DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE MATERIALS

CS 463 DIGITAL IMAGE PROCESSING

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- ♦ Established in: 2002
- ♦ Course offered : B.Tech in Computer Science and Engineering
  M.Tech in Computer Science and Engineering
  M.Tech in Cyber Security
- ♦ Approved by AICTE New Delhi and Accredited by NAAC
- ♦ Affiliated to A P J Abdul Kalam Technological University.

DEPARTMENT VISION

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

DEPARTMENT MISSION

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.
5.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.

PEO2: Graduates will be able to Analyse, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.

PEO3: Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.

PEO4: Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Teamworkand leadership qualities.

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

COURSE OUTCOMES

SUBJECT CODE: C406		
COURSE OUTCOMES		
C406.1	K3	**Acquire** knowledge on different methods for image acquisition, storage and representation in digital devices and computers
C406.2	K1	**Appreciate** role of image transforms in representing, highlighting, and modifying image features
C406.3	K5	**Interpret** the mathematical principles in digital image enhancement and apply them in spatial domain and frequency domain
C406.4	K6	**Apply** various methods for segmenting image and identifying image components
C406.5	K5	**Summarize** different reshaping operation on the image and their practical applications
C406.6	K1	**Identify** image representation techniques that enables encoding and decoding images

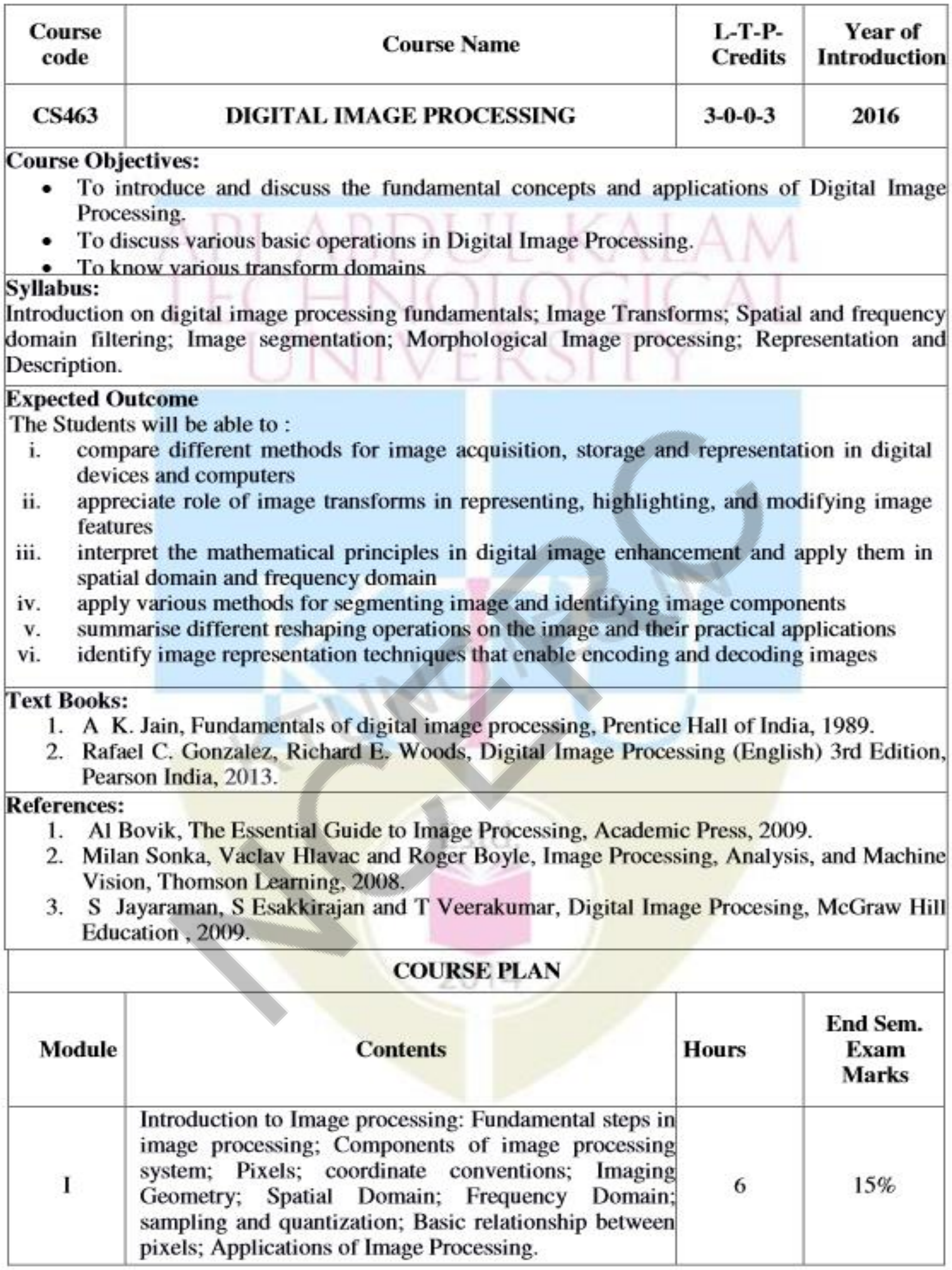
MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

CO'S	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C406.1	3	2	-	-		-	-	-	-	-	-	-
C406.2	3	-	3	3	2	-	-	-	-	-	-	-
C406.3	3	-	2	3	3	-	-	-	-	-	-	-
C4064	3	2	3	3	2	-	-	-	-	-	-	-
C406.5	2	3	3	3	3	-	-	-	-	-	2	3
C406.6	2	2	3	3	3	-	-	-	-	-	3	3
C406	2.67	2.25	2.8	3	2.6	-	-	-	-	-	2.5	3

CO'S	PSO1	PSO2	PSO3
C406.1	2	-	2
C406.2	2	2	3
C406.3	3	2	-
C4064	-	2	3
C406.5	-	3	2
C406.6	-	2	3
C406	2.33	2.2	2.6

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

SYLLABUS



Course code	Course Name	L-T-P-Credits	Year of Introduction
CS463	DIGITAL IMAGE PROCESSING	3-0-0-3	2016

Course Objectives:

- To introduce and discuss the fundamental concepts and applications of Digital Image Processing.
- To discuss various basic operations in Digital Image Processing.
- To know various transform domains

Syllabus:

Introduction on digital image processing fundamentals; Image Transforms; Spatial and frequency domain filtering; Image segmentation; Morphological Image processing; Representation and Description.

Expected Outcome

The Students will be able to :

i. compare different methods for image acquisition, storage and representation in digital devices and computers

ii. appreciate role of image transforms in representing, highlighting, and modifying image features

iii. interpret the mathematical principles in digital image enhancement and apply them in spatial domain and frequency domain

iv. apply various methods for segmenting image and identifying image components

v. summarise different reshaping operations on the image and their practical applications

vi. identify image representation techniques that enable encoding and decoding images

Text Books:

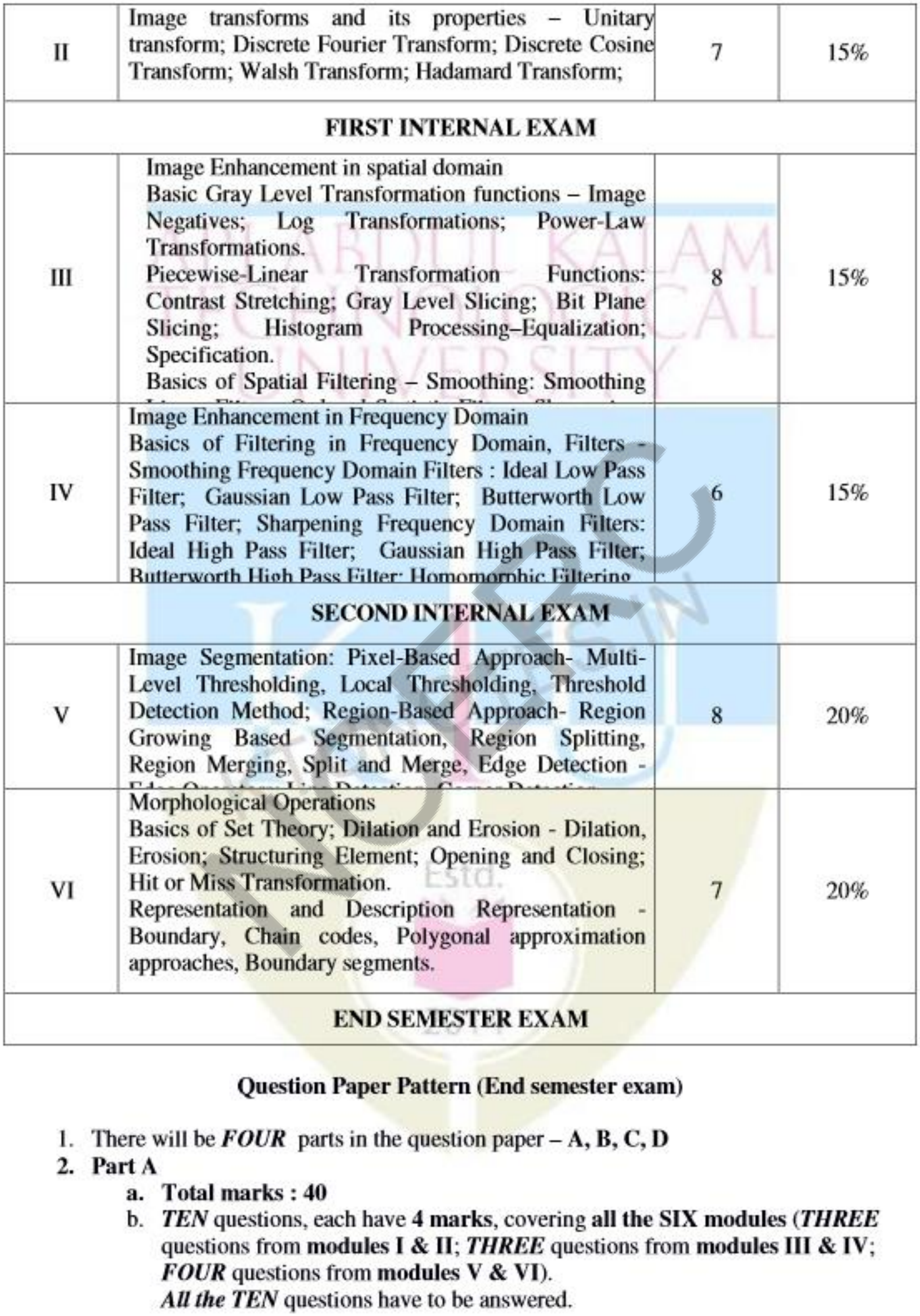
1. A K. Jain, Fundamentals of digital image processing, Prentice Hall of India, 1989.
2. Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing (English) 3rd Edition, Pearson India, 2013.

References:

1. Al Bovik, The Essential Guide to Image Processing, Academic Press, 2009.
2. Milan Sonka, Vaclav Hlavac and Roger Boyle, Image Processing, Analysis, and Machine Vision, Thomson Learning, 2008.
3. S Jayaraman, S Esakkirajan and T Veerakumar, Digital Image Procesing, McGraw Hill Education , 2009.

COURSE PLAN

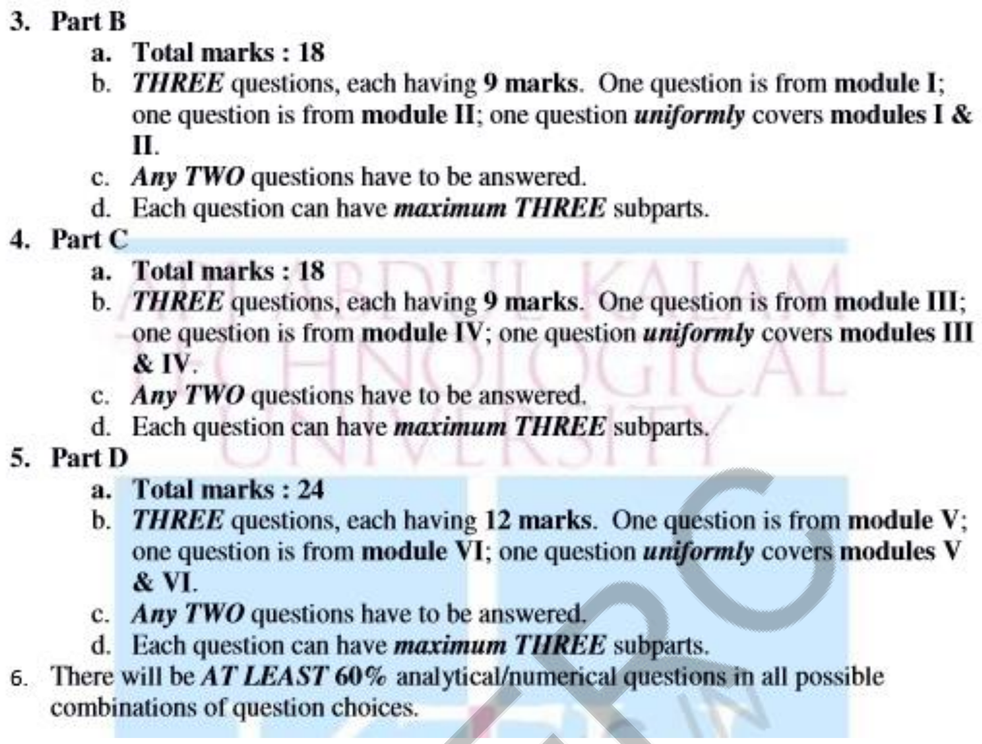
Module	Contents	Hours	End Sem. Exam Marks
I	Introduction to Image processing: Fundamental steps in image processing; Components of image processing system; Pixels; coordinate conventions; Imaging Geometry; Spatial Domain; Frequency Domain; sampling and quantization; Basic relationship between pixels; Applications of Image Processing.	6	15%



II	Image transforms and its properties – Unitary transform; Discrete Fourier Transform; Discrete Cosine Transform; Walsh Transform; Hadamard Transform;	7	15%
FIRST INTERNAL EXAM			
III	Image Enhancement in spatial domain Basic Gray Level Transformation functions – Image Negatives; Log Transformations; Power-Law Transformations. Piecewise-Linear Transformation Functions: Contrast Stretching; Gray Level Slicing; Bit Plane Slicing; Histogram Processing–Equalization; Specification. Basics of Spatial Filtering – Smoothing: Smoothing	8	15%
IV	Image Enhancement in Frequency Domain Basics of Filtering in Frequency Domain, Filters - Smoothing Frequency Domain Filters : Ideal Low Pass Filter; Gaussian Low Pass Filter; Butterworth Low Pass Filter; Sharpening Frequency Domain Filters: Ideal High Pass Filter; Gaussian High Pass Filter; Butterworth High Pass Filter; Homomorphic Filtering	6	15%
SECOND INTERNAL EXAM			
V	Image Segmentation: Pixel-Based Approach- Multi-Level Thresholding, Local Thresholding, Threshold Detection Method; Region-Based Approach- Region Growing Based Segmentation, Region Splitting, Region Merging, Split and Merge, Edge Detection -	8	20%
VI	Morphological Operations Basics of Set Theory; Dilation and Erosion - Dilation, Erosion; Structuring Element; Opening and Closing; Hit or Miss Transformation. Representation and Description Representation - Boundary, Chain codes, Polygonal approximation approaches, Boundary segments.	7	20%
END SEMESTER EXAM			

Question Paper Pattern (End semester exam)

1. There will be ***FOUR*** parts in the question paper – **A, B, C, D**
2. **Part A**
   a. **Total marks : 40**
   b. ***TEN*** questions, each have **4 marks**, covering **all the SIX modules** (***THREE*** questions from **modules I & II**; ***THREE*** questions from **modules III & IV**; ***FOUR*** questions from **modules V & VI**).
   All the TEN questions have to be answered.



3. **Part B**
    a. **Total marks : 18**
    b. ***THREE*** questions, each having **9 marks**. One question is from **module I**; one question is from **module II**; one question ***uniformly*** covers **modules I & II**.
    c. ***Any TWO*** questions have to be answered.
    d. Each question can have ***maximum THREE*** subparts.
4. **Part C**
    a. **Total marks : 18**
    b. ***THREE*** questions, each having **9 marks**. One question is from **module III**; one question is from **module IV**; one question ***uniformly*** covers **modules III & IV**.
    c. ***Any TWO*** questions have to be answered.
    d. Each question can have ***maximum THREE*** subparts.
5. **Part D**
    a. **Total marks : 24**
    b. ***THREE*** questions, each having **12 marks**. One question is from **module V**; one question is from **module VI**; one question ***uniformly*** covers **modules V & VI**.
    c. ***Any TWO*** questions have to be answered.
    d. Each question can have ***maximum THREE*** subparts.
6. There will be ***AT LEAST 60%*** analytical/numerical questions in all possible combinations of question choices.

QUESTION BANK

MODULE I				
Q:NO:	**QUESTIONS**	**CO**	**KL**	**PAGE NO:**
1	Define image processing	CO1	K1	1
2	Explain fundamental steps in image processing	CO1	K2	1
3	Explain Components of image processing system	CO1	K2	4
4	Explain how images are represented	CO1	K3	7
5	Differentiate spatial domain and frequency domain representations of image	CO1	K4	8
6	Illustrate the relationships among pixels with examples	CO1	K4	11
7	Explain distance measures	CO1	K2	15
MODULE II				
1	Explain the concept of image transforms	CO2	K2	17
2	Explain 2D linear transform	CO2	K2	19
3	Analyze the applications of image transforms	CO2	K4	21
4	Explain the concept of unitary matrix	CO2	K1	22
5	Differentiate unitary and orthogonal matrix	CO2	K4	22
6	Explain properties of 2D transforms	CO2	K2	24
MODULE III				
1	Explain gray level transformations	CO3	K2	35
2	Differentiate Log transforms and power-law transforms	CO3	K4	36
3	Explain contrast stretching in detail	CO3	K2	37
4	Explain gray level slicing	CO3	K2	38
5	Differentiate gray level slicing and bit plane slicing	CO3	K4	38

6	Explain Gray level transformations in detail	CO3	K2	41
7	Explain the process of histogram equalization	CO3	K2	43
8	Explain the process of histogram matching	CO3	K2	46
9	Explain image subtraction	CO3	K2	49
10	Explain image averaging	CO3	K2	50
11	Explain how smoothing in spatial filters are done	CO3	K2	54
12	Explain various masks used in image processing	CO3	K2	57
MODULE IV				
1	Explain the transfer function of Butterworth low pass filter	CO4	K2	78
2	Analyze the disadvantages of ideal low pass filter	CO4	K3	74
3	List out the steps involved in frequency domain filtering	CO4	K2	73
4	Explain how edge detection is done	CO4	K2	92
5	With appropriate figure explain the steps involved in frequency domain filtering	CO4	K2	74
6	Explain ideal low pass filter ,Analyze the advantages and disadvantages of ideal low pass filter	CO4	K2	74
7	Explain un sharp masking and high boost filtering	CO4	K2	83
8	Differentiate the following image enhancement techniques in frequency domain i) Gaussian high pass filter ii) Butterworth high pass filter	CO4	K4	82

MODULE V				
1	Differentiate various types of image segmentations	CO5	K4	89
2	Explain and illustrate How is a line detected? Give the mask to detect horizontal, vertical, + 45o slope and -45o slope line	CO5	K3	93
3	Write short notes on region splitting	CO5	K3	106
4	Differentiate various thresholding methods	CO4	K4	98
4	Explain region merging	CO5	K2	106
5	Explain region based segmentation in detail	CO5	K3	103
6	Differentiate any three similarity based segmentations	CO5	K2	89
7	Differentiate short note on Prewitt, Robert's and Sobel edge detectors	CO5	K4	91
8	Explain different edge detection methods	CO5	K3	92
MODULE VI				
1	Differentiate two image representation schemes	CO6	K4	108
2 3	Briefly explain hit or miss transformation Differentiate various threshold based segmentation	CO6	K3	113
4	Explain boundary based representations	CO6	K2	108
5	Explain and illustrate Hit or miss transform morphological algorithm with an example	CO6	K2	113
6	Differentiate dilation and erosion with an example	CO6	K4	118
7	Morphological operations are important in image processing, justify your answers	CO6	K2	116

APPENDIX 1

CONTENT BEYOND THE SYLLABUS

MODULE 1

Introduction to Image processing: Fundamental steps in image processing; Components of image processing system; Pixels; coordinate conventions; Imaging Geometry; Spatial Domain; Frequency Domain; sampling and quantization; Basic relationship between pixels; Applications of Image Processing.

What Is Digital Image Processing?

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 of 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*. The field of *digital image processing* refers to processing digital images by means of a digital computer. 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*, *pels*, and *pixels*. *Pixel* is the term most widely used to denote the basic elements of a digital image.

1.2 Fundamental Steps in Digital Image Processing

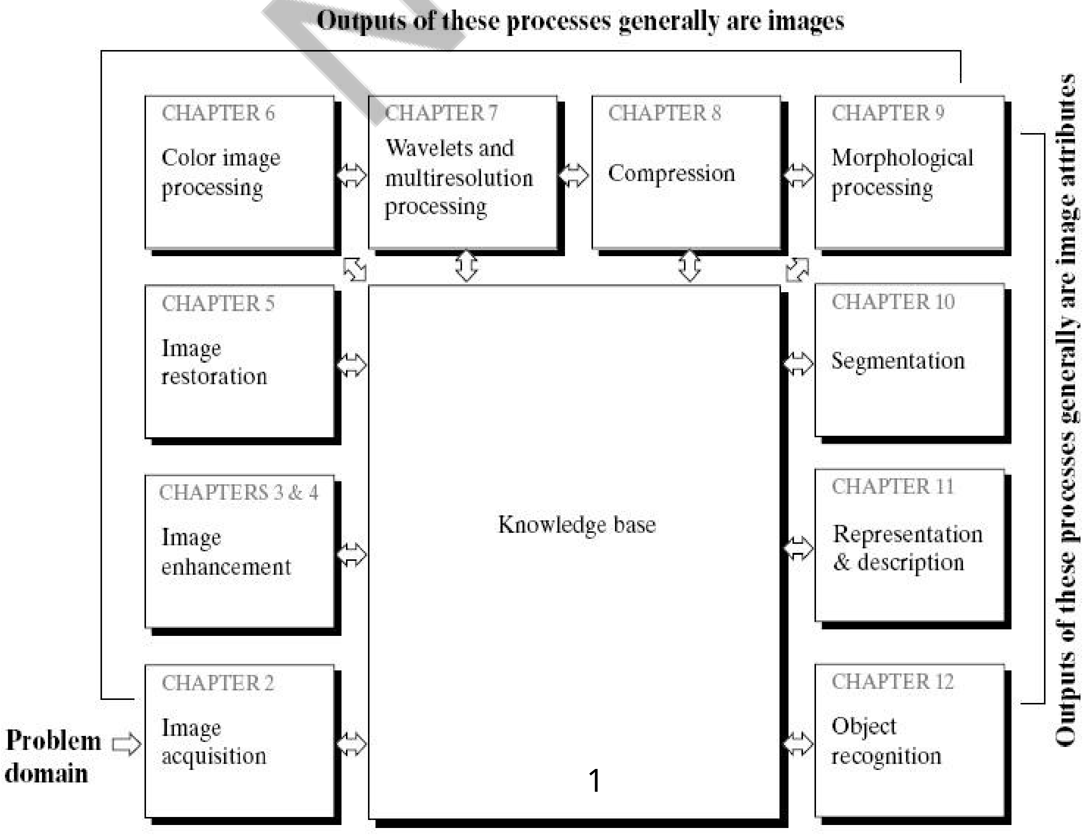


Image acquisition is the first process, Before any video or image processing can commence an image must be captured by a camera and converted into a manageable entity. This is the process known as image acquisition.

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interest in an image. Example : enhancement, when we increase the contrast of an image **"it looks better."** It is important to keep in mind that enhancement is a very subjective area of image processing

Image restoration Unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation. Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a "good" enhancement result.

Color image processing is an area that gained importance because of the significant increase in the use of digital images over the Internet. Color is used as the basis for extracting features of interest in an image.

Wavelets are the foundation for representing images in various degrees of resolution. In particular, this material is used for image data compression and for pyramidal representation, in which images are subdivided successively into smaller regions.

Compression, as the name implies, deals with techniques for reducing the storage required to save an image, or the bandwidth required to transmit it. Although storage technology has improved significantly over the past decade, the same cannot be said for transmission capacity. This is true particularly in uses of the Internet, which are characterized by significant pictorial content. Image compression is familiar (perhaps inadvertently) to most users of computers in the form of image file extensions, such as the jpg file extension used in the JPEG(Joint Photographic Experts Group) image compression standard.

Morphological processing deals with tools for extracting image components that are useful in the representation and description of shape.

Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually. On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure.

Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region (i.e., the set of pixels separating one image region from another) or all the points in the region itself. In either case, converting the data to a form suitable for computer processing is necessary. The first decision that must be made is whether the data should be represented as a boundary or as a complete region. Boundary representation is appropriate when the focus is on external shape characteristics, such as corners and inflections. Regional representation is appropriate when the focus is on internal properties, such as texture or skeletal shape. In some applications, these representations complement each other. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. A method must also be specified for describing the data so that features of interest are highlighted. *Description*, also called *feature selection*, deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

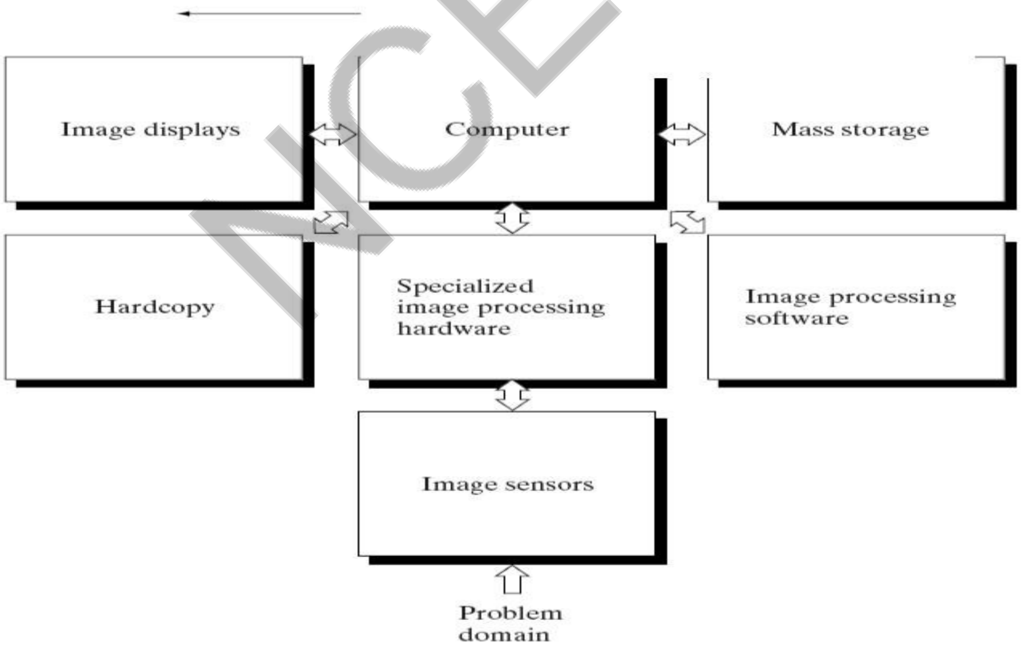
Recognition is the process that assigns a label (e.g., “vehicle”) to an object based on its descriptors.

Knowledge base **:** Knowledge about a problem domain is coded into an image processing system in the form of a knowledge database. This knowledge may be as simple as detailing regions of an image where the information of interest is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex, such as an interrelated list of all major possible defects in a materials inspection problem or an image database containing high-resolution satellite images of a region in connection with change-detection applications. In addition to guiding the operation of each processing module, the knowledge base also controls the interaction between modules. This distinction is made in Fig. by the use of double headed arrows between the processing modules and the knowledge base, as opposed to single-headed arrows linking the processing modules.

Components of an Image Processing System

The function of each component is discussed in the following paragraphs, starting with image sensing. With reference to ***sensing,*** two elements are required to acquire digital images. The first is a physical device that is sensitive to the energy radiated by the object we wish to image. The second, called a ***digitizer***, is a device for converting the output of the physical sensing device into digital form. For instance, in a digital video camera, the sensors produce an electrical output proportional to light intensity. The digitizer converts these outputs to digital data.

Specialized image processing hardware usually consists of the digitizer just mentioned, 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. One example of how an ALU is used is in averaging images as quickly as they are digitized, for the purpose of noise reduction. 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 (e.g., digitizing and averaging video images at 30 frames_s) that the typical main computer cannot handle.



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, but our interest here is on general-purpose image processing systems. In these systems, almost any well-equipped PC-type machine is suitable for offline image processing tasks.

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. More sophisticated software packages allow the integration of those modules and general- purpose software commands from at least one computer language.

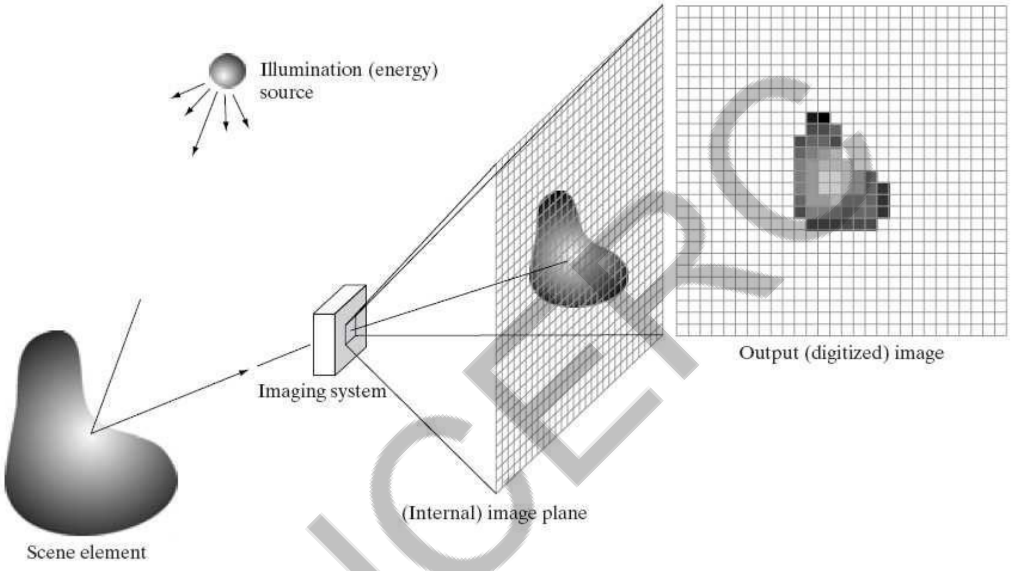
Mass storage capability is a must in image processing applications.An image of size 1024*1024 pixels, in which the intensity of each pixel is an 8-bit quantity, requires one megabyte of storage space if the image is not compressed. When dealing with thousands, or even millions, of images, providing adequate storage in an image processing system can be a challenge. 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, and (3) archival storage, characterized by infrequent access. Storage is measured in bytes (eight bits), Kbytes (one thousand bytes), Mbytes (one million bytes), Gbytes (meaning giga, or one billion, bytes), and T bytes (meaning tera, or one trillion, bytes).

Image displays in use today are mainly color (preferably flat screen) TV monitors. Monitors are driven by the outputs of image and graphics display cards that are an integral part of the computer system. Seldom are there requirements for image display applications that cannot be met by display cards available commercially as part of the computer system. In some cases, it is necessary to have stereo displays, and these are implemented in the form of headgear containing two small displays embedded in goggles worn by the user.

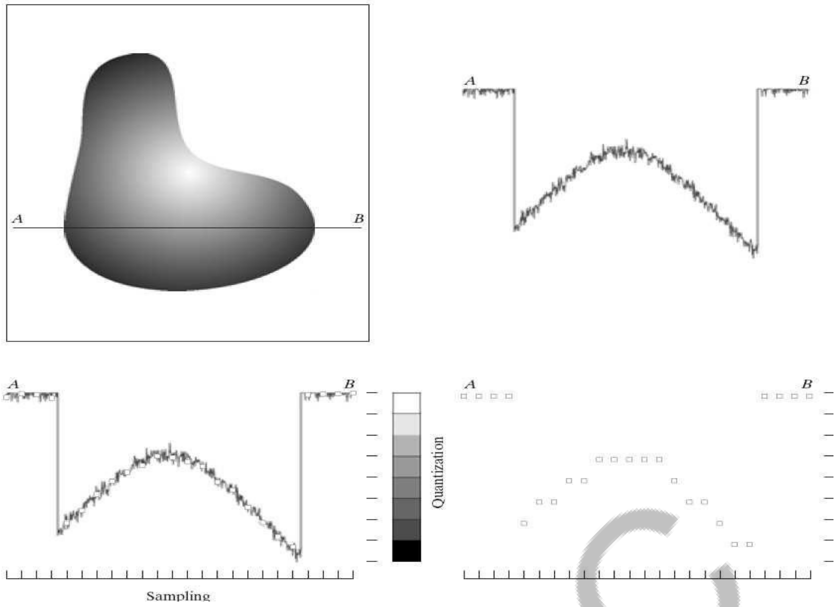
Hardcopy devices for recording images include laser printers, film cameras, heat-sensitive devices, inkjet units, and digital units, such as optical and CD-ROM disks. Film provides the highest possible resolution, but paper is the obvious medium of choice for written material. For presentations, images are displayed on film transparencies or in a digital medium if image projection equipment is used. The latter approach is gaining acceptance as the standard for image presentations.

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. This is improving quickly as a result of optical fiber and other broadband technologies.

Image Sampling and Quantization



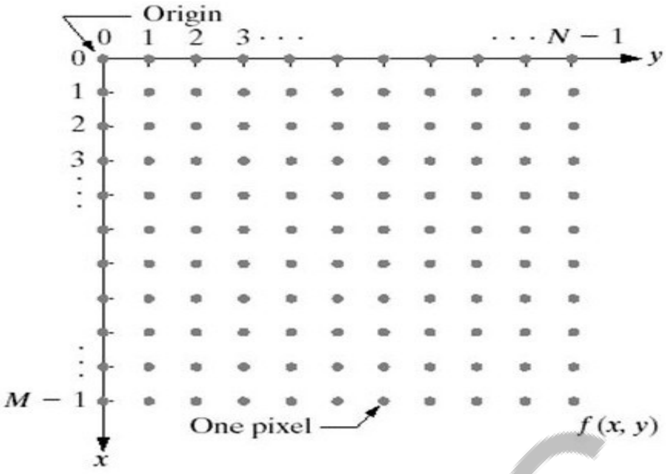
To create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: *sampling* and *quantization*. A continuous image, f(x, y), that we want to convert to digital form. An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. To convert it to digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.



The one-dimensional function shown in Fig is a plot of amplitude (gray level) values of the continuous image along the line segment AB. The random variations are due to image noise. To sample this function, we take equally spaced samples along line AB, The location of each sample is given by a vertical tick mark in the bottom part of the figure. The samples are shown as small white squares superimposed on the function. The set of these discrete locations gives the sampled function. However, the values of the samples still span (vertically) a continuous range of gray-level values. In order to form a digital function, the gray-level values also must be converted (*quantized*) into discrete quantities. The right side gray-level scale divided into eight discrete levels, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight gray levels. The continuous gray levels are quantized simply by assigning one of the eight discrete gray levels to each sample. The assignment is made depending on the vertical proximity of a sample to a vertical tick mark. The digital samples resulting from both sampling and quantization.

Co-ordinate conventions and Representation of Digital Images:

The result of sampling and quantization is matrix of real numbers. Assume that an image f(x,y) is sampled so that the resulting digital image has M rows and N Columns. The values of the coordinates (x,y) now become discrete quantities thus the value of the coordinates at orgin become (x,y) =(o,o) The next Coordinates value along the first signify the image along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Thus the right side of the matrix represents a digital element, pixel or pel. Co-ordinate convention used to represent digital image is shown in following figure.



Above co-ordiante conventions help us to represent an MXN image in the following form.

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

Gray level values

Due to processing storage and hardware consideration, the number gray levels typically is an integer power of 2.

$L=2^k$

Then, the number, b, of bites required to store a digital image is B=M *N* k

When M=N, the equation become b=N^2 *k

When an image can have 2k gray levels, it is referred to as “k- bit”. An image with 256 possible gray levels is called an “8- bit image” ($256=2^8$).

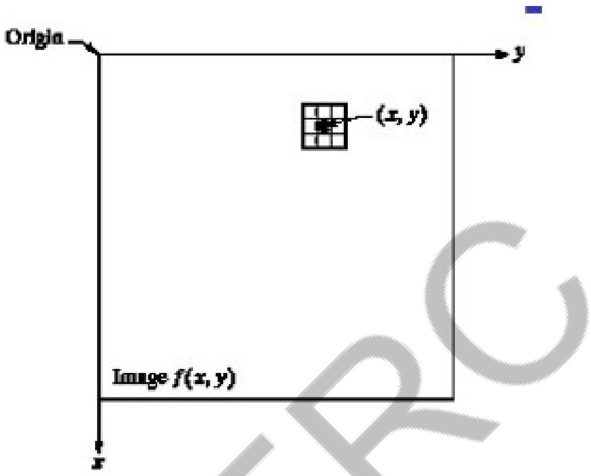
Spatial domain

The term spatial domain refers to the image plane itself and approaches in this categories are based on direct manipulation of pixel in an image. Spatial domain process are denoted by the expression

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

Where f(x,y)- input image, T- operator on f, defined over some neighborhood of f(x,y) and g(x,y)-

processed image The neighborhood of a point (x,y) can be explain by using as square or rectangular sub image area centered at (x,y). The center of sub image is moved from pixel to pixel starting at the top left corner. The operator T is applied to each location (x,y) to find the output g at that location . The process utilizes only the pixel in the area of the image spanned by the neighborhood.



Frequency domain

An image is a signal that's perceived in 2 spatial dimensions: height and width. When analyzing images(or signals), it's often helpful to represent it in a form other than spatial extent (or the time domain). If you perform a fourier transform on the image (signal), you can represent the signal another way, in a frequency domain, where the signal has been decomposed into a series of constituent trigonometric functions. This representation allows you to see, measure, and modify the signal in a different way than is possible in the spatial domain. The inverse Fourier transform converts the frequency-domain function back to the spatial domain. In this domain, pixel location is represented by its x- and y-frequencies and its value is represented by an amplitude.

From the context of image processing, it is to study the change in pixel values in the image. These change in frequency is a characteristic of change in geometry of the image(spatial distribution). Edges reflects high frequency components, smooth regions have low frequency components.

Difference between spatial domain and frequency domain

Spatial domain :

- Deal with images as it is.
- The value of the pixels of the image change with respect to scene.

Frequency domain :

- Deal with the rate at which the pixel values are changing in spatial domain

2.5 Some Basic Relationships between Pixels

Here, we consider some important relationships between pixels in a digital image.

Neighbours of a Pixel

A pixel *p* at coordinates (x, y) has four horizontal and vertical neighbours:

$$(x+1, y),\ (x-1, y),\ (x, y+1),\ (x, y-1)\,.$$

This set of pixels is called the 4-neighbuors of *p*, and denoted by $N_4(p)$.

The four diagonal neighbours of *p* are

$$(x+1, y+1),\ (x+1, y-1),\ (x-1, y+1),\ (x-1, y-1)\,,$$

and are denoted by $N_D(p)$.

Adjacency, Connectivity, Regions, and Boundaries

Let V be the set of intensity values used to define adjacency. In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1.

In a gray-scale image, set V typically contains more elements. For example, with a range of possible intensity values 0 to 255, set V could be any subset of these 256 values.

Consider three types of adjacency:

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

(b) 8-adjacency. Two pixels p and q with values from V are 8-adjacency if q is in the set $N_8(p)$.

(c) *m*-adjacency (mixed adjacency). Two pixels p and q with values from V are *m* -adjacency if

(i) q is in the $N_4(p)$, or

(ii) q is in the $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.

Mixed adjacency is a modified of 8-adjacency.

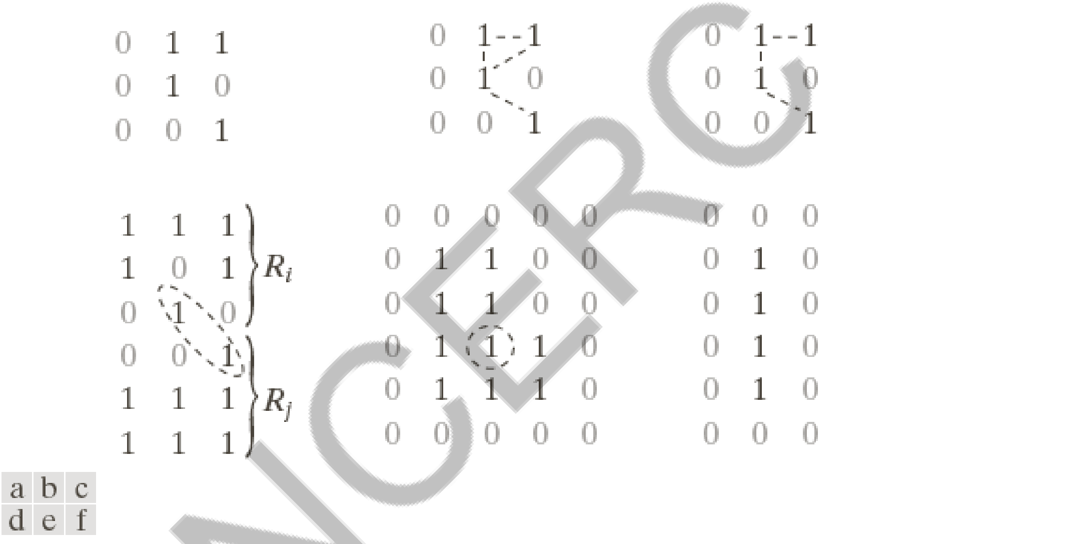


FIGURE 2.25 (a) An arrangement of pixels. (b) Pixels that are 8-adjacent (adjacency is shown by dashed lines; note the ambiguity). (c) *m*-adjacency. (d) Two regions that are adjacent if 8-adjecency is used. (e) The circled point is part of the boundary of the 1-valued pixels only if 8-adjacency between the region and background is used. (f) The inner boundary of the 1-valued region does not form a closed path, but its outer boundary does.

For example, consider the arrangement shown in Figure 2.25 (a) for $V = \{1\}$.

The three pixels at the top of Figure 2.25 (b) show ambiguous 8-adjacency, which is removed by using *m*-adjacency, as shown in Figure 2.25 (c).

A path from pixel *p* with coordinates (x, y) to pixel *q* with coordinates (s, t) is a sequence of distinct pixels with coordinates

$$(x_0, y_0),\ (x_1, y_1),\ \cdots,\ (x_n, y_n),$$

where $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$, and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \le i \le n$. n is the length of the path. If $(x_0, y_0) = (x_n, y_n)$, the path is a closed path.

We can define 4-, 8-, or *m*-paths depending on the type of adjacency. The path shown in Figure 2.25 (b) between the top right and bottom right points are 8-paths, and the path in Figure 2.25 (c) is an *m*-path.

Let S represent a subset of pixels in an image. Two pixels *p* and *q* are said to be connected in S if there exists a path between them consisting entirely of pixels in S. If it only has one connected component, set S is called a connected set.

Let R be a subset of pixels in an image. We call R a region of the image if R is a connected set.

Two regions, R_i and R_j are said to be adjacent if their union forms a connected set. Regions that are not adjacent are said to be disjoint.

The two regions (of 1s) in Figure 2.25 (d) are adjacent only if 8-adjacency is used.

Suppose that an image contains K disjoint regions, R_k, $k = 1, 2, \ldots, K$, and none of which touches the image border. Let R_u denote the union of all the K regions, and let $(R_u)^c$ denote its complement. We call all the points in R_u the foreground, and all the points in $(R_u)^c$ the background of the image.

The boundary (also called the border or contour) of a region R is the set of points that are adjacent to points in the complement of R.

Again, we must specify the connectivity being used to define adjacency. For example, the point circled in Figure 2.25 (e) is not a member of the border of the 1-valued region if 4-connectivity is used between the region and its background.

As a rule, adjacency between points in a region and its background is defined in terms of 8-adjacency to handle situations like above.

The preceding definition is referred to as the inner border of the region to distinguish it from its outer border, which is the corresponding border in the background.

This issue is important in the development of border-following algorithms. Such algorithms usually are formulated to follow the outer boundary in order to guarantee that the result will form a closed path.

For example, the inner border of the 1-valued region in Figure 2.25 (f) is the region itself.

If R happens to be an entire image, then its boundary is defined as the set of pixels in the first and last rows and columns of the image. This extra definition is required because an image has no neighbours beyond its border.

Distance Measures

For pixels p, q, and z, with coordinates (x,y), (s,t), and (v,w), D is a distance function or metric if

(a) $D(p,q) \geq 0$ ($D(p,q) = 0$ *iff* $p = q$)

(b) $D(p,q) = D(q,p)$, and

(c) $D(p,z) \leq D(p,q) + D(q,z)$.

The Euclidean distance between p and q is defined as

$$D_e(p,q) = \left[(x-s)^2 + (y-t)^2\right]^{1/2} . \tag{2.5-1}$$

The D_4 distance (called the city-block distance) between p and q is defined as

$$D_4(p,q) = \mid x - s \mid + \mid y - t \mid . \tag{2.5-2}$$

Example: the pixels with D_4 distance ≤ 2 from (x,y) form the following contours of constant distance:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

The pixels with $D_4 = 1$ are the 4-neighbuors of (x,y).

The D_8 distance (called the chessboard distance) between p and q is defined as

$$D_8(p,q) = \max(|\ x - s\ |,|\ y - t\ |). \qquad (2.5\text{-}3)$$

Example: the pixels with D_8 distance ≤ 2 from (x, y) form the following contours of constant distance:

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

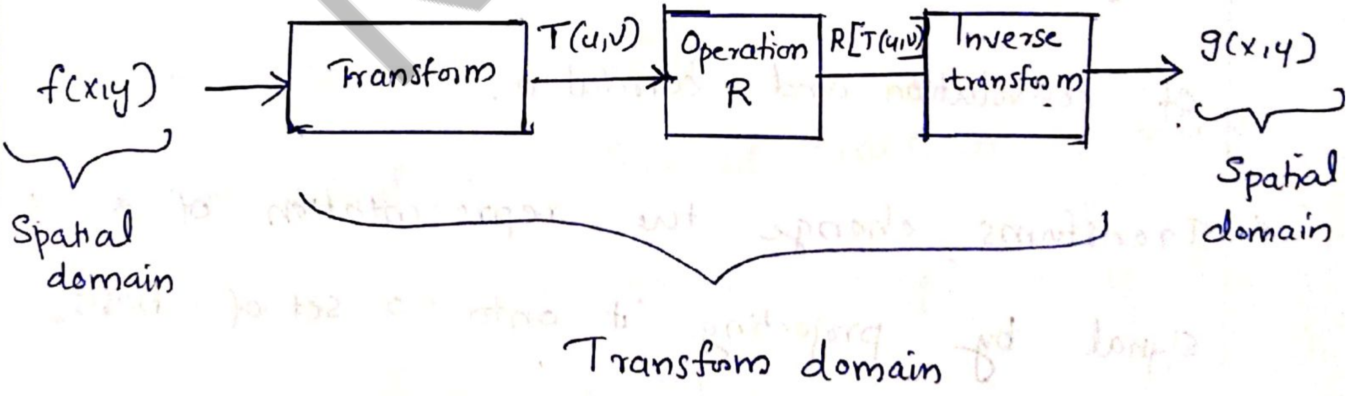
The pixels with $D_8 = 1$ are the 8-neighbuors of (x, y).

Image transforms and its properties _ Unitary transform; Discrete Fourier Transform; Discrete Cosine Transform; Walsh Transform; Hadamard Transform;

Image Transforms :

- Image processing tasks are best formulated by transforming the input images, carrying the specified task in a transform domain, and applying the inverse transform to return to the spatial domain.

- Figure1. General approach for operating in the linear transform domain.



- Figure1 shows the basic steps for performing image processing in the linear transform domain.
- First the input image is transformed, the

transform is then modified by a predefined operation and finally, the output image is obtained by computing the inverse of the modified transform.

- Thus we see that the process goes from the spatial domain to transform domain and then back to the spatial domain.

* Image transforms are extensively used in image processing and image analysis.

* The reason to migrate from one domain to another domain is to perform the task at hand in an easier manner.

* Image transforms are useful for fast computation of convolution and correlation.

* Transforms change the representation of a signal by projecting it onto a set of basis functions.

* The transforms do not change the information content present in the signal.

* Most of the image transforms like Fourier Transform, Discrete Cosine Transform, wavelet transform etc give information about the frequency contents in an image.

Note: → All transforms will not give ~~info~~ frequency domain information.

2-D Linear Transform :

2-D linear transforms, denoted $T(u,v)$, can be expressed in the general form

$$\boxed{T(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot r(x,y,u,v)} \longrightarrow (1)$$

where

equ^n (1) is evaluated for $u = 0, 1, 2, \dots, M-1$ and $v = 0, 1, 2, \dots, N-1$.

$f(x,y)$ = input image

$r(x,y,u,v)$ = Forward transformation kernel.

x, y = spatial variables.

M, N = Row and column dimension of f.

u, v = transform variables.

$T(u,v)$ is called the forward transform of $f(x,y)$.

- Given $T(u,v)$, we can recover $f(x,y)$ using the inverse transform of $T(u,v)$.

$$\boxed{f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v) \cdot s(x,y,u,v)} \longrightarrow (2)$$

for $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$.

where,

$s(x,y,u,v)$ is called the inverse transformation kernel.

- The equations ① and ② together called as a Transform pair.

Separable :

- The forward transformation kernel is said to be separable if

$$r(x,y,u,v) = r_1(x,u) \cdot r_2(y,v).$$

* Inverse transformation kernel is separable if

$$s(x,y,u,v) = s_1(x,u) \cdot s_2(y,v).$$

Symmetric :

- Forward transformation kernel is said to be symmetric if $r_1(x,u)$ is functionally ~~equivale~~ equal to $r_2(y,v)$, so that

$$r(x,y,u,v) = r_1(x,u) \cdot r_1(y,v).$$

- Inverse transformation kernel is symmetric if

$$s(x,y,u,v) = s_1(x,u) \cdot s_1(y,v).$$

Applications of Image Transfoms :

* Preprocessing
  - Filtering
  - Enhancement, etc
* Data Compression.
* Feature Extraction.
  - Edge detection
  - Cornor detection, etc.
* Pattern Recognition. → analyze the principal (dominating) components.

* What does the Image Transform do?

* It represents the given image as a series summation of a set of Unitary Matrices.

* There are two reasons for transforming an image from one representation to another.

* First the transformation may isolate critical components of the image pattern so that they are directly accessible for analysis.

* Second, the transformation may place the image data in a more compact form so that they can be stored and transmitted efficiently.

What is a Unitary Matrix ? :

* A matrix, whose inverse is equal to its conjugate transpose.

* A matrix 'A' is a unitary matrix if

$$\boxed{A^{-1} = A^{*T}}$$

where A^* is conjugate of A (Complex conjugate of matrix A).

Unitary matrixes $\xrightarrow{\text{are called as}}$ Basis functions (Basis images).

Orthogonal Matrix :-

⇒ A matrix, whose inverse is equal to its Transpose.

ie A is orthogonal if

$$\boxed{\begin{gathered} A^{-1} = A^T \\ \text{or} \\ AA^T = A^TA = I \end{gathered}}$$

I = Identity matrix.

⇒ A real orthogonal matrix is Unitary, but a unitary matrix need not be orthogonal.

* The term 'Image transforms' usually refers to a class of unitary matrices used for representing images.

* Just as a one-dimensional signal can be represented by an orthogonal series of basis functions, an image can also be expanded in terms of a discrete set of basis arrays called basis images.

* These basis images ~~are~~ can be generated by unitary matrices.

* An arbitrary continuous signal x(t) can be represented by a series summation of set of orthogonal basis functions.

* The series expansion is given as :

$$x(t) = \sum_{n=0}^{\infty} c_n \, a_n(t)$$ infinite series expansion

$$\hat{x}(t) = \sum_{n=0}^{N-1} c_n \cdot a_n(t)$$ finite series expansion.

↓

It gives an approximate representation of x(t).

$a_n(t)$ is the set of orthogonal basis functions.

c_n is called nth coefficient of expansion

* For continuous functions, orthogonal series expansions provide series coefficients which can be used for any further processing or analysis of the functions.

* But in our case, we are not dealing with continuous signal.

* we are dealing with discrete signals.

* In the case of discrete signals, we have a set of samples (series of samples).

* Let u(n) represents the series of samples.

  ie $\{u(n) ; 0 \leq n \leq N-1\}$.

* we have N number of samples.

* For a one-dimensional sequence $\{u(n), 0 \leq n \leq N-1\}$, represented as a vector u of size N, a unitary transformation is written as

$$\boxed{v = Au.} \rightarrow (1)$$

u → vector of dimension N.

A → unitary matrix of dimension N x N. (Transformation matrix).

v – transformed vector. (dimension N)

⇒ After expansion :

$$\boxed{v(k) = \sum_{n=0}^{N-1} a(k,n)\, u(n) \quad ; \quad k = 0,1,\dots,N-1} \rightarrow (2)$$

- If 'A' is an unitary matrix, we can get back our original vector u.

ie
$$\boxed{\begin{aligned} u &= A^{-1} v \\ u &= A^{*T} . v \end{aligned}} \longrightarrow (3) \quad // . A^{-1} = A^{*T} \text{ ie unitary.}$$

- Representing the above equation in the form of a series summation,

$$\boxed{u(n) = \sum_{k=0}^{n-1} a^{*}(k,n) . v(k) \; ; \quad n = 0, 1, \ldots, N-1} \longrightarrow (4)$$

where u(n) is series summation of basis vectors.

- Expanding the matrix a(k,n), it is of the form,

$$a(k,n) = \begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & \ldots\ldots & a(0,n) \\ a(1,0) & a(1,1) & a(1,2) & \ldots\ldots & a(1,n) \\ a(2,0) & a(2,1) & a(2,2) & \ldots\ldots & a(2,n) \\ \vdots & & & & \\ a(k,0) & \ldots & \ldots & \ldots & a(k,n) \end{bmatrix}$$

- $a^{*}(k,n)$ is the column vector of matrix A^{*T}.

- These column vectors are usually called the <u>basis vectors of A.</u>

- The concept of representing the vector as a series summation of basis vectors can be expanded to 2D signals (image) as well.

- Consider an image $u(m,n)$ of dimension $N \times N$.

- $0 \leq m,n \leq N-1$

- Transformation on this image is given by :

$$\boxed{v(k,l) = \sum\sum_{m,n=0}^{N-1} a_{k,l}(m,n) \cdot u(m,n)} \longrightarrow (5)$$

$$0 \leq k,l \leq N-1$$

- Thus, we have <u>N^2 number of unitary matrices.</u>

- Inverse transformation is given by :

$$\boxed{u(m,n) = \sum\sum_{k,l=0}^{N-1} a^{*}_{k,l}(m,n) \cdot v(k,l)} \longrightarrow (6)$$

$$0 \leq m,n \leq N-1$$

- where $\{a_{k,l}(m,n)\}$, called an image transform, is a set of complete orthonormal discrete basis functions satisfying the properties :

  * orthonormality
  * completeness.

Orthonormality :

$$\sum\sum_{m,n=0}^{N-1} a_{k,l}(m,n) \cdot a^*_{k',l'}(m,n) = \delta(k-k', l-l')$$

Completeness :

$$\sum\sum_{k,l=0}^{N-1} a_{k,l}(m,n) \cdot a^*_{k,l}(m',n') = \delta(m-m', n-n').$$

- The elements $v(k,l)$ are called the transform coefficients and $v \approx \{v(k,l)\}$ is called the transformed image.
- During Inverse transformation if all coefficients are not considered then an approximate reconstructed image will be generated.

$$\hat{u}(m,n) = \sum_{k=0}^{P-1}\sum_{l=0}^{Q-1} a^*_{k,l}(m,n) \cdot v(k,l).$$

- So, we are considering only $P \times Q$ coefficients instead of N^2.
- The sum of squared error is given by :

$$\sigma_e^2 = \sum\sum_{m,n=0}^{N-1} \left[u(m,n) - \hat{u}(m,n)\right]^2$$

- The completeness property assures that this error will be zero for $P=Q=N$.

- Computations required : $O(N^4)$

Separable Unitary Transforms :→

- The number of multiplications and additions required to compute the transform coefficients $v(k,l)$ using equ (5) is $O(N^4)$, which is quite excessive for practical-size images.

- The dimensionality of the problem is reduced ~~to~~ to $O(N^3)$ when the transform is restricted to be separable.

- $a_{k,l}(m,n)$ is separable if it can be represented in the form

$$a_{k,l}(m,n) = a_k(m) \cdot b_l(n)$$
$$\approx a(k,m) \cdot b(l,n)$$

where $\{a_k(m),\ k=0,\dots N-1\}$, $\{b_l(n),\ l=0,\dots,N-1\}$ are one-dimensional complete orthonormal sets of basis vectors.

- If we represent $A \approx \{a(k,m)\}$ and $B \approx \{b(l,n)\}$ in the form of matrices then both should be unitary so that

$$A A^{*T} = A^T A^* = I$$

- If this is correct then we say that the transformation is a separable transformation.

- Usually we assume both the matrices A and B to be same.

- Then the transformation equation changes to

$$\boxed{v(k,l) = \sum_{m,n=0}^{N-1}\sum a(k,m).\ u(m,n)\ .\ a(l,n)} \longrightarrow (7)$$

- In terms of matrices, it is given by :

$$\boxed{V = AUA^{T}} \longrightarrow (8)$$

where all the matrices are of order NxN.

- By applying inverse transformation we will have the original matrix from the coefficient matrix.

- Similarly, inverse transformation can now be written as :

$$\boxed{u(m,n) = \sum_{k,l=0}^{N-1}\sum a^{*}(k,m).\ v(k,l)\ .\ a^{*}(l,n).} \longrightarrow (8)$$

- In terms of matrices, it is given by :

$$\boxed{U = A^{*T}VA^{*}} \longrightarrow (9)$$

- For an MxN rectangular image, the transform pair is

$$V = A_{M}.U.A_{N}.$$

$$U = A_{M}^{*T}\,V\,A_{N}^{*T}$$

where A_M and A_N are MxM and NxN unitary matrices, respectively. These are called <u>two-dimensional separable transformations</u>.

Equation ⑧ can be written as

$$V^T = A\,[AU]^T$$

Transpose properties :

$$(AB)^T = B^T \cdot A^T.$$

$$V = \underbrace{AU}_{A}\,\underbrace{A^T}_{B}$$

$$= (A^T)^T \cdot (AU)^T$$

$$= A\,[AU]^T$$

- We know

if two matrices A and U of order N×N when multiplied requires $O(N^3)$ computations.

- Thus it takes $O(2N^3)$ computations over all instead of $O(N^4)$ when separable transform ~~ate~~ are considered.

Basis Images :

- Let a_k^* denote the k^{th} column of A^{*T}.

- Define the matrices

$$\boxed{A^*_{k,l} = a^*_k \cdot a^{*T}_l} \longrightarrow (10)$$

- The equation ⑥ give a series representation for the image as

$$U = \sum_{k,l=0}^{N-1}\sum v(k,l) \cdot A^*_{k,l} \cdot \longrightarrow (11)$$

- Equation ⑪ expresses any image U as a linear combination of the N^2 matrices $A^*_{k,l}$, with each having dimension of N×N.

$k,l = 0, \ldots, N-1$.

- The matrices $A^*_{k,l}$ are known as the basis images.

- The aim of the transform is to represent the input image in the form of linear combination of a set of basis images.

Que.1 Given the orthogonal matrix $A = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and the image $U = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

Find the unitary forward and inverse transforms.

Also calculate the basis images.

Ans: Given $A = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

$U = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

The transformed image, $V = AUA^T$

$A^T = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

$$\therefore V = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} . \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} . \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 1+3 & 2+4 \\ 1-3 & 2-4 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 4 & 6 \\ -2 & -2 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$V = \frac{1}{2}\begin{bmatrix} 4+6 & 4-6 \\ -2+-2 & -2+2 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 10 & -2 \\ -4 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix}$$

Inverse transform, $U = A^{*T} V A^{*}$

$$A^{*} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$A^{*T} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\therefore U = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 5-2 & -1 \\ 5+2 & -1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 3 & -1 \\ 7 & -1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 3-1 & 3+1 \\ 7-1 & 7+1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = U$$

= the original image

calculate basis images :

$$A^{*T} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

$$A^*_{0,0} = a_0^* \cdot a_0^{*T}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

a_0^* means 0th column of A^{*T}

$$= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$A^*_{0,1} = a_0^* \cdot a_1^{*T}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & \frac{-1}{2} \\ \frac{1}{2} & \frac{-1}{2} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

$$A^*_{1,0} = a_1^* \cdot a_0^{*T}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$A_{1,0}^{*} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{-1}{2} & \frac{-1}{2} \end{bmatrix} = \frac{1}{2.} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

$$A_{1,1}^{*} = a_1^{*} \cdot a_1^{*T}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{-1}{2} & \frac{1}{2} \end{bmatrix} = \underline{\underline{\frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}}}$$

Image enhancement in spatial domain

Basic Gray Level Transformations:

The study of image enhancement techniques is done by discussing gray-level transformation functions. These are among the simplest of all image enhancement techniques. The values of pixels, before and after processing, will be denoted by r and s, respectively. As indicated in the previous section, these values are related by an expression of the form s=T(r), where T is a transformation that maps a pixel value r into a pixel value s. Since we are dealing with digital quantities, values of the transformation function typically are stored in a one-dimensional array and the mappings from r to s are implemented via table lookups. For an 8-bit environment, a lookup table containing the values of T will have 256 entries. As an introduction to gray-level transformations, consider Fig. 1.1, which shows three basic types of functions used frequently for image enhancement: linear (negative and identity transformations), logarithmic (log and inverse-log transformations), and power-law (nth power and nth root transformations).The identity function is the trivial case in which output intensities are identical to input intensities. It is included in the graph only for completeness.

Image Negatives:

The negative of an image with gray levels in the range [0, L-1] is obtained by using the negative transformation shown in Fig.1.1, which is given by the expression

$$s = L - 1 - r.$$

Reversing the intensity levels of an image in this manner produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size.

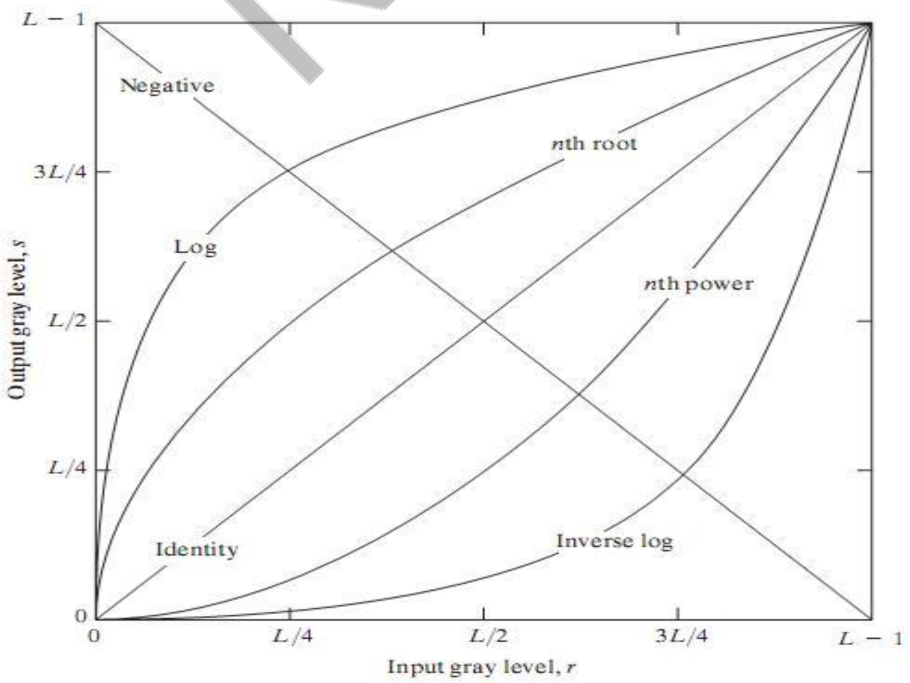


Fig.1.1 Some basic gray-level transformation functions used for image enhancement

Log Transformations:

The general form of the log transformation shown in Fig.1.1 is

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

where c is a constant, and it is assumed that r ≥ 0.The shape of the log curve in Fig. 1.1 shows that this transformation maps a narrow range of low gray-level values in the input image into a wider range of output levels.The opposite is true of higher values of input levels.We would use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values.The opposite is true of the inverse log transformation.

Any curve having the general shape of the log functions shown in Fig. 1.1 would accomplish this spreading/compressing of gray levels in an image. In fact, the power-law transformations discussed in the next section are much more versatile for this purpose than the log transformation. However, the log function has the important characteristic that it compresses the dynamic range of images with large variations in pixel values. A classic illustration of an application in which pixel values have a large dynamic range is the Fourier spectrum. At the moment,we are concerned only with the image characteristics of spectra. It is not unusual to encounter spectrum values that range from 0 to or higher.While processing numbers such as these presents no problems for a computer, image display systems generally will not be able to reproduce faithfully such a wide range of intensity values. The net effect is that a significant degree of detail will be lost in the display of a typical Fourier spectrum.

Power-Law Transformations:

Power-law transformations have the basic form

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

where c and g are positive constants. Sometimes Eq. is written as $s = c(r + \varepsilon)^{\gamma}$

to account for an offset (that is, a measurable output when the input is zero).However, offsets typically are an issue of display calibration and as a result they are normally ignored in Eq. Plots of s versus r for various values of g are shown in Fig. 1.2. As in the case of the log transformation, power-law curves with fractional values of g map a narrow range of dark input values into a wider range of output values,with the opposite being true for high-er values of input levels. Unlike the log function, however, we notice here a family of possible transformation curves obtained simply by varying γ. As expected, we see in Fig.1.2 that curves generated with values of g>1 have exactly the opposite effect as those generated with values of g<1. Finally, we note that Eq. reduces to the identity transformation when c = γ = 1. A variety of devices used for image capture, printing, and display respond according to a power law.By convention, the exponent in the power-law equation is referred to as gamma. The proces used to correct this power-law response phenomena is called gamma correction. For example, cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with exponents varying from approximately 1.8 to 2.5.With reference to the curve for g=2.5 in Fig.1.2, we see that such display systems would tend to produce images that are darker than intended.

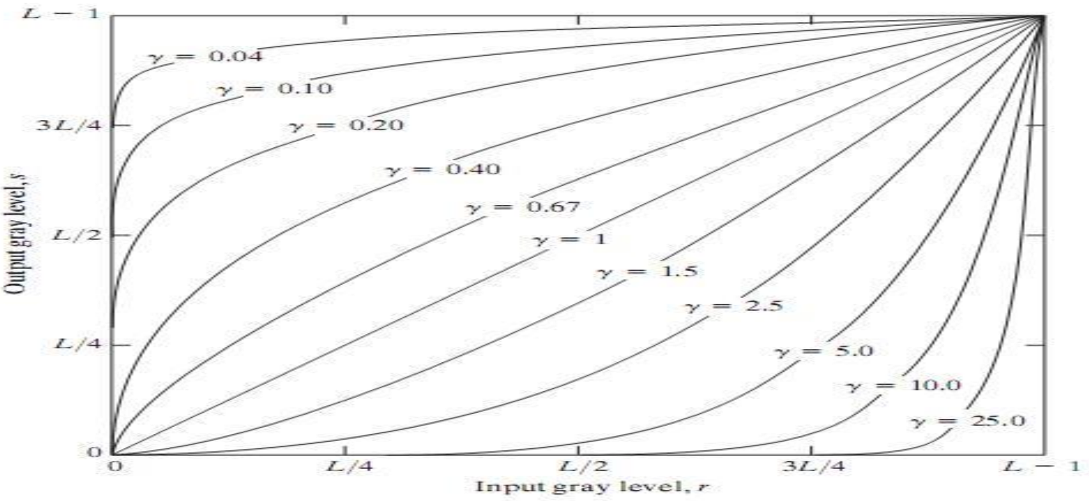


Fig.1.2 Plots of the equation $s = cr^{\gamma}$ for various values of γ (c=1 in all cases).

Piecewise-Linear Transformation Functions:

The principal advantage of piecewise linear functions over the types of functions we have discussed above is that the form of piecewise functions can be arbitrarily complex. In fact, as we will see shortly, a practical implementation of some important transformations can be formulated only as piecewise functions. The principal disadvantage of piecewise functions is that their specification requires considerably more user input.

Contrast stretching:

One of the simplest piecewise linear functions is a contrast-stretching transformation. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture during image acquisition.The idea behind contrast stretching is to increase the dynamic range of the gray levels in the image being processed.

Figure 1.3 (a) shows a typical transformation used for contrast stretching.

The locations of points (r_1 , s_1) and (r_2 , s_2) control the shape of the transformation

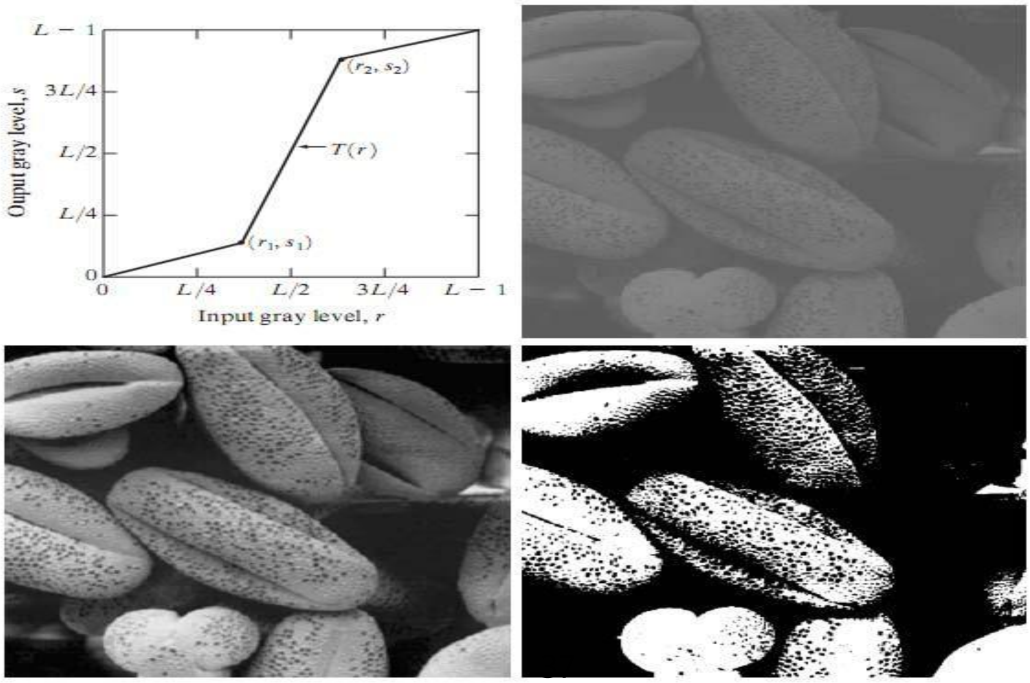


Fig.1.3 Contrast Stretching (a) Form of Transformation function (b) A low-contrast image (c) Result of contrast stretching (d) Result of thresholding.

function. If r1=s1 and r2=s2, the transformation is a linear function that produces no changes in gray levels. If r1=r2,s1=0 and s2=L-1, the transformation becomes a thresholding function that creates a binary image, as illustrated in Fig. 1.3 (b). 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. In general, r1 ≤ r2 and s1 ≤ s2 is assumed so that the function is single valued and monotonically increasing.This condition preserves the order of gray levels, thus preventing the creation of intensity artifacts in the processed image.

Figure 1.3 (b) shows an 8-bit image with low contrast. Fig. 1.3(c) shows the result of contrast stretching, obtained by setting (r_1 , s_1) = (r_{min} , 0) and (r_2 , s_2) = (r_{max} , L-1) where r_{min} and r_{max} denote the minimum and maximum gray levels in the image, respectively.Thus, the transformation function stretched the levels linearly from their original range to the full range [0, L-1]. Finally, Fig. 1.3 (d) shows the result of using the thresholding function defined previously,with r1 = r2 = m, the mean gray level in the image.The original image on which these results are based is a scanning electron microscope image of pollen,magnified approximately 700 times.

Gray-level slicing:

Highlighting a specific range of gray levels in an image often is desired. Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images.There are several ways of doing level slicing, but most of them are variations of two basic themes.One approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels.This transformation, shown in Fig. 1.4 (a), produces a binary image.The second approach, based on the transformation shown in Fig. 1.4 (b), brightens the desired range of gray levels but preserves the background and gray-level tonalities in the image. Figure 1.4(c) shows a gray-scale image, and Fig. 1.4 (d) shows the result of using the transformation in Fig. 1.4 (a).Variations of the two transformations shown in Fig. 1.4 are easy to formulate.

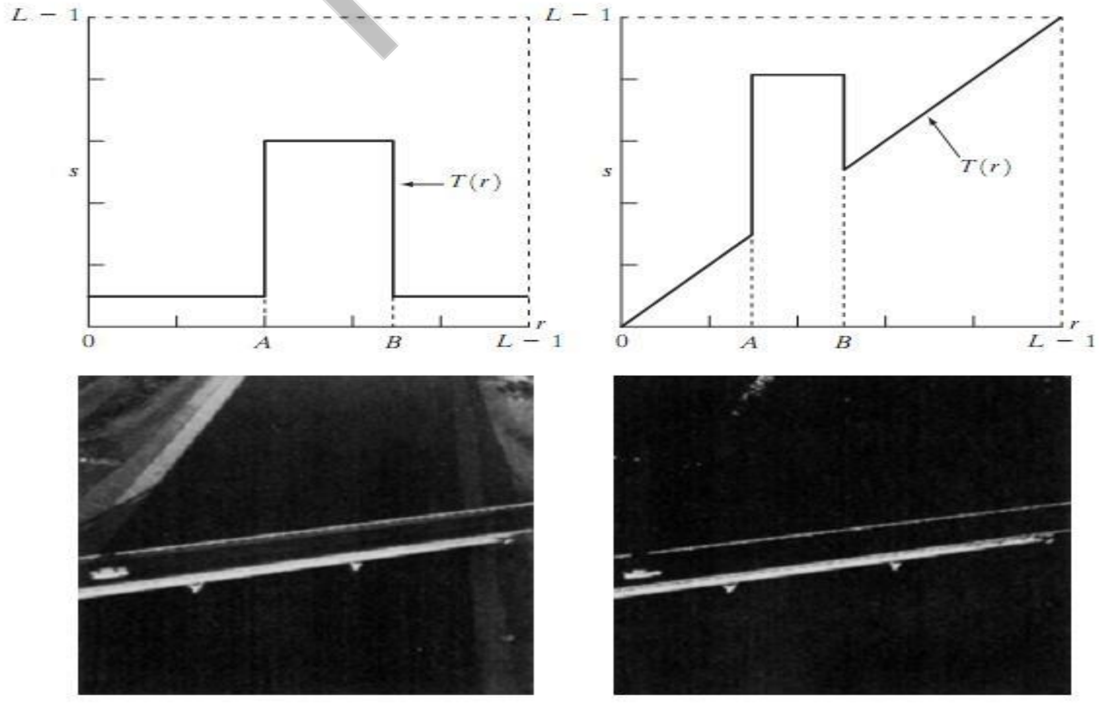


Fig.1.4 (a) This transformation highlights range [A, B] of gray levels and reduce all others to a constant level (b) This transformation highlights range [A, B] but preserves all other levels (c) An image (d) Result of using the transformation in (a).

Bit-plane slicing:

Instead of highlighting gray-level ranges, highlighting the contributionmade to total image appearance by specific bits might be desired. Suppose that each pixel in an image is represented by 8 bits. Imagine that the image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit plane 7 for the most significant bit. In terms of 8-bit bytes, plane 0

contains all the lowest order bits in the bytes comprising the pixels in the image and plane 7 contains all the high-order bits.Figure 1.5 illustrates these ideas, and Fig. 1.7 shows the various bit planes for the image shown in Fig.1.6 . Note that the higher-order bits (especially the top four) contain themajority of the visually significant data.The other bit planes contribute tomore subtle details in the image. Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image, a process that aids in determining the adequacy of the number of bits used to quantize each pixel.

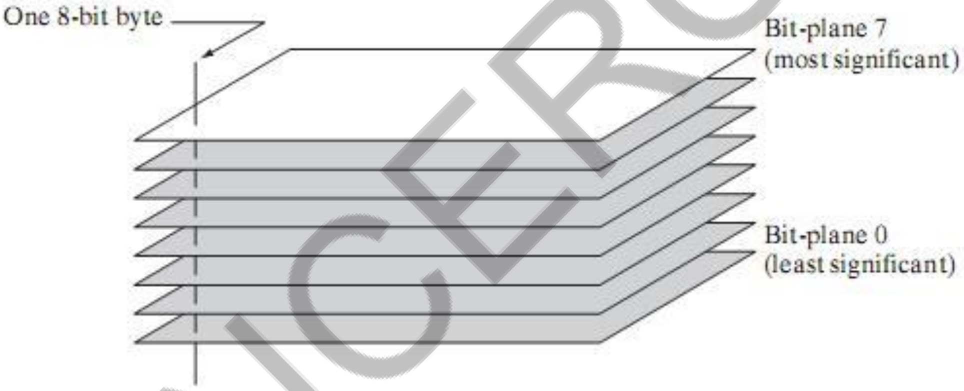


Fig.1.5 Bit-plane representation of an 8-bit image.

In terms of bit-plane extraction for an 8-bit image, it is not difficult to show that the (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding gray-level transformation function that (1) maps all levels in the image between 0 and 127 to one level (for example, 0); and (2) maps all levels between 129 and 255 to another (for example, 255).

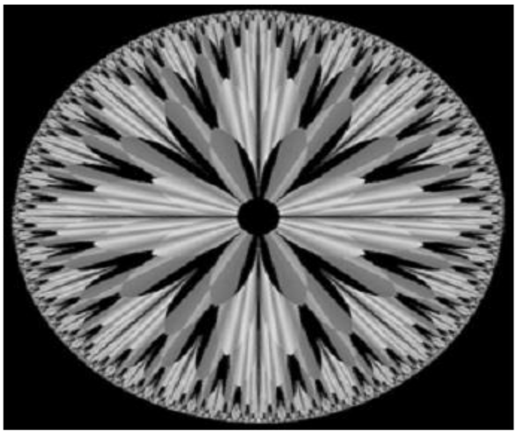


Fig.1.6 An 8-bit fractal image

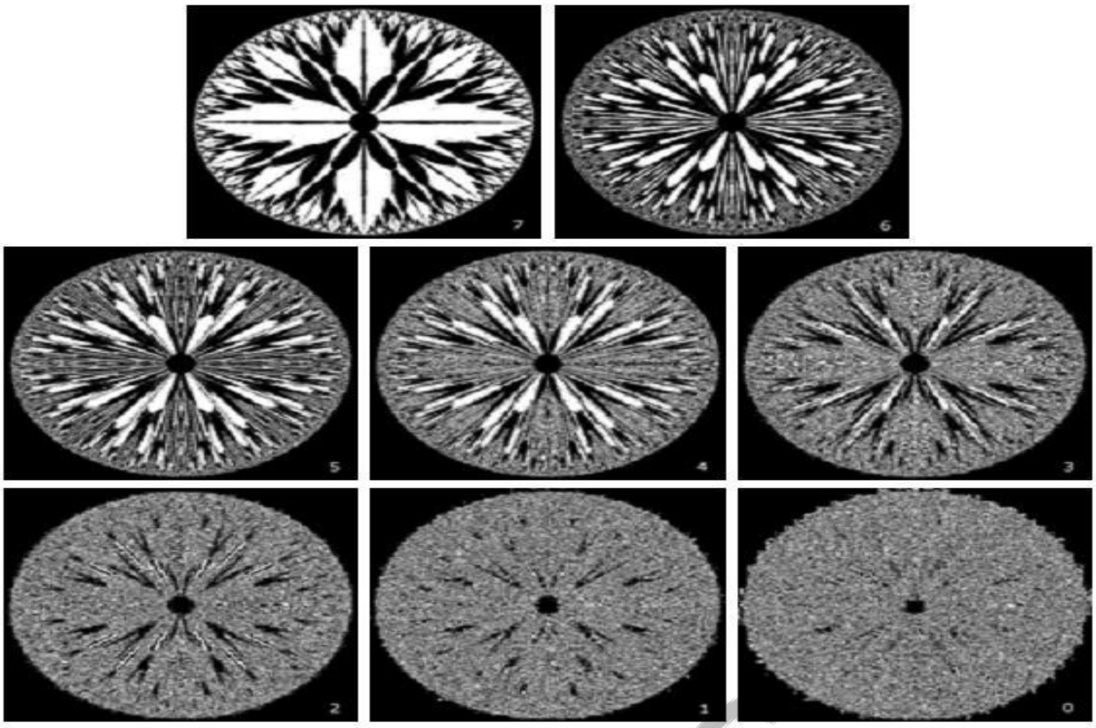


Fig.1.7 The eight bit planes of the image in Fig.1.6. The number at the bottom, right of each image identifies the bit plane.

objective of image enhancement. Define spatial domain.

The term spatial domain refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression

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

where f(x, y) is the input image, g(x, y) is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). In addition,T can operate on a set of input images, such as performing the pixel-by-pixel sum of K images for noise reduction.

The principal approach in defining a neighborhood about a point (x, y) is to use a square or rectangular subimage area centered at (x, y), as Fig.2.1 shows. The center of the subimage is moved from pixel to pixel starting, say, at the top left corner. The operator T is applied at each location (x, y) to yield the output, g, at that location.The process utilizes only the pixels in the area of the image spanned by the neighborhood.

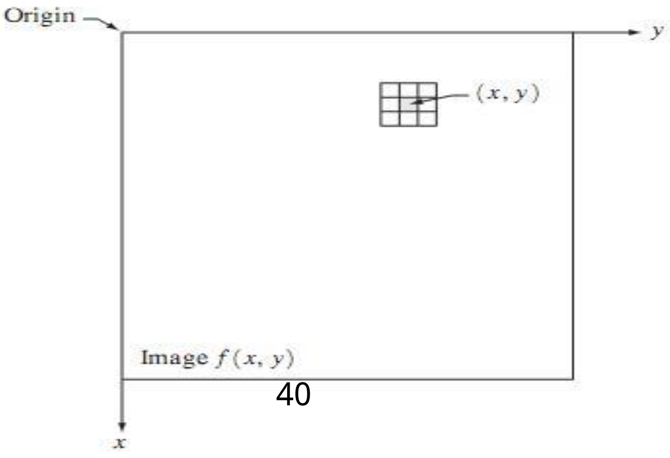


Fig.2.1 A 3*3 neighborhood about a point (x, y) in an image.

Although other neighborhood shapes, such as approximations to a circle, sometimes are used, square and rectangular arrays are by far the most predominant because of their ease of implementation. The simplest form of T is when the neighborhood is of size 1*1 (that is, a single pixel). In this case, g depends only on the value of f at (x, y), and T becomes a gray-level (also called an intensity or mapping) transformation function of the form

$$s = T(r)$$

where, for simplicity in notation, r and s are variables denoting, respectively, the gray level of f(x, y) and g(x, y) at any point (x, y). For example, if T(r) has the form shown in Fig. 2.2(a), the effect of this transformation would be to produce an image of higher contrast than the original by darkening the levels below m and brightening the levels above m in the original image. In this technique, known as contrast stretching, the values of r below m are compressed by the transformation function into a narrow range of s, toward black.The opposite effect takes place for values of r above m. In the limiting case shown in Fig. 2.2(b), T(r) produces a two-level (binary) image. A mapping of this form is called a thresholding function. Some fairly simple, yet powerful, processing approaches can be formulated with gray-level transformations. Because enhancement at any point in an image depends only on the gray level at that point, techniques in this category often are referred to as point processing.

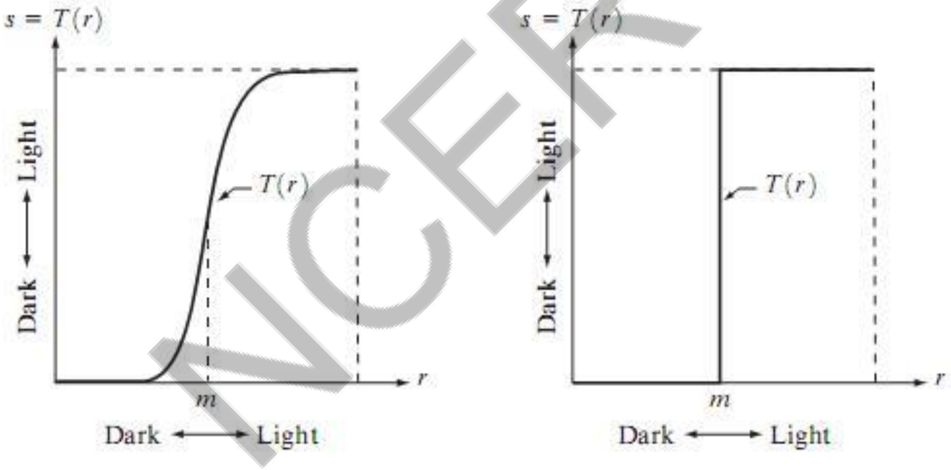


Fig.2.2 Graylevel transformation functions for contrast enhancement.

Larger neighborhoods allow considerably more flexibility. The general approach is to use a function of the values of f in a predefined neighborhood of (x, y) to determine the value of g at (x, y).One of the principal approaches in this formulation is based on the use of so-called masks

(also referred to as filters, kernels, templates, or windows). Basically, a mask is a small (say, 3*3) 2-D array, such as the one shown in Fig. 2.1, in which the values of the mask coefficients determine the nature of the process, such as image sharpening.

Histogram of a digital image.

Histogram Processing:

The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function $h(r_k)$ $= (n_k)$, where r_k is the kth gray level and n_k is the number of pixels in the image having gray level r_k. It is common practice to normalize a histogram by dividing each of its values by the total number of pixels in the image, denoted by n. Thus, a normalized histogram is given by for k=0,1,…… .,L-1. Loosely speaking, $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k. Note that the sum of all components of a normalized histogram is equal to 1.

Histograms are the basis for numerous spatial domain processing techniques.Histogram manipulation can be used effectively for image enhancement. Histograms are simple to calculate in software and also lend themselves to economic hardware implementations, thus making them a popular tool for real-time image processing.

As an introduction to the role of histogram processing in image enhancement, consider Fig. 3, which is the pollen image shown in four basic gray-level characteristics: dark, light, low contrast, and high contrast.The right side of the figure shows the histograms corresponding to these images. The horizontal axis of each histogram plot corresponds to gray level values, r_k.

The vertical axis corresponds to values of $h(r_k) = n_k$ or $p(r_k) = n_k/n$ if the values are normalized.Thus, as indicated previously, these histogram plots are simply plots of $h(r_k) = n_k$ versus r_k or $p(r_k) = n_k/n$ versus r_k.

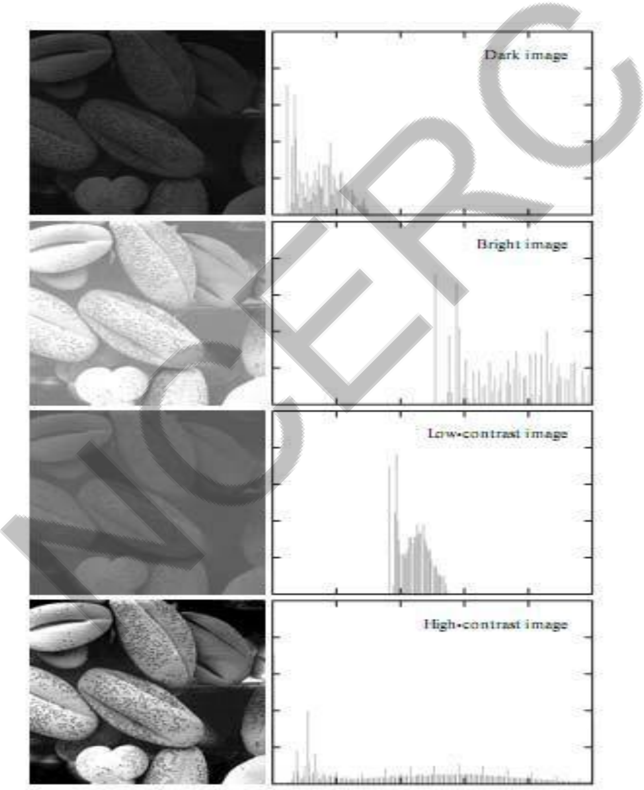


Fig.3 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms.

We note in the dark image that the components of the histogram are concentrated on the low (dark) side of the gray scale. Similarly, the components of the histogram of the bright image are biased toward the high side of the gray scale.An image with low contrast has a histogram that will be narrow and will be centered toward the middle of the gray scale. For a monochrome image this implies a dull,washed-out gray look. Finally,we see that the components of the histogram in the high-contrast image cover a broad range of the gray scale and, further, that the distribution of pixels is not too far from uniform,with very few vertical lines being much higher than the others. Intuitively, it is reasonable to conclude that an image whose pixels tend to occupy the entire range of possible gray levels and, in addition, tend to be distributed uniformly,will have an appearance of high contrast and will exhibit a large variety of gray tones. The net effect will be an image that shows a great deal of gray-level detail and has high dynamic range. It will be shown shortly that it is possible to develop a transformation function that can automatically achieve this effect, based only on information available in the histogram of the input image.

Histogram equalization.

Histogram Equalization:

Consider for a moment continuous functions, and let the variable r represent the gray levels of the image to be enhanced. We assume that r has been normalized to the interval [0, 1], with r=0 representing black and r=1 representing white. Later, we consider a discrete formulation and allow pixel values to be in the interval [0, L-1]. For any r satisfying the aforementioned conditions, we focus attention on transformations of the form

$$s = T(r) \qquad 0 \le r \le 1$$

that produce a level s for every pixel value r in the original image. For reasons that will become obvious shortly, we assume that the transformation function T(r) satisfies the following conditions:

(a) T(r) is single-valued and monotonically increasing in the interval $0 \le r \le 1$; and

(b) $0 \le T(r) \le 1$ for $0 \le r \le 1$.

The requirement in (a) that T(r) be single valued is needed to guarantee that the inverse transformation will exist, and the monotonicity condition preserves the increasing order from black to white in the output image.A transformation function that is not monotonically increasing could result in at least a section of the intensity range being inverted, thus producing some inverted gray levels in the output image. Finally, condition (b) guarantees that the output gray levels will be in the same range as the input levels. Figure 4.1 gives an example of a transformation function that satisfies these two conditions.The inverse transformation from s back to r is denoted

$$r = T^{-1}(s) \qquad 0 \le s \le 1.$$

It can be shown by example that even if T(r) satisfies conditions (a) and (b), it is possible that the corresponding inverse T^{-1} (s) may fail to be single valued.

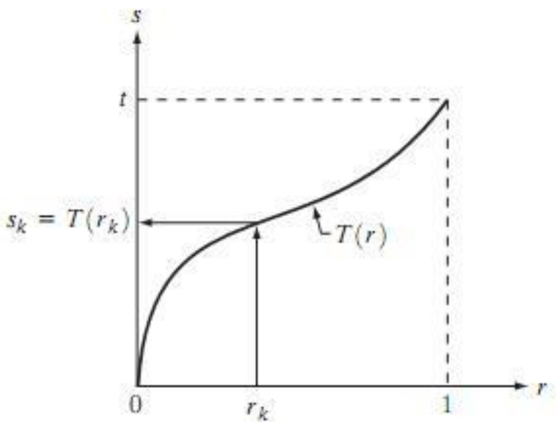


Fig.4.1 A gray-level transformation function that is both single valued and monotonically increasing.

The gray levels in an image may be viewed as random variables in the interval [0, 1].One of the most fundamental descriptors of a random variable is its probability density function (PDF).Let $p_r(r)$ and $p_s(s)$ denote the probability density functions of random variables r and s, respectively,where the subscripts on p are used to denote that p_r and p_s are different functions.A basic result from an elementary probability theory is that, if $p_r(r)$ and T(r) are known and T^{-1} (s) satisfies condition (a), then the probability density function $p_s(s)$ of the transformed variable s can be obtained using a rather simple formula:

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

Thus, the probability density function of the transformed variable, s, is determined by the gray-level PDF of the input image and by the chosen transformation function. A transformation function of particular importance in image processing has the form

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

where w is a dummy variable of integration.The right side of Eq. above is recognized as the cumulative distribution function (CDF) of random variable r. Since probability density functions are always positive, and recalling that the integral of a function is the area under the function, it follows that this transformation function is single valued and monotonically increasing, and, therefore, satisfies condition (a). Similarly, the integral of a probability density function for variables in the range [0, 1] also is in the range [0, 1], so condition (b) is satisfied as well.

Given transformation function T(r),we find $p_s(s)$ by applying Eq. We know from basic calculus (Leibniz’s rule) that the derivative of a definite integral with respect to its upper limit is simply the integrand evaluated at that limit. In other words,

$$\begin{aligned}\frac{ds}{dr} &= \frac{dT(r)}{dr}\\ &= \frac{d}{dr}\left[\int_0^r p_r(w)\,dw\right]\\ &= p_r(r).\end{aligned}$$

Substituting this result for dr/ds, and keeping in mind that all probability values are positive, yields

$$\begin{aligned}p_s(s) &= p_r(r)\left|\frac{dr}{ds}\right|\\ &= p_r(r)\left|\frac{1}{p_r(r)}\right|\\ &= 1 \qquad 0 \le s \le 1.\end{aligned}$$

Because $p_s(s)$ is a probability density function, it follows that it must be zero outside the interval [0, 1] in this case because its integral over all values of s must equal 1.We recognize the form of $p_s(s)$ as a uniform probability density function. Simply stated, we have demonstrated that performing the transformation function yields a random variable s characterized by a uniform probability density function. It is important to note from Eq. discussed above that T(r) depends on $p_r(r)$, but, as indicated by Eq. after it, the resulting $p_s(s)$ always is uniform, independent of the form of $p_r(r)$. For discrete values we deal with probabilities and summations instead of probability density functions and integrals. The probability of occurrence of gray level r in an image is approximated by

$$p_r(r_k) = \frac{n_k}{n} \qquad k = 0, 1, 2, \ldots, L - 1$$

where, as noted at the beginning of this section, n is the total number of pixels in the image, n_k is the number of pixels that have gray level r_k, and L is the total number of possible gray levels in the image.The discrete version of the transformation function given in Eq. is

$$\begin{aligned} s_k = T(r_k) &= \sum_{j=0}^{k} p_r(r_j) \\ &= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, \ldots, L - 1. \end{aligned}$$

Thus, a processed (output) image is obtained by mapping each pixel with level r_k in the input image into a corresponding pixel with level s_k in the output image. As indicated earlier, a plot of p_r (r_k) versus r_k is called a histogram. The transformation (mapping) is called histogram equalization or histogram linearization. It is not difficult to show that the transformation in Eq. satisfies conditions (a) and (b) stated previously. Unlike its continuos counterpart, it cannot be proved in general that this discrete transformation will produce the discrete equivalent of a uniform probability density function, which would be a uniform histogram.

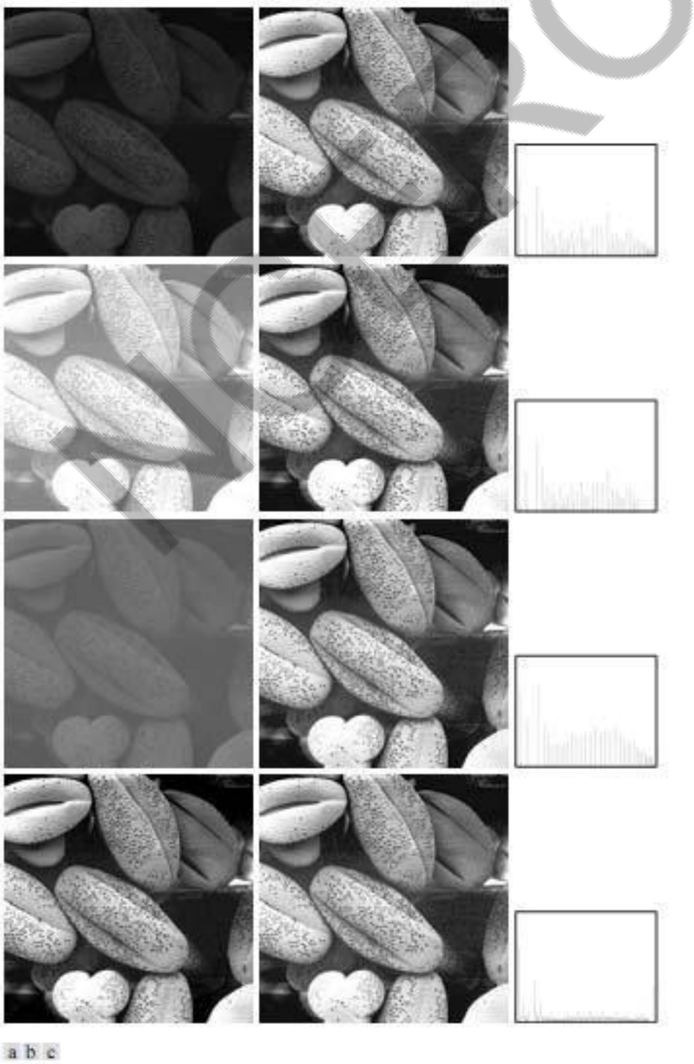


Fig.4.2 (a) Images from Fig.3 (b) Results of histogram equalization. (c) Corresponding histograms.

The inverse transformation from s back to r is denoted by

$$r_k = T^{-1}(s_k) \qquad k = 0, 1, 2, \dots, L - 1$$

Histogram specification.

Histogram Matching (Specification):

Histogram equalization automatically determines a transformation function that seeks to produce an output image that has a uniform histogram.When automatic enhancement is desired, this is a good approach because the results from this technique are predictable and the method is simple to implement. In particular, it is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have.The method used to generate a processed image that has a specified histogram is called histogram matching or histogram specification.

Development of the method:

Let us return for a moment to continuous gray levels r and z (considered continuous random variables), and let $p_r(r)$ and $p_z(z)$ denote their corresponding continuos probability density functions. In this notation, r and z denote the gray levels of the input and output (processed) images, respectively.We can estimate $p_r(r)$ from the given input image, while $p_z(z)$ is the specified probability density function that we wish the output image to have.

Let s be a random variable with the property

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

where w is a dummy variable of integration.We recognize this expression as the continuos version of histogram equalization. Suppose next that we define a random variable z with the property

$$G(z) = \int_0^z p_z(t)\, dt = s$$

where t is a dummy variable of integration. It then follows from these two equations that G(z)=T(r) and, therefore, that z must satisfy the condition

$$z = G^{-1}(s) = G^{-1}[T(r)].$$

The transformation T(r) can be obtained once $p_r(r)$ has been estimated from the input image. Similarly, the transformation function G(z) can be obtained because $p_z(z)$ is given. Assuming that G^{-1} exists and that it satisfies conditions (a) and (b) in the histogram equalization process, the above three equations show that an image with a specified probability density function can be obtained from an input image by using the following procedure:

(1) Obtain the transformation function T(r).

(2) To obtain the transformation function G(z).

(3) Obtain the inverse transformation function G^{-1}

(4) Obtain the output image by applying above Eq. to all the pixels in the input image.

The result of this procedure will be an image whose gray levels, z, have the specified probability density function $p_Z(z)$. Although the procedure just described is straightforward in principle, it is seldom possible in practice to obtain analytical expressions for T(r) and for $G^{-1.}$ Fortunately, this problem is simplified considerably in the case of discrete values.The price we pay is the same as in histogram equalization,where only an approximation to the desired histogram is achievable. In spite of this, however, some very useful results can be obtained even with crude approximations.

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$
$$= \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, \dots, L - 1$$

where n is the total number of pixels in the image, n_j is the number of pixels with gray level r_j, and L is the number of discrete gray levels. Similarly, the discrete formulation is obtained from the given histogram $p_Z (z_i)$, i=0, 1, 2,……, L-1, and has the form

$$v_k = G(z_k) = \sum_{i=0}^{k} p_z(z_i) = s_k \qquad k = 0, 1, 2, \dots, L - 1.$$

As in the continuos case, we are seeking values of z that satisfy this equation.The variable v_k was added here for clarity in the discussion that follows. Finally, the discrete version of the above Eqn. is given by

$$z_k = G^{-1}[T(r_k)] \qquad k = 0, 1, 2, \dots, L - 1$$

Or

$$z_k = G^{-1}(s_k) \qquad k = 0, 1, 2, \dots, L - 1.$$

Implementation:

We start by noting the following: (1) Each set of gray levels $\{r_j\}$, $\{s_j\}$, and $\{z_j\}$, j=0, 1, 2, p , L-1, is a one-dimensional array of dimension L X 1. (2) All mappings from r to s and from s to z are simple table lookups between a given pixel value and these arrays. (3) Each of the elements of these arrays, for example, s_k, contains two important pieces of information: The subscript k denotes the location of the element in the array, and s denotes the value at that location. (4) We need to be concerned only with integer pixel values. For example, in the case of an 8-bit image, L=256 and the elements of each of the arrays just mentioned are integers between 0 and 255.This implies that we now work with gray level values in the interval [0, L-1] instead of the normalized interval [0, 1] that we used before to simplify the development of histogram processing techniques.

In order to see how histogram matching actually can be implemented, consider Fig. 5(a), ignoring for a moment the connection shown between this figure and Fig. 5(c). Figure 5(a) shows a hypothetical discrete transformation function s=T(r) obtained from a given image. The first gray level in the image, r_1 , maps to s_1 ; the second gray level, r_2 , maps to s_2 ; the kth level r_k maps to s_k; and so on (the important point here is the ordered correspondence between these values). Each value s_j in the array is precomputed, so the process of mapping simply uses the actual value of a pixel as an index in an array to determine the corresponding value of s.This process is particularly easy because we are dealing with integers. For example, the s mapping for an 8-bit pixel with value 127 would be found in the 128th position in array $\{s_j\}$ (recall that we start at 0) out of the possible 256 positions. If we stopped here and mapped the value of each pixel of an input image by the

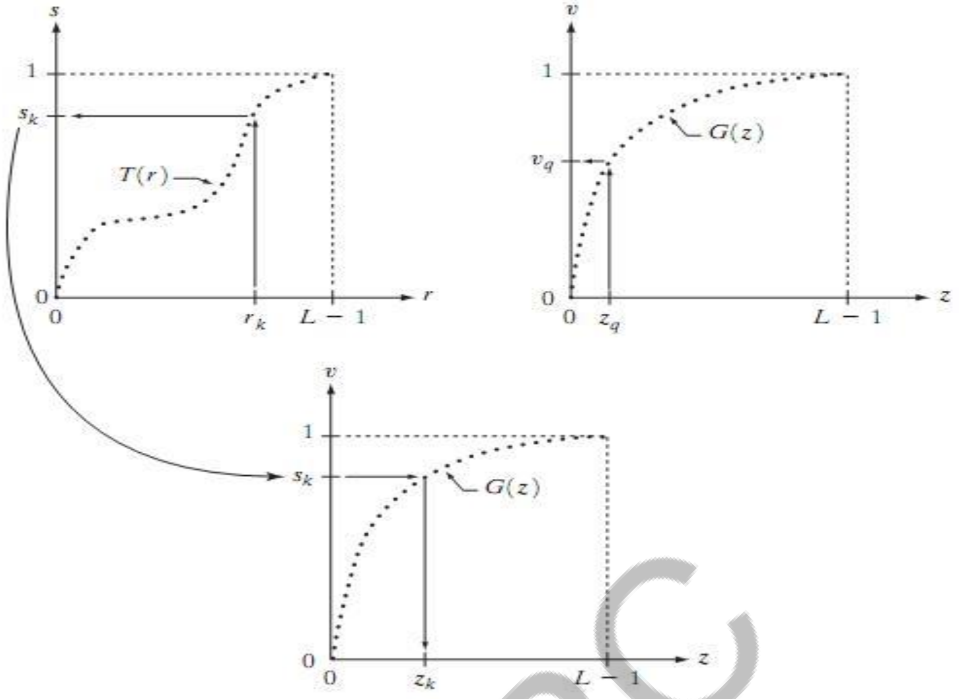


Fig.5. (a) Graphical interpretation of mapping from r_k to s_k via T(r). (b) Mapping of z_q to its corresponding value v_q via G(z) (c) Inverse mapping from s_k to its corresponding value of z_k.

method just described, the output would be a histogram-equalized image. In order to implement histogram matching we have to go one step further. Figure 5(b) is a hypothetical transformation function G obtained from a given histogram $p_z(z)$. For any z_q , this transformation function yields a corresponding value v_q. This mapping is shown by the arrows in Fig. 5(b). Conversely, given any value v_q, we would find the corresponding value z_q from G^{-1}. In terms of the figure, all this means graphically is that we would reverse the direction of the arrows to map v_q into its corresponding z_q. However, we know from the definition that v=s for corresponding subscripts, so we can use exactly this process to find the z_k corresponding to any value s_k that we computed previously from the equation $s_k = T(r_k)$ .This idea is shown in Fig.5(c).

Since we really do not have the z's (recall that finding these values is precisely the objective of histogram matching),we must resort to some sort of iterative scheme to find z from s.The fact that we are dealing with integers makes this a particularly simple process. Basically, because $v_k = s_k$, we have that the z's for which we are looking must satisfy the equation $G(z_k)=s_k$, or $(G(z_k)-s_k)=0$. Thus, all we have to do to find the value of z_k corresponding to s_k is to iterate on values of z such that this equation is satisfied for k=0,1,2,…...., L-1. We do not have to find the inverse of G because we are going to iterate on z. Since we are dealing with integers, the closest we can get to satisfying the equation $(G(z_k)-s_k)=0$ is to let $z_k= \hat{z}$ for each value of k, where $\hat{z}$ is the smallest integer in the interval [0, L-1] such that

$$\left(G(\hat{z}) - s_k\right) \geq 0 \qquad k = 0, 1, 2, \dots, L-1.$$

Given a value s_k, all this means conceptually in terms of Fig. 5(c) is that we would start with and increase it in integer steps until Eq is satisfied, at which point we let repeating this process for all values of k would yield all the required mappings from s to z, which constitutes the implementation of Eq. In practice, we would not have to start with each time because the values of sk are known to increase monotonically. Thus, for k=k+1, we would start with $\hat{z} = z_k$ and increment in integer values from there.

Local enhancement.

Local Enhancement:

The histogram processing methods discussed in the previous two sections are global, in the sense that pixels are modified by a transformation function based on the gray-level content of an entire image. Although this global approach is suitable for overall enhancement, there are cases in which it is necessary to enhance details over small areas in an image. The number of pixels in these areas may have negligible influence on the computation of a global transformation whose shape does not necessarily guarantee the desired local enhancement. The solution is to devise transformation functions based on the gray-level distribution—or other properties—in the neighborhood of every pixel in the image.

The histogram processing techniques are easily adaptable to local enhancement.The procedure is to define a square or rectangular neighborhood and move the center of this area from pixel to pixel. At each location, the histogram of the points in the neighborhood is computed and either a histogram equalization or histogram specification transformation function is obtained. This function is finally used to map the gray level of the

pixel centered in the neighborhood.The center of the neighborhood region is then moved to an adjacent pixel location and the procedure is repeated. Since only one new row or column of the neighborhood changes during a pixel-to-pixel translation of the region, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible. This approach has obvious a dvantages over repeatedly computing the histogram over all pixels in the neighborhood region each time the region is moved one pixel location.Another approach used some times to reduce computation is to utilize nonoverlapping regions, but this method usually produces an undesirable checkerboard effect.

Image subtaction.

Image Subtraction:

The difference between two images f(x, y) and h(x, y), expressed as

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

is obtained by computing the difference between all pairs of corresponding pixels from f and h. The key usefulness of subtraction is the enhancement of differences between images. The higher-order bit planes of an image carry a significant amount of visually relevant detail, while the lower planes contributemore to fine (often imperceptible) detail. Figure 7(a) shows the fractal image used earlier to illustrate the concept of bit planes. Figure 7(b) shows the result of discarding (setting to zero) the four least significant bit planes of the original image.The images are nearly identical visually, with the exception of a very slight drop in overall contrast due to less variability of the graylevel values in the image of Fig. 7(b).The pixel-by-pixel difference between these two images is shown in Fig. 7(c).The differences in pixel values are so small that the difference image appears nearly black when displayed on an 8-bit display. In order to bring out more detail,we can perform a contrast stretching transformation. We chose histogram equalization, but an appropriate power-law transformation would have done the job also. The result is shown in Fig. 7(d). This is a very useful image for evaluating the effect of setting to zero the lower-order planes.

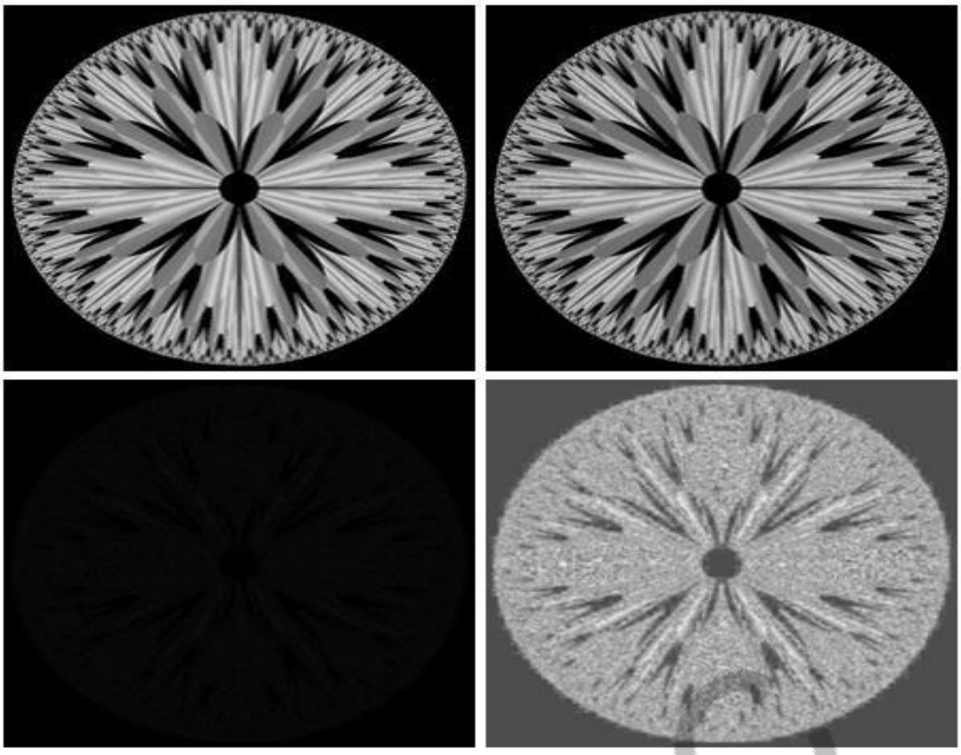


Fig.7 (a) Original fractal image (b) Result of setting the four lower-order bit planes to zero (c) Difference between (a) and(b) (d) Histogram equalized difference image.

One of the most commercially successful and beneficial uses of image subtraction is in the area of medical imaging called mask mode radiography. In this case h(x, y), the mask, is an X-ray image of a region of a patient’s body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source.The procedure consists of injecting a contrast medium into the patient’s bloodstream, taking a series of images of the same anatomical region as h(x, y), and subtracting this mask from the series of incoming images after injection of the contrast medium. The net effect of subtracting the mask from each sample in the incoming stream of TV images is that the areas that are different between f(x, y) and h(x, y) appear in the output image as enhanced detail. Because images can be captured at TV rates, this procedure in essence gives a movie showing how the contrast medium propagates through the various arteries in the area being observed.

Image averaging process.

Image Averaging:

Consider a noisy image g(x, y) formed by the addition of noise h(x, y) to an original image f(x,y); that is,

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

where the assumption is that at every pair of coordinates (x, y) the noise is uncorrelated and has zero average value.The objective of the following procedure is to reduce the noise content by adding a set of noisy images, $\{g_i (x, y)\}$. If the noise satisfies the constraints just stated, it can be shown that if an image $\bar{g}(x, y)$ is formed by averaging K different noisy images,

Then it follows that

$$E\{\bar{g}(x, y)\} = f(x, y)$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)}$$

Where $E\{\bar{g}(x, y)\}$ is the expected value of $\bar{g}$ and $\sigma^2_{\bar{g}(x,y)}$ and $\sigma^2_{\bar{\eta}(x,y)}$ are the variances of and $\bar{g}$ η, all at coordinates (x, y). The standard deviation at any point in the average image is

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)}.$$

As K increases, the above equations indicate that the variability (noise) of the pixel values at each location (x, y) decreases.Because $E\{\bar{g}(x, y)\} = f(x, y)$, this means that $\bar{g}(x, y)$ approaches f(x, y) as the number of noisy images used in the averaging process increases. In practice, the images $g_i(x, y)$ must be registered (aligned) in order to avoid the introduction of blurring and other artifacts in the output image.

Filtering in spatial domain.

Basics of Spatial Filtering:

Some neighborhood operations work with the values of the image pixels in the neighborhood and the corresponding values of a subimage that has the same dimensions as the neighborhood.The subimage is called a filter,mask, kernel, template, or window,with the first three terms being the most prevalent terminology.The values in a filter subimage are referred to as coefficients, rather than pixels. The concept of filtering has its roots in the use of the Fourier transform for signal processing in the so-called frequency domain. We use the term spatial filtering to differentiate this type of process from the more traditional frequency domain filtering.

The mechanics of spatial filtering are illustrated in Fig.9.1. The process consists simply of moving the filter mask from point to point in an image. At each point (x, y), the response of the filter at that point is calculated using a predefined relationship. The response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask. For the 3 x 3 mask shown in Fig. 9.1, the result (or response), R, of linear filtering with the filter mask at a point (x, y) in the image is

$$\begin{aligned} R = {} & w(-1,-1)f(x-1, y-1) + w(-1,0)f(x-1, y) + \cdots \\ & + w(0,0)f(x, y) + \cdots + w(1,0)f(x+1, y) + w(1,1)f(x+1, y+1), \end{aligned}$$

which we see is the sum of products of the mask coefficients with the corresponding pixels directly under the mask. Note in particular that the coefficient w(0, 0) coincides with image

value f(x, y), indicating that the mask is centered at (x, y) when the computation of the sum of products takes place. For a mask of size m x n,we assume that m=2a+1 and n=2b+1,where a and b are nonnegative integers.

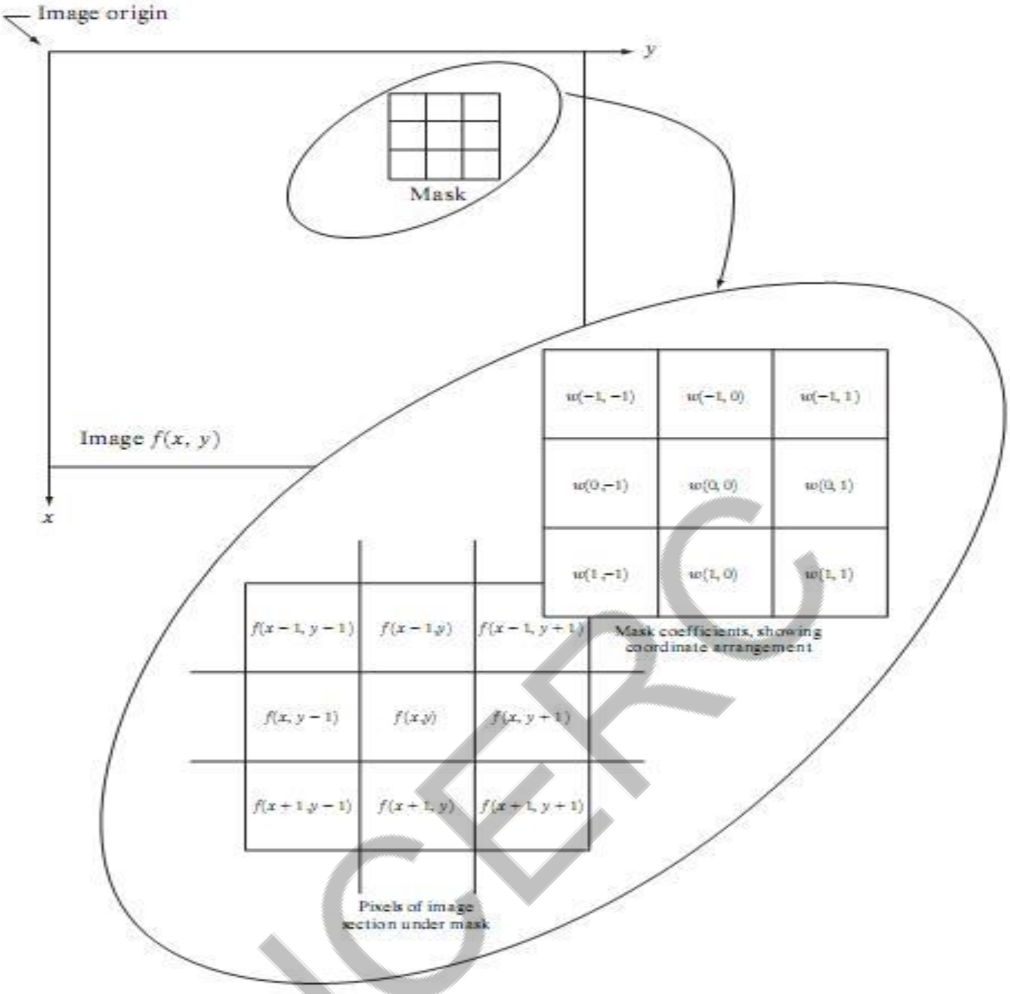


Fig.9.1 The mechanics of spatial filtering. The magnified drawing shows a 3X3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

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

where, from the previous paragraph, a=(m-1)/2 and b=(n-1)/2. To generate a complete filtered image this equation must be applied for x=0,1,2,……, M-1 and y=0,1,2,……, N-1. In this way, we are assured that the mask processes all pixels in the image. It is easily verified when m=n=3 that this expression reduces to the example given in the previous paragraph.

The process of linear filtering is similar to a frequency domain concept called convolution. For this reason, linear spatial filtering often is referred to as “convolving a mask with an image.” Similarly, filter masks are sometimes called convolution masks. The term convolution kernel also is in common use. When interest lies on the response, R, of an m x n mask at any point (x,y), and not on the mechanics of implementing mask convolution, it is common practice to simplify the notation by using the following expression:

where the w’s are mask coefficients, the z’s are the values of the image graylevels corresponding to those coefficients, and mn is the total number of coefficients in the mask. For the 3 x 3 general mask shown in Fig.9.2 the response at any point (x, y) in the image is given by

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \dots w_9 z_9 \\ &= \sum_{i=1}^{9} w_i z_i. \end{aligned}$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Fig.9.2 Another representation of a general 3 x 3 spatial filter mask.

An important consideration in implementing neighborhood operations for spatial filtering is the issue of what happens when the center of the filter approaches the border of the image.Consider for simplicity a square mask of size n x n.At least one edge of such a mask will coincide with the border of the image when the center of the mask is at a distance of (n-1)/2 pixels away from the border of the image. If the center of the mask moves any closer to the border, one or more rows or columns of the mask will be located outside the image plane.There are several ways to handle this situation.The simplest is to limit the excursions of the center of the mask to be at a distance no less than (n-1)/2 pixels from the border. The resulting filtered image will be smaller than the original, but all the pixels in the filtered imaged will have been processed with the full mask. If the result is required to be the same size as the original, then the approach typically employed is to filter all pixels only with the section of the mask that is fully contained in the image.With this approach, there will be bands of pixels near the border that will have been processed with a partial filter mask.Other approaches include “padding” the image by adding rows and columns of 0’s (or other constant gray level), or padding by replicating rows or columns.The padding is then stripped off at the end of the process.

This keeps the size of the filtered image the same as the original, but the values of the padding will have an effect near the edges that becomes more prevalent as the size of the mask increases.The only way to obtain a perfectly filtered result is to accept a somewhat smaller filtered image by limiting the excursions of the center of the filter mask to a distance no less than (n-1)/2 pixels from the border of the original image.

Smoothing Spatial filters.

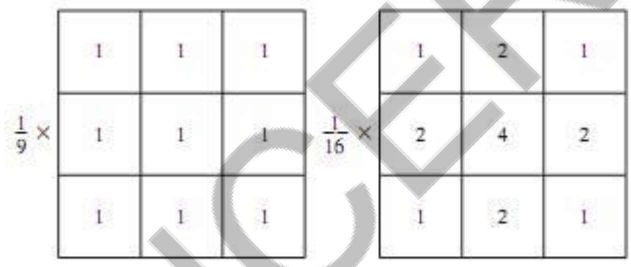
Smoothing Spatial Filters:

Smoothing filters are used for blurring and for noise reduction. Blurring is used in preprocessing steps, such as removal of small details from an image prior to (large) object extraction, and bridging of small gaps in lines or curves. Noise reduction can be accomplished by blurring with a linear filter and also by non-linear filtering.

(1) Smoothing Linear Filters:

The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called averaging filters. The idea behind smoothing filters is straightforward.By replacing the value of every pixel

in an image by the average of the gray levels in the neighborhood defined by the filter mask, this process results in an image with reduced “sharp” transitions in gray levels. Because random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing is noise reduction.However, edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels, so averaging filters have the undesirable side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result from using an insufficient number of gray levels.



$$\frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}\qquad \frac{1}{16}\times\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Fig.10.1 Two 3 x 3 smoothing (averaging) filter masks.The constant multiplier in front of each mask is equal to the sum of the values of its coefficients, as is required to compute an average.

A major use of averaging filters is in the reduction of “irrelevant” detail in an image. By “irrelevant”we mean pixel regions that are small with respect to the size of the filter mask.

Figure 10.1 shows two 3 x 3 smoothing filters. Use of the first filter yields the standard average of the pixels under the mask.This can best be seen by substituting the coefficients of the mask in

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i,$$

which is the average of the gray levels of the pixels in the 3 x 3 neighborhood defined by the mask.Note that, instead of being 1/9, the coefficients of the filter are all 1’s.The idea here is that it is computationally more efficient to have coefficients valued 1. At the end of the filtering process the entire image is divided by 9. An m x n mask would have a normalizing constant equal to 1/mn.

A spatial averaging filter in which all coefficients are equal is sometimes called a box filter.

The second mask shown in Fig.10.1 is a little more interesting. This mask yields a so-called weighted average, terminology used to indicate that pixels are multiplied by different coefficients, thus giving more importance (weight) to some pixels at the expense of others. In the mask shown in Fig. 10.1(b) the pixel at the center of the mask is multiplied by a higher value than any other, thus giving this pixel more importance in the calculation of the average.The other pixels are inversely weighted as a function of their distance from the center of the mask. The diagonal terms are further away from the center than the orthogonal neighbors (by a factor of √2) and, thus, are weighed less than these immediate neighbors of the center pixel. The basic strategy behind weighing the center point the highest and then reducing the value of the coefficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the smoothing process. We could have picked other weights to accomplish the same general objective. However, the sum of all the coefficients in the mask of Fig. 10.1(b) is equal to 16, an attractive feature for computer implementation because it has an integer power of 2. In practice, it is difficult in general to see differences between images smoothed by using either of the masks in Fig. 10.1, or similar arrangements, because the area these masks span at any one location in an image is so small.

The general implementation for filtering an M x N image with a weighted averaging filter of size m x n (m and n odd) is given by the expression

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

.

(2) Order-Statistics Filters:

Order-statistics filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result. The best-known example in this category is the median filter, which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median). Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of impulse noise, also called salt-and-pepper noise because of its appearance as white and black dots superimposed on an image.

The median, ε, of a set of values is such that half the values in the set are less than or equal to ε, and half are greater than or equal to ε. In order to perform median filtering at a point in an image, we first sort the values of the pixel in question and its neighbors, determine their median, and assign this value to that pixel. For example, in a 3 x 3 neighborhood the median is the 5th largest value, in a 5 x 5 neighborhood the 13th largest value, and so on. When several values in a neighborhood are the same, all equal values are grouped. For example, suppose that a 3 x 3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus, the principal function of median filters is to force points with distinct gray levels to be more like their neighbors. In fact, isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $n^{2\ /}$ 2 (one-half the filter area), are eliminated by an n x n median filter. In this case "eliminated" means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.

Gradiant and the Laplacian in image enhancement.

Use of Second Derivatives for Enhancement–The Laplacian:

The approach basically consists of defining a discrete formulation of the second-order derivative and then constructing a filter mask based on that formulation. We are interested in isotropic filters, whose response is independent of the direction of the discontinuities in the image to which the filter is applied. In other words, isotropic filters are rotation invariant, in the sense that rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.

Development of the method:

It can be shown (Rosenfeld and Kak [1982]) that the simplest isotropic derivative operator is the Laplacian, which, for a function (image) f(x, y) of two variables, is defined as

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

Because derivatives of any order are linear operations, the Laplacian is a linear operator. In order to be useful for digital image processing, this equation needs to be expressed in discrete form. There are several ways to define a digital Laplacian using neighborhoods. digital second.Taking into account that we now have two variables, we use the following notation for the partial second-order derivative in the x-direction:

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

and, similarly in the y-direction, as

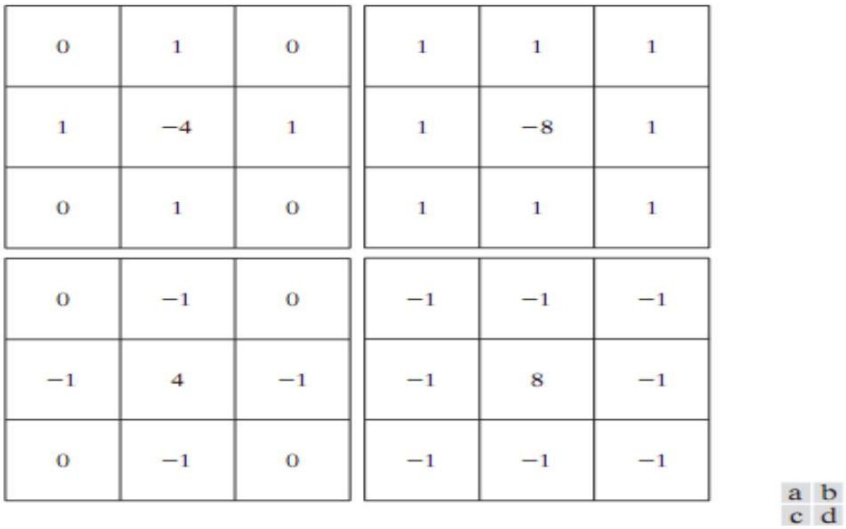
$$\frac{\partial^2 f}{\partial^2 y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

The digital implementation of the two-dimensional Laplacian in Eq. is obtained by summing these two components

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

This equation can be implemented using the mask shown in Fig.11.1(a), which gives an isotropic result for rotations in increments of 90°.

The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms to Eq., one for each of the two diagonal directions.The form of each new term is the same as either Eq.



0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Fig.11.1. (a) Filter mask used to implement the digital Laplacian (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

but the coordinates are along the diagonals. Since each diagonal term also contains a –2f(x, y) term, the total subtracted from the difference terms now would be –8f(x, y). The mask used to implement this new definition is shown in Fig.11.1(b). This mask yields isotropic results for increments of 45°. The other two masks shown in Fig. 11 also are used frequently in practice.

They are based on a definition of the Laplacian that is the negative of the one we used here. As such, they yield equivalent results, but the difference in sign must be kept in mind when combining (by addition or subtraction) a Laplacian-filtered image with another image.

Because the Laplacian is a derivative operator, its use highlights gray-level discontinuities in an image and deemphasizes regions with slowly varying gray levels.This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background.Background features can be “recovered” while still preserving the sharpening effect of the Laplacian operation simply by adding the original and Laplacian images. As noted in the previous paragraph, it is important to keep in mind which definition of the Laplacian is used. If the definition used has a negative center coefficient, then we subtract, rather

than add, the Laplacian image to obtain a sharpened result.Thus, the basic way in which we use the Laplacian for image enhancement is as follows:

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is positive.} \end{cases}$$

Use of First Derivatives for Enhancement—The Gradient:

First derivatives in image processing are implemented using the magnitude of the gradient. For a function f(x, y), the gradient of f at coordinates (x, y) is defined as the two-dimensional column vector

The magnitude of this vector is given by

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

The components of the gradient vector itself are linear operators, but the magnitude of this vector obviously is not because of the squaring and square root operations. On the other hand, the partial derivatives are not rotation invariant (isotropic), but the magnitude of the gradient vector is. Although it is not strictly correct, the magnitude of the gradient vector often is referred to as the gradient.

The computational burden of implementing over an entire image is not trivial, and it is common practice to approximate the magnitude of the gradient by using absolute values instead of squares and square roots:

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

This equation is simpler to compute and it still preserves relative changes in gray levels, but the isotropic feature property is lost in general. However, as in the case of the Laplacian, the isotropic properties of the digital gradient defined in the following paragraph are preserved only for a limited number of rotational increments that depend on the masks used to approximate the derivatives. As it turns out, the most popular masks used to approximate the gradient give the same result only for vertical and horizontal edges and thus the isotropic properties of the gradient are preserved only for multiples of 90°.

As in the case of the Laplacian, we now define digital approximations to the preceding equations, and from there formulate the appropriate filter masks. In order to simplify the discussion that follows, we will use the notation in Fig. 11.2 (a) to denote image points in a 3 x 3 region. For example, the center point, z5 , denotes f(x, y), z1 denotes f(x-1, y-1), and so on. The simplest approximations to a first-order derivative that satisfy the conditions stated in that section are $G_x = (z_8 - z_5)$ and $G_y = (z_6 - z_5)$. Two other definitions proposed by Roberts [1965] in the early development of digital image processing use cross differences:

$$G_x = (z_9 - z_5) \quad \text{and} \quad G_y = (z_8 - z_6).$$

we compute the gradient as

$$\nabla f = \left[(z_9 - z_5)^2 + (z_8 - z_6)^2\right]^{1/2}$$

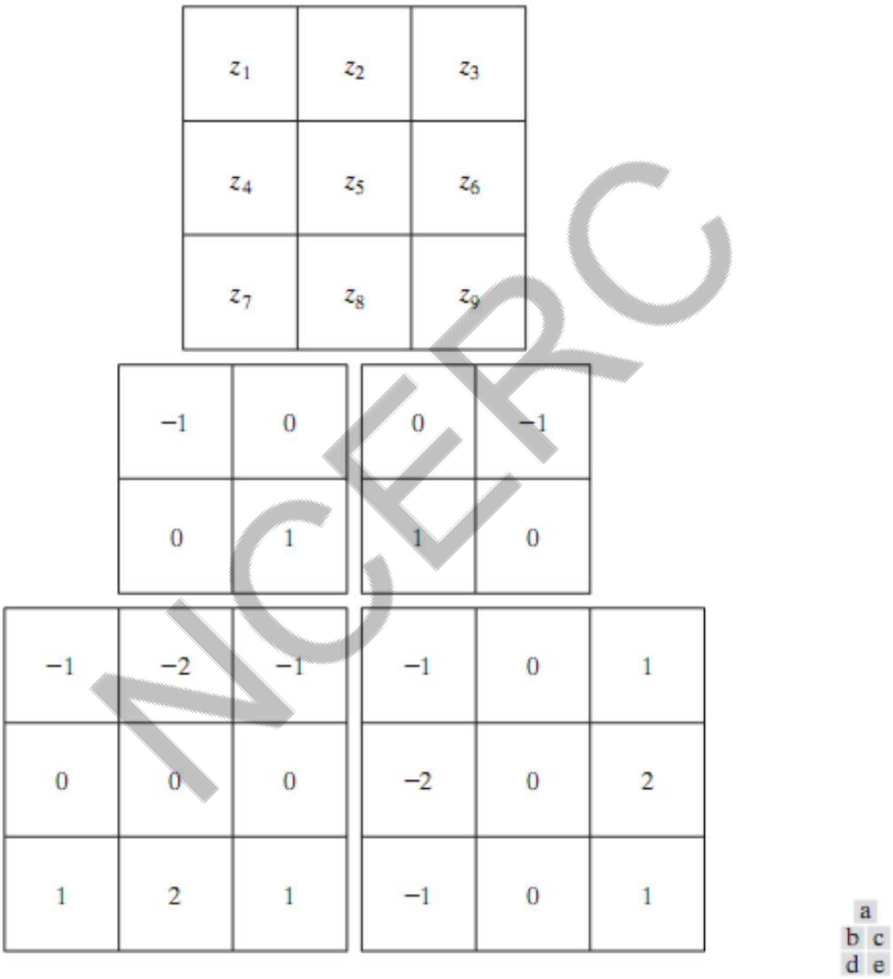
If we use absolute values, then substituting the quantities in the equations gives us the following approximation to the gradient:

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|.$$

This equation can be implemented with the two masks shown in Figs. 11.2 (b) and(c). These masks are referred to as the Roberts cross-gradient operators. Masks of even size are awkward to implement. The smallest filter mask in which we are interested is of size 3 x 3.An approximation using absolute values, still at point z_5 , but using a 3*3 mask, is

$$\begin{aligned}\nabla f \approx & \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| \\ & + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|.\end{aligned}$$

The difference between the third and first rows of the 3 x 3 image region approximates the derivative in the x-direction, and the difference between the third and first columns approximates the derivative in the y-direction. The masks shown in Figs. 11.2 (d) and (e), called the Sobel operators. The idea behind using a weight value of 2 is to achieve some smoothing by giving more importance to the center point. Note that the coefficients in all the masks shown in Fig. 11.2 sum to 0, indicating that they would give a response of 0 in an area of constant gray level, as expected of a derivative operator.



z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Fig.11.2 A 3 x 3 region of an image (the z's are gray-level values) and masks used to compute the gradient at point labeled z_5 . All masks coefficients sum to zero, as expected of a derivative operator.

Frequency domain techniques of image enhancement

Enhancement In Frequency Domain:

The frequency domain methods of image enhancement are based on convolution theorem. This is represented as,

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

Where.

$g(x, y)$ = Resultant image
$h(x, y)$ = Position invariant operator
$f(x, y)$= Input image

The Fourier transform representation of equation above is,

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

The function H (u, v) in equation is called transfer function. It is used to boost the edges of input image f (x, y) to emphasize the high frequency components.

The different frequency domain methods for image enhancement are as follows.

2. Contrast stretching.
3. Clipping and thresholding.
4. Digital negative.
5. Intensity level slicing and
6. Bit extraction.

1. Contrast Stretching:

Due to non-uniform lighting conditions, there may be poor contrast between the background and the feature of interest. Figure 1.1 (a) shows the contrast stretching transformations.

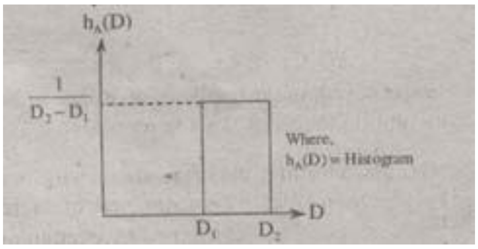


Fig.1.1 (a) Histogram of input image

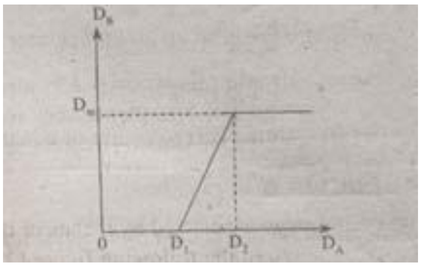


Fig.1.1 (b) Linear Law

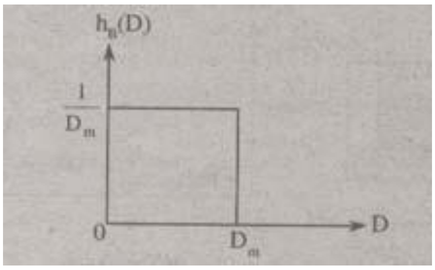


Fig.1.1 (c) Histogram of the transformed image

These stretching transformations are expressed as

In the area of stretching the slope of transformation is considered to be greater than unity. The parameters of stretching transformations i.e., a and b can be determined by examining the histogram of the image.

2. Clipping and Thresholding:

Clipping is considered as the special scenario of contrast stretching. It is the case in which the parameters are $\alpha = \gamma = 0$. Clipping is more advantageous for reduction of noise in input signals of range [a, b].

Threshold of an image is selected by means of its histogram. Let us take the image shown in the following figure 1.2.

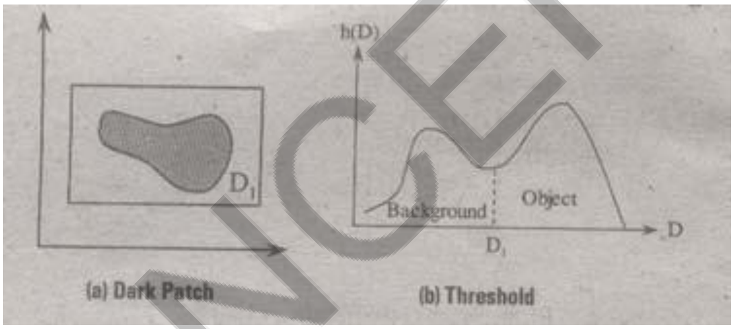


Fig. 1.2

The figure 1.2 (b) consists of two peaks i.e., background and object. At the abscissa of histogram minimum (D1) the threshold is selected. This selected threshold (D1) can separate background and object to convert the image into its respective binary form. The thresholding transformations are shown in figure 1.3.

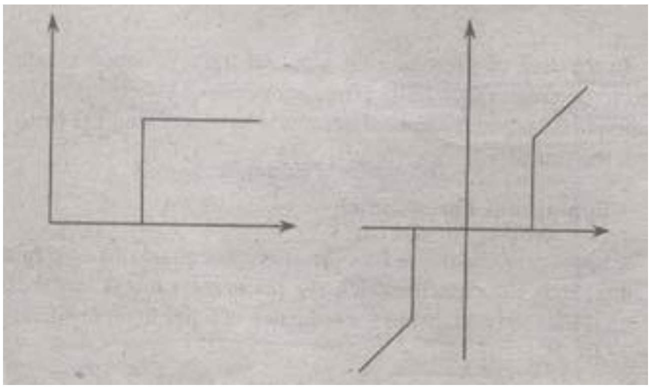


Fig.1.3

3. Digital Negative:

The digital negative of an image is achieved by reverse scaling of its grey levels to the transformation. They are much essential in displaying of medical images.

A digital negative transformation of an image is shown in figure 1.4.

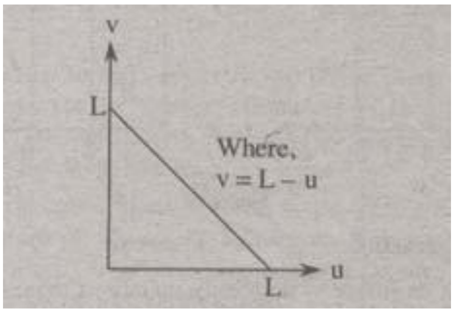


Fig.1.4

4. Intensity Level Slicing:

The images which consist of grey levels in between intensity at background and other objects require to reduce the intensity of the object. This process of changing intensity level is done with the help of intensity level slicing. They are expressed as

$$V = \begin{cases} L, & a \le u \le b \\ 0, & \text{elsewhere} \end{cases} \quad \text{without background}$$

$$\text{And} \quad V = \begin{cases} L, & a \le u \le b \\ u, & \text{elsewhere} \end{cases} \quad \text{with background}$$

The histogram of input image and its respective intensity level slicing is shown in the figure 1.5.

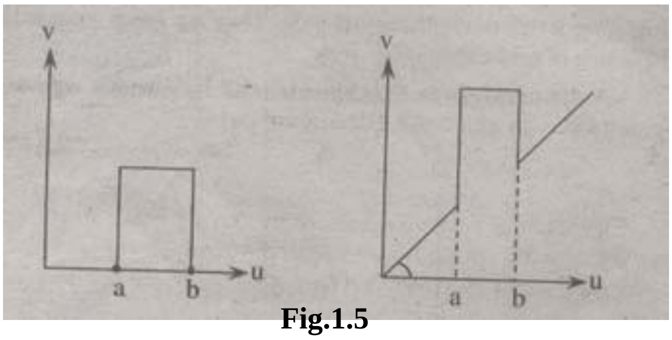


Fig.1.5

When an image is uniformly quantized then, the nth most significant bit can be extracted and displayed.

$$\text{Let, } u = k_1 2^{B-1} + k_2 2^{B-2} + \ldots\ldots\ldots\ldots\ldots + k_{B-1} 2 + k_B$$

Then, the output is expressed as

$$V = \begin{cases} L, & \text{for } k_n \\ 0, & \text{elsewhere} \end{cases}$$

Difference between spatial domain and frequency domain enhancement techniques.

The spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. Frequency domain processing techniques are based on modifying the Fourier transform of an image.

The term spatial domain refers to the aggregate of pixels composing an image and spatial domain methods are procedures that operate directly on these pixels. Image processing function in the spatial domain may he expressed as.

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

Where

f(x, y) is the input image
g(x, y) is the processed image and
T is the operator on f defined over some neighborhood values of (x, y).

Frequency domain techniques are based on convolution theorem. Let g(x, y) be the image formed by the convolution of an image f(x, y) and linear position invariant operation h(x, y) i.e.,

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

Applying convolution theorem

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

Where G, H and F are the Fourier transforms of g, h and f respectively. In the terminology of linear system the transform H (u, v) is called the transfer function of the process. The edges in f(x, y) can he boosted by using H (u, v) to emphasize the high frequency components of F (u, v).

Ideal Low Pass Filter (ILPF) in frequency domain.

Lowpass Filter:

The edges and other sharp transitions (such as noise) in the gray levels of an image contribute significantly to the high-frequency content of its Fourier transform. Hence blurring (smoothing) is achieved in the frequency domain by attenuating us the transform of a given image.

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

where F (u, v) is the Fourier transform of an image to be smoothed. The problem is to select a filter transfer function H (u, v) that yields G (u, v) by attenuating the high-frequency components of F (u, v). The inverse transform then will yield the desired smoothed image g (x, y).

Ideal Filter:

A 2-D ideal lowpass filter (ILPF) is one whose transfer function satisfies the relation

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

where D is a specified nonnegative quantity, and D(u, v) is the distance from point (u, v) to the origin of the frequency plane; that is,

$$D(u, v) = (u^2 + v^2)^{1/2}.$$

Figure 3 (a) shows a 3-D perspective plot of H (u, v) u a function of u and v. The name ideal filter indicates that oil frequencies inside a circle of radius

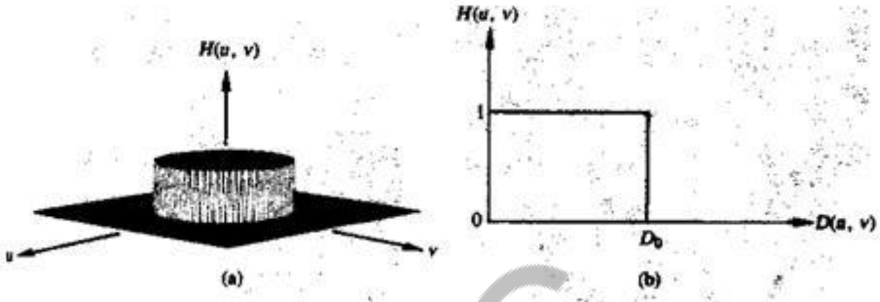


Fig.3 (a) Perspective plot of an ideal lowpass filter transfer function; (b) filter cross section.

Do are passed with no attenuation, whereas all frequencies outside this circle are completely attenuated.

The lowpass filters are radially symmetric about the origin. For this type of filter, specifying a cross section extending as a function of distance from the origin along a radial line is sufficient, as Fig. 3 (b) shows. The complete filter transfer function can then be generated by rotating the cross section 360 about the origin. Specification of radially symmetric filters centered on the N x N frequency square is based on the assumption that the origin of the Fourier transform has been centered on the square.

For an ideal lowpass filter cross section, the point of transition between H(u, v) = 1 and H(u, v) = 0 is often called the cutoff frequency. In the case of Fig.3 (b), for example, the cutoff frequency is Do. As the cross section is rotated about the origin, the point Do traces a circle giving a locus of cutoff frequencies, all of which are a distance Do from the origin. The cutoff frequency concept is quite useful in specifying filter characteristics. It also serves as a common base for comparing the behavior of different types of filters.

The sharp cutoff frequencies of an ideal lowpass filter cannot be realized with electronic components, although they can certainly be simulated in a computer.

Butterworth lowpass filter with a suitable example.

Butterworth filter:

The transfer function of the Butterworth lowpass (BLPF) of order n and with cutoff frequency locus at a distance Do, from the origin is defined by the relation

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

A perspective plot and cross section of the BLPF function are shown in figure 4.

Fig.4 (a) A Butterworth lowpass filter (b) radial cross section for n = 1.

Unlike the ILPF, the BLPF transfer function does not have a sharp discontinuity that establishes a clear cutoff between passed and filtered frequencies. For filters with smooth transfer functions, defining a cutoff frequency locus at points for which H (u, v) is down to a certain fraction of its maximum value is customary. In the case of above Eq. H (u, v) = 0.5 (down 50 percent from its maximum value of 1) when D (u, v) = Do. Another value commonly used is 1/√2 of the maximum value of H (u, v). The following simple modification yields the desired value when D (u, v) = Do:

$$H(u, v) = \frac{1}{1 + [\sqrt{2} - 1][D(u, v)/D_0]^{2n}} = \frac{1}{1 + 0.414[D(u, v)/D_0]^{2n}}$$

Ideal High Pass Filter and Butterworth High Pass filter.

High pass Filtering:

An image can be blurred by attenuating the high-frequency components of its Fourier transform. Because edges and other abrupt changes in gray levels are associated with high-frequency components, image sharpening can be achieved in the frequency domain by a high pass filtering process, which attenuates the low-frequency components without disturbing high-frequency information in the Fourier transform.

Ideal filter:

2-D ideal high pass filter (IHPF) is one whose transfer function satisfies the relation

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

where Do is the cutoff distance measured from the origin of the frequency plane. Figure 5.1 shows a perspective plot and cross section of the IHPF function. This filter is the opposite of the ideal lowpass filter, because it completely attenuates all frequencies inside a circle of radius Do while passing, without attenuation, all frequencies outside the circle. As in the case of the ideal lowpass filler, the IHPF is not physically realizable.

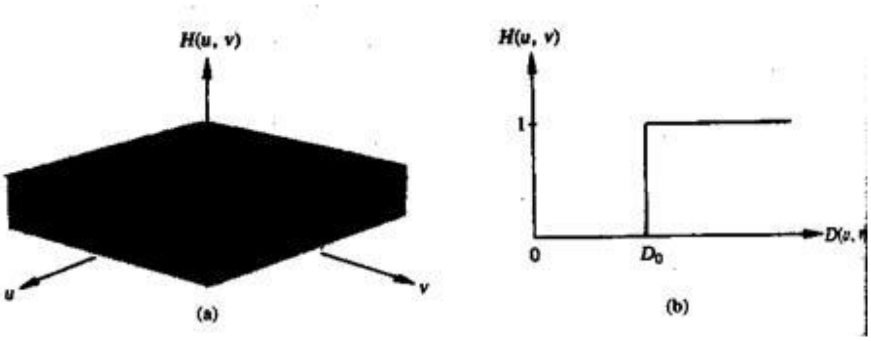


Fig.5.1 Perspective plot and radial cross section of ideal high pass filter

Butterworth filter:

The transfer function of the Butterworth high pass filter (BHPF) of order n and with cutoff frequency locus at a distance Do from the origin is defined by the relation

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

Figure 5.2 shows a perspective plot and cross section of the BHPF function. Note that when D (u, v) = Do, H (u, v) is down to ½ of its maximum value. As in the case of the Butterworth lowpass filter, common practice is to select the cutoff frequency locus at points for which H (u, v) is down to 1/√2 of its maximum value.

$$\begin{aligned} H(u, v) &= \frac{1}{1 + [\sqrt{2} - 1][D_0/D(u, v)]^{2n}} \\ &= \frac{1}{1 + 0.414[D_0/D(u, v)]^{2n}}. \end{aligned}$$

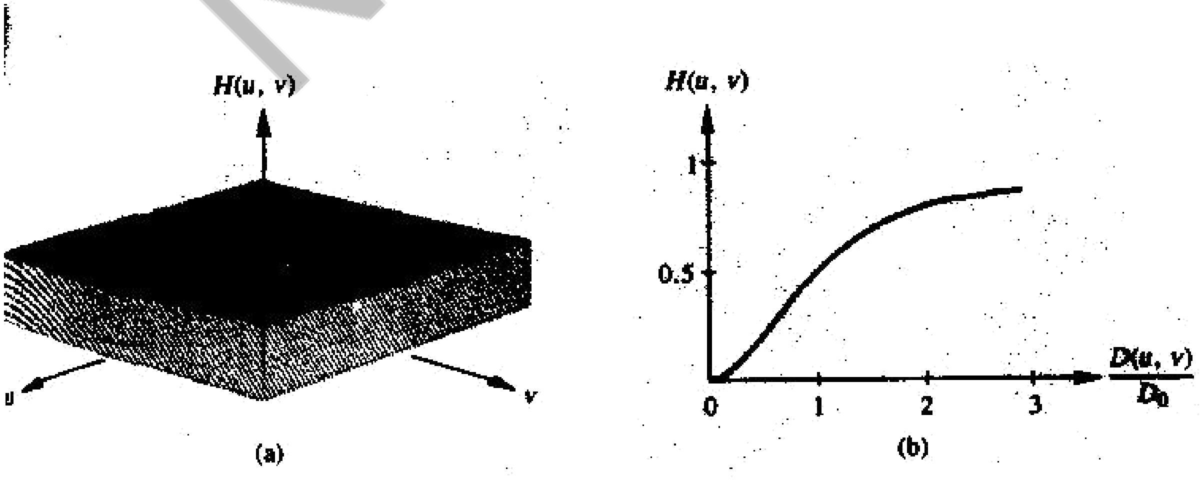


Fig.5.2 Perspective plot and radial cross section for Butterworth High Pass Filter with n = 1

Gaussian High Pass and Gaussian Low Pass Filter.

Gaussian Lowpass Filters:

The form of these filters in two dimensions is given by

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

where, D(u, v) is the distance from the origin of the Fourier transform.

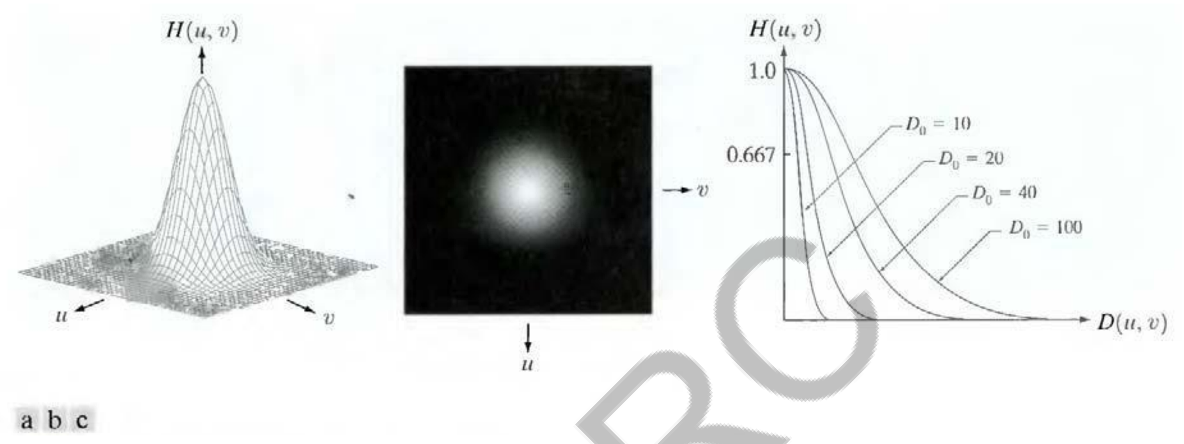


Fig.6.1 (a) Perspective plot of a GLPF transfer function, (b) Filter displayed as an image, (c) Filter radial cross sections for various values of Do.

= is a measure of the spread of the Gaussian curve. By letting σ = Du, we can express the filter in a more familiar form in terms of the notation:

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

where Do is the cutoff frequency. When D (u, v) = Do, the filter is down to 0.607 of its maximum value.

Gaussian Highpass Filters:

The transfer function of the Gaussian highpass filter (GHPF) with cutoff frequency locus at a distance Do from the origin is given by

The figure 6.2 shows a perspective plot, image, and cross section of the GHPF function.

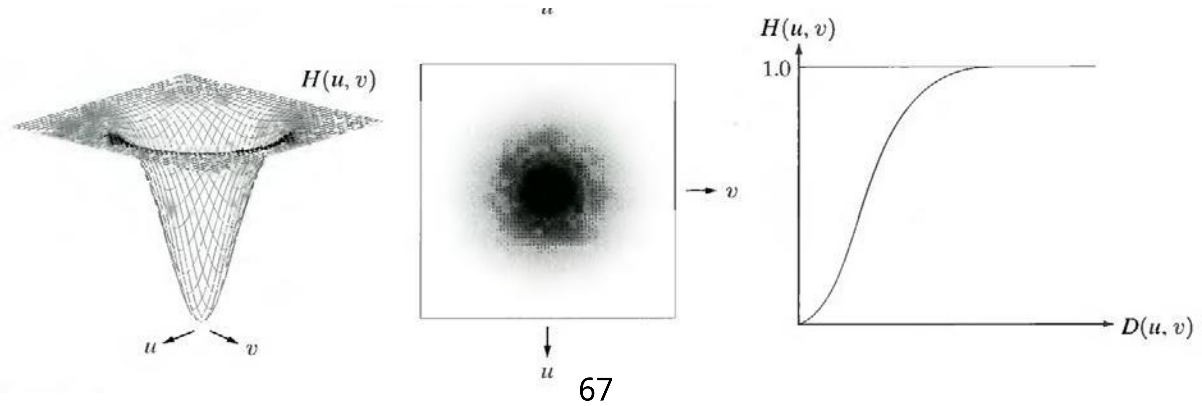


Fig.6.2. Perspective plot, image representation, and cross section of a typical Gaussian high pass filter

Even the filtering of the smaller objects and thin bars is cleaner with the Gaussian filler.

Laplacian in frequency domain.

The Laplacian in the Frequency Domain:

It can be shown that

$$\Im\left[\frac{d^n f(x)}{dx^n}\right] = (ju)^n F(u).$$

From this simple expression, it follows that

$$\Im\left[\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}\right] = (ju)^2 F(u,v) + (jv)^2 F(u,v)$$
$$= -(u^2 + v^2)F(u,v).$$

The expression inside the brackets on the left side of the above Eq. is recognized as the Laplacian of f(x, y). Thus, we have the important result

$$\Im\left[\nabla^2 f(x,y)\right] = -(u^2 + v^2)F(u,v),$$

which simply says that the Laplacian can be implemented in the frequency domain by using the filter

$$H(u,v) = -(u^2 + v^2).$$

As in all filtering operations, the assumption is that the origin of F (u, v) has been centered by performing the operation f(x, y) (-1) $^{x+y}$ prior to taking the transform of the image. If f (and F) are of size M X N, this operation shifts the center transform so that (u, v) = (0, 0) is at point (M/2, N/2) in the frequency rectangle. As before, the center of the filter function also needs to be shifted:

$$H(u,v) = -\left[(u - M/2)^2 + (v - N/2)^2\right].$$

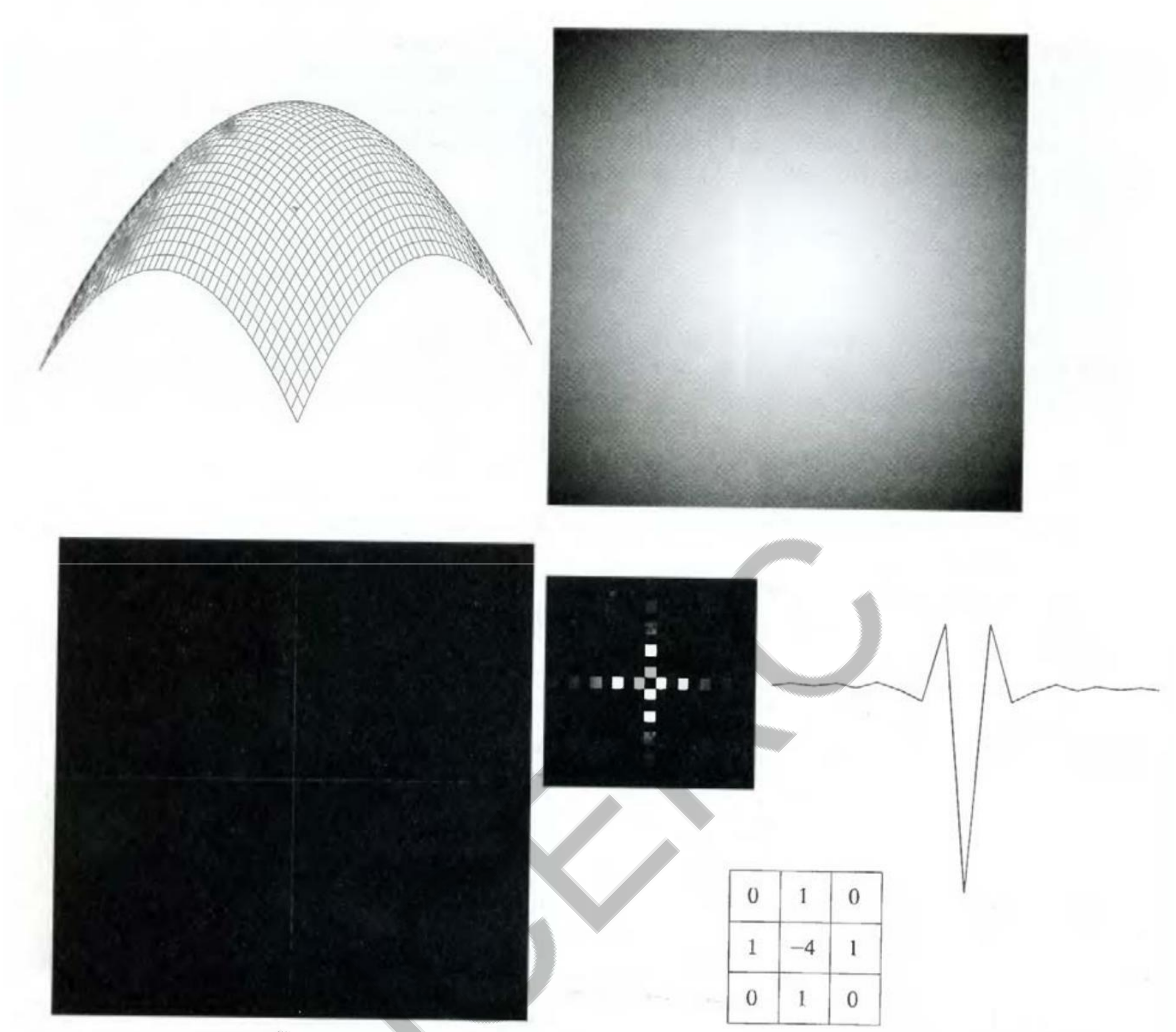
The Laplacian-filtered image in the spatial domain is obtained by computing the inverse Fourier transform of H (u, v) F (u, v):

$$\nabla^2 f(x,y) = \Im^{-1}\left\{-\left[(u - M/2)^2 + (v - N/2)^2\right]F(u,v)\right\}.$$

Conversely, computing the Laplacian in the spatial domain and computing the Fourier transform of the result is equivalent to multiplying F(u, v) by H(u, v). We express this dual relationship in the familiar Fourier-transform-pair notation

$$\nabla^2 f(x,y) \Leftrightarrow -\left[(u - M/2)^2 + (v - N/2)^2\right]F(u,v).$$

The spatial domain Laplacian filter function obtained by taking the inverse Fourier transform of Eq. has some interesting properties, as Fig.7 shows. Figure 7(a) is a 3 -D perspective plot. The function is centered at (M/2, N/2), and its value at the top of the dome is zero. All other values are negative. Figure 7(b) shows H (u, v) as an image, also centered. Figure 7(c) is the Laplacian in the spatial domain, obtained by multiplying by H (u, v) by (-1)$^{u+v}$, taking the inverse Fourier transform, and multiplying the real part of the result by (-l)$^{x+y}$. Figure 7(d) is a zoomed section at about the origin of Fig.7(c).' Figure 7(e) is a horizontal gray-level profile passing through the center of the zoomed section. Finally, Fig.7 (f) shows the mask to implement the definition of the discrete Laplacian in the spatial domain.



0	1	0
1	−4	1
0	1	0

Fig.7 (a) 3 -D plot of Laplacian in the frequency domain, (b) Image representation of (a), (c) Laplacian in the spatial domain obtained from the inverse DFT of (b) (d) Zoomed section of the origin of (c). (e) Gray-level profile through the center of (d). (f) Laplacian mask

horizontal profile through the center of this mask has the same basic shape as the profile in Fig. 7(e) (that is, a negative value between two smaller positive values). We form an enhanced image g(x, y) by subtracting the Laplacian from the original image:

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

High boost and high frequency filtering.

High-Boost Filtering and High-Frequency Emphasis Filtering:

All the filtered images have one thing in common: Their average background intensity has been reduced to near black. This is due to the fact that the highpass filters we applied to those images eliminate the zero-frequency component of their Fourier transforms. In fact, enhancement using the Laplacian does precisely this, by adding back the entire image to the filtered result. Sometimes it is advantageous to increase the contribution made by the original image to the overall filtered result. This approach, called high-boost filtering, is a generalization of unsharp masking. Unsharp masking consists simply of generating a sharp image by subtracting from an image a blurred version of itself. Using frequency domain terminology, this means obtaining a highpass-filtered image by subtracting from the image a lowpass-filtered version of itself. That is

$$f_{\mathrm{hp}}(x, y) = f(x, y) - f_{\mathrm{lp}}(x, y).$$

High-boost filtering generalizes this by multiplying f (x, y) by a constant A > 1:

$$f_{\mathrm{hb}} = Af(x, y) - f_{\mathrm{lp}}(x, y).$$

Thus, high-boost filtering gives us the flexibility to increase the contribution made by the image to the overall enhanced result. This equation may be written as

$$f_{\mathrm{hb}}(x, y) = (A - 1)f(x, y) + f(x, y) - f_{\mathrm{lp}}(x, y).$$

Then, using above Eq. we obtain

$$f_{\mathrm{hb}}(x, y) = (A - 1)f(x, y) + f_{\mathrm{hp}}(x, y).$$

This result is based on a highpass rather than a lowpass image. When A = 1, high-boost filtering reduces to regular highpass filtering. As A increases past 1, the contribution made by the image itself becomes more dominant.

We have F_{hp} (u,v) = F (u,v) – F_{lp} (u,v). But F_{lp} (u,v) = H_{lp} (u,v)F(u,v), where H_{lp} is the transfer function of a lowpass filter. Therefore, unsharp masking can be implemented directly in the frequency domain by using the composite filter

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

Similarly, high-boost filtering can be implemented with the composite filter

$$H_{\mathrm{hb}}(u, v) = (A - 1) + H_{\mathrm{hp}}(u, v)$$

with A > 1. The process consists of multiplying this filter by the (centered) transform of the input image and then taking the inverse transform of the product. Multiplication of the real part of this result by (-l) $^{x+y}$ gives us the high-boost filtered image f_{hb} (x, y) in the spatial domain.

Concept of homomorphic filtering.

Homomorphic filtering:

The illumination-reflectance model can be used to develop a frequency domain procedure for improving the appearance of an image by simultaneous gray-level range compression and contrast enhancement. An image f(x, y) can be expressed as the product of illumination and reflectance components:

$$f(x, y) = i(x, y)r(x, y).$$

Equation above cannot be used directly to operate separately on the frequency components of illumination and reflectance because the Fourier transform of the product of two functions is not separable; in other words,

$$\Im\{f(x, y)\} \neq \Im\{i(x, y)\}\Im\{r(x, y)\}.$$

Suppose, however, that we define

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y). \end{aligned}$$

Then

$$\begin{aligned} \Im\{z(x, y)\} &= \Im\{\ln f(x, y)\} \\ &= \Im\{\ln i(x, y)\} + \Im\{\ln r(x, y)\} \end{aligned}$$

or

$$Z(u, v) = F_i(u, v) + F_r(u, v)$$

where F_i (u, v) and F_r (u, v) are the Fourier transforms of ln i(x, y) and ln r(x, y), respectively. If we process Z (u, v) by means of a filter function H (u, v) then, from

$$\begin{aligned} S(u,v) &= H(u,v)Z(u,v) \\ &= H(u,v)F_i(u,v) + H(u,v)F_r(u,v) \end{aligned}$$

where S (u, v) is the Fourier transform of the result. In the spatial domain,

$$\begin{aligned} s(x,y) &= \Im^{-1}\{S(u,v)\} \\ &= \Im^{-1}\{H(u,v)F_i(u,v)\} + \Im^{-1}\{H(u,v)F_r(u,v)\}. \end{aligned}$$

By letting

$$i'(x,y) = \Im^{-1}\{H(u,v)F_i(u,v)\}$$

and

$$r'(x,y) = \Im^{-1}\{H(u,v)F_r(u,v)\},$$

Now we have

$$s(x,y) = i'(x,y) + r'(x,y).$$

Finally, as z (x, y) was formed by taking the logarithm of the original image f (x, y), the inverse (exponential) operation yields the desired enhanced image, denoted by g(x, y); that is,

$$\begin{aligned} g(x,y) &= e^{s(x,y)} \\ &= e^{i'(x,y)} \cdot e^{r'(x,y)} \\ &= i_0(x,y)r_0(x,y) \end{aligned}$$

where

$$i_0(x,y) = e^{i'(x,y)}$$

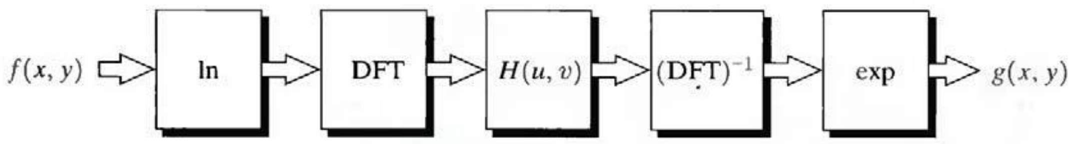


Fig.9.1 Homomorphic filtering approach for image enhancement

and

$$r_0(x,y) = e^{r'(x,y)}$$

are the illumination and reflectance components of the output image. The enhancement approach using the foregoing concepts is summarized in Fig. 9.1. This method is based on a special case of a class of systems known as homomorphic systems. In this particular application, the key to the approach is the separation of the illumination and reflectance components achieved. The homomorphic filter function H (u, v) can then operate on these components separately.

The illumination component of an image generally is characterized by slow spatial variations, while the reflectance component tends to vary abruptly, particularly at the junctions of dissimilar objects. These characteristics lead to associating the low frequencies of the Fourier transform of the logarithm of an image with illumination and the high frequencies with reflectance. Although these associations are rough approximations, they can be used to advantage in image enhancement.

A good deal of control can be gained over the illumination and reflectance components with a homomorphic filter. This control requires specification of a filter function H (u, v) that affects the low- and high-frequency components of the Fourier transform in different ways. Figure 9.2 shows a cross section of such a filter. If the parameters γ_L and γ_H are chosen so that $\gamma_L < 1$ and $\gamma_H > 1$, the filter function shown in Fig. 9.2 tends to decrease the contribution made by the low frequencies (illumination) and amplify the contribution made by high frequencies (reflectance). The net result is simultaneous dynamic range compression and contrast enhancement.

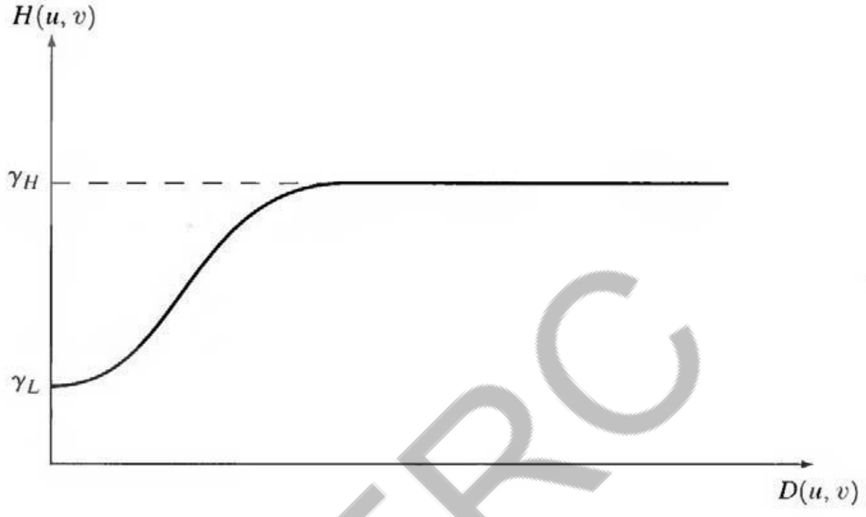


Fig.9.2 Cross section of a circularly symmetric filter function D (u. v) is the distance from the origin of the centered transform.

MODULE IV

Image Enhancement: Frequency domain methods

- The concept of filtering is easier to visualize in the frequency domain. Therefore, enhancement of image $f(m,n)$ can be done in the frequency domain, based on its DFT $F(u,v)$.

- This is particularly useful, if the spatial extent of the point-spread sequence $h(m,n)$ is large. In this case, the convolution

$$\boxed{\underset{\text{Enhanced Image}}{g}(m,n) = \overset{\text{PSS}}{h}(m,n) * \underset{\text{Given Image}}{f}(m,n)}$$

may be computationally unattractive.

- We can therefore directly design a transfer function $H(u,v)$ and implement the enhancement in the frequency domain as follows:

$$\boxed{\underset{\text{Enhanced Image}}{G}(u,v) = \overset{\text{Transfer function}}{H}(u,v)\underset{\text{Given Image}}{F}(u,v)}$$

Lowpass filtering

- Edges and sharp transitions in grayvalues in an image contribute significantly to high-frequency content of its Fourier transform.

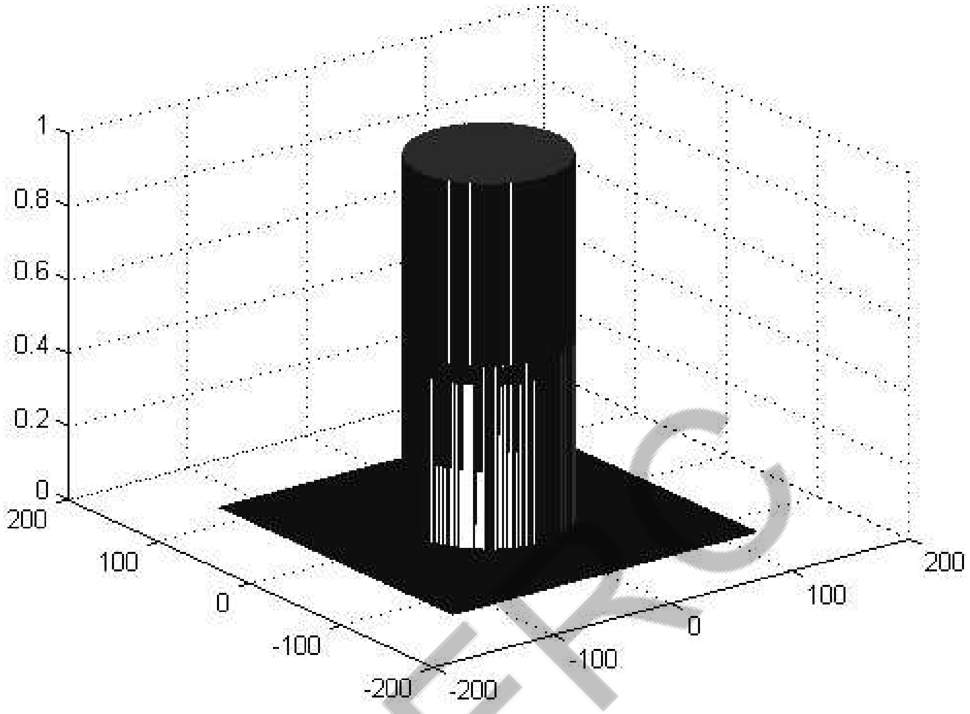
- Regions of relatively uniform grayvalues in an image contribute to low-frequency content of its Fourier transform.

- Hence, an image can be smoothed in the Frequency domain by attenuating the high-frequency content of its Fourier transform. This would be a lowpass filter!

- For simplicity, we will consider only those filters that are real and radially symmetric.

- An **ideal lowpass filter** with **cutoff frequency** r_0:

$$H(u,v) = \begin{matrix} 1, \text{ if } \sqrt{u^2+v^2} \le r_0 \\ 0, \text{ if } \sqrt{u^2+v^2} > r_0 \end{matrix}$$

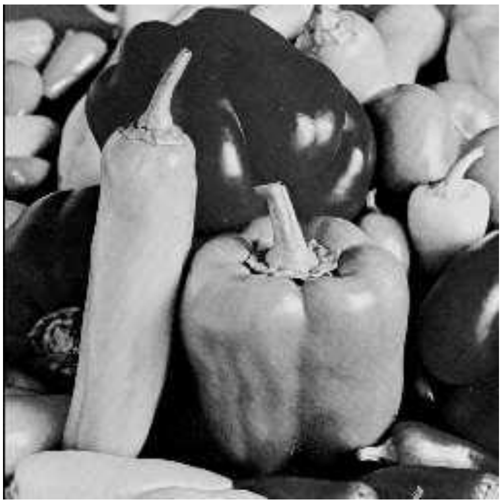


Ideal LPF with $r_0 = 57$

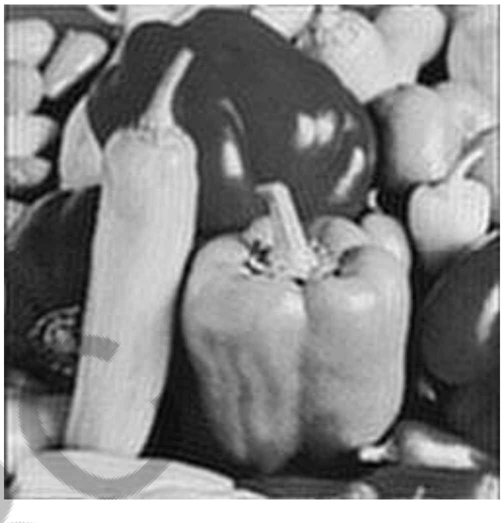
- Note that the origin (0, 0) is at the center and not the corner of the image (recall the "`fftshift`" operation).

- The abrupt transition from 1 to 0 of the transfer function $H(u,v)$ cannot be realized in practice, using electronic components. However, it can be simulated on a computer.

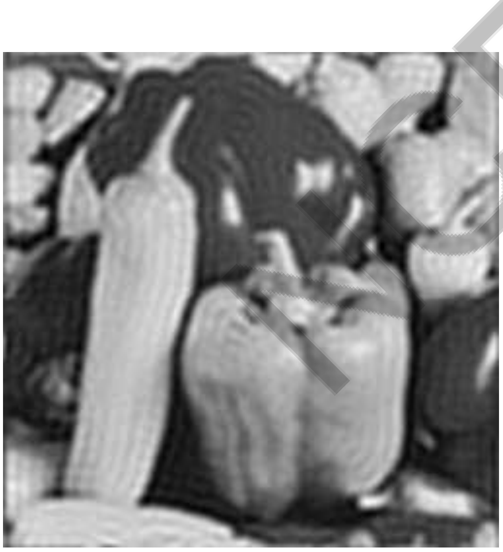
Ideal LPF examples



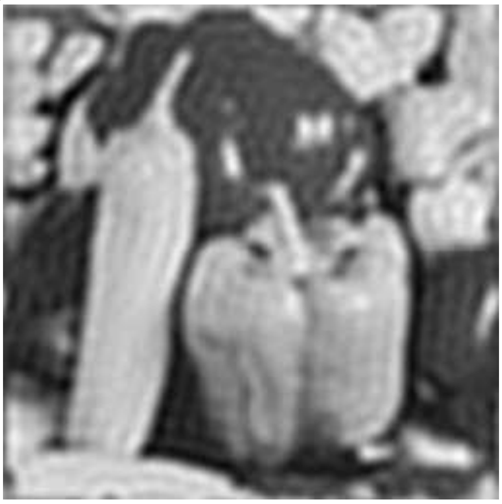
Original Image



LPF image, $r_0 = 57$



LPF image, $r_0 = 36$



LPF image, $r_0 = 26$

- Notice the severe **ringing** effect in the blurred images, which is a characteristic of ideal filters. It is due to the discontinuity in the filter transfer function.

Choice of cutoff frequency in ideal LPF

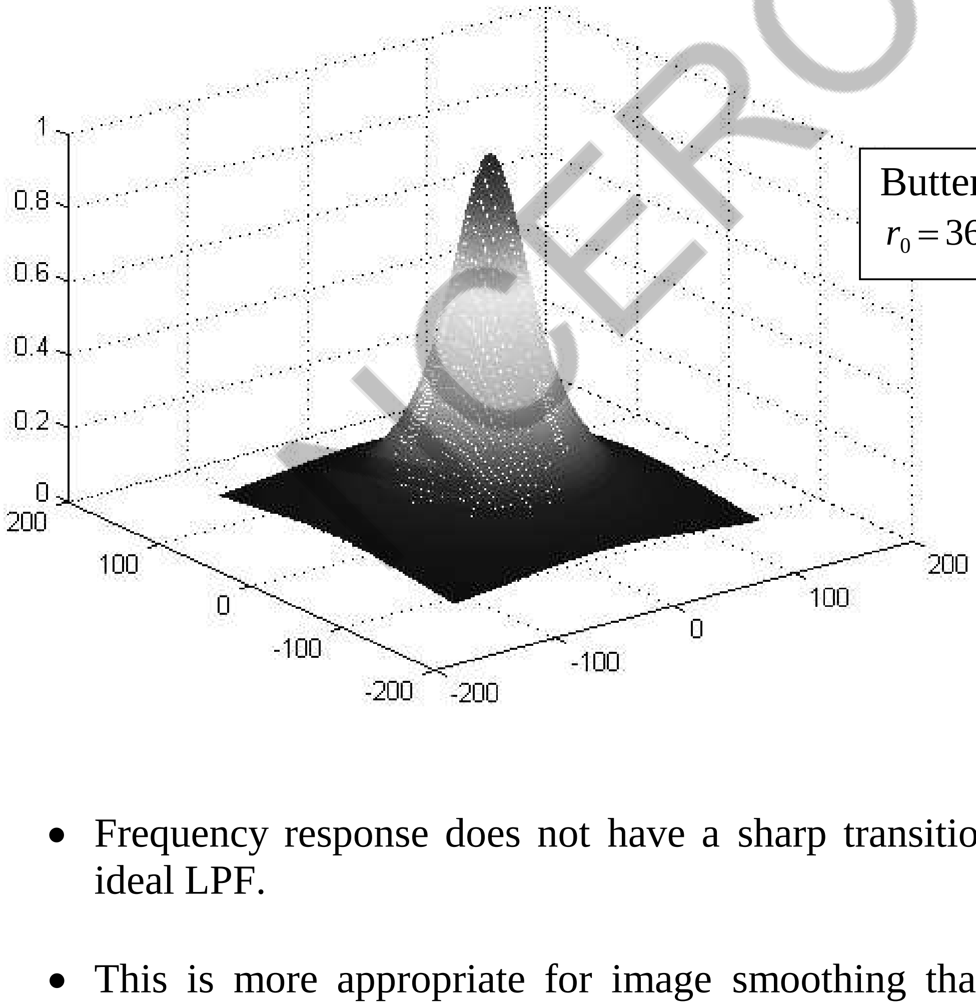
- The cutoff frequency r_0 of the ideal LPF determines the amount of frequency components passed by the filter.
- Smaller the value of r_0, more the number of image components eliminated by the filter.
- In general, the value of r_0 is chosen such that most components of interest are passed through, while most components not of interest are eliminated.
- Usually, this is a set of conflicting requirements. We will see some details of this is image restoration
- A useful way to establish a set of standard cut-off frequencies is to compute circles which enclose a specified fraction of the total image power.
- Suppose $P_T = \sum_{v=0}^{N-1}\sum_{u=0}^{M-1} P(u,v)$, where $P(u,v) = |F(u,v)|^2$, is the total image power.
- Consider a circle of radius $r_0(\alpha)$ as a cutoff frequency with respect to a threshold α such that $\sum_v \sum_u P(u,v) = \alpha P_T$.
- We can then fix a threshold α and obtain an appropriate cutoff frequency $r_0(\alpha)$.

Butterworth lowpass filter

- A two-dimensional Butterworth lowpass filter has transfer function:

$$H(u,v)=\frac{1}{1+{\frac{\sqrt{u^2+v^2}}{r_0}}^{2n}}$$

  - n: filter order, r_0: cutoff frequency

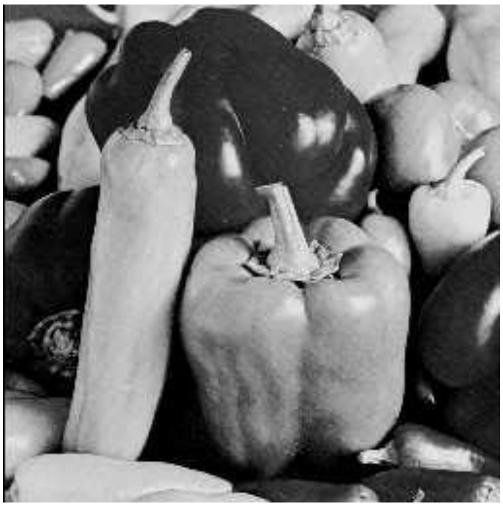


Butterworth LPF with $r_0 = 36$ and $n = 1$

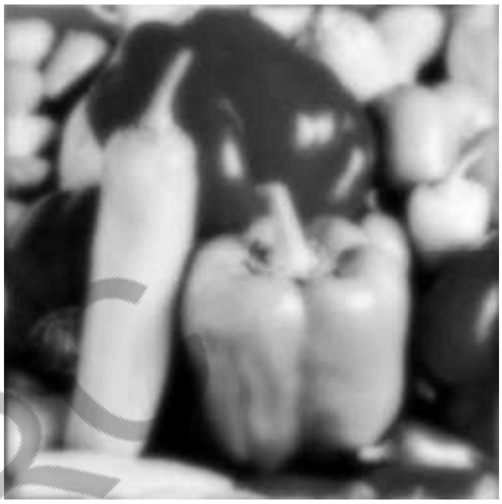
- Frequency response does not have a sharp transition as in the ideal LPF.

- This is more appropriate for image smoothing than the ideal LPF, since this not introduce ringing.

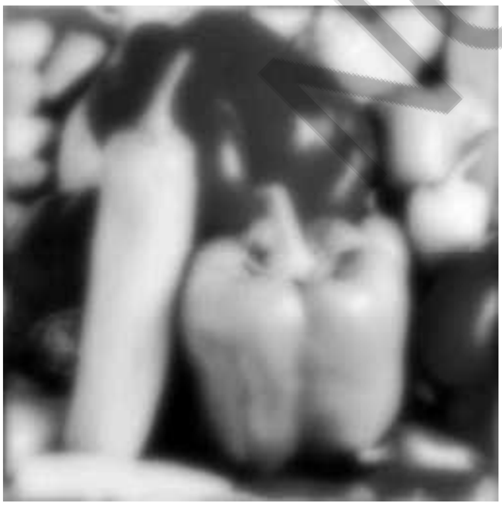
Butterworth LPF example



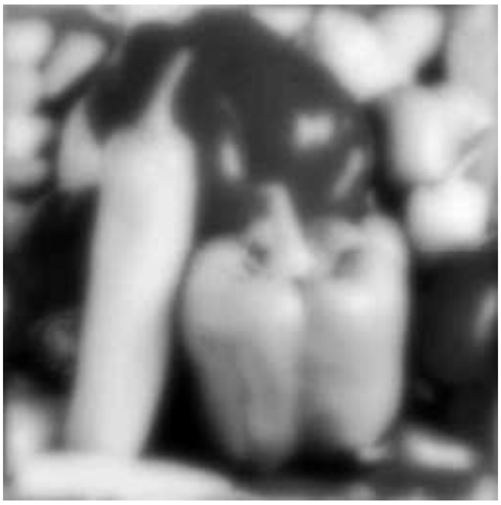
Original Image



LPF image, $r_0 = 18$



LPF image, $r_0 = 13$



LPF image, $r_0 = 10$

Butterworth LPF example: False contouring

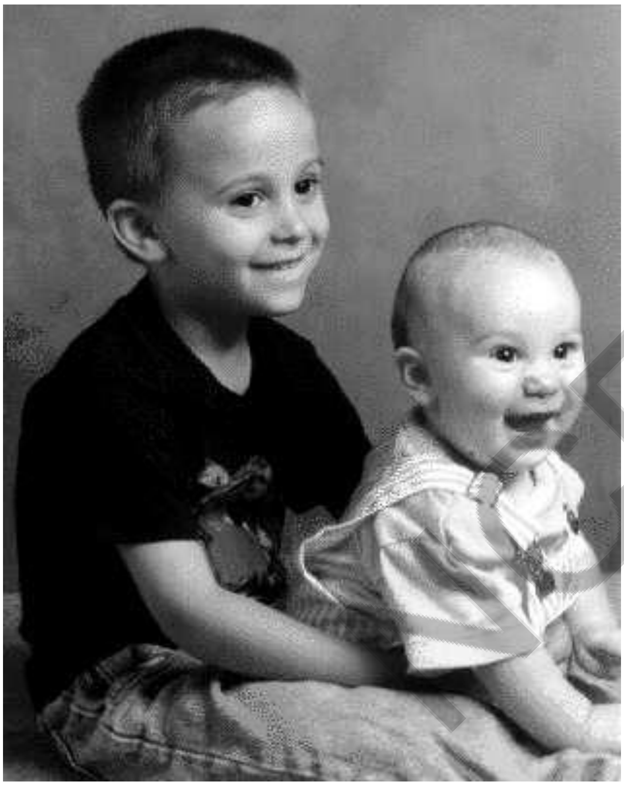
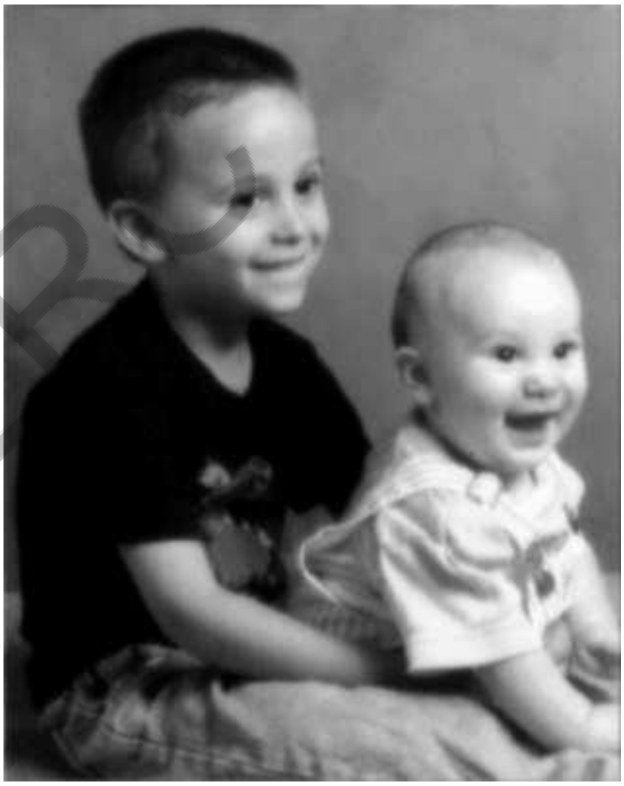
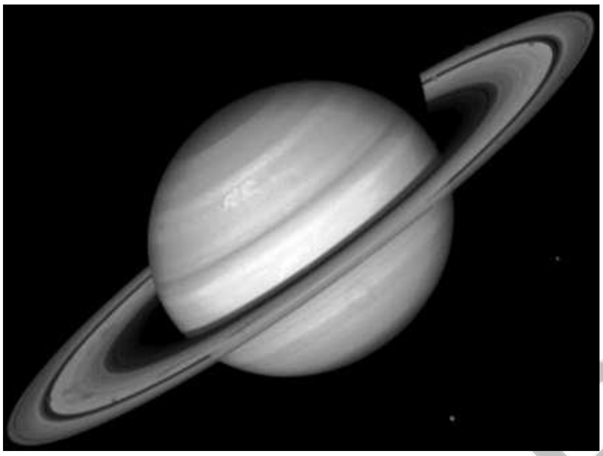


Image with false contouring due to insufficient bits used for quantization

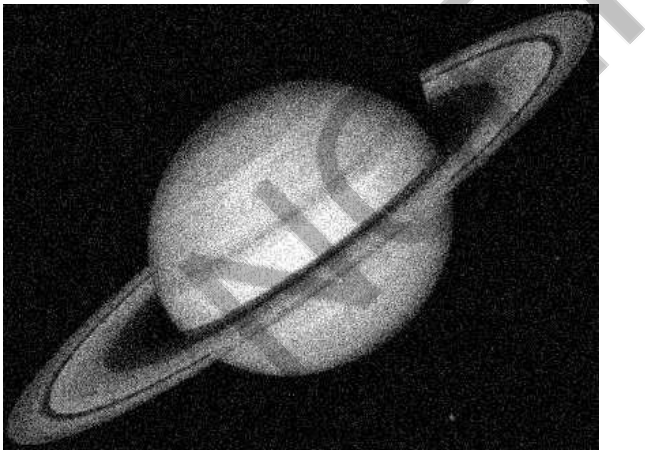


Lowpass filtered version of previous image

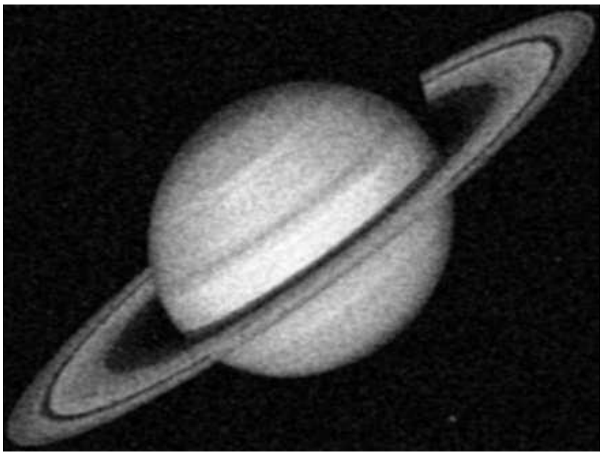
Butterworth LPF example: Noise filtering



Original Image



Noisy Image



LPF Image

Gaussian Low pass filters

- The form of a Gaussian lowpass filter in two-dimensions is given by $H(u,v) = e^{-D^2(u,v)/2\sigma^2}$, where $D(u,v) = \sqrt{u^2 + v^2}$ is the distance from the origin in the frequency plane.
- The parameter σ measures the spread or dispersion of the Gaussian curve. Larger the value of σ, larger the cutoff frequency and milder the filtering.
- When $D(u,v) = \sigma$, the filter is down to 0.607 of its maximum value of 1.
- See Example 4.6 in the text for an illustration.
- Also read section 4.3.4 for an application of lowpass filtering to text images.

Highpass filtering

- Edges and sharp transitions in grayvalues in an image contribute significantly to high-frequency content of its Fourier transform.

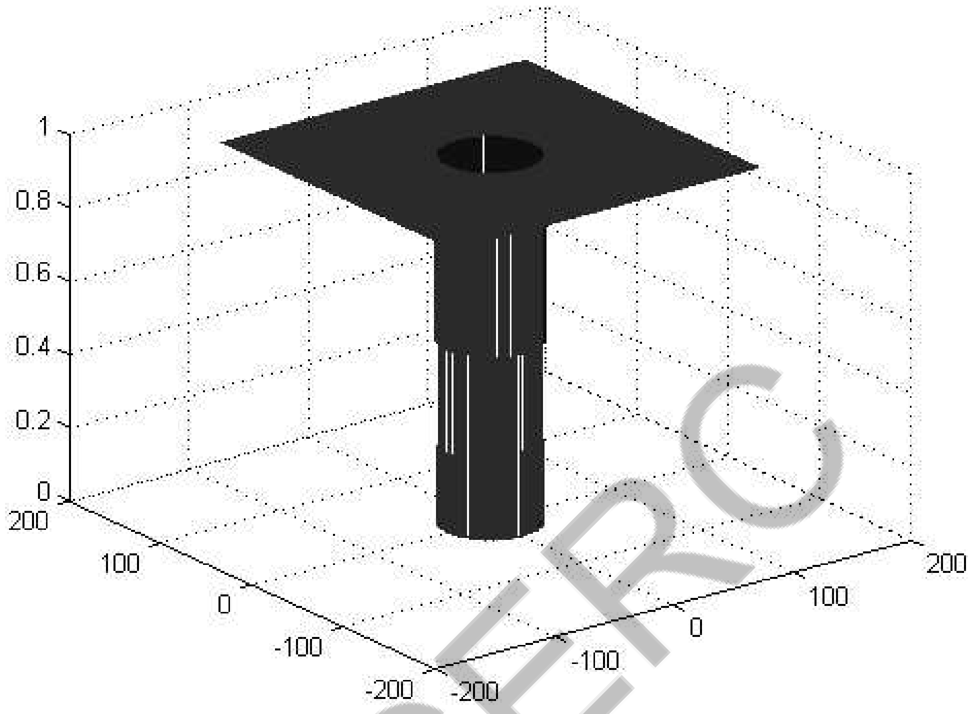
- Regions of relatively uniform grayvalues in an image contribute to low-frequency content of its Fourier transform.

- Hence, image sharpening in the Frequency domain can be done by attenuating the low-frequency content of its Fourier transform. This would be a highpass filter!

- For simplicity, we will consider only those filters that are real and radially symmetric.

- An **ideal highpass filter** with **cutoff frequency** r_0:

$$H(u,v) = \begin{cases} 0, & \text{if } \sqrt{u^2+v^2} \le r_0 \\ 1, & \text{if } > \sqrt{u^2+v^2} \quad r_0 \end{cases}$$

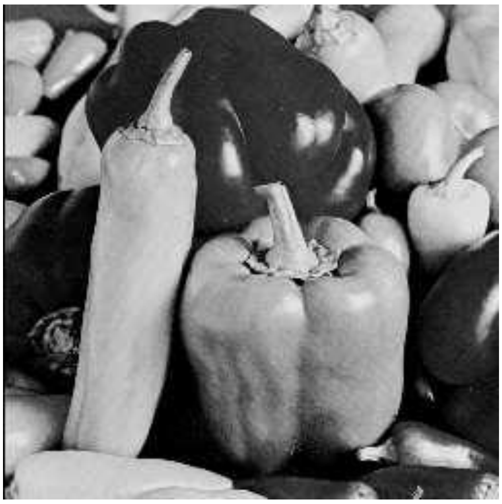


Ideal HPF with $r_0 = 36$

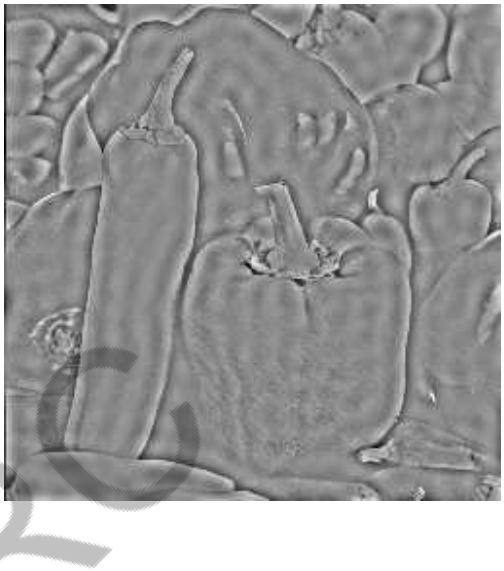
- Note that the origin (0, 0) is at the center and not the corner of the image (recall the "`fftshift`" operation).

- The abrupt transition from 1 to 0 of the transfer function $H(u,v)$ cannot be realized in practice, using electronic components. However, it can be simulated on a computer.

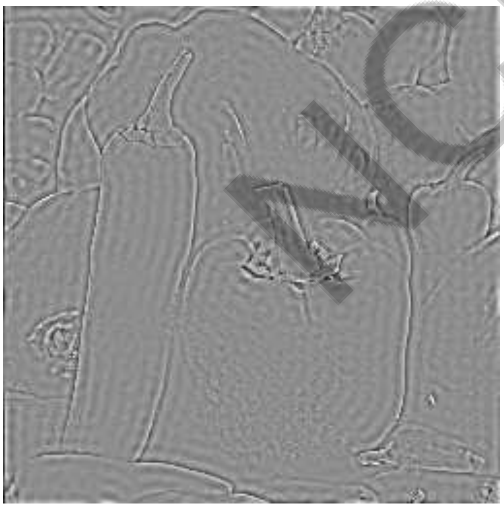
Ideal HPF examples



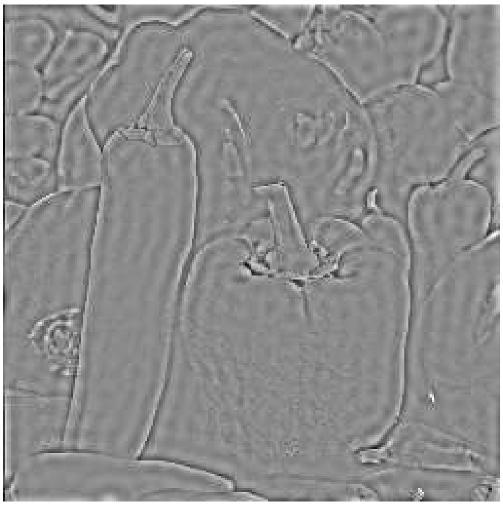
Original Image



HPF image, $r_0 = 18$



HPF image, $r_0 = 36$



HPF image, $r_0 = 26$

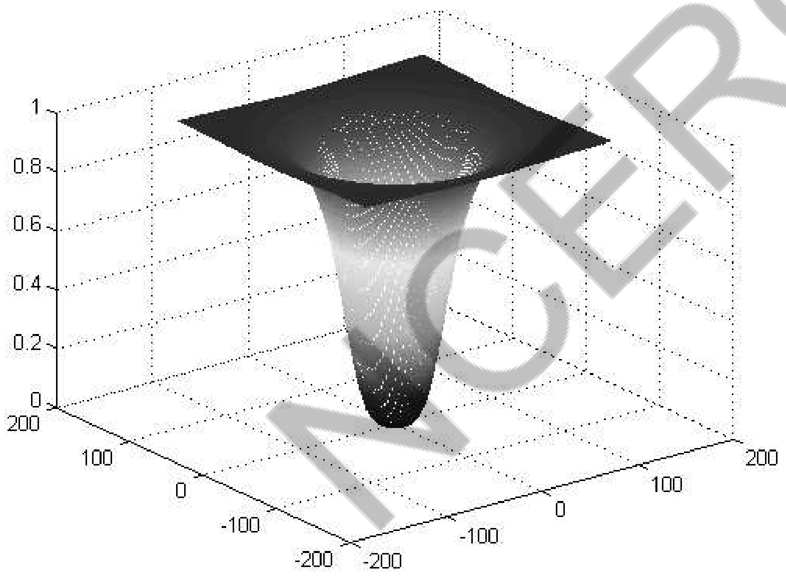
- Notice the severe **ringing** effect in the output images, which is a characteristic of ideal filters. It is due to the discontinuity in the filter transfer function.

Butterworth highpass filter

- A two-dimensional Butterworth highpass filter has transfer function:

$$H(u,v) = \frac{1}{1 + \left(\frac{r_0}{\sqrt{u^2 + v^2}} \right)^{2n}}$$

  - n: filter order, r_0: cutoff frequency



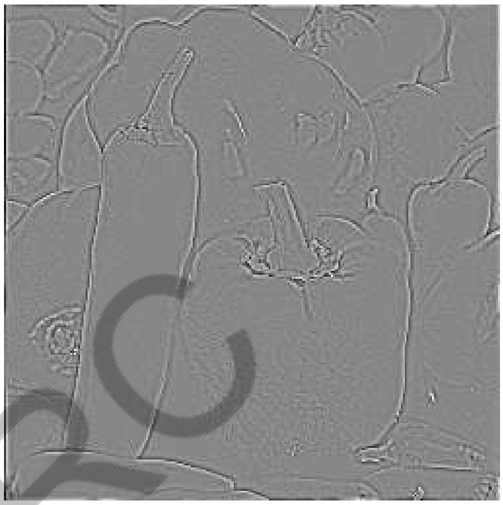
Butterworth HPF with $r_0 = 47$ and 2

- Frequency response does not have a sharp transition as in the ideal HPF.

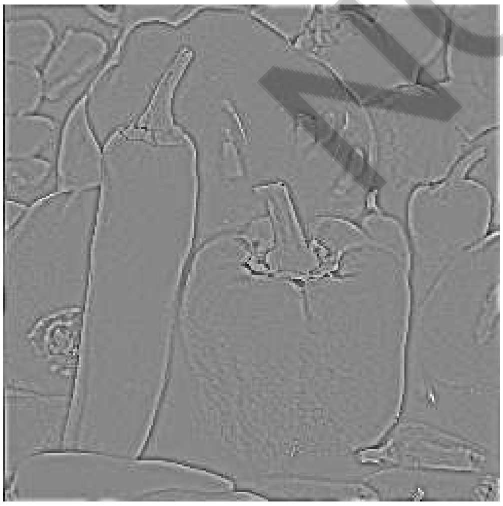
- This is more appropriate for image sharpening than the ideal HPF, since this not introduce ringing.

Butterworth HPF example

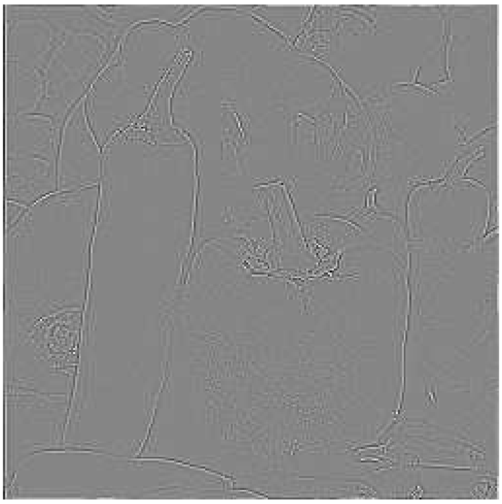
Original Image



HPF image, $r_0 = 47$



HPF image, $r_0 = 36$



HPF image, $r_0 = 81$

Gaussian High pass filters

- The form of a Gaussian lowpass filter in two-dimensions is given by $H(u,v) = 1 - e^{-D^2(u,v)/2\sigma^2}$, where $D(u,v) = \sqrt{u^2 + v^2}$ is the distance from the origin in the frequency plane.
- The parameter σ measures the spread or dispersion of the Gaussian curve. Larger the value of σ, larger the cutoff frequency and more severe the filtering.
- See Example in section 4.4.3 of text for an illustration.

Derivative operators useful in image segmentation

Gradient operators:

First-order derivatives of a digital image are based on various approximations of the 2-D gradient. The gradient of an image f (x, y) at location (x, y) is defined as the vector

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

It is well known from vector analysis that the gradient vector points in the direction of maximum rate of change of f at coordinates (x, y). An important quantity in edge detection is the magnitude of this vector, denoted by f, where

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}.$$

This quantity gives the maximum rate of increase of f (x, y) per unit distance in the direction of f. It is a common (although not strictly correct) practice to refer to f also as the gradient. The direction of the gradient vector also is an important quantity. Let α (x, y) represent the direction angle of the vector f at (x, y). Then, from vector analysis,

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

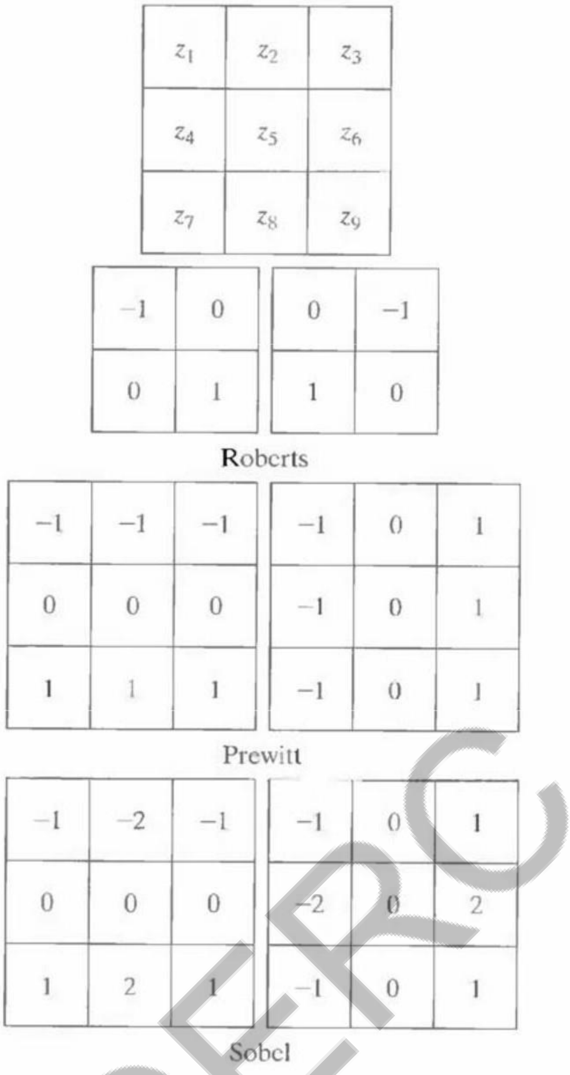
where the angle is measured with respect to the x-axis. The direction of an edge at (x, y) is perpendicular to the direction of the gradient vector at that point. Computation of the gradient of an image is based on obtaining the partial derivatives f/ x and f/ y at every pixel location. Let the 3x3 area shown in Fig. 1.1 (a) represent the gray levels in a neighborhood of an image. One of the simplest ways to implement a first-order partial derivative at point z5 is to use the following Roberts cross-gradient operators:

$$G_x = (z_9 - z_5)$$

and

$$G_y = (z_8 - z_6).$$

These derivatives can be implemented for an entire image by using the masks shown in Fig. 1.1(b). Masks of size 2 X 2 are awkward to implement because they do not have a clear center. An approach using masks of size 3 X 3 is given by



−1	0		0	−1
0	1		1	0

Roberts

−1	−1	−1		−1	0	1
0	0	0		−1	0	1
1	1	1		−1	0	1

Prewitt

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

Sobel

Fig.1.1 A 3 X 3 region of an image (the z's are gray-level values) and various masks used to compute the gradient at point labeled z_5.

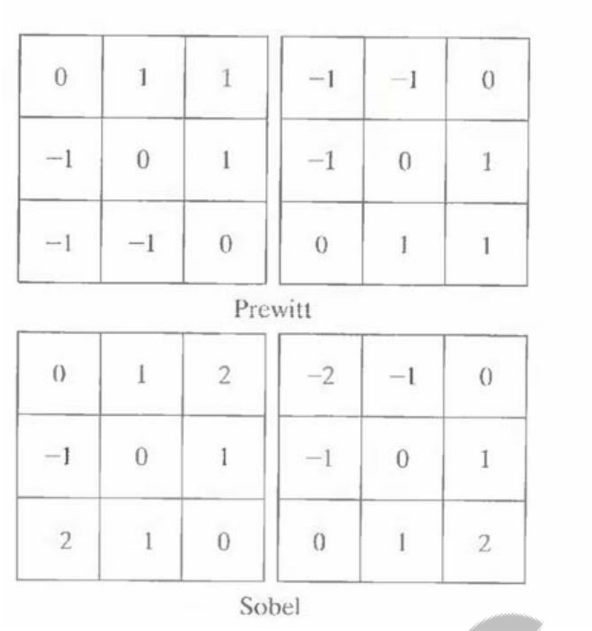
A weight value of 2 is used to achieve some smoothing by giving more importance to the center point. Figures 1.1(f) and (g), called the Sobel operators, and are used to implement these two equations. The Prewitt and Sobel operators are among the most used in practice for computing digital gradients. The Prewitt masks are simpler to implement than the Sobel masks, but the latter have slightly superior noise-suppression characteristics, an important issue when dealing with derivatives. Note that the coefficients in all the masks shown in Fig. 1.1 sum to 0, indicating that they give a response of 0 in areas of constant gray level, as expected of a derivative operator.

The masks just discussed are used to obtain the gradient components G_x and G_y. Computation of the gradient requires that these two components be combined. However, this implementation is not always desirable because of the computational burden required by squares and square roots. An approach used frequently is to approximate the gradient by absolute values:

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

This equation is much more attractive computationally, and it still preserves relative changes in gray levels. However, this is not an issue when masks such as the Prewitt and Sobel masks are used to compute Gx and Gy.

It is possible to modify the 3 X 3 masks in Fig. 1.1 so that they have their strongest responses along the diagonal directions. The two additional Prewitt and Sobel masks for detecting discontinuities in the diagonal directions are shown in Fig. 1.2.



0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Prewitt

0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

Sobel

Fig.1.2 Prewitt and Sobel masks for detecting diagonal edges

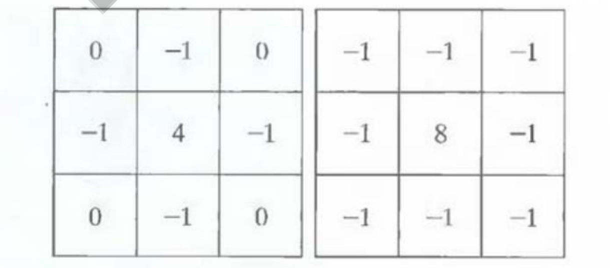
The Laplacian:

The Laplacian of a 2-D function f(x, y) is a second-order derivative defined as

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

For a 3 X 3 region, one of the two forms encountered most frequently in practice is

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$



0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Fig.1.3 Laplacian masks used to implement Eqns. above.

where the z's are defined in Fig. 1.1(a). A digital approximation including the diagonal neighbors is given by

$$\nabla^2 f = 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8 + z_9).$$

Masks for implementing these two equations are shown in Fig. 1.3. We note from these masks that the implementations of Eqns. are isotropic for rotation increments of 90° and 45°, respectively.

Edge detection.

Intuitively, an edge is a set of connected pixels that lie on the boundary between two regions. Fundamentally, an edge is a "local" concept whereas a region boundary, owing to the way it is defined, is a more global idea. A reasonable definition of "edge" requires the ability to measure gray-level transitions in a meaningful way. We start by modeling an edge intuitively. This will lead us to formalism in which "meaningful" transitions in gray levels can be measured. Intuitively, an ideal edge has the properties of the model shown in Fig. 2.1(a). An ideal edge according to this model is a set of connected pixels (in the vertical direction here), each of which is located at an orthogonal step transition in gray level (as shown by the horizontal profile in the figure).

In practice, optics, sampling, and other image acquisition imperfections yield edges that are blurred, with the degree of blurring being determined by factors such as the quality of the image acquisition system, the sampling rate, and illumination conditions under which the image is acquired. As a result, edges are more closely modeled as having a "ramp like" profile, such as the one shown in Fig.2.1 (b).

Fig.2.1 (a) Model of an ideal digital edge (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

The slope of the ramp is inversely proportional to the degree of blurring in the edge. In this model, we no longer have a thin (one pixel thick) path. Instead, an edge point now is any point contained in the ramp, and an edge would then be a set of such points that are connected. The "thickness" of the edge is determined by the length of the ramp, as it transitions from an initial to a final gray level. This length is determined by the slope, which, in turn, is determined by the degree of blurring. This makes sense: Blurred edges lend to be thick and sharp edges tend to be thin. Figure 2.2(a) shows the image from which the close-up in Fig. 2.1(b) was extracted. Figure 2.2(b) shows a horizontal gray-level profile of the edge between the two regions. This figure also shows the first and second derivatives of the gray-level profile. The first derivative is positive at the points of transition into and out of the ramp as we move from left to right along the profile; it is constant for points in the ramp; and is zero in areas of constant gray level. The second derivative is positive at the transition associated with the dark side of the edge, negative at the transition associated with the light side of the edge, and zero along the ramp and in areas of constant gray level. The signs of the derivatives in Fig. 2.2(b) would be reversed for an edge that transitions from light to dark.

We conclude from these observations that the magnitude of the first derivative can be used to detect the presence of an edge at a point in an image (i.e. to determine if a point is on a ramp). Similarly, the sign of the second derivative can be used to determine whether an edge pixel lies on the dark or light side of an edge. We note two additional properties of the second derivative around an edge: A) It produces two values for every edge in an image (an undesirable feature); and B) an imaginary straight line joining the extreme positive and negative values of the second derivative would cross zero near the midpoint of the edge. This zero-crossing property of the second derivative is quite useful for locating the centers of thick edges.

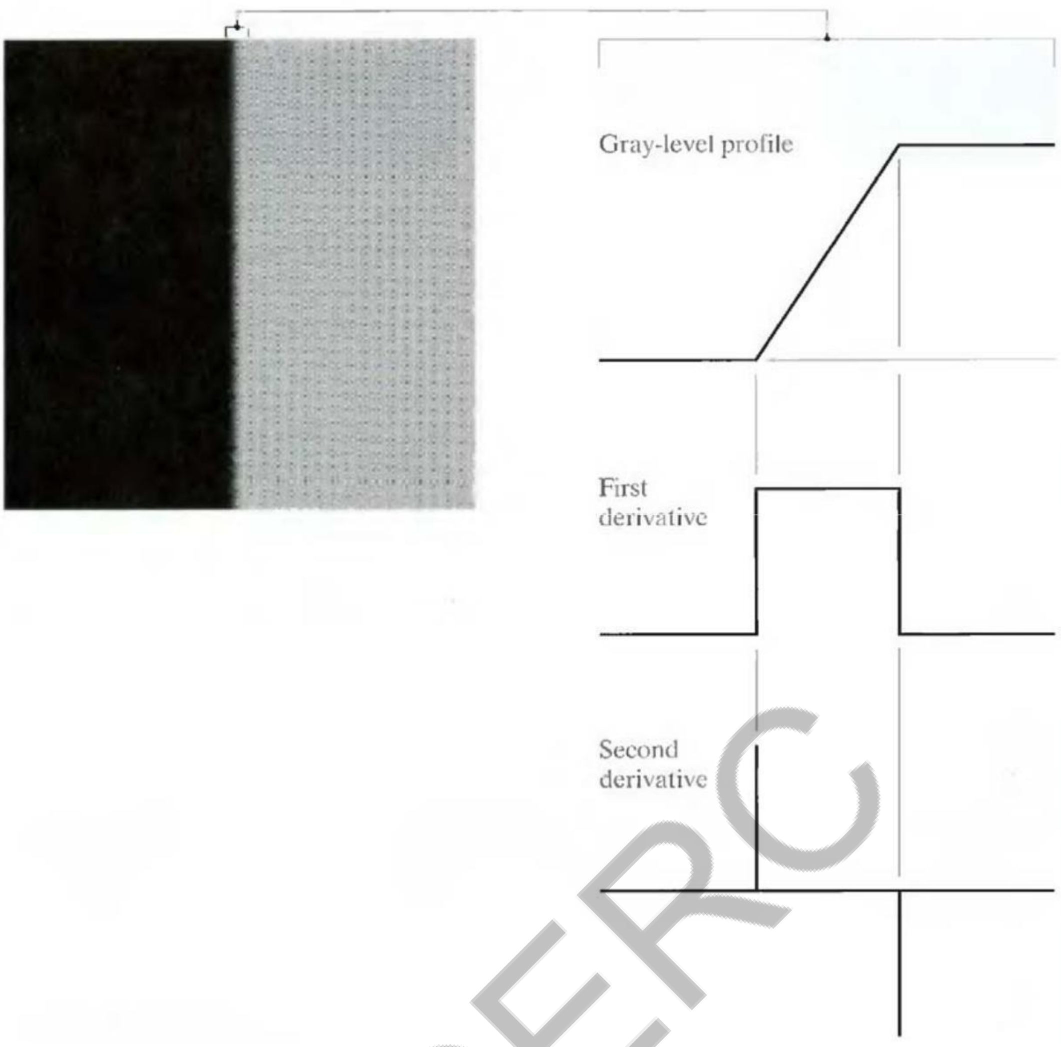


Fig.2.2 (a) Two regions separated by a vertical edge (b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.

Edge linking procedures.

The different methods for edge linking are as follows

(i) Local processing

(ii) Global processing via the Hough Transform

(iii) Global processing via graph-theoretic techniques.

(i) Local Processing:

One of the simplest approaches for linking edge points is to analyze the characteristics of pixels in a small neighborhood (say, 3 X 3 or 5 X 5) about every point (x, y) in an image that has been labeled an edge point. All points that are similar according to a set of predefined criteria are linked, forming an edge of pixels that share those criteria.

The two principal properties used for establishing similarity of edge pixels in this kind of analysis are (1) the strength of the response of the gradient operator used to produce the edge pixel; and (2) the direction of the gradient vector. The first property is given by the value of f.

Thus an edge pixel with coordinates (x_o, y_o) in a predefined neighborhood of (x, y), is similar in magnitude to the pixel at (x, y) if

$$\left|\nabla f(x, y) - \nabla f(x_0, y_0)\right| \le E$$

The direction (angle) of the gradient vector is given by Eq. An edge pixel at (x_o, y_o) in the predefined neighborhood of (x, y) has an angle similar to the pixel at (x, y) if

$$\left|\alpha(x, y) - \alpha(x_0, y_0)\right| < A$$

where A is a nonnegative angle threshold. The direction of the edge at (x, y) is perpendicular to the direction of the gradient vector at that point.

A point in the predefined neighborhood of (x, y) is linked to the pixel at (x, y) if both magnitude and direction criteria are satisfied. This process is repeated at every location in the image. A record must be kept of linked points as the center of the neighborhood is moved from pixel to pixel. A simple bookkeeping procedure is to assign a different gray level to each set of linked edge pixels.

(ii) Global processing via the Hough Transform:

In this process, points are linked by determining first if they lie on a curve of specified shape. We now consider global relationships between pixels. Given n points in an image, suppose that we want to find subsets of these points that lie on straight lines. One possible solution is to first find all lines determined by every pair of points and then find all subsets of points that are close to particular lines. The problem with this procedure is that it involves finding n(n - 1)/2 ~ n^2 lines and then performing (n)(n(n - l))/2 ~ n^3 comparisons of every point to all lines. This approach is computationally prohibitive in all but the most trivial applications.

Hough [1962] proposed an alternative approach, commonly referred to as the Hough transform. Consider a point (x_i, y_i) and the general equation of a straight line in slope-intercept form, y_i = a.x_i + b. Infinitely many lines pass through (x_i, y_i) but they all satisfy the equation y_i = a.x_i + b for varying values of a and b. However, writing this equation as b = -a.x_i + y_i, and considering the ab-plane (also called parameter space) yields the equation of a single line for a fixed pair (x_i, y_i). Furthermore, a second point (x_j, y_j) also has a line in parameter space associated with it, and this line intersects the line associated with (x_i, y_i) at (a', b'), where a' is the slope and b' the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy-plane. In fact, all points contained on this line have lines in parameter space that intersect at (a', b'). Figure 3.1 illustrates these concepts.

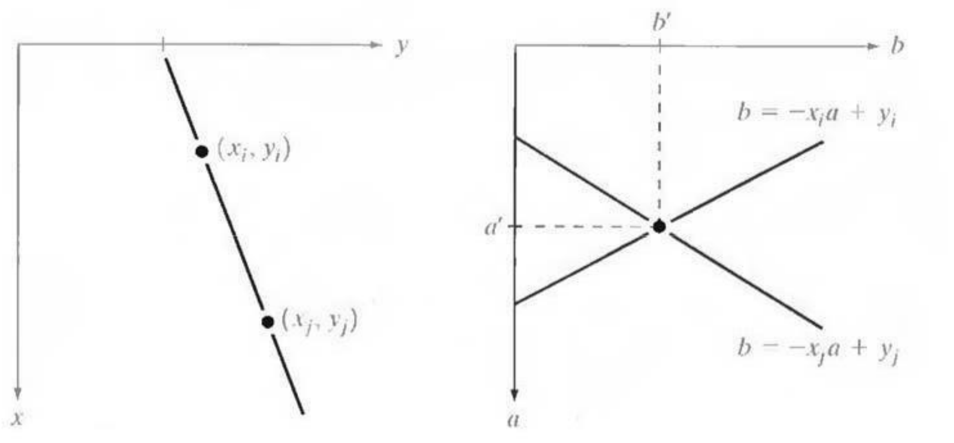


Fig.3.1 (a) xy-plane (b) Parameter space

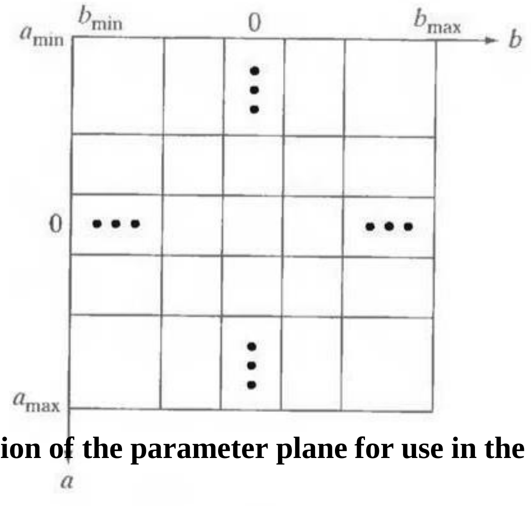


Fig.3.2 Subdivision of the parameter plane for use in the Hough transform

The computational attractiveness of the Hough transform arises from subdividing the parameter space into so-called accumulator cells, as illustrated in Fig. 3.2, where (a_{max} , a_{min}) and (b_{max} , b_{min}), are the expected ranges of slope and intercept values. The cell at coordinates (i, j), with accumulator value A(i, j), corresponds to the square associated with parameter space coordinates (a_i , b_i).

Initially, these cells are set to zero. Then, for every point (x_k, y_k) in the image plane, we let the parameter a equal each of the allowed subdivision values on the fl-axis and solve for the corresponding b using the equation b = - x_k a + y_k .The resulting b's are then rounded off to the nearest allowed value in the b-axis. If a choice of a_p results in solution b_q, we let A (p, q) = A (p, q) + 1. At the end of this procedure, a value of Q in A (i, j) corresponds to Q points in the xy-plane lying on the line y = a_i x + b_j. The number of subdivisions in the ab-plane determines the accuracy of the co linearity of these points. Note that subdividing the a axis into K increments gives, for every point (x_k, y_k), K values of b corresponding to the K possible values of a. With n image points, this method involves nK computations. Thus the procedure just discussed is linear in n, and the product nK does not approach the number of computations discussed at the beginning unless K approaches or exceeds n.

A problem with using the equation y = ax + b to represent a line is that the slope approaches infinity as the line approaches the vertical. One way around this difficulty is to use the normal representation of a line:

$$x \cos\theta + y \sin\theta = \rho$$

Figure 3.3(a) illustrates the geometrical interpretation of the parameters used. The use of this representation in constructing a table of accumulators is identical to the method discussed for the slope-intercept representation. Instead of straight lines, however, the loci are sinusoidal curves in the ρθ -plane. As before, Q collinear points lying on a line x $\cos\theta_j$ + y $\sin\theta_j$ = ρ, yield Q sinusoidal curves that intersect at (p_i, θ_j) in the parameter space. Incrementing θ and solving for the corresponding p gives Q entries in accumulator A (i, j) associated with the cell determined by (p_i, θ_j). Figure 3.3 (b) illustrates the subdivision of the parameter space.

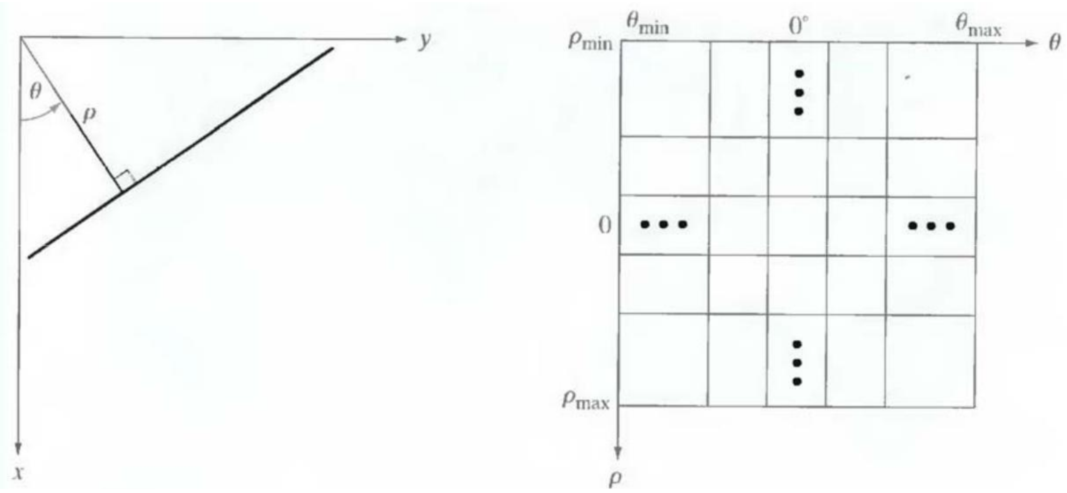


Fig.3.3 (a) Normal representation of a line (b) Subdivision of the ρθ-plane into cells

The range of angle θ is ±90°, measured with respect to the x-axis. Thus with reference to Fig. 3.3 (a), a horizontal line has θ = 0°, with ρ being equal to the positive x-intercept. Similarly, a vertical line has θ = 90°, with p being equal to the positive y-intercept, or θ = - 90°, with ρ being equal to the negative y-intercept.

(iii) Global processing via graph-theoretic techniques

In this process we have a global approach for edge detection and linking based on representing edge segments in the form of a graph and searching the graph for low-cost paths that correspond to significant edges. This representation provides a rugged approach that performs well in the presence of noise.

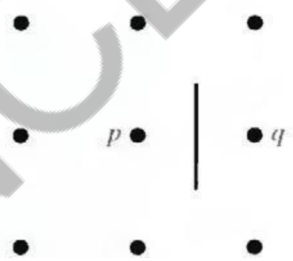


Fig.3.4 Edge clement between pixels p and q

We begin the development with some basic definitions. A graph G = (N,U) is a finite, nonempty set of nodes N, together with a set U of unordered pairs of distinct elements of N. Each pair (n_i, n_j) of U is called an arc. A graph in which the arcs are directed is called a directed graph. If an arc is directed from node n_i to node n_j, then n_j is said to be a successor of the parent node n_i. The process of identifying the successors of a node is called expansion of the node. In each graph we define levels, such that level 0 consists of a single node, called the start or root node, and the nodes in the last level are called goal nodes. A cost c (n_i, n_j) can be associated with every arc (n_i, n_j). A sequence of nodes n_1, n_2... n_k, with each node n_i being a successor of node n_{i-1} is called a path from n_1 to n_k. The cost of the entire path is

$$c = \sum_{i=2}^{k} c(n_{i-1}, n_i).$$

he following discussion is simplified if we define an edge element as the boundary between two pixels p and q, such that p and q are 4-neighbors, as Fig.3.4 illustrates. Edge elements are identified by the xy-coordinates of points p and q. In other words, the edge element in Fig. 3.4 is defined by the pairs (x_p, y_p) (x_q, y_q). Consistent with the definition an edge is a sequence of connected edge elements.

We can illustrate how the concepts just discussed apply to edge detection using the 3 X 3 image shown in Fig. 3.5 (a). The outer numbers are pixel

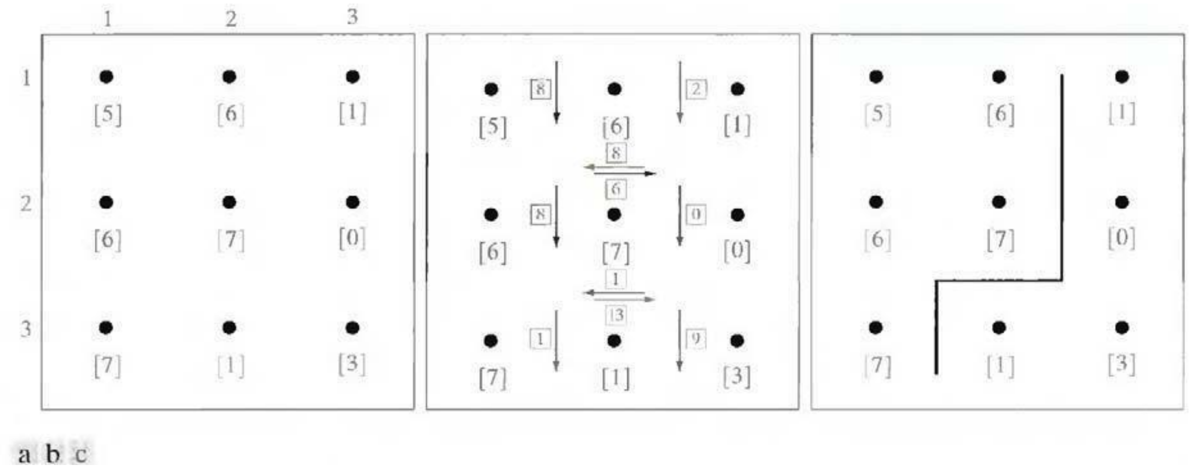


Fig.3.5 (a) A 3 X 3 image region, (b) Edge segments and their costs, (c) Edge corresponding to the lowest-cost path in the graph shown in Fig. 3.6

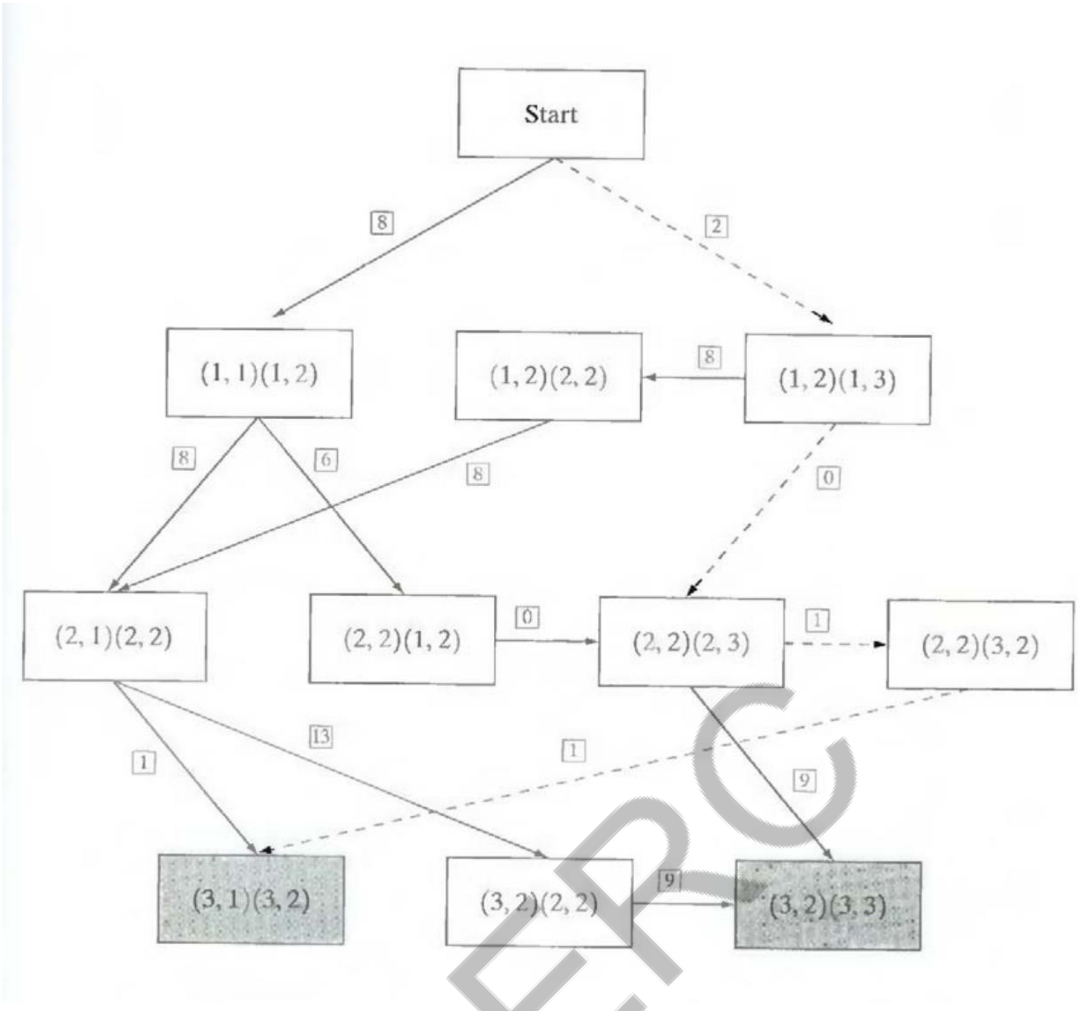
coordinates and the numbers in brackets represent gray-level values. Each edge element, defined by pixels p and q, has an associated cost, defined as

$$c(p, q) = H - [f(p) - f(q)]$$

where H is the highest gray-level value in the image (7 in this case), and f(p) and f(q) are the gray-level values of p and q, respectively. By convention, the point p is on the right-hand side of the direction of travel along edge elements. For example, the edge segment (1, 2) (2, 2) is between points (1, 2) and (2, 2) in Fig. 3.5 (b). If the direction of travel is to the right, then p is the point with coordinates (2, 2) and q is point with coordinates (1, 2); therefore, c (p, q) = 7 - [7
- 6] = 6. This cost is shown in the box below the edge segment. If, on the other hand, we are traveling to the left between the same two points, then p is point (1, 2) and q is (2, 2). In this case the cost is 8, as shown above the edge segment in Fig. 3.5(b). To simplify the discussion, we assume that edges start in the top row and terminate in the last row, so that the first element of an edge can be only between points (1, 1), (1, 2) or (1, 2), (1, 3). Similarly, the last edge element has to be between points (3, 1), (3, 2) or (3, 2), (3, 3). Keep in mind that p and q are 4-neighbors, as noted earlier. Figure 3.6 shows the graph for this problem. Each node (rectangle) in the graph corresponds to an edge element from Fig. 3.5.

-
-
-
-
-
-
-
-
-
-
-
-
-

- An arc exists between two nodes if the two corresponding edge elements taken in succession can be part of an edge.



3.6 Graph for the image in Fig.3.5 (a). The lowest-cost path is shown dashed.

As in Fig. 3.5 (b), the cost of each edge segment, is shown in a box on the side of the arc leading into the corresponding node. Goal nodes are shown shaded. The minimum cost path is shown dashed, and the edge corresponding to this path is shown in Fig. 3.5 (c).

Thresholding.

Thresholding:

Because of its intuitive properties and simplicity of implementation, image thresholding enjoys a central position in applications of image segmentation.

Global Thresholding:

The simplest of all thresholding techniques is to partition the image histogram by using a single global threshold, T. Segmentation is then accomplished by scanning the image pixel by pixel and labeling each pixel as object or back-ground, depending on whether the gray level of that pixel is greater or less than the value of T. As indicated earlier, the success of this method depends entirely on how well the histogram can be partitioned.

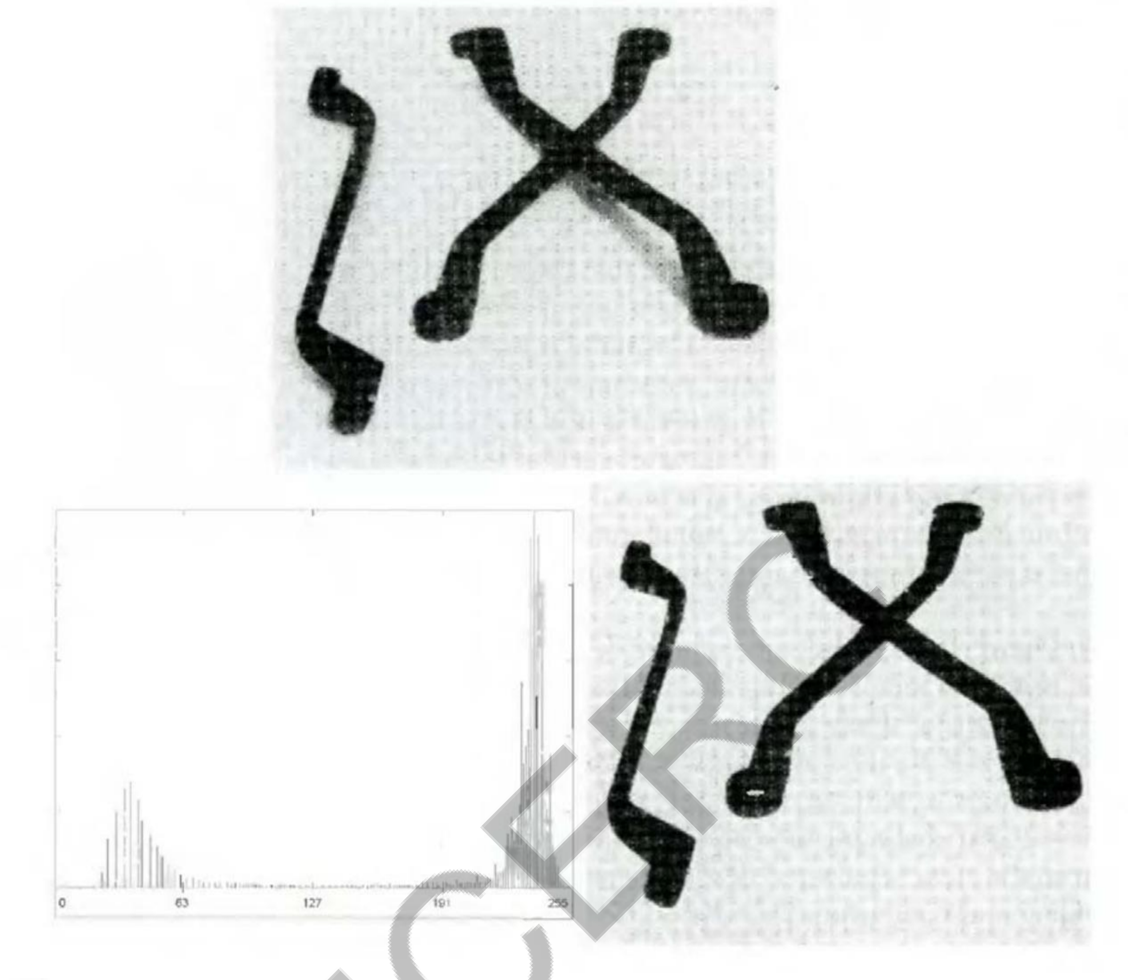


Fig.4.1 FIGURE 10.28 (a) Original image, (b) Image histogram, (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

Figure 4.1(a) shows a simple image, and Fig. 4.1(b) shows its histogram. Figure 4.1(c) shows the result of segmenting Fig. 4.1(a) by using a threshold T midway between the maximum and minimum gray levels. This threshold achieved a "clean" segmentation by eliminating the shadows and leaving only the objects themselves. The objects of interest in this case are darker than the background, so any pixel with a gray level ≤ T was labeled black (0), and any pixel with a gray level ≥ T was labeled white (255).The key objective is merely to generate a binary image, so the black-white relationship could be reversed. The type of global thresholding just described can be expected to be successful in highly controlled environments. One of the areas in which this often is possible is in industrial inspection applications, where control of the illumination usually is feasible.

The threshold in the preceding example was specified by using a heuristic approach, based on visual inspection of the histogram. The following algorithm can be used to obtain T automatically:

1. Select an initial estimate for T.

2. Segment the image using T. This will produce two groups of pixels: G_1 consisting of all pixels with gray level values >T and G_2 consisting of pixels with values < T.

3. Compute the average gray level values μ_1 and μ_2 for the pixels in regions G_1 and G_2.

4. Compute a new threshold value:

$$T = \frac{1}{2}(\mu_1 + \mu_2).$$

5. Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than apredefined parameter T_o.

When there is reason to believe that the background and object occupy comparable areas in the image, a good initial value for T is the average gray level of the image. When objects are small compared to the area occupied by the background (or vice versa), then one group of pixels will dominate the histogram and the average gray level is not as good an initial choice. A more appropriate initial value for T in cases such as this is a value midway between the maximum and minimum gray levels. The parameter T_o is used to stop the algorithm after changes become small in terms of this parameter. This is used when speed of iteration is an important issue.

Basic adaptive thresholding process used in image segmentation.

Basic Adaptive Thresholding:

Imaging factors such as uneven illumination can transform a perfectly segmentable histogram into a histogram that cannot be partitioned effectively by a single global threshold. An approach for handling such a situation is to divide the original image into subimages and then utilize a different threshold to segment each subimage. The key issues in this approach are how to subdivide the image and how to estimate the threshold for each resulting subimage. Since the threshold used for each pixel depends on the location of the pixel in terms of the subimages, this type of thresholding is adaptive.

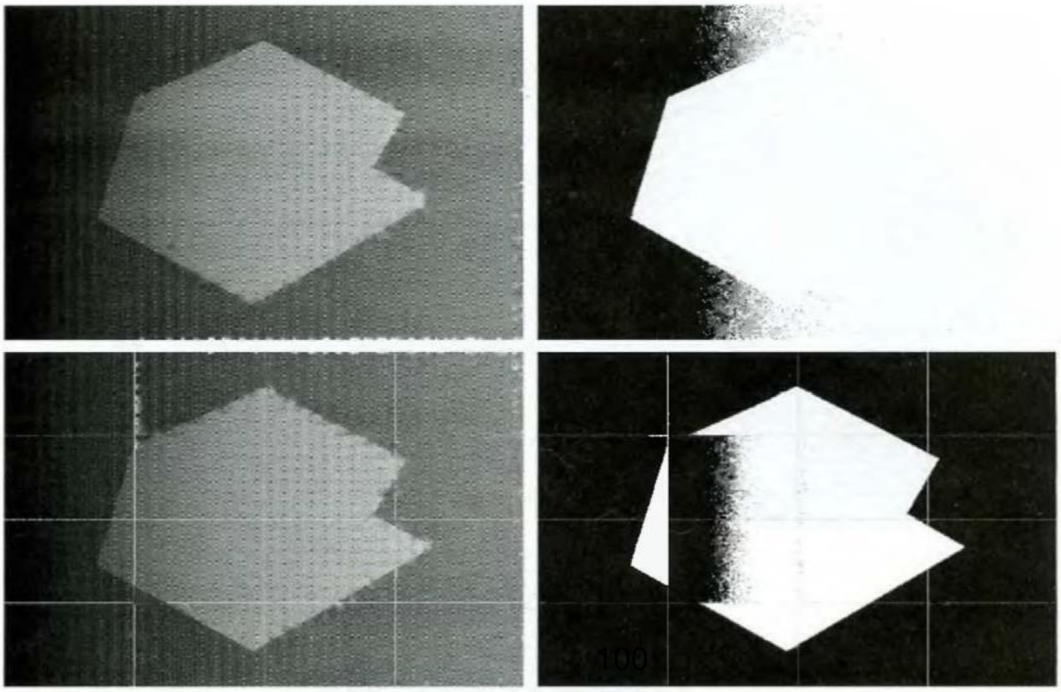


Fig.5 (a) Original image, (b) Result of global thresholding. (c) Image subdivided into individual subimages (d) Result of adaptive thresholding.

We illustrate adaptive thresholding with a example. Figure 5(a) shows the image, which we concluded could not be thresholded effectively with a single global threshold. In fact, Fig. 5(b) shows the result of thresholding the image with a global threshold manually placed in the valley of its histogram. One approach to reduce the effect of nonuniform illumination is to subdivide the image into smaller subimages, such that the illumination of each subimage is approximately uniform. Figure 5(c) shows such a partition, obtained by subdividing the image into four equal parts, and then subdividing each part by four again. All the subimages that did not contain a boundary between object and back-ground had variances of less than 75. All subimages containing boundaries had variances in excess of 100. Each subimage with variance greater than 100 was segmented with a threshold computed for that subimage using the algorithm. The initial
value for T in each case was selected as the point midway between the minimum and maximum gray levels in the subimage. All subimages with variance less than 100 were treated as one composite image, which was segmented using a single threshold estimated using the same algorithm. The result of segmentation using this procedure is shown in Fig. 5(d).

With the exception of two subimages, the improvement over Fig. 5(b) is evident. The boundary between object and background in each of the improperly segmented subimages was small and dark, and the resulting histogram was almost unimodal.

Threshold selection based on boundary characteristics.

Use of Boundary Characteristics for Histogram Improvement and Local Thresholding:

It is intuitively evident that the chances of selecting a "good" threshold are enhanced considerably if the histogram peaks are tall, narrow, symmetric, and separated by deep valleys. One approach for improving the shape of histograms is to consider only those pixels that lie on or near the edges between objects and the background. An immediate and obvious improvement is that histograms would be less dependent on the relative sizes of objects and the background. For instance, the histogram of an image composed of a small object on a large background area (or vice versa) would be dominated by a large peak because of the high concentration of one type of pixels.

If only the pixels on or near the edge between object and the background were used, the resulting histogram would have peaks of approximately the same height. In addition, the probability that any of those given pixels lies on an object would be approximately equal to the probability that it lies on the back-ground, thus improving the symmetry of the histogram peaks.

Finally, as indicated in the following paragraph, using pixels that satisfy some simple measures based on gradient and Laplacian operators has a tendency to deepen the valley between histogram peaks.

The principal problem with the approach just discussed is the implicit assumption that the edges between objects and background arc known. This information clearly is not available during segmentation, as finding a division between objects and background is precisely what segmentation is all about. However, an indication of whether a pixel is on an edge may be obtained by computing its gradient. In addition, use of the Laplacian can yield information regarding whether a given pixel lies on the dark or light side of an edge. The average value of the Laplacian is 0 at the transition of an edge, so in practice the valleys of histograms formed from

the pixels selected by a gradient/Laplacian criterion can be expected to be sparsely populated.

This property produces the highly desirable deep valleys.

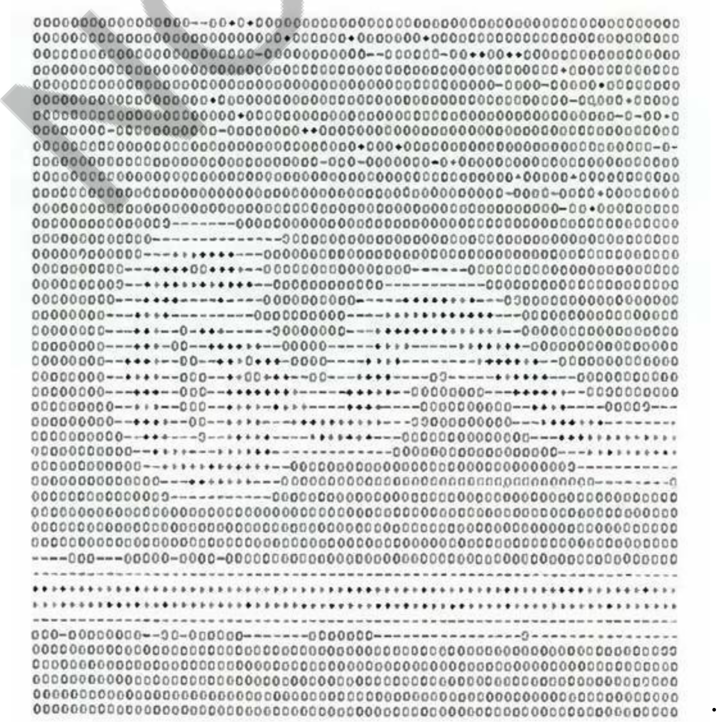
The gradient at any point (x, y) in an image can be found. Similarly, the Laplacian ^{2}f can also be found. These two quantities may be used to form a three-level image, as follows:

$$s(x, y) = \begin{cases} 0 & \text{if } \nabla f < T \\ + & \text{if } \nabla f \geq T \quad \text{and} \quad \nabla^2 f \geq 0 \\ - & \text{if } \nabla f \geq T \quad \text{and} \quad \nabla^2 f < 0 \end{cases}$$

where the symbols 0, +, and - represent any three distinct gray levels, T is a threshold, and the gradient and Laplacian are computed at every point (x, y). For a dark object on a light background, the use of the Eqn. produces an image s(x, y) in which (1) all pixels that are not on an edge (as determined by being less than T) are labeled 0; (2) all pixels on the dark side of an edge are labeled +; and (3) all pixels on the light side of an edge are labeled -. The symbols + and - in Eq. above are reversed for a light object on a dark background. Figure 6.1 shows the labeling produced by Eq. for an image of a dark, underlined stroke written on a light background.

The information obtained with this procedure can be used to generate a segmented, binary image in which l's correspond to objects of interest and 0's correspond to the background. The transition (along a horizontal or vertical scan line) from a light background to a dark object must be characterized by the occurrence of a - followed by a + in s (x, y). The interior of the object is composed of pixels that are labeled either 0 or +. Finally, the transition from the object back to the background is characterized by the occurrence of a + followed by a -. Thus a horizontal or vertical scan line containing a section of an object has the following structure:

$$(\cdots)(-, +)(0 \text{ or } +)(+, -)(\cdots)$$



.

Fig.6.1 Image of a handwritten stroke coded by using Eq. discussed above

where (…) represents any combination of +, -, and 0. The innermost parentheses contain object points and are labeled 1. All other pixels along the same scan line are labeled 0, with the exception of any other sequence of (- or +) bounded by (-, +) and (+, -).

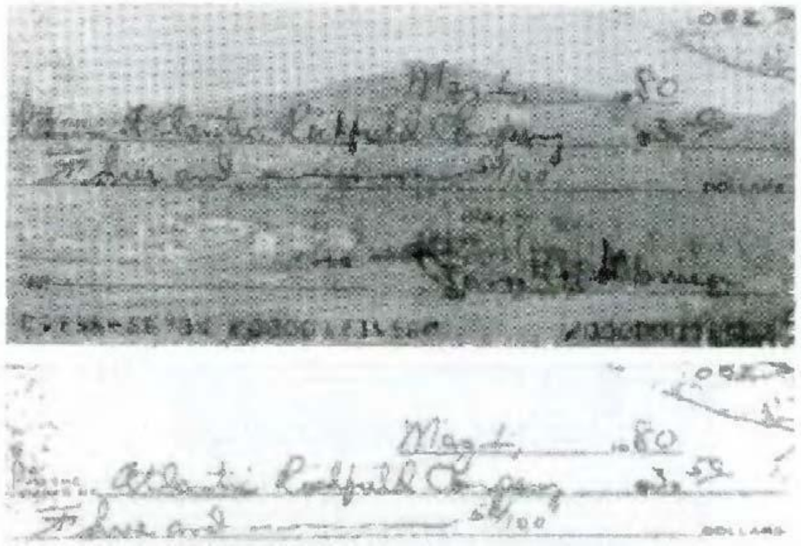


Fig.6.2 (a) Original image, (b) Image segmented by local thresholding.

figure 6.2 (a) shows an image of an ordinary scenic bank check. Figure 6.3 shows the histogram as a function of gradient values for pixels with gradients greater than 5. Note that this histogram has two dominant modes that are symmetric, nearly of the same height, and arc separated by a distinct valley. Finally, Fig. 6.2(b) shows the segmented image obtained by with T at or near the midpoint of the valley. Note that this example is an illustration of local thresholding, because the value of T was determined from a histogram of the gradient and Laplacian, which are local properties.

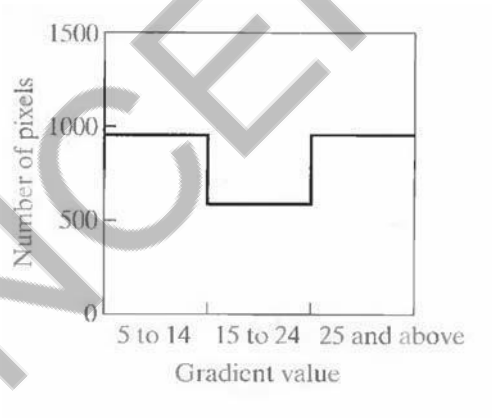


Fig.6.3 Histogram of pixels with gradients greater than 5

Region based segmentation.

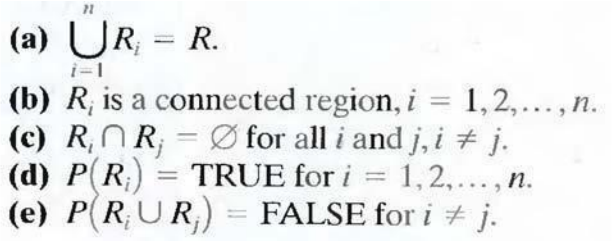
Region-Based Segmentation:

The objective of segmentation is to partition an image into regions. We approached this problem by finding boundaries between regions based on discontinuities in gray levels, whereas segmentation was accomplished via thresholds based on the distribution of pixel properties, such as gray-level values or color.

Basic Formulation:

Let R represent the entire image region. We may view segmentation as a process that partitions R into n subregions, R_1, R $_2$..., R_n, such that

\



(a) $\bigcup_{i=1}^{n} R_i = R.$

(b) R_i is a connected region, $i = 1, 2, \ldots, n.$

(c) $R_i \cap R_j = \varnothing$ for all i and $j, i \neq j.$

(d) $P(R_i) =$ TRUE for $i = 1, 2, \ldots, n.$

(e) $P(R_i \cup R_j) =$ FALSE for $i \neq j.$

Here, P (R_i) is a logical predicate defined over the points in set R_i and Ǿ` is the null set. Condition (a) indicates that the segmentation must be complete; that is, every pixel must be in a region. Condition (b) requires that points in a region must be connected in some predefined sense. Condition (c) indicates that the regions must be disjoint. Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region—for example P (R_i) = TRUE if all pixels in R_i, have the same gray level. Finally, condition (c) indicates that regions R_i and R_j are different in the sense of predicate P.

Region Growing:

As its name implies, region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria. The basic approach is to start with a set of "seed" points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed (such as specific ranges of gray level or color). When a priori information is not available, the procedure is to compute at every pixel the same set of properties that ultimately will be used to assign pixels to regions during the growing process. If the result of these computations shows clusters of values, the pixels whose properties place them near the centroid of these clusters can be used as seeds.

The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available. For example, the analysis of land-use satellite imagery depends heavily on the use of color. This problem would be significantly more difficult, or even impossible, to handle without the inherent information available in color images. When the images are monochrome, region analysis must be carried out with a set of descriptors based on gray levels and spatial properties (such as moments or texture).

Basically, growing a region should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as gray level, texture, and color, are local in nature and do not take into account the "history" of region growth. Additional criteria that increase the power of a region-growing algorithm utilize the concept of size, likeness between a candidate pixel and the pixels grown so far (such as a comparison of the gray level of a candidate and the average gray level of the grown region), and the shape of the region being grown. The use of these types of descriptors is based on the assumption that a model of expected results is at least partially available.

Figure 7.1 (a) shows an X-ray image of a weld (the horizontal dark region) containing several cracks and porosities (the bright, white streaks running horizontally through the middle of the image). We wish to use region growing to segment the regions of the weld failures. These segmented features could be used for inspection, for inclusion in a database of historical studies, for controlling an automated welding system, and for other numerous applications.

Fig.7.1 (a) Image showing defective welds, (b) Seed points, (c) Result of region growing, (d) Boundaries of segmented ; defective welds (in black).

The first order of business is to determine the initial seed points. In this application, it is known that pixels of defective welds tend to have the maximum allowable digital value B55 in this case). Based on this information, we selected as starting points all pixels having values of 255. The points thus extracted from the original image are shown in Fig. 10.40(b). Note that many of the points are clustered into seed regions.

The next step is to choose criteria for region growing. In this particular example we chose two criteria for a pixel to be annexed to a region: (1) The absolute gray-level difference between any pixel and the seed had to be less than 65. This number is based on the histogram shown in Fig. 7.2 and represents the difference between 255 and the location of the first major valley to the left, which is representative of the highest gray level value in the dark weld region. (2) To be included in one of the regions, the pixel had to be 8-connected to at least one pixel in that region.

If a pixel was found to be connected to more than one region, the regions were merged. Figure 7.1 (c) shows the regions that resulted by starting with the seeds in Fig. 7.2 (b) and utilizing the criteria defined in the previous paragraph. Superimposing the boundaries of these regions on the original image [Fig. 7.1(d)] reveals that the region-growing procedure did indeed segment the defective welds with an acceptable degree of accuracy. It is of interest to note that it was not necessary to specify any stopping rules in this case because the criteria for region growing were sufficient to isolate the features of interest.

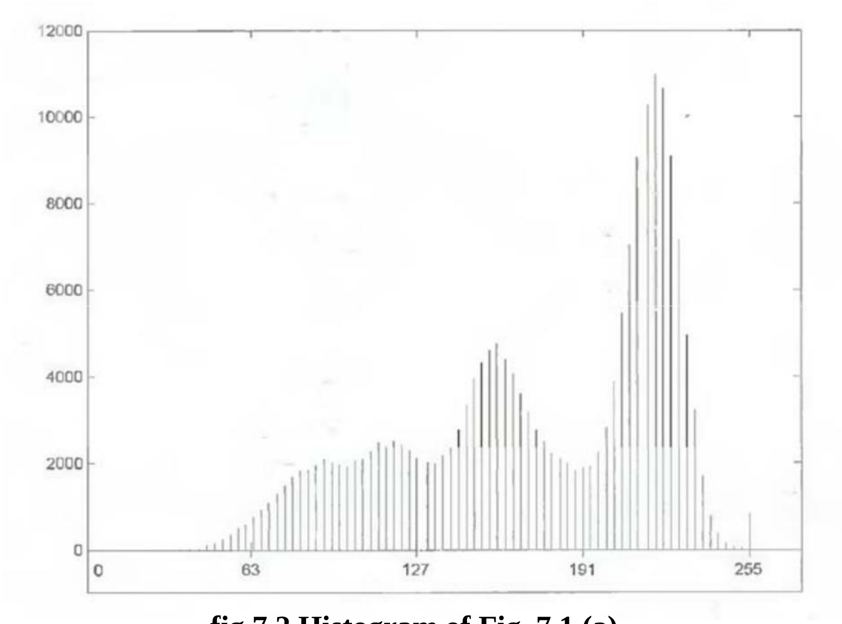


fig.7.2 Histogram of Fig. 7.1 (a)

Region Splitting and Merging:

The procedure just discussed grows regions from a set of seed points. An alternative is to subdivide an image initially into a set of arbitrary, disjointed regions and then merge and/or split the regions in an attempt to satisfy the conditions. A split and merge algorithm that iteratively works toward satisfying these constraints is developed.

Let R represent the entire image region and select a predicate P. One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $P(R_i)$ = TRUE. We start with the entire region. If P(R) = FALSE, we divide the image into quadrants. If P is FALSE for any quadrant, we subdivide that quadrant into subquadrants, and so on. This particular splitting technique has a convenient representation in the form of a so-called quadtree (that is, a tree in which nodes have exactly four descendants), as illustrated in Fig. 7.3. Note that the root of the tree corresponds to the entire image and that each node corresponds to a subdivision. In this case, only R_4 was subdivided further.

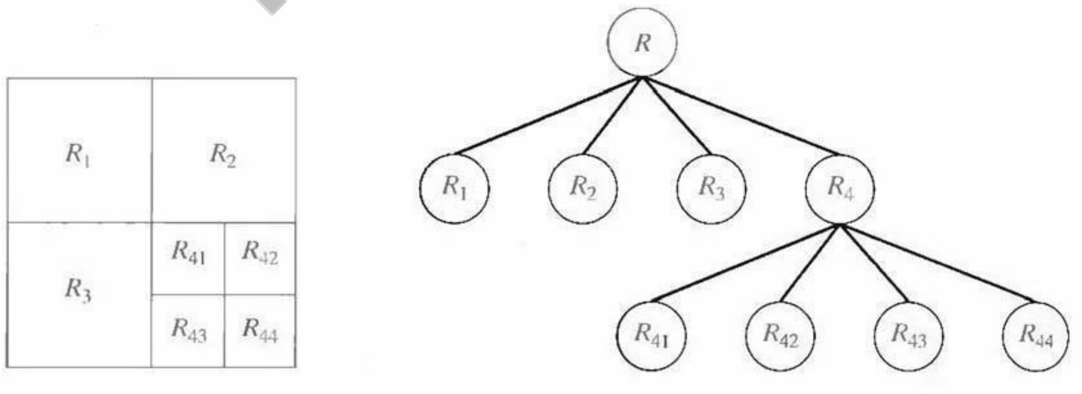


Fig. 7.3 (a) Partitioned image (b) Corresponding quadtree.

If only splitting were used, the final partition likely would contain adjacent regions with identical properties. This drawback may be remedied by allowing merging, as well as splitting. Satisfying the constraints, requires merging only adjacent regions whose combined pixels satisfy the predicate P. That is, two adjacent regions R_j and R_k are merged only if P (R_j U R_k) = TRUE.

The preceding discussion may be summarized by the following procedure, in which, at any step we

1. Split into four disjoint quadrants any region R_i, for which P (R_i) = FALSE.

2. Merge any adjacent regions R_j and R_k for which P (R_j U R_k) = TRUE.
3. Stop when no further merging or splitting is possible.

Several variations of the preceding basic theme are possible. For example, one possibility is to split the image initially into a set of blocks. Further splitting is carried out as described previously, but merging is initially limited to groups of four blocks that are descendants in the quadtree representation and that satisfy the predicate P. When no further mergings of this type are possible, the procedure is terminated by one final merging of regions satisfying step 2. At this point, the merged regions may be of different sizes. The principal advantage of this approach is that it uses the same quadtree for splitting and merging, until the final merging step.

Digital image processing :Morphological operations

The term morphology is being used in a variety of streams like linguistics, biology, astronomy, mathematics, and it is also used with prefixes like Geo-morphology, River morphology, urban morphology, etc. The term morphology used in image processing refers to the tools which are developed using the theory developed as part of mathematical morphology.

Morphology	In its most general form, the term morphology refers to a branch of biology that deals with Form and Structure of animals and plants.
Mathematical Morphology	The term has been used in mathematics where it deals with Form and Structure of regions.
Morphology in Image Processing	In image processing the term morphology deals with developing tools for extracting Form and Structure of image regions (objects).

Extraction of features from in image is the first step towards image analysis. Morphology plays an important role in image processing because it can be used to develop techniques for feature extraction in binary images.

Input Image
Morphological Tools
Output Image

Image components generally used for describing region shapes are:

- **Boundaries**
- **Skeletons**
- **Convex Hulls**

Morphological techniques are used for pre-processing and post-processing:

- **to identify and enhance useful features,**
- **to discard (prune) noisy features.**

Mathematical operations are applied to shapes/ objects. But how to represent shapes or objects in images?

Objects in Morphology

- Objects are represented as Sets.
- For binary images, each element of a set is (x;y) coordinates of white/ black pixel. These elements are $\in Z^2$ (2-D integer space). Note that we don't have to explicitly code the binary value as part of the pixel representation. Since there are only 2 possible pixels (black or white) in the image, we can form a set of white pixels. All other pixels are implied to be black.
- For grayscale images such sets are $\in Z^3$ i.e. 3-D integer space. The first 2 integers in the 3-tuple are the x,y coordinates and the third integer is the intensity value.

Morphology involves the use of subimages called as structuring elements. The pixels in a structuring element can have values 0 (black), 1 (white), or may even be don't care (either black or white). The structuring element is used to assess or probe the attributes and properties of the images under study.

The origin of the structuring element is generally taken as the center of the rectangular array which contains the structuring element. However the origin need not be specified as the center. Changing the origin of the structuring element also changes the output of the morphological operations.

- We talk of morphological operations between two image objects.
- The first one is the object/ region under study.
- The second one is an object (a subimage depicting a region) used to *probe* the first one to identify its structural characteristics.
- All sets are padded with background elements to form a rectangular array or to provide a background border.

The structuring element is also called as a mask or a kernel.

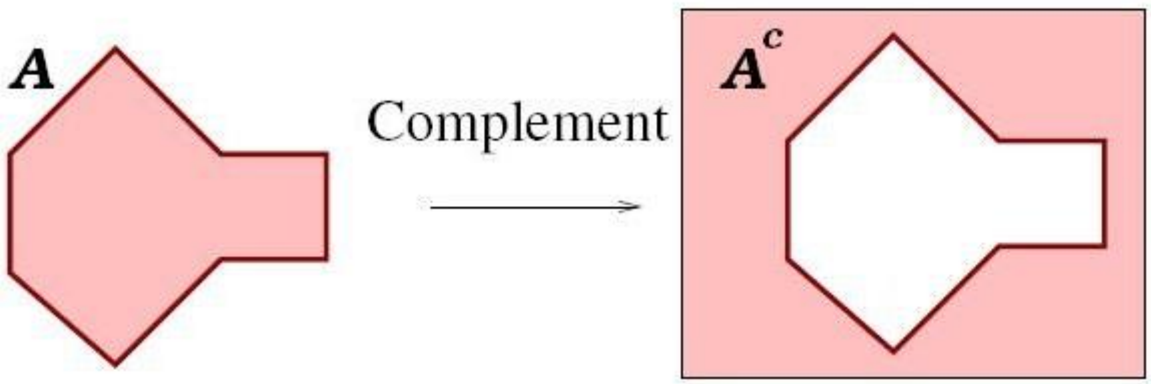


Figure 1: Complement of a set

Translation and reflection are set operations which do not involve any structuring element. Translation of a set means that each element of the set is displaced by a fixed translation distance. Reflection of a set means that the coordinate of each pixel will shift to the other side of the axis. So x becomes - x and y becomes - y.

Reflection of a set

The reflection of a set B is denoted as $\hat{B}$

$$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$$

Translation of a set

The translation of a set B by z is denoted as $(B)_z$

$$(B)_z = \{c \mid c = b + z, \text{ for } b \in B\}$$

Set intersection

This is the traditional intersection of two sets. If the sets indicate image regions, then their intersection would give the region overlap.

Set union

This is the traditional union of two sets. If the sets indicate image regions, then their union would give the aggregate of the two regions.

The basic morphological operations which can be performed using a structuring element are erosion and dilation.

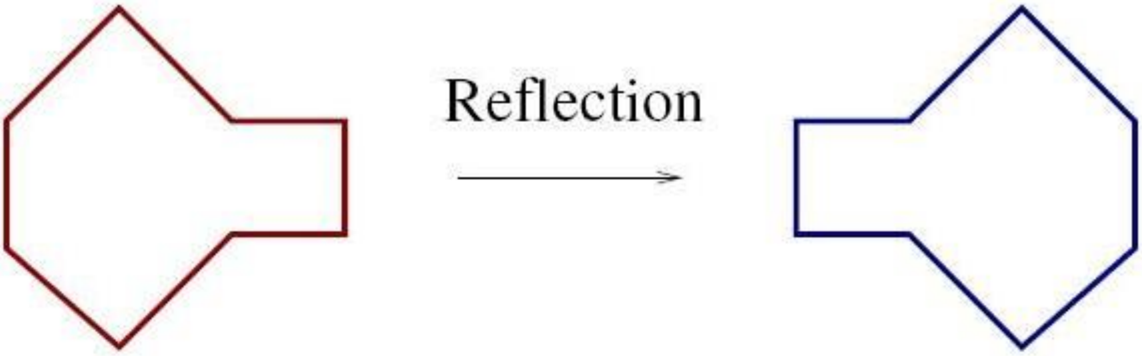


Figure 2: Reflection of a set.

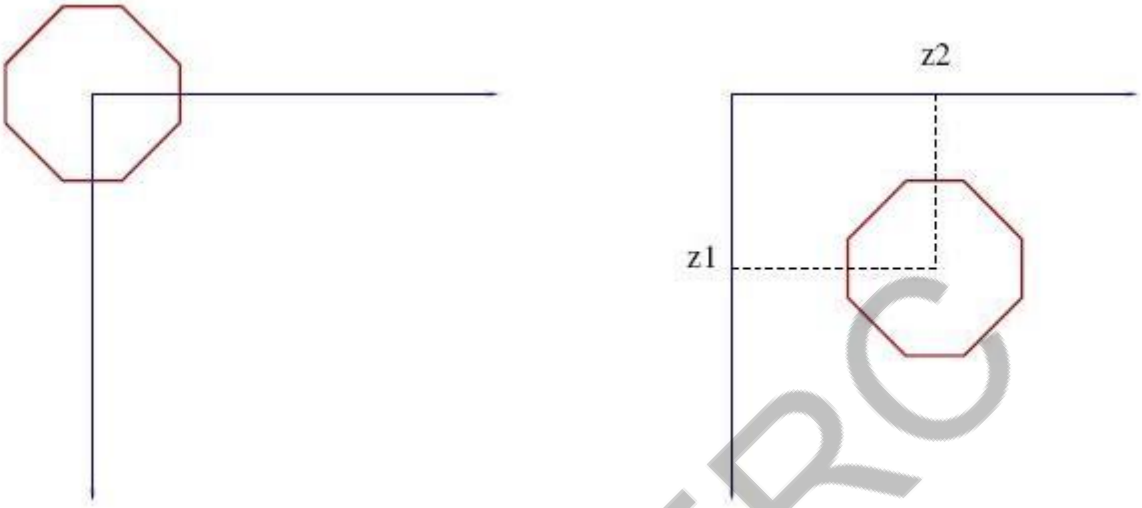


Figure 3: Translation of a set

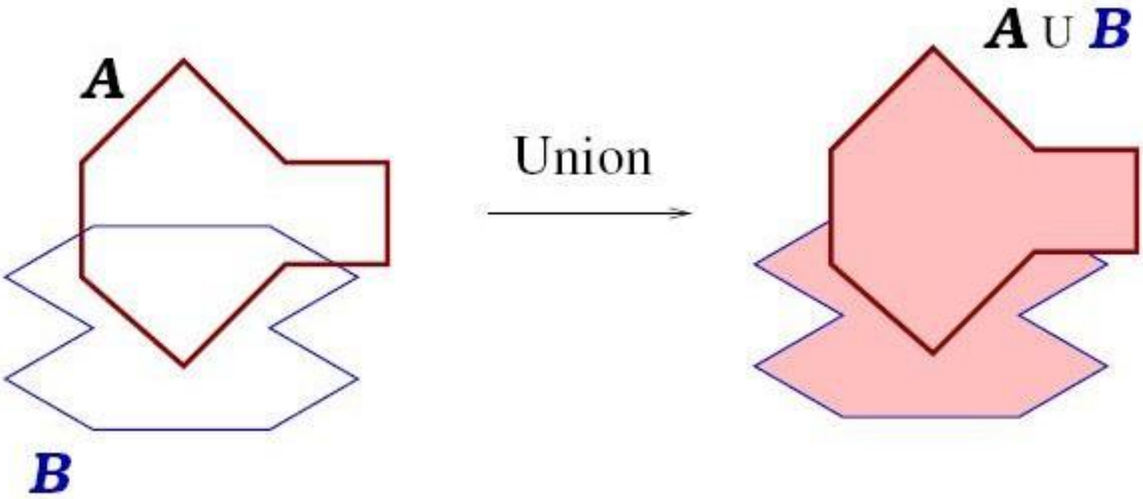


Figure 4: Union of two sets A and B

Alternatively it can be formulated as:

$$A \ominus B \;=\; \{\, z \mid (B)_z \cap A^c = \emptyset \,\}$$

Erosion can be viewed as a morphological filtering operation. Image details smaller than the structuring element B are filtered out (removed) from the image.

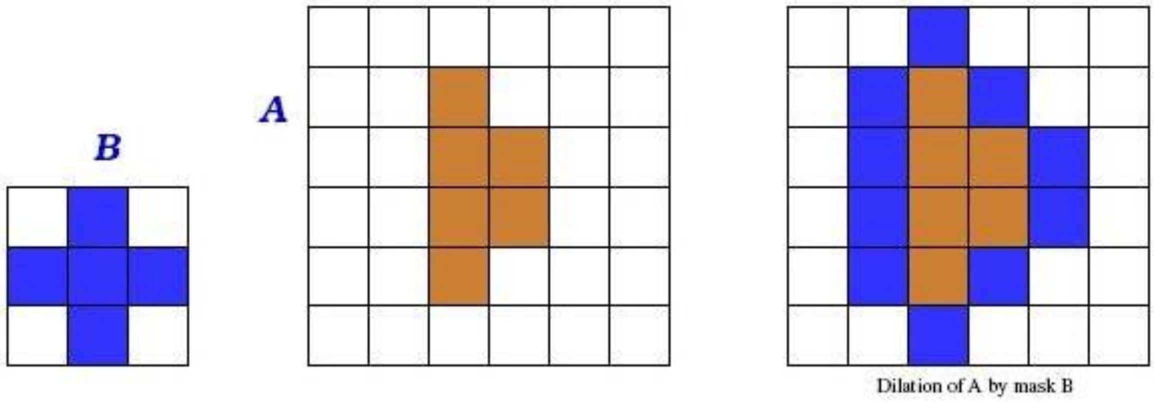


Figure 8: Dilation of a set A using structuring element B.

Dilation of a given set (equivalently the region) yields all the points where the origin of the structuring element can be placed while satisfying the condition that there is some overlap of the structuring element with the given region.

Erosion of a set shrinks it while dilation expands it. There exists a duality between erosion and dilation. If we erode a set and then take its complement, the result will be the same as dilating the set's complement i.e. the background.

Dilation of a subimage(set) A, by a subimage(set) B is

> set of all displacements z, such that $(\hat{B})_z$, i.e. $\hat{B}$ translated by z, and A OVERLAP at at least one element.

As a result of dilation

- Extra elements are added to A i.e. it grows (dilates) as long as $(\hat{B})_z$ overlaps with A.
- The amount of dilation depends on the size of B. Larger the B → more dilation.

The dilation operation is denoted by the symbol $\oplus$ and defined as: $A \oplus B \;=\; \{\, z \mid (\hat{B})_z \cap A \neq \emptyset \,\}$

Alternatively it can be formulated as:

$$A \oplus B \;=\; \{\, z \mid [(\hat{B})_z \cap A] \subseteq A \,\}$$

Dilation can be viewed as a morphological "reconstruction" operation. Image details smaller than the structuring element B are filled-up in the image.

Analogy with convolution

- Dilation involves flipping B about its origin and then successively displacing it so that it slides over A.
- If SE is symmetric $\hat{B} = B$

Duality between Erosion and Dilation

The erosion and dilation operations are dual of each other.

$$(A \ominus B)^c = A^c \oplus \widehat{B}$$
$$(A \oplus B)^c = A^c \ominus \widehat{B}$$

Opening of a set by a structuring element yields all the points where the structuring element would overlap (encompass) while satisfying the condition that the structuring element is completely within the set. Opening is performed by applying erosion followed by dilation. Closing is performed by applying dilation followed by erosion.

We give here a comparison of the opening and closing operations.

Opening	Closing
• Smooths contours of an image • Breaks narrow isthmuses • Eliminates thin protrusions	• Smooths sections of contours • Fuses narrow breaks and long thin gulfs • Eliminates small holes and fills gaps in the contour.

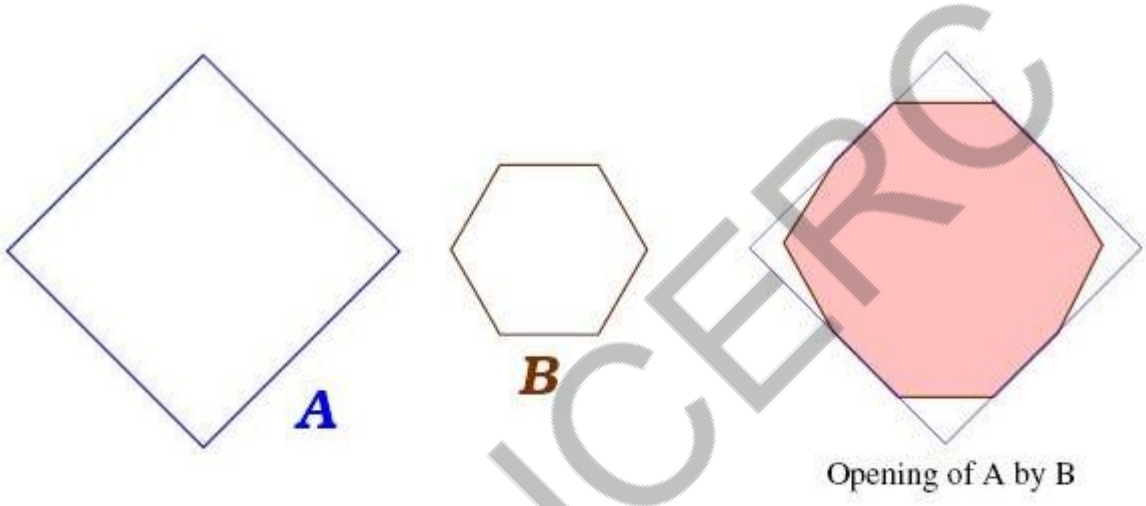


Figure 9: Opening of a set A using structuring element B.

1. $A \circ B = (A \ominus B) \oplus B$

2. Roll B on the inside of the boundary of A.

3. New boundary on A is defined by points on B which are closest to the boundary of A.

1. $A \bullet B = (A \oplus B) \ominus B$

2. Roll B on the outside of the boundary of A.

3. New boundary on A is defined by points on B which are closest to the boundary of A.

Consider the closing operation. Dilation expands a set and erosion shrinks it. It seems that the result will be the same as the original set since we have applied two opposite operations. But this does not happen if the original set has got grooves or gulfs which are smaller in size compared to the structuring element. Once the gulf gets filled up because of dilation, subsequent erosion will not be able to reconstruct it.

DUALITY between opening and closing

Opening and closing are dual operations. Closing a set and then taking its complement will yield the same result as opening the complement of the set with the same structuring element.

$$(A \bullet B)^c = (A^c \circ \hat{B})$$

$$(A \circ B)^c = (A^c \bullet \hat{B})$$

The opening operation satisfies the following properties:

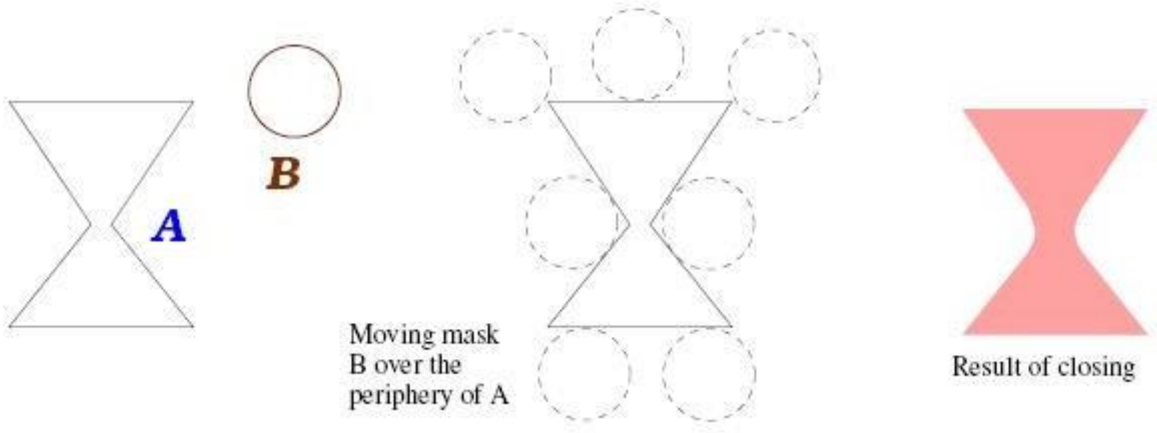


Figure 10: Closing of a set A using structuring element B.

1) $A \circ B$ is a subset (subimage) of A.
2) If C is a subset of D, then $C \circ B$ is a subset of $D \circ B$
3) $(A \circ B) \circ B = A \circ B$

The closing operation satisfies the following properties:

1) A is a subset (subimage) of $A \bullet B.$
2) If C is a subset of D, then $C \bullet B$ is a subset of $D \bullet B$

The **Hit-or-Miss transform** is used for detecting shapes. It uses two structuring elements.

1. The first one contains the foreground shape of the object which is to be detected.

2. The second structuring element contains the background shape around the object which is to be detected. It is like a window frame (a thin strip of background) to the foreground in the first structuring element. The background pixels in this mask are marked with foreground intensity and the object pixels with background intensity.

At any point on the given image,

if the foreground matches with the first structuring element

AND

if the complement (i.e. the background) matches with the second structuring element, then we can say that the object shape exists at that point.

The set of points where a structuring element fits can be identified by eroding the given image with

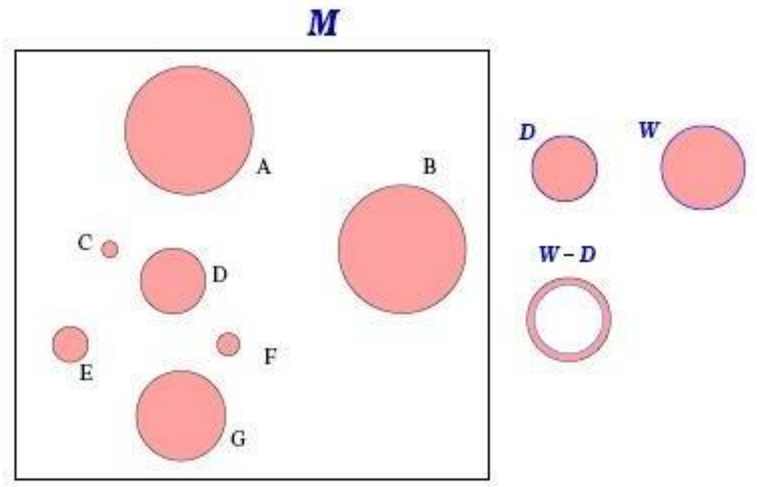


Figure 11: The set M is the union of sets A,B, C, D, E, F, G. The object we want to detect is D. Hence we use a structuring element which is same as D. The element W has been chosen slightly larger than D so that W - D gives a thin strip of background surrounding set D.

the structuring element.

Mathematical formulation

Given a shape A with 3 components shapes C, D, E. What is the location of D?

- Step 1: Detect where shape D can be. We might detect larger shapes in which D is contained.
- Step 2: Verify if background (fitting D) is present exactly around this shape.

Mathematically Hit-or-Miss transform is:

$$A \circledast B = (A \ominus D) \cap [A^c \ominus (W - D)]$$

$$A \circledast B = (A \ominus D) - \left[A \oplus (\widehat{W - D})\right]$$

The procedure is illustrated in Fig 11 to 14.

Several features can be extracted from the image using the five basic morphological operations of dilation, erosion, opening, closing, and hit-or-miss transform.

Morphological Image Processing

- Morphology is concerned with image analysis methods whose outputs describe image content (i.e. extract "meaning" from an image).
- Mathematical morphology is a tool for extracting image components that can be used to represent and describe region shapes such as boundaries and skeletons.
- Morphological methods include filtering, thinning and pruning. These techniques are based on set theory. All morphology functions are defined for binary images, but most have natural extension to grayscale images.

Basic Concepts of Set Theory

A set is specified by the elements between two braces: { }. The elements of the sets are the coordinates *(x,y)* of pixels representing objects or other features in an image.

Let *A* be a set in 2D image space Z^2:

- If $a = (a_1 , a_2)$ is an element of *A*, then $a \in A$
- If a is not an element of *A*, then $a \notin A$
- Empty set is a set with no elements and is denoted by $\emptyset$
- If every element of a set *A* is also an element of another set *B*, then *A* is said to be a subset of *B*, denoted as $A \subseteq B$
- The union of two sets *A* and *B*, denoted by $C = A \cup B$
- The intersection of two sets *A* and *B*, denoted by $C = A \cap B$
- Two sets *A* and *B* are said to be disjoint, if they have no common elements. This is denoted by $A \cap B = \emptyset$

- The *complement* of a set *A* is the set of elements not contained in *A*. This is denoted by $A^c = \{w \mid w \notin A\}$
- The *difference* of two sets *A* and *B*, denoted *A* – *B*, is defined as $A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c$
- The *reflection* of set *B*, denoted $\hat{B}$, is defined as $\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$
- The *translation* of set *A* by point $z = (z_1, z_2)$, denoted $(A)_z$ is defined as $(A)_z = \{c \mid c = a + z, \text{ for } a \in A\}$

The figure below illustrates the preceding concepts.

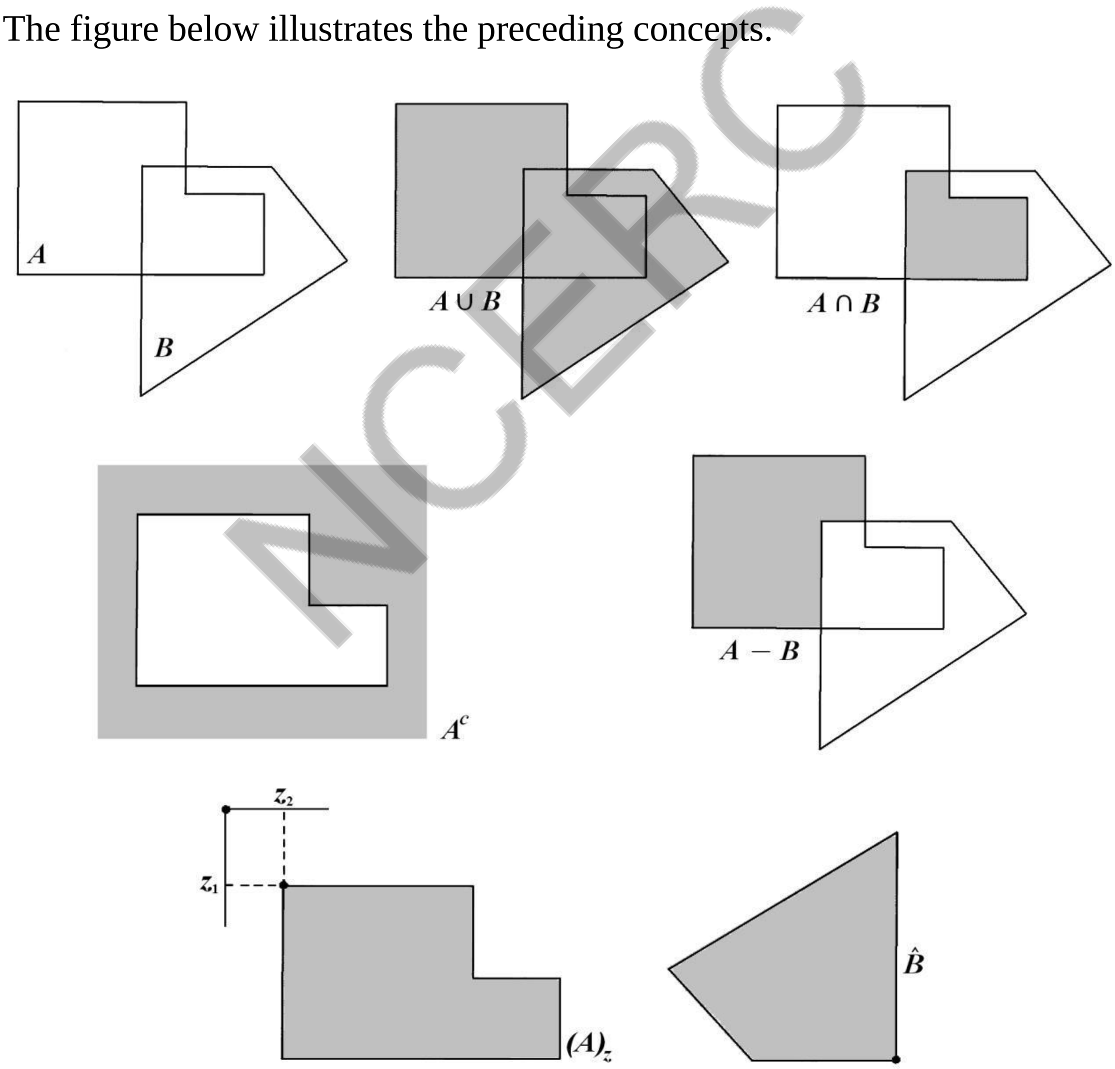


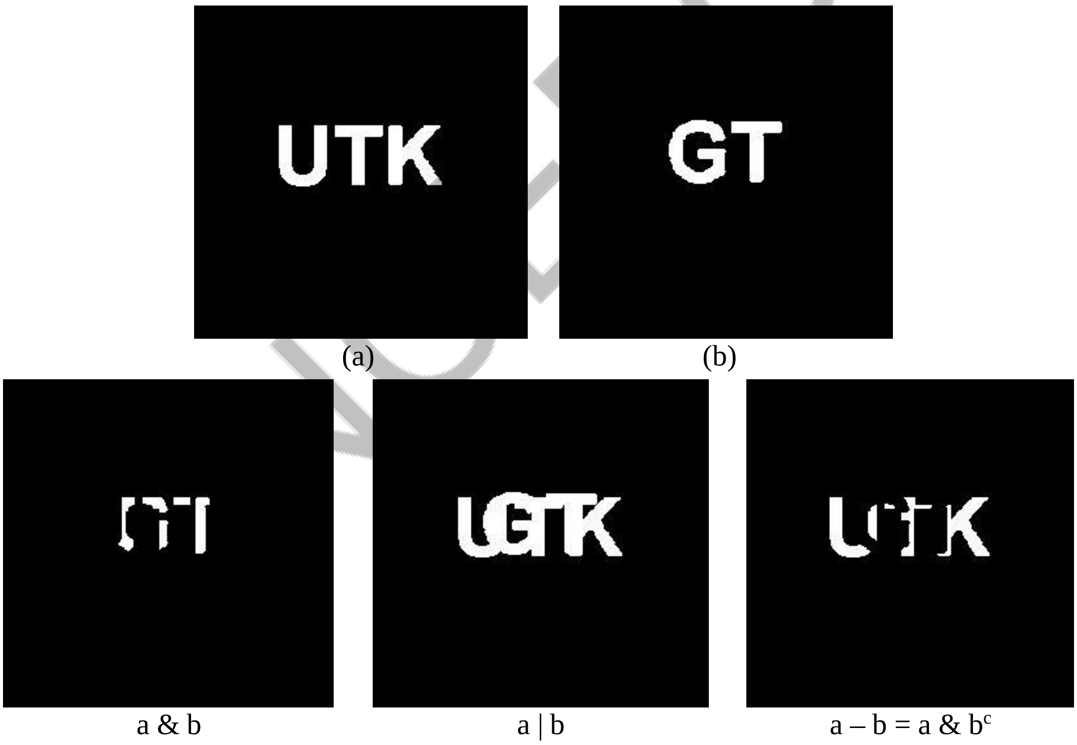
Figure 11.1 Basic concepts of Set Theory

Logic Operations Involving Binary Images

A binary image is an image whose pixel values are 0 (representing black) or 1 (representing white, i.e. 255). The usual set operations of complement, union, intersection, and difference can be defined easily in terms of the corresponding logic operations NOT, OR and AND. For example:

- Intersection operation ∩ is implemented by AND operation
- Union operation ∪ is implemented by OR operation

The figure below shows an example of using logic operations to perform set operations on two binary images.



(a) (b)

a & b a | b $a - b = a \& b^c$

Figure 11.2 Using logic operations for applying set operations on two binary images

Structuring Element

A morphological operation is based on the use of a filter-like binary pattern called the structuring element of the operation. Structuring element is represented by a matrix of 0s and 1s; for simplicity, the zero entries are often omitted.

Symmetric with respect to its origin:

Lines:

$$\begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&1\\ \hline 0&0&0&1&0\\ \hline 0&0&\boxed{1}&0&0\\ \hline 0&1&0&0&0\\ \hline 1&0&0&0&0\\ \hline \end{array} = \quad \begin{array}{ccccc} &&&&1\\ &&&1&\\ &&\boxed{1}&&\\ &1&&&\\ 1&&&& \end{array} \qquad \begin{array}{c} 1\\ \boxed{1}\\ 1 \end{array}$$

Diamond:

$$\begin{array}{ccc} 0&1&0\\ 1&\boxed{1}&1\\ 0&1&0 \end{array}$$

Non-symmetric:

$$\begin{array}{ccccccc} &&&&&&1\\ 1&1&\boxed{1}&1&1&1&\\ 1&1&&&&&\\ 1&&&&&& \end{array} \xrightarrow{\text{Reflection on origin}} \begin{array}{ccccccc} &&&&&&1\\ &&&&&1&1\\ &1&1&1&\boxed{1}&1&1\\ 1&&&&&& \end{array}$$

Dilation

Dilation is an operation used to grow or thicken objects in binary images. The dilation of a binary image *A* by a structuring element *B* is defined as:

$$A \oplus B = \{ z : (\hat{B})_z \cap A \neq \emptyset \}$$

This equation is based on obtaining the reflection of *B* about its origin and translating (shifting) this reflection by z. Then, the dilation of *A* by *B* is the set of all structuring element origin locations where the reflected and translated *B* overlaps with *A* by at least one element.

Example: Use the following structuring element to dilate the binary image below.

$$\begin{array}{ccccc} &&&&1\\ &&&1&\\ &&\boxed{1}&&\\ &1&&&\\ 1&&&& \end{array}$$

Structuring element

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

Binary image

Solution:

We find the reflection of B:

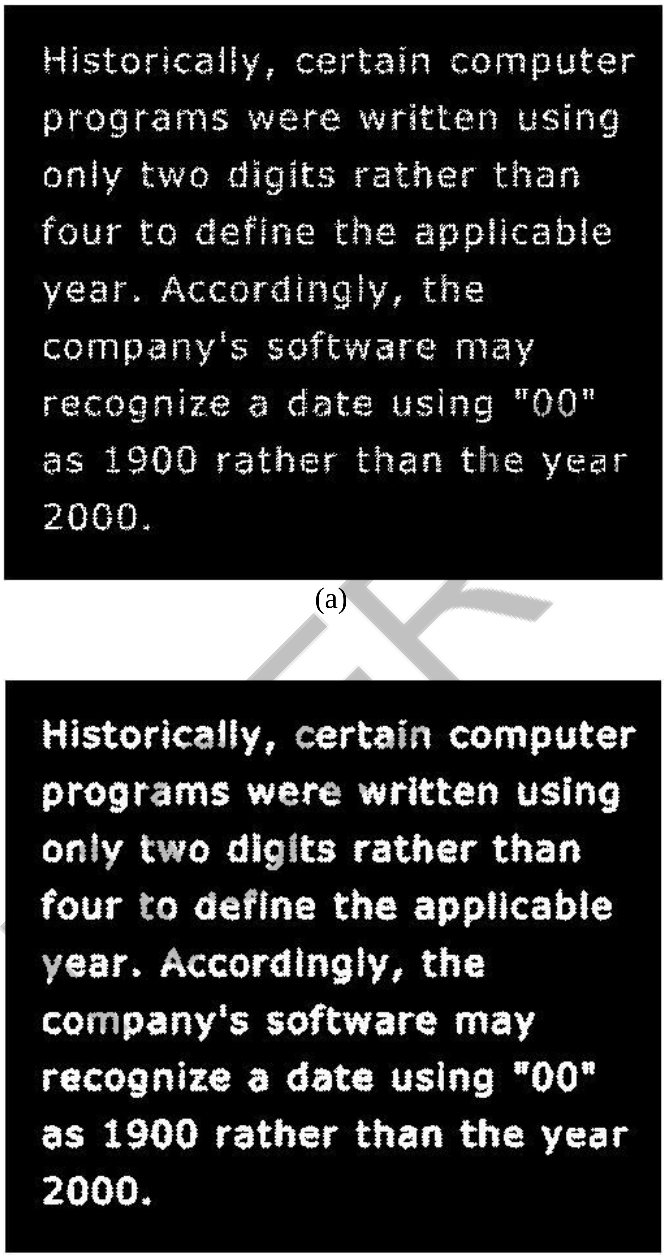
$$B = \begin{array}{ccccc} &&&&1\\ &&&1&\\ &&\boxed{1}&&\\ &1&&&\\ 1&&&& \end{array}$$

In this case $\hat{B} = B$

$A \oplus B =$

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

Dilation can be used for bridging gaps, for example, in broken/unclear characters as shown in the figure below.



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

(a)

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

(b) (b)

Figure 11.3 (a) Broken-text binary image. (b) Dilated image.

Erosion

Erosion is used to shrink or thin objects in binary images. The erosion of a binary image *A* by a structuring element *B* is defined as:

$$A \ominus B = \{\, z : \ (B)_z \cap A^c \neq \emptyset \,\}$$

The erosion of *A* by *B* is the set of all structuring element origin locations where the translated *B* does not overlap with the background of *A*.

Example: Use the following structuring element to erode the binary image below.

$$\begin{array}{c} 1 \\ \boxed{1} \\ 1 \end{array}$$

Structuring element

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

Binary image

Solution

$A \ominus B =$

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

Erosion can be used to remove isolated features (i.e. irrelevant detail) which may include noise or thin edges as shown in the figure below.

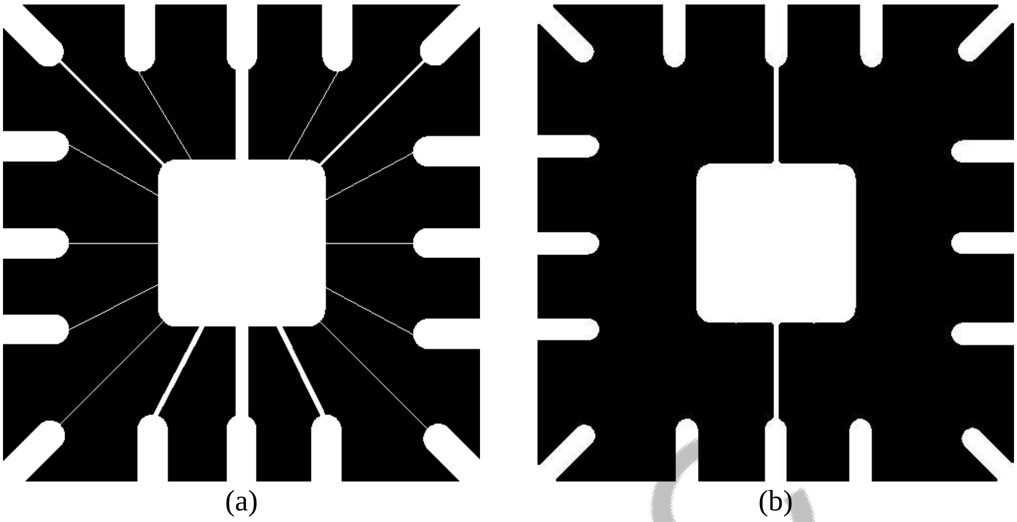


Figure 11.4 (a) Binary image. (b) Eroded image.

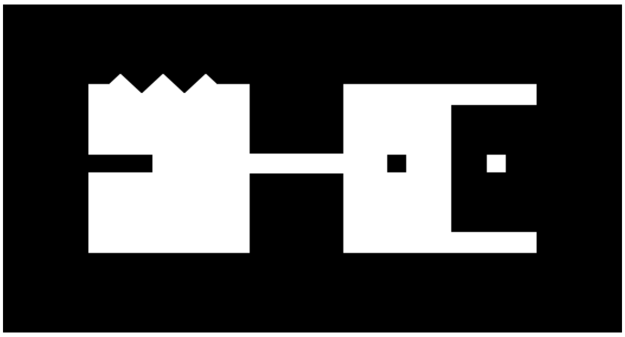
Combining Dilation & Erosion - Opening Morphology

The opening operation erodes an image and then dilates the eroded image using the same structuring element for both operations, i.e.

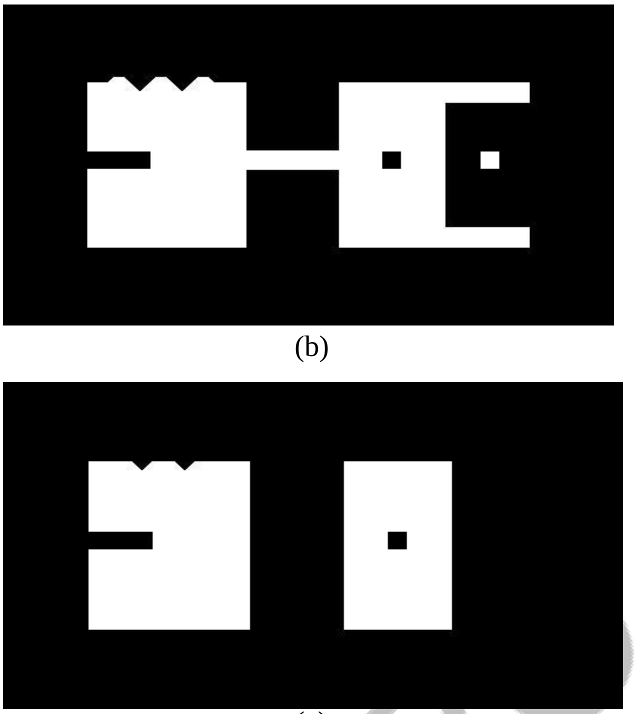
$$A \circ B = (A \ominus B) \oplus B$$

where *A* is the original image and *B* is the structuring element.

The opening operation is used to remove regions of an object that cannot contain the structuring element, smooth objects contours, and breaks thin connections as shown in the figure below.



(a)

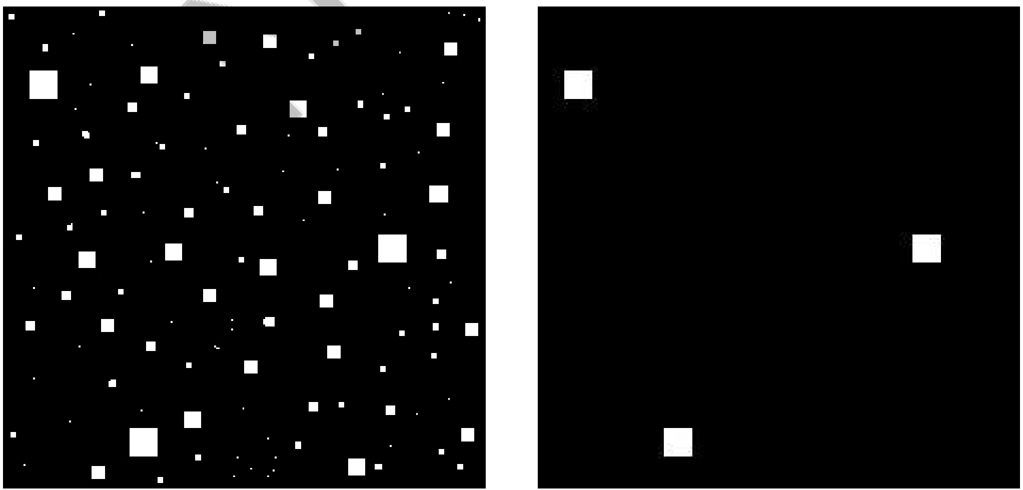


(b)

(c)

Figure 11.5 (a) Original binary image. (b) Result of opening with square structuring element of size 10 pixels. (c) Result of opening with square structuring element of size 20 pixels.

The opening operation can also be used to remove small objects in an image while preserving the shape and size of larger objects as illustrated in the figure below.



(a) (b)

Figure 11.6 (a) Original binary image. (b) Result of opening with square structuring element of size 13 pixels.

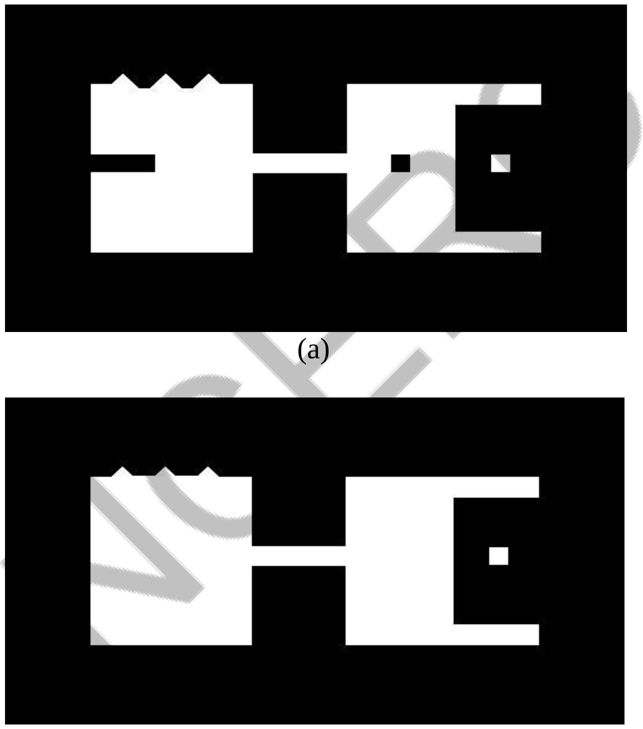
Combining Dilation & Erosion - Closing Morphology

The closing operation dilates an image and then erodes the dilated image using the same structuring element for both operations, i.e.

$$A \bullet B = (A \oplus B) \ominus B$$

where A is the original image and B is the structuring element.

The closing operation fills holes that are smaller than the structuring element, joins narrow breaks, fills gaps in contours, and smoothes objects contours as shown in the figure below.



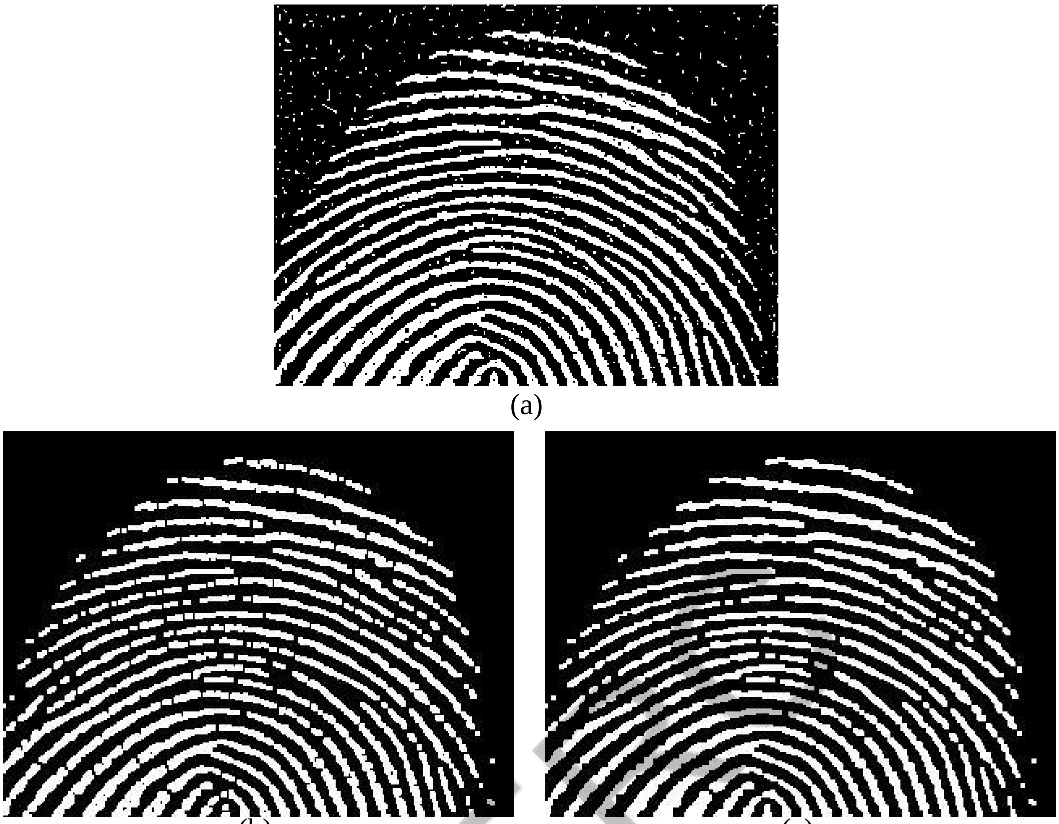
(a)

(b)

Figure 11.7 (a) Result of closing with square structuring element of size 10 pixels. (c) Result of closing with square structuring element of size 20 pixels.

Combining Opening & Closing Morphology

Combining opening and closing can be quite effective in removing noise as illustrated in the next figure.



(a)

(b) (c)

Figure 11.8 (a) Noisy fingerprint. (b) Result of opening (a) with square structuring element of size 3 pixels. (c) Result of closing (b) with the same structuring element.

Note that the noise was removed by opening the image, but this process introduced numerous gaps in the ridges of the fingerprint. These gaps can be filled by following the opening with a closing operation.

Content Beyond syllabus

Convolutional neural networks: an overview

Key Points

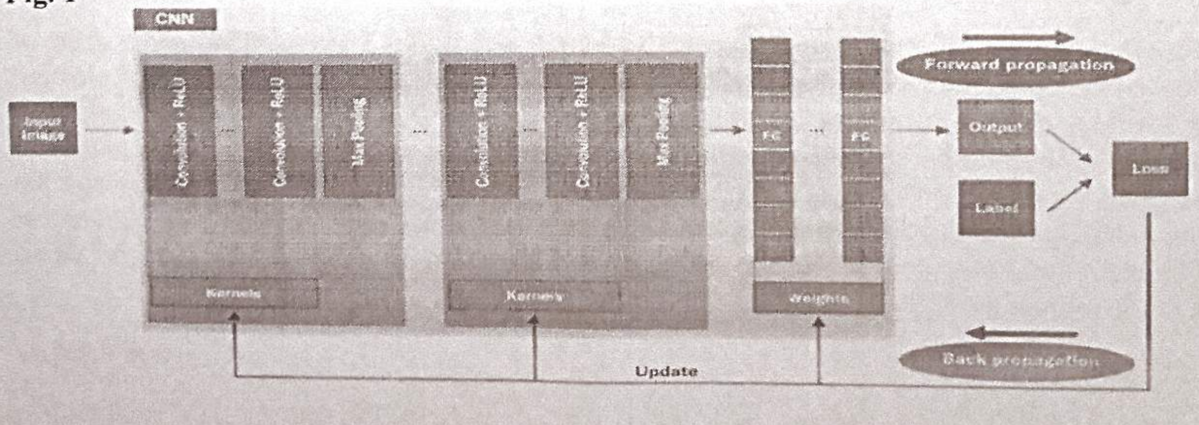
- *Convolutional neural network is a class of deep learning methods which has become dominant in various computer vision tasks and is attracting interest across a variety of domains, including radiology.*

- *Convolutional neural network is composed of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers, and is designed to automatically and adaptively learn spatial hierarchies of features through a backpropagation algorithm.*

What is CNN

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex [13, 14] and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers (Fig. 2), and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

Fig. 1



An overview of a convolutional neural network (CNN) architecture and the training process. A CNN is composed of a stacking of several building blocks: convolution layers, pooling layers (e.g., max pooling), and fully connected (FC) layers. A model's performance under particular kernels and weights is calculated with a loss function through forward propagation on a training dataset, and learnable parameters, i.e., kernels and weights, are updated according to the loss value through backpropagation with gradient descent optimization algorithm. ReLU, rectified linear unit

Building blocks of CNN architecture

The CNN architecture includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A typical architecture consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers. The step where input data are transformed into output through these layers is called forward propagation (Fig. 1). Although convolution and pooling operations described in this section are for 2D-CNN, similar operations can also be performed for three-dimensional (3D)-CNN.

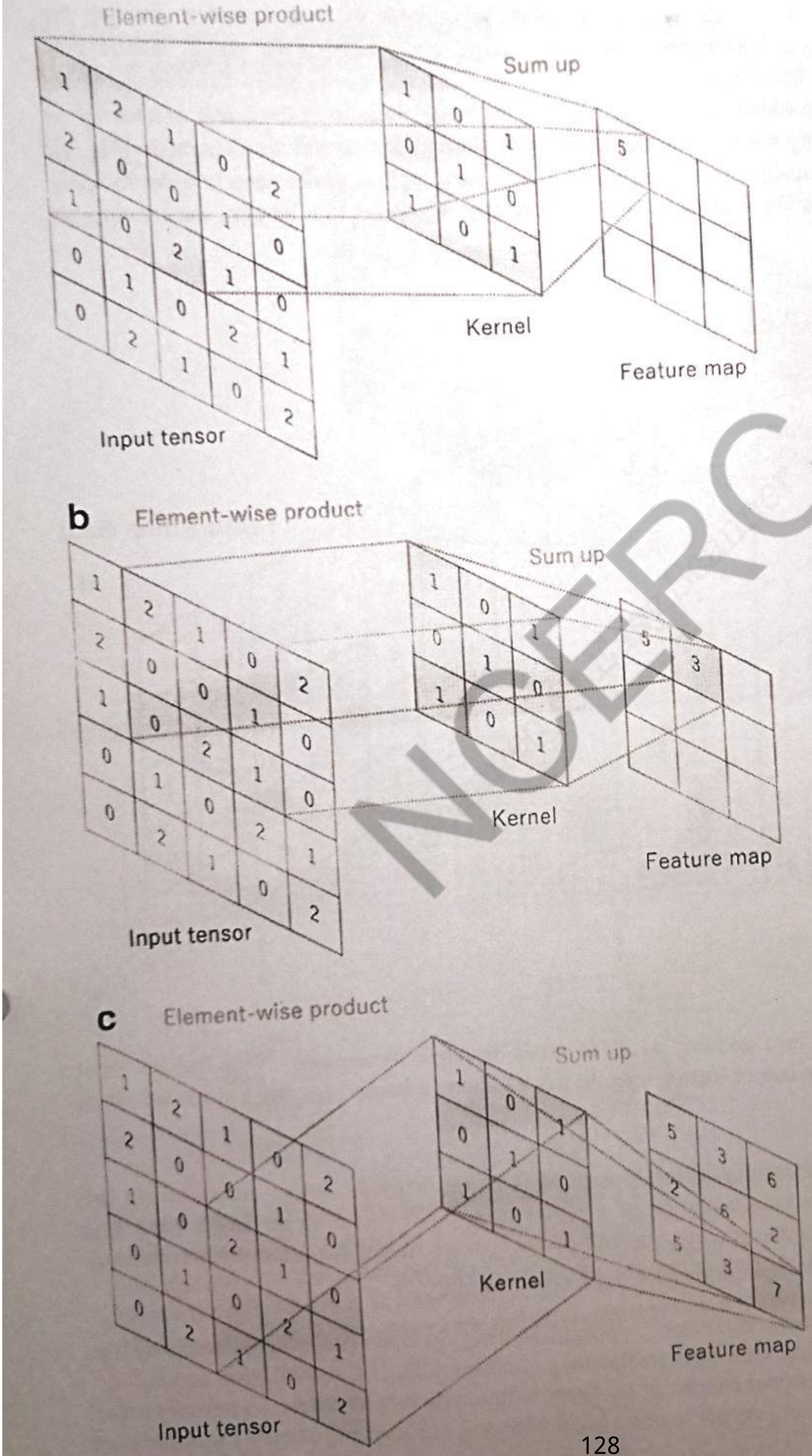
Convolution layer

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, i.e., convolution operation and activation function.

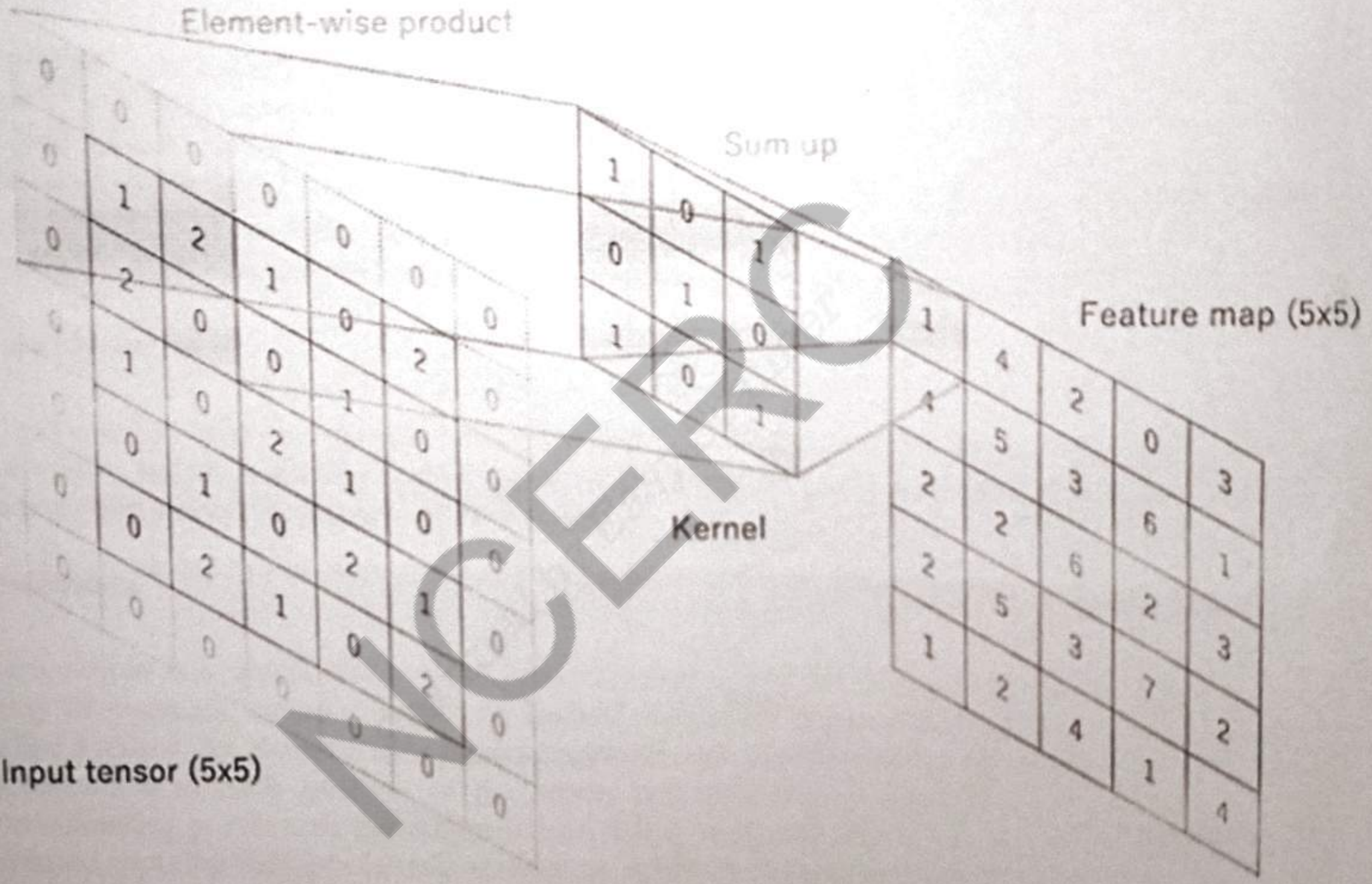
Convolution

Convolution is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map (Fig. 3a–c). This procedure is repeated applying multiple kernels to form an arbitrary number of feature maps, which represent different characteristics of the input tensors; different kernels can, thus, be considered as different feature extractors (Fig. 3d). Two key hyperparameters that define the convolution operation are size and number of kernels. The former is typically 3 × 3, but sometimes 5 × 5 or 7 × 7. The latter is arbitrary, and determines the depth of output feature maps.

a–c An example of convolution operation with a kernel size of 3 × 3, no padding, and a stride of 1. A kernel is applied across the input tensor, and an element-wise product between each element of the kernel and the input tensor is calculated at each location and summed to obtain the output value in the corresponding position of the output tensor, called a feature map. **d** Examples of how kernels in convolution layers extract features from an input tensor are shown. Multiple kernels work as different feature extractors, such as a horizontal edge detector (top), a vertical edge detector (middle), and an outline detector (bottom). Note that the left image is an input, those in the middle are kernels, and those in the right are output feature maps


Element-wise product
Sum up
Kernel
Feature map
Input tensor
b
Element-wise product
Sum up
Kernel
Feature map
Input tensor
c
Element-wise product
Sum up
Kernel
Feature map
Input tensor

he convolution operation described above does not allow the center of each kernel to overlap the outermost element of the input tensor, and reduces the height and width of the output feature map compared to the input tensor. Padding, typically zero padding, is a technique to address this issue, where rows and columns of zeros are added on each side of the input tensor, so as to fit the center of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation (Fig. 4). Modern CNN architectures usually employ zero padding to retain in-plane dimensions in order to apply more layers. Without zero padding, each successive feature map would get smaller after the convolution operation.



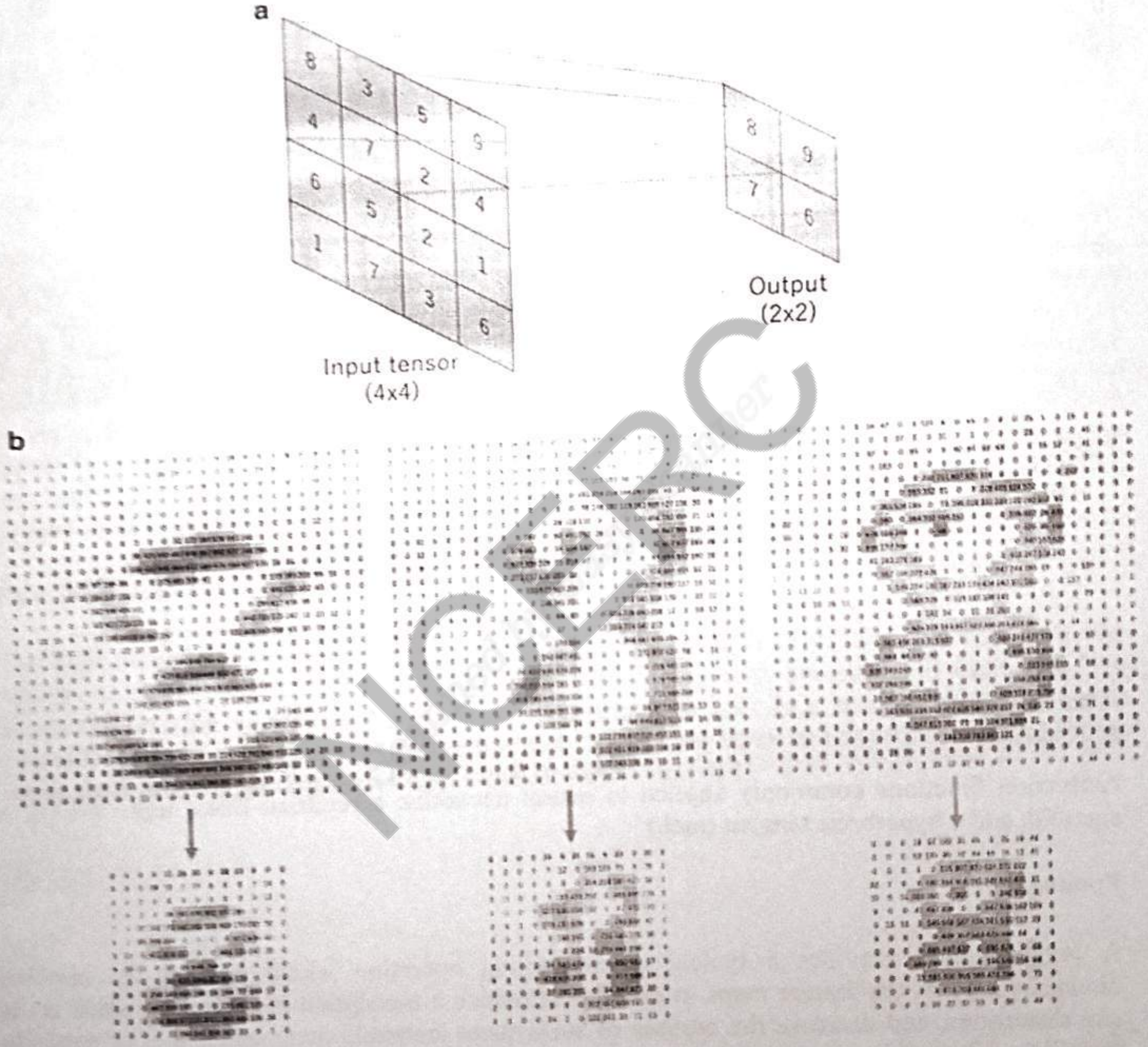
A convolution operation with zero padding so as to retain in-plane dimensions. Note that an input dimension of 5 × 5 is kept in the output feature map. In this example, a kernel size and a stride are set as 3 × 3 and 1, respectively

The distance between two successive kernel positions is called a stride, which also defines the convolution operation. The common choice of a stride is 1; however, a stride larger than 1 is sometimes used in order to achieve downsampling of the feature maps. An alternative technique to perform downsampling is a pooling operation, as described below.

The key feature of a convolution operation is weight sharing: kernels are shared across all the image positions. Weight sharing creates the following characteristics of convolution operations: (1) letting the local feature patterns extracted by kernels translation b invariant as kernels travel across all the image positions and detect learned local patterns, (2) learning spatial hierarchies of

This downsamples the in-plane dimension of feature maps by a factor of 2. Unlike height and width, the depth dimension of feature maps remains unchanged.

Fig. 6



8	3	5	9
4	7	2	4
6	5	2	1
1	7	3	6

8	9
7	6

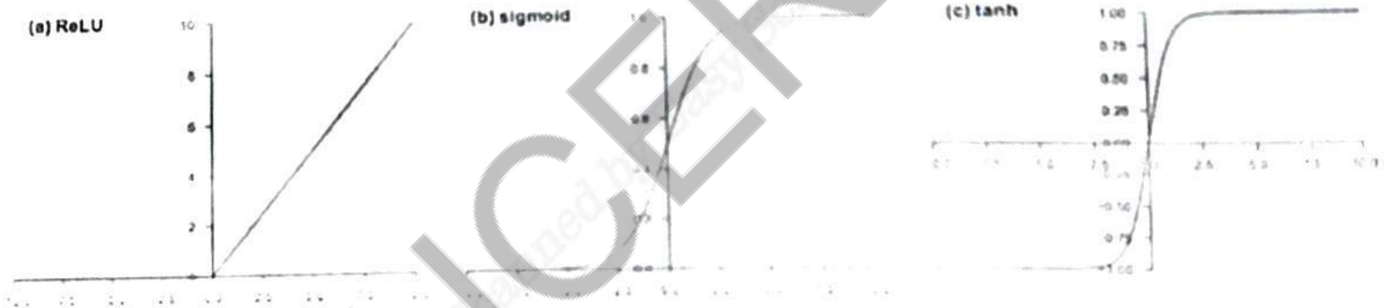
a An example of max pooling operation with a filter size of 2 × 2, no padding, and a stride of 2, which extracts 2 × 2 patches from the input tensors, outputs the maximum value in each patch, and discards all the other values, resulting in downsampling the in-plane dimension of an input tensor by a factor of 2. **b** Examples of the max pooling operation on the same images in Fig. 3b. Note that images in the upper row are downsampled by a factor of 2, from 26 × 26 to 13 × 13 feature patterns by down-sampling in conjunction with a pooling operation, resulting in capturing an increasingly larger field of view, and (3) increasing model efficiency by reducing the number of parameters to learn in comparison with fully connected neural networks.

As described later, the process of training a CNN model with regard to the convolution layer is to identify the kernels that work best for a given task based on a given training dataset. Kernels are the only parameters automatically learned during the training process in the convolution layer; on the other hand, the size of the kernels, number of kernels, padding, and stride are hyperparameters that need to be set before the training process starts .

Nonlinear activation function

The outputs of a linear operation such as convolution are then passed through a nonlinear activation function. Although smooth nonlinear functions, such as sigmoid or hyperbolic tangent (tanh) function, were used previously because they are mathematical representations of a biological neuron behavior, the most common nonlinear activation function used presently is the rectified linear unit (ReLU), which simply computes the function: $f(x) = \max(0, x)$ (Fig. 5) [1, 3, 17,18,19].

Fig. 5



Activation functions commonly applied to neural networks: **a** rectified linear unit (ReLU), **b** sigmoid, and **c** hyperbolic tangent (tanh)

Pooling layer

A pooling layer provides a typical downsampling operation which reduces the in-plane dimensionality of the feature maps in order to introduce a translation invariance to small shifts and distortions, and decrease the number of subsequent learnable parameters. It is of note that there is no learnable parameter in any of the pooling layers, whereas filter size, stride, and padding are hyperparameters in pooling operations, similar to convolution operations.

Max pooling

The most popular form of pooling operation is max pooling, which extracts patches from the input feature maps, outputs the maximum value in each patch, and discards all the other values (Fig. 6). A max pooling with a filter of size 2 × 2 with a stride of 2 is commonly used in practice

Global average pooling

Another pooling operation worth noting is a global average pooling [20]. A global average pooling performs an extreme type of downsampling, where a feature map with size of height × width is downsampled into a 1 × 1 array by simply taking the average of all the elements in each feature map, whereas the depth of feature maps is retained. This operation is typically applied only once before the fully connected layers. The advantages of applying global average pooling are as follows: (1) reduces the number of learnable parameters and (2) enables the CNN to accept inputs of variable size.

Fully connected layer

The output feature maps of the final convolution or pooling layer is typically flattened, i.e., transformed into a one-dimensional (1D) array of numbers (or vector), and connected to one or more fully connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight. Once the features extracted by the convolution layers and downsampled by the pooling layers are created, they are mapped by a subset of fully connected layers to the final outputs of the network, such as the probabilities for each class in classification tasks. The final fully connected layer typically has the same number of output nodes as the number of classes. Each fully connected layer is followed by a nonlinear function, such as ReLU, as described above.

Last layer activation function

The activation function applied to the last fully connected layer is usually different from the others. An appropriate activation function needs to be selected according to each task. An activation function applied to the multiclass classification task is a softmax function which normalizes output real values from the last fully connected layer to target class probabilities, where each value ranges between 0 and 1 and all values sum to 1.