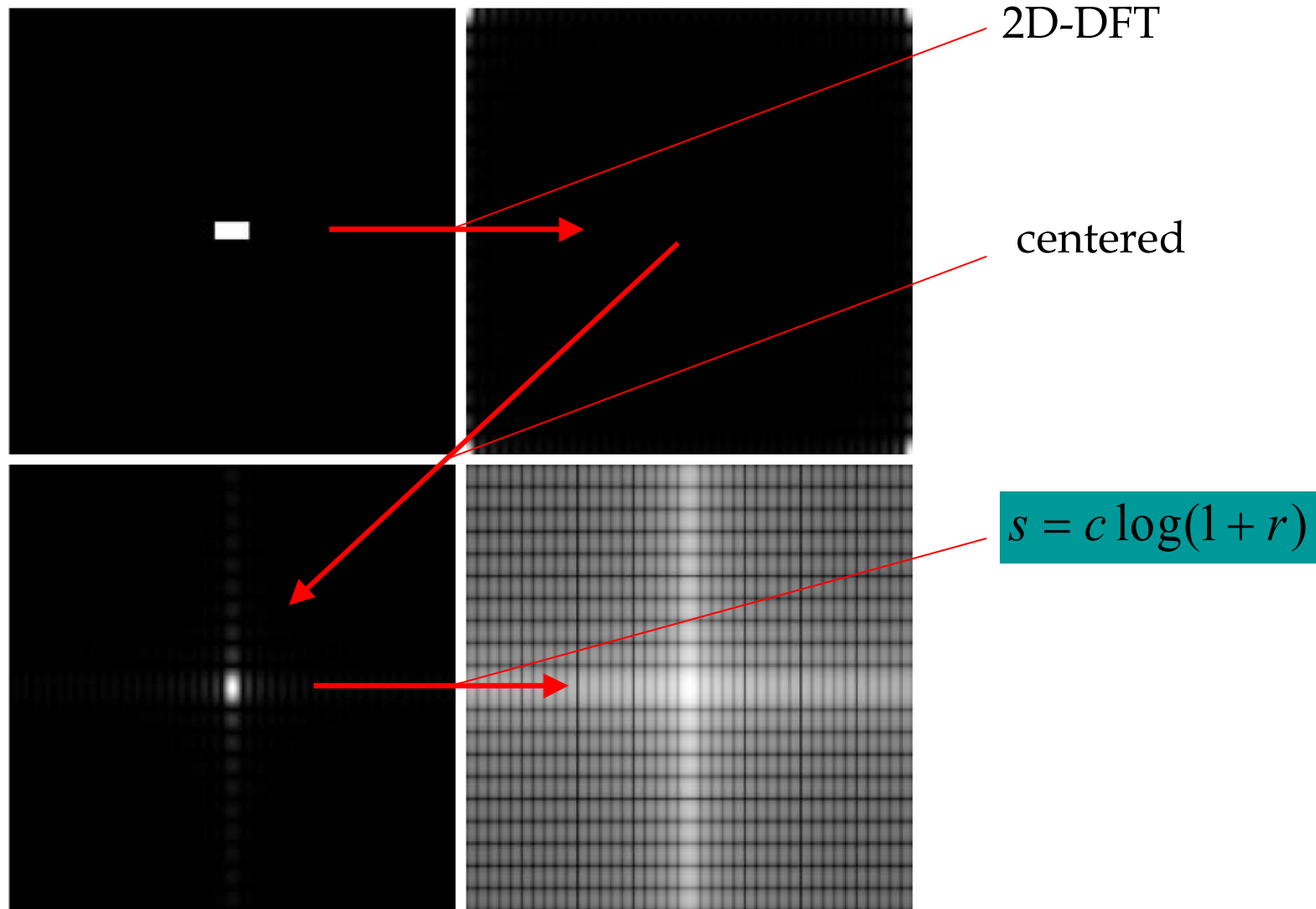
From Prof. Al Bovik

Example:

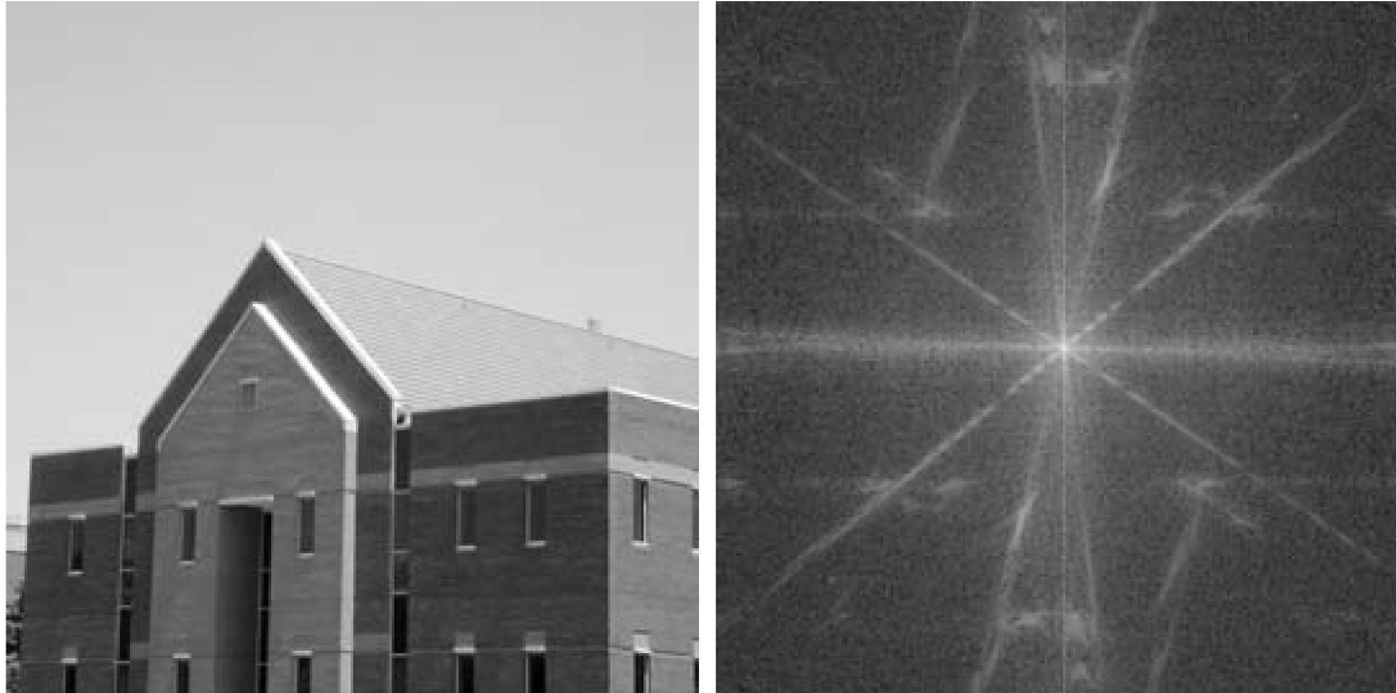


From [Gonzalez
& Woods]

Log-Magnitude Visualization



Apply to Images



2D-DFT \rightarrow centered \rightarrow log intensity transformation

From [Gonzalez & Woods]

How to Display a Fourier Spectrum using MATLAB

The following table is meant to describe the various steps behind displaying the Fourier Spectrum.

MATLAB code	Image Produced
<pre>%Create a black 30x30 image f=zeros(30,30); %With a white rectangle in it. f(5:24,13:17)=1; imshow(f,'InitialMagnification', 'fit')</pre>	 The image produced is a 30x30 pixel grayscale image. It features a solid black background. In the center of the image, there is a white vertical rectangle. The rectangle is 15 pixels wide (from column 13 to 17) and 20 pixels high (from row 5 to 24). The image is displayed using the 'fit' option, meaning it is scaled to fit the window.

%Calculate the DFT.

```
F=fft2(f);
```

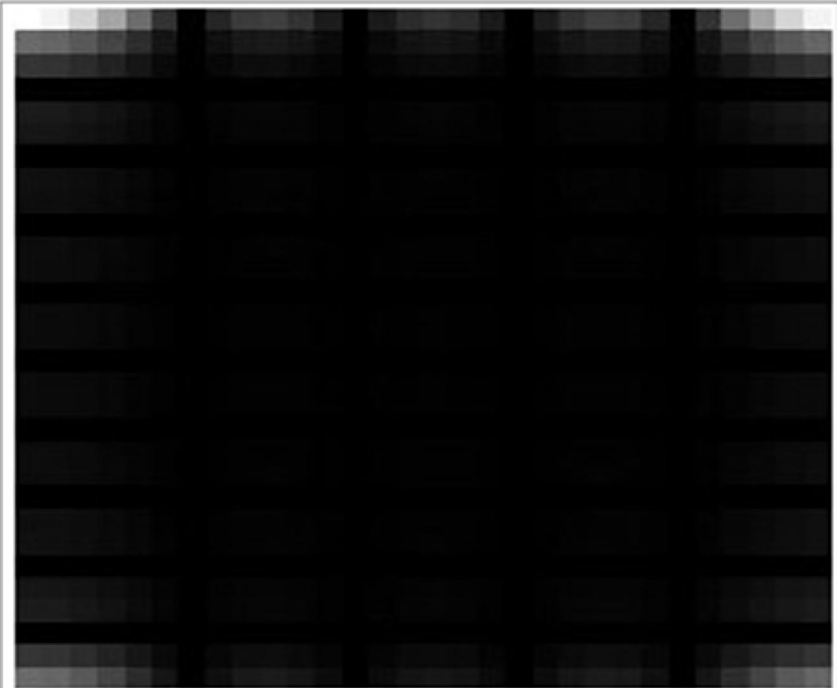
%There are real and imaginary parts to F.

%Use the abs function to compute the
magnitude

%of the combined components.

```
F2=abs(F);
```

```
figure, imshow(F2,[],  
'InitialMagnification','fit')
```



%To create a finer sampling of the Fourier
transform,

%you can add zero padding to f when
computing its DFT

%Also note that we use a power of 2,
2^256

%This is because the FFT -Fast Fourier
Transform -

%is fastest when the image size has many
factors.

```
F=fft2(f, 256, 256);
```

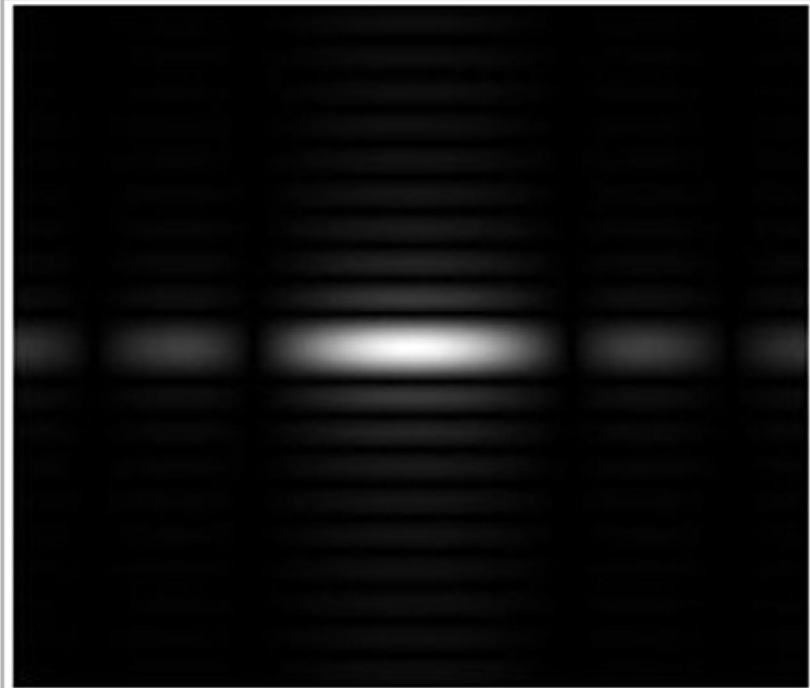
```
F2=abs(F);
```

```
figure, imshow(F2, [])
```



%The zero-frequency coefficient is displayed in the upper left hand corner. To display it in the center,
%you can use the function `fftshift`.
`F2=fftshift(F);`

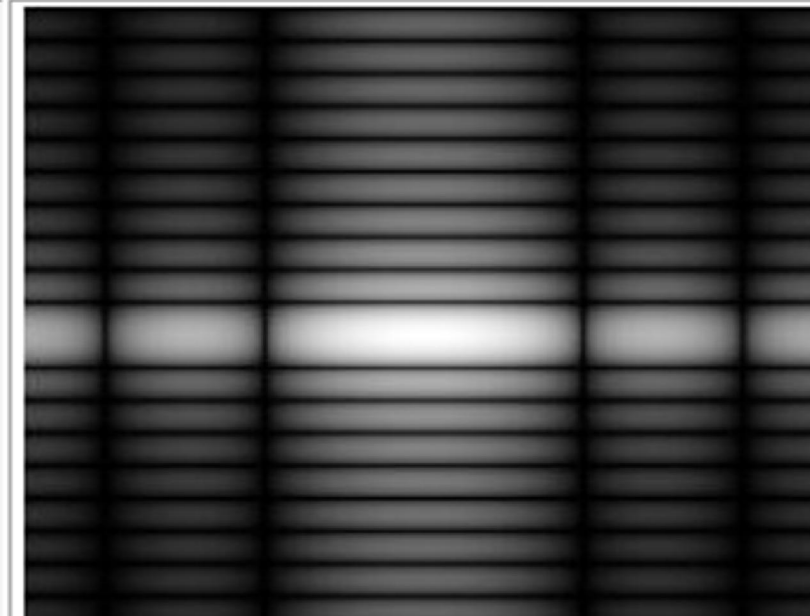
`F2=abs(F2);`
`figure,imshow(F2,[])`



%In Fourier transforms, high peaks are so high they hide details. Reduce contrast with the log function.

`F2=log(1+F2);`

`figure,imshow(F2,[])`



To get the results shown in the last image of the table, you can also combine MATLAB calls as in

:

```
f=zeros(30,30);  
f(5:24,13:17)=1;  
F=fft2(f, 256,256);  
F2=fftshift(F);  
figure,imshow(log(1+abs(F2)),[])
```

Notice in these calls to `imshow`, the second argument is empty square brackets. This maps the minimum value in the image to black and the maximum value in the image to white.

- The following convolution theorem shows an interesting relationship between the spatial domain and frequency domain:

$$f(x,y) * h(x,y) \Leftrightarrow H(u,v) F(u,v)$$

- and, conversely,

$$f(x,y)h(x,y) \Leftrightarrow H(u,v) * G(u,v)$$

where the symbol "*" indicates convolution of the two functions. The important thing to extract out of this is that the multiplication of two Fourier transforms corresponds to the convolution of the associated functions in the spatial domain. This means we can

perform linear spatial filters as a simple component-wise multiply in the frequency domain.

This suggests that we could use Fourier transforms to speed up spatial filters. This only works for large images that are correctly padded, where multiple transformations are applied in the frequency domain before moving back to the spatial domain.

When applying Fourier transforms padding is very important.

Note that, because images are infinitely tiled in the frequency domain, filtering produces wraparound artefacts if you don't zero pad the image to a larger size. The `paddedsiz` function below calculates a correct padding size to avoid this problem. The `paddedsiz` function can also help optimize the performance of the DFT by providing power of 2 padding sizes. See `paddedsiz`'s help header for details on how to do this.

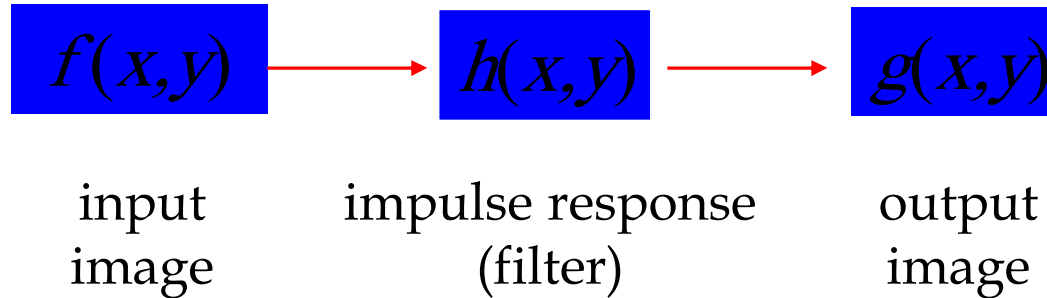
Basic Steps in DFT Filtering

The following summarize the basic steps in DFT Filtering (taken directly from page 121 of *Digital Image Processing Using MATLAB*):

1. Obtain the padding parameters using function padddsize:
`PQ=padddsize(size(f));`
2. Obtain the Fourier transform of the image with padding:
`F=fft2(f, PQ(1), PQ(2));`
3. Generate a filter function, H, the same size as the image
4. Multiply the transformed image by the filter:
`G=H.*F;`
5. Obtain the real part of the inverse FFT of G:
`g=real(ifft2(G));`
6. Crop the top, left rectangle to the original size:
`g=g(1:size(f, 1), 1:size(f, 2));`

2D-DFT (Frequency) Domain Filtering

Convolution Theorem



$$g(x, y) = f(x, y) \otimes h(x, y)$$

Diagram illustrating the transformation between the spatial domain and the frequency domain for the input, filter, and output. For each of the three components, a pair of red arrows connects the spatial domain expression to the frequency domain expression. The leftmost pair connects $f(x,y)$ to $F(u,v)$ via a vertical arrow labeled 'DFT' (downward) and 'IDFT' (upward). The middle pair connects $h(x,y)$ to $H(u,v)$ via a slanted arrow labeled 'DFT' (downward) and 'IDFT' (upward). The rightmost pair connects $g(x,y)$ to $G(u,v)$ via a slanted arrow labeled 'DFT' (downward) and 'IDFT' (upward). Below these transformations, the frequency domain convolution equation is given: $G(u, v) = F(u, v) H(u, v)$.

Frequency Domain Filtering

Frequency domain filtering operation

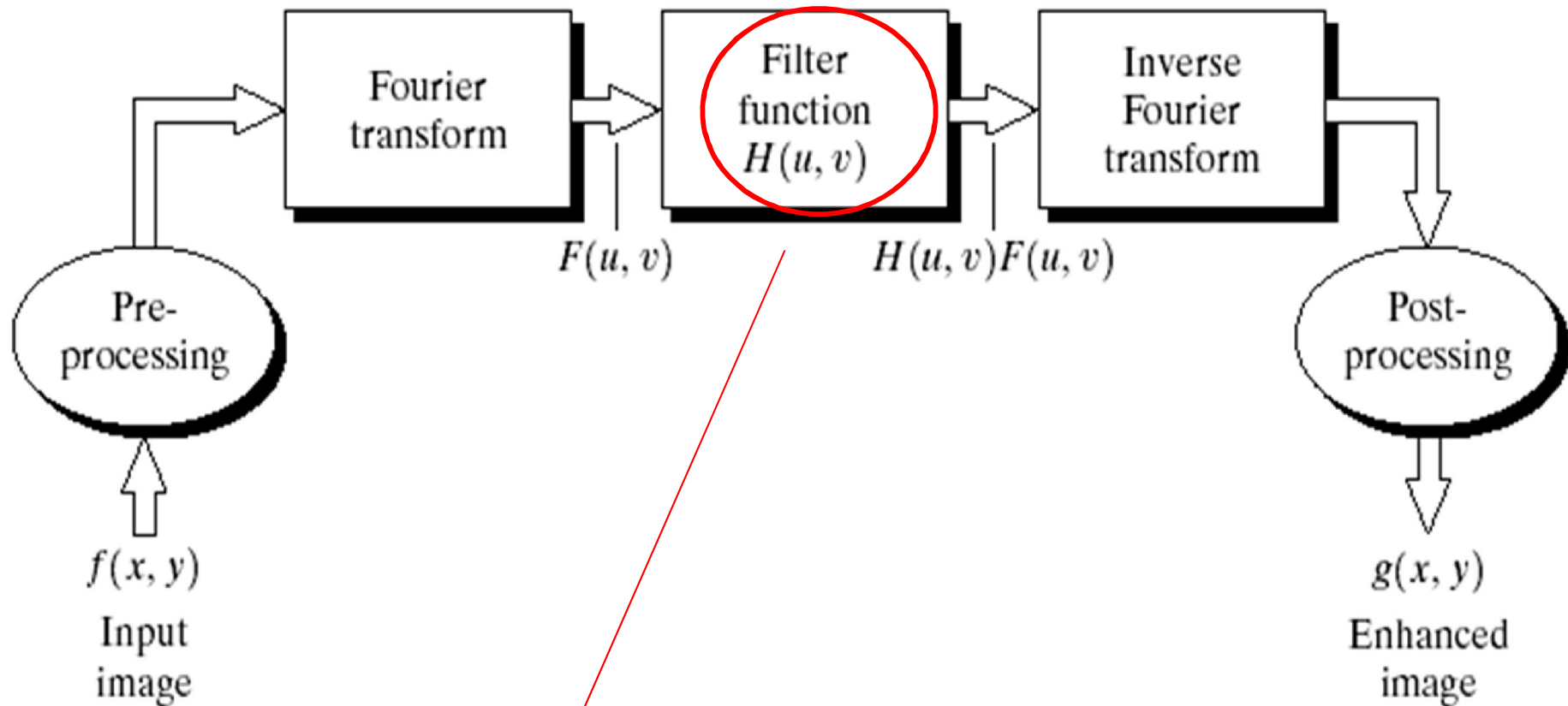


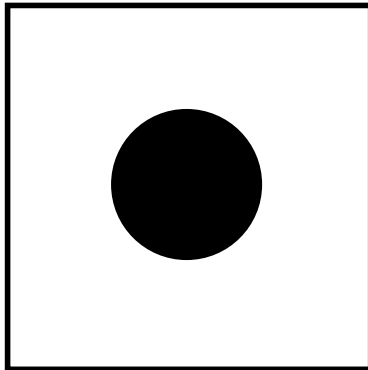
FIGURE 4.5 Basic steps for filtering in the frequency domain.

Filter design: design $H(u, v)$

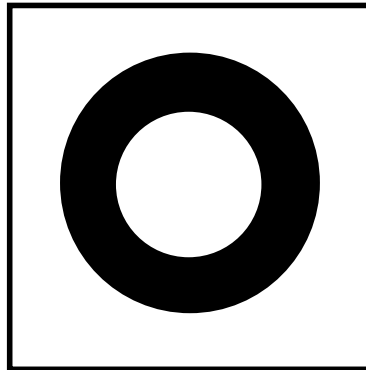
From [Gonzalez & Woods]

2D-DFT Domain Filter Design

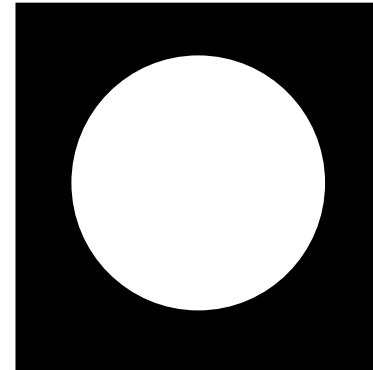
- Ideal lowpass, bandpass and highpass



low-frequency
mask



mid-frequency
mask



high-frequency
mask

Example: Applying the Sobel Filter in the Frequency Domain

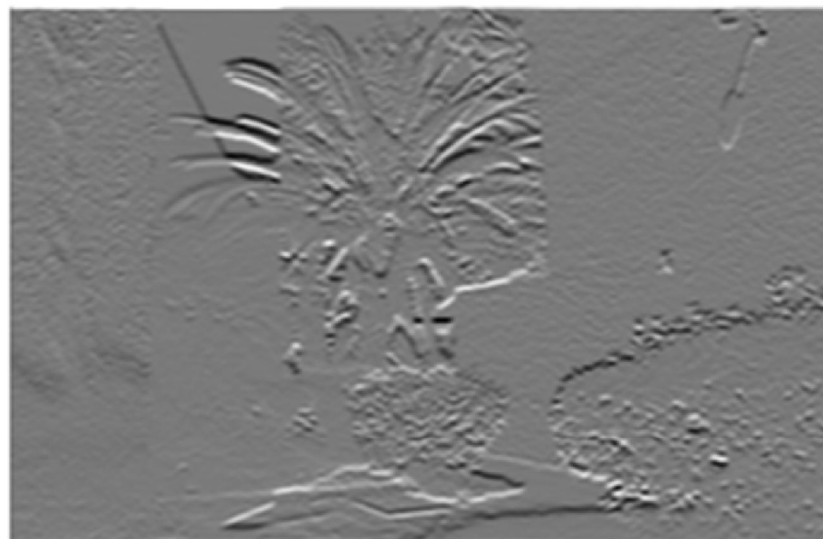
For example, let's apply the Sobel filter to the following picture in both the spatial domain and frequency domain.



Spatial Domain Filtering

```
%Create the Spatial Filtered Image  
f = imread('entry2.png');  
h = fspecial('sobel');  
sfi = imfilter(double(f),h, 0,  
'conv');
```

```
%Display results (show all values)  
figure,imshow(sfi, []);
```



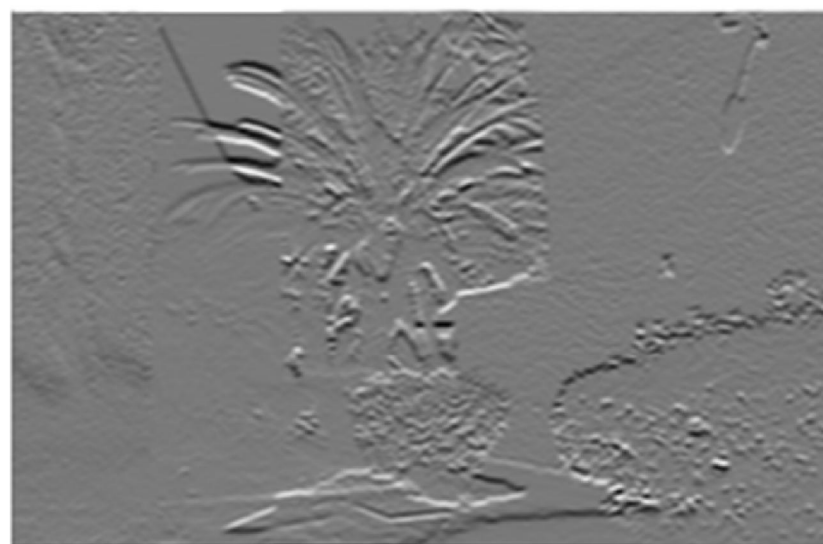
```
%The abs function gets correct magnitude  
%when used on complex numbers
```

```
sfim = abs(sfi);  
figure, imshow(sfim, []);
```

Frequency Domain Filtering

```
%Create the Frequency Filtered Image  
f = imread('entry2.png');  
h = fspecial('sobel');  
PQ = paddedsize(size(f));  
F = fft2(double(f), PQ(1),  
PQ(2));  
H = fft2(double(h), PQ(1),  
PQ(2));  
F_fH = H.*F;  
ffi = ifft2(F_fH);  
ffi = ffi(2:size(f,1)+1,  
2:size(f,2)+1);
```

```
%Display results (show all values)  
figure, imshow(ffi, [])
```

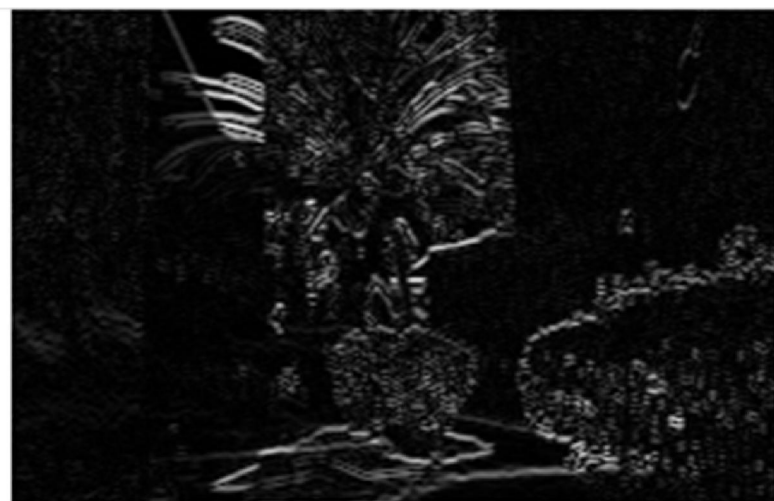


```
%The abs function gets correct magnitude  
%when used on complex numbers
```

```
ffim = abs(ffi);
```



```
%threshold into a binary image  
figure, imshow(sfim >  
0.2*max(sfim(:)));
```



```
%threshold into a binary image  
figure, imshow(ffim >  
0.2*max(ffim(:)));
```



You will notice that both approaches result in a similar looking, if not identical filtered image. You may have to adjust how you crop the image slightly as shown in the

Frequency Domain Specific Filters

As you have already seen, based on the property that multiplying the FFT of two functions from the spatial domain produces the convolution of those functions, you can use Fourier transforms as a fast convolution on large images. Note that on small images it is faster to work in the spatial domain.

However, you can also create filters directly in the frequency domain. There are three commonly discussed filters in the frequency domain:

Lowpass filters, sometimes known as smoothing filters

Highpass filters, sometimes known as sharpening filters

Notch filters, sometimes known as band-stop filters

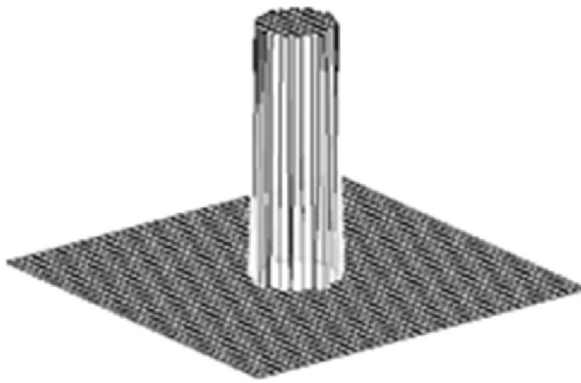
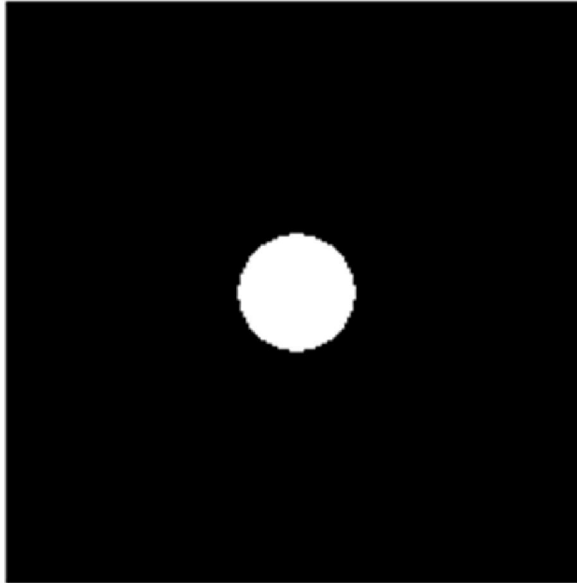
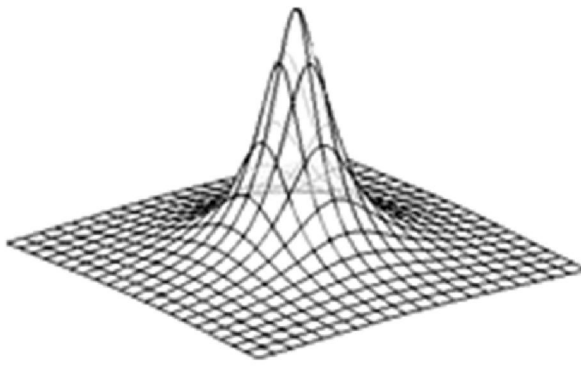
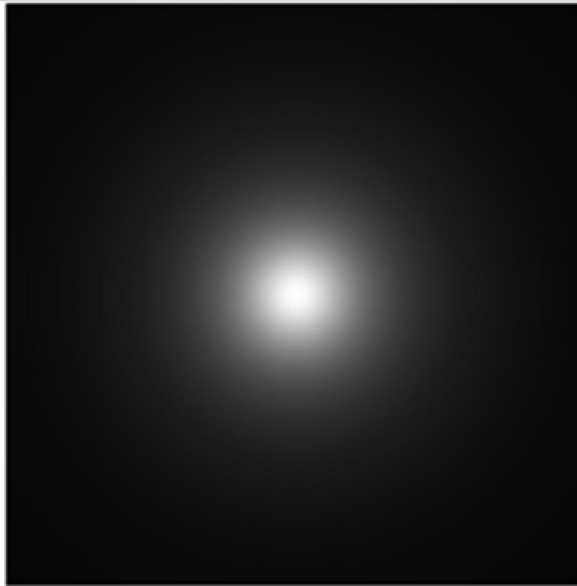
Lowpass filters:

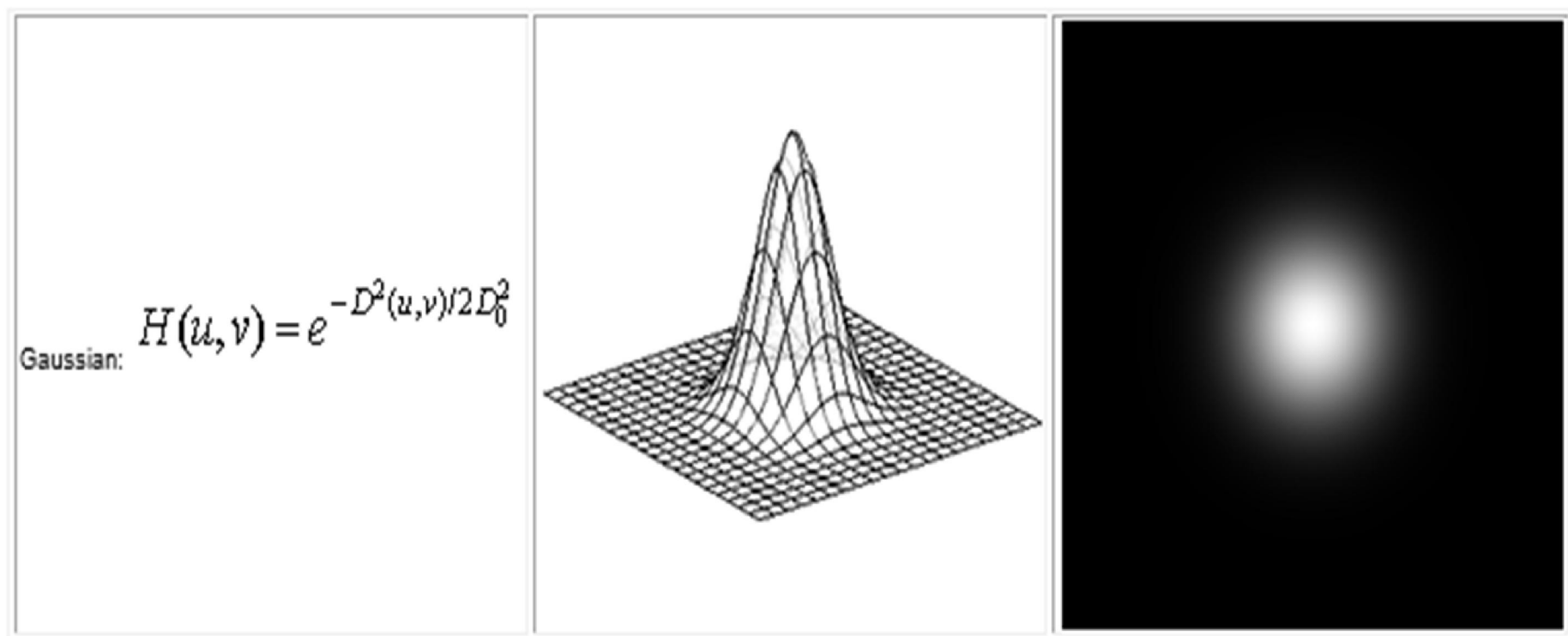
- create a blurred (or smoothed) image
- attenuate the high frequencies and leave the low frequencies of the Fourier transform relatively unchanged

Three main lowpass filters are discussed in *Digital Image Processing Using MATLAB*:

1. ideal lowpass filter (ILPF)
2. Butterworth lowpass filter (BLPF)
3. Gaussian lowpass filter (GLPF)

The corresponding formulas and visual representations of these filters are shown in the table below. In the formulae, D_0 is a specified nonnegative number. $D(u, v)$ is the distance from point (u, v) to the center of the filter.

Lowpass Filter	Mesh	Image
<p>Ideal:</p> $H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$		
<p>Butterworth:</p> $H(u, v) = \frac{1}{1 + [D(u, v) / D_0]^{2n}}$		



To view the MATLAB calls that were used to create the images in the above table, click [on this link](#).

The following is the result of applying a Gaussian lowpass filter on an image.

Original Image



Fourier Spectrum of Image

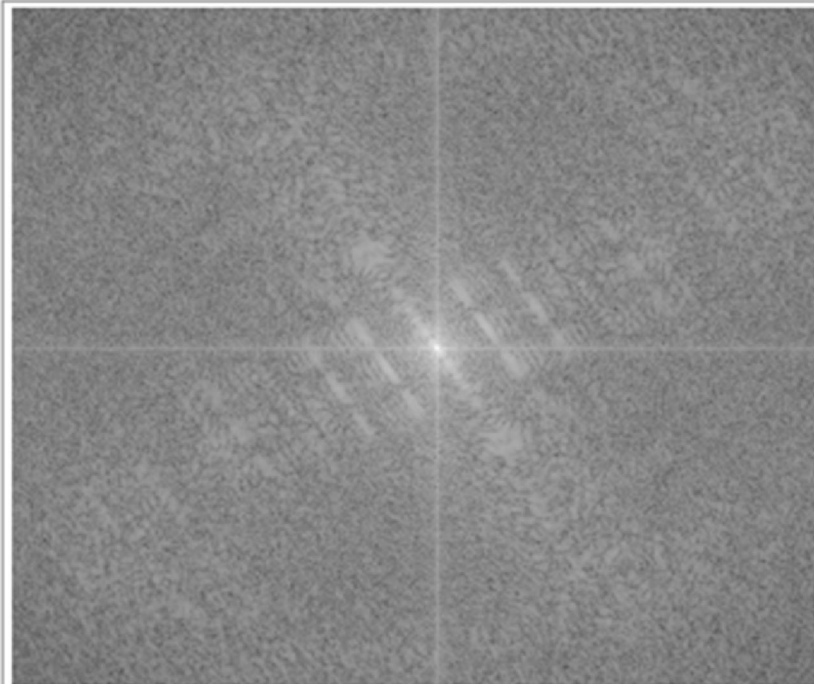
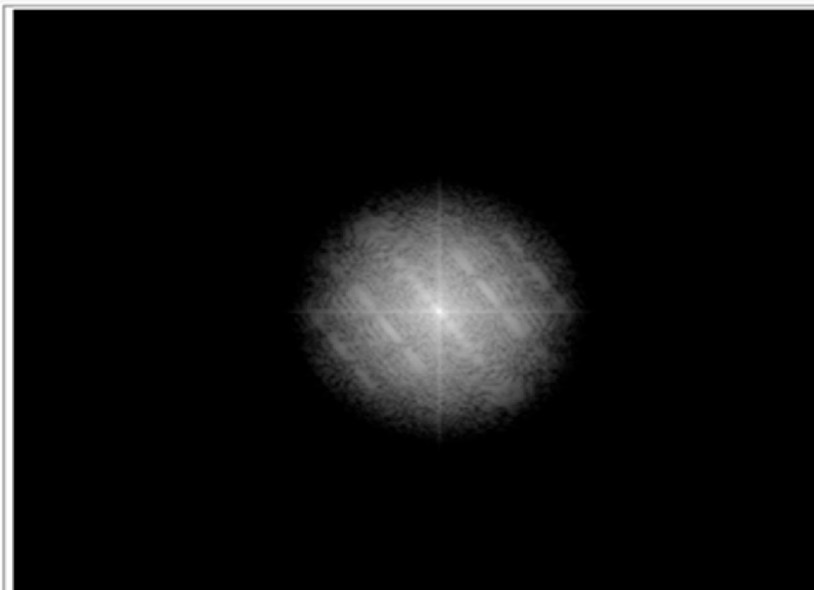


Image with Gaussian lowpass filter



Spectrum of image with Gaussian lowpass filter



Matlab code

```
football=imread('football.jpg');  
%Convert to grayscale  
football=rgb2gray(football);  
imshow(football)  
  
%Determine good padding for Fourier transform  
PQ = paddedsize(size(football));  
  
%Create a Gaussian Lowpass filter 5% the width of the Fourier transform  
D0 = 0.05*PQ(1);  
H = lpfilter('gaussian', PQ(1), PQ(2), D0);  
  
% Calculate the discrete Fourier transform of the image  
F=fft2(double(football),size(H,1),size(H,2));  
  
% Apply the highpass filter to the Fourier spectrum of the image  
LPFS_football = H.*F;  
  
% convert the result to the spacial domain.  
LPF_football=real(ifft2(LPFS_football));  
  
% Crop the image to undo padding  
LPF_football=LPF_football(1:size(football,1), 1:size(football,2));  
  
%Display the blurred image  
figure, imshow(LPF_football, [])
```


%Display the blurred image

```
figure, imshow(LPF_football, [])
```

% Display the Fourier Spectrum

% Move the origin of the transform to the center of the frequency rectangle.

```
Fc=fftshift(F);
```

```
Fcf=fftshift(LPFS_football);
```

% use abs to compute the magnitude and use log to brighten display

```
S1=log(1+abs(Fc));
```

```
S2=log(1+abs(Fcf));
```

```
figure, imshow(S1, [])
```

```
figure, imshow(S2, [])
```

Highpass filters:



sharpen (or shows the edges of) an image

attenuate the low frequencies and leave the high frequencies of the Fourier transform relatively unchanged

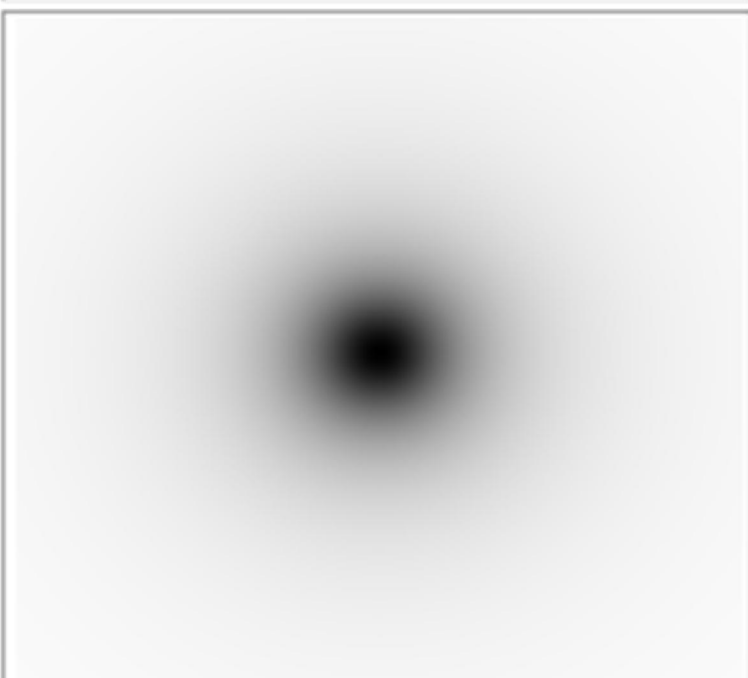
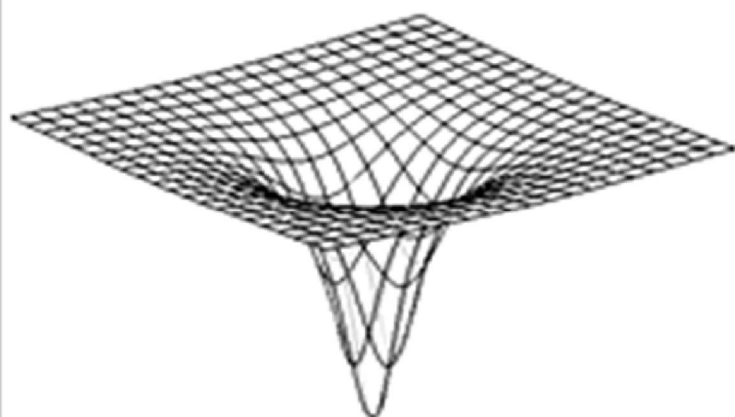
The highpass filter (H_{hp}) is often represented by its relationship to the lowpass filter (H_{lp}):

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

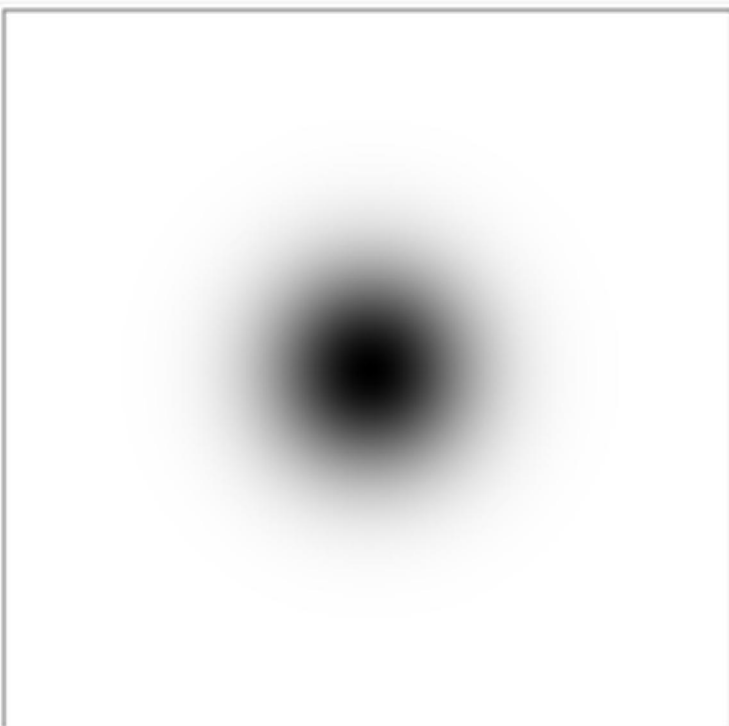
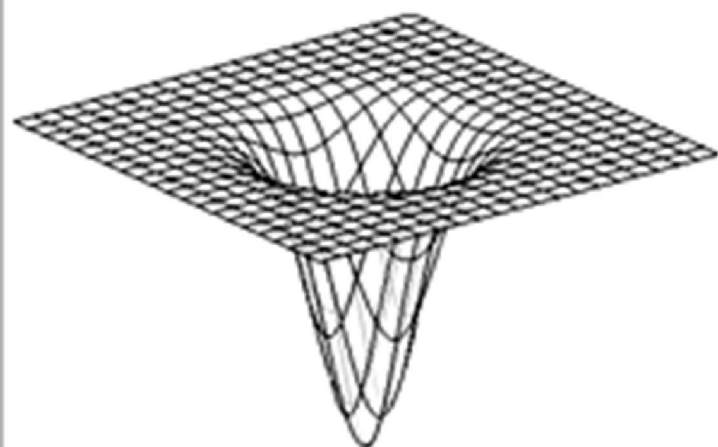
Because highpass filters can be created in relationship to lowpass filters, the following table shows the three corresponding highpass filters by their visual representations:

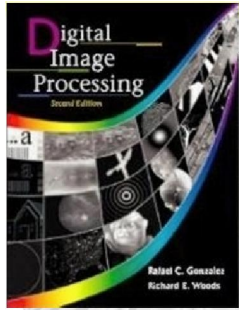
Lowpass Filter	Mesh	Image
Ideal		

Butterworth



Gaussian

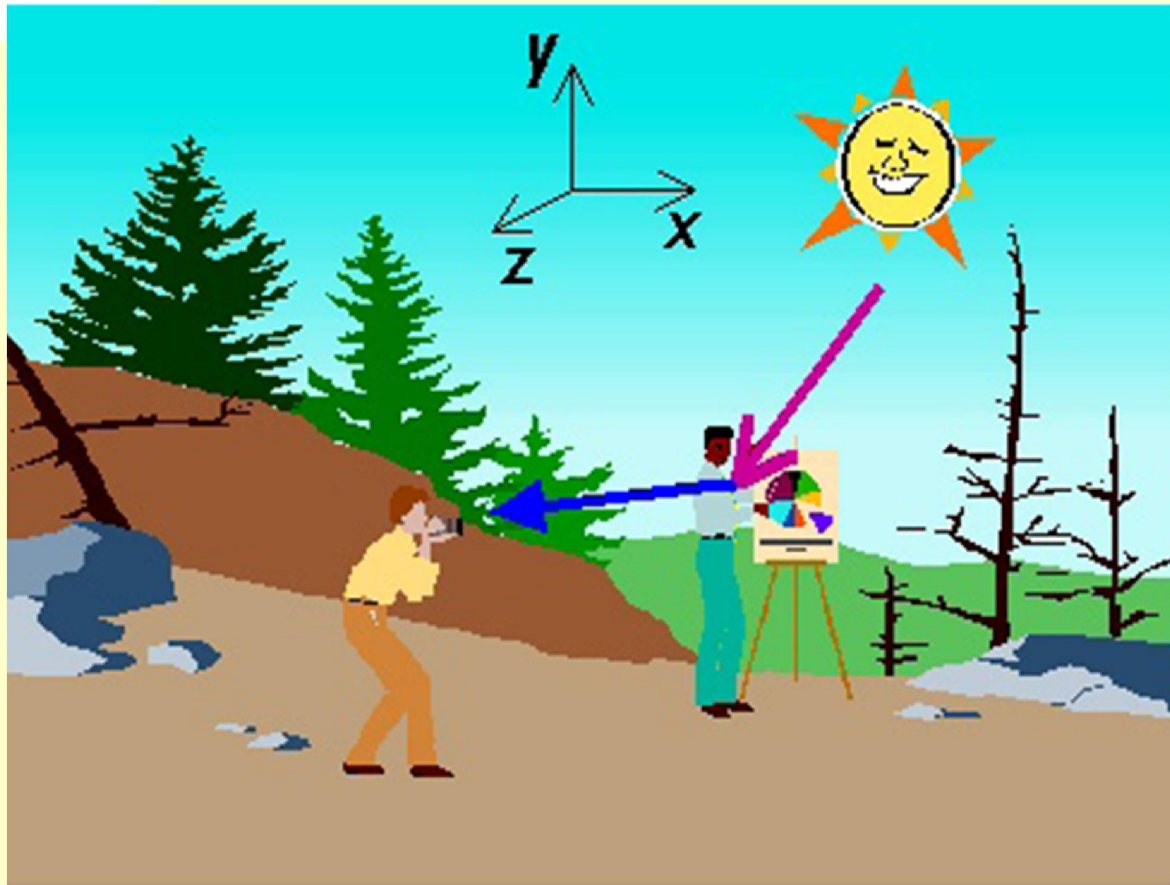




Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

IMAGE FORMATION



The Electromagnetic Spectrum

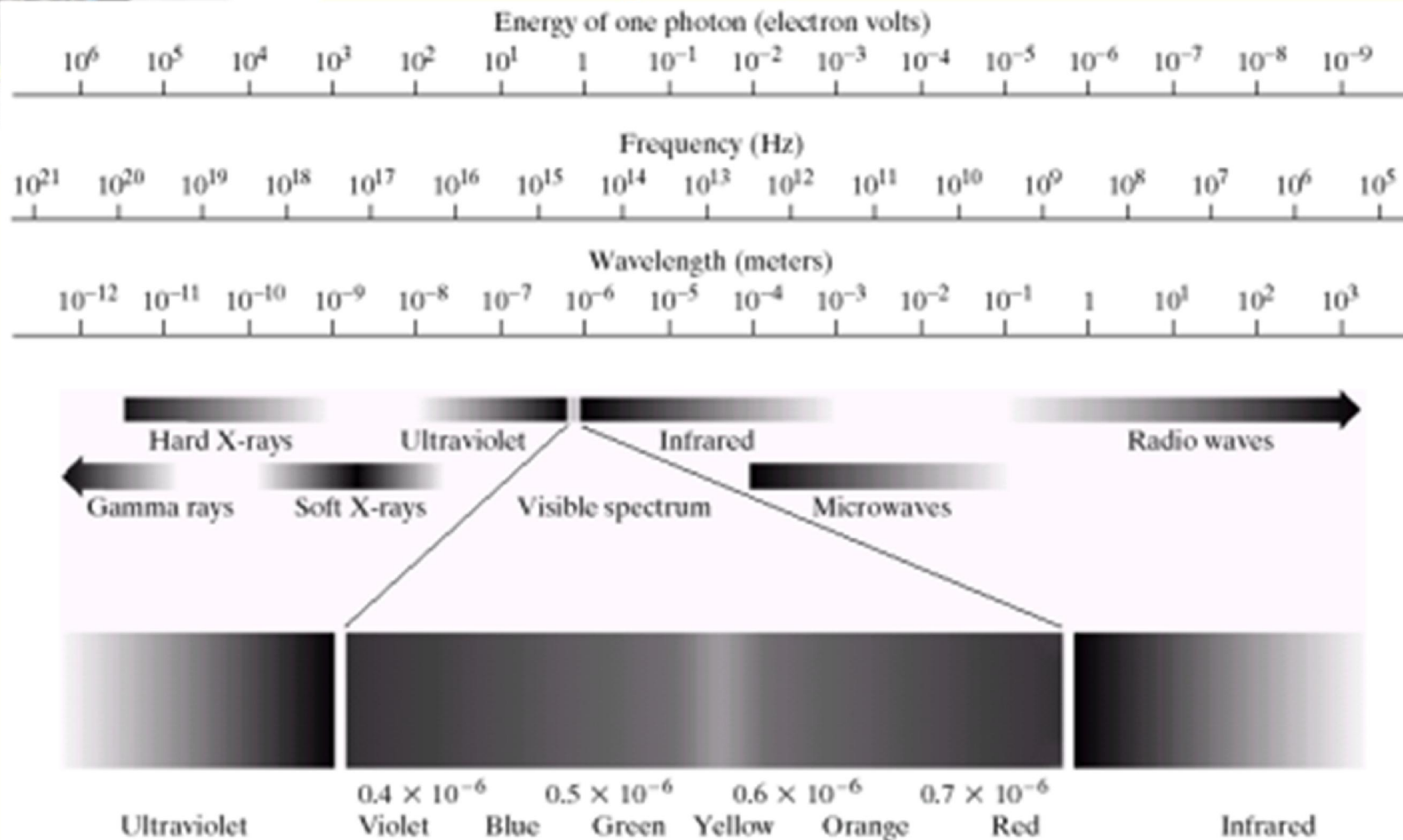
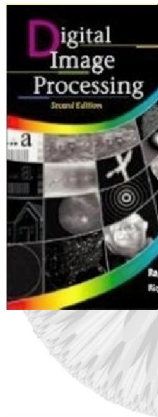


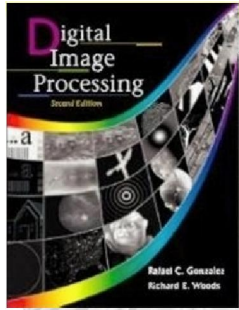
FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum.

© 200



- For natural images we need a light source (λ : wavelength of the source) ^(?).
 - $E(x, y, z, \lambda)$: incident light on a point (x, y, z world coordinates of the point)
- Each point in the scene has a reflectivity function.
 - $r(x, y, z, \lambda)$: reflectivity function
- Light reflects from a point and the reflected light is captured by an imaging device.
 - $c(x, y, z, \lambda) = E(x, y, z, \lambda) \times r(x, y, z, \lambda)$: reflected light.





Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

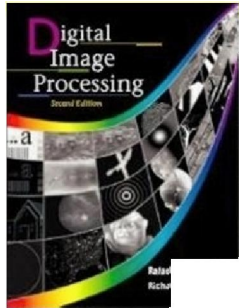


→ $E(x, y, z, \lambda)$

→ $c(x, y, z, \lambda) = E(x, y, z, \lambda) \cdot r(x, y, z, \lambda)$

Camera($c(x, y, z, \lambda)$) =





- The image function $f_c(x', y')$ ($C = R, G, B$) is formed as:

$$f_c(x', y') = \int c_p(x', y', \lambda) V_c(\lambda) d\lambda \quad (2)$$

- It is the result of:

1. Incident light $E(x, y, z, \lambda)$ at the point (x, y, z) in the scene,
2. The reflectivity function $r(x, y, z, \lambda)$ of this point,
3. The formation of the reflected light $c(x, y, z, \lambda) = E(x, y, z, \lambda) \times r(x, y, z, \lambda)$,
4. The **projection** of the reflected light $c(x, y, z, \lambda)$ from the *three* dimensional world coordinates to *two* dimensional camera coordinates which forms $c_p(x', y', \lambda)$,
5. The **sensitivity** function(s) of the camera $V(\lambda)$.

Digital Image Formation

- The **image function** $f_c(x', y')$ is still a function of $x' \in [x'_{min}, x'_{max}]$ and $y' \in [y'_{min}, y'_{max}]$ which vary in a continuum given by the respective intervals.
- The values taken by the image function are real numbers which again vary in a continuum or interval $f_c(x', y') \in [f_{min}, f_{max}]$.
- Digital computers cannot process parameters/functions that vary in a continuum.
- We have to *discretize*:

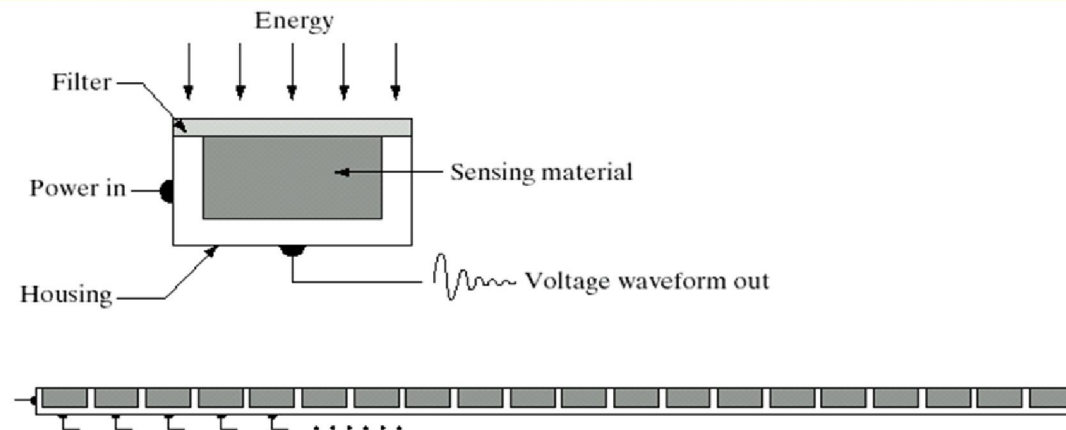
1. $x', y' \Rightarrow x'_i, y'_j \quad (i = 0, \dots, N - 1, j = 0, \dots, M - 1)$:
2. $f_c(x'_i, y'_j) \Rightarrow \hat{f}_c(x'_i, y'_j)$: Quantization.

54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

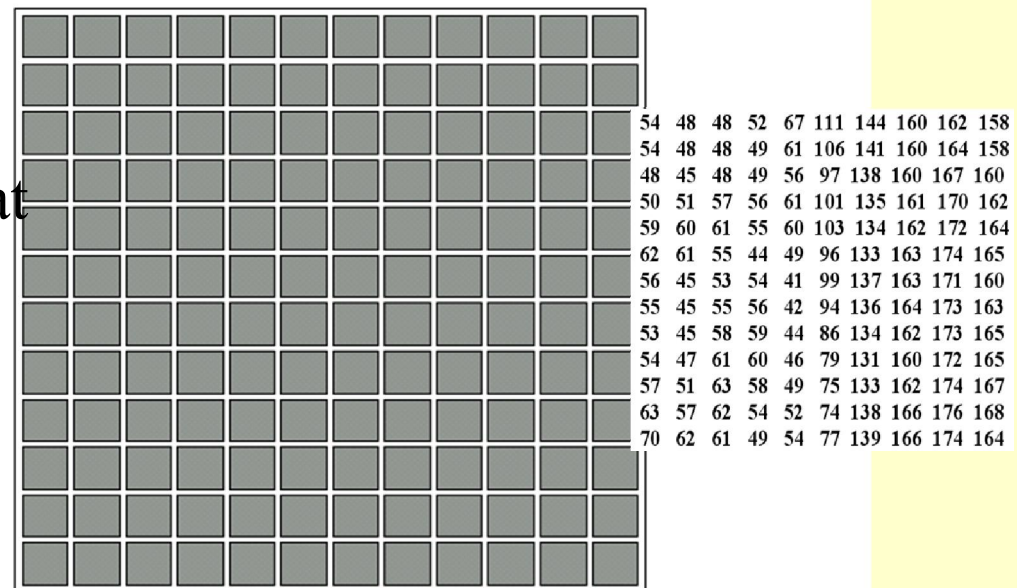
When x, y , and the amplitude values of f are all **finite**,
discrete quantities, we call the image a **digital image**.

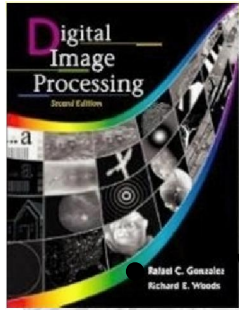
Image Sensing and Acquisition

FIGURE 2.12
 (a) Single imaging sensor.
 (b) Line sensor.
 (c) Array sensor.



A digital image is nothing more than data—numbers indicating variations of red, green, and blue at a particular location on a grid of pixels.





Introduction

What is Digital Image Processing?

Digital Image

— A two-dimensional function $f(x, y)$ x and y are spatial coordinates

The amplitude of f is called **intensity** or **gray level** at the point (x, y)

Digital Image Processing

— Process digital images by means of computer, it covers low-, mid-, and high-level processes

low-level: inputs and outputs are images

mid-level: outputs are attributes extracted from input images

high-level: an ensemble of recognition of individual objects

Pixel

— The elements of a digital image

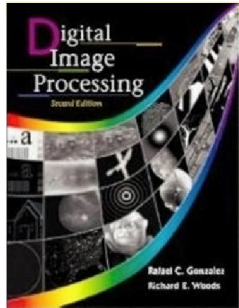
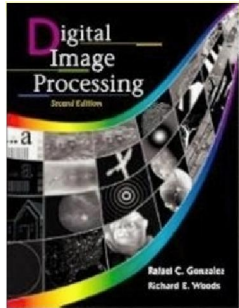


image definition

- An image may be defined as a two – dimensional function, $f(x, y)$, where x and y are spatial (plane) coordinates, and the amplitude of f at any pair coordinates (x, y) is called the intensity or gray level of the image at that point.
- When x , y , and the amplitude values of f are all finite, discrete quantities, we call the image a digital image.



Digital image representation

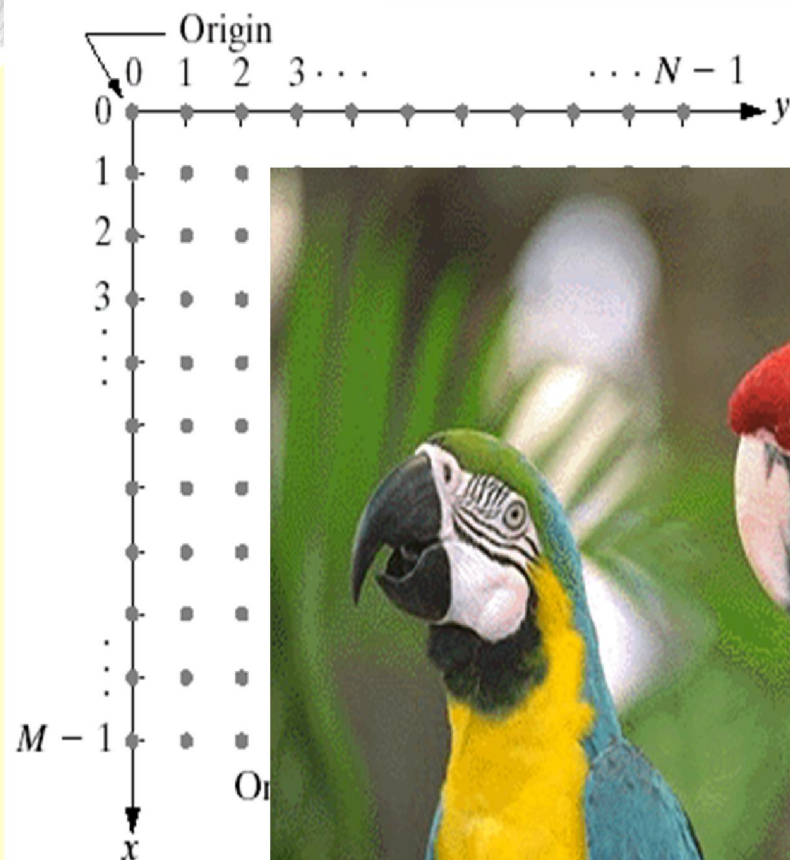


FIGURE 2.18
Coordinate convention used in this book to represent digital images.



Blue

Green

Red

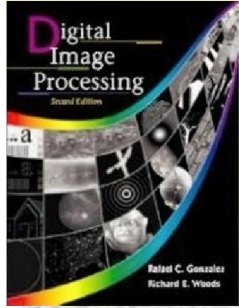


Images as Matrices

- An image matrix ($N \times M$):

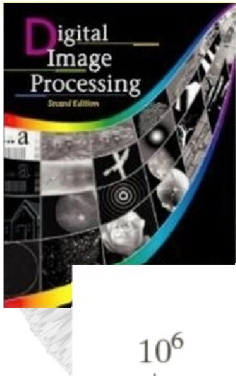
$$\mathbf{A} = \left[\begin{array}{cccc} A(0,0) & A(0,1) & A(0,2) & \dots A(0,M-1) \\ A(1,0) & A(1,1) & A(1,2) & \dots A(1,M-1) \\ \vdots & & & \\ A(N-1,0) & A(N-1,1) & A(N-1,2) & \dots A(N-1,M-1) \end{array} \right] \left. \vphantom{\begin{array}{c} A(0,0) \\ A(1,0) \\ \vdots \\ A(N-1,0) \end{array}} \right\} N \text{ rows } \textcircled{?}$$

- $A(i,j) \in \{0,1,\dots,255\}$.
- $A(i,j)$:
 - “Matrix case:” The matrix element (i,j) with value $A(i,j)$.
 - “Image case:” The pixel (i,j) with value $A(i,j)$.
 - Will use both terminologies.



Sources for Images

- Electromagnetic (EM) energy spectrum
 - Acoustic
 - Ultrasonic
 - Electronic
 - Synthetic images produced by computer



Electromagnetic (EM) energy spectrum

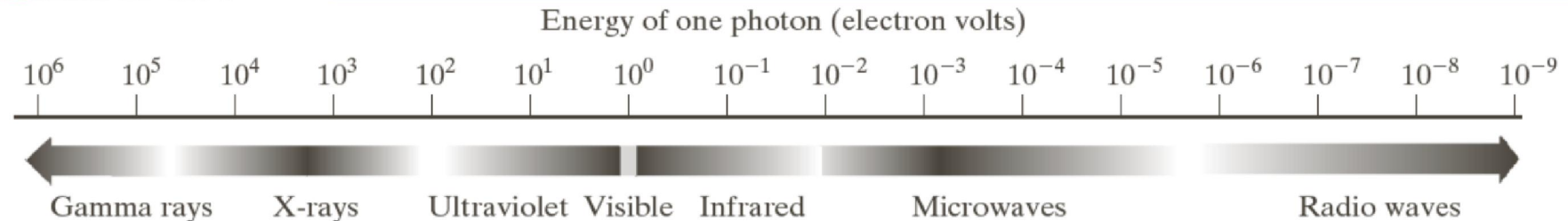


FIGURE 1.5 The electromagnetic spectrum arranged according to energy per photon.

Major fields in which digital image processing is widely used

Gamma-ray imaging: nuclear medicine and astronomical observations

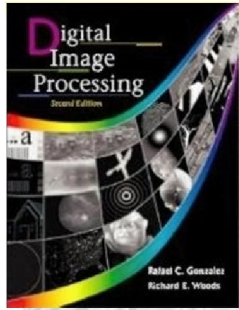
X-rays: medical diagnostics (**X-rays of body**), industry, and astronomy, etc.

Ultraviolet: lithography, industrial inspection, microscopy, lasers, biological imaging, and astronomical observations

Visible and infrared bands: light microscopy, astronomy, remote sensing, industry, and law enforcement

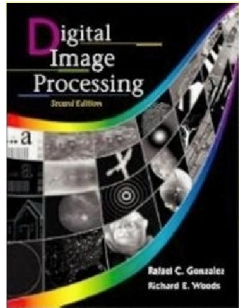
Microwave band: Radar imaging

Radio band: medicine (such as **MRI**) and astronomy



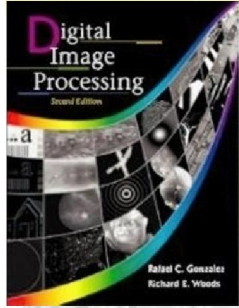
Types of Computerized Processes

- 1-Low Level Process
- 2-Mid Level Process
- 3-High Level Process
- =====
- 1-low Level Process involves primitive operations, such as image processing to reduce noise, contrast, enhancement and image sharpening. In this level, both its input and output are digital images



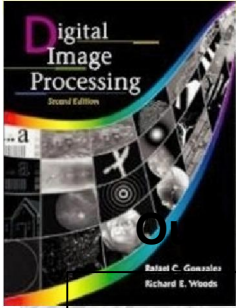
2-Mid Level Process

- Involves tasks such as ,**segmentation** (partitioning an image into regions or objects), description these objects to reduce them to a form suitable to computer, and **classification (recognition) of individual objects**. The inputs are digital images and the outputs are attributes extracted from those images (i.e, edges, contours, and the identity of individual objects).



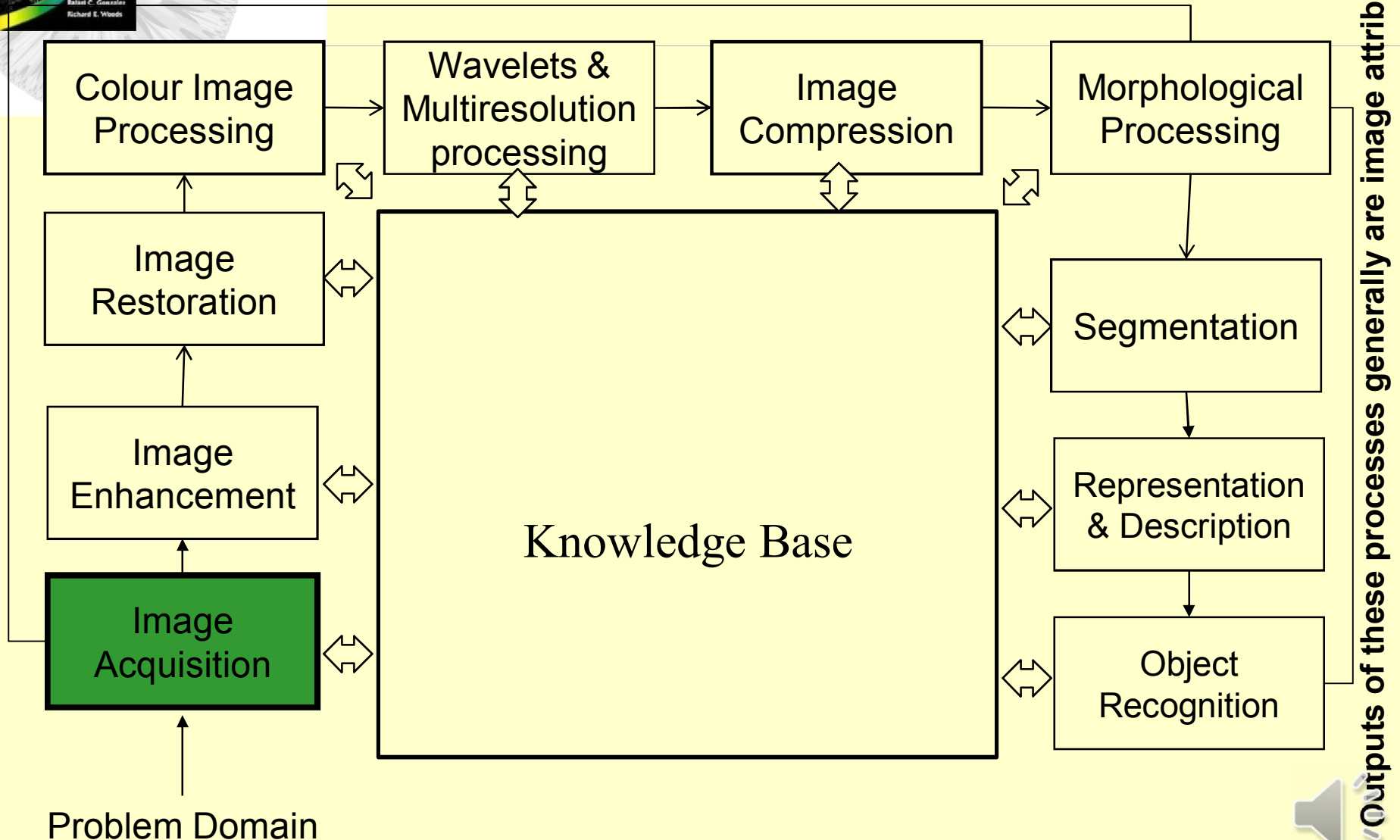
3-High Level Process

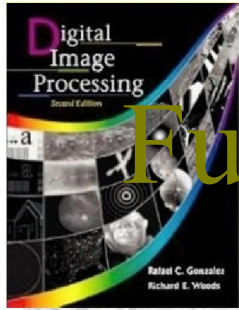
- Involves making sense of a recognized objects as in image analysis , for example : if a digital image contains a number of objects , a program may analyzed the image and extract the objects.
- So the digital image process encompasses processes whose inputs and outputs are images and in addition, encompasses processes that extract attributes from images up to and including recognition of individual objects.



Fundamental Steps in Digital Image Processing:

Outputs of these processes generally are images





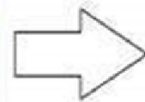
Fundamental Steps in DIP:

Step 1: Image Acquisition

The image is captured by a sensor (eg. **Camera**), and digitized if the output of the camera or sensor is not already in digital form, using **Analogue-to-Digital** convertor



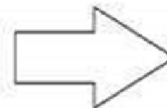
3d world around us



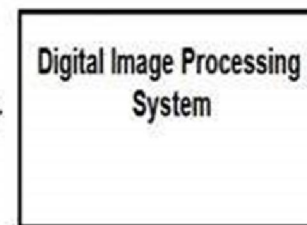
captured
by



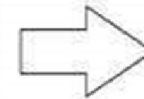
a camera



and sent to



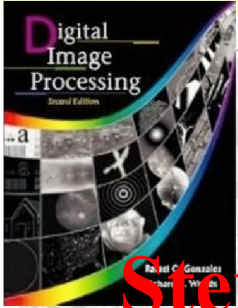
a particular system to
focus on a water drop,



that's gives its
ouput as an



Processed image



Cont. Fundamental Steps in DIP:

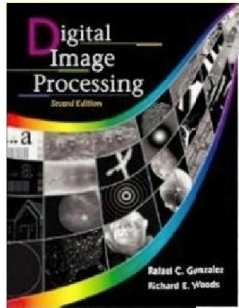
Step 2: Image Enhancement

The process of manipulating an image so that the result is more suitable than the original for specific applications.

The idea behind enhancement techniques is to bring out details that are hidden, or simple to highlight certain features of interest in an image.

- Filtering with morphological operators.
- Histogram equalization.
- Noise removal using a Wiener filter.
- Linear contrast adjustment.
- Median filtering.
- Unsharp mask filtering.



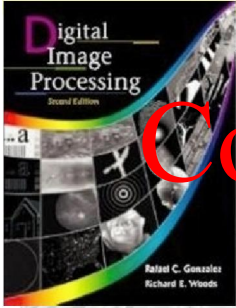


Dioital Image Processing 2nd ed

imageprocessingbook.com

Image Enhancement

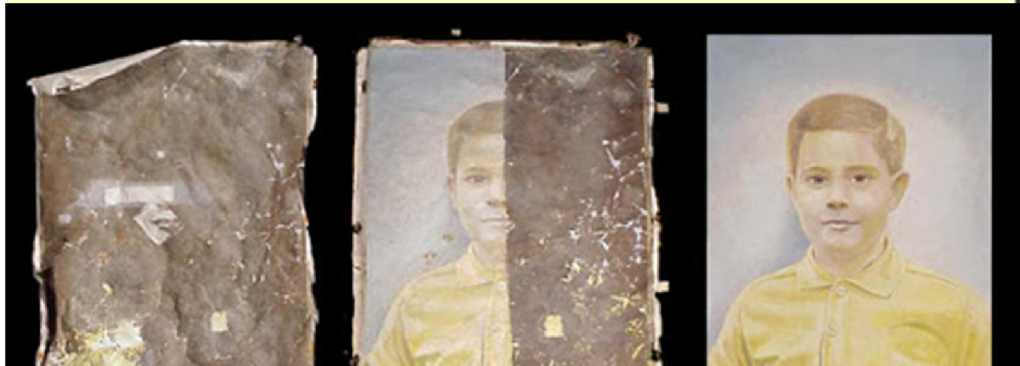




Cont. Fundamental Steps in DIP:

Step 3: Image Restoration

- Improving the appearance of an image
- Tend to be mathematical or probabilistic models. Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result.



What is Image Restoration?

dm

- Image restoration attempts to restore images that have been degraded
 - ✓ Identify the degradation process and attempt to reverse it.
 - ✓ Almost Similar to image enhancement, but more objective.

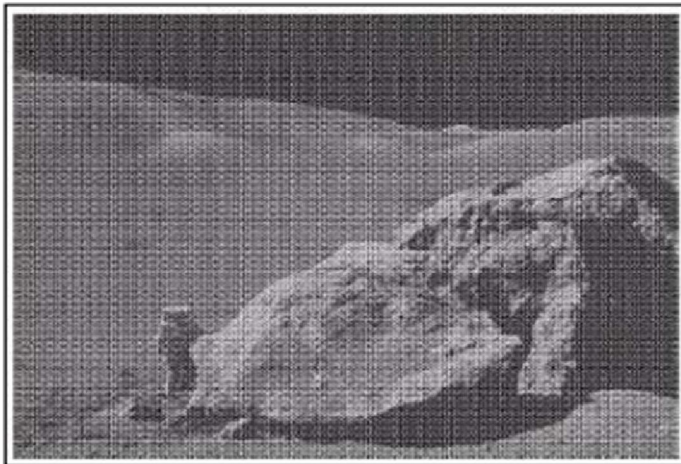
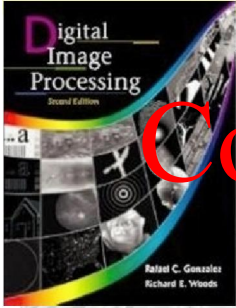


Fig: Degraded image



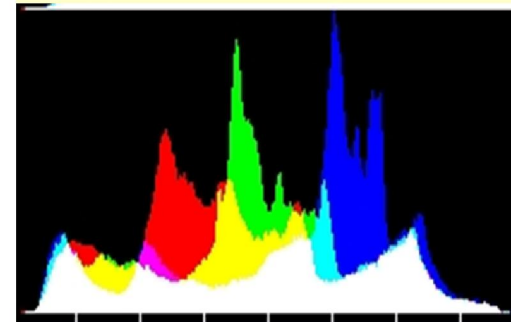
Fig: Restored image

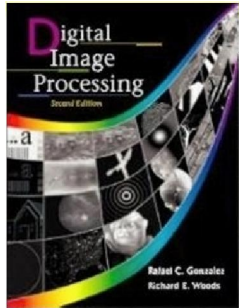


Cont. Fundamental Steps in DIP:

Step 4: Colour Image Processing

Use the colour of the image to extract features of interest in an image





Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

Color to grey and negative

origin

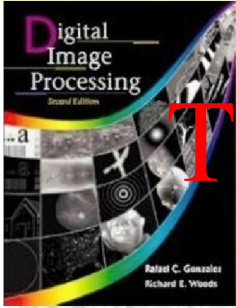


gray



negative





The DIP steps can be summarized as:

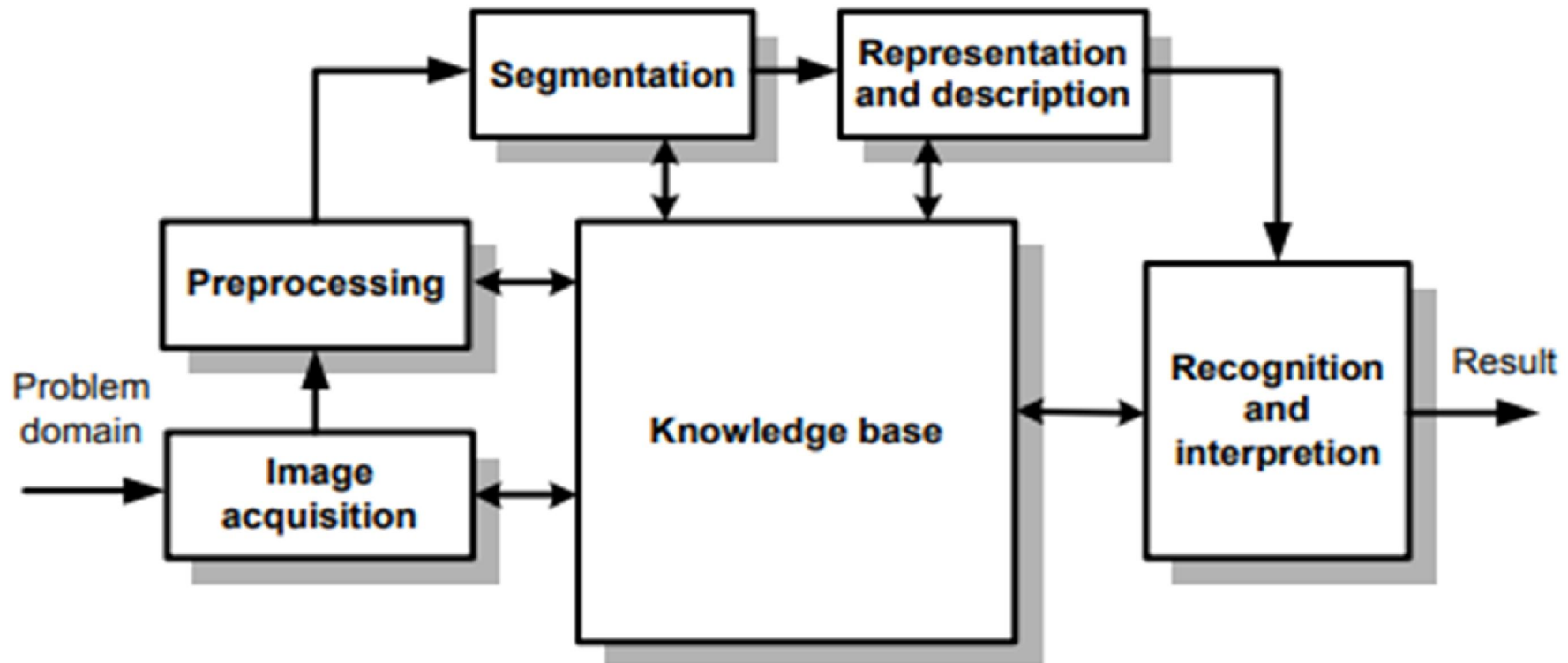


Fig 1. Fundamental steps in digital image processing



Elements of digital image processing systems:

- The basic operations performed in a digital image processing systems include (1) acquisition, (2) storage, (3) processing, (4) communication and (5) display.

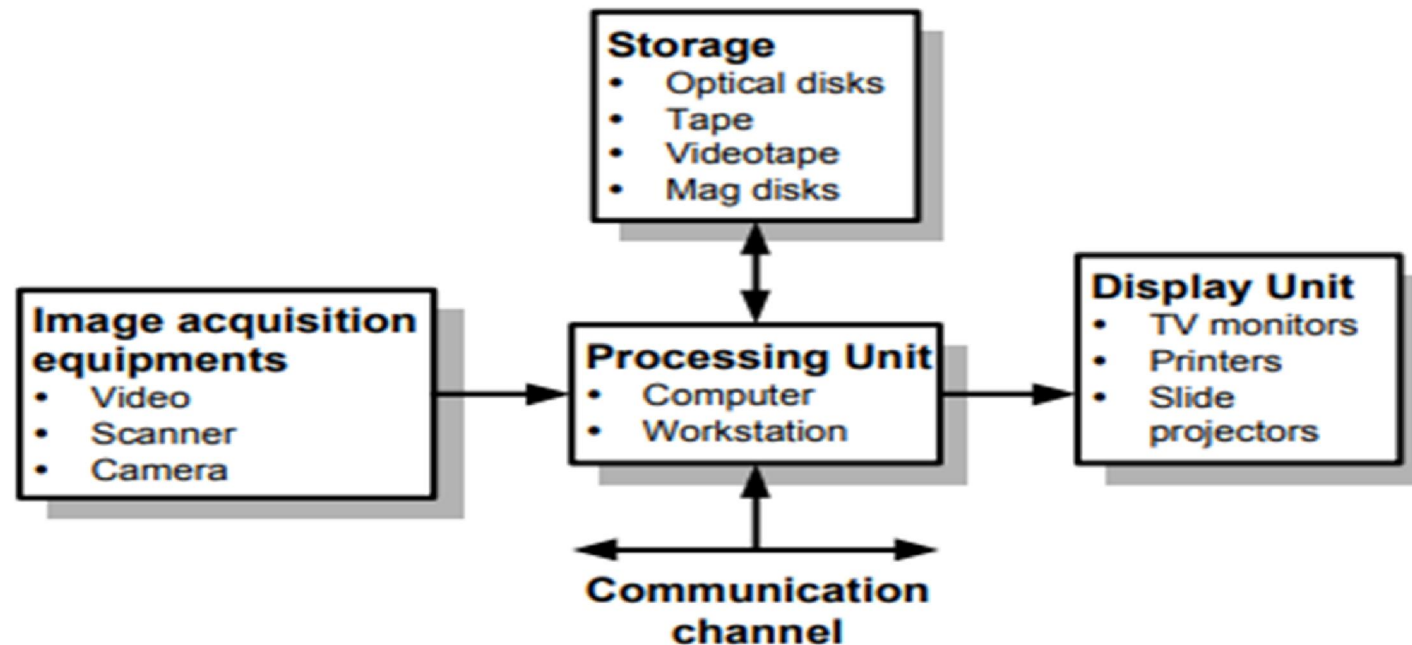
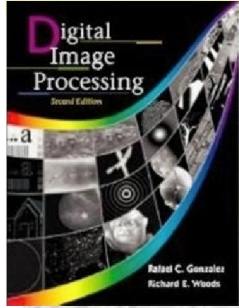
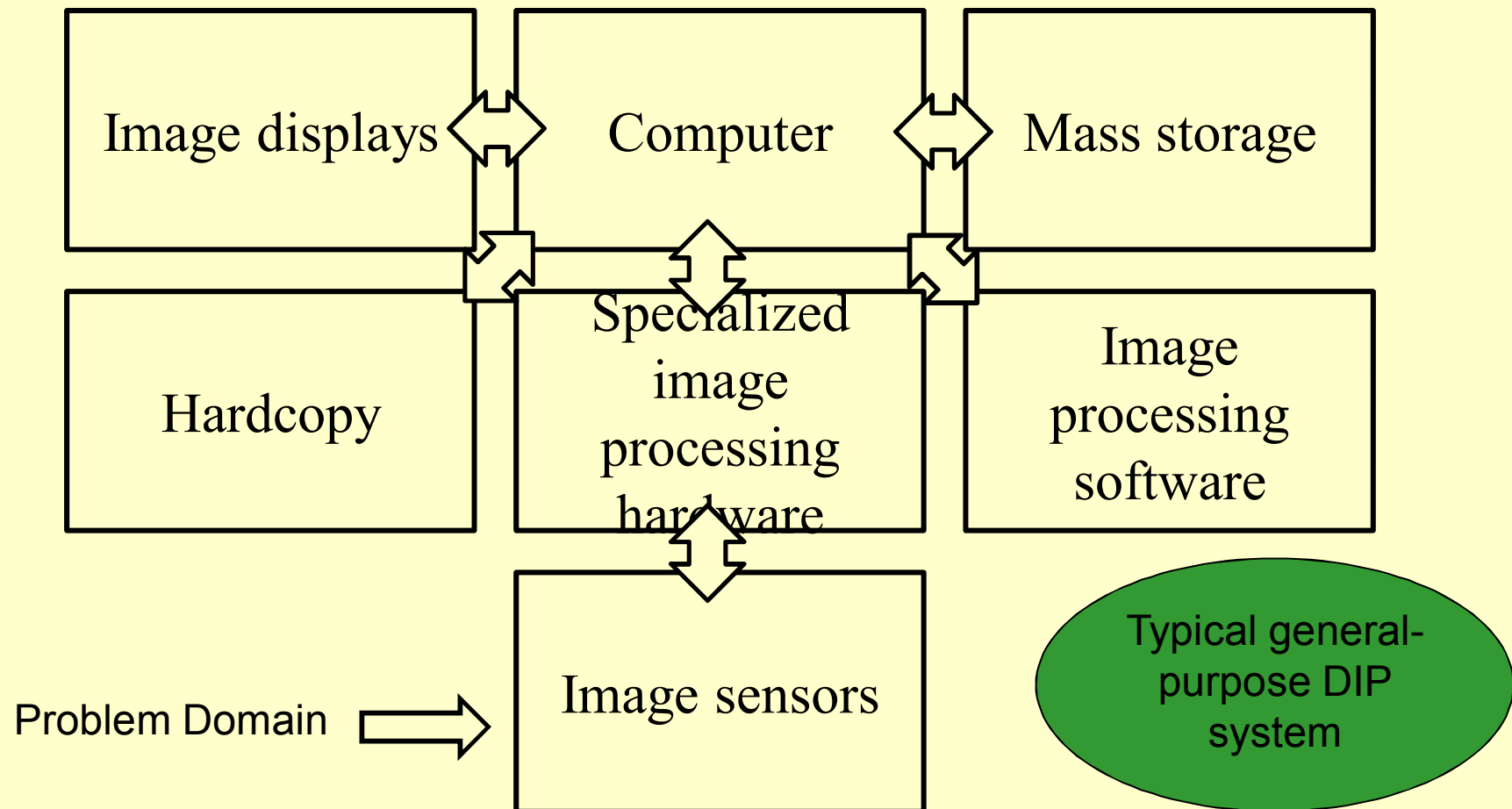


Fig 2. Basic fundamental elements of an image processing system



Components of an Image Processing System

Network



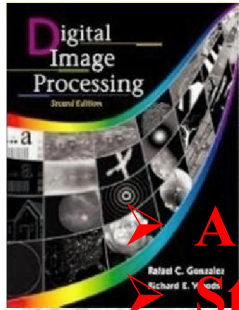


Image Statistics

الخصائص الإحصائية للصورة الرقمية

مع التطبيق

- Arithmetic Mean,
- Standard Deviation,
- and Variance

54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164

..com

- Useful statistical features of an image **are its arithmetic mean,**
- **standard deviation, and variance.** These are well known mathematical constructs that, when applied to a digital image, can reveal important information.
- **The arithmetic mean is the image's average value.**
- The standard deviation is a measure of the frequency distribution, or range of pixel values, of an image. If an image is supposed to be uniform throughout, the standard deviation should be small. *A small standard deviation indicates that the pixel intensities do not stray very far from the mean; a large value indicates a greater range.*
- The standard deviation is the square root of the variance.
- **The variance** is a measure of how spread out a distribution is. It is computed as the average squared deviation from its mean

- The **sample mean** (m_A) of an image A ($N \times M$):

$$m_A = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} A(i,j)}{NM}$$

- The **sample variance** (σ_A^2) of A :

$$\sigma_A^2 = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A(i,j) - m_A)^2}{NM}$$

Simple Image Statistics - Sample Mean and Sample Variance

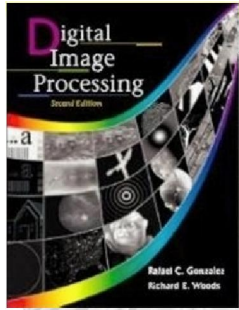
- The **sample mean** (m_A) of an image A ($N \times M$):

$$m_A = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} A(i, j)}{NM} \quad (1)$$

- The **sample variance** (σ_A^2) of A:

$$\sigma_A^2 = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (A(i, j) - m_A)^2}{NM} \quad (2)$$

- The **sample standard deviation**, $\sigma_A = \sqrt{\sigma_A^2}$.



filters

1	1	1
1	1	1
1	1	1

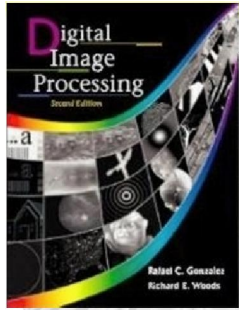
-1	-2	-1
0	0	0
1	2	1

3 x 3

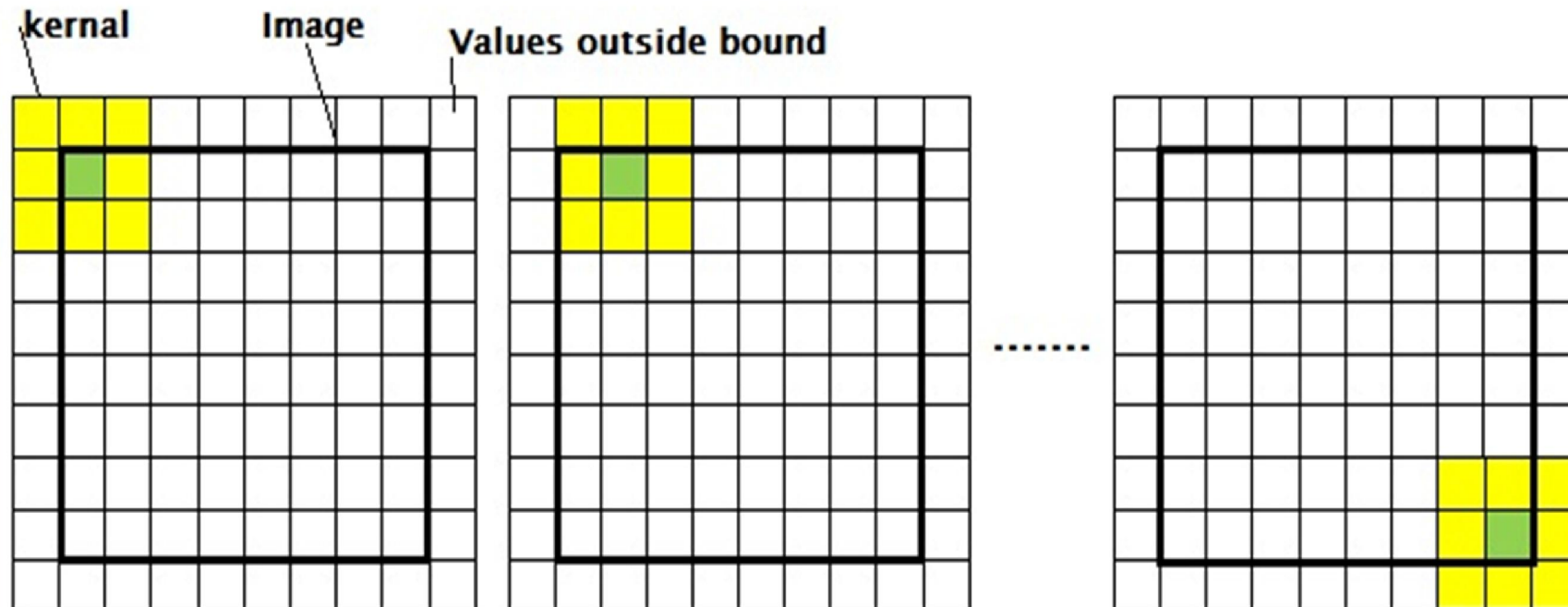
5 x 5

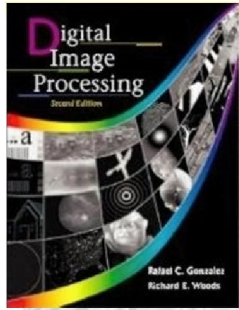
7 x 7

Masks ,
windows



Using filters





Filter and image

What value should these
outside pixels have?

$?^8$ $?^1$ $?^6$

17	24	1^3	8^5	15^7
23	5	7^4	14^9	16^2
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

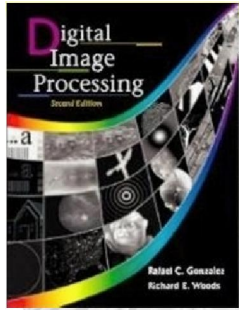
Center of kernel

Outside pixels are
assumed to be 0.

0^8 0^1 0^6

17	24	1^3	8^5	15^7
23	5	7^4	14^9	16^2
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

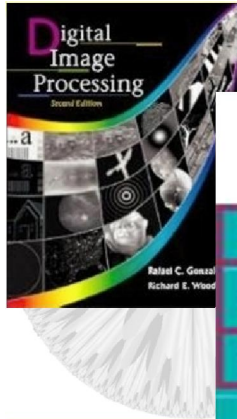
Center of kernel



- Linear filtering can be used to **smooth, blur, sharpen, or find the edges of an image**. The following four images are meant to demonstrate what spatial filtering can do. The **original image is shown in the upper left-hand corner**.

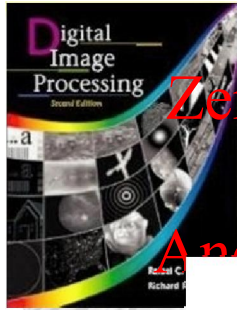


- Smooth blur sharpen find the edges



Zero Padding

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	0
0	251	244	255	246	7	255	254	255	255	242	0	255	248	255	255	254	0
0	255	255	240	183	0	231	247	255	244	255	0	168	255	241	255	252	0
0	254	255	250	12	87	2	255	240	255	0	73	7	253	255	239	255	0
0	242	247	255	0	94	0	254	254	241	0	95	0	255	255	248	247	0
0	255	255	250	69	87	83	2	255	6	107	79	74	249	245	255	250	0
0	255	243	255	156	95	88	0	255	12	58	77	201	239	255	251	253	0
0	255	248	255	255	0	74	106	0	85	84	8	250	255	255	242	255	0
0	250	255	249	255	255	0	78	4	89	49	251	252	255	255	241	255	0
0	255	246	255	252	255	161	57	30	44	150	249	255	255	239	255	255	0
0	254	255	165	12	0	53	105	230	119	66	21	0	148	255	255	238	0
0	254	255	0	174	215	0	201	252	175	6	178	218	0	255	247	255	0
0	255	238	25	213	236	11	232	255	254	7	214	214	14	249	255	255	0
0	255	255	0	225	214	1	255	246	253	0	241	213	0	255	245	248	0
0	253	255	164	5	0	178	255	255	251	167	4	0	183	255	255	255	0
0	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Digital Image Processing, 2nd ed.

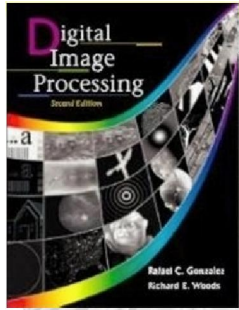
www.imageprocessingbook.com

Zero padding is the default. You can also specify a value other than zero to use as a padding value.

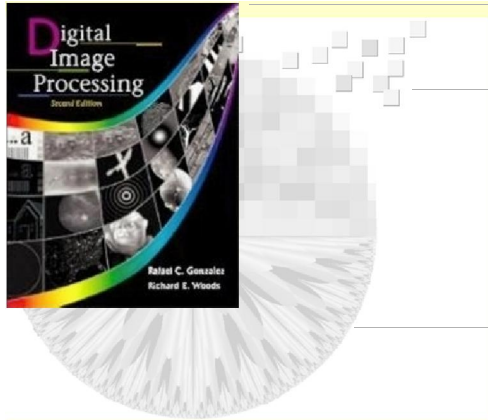
Another solution is replicating the pixel values along the edges:

Replicating

251	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	255
251	251	255	250	255	249	253	255	255	255	255	255	253	251	255	250	255	255
251	251	244	255	246	7	255	254	255	255	242	0	255	248	255	255	254	254
255	255	255	240	183	0	231	247	255	244	255	0	168	255	241	255	252	252
254	254	255	250	12	87	2	255	240	255	0	73	7	253	255	239	255	255
242	242	247	255	0	94	0	254	254	241	0	95	0	255	255	248	247	247
255	255	255	250	69	87	83	2	255	6	107	79	74	249	245	255	250	250
255	255	243	255	156	95	88	0	255	12	58	77	201	239	255	251	253	253
255	255	248	255	255	0	74	106	0	85	84	8	250	255	255	242	255	255
250	250	255	249	255	255	0	78	4	89	49	251	252	255	255	241	255	255
255	255	246	255	252	255	161	57	30	44	150	249	255	255	239	255	255	255
254	254	255	165	12	0	53	105	230	119	66	21	0	148	255	255	238	238
254	254	255	0	174	215	0	201	252	175	6	178	218	0	255	247	255	255
255	255	238	25	213	236	11	232	255	254	7	214	214	14	249	255	255	255
255	255	255	0	225	214	1	255	246	253	0	241	213	0	255	245	248	248
253	253	255	164	5	0	178	255	255	251	167	4	0	183	255	255	255	255
255	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	254
255	255	255	246	255	255	254	253	253	255	255	248	255	252	242	255	254	254



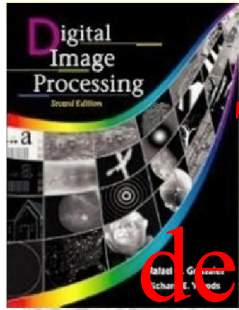
- As a note, if your filter were larger than 3x3, then the "border padding" would have to be extended. For a filter of size 3x3, 'replicate' and 'symmetric' yield the same results.
- The following images show the results of the four different boundary options. The filter used below is a 5x5 averaging filter that was created with the following syntax:
`h=fspecial('average',5)`



Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

[illegible]

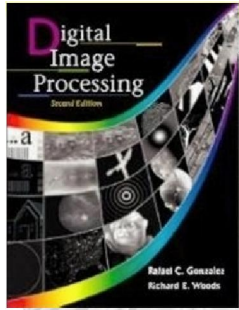


The following MATLAB function demonstrates how spatial filtering may be applied to an image

- function img = myfilter(f, w)
- %MYFILTER Performs spatial correlation
- % I=MYFILTER(f, w) produces an image that has undergone correlation.
- % f is the original image
- % w is the filter (assumed to be 3x3)
- % The original image is padded with 0's
- %Author: Nova Scheidt
-
- % check that w is 3x3
- [m,n]=size(w);
- if m~=3 | n~=3
- error('Filter must be 3x3')
- end

© 2002 R. C. Gonzalez & R. E. Woods

[illegible]



Fourier Transforms

تحويلات فوريير الرياضية للصور الرقمية

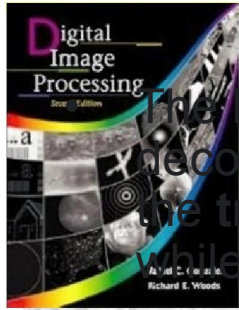
The Fourier transform is a representation of an image as a sum of complex exponentials of varying magnitudes, frequencies, and phases.

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

- Fourier components: sinusoidal patterns

$F(u, v)$ Fourier Coefficients

$f(x, y)$ is an image



Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

The **Fourier Transform** is an important **image** processing tool which is used to decompose an **image** into its sine and cosine components. The output of the **transformation** represents the **image** in the **Fourier** or frequency domain, while the input **image** is the spatial domain equivalent.

- The **DFT is used** to convert an **image** from the spatial domain into frequency domain, in other words it allows us to separate high frequency from low frequency coefficients and neglect or alter specific frequencies leading to an **image** with less information but still with a convenient level of quality .
- Fourier transform is a mathematical formula by which we can extract out the frequency domain components of a continuous time domain signal. Using fourier transform we can process time domain signal in frequency domain. We can use various Frequency domain filters to process the signal.
- If signal is discrete, Discrete Fourier Transform use to analyse discrete signal
- The **Fast Fourier Transform (FFT)** is commonly used to transform an **image** between the spatial and frequency domain. Unlike other domains such as Hough and Radon, the **FFT** method preserves all original data. Plus, **FFT** fully transforms **images** into the frequency domain, unlike time-frequency or wavelet transforms.



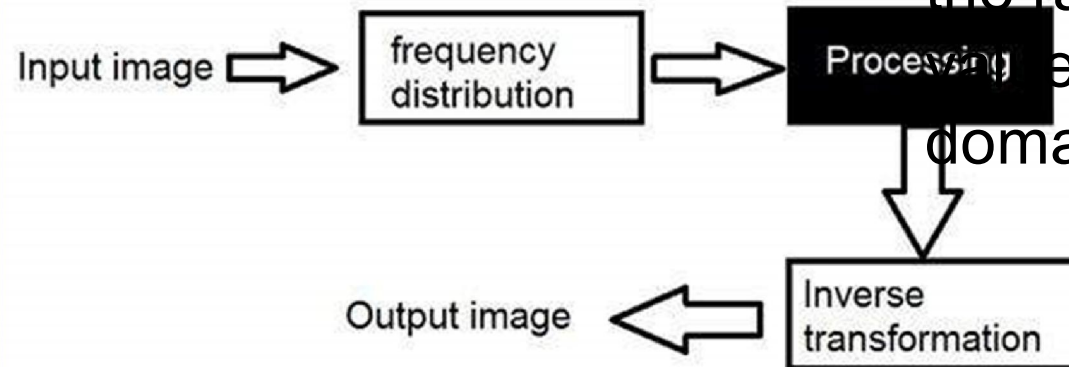
Difference between spatial domain and frequency domain

input image matrix

processing

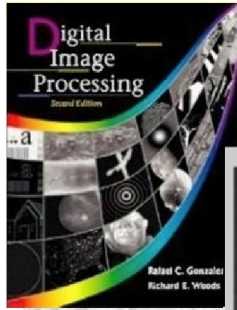
output image matrix

In spatial domain, we deal with images as it is. The value of the pixels of the image change with respect to scene. Whereas in frequency domain, we deal with the rate at which the pixel values are changing in spatial domain.



Frequency domain

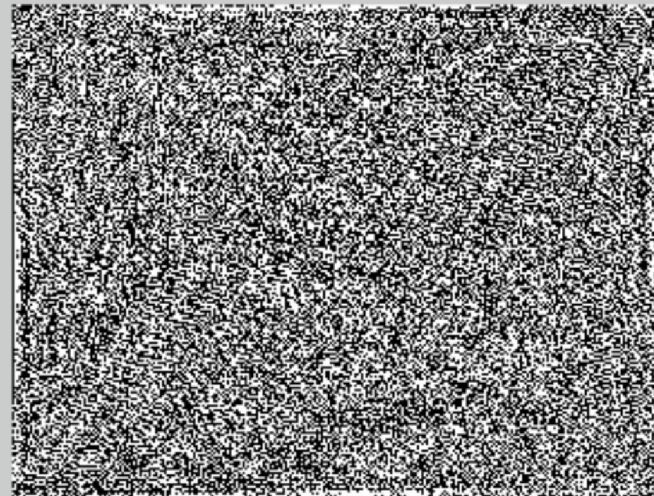
We first transform the image to its frequency distribution. Then our black box system performs whatever processing it has to perform, and the output of the black box in this case is not an image, but a transformation. After performing inverse transformation, it is converted into an image which is then



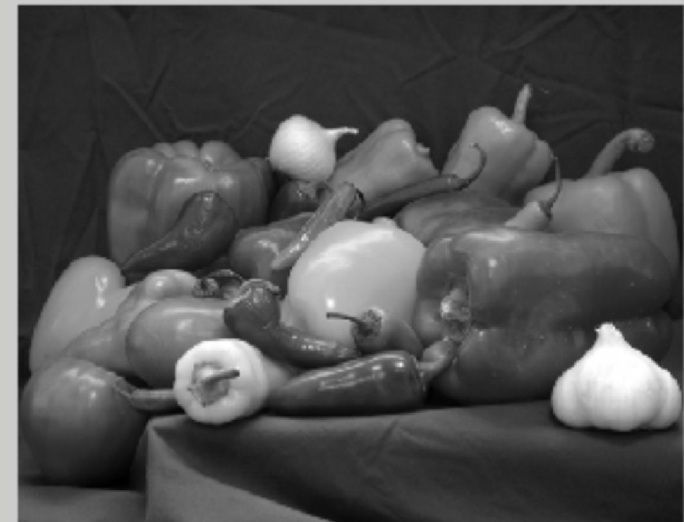
example

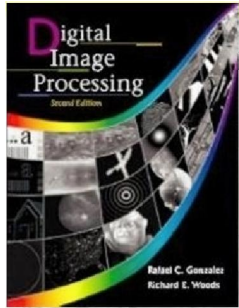


fourier transforms



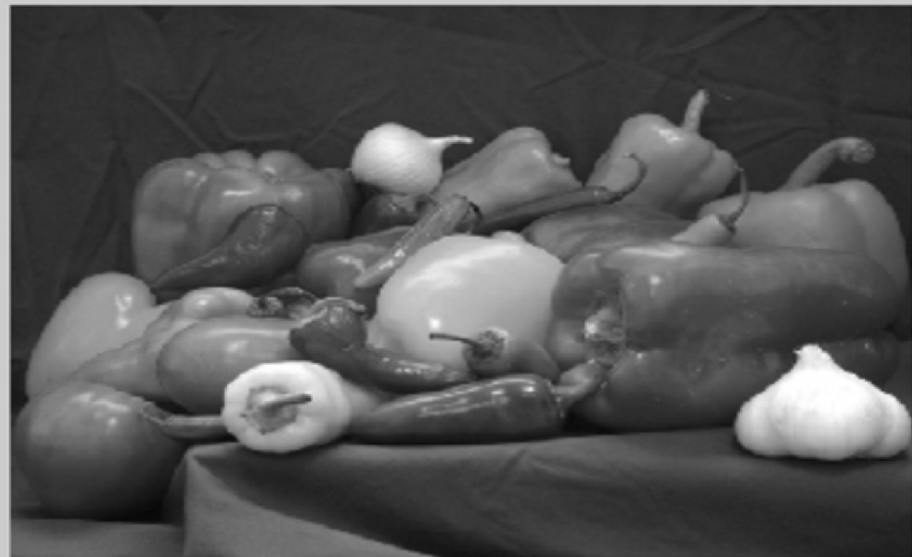
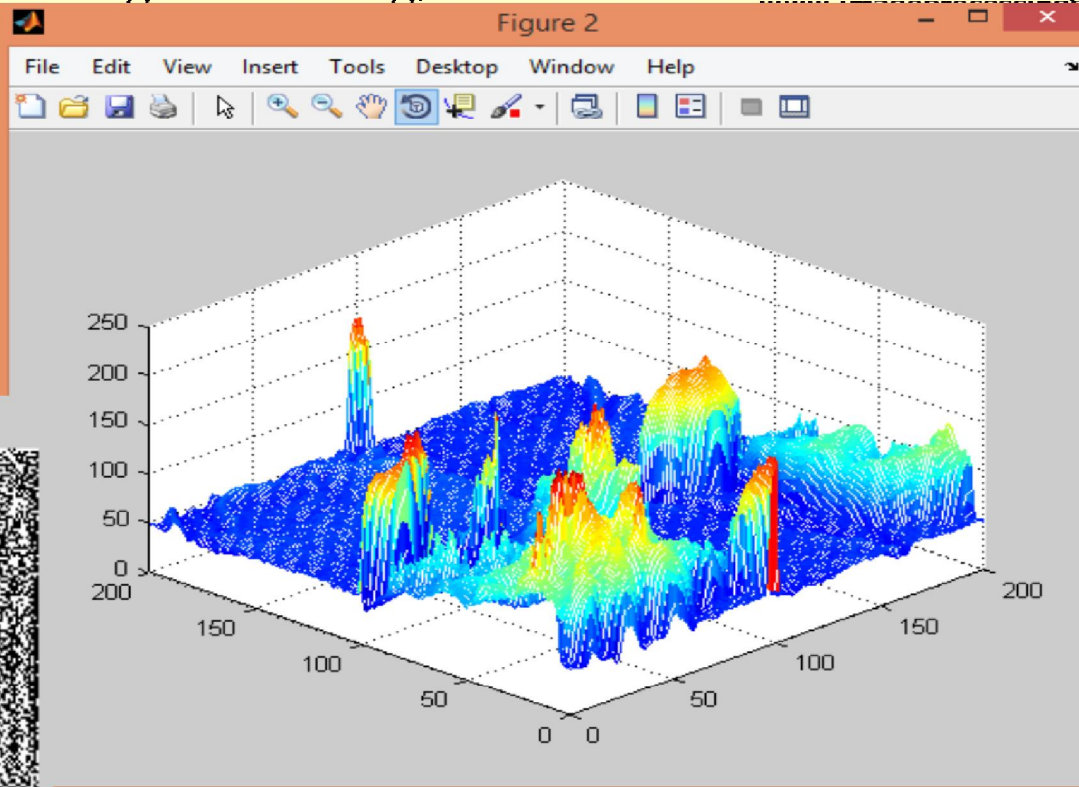
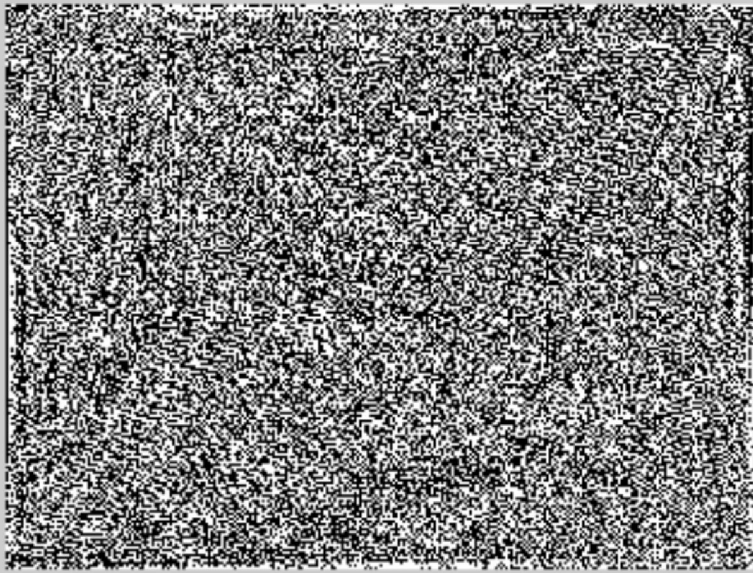
inverse fourier transforms





Digital Image Processing, 2nd ed.

fourier transforms



Background

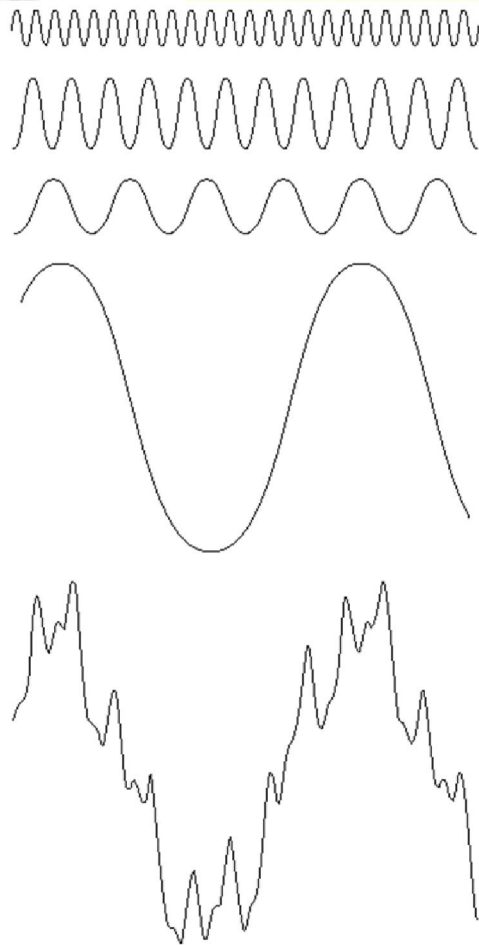
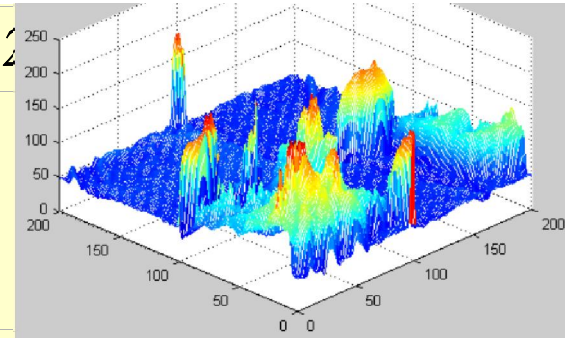
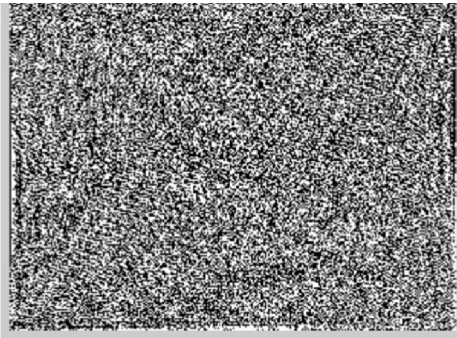
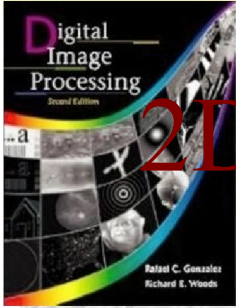


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

- The **frequency domain** refers to the plane of the two dimensional discrete Fourier transform of an image.
- The purpose of the Fourier transform is to represent a signal as a linear combination of sinusoidal signals of various frequencies.



54	48	48	52	67	111	144	160	162	158
54	48	48	49	61	106	141	160	164	158
48	45	48	49	56	97	138	160	167	160
50	51	57	56	61	101	135	161	170	162
59	60	61	55	60	103	134	162	172	164
62	61	55	44	49	96	133	163	174	165
56	45	53	54	41	99	137	163	171	160
55	45	55	56	42	94	136	164	173	163
53	45	58	59	44	86	134	162	173	165
54	47	61	60	46	79	131	160	172	165
57	51	63	58	49	75	133	162	174	167
63	57	62	54	52	74	138	166	176	168
70	62	61	49	54	77	139	166	174	164



2D Fourier transform for digital image

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)} \quad u = 0, 1, 2, \dots, M-1 \text{ and } v = 0, 1, 2, \dots, N-1$$

$$\tilde{X} = F_N X F_N \quad \text{for image } f(x, y) \text{ of size } M \times N$$

$$F_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & \dots & W_N^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \dots & W_N^{(N-1)^2} \end{bmatrix}$$

IDFT inverse discrete Fourier Transform

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{-j2\pi(ux/M + vy/N)} \quad x = 0, 1, 2, \dots, M-1 \text{ and } y = 0, 1, 2, \dots, N-1$$

$$X = \frac{1}{N^2} F_N^* \tilde{X} F_N^*$$

$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

where $\omega = e^{-\frac{2\pi j}{N}}$ is a primitive N th root of unity in which $j = \sqrt{-1}$.

$$F_N^* = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N^{-1} & W_N^{-2} & \dots & W_N^{1-N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{1-N} & W_N^{2(1-N)} & \dots & W_N^{-(N-1)} \end{bmatrix}$$