

Digital Image Processing

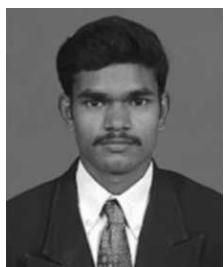
About the Authors



S Jayaraman obtained his BE and ME from PSG College of Technology in the years 1974 and 1976 respectively, and his PhD from Bharathiar University in 1993. He has more than thirty years of experience in teaching and research and has been associated with many sponsored projects funded by AICTE, DST, DMIT and SDC. He has guided three PhD research works and is currently supervising three scholars in their PhD studies. He has published more than 80 papers in various national and international journals and conferences, and has delivered many special lectures in short-term training programmes and faculty development programmes sponsored by the AICTE and ISTE. His specific areas of interest include Bio-Medical Signal Processing, Multidimensional System Analysis, Image Compression and Non-Linear Signal Processing.



S Esakkirajan completed his BSc in Physics from Sadakathullah Appa College, Palayamkottai, BTech from Cochin University of Science and Technology, Cochin, and ME from PSG College of Technology, Coimbatore, also being an ME rank holder. He has received the Alumni Award in his ME degree programme. He has published many papers in international and national journals and his areas of interest include Database Management Systems, and Digital Image Compression.



T Veerakumar received his BE in Electronics and Communication Engineering from RVS College of Engineering, and did his ME in Applied Electronics from PSG College of Technology. He has presented papers in international and national journals and his areas of interest include Artificial Neural Networks, Image Compression and Video Compression.

Digital Image Processing

S Jayaraman

Retired Professor

*Department of Electronics and Communication Engineering
PSG College of Technology
Coimbatore, Tamil Nadu*

S Esakkirajan

Lecturer

*Department of Instrumentation and Control Engineering
PSG College of Technology
Coimbatore, Tamil Nadu*

T Veerakumar

Lecturer

*Department of Electronics and Communication Engineering
PSG College of Technology
Coimbatore, Tamil Nadu*



Tata McGraw Hill Education Private Limited
NEW DELHI

McGraw-Hill Offices

New Delhi New York St. Louis San Francisco Auckland Bogotá
Caracas Kuala Lumpur Lisbon London Madrid Mexico City
Milan Montreal San Juan Santiago Singapore Sydney Tokyo Toronto



Tata McGraw Hill

Published by the Tata McGraw Hill Education Private Limited,
7 West Patel Nagar, New Delhi 110 008.

Copyright © 2009 by Tata McGraw Hill Education Private Limited.

No part of this publication may be reproduced or distributed in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a database or retrieval system without the prior written permission of the publishers. The program listings (if any) may be entered, stored and executed in a computer system, but they may not be reproduced for publication.

This edition can be exported from India only by the publishers,
Tata McGraw Hill Education Private Limited

ISBN (13): 978-0-07-014479-8

ISBN (10): 0-07-014479-6

Managing Director: *Ajay Shukla*

General Manager: Publishing—SEM & Tech Ed: *Vibha Mahajan*

Manager—Sponsoring: *Shalini Jha*

Asst. Sponsoring Editor: *Suman Sen*

Executive—Editorial Services: *Sohini Mukherjee*

Senior Production Manager: *P L Pandita*

General Manager: Marketing—Higher Education & School: *Michael J Cruz*

Sr. Product Manager: SEM & Tech Ed: *Biju Ganesan*

General Manager—Production: *Rajender P Ghansela*

Asst. General Manager—Production: *B L Dogra*

Information contained in this work has been obtained by Tata McGraw Hill, from sources believed to be reliable. However, neither Tata McGraw Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither Tata McGraw Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that Tata McGraw Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Typeset at Text-o-Graphics, B-1/56, Aravali Apartment, Sector-34, Noida 201 301 and printed at
Avon Printers, Plot No. 16, Main Loni Road, Jawahar Nagar Industrial Area, Shahdara, Delhi 110 094

Cover: SDR

RAZLCRAFDQDL

The McGraw.Hill Companies

Contents

<i>Preface</i>	<i>x</i>
<i>Acknowledgements</i>	<i>xiii</i>
<i>List of Acronyms</i>	<i>xvi</i>
<i>Image-Processing Taxonomy</i>	<i>xviii</i>
<i>Visual Walkthrough</i>	<i>xix</i>
1. Introduction to Image-processing System	1
1.1 Introduction	1
1.2 Image Sampling	4
1.3 Quantisation	12
1.4 Resolution	13
1.5 Human Visual System	14
1.6 Classification of Digital Images	19
1.7 Image Types	20
1.8 Elements of an Image-processing System	22
1.9 Image File Formats	38
1.10 Applications of Digital Image Processing	42
<i>Summary</i>	43
<i>Review Questions</i>	43
<i>Problems</i>	45
<i>References</i>	46
2. 2D Signals and Systems	47
2.1 Introduction	47
2.2 2D Signals	48
2.3 Separable Sequence	50
2.4 Periodic Sequence	50
2.5 2D Systems	51
2.6 Classification of 2D Systems	51
2.7 2D Convolution	55
2.8 2D Z-Transform	55
2.9 2D Digital Filter	71
<i>Summary</i>	77
<i>Review Questions</i>	77
<i>Problems</i>	81
<i>References</i>	82
3. Convolution and Correlation	84
3.1 Introduction	84
3.2 2D Convolution Through Graphical Method	85
3.3 Convolution Through Z-Transform	106
3.4 2D Convolution Through Matrix Analysis	110
3.5 Circular Convolution Through Matrix Method	122
3.6 Application of Circular Convolution	124
3.7 2D Correlation	127
<i>Summary</i>	148
<i>Review Questions</i>	149
<i>Problems</i>	150
<i>References</i>	151

4. Image Transforms	152
4.1 Introduction	152
4.2 Need for Transform	153
4.3 Image Transforms	153
4.4 Fourier Transform	155
4.5 2D Discrete Fourier Transform	158
4.6 Properties of 2D-DFT	159
4.7 Importance of Phase	174
4.8 Walsh Transform	175
4.9 Hadamard Transform	181
4.10 Haar Transform	182
4.11 Slant Transform	193
4.12 Discrete Cosine Transform	194
4.13 Karhunen-Loeve Transform (KL Transform)	202
4.14 Singular Value Decomposition	209
4.15 Radon Transform	222
4.16 Comparison of Different Image Transforms	229
<i>Solved Problems</i>	230
<i>Summary</i>	238
<i>Review Questions</i>	239
<i>Problems</i>	240
<i>References</i>	242
5. Image Enhancement	243
5.1 Introduction	243
5.2 Image Enhancement in Spatial Domain	244
5.3 Enhancement Through Point Operation	245
5.4 Types of Point Operation	245
5.5 Histogram Manipulation	248
5.6 Linear Gray-level Transformation	252
5.7 Nonlinear Gray-level Transformation	253
5.8 Local or Neighbourhood Operation	260
5.9 Median Filter	270
5.10 Spatial Domain High-pass Filtering or Image Sharpening	273
5.11 Bit-plane Slicing	275
5.12 Image Enhancement in the Frequency Domain	278
5.13 Homomorphic Filter	292
5.14 Zooming Operation	293
5.15 Image Arithmetic	297
<i>Solved Problems</i>	304
<i>Summary</i>	316
<i>Review Questions</i>	317
<i>Problems</i>	320
<i>Matlab Exercises</i>	322
<i>References</i>	323
6. Image Restoration and Denoising	324
6.1 Introduction	324
6.2 Image Degradation	325
6.3 Types of Image Blur	325
6.4 Classification of Image-restoration Techniques	327
6.5 Image-restoration Model	328
6.6 Linear Image-restoration Techniques	329
6.7 Non-linear Image-restoration Techniques	340

6.8	Blind Deconvolution	344
6.9	Classification of Blind-deconvolution Techniques	344
6.10	Image Denoising	348
6.11	Classification of Noise in Image	348
6.12	Median Filtering	349
6.13	Trimmed Average Filter	354
6.14	Performance Metrics in Image Restoration	357
6.15	Applications of Digital Image Restoration	357
	<i>Solved Problems</i>	358
	<i>Summary</i>	362
	<i>Review Questions</i>	363
	<i>Problems</i>	365
	<i>References</i>	367
7.	Image Segmentation	368
7.1	Introduction	368
7.2	Classification of Image-segmentation Techniques	369
7.3	Region Approach to Image Segmentation	369
7.4	Clustering Techniques	376
7.5	Image Segmentation Based on Thresholding	379
7.6	Edge-based Segmentation	380
7.7	Classification of Edges	380
7.8	Edge Detection	381
7.9	Edge Linking	391
7.10	Hough Transform	392
7.11	Active Contour	393
7.12	Watershed Transformation	394
7.13	Shape Representation	397
7.14	Classification of Shape-representation Techniques	397
	<i>Summary</i>	400
	<i>Review Questions</i>	401
	<i>Problems</i>	405
	<i>References</i>	407
8.	Object Recognition	408
8.1	Introduction	409
8.2	Need for an Object-recognition System	409
8.3	Automated Object-recognition Systems	409
8.4	Patterns and Pattern Class	411
8.5	Selection of Measurement Parameters	414
8.6	Relationship between Image Processing and Object Recognition	415
8.7	Approaches to Object Recognition	415
8.8	Bayes' Parametric Classification	416
8.9	Template-Matching-based Object Recognition	418
8.10	Non-parametric Density Estimation	418
8.11	Neural-network Approach to Object Recognition	419
8.12	Classification of Artificial Neural Networks	423
8.13	Learning Rules in Artificial Neural Networks	425
8.14	Perceptron	426
8.15	Multi-layer Perceptron	427
8.16	Radial-basis Function Network	431
8.17	Advantages of Neural Networks	431
8.18	Structural Pattern Recognition	432
8.19	Applications of Object Recognition	434

<i>Summary</i>	437
<i>Review Questions</i>	438
<i>References</i>	443
9. Image Compression	444
9.1 Introduction	444
9.2 Need for Image Compression	445
9.3 Redundancy in Images	445
9.4 Classification of Redundancy in Images	445
9.5 Image-compression Scheme	446
9.6 Classification of Image-compression Schemes	447
9.7 Fundamentals of Information Theory	447
9.8 Run-length Coding	449
9.9 Shannon-Fano Coding	450
9.10 Huffman Coding	452
9.11 Arithmetic Coding	457
9.12 Dictionary-based Compression	469
9.13 Predictive Coding	469
9.14 Transform-based Compression	487
9.15 Image-compression Standard	488
9.16 Scalar Quantisation	494
9.17 Vector Quantisation	497
9.18 Types of Vector Quantisation	505
9.19 Wavelet-based Image Compression	507
9.20 Fractal Image Compression	507
9.21 Block Truncation Coding	511
<i>Solved Problems</i>	515
<i>Summary</i>	536
<i>Review Questions</i>	537
<i>Problems</i>	538
<i>References</i>	541
10. Binary Image Processing	543
10.1 Introduction	543
10.2 Binarisation	544
10.3 Mathematical Morphology	544
10.4 Structuring Elements	544
10.5 Morphological Image Processing	544
10.6 Basic Set Theory	546
10.7 Logical Operations	547
10.8 Standard Binary Morphological Operations	548
10.9 Dilation- and Erosion-based Operations	554
10.10 Properties of Morphological Operation	555
10.11 Morphological Algorithms	564
10.12 Distance Transform	576
10.13 Salient Features of Morphological Approach	578
<i>Summary</i>	578
<i>Review Questions</i>	579
<i>Problems</i>	580
<i>References</i>	583
11. Colour-Image Processing	584
11.1 Introduction	585
11.2 Light and Colour	585
11.3 Colour Formation	585
11.4 Human Perception of Colour	586

11.5 Colour Model	586
11.6 The Chromaticity Diagram	590
11.7 Colour-image Quantisation	593
11.8 Histogram of a Colour Image	594
11.9 Colour-image Filtering	597
11.10 Gamma Correction of a Colour Image	600
11.11 Pseudo-colour	604
11.12 Colour-image Segmentation	605
<i>Summary</i>	606
<i>Review Questions</i>	607
<i>Problems</i>	608
<i>References</i>	610

12. Wavelet-based Image Processing 611

12.1 Introduction	612
12.2 Evolution of Wavelet Transform	612
12.3 Wavelet	613
12.4 Wavelet Transform	614
12.5 Continuous Wavelet Transform	615
12.6 2D Continuous Wavelet Transform	615
12.7 Multi-resolution Analysis	616
12.8 Examples of Wavelets	617
12.9 Wavelet-based Image Compression	619
12.10 Embedded Image Coding	626
12.11 Embedded Zero Tree Wavelet	626
12.12 Set Partitioning in Hierarchical Trees (SPIHT)	634
12.13 JPEG2000 Compression Standard	639
12.14 Desirable Properties of Wavelets	644
12.15 Wavelet Packet Transform	645
12.16 Multi-wavelets	646
12.17 Contourlet Transform	648
12.18 Image Pyramid	650
12.19 Wavelet-based Denoising	655
12.20 Wavelet-thresholding Methods	656
12.21 Comparison of Different Thresholding Methods	659
12.22 Digital Image Watermarking	659
12.23 Classification of Watermarking Methods	660
12.24 Watermarking in the Spatial Domain	661
12.25 Watermarking in Frequency Domain	664
12.26 Applications of Digital Watermarking	667
<i>Summary</i>	667
<i>Review Questions</i>	668
<i>Problems</i>	670
<i>References</i>	671

13. An Introduction to Video Processing *Available on the website*

Appendices

Appendix I: Image Processing Related MATLAB Commands	673
Appendix II: Overview of Vector Space Concepts	680
Appendix III: Fundamentals of Matrices	684
Appendix IV: Objective Type Questions	695

Glossary

710

Index

719

Preface

Vision is the most powerful of the five human senses. Visual information, conveyed in the form of images gives better impact than textual information. The fields of digital image processing have grown tremendously over the past 30 years. The growth of digital image processing has been fueled by technological advances in digital imaging, computer processors and mass storage devices. Research and development of image processing technologies have advanced very rapidly in the past decade. Digital image processing is concerned primarily with the extraction of useful information from images. In this process, it also deals with 1) image representation, 2) enhancing the quality of an image, 3) restoration of the original image from its degraded version, and 4) compression of the large amount of data in the images for efficient archiving and retrieval. Image-processing algorithms can be classified into three different categories. At the lowest level, the algorithm deals directly with the raw pixel values (image denoising and edge detection are good examples). In the middle level, the algorithm utilises low-level results for processes such as segmentation and edge linking. At the highest level, the algorithm attempts to extract semantic information from those provided by the lower levels. Examples of high-level algorithms are handwriting recognition, face recognition and machine vision algorithms. The objective of this book is to not only introduce different concepts of digital image processing to undergraduate and postgraduate students but also to serve as a handbook for practicing engineers.

Simulation is an essential tool in any field related to engineering techniques. In this book, the image-processing algorithms are simulated using MATLAB. It has been the endeavour of the authors to present a large number of detailed worked examples to illustrate the various digital image-processing concepts.

Organisation of the Book

This book contains twelve chapters. **Chapter 1** gives an overview of the digital image-processing system. The different elements of image-processing systems like image acquisition, image sampling, quantisation, image sensors, image scanners and image storage devices are covered in this chapter. The highlight of this chapter is the discussion of the different types of image file formats in practice.

Chapter 2 deals with two-dimensional signals and systems. Different types of two-dimensional signals, and properties of two-dimensional signals like separability and periodicity are discussed in this chapter. This chapter also gives an overview of the two-dimensional system. The focus is mainly on the linear shift-invariant system. As Z-transform is widely used to analyse two dimensional signals and systems, the understanding of the 2D signals and systems is enriched through numerous examples of forward and inverse two-dimensional Z transforms in this chapter.

Chapter 3 is devoted to 2D convolution and correlation. Convolution is one of the most powerful mathematical operators which is widely used in the field of digital image processing. In this chapter, the computation of 2D convolution and correlation using graphical, matrix and Z-transform methods are discussed in a step-by-step approach. The examples related to 2D convolution and correlations are illustrated through MATLAB examples.

The focus in **Chapter 4** is on image transforms. The need for image transforms, different types of image transforms and their properties are explained in this chapter. The image transforms discussed in this chapter include two-dimensional Fourier transform, Walsh transform, Hadamard transform, Slant transform, KL transform, Discrete Cosine Transform, Radon transform and Singular Value Decomposition.

Chapter 5 discusses different techniques employed for the enhancement of images. This chapter includes techniques in spatial as well as frequency domains. While the frequency domain methods discuss the design of low-pass, high-pass and band-pass filters, the discussion of techniques related to the spatial domain include different gray-level transformations together with spatial filtering methods such as low-pass, high-pass and high boost filtering which are illustrated with MATLAB examples. Further, in this chapter, certain test images like flowers, animals and popular monuments are considered rather than the usual images, like Cameraman and Lena, to bring out variety and emphasis on generality in approach.

Chapter 6 provides an overview of image restoration and image denoising techniques. The different causes for image degradation, and deterministic and stochastic methods of image restoration are the crucial topics that are discussed in this chapter. Also, this chapter covers the different types of image denoising techniques like average filtering, median filtering, alpha-trimmed filtering and min-max filtering.

Chapter 7 deals with different techniques in image segmentation. The different techniques include region, boundary and edge-based segmentation methods. Advanced image segmentation algorithms like the Snake algorithm and Watershed algorithm are covered in this chapter.

Chapter 8 gives an overview of different types of object-recognition techniques. Different types of representation of patterns are discussed in this chapter, namely Statistical approach, Structural approach and Neural-Network approach. Different applications of object recognition are also discussed in this chapter.

Chapter 9 is devoted to image compression. The need for image compression, spatial domain and frequency domain techniques of image compression and the applications of image compression form the heart of this chapter. One of the powerful tools to achieve compression is vector quantisation. This chapter gives the basic idea of vector quantisation (VQ) and different approaches to VQ through illustrative examples.

Chapter 10 deals with binary image processing. In this chapter, different morphological operations like dilation, erosion, opening and closing are given with suitable MATLAB examples along with their properties. Other topics discussed in this chapter include thinning, thickening, distance transform and hit-or-miss transform.

The concepts related to colour image processing are given in **Chapter 11**. The concepts discussed in this chapter include human perception of colour, colour models, colour image quantisation, colour image filtering, pseudo-colouring, colour image histogram and colour image segmentation.

Finally, in **Chapter 12**, the application of wavelets in image processing are dealt with. The different wavelet-based applications considered in this chapter are image compression, image denoising and watermarking. The embedded quantisation techniques widely used in the field of image compression like EZW and SPIHT are explained through numerical examples. An introduction to advanced concepts like multi-wavelet transforms, contourlet transforms and directional filter banks are also given in this chapter to provide some sense of completeness and stimulate the researchers' interest.

Each chapter is equipped with a set of solved problems with solutions, and review questions with answers. The problem sets help readers to understand the basic concepts in digital image processing.

In addition to the twelve chapters, four appendices are also given in this book. Since MATLAB has been used extensively to highlight the various concepts throughout this book, a list of commonly used MATLAB commands in image processing are provided in **Appendix I**.

From our past teaching experience, it is the considered opinion of the authors that a holistic picture of Image Transforms can be brought about by providing a unified treatment from the standpoint of vector spaces. An introductory knowledge related to Vector Space, thus, becomes essential to the understanding of image transforms. An introduction to Vector Spaces is provided in **Appendix II**.

A digital image can be represented in terms of a matrix. Hence, a basic knowledge of matrices will be useful for effective manipulation of images. Fundamental concepts related to different types of matrices are given in *Appendix III*.

Objective-type questions measure one's ability to remember facts and figures and comprehend the concepts related to any subject. In this book, a rich set of objective-type questions along with answers are separately provided in *Appendix IV*.

Apart from this, the book also has additional web supplements and an accompanying CD. The online contents can be accessed at <http://www.mhhe.com/jayaraman/dip> and contain the following material:

- **An Introduction to Video Processing**
 - Introduction to video processing
 - Spatio-temporal sampling
 - Interframe and Intraframe coding
 - Motion Estimation Techniques
 - Video Compression Standards
- **Interactive Quiz**
- **Downloadable images from the text**
- **Solution Manual for Instructors**
- **The accompanying CD contains**
 - Powerpoint tutorials
 - Images from the text
 - MATLAB codes

We earnestly hope that this book will initiate many persons to the exciting world of digital image processing. Though we have spared no pains to make this book free from mistakes, some errors may still have survived our scrutiny. We gladly welcome all corrections, recommendations, suggestions and constructive criticism from our readers.

S Jayaraman
S Esakkirajan
T Veerakumar

Acknowledgements

The authors are always thankful to the Almighty for guiding them in their perseverance and blessing them with achievements.

The authors wish to thank Mr Rangaswamy, Managing Trustee, PSG Institutions; Mr C R Swaminathan, Chief Executive; and Dr R Rudramoorthy, Principal, PSG College of Technology, Coimbatore, for their wholehearted cooperation and constant encouragement given in this successful endeavour.

Dr S Jayaraman would like to thank his parents, Sri P S Subramanyam and Mrs J Chellam, for inculcating values and providing inspiring guidance; wife, Mrs V Ganga; and daughter, J Pavitra, for their constant encouragement and moral support along with patience and understanding. Sincere thanks is also due to his teacher, friend, philosopher and guide, Dr S N Sivanandan, Professor and Head, Department of CSE, PSG College of Technology, for being a source of inspiration in all his endeavours. His enriching wisdom, experience and timely advice have been extremely influential in shaping the author's outlook on life.

Mr S Esakkirajan would like to thank his father, Mr G Sankaralingam, and wife, Mrs Akila, who shouldered a lot of extra responsibilities during the months this book was being written. He also likes to thank his teachers Dr N Malmurugan and Dr R Sudhakar, who taught him the basics of digital image processing.

Mr T Veerakumar would like to thank his parents, Mr N Thangaraj and Mrs T Muniammal, for their support.

The authors wholeheartedly thank, appreciate and sincerely acknowledge Mr V Senthil Murugan, Mr Paalraj, Mr Afsar Ahmed, Ms Kavitha and Ms Padmapriya for their excellent, unforgettable help and assistance towards the documentation related to this book. The authors wish to thank Mr D Sivaraj, Mr K Joseph Mathew and Mr A R Ramakrishnan for all their support.

The authors would also like to thank the following reviewers for taking out time to review the book.

B Lokeswara Rao *Geethanjali College of Engineering and Technology
Hyderabad, Andhra Pradesh*

G N Srinivasan *R V College of Engineering
Bangalore, Karnataka*

E Bengeorge *QIS College of Engineering
Ongole, Andhra Pradesh*

K Krishna Prasad *National Institute of Technology
Warangal, Andhra Pradesh*

M V Raghunadh *National Institute of Technology
Warangal, Andhra Pradesh*

Tessamma Thomas *Cochin University of Science and Technology
Cochin, Kerala*

Y Wiselin Jiji *Dr Sivanthi Adithanar College of Engineering
Tiruchendur, Tamil Nadu*

D Laxmi *Bannari Amman Institute of Technology
Sathyamangalam, Tamil Nadu*

A N Rajagopalan	<i>Indian Institute of Technology Madras Chennai, Tamil Nadu</i>
S Allin Christy	<i>PSG College of Technology Coimbatore, Tamil Nadu</i>
A K Sachan	<i>Truba College of Research and Technology Bhopal, Madhya Pradesh</i>
Manish Kumar Verma	<i>Bharat Institute of Technology Meerut, Uttar Pradesh</i>
Udai Shanker	<i>MMMEC University Gorakhpur, Uttar Pradesh</i>
S Bhattacharya	<i>Indian School of Mines Dhanbad, Jharkhand</i>
S M Patil	<i>Bharati Vidyapeeth College of Engineering Navi Mumbai, Maharashtra</i>
Tanish Zaveri	<i>Nirma Institute of Technology Ahmedabad, Gujarat</i>
Jignasa P Mehta	<i>VVP Institute of Technology Rajkot, Gujarat</i>
J N Sarvaiya	<i>SVNIT Surat, Gujarat</i>

US Reviews

Dapeng Wu	<i>University of Florida Gainesville, Florida</i>
Frank Shih	<i>Department of Computer Science New Jersey Institute of Technology Newark, New Jersey</i>
Amar Raheja	<i>Department of Computer Engineering California State Polytechnic University Pomona, California</i>
Bahadir K Gunturk	<i>Department of Electrical and Computer Engineering, Louisiana State University Baton Rouge, Los Angels</i>
Artyom M Grigoryan	<i>Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, Texas</i>
Xin Li	<i>34 Antietam Drive, Morgantown West Virginia</i>

Marketing Reviews

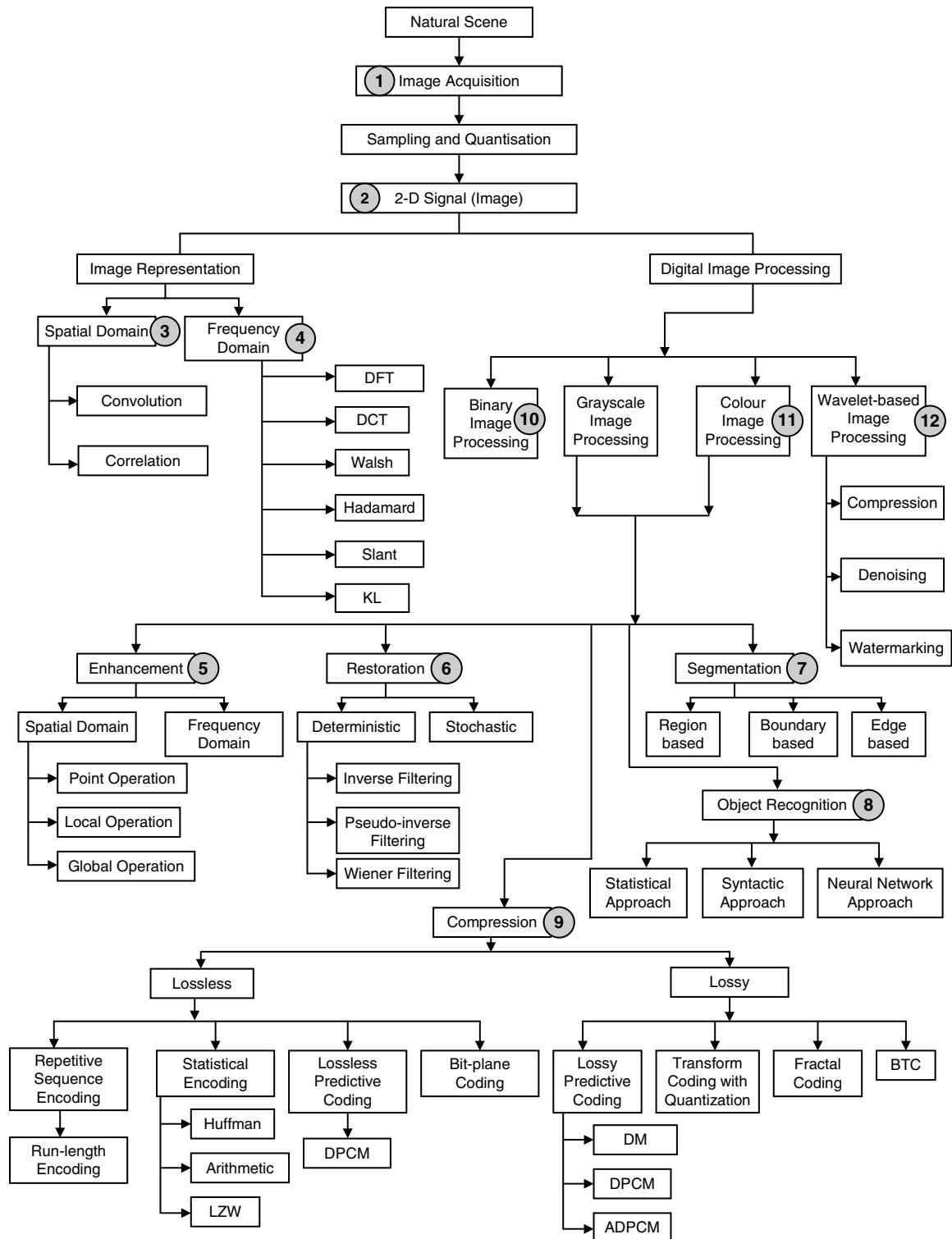
G Sudha Vani	<i>RVR and JC College of Engineering Guntur, Andhra Pradesh</i>
Kilari Veeraswamy	<i>QIS College of Engineering and Technology Bapatla, Andhra Pradesh</i>
S Selvan	<i>St Peter's Engineering College Chennai, Tamil Nadu</i>
Mary Joans	<i>Vellamal Engineering College Chennai, Tamil Nadu</i>
Sudhanshu Tripathi	<i>Amity School of Engineering and Technology New Delhi, Delhi</i>
Narendra Kohli	<i>HBTI, Kanpur, Uttar Pradesh</i>
R P Arora	<i>Dehradun Institute of Technology Dehradun, Uttarakhand</i>
Satish Chavan	<i>Don Bosco Institute of Technology Mumbai, Maharashtra</i>
S D Joshi	<i>Vishwakarma Government Engineering College Ahmedabad, Gujarat</i>

LIST OF ACRONYMS

2-D	Two-Dimensional
ADALINE	Adaptive Linear Element
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
AWGN	Additive White Gaussian Noise
BAM	Bidirectional Associative Memory
BIBO	Bounded Input Bounded Output
BPN	Back Propagation Network
BTC	Block Truncation Coding
CCD	Charge Coupled Device
CIE	Commission International d'Eclairage
CLS	Constrained Least Square
CLUT	Color Look-up Table
CMOS	Complementary Metal Oxide Semiconductor
CODEC	Coder Decoder
CPN	Counter Propagation Network
CR	Compression Ratio
CRT	Cathode Ray Tube
CTE	Charge Transfer Efficiency
DCT	Discrete Cosine Transform
DFB	Directional Filter Bank
DFT	Discrete Fourier Transform
DICOM	Digital Imaging and Communications in Medicine
DM	Delta Modulation
DPCM	Differential Pulse Code Modulation
DPI	Dots Per Inch
DWT	Discrete Wavelet Transform
EBCOT	Embedded Block Coding with Optimized Truncation
ECG	Electro Cardiogram
EPS	Encapsulated Post Script
EZW	Embedded Zero Tree Wavelet
FDCT	Forward Discrete Cosine Transform
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FLC	Fixed Length Code
GIF	Graphic Interchange Format
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
IEEE	Institute of Electrical and Electronic Engineers
iid	Independent and identically distributed
IIR	Infinite Impulse Response
IFS	Iterated Function System
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Expert Group
KLT	Karhunen-Loeve Transform

LMS	Least Mean Square
LP	Laplacian Pyramid
LSI	Linear Shift Invariant System
LZW	Lempel-Ziv-Welch
MADALINE	Multiple Adaptive Linear Element
MAE	Mean Absolute Error
MPEG	Moving Pictures Expert Group
MR	Multiresolution
MSE	Mean Square Error
MSVQ	Multi-Stage Vector Quantization
NIST	National Institute of Standards and Technology
NTSC	National Television System Committee
PACS	Picture Archiving and Communication Systems
PCA	Principal Component Analysis
PDFB	Pyramidal Directional Filter Bank
PEL	Picture Element
PIFS	Partitioned Iterated Function System
PNG	Portable Network Graphics
PSF	Point Spread Function
PSNR	Peak Signal-to-Noise Ratio
RBF	Radial Basis Function
R-D	Rate-Distortion
RLC	Run Length Coding
RMSE	Root Mean Square Error
SECAM	Sequential Chrominance Signal and Memory
SNR	Signal-to-Noise Ratio
SOM	Self Organizing Map
SONAR	Sound Navigation and Ranging
SPIHT	Set Partitioning in Hierarchical Trees
SQ	Scalar Quantization
STFT	Short Time Fourier Transform
SURE	Stein's Unbiased Risk Estimator
SVD	Singular Value Decomposition
SVG	Scalable Vector Graphics
TIFF	Tagged Image File Format
TSVQ	Tree Structure Vector Quantization
VLC	Variable Length Coding
VOXEL	Volume Pixel
VQ	Vector Quantization
WHT	Walsh Hadamard Transform
WP	Wavelet Packet
WT	Wavelet Transform
WWW	World Wide Web

IMAGE-PROCESSING TAXONOMY



VISUAL WALKTHROUGH

Each chapter begins with a set of learning objectives. Learning Objectives describe what the reader should be able to do after participating in the learning activity. Learning Objectives give learners a clear picture of what to expect and what is expected of them.

1

Learning Objectives

This chapter provides an overview of the image-processing system which includes various elements like image sampling, quantisation, processing, storage and display. On completion of this chapter, the reader is expected to be familiar with the following concepts:

- Image sampling*
- Image types*
- Image sensors*
- Image storage*
- Image processing*
- Image display*

Introduction to Image-processing System

1.1 INTRODUCTION

Digital images play an important role, both in daily-life applications such as satellite television, magnetic resonance imaging, computer tomography, as well as in areas of research and technology such as geographical information systems and astronomy. An image is a 2D representation of a three-dimensional scene. A digital image is basically a numerical representation of an object. The term *digital image processing* refers to the manipulation of an image by means of a processor. The different elements of an image-processing system include image acquisition, image storage, image processing and display. This chapter begins with the basic definition of a digital image and is followed by a detailed discussion on two-dimensional sampling. This is followed by different elements of image processing systems.

2

Learning Objectives

This chapter deals with 2D signals and systems. The transform which is widely used in system study is the Z-transform. An introduction to 2D Z-transforms is given in this chapter. After reading this chapter, the reader should be familiar with the following concepts:

- 2D signals*
- Periodic, aperiodic signals*
- 2D linear shift invariant systems*
- Properties of 2D systems*
- Forward and inverse 2D Z-transforms*

2D Signals and Systems

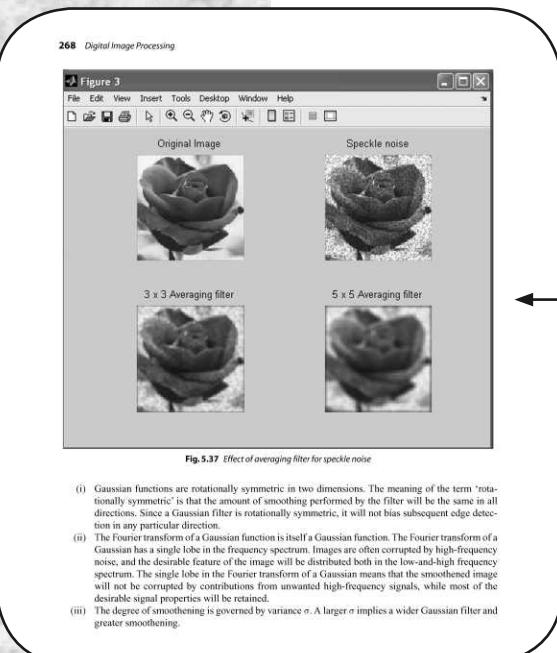
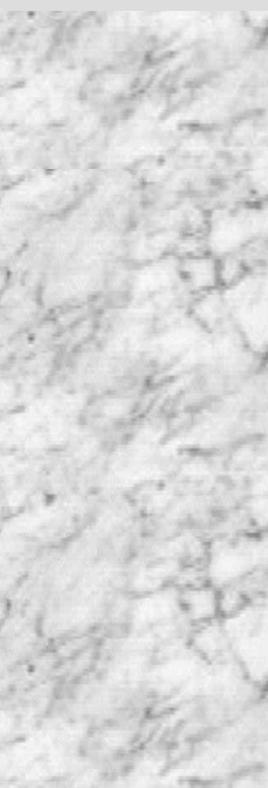
2.1 INTRODUCTION

Signals are variables that carry information. A signal is a function of one or more variables. An example of a one-dimensional signal is an ECG signal. An example of a 2D signal is a still image. The intensity of the image can vary along the 'x' and 'y' directions, and hence it is a 2D signal. Here 'x' and 'y' are called *spatial variables*.

A system basically processes the input signal and produces output signals. Thus, a system can be viewed as an operation or a set of operations performed on the input signal to produce an output signal.

Each chapter begins with an Introduction that gives a brief summary of the background and contents of the chapter.

Worked Examples are provided in sufficient number in each chapter and at appropriate locations, to aid the understanding of the text material.



270 Digital Image Processing

5.9 MEDIAN FILTER

Median filters are state-of-the-art non-linear filters that are often described in the spatial domain. A median filter smoothens the image by utilising the median of the neighbourhood. The concept of a median filter was introduced by Tukey in 1977. Its extension to two-dimensional images was discussed by Pratt in 1978. Median filters perform the following tasks to find each pixel value in the processed image:

1. All pixels in the neighbourhood of the pixel in the original image which are identified by the mask are sorted in the ascending (or) descending order.
2. The median of the sorted value is computed and is chosen as the pixel value for the processed image.

Example 5.5 Compute the median value of the marked pixel shown in Fig. 5.40 using a 3×3 mask.

1	5	7
2	4	6
3	2	1

Fig. 5.40 Data for Example 5.5

Solution The median value of the marked pixel is computed as follows:

Step 1 First, the pixel values are arranged in ascending order as follows:

1 2 2 3 4 5 6 7

Step 2 The median value of the ordered pixel is computed as follows:

X X Z Z 3 A A A 7

The median value is computed to be 3. Then, the original pixel value of 4 will be replaced by the computed median value of 3.

1	5	7
2	4	6
3	2	1

Original image data

After median filtering

When median filters are applied to an image, the pixel values which are very different from their neighbouring pixels will be eliminated. By eliminating the effect of such odd pixels, the values are assigned to the pixels that are representative of the values of the typical neighbouring pixels in the original image. This fact is illustrated in Example 5.6.

Example 5.6 Compute the median value of the marked pixels shown in Fig. 5.41 using a 3×3 mask.

18	29	33	26	32	24
34	128	24	172	28	23
22	19	32	31	20	26

Fig. 5.41 Input image

Solution Here the goal is to compute the median values of the marked pixels and replace the pixels 128, 24, 172 and 26 by their median values of the neighbourhood defined by the mask. The mask to be used is a 3×3 mask.

Other than conventional images like Lena and Barbara, natural images are used to illustrate the concept.

MATLAB is a scientific programming language that provides strong mathematical and numerical support for the implementation of algorithms. MATLAB examples related to image-processing concepts are given wherever necessary in the text.

514 Digital Image Processing

$$\hat{f}(m, n) = \begin{bmatrix} 67 & 81 & 81 & 67 \\ 67 & 81 & 67 & 67 \\ 81 & 67 & 67 & 67 \\ 67 & 67 & 67 & 81 \end{bmatrix}$$

By comparing the original block $f(m, n)$ with the reconstructed block $\hat{f}(m, n)$, we find that error is inevitable.

MATLAB Example 1: BTC *Read an image, apply BTC by choosing different block sizes. Comment on the observed result.*

Solution The MATLAB code that performs BTC of the input image is shown in Fig. 9.59 and the corresponding output is shown in Fig. 9.60. From the observed result, it is clear that as the block size increases, the quality of the reconstructed image decreases and the blocking artifact becomes visible.

```
%This program performs BTC of the input image
x = imread('C:\Documents and Settings\esakk1\Desktop\icecream.jpg');
x = im2gray(x);
x = imresize(x, [256 256]);
x = double(x);
x1 = x;
[n1, n1] = size(x);
blk = input('Enter the block size:');
for i = 1 : blk : n1
    for j = 1 : blk : n1
        y = x(i : i + (blk-1), j : j + (blk-1));
        m = sum(y);
        sig = std2(y);
        b = y > m;
        K = sum(b);
        if (K == blk) & (K == m)
            mu = 1;
        else
            mu = sig*sqrt((blk^2-K)/K);
        end
        x(i : i + (blk-1), j : j + (blk-1)) = b*mu + (1-b)*m;
    end
end
imshow(uint8(x1)), title('Original image')
figure, imshow(uint8(x)), title('Reconstructed image'),
xlabel(sprintf('The block size is %d', blk))
```

Fig. 9.59 MATLAB code to perform BTC

672 Digital Image Processing

Contourlet Transform

1. R H Bamberger and M-JT Smith, A Filter Bank for the Directional Decomposition of Images: Theory and Design, IEEE Trans. Signal Process., vol. 40, no. 4, pp. 882-893, April 1992
2. M N Do and M Vetterli, The Contourlet Transform: An Efficient Directional Multiresolution Image Representation, IEEE Trans. Image Process., vol. 14, no. 12, pp. 2091-2106, December 2005

Denoising

1. DL Donoho, Nonlinear Wavelet Methods for Recovery of Signals, Densities, and Spectra from Indirect and Noisy Data, in Proc. of Symposia in Applied Mathematics, pp. 173-205, AMS, 1993
2. DL Donoho, Denoising and Soft-Thresholding, IEEE Trans. Inf. Theory, vol. 41, pp. 613-627, 1995

Watermarking

1. C Voyatzis, N Nikita and I Pitas, Digital Watermarking: An Overview, 9th European Signal Processing Conference (EUSIPCO 98).
2. N Kaewkhamerd and KR Rao, Wavelet Based Image Watermarking Scheme, EUSIPCO 2000, Tampere, Finland, September 2000

Web References

1. Robi Polikar's excellent introduction to the concepts of wavelet <http://users.rowan.edu/~polikar/>
2. <http://www.wavelet.org/> is a very good site to know the theory and application of wavelets.
3. William Pearlman's website is a treasure island for people working in the area of SPIHT: <http://www.eeprf.princeton.edu/~pearlm/>
4. Muri Do's homepage give rich information related to contourlets and directional decomposition: <http://www.ece.tamu.edu/~murd/ publications/>
5. Martin Vetterli's homepage is a very useful source for a variety of wavelet information: <http://caesarwww.eepli.ee.tum.de/~vetterli/>
6. Professor Deepa Kundur's research page contains good articles related to watermarking: <http://www.ece.tamu.edu/~deepa/pubs.html>
7. For JPEG and JPEG2000 image compression standard the authors recommend the website www.jpeg.org/jpeg2000

To effectively use the internet resources, references to relevant web addresses are provided at the end of each chapter.

At the end of each chapter, a comprehensive list of book and journal references are provided.

242 Digital Image Processing

4.15 The 4×4 Hadamard matrix is given by $H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$. Check whether premultiplication of the

matrix H by the matrix $S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ puts the row in the Walsh sequence order.

References

Books

1. K R Rao and P C Yuen, *The Transform and Data Compression Handbook*, CRC Press, 2001
2. Alexander D Poularikas, *Transform and Applications*, CRC Press, 1996
3. R Gopinath, *The Fourier Transform and Its Applications*, McGraw-Hill, New York
4. K O Beauchamp, *Walsh Functions and their Applications*, Academic Press, 1975
5. Adam Druzd, *Elements of Data Compression*, Thomson Brookcole, 2002
6. N Ahmed and K R Rao, *Orthogonal Transforms for Digital Signal Processing*, Springer-Verlag, Berlin
7. S R Deans, *The Radon Transform and some of its Applications*, John Wiley, 1993

Journals

1. D A Bell, *Walsh Functions and Hadamard Matrices*, Electron. Lett., vol. 2, pp. 340-341, September 1966
2. D A Bell, *Walsh Functions and Computation of Statistical Variables from Principal Components*, Journal of Educational Psychology, vol. 24, pp. 498-520, 1933
3. J L Walsh, A closed set of normal orthogonal functions, *American Journal of Mathematics*, 1923
4. Russell M Mersereau and Alan V Oppenheim, *Digital Reconstruction of Multidimensional Signals from their Projections*, Proc. IEEE, vol. 62, pp. 1319-1338, October 1974
5. Frantisek Matas and Jan Flusser, *Image Representations via a Finite Radon Transform*, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 15, no. 10, pp. 996-1006, October 1993
6. V C Kiema and A J Laub, *The Singular Value Decomposition: Its Computation and Some Applications*, IEEE Trans. Autom. Control, vol. 25, pp. 164-176, April 1980

Web References

1. Dr K R Rao's teaching material: <http://www-ee.uta.edu/dip/>
2. Professor Min Wu's lecture material: <http://www.ece.umd.edu/class/enee631/>

Convolution and Correlation 149

Review Questions

1. Prove that convolution with a 2D separable filter can be accomplished by performing two one-dimensional convolutions.

Let $f(m, n)$ represent the input image and $h(m, n)$ represent the 2D separable filter. First the rows of the image is convolved with $h_0(m, n)$ and then the columns of that result with $h_1(n)$ or vice versa. This concept is represented mathematically as

$$h * f = \sum_{x=-\frac{M}{2}}^{\frac{M}{2}} \sum_{y=-\frac{N}{2}}^{\frac{N}{2}} h_0(x, y)f(m-x, n-y)$$

If the filter $h(m, n)$ is separable then

$$h * f = \sum_{x=-\frac{M}{2}}^{\frac{M}{2}} h_0(x, n) \sum_{y=-\frac{N}{2}}^{\frac{N}{2}} h_1(y)f(m-x, n-y)$$

$$h * f = h_0(m) * [h_1(n) * f(m, n)]$$

From the above expressions, it is clear that 2D convolution can be performed as two 1D convolutions.

2. Calculate the number of multiplications required to convolve a 2D filter with a 2D image (a)

(a) Compute the 2D convolution at a stretch. (b) Perform the 2D convolution as two 1D convolutions.

Assume the image is of size 100×100 pixels, and the filter is of size 10×10 .

- (a) The number of multiplications required to perform a 2D convolution, ignoring the border effect is given by

$$100 \times 100 \times 10 \times 10 = 10^6$$

- (b) The number of multiplications for two 1D convolution is

$$100 \times 100 \times 10 + 100 \times 100 \times 10 = 2 \times 10^6$$

From the results, it is obvious that performing a 2D convolution as two 1D convolutions reduces the computational complexity.

3. Give few applications of 2D convolution in the field of image processing.

Convolution is a powerful operation that can be used to perform filtering. A high-pass filter can be used to enhance the edges in an image. The high-pass filtering operation can be achieved by convolving the input image $f(m, n)$ with the spatial mask $h(m, n)$ which is given by

$$h(m, n) = \frac{1}{9} \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

A low-pass filter can be used to remove high-frequency noise in an image. The low-pass filtering can be achieved by convolving the input image $f(m, n)$ with the spatial mask $g(m, n)$ which is given by

$$g(m, n) = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Each chapter contains a set of Review Questions with answers. These review questions provide the essence of the concepts discussed in each chapter.

A set of problems are given as exercise to the students. These are very helpful to teachers in setting class work, assignments, quizzes and examinations.

670 Digital Image Processing

7. Mention two differences between EZW and EBCOT coding.

The EZW coding explores the interband dependencies of the wavelet coefficients, whereas in EBCOT coding the interband dependencies are not exploited. In EZW, there are two passes, namely, the dominant pass and the refinement pass; whereas in EBCOT coding there are three passes, namely, significant pass, refinement pass and clean-up pass.

Problems

12.1 Prove the Heisenberg uncertainty principle in signal processing.

12.2 Calculate the Haar transform of the following image:

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

12.3 For the coefficients given in Fig. 12.66, find the bit stream generated by EZW coder. Also, decode the generated bit stream.

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

Fig.12.66 Coefficients

12.4 For the coefficients given in Fig. 12.66, find the bit stream generated by SPIHT coder. Also, decode the generated bit stream.

12.5 What are the advantages of representing the signal in terms of contourlet basis?

12.6 List the advantages of performing watermarking in the frequency domain.

12.7 Derive the perfect reconstruction condition for a two-channel filter bank.

12.8 Construct a fully populated approximation pyramid and corresponding prediction residual pyramid for the image

1	2	3	4
6	6	7	8
9	10	11	12
13	14	15	16

Use 2×2 block neighbourhood averaging for the approximation filter and omit the interpolation filter.

12.9 Compute the Haar transform $T = HFH^T$ of the 2×2 image $F = \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix}$. Compute also the inverse Haar transform $F = H^T T^H$ of the obtained result.

APPENDIX-IV

Objective Type Questions

1. The third bit-plane corresponding to the image $\begin{bmatrix} 4 & 3 \\ 5 & 2 \end{bmatrix}$ is

(a) $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$

2. The 2D DFT of the image $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ is

(a) $\begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} 0 & 4 \\ 0 & 0 \end{bmatrix}$ (c) $\begin{bmatrix} 0 & 0 \\ 4 & 0 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

3. The transform which possesses the 'multi-resolution' property is

(a) Fourier transform (b) short-time Fourier transform
(c) cosine transform (d) wavelet transform

4. The transform which is widely used to detect 'lines' in an image is

(a) Fourier transform (b) Hough transform
(c) cosine transform (d) Haar transform

5. The transform which possesses the highest 'energy compaction' property is

(a) Fourier transform (b) Walsh transform
(c) slant transform (d) KL transform

6. The condition for the transform A to be unitary is (A denotes conjugate and T denotes transpose)

(a) $A^{-1} = \frac{1}{A}$ (b) $A^{-1} = A^T$ (c) $A^{-1} = \frac{1}{A^T}$ (d) $A^{-1} = \frac{1}{\det(A)}$

7. Statement 1: All prefix codes are uniquely decodable.
Statement 2: All uniquely decodable codes must be prefix codes.

(a) Statement 1 and Statement 2 are always true.
(b) Statement 1 is always true, but Statement 2 is not always true.
(c) Both statements 1 and 2 are wrong.
(d) Statement 2 is always true and Statement 1 is not always true.

8. Which one of the following is a lossy coding?

(a) Run-length coding (b) Uniform quantizer
(c) Huffman coding (d) Predictive coding without quantizer

9. In a DPCM coder, which of the following needs to be quantized?

(a) The reconstruction value (b) The difference between prediction value and the original value
(c) The prediction value (d) The transform coefficient

10. What does the definition of entropy tell us?

(a) The minimum number of bits required to encode a source without distortion
(b) The upper bound to encode a source without distortion
(c) The average number of bits to encode a source without distortion
(d) The average number of bits to encode a source given a certain distortion

11. In an image compression system, 16384 bits are used to represent a 128×128 image with 256 gray levels. What is the compression ratio for this system?

(a) 4 (b) 8 (c) 12 (d) 16

Objective questions enable the user to have a clear comprehension of the subject matter. Answers to all the objective questions are provided.

A bulleted summary gives the essence of each chapter in brief.

578 Digital Image Processing

10.13 SALIENT FEATURES OF MORPHOLOGICAL APPROACH

The salient features of morphological approach are summarised below:

1. Morphological operations provide the systematic alteration of the geometric content of an image while maintaining the stability of the important geometric characteristics.
2. There exists a well-developed morphological algebra that can be employed for representation and optimisation.
3. It is possible to express digital algorithms in terms of a very small class of primitive morphological operations.
4. There exist rigorous representation theorems by means of which one can obtain the expression of morphological filters in terms of the primitive morphological operations.
5. The linear transformations of functions and morphological operations are characterised by their non-invertibility.
6. They remove information of greater and greater extent as the size of the structural element increases.
7. Image processing through iterative morphological transformations can, therefore, be conceived as a process of selective information removal where irrelevant details are irrecoverably destroyed, thereby enhancing the contrast of essential function features.

Summary

- Morphological image processing is based on the idea of probing an image with a small shape or template known as a structuring element. It is possible to use structuring elements of different shapes to perform a specific task.
- The basic morphological operations are dilation and erosion.
- In the dilation operation, the image is probed with the structuring element by successively placing the origin of the structuring element to all possible pixel locations; the output is 1 if the structuring element and the image have a non-zero intersection and 0 otherwise.
- In the erosion operation, the image is probed with the structuring element by successively placing the origin of the structuring element to all possible pixel locations; the output is 1 if the structuring element is completely contained in the image and 0 otherwise.
- The opening operation is defined as erosion followed by dilation. The opening operation has the effect of eliminating small and thin objects, smoothing the boundaries of large objects without changing the general appearance.
- The closing operation is defined as dilation followed by erosion. The closing operation has the effect of filling small and thin holes in an object, connecting nearby objects and generally smoothing the boundaries of large objects without changing the general appearance.
- Morphological operations such as opening and closing can be regarded as morphological filters.
- The hit-or-miss operation probes the inside and outside of objects at the same time, using two separate structuring elements. A pixel belonging to an object is preserved by the hit-or-miss operation if the first structuring element translated to that pixel fits the object, and the second structuring element misses the object. The hit-or-miss operation is useful for the detection of specific shapes.
- The thinning operation accomplishes erosion without breaking objects. Thinning can be accomplished as a two-step approach, with the first step being erosion that marks all candidates for removal without actually removing them, and in the second pass, candidates that do not destroy the connectivity are removed.

Glossary of Image Processing Terms

- Archival image** A digital image taken at the highest practicable resolution and stored securely
- Access images** A term used to denote low-resolution images (thumbnails, 1/4 screen images) that are made available (usually at no cost) through the Internet
- Adaptive filter** A filter whose behaviour changes in response to variations in local image properties
- Additive primary colour** The colours red, green and blue which, when added in different combinations, can produce all the colours
- Affine transformation** A first-order geometric transformation that involves a combination of translation, scaling, rotation and skewing
- Algorithm** A computer program that will perform tasks, e.g., compression of file sizes
- Aliasing** A phenomenon which occurs when an image is undersampled; due to aliasing, the information with a high spatial frequency is incorrectly represented, manifesting itself as an artifact with a lower spatial frequency
- Alpha-trimmed mean filter** A filter that sorts pixel values from the neighbourhood into ascending order, discards a certain number of values at either end of the list and then outputs the mean of the remaining values
- Amplitude spectrum** A measure of how much of each frequency component is present in an image
- Analog image** An image characterised by a physical magnitude varying continuously in space
- Analog-to-digital converter** Used to convert an analog signal into digital form
- Anti-aliasing** The technique of minimising the distortion artifacts, known as aliasing, when representing a high-resolution signal at a lower resolution
- Area array** A common type of detector arrangement within a digital camera containing a fixed number of horizontal and vertical pixels
- Artifact** Unwanted blemishes, which may have been introduced to an image by electrical noise during scanning or compression
- Bandwidth** A measure of data speed in bits per second in digital systems; a high bandwidth network is required for fast transfer of image files as they typically contain large amounts of data
- Basis function** Used to represent an image; the basis function should be linearly independent and it should span the space
- Binary** Computer data made up of a series of 0s or 1s; each individual character is referred as a bit
- Binary image** Binary image takes only two pixel values which are either '0' or '1'
- Byte** A collection of eight bits
- Bit Depth** Number of bits used to describe the colour of each pixel; greater bit depth allows more colours to be used in the colour palette for the image—8-bits per pixel will allow 256 colours; 8-bits per colour component in a RGB image will allow 16777216 colours (256 × 256 × 256)

A list of commonly used terms in digital image processing and their meanings are given in the form of a glossary. The glossary will serve as a dictionary for the students.

1

Learning Objectives

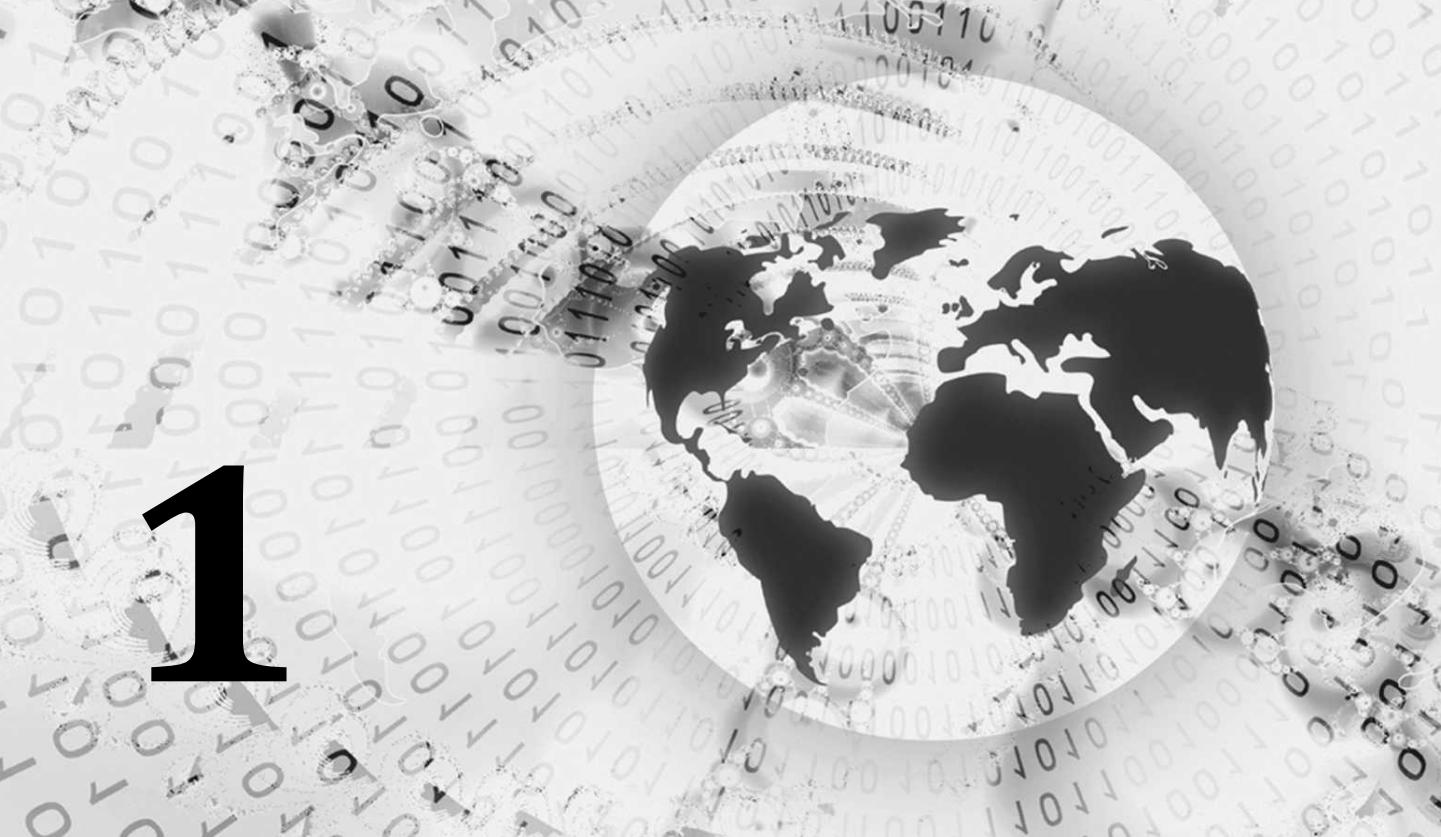
This chapter provides an overview of the image-processing system which includes various elements like image sampling, quantisation, processing, storage and display. On completion of this chapter, the reader is expected to be familiar with the following concepts:

image sampling
image types
image sensors
image storage
image processing
image display

Introduction to Image-processing System

1.1 INTRODUCTION

Digital images play an important role, both in daily-life applications such as satellite television, magnetic resonance imaging, computer tomography as well as in areas of research and technology such as geographical information systems and astronomy. An image is a 2D representation of a three-dimensional scene. A digital image is basically a numerical representation of an object. The term *digital image processing* refers to the manipulation of an image by means of a processor. The different elements of an image-processing system include image acquisition, image storage, image processing and display. This chapter begins with the basic definition of a digital image and is followed by a detailed discussion on two-dimensional sampling. This is followed by different elements of image processing systems.



2 Digital Image Processing

1.1.1 Image

An image is a two-dimensional function that represents a measure of some characteristic such as brightness or colour of a viewed scene. An image is a projection of a 3D scene into a 2D projection plane. It can be defined as a two-variable function $f(x, y)$ where for each position (x, y) in the projection plane, $f(x, y)$ defines the light intensity at this point.

1.1.2 Analog Image

An analog image can be mathematically represented as a continuous range of values representing position and intensity. An analog image is characterised by a physical magnitude varying continuously in space. For example, the image produced on the screen of a CRT monitor is analog in nature.

1.1.3 Digital Image

A digital image is composed of picture elements called *pixels*. Pixels are the smallest sample of an image. A pixel represents the brightness at one point. Conversion of an analog image into a digital image involves two important operations, namely, sampling and quantisation, which are illustrated in Fig. 1.1.

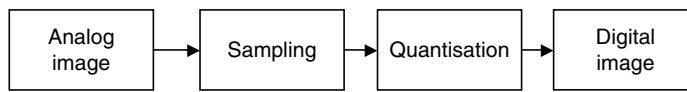


Fig. 1.1 Digital image from analog image

Advantages of Digital Images The advantages of digital images are summarised below:

- The processing of images is faster and cost-effective.
- Digital images can be effectively stored and efficiently transmitted from one place to another.
- When shooting a digital image, one can immediately see if the image is good or not.
- Copying a digital image is easy. The quality of the digital image will not be degraded even if it is copied for several times.
- Whenever the image is in digital format, the reproduction of the image is both faster and cheaper.
- Digital technology offers plenty of scope for versatile image manipulation.

Drawbacks of Digital Images Some of the drawbacks of digital image are given below:

- Misuse of copyright has become easier because images can be copied from the Internet just by clicking the mouse a couple of times.
- A digital file cannot be enlarged beyond a certain size without compromising on quality.
- The memory required to store and process good-quality digital images is very high.
- For real-time implementation of digital-image-processing algorithms, the processor has to be very fast because the volume of data is very high.

1.1.4 Digital Image Processing

The processing of an image by means of a computer is generally termed digital image processing. The advantages of using computers for the processing of images are summarised below:

(1) Flexibility and Adaptability The main advantage of digital computers when compared to analog electronic and optical information processing devices is that no hardware modifications are necessary in order to reprogram digital computers to solve different tasks. This feature makes digital computers an ideal device for processing image signals adaptively.

(2) Data Storage and Transmission With the development of different image-compression algorithms, the digital data can be effectively stored. The digital data within the computer can be easily transmitted from one place to another.

The only limitation of digital imaging and digital image processing are memory and processing speed capabilities of computers. Different image-processing techniques include image enhancement, image restoration, image fusion and image watermarking.

1.1.5 Digital Image Representation

A digital image is a two-dimensional discrete signal. A digital image is also an $N \times N$ array of elements. Each element in the array is a number which represents the sampled intensity. For example, the representation of a 4×4 image in matrix format and its three-dimensional view is shown in Fig. 1.2.

Converting an image into a digital format can be done either with a digital camera, or by a scanner. Digital images can be created directly on a computer screen. However, it is restricted both in spatial coordinates (sampling) and in its allowed intensities (quantisation).

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

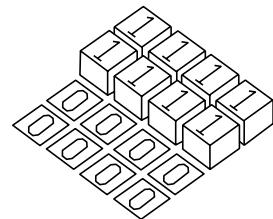


Fig. 1.2 Digital-image representation

1.1.6 Neighbours of a Pixel

A pixel will have four neighbours if the neighbours exist in the EAST, WEST, NORTH and SOUTH direction. The four neighbours of the pixel 'P' pixel are represented in Fig. 1.3.

A pixel 'P' will have eight neighbours if the neighbours are in eight directions such as EAST, WEST, NORTH, SOUTH, NORTH-EAST (NE), NORTH-WEST (NW), SOUTH-EAST (SE) and SOUTH-WEST (SW). The eight neighbours of the pixel 'P' are represented in Fig. 1.4.

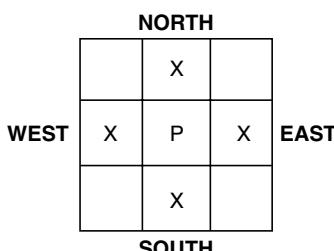


Fig. 1.3 Four neighbours of the pixel P

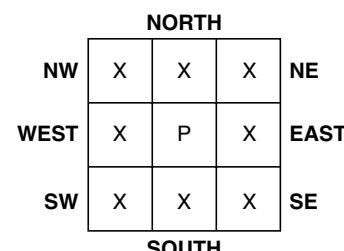


Fig. 1.4 Eight neighbours of the pixel P

1.2 IMAGE SAMPLING

Sampling is the process of measuring the brightness information only at a discrete spatial location. A continuous image function $f(x, y)$ can be sampled using a discrete grid of sampling points in the plane.

1.2.1 Theory of 2D Sampling

Let $f(x, y)$ represent the analog image. It means that function $f(x, y)$ is defined at each and every value of x and y . The discrete version of $f(x, y)$ is obtained by defining $f(x, y)$ at specific instants. This is mathematically represented as

$$f(m, n) = f(m\Delta x, n\Delta y) \quad (1.1)$$

where Δx and Δy are positive real constants. It is to be noted that $f(m, n)$ is not defined for all values of m and n . Here Δx and Δy are known as *sampling intervals*.

The 2D sampling starts with the analog signal $f(x, y)$. On taking the Fourier transform of this analog signal, we get the spectrum of $f(x, y)$ which is denoted by $F(\Omega_1, \Omega_2)$ where

$$F(\Omega_1, \Omega_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j\Omega_1 x} e^{-j\Omega_2 y} dx dy \quad (1.2)$$

In Eq. (1.2) $e^{-j\Omega_1 x}$ and $e^{-j\Omega_2 y}$ represent the Fourier basis.

On taking inverse Fourier transform, we get

$$f(x, y) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\Omega_1, \Omega_2) e^{j\Omega_1 x} e^{j\Omega_2 y} d\Omega_1 d\Omega_2 \quad (1.3)$$

When we perform sampling of the analog signal $f(x, y)$, the values are specified only at specific instants which is represented mathematically as

$$f(m, n) = f(m\Delta x, n\Delta y) \quad (1.4)$$

On taking the inverse Fourier transform of this sampled signal, we get

$$f(m, n) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\Omega_1, \Omega_2) e^{j\Omega_1 m\Delta x} e^{j\Omega_2 n\Delta y} d\Omega_1 d\Omega_2 \quad (1.5)$$

For a discrete signal, we define

$$\omega_1 = \Omega_1 \Delta x \quad (1.6)$$

and

$$\omega_2 = \Omega_2 \Delta y \quad (1.7)$$

where ω_1 and ω_2 are expressed in radians.

From Eq. (1.6), we have $\frac{\omega_1}{\Delta x} = \Omega_1$. Differentiating Ω_1 , we get

$$\frac{d\omega_1}{\Delta x} = d\Omega_1 \quad (1.8)$$

Similarly from Eq. (1.7), we get

$$\frac{d\omega_2}{\Delta y} = d\Omega_2 \quad (1.9)$$

Substituting Eq. (1.8) and (1.9) in Eq. (1.5), we get

$$f(m, n) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F\left(\frac{\omega_1}{\Delta x}, \frac{\omega_2}{\Delta y}\right) e^{j\omega_1 m} e^{j\omega_2 n} \frac{d\omega_1}{\Delta x} \frac{d\omega_2}{\Delta y} \quad (1.10)$$

$$f(m, n) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{\Delta x \Delta y} F\left(\frac{\omega_1}{\Delta x}, \frac{\omega_2}{\Delta y}\right) e^{j\omega_1 m} e^{j\omega_2 n} d\omega_1 d\omega_2 \quad (1.11)$$

The signal $f(m, n)$ is discretised; hence the integral can be replaced by summation. The double integral over the entire (ω_1, ω_2) plane can be broken into an infinite series of integrals, each of which is over a square of area $4\pi^2$. The range of ω_1 and ω_2 is given by $-\pi + 2\pi k_1 \leq \omega_1 < \pi + 2\pi k_1$; $-\pi + 2\pi k_2 \leq \omega_2 < \pi + 2\pi k_2$. Incorporating this concept in Eq. (1.11), we get

$$f(m, n) = \frac{1}{4\pi^2} \iint_{\text{SQ}(k_1, k_2)} \frac{1}{\Delta x \Delta y} \sum_{k_1} \sum_{k_2} F\left(\frac{\omega_1}{\Delta x}, \frac{\omega_2}{\Delta y}\right) e^{j\omega_1 m} e^{j\omega_2 n} d\omega_1 d\omega_2 \quad (1.12)$$

In order to change the limits of the integral so as to remove the dependence of the limits of integration on k_1 and k_2 , ω_1 is replaced by $\omega_1 - 2\pi k_1$ and ω_2 is replaced by $\omega_2 - 2\pi k_2$. On including this modification in Eq. (1.12), we get

$$f(m, n) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{1}{\Delta x \Delta y} \sum_{k_1} \sum_{k_2} F\left(\frac{\omega_1 - 2\pi k_1}{\Delta x}, \frac{\omega_2 - 2\pi k_2}{\Delta y}\right) e^{j(\omega_1 - 2\pi k_1)m} e^{j(\omega_2 - 2\pi k_2)n} d\omega_1 d\omega_2 \quad (1.13)$$

$$f(m, n) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{1}{\Delta x \Delta y} \sum_{k_1} \sum_{k_2} F\left(\frac{\omega_1 - 2\pi k_1}{\Delta x}, \frac{\omega_2 - 2\pi k_2}{\Delta y}\right) e^{j\omega_1 m} e^{-j2\pi k_1 m} e^{j\omega_2 n} e^{-j2\pi k_2 n} d\omega_1 d\omega_2 \quad (1.14)$$

The exponential term in Eq. (1.14) is equal to one for all values of the integer variables m, k_1, n and k_2 . Hence Eq. (1.14) reduces to

$$f(m, n) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{1}{\Delta x \Delta y} \sum_{k_1} \sum_{k_2} F\left(\frac{\omega_1 - 2\pi k_1}{\Delta x}, \frac{\omega_2 - 2\pi k_2}{\Delta y}\right) e^{j\omega_1 m} e^{j\omega_2 n} d\omega_1 d\omega_2 \quad (1.15)$$

The above equation represents the inverse Fourier transform of $F(\omega_1, \omega_2)$ where

$$F(\omega_1, \omega_2) = \frac{1}{\Delta x \Delta y} \sum_{k_1} \sum_{k_2} F\left(\frac{\omega_1 - 2\pi k_1}{\Delta x}, \frac{\omega_2 - 2\pi k_2}{\Delta y}\right) \quad (1.16)$$

6 Digital Image Processing

Equation (1.16) can be written as

$$F(\omega_1, \omega_2) = \frac{1}{\Delta x \Delta y} \sum_{k_1} \sum_{k_2} F\left(\Omega_1 - \frac{2\pi k_1}{\Delta x}, \Omega_2 - \frac{2\pi k_2}{\Delta y}\right) \quad (1.17)$$

The concepts discussed so far can be approached in another way, i.e., by multiplying the analog image with a two-dimensional comb function. The 2D comb function is a rectangular grid of points and is illustrated in Fig. 1.5. The spaces between successive grid points in the x and y direction are Δx and Δy respectively. The three-dimensional view of the comb function is shown in Fig. 1.6. The 2D comb function, which is otherwise known as *bed-of-nail function*, is defined as

$$\text{comb}(x, y, \Delta x, \Delta y) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \delta(x - k_1 \Delta x, y - k_2 \Delta y) \quad (1.18)$$

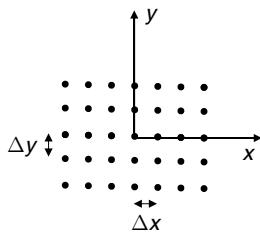


Fig. 1.5 2D view of a comb function

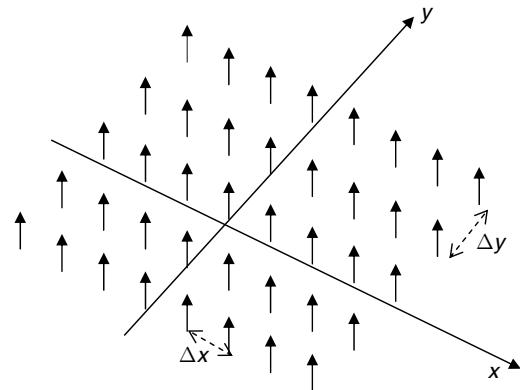


Fig. 1.6 Three-dimensional view of a comb function

After multiplying the analog image $f(x, y)$ with the 2D comb function, we get the discrete version of the analog image which is given by

$$f(m, n) = f(x, y) \times \text{comb}(x, y, \Delta x, \Delta y) \quad (1.19)$$

$$f(m, n) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} f(k_1 \Delta x, k_2 \Delta y) \delta(x - k_1 \Delta x, y - k_2 \Delta y) \quad (1.20)$$

We know that convolution in spatial domain is equal to multiplication in the frequency domain and vice versa. For the purpose of analysis in the frequency domain, let us take the Fourier transform of the input analog image and the 2D comb function.

The Fourier transform of the signal $f(x, y)$ is $f(\Omega_1, \Omega_2)$.

The Fourier transform of 2D comb function is another comb function which is represented by

$$\text{comb}(\Omega_1, \Omega_2) = FT(\text{comb}(x, y, \Delta x, \Delta y)) \quad (1.21)$$

$$\text{comb}(\Omega_1, \Omega_2) = \frac{1}{\Delta x} \frac{1}{\Delta y} \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} \delta\left(\Omega_1 - \frac{p}{\Delta x}, \Omega_2 - \frac{q}{\Delta y}\right) \quad (1.22)$$

$$\text{comb}(\Omega_1, \Omega_2) = \frac{1}{\Delta x} \frac{1}{\Delta y} \text{comb}\left(\Omega_1, \Omega_2, \frac{1}{\Delta x}, \frac{1}{\Delta y}\right) \quad (1.23)$$

Now the spectrum of the 2D comb function is convolved with the spectrum of the analog image which is given by $F(\Omega_1, \Omega_2) \otimes \text{comb}(\Omega_1, \Omega_2)$.

$$F(\omega_1, \omega_2) = F(\Omega_1, \Omega_2) \otimes \text{comb}(\Omega_1, \Omega_2) \quad (1.24)$$

Substituting Eq. (1.22) in Eq. (1.24), we get

$$F(\omega_1, \omega_2) F = (\Omega_1, \Omega_2) \otimes \frac{1}{\Delta x} \frac{1}{\Delta y} \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} \delta\left(\Omega_1 - \frac{p}{\Delta x}, \Omega_2 - \frac{q}{\Delta y}\right) \quad (1.25)$$

Upon convolving the spectrum of the signal with the spectrum of the comb function, we get

$$F(\omega_1, \omega_2) = \frac{1}{\Delta x} \frac{1}{\Delta y} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} F(\Omega_1 - k, \Omega_2 - l) \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} \delta\left(k - \frac{p}{\Delta x}, l - \frac{q}{\Delta y}\right) \quad (1.26)$$

As summation is a linear operator, interchanging the order of summation, we get

$$F(\omega_1, \omega_2) = \frac{1}{\Delta x} \frac{1}{\Delta y} \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} F(\Omega_1 - k, \Omega_2 - l) \delta\left(k - \frac{p}{\Delta x}, l - \frac{q}{\Delta y}\right) \quad (1.27)$$

$$F(\omega_1, \omega_2) = \frac{1}{\Delta x} \frac{1}{\Delta y} \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} F\left(\Omega_1 - \frac{p}{\Delta x}, \Omega_2 - \frac{q}{\Delta y}\right) \quad (1.28)$$

Equation (1.28) resembles Eq. (1.17).

1.2.2 Retrieving the Image from its Samples

We know that discreteness in one domain leads to periodicity in another domain. Hence sampling in the spatial domain leads to periodic spectrum in the frequency domain, which is illustrated in Fig. 1.7.

In order to retrieve the original image from the sampled spectrum, the following conditions have to be satisfied.

$$\omega_{xs} > 2\omega_{x0} \quad (1.29)$$

where $\omega_{xs} = \frac{1}{\Delta x}$ and $2\omega_{x0}$ is the bandwidth of the spectrum in the ω_1 direction.

Similarly,

$$\omega_{ys} > 2\omega_{y0} \quad (1.30)$$

where $\omega_{ys} = \frac{1}{\Delta y}$ and $2\omega_{y0}$ is the bandwidth of the spectrum in the ω_2 direction. The condition given in Eqs. (1.29) and (1.30) implies that the sampling frequency should be greater than twice the maximum signal frequency, which is generally termed the *sampling theorem*. Here, $2\omega_{x0}$ and $2\omega_{y0}$ are called *Nyquist rates*.

8 Digital Image Processing

A low-pass filter is normally employed in order to extract the desired spectrum. The transfer function of the low-pass filter is given as follows:

$$H(\omega_1, \omega_2) = \begin{cases} \frac{1}{\omega_{1s} \omega_{2s}}, & (\omega_1, \omega_2) \in \text{region of support} \\ 0, & \text{otherwise} \end{cases} \quad (1.31)$$

The region of support is indicated as \mathfrak{R} in Fig. 1.7. The continuous image can be obtained from the sampled spectrum by multiplying the sampled spectrum with the low pass filter which is given as

$$\hat{F}(\omega_1, \omega_2) = H(\omega_1, \omega_2) \times F(\omega_1, \omega_2) \quad (1.32)$$

By taking inverse Fourier transform, we get the continuous image as

$$\hat{F}(x, y) = F^{-1}\{\hat{F}(\omega_1, \omega_2)\} \quad (1.33)$$

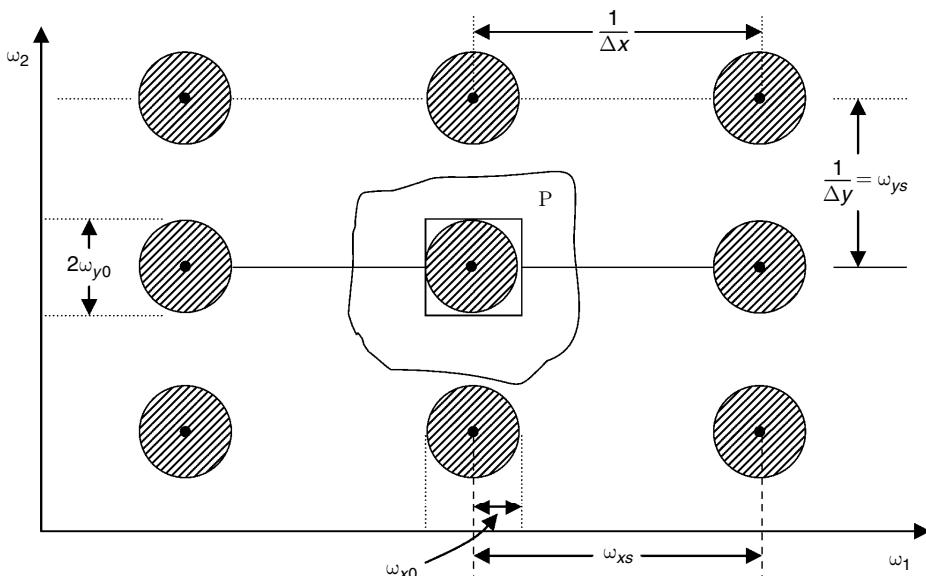


Fig. 1.7 Periodic spectrum of the sampled image

1.2.3 Violation of Sampling Criterion

In this section, let us discuss the consequences of violation of the sampling criterion. Violation of sampling criterion given in Eq. (1.29) and (1.30) leads to aliasing. Aliasing basically occurs due to under-sampling.

Violation of sampling criterion given in Eq. (1.29) leads to overlapping of the spectrum in the ω_1 direction, which is illustrated in Fig. 1.8. Here, $\omega_{xs} < 2\omega_{x0}$, whereas $\omega_{ys} > 2\omega_{y0}$.

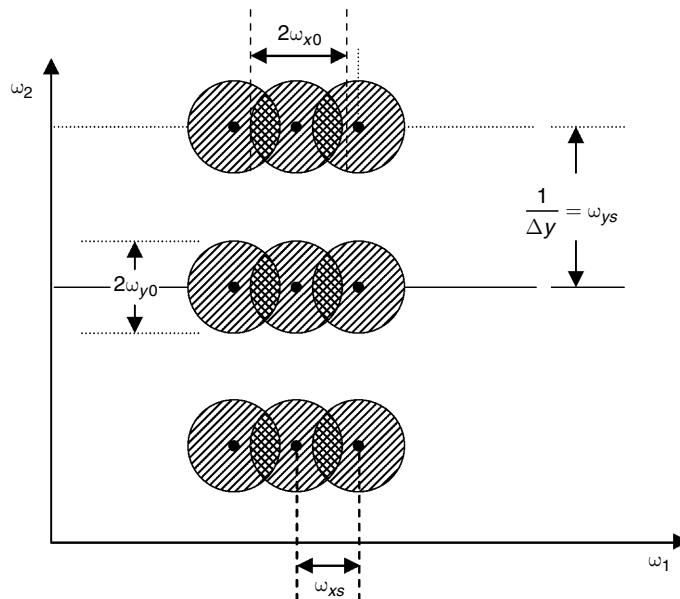


Fig. 1.8 Under-sampling along the ω_1 direction

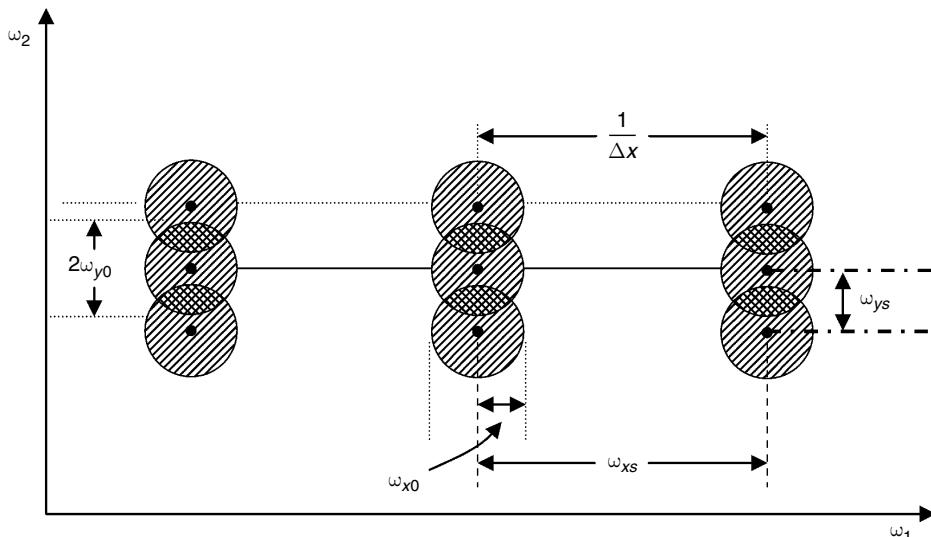


Fig. 1.9 Under-sampling along the ω_2 direction

Similarly, violation of the sampling criterion given in Eq. (1.30) leads to overlapping of the spectrum in the ω_2 direction which is illustrated in Fig. 1.9. Here, $\omega_{xs} > 2\omega_{x0}$, whereas $\omega_{ys} < 2\omega_{y0}$.

Simultaneous violation of the sampling criterion in Eqs (1.29) and (1.30) leads to the overlapping of the spectrum in both the ω_1 and ω_2 directions which is illustrated in Fig. 1.10. Here, $\omega_{xs} < 2\omega_{x0}$ and $\omega_{ys} < 2\omega_{y0}$.

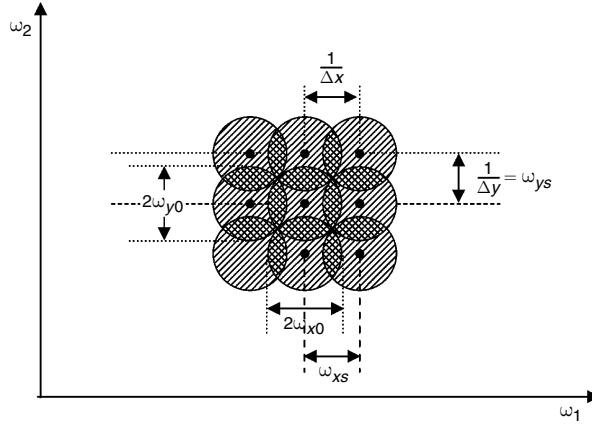


Fig. 1.10 Under-sampling along the ω_1 and ω_2 directions

Example 1.1 The image $f(x, y) = 4\cos 2\pi(2x + y)$ is to be sampled such that one can reconstruct the signal from its samples without errors. Suggest a sampling scheme.

Solution The image is given by $f(x, y) = 4\cos 2\pi(2x + y)$. The given image can be written as

$$f(x, y) = 4 \times \left(\frac{e^{j2\pi(2x+y)} + e^{-j2\pi(2x+y)}}{2} \right) = 2 \times [e^{j2\pi(2x+y)} + e^{-j2\pi(2x+y)}] \quad (1.34)$$

Taking Fourier transform of the image, we get

$$F(\omega_1, \omega_2) = 2[\delta(\omega_1 + 2, \omega_2 + 1) + \delta(\omega_1 - 2, \omega_2 - 1)]. \quad (1.35)$$

This implies $\omega_{x0} = 2$ and $\omega_{y0} = 1$. To avoid aliasing, $\omega_{xs} > 2\omega_{x0}$ and $\omega_{ys} > 2\omega_{y0}$. In this case, $\omega_{xs} > 4$ and $\omega_{ys} > 2$. Also, Δx and Δy should be smaller than the Nyquist sampling rate. That is, $\Delta x < \frac{1}{2\omega_{x0}}$ and $\Delta y < \frac{1}{2\omega_{y0}}$. Δx should be less than 0.25 and Δy should be less than 0.5. Choosing $\Delta x = 0.2$ and $\Delta y = 0.4$, aliasing can be avoided. This corresponds to $\omega_{xs} = \frac{1}{\Delta x} = \frac{1}{0.2} = 5$ and $\omega_{ys} = \frac{1}{\Delta y} = \frac{1}{0.4} = 2.5$. The sampled spectrum is given by

$$F(\omega_1, \omega_2) = \omega_{xs}\omega_{ys} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} X(\omega_1 - k\omega_{xs}, \omega_2 - l\omega_{ys}) \quad (1.36)$$

Substituting the values of ω_{1s} , ω_{2s} we get

$$F(\omega_1, \omega_2) = 12.5 \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} X(\omega_1 - 5k, \omega_2 - 2.5l) \quad (1.37)$$

Substituting the expression of $F(\omega_1, \omega_2)$ from Eq. (1.37) in Eq. (1.35), we get

$$F(\omega_1, \omega_2) = 25 \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(\omega_1 + 2 - 5k, \omega_2 + 1 - 2.5l) + \delta(\omega_1 - 2 - 5k, \omega_2 - 1 - 2.5l)$$

Example 1.2 An image is given by the 2D function $f(x, y) = 2\cos[2\pi(4x + 6y)]$ which is sampled on an infinite grid of points with sampling intervals $\Delta x = 0.1$ and $\Delta y = 0.2$, in the x and y directions respectively. Determine the

- (i) Fourier transform of the sampling function
- (ii) Fourier transform of the sampled image before low-pass filtering
- (iii) Fourier transform of the image after it has been low-pass filtered.
- (iv) reconstructed image

In order to reconstruct the original image from sampled data without distortion, what are the maximum intervals Δx and Δy that can be used, and the minimum and maximum bandwidths of the low-pass reconstruction filters that can be used?

Solution The sampling frequency along the x -direction is given by $\omega_{xs} = \frac{1}{0.1} = 10$.

The sampling frequency along the y -direction is given by $\omega_{ys} = \frac{1}{0.2} = 5$.

- (i) The Fourier transform of the given 2D function $f(x, y)$ is given by

$$F(\omega_1, \omega_2) = \delta(\omega_1 - 4, \omega_2 - 6) + \delta(\omega_1 + 4, \omega_2 + 6) \quad (1.38)$$

Also, $F(\omega_1, \omega_2)$ is zero for $|\omega_1| > 4$, $|\omega_2| > 6$. This implies $\omega_{x0} = 4$, $\omega_{y0} = 6$.

The spectrum $F(\omega_1, \omega_2)$ will be periodic due to sampling and it is diagrammatically represented in Fig. 1.11. The periodicity along the ω_1 axis is 10, and along the ω_2 axis is 5. In Fig. 1.11, \star represents the sampling interval along the x and y direction. Here the symbol \bullet represents the occurrences of $F(\omega_1, \omega_2)$. The letter \mathfrak{R} indicates the region of support of the low pass filter. The points which lie within the region of support of the low-pass filter alone will be considered for reconstruction.

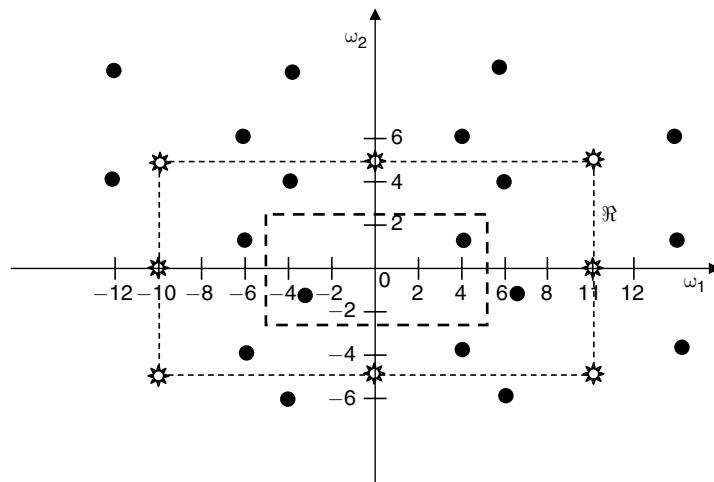


Fig. 1.11 Representation of $F(\omega_1, \omega_2)$

The Fourier transform of the sampling function is given by

$$50 \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(\omega_1 - 10k, \omega_2 - 5l) \quad (1.39)$$

(ii) The Fourier transform of the sampled image before low-pass filtering is given by

$$F(\omega_1, \omega_2) = 50 \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(\omega_1 - 4 - 10k, \omega_2 - 6 - 5l) + \delta(\omega_1 + 4 - 10k, \omega_2 + 6 - 5l) \quad (1.40)$$

(iii) The low-pass filter has a frequency response

$$H(\omega_1, \omega_2) = \begin{cases} \frac{1}{\omega_{xs}\omega_{ys}}, & |\omega_1| \leq 5 \text{ and } |\omega_2| \leq 2.5 \\ 0, & \text{otherwise} \end{cases} \quad (1.41)$$

The Fourier transform of the image after it has been low-pass filtered is given by the equation

$$\tilde{F}(\omega_1, \omega_2) = \delta(\omega_1 - 4, \omega_2 - 1) + \delta(\omega_1 + 4, \omega_2 + 1) \quad (1.42)$$

(iv) The reconstructed image is obtained by taking the inverse Fourier transform of the equation

$$\tilde{F}(\omega_1, \omega_2) = \delta(\omega_1 - 4, \omega_2 - 1) + \delta(\omega_1 + 4, \omega_2 + 1) \text{ to get}$$

$$\hat{f}(x, y) = 2 \cos[2\pi(4x + y)] \quad (1.43)$$

We can expect aliasing effect as the sampling rate is lower than the Nyquist frequency in the y axis.

According to the Nyquist criterion, $\frac{1}{\Delta x} > 2\omega_{x0}$. Hence, $\Delta x < 0.125$ and similarly, $\Delta y < 0.0833 = \frac{1}{12}$.

Using low-pass filters, the maximum values for Δx and Δy are $0.125 - \xi$ and $0.0833 - \xi$ where $\xi > 0$ can be taken as an arbitrarily small positive real number.

Example 1.3 Suppose an image of dimension 4×6 inches has details to the frequency of 400 dots per inch in each direction. How many samples are required to preserve the information in the image?

Solution The bandwidth is 400 in both the directions; therefore samples must be taken at 800 dots per inch in both the dimensions. A total of $4 \times 800 \times 3 \times 800 = 7680000$ samples are needed.

1.3 QUANTISATION

Quantisation involves representing the sampled data by a finite number of levels based on some criteria such as minimisation of quantiser distortion. Quantiser design includes input decision level, output representation level and the number of levels. Quantisers can be broadly classified into two types, namely, (i) scalar quantisers and (ii) vector quantisers. The classification of quantisers is shown in Fig. 1.12.

Examples of uniform scalar quantisers are *midtread* and *midrise quantisers*. An example of a non-uniform scalar quantiser is the *Lloyd–Max quantiser*. A detailed description of scalar and vector quantisers is given in the chapter on image compression.

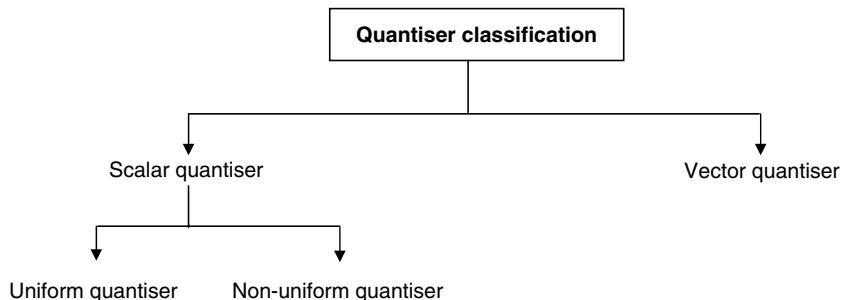


Fig. 1.12 Classification of quantiser

1.4 RESOLUTION

Resolution gives the degree of distinguishable details. Resolution can be broadly classified into (i) spatial resolution, and (ii) gray-level resolution.

(i) **Spatial Resolution** Spatial resolution is the smallest discernible detail in an image. Spatial resolution depends on the number of pixels. The principal factor determining spatial resolution is sampling.

(ii) **Gray-level Resolution** Gray-level resolution refers to the smallest discernible change in the gray level. Gray-level resolution depends on the number of gray levels.

The use of insufficient number of gray levels in smooth areas of the digital image is termed *false contouring*. The MATLAB code that illustrates the concept of false contouring is shown in Fig. 1.13 and the corresponding results are shown in Fig. 1.14.

```

%This program illustrates false contouring
clc
clear all
close all
a=imread('tigerpub.jpg');
imshow(a),title('original image')
%using 128 gray levels
figure,imshow(grayslice(a,128),gray(128)),
title('Image with 128 gray level')
%using 64 gray levels
figure,imshow(grayslice(a,64),gray(64)),
title('Image with 64 gray level')
%using 32 gray levels
figure,imshow(grayslice(a,32),gray(32)),
title('Image with 32 gray level')
%using 16 gray levels
figure,imshow(grayslice(a,16),gray(16)),
title('Image with 16 gray level')
%using 8 gray levels
figure,imshow(grayslice(a,8),gray(8)),
title('Image with 8 gray level')
  
```

Fig. 1.13 False contouring MATLAB code

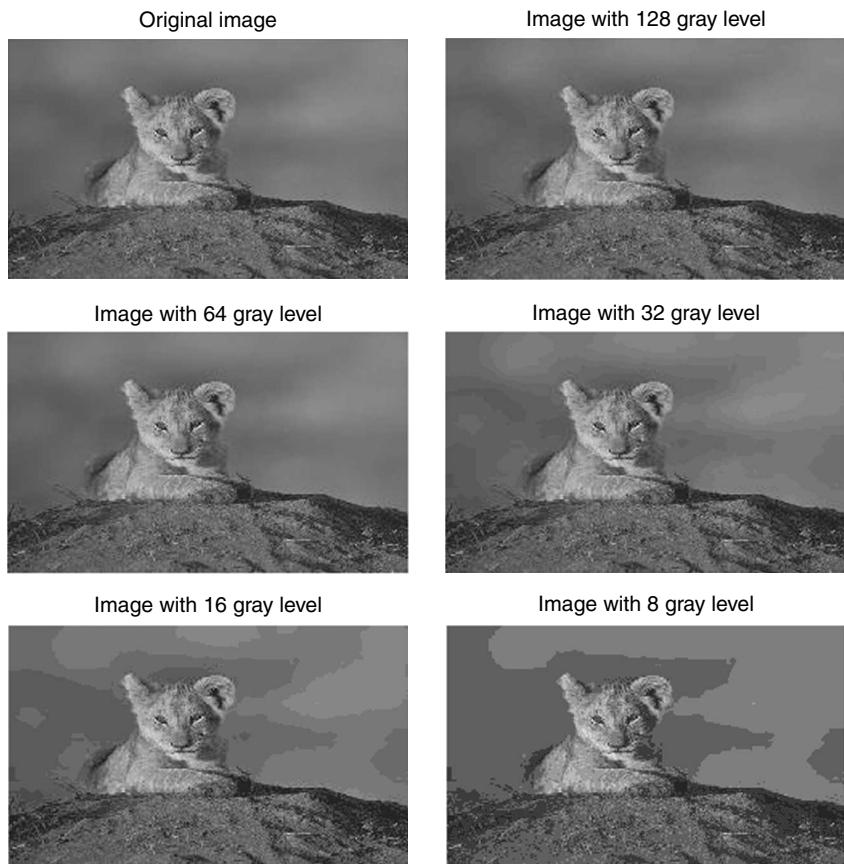


Fig. 1.14 Illustration of false contouring

1.5 HUMAN VISUAL SYSTEM

The Human Visual System (HVS) is one of the most complex systems in existence. Our visual system allows us to organise and understand the many complex elements in our environment. The visual system consists of an eye that transforms light into neural signals, and the related parts of the brain that process the neural signals and extract necessary information. The human eye serves to project and convert light into neural activity. Light enters the cornea, passes through the aqueous humor, then through the lens into the vitreous humor, and finally onto the photoreceptors located at the back of the retina. The ciliary muscles are responsible for accommodating the lens so as to focus the light rays onto the fovea, the region of the retina containing the greatest density of cones, and thus the high acuity for spatial and colour vision.

1.5.1 Anatomy of the Human Visual System

The human eye is a slightly asymmetrical sphere with an approximate sagittal diameter or length of 24 to 25 mm and a transverse diameter of 24 mm. It has a volume of about 6.5 cc. The view of a human eye is shown in Fig. 1.15.

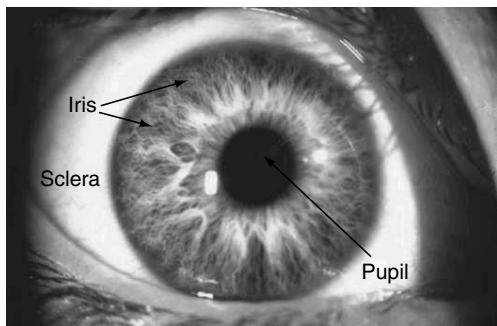


Fig. 1.15 View of the human eye

A black-looking aperture, the *pupil* allows light to enter the eye (it appears dark because of the absorbing pigments in the retina). The pupil of the eye contracts when exposed to bright light. The effect is to reduce the amount of light that falls on the retina. However, as time passes, the retina adapts to the new level and the pupil returns to its original size. The pupil can control the amount of light entering by about a factor of 30. A coloured circular muscle, the *iris*, which is beautifully pigmented, gives us our eye colour (the central aperture of the iris is the pupil). This circular muscle controls the size of the pupil so that more or less light, depending on conditions, is allowed to enter the eye.

Eye colour, or more correctly, iris colour is due to variable amounts of eumelanin (brown/black melanins) and pheomelanin (red/yellow melanins) produced by melanocytes. More of the former is present in brown-eyed people and of the latter in blue- and green-eyed people. The horizontal section of the human eye is shown in Fig. 1.16.

A transparent external surface, the **cornea**, covers both the pupil and the iris. This is the first and most powerful lens of the optical system of the eye and allows, together with the **crystalline lens** the production of a sharp image at the retinal photoreceptor level. The cornea and lens act together like a camera lens to focus an image on the retina at the back of the eye, which acts like the film.

The 'white of the eye', the **sclera**, forms part of the supporting wall of the eyeball. The sclera is continuous with the cornea. The sclera is a nearly spherical shell with a radius of 11 mm and is 1-mm thick. At the front of the eye, the sclera merges into the transparent *cornea*.

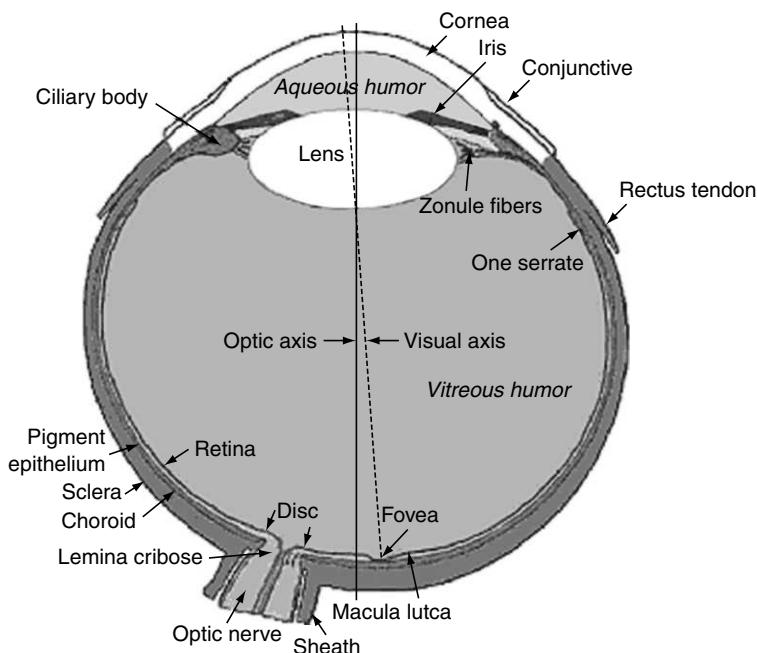


Fig. 1.16 Horizontal section of the human eye

The cross-sectional view of the eye shows three different layers. They are (i) the external layer formed by the sclera and cornea, (ii) the intermediate layer, divided into two parts—anterior (iris and ciliary body) and posterior choroids, (iii) the internal layer or the sensory part of the eye, the retina.

There are three chambers of fluid—anterior chamber (between cornea and iris), posterior chamber (between iris, zonule fibers and lens) and the vitreous chamber (between the lens and the retina). The first two chambers are filled with aqueous humor whereas the vitreous chamber is filled with a more viscous fluid, the vitreous humor.

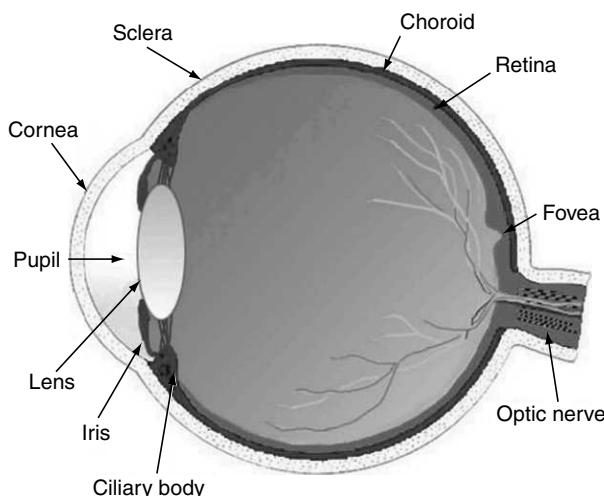


Fig. 1.17 Vertical section of the human eye

the visual axis is the optic axis projecting closer to the optic nerve head. The optic axis is the longest sagittal distance between the front or vertex of the cornea and the furthest posterior part of the eyeball. It is about the optic axis that the eye is rotated by the eye muscles. The neural signals from the photoreceptors are processed in the retina by other cell types, and also passed directly over the optic nerve, together with the outputs from the other retinal cell types, to the brain.

The sagittal section of the eye also reveals the *lens* which is a transparent body located behind the iris. The lens is suspended by ligaments (called *zonule fibers*) attached to the anterior portion of the ciliary body. The contraction or relaxation of these ligaments as a consequence of ciliary muscle actions, changes the shape of the lens, a process called accommodation, that allows us to form a sharp image on the retina. The vertical section of the human eye is shown in Fig. 1.17.

Light rays are focussed through the transparent cornea and lens upon the retina. The central point for image focus (the visual axis) in the human retina is the fovea. Here, a maximally focussed image initiates resolution of the finest detail and direct transmission of that detail to the brain for the higher operations needed for perception. Slightly more nasally situated than

1.5.2 Photopic and Scotopic Vision

The rods are sensitive to very low illumination and are responsible for *scotopic vision*. The order of minimum detectable luminance is about 1 nL. The cones, which are very tightly packed in the fovea, lie in line with the visual axis and are responsible for the most acute vision, *photopic vision*. The minimum sensitivity of cones is of the order of a microlambert. Rods are used to see at night or under very low illumination. Colour vision, also known as photopic vision, is provided by the cones, of which there are three distinct classes, each containing a different photosensitive pigment. The three pigments have maximum absorptions at 430, 530 and 560 nm and the cones are often called blue, green and red. The cones provide colour vision that can distinguish fine wavelength changes.

1.5.3 Brightness and Contrast

Brightness is the psychological concept or sensation associated with the amount of light stimulus. Light source intensity depends upon the total light emitted and the size of the solid angle from which

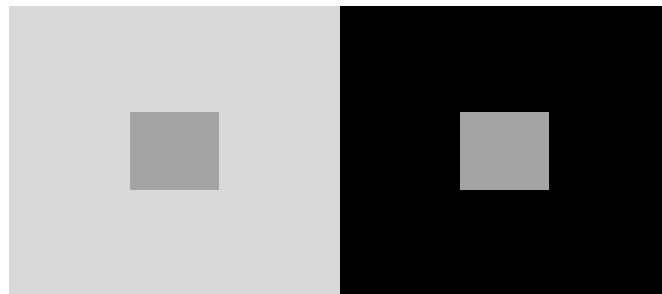


Fig. 1.18 Illustration of simultaneous contrast

it is emitted. Two sources of equal intensity do not appear equally bright. Luminance, the intensity per unit area, is a psychophysical property that can be measured.

The term *contrast* is used to emphasise the difference in luminance of objects. The perceived brightness of a surface depends upon the local background which is illustrated in Fig. 1.18. In Fig. 1.18, the small square on the right-hand side appears brighter when compared to the brightness of the square on the left-hand side, even though the gray level of both the squares are the same. This phenomenon is termed '*simultaneous contrast*'.

Colours tend to look darker and smaller against white, and lighter and larger against black. This fact is illustrated in Fig. 1.19 (Plate 1). The apparent changes in size arise from the flare in the eye's optics, which causes light from the bright areas to be scattered into the neighbouring dark areas. It is to be noted that simultaneous contrast can make the same colours look different. Also, it can make different colours look the same.

1.5.4 Laws of Physics Associated with Vision

The important laws associated with vision are given below.

Weber's Law First, let us define the term Just Noticeable Difference (JND). Just noticeable difference is the magnitude of the smallest stimulus that is perceivable. For luminance ' L ', the just noticeable difference ΔL when making the experiment of having a uniform background with intensity L and a spot intensity $L + \Delta L$ can be written as

$$\Delta L = KL \quad (1.44)$$

where K is a constant and Eq. (1.44) is known as Weber's law. Weber's law holds over a large range of intensity magnitudes. The visual system is not linear but logarithmic in amplitude sensitivity. The amount of light coming from a surface is the product of the illuminance and the surface reflectance.

Steven's Law Steven's law models how different types of stimuli are perceived by humans. If L is the physical stimulus and I the perceived sensation then the mathematical form of Steven's law is given by

$$I = \alpha L^n \quad (1.45)$$

For a point with intensity L against a black background, the value of n is 0.5.

Steven's law is taken into consideration when digitising image samples by introducing a non-linearity prior to quantisation which is given by

$$I = cL^{\frac{1}{n}} \quad (1.46)$$

where γ is typically slightly over 2. Eq. (1.46) describes transformation of luminance into intensity I . The process of using Eq. (1.46) at the input of the image-processing system and the inverse at the output side is known as *gamma correction*.

The energy of the spot is important for detection. This is given by Bloch's law and Ricco's law. If the area of the spot is A and the duration of the stimulus is T then the energy deviation ΔE is given by

$$\Delta E = A \times T \times \Delta L \quad (1.47)$$

Bloch's Law Bloch's law is given by

$$T \times \Delta L = C \quad (1.48)$$

where C is a constant which is valid for a duration time shorter than about 0.1 second.

Ricco's Law Ricco's law is given by

$$A \times \Delta L = C' \quad (1.49)$$

where C' is a constant, which holds good for stimulus sizes with a diameter of less than about 10 minutes of an arc in the visual angle.

Piper's Law The mathematical form of Piper's law is given by

$$\sqrt{A} \times \Delta L = C'' \quad (1.50)$$

where C'' is a constant which is in accordance with experiments for stimulus sizes between 10 minutes of an arc and 24 degrees of an arc.

1.5.5 Machband Effect

The Machband describes an effect where the human brain subconsciously increases the contrast between two surfaces with different luminance. The Machband effect is illustrated in Fig. 1.20. The intensity is uniform over the width of the bar. However, the visual appearance is that each strip is darker at its left side than its right. The spatial interaction of luminance from an object and its surrounding creates the Machband effect, which shows that brightness is not a monotonic function of luminance.

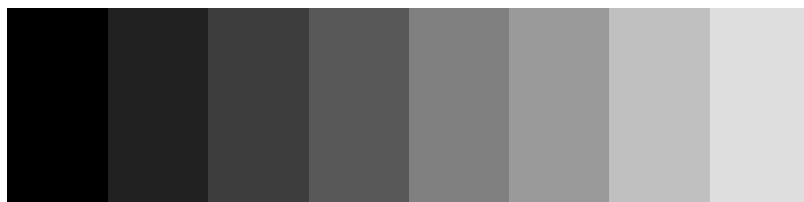


Fig. 1.20 *Machband effect*

Machbanding is caused by lateral inhibition of the receptors in the eye. As the receptors receive light, they draw light-sensitive chemical compounds from adjacent regions, thus inhibiting the response of receptors in those regions. Receptors directly on the lighter side of the boundary can pull in unused chemicals from the darker side, and thus produce a stronger response. Receptors on the darker side of the boundary, however, produce a weaker effect because of that same migration.

The human eye senses the visible spectrum using the combination of rods and cone sensors in the retina. Rod cells are better for low light vision, but can only sense the intensity of light, whereas the cone cells are sensitive to colour vision. Three types of cone cells exist in the human eye, each being more sensitive to either short or medium or long wavelengths of light. The set of signals at all the three cone cells describes the range of colours one sees with the eyes.

1.6 CLASSIFICATION OF DIGITAL IMAGES

Digital images can be broadly classified into two types and they are (i) raster image, and (ii) vector image.

1.6.1 Raster Image or Bitmap Image

A raster image file is generally defined as a rectangular array of regularly sampled values known as pixels. Scanned graphics and web graphics are the most common forms of raster images. Raster images are mapped to grids which are not easily scalable. A raster image is resolution dependent because it contains a fixed number of pixels that are used to create the image. Since there are a fixed and limited number of pixels, a raster image will lose its quality if it is enlarged beyond that number of pixels as the computer will have to 'make up' for the missing information. This fact is illustrated in Fig. 1.21 which shows the portion of the raster image being zoomed by a factor of 3 and 24. When the image is zoomed by a factor of 24, the clarity is lost. Bitmaps are used for photorealistic images, and therefore, involve complex colour variations. Raster images can show well the gradations of colour and detailed images such as photographs. Also, they can be acquired by optical scanners, digital CCD cameras and other raster-imaging devices. The spatial resolution of a raster image is determined by the resolution of the acquisition device and the quality of the original data source.



Fig. 1.21 Zooming of a raster image

Common raster image formats include BMP (Windows Bitmap), PCX (Paintbrush), TIFF (Tag Interleave Format), JPEG (Joint Photographic Experts Group), GIF (Graphics Interchange Format), PNG (Portable Network Graphics), PSD (Adobe Photoshop) and CPT (Corel PhotoPaint).

1.6.2 Vector Image

A vector image is defined by objects which are made of lines and curves that are mathematically defined in the computer. A vector can have various attributes such as line thickness, length and colour. Vector images are mathematically defined and hence, they are easily scalable. This implies that vectors can be printed at

any size, on any output device, at any resolution, without losing the detail and without altering the resolution of the image. Fig. 1.22 shows the zooming of a vector image. The vector image is zoomed by a factor of three and twenty-four. From the figure, it is evident that vector images can be scaled by several factors without altering the resolution of the image. Vector images are thus suitable for typography, line art and illustrations.

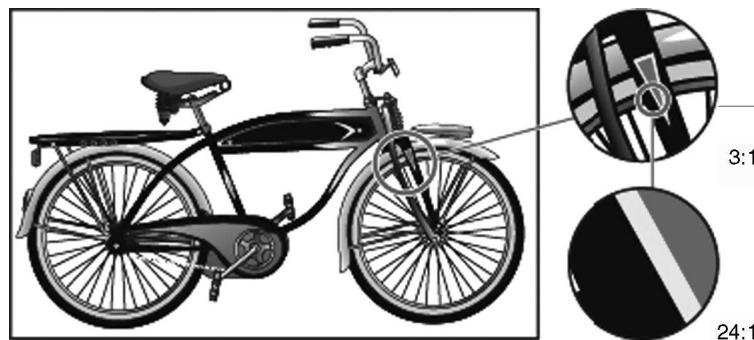


Fig. 1.22 *Zooming of a vector image*

1.7 IMAGE TYPES

Images can be broadly classified under four categories: (i) Black and white or binary images, (ii) grayscale images, (iii) colour images, and (iv) multispectral images.

1.7.1 Binary Images

Binary images take only two values, i.e., either '0' or '1'. The brightness graduation cannot be differentiated in the binary image. The binary image representation is illustrated in Fig. 1.23.

Mother Teresa's image in Fig. 1.24 is an example of a black-and-white image. A grayscale image can be converted to a black-and-white or binary image by the thresholding operation. The detailed description of the thresholding operation is given in Chapter 5. Geometric properties of an object, like the location of the centroid of the object or the orientation of the object, can be easily extracted from a binary image.

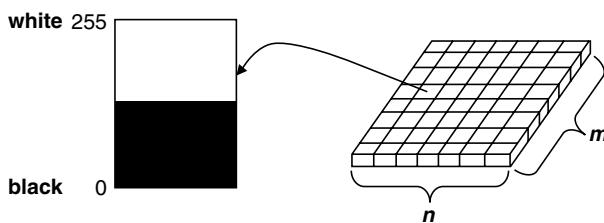


Fig. 1.23 *Binary image representation*

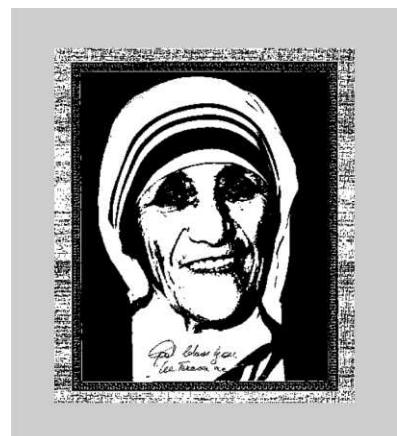


Fig. 1.24 *Mother Teresa in black and white*

1.7.2 Grayscale Images

Grayscale images contain only brightness information. Each pixel value in a grayscale image corresponds to an amount or quantity of light. The brightness graduation can be differentiated in a grayscale image. In a grayscale image, each pixel is represented by a byte or word, the value of which represents the light intensity at that point in the image. An 8-bit image will have a brightness variation from 0 to 255 where '0' represents black and '255' represents white, as shown in Fig. 1.25. A grayscale image measures only the light intensity. Each pixel is a scalar proportional to the brightness. Mother Teresa's image is given as an example of a grayscale image in Fig. 1.26.

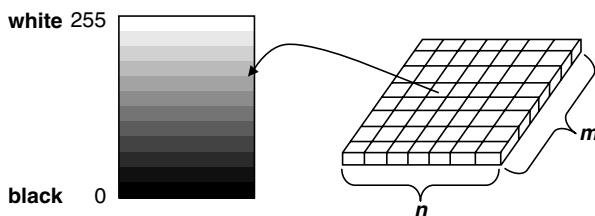


Fig. 1.25 Grayscale image representation



Fig. 1.26 Mother Teresa in grayscale

1.7.3 Colour Image

A colour image has three values per pixel and they measure the intensity and chrominance of light. Each pixel is a vector of colour components. Colour images can be modeled as three-band monochrome image data, where each band of data corresponds to a different colour. The actual information stored in the digital image data is the brightness information in each spectral band. Common colour spaces are RGB (Red, Green and Blue), HSV (Hue, Saturation, Value), and CMYK (Cyan, Magenta, Yellow, Black). Colours can be represented with three colour components as shown in Fig. 1.27 (Plate 1), so the typical uncompressed data rate of a colour image is three times the data rate of a grayscale image.

Kalpana Chawla's image in Fig. 1.28 (Plate 1) is an example of an RGB image. Colour images usually consist of three separate image representations called colour planes, with pixel values in each plane corresponding to the intensity of a certain colour at a specific point in the image. Each colour plane consists of an array of pixel values similar to that of the grayscale representation. The most popular system is RGB, where three colour planes are used to represent the intensity of red, green and blue in the scene.

1.7.4 Volume Image

A three-dimensional image is an example of volume image. The volume image can be obtained from some medical imaging equipment in which the individual data points are called 'voxels'. Voxels stands for volume pixels. A CAT scan is an example of a volume image.

1.7.5 Range Image

Range images are a special class of digital images. Each pixel of a range image expresses the distance between a known reference frame and a visible point in the screen. The range image reproduces the 3D structure of a scene. Range images are also referred as depth images.

1.7.6 Multispectral Image

Multispectral images are images of the same object taken in different bands of visible or infrared regions of the electromagnetic spectrum. Images acquired for remote-sensing applications are generally multi-spectral in nature. Multispectral images typically contain information outside the normal human perceptual range. This includes infrared, ultraviolet and other bands in the electromagnetic spectrum. The information available in a multispectral image is not visible to a human observer. However, the information is often represented in visual form by mapping the different spectral bands to RGB components.

A hyper-spectral image is a set of 224 images, which are referred as bands, at a specific location on the earth's surface. These hyper-spectral images are taken from a Moderate Resolution Imaging Spectroradiometer through Terra and Aqua satellites. They are used by scientists on the earth to study and analyse dynamics and processes occurring in the earth's surface. All the 224 bands are pictures at the same location.

1.8 ELEMENTS OF AN IMAGE-PROCESSING SYSTEM

The different elements in an image-processing system are (i) image-acquisition element which involves image sensors like CCD sensors, CMOS sensors, and image scanners, (ii) image-storage devices, (iii) image-processing elements, and (iv) image-display devices.

1.8.1 Image Sensor and Acquisition

The term *image acquisition* refers to the process of capturing real-world images and storing them into a computer. Conventional silver-based photographs in the form of negatives, transparencies or prints can be scanned using a variety of scanning devices. Digital cameras which capture images directly in digital form are more popular nowadays. Films are not used in digital cameras. Instead, they use a charge-coupled device or CMOS device as the image sensor that converts light into electrical charges.

An image sensor is a 2D array of light-sensitive elements that convert photons to electrons. Most of the digital cameras use either a CCD or a CMOS image sensor. A typical solid-state image sensor consists of many discrete photo-sensing elements, some form of charge-transport mechanism, and an output circuit. The photosensitive sites convert the incoming photons into electrical charges and integrate these charges into a charge packet. The charge packet is then transferred through the transport mechanism to the output circuit where it is converted into a measurable voltage. The types of photo-sensing elements used in solid-state imagers include photodiodes, MOS capacitors, Schottky-barrier diodes and photoconductive layers. The output circuit typically consists of a floating diffusion and source-follower amplifier. In practical applications, image sensors are configured in a one-dimensional (linear devices) or a two-dimensional (area devices) manner.

Image-Sensor Terminology

Some of the definitions of the most commonly used terms in solid-state image sensors are given below:

- (i) **Charge-coupled Device (CCD)** CCD is a charge-transfer device that collects light in pixels and then uses clock pulses to shift the charge along a chain of pixels.
- (ii) **Dark Current** The charge of the signal collected by the pixel in the absence of light is termed dark current.
- (iii) **Photo Site** Photo site is the portion of the silicon that functions as a light-sensitive area.
- (iv) **Pixel** Pixel is a discrete photosensitive cell that collects and holds a photo charge.

(v) **Fill Factor** A pixel is divided into a sensing portion and a read-out portion. Fill factor is the ratio of the sensing area to the total area.

(vi) **Quantum Efficiency** Quantum efficiency is the ratio of the photon-generated electrons that the pixel captures to the photons incident on the pixel area.

1.8.2 CCD Sensor

Charge-coupled devices (CCDs) find wide applications in almost all digital image-acquisition devices. Invented in the late 1960s by researchers at Bell Labs, these were initially conceived as a new type of computer memory circuit, and were demonstrated in 1970 for that facility. It soon became apparent that CCDs have many other potential applications, including that of signal processing and imaging, the latter because of silicon's light sensitivity that responds to wavelengths less than $1.1 \mu\text{m}$. (The visible spectrum falls between $0.4 \mu\text{m}$ and $0.7 \mu\text{m}$.) The CCDs early promise as a memory element has since disappeared, but its superb ability to detect light has turned the CCD into a premier image sensor. Like the integrated circuits (ICs), CCDs begin on thin silicon wafers, which are processed in a series of elaborate steps that define various functions within the circuit. Each wafer contains several identical devices (chips), each capable of yielding a functional device. From the wafer, a few chips are selected based on applications. The selected chips, based on a variety of preliminary screening tests, are then cut from the wafer and packaged into a carrier for use in a system.

Operation of CCD Sensor

The charge-coupled device (CCD) is basically a series of closely spaced MOS capacitors. CCD imaging is performed in three steps:

Step 1: Exposure In this step, the sensor is exposed to the incident light. Upon exposure, light is converted into an electronic charge at discrete sites called pixels.

Step 2: Charge transfer Charge transfer moves the packets of charge within the silicon substrate.

Step 3: Charge-to-voltage conversion and output amplification

Converting Light (Photons) into Electronic Charge An image is acquired when incident light, in the form of photons falls on the array of pixels. The energy associated with each photon is absorbed by the silicon and causes a reaction to occur. This reaction yields an electron-hole charge pair. Fig. 1.29 illustrates the photon interaction with silicon. The number of electrons collected at each pixel is linearly dependent on the light level and exposure time and non-linearly dependent on the wavelength of the incident light.

Many factors can affect the ability to detect a photon. Thin films of materials intentionally grown and deposited on the surface of the silicon during fabrication might have a tendency to absorb or reflect light. Photons are absorbed at different depths in the silicon depending on their wavelengths. There are instances during which photon-induced electrons cannot be detected because of the location within the silicon where they were created.

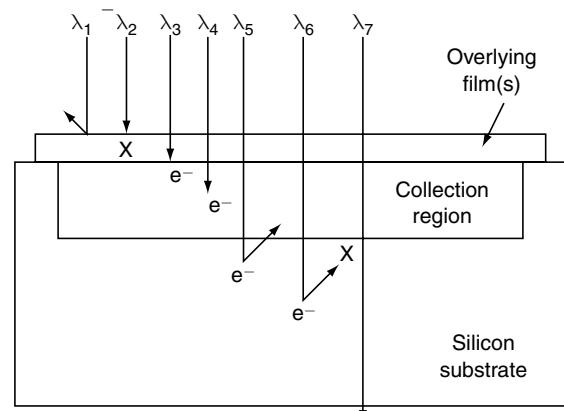


Fig. 1.29 Photon interaction with silicon

Potential Wells and Barrier CCDs follows the basic physics of basic metal-oxide semiconductor (MOS) devices. A CCD MOS structure simply consists of a vertically stacked conductive material (doped polysilicon) overlying a semiconductor (silicon), which is separated by a highly insulating material (silicon dioxide).

By applying a voltage potential to the polysilicon or ‘gate’ electrode, the electrostatic potentials within the silicon can be changed. With an appropriate voltage, a potential ‘well’ can be formed which is capable of collecting the localised electrons created by the incident light. Fig. 1.30 shows the potential well and barrier. The electrons can be confined under this gate by forming zones of higher potentials called barriers surrounding the well.

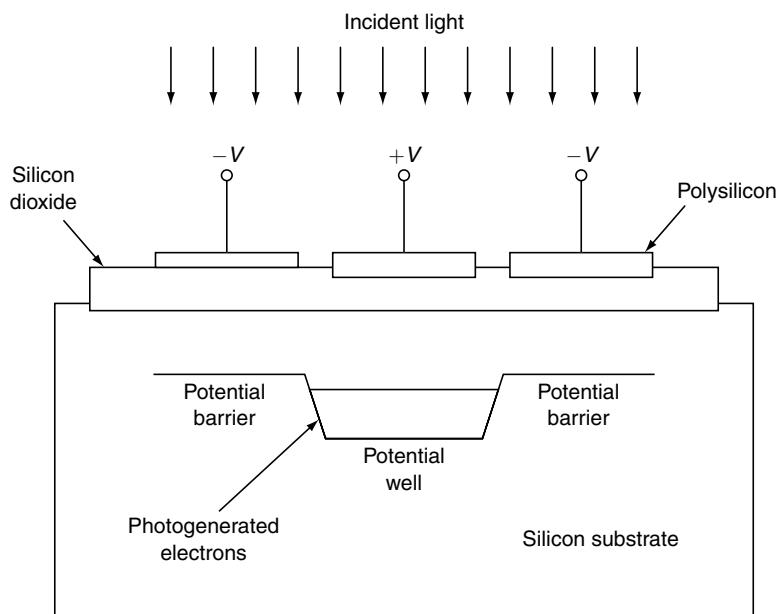


Fig. 1.30 Potential well and barrier

Depending on the voltage, each gate can be biased to form a potential well or a barrier to the integrated charge.

Charge Transfer Once charge has been integrated and held locally by the bounds of the pixel architecture, one should have a means of getting that charge to the sense amplifier that is physically separated from the pixels.

1.8.3 Four-Phase CCD

A four-phase CCD structure is illustrated in Fig. 1.31. Application of bias voltage to the gate of the MOS capacitor results in the creation of a localised potential well in the semiconductor. The photo-generated charges are collected and stored in the potential well.

By varying the gate voltages in a systematic and sequential manner, the charge packet can be transported from under one electrode to the adjacent electrode. This provides a simple mechanism for moving a charge through the CCD shift register. A CCD array consists of a series of column registers. The charge packets are confined within each row or column by channel stops. At the end of each column there is a horizontal shift

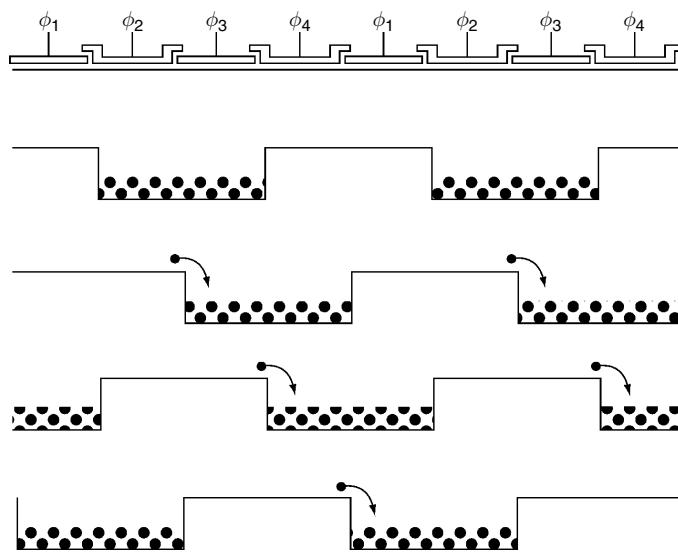


Fig. 1.31 Four-phase CCD

register. The charged packets are shifted, one line at a time, into the horizontal shift register. The entire line is then clocked out from the horizontal shift register to the output amplifier before the next line enters horizontal register.

One important requirement for a good CCD operation is to maintain high charge transfer efficiency (CTE) in the CCD shift register. CTE is a measure of the percentage of electrons that are successfully transferred from under one electrode to under the adjacent electrode. Typically, CTE needs to be no less than 0.99999. Limited time available for charge transfer during high-speed operation, and potential obstacles (barriers and wells) that arise from the device design and processing, can cause an incomplete charge transport and reduction of CTE.

1.8.4 CCD Formats

Image sensing can be performed using three basic techniques—point scanning, line scanning and area scanning.

Point Scanning Point scanning uses a single-cell detector or pixel (picture element). An image can be scanned by sequentially detecting scene information at discrete X , Y coordinates as illustrated in Fig. 1.32. Some of the advantages of this approach include high resolution, uniformity of measurement from one site to another and the simplicity of the detector.

Disadvantages of the point-scanning mechanism include registration errors due to the X - Y movement of the scene or detector, frame-scanning rates because of the repeated number of exposures and system complexity due to the X - Y movement.

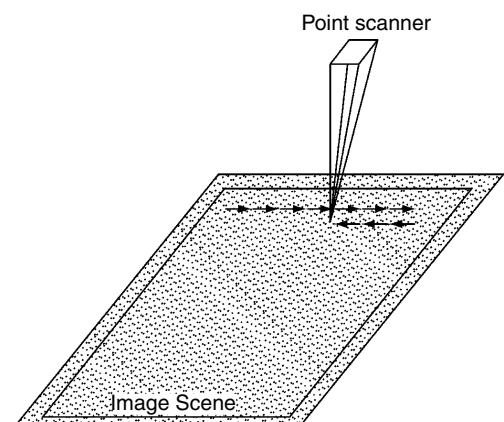


Fig. 1.32 Point scanning

Line Scanning In line scanning, an array of single-cell detectors can be placed along a single axis such that scanning thereupon takes place in only one direction. In this case, a line of information from the scene is captured and subsequently read out of the device before stepping to the next line index which is shown in Fig. 1.33.

The physical length of a linear CCD scanner is limited only by the size of the starting silicon wafer used to fabricate the device. This limitation can be overcome at times by mounting several linear CCDs end to end to increase the overall length.

Line scanning greatly improves the scan time over point scanning. Other benefits include reasonably high resolution and less complicated scanning mechanics. However, the pixel spacing and size in one direction limit the resolution. The measurement accuracy at each pixel has finite non-uniformities that must be occasionally factored out with the system. Scan times, of the order of several seconds or minutes, are still unsuitable for many applications and the costs of linear CCDs are considerably higher than single-cell detectors. The finite number of CCD chips on each silicon wafer and the resulting yield loss dictate the costs from processing variations.

Area Scanning A two-dimensional array of detectors can be created such that the entire image can be captured with a single exposure, thus eliminating the need for any movement by the detector or scene. The area-scanning mechanism is illustrated in Fig. 1.34. Area scanners are capable of producing the highest frame rates with the greatest amount of registration accuracy between pixels. Also, the system complexities are kept to a minimum.

However, resolution is now limited in two directions. Other disadvantages include generally lower signal-to-noise and higher cost arising due to the fact that fewer devices can be placed on a wafer with the yield being inherently lower for a number of reasons.

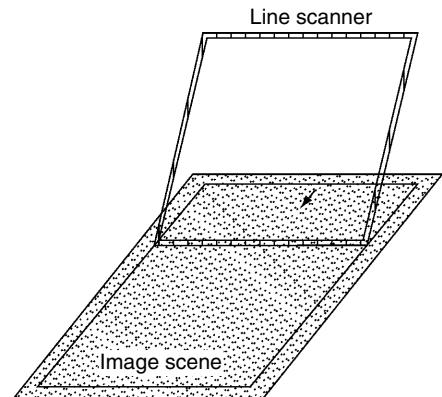


Fig. 1.33 Line scanning

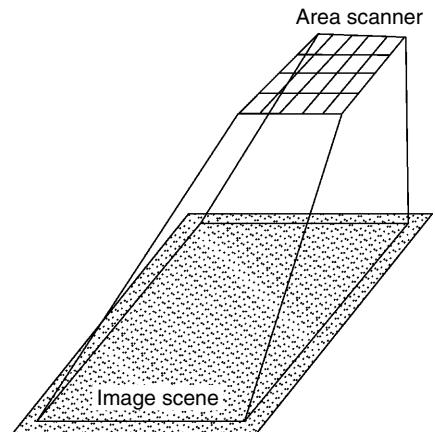


Fig. 1.34 Area scanning

1.8.5 The Architecture of CCD

There are two different types of CCD architecture and they are (i) full-frame CCD, and (ii) frame-transfer CCD.

Full-frame CCD Full-frame CCDs have the simplest architecture and are the easiest to fabricate and operate. These consist of a parallel shift register, a serial shift register and a signal-sensing output amplifier as shown in Fig. 1.35.

Images are optically projected onto the parallel array that acts as the image plane. The device takes the scene information and partitions the image into discrete elements that are defined by the number of pixels, thus 'quantising' the scene. The resulting rows of scene information are then shifted in a parallel fashion to the serial register, which subsequently shifts the row of information to the output as a serial stream of data. The process repeats until all rows are transferred off the chip. The image is then reconstructed as dictated by the system.

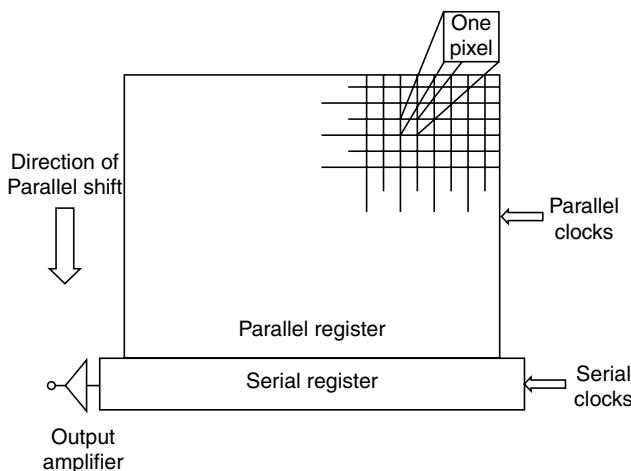


Fig. 1.35 Full frame CCD

Since the parallel register is used for both scene detection and read-out, a mechanical shutter or synchronised strobe illumination must be used to preserve the scene integrity. The simplicity of the FF design yields CCD images with qualities of highest resolution and highest density.

Frame-transfer CCD Frame-transfer CCDs are much like full-frame CCDs, except that they have an additional identical parallel register called a *storage array* which is not light-sensitive. The idea is to shift a captured scene from the photo-sensitive image array quickly to the storage array. Read-out off the chip from the storage register is then performed as described in the full-frame device, while the storage array is integrating with the next frame.

The advantage of this architecture is that it allows for faster frame updates due to a continuous or shutterless operation. The resulting performance is compromised, however, because integration is still occurring during the image dump on the storage array, thus resulting in an image ‘smear’. Since twice the silicon area is required to implement this architecture, frame transfer CCDs have lower resolution and cost much higher than full frame CCDs.

Advantage of a CCD Image Sensor The quality of the image obtained using a CCD image sensor is high due to optimised photo detectors. Also, CCD image sensors do not introduce noise or non-uniformity.

Disadvantage of CCD image Sensor CCD image sensor power requirement is high-due to high speed shifting clocks. Also, the frame rate is limited due to analog serial read-out. Yet another drawback of a CCD image sensor is its inability to integrate with other camera functions such as clock drivers and timing logic. Signal processing on the same chip is another drawback of a CCD image sensor.

1.8.6 CMOS Image Sensor

CMOS image sensors have been in existence since the late 1960s. The basic structure of a pixel consists of a light-sensitive element and a MOS transistor acting as a switch. The early design had a photodiode as the photo-sensing element. However, due to the issues of large fixed-pattern noise (FPN), scalability, and difficulty to obtain fast read-out, MOS sensors have lost out to CCDs as the sensor of choice. In the past few years, CMOS sensors have attracted a lot of attention from the imaging industry. The demand for low-cost, low-power, and compact imaging systems and the significant improvement in device scaling are the major contributing factors to the renewed interest in CMOS sensors.

Advantages of CMOS Image Sensors The advantages of CMOS image sensors are

- low-voltage operation and reduced power consumption
- the ability to integrate timing, control, and image processing circuitry onto the sensor chip
- random access of the image data that allows electronic windowing, pan and zoom

- reduced silicon processing cost, benefits from the compatible fabrication process, and the huge volumes in the generic CMOS industry

Drawback of CMOS Image Sensor One major problem associated with CMOS sensors is the relatively large fixed-pattern noise (FPN). Unlike CCD sensors, each pixel in a CMOS sensor has its own amplifier. Variations of amplifier gain and offset result in a static pattern noise in the background of the image.

Applications of CCD and CMOS Image Sensor CCD image sensors have been used in a variety of consumer, commercial, medical, scientific, and military applications. In the consumer market, the home-use CCD-based camcorder has been the dominant product for many years. Only in recent years have low-end digital electronic cameras started winning the acceptance of consumers. In the commercial markets, high-resolution digital still cameras are widely being used in photojournalism, catalog preparation, advertising, and studio photography for advantages which include good image quality, instant feedback, and ease of editing and management. Low-to-medium resolution electronic cameras have been used in insurance, real estate, law enforcement, security and surveillance. Traffic monitoring is an area of application that has been gaining a lot of interest. Scanning applications include document scanning, FAX, graphic arts, digital archiving of movie and artistic works, automated inspection, and sorting. Machine vision can be used in robotics manufacturing, and remote inspection of areas unsafe to humans. Ultra-high-frame rate cameras have been used in the automotive crash testing and inspection of high-speed moving processes. Medical imaging applications such as microscopy, endoscopy, mammography and x-ray dental detection have been moving towards the use of electronic imaging for better detection instant diagnostics, lower x-ray dose, and increased patient comfort. In scientific applications, high-resolution, low-noise, high-sensitivity and high-dynamic range CCD cameras are used in astronomical observatories, spectroscopic instruments, and space-based systems such as the Hubble Space Telescope. Recently, images of Mars captured by the CCD cameras on the Mars Rover have created worldwide interest on the space programme. Satellite and remote sensing provide monitoring of earth resources, weather tracking, military detection, and identification in the military applications. CCDs have the advantages of small size and ruggedness. They are used for night vision, target identification, and mapping.

With the advantages of low-power consumption, high level of integration and low production cost, CMOS sensors are ideal for multimedia applications, application-specific imaging systems and low-cost cameras. Applications such as PC-peripheral digital camera for input to web pages and video conferencing, interactive games, security monitoring, toys, video phone, automobile sensor, fax, and document scanning will see increased adoption of the CMOS imagers as most of them become commercially available. With the improved performance of CMOS imagers, more sophisticated and high-resolution imaging applications will be possible.

Comparison of CCD and CMOS Image Sensors

The parameters like power consumption, system integration, quality of the image, read-out mode, fixed pattern noise, dynamic range, fill factor and cost are taken into account to compare CCDs with CMOS image sensors.

(i) Power Consumption CMOS sensors consume less power than similar CCD sensors. This advantage is important for applications such as digital cameras and cellular phones. CCD systems require multiple 5–15 V power supply levels and voltage regulators for operation. A CMOS sensor uses a single 5-V supply.

(ii) System Integration With CMOS, signal processing can be integrated directly on the chip. On-chip analog-to-digital converter facilitates driving of high-speed signals. Also, the digital output is less sensitive to pickup and crosstalk, and more convenient for a system designer to use, as it facilitates computer and

digital-controller interfacing while increasing system robustness. It is possible to perform functions such as Automatic Gain Control (AGC), colour balance, and automatic exposure and focus control on a single chip.

(iii) *Image Quality* CCD creates high-quality, low-noise images, while CMOS image sensors are more susceptible to noise.

(iv) *Read-out mode* CMOS image sensors allow various modes of read-out like windowed read-out, scanning read-out and accelerated read-out. On the other hand, CCDs perform read-out by transferring the charge from pixel to pixel that requires the entire array of pixels.

(v) *Fixed-pattern Noise* In CMOS image sensors, fixed-pattern noise results from mismatches between threshold voltages of the transistor and parasitic gate-source, gate-drain capacitances. CCDs are not affected by fixed-pattern noise from the mismatch of transistors. However, they experience fixed-pattern noise resulting from the capacitance between clock lines and output lines. The noise resulting from this can be filtered using a low-pass filter.

(vi) *Dynamic Range* The dynamic range of CCDs is larger than that of CMOS pixels because of lower dark current and higher sensitivity of CCDs. The lower dark current in CCDs is achieved by employing surface-state pinning.

(vii) *Fill Factor* CMOS-based pixels typically have a 20 to 30 per cent fill factor while that of CCDs is more than 80 per cent. This is mostly due to integration of circuitry into the pixel area in CMOS-based pixels. To counter the low fill factor, the CMOS-based pixels can use micro lenses to improve fill factor.

(viii) *Cost* Higher yields and less susceptibility to defects give CMOS sensor an edge over CCD image sensors. Also, the cost of a CMOS sensor is less when compared to a CCD image sensor.

1.8.7 Scanning Mechanism

The purpose of a scanner is to convert light into ‘0’s and ‘1’s. Scanners convert analog images into digital image files. A scanner allows one to capture documents that are printed on paper and turn them into digital or online format to be viewed on a computer. For desktop scanners, speed is of greater importance than colourimetric accuracy, and therefore they usually employ three linear CCD sensors with red, green and blue colour filters.

Components of a scanner The components of a scanner include (i) an optical system, (ii) a light sensor, (iii) an interface, and (iv) driver software.

The quality of a scan is primarily dependent on the quality of the optics and the light detector, but the productivity is mainly dependent on the interface and driver.

Drum Scanner Drum scanners are mainly used for reproduction in printing offices and in advertising agencies. Drum scanners are used to obtain high-quality scans for professional purposes. The quality of the scan obtained using a drum scanner is better than that obtained from a flatbed scanner. A drum scanner consists of a cylindrical drum made of translucent plastic-like material. The image to be scanned, usually a film, is wet-mounted by means of a fluid on the drum. The fluid is either oil-based or alcohol-based. The commonly used sensing element in a drum scanner is a photo-multiplier tube. The photo-multiplier tube is much more sensitive to light than a CCD.

Flying-spot Scanner Flying-spot scanners are used for scanning still images, usually photographic slides (transparencies), or pictures from the cinema film. Conceptually, it is the simplest method which is capable of generating very high-quality pictures. However, it is complex in terms of implementation.

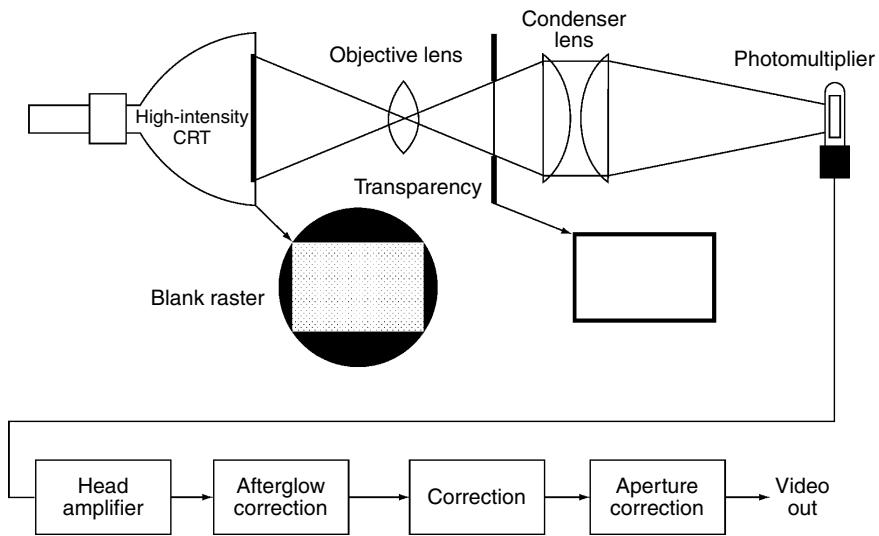


Fig. 1.36 Flying-spot scanner

The principle is that of modulation of a scanning light beam by the density of the photographic film. The special cathode ray tube (CRT) used for this purpose has an optically flat face-plate and a high-precision deflection and focus system to generate a blank raster (flying spot). An image of the raster is projected onto the film and the modulated light passing through the film/slide is collected by a condenser lens system and is focused on the target cathode of a photomultiplier tube as illustrated in Fig. 1.36.

Optically, the process is identical to that of a slide projector, except for a reversal of the direction of light flow; the light source in the projector is replaced by a light receptor, i.e., the photomultiplier, which functions as an amplifier.

A photomultiplier is, in fact, a cold electron tube in which photons falling onto a light-sensitive cathode cause emission of electrons. The electrons are accelerated to a second electrode, which is at a potential of several hundred volts with respect to the cathode, where their velocity is such as to dislodge more than one electron from each electron hitting it. This secondary emission ratio is greater than '1'. This process continues through a series of stages, as many as ten, before the final output current is delivered. The original electron emission from the photo-cathode is amplified by a factor of 10^4 without any noise.

The photomultiplier output current (1 to 2 mA) is converted into the standard video voltage level by the head amplifier. The three blocks following the head amplifier, as shown in Fig. 1.36, perform the analog signal processing. The stages involved include afterglow correction, gamma correction and aperture correction respectively.

1. Afterglow Correction The afterglow corrector processes the signal to negate the effect of non-zero afterglow from the CRT phosphor. Although the phosphor is chosen such that it has a very short afterglow (a time constant of less than 100 ns compared to 4.3 to 430 ms for direct-view CRTs), some residual effect is still felt.

Assuming that the spot has zero diameter (in practice, this is not possible as the electron density is more likely to be approximated by a Gaussian function), and that the impulse response of the phosphor decays with time, the effective spot profile is roughly a negative exponential. The visual effect can be clearly seen on a conventional oscilloscope at very low speeds, since these are generally fitted with long-persistence phosphors.

The effect of afterglow on a picture waveform can be worked out by convolving it with the spot profile, where it acts as a non-linear-phase (asymmetrical impulse response) low-pass filter. The afterglow corrector has to have a complementary high-boost frequency response, which is usually approximated by a series of CR differentiators possessing a wide range of time constants. The contributions from each time constant are adjusted experimentally for the best picture quality.

2. Gamma Correction Gamma correction is necessary in flying-spot scanners to compensate for the CRT display response as usual, since the photomultiplier is a linear device. However, the process may be considerably complicated by non-linearities in the film, which the broadcaster may wish to correct also (this is even more in colour systems). Still, other characteristics are needed when scanning the negative film—more than a simple signal inversion is required to get a correct positive image.

3. Aperture Correction The low-pass filtering effect of the smeared spot is removed in the afterglow corrector. Also, there has to be a different kind of high-frequency boost inserted because of the finite size of the scanning spot. This is a universal requirement in all electron-beam scanned image-generation devices.

Applications of Flying-spot Scanners Flying-spot scanning is used in video mappers employed in ground-control approach (GCA) radars and ground defense radars for displaying the map of the area over the plan position indicator CRT in synchronisation with the trace.

Flatbed Scanners Flatbed scanners are widely used in office environments and are most suitable for small-scale image printouts and printing matters. The core component of a flatbed scanner is the CCD array. The CCD is a collection of tiny light-sensitive diodes called photosites that convert photons (light) into electrons (electrical charge). The diodes are sensitive to light—the brighter the light that hits a single photosite, the greater the electrical charge accumulated at that site.

The image of the document that is to be scanned reaches the CCD array through a series of mirrors, filters and lenses. The exact configuration of these components depends on the model of the scanner, but the basics are the same.

- i. A *fluorescent lamp* or xenon lamp is used to illuminate the document. The entire mechanism (mirrors, lens, filter and CCD array) make up the scan head. The scan head is moved slowly across the document by a belt that is attached to a stepper motor. The scan head is attached to a stabiliser bar to ensure that there is no wobble or deviation as the scan head completes a single complete scan (called ‘pass’) of the document.
- ii. The image of the document is reflected by an angled *mirror* to another mirror. Some scanners use only two mirrors, while others use three mirrors. Each mirror is slightly curved in order to focus the image it reflects onto a smaller surface. The last mirror reflects the image onto a lens and the lens focuses the image through a filter on the CCD array.
- iii. The *filter* and lens arrangement varies depending on the scanner. Some scanners use a three-pass scanning method. Each pass uses a different-colour filter (red, green or blue) between the lens and CCD array. After the three passes are completed, the scanner software resembles the three filtered images into a single full-colour image.

Scanning the document is only part of the process. For the scanned image to be useful, it must be transferred to the computer. The connections commonly used for this purpose are the following:

1. *Parallel* Connecting through the parallel port is the slowest transfer method available.
2. *Small Computer System Interface (SCSI)* SCSI requires a special SCSI connection. Most SCSI scanners include a dedicated SCSI card to be inserted into the computer.

3. *Universal Serial Bus (USB)* USB scanners combine high speed, ease of use and affordability in a single package.

4. *FireWire* This is usually found on higher-end scanners. FireWire connections are faster than USB and SCSI. These are ideal for scanning high-resolution images.

Colour Scanning The CCD elements are sensitive to the brightness of light hence the pixels store only the brightness information of the original image. This brightness information is generally termed *luminance information*. To obtain the colour or chrominance information, three CCD elements for each pixel of the image is formed. The signal output from each of the red, green and blue components are combined to produce a colour scanned image.

Scan Quality The quality of a scanned image is determined by its resolution and colour depth. The scanner resolution is measured in dots per inch (dpi). The scanner resolution can be broadly classified into (i) optical resolution and (ii) interpolated resolution. The optical resolution refers to the actual number of sensor elements per inch on the scan head. Most flatbed scanners nowadays have optical resolutions of the order of 1600 dpi to 3200 dpi. The interpolated resolution involves an interpolation process for generating new pixel values. Interpolated resolution is not a true resolution because in the case of an interpolated resolution, an algorithm is used to compute the values in between dots. Interpolated images will always seem to be either too smooth or slightly out of focus. The colour depth indicates the total number of colours that can be represented in an image. It is usually 24 bits for general scanners but can have higher values for high-end scanners.

1.8.8 Photoconductive Cameras—The Vidicon

The vidicon is a storage-type camera tube in which a charge density pattern is formed by the image scene radiation on a photoconductive surface which is then scanned by a beam of low-velocity electrons. The fluctuating voltage coupled out to a video amplifier can be used to reproduce the scene being imaged.

Fig. 1.37 shows the structure of a vidicon camera tube. The photoconductive principle is nearly universal in electron-tube type video sources; the differences are variations in the photosensitive material. In vidicon, the material is antimony trisulphide, while in plumbicon it is an oxide of lead. Other target materials are arrays of diodes formed on a thin sheet of silicon.

In the vidicon, a thin layer of the target material is deposited on the rear of a conducting glass sheet behind an optically flat face plate. A metal ring is used to provide electrical connection.

Operation of the Vidicon In operation, the back surface of the target is scanned by a sharply focused low-velocity electron beam (with accelerating potential of 300 to 800 V), which ensures that the secondary emission ratio is less than unity. In the dark, the target is an insulator and electrons accumulate on it building a negative charge, until an equilibrium is reached and the surface potential is the same as that of the cathode; no additional electrons can land, and in fact electrons which do not land on the target retrace their paths, as they 'see' an equal and opposite accelerating field back to the cathode. This phenomenon is used in the so-called *return-beam vidicon* and the returning beam is noiselessly amplified using a built-in photomultiplier structure that surrounds the electron gun. It has very high sensitivity and resolution. The mesh electrode through which the beam strikes the target forms an electro-optical lens to correct the focusing errors at the edge of the picture introduced by the electromagnetic focus and deflection system. It is normally operated at a level slightly above the wall anode potential.

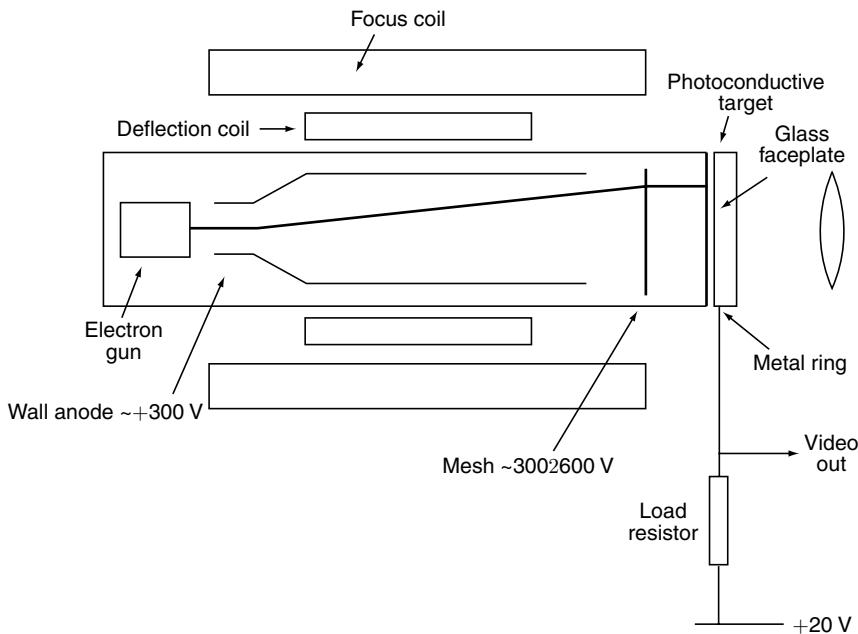


Fig. 1.37 Vidicon camera tube

When the target is illuminated, it becomes conductive and due to the small positive bias on the front surface (> 20 V), electrons travel through from the rear between passes of the electron beam thus allowing a positive charge to accumulate. The target can be regarded as a very large number of elemental capacitors bridged by light-dependent resistors (lateral conduction is negligible). Any accumulation of positive charge on the rear of the target, corresponding to a discharge of the capacitors by the light-dependent resistors, is made good during the next passage of the electron beam, which resets the potential approximately to that of the cathode. It is this recharging current (of the order of 100 nA) that constitutes the video signal, and it is negative-going for white picture content.

1.8.9 Analog Colour TV Camera

The working principle of an analog colour TV camera is analogous to that of a Vidicon camera. However, three cameras are employed in this case, one for each of the primary colours. The conceptual arrangement is shown in Fig. 1.38.

Two types of cameras are available in the market: SLR (Single lens reflex) cameras and point-and-shoot cameras. The difference between the two depends on how the photographer sees the scene via the view-finder.

In a point-and-shoot camera, the viewfinder is a simple window (hole) through the body of the camera. In an SLR camera, the actual image to be filmed can be seen. The camera has a slanted mirror positioned between the shutter and lens, with a piece of translucent glass and a prism arrangement placed above it as shown in Fig. 1.39. The prism flips the image on the screen, so that it appears right side up again, and then redirects it to the viewfinder window. The advantage of this design is that it is possible to adjust the focus and compose the scene to get the exact picture.

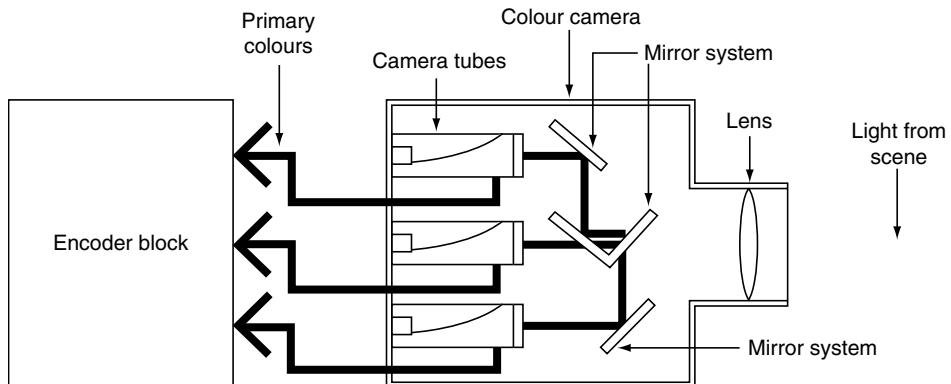


Fig. 1.38 *The colour TV camera system*

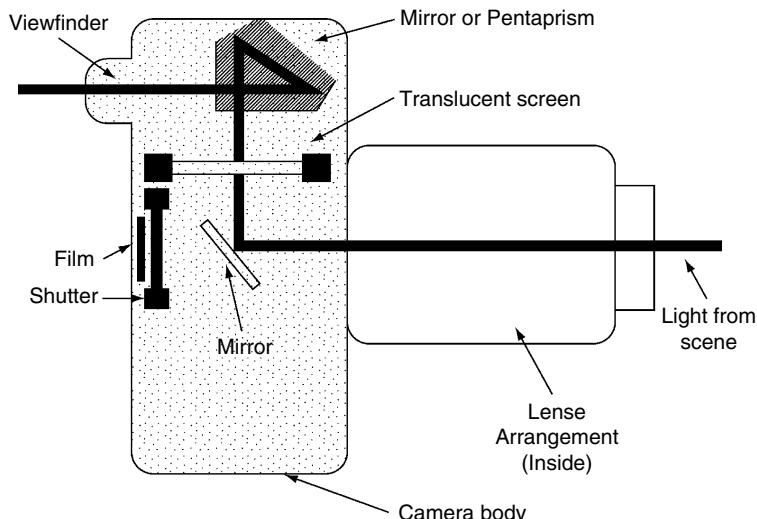


Fig. 1.39 *Side view of an SLR camera*

Upon pressing the shutter button, the camera takes the mirror out of the way, so the image is directed to the exposed film. The mirror is connected to the shutter timer system and thus, it stays put until the shutter is pressed. Analog or digital data resolution is a measure of the ability of an optical system to distinguish between signals that are spatially closer or spectrally similar.

1.8.10 Digital Camera

The key difference between digital and film-based cameras is that digital cameras use no film. Instead, the digital camera uses a sensor that converts light into electrical charges. Like a conventional camera, the digital camera also has a lens through which light from real-world objects enter the camera. Instead of falling on a film, the light falls on the image sensor. The image sensor employed by most digital cameras is either CCD or CMOS. Some low-end cameras use complementary metal-oxide semiconductor (CMOS) technology. While

CMOS sensors will almost certainly improve and become more popular in the future, they probably won't replace CCD sensors in higher-end digital cameras. The CCD is a collection of tiny light-sensitive diodes, which convert photons (light) into electrons (electrical charge). These diodes are called photosites. Each photosite is sensitive to light—the brighter the light that hits a signal photosite, the greater the electrical charge that will accumulate at that site.

One of the key reasons for the falling prices of digital cameras is the use of CMOS image sensors. The manufacturing cost of a CMOS sensor is much less when compared to that of a CCD sensor.

Both CCD and CMOS image sensors have to convert light into electrons at the photosites. A simple way to think about the sensor used in a digital camera is to think of it as having a 2-D array of thousands or millions of tiny solar cells, each of which transforms light from one small portion of the image into electrons. Both CCD and CMOS devices perform this task using a variety of technologies.

The next step is to read the value (accumulated charge) of each cell in the image. In a CCD, the charge is actually transported across the chip and read at one corner of the array. An analog-to-digital converter turns each pixel's value into a digital value. In most CMOS devices, there are several transistors at each pixel that amplify and move the charge using more traditional wires. The CMOS approach is more flexible because each pixel can be read individually.

CCDs use a special manufacturing process to create the ability to transport charge across the chip without distortion. This process leads to very high-quality sensors in terms of fidelity and light sensitivity. CMOS chips, on the other hand, use completely standard manufacturing processes to create the chip. CCDs can be used in cameras that focus on high-quality images with lots of pixels and excellent light sensitivity. CMOS sensors usually have lower quality, lower resolution and lower sensitivity. However, CMOS cameras are less expensive and have longer battery life.

The amount of detail that the camera can capture is called the resolution, and it is measured in pixels. With the increase in the number of pixels, the ability of the camera to capture finer details increases.

Some of the typical resolutions found in digital cameras are the following:

1. *640 × 480 pixels* This is the low-end resolution in most 'real' cameras. The total number of pixels corresponding to this resolution is 307,200.
2. *1216 × 912 pixels* This resolution is needed to print the image. The total number of pixels corresponding to this resolution is 1,108,992.
3. *1600 × 1200 pixels* Images taken with this high resolution (1,920,000 pixels) can be printed in larger sizes, such as 20.3 × 25.4 cm (8 × 10 inches), with good results. The total number of pixels corresponding to this resolution is 10.2 million pixels.

There are several ways for recording the three colours in a digital camera. The highest quality cameras use three separate sensors, each with a different filter over it. Light is directed to the different sensors by placing a beam splitter in the camera. Imagine light entering the camera as water flowing through a pipe. Using a beam splitter would be like dividing the same amount of water into three different pipes. Each sensor gets an identical look of the image, but because of the presence of filters, they only respond to one of the primary colours.

The advantage of this method is that the camera records each of the three colours at each pixel location. Unfortunately, cameras using this method tend to be bulky and expensive.

The second method is to rotate a series of red, blue and green filters in front of a single sensor. The sensor records the three separate images in rapid succession. This method also provides information about all the three colours at each pixel location. But since the three images aren't taken at precisely the same moment, both the camera and the target of the photo must remain stationary for all the three readings. This isn't practical in the case of candid photography or handheld cameras.

A more economical and practical way to record the three primary colours from a single image is to permanently place a filter over each individual photosite. By breaking up the sensor into a variety of red, blue and green pixels, it is possible to get enough information about the general vicinity of each sensor in order to enable us to make accurate guesses about the true colour at that location. This process of looking at the other pixels in the neighbourhood of a sensor and making an educated guess is called 'interpolation.'

1.8.11 Camcorder

A typical analog camcorder contains two basic parts:

- i. A camera section, made up of a CCD, lens, and motors to handle the zoom, focus and aperture.
- ii. A video-cassette recorder (VCR) section, in which a typical TV VCR is shrunk down to fit in a much smaller space.

The camera component receives the visual information and interprets it as an electronic video signal. The VCR component receives an electronic video signal and records it on a video tape as magnetic patterns. The latest ones have a CD-R for recording after some preprocessing.

A third component, the viewfinder, receives the video image as well, so that the photographer can view the scene to be shot. Viewfinders are actually small, B & W or colour televisions but modern camcorders also have larger full-colour liquid-crystal display (LCD) screens.

Besides these elements, digital camcorders have an analog-to-digital converter or a digital signal processor that takes the analog information from the camera and translates it into digital data, in addition to doing some image preprocessing. The audio is also converted into the digital format.

Video information in the digital form can also be loaded onto the computers, where the scene can be edited, copied and manipulated.

1.8.12 Ultrasound

Ultrasound or ultrasonography is a medical imaging technique that uses high-frequency sound waves and their echoes. The technique is similar to echolocation used by bats, whales and dolphins, as well as SONAR used by submarines. An ultrasound operates as follows:

- i. The ultrasound machine transmits high-frequency (1 to 5 MHz) sound pulses into the body using a probe.
- ii. The sound waves travel into the body and hit a boundary between tissues (e.g., the boundary between fluid and soft tissue, or soft tissue and bone).
- iii. Some of the sound waves are reflected back to the probe, while some travel further until they reach another boundary and get reflected.
- iv. The reflected waves are picked by the probe and relayed to the machine.
- v. The machine calculates the distance from the probe to the tissue or organ (boundaries) using the speed of sound in tissue (1.540 m/s) as a basis and the time of each echo's return (usually of the order of millionths of a second).
- vi. The machine calculates the distances and intensities of the echoes on the screen, forming a two-dimensional image.

In a typical ultrasound, millions of pulses and echoes are sent and received in each second. The probe can be moved along the surface of the body and angled to obtain various views. With the latest advancements, 3-D ultrasound imaging is possible.

1.8.13 Image Storage Mechanisms

The digital image acquired by a camera must be stored for processing and future use. Different ways of storing the digital images include

- (1) Connecting the camera to a computer and storing the images directly on the computer
- (2) Storing the image within the camera body in an integrated random access memory chip
- (3) Storing the image on a removable storage system that can be read by the computer

Basically, there are two types of removable storage systems in use, namely, (i) disk system, and (ii) flash memory.

Disk System

Several types of removable disks have been devised for use in cameras, but most of them conform to a set of standards from the Personal Computer Memory Card International Association (PCMCIA). There are three types of PCMCIA cards—Type I, Yype II and Type III. Types I and II were introduced in 1990, and Type III in 1992. Many cards offer functions other than memory such as modems, or network connection.

Type-I Card Type-I cards are 3.3-mm thick static RAM devices with no moving parts.

Type-II Card Type-II classification covers devices such as network connect cards, modems, and other connection units and is not directly applicable to digital camera storage.

Type-III Card Type-III cards are 10.5-mm thick cards that contain miniature hard drives using the same mechanics as that of a hard drive unit in a desktop computer.

Two types of PC storage cards are SRAM (static RAM) and flash memory. In SRAM, a small lithium battery is used to retain the data. In flash memory, data can be stored and erased as and when required.

1.8.14 Image Display

Image output is the final stage of the image-processing step. Often computer monitors are used to display the image. Computer monitors use cathode ray tube (CRT) displays.

The following factors should be taken into account while choosing monitors for image display.

(i) Size of the Monitor With the advancement in VLSI technology, the size of the monitor is decreasing year by year. Smaller the size of the monitor, better will be the portability. Nowadays flat-screen thin-film transistor (TFT) LCD monitors are available which consume less power when compared to conventional CRT monitors. TFT active matrix liquid crystal displays (LCD) allow flat-screen monitors of high definition. The major advantage of flat-screen monitors is the smaller amount of space they take up on a desk.

(ii) Number of Colours These are colours which can be displayed on a monitor. With the increase in the number of colours, high resolution images can be seen distinctly.

(iii) Spatial Resolution Sharpness of the image is given in terms of the spatial resolution. Spatial resolution is the number of pixels which can be displayed both vertically and horizontally.

With low spatial resolution, edges of objects, particularly curves and diagonal lines, will exhibit a stair-casing effect known as 'aliasing'. The aliasing effect is less with the increase in the number of pixels per inch. Monitor resolutions are quoted in pixels (example 640 horizontally x 480 vertically). A single pixel is created by several adjoining points of light on the display. Some typical monitor resolutions are given in Table 1.1.

Table 1.1 *Monitor resolution*

<i>Display mode</i>	<i>Resolution</i>
VGA	640×480
SVGA	800×600
XGA	1024×768
SXGA	1280×1024
UXGA	1600×1200

The term VGA stands for Video Graphics Array, SVGA for Super Video Graphics Array, XGA for Extended Graphics Array, SXGA for Super Extended Graphics Array, UXGA stands for Ultra Extended Graphics Array.

1.9 IMAGE FILE FORMATS

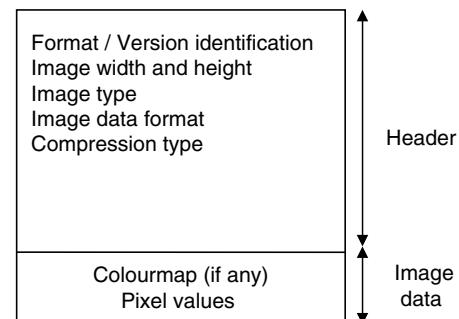
A digital image is often encoded in the form of binary files for the purpose of storage and transmission. A file format is a method used to store digital data and different file formats exist for storing images. Different file formats may compress the image data by differing amounts. Whatever be the format, the image file consists of two parts: (i) file header, and (ii) image data. The information stored in the header and image data sections are shown in Fig. 1.40.

Header Information The header part contains some of the vital information like format or version identification, width and height of the image, type of the image (binary, grayscale, colour image), and image data format which specifies the order in which pixel values are stored in the image data section. The header also specifies the type of compression mechanism. The length of the file header is often fixed. The header information must provide all the necessary information to reconstruct the original data and its organisation from the stored data.

Common Image File Formats Various image file formats are widely used for the purpose of digital image storage and retrieval. Each of these file formats possess certain characteristics which make them suitable for various applications. In the following section, we shall briefly discuss the basic image file formats. The common image file formats are (i) GIF (Graphics Interchange Format), (ii) JPEG (Joint Photographic Expert Group), (iii) PNG (Portable Network Graphics), (iv) TIFF (Tagged Image File Format), (v) PSD file format, and (vi) EPS.

1.9.1 GIF File Format

Graphics Interchange Format (GIF) was devised by UNISYS Corporation and Compuserve for transmitting graphical images over phone lines through modems. The GIF standard uses the Lempel–Ziv–Welch algorithm, modified slightly for image scan-line packets in order to use the line grouping of pixels effectively. The GIF standard is limited to 8-bit colour images only; hence it is best suited for images with few distinctive colours. The GIF format comes in two formats: (i) GIF 87a, and (ii) GIF 89a.

**Fig. 1.40** *Image file format*

The GIF 87a file format is shown in Fig. 1.41. The signature is 6 bytes. A GIF 87 file can contain more than one image definition. The screen descriptor, which is shown in Fig. 1.42 comprises a set of attributes that belong to each image in the file. The screen width is given in the first two bytes. Screen height is given in the next 2 bytes. The m in the byte 5 is 0 if no global colour map is given. Colour resolution, cr , is 3 bits in 0.....7. The next bit shown as 0 is extra and is not used in this standard. Pixel is another 3 bits, indicating the number of bits per pixel in the image, as stored in the file. Each image in the file has its own image descriptor.

The important features of GIF file format are summarised below:

- (i) The GIF file format uses lossless compression scheme. As a result, the quality of the image is preserved.
- (ii) GIF interlaced images can be displayed as low-resolution images initially and then develop clarity and detail gradually.
- (iii) GIF images can be used to create simple animations.
- (iv) GIF 89a images allow for one transparent colour.

Advantages of GIF File Format The advantages of GIF file format are summarised below:

- (i) GIF uses lossless compression algorithm that provides up to 4:1 compression of images which is the most widely supported graphics format on the Web.
- (ii) GIF supports transparency and interlacing.

Limitations of GIF File Format GIF file format supports a maximum of 256 colours. Due to this particular limitation, complex images may lose some detail when translated into GIF.

1.9.2 JPEG

JPEG is not actually a file type. JPEG is the most important current standard for image compression. JPEG standard was created by a working group of the International Organisation for Standardisation (ISO). This format provides the most dramatic compression option for photographic images. JPEG compression is used within the JFIF file format that uses the file extension (.jpg). This format is useful when the storage space is at a premium. JPEG pictures store a single raster image in 24-bit colour. JPEG is a platform-independent format that supports the highest levels of compression; however, this compression is lossy. Progressive JPEG files support interlacing.

The important features of JPEG file format are summarised below:

- (i) JPEG uses a lossy compression scheme.
- (ii) JPEG images are not interlaced; however, progressive JPEG images can be interlaced.

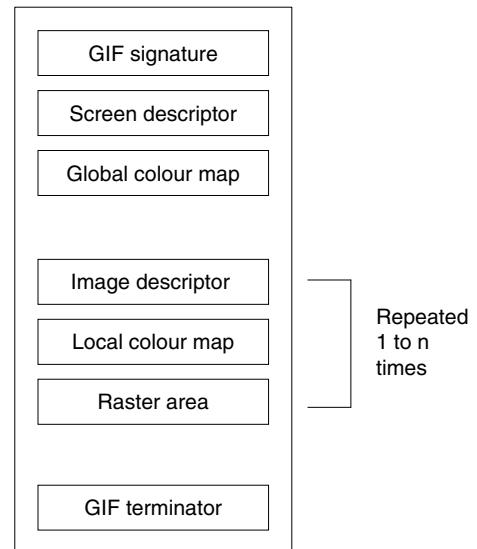


Fig. 1.41 GIF file format

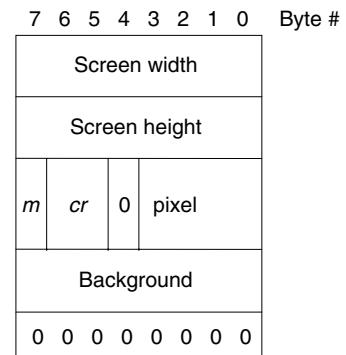


Fig. 1.42 GIF screen descriptor

Advantage of JPEG File Format The strength of JPEG file format is its ability to compress larger image files. Due to this compression, the image data can be stored effectively and transmitted efficiently from one place to another.

Limitations of JPEG File Format JPEG in its base version does not support multiple layers, high dynamic range. Hence JPEG will not be a wise choice if one is interested in maintaining high quality pictures.

1.9.3 JPEG2000

Recently, the JPEG committee has released a new version of their file format called JPEG2000. JPEG2000 employs wavelet-based image compression algorithm whereas JPEG employs DCT based image compression. JPEG2000 offers progressive image downloading. The progressive image downloading allows the user to download a lower resolution version of an image and to continue downloading a more detailed version if needed.

1.9.4 PNG

PNG stands for Portable Network Graphics. PNG is a bitmapped image format that employs lossless data compression. PNG was created to improve and replace the GIF format. The PNG file format is regarded, and was made as a free and open-source successor to the GIF file format. The PNG file format supports true colour (16 million colours), whereas the GIF file format only allows 256 colours. PNG excels when the image has large areas of uniform colour. The lossless PNG format is best suited for editing pictures, whereas the lossy formats like JPG are best for the final distribution of photographic-type images due to smaller file size. Yet many earlier browsers do not support the PNG file format; however with the release of Internet Explorer 7 all popular modern browsers fully support PNG. Special features of PNG files include support for up to 48 bits of colour information. The PNG format utilises a 2D interlacing scheme, which progressively displays the image much faster than a GIF image file. Gamma correction allows the values displayed on any platform to be the same as that of the original.

The important features of PNG images are summarised below:

- (i) PNG images use a lossless compression scheme.
- (ii) PNG images are interlaced.
- (iii) PNG images support 8-bit transparency.

1.9.5 TIFF

TIFF stands for Tagged Image File Format and was developed by the Aldus Corporation in the 1980s. It was later supported by Microsoft. TIFF files are often used with scanned images. Since a TIFF file does not compress an image file, hence images are often large but the quality is preserved. It can handle multiple images and data in a single file through the inclusion of ‘tags’ in the file header. Tags can indicate the basic geometry of the image, such as its size, or define how the image data is arranged and whether various image compression options are used, and uses a filename extension of TIFF or TIF. The TIFF format is often used to exchange files between applications and computer platforms. Within TIFF, a lossless compression routine known as LZW is available. This reduces the size of the stored file without perceptible loss in quality. The goals of the TIFF specification include extensibility, portability, and revisability.

Advantages of TIFF File Format The advantage of TIFF file format is that it can support any range of image resolution, size, and colour depth and different compression techniques.

Disadvantages of TIFF File Format The weakness of TIFF is its larger file size that limits its use in the Internet educational environment. Due to the same reason TIFF images are rarely utilised in web applications.

1.9.6 PSD

PSD is the Photoshop's default format which is capable of supporting layers, editable text and millions of colours. Also, the PSD file format has the capacity to retain information related to layers, masking channels, etc., which tend to be lost when saved in another file format. The PSD format has no compression features, but should still be used to archive complex images with multiple layers and sophisticated selections or paths. Photoshop 5.5 has a lossless compression routine built into it. The extra space needed for storage is compensated by the easy provision it offers future editing tasks.

1.9.7 EPS

EPS can be expanded as Encapsulated PostScript. EPS uses the PostScript page description language to describe images and is used to export files to page layout programs. This format supports both vector and bitmap graphics and has the unique benefit of being supported by virtually all graphics, illustrations and page layout programs. Each EPS image is made up of the same postscript commands that will eventually be sent to the postscript-based output device. EPS files can also include halftoning information such as screen ruling, angle and dot shape.

1.9.8 SVG

SVG stands for Scalable Vector Graphics and is a vector-based graphic file format meant for Internet use. This format is specifically based on XML, intended to create a smaller, scalable image file format for Internet use.

Advantage of SVG SVG file format has the advantage of vector graphics in which it is generally smaller than similar bitmapped files and scalable without any loss of resolution. The information for the image is stored in text data so that it can be searched and edited.

1.9.9 BMP

The Microsoft Windows Bitmap (BMP) file format is a basic file format for digital images in the Microsoft Windows world. BMP files have (i) a file header, (ii) a bitmap header, (iii) a colour table, and (iv) image data. The file header occupies the first 14 bytes of all BMP files. The structure of the BMP file header is shown in Fig. 1.43.

The first two bytes in the file header are reserved for the file type. The following four bytes give the size of the BMP file. The next two bytes are reserved and are always zero. The last four bytes of the header give the offset to the image data. This value points to the beginning of the image data in the file. The next 40 bytes are reserved for the bitmap header. The bitmap header is shown in Fig. 1.44.

The bitmap header begins with the size of the header, which is then followed by the width and height of the image data. If the height is a negative number, the image is stored bottom up. Usually the number of colour planes is one. The bits per pixel gives the number of bits required to represent a particular pixel. The next two fields deal with image compression. The compression field is 0 for no compression and 1 for run-length encoding compression. The next two fields deal with the resolution of the image

File type	0
File size	2
Zero	6
Zero	8
Bit map offset	10

Fig. 1.43 The BMP file header

data and the final two fields deal with the colours or gray shades in the image. The horizontal and vertical resolutions are expressed in pixels per metre. The colour field gives the number of colours or gray shades in the image. A colour table is a lookup table that assigns a gray shade or colour to a number given in the image data. The BMP colour table has four bytes in each colour table entry. The bytes are for the blue, green and red colour values. The final part of the BMP file is the image data. The data is stored row by row with padding at the end of each row.

1.10 APPLICATIONS OF DIGITAL IMAGE PROCESSING

Digital image processing is widely used in different fields like (i) Medicine, (ii) Forensics, (iii) remote sensing, (iv) communications, and (v) Automobiles.

(i) Medicine Digital image processing techniques like image segmentation and pattern recognition is used in digital mammography to identify tumours. Techniques like image registration and fusion play a crucial role in extracting information from medical images. In the field of telemedicine, lossless compression algorithms allow the medical images to be transmitted effectively from one place to another.

(ii) Forensics Security can be enhanced by personal identification. Different biometrics used in personal identification are face, fingerprint, iris, vein pattern, etc. Preprocessing of the input data is necessary for efficient personal identification. Commonly used preprocessing techniques include edge enhancement, denoising, skeletonisation, etc. Template-matching algorithms are widely used for proper identification.

(iii) Remote Sensing Remote sensing is the use of remote observations to make useful inferences about a target. Observations usually consist of measurements of electromagnetic radiation with different wavelengths of the radiation carrying a variety of information about the earth's surface and atmosphere. Remote sensing is being used increasingly to provide data for diverse applications like planning, hydrology, agriculture, geology and forestry. The image processing techniques used in the field of remote sensing include image enhancement, image merging and image-classification techniques. Multispectral image processing and texture classification are some of the useful image-processing techniques in the remote-sensing field.

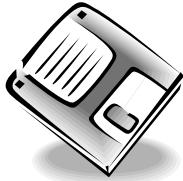
(iv) Communications With the growth of multimedia technology, information can be easily transmitted through the Internet. Video conferencing helps the people in different locations to interact lively. For video conferencing to be effective, the information has to be transmitted fast. Effective image and video compression algorithms like JPEG, JPEG2000, H.26X standards help to transmit the data effectively for live video conference.

(v) Automotives The latest development in the automotive sector is 'night vision system'. Night vision system helps to identify obstacles during night time to avoid accidents. Infrared cameras are invariably used in a night-vision system. The image-processing techniques commonly used in a night-vision system include image enhancement, boundary detection and object recognition.

Header size	0
Image width	4
Image height	8
Colour planes	12
Bits per pixel	14
Compression	16
Size of bitmap	20
Horizontal resolution	24
Vertical resolution	28
Colours	32
Important colours	36

Fig. 1.44 The Bitmap header

Summary



- Images are a way of recording and presenting information in a visual form.
- Digital image processing is a collection of techniques for the manipulation of digital images by computers.
- A digital image is composed of finite number of elements called pixels. Each pixel has a particular location and value.
- Sampling is the process of measuring the value of the physical image at discrete intervals in space. In order to sample the signal, it is necessary that the signal should be a band-limited signal.
- Each image sample corresponds to a small region of the physical image, which is called a picture element or pixel. A digital image corresponds to a two-dimensional array of pixels.
- Quantisation is the process of replacing the continuous values of the sampled image with a discrete set of quantisation levels. The number of quantisation levels employed governs the accuracy by which the image values are displayed.
- A vector image or geometric image represents an image mathematically through the use of geometrical primitives such as points, lines, curves and polygons.
- Image sensor is a 2D array of light-sensitive elements that convert photons to electrons. CCD and CMOS image sensors are widely used in image-capturing devices like digital cameras and camcorders.
- The resolution of a CCD image sensor is better than a CMOS image sensor, whereas a CMOS image sensor is cost-effective when compared to a CCD.
- Scanners convert analog images into digital image files. Common image scanners are drum scanners, flatbed scanners and flying-spot scanners.
- A file format is a method used to store digital data. The image file consists of two parts: (i) file header, and (ii) image data. The common image file formats include TIFF, BMP, JPEG, PNG, GIF and PSD.
- Popular image file formats such as TIFF, GIF, PNG and BMP use lossless compression algorithm whereas JPEG standard comes with both lossless and lossy compression algorithm.
- GIF comes in two formats: GIF87 and GIF89a. GIF89a supports simple animations.

Review Questions

1. What is the function of an image sensor?

The image sensor must capture incoming light, convert it into an electric signal, measure that signal and output it to supporting electronics.

2. Give the major classifications of an image sensor.

Image sensors can be broadly classified into (a) charge-coupled device (CCD) image sensors, and (b) Complementary Metal Oxide Semiconductor (CMOS) image sensors.

44 Digital Image Processing

3. Compare a CCD with a CMOS image sensor.

The following table shows the comparison of a CCD with a CMOS image sensor:

Feature	CCD	CMOS
Power consumption	High	Low
Fill factor	High	Low
Dynamic range	High	Moderate
Noise level	Low	High
System complexity	High	Low

4. Distinguish between line scan and area scan sensor.

Line scan sensor	Area scan sensor
<p>A line scan sensor scans one line of pixels at a time.</p> <p>A line scan sensor is more expensive.</p> <p>A line scan sensor is good for scanning objects on an assembly line or rotating cylindrical objects.</p>	<p>Area scan sensors scan one area of pixels at a time.</p> <p>A area scan sensor is less expensive.</p> <p>Comparatively in most applications, an area scan sensor is used.</p>

5. Distinguish between a monochrome and a grayscale image.

In a monochrome image, each pixel is stored as a single bit which is either '0' or '1'. In a grayscale image, each pixel is usually stored as a byte with a value between '0' and '255'.

6. Distinguish between a raster and a vector image.

Raster image	Vector image
<p>The raster image format is widely used for the representation of a continuous tone image.</p> <p>Raster images are resolution dependent. That is, scaling the raster image will diminish the image quality.</p> <p>Raster images are usually larger in file size.</p> <p>Raster images are quicker to display.</p>	<p>The vector image format is used for drawings and diagrams that can be described mathematically.</p> <p>Vector images are resolution independent. Vector images can be scaled up without any loss in image quality.</p> <p>Vector images are smaller in file size when compared to raster images.</p> <p>Vector images are slower to display.</p>

7. What do you mean by the term image file format? Mention some of the frequently used image file formats.

Image file format is an algorithm or a method used to store and display an image. Some of the popular image file formats are (i) JPEG, (ii) PNG, (iii) TIFF, (iv) GIF, (v) BMP, and (vi) PSD file formats.

8. Mention the situation in which the usage of GIF file format will be appropriate.

The usage of GIF file format is appropriate when graphics such as logos, illustrations or cartoons, rather than photographs are used. GIFs can contain only 256 colours or less and can be viewed by all the common browsers. GIFs also support animation, transparency and interlacing which is not common in other file formats.

9. When should we use the PNG file format?

The World Wide Web (WWW) consortium approved PNG to replace the GIF file format because the compression algorithm used by PNG is completely patent and license free. The PNG file format best handles monochromatic images and those images which may not be subjected to reduction and enlargement in future. PNG has a superior interlacing graphics scheme and supports a great deal of transparency.

10. When should we use the JPEG file format?

When an image has a lot of colours and if it is not possible to define the palette for them, it is better to use the 24-bit JPEG image file format. The JPEG file format can be saved in a variety of compression levels.

11. Compare JPEG, GIF and PNG image file formats with reference to features like number of bits, compression and animation.

The comparison of JPEG, GIF and PNG image file formats in the form of a table is given below.

Attribute	JPEG	GIF	PNG
Compression	Lossy	Lossless	Lossless
Bits	8 bits	24 bits	8 or 24 bits
Colour	Not transparent	Transparent	Better transparency
Animation	Not possible	Possible	Not possible

12. What do you mean by aliasing in the context of image sampling? Explain.

Aliasing is a phenomenon that occurs when an image is under-sampled. This phenomenon would not occur if the image is sampled at a rate of at least twice the highest frequency component in the image. Additional frequency components are present in an undersampled image that results in a change in the way the image looks.

Problems

- 1.1 You have a digital image that takes up 240 kb. The spatial resolution of the image is given by 600×200 . What is the bit depth? Hint: Bit depth \times No. of pixels = Total memory
- 1.2 Suppose that an image of dimensions 4×6 inches has frequency of 300 dots per inch in each direction. How many samples are required to preserve the information in the image?
- 1.3 The image $f(x, y) = 4 \cos(4\pi x) \cos(6\pi y)$ is sampled with $\Delta x = \Delta y = 0.5$. The sampled image is reconstructed with an ideal low-pass filter with cut off frequencies of $\pm \frac{1}{2\Delta x}$ and $\pm \frac{1}{2\Delta y}$. Find the reconstructed image.
- 1.4 A digital camera captures a 200×200 pixel image of a black-and-white football on a gray background. The ball is 30 cm in diameter and is 150 cm away from the camera. The camera has a field of 90 degrees horizontally and vertically. What is the diameter of the ball in pixels?
- 1.5 The 8.4 Mpixel Panasonic Lumix LX1 digital camera has been advertised as the world's first compact camera with a 16:9 aspect ratio CCD. At maximum resolution, the digital image consists of 3840 pixels horizontally by 2160 pixels vertically, arranged on a rectangular lattice. Given this information, what is the width X and the height Y of each CCD sensor element in units of picture height? Express the answer numerically as rational number such as 3/64. What is the sampling density and the number of samples per image?
- 1.6 If the object distance is +400 mm, locate the image created by a lens with focal length $f = +200$ mm and find the magnification of that image.
- 1.7 A sphere of radius = 1 metre is projected onto an image. The camera has a focal length of 10 mm, and each pixel corresponds to 0.01 mm on the image plane. The sphere projects to a circle of radius = 20 pixels on the image plane. How far away is the sphere from the camera?

46 Digital Image Processing

- 1.8 Describe in your own words how a charge-coupled device (CCD) is able to create and read out an image. Use figures to illustrate your answer.
- 1.9 Describe the difference between a bitmap and a vector image.

References

Books

1. Kenneth R Castleman, *Digital Image Processing*, Pearson Education, 1996
2. John C Russ, *The Image Processing Handbook*, CRC Press, fourth edition, 2002
3. Rafael C Gonzalez and Richard E Woods, *Digital Image Processing*, Pearson Education, second edition, 2002
4. Anil K Jain, *Fundamentals of Digital Image Processing*, Pearson Education, 1989
5. W F Schriener, *Fundamentals of Electronic Imaging Systems*, Springer-Verlag, 1986
6. G C Holst, *CCD Arrays, Cameras, and Displays*, SPIE Press, 1996
7. K Aizawa, K Saukaue and Y Suenaga eds., *Image Processing Technologies: Algorithms, Sensors, and Applications*, Marcel Dekker, 2004
8. J Miano, *Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP, Reading*, MA: Addison-Wesley, 1999
9. David C Kay and John R Levine, *Graphics File Formats*, McGraw-Hill, 1995

Journal Papers

1. R E Flory, *Image Acquisition Technology*, Proc. IEEE, vol. 73, pp. 613–637, April 1985
2. G J Declerck, ed., *Solid State Imagers and their Applications*, SPIE, vol. 591, 1986
3. Russell M Mersereau, *The Processing of Hexagonally Sampled Two-Dimensional Signals*, Proc. IEEE, vol. 67, no. 6, June 1979
4. H C Andrews, A G Tescher and R P Kruger, *Image Processing by Digital Computers*, IEE Spectrum, vol. 9, pp. 20–32, 1972

Web Resources

1. The webpage of Technical Advisory Committee for Images gives a good introduction to image file formats. Their URL is <http://www.tasi.ac.uk>
2. Introduction to image sensors: <http://www.shortcourses.com/sensors/>
3. Introduction to Raster and Vector Images: http://www.adobe.com/education/webtech/CS2/unit_graphics1/gb_bitmap_id.htm
4. Rich information about image sensors and papers presented in the conference: www.imagesensors.org
5. Reinhard R Beichel's lecture notes provides useful information regarding 2D sampling concepts: <http://www.icaen.uiowa.edu/~dip/LECTURE/ImageProperties2.html>
6. Digital image-processing fundamentals are given in this website: <http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip.html>

2

Learning Objectives

This chapter deals with 2D signals and systems. The transform which is widely used in system study is the Z-transform. An introduction to 2D Z-transforms is given in this chapter. After reading this chapter, the reader should be familiar with the following concepts:

2D signals

Periodic, aperiodic signals

2D linear shift Invariant systems

Properties of 2D systems

forward and inverse 2D Z-transforms

2D Signals and Systems

2.1 INTRODUCTION

Signals are variables that carry information. A signal is a function of one or more variables. An example of a one-dimensional signal is an ECG signal. An example of a 2D signal is a still image. The intensity of the image can vary along the 'x' and 'y' directions, and hence it is a 2D signal. Here 'x' and 'y' are called *spatial variables*.

A system basically processes the input signal and produces output signals. Thus, a system can be viewed as an operation or a set of operations performed on the input signal to produce an output signal.

2.2 2D SIGNALS

2D Discrete Signal 2D discrete signals are represented as $x(n_1, n_2)$ where $n_1, n_2 \rightarrow$ Pair of integers and $x \rightarrow$ real or complex value.

$x(n_1, n_2)$ represents the value or pixel intensity (in the case of an image) at the location (n_1, n_2) .

2.2.1 2D Unit Impulse Sequence

The 2D impulse sequence is given by $x(n_1, n_2) = \delta(n_1, n_2) = \begin{cases} 1, & n_1 = n_2 = 0 \\ 0, & \text{else} \end{cases}$. The 2D impulse sequence is depicted in Fig. 2.1.

2.2.2 Line Impulse

The different types of line impulses to be discussed are (i) vertical line impulse, (ii) horizontal line impulse, and (iii) diagonal impulse.

(i) Vertical Line Impulse The expression for vertical line impulse is given by $x(n_1, n_2) = \delta(n_1)$. The vertical line impulse is depicted in Fig. 2.2.

(ii) Horizontal Line Impulse The horizontal line impulse is given by $x(n_1, n_2) = \delta(n_2)$ and is shown in Fig. 2.3.

(iii) Diagonal Impulse The diagonal impulse has values only at the diagonal points, and at all other points, the value is zero. The mathematical expression for a diagonal impulse function is given by $x(n_1, n_2) = \delta(n_1 + n_2)$. The pictorial representation of the function is given in Fig. 2.4.

Another form of diagonal representation is $x(n_1, n_2) = \delta(n_1 - n_2)$. The pictorial representation of the function is given in Fig. 2.5.

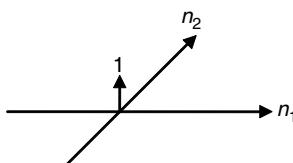


Fig. 2.1 2D impulse sequence

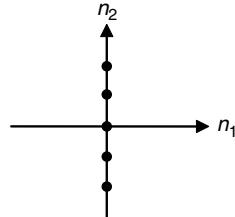


Fig. 2.2 Vertical line impulse

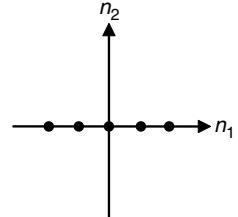


Fig. 2.3 Horizontal line impulse

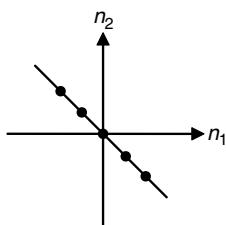


Fig. 2.4 Diagonal impulse function

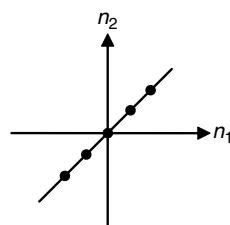


Fig. 2.5 Diagonal function representation

Example 2.1 Sketch the sequence $x(n_1, n_2) = \delta(2n_1 - n_2)$.

Solution From the given expression, it is clear that $x(n_1, n_2)$ would have a value only when $2n_1 - n_2 = 0$. This implies $2n_1 = n_2$. The sequence is represented in Fig. 2.6.

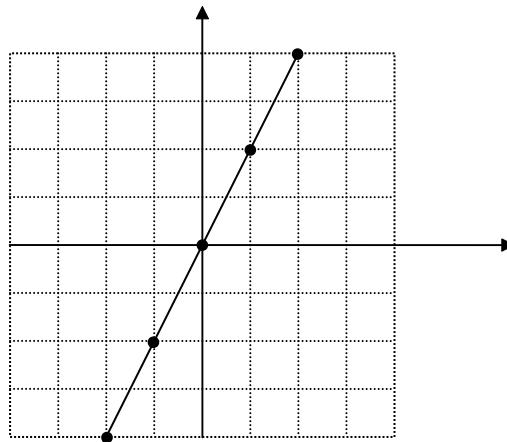


Fig. 2.6 Representation of the expression $x(n_1, n_2) = \delta(2n_1 - n_2)$

Example 2.2 Sketch the sequence $x(n_1, n_2) = \delta(n_1 + n_2 - 1)$

Solution From the given expression, it is clear that $x(n_1, n_2)$ will have a value only at $(n_1 + n_2 - 1) = 0$. This implies $n_2 = 1 - n_1$.

Case 1 When $n_1 = 0$, this implies $n_2 = 1$

Case 2 When $n_1 = 1$, this implies $n_2 = 0$. Plotting n_1 against n_2 we get the line as shown in Fig. 2.7.

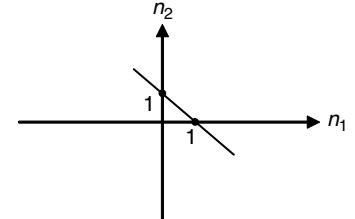


Fig. 2.7 Sequence for the expression $x(n_1, n_2) = \delta(n_1 + n_2 - 1)$

2.2.3 Exponential Sequence

The exponential sequences are defined by

$$x(n_1, n_2) = a^{n_1} b^{n_2}, -\infty < n_1, n_2 < \infty \quad (2.1)$$

where a and b are complex numbers. When a and b have unit magnitude they can be written as

$$a = e^{j\omega_1} \quad (2.2)$$

$$\text{and} \quad b = e^{j\omega_2} \quad (2.3)$$

in which case the exponential sequence becomes the complex sinusoidal sequence. Substituting (2.2) and (2.3) in Eq. (2.1), we get

$$x(n_1, n_2) = e^{j\omega_1 n_1 + j\omega_2 n_2} \quad (2.4)$$

$$x(n_1, n_2) = \cos(\omega_1 n_1 + \omega_2 n_2) + j \sin(\omega_1 n_1 + \omega_2 n_2) \quad (2.5)$$

The exponential sequences are eigen functions of 2D linear shift-invariant systems.

2.3 SEPARABLE SEQUENCE

A signal $x(n_1, n_2)$ is separable if it can be represented as a product of the function of n_1 alone and a function of n_2 alone. This is represented as

$$x(n_1, n_2) = x_1(n_1) x_2(n_2) \quad (2.6)$$

Examples of Separable Sequences The impulse sequence $\delta(n_1, n_2)$ is a separable sequence because $\delta(n_1, n_2)$ can be expressed as

$$\delta(n_1, n_2) = \delta(n_1)\delta(n_2)$$

where $\delta(n_1)$ and $\delta(n_2)$ are 1D impulses.

Another example of a separable sequence is a unit step sequence $u(n_1, n_2)$ since $u(n_1, n_2)$ can be expressed as

$$u(n_1, n_2) = u(n_1)u(n_2)$$

2.4 PERIODIC SEQUENCE

A 2D sequence is periodic if it repeats itself in a regularly spaced interval. A 2D sequence $x(n_1, n_2)$ is periodic with a period $N_1 \times N_2$ if the following equalities hold for all integers n_1 and n_2 :

$$x(n_1, n_2) = x(n_1 + N_1, n_2) = x(n_1, n_2 + N_2) \quad (2.7)$$

where N_1 and N_2 are positive integers.

Example 2.3 Determine whether the 2D signal $x(n_1, n_2) = \cos\left(\left(\frac{\pi}{4}\right)n_1 + \left(\frac{\pi}{2}\right)n_2\right)$ is periodic or not. If

periodic, determine the fundamental period.

Solution The signal $x(n_1, n_2)$ is of the form $x(n_1, n_2) = \cos(\omega_1 n_1 + \omega_2 n_2)$. For the signal to be periodic,

$$\frac{\omega_1}{2\pi} = \frac{\omega_2}{2\pi} = \text{ratio of two integers.}$$

In this problem, $\omega_1 = \frac{\pi}{4}$ and $\omega_2 = \frac{\pi}{2}$.

Now $\frac{\omega_1}{2\pi} = \frac{\pi}{4} \times \frac{1}{2\pi} = \frac{1}{8}$. This implies $N_1 = 8$ and $\frac{\omega_2}{2\pi} = \frac{\pi}{2} \times \frac{1}{2\pi} = \frac{1}{4}$. This implies $N_2 = 4$. The

signal is periodic because $\frac{\omega_1}{2\pi} = \frac{\omega_2}{2\pi} = \text{ratio of two integers}$. The fundamental period is given by $N_1 \times N_2 = 8 \times 4$.

Example 2.4 Determine whether the 2D signal $x(n_1, n_2) = \cos(n_1 + n_2)$ is periodic or not. If periodic, determine the fundamental period.

Solution For the 2D signal to be periodic, it should satisfy the condition

$$\frac{\omega_1}{2\pi} = \frac{\omega_2}{2\pi} = \text{ratio of two integers.}$$

In this problem, $\omega_1 = \omega_2 = 1$.

Therefore $\frac{\omega_1}{2\pi} = \frac{1}{2\pi}$ is not an integer and $\frac{\omega_2}{2\pi} = \frac{1}{2\pi}$ is not an integer and also the given sequence

cannot be expressed as $\cos((n_1 + N_1), n_2) = \cos(n_1, (n_2 + N_2))$ for all (n_1, n_2) for any non-zero integers N_1 and N_2 . Hence the given sequence is NOT periodic.

2.5 2D SYSTEMS

A 2D system is a device or algorithm that performs some prescribed operation on a 2D signal. If $x(n_1, n_2)$ is the input signal to a system and $y(n_1, n_2)$ is the output of the system then the relationship between the input and output of the system is given by

$$y(n_1, n_2) = T[x(n_1, n_2)]$$

where T denotes the transformation or processing performed by the system on the input $x(n_1, n_2)$ to produce the output $y(n_1, n_2)$. The 2D digital system can be defined as an operator that transforms the input sequence $x(n_1, n_2)$ to an output sequence $y(n_1, n_2)$.

2.6 CLASSIFICATION OF 2D SYSTEMS

The 2D systems can be broadly classified under four categories: (i) linear system, (ii) shift-invariant system, (iii) stable system, and (iv) static system.

2.6.1 Linear Versus Non-linear Systems

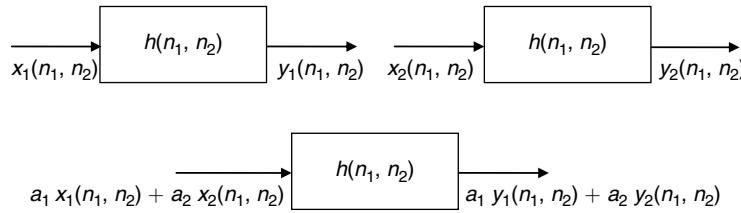
A linear system is one that satisfies the superposition principle. The principle of superposition requires that the response of the system to a weighted sum of signals should be equal to the corresponding weighted sum of the output of the system to each of the individual input signals. The linearity of the system T is defined as

$$T[ax_1(n_1, n_2) + bx_2(n_1, n_2)] = ay_1(n_1, n_2) + by_2(n_1, n_2)$$

Here 'a' and 'b' are any scalar constants. This property is illustrated in Fig. 2.8.

The superposition principle obeys both the scaling property and additive property.

According to the scaling property if the response of the system to the input $x_1(n_1, n_2)$ is $y_1(n_1, n_2)$, then the response of the system to $a_1x_1(n_1, n_2)$ is $a_1y_1(n_1, n_2)$. Any scaling of the input results in an identical scaling of the corresponding output.

**Fig. 2.8** Illustration of the superposition theorem

The additive property implies that

$$T[x_1(n_1, n_2) + x_2(n_1, n_2)] = T[x_1(n_1, n_2)] + T[x_2(n_1, n_2)] = y_1(n_1, n_2) + y_2(n_1, n_2) \quad (2.8)$$

The additivity and multiplicative (scalar) properties constitute the superposition principle. In a nutshell, the system is linear if the following two conditions hold good:

- (1) If the input signal is the sum of two sequences then the output signal is the sum of the two corresponding output sequences.
- (2) Scaling the input signal produces a scaled output signal.

Example 2.5 Determine whether the system described by the following input–output relation is linear or not.

$$y(n_1, n_2) = nx(n_1, n_2)$$

Solution If the input to the system is $x_1(n_1, n_2)$, the corresponding output is given by

$$y_1(n_1, n_2) = nx_1(n_1, n_2)$$

If the input to the system is $x_2(n_1, n_2)$, the corresponding output is given by

$$y_2(n_1, n_2) = nx_2(n_1, n_2)$$

Now the linear combination of the two input sequences $x_1(n_1, n_2)$ and $x_2(n_1, n_2)$ results in the output sequence $y_3(n_1, n_2)$ which is given by

$$\begin{aligned} y_3(n_1, n_2) &= T[a_1x_1(n_1, n_2) + a_2x_2(n_1, n_2)] \\ &= n[a_1x_1(n_1, n_2) + a_2x_2(n_1, n_2)] \\ y_3(n_1, n_2) &= a_1nx_1(n_1, n_2) + a_2nx_2(n_1, n_2) \end{aligned} \quad (2.9)$$

The linear combination of the two outputs is given by

$$a_1y_1(n_1, n_2) + a_2y_2(n_1, n_2) = a_1nx_1(n_1, n_2) + a_2nx_2(n_1, n_2) \quad (2.10)$$

Equations (2.9) and (2.10) are same, and thus the scaling and additivity properties are verified. Hence, the system is a linear system.

Example 2.6 Determine whether the system described by the following input–output relation is linear or not.

$$y(n_1, n_2) = e^{x(n_1, n_2)}$$

Solution For the system to be linear, it should be a relaxed system. This means that if the input is zero, the output should also be zero.

If the input $x(n_1, n_2)$ is zero, the corresponding output is given by

$$y(n_1, n_2) = e^0 = 1.$$

In this case, since the input to the system is zero and the output is not zero, the system cannot be considered to be a relaxed system and this implies that the given system is non-linear.

Example 2.7 Determine whether the system described by the following input–output relation is linear or not.

$$y(n_1, n_2) = Ax(n_1, n_2) + B$$

Solution First the input to the system is zero, that is $x(n_1, n_2) = 0$. Then the system output is given by $y(n_1, n_2) = A \times 0 + B = B$. The output of the system is not zero. Hence the system is non-linear.

The system can be made linear by making the value of B to be zero. If B is made zero then the system equation is given by $y(n_1, n_2) = Ax(n_1, n_2)$. If the input to the system is zero then the output is also linear. The system $y(n_1, n_2) = Ax(n_1, n_2)$ can be considered as either an amplifier or attenuator depending upon whether the value of ' A ' is greater or lesser than one. If the value of A is greater than one then the system behaves like an amplifier.

2.6.2 Shift-Variant Versus Shift-Invariant Systems

A system is a shift-invariant system if its input–output characteristics do not change with time. Mathematically, shift invariance is given by

$$T[x(n_1 - m_1, n_2 - m_2)] = y(n_1 - m_1, n_2 - m_2)$$

Note Linearity and shift-invariance are independent properties of a system in that neither property implies the other.

Example 2.8 Determine whether the system described by the following input–output relation is shift invariant or not.

$$y(n_1, n_2) = 5x(n_1, n_2)$$

Solution The time shift in the input produces the output $y_1(n_1, n_2)$ which is given by

$$y_1(n_1, n_2) = 5x(n_1 - k, n_2 - k) \quad (2.11)$$

The corresponding time shift in the output is represented by $y_2(n_1, n_2)$ which is given by

$$y_2(n_1, n_2) = 5x(n_1 - k, n_2 - k) \quad (2.12)$$

Equations (2.11) and (2.12) are the same and this shows that time shift in the input is equal to the time shift in the output. Hence, the system is a shift-invariant system.

Example 2.9 Determine whether the system described by the following input-output relation is shift invariant or not.

$$y(n_1, n_2) = nx(n_1, n_2)$$

Solution

$$\text{The system is given by } y(n_1, n_2) = nx(n_1, n_2) \quad (2.13)$$

$$\text{Time shift in the input is given by } y_1(n_1, n_2) = nx(n_1 - k, n_2 - k) \quad (2.14)$$

Time shift in the output is obtained by replacing ‘ n ’ by ‘ $n - k$ ’ in Eq. (2.13)

$$y_2(n_1, n_2) = (n - k)x(n_1 - k, n_2 - k) \quad (2.15)$$

Equations (2.14) and (2.15) are not the same. This implies that time shift in the input is not equal to the time-shift in the output. Hence, the system is time variant.

2.6.3 Linear Shift Invariant System

The main advantages of linear shift-invariant system are summarised below:

- (1) LSI systems are uniquely represented by their 2D impulse response. This means that if the impulse response of the system and the input to the system is known then the output of the system can be uniquely determined.
- (2) Complex exponentials are eigen functions of an LSI system. Fourier, Laplace and Z-transforms are based on complex exponential kernels.

2.6.4 Static versus Dynamic Systems

A system is static or memoryless if its output at any instant depends at most on the input sample but not on the past and future samples of the input. In any other case, the system is said to be dynamic or to have memory.

Example 2.10 Determine whether the system given by the input-output relation $y(n_1, n_2) = nx(n_1, n_2)$ is static or not.

Solution The output of the system depends only on the present value of the input and not on the past or future value of the input. Hence, the system is a static or a memoryless system.

2.6.5 Stability of a 2D LSI System

A 2D LSI system is Bounded Input Bounded Output (BIBO) stable if and only if its impulse response is absolutely summable which is given by

$$\sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} |h(n_1, n_2)| < \infty$$

If one considers the rational polynomials then non-essential singularity of the second kind will complicate the conditions.

2.7 2D CONVOLUTION

The response of a linear shift-invariant system can be described by convolution operation.

Properties of 2D Convolution

The following properties hold good for 2D convolution: (i) commutative property, (ii) associative property, (iii) distributive property, and (iv) convolution with shifted impulses.

(i) *Commutative Property* The commutative property is mathematically expressed as

$$x(n_1, n_2) * y(n_1, n_2) = y(n_1, n_2) * x(n_1, n_2) \quad (2.16)$$

(ii) *Associative Property* The associative property is given by

$$(x(n_1, n_2) * y(n_1, n_2)) * z(n_1, n_2) = x(n_1, n_2) * (y(n_1, n_2) * z(n_1, n_2)) \quad (2.17)$$

(iii) *Distributive Property*

$$x(n_1, n_2) * (y(n_1, n_2) + z(n_1, n_2)) = x(n_1, n_2) * y(n_1, n_2) + x(n_1, n_2) * z(n_1, n_2) \quad (2.18)$$

(iv) *Convolution with Shifted Impulses* The property of convolution with shifted impulses is given by

$$x(n_1, n_2) * \delta(n_1 - m_1, n_2 - m_2) = x(n_1 - m_1, n_2 - m_2) \quad (2.19)$$

2.8 2D Z-TRANSFORM

Continuous-time systems are commonly analyzed with the aid of the Laplace transform. For discrete-time systems, the transform corresponding to the Laplace transform is the Z-transform.

The Z-transform of a 2D sequence $x(n_1, n_2)$ is given by

$$X(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \quad (2.20)$$

where z_1 and z_2 are complex variables. The space of (z_1, z_2) is four dimensional. If z_i is restricted to lie on the unit circle, by substituting $z_i = e^{j\omega_i}$, the Z-transform is reduced to the Fourier transform:

$$X(z_1, z_2) \Big|_{z_1 = e^{j\omega_1}, z_2 = e^{j\omega_2}} = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) e^{-jn_1\omega_1} e^{-jn_2\omega_2} = X(\omega_1, \omega_2) \quad (2.21)$$

Thus, the Z-transform is a generalisation of the Fourier transform. Often a 2D Z-transform on an image can be brought about by a series of 1D Z-transforms.

2.8.1 Region of Convergence

The sum given in Eq. (2.20) may or may not converge for all values of z_1 and z_2 . The set of values for which the sum converges is known as the region of convergence (ROC) of the transform.

2.8.2 Z-transform of Sequences

Example 2.11 Compute the 2D Z-transform of the 2D impulse sequence $\delta(n_1, n_2)$.

Solution The 2D Z-transform of impulse sequence is given by

$$X(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \delta(n_1, n_2) z_1^{-n_1} z_2^{-n_2}$$

$\delta(n_1, n_2)$ has a value of '1' only at $n_1 = 0$ and $n_2 = 0$. Substituting the value of $n_1 = 0$ and $n_2 = 0$, we get

$$X(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \delta(n_1, n_2) z_1^{-0} z_2^{-0} = 1.$$

The region of convergence is the entire z_1 and z_2 plane.

Example 2.12 Compute the 2D Z-transform of the sequence $\delta(n_1 - 1, n_2)$.

Solution The 2D Z-transform of the impulse sequence is given by

$$X(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \delta(n_1 - 1, n_2) z_1^{-n_1} z_2^{-n_2} \quad (2.22)$$

The input signal will have a value of '1' only at $n_1 = 1$ and $n_2 = 0$. Substituting the value of $n_1 = 1$ and $n_2 = 0$ in Eq. (2.22), we get $X(z_1, z_2) = z_1^{-1}$.

Therefore Z-transform of $\delta(n_1 - 1, n_2) = z_1^{-1}$.

Example 2.13 Compute the 2D Z-transform of the sequence $\delta(n_1, n_2 - 1)$.

Solution The 2D Z-transform of the impulse sequence is given by

$$X(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \delta(n_1, n_2 - 1) z_1^{-n_1} z_2^{-n_2} \quad (2.23)$$

The signal will have a value of one only at $n_1 = 0$ and $n_2 = 1$. Substituting this value in Eq. (2.23), we get $X(z_1, z_2) = z_2^{-1}$.

Therefore, Z-transform of $\delta(n_1, n_2 - 1) = z_2^{-1}$ (2.24)

Example 2.14 Compute the 2D Z-transform of the sequence $\delta(n_1 - 1, n_2 - 1)$.

Solution The 2D Z-transform of the impulse sequence is given by

$$X(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \delta(n_1 - 1, n_2 - 1) z_1^{-n_1} z_2^{-n_2} \quad (2.25)$$

The signal will have a value of one only at $n_1 = 1$ and $n_2 = 1$. Substituting these values in Eq. (2.25), we get $X(z_1, z_2) = z_1^{-1} z_2^{-1}$.

Example 2.15 Compute the 2D Z-transform of $x(n_1, n_2) = u(n_1, n_2)$.

Solution The 2D Z-transform is given by

$$\begin{aligned} X(z_1, z_2) &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} u[n_1, n_2] z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} 1 \times z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=0}^{\infty} (z_1^{-1})^{n_1} \sum_{n_2=0}^{\infty} (z_2^{-1})^{n_2} = \frac{1}{1-z_1^{-1}} \times \frac{1}{1-z_2^{-1}} \end{aligned}$$

$$X(z_1, z_2) = \frac{z_1}{z_1 - 1} \times \frac{z_2}{z_2 - 1}$$

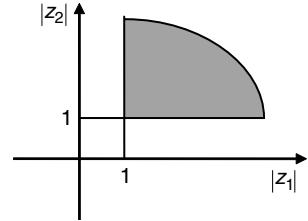


Fig. 2.9 Region of convergence of $u(n_1, n_2)$

The region of convergence is $|z_1| > 1$ and $|z_2| > 1$. The region of convergence is given in Fig. 2.9.

Example 2.16 Compute the 2D Z-transform of the function $x[n_1, n_2] = a^{n_1} b^{n_2} u(n_1, n_2)$.

Solution The Z-transform is given by

$$\begin{aligned} X(z_1, z_2) &= \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} a^{n_1} b^{n_2} z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=0}^{\infty} (az_1^{-1})^{-1} \sum_{n_2=0}^{\infty} (bz_2^{-1})^{-1} = \frac{1}{1-az_1^{-1}} \times \frac{1}{1-bz_2^{-1}} \\ &= \frac{z_1}{z_1 - a} \times \frac{z_2}{z_2 - b} \end{aligned}$$

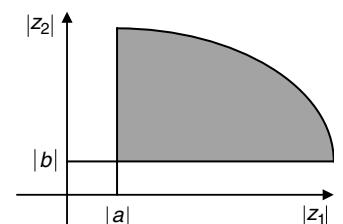


Fig. 2.10 Region of convergence

$X(z_1, z_2)$ is convergent for $|z_1| > a$ and $|z_2| > b$. This is illustrated in Fig. 2.10.

Example 2.17 Compute the 2D Z-transform of $x(n_1, n_2) = a^{n_1} \delta(n_1 - n_2) u(n_1, n_2)$.

Solution The 2D Z-transform is given by

$$\begin{aligned} X[z_1, z_2] &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} a^{n_1} \delta(n_1 - n_2) u(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} a^{n_1} \delta(n_1 - n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_2=0}^{\infty} z_2^{-n_2} \sum_{n_1=0}^{\infty} a^{n_1} \delta(n_1 - n_2) z_1^{-n_1} \end{aligned} .$$

The second summation will have a value only when $n_1 = n_2$. Substituting $n_1 = n_2$ in the above expression we get,

$$\begin{aligned} &= \sum_{n_2=0}^{\infty} a^{n_2} z_1^{-n_2} z_2^{-n_2} = \sum_{n_2=0}^{\infty} \left(a^1 z_1^{-1} z_2^{-1} \right)^{n_2} \\ X[z_1, z_2] &= \frac{1}{1 - a z_1^{-1} z_2^{-1}} \end{aligned}$$

The region of convergence is $|z_1 z_2| > |a|$.

Example 2.18 Determine the 2D Z-transform and its ROC for the following sequence:

$$x(n_1, n_2) = (-a^{n_1}) (-b^{n_2}) u(-n_1 - 1, -n_2 - 1).$$

Solution The Z-transform of the sequence is given by

$$\begin{aligned} X[z_1, z_2] &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} (-a^{n_1}) (-b^{n_2}) u(-n_1 - 1, -n_2 - 1) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=-\infty}^{-1} (-a^{n_1}) z_1^{-n_1} \times \sum_{n_2=-\infty}^{-1} (-b^{n_2}) z_2^{-n_2} \\ &= - \sum_{n_1=-\infty}^{-1} \left(a^{n_1} \right) z_1^{-n_1} \times - \sum_{n_2=-\infty}^{-1} \left(b^{n_2} \right) z_2^{-n_2} \end{aligned}$$

Let us substitute $n_1 = -m_1$ and $n_2 = -m_2$ in the above expression

$$\begin{aligned}
X[z_1, z_2] &= -\sum_{m_1=1}^{\infty} \left(a^{-m_1}\right) z_1^{m_1} \times -\sum_{m_2=1}^{\infty} \left(b^{-m_2}\right) z_2^{m_2} \\
&= -\sum_{m_1=1}^{\infty} \left(a^{-1} z_1\right)^{m_1} \times -\sum_{m_2=1}^{\infty} \left(b^{-1} z_2\right)^{m_2} \\
&= -\left\{ \left(a^{-1} z_1\right) + \left(a^{-1} z_1\right)^2 + \left(a^{-1} z_1\right)^3 + \dots \right\} \\
&\quad \times -\left\{ \left(b^{-1} z_2\right) + \left(b^{-1} z_2\right)^2 + \left(b^{-1} z_2\right)^3 + \dots \right\} \\
&= -\left\{ a^{-1} z_1 \left(1 + \left(a^{-1} z_1\right) + \left(a^{-1} z_1\right)^2 + \left(a^{-1} z_1\right)^3 + \dots \right) \right\} \\
&\quad \times -\left\{ b^{-1} z_2 \left(1 + \left(b^{-1} z_2\right) + \left(b^{-1} z_2\right)^2 + \left(b^{-1} z_2\right)^3 + \dots \right) \right\} \\
&= -\left(a^{-1} z_1\right) \times \frac{1}{1 - a^{-1} z_1} \times -\left(b^{-1} z_2\right) \times \frac{1}{1 - b^{-1} z_2} \\
&= -\frac{z_1}{a} \times \frac{a}{a - z_1} \times -\frac{z_2}{b} \times \frac{b}{b - z_2} \\
&= \frac{z_1}{a} \times \frac{a}{z_1 - a} \times \frac{z_2}{b} \times \frac{b}{z_2 - b}
\end{aligned}$$

$$Z\left\{ \left(-a^{n_1}\right) \left(-b^{n_2}\right) u(-n_1-1, -n_2-1) \right\} = \frac{z_1}{z_1 - a} \times \frac{z_2}{z_2 - b}$$

The ROC is $|z_1| < a$ and $|z_2| < b$.

2.8.3 Z-Transform of Finite Duration Sequence

In this section, problems and solutions related to 2D Z-transform of finite duration sequence are given.

Example 2.19 Determine the 2D Z-transform of the sequence given in Fig. 2.11.

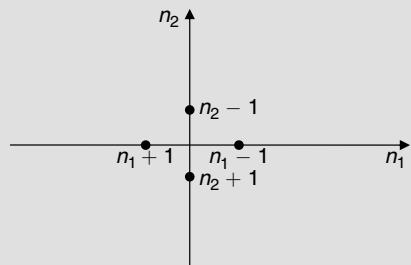


Fig. 2.11 2D sequence

Solution The 2D sequence is given by

$$x(n_1, n_2) = \delta(n_1 - 1) + \delta(n_1 + 1) + \delta(n_2 - 1) + \delta(n_2 + 1) \quad (2.26)$$

We know that

$$\begin{aligned} Z\{\delta(n_1 - 1)\} &= z_1^{-1} \\ Z\{\delta(n_1 + 1)\} &= z_1^1 \\ Z\{\delta(n_2 - 1)\} &= z_2^{-1} \\ Z\{\delta(n_2 + 1)\} &= z_2^1 \end{aligned}$$

Substituting these values in Eq. (2.26), we get

$$X[z_1, z_2] = z_1^{-1} + z_1^1 + z_2^{-1} + z_2^1 \quad (2.27)$$

In order to get the frequency response, substituting $z_1 = e^{j\omega_1}$ and $z_2 = e^{j\omega_2}$ in Eq. (2.27) we get

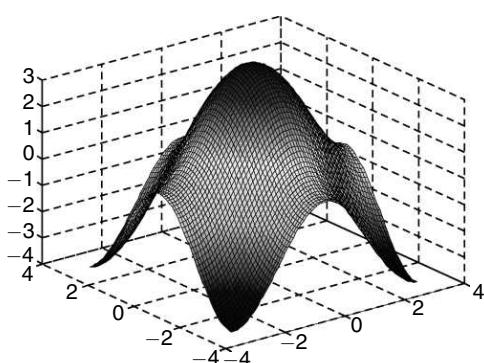
$$X[\omega_1, \omega_2] = e^{-j\omega_1} + e^{-j\omega_1} + e^{-j\omega_2} + e^{j\omega_2}$$

Multiplying and dividing by two, we get

$$X[\omega_1, \omega_2] = 2 \times \left(\frac{e^{-j\omega_1} + e^{-j\omega_1}}{2} \right) + 2 \times \left(\frac{e^{-j\omega_2} + e^{j\omega_2}}{2} \right)$$

$$X[\omega_1, \omega_2] = 2\cos\omega_1 + 2\cos\omega_2$$

The frequency response and the corresponding MATLAB code are given in Fig. 2.12.



```
close all;
clear all;
clc;
[X, Y] = meshgrid(-pi:0.09:pi);
Z = 2*cos(X) + 2*cos(Y);
surf(X, Y, Z),
axis([-4 4, -4 4, -4 3])
```

Fig. 2.12 Frequency response

Example 2.20 Determine the 2D Z-transform and the frequency response of the 2D sequence shown in Fig. 2.13.

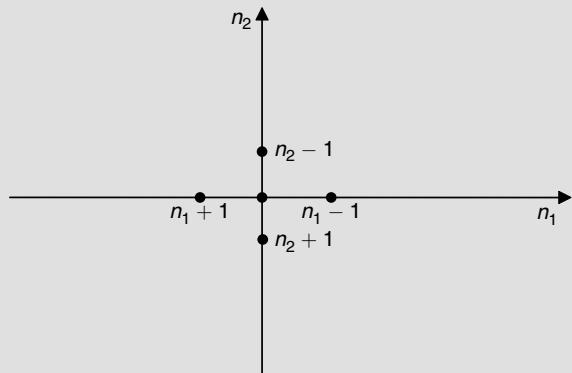


Fig. 2.13 2D Sequence

Solution The basic difference between the sequence shown in Fig. 2.13 and the sequence shown in Fig. 2.11 is the origin. In Fig. 2.13, the origin is included. The 2D sequence shown in Fig. 2.13 can be represented by

$$x(n_1, n_2) = \delta(n_1, n_2) + \delta(n_1 - 1) + \delta(n_1 + 1) + \delta(n_2 - 1) + \delta(n_2 + 1) \quad (2.28)$$

We know that

$$\begin{aligned} Z\{\delta(n_1 - 1)\} &= z_1^{-1} \\ Z\{\delta(n_1 + 1)\} &= z_1^1 \\ Z\{\delta(n_2 - 1)\} &= z_2^{-1} \\ Z\{\delta(n_2 + 1)\} &= z_2^1 \\ Z\{\delta(n_1, n_2)\} &= 1 \end{aligned}$$

Substituting these values in Eq. (2.28), we get

$$X[z_1, z_2] = 1 + z_1^{-1} + z_1^1 + z_2^{-1} + z_2^1 \quad (2.29)$$

Frequency response In order to obtain the frequency response, replacing $z_1 = e^{j\omega_1}$ and $z_2 = e^{j\omega_2}$ in Eq. (2.29), we get

$$X[\omega_1, \omega_2] = 1 + e^{-j\omega_1} + e^{-j\omega_1} + e^{-j\omega_2} + e^{j\omega_2}$$

Multiplying and dividing by two, we get

$$X[\omega_1, \omega_2] = 1 + 2 \times \left(\frac{e^{-j\omega_1} + e^{-j\omega_1}}{2} \right) + 2 \times \left(\frac{e^{-j\omega_2} + e^{j\omega_2}}{2} \right)$$

$$X[\omega_1, \omega_2] = 1 + 2\cos\omega_1 + 2\cos\omega_2$$

Example 2.21 Determine the 2D Z-transform and the frequency response of the 2D sequence shown in Fig. 2.14.

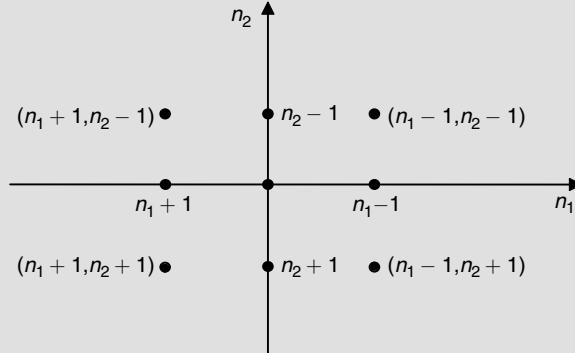


Fig. 2.14 2D sequence

Solution The expression for the 2D sequence shown in Fig. 2.14 is

$$x(n_1, n_2) = \delta(n_1, n_2) + \delta(n_1 - 1) + \delta(n_1 + 1) + \delta(n_2 - 1) + \delta(n_2 + 1) + \delta(n_1 - 1, n_2 - 1) \\ + \delta(n_1 - 1, n_2 + 1) + \delta(n_1 + 1, n_2 - 1) + \delta(n_1 + 1, n_2 + 1)$$

Taking Z-transform on both sides we get,

$$X[z_1, z_2] = 1 + z_1^{-1} + z_1^1 + z_2^{-1} + z_2^1 + z_1^{-1}z_2^{-1} + z_1^{-1}z_2^1 + z_1^1z_2^{-1} + z_1z_2 \quad (2.30)$$

Frequency response In order to obtain the frequency response, replacing $z_1 = e^{j\omega_1}$ and $z_2 = e^{j\omega_2}$ in Eq. (2.30), we get

$$X[\omega_1, \omega_2] = 1 + e^{-j\omega_1} + e^{j\omega_1} + e^{-j\omega_2} + e^{j\omega_2} + e^{-j\omega_1}e^{-j\omega_2} + e^{-j\omega_1}e^{j\omega_2} + e^{j\omega_1}e^{-j\omega_2} + e^{j\omega_1}e^{j\omega_2}$$

$$X[\omega_1, \omega_2] = 1 + e^{-j\omega_1} + e^{j\omega_1} + e^{-j\omega_2} + e^{j\omega_2} + e^{-j\omega_1}(e^{-j\omega_2} + e^{j\omega_2}) + e^{j\omega_1}(e^{-j\omega_2} + e^{j\omega_2})$$

$$X[\omega_1, \omega_2] = 1 + 2 \times \left(\frac{e^{-j\omega_1} + e^{j\omega_1}}{2} \right) + 2 \times \frac{e^{-j\omega_2} + e^{j\omega_2}}{2} \\ + e^{-j\omega_1} \left[2 \times \frac{e^{-j\omega_2} + e^{j\omega_2}}{2} \right] + e^{j\omega_1} \left[2 \times \frac{e^{-j\omega_2} + e^{j\omega_2}}{2} \right]$$

$$X[\omega_1, \omega_2] = 1 + 2 \cos \omega_1 + 2 \cos \omega_2 + e^{-j\omega_1} \times 2 \cos \omega_2 + e^{j\omega_1} \times 2 \cos \omega_2$$

$$X[\omega_1, \omega_2] = 1 + 2 \cos \omega_1 + 2 \cos \omega_2 + 2 \cos \omega_2 (e^{j\omega_1} + e^{-j\omega_1})$$

$$X[\omega_1, \omega_2] = 1 + 2 \cos \omega_1 + 2 \cos \omega_2 + 2 \cos \omega_2 2 \times \frac{(e^{j\omega_1} + e^{-j\omega_1})}{2}$$

$$X[\omega_1, \omega_2] = 1 + 2 \cos \omega_1 + 2 \cos \omega_2 + 2 \cos \omega_2 2 \cos \omega_1$$

$$X[\omega_1, \omega_2] = 1 + 2 \cos \omega_1 + 2 \cos \omega_2 + 4 \cos \omega_1 \cos \omega_2.$$

Example 2.22 Determine the 2D Z-transform and the frequency response for the pattern shown in Fig. 2.15.

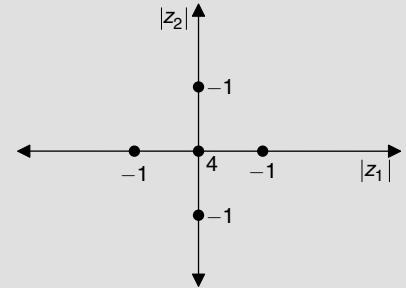


Fig. 2.15 2D pattern

Solution The pattern in Fig. 2.15 is represented by

$$x(n_1, n_2) = 4\delta(n_1, n_2) - \delta(n_1 - 1) - \delta(n_1 + 1) - \delta(n_2 - 1) - \delta(n_2 + 1) \quad (2.31)$$

Taking Z-transform both sides we get,

$$X[z_1, z_2] = 4 - z_1 - z_1^{-1} - z_2 - z_2^{-1} \quad (2.32)$$

Frequency response To determine the frequency response, replacing $z_1 = e^{j\omega_1}$ and $z_2 = e^{j\omega_2}$ in the expression for $X[z_1, z_2]$, we get

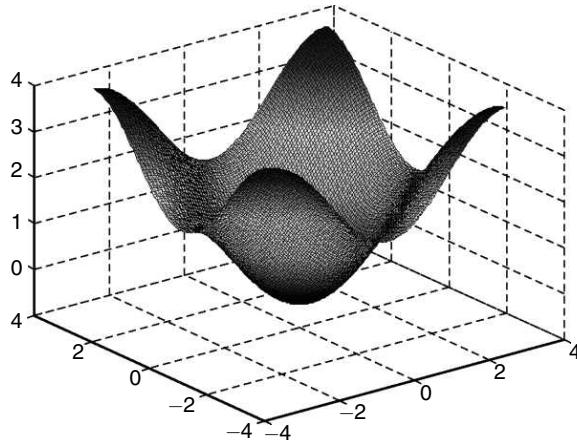
$$X(\omega_1, \omega_2) = 4 - (e^{j\omega_1} + e^{-j\omega_1}) - (e^{j\omega_2} + e^{-j\omega_2})$$

$$X(\omega_1, \omega_2) = 4 - 2 \times \frac{(e^{j\omega_1} + e^{-j\omega_1})}{2} - 2 \times \frac{(e^{j\omega_2} + e^{-j\omega_2})}{2}$$

$$X(\omega_1, \omega_2) = 4 - 2 \cos \omega_1 - 2 \cos \omega_2$$

Dividing throughout by two, we get

$$X(\omega_1, \omega_2) = 2 - \cos \omega_1 - \cos \omega_2$$



```

close all;
clear all;
clc;
[X,Y]=meshgrid(
-pi:.05:pi);
Z = 2-cos(X)-
cos(Y);
surf(X,Y,Z),
axis([-4 4,-4
4,-0.5 4])

```

Fig. 2.16 Frequency response and the MATLAB code

2.8.4 Properties of Z-transform

The Z-transform operation has a number of properties that can be useful in solving problems. In this section, we list some of the useful properties of 2D Z-transforms. Some of the properties can be viewed as an extension of 1D case.

(i) Linearity Property If the signal $x(n_1, n_2)$ has Z-transform of $X(z_1, z_2)$ with ROC denoted as R_x and the signal $y(n_1, n_2)$ has a Z-transform of $Y(z_1, z_2)$ with ROC denoted as R_y then the Z-transform of the sequence $ax(n_1, n_2) + by(n_1, n_2)$ is given by

$$Z\{ax(n_1, n_2) + by(n_1, n_2)\} = aX(z_1, z_2) + bY(z_1, z_2) \quad (2.33)$$

with ROC $R_x \cap R_y$.

Proof The Z-transform of LHS

$$\begin{aligned}
Z\{ax(n_1, n_2) + by(n_1, n_2)\} &= \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} \{ax(n_1, n_2) + by(n_1, n_2)\} z_1^{-n_1} z_2^{-n_2} \\
&= \sum_{n_1=0}^{\infty} ax(n_1, n_2) z_1^{-n_1} + \sum_{n_2=0}^{\infty} by(n_1, n_2) z_2^{-n_2} \\
&= a \sum_{n_1=0}^{\infty} x(n_1, n_2) z_1^{-n_1} + b \sum_{n_2=0}^{\infty} y(n_1, n_2) z_2^{-n_2}
\end{aligned}$$

$$Z\{ax(n_1, n_2) + by(n_1, n_2)\} = aX[z_1, z_2] + bY[z_1, z_2] \text{ with ROC } R_x \cap R_y.$$

(ii) Convolution Property If the signal $x(n_1, n_2)$ has a Z-transform of $X(z_1, z_2)$ with ROC denoted as R_x and the signal $y(n_1, n_2)$ has a Z-transform of $Y(z_1, z_2)$ with ROC denoted as R_y then the convolution property is given by

$$Z\{x(n_1, n_2) * y(n_1, n_2)\} = X(z_1, z_2) \times Y(z_1, z_2) \quad (2.34)$$

with ROC at least $R_x \cap R_y$.

Proof The convolution between $x(n_1, n_2)$ and $y(n_1, n_2)$ is given by

$$x(n_1, n_2) * y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} [x(n_1, n_2) * y(n_1, n_2)] z_1^{-n_1} z_2^{-n_2} \quad (2.35)$$

Taking Z-transforms on both sides, we get

$$\begin{aligned} Z\{x(n_1, n_2) * y(n_1, n_2)\} &= Z\left\{ \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} [x(k_1, k_2) y(n_1 - k_1, n_2 - k_2)] \right\} \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} [x(k_1, k_2) y(n_1 - k_1, n_2 - k_2)] z_1^{-n_1} z_2^{-n_2} \quad (2.36) \end{aligned}$$

Let $n_1 - k_1 = l_1$ and $n_2 - k_2 = l_2$. Substituting this in Eq. (3.26), we get

$$\begin{aligned} &= \sum_{l_1=-\infty}^{\infty} \sum_{l_2=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} [x(k_1, k_2) y(l_1, l_2)] z_1^{-(l_1+k_1)} z_2^{-(l_1+k_1)} \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \sum_{l_1=-\infty}^{\infty} \sum_{l_2=-\infty}^{\infty} [x(k_1, k_2) y(l_1, l_2)] z_1^{-k_1} z_1^{-l_1} z_2^{-k_2} z_2^{-l_2} \\ &= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) z_1^{-k_1} z_2^{-k_2} \sum_{l_1=-\infty}^{\infty} \sum_{l_2=-\infty}^{\infty} y(l_1, l_2) z_1^{-l_1} z_2^{-l_2} \end{aligned}$$

$$Z\{x(n_1, n_2) * y(n_1, n_2)\} = X[z_1, z_2] \times Y[z_1, z_2].$$

This shows that convolution in spatial domain is equal to multiplication in the frequency domain.

(iii) Delay Property or Shifting Property If the 2D Z-transform of the sequence $x(n_1, n_2)$ is $X[z_1, z_2]$ then the 2D Z-transform of delayed version of $x(n_1, n_2)$ by a factor of m_1 and m_2 is given by

$$Z\{x(n_1 - m_1, n_2 - m_2)\} = z_1^{-m_1} z_2^{-m_2} X[z_1, z_2]$$

Proof The Z-transform of the shifting property is

$$Z\{x(n_1 - m_1, n_2 - m_2)\} = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} x(n_1 - m_1, n_2 - m_2) z_1^{-n_1} z_2^{-n_2} \quad (2.37)$$

Substituting $n_1 - m_1 = k_1$ and $n_2 - m_2 = k_2$ in Eq. (2.37), we get

$$\begin{aligned} &= \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} x(k_1, k_2) z_1^{-[m_1+k_1]} z_2^{-[m_2+k_2]} \\ &= \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} x(k_1, k_2) z_1^{-m_1} z_1^{-k_1} z_2^{-m_2} z_2^{-k_2} \\ &= z_1^{-m_1} z_2^{-m_2} \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} x(k_1, k_2) z_1^{-k_1} z_2^{-k_2} \\ Z\{x(n_1 - m_1, n_2 - m_2)\} &= z_1^{-m_1} z_2^{-m_2} X[z_1, z_2]. \end{aligned}$$

The region of convergence is $\text{ROC} = R_x$.

(iv) Conjugation Property If the 2D Z-transform of the sequence $x(n_1, n_2)$ is $X[z_1, z_2]$ then the 2D Z-transform of $x^*(n_1, n_2)$ is given by

$$Z\{x^*(n_1, n_2)\} = X^*(z_1^*, z_2^*) \quad (2.38)$$

Proof The Z-transform of the sequence $x(n_1, n_2)$ is represented by $X[z_1, z_2]$. The expression of $X[z_1, z_2]$ is given by

$$X[z_1, z_2] = \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}$$

Taking conjugate both sides, we get

$$X^*[z_1, z_2] = \left[\sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \right]^*$$

$$X^*[z_1, z_2] = \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x^*(n_1, n_2) (z_1^*)^{-n_1} (z_2^*)^{-n_2}$$

Now substituting $z_1 = z_1^*$ and $z_2 = z_2^*$ in the above expression, we get

$$X^*[z_1^*, z_2^*] = \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x^*(n_1, n_2) z_1^{-n_1} z_2^{-n_2} = Z\{x^*(n_1, n_2)\}.$$

(v) Reflection Property The 2D Z-transform of the sequence $x(-n_1, -n_2)$ is given by

$$Z\{x(-n_1, -n_2)\} = X(z_1^{-1}, z_2^{-1}) \quad (2.39)$$

Proof The Z-transform of the sequence $x(n_1, n_2)$ is represented by $X[z_1, z_2]$. The expression of $X[z_1, z_2]$ is given by

$$X[z_1, z_2] = \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}$$

The Z-transform of $x(-n_1, -n_2)$ is

$$= \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(-n_1, -n_2) z_1^{-n_1} z_2^{-n_2}$$

Now replacing $-n_1$ by n_1 and $-n_2$ by n_2 in the above expression we get

$$\begin{aligned} &= \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{n_1} z_2^{n_2} \\ &= \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) (z_1^{-1})^{-n_1} (z_2^{-1})^{-n_2} = X[z_1^{-1}, z_2^{-1}] \end{aligned}$$

(vi) Differentiation Property The differentiation property of 2D Z-transform is given by

$$-n_1 x(n_1, n_2) \leftrightarrow z_1 \frac{\partial X(z_1, z_2)}{\partial z_1} \text{ ROC: } R_x \quad (2.40)$$

$$-n_2 x(n_1, n_2) \leftrightarrow z_2 \frac{\partial X(z_1, z_2)}{\partial z_2} \text{ ROC: } R_y \quad (2.41)$$

$$-n_1 n_2 x(n_1, n_2) \leftrightarrow z_1 z_2 \frac{\partial^2 X(z_1, z_2)}{\partial z_1 \partial z_2} \quad (2.42)$$

Proof The Z-transform of the sequence $x(n_1, n_2)$ is represented by $X[z_1, z_2]$. The expression of $X[z_1, z_2]$ is given by

$$X[z_1, z_2] = \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}$$

Now, differentiating partially the above equation with respect to z_1 , we get

$$\begin{aligned}\frac{\partial X(z_1, z_2)}{\partial z_1} &= \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) (-n_1) z_1^{-n_1-1} z_2^{-n_2} \\ &= -n_1 \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_1^{-1} z_2^{-n_2} \\ &= -\frac{n_1}{z_1} \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ z_1 \frac{\partial X(z_1, z_2)}{\partial z_1} &= -n_1 \sum_{n=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}\end{aligned}$$

This implies the Z-transform of $n_1 x(n_1, n_2)$ is given by $-z_1 \frac{\partial X(z_1, z_2)}{\partial z_1}$.

Similarly, the Z-transform of $n_2 x(n_1, n_2)$ is given by $-z_2 \frac{\partial X(z_1, z_2)}{\partial z_2}$ and

The Z-transform of $n_1 n_2 x(n_1, n_2)$ is given by $-z_1 z_2 \frac{\partial^2 X(z_1, z_2)}{\partial z_1 \partial z_2}$.

(vii) Modulation Property The multiplication property of a 2D Z-transform is given by

$$Z\left\{\alpha^{-n_1} \beta^{-n_2} x(n_1, n_2)\right\} = X(\alpha z_1, \beta z_2) \quad (2.43)$$

$$\begin{aligned}\text{Proof} \quad Z\left\{\alpha^{-n_1} \beta^{-n_2} u(n_1, n_2)\right\} &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \alpha^{-n_1} \beta^{-n_2} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \alpha^{-n_1} \beta^{-n_2} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) \alpha^{-1} z_1^{-1} \beta^{-1} z_2^{-1} \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) \left(\alpha^{-1} z_1^{-1}\right)^{n_1} \left(\beta^{-1} z_2^{-1}\right)^{n_2} \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) (\alpha z_1)^{-n_1} (\beta z_2)^{-n_2}\end{aligned}$$

$$Z\left\{\alpha^{-n_1}\beta^{-n_2}u(n_1, n_2)\right\} = X(z_1, \beta z_2)$$

The other properties of 2D Z-transforms such as multiplication property or product of two sequences and Parseval's theorem, you need some idea about 2D inverse Z-transforms. Hence, 2D inverse Z-transforms is presented here.

2.8.5 2D Inverse Z-Transform

The Z-Transform of the signal $x(k_1, k_2)$ is

$$X(z_1, z_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) z_1^{-k_1} z_2^{-k_2} \quad (2.44)$$

Suppose that we multiply both sides of (2.44) by $z_1^{n_1-1} z_2^{n_2-1}$ and integrate both sides over a closed contour within the ROC of $X(z_1, z_2)$ which enclose the origin.

$$\begin{aligned} \oint_{c_1} \oint_{c_2} X(z_1, z_2) z_1^{n_1-1} z_2^{n_2-1} dz_1 dz_2 &= \oint_{c_1} \oint_{c_2} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) z_1^{-k_1} z_2^{-k_2} z_1^{n_1-1} z_2^{n_2-1} dz_1 dz_2 \\ \oint_{c_1} \oint_{c_2} X(z_1, z_2) z_1^{n_1-1} z_2^{n_2-1} dz_1 dz_2 &= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) \oint_{c_1} \oint_{c_2} z_1^{n_1-1-k_1} z_2^{n_2-1-k_2} dz_1 dz_2 \end{aligned} \quad (2.45)$$

Now we can invoke the Cauchy integral theorem, which states that

$$= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} z_1^{n_1-1-k_1} z_2^{n_2-1-k_2} dz_1 dz_2 = \begin{cases} 1 & k_1 = n_1 \text{ and } k_2 = n_2 \\ 0 & k_1 \neq n_1 \text{ and } k_2 \neq n_2 \end{cases} \quad (2.46)$$

Then, multiply the term $\left(\frac{1}{2\pi j} \right)^2$ on both sides of Eq. (2.46)

$$\left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X(z_1, z_2) z_1^{n_1-1} z_2^{n_2-1} dz_1 dz_2 = \left(\frac{1}{2\pi j} \right)^2 \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) \oint_{c_1} \oint_{c_2} z_1^{n_1-1-k_1} z_2^{n_2-1-k_2} dz_1 dz_2$$

So using Cauchy integral theorem to the above equation, we get

$$\left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X(z_1, z_2) z_1^{n_1-1} z_2^{n_2-1} dz_1 dz_2 = x(n_1, n_2) \quad (2.47)$$

(i) Product of two sequences

If

$$x_1(n_1, n_2) \xrightarrow{Z} X_1(z_1, z_2)$$

$$x_2(n_1, n_2) \xrightarrow{Z} X_2(z_1, z_2)$$

Then, the 2D Z-transform of product of these two sequences is given by,

$$\begin{aligned} x(n_1, n_2) &= x_1(n_1, n_2) \times x_2(n_1, n_2) \xrightarrow{Z} X(z_1, z_2) \\ &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) X_2 \left(\frac{z_1}{u}, \frac{z_2}{v} \right) u^{-1} v^{-1} du dv \end{aligned}$$

Proof The Z-transform of $x(n_1, n_2)$ is

$$\begin{aligned} X(z_1, z_2) &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_1(n_1, n_2) x_2(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \end{aligned} \quad (2.48)$$

The inverse Z-transform of $x_1(n_1, n_2)$ is

$$x_1(n_1, n_2) = \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) u^{n_1-1} v^{n_2-1} du dv \quad (2.49)$$

Substituting Eq. (2.49) in (2.48), we get

$$\begin{aligned} X(z_1, z_2) &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) u^{n_1-1} v^{n_2-1} du dv x_2(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \\ &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_2(n_1, n_2) u^{n_1-1} v^{n_2-1} z_1^{-n_1} z_2^{-n_2} du dv \\ &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_2(n_1, n_2) \left(\frac{z_1}{u} \right)^{-n_1} u^{-1} \left(\frac{z_2}{v} \right)^{-n_2} v^{-1} du dv \\ X(z_1, z_2) &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_2(n_1, n_2) \left(\frac{z_1}{u} \right)^{-n_1} \left(\frac{z_2}{v} \right)^{-n_2} u^{-1} v^{-1} du dv \end{aligned} \quad (2.50)$$

(ii) Parseval's Theorem If $x_1(n_1, n_2)$ and $x_2(n_1, n_2)$ are complex-valued sequence then the Parseval's relation is given by

$$\sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_1(n_1, n_2) x_2^*(n_1, n_2) = \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) X_2^* \left(\frac{1}{u^*}, \frac{1}{v^*} \right) u^{-1} v^{-1} du dv \quad (2.51)$$

Proof The inverse Z-transform

$$x_1(n_1, n_2) = \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) u^{n_1-1} v^{n_2-1} du dv \quad (2.52)$$

Applying the Eq. (2.52) into Eq. (2.51), we get

$$\begin{aligned} \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_1(n_1, n_2) x_2^*(n_1, n_2) &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) u^{n_1-1} v^{n_2-1} du dv x_2^*(n_1, n_2) \\ &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_2^*(n_1, n_2) u^{n_1-1} v^{n_2-1} du dv \\ &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_2^*(n_1, n_2) \left(\frac{1}{u} \right)^{-n_1} \left(\frac{1}{v} \right)^{-n_2} u^{-1} v^{-1} du dv \end{aligned}$$

Now we apply the conjugation property,

$$\begin{aligned} &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) X_2^* \left(\frac{1}{u^*}, \frac{1}{v^*} \right) u^{-1} v^{-1} du dv \\ \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_1(n_1, n_2) x_2^*(n_1, n_2) &= \left(\frac{1}{2\pi j} \right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) X_2^* \left(\frac{1}{u^*}, \frac{1}{v^*} \right) u^{-1} v^{-1} du dv \end{aligned}$$

This shows that energy in spatial domain = energy in the frequency domain.

The properties of 2D Z-transforms are summarised in Table 2.1.

2.9 2D DIGITAL FILTER

2D digital filters are central to many image-processing applications such as image enhancement, image deblurring, target matching, etc. The 2D filters can be broadly classified into two types: (i) Finite Impulse Response (FIR), filter and (ii) Infinite Impulse Response (IIR) filter. FIR digital filters of the non-recursive type can be realised by means of a simple hardware or software. IIR digital filters of the recursive type, can have higher efficiency than the FIR filters, but they are not frequently used for image processing due to the difficulties of factorising two variable transfer functions and the resulting problems of designing and implementing them with sufficient accuracy and stability. The choice between FIR filters and an IIR filters depends upon the application, although most applications are probably better served by FIR designs.

2.9.1 2D FIR Filter through Frequency Sampling

An FIR filter is one whose impulse response possesses only a finite number of non-zero samples. Thus, the impulse response of an FIR filter is absolutely summable which implies that FIR filters are mostly stable.

Table 2.1 Properties of 2D Z-transforms

Property	Sequence	Z-transform
Linearity	$ax(n_1, n_2) + by(n_1, n_2)$	$aX[z_1, z_2] + bY[z_1, z_2]$
Convolution	$x(n_1, n_2) * y(n_1, n_2)$	$X[z_1, z_2] * Y[z_1, z_2]$
Shifting	$x(n_1 - m_1, n_2 - m_2)$	$z_1^{-m_1} z_2^{-m_2} X[z_1, z_2]$
Conjugation	$x^*(n_1, n_2)$	$X^*[z_1^*, z_2^*]$
Reflection	$x(-n_1, -n_2)$	$X[z_1^{-1}, z_2^{-2}]$
Differentiation	$-n_1 n_2 x(n_1, n_2)$	$z_1 z_2 \frac{\partial^2 X(z_1, z_2)}{\partial z_1 \partial z_2}$
Modulation	$\alpha^{-n_1} \beta^{-n_2} x(n_1, n_2)$	$X(\alpha z_1, \beta z_2)$
Multiplication	$x_1(n_1, n_2) \times x_2(n_1, n_2)$	$\left(\frac{1}{2\pi j}\right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) X_2\left(\frac{z_1}{u}, \frac{z_2}{v}\right) u^{-1} v^{-1} du dv$
Parseval's theorem	$\sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x_1(n_1, n_2) x_2^*(n_1, n_2)$	$\left(\frac{1}{2\pi j}\right)^2 \oint_{c_1} \oint_{c_2} X_1(u, v) X_2^*\left(\frac{1}{u^*}, \frac{1}{v^*}\right) u^{-1} v^{-1} du dv$

Let $h(n_1, n_2)$ represent the impulse response of a 2D filter where $-\infty < n_1, n_2 < \infty$. On taking the Z-transform of the impulse response, we get $H(z_1, z_2)$.

$$H(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} h(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \quad (2.53)$$

If the impulse response is absolutely summable then the resulting filter is stable which is given by

$$\sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} |h(n_1, n_2)| < \infty \quad (2.54)$$

The impulse response of an FIR filter is a finite sequence which is defined from $0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1$. Then Eq. (2.53) will be reduced to

$$H(z_1, z_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} h(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \quad (2.55)$$

The frequency response of the filter is obtained by substituting $z_1 = e^{j\omega_1}$ and $z_2 = e^{j\omega_2}$ in Eq. (2.55):

$$H\left(e^{j\omega_1}, e^{j\omega_2}\right) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} h(n_1, n_2) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2} \quad (2.56)$$

The Discrete Fourier Transform (DFT) of the filter is obtained by evaluating Eq. (2.56) at a discrete set of values

$$\omega_{k1} = k_1 \left(\frac{2\pi}{N_1} \right), k_1 = 0, 1, \dots, N_1 - 1 \quad (2.57)$$

$$\omega_{k2} = k_2 \left(\frac{2\pi}{N_2} \right), k_2 = 0, 1, \dots, N_2 - 1 \quad (2.58)$$

$$H(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} h(n_1, n_2) e^{-j2\pi \left(\frac{k_1 n_1}{N_1} \right)} e^{-j2\pi \left(\frac{k_2 n_2}{N_2} \right)} \quad (2.59)$$

The inverse 2D DFT is given by

$$h(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} H(k_1, k_2) e^{j2\pi \left(\frac{k_1 n_1}{N_1} \right)} e^{j2\pi \left(\frac{k_2 n_2}{N_2} \right)} \quad (2.60)$$

Substituting Eq. (2.60) in Eq. (2.56), we get

$$H(e^{j\omega_1}, e^{j\omega_2}) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \frac{1}{N_1 N_2} \left[\sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} H(k_1, k_2) e^{j2\pi \left(\frac{k_1 n_1}{N_1} + \frac{k_2 n_2}{N_2} \right)} \right] e^{-j(\omega_1 n_1 + \omega_2 n_2)} \quad (2.61)$$

Equation (2.61) represents the main basis for designing 2D digital filters with the sampling method. However the frequency response assumes the values $H(k_1, k_2)$ in the fixed points, while it presents fluctuations in the intervals among these points. Therefore, it is useful to modify the sampled impulse response in order to reduce the fluctuations. This is in general obtained through linear programming methods.

2.9.2 2D FIR Filter through Windowing Technique

It can be observed from Eq. (2.56) that if certain symmetries are maintained in the filter impulse response coefficients, the filter's frequency response becomes purely real. This fact is represented by

$$H(k_1, k_2) = H^*(k_1, k_2) \quad (2.62)$$

The filter which obeys Eq. (2.62) is generally termed a *zero-phase filter*. Strictly speaking, not all filters with real frequency responses are necessarily zero-phase filters, since $H(k_1, k_2)$ can become negative. In practice, the frequency regions for which $H(k_1, k_2)$ is negative and typically corresponds to the stop-band regions, and a phase of 180° in the stop-band regions, has little significance.

The equivalent spatial domain representation of Eq. (2.62) is given in Eq. (2.63)

$$h(n_1, n_2) = h^*(-n_1, -n_2) \quad (2.63)$$

If we consider only the real part of $h(n_1, n_2)$ the above equation reduces to

$$h(n_1, n_2) = h(-n_1, -n_2) \quad (2.64)$$

The above equation states that the impulse response of a zero-phase filter is symmetric with respect to the origin. The symmetric condition is given by

$$h(n_1, n_2) = h(N_1 - 1 - n_1, n_2) = h(n_1, N_2 - 1 - n_2) \quad (2.65)$$

Equation (2.65) is applied to Eq. (2.56) by assuming N_1, N_2 to be even results in

$$H(e^{j\omega_1}, e^{j\omega_2}) = e^{-j\omega_1((N_1-1)/2)} e^{-j\omega_2((N_2-1)/2)} \sum_{n_1=0}^{(N_1/2)-1} \sum_{n_2=0}^{(N_2/2)-1} h(n_1, n_2) \times \left[\cos\left[\left(\frac{N_1-1}{2}\right) - n_1\right] \omega_1 \right] \left[\cos\left[\left(\frac{N_2-1}{2}\right) - n_2\right] \omega_2 \right]$$

which is purely real, except for the linear phase terms outside the summation.

The problem of designing the digital filter in the 2D frequency domain is then connected to the evaluation of the coefficient matrix $h(n_1, n_2)$ in such a way that the obtained frequency response satisfies the required characteristics. In the windowing method of FIR filter design, the impulse response $h(n_1, n_2)$ is multiplied by the samples of $w(n_1, n_2)$ of a suitable function having zero value in the tails of the truncation and high values in the central region. The obtained frequency response of the digital filter is the convolution between $H(k_1, k_2)$ and $W(k_1, k_2)$.

Windows with frequency response $W(k_1, k_2)$ is chosen in such a way that they have narrow main lobes and low side lobes. This leads to filters with sharp transition bands and low ripples. The 2D windows can be obtained directly from 1D windows. The windows can be broadly classified into two types: (a) separable window, and (b) circularly symmetric window.

The separable window is given by

$$w(n_1, n_2) = w_1(n_1)w_2(n_2) \quad (2.66)$$

where $w_1(n_1)$ and $w_2(n_2)$ are 1D windows. The size of the window is denoted by $(N_1 \times N_2)$. For non-causal filters, the support region for the window is $-N_1 \leq n_1 \leq N_1, -N_2 \leq n_2 \leq N_2$. This is illustrated in Fig. 2.17.

The frequency response of the 2D separable ideal low-pass filter is given by

$$H_{LPF}(k_1, k_2) = \begin{cases} 1, & |k_1| \leq k_{c1} \text{ and } |k_2| \leq k_{c2} \\ 0, & k_{c1} < |k_1| \leq \pi \text{ and } k_{c2} < |k_2| \leq \pi \end{cases} \quad (2.67)$$

The corresponding impulse response is given by

$$h_{LPF}(n_1, n_2) = \frac{\sin k_{c1} n_1}{\pi n_1} \frac{\sin k_{c2} n_2}{\pi n_2} \quad (2.68)$$

The 2D circularly symmetric window is obtained by rotating the analog 1D window function $w(t)$ and then sampling it. The Huang's method of 2D window design is given as follows

$$w(t_1, t_2) = w(t) \Big|_{t=\sqrt{t_1^2 + t_2^2}} \quad (2.69)$$

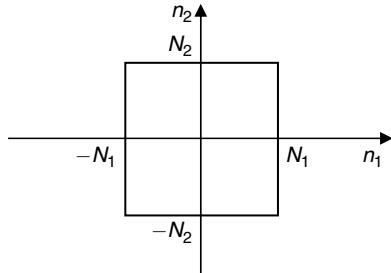


Fig. 2.17 Region of support of a separable window

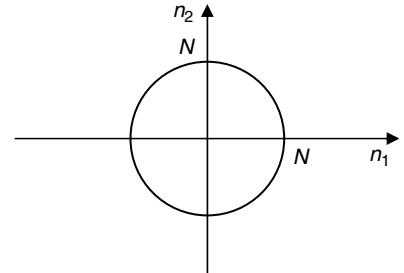


Fig. 2.18 Region of support of a circularly symmetric window

$$w(n_1, n_2) = w(t_1, t_2) |_{n_1 = t_1, n_2 = t_2} \quad (2.70)$$

The size of the window is denoted by its radius N such that $\sqrt{n_1^2 + n_2^2} \leq N$. The region of support of this circularly symmetric window is shown in Fig. 2.18.

The frequency response of the 2D circularly symmetric ideal low-pass filter is given by

$$H_{\text{LPF}}(k_1, k_2) = \begin{cases} 1, & \sqrt{k_1^2 + k_2^2} < k_c \\ 0, & \sqrt{k_1^2 + k_2^2} \geq k_c, \text{ and } |k_1| \leq \pi \text{ and } |k_2| \leq \pi \end{cases} \quad (2.71)$$

The impulse response is given by

$$h_{\text{LPF}}(n_1, n_2) = \frac{k_c}{2\pi\sqrt{n_1^2 + n_2^2}} J_1\left(k_c \sqrt{n_1^2 + n_2^2}\right) \quad (2.72)$$

where $J_1(\cdot)$ represents the Bessel function of the first kind and first order.

2.9.3 2D Recursive Filter

The use of recursive filters instead of non-recursive filters has the potential of saving computation time. A two-dimensional digital recursive filter is characterised by the two-dimensional Z-transform:

$$H(z_1, z_2) = \frac{\sum_{m=0}^p \sum_{n=0}^q a_{mn} z_1^m z_2^n}{\sum_{m=0}^p \sum_{n=0}^q b_{mn} z_1^m z_2^n} \quad (2.73)$$

where a_{mn} and b_{mn} are constants. The degrees of the numerator and the denominator polynomials do not have to be equal. The variables z_1 and z_2 are given by

$$z_1 = e^{-s_1 A} \quad (2.74)$$

$$z_2 = e^{-s_2 B} \quad (2.75)$$

where s_1 and s_2 are horizontal and vertical complex spatial frequency variables, and A and B are constants. If $f(m, n)$ and $g(m, n)$ are the input and the output of the filter then the spatial domain difference equation corresponding to the transfer function $H(z_1, z_2)$ is given by

$$\sum_{m=0}^p \sum_{n=0}^q b_{mn} g(M-m, N-n) = \sum_{m=0}^p \sum_{n=0}^q a_{mn} f(M-m, N-n) \quad (2.76)$$

The recursive filter that recourses in $(+m, +n)$ directions can be considered as causal. The two major problems in the design of a recursive filter are approximation and stability. The approximation problem consists of determining the coefficients a_{mn} and b_{mn} so that $H(e^{-juA}, e^{-jvB})$ approximates a given frequency response $H_1(ju, jv)$, where u and v are horizontal and vertical spatial frequencies. The filter $H(z_1, z_2)$ is expanded into a power series

$$H(z_1, z_2) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} h_{mn} z_1^m z_2^n \quad (2.77)$$

The impulse response of the filter should satisfy the condition

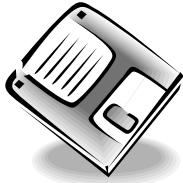
$$\sum_{m=0}^{\infty} \sum_{n=0}^{\infty} |h_{mn}| < \infty \quad (2.78)$$

If the filter is unstable, any noise will propagate through the output and it will be amplified.

Shanks Stability Criterion The Shanks stability theorem gives the stability conditions in the frequency domain. The Shanks stability theorem is given below

Theorem A causal recursive filter with the Z-transform $H(z_1, z_2) = \frac{A(z_1, z_2)}{B(z_1, z_2)}$, where A and B are polynomials in z_1 and z_2 , is stable if and only if there are no values of z_1 and z_2 such that $B(z_1, z_2) = 0$ and $|z_1| \leq 1$, $|z_2| \leq 1$.

Summary



- Signals are variables that carry information. An example of a 2D signal is a still image.
- A 2D sequence is periodic if it repeats itself in a regularly spaced interval.
- A 2D sequence $x(n_1, n_2)$ is separable if it can be represented as $x(n_1, n_2) = x_1(n_1)x_2(n_2)$.
- A system can be viewed as an operation or a set of operations performed on the input signal to produce an output signal.
- A system is a shift-invariant system if its input–output characteristics do not change with time.
- A system is linear if it obeys the superposition theorem. The principle of superposition requires that the response of the system to a weighted sum of signals should be equal to the corresponding weighted sum of the output of the system to each of the individual input signals.
- Linear Shift Invariant (LSI) systems are uniquely represented by their 2D impulse response.
- A system is static or memoryless if its output at any instant depends at most on the input sample but not on the past and future samples of the input.
- A 2D LSI system is BIBO stable if its impulse response is absolutely summable.
- A Z-transform is widely used to analyse a 2D system. The Z-transform of a 2D sequence $x(n_1, n_2)$ is given by

$$X(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1, n_2) z_1^{-n_1} z_2^{-n_2}$$

- Filters can be considered as systems that process the input signal and give an output signal. Filters can be broadly classified into (a) FIR filters, and (b) IIR filters. FIR filters are more popular since they exhibit linear phase characteristics.
- 2D FIR filters can be designed using windowing technique, frequency sampling technique and the optimal method.
- 2D IIR filters are recursive in nature. The stability of a 2D IIR filter is not always guaranteed.

Review Questions

- Given a discrete sequence $f(m, n) = (m + n)^2$, find $f(m, n) \delta(m - 1, n - 1)$.

$$\delta(m - 1, n - 1) = \begin{cases} 1 & m = 1, n = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$f(m, n) \delta(m - 1, n - 1) = \begin{cases} (m + n)^2 & m = 1, n = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$f(m, n) \delta(m - 1, n - 1) = \begin{cases} (1+1)^2 & m = 1, n = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$f(m, n) \delta(m - 1, n - 1) = \begin{cases} 4 & m = 1, n = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$f(m, n) \delta(m - 1, n - 1) = 4 \delta(m - 1, n - 1)$$

2. Determine whether the 2D signal $x(n_1, n_2) = \cos\left(\frac{\pi}{2}n_1 + \pi n_2\right)$ is periodic or not. If periodic, determine the fundamental period.

For the 2D signal to be periodic, it should satisfy the condition:

$$\frac{\omega_1}{2\pi} = \frac{\omega_2}{2\pi} = \text{ratio of two integers}$$

In this problem, $\omega_1 = \frac{\pi}{2}$ and $\omega_2 = \pi$. This implies

$\frac{\omega_1}{2\pi} = \frac{\pi}{2 \times 2\pi} = \frac{1}{4}$ and $\frac{\omega_2}{2\pi} = \frac{\pi}{2\pi} = \frac{1}{2}$ which are rational numbers. Hence the signal is periodic and the fundamental period is 4×2 .

3. Determine whether the system described by the following input-output relation is shift invariant or not.

$$y(n_1, n_2) = x(2n_1, 2n_2)$$

From the input-output relation, the system is identified as a down-sampler. That is, the number of samples in the output is half the number of samples in the input.

The given system is $y(n_1, n_2) = x(2n_1, 2n_2)$ (2.53)

The time shift in the input by a factor of k is given by

$$y(n_1, n_2) = x(2n_1 - k, 2n_2 - k) \quad (2.54)$$

Now the time shift in the output is obtained by replacing n_1, n_2 by $n_1 - k$ and $n_2 - k$.

$$y(n_1, n_2) = x(2(n_1 - k), 2(n_2 - k)) = x(2n_1 - 2k, 2n_2 - 2k) \quad (2.55)$$

Equations (2.54) and (2.55) are not same. This shows that time shift in the input is not equal to the time shift in the output. Hence, the system 'down-sampler' is a shift-variant system.

4. Determine whether the system described by the following input-output relation is shift invariant or not.

$$y(n_1, n_2) = x\left(\frac{n_1}{2}, \frac{n_2}{2}\right)$$

The given system is identified as an interpolator. The output sample is twice the number of the input sample.

The given system is $y(n_1, n_2) = x\left(\frac{n_1}{2}, \frac{n_2}{2}\right)$ (2.56)

Time shift in the input by a factor of k is given by

$$y(n_1, n_2) = x\left(\frac{n_1}{2} - k, \frac{n_2}{2} - k\right) \quad (2.57)$$

Now the time shift in the output by the same factor k is

$$y(n_1, n_2) = x\left(\frac{n_1 - k}{2}, \frac{n_2 - k}{2}\right) \quad (2.58)$$

Equations (2.57) and (2.58) are not the same equation. This shows that time shift in the input is not equal to the time shift in the output and hence, the system is shift variant.

- 5. Determine the 2D Z-transform of $x(n_1, n_2) = \left(\frac{1}{2}\right)^{n_1} \left(\frac{1}{3}\right)^{n_2} u(n_1, n_2)$.**

$$\text{The 2D Z-transform is given by } X[z_1, z_2] = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} \left(\frac{1}{2}\right)^{n_1} \left(\frac{1}{3}\right)^{n_2} z_1^{-n_1} z_2^{-n_2}$$

$$= \sum_{n_1=0}^{\infty} \left(\frac{1}{2}\right)^{n_1} z_1^{-n_1} \sum_{n_2=0}^{\infty} \left(\frac{1}{3}\right)^{n_2} z_2^{-n_2} = \sum_{n_1=0}^{\infty} \left(\frac{1}{2} z_1^{-1}\right)^{n_1} \sum_{n_2=0}^{\infty} \left(\frac{1}{3} z_2^{-1}\right)^{n_2}$$

$$= \frac{1}{1 - \frac{1}{2} z_1^{-1}} \times \frac{1}{1 - \frac{1}{3} z_2^{-1}}$$

$$X[z_1, z_2] = \frac{2z_1}{2z_1 - 1} \times \frac{3z_2}{3z_2 - 1}$$

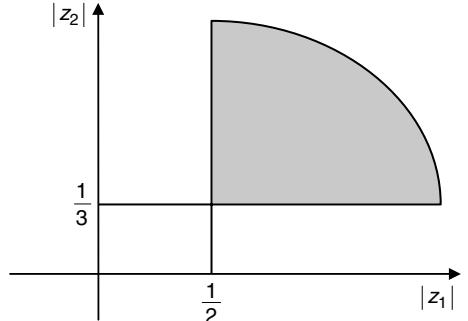


Fig. 2.19 ROC

The ROC is $|z_1| > \frac{1}{2}$ and $|z_2| > \frac{1}{3}$. The ROC is shown in Fig. 2.19.

- 6. Find the 2D Z-transform of the sequence**

$$x(n_1, n_2) = \begin{cases} a^{n_1+4n_2} & : n_1, n_2 \geq 0 \\ 0 & : \text{else} \end{cases}$$

$$\text{The 2D Z-transform is given by } X[z_1, z_2] = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} a^{n_1+4n_2} z_1^{-n_1} z_2^{-n_2}$$

$$= \sum_{n_1=0}^{\infty} \left(a z_1^{-1}\right)^{n_1} \sum_{n_2=0}^{\infty} \left(a^4 z_2^{-1}\right)^{n_2} = \frac{1}{1 - az_1^{-1}} \times \frac{1}{1 - a^4 z_2^{-1}}$$

$$X[z_1, z_2] = \frac{z_1}{z_1 - a} \times \frac{z_2}{z_2 - a^4}$$

For convergence, the value of $|a| < 1$.

7. Determine the 2D Z-transform of the sequence

$$x(n_1, n_2) = \begin{cases} a^{n_1} (a-b) \delta(n_1 - 2n_2) & : n_1 \geq 0 \\ 0 & : \text{else} \end{cases}$$

The 2D Z-transform is given by

$$X[z_1, z_2] = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} a^{n_1} (a-b) \delta(n_1 - 2n_2) z_1^{-n_1} z_2^{-n_2}$$

The sequence has a value only at $n_1 = 2n_2$. The above equation can be modified as

$$\begin{aligned} X[z_1, z_2] &= \sum_{n_2=0}^{\infty} a^{2n_2} (a-b)^{n_2} z_1^{-2n_2} z_2^{-n_2} \\ &= \sum_{n_2=0}^{\infty} [a^2(a-b)z_1^{-2}z_2^{-1}]^{n_2} \\ X[z_1, z_2] &= \frac{1}{1 - a^2(a-b)z_1^{-2}z_2^{-1}} \end{aligned}$$

For convergence, $|a^2(a-b)| < 1$.

- 8. The frequency response of the filter is given by $H(e^{j\omega_1}, e^{j\omega_2}) = 1 - \cos(\omega_1 + \omega_2)\cos(\omega_1 - \omega_2)$. Obtain the expression of 2D Z-transform of this filter.**

The given expression is

$$H(e^{j\omega_1}, e^{j\omega_2}) = 1 - \cos(\omega_1 + \omega_2)\cos(\omega_1 - \omega_2) \quad (2.59)$$

$$\text{But the expression of } \cos\theta \text{ is } \cos\theta = \frac{e^{j\theta} + e^{-j\theta}}{2} \quad (2.60)$$

Substituting the expression of $\cos\theta$ in Eq. (2.59), we get

$$H(e^{j\omega_1}, e^{j\omega_2}) = 1 - \left[\frac{e^{j(\omega_1+\omega_2)} + e^{-j(\omega_1+\omega_2)}}{2} \times \frac{e^{j(\omega_1-\omega_2)} + e^{-j(\omega_1-\omega_2)}}{2} \right] \quad (2.60)$$

The above equation can be simplified as

$$H(e^{j\omega_1}, e^{j\omega_2}) = 1 - \frac{e^{j2\omega_1} + e^{j2\omega_2} + e^{-j2\omega_1} + e^{-j2\omega_2}}{4} \quad (2.61)$$

If $z = e^{j\omega_1}$ then $z^2 = e^{j2\omega_1}$. Substituting this in Eq. (2.61), we get

$$H(z_1, z_2) = 1 - \frac{z_1^2 + z_2^2 + z_1^{-2} + z_2^{-2}}{4}$$

9. Mention the steps involved in the design of a 2D FIR filter using the windowing technique.

The steps involved in the design of a 2D FIR filter using the windowing technique are summarised below:

- (i) First, the desired frequency response should be known to the designer which is generally represented as $H_d(k_1, k_2)$
- (ii) Ideal frequency-domain filters result in infinite impulse response.
- (iii) Limit the extent of the spatial response with a window function $w(n_1, n_2)$ which is represented as

$$h(n_1, n_2) = h_d(n_1, n_2)w(n_1, n_2)$$

- (iv) The multiplication in the spatial domain is equivalent to convolution in the frequency domain

$$H(k_1, k_2) = H_d(k_1, k_2) * W(k_1, k_2)$$

Problems

2.1 Sketch the following sequences

- (i) $x(n_1, n_2) = \delta(n_1 - 1, n_2 - 1)$
- (ii) $x(n_1, n_2) = \delta(n_1 - 1, n_2 - 1) + \delta(n_1 - 2, n_2 - 1)$
- (iii) $x(n_1, n_2) = u(n_1, n_2) - u(n_1 - 1, n_2 - 1)$
- (iv) $x(n_1, n_2) = \begin{cases} 1 & n_1 \\ 3 & n_2 \end{cases} u(n_1, n_2)$

2.2 Determine whether the following 2D signals are periodic or not.

$$(i) x(n_1, n_2) = \cos\left(\left(\frac{\pi}{3}\right)n_1 + \left(\frac{\pi}{6}\right)n_2\right)$$

$$(ii) x(n_1, n_2) = \cos(n_1 + (\pi)n_2)$$

2.3 For each of the following systems, determine whether or not the system is (a) linear, or (b) shift invariant.

- (i) $y(n_1, n_2) = x(n_1, n_2)$
- (ii) $y(n_1, n_2) = x(n_1, n_2) + 5$
- (iii) $y(n_1, n_2) = x(-n_1, -n_2)$

2.4 Determine the 2D Z-transform of the following signals and the corresponding ROC:

- (i) $x(n_1, n_2) = \delta(n_1 - n_2)$
- (ii) $x(n_1, n_2) = \delta(n_1, n_2 - 1)$
- (iii) $x(n_1, n_2) = \delta(n_1 - 1, n_2)$
- (iv) $x(n_1, n_2) = u(n_1, n_2) - u(n_1 - 3, n_2 - 3)$

2.5 Determine the frequency response of the system:

$$h(n_1, n_2) = \begin{cases} \frac{1}{4}, & (n_1, n_2) = (0, 0), (0, 1), (1, 0) \text{ and } (1, 1) \\ 0, & \text{otherwise} \end{cases}$$

82 Digital Image Processing

2.6 The impulse response of the system is given by

$$h(n_1, n_2) = \delta(n_1, n_2) + 2\delta(n_1 - 1, n_2) + 2\delta(n_1 + 1, n_2) + 4\delta(n_1, n_2 - 1) + 4\delta(n_1, n_2 + 1)$$

2.7 Determine the 2D Z-transform for the two filters given below by assuming the origin at the centre.

$$H_1 = \frac{1}{24} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 12 & -2 \\ 1 & -2 & 1 \end{bmatrix}, \quad H_2 = \frac{1}{24} \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & 1 \end{bmatrix}$$

2.8 A 2D LSI system has an impulse response $h(n_1, n_2) = \left(\frac{1}{2}\right)^{n_1} \left(\frac{1}{3}\right)^{n_2} u(n_1, n_2)$. Determine the system response $y(n_1, n_2)$ for the input $x(n_1, n_2) = \delta(n_1, n_2)$.

2.9 Determine the Inverse Z-transform of each of the following system functions by assuming that the ROC includes the unit surface.

$$(i) \quad H(z_1, z_2) = \frac{1}{1 - \frac{1}{2}z_1^{-1}z_2^{-1}}$$

$$(ii) \quad H(z_1, z_2) = \frac{1}{\left(1 - \frac{1}{2}z_1^{-1}\right)\left(1 - \frac{1}{4}z_2^{-1}\right)}$$

2.10 The impulse response of the system is given by $h(n_1, n_2) = \left(\frac{1}{4}\right)^{n_1} \left(\frac{1}{3}\right)^{n_2} u(n_1, n_2)$. Determine the frequency response of the system.

2.11 Under what conditions is the following statement true?

$$T[A + B] = T[A] + T[B]$$

where A and B are arrays and T is a vector of suitable length, all of integers.

2.12 Find the poles and zeros of the following system

$$(i) \quad H(z_1, z_2) = \frac{z_1 z_2}{z_1 z_2 - 1}$$

$$(ii) \quad H(z_1, z_2) = \frac{1 - z_1^{-1}z_2^{-1}}{2 - z_1^{-1} - z_2^{-1}}$$

References

Books

1. D E Dudgeon and R M Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983
2. J. Lim, *2D Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990
3. T S Huang, editor, *Two-Dimensional Digital Signal Processing*, Topics in Applied Physics, vol. 42 and vol. 43, Springer-Verlag, 1981
4. Spyros G Tzafestas (Ed), *Multidimensional Systems Techniques and Applications*, Marcel Dekker, Inc., 1986

5. S K Mitra and M P Ekstrom, editors, *Two-Dimensional Digital Signal Processing*, Dowden, Hutchinson, and Ross, 1978
6. John W Woods, *Multidimensional Signal, Image, and Video Processing and Coding*, Academic Press, 2006
7. W Lu and A Antoniou, *Two-dimensional Digital Filters*, Marcel Dekker, New York, 1992

Journal Papers

1. T S Huang, *2-D Windows*, IEEE Transactions Audio Electro acoustics, AU-20, pp. 88–90, March 1972
2. T Bose and M Q Chen, *Design of Two-dimensional Digital Filters in the Spatial Domain*, IEEE Transactions on Signal Processing 42, pp. 1464–1469, March 1993
3. J V Hu and L R Rabiner, *Design Techniques for Two-Dimensional Digital Filters*, IEEE Trans. Audio Electroacoustics, vol. 20, pp. 249–257, October 1972
4. J L Shanks, S Treitel and J H Justice, *Stability and Synthesis of Two-dimensional Recursive Filters*, IEEE Trans. Audio Electroacous., vol. 20, pp. 115–128, June 1972

Web Resources

1. Professor Brian L. Evans lectures on 2D signals and systems is a very good material, and the reader should go through the website at least once before completing the course on digital image processing
<http://users.ece.uvic.ca/~bevans/courses/ee381k/lectures/>
2. Professor Lina J. Karam's well-organised teaching material related *Digital Image Processing and Compression*:
<http://www.eas.asu.edu/~karam/eee508/>
3. Joan Lasenby's website gives useful information on two-dimensional systems:
<http://www-sigproc.eng.cam.ac.uk/%7Ejl/imageproc/index.html>

3

Learning Objectives

This chapter focuses on two-dimensional convolution and correlation. Convolution and correlation are the basic operations of most image-analysis systems. Step by step approaches to two-dimensional convolution and correlation through the graphical method, matrix method and Z-transform method are given through suitable examples in this chapter. After completing the chapter, the reader should be familiar with the following concepts:

Computation of 2D convolution and correlation through graphical method

Computation of 2D convolution and correlation through Z-transform

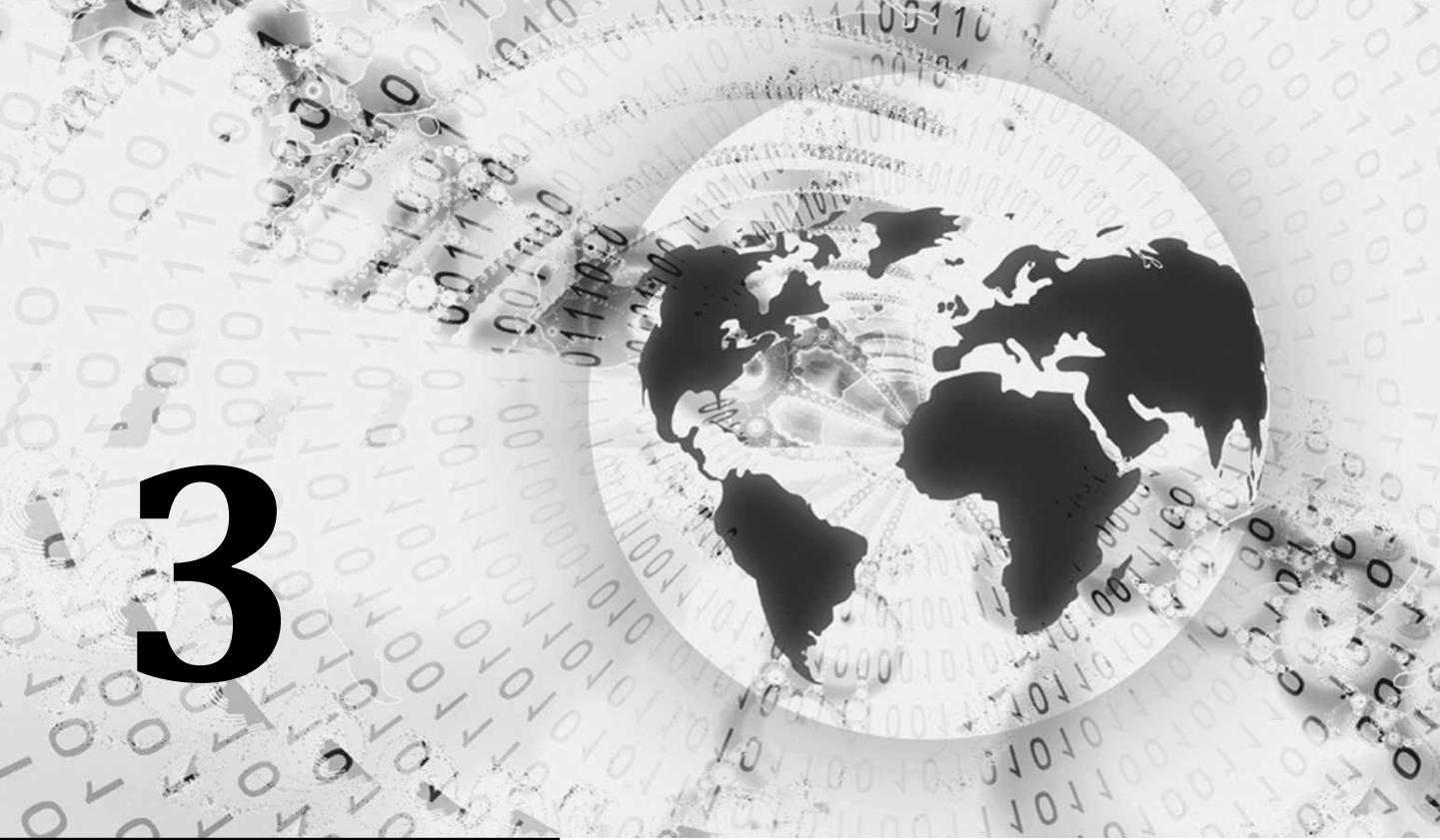
Determination of 2D convolution and correlation through matrix method

Significance of 2D convolution and correlation

Convolution and Correlation

3.1 INTRODUCTION

Convolution and correlation operations are basically used to extract information from images. Convolution and correlation are basically linear and shift-invariant operations. The term *linear* indicates that a pixel is replaced by the linear combination of its neighbours. The term *shift invariant* indicates that the same operation is performed at every point in the image.



Convolution is basically a mathematical operation where each value in the output is expressed as the sum of values in the input multiplied by a set of weighting coefficients. Depending upon the weighting coefficients, convolution operation is used to perform spatial domain low-pass and high-pass filtering of the image. An image can be either smoothed or sharpened by convolving the image with respect to low-pass and high-pass filter mask respectively. This principle is widely used in the construction of the image pyramid. Convolution has a multitude of applications including image filtering, image enhancement, image restoration, feature extraction and template matching.

The two-dimensional discrete convolution between two signals $x[n_1, n_2]$ and $h[n_1, n_2]$ is given by

$$y[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2)h(n_1 - k_1, n_2 - k_2) \quad (3.1)$$

2D convolution can be represented as a sequence of two 1D convolutions only if the signals are separable. Convolution can be performed either in the spatial domain or in the frequency domain.

3.2 2D CONVOLUTION THROUGH GRAPHICAL METHOD

In this section, the two-dimensional convolution is performed through graphical analysis. The basic operations involved in 2D convolution are folding, shifting and addition operations. The step-by-step approach to 2D convolution through the graphical method is illustrated below:

Example 3.1 The input matrix $x(m, n)$ and $h(m, n)$. Perform the linear convolution between these two matrices.

$$x(m, n) = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Solution The indices of the given input matrices are given below:

$$x(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 4 & 5 & 6 \\ (1,0) & (1,1) & (1,2) \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} (0,0) \\ 1 \\ (1,0) \\ 1 \\ (2,0) \\ 1 \end{pmatrix}$$

Determining the dimension of the resultant matrix The dimension of the resultant matrix depends upon the dimension of the input matrices, $x(m, n)$ and $h(m, n)$. The dimension of $x(m, n)$ is 2×3 (two rows and three columns). The dimension of $h(m, n)$ is 3×1 (three rows and one column). The resultant matrix dimension is calculated as

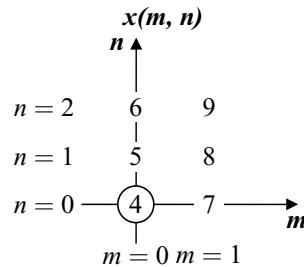
$$\text{Dimension of resultant matrix} = \begin{cases} (\text{No. of rows of } x(m, n) + \text{No. of rows of } h(m, n) - 1) \\ \times \\ (\text{No. of columns of } x(m, n) + \text{No. of columns of } h(m, n) - 1) \end{cases}$$

$$\text{Dimension of resultant matrix} = (2 + 3 - 1) \times (3 + 1 - 1) = 4 \times 3$$

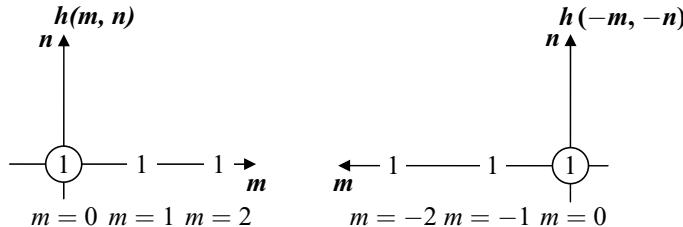
The resultant matrix $y(m, n)$ of size 4×3 is given as

$$y(m, n) = \begin{pmatrix} y(0, 0) & y(0, 1) & y(0, 2) \\ y(1, 0) & y(1, 1) & y(1, 2) \\ y(2, 0) & y(2, 1) & y(2, 2) \\ y(3, 0) & y(3, 1) & y(3, 2) \end{pmatrix}$$

The graphical representation of $x(m, n)$ is shown below:

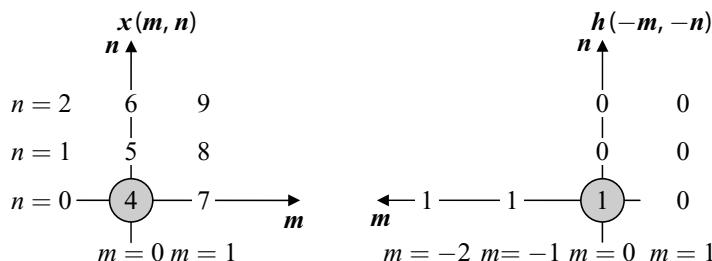


Here, the encircled element is the origin. The graphical representation of $h(m, n)$ and its folded version, $h(-m, -n)$ are given below:



1. To determine the value of $y(0, 0)$

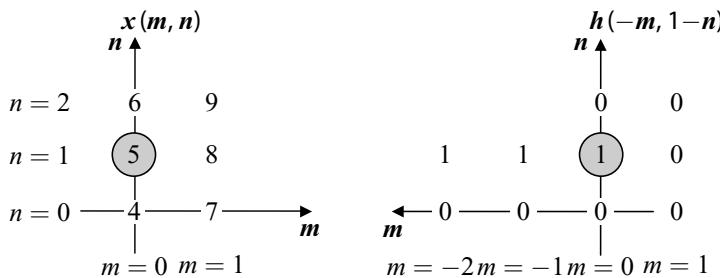
The common values of $x(m, n)$ and $h(-m, -n)$ are multiplied and then added to get the value of $y(0, 0)$. The shaded circle indicates the common area between the two signals.



The value $y(0, 0)$ is obtained as $y(0, 0) = 4 \times 1 = 4$.

2. To determine the value of $y(0, 1)$

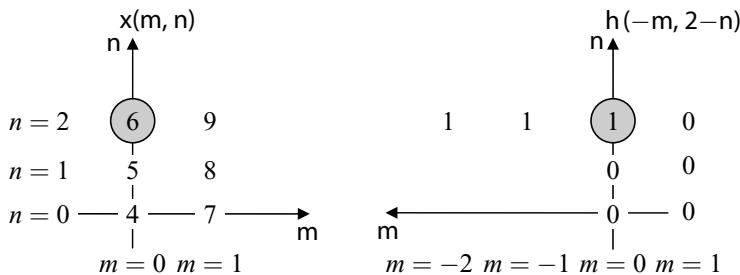
Now the signal $h(-m, -n)$ is shifted along the 'n' axis by one unit to get $h(-m, 1-n)$ and $x(m, n)$ is unaltered. The common value between the two signals is multiplied and then added to get $y(0, 1)$.



The value of $y(0, 1)$ is given as $y(0, 1) = 5 \times 1 = 5$.

3. To determine the value of $y(0, 2)$

The value of $h(-m, -n)$ is shifted by two units along the ' n ' axis to get $h(-m, 2-n)$. Then, the common values between $x(m, n)$ and $h(-m, 2-n)$ are multiplied and then added to get $y(0, 2)$.

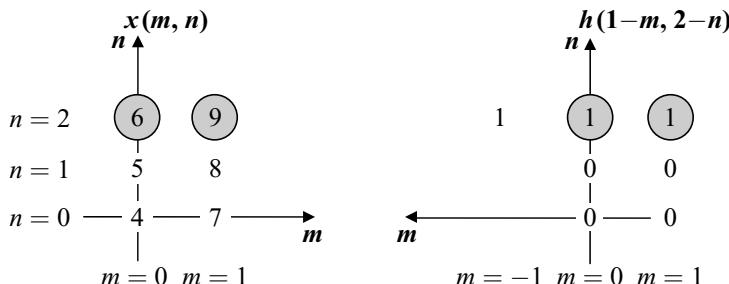


The resultant value of $y(0, 2)$ is $y(0, 2) = 6 \times 1 = 6$.

If we do one more shift of $h(-m, -n)$ along the ' n ' axis then there is no common value between $x(m, n)$ and $h(-m, 3-n)$ so that the resultant value will be zero which is not illustrated here.

4. To determine the value of $y(1, 2)$

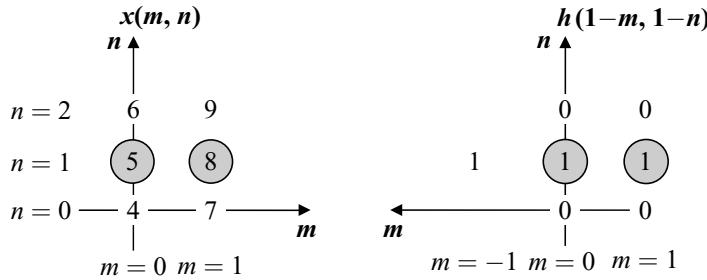
Here, $h(-m, 2-n)$ is shifted along the ' m ' axis to one unit towards right to get $h(1-m, 2-n)$. Then the common values between $x(m, n)$ and $h(1-m, 2-n)$ are multiplied and added to get $y(1, 2)$.



The final value of $y(1, 2) = 6 \times 1 + 9 \times 1 = 15$.

5. To determine the value of $y(1, 1)$

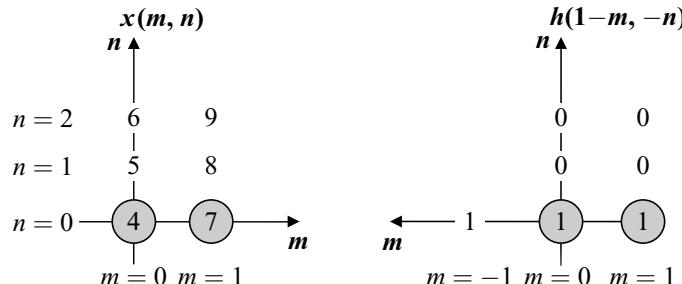
Now the value of $h(1-m, 2-n)$ is shifted down along the 'n' axis to get $h(1-m, 1-n)$. The common values between $x(m, n)$ and $h(1-m, 1-n)$ are multiplied and added to get $y(1, 1)$.



The value of $y(1, 1)$ is obtained as $y(1, 1) = 5 \times 1 + 8 \times 1 = 13$.

6. To determine the value of $y(1, 0)$

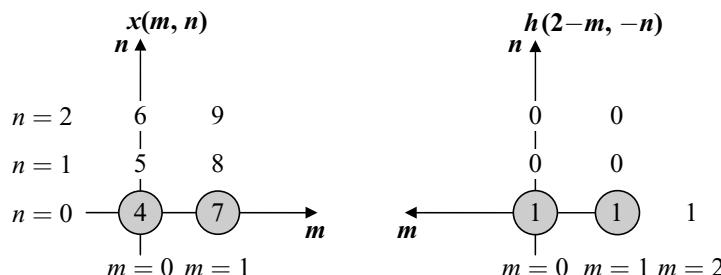
Next, the value of $h(1-m, 1-n)$ is shifted down along the 'n' axis by one unit to get $h(1-m, -n)$ and it is multiplied with $x(m, n)$ and then added to get $y(1, 0)$.



There are two coincide values. Therefore, the result $y(1, 0) = 4 \times 1 + 7 \times 1 = 11$.

7. To determine the value of $y(2, 0)$

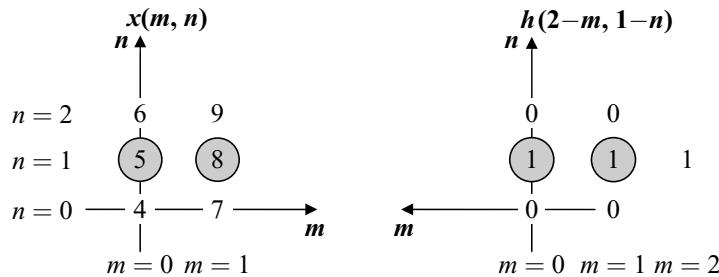
The signal $h(1-m, -n)$ is shifted along the 'm' axis towards right by one unit to get $h(2-m, -n)$. The common value between the signals $x(m, n)$ and $h(2-m, -n)$ is multiplied and then added to get $y(2, 0)$.



The resultant value of $y(2, 0) = 4 \times 1 + 7 \times 1 = 11$.

8. To determine the value of $y(2, 1)$

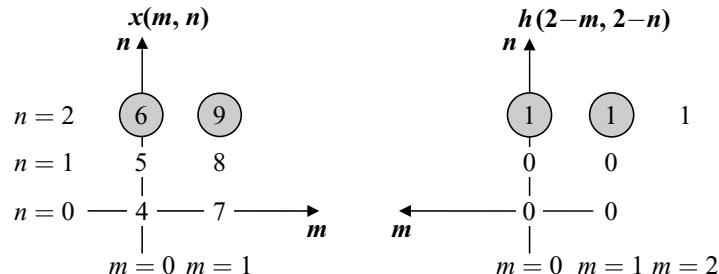
The signal $h(2-m, -n)$ in the previous step is then shifted up along the ' n ' axis by one unit to get $h(2-m, 1-n)$. This signal is multiplied with $x(m, n)$ and added to get $y(2, 1)$.



The value $y(2, 1)$ is obtained as $y(2, 1) = 5 \times 1 + 8 \times 1 = 13$.

9. To determine the value of $y(2, 2)$

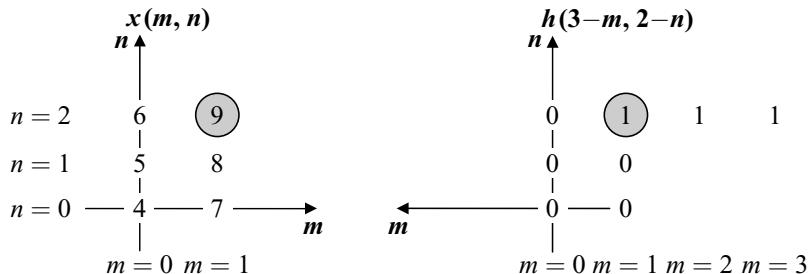
The signal $h(2-m, 1-n)$ is then shifted up by one unit along the ' n ' axis to get $h(2-m, 2-n)$. This signal $h(2-m, 2-n)$ is multiplied with $x(m, n)$ and added to get $y(2, 2)$.



The resultant value $y(2, 1) = 6 \times 1 + 9 \times 1 = 15$.

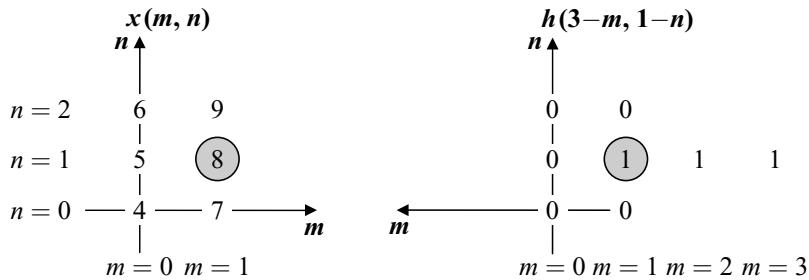
10. To calculate the value of $y(3, 2)$

The signal $h(2-m, 2-n)$ is shifted along the m -axis towards right by one unit to get $h(3-m, 2-n)$. The common values are multiplied and added to get $y(3, 2)$.



The value of $y(3, 2) = 9 \times 1 = 9$.

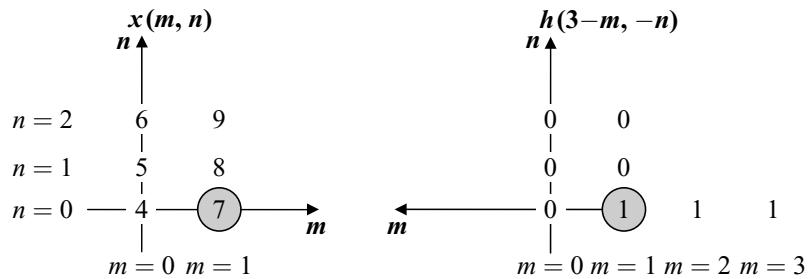
11. To determine the value of $y(3, 1)$



$$y(3, 1) = 8 \times 1 = 8.$$

12. To calculate the result of $y(3, 0)$

Finally the signal $h(3-m, 1-n)$ is shifted down by one unit along the 'n' axis to get $h(3-m, -n)$. This signal is multiplied with $x(m, n)$ and then added to get $y(3, 0)$.

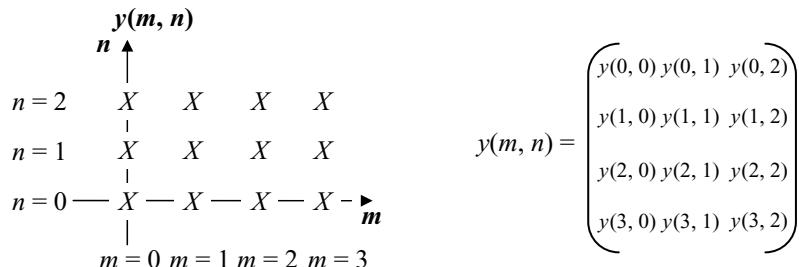


$$\text{The value of } y(3, 0) = 7 \times 1 = 7.$$

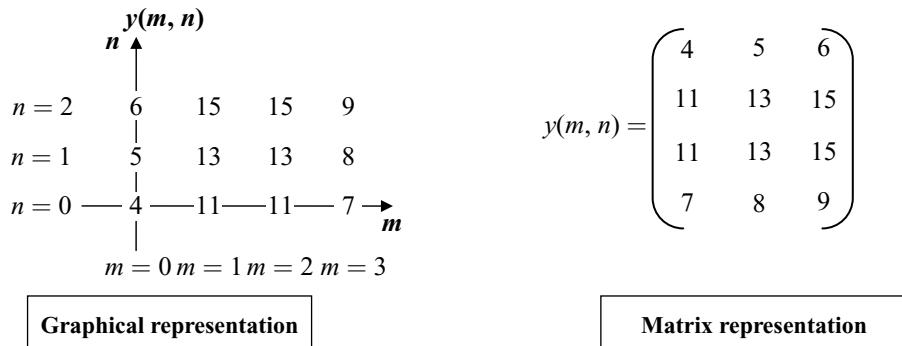
The resultant values obtained from steps 1 to 12 are given below:

$$\begin{array}{lll} y(0, 0) = 4 & y(0, 1) = 5 & y(0, 2) = 6 \\ y(1, 2) = 15 & y(1, 1) = 13 & y(1, 0) = 11 \\ y(2, 0) = 11 & y(2, 1) = 13 & y(2, 1) = 15 \\ y(3, 2) = 9 & y(3, 1) = 8 & y(3, 0) = 7 \end{array}$$

Substituting the above values in the corresponding positions in the matrix and graphic form, the resultant matrix is obtained as



The final result in graphical and matrix representations is given below.



Graphical representation

Matrix representation

Example 3.2 Perform the linear convolution between the two matrices $x(m, n)$ and $h(m, n)$ given below.

$$x(m, n) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Solution The indices of the given input matrices are shown below.

$$x(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 1 & 2 & 3 \\ (1,0) & (1,1) & (1,2) \\ 4 & 5 & 6 \\ (2,0) & (2,1) & (2,2) \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} (0,0) & (0,1) \\ 1 & 1 \\ (1,0) & (1,1) \\ 1 & 1 \\ (2,0) & (2,1) \\ 1 & 1 \end{pmatrix}$$

Determining the dimension of the resultant matrix The dimension of the resultant matrix depends on the dimensions of the input matrices, $x(m, n)$ and $h(m, n)$. The dimension of $x(m, n)$ is 3×3 (three rows and three columns). The dimension of $h(m, n)$ is 3×2 (three rows and two columns). Therefore, the resultant matrix dimension is calculated as

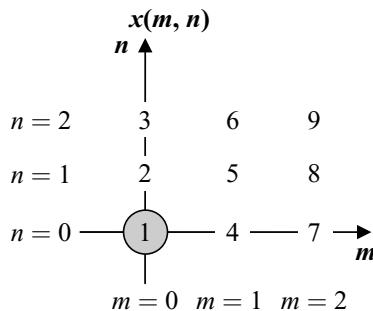
$$\text{Dimension of resultant matrix} = \begin{cases} (\text{No. of rows of } x(m, n) + \text{No. of rows of } h(m, n) - 1) \\ \times \\ (\text{No. of columns of } x(m, n) + \text{No. of columns of } h(m, n) - 1) \end{cases}$$

$$\text{Dimension of resultant matrix} = (3 + 3 - 1) \times (3 + 2 - 1) = 5 \times 4$$

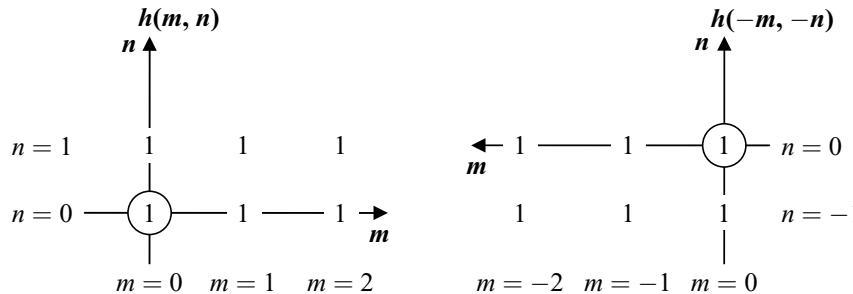
The resultant matrix, $y(m, n)$ of size 5×4 is given as

$$y(m, n) = \begin{pmatrix} y(0, 0) & y(0, 1) & y(0, 2) & y(0, 3) \\ y(1, 0) & y(1, 1) & y(1, 2) & y(1, 3) \\ y(2, 0) & y(2, 1) & y(2, 2) & y(2, 3) \\ y(3, 0) & y(3, 1) & y(3, 2) & y(3, 3) \\ y(4, 0) & y(4, 1) & y(4, 2) & y(4, 3) \end{pmatrix}$$

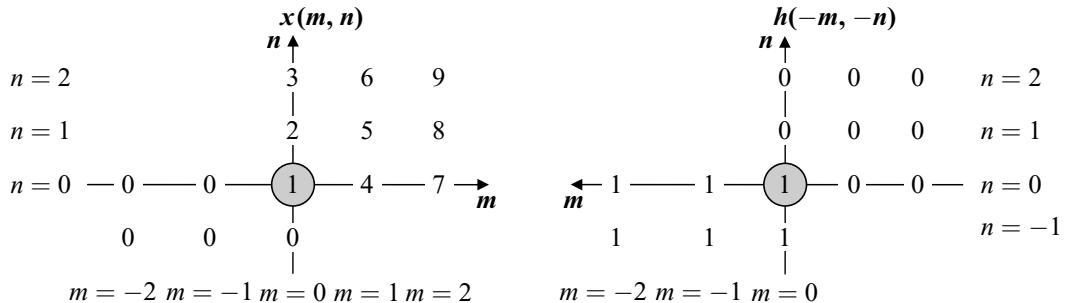
The graphical representation of $x(m, n)$ is shown below:



The graphical representation of $h(m, n)$ and its folded version, $h(-m, -n)$ are given below:

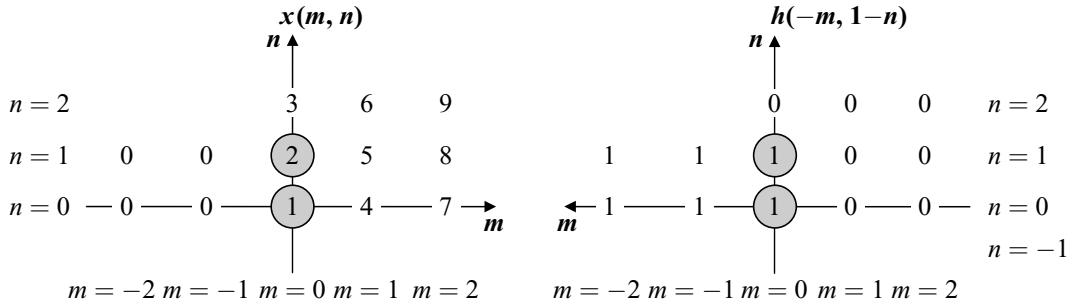


1. To determine the value of $y(0, 0)$ The signal $h(m, n)$ is folded in the ' m ' and ' n ' axes to get $h(-m, -n)$. On multiplying the common values of $x(m, n)$ with $h(-m, -n)$ and then adding, we get $y(0, 0)$.



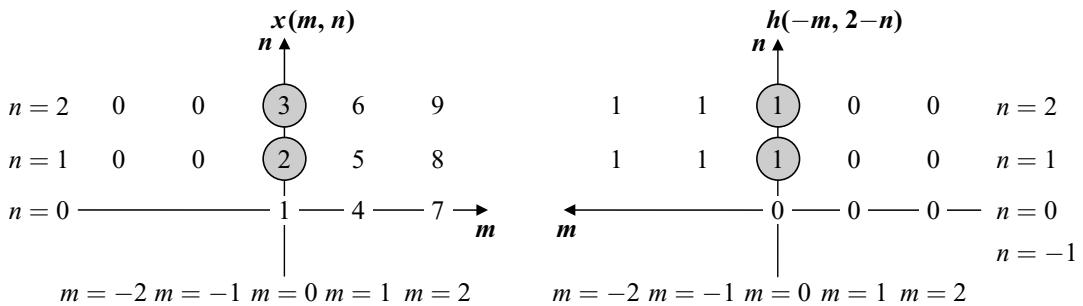
The value of $y(0, 0) = 1 \times 1 = 1$.

2. To determine the value of $y(0, 1)$ Next, the signal $h(-m, -n)$ is shifted along the ‘ n ’ axis towards up by one unit to get $h(-m, 1-n)$. From the signal $x(m, n)$ and $h(-m, 1-n)$ we get the output $y(0, 1)$.



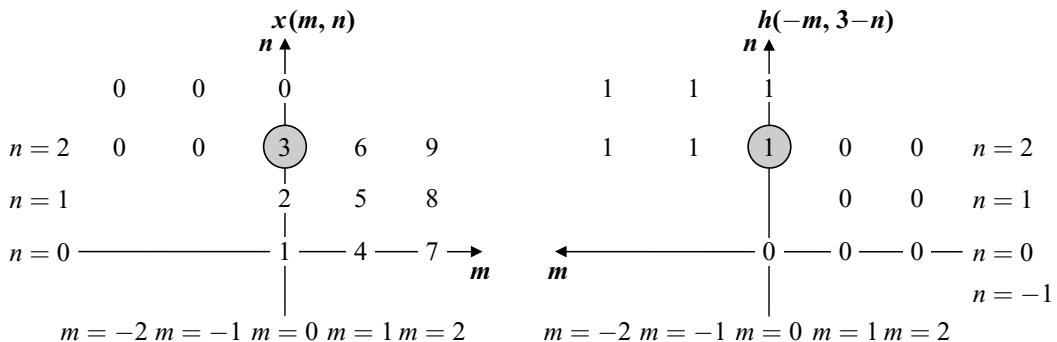
The output $y(0, 1) = 1 \times 1 + 2 \times 1 = 3$.

3. Finding the value of $y(0, 2)$ The signal $h(-m, 2-n)$ is obtained by shifting the signal $h(-m, 1-n)$ along the ‘ n ’ axis by one unit.



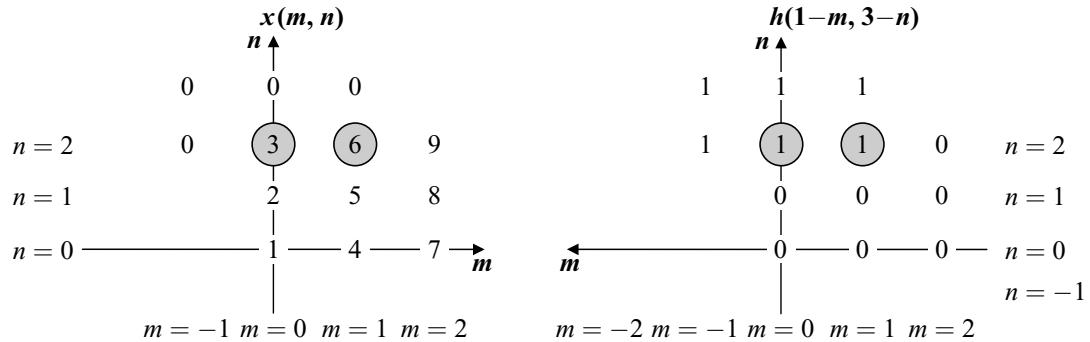
The signal $y(0, 2)$ is obtained by multiplying the common elements of $x(m, n)$ with $h(-m, 2-n)$ and then added to get $y(0, 2) = 2 \times 1 + 3 \times 1 = 5$.

4. Finding the value of $y(0, 3)$ The signal $h(-m, 3-n)$ is obtained by shifting the signal $h(-m, 2-n)$ along the ‘ n ’ axis by one unit. The value $y(0, 3)$ is obtained from $x(m, n)$ and $h(-m, 3-n)$ which is illustrated below.



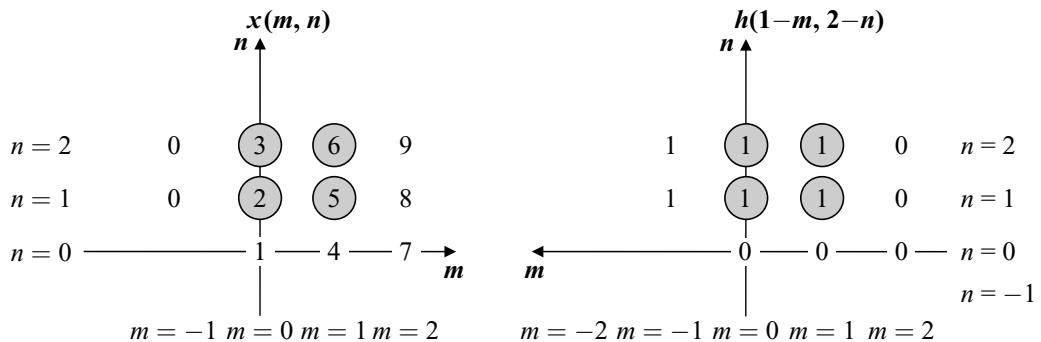
The value $y(0, 3) = 3 \times 1 = 3$.

5. Finding the value of $y(1, 3)$ The signal $h(1-m, 3-n)$ is obtained by shifting $h(-m, 3-n)$ along 'm' axis by one unit towards the right. The values that are common to the signal $h(1-m, 3-n)$ and $x(m, n)$ are multiplied and then added to get $y(1, 3)$.



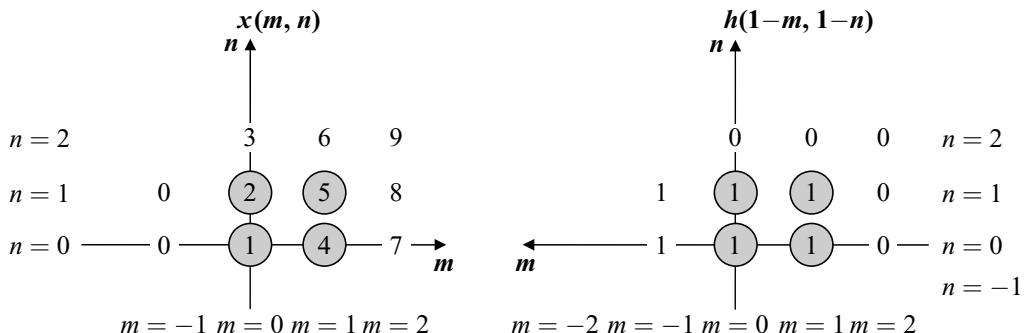
The value $y(1, 3) = 3 \times 1 + 6 \times 1 + 2 \times 1 + 5 \times 1 = 16$.

6. Finding the value of $y(1, 2)$ The signal $h(1-m, 2-n)$ is obtained from $h(1-m, 3-n)$ by shifting it down along the 'n' axis by a factor of one. The signal $h(1-m, 2-n)$ is multiplied with $x(m, n)$ and then added to get $y(1, 2)$.



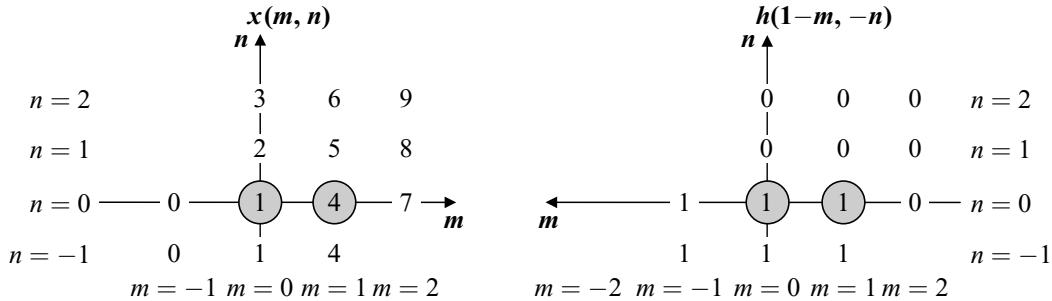
The value $y(1, 2) = 3 \times 1 + 6 \times 1 + 2 \times 1 + 5 \times 1 = 16$.

7. Finding the value of $y(1, 1)$ The signal $h(1-m, 1-n)$ is obtained from $h(1-m, 2-n)$ by shifting it down along the 'n' axis by a factor of one. The common values between the two signals are indicated by shaded circle. The values in the shaded circles are multiplied and then added to get $y(1, 1)$.



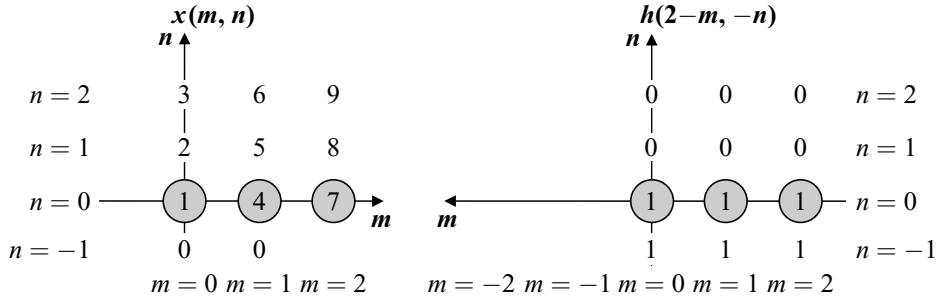
The value $y(1, 1) = 2 \times 1 + 5 \times 1 + 1 \times 1 + 4 \times 1 = 12$.

- 8. Finding the value of $y(1, 0)$** The signal $h(1-m, -n)$ is obtained from $h(1-m, 1-n)$ by shifting it down along the 'n' axis by a factor of one. The signal $h(1-m, -n)$ is multiplied with $x(m, n)$ and then it is added to get $y(1, 0)$.



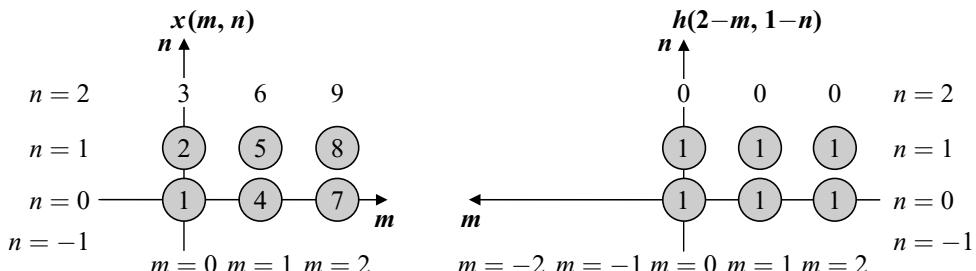
The value $y(1, 0) = 1 \times 1 + 4 \times 1 = 5$.

- 9. Finding the value of $y(2, 0)$** The signal $h(2-m, -n)$ is obtained from $h(1-m, -n)$ by shifting it right along the 'm' axis by a factor of one. The signal $h(2-m, -n)$ is multiplied with $x(m, n)$ and then it is added to get $y(2, 0)$.



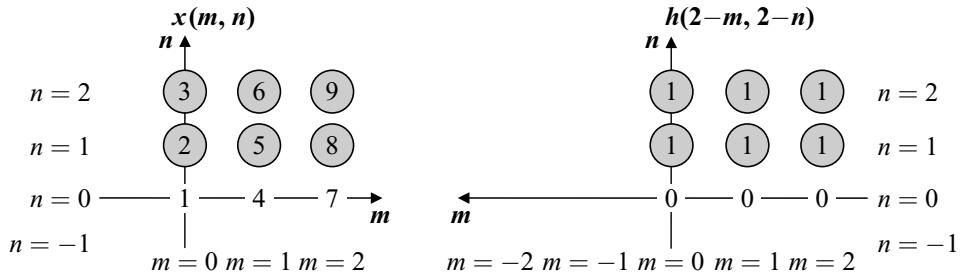
The value $y(2, 0) = 1 \times 1 + 4 \times 1 + 7 \times 1 = 12$.

- 10. Finding the value of $y(2, 1)$** The signal $h(2-m, 1-n)$ is obtained by shifting $h(2-m, -n)$ along the 'n' axis by a factor of one. The common values between $x(m, n)$ with $h(2-m, 1-n)$ are shown by shaded circles. The output signal $y(2, 1)$ is obtained by multiplying the common values and then adding the resultant values.



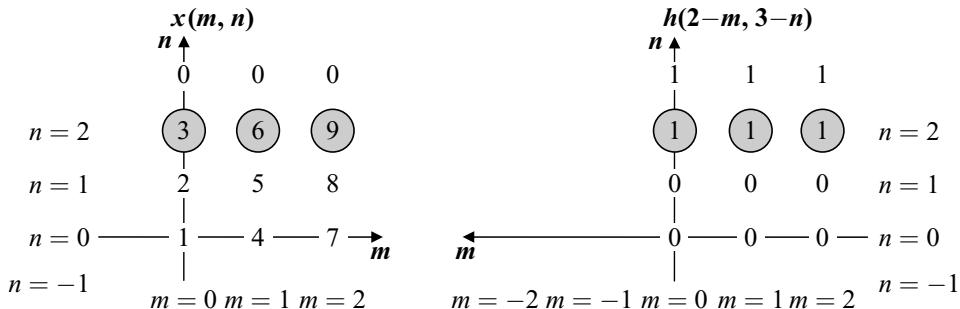
The value $y(2, 1) = 1 \times 1 + 4 \times 1 + 7 \times 1 + 2 \times 1 + 5 \times 1 + 8 \times 1 = 27$.

- 11. Finding the value of $y(2, 2)$** The signal $h(2-m, 2-n)$ is obtained by shifting $h(2-m, 1-n)$ along the 'n' axis by a factor of one.



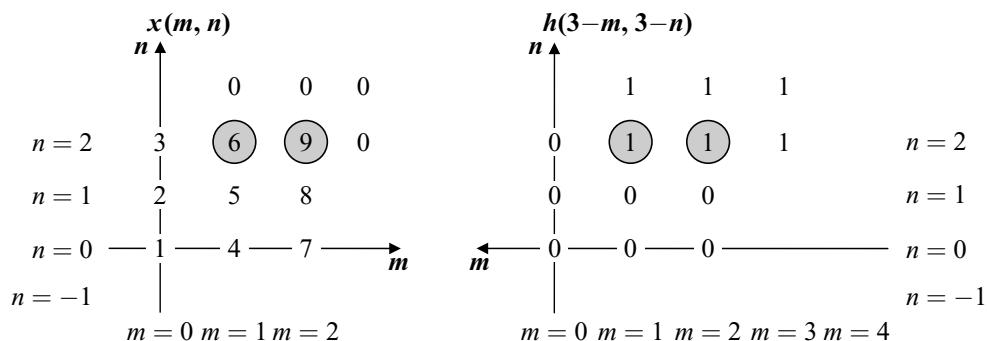
The resultant signal $y(2, 2) = 3 \times 1 + 6 \times 1 + 9 \times 1 + 2 \times 1 + 5 \times 1 + 8 \times 1 = 33$.

12. Finding the value of $y(2, 3)$ The signal $h(2-m, 3-n)$ is obtained by shifting $h(2-m, 2-n)$ along the 'n' axis by a factor of one. The common values between $x(m, n)$ and $h(2-m, 3-n)$ is multiplied and then added to get $y(2, 3)$.



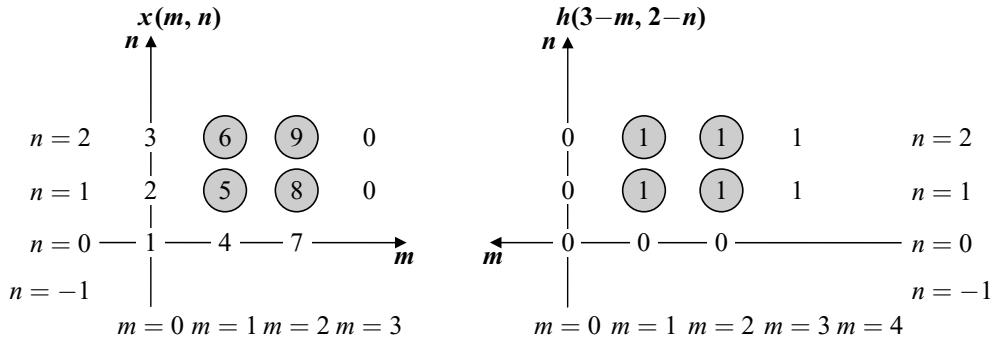
The value $y(2, 3) = 3 \times 1 + 6 \times 1 + 9 \times 1 = 18$.

13. Finding the value of $y(3, 3)$ The signal $h(3-m, 3-n)$ is obtained by shifting the signal $h(2-m, 3-n)$ along the m -axis towards right by a factor of one.



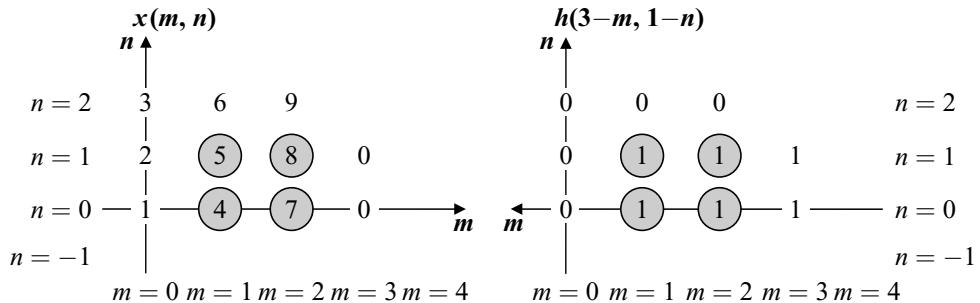
The resultant value $y(3, 3) = 6 \times 1 + 9 \times 1 = 15$.

14. To determine the value of $y(3, 2)$ The signal $h(3-m, 2-n)$ is obtained by shifting down the signal $h(3-m, 3-n)$ along the n -axis by a factor of one. The common values are multiplied and then added together to get $y(3, 2)$.



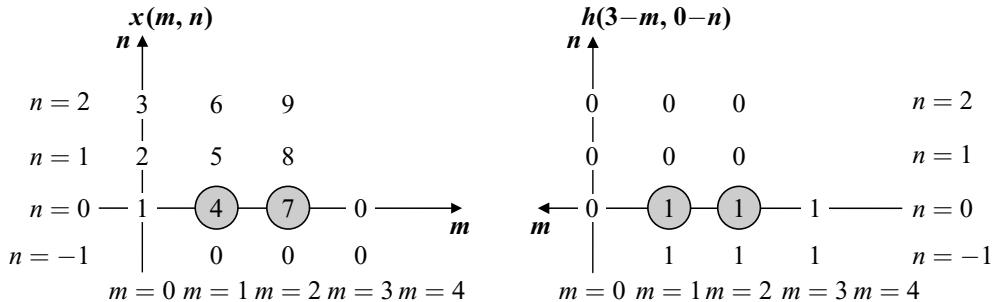
The value $y(3, 2) = 6 \times 1 + 9 \times 1 + 5 \times 1 + 8 \times 1 = 28$.

15. Finding the value of $y(3, 1)$ The signal $h(3-m, 1-n)$ is obtained by shifting down the signal $h(3-m, 2-n)$ along the n -axis by a factor of one. The common values are multiplied and then added together to get $y(3, 1)$.



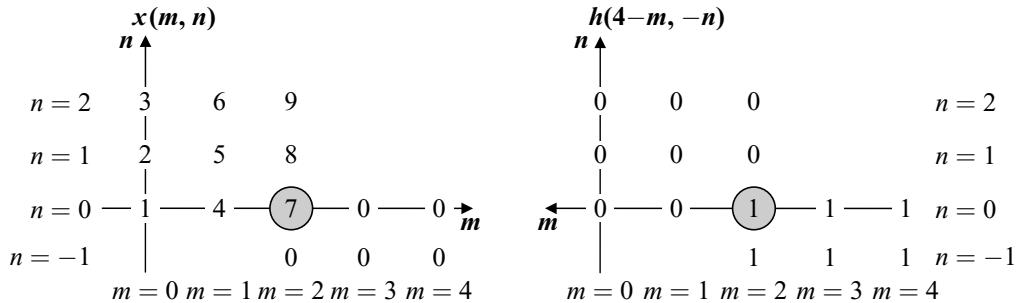
The value of $y(3, 1) = 5 \times 1 + 8 \times 1 + 4 \times 1 + 7 \times 1 = 24$.

16. Finding the value of $y(3, 0)$ The signal $h(3-m, -n)$ is obtained by shifting down the signal $h(3-m, 1-n)$ along the n -axis by a factor of one. The common values are multiplied and then added together to get $y(3, 0)$.



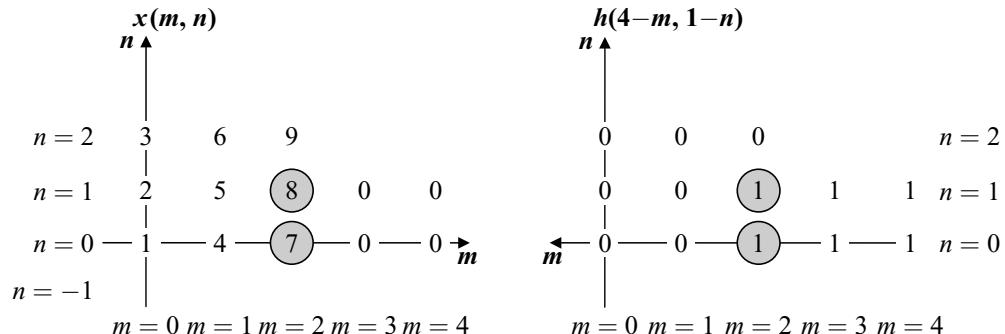
The value $y(3, 0)$ is given by $y(3, 0) = 4 \times 1 + 7 \times 1 = 11$.

17. Finding the value of $y(4, 0)$ The signal $h(4-m, -n)$ is obtained by shifting the signal $h(3-m, -n)$ along the m -axis towards the right by a factor of one. The common values are multiplied and then added together to get $y(4, 0)$.



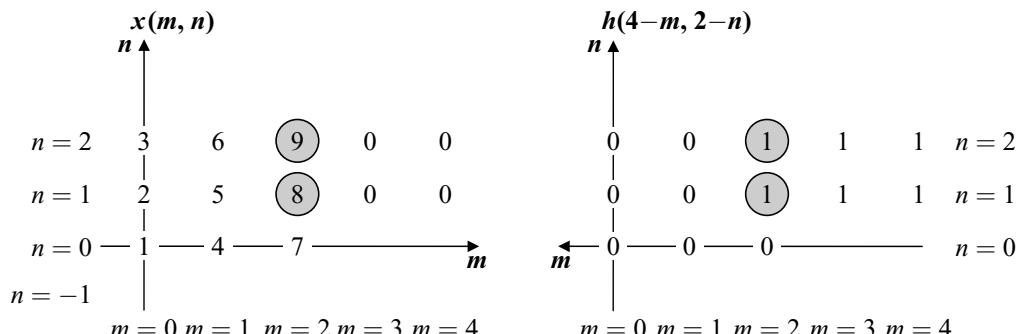
The resultant value $y(4, 0) = 7 \times 1 = 7$.

18. To determine the value of $y(4, 1)$ The signal $h(4-m, 1-n)$ is obtained by shifting up the signal $h(4-m, -n)$ along the n -axis by a factor of one. The common values are multiplied and then added together to get $y(4, 1)$.



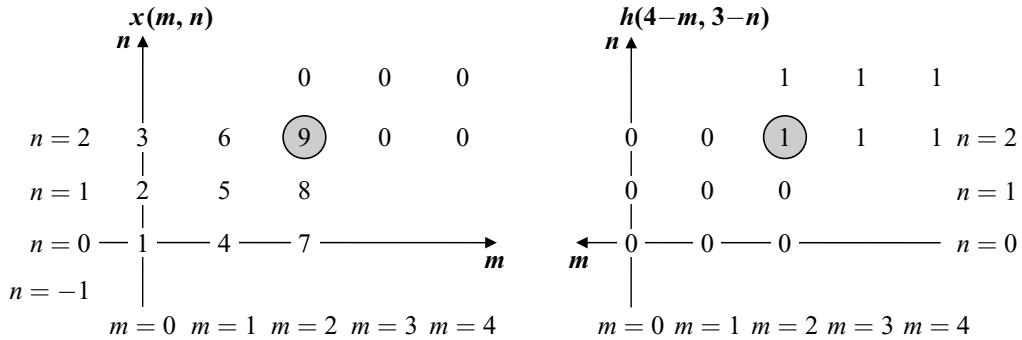
The value $y(4, 1) = 8 \times 1 + 7 \times 1 = 15$.

19. Finding the value of $y(4, 2)$ The signal $h(4-m, 2-n)$ is obtained by shifting up the signal $h(4-m, 1-n)$ along the n -axis by a factor of one. The common values are multiplied and then added together to get $y(4, 2)$.



The value $y(4, 2) = 8 \times 1 + 9 \times 1 = 17$.

20. Finding the value of $y(4, 3)$ The signal $h(4-m, 3-n)$ is obtained by shifting up the signal $h(4-m, 2-n)$ along the n -axis by a factor of one. The common values are multiplied and then added together to get $y(4, 3)$.

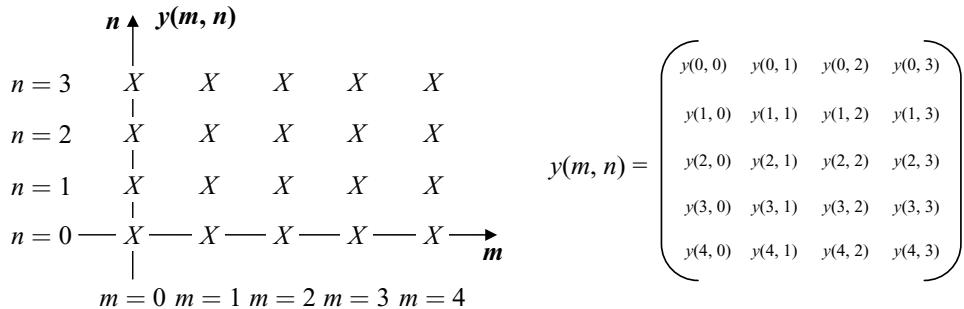


The value $y(4, 3) = 9 \times 1 = 9$.

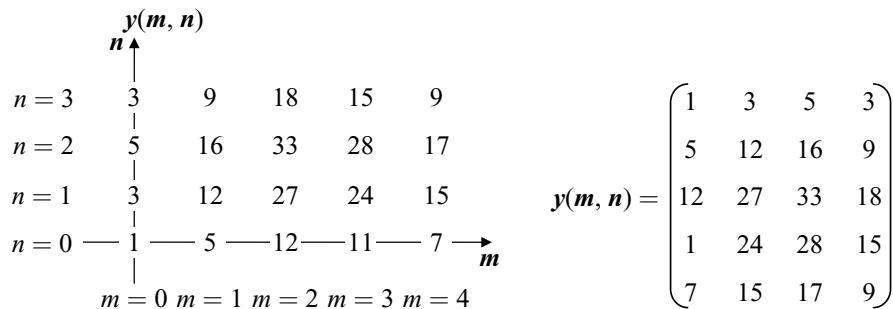
The resultant values obtained from steps 1 to 20 are given below:

$$\begin{array}{llll}
 y(0, 0) = 1 & y(0, 1) = 3 & y(0, 2) = 5 & y(0, 3) = 3 \\
 y(1, 3) = 9 & y(1, 2) = 16 & y(1, 1) = 12 & y(1, 0) = 5 \\
 y(2, 0) = 12 & y(2, 1) = 27 & y(2, 2) = 33 & y(2, 3) = 18 \\
 y(3, 3) = 15 & y(3, 2) = 28 & y(3, 1) = 24 & y(3, 0) = 11 \\
 y(4, 0) = 7 & y(4, 1) = 15 & y(4, 2) = 17 & y(4, 3) = 9
 \end{array}$$

Substituting the above values in the corresponding positions in the given matrix and graphic form, we can get the resultant values.



The final result in the graphical and matrix forms are given below:



Graphical representation

Matrix representation

Example 3.3 Perform the linear convolution between these two matrices $x(m, n)$ and $h(m, n)$ given below.

$$x(m, n) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = (3 \quad 4 \quad 5)$$

Solution The indices of the given input matrices are shown below.

$$x(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 1 & 2 & 3 \\ (1,0) & (1,1) & (1,2) \\ 4 & 5 & 6 \\ (2,0) & (2,1) & (2,2) \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 3 & 4 & 5 \end{pmatrix}$$

Determining the dimension of the resultant matrix The dimension of the resultant matrix will depend upon the dimension of the input matrices, $x(m, n)$ and $h(m, n)$. The dimension of $x(m, n)$ is given by 3×3 (three rows and three columns). The dimension of $h(m, n)$ is given by 1×3 (one row and three columns). Therefore, the resultant matrix dimension will be calculated as

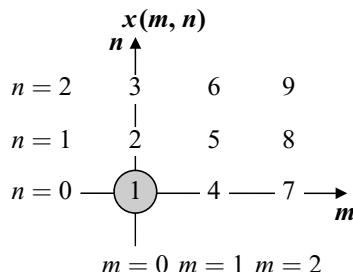
$$\text{Dimension of resultant matrix} = \begin{cases} (\text{No. of rows of } x(m, n) + \text{No. of rows of } h(m, n) - 1) \\ \times \\ (\text{No. of columns of } x(m, n) + \text{No. of columns of } h(m, n) - 1) \end{cases}$$

$$\text{Dimension of resultant matrix} = (3 + 1 - 1) \times (3 + 3 - 1) = 3 \times 5$$

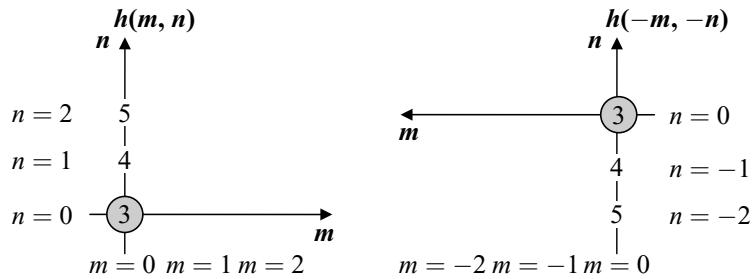
The resultant matrix, $y(m, n)$ of size 3×5 is given as

$$y(m, n) = \begin{pmatrix} y(0, 0) & y(0, 1) & y(0, 2) & y(0, 3) & y(0, 4) \\ y(1, 0) & y(1, 1) & y(1, 2) & y(1, 3) & y(1, 4) \\ y(2, 0) & y(2, 1) & y(2, 2) & y(2, 3) & y(2, 4) \end{pmatrix}$$

The graphical representation of $x(m, n)$ is shown below:

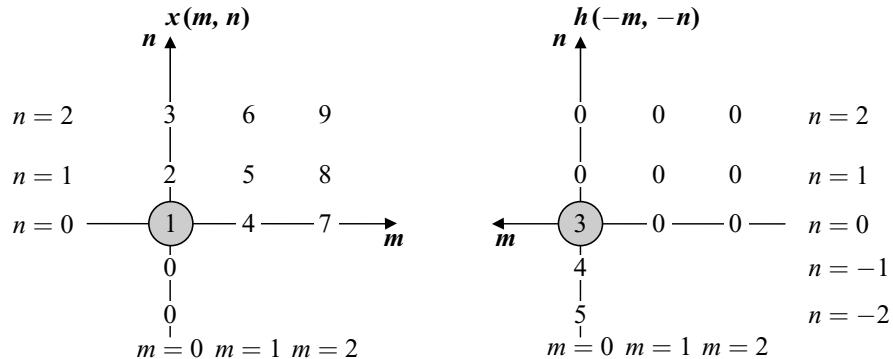


The graphical representation of $h(m, n)$ and its folded version, $h(-m, -n)$ are given below:



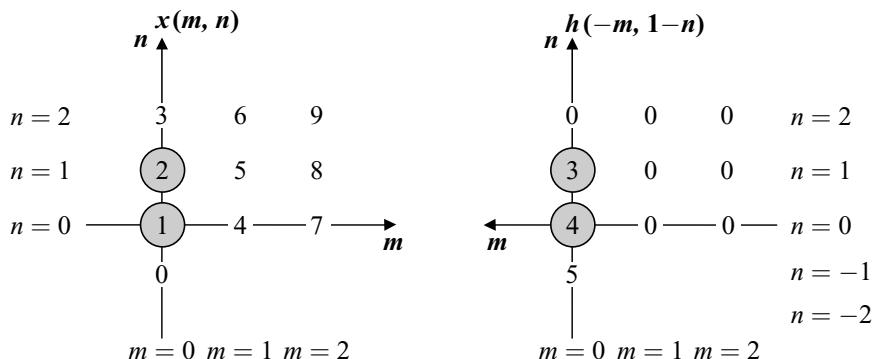
The steps followed in this example are the same as given in Example 3.1 and Example 3.2. Hence, the steps are not explained in detail.

1. To determine the value of $y(0, 0)$



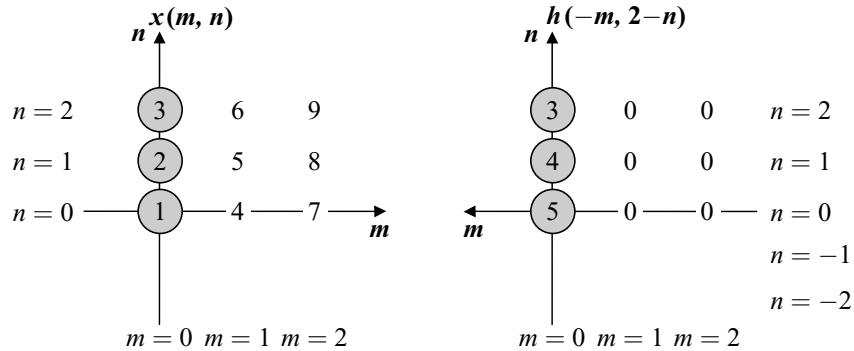
$$y(0, 0) = 1 \times 3 = 3$$

2. Finding the value of $y(0, 1)$



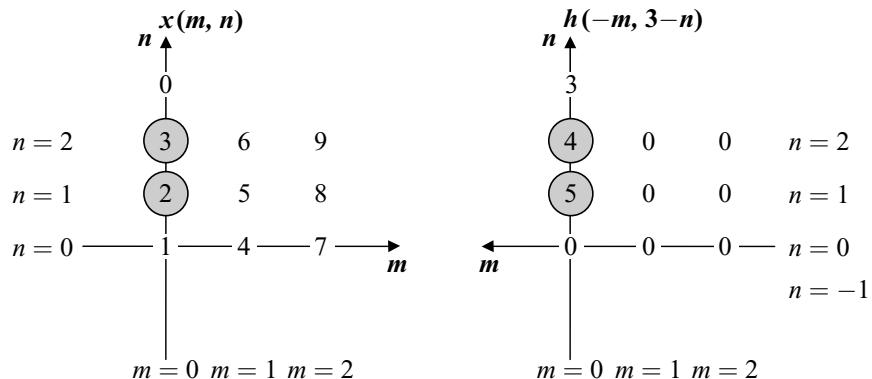
$$y(0, 1) = 2 \times 3 + 1 \times 4 = 10$$

3. Finding the value of $y(0, 2)$



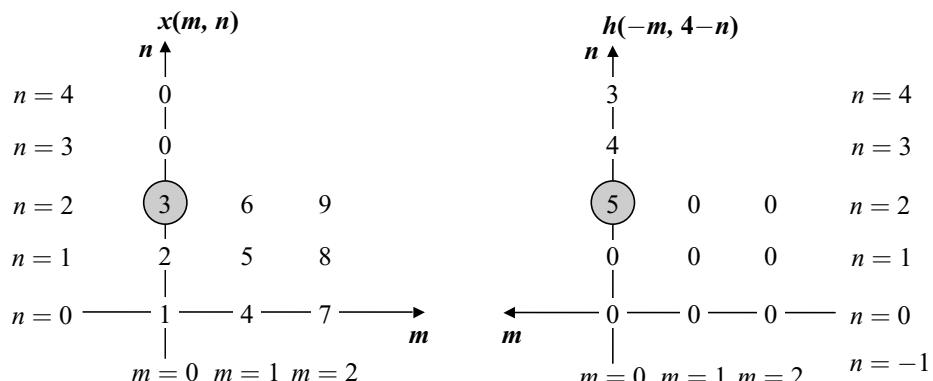
$$y(0, 2) = 3 \times 3 + 2 \times 4 + 1 \times 5 = 22$$

4. Finding the value of $y(0, 3)$



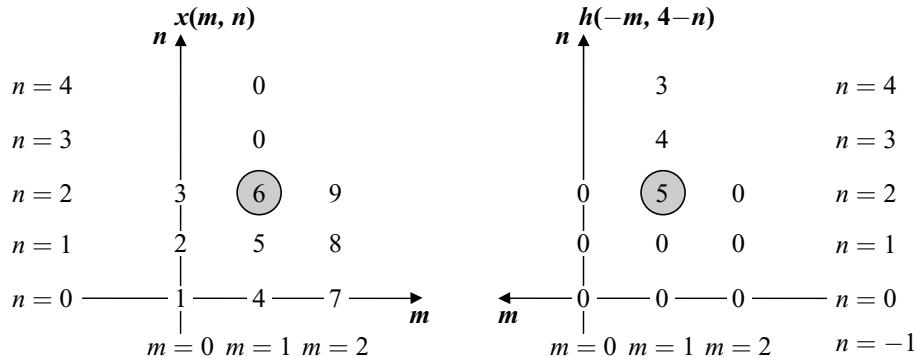
$$y(0, 3) = 3 \times 4 + 2 \times 5 = 22$$

5. Finding the value of $y(0, 4)$



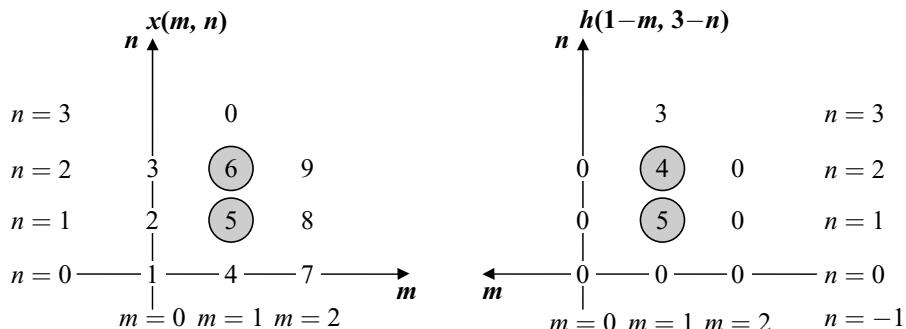
$$y(0, 4) = 3 \times 5 = 15$$

6. Finding the value of $y(1, 4)$



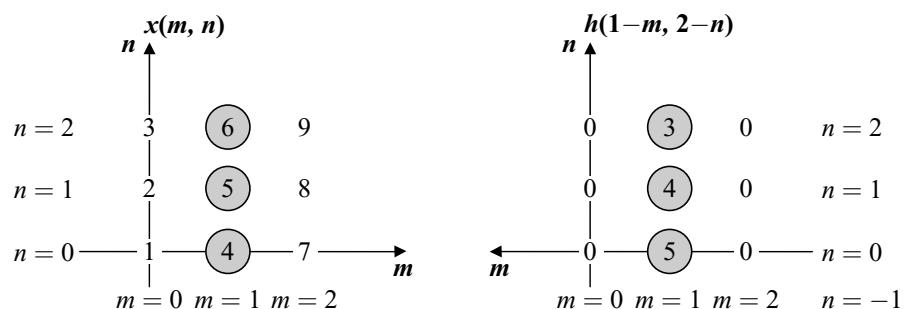
$$y(1, 4) = 6 \times 5 = 30$$

7. Finding the value of $y(1, 3)$

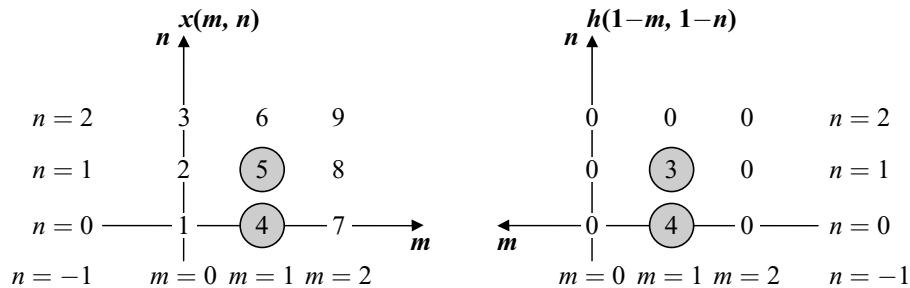


$$y(1, 3) = 6 \times 4 + 5 \times 5 = 49$$

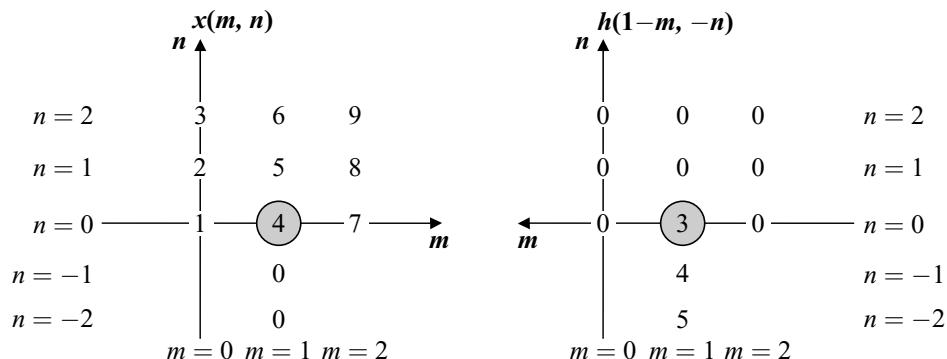
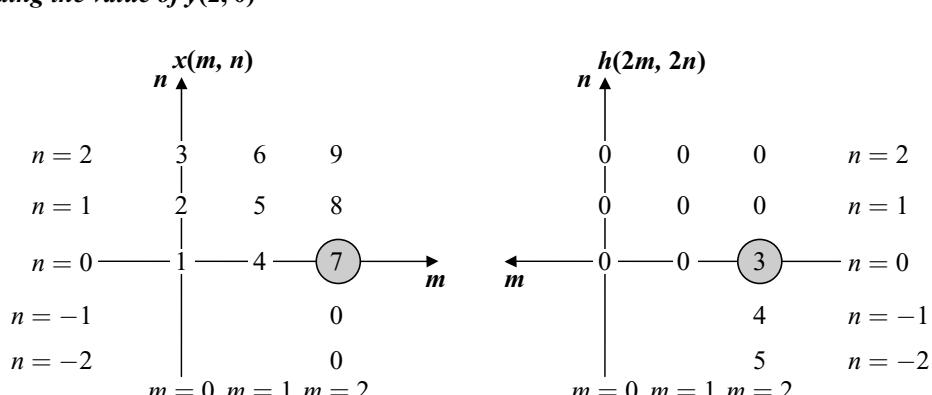
8. Finding the value of $y(1, 2)$



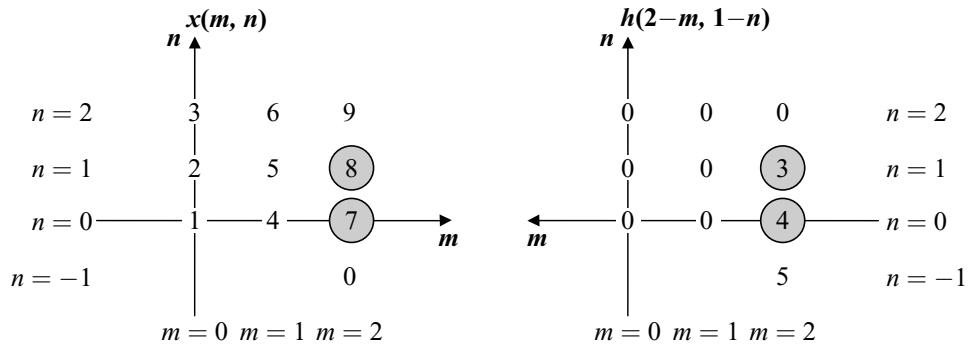
$$y(1, 2) = 6 \times 3 + 5 \times 4 + 4 \times 5 = 58$$

9. Finding the value of $y(1, 1)$ 

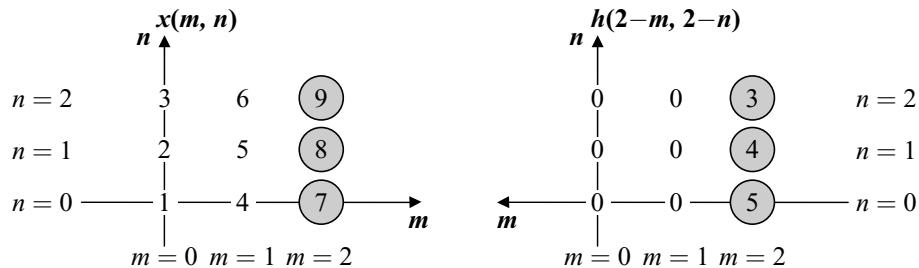
$$y(1, 1) = 5 \times 3 + 4 \times 4 = 31$$

10. Finding the value of $y(1, 0)$ 11. Finding the value of $y(2, 0)$ 

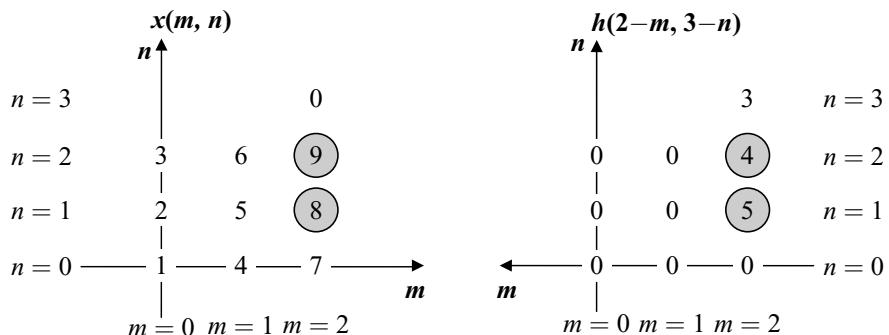
$$y(2, 0) = 7 \times 3 = 21$$

12. Finding the value of $y(2, 1)$ 

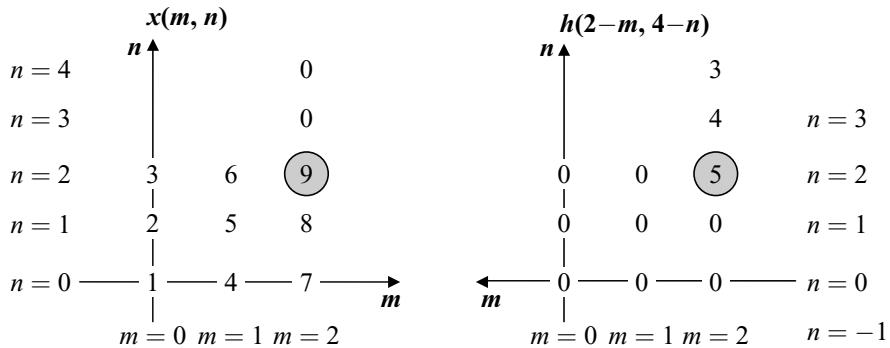
$$y(2, 1) = 8 \times 3 + 7 \times 4 = 52$$

13. Finding the value of $y(2, 2)$ 

$$y(2, 2) = 9 \times 3 + 8 \times 4 + 7 \times 5 = 94$$

14. Finding the value of $y(2, 3)$ 

$$y(2, 3) = 9 \times 4 + 8 \times 5 = 76$$

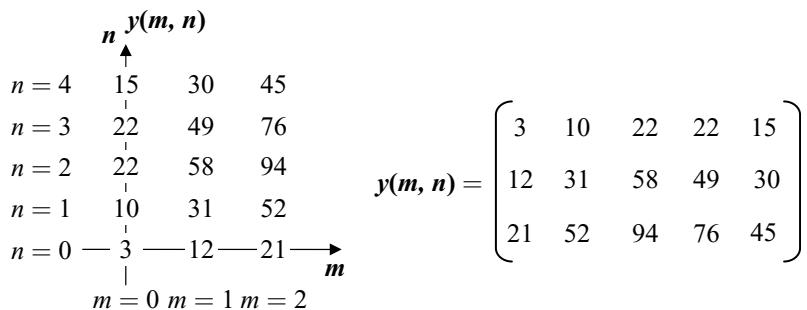
15. Finding the value of $y(2, 4)$ 

$$y(2, 4) = 9 \times 5 = 45$$

The resultant values obtained from steps 1 to 15 are given below:

$$\begin{array}{lllll}
 y(0, 0) = 3 & y(0, 1) = 10 & y(0, 2) = 22 & y(0, 3) = 22 & y(0, 4) = 15 \\
 y(1, 0) = 12 & y(1, 1) = 31 & y(1, 2) = 58 & y(1, 3) = 49 & y(0, 4) = 30 \\
 y(2, 0) = 21 & y(2, 1) = 52 & y(2, 2) = 94 & y(2, 3) = 76 & y(2, 4) = 45
 \end{array}$$

The graphical and the matrix forms of representation of the resultant matrix indices are shown below:



Graphical representation

Matrix representation

3.3 CONVOLUTION THROUGH Z-TRANSFORM

The concept of convolution in spatial domain, which is equal to multiplication in frequency domain, is utilised to compute convolution through Z-transform. The convolution between two sequences $x(n_1, n_2)$ and $h(n_1, n_2)$ is given by

$$y(n_1, n_2) = x(n_1, n_2) \ast\ast h(n_1, n_2) \quad (3.2)$$

Here $\ast\ast$ indicates the 2D convolution between $x(n_1, n_2)$ and $h(n_1, n_2)$.

Taking Z-transform on both sides, we get

$$Y(Z_1, Z_2) = X(Z_1, Z_2) \times H(Z_1, Z_2) \quad (3.3)$$

Thus, convolution in one domain is equal to multiplication in another domain.

Example 3.4 The input matrices are $x(m, n)$ and $h(m, n)$. Perform the linear convolution between these two matrices.

$$x(m, n) = \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}; \quad h(m, n) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Solution The indices of the given input matrices are shown below.

$$x(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 4 & 5 & 6 \\ (1,0) & (1,1) & (1,2) \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} (0,0) \\ 1 \\ (1,0) \\ 1 \\ (2,0) \\ 1 \end{pmatrix}$$

Step 1 Computation of Z-transform of the sequence $x(m, n)$

$$X(Z_1, Z_2) = 4(Z_1^{-0}, Z_2^{-0}) + 5(Z_1^{-0}, Z_2^{-1}) + 6(Z_1^{-0}, Z_2^{-2}) + 7(Z_1^{-1}, Z_2^{-0}) + 8(Z_1^{-1}, Z_2^{-1}) + 9(Z_1^{-1}, Z_2^{-2})$$

Step 2 Computation of Z-transform of the sequence $h(m, n)$

$$H(Z_1, Z_2) = 1(Z_1^{-0}, Z_2^{-0}) + 1(Z_1^{-1}, Z_2^{-0}) + 1(Z_1^{-2}, Z_2^{-0})$$

Step 3 Computation of the product of $X(Z_1, Z_2)$ and $H(Z_1, Z_2)$

$$\begin{aligned} Y(Z_1, Z_2) &= X(Z_1, Z_2) \times H(Z_1, Z_2) \\ &= \{4(Z_1^{-0}, Z_2^{-0}) + 5(Z_1^{-0}, Z_2^{-1}) + 6(Z_1^{-0}, Z_2^{-2}) + 7(Z_1^{-1}, Z_2^{-0}) + 8(Z_1^{-1}, Z_2^{-1}) + 9(Z_1^{-1}, Z_2^{-2})\} \\ &\quad \times \{(1(Z_1^{-0}, Z_2^{-0}) + 1(Z_1^{-1}, Z_2^{-0}) + 1(Z_1^{-2}, Z_2^{-0})\} \\ &= 4(Z_1^{-0}, Z_2^{-0}) + 4(Z_1^{-1}, Z_2^{-0}) + 4(Z_1^{-2}, Z_2^{-0}) + 5(Z_1^{-0}, Z_2^{-1}) + 5(Z_1^{-1}, Z_2^{-1}) + 5(Z_1^{-2}, Z_2^{-1}) \\ &\quad + 6(Z_1^{-0}, Z_2^{-2}) + 6(Z_1^{-1}, Z_2^{-2}) + 6(Z_1^{-2}, Z_2^{-2}) + 7(Z_1^{-1}, Z_2^{-0}) + 7(Z_1^{-2}, Z_2^{-0}) + 7(Z_1^{-3}, Z_2^{-0}) \\ &\quad + 8(Z_1^{-1}, Z_2^{-1}) + 8(Z_1^{-2}, Z_2^{-1}) + 8(Z_1^{-3}, Z_2^{-1}) + 9(Z_1^{-1}, Z_2^{-2}) + 9(Z_1^{-2}, Z_2^{-2}) + 9(Z_1^{-3}, Z_2^{-2}) \\ &= 4(Z_1^{-0}, Z_2^{-0}) + 11(Z_1^{-1}, Z_2^{-0}) + 11(Z_1^{-2}, Z_2^{-0}) + 5(Z_1^{-0}, Z_2^{-1}) + 13(Z_1^{-1}, Z_2^{-1}) + 13(Z_1^{-2}, Z_2^{-1}) \\ &\quad + 6(Z_1^{-0}, Z_2^{-2}) + 15(Z_1^{-1}, Z_2^{-2}) + 15(Z_1^{-2}, Z_2^{-2}) + 7(Z_1^{-3}, Z_2^{-0}) + 8(Z_1^{-3}, Z_2^{-1}) + 9(Z_1^{-3}, Z_2^{-2}) \end{aligned}$$

Step 4 Grouping of the terms

The result obtained in Step 3 is grouped together to get the final result as

$$y(m, n) = \begin{pmatrix} 4 & 5 & 6 \\ 11 & 13 & 15 \\ 11 & 13 & 15 \\ 7 & 8 & 9 \end{pmatrix}$$

Example 3.5 Perform the linear convolution between the two matrices $x(m, n)$ and $h(m, n)$ given below using Z-transform.

$$x(m, n) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Solution The indices of the given input matrices are shown below.

$$x(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 1 & 2 & 3 \\ (1,0) & (1,1) & (1,2) \\ 4 & 5 & 6 \\ (2,0) & (2,1) & (2,2) \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} (0,0) & (0,1) \\ 1 & 1 \\ (1,0) & (1,1) \\ 1 & 1 \\ (2,0) & (2,1) \\ 1 & 1 \end{pmatrix}$$

Step 1 Computation of Z transform of $x(m, n)$

The two-dimensional Z-transform of the input matrix $x(m, n)$ is given by

$$X(Z_1, Z_2) = 1(Z_1^0, Z_2^0) + 2(Z_1^0, Z_2^{-1}) + 3(Z_1^0, Z_2^{-2}) + 4(Z_1^{-1}, Z_2^0) + 5(Z_1^{-1}, Z_2^{-1}) + 6(Z_1^{-1}, Z_2^{-2}) \\ + 7(Z_1^{-2}, Z_2^0) + 8(Z_1^{-2}, Z_2^{-1}) + 9(Z_1^{-2}, Z_2^{-2}).$$

Step 2 Computation of Z transform of $h(m, n)$

The two-dimensional Z-transform of the input matrix $h(m, n)$ is given by

$$H(Z_1, Z_2) = 1(Z_1^0, Z_2^0) + 1(Z_1^0, Z_2^{-1}) + 1(Z_1^{-1}, Z_2^0) + 1(Z_1^{-1}, Z_2^{-1}) + 1(Z_1^{-2}, Z_2^0) + 1(Z_1^{-2}, Z_2^{-1}).$$

Step 3 Computation of the product of $X(Z_1, Z_2)$ and $H(Z_1, Z_2)$

The product of $X(Z_1, Z_2)$ and $H(Z_1, Z_2)$ is given by

$$Y(Z_1, Z_2) = X(Z_1, Z_2) \times H(Z_1, Z_2)$$

$$Y(Z_1, Z_2) = \{1(Z_1^0, Z_2^0) + 2(Z_1^0, Z_2^{-1}) + 3(Z_1^0, Z_2^{-2}) + 4(Z_1^{-1}, Z_2^0) + 5(Z_1^{-1}, Z_2^{-1}) + 6(Z_1^{-1}, Z_2^{-2}) \\ + 7(Z_1^{-2}, Z_2^0) + 8(Z_1^{-2}, Z_2^{-1}) + 9(Z_1^{-2}, Z_2^{-2})\} \times \{(1(Z_1^0, Z_2^0) + 1(Z_1^0, Z_2^{-1}) + 1(Z_1^{-1}, Z_2^0) \\ + 1(Z_1^{-1}, Z_2^{-1}) + 1(Z_1^{-2}, Z_2^0) + 1(Z_1^{-2}, Z_2^{-1})\}$$

$$\begin{aligned}
Y(Z_1, Z_2) = & 1(Z_1^{-0}Z_2^{-0}) + 1(Z_1^{-0}Z_2^{-1}) + 1(Z_1^{-1}Z_2^{-0}) + 1(Z_1^{-1}Z_2^{-1}) + 1(Z_1^{-2}Z_2^{-0}) \\
& + 1(Z_1^{-2}Z_2^{-1}) + 2(Z_1^{-0}Z_2^{-1}) + 2(Z_1^{-0}Z_2^{-2}) + 2(Z_1^{-1}Z_2^{-1}) + 2(Z_1^{-1}Z_2^{-2}) + 2(Z_1^{-2}Z_2^{-1}) \\
& + 2(Z_1^{-2}Z_2^{-2}) + 3(Z_1^0Z_2^{-2}) + 3(Z_1^{-0}Z_2^{-3}) + 3(Z_1^{-1}Z_2^{-2}) + 3(Z_1^{-1}Z_2^{-3}) + 3(Z_1^{-2}Z_2^{-2}) \\
& + 3(Z_1^{-2}Z_2^{-3}) + 4(Z_1^{-1}Z_2^{-0}) + 4(Z_1^{-1}Z_2^{-1}) + 4(Z_1^{-2}Z_2^{-0}) + 4(Z_1^{-2}Z_2^{-1}) + 4(Z_1^{-3}Z_2^{-0}) \\
& + 4(Z_1^{-3}Z_2^{-1}) + 5(Z_1^{-1}Z_2^{-1}) + 5(Z_1^{-1}Z_2^{-2}) + 5(Z_1^{-2}Z_2^{-1}) + 5(Z_1^{-2}Z_2^{-2}) + 5(Z_1^{-3}Z_2^{-1}) \\
& + 5(Z_1^{-3}Z_2^{-2}) + 6(Z_1^{-1}Z_2^{-2}) + 6(Z_1^{-1}Z_2^{-3}) + 6(Z_1^{-2}Z_2^{-2}) + 6(Z_1^{-2}Z_2^{-3}) + 6(Z_1^{-3}Z_2^{-2}) \\
& + 6(Z_1^{-3}Z_2^{-3}) + 7(Z_1^{-2}Z_2^{-0}) + 7(Z_1^{-2}Z_2^{-1}) + 7(Z_1^{-3}Z_2^{-0}) + 7(Z_1^{-3}Z_2^{-1}) + 7(Z_1^{-4}Z_2^{-0}) \\
& + 7(Z_1^{-4}Z_2^{-1}) + 8(Z_1^{-2}Z_2^{-1}) + 8(Z_1^{-2}Z_2^{-2}) + 8(Z_1^{-3}Z_2^{-1}) + 8(Z_1^{-3}Z_2^{-2}) + 8(Z_1^{-4}Z_2^{-1}) \\
& + 8(Z_1^{-4}Z_2^{-2}) + 9(Z_1^{-2}Z_2^{-2}) + 9(Z_1^{-2}Z_2^{-3}) + 9(Z_1^{-3}Z_2^{-2}) + 9(Z_1^{-3}Z_2^{-3}) + 9(Z_1^{-4}Z_2^{-2}) \\
& + 9(Z_1^{-4}Z_2^{-3}).
\end{aligned}$$

$$\begin{aligned}
Y(Z_1, Z_2) = & 1(Z_1^{-0}Z_2^{-0}) + 3(Z_1^{-0}Z_2^{-1}) + 5(Z_1^{-0}Z_2^{-2}) + 3(Z_1^{-0}Z_2^{-3}) + 5(Z_1^{-1}Z_2^{-0}) \\
& + 12(Z_1^{-1}Z_2^{-1}) + 16(Z_1^{-1}Z_2^{-2}) + 9(Z_1^{-1}Z_2^{-3}) + 12(Z_1^{-2}Z_2^{-0}) + 27(Z_1^{-2}Z_2^{-1}) \\
& + 33(Z_1^{-2}Z_2^{-2}) + 18(Z_1^{-2}Z_2^{-3}) + 11(Z_1^{-3}Z_2^{-0}) + 24(Z_1^{-3}Z_2^{-1}) + 28(Z_1^{-3}Z_2^{-2}) \\
& + 15(Z_1^{-3}Z_2^{-3}) + 7(Z_1^{-4}Z_2^{-0}) + 15(Z_1^{-4}Z_2^{-1}) + 17(Z_1^{-4}Z_2^{-2}) + 9(Z_1^{-4}Z_2^{-3}).
\end{aligned}$$

Step 4 Rearranging the output to obtain the output $y(m, n)$

$$y(m, n) = \begin{pmatrix} 1 & 3 & 5 & 3 \\ 5 & 12 & 16 & 9 \\ 12 & 27 & 33 & 18 \\ 11 & 24 & 28 & 15 \\ 7 & 15 & 17 & 9 \end{pmatrix}$$

Example 3.6 Perform the linear convolution between the two matrices $x(m, n)$ and $h(m, n)$ through the Z-transform method.

$$x(m, n) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = (3 \quad 4 \quad 5)$$

Solution The indices of the given input matrices are shown below.

$$x(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 1 & 2 & 3 \\ (1,0) & (1,1) & (1,2) \\ 4 & 5 & 6 \\ (2,0) & (2,1) & (2,1) \\ 7 & 8 & 9 \end{pmatrix} \quad h(m, n) = \begin{pmatrix} (0,0) & (0,1) & (0,2) \\ 3 & 4 & 5 \end{pmatrix}$$

Step 1 Computation of Z transform of $x(m, n)$

The two-dimensional Z-transform of the input matrix $x(m, n)$ is given by

$$\begin{aligned} X(Z_1, Z_2) = & 1(Z_1^{-0}Z_2^{-0}) + 2(Z_1^{-0}Z_2^{-1}) + 3(Z_1^{-0}Z_2^{-2}) + 4(Z_1^{-1}Z_2^{-0}) \\ & + 5(Z_1^{-1}Z_2^{-1}) + 6(Z_1^{-1}Z_2^{-2}) + 7(Z_1^{-2}Z_2^{-0}) + 8(Z_1^{-2}Z_2^{-1}) + 9(Z_1^{-2}Z_2^{-2}) \end{aligned}$$

Step 2 Computation of Z transform of $h(m, n)$

The two-dimensional Z transform of the input matrix $h(m, n)$ is given by

$$H(Z_1, Z_2) = 3(Z_1^{-0}Z_2^{-0}) + 4(Z_1^{-0}Z_2^{-1}) + 5(Z_1^{-0}Z_2^{-2})$$

Step 3 Computation of product of $X(Z_1, Z_2)$ and $H(Z_1, Z_2)$

Taking the product of $X(Z_1, Z_2)$ and $H(Z_1, Z_2)$ yields

$$\begin{aligned} Y(Z_1, Z_2) &= X(Z_1, Z_2) \times H(Z_1, Z_2) \\ &= \{(1(Z_1^{-0}Z_2^{-0}) + 2(Z_1^{-0}Z_2^{-1}) + 3(Z_1^{-0}Z_2^{-2}) + 4(Z_1^{-1}Z_2^{-0}) + 5(Z_1^{-1}Z_2^{-1}) + 6(Z_1^{-1}Z_2^{-2}) + 7(Z_1^{-2}Z_2^{-0}) \\ &\quad + 8(Z_1^{-2}Z_2^{-1}) + 9(Z_1^{-2}Z_2^{-2})) \times \{ (3(Z_1^{-0}Z_2^{-0}) + 4(Z_1^{-0}Z_2^{-1}) + 5(Z_1^{-0}Z_2^{-2})) \\ &= 3(Z_1^{-0}Z_2^{-0}) + 4(Z_1^{-0}Z_2^{-1}) + 5(Z_1^{-0}Z_2^{-2}) + 6(Z_1^{-0}Z_2^{-1}) + 8(Z_1^{-0}Z_2^{-2}) + 10(Z_1^{-0}Z_2^{-3}) + 9(Z_1^{-0}Z_2^{-2}) \\ &\quad + 12(Z_1^{-0}Z_2^{-3}) + 15(Z_1^{-0}Z_2^{-4}) + 12(Z_1^{-1}Z_2^{-0}) + 16(Z_1^{-1}Z_2^{-1}) + 20(Z_1^{-1}Z_2^{-2}) + 15(Z_1^{-1}Z_2^{-1}) + 20(Z_1^{-1}Z_2^{-2}) \\ &\quad + 25(Z_1^{-1}Z_2^{-3}) + 18(Z_1^{-1}Z_2^{-2}) + 24(Z_1^{-1}Z_2^{-3}) + 30(Z_1^{-1}Z_2^{-4}) + 21(Z_1^{-2}Z_2^{-0}) + 28(Z_1^{-2}Z_2^{-1}) + 35(Z_1^{-2}Z_2^{-2}) \\ &\quad + 24(Z_1^{-2}Z_2^{-1}) + 32(Z_1^{-2}Z_2^{-2}) + 40(Z_1^{-2}Z_2^{-3}) + 27(Z_1^{-2}Z_2^{-2}) + 36(Z_1^{-2}Z_2^{-3}) + 45(Z_1^{-2}Z_2^{-4}). \\ Y(Z_1, Z_2) &= 3(Z_1^{-0}Z_2^{-0}) + 10(Z_1^{-0}Z_2^{-1}) + 22(Z_1^{-0}Z_2^{-2}) + 22(Z_1^{-0}Z_2^{-3}) + 15(Z_1^{-0}Z_2^{-4}) + 12(Z_1^{-1}Z_2^{-0}) \\ &\quad + 31(Z_1^{-1}Z_2^{-1}) + 58(Z_1^{-1}Z_2^{-2}) + 49(Z_1^{-1}Z_2^{-3}) + 30(Z_1^{-1}Z_2^{-4}) + 21(Z_1^{-2}Z_2^{-0}) + 52(Z_1^{-2}Z_2^{-1}) + 94(Z_1^{-2}Z_2^{-2}) \\ &\quad + 76(Z_1^{-2}Z_2^{-3}) + 45(Z_1^{-2}Z_2^{-4}). \end{aligned}$$

Step 4 Grouping the related terms

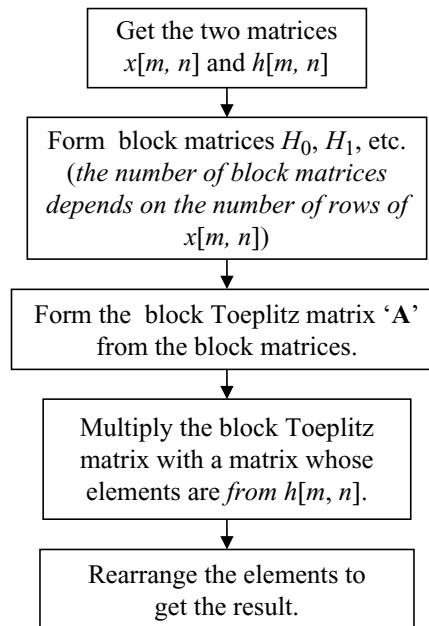
After grouping the related terms, the output is obtained as

$$y(m, n) = \begin{pmatrix} 3 & 10 & 22 & 22 & 15 \\ 12 & 31 & 58 & 49 & 30 \\ 21 & 52 & 94 & 76 & 45 \end{pmatrix}$$

3.4 2D CONVOLUTION THROUGH MATRIX ANALYSIS

2D convolution can be performed through matrix multiplication. To do this, the block Toeplitz matrix has to be formed from one of the matrices. The other matrix is written in column form. Then the multiplication between the block Toeplitz matrix and the matrix written in column form will yield a result similar to 2D convolution. The flow chart to perform 2D convolution through matrix multiplication is given below.

Flow Chart The flow chart to perform linear convolution between two matrices $x[m, n]$ and $h[m, n]$ through the matrix method is given below.



The crucial step is the formation of block matrices H_0, H_1, H_2 , etc. It should be noted that the number of block matrices depends on the number of rows of $x[m, n]$.

Example 3.7 Compute the linear convolution between two matrices $x[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $h[m, n] = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ through the matrix method.

Solution Let the size of the matrix $x[m, n]$ be $M_1 \times N_1$ where M_1 is the number of rows and N_1 is the number of columns of the matrix ' $x[m, n]$ '.

Let the size of the matrix $h[m, n]$ be $M_2 \times N_2$ where M_2 is the number of rows and N_2 is the number of columns of the matrix ' $h[m, n]$ '.

The convolution between $x[m, n]$ and $h[m, n]$ is denoted by $y[m, n]$, where $y[m, n] = x[m, n] * h[m, n]$. The size of the matrix $y[m, n]$ is $M_3 \times N_3$ where $M_3 = M_1 + M_2 - 1$ and $N_3 = N_1 + N_2 - 1$.

In our example, $M_1 = 2, N_1 = 2$ and $M_2 = 2, N_2 = 2$. Hence, the resultant matrix that is, $y[m, n]$ will have size $M_3 = 2 + 2 - 1$ which is 3 and $N_3 = 2 + 2 - 1$ which is 3. The size of the matrix $y[m, n] = 3 \times 3$.

The block matrix is denoted by H_0, H_1, H_2 etc.

The number of block matrices depends on the number of rows of $x[m, n]$. In case of $x[m, n]$, it has two rows which are (1, 2) and (3, 4). Hence, the number of block matrices should be two. Let them be H_0 and H_1 .

Number of zeros to be appended in H_0 = Number of columns in $h[m, n]$ – 1

$x[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. The element (1, 2) which is circled is used to form the block matrix H_0 . This element (1, 2) is to be inserted into H_0 .

Steps in the formation of the block matrix H_0

Step 1 First, the element '1' is inserted in H_0 . It is shown by

$H_0 = [1 \ 0]$. Here, only one zero is inserted as the number of zeros in the block matrix is equal to the number of columns in $h[m, n] - 1$.

Step 2 Then, the second element '2' is inserted as shown below.

$$H_0 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Step 3 Then, the element is shifted to form the matrix H_0 as shown below.

$$H_0 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 2 \end{bmatrix}$$

Steps in the formation of the block matrix H_1

$x[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. The element (3, 4), which is circled, is used to form the block matrix H_1 . This element (3, 4) is to be inserted into H_1 .

Step 1 First, the element '3' is inserted in H_1 . It is shown by $H_1 = [3 \ 0]$. Here, only one zero is inserted as the number of zeros in the block matrix is equal to the number of columns in $h[m, n] - 1$.

Step 2 Now, the second element '4' is inserted as shown below.

$$H_1 = \begin{bmatrix} 3 & 0 \\ 4 & 3 \end{bmatrix}$$

Step 3 Then, the element is shifted to form the matrix H_1 as shown below.

$$H_1 = \begin{bmatrix} 3 & 0 \\ 4 & 3 \\ 0 & 4 \end{bmatrix}$$

Steps in the formation of the block Toeplitz matrix

Let the block Toeplitz matrix be denoted as ' A '. This block Toeplitz matrix ' A ' consists of block matrices H_0 and H_1 .

Number of zeros to be appended in ' A ' = Number of rows of $h[m, n] - 1$

Then, the block Toeplitz matrix A is given by

$$A = \begin{bmatrix} H_0 & O \\ H_1 & H_0 \\ O & H_1 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 4 & 3 & 2 & 1 \\ 0 & 4 & 0 & 2 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

Here, 'O' is not a single value, but it is a group of zeros as indicated by the arrow.

Then, the convolution result is obtained by multiplying the block Toeplitz matrix 'A' with the matrix $h[m, n]$. The matrix $h[m, n]$ is written in columnar form.

$$y[m, n] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 4 & 3 & 2 & 1 \\ 0 & 4 & 0 & 2 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 4 \end{bmatrix} \times \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 5 \\ 16 \\ 12 \\ 22 \\ 60 \\ 40 \\ 21 \\ 52 \\ 32 \end{bmatrix}$$

The result is given by

$$y[m, n] = \begin{bmatrix} 5 & 16 & 12 \\ 22 & 60 & 40 \\ 21 & 52 & 32 \end{bmatrix}$$

which is a 3×3 matrix. The result is written by taking the first three rows and forming the first row, then taking the next three rows and forming the next row, and so on.

The MATLAB command and the corresponding output are shown in Fig. 3.1.

Example 3.8 Perform linear convolution between two matrix $x[m, n] = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ and $h[m, n] = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

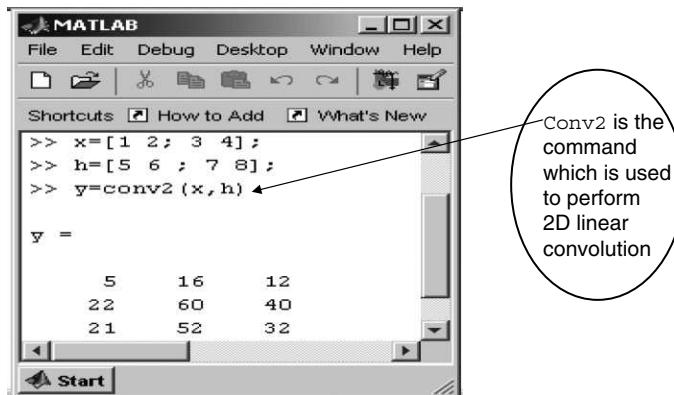


Fig. 3.1 Convolution command in MATLAB

Solution The Step-by-step procedure to perform linear convolution between two matrices $x(m, n)$ and $h(m, n)$ is given below.

$$x[m, n] = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \text{and} \quad h[m, n] = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

The objective is to find $y[m, n] = x[m, n] * h[m, n]$, that is, a linear convolution between $x[m, n]$ and $h[m, n]$. The dimension of the matrix $x[m, n]$ be represented by $M_1 \times N_1$. In this case, $M_1 = 3$ and $N_1 = 3$. The dimension of the matrix $h[m, n]$ is represented by $M_2 \times N_2$. In this case, $M_2 = 3$ and $N_2 = 1$.

The dimension of the matrix $y[m, n]$ which is obtained by the linear convolution of $x[m, n]$ with $h[m, n]$ is represented by $M_3 \times N_3$ where $M_3 = M_1 + M_2 - 1$ and $N_3 = N_1 + N_2 - 1$. In this case, $M_3 = 3 + 3 - 1 = 5$ and $N_3 = 3 + 1 - 1 = 3$. Therefore, the dimension of the resultant matrix should be 5×3 .

Step 1 Formation of the block matrix

The number of the block matrix depends on the number of rows of $x[m, n]$. In this case, $x[m, n]$ has three rows; hence, three block matrices H_0 , H_1 and H_2 have to be found out. The number of zeros to be appended in the formation of H_0 , H_1 and H_2 depends on the number column of the matrix $h[m, n]$.

Number of zeros to be appended in the column of the block matrix = Number of columns in $h[m, n] - 1$

In this case, the number of columns in $h[m, n] = 1$. Hence $1 - 1 = 0$ which implies 'NO' zero has to be appended in the formation of H_0 , H_1 and H_2 .

Step 2 Formation of the block matrix H_0

The matrix H_0 is formed from the rows of $x[m, n]$. The matrix H_0 is given by

$$H_0 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad H_1 = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad \text{and} \quad H_2 = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}. \quad \text{It is to be observed that no zeros are appended in the formation}$$

of H_0 , H_1 and H_2 .

Step 3 Formation of block Toeplitz matrix

Let the block Toeplitz matrix be denoted by 'A'. The elements of the matrix 'A' are H_0 , H_1 and H_2 . It is to be noted that the number of zeros to be appended in the formation of the matrix 'A' depends on the number of rows of $h[m, n]$.

$$\boxed{\text{Number of zeros to be appended in 'A' = Number of rows of } h[m, n] - 1}$$

In our case, the number of rows of $h[m, n] = 3$. Hence $3 - 1 = 2$ zeros have to be appended in the matrix 'A'.

The block Toeplitz matrix $A =$

$$\begin{bmatrix} H_0 & o & o \\ H_1 & H_0 & o \\ H_2 & H_1 & H_0 \\ o & H_2 & H_1 \\ o & o & H_2 \end{bmatrix}$$

Number of zeros appended is two which is the number of rows of $h[n]$ minus one

Then, substituting the value of $H_0 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, $H_1 = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$ and $H_2 = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}$ in the expression of the block

Toeplitz matrix 'A', we get

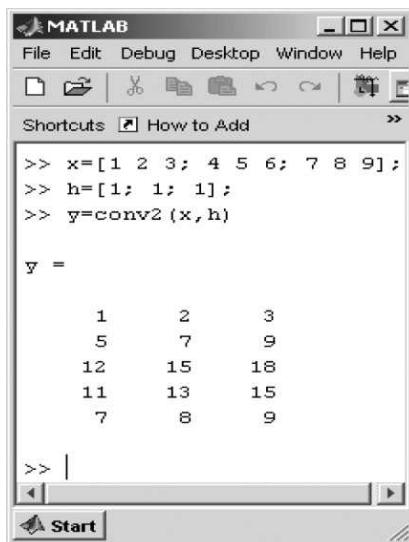
$$A = \begin{bmatrix} H_0 & o & o \\ H_1 & H_0 & o \\ H_2 & H_1 & H_0 \\ o & H_2 & H_1 \\ o & o & H_2 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 0 \\ 4 & 1 & 0 \\ 5 & 2 & 0 \\ 6 & 3 & 0 \\ 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \\ 0 & 7 & 4 \\ 0 & 8 & 5 \\ 0 & 9 & 6 \\ 0 & 0 & 7 \\ 0 & 0 & 8 \\ 0 & 0 & 9 \end{bmatrix}; \quad y[m, n] = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \\ 18 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \\ 7 \\ 9 \\ 12 \\ 15 \\ 18 \\ 11 \\ 13 \\ 15 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

The result is given by rearranging the elements $y[m, n] =$

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 7 & 9 \\ 12 & 15 & 18 \\ 11 & 13 & 15 \\ 7 & 8 & 9 \end{bmatrix}$$

The corresponding MATLAB code for the above problem is given in Fig. 3.2.



The screenshot shows the MATLAB desktop environment. The command window displays the following code and its output:

```

>> x=[1 2 3; 4 5 6; 7 8 9];
>> h=[1; 1; 1];
>> y=conv2(x, h)

y =

```

1	2	3
5	7	9
12	15	18
11	13	15
7	8	9

Fig. 3.2 MATLAB code to compute the 2D linear convolution

Example 3.9 Compute the linear convolution between the two matrices $x[m, n] =$

$$h[m, n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \text{ and}$$

Solution

Number of block matrices depends on the number of rows of $x[m, n]$.

The number of rows of $x[m, n]$ is three. Hence, three block matrices have to be formed. Let it be H_0 , H_1 and H_2 respectively. The number of rows of the block matrices depends on the columns of $h[m, n]$. The number of columns of $h[m, n]$ in this case is two; hence one zero has to be appended.

Step 1 Formation of block matrix

$$H_0 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 2 \\ 0 & 3 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 4 & 0 \\ 5 & 4 \\ 6 & 5 \\ 0 & 6 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 7 & 0 \\ 8 & 7 \\ 9 & 8 \\ 0 & 9 \end{bmatrix}$$

H_0 is formed from the first row of $x[m,n]$

H_1 is formed from the second row of $x[m,n]$

H_2 is formed from the third row of $x[m,n]$

Step 2 Formation of block Toeplitz matrix

The block Toeplitz matrix is denoted by 'A'. The number of rows to be appended in 'A' is governed by the number of rows of $h[m, n]$. In this example, the number of rows of $h[m, n]$ is three; hence two zeros have to be appended.

$$A = \begin{bmatrix} H_0 & O & O \\ H_1 & H_0 & O \\ H_2 & H_1 & H_0 \\ O & H_2 & H_1 \\ O & O & H_2 \end{bmatrix}$$

Two zeros (size same as H_0) are appended as shown by encircling.

$$A = \begin{bmatrix} H_0 & O & O \\ H_1 & H_0 & O \\ H_2 & H_1 & H_0 \\ O & H_2 & H_1 \\ O & O & H_2 \end{bmatrix} A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 \\ 5 & 4 & 2 & 1 & 0 & 0 \\ 6 & 5 & 3 & 2 & 0 & 0 \\ 0 & 6 & 0 & 3 & 0 & 0 \\ 7 & 0 & 4 & 0 & 1 & 0 \\ 8 & 7 & 5 & 4 & 2 & 1 \\ 9 & 8 & 6 & 5 & 3 & 2 \\ 0 & 9 & 0 & 6 & 0 & 3 \\ 0 & 0 & 7 & 0 & 4 & 0 \\ 0 & 0 & 8 & 7 & 5 & 4 \\ 0 & 0 & 9 & 8 & 6 & 5 \\ 0 & 0 & 0 & 9 & 0 & 6 \\ 0 & 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 8 & 7 \\ 0 & 0 & 0 & 0 & 9 & 8 \\ 0 & 0 & 0 & 0 & 0 & 9 \end{bmatrix}$$

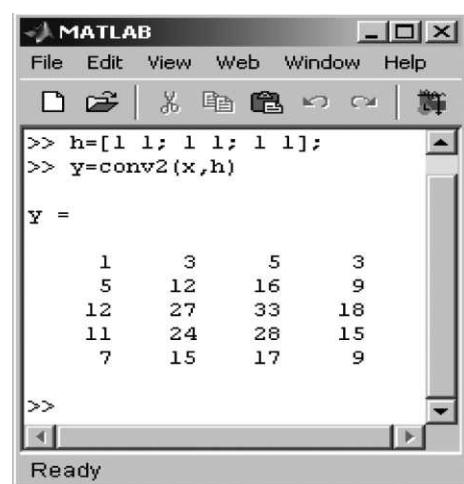
The resultant matrix $y[m, n]$ is obtained by multiplying the matrix 'A' with $h[m, n]$ written in columnar form.

$$y[m, n] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 \\ 5 & 4 & 2 & 1 & 0 & 0 \\ 6 & 5 & 3 & 2 & 0 & 0 \\ 0 & 6 & 0 & 3 & 0 & 0 \\ 7 & 0 & 4 & 0 & 1 & 0 \\ 8 & 7 & 5 & 4 & 2 & 1 \\ 9 & 8 & 6 & 5 & 3 & 2 \\ 0 & 9 & 0 & 6 & 0 & 3 \\ 0 & 0 & 7 & 0 & 4 & 0 \\ 0 & 0 & 8 & 7 & 5 & 4 \\ 0 & 0 & 9 & 8 & 6 & 5 \\ 0 & 0 & 0 & 9 & 0 & 6 \\ 0 & 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 8 & 7 \\ 0 & 0 & 0 & 0 & 9 & 8 \\ 0 & 0 & 0 & 0 & 0 & 9 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 3 \\ 12 \\ 16 \\ 9 \\ 27 \\ 33 \\ 18 \\ 11 \\ 24 \\ 28 \\ 15 \\ 7 \\ 15 \\ 17 \\ 9 \end{bmatrix}$$

The matrix $y[m, n]$ should be of size 5×4 which is given below:

$$y[m, n] = \begin{bmatrix} 1 & 3 & 5 & 3 \\ 5 & 12 & 16 & 9 \\ 12 & 27 & 33 & 18 \\ 11 & 24 & 28 & 15 \\ 7 & 15 & 17 & 9 \end{bmatrix}$$

The corresponding MATLAB code is given in Fig. 3.3.



```

>> h=[1 1; 1 1; 1 1];
>> y=conv2(x,h)

y =

```

1	3	5	3
5	12	16	9
12	27	33	18
11	24	28	15
7	15	17	9

Fig. 3.3 MATLAB code for 2D convolution

Example 3.10 Perform the linear convolution between two matrices $x[m, n] = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ and $h[m, n] = [3 \ 4 \ 5]$.

Solution The size of the input matrix $x[m, n]$ is 3×3 . The size of the matrix $h[m, n]$ is 1×3 . The resultant matrix should have a dimension of 3×5 .

Step 1 Formation of block matrix

The number of block matrices depend on the rows of $x[m, n]$. In this example, the number of rows of $x[m, n]$ is three; hence three block matrices have to be formed. Let the three block matrices be H_0 , H_1 and H_2 respectively. H_0 is formed from the first row of the matrix $x[m, n]$; H_1 is formed from the second row of the matrix $x[m, n]$ and H_2 is formed from the third row of the matrix $x[m, n]$. The three matrices are shown below.

$$H_0 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \\ 0 & 3 & 2 \\ 0 & 0 & 3 \end{bmatrix}$$

The elements of H_0 are formed from the first row of $x[m, n]$.

$$H_1 = \begin{bmatrix} 4 & 0 & 0 \\ 5 & 4 & 0 \\ 6 & 5 & 4 \\ 0 & 6 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

The elements of H_1 are formed from the second row of $x[m, n]$.

$$H_2 = \begin{bmatrix} 7 & 0 & 0 \\ 8 & 7 & 0 \\ 9 & 8 & 7 \\ 0 & 9 & 8 \\ 0 & 0 & 9 \end{bmatrix}$$

The elements of H_2 are formed from the third row of $x[m, n]$.

Step 2 Formation of block Toeplitz matrix

The Block Toeplitz matrix is denoted by ' A '. The number of rows to be appended in ' A ' is governed by the number of rows of $h[m, n]$. In this example, the number of rows of $h[m, n]$ is one. Hence, NO zero has to be appended. Then substituting the values of H_0 , H_1 and H_2 in the matrix ' A ', we get

$$A = \begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix}$$

Then, the resultant matrix is obtained by multiplying the matrix 'A' with the matrix whose elements are from $h[m, n]$.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \\ 0 & 3 & 2 \\ 0 & 0 & 3 \\ 4 & 0 & 0 \\ 5 & 4 & 0 \\ 6 & 5 & 4 \\ 0 & 6 & 5 \\ 0 & 0 & 6 \\ 7 & 0 & 0 \\ 8 & 7 & 0 \\ 9 & 8 & 7 \\ 0 & 9 & 8 \\ 0 & 0 & 9 \end{bmatrix}$$

Note: No zeros are appended in the formation of the matrix 'A' since the number of rows of $h[m, n]$ is only one.

Then, the output $y[m, n]$ is obtained by multiplying the matrix 'A' with a matrix whose elements are obtained from the matrix $h[m, n]$.

$$y[m, n] = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \\ 0 & 3 & 2 \\ 0 & 0 & 3 \\ 4 & 0 & 0 \\ 5 & 4 & 0 \\ 6 & 5 & 4 \\ 0 & 6 & 5 \\ 0 & 0 & 6 \\ 7 & 0 & 0 \\ 8 & 7 & 0 \\ 9 & 8 & 7 \\ 0 & 9 & 8 \\ 0 & 0 & 9 \end{bmatrix} \times \begin{bmatrix} 3 \\ 10 \\ 22 \\ 22 \\ 15 \\ 12 \\ 31 \\ 58 \\ 49 \\ 30 \\ 21 \\ 52 \\ 94 \\ 76 \\ 45 \end{bmatrix}$$

The resultant matrix size is 3×5 which is given by $y[m, n] = \begin{bmatrix} 3 & 10 & 22 & 22 & 15 \\ 12 & 31 & 58 & 49 & 30 \\ 21 & 52 & 94 & 76 & 45 \end{bmatrix}$

Example 3.11 Prove that convolution of any signal with an impulse signal gives rise to the same signal.

Solution The aim is to prove $x(m, n) * \delta(m, n) = x(m, n)$.

Proof The convolution of two signals $x(m, n)$ and $h(m, n)$ is given by

$$y(m, n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k, l)h(m-k, n-l)$$

In our case, $h(m, n) = \delta(m, n)$; thus substituting for $h(m, n)$ in the above equation, we get

$$y(m, n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k, l)\delta(m-k, n-l)$$

We know that $\delta(m, n)$ will have a value of one only at $m = 0$ and $n = 0$ which is given as

$$\delta(m, n) = \begin{cases} 1, & m = 0 \text{ and } n = 0 \\ 0, & \text{otherwise} \end{cases}$$

Substituting $k = m$ and $l = n$ in the expression of $y[m, n]$, we get

$$y(m, n) = x(m, n)$$

Example Let $x[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $h[m, n] = [1]$.

Step 1 Formation of block matrix

The number of block matrices depends on the number of rows of $x[m, n]$. In this case, two block matrices are needed. Let them be H_0 and H_1 .

$H_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $H_1 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$. The matrix H_0 is formed from the first row of $x[m, n]$ and the matrix H_1 is formed from the second row of $x[m, n]$.

Step 2 Formation of block Toeplitz matrix

The block Toeplitz matrix is denoted by ' A '. The number of zeros to be appended depends on the rows of $h[m, n]$. In our case, no zeros have to be appended and hence the matrix ' A ' is given by

$$A = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}$$

Then, substituting the values of H_0 and H_1 , we get

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

The output matrix $y[m, n]$ is obtained by multiplying the matrix 'A' with the matrix whose elements are from the matrix $y[m, n]$.

Step 3 Output matrix $y[m, n]$

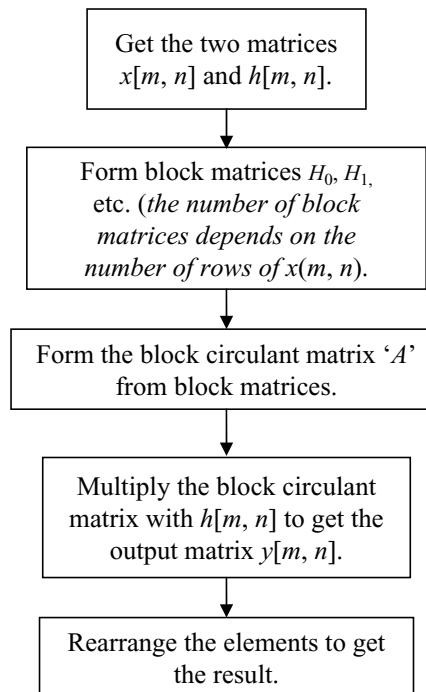
$$y[m, n] = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \times 1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$\text{The output matrix } y[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

3.5 CIRCULAR CONVOLUTION THROUGH MATRIX METHOD

Circular convolution can be performed for periodic signals. The circular convolution between two signals $x(m, n)$ and $h(m, n)$ is given by $x(m, n) \otimes h(m, n)$.

The flow chart to perform circular convolution between two matrices $x[m, n]$ and $h[m, n]$ is given below.



Example 3.12 Perform the circular convolution between two matrices $x[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and

$$h[m, n] = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}.$$

Solution

Step 1 Formation of block matrix

First, the block matrices are formed. Then the block circulant matrix is formed. The number of rows in the matrix $x[m, n]$ is two; hence two block matrices have to be formed.

$$H_0 = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

and $H_1 = \begin{bmatrix} 3 & 4 \\ 4 & 3 \end{bmatrix}$

Note: H_0 is formed from the first row of $x[m, n]$ written in circulant form.

Note: H_1 is formed from the second row of $x[m, n]$ written in circulant form.

Step 2 Formation of block circulant matrix

Let the block circulant matrix be formed from H_0 and H_1 .

$$A = \begin{bmatrix} H_0 & H_1 \\ H_1 & H_0 \end{bmatrix}$$

Substituting the values of H_0 and H_1 , we get

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

Step 3 Computation of output matrix $y[m, n]$

$$y[m, n] = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} = \begin{bmatrix} 70 \\ 68 \\ 62 \\ 60 \end{bmatrix}$$

The output $y[m, n]$ is a 2×2 matrix which is given by

$$y[m, n] = \begin{bmatrix} 70 & 68 \\ 62 & 60 \end{bmatrix}.$$

The dimension of $y[m, n]$ is the same as $x[m, n]$ and $h[m, n]$.

Example 3.13 Prove that circular convolution can be expressed as linear convolution plus alias.

Solution Let us consider two signals $x[m, n]$ and $h[m, n]$. The two signals are given by $x[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

and $h[m, n] = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$. The result of linear convolution between $x[m, n]$ and $h[m, n]$ is given by

$y[m, n] = x[m, n] * h[m, n]$. The result of linear convolution between $x[m, n]$ and $h[m, n]$ is given by

$y[m, n] = \begin{bmatrix} 5 & 16 & 12 \\ 22 & 60 & 40 \\ 21 & 52 & 32 \end{bmatrix}$. The matrix $y[m, n]$ has three rows and three columns. It is known that in the

case of circular convolution, the length of the result is 2×2 .

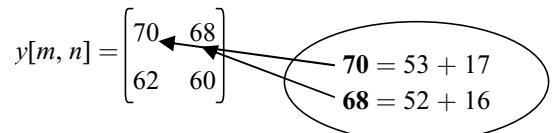
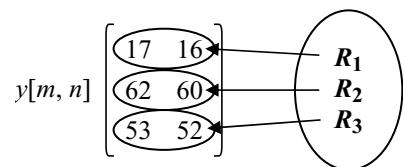
Step 1

$$y[m, n] = \begin{bmatrix} C_1 & C_2 & C_3 \\ (5) & (16) & (12) \\ (22) & (60) & (40) \\ (21) & (52) & (32) \end{bmatrix}$$

The three columns are denoted by C_1 , C_2 and C_3 . Here, it is noted that the third column is an alias of the first column. Hence, the third column (C_3) is added with the first column (C_1) and the result is shown below.

Step 2 The three rows are denoted by R_1 , R_2 and R_3 . Here, it is noted that the third row is an alias of the first row. Hence, the row R_1 is added with the row R_3 to get the result of circular convolution between $x[m, n]$ and $h[m, n]$ which is given by

From this example, it can be concluded that circular convolution is nothing but linear convolution plus alias.



3.6 APPLICATION OF CIRCULAR CONVOLUTION

Circular convolution can be used to perform the operation of ‘zooming’ which is widely used in digital cameras.

The concept of zooming can be illustrated as explained below.

Consider two matrices $x[m, n]$ and $h[m, n]$ as shown below.

$x[m, n] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $h[m, n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. Then the circular convolution between $x[m, n]$ and $h[m, n]$ is performed as below.

```

1. x = [1 0; 0 0]; %Input matrix 1
2. h = [1 1; 1 1]; %Input matrix 2
3. x1 = fft2(x); %FFT of the first matrix
4. h1 = fft2(h); %FFT of the second matrix
5. y1 = x1.*h1; %Element by element multiplication
6. res = ifft2(y1) %Inverse FFT

```

The input and the output of the above code is shown below.

$$x[m, n] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad h[m, n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Note that the zeros (which are rounded) in $x[m, n]$ are replaced by ones as rounded in 'res'.

$$\text{res} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

If $h[m, n] = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$ and $x[m, n] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ then the result of circular convolution between $x[m, n]$ and $h[m, n]$ is

is shown below.

$x[m, n] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $h[m, n] = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$. The result of circular convolution between $x[m, n]$, and $h[m, n]$ is

given as 'res' which is given by $\text{res} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$. Note that zeros in the input is replaced by '2'. Hence, circular

convolution performs some sort of 'interpolation' which can be used to perform the 'zooming' operation. The MATLAB code to perform 'zooming' operation through circular convolution is shown below.

Line	Code	Comments
No		
1. clc		
2. clear all		
3. close all		
4. a=imread('cameraman.tif');		
5. s=input('Enter the resizing ratio:');		
6. [m n]=size(a);		
7. s1=s*m;		
8. re=zeros(s1, s*n);		
9. for i=1:m,		

```
10. for j=1:n,
11. k=s*(i-1);
12. l=s*(j-1);
13. re(k+1, l+1)=a(i, j);           % To get the pixel value in the
   interpolated matrix
14. end
15. end
16. i=1;
17. while (i<=(s1))
18. j=1;
19. while (j<=(s*n))
20. x=ones(s, s);
21. for p=1:s,
22. for q=1:s,
23. c(p, q)=re(i, j); % To extract the pixel matrix
24. j=j+1;
25. end
26. i=i+1;
27. j=j-s;
28. end
29. z=ifft2(fft2(c).*fft2(x)); % Circular convolution using DFT and
   IDFT
30. i=i-s;
31. for p=1:s,
32. for q=1:s,
   re(i, j)=z(p, q); % To generate the interpolated matrix
   j=j+1;
33. end
34. i=i+1;
35. j=j-s;
36. end
37. i=i-s;
38. j=j+s;
39. end
40. i=i+s;
41. end
42. figure, imshow(uint8(a)), title('original input image to be
   interpolated');
43. figure, imshow(uint8(re)), title('Interpolated image');
```

On executing the above code for ‘checkerboard’, using the resizing ratio as *two* we get the output in Fig. 3.4.

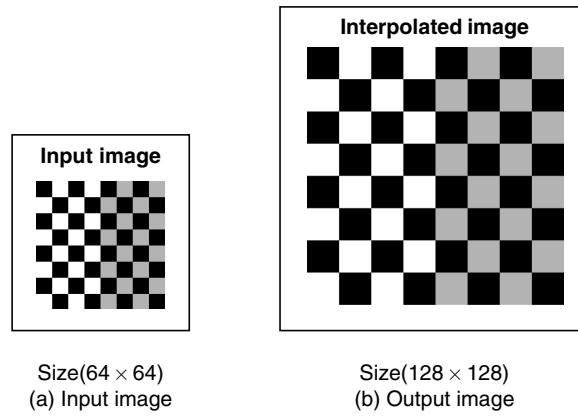
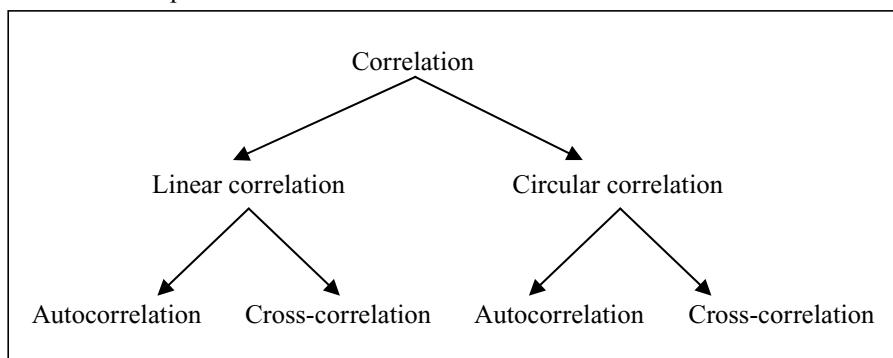


Fig. 3.4 Circular convolution as interpolation operator

3.7 2D CORRELATION

Correlation is a signal-matching technique. It is an important component of radar, sonar, digital communications and many other systems. Correlation is mathematically similar to convolution. Correlation can be classified into two types as linear correlation and circular correlation. Correlation can be performed either in the spatial domain or in the spectral domain.



Theory Correlation is a mathematical operation that is similar to convolution. Correlation is basically used to obtain similarity between two signals. Autocorrelation is basically a signal correlated with itself. The amplitude of each sample in the cross-correlation is a measure of how much one signal resembles the other. The method to perform linear and circular correlation is given in detail in this section.

3.7.1 Correlation through Matrix Method

The step-by-step approach to the computation of linear and circular correlation through the formation of block Toeplitz matrix and block Circulant matrix is given in this section. The correlation can be performed

in terms of convolution. While performing correlation through convolution, the second sequence $x_2[m, n]$ is folded to form $x_2[-m, -n]$ and it is convolved with the first sequence $x_1[m, n]$.

Let us find the convolution between $x[n]$ and $x^*[-n]$. The formula to compute the convolution between $x[n]$ and $x^*[-n]$ is given by

$$x[n]*x^*[-n] = \sum_{k=-\infty}^{\infty} x[k]x^*[-(n-k)]$$

$$x[n]*x^*[-n] = \sum_{k=-\infty}^{\infty} x[k]x^*[(k-n)] = r_{xx}[n] \text{ where } r_{xx}[n] \text{ is the autocorrelation of the signal } x[n].$$

The same idea can be extended to a two-dimensional signal. The aim is to determine the autocorrelation of the two-dimensional signal $x[m, n]$. The autocorrelation of the signal can be performed in terms of convolution. When correlation is performed in terms of convolution, the second sequence is folded and then convolution is performed with respect to the first sequence. In this case, both the signals are $x[m, n]$.

$$x[m, n]*x^*[-m, -n] = \sum_a \sum_b x(a, b)x^*[-(m-a), -(n-b)]$$

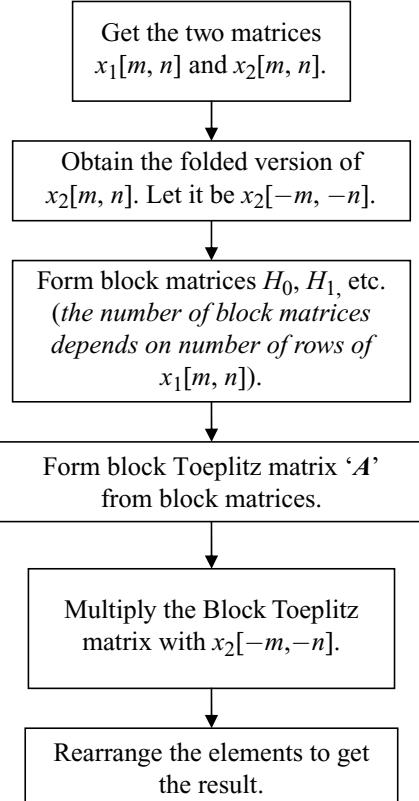
$$x[m, n]*x^*[-m, -n] = \sum_a \sum_b x(a, b)x^*[(a-m), (b-n)] = r_{xx}(m, n)$$

If $m = 0$ and $n = 0$, we get

$$r_{xx}(0, 0) = \sum_a \sum_b x(a, b)x^*(a, b) = \sum_a \sum_b |x(a, b)|^2 \text{ which is nothing but the energy of the signal } x[m, n].$$

Thus, autocorrelation can be performed in terms of convolution. Similarly, cross-correlation can be performed in terms of convolution.

The flow chart to perform correlation between two signals $x_1[m, n]$ and $x_2[m, n]$ through the matrix method is given below.



Example 3.14 Determine the correlation between two matrices $x_1[m, n] = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$ and $x_2[m, n] = \begin{bmatrix} 1 & 5 \\ 2 & 3 \end{bmatrix}$.

Solution Correlation is same as convolution except that the second sequence $x_2[m, n]$ is folded. That is, $x_2[m, n]$ is folded so that it becomes $x_2[-m, -n]$ and then we find the convolution between $x_1[m, n]$ and $x_2[-m, -n]$ which is the correlation between $x_1[m, n]$ and $x_2[m, n]$.

Correlation between $x_1[m, n]$ and $x_2[m, n] = x_1[m, n] * x_2[-m, -n]$

Step 1

From the given $x_2[m, n]$ determine $x_2[-m, -n]$.

Step 1a To get the folded version of $x_2[m, n]$, first fold $x_2[m, n]$ columnwise to get $x_2[m, -n] = \begin{bmatrix} 5 & 1 \\ 3 & 2 \end{bmatrix}$.

Step 1b Then fold $x_2[m, -n]$ along row-wise to get $x_2[-m, -n]$ which is given by $x_2[-m, -n] = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix}$.

Step 2

Now we have to perform linear convolution between $x_1[m, n]$ and $x_2[-m, -n]$.

Step 2a Formation of block matrix The number of block matrices depends on the number of rows of $x_1[m, n]$. In this case, the number of columns of $x_2[m, n]$ is two. Hence, two block matrices have to be formed. Let the two block matrices be H_0 and H_1 respectively.

Step 2b Formation of block matrix H_0 The block matrix H_0 formed from the first row of $x_1[m, n]$ is given below.

$$H_0 = \begin{bmatrix} 3 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix}$$

Step 2b Formation of block matrix H_1 The block matrix H_1 formed from the second row of $x_1[m, n]$ is given below.

$$H_1 = \begin{bmatrix} 2 & 0 \\ 4 & 2 \\ 0 & 1 \end{bmatrix}$$

Step 3 Formation of block Toeplitz matrix

Let the block Toeplitz matrix be denoted by A . The number of zeros to be appended in the block matrix A depends on the number of rows of $x_2[m, n]$. In this case, $x_2[m, n]$ has two rows; hence one zero has to be appended. The matrix A is given by

$A = \begin{bmatrix} H_0 & 0 \\ H_1 & H_0 \\ 0 & H_1 \end{bmatrix}$. Then, substituting the values of H_0 and H_1 in the matrix A , we get

$$A = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 \\ 4 & 2 & 1 & 3 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

The resultant matrix $y[m, n]$ is obtained by multiplying the matrix A with a matrix whose elements are from $x_2[-m, -n]$.

$$y[m, n] = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 \\ 4 & 2 & 1 & 3 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix} \times \begin{bmatrix} 3 \\ 2 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 9 \\ 2 \\ 21 \\ 24 \\ 9 \\ 10 \\ 22 \\ 4 \end{bmatrix}$$

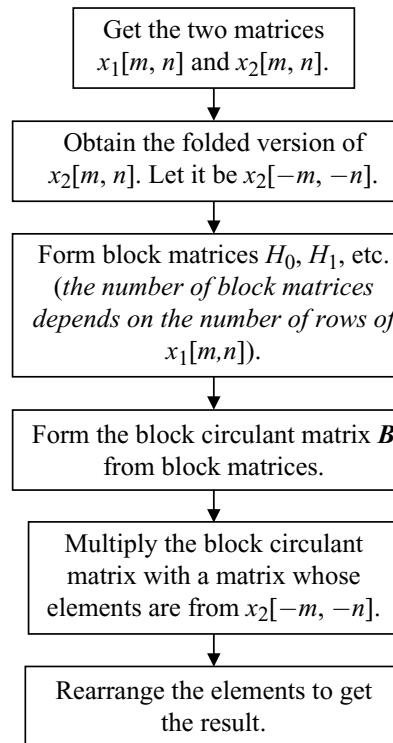
The result $y[m, n]$ is a 3×3 matrix which is given by

$$y[m, n] = \begin{bmatrix} 9 & 9 & 2 \\ 21 & 24 & 9 \\ 10 & 22 & 4 \end{bmatrix}$$

3.7.2 Circular Correlation through Matrix Method

Circular correlation can be performed through the formation of a block circulant matrix. The step-by-step approach to the computation of circular correlation through the formation of a block circulant matrix is given in this section.

The flow chart to perform circular cross-correlation between two matrices $x_1[m, n]$ and $x_2[m, n]$ through the matrix method is given below.



Example 3.15 Determine the circular correlation between the two matrices $x_1[m, n] = \begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix}$ and $x_2[m, n] = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$.

Solution To find the circular correlation between $x_1[m, n]$ with $x_2[m, n]$, it is sufficient to perform circular convolution between $x_1[m, n]$ and $x_2[-m, -n]$.

Step 1 Formation of block matrices

The number of block matrices depends on the number of rows of $x_1[m, n]$. In this case, there are two rows in $x_1[m, n]$; hence, two block matrices are required. Let the two block matrices be H_0 and H_1 respectively.

Step 1a Formation of H_0 The block matrix H_0 formed from the first row of $x_1[m, n]$ is given below.

$$H_0 = \begin{bmatrix} 1 & 5 \\ 5 & 1 \end{bmatrix}$$

Step 1b Formation of H_1 The block matrix H_1 formed from the second row of $x_1[m, n]$ is given below.

$$H_1 = \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix}$$

Step 2 Formation of block circulant matrix

The block circulant matrix B is formed from the block matrices H_0 and H_1 . The number of zeros to be padded in the block circulant matrix B depends on the number of rows of $h[-m, -n]$. In this problem, the number of rows of $h[-m, -n]$ is two; hence one zero has to be padded in the block circulant matrix B .

$$B = \begin{bmatrix} H_0 & H_1 \\ H_1 & H_0 \end{bmatrix}$$

Then substituting the values of the block matrices which are obtained from steps 1a and 1b,

$$B = \begin{bmatrix} 1 & 5 & 2 & 4 \\ 5 & 1 & 4 & 2 \\ 2 & 4 & 1 & 5 \\ 4 & 2 & 5 & 1 \end{bmatrix}$$

Step 3 The resultant matrix $y[m, n]$

The resultant matrix $y[m, n]$ is obtained by multiplying the matrix B with the matrix whose elements are the values obtained from $x_2[-m, -n]$.

Step 3a Formation of $x_2[-m, -n]$ from $x_2[m, n]$

In this step, we wish to obtain $x_2[-m, -n]$ from $x_2[m, n]$ by folding $x_2[m, n]$ both row and columnwise.

From $x_2[m, n] = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$, we first fold $x_2[m, n]$ columnwise to get $x_2[m, -n]$.

$x_2[m, -n] = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$. Then we fold $x_2[m, -n]$ row-wise to get $x_2[-m, -n]$ which is given by

$$x_2[-m, -n] = \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix}$$

Step 3b Multiplying matrix B with a matrix whose elements are elements from $x_2[-m, -n]$, we get

$$y[m, n] = \begin{bmatrix} 1 & 5 & 2 & 4 \\ 5 & 1 & 4 & 2 \\ 2 & 4 & 1 & 5 \\ 4 & 2 & 5 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 4 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 37 \\ 23 \\ 35 \\ 25 \end{bmatrix}$$

The matrix $y[m, n] = \begin{bmatrix} 37 & 23 \\ 35 & 25 \end{bmatrix}$

3.7.3 Circular Autocorrelation between Two Matrices

Let us find circular autocorrelation between the two matrices $x_1[m, n] = \begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix}$ and $x_2[m, n] = \begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix}$.

Since both the sequences are same, we determine the autocorrelation between $x_1[m, n]$ and $x_2[m, n]$.

Step 1**Form the block matrices**

The number of block matrices depends on the number of rows of $x_1[m, n]$. In this case, the number of rows of $x_1[m, n] = 2$. Hence, two block matrices H_0, H_1 have to be formed.

Step 1a Formation of the block matrix H_0

$$H_0 = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$$

Step 1b Formation of block matrix H_1

$$H_1 = \begin{bmatrix} 1 & 5 \\ 5 & 1 \end{bmatrix}$$

Step 2**Formation of block circulant matrix B**

$B = \begin{bmatrix} H_0 & H_1 \\ H_1 & H_0 \end{bmatrix}$. Now substituting the values of H_0, H_1 , etc., we get

$$B = \begin{bmatrix} 3 & 2 & 1 & 5 \\ 2 & 3 & 5 & 1 \\ 1 & 5 & 3 & 2 \\ 5 & 1 & 2 & 3 \end{bmatrix}.$$

Now multiplying the block circulant matrix B with $x_2[-m, -n]$, we get

$$y[m, n] = \begin{bmatrix} 3 & 2 & 1 & 5 \\ 2 & 3 & 5 & 1 \\ 1 & 5 & 3 & 2 \\ 5 & 1 & 2 & 3 \end{bmatrix} \times \begin{bmatrix} 5 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 34 \\ 26 \\ 22 \\ 39 \end{bmatrix}$$

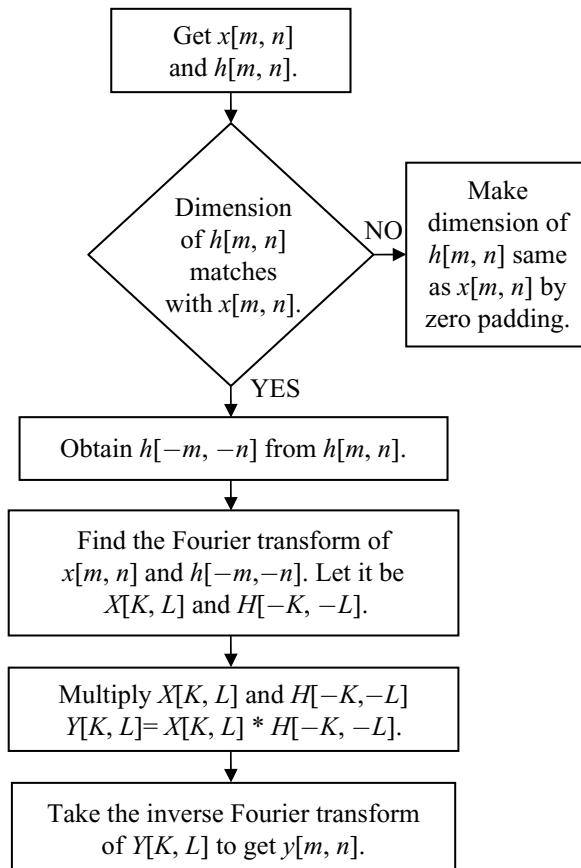
The elements are rearranged to get the result

$$y[m, n] = \begin{bmatrix} 34 & 26 \\ 22 & 39 \end{bmatrix}$$

3.7.4 Circular Correlation through Transform Method

The circular correlation between two signals can be determined through the transform method. The flow chart to determine the circular correlation between two signals is shown below.

Flow Chart to Determine Circular Correlation between Two Signals $x[m, n]$ and $h[m, n]$ through Transform The flow chart to get correlation (linear) between two signals $x[m, n]$ and $h[m, n]$ is shown below.



Example 3.16 Perform the correlation between two signals $x[m, n] = \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$ and $h[m, n] = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$ through the transform method.

Solution The solution to this problem is obtained by following the steps given in the flow chart.

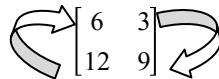
First, check whether the dimension of $h[m, n]$ matches with the dimension of $x[m, n]$. In our case, the dimension of $h[m, n]$ is the same as the dimension of $x[m, n]$.

Step 1 Obtain $h[-m, -n]$ from $h[m, n]$.

Given $h[m, n] = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$. From this, $h[-m, -n]$ is obtained by folding along rows and folding along columns.

$h[m, n] \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$ Folding $h[m, n]$ columnwise, we get $h[m, -n] = \begin{bmatrix} 6 & 3 \\ 12 & 9 \end{bmatrix}$.

Now folding $h[m, -n]$ row-wise, we get $h[-m, -n]$.



$$h[m, -n] = \begin{bmatrix} 6 & 3 \\ 12 & 9 \end{bmatrix}. \text{ Now folding } h[m, -n] \text{ row-wise, } h[-m, -n] = \begin{bmatrix} 12 & 9 \\ 6 & 3 \end{bmatrix}$$

Step 2 Obtain the Fourier transform of $x[m, n]$. Let it be $X[K, L]$.

$$x[m, n] = \begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix} \xrightarrow{\text{Fourier transform}} X[K, L] = \begin{bmatrix} 50 & -10 \\ -20 & 0 \end{bmatrix}$$

Step 3 Obtain Fourier transform of $h[-m, -n]$. Let it be $H[-K, -L]$.

$$h[-m, -n] = \begin{bmatrix} 12 & 9 \\ 6 & 3 \end{bmatrix} \xrightarrow{\text{Fourier transform}} H[-K, -L] = \begin{bmatrix} 30 & 6 \\ 12 & 0 \end{bmatrix}$$

Step 4 Multiply the spectrums $X[K, L]$ and $H[-K, -L]$. Let it be $Y[K, L]$. $Y[K, L] = X[K, L] * H[-K, -L]$

$$Y[K, L] = \begin{bmatrix} 1500 & -60 \\ -240 & 0 \end{bmatrix}$$

Step 5 Obtain the inverse Fourier Transform of $Y[K, L]$ to get $y[m, n]$

$$y[m, n] \xrightarrow{\text{Inverse fourier transform}} Y[K, L]$$

$$y[m, n] = \begin{bmatrix} 300 & 330 \\ 420 & 450 \end{bmatrix}.$$

The MATLAB code for this problem and the corresponding output are shown in Fig. 3.5.

```
%This program computes the circular correlation between two signals
clc
clear all
close all
x=[5 10; 15 20]; %First signal x[m,n]
h=[3 6; 9 12]; %Second signal h[m,n]
h1=flipr(h); %To fold the signal along column-wise
h2=flipud(h1); %To fold the signal along row-wise
x1=fft2(x); %Fourier transform of x[m,n]
h3=fft2(h2); %Fourier transform of h[-m,-n];
y1=x1.*h3; %Multiplication of spectrum of x[m,n] and h[-m,-n]
%to get Y [K,L]
y2=ifft2(y1); %Inverse Fourier transform of Y[K,L] yields
y[m,n]
```

The corresponding output which is the variable $y2$ is given below.

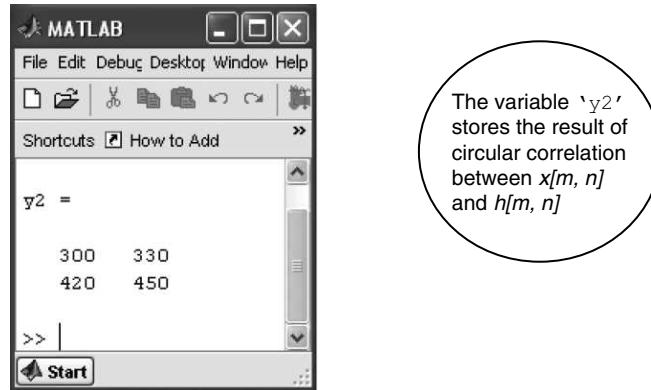
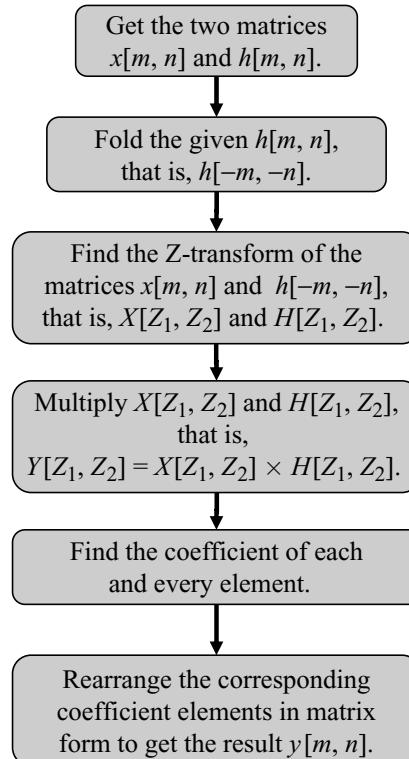


Fig. 3.5 Circular correlation between two matrices

3.7.5 2D Auto-correlation through Z-Transform

This section deals with the computation of auto-correlation between similar 2D signals through 2D Z-transform.

Flow Chart The chart to perform linear correlation, that is $y[m, n]$, between two matrices $x[m, n]$ and $h[m, n]$ through Z-transform is given below.



Example 3.17 Compute the linear auto-correlation between the two given matrices $x[m, n]$ and $h[m, n]$.

$$x[m, n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ and } h[m, n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Solution Let the size of the matrix $x[m, n]$ be $M_1 \times N_1$, where M_1 is the number of rows and N_1 is the number of columns in the matrix $x[m, n]$. Let the size of the matrix $h[m, n]$ be $M_2 \times N_2$, where M_2 is the number of rows and N_2 is the number of columns in the matrix $h[m, n]$. The correlation is convolution between $x[m, n]$ and $h[-m, -n]$ is denoted by $y[m, n]$, where

$$y[m, n] = x[m, n] \otimes h[-m, -n]$$

The size of the matrix $y[m, n]$ is $M_3 \times N_3$, where $M_3 = M_1 + M_2 - 1$ and $N_3 = N_1 + N_2 - 1$.

In this example, $M_1 = 2$, $N_1 = 2$ and $M_2 = 2$, $N_2 = 2$. Hence, the resultant matrix that is $y[m, n]$ will have the size $M_3 = 2 + 2 - 1$ which is 3 and $N_3 = 2 + 2 - 1$ which is 3. The size of the matrix $y[m, n] = 3 \times 3$.

Step 1 Fold the signal $h[m, n]$ (convert the signal to the form of $h[-m, -n]$)

Before the signal $h[m, n]$ is folded, the representation of the element's position in the matrix with the size of 2×2 looks like

$$h[m, n] = \begin{bmatrix} (0,0) & (0,1) \\ 1 & 1 \\ (1,0) & (1,1) \\ 1 & 1 \end{bmatrix}$$

Here, in the 2×2 matrix, 'm' takes a value '0' and '1', and 'n' takes a value of '0' and '1'.

After the signal $h[m, n]$ is folded, the representation of the element's position in the matrix with the size of 2×2 , looks like

$$h[-m, -n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Step 2 Take Z-transform of the given matrices

Method to take Z-transform,

$$x[m, n] = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}$$

A_{00} is the element of the matrix $x[m, n]$ at the position $m = 0$ and $n = 0$; $x[0, 0] = A_{00}$.

A_{01} is the element of the matrix $x[m, n]$ at the position $m = 0$ and $n = 1$; $x[0, 1] = A_{01}$.

A_{10} is the element of the matrix $x[m, n]$ at the position $m = 1$ and $n = 0$; $x[1, 0] = A_{10}$.

A_{11} is the element of the matrix $x[m, n]$ at the position $m = 1$ and $n = 1$; $x[1, 1] = A_{11}$.

The simplified formulae for Z-transform for the matrix with the size of $m \times n$ is given below,

$$Z\{x[m, n]\} = X[Z_1, Z_2] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} A_{mn} Z_1^{-m} Z_2^{-n}$$

For a 2×2 matrix, after transforming the signal, the non-zero values will be only at $Z\{x[0, 0]\}$, $Z\{x[0, 1]\}$, $Z\{x[1, 0]\}$ and $Z\{x[1, 1]\}$. So the Z-transform of a 2×2 matrix will be,

$$X[Z_1, Z_2] = Z\{x[0, 0]\} + Z\{x[0, 1]\} + Z\{x[1, 0]\} + Z\{x[1, 1]\}$$

where,

$$\begin{aligned} Z\{x[0, 0]\} &= A_{00} \times Z_1^{-0} Z_2^{-0} = A_{00} \\ Z\{x[0, 1]\} &= A_{01} \times Z_1^{-0} Z_2^{-1} = A_{01} Z_2^{-1} \\ Z\{x[1, 0]\} &= A_{10} \times Z_1^{-1} Z_2^{-0} = A_{10} Z_1^{-1} \\ Z\{x[1, 1]\} &= A_{11} \times Z_1^{-1} Z_2^{-1} = A_{11} Z_1^{-1} Z_2^{-1} \end{aligned}$$

In this example, it is given that

$$x[m, n] = \begin{bmatrix} (0,0) & (0,1) \\ 1 & 1 \\ (1,0) & (1,1) \\ 1 & 1 \end{bmatrix}$$

Here, $A_{00} = A_{01} = A_{10} = A_{11} = 1$;

The Z-transform of $x[m, n]$ is calculated by

$$X[Z_1, Z_2] = Z\{x[0, 0]\} + Z\{x[0, 1]\} + Z\{x[1, 0]\} + Z\{x[1, 1]\}$$

where,

$$\begin{aligned} Z\{x[0, 0]\} &= 1 \times Z_1^{-0} Z_2^{-0} = 1 \\ Z\{x[0, 1]\} &= A_{01} \times Z_1^{-0} Z_2^{-1} = 1 \times Z_2^{-1} = Z_2^{-1} \\ Z\{x[1, 0]\} &= A_{10} \times Z_1^{-1} Z_2^{-0} = 1 \times Z_1^{-1} = Z_1^{-1} \\ Z\{x[1, 1]\} &= A_{11} \times Z_1^{-1} Z_2^{-1} = 1 \times Z_1^{-1} Z_2^{-1} = Z_1^{-1} Z_2^{-1} \\ X(Z_1, Z_2) &= 1 + Z_1^{-0} Z_2^{-1} + Z_1^{-1} Z_2^{-0} + Z_1^{-1} Z_2^{-1} \\ X(Z_1, Z_2) &= 1 + Z_2^{-1} + Z_1^{-1} + Z_1^{-1} Z_2^{-1} \end{aligned}$$

The folded $h[m, n]$

$$h[-m, -n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

The Z-transform of $h[-m, -n]$ is given by

$$H(\overline{Z_1}, \overline{Z_2}) = 1 + Z_1^0 Z_2^1 + Z_1^1 Z_2^0 + Z_1^1 Z_2^1$$

$$H(\overline{Z_1}, \overline{Z_2}) = 1 + Z_2^1 + Z_1^1 + Z_1^1 Z_2^1$$

$$H(\overline{Z_1}, \overline{Z_2}) = 1 + Z_2 + Z_1 + Z_1 Z_2$$

Step 3 Multiply and add

$$\begin{aligned} Y[Z_1, Z_2] &= X[Z_1, Z_2] \cdot H[\overline{Z_1}, \overline{Z_2}] \\ &= (1 + Z_2^{-1} + Z_1^{-1} + Z_1^{-1} Z_2^{-1}) \cdot (1 + Z_2 + Z_1 + Z_1 Z_2) \\ &= 1 + Z_2 + Z_1 + Z_1 Z_2 + Z_2^{-1} + 1 + Z_1 Z_2^{-1} + Z_1 + Z_1^{-1} + Z_1^{-1} Z_2 + 1 + Z_2 \\ &\quad + Z_1^{-1} Z_2^{-1} + Z_1^{-1} + Z_2^{-1} + 1 \\ &= 4 + 2Z_2 + 2Z_1 + 1 \cdot Z_1 Z_2 + 2Z_2^{-1} + 1 \cdot Z_1 Z_2^{-1} + 2Z_1^{-1} + 1 \cdot Z_1^{-1} Z_2 + 1 \cdot Z_1^{-1} Z_2^{-1} \end{aligned}$$

Step 4 Collecting the coefficients of each element

The coefficient of $Z_1^{-1} Z_2^{-1} = A_{(-1,-1)} = 1$

The coefficient of $Z_1^{-1} Z_2^0 = A_{(-1,0)} = 2$

The coefficient of $Z_1^{-1} Z_2^1 = A_{(-1,1)} = 1$

The coefficient of $Z_1^0 Z_2^{-1} = A_{(0,-1)} = 2$

The coefficient of $Z_1^0 Z_2^0 = A_{(0,0)} = 4$

The coefficient of $Z_1^0 Z_2^1 = A_{(0,1)} = 2$

The coefficient of $Z_1^1 Z_2^{-1} = A_{(1,-1)} = 1$

The coefficient of $Z_1^1 Z_2^0 = A_{(1,0)} = 2$

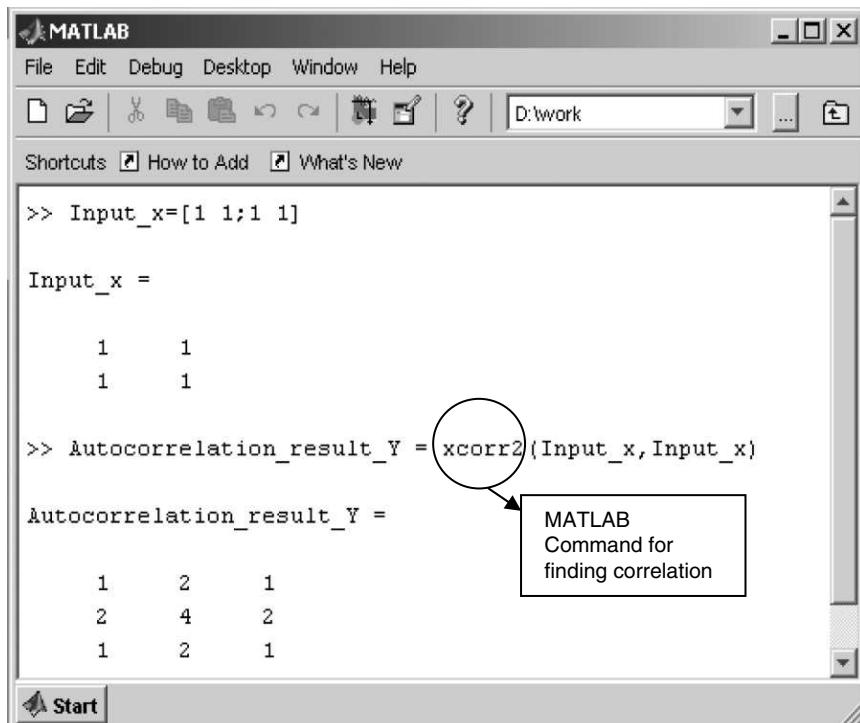
The coefficient of $Z_1^1 Z_2^1 = A_{(1,1)} = 1$

Step 5 Substituting the coefficient in the following equation

The result is given by

$$Y(m, n) = \begin{array}{c} \begin{matrix} n = -1 & n = 0 & n = 1 \\ \hline m = -1 & A_{(-1,-1)} & A_{(-1,0)} & A_{(-1,1)} \\ m = 0 & A_{(0,-1)} & A_{(0,0)} & A_{(0,1)} \\ m = 1 & A_{(1,-1)} & A_{(1,0)} & A_{(1,1)} \end{matrix} \end{array}$$

The above shown result is $Y(m, n) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ proved in the MATLAB simulation below.

a) Using **Autocorrelation** command in MATLAB


```

>> Input_x=[1 1;1 1]

Input_x =
1 1
1 1

>> Autocorrelation_result_Y = xcorr2(Input_x,Input_x)

Autocorrelation_result_Y =
1 2 1
2 4 2
1 2 1

```

Fig. 3.6 2D Autocorrelation between two signals

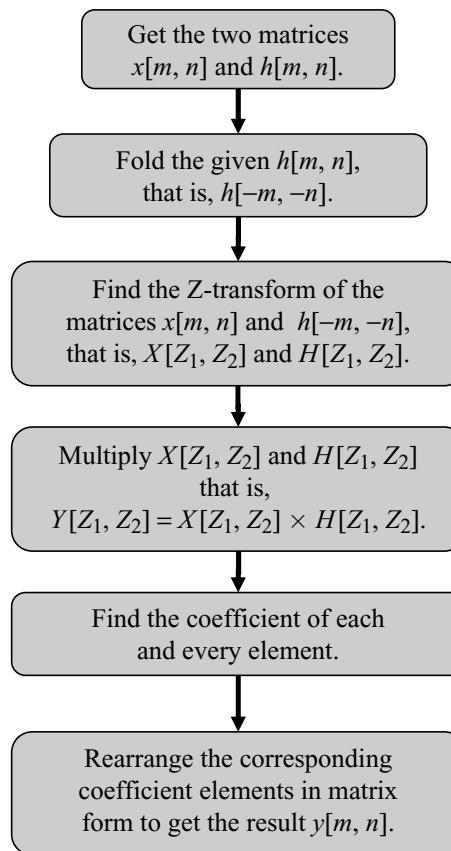
b) Computing Autocorrelation through **Convolution** command in MATLAB for the same in inputs given

Instead of using the `xcorr2` command in MATLAB, it is possible to perform autocorrelation by using `conv2` command. In order to perform correlation in terms of convolution, one of the signals should be folded in the x and y directions. This is done by using the commands `fliplr` and `flipud`. The impacts of `fliplr` and `flipud` commands are illustrated in Fig. 3.7. After the folding operation, the `conv2` command is used to get the autocorrelation result between two signals which is illustrated in Fig. 3.8. This example illustrates the relationship between convolution and correlation.

3.7.6 2D Cross-Correlation through Z Transform

This section presents the computation of 2D cross-correlation through 2D Z-transform. In order to perform cross-correlation through the Z-transform method, one of the signals is folded. The step-by-step approach to perform 2D cross-correlation is given in detail in this section.

Flow Chart The chart to perform linear correlation, that is $y[m, n]$, between two matrices $x[m, n]$ and $h[m, n]$ through Z-transform is as follows.



Example 3.18 Compute the linear cross-correlation between the two given matrices $x[m, n]$ and $h[m, n]$.

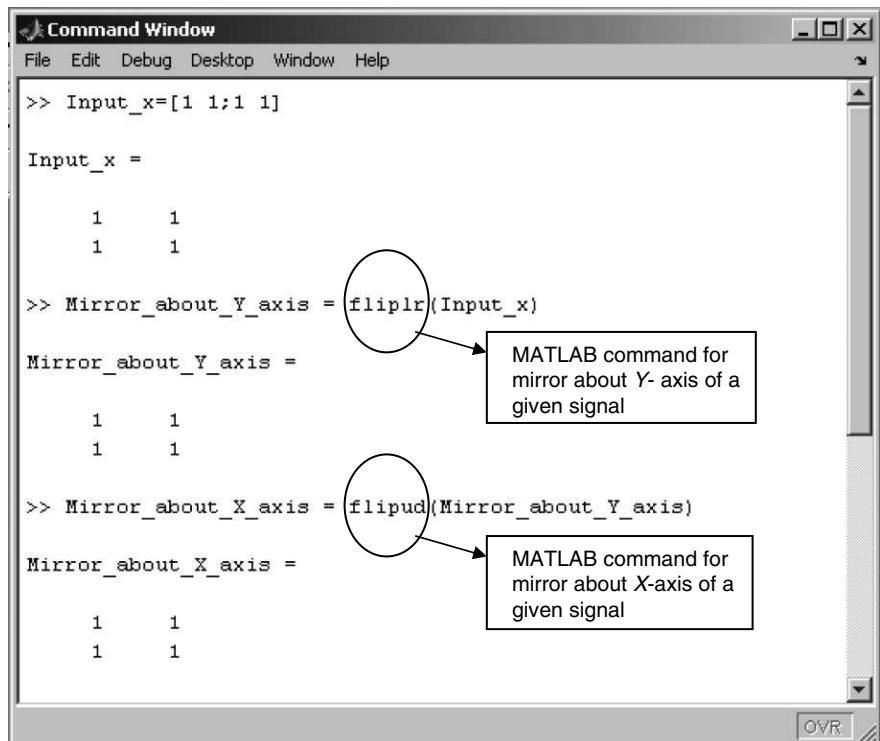
$$x[m, n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$h[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Solution Let the size of the matrix $x[m, n]$ be $M_1 \times N_1$, where M_1 is the number of rows and N_1 is the number of columns in the matrix ' $x[m, n]$ '. Let the size of the matrix $h[m, n]$ be $M_2 \times N_2$, where M_2 is the number of rows and N_2 is the number of columns in the matrix ' $h[m, n]$ '. The correlation which is a convolution between $x[m, n]$ and $h[-m, -n]$ is denoted by $y[m, n]$, where

$$y[m, n] = x[m, n] \otimes h[-m, -n]$$

The size of the matrix $y[m, n]$ is $M_3 \times N_3$, where $M_3 = M_1 + M_2 - 1$ and $N_3 = N_1 + N_2 - 1$



Command Window

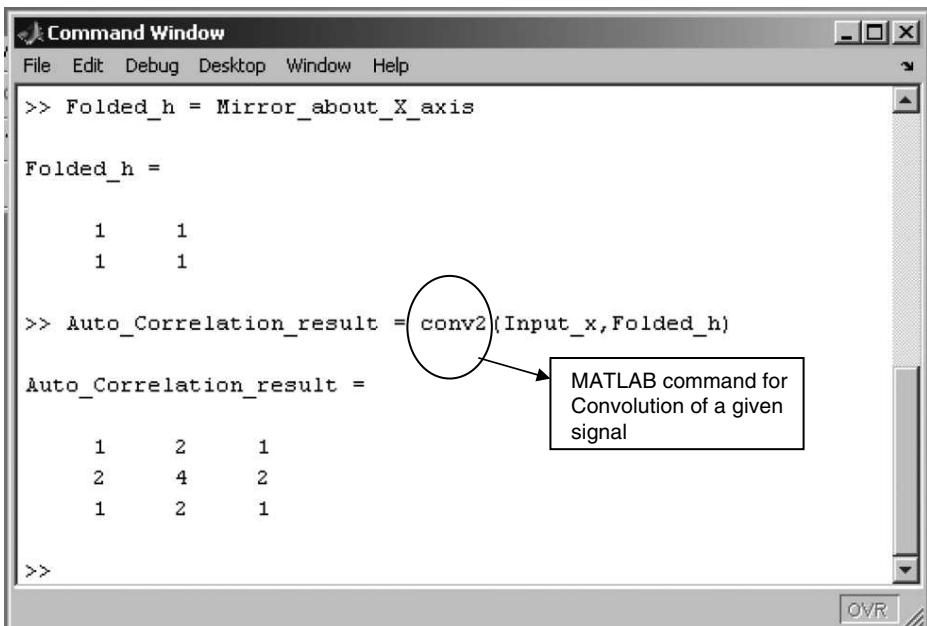
```

>> Input_x=[1 1;1 1]
Input_x =
1 1
1 1
>> Mirror_about_Y_axis = flipir(Input_x)
Mirror_about_Y_axis =
1 1
1 1
>> Mirror_about_X_axis = flipud(Mirror_about_Y_axis)
Mirror_about_X_axis =
1 1
1 1

```

MATLAB command for mirror about Y-axis of a given signal

MATLAB command for mirror about X-axis of a given signal

Fig. 3.7 *Folding operation*


Command Window

```

>> Folded_h = Mirror_about_X_axis
Folded_h =
1 1
1 1
>> Auto_Correlation_result = conv2(Input_x,Folded_h)
Auto_Correlation_result =
1 2 1
2 4 2
1 2 1
>>

```

MATLAB command for Convolution of a given signal

Fig. 3.8 *Autocorrelation in terms of convolution*

In this example, $M_1 = 2$, $N_1 = 2$ and $M_2 = 2$, $N_2 = 2$. Hence the resultant matrix that is $y[m, n]$ will have the size $M_3 = 2 + 2 - 1$ which is 3 and $N_3 = 2 + 2 - 1$ which is 3. The size of the matrix $y[m, n] = 3 \times 3$.

Step 1 Fold the signal $h[m, n]$ (convert the signal to the form of $h[-m, -n]$)

Before the signal $h[m, n]$ gets folded, the representation of the element's position in the matrix with the size of 2×2 looks like

$$h[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Here in the 2×2 matrix, 'm' takes a value '0', and '1' and 'n' takes a value of '0' and '1'.

After the signal $h[m, n]$ gets folded, the representation of the element's position in the matrix with the size of 2×2 , looks like

$$h[-m, -n] = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

Step 2 Take Z-Transform of the given matrices

Method to take Z-transform,

$$x[m, n] = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}$$

A_{00} is the element of the matrix $x[m, n]$ at the position $m = 0$ and $n = 0$; $x[0, 0] = A_{00}$.

A_{01} is the element of the matrix $x[m, n]$ at the position $m = 0$ and $n = 1$; $x[0, 1] = A_{01}$.

A_{10} is the element of the matrix $x[m, n]$ at the position $m = 1$ and $n = 0$; $x[1, 0] = A_{10}$.

A_{11} is the element of the matrix $x[m, n]$ at the position $m = 1$ and $n = 1$; $x[1, 1] = A_{11}$.

The simplified formulae for Z-transform for the matrix with the size of $m \times n$ is given below,

$$Z\{x[m, n]\} = X[Z_1, Z_2] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} A_{mn} Z_1^{-m} Z_2^{-n}$$

For a 2×2 matrix, after transforming the signal, the non-zero values will be only at $Z\{x[0, 0]\}$, $Z\{x[0, 1]\}$, $Z\{x[1, 0]\}$ and $Z\{x[1, 1]\}$. So the Z-transform of 2×2 matrix will be,

$$X[Z_1, Z_2] = Z\{x[0, 0]\} + Z\{x[0, 1]\} + Z\{x[1, 0]\} + Z\{x[1, 1]\}$$

Where,

$$Z\{x[0, 0]\} = A_{00} \times Z_1^{-0} Z_2^{-0} = A_{00}$$

$$Z\{x[0, 1]\} = A_{01} \times Z_1^{-0} Z_2^{-1} = A_{01} Z_2^{-1}$$

$$Z\{x[1, 0]\} = A_{10} \times Z_1^{-1} Z_2^{-0} = A_{10} Z_1^{-1}$$

$$Z\{x[1, 1]\} = A_{11} \times Z_1^{-1} Z_2^{-1} = A_{11} Z_1^{-1} Z_2^{-1}$$

In this example, it is given that

$$x[m, n] = \begin{bmatrix} (0, 0) & (0, 1) \\ 1 & 1 \\ (1, 0) & (1, 1) \\ 1 & 1 \end{bmatrix}$$

Here, $A_{00} = A_{01} = A_{10} = A_{11} = 1$;

The Z-transform of $x[m, n]$ is calculated by

$$X[Z_1, Z_2] = Z\{x[0, 0]\} + Z\{x[0, 1]\} + Z\{x[1, 0]\} + Z\{x[1, 1]\}$$

Where,

$$Z\{x[0, 0]\} = 1 \times Z_1^{-0} Z_2^{-0} = 1$$

$$Z\{x[0, 1]\} = A_{01} \times Z_1^{-0} Z_2^{-1} = 1 \times Z_2^{-1} = Z_2^{-1}$$

$$Z\{x[1, 0]\} = A_{10} \times Z_1^{-1} Z_2^{-0} = 1 \times Z_1^{-1} = Z_1^{-1}$$

$$Z\{x[1, 1]\} = A_{11} \times Z_1^{-1} Z_2^{-1} = 1 \times Z_1^{-1} Z_2^{-1} = Z_1^{-1} Z_2^{-1}$$

$$X(Z_1, Z_2) = 1 + Z_1^{-0} Z_2^{-1} + Z_1^{-1} Z_2^{-0} + Z_1^{-1} Z_2^{-1}$$

$$X(Z_1, Z_2) = 1 + Z_2^{-1} + Z_1^{-1} + Z_1^{-1} Z_2^{-1}$$

The folded $h[m, n]$

$$h[-m, -n] = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

The Z-transform of $h[-m, -n]$ is given by

$$H(\overline{Z_1}, \overline{Z_2}) = 4 + 3 \cdot Z_1^0 Z_2^1 + 2 \cdot Z_1^1 Z_2^0 + 1 \cdot Z_1^1 Z_2^1$$

$$H(\overline{Z_1}, \overline{Z_2}) = 4 + 3 \cdot Z_2^1 + 2 \cdot Z_1^1 + Z_1^1 Z_2^1$$

$$H(\overline{Z_1}, \overline{Z_2}) = 4 + 3 \cdot Z_2 + 2 \cdot Z_1 + Z_1 Z_2$$

Step 3 Multiply $X[Z_1, Z_2]$ and $H[\overline{Z_1}, \overline{Z_2}]$

$$Y[Z_1, Z_2] = X[Z_1, Z_2] \cdot H[\overline{Z_1}, \overline{Z_2}]$$

$$= (1 + Z_2^{-1} + Z_1^{-1} + Z_1^{-1} Z_2^{-1}) \cdot (4 + 3 \cdot Z_2 + 2 \cdot Z_1 + Z_1 Z_2)$$

$$\begin{aligned}
&= 4 + 3.Z_2 + 2.Z_1 + Z_1 Z_2 + 4.Z_2^{-1} + 3 + 2.Z_1 Z_2^{-1} + 1.Z_1 \\
&\quad + 4.Z_1^{-1} + 3.Z_1^{-1} Z_2 + 2 + 1.Z_2 + 4.Z_1^{-1} Z_2^{-1} + 3.Z_1^{-1} + 2.Z_2^{-1} + 1 \\
&= 10 + 4.Z_2 + 3.Z_1 + 1.Z_1 Z_2 + 6.Z_2^{-1} + 2.Z_1 Z_2^{-1} + 7.Z_1^{-1} + 3.Z_1^{-1} Z_2 + 4.Z_1^{-1} Z_2^{-1}
\end{aligned}$$

Step 4 Collecting the coefficients of each elements

The coefficient of $Z_1^{-1} Z_2^{-1} = A_{(-1, -1)} = 4$

The coefficient of $Z_1^{-1} Z_2^0 = A_{(-1, 0)} = 7$

The coefficient of $Z_1^{-1} Z_2^1 = A_{(-1, 1)} = 3$

The coefficient of $Z_1^0 Z_2^{-1} = A_{(0, -1)} = 6$

The coefficient of $Z_1^0 Z_2^0 = A_{(0, 0)} = 10$

The coefficient of $Z_1^0 Z_2^1 = A_{(0, 1)} = 4$

The coefficient of $Z_1^1 Z_2^{-1} = A_{(1, -1)} = 2$

The coefficient of $Z_1^1 Z_2^0 = A_{(1, 0)} = 3$

The coefficient of $Z_1^1 Z_2^1 = A_{(1, 1)} = 1$

Step 5 Substituting the coefficient in the following equation

$$Y(m, n) = \begin{bmatrix} n = -1 & n = 0 & n = 1 \\ m = -1 & A_{(-1, -1)} & A_{(-1, 0)} & A_{(-1, 1)} \\ m = 0 & A_{(0, -1)} & A_{(0, 0)} & A_{(0, 1)} \\ m = 1 & A_{(1, -1)} & A_{(1, 0)} & A_{(1, 1)} \end{bmatrix}$$

The result is given below

$$Y(m, n) = \begin{bmatrix} 4 & 7 & 3 \\ 6 & 10 & 4 \\ 2 & 3 & 1 \end{bmatrix}$$

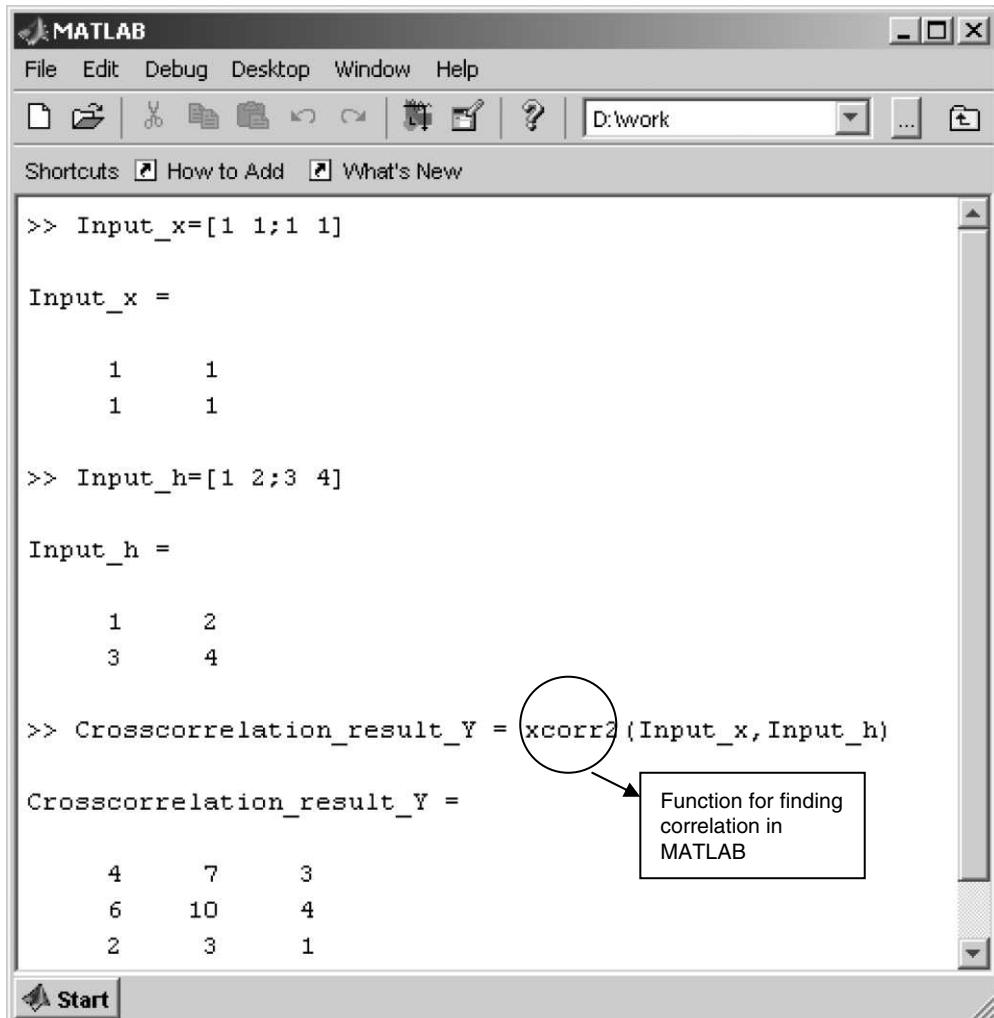
The above result can be verified through MATLAB simulation.

a) Using Direct Correlation command

The MATLAB command ‘xcorr2’ used to compute directly the cross correlation between two 2D signals is given in Fig. 3.9.

b) Compute the cross-correlation through convolution (conv2) command in MATLAB for the input signals given below.

$$x[m, n] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad h[m, n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$



```

>> Input_x=[1 1;1 1]
Input_x =
1 1
1 1

>> Input_h=[1 2;3 4]
Input_h =
1 2
3 4

>> Crosscorrelation_result_Y = xcorr2(Input_x, Input_h)
Crosscorrelation_result_Y =
4 7 3
6 10 4
2 3 1

```

Function for finding correlation in MATLAB

Fig. 3.9 *Cross correlation between two signals*

Instead of using ‘`xcorr2`’ to compute the cross-correlation between two signals, the convolution command ‘`conv2`’ can be used to perform the correlation. To do this, one of the signals has to be folded. The MATLAB command which is used to fold one of the input sequences is shown in Fig. 3.10.

Command Window

```

>> Input_x=[1 1;1 1]
Input_x =
1 1
1 1

>> Input_h=[1 2;3 4]
Input_h =
1 2
3 4

>> Mirror_about_Y_axis = fliplr(Input_h)
Mirror_about_Y_axis =
2 1
4 3

>> Mirror_about_X_axis = flipud(Mirror_about_Y_axis)
Mirror_about_X_axis =
4 3
2 1

>> Folded_h = Mirror_about_X_axis
Folded_h =
4 3
2 1

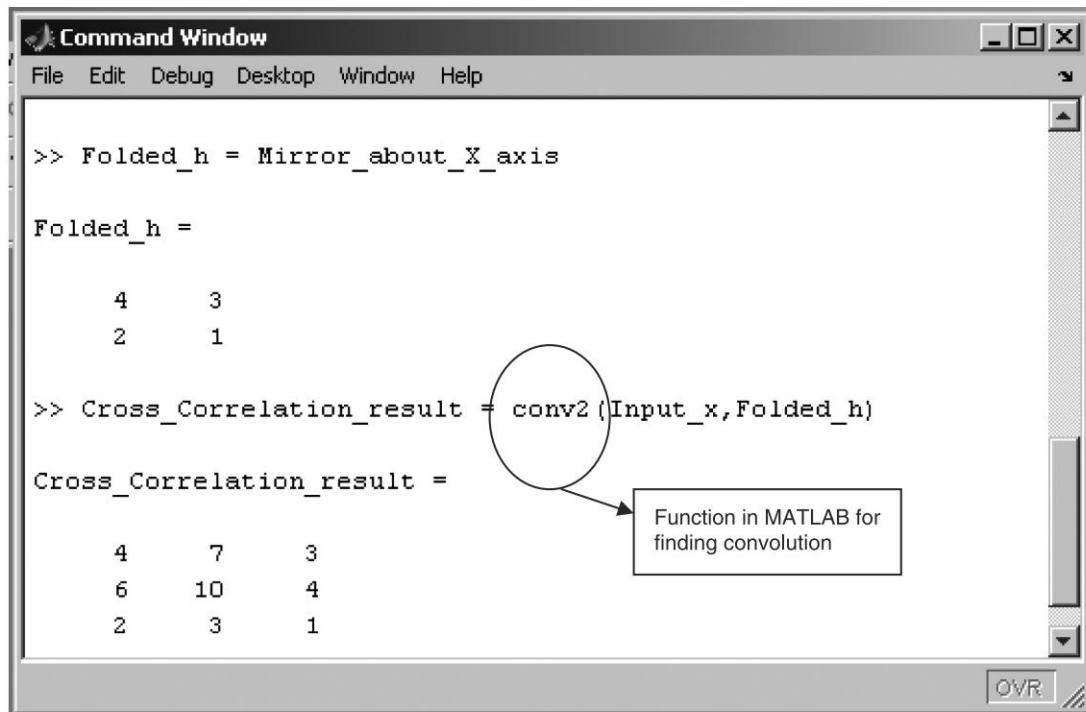
```

MATLAB command for finding the mirror along Y-axis of a given Signal

MATLAB command for finding the mirror along X-axis of a given Signal

Fig. 3.10 Folding operation

After folding, the convolution command (`conv2`) can be used to perform the cross-correlation between the two signals which is shown in Fig. 3.11.



```

>> Folded_h = Mirror_about_X_axis

Folded_h =

    4     3
    2     1

>> Cross_Correlation_result = conv2(Input_x,Folded_h)

Cross_Correlation_result =

    4     7     3
    6    10     4
    2     3     1

```

Function in MATLAB for finding convolution

Fig. 3.11 *Cross correlation in terms of convolution*



Summary

- Convolution basically computes the new value of a pixel as a weighted sum of pixel values in a certain neighbourhood surrounding the pixel.
- Convolution can be used to carry out the linear filtering process.
- Linear and circular convolution can be computed through (i) graphical method, (ii) Z-transform method, and (iii) matrix method.
- The operation of convolution is commutative.
- The computational load of convolution is reduced considerably if the utilised kernel is separable.
- Circular convolution can be used to perform interpolation of images.
- Correlation is commonly used to determine similarities between images or parts of images. It is possible to use correlation to determine the global displacement between two images of the same scene.
- Correlation can be broadly classified into (i) autocorrelation, and (ii) cross-correlation. If one finds the similarity of a signal to itself then it is autocorrelation. Cross-correlation is the similarity between two different signals.
- Autocorrelation is self-convolution without reflection of either function.
- The Fourier transform of the autocorrelation function is the power spectrum.
- Convolution in spatial domain is equal to multiplication in the frequency domain.

Review Questions

1. Prove that convolution with a 2D separable filter can be accomplished by performing two one-dimensional convolutions.

Let $f(m, n)$ represent the input image and $h(m, n)$ represent the 2D separable filter. First the rows of the image is convolved with $h_m^*(m)$ and then the columns of that result with $h_n(n)$ or vice versa. This concept is represented mathematically as

$$h * f = \sum_{x=-\frac{M}{2}}^{\frac{M}{2}} \sum_{y=-\frac{M}{2}}^{\frac{M}{2}} h(x, y) f(m-x, n-y)$$

If the filter $h(m, n)$ is separable then

$$h * f = \sum_{x=-\frac{M}{2}}^{\frac{M}{2}} h_m(x) \sum_{y=-\frac{N}{2}}^{\frac{N}{2}} h_n(y) f(m-x, n-y)$$

$$h * f = h_m(m) * [h_n(n) * f(m, n)]$$

From the above expressions, it is clear that 2D convolution can be performed as two 1D convolutions.

2. Calculate the number of multiplications required to convolve a 2D filter with a 2D image (a) Compute the 2D convolution at a stretch. (b) Perform the 2D convolution as two 1D convolutions. Assume the image is of size 100×100 pixels, and the filter is of size 10×10 .

- (a) The number of multiplications required to perform a 2D convolution, ignoring the border effect is given by

$$100 \times 100 \times 10 \times 10 = 10^6$$

- (b) The number of multiplications for two 1D convolution is

$$100 \times 100 \times 10 + 100 \times 100 \times 10 = 2 \times 10^5$$

From the results, it is obvious that performing a 2D convolution as two 1D convolutions reduces the computational complexity.

3. Give few applications of 2D convolution in the field of image processing.

Convolution is a powerful operation which can be used to perform filtering. A high-pass filter can be used to enhance the edges in an image. The high-pass filtering operation can be achieved by convolving the input image $f(m, n)$ with the spatial mask $h(m, n)$ which is given by

$$h(m, n) = \frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

A low-pass filter can be used to remove high-frequency noise in an image. The low-pass filtering can be achieved by convolving the input image $f(m, n)$ with the spatial mask $g(m, n)$ which is given by

$$g(m, n) = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The process of deconvolution can be used to remove motion blur in an image. The convolution operation is used in the construction of image pyramids like Gaussian pyramids and Laplacian pyramids which are more popular image representations.

Problems

- 3.1 Compute the 2D convolution between the two signals using the graphical method. The signals are given by

$$x(n_1, n_2) = \begin{bmatrix} 1 & 5 \\ 4 & 3 \end{bmatrix} \text{ and } h(n_1, n_2) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

- 3.2 Compute the 2D convolution between two signals illustrated in Fig. 3.12 by the graphical method. Assume the distance between two consecutive dots is one unit.

- 3.3 Perform the 2D convolution between two signals shown in Fig. 3.13 using the graphical method.

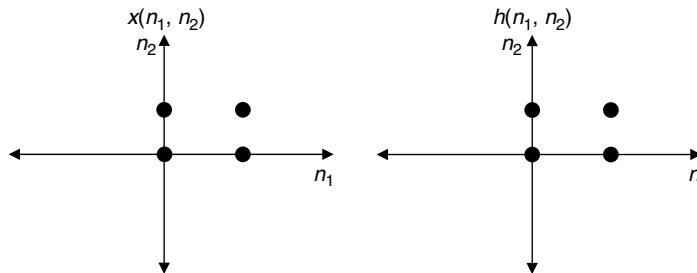


Fig. 3.12 Signals to be convolved

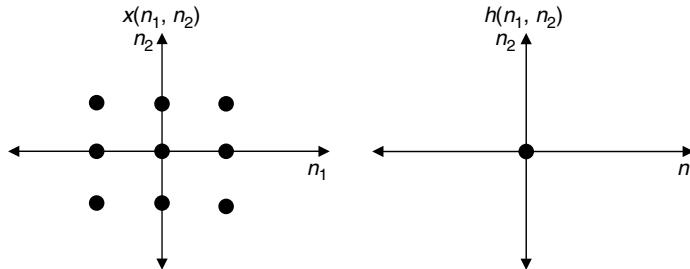


Fig. 3.13 2D signals

- 3.4 Perform the 2D linear convolution between the two signals $x(m, n) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ and $h(m, n) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ through the matrix method.

- 3.5 Compute the 2D linear convolution between the signal $x(m, n) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ with the two signals

$$h_1(m, n) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } h_2(m, n) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \text{ and comment on the observed result.}$$

- 3.6 Perform the 2D linear correlation between the signal $x(m, n) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ with the signals $h_1(m, n) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and $h_2(m, n) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and comment on the obtained result.

- 3.7 Compute the 2D linear convolution between the two signals $x(m, n) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ and $h(m, n) = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$

through any one of the techniques discussed in this chapter. The circle represents the value at $m = n = 0$.

- 3.8 Compare the computation complexity of 2D direct convolution with 2D separable convolution.

- 3.9 Prove that convolution in spatial domain is equal to multiplication in the frequency domain by taking

$$x(m, n) = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ and } h(m, n) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

- 3.10 Prove the following properties of two-dimensional convolution

(i) Commutative property:

$$x(m, n) \star h(m, n) = h(m, n) \star x(m, n)$$

(ii) Associative property:

$$(x(m, n) \star h(m, n)) \star g(m, n) = x(m, n) \star (h(m, n) \star g(m, n))$$

(iii) Distributive property:

$$x(m, n) \star (h(m, n) + g(m, n)) = (x(m, n) \star h(m, n)) + (x(m, n) \star g(m, n))$$

References

Books

1. J Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990
2. D E Dudgeon and R M Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983
3. Anil K Jain, *Fundamentals of Digital Image Processing*, Pearson Education, 2003
4. N K Bose, *Applied Multidimensional Systems Theory*, New York: Van Nostrand Reinhold, 1982
5. Tamal Bose, *Digital Signal and Image Processing*, John Wiley & Sons, 2004
6. A Popoulis, *Systems and Transforms with Applications in Optics*, McGraw-Hill, New York, 1968

Web Resources

1. Wally Block lecture notes on two-dimensional convolution: zoot.radiology.wisc.edu/~block/bme530lectures/L01Systemtheory.ppt

4

Learning Objectives

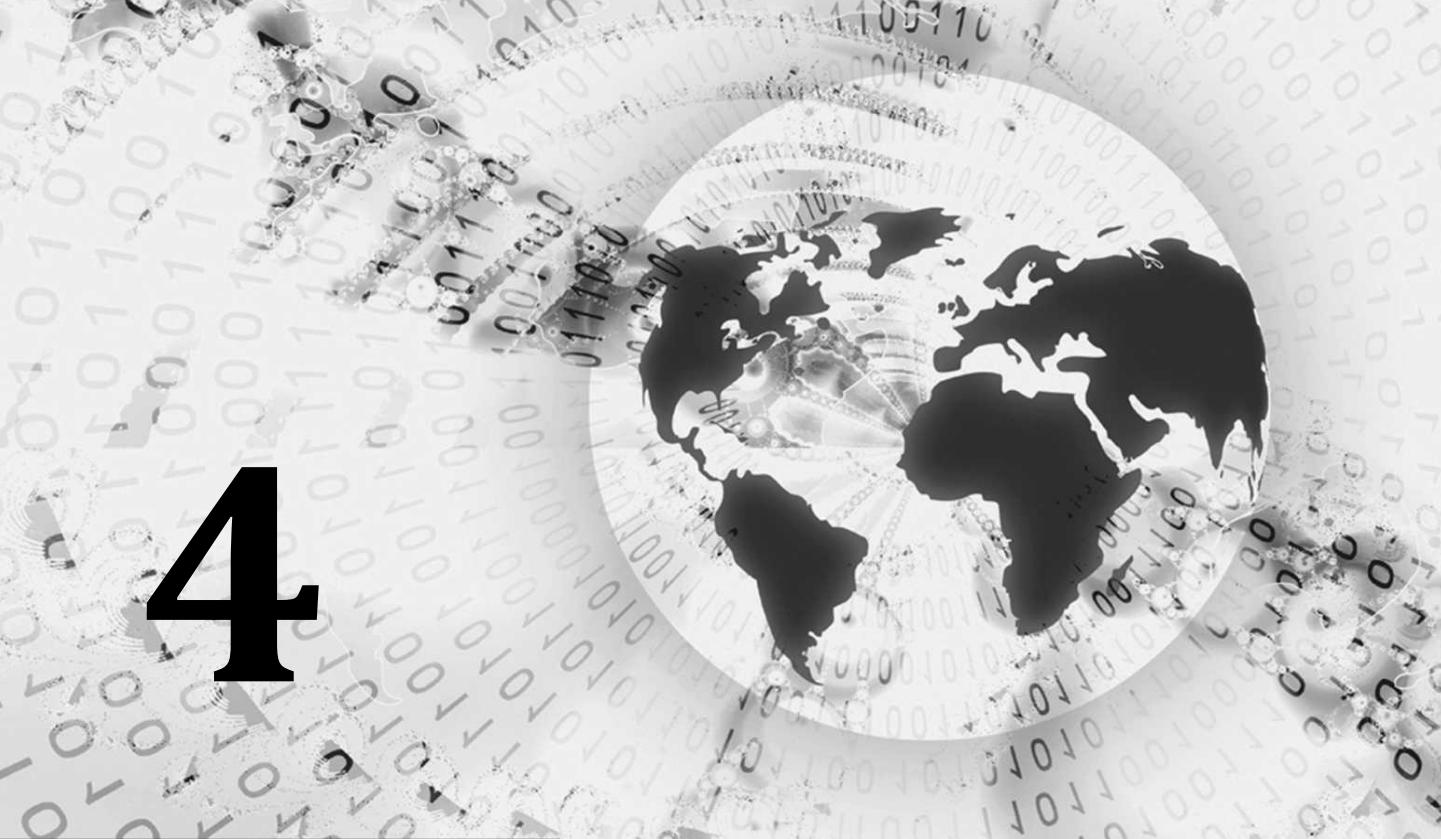
This chapter deals with different image transforms. Transform is basically a representation of the signal. Efficient representation of visual information lies at the foundation of many image-processing tasks which include image filtering, image compression and feature extraction. Efficiency of a representation refers to the ability to capture significant information of an image in a small description. After reading this chapter, the reader should be familiar with the following concepts:

Need for image transforms
different types of image transforms
Properties of image transforms
Applications of image transforms

Image Transforms

4.1 INTRODUCTION

Image transforms are extensively used in image processing and image analysis. Transform is basically a mathematical tool, which allows us to move from one domain to another domain (time domain to the frequency domain). 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. It is to be noted that all the transforms will not give frequency domain information. Transforms play a significant role in various image-processing applications such as image analysis, image enhancement, image filtering and image compression.

4.2 NEED FOR TRANSFORM

Transform is basically a mathematical tool to represent a signal. The need for transforms is given as follows:

(i) Mathematical Convenience Every action in time domain will have an impact in the frequency domain.



The complex convolution operation in time domain is equal to simple multiplication operation in the frequency domain.

(ii) To Extract more Information Transforms allow us to extract more relevant information. To illustrate this, consider the following example.

Person X is on the left-hand side of the prism, whereas the person Y is on the right-hand side of the prism as illustrated in Fig. 4.1. Person X sees the light as white light whereas the person Y sees the white light as a combination of seven colours (VIBGYOR).

Obviously, the person Y is getting more information than the person X by using the prism. Similarly, a transform is a tool that allows one to extract more information from a signal, as shown in Fig. 4.2.

Here, the person X is in the time domain and the person Y is in the frequency domain. The tool which allows us to move from time domain to frequency domain is the TRANSFORM. It is worthwhile to note that transforms do not change the information content present in a signal.

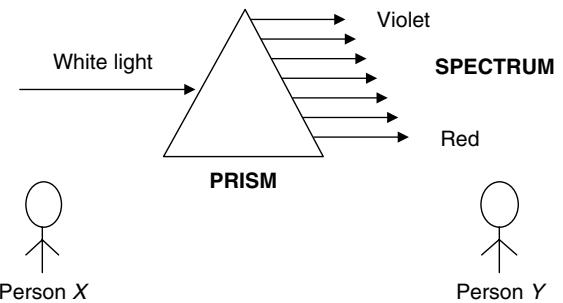


Fig. 4.1 Spectrum of white light

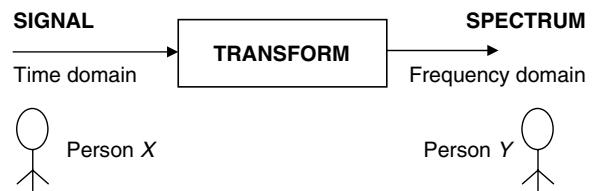


Fig. 4.2 Concept of transformation

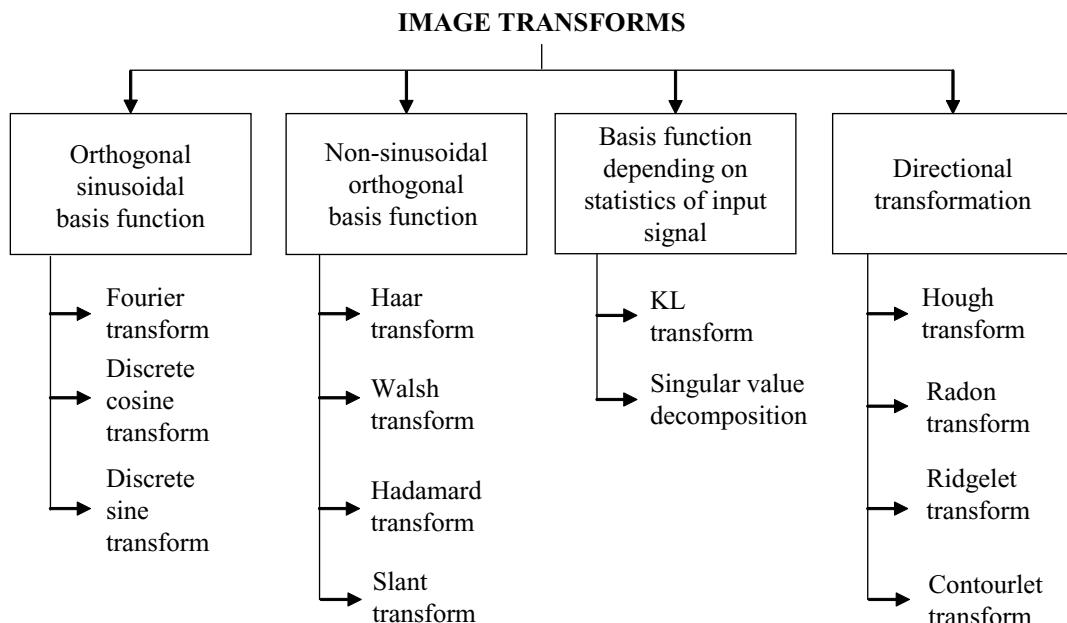
4.3 IMAGE TRANSFORMS

Image transform is basically a representation of an image. 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. The different types of image transforms discussed in this section are

1. Fourier transform
2. Walsh transform

3. Hadamard transform
4. Slant transform
5. Discrete cosine transform
6. KL transform
7. Radon transform
8. Wavelet transform

Classification of Image Transforms Image transforms can be classified based on the nature of the basis functions as (i) transforms with orthogonal basis functions, (ii) transforms with non-sinusoidal orthogonal basis functions, (iii) transforms whose basis functions depend on the statistics of the input data, and (iv) directional transformation, that is transforms whose basis functions are capable of representing the directional information present in the image.



One of the most powerful transforms with orthogonal sinusoidal basis function is the Fourier transform. The transforms whose basis functions are sinusoidal in nature include *Fourier transform*, *discrete cosine transform* and *discrete sine transform*. The transform which is widely used in the field of image compression is discrete cosine transform.

The transforms whose basis functions are non-sinusoidal in nature includes *Haar transform*, *Walsh transform*, *Hadamard transform* and slant transform. The Haar transform is the simplest example of a wavelet transform. One of the important advantages of wavelet transform is that signals (images) can be represented in different resolutions.

The transform whose basis function depends on the statistics of input signal are *KL transform* and *singular value decomposition*. The KL transform is considered to be the best among all linear transforms with respect to energy compaction.

The transforms whose basis functions are effective in representing the directional information of a signal include *Hough transform*, *Radon transform*, *Ridgelet transform* and *Contourlet transform*. Hough transform is discussed in Chapter 7. Contourlet transformation is given in Chapter 12.

4.4 FOURIER TRANSFORM

The Fourier transform was developed by Jean Baptiste Joseph Fourier to explain the distribution of temperature and heat conduction. Fourier transform is widely used in the field of image processing. An image is a spatially varying function. One way to analyse spatial variations is to decompose an image into a set of orthogonal functions, one such set being the Fourier functions. A Fourier transform is used to transform an intensity image into the domain of spatial frequency.

For a continuous time signal $x(t)$, the Fourier transform is defined as $X(\Omega)$

$$x(t) \xrightarrow{\text{CTFT}} X(\Omega)$$

Continuous Time Fourier Transform (CTFT) is defined as

$$X(\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt \quad (4.1)$$

A continuous time signal $x(t)$ is converted into discrete time signal $x(nT)$ by sampling process, where T is the sampling interval.

$$x(t) \xrightarrow{\text{sampling}} x(nT) \quad (4.2)$$

The Fourier transform of a finite energy discrete time signal $x(nT)$ is given by

$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x(nT) e^{-j\Omega nt} \\ X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x(n) e^{-j(\Omega t)n} \end{aligned} \quad (4.3)$$

where $X(e^{j\omega})$ is known as Discrete-Time Fourier Transform (DTFT) and is a continuous function of ω .

The relation between ω and Ω is given by

$$\omega = \Omega T \quad (4.4)$$

Replacing Ω by $2\pi f$ in Eq. (4.4), we get

$$\omega = 2\pi f \times T \quad (4.5)$$

where T is the sampling interval and is equal to $\frac{1}{f_s}$. Replacing T by $\frac{1}{f_s}$ in Eq. (4.5), we get

$$\omega = 2\pi f \times \frac{1}{f_s} \quad (4.6)$$

$$\frac{f}{f_s} = k \quad (4.7)$$

By substituting Eq. (4.7) in Eq. (4.6), we get

$$\omega = k \times 2\pi \quad (4.8)$$

To limit the infinite number of values to a finite number, Eq. (4.8) is modified as

$$\frac{\omega}{2\pi} = \frac{k}{N} \quad (4.9)$$

The Discrete Fourier Transform (DFT) of a finite duration sequence $x(n)$ is defined as

$$X(K) = \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi}{N} kn} \quad (4.10)$$

where $k = 0, 1, \dots, N-1$

The DFT is a discrete-frequency representation that projects a discrete signal onto a basis of complex sinusoids.

Unitary Transform A discrete linear transform is unitary if its transform matrix conforms to the unitary condition

$$A \times A^H = I \quad (4.11)$$

where A = transformation matrix, A^H represents Hermitian matrix.

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

I = identity matrix

When the transform matrix A is unitary, the defined transform is called unitary transform.

Example 4.1 Check whether the DFT matrix is unitary or not.

Solution

Step 1 Determination of the matrix A

Finding 4-point DFT (where $N = 4$)

The formula to compute a DFT matrix of order 4 is given below.

$$X(K) = \sum_{n=0}^3 x(n) e^{-j \frac{2\pi}{4} kn} \quad \text{where } k = 0, 1, \dots, 3$$

1. *Finding $X(0)$*

$$X(0) = \sum_{n=0}^3 x(n) = x(0) + x(1) + x(2) + x(3)$$

2. Finding $X(1)$

$$\begin{aligned}
 X(1) &= \sum_{n=0}^3 x(n) e^{-j\frac{\pi}{2}n} \\
 &= x(0) + x(1)e^{-j\frac{\pi}{2}} + x(2)e^{-j\pi} + x(3)e^{-j\frac{3\pi}{2}} \\
 X(1) &= x(0) - jx(1) - x(2) + jx(3)
 \end{aligned}$$

3. Finding $X(2)$

$$\begin{aligned}
 X(2) &= \sum_{n=0}^3 x(n) e^{-j\pi n} \\
 &= x(0) + x(1)e^{-j\pi} + x(2)e^{-j2\pi} + x(3)e^{-j3\pi} \\
 X(2) &= x(0) - x(1) + x(2) - x(3)
 \end{aligned}$$

4. Finding $X(3)$

$$\begin{aligned}
 X(3) &= \sum_{n=0}^3 x(n) e^{-j\frac{3\pi}{2}n} \\
 &= x(0) + x(1)e^{-j\frac{3\pi}{2}} + x(2)e^{-j3\pi} + x(3)e^{-j\frac{9\pi}{2}} \\
 X(3) &= x(0) + jx(1) - x(2) - jx(3)
 \end{aligned}$$

Collecting the coefficients of $X(0)$, $X(1)$, $X(2)$ and $X(3)$, we get

$$X[k] = A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Step 2 Computation of A^H

To determine A^H , first determine the conjugate and then take its transpose.

$$A \xrightarrow{\text{Conjugate}} A^* \xrightarrow{\text{Transpose}} A^H$$

Step 2a Computation of conjugate A^*

$$A^* = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

Step 2b Determination of transpose of A^*

$$(A^*)^T = A^H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

Step 3 Determination of $A \times A^H$

$$A \times A^H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} = 4 \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The result is the identity matrix, which shows that Fourier transform satisfies unitary condition.

Seqency

Seqency refers to the number of sign changes. The seqency for a DFT matrix of order 4 is given below.

Transform Coefficients

SEQUENCY	
$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$	Zero sign change
$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$	Two sign changes
$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & 1 & -j \\ 1 & -1 & -1 & 1 \\ 1 & j & -1 & -j \end{bmatrix}$	Three sign changes
$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & 1 & -1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$	One sign change

4.5 2D DISCRETE FOURIER TRANSFORM

The 2D-DFT of a rectangular image $f(m, n)$ of size $M \times N$ is represented as $F(k, l)$

$$f(m, n) \xrightarrow{\text{2D-DFT}} F(k, l)$$

where $F(k, l)$ is defined as

$$F(k, l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{M} mk} e^{-j \frac{2\pi}{N} nl} \quad (4.12)$$

$$F(k, l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{N} nl} e^{-j \frac{2\pi}{M} mk}$$

For a square image $f(m, n)$ of size $N \times N$, the 2D DFT is defined as

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{N} mk} e^{-j \frac{2\pi}{N} nl} \quad (4.13)$$

The inverse 2D Discrete Fourier Transform is given by

$$f(m, n) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k, l) e^{j \frac{2\pi}{N} mk} e^{j \frac{2\pi}{N} nl} \quad (4.14)$$

The Fourier transform $F(k, l)$ is given by

$$F(k, l) = R(k, l) + jI(k, l)$$

where $R(k, l)$ represents the real part of the spectrum and $I(k, l)$ represents the imaginary part.

The Fourier transform $F(k, l)$ can be expressed in polar coordinates as

$$F(k, l) = |F(k, l)| e^{j\varphi(k, l)} \quad (4.15)$$

where $|F(k, l)| = \left(R^2 \{F(k, l)\} + I^2 \{F(k, l)\} \right)^{\frac{1}{2}}$ is called the magnitude spectrum of the Fourier transform and $\varphi(k, l) = \tan^{-1} \frac{I \{F(k, l)\}}{R \{F(k, l)\}}$ is the phase angle or phase spectrum. Here, $R \{F(k, l)\}$, $I \{F(k, l)\}$ are the real and imaginary parts of $F(k, l)$ respectively.

The computationally efficient form of DFT is the Fast Fourier Transform (FFT). There are two possible representations of the FFT of an image which are (a) standard representation, and (b) optical representation. In the standard representation, high frequencies are grouped at the centre of the image while the low frequencies are located at the edges which are illustrated in Fig. 4.3. The null frequency is in the upper-left corner of the image. The frequency range is given by

$$[0, N] \times [0, M]$$

where M is the horizontal resolution of the image, and N is the vertical resolution of the image.

We know that discreteness in one domain results in periodicity in another domain. Hence, in the case of a digital image, the spectrum will be unique in the range $-\pi$ to π or between 0 to 2π .

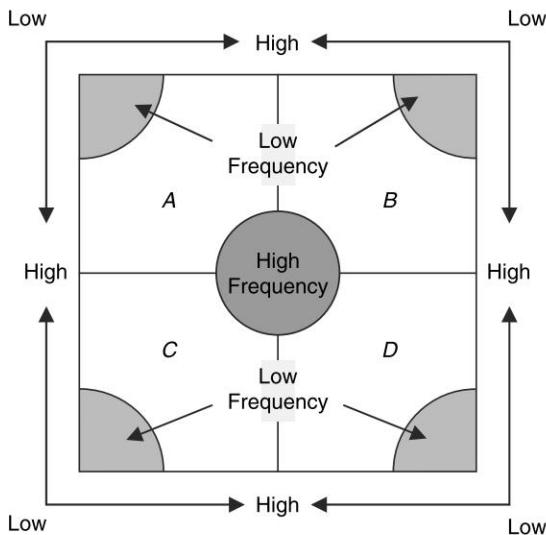
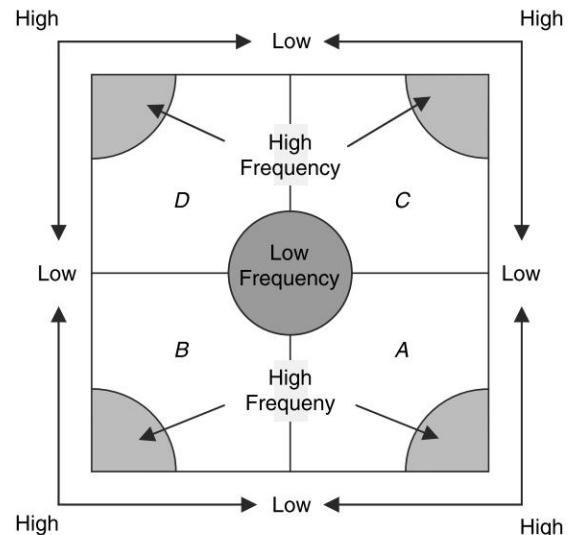
The optical representation of the FFT of the same image is illustrated in Fig. 4.4.

4.6 PROPERTIES OF 2D-DFT

The properties of 2D-DFT like (i) separable property, (ii) spatial shift property, (iii) periodicity property, (iv) convolution property, (v) correlation property, (vi) scaling property, (vii) conjugate symmetry property, and (viii) rotation property are given with proofs in this section.

4.6.1 Separable Property

The separable property allows a 2D transform to be computed in two steps by successive 1D operations on rows and columns of an image.

**Fig. 4.3** Standard representation of FFT of an image**Fig. 4.4** Optical representation of FFT of the image

Proof Mathematically, it is represented as

$$F(k, l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{N} mk} e^{-j \frac{2\pi}{N} nl}$$

$$F(k, l) = \sum_{m=0}^{N-1} \left(\sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{N} nl} \right) e^{-j \frac{2\pi}{N} mk}$$

$$F(k, l) = \sum_{m=0}^{N-1} f(m, l) e^{-j \frac{2\pi}{N} mk} = F(k, l)$$

Thus, performing a 2D Fourier transform is equivalent to performing two 1D transforms as

- Performing a 1D transform on each row of image $f(m, n)$ to get $F(m, l)$
- Performing a 1D transform on each column of $F(m, l)$ to get $F(k, l)$

The main advantage of separability is that a Fourier transform of any dimension can be performed by applying a 1D transform on each dimension.

This idea is graphically illustrated in Fig. 4.5.

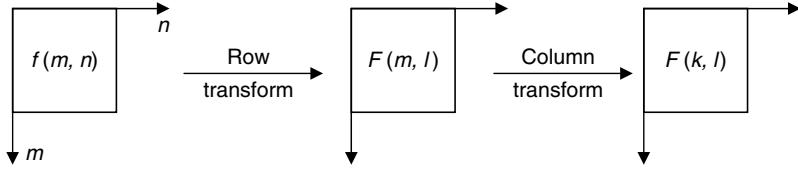


Fig. 4.5 Computation of 2D-DFT using separable property

4.6.2 Spatial Shift Property

The 2D DFT of a shifted version of the image $f(m, n)$, i.e., $f(m - m_0, n)$ is given by

$$f(m - m_0, n) \xrightarrow{\text{DFT}} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m - m_0, n) e^{-j \frac{2\pi}{N} m k} e^{-j \frac{2\pi}{N} n l} \quad (4.16)$$

where m_0 represents the number of times that the function $f(m, n)$ is shifted.

Proof Adding and subtracting m_0 to $e^{-j \frac{2\pi}{N} m k}$ in Eq. (4.16), we get

$$\text{DFT}[f(m - m_0, n)] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m - m_0, n) e^{-j \frac{2\pi}{N} (m - m_0 + m_0) k} e^{-j \frac{2\pi}{N} n l} \quad (4.17)$$

Splitting $e^{-j \frac{2\pi}{N} (m - m_0 + m_0) k}$ in to $e^{-j \frac{2\pi}{N} (m - m_0) k}$ and $e^{-j \frac{2\pi}{N} m_0 k}$ results in

$$= \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m - m_0, n) e^{-j \frac{2\pi}{N} (m - m_0) k} e^{-j \frac{2\pi}{N} m_0 k} e^{-j \frac{2\pi}{N} n l} \quad (4.18)$$

By taking the $e^{-j \frac{2\pi}{N} m_0 k}$ term outside in Eq. (4.18), we get

$$\text{DFT}[f(m - m_0, n)] = e^{-j \frac{2\pi}{N} m_0 k} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m - m_0, n) e^{-j \frac{2\pi}{N} (m - m_0) k} e^{-j \frac{2\pi}{N} n l} \quad (4.19)$$

From the definition of forward two-dimensional discrete fourier transform, we can write

$$\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m - m_0, n) e^{-j \frac{2\pi}{N} (m - m_0) k} e^{-j \frac{2\pi}{N} n l} = F(k, l).$$

Substituting this in Eq. (4.19) we get

$$\text{DFT}[f(m - m_0, n)] = e^{-j\frac{2\pi}{N}m_0 k} F(k, l) \quad (4.20)$$

This theorem proves that the DFT of a shifted function is unaltered except for a linearly varying phase factor.

4.6.3 Periodicity Property

The 2D DFT of a function $f(m, n)$ is said to be periodic with a period N if

$$F(k, l) \rightarrow F(k + pN, l + qN) \quad (4.21)$$

Proof $F(k + pN, l + qN) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j\frac{2\pi}{N}m(k+pN)} e^{-j\frac{2\pi}{N}n(l+qN)}$ (4.22)

Splitting the terms $e^{-j\frac{2\pi}{N}m(k+pN)}$ and $e^{-j\frac{2\pi}{N}n(l+qN)}$ in Eq. (4.22) results in

$$F(k + pN, l + qN) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}mpN} e^{-j\frac{2\pi}{N}nl} e^{-j\frac{2\pi}{N}nqN} \quad (4.23)$$

By taking $e^{-j2\pi mp}$ and $e^{-j2\pi nq}$ out of summation in Eq. (4.23), we get

$$F(k + pN, l + qN) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} [f(m, n) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}nl}] e^{-j2\pi mp} e^{-j2\pi nq} \quad (4.24)$$

By substituting Eq. (4.12) in Eq. (4.24), we get

$$F(k + pN, l + qN) = F(k, l) e^{-j2\pi mp} e^{-j2\pi nq} \quad (4.25)$$

The $e^{-j2\pi mp}$ and $e^{-j2\pi nq}$ values are always '1' for any integer values of n, q, p and m .

So the product of these two exponential terms $e^{-j2\pi mp} \cdot e^{-j2\pi nq}$ is also equal to '1'. By replacing $e^{-j2\pi mp} \cdot e^{-j2\pi nq}$ with '1' in Eq. (4.25), we get

$$F(k + pN, l + qN) = F(k, l) \cdot 1 \quad (4.26)$$

$$F(k + pN, l + qN) = F(k, l) \quad (4.27)$$

4.6.4 Convolution Property

Convolution is one of the most powerful operations in digital image processing. Convolution in spatial domain is equal to multiplication in the frequency domain.

Convolution of two sequences $x(n)$ and $h(n)$ is defined as

$$x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (4.28)$$

Two-dimensional convolution of two arrays (or) matrices $f(m, n)$ and $g(m, n)$ is given by

$$f(m, n) * g(m, n) = \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} f(a, b)g(m-a, n-b) \quad (4.29)$$

Proof DFT of the convolution of the two sequences $f(m, n)$ and $g(m, n)$ is given by

$$\text{DFT}\{f(m, n) * g(m, n)\} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \left\{ \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} f(a, b)g(m-a, n-b) \right\} e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}nl} \quad (4.30)$$

By taking all \sum outside Eq. (4.30), we get

$$\text{DFT}\{f(m, n) * g(m, n)\} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} f(a, b)g(m-a, n-b) e^{-j\frac{2\pi}{N}(m-a+a)k} e^{-j\frac{2\pi}{N}(n-b+b)l} \quad (4.31)$$

Splitting the terms $e^{-j\frac{2\pi}{N}(m-a+a)k}$ and $e^{-j\frac{2\pi}{N}(n-b+b)l}$ in Eq. (4.31) results in

$$\begin{aligned} \text{DFT}\{f(m, n) * g(m, n)\} &= \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} f(a, b)g(m-a, n-b) e^{-j\frac{2\pi}{N}(m-a)k} e^{-j\frac{2\pi}{N}ak} e^{-j\frac{2\pi}{N}(n-b)l} e^{-j\frac{2\pi}{N}bl} \\ \text{DFT}\{f(m, n) * g(m, n)\} &= \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} f(a, b) e^{-j\frac{2\pi}{N}ak} e^{-j\frac{2\pi}{N}bl} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g(m-a, n-b) e^{-j\frac{2\pi}{N}(m-a)k} e^{-j\frac{2\pi}{N}(n-b)l} \end{aligned} \quad (4.32)$$

Substituting Eq. (4.12) in Eq. (4.32) results in

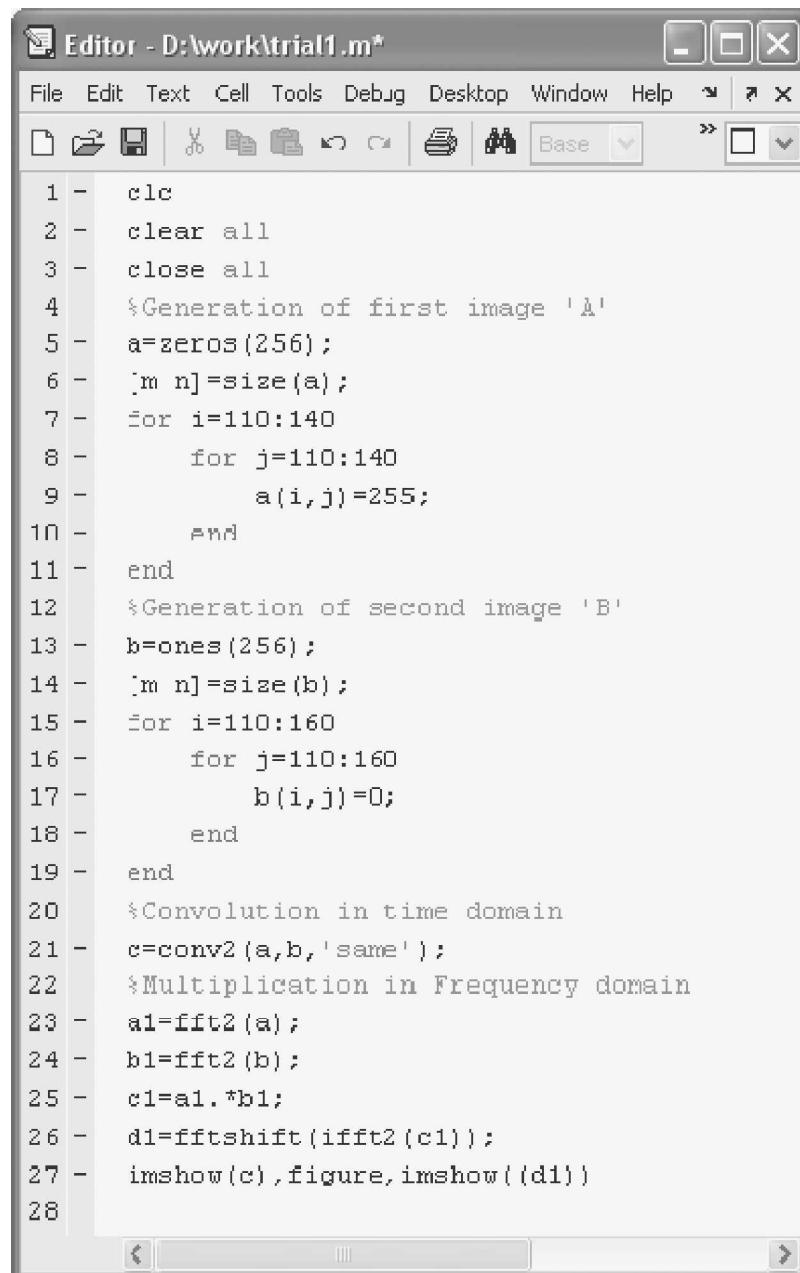
$$\text{DFT}\{f(m, n) * g(m, n)\} = F(k, l) \times G(k, l) \quad (4.33)$$

The convolution theorem tells us that the convolution of two functions in the spatial domain corresponds to multiplication in the frequency domain and vice versa.

Example 4.2 Convolution in spatial domain is equal to multiplication in the frequency domain. Prove this by a MATLAB code.

Solution The MATLAB code is given in Fig. 4.6.

From Fig. 4.6, it is clear that two images A and B are generated. The two images are convolved in the spatial domain. Then Fourier transform of the two images are taken to get the spectrum of A and B . The corresponding spectrum is then multiplied. It is found that convolution in spatial domain is equal to multiplication in the frequency domain.



```

1 - clc
2 - clear all
3 - close all
4 - %Generation of first image 'A'
5 - a=zeros(256);
6 - [m n]=size(a);
7 - for i=110:140
8 -     for j=110:140
9 -         a(i,j)=255;
10 -    end
11 - end
12 - %Generation of second image 'B'
13 - b=ones(256);
14 - [m n]=size(b);
15 - for i=110:160
16 -     for j=110:160
17 -         b(i,j)=0;
18 -     end
19 - end
20 - %Convolution in time domain
21 - c=conv2 (a,b, 'same');
22 - %Multiplication in Frequency domain
23 - a1=fft2 (a);
24 - b1=fft2 (b);
25 - c1=a1.*b1;
26 - d1=fftshift (ifft2 (c1));
27 - imshow(c), figure, imshow((d1))
28

```

Fig. 4.6 MATLAB code to prove convolution property

The result of the MATLAB code is given in Fig. 4.7. Figure 4.7 has (a) original image *A*, and (b) original image *B*.

From Fig. 4.7 (c) and (d), it is clear that convolution in spatial domain is equal to multiplication in the frequency domain.

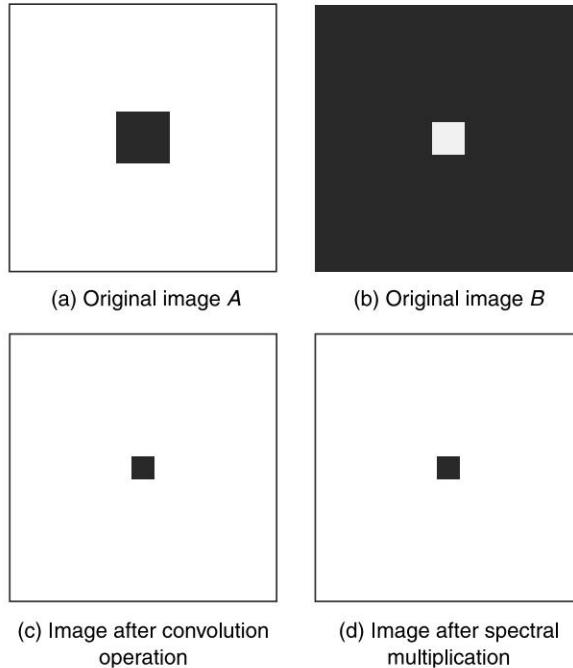


Fig. 4.7 Result of MATLAB code given in Fig. 4.6

4.6.5 Correlation Property

Correlation is basically used to find the relative similarity between two signals. The process of finding similarity of a signal to itself is autocorrelation, whereas the process of finding the similarity between two different signals is cross correlation.

The cross-correlation of two sequences $x(n)$ and $h(n)$ is equivalent to performing the convolution of one sequence with the folded version of the other sequence.

Proof The DFT of correlation of two sequences $x(n)$ and $h(n)$ is defined as

$$\text{DFT}\{R_{x, h}\} = \sum_{m=0}^{N-1} \left\{ \sum_{n=0}^{N-1} x(n)h(n+m) \right\} e^{-j\frac{2\pi}{N}mk} \quad (4.34)$$

Here, $R_{x, h}$ denotes the correlation between the signals $x(n)$ and $h(n)$.

By adding and subtracting n to the power of the exponential term $e^{-j\frac{2\pi}{N}mk}$ in Eq. (4.34), we get

$$\text{DFT}\{R_{x, h}\} = \sum_{m=0}^{N-1} \left\{ \sum_{n=0}^{N-1} x(n)h(n+m) \right\} e^{-j\frac{2\pi}{N}(m+n-n)k} \quad (4.35)$$

By splitting $e^{-j\frac{2\pi}{N}(m+n-n)k}$ into $e^{-j\frac{2\pi}{N}(n+m)k}$ and $e^{j\frac{2\pi}{N}nk}$ in Eq. (4.35), we get

$$\begin{aligned}
&= \sum_{m=0}^{N-1} \left\{ \sum_{n=0}^{N-1} x(n)h(n+m) \right\} e^{-j\frac{2\pi}{N}(n+m)k} e^{j\frac{2\pi}{N}nk} \\
&= \sum_{m=0}^{N-1} h(n+m) e^{-j\frac{2\pi}{N}(n+m)k} \sum_{n=0}^{N-1} x(n) e^{j\frac{2\pi}{N}nk}
\end{aligned} \tag{4.36}$$

From the definition of DFT, we can write

$$\text{DFT}\{R_{x, h}\} = H(k) \cdot \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}n(-k)} \tag{4.37}$$

which is reduced to

$$\text{DFT}\{R_{x, h}\} = H(k) \cdot X(-k) \tag{4.38}$$

The correlation property tells us that the correlation of two sequences in time domain is equal to the multiplication of DFT of one sequence and time reversal of the DFT of another sequence in the frequency domain.

4.6.6 Scaling Property

Scaling is basically used to increase or decrease the size of an image. According to this property, the expansion of a signal in one domain is equal to compression of the signal in another domain.

The 2D DFT of a function $f(m, n)$ is defined as

$$f(m, n) \xrightarrow{\text{DFT}} F(k, l)$$

If DFT of $f(m, n)$ is $F(k, l)$ then $\text{DFT}\{f(am, bn)\} = \frac{1}{|ab|} F(k/a, l/b)$

Proof The DFT of a function $f(am, bn)$ is given by

$$\text{DFT}\{f(am, bn)\} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(am, bn) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}nl} \tag{4.39}$$

By multiplying and dividing the power of the exponential term $e^{-j\frac{2\pi}{N}mk}$ with a and $e^{-j\frac{2\pi}{N}nl}$ with b in Eq. (4.39), we get

$$\text{DFT}\{f(am, bn)\} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(am, bn) e^{-j\frac{2\pi}{N}mk(a/b)} e^{-j\frac{2\pi}{N}nl(b/a)} \tag{4.40}$$

$$\text{DFT}\{f(am, bn)\} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(am, bn) e^{-j\frac{2\pi}{N}ma(k/b)} e^{-j\frac{2\pi}{N}nb(l/a)} \tag{4.41}$$

By substituting Eq. (4.12) in Eq. (4.41), we get

$$\text{DFT}\{f(am, bn)\} = \frac{1}{|ab|} F(k/a, l/b) \tag{4.42}$$

The scaling theorem tells us that compression in one domain produces a corresponding expansion in the Fourier domain (frequency domain) and vice versa.

Example 4.3 When $a = -1$, $b = -1$, the Eq. 4.42 reduces to

$$\begin{aligned} DFT\{f(-m, -n)\} &= \frac{1}{1} F(-k, -l) \\ DFT\{f(-m, -n)\} &= F(-k, -l) \end{aligned} \quad (4.43)$$

From Eq. (4.43), it is obvious that folding of the image in the spatial domain leads to folding of the spectrum.

4.6.7 Conjugate Symmetry

If the DFT of $f(m, n)$ is $F(k, l)$ then the DFT $[f^*(m, n)] = F^*(-k, -l)$

$$F(k, l) = F^*(-k, -l) \quad (4.44)$$

Proof The DFT of a function $f(m, n)$ is defined as

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{N} mk} e^{-j \frac{2\pi}{N} nl}$$

By applying complex conjugate to $F(k, l)$, we get

$$F^*(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{j \frac{2\pi}{N} mk} e^{j \frac{2\pi}{N} nl} \quad (4.45)$$

By applying reversal to $F^*(k, l)$ in Eq. (4.45), we get

$$F^*(-k, -l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{j \frac{2\pi}{N} m(-k)} e^{j \frac{2\pi}{N} n(-l)} \quad (4.46)$$

4.6.8 Orthogonality Property

The orthogonality property of a 2D DFT is given as

$$\frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{k, l}(m, n) a_{k', l'}^*(m, n) = \delta(k - k', l - l') \quad (4.47)$$

Where $\delta(k-k', l-l')$ is the Kronecker delta. This orthogonality condition can be used to derive the formula for the IDFT from the definition of the DFT.

4.6.9 Multiplication by Exponential

If the DFT of $f(m, n)$ is $F(k, l)$ then

$$\text{DFT}[e^{j\frac{2\pi}{N}mk_0} e^{j\frac{2\pi}{N}nl_0} f(m, n)] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}mk_0} e^{j\frac{2\pi}{N}nl_0} f(m, n) e^{-j\frac{2\pi}{N}mk} e^{-j\frac{2\pi}{N}nl} \quad (4.48)$$

Proof From the definition of a 2D DFT, we can write

$$\text{DFT}[e^{j\frac{2\pi}{N}mk_0} e^{j\frac{2\pi}{N}nl_0} f(m, n)] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{j\frac{2\pi}{N}m(k-k_0)} e^{j\frac{2\pi}{N}n(l-l_0)} \quad (4.49)$$

By combining $e^{j\frac{2\pi}{N}mk_0}$, $e^{-j\frac{2\pi}{N}mk}$ and $e^{-j\frac{2\pi}{N}nl}$, $e^{j\frac{2\pi}{N}nl_0}$ into a single exponential function in Eq. (4.49), we get

$$\text{DFT}[e^{j\frac{2\pi}{N}mk_0} e^{j\frac{2\pi}{N}nl_0} f(m, n)] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j\frac{2\pi}{N}m(k-k_0)} e^{-j\frac{2\pi}{N}n(l-l_0)} \quad (4.50)$$

By substituting Eq. (4.12) in Eq. (4.50), we get

$$\text{DFT}[e^{j\frac{2\pi}{N}mk_0} e^{j\frac{2\pi}{N}ml_0} f(m, n)] = F(k-k_0, l-l_0) \quad (4.51)$$

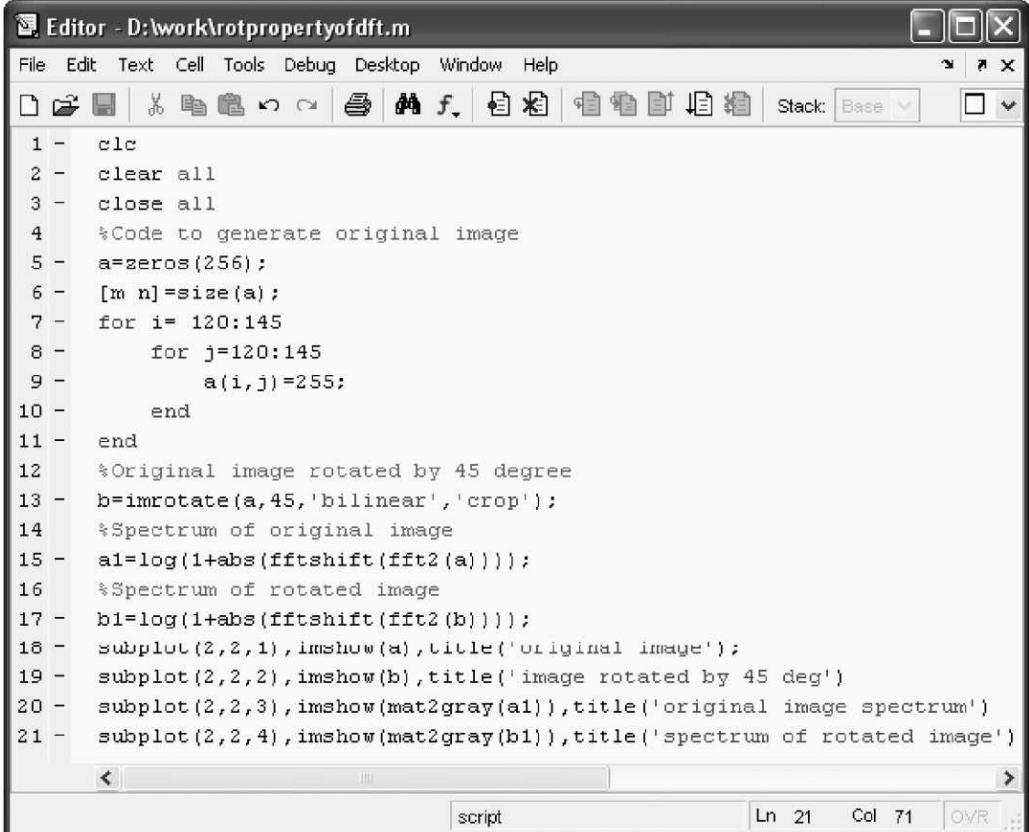
This theorem proves that multiplication of a function $f(m, n)$ with an exponential in the spatial domain leads to a frequency shift.

4.6.10 Rotation Property

The rotation property states that if a function is rotated by the angle, its Fourier transform also rotates by an equal amount.

$$\begin{aligned} f(m, n) &\rightarrow f(r \cos \theta, r \sin \theta) \\ \text{DFT}[f(r \cos \theta, r \sin \theta)] &\rightarrow F[R \cos \Phi, R \sin \Phi] \\ \text{DFT}[f(r \cos(\theta + \theta_0), r \sin(\theta + \theta_0))] &\rightarrow F[R \cos(\Phi + \Phi_0), R \sin(\Phi + \Phi_0)] \end{aligned} \quad (4.52)$$

The MATLAB code to prove the rotation property and the corresponding output are shown in Fig. 4.8 and Fig. 4.9 respectively.



```

1 - clc
2 - clear all
3 - close all
4 - %Code to generate original image
5 - a=zeros(256);
6 - [m n]=size(a);
7 - for i= 120:145
8 -     for j=120:145
9 -         a(i,j)=255;
10 -    end
11 - end
12 - %Original image rotated by 45 degree
13 - b=imrotate(a,45,'bilinear','crop');
14 - %Spectrum of original image
15 - a1=log(1+abs(fftshift(fft2(a))));
16 - %Spectrum of rotated image
17 - b1=log(1+abs(fftshift(fft2(b))));
18 - subplot(2,2,1),imshow(a),title('original image');
19 - subplot(2,2,2),imshow(b),title('image rotated by 45 deg')
20 - subplot(2,2,3),imshow(mat2gray(a1)),title('original image spectrum')
21 - subplot(2,2,4),imshow(mat2gray(b1)),title('spectrum of rotated image')

```

Fig. 4.8 MATLAB code to demonstrate rotation property of DFT

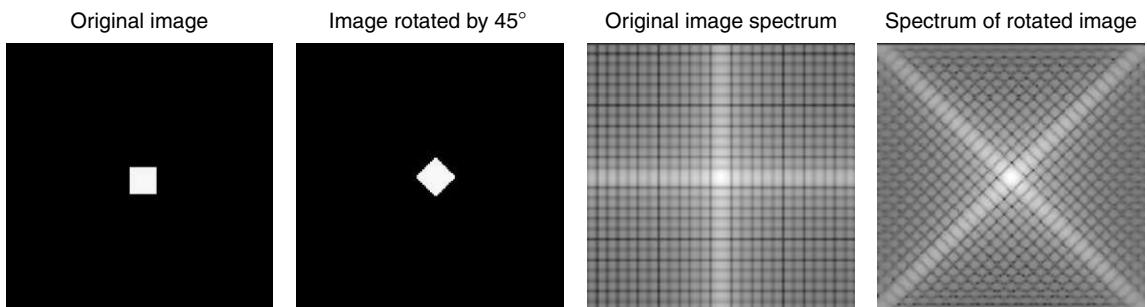


Fig. 4.9 Result of MATLAB code shown in Fig. 4.8.

From Fig. 4.8, it is clear that an image A is created; it is then rotated by an angle of 45° . The spectrum of the original and the rotated image are shown in Fig. 4.9 to prove that if the image is rotated by a specific angle, its spectrum will be rotated.

The properties of 2D Fourier transforms are given in Table 4.1.

Table 4.1 Properties of 2D DFT

Property	Sequence	Transform
Spatial shift	$f(m-m_0, n)$	$e^{-j\frac{2\pi}{N}m_0k} F(k, l)$
Periodicity	—	$F(k + pN, l + qN) = F(k, l)$
Convolution	$f(m, n)*g(m, n)$	$F(k, l) \times G(k, l)$
Scaling	$f(am, bn)$	$\frac{1}{ ab } F(k/a, l/b)$
Conjugate symmetry		$F(k, l) = F^*(-k, -l)$
Multiplication by exponential	$e^{j\frac{2\pi}{N}mk_0} e^{j\frac{2\pi}{N}ml_0} f(m, n)$	$F(k-k_0, l-l_0)$
Rotation property	$f(r \cos(\theta + \theta_0), r \sin(\theta + \theta_0))$	$F[R \cos(\Phi + \Phi_0), R \sin(\Phi + \Phi_0)]$
Folding property	$f(-m, -n)$	$F(-k, -l)$

Example 4.4 Compute the 2D DFT of the 4×4 grayscale image given below.

$$f[m, n] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Solution The 2D DFT of the image $f[m, n]$ is represented as $F[k, l]$.

$$F[k, l] = \text{kernel} \times f[m, n] \times (\text{kernel})^T \quad (4.53)$$

The kernel or basis of the Fourier transform for $N = 4$ is given by

$$\text{The DFT basis for } N = 4 \text{ is given by } \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \quad (4.54)$$

Substituting Eq. (4.54) in (4.53), we get

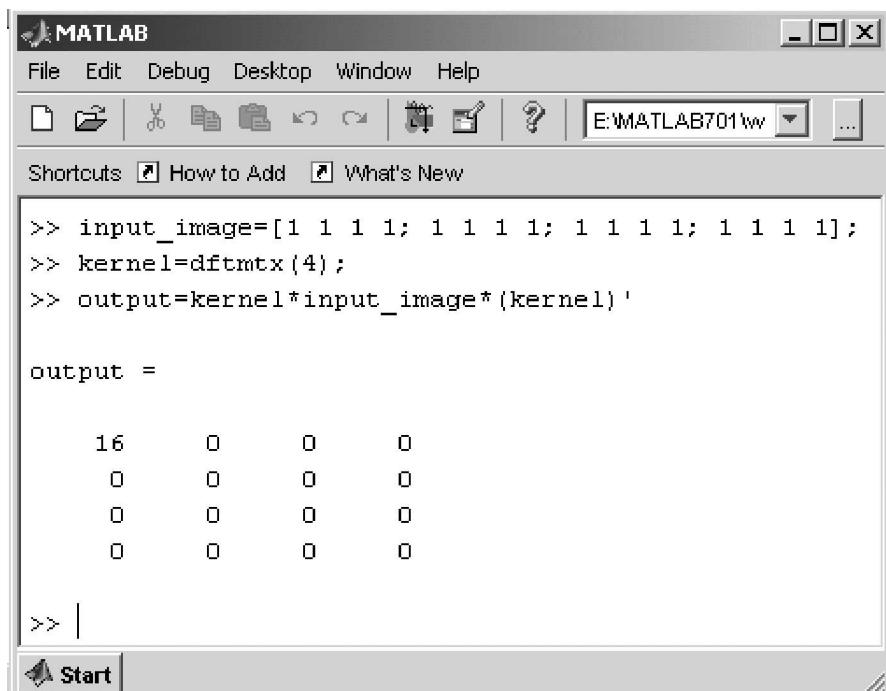
$$F(k, l) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \quad (4.55)$$

Upon simplification, we get

$$F(k, l) = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Note The Fourier transform of the input image results in only one value and all the other values are zero. The MATLAB code which can be used to compute the 2D DFT result is shown in Fig. 4.10.

Aliter The same result can be obtained using a built-in MATLAB command ‘**fft2**’ as shown in Fig. 4.11.



```

>> input_image=[1 1 1 1; 1 1 1 1; 1 1 1 1; 1 1 1 1];
>> kernel=dftmtx(4);
>> output=kernel*input_image*(kernel)'

output =

```

16	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Fig. 4.10 MATLAB code to compute 2D DFT

Example 4.5 Compute the inverse 2D DFT of the transform coefficients given by

$$F[k, l] = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

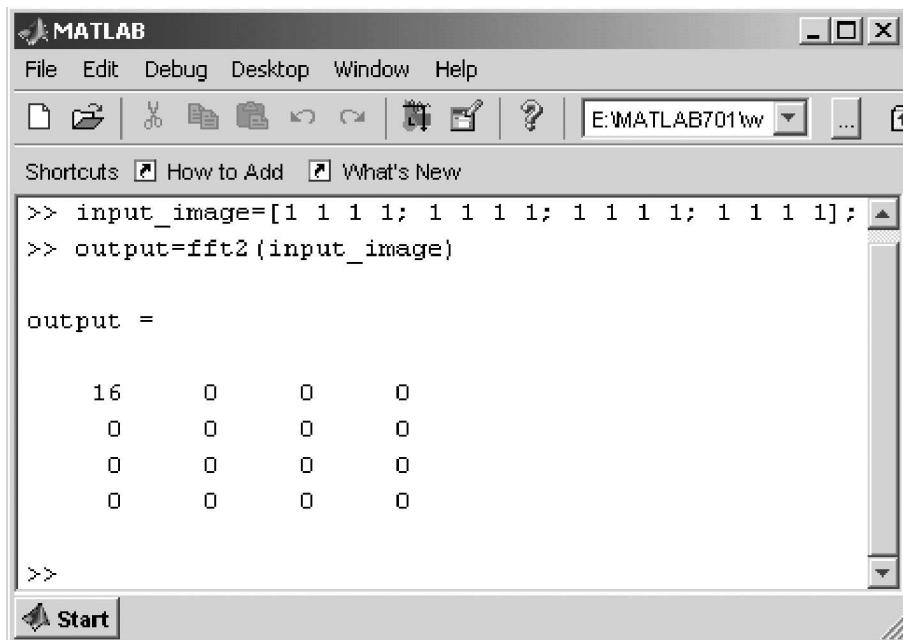


Fig. 4.11 Computation of 2D DFT of input matrix

Solution The inverse 2D DFT of the Fourier coefficients $F[k, l]$ is given by $f[m, n]$ as

$$f[m, n] = \frac{1}{N^2} \times \text{kernel} \times F[k, l] \times (\text{kernel})^T \quad (4.56)$$

In this example, $N = 4$

$$f[m, n] = \frac{1}{16} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

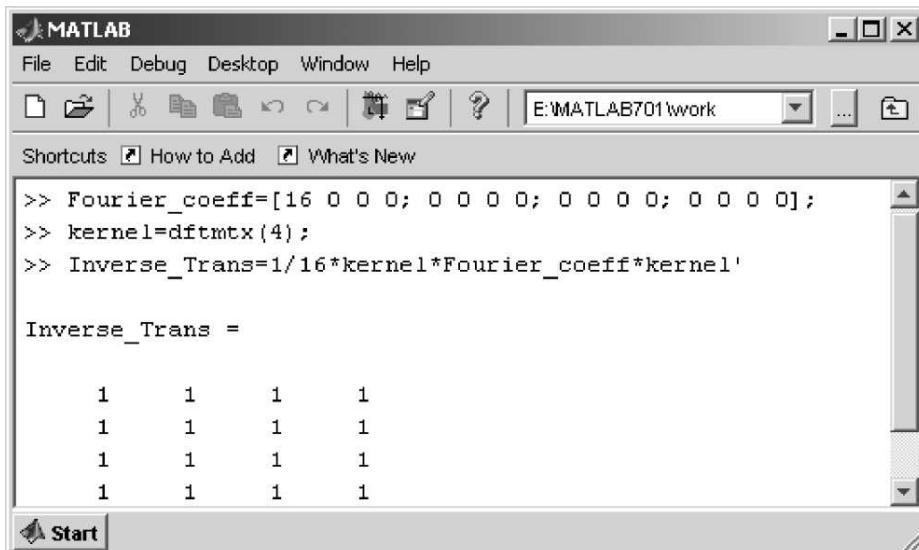
$$f[m, n] = \frac{1}{16} \times \begin{bmatrix} 16 & 0 & 0 & 0 \\ 16 & 0 & 0 & 0 \\ 16 & 0 & 0 & 0 \\ 16 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$f[m, n] = \frac{1}{16} \times \begin{bmatrix} 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 \end{bmatrix}$$

$$f[m, n] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

The same problem can be solved using MATLAB as shown in Fig. 4.12. In the figure, for getting the kernel, a built-in MATLAB command '**dftmtx()**' is used where '**dftmtx**' stands for Discrete Fourier Transform Matrix.

Aliter The same problem can be solved easily by using the built-in MATLAB command '**ifft2**' as shown in Fig. 4.13.



The screenshot shows a MATLAB graphical interface. The menu bar includes File, Edit, Debug, Desktop, Window, and Help. The toolbar contains icons for file operations like Open, Save, and Print. The current workspace is E:\MATLAB701\work. The command window displays the following MATLAB code:

```

>> Fourier_coeff=[16 0 0 0; 0 0 0 0; 0 0 0 0; 0 0 0 0];
>> kernel=dftmtx(4);
>> Inverse_Trans=1/16*kernel*Fourier_coeff*kernel';

Inverse_Trans =

```

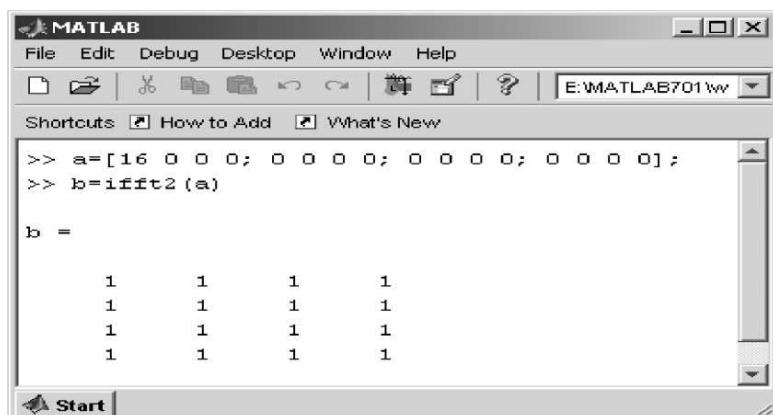
Below the code, the command window shows the resulting 4x4 matrix:

```

1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

```

Fig. 4.12 Inverse Fourier transform



The screenshot shows a MATLAB graphical interface. The menu bar includes File, Edit, Debug, Desktop, Window, and Help. The toolbar contains icons for file operations like Open, Save, and Print. The current workspace is E:\MATLAB701\work. The command window displays the following MATLAB code:

```

>> a=[16 0 0 0; 0 0 0 0; 0 0 0 0; 0 0 0 0];
>> b=ifft2(a);

b =

```

Below the code, the command window shows the resulting 4x4 matrix:

```

1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

```

Fig. 4.13 Inverse Fourier transform using **fft2** command

4.7 IMPORTANCE OF PHASE

The Fourier transform $F(k, l)$ can be expressed in polar coordinates as

$$F(k, l) = |F(k, l)|e^{j\phi(k, l)}$$

where $|F(k, l)| = \left(R^2 \{F(k, l)\} + I^2 \{F(k, l)\} \right)^{\frac{1}{2}}$ is called the magnitude spectrum of the Fourier transform and $\phi(k, l) = \tan^{-1} \frac{I\{F(k, l)\}}{R\{F(k, l)\}}$ is the phase angle or phase spectrum. Here, $R\{F(k, l)\}$, $I\{F(k, l)\}$ are the real and imaginary parts of $F(k, l)$ respectively. The Fourier transform of the image gives two important informations, one is the magnitude of the transform coefficient and the other one is the phase information. In order to understand the importance of phase, a simple exercise is given below

Example: 4.6 Phase interchange You are given two images: Image 1 and Image 2. First, take the Fourier transform of the two images. Determine the magnitude and phase component. During reconstruction using inverse Fourier transform, interchange the phase of the two images and reconstruct the image and observe the difference. This concept is represented in Fig. 4.14.

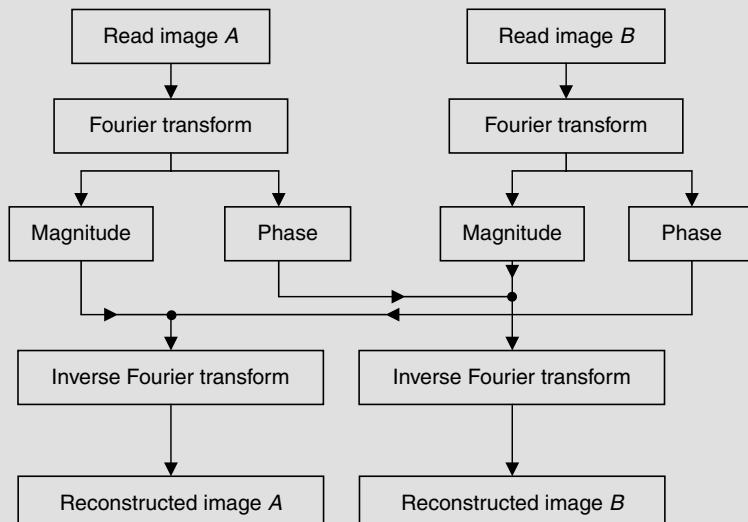
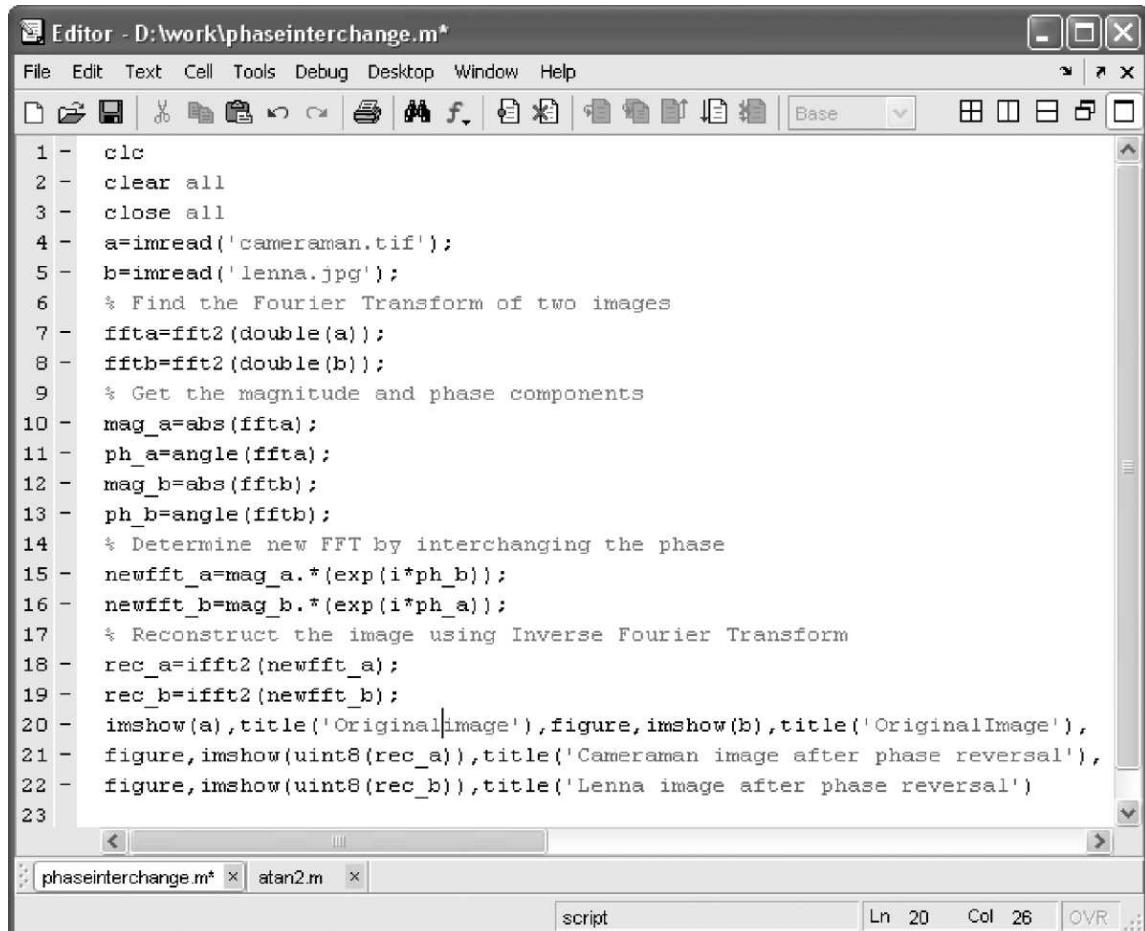


Fig. 4.14 Concept of phase interchange

Solution The MATLAB code which interchanges the phase between the two images is given in Fig. 4.15 and the corresponding output is illustrated in Fig. 4.16.

From Fig. 4.16 (c) and (d) we can see the cameraman and Lena image after phase interchange. From the figure, the effect of phase interchange in the two images is obvious.



The screenshot shows a MATLAB Editor window with the title 'Editor - D:\work\phaseinterchange.m*'. The window contains the following MATLAB code:

```

1 - clc
2 - clear all
3 - close all
4 - a=imread('cameraman.tif');
5 - b=imread('lenna.jpg');
6 - % Find the Fourier Transform of two images
7 - ffta=fft2(double(a));
8 - fftb=fft2(double(b));
9 - % Get the magnitude and phase components
10 - mag_a=abs(ffta);
11 - ph_a=angle(ffta);
12 - mag_b=abs(fftb);
13 - ph_b=angle(fftb);
14 - % Determine new FFT by interchanging the phase
15 - newfft_a=mag_a.*exp(i*ph_b);
16 - newfft_b=mag_b.*exp(i*ph_a);
17 - % Reconstruct the image using Inverse Fourier Transform
18 - rec_a=ifft2(newfft_a);
19 - rec_b=ifft2(newfft_b);
20 - imshow(a),title('OriginalImage'),figure,imshow(b),title('OriginalImage'),
21 - figure,imshow(uint8(rec_a)),title('Cameraman image after phase reversal'),
22 - figure,imshow(uint8(rec_b)),title('Lenna image after phase reversal')
23

```

The code reads two images ('cameraman.tif' and 'lenna.jpg'), performs a Fourier Transform on each, extracts magnitude and phase, then swaps the phase components. It then performs an inverse Fourier Transform to reconstruct the images and displays the original and reconstructed versions.

Fig. 4.15 MATLAB code to interchange phase between two images

4.8 WALSH TRANSFORM

Fourier analysis is basically the representation of a signal by a set of orthogonal sinusoidal waveforms. The coefficients of this representation are called *frequency components* and the waveforms are ordered by frequency. Walsh in 1923 introduced a complete set of orthonormal square-wave functions to represent these functions. The computational simplicity of the Walsh function is due to the fact that Walsh functions are real and they take only two values which are either +1 or -1.

The one-dimensional Walsh transform basis can be given by the following equation:

$$g(n, k) = \frac{1}{N} \prod_{i=0}^{m-1} (-1)^{b_i(n)b_{m-1-i}(k)} \quad (4.57)$$

Here n represents the time index, k represents the frequency index and N represents the order. Also, m represents the number bits to represent a number and $b_i(n)$ represents the i^{th} (from LSB) bit of the binary value, of n decimal number represented in binary. The value of m is given by $m = \log_2 N$.

**Fig. 4.16** Result of MATLAB code given in Fig. 4.14

The two-dimensional Walsh transform of a function $f(m, n)$ is given by

$$F(k, l) = \frac{1}{N} \sum_m \sum_n f(m, n) \prod_{i=0}^{p-1} (-1)^{[b_i(m)b_{p-1-i}(k) + b_i(n)b_{p-1-i}(l)]}$$

Example 4.7 Find the 1D Walsh basis for the fourth-order system ($N = 4$).

Solution Here, the value of N is given as four. From the value of N , the value of m is calculated as

$$N = 4;$$

$$\begin{aligned} m &= \log_2 N \\ &= \log_2 4 = \log_2 2^2 \\ &= 2 \cdot \log_2 2 \\ m &= 2 \end{aligned}$$

Also, n and k vary from 0 to $N-1$. In our example, $N = 4$. So n and k have the values of 0, 1, 2 and 3. i varies from 0 to $m-1$. From the above computation, $m = 2$. So i has the value of 0 and 1. The construction of Walsh basis for $N = 4$ is given in Table 4.2.

When k or n is equal to zero, the basis value will be $1/N$.

That is, $g(n, k) = \frac{1}{N}$; for $n = 0$, or $k = 0$

Table 4.2 Construction of walsh basis for $N = 4$

Decimal value	Binary values	
n	$b_1(n)$	$b_0(n)$
0	$b_1(0) = 0$	$b_0(0) = 0$
1	$b_1(1) = 0$	$b_0(1) = 1$
2	$b_1(2) = 1$	$b_0(2) = 0$
3	$b_1(3) = 1$	$b_0(3) = 1$

Seqency The Walsh functions may be ordered by the number of zero crossings or seqency, and the coefficients of the representation may be called seqency components. The seqency of the Walsh basis function for $N = 4$ is shown in Table 4.3.

Table 4.3 Walsh transform basis for $N = 4$

$k \backslash n$	0	1	2	3	Seqency
0	1/4	1/4	1/4	1/4	Zero sign change (DC value)
1	1/4	1/4	-1/4	-1/4	One sign change
2	1/4	-1/4	1/4	-1/4	Three sign changes
3	1/4	-1/4	-1/4	1/4	Two sign changes

Calculating the value for $g(1, 2)$

$$g(2, 1) = \frac{1}{4} \prod_{i=0}^1 (-1)^{b_i(2)b_{m-1-i}(1)}$$

$$g(2, 1) = \frac{1}{4} \left\{ \left((-1)^{b_0(2)b_1(1)} \right) \times \left((-1)^{b_1(2)b_0(1)} \right) \right\}$$

$$g(2,1) = \frac{1}{4} \left\{ \left((-1)^{0 \times 0} \right) \times \left((-1)^{1 \times 1} \right) \right\}$$

$$g(2,1) = \frac{1}{4} \left\{ (1) \times (-1) \right\}$$

$$g(2,1) = -\frac{1}{4}$$

Likewise, all the values of the Walsh transform can be calculated. After the calculation of all values, the basis for $N = 4$ is given below.

$$g(n, k) = \begin{pmatrix} +\frac{1}{4} & +\frac{1}{4} & +\frac{1}{4} & +\frac{1}{4} \\ +\frac{1}{4} & +\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ +\frac{1}{4} & -\frac{1}{4} & +\frac{1}{4} & -\frac{1}{4} \\ +\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & +\frac{1}{4} \end{pmatrix}$$

The MATLAB code to generate the Walsh basis for a given order is shown in Fig. 4.17.

The MATLAB code to determine the Walsh basis as given in Fig. 4.17 is executed and the output for order $N = 4$ is shown in Fig. 4.18.

Note Analyzing the Walsh basis, the value for every entity holds the same magnitude ($1/N$), but the only difference between them is the sign (whether it is positive or negative). So the shortcut method for finding the sign is given below;

Step 1 Write the binary representation of n .

Step 2 Write the binary representation of k in the reverse order.

Step 3 Check for the number of overlaps of 1 between n and k .

Step 4 If the number of overlaps of 1 is

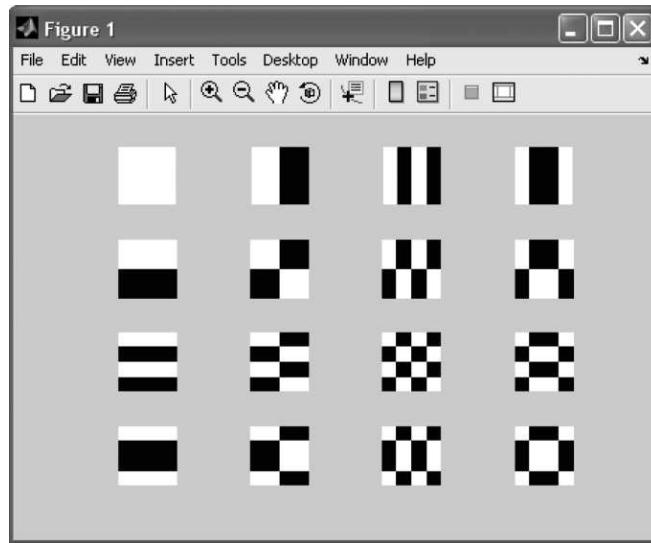
- i) zero then the sign is positive
- ii) even then the sign is positive
- iii) odd then the sign is negative

The Walsh Basis for $N = 8$

$$g(n, k) = \begin{pmatrix} +1/8 & +1/8 & +1/8 & +1/8 & +1/8 & +1/8 & +1/8 & +1/8 \\ +1/8 & +1/8 & +1/8 & +1/8 & -1/8 & -1/8 & -1/8 & -1/8 \\ +1/8 & +1/8 & -1/8 & -1/8 & +1/8 & +1/8 & -1/8 & -1/8 \\ +1/8 & +1/8 & -1/8 & -1/8 & -1/8 & +1/8 & +1/8 & +1/8 \\ +1/8 & -1/8 & +1/8 & -1/8 & +1/8 & -1/8 & +1/8 & -1/8 \\ +1/8 & -1/8 & +1/8 & +1/8 & -1/8 & +1/8 & -1/8 & +1/8 \\ +1/8 & -1/8 & -1/8 & +1/8 & +1/8 & -1/8 & -1/8 & +1/8 \\ +1/8 & -1/8 & -1/8 & +1/8 & -1/8 & +1/8 & +1/8 & -1/8 \end{pmatrix}$$

```
% This program displays the Walsh transform basis
clear all;
close all;
clc;
n=input('Enter the basis matrix dimension:');
m=n;
for u=0:n-1
    for v=0:n-1
        for x=0:n-1
            for y=0:n-1
                powervalue=1;
                sn=log2(n);
                for i=0:sn-1
                    a=dec2bin(x, sn);
                    b=bin2dec(a(sn-i));
                    c=dec2bin(y, sn);
                    d=bin2dec(c(sn-i));
                    e=dec2bin(u, sn);
                    f=bin2dec(e(i+1));
                    e=dec2bin(v, sn);
                    a=bin2dec(e(i+1));
                    powervalue=powervalue*(-1)^(b*f+d*a);
                end
                basis{u+1, v+1}(x+1, y+1)=powervalue;
            end
        end
    end
end
mag=basis;
figure(1)
k=1;
for i=1:m
    for j=1:n
        subplot(m, n, k)
        imshow(mag{i, j}, 256)
        k=k+1;
    end
end
```

Fig. 4.17 MATLAB code to generate Walsh basis for a given order

**Fig. 4.18** *Walsh basis functions for $N = 4$*

The tabular column for the corresponding n and k values is given in Table 4.4.

Table 4.4 *Walsh basis for $N = 8$*

$n \backslash k$	0	1	2	3	4	5	6	7
0	+1/8	+1/8	+1/8	+1/8	+1/8	+1/8	+1/8	+1/8
1	+1/8	+1/8	+1/8	+1/8	-1/8	-1/8	-1/8	-1/8
2	+1/8	+1/8	-1/8	-1/8	+1/8	+1/8	-1/8	-1/8
3	+1/8	+1/8	-1/8	-1/8	-1/8	+1/8	+1/8	+1/8
4	+1/8	-1/8	+1/8	(-1/8)	+1/8	-1/8	+1/8	-1/8
5	+1/8	-1/8	+1/8	+1/8	-1/8	+1/8	-1/8	+1/8
6	+1/8	-1/8	-1/8	+1/8	+1/8	-1/8	-1/8	+1/8
7	+1/8	-1/8	-1/8	+1/8	-1/8	+1/8	+1/8	-1/8

For example, we can go for $n = 4$ and $k = 3$.

Step 1 Write the binary representation of n .

$n = 4$ and its binary representation is 100

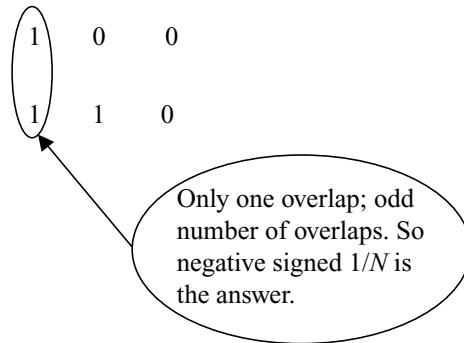
Step 2 Write the binary representation of k in the reverse order.

$k = 3$ and its binary representation is 011. When we reverse it, we get 110.

Step 3 Check for the number of overlaps of 1 between n and k .

$n = 4$ and its binary representation is 100.

$k = 3$ and its reverse binary representation is 110.



Step 4 If the number of overlaps is odd then the sign is negative which is given below.

$$g(4, 3) = -\frac{1}{8}$$

4.9 HADAMARD TRANSFORM

The Hadamard transform is basically the same as the Walsh transform except the rows of the transform matrix are re-ordered. The elements of the mutually orthogonal basis vectors of a Hadamard transform are either +1 or -1, which results in very low computational complexity in the calculation of the transform coefficients. Hadamard matrices are easily constructed for $N = 2^n$ by the following procedure

The order $N = 2$ Hadamard matrix is given as

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4.58)$$

The Hadamard matrix of order $2N$ can be generated by Kronecker product operation:

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix} \quad (4.59)$$

Substituting $N = 2$ in Eq. (4.59), we get

$$H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad (4.60)$$

Similarly, substituting $N = 4$ in Eq. (4.59), we get

$$H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \quad (4.61)$$

The Hadamard matrix of order $N = 2^n$ may be generated from the order two core matrix. It is not desirable to store the entire matrix.

4.10 HAAR TRANSFORM

The Haar transform is based on a class of orthogonal matrices whose elements are either 1, -1, or 0 multiplied by powers of $\sqrt{2}$. The Haar transform is a computationally efficient transform as the transform of an N -point vector requires only $2(N-1)$ additions and N multiplications.

Algorithm to Generate Haar Basis The algorithm to generate Haar basis is given below:

Step 1 Determine the order of N of the Haar basis.

Step 2 Determine n where $n = \log_2 N$.

Step 3 Determine p and q .

- (i) $0 \leq p < n-1$
- (ii) If $p = 0$ then $q = 0$ or $q = 1$
- (iii) If $p \neq 0$, $1 \leq q \leq 2^p$

Step 4 Determine k .

$$k = 2^p + q - 1$$

Step 5 Determine Z .

$$Z \rightarrow [0, 1) \Rightarrow \left\{ \frac{0}{N}, \quad \frac{1}{N}, \quad \dots \quad \frac{N-1}{N} \right\}$$

Step 6

$$\text{If } k = 0 \text{ then } H(Z) = \frac{1}{\sqrt{N}}$$

Otherwise,

$$H_k(Z) = H_{pq}(Z) = \frac{1}{\sqrt{N}} \begin{cases} +2^{p/2} & \frac{(q-1)}{2^p} \leq Z < \frac{(q-1/2)}{2^p} \\ -2^{p/2} & \frac{(q-1/2)}{2^p} \leq Z < \frac{q}{2^p} \\ 0 & \text{otherwise} \end{cases} \quad (4.62)$$

The flow chart to compute Haar basis is given Fig. 4.19.

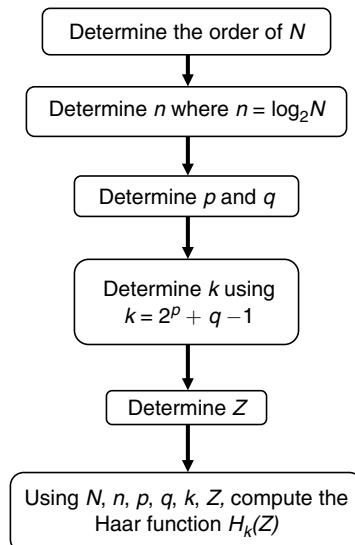


Fig. 4.19 Flow chart to compute Haar basis

Example 4.8 Generate one Haar Basis for $N = 2$.

Solution

Step 1 $N = 2$

Step 2 $n = \log_2 2 = 1$

Step 3

- (i) Since $n = 1$, the only value of p is 0.
- (ii) So q takes the value of 0 or 1.

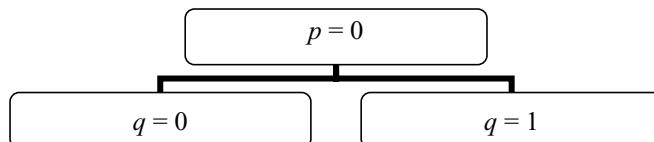
Step 4 Determining the value k using the formulae $k = 2^p + q - 1$

p	q	k
0	0	0
0	1	1

Step 5 Determining the Z Value.

$$Z \rightarrow [0, 1) \Rightarrow \left\{ \frac{0}{2}, \frac{1}{2} \right\}$$

Step 6



Case (i)

$$\text{If } k = 0 \text{ then } H(Z) = \frac{1}{\sqrt{N}} = \frac{1}{\sqrt{2}}; \in \forall Z$$

n	0	1	
k	0	1	
0	$1/\sqrt{2}$	$1/\sqrt{2}$	
1	—	—	
Z	0	1	

Since the value for k is 0, for all 'Z' $H(Z)$ is $1/\sqrt{2}$.

Case (ii)

For $k = 1; p = 0; q = 1$

Condition (i) $0 \leq Z < 1/2$

Condition (ii) $1/2 \leq Z < 1$

Condition (iii) Otherwise

$$H_k(Z) = H_{pq}(Z) = \frac{1}{\sqrt{2}} \begin{cases} +2^{p/2} \frac{(q-1)}{2^p} & \leq Z < \frac{(q-1/2)}{2^p} \\ -2^{p/2} \frac{(q-1/2)}{2^p} & \leq Z < \frac{q}{2^p} \\ 0 & \text{otherwise} \end{cases}$$

For $Z = 0$, The boundary condition (i) is been satisfied.

So,

$$H(Z) = \frac{1}{\sqrt{2}} \cdot 2^{0/2} = \frac{1}{\sqrt{2}}$$

For $Z = 1/2$, The boundary condition (ii) is been satisfied.

So,

$$H(Z) = -\frac{1}{\sqrt{2}} \cdot 2^{0/2} = -\frac{1}{\sqrt{2}}$$

The Haar basis for $N = 2$ is given below.

n	0	1
k		
0	$1/\sqrt{2}$	$1/\sqrt{2}$
1	$1/\sqrt{2}$	$-1/\sqrt{2}$

Example 4.9 Compute the Haar basis for $N = 8$.

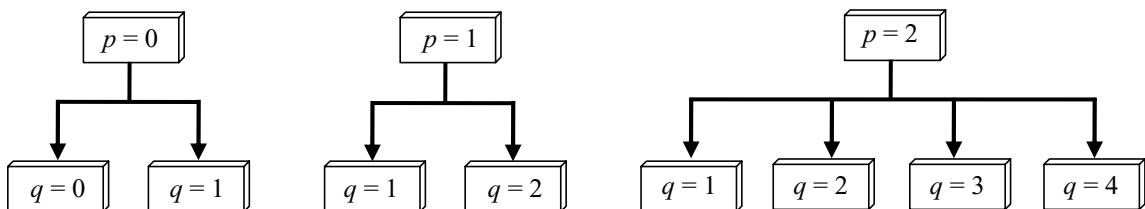
Solution

Step 1 Determine the order of $N = 8$.

Step 2 Determine n where $n = \log_2 N = \log_2 8 = \log_2 2^3 = 3 \cdot \log_2 2 = 3$

Step 3 Determine p and q .

- (i) $0 \leq p < 2$
- (ii) If $p = 0$ then $q = 0$ or $q = 1$
- (iii) If $p \neq 0$, $1 \leq q \leq 2^p$



Step 4 Determine k

$$k = 2^p + q - 1$$

Tabular column to get the k values for different possible combination of p and q .

Combinations	p	q	k
0	0	0	0
1	0	1	1
2	1	1	2
3	1	2	3
4	2	1	4
5	2	2	5
6	2	3	6
7	2	4	7

Step 5 Determine Z

$$Z \rightarrow [0, 1) \Rightarrow \left\{ \frac{0}{8}, \frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8} \right\}$$

$$Z \rightarrow [0, 1) \Rightarrow \left\{ 0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8} \right\}$$

Step 6

$$\text{If } k = 0 \text{ then } H(Z) = \frac{1}{\sqrt{8}} = \frac{1}{2\sqrt{2}} \quad \forall Z$$

Otherwise,

$$H_k(Z) = H_{pq}(Z) = \frac{1}{\sqrt{N}} \begin{cases} +2^{p/2} \frac{(q-1)}{2^p} & \leq Z < \frac{(q-1/2)}{2^p} \\ -2^{p/2} \frac{(q-1/2)}{2^p} & \leq Z < \frac{q}{2^p} \\ 0 & \text{otherwise} \end{cases}$$

When $k = 1$,

$p = 0; q = 1$;

Condition

$$(i) \quad 0 \leq z < \frac{1}{2} \Rightarrow H_1(z) = \frac{1}{\sqrt{N}} 2^{p/2}$$

$$(ii) \quad \frac{1}{2} \leq z < 1 \Rightarrow H_1(z) = \frac{-1}{\sqrt{N}} 2^{p/2}$$

$$(iii) \quad \text{Otherwise} \Rightarrow H_1(z) = 0$$

(a) For $z = 0$, the first condition is satisfied.

$$\text{So } 0 \leq z < \frac{1}{2} \Rightarrow H_1(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{0/2} = \frac{1}{2\sqrt{2}}$$

(b) For $z = 1/8$, the first condition is satisfied.

$$\text{So } 0 \leq z < \frac{1}{2} \Rightarrow H_1(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{0/2} = \frac{1}{2\sqrt{2}}$$

(c) For $z = 1/4$, the first condition is satisfied.

$$\text{So } 0 \leq z < \frac{1}{2} \Rightarrow H_1(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{0/2} = \frac{1}{2\sqrt{2}}$$

(d) For $z = 3/8$, the first condition is satisfied.

$$\text{So } 0 \leq z < \frac{1}{2} \Rightarrow H_1(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{0/2} = \frac{1}{2\sqrt{2}}$$

(e) For $z = 1/2$, the second condition is satisfied.

$$\text{So } \frac{1}{2} \leq z < 1 \Rightarrow H_1(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{0/2} = \frac{-1}{2\sqrt{2}}$$

(f) For $z = 5/8$, the second condition is satisfied.

$$\text{So } \frac{1}{2} \leq z < 1 \Rightarrow H_1(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{0/2} = \frac{-1}{2\sqrt{2}}$$

(g) For $z = 3/4$, the second condition is satisfied.

$$\text{So } \frac{1}{2} \leq z < 1 \Rightarrow H_1(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{0/2} = \frac{-1}{2\sqrt{2}}$$

(h) For $z = 7/8$, the second condition is satisfied.

$$\text{So } \frac{1}{2} \leq z < 1 \Rightarrow H_1(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{0/2} = \frac{-1}{2\sqrt{2}}$$

When $k = 2$,

$p = 1; q = 1$;

Condition

$$(i) \quad 0 \leq z < \frac{1}{4} \Rightarrow H_2(z) = \frac{1}{\sqrt{N}} 2^{p/2}$$

$$(ii) \quad \frac{1}{4} \leq z < \frac{1}{2} \Rightarrow H_2(z) = \frac{-1}{\sqrt{N}} 2^{p/2}$$

$$(iii) \quad \text{Otherwise} \Rightarrow H_1(z) = 0$$

(a) For $z = 0$, the first condition is satisfied.

$$\text{So } 0 \leq z < \frac{1}{4} \Rightarrow H_2(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{1/2} = \frac{1}{2\sqrt{2}} \times \sqrt{2} = \frac{1}{2}$$

(b) For $z = 1/8$, the first condition is satisfied.

$$\text{So } 0 \leq z < \frac{1}{4} \Rightarrow H_2(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{1/2} = \frac{1}{2\sqrt{2}} \times \sqrt{2} = \frac{1}{2}$$

(c) For $z = 1/4$, the second condition is satisfied.

$$\text{So } \frac{1}{4} \leq z < \frac{1}{2} \Rightarrow H_2(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{1/2} = \frac{-1}{2\sqrt{2}} \times \sqrt{2} = -\frac{1}{2}$$

(d) For $z = 3/8$, the second condition is satisfied.

$$\text{So } \frac{1}{4} \leq z < \frac{1}{2} \Rightarrow H_2(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{1/2} = \frac{-1}{2\sqrt{2}} \times \sqrt{2} = -\frac{1}{2}$$

(e) For $z = 1/2$, the third condition is satisfied.

$$\text{So otherwise} \Rightarrow H_2(z) = 0$$

(f) For $z = 5/8$, the third condition is satisfied.

$$\text{So otherwise} \Rightarrow H_2(z) = 0$$

(g) For $z = 3/4$, the third condition is satisfied.

$$\text{So otherwise} \Rightarrow H_2(z) = 0$$

(h) For $z = 7/8$, the third condition is satisfied.

$$\text{So otherwise} \Rightarrow H_2(z) = 0$$

When $k = 3$,

$p = 1; q = 3$;

Condition

$$(i) \quad \frac{1}{2} \leq z < \frac{3}{4} \Rightarrow H_3(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{1/2} = \frac{1}{2\sqrt{2}} \times \sqrt{2} = \frac{1}{2}$$

$$(ii) \frac{3}{4} \leq z < 1 \Rightarrow H_3(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = -\frac{1}{\sqrt{8}} 2^{1/2} = \frac{-1}{2\sqrt{2}} \times \sqrt{2} = -\frac{1}{2}$$

(iii) Otherwise $\Rightarrow H_3(z) = 0$

(a) For $z = 0, 1/8, 1/4, 3/8$, the third condition is satisfied.

$$\text{So Otherwise } \Rightarrow H_3(z) = 0$$

(b) For $z = 1/2, 5/8$, the first condition is satisfied.

$$\text{So } \frac{1}{2} \leq z < \frac{3}{4} \Rightarrow H_3(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{1/2} = \frac{1}{2\sqrt{2}} \times \sqrt{2} = \frac{1}{2}$$

(c) For $z = 3/4, 7/8$, the second condition is satisfied.

$$\text{So } \frac{3}{4} \leq z < 1 \Rightarrow H_3(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = -\frac{1}{\sqrt{8}} 2^{1/2} = \frac{-1}{2\sqrt{2}} \times \sqrt{2} = -\frac{1}{2}$$

When $k = 4$,

$p = 2; q = 1$;

Condition

$$(i) \ 0 \leq z < \frac{1}{8} \Rightarrow H_4(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

$$(ii) \ \frac{1}{8} \leq z < \frac{1}{4} \Rightarrow H_4(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

(iii) Otherwise $\Rightarrow H_4(z) = 0$

(a) For $z = 0$, the first condition is satisfied.

$$\text{So } 0 \leq z < \frac{1}{8} \Rightarrow H_4(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

(b) For $z = 1/8$, the second condition is satisfied.

$$\text{So } \frac{1}{8} \leq z < \frac{1}{4} \Rightarrow H_4(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

(c) For $z = 1/4, 3/8, 1/2, 5/8, 3/4, 7/8$, the second condition is satisfied.

$$\text{So otherwise } \Rightarrow H_4(z) = 0$$

When $k = 5$,

$p = 2; q = 2$;

Condition

$$(i) \ \frac{1}{4} \leq z < \frac{3}{8} \Rightarrow H_5(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

$$(ii) \frac{3}{8} \leq z < \frac{1}{2} \Rightarrow H_5(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

(iii) Otherwise $\Rightarrow H_5(z) = 0$

(a) For $z = 0, 1/8$, the third condition is satisfied.

$$\text{So otherwise } \Rightarrow H_5(z) = 0$$

(b) For $z = 1/4$, the first condition is satisfied.

$$\text{So } \frac{1}{4} \leq z < \frac{3}{8} \Rightarrow H_5(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

(c) For $z = 3/8$, the second condition is satisfied.

$$\text{So } \frac{3}{8} \leq z < \frac{1}{2} \Rightarrow H_5(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

(d) For $z = 1/2, 5/8, 3/4, 7/8$, the third condition is satisfied.

$$\text{So otherwise } \Rightarrow H_5(z) = 0$$

When $k = 6$,

$p = 2; q = 3$;

Condition

$$(i) \frac{1}{2} \leq z < \frac{5}{8} \Rightarrow H_6(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

$$(ii) \frac{5}{8} \leq z < \frac{3}{4} \Rightarrow H_6(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

(iii) Otherwise $\Rightarrow H_6(z) = 0$

(a) For $z = 0, 1/8, 1/4, 3/8, 3/4, 7/8$, the third condition is satisfied.

$$\text{So otherwise } \Rightarrow H_6(z) = 0$$

(b) For $z = 1/2$, the first condition is satisfied.

$$\text{So } \frac{1}{2} \leq z < \frac{5}{8} \Rightarrow H_6(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

(c) For $z = 5/8$, the second condition is satisfied.

$$\text{So } \frac{5}{8} \leq z < \frac{3}{4} \Rightarrow H_6(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

When $k = 7$,

$p = 2$; $q = 4$;

Condition

$$(i) \frac{3}{4} \leq z < \frac{7}{8} \Rightarrow H_7(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

$$(ii) \frac{7}{8} \leq z < 1 \Rightarrow H_7(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

$$(iii) \text{ Otherwise } \Rightarrow H_7(z) = 0$$

(a) For $z = 0, 1/8, 1/4, 3/8, 1/2, 5/8$, the third condition is satisfied.

$$H_7(z) = 0$$

(b) For $z = 6/8$, the first condition is satisfied.

$$\text{So } \frac{3}{4} \leq z < \frac{7}{8} \Rightarrow H_7(z) = \frac{1}{\sqrt{N}} 2^{p/2} = \frac{1}{\sqrt{8}} 2^{2/2} = \frac{1}{2\sqrt{2}} \times 2 = \frac{1}{\sqrt{2}}$$

(c) For $z = 7/8$, the second condition is satisfied.

$$\text{So } \frac{7}{8} \leq z < 1 \Rightarrow H_7(z) = \frac{-1}{\sqrt{N}} 2^{p/2} = \frac{-1}{\sqrt{8}} 2^{2/2} = \frac{-1}{2\sqrt{2}} \times 2 = -\frac{1}{\sqrt{2}}$$

The Haar basis for $N = 8$ is given in Table 4.5.

Table 4.5 Haar basis for $N = 8$

$k \backslash n$	0	1	2	3	4	5	6	7
0	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$
1	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$	$-\frac{1}{2\sqrt{2}}$	$-\frac{1}{2\sqrt{2}}$	$-\frac{1}{2\sqrt{2}}$	$-\frac{1}{2\sqrt{2}}$
2	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	0	0	0	0
3	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
4	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	0	0	0	0	0	0
5	0	0	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	0	0	0	0
6	0	0	0	0	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	0	0
7	0	0	0	0	0	0	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$
Z	0/8	1/8	2/8	3/8	4/8	5/8	6/8	7/8

MATLAB code for the generation of Haar basis

The MATLAB code for the generation of Haar basis of order N is shown in Fig. 4.20.

The code given in Fig. 4.20 is executed for order $N = 2$ and the corresponding result is shown in Fig. 4.21.

```
% This program will Haar transfrom basis of given order
clear all;
close all;
clc;
n=input('Enter the basis matrix dimension: ');
m=n;
for u=0:n-1
    for v=0:n-1
        for x=0:n-1
            for y=0:n-1
                powervalue=0;
                sn=log2(n);
                for i=0:sn-1
                    a=dec2bin(x, sn);
                    b=bin2dec(a(sn-i));
                    c=dec2bin(y, sn);
                    d=bin2dec(c(sn-i));
                    e=dec2bin(u, sn);
                    f=bin2dec(e(sn-i));
                    e=dec2bin(v, sn);
                    a=bin2dec(e(sn-i));
                    powervalue=powervalue+(b*f+d*a);
                end
                basis{u+1, v+1}(x+1, y+1)=(-1)^powervalue;
            end
        end
    end
end
mag=basis;
figure(4)
k=1;
%Code to plot Haar basis
for i=1:m
    for j=1:n
        subplot(m, n, k)
        imshow(mag{i, j}, 256)
        k=k+1;
    end
end
```

Fig. 4.20 *MATLAB code to compute and plot Haar basis*

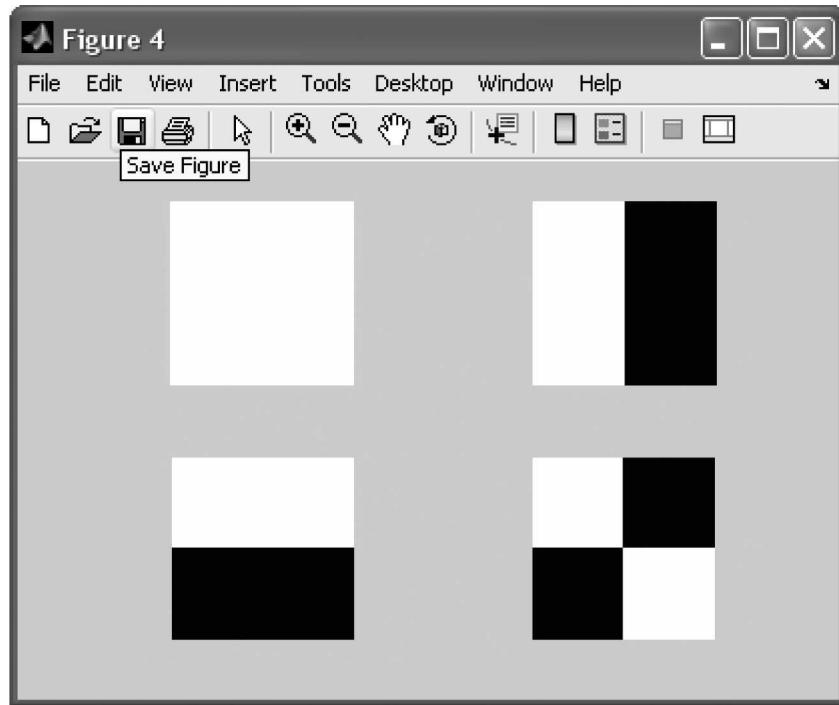


Fig. 4.21 Haar basis for $N = 2$

4.11 SLANT TRANSFORM

The slant transform was introduced by Enomoto and Shibata as an orthogonal transform containing sawtooth waveforms or ‘slant’ basis vectors. A slant basis vector \mathbf{v} that is monotonically decreasing in constant steps from maximum to minimum has the sequency property and has a fast computational algorithm. Let S_N denote an $N \times N$ slant matrix with $N = 2^n$. Then

$$S_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4.63)$$

The S_4 matrix is obtained by the following operation:

$$S_4 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ a & b & -a & b \\ 0 & 1 & 0 & -1 \\ -b & a & b & a \end{bmatrix} \begin{bmatrix} S_2 & 0 \\ 0 & S_2 \end{bmatrix}$$

If $a = 2b$ and $b = \frac{1}{\sqrt{5}}$, the slant matrix is given by

$$S_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{\sqrt{5}} & \frac{1}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 1 & -1 & -1 & 1 \\ \frac{1}{\sqrt{5}} & -\frac{3}{\sqrt{5}} & \frac{3}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{bmatrix} \quad (4.64)$$

The sequency of the slant matrix of order four is given below:

$$S_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{\sqrt{5}} & \frac{1}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & -\frac{3}{\sqrt{5}} \\ 1 & -1 & -1 & 1 \\ \frac{1}{\sqrt{5}} & -\frac{3}{\sqrt{5}} & \frac{3}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{bmatrix} \quad \begin{array}{l} \text{Sequency} \\ \text{Zero sign change} \\ \text{One sign change} \\ \text{Two sign changes} \\ \text{Three sign changes} \end{array}$$

From the sequency property, it is clear that the rows are ordered by the number of sign changes.

The slant transform reproduces linear variations of brightness very well. However, its performance at edges is not as optimal as the KLT or DCT. Because of the ‘slant’ nature of the lower order coefficients, its effect is to smear the edges.

4.12 DISCRETE COSINE TRANSFORM

The Discrete Cosine Transform was developed by Ahmed, Natarajan and Rao in 1974. The discrete cosine transforms are the members of a family of real-valued discrete sinusoidal unitary transforms. A discrete cosine transform consists of a set of basis vectors that are sampled cosine functions. DCT is a technique for converting a signal into elementary frequency components and it is widely used in image compression.

If $x[n]$ is the signal of length N , the Fourier transform of the signal $x[n]$ is given by $X[k]$
where

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi k n}{N}} \quad (4.65)$$

where k varies between 0 to $N - 1$.

Now consider the extension of the signal $x[n]$ which is denoted by $x_e[n]$ so that the length of the extended sequence is $2N$. The sequence $x[n]$ can be extended in two ways.

Let us consider a sequence (original sequence) of length four given by $x[n] = [1, 2, 3, 4]$. The original sequence is shown in Fig. 4.22. The sequence can be extended in two ways. The original sequence can be extended by simply copying the original sequence again which is shown in Fig. 4.23.

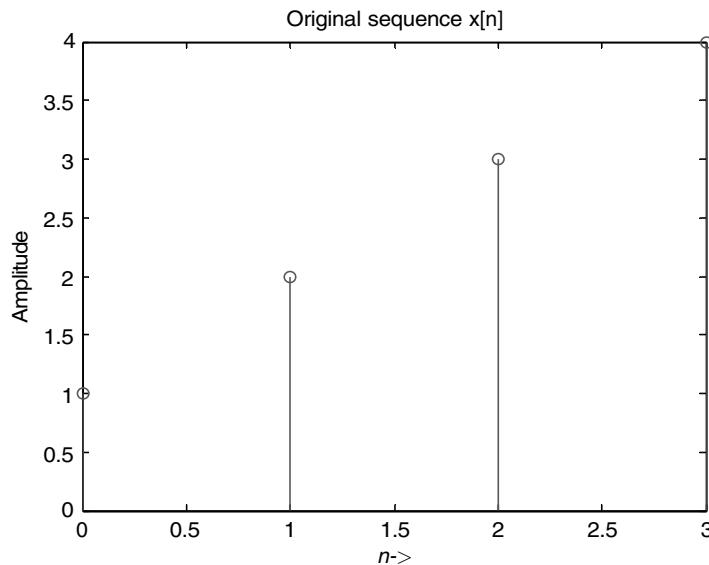


Fig. 4.22 Original sequence

The extended sequence can be produced by simply copying the original sequence as shown in Fig. 4.23.

The main drawback of this method is the variation in the value of the sample at $n = 3$ and at $n = 4$. Since the variation is drastic, the phenomenon of 'ringing' is inevitable. To overcome this, a second method of obtaining the extended sequence is by copying the original sequence in a folded manner as shown in Fig. 4.24.

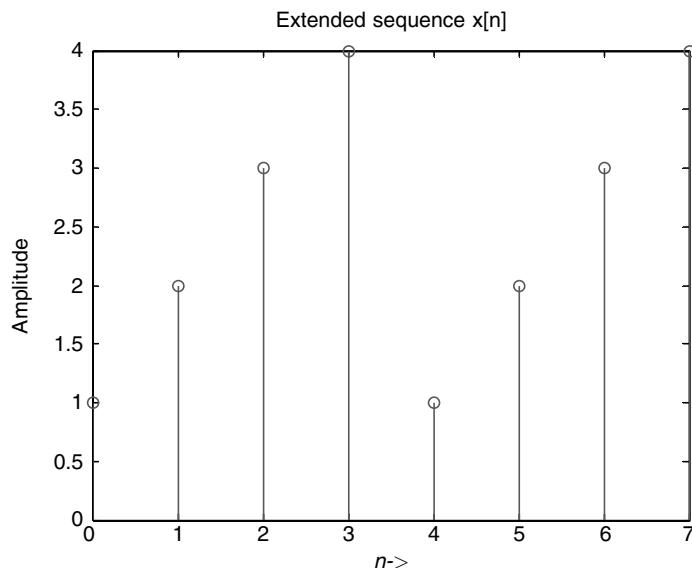


Fig. 4.23 Extended sequence obtained by simply copying the original sequence

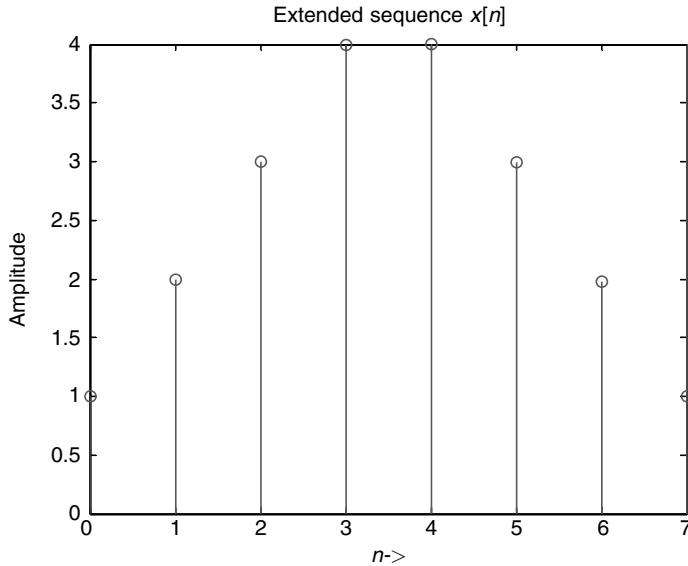


Fig. 4.24 Extended sequence obtained by folding the original sequence

Now comparing Fig. 4.23 with Fig. 4.24, it is clear that the variation in the value of the sample at $n = 3$ and at $n = 4$ is minimum in Fig. 4.24 when compared to Fig. 4.23. From this, the extended sequence obtained by folding the first sequence is found to be a better choice.

From both Fig. 4.23 and Fig. 4.24, we can observe that the length of the extended sequence is $2N$ if N is the length of the original sequence. In our case, the length of the original sequence is 4 (refer Fig. 4.22) and the length of the extended sequence is 8 (refer Fig. 4.23 and Fig. 4.24).

The Discrete Fourier Transform (DFT) of the extended sequence is given by $X_e[k]$ where

$$X_e[k] = \sum_{n=0}^{2N-1} x_e[n] e^{-j \frac{2\pi kn}{2N}} \quad (4.66)$$

Now, we can split the interval 0 to $2N - 1$ into two.

$$X_e[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi kn}{2N}} + \sum_{n=N}^{2N-1} x[2N-1-n] e^{-j \frac{2\pi kn}{2N}} \quad (4.67)$$

Let $m = 2N - 1 - n$. Substituting in Eq. (4.67), we get

$$X_e[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi kn}{2N}} + \sum_{m=0}^{N-1} x[m] e^{-j \frac{2\pi k(2N-1-m)}{2N}}$$

$$X_e[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi kn}{2N}} + \sum_{m=0}^{N-1} x[m] e^{-j \frac{2\pi k}{2N} 2N} e^{\frac{j2\pi}{2N}(m+1)} \quad (4.68)$$

$$\text{But } e^{-\frac{j2\pi k}{2N}2N} = e^{-j2\pi k} = \cos(2\pi k) - j\sin(2\pi k) = 1 \quad (4.69)$$

Substituting (4.69) in (4.68), we get

$$X_e[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{2N}} + \sum_{m=0}^{N-1} x(m) \cdot 1 \cdot e^{\frac{j2\pi(m+1)}{2N}} \quad (4.70)$$

$$X_e[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{j2\pi kn}{2N}} + \sum_{m=0}^{N-1} x[m] e^{\frac{j2\pi(m+1)}{2N}} \quad (4.71)$$

Replacing m by n we get

$$X_e[k] = \sum_{n=0}^{N-1} x[n] \{ e^{-\frac{j2\pi kn}{2N}} + e^{\frac{j2\pi(n+1)}{2N}} \} \quad (4.72)$$

Multiplying by $e^{\frac{-j\pi k}{2N}}$ on both sides of Eq. (4.72), we get

$$X_e[k] e^{\frac{-j\pi k}{2N}} = \sum_{n=0}^{N-1} x[n] \{ e^{-\frac{j2\pi kn}{2N}} + e^{\frac{j2\pi(n+1)}{2N}} \} e^{\frac{-j\pi k}{2N}} \quad (4.73)$$

Taking $e^{\frac{-j\pi k}{2N}}$ inside, we get

$$X_e[k] e^{\frac{-j\pi k}{2N}} = \sum_{n=0}^{N-1} x[n] \{ e^{-\frac{j2\pi kn}{2N}} e^{\frac{-j\pi k}{2N}} + e^{\frac{j2\pi(n+1)}{2N}} e^{\frac{-j\pi k}{2N}} \} \quad (4.74)$$

Upon simplifying Eq. 4.74, we get

$$X_e[k] e^{\frac{-j\pi k}{2N}} = \sum_{n=0}^{N-1} x[n] \{ e^{-\frac{j\pi kn}{2N}(2n+1)} + e^{\frac{j\pi kn}{2N}(2n+1)} \} \quad (4.75)$$

Multiplying and dividing by 2 on the RHS of Eq. (4.75), we get

$$X_e[k] e^{\frac{-j\pi k}{2N}} = \sum_{n=0}^{N-1} 2x[n] \cos \left\{ \frac{(2n+1)\pi k}{2N} \right\} \quad (4.76)$$

Upon simplification, we get

$$\frac{X_e[k] e^{\frac{-j\pi k}{2N}}}{2} = \sum_{n=0}^{N-1} x[n] \cos \left\{ \frac{(2n+1)\pi k}{2N} \right\} \quad (4.77)$$

Thus, the kernel of a one-dimensional discrete cosine transform is given by

$$X[k] = \alpha(k) \sum_{n=0}^{N-1} x[n] \cos \left\{ \frac{(2n+1)\pi k}{2N} \right\}, \text{ where } 0 \leq k \leq N-1 \quad (4.78)$$

$$\alpha(k) = \sqrt{\frac{1}{N}} \text{ if } k = 0$$

$$\alpha(k) = \sqrt{\frac{2}{N}} \text{ if } k \neq 0$$

The process of reconstructing a set of spatial domain samples from the DCT coefficients is called the inverse discrete cosine transform (IDCT). The inverse discrete cosine transformation is given by

$$x[n] = \alpha(k) \sum_{k=0}^{N-1} X[k] \cos \left[\frac{(2n+1)\pi k}{2N} \right], \quad 0 \leq n \leq N-1 \quad (4.79)$$

The forward 2D discrete cosine transform of a signal $f(m, n)$ is given by

$$F[k, l] = \alpha(k) \alpha(l) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) \cos \left[\frac{(2m+1)\pi k}{2N} \right] \cos \left[\frac{(2n+1)\pi l}{2N} \right] \quad (4.80)$$

$$\text{where } \alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } k = 0 \\ \sqrt{\frac{2}{N}} & \text{if } k \neq 0 \end{cases}$$

$$\text{Similarly } \alpha(l) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } l = 0 \\ \sqrt{\frac{2}{N}} & \text{if } l \neq 0 \end{cases}$$

The 2D inverse discrete cosine transform is given by

$$f[m, n] = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \alpha(k) \alpha(l) F(k, l) \cos \left[\frac{(2m+1)\pi k}{2N} \right] \cos \left[\frac{(2n+1)\pi l}{2N} \right] \quad (4.81)$$

Example 4.10 Compute the discrete cosine transform (DCT) matrix for $N = 4$.

Solution The formula to compute the DCT matrix is given by

$$X[k] = \alpha(k) \sum_{n=0}^{N-1} x[n] \cos \left[\frac{(2n+1)\pi k}{2N} \right], \text{ where } 0 \leq k \leq N-1$$

where

$$\alpha(k) = \sqrt{\frac{1}{N}} \text{ if } k = 0$$

$$\alpha(k) = \sqrt{\frac{2}{N}} \text{ if } k \neq 0$$

In our case, the value of $N = 4$. Substituting $N = 4$ in the expression of $X[k]$ we get

$$X[k] = \alpha(k) \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1)\pi k}{8} \right] \quad (4.82)$$

Substituting $k = 0$ in Eq. (4.82), we get

$$\begin{aligned} X[0] &= \sqrt{\frac{1}{4} \times \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1)\pi \times 0}{8} \right]} \\ &= \frac{1}{2} \times \sum_{n=0}^3 x[n] \cos(0) = \frac{1}{2} \times \sum_{n=0}^3 x[n] \times 1 \\ &= \frac{1}{2} \times \sum_{n=0}^3 x[n] \\ &= \frac{1}{2} \times \{x(0) + x(1) + x(2) + x(3)\} \\ X[0] &= \frac{1}{2} x(0) + \frac{1}{2} x(1) + \frac{1}{2} x(2) + \frac{1}{2} x(3) \end{aligned} \quad (4.83)$$

Substituting $k = 1$ in Eq. (4.82), we get

$$\begin{aligned} X[1] &= \sqrt{\frac{2}{4} \times \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1)\pi \times 1}{8} \right]} \\ &= \sqrt{\frac{1}{2} \times \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1)\pi}{8} \right]} \\ &= \sqrt{\frac{1}{2} \times \left\{ x(0) \cos \left(\frac{\pi}{8} \right) + x(1) \cos \left(\frac{3\pi}{8} \right) + x(2) \cos \left(\frac{5\pi}{8} \right) + x(3) \cos \left(\frac{7\pi}{8} \right) \right\}} \\ &= 0.707 \times \{x(0) \times 0.9239 + x(1) \times 0.3827 + x(2) \times (-0.3827) + x(3) \times (-0.9239)\} \end{aligned}$$

$$X(1) = 0.6532x(0) + 0.2706x(1) - 0.2706x(2) - 0.6532x(3) \quad (4.84)$$

Substituting $k = 2$ in Eq. (4.82), we get

$$\begin{aligned} X(2) &= \sqrt{\frac{2}{4} \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1)\pi \times 2}{8} \right]} \\ &= \sqrt{\frac{1}{2} \times \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1)\pi}{4} \right]} \\ &= \sqrt{\frac{1}{2} \times \left\{ x(0) \cos \left(\frac{\pi}{4} \right) + x(1) \cos \left(\frac{3\pi}{4} \right) + x(2) \cos \left(\frac{5\pi}{4} \right) + x(3) \cos \left(\frac{7\pi}{4} \right) \right\}} \end{aligned}$$

$$\begin{aligned}
&= 0.7071 \times \{x(0) \times 0.7071 + x(1) \times (-0.7071) + x(2) \times (-0.7071) + x(3) \times (0.7071)\} \\
X(2) &= 0.5x(0) - 0.5x(1) - 0.5x(2) + 0.5x(3) \tag{4.85}
\end{aligned}$$

Substituting $k = 3$ in Eq. (4.82), we get

$$\begin{aligned}
X(3) &= \sqrt{\frac{2}{4}} \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1)\pi \times 3}{8} \right] \\
&= \sqrt{\frac{1}{2}} \times \sum_{n=0}^3 x[n] \cos \left[\frac{(2n+1) \times 3\pi}{8} \right] \\
&= \sqrt{\frac{1}{2}} \times \left\{ x(0) \times \cos \left(\frac{3\pi}{8} \right) + x(1) \times \cos \left(\frac{9\pi}{8} \right) + x(2) \times \cos \left(\frac{15\pi}{8} \right) + x(3) \times \cos \left(\frac{21\pi}{8} \right) \right\} \\
&= 0.7071 \times \{x(0) \times 0.3827 + x(1) \times (-0.9239) + x(2) \times (0.9239) + x(3) \times (-0.3827)\}
\end{aligned}$$

$$X(3) = 0.2706x(0) - 0.6533x(1) + 0.6533x(2) - 0.2706x(3) \tag{4.86}$$

Collecting the coefficients of $x(0)$, $x(1)$, $x(2)$ and $x(3)$ from $X(0)$, $X(1)$, $X(2)$ and $X(3)$, we get

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6532 & 0.2706 & -0.2706 & -0.6532 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix} \tag{4.87}$$

The MATLAB code to obtain the DCT matrix of order 4 is shown in Fig. 4.25.

Properties of Discrete Cosine Transform

The discrete cosine transform is real and orthogonal.

If A is a DCT matrix of order N , and if the matrix A is orthogonal then

$$A \times A^T = I \tag{4.88}$$

Let A be a DCT matrix of order four. The matrix A is given by

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6532 & 0.2706 & -0.2706 & -0.6532 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix}$$

$$\text{Then } A^T = \begin{bmatrix} 0.5 & 0.6532 & 0.5 & 0.2706 \\ 0.5 & 0.2706 & -0.5 & -0.6532 \\ 0.5 & -0.2706 & -0.5 & 0.6533 \\ 0.5 & -0.6532 & 0.5 & -0.2706 \end{bmatrix}$$

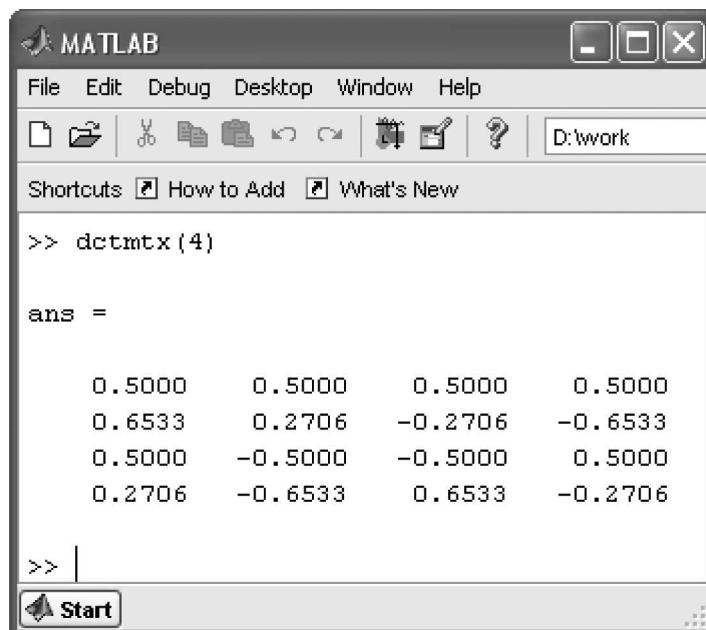


Fig. 4.25 DCT matrix of order four

$$A \times A^T = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6532 & 0.2706 & -0.2706 & -0.6532 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix} \times \begin{bmatrix} 0.5 & 0.6532 & 0.5 & 0.2706 \\ 0.5 & 0.2706 & -0.5 & -0.6532 \\ 0.5 & -0.2706 & -0.5 & 0.6533 \\ 0.5 & -0.6532 & 0.5 & -0.2706 \end{bmatrix}$$

$$A \times A^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The above result implies $A^{-1} = A^T$

1. The discrete cosine transform has excellent energy compaction for highly correlated data.
2. *Separable property* A 2D DCT can be computed as two separate one-dimensional transforms. Because of this separable property, the 2D DCT of an image can be computed in two steps by successive 1D operations on rows and columns of an image.

MATLAB Code to Compute DCT Basis The MATLAB code to compute and plot DCT basis is given in Fig. 4.26.

```

% DCT basis calculation
clear all;
close all;
clc;
m=input('Enter the basis matrix dimension:');
n=m;
alpha2=ones(1, n)*sqrt(2/n);
alpha2(1)=sqrt(1/n);
alpha1=ones(1, m)*sqrt(2/m);
alpha1(1)=sqrt(1/m);
for u=0:m-1
    for v=0:n-1
        for x=0:m-1
            for y=0:n-1
                a{u+1, v+1}(x+1, y+1)=alpha1(u+1)*alpha2(v+1)*...
                cos((2*x+1)*u*pi/(2*n))*cos((2*y+1)*v*pi/(2*n));
            end
        end
    end
end
mag=a;
figure(3)
k=1;
% Code to plot the basis
for i=1:m
    for j=1:n
        subplot(m, n, k)
        imshow(mag{i, j}, 256)
        k=k+1;
    end
end

```

Fig. 4.26 MATLAB code to compute and plot DCT basis

The code given in Fig. 4.26 is executed for $N = 4$ and the corresponding result is shown in Fig. 4.27.

4.13 KARHUNEN-LOEVE TRANSFORM (KL TRANSFORM)

The KL transform is named after Kari Karhunen and Michel Loeve who developed it as a series expansion method for continuous random processes. Originally, Harold Hotelling studied the discrete formulation of the KL transform and for this reason, the KL transform is also known as the Hotelling transform.

The KL transform is a reversible linear transform that exploits the statistical properties of a vector representation. The basic functions of the KL transform are orthogonal eigen vectors of the covariance matrix of a data set. A KL transform optimally decorrelates the input data. After a KL transform, most of the ‘energy’ of the transform coefficients is concentrated within the first few components. This is the energy compaction property of a KL transform.

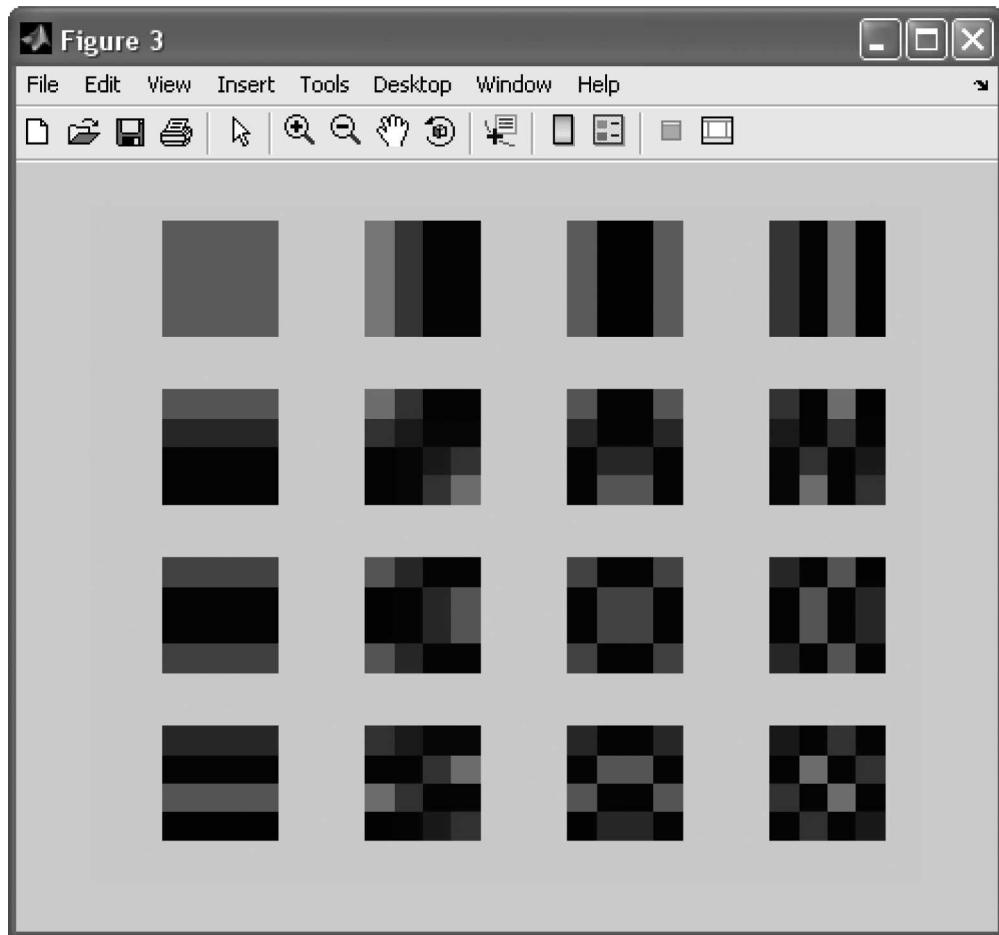


Fig. 4.27 DCT basis for $N = 4$

Drawbacks of KL Transforms

The two serious practical drawbacks of KL transform are the following:

- i. A KL transform is input-dependent and the basic function has to be calculated for each signal model on which it operates. The KL bases have no specific mathematical structure that leads to fast implementations.
- ii. The KL transform requires $O(m^2)$ multiply/add operations. The DFT and DCT require $O(\log_2^m)$ multiplications.

Applications of KL Transforms

(i) *Clustering Analysis* The KL transform is used in clustering analysis to determine a new coordinate system for sample data where the largest variance of a projection of the data lies on the first axis, the next largest variance on the second axis, and so on. Because these axes are orthogonal, this approach allows for reducing the dimensionality of the data set by eliminating those coordinate axes with small variances. This data-reduction technique is commonly referred as Principle Component Analysis (PCA).

(ii) *Image Compression* The KL transform is heavily utilised for performance evaluation of compression algorithms since it has been proven to be the optimal transform for the compression of an image sequence in the sense that the KL spectrum contains the largest number of zero-valued coefficients.

Example 4.11 Perform KL transform for the following matrix:

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

Solution

Step 1 Formation of vectors from the given matrix

The given matrix is a 2×2 matrix; hence two vectors can be extracted from the given matrix. Let it be x_0 and x_1 .

$$x_0 = \begin{bmatrix} 4 \\ -1 \end{bmatrix} \text{ and } x_1 = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$$

Step 2 Determination of covariance matrix

The formula to compute covariance of the matrix is $\text{cov}(x) = E[xx^T] - \bar{x}\bar{x}^T$

In the formula for covariance matrix, \bar{x} denotes the mean of the input matrix. The formula to compute the mean of the given matrix is given below:

$$\bar{x} = \frac{1}{M} \sum_{k=0}^{M-1} x_k \quad (4.89)$$

where M is the number of vectors in x .

$$\begin{aligned} \bar{x} &= \frac{1}{2} \sum_{k=0}^1 x_k = \frac{1}{2} \{x_0 + x_1\} = \frac{1}{2} \left\{ \begin{bmatrix} 4 \\ -1 \end{bmatrix} + \begin{bmatrix} -2 \\ 3 \end{bmatrix} \right\} \\ &= \frac{1}{2} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

The mean value is calculated as $\bar{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Now multiplying the mean value with its transpose yields

$$\bar{x}\bar{x}^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Now to find the $E[xx^T]$, we use the formula

$$E[xx^T] = \frac{1}{M} \sum_{k=0}^{M-1} x_k x_k^T \quad (4.90)$$

In our case, $M = 2$ hence

$$E[xx^T] = \frac{1}{2} \sum_{k=0}^1 x_k x_k^T = \frac{1}{2} \left\{ \begin{bmatrix} 4 \\ -1 \end{bmatrix} \begin{bmatrix} 4 & -1 \end{bmatrix} + \begin{bmatrix} -2 \\ 3 \end{bmatrix} \begin{bmatrix} -2 & 3 \end{bmatrix} \right\}$$

$$E[xx^T] = \frac{1}{2} \left\{ \begin{bmatrix} 16 & -4 \\ -4 & 1 \end{bmatrix} + \begin{bmatrix} 4 & -6 \\ -6 & 9 \end{bmatrix} \right\} = \begin{bmatrix} 10 & -5 \\ -5 & 5 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

Now using the value of $E[xx^T]$ and $\bar{x} \bar{x}^T$, we find the covariance matrix,

$$\text{cov}(x) = E[xx^T] - \bar{x} \bar{x}^T \quad (4.91)$$

$$\text{cov}(x) = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -2 \\ -2 & 0 \end{bmatrix}$$

Step 3 Determination of eigen values of the covariance matrix

To find the eigen values λ , we solve the characteristic equation,

$$|\text{cov}(x) - \lambda I| = 0 \quad (4.92)$$

$$\det \left(\begin{bmatrix} 1 & -2 \\ -2 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) = 0$$

$$\det \left(\begin{bmatrix} 1 & -2 \\ -2 & 0 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right) = 0$$

$$\det \left(\begin{bmatrix} 1-\lambda & -2 \\ -2 & -\lambda \end{bmatrix} \right) = 0$$

$$(1-\lambda)(-\lambda) + 2(-2) = 0$$

$$\begin{aligned} -\lambda + \lambda^2 - 4 &= 0 \\ \lambda^2 - \lambda - 4 &= 0 \end{aligned}$$

From the last equation, we have to find the eigen values λ_0 and λ_1 . Solving the above equation, we get

$$\lambda = \frac{1 \pm \sqrt{1+16}}{2} = \frac{1 \pm \sqrt{17}}{2} = \frac{1 \pm 4.1231}{2}$$

$$\lambda_0 = \frac{1+4.1231}{2} = 2.5615$$

$$\lambda_1 = \frac{1-4.1231}{2} = -1.5615$$

Therefore, the eigenvalues of $\text{cov}(x)$ are $\lambda_0 = 2.5615$ and $\lambda_1 = -1.5615$.

Step 4 Determination of eigen vectors of the covariance matrix

The first eigen vector ϕ_0 is found from the equation,

$$(\text{cov}(x) - \lambda_0 I) \phi_0 = 0.$$

$$\begin{aligned} (\text{cov}(x) - \lambda_0 I) \phi_0 &= \begin{bmatrix} 1 & -2 \\ -2 & 0 \end{bmatrix} - \begin{bmatrix} 2.5615 & 0 \\ 0 & 2.5615 \end{bmatrix} \begin{bmatrix} \phi_{00} \\ \phi_{01} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -1.5615 & -2 \\ -2 & -2.5615 \end{bmatrix} \begin{bmatrix} \phi_{00} \\ \phi_{01} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

Using row detection method, we have found, ϕ_{01} to be a free variable. So, we choose the value of ϕ_{01} as 1.

$$-1.5615\phi_{00} - 2\phi_{01} = 0$$

$$\phi_{00} = \frac{2}{-1.5615} = -1.2808$$

$$\text{The eigen vector } \phi_0 = \begin{bmatrix} -1.2808 \\ 1 \end{bmatrix}$$

Similarly, find the next eigen vector ϕ_1 ; the eigen value is $\lambda_1 = -1.5615$

$$\begin{aligned} (\text{cov}(x) - \lambda_1 I) \phi_1 &= \begin{bmatrix} 1 & -2 \\ -2 & 0 \end{bmatrix} - \begin{bmatrix} -1.5615 & 0 \\ 0 & -1.5615 \end{bmatrix} \begin{bmatrix} \phi_{10} \\ \phi_{11} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2.5615 & -2 \\ -2 & -1.5615 \end{bmatrix} \begin{bmatrix} \phi_{10} \\ \phi_{11} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

Using row detection method, we have found ϕ_{11} to be a free variable. So choose, we the value of ϕ_{11} as 1.

$$2.5615\phi_{10} - 2\phi_{11} = 0$$

$$\phi_{10} = \frac{2}{2.5615} = 0.7808$$

The eigen vector $\phi_1 = \begin{bmatrix} 0.7808 \\ 1 \end{bmatrix}$

Step 5 Normalisation of the eigen vectors

The normalisation formula to normalise the eigen vector ϕ_0 is given below

$$\frac{\phi_0}{\|\phi_0\|} = \frac{1}{\sqrt{\phi_{00}^2 + \phi_{01}^2}} \begin{bmatrix} \phi_{00} \\ \phi_{01} \end{bmatrix}$$

$$\frac{\phi_0}{\|\phi_0\|} = \frac{1}{\sqrt{(-1.2808)^2 + 1^2}} \begin{bmatrix} -1.2808 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.7882 \\ 0.6154 \end{bmatrix}$$

Similarly, the normalisation of the eigen vector ϕ_1 is given by

$$\frac{\phi_1}{\|\phi_1\|} = \frac{1}{\sqrt{(0.7808)^2 + 1^2}} \begin{bmatrix} 0.7808 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.6154 \\ 0.7882 \end{bmatrix}$$

Step 6 KL transformation matrix from the eigen vector of the covariance matrix

From the normalised eigen vector, we have to form the transformation matrix.

$$T = \begin{bmatrix} -0.7882 & 0.6154 \\ 0.6154 & 0.7882 \end{bmatrix}$$

Now we have to check, which is orthogonal (i.e.,) $TT^T = TT^{-1} = I$

$$TT^T = \begin{bmatrix} -0.7882 & 0.6154 \\ 0.6154 & 0.7882 \end{bmatrix} \begin{bmatrix} -0.7882 & 0.6154 \\ 0.6154 & 0.7882 \end{bmatrix} = \begin{bmatrix} 0.9999 & 0 \\ 0 & 0.9999 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Step 7 KL transformation of the input matrix

To find the KL transform of the input matrix, the formula used is $Y = T[x]$.

$$Y_0 = T[x_0] = \begin{bmatrix} -0.7882 & 0.6154 \\ 0.6154 & 0.7882 \end{bmatrix} \begin{bmatrix} 4 \\ -1 \end{bmatrix} = \begin{bmatrix} -3.7682 \\ 1.6734 \end{bmatrix}$$

$$Y_1 = T[x_1] = \begin{bmatrix} -0.7882 & 0.6154 \\ 0.6154 & 0.7882 \end{bmatrix} \begin{bmatrix} -2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3.4226 \\ 1.1338 \end{bmatrix}$$

The final transform matrix

$$Y = \begin{bmatrix} -3.7682 & 3.4226 \\ 1.6734 & 1.1338 \end{bmatrix}.$$

Step 8 Reconstruction of input values from the transformed coefficients

From the transform matrix, we have to reconstruct value of the given sample matrix X using the formula $X = T^T Y$.

$$x_0 = T^T Y_0 = \begin{bmatrix} -0.7882 & 0.6154 \\ 0.6154 & 0.7882 \end{bmatrix} \begin{bmatrix} -3.7682 \\ 1.6734 \end{bmatrix} = \begin{bmatrix} 3.9998 \\ -1 \end{bmatrix}$$

$$x_1 = T^T Y_1 = \begin{bmatrix} -0.7882 & 0.6154 \\ 0.6154 & 0.7882 \end{bmatrix} \begin{bmatrix} 3.4226 \\ 1.1338 \end{bmatrix} = \begin{bmatrix} -1.9999 \\ 2.9999 \end{bmatrix}$$

$$X = [x_0 \ x_1] = \begin{bmatrix} 4 & -2 \\ -1 & 3 \end{bmatrix}.$$

Example 4.12 Compute the basis of the KL transform for the input data $x_1 = (4, 4, 5)^T$, $x_2 = (3, 2, 5)^T$, $x_3 = (5, 7, 6)^T$ and $x_4 = (6, 7, 7)^T$.

Solution

Step 1 Calculation of Mean of the input vectors

$$m_x = \frac{1}{4} \times \begin{bmatrix} 18 \\ 20 \\ 23 \end{bmatrix} = \begin{bmatrix} 4.5 \\ 5.0 \\ 5.75 \end{bmatrix}$$

Step 2 Computation of autocorrelation matrix

The expression for the autocorrelation matrix is given by

$$R_x = \frac{1}{M} \sum_{i=1}^n x_i x_i^T - m_x m_x^T \quad (4.93)$$

Where n is the number of input vectors.

$$m_x m_x^T = \begin{bmatrix} 4.5 \\ 5 \\ 5.75 \end{bmatrix} \times \begin{bmatrix} 4.5 & 5 & 5.75 \end{bmatrix} = \begin{bmatrix} 20.25 & 22.5 & 25.875 \\ 22.5 & 25 & 28.75 \\ 25.875 & 28.75 & 33.0625 \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1.25 & 2.25 & 0.88 \\ 2.25 & 4.5 & 1.5 \\ 0.88 & 1.5 & 0.69 \end{bmatrix}$$

The eigen values of R_x are $\lambda_1 = 6.1963$, $\lambda_2 = 0.2147$ and $\lambda_3 = 0.0264$.

The corresponding Eigen vectors are

$$u_1 = \begin{bmatrix} 0.4385 \\ 0.8471 \\ 0.3003 \end{bmatrix}, u_2 = \begin{bmatrix} 0.4460 \\ -0.4952 \\ 0.7456 \end{bmatrix} \text{ and } u_3 = \begin{bmatrix} -0.7803 \\ 0.1929 \\ 0.5949 \end{bmatrix}.$$

From this, the KL transform basis is given by

$$T = \begin{bmatrix} 0.4385 & 0.8471 & 0.3003 \\ 0.4460 & -0.4952 & 0.7456 \\ -0.7803 & 0.1929 & 0.5949 \end{bmatrix}.$$

Since the rows of T are orthonormal vectors, the inverse transform is just the transpose $T^{-1} = T^T$.

4.14 SINGULAR VALUE DECOMPOSITION

The Singular Value Decomposition (SVD) for square matrices was discovered independently by Beltrami in 1873 and Jordan in 1874, and extended to rectangular matrices by Eckart and Young in the 1930s.

The singular value decomposition of a rectangular matrix A is a decomposition of the form

$$A = UDV^T \quad (4.94)$$

Where A is an $m \times n$ matrix

U, V are orthonormal matrices.

D is a diagonal matrix comprised of singular values of A

The singular values $\sigma_1 \geq \sigma_2 \dots \geq \sigma_n \geq 0$ appear in descending order along the main diagonal of D . The singular values are obtained by taking the square root of the eigen values of AA^T and A^TA .

Equation (4.94) can be written as

$$A = UDV^T \quad (4.95)$$

$$A = [u_1, u_2, \dots, u_n] \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix} \quad (4.96)$$

The relation between SVD and eigen values are given below:

$$A = UDV^T$$

$$\text{Now } AA^T = UDV^T(UDV^T)^T = UDV^TVDU^T = UD^2U^T$$

Also,

$$A^TA = (UDV^T)^TUDV^T = VDU^TUDV^T = VD^2V^T$$

Thus, U and V are calculated as the eigen vectors of AA^T and A^TA respectively. The square root of the eigen values are the singular values along the diagonal of the matrix D . If the matrix A is real, then the singular values are always real numbers, and U and V are also real.

4.14.1 Properties of SVD

The important properties of SVD are given below

1. The singular values $\sigma_1, \sigma_2, \dots, \sigma_n$ are unique; however the matrices U and V are not unique.
2. $A^T A = (UDV^T)^T UDV^T = VD^T U^T UDV^T = VD^T DV^T$; hence V diagonalises $A^T A$. It follows that the matrix V can be computed through the eigen vector of $A^T A$.
3. The matrix U can be computed through the eigen vectors of AA^T .
4. The rank of the matrix A is equal to the number of its non-zero singular values.

4.14.2 Applications of SVD in Image Processing

The applications of SVD in image processing are summarised below:

1. SVD approach can be used in image compression.
2. SVD can be used in face recognition.
3. SVD can be used in watermarking.
4. SVD can be used for texture classification.

Example 4.13 Find a singular value decomposition of $A = \begin{bmatrix} 1 & -2 & 3 \\ 3 & 2 & -1 \end{bmatrix}$.

Solution

Step 1 Conversion of rectangular matrix into square matrix

$$X = A^T A = \begin{bmatrix} 1 & 3 \\ -2 & 2 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 3 \\ 3 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 10 & 4 & 0 \\ 4 & 8 & -8 \\ 0 & -8 & 10 \end{bmatrix}$$

Step 2 Find the eigen values of the matrix X

Using the characteristic equation, we have to find the eigen value

$$|X - \lambda I| = 0$$

$$\det \left(\begin{bmatrix} 10 & 4 & 0 \\ 4 & 8 & -8 \\ 0 & -8 & 10 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} \right) = 0$$

$$\det \begin{pmatrix} 10-\lambda & 4 & 0 \\ 4 & 8-\lambda & -8 \\ 0 & -8 & 10-\lambda \end{pmatrix} = 0$$

$$(10-\lambda)[(8-\lambda)(10-\lambda)-64]-4[4(10-\lambda)] = 0$$

$$(10-\lambda)[80-8\lambda-10\lambda+\lambda^2-64]-4[40-4\lambda] = 0$$

$$800-80\lambda-100\lambda+10\lambda^2-640-80\lambda+8\lambda^2+10\lambda^2-\lambda^3+64\lambda-160+16\lambda = 0$$

$$-\lambda^3+28\lambda^2-180\lambda = 0$$

$$\lambda^3-28\lambda^2+180\lambda = 0$$

$$\lambda(\lambda^2-28\lambda+180) = 0$$

$$\lambda = 0; \lambda(\lambda-10)-18(\lambda-10) = 0$$

$$\lambda = 0; \lambda = 10; \lambda = 18$$

Step 3 Formation of eigen vector

Now arrange the eigen values in descending order, $\lambda = 18, \lambda = 10$ and $\lambda = 0$.

The first eigen vector is formed from the first eigen value i.e. $\lambda = 18$.

So the value of $\lambda = 18$,

$$[X - \lambda I] \times [V_0] = 0$$

where V_0 is the eigen vector in the form of $\begin{bmatrix} v_{01} \\ v_{02} \\ v_{03} \end{bmatrix}$.

$$\begin{bmatrix} 10 & 4 & 0 \\ 4 & 8 & -8 \\ 0 & -8 & 10 \end{bmatrix} - \begin{bmatrix} 18 & 0 & 0 \\ 0 & 18 & 0 \\ 0 & 0 & 18 \end{bmatrix} \times \begin{bmatrix} v_{01} \\ v_{02} \\ v_{03} \end{bmatrix} = 0$$

$$\begin{bmatrix} 10-18 & 4 & 0 \\ 4 & 8-18 & -8 \\ 0 & -8 & 10-18 \end{bmatrix} \times \begin{bmatrix} v_{01} \\ v_{02} \\ v_{03} \end{bmatrix} = 0$$

$$\begin{bmatrix} -8 & 4 & 0 \\ 4 & -10 & -8 \\ 0 & -8 & -8 \end{bmatrix} \times \begin{bmatrix} v_{01} \\ v_{02} \\ v_{03} \end{bmatrix} = 0$$

Using row-reduction method,

$$\begin{bmatrix} -8 & 4 & 0 & 0 \\ 4 & -10 & -8 & 0 \\ 0 & -8 & -8 & 0 \end{bmatrix}$$

In the second row, the first column element is to be reduced to zero.

$$\begin{bmatrix} -8 & 4 & 0 & 0 \\ 0 & -8 & -8 & 0 \\ 0 & -8 & -8 & 0 \end{bmatrix} R_2 \rightarrow R_2 + \frac{R_1}{2}$$

In the third row, the second element is reduced to zero.

$$\begin{bmatrix} -8 & 4 & 0 & 0 \\ 0 & -8 & -8 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} R_3 \rightarrow R_3 - R_2$$

From the reduced matrix, the last row values are all zero, and the value of v_{03} becomes a free variable. Hence, we assume the value of $v_{03} = 1$.

Now forming the equation from the final matrix, we get

$$\begin{aligned} -8v_{01} + 4v_{02} &= 0 \\ -8v_{02} - 8v_{03} &= 0 \end{aligned}$$

Solving the above two equations we get the values of v_{01} and v_{02} :

$$v_{01} = \frac{-1}{2}; v_{02} = -1 \text{ and } v_{03} = 1$$

$$V_0 = \begin{bmatrix} -\frac{1}{2} \\ -1 \\ 1 \end{bmatrix}$$

Similarly, when $\lambda = 10$,

$$[X - \lambda I] \times [V_0] = 0$$

Where V_1 is the eigen vector in the form of $\begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \end{bmatrix}$.

$$\left(\begin{bmatrix} 10 & 4 & 0 \\ 4 & 8 & -8 \\ 0 & -8 & 10 \end{bmatrix} - \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \right) \times \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \end{bmatrix} = 0$$

$$\left(\begin{bmatrix} 10-10 & 4 & 0 \\ 4 & 8-10 & -8 \\ 0 & -8 & 10-10 \end{bmatrix} \right) \times \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \end{bmatrix} = 0$$

$$\left(\begin{bmatrix} 0 & 4 & 0 \\ 4 & -2 & -8 \\ 0 & -8 & 0 \end{bmatrix} \right) \times \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \end{bmatrix} = 0$$

Using row-reduction method,

$$\left[\begin{array}{cccc} 0 & 4 & 0 & 0 \\ 4 & -2 & -8 & 0 \\ 0 & -8 & 0 & 0 \end{array} \right]$$

Here, in the first row, the first element is zero, and interchanging the row 1 into row 2 and vice versa, we get

$$\left[\begin{array}{cccc} 4 & -2 & -8 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & -8 & 0 & 0 \end{array} \right] R_1 \leftrightarrow R_2$$

In the third row, the second element is reduced to zero.

$$\left[\begin{array}{cccc} 4 & -2 & -8 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] R_3 \rightarrow R_3 + 2R_2$$

From the reduced matrix, the last row values are all zero, and the value of v_{03} becomes a free variable, Hence we assume the value of $v_{03} = 1$.

Now, forming the equation from the final matrix, we get

$$\begin{aligned} 4v_{11} - 2v_{12} - 8v_{13} &= 0 \\ 4v_{12} &= 0 \end{aligned}$$

Solving the above two equations, we get the values of v_{01} and v_{02} :

$$v_{11} = 2; v_{12} = 0 \text{ and } v_{13} = 1$$

$$V_1 = \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

Similarly, when $\lambda = 0$,

$$[X - \lambda I] \times s[V_0] = 0,$$

where V_2 is the eigen vector in the form of $\begin{bmatrix} v_{21} \\ v_{22} \\ v_{23} \end{bmatrix}$.

$$\left(\begin{bmatrix} 10 & 4 & 0 \\ 4 & 8 & -8 \\ 0 & -8 & 10 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \times \begin{bmatrix} v_{21} \\ v_{22} \\ v_{23} \end{bmatrix} = 0$$

$$\left(\begin{bmatrix} 10 & 4 & 0 \\ 4 & 8 & -8 \\ 0 & -8 & 10 \end{bmatrix} \right) \times \begin{bmatrix} v_{21} \\ v_{22} \\ v_{23} \end{bmatrix} = 0$$

Using row-reduction method,

$$\begin{bmatrix} 10 & 4 & 0 & 0 \\ 4 & 8 & -8 & 0 \\ 0 & -8 & 10 & 0 \end{bmatrix}$$

In the second row, the first column element is to be reduced to zero.

$$\begin{bmatrix} 10 & 4 & 0 & 0 \\ 0 & \frac{32}{5} & -8 & 0 \\ 0 & -8 & 10 & 0 \end{bmatrix} R_2 \rightarrow R_2 - \left(\frac{4}{10} \right) R_1$$

In the third row, the second element is reduced to zero.

$$\begin{bmatrix} 10 & 4 & 0 & 0 \\ 0 & \frac{32}{5} & -8 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} R_3 \rightarrow R_3 - \left(\frac{5}{32} \right) R_2$$

From the reduced matrix, the last row values are all zero, and the value of v_{23} becomes a free variable. Hence we assume the value of $v_{23} = 1$.

Now forming the equation from the final matrix, we get

$$\begin{aligned} 10v_{21} + 4v_{22} &= 0 \\ \left(\frac{32}{5}\right)v_{22} - 8v_{23} &= 0 \end{aligned}$$

Solving the above two equations we get the values of v_{01} and v_{02} ,

$$v_{21} = \frac{-1}{2}; v_{22} = \frac{5}{4} \text{ and } v_{23} = 1.$$

$$V_2 = \begin{bmatrix} -1/2 \\ 5/4 \\ 1 \end{bmatrix}$$

Step 4 Normalisation of the eigen vector

$$\text{Normalisation formula is } \|V_i\| = \frac{1}{\sqrt{V_{i1}^2 + V_{i2}^2 + V_{i3}^2}} V_i$$

To normalise the eigen vector, V_0 is

$$V_0 = \|V_0\| = \frac{1}{\sqrt{(-1/2)^2 + (-1)^2 + 1}} \begin{bmatrix} -1/2 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1/3 \\ -2/3 \\ 2/3 \end{bmatrix}$$

Similarly, for the second and third eigen vector as

$$V_1 = \|V_1\| = \frac{1}{\sqrt{(2)^2 + 0 + 1}} \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2/\sqrt{5} \\ 0 \\ 1/\sqrt{5} \end{bmatrix}$$

$$V_2 = \|V_2\| = \frac{1}{\sqrt{(-1/2)^2 + (5/4)^2 + 1}} \begin{bmatrix} -1/2 \\ 5/4 \\ 1 \end{bmatrix} = \begin{bmatrix} -2/3\sqrt{5} \\ \sqrt{5}/3 \\ 4/3\sqrt{5} \end{bmatrix}$$

Step 5 *Generation of U vector*

The formula for the U vector generation is, $U_i = \frac{1}{\|AV_i\|} AV_i$

Using the above formula, we have to find the vector U_0 as

$$U_0 = \frac{1}{\|AV_0\|} AV_0$$

First, find AV_0 .

$$AV_0 = \begin{bmatrix} 1 & -2 & 3 \\ 3 & 2 & -1 \end{bmatrix} \begin{bmatrix} -1/3 \\ -2/3 \\ 2/3 \end{bmatrix} = \begin{bmatrix} \frac{-1+4+6}{3} \\ \frac{-3-4-2}{3} \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \end{bmatrix}$$

$$\|AV_0\| = \sqrt{3^2 + (-3)^2} = \sqrt{18}$$

$$U_0 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

Similarly, find the vector U_1 .

$$U_1 = \frac{1}{\|AV_1\|} AV_1$$

First, find the value of AV_1 .

$$AV_1 = \begin{bmatrix} 1 & -2 & 3 \\ 3 & 2 & -1 \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} \\ 0 \\ 1/\sqrt{5} \end{bmatrix} = \begin{bmatrix} \frac{2+0+3}{\sqrt{5}} \\ \frac{6+0-1}{\sqrt{5}} \end{bmatrix} = \begin{bmatrix} \sqrt{5} \\ \sqrt{5} \end{bmatrix}$$

$$\|AV_1\| = \sqrt{(\sqrt{5})^2 + (\sqrt{5})^2} = \sqrt{10}$$

$$U_0 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

Similarly, find the vector U_2 .

$$U_2 = \frac{1}{\|AV_2\|} AV_2$$

The value of AV_2 is given by

$$AV_2 = \begin{bmatrix} 1 & -2 & 3 \\ 3 & 2 & -1 \end{bmatrix} \begin{bmatrix} -2/3\sqrt{5} \\ \sqrt{5}/3 \\ 4/3\sqrt{5} \end{bmatrix} = \begin{bmatrix} -2-10+12 \\ 3\sqrt{5} \\ -6+10-4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\|AV_2\| = \sqrt{0^2 + 0^2} = 0$$

$$U_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{The } U \text{ vector is } U = [U_0, U_1, U_2]; U = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \end{bmatrix}$$

Step 6 Find the \sum vector

$$\text{The } \sum = [D \ 0],$$

where D is the square root of the eigen value and its value is singular.

$$D = \begin{bmatrix} \sqrt{18} & 0 \\ 0 & \sqrt{10} \end{bmatrix}$$

$$\sum = \begin{bmatrix} \sqrt{18} & 0 & 0 \\ 0 & \sqrt{10} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Step 7 Reconstruction of matrix A , from the matrix as U , \sum and V .

The formula for reconstruction of the matrix $A = U \sum V^T$,

$$\begin{aligned}
 A = U \sum V^T &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix} \times \begin{bmatrix} \sqrt{18} & 0 & 0 \\ 0 & \sqrt{10} & 0 \\ 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} -\frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \\ \frac{2}{\sqrt{5}} & 0 & \frac{1}{\sqrt{5}} \\ -\frac{2}{3\sqrt{5}} & \frac{\sqrt{5}}{3} & \frac{4}{3\sqrt{5}} \end{bmatrix} \\
 &= \begin{bmatrix} \sqrt{9} & \sqrt{5} & 0 \\ -\sqrt{9} & \sqrt{5} & 0 \end{bmatrix} \times \begin{bmatrix} -\frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \\ \frac{2}{\sqrt{5}} & 0 & \frac{1}{\sqrt{5}} \\ -\frac{2}{3\sqrt{5}} & \frac{\sqrt{5}}{3} & \frac{4}{3\sqrt{5}} \end{bmatrix} = \begin{bmatrix} 1 & -2 & 3 \\ 3 & 2 & -1 \end{bmatrix}
 \end{aligned}$$

MATLAB Example 1 Write a MATLAB code to plot the singular values of a low-frequency and high-frequency image.

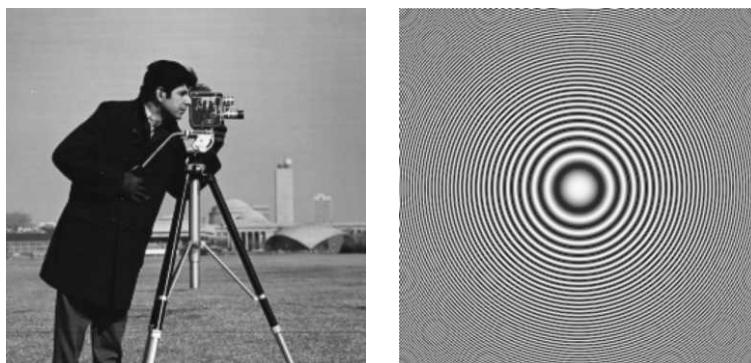
Solution The ‘cameraman’ is taken as the low-frequency image and the ‘zone-plate’ is taken as the high-frequency image. The MATLAB code which plots the singular values of the two images are shown in Fig. 4.28 and the corresponding results are shown in Fig. 4.29.

```

a = imread('cameraman.tif');
b = imread('zoneplate.png');
[p q r] = svd(double(a));
[p1 q1 r1] = svd(double(b));
semilogy(diag(q)), title('logplot of singular value of cameraman');
figure, semilogy(diag(q1)), title('logplot of singular value of zoneplate');

```

Fig. 4.28 MATLAB code to plot the singular values of the images



(a) Cameraman image

(b) Zone-plate image

Fig. 4.29 Test images

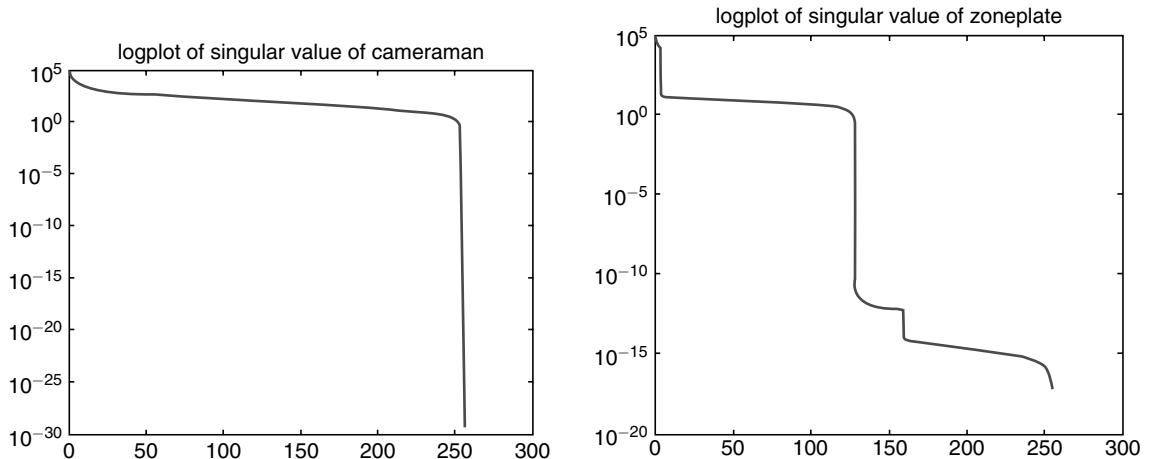


Fig. 4.30 Singular value plot of test images

The logarithm of the singular values of the cameraman image and the zone-plate image are shown in Fig. 4.30. From the figure it is obvious that the singular values of the cameraman image decrease smoothly, whereas in the zone-plate image, we can observe some fluctuation.

4.14.3 Image Compression using SVD

When an image is transformed using SVD, it is not compressed, but the data takes a form in which the first singular value has a great amount of the image information. This implies that few singular values can be used to represent the image with little differences from the original image.

When SVD is applied to the input image A , it can be written as

$$A = UDV^T = \sum_{i=1}^r \sigma_i u_i v_i^T \quad (4.97)$$

Thus the input image A can be represented by the outer product expansion

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_r u_r v_r^T \quad (4.98)$$

When compressing the image, the sum is not performed to the very last singular values, and the singular values with smaller values are dropped. After truncating the smaller singular values, the reconstructed image using the first k larger singular values is given by

$$A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_k u_k v_k^T \quad (4.99)$$

The total storage for the reconstructed image A_k is $k(m + n + 1)$. The integer k can be chosen to be less than n , and the digital image corresponding to A_k still will be closer to the original image.

One of the performance measures of image compression is *compression ratio*. The compression ratio using SVD is given below:

$$\text{Compression ratio} = \frac{m \times n}{k(m + n + 1)} \quad (4.100)$$

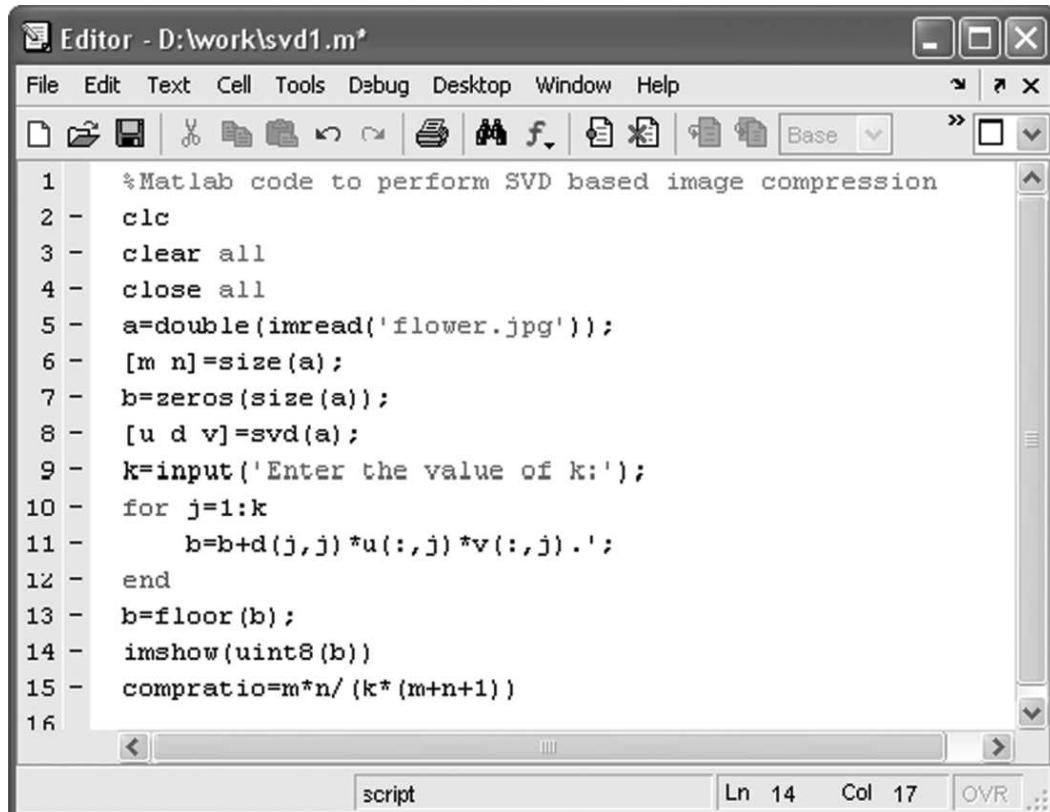
MATLAB Example 2 *Read an image and perform singular value decomposition. Retain only k largest singular values and reconstruct the image. The value k should be user-specific. Observe the quality of the reconstructed image with change in the value of k . Also, compute the Compression ratio.*

Solution The MATLAB code to perform the specified task is shown in Fig. 4.31.

From the code, it is obvious that only k (singular) values are retained to get the reconstructed image.

The reconstructed images for different values of k are given below:

From Fig. 4.32, it is clear that as the value of k increases, the visual quality of the reconstructed image approaches the original image. At the same time, the compression ratio decreases as the value of k increases. The compression ratio obtained for different values of k are given in Table 4.6.

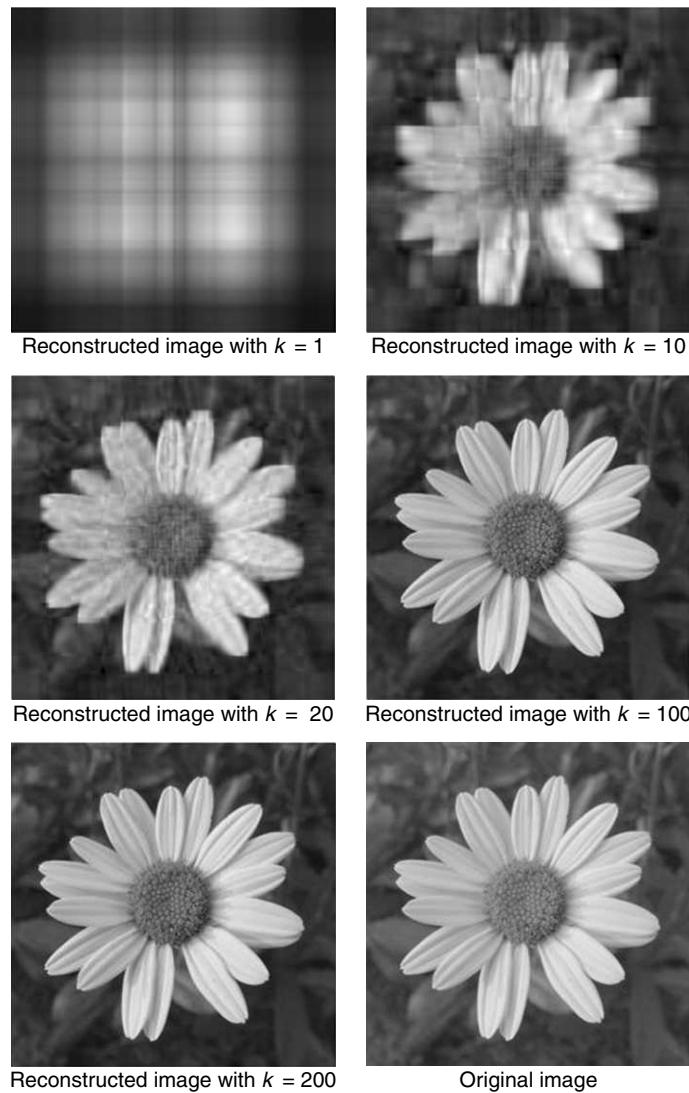


```

1 %Matlab code to perform SVD based image compression
2 clc
3 clear all
4 close all
5 a=double(imread('flower.jpg'));
6 [m n]=size(a);
7 b=zeros(size(a));
8 [u d v]=svd(a);
9 k=input('Enter the value of k:');
10 for j=1:k
11     b=b+d(j,j)*u(:,j)*v(:,j).';
12 end
13 b=floor(b);
14 imshow(uint8(b))
15 compratio=m*n/ (k*(m+n+1))
16

```

Fig. 4.31 MATLAB code to perform SVD-based image compression

**Fig. 4.32** Reconstructed images**Table 4.6** Compression ration against k

S. No	Value of k	Compression ratio
1	1	124.75
2	10	12.275
3	20	6.2375
4	100	1.2475
5	200	0.6238

4.15 RADON TRANSFORM

The Radon transform is named after the Austrian mathematician Johann Karl August Radon. Radon transform is used to compute the projection of an object in its image. Applying the Radon transform to an image $f(m, n)$ for a given set of angles can be thought of as computing the projection of the image along the given angles. The resulting projection is the sum of the intensities of the pixels in each direction, i.e., a line integral. The Radon transform is a mapping from the Cartesian rectangular coordinates to a distance and angle (s, θ) , known as polar coordinates. Figure 4.33 illustrates the 1D projection $g(s, \theta)$ of the 2D function $f(m, n)$. The Radon transform of a function $f(m, n)$ is given by $g(s, \theta)$ where

$$g(s, \theta) \stackrel{\Delta}{=} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(m, n) \delta(m \cos \theta + n \sin \theta - s) dm dn, -\infty < s < \infty, 0 \leq \theta < \pi \quad (4.101)$$

The angle of projection is illustrated clearly in Fig. 4.34. Also, Figs. 4.34 and 4.35 are derived from Fig. 4.33.

From Fig. 4.34, we can write

$$s = m \cos \theta + n \sin \theta \quad (4.102)$$

From Fig. 4.36, we can write

$$\begin{aligned} u &= -m \cos(90 - \theta) + n \sin(90 - \theta) \\ u &= -m \sin \theta + n \cos \theta \end{aligned} \quad (4.103)$$

Multiplying Eq. (4.103) on both sides by $\sin \theta$, we get

$$s \sin \theta = m \sin \theta \cos \theta + n \sin^2 \theta \quad (4.104)$$

Multiplying Eq. (4.103) on both sides by $\cos \theta$, we get

$$s \cos \theta = m \cos^2 \theta + n \sin \theta \cos \theta \quad (4.105)$$

Multiplying Eq. (4.103) on both sides by $\sin \theta$, we get

$$u \sin \theta = -m \sin^2 \theta + n \sin \theta \cos \theta \quad (4.106)$$

Multiplying Eq. (4.103) on both sides by $\cos \theta$, we get

$$u \cos \theta = -m \cos \theta \sin \theta + n \cos^2 \theta \quad (4.107)$$

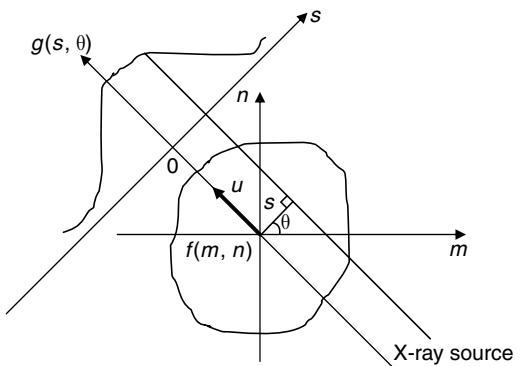


Fig. 4.33 Schematic illustration of radon transform in tomography

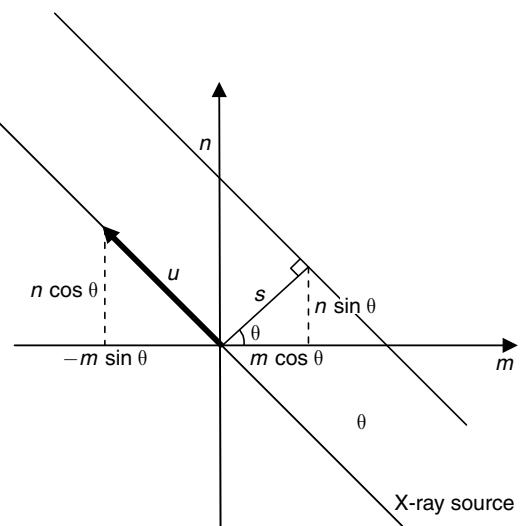


Fig. 4.34 Angle of projection

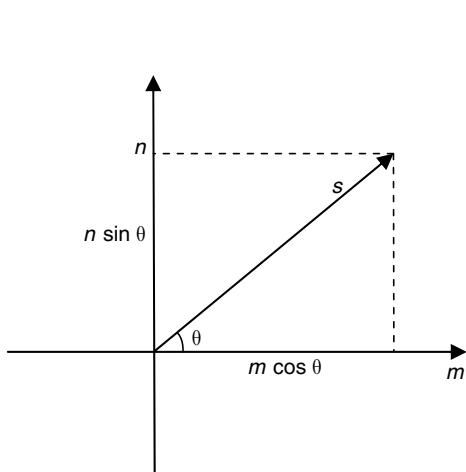


Fig. 4.35 Projection of the distance s into m and n planes

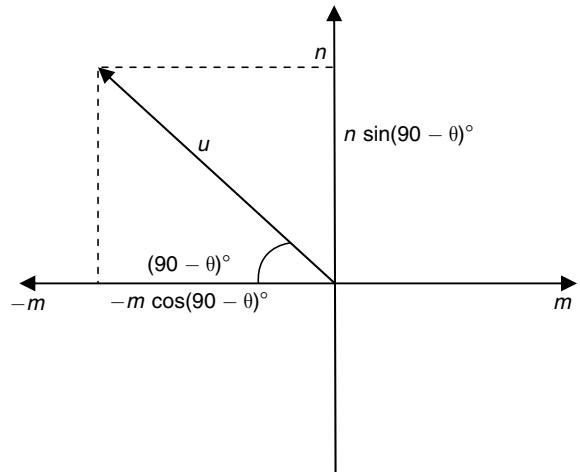


Fig. 4.36 Projection of vector u in m and n planes

Now adding Eq. (4.104) and (4.107), we get

$$s \sin \theta + u \cos \theta = m \sin \theta \cos \theta + n \sin^2 \theta - m \sin \theta \cos \theta + n \cos^2 \theta \quad (4.108)$$

Upon simplification, we get

$$s \sin \theta + u \cos \theta = n \quad (4.109)$$

Now subtract Eq. (4.106) from (4.105),

$$s \cos \theta - u \sin \theta = m \cos^2 \theta + n \sin \theta \cos \theta + m \sin^2 \theta - n \sin \theta \cos \theta \quad (4.110)$$

Upon simplification, we get

$$s \cos \theta - u \sin \theta = m \quad (4.111)$$

Substitute the values of m and n from Eqs. (4.109) and (4.111) in Eq. (4.110), we get Eq. (4.113)

$$g(s, \theta) = \int_L f(m, n) du \quad (4.112)$$

So that,

$$g(s, \theta) = \int_0^L f(s \cos \theta - u \sin \theta, s \sin \theta + u \cos \theta) du \quad (4.113)$$

Hence the Radon transform,

$$\mathcal{R}f = \int_0^L f(s \cos \theta - u \sin \theta, s \sin \theta + u \cos \theta) du = g(s, \theta) \quad (4.114)$$

The Radon operator maps the spatial domain (m, n) to the projection domain (s, θ) , in which each point corresponds to a straight line in the spatial domain. Conversely, each point in the spatial domain becomes a sine curve in the projection domain.

4.15.1 The Projection-Slice Theorem

The Projection-Slice theorem is the mathematical basis for computer tomography. The word ‘tomography’ comes from the Greek word *tomo* which means sectional, and *graphy*, which means representation. A tomographic image is a cross-sectional image of an object.

The Fourier transform of $g(s, \theta)$ is given as $G(\omega, \theta)$ where

$$G(\omega, \theta) = \int_{-\infty}^{\infty} g(s, \theta) e^{-j2\pi\omega s} ds \quad (4.115)$$

but

$$g(s, \theta) = \int_{-\infty}^{\infty} f(s \cos \theta - u \sin \theta, s \sin \theta + u \cos \theta) du \quad (4.116)$$

Substituting Eq. (4.116) in Eq. (4.115) we get

$$G(\omega, \theta) = \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} f(s \cos \theta - u \sin \theta, s \sin \theta + u \cos \theta) du \right) e^{-j2\pi\omega s} ds \quad (4.117)$$

But from Eqs (4.109) and (4.111), we have

$s \sin \theta + u \cos \theta = n s \cos \theta - u \sin \theta = m$. Substituting this in Eq. (4.117), we get

$$G(\omega, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(m, n) e^{-j2\pi\omega(m \cos \theta + n \sin \theta)} dm dn \quad (4.118)$$

$$G(\omega, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(m, n) e^{-j2\pi\omega m \cos \theta} e^{-j2\pi\omega n \sin \theta} dm dn \quad (4.119)$$

$$G(\omega, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(m, n) e^{-j2\pi(\omega \cos \theta)m} e^{-j2\pi(\omega \sin \theta)n} dm dn \quad (4.120)$$

The above equation can be written as

$$G(\omega, \theta) = F(\omega \cos \theta, \omega \sin \theta) \quad (4.121)$$

4.15.2 Inverse Radon Transform

The inverse radon transform is used to reconstruct an image from a set of projections.

$$f(m, n) = \left(\frac{1}{2\pi^2} \right) \int_0^{\pi} \int_{-\infty}^{\infty} \frac{\left(\frac{\partial}{\partial s} \right) g(s, \theta)}{m \cos \theta + n \sin \theta - s} ds d\theta \quad (4.122)$$

Proof The inverse Fourier transform is given by

$$f(m, n) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\omega_1, \omega_2) e^{j2\pi\omega_1 m} e^{j2\pi\omega_2 n} d\omega_1 d\omega_2 \quad (4.123)$$

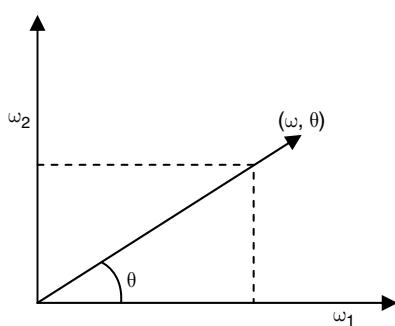


Fig. 4.37 Polar coordinates in frequency plane

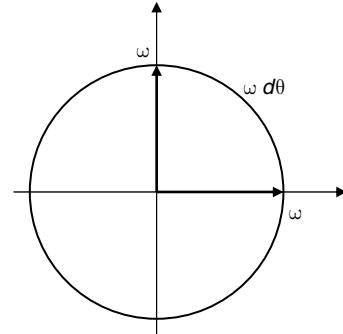


Fig. 4.38 Angle in the frequency plane

When written in polar coordinates in the frequency plane, ω_1 and ω_2 are frequencies in the m and n axes. From Fig. 4.37,

$$\omega = \sqrt{\omega_1^2 + \omega_2^2}; \theta = \tan^{-1} \left(\frac{\omega_2}{\omega_1} \right)$$

$$\omega_1 = \omega \cos \theta; \omega_2 = \omega \sin \theta$$

From Fig. 4.38, $d\omega = \omega d\theta$. Substituting $\omega_1 = \omega \cos \theta$, $\omega_2 = \omega \sin \theta$ and $d\omega = \omega d\theta$ in Eq. (4.123), we get

$$f(m, n) = \int_0^{2\pi} \int_{-\infty}^{\infty} F(\omega, \theta) e^{j2\pi\omega \cos \theta m} e^{j2\pi\omega \sin \theta n} d\omega d\theta \times \omega \quad (4.124)$$

Upon rearranging Eq. (4.124), we get

$$f(m, n) = \int_{-\infty}^{\infty} \int_0^{2\pi} \omega F(\omega, \theta) e^{j2\pi\omega \cos \theta m} e^{j2\pi\omega \sin \theta n} d\omega d\theta \quad (4.125)$$

$$f(m, n) = \int_{-\infty}^{\infty} \int_0^{\pi} |\omega| F(\omega, \theta) e^{j2\pi\omega \cos \theta m} e^{j2\pi\omega \sin \theta n} d\omega d\theta \quad (4.126)$$

$$f(m, n) = \int_{-\infty}^{\infty} \int_0^{\pi} |\omega| F(\omega, \theta) e^{j2\pi\omega(m \cos \theta + n \sin \theta)} d\omega d\theta \quad (4.127)$$

Using projection theorem,

$$f(m, n) = \int_{-\infty}^{\infty} \int_0^{\pi} |\omega| G(\omega, \theta) e^{j2\pi\omega(m \cos \theta + n \sin \theta)} d\omega d\theta \quad (4.128)$$

In the projection theorem, the Fourier transform of $G(s, \theta)$ gives the $G(\omega, \theta)$,

$$\text{Therefore, } G(\omega, \theta) = \int_{-\infty}^{\infty} G(s, \theta) e^{-j2\pi\omega s} ds \quad (4.129)$$

And also the inverse Fourier transform of $G(\omega, \theta)$, gives $g(s, \theta)$,

$$\text{That is, } g(s, \theta) = \int_{-\infty}^{\infty} G(\omega, \theta) e^{j2\pi\omega s} d\omega \quad (4.130)$$

If the value of $\hat{g}(s, \theta)$ is given by

$$\hat{g}(s, \theta) = \int_{-\infty}^{\infty} |\omega| G(\omega, \theta) e^{j2\pi\omega s} d\omega \quad (4.131)$$

Also, the expression $\hat{g}(s, \theta)$ is given by

$$\hat{g}(s, \theta) = \text{IFT} \{ |\omega| G(\omega, \theta) \} \quad (4.132)$$

$$\hat{g}(s, \theta) = \int_0^{\pi} \left[\int_{-\infty}^{\infty} |\omega| G(\omega, \theta) e^{j2\pi\omega(m \cos \theta + n \sin \theta)} d\omega \right] d\theta \quad (4.133)$$

Rewriting the above equation,

$$\hat{g}(s, \theta) = \int_0^{\pi} \text{IFT} \{ |\omega| G(\omega, \theta) \} d\theta \quad (4.134)$$

The above equation can be rewritten as

$$\hat{g}(s, \theta) = \int_0^{\pi} \hat{g}(s, \theta) d\theta \quad (4.135)$$

$$\hat{g}(s, \theta) = \int_0^{\pi} \hat{g}(m \cos \theta + n \sin \theta, \theta) d\theta \quad (4.136)$$

Now, writing $|\omega| G(\omega, \theta)$ as $\omega G(\omega, \theta) \text{sgn}(\omega)$ and applying the convolution theorem for Eq. (4.136), we get

$$\hat{g}(s, \theta) = \text{IFT} \{ \omega G(s, \theta) \cdot \text{sgn}(\omega) \} \quad (4.137)$$

$$\hat{g}(s, \theta) = \text{IFT} \{ \omega G(s, \theta) \} \otimes \text{IFT} \{ \text{sgn}(\omega) \} \quad (4.138)$$

Using differentiation property of Fourier transform, we can write,

$$\hat{g}(s, \theta) = \frac{1}{2\pi j} \left\{ \frac{\partial g(s, \theta)}{\partial s} \right\} \otimes \left(-\frac{1}{j\pi s} \right) \quad (4.139)$$

After applying convolution, we get

$$\hat{g}(s, \theta) = \left(\frac{1}{2\pi^2} \right) \left\{ \int_{-\infty}^{\infty} \frac{\partial g(t, \theta)}{\partial t} \cdot \left(\frac{1}{(s-t)} \right) dt \right\} \quad (4.140)$$

where t is a dummy variable, and substituting the value of s into Eq. (4.140), we get

$$\hat{g}(s, \theta) = \left(\frac{1}{2\pi^2} \right) \left\{ \int_{-\infty}^{\infty} \frac{\partial g(t, \theta)}{\partial t} \cdot \left(\frac{1}{(m \cos \theta + n \sin \theta - t)} \right) dt \right\} \quad (4.141)$$

Now substituting Eq. 4.141 into Eq. 4.122, we get

$$f(m, n) = \left(\frac{1}{2\pi^2} \right) \int_0^{\pi} \left\{ \int_{-\infty}^{\infty} \frac{\partial g(s, \theta)}{\partial s} \cdot \left(\frac{1}{(m \cos \theta + n \sin \theta - s)} \right) ds \right\} d\theta \quad (4.142)$$

MATLAB Example 3 Write a MATLAB code to generate a straight line. Take the Radon transform of the straight line image you have generated and observe the output. Comment on the result.

Solution The problem can be solved step by step.

Step 1 Generation of image having a straight line.

Step 2 Computing the Radon transform of the image generated in Step 1.

Step 3 Observation of the output and conclusion

The MATLAB code which generates the image and performs Radon transform of the image generated is shown in Fig. 4.39 and the corresponding output is shown in Fig. 4.40 (Plate 2).

Conclusion From the result seen in Fig. 4.40, it is clear that each point in the transform domain (Radon transform domain) corresponds to a line in the spatial domain.

```

clc
clear all
close all
%Step 1: Generation of straight line image
a=zeros(256);
[m n]=size(a);
for i=1:256
    for j=5:256
        a(128, j)=255;
    end
end
%Step 2: Computing Radon transform
theta=0:180;
[p q]=radon(a, theta);
subplot(2, 2, 1), imshow(a), title('Input image')
subplot(2, 2, 2), imshow(p, [], 'Xdata', theta, 'Ydata', q,
'InitialMagnification', 'fit'),
title('Radon transform of input image'),
colormap(hot), colorbar

```

Fig. 4.39 MATLAB code to compute radon transform

MATLAB Example 4 *Slightly modify the MATLAB Example 3, by generating more than one line, and then take Random transform for a set of lines and see the output.*

Solution The MATLAB code and the corresponding output are shown in Fig. 4.41 and Fig. 4.42 (Plate 2) respectively.

MATLAB Example 5: Radon transform of a point image *Generate a point image and compute the Radon transform for the generated image.*

Solution The MATLAB code to generate the point image is shown in Fig. 4.43, and the corresponding output is shown in Fig. 4.44 (Plate 3).

From the result shown in Fig. 4.44, it is clear that each point in the spatial domain corresponds to a sine curve in the projection domain.

MATLAB Example 6: Radon transform of multiple point image *This exercise is an extension of Example 5, and instead of generating a single point, it generates a multiple-point input image. Compute the Radon transform of this multiple-point image and comment on the result.*

Solution The MATLAB code to generate a multi-point image is shown in Fig. 4.45, and the corresponding output is shown in Fig. 4.46 (Plate 3). From Fig. 4.45, it is obvious that each point corresponds to a sinusoidal curve in the projection domain.

```

clc
clear all
close all
a=zeros(256)
[m n]=size(a);
for i=1:m
    for j=1:n
        a(40, j)=255;
        a(80, j)=255;
        a(124, j)=255;
        a(150, j)=255;
    end
end
theta=0:180;
[p q]=radon(a, theta);
imshow(a, title('Input image'))
figure, imshow(p, [], 'Xdata', theta, 'Ydata', q, 'InitialMagnification',
'fit'),
title('Radon transform of input image'),
colormap(hot), colorbar

```

Fig. 4.41 Radon transform for a series of lines

```

clc
clear all
close all
%Generating a point image
a=zeros(256);
[m n]=size(a);
a(50, 50)=255;
%Computing Radon tranform for point image
theta=0:180;
[p q]=radon(a, theta);
subplot(2, 2, 1), imshow(a), title('Input image')
subplot(2, 2, 2), imshow(p, [], 'Xdata', theta, 'Ydata', q,
'InitialMagnification', 'fit'),
title('Radon transform of input image'),
colormap(hot), colourbar

```

Fig. 4.43 MATLAB code for the MATLAB Example 5

```

clc
clear all
close all
%Generating a multi-point image
a=zeros(256);
[m n]=size(a);
a(50, 50)=255;
a(100, 100)=255;
a(142, 142)=255;
a(225, 225)=255;
%Computing Radon tranform for point image
theta=0:180;
[p q]=radon(a, theta);
subplot(2, 2, 1), imshow(a), title('Input image')
subplot(2, 2, 2), imshow(p, [], 'Xdata', theta, 'Ydata', q,
'InitialMagnification', 'fit'),
title('Radon transform of input image'),
colormap(hot), colorbar

```

Fig. 4.45 MATLAB code for Example 6

4.16 COMPARISON OF DIFFERENT IMAGE TRANSFORMS

In this section, let us compare different image transforms based on their representation, and computation complexity. The comparison of different image transforms are given in Table 4.7.

Table 4.7 Comparison of different image transforms

S No	Transform name	Basis	Parameters to be computed	Computational complexity
1	Fourier transform	Complex exponential	Transform coefficients	$2N^2 \log_2^N$ (complex)
2	Walsh transform	Walsh basis is either +1 or -1	Transform coefficients	$2N^2 \log_2^N$ (additions)
3	Slant transform		Transform coefficients	$2N^2 \log_2^N$
4	Haar transform		Transform coefficients	$2(N-1)$
5	Cosine transform	Cosine function	Transform coefficients	$4 N^2 \log_2^{2N}$
6	Karhunen–Loeve transform	Eigen vector of the covariance matrix	Transform coefficients	N^3
7	Singular value decomposition	$U\Sigma V^T$	Basis vectors and singular values	N^3

SOLVED PROBLEMS

1. The basis image of a 2D unitary transform of size 2×2 are

$$H_1 = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, H_2 = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, H_3 = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, H_4 = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Determine the transform coefficients if the input image is $f(m, n) = \begin{bmatrix} 6 & 4 \\ 2 & 1 \end{bmatrix}$. Also, reconstruct the image using the first two largest coefficients.

First, the transform coefficients are determined by taking the inner product of the basis with the input image. Let the transform coefficients be t_1, t_2, t_3 and t_4 respectively.

$$t_1 = \langle H_1, f \rangle = \frac{1}{2} [1 \times 6 + 1 \times 4 + 1 \times 2 + 1 \times 1] = \frac{13}{2} \approx 7$$

$$t_2 = \langle H_2, f \rangle = \frac{1}{2} [1 \times 6 - 1 \times 4 + 1 \times 2 - 1 \times 1] = \frac{3}{2} \approx 2$$

$$t_3 = \langle H_3, f \rangle = \frac{1}{2} [1 \times 6 + 1 \times 4 - 1 \times 2 - 1 \times 1] = \frac{7}{2} \approx 4$$

$$t_4 = \langle H_4, f \rangle = \frac{1}{2} [1 \times 6 - 1 \times 4 - 1 \times 2 + 1 \times 1] = \frac{1}{2} \approx 1$$

The transform coefficients are computed as 7, 2, 4 and 1.

For reconstruction, the first two largest coefficients 7 and 4 are taken and the reconstructed image is obtained as

$$\hat{f}(m, n) = t_1 H_1 + t_3 H_3 = 7 \times \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \frac{4}{2} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} \frac{11}{2} & \frac{11}{2} \\ \frac{3}{2} & \frac{3}{2} \end{bmatrix} = \begin{bmatrix} 6 & 6 \\ 2 & 2 \end{bmatrix}.$$

Here $\hat{f}(m, n)$ represents the reconstructed image using the first two largest coefficients.

2. Find the $N \times N$ point DFT of the following 2D image $f(m, n)$, $0 \leq m, n \leq N$

- (a) $f(m, n) = 1, n = 0; f(m, n) = 0; n \neq 0$.
- (b) $f(m, n) = 1, m = 0; f(m, n) = 0; m \neq 0$.
- (c) $f(m, n) = 1, m = n; f(m, n) = 0; \text{otherwise}$
- (d) $f(m, n) = 1, m = N-1-n; f(m, n) = 0, \text{otherwise}$

Interpret the result obtained in the four cases mentioned above.

First let us compute the DFT of the input image $f(m, n) = 1, n = 0; f(m, n) = 0; n \neq 0$. From this definition, it is obvious that the input image is a vertical line at $n = 0$. Its DFT is given by

$$F(k, l) = \sum_{m=0}^{N-1} e^{-j2\pi \frac{mk}{N}} = \begin{cases} 1, & k = 0 \\ \frac{1 - e^{-j2\pi k}}{1 - e^{-j2\pi \frac{k}{N}}}, & k \neq 0 \end{cases}$$

From the result, it is obvious that if the input image is a vertical line at $n = 0$, its DFT is a horizontal line at $k = 0$.

Now the DFT of the input image $f(m, n) = 1, m = 0; f(m, n) = 0; m \neq 0$. is computed as

$$F(k, l) = \sum_{n=0}^{N-1} e^{-j2\pi \frac{nl}{N}} = \begin{cases} 1, & l = 0 \\ \frac{1 - e^{-j2\pi l}}{1 - e^{-j2\pi \frac{l}{N}}}, & l \neq 0 \end{cases}$$

From the above result, it is clear that if the input image is a horizontal line at $m = 0$, its DFT is a vertical line at $l = 0$.

Next the 2D DFT of the input signal $f(m, n) = 1, m = n; f(m, n) = 0, \text{otherwise}$ is computed as

$$F(k, l) = \sum_{m=0}^{N-1} e^{-j2\pi \frac{m(k+l)}{N}} = \begin{cases} 1, & ((k+l))_N = 0 \\ \frac{1 - e^{-j2\pi(k+l)}}{1 - e^{-j2\pi \frac{k+l}{N}}}, & ((k+l))_N \neq 0 \end{cases}$$

The input signal is a diagonal line in the 45-degree direction; its DFT is an anti-diagonal line in the 135-degree direction.

Finally, the 2D DFT of the input signal $f(m, n) = 1, m = N-1-n; f(m, n) = 0, \text{ otherwise}$ is computed as

$$F(k, l) = \sum_{m=0}^{N-1} e^{-j2\pi \frac{mk + (N-1-m)l}{N}} = \begin{cases} e^{j2\pi \frac{l}{N}}, & k=l \\ \frac{1-e^{-j2\pi(k-l)}}{1-e^{-j2\pi \frac{(k-l)}{N}}}, & k \neq l \end{cases}$$

From the above result we can infer that if the input image is a diagonal line in the 135-degree direction, the magnitude of its DFT is a diagonal line in the 45-degree direction.

From the above results, it is clear that if the input image rotates, its DFT also rotates. Moreover, when an image contains an edge in one direction, its DFT contains an edge in its orthogonal direction.

3. Compute the 2D Haar transform of the signal $f(m, n) = \begin{bmatrix} 4 & -1 \\ 2 & 3 \end{bmatrix}$

The 2D Haar transform of the signal $f(m, n)$ is given by $F(k, l)$ where

$$F(k, l) = H_2 \times f(m, n) \times H_2^T$$

where $H_2 = \frac{1}{\sqrt{2}} \times \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. The Haar transform of the given signal is computed as

$$F(k, l) = \frac{1}{\sqrt{2}} \times \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} 4 & -1 \\ 2 & 3 \end{bmatrix} \times \frac{1}{\sqrt{2}} \times \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$F(k, l) = \frac{1}{2} \times \begin{bmatrix} 6 & 2 \\ 2 & -4 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \times \begin{bmatrix} 8 & 4 \\ -2 & 6 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ -1 & 3 \end{bmatrix}$$

4. The discrete Fourier transform performed for the image $f(m, n)$ is given below. What will be the value of $F(0, 0)$?

$$f[m, n] = \begin{bmatrix} 0 & 1 & 2 & 1 & 4 \\ 4 & 1 & 4 & 5 & 6 \\ 1 & 2 & 1 & 0 & 4 \\ 5 & 4 & 1 & 3 & 5 \\ 4 & 2 & 4 & 5 & 6 \end{bmatrix}$$

The input image in spatial domain is given and we are asked to compute the 'dc value'. The dc value is given by

$$F(0, 0) = \frac{1}{M \times N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n)$$

In this problem, $M = N = 5$. Hence $F(0, 0) = \frac{1}{5 \times 5} \sum_{m=0}^4 \sum_{n=0}^4 f(m, n)$. It means we have to add all the pixel values and divide the result by the total number of pixel values.

$$\sum_{m=0}^4 \sum_{n=0}^4 f(m, n) = 0 + 1 + 2 + 1 + 4 + 4 + 1 + 4 + 5 + 6 + 1 + 2 + 1 + 0 + 4 + 5 + 4 + 1 + 3 + 5 + 4 + 2 + 4 + 5 + 6$$

$$\sum_{m=0}^4 \sum_{n=0}^4 f(m, n) = 75. \text{ This implies the dc value is } \frac{75}{25} = 3$$

5. Prove that if an image $f(m, n)$, $0 \leq m \leq M-1$ and $0 \leq n \leq N-1$ is multiplied by the checkerboard pattern $(-1)^{m+n}$ then its DFT is centred at $\left(\frac{M}{2}, \frac{N}{2}\right)$.

The discrete Fourier transform of the function $f(m, n)$ is given by $F(k, l)$ where

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{M} mk} e^{-j \frac{2\pi}{N} nl}$$

The DFT of $(-1)^{m+n} f(m, n)$ is given by

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} (-1)^{m+n} f(m, n) e^{-j \frac{2\pi}{N} mk} e^{-j \frac{2\pi}{N} nl}$$

But $(-1)^{m+n}$ can be written as

$$(-1)^{m+n} = e^{j\pi(m+n)}$$

Substituting the above equation in the expression of $F(k, l)$ we get

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} e^{j\pi(m+n)} f(m, n) e^{-j \frac{2\pi}{M} mk} e^{-j \frac{2\pi}{N} nl}$$

The above equation can be written as

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{M} mk} e^{j\pi m} e^{-j \frac{2\pi}{N} nl} e^{j\pi n}$$

Simplifying the above equation results in

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{M} m \left(k - \frac{M}{2}\right)} e^{-j \frac{2\pi}{N} n \left(l - \frac{N}{2}\right)}$$

This implies

$$F(k, l) = F\left(k - \frac{M}{2}, l - \frac{N}{2}\right)$$

The above result shows that DFT is centred at $\left(\frac{M}{2}, \frac{N}{2}\right)$.

6. Prove that the inverse two-dimensional Fourier transform of the two-dimensional Fourier transform of $f(m, n)$ is $f(-m, -n)$.

The definition of forward and inverse 2D DFT is given below:

$$\begin{array}{ccc} f(m, n) & \xrightarrow{\text{2D DFT}} & F(k, l) \\ F(k, l) & \xrightarrow{\text{2D IDFT}} & f(m, n) \end{array}$$

The forward 2D DFT is given by

$$F(k, l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{N} mk} e^{-j \frac{2\pi}{N} nl}$$

The inverse 2D-DFT is given by

$$f(m, n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k, l) e^{j \frac{2\pi}{N} mk} e^{j \frac{2\pi}{N} nl}$$

For simplicity, the normalisation factor is not taken into consideration.

If $f(m, n)$ is the input image, on taking 2D-DFT, the spectrum of the input image is represented by $F(k, l)$. Now the 2D-DFT of the spectrum $F(k, l)$ is taken and is given by

$$F(k', l') = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k, l) e^{-j \frac{2\pi}{N} k' k} e^{-j \frac{2\pi}{N} l' l} \quad (4.143)$$

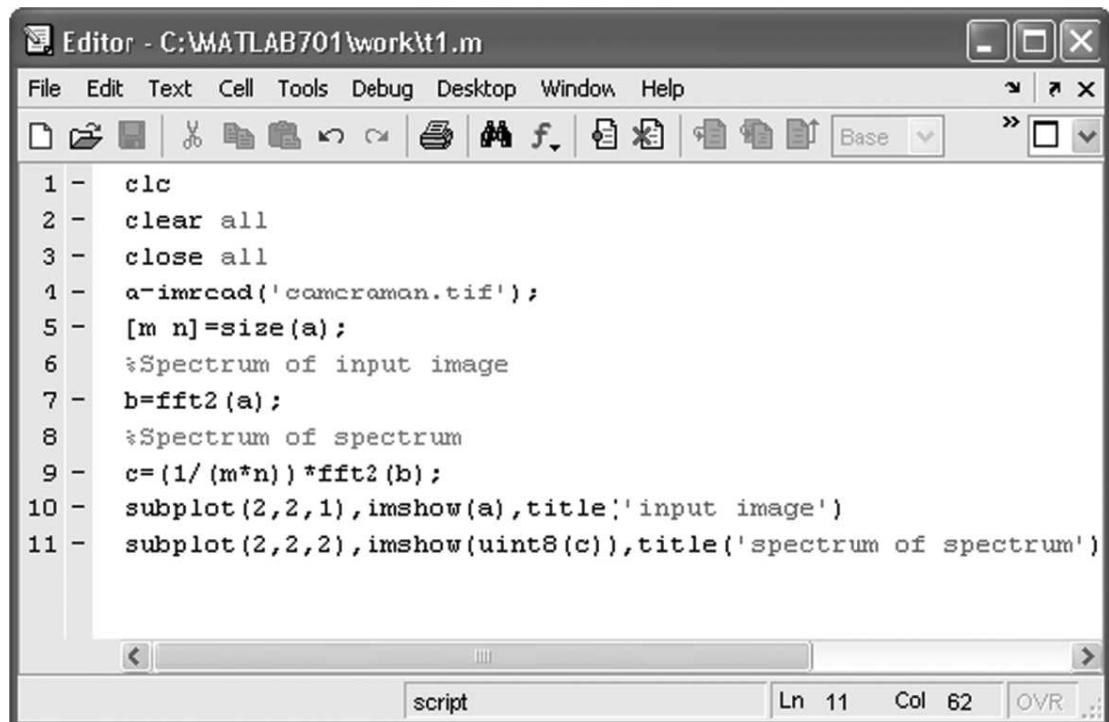
By substituting $-k' = m$ and $-l' = n$ in the above equation, we get

$$= \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k, l) e^{j \frac{2\pi}{N} k' k} e^{j \frac{2\pi}{N} l' l} \quad (4.144)$$

By Eq. (4.143) with Eq. (4.144), we get

$$F(k', l') = f(-m, -n)$$

The MATLAB command to prove the concept and the corresponding output are shown in Fig. 4.47 and Fig. 4.48 respectively.



```

1 - clc
2 - clear all
3 - close all
4 - a=imread('cameraman.tif');
5 - [m n]=size(a);
6 - %Spectrum of input image
7 - b=fft2(a);
8 - %Spectrum of spectrum
9 - c=(1/(m*n))*fft2(b);
10 - subplot(2,2,1),imshow(a),title('input image')
11 - subplot(2,2,2),imshow(uint8(c)),title('spectrum of spectrum')

```

Fig. 4.47 MATLAB code for Problem 6

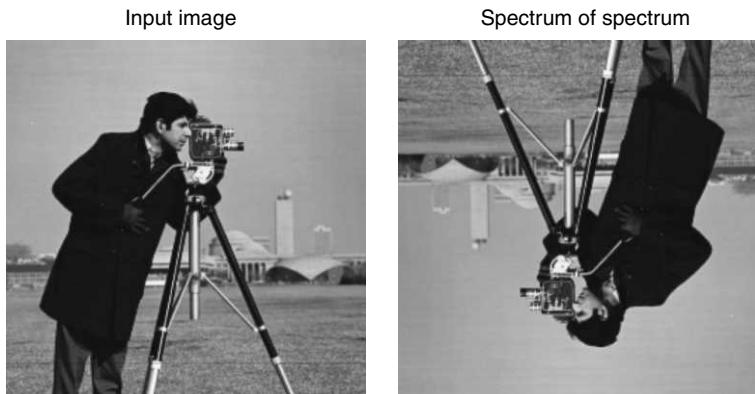


Fig. 4.48 Result of MATLAB command shown in Fig. 4.47

7. Prove that DFT diagonalises the circulant matrix.

In order to prove the fact DFT diagonalises all circulant matrices, we should know the meaning of the term circulant matrix.

Circulant matrix A square matrix A is called a circulant matrix if the entries of A that is $a(i, j)$ depend only on $(i - j) \bmod n$. An example of a circulant matrix is given below;

$$A = \begin{bmatrix} 2 & 8 & 6 & 4 \\ 4 & 2 & 8 & 6 \\ 6 & 4 & 2 & 8 \\ 8 & 6 & 4 & 2 \end{bmatrix}$$

If we observe the matrix A , it is clear that each row is a circular shift of the previous row. Hence, it is a circulant matrix. Now the 2D DFT of the matrix A has to be found out to prove that it results in a diagonal matrix.

Computation of 2D DFT of the matrix A

The 2D DFT of the matrix A is computed as

$$A(k, l) = h^* a^* h'$$

where h is the kernel and a is the input matrix. The 4×4 kernel is given by

$$h = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

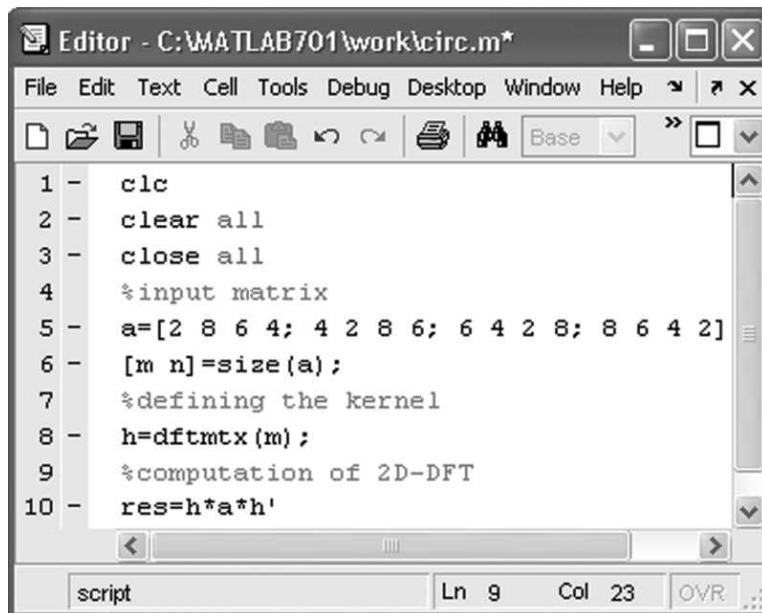
The matrix h is symmetric and hence the transpose h' of the matrix h is the same as the matrix h . Substituting the value of h in the expression of $A(k, l)$, we get

$$A(k, l) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 2 & 8 & 6 & 4 \\ 4 & 2 & 8 & 6 \\ 6 & 4 & 2 & 8 \\ 8 & 6 & 4 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

After matrix multiplication, we get

$$A(k, l) = \begin{bmatrix} 80 & 0 & 0 & 0 \\ 0 & -16 + 16j & 0 & 0 \\ 0 & 0 & -16 & 0 \\ 0 & 0 & 0 & -16 - 16j \end{bmatrix}$$

Thus, it is proved that DFT diagonalises the circulant matrix concept. The MATLAB code to solve this problem is given in Fig. 4.49.



```

Editor - C:\MATLAB701\work\circ.m*
File Edit Text Cell Tools Debug Desktop Window Help
clc
clear all
close all
%input matrix
a=[2 8 6 4; 4 2 8 6; 6 4 2 8; 8 6 4 2]
[m n]=size(a);
%defining the kernel
h=dftmtx(m);
%computation of 2D-DFT
res=h*a*h'

```

Fig. 4.49 MATLAB code for the Problem 7

8. Determine whether the matrix $A = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ is unitary or not.

The given matrix is real, and the condition for a real matrix to be unitary is

$$A^{-1} = A^T$$

Determining A^{-1}

Determinant of $A = 1$

The inverse of the matrix A is given by $A^{-1} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$.

Determining A^T

The transpose of the matrix A is given by $A^T = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

From the expressions of A^{-1} and A^T we get $A^{-1} = A^T$. Hence the matrix A is unitary.

Summary



- A transform is basically a representation of a signal. A transform changes the representation of a signal by projecting it onto a set of basis functions. The transform does not change the information content present in the signal.
- Different types of image transforms are Fourier, Walsh, Hadamard, Slant, Cosine, Sine, KL, Radon and Singular Value Decomposition.
- The transformation matrix A is unitary if it obeys the following condition

$$A^{-1} = A^T$$

- The transformation matrix A is orthogonal if it obeys the following relation

$$A^{-1} = A^T$$

- The king of all transforms is Fourier transform. The 2D Fourier transform of a signal $f(m, n)$ is given by

$$F(k, l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j \frac{2\pi}{M} ml} e^{-j \frac{2\pi}{N} mk}$$

- $e^{-j \frac{2\pi}{M} ml}, e^{-j \frac{2\pi}{N} mk}$ represents the Fourier basis. The significance of Fourier basis are the following
 1. Fourier basis is with both real and imaginary terms.
 2. Fourier basis will not lose its identity due to mathematical operations like addition, subtraction, differentiation and integration.
- A Fourier transform is represented by its magnitude and phase. The magnitude information gives how much of a certain frequency component is present and phase information gives where the frequency component is in the image.
- The number of sign changes along a column of the transform matrix is known as the sequency of that column.
- Fourier transform is separable. The separable property allows one to compute a two-dimensional DFT as two separate one-dimensional DFTs.
- Narrowing a function broadens its Fourier transform and vice versa, i.e., compression in one domain is equal to expansion in another domain.
- Spatial shift will result in phase shift (shifting property of Fourier transform)
- Rotating a function in the spatial domain will result in the rotation of the spectrum in the frequency domain.
- Convolution of two functions is similar to multiplication of their Fourier transforms.
- The Fourier transform of a Gaussian function is another Gaussian.
- A set of mutually orthonormal basis functions, with values +1 or -1 constitutes the Walsh transform kernels.
- The computation of Walsh coefficients is simple and it involves only addition and subtraction operations.
- The Hadamard matrix of order n satisfies the orthogonality condition $H_n \times H_n^T = nI_n$

- A KL transform depends on the second-order statistics of the input data. It is an optimal transform with respect to energy compaction.
- Discrete Cosine Transform (DCT) is real and orthogonal. DCT has excellent energy compaction for highly correlated data. DCT is used in JPEG standard.
- Haar functions are non-sinusoidal orthogonal functions. The Haar transform is the simplest discrete wavelet transform.
- The slant transform is real and orthogonal. Slant transforms possess good energy compaction property.
- Some of the applications of image transform include filtering, restoration, compression, enhancement and image analysis.
- Many common unitary transforms tend to pack a large fraction of signal energy into a few transform coefficients which is generally termed energy compaction.

Review Questions

1. Give four important unitary image transforms.

The most popular unitary transforms are (i) Discrete Fourier Transform, (ii) Discrete Cosine Transform, (iii) Hadamard Transform, and (iv) Haar Transform.

2. Mention two important properties of unitary transforms.

The important properties of unitary transforms are (i) energy compaction, and (ii) energy conservation.

Because of energy compaction, only few transform coefficients have large magnitude. This is basically due to the decorrelating role of unitary transforms. Energy conservation means that unitary transform preserves the L_2 norm of input vectors.

3. What is the goal of an image transform?

The goal of an image transform is to find an alternative domain where the processing is easier to perform. Image transforms are useful for fast computation of convolution and correlation.

4. Give the advantage of Walsh transform over Fourier Transform.

Fourier transform is based on trigonometric terms, whereas Walsh transform consists of a series expansion of basis functions whose values are only +1 or -1. These functions can be implemented more efficiently in a digital environment than the exponential basis functions of the Fourier transform.

5. What is the main difference between Walsh transform and Hadamard transform?

The Hadamard transform differs from the Walsh transform only in the order of the basis functions.

6. Explain why the Fourier-transform phase of an image often captures most of the intelligibility of the image?

The important visual information contained in images is in the edges and regions of high contrast. The regions of maximum and minimum intensity in an image are places at which complex exponentials at different frequencies are in phase. Therefore, that phase of the Fourier transform contains much of the information in the image, and in particular, the phase should capture information about the edges.

7. Define the property of energy compaction of a unitary transform and give reasons why this property is useful in image processing. Also compare the energy compaction of DCT with respect to DFT.

Most unitary transforms pack a large fraction of the average energy of the image into a relatively few components of the transform coefficients. But the total energy will be always preserved. Because of energy compaction, only first few coefficients will be significant. As a result the other insignificant coefficients may be neglected and replaced by zeros for the recovery of the original signal.

Thus the property of energy compaction is useful for the compression of images. The energy compaction of DCT is better than DFT.

Problems

4.1 Let the rows of the matrix shown in Fig. 4.50 represent the basis vector. Determine whether or not

- (i) the basis vectors are in sequency order
- (ii) the basis vectors are orthogonal.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}$$

Fig. 4.50 Basis vector

4.2 Prove that the 2D Fourier transform of the Gaussian function $f(m, n) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{m^2 + n^2}{2\sigma^2}\right)$ is $F(k, l) = \sigma\sqrt{2\pi} \exp\left(-\frac{4\pi^2(k^2 + l^2)}{2\beta^2}\right)$, with $\beta = \frac{1}{\sigma}$.

4.3 Determine the constants P, Q, R, S and T such that the 2D DFT of the signal $f(m, n)$ can be expressed as

$$F(k, l) = P \sum_{m=0}^Q \sum_{n=0}^R f(m, n) e^{-j(Smk + Tnl)}$$

4.4 Compute the 2D DFT of the 4×4 grayscale image $f(m, n)$ shown below.

$$f(m, n) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

4.5 Compute the Inverse 2D DFT of the transform coefficients $F(k, l)$ given below.

$$F(k, l) = \begin{bmatrix} 64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

4.6 You are given two image segments $f_1(m, n)$ and $f_2(m, n)$. Prove the additive property of Fourier transforms.

$$f_1(m, n) = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix} \text{ and } f_2(m, n) = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \end{bmatrix}.$$

That is prove

$$\text{DFT}\{f_1(m, n)\} + \text{DFT}\{f_2(m, n)\} = \text{DFT}\{f_1(m, n) + f_2(m, n)\}$$

4.7 An image matrix is given by $f(m, n) = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 2 & 1 & 1 & 2 \\ 1 & 3 & 2 & 1 \\ 2 & 1 & 2 & 1 \end{bmatrix}$. Find the 2D Hadamard transform for this image matrix.

4.8 Determine the Hadamard matrix of order $N = 8$ and obtain its inverse.

4.9 You are given an image patch $\begin{bmatrix} 12 & 4 & 2 & 6 \\ 5 & 10 & 12 & 24 \\ 6 & 8 & 10 & 12 \\ 14 & 12 & 8 & 10 \end{bmatrix}$. Compute the 2D DFT and 2D DCT for the image

patch. Then reconstruct the original image patch by neglecting the last four coefficients in 2D DFT and 2D DCT. Comment on the observed result.

4.10 Obtain the DCT matrix for $N = 4$ and verify that it obeys the orthogonality property.

4.11 Obtain the KL transform basis for the following matrix of samples:

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

4.12 Perform singular value decomposition of the matrix

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

4.13 For the following set of vectors,

$$x_1 = [1 \ 1 \ 1]^T; x_2 = [1 \ 0 \ 1]^T; x_3 = [0 \ 1 \ 0]^T \text{ and } x_4 = [1 \ 0 \ 0]^T$$

Compute the Eigen vector of the covariance matrix.

If a function is circularly symmetric, prove that its Fourier transform is also circularly symmetric.

Show that the cosine transform of an $N \times 1$ sequence can be calculated from the DFTs of the $2N \times 1$ symmetrically extended sequence.

4.15 The 4×4 Hadamard matrix is given by $H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$. Check whether premultiplication of the

matrix H by the matrix $S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ puts the row in the Walsh sequency order.

References

Books

1. K R Rao and P C Yip, *The Transform and Data Compression Handbook*, CRC Press, 2001
2. Alexander D Poularikas, *The Transforms and Applications Handbook*, CRC Press, 1996
3. R N Bracewell, *The Fourier Transform and its Application*, McGraw-Hill, New York
4. K G Beauchamp, *Walsh Functions and their Applications*, Academic Press, 1975
5. Adam Drozdek, *Elements of Data Compression*, Thomson Brooks/cole, 2002
6. N Ahmed and K R Rao, *Orthogonal Transforms for Digital Signal Processing*, Springer-Verlag, Berlin
7. S R Deans *The Radon Transform and some of its Applications*, John Wiley, 1993

Journals

1. D A Bell, *Walsh Functions and Hadamard Matrices*, Electron. Lett., vol. 2, pp. 340–341, September 1966
2. Hotelling, Harold, *Analysis of a Complex of Statistical Variables into Principal Components*, Journal of Educational Psychology, vol. 24, pp. 498–520, 1933
3. J L Walsh, A closed set of normal orthogonal functions, American Journal of Mathematics, 1923
4. Russell M Mersereau and Alan V Oppenheim, *Digital Reconstruction of Multidimensional Signals from their Projections*, Proc. IEEE, vol. 62, pp. 1319–1338, October 1974
5. Frantisek Matus and Jan Flusser, *Image Representations via a Finite Radon Transform*, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 15, no. 10, pp. 996–1006, October 1993
6. V C Klema and A J Laub, *The Singular Value Decomposition: Its Computation and Some Applications*, IEEE Trans. Autom. Control, vol. 25, pp. 164–176, April 1980

Web References

1. Dr K R Rao's teaching material: <http://www-ee.uta.edu/dip/>
2. Professor Min Wu's lecture material: <http://www.ece.umd.edu/class/enee631/>

5

Learning Objectives

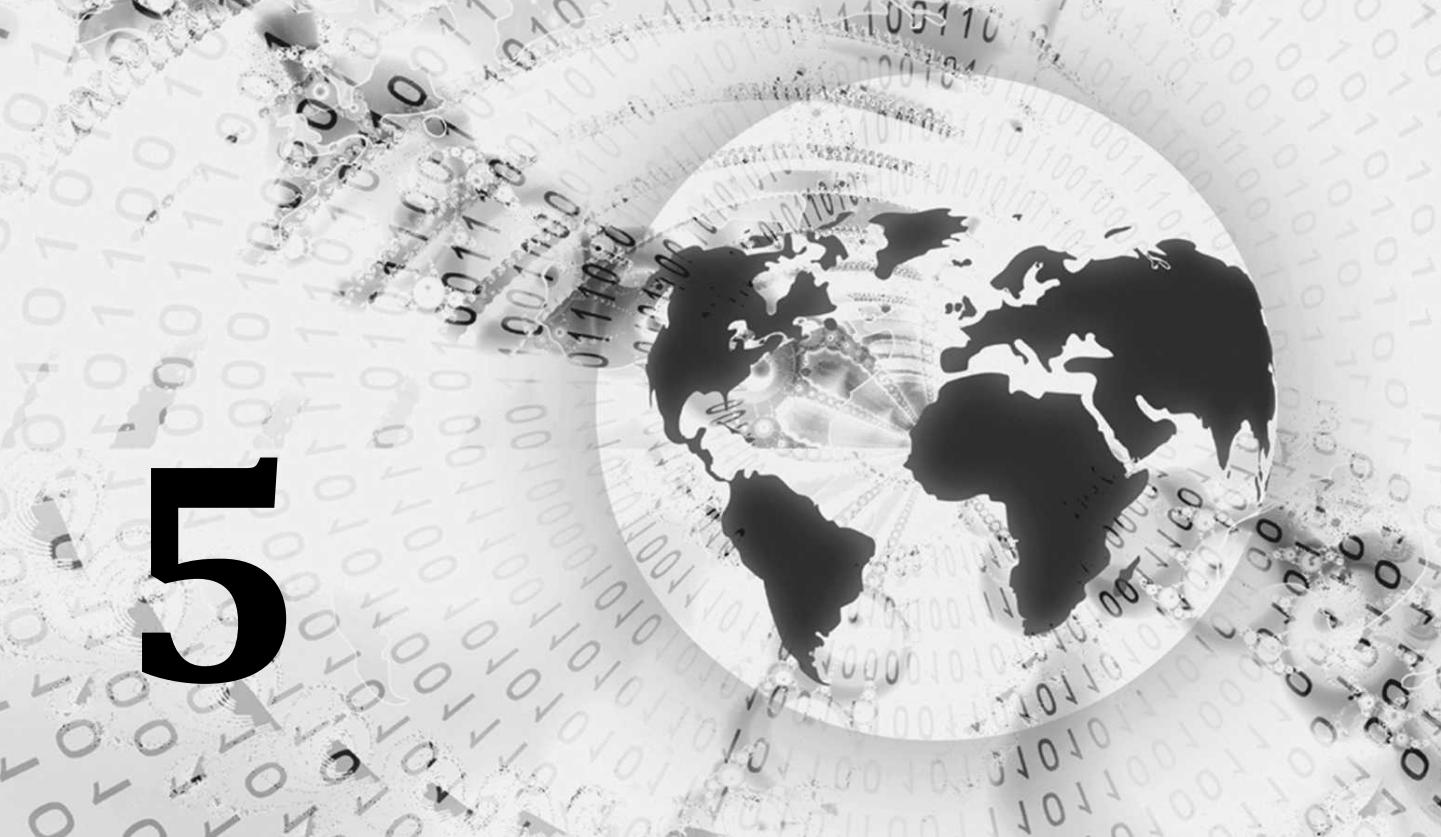
Image-enhancement techniques are designed to improve the quality of an image as perceived by a human being. Image enhancement can be performed both in the spatial as well as in the frequency domain. After reading this chapter, the reader should have a basic knowledge about the following concepts:

*Objectives of image enhancement
image enhancement in spatial and frequency domain
point operations and mask operations in spatial domain
different types of gray-level transformation
histogram and histogram equalisation
frequency domain filtering*

Image Enhancement

5.1 INTRODUCTION

The objective of image enhancement is to improve the interpretability of the information present in images for human viewers. An enhancement algorithm is one that yields a better-quality image for the purpose of some particular application which can be done by either suppressing the noise or increasing the image contrast. Image-enhancement algorithms are employed to emphasise, sharpen or smoothen image features for display



and analysis. Enhancement methods are application specific and are often developed empirically. Image-enhancement techniques emphasise specific image features to improve the visual perception of an image. Image-enhancement techniques can be classified into two broad categories as

- (1) spatial domain method, and (2) transform domain method.

The *spatial domain method* operates directly on pixels, whereas the transform domain method operates on the Fourier transform of an image and then transforms it back to the spatial domain. Elementary enhancement techniques are histogram-based because they are simple, fast, and with them acceptable results for some applications can be achieved. Unsharp masking sharpens the edges by subtracting a portion of a filtered component from the original image. The technique of unsharp masking has become a popular enhancement tool to assist in diagnosis.

5.2 IMAGE ENHANCEMENT IN SPATIAL DOMAIN

The spatial domain technique deals with the manipulation of pixel values. The spatial domain technique can be broadly classified into (i) point operation, (ii) mask operation, and (iii) global operation.

5.2.1 Point Operation

In point operation, each pixel is modified by an equation that is not dependent on other pixel values. The point operation is illustrated in Fig. 5.1.

The point operation is represented by

$$g(m, n) = T[f(m, n)] \quad (5.1)$$

In point operation, T operates on one pixel, or there exists a one-to-one mapping between the input image $f(m, n)$ and the output image $g(m, n)$.

5.2.2 Mask Operation

In mask operation, each pixel is modified according to the values in a small neighbourhood as illustrated in Fig. 5.2. Examples of mask operations are spatial low-pass filtering using a box filter or median filter. In mask operation, the operator T operates on the neighbourhood of pixels.

Here, mask is a small matrix whose values are often termed as weights. Each mask has an origin. The origins of symmetric masks are usually their centre pixel position. For non-symmetric masks, any pixel location may be chosen as the origin, depending on the intended use.

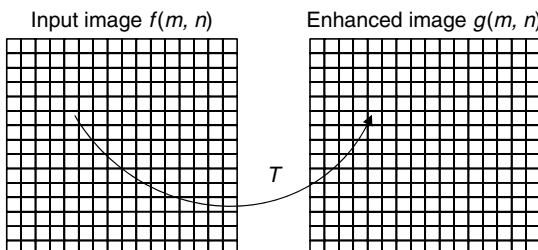


Fig. 5.1 Point operation

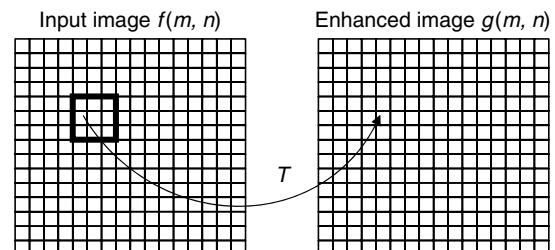


Fig. 5.2 Mask processing

5.2.3 Global Operation

In global operation, all pixel values in the image are taken into consideration. Usually, frequency domain operations are global operations.

5.3 ENHANCEMENT THROUGH POINT OPERATION

In point operation, each pixel value is mapped to a new pixel value. Point operations are basically memoryless operations. In a point operation, the enhancement at any point depends only on the image value at that point. The point operation maps the input image $f(m, n)$ to the output image $g(m, n)$ which is illustrated in Fig. 5.3. From the figure, it is obvious that every pixel of $f(m, n)$ with the same gray level maps to a single gray value in the output image.

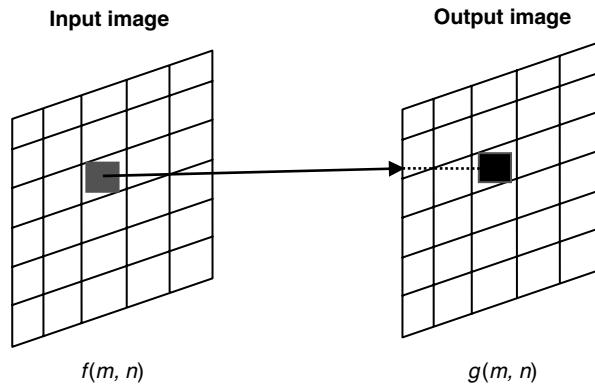


Fig. 5.3 Illustration of point operation

5.4 TYPES OF POINT OPERATION

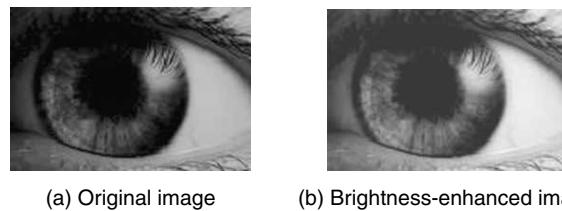
Some of the examples of point operation include (i) brightness modification, (ii) contrast manipulation, and (iii) histogram manipulation.

5.4.1 Brightness Modification

The brightness of an image depends on the value associated with the pixel of the image. When changing the brightness of an image, a constant is added or subtracted from the luminance of all sample values. The brightness of the image can be increased by adding a constant value to each and every pixel of the image. Similarly the brightness can be decreased by subtracting a constant value from each and every pixel of the image.

(a) Increasing the Brightness of an Image A simple method to increase the brightness value of an image is to add a constant value to each and every pixel of the image. If $f[m, n]$ represents the original image then a new image $g[m, n]$ is obtained by adding a constant k to each pixel of $f[m, n]$. This is represented by

$$g[m, n] = f[m, n] + k \quad (5.2)$$

**Fig. 5.4** Brightness enhancement

```

1 - a=imread('eye1.jpg');
2 - b=double(a)+50;
3 - imshow(a),title('original image')
4 - figure,imshow(uint8(b)),title('Enhanced image')

```

Fig. 5.5 MATLAB code for enhancement brightness

Figure 5.4 shows the original and the brightness-enhanced image. The original image $f[m, n]$ is shown in Fig. 5.4 (a). The enhanced image $g[m, n]$ is shown in Fig. 5.4 (b). Here, the value of k is chosen to be 50. Each pixel in $f[m, n]$ is increased by a factor of fifty. The corresponding MATLAB code is shown in Fig. 5.5.

(b) Decreasing the Brightness of an Image The brightness of an image can be decreased by subtracting a constant k from all the pixels of the input image $f[m, n]$. This is represented by

$$g[m, n] = f[m, n] - k \quad (5.3)$$

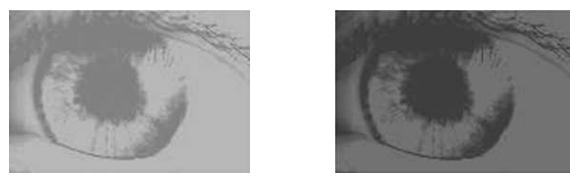
The original and the brightness-suppressed image are shown in Figs 5.6 (a) and (b) respectively. The brightness is suppressed by subtracting a value of 70 from each and every pixel of the original image. The corresponding MATLAB code is given in Fig. 5.7.

5.4.2 Contrast Adjustment

Contrast adjustment is done by scaling all the pixels of the image by a constant k . It is given by

$$g[m, n] = f[m, n] * k \quad (5.4)$$

Changing the contrast of an image, changes the range of luminance values present in the image.



(a) Original image (b) Brightness-suppressed image

Fig. 5.6 Brightness suppression

```

1 - clc
2 - clear all
3 - close all
4 - a=imread('eye1.jpg');
5 - b=double(a)-70;
6 - imshow(a),title('original image');
7 - figure,imshow(uint8(b)),title('Brightness suppressed image')

```

Fig. 5.7 MATLAB code for brightness suppression

Specifying a value above 1 will increase the contrast by making bright samples brighter and dark samples darker, thus expanding on the range used. A value below 1 will do the opposite and reduce a smaller range of sample values. An original image and its contrast-manipulated images are illustrated in Fig. 5.8 and the corresponding MATLAB code is given in Fig. 5.9.



(a) Original image

(b) Increase in contrast

(c) Decrease in contrast

Fig. 5.8 Contrast manipulation

```

1 - clc
2 - clear all
3 - close all
4 - a=imread('monarch.jpg');
5 - b=rgb2gray(a);
6 - c=b^.5;
7 - d=b*20;
8 - imshow(a), title('original image')
9 - figure, imshow(c), title('Increase in contrast')
10 - figure, imshow(d), title('Decrease in contrast')

```

Fig. 5.9 MATLAB code for contrast manipulation

5.5 HISTOGRAM MANIPULATION

Histogram manipulation basically modifies the histogram of an input image so as to improve the visual quality of the image. In order to understand histogram manipulation, it is necessary that one should have some basic knowledge about the histogram of the image. The following section gives basic idea about histograms of an image and the histogram-equalisation technique used to improve the visual quality of an image.

(a) Histogram The histogram of an image is a plot of the number of occurrences of gray levels in the image against the gray-level values. The histogram provides a convenient summary of the intensities in an image, but it is unable to convey any information regarding spatial relationships between pixels. The histogram provides more insight about image contrast and brightness.

1. The histogram of a dark image will be clustered towards the lower gray level.
2. The histogram of a bright image will be clustered towards higher gray level.
3. For a low-contrast image, the histogram will not be spread equally, that is, the histogram will be narrow.
4. For a high-contrast image, the histogram will have an equal spread in the gray level.

Image brightness may be improved by modifying the histogram of the image.

(b) Histogram Equalisation Equalisation is a process that attempts to spread out the gray levels in an image so that they are evenly distributed across their range. Histogram equalisation reassigns the brightness values of pixels based on the image histogram. Histogram equalisation is a technique where the histogram of the resultant image is as flat as possible. Histogram equalisation provides more visually pleasing results across a wider range of images.

(c) Procedure to Perform Histogram Equalisation

Histogram equalisation is done by performing the following steps:

1. Find the running sum of the histogram values.
2. Normalise the values from Step (1) by dividing by the total number of pixels.
3. Multiply the values from Step (2) by the maximum gray-level value and round.
4. Map the gray level values to the results from Step (3) using a one-to-one correspondence.

Example 5.1 Perform histogram equalisation of the image

4	4	4	4	4
3	4	5	4	3
3	5	5	5	3
3	4	5	4	3
4	4	4	4	4

Solution The maximum value is found to be 5. We need a minimum of 3 bits to represent the number. There are eight possible gray levels from 0 to 7. The histogram of the input image is given below:

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0

Step 1 Compute the running sum of histogram values.

The running sum of histogram values is otherwise known as cumulative frequency distribution.

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running sum	0	0	0	6	20	25	25	25

Step 2 Divide the running sum obtained in Step 1 by the total number of pixels. In this case, the total number of pixels is 25.

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running sum	0	0	0	6	20	25	25	25
Running Sum/Total number of pixels	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25

Step 3 Multiply the result obtained in Step 2 by the maximum gray-level value, which is 7 in this case.

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running Sum	0	0	0	6	20	25	25	25
Running sum/Total number of pixels	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25
Multiply the above result by maximum gray level	$\frac{0}{25} * 7$	$\frac{0}{25} * 7$	$\frac{0}{25} * 7$	$\frac{6}{25} * 7$	$\frac{20}{25} * 7$	$\frac{25}{25} * 7$	$\frac{25}{25} * 7$	$\frac{25}{25} * 7$

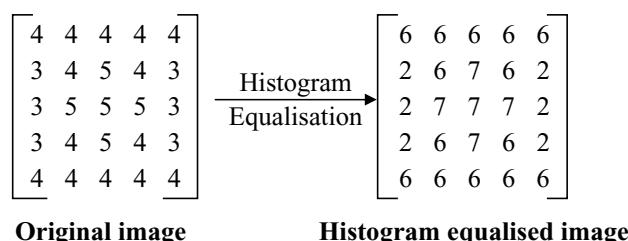
The result is then rounded to the closest integer to get the following table:

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Running Sum	0	0	0	6	20	25	25	25
Running Sum/Total number of pixels	0/25	0/25	0/25	6/25	20/25	25/25	25/25	25/25
Multiply the above result by maximum gray level	0	0	0	2	6	7	7	7

Step 4 Mapping of gray level by a one-to-one correspondence:

Original gray level	Histogram equalised values
0	0
1	0
2	0
3	2
4	6
5	7
6	7
7	7

The original image and the histogram equalised image are shown side by side.



MATLAB Example 1: Histogram equalisation Read an image and perform histogram equalisation of the input image and analyse the result.

Solution The MATLAB code that performs the histogram equalisation and the corresponding result are given in Figs. 5.10 and 5.11 respectively.

```
%This program performs histogram equalisation
clc
clear all
close all
a=imread('babyincradle.png');
%perform histogram equalisation
b=histeq(a);
subplot(2,2,1),imshow(a),title('original image'),
subplot(2,2,2),imshow(b),title('After histogram equalisation'),
subplot(2,2,3),imhist(a),title('original histogram')
subplot(2,2,4),imhist(b),title('After histogram equalisation')
```

Fig. 5.10 MATLAB code to perform histogram equalisation

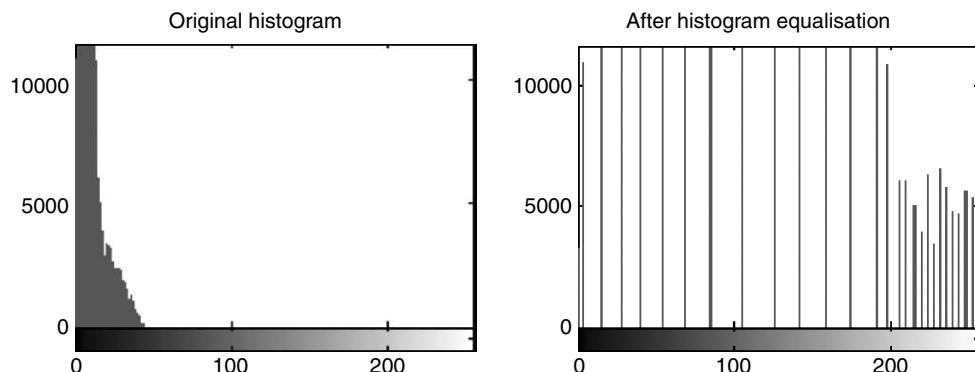
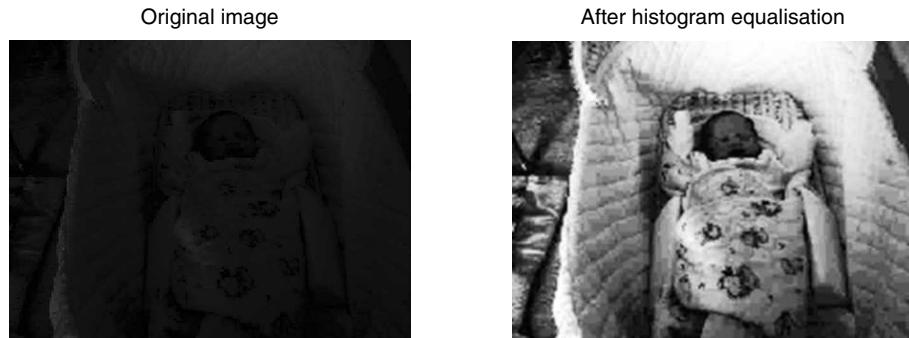
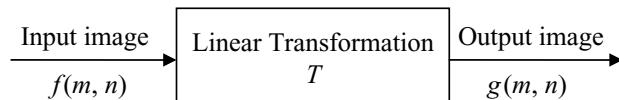


Fig. 5.11 Results of histogram equalisation

From the results of histogram equalisation, it is obvious that histogram equalisation makes it possible to see minor variations within regions that appeared nearly uniform in the original image. In this example, histogram equalisation allows to interpret the original image as a ‘baby in the cradle’.

5.6 LINEAR GRAY-LEVEL TRANSFORMATION

A linear transformation of an image is a function that maps each pixel gray-level value into another gray-level at the same position according to a linear function. The linear transformation is represented by



The linear transformation is given by $g(m, n) = T[f(m, n)]$.

5.6.1 Image Negative or Inverse Transformation

The inverse transformation reverses light and dark. An example of inverse transformation is an image negative. A negative image is obtained by subtracting each pixel from the maximum pixel value. For an 8-bit image, the negative image can be obtained by reverse scaling of the gray levels, according to the transformation

$$g(m, n) = 255 - f(m, n) \quad (5.5)$$

The graphical representation of negation is shown Fig. 5.12.

Negative images are useful in the display of medical images and producing negative prints of images. The MATLAB code to determine the negative of the input image and the corresponding output are shown in Figs 5.13 and 5.14 respectively.

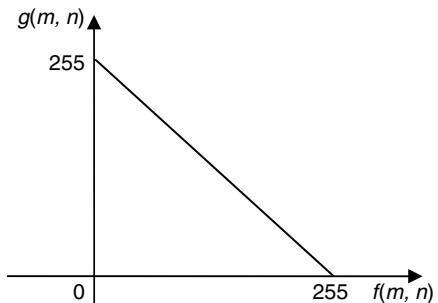


Fig. 5.12 Graphical representation of negation

```

Editor - D:\work\imnegative.m*
File Edit Text Cell Tools Debug Desktop Window Help
File Text Cell Tools Debug Desktop Window Help
3 - clear all
4 - close all
5 - a=imread('lotus.jpg');
6 - %Subtract the maximum value from each pixel
7 - k=255-a;
8 - subplot(2,1,1),imshow(a),title('original image')
9 - subplot(2,1,2),imshow(b),title('negative of original image')

```

Fig. 5.13 MATLAB code to determine image negative

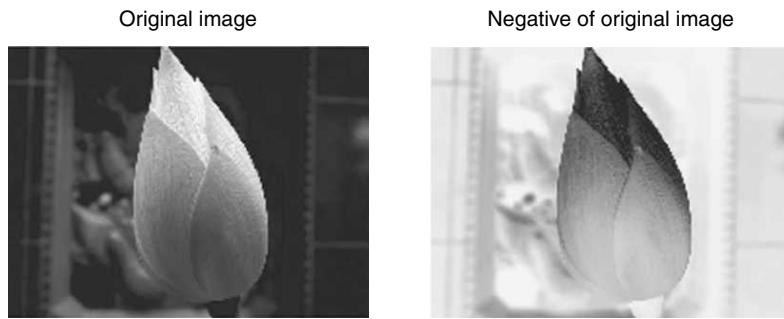


Fig. 5.14 Output of the MATLAB code for image inversion

From Fig. 5.14, it is clear that the negative of the image sometimes gives more information for interpretation than the original image.

5.7 NONLINEAR GRAY-LEVEL TRANSFORMATION

Non-linear transformation maps small equal intervals into non-equal intervals. Some of the non-linear transformations to be discussed in this section are (i) thresholding, (ii) gray-level slicing, (iii) logarithmic transformation, (iv) exponential transformation, and (v) power law transformation.

5.7.1 Thresholding

Thresholding is required to extract a part of an image which contains all the information. Thresholding is a part of a more general segmentation problem. Thresholding can be broadly classified into (i) hard thresholding, and (ii) soft thresholding.

(a) Hard Thresholding In hard thresholding, pixels having intensity lower than the threshold T are set to zero and the pixels having intensity greater than the threshold are set to 255 or left at their original intensity depending on the effect that is required. This type of hard thresholding allows us to obtain a binary image from a grayscale image.

Application of Hard Thresholding Hard thresholding can be used to obtain a binary image from a grayscale image. The grayscale mapping which allows us to obtain a binary image from a grayscale image is shown in Fig. 5.15.

The mathematical expression for hard thresholding is given below:

$$g(m, n) = \begin{cases} 0 & \text{for } f(m, n) < t \\ 255 & \text{otherwise} \end{cases} \quad (5.6)$$

Here, $f(m, n)$ represents the input image, $g(m, n)$ represents the output image and t represents the threshold parameter.

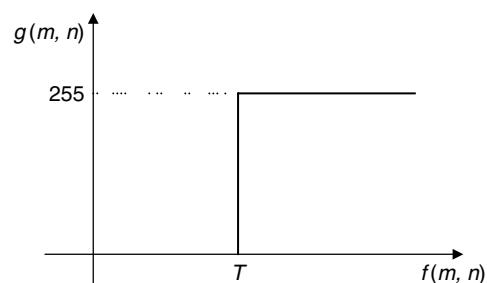
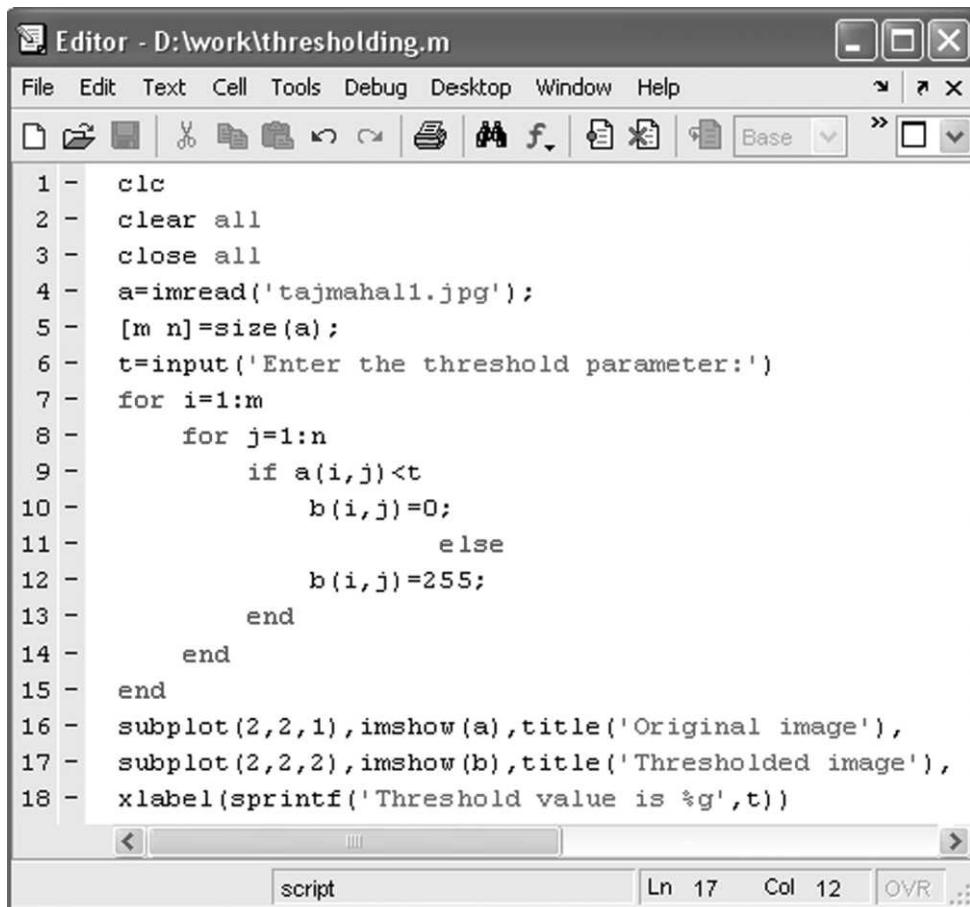


Fig. 5.15 Hard thresholding

MATLAB Example 2 *Read a grayscale image and convert it into a binary image using hard thresholding. Make the threshold value as a user-defined parameter. Vary the threshold and observe the result.*

Solution The MATLAB code performs the thresholding operation on the input grayscale image as shown in Fig. 5.16. The corresponding outputs are shown in Figs 5.17 and 5.18.



```

1 - clc
2 - clear all
3 - close all
4 - a=imread('tajmahali1.jpg');
5 - [m n]=size(a);
6 - t=input('Enter the threshold parameter:');
7 - for i=1:m
8 -     for j=1:n
9 -         if a(i,j)<t
10 -             b(i,j)=0;
11 -         else
12 -             b(i,j)=255;
13 -         end
14 -     end
15 - end
16 - subplot(2,2,1),imshow(a),title('Original image'),
17 - subplot(2,2,2),imshow(b),title('Thresholded image'),
18 - xlabel(sprintf('Threshold value is %g',t))

```

Fig. 5.16 MATLAB code that performs threshold operation

From the results, it is obvious that as the value of the threshold is increased, the image becomes too dark. The optimal threshold in this case is found to be 120 to 160.

5.7.2 Gray-level Slicing

The purpose of gray-level slicing is to highlight a specific range of gray values. Two different approaches can be adopted for gray-level slicing.

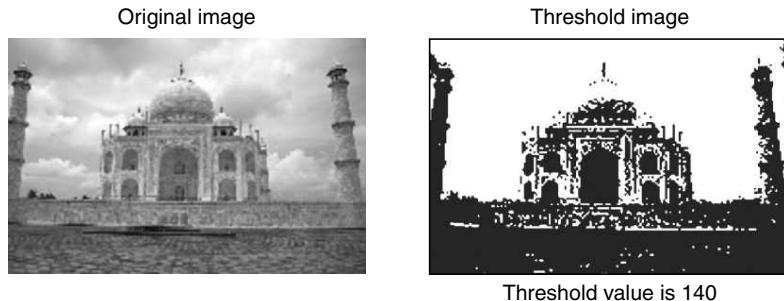


Fig. 5.17 Result of low threshold



Fig. 5.18 Results of threshold operation

(a) Gray-level Slicing without Preserving Background

This displays high values for a range of interest and low values in other areas. The main drawback of this approach is that the background information is discarded. The pictorial representation of this approach is shown in Fig. 5.19.

The MATLAB code which performs gray level slicing without preserving the background is shown in Fig. 5.20 and the corresponding output is shown in Fig. 5.21.

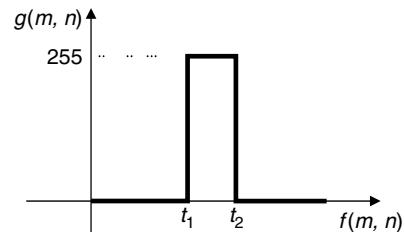


Fig. 5.19 Gray-level slicing without background

(b) Gray-level Slicing with Background In gray-level slicing with background, the objective is to display high values for the range of interest and original gray level values in other areas. This approach preserves the background of the image. The graphical representation of this approach to gray-level slicing is shown in Fig. 5.22.

The MATLAB code which performs gray-level slicing by preserving background information is shown in Fig. 5.23 and the corresponding output is shown in Fig. 5.24.

By comparing Fig. 5.21 with Fig. 5.24, it is obvious that the background information is preserved in Fig. 5.24. The value of the threshold t_1 is fixed as 204 and the value of the threshold t_2 is fixed as 255 in both gray-level slicing without background and with background.

```
%This program performs gray level slicing without background
clc;
clear all;
close all;
x=imread('goldfish.tif');
x=imresize(x, [256 256]);
y=double(x);
[m,n]=size(y);
L=double(255);
a=double(round(L/1.25));
b=double(round(2*L/2));
for i=1:m
    for j=1:n
        if (y(i,j)>=a & y(i,j)<=b)
            z(i,j)=L;
        else
            z(i,j)=0;
        end
    end
end
imshow(uint8(y));
Figure,imshow(uint8(z));
```

Fig. 5.20 Gray-level slicing without preserving background

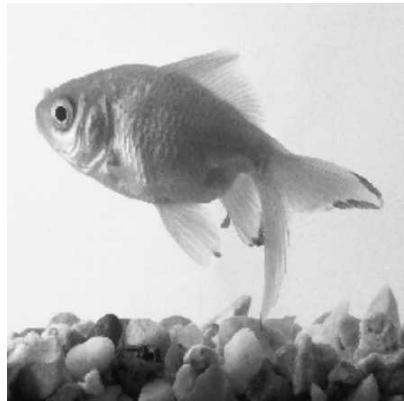


Fig. 5.21 Result of gray-level slicing without preserving background

5.7.3 Logarithmic Transformation

The logarithmic transformation is given by

$$g(m, n) = c \log_2(f(m, n) + 1) \quad (5.7)$$

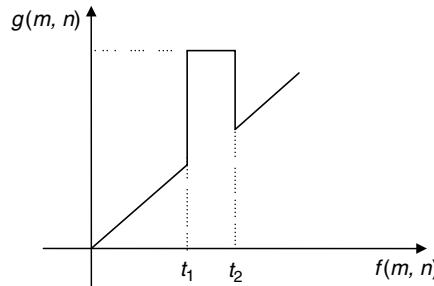


Fig. 5.22 Gray-level slicing with background

```
%This program performs gray level slicing with background
clc;
clear all;
close all;
x=imread('goldfish.tif');
x=imresize(x,[256 256]);
y=double(x);
[m,n]=size(y);
L=double(255);
a=double(round(L/1.25));
b=double(round(2*L/2));
for i=1:m
    for j=1:n
        if (y(i,j)>=a & y(i,j)<=b)
            z(i,j)=L;
        else
            z(i,j)=y(i,j);
        end
    end
end
imshow(uint8(y));
Figure,imshow(uint8(z));
```

Fig. 5.23 MATLAB code for gray-level slicing with background

This type of mapping spreads out the lower gray levels. For an 8-bit image, the lower gray level is zero and the higher gray level is 255. It is desirable to map 0 to 0 and 255 to 255. The function $g(m, n) = \text{clog}(f(m, n) + 1)$ spreads out the lower gray levels.

The MATLAB code which performs logarithmic transformation is shown in Fig. 5.25 and the corresponding result is shown in Fig. 5.26.

5.7.4 Exponential Transformation

The non-linear transformation that has been used mainly in conjunction with logarithmic transformation in multiplicative filtering operations is exponential transformation. The effect of an exponential transfer

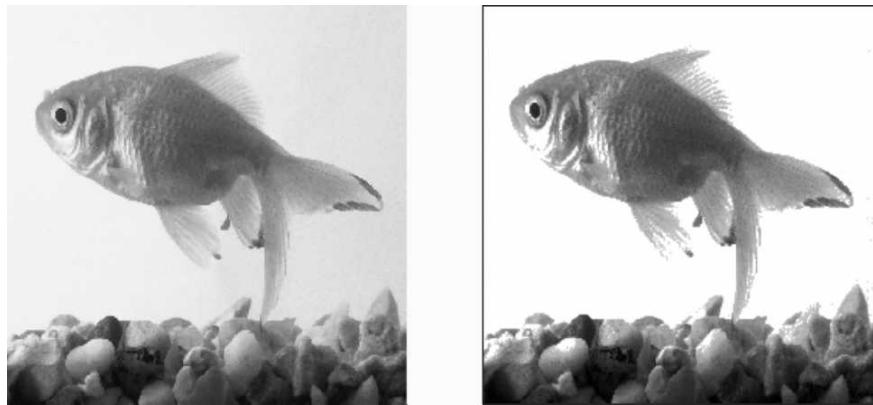


Fig. 5.24 Result of gray-level slicing with background preservation

```
%This code performs logarithmic transformation
a=imread('crow.jpg');
L=255;
c=L/log10(1+L);
d=c*log10(1+double(a));
imshow(a),title('original image')
figure,imshow(uint8(d)),title('Log transformation image')
```

Fig. 5.25 Logarithmic transformation code

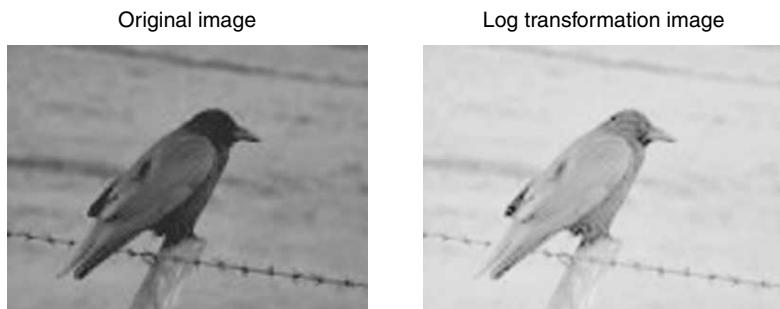


Fig. 5.26 Result of logarithmic transformation

function on edges in an image is to compress low-contrast edges while expanding high-contrast edges. This generally produces an image with less visible detail than the original and this is not a desirable enhancement transformation.

5.7.5 Gamma Correction or Power Law Transformation

The intensity of light generated by a physical device such as a CRT is not a linear function of the applied signal. The intensity produced at the surface of the display is approximately the applied voltage, raised to the power of 2.5.

The numerical value of the exponent of this power function is termed as gamma. This non-linearity must be compensated in order to achieve correct reproduction of intensity.

The power law transformation is given by

$$g(m, n) = [f(m, n)]^\gamma \quad (5.8)$$

where $f(m, n)$ is the input image and $g(m, n)$ is the output image. Gamma (γ) can take either integer or fraction values. The MATLAB code that performs gamma correction is shown in Fig. 5.27 and the corresponding output is shown in Fig. 5.28.

Figure 5.28 shows the power-law transformed image for two different values of gamma. When the value of gamma is less than one, the image appears to be a dark image and when the value of gamma is greater than one, the image appears to be a bright image.

```
clc
clear all
close all
a=imread('myna.jpg');
a=rgb2gray(a);
gamma=1.1;
d=double(a).^gamma;
imshow(a),title('original image')
Figure,imshow(uint8(d)),title('Powerlaw transformation')
```

Fig. 5.27 MATLAB code to perform power law transformation

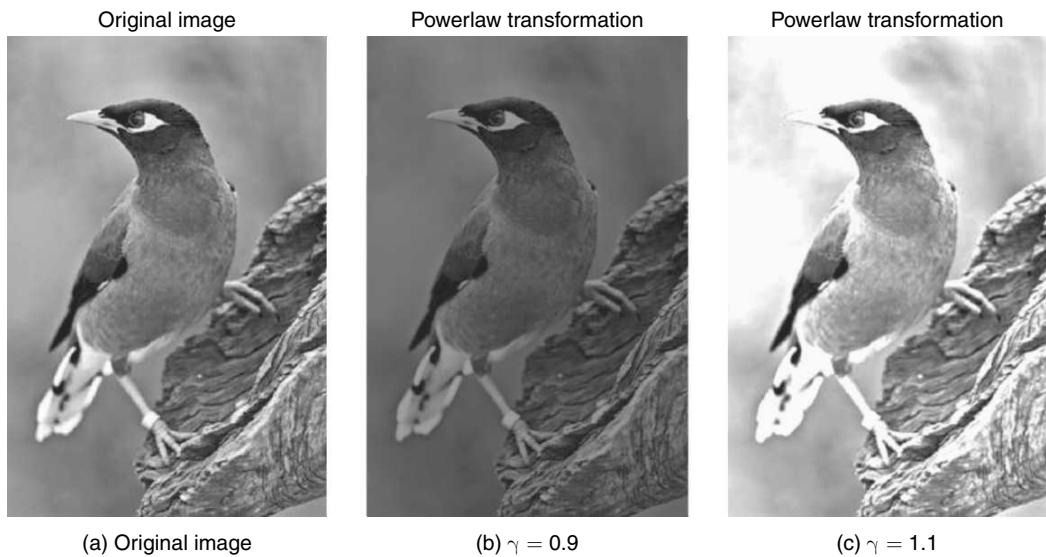


Fig. 5.28 Results of power law transformation

5.8 LOCAL OR NEIGHBOURHOOD OPERATION

In neighbourhood operation, the pixels in an image are modified based on some function of the pixels in their neighbourhood. Linear spatial filtering is often referred to as convolving a mask with an image. The filter masks are sometimes called *convolution masks* or *convolution kernels*.

5.8.1 Spatial Filtering

Spatial filtering involves passing a weighted mask, or kernel over the image and replacing the original image pixel value corresponding to the centre of the kernel with the sum of the original pixel values in the region corresponding to the kernel multiplied by the kernel weights.

5.8.2 Linear Filtering

In linear filtering, each pixel in the input image is replaced by a linear combination of intensities of neighbouring pixels. That is, each pixel value in the output image is a weighted sum of the pixels in the neighbourhood of the corresponding pixel in the input image. Linear filtering can be used to smoothen an image as well as sharpen the image. A spatially invariant linear filter can be implemented using a convolution mask. If different filter weights are used for different parts of the image then the linear filter is spatially varying.

5.8.3 Mean Filter or Averaging Filter or Low-pass Filter

The mean filter replaces each pixel by the average of all the values in the local neighbourhood. The size of the neighbourhood controls the amount of filtering. In a spatial averaging operation, each pixel is replaced by a weighted average of its neighbourhood pixels. The low-pass filter preserves the smooth region in the image and it removes the sharp variations leading to blurring effect. The 3 by 3 spatial mask which can perform the averaging operation is given below:

$$3 \text{ by } 3 \text{ low-pass spatial mask} = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{Similarly, the } 5 \text{ by } 5 \text{ averaging mask} = \frac{1}{25} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

It is to be noted that the sum of the elements is equal to 1 in the case of a low-pass spatial mask. The blurring effect will be more with the increase in the size of the mask. Normally, the size of the mask will be odd so that the central pixel can be located exactly.

5.8.4 Limitations of Averaging Filter

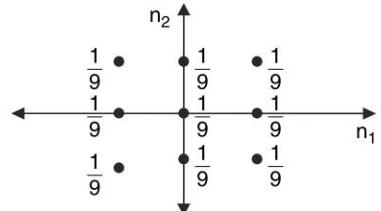
The limitations of averaging filters are given below:

1. Averaging operation leads to the blurring of an image. Blurring affects feature localisation.
2. If the averaging operation is applied to an image corrupted by impulse noise then the impulse noise is attenuated and diffused but not removed.

3. A single pixel, with a very unrepresentative value can affect the mean value of all the pixels in its neighbourhood significantly.

Example 5.2 Analyse 3×3 mean filter in the frequency domain and prove that it behaves like a low-pass filter.

Solution The 3×3 mean filter is given by $\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. The pictorial representation is given in Fig. 5.29.



The representation of the pattern shown in Fig. 5.29 is given below:

$$x(n_1, n_2) = \frac{1}{9} \delta(n_1, n_2) + \frac{1}{9} \delta(n_1 - 1) + \frac{1}{9} \delta(n_1 + 1) + \frac{1}{9} \delta(n_2 - 1) + \frac{1}{9} \delta(n_2 + 1) + \frac{1}{9} \delta(n_1 - 1, n_2 - 1) \\ + \frac{1}{9} \delta(n_1 + 1, n_2 - 1) + \frac{1}{9} \delta(n_1 + 1, n_2 + 1) + \frac{1}{9} \delta(n_1 - 1, n_2 + 1)$$

Taking Z-transform on both sides, we get

$$X[z_1, z_2] = \frac{1}{9} + \frac{1}{9} z_1^{-1} + \frac{1}{9} z_1 + \frac{1}{9} z_2^{-1} + \frac{1}{9} z_2 + \frac{1}{9} z_1^{-1} z_2^{-1} + \frac{1}{9} z_1 z_2^{-1} + \frac{1}{9} z_1 z_2 + \frac{1}{9} z_1^{-1} z_2 \\ X[\omega_1, \omega_2] = \frac{1}{9} + \frac{1}{9} e^{-j\omega_1} + \frac{1}{9} e^{j\omega_1} + \frac{1}{9} e^{-j\omega_2} + \frac{1}{9} e^{j\omega_2} + \frac{1}{9} e^{-j\omega_1} e^{-j\omega_2} + \frac{1}{9} e^{j\omega_1} e^{-j\omega_2} + \frac{1}{9} e^{j\omega_1} e^{j\omega_2} \\ + \frac{1}{9} e^{-j\omega_1} e^{j\omega_2} \\ = \frac{1}{9} + \frac{1}{9} (e^{-j\omega_1} + e^{j\omega_1}) + \frac{1}{9} (e^{-j\omega_2} + e^{j\omega_2}) + \frac{1}{9} e^{-j\omega_2} (e^{j\omega_1} + e^{-j\omega_1}) + \frac{1}{9} e^{j\omega_2} (e^{j\omega_1} + e^{-j\omega_1}) \\ = \frac{1}{9} + \frac{1}{9} \times 2 \cos \omega_1 + \frac{1}{9} \times 2 \cos \omega_2 + \frac{1}{9} e^{-j\omega_2} \times 2 \cos \omega_1 + \frac{1}{9} e^{j\omega_2} \times 2 \cos \omega_1 \\ = \frac{1}{9} + \frac{1}{9} \times 2 \cos \omega_1 + \frac{1}{9} \times 2 \cos \omega_2 + \frac{1}{9} \times 2 \cos \omega_1 (e^{-j\omega_2} + e^{j\omega_2}) \\ = \frac{1}{9} + \frac{1}{9} \times 2 \cos \omega_1 + \frac{1}{9} \times 2 \cos \omega_2 + \frac{1}{9} \times 2 \cos \omega_1 \times 2 \cos \omega_2 \\ X[\omega_1, \omega_2] = \frac{1}{9} + \frac{1}{9} \times 2 \cos \omega_1 + \frac{1}{9} \times 2 \cos \omega_2 + \frac{1}{9} \times 4 \cos \omega_1 \cos \omega_2 \quad (5.9)$$

Case (i) Substitute $\omega_1 = \omega_2 = 0$ in the expression of $X[\omega_1, \omega_2]$ given in Eq. (5.9).

$$X[0, 0] = \frac{1}{9} + \frac{1}{9} \times 2 \cos(0) + \frac{1}{9} \times 2 \cos(0) + \frac{1}{9} \times 4 \cos(0) \cos(0)$$

$$X[0, 0] = \frac{1}{9} + \frac{2}{9} + \frac{2}{9} + \frac{4}{9} = \frac{9}{9} = 1.$$

$X[0,0]$ represents a very low frequency or dc value. The system output is 1 when the frequency is 0. This implies that the system behaves like a low-pass filter.

Case (ii) Substitute $\omega_1 = \omega_2 = \frac{\pi}{2}$ in the expression of $X[\omega_1, \omega_2]$ given in Eq. (5.9)

$$X\left[\frac{\pi}{2}, \frac{\pi}{2}\right] = \frac{1}{9} + \frac{1}{9} \times 2 \cos\left(\frac{\pi}{2}\right) + \frac{1}{9} \times 2 \cos\left(\frac{\pi}{2}\right) + \frac{1}{9} \times 4 \cos\left(\frac{\pi}{2}\right) \cos\left(\frac{\pi}{2}\right)$$

$$X\left[\frac{\pi}{2}, \frac{\pi}{2}\right] = \frac{1}{9}$$

Case (iii) Substitute $\omega_1 = \omega_2 = \pi$ in the expression of $X[\omega_1, \omega_2]$ given in Eq. (5.9)

$$X[\pi, \pi] = \frac{1}{9} + \frac{1}{9} \times 2 \cos(\pi) + \frac{1}{9} \times 2 \cos(\pi) + \frac{1}{9} \times 4 \cos(\pi) \cos(\pi)$$

$$X[\pi, \pi] = \frac{1}{9} - \frac{2}{9} - \frac{2}{9} + \frac{4}{9} = \frac{1}{9}$$

Comparing the outputs of Case(i), Case(ii) and Case(iii), we can observe that the output in Case (i) when both $\omega_1 = \omega_2 = 0$ is high. This shows that the mask acts as a low-pass filter.

The frequency response of the mask and the corresponding MATLAB code are shown in Figs 5.30 and 5.31 respectively.

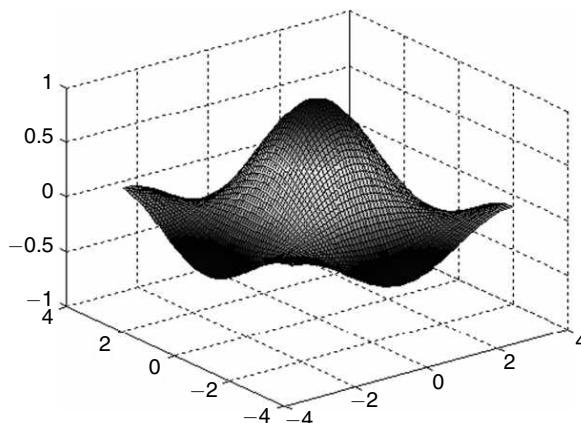


Fig. 5.30 Frequency response of a 3×3 mask

```
close all;
clear all;
clc;
[X,Y] = meshgrid(-pi:.09:pi);
Z = (1/9)*(1+2*cos(X)+2*cos(Y)+4*(cos(X).*cos(Y)));
surf(X,Y,Z),axis([-4 4,-4 4,-1 1])
```

Fig. 5.31 MATLAB code to generate the frequency response

Example 5.3: Programming task Read an image, convolve the image with the mask $\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ and

show that it performs averaging operation which results in blurring of the image. Also, analyse the impact

of increasing the size of the mask to 5×5 , that is, mask is $\frac{1}{25} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$.

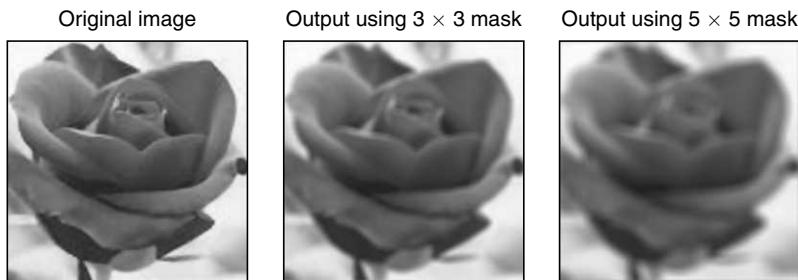
Solution The MATLAB code to perform spatial-domain operation filtering is shown in Fig. 5.32. The corresponding results are shown in Fig. 5.33. From the result, it is obvious that as the size of the mask increases, the blurring increases. Also, one can observe that a spatial filter is implemented through convolution. The masks $h1$ and $h2$ are generally called as *box filters*. It is to be noted that the masks $h1$ and $h2$ basically perform the averaging operation.

```

1 %This program performs spatial domain filtering
2 - clc
3 - clear all
4 - close all
5 - a=imread('rose.jpg');
6 - h1=1/9*ones(3,3); % 3 x 3 mask
7 - h2=1/25*ones(5,5); % 5 x 5 mask
8 - b1=conv2(a,h1,'same');
9 - b2=conv2(a,h2,'same');
10 - imshow(a),title('original image'),
11 - figure,imshow(uint8(b1)),title('Output using 3 x 3 mask')
12 - ,figure,imshow(uint8(b2)),title('Output using 5x5 mask')
13
14

```

Fig. 5.32 Spatial domain filtering

**Fig. 5.33** Results of spatial domain filters

Example 5.4: Noise minimisation using averaging filter *Read an image and then corrupt the image by salt-and-pepper noise and Gaussian noise. Then apply an averaging filter of size 3×3 and 5×5 to this corrupted image. Comment on the result obtained.*

Solution The MATLAB code that reads the image and performs the task is given in Fig. 5.34.

After executing the code given in Fig. 5.34, the results are shown in Figs 5.35, 5.36 and 5.37 respectively.

Figure 5.35 shows the impact of applying a 3×3 and 5×5 averaging filter to an image corrupted by salt and pepper noise. From the Figure, it is clear that a 5×5 mask is more effective in minimising the impact of noise. Also, one can observe that as the size of the mask increases, it leads to blurring of the image.

Conclusion

Increasing the size of the mask results in more blurring of the image.

Question

Suggest a suitable filter that will reduce the impact of salt-and-pepper noise at the same time it preserves the edges in the image.

Answer

The filter that will reduce the impact of salt-and-pepper noise with minimum blurring is a median filter.

Figure 5.36 shows the impact of applying a 3×3 and 5×5 averaging filter to an image corrupted by Gaussian noise.

From the figure, it is clear that a 5×5 mask is more effective in minimising the impact of noise. Also, one can observe that as the size of the mask increases, it leads to blurring of the image.

Question

Suggest a suitable filter that could minimise the impact of 'Gaussian noise'?

Answer

Gaussian filter is an optimum filter that could reduce the impact of Gaussian noise.

Figure 5.37 shows the impact of applying a 3×3 and 5×5 averaging filter to an image corrupted by speckle noise.

From the figure, it is clear that a 5×5 mask is more effective in minimising the impact of noise.

```

a=imread('rose.jpg');
% Addition of noise to the input image
b=imnoise(a,'salt & pepper');
c=imnoise(a,'gaussian');
d=imnoise(a,'speckle');
% Defining 3x3 and 5x5 kernel
h1=1/9*ones(3,3);
h2=1/25*ones(5,5);
% Attempt to recover the image
b1=conv2(b,h1,'same');
b2=conv2(b,h2,'same');
c1=conv2(c,h1,'same');
c2=conv2(c,h2,'same');
d1=conv2(d,h1,'same');
d2=conv2(d,h2,'same');
%Displaying the result
figure, subplot(2,2,1), imshow(a), title('Original Image'),
subplot(2,2,2), imshow(b), title('Salt & Pepper noise'),
subplot(2,2,3), imshow(uint8(b1)), title('3 x 3 Averaging
filter'),
subplot(2,2,4), imshow(uint8(b2)), title('5 x 5 Averaging
filter')
%.....
figure, subplot(2,2,1), imshow(a), title('Original Image'),
subplot(2,2,2), imshow(c), title('Gaussian noise'),
subplot(2,2,3), imshow(uint8(c1)), title('3 x 3 Averaging
filter'),
subplot(2,2,4), imshow(uint8(c2)), title('5 x 5 Averaging
filter'),
%.....
figure, subplot(2,2,1), imshow(a), title('Original Image'),
subplot(2,2,2), imshow(d), title('Speckle noise'),
subplot(2,2,3), imshow(uint8(d1)), title('3 x 3 Averaging
filter'),
subplot(2,2,4), imshow(uint8(d2)), title('5 x 5 Averaging
filter'),

```

Fig. 5.34 MATLAB code for Example 5.4

5.8.5 Weighted Average Filter

The mask of a weighted average filter is given by $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$. From the mask, it is obvious that the pixels nearest to the centre are weighted more than the distant pixels; hence it is given the name *weighted average filter*.

The pixel to be updated is replaced by a sum of the nearby pixel value times the weights given in the matrix and divided by the sum of the coefficients in the matrix.

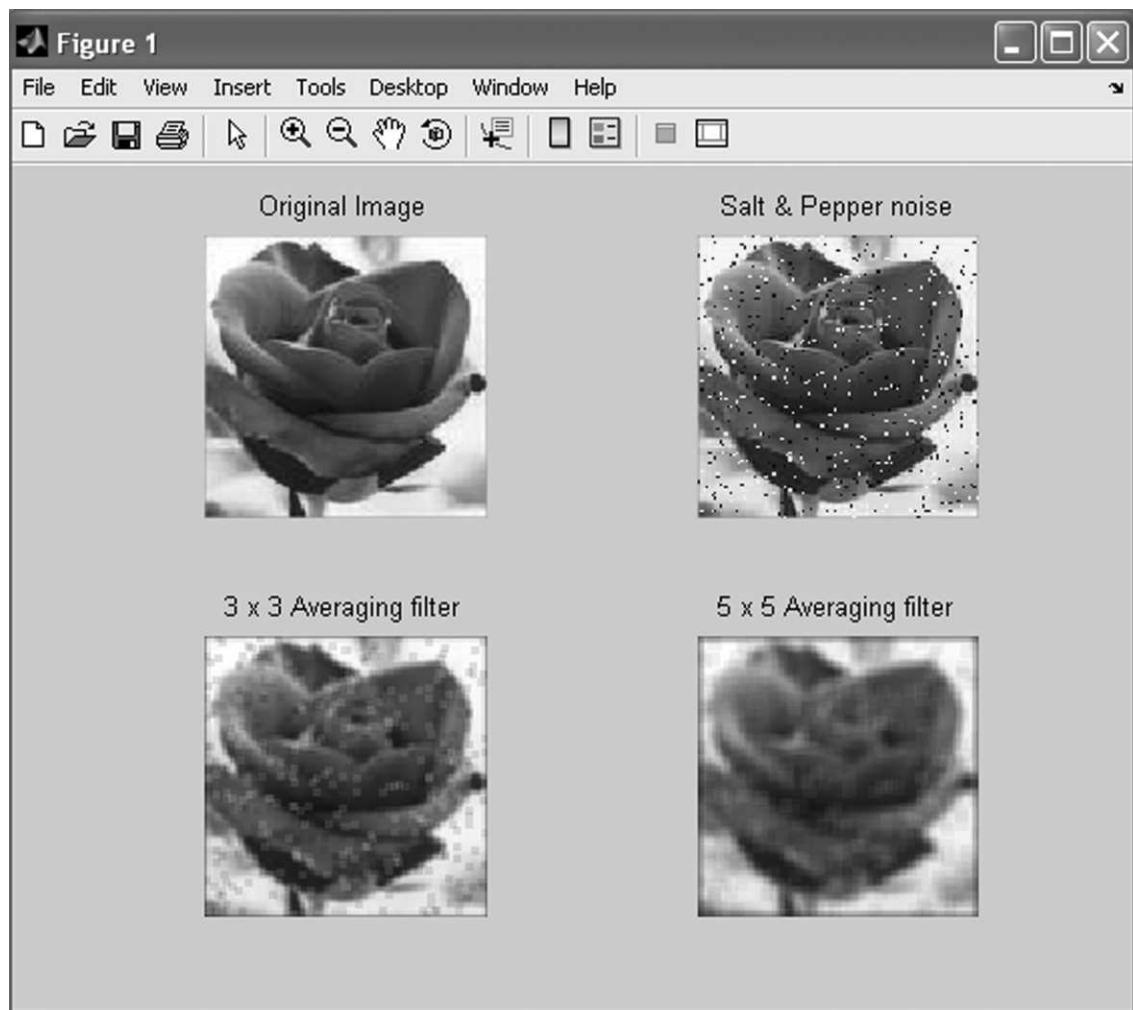


Fig. 5.35 Effect of averaging filter for salt-and-pepper noise

5.8.6 Bartlett Filter

The bartlett filter is a triangle-shaped filter in the spatial domain. A bartlett filter is obtained by convolving two box filters in the spatial domain, or it can be obtained by taking the product of two box filters in the frequency domain.

The 3×3 box filter is given by $\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. From this, a Barlett window in the spatial domain is given by

$$\text{Bartlett window} = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} * \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{8} \times \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}.$$

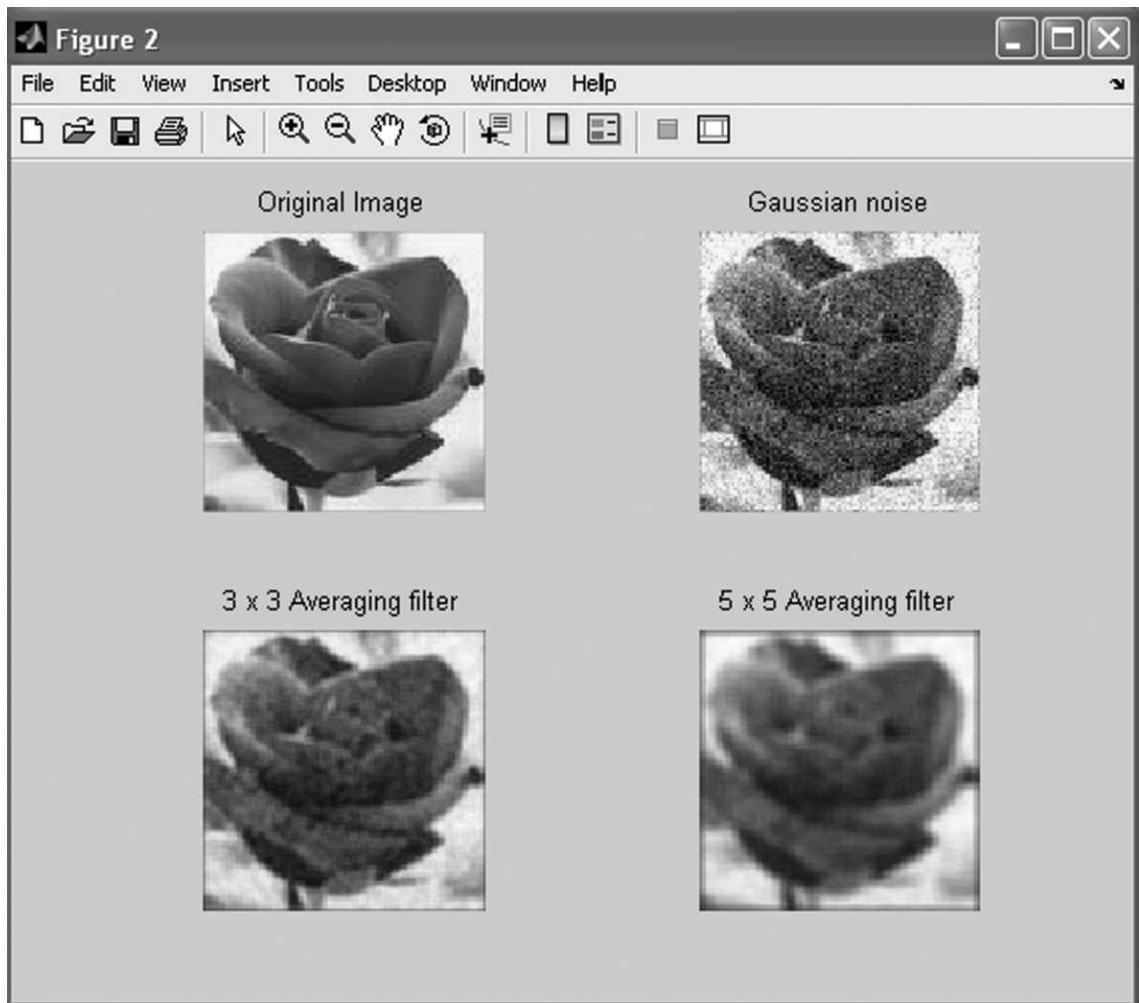


Fig. 5.36 Effect of averaging filter for Gaussian noise

5.8.7 Gaussian Filter

Gaussian filters are a class of linear smoothing filters with the weights chosen according to the shape of a Gaussian function. The Gaussian kernel is widely used for smoothing purpose. The Gaussian filter in the continuous space is given by

$$h(m, n) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{m^2}{2\sigma^2}} \right) \times \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{n^2}{2\sigma^2}} \right) \quad (5.10)$$

The above expression shows that a Gaussian filter is separable. The Gaussian smoothing filter is a very good filter for removing noise drawn from a normal distribution. Gaussian smoothing is a particular class of averaging, in which the kernel is a 2D Gaussian. Gaussian functions have the following properties that make them useful in image processing:

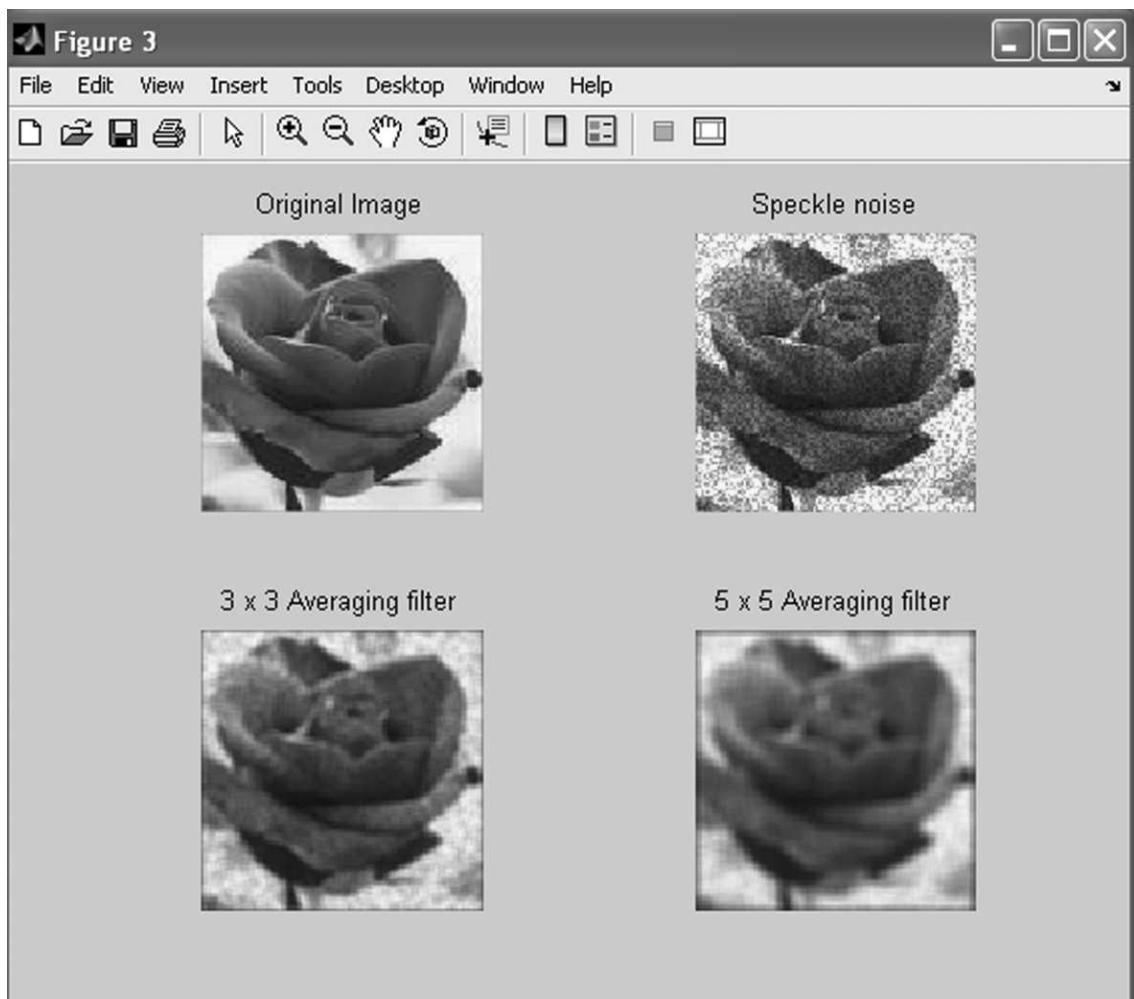


Fig. 5.37 Effect of averaging filter for speckle noise

- (i) Gaussian functions are rotationally symmetric in two dimensions. The meaning of the term 'rotationally symmetric' is that the amount of smoothing performed by the filter will be the same in all directions. Since a Gaussian filter is rotationally symmetric, it will not bias subsequent edge detection in any particular direction.
- (ii) The Fourier transform of a Gaussian function is itself a Gaussian function. The Fourier transform of a Gaussian has a single lobe in the frequency spectrum. Images are often corrupted by high-frequency noise, and the desirable feature of the image will be distributed both in the low-and-high frequency spectrum. The single lobe in the Fourier transform of a Gaussian means that the smoothed image will not be corrupted by contributions from unwanted high-frequency signals, while most of the desirable signal properties will be retained.
- (iii) The degree of smoothening is governed by variance σ . A larger σ implies a wider Gaussian filter and greater smoothening.

- (iv) Two-dimensional Gaussian functions are separable. This property implies that large Gaussian filters can be implemented very efficiently. Two-dimensional Gaussian convolution can be performed by convolving the image with a one-dimensional Gaussian and then convolving the result with the same one-dimensional filter oriented orthogonal to the Gaussian used in the first stage.

Gaussian filters can be applied recursively to create a Gaussian pyramid. A Gaussian filter can be generated from PASCAL's triangle which is shown in Fig. 5.38.

3 × 3 Gaussian Kernel The generation of a 3×3 Gaussian mask from PASCAL's triangle is illustrated in Fig. 5.39.

The 3×3 Gaussian kernel can be generated from the third row of PASCAL's triangle as given below:

The third row of PASCAL's triangle is 1 2 1.

The sum of the elements in the third row of PASCAL's triangle is $1 + 2 + 1 = 4$.

$$\text{The Gaussian kernel is generated as } \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$\text{The } 3 \times 3 \text{ Gaussian kernel is given by } \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Generation of a 4×4 Gaussian Kernel The 4×4 Gaussian kernel can be generated from the fourth row of PASCAL's triangle.

The fourth row of PASCAL's triangle is given by 1 3 3 1.

Sum of the elements in the fourth row is given by: $1 + 3 + 3 + 1 = 8$

$$\text{A } 4 \times 4 \text{ Gaussian kernel is generated as } \frac{1}{8} \times \begin{bmatrix} 1 \\ 3 \\ 3 \\ 1 \end{bmatrix} \times \frac{1}{8} \times \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix} = \frac{1}{64} \times \begin{bmatrix} 1 \\ 3 \\ 3 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix}$$

$$\text{The } 4 \times 4 \text{ Gaussian kernel is given by } \frac{1}{64} \times \begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}.$$

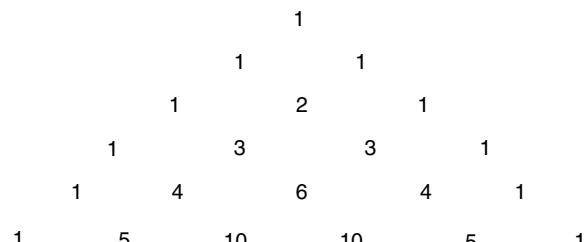


Fig. 5.38 Pascal's triangle

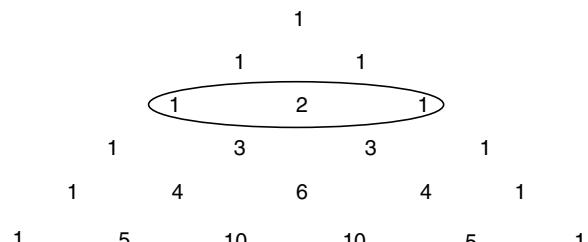


Fig. 5.39 Generation of a 3×3 Gaussian kernel from Pascal's triangle

5.9 MEDIAN FILTER

Median filters are statistical non-linear filters that are often described in the spatial domain. A median filter smoothes the image by utilising the median of the neighbourhood. The concept of a median filter was introduced by Tukey in 1977. Its extension to two-dimensional images was discussed by Pratt in 1978. Median filters perform the following tasks to find each pixel value in the processed image:

1. All pixels in the neighbourhood of the pixel in the original image which are identified by the mask are sorted in the ascending (or) descending order.
2. The median of the sorted value is computed and is chosen as the pixel value for the processed image.

Example 5.5 Compute the median value of the marked pixel shown in Fig. 5.40 using a 3×3 mask.

$$\begin{bmatrix} 1 & 5 & 7 \\ 2 & 4 & 6 \\ 3 & 2 & 1 \end{bmatrix}$$

Fig. 5.40 Data for Example 5.5

Solution The median value of the marked pixel is computed as follows:

Step 1 First, the pixel values are arranged in ascending order as follows:

1 1 2 2 3 4 5 6 7

Step 2 The median value of the ordered pixel is computed as follows:

X X Z Z 3 A S K 7

The median value is computed to be 3. Then, the original pixel value of 4 will be replaced by the computed median value of 3.

$$\begin{bmatrix} 1 & 5 & 7 \\ 2 & 4 & 6 \\ 3 & 2 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 5 & 7 \\ 2 & 3 & 6 \\ 3 & 2 & 1 \end{bmatrix}$$

Original image data

After median filtering

When median filters are applied to an image, the pixel values which are very different from their neighbouring pixels will be eliminated. By eliminating the effect of such odd pixels, the values are assigned to the pixels that are representative of the values of the typical neighbouring pixels in the original image. This fact is illustrated in Example 5.6.

Example 5.6 Compute the median value of the marked pixels shown in Fig. 5.41 using a 3×3 mask.

$$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & 128 & 24 & 172 & 26 & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$$

Fig. 5.41 Input image

Solution Here the goal is to compute the median values of the marked pixels and replace the pixels 128, 24, 172 and 26 by their median values of the neighbourhood defined by the mask. The mask to be used is a 3×3 mask.

Step 1 To compute the median value of the marked pixel 128

18	22	33	25	32	24
34	128	24	172	26	23
22	19	32	31	28	26

Step 1a The eight neighbours of the marked pixel '128' are now arranged in ascending order as follows:

18 19 22 22 24 32 33 34 128

Step 1b The median value is computed as 24.

18 19 22 24 26 32 33 34 128

Step 2 To Compute the median value of the marked pixel 24

Step 2a Arrange the pixels in the neighbourhood (shown by dotted lines) in the ascending order.

18	22	33	25	32	24
34	128	24	172	26	23
22	19	32	31	28	26

After arranging the pixels in the neighbourhood in the ascending order, the values are displayed as follows:

19 22 24 25 31 32 33 128 174

Step 2b The median value is computed to be 31.

19 22 24 25 31 32 33 128 174

Step 3 To compute the median value of the marked pixel 172

Step 3a Arrange the pixels in the neighbourhood of 172 in the ascending order.

18	22	33	25	32	24
34	128	24	172	26	23
22	19	32	31	28	26

After arranging the pixel values in the neighbourhood in the ascending order, we have

24 25 26 28 31 32 33 172

Step 3b The median value is computed as 31.

24 25 26 28 31 32 33 172

Step 4 To compute the median value of the marked pixel 26

Step 4a Arrange the pixels in the neighbourhood of 26 in the ascending order.

18	22	33	25	32	24
34	128	24	172	26	23
22	19	32	31	28	26

After arranging the pixel values in the neighbourhood in the ascending order, we have

23 24 25 26 26 28 31 32 172

Step 4b The median value is computed as 26.

$$23 \ 24 \ 25 \ 26 \ 27 \ 28 \ 29 \ 30 \ 31 \ 32 \ 33$$

The original pixel values and the values replaced by their median are shown side by side below:

Original pixel value	Median value
$\begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & 128 & 24 & 172 & 26 & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$	$\longrightarrow \begin{bmatrix} 18 & 22 & 33 & 25 & 32 & 24 \\ 34 & 24 & 31 & 31 & 26 & 23 \\ 22 & 19 & 32 & 31 & 28 & 26 \end{bmatrix}$

From the above illustration it is clear that the pixel value '128' is replaced by the median value 24 and the pixel value '172' is replaced by the median value 31. Here, the values 128 and 172 are entirely different from their neighbouring pixels. When we take the median value, the pixel values which are very different from their neighbouring pixels are replaced by a value equal to the neighbouring pixel value. Hence, a median filter is capable of reducing salt-and-pepper noise. Here, salt corresponds to the maximum gray value (white) and pepper corresponds to the minimum gray value (black). Random occurrence of black and white pixels in an image is generally termed 'salt-and-pepper' noise. A median filter is an effective tool to minimise salt-and-pepper noise.

Example 5.7 *Read an image, then corrupt the image using 'salt-and-pepper' noise. Now apply a 3×3 box filter, a 5×5 box filter and a median filter to the corrupted image and comment on the result obtained.*

Solution The MATLAB code that reads the input image, corrupts it by salt-and-pepper noise, filters it using a 3×3 , a 5×5 box filter and a 3×3 and 5×5 median filter is shown in Fig. 5.42. The output of the MATLAB code is shown in Fig. 5.43.

```

clc
clear all
close all
a=imread('horse.jpg');
%Addition of salt and pepper noise
b=imnoise(a,'salt & pepper',0.1);
%Defining the box and median filters
h1=1/9*ones(3,3);
h2=1/25*ones(5,5);
c1=conv2(b,h1,'same');
c2=conv2(b,h2,'same');
c3=medfilt2(b,[3 3]);
c4=medfilt2(b,[5 5]);
subplot(3,2,1),imshow(a),title('Original image')
subplot(3,2,2),imshow(b),title('Salt & pepper noise')
subplot(3,2,3),imshow(uint8(c1)),title('3 x 3 smoothing')
subplot(3,2,4),imshow(uint8(c2)),title('5 x 5 smoothing')
subplot(3,2,5),imshow(uint8(c3)),title('3x 3 Median filter')
subplot(3,2,6),imshow(uint8(c4)),title('5 x 5 Median filter')

```

Fig. 5.42 MATLAB code to perform box and median-filtering operation

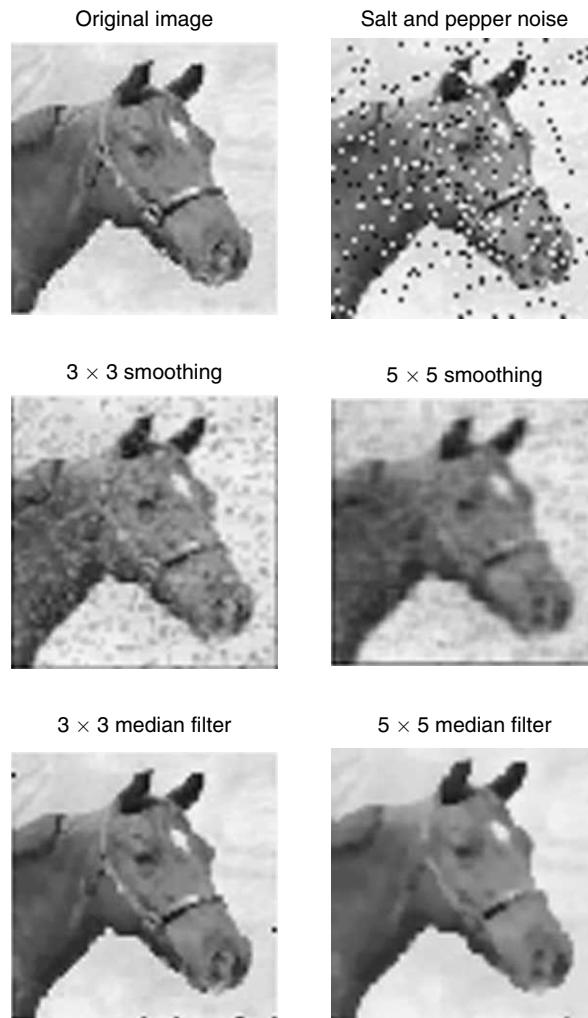


Fig. 5.43 Output of the box and median-filtered image

5.10 SPATIAL DOMAIN HIGH-PASS FILTERING OR IMAGE SHARPENING

The main aim of image sharpening is to highlight fine details in the image. Image sharpening is used to enhance the high-frequency components. The spatial filter or spatial mask which performs image sharpening is given below:

The spatial mask which performs image sharpening is given as $\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

The sum of all the weights is zero; this implies that the resulting signal will have zero dc value.

5.10.1 High-boost Filtering

A high-boost filter is also known as a high-frequency emphasis filter. A high-boost filter is used to retain some of the low-frequency components to aid in the interpretation of an image. In high-boost filtering, the input image $f(m, n)$ is multiplied by an amplification factor A before subtracting the low-pass image. Thus, the high-boost filter expression becomes

$$\text{High boost} = A \times f(m, n) - \text{low pass} \quad (5.11)$$

Adding and subtracting 1 with the gain factor, we get

$$\text{High boost} = (A-1) \times f(m, n) + f(m, n) - \text{low pass} \quad (5.12)$$

$$\text{But } f(m, n) - \text{low pass} = \text{high pass} \quad (5.13)$$

Substituting Eq. (5.13) in Eq. (5.12), we get

$$\text{high boost} = (A-1) \times f(m, n) + \text{high pass} \quad (5.14)$$

5.10.2 Unsharp Masking

Unsharp masking is one of the techniques typically used for edge enhancement. In this approach, a smoothed version of the image is subtracted from the original image; hence, tipping the image balance towards the sharper content of the image. The procedure to perform unsharp masking is given below:

1. Blur filter the image.
2. Subtract the result obtained from Step 1 from the original image.
3. Multiply the result obtained in Step 2 by some weighting fraction.
4. Add the result obtained in Step 3 to the original image.

Mathematically, the unsharp masking operation is given by

$$f'(m, n) = f(m, n) + \alpha [f(m, n) - \bar{f}(m, n)] \quad (5.15)$$

where $f(m, n)$ is the original image,

$\bar{f}(m, n)$ is the blurred version of the original image,

α is the weighting fraction, and

$f'(m, n)$ is the sharpened result.

The MATLAB code that performs unsharp masking is shown in Fig. 5.44 and the corresponding output is shown in Fig. 5.45.

Original image



Unsharp mask



```
clc
clear all
close all
a=imread('babylephant.jpg');
h=fspecial('unsharp');
b=imfilter(a,h);
imshow(a),title('original image')
Figure,imshow(b),title('Unsharp mask')
```

Fig. 5.44 MATLAB code to perform unsharp masking

Fig. 5.45 Result of unsharp masking

5.11 BIT-PLANE SLICING

The gray level of each pixel in a digital image is stored as one or more bytes in a computer. For an 8-bit image, 0 is encoded as 0 0 0 0 0 0 0 0, and 255 is encoded as 1 1 1 1 1 1 1 1. Any number between 0 and 255 is encoded as one byte. The bit in the far left side is referred as the Most Significant Bit (MSB), because a change in that bit would significantly change the value encoded by the byte. The bit in the far right is referred as the Least Significant Bit (LSB), because a change in this bit does not change the encoded gray value much. The bit-plane representation of an eight-bit digital image is shown in Fig. 5.46.

Bit-plane slicing is a method of representing an image with one or more bits of the byte used for each pixel. One can use only the MSB to represent a pixel, which reduces the original gray level to a binary image. The three main goals of bit plane slicing are

- (i) Converting a gray level image to a binary image
- (ii) Representing an image with fewer bits and compressing the image to a smaller size
- (iii) Enhancing the image by focussing

MATLAB Example 3 Bit-plane slicing Read an eight-bit image and extract the eight bit planes in the image.

Solution The MATLAB code to extract the eight planes in the image is given in Fig. 5.48 and the corresponding output is shown in Fig. 5.47.

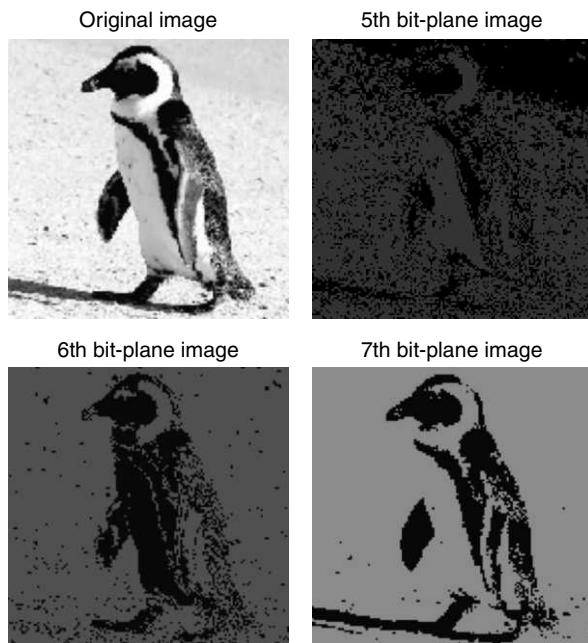


Fig. 5.47 Results of bitplane slicing

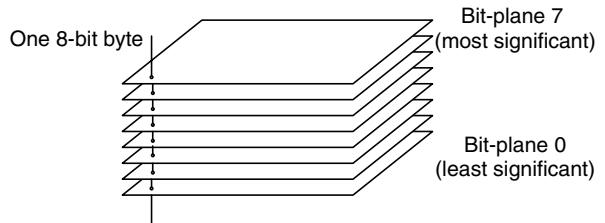


Fig. 5.46 Bit-plane representation of a digital image

```

%This code is used to extract the bitplanes in the image
a=imread('penguin.jpg');
[m n]=size(a);
%To extract 7th bit plane
for i=1:m,
    for j=1:n,
        b7(i,j)=bitand(a(i,j),128);
    end
end
%To extract 6th bit plane
for i=1:m,
    for j=1:n,
        b6(i,j)=bitand(a(i,j),64);
    end
end
%To extract 5th bit plane
for i=1:m,
    for j=1:n,
        b5(i,j)=bitand(a(i,j),32);
    end
end
%To extract 4th bit plane
for i=1:m,
    for j=1:n,
        b4(i,j)=bitand(a(i,j),16);
    end
end
%To extract 3rd bit plane
for i=1:m,
    for j=1:n,
        b3(i,j)=bitand(a(i,j),8);
    end
end
%To extract 2nd bit plane
for i=1:m,
    for j=1:n,
        b2(i,j)=bitand(a(i,j),4);
    end
end
%To extract 1st bit plane
for i=1:m,
    for j=1:n,
        b1(i,j)=bitand(a(i,j),2);
    end
end
%To extract 0th bit plane
for i=1:m,
    for j=1:n,
        b0(i,j)=bitand(a(i,j),1);
    end
end
%Code to show the resultant images
subplot(2,2,1),imshow(a),title('original image'),
subplot(2,2,2),imshow(b5),title('5th bitplane image'),
subplot(2,2,3),imshow(b6),title('6th bitplane image'),
subplot(2,2,4),imshow(b7),title('7th bitplane image'),

```

Fig. 5.48 MATLAB code to perform bit-plane slicing

Only the bit planes 5,6 and 7 are shown in the figure. The lower order bit planes will be completely black; hence it is not shown in the figure.

MATLAB Example 4 *Read an eight-bit image, set any of the bit planes 0 to 7 to zero in a user-defined manner and reconstruct the image. Observe the impact of zeroing the least significant and most significant bit planes.*

Solution The MATLAB codes that sets any of the bit planes from 0 to 7 in a user-defined manner is given in Fig. 5.49 and the corresponding outputs are shown in Fig. 5.50.

```
a=imread('penguin.jpg');
[m n]=size(a);
n1=input('enter the bitplane No
(8,7,6,5,4,3,2,1) that to be removed');
s=255-(2^(n1-1));
for i=1:m,
for j=1:n,
Out_I(i,j)=bitand(a(i,j),s);
end
end
figure,imshow(uint8(Out_I));
```

Fig. 5.49 MATLAB code to remove specific bit plane

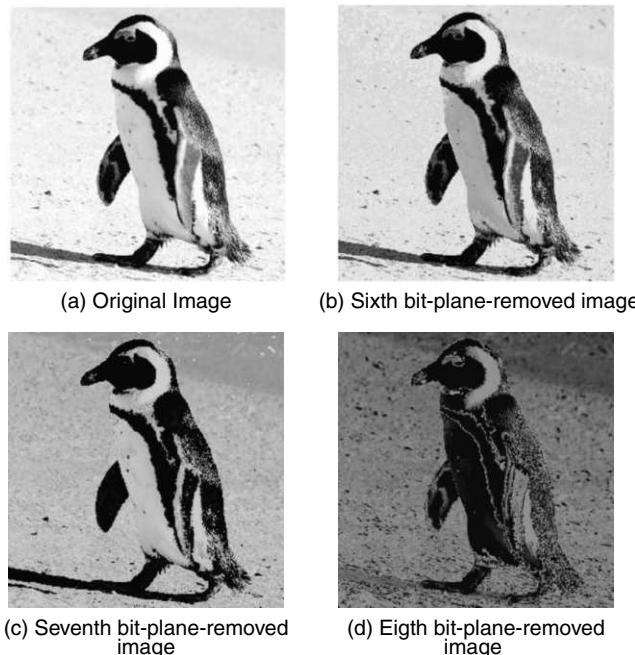


Fig. 5.50 Result of MATLAB code given in Fig. 5.49

5.12 IMAGE ENHANCEMENT IN THE FREQUENCY DOMAIN

Frequency refers to the rate of repetition of some periodic event. In image processing, spatial frequency refers to the variation of image brightness with its position in space. A varying signal can be transformed into a series of simple periodic variations. The Fourier transform decomposes a signal into a set of sine waves of different characteristics like frequency and phase. If we compute the Fourier Transform of an image, and then immediately inverse transform the result, we can regain the same image because a Fourier transform is perfectly reversible. On the other hand, if we multiply each element of the Fourier coefficient by a suitably chosen weighting function then we can accentuate certain frequency components and attenuate others. The corresponding changes in the spatial form can be seen after an inverse transform is computed. This selective enhancement or suppression of frequency components is termed *Fourier filtering* or *frequency domain filtering*. The spatial representation of image data describes the adjacency relationship between the pixels. On the other hand, the frequency domain representation clusters the image data according to their frequency distribution. In frequency domain filtering, the image data is dissected into various spectral bands, where each band depicts a specific range of details within the image. The process of selective frequency inclusion or exclusion is termed frequency domain filtering.

It is possible to perform filtering in the frequency domain by specifying the frequencies that should be kept and the frequencies that should be discarded. Spatial domain filtering is accomplished by convolving the image with a filter kernel. We know that

$$\text{Convolution in spatial domain} = \text{Multiplication in the frequency domain}$$

If filtering in spatial domain is done by convolving the input image $f(m, n)$ with the filter kernel $h(m, n)$,

$$\text{Filtering in spatial domain} = f(m, n) * h(m, n)$$

In the frequency domain, filtering corresponds to the multiplication of the image spectrum by the Fourier transform of the filter kernel, which is referred to as the frequency response of the filter

$$\text{Filtering in the frequency domain} = F(k, l) \times H(k, l)$$

Here, $F(k, l)$ is the spectrum of the input image and $H(k, l)$ is the spectrum of the filter kernel. Thus, frequency domain filtering is accomplished by taking the Fourier transform of the image and the Fourier transform of the kernel, multiplying the two Fourier transforms, and taking the inverse Fourier transform of the result. The multiplication of the Fourier transforms need to be carried out point by point. This point-by-point multiplication requires that the Fourier transforms of the image and the kernel themselves have the same dimensions. As convolution kernels are commonly much smaller than the images they are used to filter, it is required to zero pad out the kernel to the size of the image to accomplish this process.

5.12.1 Low-pass Filtering in Frequency Domain

The low-pass filter mask can be broadly classified into two types: (i) non-separable filter mask, and (ii) separable filter mask.

(i) **Non-separable Filter Transfer Function** The non-separable filter transfer function is given by

$$H(k, l) = \begin{cases} 1, & \text{for } \sqrt{k^2 + l^2} \leq D_0 \\ 0, & \text{otherwise} \end{cases} \quad (5.16)$$

```

1 %Low pass filter transfer function
2 [a b]=freqspace(256,'meshgrid');
3 H=zeros(256,256);
4 d=sqrt(a.^2+b.^2)<0.5;
5 H(d)=1;
6 imshow(H),title('LPF transfer fn.')

```

Fig. 5.51 MATLAB code for a low-pass filter transfer function

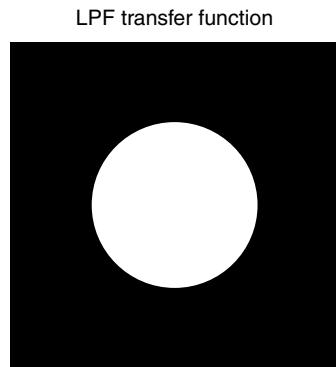


Fig. 5.52 Low-pass filter transfer function

Here, D_0 is the cut-off frequency of the low-pass filter. The cut-off frequency determines the amount of frequency components passed by the filter. The smaller the value of D_0 , the greater the number of image components eliminated by the filter. The value of D_0 is chosen in such a way that the components of interest are passed through.

The MATLAB code to generate the low-pass filter transfer function and the corresponding output are shown in Figs 5.51 and 5.52 respectively.

(ii) Separable Low-pass Filter Transfer Function The expression for a separable low-pass filter transfer function is given below:

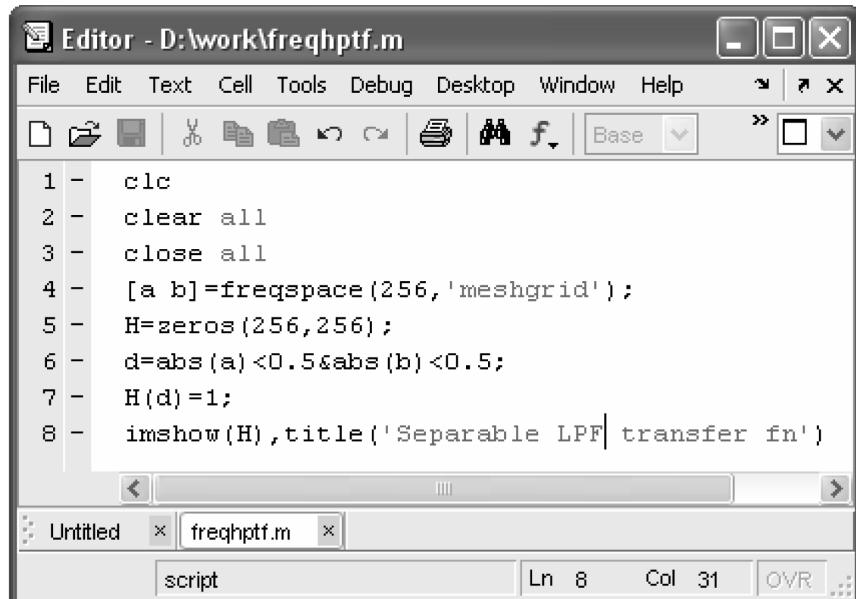
$$H(k, l) = \begin{cases} 1, & \text{for } k \leq D_k \text{ and } l \leq D_l \\ 0, & \text{otherwise} \end{cases} \quad (5.17)$$

The MATLAB code and the corresponding output are shown in Fig. 5.53 and Fig. 5.54 respectively.

Butterworth Low-pass Filter The transfer function of a two-dimensional Butterworth low-pass filter is given by

$$H(k, l) = \frac{1}{1 + \left[\frac{\sqrt{k^2 + l^2}}{D_0} \right]^{2n}} \quad (5.18)$$

In the above expression, n refers to the filter order and D_0 represents the cut-off frequency.



```

1 - clc
2 - clear all
3 - close all
4 - [a b]=freqspace(256,'meshgrid');
5 - H=zeros(256,256);
6 - d=abs(a)<0.5&abs(b)<0.5;
7 - H(d)=1;
8 - imshow(H),title('Separable LPF transfer fn')

```

Fig. 5.53 MATLAB code to generate a separable low-pass filter transfer function

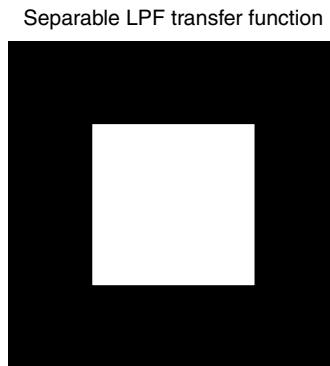


Fig. 5.54 Separable LPF transfer function

MATLAB Example 5 Write a MATLAB program that performs a two-dimensional Butterworth low-pass filter of the given image for two different cut-off frequencies.

Solution: The MATLAB code to perform a two-dimensional Butterworth low-pass filter is shown in Fig. 5.55.

The outputs corresponding to the MATLAB code for different cut-off frequencies are shown in Fig. 5.56. The 2D and 3D views of the transfer function for the three cut-off frequencies $D_0 = 10$, $D_0 = 20$ and $D_0 = 30$ are illustrated in Fig. 5.57 (a) to (f) respectively.

```
%This code is used to Butterworth lowpass filter
close all;
clear all;
clc;
im=imread('d:\work\hibiscus.tif');
fc=20;%Cutoff frequency
n=1;
[co,ro] = size(im);
cx = round(co/2); % find the centre of the image
cy = round (ro/2);
imf=fftshift(fft2(im));
H=zeros(co,ro);
for i = 1 : co
    for j =1 : ro
        d = (i-cx).^2 + (j-cy).^ 2;
        H(i,j) = 1/(1+((d/fc/fc).^(2*n)));
    end;
end;
outf = imf .* H;
out = abs(ifft2(outf));
imshow(im),title('Original Image'),
figure,imshow(uint8(out)),title('Lowpass Filterd Image')
```

Fig. 5.55 MATLAB code to perform a two-dimensional Butterworth low-pass filter

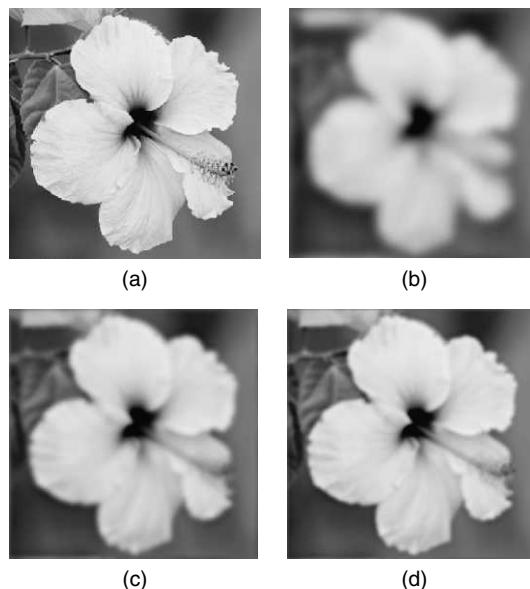


Fig. 5.56 (a) Original image (b) Low-pass filtered image with $D_0 = 10$ (c) Low-pass filtered image with $D_0 = 20$ (d) Low-pass filtered image with $D_0 = 30$

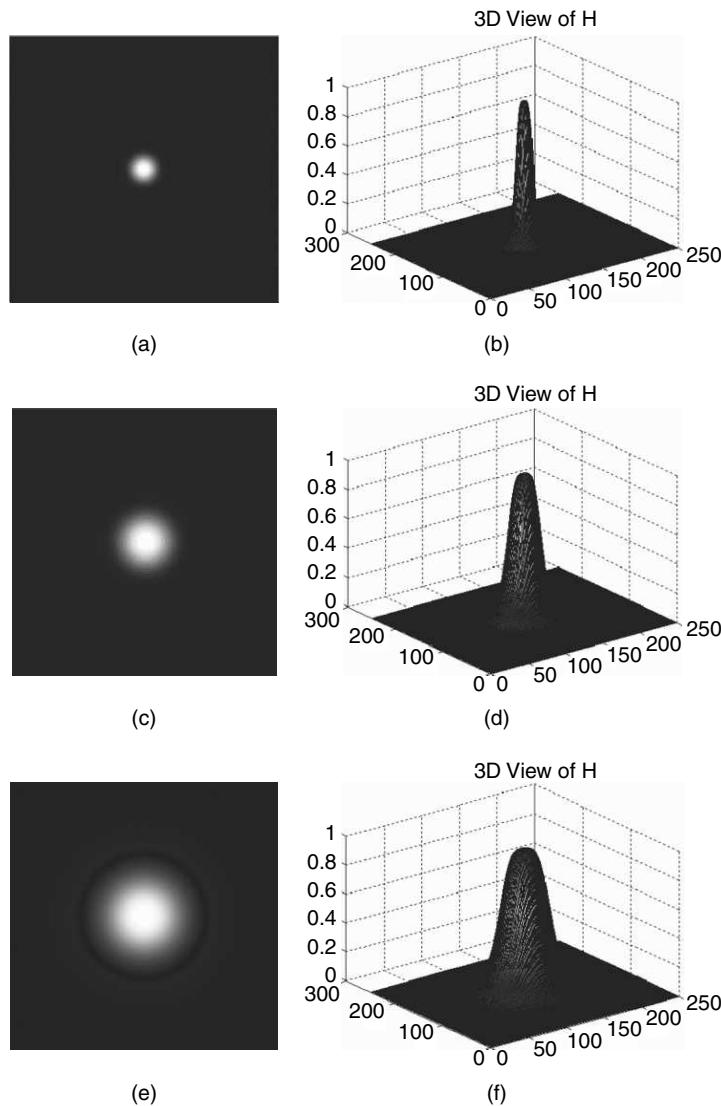


Fig. 5.57 2D and 3D views of the transfer function

5.12.2 High-pass Filter in Frequency Domain

The low-pass filters and high-pass filters are complementary to each other. The 2D ideal high-pass filter is one whose transfer function satisfies the relation

$$H(k, l) = \begin{cases} 0, & \text{if } D(k, l) \leq D_0 \\ 1, & \text{if } D(k, l) > D_0 \end{cases} \quad (5.19)$$

where D_0 is the cut-off distance measured from the origin of the frequency plane. The cross-section of an ideal high-pass filter is shown in Fig. 5.58. The ideal high-pass filter is not practically realisable.

5.12.3 Butterworth High-pass Filter

The transfer function of a two-dimensional Butterworth filter is given by

$$H(k, l) = \frac{1}{1 + \left[\frac{D_0}{\sqrt{k^2 + l^2}} \right]^{2n}} \quad (5.20)$$

Where n is the order of the filter, D_0 is the cut-off frequency.

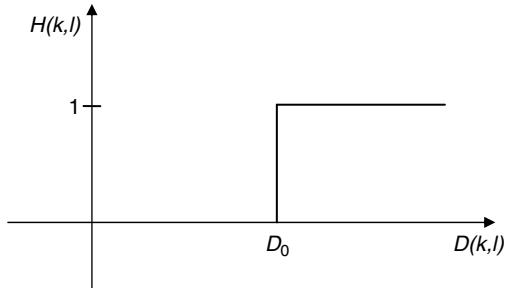


Fig. 5.58 Cross-section of an ideal high-pass filter

MATLAB Example 6 Write a MATLAB program that performs a two-dimensional Butterworth high-pass filter of the given image for two different cut-off frequencies.

Solution: The MATLAB code to perform a two-dimensional Butterworth high-pass filter is shown in Fig. 5.59.

```
%This code is used to Butterworth highpass filter
close all;
clear all;
clc;
im=imread('d:\work\hibiscus.tif');
fc=40;
n=1;
[co,ro] = size(im);
cx = round(co/2); % find the centre of the image
cy = round (ro/2);
imf=fftshift(fft2(im));
H=zeros(co,ro);
for i = 1 : co
    for j =1 : ro
        d = (i-cx).^2 + (j-cy).^ 2;
        if d ~= 0
            H(i,j) = 1/(1+((fc*fc/d).^(2*n)));
        end;
    end;
end;
outf = imf .* H;
out = abs(ifft2(outf));
imshow(im),title('Original Image'),figure,imshow(uint8(out)),title
('Highpass Filterd Image')
figure,imshow(H),title('2D View of H'),figure,surf(H),
title('3D View of H')
```

Fig. 5.59 MATLAB code to perform a Butterworth high-pass filter

The output of the MATLAB code for two different cut-off frequencies $D_0 = 40$ and $D_0 = 80$ are shown in Fig. 5.60.

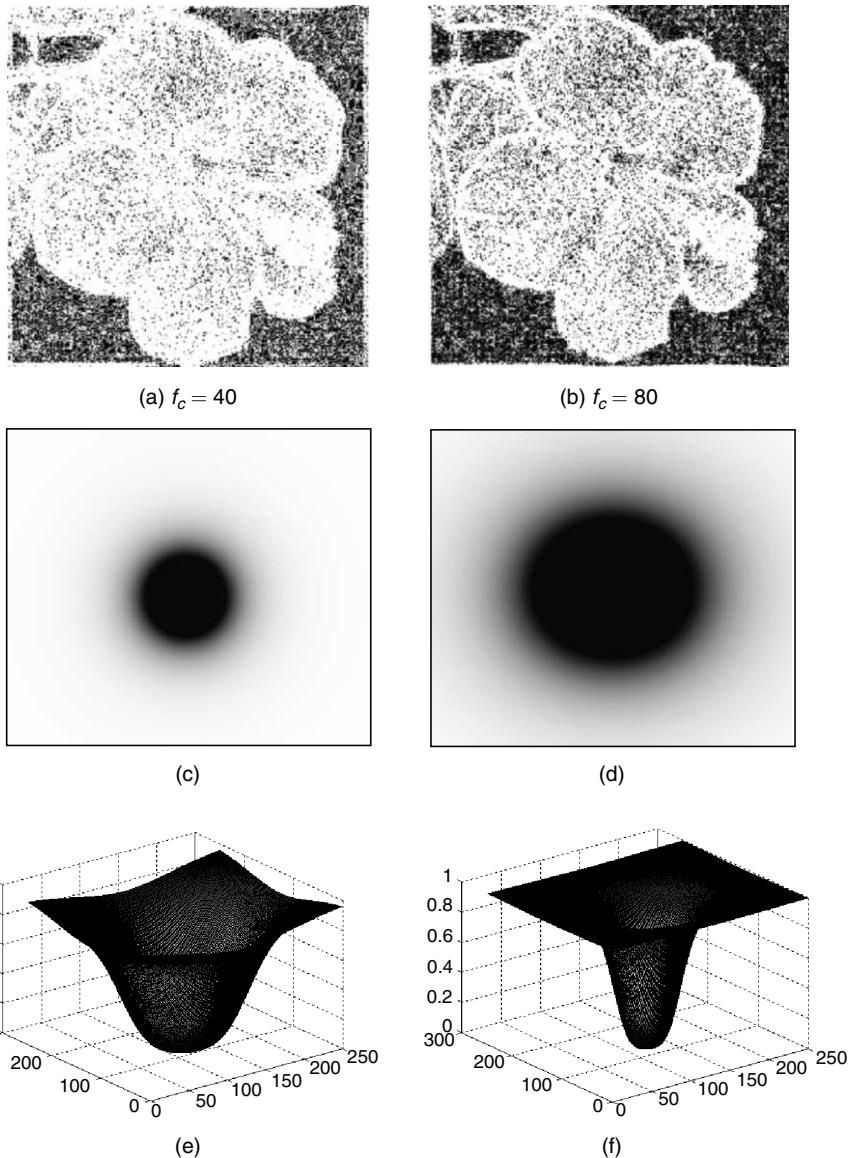


Fig. 5.60 (a) and (b) High-pass filtered images for $D_0 = 40$ and $D_0 = 80$ (c) and (d) Two-dimensional views of the transfer function for $D_0 = 40$ and $D_0 = 80$ (e) and (f) Three-dimensional views of the transfer function for $D_0 = 40$ and $D_0 = 80$

5.12.4 Low-pass and High-pass Gaussian Filter

The Gaussian filter is popular for the absence of ringing and noise leakage artifacts. The transfer function of a two-dimensional Gaussian low-pass filter is given by

$$h(m, n) = \frac{1}{2\pi\sigma^2} e^{-\frac{(m^2+n^2)}{2\sigma^2}} \quad (5.21)$$

The transfer function is separable which can be written as

$$h(m, n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{m^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n^2}{2\sigma^2}} \quad (5.22)$$

The above equation implies that the Gaussian filter can be implemented by applying two 1D Gaussian filters. The true Gaussian filter has infinite support, hence it is not practical to implement. The MATLAB code to implement a Gaussian low-pass filter and a high-pass filter is given in Figs 5.61 and 5.62 respectively.

Figures 5.63 (b) and (c) show a Gaussian low-pass filter for two different cut-off frequencies.

```
%This code is used to gaussian lowpass filter
close all;
clear all;
clc;
im=imread('temp.tif');
fc=10;
imf = fftshift(fft2(im));
[co,ro]=size(im);
out = zeros(co,ro);
cx = round(co/2); % find the centre of the image
cy = round (ro/2);
H = zeros(co,ro);
for i = 1 : co
    for j = 1 : ro
        d = (i-cx).^2 + (j-cy).^2;
        H(i,j) = exp(-d/2/fc/fc);
    end;
end;
outf= imf.*H;
out=abs(ifft2(outf));
imshow(im),title('Original Image'),figure,imshow(uint8(out)),
title('GaussianLowpass Filterd Image')
figure,imshow(H),title('2D View of H'),figure,surf(H),
title('3D View of H')
```

Fig. 5.61 Gaussian low-pass filter

```
%This code is used to gaussian highpass filter
close all;
clear all;
clc;
im=imread('temp.tif');
fc=100;
imf = fftshift(fft2(im));
[co,ro]=size(im);
cx = round(co/2); % find the centre of the image
cy = round (ro/2);
H = zeros(co,ro);
for i = 1 : co
    for j = 1 : ro
        d = (i-cx).^2 + (j-cy).^2;
        H(i,j) = exp(-d/2/fc/fc);
    end;
end;
% H = gaussian_filter(co,ro, fc);
H = 1-H;
out = zeros(co,ro);
outf= imf.*H;
out=abs(ifft2(outf));
imshow(im),title('Original Image'),figure,imshow((out)),title('Filtered Image')
figure,imshow(H),title('2D View of H'),figure,surf(H),
title('3D View of H')
```

Fig. 5.62 Gaussian high-pass filter

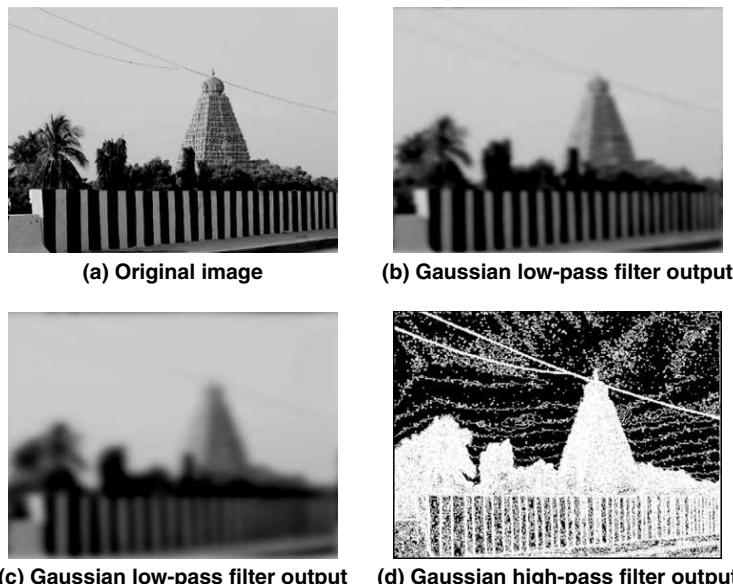


Fig. 5.63 Output of a Gaussian low-pass and high-pass filter

MATLAB Example 7 Read an input image and perform high-pass filtering in the frequency domain. That is, you are asked to perform frequency-domain high-pass filtering operation on the input image.

Solution The input image chosen is a dew drop on the tip of a blade of grass. The following steps are done to perform a frequency-domain high-pass filtering operation:

1. The Fourier transform of the input image is taken. Let it be $F(k,l)$.
2. The high-pass filter mask is defined and its Fourier transform is taken. Let it be $H(k,l)$.
3. An element-by-element multiplication of $F(k,l)$ and $H(k,l)$ is done.
4. The inverse Fourier transform of the result obtained in Step (3) is taken to get the high-pass filtered image.

The MATLAB code and the corresponding outputs are shown in Fig. 5.64 and Fig. 5.65 respectively.

```
%Frequency domain high pass filter
clc;
clear all;
close all;
a=imread('dewdrop.jpg');
[m n]=size(a);
mask=ones(m,n);
for i=150:180
    for j=210:240
        mask(i,j)=0;
    end
end
c=fft2(a);
d=fftshift(mask);
e=c.*d;
f=abs(ifft2(e));
subplot(2,2,1),imshow(a),title('original image')
subplot(2,2,2),imshow(mat2gray(f)),title('High pass filtered image')
subplot(2,2,3),imshow(mask),title('high pass filter mask')
subplot(2,2,4),imshow(d),title('Mask after fftshift operation')
```

Fig. 5.64 MATLAB code to perform frequency-domain high-pass filtering

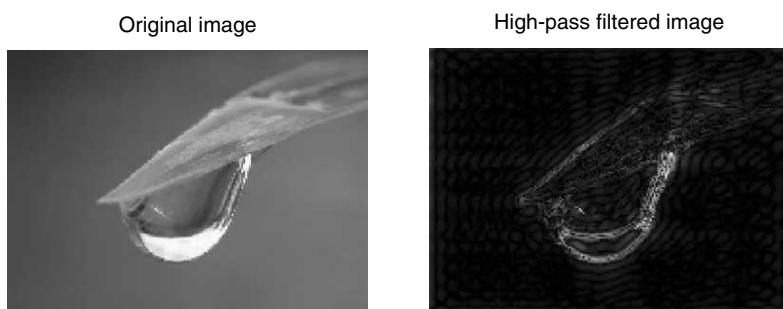


Fig. 5.65 (Continued)

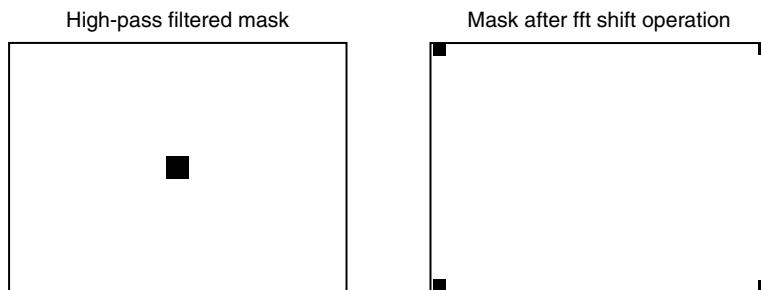


Fig. 5.65 Result of MATLAB code shown in Fig. 5.64

MATLAB Example 8 *Read an input image and perform a frequency-domain low-pass filter on the input image.*

Solution The same image of a dew drop is read and a frequency-domain low-pass filter is done on the input image. The MATLAB code and the corresponding outputs are shown in Fig. 5.66 and Fig. 5.67 respectively.

```
%Frequency domain Low pass filter
clc;
clear all;
close all;
a=imread('dewdrop.jpg');
[m n]=size(a);
mask=zeros(m, n);
for i=150:180
    for j=210:240
        mask(i, j)=1;
    end
end
c=fftshift(mask);
b=fft2(a);
d=b.*c;
e=abs(ifft2(d));
subplot(2,2,1),imshow(a),title('original image')
subplot(2,2,2),imshow(uint8(e)),title('Low pass filtered image')
subplot(2,2,3),imshow(mask),title('Low pass filter mask')
```

Fig. 5.66 MATLAB code to perform frequency domain low-pass filter

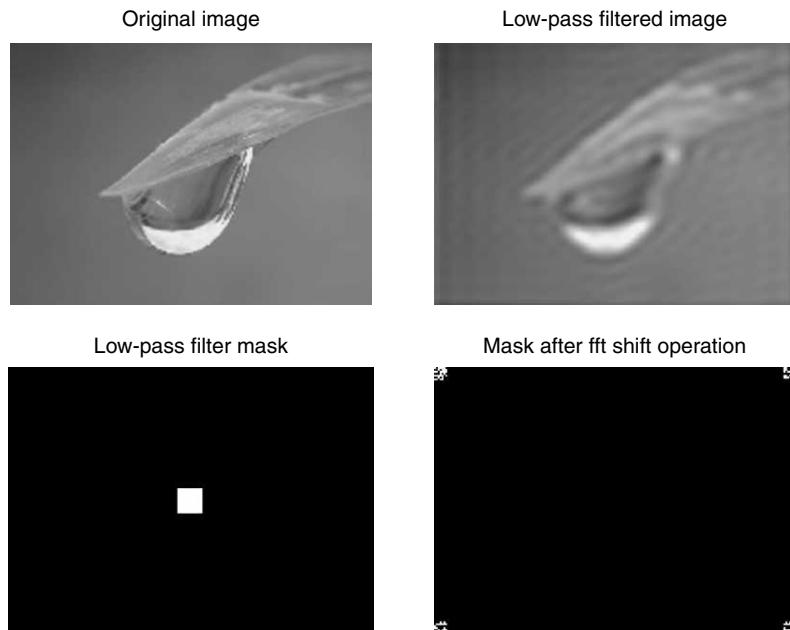


Fig. 5.67 Result of MATLAB code shown in Fig. 5.66

From Fig. 5.67, it is obvious that the effect of low-pass filtering of the input image of the dew drop is a smearing of edges. Thus, a low-pass filter blurs the edges of the image.

MATLAB Example 9 *Read an input image and perform a frequency-domain band-pass filter on the input image.*

Solution The MATLAB code to perform a frequency-domain band-pass filtering of the input ‘dew drop’ image is shown in Fig. 5.68 and the corresponding result is shown in Fig. 5.69. From the figure, it is clear that the edges are blurred, but it is not so intense as in the case of low-pass filtering.

```
%Frequency domain band pass filter
clc;
clear all;
close all;
a=imread('dewdrop.jpg');
[m n]=size(a);
mask=zeros(m,n);
for i=130:220
    for j=200:300
        mask(i,j)=1;
    end
end
```

Fig. 5.68 (Continued)

```

for i=150:190
    for j=230:270
        mask(i,j)=0;
    end
end
b=fftshift(mask);
b=fft2(a);
c=fftshift(mask);
d=b.*c;
e=abs(ifft2(d));
subplot(2,2,1),imshow(a),title('original image')
subplot(2,2,2),imshow(uint8(e)),title('Band pass filtered image')
subplot(2,2,3),imshow(mask),title('Band pass filter mask')
subplot(2,2,4),imshow(c),title('Mask after fftshift operation')

```

Fig. 5.68 MATLAB code to perform frequency-domain band-pass filter

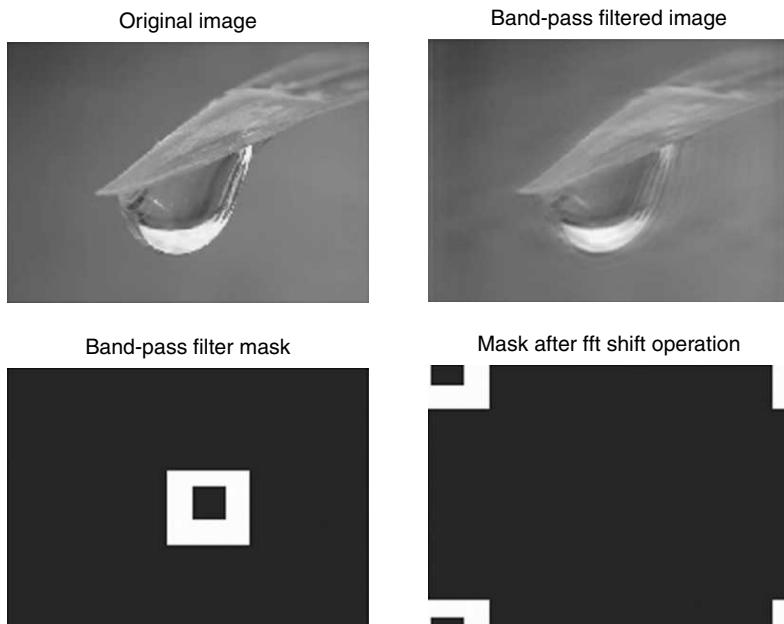


Fig. 5.69 Results of MATLAB code shown in Fig. 5.68

MATLAB Example 10 *Read an image and perform band-stop filtering of the input image.*

Solution The MATLAB code that performs the band-stop filtering of the input image and the corresponding output are shown in Fig. 5.70 and Fig. 5.71 respectively.

The result of the MATLAB code shown in Fig. 5.70 is shown in Fig. 5.71.

```
%Frequency domain band stop filter
clc;
clear all;
close all;
a=imread('dewdrop.jpg');
[m n]=size(a);
mask=ones(m,n);
for i=130:200
    for j=210:280
        mask(i,j)=0;
    end
end
for i=150:175
    for j=225:260
        mask(i,j)=1;
    end
end
b=fftshift(mask);
imshow(mask)
b=fft2(a);
c=fftshift(mask);
d=b.*c;
e=abs(ifft2(d));
subplot(2,2,1),imshow(a),title('original image')
subplot(2,2,2),imshow(uint8(e)),title('Band stop filtered image')
subplot(2,2,3),imshow(mask),title('Band stop filter mask')
subplot(2,2,4),imshow(c),title('Mask after fftshift operation')
```

Fig. 5.70 MATLAB code to perform band-stop filtering

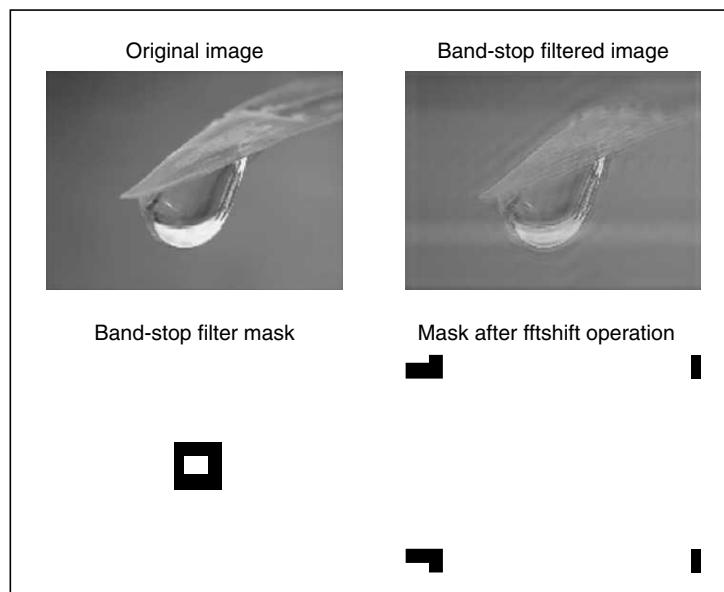


Fig. 5.71 Result of the MATLAB code shown in Fig. 5.70

5.13 HOMOMORPHIC FILTER

An image can be modeled as the product of an illumination function and the reflectance function at every point. Based on this fact, the simple model for an image is given by

$$f(n_1, n_2) = i(n_1, n_2) \times r(n_1, n_2) \quad (5.23)$$

This model is known as illumination-reflectance model. The illumination-reflectance model can be used to address the problem of improving the quality of an image that has been acquired under poor illumination conditions.

In the above equation, $f(n_1, n_2)$ represents the image, $i(n_1, n_2)$ represents the illumination component and $r(n_1, n_2)$ represents the reflectance component. For many images, the illumination is the primary contributor to the dynamic range and varies slowly in space, while the reflectance component $r(n_1, n_2)$ represents the details of the object and varies rapidly in space. If the illumination and the reflectance components have to be handled separately, the logarithm of the input function $f(n_1, n_2)$ is taken. Because $f(n_1, n_2)$ is the product of $i(n_1, n_2)$ with $r(n_1, n_2)$, the log of $f(n_1, n_2)$ separates the two components as illustrated below:

$$\ln[f(n_1, n_2)] = \ln[i(n_1, n_2)r(n_1, n_2)] = \ln[i(n_1, n_2)] + \ln[r(n_1, n_2)] \quad (5.24)$$

Taking Fourier transform on both sides, we get

$$F(k, l) = F_I(k, l) + F_R(k, l) \quad (5.25)$$

where $F_I(k, l)$ and $F_R(k, l)$ are the Fourier transform of the illumination and reflectance components respectively.

Then, the desired filter function $H(k, l)$ can be applied separately to the illumination and the reflectance component separately as shown below:

$$F(k, l) \times H(k, l) = F_I(k, l) \times H(k, l) + F_R(k, l) \times H(k, l) \quad (5.26)$$

In order to visualise the image, inverse Fourier transform followed by exponential function is applied. First, the inverse Fourier transform is applied as shown below:

$$f'(n_1, n_2) = \mathcal{I}^{-1}[F(k, l) \times H(k, l)] = \mathcal{I}^{-1}[F_I(k, l) \times H(k, l)] + \mathcal{I}^{-1}[F_R(k, l) \times H(k, l)] \quad (5.27)$$

The desired enhanced image is obtained by taking the exponential operation as given below:

$$g(n_1, n_2) = e^{f'(n_1, n_2)} \quad (5.28)$$

Here, $g(n_1, n_2)$ represents the enhanced version of the original image $f(n_1, n_2)$. The sequence of operation can be represented by a block diagram as shown in Fig. 5.72.

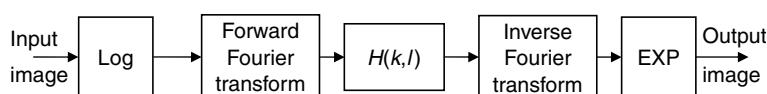


Fig. 5.72 Block diagram of homomorphic filtering

5.14 ZOOMING OPERATION

When an image has fine details, it may not be possible to examine the detail easily in a standard display on a monitor. In such cases, the zooming operation helps to view fine details in the image. An image can be expanded by the zooming operation. Zooming is equivalent to holding a magnifying glass in front of the screen. The simplest zooming operation is through ‘replication’ of pixels. The principle of replication is ‘for every pixel of the image, put the same value of the pixel in a grid of $N \times N$ pixels’. This is diagrammatically shown in Fig. 5.73.

The MATLAB code that performs zooming operation is shown in Fig. 5.74 and the corresponding output is shown in Fig. 5.75.

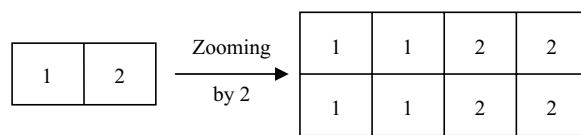


Fig. 5.73 Zooming illustration

```
%MATLAB code that performs zooming operation through pixel replication
clc;
clear all;
close all;
a=imread('dove1.jpg');
[m n]=size(a);
p=input('Enter the size you want: ');
for i=1:m % loop to extract every row
    for j=1:n % loop to extract every column
        for k=1:p % loop to control the number of replication
            b(i,(j-1)*p+k)=a(i,j);% replication of pixels in row wise
        end
    end
end
c=b;
[m n]=size(c);
for i=1:n % loop to extract every column
    for j=1:m % loop to extract every row
        for k=1:p % loop to control the number of replication
            b((j-1)*p+k,i)=c(j,i); % replication of pixel in column wise
        end
    end
end
imshow(a),title('original image')
figure,imshow(b),title('Zoomed image')
xlabel(sprintf('Zooming factor is %g',p))
```

Fig. 5.74 Zooming through pixel replication

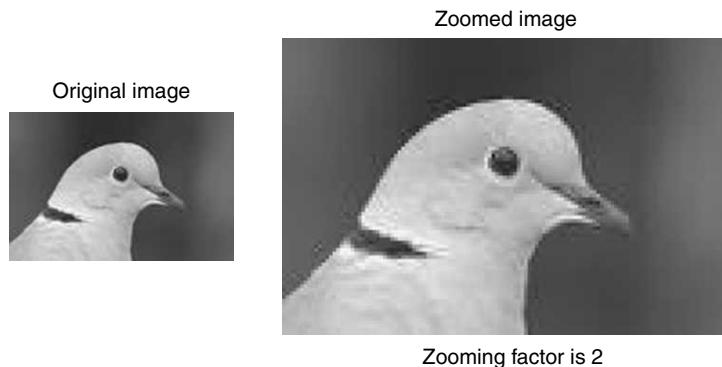


Fig. 5.75 *Output of zooming operation*

5.14.1 Drawback of Pixel Replication

The main drawback of pixel replication is ‘pixelisation’. Pixelisation can be defined as the stair-stepped appearance of a curved or angled line in digital imaging. The pixelisation effect is visible if the enlargement of the digital image is too much and the pixels become obvious. To overcome the problem of pixelisation, the adaptive zooming method is employed.

5.14.2 Zooming through Linear Interpolation

Linear interpolation is basically a first-order hold, where a straight line is first fitted in between pixels along a row. Then the pixels along each column are interpolated along a straight line.

Linear interpolation can be established through circular convolution.

Example 5.8 Consider an image $a = \begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix}$. Then, the pixel value 5 in the image a has to be replicated. The steps involved in replication are given below.

Step 1 Zero interlacing

Through zero interlacing, zeros are inserted as shown below.

$$a = \begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix}$$

Step 2 Defining the matrix H

In this problem, the size of the input matrix a is 2×2 . Hence, the matrix H is defined as $H = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

Step 3 Performing circular convolution between the input matrix a and the matrix H .

The circular convolution is performed by the following steps:

- 3a. Computation of FFT of the matrix a .
- 3b. Computation of FFT of the matrix H .

3c. Performing elementwise multiplication of the result obtained in steps 3a and 3b.

3d. Take the inverse Fourier transform of the result obtained in Step 3c.

The part of the MATLAB code and the corresponding output are shown in Fig. 5.76.

On executing the above MATLAB code, the value of the result matrix is obtained as

$$\text{result} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$$

The same concept can be extended to images and flexible zooming can be achieved. Here, flexible zooming means zooming can be performed as 1.5, 2, 2.5, etc. The MATLAB code that performs flexible zooming is shown in Fig. 5.77.

```
clc
clear all
close all
a=[5 0 ; 0 0];
h=ones(2);
%Performing circular convolution
a1=fft2(a);
h1=fft2(h);
c=a1.*h1;
result=ifft2(c)
```

Fig. 5.76 MATLAB code for Example 5.8

```
a=imread('key.jpg');
a=rgb2gray(a);
zooming_factor=input('enter the zooming factor:');
num=zooming_factor; den=1;
%Repeated multiplication to convert decimal into fraction
while (num - floor(num) ~= 0)
    num=num*2; den=den*2;
end
[m n]=size(a); s1=num*m;
re=zeros(s1,num*n);
for i=1:m,
    for j=1:n,
        k=num*(i-1);
        l=num*(j-1);
        re(k+1,l+1)=a(i,j);
    end
end
```

Fig. 5.77 (Continued)

```

end
i=1;
while (i<=(s1))
    j=1;
    while (j<=(num*n))
        x=ones (num,num);
        for p=1:num,
            for q=1:num,
                c(p,q)=re(i,j); % To extract the pixel matrix
                j=j+1;
            end
            i=i+1;j=j-num;
        end
        z=ifft2(fft2(c).*fft2(x)); % To perform the circular convolution
        using DFT and IDFT
        i=i-num;
        for p=1:num,
            for q=1:num,
                re(i,j)=z(p,q); % To generate the interpolated matrix
                j=j+1;end
            i=i+1; j=j-num;end
        i=i-num; j=j+num;end
    i=i+num; end
if (den>1) %to check whether the zooming factor is integer or not.
    m=den; [p,q]=size(re);
    a=double(re);
    for i=1:ceil(p/m),
        for j=1:ceil(q/m),
            if((m*i)<p) & ((m*j)<q)
                b(i,j)=re(m*i,m*j);
            else b(i,j)=0;
            end
        end
    end
else b=re;end
figure, imshow (uint8(b));

```

Fig. 5.77 MATLAB Program to perform flexible zooming operation

The result of flexible zooming is shown in Fig. 5.78. In the figure, the third image is the original image. The leftmost image is obtained by using the zooming factor as 0.25, and the right most image is obtained by using the zooming factor as 2.



Fig. 5.78 Result of flexible zooming

5.15 IMAGE ARITHMETIC

In image arithmetic, different arithmetic operations like image addition, image subtraction, image multiplication, image averaging and alpha blending are considered. Image addition and subtraction can place objects into and remove objects from images.

5.15.1 Image Addition

Image addition is used to create double exposure. If $f(m, n)$ and $g(m, n)$ represent two images then the addition of these two images to get the resultant image is given by

$$c(m, n) = f(m, n) + g(m, n) \quad (5.29)$$

If multiple images of a given region are available for approximately the same date and if a part of one of the images has some noise then that part can be compensated from other images available through image addition.

The concept of image addition is illustrated in Fig. 5.79.

In Fig. 5.79, there are three images (a), (b) and (c). Here, image (c) is obtained by addition of images in (a) and (b). The MATLAB code to implement image addition is given in Fig. 5.80.

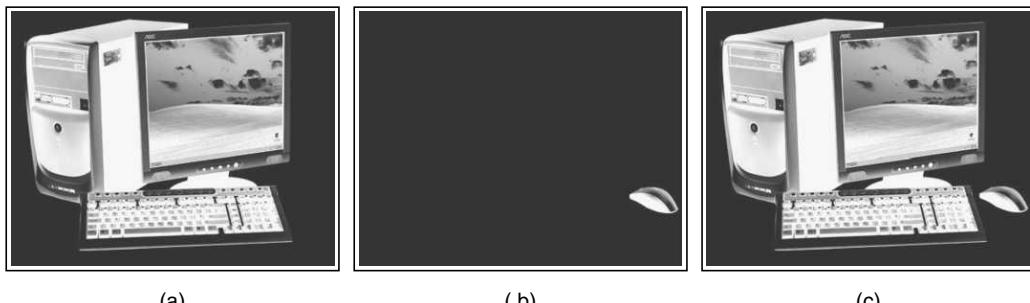


Fig. 5.79 Image addition

```

a=imread('image1.bmp');
b=imread('image2.bmp');
c=double(a)+double(b);
imshow(a),
figure,imshow(b),
figure,imshow(uint8(c))

```

Fig. 5.80 MATLAB code to implement image addition

5.15.2 Image Subtraction

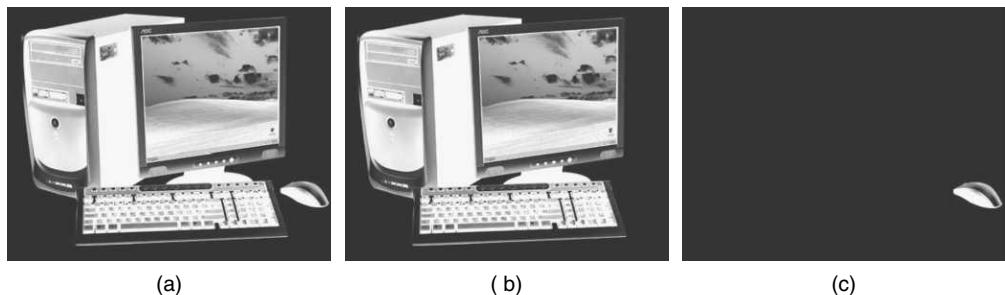
Image subtraction is used to find the changes between two images of a same scene. The mathematical representation of image subtraction is given by

$$c(m, n) = f(m, n) - g(m, n) \quad (5.30)$$

To assess the degree of change in an area, two dates of co-registered images can be used with the subtraction operation. Image subtraction can be used to remove certain features in the image.

The image subtraction is illustrated in Fig. 5.81. In Fig. 5.81, the images (a) and (b) are subtracted to get the image (c).

The MATLAB code used to perform image subtraction is shown in Fig. 5.82.

**Fig. 5.81** Image subtraction

```

a=imread('image3.bmp');
b=imread('image4.bmp');
c=double(a)-double(b);
imshow(a),figure,imshow(b),
figure,imshow(uint8(c))

```

Fig. 5.82 MATLAB code to implement image subtraction

5.15.3 Image Multiplication

Image multiplication is basically used for masking. If the analyst is interested in a part of an image then extracting that area can be done by multiplying the area by one and the rest by zero.

The MATLAB code which performs image multiplication is shown in Fig. 5.83 and the corresponding output is illustrated in Fig. 5.84.

From Fig. 5.84, it is obvious that the multiplication operation is used to extract specific information from the image. From the figure, it is obvious that the coin of interest is highlighted.

```
b=imread('eight.tif');
a=0.35+zeros(242,308);
[m n]=size(a);
for i=20:98
    for j=85:164
        a(i,j)=1;
    end
end
c=double(b).*a;
imshow(a),title('Mask'),figure,imshow(b),title('Original image'),
figure,imshow(uint8(c)),title('Resultant image')
```

Fig. 5.83 MATLAB code to implement image multiplication

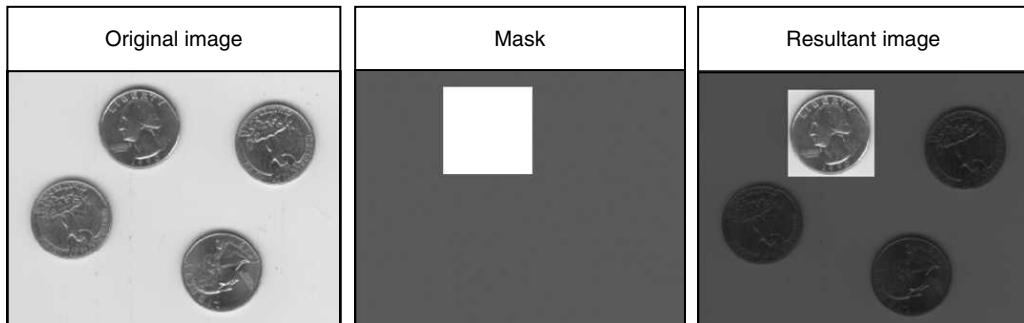


Fig. 5.84 Image multiplication

5.15.4 Image Division

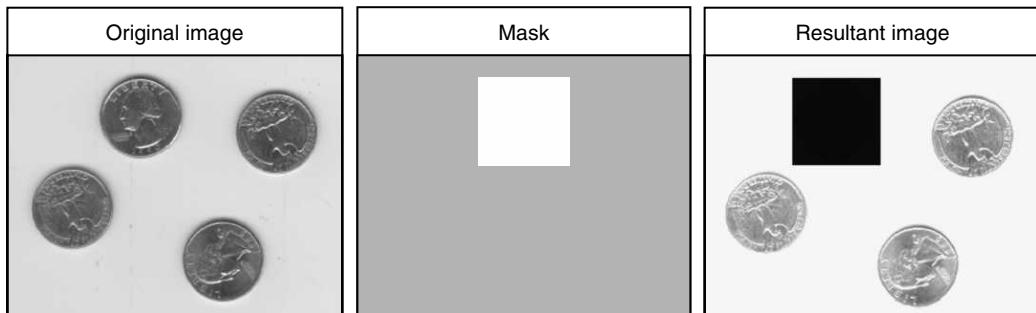
Dividing the pixels in one image by the corresponding pixels in a second image is commonly used in transformation. The MATLAB code which performs the division operation is shown in Fig. 5.85 and the corresponding output is illustrated in Fig. 5.86.

From Fig. 5.86, it is obvious that the result of image division is just opposite to that of image multiplication. The application of arithmetic operation is summarised in Table 5.1.

```

b=imread('eight.tif');
a=1+zeros(242,308);
[m n]=size(a);
for i=20:98
    for j=85:164
        a(i,j)=255;
    end
end
c=double(b)./a;
imshow(a),title('Mask'),figure,imshow(b),title('Original image'),
figure,imshow(uint8(c)),title('Resultant image')

```

Fig. 5.85 MATLAB code to implement image Division code**Fig. 5.86** Image Division**Table 5.1** Applications of arithmetic operation

S.No	Arithmetric operation	Application
1	Addition	Superimposing one image upon another
2	Subtraction	Change detection
3	Multiplication	Masking operation which can be used for background suppression
4	Division	Background Suppression

5.15.5 Alpha Blending

Alpha blending refers to addition of two images, each with 0 to 1 fractional masking weights. Alpha blending is useful for transparency and compositing.

MATLAB Example 11: Blending of two images Write a MATLAB code that will read two images and blend the two images in a user-defined manner.

Solution Here, we have taken two famous test images ‘cameraman’ and ‘Lena’. The cameraman image is read in the variable a , and the Lena image is read in the variable b . Then, the two images are blended and stored in the variable ‘ c ’ using the formula

$$c = (1 - \alpha)a + \alpha b \quad (5.31)$$

Here, α is the user-defined value. Different values of the variable α will increase the emphasis on the image a or image b . The MATLAB code that implements alpha blending is shown in Fig. 5.87 and the corresponding output is illustrated in Fig. 5.88.

```
% Image blending
clc
clear all
close all
a=imread('cameraman.tif');
b=imread('lena.bmp');
b=imresize(b,[256 256]);
[m n]=size(a);
alpha1=input('Enter the value of alpha:');
alpha2=input('Enter the value of alpha:');
alpha3=input('Enter the value of alpha:');
alpha4=input('Enter the value of alpha:');
for i=1:m
    for j=1:n
        c1(i,j)=(1-alpha1)*a(i,j)+alpha1*b(i,j);
        c2(i,j)=(1-alpha2)*a(i,j)+alpha2*b(i,j);
        c3(i,j)=(1-alpha3)*a(i,j)+alpha3*b(i,j);
        c4(i,j)=(1-alpha4)*a(i,j)+alpha4*b(i,j);
    end
end
subplot (2,2,1),imshow(c1),title('Blended image')
xlabel(sprintf('alpha value is %g',alpha1))
subplot(2,2,2),imshow(c2),title('Blended image')
xlabel(sprintf('alpha value is %g',alpha2))
subplot(2,2,3),imshow(c3),title('Blended image')
xlabel(sprintf('alpha value is %g',alpha3))
subplot(2,2,4),imshow(c4),title('Blended image')
xlabel(sprintf('alpha value is %g',alpha4))
```

Fig. 5.87 MATLAB code to implement alpha blending

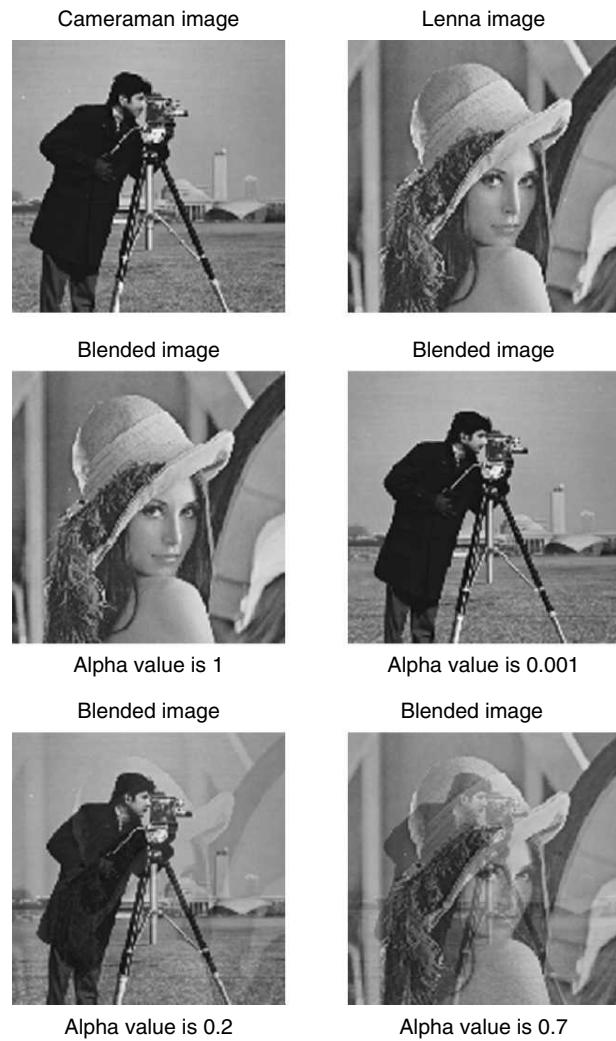
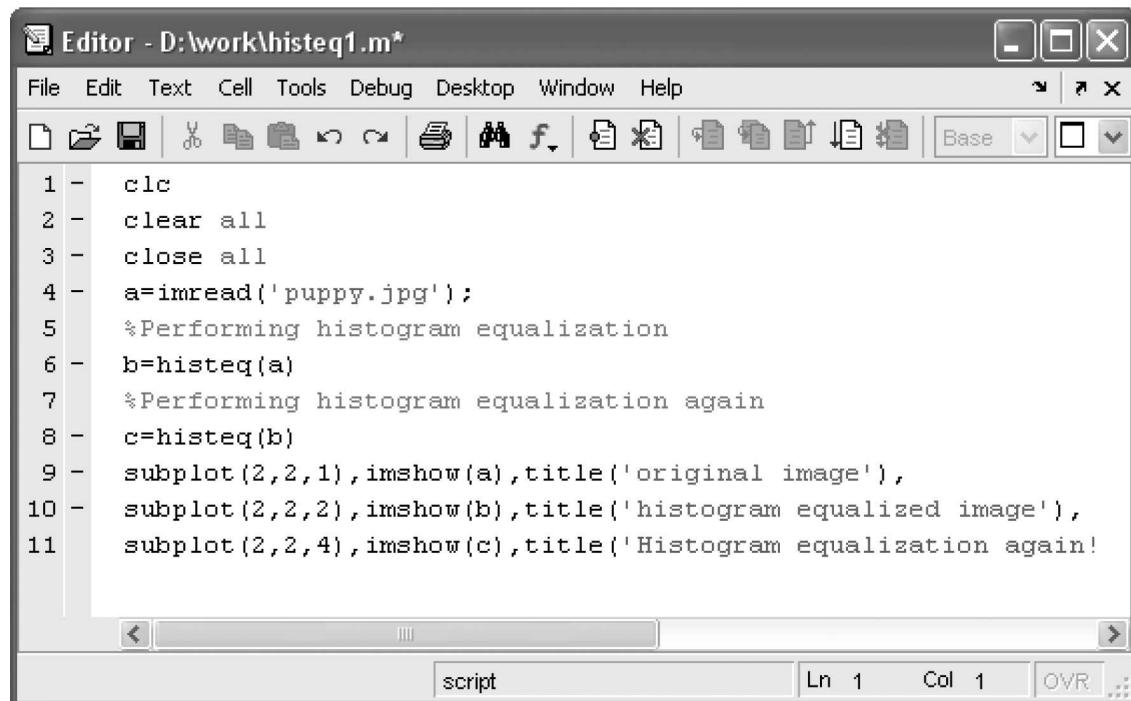


Fig. 5.88 *Output of alpha blending*

MATLAB Example 12: Histogram equalisation *An image is histogram equalised. What will happen if it is histogram equalised again?*

Solution If we perform histogram equalisation of an image which is already histogram equalised then there will not be any effect in the visual quality of the image. This fact can be illustrated through a simple MATLAB code and the corresponding output is shown in Fig. 5.89 and Fig. 5.90 respectively.



The screenshot shows the MATLAB Editor window with the file 'histeq1.m' open. The code performs histogram equalization on a puppy image. It starts with clearing the workspace, then reads the image 'puppy.jpg'. It performs histogram equalization twice, resulting in three images: the original image, the histogram-equalized image, and the image after a second histogram equalization. The code is as follows:

```
1 - clc
2 - clear all
3 - close all
4 - a=imread('puppy.jpg');
5 - %Performing histogram equalization
6 - b=histeq(a)
7 - %Performing histogram equalization again
8 - c=histeq(b)
9 - subplot(2,2,1),imshow(a),title('original image'),
10 - subplot(2,2,2),imshow(b),title('histogram equalized image'),
11 - subplot(2,2,4),imshow(c),title('Histogram equalization again!')
```

Fig. 5.89 MATLAB code for Example 13

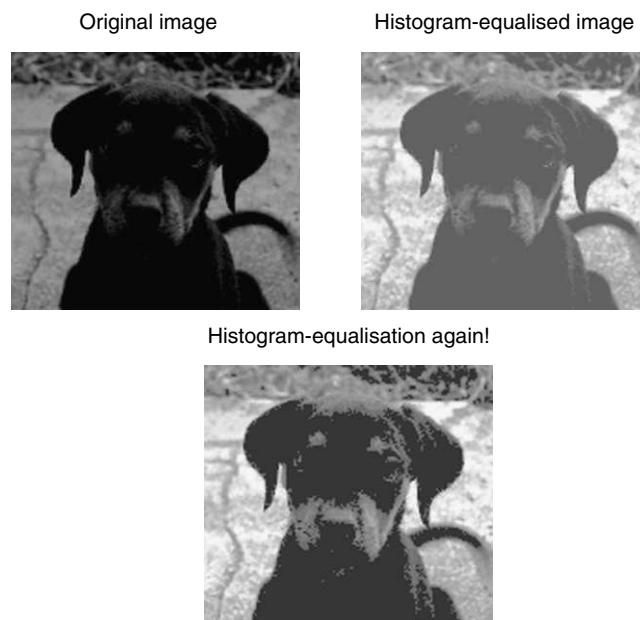
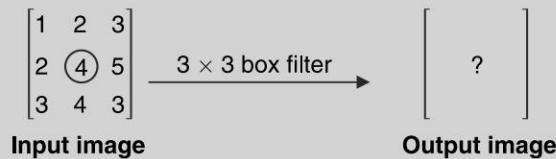


Fig. 5.90 Result of MATLAB code shown in Fig. 5.89

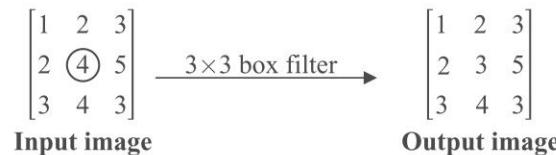
SOLVED PROBLEMS

1. What is the value of the central pixel (marked by a round) if it is smoothed by a 3×3 box filter?



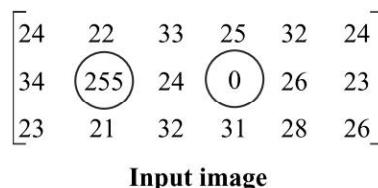
The 3×3 box filter is given by $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. If this box filter is applied to the input image, the central value '4' will be replaced by the average of the neighbourhood defined by the mask. This is given by

averaging operation $= \frac{1}{9} \times [1+2+3+2+4+5+3+4+3] = \frac{27}{9} = 3$. The pixel value '4' will be replaced by the average of the neighbourhood which results in '3'.



2. Justify the statement "Median filter is an effective tool to minimise salt-and-pepper noise" through simple illustration.

Let us consider an image corrupted by salt-and-pepper noise. Here 'salt' refers to high gray-level value and 'pepper' refers to low gray-level value. Random occurrence of white and black values is generally termed salt-and-pepper noise. The image corrupted by salt-and-pepper noise is given below.



In the input image, two values 255 and 0 are circled to indicate that they are different from their neighbours. Hence, 255 is equivalent to salt noise, and 0 is equivalent to pepper noise. Then a 3×3 median filter is applied to the input image as follows:

Step 1 Apply 3×3 median filter to the pixel value 255.

$$\begin{bmatrix} 24 & 22 & 33 & 25 & 32 & 24 \\ 34 & 255 & 24 & 0 & 26 & 23 \\ 23 & 21 & 32 & 31 & 28 & 26 \end{bmatrix}$$

Step 1a Arrange the pixel values in the ascending order:

21 22 23 24 24 32 33 34 255

Step 1b Compute the median of the pixel value 255:

~~21~~ ~~22~~ ~~23~~ ~~24~~ ~~24~~ ~~32~~ ~~33~~ ~~34~~ ~~255~~

The median value is computed to be 24.

Step 2 Apply a 3×3 median filter to the pixel value '24'.

$$\begin{bmatrix} 24 & 22 & 33 & 25 & 32 & 24 \\ 34 & 255 & 24 & 0 & 26 & 23 \\ 23 & 21 & 32 & 31 & 28 & 26 \end{bmatrix}$$

Step 2a Arrange the pixel values in the ascending order:

0 21 22 24 25 31 32 33 255

Step 2b Compute the median of the pixel value '24'.

~~0~~ ~~21~~ ~~22~~ ~~24~~ ~~25~~ ~~31~~ ~~32~~ ~~33~~ ~~255~~

The median of the pixel value is computed to be 25.

Step 3 Apply a 3×3 median filter to the pixel value '0'.

$$\begin{bmatrix} 24 & 22 & 33 & 25 & 32 & 24 \\ 34 & 255 & 24 & 0 & 26 & 23 \\ 23 & 21 & 32 & 31 & 28 & 26 \end{bmatrix}$$

Step 3a Arrange the pixels in the neighbourhood of 0 in the ascending order:

0 24 25 26 28 31 32 33 33

Step 3b The median value is computed to be 28.

~~0~~ ~~24~~ ~~25~~ ~~26~~ ~~28~~ ~~31~~ ~~32~~ ~~33~~ ~~33~~

Step 4 Apply a 3×3 median filter to the pixel value '26'.

$$\begin{bmatrix} 24 & 22 & 33 & 25 & 32 & 24 \\ 34 & 255 & 24 & 0 & 26 & 23 \\ 23 & 21 & 32 & 31 & 28 & 26 \end{bmatrix}$$

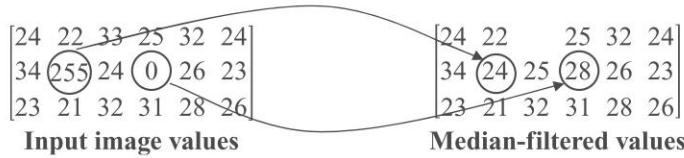
Step 4a Arrange the pixels in the neighbourhood of 26 in the ascending order:

0 23 24 25 26 26 28 31 32

Step 4b The median value is computed to be 26.

$\emptyset 23 24 25 26 26 28 31 32$

Then the input image and the median-filtered image are shown below:



From the input and the median-filtered values, we can observe that the value 255 is replaced by 25 and the value 0 is replaced by 28. Here, we can observe that the salt-and-pepper noise are values replaced by a value similar to that of their neighbours. From this, it is evident that a median filter is an effective tool to suppress salt-and-pepper noise.

3. Filter the following image using a 3×3 neighbourhood averaging by assuming (a) zero padding, and (b) pixel replication.

$$\begin{bmatrix} 1 & 2 & 3 & 2 \\ 4 & 2 & 5 & 1 \\ 1 & 2 & 6 & 3 \\ 2 & 6 & 4 & 7 \end{bmatrix}$$

Case (i): 3×3 averaging using zero padding

Step 1 Zero padding

If zero padding is assumed, the image is padded with zero along the edges as shown below:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 2 & 0 \\ 0 & 4 & 2 & 5 & 0 \\ 0 & 2 & 6 & 3 & 0 \\ 0 & 2 & 6 & 4 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 2 Determine the average value

As an illustration, the procedure to compute the average value of the first pixel 1 using a 3×3 neighbourhood is given below:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 2 & 0 \\ 0 & 4 & 2 & 5 & 1 & 0 \\ 0 & 1 & 2 & 6 & 3 & 0 \\ 0 & 2 & 6 & 4 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The average value of the pixel '1' in the 3×3 neighbourhood is given by

$$\frac{1}{9} \times [0 + 0 + 0 + 0 + 1 + 2 + 0 + 4 + 2]$$

$\frac{1}{9} \times 9 = 1$. The average value is calculated to be equal to 1.

Then the average value of the next pixel 2 in the 3×3 neighbourhood is calculated as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & (2) & 3 & 2 & 0 \\ 0 & 4 & 2 & 5 & 1 & 0 \\ 0 & 1 & 2 & 6 & 3 & 0 \\ 0 & 2 & 6 & 4 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The average value of the pixel 2 = $\frac{1}{9} \times [0 + 0 + 0 + 1 + 2 + 3 + 4 + 2 + 5] = 1.89 = 2$

Similarly, the average of other pixel values are calculated and given as

$$\text{average value of the pixel 3} = \frac{1}{9} \times [0 + 0 + 0 + 2 + 3 + 2 + 2 + 5 + 1] = \frac{15}{9} = 1.6676 = 2$$

$$\text{Average value of the pixel 2} = \frac{1}{9} \times [0 + 0 + 0 + 3 + 2 + 0 + 5 + 1 + 0] = \frac{11}{9} = 1.22 = 1$$

$$\text{Average value of the pixel 4} = \frac{1}{9} \times [0 + 1 + 2 + 0 + 4 + 2 + 0 + 1 + 2] = \frac{12}{9} = 1.33 = 1$$

$$\text{Average value of the pixel 2} = \frac{1}{9} \times [1 + 2 + 3 + 4 + 2 + 5 + 1 + 2 + 6] = \frac{26}{9} = 2.88 = 3$$

$$\text{Average value of the pixel 5} = \frac{1}{9} \times [2 + 3 + 2 + 2 + 5 + 1 + 2 + 6 + 3] = \frac{26}{9} = 2.88 = 3$$

$$\text{Average value of the pixel 1} = \frac{1}{9} \times [3 + 2 + 0 + 5 + 1 + 0 + 6 + 3 + 0] = \frac{20}{9} = 2.22 = 2$$

$$\text{Average value of the pixel 1} = \frac{1}{9} \times [0 + 4 + 2 + 0 + 1 + 2 + 0 + 2 + 6] = \frac{17}{9} = 1.88 = 2$$

$$\text{Average value of the pixel 2} = \frac{1}{9} \times [4 + 2 + 5 + 1 + 2 + 6 + 2 + 6 + 4] = \frac{32}{9} = 3.55 = 4$$

$$\text{Average value of the pixel 6} = \frac{1}{9} \times [2 + 5 + 1 + 2 + 6 + 3 + 6 + 4 + 7] = \frac{36}{9} = 4$$

$$\text{Average value of the pixel 3} = \frac{1}{9} \times [5 + 1 + 0 + 6 + 3 + 0 + 4 + 7 + 0] = \frac{26}{9} = 2.88 = 3$$

$$\text{Average value of the pixel } 2 = \frac{1}{9} \times [0+1+2+0+2+6+0+0+0] = \frac{11}{9} = 1.22 = 1$$

$$\text{Average value of the pixel } 6 = \frac{1}{9} \times [1+2+6+2+6+4+0+0+0] = \frac{21}{9} = 2.33 = 2$$

$$\text{Average value of the pixel } 4 = \frac{1}{9} \times [2+6+3+6+4+7+0+0+0] = \frac{28}{9} = 3.11 = 3$$

$$\text{Average value of the pixel } 7 = \frac{1}{9} \times [6+3+0+4+7+0+0+0+0] = \frac{20}{9} = 2.22 = 2$$

The original image and the 3×3 neighbourhood average-filtered images are shown below:

$$\begin{array}{c} \begin{bmatrix} 1 & 2 & 3 & 2 \\ 4 & 2 & 5 & 1 \\ 1 & 2 & 6 & 3 \\ 2 & 6 & 4 & 7 \end{bmatrix} \xrightarrow{\hspace{10em}} \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 3 & 3 & 2 \\ 2 & 4 & 4 & 3 \\ 1 & 2 & 3 & 2 \end{bmatrix} \\ \text{Original image} \qquad \qquad \qquad \text{3} \times 3 \text{ average-filtered image} \end{array}$$

Case (ii) Averaging through pixel replication

Instead of zero padding, pixel replication is done first, and then averaging of the pixel in the neighbourhood is determined.

Step 1 Pixel replication

The input image after pixel replication is shown below:

$$\begin{array}{c} \begin{bmatrix} 1 & 2 & 3 & 2 \\ 4 & 2 & 5 & 1 \\ 1 & 2 & 6 & 3 \\ 2 & 6 & 4 & 7 \end{bmatrix} \qquad \qquad \begin{bmatrix} 1 & 1 & 2 & 3 & 2 & 2 \\ 1 & 2 & 3 & 2 & 2 & 2 \\ 4 & 4 & 2 & 5 & 1 & 1 \\ 1 & 1 & 2 & 6 & 3 & 3 \\ 2 & 2 & 6 & 4 & 7 & 7 \\ 2 & 2 & 6 & 4 & 7 & 7 \end{bmatrix} \\ \text{Input image} \qquad \qquad \qquad \text{Pixel replicated image} \end{array}$$

Step 2 Determination of average value

As an illustration, the determination of the average value using a 3×3 mask for the pixel value 1 is shown below:

$$\begin{bmatrix} 1 & 1 & 2 & 3 & 2 & 2 \\ 1 & \textcircled{1} & 2 & 3 & 2 & 2 \\ 4 & 4 & 2 & 5 & 1 & 1 \\ 1 & 1 & 2 & 6 & 3 & 3 \\ 2 & 2 & 6 & 4 & 7 & 7 \\ 2 & 2 & 6 & 4 & 7 & 7 \end{bmatrix}$$

The procedure to determine the average value of the marked pixel (the pixel which is rounded) using a 3×3 window is shown below:

$$\begin{bmatrix} 1 & 1 & 2 & 3 & 2 & 2 \\ 1 & \textcircled{1} & 2 & 3 & 2 & 2 \\ 4 & 4 & 2 & 5 & 1 & 1 \\ 1 & 1 & 2 & 6 & 3 & 3 \\ 2 & 2 & 6 & 4 & 7 & 7 \\ 2 & 2 & 6 & 4 & 7 & 7 \end{bmatrix}$$

$$\text{Average value of the pixel 1} = \frac{1}{9} \times [1+1+2+1+1+2+4+4+2] = \frac{18}{9} = 2$$

Then, the average value of the pixel 2 is calculated as

$$\begin{bmatrix} 1 & 1 & 2 & 3 & 2 & 2 \\ 1 & 1 & \textcircled{2} & 3 & 2 & 2 \\ 4 & 4 & 2 & 5 & 1 & 1 \\ 1 & 1 & 2 & 6 & 3 & 3 \\ 2 & 2 & 6 & 4 & 7 & 7 \\ 2 & 2 & 6 & 4 & 7 & 7 \end{bmatrix}$$

$$\text{Average value of the pixel 2} = \frac{1}{9} \times [1+2+3+1+2+3+4+2+5] = \frac{23}{9} = 2.55 \approx 3$$

Similarly,

$$\text{Average value of the pixel 3} = \frac{1}{9} \times [2+3+2+2+3+2+2+5+1] = \frac{22}{9} = 2.44 \approx 2$$

$$\text{Average value of the pixel 2} = \frac{1}{9} \times [3+2+2+3+2+2+5+1+1] = \frac{21}{9} = 2.33 \approx 2$$

$$\text{Average value of the pixel 4} = \frac{1}{9} \times [1+1+2+4+4+2+1+1+2] = \frac{18}{9} = 2$$

$$\text{Average value of the pixel 2} = \frac{1}{9} \times [1+2+3+4+2+5+1+2+6] = \frac{26}{9} = 2.88 \approx 3$$

$$\text{Average value of the pixel 5} = \frac{1}{9} \times [2+3+2+2+5+1+2+6+3] = \frac{26}{9} = 2.88 \approx 3$$

$$\text{Average value of the pixel 1} = \frac{1}{9} \times [3+2+2+5+1+1+6+3+3] = \frac{26}{9} = 2.88 \approx 3$$

$$\text{Average value of the pixel 1} = \frac{1}{9} \times [4+4+2+1+1+2+2+2+6] = \frac{24}{9} = 2.66 \approx 3$$

$$\text{Average value of the pixel 2} = \frac{1}{9} \times [4+2+5+1+2+6+2+6+4] = \frac{32}{9} = 3.55 \approx 4$$

$$\text{Average value of the pixel } 6 = \frac{1}{9} \times [2+5+1+2+6+3+6+4+7] = \frac{36}{9} = 4$$

$$\text{Average value of the pixel } 3 = \frac{1}{9} \times [5+1+1+6+3+3+4+7+7] = \frac{37}{9} = 4.11 \approx 4$$

$$\text{Average value of the pixel } 2 = \frac{1}{9} \times [1+1+2+2+2+6+2+2+6] = \frac{24}{9} = 2.66 \approx 3$$

$$\text{Average value of the pixel } 6 = \frac{1}{9} \times [1+2+6+2+6+4+2+6+4] = \frac{33}{9} = 3.66 \approx 4$$

$$\text{Average value of the pixel } 4 = \frac{1}{9} \times [2+6+3+6+4+7+6+4+7] = \frac{45}{9} = 5$$

$$\text{Average value of the pixel } 7 = \frac{1}{9} \times [6+3+3+4+7+7+4+7+7] = \frac{48}{9} = 5.33 \approx 5$$

The original image and the 3×3 average-filtered images are given below:

$$\begin{array}{c} \begin{bmatrix} 1 & 2 & 3 & 2 \\ 4 & 2 & 5 & 1 \\ 1 & 2 & 6 & 3 \\ 2 & 6 & 4 & 7 \end{bmatrix} \xrightarrow{\hspace{10em}} \begin{bmatrix} 2 & 3 & 2 & 2 \\ 2 & 3 & 3 & 3 \\ 3 & 4 & 4 & 4 \\ 3 & 4 & 5 & 5 \end{bmatrix} \\ \text{Original image} \qquad \qquad \qquad \text{3} \times 3 \text{ Filtered image} \end{array}$$

4. What is the value of the marked pixel after a 5×5 median filter?

$$\begin{bmatrix} 2 & 1 & 3 & 4 & 5 \\ 1 & 1 & 0 & 2 & 3 \\ 2 & 0 & \textcircled{0} & 1 & 2 \\ 5 & 1 & 2 & 3 & 1 \\ 4 & 3 & 1 & 2 & 0 \end{bmatrix}$$

In order to perform a median filter, the pixels have to be sorted either in the ascending or descending order.

Step 1 Arranging the pixels in ascending order:

0 0 0 0 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 4 4 5 5

Step 2 To determine the median value:

0 0 0 0 1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 4 4 5 5

The median value is found to be 2. Hence, the marked pixel 0 will be replaced by 2.

5. What is the value of the marked pixel after a 5×5 mode filter?

$$\begin{bmatrix} 3 & 3 & 1 & 2 & 1 \\ 4 & 5 & 2 & 3 & 0 \\ 3 & 2 & 1 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 3 & 1 & 2 & 3 & 5 \end{bmatrix}$$

Mode refers to the most frequently occurring value. Here, a 5×5 mask is employed. Hence, we have to take into consideration all the pixel values given in the image. From the observation, pixel value 0 occurs twice, the pixel value 1 occurs six times, the pixel value 2 occurs five times, the pixel value 3 occurs eight times, pixel value 4 occurs twice and the pixel value 5 occurs twice. It is clear that the pixel value 3 occurs more frequently than other pixel values. Hence, the mode of the marked pixel is 3.

6. Consider a point transformation which maps the input image $f[m,n]$ into an output image $g[m,n]$ through the transformation T . The input and the output images are defined as follows:

- (i) Input image $f[m, n] = r$ where $0 \leq r \leq 1$
- (ii) Output image $g[m, n] = s = T(r)$ where $0 \leq s \leq 1$
- (iii) The transformation is given by $T(r) = ar + b$

Determine the values of a and b in terms of m_r and σ_r^2 , the mean and variance of r . Assume $m_s = 0.5$ and $\sigma_s^2 = 0.25\sigma_r^2$.

It is given that $s = T(r) = ar + b$ (5.32)

Taking the expectation operation on both sides, we get

$$m_s = E(s) = E(ar + b) \quad (5.33)$$

Since, expectation is a linear operation, the above equation can be written as

$$E(s) = aE(r) + b \quad (5.34)$$

$$m_s = am_r + b \quad (5.35)$$

The variance of the output is given by

$$\sigma_s^2 = E\left\{(s - m_s)^2\right\} \quad (5.36)$$

$$\text{But } s = ar + b \quad (5.37)$$

and

$$m_s = am_r + b \quad (5.38)$$

substituting Eqs. (5.37) and (5.38) in Eq. (5.36), we get

$$\begin{aligned}
 \sigma_s^2 &= E \left\{ [ar + b - (am_r + b)]^2 \right\} \\
 \sigma_s^2 &= E \left\{ [ar + b - am_r - b]^2 \right\} \\
 \sigma_s^2 &= E \left\{ [a(r - m_r)]^2 \right\} \\
 \sigma_s^2 &= a^2 E \left\{ (r - m_r)^2 \right\} \\
 \sigma_s^2 &= a^2 \sigma_r^2
 \end{aligned} \tag{5.39}$$

In the problem it is given as $m_s = \frac{1}{2}$ and $\sigma_s^2 = \frac{1}{4} \sigma_r^2$. Substituting these values in Eq. (5.35) and Eq. (5.39), we get

$$\frac{1}{2} = am_r + b \text{ and}$$

$$\frac{1}{4} \sigma_r^2 = a^2 \sigma_r^2, \text{ this implies } a = \frac{1}{2}.$$

To determine b From Eq. (5.38), we have $m_s = am_r + b$

But $m_s = \frac{1}{2}$. Substituting this value in the expression m_s , we get

$\frac{1}{2} = am_r + b$. Also, $a = \frac{1}{2}$; substituting this value of a , we get

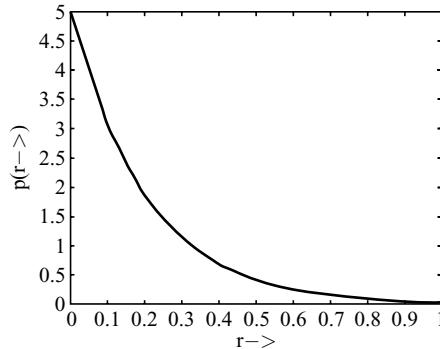
$$\frac{1}{2} = \frac{1}{2} m_r + b$$

From this, the expression of the parameter b is given by

$$b = \frac{1}{2} (1 - m_r)$$

7. The gray-level probability density function of an image is given by $p(r) = 5 \exp(-5r)$, $0 \leq r \leq 1$. Which of the transformations $s = r^2$ or $s = \sqrt{r}$ would produce a better image? Explain.

On plotting the given gray-level probability density function, we get



From the plot, it is obvious that most pixels have low gray level. Then, the goal is to spread out the low gray level. Two choices available to us are $s = r^2$ and $s = \sqrt{r}$.

$s = \sqrt{r}$ stretches the low gray level and $s = r^2$ compresses the low gray level. Hence, $s = \sqrt{r}$ is a better choice.

8. The Fourier transform of a spatial filter is given by $F(k, l) = 8 - 2\cos(2\pi k) - 2\cos(2\pi l) - 2\cos(2\pi(k+l)) - 2\cos(2\pi(k-l))$. Determine the coefficients of the filter and present it as a 3×3 operator. If any positive image is convolved with the operator, what will be the mean pixel intensity in the output image?

We know that $\cos \theta = \frac{e^{j\theta} + e^{-j\theta}}{2}$. The second term in the expression of $F(k, l)$ can be written as

$$2\cos(2\pi k) = 2 \times \left(\frac{e^{j2\pi k} + e^{-j2\pi k}}{2} \right) = e^{j2\pi k} + e^{-j2\pi k}.$$

$e^{j2\pi k}$ is the Fourier transform of the delta function at $x = -1$ and $e^{-j2\pi k}$ is the Fourier transform of the delta function at $x = 1$.

Similarly, the expression $2\cos(2\pi(k+l))$ can be expanded as

$$2\cos(2\pi(k+l)) = 2 \times \left(\frac{e^{j2\pi(k+l)} + e^{-j2\pi(k+l)}}{2} \right) = e^{j2\pi(k+l)} + e^{-j2\pi(k+l)} \text{ which is the Fourier transform of the delta function at } x, y = -1, -1.$$

Also, the expression $2\cos(2\pi(k-l))$ can be expanded as

$$2\cos(2\pi(k-l)) = 2 \times \left(\frac{e^{j2\pi(k-l)} + e^{-j2\pi(k-l)}}{2} \right) = e^{j2\pi(k-l)} + e^{-j2\pi(k-l)} \text{ which corresponds to the Fourier transform of the delta function at } x, y = 1, 1.$$

The constant term 8 corresponds to the delta function at $0, 0$. Putting it all together we can write the coefficients as a 3×3 operator as

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

If we convolve any positive image with the above operator, the mean pixel value will be zero.

9. Given an image of size 3×3 as $f(m, n) = \begin{bmatrix} 128 & 212 & 255 \\ 54 & 62 & 124 \\ 140 & 152 & 156 \end{bmatrix}$. Determine the output image

(i) $c = \frac{L}{\log_{10}(1+L)}$

Case (i) $c = 1$

When we choose $c = 1$, the logarithmic transformation is given by $g(m, n) = \log_{10}(1 + f(m, n))$. Applying this transformation to the input image, we get

$$g(1, 1) = \log_{10}(1 + f(1, 1)) = \log_{10}(1 + 128) = \log_{10}(129) = 2.11 \approx 3$$

$$g(1, 2) = \log_{10}(1 + f(1, 2)) = \log_{10}(1 + 212) = \log_{10}(213) = 2.328 \approx 3$$

$$g(1, 3) = \log_{10}(1 + f(1, 3)) = \log_{10}(1 + 255) = \log_{10}(256) = 2.408 \approx 3$$

$$g(2, 1) = \log_{10}(1 + f(2, 1)) = \log_{10}(1 + 54) = \log_{10}(55) \approx 2$$

$$g(2, 2) = \log_{10}(1 + f(2, 2)) = \log_{10}(1 + 62) = \log_{10}(63) \approx 2$$

$$g(2, 3) = \log_{10}(1 + f(2, 3)) = \log_{10}(1 + 124) = \log_{10}(125) \approx 3$$

$$g(3, 1) = \log_{10}(1 + f(3, 1)) = \log_{10}(1 + 140) = \log_{10}(141) \approx 3$$

$$g(3, 2) = \log_{10}(1 + f(3, 2)) = \log_{10}(1 + 152) = \log_{10}(153) \approx 3$$

$$g(3, 3) = \log_{10}(1 + f(3, 3)) = \log_{10}(1 + 156) = \log_{10}(157) \approx 3$$

The output image $g(m, n) = \begin{bmatrix} 3 & 3 & 3 \\ 2 & 2 & 3 \\ 3 & 3 & 3 \end{bmatrix}$

By looking at the input and output images, it is clear that the dynamic range of the input image is compressed. Thus, logarithmic transformation is used for dynamic range compression.

$$\text{Case (ii)} \quad c = \frac{L}{\log_{10}(1+L)}$$

In this problem, if $L = 255$, $c = \frac{L}{\log_{10}(1+L)} = \frac{255}{\log_{10}(256)} = \frac{255}{2.408} = 105.89 \approx 106$.

The output image is calculated as

$$g(1, 1) = \lceil 106 \times \log_{10}(1 + f(1, 1)) \rceil = \lceil 106 \times \log_{10}(129) \rceil \approx 224$$

$$g(1, 2) = \lceil 106 \times \log_{10}(1 + f(1, 2)) \rceil = \lceil 106 \times \log_{10}(213) \rceil \approx 247$$

$$g(1, 3) = \lceil 106 \times \log_{10}(1 + f(1, 3)) \rceil = \lceil 106 \times \log_{10}(256) \rceil \approx 256$$

$$g(2,1) = \lceil 106 \times \log_{10}(1 + f(2,1)) \rceil = \lceil 106 \times \log_{10}(55) \rceil \approx 185$$

$$g(2,2) = \lceil 106 \times \log_{10}(1 + f(2,2)) \rceil = \lceil 106 \times \log_{10}(63) \rceil \approx 191$$

$$g(2,3) = \lceil 106 \times \log_{10}(1 + f(2,3)) \rceil = \lceil 106 \times \log_{10}(125) \rceil \approx 223$$

$$g(3,1) = \lceil 106 \times \log_{10}(1 + f(3,1)) \rceil = \lceil 106 \times \log_{10}(141) \rceil \approx 228$$

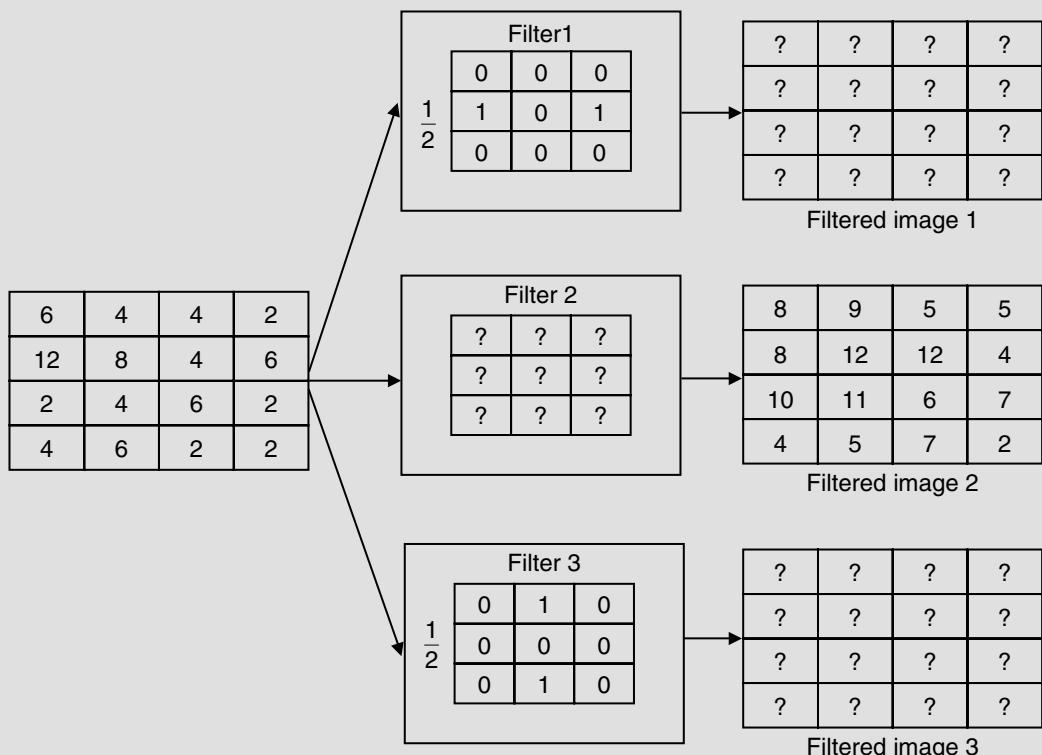
$$g(3,2) = \lceil 106 \times \log_{10}(1 + f(3,2)) \rceil = \lceil 106 \times \log_{10}(153) \rceil \approx 232$$

$$g(3,3) = \lceil 106 \times \log_{10}(1 + f(3,3)) \rceil = \lceil 106 \times \log_{10}(157) \rceil \approx 233$$

The output image $g(m,n)$ is given as

$$g(m,n) = \begin{bmatrix} 224 & 247 & 256 \\ 185 & 191 & 223 \\ 228 & 232 & 233 \end{bmatrix}$$

10. A 4×4 grayscale image passes through three spatial linear shift-invariant filters, resulting in three filtered images.



Compute the filtered image 1 and filtered image 3. Based on the relationship between Filtered image 1, Filtered image 2, and Filtered image 3, determine the filter coefficients in the shift-invariant linear filter 2. (Assume zero padding.)

Step 1 Computing the filtered image 1

After zero padding, the input image is shown in Fig. 5.91.

After zero padding, the input image is convolved with the linear filter 1 to get the filtered image 1. The filtered image 1 is given in Fig. 5.92.

Step 2 Computing the filtered image 3

After zero padding, the input image is convolved with the filter 3 to get the filtered image 3 which is shown in Fig. 5.93.

Step 3 Comparison of filtered images 1, 2 and 3

By comparing the filtered images 1, 2 and 3

Filtered image 2 = Filtered image 1 + Filtered image 3

Step 4 Computation of the filter 2

2	5	3	2
4	8	7	2
2	4	3	3
3	3	4	1

Fig. 5.92 *Filtered image 1*

0	0	0	0	0	0
0	6	4	4	2	0
0	12	8	4	6	0
0	2	4	6	2	0
0	4	6	2	2	0
0	0	0	0	0	0

Fig. 5.91 Input image after zero padding

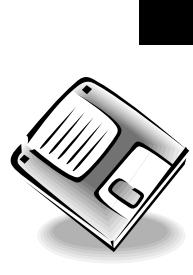
The filters are linear shift-invariant filters. Hence,

Filter 2 = Filter 1 + Filter 3

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

Fig. 5.93 Filtered image 3

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



Summary

- 
 - The goal of image enhancement is to improve the perception of the image through modification of intensity functions, image spectral content, or a combination of these functions.
 - Removing blurring and noise, increasing contrast, and revealing details are examples of image-enhancement operations.
 - Image enhancement can be performed either in the spatial domain or in the frequency domain. The selection of a particular image-enhancement technique depends on application.
 - The spatial-domain image-enhancement technique includes gray-level transformations like image negative, image slicing, image thresholding, gamma correction, etc.
 - In a point operation, the value of each pixel is recalculated according to a certain transformation. Point operations commonly involve the adjustment of brightness and contrast in an image.
 - The histogram of an image gives the frequency of occurrence of the gray levels. Two different images can have the same histogram. A histogram is invariant to rotation.

- The goal in histogram equalisation is to approximate the grayscale value distribution of an image to the uniform distribution. Histogram equalisation spreads the grayscale values and allows one to see a larger range of grayscale values.
- Spatial domain filtering is accomplished by convolving the image with a filter kernel. Low pass filtering is used to remove the noise at the expense of blurring of the image. High-pass filtering enhances the details in the image.
- Frequency domain filtering is accomplished by the multiplication of the image spectrum by the Fourier transform of the filter kernel. The filtered image in spatial domain is obtained by taking inverse Fourier Transform.
- Gaussian filters are local smoothing operators whose impulse response has the shape of a 2D Gaussian distribution. Gaussian filters can be used to filter out noise in images. The user can vary the standard deviation of the Gaussian and the convolution mask size.
- Image subtraction is useful for background removal. Image multiplication is useful for masking out a portion of an image.

Review Questions

1. If all the pixels in an image are shuffled, will there be any change in the histogram? Justify your answer.

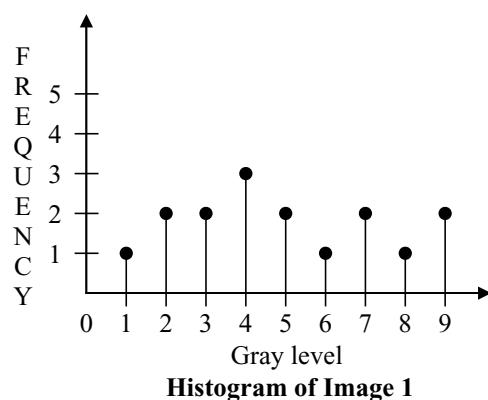
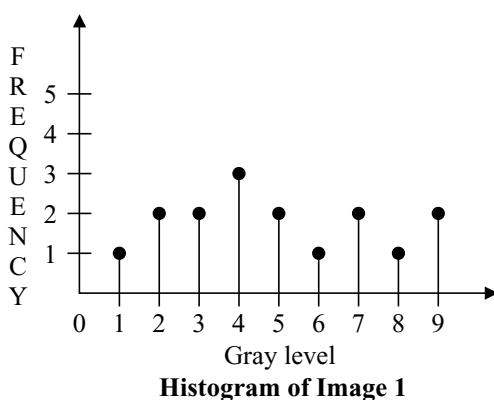
If all the pixels in an image are shuffled, there will not be any change in the histogram of the image. A histogram gives only the frequency of occurrence of the gray level. Consider two images, 1 and 2, as given below:

Image 2 is obtained by shuffling the rows of Image 1. Their corresponding histograms are shown below:

$\begin{bmatrix} 1 & 2 & 4 & 8 \\ 3 & 5 & 7 & 9 \\ 4 & 2 & 4 & 6 \\ 9 & 7 & 3 & 5 \end{bmatrix}$	$\begin{bmatrix} 9 & 7 & 3 & 5 \\ 3 & 5 & 7 & 9 \\ 1 & 2 & 4 & 8 \\ 4 & 2 & 4 & 6 \end{bmatrix}$
--	--

Image 1

Image 2



From the two histograms, it is clear that even if all the pixels in an image are shuffled, there will not be any change in the histogram of the image.

2. Can two different images have the same histogram? Justify your answer.

Yes, there is a possibility that two different images can have the same histogram. Here, we have taken an original image (humming bird), and then the image is flipped in the left/right direction. The histograms of the original image and the flipped image are taken. Since the values of the pixels are not affected by a 'flipping' operation, the histogram of the original image is the same as the histogram of the flipped image. The MATLAB code and the corresponding output are shown in Fig. 5.94 and Fig. 5.95 respectively.

From Fig. 5.95, it is clear that even though the original and flipped images are different, their histograms remain the same. From this illustration, it is obvious that two different images can have the same histogram.

Note: A histogram fails to recognise reflective symmetry.

3. What is the impact of applying the following look-up table?

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LUT	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15

on the 8-bit image given below.

$$\begin{bmatrix} 14 & 9 & 2 & 6 & 11 & 3 & 8 \\ 1 & 2 & 7 & 14 & 12 & 0 & 13 \\ 7 & 0 & 11 & 12 & 5 & 13 & 12 \\ 6 & 13 & 15 & 12 & 12 & 2 & 2 \\ 0 & 1 & 3 & 4 & 8 & 15 & 12 \\ 8 & 13 & 9 & 6 & 1 & 10 & 5 \\ 13 & 0 & 4 & 5 & 15 & 3 & 4 \end{bmatrix}$$

```

Editor - D:\work\histogram1.m
File Edit Text Cell Tools Debug Desktop Window Help
File Open Recent Home Editor Cell Window Help
1 - clc
2 - clear all
3 - close all
4 - a=imread('humm.jpg');;
5 - b=fliplr(a);;
6 - subplot(2,2,1),imshow(a),title('original image');;
7 - subplot(2,2,2),imshow(b),title('Flipped image');;
8 - subplot(2,2,3),imhist(a),subplot(2,2,4),imhist(b);
salt.m histogram1.m
script Ln 4 Col 22 OVR

```

Fig. 5.94 MATLAB code to plot the histogram of the original and flipped images

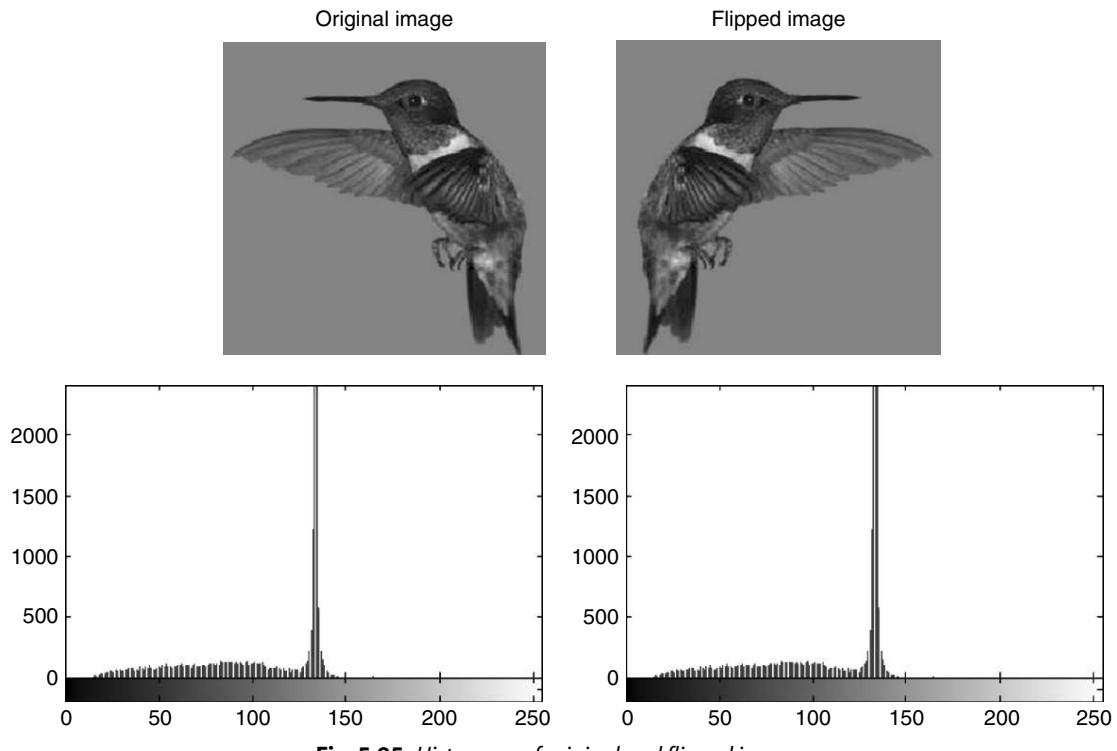


Fig. 5.95 Histogram of original and flipped images

On applying the look-up table on the given image, we get the resultant image as

15	12	9	11	13	9	12
8	9	11	15	14	8	14
11	8	13	14	10	14	14
11	14	15	14	14	9	9
8	8	9	10	12	15	14
12	14	12	11	8	13	10
14	8	10	10	15	9	10

By comparing the resultant image with the original image, it is obvious that the brightness of the resultant image is better than the original image.

4. What is a bit plane?

A bit plane contains a single bit of information for each pixel that will be displayed on the screen.

5. What is image filtering?

Image filtering is the process of modifying the pixels in an image based on some function of a local neighbourhood of the pixels.

6. Why does histogram equalisation (discrete histogram equalisation) not produce a perfectly flat histogram?

Discrete histogram equalisation basically performs the remapping of histogram components on the intensity scale. To obtain a flat histogram, the pixel intensities should be redistributed in such a way that there are L groups of n/L pixels with the same intensity, where L is the number of allowed discrete intensity levels and n is the total number of pixels in the input image. The histogram equalisation method has no provisions for this type of redistribution process hence discrete histogram equalisation does not yield a flat histogram.

7. What will we obtain if the arithmetic mean filter is applied to an image again and again? What will happen if we use the median filter instead?

Iterative arithmetic mean filtering will ultimately blur the whole image with constant gray level. On the other hand, iterative median filtering will only change once after the first round of filtering, and will remain stable from that moment on which is invariant to the filter.

8. Two images have the same histogram. Which of the following properties must they have in common? (i) Same total power (ii) Same entropy (iii) Same interpixel covariance function

The total power and entropy must be the same when two images have the same histogram, because the total power and entropy depend only on the pixel values and not on the order they are arranged in the image. The interpixel covariance function is not necessarily the same when two images have the same histogram.

9. Are convolutional filters linear? Justify your answer.

A convolutional mask operation provides a result that is a weighted sum of the values of a pixel and its neighbours. Hence, convolutional filters are linear.

10. What does the standard deviation of a histogram tell us about the image?

The standard deviation is also known as the square root of the variance. It tells us about the contrast. It describes the spread in the data, so a high-contrast image will have a high variance, and a low-contrast image will have a low variance.

Problems

5.1 A 4×4 , 4 bits/pixel original image is given by

$$\begin{bmatrix} 10 & 12 & 8 & 9 \\ 10 & 12 & 12 & 14 \\ 12 & 13 & 10 & 9 \\ 14 & 12 & 10 & 12 \end{bmatrix}$$

- (a) Apply histogram equalisation to the image by rounding the resulting image pixels to integers.
 (b) Sketch the histograms of the original image and the histogram-equalised image.
- 5.2 In a given application, an averaging mask is applied to input images to reduce noise and then a Laplacian mask is applied to enhance small details. Would mathematics predict that the result should be the same if the order of the operations were reversed?

- 5.3 A 5×5 image patch is shown below. Compute the value of the marked pixel if it is smoothed by a 3×3 average filter.

$$f(m, n) = \begin{bmatrix} 0 & 1 & 2 & 3 & 2 \\ 5 & 6 & 7 & 8 & 4 \\ 4 & 3 & \textcircled{2} & 1 & 2 \\ 8 & 7 & 6 & 5 & 3 \\ 1 & 5 & 3 & 7 & 8 \end{bmatrix}$$

- 5.4 A 4×4 , 4bits/pixel image is given by

$$f(m, n) = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 0 & 2 & 4 & 6 \\ 8 & 12 & 15 & 13 \\ 4 & 8 & 9 & 10 \end{bmatrix}.$$

Sketch its four bit planes.

- 5.5 A 4×4 , 4bits/pixel image $f(m, n)$ is passed through point-wise intensity transformation $g(m, n) = \text{round}(10\sqrt{f(m, n)})$. Determine the output image $g(m, n)$ if $f(m, n)$ is given by

$$f(m, n) = \begin{bmatrix} 12 & 8 & 4 & 9 \\ 10 & 5 & 3 & 6 \\ 8 & 12 & 9 & 13 \\ 4 & 12 & 9 & 10 \end{bmatrix}$$

- 5.6 A 4×4 , 4bits/pixel image is given by

$$f(m, n) = \begin{bmatrix} 12 & 13 & 15 & 7 \\ 8 & 2 & 4 & 6 \\ 7 & 12 & 15 & 13 \\ 4 & 8 & 9 & 10 \end{bmatrix}$$

Apply full-scale contrast stretch to the image.

- 5.7 A two-dimensional process is defined as follows. A pixel is randomly chosen with uniform probability out of the $N \times N$ pixels. Then, another pixel is chosen independently from the remaining $N^2 - 1$ pixels. The two chosen pixels get the value 1 and the rest get the value 0. Determine the mean of this process.
- 5.8 A popular procedure for image enhancement combines high-frequency emphasis and histogram equalisation to achieve edge sharpening and contrast enhancement.
- Prove whether or not it matters which process is applied first.
 - If the order does matter, give a rationale for using one or the other method first.

- 5.9 An image has gray levels ranging from 0 to 19. The histogram of the image is given below:

$$\begin{bmatrix} \text{Graylevel: } 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 \\ \text{Frequency: } 11 & 27 & 43 & 33 & 21 & 24 & 56 & 78 & 99 & 67 & 52 & 36 & 22 & 9 & 0 & 4 & 7 & 3 & 5 & 3 \end{bmatrix}$$

Compute the mean and standard deviation of this image.

- 5.10 The input image $f(m, n)$ is passed through a linear shift-invariant system $h(m, n)$.

Determine the output image if $f(m, n)$ and $h(m, n)$ are given below:

$$f(m, n) = \begin{bmatrix} 12 & 10 & 8 & 4 \\ 8 & 14 & 6 & 9 \\ 5 & 9 & 13 & 8 \\ 14 & 5 & 7 & 9 \end{bmatrix} \text{ and } h(m, n) = \frac{1}{4} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \text{ Assume zero padding of the original image.}$$

- 5.11 If $h(n_1, n_2)$ denotes the impulse response of a 2D lowpass filter, a new filter impulse response is given by $h'(n_1, n_2) = (-1)^{n_1} (-1)^{n_2} h(n_1, n_2)$. What type of filter is this? Justify your answer.
- 5.12 Consider the result of a 5×5 uniform averaging filter to a digital image N times. Characterise the expected effect.
- 5.13 Is the histogram equalisation operation idempotent?
- 5.14 Give the linear filter masks for the following operations:
- Region averaging
 - Weighted region averaging
 - Image sharpening
- 5.15 What is the difference between a high-pass filter and a high-frequency emphasis filter? How does this difference affect the resultant image?

Matlab Exercises

1. Write a MATLAB code to plot the histogram of the input grayscale image. You can use `for` loops for indexing but not the built-in function `imhist`.

2. Read the following MATLAB code and describe what it does.

HINT: The image ‘rice.png’ is an eight-bit grayscale image.

```
a=imread('rice.png');
b=255-a;
imshow(a), figure, imshow(b)
```

3. Write a MATLAB program that performs histogram equalisation of the input grayscale image. You can use any built-in MATLAB functions in your code except for `histeq`. Also, plot the histogram of the original and the histogram-equalised image.
4. The powerlaw transformation is commonly used for contrast enhancement. Write a MATLAB function called `powerlaw` that accepts a unit 8 intensity image, and a variable `gamma` and applies the power-law transformation to the input image. The output image should be a unit 8 intensity image. Observe the output image by varying the value of `gamma` between 0.2 and 1.2 in the steps of 0.2.
5. Write a MATLAB code that performs bit-plane slicing of the input grayscale image.
6. Write a MATLAB code to perform the following:
- Spatial-domain low-pass filtering of the input image using different window sizes like 3×3 , 5×5 , 7×7 . What is the impact of window size on the filtered image?
 - Spatial-domain high-pass filtering of the input image using the mask

$$h(m, n) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

References

Books

1. Ernest L. Hall, *Computer Image Processing and Recognition*, Academic Press, 1979
2. A Rosenfeld and A C Kak, *Digital Picture Processing*, Academic Press, New York, 1976
3. W K Pratt, *Digital Image Processing*, John Wiley and Sons, 1992
4. R C Gonzalez and P Wintz, *Digital Image Processing*, Addison-Wesley, Reading, Massachusetts, 1977
5. Paul J Gibson and Clarke H Power, *Introductory Remote Sensing: Digital Image Processing and Applications*, Routledge, 2000
6. R A Schowengerdt, *Remote Sensing-Models and Methods for Image Processing*, Academic Press, 1997

Journals

1. B H Brinkman, A Manduca and R A Robb, *Optimised Homomorphic Unsharp Masking for MR Grayscale Inhomogeneity Correction*, IEEE Transaction in Medical Imaging, vol. 17, no.2, pp. 161–171, 1998
2. J Lee, *Digital Image Enhancement and Noise Filtering by use of Local Statistics*, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 165–168, 1980
3. M Nagao and T Matsuyama, *Edge Preserving Smoothing*, Computer Graphics and Image Processing, vol. 9, pp.394–407, 1979
4. D A Handley and W B Green, *Recent Developments in Digital Image Processing at the Image Processing Laboratory at the Jet Propulsion Laboratory*, Proceedings of IEEE, vol. 60, no. 7, pp. 821–828, 1972
5. J Lee, *Digital Image Smoothing and the Sigma Filter*, Computer Vision, Graphics and Image Processing, vol.24, pp. 255–269, 1983
6. S Aghagolzadeh and O K Ersoy, *Transform image Enhancement*, Optical Engineering, vol. 31, pp. 614–626, March 1992
7. A Polesel, *Image Enhancement via Adaptive Unsharp Masking*, IEEE Transaction on Image Processing, vol. 9, pp. 505–510, March 2000

Web Resources

1. Professor **Zhou Wang's** lecture notes give very good introduction to spatial-domain linear filtering and non-linear image filtering
<http://www.uta.edu/faculty/zhouwang/teaching/IDIP06/handouts.htm>
2. Dr Jorma Laaksonen working in Helsinki University of Technology, Finland; course material on digital image processing
http://www.cis.hut.fi/Opinnot/T-61.5100/index_en.shtml
3. Dr William Hoff's lecture notes for the course *Image and Multidimensional Signal Processing*:
<http://egweb.mines.edu/eges510/lectures/lectures.htm>
4. Image Processing Teaching Materials with JAVA:
<http://homepages.inf.ed.ac.uk/rbf/HIPR2/>
5. CV online: <http://homepages.inf.ed.ac.uk/rbf/CVonline/> edited by Robert B Fisher contains useful information related to image transformation and filters

6

Learning Objectives

The goal of image restoration is to reconstruct the original scene from a degraded observation. After reading this chapter, the reader should be familiar with the following concepts:

*different types of image degradation
linear image-restoration techniques
Non-linear image-restoration techniques
blind deconvolution*

Image Restoration and Denoising

6.1 INTRODUCTION

Image-restoration techniques aim at reversing the degradation undergone by an image to recover the true image. Images may be corrupted by degradation such as linear frequency distortion, noise, and blocking artifacts. The degradation consists of two distinct processes:—the deterministic blur and the random noise. The blur may be due to a number of reasons, such as motion, defocusing, and atmospheric turbulence. The noise may originate in the image-formation process, the transmission process, or a combination of them. Most restoration techniques model the degradation process and attempt to apply an inverse procedure to obtain an approximation of the original image. Many image-restoration algorithms have their roots in well-developed areas of mathematics such as estimation theory, the solution of ill-posed problems, linear algebra and numerical analysis. Iterative image-restoration techniques often attempt to restore an image linearly or non-linearly by minimising some measures of degradation such as maximum likelihood, constrained least square, etc. Blind restoration techniques attempt to solve the restoration problem without knowing the blurring function. No general theory of image restoration has yet evolved; however, some solutions have been developed for linear and planar invariant systems.

6.2 IMAGE DEGRADATION

The process by which the original image is blurred is usually very complex and often unknown. To simplify the calculations, the degradation is often modeled as a linear function which is often referred as *point-spread function*.

The different causes of image degradation are

- Improper opening and closing of the shutter
- Atmospheric turbulence
- Misfocus of lens
- Relative motion between camera and object which causes motion blur

6.3 TYPES OF IMAGE BLUR

Blur can be introduced by an improperly focused lens, relative motion between the camera and the scene, or atmospheric turbulence. Blurring is a form of bandwidth reduction of an ideal image owing to the imperfect image-formation process. It can be caused by relative motion between the camera and the scene or by an optical system that is out of focus.

Image blurs can be broadly classified as (i) Gauss blur, (ii) out-of-focus blur, and (iii) motion blur.

6.3.1 Gauss Blur

Gauss blur is defined by the following point-spread function:

$$h(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (6.1)$$

Here, σ is called the variance of the blur. Gauss blur occurs due to long-time atmosphere exposure.

6.3.2 Out-of-focus Blur

This blurring is produced by a defocused optical system. It distributes a single point uniformly over a disk surrounding the point. The point-spread function of the out-of-focus blur is given by

$$h(x, y) = c \begin{cases} 1, & \sqrt{(x - c_x)^2 + (y - c_y)^2} \leq r \\ 0, & \text{otherwise} \end{cases} \quad (6.2)$$

where r is the radius and (c_x, c_y) is the centre of the out-of-focus point-spread function. The scaling factor c has to be chosen such that $\iint h(x, y) dx dy = 1$.

6.3.3 Motion Blur

Motion blur is due to relative motion between the recording device and the scene. When an object or the camera is moved during light exposure, a motion-blurred image is produced. The motion blur can be in the form of a translation, a rotation, a sudden change of scale, or some combinations of these. When the scene to be recorded translates relative to the camera at a constant velocity $\vartheta_{\text{relative}}$ under an angle of ϕ radians with

the horizontal axis during the exposure interval $[0, t_{\text{exposure}}]$, the distortion is one-dimensional. Defining the length of motion by

$$L = \vartheta_{\text{relative}} \times t_{\text{exposure}}$$

The point-spread function is given by

$$h(x, y, L, \phi) = \begin{cases} \frac{1}{L} & \text{if } \sqrt{x^2 + y^2} \leq \frac{L}{2} \text{ and } \frac{x}{y} = -\tan \phi \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

The discrete version of Eq. (6.3) is not easily captured in a closed form expression in general. For a special case, when $\phi = 0$, an appropriate approximation is given by

$$h(n_1, n_2, L) = \begin{cases} \frac{1}{L} & \text{if } n_1 = 0, |n_2| \leq \left\lfloor \frac{L-1}{2} \right\rfloor \\ \frac{1}{2L} \left\{ (L-1) - 2 \left\lfloor \frac{L-1}{2} \right\rfloor \right\} & \text{if } n_1 = 0, |n_2| = \left\lceil \frac{L-1}{2} \right\rceil \\ 0 & \text{elsewhere} \end{cases} \quad (6.4)$$

MATLAB Example 1 *Read an image and degrade the image using motion blur.*

Solution The MATLAB code which is used to create motion blur and the corresponding output are shown in Figs. 6.1 and 6.2 respectively.

The keyword *motion* in *fspecial* command is used to create the motion blur. The extent of blurring can be varied by two parameters LEN and THETA, where LEN refers to the linear motion of a camera by LEN pixels, with an angle of THETA degrees in a counter-clockwise direction. Here, we have varied both LEN and THETA and shown the results.

```
close all;
clear all;
clc;
a = imread(horse.jpg);
a = rgb2gray(a);
H = fspecial('motion', 10, 25);
MotionBlur_a = imfilter(a, H, 'replicate');
imshow(a), title('Original Image');
Fig., imshow(MotionBlur_a), title('Motion Blurred Image');
```

Fig. 6.1 MATLAB code to create motion blur

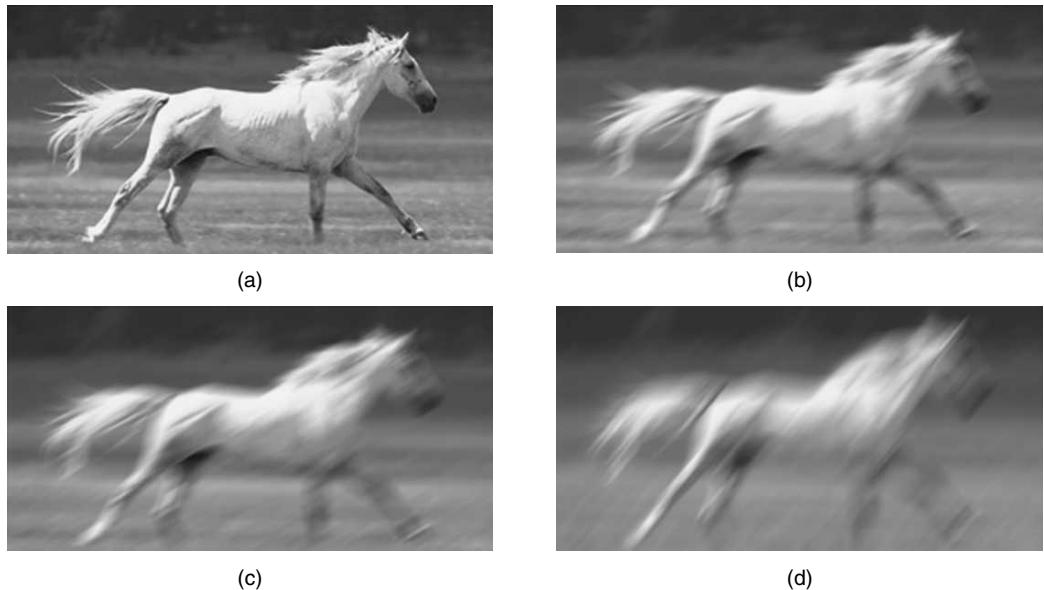


Fig. 6.2 (a) Original image (b) Motion-blurred image with [10 25] (c) Motion-blurred image with [15 35] (d) Motion-blurred image with [25 50]

6.3.4 Atmospheric Turbulence Blur

Atmospheric turbulence is a severe limitation in remote sensing. Although the blur introduced by atmospheric turbulence depends on a variety of factors like temperature, wind speed, exposure time, for long-time exposures, the point spread function can be described reasonably well by a Gaussian function

$$h(x, y, \sigma_G) = C \exp\left(-\frac{x^2 + y^2}{2\sigma_G^2}\right) \quad (6.5)$$

Here, σ_G determines the amount of spread of the blur, and the constant C is to be chosen so that Eq. (6.5) is satisfied.

6.4 CLASSIFICATION OF IMAGE-RESTORATION TECHNIQUES

Image-restoration techniques are methods which attempt the inversion of some degrading process. Image-restoration technique can be broadly classified into two types depending upon the knowledge of degradation. If the prior knowledge about degradation is known then the deterministic method of image restoration can be employed. If it is not known then the stochastic method of image restoration has to be employed. Classical linear techniques restore the true image by filtering the observed image using a properly designed filter. Examples are inverse filtering, Wiener filtering and the Tikhonov–Miller algorithm. Restoration often exhibits ringing artifacts near the edges, as linear methods are unable to recover missing frequency components which lead to the Gibbs effect. Linear methods do not necessarily

maintain image non-negativity or signal-dependent noise. This has led to the development of non-linear and iterative restoration algorithms. The classification of image-restoration techniques is shown in Fig. 6.3.

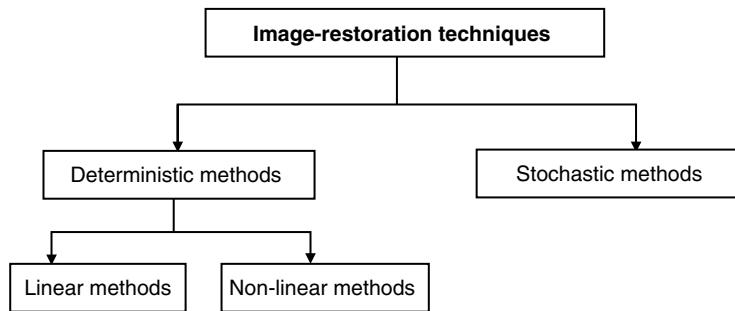


Fig. 6.3 Classification of image-restoration techniques

6.5 IMAGE-RESTORATION MODEL

In the image restoration model shown in Fig. 6.4, the input image is represented as $f(m, n)$, and $h(m, n)$ represents the degradation. $\hat{f}(m, n)$ represents the restored image and $\eta(m, n)$ represents the additive noise. The block $g(m, n)$ represents the restoration function. The error is given by $f(m, n) - \hat{f}(m, n)$. The objective of the restoration technique is to minimise the error so that $\hat{f}(m, n)$ resembles $f(m, n)$. The digital image restoration can be viewed as the process of estimating $\hat{f}(m, n)$.

In the spatial domain, the degraded image $g(m, n)$ is created through the convolution of the input image $f(m, n)$ with the linear operator $h(m, n)$ of the degradation system. It is represented as

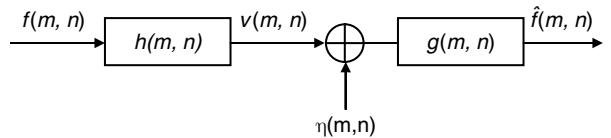


Fig. 6.4 Image-restoration model

$$g(m, n) = h(m, n) * f(m, n) \quad (6.6)$$

where $*$ denotes the convolution operation. The representation in the frequency domain is given by

$$G(k, l) = H(k, l) \times F(k, l) \quad (6.7)$$

Where $G(k, l)$, $H(k, l)$ and $F(k, l)$ are the Fourier transform of $g(m, n)$, $h(m, n)$ and $f(m, n)$ respectively.

6.5.1 Point-spread Functions

In a linear model, the point-spread function h models the blurring of the image. This process is not reversible. Convolution with the point-spread function can be reversed if the spectrum of h has no zeros in it.

In optics, $H(k, l)$ is called the *optical transfer function* (OTF), and its magnitude is called the *modulation transfer function*. In optics, $h(m, n)$, the inverse of the optical transfer function, is called the *point-spread*

function (PSF). The restoration techniques are oriented toward mathematically modeling the degradation and applying the inverse process to recover the original image.

6.6 LINEAR IMAGE-RESTORATION TECHNIQUES

The linear image-restoration techniques that we are going to discuss in this section are (a) inverse filter, (b) pseudo-inverse filter, (c) wiener filter, and (d) constrained least-square filter. The linear-restoration techniques are quick and simple but have limited capabilities.

6.6.1 Inverse Filtering

The inverse filter is a straightforward image-restoration method. If we know the exact point-spread function (PSF) model in the image-degradation system and ignore the noise effect, the degraded image can be restored using the inverse filter approach. In practice, the PSF models of the blurred images are usually unknown and the degraded process is also affected by the noise, so the restoration result with inverse filter is not usually perfect. But the major advantage of inverse-filter-based image restoration is that it is simple. From the image-degradation model shown in Fig. 6.4, the following equation can be derived:

$$g(m, n) = f(m, n) * h(m, n) + \eta(m, n) \quad (6.8)$$

Here, $f(m, n)$ represents the original image; $h(m, n)$ represents the degradation system; $\eta(m, n)$ is the additive noise term and $g(m, n)$ is the degraded image.

For simplicity, the coordinates of the image are ignored so that the Eq. (6.8) is given as

$$g = Hf + \eta \quad (6.9)$$

From Eq. (6.9), the error function is given by

$$\eta = g - Hf \quad (6.10)$$

Here, we wish to ignore η and use \hat{f} to let $H\hat{f}$ approximate it under the least square sense. Then the error function is given by

$$J(\hat{f}) = \|g - H\hat{f}\|^2 \quad (6.11)$$

Here, we wish to obtain minimum $J(f)$ and \hat{f} is not constrained in any other way—hence it can be termed unconstrained restoration. Equation (6.11) can be written as

$$J(\hat{f}) = \|g - H\hat{f}\|^2 = g^2 + H^2\hat{f}^2 - 2gH\hat{f} \quad (6.12)$$

In order to find the minimum of $J(\hat{f})$, Eq. (6.12) is differentiated with respect to \hat{f} , and equating it to zero, we get

$$\frac{\partial J(\hat{f})}{\partial \hat{f}} = 0 + 2H^2\hat{f} - 2gH = -2H^T(g - H\hat{f}) \quad (6.13)$$

Solving Eq. (6.13) for \hat{f} , we get

$$\hat{f} = (H^T H)^{-1} H^T g = H^{-1} g \quad (6.14)$$

Taking Fourier transform on both sides of Eq. (6.14), we get

$$\hat{F}(k, l) = \frac{G(k, l)}{H(k, l)} \quad (6.15)$$

The restored image in the spatial domain is obtained by taking the inverse Fourier transform of Eq. (6.15):

$$\hat{f}(m, n) = \mathfrak{I}^{-1} [\hat{F}(k, l)] = \mathfrak{I}^{-1} \left[\frac{G(k, l)}{H(k, l)} \right] \quad (6.16)$$

The advantage of inverse filter is that it requires only the blur point-spread function as *a priori* knowledge. The inverse filter produces perfect reconstruction in the absence of noise.

MATLAB Example 2 *Read an image, then blur the image. Then degrade the image by means of a known blur. Apply the inverse filter to the blurred image and see the restored image.*

Solution The MATLAB code which performs the inverse filtering operation is shown in Fig. 6.5 and the corresponding output is shown in Fig. 6.6.

```
%This program performs inverse filtering
close all;
clear all;
clc;
x = imread(C:\Documents and Settings\esakki\Desktop\flower2.jpg);
x = double(rgb2gray(x));
[M N] = size(x);
h = ones(11, 11)/121;
sigma = sqrt(4*10^(-7));
freqx = fft2(x);% Fourier transform of input image
freqh = fft2(h,M,N);%Fourier transform of degradation
y = real(ifft2(freqh.*freqx));
freqy = fft2(y);
powfreqx = freqx.^2/(M*N);
alpha = 0.5;%Indicates inverse filter
freqg = ((freqh)).*abs(powfreqx) ...
    ./ (abs(freqh).^2).*abs(powfreqx)+ alpha*sigma^2;
Resfreqx = freqg.*freqy;
Resa = real(ifft2(Resfreqx));
imshow(uint8(x)), title(Original Image)
Fig., imshow(uint8(y)), title(Degraded Image)
Fig., imshow(uint8(Resa)), title(Restored Image)
```

Fig. 6.5 MATLAB code which performs inverse filtering operation

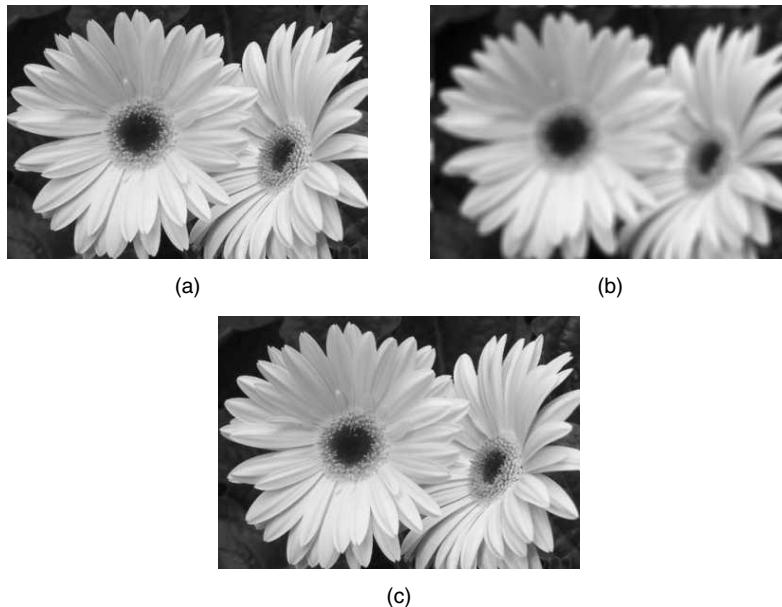


Fig. 6.6 (a) Original image (b) Degraded image (c) Restored image

In this case, noise is added to the image. In the absence of noise, inverse filter is a simple and effective method to minimise the degradation present in the image.

Drawbacks of Inverse Filtering The main drawback of inverse filtering is that it is not always possible to obtain an inverse. For an inverse to exist, the matrix should be non-singular. In case, it is difficult to obtain inverse filtering, the choice is to use a pseudo-inverse filter.

Another main drawback of an inverse filter is that an inverse filter will not perform well in the presence of noise. If noise is present in the image, the inverse filter will tend to amplify noise which is undesirable. In the presence of noise, it is better to go for a Wiener filter.

MATLAB Example 3 Read an image, then degrade the image. Then add Additive White Gaussian Noise (AWGN) to the degraded image. The image is now degraded as well as corrupted by noise. Then apply inverse filter to restore the image. Comment on the observed result.

Solution The MATLAB code that performs the inverse filtering of the image which is degraded and corrupted by noise is shown in Fig. 6.7 and the corresponding output is shown in Fig. 6.8.

From Fig. 6.8, it is obvious that the image is degraded as well as corrupted by noise.

From Fig. 6.8, it is obvious that whenever noise is present in the input image, an inverse filter fails to restore the image. To overcome this drawback, use of a pseudo-inverse filter is proposed.

```
%This program performs inverse filtering
close all;
clear all;
clc;
x = imread(C:\Documents and Settings\esakki\Desktop\flower2.jpg);
x = double(rgb2gray(x));
[M N] = size(x);
h = ones(11, 11)/121;
sigma = sqrt(4*10^(-7));
freqx = fft2(x); % Fourier transform of input image
freqh = fft2(h, M, N);%Fourier transform of degradation
y = real(ifft2(freqh.*freqx))+ 10*randn(M, N);
freqy = fft2(y);
powfreqx = freqx.^2/ (M*N);
alpha = 0.5;%Indicates inverse filter
freqg = ((freqh).*abs(powfreqx) ...
    ./ (abs(freqh.^2).*abs(powfreqx)+ alpha*sigma^2);
Resfreqx = freqg.*freqy;
Resa = real(ifft2(Resfreqx));
imshow(uint8(x)), title(Original Image)
Fig., imshow(uint8(y)), title(Degraded Image)
Fig., imshow(uint8(Resa)), title(Restored Image)
```

Fig. 6.7 MATLAB code for Example 3

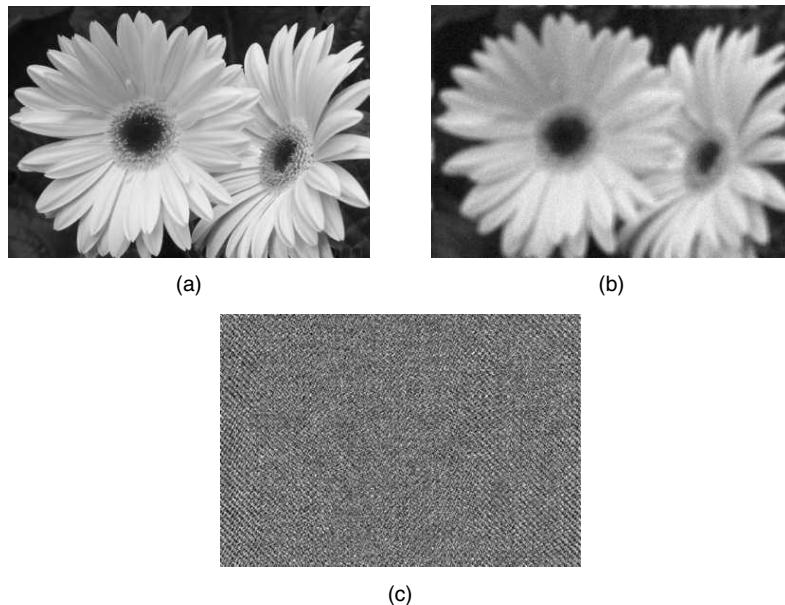


Fig. 6.8 (a) Original image (b) Image degrade with noise (c) Restored image

6.6.2 Pseudo-Inverse Filter

The equation of an inverse filter in the frequency domain is given by $\hat{F}(k, l) = \frac{G(k, l)}{H(k, l)}$. Here, $H(k, l)$ represents the spectrum of point-spread function. Mostly, the point-spread function is a low-pass filter, which implies that $H(K, l) \approx 0$ at high frequencies. The division of $H(k, l)$ leads to large amplification at high frequencies, where the noise dominates over the image. This frequency-dependent amplification leads to significant errors in the restored image, and amplification of noise. To avoid these problems, a pseudo-inverse filter is defined as

$$\frac{1}{H} = \begin{cases} 1/H & \text{if } H > \varepsilon \\ \varepsilon & \text{if } H \leq \varepsilon \end{cases} \quad (6.17)$$

The value of ε affects the restored image. With no clear objective selection of ε , restored images are generally noisy and not suitable for further analysis.

MATLAB Example 4 *Read an image, degrade the image, and then add Additive White Gaussian Noise to the image. Apply a pseudo-inverse filter to the resultant image and observe the result.*

Solution The MATLAB code that performs pseudo-inverse filtering on the image which is degraded and corrupted by noise is shown in Fig. 6.9 and the corresponding output is shown in Fig. 6.10.

```
%This program is Pseudo inverse filtering
close all;
clear all;
clc;
x = imread('C:\Documents and Settings\esakki\Desktop\flower2.jpg');
x = double(rgb2gray(x));
Thr_Freq = 0.2;
mask_b = ones(11,11)/121;
[M N] = size(x);
[m1 n1] = size(mask_b);
freqa = fft2(x);
freqh = fft2(mask_b,M,N);
blurr_img = real(ifft2(freqh.*freqa))+ 25*randn(M,N);
in_aspec = fft2(blurr_img);
psf = zeros(M, N);
psf(M/2 + 1 - (m1 - 1)/2:...
M/2 + 1 + (m1 - 1)/2, N/2 + 1 - (n1 - 1)/2:N/2 + 1 + (n1 - 1)/2) = mask_b;
psf = fftshift(psf);
freq_res = fft2(psf);
Inv_filt = freq_res./((abs(freq_res)).^2 + Thr_Freq);
y = real(ifft2(in_aspec.*Inv_filt));
imshow(uint8(x)), title('Original Image')
Fig., imshow(uint8(blurr_img)), title('Degraded Image')
Fig., imshow(uint8(y)), title('Restored Image')
```

Fig. 6.9 MATLAB code for a pseudo-inverse filter

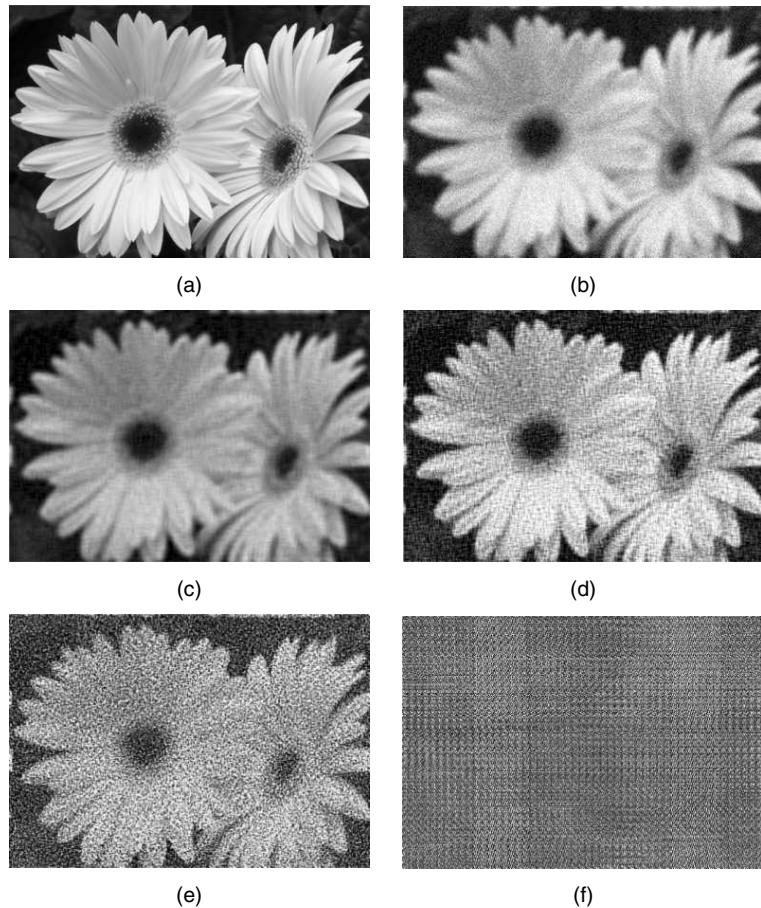


Fig. 6.10 (a) Original image (b) Image degraded with noise (c) Restored image with $\varepsilon = 0.2$ (d) Restored image with $\varepsilon = 0.02$ (e) Restored image with $\varepsilon = 0.002$ (f) Restored image with $\varepsilon = 0$

From Fig. 6.9, it is obvious that a parameter `Thr_freq` is defined in the MATLAB code. This parameter affects the performance of the inverse filter. It is to be noted that when the parameter `Thr_freq` approaches zero, the behaviour of the pseudo-inverse filter approaches the behaviour of an inverse filter.

Figure 6.10 shows the restored image for different values of the parameter `Thr_freq` which is given as ε in theory. From the figure, it is evident that as the value of ε or `Thr_freq` approaches zero, the behaviour of the pseudo-inverse filter matches with that of the inverse filter.

6.6.3 SVD Approach to Pseudo-Inverse Filtering

SVD stands for Singular Value Decomposition. Using the technique of SVD, any matrix can be decomposed into a series of eigen matrices. The basic strategy in SVD-based image restoration is to decompose the blur matrix into eigen matrices. From the image-restoration model, we have

$$G = Hf + \eta \quad (6.18)$$

The blur matrix is represented by H . Now the matrix H is decomposed into eigen matrices through SVD. Using SVD, the matrix H can be written as

$$H = UDV^T \quad (6.19)$$

The matrices U and V are unitary, and D is a diagonal matrix. It is easy to write down the pseudo-inverse of H as

$$H^\dagger = VD^{-1}U^T = \sum_{i=1}^R [D(i)]^{-1} v_i u_i^T \quad (6.20)$$

The generalised inverse estimate is the result of multiplying this expression for H^\dagger on the right by g . R indicates the rank of the matrix. Also, since $u_i^T g$ is a scalar, it can be moved around like a scalar, and the resulting sequential estimation formula is given by

$$\hat{f}_k = \hat{f}_{k-1} + [D(i)]^{-1} (u_i^T g) v_k$$

Advantages of SVD-Based Approach This method is effective with problems of ill-conditioning, since it is possible to interactively terminate the expansion before numerical problems result. However, the presence of noise in the observed image can often lead to numerical instability. Considerable computational savings can be achieved if the blur is spatially invariant.

6.6.4 Wiener Filter

The Wiener filter tries to build an optimal estimate of the original image by enforcing a minimum mean-square error constraint between estimate and original image. The Wiener filter is an optimum filter. The objective of a Wiener filter is to minimise the mean square error. A Wiener filter has the capability of handling both the degradation function as well as noise.

From the image-degradation model shown in Fig. 6.11, the error between the input signal $f(m, n)$ and the estimated signal $\hat{f}(m, n)$ is given by

$$e(m, n) = f(m, n) - \hat{f}(m, n)$$

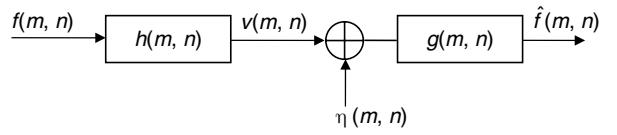


Fig. 6.11 Image-degradation model

The square error is given by $[f(m, n) - \hat{f}(m, n)]^2$

The mean square error is given by $E\{[f(m, n) - \hat{f}(m, n)]^2\}$

The objective of the Wiener filter is to minimise $E\{[f(m, n) - \hat{f}(m, n)]^2\}$.

According to the principle of orthogonality,

$$E\{f(m, n) - \hat{f}(m, n)v(k', l')\} = 0 \quad (6.21)$$

$$\text{But } \hat{f}(m, n) = v(m, n) * g(m, n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(m-k, n-l)v(k, l) \quad (6.22)$$

Substituting Eq. (6.22) in Eq. (6.21), we get

$$E \left\{ \left[f(m, n) - \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(m-k, n-l) v(k, l) \right] v(k', l') \right\} = 0 \quad (6.23)$$

$$E \{ f(m, n) v(k', l') \} = E \left\{ \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(m-k, n-l) v(k, l) v(k', l') \right\} \quad (6.24)$$

$$r_{fv}(m, n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(m-k, n-l) E \{ v(k, l) v(k', l') \} \quad (6.25)$$

where $r_{fv}(m, n)$ represents the cross correlation between $f(m, n)$ and $v(m, n)$.

$$r_{fv}(m, n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} g(m-k, n-l) r_{vv}(k, l) \quad (6.26)$$

$$r_{fv}(m, n) = g(m, n) * r_{vv}(m, n) \quad (6.27)$$

Taking Fourier transform on both sides, we get

$$S_{fv}(\omega_1, \omega_2) = G(\omega_1, \omega_2) \times S_{vv}(\omega_1, \omega_2) \quad (6.28)$$

where, $S_{fv}(\omega_1, \omega_2)$, $S_{vv}(\omega_1, \omega_2)$ represents the power spectral density. From Eq. (6.28),

$$\frac{S_{fv}(\omega_1, \omega_2)}{S_{vv}(\omega_1, \omega_2)} = G(\omega_1, \omega_2) \quad (6.29)$$

The goal is now to determine $S_{fv}(\omega_1, \omega_2)$ and $S_{vv}(\omega_1, \omega_2)$.

To Determine $S_{fv}(\omega_1, \omega_2)$ To determine $S_{fv}(\omega_1, \omega_2)$, consider the block diagram as shown in Fig. 6.12.

The cross-correlation between the signals f and v is given by

$$r_{fv}(l_1, l_2) = E \{ f(m + l_1, n + l_2) v^*(m, n) \} \quad (6.30)$$

But,

$$v(m, n) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h^*(k_1, k_2) f(m-k_1, n-k_2)$$

This implies

$$v^*(m, n) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h^*(k_1, k_2) f^*(m-k_1, n-k_2) \quad (6.31)$$

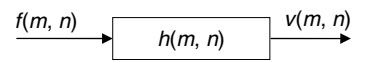


Fig. 6.12 Transfer function

Substituting Eq. (6.31) in Eq. (6.30), we get

$$r_{fv}(l_1, l_2) = E \left\{ f(m + l_1, n + l_2) \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h^*(k_1, k_2) f^*(m - k_1, n - k_2) \right\} \quad (6.32)$$

$$= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h^*(k_1, k_2) E \left\{ f(m + l_1, n + l_2) f^*(m - k_1, n - k_2) \right\} \quad (6.33)$$

$$= \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h^*(k_1, k_2) r_{ff}(l_1 + k_1, l_2 + k_2) \quad (6.34)$$

Let $k_1 = -m_1$ and $k_2 = -m_2$. Substituting this in Eq. (6.34), we get

$$= \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} h^*(-m_1, -m_2) r_{ff}(l_1 - m_1, l_2 - m_2) \quad (6.35)$$

Equation (6.35) resembles the equation of convolution. Hence, it can be written as

$$r_{fv}(l_1, l_2) = h^*(-l_1, -l_2) * r_{ff}(l_1, l_2) \quad (6.36)$$

Taking Fourier transform on both sides, we get

$$S_{fv}(\omega_1, \omega_2) = H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2) \quad (6.37)$$

To Determine $S_{vv}(\omega_1, \omega_2)$

The autocorrelation of the output is given by

$$r_{vv}(l_1, l_2) = r_{hh}(l_1, l_2) * r_{ff}(l_1, l_2) \quad (6.38)$$

$$r_{vv}(l_1, l_2) = h(l_1, l_2) * h^*(-l_1, -l_2) * r_{ff}(l_1, l_2) \quad (6.39)$$

Taking Fourier transform on both sides, we get

$$S_{vv}(\omega_1, \omega_2) = H(\omega_1, \omega_2) \times H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2) \quad (6.40)$$

$$S_{vv}(\omega_1, \omega_2) = |H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2) \quad (6.41)$$

If the noise is taken into consideration then the expression becomes

$$S_{vv}(\omega_1, \omega_2) = |H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2) + S_{\eta\eta}(\omega_1, \omega_2) \quad (6.42)$$

Now substituting,

$$S_{fv}(\omega_1, \omega_2) = H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)$$

and $S_{vv}(\omega_1, \omega_2) = |H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2) + S_{\eta\eta}(\omega_1, \omega_2)$ in the expression for $G(\omega_1, \omega_2)$ given in Eq. (6.29), we get

$$G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2) + S_{\eta\eta}(\omega_1, \omega_2)} \quad (6.43)$$

Case (1) If the impact of noise is not taken into consideration then $S_{\eta\eta} = 0$. Substituting $S_{\eta\eta} = 0$ in $G(\omega_1, \omega_2)$ given in Eq. (6.43), we get

$$G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2)} \quad (6.44)$$

$$G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)}{H(\omega_1, \omega_2) H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)} \quad (6.45)$$

$G(\omega_1, \omega_2) = \frac{1}{H(\omega_1, \omega_2)}$. Thus Wiener filter reduces to an inverse filter if the impact of noise is not taken into consideration.

Case (2) If $H(\omega_1, \omega_2) = 1$ then the expression of $G(\omega_1, \omega_2)$ given in Eq. (6.44) reduces to

$$G(\omega_1, \omega_2) = \frac{S_{ff}(\omega_1, \omega_2)}{S_{ff}(\omega_1, \omega_2) + S_{\eta\eta}(\omega_1, \omega_2)} \quad (6.46)$$

Divide the numerator and denominator by $S_{\eta\eta}(\omega_1, \omega_2)$ then the above equation is reduced to

$$G(\omega_1, \omega_2) = \frac{\frac{S_{ff}(\omega_1, \omega_2)}{S_{\eta\eta}(\omega_1, \omega_2)}}{\frac{S_{ff}(\omega_1, \omega_2)}{S_{\eta\eta}(\omega_1, \omega_2)} + 1} \quad (6.47)$$

Here, $\frac{S_{ff}}{S_{\eta\eta}}$ stands for signal-to-noise ratio. If the signal-to-noise ratio is very much greater than unity then the Eq. (6.47) is reduced to

$$G(\omega_1, \omega_2) = 1 \quad (6.48)$$

Thus, Wiener filter act as an all-pass filter.

MATLAB Example 5 Write a MATLAB code that reads an image and degrades it. For this degraded image, add (i) AWGN, and (ii) Gaussian noise. Then apply Wiener filter to restore the image. Comment on the observed output.

Solution The MATLAB code that performs Wiener filtering of the degraded image is shown in Fig. 6.13 and the corresponding output is shown in Fig. 6.14.

From Fig. 6.13, it is clear that the input image is first degraded then it is corrupted by a random noise. From Fig. 6.14, it is obvious that the Wiener filter is effective in minimising degradation and to a certain extent, it minimises the random noise.

Drawbacks of Wiener Filter The Wiener filter requires a prior knowledge of the power spectral density of original image which is unavailable in practice.

```
%This program is a wiener filter
close all;
clear all;
clc;
x = imread('C:\Documents and Settings\esakki\Desktop\flower2.jpg');
x = double(rgb2gray(x));
sigma = 50;
gamma = 1;
alpha = 1;% It indicates Wiener filter
[M N] = size(x);
h = ones(5, 5)/25;
Freqa = fft2(x);
Freqh = fft2(h, M, N);
y = real(ifft2(Freqh.*Freqa))+25*randn(M, N);
Freqy = fft2(y);
Powy = abs(Freqy).^2/(M*N);
sFreqh = Freqh.* (abs(Freqh)>0)+1/gamma* (abs(Freqh)==0);
iFreqh = 1./sFreqh;
iFreqh = iFreqh.* (abs(Freqh)*gamma>1)...
+gamma*abs(sFreqh).*iFreqh.* (abs(sFreqh)*gamma<=1);
Powy = Powy.* (Powy>sigma^2)+sigma^2*(Powy<=sigma^2);
Freqg = iFreqh.* (Powy-sigma^2)./(Powy-(1-alpha)*sigma^2);
ResFreqa = Freqg.*Freqy;
Resa = real(ifft2(ResFreqa));
imshow(uint8(x)), title('Original Image')
Fig., imshow(uint8(y)), title('Degraded Image')
Fig., imshow(uint8(Resa)), title('Restored Image')
```

Fig. 6.13 MATLAB code that performs Wiener filtering of the corrupted image

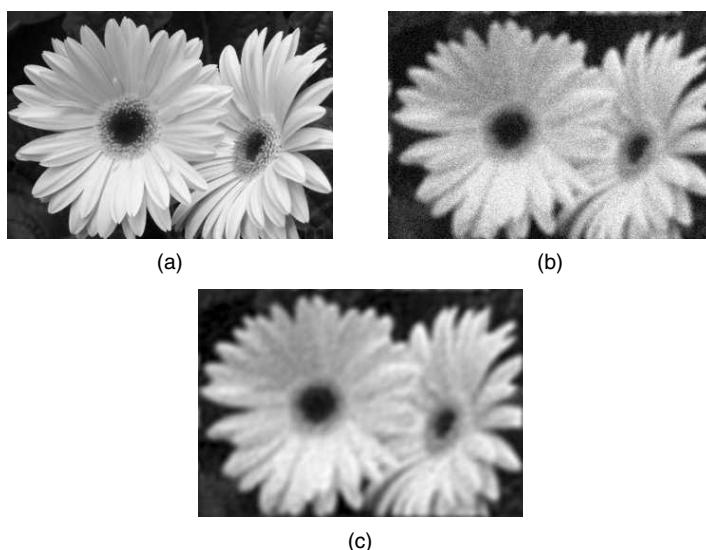


Fig. 6.14 (a) Original image (b) Degraded image with noise (c) Restored image

6.6.5 Constrained Least Square Filter

The effect of information loss in the degraded image can often be mitigated by constraining the restoration. Constraints have the effect of adding information to the restoration process. If these constraints represent additional knowledge of the original scene to be recovered then this knowledge should contribute to a more faithful restoration of the image. Constrained restoration refers to the process of obtaining a meaningful restoration by biasing the solution toward the minimiser of some specified constraint functional. Constrained least-square filter is a regularisation technique which adds the Lagrange multiplier, λ , to control the balance between noise artifacts and consistency with the observed data.

The constrained least square reconstruction is given by

$$\hat{F}(k, l) = \left[\frac{H^*(k, l)}{|H(k, l)|^2 + \lambda |P(k, l)|^2} \right] G(k, l) \quad (6.49)$$

Here, $P(k, l)$ is the Fourier transform of the Laplacian filter. The filter $P(k, l)$ has a large amplitude at high frequencies, where the noise tends to be dominant. It modifies the denominator to reduce the noise effects at high frequencies. Proper choice of $P(k, l)$ and λ can minimise higher-order derivatives. The iterations have to be monitored to prevent noise amplification and various restoration parameters modified throughout the process.

6.7 NON-LINEAR IMAGE-RESTORATION TECHNIQUES

A linear image-restoration algorithm is capable of producing a result directly from the observed data but requires some form of inverse operator to be applied. Non-linear technique does not explicitly implement the inverse; instead it uses an iterative approach to produce successive improvement to the restoration until a termination condition is reached. Non-linear restoration can introduce spatial adaptivity, impose constraints such as non-negativity, recover missing spatial or frequency components, and cope with non-Gaussian noise.

6.7.1 Iterative Method of Image Restoration

In the iterative method of image restoration, Landweber iteration is derived by solving the least-squares solution iteratively instead of directly. The error metric is given by

$$e = \|g - H\hat{f}\|^2 \quad (6.50)$$

$$e = (g - H\hat{f})^T (g - H\hat{f}) \quad (6.51)$$

The partial derivative of the error metric with respect to image estimate is given by

$$\frac{\partial e}{\partial \hat{f}} = -2H^T (g - H\hat{f}) \quad (6.52)$$

The objective is to minimise the error which can be done through gradient descent approach given by

$$\hat{f}_0 = \beta H^T g \quad (6.53)$$

$$\hat{f}_{k+1} = \hat{f}_k + \beta p_k \quad (6.54)$$

$$\text{where } p_k = H^T (g - H f_k) \quad (6.55)$$

where $0 < \beta < \frac{2}{|\lambda_{\max}|}$. Here, λ_{\max} is the largest eigen value of $H^T H$. If β is constant and no constraints are imposed then the result after N iterations is equivalent to a single filtering step which can be represented in Fourier domain as

$$\hat{F}_N(k, l) = \left[1 - \left(1 - \beta |H(k, l)|^2 \right)^{N+1} \right] \times \frac{G(k, l)}{H(k, l)} \quad (6.56)$$

However, if a positive constraint is imposed on the solution after each iteration then the algorithm becomes non-linear. Alternatively, a steepest descent approach can be taken to optimise the step length which is given by

$$\beta_k = \frac{p_k^T p_k}{(H p_k)^T H p_k} \quad (6.57)$$

To increase the rate of convergence, a conjugate gradient method can be applied to the algorithm. Regularisation can be accomplished by simply terminating the restoration process before the noise amplification becomes prevalent.

The flow chart of iterative image restoration is given in Fig. 6.15.

From the flow chart shown in Fig. 6.15, the following conclusions can be drawn.

The iterative image-restoration procedure starts with an initial estimate of the original image. It then compares the observed image with the deblurred image and the error is used to update the estimate. The iterative restoration algorithm can be terminated after the specified conditions are met.

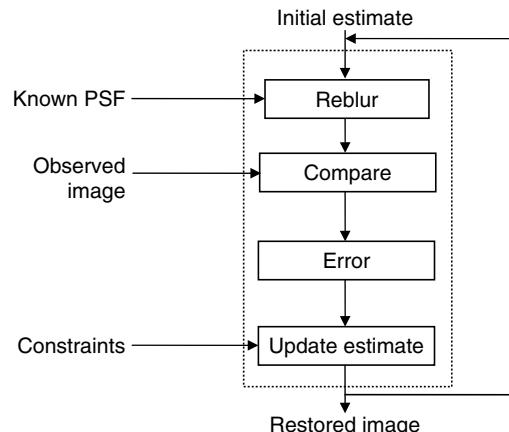


Fig. 6.15 Flow chart of iterative image-restoration procedure

6.7.2 Iterative Constrained Least-Square Image Restoration

Iterative constrained least-square restoration technique improves the Landweber iteration by incorporating a regularisation parameter. The algorithm is given below:

$$\hat{f}_0 = \beta H^T H \quad (6.58)$$

$$\hat{f}_{k+1} = \hat{f}_k + \beta p_k \quad (6.59)$$

$$p_k = H^T g - (H^T H + \lambda C^T C) \hat{f}_k \quad (6.60)$$

The algorithm converges when $0 < \beta < \frac{2}{|\lambda_{\max}|}$ and $\lambda = \frac{\|n\|^2}{\|f\|^2}$. Here, λ_{\max} is the largest eigen value of the matrix $H^T H + \lambda C^T C$. The value of λ can be calculated iteratively and updated during the restoration. The image estimate is constrained to be positive and after each iteration, the conjugate gradient algorithm can be applied to accelerate convergence.

Advantage of Iterative Image-Restoration Technique The advantages of iterative image-restoration technique are summarised below:

1. Iterative algorithms have greater flexibility compared to linear methods in incorporating non-linear constraints and preventing the over-fitting noise components.
2. The iterative method does not require the convolution of images with 2D point-spread function (PSF) containing many coefficients. The only convolution is that of the restored image with the PSF of the blur, which has relatively few coefficients.
3. There is no need to take transform in iterative scheme, which makes it applicable to images of arbitrary size.
4. The iteration can be terminated whenever an acceptable restoration result has been achieved.
5. The advantage of iterative method of image restoration is that these methods can be easily extended for spatially variant restoration that is the restoration where either the PSF of the blur or the models of the image vary locally.

Drawbacks of Iterative Image Restoration The computational demand is the major drawback of the iterative image-restoration technique. Iterative techniques require many times the computation of the recursive method. If computation is not a factor, iterative techniques are generally preferable to other methods of image restoration.

6.7.3 Maximum Likelihood Method of Image Restoration

The maximum likelihood of image restoration is based on knowledge of the random properties of the observed image. Let the observed image be denoted by g . If the PSF is known, the probability density function $P_r(g|f)$, the likelihood, is a function only of the true image f . Then, the problem of image restoration is to estimate the unknown parameters $f(X)$, $\forall X \in S_f$, where S_f is the support of f . The maximum likelihood estimate is the image f which is most likely to give rise to the observed image g . The log-likelihood function is often used for mathematical convenience. The ML solution is found by solving

$$\hat{f}(x, y) = \arg \max_{\hat{f}(x, y)} \sum_{x, y} \log(p(g(x, y) | \hat{f}(x, y), h(x, y))) + \log(p(\hat{f}(x, y), h(x, y))) \quad (6.61)$$

Here, the first term $p(g(x, y) | \hat{f}(x, y), h(x, y))$ is the goodness-of-fit measure which requires a statistical model of the image-formation process. The second term is the image-prior term which is independent of the observed data, incorporates any knowledge about the form of the solution, and is generally used to regularise the solution.

Maximum-likelihood Methods for Poisson Statistics In applications that involve counting events, such as photons in astronomy and positron emission tomography (PET), the statistics at each point in the observed image are best described by a Poisson distribution:

$$p(x / \mu) = \frac{e^{-\mu} \mu^x}{x!} \quad (6.62)$$

where μ is the mean value of the random event x . The probability of the observed image $g(x, y)$, given the reblurred image and $r(x, y)$ formed from the current estimate $\hat{f}(x, y)$, is given by

$$\begin{aligned} p(g(x, y) | \hat{f}(x, y)) &= p(g(x, y) | r(x, y)) \\ &= \frac{e^{-r(x, y)} r(x, y)^{g(x, y)}}{g(x, y)!} \end{aligned} \quad (6.63)$$

Taking logarithm on both sides of Eq. (6.63), we get

$$\log(p(g(x, y) | \hat{f}(x, y))) = -r(x, y) + g(x, y) \log(r(x, y)) - \log(g(x, y)!) \quad (6.64)$$

The restoration technique assumes no prior knowledge about the image. Taking the logarithm of the likelihood function and estimating terms independent of $\hat{f}(x, y)$ leaves the equation given below to be maximised:

$$\log(L) = \sum_{x, y} \log(p(g(x, y) | \hat{f}(x, y))) = \sum_{x, y} g(x, y) \log(r(x, y)) - r(x, y) \quad (6.65)$$

The maximisation can be achieved through the expectation-maximisation (EM) algorithm producing the iterative scheme

$$\hat{f}_{k+1}(x, y) = \hat{f}_k(x, y) \left(h(x, y) * \frac{g(x, y)}{r_k(x, y)} \right) \quad (6.66)$$

where $*$ is the correlation operator.

6.7.4 Stochastic Image-Restoration Technique

The stochastic image-restoration technique uses the probability theory to form the ‘most likely’ result, given the observed data. It uses the philosophy of Bayesian inference to construct the most statistically valid result in an attempt to eliminate the heuristics associated with deterministic methods. Restoration algorithms developed within the Bayesian framework are distinguished from other algorithms by the inclusion of prior knowledge about the true image in the form of a prior probability distribution over images.

To tackle the problem of image restoration within the Bayesian framework, the first step is to construct the prior probability distribution $P_r(f)$. The probability density function $P_r(f)$ represents the knowledge about

the true image. $P_r(f)$ assigns a high probability to solutions that agree with our prior knowledge about the true image. The prior distribution is constructed independent of the observed image. Using Bayes' theorem, this prior distribution can be modified, based on the observation model, into a posterior distribution. According to Bayes' theorem, the posterior can be calculated as

$$P_r(f|g) = \frac{P_r(g|f)P_r(f)}{P_r(g)} \quad (6.67)$$

where the evidence $P_r(g)$ depends on the observed image only and can be regarded as a normalising constant, the likelihood $P_r(g|f)$ depends on the observation model and $P_r(f)$ is the prior. The mode of the posterior distribution is often selected as the estimated true image.

6.8 BLIND DECONVOLUTION

The blind deconvolution is a powerful tool to restore images with little or without any priori knowledge about the point-spread function that blurred the image.

6.9 CLASSIFICATION OF BLIND-DECONVOLUTION TECHNIQUES

The classification of blind-deconvolution techniques is shown in Fig. 6.16.

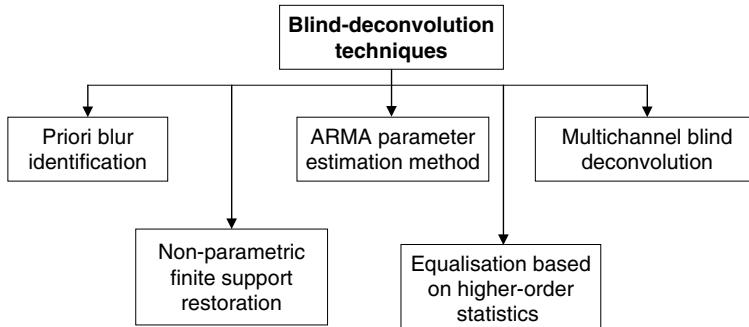


Fig. 6.16 Classification of blind-deconvolution technique

6.9.1 Prior Blur Identification Method

The point-spread function is often identified from the degraded-image characteristics before restoration. Some commonly used identification techniques assume that a parametric model for the PSF is known. In particular, the blurring may be known by the result from camera motion or an out-of-focus lens. For each of these PSFs, a single parameter is used in the model, and the associated power spectrum of the PSF is known to contain zeros at locations related to this parameter value. Assuming that the power spectrum of the image has no zeros, an estimate of the PSFs parameter value is made using the location of the zeros of the power spectrum of the degraded image. A major drawback of this method is that it is highly sensitive to noise.

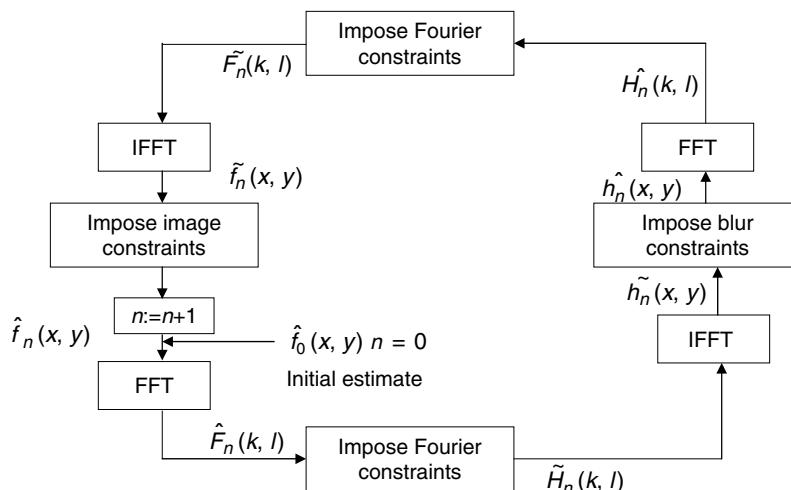
6.9.2 Non-parametric Finite Support Restoration

The algorithms in non-parametric finite support restoration do not assume parametric models for either the image or the blur. The image is assumed to be positive, and comprised of an object with known finite support against a uniformly black, gray or white background. The support refers to the smallest rectangle within which the object is completely encompassed. If the background is black, which corresponds to a pixel value of zero, the support is the smallest rectangle within which the true image pixels are non-zero. The three most popular non-parametric finite support restoration methods are (i) iterative blind-deconvolution method, (ii) conjugate gradient method, and (iii) simulated annealing method.

(i) Iterative Blind-Deconvolution Method The iterative blind deconvolution method was proposed by Ayers and Dainty. The basic structure of the iterative deconvolution method is shown in Fig. 6.17.

In Fig. 6.17, the image estimate is denoted by $\hat{f}(x, y)$ and the point-spread function estimate is denoted by $\hat{h}(x, y)$. The subscript n denotes the iteration number of the algorithm. After a random initial guess is made for the image, the algorithm alternates between the image and Fourier domains, enforcing known constraints in each domain. The constraints are based on information available about the image and PSF. The image $f(x, y)$ and PSF $h(x, y)$ are assumed to be positive with finite known support. The image-domain constraints are imposed by replacing negative-valued pixels within the region of support and non-zero pixels outside the region of support with zero-valued pixels. The Fourier domain constraint involves estimating the PSF using the FFT of the degraded image and image estimate. At the n^{th} iteration we have

$$\hat{H}_n(k, l) = \frac{G(k, l)\hat{F}_{n-1}^*(k, l)}{\left|\hat{F}_{n-1}(k, l)\right|^2 + \frac{\alpha}{\left|\hat{H}_{n-1}(k, l)\right|^2}} \quad (6.68)$$



$$\hat{F}_n(k, l) = \frac{G(k, l)\hat{H}_{n-1}^*(k, l)}{\left|\hat{H}_{n-1}(k, l)\right|^2 + \frac{\alpha}{\left|\hat{F}_{n-1}(k, l)\right|^2}} \quad (6.69)$$

The real constant α represents the energy of the additive noise and is determined by prior knowledge of the noise-contamination level, if available. The value of α must be chosen carefully for reliable restoration.

Initial Estimate When the PSF is known, the initial image estimate is the result of the correlation between the observed data and the PSF. When the PSF is unknown, and unless additional information is available, the initial image estimate used is the observed image. The initial PSF estimate is more difficult to approximate accurately.

The algorithm can be terminated after a specified number of iterations or until the estimate begins to converge.

Advantage of Iterative Blind-Deconvolution Method The computational complexity of the iterative blind-deconvolution method is low. This method is robust to noise.

Disadvantage of Iterative Blind-Deconvolution Method The major drawback of the iterative blind-deconvolution method is lack of reliability. This method is sensitive to initial image estimate and the algorithm often exhibits instability.

(ii) **Conjugate Gradient Method**

The conjugate method was proposed by Lane to overcome the problems associated with the instability of the iterative blind-deconvolution method. The image and PSF are assumed to be non-negative with known finite support. The procedure involves the minimisation of the cost function given in Eq. (6.70) using the conjugate gradient optimisation routine.

$$J(\hat{f}(x, y), \hat{h}(x, y)) = \sum_{x, y \in \gamma_f} \hat{f}^2(x, y) + \sum_{x, y \in \gamma_h} \hat{h}^2(x, y) + \sum_{\forall(k, l)} \left| G(k, l) - \hat{F}(k, l)\hat{H}(k, l) \right|^2 \quad (6.70)$$

where γ_f and γ_h represent pixels for which the image and PSF estimates violate their known constraints. The algorithm attempts to minimise eq. (6.70) by varying the pixels corresponding to the image estimate $\hat{f}(x, y)$ and PSF estimate $\hat{h}(x, y)$ with arbitrary initial conditions for all pixels.

Advantages of Conjugate Gradient Method The main advantages of the conjugate gradient method are the following:

- (i) The algorithm has low computational complexity.
- (ii) The algorithm is robust to noise.

Drawbacks of Conjugate Gradient Method The main drawbacks of the conjugate gradient method are the following:

- (i) The conjugate gradient method suffers from incorrect convergence to local minima.
- (ii) The main reason for convergence to local minima is that the cost function J is often multimodal.
- (iii) For realistic images, it is very difficult to achieve proper convergence.

$$j(\hat{f}(g(x, y) \mid \hat{h}(x, y))) = -r(x, y) + g(x, y)$$

(iii) Simulated Annealing Method of Image Deconvolution

The simulated annealing approach was developed by McCallum. The simulated annealing method attempts to attain the minimum of the cost function given by

$$J(\hat{f}(x, y), \hat{h}(x, y)) = \sum_{\forall(x, y)} [f(x, y) * h(x, y) - g(x, y)]^2 \quad (6.71)$$

The image and PSF are assumed to be positive with known finite support. Using these constraints on $\hat{f}(x, y)$ and $\hat{h}(x, y)$, a simulated annealing procedure is employed for the minimisation of J . In simulated annealing, estimates of the cost function parameters are iteratively varied to globally minimise J . The parameter values are randomly perturbed. If the perturbation reduces J then it is accepted. If it increases J then it is accepted with a probability p . The probability p is reduced as iterations progress. In the case of infinite precision and infinite iterations, the procedure is guaranteed to reach the global minimum of a multimodal cost function.

Advantage of Simulated Annealing Approach The restoration algorithm using simulated annealing approach is reliable and provides reasonable results in the presence of noise.

Disadvantage of Simulated Annealing Approach The convergence of the simulated annealing approach to the global minimum of the cost function is slow. The speed of convergence of the algorithm depends to a large extent on how quickly p is reduced.

6.9.3 ARMA Parameter Estimation Methods

The parametric method of image restoration represents a significant departure from the non-parametric methods. While the non-parametric methods make no assumption about the image model, the parametric methods assume a model for the image that is fundamental to the method. Specifically the true image is modeled as a two-dimensional autoregressive (AR) process and the PSF as a two-dimensional linear system with finite impulse response. The degraded image is modeled as a noisy observation of an autoregressive moving average (ARMA) process. Thus, the blind deconvolution problem becomes an ARMA parameter identification problem. The ARMA parameters can be estimated using maximum likelihood estimation or generalised cross-validation. The main drawback of this approach is the model's insensitivity to abrupt changes in local image characteristics which often leads to ringing artifacts. Hence, the AR model is invalid for images with sharp edges.

6.9.4 Equalisation Method Based on Higher-Order Statistics

The equalisation method based on higher-order statistics depends on the minimisation of a given cost function which accounts for the probabilistic nature of the true image. The degraded image is passed through an inverse FIR filter yielding an estimate of the true image. The FIR filter parameters are updated in order to minimise a cost function which incorporates the higher-order statistics model of the true image.

The main advantage of the equalisation method based on higher order statistics is that it allows the identification of non-minimum phase PSFs. The major drawback of this method is that the true image must be accurately modeled by a known non-Gaussian probability distribution.

6.9.5 Multichannel Blind-Deconvolution Technique

The multichannel blind deconvolution technique deals with the situation in which differently blurred versions of the same image are available for processing. The most popular method in multichannel blind-deconvolution technique is a cepstrum-based higher-order statistics algorithm. This approach combines partial, higher-order cepstral information from two differently blurred frames to provide information about the individual PSFs. Using this information, the individual PSFs are reconstructed and an estimate of the true image is computed. The PSFs are assumed to be FIR sequences with no zeros in common with each other, as well as with the true image. Multichannel blind-deconvolution technique is robust to additive noise, and does not require any particular information about the image or PSF. The major limitation of this method is that it is computationally intensive.

6.10 IMAGE DENOISING

An image is often corrupted by noise in its acquisition or transmission. Noise is any undesired information that contaminates an image. Noise appears in images from a variety of sources. The goal of denoising is to remove the noise while retaining as much as possible the important signal features. Denoising can be done through filtering, which can be either linear filtering or non-linear filtering.

6.11 CLASSIFICATION OF NOISE IN IMAGE

Basically, there are three standard noise models which model the types of noise encountered in most images. They are additive noise, multiplicative noise and impulse noise.

6.11.1 Additive Noise

Let $f[m, n]$ be the original image, $f'[m, n]$ be the noise digitised version and $\eta[m, n]$ be the noise function which returns random values coming from an arbitrary distribution. Then the additive noise is given by the equation

$$f'[m, n] = f[m, n] + \eta[m, n] \quad (6.72)$$

Additive noise is independent of the pixel values in the original image. Typically, $\eta[m, n]$ is symmetric about zero. This has the effect of not altering the average brightness of the image. Additive noise is a good model for the thermal noise within photo-electronic sensors.

6.11.2 Multiplicative Noise

Multiplicative noise or speckle noise, is a signal-dependent form of noise whose magnitude is related to the value of the original pixel. The simple mathematical expression for a multiplicative noise model is given by

$$f'[m, n] = f[m, n] + \eta[m, n]f[m, n] \quad (6.73)$$

$$f'[m, n] = f[m, n][1 + \eta[m, n]] \quad (6.74)$$

6.11.3 Impulse Noise

Impulse noise has the property of either leaving a pixel unmodified with probability $1 - p$, or replacing it altogether with a probability p . Restricting $\eta[m, n]$ to produce only the extreme intensities 0 or 1 results in salt-and-pepper noise. The source of impulse noise is usually the result of an error in transmission or an atmospheric or man-made disturbance. The MATLAB code that adds salt-and-pepper noise to the input image and then performs the averaging operation on the noisy image is shown in Fig. 6.18. The result of performing 3×3 average filtering and 5×5 average filtering of the noisy image is shown in Fig. 6.19.

Comparing Fig. 6.19(b) with Fig. 6.19(d), it is found that the noise has been removed, and at the same time the image has got blurred.

As the size of the window increases, the ability to remove noise increases at the expense of blurring of the image.

This program is used to perform average filtering operation

```

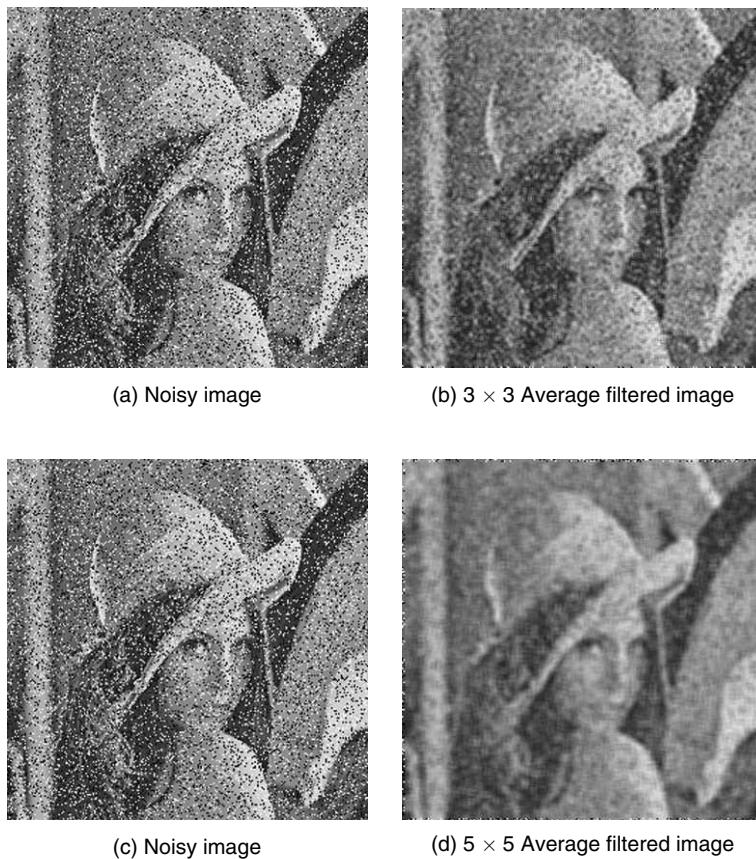
a = imread(lena.tif);
a = imresize(a,[256,256]);
a = imnoise(a,salt & pepper, 0.2);% Add salt & pepper noise to the image
a = double(a);
[m n] = size(a);
N=input('enter the window size=');% The window size can be 3 x 3, 5 x 5 etc
Start = (N+1)/2;
Out_Img=a;
for i = Start@m-Start+1),
for j = Start@n-Start+1),
    limit = (N-1)/2;
    Sum = 0;
    for k =-limit:limit,
        for l=-limit:limit,
            Sum=Sum+a(i+k, j+l);
        end
    end
    Out_Img(i, j)=Sum/(N*N);
end
end
imshow(uint8(a)), title('original Image'), Fig.,
imshow(uint8(Out_Img)), title('average filtered Image');

```

Fig.6.18 MATLAB code that performs averaging operation

6.12 MEDIAN FILTERING

A median filter is a non-linear filter and is efficient in removing salt-and-pepper noise. Median tends to preserve the sharpness of image edges while removing noise. The variants of median filters are (a) weighted median filter, (b) centre-weighted median filter, and (c) max-median filter.

**Fig. 6.19** *Results of average filtering*

The effect of increasing the size of the window in median filtering is illustrated in Fig. 6.20 and the corresponding MATLAB code is given in Fig. 6.21.

Comparing Fig. 6.20 (b), (d), (f) and (h), it is found that the noise is removed effectively as the size of the window increases. Also, the ability to suppress noise increases only at the expense of blurring of edges.

6.12.1 Median Filter for Colour Images

If a colour image is corrupted by salt-and-pepper noise, we cannot apply median filter directly. Instead, we divide the colour image into three planes (red plane, green plane and blue plane) and apply median filtering to the three planes individually so that the impact of salt-and-pepper noise is suppressed. The resulting MATLAB code that performs the median filtering of the colour image is shown in Fig. 6.22 and the corresponding MATLAB code is given in Fig. 6.23. From Fig. 6.22 (Plate 4), it is obvious that



(a) Original image



(b) 2×2 median-filtered image



(c) Original image



(d) 3×3 median-filtered image



(e) Original image



(f) 4×4 median-filtered image



(g) Original image



(h) 5×5 median-filtered image

Fig. 6.20 Effect of size of median filter window

```

% This program is to perform median filtering of the image
a = imread(cameraman.tif);
a = imnoise(a, salt & pepper,.2);
a = double(a);
[m n] = size(a);
N=input(enter the window size=);
Out_Img = a;
if(mod(N,2)==1)
Start =(N+1)/2;
End=Start;
else
    Start =N/2;
End=Start+1;
end
if(mod(N, 2)==1)
limit1 =(N-1)/2;
limit2 =limit1;
else
limit1 =(N/2)-1;
limit2 =limit1+1;
end
for i =Start:(m-End+1),
for j =Start:(n-End+1),
    I=1;
    for k =-limit1:limit2,
        for l =-limit1:limit2,
            mat(I) =a(i+k, j+l);
            I=I+1;
        end
    end
    mat=sort(mat); %To sort the elements in the window
    if(mod(N, 2)==1)
        Out_Img(i, j) =(mat((N^2)+1)/2));
    else
        Out_Img(i, j) =(mat((N^2)/2)+mat((N^2)/2+1))/2;
    end
end
imshow(uint8(a)), title(original Image), Fig.,
imshow(uint8(Out_Img)), title(median filtered Image);

```

Fig. 6.21 MATLAB code to perform median filtering

as the size of the median filter is increased, it can effectively remove the salt-and-pepper noise at the expense of blurring of edges.

The program in Fig. 6.23 (a) will work if the function *Func_medianall* is included. The function *Func_medianall* is used to apply median filters to the three (red, green and blue) planes individually.

```
%This program is to apply median filtering to colour image
a = imread(pepper.tif);
N = input(enter the window size);
b = imresize(a, [256, 256]);
b = imnoise(b, salt & pepper,.1);
[m n] = size(b);
R=b(:,:1); G=b(:,:2);B=b(:,:3);
Out_R=Func_medianall(R,N);%Applying Median filter to R plane
Out_G=Func_medianall(G,N);
Out_B=Func_medianall(B,N);
Out_Image(:,:1)=Out_R;
Out_Image(:,:2)=Out_G;
Out_Image(:,:3)=Out_B;
imshow(uint8(b)), title(noise added), Fig.,
imshow(uint8(Out_Image)), title(average filtered)
```

Fig 6.23 (a) MATLAB program to perform median filtering

The input to the function are the corrupted image a and the dimension

```
function [Out_Img]=Func_medianall(a,N)
a=double(a);
[m n]=size(a);
Out_Img=a;
if(mod(N,2)==1)
Start=(N+1)/2;
End=Start;
else
    Start=N/2;
    End=Start+1;
end
if(mod(N,2)==1)
    limit1=(N-1)/2;
    limit2=limit1;
else
    limit1=(N/2)-1;
    limit2=limit1+1;
end
for i=Start:(m-End+1),
    for j=Start:(n-End+1),
        I=1;
        for k=-limit1:limit2,
            for l=-limit1:limit2,
                mat(I)=a(i+k, j+l);
                I=I+1;
            end
        end
    end
end
```

```

mat=sort(mat); %Sort the elements to find the
median
if(mod(N, 2)==1)
    Out_Img(i, j)=(mat(((N^2)+1)/2));
else
    Out_Img(i, j)=(mat((N^2)/2)+mat(((N^2)/2)+1))/2;
end
end
end

```

Fig. 6.23 (b) Function to compute the median value

6.13 TRIMMED AVERAGE FILTER

The average filter at a spatial location (x, y) on the image is defined by finding the average of the interested pixel (centre pixel) with its 8 (if 3×3 window is used) near neighbours. The centre pixel is assigned to the average value. The same averaging is applied all over the image for all pixels which results in the averaging filter operation of the image. The significance of average filtering lies in its impulse-noise-suppression capability. Along with noise removal, it creates a blurring effect as an unintended side effect. Blurring can be controlled by reducing the number of ‘interested’ neighbour pixels. The solution for blurring converges towards ‘eliminating’ a few neighbouring pixels based on the statistical relationships for the calculation of the average value. Objectives converge towards trimming the ‘un-interested pixels’ from the centre pixel based on the statistical criteria.

Trimming Criteria The extreme minimum value and extreme maximum value in the windowed pixels will affect the average value calculated. So by gradually trimming the pixels from both minimum extreme and maximum extreme, the optimum performance can be achieved. Consecutive extreme values were identified and trimmed by ordering the windowed pixels in ascending order.

R and S Factors

R and S factors are used to quantify the number of pixels to be trimmed on both the extremes.

R-Factor R is the number of consecutive pixels to be trimmed from the minimum extreme.

Let a be the windowed and ordered list.

$a = [10, 20, 30, 40, 50, 60, 70, 80, 90]$

If $R = 1$ (one minimum pixel to be trimmed),

after trimming, a becomes $[20, 30, 40, 50, 60, 70, 80, 90]$.

If $R = 2$ (two consecutive minimum pixels to be trimmed),

after trimming, a becomes $[30, 40, 50, 60, 70, 80, 90]$.

S-Factor S is the number of consecutive pixels to be trimmed from both the minimum extreme and maximum extreme.

Let a be the windowed and ordered list.

$a = [10, 20, 30, 40, 50, 60, 70, 80, 90]$

If $S = 1$ (minimum pixel, maximum to be trimmed),
after trimming, a becomes $[20, 30, 40, 50, 60, 70, 80]$.

If $S = 2$ (two consecutive minimum and maximum pixels to be trimmed),
after trimming, a becomes $[30, 40, 50, 60, 70]$.

In a trimmed average filter, we have the flexibility of retaining specific values. The MATLAB code that performs trimmed average filter is shown in Fig. 6.24 and the corresponding output is shown in Fig. 6.25.

Hint In the above program, S and R are the variables. By changing those variables we can see the impact in the output image.

From Fig. 6.25, the impact of changing the values of S and R are visible.

MAX-MIN filter Maximum and minimum filters attribute to each pixel in an image a new value equal to the maximum or minimum value in a neighbourhood around that pixel. The neighbourhood stands for the shape of the filter. Maximum and minimum filters have been used in contrast enhancement.

```
%This program is to perform Trimmed Average filter
a=imread(girl512.bmp);
a=imresize(a,[256,256]);
s=1;      % S denotes the number of values to be left in the end
r=1;
N=9;      % Because 3*3 window
a=double(imnoise(a, gaussian));
[m n]=size(a);
b=zeros(m, n);
for i=2:m-1,
for j=2:n-1,
    mat=[a(i,j), a(i,j-1), a(i,j+1), a(i-1,j), a(i+1,j),
    a(i-1,j-1),...
    a(i-1, j+1), a(i-1, j+1), a(i+1, j+1)];
    Sorted_mat=sort(mat);
    Sum=0;
    for k=r+s:(N-s),
        Sum=Sum+mat(k);
    end
    b(i,j)=Sum/(N-r-s);
end
end
imshow(uint8(a)), title(original image), fig.,
imshow(uint8(b)), title(Trimmed averaging filter);
```

Fig. 6.24 Alpha trimmed average filter



(a) Image corrupted by Gaussian noise



(b) Trimmed average filter with $s = 1, r = 1$



(a) Image corrupted by Gaussian noise



(b) Trimmed average filter with $s = 2, r = 1$



(a) Image corrupted by Gaussian noise



(b) Trimmed average filter with $s = 3, r = 1$

Fig. 6.25 Results of trimmed average filter

6.14 PERFORMANCE METRICS IN IMAGE RESTORATION

Two popular metrics widely used in the image-restoration field are (i) Blurred Signal-to-Noise Ratio (BSNR), and (ii) Improvement in Signal-to-Noise Ratio (ISNR).

6.14.1 Blurred Signal-to-Noise Ratio (BSNR)

The degradation modeled by blur and additive noise is referred by a metric called Blurred Signal-to-Noise Ratio (BSNR).

If $f(m, n)$ represents the $M \times N$ original image and if it is blurred by the point-spread function $h(m, n)$ resulting in $y[m, n]$ then the expression for BSNR is given by

$$\text{BSNR} = 10 \log \left(\frac{\text{blurred image power}}{\text{noise variance}} \right) \quad (6.75)$$

$$\text{BSNR} = 10 \log_{10} \left\{ \frac{\frac{1}{MN} \sum_{m, n} [y[m, n] - \bar{y}[m, n]]^2}{\sigma_n^2} \right\} \quad (6.76)$$

where σ_n^2 represents the additive noise variance and $\bar{y}[m, n]$ represents the mean of $y[m, n]$.

6.14.2 Improvement in Signal-to-Noise Ratio (ISNR)

Improvement in Signal-to-Noise Ratio is used to test the performance of the image-restoration algorithm objectively. If $f[m, n]$ and $y[m, n]$ represent the original and the degraded image, the expression of ISNR is given by

$$\text{ISNR} = 10 \log_{10} \left\{ \frac{\sum_{m, n} [f[m, n] - y[m, n]]^2}{\sum_{m, n} [f[m, n] - \hat{f}[m, n]]^2} \right\} \quad (6.77)$$

Here, $\hat{f}[m, n]$ is the restored image. This metric can be used for simulation purposes only because the original image is assumed to be available which is not true practically.

6.15 APPLICATIONS OF DIGITAL IMAGE RESTORATION

The digital image-restoration technique is widely used in the following fields of image processing:

1. Astronomical imaging
2. Medical imaging
3. Printing industry

4. Defense applications

6.15.1 Astronomical Imaging

Astronomical imaging was one of the pioneering applications that drove much of the development of early image-restoration techniques. The technical challenges faced by astronomers in gathering and interpreting images and spectra forced the image-processing community to develop efficient image-restoration algorithms. Some of the problems include imaging through turbulent atmosphere, imaging of faint objects with low photon counts and thus noisy data and the loss of phase information in radio interferometry. Astronomical image degradations are characterised by Poisson noise. Poisson noise is signal dependent. Another type of noise that is common is Gaussian noise which is due to electronic components in the imaging system and broadcast transmission effect. Astronomical images are degraded by motion blur which is due to slow camera shutter speeds relative to rapid spacecraft motion. To minimise the noise as well as degradation, astronomical imaging is one of the primary applications of digital image restoration.

6.15.2 Medical Imaging

X-rays, mammograms and digital angiographic images are often corrupted by Poisson noise. Additive noise is common in magnetic resonance imaging. These noises should be removed for proper diagnostics of diseases. This can be accomplished through digital image-restoration techniques.

6.15.3 Printing Applications

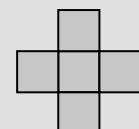
Printing applications often require the use of restoration to ensure that halftone reproductions of continuous images are of high quality. Image restoration can improve the quality of continuous images generated from halftone images.

6.15.4 Defense Applications

The image obtained from guided missiles may be degraded due to the effects of pressure differences around a camera mounted on the missile. Proper image-restoration technique is a must to restore these images.

SOLVED PROBLEMS

1. A 4×4 image is given by
$$\begin{bmatrix} 3 & 2 & 1 & 4 \\ 5 & 2 & 6 & 3 \\ 7 & 9 & 1 & 4 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$
. Filter this image using a median filter with the filter mask as given in Fig. 6.26. Assume replicate padding.

**Fig. 6.26** Filter mask

The first step is to perform replicate padding. After replicate padding, the input image is shown in Fig. 6.27.

The second step is to compute the median using the given mask.

The median of the pixel 3 marked in Fig. 6.28 is calculated as follows:

As per the given mask, we have to consider only the horizontal and the vertical neighbours and not the diagonal neighbours. The four neighbours including the rounded pixel as shown in Fig. 6.28 are 3, 3, 3, 2 and 5.

$$\begin{bmatrix} 3 & 3 & 2 & 1 & 4 & 4 \\ 3 & 3 & 2 & 1 & 4 & 4 \\ 5 & 5 & 2 & 6 & 3 & 3 \\ 7 & 7 & 9 & 1 & 4 & 4 \\ 2 & 2 & 4 & 6 & 8 & 8 \\ 2 & 2 & 4 & 6 & 8 & 8 \end{bmatrix}$$

Fig. 6.27 Input image after replicate padding

$$\begin{bmatrix} 3 & 3 & 2 & 1 & 4 & 4 \\ 3 & \textcircled{3} & 2 & 1 & 4 & 4 \\ 5 & 5 & 2 & 6 & 3 & 3 \\ 7 & 7 & 9 & 1 & 4 & 4 \\ 2 & 2 & 4 & 6 & 8 & 8 \\ 2 & 2 & 4 & 6 & 8 & 8 \end{bmatrix}$$

Fig. 6.28 Computation of median of marked pixel

The pixels are then arranged in the descending order as 5 3 3 3 2. The median value is then 3. Similarly, the median value of the other pixels are calculated and the median filtered image is shown in Fig. 6.29.

$$\begin{bmatrix} 3 & 2 & 2 & 4 \\ 5 & 5 & 2 & 4 \\ 5 & 4 & 6 & 4 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$

Fig 6.29 Median-filtered image

2. Image blurring caused by long-term exposure to atmospheric turbulence can be modeled by the transfer function $H(k, l) = \exp\left\{-\left(k^2 + l^2\right)/\left(2\sigma^2\right)\right\}$. Assume negligible noise. What is the equation of the Wiener filter to restore an image blurred by this type of degradation?

The equation of the Wiener filter is given by

$$G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2) + S_{\eta\eta}(\omega_1, \omega_2)}. \text{ In this problem, the noise is assumed to be negligible, hence } S_{\eta\eta}(\omega_1, \omega_2) = 0. \text{ Substituting } S_{\eta\eta}(\omega_1, \omega_2) = 0 \text{ in the expression of } G(\omega_1, \omega_2), \text{ we get}$$

$$G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2} = \frac{1}{H(\omega_1, \omega_2)}. \text{ Thus, Wiener filter is reduced to an inverse filter. This implies that the equation of the Wiener filter is } G(k, l) = \exp\left\{\left(k^2 + l^2\right)/\left(2\sigma^2\right)\right\}.$$

3. A blur filter $h(m, n)$ is given by $h(m, n) = \begin{bmatrix} 0 & 0.1 & 0.1 & 0 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.05 & 0.1 & 0.1 & 0.05 \\ 0 & 0.05 & 0.05 & 0 \end{bmatrix}$. Find the deblur filter using

- (a) inverse filter approach, (b) pseudo-inverse filter approach with $\varepsilon = 0.05$, and (c) pseudo-inverse filter approach with $\varepsilon = 0.2$.

We have to determine the Fourier transform of $h(m, n)$ to get $H(k, l)$.

The kernel of the Fourier transform is

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

The 2D Fourier transform is obtained as $H(k, l) = \text{kernel} \times h(m, n) \times (\text{kernel})^T$

$$H(k, l) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 0 & 0.1 & 0.1 & 0 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.05 & 0.1 & 0.1 & 0.05 \\ 0 & 0.05 & 0.05 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$H(k, l) = \begin{bmatrix} 1 & -0.2 - 0.2j & 0 & -0.2 + 0.2j \\ -0.1 - 0.3j & -0.1j & 0 & -0.1 \\ 0 & -0.1 - 0.1j & 0 & -0.1 + 0.1j \\ -0.1 + 0.3j & -0.1 & 0 & 0.1j \end{bmatrix}$$

(a) Inverse filter

The inverse filter $G(k, l) = \frac{1}{H(k, l)}$ which is given by

$$G(k, l) = \begin{bmatrix} 1 & -2.5 + 2.5j & \text{Inf} & -2.5 - 2.5j \\ -1 + 3j & 10j & \text{Inf} & -10 \\ \text{Inf} & -5 + 5j & \text{Inf} & -5 - 5j \\ -1 - 3j & -10 & \text{Inf} & -10j \end{bmatrix}$$

(b) Pseudo-inverse filter with $\varepsilon = 0.05$

The pseudo-inverse filter is defined as

$$G(k, l) = \begin{cases} \frac{1}{H(k, l)} & \text{if } |H(k, l)| > \varepsilon \\ \varepsilon & \text{if } |H(k, l)| \leq \varepsilon \end{cases}$$

Then we have to determine $|H(k, l)|$.

$|H(k, l)|$ is given by

$$|H(k, l)| = \begin{bmatrix} 1 & 0.2828 & 0 & 0.2828 \\ 0.316 & 0.1 & 0 & 0.1 \\ 0 & 0.1414 & 0 & 0.1414 \\ 0.316 & 0.1 & 0 & 0.1 \end{bmatrix}$$

Then $|H(k, l)|$ is compared with the value of $\varepsilon = 0.05$ to get the inverse filter as

$$G(k, l) = \begin{bmatrix} 1 & -2.5 + 2.5j & 0.05 & -2.5 - 2.5j \\ -1 + 3j & 10j & 0.05 & -10 \\ 0.05 & -5 + 5j & 0.05 & -5 - 5j \\ -1 - 3j & -10 & 0.05 & -10j \end{bmatrix}$$

(c) Pseudo-inverse filter with $\varepsilon = 0.2$

After imposing the condition $G(k, l) = \begin{cases} \frac{1}{H(k, l)} & \text{if } |H(k, l)| > \varepsilon \\ \varepsilon & \text{if } |H(k, l)| \leq \varepsilon \end{cases}$ with $\varepsilon = 0.2$, we get the pseudo inverse filter as

$$G(k, l) = \begin{bmatrix} 1 & -2.5 + 2.5j & 0.2 & -2.5 - 2.5j \\ -1 + 3j & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 \\ -1 - 3j & 0.2 & 0.2 & 0.2 \end{bmatrix}$$

4. A blur filter is given by $h(m, n) = \begin{bmatrix} 0 & 0.05 & 0.05 & 0 \\ 0.15 & 0.1 & 0.1 & 0.15 \\ 0 & 0.1 & 0.1 & 0 \\ 0 & 0.1 & 0.1 & 0 \end{bmatrix}$. Find the deblur filter using Wiener filter

approach with $\sigma_x^2 = 200$ and $\sigma_w^2 = 100$.

The deblur filter using Wiener filter approach is given by $G(k, l) = \frac{H^*(k, l)}{|H(k, l)|^2 + \frac{\sigma_w^2}{\sigma_x^2}}$.

Step 1 Determine the Fourier transform of $h(m, n)$ to get $H(k, l)$.

$$H(k, l) = \begin{bmatrix} 1 & -0.2 - 0.2j & 0 & -0.2 + 0.2j \\ -0.1 - 0.3j & 0.2 - 0.1j & 0 & -0.1 - 0.2j \\ -0.4 & -0.1 - 0.1j & 0 & -0.1 + 0.1j \\ -0.1 + 0.3j & -0.1 + 0.2j & 0 & 0.2 + 0.1j \end{bmatrix}$$

Step 2 Determine the conjugate of $H(k, l)$ to get $H^*(k, l)$.

$$H^*(k, l) = \begin{bmatrix} 1 & -0.2 + 0.2j & 0 & -0.2 - 0.2j \\ -0.1 + 0.3j & 0.2 + 0.1j & 0 & -0.1 + 0.2j \\ -0.4 & -0.1 + 0.1j & 0 & -0.1 - 0.1j \\ -0.1 - 0.3j & -0.1 - 0.2j & 0 & 0.2 - 0.1j \end{bmatrix}$$

Step 3 Determine $|H(k, l)|^2$.

$$|H(k, l)|^2 = \begin{bmatrix} 1 & 0.08 & 0 & 0.08 \\ 0.1 & 0.05 & 0 & 0.05 \\ 0.16 & 0.02 & 0 & 0.02 \\ 0.1 & 0.05 & 0 & 0.05 \end{bmatrix}$$

Step 4 Determine $|H(k, l)|^2 + \frac{\sigma_w^2}{\sigma_x^2}$

From the given data $\frac{\sigma_w^2}{\sigma_x^2} = \frac{100}{200} = 0.5$

$$|H(k, l)|^2 + 0.5 = \begin{bmatrix} 1.5 & 0.58 & 0.5 & 0.58 \\ 0.6 & 0.55 & 0.5 & 0.55 \\ 0.66 & 0.52 & 0.5 & 0.52 \\ 0.6 & 0.55 & 0.5 & 0.55 \end{bmatrix}$$

Step 5 Determine $G(k, l) = \frac{H^*(k, l)}{|H(k, l)|^2 + \frac{\sigma_w^2}{\sigma_x^2}}$

$$G(k, l) = \begin{bmatrix} 0.67 & -0.34 + 0.34j & 0 & -0.34 - 0.34j \\ -0.16 + 0.5j & 0.36 + 0.18j & 0 & -0.18 + 0.36j \\ -0.606 & -0.192 + 0.192j & 0 & -0.192 - 0.192j \\ -0.166 - 0.5j & -0.18 - 0.36j & 0 & 0.36 - 0.18j \end{bmatrix}$$

Summary



- The objective of image restoration is to compensate for defects which degrade an image. Degradations are due to motion blur, noise and camera misfocus.
- The deterministic method of image restoration can be effective if information about the degradation is known.
- The general model of image degradation is given by

$$g(m, n) = f(m, n) * h(m, n) + \eta(m, n)$$

where $f(m, n)$ represents the input image, $h(m, n)$ represents the point spread function and $\eta(m, n)$ represents the noise in the image.

- The inverse filter is an effective method to minimise the degradation in the absence of noise. The pseudo-inverse filter is an alternative to inverse filter if inverse does not exist.
- The wiener filter is an optimal filter with respect to minimum mean square error. It will reduce to an inverse filter in the absence of noise.
- Blind deconvolution is an effective method to restore an image if information about the degradation is not known.
- The different blind-deconvolution techniques include priori blur identification, non-parametric finite support identification, ARMA parameter estimation and multichannel blind deconvolution.
- A median filter is basically a non-linear filter. It is effective in minimising salt-and-pepper noise.
- The performance metrics related to image restoration are (i) Blind Signal to Noise Ratio (BSNR), and (ii) Improvement in Signal to Noise Ratio (ISNR).

Review Questions

1. List three main properties of a median filter.

The three main properties of median filters are the following

- (i) A median filter smoothens additive white noise.
- (ii) A median filter does not degrade edges.
- (iii) A median filter is effective in removing impulses.

2. What is the difference between image restoration and image enhancement? What do they have in common?

Image-restoration and the image-enhancement techniques aim at improving the image quality, and both the techniques can be performed in both spatial and frequency domains.

The difference between image enhancement and image restoration is given below:

Criterion	Enhancement	Restoration
Result evaluation	Subjective	Objective
Modelling of degradation	No	Yes
Use of prior knowledge	No	Yes

Image enhancement is largely a subjective process, which means that it is a heuristic procedure designed to manipulate an image in order to achieve the pleasing aspects of a viewer. On the other hand, image restoration involves formulating a criterion of goodness that will yield an optimal estimate of the desired result.

In image enhancement, the degradation is not usually modeled. Image restoration attempts to reconstruct or recover an image that has been degraded by using the priori knowledge of the degradation. That is, restoration techniques try to model the degradation and then apply the inverse process in order to recover the original image.

3. When will Wiener filter reduce to an inverse filter?

The frequency domain representation of a Wiener filter is given by

$$G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2) + S_{\eta\eta}(\omega_1, \omega_2)}$$

In the absence of noise, $S_{\eta\eta}(\omega_1, \omega_2) = 0$, the expression for a Wiener filter is given by $G(\omega_1, \omega_2) = \frac{1}{H(\omega_1, \omega_2)}$. Thus, a Wiener filter reduces to an inverse filter in the absence of noise.

4. When will a Constrained Least Square Filter (CLS) reduce to an inverse filter?

The frequency domain representation of a constrained least square filter is given by

$$\hat{F}(k, l) = \left[\frac{H^*(k, l)}{|H(k, l)|^2 + \lambda |P(k, l)|^2} \right] G(k, l)$$

If $P(k, l)$ is zero then $\hat{F}(k, l) = \frac{1}{H(k, l)}$.

5. What are the advantages of a Wiener filter over an inverse filter?

A Wiener filter is better than the inverse filter in the presence of noise because a Wiener filter uses a prior statistical knowledge of the noise field. The transfer function of the Wiener filter is chosen to minimise the mean square error using statistical information on both image and noise fields.

6. Comment on the choice of regularisation parameter used in constrained least square restoration technique?

If the regularisation parameter is large, it leads to more regularisation, and the restored image tends to have more ringing. With smaller values of the regularisation parameter, the restored image tends to have more amplified noise effects.

7. An image has been distorted by convolution by a space-invariant point-spread function. As a first step in removing the distortion, the image is Fourier transformed. Why do we choose the Fourier transform rather than the cosine, Hadamard or some other transform?

The Fourier transform is the only transform in which the convolution theorem holds. Therefore, by applying the Fourier transform to an image in which $f = g^*h$, the convolution operation is converted to multiplication that is $F = G.H$ which can be solved by inverse or Wiener filter methods.

8. Give the mathematical expression for a Wiener filter. Also, give the advantage and drawback of a Wiener filter over an inverse filter.

$$\text{The Wiener filter expression is } G(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2) \times S_{ff}(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 \times S_{ff}(\omega_1, \omega_2) + S_{\eta\eta}(\omega_1, \omega_2)}.$$

$S_{ff}(\omega_1, \omega_2)$ and $S_{\eta\eta}(\omega_1, \omega_2)$ are Fourier transforms of the image covariance and noise covariance over the ensemble of convolution problem. The main advantage of a Wiener filter is that it is stable in the presence of nulls in $H(\omega_1, \omega_2)$. It also reduces the effect of additive noise and produces best reconstruction in terms of the mean least square error. The main drawback of a Wiener filter is that to get an optimum solution, one should know the statistics of the image and of the noise.

9. A weather satellite is launched into geostationary orbit such that it looks down over the same part of the earth. Unfortunately, after launch, it is found to be rapidly spinning on its axis so that images taken with even the shortest exposure are rotationally blurred. Describe a system to restore the images.

In Cartesian coordinates, the point-spread function is space variable. Hence, it is necessary to view the problem in polar coordinates. In polar coordinates, the problem becomes one of convolution with a space-invariable point-spread function.

For each radius $r = r_0$, the problem is reduced to a circular convolution given by

$$f'(r_0, \theta) = g'(r_0, \theta) \text{**} h'(\theta)$$

where ** indicates circular convolution and $h'(\theta)$ is a boxcar function, which is zero if $|\theta| > \frac{\theta_r}{2}$ and one if $|\theta| < \frac{\theta_r}{2}$ where θ_r is the angle rotated through during the exposure. By taking Fourier transforms at each radius, the convolution problem can be converted to a multiplication problem and it is possible to recover an estimate g' using a Wiener filter.

After doing this at each r_0 , back interpolate onto a Cartesian grid to get the final restored image.

10. A photograph is taken through the front window of a car moving at constant velocity on a flat road. Because of the movement of the car, the image is distorted. State whether this image distortion is linear or non-linear. Also, find whether the point-spread function is space-invariant or not.

The degradation of the image is due to motion blur. The motion blur is linear but space-variant, due to the fact that perspective objects appear to move away from a central point.

11. A photograph is taken out of a side window of a car moving at a constant velocity of 80 km/hour. Why is it not possible to use an inverse or a Wiener filter in general to restore the blurring in this image?

In this case, objects are at different distances and are blurred at different amounts. As a result, the point-spread function is space variable. Hence, a Wiener filter or an inverse filter is not effective in minimising this blurring.

Problems

- 6.1 The blur filter $h(m, n)$ is given below:

$$h(m, n) = \begin{bmatrix} 0.0187 & 0.0563 & 0.0563 & 0.0187 \\ 0.0562 & 0.1187 & 0.1187 & 0.0562 \\ 0.0562 & 0.1187 & 0.1187 & 0.0562 \\ 0.0187 & 0.0563 & 0.0563 & 0.0187 \end{bmatrix}$$

Design the deblur filter ($G(k, l)$) using inverse filtering approach.

- 6.2 A 4×4 image is given by

$$\begin{bmatrix} 2 & 4 & 8 & 7 \\ 12 & 6 & 9 & 8 \\ 13 & 7 & 4 & 3 \\ 8 & 12 & 4 & 9 \end{bmatrix}$$

Filter the above image using (a) MIN filter, and (b) MAX filter using the filter mask given in Fig. 6.30. Assume replicate padding of the input image.

- 6.3 A blur filter is given by $h(m, n) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$. Design a deblur filter $G(k, l)$ using the pseudo-inverse filter approach with (a) $\epsilon = 0.05$, and (b) $\epsilon = 0.15$.

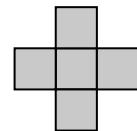


Fig. 6.30 Filter mask

- 6.4 A blur filter is given by $h(m, n) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$. Design a deblur filter $G(k, l)$ using the Wiener filter approach, given that the signal and the noise variances are 200 and 50 respectively.

- 6.5 The 2D DFT of the blur filter $h(m, n)$ is given by

$$H(k, l) = \begin{bmatrix} 1 & -0.2 - 0.2j & 0 & -0.2 + 0.2j \\ -0.2 - 0.2j & 0.05j & 0 & 0.05 \\ 0 & 0 & 0 & 0 \\ -0.2 + 0.2j & 0.05 & 0 & -0.05j \end{bmatrix}$$

Design the deblur filter using

- (a) Inverse filtering approach
 (b) Pseudo-inverse filter approach with $\epsilon = 0.1$ and $\epsilon = 0.5$

What will be the value of the marked central pixel after applying a 5×5 median filter?

$$\begin{bmatrix} 1 & 2 & 5 & 4 & 3 \\ 4 & 3 & 2 & 0 & 0 \\ 6 & 2 & \textcircled{4} & 1 & 0 \\ 4 & 3 & 6 & 2 & 1 \\ 2 & 5 & 6 & 1 & 3 \end{bmatrix}$$

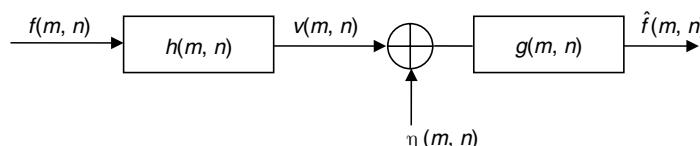
Derive a Wiener filter for image restoration using the minimum mean-square approach. Also, mention the situation in which the behavior of a Wiener filter resembles the behavior of Inverse filter.

- 6.6 Prove that a median filter is a non-linear filter.

- 6.7 Distinguish between deterministic and stochastic methods of image restoration.

- 6.8 For the image restoration model given below, show that the power spectrum of the output is given by

$$|G(k, l)|^2 = |H(k, l)|^2 |F(k, l)|^2 + |N(k, l)|^2$$



References

1. H C Andrews and B R Hunt, *Digital Image Restoration*, Prentice Hall, Sydney, 1977
2. A K Katsaggelos, *Digital Image Restoration*, Springer Series in Information Sciences, vol. 23, Springer-Verlag, 1991
3. A K Katsaggelos and N P Galatsanos, *Signal Recovery Techniques for Image and Video Compression and Transmission*, Kluwer Academic Publishers, 1998
4. H Stark (ed.), *Image Recovery: Theory and Application*, Academic Press, New York
5. R H T Bates and M J McDonell, *Image Restoration and Reconstruction*, Oxford: Clarendon Press, 1986

Journal Papers

1. Y M R Banham and A K Katsaggelos, *Digital Image Restoration*, IEEE Signal Processing Magazine, vol. 14, no. 2, pp. 24–41, March 1997
2. D Kundur and D Hatzinakos, *Blind Deconvolution*, IEEE Signal Processing Magazine, vol. 13, no. 3, pp. 43–64, May 1996
3. D Kundur, *Blind Deconvolution of Still Images using Recursive Inverse Filtering*, MSc Thesis, Dept. of Electrical and Computer Engineering, University of Toronto, May 1995
4. M M Sondhi, *Image Restoration: The Removal of Spatially Invariant Degradation*, Proc. IEEE, vol. 6, no. 7, pp. 842–853, August 1972
5. R Lagendijk, J Biemond and D Boekee, *Regularised Iterative Image Restoration with Ringing Reduction*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 36, no. 12, 1988
6. R G Lane, *Blind Deconvolution of Speckle Images*, Journal of Optical Society of America A", vol. 9, no. 9, pp. 1508–1514, September 1992

Web Resources

1. DIAL- Digital Image Analysis Laboratory: <http://www.ece.arizona.edu/~dial/>

7

Learning Objectives

The main objective of image segmentation is to extract various features of the image which can be merged or split in order to build objects of interest on which analysis and interpretation can be performed. Image segmentation represents the first step in image analysis and pattern recognition. After reading this chapter, the reader should have a basic idea of

Need of image segmentation

Classification of image-segmentation techniques

Region-based image-segmentation techniques

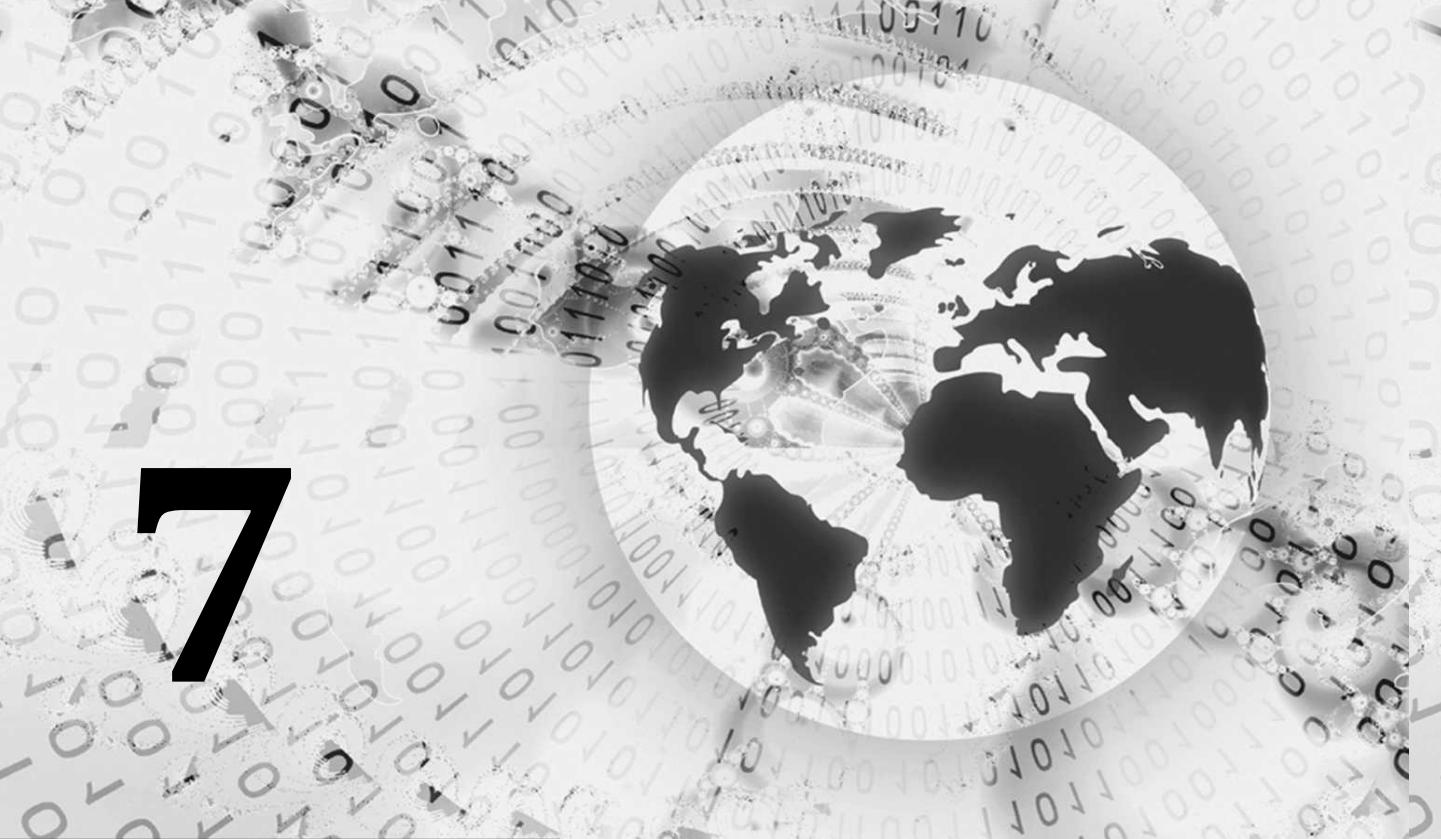
Boundary-based image-segmentation techniques

Edge-based image-segmentation

Image Segmentation

7.1 INTRODUCTION

Image segmentation refers to the process of partitioning an image into groups of pixels which are homogeneous with respect to some criterion. Different groups must not intersect with each other, and adjacent groups must be heterogeneous. Segmentation algorithms are area oriented instead of pixel-oriented. The result of segmentation



is the splitting up of the image into connected areas. Thus segmentation is concerned with dividing an image into meaningful regions.

7.2 CLASSIFICATION OF IMAGE-SEGMENTATION TECHNIQUES

Image segmentation can be broadly classified into two types: (i) local segmentation, and (ii) global segmentation.

7.2.1 Local Segmentation

Local segmentation deals with segmenting sub-images which are small windows on a whole image. The number of pixels available to local segmentation is much lower than global segmentation. Local segmentation must be frugal in its demands for pixel data.

7.2.3 Global Segmentation

Global segmentation is concerned with segmenting a whole image. Global segmentation deals mostly with segments consisting of a relatively large number of pixels. This makes estimated parameter values for global segments more robust.

Image segmentation can be approached from three different philosophical perspectives. They are (i) region approach, (ii) boundary approach, and (iii) edge approach as illustrated in Fig. 7.1.

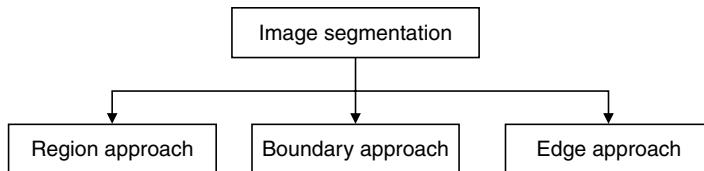


Fig. 7.1 Image-segmentation approaches

7.3 REGION APPROACH TO IMAGE SEGMENTATION

Regions in an image are a group of connected pixels with similar properties. In the region approach, each pixel is assigned to a particular object or region. In the boundary approach, the attempt is to locate directly the boundaries that exist between the regions. In the edge approach, the edges are identified first, and then they are linked together to form required boundaries.

All the three approaches are useful for visualising the problem and implementing a solution.

7.3.1 Region Growing

Region growing is an approach to image segmentation in which neighbouring pixels are examined and added to a region class if no edges are detected. This process is iterated for each boundary pixel in the region. If adjacent regions are found, a region-merging algorithm is used in which weak edges are dissolved and strong edges are left intact.

Region growing requires a seed to begin with. Ideally, the seed would be a region, but it could be a single pixel. A new segment is grown from the seed by assimilating as many neighbouring pixels as possible that

meet the homogeneity criterion. The resultant segment is then removed from the process. A new seed is chosen from the remaining pixels. This continues until all pixels have been allocated to a segment. As pixels are aggregated, the parameters for each segment have to be updated. The resulting segmentation could depend heavily on the initial seed chosen and the order in which neighbouring pixels are examined. The selection of homogeneity criteria in image growing depends not only on the problem under consideration but also on the type of image to be segmented. Region-growing algorithms vary depending on the criteria used to decide whether a pixel should be included in the region or not, the connectivity type used to determine neighbours, and the strategy used to visit neighbouring pixels.

Region growing offers several advantages over conventional segmentation techniques. The borders of regions found by region growing are perfectly thin and connected. The algorithm is also very stable with respect to noise.

7.3.2 Region Splitting

Region splitting is a top-down approach. It begins with a whole image and divides it up such that the segregated parts are more homogenous than the whole. Splitting alone is insufficient for reasonable segmentation, as it severely limits the shapes of segments. Hence, a merging phase after the splitting phase is always desirable, which is termed as the split-and-merge algorithm.

7.3.3 Region Splitting and Merging

Region splitting and merging is an image-segmentation technique that takes spatial information into consideration. The region-splitting-and-merging method is as follows:

Splitting

1. Let R represent the entire image. Select a predicate P .
2. Split or subdivide the image successively into smaller and smaller quadrant regions.

The splitting technique has a convenient representation in the form of a structure called a *quadtree* as shown in Fig. 7.2. In a quadtree, the root of the tree corresponds to the entire image and each node corresponds to a subdivision.

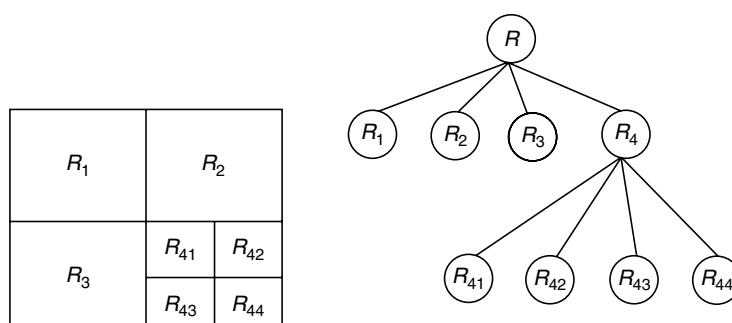


Fig. 7.2 (a) Splitting of an image (b) Representation by a quadtree

The final partition is likely to contain adjacent regions with identical properties. This drawback may be fixed by applying merging, and merging only adjacent regions whose combined pixels satisfy the predicate P .

Merging Merge any adjacent regions that are similar enough. The procedure for split and merge algorithm is given below:

1. Start with the whole image.
 2. If the variance is too large, break it into quadrants.
 3. Merge any adjacent regions that are similar enough.
 4. Repeat steps (2) and (3) iteratively until no more splitting or merging occurs.

Figure 7.2 (a) represents the image after split and merge. The quadtree representation of the image is shown in Fig. 7.2 (b).

Example 7.1 Apply the split-and-merge technique to segment the image shown in Fig. 7.3.

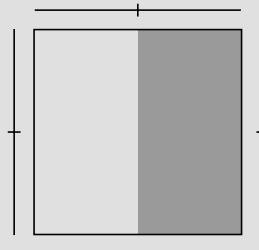


Fig. 7.3 *Image*

Solution First, the image is split up into four regions R_1 , R_2 , R_3 , and R_4 which is illustrated in Fig. 7.4(b). Then the regions which are homogenous are merged together. In this case, the regions R_1 and R_3 are homogeneous. Similarly, the regions R_2 and R_4 are homogeneous. Hence they are merged together which is illustrated in Fig. 7.4(c).

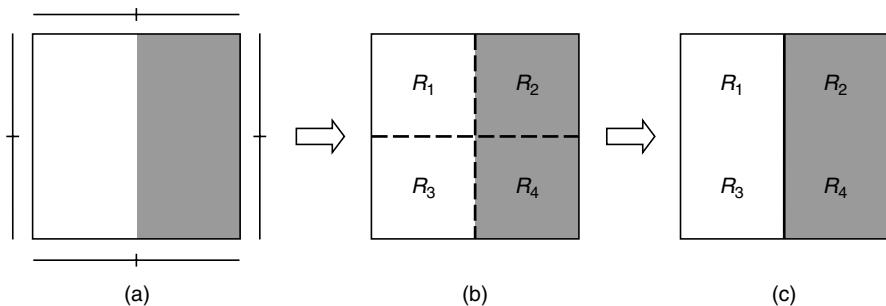


Fig. 7.4 Region split-and-merge

Example 7.2 Apply the split-and-merge technique to segment the image shown in Fig. 7.5.

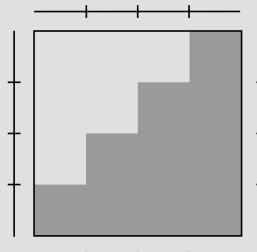


Fig. 7.5 Image

Solution First, the input image is split into four regions R_1 , R_2 , R_3 and R_4 . This is illustrated in Fig. 7.5(b). In Fig. 7.5(b), only two homogeneous regions are identified which are R_1 and R_4 . Hence, the regions R_1 and R_4 are not disturbed. The regions to be splitted further are R_2 and R_3 . The quadtree partitioning of the regions R_2 and R_3 is shown in Fig. 7.5(e). After this step, the homogeneous regions are merged together to get the result which is shown in Fig. 7.5(f). The quadtree representation of the procedure is shown in Fig. 7.6.

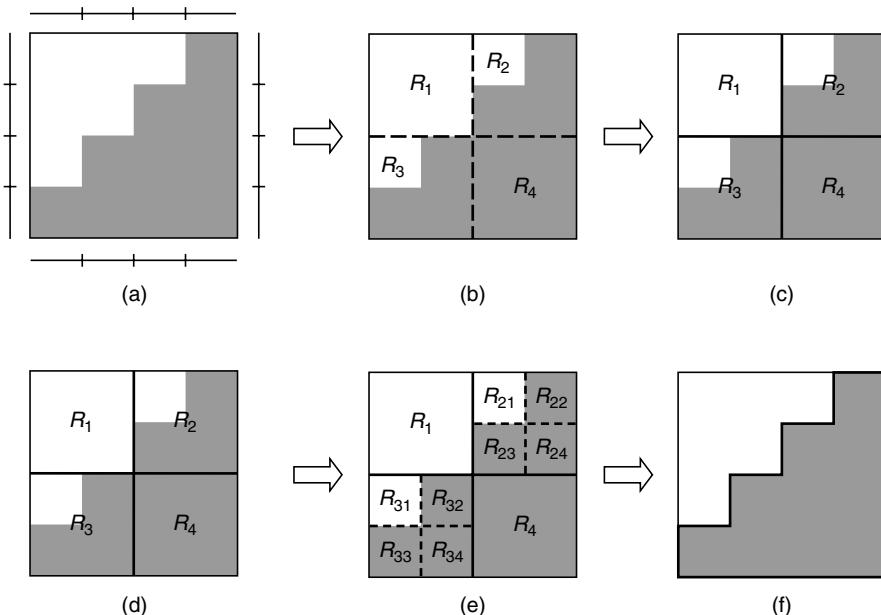


Fig. 7.5 Split-and-merge technique

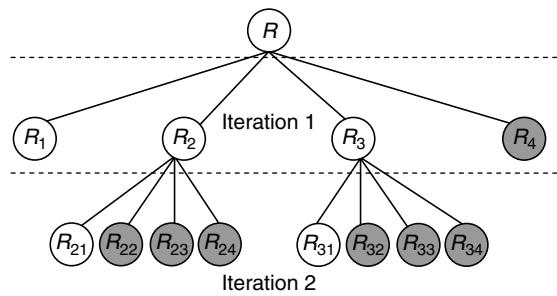


Fig. 7.6 Quadtree representation

Example 7.3 Segment the given arbitrary shape shown in Fig. 7.7 by the quadtree approach.

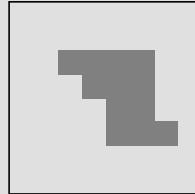


Fig. 7.7 Arbitrary shape

Solution The shape given in Fig. 7.7 can be segmented using the region split-merge algorithm. The different steps involved in the region-split-and-merge technique is given below.

Step 1

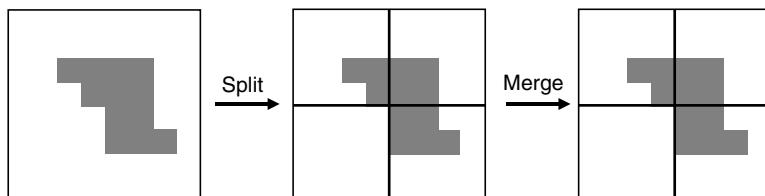


Fig. 7.8 Result after first iteration

In the first iteration, the given image is split into 4 regions. As the four regions are mutually exclusive, merging will not have any impact.

Step 2

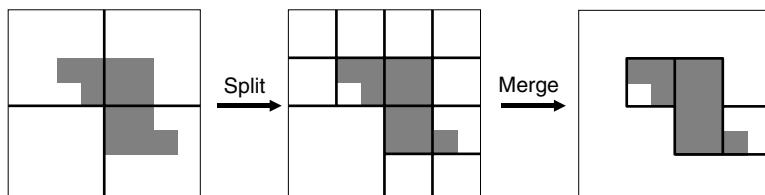


Fig. 7.9 Result after second iteration

The result of the first iteration is the input to the second iteration. In the second iteration, the image is split into 13 regions. The split operation is then followed by the merge operation in which the homogeneous regions are merged together.

Step 3

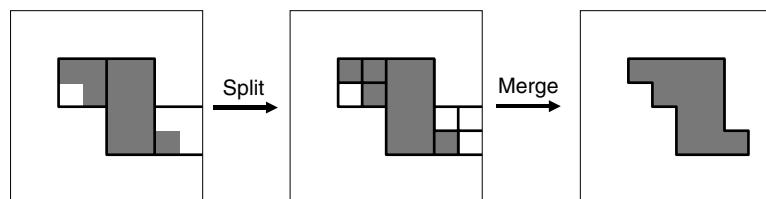


Fig. 7.10 Result after third iteration

The procedure followed in the step 2 is repeated in the step 3 to yield the desired result.

Example 7.4 Apply the split-and-merge technique for the given image shown in Fig. 7.11.

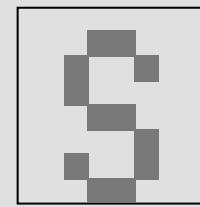


Fig. 7.11 Image to be segmented

Solution Split the given image into equal units and mark the entire given region as R . This is shown in Fig. 7.12.

Step 1 Each iteration has one split and one merge process. The first iteration splits the image into four regions R_1 , R_2 , R_3 and R_4 . This is followed by the merging process. As there is no homogenous region after splitting, the merging process will not make any impact. That is, the region after merging is same as the region before splitting.

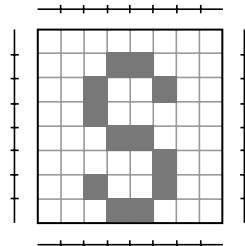


Fig. 7.12 Splitting the image into equal intervals

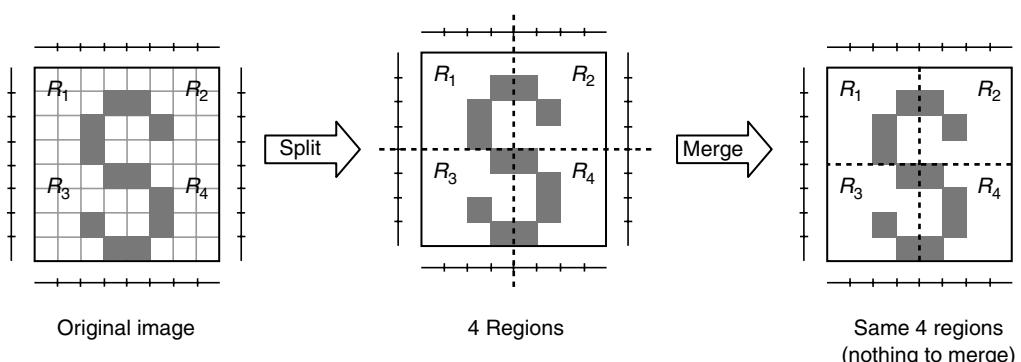


Fig. 7.13 Result after first iteration

Step 2 In the step 2, 4 regions are split into 16 regions and when they get merged, the region is reduced to 10.

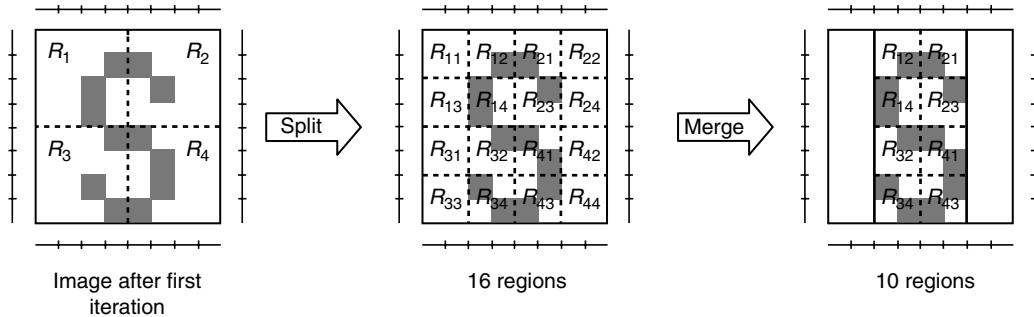


Fig. 7.14 Image after second iteration

Step 3 In the step 3, 8 regions are split into 34 regions and when they get merged, the region is reduced to 8.

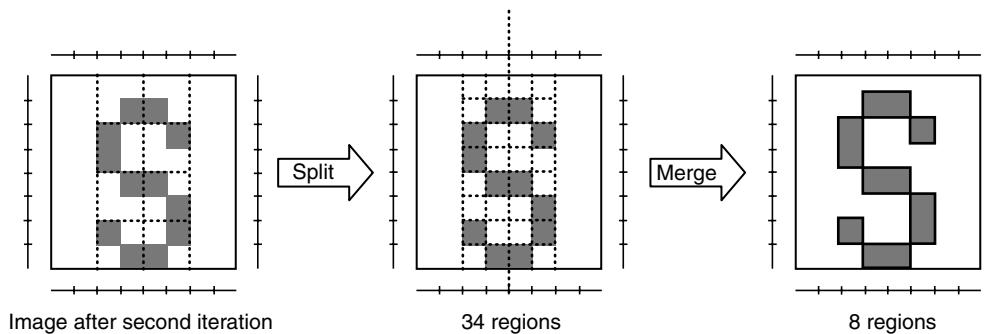


Fig. 7.15 Image after third iteration

The quadtree representation of the entire split-and-merge procedure is shown in Fig. 7.16.

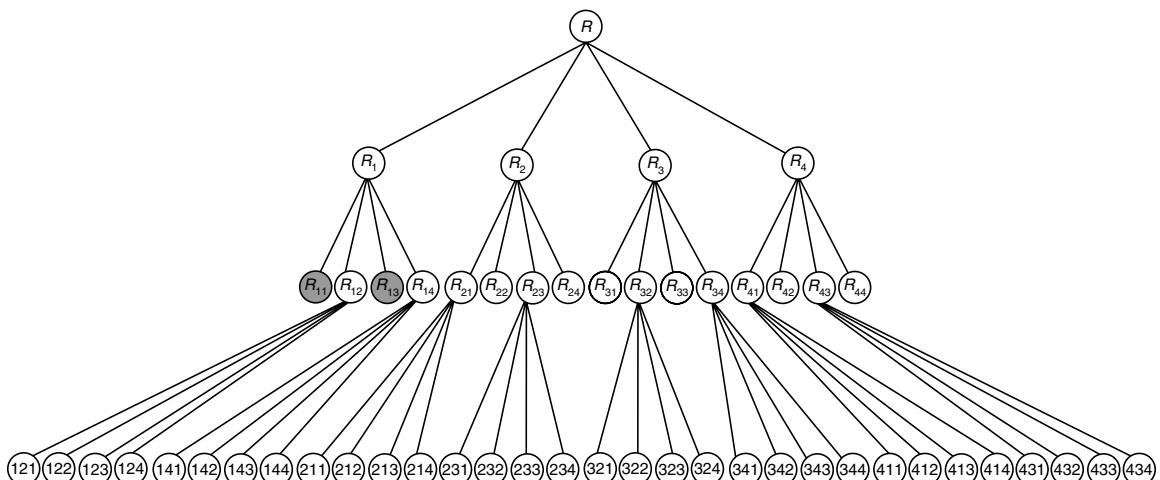


Fig. 7.16 Quadtree representation

7.4 CLUSTERING TECHNIQUES

The clustering technique attempts to access the relationships among patterns of the data set by organising the patterns into groups or clusters such that patterns within a cluster are more similar to each other than patterns belonging to different clusters. That is, clustering refers to the classification of objects into groups according to certain properties of these objects. In the clustering technique, an attempt is made to extract a feature vector from local areas in the image. A standard procedure for clustering is to assign each pixel to the class of the nearest cluster mean. Clustering methods can be divided into two categories—hierarchical and partitional. Within each category, there exist many types of algorithms for finding clusters.

7.4.1 Hierarchical Clustering

Hierarchical clustering techniques are based on the use of a proximity matrix indicating the similarity between every pair of data points to be clustered. The end result is a tree of clusters representing the nested group of patterns and similarity levels at which groupings change. The resulting clusters are always produced as the internal nodes of the tree, while the root node is reserved for the entire dataset and leaf nodes are for individual data samples. The clustering methods differ in regard to the rules by which two small clusters are merged or a large cluster is split.

The two main categories of algorithms used in the hierarchical clustering framework are agglomerative and divisive. *Agglomerative algorithms* seek to merge clusters to be larger and larger by starting with N single-point clusters. The algorithm can be divided into three classes: (i) single-link algorithm, (ii) complete-link algorithm, and (iii) minimum-variance algorithm. The single-link algorithm merges two clusters according to the minimum distance between the data samples from two clusters. Accordingly, the algorithm allows for a tendency to produce clusters with elongated shapes. In contrast, the complete-link algorithm incorporates the maximum distance between data samples in clusters, but its application always results in compact clusters. The quality of hierarchical clustering depends on how the dissimilarity measurement between two clusters is defined. The minimum-variance algorithm combines two clusters in the sense of minimising the cost function, namely, to form a new cluster with the minimum increase of the cost function. This algorithm has attracted considerable interest in vector quantisation, where it is termed *pairwise-nearest-neighbourhood algorithm*.

Divisive clustering begins with the entire dataset in the same cluster, followed by iterative splitting of the dataset until the single-point clusters are attained on leaf nodes. It follows a reverse clustering strategy against agglomerative clustering. On each node, the divisive algorithm conducts a full search for all possible pairs of clusters for data samples on the node.

Some of the hierarchical algorithms include COBWEB, CURE and CHAMELEON.

7.4.2 Partitional Clustering

Partition-based clustering uses an iterative optimisation procedure that aims at minimising an objective function f , which measures the goodness of clustering. Partition-based clusterings are composed of two learning steps—the partitioning of each pattern to its closest cluster and the computation of the cluster centroids. A common feature of partition-based clusterings is that the clustering procedure starts from an initial solution with a known number of clusters. The cluster centroids are usually computed based on the optimality criterion such that the objective function is minimised. Partitional algorithms are categorised into *partitioning relocation algorithms* and *density-based partitioning*. Algorithms of the first type are further categorised into *probabilistic clustering*, *K-medoids*, and *K-means*. The second types of partitional algorithms, called density-based partitioning, include algorithms such as DBSCAN, OPTICS DBCLAS, DENCLUE.

Partitional clustering techniques such as K-means clustering and ISODATA have an advantage over hierarchical clustering techniques, where a partition of the data points optimises some criterion functions.

7.4.3 K-means Clustering

The K-means method is the simplest method in unsupervised classification. The clustering algorithms do not require training data. K-means clustering is an iterative procedure. The K-means clustering algorithm clusters data by iteratively computing a mean intensity for each class and segmenting the image by classifying each pixel in the class with the closest mean. The steps involved in the K-means clustering algorithm is given below:

1. Choose K initial clusters $z_1(I), z_2(I), \dots, z_k(I)$.
2. At the k^{th} iterative step, distribute the samples x among the K clusters using the relation

$$x \in C_j(k) \quad \text{if } \|x - z_j(k)\| < \|x - z_i(k)\|$$

For $i = 1, 2, \dots, K$, $i \neq j$, where $C_j(k)$ denotes the set of samples whose cluster centre is $z_j(k)$.

3. Compute the new cluster centres $z_j(k+1)$, $j = 1, 2, \dots, K$, such that the sum of the squared distance from all points in $C_j(k)$ to the new cluster is minimised. The measure which minimises this is simply the sample mean of $C_j(k)$. Therefore, the new cluster centre is given by

$$z_j(k+1) = \frac{1}{N_j} \sum_{x \in C_j(k)} x, \quad j = 1, 2, \dots, K$$

where N_j is the number of samples in $C_j(k)$.

4. If $z_j(k+1)$, $j = 1, 2, \dots, K$, the algorithm has converged and the procedure is terminated. Otherwise go to Step 2.

The drawback the K-means algorithm is that the number of clusters is fixed, once K is chosen and it always returns K cluster centres.

7.4.4 Fuzzy Clustering

Clustering methods can be classified as either hard or fuzzy depending on whether a pattern data belongs exclusively to a single cluster or to several clusters with different degrees. In hard clustering, a membership value of zero or one is assigned to each pattern data, whereas in fuzzy clustering, a value between zero and one is assigned to each pattern by a membership function. In general, fuzzy clustering methods can be considered to be superior to those of their hard counterparts since they can represent the relationship between the input pattern data and clusters more naturally. Fuzzy clustering seeks to minimise a heuristic global cost function by exploiting the fact that each pattern has some graded membership in each cluster. The clustering criterion allows each pattern for multiple assignments of clusters. The fuzzy K-means algorithm iteratively updates the cluster centroid and estimates the class membership function by using the gradient descent approach.

Dissimilarity Function An essential quantity to measure the distinction between data patterns is the dissimilarity function. The dissimilarity functions must be carefully chosen due to the diversity and scale of features inhabited in patterns. Different choices of clustering dissimilarities lead to distinct clustering results. A common method to formulate the dissimilarity of two data vectors is the Minkowski metric which is given by

$$d_p(x_i, x_j) = \left(\sum_{s=1}^{\dim} \|x_{i,s} - x_{j,s}\|^p \right)^{1/p} \quad (7.1)$$

If $p = 1$, it implies the L_1 distance or Manhattan distance

If $p = 2$, it implies the Euclidean distance or L_2 distance

A metric should satisfy the following requirements of a distance function

$$\begin{aligned} d(x, x) &= 0, \quad \forall x \in X \\ 0 \leq d(x, y) &, \quad \forall x, y \in X \\ d(x, y) &= d(y, x) \end{aligned} \quad (7.2)$$

The definition of metric is also restricted by a more rigorous condition, the triangular inequality, which is given by

$$d(x, z) \leq d(x, y) + d(y, z) \quad \text{and} \quad d(x, y) = 0 \Leftrightarrow x = y$$

The case with $l = \infty$ in Minkowski metric comes to the maximum metric:

$$d(x_i, x_j) = \max_{1 \leq s \leq \dim} |x_{i,s} - x_{j,s}|$$

These measures are invariant to translation but variant to scaling. The scale invariant dissimilarities include the Canberra metric given by

$$d(x_i, x_j) = \sum_{s=1}^{\dim} \frac{|x_{i,s} - x_{j,s}|}{|x_{i,s}| + |x_{j,s}|} \quad (7.3)$$

and the cosine similarity is given by

$$d(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \times \|x_j\|}$$

Here the dot represents the inner product of two vectors and $\| \cdot \|$ is the L_2 norm in Euclidean space. Another way to generalise the Euclidean distance to scale invariant case is the Mahalanobis distance which is given by

$$d(x_i, x_j) = (x_i - x_j)^T \sum^{-1} (x_i - x_j)$$

where \sum is the covariance matrix with respect to the entire data set

$$\sum(X) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T \quad (7.4)$$

The Mahalanobis distance serves as a dissimilarity measurement in model-based clustering such as the EM clustering algorithm. The mahalanobis distance application always suffers from the singularity problem and a high computational cost when applied in the framework of K-means clustering.

7.5 IMAGE SEGMENTATION BASED ON THRESHOLDING

Thresholding techniques produce segments having pixels with similar intensities. Thresholding is a useful technique for establishing boundaries in images that contain solid objects resting on a contrasting background. There exist a large number of gray-level based segmentation methods using either global or local image information. The thresholding technique requires that an object has homogenous intensity and a background with a different intensity level. Such an image can be segmented into two regions by simple thresholding.

7.5.1 Global Thresholding

Global thresholding is the simplest and most widely used of all possible segmentation methods. In global thresholding, a threshold value of θ is chosen and the following condition is imposed:

$$f(m, n) = \begin{cases} 1 & \text{if } f(m, n) \geq \theta \\ 0 & \text{else} \end{cases} \quad (7.5)$$

Equation (7.5) is a complete description of a binarisation algorithm; it contains no indication on how to select the value of the threshold parameter θ . The value of θ has to be selected in an optimal manner. Global thresholding will suffer when pixels from different segments overlap in their use of intensities. If this is due to noise, a technique such as the minimum-error method can estimate the underlying cluster parameters and choose the thresholds to minimise the classification error. If the overlap is due to variation in illumination across the image, variable thresholding could be used. This can be visualised as a form of local segmentation.

7.5.2 Adaptive Thresholding

Global thresholding, or fixed thresholding, works well if the objects of interest have a reasonably uniform interior gray level and rest on a background of unequal but a relatively uniform gray level. In many cases, the background gray level is not constant, and object contrast varies within an image. In such cases, a threshold that works well in one area might not work well in other areas of the image. In these cases, it is convenient to use a threshold gray level that is a slowly varying function of position in the image.

7.5.3 Histogram-Based Threshold Selection

An image containing an object on a contrasting background has a bimodal gray-level histogram. The two peaks correspond to the relatively large number of points inside and outside the object. The dip between the peaks corresponds to the relatively few points around the edge of the object. This dip is commonly used to establish the threshold gray level.

The histogram is the derivative of the area function for an object whose boundary is defined by thresholding:

$$H(D) = -\frac{d}{dD} A(D) \quad (7.6)$$

where D represents the gray level, $A(D)$ represents the area of the object obtained by thresholding at gray level D , and $H(D)$ represents the histogram.

Increasing the threshold from D to $D + \Delta D$ cause only a slight decrease in area if D is near the dip of the histogram. Therefore, placing the threshold at the dip in the histogram minimises the sensitivity of the area measurement to small errors in threshold selection.

If the image containing the object is noisy and not large, the histogram itself will be noisy. Unless the dip is uncommonly sharp, the noise can make its location obscure, or at least unreliable from one image to the next. This effect can be overcome to some extent by smoothing the histogram using either a convolution filter or the curve-fitting procedure.

7.5.4 Limitation of Thresholding Technique

Thresholding is often used as an initial step in a sequence of image-processing operations. The main limitation of thresholding techniques is that in its simplest form, only two classes are generated and it cannot be applied to multi-channel images. A thresholding technique does not take into account the spatial characteristics of an image. This causes it to be sensitive to noise and intensity inhomogeneities.

7.6 EDGE-BASED SEGMENTATION

Edge-based segmentation exploits spatial information by detecting the edges in an image. Edges correspond to discontinuities in the homogeneity criterion for segments. Edge detection is usually done with local linear gradient operators such as Prewitt, Sobel and Laplacian filters. These operators work well for images with sharp edges and low amounts of noise. The detected boundaries using these operators may not necessarily form a set of closed connected curves, so some edge linking may be required.

7.7 CLASSIFICATION OF EDGES

Points in an image where brightness changes abruptly are called edges or edge points. Edges are significant local changes of intensity in an image. Edges are the boundaries between segments. Edges are a very important portion of the perceptual information content in an image. Edges can be broadly classified into (i) step edge, (ii) line edge, (iii) ramp edge, and (iv) roof edge.

7.7.1 Step Edge

The step edge defines a perfect transition from one segment to another. In the case of a step edge, the image intensity abruptly changes from one value to one side of the discontinuity to a different value on the opposite side. If segments are piecewise constant and pixels can only belong to one segment, then a step edge model is implicitly used. The step edge is illustrated in Fig. 7.17.

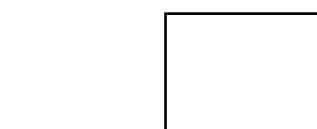


Fig. 7.17 Step edge

7.7.2 Line Edge

If a segment of an image is very narrow, it necessarily has two edges in close proximity. This arrangement is called a line. The line edge is shown in Fig. 7.18.

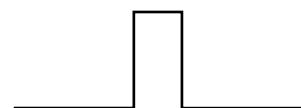


Fig. 7.18 Line edge

7.7.3 Ramp Edge

A ramp allows for a smoother transition between segments. A ramp edge is useful for modeling the blurred edges created from sampling a scene containing objects not aligned to the pixel grid. The ramp edge is shown in Fig. 7.19.

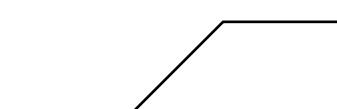


Fig. 7.19 Ramp edge

7.7.4 Roof Edge

Two nearby ramp edges result in a line structure called a roof. Basically, there are two types of roof edges: (i) convex roof edges, and (ii) concave roof edges which are shown in Fig. 7.20.

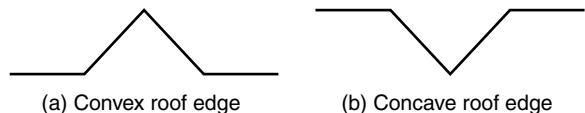


Fig. 7.20 Roof edge

Causes of Edges Edges can be created by shadows, texture, geometry, and so forth. Edges are basically discontinuities in the image intensity due to changes in the image structure. These discontinuities originate from different features in an image. Edge points are to be associated with the boundaries of objects and other kinds of changes. Edges within an image generally occur at various resolutions or scales and represent transitions of different degrees or gradient levels.

7.8 EDGE DETECTION

Edge detection is the process of finding meaningful transitions in an image. Edge detection is one of the central tasks of the lower levels of image processing. The points where sharp changes in the brightness occur typically form the border between different objects. These points can be detected by computing intensity differences in local image regions. That is, the edge-detection algorithm should look for a neighbourhood with strong signs of change. Most of the edge detectors work on measuring the intensity gradient at a point in the image.

Importance of Edge Detection Edge detection is a problem of fundamental importance in image analysis. The purpose of edge detection is to identify areas of an image where a large change in intensity occurs. These changes are often associated with some physical boundary in the scene from which the image is derived. In typical images, edges characterise object boundaries and are useful for segmentation, registration and identification of objects in a scene.

7.8.1 Gradient Operator

A gradient is a two-dimensional vector that points to the direction in which the image intensity grows fastest. The gradient operator ∇ is given by

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \quad (7.7)$$

If the operator ∇ is applied to the function f then

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (7.8)$$

The two functions that can be expressed in terms of the directional derivatives are the gradient magnitude and the gradient orientation. It is possible to compute the magnitude $\|\nabla f\|$ of the gradient and the orientation $\phi(\nabla f)$.

The gradient magnitude gives the amount of the difference between pixels in the neighbourhood which gives the strength of the edge. The gradient magnitude is defined by

$$|\nabla f| = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \left[G_x^2 + G_y^2 \right]^{\frac{1}{2}} \quad (7.9)$$

The magnitude of the gradient gives the maximum rate of increase of $f(x, y)$ per unit distance in the gradient orientation of $|\nabla f|$.

The gradient orientation gives the direction of the greatest change, which presumably is the direction across the edge. The gradient orientation is given by

$$\phi(\nabla f) = \tan^{-1} \left(\frac{G_y}{G_x} \right) \quad (7.10)$$

where the angle is measured with respect to the x -axis.

The direction of the edge at (x, y) is perpendicular to the direction of the gradient vector at that point. Figure 7.21 indicates the gradient of the edge pixel. The circle indicates the location of the pixel. An edge pixel is described using two important features:

- (i) *Edge strength*, which is equal to the magnitude of the gradient
- (ii) *Edge direction*, which is equal to the angle of the gradient

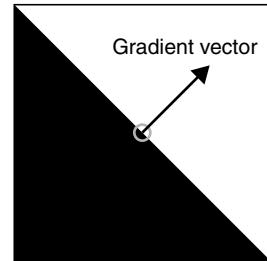


Fig. 7.21 Gradient of edge pixel

7.8.2 Edge Detection Using First-order Derivatives

The derivative of a digital pixel grid can be defined in terms of differences. The first derivative of an image containing gray-value pixels must fulfill the following conditions—it must be zero in flat segments, i.e. in area of constant gray-level values; it must be non-zero at the beginning of a gray-level step or ramp; and it must be non-zero along the ramp (constant change in gray values). The first-order derivative of a one-dimensional function $f(x)$ can be obtained using

$$\frac{df}{dx} = f(x+1) - f(x) \quad (7.11)$$

An image is a function of two variables $f(x, y)$. Equation (7.11) only refers to the partial derivative along the x -axis. Pixel discontinuity can be determined along eight possible directions such as up, down, left, right and along the four diagonals.

The other method of calculating the first-order derivative is given by estimating the finite difference:

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h, y) - f(x, y)}{h} \quad (7.12)$$

$$\frac{\partial f}{\partial y} = \lim_{h \rightarrow 0} \frac{f(x, y+h) - f(x, y)}{h} \quad (7.13)$$

The finite difference can be approximated as

$$\frac{\partial f}{\partial x} = \frac{f(x+h, y) - f(x, y)}{h_x} = f(x+1, y) - f(x, y), (h_x = 1) \quad (7.14)$$

and

$$\frac{\partial f}{\partial y} = \frac{f(x, y+h) - f(x, y)}{h_y} = f(x, y+1) - f(x, y), (h_y = 1) \quad (7.15)$$

Using the pixel coordinate notation and considering that j corresponds to the direction of x , and i corresponds to the y direction, we have

$$\frac{\partial f}{\partial x} = f(i, y+1) - f(i, y) \quad (7.16)$$

and

$$\frac{\partial f}{\partial y} = f(i, y) - f(i, y+1) \quad (7.17)$$

Most edge-detecting operators can be thought of as gradient calculators. Because the gradient is a continuous function concept and the input signal is a finite signal (image), the gradient computations have to be approximated. Since derivatives are linear and shift-invariant, gradient calculation is most often done using convolution. Numerous kernels have been proposed for finding edges.

7.8.3 Roberts Kernel

The main objective is to determine the differences between adjacent pixels, one way to find an edge is to explicitly use $\{+1, -1\}$ that calculates the difference between adjacent pixels. Mathematically, these are called forward differences. The Roberts kernels are, in practice, too small to reliably find edges in the presence of noise. The simplest way to implement the first-order partial derivative is by using the Roberts cross-gradient operator.

$$\frac{\partial f}{\partial x} = f(i, j) - f(i+1, j+1) \quad (7.18)$$

and

$$\frac{\partial f}{\partial y} = f(i+1, j) - f(i, j+1) \quad (7.19)$$

The partial derivatives given above can be implemented by approximating them to two 2×2 masks. The Roberts operator masks are given by

$$G_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad (7.20)$$

These filters have the shortest support, thus the position of the edges is more accurate, but the problem with the short support of the filters is its vulnerability to noise.

7.8.4 Prewitt Kernel

The Prewitt kernels are named after Judy Prewitt. Prewitt kernels are based on the idea of central difference. The Prewitt edge detector is a much better operator than the Roberts operator. Consider the arrangement of pixels about the central pixel $[i, j]$ as shown below:

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ a_7 & [i, j] & a_3 \\ a_6 & a_5 & a_4 \end{bmatrix} \quad (7.21)$$

The partial derivatives of the Prewitt operator are calculated as

$$G_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \quad (7.22)$$

and

$$G_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2) \quad (7.23)$$

The constant c in the above expressions implies the emphasis given to pixels closer to the centre of the mask. G_x and G_y are the approximations at $[i, j]$.

Setting $c = 1$, the Prewitt operator mask is obtained as

$$G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad (7.24)$$

The Prewitt masks have longer support. The Prewitt mask differentiates in one direction and averages in other direction; so the edge detector is less vulnerable to noise.

7.8.5 Sobel Kernel

The Sobel kernels are named after Irwin Sobel. The Sobel kernel relies on central differences, but gives greater weight to the central pixels when averaging. The Sobel kernels can be thought of as 3×3 approximations to first derivatives of Gaussian kernels. The partial derivatives of the Sobel operator are calculated as

$$G_x = (a_2 + 2a_3 + a_4) - (a_0 + 2a_7 + a_6) \quad (7.25)$$

and

$$G_y = (a_6 + 2a_5 + a_4) - (a_0 + 2a_1 + a_2) \quad (7.26)$$

The Sobel masks in matrix form are given as

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (7.27)$$

The noise-suppression characteristics of a Sobel mask is better than that of a Prewitt mask.

7.8.6 Frei–Chen Edge Detector

Edge detection using Frei–Chen masks is implemented by mapping the intensity vector using a linear transformation and then detecting edges based on the angle between the intensity vector and its projection onto the edge subspace. Frei–Chen edge detection is realised with the normalised weights. Frei–Chen masks are unique masks, which contain all the basis vectors. This implies that 3×3 image area is represented with the weighted sum of nine Frei–Chen masks. Primarily, the image is convolved with each of the nine masks. Then an innerproduct of the convolution results of each mask is performed. The nine Frei–Chen masks are given below:

$$G_1 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}; \quad G_2 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}; \quad G_3 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}$$

$$G_4 = \frac{1}{2\sqrt{2}} \begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix}; \quad G_5 = \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}; \quad G_6 = \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_7 = \frac{1}{6} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}; \quad G_8 = \frac{1}{6} \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}; \quad G_9 = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The first four Frei–Chen masks are used for edges and the next four are used for lines and the last mask is used to compute averages. For edge detection, appropriate masks are chosen and the image is projected onto it.

7.8.7 Second-Derivative Method of Detecting Edges in an Image

Finding the ideal edge is equivalent to finding the point where the derivative is maximum or minimum. The maximum or minimum value of a function can be computed by differentiating the given function and finding places where the derivative is zero. Differentiating the first derivative gives the second derivative. Finding the optimal edges is equivalent to finding places where the second derivative is zero. The differential operators can be applied to images; the zeros rarely fall exactly on a pixel. Typically, they fall between pixels. The zeros can be isolated by finding the zero crossings. Zero crossing is the place where one pixel is positive and a neighbouring pixel is negative. The problems with zero-crossing methods are the following:

- (i) Zero crossing methods produce two-pixel thick edges.
- (ii) Zero crossing methods are extremely sensitive to noise.

For images, there is a single measure, similar to the gradient magnitude, that measures the second derivative, which is obtained by taking the dot product of ∇ with itself.

$$\nabla \cdot \nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \quad (7.28)$$

$$\nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad (7.29)$$

The operator $\nabla \cdot \nabla = \nabla^2$ is called the Laplacian operator.

The Laplacian operator when applied to the function f , we get

$$\nabla^2 f = \left(\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \right) f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (7.30)$$

The Laplacian operation can be expressed in terms of difference equations as given below:

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y) \quad (7.31)$$

and

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) - 2f(x, y) + f(x-1, y) \quad (7.32)$$

Also

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1) \quad (7.33)$$

This implies that

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y) \quad (7.34)$$

The 3×3 Laplacian operator is given by

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

The Laplacian operator subtracts the brightness values of each of the neighbouring pixels from the central pixel. When a discontinuity is present within the neighbourhood in the form of a point, line or edge, the result of the Laplacian is a non-zero value. It may be either positive or negative depending where the central point lies with respect to the edge.

The Laplacian operator is rotationally invariant. The Laplacian operator does not depend on direction as long as they are orthogonal.

Drawbacks of the Laplacian Operator The main drawbacks of the Laplacian operator are summarised below:

1. Useful directional information is not available by the use of a Laplacian operator.
2. The Laplacian, being an approximation to the second derivative, doubly enhances any noise in the image.

7.8.8 Laplacian of Gaussian (LOG)

A prominent source of performance degradation in the Laplacian operator is noise in the input image. The noise effects can be minimised by smoothing the image prior to edge enhancement. The Laplacian-of-Gaussian (LOG) operator smooths the image through convolution with a Gaussian-shaped kernel followed by applying the Laplacian operator. The sequence of operation involved in an LOG operator is given below:

Step 1 *Smoothing of the input image $f(m, n)$*

The input image $f(m, n)$ is smoothed by convolving it with the Gaussian mask $h(m, n)$ to get the resultant smooth image $g(m, n)$.

$$g(m, n) = f(m, n) \otimes h(m, n) \quad (7.35)$$

Step 2 The Laplacian operator is applied to the result obtained in Step 1. This is represented by

$$g'(m, n) = \nabla^2(g(m, n)) \quad (7.36)$$

Substituting Eq. (7.35) in Eq. (7.36), we get

$$g'(m, n) = \nabla^2(g(m, n)) = \nabla^2(f(m, n) \otimes h(m, n)) \quad (7.37)$$

Here, $f(m, n)$ represents the input image and $h(m, n)$ represents the Gaussian mask. The Gaussian mask is given by

$$h(m, n) = e^{\frac{-r^2}{2\sigma^2}} \quad (7.38)$$

Here, $r^2 = m^2 + n^2$ and σ is the width of the Gaussian.

We know that convolution is a linear operator, and hence Eq. (7.37) can be rewritten as

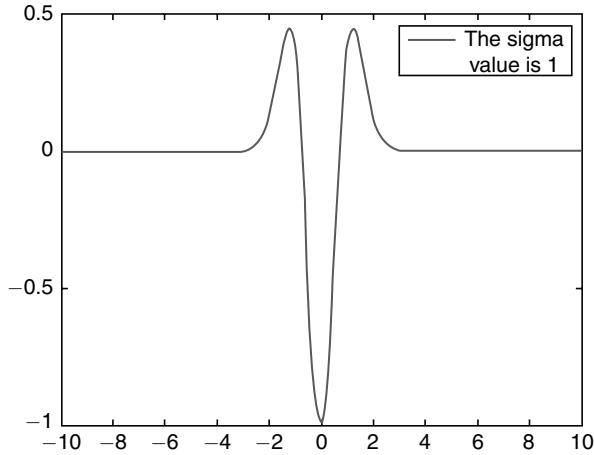
$$g'(m, n) = [\nabla^2(h(m, n))] \otimes f(m, n) \quad (7.38)$$

On differentiating the Gaussian kernel, we get

$$\nabla^2(h(m, n)) = \frac{1}{\sigma^2} \left[\frac{r^2}{\sigma^2} - 1 \right] \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad (7.39)$$

The shape of $\nabla^2\{h(m, n)\}$ for $\sigma = 1$ is given in Fig. 7.22 and the corresponding MATLAB code is given in Fig. 7.23.

Disadvantages of LOG Operator The LOG operator being a second-derivative operator, the influence of noise is considerable. It always generates closed contours, which is not realistic.

**Fig. 7.22** Differentiation of Gaussian function

```

clc
clear all
close all
sigma=input('Enter the value of sigma:');
i=-10:1:10;
j=-10:1:10;
r=sqrt(i.*i+j.*j);
y=(1/(sigma^2))*(((r.*r)/sigma^2)-1).*exp(-r.*r/2*sigma^2);
plot(i, y), legend(sprintf('The sigma value is %g, sigma'))
```

Fig. 7.23 MATLAB code for differentiation of Gaussian function

7.8.9 Difference of Gaussians Filter (DoG)

The DoG filter is obtained by taking the difference of two Gaussian functions. The expression of a DoG filter is given by

$$h(m, n) = h_1(m, n) - h_2(m, n)$$

where $h_1(m, n)$ and $h_2(m, n)$ are two Gaussian functions which are given by

$$h_1(m, n) = e^{\frac{r^2}{2\sigma_1^2}} \quad \text{and} \quad h_2(m, n) = e^{\frac{r^2}{2\sigma_2^2}}$$

Hence

$$h(m, n) = e^{\frac{r^2}{2\sigma_1^2}} - e^{\frac{r^2}{2\sigma_2^2}} \quad (7.40)$$

The shapes of the DoG filter for $\sigma_1 = 4$ and $\sigma_2 = 1$ is illustrated in Fig. 7.24 and the corresponding MATLAB code is given in Fig. 7.25.

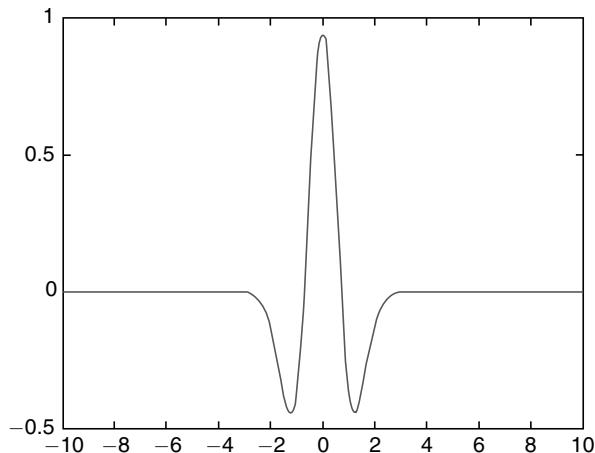


Fig. 7.24 Shape of DoG filter

```

clc
clear all
close all
sigma=input('Enter the value of sigma:');
sigma1=input('Enter the value of sigma1:');
i=-10:.1:10;
j=-10:.1:10;
r=sqrt(i.*i+j.*j);
y1=(1/(sigma^2))*(((r.*r)/sigma^2)-1).*exp(-r.*r/2*sigma^2);
y2=(1/(sigma1^2))*(((r.*r)/sigma1^2)-1).*exp(-r.*r/2*sigma1^2);
y=y1-y2;
plot(i, y)

```

Fig. 7.25 MATLAB code to plot DoG filter function

From Fig. 7.24, it is clear that the DoG filter function resembles a Mexican-hat wavelet. Therefore, a Mexican-hat wavelet is obtained by taking the difference of two Gaussian functions.

7.8.10 Canny Edge Detector

One problem with a Laplacian zero-crossing as an edge detector is that it simply adds the principal curvatures together. That is, it does not really determine the maximum of the gradient magnitude. The Canny edge detector defines edges as zero-crossings of second derivatives in the direction of the greatest first derivative. The Canny operator works in a multi-stage process. First, the image is smoothed by a Gaussian convolution. Then, a 2D first derivative operator is applied to the smoothed image to highlight regions of the image with high spatial derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges

and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as non-maximal suppression. The tracking process exhibits hysteresis controlled by two thresholds T_1 and T_2 , with $T_1 > T_2$. Tracking can only begin at a point on a ridge higher than T_1 . Tracking then continues in both directions out from that point until the height of the ridge falls below T_2 . This hysteresis helps to ensure that noisy edges are not broken into multiple edge fragments. The effectiveness of a Canny edge detector is determined by three parameters —(i) width of the Gaussian kernel, (ii) upper threshold, and (iii) lower threshold used by the tracker.

Increasing the width of the Gaussian kernel reduces the detector's sensitivity to noise, at the expense of losing some of the finer details in the image. The localisation error in the detected edges also increases slightly as the Gaussian width is increased. The Gaussian smoothing in the Canny edge detector fulfills two purposes. First, it can be used to control the amount of detail that appears in the edge image, and second, it can be used to suppress noise.

The upper tracking threshold is usually set quite high and the lower threshold value is set quite low for good results. Setting the lower threshold too high will cause noisy edges to break up. Setting the upper threshold too low increases the number of spurious and undesirable edge fragments appearing in the output.

MATLAB Example 1: Edge Detection *Read an input image and compute the edges in the image using different edge detectors like Robert, Prewitt, Sobel, Log and Canny. Comment on the result obtained.*

Solution The MATLAB code that computes the edges in the image is given in Fig. 7.26 and the corresponding results are given in Fig. 7.27.

From Fig. 7.27, it is obvious that the Canny edge detector is more effective than other edge detectors in detecting the edges in the image.

```
% This program computes the edges in the image
clear all;
close all;
clc;
a=imread(deer1.jpg);
a=rgb2gray(a);
b=edge(a, roberts);
c=edge(a, sobel);
d=edge(a, prewitt);
e=edge(a, log);
f=edge(a, canny);
imshow(a), title(Original Image)
Fig., imshow(b), title(Roberts)
Fig., imshow(c), title(Sobel)
Fig., imshow(d), title(Prewitt)
Fig., imshow(e), title(Log)
Fig., imshow(f), title(Canny)
```

Fig. 7.26 MATLAB code to compute the edges in the image

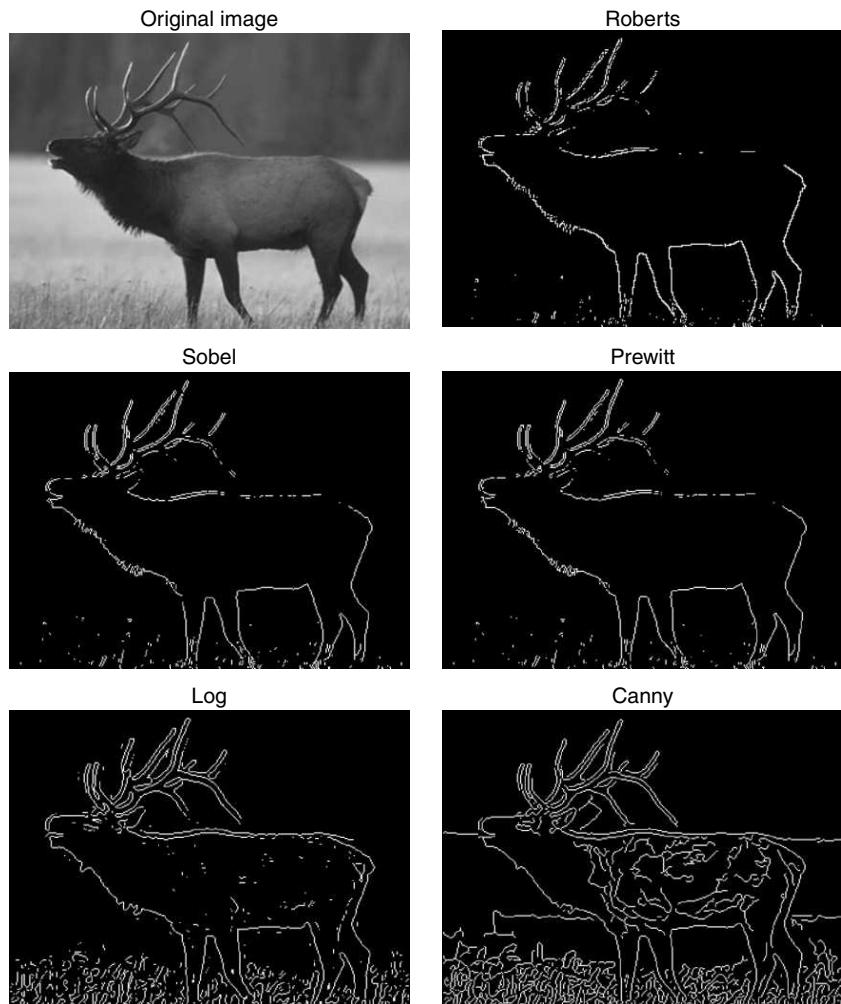


Fig. 7.27 Result of different edge detectors

7.9 EDGE LINKING

If the edges are relatively strong and the noise level is low, one can threshold an edge-magnitude image and thin the resulting binary image down to single pixel-wide closed, connected boundaries. Under less-than-ideal conditions, however, such an edge image will have gaps that must be filled. Small gaps can be filled by searching a 5×5 or larger neighbourhood, centred on an end point, for other end points, and then filling in boundary pixels as required to connect them. In images with many edge points, however, this can oversegment the image.

7.9.1 Edge Linking Through Heuristic Search

If there is a gap in a boundary in an edge image, but it is too long to fill accurately with a straight line, it may not really be a gap in the same boundary. In such cases, a quality measure that is a function, can be

computed for every connected path between the two end points, say P and Q . The edge quality function could include the average of the edge strengths of the points. Heuristic search algorithms become computationally expensive if the edge quality function is complex and the gaps to be evaluated are many and long. It performs well in relatively simple images, but it does not necessarily converge upon the globally optimal path between end points.

Heuristic Search Algorithm If there is a gap between two edge points P and Q then the heuristic search algorithm to link the two points P and Q is given below:

A. Evaluate Neighbours of P as Candidates for the First Step Towards Q .

1. Identify three neighbours of P that lies in the general direction of Q .
2. Compute the edge quality function from P for each point from Step 1.
3. Select the one that maximises the edge quality function from P to those points.
4. Use the point from Step 2 as the starting point for the next iteration.
5. Repeat this process until point Q is reached.

B. Qualify the Path.

1. Compare the edge quality function over the newly created path to a threshold.
2. If the newly created edge is not sufficiently strong, discard it.
3. If the newly created edge is sufficiently strong, take the resulting graph as the boundary.

Curve-fitting Approach to Edge Linking If the edge points are generally sparse it might be desirable to fit a piecewise linear or higher-order spline through them to establish a suitable boundary. A simple example is iterative end-point fitting.

Iterative End-point Fitting Algorithm Suppose there is a group of edge points lying scattered between two particular edge points P and Q . If one wishes to select a subset of these to form the nodes of a piecewise linear path from P to Q then the iterative end-point algorithm used to link the two points is given below:

A. Link the Edge Points.

1. Begin by establishing straight line from P to Q .
2. Compute the perpendicular distance from that line to each of the remaining edge points.
3. Identify the furthermost one, which becomes the next node on the path. The path now has two branches.
4. Repeat this process on each new branch of the path until no remaining edge points lie more than the same fixed distance away from the nearest branch.

B. Find the Boundary.

1. Repeat P for each pair of points P, Q all around the object.
2. Take the resulting graph as a polygonal approximation to the boundary.

7.10 HOUGH TRANSFORM

The straight line $y = mx + b$ can be expressed in polar coordinate as

$$\rho = x \cos(\theta) + y \sin(\theta) \quad (7.41)$$

where ρ, θ defines a vector from the origin to the nearest point on the straight line $y = mx + b$. This vector will be perpendicular from the origin to the nearest point to the line as shown in Fig. 7.28.

Any line in the x, y plane corresponds to a point in the 2D space defined by the parameter ρ and θ . Thus the Hough transform of a straight line in the x, y space is a single point in the ρ, θ space. Every straight line that passes through a particular point x_i, y_i in the x, y plane plots to a point in the ρ, θ space and these points should satisfy Eq. (7.41) with x_1, y_1 as constants. Thus the locus of all such lines in the x, y space is a sinusoid in parameter space and any point in the x, y plane corresponds to a particular sinusoidal curve in the ρ, θ space.

Suppose we have a set of edge points x_i, y_i that lie along a straight line having parameters ρ_0, θ_0 . Each edge point plots to a sinusoidal curve in the ρ, θ space, but these curves must intersect at a point ρ_0, θ_0 since this is a line they all have common.

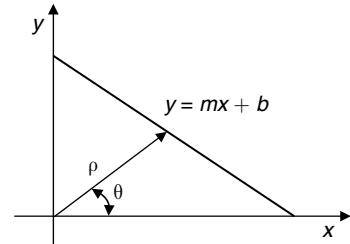


Fig. 7.28 Straight line

7.10.1 Piecewise Linear Boundary Fitting with the Hough Transform

A. Accumulate a 2D Histogram in the ρ, θ Space.

1. For each edge point x_i, y_i increment all the histogram bins in the ρ, θ space that correspond to the Hough transform for a point at that location.
2. Repeat Step 1 for all edge points.

B. Find Line Segments.

1. Search ρ versus θ histograms for local maxima.
2. Generate lines corresponding to the ρ, θ coordinates of the local maxima.
3. Take the polygon formed by the resulting linear boundary segments as the boundary.

7.11 ACTIVE CONTOUR

A promising image-analysis tool for refinement of an object boundary is the active contour or snake. The curve formed by the connected edge points delineates the active contour. It is a set of connected points which interactively move so as to minimise a specified energy function. Local properties of the image such as gray level, gradient, etc., contribute to the energy of the snake. The snake reacts to the image, but moves in a smooth continuous manner towards the desired object boundary.

Greedy-Snake Algorithm The steps involved in a greedy-snake algorithm are given below.

A. Prepare the Image.

1. Define an energy function which could naturally take on low values for points located on the actual boundary.
2. Generate an initial boundary by a segmentation technique such as thresholding.

B. Wiggle the Snake.

1. Compare the boundary of each point on the boundary with the energies calculated for points in its neighbourhood.

2. For each current boundary point, move the boundary to the neighbouring point that has the lowest energy.
3. Perform this operation once on all points on the boundary to complete one iteration of the snake algorithm.
4. Repeat this iteration until it causes no further movement of the boundary points.
5. When this happens, the snake is said to have 'converged' and the segmentation is complete.

The choice of the energy function to be minimised determines the behaviour of the algorithm. Given the parametric representation of the snake $v(s) = (x(s), y(s))$ where $s \in [0, 1]$, the energy function is given by

$$E_{\text{snake}} = \int_0^1 E_{\text{snake}}(v(s))ds \quad (7.42)$$

$$E_{\text{snake}} = \int_0^1 [E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{ext}}(v(s))]ds \quad (7.43)$$

Thus, the energy of the snake corresponds to three different forces:

- (i) Internal forces between points of the contour
- (ii) Image forces such as gray level and gradient magnitude that pull the snake towards the optimal boundary position
- (iii) External constraints such as user forces applied to the contour

The internal energy of the snake can be modeled using two terms:

$$E_{\text{int}} = E_{\text{cont}} + E_{\text{curve}} \quad (7.44)$$

where E_{cont} corresponds to the continuity term and E_{curve} corresponds to the curvature term. The continuity term can be made proportional to the difference between the curve interpoint distance and the average interpoint distance of the entire snake. This formulation encourages equal spacing of contour points around the boundary. The energy associated with the curvature at a point can be approximated by taking the square of the magnitude of the difference between two adjacent unit tangent vectors. This gives a reasonable estimate of the local curvature. The continuity measurement of the image energy term can be simply the result of an edge operator such as a Sobel gradient operator.

The main advantages of the greedy-snake algorithm are computational efficiency and relative simplicity. Its main drawback is the extreme local nature of its decision criteria.

Problem with Snake Formation The two main drawbacks of snake formation are the following:

- (i) Poor convergence property of the active contours—specifically, concavities in the object of interest are rarely covered.
- (ii) The second problem with snakes is the limited capture range, which is related to the initialisation of the snake around the object of interest.

7.12 WATERSHED TRANSFORMATION

The watershed transformation is a powerful tool for image segmentation. In a watershed transformation, the image is considered as a topographic surface. The gray-level of the image represents the altitudes. The region

with a constant gray level constitutes the flat zone of an image. Also, region edges correspond to high watersheds and low-gradient region interiors correspond to catchment basins. Catchment basins of the topographic surface are homogeneous in the sense that all pixels belonging to the same catchment basin are connected with the basin region of minimum altitude by a simple path of pixels that have a monotonically decreasing altitude along the path. Such catchment basins represent the regions of the segmented image. By viewing the intensity in an image as elevation and simulating rainfall, it is possible to decompose an image into watershed regions.

Watersheds are defined as the lines separating catchment basins, which belong to different minima. A watershed can be imaged as a high mountain that separates two regions. Each region has its own minimum, and if a drop of water falls on one side of the watershed, it will reach the minimum of the regions. The regions that the watershed separates are called catchment basins. The catchment basin is illustrated in Fig. 7.29.

The basic watershed algorithm works well if each local minima corresponds to an object of interest. In such a case, the watershed lines represent the boundaries of the objects. If there are many more minima in the image than the objects of interest, the image will be over-segmented. To avoid over-segmentation, the watershed algorithm is constrained by an appropriate marking function.

Drawbacks of Watershed-algorithm-based Image Segmentation The drawbacks of watershed-algorithm-based image segmentation are (i) over-segmentation, (ii) sensitivity to noise, (iii) poor performance in the area of low-contrast boundaries, and (iv) poor detection of thin structures.

(i) Over-segmentation When the watershed transform infers catchment basins from the gradient of the image, the result of the watershed transform contains a myriad of small regions. This over-segmentation will not give any fruitful result. One of the solutions to the problem of over-segmentation is to use a marker-based watershed transformation.

In a marker-controlled watershed algorithm, two markers are defined—(a) foreground marker, and (b) background marker. A marker is a region involved in a watershed transform. A marker can be a single point or a set of connected or disconnected points. Each marker placed in the image will grow to generate a catchment basin in the final segmentation. The input to the watershed transformation is usually the gradient image. This gradient image is modified to keep only the most significant contours in the areas of interest between the markers. The contour search is then performed on the modified gradient image.

(ii) Sensitivity to Noise The main drawback of watershed-algorithm-based image segmentation is the presence of noise in the image. The noise in the image results in over-segmentation. The solution to this problem is to employ anisotropic filters.

(iii) Low-contrast Boundaries If the signal-to-noise ratio is not high enough at the contour of interest, the watershed transform will be unable to detect it accurately. Moreover, the watershed transform detects the contours with higher value between markers, which are not always the contours of interest.

(iv) Poor Detection of Thin Edges When the watershed transform is applied on the gradient image, the smoothing associated with the gradient estimation, together with usual approach of storing gradient values only at image pixel positions rather than with sub-pixel accuracy, make it difficult to detect thin-catchment basin areas.

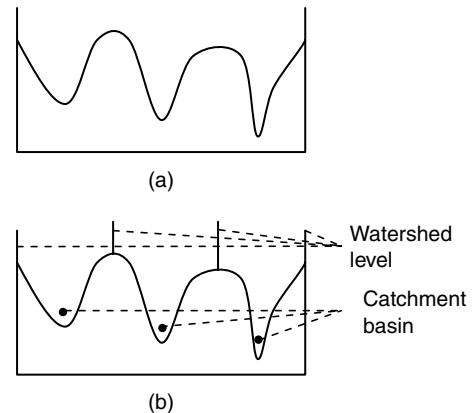


Fig. 7.29 (a) Gray-level profile of image data
(b) Watershed segmentation

MATLAB Example 2: Watershed transformation *Read an image, and perform the watershed transformation. Now corrupt the image by adding salt-and-pepper noise and perform the watershed transformation for this noisy image. Comment on the results obtained.*

Solution The input image chosen is a checkerboard pattern. The MATLAB command to create the checkerboard pattern is `checkerboard`. After creating the checkerboard pattern, it is corrupted by applying salt-and-pepper noise. The MATLAB code which performs watershed transformation of the checkerboard pattern and the noisy checkerboard pattern is shown in Fig. 7.30. The corresponding results are shown in Fig. 7.31.

From the results shown in Fig. 7.31, it is obvious that the presence of noise in the image results in over-segmentation.

```
a=checkerboard(32);
a1=imnoise(a, salt & pepper, 0.1);
b=watershed(a, 4);
b1=watershed(a1, 4);
imshow(a), title(Original Image)
Fig., imshow(a1), title(Noisy Image)
Fig., imshow(b), title(Watershed of Original)
Fig., imshow(b1), title(Watershed of Noisy Image)
```

Fig. 7.30 MATLAB code to perform watershed transformation

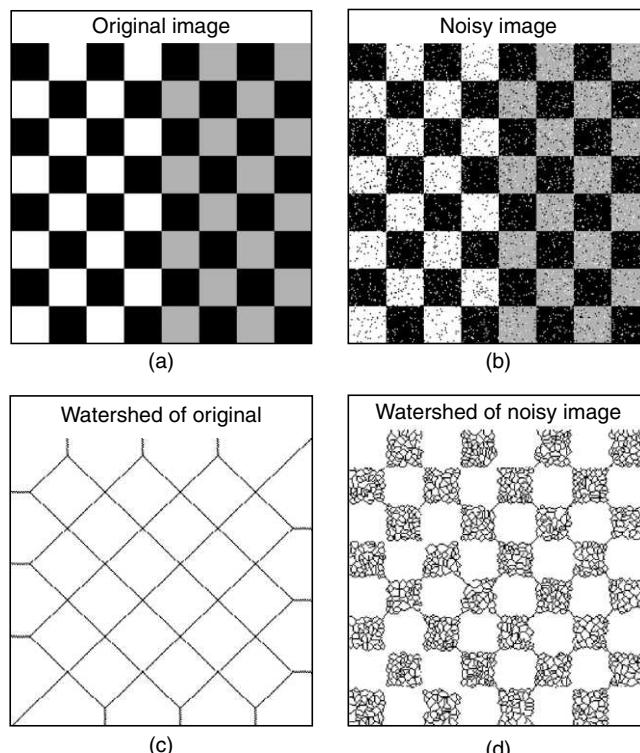


Fig. 7.31 Results of watershed transformation

7.13 SHAPE REPRESENTATION

Shape representation looks for effective ways to capture the essence of shape features that make it easier for a shape to be stored, transmitted, compared against, and recognised. By efficient representation, the object information is made accessible for computer-interpretation. These features must be independent of translation, rotation, and scaling of the shape. Shape representation and description is one of the major problems in content-based image retrieval. Content-based image retrieval is a technique for retrieving images on the basis of automatically derived features such as colour, texture and shape. Shape representation is crucial in image understanding, both for object recognition and signal representation.

7.14 CLASSIFICATION OF SHAPE-REPRESENTATION TECHNIQUES

Shape representation techniques can be broadly classified into two categories—(i) boundary or contour-based shape-representation technique, and (ii) region-based shape-representation technique. In the boundary-based technique, objects are represented in terms of their external characteristics, while in region-based technique, shape features are extracted from the whole shape region.

The taxonomy of shape-representation techniques is given in Fig. 7.32.

7.14.1 Chain Code

The method of chain code was introduced in 1961 by Freeman. In this approach, an arbitrary curve is represented by a sequence of small vectors of unit length and a limited set of possible directions. Chain codes describe an object by a sequence of unit-size line segments with a given orientation.

The chain-code representation is constructed as follows:

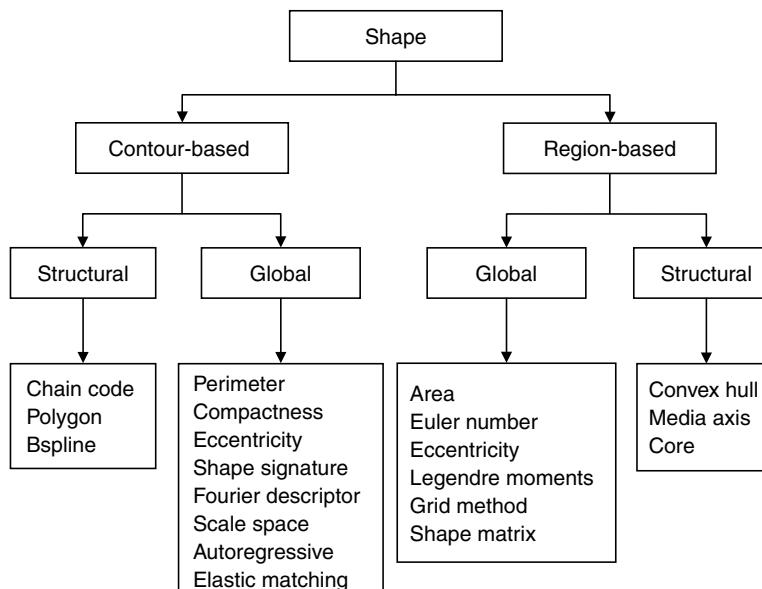


Fig. 7.32 Taxonomy of shape-representation technique

Step 1 Select a starting point of the contour. Represent this point by its absolute coordinates in the image.

Step 2 Represent every consecutive point by a chain code showing the transition needed to go from the current point to the next point on the contour.

Step 3 Stop if the next point is the initial point, or the end of the contour. Store the lengths of the contours into the file.

A variation of chain-code representation is differential chain codes. Each differential chain code K_i is represented by the difference of the current chain code C_i and the preceding chain code C_{i-1} : $K_i = C_i - C_{i-1}$.

Types of Chain Codes Basically, there are two types of chain codes—(i) 4-directional chain codes, and (ii) 8-directional chain codes as illustrated in Fig. 7.33.

Other than this, we have differential chain codes as illustrated in Fig. 7.34.

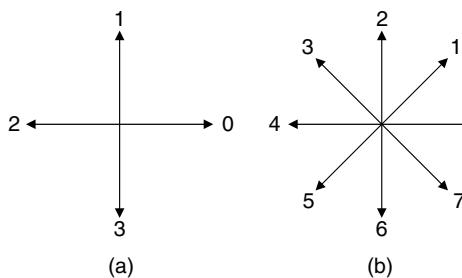


Fig. 7.33 (a) 4-directional chain code
(b) 8-directional chain code

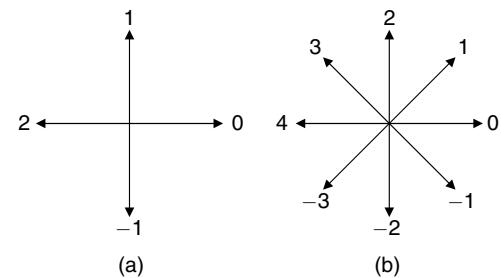


Fig. 7.34 (a) 4-directional differential chain code
(b) 8-directional differential chain code

Example 7.5 Construct a 4-directional and 8-directional chain code for the arbitrary shape given in Fig. 7.35(a).

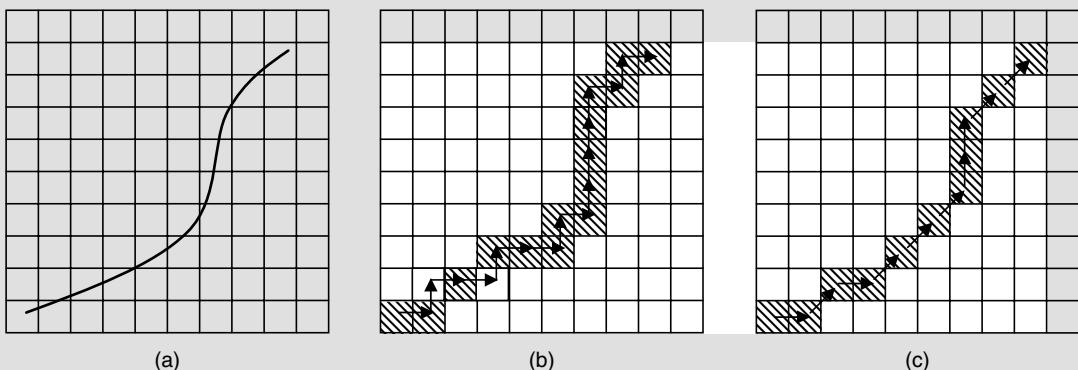


Fig. 7.35 (a) Given arbitrary curve (b) 4-directional chain-code representation (c) 8-directional chain-code representation

Solution The 4-directional chain code and the 8-directional chain code shown in Figs 7.35 (b) and 7.35 (c) respectively.

From Fig. 7.35(b) and Fig. 7.35 (c), it is obvious that the 4-directional chain code will not take the diagonal direction whereas the 8-directional chain code takes the diagonal direction into consideration. Hence, an 8-directional chain code can represent the contours more effectively than a 4-directional chain code.

Example 7.6 Prove that chain codes are invariant to translation.

Solution The chain codes are invariant to translation. This concept can be proved by means of a simple example. Consider an arbitrary shape as shown in Fig. 7.36.

In Fig. 7.36, the dot represents the starting point. The 4-directional chain code, difference code and the shape number of the shape is given below:

Chain code 1 0 0 0 0 3 3 2 1 2 2 2

Difference code 3 3 0 0 0 3 0 3 3 1 0 0

Shape number 0 0 0 3 0 3 3 1 0 0 3 3

Now, the arbitrary shape shown in Fig. 7.36 is translated as shown in Fig. 7.37.

The chain code, difference code and the shape number for the shape shown in Fig. 7.37 is given below:

Chain code 1 0 0 0 0 3 3 2 1 2 2 2

Difference code 3 3 0 0 0 3 0 3 3 1 0 0

Shape number 0 0 0 3 0 3 3 1 0 0 3 3

By comparing the chain code obtained for Fig. 7.36 with the chain code obtained for Fig. 7.37, it is obvious that the chain code is invariant to translation.

Drawbacks of Chain Code Any small disturbance along the boundary due to noise or imperfect segmentation causes changes in the code that may not be related to the shape of the boundary.

Fourier Descriptors Fourier descriptors are useful in describing the shapes of 2D closed contours. A Fourier descriptor is a Fourier transform derived from the outline of an object and has been used in pattern recognition. The region is assumed to be in the complex plane where the x -coordinate of each point represents a distance along the real axis, and the y -coordinate represents the distance along the imaginary axis. Thus, each boundary point can be viewed as a single complex number. Then tracing the boundary points results in a series of complex numbers. Taking the Fourier transform for this sequence of complex numbers will result in another sequence of complex numbers. Whatever changes are happening in the spatial domain will have an equal impact in the frequency or transform domain. For example, a change in size in the spatial domain is equivalent

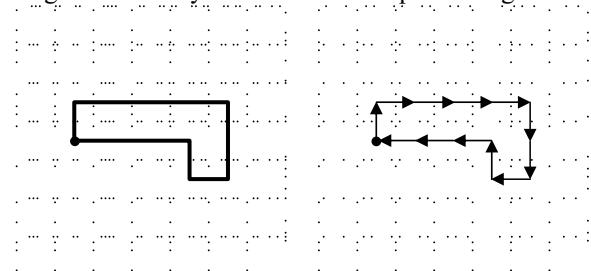


Fig. 7.36 Arbitrary shape for Example 7.6

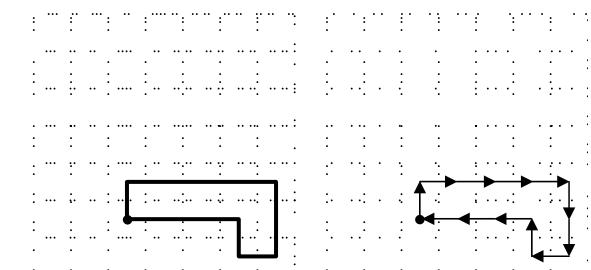


Fig. 7.37 Translation of the arbitrary shape shown in Fig. 7.36

to multiplication by a constant in the transform domain. The rotation of an angle about the origin will result in phase shift in the frequency domain. The translation of the image will result in change in the DC term of the transform coefficients. In order to move from one boundary point to another boundary point, either a four-neighbour or an eight-neighbour chain code can be used. The eight-neighbour chain code is more effective when compared to a four-neighbour chain code.

7.14.2 Polygonal Approximation

In polygonal approximation, a closed curve or a contour is approximated as a 2D polygon. Polygonal approximation provides a simple representation of the planar object boundary.

Let $X = \{x_1, x_2, \dots, x_n\}$ be a set of points on the boundary of a planar object to be approximated using a polygon. Polygonal approximation can be defined as a partitioning of the set into N mutually exclusive and collectively exhaustive subsets $\lambda_1, \lambda_2, \dots, \lambda_k$ such that each of the subsets can be approximated using a linear sequence of points. This can be achieved by minimisation of an objective function of the form

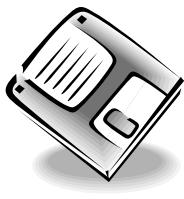
$$J = \sum_{i=1}^n d(x_i, l_j), \quad x_i \in \lambda_j \quad (7.45)$$

where l_j is the linear structure which approximates the points λ_j and d is a measure of deviation.

A powerful shape-representation technique should satisfy the following criteria:

1. The technique should be efficient and simple.
2. The shape-representation technique should be accurate and complete reconstruction should be possible.
3. The technique should be suitable for shape analysis and shape recognition.
4. The shape-representation technique should be compact, which implies that the representation should require a few bits in order to allow for its archiving.
5. The technique of shape representation should be robust, which implies that it should be as less sensitive as possible to small deformations which could alter the shape.

Summary



- The objective of image segmentation is to partition an image into meaningful parts that are relatively homogenous in a certain sense.
- Local segmentation deals with segmenting sub-images which are small windows on a whole image whereas global segmentation is concerned with segmenting a whole image.
- Image segmentation can be performed in three different approaches which are (i) region approach, (ii) boundary approach, and (iii) edge approach.
- Region growing is an approach to image segmentation in which neighbouring pixels are examined and added to a region class if no edges are detected. Region splitting is a top-down approach and it begins with a whole image and divides it up such that the segregated parts are more homogenous than the whole.
- The clustering technique attempts to access the relationships among patterns of the data set by organising the patterns into groups or clusters such that patterns within a cluster are more similar to each other than patterns belonging to different clusters.
- An image can be segmented by the thresholding operation in which a histogram of the image can be used as a tool to select the threshold value.

- Edges are basically discontinuities in the image intensity due to changes in the image structure. Robert, Prewitt, Sobel and Frei-Chen kernels can be used to detect the edges in an image.
- The Canny edge-detection algorithm is the most powerful and widely used algorithm used to detect edges in an image. The three phases in the Canny edge-detection algorithm are (i) smoothing and differentiation, (ii) non-maximal suppression, and (iii) thresholding.
- The snake algorithm and the watershed algorithm are powerful algorithms used to segment an image. In the watershed transformation, the image is considered as a topographic surface.
- Chain codes and polygonal approximation are widely used to represent the shapes in an image.

Review Questions

1. What is content-based image retrieval?

Content-based image retrieval is a technique for retrieving images on the basis of automatically derived features such as colour, texture and shape.

2. What is an 'edge' in an image? On what mathematical operation are the two basic approaches for edge detection based on?

An 'edge' is a region of an image where there is a sudden change in the image intensity. The basic edge detectors approximate the first or second derivative of the image function.

3. What are the three stages of the Canny edge detector? Briefly explain each phase.

The three phases in the Canny edge-detection algorithm are (i) smoothing and differentiation (ii) non-maximal suppression, and (iii) thresholding.

Smoothing and differentiation In this phase, the input image is smoothed with a Gaussian and the gradient is obtained in the x and y directions.

Non-maximal suppression This phase is used to eliminate edges that are not maxima in the direction perpendicular to the edge.

Thresholding Thresholding is used to eliminate edges lower than the threshold T_1 , keeping edges greater than the threshold T_2 . The edges between T_1 and T_2 are kept if they are connected to another edge.

4. Compare the Canny edge detector with the Laplacian of Gaussian edge detector.

Canny edge detector	Laplacian of Gaussian edge detector
<p>The Canny edge detector is non-isotropic.</p> <p>The Canny edge-detection algorithm uses first derivative to compute the edge.</p> <p>The Canny edge detector is more likely to produce long, thin contours because of non-maximum suppression which thins, and hysteresis thresholding which can fill in weak edge gaps.</p>	<p>Laplacian-of-Gaussian edge detector is isotropic (invariant to rotation).</p> <p>Laplacian-of-Gaussian edge detector computes second derivative to compute the edge.</p> <p>This is not so in Laplacian of Gaussian edge detection.</p>

5. Give one representation scheme for the boundary of an object and describe how it can be computed.

Chain code is one representation scheme for the boundary of an object. In chain code, the boundary is defined by a code describing the direction of the steps that are taken between boundary pixels when moving around the object.

Polygonal approximation is another representation for the boundary of an object. In polygonal approximation, the boundary is approximated by a polygon.

6. When the gradient of an image is used to detect edges, the main problem is that differentiation enhances noise. How is this problem generally addressed?

A smoothing filter is usually used to reduce the noise. However, this also reduces the clarity of the edges and delocalises the edges from their original location.

7. Distinguish between local and global thresholding techniques for image segmentation.

In the global thresholding technique, a grayscale image is converted into a binary image based on image intensity value. All pixels having values greater than the global threshold values are marked as 1 and the remaining pixels are marked as 0.

In the local thresholding method, the thresholding operation depends on local image characteristics. Local thresholding is superior to the global threshold method in the case of poorly illuminated images.

8. Show that a two-dimensional Gaussian is separable, while the Laplacian of a Gaussian operator (LOG) is not separable.

A spatial filter is separable if it can be written as a product of two functions, one that is a function of only m and the other that is a function of only n . The two-dimensional Gaussian function is given by

$$f(m, n) = ce^{-(m^2+n^2)/2\sigma^2}$$

The above expression can be written as

$$f(m, n) = ce^{-m^2/2\sigma^2} e^{-n^2/2\sigma^2} = f(m)f(n)$$

Thus 2D Gaussian function is separable.

The Laplacian of Gaussian operator is given by

$$\nabla^2 f(m, n) = \frac{1}{\pi\sigma^4} \left(1 - \frac{m^2 + n^2}{2\sigma^2}\right) e^{-(m^2+n^2)/2\sigma^2}$$

The Laplacian of Gaussian function cannot be decomposed into two functions, one only for m and the other only for n . Hence, Laplacian of Gaussian operator is not separable.

9. What is the advantage of separable filters?

The convolution of an image with an $N \times N$ filter implies N^2 operations per pixel. If the filter can be written as the outer product of two vectors of size $N \times 1$ each, these two vectors may be used as two 1D filters applied one after the other to the image. In such a case, the N^2 operations per pixel are replaced by $2N$ operations.

10. The region-growing algorithm starts with a seed pixel. The selection of the seed pixel depends on application. You are given two applications: (a) target detection in night vision, and (b) mammogram. Suggest a way to choose the seed pixel in these two applications.

For night-vision applications, infrared imaging is commonly used. In infrared imaging, the targets of interest are generally hotter than the background. Hence, it is better to start with the brightest pixels.

A mammogram is widely used to detect breast cancer. In a mammogram, calcifications are brighter than almost everything else. Hence, it is better to choose the seed pixel as the brightest pixel.

11. What are the advantages/disadvantages if we use more than one seed in a region-growing technique?

The advantage of using more than one seed is that better segmentation of the image can be expected, since more seeds lead to more homogeneous regions.

The drawback of using more than one seed is that the probability of splitting a homogeneous region in two or more segments increases.

12. Does the use of chain code compress the description information of an object contour?

Chain codes provide a good compression of boundary description, because each chain-code element can be coded with only two bits.

13. What is the basic concept behind Fourier-descriptor-based boundary representation?

The basic idea in a Fourier descriptor is that the boundary of an object is viewed as a 1D periodic signal. For this, a periodic signal forward Fourier transform is taken to get the coefficients. The DC coefficient $F(0)$ contains the centre of mass of the object, and the coefficients $F(1), F(2) \dots F(M - 1)$ describe the object in increasing detail. These features depend on the rotation, scaling and starting point on the contour. All the coefficients are not used to describe the features. Some of the coefficients are truncated to $F(N)$ where $N < M$. On taking inverse Fourier transform of the truncated coefficients, one gets the approximation of the original contour. The Fourier descriptors can be invariant to translation and rotation if the boundary description is approximately chosen.

14. Give two applications of image-segmentation techniques.

Image segmentation is used to detect cancerous cells from medical images. It can also be used to detect roads from satellite images.

15. A person wishes to apply the Laplacian-of-a-Gaussian edge operator to an image $f(m, n)$ of size 256×256 . The size of the edge operator is 32×32 , and the origin is at its centre. Describe in words how to perform the operation in frequency domain.

The following operations have to be performed in the frequency domain.

Step 1 Create the 32×32 edge operator in the spatial domain, and zero pad it to size 256×256 .

Step 2 Compute the 2D Fourier transforms of the image and the zero-padded edge operator.

Step 3 Compute the product of the Fourier transforms.

Step 4 Take the inverse Fourier transform of the result obtained in Step 3.

Step 5 Shift the resultant image obtained in Step 4 by $(-16, -16)$. This is because the convolution assumes that the origin of the operator is at its upper left corner, instead of the middle.

16. Consider the image segment

128	128	128	64	64	32	32	8
64	64	128	128	128	8	32	32
32	8	64	128	128	64	64	64
8	128	128	64	64	8	64	64
128	64	64	64	128	128	8	8
64	64	64	128	128	128	32	32
8	128	32	64	64	128	128	128
8	8	64	64	128	128	64	64

Based on the histogram, segment the image into two regions.

Step 1 Computation of the histogram of the input image

The histogram of the image gives the frequency of occurrence of the gray level. The histogram of the image is given as

Gray level	128	64	32	8
Frequency of occurrence	22	24	8	10

Step 2 Segmentation based on histogram

From the histogram of the image, it is obvious that the gray level which occurs least is '32'. The threshold is fixed as 32. Now the input image is divided into two regions as follows:

Region 1: Gray level ≤ 32

Region 2: Gray level > 32

2	2	2	2	2	1	1	1
2	2	2	2	2	1	1	1
1	1	2	2	2	2	2	2
1	2	2	2	2	1	2	2
2	2	2	2	2	2	1	1
2	2	2	2	2	2	1	1
1	2	1	2	2	2	2	2
1	1	2	2	2	2	2	2

The input image after this decision is given as

17. Distinguish between image segmentation based on thresholding with image segmentation based on region-growing techniques.

Image segmentation based on thresholding applies a single fixed criterion to all pixels in the image simultaneously. Hence, it is rigid. On the other hand, image segmentation based on the region-based approach is more flexible; hence it is possible to adjust the acceptance criteria in the course of region-growing process so that they can depend on the shape of the growing regions if desired.

Problems

7.1 Consider an image $f(m, n) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 & 9 \\ 7 & 8 & 9 & 10 & 11 \\ 9 & 10 & 11 & 12 & 13 \end{bmatrix}$

- Compute the magnitude and the direction of the gradient for the marked pixel.
- 7.2 Prove that the Prewitt edge detector along a horizontal direction can be obtained by convolving two one-dimensional signals $[1, 1, 1]$ and $[-1, 0, 1]^T$ and then scaling the result by a factor of $\frac{1}{3}$.
- 7.3 Obtain the frequency response of the following edge detectors:

- (a) Prewitt operator
 - (b) Sobel operator
 - (c) Laplacian operator
- 7.4 Explain the principle of the following region-based segmentation procedures:
- (i) Region growing
 - (ii) Region splitting
 - (iii) Split and merge

- What is difference do you find in these three approaches?
- 7.5 Explain why watershed segmentation tends to over-segment images. Also, mention one solution to overcome the problem of over-segmentation.
- 7.6 The Sobel operator computes the following quantity at each pixel location in an image array A .

$$G_x[m, n] = (A[m+1, n+1] + 2A[m+1, n] + A[m+1, n-1]) \\ - (A[m-1, n+1] + 2A[m-1, n] + A[m-1, n-1])$$

$$G_y[m, n] = (A[m-1, n-1] + 2A[m, n-1] + A[m+1, n-1]) \\ - (A[m-1, n+1] + 2A[m, n+1] + A[m+1, n+1])$$

- $G[m, n] = |G_x[m, n]| + |G_y[m, n]|$. The position of $A[m, n]$ is the column m and row n of the array.
- Write the 3×3 array for each mask, M_x and M_y . What mathematical operation on an image array is approximated by the Sobel operator? Show how the Sobel operator is related to the mathematical operation.
- 7.7 Explain region splitting and merging technique for image segmentation with suitable examples.
- 7.8 Obtain the 4-directional chain code and the shape number for the arbitrary-shape shown in Fig. 7.38.

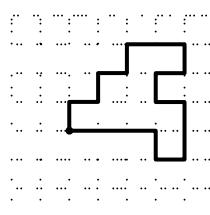


Fig. 7.38 Arbitrary shape

- 7.9 Give the 8-directional chain code and the shape number for the arbitrary shape shown in Fig. 7.39.

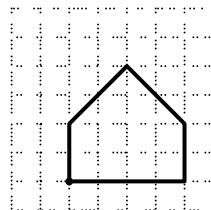


Fig. 7.39 Arbitrary shape

- 7.10 The energy functional that is used with the active contours usually contains three terms— E_{int} , E_{image} and E_{ext} . For each of these three terms, explain briefly what it measures, how it is defined, and what happens if the term is omitted.

7.11 Segment the image shown in Fig. 7.40 using the split-and-merge procedure. Also, show the quadtree corresponding to the segmentation.

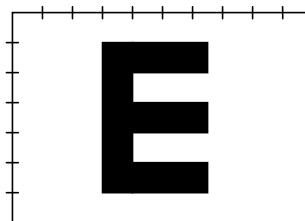


Fig. 7.40 Image to be segmented

- 7.12 Give the quadtree representation of the binary image given in Fig. 7.41.

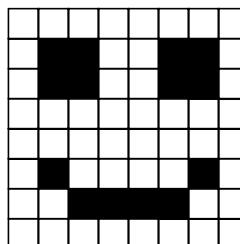
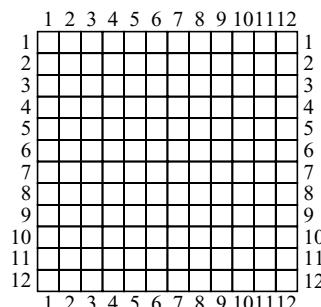


Fig. 7.41 *Binary image*

- 7.13 The 8-directional chain code of the image is given by

(2, 4), 0, 7, 7, 7, 7, 7, 5, 5, 5, 3, 2, 2, 1, 1, 1, 1, 1, 1, 1

where the first two numbers (2, 4) provide the column and row axes of the initial point respectively. Decode the chain code and draw the decoded image in the grid shown below.



7.14 Give the linear filter masks for the following operations:

- (a) Detecting horizontal lines
- (b) Detecting vertical edges

References

BOOKS

1. M Sonka, V Hlavac and R Boyle, *Image Processing, Analysis and Machine Vision*, Boston: PWS Publishing, 1999
2. Gregory A Baxes, *Digital Image Processing: A Primer*, John Wiley & Sons, 1994
3. D Ballard and C Brown, *Computer Vision*, Prentice Hall, Englewood Cliffs, New Jersey, 1982
4. R J Schalkoff, *Digital Image Processing and Computer Vision*, John Wiley and Sons Inc. 1989
5. J Hartigan, *Clustering Algorithms*, Wiley, New York, 1975
6. Jean Michel Morel and Sergio Solimini, *Variational Methods in Image Segmentation*, Birkhauser Boston, 1995

JOURNALS

1. Densheng Zhang and Guojun Lu, *Review of Shape Representation and Description Techniques*, Pattern Recognition, pp. 1–19, 2004
2. V Grau et al, *Improved Watershed Tranform for Medical Image Segmentation using Prior Information*, IEEE Transactions on Medical Imaging, vol. 23, No. 4, April 2004
3. J Canny, *A Computational Approach to Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, pp. 639–643, Nov. 1986
4. R M Haralick and L G Shapiro, *Image Segmentation Techniques*, Computer Vision, Graphics, and Image Processing, vol. 29, pp. 100–132, 1985
5. S Y Chen, W C Lin and C T Chen, *Split-and-Merge Image Segmentation based on Localised Feature Analysis and Statistical Tests*, CVGIP: Graphical Models and Image Processing vol. 5, pp. 457–475, 1991

8

Learning Objectives

Object recognition is basically an attempt to mimic the human capability to distinguish different objects in an image. The automatic object-recognition concept is widely used in industry. Many applications involve recognition of patterns in data. The field of pattern or object recognition provides a general, domain-independent technique for data analysis. After reading this chapter, the reader should be familiar with the following topics:

patterns and pattern classes

parametric and non-parametric method of object recognition

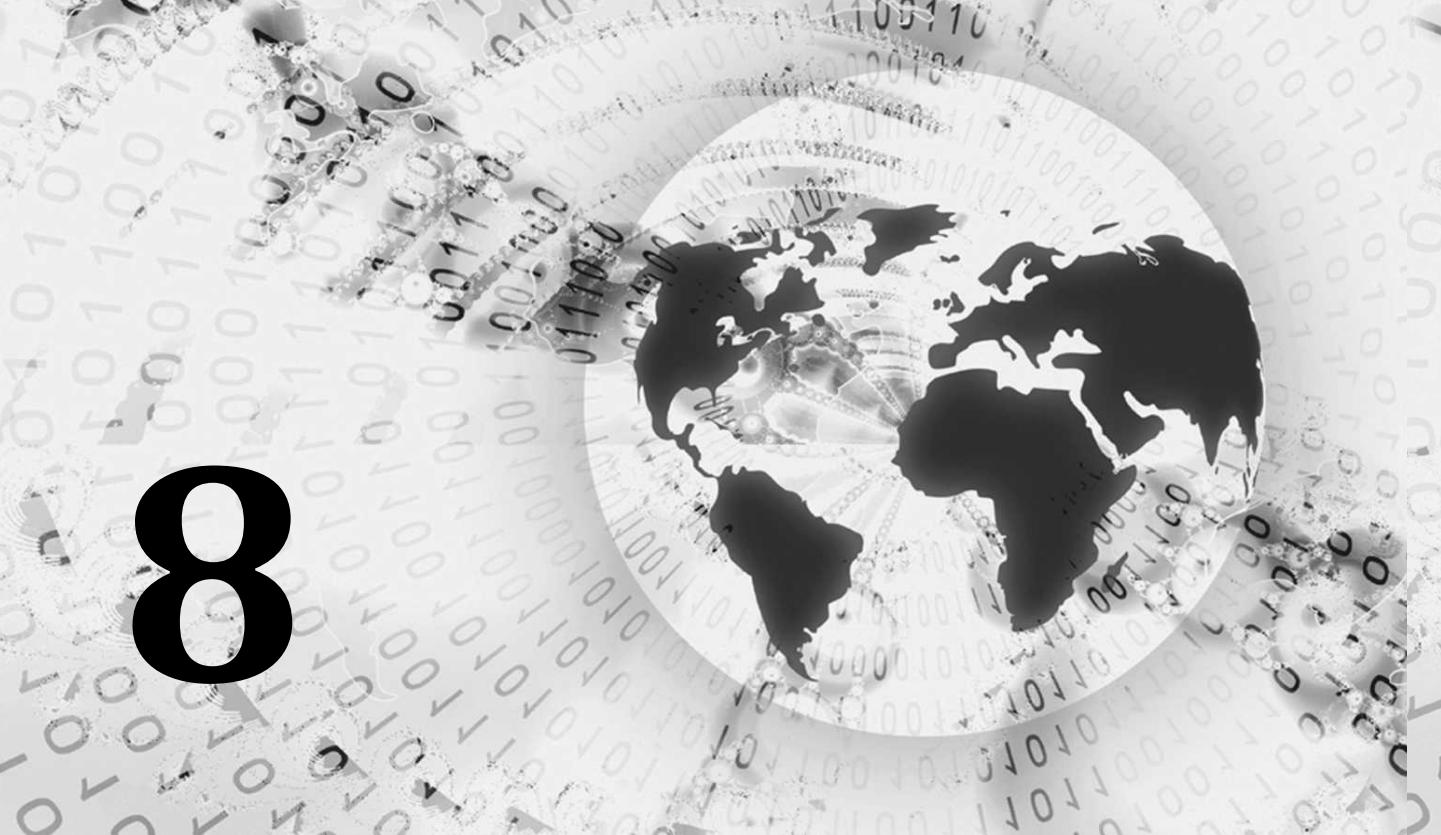
bayesian approach to pattern recognition

Neural network approach to pattern recognition

Structural pattern recognition

Applications of object recognition

Object Recognition



8.1 INTRODUCTION

Object recognition is an amazing human feat. Humans effortlessly recognise objects within a fraction of a second. Humans possess a remarkable ability to determine what things are by simply looking at them. Unlike computers, humans can easily recognise a face, understand spoken words, and distinguish different styles of handwriting. An attempt to develop a machine to mimic human capability is the starting point of object recognition. The term 'recognition' has been used to refer to many different visual abilities, including identification, categorisation, and discrimination. In fact, recognising an object means successfully categorising it as an instance of a particular object class.

8.2 NEED FOR AN OBJECT-RECOGNITION SYSTEM

Object recognition is the study of how machines can observe the environment, learn to distinguish patterns of interest and make reasonable decisions about the categories of patterns. Object recognition is widely used in the machine-vision industry for the purpose of inspection, registration and manipulation. As the number of expected targets becomes larger and larger, it becomes difficult for a human observer to identify an object from images of poor quality. Hence, there is a need to develop a knowledge-based recognition system. The performance of a machine may be better than the performance of a human in a noisy environment due to the following factors:

- (i) Human performance degrades with increasing number of targets, whereas the performance of a machine does not depend on the size of the set of targets.
- (ii) The performance of a machine does not degrade due to fatigue caused by prolonged effort.

A knowledge-based system is desirable for reliable, quick and accurate recognition of objects from noisy and partial input images.

8.3 AUTOMATED OBJECT-RECOGNITION SYSTEMS

Automated object-recognition systems use computers that execute programs of instruction to implement mathematical algorithms. The general block diagram of an object-recognition system used by a living organism and a computer is illustrated in Fig. 8.1.

The ability of living organisms to solve identification problems is rooted in their perceptual and cognitive abilities to collect and analyse information from the environment. An automated pattern-recognition system focuses on mechanising the abilities of living organisms. In contrast to biological systems, automated

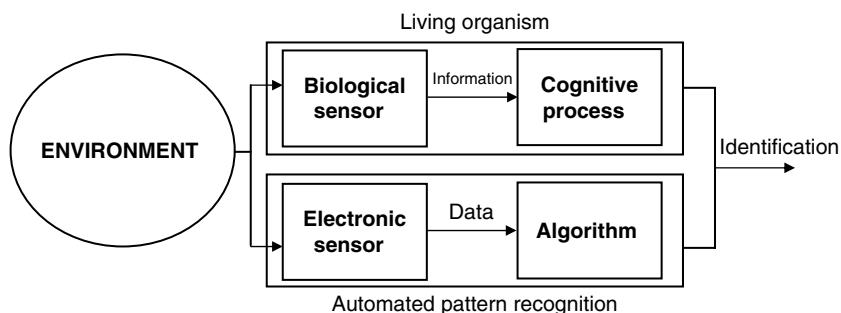
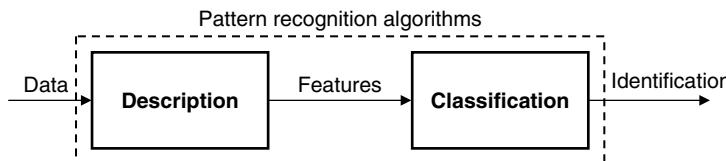


Fig. 8.1 Automated pattern-recognition system

**Fig. 8.2** Pattern-recognition algorithm

pattern-recognition systems use algorithms to process data collected either electronically through sensors or transcribed by a human. Figure 8.1 illustrates the parallel between living organisms and automated object or pattern-recognition systems.

The object-recognition system is divided into two parts—feature extractor and classifier as illustrated in Fig. 8.2. The feature extractor reduces the dimensionality of the input vectors to the classifier. The classification task uses a classifier to map a feature vector to a group. Once formulated, the mapping can be used to assign identification to each unlabeled feature vector subsequently presented to the classifier. A special case of feature extraction is *feature selection* that selects a subset of given measurements as features. Feature selection can be considered as a mapping from the primitive n -dimensional space to a lower-dimensional space. The features selected should discriminate an object belonging to a different class. For example, we want to distinguish an African person from a Japanese person, then skin colour can be a good feature. On the other hand, if want to distinguish an Indian from a Japanese then colour cannot be an ideal feature. In such a case, the sharpness of the nose (length of the nose) can also be considered as one of the factors to distinguish an Indian from a Japanese. Many important applications of pattern (object) recognition can be characterised either as waveform classification or classification of geometric figures. For example, consider the problem of testing the functionality of the heart. Electrocardiogram (ECG) is a recording of the electrical activity of the heart. An ECG waveform is given in Fig. 8.3.

The most primitive way to distinguish the normal activity of the heart and cardiac arrhythmia (which disrupts the rhythm of the heartbeat), is to measure the time samples of the ECG waveform, $x(t_1), \dots, x(t_n)$. These measurements of time samples are represented in Eq. (8.1).

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ \vdots \\ x(t_n) \end{bmatrix} \quad (8.1)$$

The measurements $x(t_1), x(t_2), \dots, x(t_n)$ form a vector ' \mathbf{X} ' which is given in Eq. (8.1). The measured ECG values will vary with the physiological activity. For example, the ECG taken after jogging will be different from the

**Fig. 8.3** ECG waveform

ECG taken after deep sleep. Hence, $x(t_i)$ can be considered as a random variable and X is considered as a random vector if its components are random variables. Since each of the primitive measurements $x(t_i)$ carries a small of information about the sample, the number of measurements n usually becomes very large. This high dimensionality makes many pattern-recognition problems difficult to solve. As the number of inputs to the classifier becomes smaller, the design of the classifier would be simple. In order to enjoy this advantage, it is important to extract important features from the observed samples. This problem is called feature selection or feature extraction, and it is very vital for efficient classification.

8.4 PATTERNS AND PATTERN CLASS

A pattern is used to denote P -dimensional data vectors $\mathbf{x} = (x_1, \dots, x_p)^T$ of measurements whose components x_i are measurements of features of the object. Pattern is the language of nature. In order to understand the objects in an image, it is necessary to perceive the patterns in the image. Statistical pattern recognition assumes that an image may contain one or more objects and that each object belongs to one of the several predetermined types, categories or pattern classes. In the case of English character recognition, there are twenty-six letters in the English alphabet from A, B, C ... Z. Hence, twenty-six classes are necessary for English character recognition.

If one wants to classify the elephants on the earth, they can obviously be classified into two classes, namely 'African elephant' and 'Asian elephant'. Within the class 'African elephant', we can further subdivide into 'male' and 'female' elephant, and so on. One possible pattern class for elephants is given in Fig. 8.4.

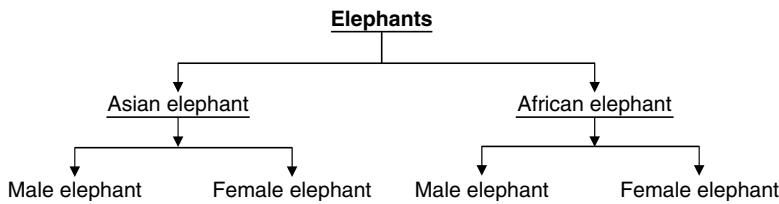


Fig. 8.4 Pattern class for elephants

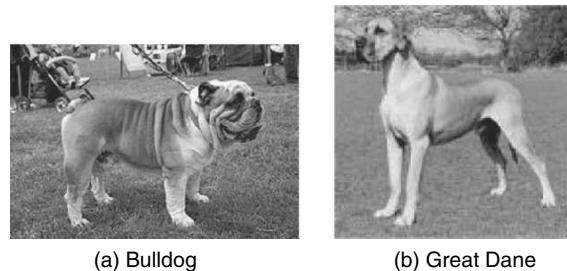
8.4.1 Representations of Pattern Classes

Three common representations of patterns are (i) vector representation, (ii) string representation, and (iii) tree representation.

(i) Vector Representation of Patterns Vectors are basically quantitative description of patterns. Suppose that N features are to be measured from each input pattern. Each set of N features can be considered as a vector X called a feature vector. Consider two breeds of dogs—'Bulldog' and 'Great Danes' as shown in Fig. 8.5.

Two features which distinguishes the bulldog from the Great Dane are 'size of the mouth' and 'height'. The height of a Great Dane is more than the height of a bulldog. Also, the width of the mouth of a bulldog is greater than the width of the mouth of a Great Dane. Hence, 'height' and 'size of the mouth' can be considered as two measurements which lead to a 2D pattern vector of the form

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (8.2)$$

**Fig. 8.5** Dog breeds

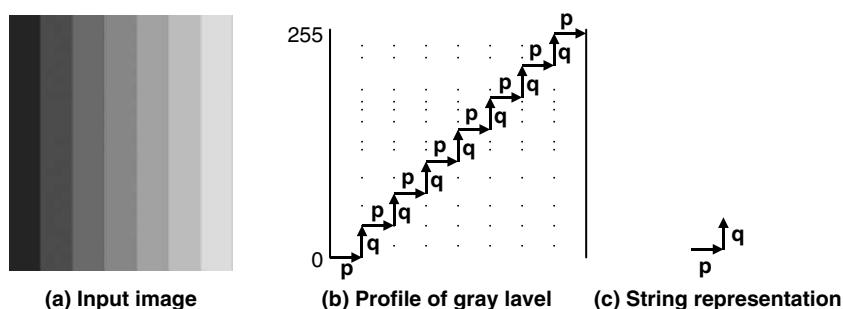
where x_1 and x_2 corresponds to the height and size of the mouth of the dog. The two pattern classes are ω_1 and ω_2 where ω_1 corresponds to the class 'Bulldog' and ω_2 corresponds to the class 'Great Dane'.

(ii) String Representation of Patterns String descriptions generate patterns of entities whose structures are based on simple connectivity of primitives. In order to understand string representation of patterns, consider the image shown in Fig. 8.6 (a). The image is basically a gradual variation of gray level from black to white.

The gray-level profile of the image is shown in Fig. 8.6 (b) which depicts the fact that black will take the gray level of zero, and white will take the gray level of 255. The gray-level profile of the input image can be represented as a staircase pattern. The staircase pattern can be represented by two primitives p and q which are shown in Fig. 8.6(c). The input image can be represented by a strings of symbols p , q , p , q In this example, gray level is considered as the pattern and this gray level is represented by series of strings p , q , p , q

(iii) Tree Description of Patterns Tree descriptions represent the hierarchical ordering of objects in an image. Consider the group of animals illustrated in Fig. 8.7.

The group of animals illustrated in Fig. 8.7 can be represented in a hierarchical manner as shown in Fig. 8.8. The animals can be broadly classified into domestic and wild animals, which can be further classified into herbivores and carnivores. The hierarchical representation will result in a tree-structure which is shown in Fig. 8.8.

**Fig. 8.6** String representation of an image

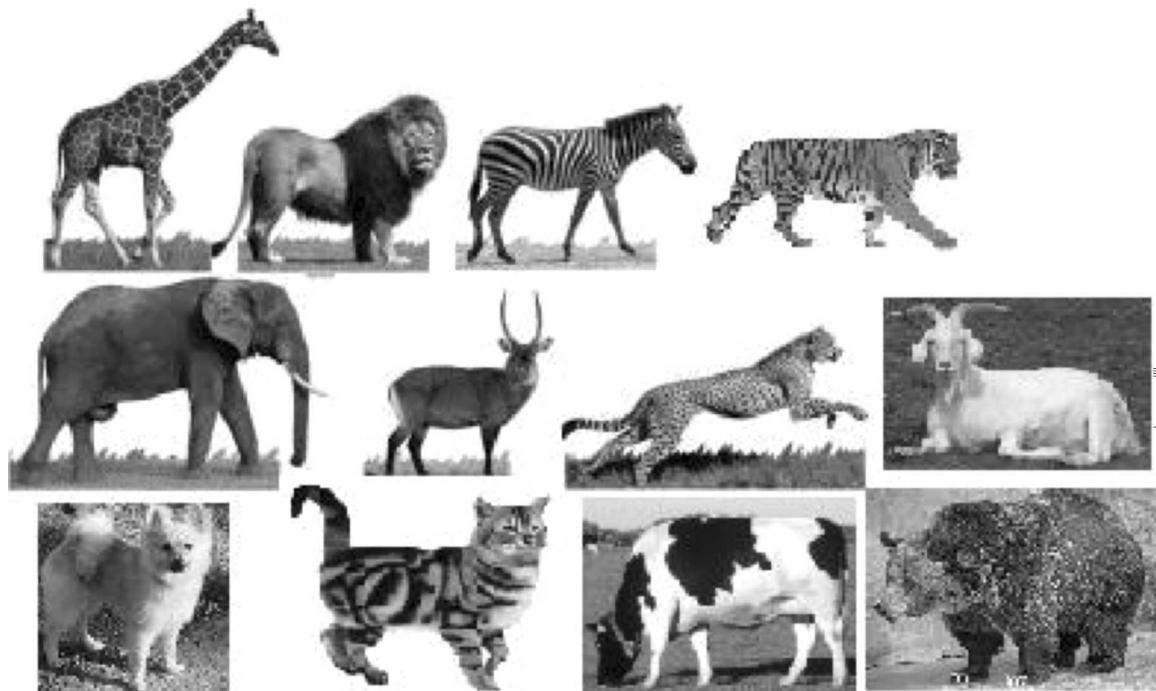


Fig. 8.7 Group of animals

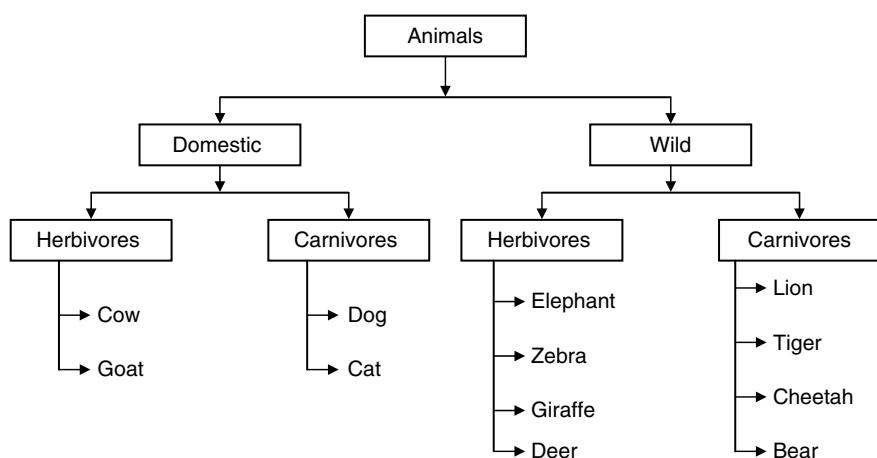


Fig. 8.8 Hierarchical representation of animals

8.5 SELECTION OF MEASUREMENT PARAMETERS

The selection of measurements is very important in designing a pattern-recognition machine, and is a major factor in its success or failure. There is no general theory on measurement selection because the selection usually depends on the particular pattern-recognition problem under consideration.

Suppose one wants to distinguish 'Asian Elephant' from 'African Elephant'. A common mode of learning is to be given a collection of labeled examples, known as *training data set*. From the training data set, structure information is distilled and used for classifying new input. By seeing different types of elephants in a zoo, one common feature that separates the Asian elephant from the African elephant is the nature of the skin. In general, the African elephant's skin is more wrinkled than the Asian elephant's skin as illustrated in Fig. 8.9.

From the set of observations, it is possible to find a few old Asian elephants which have wrinkles in their skin. The classes of Asian and African elephants overlap if wrinkle alone is taken as a feature to distinguish them. This is illustrated in Fig. 8.10. Hence wrinkles in the skin alone cannot be taken as a parameter to distinguish the Asian elephant from the African elephant. Another distinguishing feature which separates the African elephant from the Asian elephant is the size of the ear. In most of the cases, the size of the ear of the African elephant is greater than the size of the ear of the Asian elephant. The size of the ear together with wrinkles in the skin can be used as parameters to effectively classify the Asian and African elephants, which is illustrated in Fig. 8.11.

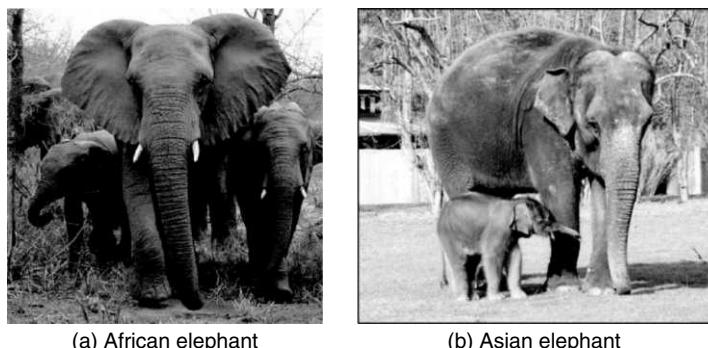


Fig. 8.9 Elephant classification

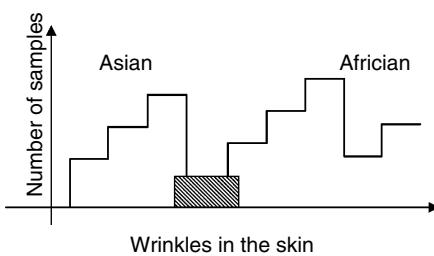


Fig. 8.10 Pattern classification using a single feature

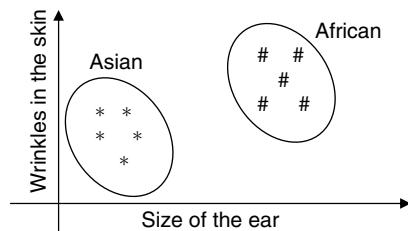


Fig. 8.11 Pattern classification using two features

8.6 RELATIONSHIP BETWEEN IMAGE PROCESSING AND OBJECT RECOGNITION

Image processing deals with different techniques which can improve the visual quality of the input image, whereas object recognition is concerned with the description and classification of objects or entities in the image. Image processing is often necessary as a preprocessing stage preceding pattern recognition. Image processing is required to improve the qualities and measurement of an image.

Consider the example of a face-recognition system as shown in Fig. 8.12. If the face image is transmitted through a channel, definitely there is a possibility that the image will be corrupted by noise. In such cases, it is necessary to preprocess the image (filtering) before extracting the feature from the image for efficient classification and recognition.



Fig. 8.12 Face-recognition system

8.7 APPROACHES TO OBJECT RECOGNITION

Basically, there are two approaches to pattern recognition. They are (i) decision-theoretic approach, which is also called statistical approach, and (ii) structural approach, or syntactic approach, as shown in Fig. 8.13. In the decision-theoretic approach, the pattern is represented as a vector in a vector space (feature space). Then a decision algorithm, which is mainly based on the statistical concept, is used to decide which class the pattern belongs to. In the structural method, the pattern is represented by its structure, e.g., a string of symbols, a graph connecting the primary elements, etc. The decision-theoretic method can be broadly classified into the classical and neural network approaches. The classical approach depends on the statistics of the input data to be classified; hence the classical method is also known as statistical approach to pattern recognition/object recognition. The classical method can be further subdivided into parametric and non-parametric methods. Parametric classification refers to the development of statically defined discriminant functions in which the underlying probability density function is assumed to be known. If the functional form of some or

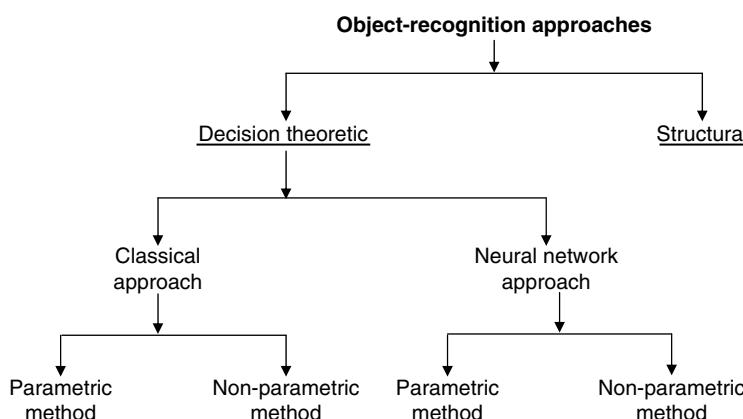


Fig. 8.13 Classification of object-recognition methods

all the conditional Probability Density Functions (PDF) is unknown, the classifier is termed non-parametric. This means that all conditional PDFs must be obtained from the training data set. No single technique or model is suited for all pattern-recognition problems, and hence we have different types of pattern-recognition approaches.

8.8 BAYES' PARAMETRIC CLASSIFICATION

Let x_1, x_2, \dots, x_N represent the random variables, where x_i is the noisy measurement of the i^{th} feature. For each pattern class $\omega_j, j = 1, 2, \dots, m$, assume that the multivariate probability density function of the feature vector \mathbf{X} , $p(X / \omega_j)$, and the probability of occurrences of, $P(\omega_j)$, are known. On the basis of the priori information $p(X / \omega_j)$ and $P(\omega_j)$, $j = 1, 2, \dots, m$, the function of the classifier is to perform the classification task for minimising the probability of misrecognition. The problem of pattern classification can now be formulated as a statistical decision problem by defining a decision function $d(X)$. Let $L(\omega_i, d_j)$ be the loss incurred by the classifier if the decision d_j is made when the input pattern is actually from ω_i . The conditional loss or conditional risk is given by

$$r(\omega_i, d) = \int_{\Omega_x} L(\omega_i, d) p(X / \omega_i) dX \quad (8.3)$$

For a given set of a priori probabilities $P = \{P(\omega_1), P(\omega_2), \dots, P(\omega_m)\}$, the average loss is given by

$$R(P, d) = \sum_{i=1}^m P(\omega_i) r(\omega_i, d) \quad (8.4)$$

Substituting (8.3) in (8.4), we get

$$r_X(P, d) = \frac{\sum_{i=1}^m L(\omega_i, d) P(\omega_i) p(X / \omega_i)}{p(X)} \quad (8.5)$$

The average loss can be expressed as

$$R(P, d) = \int_{\Omega_x} p(X) r_X(P, d) dX \quad (8.6)$$

$r_X(P, d)$ is defined as the posteriori conditional average loss of the decision d for a given measurement X . The problem is to choose a proper decision $d_j, j = 1, 2, \dots, m$ to minimise the average loss $R(P, d)$ or to minimise the maximum of the conditional average loss $r(\omega_i, d)$. The optimum decision rule which minimises the average loss is called Bayes' rule. If d^* is an optimal decision with respect to the average loss then

$$r_X(P, d^*) \leq r_X(P, d) \quad (8.7)$$

that is

$$\sum_{i=1}^m L(\omega_i, d^*) P(\omega_i) p(X / \omega_i) \leq \sum_{i=1}^m L(\omega_i, d) P(\omega_i) p(X / \omega_i) \quad (8.8)$$

For $(0, 1)$ loss function, the loss incurred by the classifier is given by

$$L(\omega_i, d_j) = 1 - \delta_{ij} = \begin{cases} 0, & i = j \\ 1, & i \neq j \end{cases} \quad (8.9)$$

The average loss is the probability of misrecognition. In this case, Bayes' decision rule is that

$$d^* = d_i, \text{ i.e., } X \sim \omega_i \text{ if}$$

$$P(\omega_i)p(X / \omega_i) \geq P(\omega_j)p(X / \omega_j) \quad \text{for all } j = 1, 2, \dots, m$$

The likelihood ratio λ between ω_i and the class ω_j is given by

$$\lambda = \frac{p(X / \omega_i)}{p(X / \omega_j)} \quad (8.10)$$

Then

$$d^* = d_i \text{ if } \lambda \geq \frac{P(\omega_j)}{P(\omega_i)} \quad \text{for all } j = 1, 2, \dots, m$$

The classifier implementing Bayes' decision rule for classification is called Bayes' classifier. The discriminant function implemented by a Bayes' classifier is given by

$$D_i(X) = P(\omega_i)p(X / \omega_i), \quad i = 1, 2, \dots, m$$

or it is given by

$$D_i(X) = \log[P(\omega_i)p(X / \omega_i)], \quad i = 1, 2, \dots, m$$

The decision boundary between regions in Ω_x associated with ω_i and ω_j is given by

$$P(\omega_i)p(X / \omega_i) - P(\omega_j)p(X / \omega_j) = 0$$

The above equation can be written as

$$\log \frac{P(\omega_i)p(X / \omega_i)}{P(\omega_j)p(X / \omega_j)} = 0 \quad (8.11)$$

8.8.1 Bayes' Classifier for a Gaussian Density Function

Let us consider $p(X / \omega_i), i = 1, 2, \dots, m$ be a multivariate Gaussian density function with mean vector M_i and covariance matrix K_i . The covariance matrix K_i is always symmetric and positive semi-definite.

$$p(X / \omega_i) = \frac{1}{2\pi|K_i|^{1/2}} \exp \left[-\frac{1}{2} (X - M_i)^T K_i^{-1} (X - M_i) \right], \quad i = 1, 2, \dots, m$$

The decision boundary is expressed as

$$\log \frac{P(\omega_i)}{P(\omega_j)} + \log \frac{p(X / \omega_i)}{p(X / \omega_j)} = \log \frac{P(\omega_i)}{P(\omega_j)} - \frac{1}{2} \left[(X - M_i)^T K_i^{-1} (X - M_i) - (X - M_j)^T K_j^{-1} (X - M_j) \right] = 0$$

In the above equation if $K_i = K_j = K$ it reduces to

$$X^T K^{-1} (M_i - M_j) - \frac{1}{2} (M_i + M_j)^T K^{-1} (M_i - M_j) + \log \frac{P(\omega_i)}{P(\omega_j)} = 0 \quad (8.12)$$

The above equation represents a hyperplane.

8.8.2 Discriminant Functions

If there are K pattern classes $\omega_1, \omega_2, \dots, \omega_k$ with the defining prototypes $y_m^{(k)}$ where $m = 1, 2, \dots, M_k$ counts the number of prototypes within a given class then it is desirable to have a function which measures each point in the pattern or feature space and assigns to that point a value as to its degree of membership to a given class. Such functions are termed *characteristic functions* in the context of fuzzy set theory, *discriminant functions* in the context of pattern recognition and *probability density functions* in statistical decision theory. Discriminant functions have the property that they partition the pattern or feature space into mutually exclusive regions, each region contributing to the domain of a class. The discriminant functions can be classified into (i) linear discriminant functions, (ii) piecewise linear discriminant functions, (iii) minimum distance classifier, and (iv) polynomial discriminant functions.

8.9 TEMPLATE-MATCHING-BASED OBJECT RECOGNITION

The current commercial systems for object recognition depend exclusively on correlation-based template matching. Given a grayscale image I , the need is to define a template or filter mask F containing a replica of an object to be located in the image. The basic matched filtering technique is to perform cross correlation of the input image with the template which may be represented as

$$I'(x, y) = \sum_m \sum_n F(m, n) I(m - x, n - y) \quad (8.13)$$

The resulting $I'(x, y)$ contains peaks at the location of the matches between the template and the underlying object. Cross-correlation with a large template is computationally expensive because the number of operations is proportional to the size of the mask. In practice, non-uniform illumination may cause uneven image background intensity and it may be necessary to first normalise the image at each pixel with respect to its local intensity level and intensity range.

Template matching is not reliable with respect to object rotation, scale, illumination and 3D pose and are allowed to vary even more when dealing with partial visibility and large model databases.

8.10 NON-PARAMETRIC DENSITY ESTIMATION

Parametric modeling of a probability density function assumes that the forms of probability density functions are known. Such knowledge typically comes from either a scientific analysis of the physical process or from empirical analysis of the observed data. However, simple parametric models do not accurately explain the physical processes. Virtually all the parametric PDF models are unimodal, but many practical models exhibit multimodal PDFs. Attempts at modeling high-dimensional multimodal PDFs as products of 1D parametric PDFs do not succeed well in practice. Hence there is a need to employ a more sophisticated non-parametric density estimation technique that does not make any assumptions about the forms of the probability density function except the mild assumption that the probability density functions are smooth functions. One such technique is Parzen-window density estimation.

8.11 NEURAL-NETWORK APPROACH TO OBJECT RECOGNITION

Neural networks are basically information-processing systems that have certain performance characteristics in common with biological neural networks. The characteristics of brain functions inspired the development of artificial neural networks. The brain contains a large number of highly connected components, called *neurons*. The number of neurons is around 10^{10} – 10^{12} . Biological networks achieve excellent recognition performance through a dense interconnection of neurons. The total number of interconnections is around 10^{14} . The neurons have three principal components—the dendrites, the cell body and the axon, which is illustrated in Fig. 8.14. The dendrites are tree-like receptive networks of nerve fibres that carry electrical signals into the cell body. The cell body effectively sums and thresholds these incoming signals. The axon is a single long fibre that carries the signal from the cell body out to the other neurons. The point of contact between an axon of one cell and a dendrite of another cell is called a *synapse*. The arrangement of neurons and the strengths of the individual synapses, determined by a complex chemical process, establish the function of the neural network.

8.11.1 Historical Development of Neural Networks

The historical development of neural networks is as follows:

- **1943 McCulloch and Pitts (Start of the Modern Era of Neural Networks)** McCulloch and Pitts developed a logical calculus of neural networks. A network consists of a sufficient number of neurons (using a simple model), and properly set synaptic connections can compute any computable function. A simple logic function is performed by a neuron, in this case, based upon the weights set in the McCulloch–Pitts neuron. The arrangement of a neuron may be represented as a combination of logic functions. The most important feature of this type of neuron is the concept of threshold. When the net input to the particular

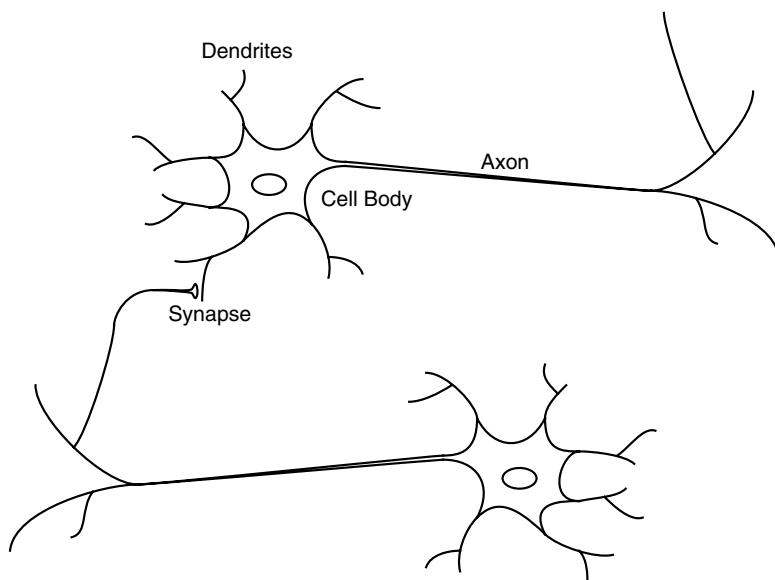


Fig. 8.14 Schematic representation of a biological neuron

neuron is greater than the specified threshold by the user then the neuron fires. Logic circuits extensively use this type of neurons.

- **1949 Donald Hebb Wrote 'The Organisation of Behavior'** An explicit statement of a physiological learning rule for *synaptic modification* was presented for the first time in 1949. Hebb proposed that the connectivity of the brain continually changes as an organism learns differing functional tasks, and that neural assemblies are created by such changes.

Hebb's work was immensely influential among psychologists. The concept behind the Hebb's theory is that if two neurons are found to be active simultaneously then the strength of connection between the two neurons should be increased. This concept is similar to that of the correlation matrix learning.

- **1958 Rosenblatt Introduced Perceptron Block 1962, Minsky and Papert 1988** In a Perceptron network, the weights on the connection paths can be adjusted. A method of iterative weight adjustment can be used in the Perceptron net. The Perceptron net is found to converge if the weights obtained allow the net to reproduce exactly all the training input and target output vector pairs.

- **1960 Widrow and Hoff** Widrow and Hoff developed a network named ADALINE, which is abbreviated from Adaptive Linear Neuron. It uses a learning rule called *least mean square rule* or *delta rule*. This rule is found to adjust the weights so as to reduce the difference between the net input to the output unit and the desired output. The convergence criteria in this case are the reduction of mean square error to a minimum value. This delta rule for a single layer net can be said to be a precursor of the backpropagation net used for multi-layer nets. The multi-layer extensions of adaline formed the Madaline [Widrow and Lehr, 1990].

- **1982 John Hopfield's Networks** Hopfield showed how to use an *Ising spin glass* type of model to store information in dynamically stable networks. His work paved the way for physicists to enter neural modeling, thereby transforming the field of neural networks. These nets are widely used as associative memory nets. The Hopfield nets are found to be both continuous valued and discrete valued. This net provides an efficient solution to the Travelling Salesman Problem.

- **1972 Kohonens Self-organising Maps (SOM)** Kohonens self-organising maps are capable of reproducing important aspects of the structure of biological neural nets and data representation using topographic maps (which are common in the nervous systems). SOM also has a wide range of applications. SOM shows how the output layer can pick up the correlational structure (from the inputs) in the form of the spatial arrangement of units. These nets are applied to many recognition problems.

- **1985 Parker, 1986 Lecum** During this period, the backpropagation net paved its way into structure of the neural networks. This method propagates the error information at the output units back to the hidden units. This uses a generalised delta rule. This net is basically a multi-layer, feed-forward net trained by means of backpropagation. Originally, even though the work was performed by Parker (1985), the credit of publishing this net goes to Rumelhart, Hinton and Williams (1986). It is the most popular learning algorithm for the training of multi-layer perceptrons and has been the workhorse for many neural-network applications.

- **1988 Grossberg** Grossberg developed a learning rule similar to that of Kohonen, which is widely used in the counter-propagation net. Grossberg type of learning is also used as outstar learning. This learning occurs for all units in a particular layer; and no competition among these units is found.

- **1987, 1990 Carpenter and Grossberg** Carpenter and Grossberg invented the Adaptive Resonance Theory (ART). ART was designed for both binary inputs and the continuous-valued inputs. The design for the binary inputs formed ART1 and ART2 and resulted when the design became applicable to the continuous-valued inputs. The most important feature of these nets are that the input patterns can be presented in any order.
- **1988 Broomhead and Lowe** Broomhead and Lowe developed Radial Basis Functions (RBF). This is also a multi-layer net that is quite similar that of the backpropagation net.
- **1990 Vapnik** Vapnik developed the support vector machine.

8.11.2 Artificial Neural Network Terminologies

The common terminologies used in Artificial Neural Networks (ANN) includes (i) weight, (ii) bias, (iii) activation function, (iv) threshold, etc. In order to understand these terminologies, consider a simple neural network illustrated in Fig. 8.15. In the figure x_1, x_2 represents the input, y represents the output, w_1 represents the weight connecting the neuron 1 to the output, and w_2 represents the weight connecting the neuron 2 to the output.

The net input is the summation of the products of the weights and the input signals. The net input is given by

$$\text{net input} = x_1 w_1 + x_2 w_2 \quad (8.14)$$

In general, the net input is given by $\text{net input} = \sum_i x_i w_i$

(i) Weight Weight is basically information used by the neural network to solve a problem. The neurons are connected to each other through each other by directed communication links, which are associated with weights. The weight may be fixed, or it can take random values, or it can be set to zero, or it can be calculated by some methods. Initialisation of weight is an important step in a neural network. The weight changes indicate the overall performance of the neural network. In other words, the weights are adaptive coefficients within the network that determine the intensity of the input signal as registered by the artificial neuron. These strengths can be modified in response to various training sets and according to a network-specific topology or through its learning rules.

(ii) Bias Bias is similar to weight. It acts exactly as a weight on a connection from a unit whose activation is always 1. Increasing the bias increases the net input to the unit. The bias improves the performance of the neural network. Similar to the initialisation of the weights, a bias can be initialised to zero, or to any specified value based on the neural network. A neural network with bias is illustrated in Fig. 8.16.

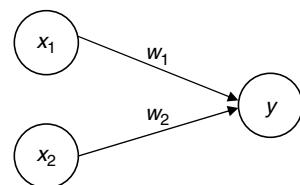


Fig. 8.15 Simple neural network without bias

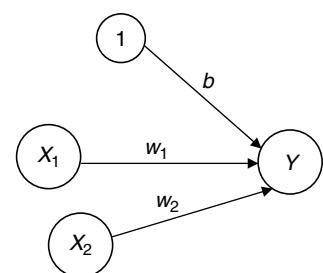


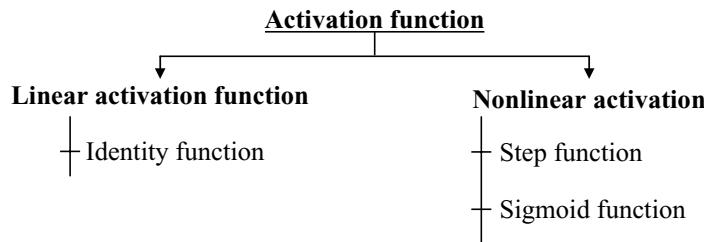
Fig. 8.16 Neural network with bias

For the network shown in Fig. 8.16, the net input with bias is given by

$$\text{Net input} = b + \sum_i x_i w_i \quad (8.15)$$

The first step in a processing-element operation is to compute the weighted sum of all the inputs. The net input signal is the dot or inner product of the input vector with the weight vector.

(iii) Activation Function The activation function is used to calculate the output response of a neuron. The sum of the weighted input signal is applied with an activation to obtain the response. For neurons in the same layer, the same activation functions are used.



Broadly, the activation function can be classified into (i) linear activation function, and (ii) non-linear activation function. An example of a linear activation function is the identity function. The examples of non-linear activation functions include step function and sigmoid function. The step and sigmoid function can be further subdivided into binary and bipolar step and sigmoid functions.

(a) Identity Function The identity function is represented mathematically as

$$f(x) = x, \quad \forall x$$

The graphical representation of the identity function is given in Fig. 8.17. In the case of identity functions, the output is simply proportional to the input; hence their use is limited.

(b) Binary Step Function The mathematical representation of a binary step function is given below:

$$f(x) = \begin{cases} 1 & \text{if } f(x) \geq \theta \\ 0 & \text{if } f(x) < \theta \end{cases}$$

Here, θ indicates the threshold value. The graphical representation of a binary step function is given in Fig. 8.18.

(c) Sigmoidal Function Sigmoidal functions resemble S-shaped curves. Sigmoid functions are widely used in multi-layer networks like backpropagation networks. There are two types of sigmoid functions—(i) binary sigmoid functions, and (ii) bipolar sigmoid functions.

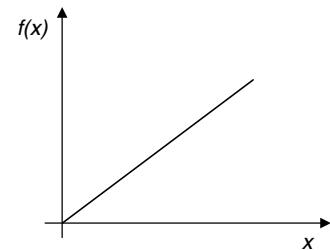


Fig. 8.17 Identity function

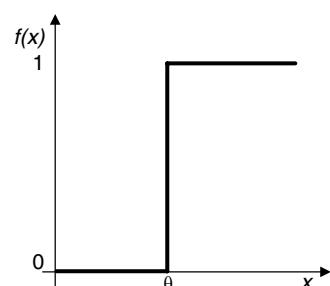


Fig. 8.18 Binary step function

Binary Sigmoid Function A binary sigmoid function is also called a *logistic function*. The value of the function ranges between 0 and 1. The mathematical representation of the binary sigmoid function is given by

$$f_{\text{binary}}(x) = \frac{1}{1 + \exp(-\sigma x)} \quad (8.16)$$

where σ represents the steepness factor. The graphical representation of a binary sigmoid function for different steepness factors is given in Fig. 8.19.

Bipolar Sigmoid Function The bipolar sigmoid function is related to the hyperbolic tangent function. The desired range of the function is between -1 and $+1$. The mathematical representation of the bipolar sigmoid function is given by

$$f_{\text{bipolar}}(x) = \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)} \quad (8.17)$$

The graphical representation of the bipolar sigmoid function is given in Fig. 8.20.

(iv) Threshold The threshold θ is a factor which is used in calculating the activations of the given net. Based on the value of threshold, the output may be calculated, i.e., the activation function is based on the value of θ . For example, the activation functions may be

$$y = f(\text{net}) = \begin{cases} +1 & \text{if net} \geq \theta; \\ -1 & \text{if net} < \theta; \end{cases}$$

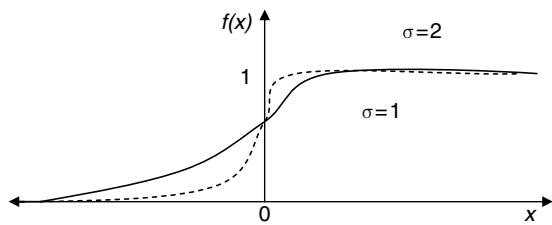


Fig. 8.19 Binary sigmoid function

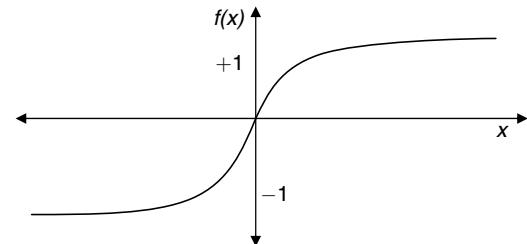


Fig. 8.20 Bipolar sigmoid function

8.12 CLASSIFICATION OF ARTIFICIAL NEURAL NETWORKS

A neural network is characterised by (i) its pattern of connections between the neurons which is generally termed architecture, (ii) method of determining the weights on the connections which is generally termed training or learning algorithm, and (iii) its activation function.

8.12.1 Classification of Neural Networks Based on Architecture

Based on architecture, artificial neural networks can be broadly classified into single layer and multi-layer networks as illustrated in Fig. 8.21.

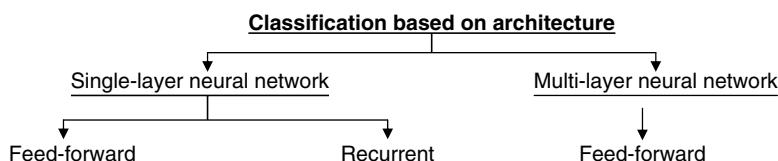


Fig. 8.21 Classification of ANN based on architecture

Single-layer Neural Network A single-layer neural network has one layer of connection weights. The input unit receives signals from the outside world. The output of the neural net can be read from the output unit. For pattern classification, each output unit corresponds to a particular category to which an input vector may or may not belong. In the case of a single-layer neural network, the weights for one output unit do not influence the weights for other output units. Examples of single-layer neural networks are (i) McCulloch–Pitts Neuron, (ii) Perceptron, and (iii) ADALINE.

The single-layer neural networks are mostly feed-forward networks. In the case of a feed-forward network, the signals flow from the input units to the output units in a forward direction. The single-layer neural network is illustrated in Fig. 8.22.

Competitive Network The competitive network is similar to a single-layer feed-forward network except that there are connections, usually negative, between the output nodes. Because of these connections, the output node tends to compete to represent the current input pattern. Sometimes the output layer is completely connected and sometimes the connections are restricted to units that are close to each other. MAXNET is a specific example of a neural network based on competition. There is no training algorithm for MAXNET as its weights are fixed. The architecture of MAXNET is given in Fig. 8.23.

Recurrent Network Recurrent networks are fully connected competitive networks in which there are closed-loop signal paths from a unit back to itself. Examples of recurrent networks are the Hopfield network and the Bidirectional Associative Memory (BAM) network.

Multi-layer Neural Network A multi-layer neural network will have more layers of nodes between the input and output units, which are generally termed *hidden units*. A Multi-layer neural network can solve more complicated problems than a single-layer neural network, but training of multi-layer networks is difficult. Multi-layer neural networks can be either feed-forward or feedback networks. An example of a feed-forward multilayer network is the MADALINE network. MADALINE consists of Many Adaptive Linear Neurons arranged in a multi-layer network. MADALINE with two hidden units is illustrated in Fig. 8.24.

From Fig. 8.24, it is obvious that the MADALINE network has two input units X_1 and X_2 , two hidden units Z_1 and Z_2 and an output unit Y . In the network, b_1 , b_2 and b_3 represent the bias.

An example of a feed-forward multi-layered network in which the errors are calculated and fed back to adjust the weight is the backpropagation network.

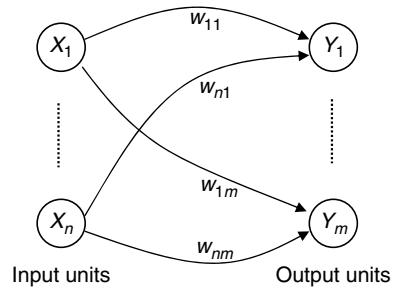


Fig. 8.22 Single-layer neural network

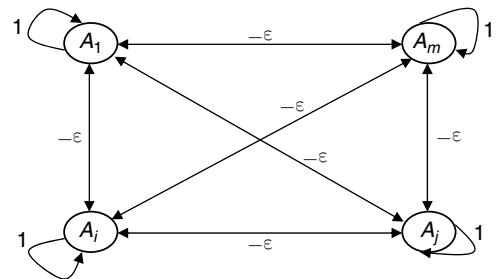


Fig. 8.23 MAXNET

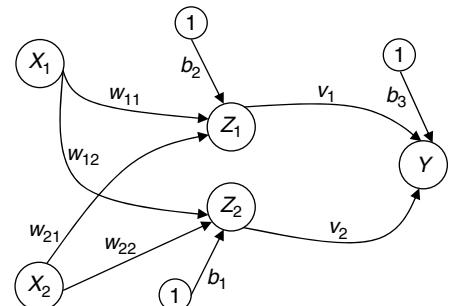


Fig. 8.24 MADALINE network

8.12.2 Classification of Neural Networks Based on the Training Algorithm

Based on the training algorithm, the neural network can be broadly classified into three types: (i) supervised learning, (ii) unsupervised learning, and (iii) reinforcement learning.

Supervised Learning Supervised learning is the process of providing the network with a series of sample inputs and comparing the output with the expected responses. The learning continues until the network is able to provide the expected response. The learning is considered complete when the neural network reaches a user-defined performance level. This level signifies that the network has achieved the desired accuracy as it produces the required outputs for a given sequence of inputs. When no further learning is necessary, the weights are typically frozen for the application. The supervised learning is also known as associative learning. In the associative learning, the network is trained by providing it with input and matching output patterns. These input–output pairs can be provided by an external teacher, or by the system which contains the network. Some of the supervised learning algorithms include the Hebb network, pattern association network, backpropagation network (BPN), associative memory network and counter-propagation network (CPN).

Unsupervised Learning The unsupervised learning approach is also termed self-organising. In the case of unsupervised learning, the target output is not known, and only a stimulus is provided to the unsupervised networks. Unsupervised networks, in general, will not use external influences to adjust their weights. Instead they internally monitor their performance. Unsupervised networks look for regularities or trends in the input signals, and make adaptations according to the function of the network. Unsupervised networks are far more complex and difficult to implement. Examples of unsupervised networks are Kohonen Self-Organising Feature Map and Adaptive Resonance Theory (ART).

Reinforcement Learning In the reinforcement learning method, a teacher is assumed to be present, but the right answer is not presented to the network. Instead, the network is only presented with an indication of whether the output answer is right or wrong. The network must then use this information to improve its performance. Reinforcement learning is a very general approach to learning that can be applied when the knowledge required to apply supervised learning is not available. If sufficient information is available, the reinforcement learning can readily handle a specific problem. In reinforcement learning, the classes are not known before training. Positive feedback is given for correct action, and negative for incorrect.

8.13 LEARNING RULES IN ARTIFICIAL NEURAL NETWORKS

A learning rule is basically a procedure for modifying the weights and biases of a network. The purpose of the learning rule is to train the network to perform some task. Research into different learning functions continues as new ideas routinely show up in trade publications. Some researchers have the modeling of biological learning as their main objective. Others are experimenting with adaptations of their perceptions of how nature handles learning. The essence of different learning rules employed in the area of neural networks is given below:

(i) Hebb Rule The Hebb rule was introduced by Donald Hebb in 1949. According to the Hebb rule, if a neuron receives an input from another neuron, and if both are highly active, the weight between the neurons should be strengthened.

(ii) Hopfield Rule The Hopfield rule is similar to the Hebb rule with the exception that it specifies the magnitude of the strengthening or weakening. According to the Hopfield rule, if the desired output and

the input are both active, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate.

(iii) Delta Rule The delta rule is based on the idea of continuously modifying the strengths of the input connections to reduce the difference between the desired output value and the actual output of a processing element. This rule changes the synaptic weights in the way that minimises the mean squared error of the network. The delta rule is also known as the Widrow–Hoff Learning Rule and the Least Mean Square (LMS) Learning Rule. In the delta rule, the delta error in the output layer is transformed by the derivative of the transfer function and is then used in the previous neural layer to adjust the input connection weights. In other words, this error is backpropagated into previous layers in one layer at a time. The process of backpropagating the network errors continues until the first layer is reached. When using the delta rule, it is important to ensure that the input data set is well randomised. A well-ordered presentation of the training set will not result in the convergence of the network. If that happens then the network is incapable of learning the problem.

(iv) Gradient Descent Rule The gradient descent rule is similar to the delta rule with respect to the fact that the derivative of the transfer function is used to modify the delta error before it is applied to the connection weights. In the gradient descent rule, an additional proportional constant tied to the learning rate is appended to the final modifying factor acting upon the weight.

(v) Kohonens Learning Rule In the Kohonens learning rule, the processing elements compete for the opportunity to learn, or update their weights. The processing element with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbours. Only the winner is permitted an output, and only the winner plus its neighbours are allowed to adjust their connection weights. The size of the network can vary during the training period. The usual approach is to start with the larger definition of the neighbourhood, and narrow in as the training process proceeds. The Kohonens rule is good for statistical or topological modeling of the data and is referred as self-organising maps or self-organising topologies.

8.14 PERCEPTRON

Together with several other researchers, Frank Rosenblatt introduced and developed a large class of artificial neural networks called perceptrons. A perceptron uses an iterative update algorithm to learn a correct set of weights. The architecture of a single-layer perceptron is illustrated in Fig. 8.25.

In the architecture shown in the Fig. 8.25, X_1, X_2, \dots, X_n represents the input; w_1, w_2, \dots, w_n represents the weights from the input to the output and b represents the bias. The output Y in the simplest case is a weighted sum of the inputs which is given by

$$Y_m = b + \sum_{i=1}^n X_i w_i \quad (8.18)$$

The output is compared with the target, and if there is a difference between the obtained output and the desired target, weight updation is done. The weight updation is based on the perceptron learning rule.

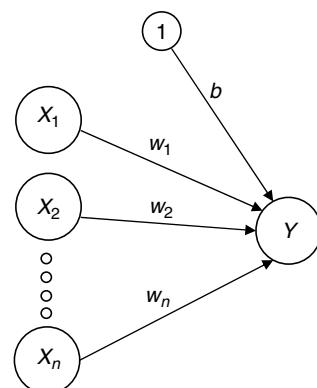


Fig. 8.25 Architecture of a single-layer perceptron

8.14.1 Perceptron Training Algorithm

The perceptron training algorithm is given as follows:

Step 1 Initialise the weights and bias. Set the learning rate α . The value of α is between 0 and 1.

Step 2 While the stopping condition is false, do steps 3 to 7.

Step 3 For each training pair $s:t$, do steps 4 to 6.

Step 4 Set activation of input units:

$$x_i = s_j \quad \text{for } i = 1 \text{ to } n$$

Step 5 Compute the output unit response:

$$Y_{in} = b + \sum_{i=1}^n X_i w_i$$

The activation function is given by

$$Y = \begin{cases} 1, & \text{if } Y_{in} > \theta \\ 0, & \text{if } -\theta \leq Y_{in} \leq \theta \\ -1, & \text{if } Y_{in} - \theta \end{cases}$$

Step 6 The weights and bias are updated if the target is not equal to the output response. If $t \neq y$ and the value of x_i is not zero then the weight and bias updation is given by

$$\begin{aligned} w_{i(\text{new})} &= w_{i(\text{old})} + \alpha t x_i \\ b_{(\text{new})} &= b_{(\text{old})} + \alpha t \end{aligned}$$

Else,

$$\begin{aligned} w_{i(\text{new})} &= w_{i(\text{old})} \\ b_{(\text{new})} &= b_{(\text{old})} \end{aligned}$$

Step 7 Test for the stopping condition.

It is to be noted that the weights are updated only for patterns that do not produce the correct output. The limitation of a simple perceptron is that the input must be a linearly separable pattern.

8.15 MULTI-LAYER PERCEPTRON

A perceptron that has a single layer of weights can only approximate linear functions of the input and cannot solve problems like the XOR, where the discriminant to be estimated is non-linear. This limitation does not apply to feed-forward networks with intermediate or hidden layers between the input and output layers. A multi-layer perceptron can implement non-linear discriminants and, if used for regression, can approximate non-linear functions of the input.

8.15.1 Backpropagation Network

A Backpropagation Network (BPN) consists of at least three layers of units—an input layer, at least one hidden layer, and an output layer. Typically, units are connected in a feed-forward fashion with input units fully connected to units in the hidden layer, and hidden units fully connected to units in the output layer. G E Hinton, Rumelhart and R O Williams first introduced the backpropagation network in 1986. The basic concept for this weight update algorithm is simply the gradient-descent method as used in the case of simple perceptron networks with differentiable units. In this method the error is propagated back to the hidden unit. When the hidden layers are increased, the network training becomes more complex. The training of the backpropagation network is done in three stages—(i) Feed-forward of the input training pattern, (ii) calculation and backpropagation of the error, and (iii) updation of weights.

8.15.2 Architecture of BPN

A backpropagation neural network is a multi-layer, feed-forward neural network consisting of an input layer, a hidden layer and an output layer. The neurons present in the hidden and output layers have biases, which are the connections from the units whose activation is always one. Figure 8.26 depicts the architecture of a BPN.

The inputs sent to the BPN and the output obtained from the net could be either binary or bipolar. The activation function could be any function which increases monotonically and is also differentiable.

8.15.3 BPN Training Algorithm

The terminologies used in the algorithm are summarised below:

x = input training vector

t = target output vector

α = learning rate

v_{0j} = bias on j^{th} hidden unit

w_{0k} = bias on k^{th} output unit

z_j = hidden unit j . The net input to z_j is

$$z_{inj} = v_{0j} + \sum_i x_i v_{ij}$$

The output of the hidden unit is represented by

$$z_j = f(z_{inj})$$

The net input to the output unit is represented by

$$y_{ink} = w_{0k} + \sum_j z_j w_{jk}$$

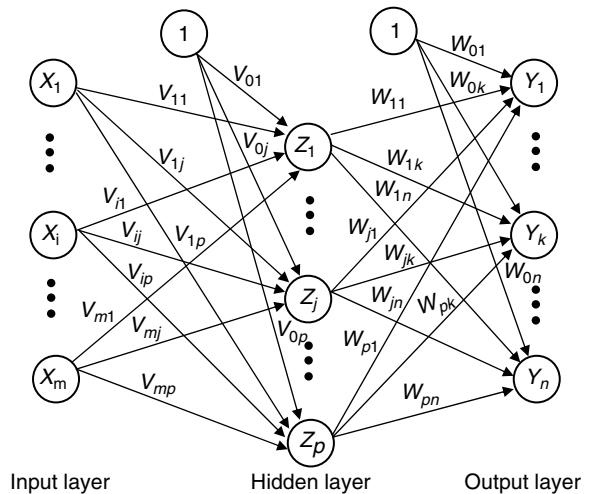


Fig 8.26 Architecture of a BPN network

The output is given by

$$y_k = f(y_{ink})$$

δ_k represents the error-correction weight adjustment for w_{jk} that is due to an error at output unit y_k , which is backpropagated to the hidden units that feed into unit y_k . Also δ_j represents the error-correction weight adjustment for v_{ij} that is due to the backpropagation of the error to the hidden unit z_j . The commonly used activation functions are binary or bipolar sigmoidal activation functions.

The different steps involved in the training algorithm of a BPN network is given below:

Step 1 Initialise the weights and learning rate.

Step 2 Perform steps 3 to 10 until the stopping condition is false.

Step 3 Perform steps 4 to 9 for each training pair.

Feed Forward Phase (Phase I)

Step 4 Each input unit receives the input signal x_i and sends it to the hidden unit.

Step 5 Each hidden unit z_j sums its weighted input signals to calculate the net input which is given below:

$$z_{inj} = v_{oj} + \sum_i x_i v_{ij}$$

Calculate the output of the hidden unit by applying its activation functions over z_{inj} .

$$z_j = f(z_{inj})$$

The output signal from the hidden unit is send to the output unit.

Step 6 For each output unit y_k , calculate the net input.

$$y_{ink} = w_{ok} + \sum_j z_j w_{jk}$$

Apply the activation function to compute the output signal which is given by

$$y_k = f(y_{ink})$$

Backpropagation of Error (Phase II)

Step 7 Each output unit y_k receives a target pattern corresponding to the input training pattern and computes the error-correction term:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

On the basis of the calculated error-correction term, update the change in weights and bias:

$$\Delta w_{jk} = \alpha \delta_k z_j$$

$$\Delta w_{0k} = \alpha \delta_k$$

Also, send δ_k to the hidden layer backwards.

Step 8 Each hidden unit z_j sums its delta inputs from the output units:

$$\delta_{inj} = \sum_{k=1}^m \delta_k w_{jk}$$

The term δ_{inj} gets multiplied with the derivative of $f(z_{inj})$ to calculate the error term:

$$\delta_j = \delta_{inj} f'(z_{inj})$$

On the basis of the calculated δ_j , update the change in weights and bias:

$$\Delta v_{ij} = \alpha \delta_j x_i$$

$$\Delta v_{0j} = \alpha \delta_j$$

Weight and Bias Updation (Phase III)

Step 9 Each output unit y_k updates the bias and weights:

$$w_{jk} (\text{new}) = w_{jk} (\text{old}) + \Delta w_{jk}$$

$$w_{0k} (\text{new}) = w_{0k} (\text{old}) + \Delta w_{0k}$$

Each hidden unit updates its bias and weights:

$$v_{ij} (\text{new}) = v_{ij} (\text{old}) + \Delta v_{ij}$$

$$v_{0j} (\text{new}) = v_{0j} (\text{old}) + \Delta v_{0j}$$

Step 10 Check for the stopping condition. The stopping condition may be a certain number of epochs reached or when the actual output equals the target output.

8.15.4 Momentum Factor in BPN

The gradient descent is very slow if the learning rate α is small and oscillates widely if α is too large. The commonly used method that allows a larger learning rate without oscillations is one which adds a momentum

factor to the normal gradient-descent method. The momentum factor is denoted by μ . The weight updation formula using the momentum factor is given by

$$w_{jk}(t+1) = w_{jk}(t) + \alpha \delta_k z_j + \mu [w_{jk}(t) - w_{jk}(t-1)] \quad (8.19)$$

The above equation can be represented as

$$\Delta w_{jk}(t+1) = \alpha \delta_k z_j + \mu \Delta w_{jk}(t)$$

Also,

$$v_{ij}(t+1) = v_{ij}(t) + \alpha \delta_j x_i + \mu [v_{ij}(t) - v_{ij}(t-1)] \quad (8.20)$$

The above equation can be represented as

$$\Delta v_{ij}(t+1) = \alpha \delta_j x_i + \mu \Delta v_{ij}(t) \quad (8.21)$$

The momentum parameter μ is constrained to be in the range from 0 to 1, exclusive of the end points. The momentum allows the net to make reasonably large weight adjustments as long as the corrections are in the same general directions for several patterns, while using a smaller learning rate to prevent a large response to the error from any one training pattern. When using momentum, the net proceeds not in the direction of the gradient, but in the direction of a combination of the current gradient and the previous direction of weight correction.

8.16 RADIAL-BASIS FUNCTION NETWORK

Radial functions are a special class of functions. A radial function response decreases or increases monotonically with distance from a central point. A typical radial function is the Gaussian. The Gaussian function in the case of scalar input is given by

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right) \quad (8.22)$$

where c stands for the centre and r for radius. The Gaussian radial-basis function with centre $c = 0$ and radius $r = 1$ is illustrated in Fig. 8.27.

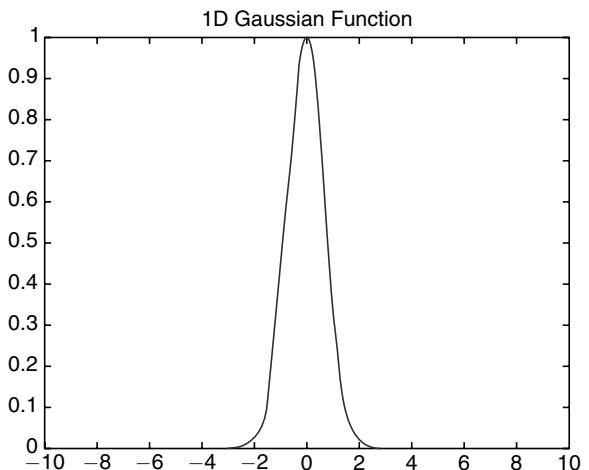


Fig. 8.27 1D Gaussian function

8.17 ADVANTAGES OF NEURAL NETWORKS

The benefits of neural networks in pattern recognition are summarised below.

1. Pattern recognition is a powerful technique for harnessing information in the data and generalising it. Neural networks learn to recognise the patterns which exist in a given data set.

2. Neural networks are flexible in a changing environment, while rule-based systems or programmed systems are limited to the situation for which they are designed. When conditions change, rule-based systems are no longer valid. Neural networks take some time to learn a sudden change, but they are capable of adapting to changing information.
3. A neural-network-based object-recognition system is developed through learning rather than programming. Programming is much more time consuming for an analyst and requires the analyst to specify the exact behavior of the model. Neural networks teach themselves the patterns in the data.
4. Neural networks can build informative models where most of the conventional approaches fail. The advantage of neural networks is that they can handle very complex interactions so that informative models can be easily built.

8.18 STRUCTURAL PATTERN RECOGNITION

Decision-theoretic methods rely on quantitative measurements. The quantitative nature of statistical pattern recognition, however, makes it difficult to discriminate among groups based on the morphological subpatterns and their inter-relationships embedded within the data. This limitation provides the impetus for the development of a structural approach to pattern or object recognition. Structural pattern recognition, sometimes referred as *syntactic pattern recognition* due to its origins in formal language theory, relies on syntactic grammars to discriminate among different groups based upon morphological interrelationships present within the data. Structural features, often referred as primitives, represent the subpatterns and the relationships among them which constitute the data. The semantics associated with each feature are determined by the selection of morphologies used to identify primitives in the data. The emphasis on relationship within data makes a structural approach to pattern recognition most sensible for data which contain an inherent, identifiable organisation such as image data, time series data, etc. Structural pattern recognition assumes that pattern structure is quantifiable and extractable so that structural similarity of patterns can be accessed. Methodologies used to extract structural features from image data such as morphological image-processing techniques result in primitives such as edges, curves and regions. The classification task arrives at an identification using parsing.

8.18.1 Statistical and Structural Approaches

In order to understand the difference between the statistical and structural approaches, consider a simple problem, the objective which is to distinguish between squares and triangles. The statistical and structural approaches for this problem are given in Fig. 8.28.

From Fig. 8.28, it is obvious that the statistical approach extracts quantitative features such as the number of horizontal, vertical, and diagonal segments which are then used to classify the object. A structural approach extracts morphological features and their inter-relationships within each figure. Using a straight-line segment as the elemental morphology, a relational graph is generated and classified by determining the syntactic grammar that can successfully parse the relational graph.

Grammars and Languages Primitives are elementary properties of the syntactically described objects. Once the primitives are extracted, the inter-primitive relations can be described relations which form structures like chains, trees and graphs, called *words*. Each pattern or object is described by a word. The set of words form the language which is described by a grammar. The grammar is described by a quadruple

$$G = [A_n, A_t, P, S] \quad (8.23)$$

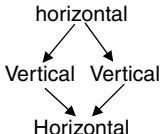
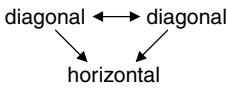
Object	Statistical approach	Structural approach
	Number of segments = 4 Number of horizontal segments = 2 Number of vertical segments = 2 Number of diagonal segments = 0	
	Number of segments = 3 Number of horizontal segments = 1 Number of vertical segments = 0 Number of diagonal segments = 2	

Fig. 8.28 Statistical and structural approach

where A_n and A_t are disjoint alphabets, elements of A_n are called non-terminal symbols, and elements of A_t are called terminal symbols. P is the finite set of production rules. Production rules are the most critical parts of the grammar. String grammars are characterised primarily by the form of their productions. The symbol S is the grammar axiom, or the start symbol. In a nutshell, the patterns are viewed as sentences belonging to a language, primitives are viewed as the alphabet of the language, and the sentences are generated according to a grammar. Thus, a large collection of complex patterns can be described by a small number of primitives and grammatical rules. The grammar for each pattern class must be inferred from the available training samples. Structural pattern recognition is intuitively appealing because, in addition to classification, this approach also provides a description of how the given pattern is constructed from the primitives. Let S represent the start symbol. S is replaced by the string on the right side of a chosen production rule. The process of rewriting a substring according to one of the production rules continues until the string consists only of terminal symbols.

$$S \rightarrow aS \mid bS \mid \epsilon$$

The symbol \mid indicates OR, and ϵ indicates the null string. The succession of strings that result from the process is a derivation from the grammar.

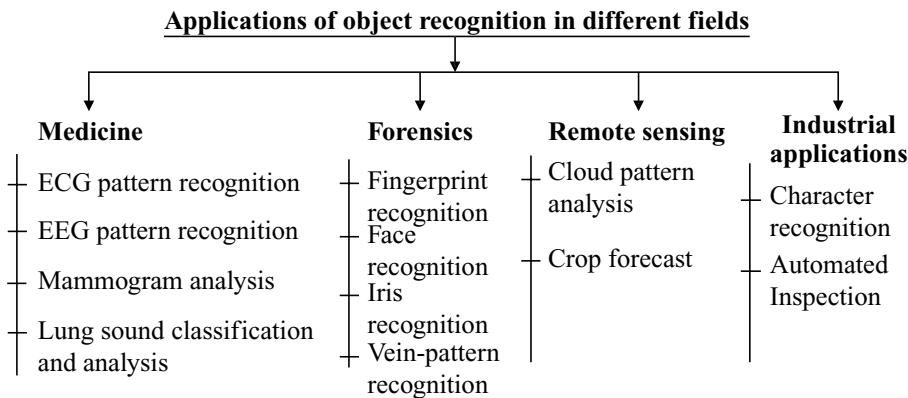
Types of Grammars According to Chomsky, grammars are classified into four categories which are

- (i) Unrestricted grammars (0-type grammars)
- (ii) Context-sensitive grammars (1-type grammars)
- (iii) Context-free grammars (2-type grammars) and
- (iv) Regular grammars or finite-state grammars (3-type grammars).

The most general grammar is obviously the 0-type, which bears no limits for rewriting rules. Types 0 and 1 are able to describe natural languages. In the 1-type grammar, for rewriting rules, the right side of a rule should have at least as many symbols as the left side. For the 2-type grammar, all rules should have just one non-terminal symbol on the left side. For the 3-type grammar, the right side has only one terminal symbol, or one terminal symbol and a non-terminal one for every production rule. A grammar normally divides strings in just two classes—the grammatically correct ones, and the others.

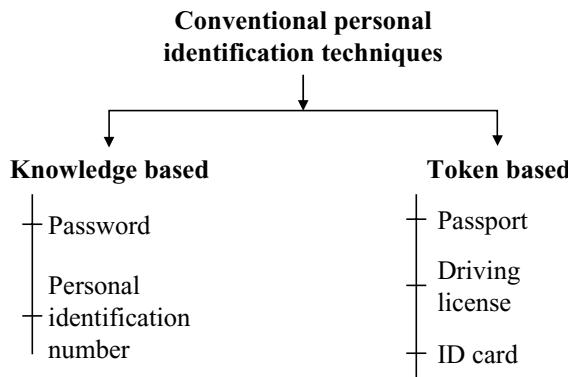
8.19 APPLICATIONS OF OBJECT RECOGNITION

Some of the fields in which object recognition is widely used are (i) medicine, (ii) forensics, (iii) remote sensing, and (iv) industry.



Personal identification is discussed in detail in this section. In personal identification, fingerprint face recognition are taken into account for discussion.

Accurate automatic personal identification is needed in different applications involving the use of passports, cellular telephones, automatic teller machines and driver licenses. The conventional personal identification techniques can be broadly classified into two types—(i) knowledge based, and (ii) token based.



Traditional knowledge-based identifications are prone to fraud because passwords may be guessed by an imposter and the tokens may be lost or stolen. Therefore, traditional knowledge-based and token-based approaches are unable to satisfy the requirements of an electronically interconnected information society.

Biometrics has been widely used in forensic applications such as criminal identification and prison security. Biometric technology is rapidly evolving and has a very strong potential to be widely adopted in civilian applications such as electronic banking, e-commerce and access control. With the progress in biometric technology, these applications will increasingly use biometrics for authentication. The biometrics normally used for personal identification are (i) fingerprint recognition, (ii) iris recognition, (iii) vein-pattern recognition.

8.19.1 Fingerprint Recognition

Fingerprints are the ridge-and-furrow patterns on the tip of a finger. Fingerprints are extensively used for personal identification of people. Fingerprint-based authentication is becoming more and more popular in a number of civilian and commercial applications such as cellular phone access and laptop computer log-in. The availability of cheap and compact solid scanners as well as robust fingerprint matchers are two important factors in the popularity of fingerprint-based identification systems. Fingerprint ridge configurations do not change throughout the life of an individual except due to accidents and cuts on the fingertips. This fact makes fingerprints a very attractive biometric identifier.

Fingerprint Representation The global representation schemes of fingerprints used for classification can be broadly categorised into four main categories:

- (i) Knowledge based
- (ii) Structure based
- (iii) Frequency based
- (iv) Syntactic

A knowledge-based approach tries to capture the knowledge of a human expert by deriving rules for each category by hand-constructing the models. A structure-based approach uses the estimated orientation field in a fingerprint image. A frequency-based approach uses the frequency spectrum of the fingerprints for representation. Hybrid approaches combine two or more approaches for representation.

Fingerprint Classification Fingerprint classification is a technique used to assign a fingerprint into one of the several pre-specified types, which can provide an indexing mechanism. Fingerprint classification can be viewed as a coarse-level matching of the fingerprints. The five different classes of fingerprints identified by the National Institute of Standards and Technology (NIST) to benchmark automatic fingerprint classification are whorl (W), left loop (L), right loop (R), arch (A) and, tented arch (T), as illustrated in Fig. 8.29.

8.19.2 Face Recognition

Faces are diverse, semi-rigid, semi-flexible, culturally significant, and part of our individual entity. There are many approaches to the face-detection problem. Some techniques rely on a single face template or a model for detection; others rely on facial sub-features. A variety of detection techniques are employed from correlation, neural networks, eigen templates, Bayesian models and flexible models. Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face.

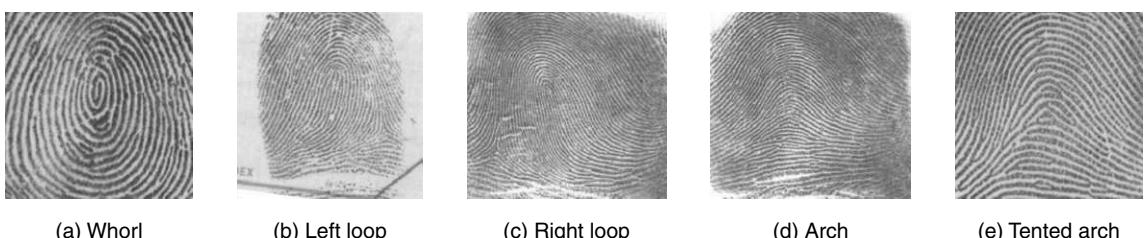


Fig. 8.29 Different classes of fingerprints

Detecting Faces in a Single Image The single-image detection can be broadly classified into four categories which are (i) Knowledge-based methods, (ii) feature-invariant approaches, (iii) template matching methods, and (iv) appearance-based methods

Knowledge-based methods are basically rule-based methods and are mainly designed for face localisation. In knowledge-based methods, the rules capture the relationships between facial features. The rules are derived from a researcher's knowledge of human faces. For example, a face often appears in an image with two eyes that are symmetric to each other, a nose and a mouth. The relationship between features can be represented by their relative distances and positions. Facial features in an input image are extracted first, and face candidates are identified based on the derived rules. A verification process is usually applied to reduce false detection.

In *feature-based face-detection methods*, facial features such as eyebrows, eyes, nose, mouth and hair-line are commonly extracted using edge detectors. Based on the extracted features, a statistical model is built to describe their relationships and to verify the existence of a face. Human skin colour has been used and proven to be an effective feature in many applications. Although different people have different skin colours, the major difference lies largely between their intensity rather than their chrominance.

In *template matching*, a standard face pattern is manually predefined, or parameterised by a function. Given an input image, the correlation values with the standard patterns are computed for the face contour, eyes, nose and mouth independently. The existence of a face is determined based on the correlation values. The face model can be built in terms of features defined by the edges. These features describe the curvatures of the left side, the hairline, and the right side of a frontal face.

Appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and non-face images. The learned characteristics are in the form of distribution models or discriminant functions that are consequently used for face detection. An image or feature vector derived from an image is viewed as a random variable x , and this random variable is characterised for faces and non-faces by the class-conditional density functions $p(x/\text{face})$ and $p(x/\text{non-face})$. Bayesian classification or maximum likelihood can be used to classify a candidate image location as a face or non-face.

Eigen Faces Face description can be obtained by computing the eigen vectors of the image's autocorrelation matrix. These eigen vectors are known as eigen faces. The task of facial recognition is discriminating the image data into several classes. The input signals are highly noisy, yet the input images are not completely random and in spite of their differences there are patterns such as eyes, nose, mouth, etc., present in the facial image. These patterns are separated by specific distances. These characteristic features are called eigen faces. The original face can be reconstructed by combining all the eigen faces in the right proportion. The eigen vector of the covariance matrix forms the basis of the KL transform which is also known as the Principal Component Analysis (PCA) or the Hotelling transform. PCA generates a set of orthogonal axes of projections known as the principal components, or the eigen vectors, of the input data distribution in the order of decreasing variance. By means of PCA, one can transform each original image into a corresponding eigen face. In order to determine whether the object in the images is the desired face or not, extract the weights from the eigen face and the face to be recognised. Similar images possess similar eigen faces, and hence similar weights. Thus, all images having similar weights are likely to be similar faces.

Neural Network Approach to Face Detection Neural networks have been applied successfully for face detection. The advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class-conditional density of face patterns. The number of layers, number of nodes and learning rates of the network has to be extensively tuned to get proper face recognition. The neural-network architectures

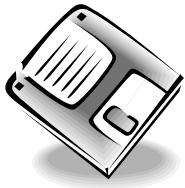
employed by researchers in the field of face recognition include (i) Kohonen Self Organising Feature Map (SOFM), (ii) auto-associative neural network, and (iii) probabilistic decision-based neural network.

Challenges in Face Recognition

The challenges associated with face detection are summarised below.

1. *Facial Expression* The appearance of faces are affected by a person's facial expression.
2. *Image Orientation* Face images directly vary for different rotations about the camera's optical axis.
3. *Imaging Conditions* When the image is formed, factors such as lighting and camera characteristics affect the appearance of the face. Due to different lighting conditions, the intensity of a facial image will change. The camera characteristics include sensor response, lenses, etc.
4. *Pose* The images of a face vary due to the relative camera-face pose, and some facial features such as an eye or nose becoming partially or wholly occluded.
5. *Presence of Glass* Face detection would be difficult in the presence of glasses or a variety of hair styles.

Summary



- Patterns are basically regularity in the input data. Patterns can be captured either in geometric or statistical terms. Object recognition deals with the recognition of the patterns in an image.
- Patterns can be represented in one of the following forms:
 - (a) vector form, (b) string form, and (c) tree representation
- Different approaches to pattern recognition include
 - (a) statistical approach, (b) structural approach, and (c) neural network approach
- The bayesian approach is a popular statistical approach to object recognition. In statistical approach each pattern is represented as a measurement and is viewed as a point in the d -dimensional space. Given a set of training patterns from each class, the objective is to establish decision boundaries in the feature space which separates patterns belonging to different classes.
- One of the simplest approaches to pattern recognition is template matching. In template matching, the correlation between the input image and target is taken into account.
- In the neural-network approach, if the target is known then it comes under the category of supervised learning, and if the target is unknown then it comes under the category of unsupervised learning.
- The multi-layer perceptron is found to be more effective in object recognition than the single-layer perceptron.
- In structural approach, the structural features, commonly known as primitives, are assumed to be known. Structural pattern recognition assumes that pattern structure is quantifiable and extractable so that structural similarity of patterns can be accessed.
- Pattern recognition finds application in different areas like industrial automation, document image analysis, biometric recognition, and remote sensing.

Review Questions

1. You are given set of data $S = \{\text{dolphin, Pomeranian dog, humming bird, frog, rattlesnake, bat}\}$. Develop a suitable classification strategy to classify the given set S according to their nature of existence.

The classification strategy to classify the data set S is shown in Fig. 8.30.

2. Consider a two-class problem, which classifies the given set of flowers as either Rose or Jasmine. Two features as illustrated in Fig. 8.31 are defined for this problem. In the following plot, each J denotes Jasmine and each R denotes Rose. Which feature has a better discrimination power? Explain.

Feature 2 has a better discrimination power than Feature 1. Feature 1 cannot clearly distinguish between Rose and Jasmine whereas Feature 2 can effectively segregate Rose and Jasmine.

3. There are two classes of integrated circuits (IC), namely, analog integrated circuits and digital integrated circuits. Give three salient features which will be effective to classify any IC into analog or digital.

One feature that distinguishes analog and digital ICs is the level of input and output. The digital IC expects binary input and produces binary output, whereas in an analog IC, the input and output take continuous values.

4. Verify that $K = \begin{bmatrix} 37 & -15 \\ -15 & 37 \end{bmatrix}$ is a valid covariance matrix.

A matrix is a valid covariance matrix if it satisfies the following criterion:

- (i) The matrix should be symmetric.
- (ii) The matrix should be positive definite which means that all its eigen values are non-negative.

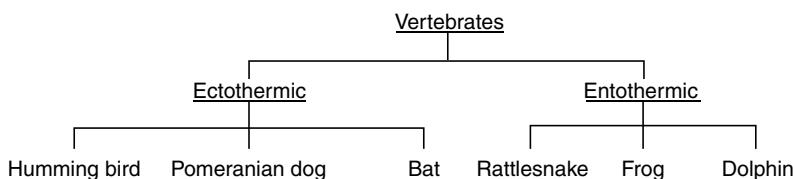


Fig. 8.30 Classification strategy for the data set S

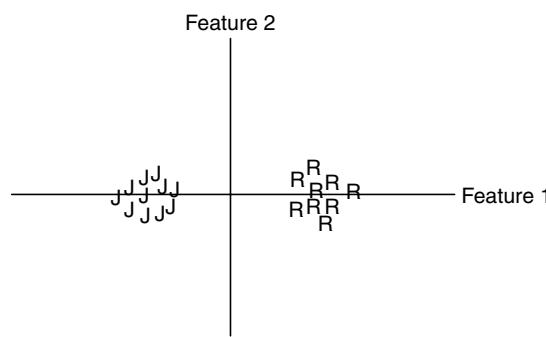


Fig. 8.31 Two-class problem

By looking into the matrix K it is obvious that the matrix is symmetric, and hence the first condition is satisfied. Next, find the eigen values of the matrix K . The eigen values of the matrix are obtained by

$$|k - \lambda I| = 0$$

$$\begin{bmatrix} 37 & -15 \\ -15 & 37 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 37 - \lambda & -15 \\ -15 & 37 - \lambda \end{bmatrix} = 0$$

$$(37 - \lambda)^2 - 15^2 = 0$$

The eigen values are found to be $\lambda_1 = 52$, and $\lambda_2 = 22$. Both the eigen values are non-negative, and hence the given matrix is a covariance matrix.

- 5. Compute the covariance matrix of the data given by $X_1 = [2 \ 1]^T$, $X_2 = [3 \ 2]^T$, $X_3 = [2 \ 3]^T$ and $X_4 = [1 \ 2]^T$.**

Step 1 To form the matrix X

$$X = [x_1; x_2; x_3; x_4]$$

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

Step 2 To find the mean value of the vector X ,

$$\bar{X} = [1.5 \ 2.5 \ 2.5 \ 1.5]$$

Step 3 To find $(X - \bar{X})$

$$(X - \bar{X}) = \begin{bmatrix} 0.5 & 0.5 & -0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \end{bmatrix}$$

Step 4 To find $(X - \bar{X})^T$

$$(X - \bar{X})^T = \begin{bmatrix} 0.5 & -0.5 \\ 0.5 & -0.5 \\ -0.5 & 0.5 \\ -0.5 & 0.5 \end{bmatrix}$$

Step 5 The covariance matrix of X is K

$$K = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^T (X_i - \bar{X})$$

where n is the row of the given matrix X .

Here, $n = 2$, hence

$$K = \sum_{i=1}^2 (X_i - \bar{X})^T (X_i - \bar{X}) = (X_1 - \bar{X})^T (X_1 - \bar{X}) + (X_2 - \bar{X})^T (X_2 - \bar{X})$$

$$= \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} [0.5 \ 0.5 \ -0.5 \ -0.5] + \begin{bmatrix} -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \end{bmatrix} [-0.5 \ -0.5 \ 0.5 \ 0.5]$$

$$= \begin{bmatrix} 0.25 & 0.25 & -0.25 & -0.25 \\ 0.25 & 0.25 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.25 & 0.25 \\ -0.25 & -0.25 & 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0.25 & 0.25 & -0.25 & -0.25 \\ 0.25 & 0.25 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.25 & 0.25 \\ -0.25 & -0.25 & 0.25 & 0.25 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \end{bmatrix}.$$

6. The common terms used in statistical pattern recognition are *model*, *estimation*, *regression*, *parameters*, *independent variables*, and *dependent variables*. What could be the equivalent terms in a neural-network-based pattern recognition?

The terms used in statistical pattern recognition and their equivalent terms in neural-network-based pattern recognition are given in the form of a table.

Sl No	Statistical approach	Neural-network approach
1	Model	Network
2	Estimation	Learning
3	Regression	Supervised learning
4	Parameters	Weights
5	Independent variables	Inputs
6	Dependent variables	Outputs

7. What are the advantages of artificial neural network approach to pattern recognition when compared to the traditional pattern-recognition approach?

The major advantages of artificial neural networks over traditional information-processing techniques are speed of computation, ready adaptation to changes in input data and the ability to deal with non-linear processes. Due to their non-linear nature, they can define non-linear decision surfaces in the feature space of pattern classification problems.

8. Distinguish between statistical and structural approaches to pattern recognition.

The main differences between statistical and structural approaches to pattern recognition are given below:

- (i) The description generated by the statistical approach is quantitative, while structural approach produces a description composed of subpatterns.
- (ii) The statistical approach discriminates upon numerical differences among features from different groups, while grammars are used by the structural approach to define a language encompassing the acceptable configurations of primitives of each group.

9. Develop a perceptron for the AND function with bipolar inputs and targets:

The truth table of AND gate for bipolar inputs is given by

Input			Target
X_1	X_2	b	t
1	1	1	1
-1	1	1	-1
1	-1	1	-1
-1	-1	1	-1

The architecture for this AND function is given in Fig. 8.32.

Step 1 The weights and bias are initialised to zero. That is, $w_1 = w_2 = b = 0$. The learning rate α is set as 1. The threshold parameter $\theta = 0$.

Step 2 Begin computation.

Step 3 For the input pair (1, 1), do steps 4 to 6.

Step 4 Set the activation of input units:

$$x_i = (1, 1)$$

Step 5 Calculate the net input:

$$Y_{in} = b + \sum_{i=1}^2 x_i w_i = b + x_1 w_1 + x_2 w_2$$

Substituting $b = 0$, $w_1 = 0$ and $w_2 = 0$ in the expression for Y_{in} , we get

$$Y_{in} = 0$$

Now the activation is applied to get the output:

$$Y = \begin{cases} 1, & \text{if } Y_{in} > \theta \\ 0, & \text{if } -\theta \leq Y_{in} \leq \theta \\ -1, & \text{if } Y_{in} < -\theta \end{cases}$$

which implies $Y = 0$.

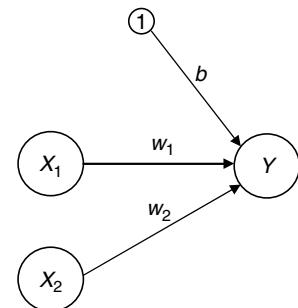


Fig. 8.32 Architecture for AND function

Step 6 For the input pair (1, 1), the desired output is 1, which is evident from the truth table, but the obtained value is zero. That is, since the desired output is not equal to the target value, weight and bias updation has to be performed. In our case, we have two weights w_1 and w_2

Step 6a Updation of weight w_1

$$w_1(\text{new}) = w_1(\text{old}) + \alpha t x_1$$

Substituting the value of the old weight, bias and the target value, we get the new weight value as $w_1(\text{new}) = 0 + 1 \times 1 \times 1 = 1$

Step 6b Updation of weight w_2

$$w_2(\text{new}) = w_2(\text{old}) + \alpha t x_2$$

Substituting the value of the old weight, bias and the target value, the new value of the weight is given by

$$w_2(\text{new}) = 0 + 1 \times 1 \times 1 = 1$$

Step 6c Updation of bias

The bias updation is done according to the formula $b_{(\text{new})} = b_{(\text{old})} + \alpha t$

Substituting the value of the old bias, learning rate and the target value, we get

$$b_{(\text{new})} = 0 + 1 \times 1 = 1$$

The new weights and bias is given by $[w_1(\text{new}), w_2(\text{new}), b_{(\text{new})}] = [1, 1, 1]$

The algorithmic steps are repeated for all the input vectors with their initial weights as the previously calculated weights. The updated weights after presenting all the input vectors are presented below:

Input			NetInput	Output	Target	Weight Changes			Weights		
X_1	X_2	B	Y_{in}	Y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
1	1	1	0	0	1	1	1	1	1	1	1
-1	1	1	1	1	-1	1	-1	-1	2	0	0
1	-1	1	2	1	-1	-1	1	-1	1	1	-1
-1	-1	1	-3	-1	-1	0	0	0	1	1	-1

The final weights after one iteration are given by $w_1 = 1$, $w_2 = 1$ and $b = -1$.

It is known that the net input $b + x_1 w_1 + x_2 w_2 = 0$. From this equation

$$x_2 = -x_1 \frac{w_1}{w_2} - b \frac{w_1}{w_2}$$

Substituting the values of $w_1 = 1$, $w_2 = 1$, and $b = -1$, we get the decision boundary as $x_2 = -x_1 + 1$. The graphical representation of the decision boundary is shown in Fig. 8.33.

Thus perceptron network is trained for the AND function. Similarly, it can be trained for the OR function.

10. Compare the template matching, statistical classification, syntactic method and neural-network approach to pattern recognition.

The comparison of pattern recognition using template matching, statistical method, structural method and neural network method are given below:

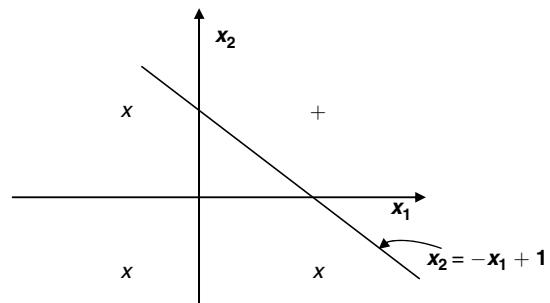


Fig. 8.33 Discrimination function for AND gate

Approach	Representation	Recognition function	Criterion for classification
Template matching	Image pixels, curves, edges in the image	Correlation, distance measure	Classification error
Statistical method	Features	Discriminant function	Classification error
Syntactic or structural method	Primitives	Roles, grammar	Acceptance error
Neural networks	Features	Network function	Mean square error

References

Books

1. K S Fu, *Digital Pattern Recognition*, Springer–Verlag, Berlin Heidelberg, 1976
2. Kenneth R Castleman, *Digital Image Processing*, Pearson Education, 2007
3. Laurene Fausett, *Fundamentals of Neural Networks*, Prentice Hall, Upper Saddle, New Jersey
4. Carl G Looney, *Pattern Recognition using Neural Networks*, Oxford University Press, 1997
5. R O Duda, P E Hart, D G Stork, *Pattern Classification*, John Wiley and Sons, 2008

Journal Articles

1. Anil K Jain, Robert P W Duin and Jianchang Mao, *Statistical Pattern Recognition: A Review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4–37, January 2000
2. Salil Prabhakar, *Fingerprint Classification and Matching Using a Filterbank*, PhD thesis, Michigan State University, 2001
3. Ming-Hsuan Yang, David J Kriegman and Narendra Ahuja, *Detecting Faces in Images: A Survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 34–58, January 2002
4. Anil K Jain, J Mao, K M Mohiuddin, *Artificial Neural Networks: a Tutorial*, IEEE Computer, vol. 29, no. 3, pp. 31–44, March 1996

Thesis Reports

1. Robert T Olszewski, *Generalised Feature Extraction for Structural Pattern Recognition in Time-Series Data*, PhD thesis report, School of Computer Science, Carnegie Mellon University, 2001

9

Learning Objectives

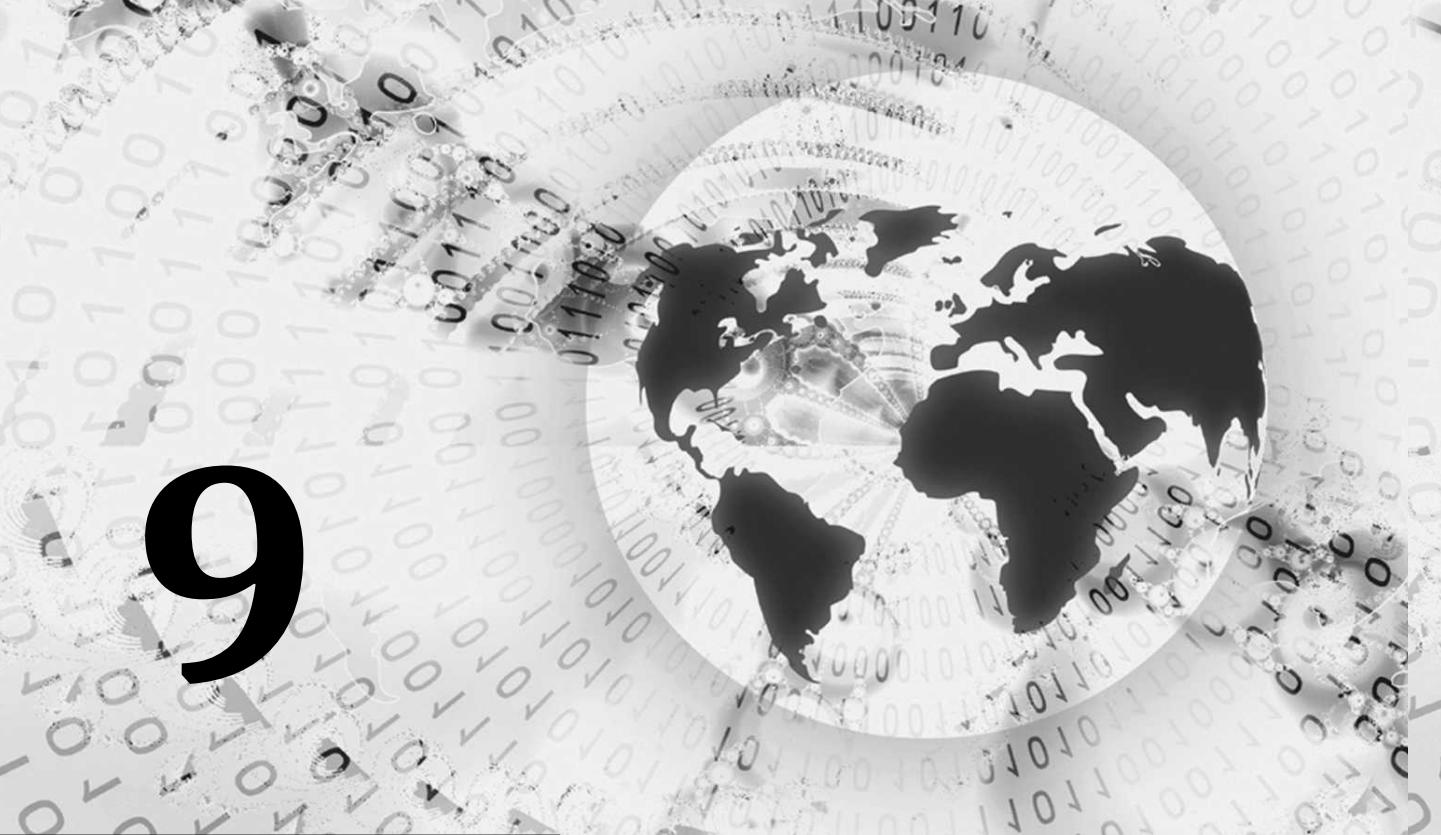
The rapid growth of digital imaging applications, including desktop publishing, multimedia, teleconferencing, and high-definition television has increased the need for effective and standardised image-compression techniques. After completing this chapter, the reader should have a good understanding of the following concepts:

*Need for image compression
lossless and lossy image compression
Spatial domain and frequency domain
image-compression techniques
image-compression metrics*

Image Compression

9.1 INTRODUCTION

With the growth of multimedia technology over the past decades, the demand for digital information increases dramatically. The advances in technology have made the use of digital images prevalent to a large extent. Still images are widely used in applications like medical and satellite images. Digital images are comprised of an enormous amount of data. Reduction in the size of the image data for both storing and transmission of digital images are becoming increasingly important as they find more applications. Image compression is a mapping from a higher dimensional space to a lower dimensional space. Image compression plays an important role in many multimedia applications, such as image storage and transmission. The basic goal



of image compression is to represent an image with minimum number of bits of an acceptable image quality. All image-compression algorithms strive to remove statistical redundancy and exploit perceptual irrelevancy while reducing the amount of data as much as possible.

9.2 NEED FOR IMAGE COMPRESSION

With the advanced development in Internet, teleconferencing, multimedia and high-definition television technologies, the amount of information that is handled by computers has grown exponentially over the past decades. Hence, storage and transmission of the digital image component of multimedia systems is a major problem. The amount of data required to present images at an acceptable level of quality is extremely large. High-quality image data requires large amounts of storage space and transmission bandwidth, something which the current technology is unable to handle technically and economically. One of the possible solutions to this problem is to compress the information so that the storage space and transmission time can be reduced.

For example, if we want to store a 1600×1200 colour image then the space required to store the image is

$$\begin{aligned} 1200 \times 1600 \times 8 \times 3 &= 46,080,000 \text{ bits} \\ &= 5,760,000 \text{ bytes} \\ &= 5.76 \text{ Mbytes} \end{aligned}$$

The maximum space available in one floppy disk is 1.44 Mb. If we have three floppies then the maximum space is $1.44 \times 4 = 5.76$ Mbytes. That is, a minimum of four floppies are required to store an RGB image of size 1600×1200 . This fact is illustrated in Fig. 9.1.

The amount of data transmitted through the Internet doubles every year, and a large portion of that data comprises of images. Reducing the bandwidth needs of any given device will result in significant cost reductions and will make the use of the device more affordable. Image compression offers ways to represent an image in a more compact way, so that images can be stored in a compact manner and can be transmitted faster.

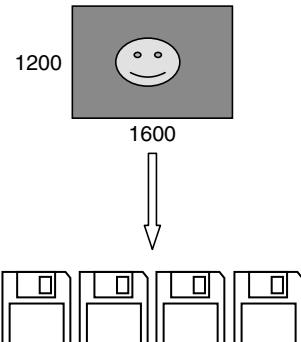


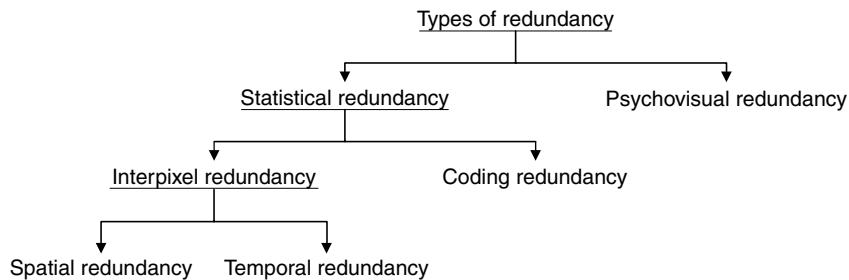
Fig. 9.1 Storage space for a colour image

9.3 REDUNDANCY IN IMAGES

Image compression is possible because images, in general, are highly coherent, which means that there is redundant information. Compression is achieved through redundancy and irrelevancy reduction. Redundancy means duplication, and irrelevancy means the part of the image information that will not be noticed by the human visual system.

9.4 CLASSIFICATION OF REDUNDANCY IN IMAGES

Redundancy can be broadly classified into (i) statistical redundancy, and (ii) psychovisual redundancy. Statistical redundancy can be classified into (a) interpixel redundancy, and (b) coding redundancy. Interpixel redundancy can be further classified into (1) spatial redundancy, and (2) temporal redundancy. The classification of redundancy is illustrated in Fig. 9.2.

Fig. 9.2 *Classification of redundancy*

9.4.1 Statistical Redundancy

As stated, statistical redundancy can be classified into two types: (i) interpixel redundancy, and (ii) coding redundancy.

Interpixel redundancy is due to the correlation between neighbouring pixels in an image. It means that the neighbouring pixels are not statistically independent. The interpixel correlation is referred as interpixel redundancy.

Coding redundancy is associated with the representation of information. The information is represented in the form of codes. The Huffman code and arithmetic codes are some examples of codes. Some codes may be more efficient than others. Codes should be efficient in order to compress the image effectively.

9.4.2 Spatial Redundancy

Spatial redundancy represents the statistical correlation between neighbouring pixels in an image. Spatial redundancy implies that there is a relationship between neighbouring pixels in an image. It is not necessary to represent each pixel in an image independently. Instead a pixel can be predicted from its neighbours. Removing spatial redundancy through prediction is the basic principle of differential coding which is widely employed in image and video compression.

9.4.3 Temporal Redundancy

Temporal redundancy is the statistical correlation between pixels from successive frames in a video sequence. The temporal redundancy is also called *interframe redundancy*. Motion compensated predictive coding is employed to reduce temporal redundancy. Removing a large amount of temporal redundancy leads to efficient video compression.

9.4.4 Psychovisual Redundancy

Psychovisual redundancy is associated with the characteristics of the human visual system (HVS). In the HVS, visual information is not perceived equally. Some information may be more important than other information. If less data is used to represent less important visual information, perception will not be affected. This implies that visual information is psychovisually redundant. Eliminating the psychovisual redundancy leads to efficient compression.

9.5 IMAGE-COMPRESSION SCHEME

The general block diagram of an image-compression scheme is shown in Fig. 9.3.

The source encoder and decoder pair is commonly referred as *source codec module*, whereas the channel encoder and decoder pair is commonly referred as *channel codec module*. Over the past decades, the separate

designs of the two coding modules was justified by Shannon's classic separation theorem.

Source Coding The goal of source coding is efficient conversion of the source data (input image data) into a sequence of bits. The source coder reduces the entropy, which means decrease in the average number of bits required to represent the image. The source decoding is essentially an inverse operation.

Channel The channel is a mathematical model of the medium over which communication occurs.

Channel Coding The channel encoder introduces controlled redundancy to the compressed output of the source encoder. The purpose of channel encoder is to protect the communication system against noise and other transmission errors in the channel. The channel decoder exploits the redundancy in the bit sequence to reproduce the compressed bits.

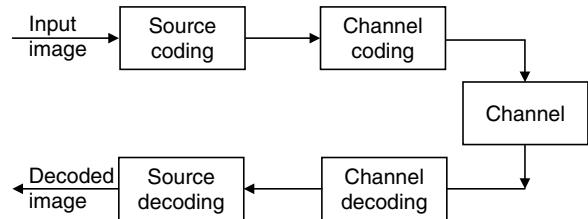


Fig. 9.3 Image-compression scheme

9.6 CLASSIFICATION OF IMAGE-COMPRESSION SCHEMES

Image-compression schemes can be broadly classified into two types: (i) lossless compression scheme, and (ii) lossy compression scheme. Lossless compression scheme is preferred in the case of medical image compression. Lossy compression scheme is preferred in the case of multimedia applications. In the case of a lossless compression scheme, the reconstructed image exactly resembles the original image without any loss of information. But the compression ratio that can be achieved using a lossless compression scheme is usually less. On the other hand, a high compression ratio can be obtained in a lossy compression scheme at the expense of the quality of the reconstructed image. There is always a trade off between the quality of the reconstructed image and the compression ratio.

9.6.1 Lossless Compression or Reversible Compression

In lossless compression, the image after compression and decompression is identical to the original image and every bit of information is preserved during the decomposition process. The reconstructed image after compression is an exact replica of the original one. Although lossless compression methods have the appeal that there is no deterioration in image quality, this scheme only achieves a modest compression rate. The lossless compression scheme is used in applications where no loss of image data can be compromised.

9.6.2 Lossy Compression or Irreversible Compression

In lossy compression, the reconstructed image contains degradations with respect to the original image. Here, a perfect reconstruction of the image is sacrificed by the elimination of some amount of redundancies in the image to achieve a higher compression ratio. In lossy compression, a higher compression ratio can be achieved when compared to lossless compression. The term 'visually lossless' is often used to characterise lossy compression schemes that result in no visible degradation under a set of designated viewing conditions.

9.7 FUNDAMENTALS OF INFORMATION THEORY

Information theory forms the mathematical basis of both lossless and lossy compression. This section deals with the concept of entropy, average information and rate-distortion concepts.

9.7.1 Entropy and Mutual Information

For a random variable X generated by a discrete memoryless source from a finite alphabet set $X = \{x_1, x_2, \dots, x_N\}$ with p_x being the probability density function, Shannon defined a measure of the information content of X , called entropy, as

$$H(X) = \sum_{x \in X} p_x(x) \log \frac{1}{p_x(x)} \quad (9.1)$$

The entropy $H(X)$ is the lower bound of the bit rate that can represent the source without distortion.

The conditional entropy of a random variable Y with a finite alphabet set y is defined as

$$H(X/Y) = \sum_{x \in X, y \in Y} p_{XY} \log_2 \frac{1}{p_{x|y}(x|y)} \quad (9.2)$$

where $p_{xy}(x, y)$ is the joint probability density function of X and Y and $p_{x|y}(x|y)$ is the conditional probability density function of X given Y . The mutual information $I(X; Y)$ between two random variables X and Y is a measurement of the reduction in uncertainty due to conditioning of Y , which is defined as

$$I(X; Y) = H(X) - H(X/Y) \quad (9.3)$$

$$I(X; Y) = H(Y) - H(Y/X) \quad (9.4)$$

The above equation implies that mutual information is symmetric.

9.7.2 Shannon's Source-Coding Theorem

The objective of source coding is to efficiently represent a random experiment's sequence of outcomes.

According to Shannon's source-coding theorem "Let X be the ensemble of letters from a discrete memoryless source with finite entropy $H(X)$. Blocks of J symbols from the source are encoded into code words of length N from a binary alphabet. For any $\varepsilon > 0$, the probability P_e of a block-decoding failure can be made arbitrarily small if

$$R \equiv \frac{N}{J} \geq H(X) + \varepsilon \quad (9.5)$$

and J is sufficiently large. Conversely, if $R \leq H(X) - \varepsilon$ then P_e becomes arbitrarily close to 1 as J is made sufficiently large.

The source-coding theorem indicates that the source entropy $H(X)$ is the lower bound of the average number of bits per symbol required to encode the output of a discrete memoryless source without distortion.

9.7.3 Rate-Distortion Theory

The rate distortion theory is concerned with the task of removing redundancy from a source subject to a pre-defined fidelity criterion. The rate-distortion function represented by $R(D)$ of the source X is defined as the infimum of the distortion of any source-coding scheme which satisfies the rate constraint R . For a discrete memoryless source using the mean square error criterion, the rate-distortion function is given by

$$R(D) = \inf_{p_{Y|X}, \text{MSE}(p_{Y|X}) \leq D} I(X; Y) \quad (9.6)$$

where $p_{Y|X}$ is the source-coding rule and MSE denotes the MSE of the encoding rule.

According to rate-distortion theorem, $R(D)$ is the lowest attainable rate needed to reconstruct the source X with the given distortion D . $R(D)$ is, in general, a convex and monotonically non-increasing function of the distortion D .

Lossy compression is accomplished by means of a trade-off between rate and distortion. Rate is the average number of bits required to represent each source symbol. The trade-off between rate and distortion is represented in the form of the rate-distortion function $R(D)$ as shown in Fig. 9.4.

From Fig. 9.4, it is obvious that the minimum possible distortion is at $D = 0$. The distortion corresponding to $R(D) = 0$ is the maximum amount of distortion incurred. The rate-distortion function $R(D)$ of a source gives the minimum rate that is required to encode the source with a maximum distortion D . The rate-distortion function gives the theoretical lower bound of the rate required to encode a source with a maximum allowable distortion. In order to obtain the rate-distortion function of a source, it is necessary to characterise the source. Theoretical rate-distortion results are only likely to be found for simple source models such as independent identically distributed (i.i.d) scalar sources with Gaussian, Laplacian or generalised Gaussian distributions.

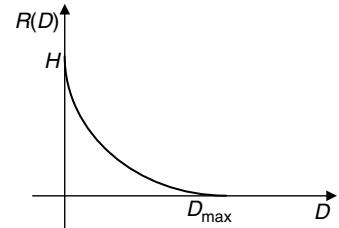


Fig. 9.4 Typical rate-distortion function

9.8 RUN-LENGTH CODING

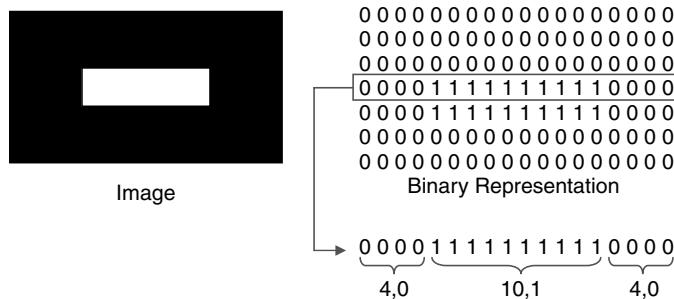
Run-length coding (RLC) is effective when long sequences of the same symbol occur. Run-length coding exploits the spatial redundancy by coding the number of symbols in a run. The term *run* is used to indicate the repetition of a symbol, while the term *run-length* is used to represent the number of repeated symbols. Run-length coding maps a sequence of numbers into a sequence of symbol pairs (run, value). Images with large areas of constant shade are good candidates for this kind of compression. Run-length coding is used in the Windows bitmap file format. Run-length coding can be classified into (i) 1D run-length coding, and (ii) 2D run-length coding. RLC using only the horizontal correlation between pixels on the same scan line is referred as 1D RLC, whereas 2D RLC utilises both horizontal and vertical correlation between pixels.

9.8.1 1D Run-length Coding

In 1D run-length coding, each scan line is encoded independently. Each scan line can be considered as a sequence of alternating, independent white runs and black runs. As an agreement between encoder and decoder, the first run in each scan line is assumed to be a white run. If the first actual pixel is black then the run-length of the first white run is set to be zero. At the end of each scan line, there is end-of-line (EOL). The decoder knows the end of a scan line when it encounters an EOL codeword.

Illustration of 1D Runlength Coding Consider a binary image and the corresponding binary representation as shown in Fig. 9.5. The objective is to code the image using run-length coding.

In run-length coding, two values are transmitted—the first value indicates the number of times a particular symbol has occurred, and the second value indicates the actual symbol. From Fig. 9.5, it is obvious that each row is scanned successively and the corresponding run-length code is transmitted. The run-length coding corresponding to the fourth row of the input image is shown as an example. In the example, the bit stream corresponding to the fourth row is 4, 0, 10, 1, 4, 0.

**Fig. 9.5** Illustration of run-length coding

9.8.2 2D Run-length Coding

The 1D run-length coding utilises the correlation between pixels within a scanline. In order to utilise correlation between pixels in neighbouring scan lines to achieve higher coding efficiency, 2D run-length coding was developed.

9.9 SHANNON–FANO CODING

The Shannon–Fano coding is a top-down binary tree method. The algorithm of Shannon–Fano coding is given below:

1. The set of source symbols are sorted in the order of non-increasing probabilities.
2. Each symbol is interpreted as the root of a tree.
3. The list is divided into two groups such that each group has nearly equal total probabilities.
4. The code word of the first group is appended with 0.
5. The code word of the second group is appended with 1.
6. Steps (3) to (4) are repeated for each group until each subset contains only one node.

Example 9.1 Construct the Shannon–Fano code for the word MUMMY.

Solution The given word is MUMMY. The number of alphabets in the word MUMMY (N) is five.

Step 1 Determining the probability of occurrence of each character

Probability of occurrence of each character in MUMMY is given by

$$P(M) = \frac{3}{5}; \quad P(U) = \frac{1}{5}; \quad P(Y) = \frac{1}{5}.$$

Step 2 Determining the number of steps

Now, the number of steps to be involved in this problem is

$$\text{number of steps} = \text{number of symbols} - 1 = 3 - 1 = 2$$

Step 3 Computation of the code word for each symbol

The steps involved in the computation of the code words is shown in the table given below.

Symbol	Probability of occurrence	Step 1	Step 2	Code
M	3/5	0		0
U	1/5	1	0	10
Y	1/5	1	1	11

Step 4 Determination of entropy

After determining the probability and the code for each character, the next step is to compute the entropy. The entropy is given by

$$H(s) = -\sum_{k=0}^{N-1} P(k) \log_2^{P(k)}$$

$$H(s) = -\frac{\sum_{k=0}^{N-1} P(k) \log_{10}^{P(k)}}{\log_{10}^2}$$

$$H(s) = -\frac{1}{0.3010} \left\{ \frac{3}{5} \log \left(\frac{3}{5} \right) + \frac{1}{5} \log \left(\frac{1}{5} \right) + \frac{1}{5} \log \left(\frac{1}{5} \right) \right\}$$

$$H(s) = \frac{-1}{0.3010} \{ 0.6 \times (-0.2218) + 0.2 \times (-0.6990) + 0.2 \times (0.6990) \}$$

$$H(s) = -\frac{1}{0.3010} \{ -0.4127 \} = 1.3711$$

Step 5 Computation of average length

The average length \bar{L} is given by

$$\bar{L} = \sum_{k=0}^{N-1} P(k) l(k)$$

$$\bar{L} = \left\{ \frac{3}{5} \times 1 + \frac{1}{5} \times 2 + \frac{1}{5} \times 2 \right\} = \left\{ \frac{3+2+2}{5} \right\} = \frac{7}{5} = 1.4$$

Step 6 Computation of efficiency

The efficiency of the code η is given by

$$\eta = \frac{H(s)}{\bar{L}} = \frac{1.3711}{1.4} = 0.9794 = 97.94\%$$

9.10 HUFFMAN CODING

Huffman coding was developed by D A Huffman in 1952. David Huffman developed the algorithm as a student in a class on information theory at MIT in 1950. Huffman codes are optimal codes that map one symbol to one code word. In Huffman coding, it is assumed that each pixel intensity has associated with it a certain probability of occurrence, and this probability is spatially invariant. Huffman coding assigns a binary code to each intensity value, with shorter codes going to intensities with higher probability. If the probabilities can be estimated *a priori* then the table of Huffman codes can be fixed in both the encoder and the decoder. Otherwise, the coding table must be sent to the decoder along with the compressed image data.

The parameters involved in Huffman coding are as follows:

- Entropy
- Average length
- Efficiency
- Variance

Prefix Code A code is a prefix code if no code word is the prefix of another code word. The main advantage of a prefix code is that it is uniquely decodable. An example of a prefix code is the Huffman code.

Types of Huffman Codes Huffman codes can be broadly classified into (i) binary Huffman code, (ii) non-binary Huffman code, (iii) extended Huffman code, and (iv) adaptive Huffman code.

9.10.1 Binary Huffman Code

In binary Huffman code, the code for each symbol will be a combination of ones and zeros. The Huffman code can be represented as a binary tree in which the leaves correspond to the symbols. The Huffman code for any symbol can be obtained by traversing the tree from the root node to the leaf corresponding to the symbol by assigning a 0 to the code word every time the traversal takes us over an upper branch, and assigning a 1 every time the traversal takes us over a lower branch. These ideas are explained in Example 9.2 in a step-by-step approach.

Example 9.2 Obtain the Huffman code for the word 'COMMITTEE'

Solution Total number of symbols in the word 'COMMITTEE' is 9.

$$\text{Probability of a symbol} = \frac{\text{Total number of occurrence of symbol in a message}}{\text{Total number of symbol in the message}}$$

$$\text{Probability of the symbol C} = p(C) = 1/9$$

$$\text{Probability of the symbol O} = p(O) = 1/9$$

$$\text{Probability of the symbol M} = p(M) = 2/9$$

$$\text{Probability of the symbol I} = p(I) = 1/9$$

$$\text{Probability of the symbol T} = p(T) = 2/9$$

$$\text{Probability of the symbol E} = p(E) = 2/9$$

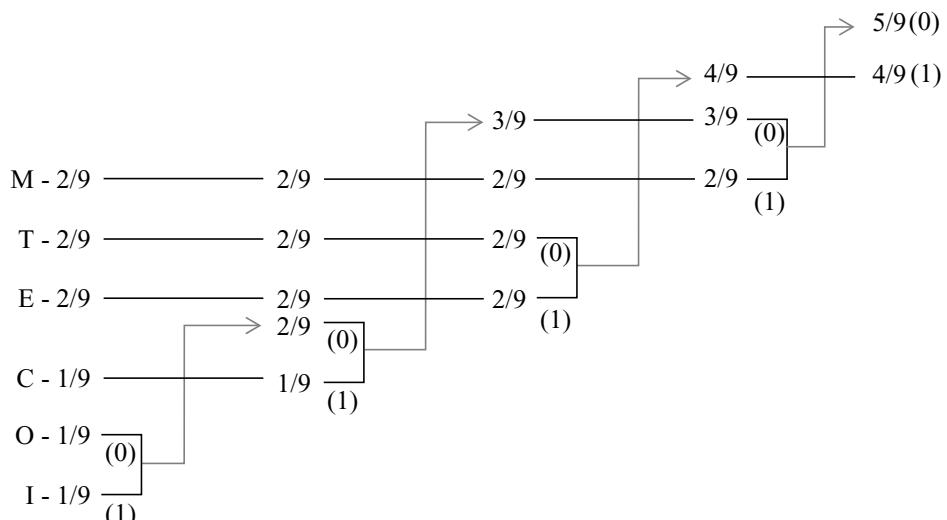
Step 1 Arrange the symbols into descending order according to the probability. This is illustrated in Table 9.1.

Table 9.1 Arrangement of symbols in descending order of probability

Symbol	Probability
M	2/9
T	2/9
E	2/9
C	1/9
O	1/9
I	1/9

Step 2 Construction of Huffman tree

The Huffman tree corresponding to the term COMMITTEE is shown below.



Step 3 Code word from the Huffman Tree

The code word for each symbol and the corresponding word length is shown in Table 9.2.

Table 9.2 Code word from Huffman tree

Symbol	Probability	Binary Huffman method	
		Codeword	Word length
M	2/9	01	2
T	2/9	10	2
E	2/9	11	2
C	1/9	001	3
O	1/9	0000	4
I	1/9	0001	4

Step 4 Determination of average length (\bar{L})

The formula to calculate the average length is given by $\bar{L} = \sum_{k=0}^{N-1} P_k l_k$ (9.7)
 where \bar{L} = average length of the symbol.

P_k = probability of occurrence of the symbol.

l_k = length of each symbol.

$$\bar{L} = \frac{2}{9} \times 2 + \frac{2}{9} \times 2 + \frac{2}{9} \times 2 + \frac{1}{9} \times 3 + \frac{1}{9} \times 4 + \frac{1}{9} \times 4$$

Solving this, we get the average length as

$$\bar{L} = 2.5553 \text{ bits/symbol}$$

Step 5 Determination of the entropy ($H(S)$)

The formula to determine the entropy of the source is given by $H(s) = -\sum_{k=0}^{N-1} P_k \log_2 P_k$. The same formula can be written as $H(s) = -\frac{1}{0.3010} \sum_{k=0}^{N-1} P_k \log_{10} P_k$. Substituting the value of the probability, we get $H(S) = \frac{-1}{0.3010} \left\{ \frac{2}{9} \log_{10} \frac{2}{9} + \frac{2}{9} \log_{10} \frac{2}{9} + \frac{2}{9} \log_{10} \frac{2}{9} + \frac{1}{9} \log_{10} \frac{1}{9} + \frac{1}{9} \log_{10} \frac{1}{9} + \frac{1}{9} \log_{10} \frac{1}{9} \right\}$. On

simplifying the value, the entropy is obtained as $H(S) = 2.5034$ bits/symbol

Step 6 Determination of efficiency

The efficiency of the Huffman code is given by $\eta = \frac{H(S)}{\bar{L}}$. Substituting the value of entropy and the average length, the value of efficiency is given by

$$\eta = \frac{2.5034}{2.5553} = 0.9797 = 97.97\%.$$

9.10.2 Non-binary Huffman Code

As a non-binary Huffman code, consider the ternary Huffman code. The number of nodes to be combined initially is given by the formula

$$J = 2 + (N - 2) \bmod (D - 1) \quad (9.8)$$

where J = number of symbols to be combined initially

N = number of probabilities

D = number of counts

Example 9.3 Form the ternary Huffman code for the word COMMITTEE.

Solution The number of nodes to be combined initially is given by

$$J = 2 + (N - 2) \bmod (D - 1)$$

In this example, $N = 6, D = 3$. Substituting these values in the formula for J , we get

$$J = 2 + (6 - 2) \bmod (3 - 1)$$

$$J = 2 + 4 \bmod 2$$

$$J = 2 + 0 = 2.$$

This implies that initially two nodes are to be combined. Following this procedure, the ternary Huffman code for the word COMMITTEE is formed as shown in Fig. 9.6.

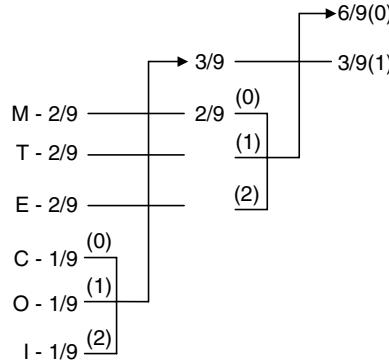


Fig. 9.6 Ternary Huffman code for the word COMMITTEE

The code word for the term COMMITTEE using ternary Huffman tree is shown in Table 9.3.

Table 9.3 Code word from ternary Huffman tree

Symbol	Probability	Ternary Huffman method	
		Code word	Word length
M	2/9	00	2
T	2/9	01	2
E	2/9	02	2
C	1/9	10	2
O	1/9	11	2
I	1/9	12	2

Finding the Entropy $H(S)$

$$H(s) = -\sum_{k=0}^{N-1} P_k \log_2 P_k \quad (9.9)$$

$$H(s) = -\frac{1}{0.3010} \sum_{k=0}^{N-1} P_k \log_{10} P_k \quad (9.10)$$

$$H(s) = \frac{1}{0.3010} \sum_{k=0}^{N-1} P_k \log_{10} \left(\frac{1}{P_k} \right) \quad (9.11)$$

$$H(S) = \frac{-1}{0.3010} \left\{ \frac{2}{9} \log_{10} \frac{2}{9} + \frac{2}{9} \log_{10} \frac{2}{9} + \frac{2}{9} \log_{10} \frac{2}{9} + \frac{1}{9} \log_{10} \frac{1}{9} + \frac{1}{9} \log_{10} \frac{1}{9} + \frac{1}{9} \log_{10} \frac{1}{9} \right\}$$

$$H(S) = 2.5034 \text{ bits/symbol}$$

Finding the average length \bar{L} (Ternary Huffman Coding)

$$\begin{aligned} \bar{L} &= \sum_{k=0}^{N-1} P_k l_k = \frac{2}{9} \times 2 + \frac{2}{9} \times 2 + \frac{2}{9} \times 2 + \frac{1}{9} \times 2 + \frac{1}{9} \times 2 + \frac{1}{9} \times 2 \\ &= 0.4444 + 0.4444 + 0.4444 + 0.2222 + 0.2222 + 0.2222 \\ \bar{L} &= 1.9998 \text{ bits/symbol} \end{aligned}$$

Efficiency of the compression in ternary Huffman Coding (η)

The efficiency of the Huffman code is given by

$$\eta = \frac{H(S)}{\bar{L}}$$

By substituting the value of $H(S)$, \bar{L} we get

$$\eta = \frac{2.5034}{1.9998} = 125.18\%$$

$$\eta = 1.2518$$

9.10.3 Adaptive Huffman Code

The adaptive Huffman code was first discovered by Newton Faller. The discovery became his post graduate thesis at the Federal University of Rio de Janeiro. The adaptive Huffman code learns the symbol probabilities by dynamically using the symbol counts to adjust the code tree. The decoder must use the same initial counts and the count incrementation algorithm used by the encoder, so the encoder-decoder pair maintains the same tree.

9.10.4 Limitations of Huffman Code

The average code-word length achieved by Huffman coding satisfies the inequality

$$H(S) \leq L_{\text{avg}} < H(S) + p_{\max} + 0.086 \quad (9.12)$$

where $H(S)$ is the entropy of the source alphabet and p_{\max} denotes the maximum occurrence probability in the set of the source symbols. The inequality indicates that the upper bound of the average code word length of the Huffman code is determined by the entropy and the maximum occurrence probability of the source symbols being encoded.

The Huffman coding always encodes a source symbol with an integer number of bits. If the probability distribution of the source symbol is such that some probabilities are small while some are quite large then

the entropy of the source alphabet will be close to zero since the uncertainty is very small. In such cases, two bits are required to represent the information, that the average code-word length is one, which means that the redundancy is very close to one. This inefficiency is due to the fact that Huffman coding always encodes a source symbol with an integer number of bits. To overcome this limitation, arithmetic coding was proposed which is stream-based. A string of source symbols is encoded as a string of code symbols.

9.11 ARITHMETIC CODING

In arithmetic coding, the interval from zero to one is divided according to the probabilities of the occurrences of the intensities. Arithmetic coding does not generate individual codes for each character but performs arithmetic operations on a block of data, based on the probabilities of the next character. Using arithmetic coding, it is possible to encode characters with a fractional number of bits, thus approaching the theoretical optimum.

Arithmetic coding performs very well for sequences with low entropy where Huffman codes lose their efficiency.

Graphical Method of Arithmetic Coding The step-by-step procedure of the graphical method of arithmetic coding is illustrated through an example which is given below.

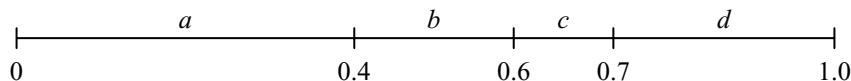
Example 9.4: A source emits four symbols $\{a, b, c, d\}$ with the probabilities 0.4, 0.2, 0.1 and 0.3 respectively. Construct arithmetic coding to encode and decode the 'word dad'.

Solution The symbol probabilities and their corresponding sub-ranges are tabulated below.

Symbol	a	b	c	d
Probability	0.4	0.2	0.1	0.3
Sub-range	$(0.0 - 0.4)$	$(0.4 - 0.6)$	$(0.6 - 0.7)$	$(0.7 - 1.0)$

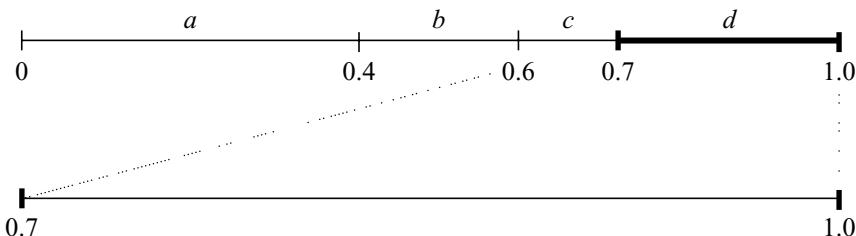
Encoder

Step 1 Our objective is to encode the 'word dad'. Initially, the range of intervals is assumed to be between 0 and 1. The individual symbol range in the interval 0 to 1 is shown below.



Step 2

Step 2a The first symbol to be transmitted is d . Hence, the new range is from 0.7 to 1.0 which is shown below.



Step 2b Find the sub-range for each symbol in between the intervals 0.7 and 1.0.

The formula that is used to compute the low and high range for each symbol within the interval is given by

$$\text{low} = \text{low} + \text{range} \times (\text{low_range})$$

$$\text{high} = \text{low} + \text{range} \times (\text{high_range})$$

Here, `low_range` refers to the symbol of low range.

(i) *To compute interval for the symbol a in the interval 0.7 to 1.0.*

First, the range is determined to be $\text{range} = 1.0 - 0.7 = 0.3$

Using the formula given in (1) the low range for the symbol *a* is computed as

$$\text{low} = \text{low} + \text{range} \times (\text{low_range})$$

In this case, $\text{low} = 0.7$, $\text{range} = 0.3$ and `low_range` for symbol *a* is 0.0. Substituting these values, the lower interval is computed as

$$\text{low} = 0.7 + 0.3 \times 0 = 0.7$$

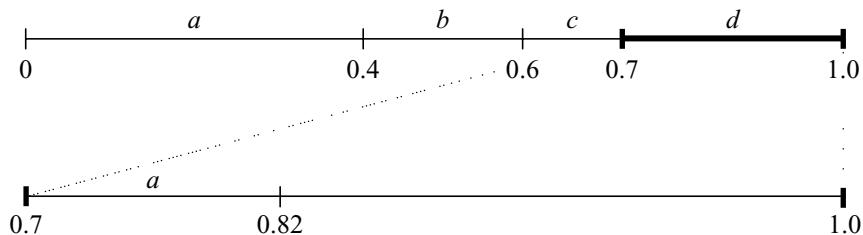
To determine the higher interval for the symbol, the formula to be used is

$$\text{high} = \text{low} + \text{range} \times (\text{high_range})$$

Here, $\text{low}=0.7$, range is 0.3 and $\text{high_range}=0.4$. Substituting these values, the higher interval is computed as

$$\text{high} = 0.7 + 0.3 \times 0.4 = 0.82$$

For symbol *a*, the lower interval is computed to be 0.7 and the higher interval is 0.82, which is illustrated below.



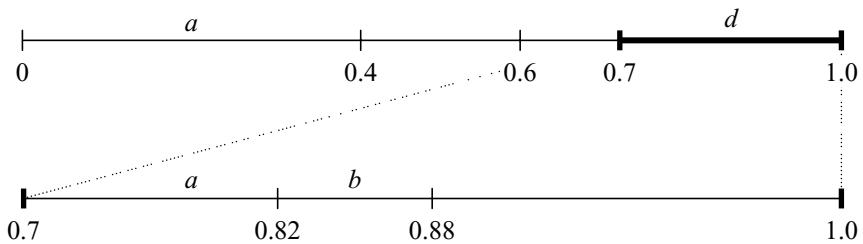
(ii) *To compute the interval for the symbol b in the interval 0.7 to 1.0*

The lower interval of the symbol *b* is computed as

$$\text{low} = 0.7 + 0.3 \times 0.4 = 0.82$$

The upper interval of the symbol *b* is computed as

$$\text{high} = 0.7 + 0.3 \times 0.6 = 0.88$$



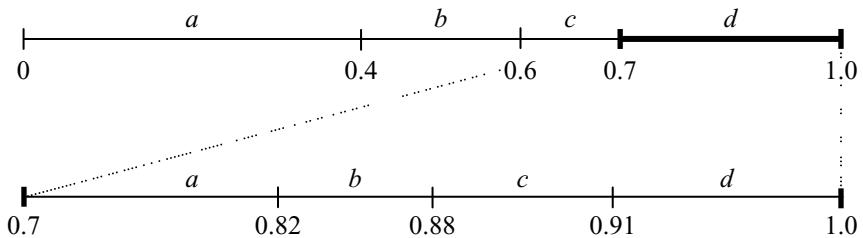
(iii) To compute the interval for the symbol c in the range 0.7 to 1.0

The lower interval of the symbol c is computed as

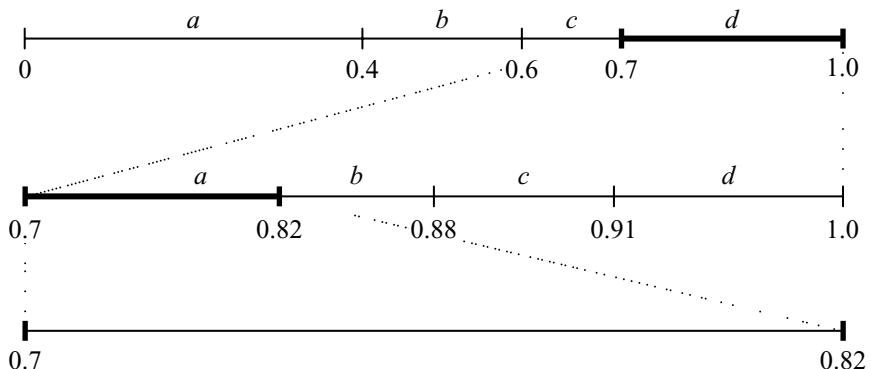
$$\text{low} = 0.7 + 0.3 \times 0.6 = 0.88$$

The upper interval for the symbol c is computed as

$$\text{high} = 0.7 + 0.3 \times 0.7 = 0.91$$



Step 3 The symbol to be transmitted is a . The next symbol to be transmitted in the word ‘dad’ is a . Hence the new interval is 0.7 to 0.82 which is illustrated below.



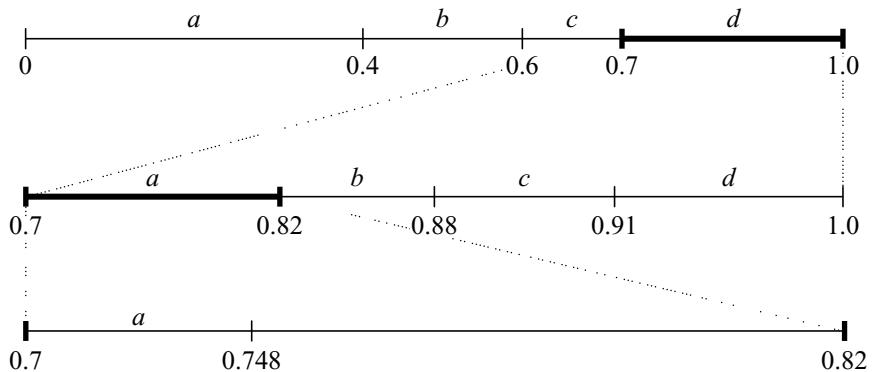
Step 3a To find the interval for the symbols in the range 0.7 to 0.82

(i) To find the interval for the symbol ‘ a ’, we find the lower interval which is given by

$$\text{low} = \text{low} + \text{range} \times (\text{low_range})$$

In this case, $\text{range} = 0.82 - 0.70 = 0.12$

$$\text{low} = 0.7 + 0.12 \times 0.0 = 0.7$$



The upper interval is given by

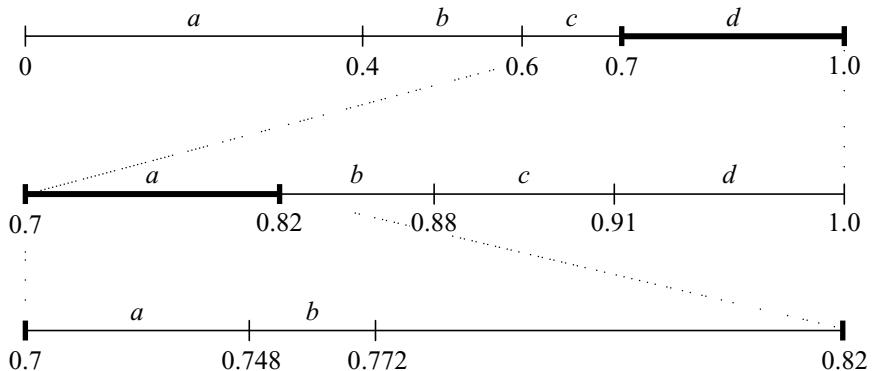
$$\text{high} = 0.7 + 0.12 \times 0.4 = 0.748$$

(ii) To find the lower and upper interval for the symbol 'b', we find the lower interval which is given by

$$\text{low} = 0.7 + 0.12 \times 0.4 = 0.748$$

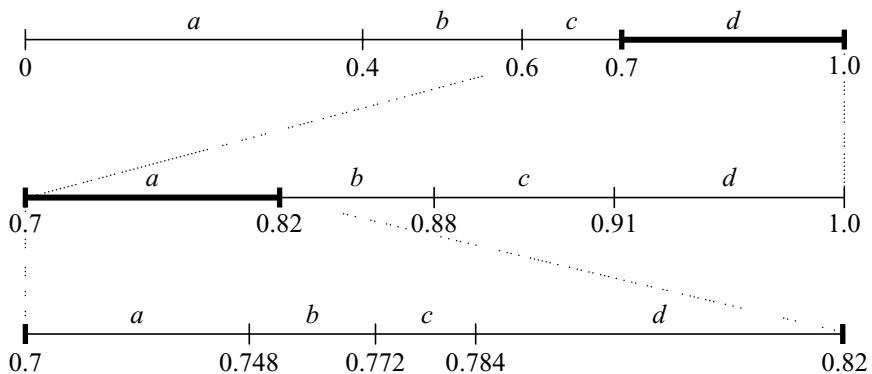
The upper interval is given by

$$\text{high} = 0.7 + 0.12 \times 0.6 = 0.772$$

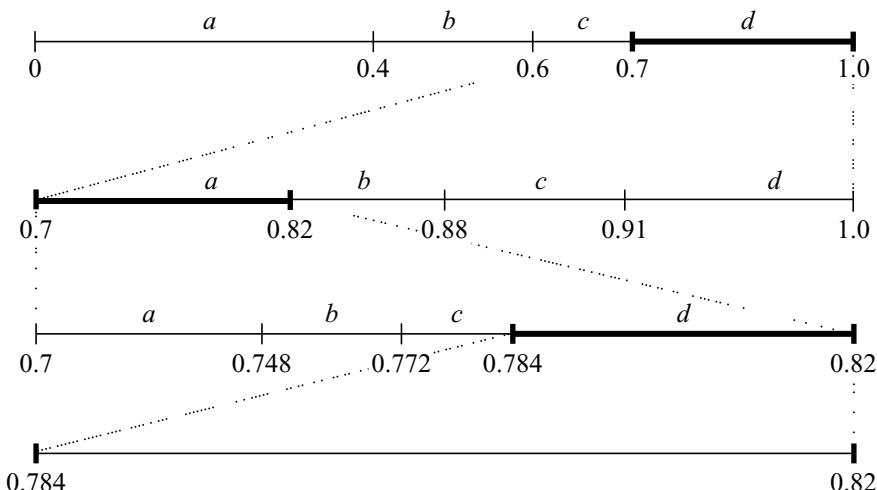


(iii) To find the lower and upper interval for the symbol 'c', we find the lower limit which is given by
 $\text{low} = 0.7 + 0.12 \times 0.6 = 0.772$

The upper limit is given by $\text{high} = 0.7 + 0.12 \times 0.7 = 0.784$



Step 4 To transmit the symbol 'd' in the 'word dad' In order to transmit the next symbol d the intervals 0.784 and 0.82 are taken as the new lower and upper intervals which is illustrated below.



Step 5 To compute the tag value Tag is the average of the lower and upper intervals. Here, the lower interval is 0.784 and the upper interval is 0.82. Hence the tag value is

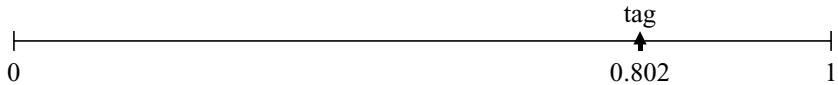
$$\text{tag} = \frac{0.784 + 0.82}{2} = 0.802$$

Decoding Procedure

The tag value and the symbol probabilities will be sent to the receiver. After receiving the tag value, the decoder decodes the encoded data. The step-by-step illustration of the decoding procedure is given below.

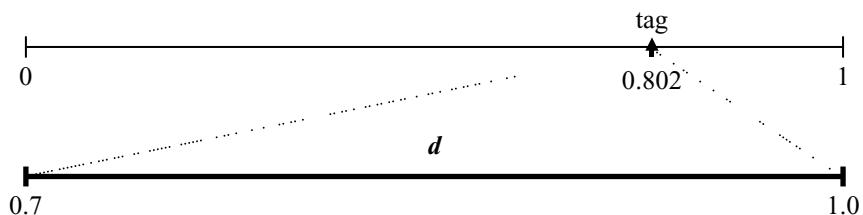
Symbol	a	b	c	d
Probability	0.4	0.2	0.1	0.3
Sub-range	(0.0 – 0.4)	(0.4 – 0.6)	(0.6 – 0.7)	(0.7 – 1.0)

Step 1 The tag received by the receiver is 0.802. The initial interval is assumed to be between 0 and 1.



The tag value is compared with the symbol sub-range. We find that 0.802 lies between 0.7 and 1.0, hence the corresponding decoded symbol is *d*.

Step 2 The decoded symbol in step 1 is *d* and hence the new interval is fixed as 0.7 to 1.0 which is illustrated below.



Step 3 The new tag value is computed in this step. The new tag value is given by

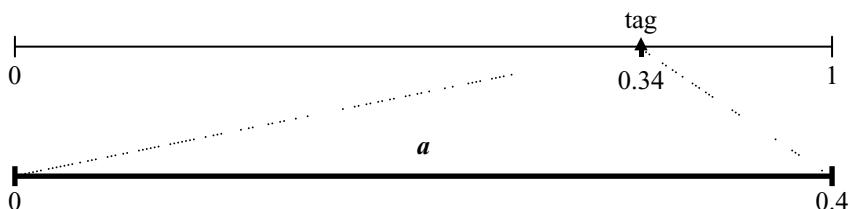
$$t^* = \frac{\text{tag} - \text{low}}{\text{range}}$$

Here, the low value is 0.7, the range is $1.0 - 0.7$ which is 0.3, and hence the new tag value is given by

$$t^* = \frac{0.802 - 0.7}{0.3} = 0.34$$

We find that this tag value lies in the interval 0 to 0.4, and hence the decoded symbol is *a*.

Step 4 The decoded symbol is *a*. Hence the new interval is changed to 0 to 0.4.

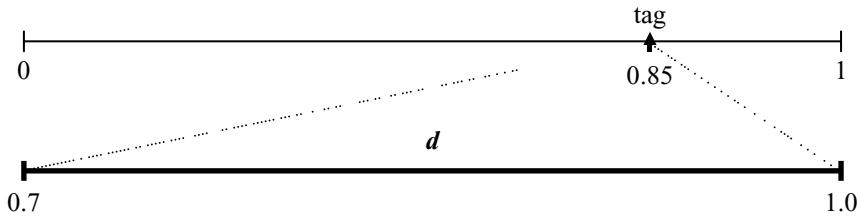


Step 5 The new tag value for the interval given in Step 4 is given by

$$t^* = \frac{\text{tag} - \text{low}}{\text{range}} = \frac{0.34 - 0}{0.4} = 0.85$$

This tag value lies between 0.7 and 1.0, and hence the decoded symbol is *d*.

By following the steps 1, 3 and 5, it is clear that the decoded word is ‘dad’.



Analytical Method of Arithmetic Coding

The encoding and the decoding procedure of analytical method of arithmetic coding along with an example are given in this section.

Encoding Procedure The encoding procedure followed in arithmetic coding is summarised below.

- (i) The encoding procedure begins with the initial interval $l^{(0)} = 0$ and $u^{(0)} = 1$. Here, l stands for the lower limit and u stands for the upper limit. Initially, the lower limit is zero and the upper limit is one.
- (ii) For each occurrence of the symbol, two steps are performed:
 - (a) The current interval is divided into sub-intervals. The size of the sub-interval is proportional to the estimated probability. The formula for dividing the interval is

$$l^{(n)} = (l^{(n-1)} + u^{(n-1)}) F_X(x_n - 1) \quad (9.13)$$

$$u^{(n)} = (l^{(n-1)} + u^{(n-1)}) F_X(x_n) \quad (9.14)$$

In Eqs. (9.13) and (9.14), l represents the lower limit and u represents the upper limit. F_X represents the cumulative density function. The cumulative density function can be obtained directly from the probability model. The cumulative function is defined as

$$F_X(i) = \sum_{k=1}^i P(X = k) \quad (9.15)$$

- (b) The sub-interval is selected corresponding to the symbol that occurs next and making it the new current interval.
- (iii) The midpoint of the final interval is used to generate the tag. The tag is given by

$$\text{tag} = \frac{l^{(n)} + u^{(n)}}{2} \quad (9.16)$$

Decoding Procedure The decoding procedure is concerned with the extraction of the encoded information from the tag. The steps followed in decoding procedure are summarised below.

- (i) Initialise $l^{(0)} = 0$ and $u^{(0)} = 1$.
- (ii) For each iteration n find $t^* = \frac{(\text{tag} - l^{(n-1)})}{(u^{(n-1)} - l^{(n-1)})}$.
- (iii) Find the value of x_n for which $F_X(x_n - l) \leq t^* < F_X(x_k)$.

- (iv) Update $u^{(n)}$ and $l^{(n)}$.
- (v) Continue until the entire sequence has been decoded.

If the length of the transmitted sequence is passed to the decoder then it knows when to terminate.

Example 9.5 Generate a tag using arithmetic coding procedure to transmit the word **INDIA**.

Solution The step-by-step procedure to generate the tag is given below.

- (i) The lower interval is initially chosen as zero that is $l^{(0)} = 0$. Similarly, the upper interval is chosen as one that is $u^{(0)} = 1$.
- (ii) The probability of each symbol in the word INDIA is computed next.
 - (a) Probability of occurrence of the symbol *A* is $p(A) = \frac{1}{5} = 0.2$.
 - (b) Probability of occurrence of the symbol *D* is $p(D) = \frac{1}{5} = 0.2$.
 - (c) Probability of occurrence of the symbol *I* is $p(I) = \frac{2}{5} = 0.4$.
 - (d) Probability of occurrence of the symbol *N* is $p(N) = \frac{1}{5} = 0.2$
- (iii) Determination of the Cumulative Density Function
 - (e) The cumulative density function $F_X(A) = 0.2$
 - (f) The cumulative density function $F_X(D) = 0.4$
 - (g) The cumulative density function $F_X(I) = 0.8$
 - (h) The cumulative density function $F_X(N) = 1.0$
- (iv) Formula to compute the interval

$$l^{(n)} = l^{(n-1)} + [u^{(n-1)} - l^{(n-1)}]F_x \{x_n - 1\}$$

$$u^{(n)} = l^{(n-1)} + [u^{(n-1)} - l^{(n-1)}]F_x \{x_n\}$$

- (v) Encoding of the symbols one by one.

(A) *Encoding of the first symbol in the word INDIA.*

The first symbol in the word 'I' in INDIA is encoded first.

The lower interval is calculated as

$$l^{(1)} = l^{(0)} + [u^{(0)} - l^{(0)}]F_x \{D\}$$

Substituting $l^{(0)} = 0$, $u^{(0)} = 1$ and $F_x(D) = 0.4$ in the above expression, we get

$$l^{(1)} = 0 + [1 - 0] \times 0.4 = 0.4$$

The upper interval is calculated as

$$u^{(1)} = l^{(0)} + [u^{(0)} - l^{(0)}]F_x\{I\}$$

Substituting $l^{(0)} = 0$, $u^{(0)} = 1$ and $F_x(I) = 0.8$ in the above expression, we get

$$u^{(1)} = 0 + (1 - 0) \times 0.8 = 0.8$$

The new interval is $l^{(1)} = 0.4$ and $u^{(1)} = 0.8$.

(B) Encoding of the second symbol 'N' in the word INDIA

$$l^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)})F_x\{N\}$$

Substituting $l^{(1)} = 0.4$, $u^{(1)} = 0.8$ and $F_x(N) = 0.8$ in the above expression, we get

$$l^{(2)} = 0.4 + (0.8 - 0.4) \times 0.8$$

$$l^{(2)} = 0.4 + (0.4) \times 0.8 = 0.72$$

Then, we have to determine the upper interval $u^{(2)}$. The formula to determine $u^{(2)}$ is given by

$$u^{(2)} = l^{(1)} + [u^{(1)} - l^{(1)}]F_x\{N\}$$

Substituting $l^{(1)} = 0.4$, $u^{(1)} = 0.8$ and $F_x(N) = 1.0$ in the above expression, we get

$$u^{(2)} = 0.4 + [0.8 - 0.4] \times 1 = 0.8$$

The new intervals are $l^{(2)} = 0.72$ and $u^{(2)} = 0.8$.

(C) Encoding of the third symbol 'D' in the word INDIA

The new interval is given by

$$l^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)})F_x\{D\}$$

Substituting $l^{(2)} = 0.72$, $u^{(2)} = 0.8$ and $F_x(D) = 0.2$ in the above expression, we get the value of the lower limit

$$l^{(3)} = 0.72 + 0.08 \times 0.2 = 0.736$$

Similarly, the upper interval is obtained using $u^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)})F_x\{D\}$

Substituting $l^{(2)} = 0.72$, $u^{(2)} = 0.8$ and $F_x(D) = 0.4$ in the above expression, we get

$$u^{(3)} = 0.72 + [0.8 - 0.72] \times 0.4 = 0.752$$

The new interval is $l^{(3)} = 0.736$ and $u^{(3)} = 0.752$.

(D) Encoding of the third symbol 'T' in the word INDIA

The new interval to code the symbol I is given by

$$l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})F_x\{D\}$$

Substituting $l^{(3)} = 0.736$, $u^{(3)} = 0.752$ and $F_x(D) = 0.4$ in the above expression, we get

$$l^{(4)} = 0.736 + [0.752 - 0.736] \times 0.4 = 0.7424$$

Then, the upper interval is obtained using the formula $u^{(4)} = l^{(3)} + [u^{(3)} - l^{(3)}]F_x\{I\}$

$$u^{(4)} = 0.736 + [0.752 - 0.736] \times 0.8 = 0.7488$$

Then the new limits are $l^{(4)} = 0.7424$ and $u^{(4)} = 0.7488$

(E) *Encoding the final symbol 'A' in the word INDIA*

The new lower limit to encode the symbol 'A' is given by

$$l^{(5)} = l^{(4)} + (u^{(4)} - l^{(4)})F_x\{0\}$$

Substituting $l^{(4)} = 0.7424$, $u^{(4)} = 0.7488$ and $F_x\{0\} = 0$, we get

$$l^{(5)} = 0.7424 + [0.7488 - 0.7424] \times 0 = 0.7424$$

Then the upper limit is obtained using the formula $u^{(5)} = l^{(4)} + (u^{(4)} - l^{(4)})F_x\{A\}$

Substituting $l^{(4)} = 0.7424$, $u^{(4)} = 0.7488$ and $F_x\{A\} = 0.2$ in the above expression, we get

$$u^{(5)} = 0.7424 + (0.7488 - 0.7424) \times 0.2 = 0.74368$$

The lower and the upper limits after the final word being encoded is

$$l^{(5)} = 0.7424 \text{ and } u^{(5)} = 0.74368$$

(vi) Tag from the lower and upper interval

The tag using the final lower and upper interval is given by

$$\text{tag} = \frac{l^{(n)} + u^{(n)}}{2}$$

In our case, $l^{(n)} = l^{(5)}$; $u^{(n)} = u^{(5)}$. Substituting the values of $l^{(5)}$ and $u^{(5)}$ in the expression of tag, we get $\text{tag} = \frac{0.7424 + 0.74368}{2} = 0.74304$.

Example 9.6 Decode the tag generated in Example 9.5 to get the word INDIA.

Solution The tag obtained in the encoding process is $t = 0.74304$. From this code, we have to get the transmitted word 'INDIA'. Let us perform the decoding process in a step-by-step approach.

Step 1 Initialisation of the lower and upper limits Initially, the lower limit is set to zero and the upper limit is set to one. This is represented by $l^{(0)} = 0$ and $u^{(0)} = 1$.

Step 2 Determining t^* The formula to compute t^* is $t^* = \frac{t - l^{(n-1)}}{u^{(n-1)} - l^{(n-1)}}$. Initially, substituting the lower limit as zero and the upper limit as one we get $t^* = \frac{0.74304}{1 - 0} = 0.74304$. The value of t^* lies between $F_x(D)$ and $F_x(I)$ which is represented as

$$F_x(D) \leq 0.74304 < F_x(I)$$

Hence, the decoded symbol is 'I'.

Step 3 After decoding the first symbol 'I', we have to update the values of the lower and upper limit The lower limit is given by $l^{(1)} = l^{(0)} + [u^{(0)} - l^{(0)}]F_x\{D\}$. Substituting $l^{(0)} = 0$, $u^{(0)} = 1$ and $F_x\{D\} = 0.4$ we get $l^{(1)} = 0 + [1 - 0] \times 0.4 = 0.4$.

The upper limit is given by $u^{(1)} = l^{(0)} + [u^{(0)} - l^{(0)}]F_x\{I\}$. Substituting $l^{(0)} = 0$, $u^{(0)} = 1$ and $F_x\{I\} = 0.8$, we get $u^{(1)} = 0.8$. Then the tag value is calculated as

$$t^* = \frac{t - l^{(1)}}{u^{(1)} - l^{(1)}}$$

Substituting $t = 0.74304$, $l^{(1)} = 0.4$ and $u^{(1)} = 0.8$ in the above expression, we get

$$t^* = \frac{0.74304 - 0.4}{0.8 - 0.4} = 0.8576$$

It can be observed that t^* value lies between $F_x^{(I)}$ and $F_x^{(N)}$ which is represented

$$F_x^{(I)} \leq 0.8576 < F_x^{(N)}$$

The decoded symbol is N.

Step 4 Update the symbols After decoding the symbol N, we have to update the lower and upper limits. The lower limit is given by

$$l^{(2)} = l^{(1)} + [u^{(1)} - l^{(1)}]F_x^{(I)}.$$

Substituting the values of $l^{(1)}$, $u^{(1)}$ and $F_x^{(I)}$, we get

$$l^{(2)} = 0.4 + (0.8 - 0.4) \times 0.8 = 0.72$$

The upper limit is given by $u^{(2)} = l^{(1)} + [u^{(1)} - l^{(1)}]F_x^{(N)}$. By substituting the values of $l^{(1)}$, $u^{(1)}$ and $F_x^{(N)}$, we get $u^{(2)} = 0.8$. The values of $l^{(2)}$ and $u^{(2)}$ are $l^{(2)} = 0.72$ and $u^{(2)} = 0.8$.

Step 5 Computation of t^* The value of t^* after the updation step is $t^* = \frac{t - l^{(2)}}{u^{(2)} - l^{(2)}}$. Then substituting the values of $l^{(2)}$, $u^{(2)}$ from Step 4, we get $t^* = \frac{0.74304 - 0.72}{0.8 - 0.72} = 0.288$. We can observe that

$$F_x^{(A)} \leq 0.288 < F_x^{(D)}$$

Hence, the decoded symbol is *D*.

Step 6 Updation of the lower and upper interval values The lower limit is given by $l^3 = l^2 + (u^2 - l^2)F_x^{(A)}$. Substituting the values of l^2 , u^2 and $F_x^{(A)}$, we get $l^3 = 0.72 + (0.8 - 0.72) \times 0.2 = 0.736$.

The upper limit is given by $u^3 = l^2 + (u^2 - l^2)F_x^{(D)}$. Substituting the values of l^2 , u^2 and $F_x^{(D)}$, we get $u^3 = 0.72 + (0.8 - 0.72) \times 0.4 = 0.752$.

After updation, the lower limit is $l^3 = 0.736$ and $u^3 = 0.752$

Step 7 Computation of t^*

$$t^* = \frac{\text{tag} - l^{(3)}}{u^{(3)} - l^{(3)}} = \frac{0.74304 - 0.736}{0.752 - 0.736} = 0.44$$

Also, $F_x^{(D)} \leq t^* < F_x^{(I)}$. Hence, the decoded symbol is '*I*'.

Step 8 Updation of the lower and the upper limits The lower limit is given by $l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})F_x^{(D)}$. Substituting the values of $l^{(3)}$, $u^{(3)}$ and $F_x^{(D)}$, we get $l^{(4)} = 0.736 + (0.752 - 0.736) \times 0.4 = 0.7424$.

The upper limit is given by $u^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})F_x^{(I)}$. Substituting the values of $l^{(3)}$, $u^{(3)}$ and $F_x^{(I)}$, we get $u^{(4)} = 0.736 + (0.752 - 0.736) \times 0.8 = 0.7488$.

After updation, the lower limit is $l^{(4)} = 0.7424$ and $u^{(4)} = 0.7488$

Step 9 Computation of t^* The value of t^* is given by $t^* = \frac{\text{tag} - l^{(4)}}{u^{(4)} - l^{(4)}}$. Substituting the values of $l^{(4)}$, $u^{(4)}$, we get

$$t^* = \frac{\text{tag} - l^{(4)}}{u^{(4)} - l^{(4)}} = \frac{0.74304 - 0.7424}{0.7488 - 0.7424} = 0.1$$

After computing t^* , we find that $F_x^{(0)} \leq t^* < F_x^{(A)}$. This means the decoded symbol is '*A*'.

Thus we find that the entire word INDIA is decoded from the transmitted tag along with the probability of occurrence of the symbols.

9.12 DICTIONARY-BASED COMPRESSION

Dictionary-based compression uses a family of algorithms that encode variable-length patterns of symbols as an index to the same pattern in a dictionary or look-up table. If this index is smaller than the actual length of the pattern then compression occurs. The dictionary used for compression can be static or adaptive (dynamic). Most dictionary-based schemes are based on the Lempel–Ziv algorithm. This algorithm was developed by Jacob Ziv and Abraham Lempel, two Israeli information theorists, who published the seminal papers on dictionary-based data compression in 1977 and 1978. These algorithms, known as the LZ77 and the LZ78, and coined after the initials of their creators and the year they were created, are the basis of all compression used in lossless applications. The Lempel–Ziv algorithm is used in the GIF file format. This is due to the fact that the algorithm is fast and not complicated.

9.12.1 LZ77 Algorithm

In the LZ77 algorithm, the data to be encoded is passed over by a window. This window is divided into two sections. The first section consists of a large block of recently encoded data. The second section is a smaller section and it is termed as look-ahead buffer. The look-ahead buffer has characters read in from the input stream but not yet encoded. The algorithm tries to match the contents of the look-ahead buffer to a string in the dictionary.

9.12.2 LZ78 Algorithm

The LZ78 algorithm takes a different approach to coding a digital signal, using a different dictionary-based method to get around the problems in the LZ77 algorithm.

9.13 PREDICTIVE CODING

Predictive methods involve predicting the current pixel value based on the value of the previously processed pixels. Usually, the difference between the predicted value and the actual value is transmitted. The receiver makes the same prediction as the transmitter and then adds the difference signal in order to reconstruct the original value.

9.13.1 Delta Encoding

The term ‘delta encoding’ refers to the techniques that store data as the difference between successive samples rather than storing the samples themselves. Delta encoding is effective if there is only a small change between adjacent sample values. The delta-encoding techniques can be broadly classified into (i) delta modulation, (ii) Pulse Code Modulation (PCM), (iii) Differential Pulse Code Modulation (DPCM), and (iv) Adaptive Differential Pulse Code Modulation (ADPCM).

9.13.2 Delta Modulation

In the delta modulation system, the difference between two consecutive pixel intensities is coded by a one-bit quantiser. Delta modulation is the simplest form of DPCM. The block diagram of a delta modulation system is shown in Fig. 9.7.

From Fig. 9.7, it is obvious that the difference between the incoming signal and its predicted value is quantised to 1 bit for transmission. With one bit, the quantiser uses two levels. For delta modulation of images, the signal is generally presented line by line, so it is meaningful to expresss the 2D signal $f(n_1, n_2)$ as the

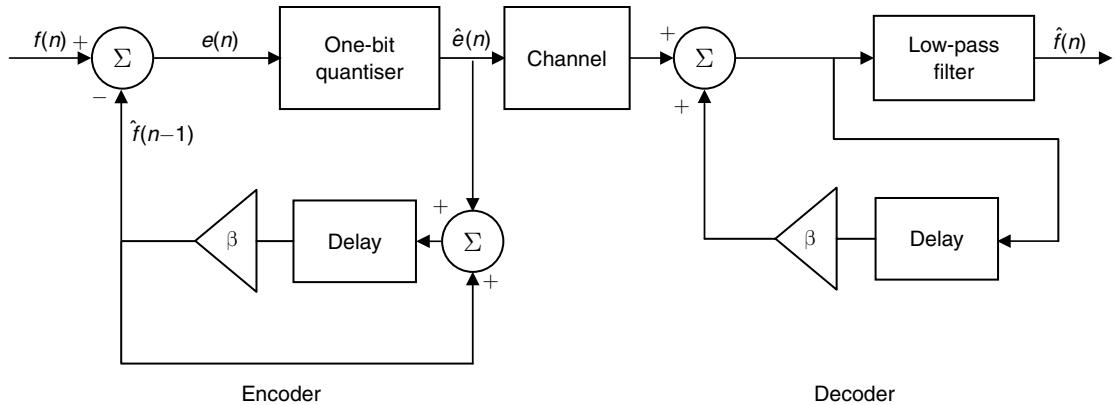


Fig. 9.7 Delta modulation system

one-dimensional signal $f(n)$. The error signal to the one-bit quantiser is the difference between the input signal $f(n)$ and the most recently reconstructed signal $\hat{f}(n-1)$ which is expressed as

$$e(n) = f(n) - \hat{f}(n-1) \quad (9.17)$$

The output of the one-bit quantiser is $\Delta/2$ if $e(n)$ is positive and it is $-\Delta/2$ if $e(n)$ is negative which is represented as

$$\hat{e}(n) = \begin{cases} \Delta/2, & e(n) > 0 \\ -\Delta/2, & \text{otherwise} \end{cases} \quad (9.18)$$

The feedback loop in the encoder is given by the transfer function

$$H(Z) = \frac{\beta Z}{1 - \beta Z} \quad (9.19)$$

The output of the feedback loop is the weighted sum of all past inputs. The transfer function of the decoder loop is given by

$$H(Z) = (1 - \beta Z)^{-1} \quad (9.20)$$

The decoder output is the sum of all past points. The signal can be low-pass filtered or integrated to estimate the input signal. The signal reconstructed at the decoder side is expressed as

$$\hat{f}(n) = \hat{f}(n-1) + \hat{e}(n) \quad (9.21)$$

The fundamental parameters of the delta modulation system are the sampling rate and the quantisation step size. If there is a sudden change in the intensity of the input image then the delta modulator cannot respond immediately to this change, because the quantiser can respond only in several delta steps. This will lead to slope overload error. Slope overload can be reduced by increasing the sampling rate. If the sampling rate increases, the number of samples will be more, and hence the compression ratio achieved will be minimum. The slope overload noise can be minimised by increasing the step size. However, if the step size is increased, the granularity noise increases. The granularity noise is the steplike nature of the output when the input signal is almost constant. The concept of slope overload error and granularity noise is depicted in Fig. 9.8.

The reduction of the granular noise and reduction of slope overload distortion are conflicting requirements, and the step size is chosen through some compromise between the two requirements.

The main advantage of delta modulation is that it does not require a digitiser. This makes it simpler to implement if the input signal that the encoder receives is in analog form.

9.13.3 DPCM

DPCM stands for Differential Pulse Code Modulation. DPCM is a predictive compression technique in which a prediction of the pixel being encoded is subtracted from the current pixel to create a difference value which is then quantised for subsequent transmission. DPCM comes with two flavours: (i) DPCM without quantiser, and (ii) DPCM with quantiser. In the DPCM without quantiser, the reconstructed signal exactly resembles the original signal and it is lossless. However, the compression ratio achieved will be poor. By including the non-linear operator quantiser in the DPCM, the reconstructed signal will not exactly resemble the input signal but it is possible to achieve a high compression ratio.

9.13.4 DPCM Without Quantiser

The block diagram of a DPCM without quantiser is shown in Fig. 9.9. The error between the input and the predictor output is given to the entropy coder. Because of the absence of the quantiser, the reconstructed signal at the receiver end is exactly the same as the transmitted signal.

Example of DPCM Without Quantiser

The step-by-step approach followed in the transmission and reception of the $\begin{bmatrix} 92 & 94 \\ 91 & 97 \end{bmatrix}$ matrix using a DPCM without quantisation technique is given below:

Step 1 Initialisation of the predictor The predictor in the transmission and reception side is set to zero initially. This is illustrated in Fig. 9.10.

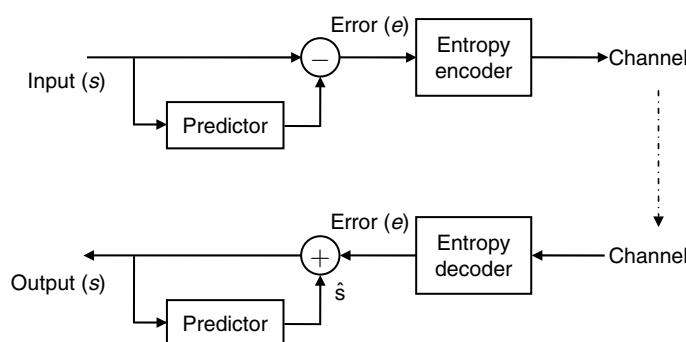


Fig. 9.9 DPCM without quantiser

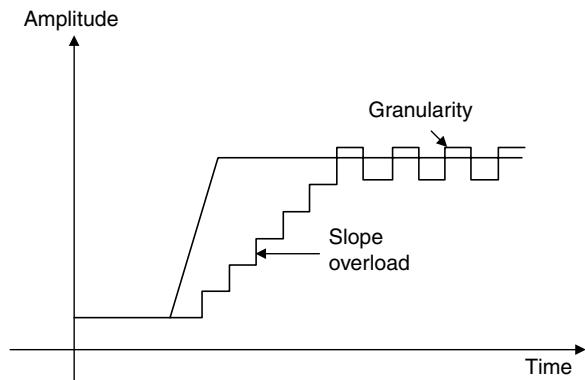


Fig. 9.8 Edge response of delta modulation

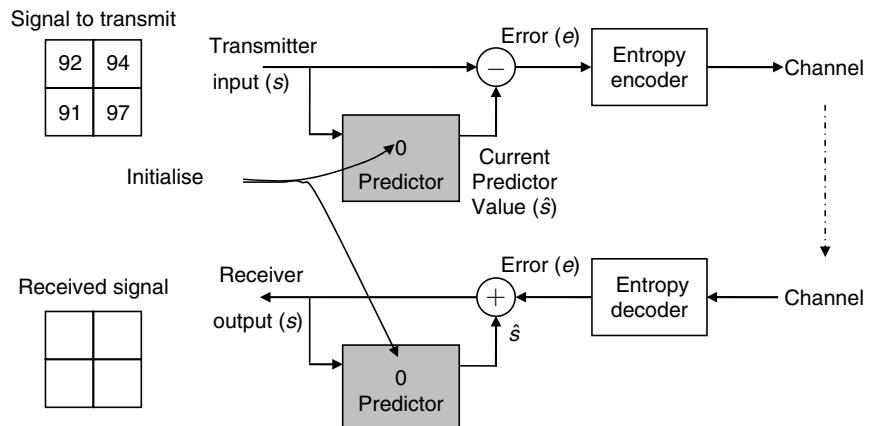


Fig. 9.10 Initialisation of the predictor

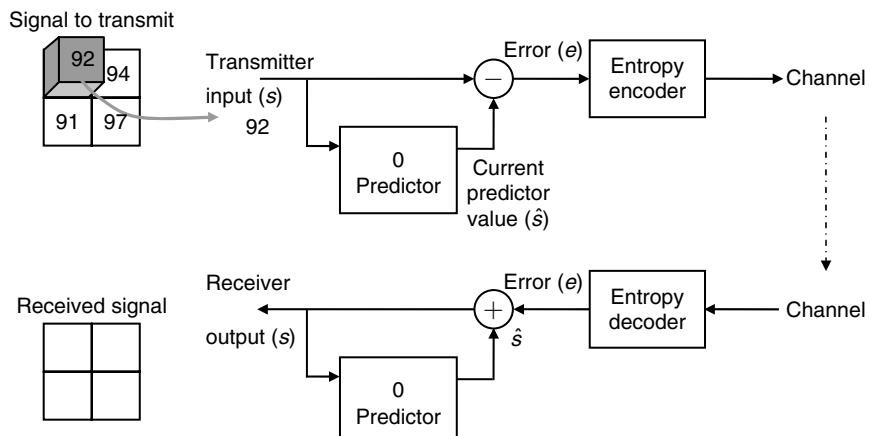


Fig. 9.11 Choosing the first element

Step 2 Choosing the first element in the matrix The first element to be transmitted is 92. This is highlighted in Fig. 9.11.

Step 3 Error computation in transmission side The predictor value is initialised to zero. Then this zero is subtracted from the first element to get $92 - 0 = 92$. This fact is illustrated in Fig. 9.12.

Step 4 Transmission of the error signal While transmitting the value 92 in the channel, the predictor value in the transmitter part gets altered to the value of the previous input 92 which is then transmitted to the receiver part. This is shown in Fig. 9.13.

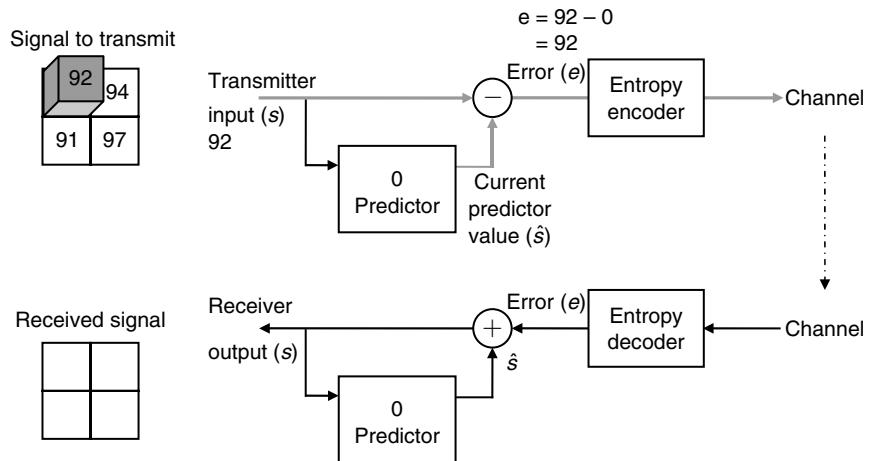


Fig. 9.12 Computation of error in transmission end

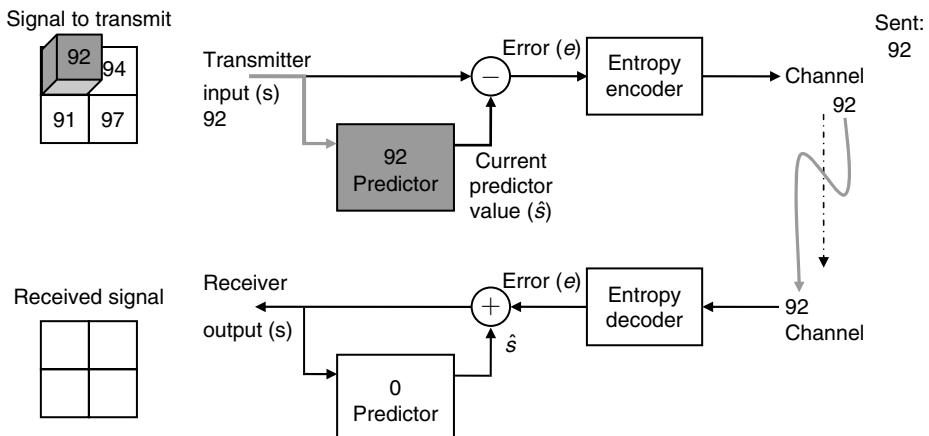


Fig. 9.13 Transmission of error signal

Step 5 Reconstruction of first value in the receiver side Initially, the value stored in the predictor (in the receiver side) is zero. The value in the receiver side is 92. This value (92) is added with the predictor value (0) to get the first output (92) at the receiver end. This concept is depicted in Fig. 9.14(a) and 9.14(b) respectively.

Then, the first element 92 is sent to the channel and received successfully and stored in the receiver side.

Step 6 Choosing the second element in the input matrix The next element 94 in the input matrix is taken into consideration. Then, the predictor value at the transmitter is 92. All these concepts are depicted in Fig. 9.15.

Step 7 Computation of error at the transmission end The value in the predictor (92) is subtracted from the current entry (94) to get the error value which is two. This is shown in Fig. 9.16.

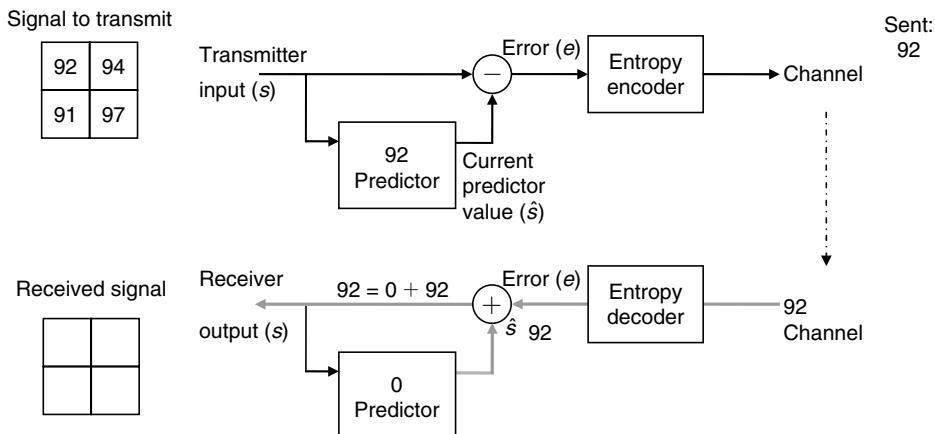


Fig. 9.14(a) Error computation at the receiver end

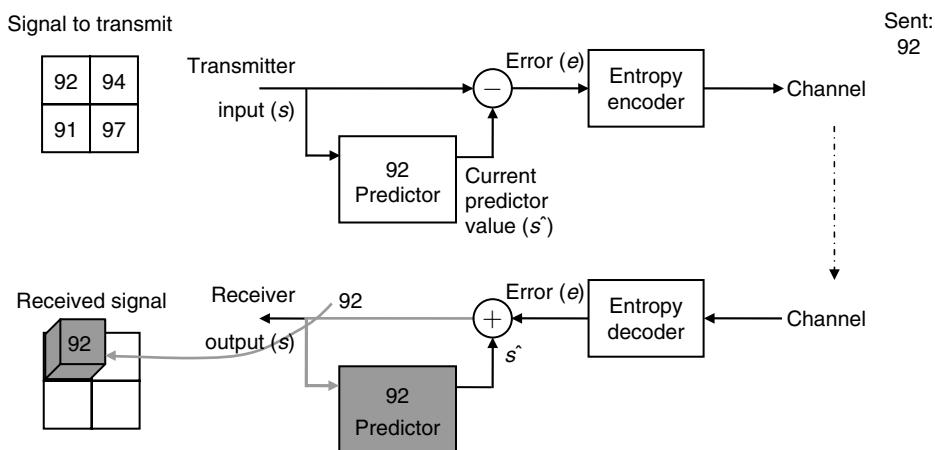


Fig. 9.14(b) Reconstruction of the first value in the receiver end

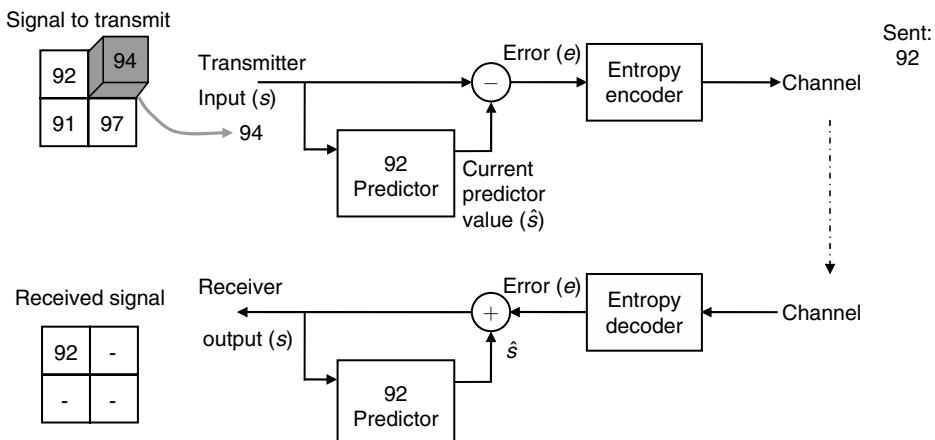


Fig. 9.15 Choosing the second element in the input matrix

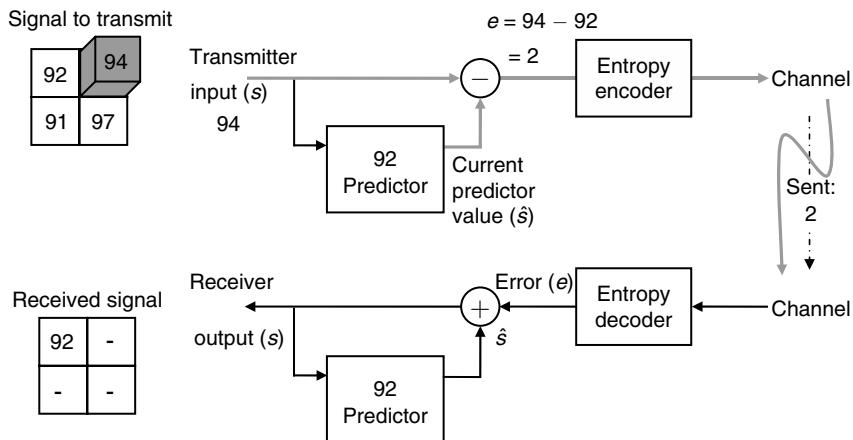


Fig. 9.16 Computation of error in transmission end

Step 8 Reconstruction of the second value in the receiver end The error value that was transmitted through the channel is 2. This value is then added with the predictor value in the receiver end which is 92 to get the second reconstructed value at the receiver end which is 94. This concept is depicted in Fig. 9.17(a) and Fig. 9.17(b) respectively.

This process is repeated successively to transmit and receive further values. In DPCM, we are able to achieve compression by transmitting the error values. Except the first value, successive error values are less than the values to be transmitted. If the magnitude of the error is less, naturally we need fewer bits to represent the error; hence compression is achieved in DPCM.

9.13.5 DPCM with Quantiser

The block diagram of a DPCM with quantiser is illustrated in Fig. 9.18. A quantiser is basically a non-linear operator. The input to the quantiser is the error between the current input and the predictor output. The predictor transforms a set of high entropy but correlated values into a set of low entropy but less correlated values.

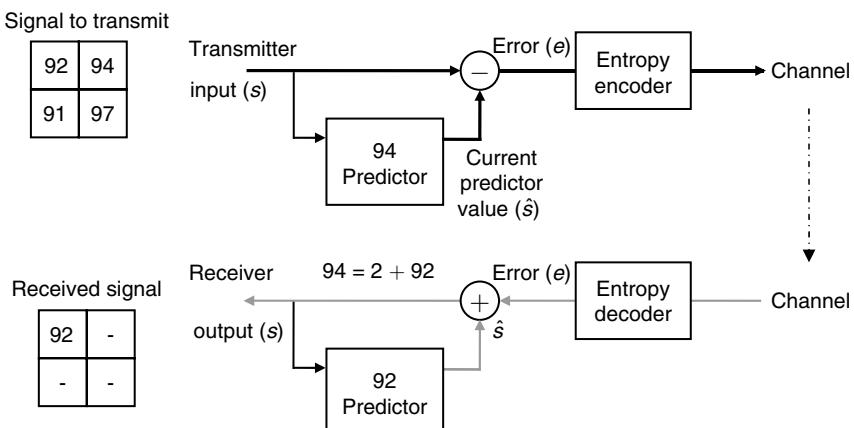


Fig. 9.17(a) Error computation in the receiver end

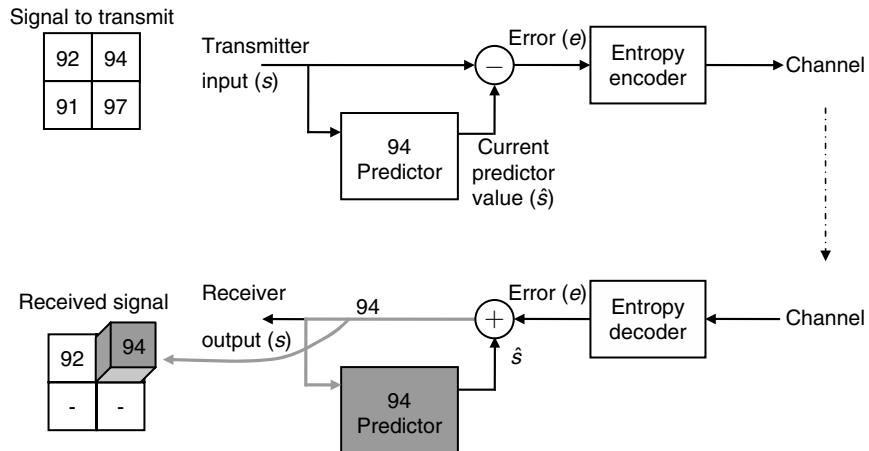


Fig. 9.17(b) Reconstruction of the second value in the receiver end

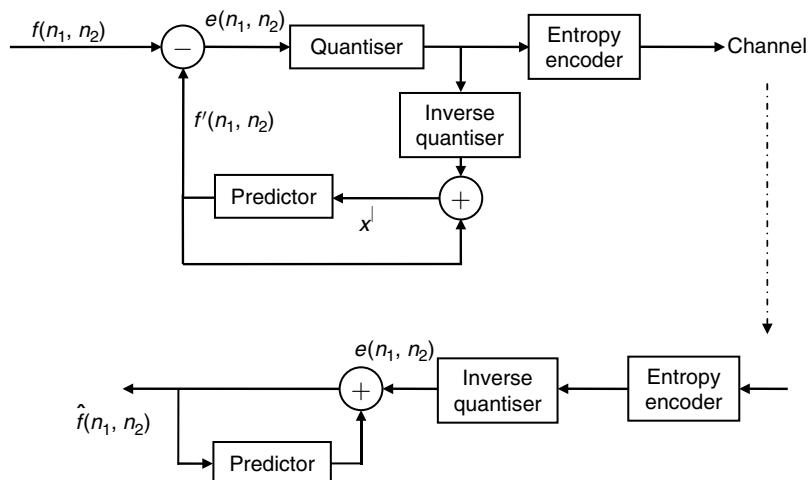


Fig. 9.18 DPCM with quantiser

In delta modulation, only one bit is used to code the error signal, whereas in DPCM, more than one bit is used to code the error.

The input signal to a DPCM coder with quantiser is $f(n_1, n_2)$. To code the current pixel intensity $f(n_1, n_2)$, $f(n_1, n_2)$ is predicted from previously reconstructed pixel intensities. The predicted value is denoted by $f'(n_1, n_2)$. The error signal is the difference between $f(n_1, n_2)$ and $f'(n_1, n_2)$ and is fed to the quantiser. The output of the quantiser is denoted by $\hat{e}(n_1, n_2)$. The reconstructed signal at the receiver end is $\hat{f}(n_1, n_2)$ which will not be exactly the same as the input signal $f(n_1, n_2)$ due to the quantiser.

Example of a DPCM with Quantiser Let us consider the concept of DPCM by including the quantiser in the transmission end. Whenever a quantiser is incorporated in the DPCM technique, loss of information is inevitable. The general block diagram of DPCM with quantiser with the input values to be transmitted is given in Fig. 9.19.

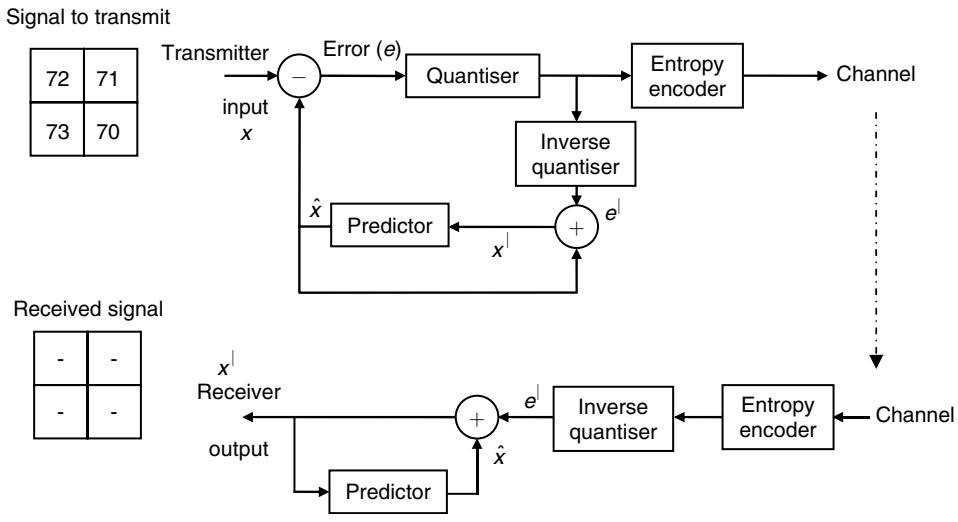


Fig. 9.19 DPCM with quantiser

Step 1 Initialisation of the predictor Initially, the predictor in the transmitter and receiver side is set to zero. This is illustrated in Fig. 9.20.

Step 2 Choosing the first element in the input matrix The first element in the input matrix 72 is to be transmitted first. This is highlighted in Fig. 9.21.

Step 3 Computation of error in the transmission end The input value is subtracted from the predictor value (0) to get the error value. In this case, the error value is 72. This is illustrated in Fig. 9.22.

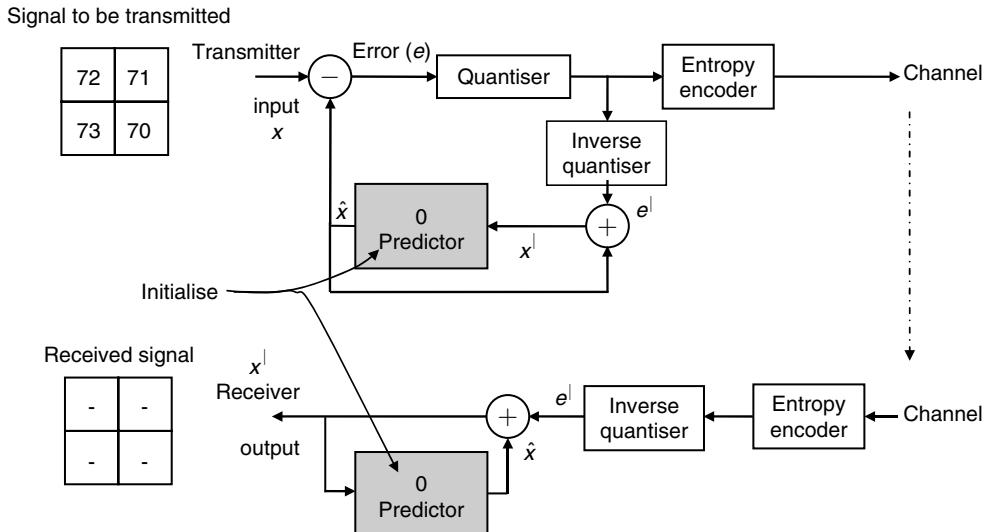


Fig. 9.20 Initialisation of the predictor

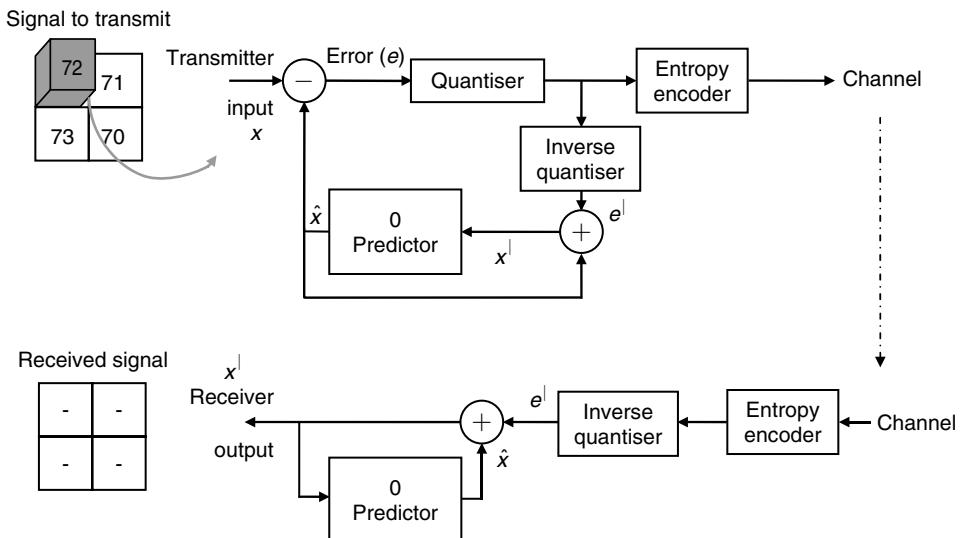


Fig. 9.21 Choosing the first element in the input matrix

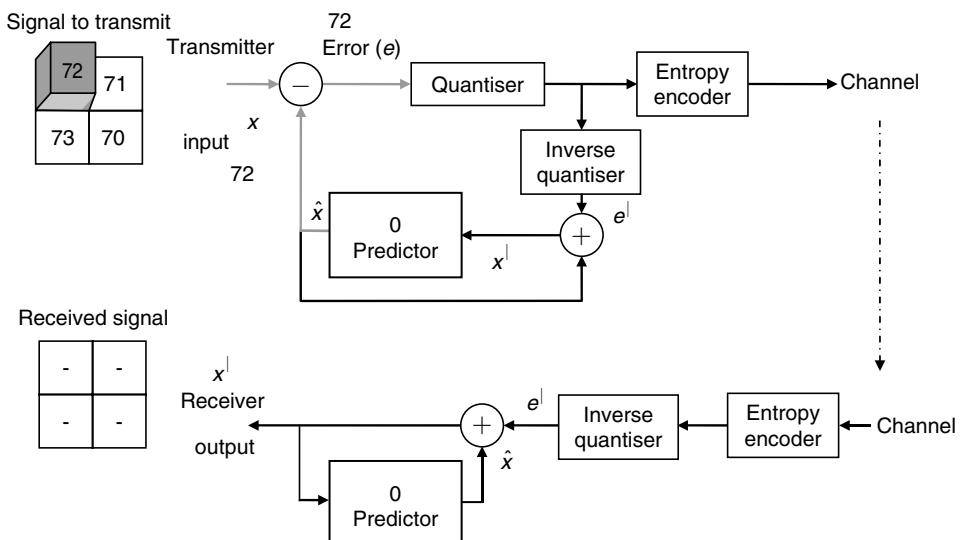


Fig. 9.22 Computation of error value in the transmission end

Step 4 Quantisation of the error value The error value (72) is then quantised. The operation used to perform quantisation is a rounding operation which is shown in Fig. 9.23.

Step 5 Performing inverse quantisation step in the transmission end The inverse quantisation is performed before the value 36 is transmitted through the channel. On performing inverse quantisation, the value 36 is changed to 72 which is depicted in Fig. 9.24.

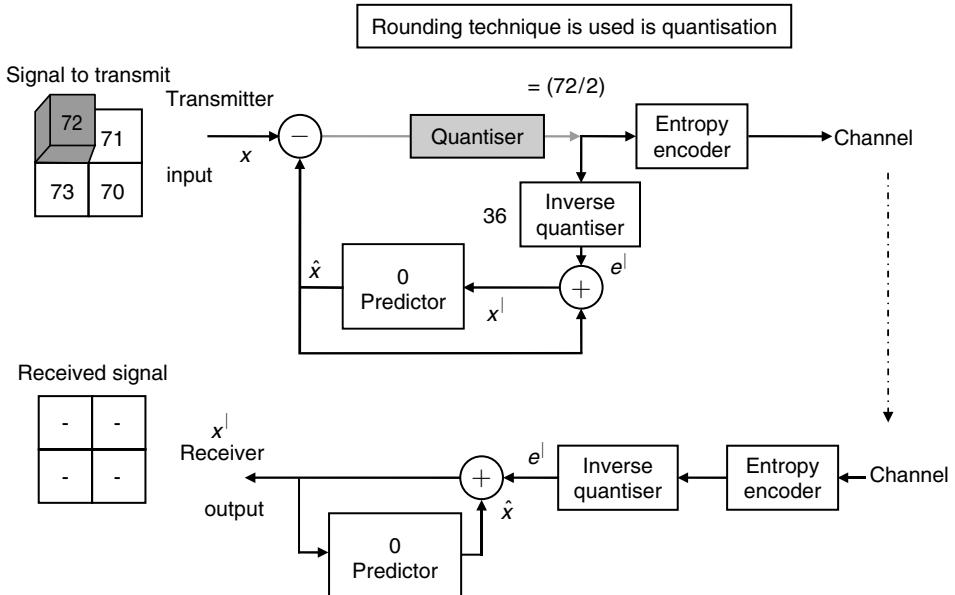


Fig. 9.23 Quantisation of the error value

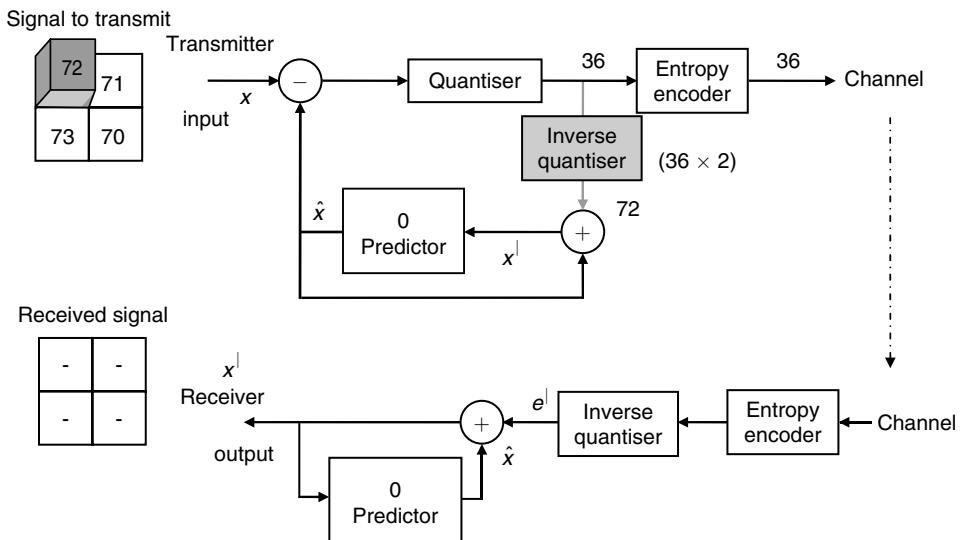


Fig. 9.24 Inverse quantisation in the transmitter side

Step 6 *Updation of predictor value in the transmitter side* The output of the inverse quantiser (72) is added with the initial value of the predictor (0) to get the value 72. Then, the predictor value is 72 which is shown in Fig. 9.25.

Step 7 *Transmission of first value through the channel* After updating the value of the predictor in the transmitter side, the first value (36) is transmitted through the receiver. This is illustrated in Fig. 9.26.

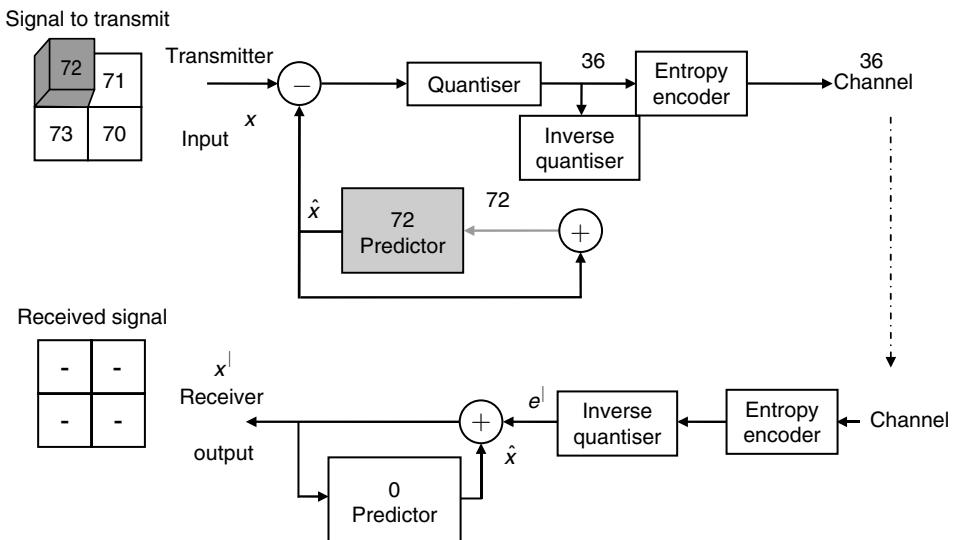


Fig. 9.25 Updation of predictor value in transmitter side

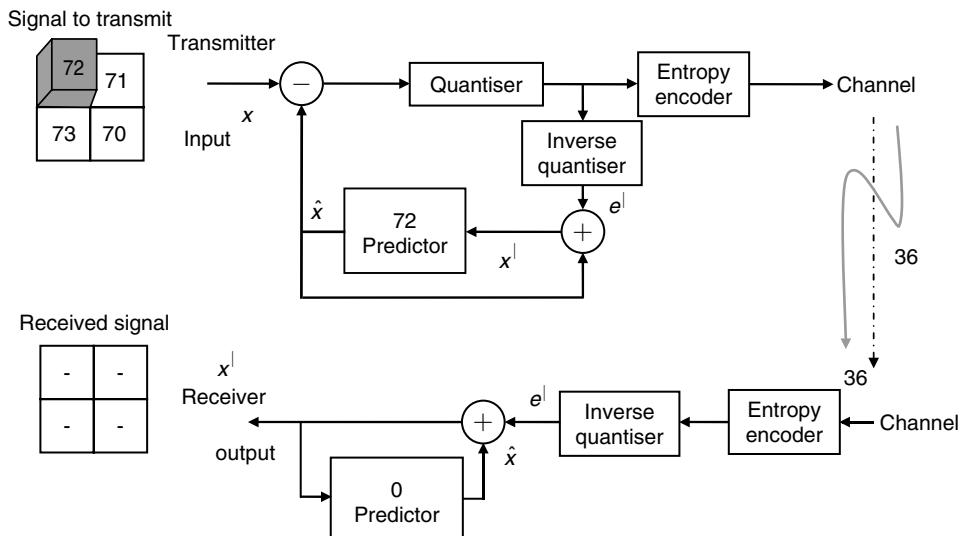


Fig. 9.26 Transmission of first value through the channel

Step 8 Inverse quantisation in the receiver end Once the value 36 reaches the receiver end, inverse quantisation is performed to get the value 72. This is shown in Fig. 9.27.

Step 9 Reconstruction of first value in the receiver The value obtained after performing inverse quantisation is added to the value in the predictor (0) to get the value 72. Then, 72 is the first reconstructed value. This is shown in Fig. 9.28(a) and 9.28(b) respectively.

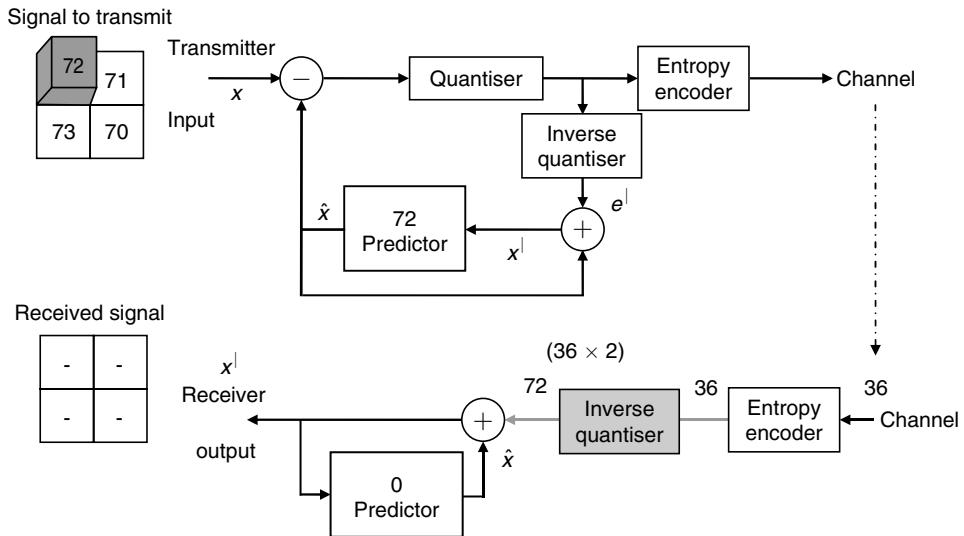


Fig. 9.27 Inverse quantisation at the receiver end

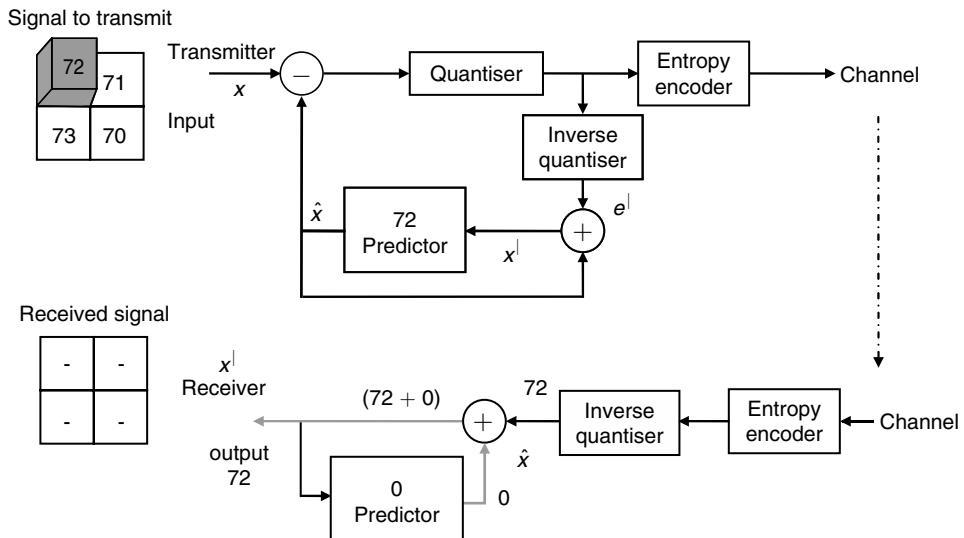


Fig. 9.28(a) Reconstruction of the first value

Step 10 Choosing the second element in the input signal The second element in the input matrix (71) is to be transmitted next. This is highlighted in Fig. 9.29.

Step 11 Computation of error in the transmitter side The error between the second element of the matrix and the predicted output is calculated. This error value is then sent to the quantiser. This step is illustrated in Fig. 9.30.

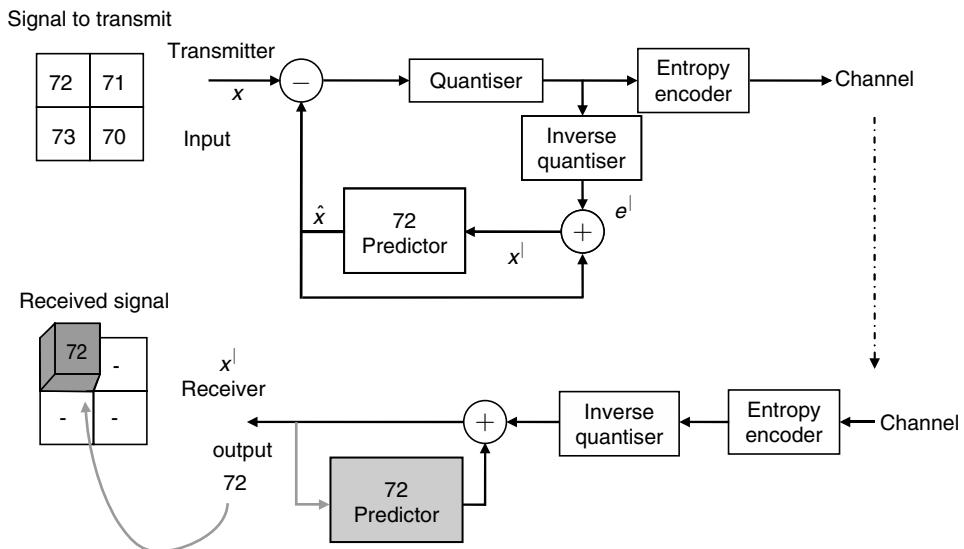


Fig. 9.28(b) First received signal at the receiver side

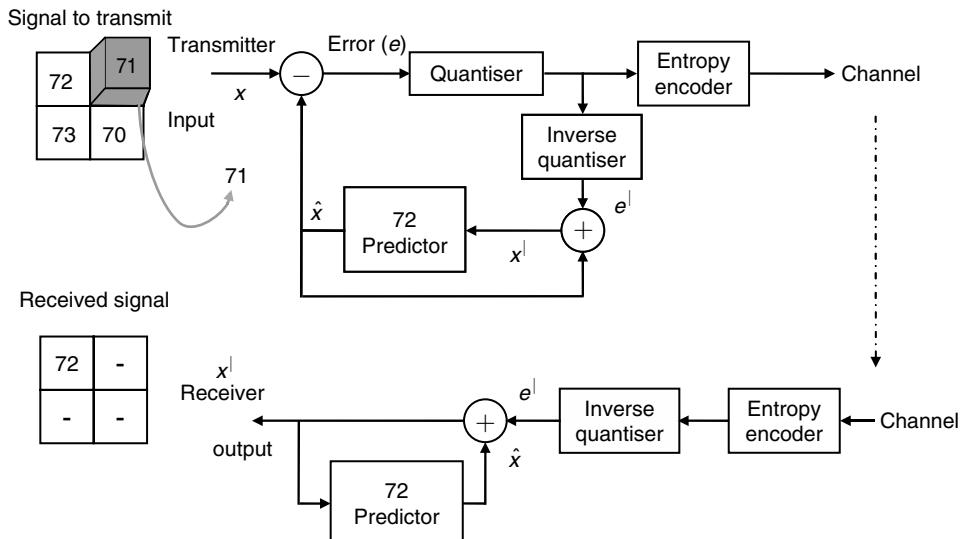


Fig. 9.29 Choosing the second element in the input matrix

Step 12 *Rounding of the error signal* The error signal obtained in the previous step is then rounded to get 0. This is illustrated in Fig. 9.31.

Step 13 *Performing inverse quantisation at the transmitter side* The null value obtained in the previous step is inverse quantised to get a null value. This is shown in Fig. 9.32.

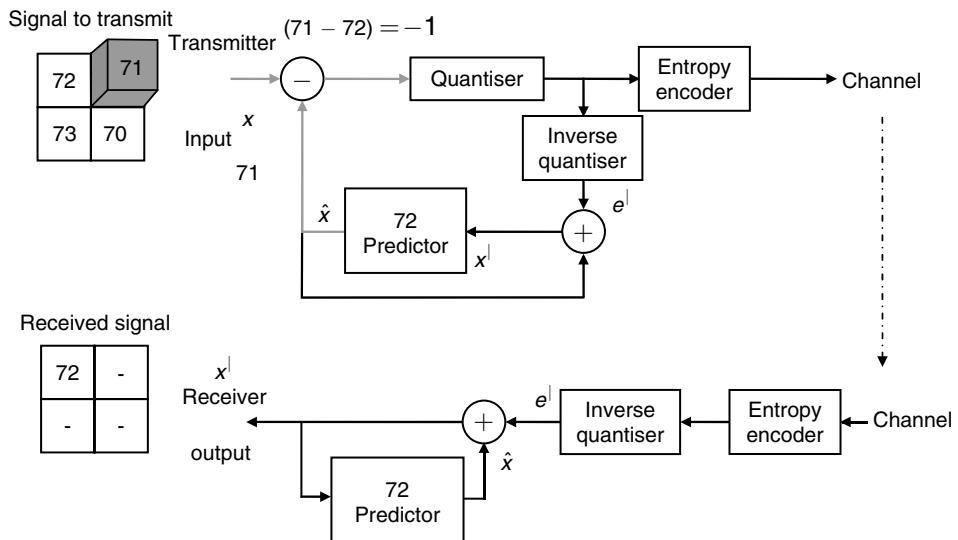


Fig. 9.30 Computation of error in the transmitter side

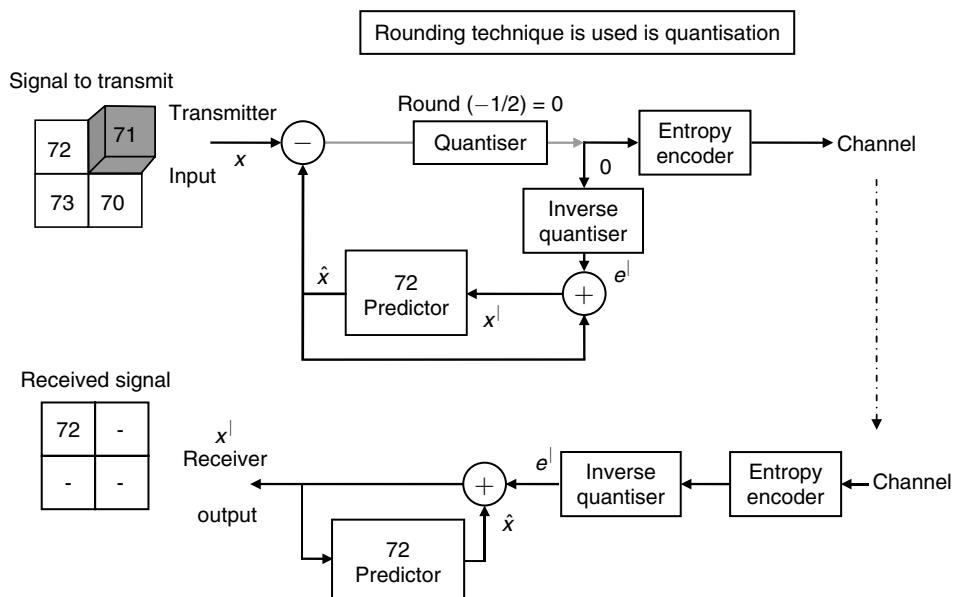


Fig. 9.31 Rounding of the error signal at the transmitter side

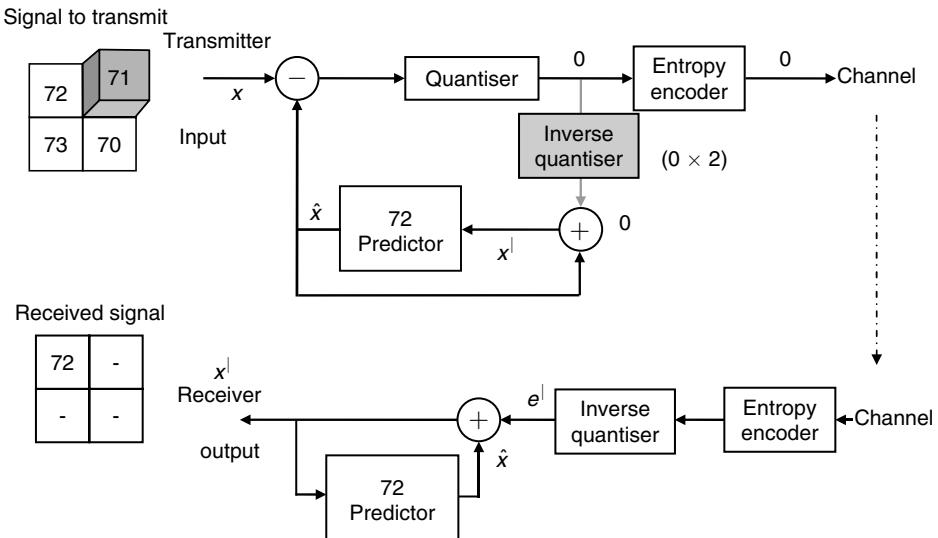


Fig. 9.32 Inverse quantisation at the transmitter side

Step 14 *Updation of predictor value at the transmitter side* In this step, the predictor value is updated to be 72, which is pictorially represented in Fig. 9.33.

Step 15 *Transmission of the quantised error signal through the channel* The illustration of the transmission of the second quantised error value (0) through the channel is shown in Fig. 9.34.

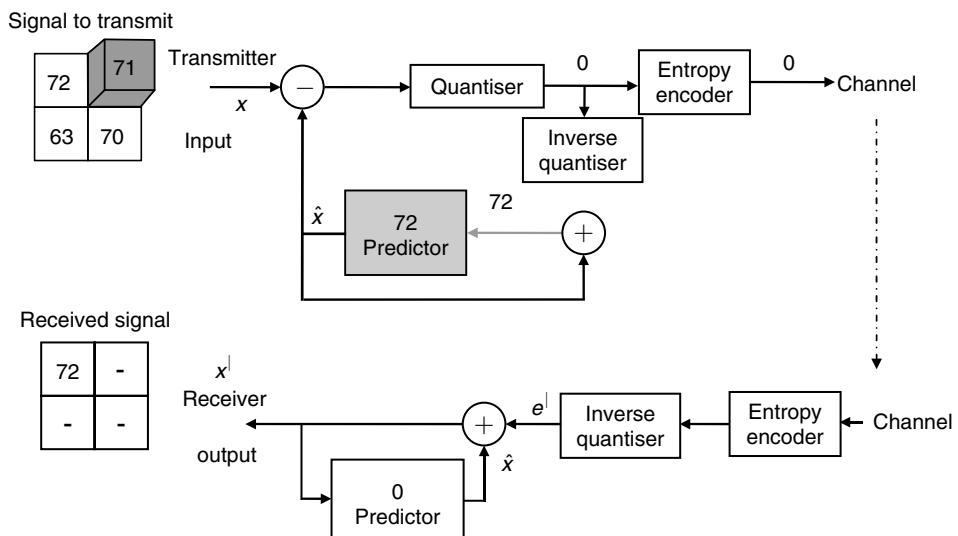


Fig. 9.33 Updation of predictor value at the transmitter

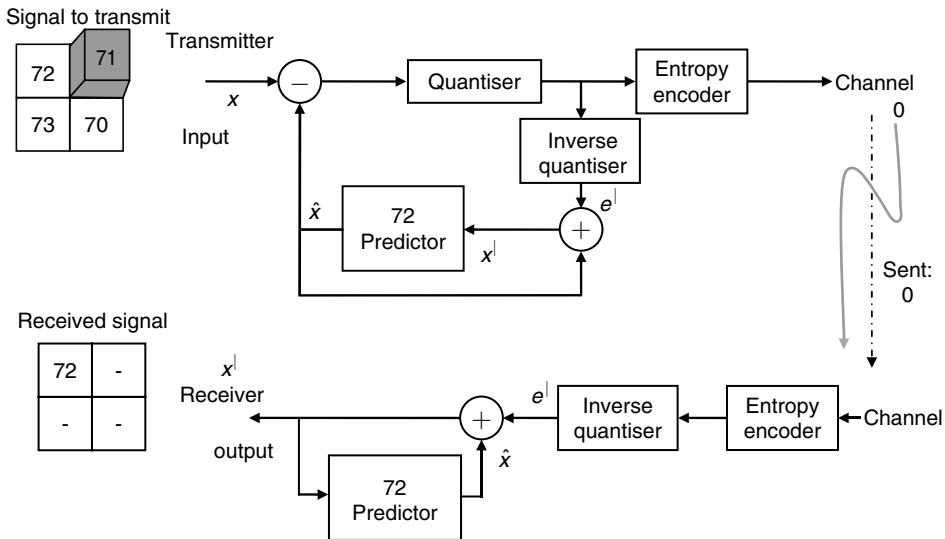


Fig. 9.34 Transmission of the second quantised error value

Step 16 Inverse quantisation of the received input At the receiver side, inverse quantisation of the received input is performed which is shown in Fig. 9.35. As the entropy encoder and decoder are perfectly lossless, the impact of entropy encoder and decoder is not highlighted in this example.

Step 17 Reconstruction of the second transmitted value The output of the inverse quantiser is then added with the predictor value at the receiver side to get the second reconstructed value. This is illustrated in Fig. 9.36(a) and 9.36(b) respectively.

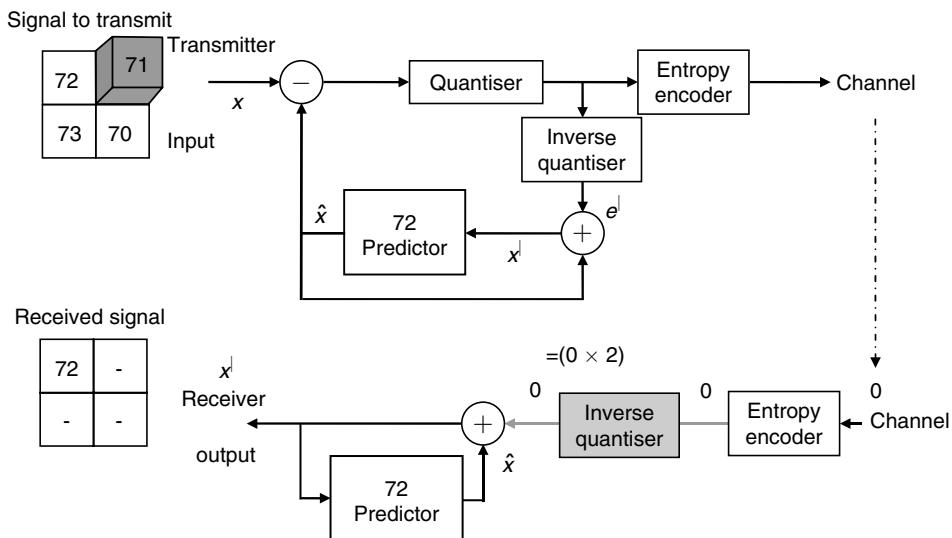


Fig. 9.35 Inverse quantisation at the receiver

From Fig. 9.36(b), it is clear that the reconstructed value is 72, but the transmitted value is 71. Hence error is inevitable if there is a quantisation scheme incorporated in DPCM technique. This process is repeated twice to transmit as well as receive the next two values.

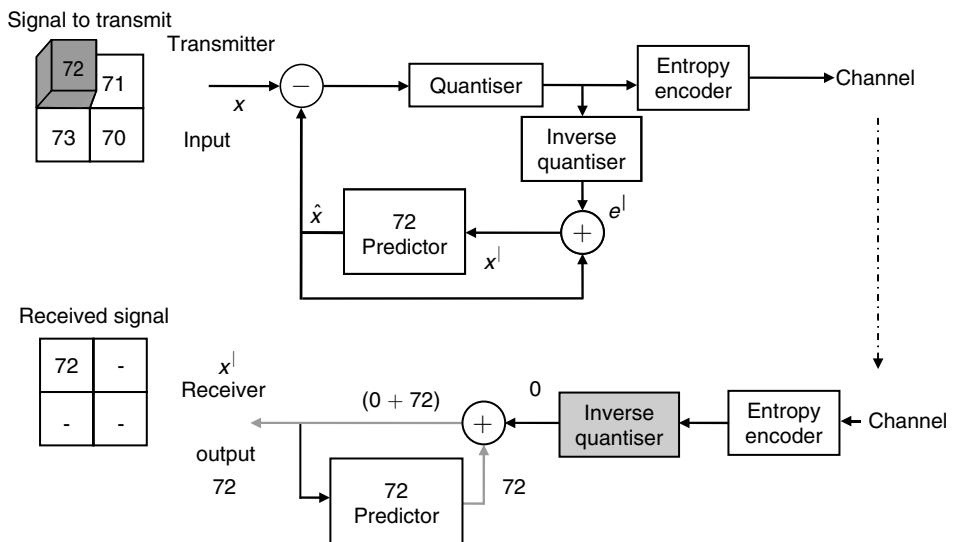


Fig. 9.36(a) Updation of the error value at the receiver side

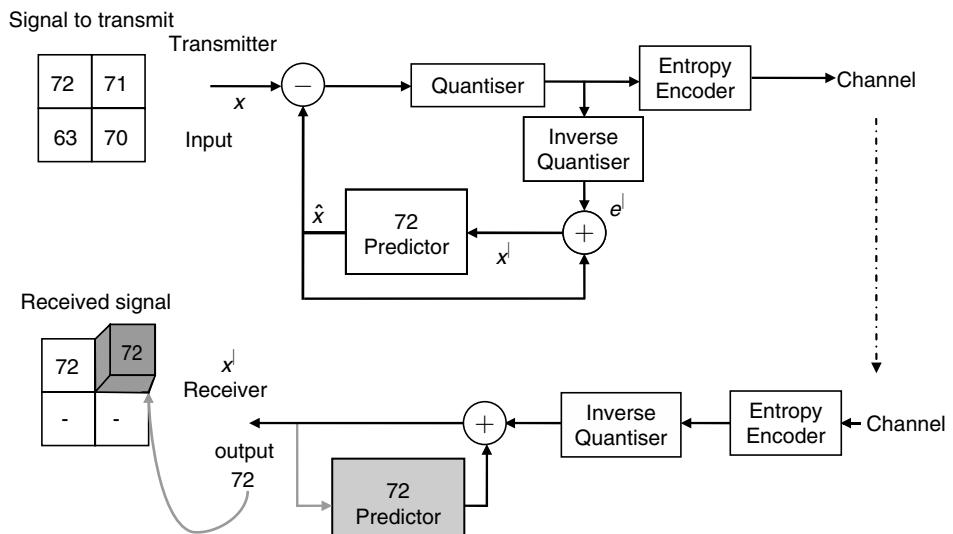


Fig. 9.36(b) Reconstruction of the second value in the receiver end

9.14 TRANSFORM-BASED COMPRESSION

Image-transform coding is the most popular method used in image-coding applications. An image is linearly transformed to produce a set of linear transform coefficients, which are usually scalar-quantised and entropy-coded for transmission. Thus, transform coding is a mathematical operation that converts a large set of highly correlated pixels into a smaller set of uncorrelated coefficients. Transform coding relies on the fact that pixels in an image exhibit a certain level of correlation with their neighbouring pixels. Transformation is a very useful tool in image compression. It is used to transform the image data in time domain to frequency domain. By transforming the data into frequency domain, the spatial redundancies in time domain can be minimised. The advantage of using transformation is that the energy of the transformed data is mainly condensed in the low-frequency region, and is represented by a few transform coefficients. Thus, most of these coefficients can be discarded without significantly affecting the reconstructed image quality. In a transform coder, the discrete data signal is segmented into non-overlapping blocks, and each block is expressed as a weighted sum of discrete basis functions. The purpose of transform coding is to decompose the correlated signal samples into a set of uncorrelated transform coefficients, such that the energy is concentrated into as few coefficients as possible.

The block diagram of transform-based image coding scheme is shown in Fig. 9.37.

Transformation The transformation ‘reorganises’ the gray values of the image and thus changes the correlation properties of the image. The transformation compacts the image information into a small number of coefficients. For efficient compression, the transform should have the following properties:

1. Decorrelation
2. Linearity
3. Orthogonality

Decorrelation The transform should generate less correlated or uncorrelated transform coefficients to achieve high compression ratio.

Linearity Linearity principle allows one-to-one mapping between pixel values and transform coefficients.

Orthogonality Orthogonal transforms have the feature of eliminating redundancy in the transformed image.

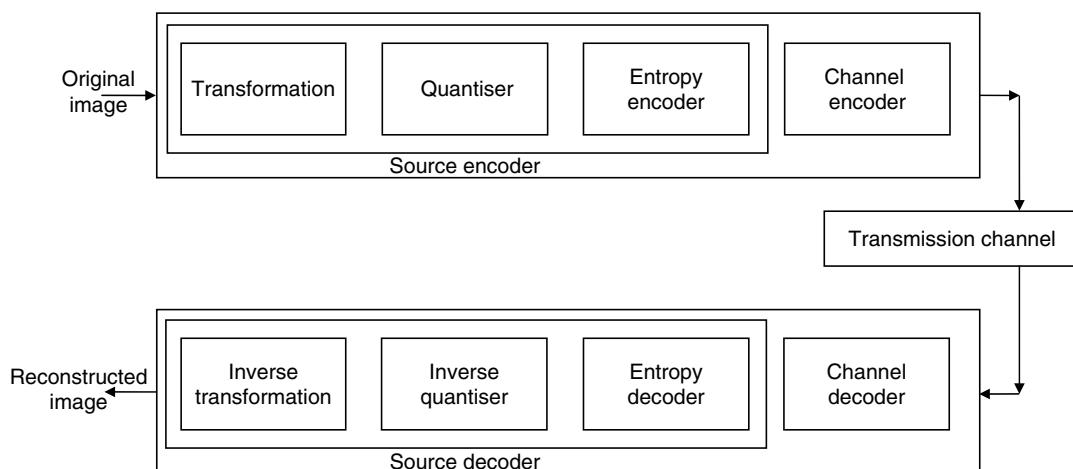


Fig. 9.37 Transform-based image-coding scheme

Quantisation Quantisation is the process of reducing the number of possible values of a quantity, thereby reducing the number of bits needed to represent it. The image transforms by themselves do not achieve any compression. Instead, they represent the image in a different domain in which the image data is separated into components of varying importance. Quantisation is basically an irreversible process.

Entropy Encoder The purpose of an entropy encoder is to reduce the number of bits required to represent each symbol at the quantiser output. Commonly used entropy-coding techniques are Huffman coding, arithmetic coding, run-length coding, etc. Entropy coding is basically a lossless coding scheme.

The output of the encoder is basically a bit stream which is transmitted through the channel to the decoder. The channel is usually assumed to be lossless if one concentrates on source coding alone. If one is interested in joint source and channel coding then the error occurs when the bitstream passes through the channel, and this should also be taken into account. The decoder basically performs the reverse process of the encoder to get the reconstructed image.

9.15 IMAGE-COMPRESSION STANDARD

The objective of an image-compression standard is to enhance the compatibility and interoperability among systems manufactured by different vendors. Since the mid 1980's, members from both ITU and ISO have been working together to establish a joint international standard for the compression of grayscale and colour still images. This effort has been known as JPEG, the Joint Photographic Experts Group. Officially, JPEG corresponds to the ISO/IEC international standard 10928-1, digital compression and coding of continuous-tone still images. After evaluating the number of coding schemes, the JPEG members selected the discrete cosine transform (DCT) based method in 1988. From 1998 to 1990, the JPEG group continued its work by simulating, testing and documenting the algorithm. JPEG became the Draft International Standard (DIS) in 1991 and International Standard (IS) in 1992. The JPEG standard was designed for compression and recovery of still photographic images. The standard provides a relatively good compression ratio. The most commonly used transform for image compression is the discrete cosine transform. One of the major reasons for its popularity is its selection as the standard for Joint Photographic Expert Group (JPEG). The JPEG standard was designed in 1992 for compression and reconstruction of still photographic images. The JPEG standard describes a family of image-compression techniques for still images. Different modes such as sequential, progressive and hierarchical modes, and options like lossy and lossless modes of the JPEG standards exist. The JPEG standard employs block-based DCT coding. The meaning of block-based technique is that the transform is not applied to the entire image at a stretch, but it is applied over fixed blocks each of size usually of 8×8 or 16×16 . The basic encoder and decoder structures used in JPEG standard are illustrated in Fig. 9.38.

In Fig. 9.38, FDCT stands for Forward Discrete Cosine Transform, and IDCT stands for Inverse Discrete Cosine Transform. From the figure, it is obvious that the input image is partitioned into an 8×8 sub-block. The FDCT is computed on each of the 8×8 blocks of pixels. The coefficient with zero frequency in both dimensions is called the 'DC coefficient' and the remaining 63 coefficients are called the 'AC coefficients'. The FDCT processing step lays the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. The 64 DCT coefficients are scalar-quantised using uniform quantisation tables based upon psychovisual experiments. After the DCT coefficients are quantised, the coefficients are ordered according to the zigzag scan as shown in Fig. 9.39. The purpose of zigzag scanning is based upon the observation that most of the high-frequency coefficients are zero after quantisation.

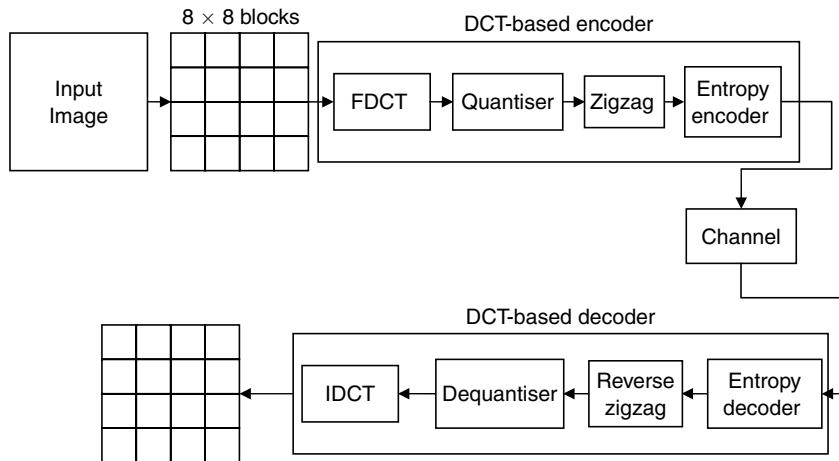


Fig. 9.38 Baseline JPEG encoder and decoder

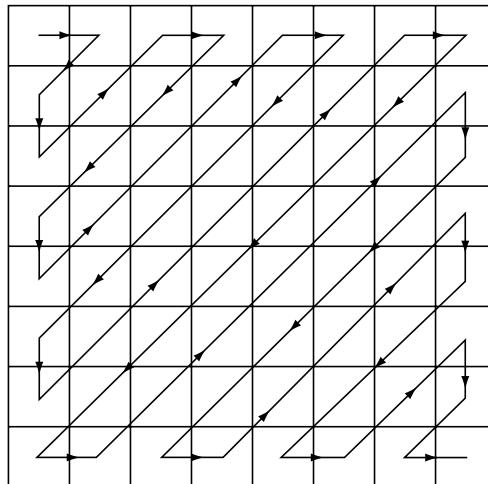


Fig. 9.39 Zigzag scan

The resulting bit stream is run-length coded to generate an intermediate symbol sequence, and then these symbols are Huffman coded for transmission or storage.

To perform JPEG decoding, the reverse process of coding is applied, that is, the encoded data stream is entropy decoded and passed through a reverse zigzag table in order to generate a 2D array. This array is then rescaled using the quantising factor and then passed through Inverse DCT.

Reasons for Block Processing The reason for taking DCT in block manner is given below:

1. Taking DCT for an entire image requires large memory, hence it is not preferable.
2. Taking DCT for an entire image is not a good idea for compression due to spatially varying statistics within an image.

An N -point DCT consists of N real basis vectors with cosine functions. There are four types of DCT, namely, DCT-I, DCT-II, DCT-III and DCT-IV, derived according to the varying boundary conditions. The DCT-II form is commonly used in image and video coding. The 2D DCT is performed by applying 1D DCT on rows and columns separately. The DCT is an orthogonal transform. The DCT possesses good variance distribution which leads to efficient energy compaction in the transform domain. The role of DCT is to decompose the original signal into its dc and ac components and the role of inverse DCT is to reconstruct the signal.

The 2D DCT of the input image $f(m, n)$ is given by $F(k, l)$ as

$$F(k, l) = \alpha(k)\alpha(l) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) \cos\left(\frac{(2m+1)\pi k}{2N}\right) \cos\left(\frac{(2n+1)\pi l}{2N}\right) \quad (9.22)$$

where

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } k = 0 \\ \sqrt{\frac{2}{N}} & \text{for } k = 1, 2, \dots, N-1 \end{cases} \quad (9.23)$$

and

$$\alpha(l) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } l = 0 \\ \sqrt{\frac{2}{N}} & \text{for } l = 1, 2, \dots, N-1 \end{cases} \quad (9.24)$$

Advantages of Block Processing The advantages of block processing are summarised below:

1. The reason for block processing is the transform of small blocks is much easier to compute than the complete image.
2. The pixel correlation will not exceed usually 16 or 32 pixels. But this assumption is not always valid. Moreover, the correlation between pixels is less between distant pixels than between neighbouring pixels.

Drawback of Block Processing One of the drawbacks of DCT-based transform coding is the introduction of block artifacts. Blocking artifacts are caused by the discontinuities that result from the rectangular windowing of the image data.

Minimisation of Blocking Artifact

The blocking artifact can be minimised by the use of

- (i) overlapping blocks,
- (ii) low-pass filtering of boundary pixels, and
- (iii) lapped Orthogonal Transform (LOT)

The use of overlapping blocks results in increased bit rate, and hence higher compression ratio cannot be achieved. Low-pass filtering of boundary pixels will result in blurring.

Zonal Coding Zonal coding is based on the fact that the transform coefficients of maximum variance carry the most picture information. The locations of the coefficients with the K largest variances are indicated by means of a zonal mask, which is the same for all blocks. All transform coefficients in the zone are retained, while all coefficients outside the zone are set to zero. To design a zonal mask, variances of each coefficient can be calculated based on a global image model, such as the Gauss–Markov model. Zonal masks can be customised for individual images, provided that they are stored with the encoded image data.

Bit Allocation in Zonal Coding The dynamic range of the retained coefficients will vary; hence the number of bits allotted to represent each coefficient should be proportional to the dynamic range of the respective coefficient. A simple bit-allocation strategy is to choose the number of bits proportional to the variance of each coefficient over the ensemble of blocks. If the number of retained coefficients are M with the variances σ_i^2 , $i = 1, 2, \dots, M$, then the number of bits allocated for each of these coefficients is given by

$$b_i = \frac{B}{M} + \frac{1}{2} \log_2 \sigma_i^2 - \frac{1}{2M} \sum_{i=1}^M \log_2 \sigma_i^2 \quad (9.25)$$

where B is the total number of bits available to represent a block. Figure 9.40 (a) and (b) shows the zonal mask and zonal bit allocation.

Threshold Coding In most images, different blocks have different spectral and statistical characteristics, necessitating the use of adaptive bit-allocation methods. In threshold coding, each transform coefficient is compared with a threshold. If it is smaller than the threshold then it is set to zero. If it is larger than the threshold, it will be retained for quantisation and encoding. The thresholding method is an adaptive method where only those coefficients whose magnitudes are above a threshold are retained within each block.

Quantiser The purpose of quantisation is to remove the components of the transformed data that are unimportant to the visual appearance of the image and to retain the visually important components. The quantiser utilises the fact that the human eye is unable to perceive some visual information in an image. Such information is deemed redundant and can be discarded without noticeable visual artifacts. Such redundancy is referred as psychovisual redundancy. The process of assigning a particular sample to a particular level is called quantisation. Quantisation involves representation of a set of fine-resolution data by a coarse-resolution approximation.

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

(a)

8	7	6	4	3			
7	6	5	4				
6	5	4					
4	4						
3							

(b)

Fig. 9.40 Illustration of (a) zonal mask, and (b) zonal bit allocation

The quantisation operation is non-linear and irreversible; hence, it is a lossy compression scheme. A simple example of quantisation is the rounding of a real value to an integer value. Rounding the number to the nearest integer can be represented with minimum number of bits.

Entropy Encoder The purpose of an entropy encoder is to reduce the number of bits required to represent each symbol at the quantiser output.

JPEG Modes The JPEG standard supports different modes. Some of the commonly used modes are

- (i) Sequential mode
- (ii) Progressive mode
- (iii) Hierarchical mode
- (iv) Lossless mode

Sequential Mode The sequential JPEG compression consists of a forward DCT transform, a quantiser, and an entropy encoder, while decompression starts with entropy decoding followed by dequantising and inverse DCT. The sequential mode is the default JPEG mode.

Progressive Mode The JPEG standard also facilitates progressive image transmission. The progressive compression mode delivers low-quality versions of the image followed by higher quality passes. This mode is widely supported in web browsers. In the progressive compression mode, a sequence of scans is produced. Such multiple scans of images are useful when the speed of the communication line is low. Three algorithms are defined as part of the JPEG progressive compression standard: (i) progressive spectral selection, (ii) progressive successive approximation, and (iii) combined progressive algorithm. In the progressive spectral selection approach, the dc coefficients are transmitted first, followed by groups of low-frequency and high-frequency coefficients. In the progressive successive approximation, all DCT coefficients are sent first with lower precision, and their precision is increased as additional scans are transmitted. The combined progressive algorithm uses both the above principles together. The advantage of progressive compression mode is that the user has a choice whether to continue receiving the image data after the first scan.

Hierarchical Mode The hierarchical JPEG mode encodes the image in a hierarchy of several different resolutions. In the hierarchical mode, the decoded images can be displayed either progressively or at different resolutions. A pyramid of images is created and each lower-resolution image is used as a prediction for the next higher-resolution pyramid level. The encoding procedure in hierarchical JPEG is summarised as follows:

- (i) Filter and down-sample the original image by the desired number of multiples of two in each dimension.
- (ii) Encode this reduced size image using one of the sequential DCT, progressive DCT, or lossless encoders.
- (iii) Decode this reduced-size image and then interpolate and up-sample it by two horizontally and/or vertically.
- (iv) Use this up-sampled image as a prediction of the original at this resolution, and encode the difference image using one of the sequential DCT, progressive DCT, or lossless encoders.
- (v) Repeat steps (iii) and (iv) until full resolution of the image has been encoded.

Hierarchical encoding is useful in applications in which a very high-resolution image must be accessed by a lower-resolution display.

Lossless Mode The lossless mode of the JPEG compression uses a simple predictive compression algorithm and Huffman coding to encode the prediction differences. It is rarely used, since its compression ratio is very low when compared to lossy modes. This mode seems to be useful in the case of medical image compression where loss of information is not tolerable.

Compression Metrics The type of compression metrics used for digital data varies depending on the type of compression employed. The basic metric used to evaluate the performance of the compression algorithm is the *compression ratio*. The simple formula to compute the compression ratio expressed in percentage is given as

$$\text{Compression ratio}(\%) = \frac{\text{Output file size (bytes)}}{\text{Input file size (bytes)}} \quad (9.26)$$

Irrespective of the type of digital media being compressed, the compression ratio given in Eq. (9.26) gives a clear indication of the compression achieved.

Rate is another metric used to evaluate the performance of the compression algorithm. Rate gives the number of bits per pixel used to encode an image rather than some abstract percentage. The formula for rate in bits-per-pixel (bpp) is given below

$$\text{Rate(bpp)} = \frac{8 \times (\text{output file size (bytes)}}{\text{Input file size (bytes)}} \quad (9.27)$$

A typical grayscale 8-bit image has a rate of 8 bpp.

Performance Index The performance index of image compression can be broadly classified into (a) objective fidelity criteria, and (b) subjective fidelity criteria.

(a) *Objective Fidelity Criteria* The mean square error between the original image $f(m, n)$ and the reconstructed image $g(m, n)$ is given by

$$\text{MSE} = \frac{1}{M \times N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (f(m, n) - g(m, n))^2 \quad (9.28)$$

Here, $M \times N$ represents the size of the image.

The mean square error is a very useful measure as it gives an average value of the energy lost in the lossy compression of the original image f . A human observing two images affected by the same type of degradation will generally judge the one with smaller MSE to be closer to the original. A very small MSE can be taken to mean that the image is very close to the original. However, the MSE has some problems when images with different types of degradation are compared. Signal-to-noise ratio (SNR) is another measure often used to compare the performance of reproduced images which is defined as

$$\text{SNR} = 10 \times \log_{10} \left[\frac{\frac{1}{M \times N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)^2}{\frac{1}{M \times N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (f(m, n) - g(m, n))^2} \right] \quad (9.29)$$

SNR is measured in DB's and gives a good indication of the ratio of signal-to-noise reproduction. A more subjective qualitative measurement of distortion is the Peak Signal-to-Noise Ratio (PSNR).

$$\text{PSNR} = 10 \times \log_{10} \left[\frac{(2^b - 1)^2}{\frac{1}{M \times N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (f(m, n) - g(m, n))^2} \right] \quad (9.30)$$

For an 8-bit image, $b = 8$ which gives the PSNR value as

$$\text{PSNR} = 10 \times \log_{10} \left[\frac{(255)^2}{\frac{1}{M \times N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (f(m, n) - g(m, n))^2} \right] \quad (9.31)$$

The PSNR is expressed in dB's. The PSNR measure is superior to other measures such as SNR as it uses a constant value in which to compare the noise against instead of a fluctuating signal as in SNR. This allows PSNR values received to be treated more meaningfully when quantifiably comparing different image-coding algorithms. For this reason, PSNR is used throughout the image-coding circles as a valid way to compare image-coding algorithms.

(b) *Subjective Fidelity Criteria* In subjective quality measurement, the original and the reconstructed image is given to set g observes. Based on their observations, the quality of the reconstructed image is judged. The five scale rating system to evaluate the quality of the reconstructed image. Used by Bell labs is given below

(1) Impairment is not noticeable (2) Impairment is just noticeable (3) Impairment is definitely noticeable, but not objectionable (4) Impairment is objectionable (5) Impairment is extremely objectionable.

Energy Compaction The effectiveness of transform-based image compression can be gauged by the ability of the transform to pack input data into as few coefficients as possible. In transform-based image compression, the more energy compacted in a fraction of total coefficients, the better energy compaction the transform has. One of the measures of energy compaction is transform coding gain G_{TC} . The transform coding gain is defined as the ratio between the arithmetic mean to the geometric mean of the variances of all the components in the transformed vector. It is given by

$$G_{TC} = \frac{\frac{1}{N} \sum_{i=0}^{N-1} \sigma_i^2}{\left(\prod_{i=0}^{N-1} \sigma_i^2 \right)^{\frac{1}{N}}} \quad (9.32)$$

If the value of G_{TC} is larger then it indicates higher energy compaction.

9.16 SCALAR QUANTISATION

The scalar quantiser is characterised by a staircase function that relates input samples with output measurement values. There are two types of scalar quantisation; (i) uniform scalar quantisation and, (ii) non-uniform scalar quantisation.

9.16.1 Uniform Scalar Quantisation

A uniform scalar quantiser divides or partitions the domain of input values into equally spaced intervals, except at the two outer intervals. The end points of partition intervals are called the quantiser's decision boundaries. Uniform scalar quantisation can be broadly classified into two types: (i) midrise quantiser, and (ii) midtread quantiser. A *midtread quantiser* has zero as one of its output values, whereas the *midrise quantiser* has a partition interval that brackets zero. The midrise and midtread uniform quantisers are shown in Fig. 9.41(a) and Fig. 9.41(b) respectively.

In a midtread quantiser, the values in the interval that includes zero are assigned zero. The quantiser is called midtread because zero is in the middle of a horizontal step, or a tread. If zero itself is a boundary level then values from its left neighbour interval are assigned a negative value, and values from the right neighbour interval are assigned a positive number. This type of output is given by the midrise quantiser. It is called midrise because zero is in the middle of the vertical interval or a riser.

A midtread quantiser is important when source data represents the zero value by fluctuating between small positive and negative numbers. Applying a midtread quantiser in this case would produce an accurate and steady representation of the zero value.

The length of each interval is referred as step size. The step size is denoted by the symbol Δ . For a special case, where $\Delta = 1$, the output of the quantisers are computed as

$$Q_{\text{midrise}}(x) = \lceil x \rceil - 0.5 \quad (9.33)$$

$$Q_{\text{midtread}}(x) = \lfloor x + 0.5 \rfloor \quad (9.34)$$

Let us analyse the performance of an M -level quantiser. Let $B = \{b_0, b_1, \dots, b_M\}$ be the set of decision boundaries and $Y = \{y_1, y_2, \dots, y_M\}$ be the set of reconstruction or output values. If the input is uniformly distributed in the range $[-X_{\max}, X_{\max}]$, the rate of the quantiser is given by

$$R = \lceil \log_2^M \rceil \quad (9.35)$$

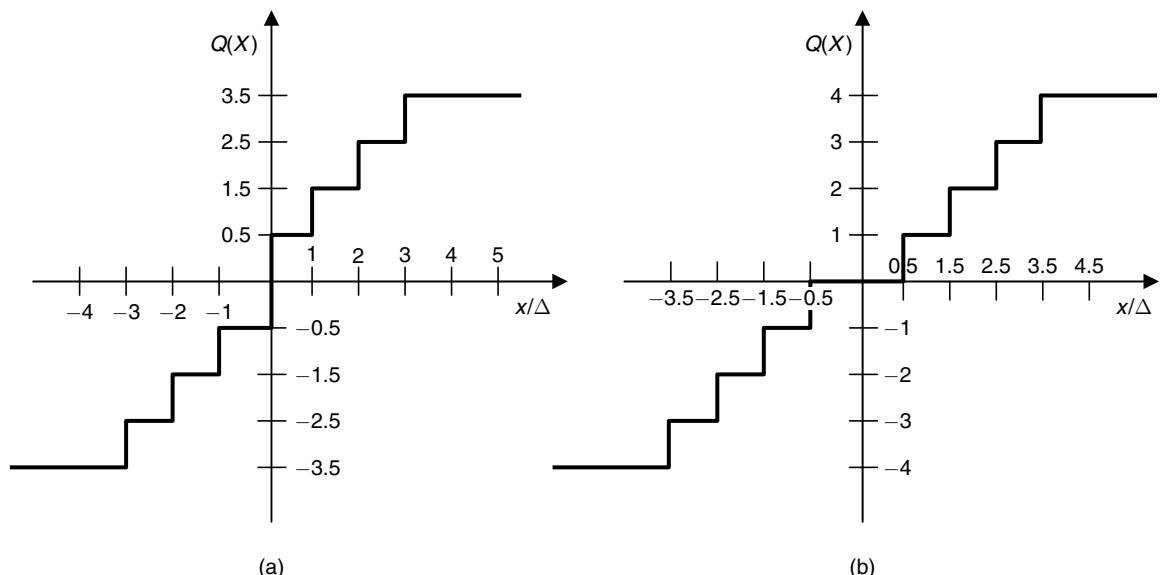


Fig. 9.41 Uniform quantiser (a) midrise (b) midtread

R is the number of bits required to code M output values. The step size is given by

$$\Delta = \frac{2X_{\max}}{M} \quad (9.36)$$

since the entire range of input values is from $-X_{\max}$ to X_{\max} .

For bounded input, the quantisation error caused by the quantiser is referred as granular distortion.

The reconstruction values y_i are the midpoints of each interval, the quantisation error must lie within the values $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$.

The quantisation error of a uniformly distributed source is shown in Fig. 9.42.

The error value at x is denoted as $e(x)$ and is given by $e(x) = x - \frac{\Delta}{2}$. The variance of the error is given by

$$\begin{aligned} \sigma_e^2 &= \frac{1}{\Delta} \int_0^{\Delta} (e(x) - \bar{e})^2 dx \\ &= \frac{1}{\Delta} \int_0^{\Delta} \left(x - \frac{\Delta}{2} - 0\right)^2 dx \\ \text{or, } \sigma_e^2 &= \frac{\Delta^2}{12} \end{aligned}$$

The signal variance is given by $\sigma_x^2 = \frac{(2X_{\max})^2}{12}$. The signal-to-quantisation noise ratio is given by

$$\text{SQNR} = 10 \log_{10} \left(\frac{\sigma_x^2}{\sigma_e^2} \right) \quad (9.37)$$

$$\begin{aligned} \text{SQNR} &= 10 \log_{10} \left(\frac{\sigma_x^2}{\sigma_e^2} \right) \\ &= 10 \log_{10} \left(\frac{(2X_{\max})^2}{12} \frac{12}{\Delta^2} \right) \end{aligned}$$

If $\Delta = \frac{2X_{\max}}{M}$, we get

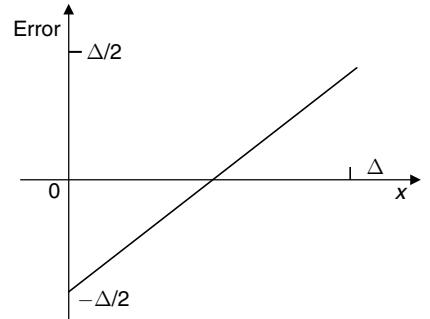


Fig. 9.42 Quantisation error

$$\text{SQNR} = 10 \log_{10} \left(\frac{(2X_{\max})^2}{12} \frac{12}{\left(\frac{2X_{\max}}{M} \right)^2} \right)$$

$$\text{SQNR} = 10 \log_{10} M^2$$

If $M = 2^n$ then we get

$$\text{SQNR} = 10 \log_{10} (2^n)^2 = \text{SQNR} = 2n \log_{10}^2 = 6.02n$$

For every additional bit, the SQNR increases by a factor of six.

9.17 VECTOR QUANTISATION

Vector quantisation (VQ) is a block-coding technique that quantises blocks of data instead of single sample. VQ exploits the correlation existing between neighbouring signal samples by quantising them together. In general, a VQ scheme can be divided into two parts: the encoding procedure, and the decoding procedure which is depicted in Fig. 9.43.

At the encoder, the input image is partitioned into a set of non-overlapping image blocks. The closest code word in the code book is then found for each image block. Here, the closest code word for a given block is the one in the code book that has the minimum squared Euclidean distance from the input block. Next, the corresponding index for each searched closest code word is transmitted to the decoder. Compression is achieved because the indices of the closest code words in the code book are sent to the decoder instead of the image blocks themselves.

The goal of VQ code-book generation is to find an optimal code book that yields the lowest possible distortion when compared with all other code books of the same size. VQ performance is directly proportional to the

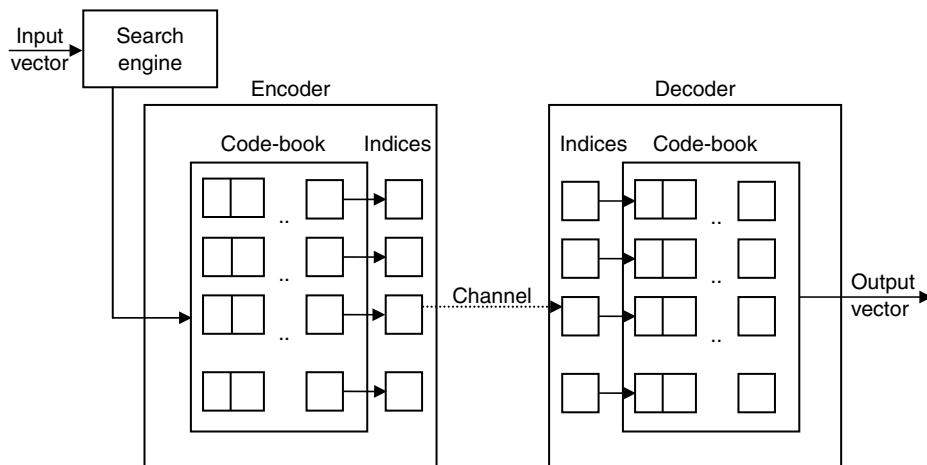


Fig. 9.43 Vector quantisation scheme

code-book size and the vector size. The computational complexity in a VQ technique increases exponentially with the size of the vector blocks. Therefore, the blocks used by VQ are usually small. The encoder searches the code book and attempts to minimise the distortion between the original image block and the chosen vector from the code book according to some distortion metric. The search complexity increases with the number of vectors in the code book. To minimise the search complexity, the tree search vector quantisation scheme was introduced.

VQ can be used to compress an image both in the spatial domain and in the frequency domain. Vector quantisation is a lossy data-compression scheme based on the principles of block coding. A vector quantiser maps a data set in an n -dimensional data space into a finite set of vectors. Each vector is called a code vector or a code word. The set of all code words is called a code book. Each input vector can be associated with an index of a code word and this index is transferred instead of the original vector. The index can be decoded to get the code word that it represented.

Before going into vector quantisation, we should have knowledge about two techniques.

1. Mapping technique (grouping the values)
2. Designing a code book (mechanism of mapping or entries of code words)

The number of code vectors (N) depends upon two parameters—rate (R) and dimensions (L). The number of code vectors is calculated through the following formulae:

$$\text{Number of code vectors } (N) = 2^{R \cdot L} \quad (9.38)$$

where

$$R \rightarrow \text{Rate (bits/pixel)}$$

$$L \rightarrow \text{Dimensions (grouping)}$$

When the rate increases, the number of code vector increases. As the number of code vectors increases, the size of the code book also increases.

Example 9.7 You are given an image of size $X(m, n) = \begin{bmatrix} 02 & 04 & 06 & 08 \\ 10 & 11 & 16 & 15 \\ 09 & 03 & 01 & 07 \\ 12 & 14 & 13 & 05 \end{bmatrix}$. Illustrate the code-book formation in a step-by-step procedure. Also, show how you will reconstruct the image from the code book indices.

Solution From the input image, the following data can be obtained through visual inspection:

- The maximum value in the input image is sixteen.
- The minimum value in the input image is one.

Step 1 Computation of dynamic range Dynamic range is the difference between the maximum and minimum values. The dynamic range decides the number of bits required to represent the pixel.

In this example, the maximum value is 16 and the minimum value is 1. Hence the dynamic range is

$$\text{Dynamic range} = 16 - 1 = 15$$

From the dynamic range, it is obvious that a minimum of four bits is required to represent each pixel value.

Step 2 Fixing the rate and dimension Actually, four bits are necessary to represent each pixel value. Then to achieve compression, the rate is fixed to be two ($R = 2$).

The way in which the pixels are grouped is determined by dimension. In our case, the dimension is fixed as two ($L = 2$) which means two pixels in the input image are grouped into a vector. Totally, we have 16 elements in the input matrix. After grouping, we have only 8 elements instead of sixteen elements which is shown in Fig. 9.44.

Step 3 Determining the number of code vectors After fixing the rate and dimension, it is possible to compute the number of code vectors in the code book. The number of code vectors in the code book is given by

$$\text{number of code vectors in the code book is } N = 2^{RL}$$

In our case, $R=2$ and $L=2$. Hence the number of code vectors is $N = 2^{RL} = 2^{(2 \times 2)} = 16$. Totally, sixteen code vectors are there in this example. The code vectors are from C_{00} to C_{15} .

Step 4 Determining the code vectors through centroid method These code vectors from C_{00} to C_{15} are determined through the centroid method. The first step in the centroid method is fixing the interval. We have used the following formula to compute the interval

$$\text{Interval} = \left\lceil \frac{\text{maximum value} - \text{minimum value}}{R \times L} \right\rceil$$

In our case, the maximum value is 16 and the minimum value is 1. Substituting this value, we obtain the interval as

$$\text{Interval} = \left\lceil \frac{\text{maximum value} - \text{minimum value}}{R \times L} \right\rceil = \left\lceil \frac{16 - 1}{4} \right\rceil = 4$$

The interval and the corresponding code vectors are illustrated in Fig. 9.45. From the figure, it is obvious that C_{00} , C_{01} , C_{15} are the code vectors of the generated code book and the corresponding positions are (2, 14), (6, 14) ... , (14, 2) respectively.

Step 5 Mapping input vectors into the code vectors Mapping of the input vector into the code vector is done through projection. In our case, there are totally eight input vectors. The projection of these eight input vectors into the code vector is illustrated by a star symbol (*) in Fig. 9.46. If one observes carefully, there are a total of eight stars (corresponding to the input vector) in Fig. 9.46.

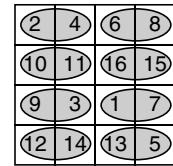


Fig. 9.44 Grouping of input elements

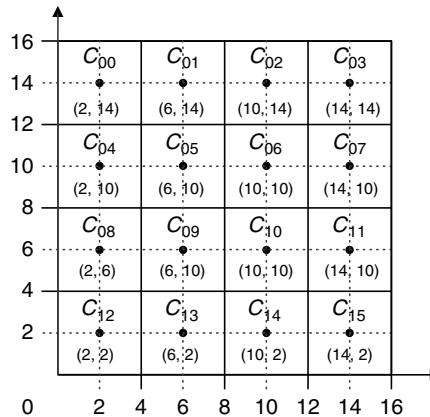


Fig. 9.45 Code vectors through centroid method

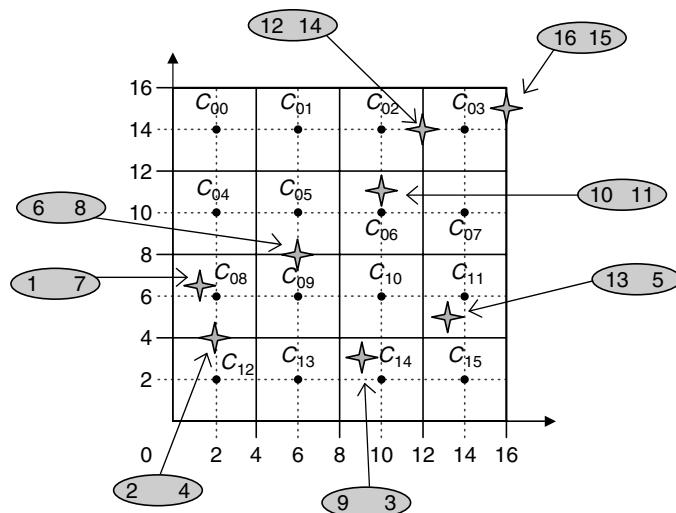


Fig. 9.46 Projection of input vectors into code vectors

Step 6 Adjusting the input vector to fall into the code vector The input vector is adjusted so as to fall into one of the nearest code vectors. This concept is illustrated in Fig. 9.47 (a) and (b).

From Fig. 9.47(a), it is obvious that the first input image vector (2, 4) is adjusted to fall into the code vector C_{08} . The next image vector (6, 8) is projected into the code vector C_{09} . The third code vector (10, 11) is adjusted to fall on to nearest the code vector C_{06} . The fourth image vector (16, 15) is adjusted to fall on the nearest code vector C_{03} . The fifth image vector (9, 3) is adjusted to fall on the nearest code vector C_{14} . The sixth image vector (1, 7) is adjusted to fall on the nearest code vector C_{08} . If we consider the seventh image vector (12, 14), there are two possible choices—it can be adjusted to fall either on to C_{02} or C_{03} . In this case we have chosen the code vector C_{03} . Finally, the eighth image vector (13, 5) is adjusted to fall on the nearest code vector C_{11} .

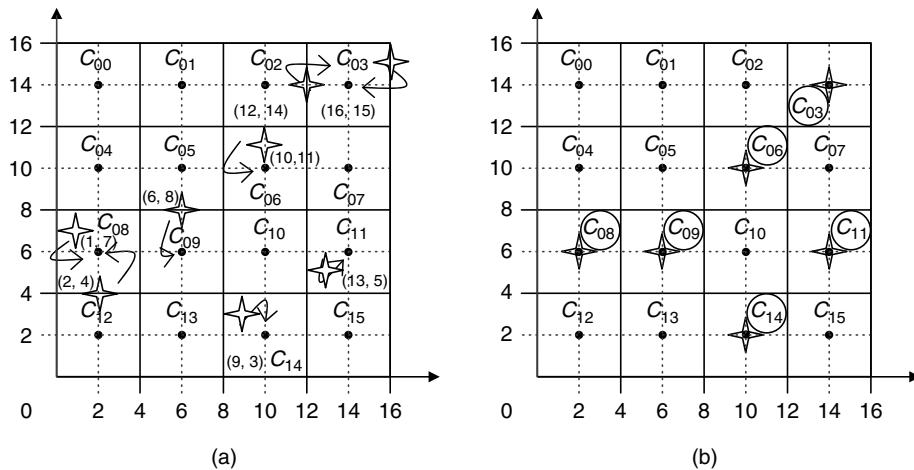


Fig. 9.47 (a) Adjusting the input vector to fall into the code vector (b) Code vectors after adjustment

Step 7 Transmission of indices In vector quantisation, the indices corresponding to the code vectors are transmitted. The indices to be transmitted for the corresponding input image vector are tabulated in Table 9.4.

Step 8 Reconstruction of the image from the transmitted indices In the receiver end, the receiver receives only the indices, not the code vectors. Upon the reception of the indices, the corresponding code vectors are reconstructed, because the same code book is maintained at the transmitter and the receiver end. The code vectors corresponding to the received indices are shown in Table 9.5.

Table 9.4 Indices corresponding to the image vector

Serial Number	Input image vector	The centroid, nearer to the input indices	Transmitted indices and the corresponding code vector
1	(2, 4)	(2, 6)	08 – C_{08}
2	(6, 8)	(6, 6)	09 – C_{09}
3	(10, 11)	(10, 10)	06 – C_{06}
4	(16, 15)	(14, 14)	03 – C_{03}
5	(9, 3)	(10, 2)	14 – C_{14}
6	(1, 7)	(2, 6)	08 – C_{08}
7	(12, 14)	(14, 14)	03 – C_{03}
8	(13, 5)	(14, 6)	11 – C_{11}

Table 9.5 Code vectors corresponding to the received indices

Received indices	Code vectors corresponding to the indices
08	(2, 6)
09	(6, 6)
06	(10, 10)
03	(14, 14)
14	(10, 2)
08	(2, 6)
03	(14, 14)
11	(14, 6)

The image is reconstructed at the receiver end from the reconstructed code vectors. The reconstructed image is shown in Fig. 9.48.

The input image and the reconstructed image at rate = 2 are shown side by side in Fig. 9.49 (a) and (b) respectively.

From the original and reconstructed image, it is obvious that error is inevitable in vector quantisation.

2	6	6	6
10	10	14	14
10	2	2	6
14	14	14	6

Fig. 9.48 Reconstructed image at the rate of two

2	4	6	8
10	11	16	15
9	3	1	7
12	14	13	5

(a)

2	6	6	6
10	10	14	14
10	2	2	6
14	14	14	6

(b)

Fig. 9.49 (a) Input image (b) Reconstructed image at $R = 2$

Example 9.8 For the same image shown in Example 9.7, construct a code book using rate = 1 and dimension = 2. Also, reconstruct the image from the transmitted indices. Compare the reconstructed image obtained in this example with Example 9.7 and comment on the observed result.

Solution The dynamic range of the image will not change because the image considered in Example 9.7 is considered in Example 9.8; hence, the computation of dynamic range is not repeated in this example.

Step 1 Fixing the rate and dimension In this exercise, the rate $R = 1$ and the dimension $L = 2$.

Step 2 Computation of number of code vectors After fixing the rate and dimension, the number of code vectors is computed as

$$N = 2^{RL} = 2^{(1 \times 2)} = 4$$

The four code vectors are labeled as C_0 , C_1 , C_2 and C_3 .

Step 3 Determining the code vectors through centroid method Determining the code vectors involves the computation of the interval. The formula which is used to compute the interval is given below:

$$\text{Interval} = \left\lceil \frac{\text{maximum} - \text{minimum}}{R \times L} \right\rceil = \left\lceil \frac{16 - 1}{1 \times 2} \right\rceil = 8$$

The interval is determined to be eight. After the determining the interval, the code vectors are determined through centroid method which is illustrated in Fig. 9.50.

From Fig. 9.50, it is obvious that C_{00} , C_{01} , C_{02} and C_{03} are the code vectors of the generated code book, and the corresponding positions are (4, 12), (12, 12), (4, 4) and (12, 4) respectively.

Step 4 Projection of the input vector into the code vector The projection of the eight input image vectors into any one of the four nearest code vectors is illustrated in Fig. 9.51(a) and (b).

From Fig. 9.51(a), it is obvious that the first input image vector (2, 4) is adjusted to fall into the code vector C_{02} , the next image vector (6, 8) is projected into

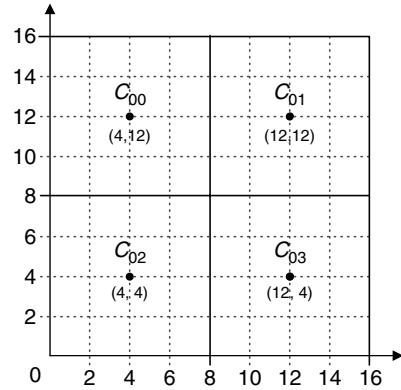


Fig. 9.50 Code vectors through centroid method

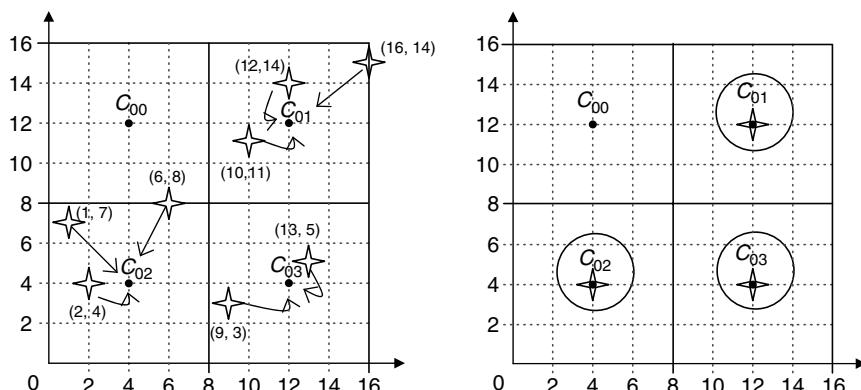


Fig. 9.51 (a) Adjusting the input vector to fall into the code vector (b) Code vectors after adjustment

the code vector C_{02} . The third code vector (10, 11) is adjusted to fall on the nearest the code vector C_{01} . The fourth image vector (16, 15) is adjusted to fall on the nearest code vector C_{01} . The fifth image vector (9, 3) is adjusted to fall on the nearest code vector C_{03} . The sixth image vector (1, 7) is adjusted to fall on the nearest code vector C_{02} . If we consider the seventh image vector (12, 14), it is adjusted to fall on C_{01} . Finally, the eighth image vector (13, 5) is adjusted to fall on the nearest code vector C_{03} .

Step 5 Transmission of indices In vector quantisation, the indices corresponding to the code vectors are transmitted. The indices to be transmitted for the corresponding input image vector are tabulated in Table 9.6.

Step 6 Reconstruction of the image from the transmitted indices In the receiver end, the receiver receives only the indices, not the code vectors. Upon the reception of the indices, the corresponding code vectors are reconstructed, because the same code book is maintained at the transmitter and the receiver end. The received indices and the corresponding code vectors are shown in Table 9.7.

Table 9.6 Indices corresponding to the image vector

Serial number	Input image vector	The centroid, nearer to the input indices	Transmitted indices and the corresponding code vector
1	(2, 4)	(4, 4)	02 – C_{02}
2	(6, 8)	(4, 4)	02 – C_{02}
3	(10, 11)	(12, 12)	01 – C_{01}
4	(16, 15)	(12, 12)	01 – C_{01}
5	(9, 3)	(12, 4)	03 – C_{03}
6	(1, 7)	(4, 4)	02 – C_{02}
7	(12, 14)	(12, 12)	01 – C_{01}
8	(13, 5)	(12, 4)	03 – C_{03}

Table 9.7 Code vectors corresponding to the received indices

Received Indices	Code vectors corresponding to the indices
02 – C_{02}	(4, 4)
02 – C_{02}	(4, 4)
01 – C_{01}	(12, 12)
01 – C_{01}	(12, 12)
03 – C_{03}	(12, 4)
02 – C_{02}	(4, 4)
01 – C_{01}	(12, 12)
03 – C_{03}	(12, 4)

The image is reconstructed at the receiver end from the reconstructed code vectors. The reconstructed image is shown in Fig. 9.52.

Step 7 Comparison of image reconstructed at different rates In Example 9.7, the image is reconstructed with rate = 2. In this example, the image is reconstructed with rate = 1 by fixing the dimension to be two in both the cases. The original image and the reconstructed image at rate = 2 and rate = 1 are shown in Fig. 9.53.

From Fig. 9.53, it is obvious that with increase in rate, the reconstructed image resembles the original image. It should be noted that increasing the rate will increase the size of the code book. There is a trade-off between the amount of compression achieved and the quality of the reconstructed image.

4	4	4	4
12	12	12	12
12	4	4	4
12	12	12	4

Fig. 9.52 Reconstructed image at rate of one

2	4	6	8
10	11	16	15
9	3	1	7
12	14	13	5

(a)

4	4	4	4
12	12	12	12
12	4	4	4
12	12	12	4

(b)

2	6	6	6
10	10	14	14
10	2	2	6
14	14	14	6

(c)

Fig. 9.53 (a) Original image (b) Reconstructed image with rate = 2 (c) Reconstructed image with rate = 1

9.18 TYPES OF VECTOR QUANTISATION

The different types of vector quantisation techniques are (i) Tree-Search Vector Quantisation (TSVQ), (ii) Multi-stage Vector Quantisation (MSVQ), (iii) Separating Mean Vector Quantisation or Mean Removed Vector Quantisation, (iv) Gain-Shape Vector Quantisation, (v) Classified Vector Quantisation, (vi) Hierarchical Vector Quantisation, (vii) Interpolative Vector Quantisation, (viii) Lapped Vector Quantisation, and (ix) Lattice Vector Quantisation.

9.18.1 Tree-Search Vector Quantisation

A modification of the full-search vector-quantisation technique is the tree-structure vector quantisation. TSVQ requires fewer searches than full-search VQ. A full-search vector quantiser uses an unstructured set of reproduction vectors and determines the appropriate index for transmission by computing the distortion between the input vector and each possible reproduction vector. The number of distortion calculations required by the full-search vector quantiser is $O(m)$. In binary tree-structured vector quantisation, a sequence of binary searches is performed rather than one large search, as in full-search vector quantisation. The structure of binary tree-structured vector quantisation is illustrated in Fig. 9.54.

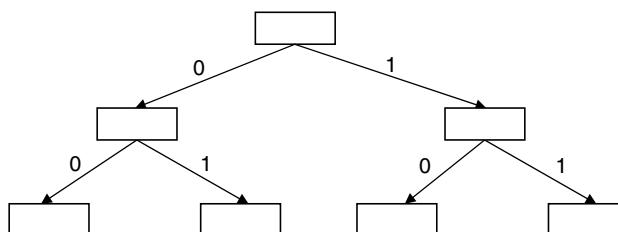


Fig. 9.54 Binary tree-structure vector quantisation

From Fig. 9.54, it is obvious that the terminal nodes are at the same depth in binary tree-structured vector quantisation. The number of distortion calculations required by tree-structured vector quantisation is $O(\log_2^m)$. The fixed-rate tree-structured vector quantisation generally has a higher distortion than a fixed-rate full-search vector quantisation due to the constraints on the search.

The code word in TSVQ is selected by a sequence of binary decisions. Vector reproductions are stored at each node in the tree. The search begins at the root node. The encoder compares the input vector to two possible candidate reproductions, chooses the one with the minimum distortion, and advances to the selected node. If the node is not a terminal node of the tree, the encoder continues and chooses the best available node of the new pair presented. The encoder produces binary symbols to represent its sequence of left/right decisions. The stored index is then a pathmap through the tree to the terminal node, which is associated with the final code word.

Advantages The main advantage of TSVQ is that the search complexity is reduced. The search complexity is reduced to $O(\log M)$ from $O(M)$ for an unstructured code book of size M .

Drawbacks The main drawback of TSVQ is the storage requirement. The performance of TSVQ is, in general, inferior to that of unstructured VQ of the same rate since a source vector may not be mapped on to the optimum code vector because of the imposed structure.

9.18.2 Multistage Vector Quantisation

Multistage vector quantisation is also known as residual vector quantisation or cascaded vector quantisation. Multistage vector quantisation was introduced by Juang and Gray in 1982. Multistage vector quantisation divides the encoding task into several stages as illustrated in Fig. 9.55. The first stage performs a relatively crude encoding of the input vector. Then the second stage operates on the error vector between the original vector and the quantised first-stage output.

The quantised error vector provides a refinement to the first approximation. At the decoder, the reconstruction vectors are obtained using a Look-up table (LUT). Additional stages of quantisers can provide further refinements. Unlike full-search VQ, whose encoding complexity and memory requirements increase exponentially with the dimension-rate product, in MSVQ, the increase is only linear. This has particular utility in sub-band coding since either the rate or the dimension can be made large, which allows it to respond to the occasional need for the locally high bit-rate sub-band coding.

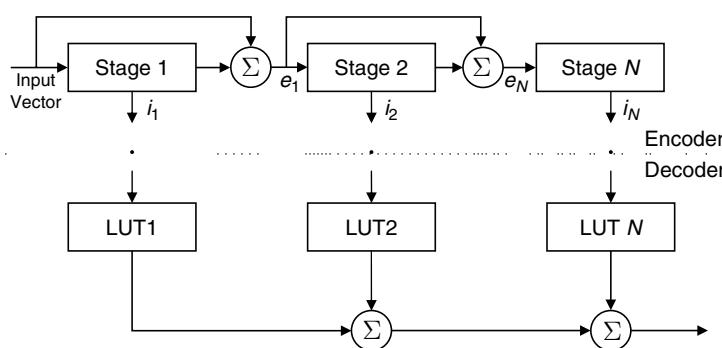


Fig. 9.55 Multistage vector quantisation scheme

9.18.3 Mean Removed Vector Quantisation

In images, the mean can often vary a lot between different parts of the image. The mean value is subtracted from the image vectors to yield a residual vector with zero mean. The residual vectors are quantised with VQ and the corresponding indices are transmitted. The block diagram of MRVQ is shown in Fig. 9.56. Reconstruction is performed by adding the mean to the decoded image vector.

From Fig. 9.56, it is obvious that the output of the overall system is the VQ indices and the mean values.

9.18.4 Gain-Shape Vector Quantisation

In gain-shape vector quantisation, the input vectors are decomposed into a scalar gain term and a gain normalised vector term, which is generally termed the shape. The gain value is given by

$$g = \|x\| = \sqrt{\sum_{i=1}^k x^2[i]} \quad (9.39)$$

The gain factor is the energy or the variance of the code vector.

The shape vector is given by

$$S = \frac{x}{g} \quad (9.40)$$

The gain term is quantised with a scalar quantiser, while the shape vectors are represented by a shape code book designed specifically for the gain-normalised shape vectors.

Advantage of Vector Quantisation Over Scalar Quantization Vector quantisation exploits the linear and non-linear dependence among vectors. It also provides flexibility in choosing multidimensional quantiser cell shapes and in choosing a desired code-book size.

The advantage of VQ over SQ is the fractional value of resolution that is achievable. This is very important for low-bit-rate applications where low resolution is sufficient.

9.19 WAVELET-BASED IMAGE COMPRESSION

The wavelet transform has the ability to decorrelate an image both in space and frequency, thereby distributing energy compactly into a few low-frequency and a few high-frequency coefficients. The efficiency of a wavelet-based image-compression scheme depends both on the wavelet filters chosen, as well as on the coefficient quantisation scheme employed. The basis functions of the wavelet transform are localised in both time and frequency. The concept of wavelet-based image compression is discussed in detail in Chapter 12.

9.20 FRACTAL IMAGE COMPRESSION

Mandelbrot is considered to be one of the first to introduce the principles of fractal geometry. He coined the word 'fractal' in 1975. The term fractal is used to describe the irregular and fragmented patterns that appear in

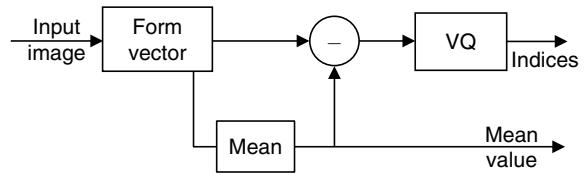


Fig. 9.56 Mean removed vector-quantisation scheme

nature. The fractal is generated by an iterative process, starting from an initial tile. The fractal object can be regarded to be composed of many copies of itself, each of which is scaled down with possible translations and rotations. Fractals possess self-similarity across scales. The meaning for the term ‘self-similarity’ is that if one zooms in or out, the image has a similar appearance. Self-similarity can be found in fern leaves as shown in Fig. 9.57.

From Fig. 9.57, it is obvious that each fern leaf resembles a smaller portion of the fern. This is known as the famous Barnsley fern. The self-similarity property makes fractals independent of scaling. Thus, there is no characteristic size associated with a fractal.



Fig. 9.57 Fern leaf exhibiting self-similarity across scales

9.20.1 Fractal Development

The fractal was developed as a kind of geometry by the IBM mathematician Benoit B Mandelbrot. In 1981, the mathematician John Hutchinson used the theory of iterated function system to model collections of contractive transformations in a metric space as a dynamical system. It was Michael Barnsley who generated the fractal model using the Iterated Function System (IFS) which was used to code images. However, Barnsley’s image-compression algorithm based on fractal mathematics was inefficient due to the searching problem. In 1988, one of Barnsley’s PhD students, Arnaud Jacquin, arrived at a modified scheme for representing images. This was the called Partitioned Iterated Function Systems (PIFS), which led to encoding of images to achieve significant compression.

9.20.2 Fractal Basics

Mathematically, a fractal can be represented by the set of equations given in Eq. (9.41)

$$W \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \quad (9.41)$$

Starting with an initial value of x and y , the above equation when recursively iterated generates a fractal. In Eq. (9.41), e and f perform translation and the parameters a , b , c and d denote the rotation and skewing operations. The set $\{W\}$ is called an *affine transformation*. These affine transformations are called *contractive affine transformations*. A fractal can be represented by a set of contractive affine transformations which are given by

$$W_n, p_n \quad n = 1, 2, \dots, N \quad (9.42)$$

where W_n are the affine transformations and p_n denotes their individual importance. This representation is called the iterated function system (IFS) code.

Fractal image compression is based on self-similarities within an image. In fractal image compression, image blocks are seen as rescaled and intensity-transformed approximate copies of blocks found elsewhere in the image. For an exactly self-similar image, the degree of compression achievable is very high.

The steps involved in fractal image compression are summarised below

1. First, the input image is partitioned into sub-images or range blocks. There are many ways of partitioning an image. Some of the popular schemes are quadtree partitioning, triangular partitioning,

horizontal–vertical partitioning and polygonal partitioning. The key point in fractal compression is to partition the image into a smaller number of range blocks that are similar to other image parts under certain transformations.

2. Search for parts of the images or domain blocks which are self-similar in a statistical sense. The problem of fractal image compression is to find the best domain that will map to a range. A domain is a region where the transformation maps from, and a range is the region where it maps to.

The possible choice of a domain representing the range block is illustrated in Fig. 9.58.

If the best match between the largest range cells and the transformed domain cells still has an error measurement that is greater than the similarity threshold, the range cell is further partitioned and re-evaluation continues similarly with each of these cells. Range cells that cannot be matched within the similarity threshold continue to be partitioned into smaller range cells until the maximum number of partitions is reached. If this limit is reached and the closest domain cell does not match a range cell within the similarity threshold, this area would be marked as an anomalous region.

3. Once a match is located, compute the transformations. The transformation is a combination of an affine transformation and a luminance transformation. In matrix form, the transformation is expressed as

$$W \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & p \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ q \end{pmatrix} \quad (9.43)$$

Here, z denotes the pixel intensity at the location (x, y) . e and f are the relative x and y positions of the range with respect to the domain. p and q are the contrast and brightness adjustments for the transformation.

4. The basis for fractal image compression is the construction of the iterated function system that approximates the original image. An IFS is a union of contractive transformations, each of which maps into itself. For the transformation to be contractive, the following equation should be satisfied:

$$d(W(P_1), W(P_2)) < d(P_1, P_2) \quad (9.44)$$

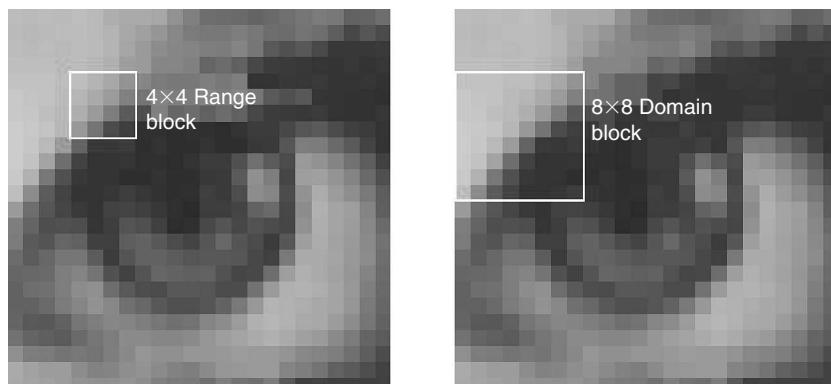


Fig. 9.58 Domain and range block

Equation (9.44) states that the distance $d(P_1, P_2)$ between any two points in a metric space X is reduced by applying the transformation W which maps X into itself. A metric to measure distance when $P_1=(x_1, y_1)$ and $P_2=(x_2, y_2)$ are in two-dimensional Euclidean space is the standard Euclidean metric given by

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (9.45)$$

By defining IFS on a space of images, any initial image will converge to one and only one final attractor. Repeatedly applying an affine transformation repetitively on any image results in an image called the attractor. The attractor is unchanged for any further application of the affine transformation. Each affine transformation has a characteristic attractor associated with it. The goal of fractal image compression is to encode an image as the fixed point of a suitable contractive mapping.

9.20.3 Fractal Encoding Parameters

In fractal image coding, there are a number of parameters and options to be selected regarding the range and domain generation used in the encoding process. The important parameters are

- (i) Level of partitioning of domain image,
- (ii) Selection of value of similarity threshold,
- (iii) Contractive vs non-contractive mapping, and
- (iv) Overlapping vs non-overlapping domain cells.

(i) Level of Partitioning of the Image The quadtree partitioning scheme is widely used in fractal image coding. In the case of quadtree partitioning, the choice of maximum and minimum level of quadtree partitioning depends on (a) the size of the structure found in the image, and (b) the expected size of the anomaly or the object to be segmented.

(ii) Selection of Similarity Threshold The optimal similarity threshold value depends on the range of root mean square value over which anomalies tend to occur and fluctuates based on partition size selected. If the similarity threshold is too high, everything in the image is mapped and no segmentation of anomalous regions occurs. If the similarity threshold is too low, nothing in the image is mapped and the entire image is seen as being representative of the anomaly.

(iii) Contractive Vs Non-contractive Mapping Fractal-based image coding uses contractive mappings, which shrink the dimensions of the domain cell for comparison with range cells. Because of contractive mapping, domain partitioning always began and terminated at one level higher than range partitioning to ensure that there were always domain cells that could be contractively mapped to range cells. Contractive mapping generally works well for amorphous-like features of mammograms. A non-contractive mapping could be employed for images with large structures like semiconductor images.

(iv) Overlapping Vs Non-overlapping Domain Cells Overlapping cells will result in much larger domain sets than non-overlapping cells. Overlapping domains result in increase in the number of comparisons. Increase in the number of comparison in overlapping domain results in longer computation time.

Advantages of Fractal Image Compression The fractal images are stored as mathematical formulas rather than as bitmaps. Hence, they can be decompressed to resolutions that are higher or lower than those of the original. The ability to scale images without distortion is fractal compression's important advantage over JPEG.

Drawbacks of Fractal Image Compression The main drawbacks of fractal image compression are summarised below:

- (i) Fractal image compression basically exploits the self-similarity present in the image. Hence, images with random content are not likely to be compressed well as only few similarities of different sizes are likely to exist.
- (ii) The encoding complexity of fractal image compression is high when compared to decoding complexity. The encoding complexity is due to the search for similarity between the blocks in the image.

9.21 BLOCK TRUNCATION CODING

The Block Truncation Coding (BTC) technique is based on preserving the first and second statistical moments of the image. The BTC method was first presented in a paper by Delp and Mitchell in 1979. Block truncation coding divides the image into blocks. The block size is usually 3×3 or 4×4 pixels. Within each block, a threshold is chosen, and the value at each pixel is coded as a 0 or 1 depending on whether it is above or below the threshold. BTC attempts to preserve the mean and variance (first and second statistical moment) of each block.

9.21.1 Procedure of BTC

The steps followed in block truncation coding are given below:

- (i) The image is partitioned into a set of non-overlapping blocks. Let $X(m, n)$ be the original image which is partitioned into blocks of pixels which is represented as a matrix $f(m, n)$.
- (ii) For each block, the mean, mean square and the variance are calculated.

$$(a) \text{ The mean is calculated as } \bar{f} = \frac{1}{m \times n} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m, n) \quad (9.46)$$

- (b) The mean square value is calculated using the formula

$$\bar{f}^2 = \frac{1}{m \times n} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f^2(m, n) \quad (9.47)$$

- (c) The variance is computed using the formula

$$\sigma^2 = \bar{f}^2 - \bar{f}^2 \quad (9.48)$$

- (iii) A binary allocation matrix $B(m, n)$ is constructed in such a way that

$$B(m, n) = \begin{cases} 1 & : f(m, n) > \bar{f} \\ 0 & : f(m, n) \leq \bar{f} \end{cases} \quad (9.49)$$

Let q represent the number of pixels greater than mean. In other words q is the number of ones in the matrix B .

- (iv) Two values a and b for each block is calculated using the formula

$$a = \bar{f} - \sigma \sqrt{\frac{q}{m - q}} \quad (9.50)$$

$$b = \bar{f} + \sigma \sqrt{\frac{m-q}{q}} \quad (9.51)$$

Here, m is the number of pixels within each block.

- (v) After calculating the values of a and b , the block values are reconstructed as

$$\hat{f}(m, n) = \begin{cases} a : B(m, n) = 0 \\ b : B(m, n) = 1 \end{cases} \quad (9.52)$$

9.21.2 Advantage of BTC

BTC lends itself very nicely to parallel processing since the coding of each of the blocks is totally independent.

Example 9.9 A block of the matrix is given as $f(m, n) = \begin{bmatrix} 65 & 75 & 80 & 70 \\ 72 & 75 & 82 & 68 \\ 84 & 72 & 62 & 65 \\ 66 & 68 & 72 & 80 \end{bmatrix}$. Apply BTC coding procedure to this block and obtain the reconstructed value.

Solution

Step 1 Computation of mean, mean square and variance of the block

(a) Mean value is computed as \bar{f}

$$\bar{f} = \frac{1}{16} \times \{65 + 75 + 80 + 70 + 72 + 75 + 82 + 68 + 84 + 72 + 62 + 65 + 66 + 68 + 72 + 80\}$$

$$\bar{f} = 72.25$$

(b) Mean square value \bar{f}^2 is calculated

$$\bar{f}^2 = \frac{1}{m \times n} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f^2(m, n)$$

$$\bar{f}^2 = \frac{1}{16} \times \{65^2 + 75^2 + 80^2 + 70^2 + 72^2 + 75^2 + 82^2 + 68^2 +$$

$$+ 84^2 + 72^2 + 62^2 + 65^2 + 66^2 + 68^2 + 72^2 + 80^2\}$$

$$\bar{f}^2 = 5261.25$$

(c) The variance of the block is computed as σ^2

$$\sigma^2 = \bar{f}^2 - \bar{f}^2$$

$$\sigma^2 = 5261.25 - (72.25)^2 = 5261.25 - 5220.06 = 41.1875. \text{ This implies } \sigma = 6.4177.$$

Step 2 Computation of binary allocation matrix $B(m, n)$

$$B(m, n) = \begin{cases} 1: & f(m, n) > \bar{f} \\ 0: & f(m, n) \leq \bar{f} \end{cases}$$

Each element in the matrix $f(m, n)$ is compared against the mean value. If $f(m, n)$ is greater than the mean value then 1 is assigned, otherwise 0 is assigned.

$$B(m, n) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The number of 1s in the matrix $B(m, n)$ is six. Hence, $q = 6$.

Step 3 Computation of a and b values

$$a = \bar{f} - \sigma \sqrt{\frac{q}{m-q}}$$

Here, m represents the number of elements in the block matrix. In this case, $m = 16$. Substituting the values of $\bar{f} = 72.25$, $\sigma = 6.4177$, $m = 16$, $q = 6$, we get the value of a as $a = 67.278$. Rounding the value, we get $a = 67$.

The value of b is calculated using the formula

$$b = \bar{f} + \sigma \sqrt{\frac{m-q}{q}}$$

Substituting the values of $\bar{f} = 72.25$, $\sigma = 6.4177$, $m = 16$, $q = 6$, we get the value of b as $b = 80.53$. Rounding this value, we get $b = 81$.

Step 4 Reconstructed block $\hat{f}(m, n)$

The reconstructed block is given as $\hat{f}(m, n) = \begin{cases} a: & B(m, n) = 0 \\ b: & B(m, n) = 1 \end{cases}$.

$$\hat{f}(m, n) = \begin{bmatrix} 67 & 81 & 81 & 67 \\ 67 & 81 & 81 & 67 \\ 81 & 67 & 67 & 67 \\ 67 & 67 & 67 & 81 \end{bmatrix}$$

By comparing the original block $f(m, n)$ with the reconstructed block $\hat{f}(m, n)$, we find that error is inevitable.

MATLAB Example 1: BTC *Read an image; apply BTC by choosing different block sizes. Comment on the observed result.*

Solution The MATLAB code that performs BTC of the input image is shown in Fig. 9.59 and the corresponding output is shown in Fig. 9.60. From the observed result, it is clear that as the block size increases, the quality of the reconstructed image decreases and the blocking artifact becomes visible.

```
%This program performs BTC of the input image
x = imread('C:\Documents and Settings\esakki\Desktop\icecream.jpg');
x = rgb2gray(x);
x = imresize(x,[256 256]);
x = double(x);
x1 = x;
[m1 n1] = size(x);
blk = input('Enter the block size:');
for i = 1 : blk : m1
for j = 1 : blk : n1
y = x( i : i + (blk-1), j : j + (blk-1) ) ;
m = mean( mean ( y ) );
sig = std2(y);
b = y > m ; %the binary block
K = sum ( sum ( b ) ) ;
if (K ~= blk^2) & (K ~= 0)
ml = m - sig*sqrt(K/((blk^2)-K));
mu = m + sig*sqrt(((blk^2)-K)/K);
x(i : i + (blk-1), j : j + (blk-1)) = b*mu + (1- b)*ml;
end
end
end
imshow(uint8(x1)), title('Original image')
figure, imshow(uint8(x)), title('Reconstructed image'),
xlabel(sprintf('The block size is %g', blk))
```

Fig. 9.59 MATLAB code to perform BTC

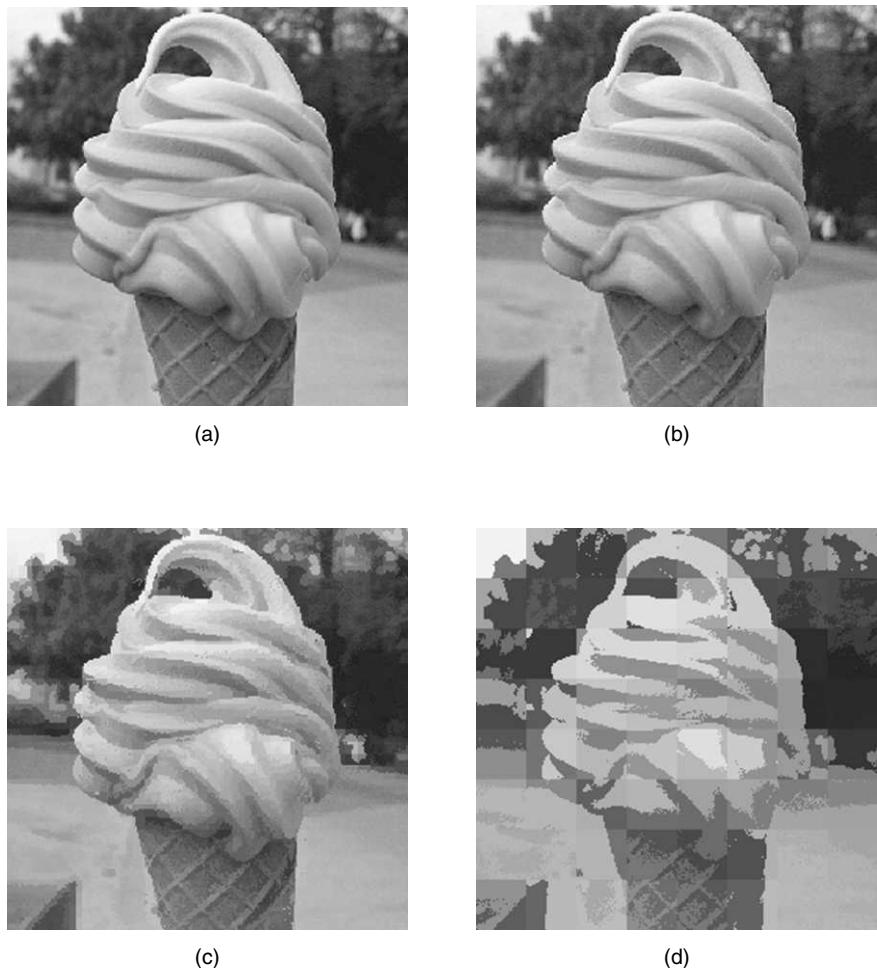


Fig. 9.60 (a) Original image (b), (c) and (d) Reconstructed images using block sizes of 2, 8 and 32 respectively

SOLVED PROBLEMS

1. A 256×256 pixel digital image has eight distinct intensity levels. What is the minimum number of bits required to code this image in a lossless manner?

Solution Eight distinct gray levels can be coded with $\lceil \log_2(8) \rceil = 3$ bits/pixel. This implies the number of bits required to code this image is $256 \times 256 \times 3 = 196608$ bits.

2. One of the objective measures of image compression is Peak Signal-to-Noise Ratio (PSNR). The original and the reconstructed image are given below. Calculate the PSNR expressed in decibels.

$$\text{Original image} = \begin{bmatrix} 1 & 8 & 6 & 6 \\ 6 & 3 & 11 & 8 \\ 8 & 8 & 9 & 10 \\ 9 & 10 & 10 & 7 \end{bmatrix} \quad \text{Reconstructed image} = \begin{bmatrix} 2 & 8 & 8 & 7 \\ 6 & 3 & 12 & 8 \\ 5 & 4 & 9 & 1 \\ 15 & 9 & 11 & 9 \end{bmatrix}$$

Solution The expression for PSNR in decibels is $10 \log_{10} \left[\frac{(2^B - 1)^2}{MSE} \right]$.

where MSE stands for Mean Square Error, which is given by

$MSE = \frac{1}{M \times N} \left\{ [f(m, n) - \hat{f}(m, n)]^2 \right\}$. Here, $f(m, n)$ represents the original image and $\hat{f}(m, n)$ represents the reconstructed image.

Calculation of MSE

The formula to calculate mean square error is given by

$$MSE = \frac{1}{M \times N} \left\{ [f(m, n) - \hat{f}(m, n)]^2 \right\}.$$

In this problem, $M = N = 4$. Substituting this value in the formula for MSE, we get

$$\begin{aligned} MSE &= \frac{1}{4 \times 4} \left\{ (1-2)^2 + (8-8)^2 + (6-8)^2 + (6-7)^2 + (6-6)^2 + (3-3)^2 + (11-12)^2 + (8-8)^2 \right. \\ &\quad \left. + (8-5)^2 + (8-4)^2 + (9-9)^2 + (10-1)^2 + (9-15)^2 + (10-9)^2 + (10-11)^2 + (7-9)^2 \right\} \\ &= \frac{1}{16} (1 + 0 + 4 + 1 + 0 + 0 + 1 + 0 + 9 + 16 + 0 + 1 + 36 + 1 + 1 + 4) \end{aligned}$$

$$MSE = 9.6875$$

In this problem, $B = 4$. Substituting the value of MSE and the value of B in the expression of PSNR, we get

$$PSNR = 10 \log_{10} \left[\frac{(2^4 - 1)^2}{9.6875} \right] = 13.7 \text{ dB}$$

3. Compute the entropy of the image given by $f(m, n) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 2 \\ 0 & 1 & 2 & 3 \\ 1 & 2 & 2 & 3 \end{bmatrix}$

Solution The gray levels in the input image are 0, 1, 2 and 3.

Step 1 Probability of occurrence of the gray level

$$\text{Probability of occurrence of gray level 0 is } p(0) = \frac{5}{16}$$

$$\text{Probability of occurrence of gray level 1 is } p(1) = \frac{4}{16}$$

$$\text{Probability of occurrence of gray level 2 is } p(2) = \frac{5}{16}$$

$$\text{Probability of occurrence of gray level 3 is } p(3) = \frac{2}{16}$$

Step 2 Computation of Entropy

The formula to compute entropy is given by $H = -\sum_{i=1}^N p_i \log_2^{p_i}$ bits/pixel

$$H = -\left(\frac{5}{16} \log_2 \left[\frac{5}{16}\right] + \frac{4}{16} \log_2 \left[\frac{1}{4}\right] + \frac{5}{16} \log_2 \left[\frac{5}{16}\right] + \frac{2}{16} \log_2 \left[\frac{2}{16}\right]\right) \text{ bits/symbol}$$

$$H = 1.924 \text{ bits/symbol}$$

4. Prove that entropy is maximum when the symbols are equiprobable.

Solution In order to prove the fact that entropy is maximum when the symbols are equiprobable, let us consider two cases. In the case (i) the symbols are equiprobable and in the case (ii) the symbols are not equiprobable. By calculating the entropy in the two cases, we can find that entropy is maximum if the symbols are equiprobable.

Case (i) Computation of entropy for equiprobable symbols

Let us consider an image $a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Here, the symbols or gray levels 0 and 1 are equiprobable.

Step 1 Computation of probability of occurrence of gray level

$$\text{Probability of occurrence of gray level 0 is } p(0) = \frac{2}{4} = \frac{1}{2}$$

$$\text{Probability of occurrence of gray level 1 is } p(1) = \frac{2}{4} = \frac{1}{2}$$

Step 2 Computation of Entropy $H(S)$

The formula to compute entropy $H(S)$ is given by

$$H(S) = \sum_{i=1}^N p_i \log_2 \left(\frac{1}{p_i} \right) \text{ bits/symbol}$$

In our case, $H(S) = p(0) \log_2 \left(\frac{1}{p(0)} \right) + p(1) \log_2 \left(\frac{1}{p(1)} \right)$

$$H(S) = \frac{1}{2} \log_2 \left(\frac{1}{1/2} \right) + \frac{1}{2} \log_2 \left(\frac{1}{1/2} \right)$$

$$H(S) = \frac{1}{2} \log_2^2 + \frac{1}{2} \log_2^2$$

$$H(S) = \frac{1}{2} + \frac{1}{2} = 1 \text{ bits/symbol}$$

Case (ii) Computation of entropy for non-equiprobable symbols

Let us consider an image $a = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Here, the symbols or gray levels 0 and 1 are not equiprobable.

Step 1 Computation of probability of occurrence of gray level

Probability of occurrence of gray level 0 is $p(0) = \frac{1}{4}$

Probability of occurrence of gray level 1 is given by $p(1) = \frac{3}{4}$

Step 2 Computation of entropy $H(S)$

The formula to compute entropy $H(S)$ is given by

$$H(S) = \sum_{i=1}^N p_i \log_2 \left(\frac{1}{p_i} \right) \text{ bits/symbol}$$

In our case, $H(S) = p(0) \log_2 \left(\frac{1}{p(0)} \right) + p(1) \log_2 \left(\frac{1}{p(1)} \right)$

$$H(S) = \frac{1}{4} \log_2 \left(\frac{1}{1/4} \right) + \frac{3}{4} \log_2 \left(\frac{1}{3/4} \right)$$

$$H(S) = \frac{1}{4} \log_2^{(4)} + \frac{3}{4} \log_2^{(3)}$$

$$H(S) = 0.5 + \frac{3}{4} \log_2 \left[\frac{4}{3} \right]$$

$$H(S) = 0.5 + 0.3113$$

$$H(S) = 0.8113 \text{ bits/symbol}$$

In the case (i), the entropy is found to be 1 bit/symbol, whereas in the case (ii) the entropy is found to be 0.8113 bits/symbol. Thus, entropy is maximum when the symbols are equiprobable.

5. Calculate the efficiency of Huffman code for the following symbol whose probability of occurrence is given below:

Symbol	Probability
a_1	0.9
a_2	0.06
a_3	0.02
a_4	0.02

Solution

Step 1 Construction of Huffman tree

Symbol	Probability	Step 1	Step 2	Code	Length of Code
a_1	0.9	0.9	0.9 (0)	0	1
a_2	0.06	0.06 (0)	0.1	10	2
a_3	0.02	0.02 (0)	0.04 (1)	110	3
a_4	0.02	(1)		111	3

Step 2 To compute the average length

$$\bar{L} = \sum_{k=0}^{N-1} P_k l_k = 0.9 \times 1 + 0.06 \times 2 + 0.02 \times 3 + 0.02 \times 3 = 1.14 \text{ bits/symbol}$$

Step 3 To compute the entropy

Entropy is computed as

$$\begin{aligned}
 H(s) &= - \sum_{k=0}^{N-1} P_k \log_2 P_k = 0.9 \times \log_2 0.9 + 0.06 \times \log_2 0.06 + 0.02 \times \log_2 0.02 + 0.02 \times \log_2 0.02 \\
 &= 0.606 \text{ bits/symbol}
 \end{aligned}$$

Step 4 To compute the efficiency of Huffman code

$$\text{Efficiency of the Huffman code is given by } \eta = \frac{H(s)}{L} = \frac{0.606}{1.14} = 53.15\%$$

This example illustrates the fact that Huffman coding requires at least one bit/symbol to transmit the source. It is important to note that Huffman coding is optimal at the block level and it is not optimal at the symbol-level encoding.

- 6.** For the image shown below, compute the degree of compression that can be achieved using (a) Huffman coding of pixel values, (b) run-length coding, assuming 2 bits to represent the pixel value and 2 bits to represent the run length.

$$\begin{bmatrix} 3 & 3 & 3 & 2 \\ 2 & 3 & 3 & 3 \\ 3 & 2 & 2 & 2 \\ 2 & 1 & 1 & 0 \end{bmatrix}$$

Solution From the given image, it is clear that the maximum value is 3. To encode this maximum value, a minimum of two bits is required. Totally, 16 pixels are present in the given matrix. This implies that a minimum of $16 \times 2 = 32$ bits are required to code this image.

(a) Huffman coding of pixels**Step 1** Probability of occurrence of pixel values

The first step in Huffman coding is to determine the probability of occurrence of each pixel value. From the given image, we get the following:

$$\text{Probability of occurrence of pixel value 0 is } p(0) = \frac{1}{16}$$

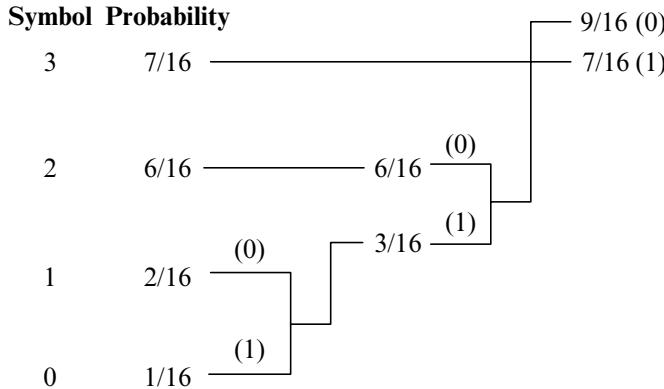
$$\text{Probability of occurrence of pixel value 1 is } p(1) = \frac{2}{16}$$

$$\text{Probability of occurrence of pixel value 2 is } p(2) = \frac{6}{16}$$

$$\text{Probability of occurrence of pixel value 3 is } p(3) = \frac{7}{16}$$

Step 2 Formation of Huffman tree

In Huffman tree formation, the pixels are arranged in descending order.



From the Huffman tree, the code for the symbols are given below:

Code for the symbol 3 is 1.

Code for the symbol 2 is 00

Code for the symbol 1 is 010

Code for the symbol 0 is 011

The mean bits per pixel using Huffman coding is given by $\sum_k p_k l_k$

For this problem, the mean bits per pixel is $\frac{7}{16} \times 1 + \frac{6}{16} \times 2 + \frac{2}{16} \times 3 + \frac{1}{16} \times 3 = \frac{28}{16} = 1.75$

Originally, we are in need of two bits to represent the pixel, but through Huffman coding we are in need of only 1.75 bits to represent the pixel. Hence the compression ratio achieved through Huffman coding is given by

$$\text{compression ratio} = \frac{2}{1.75} = 1.14$$

(b) Run-length coding of pixels

By scanning the image from left to right and from top to bottom, we can write the run length as (3, 3) (2, 2) (3, 4) (2, 4) (1, 2) (0, 1). The first element in the bracket represents the pixel value and the second element represents the number of occurrence of the pixel value. We use two bits to represent the pixel value and two bits to represent the number of occurrences of the pixel value.

This implies the following:

To code (3, 3) we need $2 + 2 = 4$ bits

To code (2, 2) we need $2 + 2 = 4$ bits

To code (3, 4) we need $2 + 2 = 4$ bits

To code (2, 4) we need $2 + 2 = 4$ bits

To code (1, 2) we need $2 + 2 = 4$ bits

To code (0, 1) we need $2 + 2 = 4$ bits

$$g(m, n) = \begin{bmatrix} 55 & 32 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 36 & 2 & 43 \\ 51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\ 52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \end{bmatrix}$$

Step 2 Computation of 2D DCT of the level-shifted image The 2D DCT of the input image $g(m, n)$ is taken to get $G(k, l)$

Step 2a DCT matrix of order 8 The DCT matrix of order 8 is given below:

$$h = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

Now the 2D DCT of $g(m, n)$ is given by

$$G(k, l) = h \times g(m, n) \times h^T$$

$$G(k, l) = \begin{bmatrix} 312.375 & 56.2483 & -26.5839 & 16.5266 & 79.1250 & -60.1429 & 26.3002 & -26.0891 \\ -37.7090 & -27.5825 & 12.7841 & 45.2456 & 30.8015 & -0.6960 & -23.5596 & -10.3478 \\ -19.7657 & -17.6359 & 9.5507 & 33.4645 & 20.5447 & -5.8761 & -15.6683 & -8.8150 \\ -11.2133 & -7.4016 & 9.4054 & 14.8335 & 10.3444 & -10.7465 & -13.4322 & 1.1844 \\ -5.6250 & 0.7573 & 5.9047 & 5.2749 & -4.3750 & -7.1102 & -5.0165 & 4.8331 \\ 2.6574 & 3.1679 & 0.2045 & -2.2738 & -7.1165 & -3.6684 & 0.5683 & 2.4624 \\ 2.9016 & 4.9947 & -0.4183 & -3.6591 & -7.5628 & -0.6655 & 1.9493 & 4.1493 \\ 3.2668 & 0.6250 & -1.0034 & -2.1679 & -2.8821 & -0.6067 & 3.5158 & 0.9174 \end{bmatrix}$$

After the rounding operation, the value of 2D DCT is obtained as

$$G(k, l) = \begin{bmatrix} 312 & 56 & -27 & 17 & 79 & -60 & 26 & -26 \\ -38 & -28 & 13 & 45 & 31 & -1 & -24 & -10 \\ -20 & -18 & 10 & 33 & 21 & -6 & -16 & -9 \\ -11 & -7 & 9 & 15 & 10 & -11 & -13 & 1 \\ -6 & 1 & 6 & 5 & -4 & -7 & -5 & 5 \\ 3 & 3 & 0 & -2 & -7 & -4 & 1 & 2 \\ 3 & 5 & 0 & -4 & -8 & -1 & 2 & 4 \\ 3 & 1 & -1 & -2 & -3 & -1 & 4 & 1 \end{bmatrix}$$

Step 3 Performing quantisation of the DCT matrix The quantisation matrix q is given in the problem. Quantisation is performed by dividing $G(k, l)$ by the quantisation matrix q .

$$Q = \frac{G(k, l)}{q}$$

$$G(k, l) = \begin{bmatrix} 312 & 56 & -27 & 17 & 79 & -60 & 26 & -26 \\ -38 & -28 & 13 & 45 & 31 & -1 & -24 & -10 \\ -20 & -18 & 10 & 33 & 21 & -6 & -16 & -9 \\ -11 & -7 & 9 & 15 & 10 & -11 & -13 & 1 \\ -6 & 1 & 6 & 5 & -4 & -7 & -5 & 5 \\ 3 & 3 & 0 & -2 & -7 & -4 & 1 & 2 \\ 3 & 5 & 0 & -4 & -8 & -1 & 2 & 4 \\ 3 & 1 & -1 & -2 & -3 & -1 & 4 & 1 \end{bmatrix} \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$Q = \begin{bmatrix} 20.1926 & -31.2172 & 43.7535 & -18.2401 & -32.0710 & -65.2438 & 138.4763 & -64.3871 \\ 2.7441 & -4.9311 & -5.6244 & 1.5451 & 5.3137 & 5.1243 & -9.3457 & 3.8013 \\ 2.1067 & -3.7684 & -2.7186 & 0.2575 & 2.9586 & 2.1820 & -3.3606 & 1.1808 \\ 1.4313 & -2.0229 & -2.1408 & 0.4978 & 1.2163 & 1.9514 & -2.4715 & 0.9132 \\ -0.1562 & 0.8078 & -1.5096 & 0.0295 & 0.9160 & 1.9348 & -3.5557 & 1.6504 \\ -0.4282 & 0.9876 & 0.3620 & -0.5393 & -0.1756 & -0.1498 & 0.1285 & 0.0131 \\ -0.6655 & 1.4543 & 0.0690 & -0.2789 & -0.0976 & 0.1943 & -0.8071 & 0.4969 \\ -0.2433 & 0.4091 & 0.7692 & -0.4860 & -0.4472 & -0.6090 & 1.2616 & -0.5713 \end{bmatrix}$$

After the rounding operation, the quantised value is given by

$$Q = \begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 4 ZigZag scanning of the quantised coefficients The output of zig zag scanning is given below:

[20, 5, -3, -1, -2, -3, 1, 1, -1, 0, 0, 1, 2, 3, -2, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, EOB]

After zigzag scan, the dc coefficient is DPCM coded whereas the ac coefficients are mapped to run-level pairs.

Decoding

Step 5 Multiplication of received values with the quantisation matrix

$$R = Q \cdot q$$

$$R = \begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Here \cdot refers to element-by-element multiplication

$$R = \begin{bmatrix} 320 & 55 & -30 & 16 & 72 & -80 & 51 & 0 \\ -36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 \\ -14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 \\ -14 & 0 & 0 & 29 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 6 Computation of the inverse discrete cosine transform In this, inverse discrete cosine transform is taken for R obtained in Step 5.

$$R1 = IDCT\{R\}$$

$$R1 = \begin{bmatrix} 67.2735 & 11.6804 & -8.5522 & 19.5462 & 69.0910 & 42.5608 & -8.1956 & 41.9083 \\ 57.6440 & 24.5649 & 15.1701 & 29.5910 & 64.6460 & 39.5821 & -4.4502 & 47.2064 \\ 46.1672 & 41.2615 & 43.9489 & 40.2064 & 59.3007 & 38.4346 & 0.3200 & 49.0710 \\ 40.9613 & 51.9760 & 59.0237 & 43.2285 & 56.8048 & 42.1119 & 3.1126 & 42.1405 \\ 43.5517 & 54.0780 & 58.1080 & 40.4705 & 57.9657 & 47.1258 & 3.0184 & 32.9064 \\ 49.2704 & 51.9250 & 52.8696 & 40.1731 & 60.7391 & 47.1377 & 1.3273 & 33.011 \\ 53.2433 & 50.4242 & 52.9211 & 46.0941 & 62.8455 & 40.8066 & -0.2434 & 44.8008 \\ 54.5383 & 50.4412 & 56.3301 & 52.8194 & 63.6698 & 34.1625 & -0.9817 & 57.1241 \end{bmatrix}$$

After rounding, the value of $R1$ is obtained as

$$R1 = \begin{bmatrix} 67 & 12 & -9 & 20 & 69 & 43 & -8 & 42 \\ 58 & 25 & 15 & 30 & 65 & 40 & -4 & 47 \\ 46 & 41 & 44 & 40 & 59 & 38 & 0 & 49 \\ 41 & 52 & 59 & 43 & 57 & 42 & 3 & 42 \\ 44 & 54 & 58 & 40 & 58 & 47 & 3 & 33 \\ 49 & 52 & 53 & 40 & 61 & 47 & 1 & 33 \\ 53 & 50 & 53 & 46 & 63 & 41 & 0 & 45 \\ 55 & 50 & 56 & 53 & 64 & 34 & -1 & 57 \end{bmatrix}$$

Step 7 Performing level-shift operation In this process, a value of 128 is added to each and every value of $R1$ to get R . Here, R represents the reconstructed image.

$$R = R1 + 128 = \begin{bmatrix} 67 & 12 & -9 & 20 & 69 & 43 & -8 & 42 \\ 58 & 25 & 15 & 30 & 65 & 40 & -4 & 47 \\ 46 & 41 & 44 & 40 & 59 & 38 & 0 & 49 \\ 41 & 52 & 59 & 43 & 57 & 42 & 3 & 42 \\ 44 & 54 & 58 & 40 & 58 & 47 & 3 & 33 \\ 49 & 52 & 53 & 40 & 61 & 47 & 1 & 33 \\ 53 & 50 & 53 & 46 & 63 & 41 & 0 & 45 \\ 55 & 50 & 56 & 53 & 64 & 34 & -1 & 57 \end{bmatrix} + \begin{bmatrix} 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 & 128 & 128 & 128 & 128 \end{bmatrix}$$

$$R = \begin{bmatrix} 195 & 140 & 119 & 148 & 197 & 171 & 120 & 170 \\ 186 & 153 & 143 & 158 & 193 & 168 & 124 & 175 \\ 174 & 169 & 172 & 168 & 187 & 166 & 128 & 177 \\ 169 & 180 & 187 & 171 & 185 & 170 & 131 & 170 \\ 172 & 182 & 186 & 168 & 186 & 175 & 129 & 161 \\ 177 & 180 & 181 & 168 & 189 & 175 & 129 & 161 \\ 181 & 178 & 181 & 174 & 191 & 169 & 128 & 173 \\ 183 & 178 & 184 & 181 & 192 & 162 & 127 & 185 \end{bmatrix}$$

8. An eight image is given by $X(m,n) = \begin{bmatrix} 62 & 66 & 68 & 70 & 75 & 76 & 78 & 80 \\ 70 & 72 & 65 & 80 & 82 & 84 & 68 & 70 \\ 66 & 68 & 72 & 74 & 66 & 70 & 73 & 65 \\ 72 & 74 & 82 & 84 & 86 & 75 & 60 & 64 \\ 74 & 72 & 80 & 82 & 84 & 78 & 62 & 66 \\ 68 & 67 & 75 & 76 & 76 & 80 & 82 & 84 \\ 65 & 64 & 63 & 60 & 60 & 62 & 66 & 64 \\ 72 & 72 & 65 & 64 & 72 & 71 & 65 & 66 \end{bmatrix}$.

Apply block truncation coding to this input image by dividing the input image into (a) 2×2 block, and (b) 4×4 block. Compare the result of 2×2 BTC with 4×4 in terms of PSNR.

Solution First the image is divided into 2×2 blocks. For each 2×2 block, BTC is applied.

I) BTC using 2×2 block

Step 1 Dividing the input image into 2×2 blocks as shown below.

$$X(m, n) = \begin{bmatrix} \begin{matrix} 62 & 66 \\ 70 & 72 \end{matrix} & \begin{matrix} 68 & 70 \\ 65 & 80 \end{matrix} & \begin{matrix} 75 & 76 \\ 82 & 84 \end{matrix} & \begin{matrix} 78 & 80 \\ 68 & 70 \end{matrix} \\ \begin{matrix} 66 & 68 \\ 72 & 74 \end{matrix} & \begin{matrix} 72 & 74 \\ 82 & 84 \end{matrix} & \begin{matrix} 66 & 70 \\ 86 & 75 \end{matrix} & \begin{matrix} 73 & 65 \\ 60 & 64 \end{matrix} \\ \begin{matrix} 74 & 72 \\ 68 & 67 \end{matrix} & \begin{matrix} 80 & 82 \\ 75 & 76 \end{matrix} & \begin{matrix} 84 & 78 \\ 76 & 80 \end{matrix} & \begin{matrix} 62 & 66 \\ 82 & 84 \end{matrix} \\ \begin{matrix} 65 & 64 \\ 72 & 72 \end{matrix} & \begin{matrix} 63 & 60 \\ 65 & 64 \end{matrix} & \begin{matrix} 60 & 62 \\ 72 & 71 \end{matrix} & \begin{matrix} 66 & 64 \\ 65 & 66 \end{matrix} \end{bmatrix}$$

Totally we will get 16 blocks.

Step 2 Taking the first block as $f(m, n) = \begin{bmatrix} 62 & 66 \\ 70 & 72 \end{bmatrix}$, we have to compute the mean, mean square value and the standard deviation.

(a) Mean value $\bar{f} = \frac{1}{4} \{62 + 66 + 70 + 72\} = 67.5$

(b) The mean square value is given by

$$\bar{f^2} = \frac{1}{4} \times \{3844 + 4356 + 4900 + 5184\} = 4571$$

(c) The variance is computed as

$$\sigma^2 = \bar{f^2} - \bar{f}^2$$

$$\sigma^2 = 14.75. \text{ From this, } \sigma = 3.8406$$

Step 2a Computation of binary allocation matrix $B(m, n)$

$$B(m, n) = \begin{cases} 1: & f(m, n) > \bar{f} \\ 0: & f(m, n) \leq \bar{f} \end{cases}$$

For this problem, $B(m, n) = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$. From the matrix $B(m, n)$, the value of q is found to be two.

Step 2b Computation of a and b values

The value of a is computed using the formula

$$a = \bar{f} - \sigma \sqrt{\frac{q}{m - q}}$$

Substituting the values of $\bar{f} = 67.25$, $q = 2$, $m = 4$ and $\sigma = 3.8406$ in the above expression, we get $a = 63.66$. Upon rounding, we get the value of a as 64.

The value of b is computed using the formula

$$b = \bar{f} + \sigma \sqrt{\frac{m-q}{q}}$$

Substituting the values of $\bar{f} = 67.25$, $q = 2$, $m = 4$ and $\sigma = 3.8406$ in the above expression, we get $b = 71.34$. Upon rounding, we get the value of b as 71.

Step 2c Computation of reconstruction values

The reconstructed block is given as $\hat{f}(m, n) = \begin{cases} a : B(m, n) = 0 \\ b : B(m, n) = 1 \end{cases}$.

$$\hat{f}(m, n) = \begin{bmatrix} 64 & 64 \\ 71 & 71 \end{bmatrix}$$

Step 3 Then, taking the second block we have to determine the reconstruction value

$$f(m, n) = \begin{bmatrix} 68 & 70 \\ 65 & 80 \end{bmatrix}$$

The mean value is computed as $\bar{f} = 70.75$. After computing the mean value, the binary allocation matrix is computed to be $B(m, n) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. The variance is computed as $\sigma^2 = 31.6875$; hence $\sigma = 5.6292$. The value of a is computed as $a = 67.5$. After rounding, it is $a = 68$. The value of b is computed as $b = 80.50$. After rounding, the value is $b = 81$.

The reconstruction value is given by $\hat{f} = \begin{bmatrix} 68 & 68 \\ 68 & 81 \end{bmatrix}$

Step 4 Then, the third block $f(m, n) = \begin{bmatrix} 75 & 76 \\ 82 & 84 \end{bmatrix}$ is taken. The value of mean is computed as $\bar{f} = 79.25$. The binary allocation matrix is obtained as $B(m, n) = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$. The value of σ is computed as $\sigma = 3.8324$. The value of a and b for this block is given as $a = 75$ and $b = 83$. After determining the values of a and b , the reconstruction value is obtained as $\hat{f} = \begin{bmatrix} 75 & 75 \\ 83 & 83 \end{bmatrix}$

Step 5 For the next block, $f(m, n) = \begin{bmatrix} 78 & 80 \\ 68 & 70 \end{bmatrix}$, the mean value is $\bar{f} = 74$, $\sigma = 5.099$, $a = 69$ and $b = 79$. The reconstruction value is $\hat{f} = \begin{bmatrix} 79 & 79 \\ 69 & 69 \end{bmatrix}$.

Step 6 The next block considered is $f(m, n) = \begin{bmatrix} 66 & 68 \\ 72 & 74 \end{bmatrix}$. The mean value is computed as $\bar{f} = 70$.

The value of $a = 67$ and $b = 73$. The reconstruction is obtained as $\hat{f} = \begin{bmatrix} 67 & 67 \\ 73 & 73 \end{bmatrix}$.

Step 7 For the next block $f(m, n) = \begin{bmatrix} 72 & 74 \\ 82 & 84 \end{bmatrix}$, the mean value is $\bar{f} = 78$, $\sigma = 5.099$, $a = 73$ and $b = 83$. The reconstruction value is obtained as $\hat{f} = \begin{bmatrix} 73 & 73 \\ 83 & 83 \end{bmatrix}$.

Step 8 For the next block, $f(m, n) = \begin{bmatrix} 66 & 70 \\ 86 & 75 \end{bmatrix}$, the mean value is $\bar{f} = 74.25$, $\sigma = 7.4958$, $a = 67$ and $b = 82$. The reconstructed value is $\hat{f} = \begin{bmatrix} 67 & 67 \\ 82 & 82 \end{bmatrix}$.

Step 9 For the next block, $f(m, n) = \begin{bmatrix} 73 & 65 \\ 60 & 64 \end{bmatrix}$, the mean value is $\bar{f} = 65.5$, $\sigma = 4.717$, $a = 63$ and $b = 74$. The reconstructed value $\hat{f} = \begin{bmatrix} 74 & 63 \\ 63 & 63 \end{bmatrix}$.

Step 10 For the successive block $f(m, n) = \begin{bmatrix} 74 & 72 \\ 68 & 67 \end{bmatrix}$, the mean value is $\bar{f} = 70.25$, $\sigma = 2.8614$, $a = 67$ and $b = 73$. The reconstructed value is given by $\hat{f} = \begin{bmatrix} 73 & 73 \\ 67 & 67 \end{bmatrix}$.

Step 11 The next block $f(m, n) = \begin{bmatrix} 80 & 82 \\ 75 & 76 \end{bmatrix}$, the mean value $\bar{f} = 78.25$, $\sigma = 2.8614$, $a = 75$, $b = 81$. The reconstructed value is $\hat{f} = \begin{bmatrix} 81 & 81 \\ 75 & 75 \end{bmatrix}$.

Step 12 For the next block $f(m, n) = \begin{bmatrix} 84 & 78 \\ 76 & 80 \end{bmatrix}$, $\bar{f} = 79.5$, $\sigma = 2.96$, $a = 77$ and $b = 82$. The reconstructed value is $\hat{f} = \begin{bmatrix} 82 & 77 \\ 77 & 82 \end{bmatrix}$.

Step 13 For the successive block $f(m, n) = \begin{bmatrix} 62 & 66 \\ 82 & 84 \end{bmatrix}$, the mean value is $\bar{f} = 73.5$, $\sigma = 9.63$, $a = 64$ and $b = 83$. The reconstructed value is obtained as $\hat{f} = \begin{bmatrix} 64 & 64 \\ 83 & 83 \end{bmatrix}$

Step 14 For the next block $f(m, n) = \begin{bmatrix} 65 & 64 \\ 72 & 72 \end{bmatrix}$, the mean value $\bar{f} = 68.25$, $\sigma = 3.799$, $a = 64$, $b = 72$.

The reconstructed value is $\hat{f} = \begin{bmatrix} 64 & 64 \\ 72 & 72 \end{bmatrix}$.

Step 15 For the successive block $f(m, n) = \begin{bmatrix} 63 & 60 \\ 65 & 64 \end{bmatrix}$, the mean value is $\bar{f} = 63$, $\sigma = 1.87$, $a = 61$

and $b = 65$. The reconstructed value is $\hat{f} = \begin{bmatrix} 61 & 61 \\ 65 & 65 \end{bmatrix}$.

Step 16 For the next block $f(m, n) = \begin{bmatrix} 60 & 62 \\ 72 & 71 \end{bmatrix}$, $\bar{f} = 66.25$, $\sigma = 5.309$, $a = 61$, $b = 72$. The recon-

structed value is $\hat{f} = \begin{bmatrix} 61 & 61 \\ 72 & 72 \end{bmatrix}$.

Step 17 For the last block $f(m, n) = \begin{bmatrix} 66 & 64 \\ 65 & 66 \end{bmatrix}$, $\bar{f} = 65.25$, $\sigma = 0.8291$, $a = 64$, $b = 66$. The recon-

structed value as $\hat{f} = \begin{bmatrix} 66 & 64 \\ 64 & 66 \end{bmatrix}$.

After combining all the reconstructed values from steps 2 to 17, we get the reconstructed value using BTC as

$$R = \begin{bmatrix} 64 & 64 & 68 & 68 & 75 & 75 & 74 & 74 \\ 71 & 71 & 68 & 81 & 83 & 83 & 69 & 69 \\ 67 & 67 & 73 & 73 & 67 & 67 & 74 & 63 \\ 73 & 73 & 83 & 83 & 82 & 82 & 63 & 63 \\ 73 & 73 & 81 & 81 & 82 & 77 & 64 & 64 \\ 67 & 67 & 75 & 75 & 77 & 82 & 83 & 83 \\ 64 & 64 & 61 & 61 & 61 & 61 & 66 & 64 \\ 72 & 72 & 65 & 65 & 72 & 72 & 64 & 66 \end{bmatrix}$$

The mean square error between the original image $f(m, n)$ and the reconstructed value R is $MSE = 3.375$ and the peak signal-to-noise ratio is $PSNR = 42.8481$.

II) BTC using 4×4 block

First, the input image is split into 4×4 non-overlapping blocks as shown below.

$$X(m, n) = \begin{bmatrix} \begin{matrix} 62 & 66 & 68 & 70 \\ 70 & 72 & 65 & 80 \\ 66 & 68 & 72 & 74 \\ 72 & 74 & 82 & 84 \end{matrix} & \begin{matrix} 75 & 76 & 78 & 80 \\ 82 & 84 & 68 & 70 \\ 66 & 70 & 73 & 65 \\ 86 & 75 & 60 & 64 \end{matrix} \\ \begin{matrix} 74 & 72 & 80 & 82 \\ 78 & 67 & 75 & 76 \\ 65 & 64 & 63 & 60 \\ 72 & 72 & 65 & 64 \end{matrix} & \begin{matrix} 84 & 78 & 62 & 66 \\ 76 & 80 & 82 & 84 \\ 60 & 62 & 66 & 64 \\ 72 & 71 & 65 & 66 \end{matrix} \end{bmatrix}$$

The block truncation coding is applied to each block individually.

Step 1 For the first block $\begin{bmatrix} 62 & 66 & 68 & 70 \\ 70 & 72 & 65 & 80 \\ 66 & 68 & 72 & 74 \\ 72 & 74 & 82 & 84 \end{bmatrix}$, the mean and the variance are computed as $\bar{f} = 71.5625$

and $\sigma^2 = 35.8715$. The value of the parameters a and b are computed as $a = 66$, $b = 78$. The reconstructed value using BTC procedure is obtained as

$$\begin{bmatrix} 66 & 66 & 66 & 66 \\ 66 & 78 & 66 & 78 \\ 66 & 66 & 78 & 78 \\ 78 & 78 & 78 & 78 \end{bmatrix}$$

Step 2 For the next block $\begin{bmatrix} 75 & 76 & 78 & 80 \\ 82 & 84 & 68 & 70 \\ 66 & 70 & 73 & 65 \\ 86 & 75 & 60 & 64 \end{bmatrix}$, the mean value is $\bar{f} = 73.25$, the variance value is $\sigma^2 = 54.1875$. The value of a is $a = 66$ and b is $b = 80$. The reconstructed value is given by

$$\begin{bmatrix} 80 & 80 & 80 & 80 \\ 80 & 80 & 66 & 66 \\ 66 & 66 & 66 & 66 \\ 80 & 80 & 66 & 66 \end{bmatrix}$$

Step 3 For the third block $\begin{bmatrix} 74 & 72 & 80 & 82 \\ 68 & 67 & 75 & 76 \\ 65 & 64 & 63 & 60 \\ 72 & 72 & 65 & 64 \end{bmatrix}$, the mean value is $\bar{f} = 69.9375$, the variance value is $\sigma^2 = 38.5586$, the parameter a is $a = 64$, the parameter b is $b = 76$. The block is replaced by

$$\begin{bmatrix} 76 & 76 & 76 & 76 \\ 64 & 64 & 76 & 76 \\ 64 & 64 & 64 & 64 \\ 76 & 76 & 64 & 64 \end{bmatrix}$$

Step 4 For the last block $\begin{bmatrix} 84 & 78 & 62 & 66 \\ 76 & 80 & 82 & 84 \\ 60 & 62 & 66 & 64 \\ 72 & 71 & 65 & 66 \end{bmatrix}$, the mean value is $\bar{f} = 71.125$, the variance value is $\sigma^2 = 66.11$, the parameter a is $a = 64$ and $b = 80$. The reconstructed value for this block using BTC approach is

$$\begin{bmatrix} 80 & 80 & 64 & 64 \\ 80 & 80 & 80 & 80 \\ 64 & 64 & 64 & 64 \\ 80 & 64 & 64 & 64 \end{bmatrix}$$

Combining the reconstructed value obtained in steps 1 to 4, we get the reconstructed value as

$$R1 = \begin{bmatrix} 66 & 66 & 66 & 66 & 80 & 80 & 80 & 80 \\ 66 & 78 & 66 & 78 & 80 & 80 & 66 & 66 \\ 66 & 66 & 78 & 78 & 66 & 66 & 66 & 66 \\ 78 & 78 & 78 & 78 & 80 & 80 & 66 & 66 \\ 76 & 76 & 76 & 76 & 80 & 80 & 64 & 64 \\ 64 & 64 & 76 & 76 & 80 & 80 & 80 & 80 \\ 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 76 & 76 & 64 & 64 & 80 & 64 & 64 & 64 \end{bmatrix}$$

The mean square error between the original image $f(m, n)$ and the reconstructed value $R1$ is $MSE = 13.4375$ and the peak signal-to-noise ratio is $PSNR = 36.8476$.

Conclusion If the blocks are chosen as 2×2 , the reconstructed value resembles the original value than if the blocks are chosen as 4×4 , which is evident from the MSE and PSNR values.

9. Consider an image strip of size 100×100 , as shown in Fig. 9.61. The image consists of four vertical stripes. The gray levels of the stripes from left to right are 64, 32, 16 and 8. The corresponding widths of the stripes are 40, 30, 20 and 10 pixels. For this striped image, compute the entropy in bits per pixel.

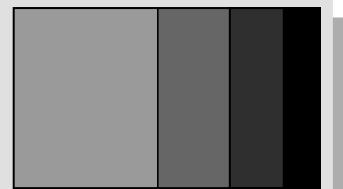


Fig. 9.61 Image strip

Solution The total number of pixels in the image is $100 \times 100 = 10,000$ pixels. The probability of occurrence of each gray level is computed in Table 9.8.

Table 9.8 Probability of occurrence of gray levels

Gray level	Count	Probability
64	$40 \times 100 = 4,000$	$\frac{4000}{10,000} = 0.4$
32	$30 \times 100 = 3,000$	$\frac{3000}{10,000} = 0.3$
16	$20 \times 100 = 2,000$	$\frac{2000}{10,000} = 0.2$
8	$100 \times 100 = 1,000$	$\frac{1000}{10,000} = 0.1$

After computing the probability of occurrence of each gray level, the entropy is given by the formula

$$H(S) = \sum_{i=1}^N p_i \log_2 \left(\frac{1}{p_i} \right) \text{ bits/symbol}$$

$$H(S) = (0.4)(-1.32) + (0.3)(-1.74) + (0.2)(-2.32) + (0.1)(-3.32) = 1.85 \text{ bits/pixel}$$

10. Consider an image strip of size 50×100 shown in Fig. 9.62. The image consists of five vertical stripes. The gray levels of the stripes from left to right are 128, 64, 32, 16 and 8. The corresponding widths of the stripes are 35, 30, 20, 10 and 5 pixels respectively. If this stripe image coded is by Huffman coding, determine its efficiency.

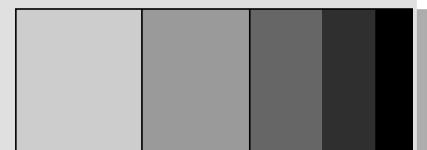


Fig. 9.62 Image strip

Solution The total number of pixels in the image is $50 \times 100 = 5,000$ pixels. Then, the probability of occurrence of each gray level is computed as shown in Table 9.9

Table 9.9 Probability of occurrence of gray levels

Gray level	Count	Probability
128	$50 \times 35 = 1750$	$\frac{1750}{5,000} = 0.35$
64	$50 \times 30 = 1,500$	$\frac{1500}{5000} = 0.3$
32	$50 \times 20 = 1,000$	$\frac{1000}{5,000} = 0.2$
16	$50 \times 10 = 500$	$\frac{500}{5000} = 0.1$
8	$50 \times 5 = 250$	$\frac{250}{5000} = 0.05$

The next step is the construction of the Huffman tree which is shown in Fig. 9.63.

From the Huffman tree, the following codes are obtained for different gray levels which are shown in Table 9.10.

The entropy of the Huffman code is given by

$$H(s) = -\sum_{k=0}^{N-1} P_k \log_2 P_k$$

$$H(s) = -\{0.35 \times (-1.52) + 0.3 \times (-1.74) + 0.2 \times (-2.32) + 0.1 \times (-3.32) + 0.05 \times (-4.32)\}$$

$$H(s) = 2.06 \text{ bits/symbol}$$

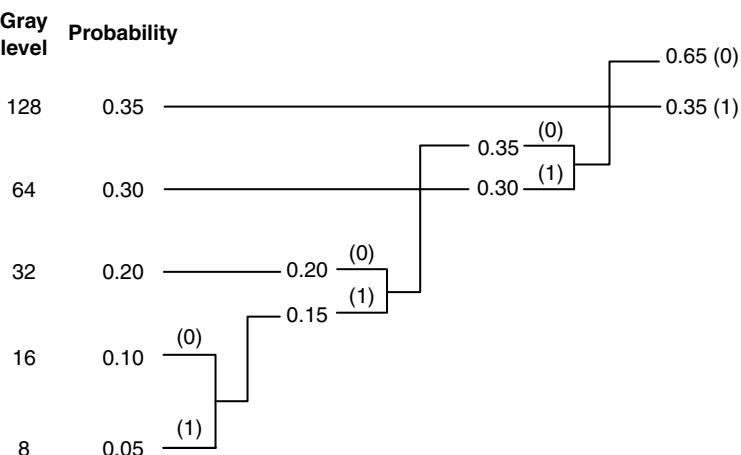


Fig. 9.63 Construction of Huffman tree

The average length is given by

$$\bar{L} = \sum_{k=0}^{N-1} P_k l_k$$

$$\begin{aligned}\bar{L} &= 0.35 \times 1 + 0.30 \times 2 + 0.20 \times 3 + 0.10 \times 4 + 0.05 \times 4 \\ &= 2.15 \text{ bits/symbol.}\end{aligned}$$

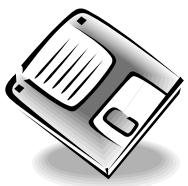
The efficiency of the of the Huffman code is given by

$$\eta = \frac{H(S)}{\bar{L}} = \frac{2.06}{2.15} = 95.81\%$$

Table 9.10 *Huffman code*

Gray level	Code
128	1
64	01
32	000
16	0010
8	0011

Summary



- Compression is compact representation. Image compression deals with the representation of an image with minimum number of bits.
- Compression is achieved by minimising the redundancies in an image. Different types of redundancies in an image are (i) spatial redundancy, (ii) temporal redundancy, and (iii) psychovisual redundancy.
- Two broad classifications of image compression are (i) lossless compression, and (ii) lossy compression. In lossless compression, the reconstructed image exactly resembles the original image. In lossy compression, there is loss of information.
- The following techniques are common in lossless image compression: (a) run-length coding, (ii) Huffman coding, (iii) arithmetic coding, (iv) Shannon–Fano coding, and (vi) Dictionary-based techniques like LZW.
- Quantisation is basically approximation. Quantisation is a non-linear and irreversible process. Quantisation can be broadly classified into (a) scalar quantisation, and (b) vector quantisation.
- Different types of vector quantisation schemes include (i) mean removed vector quantisation, (ii) gain-shape vector quantisation, (iii) multistage vector quantisation, (iv) tree-structure vector quantisation, (v) classified vector quantisation, and (vi) hierarchical vector quantisation.
- The different types of predictive coding techniques are (i) delta coding, (ii) DPCM, and (iii) ADPCM techniques.
- The transform which is widely used in JPEG compression scheme is the Discrete Cosine Transform (DCT). DCT-based image compression is basically a block-based approach and it suffers from blocking artifacts.
- Wavelet-based image-compression scheme overcomes the problem of blocking artifact. Wavelet transform is employed in the JPEG2000 standard.

Review Questions

1. Determine whether the code {0, 01, 11} is uniquely decodable or not.

The prefix property is violated because 0 is a prefix of code word 01. Hence, the code is not uniquely decodable.

2. Explain how zonal coding is different from threshold coding.

In zonal coding, the coefficients with maximum variance over the image sub-blocks are retained. It results in the same zonal masks for all sub-images. In threshold coding, location of the transform coefficients retained varies from one sub-image to another. Only the largest transform coefficients for a sub-image are kept.

3. In transform-based image compression, DCT is widely used than other transforms. Give two reasons for the popularity of DCT in transform-based image compression.

The DCT is performed on smaller blocks of the image, normally 8×8 sized blocks because the basis functions of the DCT are not spatially localised. The DCT is an orthogonal transform. Moreover, the DCT possesses good variance distribution which leads to efficient energy compaction in the transform domain.

4. What is 'blocking artifact' in DCT-based image-compression scheme?

Blocking artifact refers to artificial intensity discontinuities at the borders of neighbouring blocks. The blocking artifact appears because each block is processed independently, with a different quantisation strategy. The blocking artifact is more noticeable at lower bit rates.

5. What is the role of quantisation matrix in JPEG compression?

The quantisation matrix defines the quantisation step size. Step sizes are small for low-frequency components and large for high-frequency components. The quantisation divides the DCT coefficients by the corresponding quantums and then rounds to the nearest integer. In this process, many high-frequency components become zero and therefore, easier to code.

6. Distinguish between scalar and vector quantisation.

Quantising each pixel separately is called scalar quantisation. Quantising group of pixels together is called vector quantisation. Vector quantisation captures the maximum compression that is theoretically possible.

7. Mention the different steps employed in the coding of images using vector quantisation.

The different steps involved in the coding of images using vector quantisation are given below:

- (i) The input image is partitioned into non-overlapping blocks of sizes which may be either 2×2 , 4×4 or 8×8 .
- (ii) For each block, a vector is chosen from the code book that matches the block as closely as possible.
- (iii) The index of the chosen vector is transmitted to the receiver.
- (iv) The decoder extracts the appropriate vector and uses this to represent the original image block.

The crucial step in the vector-quantisation process is the design of the code book.

8. What is slope overload error and granular noise in delta-modulation based image compression?

If the pixel intensities vary faster than Δ , which is the quantiser step size then slope overload occurs. Slope overload manifests itself as blurring of sharp edges in images. One way to overcome slope overload is to

increase the quantiser step size, but this may lead to granular noise. To have a compromise between slope overload error and granular noise, the quantiser step size should be chosen in accordance with the local image characteristic which is termed adaptive delta modulation.

9. Explain the 'fidelity criterion' for lossy image compression.

The fidelity criterion can be broadly classified into (i) objective fidelity criterion, and (ii) subjective fidelity criterion. The most common examples of objective fidelity criterion are (a) root mean square error, and (b) peak signal-to-noise ratio. In subjective fidelity criterion, the image quality will be rated by set of observers.

10. Name a few principal modes of JPEG.

The principal modes of JPEG are (i) sequential mode, (ii) progressive mode, (iii) hierarchical mode, and (iv) lossless mode.

11. Why is zig-zag scanning preferred in JPEG standard?

In JPEG compression, the input image is first divided into 8×8 sub-blocks. For each sub-block, the discrete cosine transform is taken. The spectrum of frequencies falls off with increasing frequency. To make run-length coding more efficient, the highest frequencies should be visited last, since there will be a lot of values closer to zero there. The zig-zag pattern basically goes through 2D frequency space from low to high frequency.

12. Compare the DPCM-based image-compression technique against the transform-based image-compression technique.

- (i) DPCM is simpler than transform-based technique. This is because the linear prediction and differencing operation involved in differential coding are simpler than the 2D transforms involved in transform-based image compression.
- (ii) In terms of memory requirement, processing delay DPCM is superior to the transform-based technique. DPCM needs less memory and less processing delay than transform-based compression.
- (iii) The design of the DPCM system is sensitive to image-to-image variations, whereas transform-based coding is less sensitive to image statistics.

13. What is the fundamental principle of fractal image compression?

Fractal image compression exploits similarities within images. The image is divided into range block and domain block. The smaller partition is called range, and the larger ones are called domains. The domain blocks are transformed to match a given range block as closely as possible.

In fractal image compression, the image is represented by means of a 'fractal' rather than by pixel. Each fractal is the fixed point of an iterated function system. That is, the image is represented by an iterated function system (IFS) of which the fixed point is close to that image. This fixed point is called fractal. Each IFS is then coded as a contractive transformation with coefficients. The coefficients of the transformations are saved as the compressed file.

Problems

- 9.1 List three reasons why image compression is important.
- 9.2 About how long would it take to transmit 800×600 uncompressed colour image that has 8 bits per colour band using a 56 kb modem?
- 9.3 If the original image is 256×256 pixels, 8 bits/pixel, it would occupy 65,536 bytes. After compression it occupies 6554f bytes. Calculate the compression ratio.

9.4 Design (i) Huffman code, and (ii) Shanon–Fano code for the following set of symbols

Symbol	Probability
P	0.4
Q	0.2
R	0.3
S	0.1

9.5 How many principal modes does JPEG have? What are their names?

Compress the following 8 x 8 image using the Huffman code.

0 0 0 0 0 0 0 0
 0 0 1 1 2 3 3 3
 0 1 1 3 3 3 4 4
 0 1 3 3 5 5 4 4
 0 2 3 3 5 5 5 4
 0 0 2 3 3 4 6 6
 0 0 0 2 2 3 4 4
 0 0 0 0 0 0 0 0

9.6 Encode the sentence '**I LOVE IMAGE PROCESSING**' using arithmetic coding procedure.

9.7 Encode the word '**HELLO**' by using the Shannon–Fano coding algorithm. Also, determine the entropy.

9.8 Under which circumstances is arithmetic coding better than Huffman coding and vice versa?

9.9 What is the entropy of the image $f(m, n) = \begin{bmatrix} 20 & 0 & 20 & 0 \\ 40 & 0 & 40 & 0 \\ 0 & 20 & 0 & 20 \\ 0 & 40 & 0 & 40 \end{bmatrix}$?

9.10 An eight bit image is given by $f(m, n) = \begin{bmatrix} 100 & 89 & 60 & 60 & 80 & 70 & 64 & 56 \\ 90 & 40 & 40 & 80 & 20 & 10 & 40 & 20 \\ 75 & 60 & 32 & 40 & 56 & 38 & 42 & 56 \\ 70 & 38 & 36 & 42 & 45 & 56 & 60 & 71 \\ 80 & 23 & 53 & 56 & 41 & 52 & 25 & 62 \\ 70 & 60 & 21 & 63 & 25 & 45 & 28 & 36 \\ 70 & 80 & 75 & 85 & 100 & 96 & 85 & 92 \\ 60 & 56 & 52 & 58 & 45 & 52 & 49 & 38 \end{bmatrix}$.

Apply BTC to this input image by dividing the input image into (a) 2×2 block, and (b) 4×4 block. Compare the result of 2×2 BTC with 4×4 BTC in terms of PSNR.

9.11 Design a code book to encode the image $f(m, n) = \begin{bmatrix} 102 & 104 & 106 & 108 \\ 110 & 111 & 116 & 115 \\ 109 & 103 & 101 & 107 \\ 112 & 114 & 113 & 105 \end{bmatrix}$ using vector quantisation technique by assuming rate = dimension = 2.

9.12 Consider an 8×8 block of an image given by

$$f(m, n) = \begin{bmatrix} 56 & 62 & 54 & 89 & 54 & 65 & 68 & 69 \\ 48 & 46 & 42 & 44 & 41 & 54 & 52 & 52 \\ 21 & 28 & 25 & 26 & 24 & 23 & 28 & 29 \\ 32 & 35 & 36 & 31 & 32 & 34 & 38 & 39 \\ 71 & 75 & 74 & 72 & 73 & 78 & 74 & 75 \\ 84 & 85 & 86 & 84 & 78 & 76 & 65 & 45 \\ 25 & 26 & 24 & 28 & 29 & 31 & 32 & 36 \\ 21 & 25 & 28 & 23 & 24 & 25 & 24 & 28 \end{bmatrix}$$

Determine the quantised DCT block using the luminance and chrominance quantisation table given below.

$$\begin{array}{cccccccc} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{array} \quad \begin{array}{cccccccc} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{array}$$

(a) Luminance

(b) Chrominance

9.13 A 4×4 , four bits per pixel is given by $f(m, n) = \begin{bmatrix} 1 & 0 & 12 & 5 \\ 9 & 9 & 5 & 15 \\ 1 & 5 & 5 & 9 \\ 1 & 9 & 5 & 5 \end{bmatrix}$. Encode the image using the code book provided as

Compute the bit rate and the compression ratio after the encoding process.

Pixel value	Code
0	000001
1	001
2	0000000000
3	0000000001
4	0000000010
5	1
6	0000000011
7	0000000100
8	0000000101
9	01
10	0000000110
11	0000000111
12	0001
13	00000010
14	00000011
15	00001

- 9.14 An image clip is formed using six colours—white (W), red (R), yellow (Y), green (G), blue (B) and orange (O). These occur in the clip with the following relative frequencies:

W	R	Y	G	B	O
0.5	0.1	0.05	0.05	0.2	0.1

For the above data, construct a Huffman code that minimises the average code word length. Also, compute the efficiency of the constructed Huffman code.

References

Books

1. K R Rao and J J Hwang, *Techniques and Standards for Image, Video and Audio Coding*, Prentice Hall PTR, Upper Saddle River, New Jersey
2. Jerry D Gibson, Toby Berger, Tom Lookabaugh, Dave Lindbergh and Richard L Baker, *Digital Compression for Multimedia*, Morgan Kaufmann Publishers
3. M Rabbani and P W Jones, *Digital Image Compression Techniques*, SPIE Press, 1991
4. M F Barnsley, *Fractals Everywhere*, Academic Press, London, 1998
5. M J Turner, J M Blackledge and P R Andrews, *Fractal Geometry in Digital Imaging*, Academic Press, 1998

6. A Gersho and R Gray, *Vector Quantisation and Signal Compression*, Kluwer Academic Publishers, 1992
7. Huseyin Abut, *Vector Quantisation*, IEEE press, 1990
8. W B Pennebaker and J L Mitchell, *JPEG: Still Image Compression Standard*, Kluwer Academic Publishers, 1993

Journal Papers

1. A K Jain, *Image Data Compression: A Review* Proc. IEEE, vol. 69, no. 3, pp. 349–389, 1981
2. P C Cosman, R M Gray and M Vetterli, *Vector Quantisation of Image Subbands: A Survey*, IEEE Transactions on Image Processing, vol. 5, no. 2, pp. 202–225, February 1996
3. C F Barnes, S A Rizvi, and N M Nasrabadi, *Advances in Residual Vector Quantisation: A Review*, IEEE Transactions on Image Processing, vol. 5, no. 2, pp. 226–262, 1996
4. V R Udpikar and J P Raina, *BTC Image Coding using Vector Quantisation*, IEEE Trans. Comm., vol. COM - 35, no. 3, pp. 352–356
5. C E Shannon, *A Mathematical Theory of Communication*, Bell System Technical Journal, vol. 27, pp. 379–423, 1948

Web Resources

1. Professor Bernd Girod's *Image and Video Compression* course page: <http://www.stanford.edu/class/ee398/>
2. Shannon's mathematical theory of communication paper can be obtained from <http://plan9.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf>
3. Johns Hopkins University course material related to image compression and packet video: <http://www.apl.jhu.edu/Notes/Geckle/525759/>
4. Professor Manjunath's course page of *Introduction to Digital Image Processing*: <http://www.ece.ucsb.edu/Faculty/Manjunath/courses/ece178W03/>
5. For JPEG image compression standard, visit www.jpeg.org

10



Learning Objectives

This chapter gives an overview of different operations involved in binary image processing. After reading this chapter, the reader should be familiar with the following concepts

basic idea of mathematical morphology

different mathematical morphological operations

binary image-processing operations like thinning, thickening, bit-or-miss transform, etc.

basic idea of distance transform

Binary Image Processing

10.1 INTRODUCTION

Morphology is the science of appearance, shape and organisation. Mathematical morphology is a collection of non-linear processes which can be applied to an image to remove details smaller than a certain reference shape. The operations of mathematical morphology were originally defined as set operations and shown to be useful for processing sets of 2D points. The morphological operations can be used to extract the edges of an image, filter an image, skeletonise an image, etc. The basic morphological operations, discussed in this chapter, include dilation, erosion, opening and closing, followed by the properties of morphological operations. Even though morphological operations can be performed on binary, grayscale and colour images, our focus in this chapter is to apply different morphological operations to binary images. Binary images having only two gray levels constitute an important subset of digital images. The end of image segmentation operation will be usually a binary image.

10.2 BINARISATION

Binarisation is the process of converting a grayscale image to a black-and-white image. A grayscale image contains a pixel intensity range of 0 to 255 levels. Binarisation is done using global thresholding. Global thresholding sets all pixels above a defined value to white, and the rest of the pixels to black in the image. It is very important to decide the appropriate threshold value to binarise the image, though it is difficult to decide a global value which is suitable for all images. A binary digital image is shown in Fig. 10.1. Each square in the image represents a pixel. Each pixel has a value of either 0 or 1. The black pixel (digit 1) refers the foreground of the image and the white pixel (digit 0) refers the background of the image.

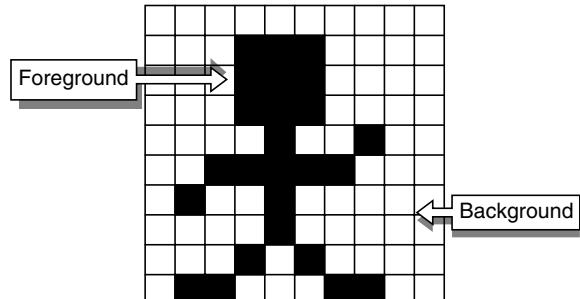


Fig. 10.1 *Binary digital image*

10.3 MATHEMATICAL MORPHOLOGY

Mathematical morphology is one of the most productive areas in image processing. Mathematical morphology is a tool for extracting image components that are useful for representation and description. The technique was originally developed by Matheron and Serra at the Ecole des Mines in Paris. The motivation comes from the collection of structural information about the image domain. The content of mathematical morphology is completely based on set theory. By using set operations, there are many useful operators defined in mathematical morphology. Sets in mathematical morphology represent objects in an image. For example, the black or white pixels as a set in a binary image mean a morphological description of the image. In a binary image, the sets are the members of the 3-D image domain with their integer elements. Each element in the image is represented by a 3-D tuple whose elements are x and y coordinates. Mathematical morphology can be used as the basis for developing image-segmentation procedures with a wide range of applications and it also plays a major role in procedures for image description.

10.4 STRUCTURING ELEMENTS

Mathematical morphology is a collection of non-linear processes which can be applied to an image to remove details smaller than a certain reference shape, which is called *structuring element*. The structuring element in a morphological operation plays an important role with its different shape and size. Shape and size are defined by a number of 0s and 1s in the structuring elements. The circle shown in Fig. 10.2 is called the centre pixel, where the resultant value is applied. This circle can be anywhere in the structuring element according to the user perspective. The examples of structuring element reference are shown in Fig. 10.2.

10.5 MORPHOLOGICAL IMAGE PROCESSING

Morphological operations are defined by moving a structuring element over the binary image to be modified, in such a way that it is centred over every image pixel at some point. When the structuring element is centred over a region of the image, a logical operation is performed on the pixels covered by the structuring element.

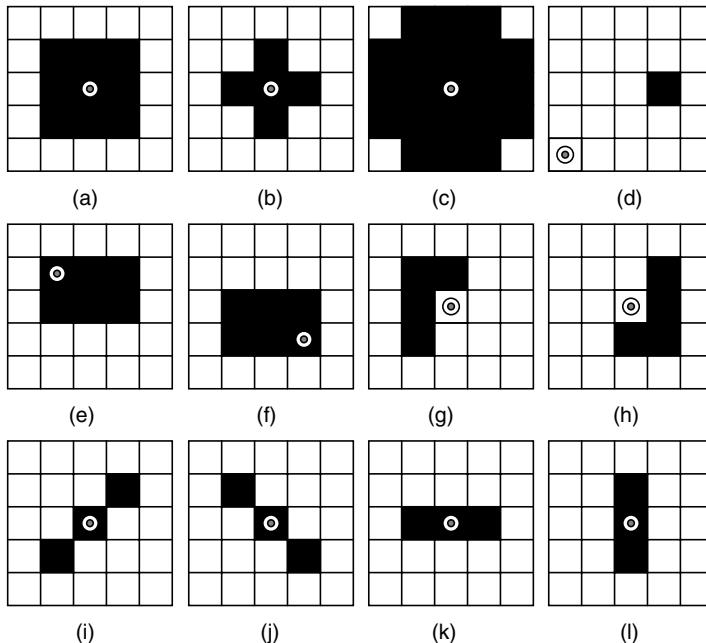


Fig. 10.2 Some possibilities of 5×5 square structuring elements are shown. They are named as (a) N8 (8-Neighbourhood centred) (b) N4 (4-Neighbourhood centred) (c) Flat plus (d) Shifted version (e) 2×3 sized rectangular (f) Reflected structuring element of Fig. (e) (g) L' Shaped structuring element (h) Reflected version of structuring element Fig. (g) (i) line-structuring element of 45° (j) line-structuring element of 135° (k) Horizontal structuring element with size 1×3 (l) Vertical structuring element with size 3×1

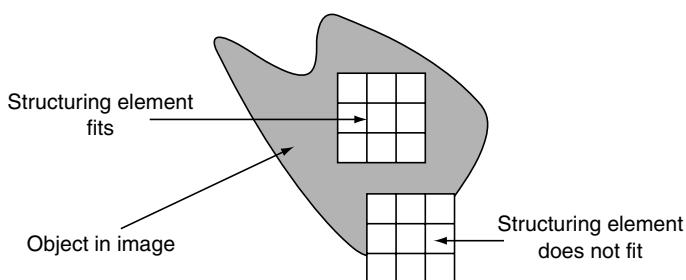


Fig. 10.3 Morphological image processing

yielding a binary output. The morphological image processing is like a convolution process as shown in Fig. 10.3. Like the convolution kernel, the structuring element can be of any size, and it can contain complements of 1s and 0s. At each pixel position, a specified logical operation is performed between the structuring element and the underlying binary image. The binary result of that logical operation is stored in the output image at that pixel position. The effect created depends upon the size and content of the structuring element and the nature of the logical operation. The binary image and structuring element sets need not be restricted to sets in the 2D plane, but could be defined in 1, 2, 3 (or higher) dimensions. If the structuring element is perfectly fit on to the binary image then perform the logical operation; else do not perform any operation into resultant binary image pixel.

10.6 BASIC SET THEORY

Morphology is based on set theory and this section deals with the necessary operations in set theory like union, intersection, complement, reflection and difference.

10.6.1 Union

The union of images A and B is written as $A \cup B$. $A \cup B$ is the set whose elements are either the elements of A or B or of both. In the predicate notation, the definition is given by

$$A \cup B = \text{def } \{x \mid x \in A \text{ or } x \in B\} \quad (10.1)$$

The union of images A and B is shown in Fig. 10.4.

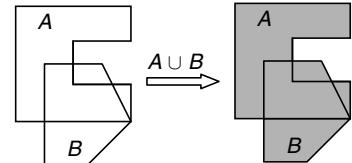


Fig. 10.4 Union operation of images A and B

10.6.2 Intersection

The intersection of images A and B is written as $A \cap B$. $A \cap B$ is the image whose elements are common to both images A and B . The expression of intersection operation is given by

$$A \cap B = \text{def } \{x \mid x \in A \text{ and } x \in B\} \quad (10.2)$$

The intersection of images A and B is shown in Fig. 10.5.

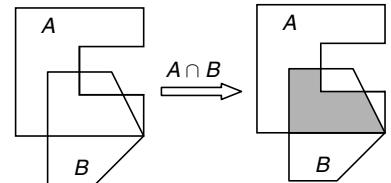


Fig. 10.5 Intersection of images A and B

10.6.3 Difference

Another binary operation on arbitrary images is the difference, Image A minus Image B , represented as $A - B$. $A - B$ subtracts from Image A all elements which are in Image B . The definition is given by

$$A - B = \text{def } \{x \mid x \in A \text{ and } x \notin B\} \quad (10.3)$$

The result of the difference between images A and B is shown in Fig. 10.6. $A - B$ is also called the relative complement of B relative to A .

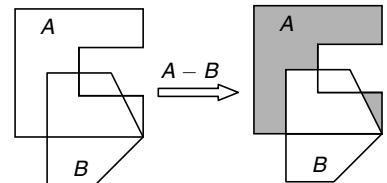


Fig. 10.6 Difference of images A and B

10.6.4 Complement

The complement of Image A is written as A^c , which is the set consisting of everything not in Image A . The definition of complement operation is represented by

$$A^c = \text{def } \{x \mid x \notin A\} \quad (10.4)$$

Equation (10.4) describes the complement operation. The result of the complement operation is shown in Fig. 10.7.

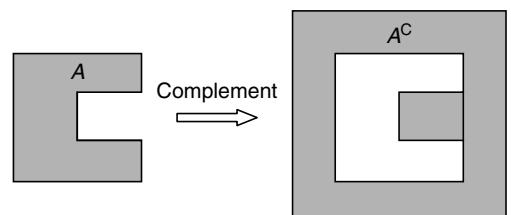


Fig. 10.7 Complement of Image A

10.6.5 Reflection

Reflection of Image B is defined as

$$B_1 = \{w \mid w = -b \text{ for } b \in B\} \quad (10.5)$$

The reflection operation result is shown in Fig. 10.8.

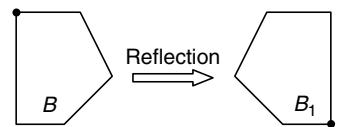


Fig. 10.8 Reflection of Image B

10.6.6 Translation

Translation operation of Image A is defined as

$$A_1 = \{c \mid c = a + z \text{ for } a \in A\} \quad (10.6)$$

The translation operation shifts the origin of Image A to some (z_1, z_2) points. This is illustrated in Fig. 10.9.

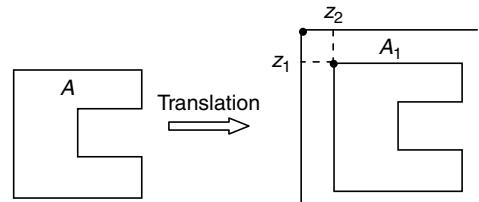


Fig. 10.9 Translation of Image A

10.7 LOGICAL OPERATIONS

The logical operations to be discussed in this section include AND, OR, NOT, EXOR operations, etc.

10.7.1 OR Operation

The OR operation is similar to the union operation and the OR operation result of images A and B is shown in Fig. 10.10.

10.7.2 AND Operation

The AND logic operation is similar to the intersection operation of set theory and the AND operation of two images is shown in Fig. 10.11. The AND logic operation computes the common area of images A and B as a resultant image.

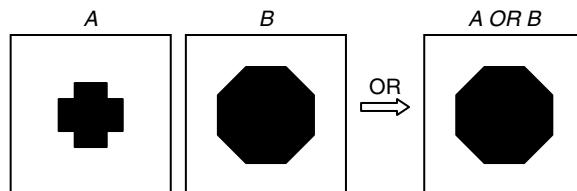


Fig. 10.10 OR operation of images A and B

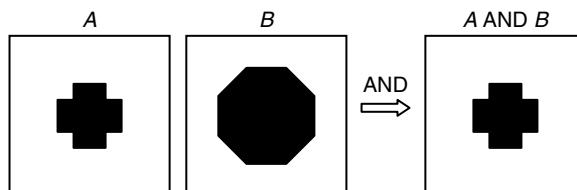


Fig. 10.11 AND operation of images A and B

10.7.3 NOT Operation

The NOT operation is similar to the complement function of set theory. In the NOT operation, the image pixel 1 is changed to 0 and vice versa. The NOT operation of Image A is shown in Fig. 10.12.

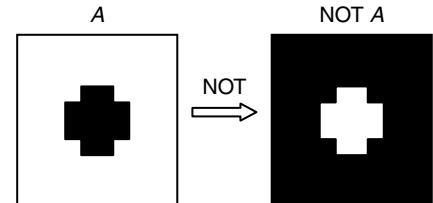


Fig. 10.12 NOT operation of Image A

10.7.4 XOR Operation

In the XOR operation, if the pixels of Image A and Image B are complementary to each other then the resultant image pixel is black, otherwise the resultant image pixel is white. The XOR operation of images A and B as shown in Fig. 10.13.

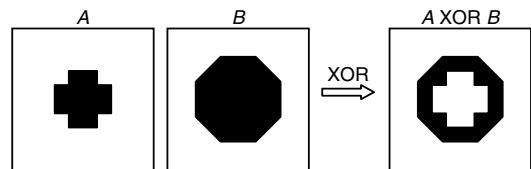


Fig. 10.13 OR operation of images A and B

10.8 STANDARD BINARY MORPHOLOGICAL OPERATIONS

The basic morphological operations are *dilation* and *erosion*. They are expressed by a kernel operating on an input binary image, X , where white pixels denote uniform regions and black pixels denote region boundaries. Erosion and dilation work conceptually by translating a structuring element, B , over the image points, and examining the intersection between the translated kernel coordinates and image coordinates.

10.8.1 Dilation

Dilation is a process in which the binary image is expanded from its original shape. The way the binary image is expanded is determined by the structuring element. This structuring element is smaller in size compared to the image itself, and normally the size used for the structuring element is 3×3 . The dilation process is similar to the convolution process, that is, the structuring element is reflected and shifted from left to right and from top to bottom, at each shift; the process will look for any overlapping similar pixels between the structuring element and that of the binary image. If there exists an overlapping then the pixels under the centre position of the structuring element will be turned to 1 or black.

Let us define X as the reference image and B as the structuring element. The dilation operation is defined by Eq. (10.7)

$$X \oplus B = \left\{ z \left| \left[\left(\hat{B} \right)_z \cap X \right] \subseteq X \right. \right\} \quad (10.7)$$

where \hat{B} is the image B rotated about the origin. Equation 10.7 states that when the image X is dilated by the structuring element B , the outcome element z would be that there will be at least one element in B that intersects with an element in X . If this is the case, the position where the structuring element is being centred on the image will be 'ON'. This process is illustrated in Fig. 10.14. The black square represents 1 and the white square represents 0.

Initially, the centre of the structuring element is aligned at position *. At this point, there is no overlapping between the black squares of B and the black squares of X ; hence at position * the square will remain white. This structuring element will then be shifted towards right. At position **, we find that one of the black squares of B is overlapping or intersecting with the black square of X . Thus, at position ** the square will be

changed to black. Similarly, the structuring element B is shifted from left to right and from top to bottom on the image X to yield the dilated image as shown in Fig. 10.14.

The dilation is an expansion operator that enlarges binary objects. Dilation has many uses, but the major one is bridging gaps in an image, due to the fact that B is expanding the features of X .

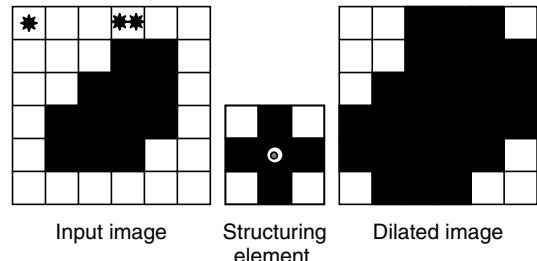
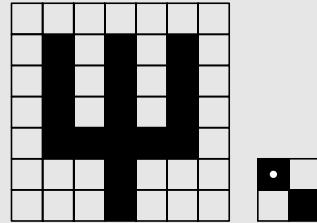


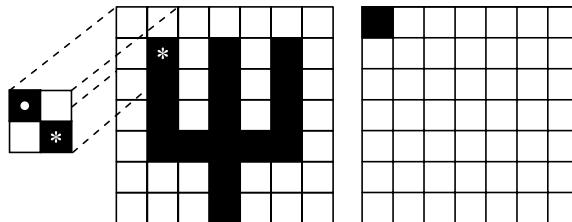
Fig. 10.14 Dilation process

Example 10.1 Using the input image and structuring element as given below, find the dilated version of the input image.

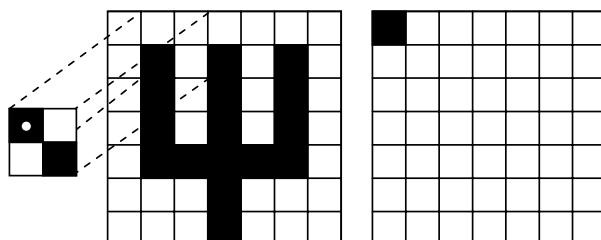


Solution

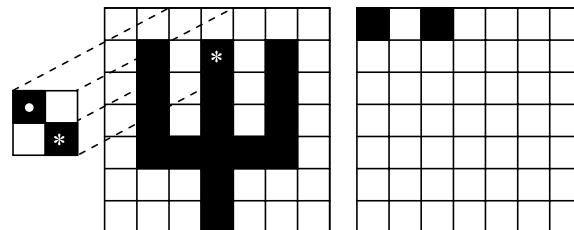
Step 1 Compare the structuring element with the input image at the centre point. An overlapping component between the structuring element and the input image is marked as *. Hence the origin of the structuring element corresponding to the input pixel is turned on.



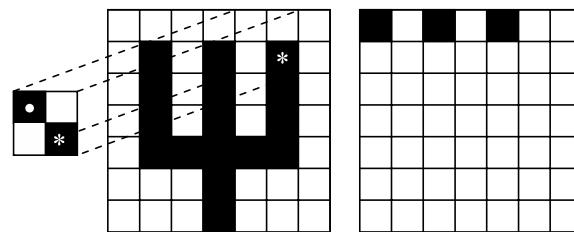
Step 2 Now, shift the structuring element over the input image from left to right and the position of the structuring element, as shown below. There is no overlapping between these two. So, the structuring element origin corresponding to the input pixel is turned to off. Therefore, there is no change for the previous output image.



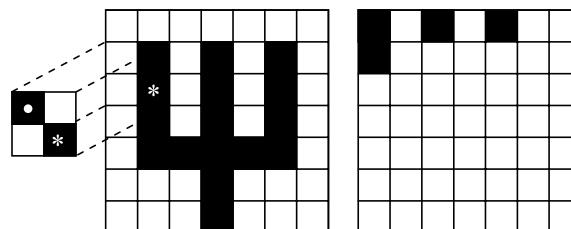
Step 3 Similar to the previous step, the structuring element is shifted towards the right side of the input image to find if any overlapping is present. In this case, overlapping is present, and so the corresponding input pixel is turned to on in the resultant image and is shown as follows:



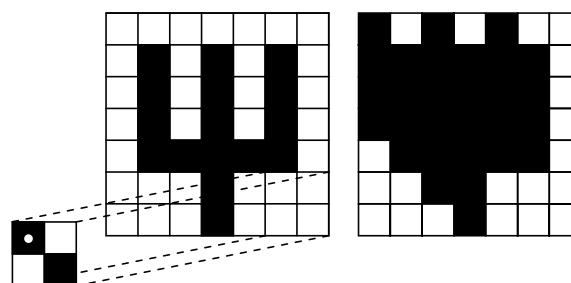
Step 4 Now the structuring element is again shifted to the right side of the input image to find any overlapping. There is an overlapping and the resultant image is shown below.



Step 5 After one shift towards the right, there is no overlapping, and hence the output image will not be changed. The next shift over the input image is on the boundary, and so this will not be matched with the structuring element. Hence, the next shift is from top to bottom which is shown below.



Step 6 The above process is repeated to get the final output image which is shown below.



10.8.2 Erosion

Erosion is the counter-process of dilation. If dilation enlarges an image then erosion shrinks the image. The way the image is shrunk is determined by the structuring element. The structuring element is normally smaller than the image with a 3×3 size. This will ensure faster computation time when compared to larger structuring-element size. Almost similar to the dilation process, the erosion process will move the structuring element from left to right and top to bottom. At the centre position, indicated by the centre of the structuring element, the process will look for whether there is a complete overlap with the structuring element or not. If there is no complete overlapping then the centre pixel indicated by the centre of the structuring element will be set white or 0.

Let us define X as the reference binary image and B as the structuring element. Erosion is defined by the equation

$$X \ominus B = \{z \mid (B)_z \subseteq X\} \quad (10.8)$$

Equation (10.8) states that the outcome element z is considered only when the structuring element is a subset or equal to the binary image X . This process is depicted in Fig. 10.15. Again, the white square indicates 0 and the black square indicates 1.

The erosion process starts at position *. Here, there is no complete overlapping, and so the pixel at the position * will remain white. The structuring element is then shifted to the right and the same condition is observed. At position **, complete overlapping is not present; thus, the black square marked with ** will be turned to white. The structuring element is then shifted further until its centre reaches the position marked by ***. Here, we see that the overlapping is complete, that is, all the black squares in the structuring element overlap with the black squares in the image. Hence, the centre of the structuring element corresponding to the image will be black. Figure 10.16 shows the result after the structuring element has reached the last pixel.

Erosion is a thinning operator that shrinks an image. By applying erosion to an image, narrow regions can be eliminated, while wider ones are thinned.

Example 10.2 The input image and structuring element are given below. Find the eroded version of the input image.

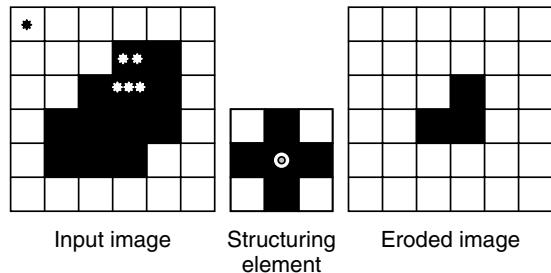
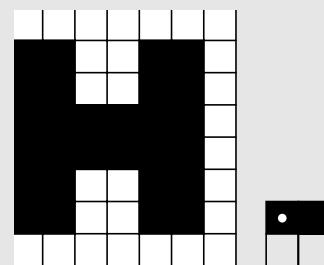
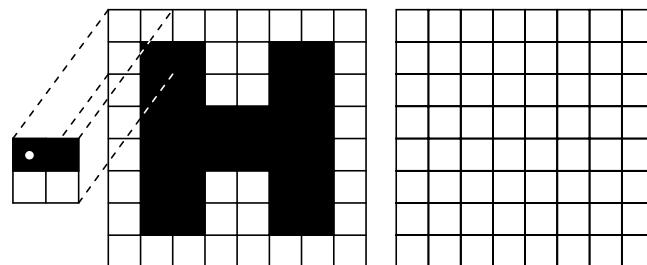


Fig. 10.15 Erosion process

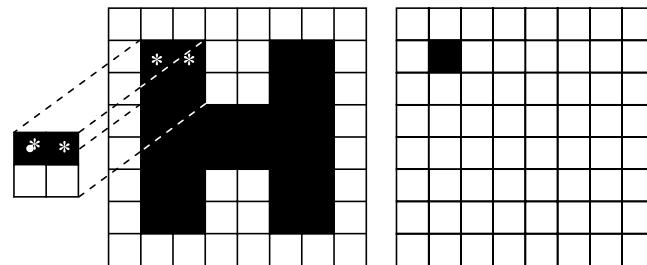


Solution

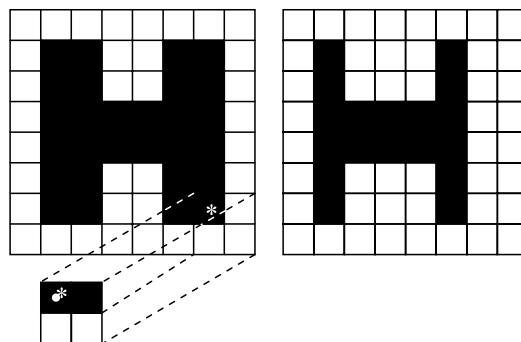
Step 1 The structuring element is compared with the input image. If all the pixels in the structuring element is overlapped with the input image then the origin corresponding to the input image pixel is turned ON. Otherwise that pixel is turned OFF. In our case, the structuring element is shifted from left to right. Initially, there are no pixels overlapping between the structuring element and input image, so the output image is shown below for first iteration. In the first row, the input image is not perfectly overlapped with the structuring element. Hence, all the pixels of the output image in the first row are OFF.



Step 2 Now the structuring element is mapped with the second row of the input image. Initially, there is no perfect matching; hence shift one pixel towards the right. After one shift towards the right, we find a perfect match, and hence the output image pixel is turned black which is illustrated below.



Step 3 The above process is repeated till the last pixel in the input image is taken into account. The eroded image after reaching the last pixel in the input image is shown below.



MATLAB Example 1 For the image shown in Fig. 10.16, perform the dilation and erosion process.



Fig. 10.16 Input image

Solution The MATLAB command to perform the dilation and erosion operation on the image shown in Fig. 10.16 is shown in Fig. 10.17, and the corresponding output is shown in Fig. 10.18.

```
close all;
clear all;
clc;
a = imread('morph1.bmp');
b = [1 1 1;1 1 1;1 1 1];
a1 = imdilate(a, b);
a2 = imerode(a, b);
imshow(a), title('Original image')
figure, imshow(a1), title('Dilated image')
figure, imshow(a2), title('Eroded image')
```

Fig. 10.17 MATLAB code for the dilation and erosion process



Fig. 10.18 Resultant image of dilation and erosion process

10.9 DILATION- AND EROSION-BASED OPERATIONS

Erosion and dilation can be combined to solve specific filtering tasks. The most widely used combinations are opening, closing and boundary detection.

10.9.1 Opening

Opening is based on the morphological operations, erosion and dilation. Opening smoothes the inside of the object contour, breaks narrow strips and eliminates thin portions of the image. It is done by first applying erosion and then dilation operations on the image.

The opening operation is used to remove noise and CCD defects in the images. The opening filters details and simplifies images by rounding corners from inside the object where the kernel uses fits.

The opening process can be mathematically represented as

$$X \circ B = (X \ominus B) \oplus B \quad (10.9)$$

where X is an input image and B is a structuring element.

10.9.2 Closing

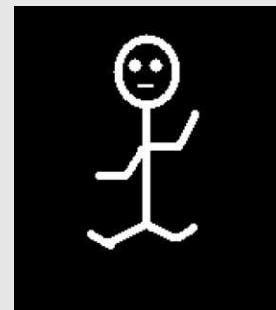
The closing operation is the opposite of the opening operation. It is a dilation operation followed by an erosion operation. The closing operation fills the small holes and gaps in a single-pixel object. It has the same effect of an opening operation, in that it smoothes contours and maintains shapes and sizes of objects.

The closing process can be mathematically represented as

$$X \bullet B = (X \oplus B) \ominus B \quad (10.10)$$

where X is an input image and B is the structuring element. Closing protects coarse structures, closes small gaps and rounds off concave corners.

MATLAB Example 2 Write the MATLAB code to perform an opening and closing operation on the image shown below.



Solution The MATLAB program that performs the opening and closing operation of the input image is shown in Fig. 10.19, and the corresponding results are shown in Fig. 10.20.

```

close all;
clear all;
clc;
a=imread(morph2.bmp);
b=[1 1 1;1 1 1;1 1 1];
a1=imopen(a, b);
a2=imclose(a, b);
imshow(a), title(Original image)
Fig., imshow(a1), title(Opened image)
Fig., imshow(a2), title(Closed image)

```

Fig. 10.19 MATLAB code for opening and closing operation

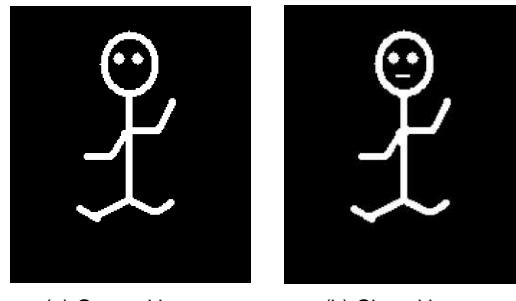


Fig. 10.20 Resultant images of opening and closing operations

10.10 PROPERTIES OF MORPHOLOGICAL OPERATION

The principal properties of the morphological operations are discussed in this section. These properties are important in order to understand the effects of erosion, dilation, opening and closing operations.

10.10.1 Increasing

Let X and Y be two images. Let the image X be smaller than the image Y , that is, $X(m, n) \leq Y(m, n)$, for all the values of m and n . Then

$$X \Theta B \leq Y \Theta B \quad (10.11)$$

$$X \oplus B \leq Y \oplus B \quad (10.12)$$

$$X \circ B \leq Y \circ B \quad (10.13)$$

$$X \bullet B \leq Y \bullet B \quad (10.14)$$

Erosion and dilation are said to be increasing operations. Since erosion and dilation are increasing operations, opening and closing are also increasing operations.

The binary images X , Y , and the structural element B are shown in Fig. 10.21.

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

X

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

Y

0	1	0
1	1	1
0	1	0

B

Fig. 10.21 *Binary image x and X; structuring element B*

The MATLAB code to demonstrate the increasing property of a morphological operation is shown in Fig. 10.22.

Equation (10.11) shows the increasing property of an erosion operation and the resultant images are shown in Fig. 10.23 (a) and 10.23 (b). Equation (10.12) represents the increasing property of dilation operation and the resultant images are shown in Fig. 10.23 (c) and Fig. 10.23 (d).

```

X = [0 1 1 0;1 0 0 1;1 0 0 1;0 1 1 0];
Y = [1 1 1 1 0 0;0 1 1 1 1 0;0 1 1 1 1 0;0 1 1 1 0 0;0 1 0 1 0 1
0 0; 1 1 0 0 1 1];
B = [0 1 0;1 1 1;0 1 0];
ErX = imerode(X, B);
ErX = imerode(Y, B);
Dix = imdilate(X, B);
DiX = imdilate(Y, B);
OpX = imopen(X, B);
OpX = imopen(Y, B);
Clx = imclose(X, B);
ClX = imclose(Y, B);
disp(Eroded image of X);
disp(Erx);
disp(Eroded image of Y);
disp(ErX);
disp(Dilated image of X);
disp(Dix);
disp(Dilated image of Y);
disp(DiX);
disp(Opened image of X);
disp(OpX);
disp(Opened image of Y);
disp(OpX);
disp(closed image of X);
disp(Clx);
disp(closed image of Y);
disp(ClX);

```

Fig. 10.22 *MATLAB code to prove the increasing property*

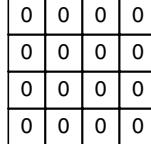
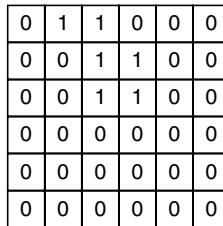
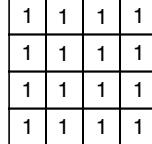
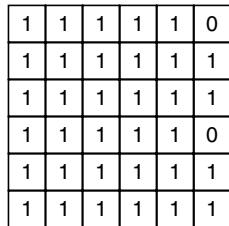
			
(a)	(b)	(c)	(d)

Fig. 10.23 Result for increasing property (a) Eroded image X (b) Eroded image Y (c) Dilated image X (d) Dilated image Y

Equation (10.13) shows the increasing property of an opening operation and the resultant images are shown in Fig. 10.24 (a) and 10.24 (b). Equation (10.14) represents the increasing property of closing operation and the resultant images are shown in Fig. 10.24 (c) and Fig. 10.24 (d).

From the above result, the number of one's present in the processed image Y is more than the processed image X .

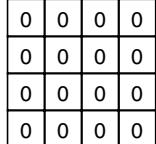
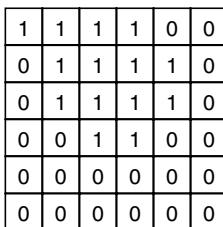
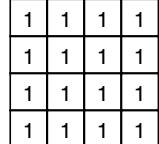
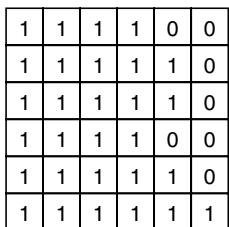
			
(a)	(b)	(c)	(d)

Fig. 10.24 Result for increasing property (a) Opened image X (b) Opened image Y (c) Closed image X (d) Closed image Y

10.10.2 Expansivity

The expansivity of the morphological operations is as follows:

$$X \ominus B \leq X \quad (10.15)$$

$$X \oplus B \geq X \quad (10.16)$$

$$X \circ B \leq X \quad (10.17)$$

$$X \bullet B \geq X \quad (10.18)$$

The MATLAB code to prove expansivity property is shown in Fig. 10.26. The input image and structuring element are shown in Fig. 10.25.

Erosion is anti-expansive, because the output of the eroded image is not expansive, i.e., the number of ones present in the output eroded image is less compared to the original image as shown in Fig. 10.27(a).

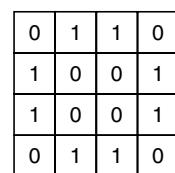
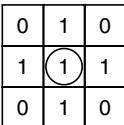
	
X	B

Fig. 10.25 Binary image and structuring element

```

close all;
clear all;
clc;
X = [0 1 1 0;1 0 0 1;1 0 0 1;0 1 1 0];
B = [0 1 0;1 1 1;0 1 0];
ErX = imerode(X, B);
Dix = imdilate(X, B);
OpX = imopen(X, B);
ClX = imclose(X, B);
disp(Eroded image of X);
disp(ErX);
disp(Dilated image of X);
disp(Dix);
disp(Opened image of X);
disp(OpX);
disp(Closed image of X);
disp(ClX);

```

Fig. 10.26 MATLAB code for expansivity property

Dilation is expansive. Here, the resultant image of dilation operation is expanded when compared to the input image as illustrated in Fig. 10.27 (b).

The opening operation is anti-expansive, as is its first constituent operation, erosion. The resultant image is shown in Fig. 10.27 (c). Closing is expansive, as is its first operation dilation. The resultant image after closing is illustrated in Fig. 10.27 (d).

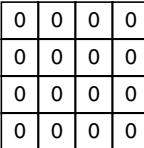
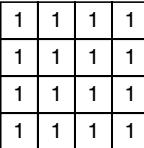
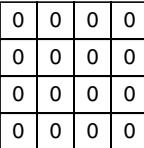
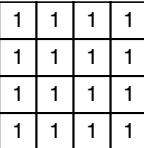
			
(a)	(b)	(c)	(d)

Fig. 10.27 Resulted image for expansivity property

10.10.3 Duality

The next property of binary morphological operations is duality. The property of duality is as follows:

$$X \Theta B \leq (X^c \oplus B)^c \quad (10.19)$$

$$X \oplus B \leq (X^c \Theta B)^c \quad (10.20)$$

$$X \circ B \leq (X^c \bullet B)^c \quad (10.21)$$

$$X \bullet B \leq (X^c \circ B)^c \quad (10.22)$$

From the Eqs (10.19) and (10.20), dilation is the dual of erosion. Thus, dilation is the erosion of the complemented function and vice versa. Here, the input image and the structuring element are shown in Fig. 10.28. The MATLAB code and the resultant of this property are shown in Figs. 10.29 and 10.30 respectively.

Similarly, opening and closing are dual operations. The closing of X corresponds to the opening of the complemented image X^c . The dual operation of the opening and closing MATLAB code is shown in Fig. 10.29, and the resultant image is shown in Fig. 10.31(c).

X	B

Fig. 10.28 Binary image and structuring element

```

X = [0 1 1 0;1 0 0 1;1 0 0 1;0 1 1 0];
Xc = [1 0 0 1;0 1 1 0;0 1 1 0;1 0 0 1];
B = [0 1 0;1 1 1;0 1 0];
c = ones(4, 4);
Er1 = imerode(X, B);
Dil1 = imdilate(Xc, B);
Dif1 = c-Dil1;
Di2 = imdilate(X, B);
Er2 = imerode(Xc, B);
Erf2 = c-Er2;
Op3 = imopen(X, B);
Cl3 = imclose(Xc, B);
Clf3 = c-Cl3;
Cl4 = imclose(X, B);
Op4 = imopen(Xc, B);
Opf4 = c-Op4;
disp(Eroded image of X);
disp(Er1);
disp(Complement Dilated image of Xc);
disp(Dif1);
disp(Dilated image of X);
disp(Di2);
disp(Complement Eroded image of Xc);
disp(Erf2);
disp(Opened image of X);
disp(Op3);
disp(Complement closed image of Xc);
disp(Clf3);
disp(closed image of X);
disp(Cl4);
disp(Complement opened image of Xc);
disp(Opf4);

```

Fig. 10.29 MATLAB code for duality property

<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	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																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
(a)	(b)	(c)	(d)																																																																

Fig. 10.30 Results of duality property of erosion and dilation

<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	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																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
(a)	(b)	(c)	(d)																																																																

Fig. 10.31 Results of duality property of opening and closing

10.10.4 Chain Rule

The chain rule of the morphological operations is as follows:

$$(X \Theta B_1) \Theta B_2 = X \Theta (B_1 \oplus B_2) \quad (10.23)$$

$$(X \oplus B_1) \oplus B_2 = X \oplus (B_1 \oplus B_2) \quad (10.24)$$

Equations (10.23) and (10.24) imply that the erosion or dilation of an image X by a wide or complex structural element B ($\equiv B_1 \oplus B_2$) can be performed using the basic components of B , namely, B_1 and B_2 . The input image X and the structuring elements B_1 and B_2 are shown in Fig. 10.32. The MATLAB code for the chain-rule property is shown in Fig. 10.33.

The resultant images of the chain-rule property are illustrated in Fig. 10.34. Figure 10.34(a) indicates the result of the LHS of Eq. 10.23, and 10.35(b) indicates the result of the RHS of Eq. 10.23. Figure 10.34(c) indicates the result of the LHS of Eq. 10.24, and 10.35(d) indicates the result of the RHS of Eq. 10.24.

<table border="1"><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	<table border="1"><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr></table>	0	1	0	1	1	1	0	1	0	<table border="1"><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1
0	1	1	0																																	
1	0	0	1																																	
1	0	0	1																																	
0	1	1	0																																	
0	1	0																																		
1	1	1																																		
0	1	0																																		
1	1	1																																		
1	1	1																																		
1	1	1																																		
X	B_1	B_2																																		

Fig. 10.32 Binary image and structuring element

10.10.5 Idempotency

The last property of binary morphological operations is idempotency. The idempotency property is defined as follows:

$$(X \circ B) \circ B = X \circ B \quad (10.25)$$

$$(X \bullet B) \bullet B = X \bullet B \quad (10.26)$$

```

close all;
clear all;
clc;
X = [0 1 1 0;1 0 0 1;1 0 0 1;0 1 1 0];
B1 = [0 1 0;1 1 1;0 1 0];
B2 = [1 1 1;1 1 1;1 1 1];
Erx = imerode(X, B1);
Erf = imerode(Erx, B2);
disp(Chain rule1 LHS);
disp(Erf);
DiB = imdilate(B1, B2);
Erx1 = imerode(X, DiB);
disp(Chain rule1 RHS);
disp(Erx1);
Dix1 = imdilate(X, B1);
Dix = imdilate(Dix1, B2);
disp(Chain rule2 LHS);
disp(Dix);
DiB = imdilate(B1, B2);
Dix2 = imdilate(X, DiB);
disp(Chain rule2 RHS);
disp(Dix2);

```

Fig. 10.33 MATLAB code for chain-rule property

<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	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																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
1	1	1	1																																																																
(a)	(b)	(c)	(d)																																																																

Fig. 10.34 Results of chain rule

Opening and closing operations are performed only once for a specific structural element. There is no equivalence to the chain rules of erosion and dilation. Therefore, opening and closing are two ways to compute a ‘smooth’ approximation of an image X by removing details of a given size. Applying these operations again with the same structural element B does not further modify the filtered image.

The idempotency property can be verified using the input image X and the structuring element B shown in Fig. 10.35. The corresponding MATLAB code is shown in Fig. 10.36. The resultant images are illustrated in Fig. 10.37.

<table border="1"><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0	<table border="1"><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr></table>	0	1	0	1	1	1	0	1	0
0	1	1	0																							
1	0	0	1																							
1	0	0	1																							
0	1	1	0																							
0	1	0																								
1	1	1																								
0	1	0																								
X	B																									

Fig. 10.35 Binary image and structuring element

```

close all;
clear all;
clc;
x = [0 1 1 0;1 0 0 1;1 0 0 1;0 1 1 0];
B = [0 1 0;1 1 1;0 1 0];
ErX = imerode(x, B);
Erf = imerode(Erx, B);
disp(Idempotency1 LHS);
disp(Erf);
ErX1 = imerode(x, B);
disp(Idempotency1 RHS);
disp(Erx1);
Dix1 = imdilate(x, B);
Dix = imdilate(Dix1, B);
disp(Idempotency2 LHS);
disp(Dix);
Dix2 = imdilate(x, B);
disp(Idempotency2 RHS);
disp(Dix2);

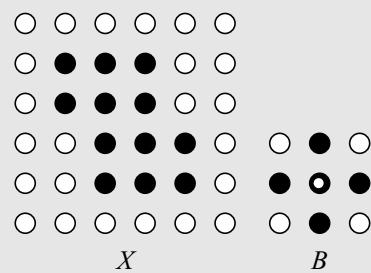
```

Fig. 10.36 MATLAB program for idempotency

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

Fig. 10.37 Results for idempotency

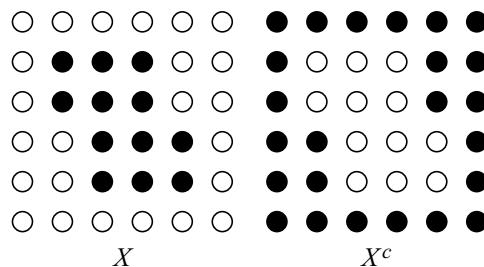
Example 10.3 A binary image X and the structuring element are given as follows. (i) Calculate X^c . (ii) Calculate $Y_1 = X \oplus B$. (iii) Calculate $Y_2 = X^c \ominus B$. (iv) Calculate $Y_3 = X \ominus B$ (v) Calculate $Y_4 = X^c \oplus B$. (vi) What is the relationship between Y_3 and Y_4 ?



Solution

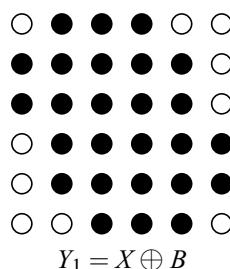
Step 1 To calculate the complement of X , that is, X^c

In the complement operation, if the input pixel is white then the output pixel value will be black and vice versa. This is illustrated below.



Step 2 To calculate $Y_1 = X \oplus B$

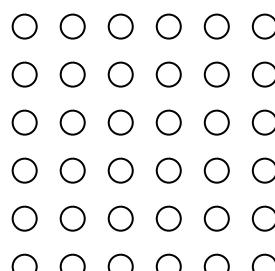
$Y_1 = X \oplus B$ indicates the dilation of the image X by the structuring element B . The result of the dilation operation is given below.



From the above figure, it is obvious that dilation leads to expansion of the input image.

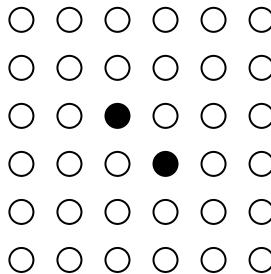
Step 3 To calculate $Y_2 = X^c \Theta B$

First, the complement of the input image is taken which is then eroded by the structuring element B . The result of the operation is shown below.



Step 4 To calculate $Y_2 \equiv X \Theta B$

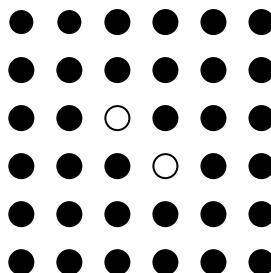
The input image X is eroded by the structuring element B . The result of the operation is given below.



From the above result, it is obvious that erosion operation is anti-expansive.

Step 5 To calculate $Y_4 = X^c \oplus B$

Now, the complement of the input image is taken and it is dilated by the structuring element B . The result is shown below.



Step 6 The relationship between Y_3 and Y_4

It is easy to see $Y_3 = Y_4^c$

10.11 MORPHOLOGICAL ALGORITHMS

The morphological algorithms discussed in this section include hit-or-miss transform, region filling, convex hull, thinning and thickening.

10.11.1 Hit-or-Miss Transform

The hit-or-miss transform is a transformation which is used for template matching. The transformation involves two template sets, B and $(W-B)$, which are disjoint. Template B is used to match the foreground image, while $(W-B)$ is used to match the background of the image. The hit-or-miss transformation is the intersection of the erosion of the foreground with B and the erosion of the background with $(W-B)$.

The hit-or-miss transform is defined as

$$HM(X, B) = (X \Theta B) \cap (X^c \Theta (W - B)) \quad (10.27)$$

The small window W is assumed to have at least one pixel, thicker than B . The input image X and the structuring element B are shown in Fig. 10.38.

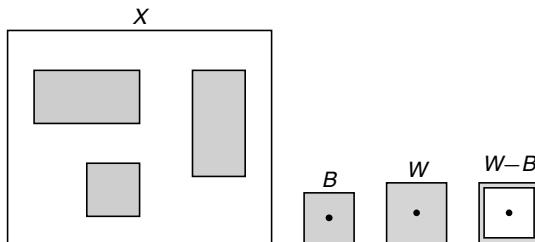
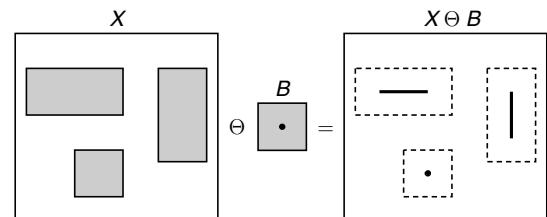
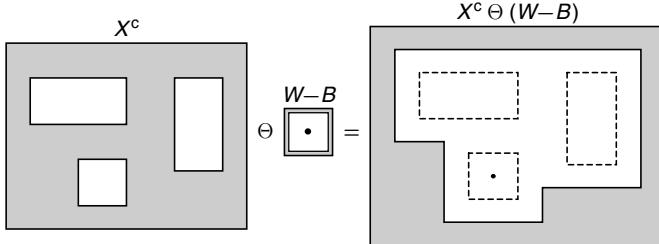
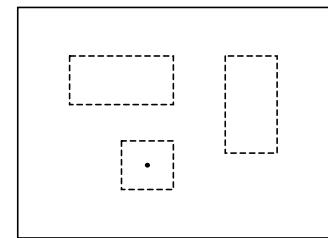
Fig. 10.38 Input image X , structuring element B , W , $W-B$ Fig. 10.39 Eroded image of X Fig. 10.40 Eroded image of X^c 

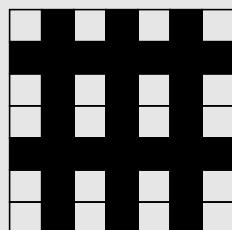
Fig. 10.41 Intersection result of above two results

Equation (10.27) describes the hit-or-miss transformation. First, we have to find the erosion of the input image X with the structuring element B as illustrated in Fig. 10.39.

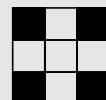
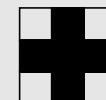
As per Eq. (10.27), find the complement of the input image X , and then erode it with the structuring element $(W-B)$. The resultant image is shown in Fig. 10.40.

Now, find the intersection of the images shown in Fig. 10.39 and Fig. 10.40 which give the hit-or-miss transformation of the input image X . The hit-or-miss transformation of the input image is shown in Fig. 10.41.

Example 10.4 The input image and structuring elements are shown below. Find the hit or miss transformation for the input image.



(a) Input image

(b) Structuring element B (c) Structuring element $W-B$

Solution

Step 1 From the definition of hit-or-miss transformation, we have to find the erosion of the input image with the structuring element B . The result of the erosion operation is shown below.

Step 2 Next, find the erosion of the complement of the input image with the structuring element $W-B$; we get the output image as shown below.

Step 3 From the result of steps 1 and 2, the hit-or-miss transformed input image is found from the intersection of the result of the image in steps 1 and 2. The resultant image is shown below.

A 5x5 grid of squares. A 3x3 block of squares in the center is filled with black, while the other squares are white.

10.11.2 Boundary Detection

Morphological operations are very effective in the detection of boundaries in a binary image X . The following boundary detectors are widely used:

$$\begin{aligned} Y &= X - (X \Theta B) \\ Y &= (X \oplus B) - X \quad \text{or} \\ Y &= (X \oplus B) - (X \Theta B) \end{aligned} \tag{10.28}$$

where Y is the boundary image, operator ' Θ ' denotes erosion and operator ' \oplus ' denotes dilation. ' $-$ ' denotes the set theoretical subtraction.

From Eq. (10.28), these are used as a boundary detector in binary image processing. This will be illustrated in Fig. 10.42, Fig. 10.44 and Fig. 10.45. Figure 10.42 shows the MATLAB program for the boundary detection operation.

The original, dilated and eroded image is shown in Fig. 10.43. From Fig. 10.44, it is obvious that Eq. (10.28) can be used to extract the boundary of the input image.

```

close all;
clear all;
clc;
a = imread('morph2.bmp');
b = [0 1 0;1 1 1;0 1 0];
a1 = imdilate(a, b);
a2 = imerode(a, b);
a3 = a-a2;
a4 = a1-a;
a5 = a1-a2;
imshow(a), title('Original image')
Fig., imshow(a1), title('Dilated image')
Fig., imshow(a2), title('Eroded image')
Fig., imshow(a3), title('First property ')
Fig., imshow(a4), title('Second property')
Fig., imshow(a5), title('Third Property')

```

Fig. 10.42 MATLAB code for the boundary detector

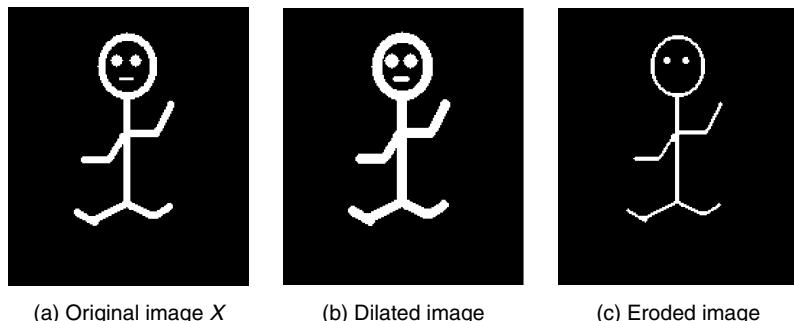


Fig. 10.43 Dilated and eroded image of X

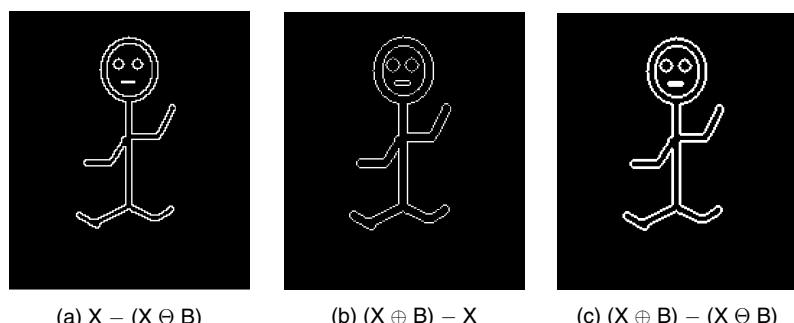


Fig. 10.44 Boundary detection of the input image X

10.11.3 Region Filling

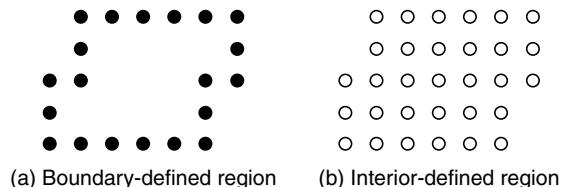
Region filling is the process of ‘colouring in’ a definite image area or region. Regions may be defined at the pixel or geometric level. At the pixel level, we describe a region either in terms of the bounding pixels that outline it, or as the totality of pixels that comprise it. In the first case, the region is called boundary-defined which is shown in Fig. 10.45 (a), and the collection of algorithms used for filling such regions, as shown in Fig. 10.45 (b), are collectively called boundary-fill algorithms. The other type of region is called an interior-defined region, and the accompanying algorithms are called flood-fill algorithms. At the geometric level, each region is defined or enclosed by such abstract contouring elements as connected lines and curves.

The region filling is mathematically represented as

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, \dots \quad (10.29)$$

In Eq. (10.29), B denotes the structuring element; A denotes a set containing a subset whose elements are 8 connected boundary points of a region. Here, k denotes the number of iterations.

Beginning with a point inside the boundary, the objective is to fill the entire region with 1s, by iteratively processing dilation. From Eq. (10.29), region filling is based on dilation, complementation and intersections. There are two ways to terminate the algorithm—if the region is filled then stop the iterations or fix the number of iterations to fill the region.



(a) Boundary-defined region (b) Interior-defined region

Fig. 10.45 Region of binary image

Example 10.5 The input image and structuring element are shown in Fig. 10.46. Perform the region filling operation.

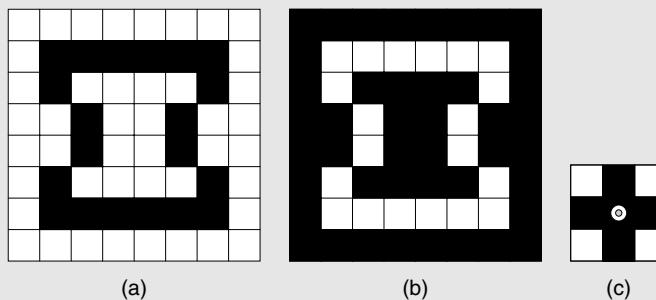
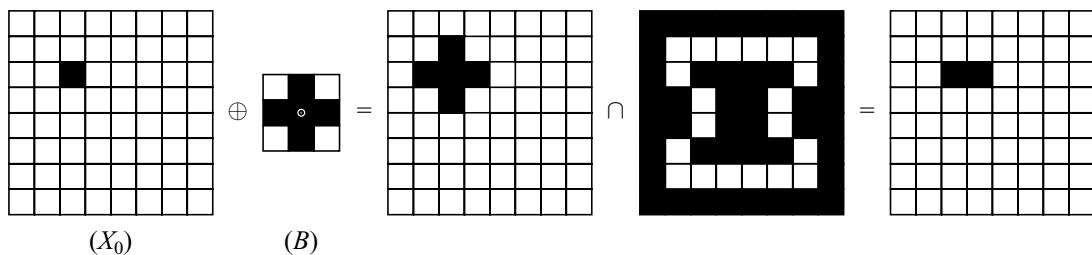


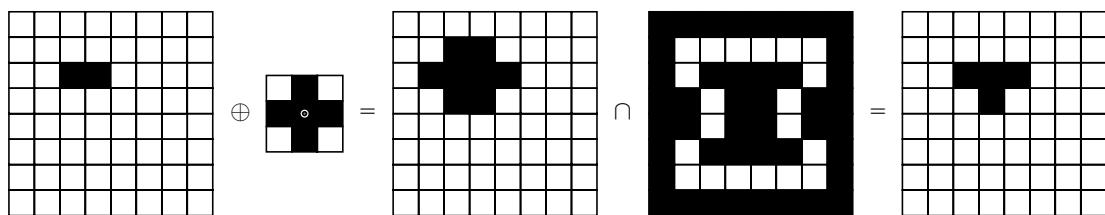
Fig. 10.46 (a) Input image A (b) Complement of input image (c) Structuring element B

Solution

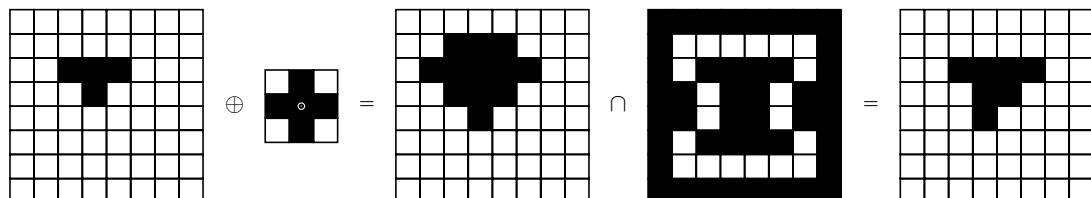
Step 1 Initially, take the X_0 as shown below. Now, perform the dilation of X_0 with the structuring element B . The resultant image is then intersected with the complement of the input image. This completes the first iteration.



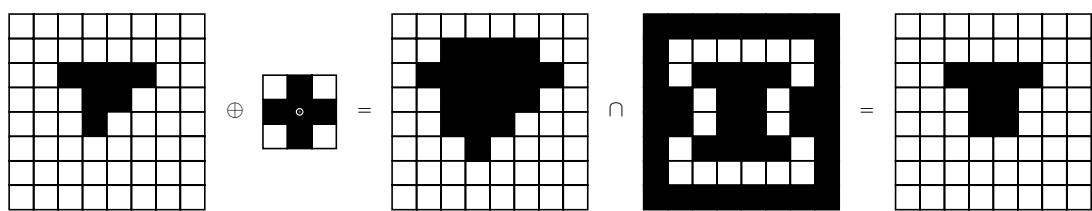
Step 2 Now, the input to the second step is the result of the first iteration. The process performed in Step 1 is repeated in Step 2.



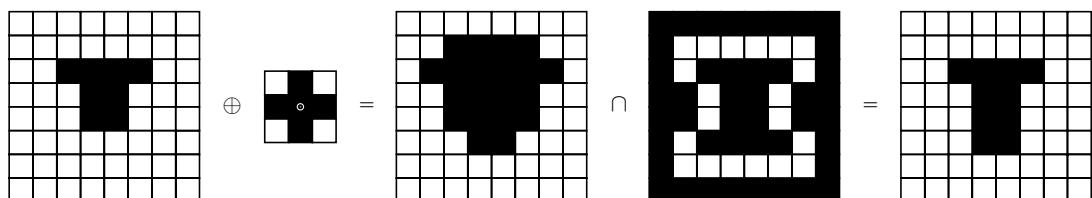
Step 3 The same process is repeated again but the input image to the third iteration is the output image of Step 2.



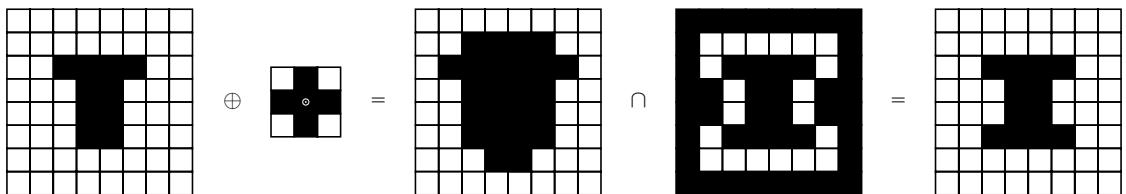
Step 4 The steps followed in steps 2 and 3 are repeated in Step 4. The resultant image is shown below.



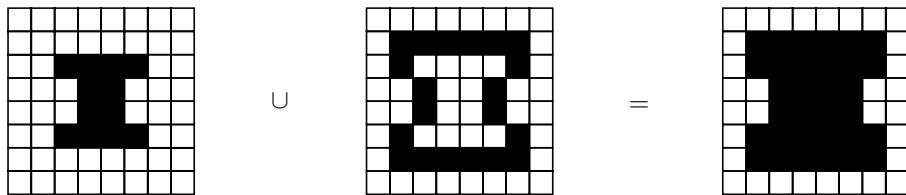
Step 5 The input to Step 5 is the output image of Step 4. The process done in Step 4 is repeated in Step 5.



Step 6 The input to Step 6 is the output image of Step 5. The process done in Step 5 is repeated in Step 6.



Step 7 Now, we perform the union of the result obtained in Step 7 with the original input image to get the region-filled image which is shown below.



10.11.4 Convex Hull

X is said to be convex if the straight-line segment joining any two points in X lies entirely within X .

$$Y_k^i = \left(HM(Y_{k-1}, B^i) \cup X \right) i = 1, 2, 3, 4 \quad \text{and} \quad k = 1, 2, 3, \dots$$

with

$$Y_0^i = X, \quad \text{and} \quad \text{let } D^i = Y_{\text{conv}}^i$$

Here, 'conv' means convergence, and the convex hull of X is $C(X) = \bigcup_{i=1}^4 D^i$.

The structuring elements for the convex hull are shown in Fig. 10.47, where 'x' denotes *don't care*. The convex hull of a binary image can be visualised quite easily by imagining stretching an elastic band around the image shape. The elastic band will follow the convex contours of the shape, but will bridge the concave contours. The resulting shape will have no concavities and will contain the original image shape.

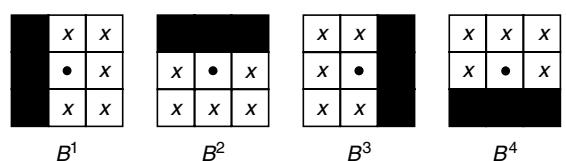
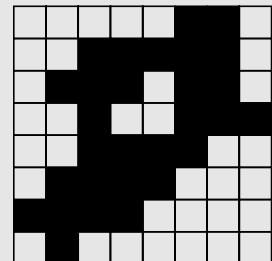


Fig. 10.47 Different structuring elements of convex hull

Example 10.6 The input image X and the structuring element are shown below. Find the convex hull in the input image.



$$X = Y_0^1$$

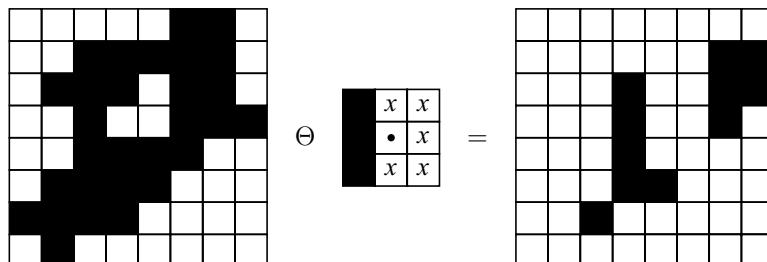
Solution The step-by-step approach to the determination of the convex hull of the input image is given below.

Step 1 The value of Y_1^1 is determined using $Y_1^1 = HM(Y_0^1, B^1) \cup X$.

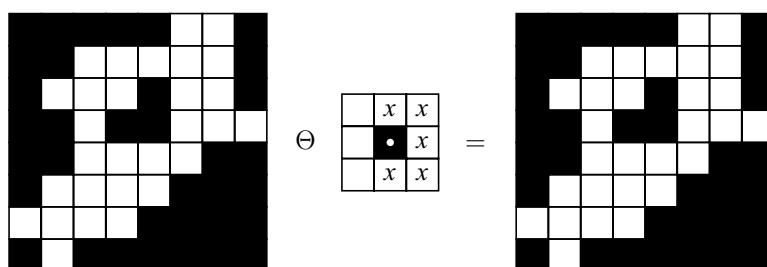
Step 2 To find $HM(Y_0^1, B^1)$

$$HM(Y_0^1, B^1) = (Y_0^1 \Theta B^1) \cap ((Y_0^1)^c \Theta (W - B^1))$$

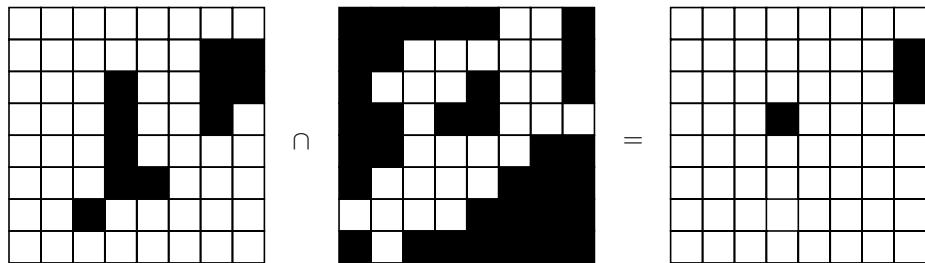
From the above definition, we have to find $Y_0^1 \Theta B^1$, the result of this operation is shown below.



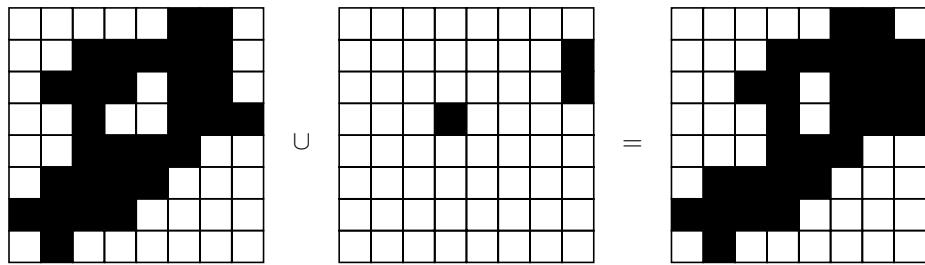
The next step is to find $(Y_0^1)^c \Theta (W - B^1)$. The result of $(Y_0^1)^c \Theta (W - B^1)$ is shown below.



Then we find the intersection of above two results which is illustrated below.



Step 3 The union of the input image with the result obtained in Step 2 will give the convex hull of the input image which is illustrated below.



10.11.5 Thinning

Thinning a binary image down to a unit-width skeleton is useful not only to reduce the amount of pixels, but also to simplify the computational procedures, required for shape description. The thinning operation is based on the hit-or-miss transformation as

$$X \otimes B = X - HM(X, B)$$

or

$$X \otimes B = X \cap HM(X, B)^c \quad (10.30)$$

The different possibilities of structuring elements are shown in Fig. 10.48.

The thinning process is split into two types. (i) It removes boundary pixels of a connected component that are neither essential for preserving the connectivity of image nor represent any significant geometrical features of an image. The process converges when the connected skeleton does not change or vanishes even

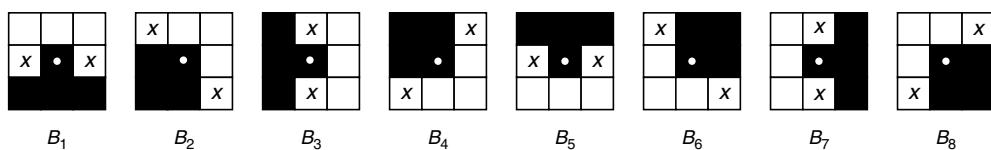
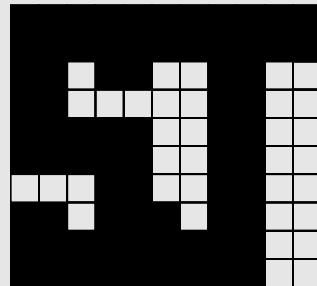


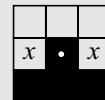
Fig. 10.48 Structuring elements of thinning operation

if the iteration process continues. (ii) It encodes distance information for every pixel of the pattern by a label representing the distance of the pixel to the boundary. The set of pixels with local minimum distance labels is used to derive the resulting skeleton.

Example 10.7 Apply the thinning process to the image using the structuring element shown below.



Input image

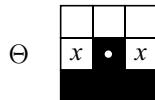
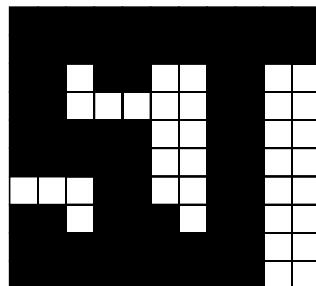


Structuring element

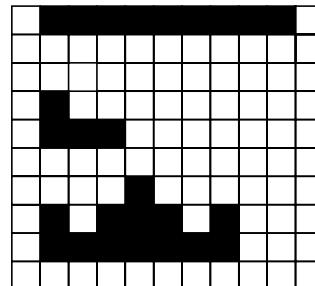
Solution The step-by-step approach of the thinning process is given below.

Step 1 To perform the eroded operation of input image with structuring element

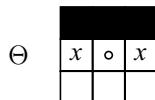
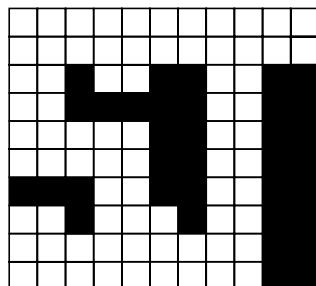
First, we find the eroded input image with the structuring element. The resultant image is shown below.



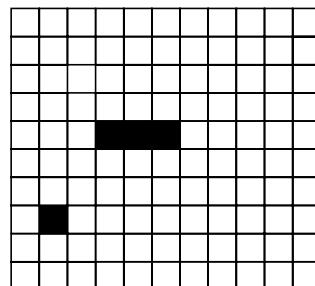
Θ



Step 2 To perform the eroded operation of the complement input image with the complement structuring element. The resultant image is illustrated below.

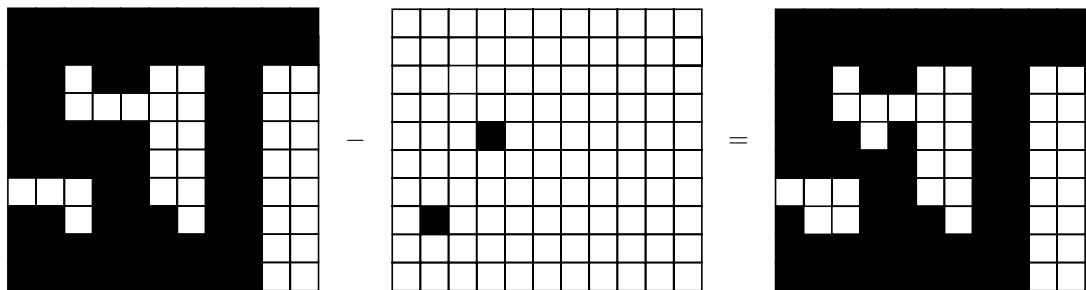


Θ



Step 3 To find the hit-or-miss transformation of the input image

The hit-or-miss transformation is the intersection of the above two results. This resultant image is subtracted from the original input image; and we get the thinned original image. This is illustrated below.



Note Changing the structuring element will yield a different extent of the thinning operation.

10.11.6 Thickening

Thickening is the morphological operation which is used to grow selected regions of foreground pixels in binary images. This process is somewhat like dilation or closing. The thickening operation is defined by

$$\bar{X} \cdot B = X \cup HM(X, B) \quad (10.31)$$

In Eq. (10.31), X is the input image and B is the structuring element. The thickened image consists of the original image plus any additional foreground pixels switched on by hit-or-miss transform. The thickening process is the dual of thinning, that is, thinning the foreground is equivalent to thickening the background. This process is normally applied repeatedly until it causes no further changes in the image. The different structuring elements that can be used in the thickening process are shown in Fig. 10.49.

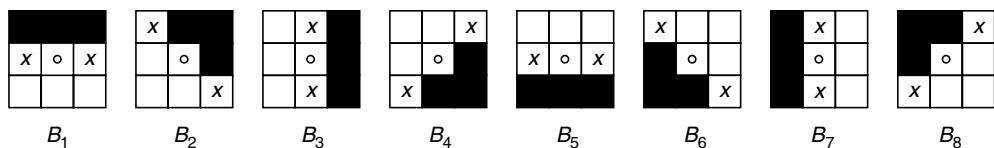
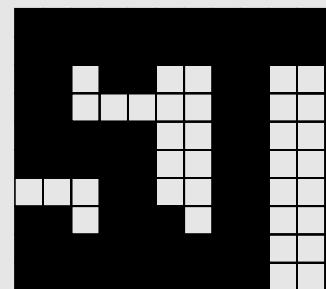


Fig. 10.49 Different structuring elements of thickening process

The thickening operation is calculated by translating the origin of the structuring element to each possible pixel position in the image, and at each position comparing it with the underlying image pixels. If the black and white pixels in the structuring element exactly match the black and white pixels in the image, then the image pixel underneath the origin of the structuring element is set to black. Otherwise, it is left unchanged i.e., white. The applications of the thickening process are determining the convex hull and finding the skeleton by zone of influence.

Example 10.8 Find the thickening process of the given image. Use B_1 as a structuring element which is shown in Fig. 10.49.



Solution

Step 1 First, find the hit-or-miss transformation of the given image with the structuring element B_1 .

$$\begin{array}{c}
 \text{Image} \\
 \Theta \\
 \text{Image} = \\
 \text{Image}
 \end{array}$$

Step 2 Then find the erosion of the complement of the input image with the complement of the structuring element.

$$\begin{array}{c}
 \text{Input Image} \\
 \Theta \\
 \text{Output Image}
 \end{array}$$

Step 3 Now, find the intersection of the output of steps 1 and 2. This result is combined (union) with the original image which is shown below.

10.12 DISTANCE TRANSFORM

The distance transform is an operator normally applied only to binary images. The distance transform assigns to each feature pixel of a binary image a value equal to its distance to the nearest non-feature pixels. It can be used to extract information about the shape and the position of the foreground pixels relative to each other. It has been applied to many fields of research such as medical imaging, pattern recognition and surface reconstruction. The distance transforms play a central role in the comparison of binary images, particularly for images resulting from local feature-detection techniques such as edge or corner detection.

There are different types of distance transforms depending on how the distance is measured. The distance between pixels can be measured by Euclidean distance, city block, chessboard and Gaussian distance transform. The distance metrics used are described as follows.

10.12.1 Euclidean Distance

In Euclidean distance transform, the value at a pixel is linearly proportional to the Euclidean distance between that pixel and the object pixel closest to it. Since the method uses the value at a single object pixel to determine the value at the pixel of interest, the process is sensitive to noise.

The straight-line distance between two pixels is given by

$$D_E[(i, j), (k, l)] = \sqrt{(i-k)^2 + (j-l)^2} \quad (10.32)$$

The input image and its Euclidean distance transform are shown in Fig. 10.50.

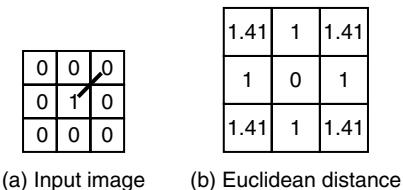


Fig. 10.50 Euclidean distance transform

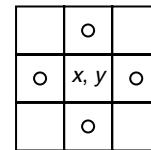


Fig. 10.51 Four connected neighbours

10.12.2 City-Block Distance

The city-block distance of two pixels $D_4[(i, j), (k, l)] = |i - k| + |j - l|$. This metric measures the path between the pixels based on a four-connected neighbourhood, as shown in Fig. 10.51. For a pixel 'p' with the coordinates (x, y) , the set of pixels given by,

$$N_4(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\} \quad (10.33)$$

Equation (10.33) is called its four neighbours, as shown in Fig. 10.51. The input and its city block distance are shown in Fig. 10.52.

10.12.3 Chessboard Distance

The chessboard distance of two pixels is given by

$$D_8[(i, j), (k, l)] = \max[|i - k|, |j - l|] \quad (10.34)$$

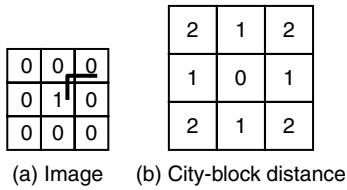


Fig. 10.52 City-block distance transform

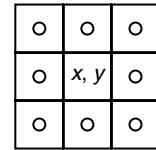


Fig. 10.53 Eight connected neighbours

The chessboard distance metric measures the path between the pixels based on an eight-connected neighbourhood, as shown in Fig. 10.54. The set of eight neighbour pixels is defined as

$$N_8(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\} \cup \{(x+1, y+1), (x-1, y+1), (x+1, y-1), (x-1, y-1)\} \quad (10.35)$$

Equation (10.35) is called the eight connected neighbours as shown in Fig. 10.53. The chessboard distance of the image and the original image is shown in Fig. 10.54.

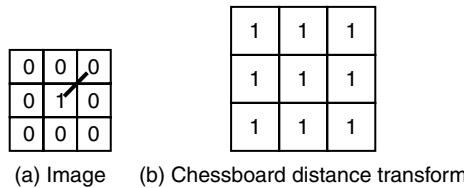


Fig. 10.54 Chessboard distance transform

10.12.4 Quasi-Euclidean Distance Transform

The quasi-Euclidean metric measures the total Euclidean distance along a set of horizontal, vertical and diagonal line segments. The quasi-Euclidean distance of two pixels is given by

$$D_{QE}[(i, j), (k, l)] = \begin{cases} |i - k| + (\sqrt{2} - 1) \times |j - l| & \text{if } |i - k| > |j - l| \\ (\sqrt{2} - 1) \times |i - k| + |j - l| & \text{otherwise} \end{cases} \quad (10.36)$$

The quasi-Euclidean distance transform of the image is shown in Fig. 10.55.

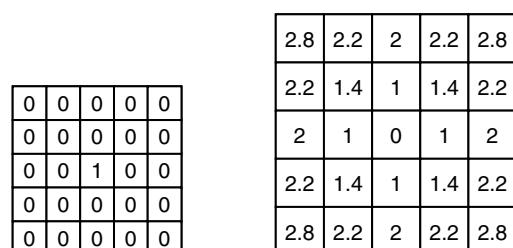


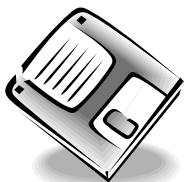
Fig. 10.55 Quasi-Euclidean distance transform

10.13 SALIENT FEATURES OF MORPHOLOGICAL APPROACH

The salient features of morphological approach are summarised below:

1. Morphological operations provide the systematic alteration of the geometric content of an image while maintaining the stability of the important geometric characteristics.
2. There exists a well-developed morphological algebra that can be employed for representation and optimisation.
3. It is possible to express digital algorithms in terms of a very small class of primitive morphological operations.
4. There exist rigorous representation theorems by means of which one can obtain the expression of morphological filters in terms of the primitive morphological operations.
5. The linear transformations of functions and morphological operations are characterised by their non-invertibility.
6. They remove information of greater and greater extent as the size of the structural element increases.
7. Image processing through iterative morphological transformations can, therefore, be conceived as a process of selective information removal where irrelevant details are irrecoverably destroyed, thereby enhancing the contrast of essential function features.

Summary



- Morphological image processing is based on the idea of probing an image with a small shape or template known as a structuring element. It is possible to use structuring elements of different shapes to perform a specific task.
- The basic morphological operations are dilation and erosion.
- In the dilation operation, the image is probed with the structuring element by successively placing the origin of the structuring element to all possible pixel locations; the output is 1 if the structuring element and the image have a non-zero intersection and 0 otherwise.
- In the erosion operation, the image is probed with the structuring element by successively placing the origin of the structuring element to all possible pixel locations; the output is 1 if the structuring element is completely contained in the image and 0 otherwise.
- The opening operation is defined as erosion followed by dilation. The opening operation has the effect of eliminating small and thin objects, smoothing the boundaries of large objects without changing the general appearance.
- The closing operation is defined as dilation followed by erosion. The closing operation has the effect of filling small and thin holes in an object, connecting nearby objects and generally smoothing the boundaries of large objects without changing the general appearance.
- Morphological operations such as opening and closing can be regarded as morphological filters that remove undesired features from an image.
- The hit-or-miss operation probes the inside and outside of objects at the same time, using two separate structuring elements. A pixel belonging to an object is preserved by the hit-or-miss operation if the first structuring element translated to that pixel fits the object, and the second translation element misses the object. The hit-or-miss operation is useful for the detection of specific shapes.
- The thinning operation accomplishes erosion without breaking objects. Thinning can be accomplished as a two-step approach, with the first step being erosion that marks all candidates for removal without actually removing them; and in the second pass, candidates that do not destroy the connectivity are removed.

Review Questions

1. What are the effects of the dilation process?

The effects of the dilation process are summarised below:

- (i) Dilation expands the input image.
- (ii) This process affects both the inside and outside borders of the input image.
- (iii) Dilation fills the holes in the image.
- (iv) The dilation process smoothes out the image contour.
- (v) The dilation process depends on the choice of the structuring element.

2. What are the major effects in the erosion process?

The effects of the erosion process are summarised below

- (i) The erosion process shrinks the given input image.
- (ii) Pixels and holes inside the image are normally removed by the erosion process.
- (iii) This process removes small objects like noise.

3. What is Beucher gradient?

Dilation and erosion can be used to extract edge information from images. The Beucher gradient is defined as

$$\rho = (X \oplus B) - (X \ominus B),$$

where X is the input image and B is the structuring element. This Beucher gradient is also called the gradient operator.

4. Briefly explain skeletons.

Skeletons are the minimal representation of objects in an image while retaining the Euler number of the image. The Euler number is the number of objects in an image minus the number of holes in those objects. The skeletons of objects in an image can be found by successive thinning until stability is reached.

5. What is geodesic dilation?

In geodesic dilation, the result after dilating the image X is masked using a mask image g . This can be defined as $X \oplus_g B = (X \oplus B) \cap g$.

6. What is geodesic erosion?

Geodesic erosion, which is the result after erosion, is masked with a mask image g . It is defined as $X \ominus_g B = (X \ominus B) \cup g$.

7. Mention the properties of opening and closing.

- (i) Opening and closing are dual operators.
- (ii) Closing is an extensive transform.
- (iii) Opening is an anti-extensive transform.
- (iv) Opening and closing keep the ordering relation between two images.
- (v) Opening and closing are idempotent transforms.

8. What is the use of the closing operation?

The closing operation is used to smoothen the contour of the input image. It can be used to connect small gaps in the image and also remove small holes in the input image.

9. What is a hit-or-miss transformation?

The hit-or-miss transformation is the morphological operator for finding local patterns of pixels. Here, local refers to the size of the structuring element. A hit-or-miss transform is a variant of template matching that finds collection of pixels with certain shape properties.

10. Give a few applications of morphological operations in the field of image processing.

The applications of morphological operations include the following:

- (i) The morphological operators can be used to remove noise in the image which is widely used in image preprocessing.
- (ii) The morphological operators can be used to enhance the object structure which includes region thinning, thickening, convex, hull etc.
- (iii) Morphological operators can be used to segment objects from the background.
- (iv) Morphological operators are effective in the quantitative description of objects.

Problems

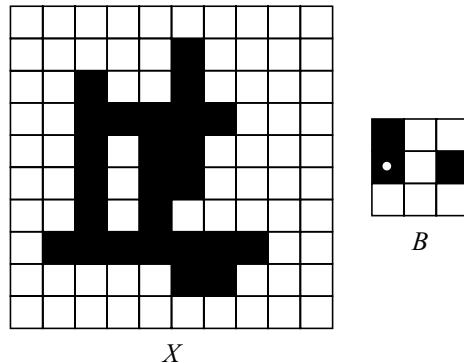
- 10.1 A binary array that represents a portion of a black-and-white image is given below. Perform the following operations on this piece of image. Assume that all the pixels that surround this segment contain a black background.

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

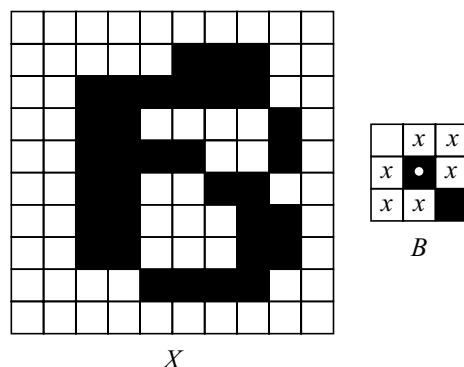
Piece of image

- (i) Dilation with the structuring element $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$
- (ii) Erosion with the structuring element given in (i)
- (iii) Opening with the structuring element given in (i) and (ii)
- (iv) Closing with the structuring element given in (i) and (ii)

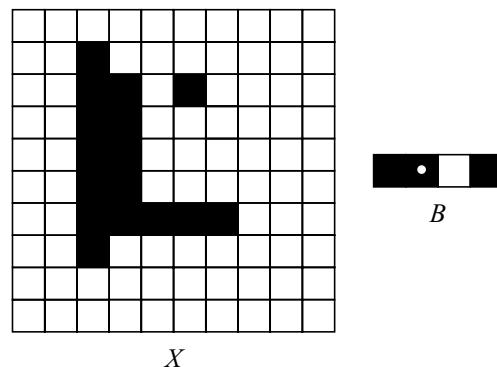
- 10.2 Given below is a binary image, where X is the set of black pixels. X is eroded by the structure element B . How many black pixels are left in the eroded image?



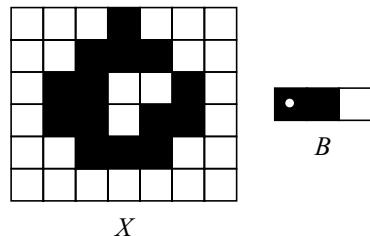
- 10.3 The input binary image X is shown below. A hit-or-miss transformation is performed with the following hit-or-miss structure element B . How many black pixels are left in the resulting image?



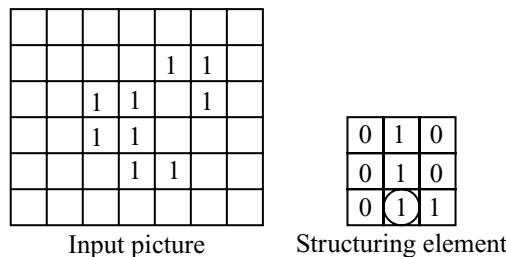
- 10.4 The input binary image X is shown below. The image X is opened with the structuring element B . Show the resultant image.



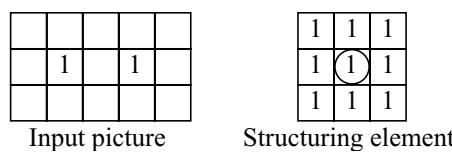
- 10.5 X is the image given on the set of black pixels, below. Perform the operation $(X \circ B) \bullet B \circ B$ with structuring element B a shown below. Illustrate the resultant image.



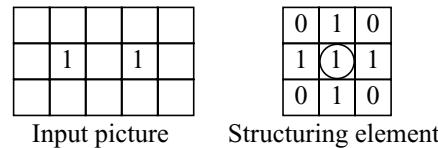
- 10.6 The input picture and structuring element are shown below. Perform the erosion and dilation of the input picture.



- 10.7 The input picture and structuring element are shown below. Perform the opening and closing of the input picture.



- 10.8 The input picture and structuring element are shown below. Perform the opening and closing of the input picture.



- 10.9 Give a morphological algorithm for converting an 8-connected binary boundary to an m -connected boundary. You may assume that the boundary is fully connected and that it is one pixel thick.

- 10.10 How can the given object be cleared up by using morphological operation? (The outlines of the zeros in the image should be closed.)

```

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

```

References

Books

1. Jean Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982
2. Rafael C Gonzalez and Richard E Woods, *Digital Image Processing*, Pearson Education, Second Edition, 2002
3. Alan C Bovik, *Handbook of Image and Video Processing*, Elsevier, Second Edition, 2005
4. Maria Petrou and Pedro Garcia Sevilla, *Image Processing Dealing with Texture*, John Wiley & Sons, 2006
5. E R Dougherty, *Morphological Image Processing*, Marcel Dekker, New York, 1993

Journals

1. L Vincent, *Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms*, IEEE Transactions on Image Processing, vol. 2, no. 2 , pp. 176–201, April 1993
2. R M Haralick, S R Stenberg and X Zhuang, *Image Analysis Using Mathematical Morphology*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no.4, pp.532–550, 1987
3. Stephan S Wilson, *Theory of Matrix Morphology*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 6, pp. 636–652, 1992

11



Learning Objectives

The subject of colour-image processing requires an understanding of the physics of light as well as the physiology of colour perception. Humans use colour information to distinguish objects, materials, food, places and time of the day. Colour images surround us in print, television, computer displays, photographs and movies. Colour images in general occupy more space when compared to grayscale and binary images. After reading this chapter, the reader should have basic understanding of the following topics:

basic concepts related to colour formation

human perception of colour information

different types of colour models and CIE chromaticity diagram

Colour quantisation

Colour histogram equalisation

Colour image filtering and pseudo-colouring

Colour-Image Processing

11.1 INTRODUCTION

Colour is the perceptual sensation of light in the visible range incident upon the retina. Colour is the most visually striking feature of any image and it has a significant bearing on the scenic beauty of an image. To understand colour, it is necessary to understand the nature of light. Light exhibits a dual nature. It behaves both as a particle and as a wave. When photons hit the retina, they give rise to electric impulses, which on reaching the brain, are translated into colour. Different wavelengths of light are perceived as different colours. However, not every wavelength can be perceived by the human eye. The wavelengths between 380 nm and 780 nm form the visible spectrum.

11.2 LIGHT AND COLOUR

The frequency of light determines the colour. The amount of light determines the intensity. The famous Einstein relation is given by

$$E = h\nu = \frac{hc}{\lambda} \quad (11.1)$$

As stated earlier, the visible spectrum is approximately between 400 nm to 700 nm. The human visual system perceives electromagnetic energy having wavelengths in the range 400–700 nm as visible light. Lightness of brightness refers to the amount of light a certain colour reflects or transmits. Light that has a dominant frequency or set of frequencies is called *chromatic*. *Achromatic light* has no colour and it contributes only to quantity or intensity. The intensity is determined by the energy, whereas brightness is determined by the perception of the colour; hence it is psychological. Colour depends primarily on the reflectance properties of an object.

11.3 COLOUR FORMATION

The colours a human being perceives in day-to-day life arise from a very diverse physiochemical process. Some objects are light sources, while others merely reflect the incident light. Basically, there are three colour-formation processes: (i) additive process, (ii) subtractive process, and (iii) pigmentation.

11.3.1 Additive Colour Formation

In additive colour formation, the spectral distributions corresponding to two or more light rays get added. The resulting colour is the sum of the number of photons in the same range present in the component colours.

The additive colour-formation principle is employed in TV monitors.

11.3.2 Subtractive Colour Formation

Subtractive colour formation occurs when the light is passed or transmitted through a light filter. A light filter partly absorbs part of the light that reaches it and transmits the rest. For example, a green filter lets through the radiation in the green part of the spectrum, while radiation with other wavelengths is blocked. Several filters can be used in series, the resulting colour being made up of those wavelengths that can go through all of them.

Subtractive colour formation occurs when colour slides are projected onto a screen.

11.3.3 Colour Formation by Pigmentation

A pigment consists of coloured particles in suspension in a liquid. These particles can absorb or reflect the light that reaches them. When a light ray reaches a surface covered with a pigment, it is scattered by the particles, with successive and simultaneous events of reflection, transmission and absorption. These events determine the nature (colour) of the light reflected by the surface. Colour formation through pigmentation allows one to see colours in a painting.

11.4 HUMAN PERCEPTION OF COLOUR

The human visual system can detect the range of light spectrum from about 400 nm to about 700 nm. Vision and hearing are the two most important means by which humans perceive the outside world. It is estimated that 75% of the information received by a human is visual. The use of visual information is referred to as *perception, sight or understanding*. An understanding of the perceptual processing capabilities of humans provided motivation for developing similar paradigms for computers. The complex aspects of visual perception are best introduced by a brief description of the anatomy of the human visual system. The Human Visual System (HVS) is an information-processing system, receiving spatially sampled images from the cones and rods in the eye, and deducing the nature of the objects it observes by processing the image data. The photoreceptors responsible for colour vision are the cones. There are three types of cones in the retina. They are

- (i) L-receptors which are sensitive to light of long wavelengths
- (ii) M-receptors which are sensitive to light of middle wavelengths
- (iii) S-receptors which are sensitive to light of short wavelengths

It is customary to denote the RGB sensors with the Greek letters rho (red), gamma (green) and beta (blue). The sensitivity curves of rho, gamma and beta are shown in Fig. 11.1.

The sensitivity curves of the rho, gamma and beta sensors in the human eye determine the intensity of the colours that one perceives for each of the wavelengths in the visual spectrum. Selective sensing of different light wavelengths allows the visual system to create the perception of colour. Colour vision also depends on the sensitivity of light or the illumination of objects. When the illumination is very low, cones are not activated. However, rods are activated even when there is very little illumination.

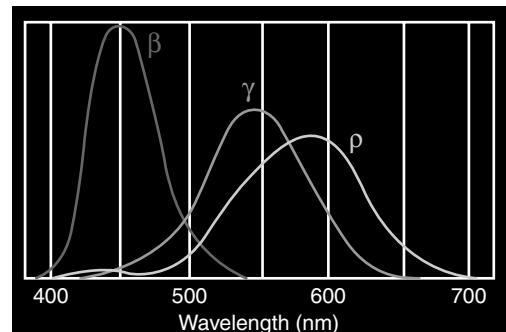


Fig. 11.1 Sensitivity of rho, gamma and beta curves

11.5 COLOUR MODEL

Colour models provide a standard way to specify a particular colour by defining a 3D coordinate system, and a subspace that contains all constructible colours within a particular model. Any colour can be specified using a model. Each colour model is oriented towards a specific software or image-processing application.

11.5.1 RGB Colour Model

In an RGB colour model, the three primary colours red, green and blue form the axis of the cube which is illustrated in Fig. 11.2. Each point in the cube represents a specific colour. This model is good for setting the electron gun for a CRT.

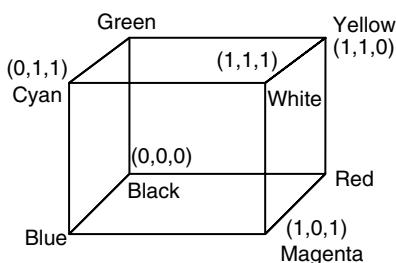


Fig. 11.2 RGB colour cube

RGB is an additive colour model. From Fig. 11.3, it is obvious that

$$\text{Magenta} = \text{Red} + \text{Blue} \quad (11.2)$$

$$\text{Yellow} = \text{Red} + \text{Green} \quad (11.3)$$

$$\text{Cyan} = \text{Blue} + \text{Green} \quad (11.4)$$

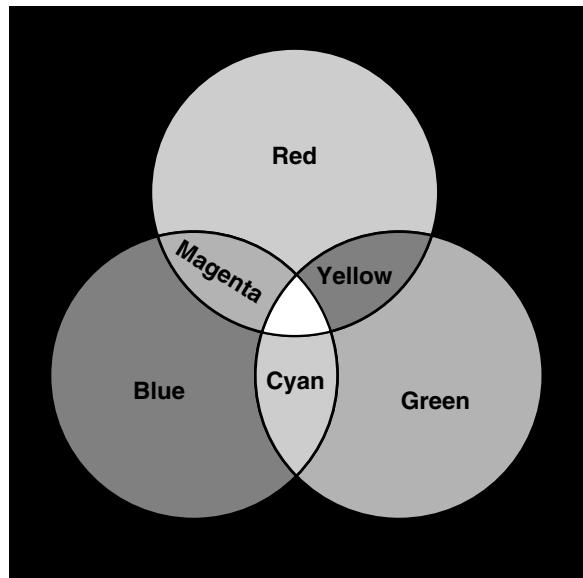


Fig. 11.3 RGB colour model

Limitations of the RGB Model The limitations of the RGB model are summarised below:

- (i) The RGB colour coordinates are device dependent. This implies that the RGB model will not in general reproduce the same colour from one display to another.
- (ii) The RGB model is not perceptually uniform. The meaning is that one unit of coordinate distance does not correspond to the same perceived colour difference in all regions of the colour space.
- (iii) It is difficult to relate this model to colour appearance because its basis is to device signals and not display luminance values.

MATLAB Example 1 (Extraction of colour components) Read an RGB image and extract the three colour components, red, green and blue.

Solution The MATLAB code which extracts the three components, red, green and blue, are shown in Fig. 11.4 and the corresponding output is shown in Fig. 11.5.

11.5.2 CMY Colour Model

Printers produce an image by reflective light, which is basically a subtractive process. Printers commonly employ the CMY model. The CMY model cube is illustrated in Fig. 11.6.

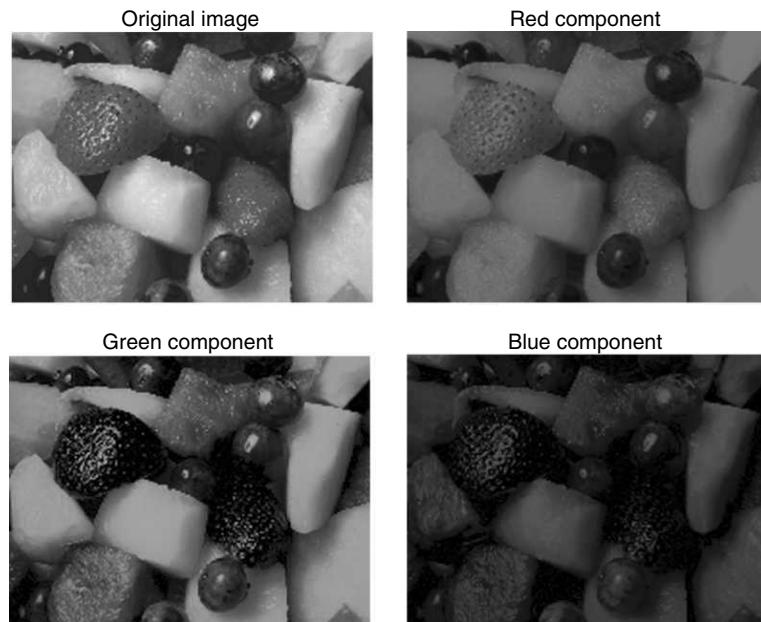
The conversion between RGB and CMY is done as follows

$$\begin{aligned} C &= 1 - R \\ M &= 1 - G \\ Y &= 1 - B \end{aligned} \quad (11.5)$$

```

RGB=imread('mixedfruit.bmp');
R=RGB;
G=RGB;
B=RGB;
R(:,:,2)=0;
R(:,:,3)=0;
G(:,:,1)=0;
G(:,:,3)=0;
B(:,:,1)=0;
B(:,:,2)=0;
subplot(2, 2, 1), imshow(RGB), title('original image')
subplot(2, 2, 2), imshow(R), title('Red Component')
subplot(2, 2, 3), imshow(G), title('Green Component')
subplot(2, 2, 4), imshow(B), title('Blue Component')

```

Fig. 11.4 *MATLAB code to extract red, green and blue components***Fig. 11.5** *Extraction of colour components*

CMY is a subtractive colour model. From Fig. 11.7, it is obvious that

$$\begin{aligned}
 \text{Magenta} &= \text{White} - \text{Green} \\
 \text{Cyan} &= \text{White} - \text{Red} \\
 \text{Yellow} &= \text{White} - \text{Blue}
 \end{aligned}$$

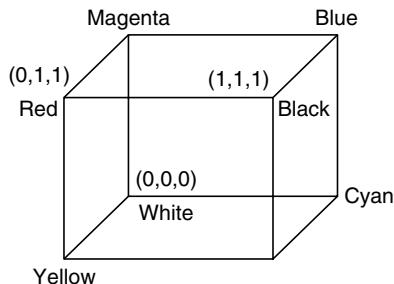


Fig. 11.6 CMY colour cube

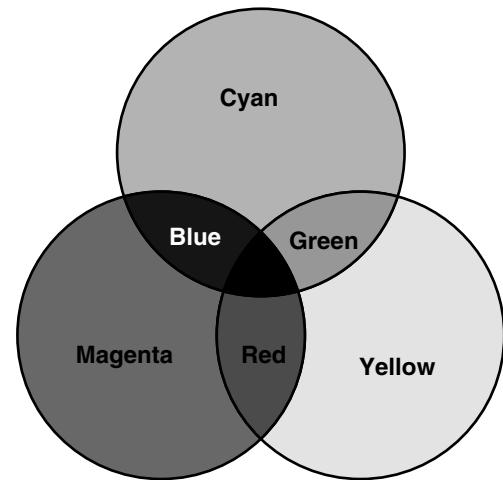


Fig. 11.7 CMY colour model

11.5.3 HSI Colour Model

HSI stands for Hue, Saturation and Intensity. *Hue* represents dominant colour as perceived by an observer. It is an attribute associated with the dominant wavelength. *Saturation* refers to the relative purity or the amount of white light mixed with a hue. *Intensity* reflects the brightness. HSI decouples the intensity information from the colour, while hue and saturation correspond to human perception, thus making this representation very useful for developing image-processing algorithms. HIS colour space is a popular colour space because it is based on human colour perception.

The conversion from RGB space to HSI space is given below.

$$I = \frac{1}{3}(R + G + B) \quad (11.6)$$

$$S = 1 - \frac{3}{(R + G + B)}[\min(R, G, B)] \quad (11.7)$$

and

$$H = \cos^{-1} \left\{ \frac{0.5[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}} \right\} \quad (11.8)$$

The HIS colour model can be considered as a cylinder, where the coordinates r , θ , and z are saturation, hue and intensity respectively. The cylindrical representation of the HIS colour model is illustrated in Fig. 11.8.

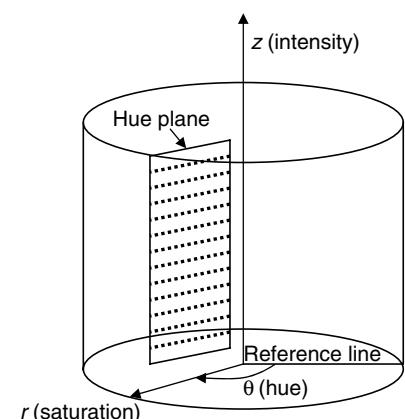


Fig. 11.8 Cylindrical representation of HIS colour model

11.5.4 YIQ Colour Model

The YIQ colour model is defined by the National Television System Committee (NTSC). In this model, Y represents the luminance; and I and Q describe the chrominance. Conversion between RGB and YIQ is given below.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.956 & 0.621 \\ 1.0 & -0.272 & -0.649 \\ 1.0 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

11.5.5 YCbCr Colour Coordinate

The YCbCr colour coordinate was developed as part of the ITU-R BT.601 during the establishment of a worldwide digital video component standard. The YCbCr signals are scaled and offset versions of the YIQ format. Y is defined to have a nominal range of 16 to 235; Cb and Cr are defined to have a range of 16 to 240, with zero signal corresponding to level 128. There are several YCbCr formats such as 4:4:4, 4:2:2 and 4:1:1. The sampling format implies that the sampling rates of Cb and Cr are one-half of that of Y .

11.6 THE CHROMATICITY DIAGRAM

Consider a three-dimensional colour space associated with some physical system. After fixing a basis, this space can be identified with the Euclidean space R^3 . Let the primary colours be denoted by P_1 , P_2 and P_3 and the coordinates of a colour C be denoted by C_1 , C_2 and C_3 . Let the chrominance plane be the plane M having the equation $x + y + z = 1$. The triangle formed by the intersection of this plane with the axes of colour space is called the Maxwell triangle. The Maxwell triangle is shown in Fig. 11.9. The Maxwell plane intersects the plane of zero luminance in a line of space, which is termed the *line of zero luminance*. Each chrominance line is entirely determined by its intersection point with the Maxwell plane. This gives a parameterisation of the chromaticity by associating to each chroma line its intersection with the Maxwell plane.

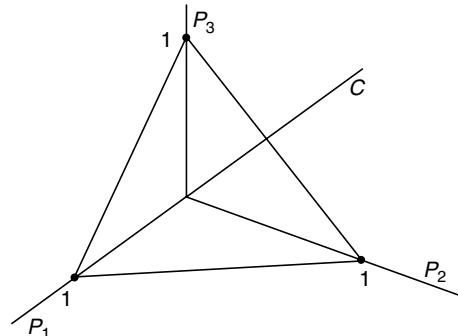


Fig. 11.9 The Maxwell triangle

11.6.1 CIE Chromaticity Diagram

The CIE chromaticity diagram is an international standard for primary colours established in 1931. It allows all other colours to be defined as weighted sums of the three primary colours.

In the CIE system, the intensities of red, green and blue are transformed into tristimulus values which are represented by X , Y and Z . The relative amounts of the three primary colours of light required to produce a colour of a given wavelength are called *tristimulus values*. These values represent the relative quantities of the primary colours. The coordinate axes of the CIE chromaticity diagram are derived from the tristimulus values as

$$x = X/(X + Y + Z) = \text{red}/(\text{red} + \text{green} + \text{blue}) \quad (11.9)$$

$$y = Y/(X + Y + Z) = \text{green}/(\text{red} + \text{green} + \text{blue}) \quad (11.10)$$

and

$$z = Z/(X + Y + Z) = \text{blue}/(\text{red} + \text{green} + \text{blue}) \quad (11.11)$$

The coordinates x , y and z are called *chromaticity coordinates* and always add up to 1. As a result, z can always be expressed in terms of x and y , which means that only x and y are required to specify any colour, and the diagram can be two-dimensional. The chromaticity diagram can be used to compare the gamuts of various possible output devices like printers and monitors.

The formulas to convert the tristimulus values X , Y , Z to RGB values and back are given below:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.9107 & -0.5326 & -0.2883 \\ -0.9843 & 1.9984 & -0.0283 \\ 0.0583 & -0.1185 & 0.8986 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (11.12)$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.6067 & 0.1736 & 0.2001 \\ 0.2988 & 0.5868 & 0.1143 \\ 0.0000 & 0.0661 & 1.1149 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (11.13)$$

The CIE chromaticity diagram as shown in Fig. 11.10 is useful in defining the colour gamut, the range of colours that can be produced on a device.

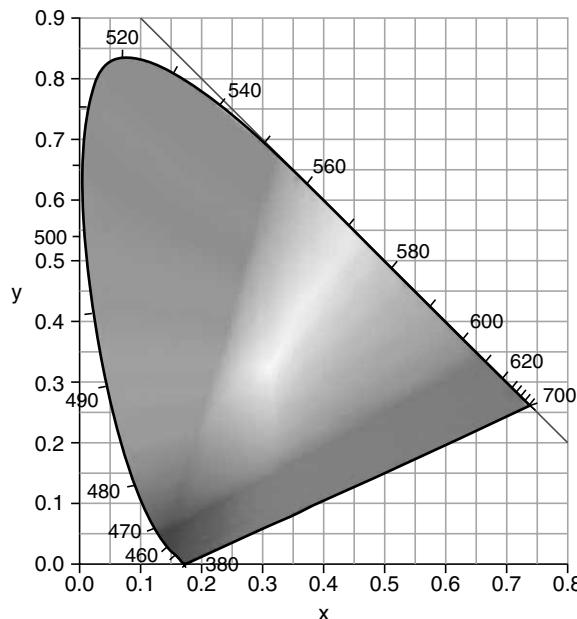


Fig. 11.10 CIE 1931 Chromaticity diagram showing various colour regions

11.6.2 Gamut

The range of colours that are physically obtainable by a colour system is called the *gamut* of the device. For a display device like a CRT, the gamut is easily prescribed, because any colour is basically a linear combination of the three primaries. The possible colours are obtained using the relation

$$c = C_{\max} \beta \quad (11.14)$$

where C_{\max} is the matrix of tristimulus values for the maximum control value of each colour gun, and β is a vector whose values vary independently between zero and unity. The graphical representation of the gamut is most often done using the chromaticity coordinates or a fixed luminance. The colour gamut of a typical CRT monitor is shown in Fig. 11.11. In the figure, the points of the triangle are the responses of the red, green and blue phosphors on the screen.

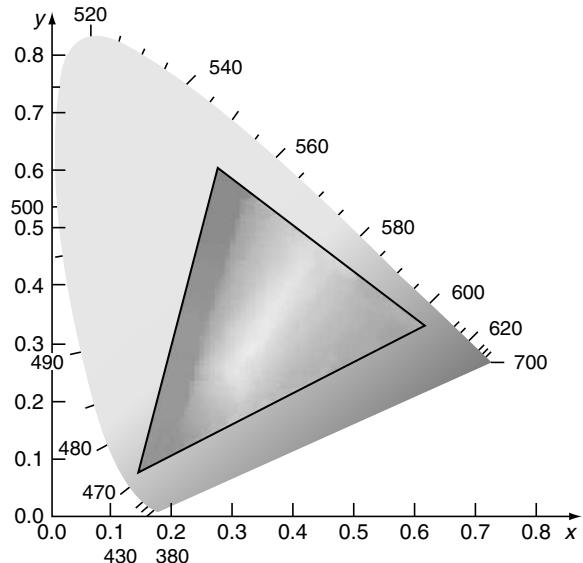


Fig. 11.11 Colour gamut of a typical monitor

MATLAB Example 2 *Read a colour image and separate the colour image into red, green and blue planes.*

Solution The MATLAB code that performs the given task is shown in Fig. 11.12. The output of the MATLAB code is given in Fig. 11.13.

```

clc
clear all
close all
a=imread('C:\Documents and Settings\esakki\My Documents\My Pictures\
f11.bmp');
a1=a;
b1=a;
c1=a;
a1(:,:,1)=0;
b1(:,:,2)=0;
c1(:,:,3)=0;
imshow(a), title('original image')
figure, imshow(a1), title('Red Missing!')
figure, imshow(b1), title('Green Missing!')
figure, imshow(c1), title('Blue Missing!')

```

Fig. 11.12 MATLAB code to remove the RGB plane

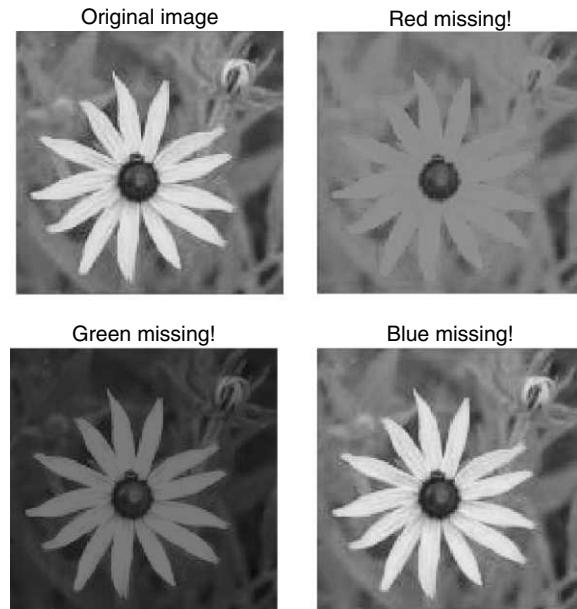


Fig. 11.13 Output of the MATLAB code shown in Fig. 11.12

11.7 COLOUR-IMAGE QUANTISATION

Colour quantisation is the term given to the effort of reducing the size of a colour space by partitioning the original colour space into larger cells. Colour-image quantisation is the process of reducing the number of colours present in a digital colour image. Colour-image quantisation is basically a lossy image-compression operation. Colour-image quantisation consists of two major steps:

- (i) Creating a colour map where a small set of colours is chosen from the possible combinations of red, green and blue.
- (ii) Mapping each colour pixel in the colour image to one of the colours in the colour map.

The main objective of colour-image quantisation is to map the set of colours in the original colour image to a much smaller set of colours in the quantised image. A smaller set of representative colours is called a *colour palette*. The mapping should minimise the difference between the original and the quantised images.

11.7.1 Classification of Colour-Quantisation Techniques

Colour quantisation, in general, can be classified into (i) uniform colour quantisation, and (ii) non-uniform colour quantisation.

11.7.2 Uniform Colour Quantisation

Uniform quantisation is the simplest method of colour quantisation. In uniform colour quantisation, each colour cell represents a cube whose edges are aligned with the colour axes and all cells have the same size. The pictorial representation of uniform colour quantisation is given in Fig. 11.14. The uniform colour quantisation scheme can be written as

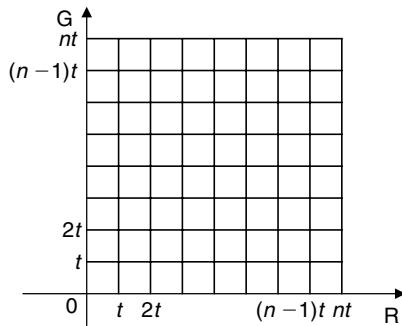


Fig. 11.14 Representation of uniform quantisation in two dimension

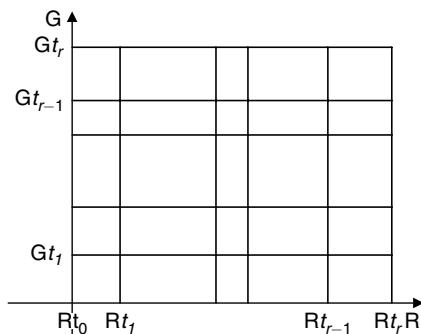


Fig. 11.15 Representation of non-uniform quantisation in two dimension

$$C = \{(R_i, G_j, B_k) \mid it \leq (i+1)t, jt \leq G < (j+1)t, kt \leq B < (k+1)t; \quad i, j, k = 0, 1..n-1\} \quad (11.15)$$

where (R, G, B) are the original colours

C is the colour space after quantisation

(R_i, G_j, B_k) is a chosen colour that corresponds to the (i, j, k)th cell

n is the number of quantisation cells in each dimension

t is the size of the quantisation cells such that nt = 256

11.7.3 Non-uniform Quantisation

In non-uniform quantisation, the colour space is divided by using multiple thresholds on each colour axis separately. The thresholds are obtained based on an analysis of the original distribution in the colour space. The pictorial representation of non-uniform quantisation is illustrated in Fig. 11.15.

The mathematical representation of non-uniform quantisation through multiple thresholds is given as

$$C = \{(R_i, G_j, B_k) \mid Rt_i < R < Rt_{i+1}, Gt_j < G < Gt_{j+1}, Bt_k < B < Bt_{k+1}; \\ i = 0, 1, \dots, r-1, j = 0, 1, \dots, g-1, k = 0, 1, \dots, b-1\} \quad (11.16)$$

where (R, G, B) is the original colour

C is the colour space after quantisation

Rt_i, Gt_j and Bt_k are chosen thresholds

Rt₀, Gt₀ and Bt₀ are equal to zero while Rt_r, Gt_g and Bt_b are equal to 256. The number of cells in the three dimensions are not necessarily equal and are represented as r, g and b. The triple (R_i, G_j, B_k) represents a chosen colour that represents the corresponding (i, j, k)th cell. In this method, the cells might not have the same size. The total number of cells can be calculated by r.g.b.

11.8 HISTOGRAM OF A COLOUR IMAGE

The histogram of the image gives the frequency of occurrence of the gray level. It gives the number of times a particular gray level occurs in the image. In the case of a colour image, the histogram is expected to give the number of times a particular colour has occurred in the image.

MATLAB Example 3: Compute the histogram of the colour image The MATLAB code that performs the histogram computation of the given image is shown in Fig. 11.16 and the corresponding outputs are given in Fig. 11.17.

```

close all;
clear all
clc;
I = imread('lavender.jpg');
imshow(I), figure
I = im2double(I);
[index, map] = rgb2ind(I);
pixels = prod(size(index));
hsv = rgb2hsv(map);
h = hsv(:, 1);
s = hsv(:, 2);
v = hsv(:, 3);
%Finds location of black and white pixels
darks = find(v < .2)';
lights = find(s < .05 & v > .85)';
h([darks lights]) = -1;
%Gets the number of all pixels for each colour bin
black = length(darks)/pixels;
white = length(lights)/pixels;
red = length(find((h > .9167 | h <= .083) & h ~= -1))/pixels;
yellow = length(find(h > .083 & h <= .25))/pixels;
green = length(find(h > .25 & h <= .4167))/pixels;
cyan = length(find(h > .4167 & h <= .5833))/pixels;
blue = length(find(h > .5833 & h <= .75))/pixels;
magenta = length(find(h > .75 & h <= .9167))/pixels;
%Plots histogram
hold on
fill([0 0 1 1], [0 red red 0], 'r')
fill([1 1 2 2], [0 yellow yellow 0], 'y')
fill([2 2 3 3], [0 green green 0], 'g')
fill([3 3 4 4], [0 cyan cyan 0], 'c')
fill([4 4 5 5], [0 blue blue 0], 'b')
fill([5 5 6 6], [0 magenta magenta 0], 'm')
fill([6 6 7 7], [0 white white 0], 'w')
fill([7 7 8 8], [0 black black 0], 'k')
axis([0 8 0 1])

```

Fig. 11.16 Histogram of a colour image

11.8.1 Histogram Equalisation of a Colour Image

The histogram equalisation of a colour image is performed by first converting the RGB image into the YIQ(NTSC) format. Then histogram equalisation is performed only for the Y component. The I and Q components are unaltered. Then in the histogram, the equalised Y , unaltered I and Q components are converted back to the RGB format.

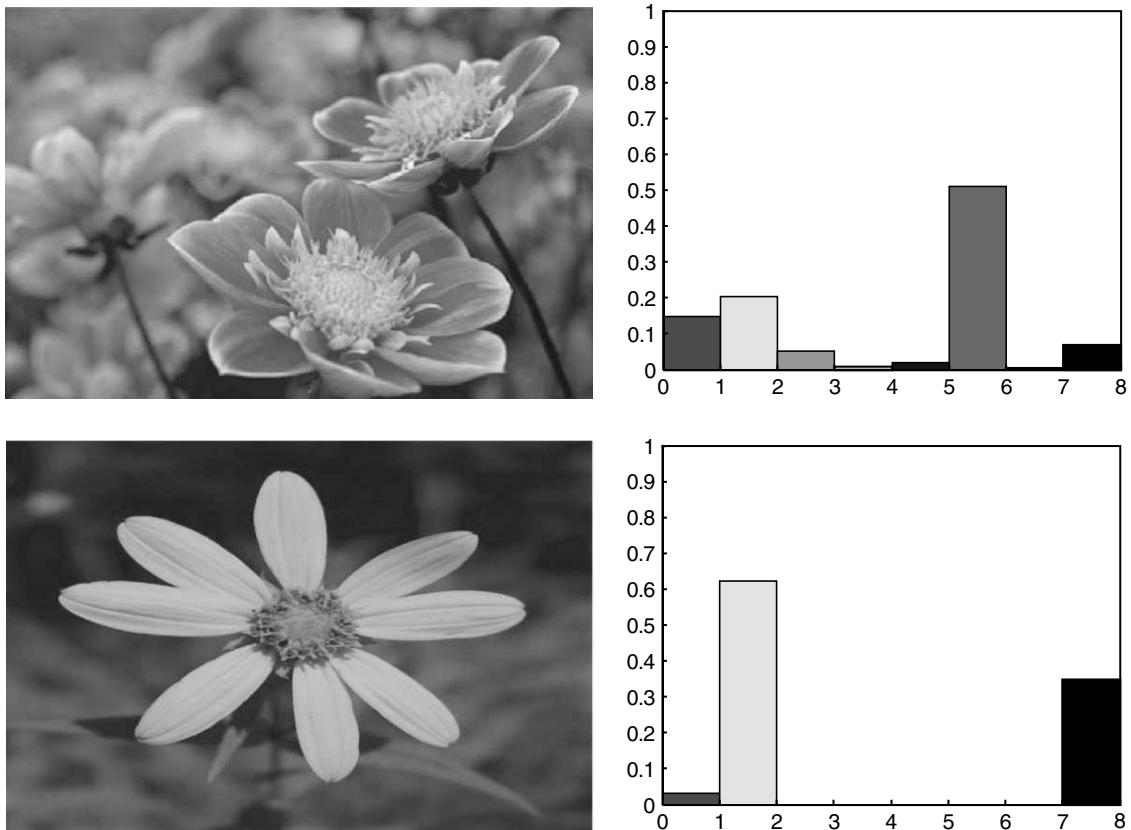


Fig. 11.17 Histogram of the colour image

MATLAB Example 4 Perform histogram equalisation of the given RGB image.

Solution The MATLAB code which performs the histogram equalisation of the colour image is given in Fig. 11.18 and the corresponding output is shown in Fig. 11.19.

```

a=imread('coconut.bmp');
%Conversion of RGB to YIQ format
b=rgb2ntsc(a);
%Histogram equalisation of Y component alone
b(:,:,1)=histeq(b(:,:,1));
%Conversion of YIQ to RGB format
c=ntsc2rgb(b);
imshow(a), title('original image')
figure, imshow(c), title('Histogram equalised image')

```

Fig. 11.18 MATLAB code for histogram equalisation of colour image

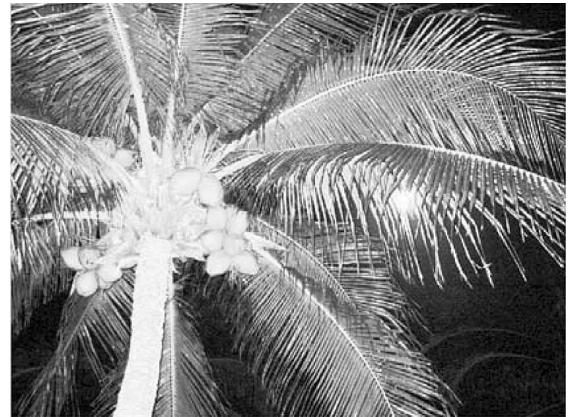


Fig. 11.19 Output of a histogram equalisation code

11.9 COLOUR-IMAGE FILTERING

Filtering of colour images can be done in two ways. In the first method, filtering of the three primaries (R, G and B) are done separately. In the second approach, the luminosity image is filtered first and then the result is utilised to a colour image. Both the approaches are valid. The second approach is computationally efficient and it is more suitable for real-time filtering.

Method 1: Filtering the Three Primaries Separately In this approach, each of the primaries is filtered separately, followed by some gain to compensate for attenuation resulting from the filter. The filtered primaries are then combined to form the coloured image. This approach is illustrated in Fig. 11.20.

Advantage of Filtering the Three Primaries Separately The main advantage is the simplicity of the approach. The concept is simple and it is extended to three primaries separately.

Drawback of Filtering the Three Primaries Separately Three separate filters are required, which is a burden with respect to the hardware point of view. The process has to be repeated three times which increases the computational complexity, is a hindrance to real-time application.

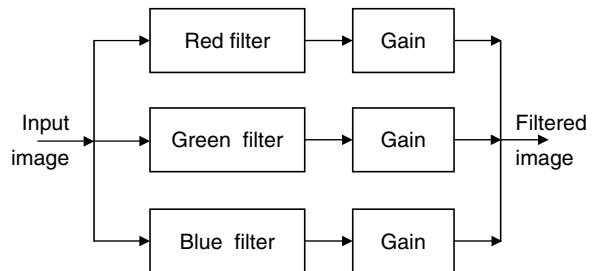


Fig. 11.20 Filtering the three primaries separately

MATLAB Example 5: Filtering of colour image corrupted by salt-and-pepper noise Read a colour image. Corrupt the image by salt-and-pepper noise. Try to restore the corrupted image using a median filter:

Solution The MATLAB code that performs the median filtering of the colour image is shown in Fig. 11.21 and the corresponding results are shown in Fig. 11.22.

```
%This program performs median filtering of the colour image
clc
clear all
close all
a=imread('C:\Documents and Settings\esakki\My Documents\My Pictures\f1.bmp');
b=imnoise(a, 'salt & pepper', 0.2);
c(:,:,1)=medfilt2(b(:,:,1));
c(:,:,2)=medfilt2(b(:,:,2));
c(:,:,3)=medfilt2(b(:,:,3));
imshow(a), title('original image')
figure, imshow(b), title('corrupted image')
figure, imshow(c), title('Median filtered image')
```

Fig. 11.21 MATLAB code that performs the median filtering of colour image

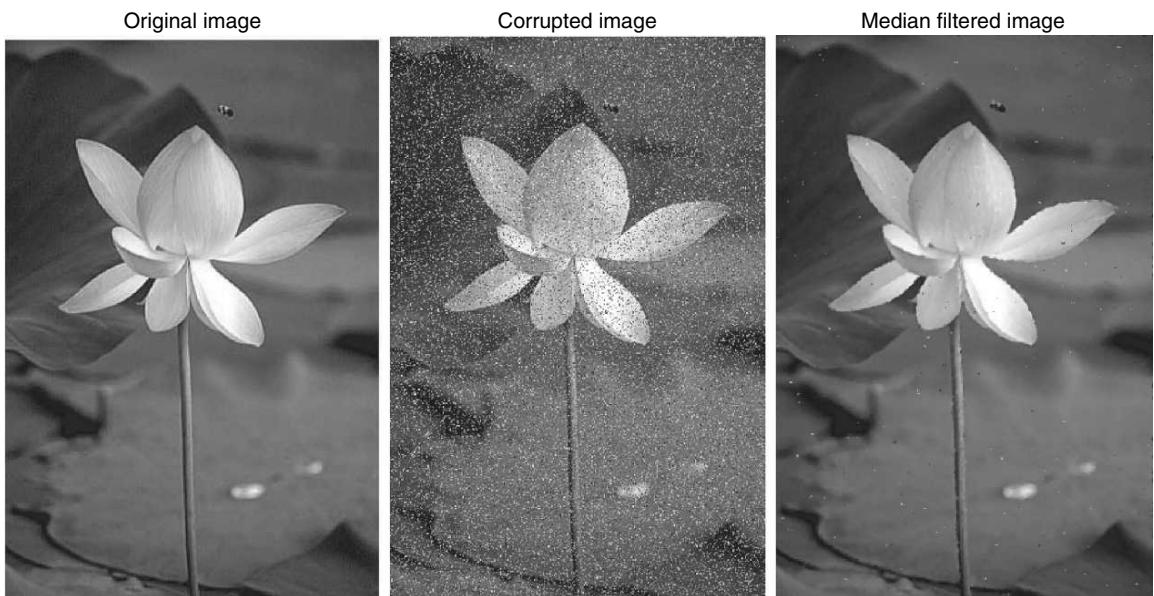


Fig. 11.22 Results of a median filter

Method 2: Filtering the Luminance Image Only In this approach, filtering is carried out on the luminance image alone. The filtered luminance image can be used to adjust the levels of the three primaries without altering the ratios of R:G:B at every pixel. The pictorial representation of this method is given in Fig. 11.23. The output primaries are adjusted using the gain at every pixel, which is obtained by dividing the output luminance or that pixel over the input luminance.

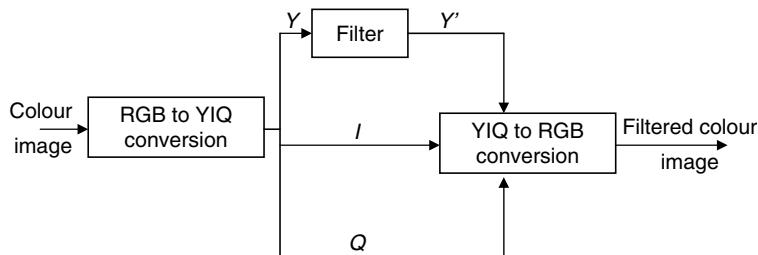


Fig. 11.23 Filtering only the luminance image

MATLAB Example 6: Filtering only the luminance component Read the colour image. Convert the RGB format to YIQ format (NTSC). Filter only the Y component (high-pass filtering). Do not disturb the I and Q components. Now convert the filtered Y component, I component and Q component back to the RGB format and check the result. Comment on the observed result.

Solution The MATLAB code that performs the high-pass filtering of only the Y component is shown in Fig. 11.24 and the corresponding outputs are illustrated in Fig. 11.25.

```

clc;
close all;
clear all;
a=imread('sunflower.jpg');
yiq=rgb2ntsc(a);
%Extract the Y component alone
b1=yiq(:, :, 1);
h=[-1 -1 -1;-1 8 -1;-1 -1 -1];
%Perform high pass filtering only on Y component
c1=conv2(b1, h, 'same');
yiq(:, :, 1)=c1;
% Convert YIQ to RGB format
a1=ntsc2rgb(yiq);
figure, imshow(a), title('original image')
figure, imshow(a1), title('High pass filtered image')
  
```

Fig. 11.24 High-pass filtering of only the Y component

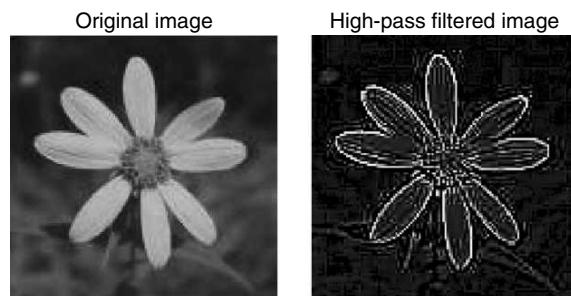


Fig. 11.25 Result of high-pass filtering of the Y component

11.10 GAMMA CORRECTION OF A COLOUR IMAGE

In a CRT monitor, the voltage V driving the electron flow is related to the intensity I of light emitted by the phosphor hit by the electrons according the formula

$$I \approx V^\gamma \quad (11.17)$$

where γ is a constant that depends on the phosphor. This non-linearity causes serious distortions in the colours of the image displayed with some areas being too dark, while others being saturated. To avoid this problem, an amplitude filter is required that corrects the intensity of each pixel before the information is passed to the CRT. The correction is given by

$$V \approx I^{\frac{1}{\gamma}} \quad (11.18)$$

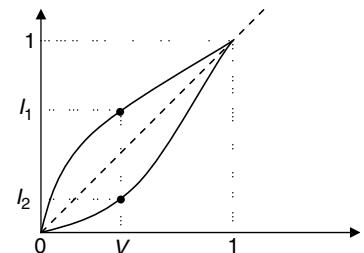


Fig. 11.26 Gamma-correction curve

The correction given in Eq. (11.18) must be applied to the pixel intensities before the signal is converted into voltage at the CRT. The gamma-correction curve for the monitor is shown in Fig. 11.26.

Nowadays gamma correction is a factor by which software can correct this non-linearity, resulting in a correctly displayed image.

11.10.1 Gamma Correction Through a Look-Up-Table

The flow chart for gamma correction through a look-up-table is given in Fig. 11.27.

The crucial step is the look-up-table formation and the selection of the gamma value. The formula to create a look-up-table for an 8-bit image is given below.

$$\text{Look-up-table} = \left[\text{Maximum intensity} \times \left(\frac{[0 : \text{Maximum intensity}]}{\text{Maximum intensity}} \right)^\gamma \right] \quad (11.19)$$

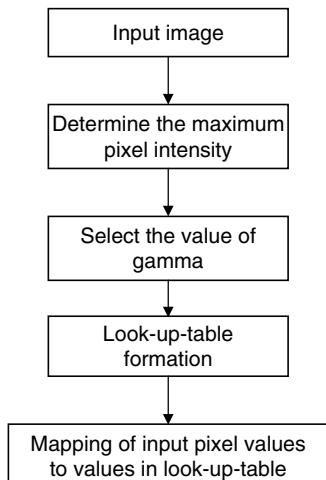


Fig. 11.27 Flowchart for gamma correction

Example 11.1 Perform gamma correction of the image

$$\begin{bmatrix} 2 & 4 & 6 & 8 \\ 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 10 \\ 12 & 14 & 15 & 13 \end{bmatrix}$$

by assuming the

value of gamma to be 0.5.

Solution

Step 1 From the given image, the maximum intensity is found to be 15.

Step 2 The value of gamma is given as 0.5.

Step 3 *Look-up-table formation*

The input image is of size 4×4 , hence a total of 16 entries will be present in the look-up table which

are given as
$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$
. Our aim is to determine these sixteen entries from 0 to 15. The formula

used to create the look-up-table entries is

$$\text{Look-up-table} = \left\lfloor \text{Maximum intensity} \times \left(\frac{[0 : \text{Maximum intensity}]}{\text{Maximum intensity}} \right)^\gamma \right\rfloor$$

$$\text{Look-up-table entry 0: } \left\lfloor 15 \times \left(\frac{0}{15} \right)^{0.5} \right\rfloor = 0$$

$$\text{Look-up-table entry 1: } \left\lfloor 15 \times \left(\frac{1}{15} \right)^{0.5} \right\rfloor = \lfloor 3.872 \rfloor = 3$$

$$\text{Look-up-table entry 2: } \left\lfloor 15 \times \left(\frac{2}{15} \right)^{0.5} \right\rfloor = \lfloor 5.475 \rfloor = 5$$

$$\text{Look-up-table entry 3: } \left\lfloor 15 \times \left(\frac{3}{15} \right)^{0.5} \right\rfloor = \lfloor 6.708 \rfloor = 6$$

$$\text{Look-up-table entry 4: } \left\lfloor 15 \times \left(\frac{4}{15} \right)^{0.5} \right\rfloor = \lfloor 7.745 \rfloor = 7$$

$$\text{Look-up-table entry 5: } \left\lfloor 15 \times \left(\frac{5}{15} \right)^{0.5} \right\rfloor = \lfloor 8.660 \rfloor = 8$$

$$\text{Look-up-table entry 6: } \left\lfloor 15 \times \left(\frac{6}{15} \right)^{0.5} \right\rfloor = \lfloor 9.4868 \rfloor = 9$$

$$\text{Look-up-table entry 7: } \left\lfloor 15 \times \left(\frac{7}{15} \right)^{0.5} \right\rfloor = \lfloor 10.2469 \rfloor = 10$$

$$\text{Look-up-table entry 8: } \left\lfloor 15 \times \left(\frac{8}{15} \right)^{0.5} \right\rfloor = \lfloor 10.954 \rfloor = 10$$

$$\text{Look-up-table entry 9: } \left\lfloor 15 \times \left(\frac{9}{15} \right)^{0.5} \right\rfloor = \lfloor 11.618 \rfloor = 11$$

$$\text{Look-up-table entry 10: } \left\lfloor 15 \times \left(\frac{10}{15} \right)^{0.5} \right\rfloor = \lfloor 12.2474 \rfloor = 12$$

$$\text{Look-up-table entry 11: } \left\lfloor 15 \times \left(\frac{11}{15} \right)^{0.5} \right\rfloor = \lfloor 12.845 \rfloor = 12$$

$$\text{Look-up-table entry 12: } \left\lfloor 15 \times \left(\frac{12}{15} \right)^{0.5} \right\rfloor = \lfloor 13.4164 \rfloor = 13$$

$$\text{Look-up-table entry 13: } \left\lfloor 15 \times \left(\frac{13}{15} \right)^{0.5} \right\rfloor = \lfloor 13.945 \rfloor = 13$$

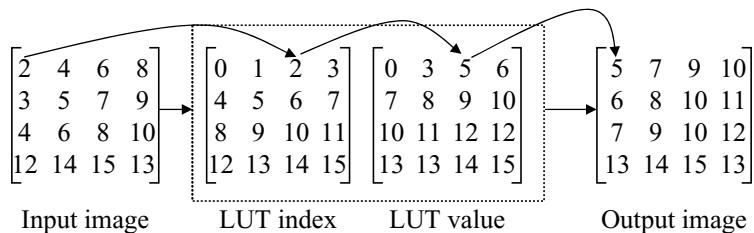
$$\text{Look-up-table entry 14: } \left\lfloor 15 \times \left(\frac{14}{15} \right)^{0.5} \right\rfloor = \lfloor 14.49 \rfloor = 14$$

$$\text{Look-up-table entry 15: } \left\lfloor 15 \times \left(\frac{15}{15} \right)^{0.5} \right\rfloor = \lfloor 15 \rfloor = 15$$

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 3 & 5 & 6 \\ 7 & 8 & 9 & 10 \\ 10 & 11 & 12 & 12 \\ 13 & 13 & 14 & 15 \end{bmatrix}$$

Step 4 *Mapping of input image pixel values to the values in the look-up table*

In our case, the first pixel value is '2'. The corresponding value in the look-up table is 5. The next pixel value is 4, and the corresponding value in the look-up table is 7. The third pixel value is 6, and the corresponding value in the look-up table is 9. This mapping procedure is repeated for all the pixels in the input image. After mapping, the new pixel values are



MATLAB Example 7: Gamma correction *Perform gamma correction for the given colour image for different values of gamma and comment on the output result.*

Solution The MATLAB code that performs the gamma correction of the input colour image is shown in Fig. 11.28 and the corresponding outputs are shown in Fig. 11.29

From Fig. 11.29, it is obvious that different values of gamma give different output images. When the value of gamma is less than one, the image appears brighter. When the value of gamma is greater than one, the image is darker. When the value of gamma is one, the output image resembles the original image.

```
close all;
clear all;
clc;
I=imread('deer4.jpg');
gamma=1;
max_intensity =255;%for uint8 image
%Look up table creation
LUT = max_intensity .* ( ([0:max_intensity]./max_intensity).^gamma );
LUT = floor(LUT);
%Mapping of input pixels into lookup table values
J = LUT(double(I)+1);
imshow(I), title('original image');
figure, imshow(uint8(J)), title('Gamma corrected image')
xlabel(sprintf('Gamma value is %g', gamma))
```

Fig. 11.28 MATLAB code to perform gamma correction

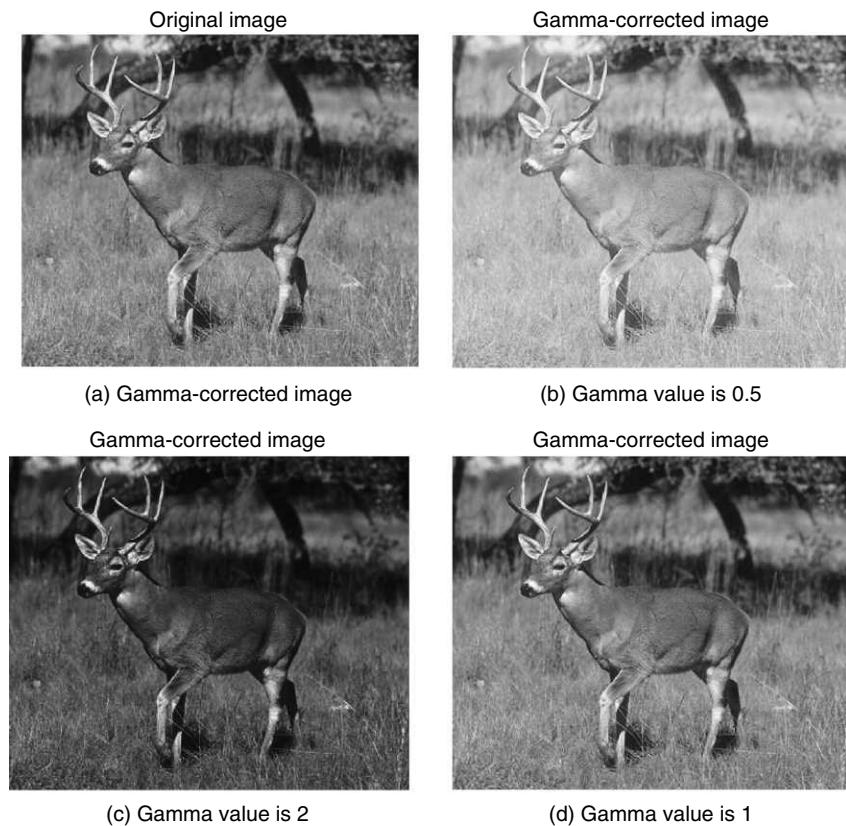


Fig. 11.29 Output of gamma correction code

11.11 PSEUDO-COLOUR

The human eye finds it easier to compare colours than variations in intensity. With colour, the human perception is less sensitive to context than it is with gray levels and the illusion whereby intensity is perceived relative to nearby intensities is less of a problem.

The MATLAB code that performs the pseudo-colouring operation is given in Fig. 11.30, and the corresponding output is shown in Fig. 11.31.

From Fig. 11.31, it is obvious that a pseudo-colour image can be derived from a grayscale image by mapping each pixel value to a colour according to a predefined function. Pseudo-colouring does not increase the information contents in the original image; it can make some details more visible by increasing the colour space between successive gray levels.

```
%This Code performs pseudo-colouring
clc;
clear all;
input_img=imread('rice.tif');
[m n]=size(input_img);
input_img=double(input_img);
for i=1:m
    for j=1:n
        if input_img(i, j)>=0 & input_img(i, j)<50
            output_img(i, j, 1)=input_img(i, j)+50;
            output_img(i, j, 2)=input_img(i, j)+100;
            output_img(i, j, 3)=input_img(i, j)+10;
        end
        if input_img(i, j)>=50 & input_img(i, j)<100
            output_img(i, j, 1)=input_img(i, j)+35;
            output_img(i, j, 2)=input_img(i, j)+128;
            output_img(i, j, 3)=input_img(i, j)+10;
        end
        if input_img(i, j)>=100 & input_img(i, j)<150
            output_img(i, j, 1)=input_img(i, j)+152;
            output_img(i, j, 2)=input_img(i, j)+130;
            output_img(i, j, 3)=input_img(i, j)+15;
        end
        if input_img(i, j)>=150 & input_img(i, j)<200
            output_img(i, j, 1)=input_img(i, j)+50;
            output_img(i, j, 2)=input_img(i, j)+140;
            output_img(i, j, 3)=input_img(i, j)+25;
        end
        if input_img(i, j)>=200 & input_img(i, j)<=256
            output_img(i, j, 1)=input_img(i, j)+120;
            output_img(i, j, 2)=input_img(i, j)+160;
            output_img(i, j, 3)=input_img(i, j)+45;
        end
    end
end
subplot(2, 2, 1), imshow(uint8(input_img)), title('Input Image')
subplot(2, 2, 2), imshow(uint8(output_img)), title('Pseudo Coloured Image')
```

Fig. 11.30 MATLAB code that performs pseudo-colouring operation

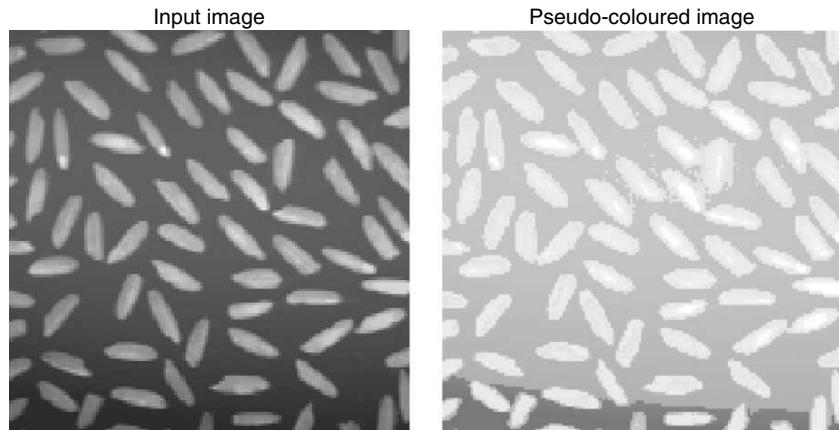


Fig. 11.31 Output of pseudo-colouring operation

11.12 COLOUR-IMAGE SEGMENTATION

The goal of image segmentation is grouping the pixels that have similar feature in an image from the stand-point of human visual system. There are many techniques for gray image segmentation. But the colour image is a multidimensional vector; the segmentation techniques for gray images cannot be applied to colour images directly. Colour has always been considered as a strong tool for image segmentation. In a static scene, colour difference is one of the easiest globe cues to tell different objects apart. In colour-image segmentation, it is important to preserve the human perceivable colour boundary.

MATLAB Example 8 Read an RGB image and segment it using the threshold method

Solution The MATLAB code that segments the image from the background using thresholding approach is shown in Fig. 11.32 and the corresponding output is illustrated in Fig. 11.33.

```

clc
clear all
close all
a=imread('Tomato2.jpg');
%Conversion of RGB to YCbCr
b=rgb2ycbcr(a);
%Threshold is applied only to Cb component
mask=b(:,:,2)>120;
imshow(a), title('original image')
figure, imshow(mask), title('Segmented image')

```

Fig. 11.32 MATLAB code to separate an object from its background

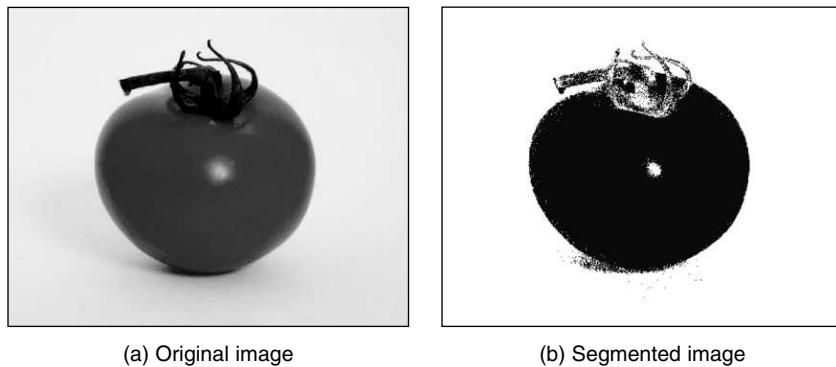


Fig. 11.33 Result of MATLAB code shown in Fig. 11.32

From Fig. 11.32, it is obvious that the given RGB is converted into YCbCr format. After conversion, the threshold condition is applied only to the Cb component; the Y and Cr components are unaltered. The choice of the threshold depends on the nature of the image.

Application of Colour-image Segmentation Colour-image segmentation can be used for the following:

- (i) To detect skin cancer—the part of the skin affected by cancer looks different from the other parts of the skin. Through colour-image segmentation, it is possible to identify the affected areas to a certain extent.
- (ii) To classify different types of plants in aerial images.
- (iii) In computer vision, image segmentation plays a crucial role. In industries, based on colour-image segmentation, it is possible to identify defective fruits from normal fruits in a conveyor mechanism.
- (iv) Colour segmentation plays a crucial role in SAR images. The segmentation technique can be used to identify vegetation area from other areas.



Summary

- Visible light is in the range 400 nm (blue) to 700 nm (red).
- Cones have three different kinds of colour-sensitive pigments, each responding to a different range of wavelengths.
- The combination of the responses of these different receptors gives us colour perception which is generally termed the tristimulus model of colour vision.
- CIE uses three primaries X , Y and Z to replace red, green and blue.
- The Y component was chosen to correspond to luminous efficiency

$$Y = \text{luminance} \quad X, Z = \text{chromaticity}$$
- The normalised values are given by

$$x = \frac{X}{X+Y+Z}; y = \frac{Y}{X+Y+Z}; z = \frac{Z}{X+Y+Z}$$

- The colour space spanned by a set of primary colours is called a colour gamut.
- The different colour models are RGB, CMY, YC_bC_r , YIQ, etc.
- The RGB model is used most often for additive models. In an RGB model, adding a colour and its complement create white.
- The most common subtractive model is the CMY model. In a CMY model, adding a subtractive colour and its complement create black.
- Each pixel of the colour image will have three values, one each for the red, green and blue components.
- The histogram of a colour image can be considered to consist of three separate one-dimensional histograms, one for each tricolour component.

Review Questions

1. What is meant by the term colour? How do human beings perceive colour?

Colour is a perceptual phenomenon related to the human response to different wavelengths of light, mainly in the region of 400 to 700 nanometres (nm). The perception of colour arises from the sensitivities of three types of neurochemical sensors in the retina known as long (L), medium (M) and short (S) cones.

2. What are tristimulus values?

The relative amounts of the three primary colours of light required to produce a colour of a given wavelength are called tristimulus values.

3. What is a colour space? Mention its classification.

A colour space allows one to represent all the colours perceived by human beings. The colour space can be broadly classified into (i) RGB, (ii) CMY, (iii) YIQ, and (iv) HSI colour space.

4. What is the difference between hardware-oriented and perceptive colour spaces? Give examples for each kind.

A hardware-oriented colour space is suitable for representing the colour components as they are handled by the display devices like monitors and printers. Examples of hardware-oriented colour spaces are RGB and CMY colour spaces.

A perceptive colour space represents colours in terms of components which are correlates of the signals perceived by the human visual system. An example of a perceptive colour space is HIS.

5. What are the CIE chromaticity coordinates associated with the CIE XYZ colour space? How are they obtained from XYZ values? Do the chromaticity coordinates encode most of the spatial high frequencies of a colour image?

The chromaticity coordinates associated with XYZ are

$$x = \frac{X}{X + Y + Z} \quad \text{and} \quad y = \frac{Y}{X + Y + Z}$$

The chromatic signals of a colour image encode low spatial frequencies.

6. What is meant by the term Weber ratio? Explain.

Weber ratio is given by $\frac{\Delta I_c}{I}$, where I_c is the increment of illumination discriminable 50% of the time with background illumination I . The Weber ratio for the human eye is large at low levels of illumination, meaning that brightness discrimination is poor and the change in illumination must be large to be perceived.

7. What is meant by pseudo-colouring? For what purpose is it useful? Explain how a pseudo coloured image can be obtained.

A grayscale image is transformed into a colour image by pseudo-colouring. This is to use the fact that a human being can differ between around 30 different gray levels but around 350000 different colours. A pseudo-colour image can be obtained from a grayscale image by gray-level slicing. The grayscale is divided into intervals to each of which a specific colour is assigned.

Problems

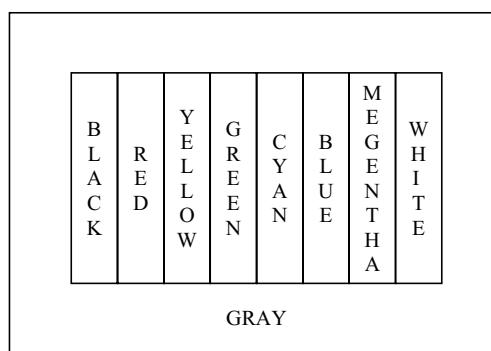
- 11.1 Three colours are defined as follows in terms of the CIE XYZ primaries:

$$[Q_1] = 0.5[X] + 0.3[Y]$$

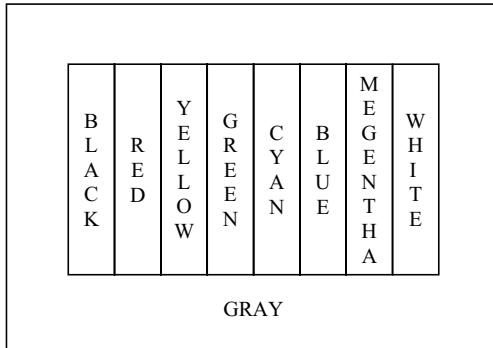
$$[Q_2] = 0.3[X] + 0.5[Y]$$

$$[Q_3] = 0.1[X] + 0.2[Y] + 0.7[Z]$$

- (i) Identify, using correct notation, the tristimulus values of the colour $[Q_3]$ with respect to the XYZ primaries.
 (ii) Compute and identify with correct notation the XYZ chromaticities of $[Q_1]$, $[Q_2]$ and $[Q_3]$ and plot them on the XYZ chromaticity diagram.
- 11.2 Consider any two valid colours C_1 and C_2 with coordinates (x_1, y_1) and (x_2, y_2) in the chromaticity diagram of Fig.11.10. Derive the necessary general expression for computing the relative percentages of colours C_1 and C_2 composing, a given colour C that is known to lie on the straight line joining these two colours.
- 11.3 Consider three valid colours C_1 , C_2 and C_3 with coordinates (x_1, y_1) , (x_2, y_2) and (x_3, y_3) in the chromaticity diagram of Fig.11.10. Derive the necessary general expressions for computing the relative percentages of C_1 , C_2 and C_3 composing a given colour C that is known to lie within the triangle whose vertices are at the coordinates of these three colours.
- 11.4 Sketch the RGB components of the following image as they would appear on a monochrome monitor. All colours are at maximum intensity and saturation. Consider the middle gray border as part of the image.



- 11.5 Sketch the HSI components as they would appear on a monochrome monitor.



- 11.6 Derive the CMY intensity transformation function

$$s_i = kr_i + (1 - k), \quad i = 1, 2, 3, \text{ for } (C, M, Y).$$

from its RGB counterpart $s_i = kr_i, i = 1, 2, 3, \text{ for } (R, G, B)$.

- 11.7 Given a colour image represented in terms of RGB components, how are the corresponding CMY coordinates derived?
- 11.8 What are the CIE chromaticity coordinates associated with the CIE XYZ colour space? How are they obtained from XYZ values? Do the chromaticity coordinates encode most of the spatial high frequencies of a colour image?
- 11.9 L^* , a^* , and b^* are defined in the CIE colour coordinates in the following way:

$$L^* = 25(100Y/Y_0)^{1/3} - 16 \quad 1 \leq 100Y \leq 100$$

$$a^* = 500 \left[\left(\frac{X}{X_0} \right)^{1/3} - \left(\frac{Y}{Y_0} \right)^{1/3} \right]$$

$$b^* = 200 \left[\left(\frac{Y}{Y_0} \right)^{1/3} - \left(\frac{Z}{Z_0} \right)^{1/3} \right]$$

$$(\Delta s)^2 = (\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2$$

Determine Δs when

- (a) X , Y and Z are changed by 5%
- (b) X , Y and Z are changed by 10%
- (c) X_0 , Y_0 and Z_0 are changed by 10%. X , Y and Z are assumed to be constant.

What can be said about Δs ?

- 11.10 Three colours are defined as follows in terms of the CIE XYZ primaries:

$$[Q_1] = 1.25[X] + 1.0[Y] + 0.25[Z]$$

$$[Q_2] = 0.2[X] + 0.6[Y] + 0.2[Z]$$

$$[Q_3] = 0.5[X] + 0.5[Y] + 1.5[Z]$$

Compute and identify with correct notation the XYZ chromaticities of $[Q_1]$, $[Q_2]$ and $[Q_3]$ and plot them on an XYZ chromaticity diagram. Indicate clearly on the diagram the chromaticities of all colours that can physically be synthesised using linear combinations of $[Q_1]$, $[Q_2]$ and $[Q_3]$ with positive coefficients. Show the chromaticity coordinates of the colour $[P] = 0.5[Q_1] + 0.4[Q_2] + 0.3[Q_3]$ on the diagram.

References

Books

1. Jonas Gomes and Luiz Velho, *Image Processing for Computer Graphics*, Springer-Verlag, New York, 1997
2. A R Smith, *Colour Gamut Transform Pairs*, SIGGRAPH 78 Proceedings, Computer Graphics, vol. 12, pp. 12–19, 1978
3. T N Cornsweet, *Visual Perception*, Academic Press, New York, 1970
4. Gaurav Sharma (Editor), *Digital Colour Imaging Handbook*, CRC Press 2000
5. G Wyszecki and W S Stiles, *Colour Science: Concepts and Methods, Quantitative Data and Formulas*, Wiley, New York, 1982
6. M D Fairchild, *Colour Appearance Models*, Addison–Wesley, Reading, MA, 1998
7. K N Plataniotis and A N Venetsanopoulos, *Colour Image Processing and Applications*, Springer–Verlag, Heidelberg, 2000

Journal Papers

1. H J Trussell, E Saber and M Vrhel, eds., *Colour Image Processing Special Issue*, IEEE Signal Processing Magazine, 22, January 2005
2. G Sharma, M J Vrhel and H J Trussell, *Digital Colour Processing*, IEEE Transaction on Image Processing, vol. 6, no.7, pp. 901–932, July 1997
3. R Balasubramanian, J P Allebach and C A Bouman, *Colour-image Quantisation with Use of a Fast Binary Splitting Technique*, Journal of Optical Society of America, vol. 11, 2777–2786, 1994
4. J Zheng, K P Valavanis and J M Gauch, *Noise Removal from Colour Images*, Journal of Intelligent and Robotic Systems, vol. 7, pp. 257–285, 1993
5. G S Robinson, *Colour Edge Detection*, *Optical Engineering*, vol. 16, pp. 479–484, 1977

Web Resources

1. Michael J Vrhel's home page gives useful information regarding colour-image processing:
<http://www.viegroup.com/mvrhelweb/colour.html>
2. Professor Charles A. Bouman's lecture notes gives introduction to colour space and chromaticity diagram:
<http://cobweb.ecn.purdue.edu/~bouman/ee637/notes/pdf/ColourSpaces.pdf>
3. Dr K R Rao lecture notes is a valuable resource: <http://www-ee.uta.edu/dip/>

12



Learning Objectives

Wavelet transform is an efficient tool to represent an image. The wavelet transform allows multi-resolution analysis of an image. The aim of the transform is to extract relevant information from an image. This chapter discusses the application of wavelet transform in the field of image processing. Wavelet transform itself is a vast area with rigorous mathematical background. The objective is not to introduce the wavelet concept, but to present the applications of wavelet transform in the field of image compression, image denoising and image watermarking. After completing this chapter, the reader should have a fundamental idea on the following concepts:

Need for wavelet transform

Concept of multi-resolution and its application to image processing

Wavelet transform in the field of image compression

Wavelet transform in the field of image denoising

Wavelet transform in the area of image watermarking

Wavelet-based Image Processing

12.1 INTRODUCTION

Wavelet transform has received considerable attention in the field of image processing due to its flexibility in representing non-stationary image signals and its ability in adapting to human visual characteristics. Wavelet transforms are the most powerful and the most widely used tool in the field of image processing. Their inherent capacity for multi-resolution representation akin to the operation of the human visual system motivated a quick adoption and widespread use of wavelets in image-processing applications. A wavelet transform divides a signal into a number of segments, each corresponding to a different frequency band. The application of wavelets in the field of image compression, image denoising and watermarking are highlighted in this chapter. The main focus of this chapter is not to introduce the theory of wavelet transform to the reader, but to give applications of wavelet transform in different fields of image processing. To gain good knowledge about wavelets, a good understanding of linear algebra and signal processing theory is a must.

12.2 EVOLUTION OF WAVELET TRANSFORM

Fourier transform is a powerful tool that has been available to signal analysts for many years. It gives information regarding the frequency content of a signal. However, the problem with using Fourier transforms is that frequency analysis cannot offer both good frequency and time resolution at the same time. A Fourier transform does not give information about the time at which a particular frequency has occurred in the signal. Hence, a Fourier transform is not an effective tool to analyse a non-stationary signal. To overcome this problem, *windowed Fourier transform*, or *short-time Fourier transform*, was introduced. Even though a short-time Fourier transform has the ability to provide time information, multi-resolution is not possible with short-time Fourier transforms. Wavelet is the answer to the multi-resolution problem. A wavelet has the important property of not having a fixed-width sampling window. The wavelet transform can be broadly classified into (i) continuous wavelet transform, and (ii) discrete wavelet transform. For long signals, continuous wavelet transform can be time consuming since it needs to integrate over all times. To overcome the time complexity, discrete wavelet transform was introduced. Discrete wavelet transforms can be implemented through sub-band coding. The DWT is useful in image processing because it can simultaneously localise signals in time and scale, whereas the DFT or DCT can localise signals only in the frequency domain.

12.2.1 Heisenberg Uncertainty Principle

The Heisenberg uncertainty principle was originally stated in physics, and claims that it is impossible to know both the position and momentum of a particle simultaneously. However, it has an analog basis in signal processing. In terms of signals, the Heisenberg uncertainty principle is given by the rule that it is impossible to know both the frequency and time at which they occur. The time and frequency domains are complimentary. If one is local, the other is global. Formally, the uncertainty principle is expressed as

$$(\Delta t)^2 (\Delta \omega)^2 \geq \frac{1}{4}$$

In the case of an impulse signal, which assumes a constant value for a brief period of time, the frequency spectrum is infinite; whereas in the case of a step signal which extends over infinite time, its frequency spectrum is a single vertical line. This fact shows that we can always localise a signal in time or in frequency but not both simultaneously. If a signal has a short duration, its band of frequency is wide and vice versa.

12.2.2 Short-Time Fourier Transform (STFT)

The STFT is a modified version of the Fourier transform. The Fourier transform separates the input signal into a sum of sinusoids of different frequencies and also identifies their respective amplitudes. Thus, the Fourier transform gives the frequency–amplitude representation of an input signal. The Fourier transform is not an effective tool to analyse non-stationary signals. STFT and wavelet transforms are effective tools to analyse non-stationary signals. In STFT, the non-stationary signal is divided into small portions, which are assumed to be stationary. This is done using a window function of a chosen width, which is shifted and multiplied with the signal to obtain small stationary signals.

The short-time Fourier transform maps a signal into a two-dimensional function of time and frequency. The STFT of a one-dimensional signal $x(t)$ is represented by $X(\tau, \omega)$ where

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t)w^*(t-\tau)e^{-j\omega t} dt \quad (12.1)$$

Here, $x(t)$ represents the input signal

$w(t)$ is a temporal window with finite support

$X(\tau, \omega)$ is the time frequency atom

Here, the non-stationary signal $x(t)$ is assumed to be approximately stationary in the span of the temporal window $w(t)$.

In the case of a 2D signal $f(x, y)$, the space–frequency atom is given by

$$X(\tau_1, \tau_2, \omega_1, \omega_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)w^*(x-\tau_1, y-\tau_2)e^{-j(\omega_1 x + \omega_2 y)} dxdy \quad (12.2)$$

Here, τ_1, τ_2 represents the spatial position of the two-dimensional window $W(x, y)$. ω_1, ω_2 represents the spatial frequency parameters. The performance of STFT for specific application depends on the choice of the window. Different types of windows that can be used in STFT are Hamming, Hanning, Gaussian and Kaiser windows.

The time–frequency tiling of STFT is given in Fig. 12.1.

Drawback of STFT The main drawback of STFT is that once a particular size time window is chosen, the window remains the same for all frequencies. To analyse the signal effectively, a more flexible approach is needed where the window size can vary in order to determine more accurately either the time or frequency information of the signal. This problem is known as the resolution problem.

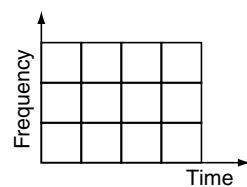


Fig. 12.1 Time–frequency tiling of STFT

12.3 WAVELET

A wave is an oscillating function of time or space that is periodic. The wave is an infinite length continuous function in time or space. In contrast, wavelets are localised waves. A wavelet is a waveform of an effectively limited duration that has an average value of zero.

A function $\psi(x)$ can be called a wavelet if it posses the following properties:

1. The function integrates to zero, or equivalently, its Fourier transform denoted as $\psi(\omega)$ is zero at the origin:

$$\int_{-\infty}^{\infty} \psi(x) dx = 0 \quad (12.3)$$

This implies $\psi(\omega)|_{\omega=0} = 0$ in the frequency domain.

2. It is square integrable, or equivalently, has finite energy:

$$\int_{-\infty}^{\infty} |\psi(x)|^2 dx < \infty \quad (12.4)$$

3. The Fourier transform $\psi(x)$ must satisfy the admissibility condition given by

$$C_{\psi} = \int_{-\infty}^{\infty} \frac{|\psi(\omega)|^2}{|\omega|} d\omega < \infty \quad (12.5)$$

Interpretation of Eqs. (12.3), (12.4) and (12.5)

Equation (12.3) suggests that the function is either oscillatory or has a wavy appearance.

Equation (12.4) implies that most of the energy in $\psi(x)$ is confined to a finite interval, or in other words, $\psi(x)$ has good space localisation. Ideally, the function is exactly zero outside the finite interval. This implies that the function is a compactly supported function.

Equation (12.5) is useful in formulating the inverse wavelet transform. From Eq. (12.5), it is obvious that $\psi(\omega)$ must have a sufficient decay in frequency. This means that the Fourier transform of a wavelet is localised, that is, a wavelet mostly contains frequencies from a certain frequency band. Since the Fourier transform is zero at the origin, and the spectrum decays at high frequencies, a wavelet has a bandpass characteristic. Thus a wavelet is a ‘small wave’ that exhibits good time–frequency localisation.

A family of wavelets can be generated by dilating and translating the mother wavelet $\psi(x)$ which is given by

$$\psi_{(a, b)}(x) = \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) \quad (12.6)$$

Here, a is the scale parameter and b is the shift parameter.

12.4 WAVELET TRANSFORM

The wavelet transform (WT) provides a time–frequency representation of the signal. The wavelet transform was developed to overcome the shortcomings of the short-time Fourier transform, which can be used to analyse non-stationary signals. The main drawback of the STFT is that it gives a constant resolution at all frequencies, while the wavelet transform uses a multi-resolution technique by which different frequencies are analysed with different resolutions. The wavelet transform is generally termed *mathematical microscope* in which big wavelets give an approximate image of the signal, while the smaller wavelets zoom in on the small details. The basic idea of the wavelet transform is to represent the signal to be analysed as a superposition of wavelets.

12.5 CONTINUOUS WAVELET TRANSFORM

The Continuous Wavelet Transform (CWT) of a one-dimensional signal $x(t)$ is given by

$$W_f(a, b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \psi^* \left(\frac{t-b}{a} \right) dt \quad (12.7)$$

The continuous wavelet transform is a function of two variables a and b . Here, a is the scaling parameter and b is the shifting parameter. $\psi(t)$ is the mother wavelet or the basis function and it is used as a prototype for generating all the basis functions. The translation parameter or the shifting parameter b gives the time information in the wavelet transform. It indicates the location of the window as it is shifted through the signal. The scale parameter a gives the frequency information in the wavelet transform. A low scale corresponds to wavelets of smaller width, which gives the detailed information in the signal. A high scale corresponds to wavelets of larger width which gives the global view of the signal.

The inverse 1D continuous wavelet transform is given by

$$x(t) = \frac{1}{C_\psi} \int_0^{\infty} \int_{-\infty}^{\infty} W_f(a, b) \Psi_{a, b}(t) db \frac{da}{a^2} \quad (12.8)$$

where

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\psi(\omega)|^2}{\omega} d\omega < \infty \quad (12.9)$$

The wavelet transform of a signal using the CWT is obtained by changing the scale of the analysis window, shifting the window in time, multiplying the signal and integrating the result over all time.

12.6 2D CONTINUOUS WAVELET TRANSFORM

The expression for 2D continuous wavelet transform of the image $f(x, y)$ is given by

$$\frac{1}{\sqrt{a}} \iint f(x, y) \psi \left(\frac{x-m}{a}, \frac{y-n}{a} \right) dx dy \quad (12.10)$$

where m, n are shifting parameters and a is the scaling parameter. The comparison between continuous wavelet transform and discrete wavelet transform is shown in Table 12.1.

12.6.1 Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) is obtained by filtering the signal through a series of digital filters at different scales. The scaling operation is done by changing the resolution of the signal by the process of subsampling.

The DWT can be computed using either convolution-based or lifting-based procedures. In both methods, the input sequence is decomposed into low-pass and high-pass sub-bands, each consisting of half the number of samples in the original sequence.

Table 12.1 Comparison of CWT with DWT

	CWT	DWT
1. Scale	At any scale	Dyadic scales
2. Translation	At any point	Integer point
3. Wavelet	Any wavelet that satisfies minimum criteria	Orthogonal, biorthogonal, ...
4. Computation	Large	Small
5. Detection	Easily detects direction, orientation	Cannot detect minute object if not finely tuned
6. Application	Pattern recognition Feature extraction Detection	Compression De-noising Transmission Characterisation

12.7 MULTI-RESOLUTION ANALYSIS

Multi-resolution offers an efficient framework for extracting information from images at various levels of resolution. Pyramid algorithms use a multi-resolution architecture to reduce the computational complexity of image-processing operations. It is believed that the human visual system incorporates a hierarchical scheme for extracting details from natural scenes.

A multi-resolution analysis of $L^2(\mathbb{R})$ is a set of closed, nested subspaces $V_j; j \in \mathbb{Z}$ satisfying the following properties:

(i) $\overline{\bigcup_{j=-\infty}^{+\infty} V_j} = L^2$. The bar over the union indicates the closure in $L^2(\mathbb{R})$.

(ii) $\bigcap_{j=-\infty}^{\infty} V_j = \Phi$. The intersection is null set.

(iii) The space L^2 can be decomposed as nested subspaces V_j . This is represented as $\dots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \dots$. The nested subspace spanned by the scaling function is illustrated in Fig. 12.2.

(iv) If a function $f(x) \in V_j$, then $f(2x) \in V_{j+1}$

(v) If a function $f(x) \in V_j$, then its translation $f(x-k) \in V_j$

(vi) There exists a scaling function $\phi(x) \in V_0$ such that $\{\phi(x-k) \mid k \in \mathbb{Z}\}$ forms a Riesz basis of $L^2(\mathbb{R})$.

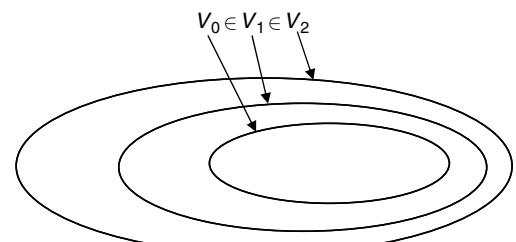


Fig. 12.2 Nested subspace spanned by the scaling functions

12.7.1 Scaling and Wavelet Function

Let the function $\phi(x) \in V_0$ such that the set of functions consisting of $\phi(x)$ and its integer translates $\{\phi(x-k) \mid k \in \mathbb{Z}\}$ form a basis for the space V_0 which is termed scaling function or father function.

The subspaces V_j are nested which implies $V_j \subset V_{j+1}$. It is possible to decompose V_{j+1} in terms of V_j and W_j . Here, W_j is the orthogonal complement of V_j in V_{j+1} . The following equations hold good:

$$V_j \oplus W_j = V_{j+1} \quad (12.11)$$

where the symbol \oplus stands for direct sum. This relationship is illustrated in Fig. 12.3.

From Fig. 12.3, it is obvious that W_j is the difference between V_{j+1} and V_j . Also it can be understood that

$$V_0 \oplus W_0 = V_1$$

$$V_0 \oplus W_0 \oplus W_1 = V_2$$

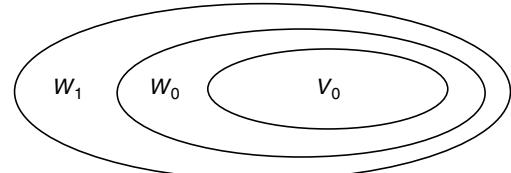


Fig. 12.3 Relationship of spaces spanned by the scaling and wavelet functions

Also, $W_j \perp V_j$. It is to be noted that the direct sum of the subspaces W_j is equal to L^2 which is represented by

$$\overline{\bigcup_{j=-\infty}^{+\infty} V_j} = \overline{\bigoplus_{j=-\infty}^{\infty} W_j} = L^2 \quad (12.12)$$

Equation (12.12) implies that V_j is a coarse-resolution representation of V_{j+1} , while W_j carries the ‘high-resolution’ difference information between V_{j+1} and V_j .

If $\psi(x) \in W_0$, it obeys the translation property such that $\psi(x-k) \in W_0$, $k \in \mathbb{Z}$ form a basis for the space W_0 which is termed as the wavelet function or mother function.

12.7.2 Relationship Between Wavelet and Filter

If the spaces V_0 and W_0 are subspaces of V_1 then the scaling function $\phi(x)$ and the wavelet function $\psi(x)$ can be expressed in terms of the basis functions of V_1 as

$$\Phi(x) = 2 \sum_k h_k \phi(2x - k) \quad (12.13)$$

$$\Psi(x) = 2 \sum_k g_k \phi(2x - k) \quad (12.14)$$

Equations (12.13) and (12.14) are known as a two-scale relationship. In Eqs. (12.13) and (12.14) h_k and g_k are the filter coefficients that uniquely define the scaling function $\Phi(x)$ and the wavelet function $\Psi(x)$.

12.8 EXAMPLES OF WAVELETS

This section discusses some of the most commonly used wavelets. Most of them are continuous wavelets. They are (i) Haar wavelet, (ii) Morlet wavelet, and (iii) Mexican-hat wavelet.

12.8.1 Haar Wavelet

The Haar wavelet was introduced by Haar in 1910. It is a bipolar step function. The expression of the Haar wavelet is given by

$$\psi(t) = \begin{cases} 1 & \text{when } 0 < t < \frac{1}{2} \\ -1 & \text{when } \frac{1}{2} < t < 1 \\ 0 & \text{otherwise} \end{cases} \quad (12.15)$$

The Haar wavelet is illustrated in Fig. 12.4.

From Fig. 12.4, it is obvious that the Haar wavelet is a real function, anti-symmetric with respect to $t = \frac{1}{2}$. The Haar wavelet is discontinuous in time. The Haar wavelet is localised in the time domain, but it has poor localisation in the frequency domain.

12.8.2 Daubechies Wavelet

The Daubechies wavelet bases are a family of orthonormal, compactly supported scaling and wavelet functions that have maximum regularity for a given length of the support of the quadrature mirror filters. Daubechies has shown that it is impossible to obtain an orthonormal and compactly supported wavelet that is either symmetric or antisymmetric except for Haar wavelets.

12.8.3 Morlet Wavelet

The Morlet wavelet is obtained by multiplying the Fourier basis with a Gaussian window. The expression of a Morlet wavelet is given as

$$\psi(t) = \exp(j\omega_0 t) \exp\left(-\frac{t^2}{2}\right) \quad (12.16)$$

On taking the Fourier transform, we get the spectrum of the Morlet wavelet which is given by

$$\psi(\omega) = \sqrt{\frac{\pi}{2}} \left\{ \exp\left[-\frac{(\omega - \omega_0)^2}{2}\right] + \exp\left[-\frac{(\omega + \omega_0)^2}{2}\right] \right\} \quad (12.17)$$

From the above expression, it is clear that the spectrum of the Morlet wavelet consists of two Gaussian functions shifted to ω_0 and $-\omega_0$.

12.8.4 Mexican-hat Wavelet

Mexican-hat wavelet is the second-order derivative of the Gaussian function. The expression of Mexican-hat wavelet is given as

$$\psi(t) = (1 - t^2) \exp\left(-\frac{t^2}{2}\right) \quad (12.18)$$

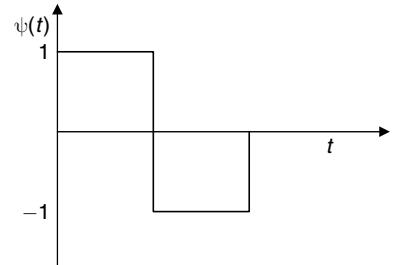


Fig. 12.4 Haar wavelet

On taking Fourier transform of the above equation, we will get the spectrum of the Mexican-hat wavelet which is given by

$$\psi(\omega) = -\omega^2 \exp\left(-\frac{\omega^2}{2}\right) \quad (12.19)$$

The two-dimensional Mexican-hat wavelet is popularly known as the Laplacian operator. The Laplacian operator is widely used in edge detection.

12.8.5 Shannon Wavelet

The expression of the Shannon wavelet is given as below:

$$\psi(t) = \frac{\sin(2\pi t) - \sin(\pi t)}{\pi t} \quad (12.20)$$

The Shannon wavelet has poor time resolution, but its frequency localisation is excellent.

12.9 WAVELET-BASED IMAGE COMPRESSION

The wavelet transform decomposes an image into a set of different resolution sub-images, corresponding to the various frequency bands. This results in a multi-resolution representation of images with localisation in both the spatial and frequency domains. This is desirable in the case of image compression, but it is not possible in the case of Fourier and cosine transforms which gives good localisation in one domain at the expense of the other.

The main advantages of wavelet-based image compression is summarised below:

- (i) Wavelets have non-uniform frequency spectra which facilitate multi-scale analysis.
- (ii) The multi-resolution property of the wavelet transform can be used to exploit the fact that the response of the human eye is different to high and low frequency components of an image.
- (iii) DWT can be applied to an entire image without imposing block structure as used by the DCT, thereby reducing blocking artifact.

DWT Implementation A Discrete Wavelet Transform (DWT) can be implemented through (i) filter bank scheme, or (ii) lifting scheme.

12.9.1 Sub-band Coding

Sub-band coding is a procedure in which the input signal is subdivided into several frequency bands. Sub-band coding can be implemented through a filter bank. A filter bank is a collection of filters having either a common input or common output. When the filters have a common input, they form an analysis bank and when they share a common output, they form a synthesis bank. The basic idea in a filter bank is to partition a signal dyadically at the frequency domain. First, let us analyse the perfect reconstruction criteria for a two-channel filter bank for a one-dimensional signal, and the same concept can be easily extended to a two-dimensional signal, if the two-dimensional signal is separable.

12.9.2 Two-channel Filter Bank

The complete two-channel filter bank is composed of two sections: (i) analysis section, and (ii) synthesis section, as shown in Fig. 12.5. The analysis section decomposes the signal into a set of sub-band components

and the synthesis section reconstructs the signal from its components. The sub-band analysis and synthesis filters should be designed to be alias-free and are also required to satisfy the perfect signal-reconstruction property. The simultaneous cancellation of aliasing as well as amplitude and phase distortions leads to perfect reconstruction filter banks which are suitable for hierarchical sub-band coding and multi-resolution signal decomposition.

The analysis filter bank splits the signal into two equal frequency bands. Here, the filters $H_0[z]$ and $H_1[z]$ act as low-pass and high-pass filters respectively. After filtering, the signal outputs at 1 and 2 are given in Eqs. (12.21) and (12.22) respectively.

$$\text{At } ① : - X[z] \cdot H_0[z] \quad (12.21)$$

$$\text{At } ② : - X[z] \cdot H_1[z] \quad (12.22)$$

After filtering, the signal's sampling frequency is too high, and hence half the samples are discarded by the down-sampling operation. After decimation, the Z transform is given in Eqs. (12.23) and (12.24) respectively.

$$\text{At } ③ : - Y[z] = \frac{1}{2} \left\{ X[z^{1/2}] \cdot H_0[z^{1/2}] + X[-z^{1/2}] \cdot H_0[-z^{1/2}] \right\} \quad (12.23)$$

$$\text{At } ④ : - Y[z] = \frac{1}{2} \left\{ X[z^{1/2}] \cdot H_1[z^{1/2}] + X[-z^{1/2}] \cdot H_1[-z^{1/2}] \right\} \quad (12.24)$$

The synthesis filter bank reconstructs the signal from the two filtered and decimated signals. The synthesis procedure involves expanding the signals in each branch by two which is termed expansion or interpolation. The interpolation is achieved by inserting zeros between successive samples. After interpolation, the Z transform of the signal at the nodes (5) and (6) are given in Eqs. (12.25) and (12.26) respectively.

$$\text{At } ⑤ : - X[z] = \frac{1}{2} \left\{ X[z] \cdot H_0[z] + X[-z] \cdot H_0[-z] \right\} \quad (12.25)$$

$$\text{At } ⑥ : - X[z] = \frac{1}{2} \left\{ X[z] \cdot H_1[z] + X[-z] \cdot H_1[-z] \right\} \quad (12.26)$$

The above Eqs. ⑤ and ⑥ can be written in matrix form as given below:

$$\frac{1}{2} \times \begin{bmatrix} H_0[z] & H_0[-z] \\ H_1[z] & H_1[-z] \end{bmatrix} \cdot \begin{bmatrix} X[z] \\ X[-z] \end{bmatrix} \quad (12.27)$$

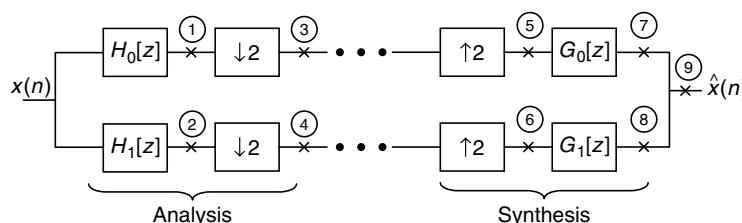


Fig. 12.5 A two-band analysis/synthesis filter bank system

At ⑦ and ⑧

$$\frac{1}{2} \times [G_0[z] \quad G_1[z]] \cdot \begin{bmatrix} H_0[z] & H_0[-z] \\ H_1[z] & H_1[-z] \end{bmatrix} \cdot \begin{bmatrix} X[z] \\ X[-z] \end{bmatrix} \quad (12.28)$$

$$\frac{1}{2} \times [G_0[z] \quad G_1[z]]_{(1 \times 2)} \cdot \begin{bmatrix} H_0[z] & H_0[-z] \\ H_1[z] & H_1[-z] \end{bmatrix}_{(2 \times 2)} \cdot \begin{bmatrix} X[z] \\ X[-z] \end{bmatrix}_{(2 \times 1)} \quad (12.29)$$

Combining both G and H matrices using matrix Multiplication, we get

$$\begin{bmatrix} \frac{G_0[z]H_0[z] + G_1[z]H_1[z]}{2} & \frac{G_0[z]H_0[-z] + G_1[z]H_1[-z]}{2} \end{bmatrix}_{(1 \times 2)} \cdot \begin{bmatrix} X[z] \\ X[-z] \end{bmatrix}_{(2 \times 1)} \quad (12.30)$$

$$F_0[z] = \frac{G_0[z]H_0[z] + G_1[z]H_1[z]}{2} \quad (12.31)$$

$$F_1[z] = \frac{G_0[z]H_0[-z] + G_1[z]H_1[-z]}{2} \quad (12.32)$$

We get,

$$[F_0[z] \quad F_1[z]]_{(1 \times 2)} \cdot \begin{bmatrix} X[z] \\ X[-z] \end{bmatrix}_{(2 \times 1)} \quad (12.33)$$

$$F_0[z]X[z] + F_1[z]X[-z] \quad (12.34)$$

In the above equation, $X[-z]$ refers to the aliasing component. This aliasing will spoil the signal. So select the filter co-efficient in order to reduce the aliasing effect, i.e., make the $F_1[z]$ as zero to neglect the aliasing effect.

Let,

$$H_0[z] = H[z]; \quad (12.35)$$

$$H_1[z] = H[-z];$$

$$G_0[z] = 2H[z];$$

$$G_1[z] = 2H[-z]$$

From the above conclusion, we can say that the four filter designs are given by a single filter co-efficient. This is the beauty of sub-band coding.

When we substitute the above assumptions,

$$F_1[z] = \frac{G_0[z]H_0[-z] + G_1[z]H_1[-z]}{2} \Rightarrow 0 \quad (12.36)$$

$$\begin{aligned} F_0[z] &= \frac{G_0[z]H_0[z] + G_1[z]H_1[z]}{2} \\ &= \frac{2 H[z] H[z] + (-2 H[-z]) H[-z]}{2} \\ &= H^2[z] + H^2[-z] \end{aligned} \quad (12.37)$$

So, finally at ⑨

$$(H^2[z] + H^2[-z]) \cdot X[z] \quad (12.38)$$

While transmitting from one place to another, the delay is unavoidable though the delay value may be in milli-seconds.

For a perfect reconstruction filter bank, the reconstructed signal is the delayed version of the original signal which is given by,

$$\begin{aligned} (H^2[z] + H^2[-z]) \cdot X[z] &= z^{-k} \cdot X[z] \\ (H^2[z] + H^2[-z]) &= z^{-k} \end{aligned} \quad (12.39)$$

That is, $H[z] = A[z] \cdot z^{-\left(\frac{N-1}{2}\right)}$

Then the signal value at ⑨ is given by

$$\begin{aligned} A^2[z] \cdot z^{-(N-1)} - A^2[-z] \cdot (-z)^{-(N-1)} &= z^{-k} \\ A^2[z] \cdot z^{-(N-1)} - A^2[-z] \cdot (-1)^{-(N-1)} \cdot (z)^{-(N-1)} &= z^{-k} \\ A^2[z] \cdot z^{-(N-1)} - A^2[-z] \cdot (z)^{-(N-1)} \cdot (-1)^{-(N-1)} &= z^{-k} \end{aligned}$$

If $k = N - 1$ (delay is governed by the filter co-efficient)

$$\begin{aligned} A^2[z] \cdot z^{-(N-1)} - A^2[-z] \cdot (z)^{-(N-1)} \cdot (-1)^{-(N-1)} &= z^{-(N-1)} \\ A^2[z] - \left(A^2[-z] \cdot (-1)^{-(N-1)} \right) &= 1 \end{aligned}$$

If N is even (for PR condition)

$$\begin{aligned} A^2[z] + A^2[-z] &= 1 \\ H^2[z] + H^2[-z] &= 1 \end{aligned}$$

The condition for perfect reconstruction is given by

$$H^2[z] + H^2[-z] = 1 \quad (12.40)$$

12.9.3 Sub-band Coding of 2D Signal

In the discrete wavelet transform, an image signal can be analysed by passing it through an analysis filter bank followed by decimation operation. The analysis filter bank consists of a low-pass and high-pass filter at each decomposition stage. When the signal passes through these filters, it splits into two bands. The low-pass filter, which corresponds to an averaging operation, extracts the coarse information of the signal. The high-pass filter, which corresponds to a differencing operation, extracts the detail information of the signal. The output of the filtering operation is then decimated by two. A two-dimensional transform is accomplished by performing two separate one-dimensional transforms. First, the image is filtered along the row and decimated by two. It is then followed by filtering the sub-image along the column and decimated by two. This operation splits the image into four bands, namely, LL, LH, HL and HH respectively as shown in Fig. 12.6.

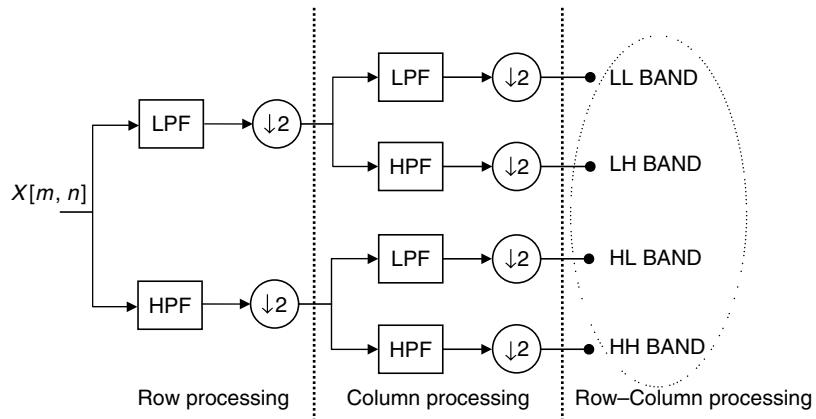


Fig. 12.6 Wavelet decomposition

Separable System (1st Level of Decomposition) The first level of decomposition of the cameraman image is illustrated in Fig. 12.7.

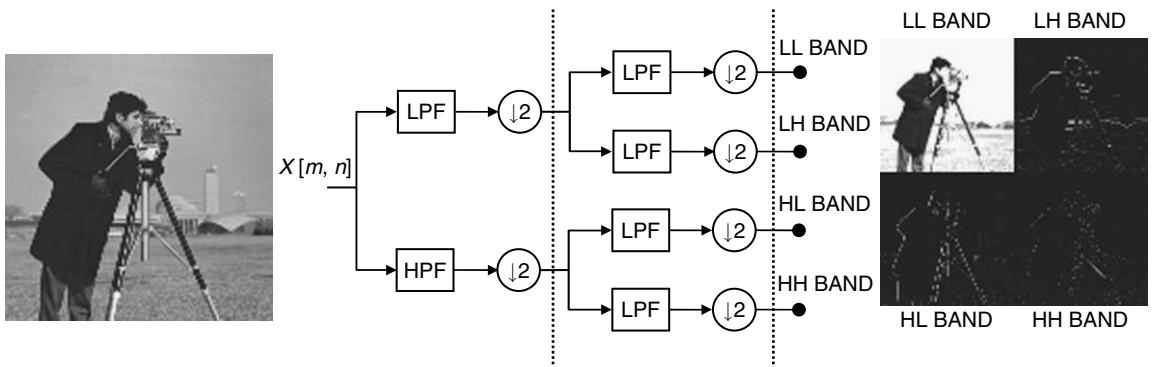


Fig. 12.7 First-level decomposition of cameraman image

Separable System (1st Level of Decomposition) Further decompositions can be achieved by acting upon the LL sub-band successively and the resultant image is split into multiple bands as shown in Fig. 12.8. The size of the input image and the size of the image at different levels of decompositions are illustrated in Fig. 12.8.

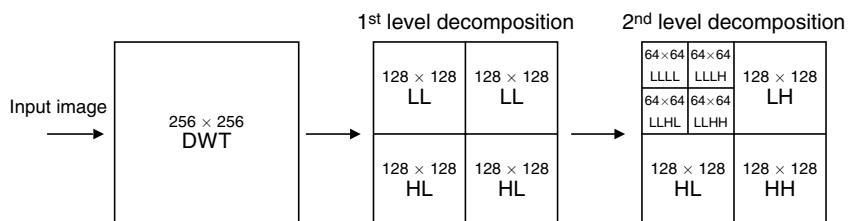


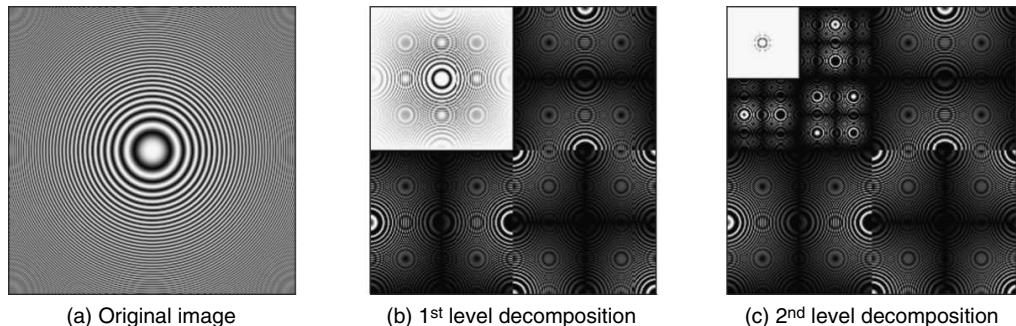
Fig. 12.8 Second-level decomposition of the input image

MATLAB Example 1 *Perform second-level decomposition of the input image using a Haar wavelet.*

Solution The MATLAB code which performs the decomposition of the zone-plate image is given in Fig. 12.9 and the corresponding output is shown in Fig. 12.10.

```
clc
clear all
close all
a=imread(zoneplate.png);
%First level decomposition
[p q r s]=dwt2(single(a),db1);
b=[uint8(p), q; r,s];
%Second level decomposition
[p1 q1 r1 s1]=dwt2(p,db1);
b1=[p1,q1; r1, s1];
b2=[uint8(b1), q; r s];
imshow(b2)
```

Fig. 12.9 MATLAB code to perform wavelet decomposition



(a) Original image

(b) 1st level decomposition

(c) 2nd level decomposition

Fig. 12.10 Result of the wavelet decomposition

12.9.4 Desirable Characteristics of a Filter Bank

The desirable characteristics of a filter bank include (i) maximal decimation, (ii) separable filtering, (iii) polyphase form, (iv) perfect reconstruction, and (v) tree structure.

(i) Maximal Decimation The maximal decimation property indicates that the number of coefficients produced by decomposition is the same as the number of input samples. This is also known as critical sampling. This property is associated with computational complexity as it keeps the number of samples to be processed to a minimum.

(ii) Separable Filtering Separable filtering indicates that two-dimensional as well as higher dimensional filtering can be performed as one-dimensional filtering. For example, two-dimensional filtering is performed as two one-dimensional filtering performed row-wise and column-wise. This property is associated with computational efficiency as separable filtering is more efficient than the equivalent non-separable filtering.

(iii) Polyphase Form Polyphase form is an efficient implementation of a decimated filter bank where the filtering is performed after the down-sampling, thus reducing the number of computations.

(iv) Perfect Reconstruction Perfect reconstruction property ensures that the reconstructed signal resembles the original signal without any error. That is, the coefficients produced by the forward transform can be sent through the inverse transform to reproduce the original signal without any error.

(v) Tree-Structure Decomposition Tree structure decomposition sub-divides radically the low-frequency region. Such decompositions are well-suited to processing of natural images which tend to have the energy concentrated in radically low-frequency regions.

12.9.5 Lifting Scheme

Sweden proposed the lifting scheme to compute the DWT. The lifting scheme is an efficient implementation of a wavelet-transform algorithm. The lifting scheme was developed as a method to improve wavelet transform. It was then extended to a generic method to create second-generation wavelets. The second-generation wavelets do not necessarily use the same function prototype at different levels. Second-generation wavelets are much more flexible and powerful than first-generation wavelets. The lifting procedure consists of three phases, namely, (i) split phase, (ii) predict phase, and (iii) update phase, as illustrated in Fig. 12.11

The first step in the lifting scheme is to separate the original sequence (X) into two sub-sequences containing the odd-indexed samples (X_0) and the even-indexed samples (X_e). This sub-sampling step is also called the *lazy wavelet transform* which is given by

$$X_0: d_i \leftarrow x_{2i+1} \quad (12.41)$$

$$X_e: s_i \leftarrow x_{2i} \quad (12.42)$$

Dual lifting (P) and primal lifting (U) are performed on the two sequences X_0 and X_e . The two sequences X_0 and X_e are highly correlated. Hence, a predictor $P()$ can be used to predict one set from the other. In this prediction step, which is also called dual lifting, the odd samples are predicted using the neighbouring

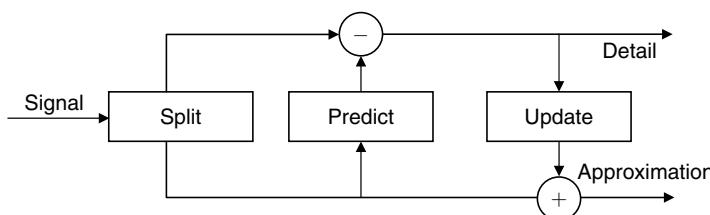


Fig. 12.11 Analysis stage in lifting scheme

even-indexed samples and the prediction error (detail) is recorded replacing the original sample value, thus providing in-place calculations.

$$d_i \leftarrow d_i - P(s_A) \quad (12.43)$$

where $A = (i - [N/2] + 1, \dots, i + [N/2])$, N is the number of dual vanishing moments in d . The number of vanishing moments sets the smoothness of the P function.

In the second lifting step, primal lifting (U), the even samples are replaced with smoothed values using the update operator $U()$ on previously computed details. The $U()$ operator is designed to maintain the correct running average of the original sequence, in order to avoid aliasing.

$$s_i \leftarrow s_i + U(d_B) \quad (12.44)$$

where $B = (i - \lfloor \tilde{N}/2 \rfloor, \dots, i + \lfloor \tilde{N}/2 \rfloor - 1)$. Here, \tilde{N} is the number of real vanishing moments. The $U()$ operator preserves the first \tilde{N} moments in the s sequence. The lazy wavelet is lifted to a transform with required properties by applying the dual and primal lifting pair of operations one or more times. Finally, the output streams are normalised using the normalising factor k .

$$d_i \leftarrow d_i - 1/k \quad (12.45)$$

$$s_i \leftarrow s_i \times k \quad (12.46)$$

The output from the s channel after the dual lifting step provides a low-pass filtered version of the input, whereas the output from the d channel after the dual lifting steps provide the high-pass filtered version of the input. The inverse transform is obtained by reversing the order and the sign of the operations performed in the forward transform.

12.10 EMBEDDED IMAGE CODING

The general principle of embedded image coding is that each successive bit of the bit stream that is received reduces the distortion of the reconstructed image by a certain amount. In order to achieve this, the information in the bit stream has to be organised in a decreasing order of importance. Embedded coding supports progressive image transmission by the property of generating the bits in the bit stream in their relative order of importance. A given image coded at a certain bit rate in an embedded fashion stores all the lower rate codes at the beginning of the bit stream. The encoding process can be stopped before or when the target bit rate is met. Similarly, the decoder can interrupt decoding at any point in the bit stream and can reconstruct any lower rate images. Thus progressive transmission is supported by transmitting a coarser version of the image first, followed progressively by the refinement details. The option of lossless reconstruction is still retained if the complete bit stream is generated, transmitted, and decoded. The embedded image coders are built upon three major components: (i) wavelet transform, (ii) quantisation scheme, and (iii) significance-map encoding.

12.11 EMBEDDED ZERO TREE WAVELET

EZW stands for Embedded Zero Tree Wavelet. An EZW encoder is an encoder specially designed to use with wavelet transforms. The EZW encoder is based on progressive coding to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail. Progressive encoding is also known as embedded coding. Both lossless and lossy coding of image is possible with the EZW encoder.

The EZW encoder is based on two important philosophies:

1. Natural images, in general, have a low-pass spectrum. When an image is wavelet transformed, the energy in the sub-bands decreases as the scale decreases. So wavelet coefficients will, on an average, be smaller in the higher sub-bands than in the lower sub-bands. This shows that progressive encoding is a very natural choice for compressing wavelet-transformed images, since higher sub-bands only add detail.
 2. Large wavelet coefficients are more important than small wavelet coefficients.

Based on the two observations, wavelet coefficients are coded in decreasing order, in several passes. For every pass, a threshold is chosen against which all coefficients are measured. If a wavelet coefficient is larger than the threshold, it is encoded and removed from the image. If it is smaller, it is left for the next pass. When all the wavelet coefficients have been visited, the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or the maximum bit-rate criterion has been satisfied. Thus, EZW applies Successive Approximation Quantisation (SAQ) in order to provide a multi-precision representation of the coefficients and to facilitate the embedded coding. The successive approximation quantisation uses a monotonically decreasing set of thresholds and indicates the significance of the wavelet coefficients with respect to any given threshold.

For each scan there are two passes—dominant pass, or significance map encoding; and refinement pass, or subordinate pass. Dominant pass generates any one of the four possible combinations like significant positive, significant negative, zero tree root and isolated zero, as shown in Fig. 12.12. Given a threshold T , if a given coefficient has a magnitude greater than T , it is called a significant coefficient at the level T . If the magnitude of the coefficient is less than T , and all its descendants have magnitudes less than T then the coefficient is called a zero tree root. If the coefficient value is less than T but some of its descendants have a value greater than T then such a coefficient is labeled as an isolated zero.

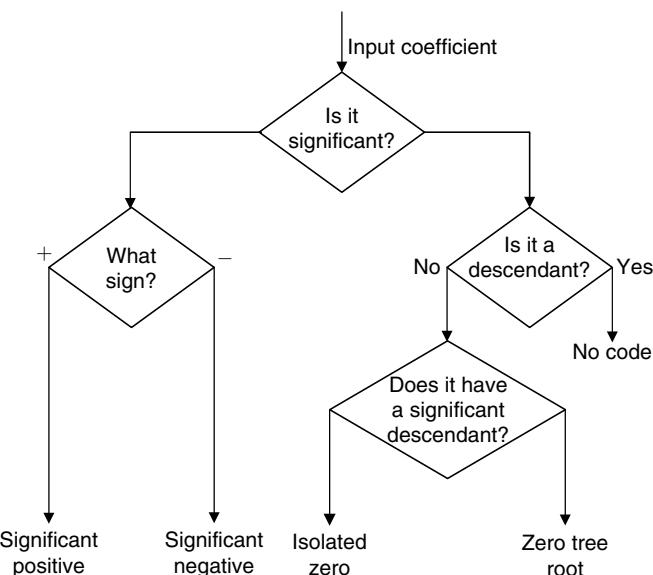


Fig. 12.12 Flow chart for the encoding of coefficients in significance map coding

12.11.1 EZW Algorithm

A simplified version of the EZW is given as follows:

1. Set the initial threshold T_0 such that $T_0 = 2^{\lfloor \log_2 C_{\max} \rfloor}$. Here, C_{\max} is the maximum coefficient value.
2. Set $k = 0$.
3. Conduct a dominant pass by scanning through the data. The output of the dominant pass is any one of the four combinations like Significant Positive(SP), Significant Negative (SN), Zero tree Root (ZR) and Isolated Zero(IZ).
 - (i) If the value of the coefficient is greater than the threshold and the value is positive, it means the output of the dominant pass is significant positive.
 - (ii) If the value of the coefficient is greater than the threshold and the value is negative, it means the output of the dominant pass is significant negative.
 - (iii) If the magnitude of the coefficient is less than the threshold and all its descendants have magnitudes less than the threshold then the coefficient is labeled as zero tree root.
 - (iv) If the coefficient magnitude is less than the threshold but some of its descendants have a value greater than the threshold then the coefficient is labeled as isolated zero.
4. All these four conditions are represented in the form of a flow chart in Fig. 12.12.
5. Conduct a subordinate pass or refinement pass by scanning through the data to refine the pixels already known to be significant in the current bit plane.
6. Set $k = k + 1$ and the threshold $T_k = \frac{T_{k-1}}{2}$.
7. Stop if the stopping criterion is met or go to Step 3.

12.11.2 Zero Tree

The crucial step in EZW is to exploit the dependency between wavelet coefficients across different scales to efficiently encode large parts of the image which are below the current threshold. When a wavelet transform of the image is performed, a coefficient in a low sub-band can be thought of having four descendants in the next higher sub-band. The four descendants each have four descendants in the next higher sub-band. As a result, every root has four leafs, and this is generally termed as quad-tree decomposition. A zero tree is a quad-tree of which all nodes are equal to or smaller than the root. The tree is coded with a single symbol and reconstructed by the decoder as a quad-tree filled with zeros. The EZW encoder exploits the zero tree based on the observation that wavelet coefficients decrease with scale.

Figure 12.13 provides a pictorial representation of the zero tree in the case of third -level wavelet decomposition. The coefficient at the coarser scale is called the *parent*, while all corresponding coefficients on the next finer scale of the same spatial location and similar orientation are called *children*. For a given parent, the set of all coefficients at all finer scales are called *descendants*.

The dominant-pass scanning order is zig-zag, right-to-left, and then top-to-bottom within each scale (resolution), before proceeding to the next higher scale. The EZW scanning order is illustrated in Fig. 12.14. On subsequent dominant passes, only those coefficients not yet found to be significant are scanned.

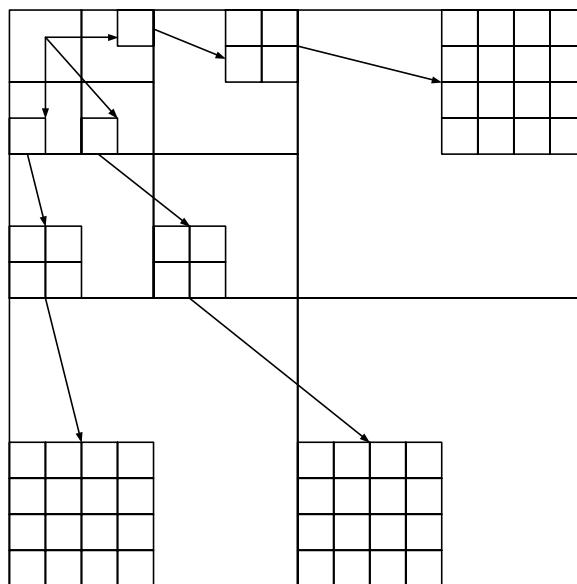


Fig. 12.13 Parent-child relationship in a zero tree

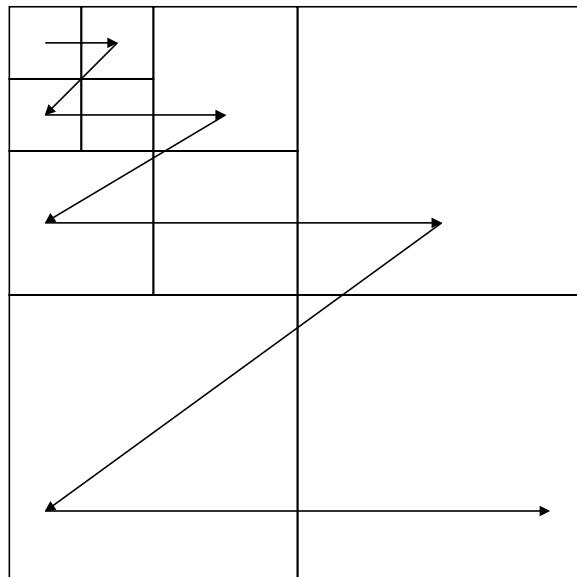


Fig. 12.14 EZW scanning order

Example 12.1: Encoding and decoding procedure using EZW

The wavelet coefficient of the image is shown in Fig. 12.15. Apply the encoding and decoding EZW procedure to this image.

Use the following codes shown in Table 12.2 given below during transmission.

Table 12.2 Codes to be followed

Zerotree root	zr	00
Significant positive	sp	11
Significant negative	sn	01
Isolated zero	iz	10

34	0	1	-1
0	0	-1	1
4	-4	10	-6
-4	4	6	-10

Fig. 12.15 Wavelet coefficient

Solution EZW is a multi-pass algorithm. Within each pass, we have both dominant and refinement pass. From the wavelet coefficient we can find the maximum value of the coefficient to be 34. This implies $C_{\max} = 34$. The steps involved in EZW encoding and decoding procedure is given below:

First pass: Determination of the threshold value T_0

The formula to compute the initial threshold T_0 is $T_0 = 2^{\lfloor \log_2 C_{\max} \rfloor}$

$$T_0 = 2^{\lfloor \log_2 C_{\max} \rfloor} = 2^{\lfloor \log_2 34 \rfloor} = 2^5 = 32$$

$$T_0 = 32$$

(a) Dominant pass The first coefficient 34 is compared with T_0 . 34 is greater than T_0 . Therefore, 34 is the significant positive (sp). So we send the code word [11]. Then, the next coefficient in the scan is 0, which is less than 32(T_0) and its descendants (1, -1, -1, and 1) are all less than 32(T_0). Therefore, 0 is a zero-tree root (zr) and we encode the entire set with the label zr, send the code word [00]. The next coefficient in the scan is 0, which is also less than 32(T_0) and its descendants (4, -4, -4 and 4) are all less than 32(T_0), which is also a zero-tree root (zr). Send once again the code word [00]. Similarly, the next coefficient in the scan is 0, which is less than 32(T_0) and its descendants (10, -6, 6 and -10) are all less than 32(T_0). So 0 is also a zero-tree root (zr), and once again, we send the code word [00].

So after the dominant pass, the transmitted code words are

[11 00 00 00]

At this level, we need 8 bits from our bit budget. The only one significant coefficient (sp) in this pass is the coefficient with a value of 34. This significant coefficient is included in the list L_s to be refined in the refinement pass. Calling the subordinate list L_s , we have

$$L_s = \{34\}$$

The reconstructed value of this coefficient is $\left(\frac{3}{2}\right)T_0 = \left(\frac{3}{2}\right) \times 32 = 48$,

and the reconstructed band after the first dominant pass is shown in Fig. 12.16.

48	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Fig. 12.16 Reconstructed coefficients after the first dominant pass

(b) Refinement pass The next step is the refinement pass, in which we obtain a correction term for the reconstruction value of the significant coefficients. In this case, the list L_s contains only one element. The difference between this element and its reconstructed value is

$$34 - 48 = -14$$

Quantising this with a two-level quantiser with reconstruction levels

$\pm \frac{T_0}{4} = \pm \frac{32}{4} = \pm 8$, we obtain a correction term of 8. Here, the difference between the L_s element and its reconstructed value is negative (-14). Thus, the reconstruction becomes $48 - 8 = 40$. Transmitting the correction term costs a single bit, i.e., the correction term is negative. So we send the one bit as [0], and add it to the transmitted codewords. Up to this level we have used 9 bits. After the first pass, using the 9 bits [11 00 00 00 0], the reconstructed values are shown in Fig. 12.17.

40	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Fig. 12.17 Reconstructed values after the first refinement pass

Second pass

After the first pass, we go for the second pass and here, the value of the threshold is reduced by a factor of 2 which is given below.

$$T_1 = \frac{T_0}{2} = \frac{32}{2} = 16$$

We re-scan the coefficients and leave the significant coefficients. That is, the significant coefficient of the first pass is 34. This value is replaced by * as illustrated in Fig. 12.18.

(a) Dominant pass In the re-scan process, the first coefficient encountered has a value of 0 which is less than $16(T_1)$ and its descendants (1, -1, -1, and 1) are all less than $16(T_1)$. Therefore, 0 is a zero-tree root (zr) and we encode the entire set with the label zr, and send the code word [00]. The next coefficient in the scan is 0, which is also less than $32(T_1)$, and its descendants (4, -4, -4 and 4) are all less than $16(T_1)$, which is also a zero-tree root (zr). Send once again the code word [00]. Similarly, for the next coefficient in the scan is 0, it is less than $16(T_1)$ and its descendants (10, -6, 6 and -10) are all less than $16(T_1)$. So, 0 is also a zero-tree root (zr), and once again we send the codeword [00].

So after the dominant pass, the transmitted code words are

[00 00 00]

In the second pass, there is no significant coefficient. So there is no need for reconstruction, and hence we proceed directly to the refinement process.

(b) Refinement pass The next step is the refinement pass, in which we obtain a correction term for the reconstruction value of the significant coefficients. In this case, the list L_s contains only one element. The difference between this element and its reconstructed value is

$$34 - 40 = -6$$

Quantising this with a two-level quantiser with reconstruction levels $\pm \frac{T_1}{4} = \pm \frac{16}{4} = \pm 4$, we obtain a correction term of 4. Here, the difference between the L_s element and its reconstructed value is negative (-6). Thus, the reconstruction becomes $40 - 4 = 36$. Transmitting the correction term costs a single bit, i.e., the correction

*	0	1	-1
0	0	-1	1
4	-4	10	-6
-4	4	6	-10

Fig. 12.18 Starting of second pass

term is negative. So we send the one bit as [0] and add in to the transmitted codewords. Up to this level we have used a 16 bits. After the second pass, using the 16 bits [11 00 00 0 00 00 00 0], the reconstructed values obtained are shown in Fig. 12.19.

Third pass

After the second pass, we go for the next pass. First, we reduce the value of the threshold by a factor of 2 and repeat the process.

$$T_2 = \frac{T_1}{2} = \frac{16}{2} = 8$$

We re-scan the coefficients, leave the significant coefficients, i.e., the significant coefficient of the second pass is 34. This value is replaced by*. The values at the beginning of the third pass are shown in Fig. 12.20.

(a) Dominant pass In the re-scan process, the first coefficient encountered has a value of 0 which is less than 8 (T_2) and its descendants (1, -1, -1, and 1) are all less than 8 (T_2). Therefore, 0 is a zero-tree root (zr). We encode the entire set with the label zr, and send the code word [00]. The next coefficient in the scan is 0, which is also less than 8 (T_2). Its descendants (4, -4, -4 and 4) are all less than 8 (T_2), which is also a zero tree root (zr). Send once again the code-word [00]. Similarly, the next coefficient in the scan is 0, which is less than 8 (T_2) and its descendants (10, and -10) are greater than 8 (T_2). So 0 is an isolated zero (iz). Send the code word [10]. Now, scan its descendant 10, which is greater than 8 (T_2), and so it is significant positive. Send the code word [11]. The other descendants are (-6 and 6) less than the threshold value 8 (T_2). So these two descendants are isolated zero, because there is no further child. Send the code words [10] and [10]. The last descendant of 0 is -10, which is greater than the threshold without sign. So it is significant but negative. It is called significant negative (sn) and we send the code word [01].

So after the dominant pass, the transmitted code words are [00 00 10 11 10 10 01]. At this level, we need 30 bits from our bit budget. In the third pass, there are two significant coefficients. Now we include these coefficients into L_s . we have

$$L_s = \{34, 10, -10\}$$

The reconstructed value of this coefficient is $\left(\frac{3}{2}\right)T_2 = \left(\frac{3}{2}\right) \times 8 = 12$, and the reconstructed band is shown in Fig. 12.21.

(b) Refinement pass The next step is refinement pass, in which we obtain a correction term for the reconstruction value of the significant coefficients. In this case, the list L_s contains three elements. The difference between these elements and the reconstructed value is

$$34 - 36 = -2$$

$$10 - 12 = -2$$

$$-10 + 12 = 2$$

Quantising this with a two-level quantiser with reconstruction level $\pm \frac{T_2}{4} = \pm \frac{8}{4} = \pm 2$, we obtain a correction term of 2. Here, the difference between the L_s elements and its reconstructed values are (-2, -2 and 2). In this

36	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Fig. 12.19 End of second pass

*	0	1	-1
0	0	-1	1
4	-4	10	-6
-4	4	6	-10

Fig. 12.20 Beginning of third pass

36	0	0	0
0	0	0	0
0	0	12	0
0	0	0	-12

Fig. 12.21 Reconstructed values after third dominant pass

pass, we have two negative values and one positive value. Thus, the reconstruction becomes $36 - 2 = 34$, $12 - 2 = 10$ and $-12 + 2 = -10$. Transmitting the correction term costs three bits, which are [0 0 1]. After the third pass, the used code words are [00 00 10 11 10 10 01 0 0 1]. Up to this level we have used 33 bits. The reconstructed values obtained after the third refinement pass are shown in Fig. 12.22.

Fourth pass

The value of the threshold is reduced by a factor of 2 and the process is then repeated.

$$T_3 = \frac{T_2}{2} = \frac{8}{2} = 4$$

We re-scan the coefficients, leaving the significant coefficients, i.e., those significant coefficients up to the third pass are $\{34, 10, -10\}$. This value is replaced by *. The values considered at the beginning of the fourth pass are shown in Fig. 12.23.

(a) Dominant pass In the re-scan process, the first coefficient encountered has a value of 0, which is less than 4 (T_3) and its descendants (1, -1, -1, and 1) are all less than 4 (T_3). Therefore, 0 is a zero-tree root (zr). We encode the entire set with the label zr, and send the code word [00]. The next coefficient in the scan is 0, which is also less than 4 (T_3) and its descendants (4, -4, -4 and 4) are all equal to 4 (T_3). But we consider it as less than the threshold value, which is also zero-tree root (zr), and send once again the code word [00]. Similarly, the next coefficient in the scan is 0, which is less than 4 (T_3) and its descendants (-6, and 6) are greater than 4 (T_3). So, 0 is also an isolated zero (zr), and we send the code word [10]. Now, its descendant -6 is greater than 4 (T_3) without sign, so it is significant negative (sn). Send the code word [01]. The last descendant of 0 is 6. It is greater than the threshold value, and so it is called significant positive (sp). We send the code word [11].

So after the dominant pass, the transmitted code words are [00 00 10 01 11]. At this level, we need 43 bits from our bit budget. In the third pass, there are two significant coefficients. Now we include these coefficients into L_s . Thus we have

$$L_s = \{34, 10, -10, -6, 6\}$$

The reconstructed value of this coefficient is $\left(\frac{3}{2}\right)T_3 = \left(\frac{3}{2}\right) \times 4 = 6$, and the reconstructed band is shown in Fig. 12.24.

(b) Refinement pass The next step is the refinement pass, in which we obtain a correction term for the reconstruction value of the significant coefficients. In this case, the list L_s contains five elements. The difference between these elements and the reconstructed value is

$$34 - 34 = 0$$

$$10 - 10 = 0$$

$$-10 + 10 = 0$$

$$-6 + 6 = 0$$

$$6 - 6 = 0$$

34	0	0	0
0	0	0	0
0	0	10	0
0	0	0	-10

Fig. 12.22 Reconstructed values after the third refinement pass

*	0	1	-1
0	0	-1	1
4	-4	*	-6
-4	4	6	*

Fig. 12.23 Values at the beginning of the fourth pass

34	0	0	0
0	0	0	0
0	0	10	-6
0	0	6	-10

Fig. 12.24 Reconstructed values at the end of the fourth dominant pass

Quantising this with a two-level quantiser with reconstruction level $\pm \frac{T_3}{4} = \frac{4}{4} = 1$, we obtain a correction term of 1. Here, the difference between the L_s elements and the reconstructed values are (0, 0, 0, 0, and 0). We consider the difference value 0 to be negative. In this pass, we have five negative values. Thus, the reconstruction becomes $34 - 1 = 33$; $10 - 1 = 9$; $-10 - 1 = -11$; $-6 - 1 = -7$ and $6 - 1 = 5$. Transmitting the correction term costs five bits. So we send the five bits [0 0 0 0 0] added in to the transmitted code words. After the fourth pass, the code words used are [00 00 10 01 11 0 0 0 0 0]. Up to this level, we have used 48 bits. But the allotting in our case is 45. So, we leave 3 bits at the end of the transmitted bit streams. Therefore, the final transmitted codewords are **[11 00 00 00 0 00 00 00 00 00 00 10 11 10 10 01 0 0 1 00 00 10 01 11 0 0]**.

After the transmitted bit streams of size 45, we obtain the reconstruction value as shown in Fig. 12.25.

33	0	0	0
0	0	0	0
0	0	9	-6
0	0	6	-11

Fig. 12.25 Final reconstructed values

12.12 SET PARTITIONING IN HIERARCHICAL TREES (SPIHT)

The set-partitioning algorithm uses the principles of self-similarity across scales as in EZW and partial ordering by magnitude of wavelet coefficients. Set partitioning into hierarchical trees, that is, sorting of the trees, is based on their significance at every applied threshold and ordered bit-plane transmission of the refinement bits it means that the magnitude of each significant coefficient is progressively refined. The essential difference of the SPIHT coding process with respect to EZW is the way trees of coefficients are partitioned and sorted.

SPIHT introduces three lists of wavelet coefficients:

- (i) List of insignificant pixels (LIP)
- (ii) List of significant pixels (LSP)
- (iii) List of insignificant sets (LIS)

LIS is further broken down into two types of sets of insignificant pixels. They are type A (all descendants are zero) and type B (all grandchildren and further descendants are zero).

The SPIHT algorithm defines four types of sets, which are sets of coordinates of coefficients:

- (i) $\bar{O}(i, j)$ is the set of coordinates of the offsprings of the wavelet coefficient at the location (i, j) .
- (ii) $D(i, j)$ is the set of all descendants of the coefficients at location (i, j) .
- (iii) H is the set of all root nodes.
- (iv) $L(i, j)$ is the set of coordinates of all the descendants of the coefficient at the location (i, j) .

Example 12.2: Encoding and Decoding using SPIHT algorithm

The wavelet coefficients of the given image are shown in Fig. 12.26.

Encode the coefficients using the SPIHT algorithm.

34	0	1	-1
0	0	-1	1
4	-4	10	-6
-4	4	6	-10

Fig. 12.26 Wavelet coefficients of the image

Solution To find the number of passes in the encoding process

Let us denote the number of passes by the letter n . The value of n is given by

$$n = \left\lceil \log_2^{C_{\max}} \right\rceil$$

where C_{\max} is the maximum value of the coefficient.

$$n = \left\lceil \log_2^{34} \right\rceil = 5$$

In the SPIHT algorithm, three sets of lists are used. They are

- (i) LIP (List of Insignificant Pixel)
- (ii) LIS (List of Insignificant Set)
- (iii) LSP (List of Significant Pixel)

Initially, the LIP contains four insignificant pixels: 34, 0, 0 and 0. Similarly, the LIS contains three sets of values (**01**, **10** and **11**). Here, **01** denotes four values (1, -1, -1 and 1), **10** denotes the four values 4, -4, -4 and 4, and finally **11** contains the four values 10, -6, 6 and -10. Initially, the LSP is empty.

First pass In the first pass, the value of $n = 5$, and the lists are

$$\begin{aligned} \text{LIP} &= \{34, 0, 0, 0\} \\ \text{LIS} &= \{01, 10, 11\} \\ \text{LSP} &= \{ \} \end{aligned}$$

From the value of n , we should find the initial threshold value T_0 .

$$T_0 = 2^n = 2^5 = 32$$

Now compare the values of LIP with the threshold T_0 . If the LIP values are greater than T_0 then those values are moved into LSP; otherwise there is no change in the values of LIP. Here, the first value of LIP is 34, which is greater than T_0 . So the value of 34 is moved into LSP. We send the code 10 to the encoding bit stream. The remaining values of LIP are lesser than T_0 . For each value of LIP, we send one bit like 0. For this case, the lesser values are three. So send the code 000 to the encoding bit stream.

After the comparison of the LIP with the initial threshold, the encoded bit stream is {10000}. Now check the LIS, if it has any value greater than T_0 . Then send one alert bit like (1) to the encoded bit stream. Otherwise send the one (0) bit to each set. In our case, the set **01**, **10** and **11** have the values lesser than T_0 . Send the bits (000) to the encoded bit stream. After the first pass, the values in the three lists are as follows:

$$\begin{aligned} \text{LIP} &= \{0, 0, 0\} \\ \text{LIS} &= \{01, 10, 11\} \\ \text{LSP} &= \{34\} \end{aligned}$$

Second pass For the second pass, the value of n is decremented by 1. Then the corresponding threshold value reduces to 16 (i.e., $2^4 = 16$). The process is repeated by examining the contents of LIP. There are three elements in LIP. Each is insignificant at this threshold, and so we transmit three 0s (000). The next step is to examine the contents of LIS. The first element of LIS is the set containing the descendants of the coefficient at location **(01)**. Of this set, the values (1, -1, -1 and 1) are insignificant and so transmit 0. Looking at the

other elements of LIS (**10** and **11**), we can clearly see that both of these are insignificant at this level; hence send a 0 for each.

Refinement pass In the refinement pass, we examine the contents of LSP from the previous pass. There is only one element in there that is not from the current sorting pass, and it has a value of 34. The fourth MSB of 34 (100010) is 0; therefore, we transmit a 0 and complete this pass.

In the second pass, we have transmitted 8 bits: 10000000. After the second pass, the values of the lists are as follows:

$$\begin{aligned} \text{LIP} &= \{0, 0, 0\} \\ \text{LIS} &= \{01, 10, 11\} \\ \text{LSP} &= \{34\} \end{aligned}$$

We can notice that there is no change from the first pass.

Third pass The third pass proceeds with $n = 3$. With $n = 3$, the threshold is 8, (i.e. $2^3 = 8$). Again, we pass by examining the contents of LIP. There are three elements in LIP. Each is insignificant at this threshold so we transmit three 0s (000). The next step is of examine the contents of LIS. The first element of LIS is the set containing the descendants of the coefficient at location (**01**). Of this set, the values are (1, -1, -1 and 1) insignificant so transmit one 0. Looking at the other elements of LIS (**10** and **11**), we can clearly see that 10 is insignificant at this level; therefore, we send a 0. The next set **11**, both 10 and -10 are significant at this value of threshold. In other words, the set 11 is significant. We signal this by sending a 1 for the alert bit. The **11** descendants have a value of 10, which is greater than 8. Hence, it is significant positive, and so we send a 1 followed by a 0 (10).

The next two offsprings are both insignificant at this level. Therefore, we move these to LIP and transmit a 0 for each.

The fourth offspring, which has a value of -10, is significant but negative. So we send 1 followed by a 1 (11). We move the coordinates of two (10 and -10) to the LSP.

Refinement pass In the refinement pass, we examine the contents of LSP from the previous pass. There is only one element in there that is not from the current sorting pass, and it has a value of 34. The third MSB of 34 (100010) is 0. Therefore, we transmit a 0 and complete this pass.

In the third pass we have transmitted 13 bits: 0000011000110. After the third pass the values of lists as follows:

$$\begin{aligned} \text{LIP} &= \{0, 0, 0, -6, 6\} \\ \text{LIS} &= \{01, 10\} \\ \text{LSP} &= \{34, 10, -10\} \end{aligned}$$

Fourth pass The fourth pass proceeds with $n = 2$. As the threshold is now smaller (4), there are more coefficients that are deemed significant, and we end up sending 28 bits. Again, we pass by examining the contents of LIP. There are five elements in LIP. The first three values are insignificant at this threshold, and so we transmit three 0s (000). The next two values are significant. But one value (-6) is significant negative, so we send 11. The other coefficient (6) is significant positive, so we transmit 10 bits. These two values are moved to LSP. The next step is to examine the contents of LIS. The first element of LIS is the set containing the descendants of the coefficient at location (**01**). Of this set, the values (1, -1, -1 and 1) are insignificant, so we transmit one 0. Looking at the other element of LIS (**10**), we can clearly see that 4 is significant at this level (consider equal to

as greater). In this set all the values (4, -4, -4 and 4) are significant at this value of threshold; in other words, the set **10** is significant. We signal this by sending a 1 for the alert bit. The **10** descendants have a value of 4, which is greater than 4, so it is significant positive. Hence, we send a 1 followed by a 0 (10). The **10** descendants have a second value is -4, which is greater than 4 but negative, so it is significant negative. We send a 1 followed by a 1 (11). The **10** descendants have a third value is of -4, which is greater than 4 but it is also negative. So it is significant negative, and hence we send a 1 followed by a 1 (11). Finally, the last coefficient of **10** is 4, which is greater than the threshold. So it is significant positive, and we transmit 10 bits. The **10** part coefficients are moved into LSP.

Now check the number of bits encoded up to this level. In the fourth pass, the encoded bit streams are (0001110011011110), and the number of bits is 17. Up to this level, the total number of bits is 45 which meet the bit-budget. Hence, stop here.

After the fourth pass the lists are changed as follows:

$$\begin{aligned} \text{LIP} &= \{0, 0, 0\} \\ \text{LIS} &= \{01\} \\ \text{LSP} &= \{34, 10, -10, -6, 6, 4, -4, -4, 4\} \end{aligned}$$

The final encoded bit stream is

$$\{10000000000000000000000000110001100001110011011110\}$$

Decoding process

The decoding process begins with the transmitted bit stream. The final encoded bit stream is
 $\{10000000000000000000000000110001100001110011011110\}$

Step 1 Initially the three lists are as follows:

$$\begin{aligned} \text{LIP} &= \{(0, 0), (0, 1), (1, 0), (1, 1)\} \\ \text{LIS} &= \{01, 10, 11\} \\ \text{LSP} &= \{ \} \end{aligned}$$

We assume that the initial value of n is transmitted to the decoder, i.e., 5. This n is used to set the threshold value which is 32 for $n = 5$.

Step 2 Receive the bit stream of the first pass (10000000). Take the first bit of the bit stream. If it is one then combine the next bit to the first one, i.e., 10. This indicates the first element of LIP is significant positive. Then read the remaining bits of the bit stream. The next three bits are zero. Clearly, we can conclude that the remaining coefficients and reconstructed values of the LIP are zero. The first element of the LIP value is reconstructed by $(3/2) \cdot 2^n = (3/2) \cdot 2^5 = 48$.

Step 3 Read the next three bits of the bit stream. If there is no one, then we can conclude that all the coefficients and reconstructed values are zero.

After the first-pass encoded bit stream, the reconstructed values are shown in Fig. 12.27, and the three lists are as follows:

$$\begin{aligned} \text{LIP} &= \{(0, 1), (1, 0), (1, 1)\} \\ \text{LIS} &= \{01, 10, 11\} \\ \text{LSP} &= \{(0, 0)\} \end{aligned}$$

48	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Fig. 12.27 Reconstructed values after the first pass

Second pass

Step 1 Reduce the value of ‘ n ’ by one and examine the transmitted bit stream: 0000000. Here, the threshold value is $2^4 = 16$.

Step 2 Read the first three bits of the second-pass encoded bit stream. Here, all three bits are zero. All the entries in LIP are still insignificant.

Step 3 Read the next three bits of the encoded bit stream. These bits all are zero. These bits give us information about the sets in LIS. The fourth bit of the second-pass encoded bit stream is 0. It indicates that the **01** descendant values are all insignificant. Similarly, the next two bits are zero, and it indicates that the remaining sets of **10** and **11** descendant values all are insignificant.

Step 4 The last bit of the second-pass encoded bit stream is 0. It gives information about the refinement pass.

Refinement pass The final bit of the encoded bit-stream is 0. So we update the reconstruction of the (0,0) coefficient to $48 - (16/2) = 48 - 8 = 40$. The reconstruction at this stage is given in Fig. 12.28.

The entries in the lists are as follows:

$$LIP = \{(0, 1), (1, 0), (1, 1)\}$$

$$LIS = \{01, 10, 11\}$$

$$LSP = \{(0, 0)\}$$

40	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Fig. 12.28 Reconstructed values after second pass

Third pass

Step 1 Now reduce the value of ‘ n ’ by one. We get $n = 4 - 1 = 3$, and the threshold value at this level is 8. The encoded bit stream in the third pass is 0000011000110.

Step 2 Read the first three bits of the second-pass encoded bit stream. Here, all three bits are zero. All the entries in LIP are still insignificant.

Step 3 Read the next three bits of the encoded bit stream. These bits give us information about the sets in LIS. The fourth bit of the second-pass encoded bit stream is 0. It indicates that the **01** descendant values are all insignificant. Similarly, of the next bit is also zero; it indicates the set of **10** descendant values all are insignificant.

Step 4 The sixth bit of the third-pass encoded bit stream is 1. It indicates that any one of the set of 11 descendants are significant. This bit is called the alert bit.

Step 5 Read the next six bits of the bit stream 100011. This sequence indicates that the first offspring is significant positive, the next two offsprings are insignificant and the last offspring is significant but negative. Therefore, the first and last offsprings are moved to LSP, and the second and third offsprings are moved to LIP. We can also approximate these two significant coefficients in our reconstruction by $(3/2)*2^n = (3/2)*2^3 = 12$. Here, the first coefficient is significant positive, and the reconstructed value is 12. The last coefficient is significant negative; the reconstructed value is -12. The second and third offsprings are still insignificant.

Step 6 The last bit of the third-pass bit stream is 0. It indicates the refinement of the (0,0) coefficient.

Refinement pass The last bit of the third-pass bit stream is 0. It indicates the refinement of the (0,0) coefficient. So we update the reconstruction of the (0,0) coefficient to $40 - 8/2 = 40 - 4 = 36$. The reconstructed values after the third pass are shown in Fig. 12.29.

The entries in the lists are as follows:

$$LIP = \{(0, 1), (1, 0), (1, 1), (2, 3), (3, 2)\}$$

$$LIS = \{01, 10\}$$

$$LSP = \{(0, 0), (2, 2), (3, 3)\}$$

36	0	0	0
0	0	0	0
0	0	12	0
0	0	0	-12

Fig. 12.29 Reconstructed values after the third pass

Fourth pass

Step 1 Now reducing the value of 'n' by one, we get $n = 3 - 1 = 2$, and threshold value at this level is 4. The third-pass encoded bit stream is 0001110011011110.

Step 2 Read the first three bits of the third pass encoded bit stream. Here all the three bits are zero. The first three entries in LIP are still insignificant. Read the next four bits of the encoded bit stream (1110). In this sequence, the first two bits are '11.' It indicates the fourth entry in LIP is significant but negative. The next two bits are 10, and it indicates the fifth entry of LIP is significant positive. The reconstructed value of these two significant values are $(3/2)*2^n = (3/2)*2^2 = 6$. Here, the first one is significant negative; the reconstructed value is -6, and the second one is significant positive. Therefore, the reconstructed value is 6.

Step 3 Read the next bits of the encoded bit stream (0110111110). These bits give us information about the sets in LIS. Here, the first bit is 0. It indicates the **01** descendant values are all insignificant. The next bit is 1, and it indicates any one of the value of the set **10** of descendants is significant. This bit is called the alert bit.

Step 4 Read the bits after the alert bit of the bit stream, which is 10111110. This sequence indicates that the first offspring of **10** is significant positive; the next two offsprings are significant but negative. The last offspring of the **10** is significant positive. Therefore, all the offsprings are moved to LSP. We can also approximate these significant coefficients in our reconstruction by $(3/2)*2^n = (3/2)*2^2 = 6$. Here, the first and last coefficients are significant positive, and the reconstructed value is 6. The second and third coefficients are significant negative; and the reconstructed value is -6.

We can find that there are no bits in the encoded bit stream. Hence, we stop here.

Step 5 The reconstructed values after the fourth pass are shown in Fig. 12.30.

And the lists are as follows:

$$LIP = \{(0, 1), (1, 0), (1, 1)\}$$

$$LIS = \{01\}$$

$$LSP = \{(0, 0), (2, 2), (3, 3), (2, 3), (3, 2), (2, 0), (2, 1), (3, 0), (3, 1)\}$$

36	0	0	0
0	0	0	0
6	-6	12	-6
-6	6	6	-12

Fig. 12.30 Reconstructed values after the fourth pass

12.13 JPEG2000 COMPRESSION STANDARD

JPEG2000 is the new international standard for still image compression. The JPEG2000 standard is based on wavelet /sub-band coding techniques and supports lossy and lossless compression of grayscale and colour images. In order to facilitate both lossy and lossless coding in an efficient manner, reversible integer-

to-integer and non-reversible real-to-real transforms are employed. To code transform data, the coder makes use of bit-plane coding techniques. For entropy coding, a context-based adaptive binary arithmetic coder is used.

12.13.1 Need for JPEG2000 Standard

With the increasing use of multimedia technologies, image compression requires higher performance as well as new features. The JPEG2000 standard was developed to address this need. The JPEG2000 is a wavelet-based image-compression standard whereas the JPEG standard uses discrete cosine transform to decorrelate the relationship between the pixels. Some of the features which influenced the need for JPEG2000 standard are summarised below:

(i) Low Bit rate Compression The JPEG standard offers excellent rate-distortion performance in the mid and high bit rates. However, at low bit rates, the distortion is visible and is unacceptable. Hence, efficient compression standard is required for low bit-rate image compression.

(ii) Lossless and Lossy Compression The JPEG standard is capable of performing lossless and lossy compression but not on a single code stream. There is a need for a standard in which both lossless and lossy compression can be performed in a single code stream.

(iii) Single Decompression Architecture The current JPEG standard has 44 modes, many of which are application specific and not used by the majority of JPEG decoders. Greater interchange between applications can be achieved if a single common decompression architecture encompasses these features.

(iv) Transmission of Images in Noisy Environment The JPEG standard has provision for restart intervals, but image quality suffers dramatically when bit errors are encountered. One of the required features for JPEG2000 is improved error resilience. This feature becomes crucial as wireless communication becomes more prominent while wireless channels remain highly error-prone.

(v) Region-based Coding Region-based coding is a very promising technique that is gaining considerable interest. In region-based coding, the compression procedure starts with segmentation of the image into a set of regions, most often corresponding to objects that can be identified in the visual scene. The JPEG standard does not support region-based coding, whereas the JPEG2000 standard supports region-based coding.

12.13.2 Features of JPEG2000 Standard

In addition to compression, the features provided in JPEG2000 compression standard include

1. Progressive recovery of an image by fidelity or resolution
2. Region-of-interest coding, in which different parts of an image can be coded with different fidelity
3. Random access to particular regions of an image without needing to decode the entire code stream
4. A flexible file format with provisions for specifying opacity information and image sequences
5. Good error resilience

12.13.3 Encoding Process

The main blocks involved in the encoding process are depicted in Fig. 12.31

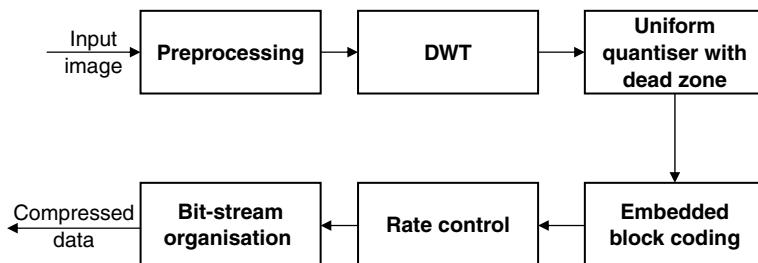


Fig. 12.31 Encoding process

(i) Preprocessing Step The different steps involved in the preprocessing (i) tiling, (ii) level offset, and (iii) irreversible colour transform. Tiling refers to partitioning of the input image into rectangular and non-overlapping tiles of equal size. Each tile is compressed independently using its own set of specified compression parameters. The level offset pre-processing stage ensures that the normal dynamic range is centred about zero. If the original B-bit image sample values are unsigned quantities, an offset of -2^{B-1} is added so that the samples have a signed representation in the range $-2^{B-1} \leq f(m, n) < 2^{B-1}$. If the data is already signed, no offset is needed. In JPEG2000, an irreversible colour transform is performed to convert RGB data into $Y C_r C_b$ data according to the formula

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.586 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(ii) Discrete Wavelet Transform JPEG2000 uses Discrete Wavelet Transform and Embedded Block Coding with Optimal Truncation (EBCOT) originally proposed by Taubman. The discrete wavelet transform can be implemented either through a filter bank or through the lifting scheme. There are two kinds of wavelet transform—reversible 5/3 integer wavelet transform and irreversible 9/7 floating-point wavelet transform. DWT is an important process to generate resolution-scalable bit streams, and decompose the spatial correlation. The output of a DWT is a set of transformed coefficients.

(iii) Quantisation Quantisation is basically approximation. The wavelet coefficients in each sub-band are scalar-quantised. The quantisation step size is based on the dynamic range of the sub-band values and constants defined in the standard. The purpose of quantisation is to reduce in precision of sub-band coefficients so that fewer bits will be needed to encode the transformed coefficients.

(iv) EBCOT EBCOT stands for Embedded Block Coding with Optimised Truncation. The EBCOT idea was proposed by David Taubman. The EBCOT algorithm uses wavelet transform to generate the sub-band. The sub-band generated by the wavelet transform is partitioned into small blocks of samples called code blocks.

Advantages of Block Coding

The advantages of block coding are the following:

- (i) Block coding exploits local variations in the statistics of the image from block to block.
- (ii) Block coding provides support for applications requiring random access to the image.
- (iii) Block coding reduces memory consumption in hardware implementation.
- (iv) Block coding allows for parallel implementation which can improve the speed of compression and decompression of blocks.

EBCOT generates a separate highly scalable bit stream for each code block. Also, the bit stream generated with the code block may be independently truncated to different lengths. EBCOT is a two-tiered coder, where Tier-1 is a context-based adaptive binary arithmetic coder and Tier-2 is for rate-distortion optimisation and bit stream layer formation. The two-tier concept is given in Fig. 12.32.

Tier-1 Coding In Tier-1, wavelet sub-bands are partitioned into small code blocks which in turn are coded by bit planes. The bit-plane structure is illustrated in Fig. 12.33. The most significant bits are coded first, and then lower-order bits are coded in descending order. Each bit plane is coded separately as if it was a bi-level image.

Each bit plane is coded in three passes; (i) significant propagation pass, (ii) magnitude refinement pass and (iii) clean-up pass.

The schematic representation of Tier-1 EBCOT is illustrated in Fig. 12.34.

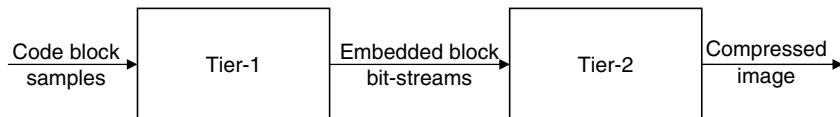


Fig. 12.32 Two tier EBCOT coder

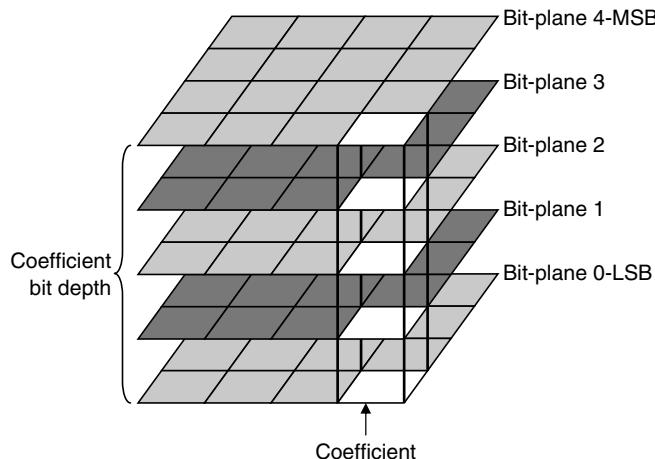


Fig. 12.33 Bit-plane representation

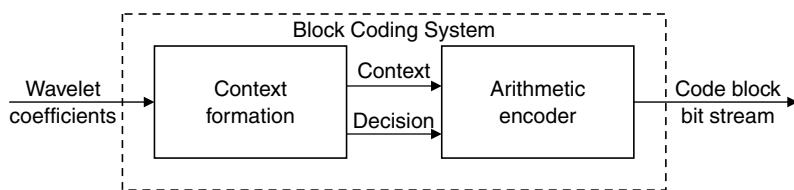


Fig. 12.34 Schematic representation of EBCOT Tier-1

(a) Significant Propagation Pass During significant propagation pass, a bit is coded if its location is not significant, but at least one of its eight-connected neighbours is significant. The meaning of the term significant is that the bit is the most significant bit of the corresponding sample in the code block.

(b) Magnitude Refinement Pass All the bits that have not been coded in the significant propagation pass and became significant in a previous bit plane are coded in this pass.

(c) Clean-up Pass All the bits that have not been coded in either significant propagation pass or magnitude refinement pass are coded in this pass. The clean-up pass also performs a form of run-length coding to efficiently code a string of zeros.

In each pass, only a part of the bit plane is coded and each bit position is coded only once by one of the three passes. Each coding pass constitutes an atomic code unit, called *chunk*. Code-word chunks are grouped into quality layers and can be transmitted in any order, provided the chunks that belongs to the same code block are transmitted in their relative order. Chunks constitute valid truncation points. Both the encoder and the decoder can truncate the bit stream in correspondence of a valid truncation point in order to recover the original data up to a target quality.

The scan pattern for samples in a code block is illustrated in Fig. 12.35. A code block is partitioned into stripes with a nominal height of four samples. The stripes are then scanned from top to bottom, and the columns within a stripe are scanned from left to right.

Tier 2 Coding The input to the Tier-2 coding process is the set of bit-plane coding passes generated during Tier-1 coding.

In Tier-2 coding, the coding pass information is packaged into data units called packets. The resulting packets are then output to the final code stream. A packet is a collection of coding pass data. The packetisation process imposes a particular organisation on coding pass data in the output code stream. Each packet is comprised of two parts—a header and body. The *header* indicates which coding passes are included in the packet, while the *body* contains the actual coding-pass data.

In the code stream, the header and body may appear together or separately, depending on the coding options in effect. Also, the rate scalability is achieved through layers. The coded data for each tile is organised into L layers, numbered from 0 to $L-1$. Each coding pass is either assigned to one of the L layers or discarded. The coding passes containing the most important data are included in the lower layers, while the coding passes associated with finer details are included in higher layers. In the decoder, the Tier-2 decoding process extracts the various coding passes from the code stream and associates each coding pass with its corresponding code block. Also, during decoding, the reconstructed image quality improves incrementally with the process of each successive layer.

(v) Rate Control To meet the target bit-rate constraints under lossy compression, the rate control module adjusts the quantiser step size or discards some of the coding-pass information. The bit stream of each block can be truncated to a variety of discrete lengths to render the possible best rate-distortion image with associated specific compression ratio.

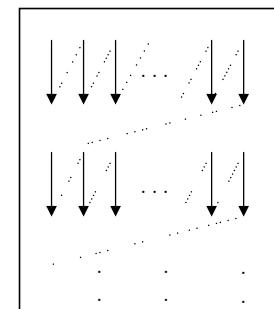


Fig. 12.35 Scan order within a code block

12.13.4 Decoding Process

The decoder basically performs a function opposite of the encoder. The decoding process in JPEG 2000 is illustrated in Fig. 12.36.

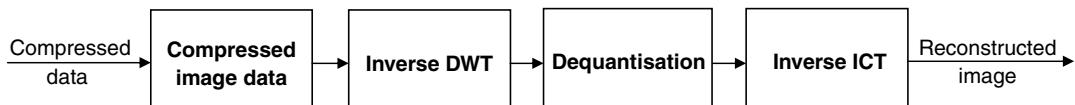


Fig. 12.36 Decoding process in JPEG2000

The code stream is received by the decoder according to the progression order stated in the header. The coefficients in the packets are then decoded, dequantised and reverse ICT is performed to get the reconstructed image. JPEG2000 has the important feature that an image can be compressed in one way and decompressed in many different ways.

12.13.5 Applications of JPEG2000

Due to its excellent coding performance and many attractive features, JPEG2000 has a very large application base. Some of the application areas include (i) image archiving, (ii) Internet, (iii) Web browsing, (iv) document imaging, (v) medical imaging, (vi) remote sensing, and (vii) desktop publishing.

12.14 DESIRABLE PROPERTIES OF WAVELETS

Some of the desirable properties of wavelets are summarised below:

Regularity The regularity of the wavelet-basis function is an important property of the wavelet transform. Due to this property, the wavelet transform can be local in both time and frequency domains. The regularity of the scaling function is determined by the decay of its Fourier transform $\phi(\omega)$ and is defined as the maximum value of ' r ' such that

$$|\phi(\omega)| \leq \frac{c}{(1+|\omega|)^r} \quad (12.47)$$

Vanishing Moments If a wavelet-basis function $\psi(t)$ satisfies the following condition

$$\int_{-\infty}^{\infty} t^k \psi(t) dt = 0 \quad \text{for } k = 1, 2, \dots, M \quad (12.48)$$

then $\psi(t)$ is said to have ' M ' vanishing moments. The vanishing moments determine the smoothness of the function $\psi(t)$. Large vanishing moments increase the smoothness of the circuit at the expense of losing the locality, because large vanishing moments usually mean a large basis support. The number of vanishing moments that a wavelet possesses provides a means for evaluating its energy compaction efficiency. The greater the number of vanishing moments, the better the energy compaction.

Orthogonal and Bi-orthogonal Wavelet Orthogonal filters lead to orthogonal wavelet-basis functions. Hence, the resulting wavelet transform is energy preserving. The wavelet basis forms an orthogonal basis if the basis vectors are orthogonal to its own dilations and translations. A less stringent condition is that the vectors be bi-orthogonal. The DWT and inverse DWT can be implemented by filter banks. This includes an analysis filter and a synthesis filter. When the analysis and synthesis filters are transposes as well as inverses of each other, the whole filter bank is orthogonal. When they are inverses, but not necessarily transposes, the filterbank is bi-orthogonal.

Smoothness Non-smooth basis functions introduce artificial discontinuities under quantisation. These discontinuities are reflected as spurious artifacts in the reconstructed images.

12.15 WAVELET PACKET TRANSFORM

Wavelet packet transform is a generalisation of wavelet transforms that offers a rich set of decomposition structures. Wavelet packet transform was first introduced by Coifman, Meyer and Wickerhauser. Dyadic wavelet decomposition is achieved by iterative two-channel perfect reconstruction filter bank operations over the low-frequency band at each level. Wavelet packet decomposition, on the other hand, is achieved when the filter bank is iterated over all frequency bands at each level. The benefit of the wavelet packets over wavelet decomposition comes from the ability of the wavelet packets to better represent high-frequency content and high-frequency oscillating signals. This allows a wavelet packet transform to perform better than wavelets for compression of images with large amount of texture like fingerprints and Barbara images.

Wavelet packets impose increased computational complexity due to the need to select a basis. Selection of a ‘best basis’ for any particular image is performed in a number of ways. Coifman et al., suggested the use of a cost function. If the sum of the costs of the children is greater than the parents’ cost, the children are pruned; otherwise the children are kept.

Best-basis Selection To select the best wavelet packet basis, (in order to find the optimal subtree of the complete decomposition tree) some criterion, namely, a cost function should be specified. The cost function should satisfy the additive property so that the best basis search can be fulfilled by a fast tree-pruning algorithm. The objective is to select a subtree of the full decomposition tree so that the global cost function is minimised.

Entropy-based Best-Basis Selection Coifman and Wickerhauser proposed the entropy-based best-basis selection. The entropy-based technique prunes the tree to minimise the overall estimated entropy of the wavelet-packet structure. The advantage of entropy-based best-basis selection is that it is simple for only pruning the complete tree once to form an optimal tree. The drawback of the entropy-based best-basis selection is that it does not always produce a valid basis because the cost function and the resulting packet sub-band structure have no relation with the subsequent coding method which determines the efficiency of the coding system.

Rate-distortion Optimisation Best-basis Selection Ramachandran and Martin Vetterli proposed the rate-distortion optimised basis selection method. The procedure consists of finding the best decomposition tree that minimises the global distortion ‘ D ’ for a given coding bit rate ‘ R ’. This constrained minimisation problem of seeking the best subtree can be converted to an equivalent unconstrained problem with the help of Lagrange’s multiplier method. The unconstrained problem reduces to finding the optimal subtree so as to minimise the Lagrangian cost function defined as

$$J(\lambda) = D + \lambda \cdot R \quad (12.49)$$

The advantage of rate-distortion optimisation-based best-basis selection is that it is very flexible for it decouples the basis-search procedure and the best-quantiser choice procedure without sacrificing optimality. The algorithm of the rate-distortion optimisation-based best-basis selection is summarised below:

- (i) Perform the sub-band decomposition (wavelet packet decomposition) of the input image to a given depth N .
- (ii) Generate rate-distortion pairs for each sub-band node of the full decomposition tree.
- (iii) For a given λ , populate each node of the full tree with the lowest Lagrangian cost $D + \lambda R$ over all the quantisers of that node.
- (iv) Prune the full tree recursively, starting from the leaf nodes to the root node, by making merge/split binary decisions at each branch according to whose costs are cheaper. From this step the best basis and the corresponding quantisation precision for each sub-band are known.
- (v) Iterate over λ using the bisection search method to meet the target bit rate ' R '.

12.16 MULTI-WAVELETS

The term '*multi*' means more than one. Multi-wavelets indicate more than one wavelet. Multiwavelets have two or more scaling and wavelet functions. More than one scaling function is represented in a vector format as

$$\Phi(t) = [\phi_1(t), \phi_2(t), \dots, \phi_r(t)]^T \quad (12.50)$$

where $\Phi(t)$ is called multi-scaling function.

The multi-wavelet function is represented by

$$\Psi(t) = [\Psi_1(t), \Psi_2(t), \dots, \Psi_r(t)]^T \quad (12.51)$$

When $r = 1$, $\Psi(t)$ is called a scalar wavelet or simply a wavelet. Scalar wavelets are the wavelets generated from one scaling function. In principle, r can be arbitrarily large, but the multi-wavelets which are widely discussed in literature is only $r=2$.

The two-scale relationship of the multi-scaling function and the multi-wavelet function for $r=2$ is given below:

$$\Phi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} H_k \phi(2t - k) \quad (12.52)$$

$$\Psi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} G_k \phi(2t - k) \quad (12.53)$$

Here, $\{H_k\}$ and $\{G_k\}$ are 2×2 matrix filters defined by

$$H_k = \begin{bmatrix} h_0(2k) & h_0(2k+1) \\ h_1(2k) & h_1(2k+1) \end{bmatrix} \quad (12.54)$$

$$G_k = \begin{bmatrix} g_0(2k) & g_0(2k+1) \\ g_1(2k) & g_1(2k+1) \end{bmatrix} \quad (12.55)$$

The matrix elements provide more degrees of freedom than a traditional scalar wavelet. Due to these extra degrees of freedom, multi-wavelets can simultaneously achieve orthogonality, symmetry and high order of approximation.

12.16.1 Balanced and Unbalanced Multi-wavelets

The balancing order of a multiwavelet is indicative of its energy compaction energy. This concept was introduced by Leburn and Martin Vetterli. High balancing order alone does not ensure good image compression performance. The equation governing the vanishing moment of a scalar wavelet is given by

$$\int t^m \psi(t) dt = 0 \quad (12.56)$$

For a scalar wavelet with vanishing order ' K ', the high-pass branch cancels a monomial of order less than ' K ', and the low-pass branch preserves it. For a multi-wavelet, the equation is given by

$$\int t^m \psi_i(t) dt = 0 \quad (12.57)$$

where $0 \leq i \leq r - 1$ and $0 \leq m \leq K - 1$.

An approximation order ' K ' implies that the high-pass branch cancels monomials of order less than ' K '. In general, for multi-wavelets, the preservation property does not automatically follow from the vanishing moments property. If the multi-filter bank preserves the monomials at the low-pass branch output, the multi-wavelet is said to be balanced. The balancing order is ' p ' if the low-pass and high-pass branches in the filter bank preserve and cancel, respectively, all monomials of order less than ' p ' where $p \leq K$. Multiwavelets that do not satisfy this preservation property are said to be unbalanced. For unbalanced multi-wavelets, the input needs suitable 'prefiltering' to compensate for the absence of the preservation property, which is not required in the case of a balanced multi-wavelet. Thus, a balanced multi-wavelet will be computationally efficient than an unbalanced multi-wavelet.

12.16.2 Need for Multi-wavelet

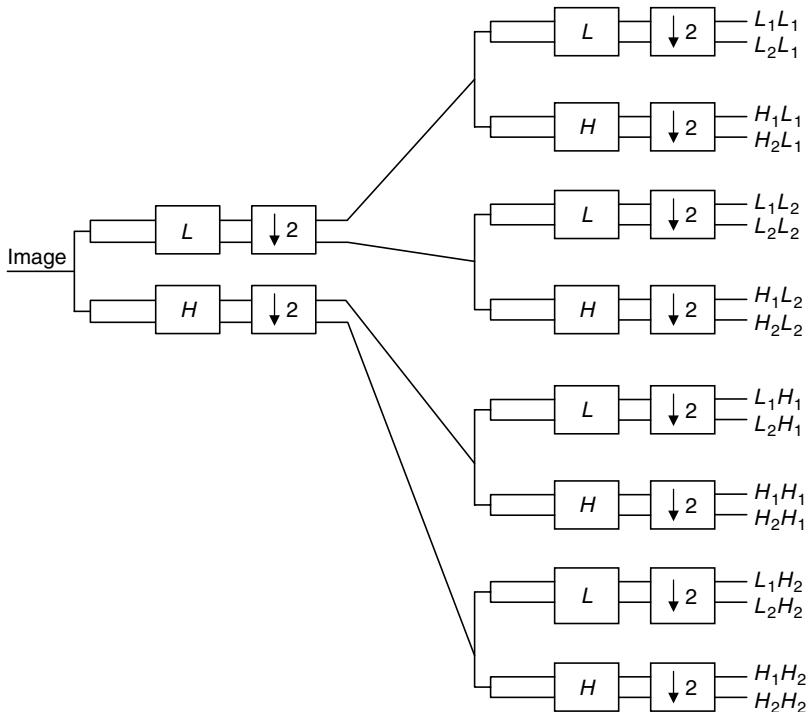
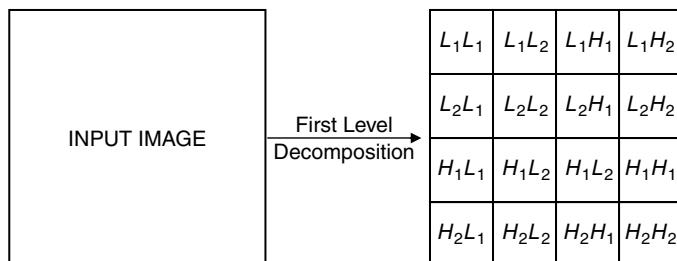
The properties of wavelets like orthogonality, compact support, linear phase, and higher-order vanishing moments of the basis function are useful in image-processing applications. Unfortunately, the scalar wavelets do not possess all the mentioned properties simultaneously. To overcome this problem, more than one scaling function and wavelet functions need to be employed.

Multi-wavelets are generated by more than one scaling function. There are many degrees of freedom in the construction of multi-wavelets. These allow for more features such as orthogonality, symmetry and high-order vanishing moments to be built into a multi-wavelet transform and enable one to construct multi-wavelet filters to suit the different image-processing requirements.

12.16.3 Sub-band Structure of Multi-wavelets

The sub-band structure of a multi-wavelets is different from a scalar wavelet due to its multi-channel nature. The image is split into a two-element vector and preprocessed before the multi-wavelet decomposition. The 1-level multiwavelet decomposition of a 2D image results in 16 sub-bands For $r = 2$ as shown in fig. 12.37 and the corresponding representation is shown in Fig. 12.38. In the case of multi-wavelets, the L and H labels have subscripts denoting the channel to which the data corresponds. For example, the sub-band labeled L_2H_1 corresponds to data from the first channel high-pass filter in the horizontal direction, and the second channel low-pass filter in the vertical direction.

The multiwavelet decompositions iterate on the low-pass coefficients from the previous decomposition.

**Fig. 12.37** 1-Level multiwavelet decomposition**Fig. 12.38** Representation of multi-wavelet decomposition of input image

12.17 CONTOURLET TRANSFORM

Contourlet transform is obtained by combining the Laplacian pyramid with a directional filter bank. Contourlet transform provides a flexible multi-resolution, local and directional expansion for images. In a contourlet transform there are two stages—a Laplacian pyramid followed by directional filter bank (DFB). Laplacian pyramids provide a multi-resolution system while directional filter banks give a directional nature to the contourlet transform. The Laplacian pyramid decomposition at each level generates a down-sampled low-pass version of the original and the difference between the original and the prediction resulting in a band-pass image. Band-pass images from the Laplacian pyramid are fed into a DFB so that the directional information can be captured. This is illustrated in Fig. 12.39.

Thus, a contourlet decomposes the image in a multi-scale and multi-directional manner. The multi-scale nature is provided by a Laplacian pyramid. In a multi-scale representation, the transform represents the image data in a hierarchical manner; each added level contributes to a finer representation of the image. The Laplacian pyramid is first used to capture the point discontinuities, which is then followed by a directional filter bank to link point discontinuities into linear structures. The overall result is an image expansion using elementary images like contour segments. Hence, it is named contourlet transform.

The contourlet basis is illustrated in Fig. 12.40. From the figure it is obvious that the contourlet basis is generated using a Laplacian pyramid followed by a directional filter bank. The figure shows two pyramid levels. The support size of the Laplacian pyramid is reduced by four times, while the number of directions of the DFB is doubled. As a result, the support size of the pyramidal directional Filter-bank basis functions are changed from one level to the next in accordance with the curve-scaling relation. In a contourlet transform, each generation doubles the spatial resolution as well as the angular resolution. In general, the contourlet construction allows for any number of DFB decomposition levels l_j to be applied at each Laplacian pyramid level ' j '. In order to satisfy the anisotropy scaling relation, the number of directions is doubled at every other finer scale of the pyramid.

Need of Contourlet Transform Wavelets are powerful tools in the representation of the signal and are good at detecting point discontinuities. However, they are not effective in representing the geometrical smoothness of the contours. This is because natural images consist of edges that are smooth curves, which cannot be efficiently captured by the wavelet transform. In order to overcome this problem, Candes and Donoho proposed the curvelet transform. The key features of the curvelet element are that they exhibit very high directionality and anisotropy. The construction of curvelet transforms was intended for functions defined in the continuum space. The curvelet transform of an image is obtained by the following steps:

1. Sub-band decomposition of the image into a sequence of sub-bands by using a pyramidal tree-structured filter bank
2. Windowing each sub-band into blocks of appropriate size, depending on the centre frequency of the sub-band.
3. Applying the discrete ridgelet transform on the blocks obtained in Step 2.

The discrete algorithmic implementation of the curvelet transform poses many problems. The windowing of the sub-band coefficients may lead to blocking effects. To minimise the blocking effect, overlapping windows can be employed but it will increase the redundancy of the transform. Inspired by curvelets, Minh Do and Martin Vetterli developed the contourlet transform based on an efficient two-dimensional multi-scale and directional filter bank that can deal effectively with images having smooth contours.

Applications of Contourlet Transform A contourlet transform can be designed to be a tight frame, which implies robustness against noise due to quantisation and thresholding. Because of this property, contourlets

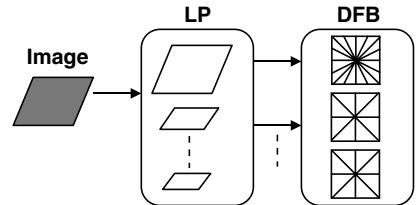


Fig. 12.39 Contourlet transform of the input image

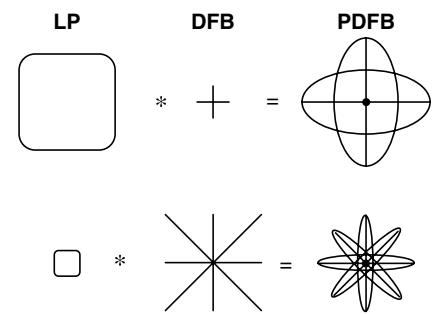


Fig. 12.40 Contourlet basis

can be used in image denoising and image watermarking. Also, the contourlet transform is almost critically sampled, with a small redundancy factor of up to 1.33. Comparing this with a much larger redundancy ratio of the discrete implementation of the curvelet transform, the contourlet transform is more suitable for image compression.

12.18 IMAGE PYRAMID

An image pyramid is a hierarchical representation of an image and is a data structure designed to support efficient scaled convolution through reduced image representation. It consists of a sequence of copies of an original image in which both sample density and resolution are decreased in regular steps. An image pyramid can be considered as a collection of images at different resolutions. The importance of analysing images at many scales arises from the nature of images themselves. Natural scenes contain objects with varying features. As a result, the analysis procedure that is applied only at a single scale may miss some information at other scales. The solution is to carry out analyses at all scales simultaneously.

An image pyramid can also be viewed as an image transformation. The original image can be exactly reconstructed from its pyramidal representation; hence the pyramid code is complete. The pyramid representation can be used in data compression. Although it has one-third more sample elements than the original image, the values of these samples are close to zero, so that they can be represented with minimum number of bits.

The pyramid offers a useful image representation for a number of tasks like image compression, image enhancement, texture analysis, image mosaicing, etc.

Although there are wide varieties of pyramids, the pyramids that are considered here are (i) Gaussian pyramid, and (ii) Laplacian pyramid.

12.18.1 Gaussian Pyramid

The Gaussian Pyramid is a sequence of low-pass, down-sampled images. Gaussian pyramids are efficient to compute coarse-scale images. Gaussian pyramid construction is equivalent to convolving the original image with a set of Gaussian-like weighting functions. The generation of a three-level Gaussian pyramid is illustrated in Fig. 12.41.

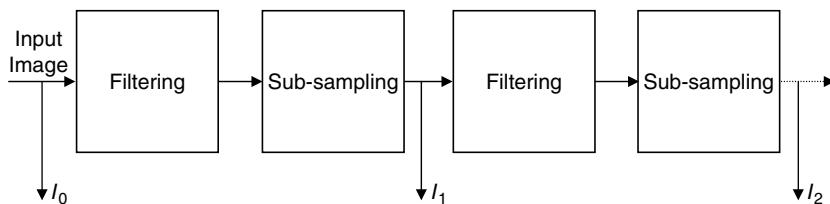


Fig. 12.41 Generation of Gaussian pyramid

The Gaussian pyramid can be used for multi-scale edge estimation. The MATLAB code to get different levels of a Gaussian Pyramid is shown in Fig. 12.42, and the corresponding output is shown in Fig. 12.43.

12.18.2 Laplacian Pyramid

A pyramid is a set of successively smoothed and down-sampled versions of the original image. The Laplacian pyramid as shown in Fig. 12.44 is a decomposition of the original image into a hierarchy of images such that

```

a = imread(apple3.bmp);
a = rgb2gray(a);
b = a;
kernelsize=input('Enter the size of the kernel:');
sd = input('Enter the standard deviation of the Gaussian window:');
rf = input('Enter the Reduction Factor:');
%Routine to generate Gaussian kernel
k = zeros(kernelsize, kernelsize);
[m n] = size(b);
t=0;
for i = 1:kernelsize
    for j = 1:kernelsize
        k(i,j) = exp(-((i-kernelsize/2).^2+(j-kernelsize/2).^2)/
(2*sd.^2))/(2*pi*sd.^2);
        t = t+k(i,j);
    end
end
for i = 1:kernelsize
    for j = 1:kernelsize
        k(i,j) = k(i,j)/t;
    end
end
for t = 1:1:rf
    % convolve it with the picture
    FilteredImg = b;
    if t==1
        FilteredImg = filter2(k,b)/255;
    else
        FilteredImg = filter2(k,b);
    end;
    % compute the size of the reduced image
    m = m/2;
    n = n/2;
    % create the reduced image through sampling
    b = zeros(m,n);
    for i=1:m
        for j = 1:n
            b(i,j) = FilteredImg(i*2,j*2);
        end;
    end;
end;
imshow(b);

```

Fig. 12.42 Code to generate different levels of a Gaussian pyramid

each level corresponds to a different band of image frequencies. A Laplacian pyramid can be described as a data structure composed of band-pass copies of an image. As a band-pass filter, pyramid construction tends to enhance image features such as edges, which are vital for image interpretation.

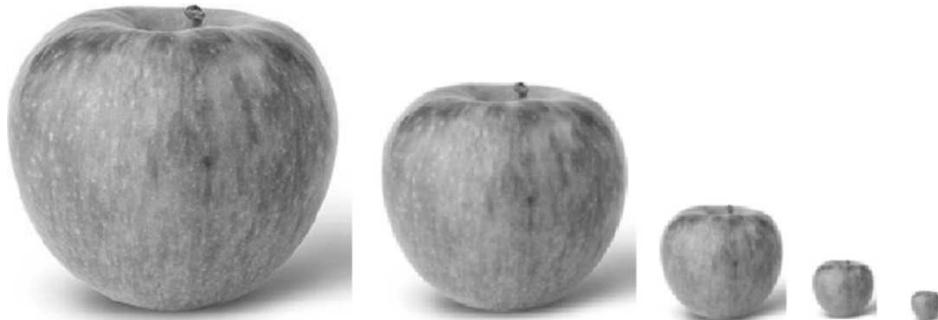


Fig. 12.43 Different levels of a Gaussian pyramid

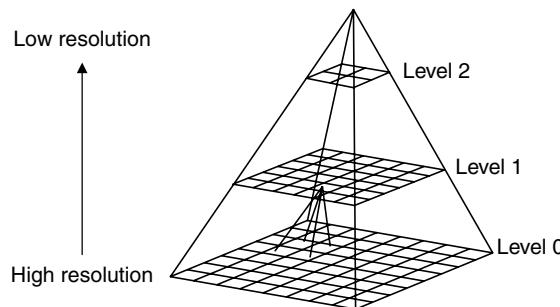


Fig. 12.44 Laplacian pyramid

The generation of a Laplacian pyramid is illustrated in Fig. 12.45. Each level in the Laplacian pyramid represents the difference between successive levels of the Gaussian pyramid. A Laplacian pyramid is a complete image representation, which means that the steps used to construct the pyramid may be reversed to recover the original image exactly. The drawback of Laplacian-pyramid representation is that the basis set is not orthogonal, and the number of transform coefficients exceeds the number of original pixels by a factor of 4/3. The Laplacian pyramids of an image at different levels are shown in Fig. 12.46.

12.18.3 Directional Filter bank

Bamberger and Smith introduced the concept of directional filter bank. A major property of the Directional Filter Bank (DFB) is its ability to extract 2D directional information of an image,

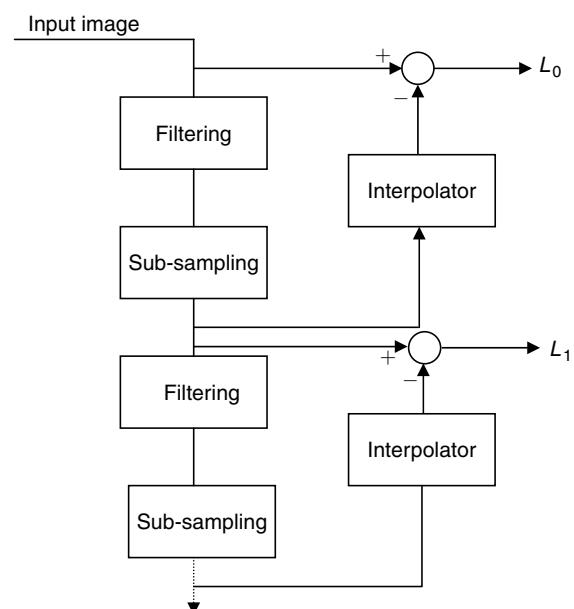


Fig. 12.45 Generation of a Laplacian pyramid

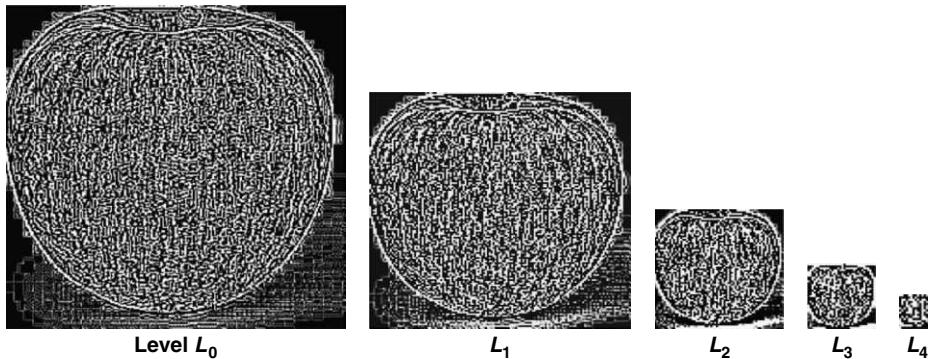


Fig. 12.46 Example of a Laplacian pyramid

which is important in image analysis and other applications. The directional filter bank provides the flexibility to obtain good resolution, both angularly and radially. The DFB is maximally decimated and obeys perfect reconstruction. The term ‘perfect reconstruction’ indicates that the total number of sub-band coefficients is the same as that of the original image, and they can be used to reconstruct the original image without error. A 2^m -band DFB can be implemented by an m -level binary tree structure consisting of ‘ m ’ levels of two-band systems. Each level can be implemented by using separable poly-phase filters, which make the structure extremely computationally efficient. The construction of a DFB is based on two building blocks (a) a fan filter bank which is illustrated in Fig. 12.47, and (b) resampling operators. The key insight in the construction of a DFB is that the signal should be resampled before and after a 2D filter to obtain different sub-band geometries. A three-level DFB tree is illustrated in Fig. 12.50.

In Fig. 12.47, Q_1 is a matrix used to decimate the sub-band signal. In the case of a quincunx matrix, the filter bank is termed *Quincunx filter bank*. In the two-channel quincunx filter bank, the fan filters divide a 2D spectrum into two directions, *horizontal* and *vertical*. The second building block of the DFB is a shearing operator, which amounts to just reordering of image samples. The key in the DFB is to use an appropriate combination of shearing operators together with two-directional partition of quincunx filter banks at each node in a binary tree-structured filterbank, to obtain the desired 2D spectrum division.

The spatial effect of the quincunx down-sampling matrix Q_1 on the input sample is illustrated in Fig. 12.49. The input sample is illustrated in Fig. 12.48. The shaded region represents the quincunx lattice.

The frequency partitioning of the directional filter bank is shown in Fig. 12.51. The DFB realises a division of 2D spectrum into 2^n wedge-shaped slices using an n -level iterated tree-structured filter bank.

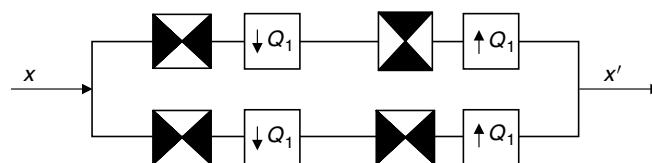


Fig. 12.47 Two-channel fan filter bank

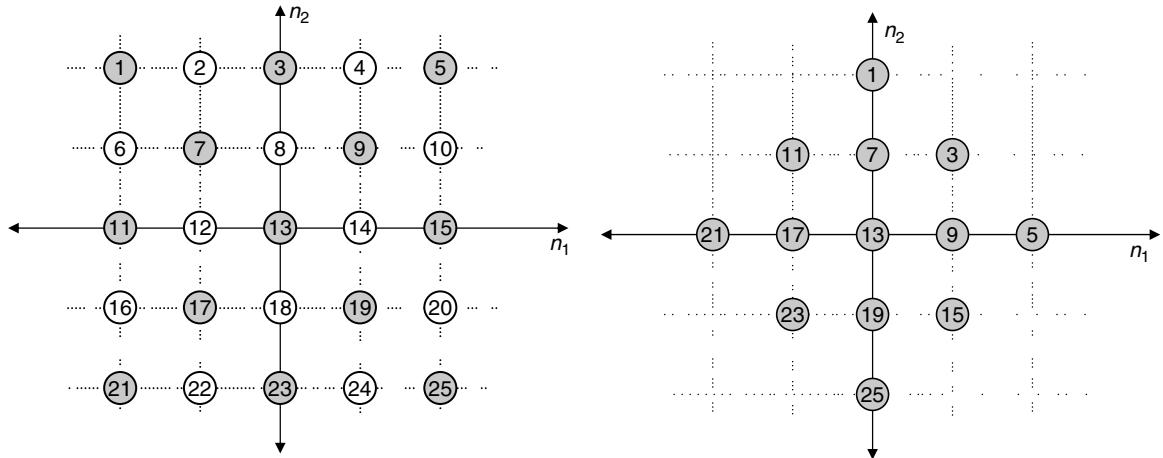


Fig. 12.48 Input data

Fig. 12.49 The spatial effect of down-sampling of input data by Q_1

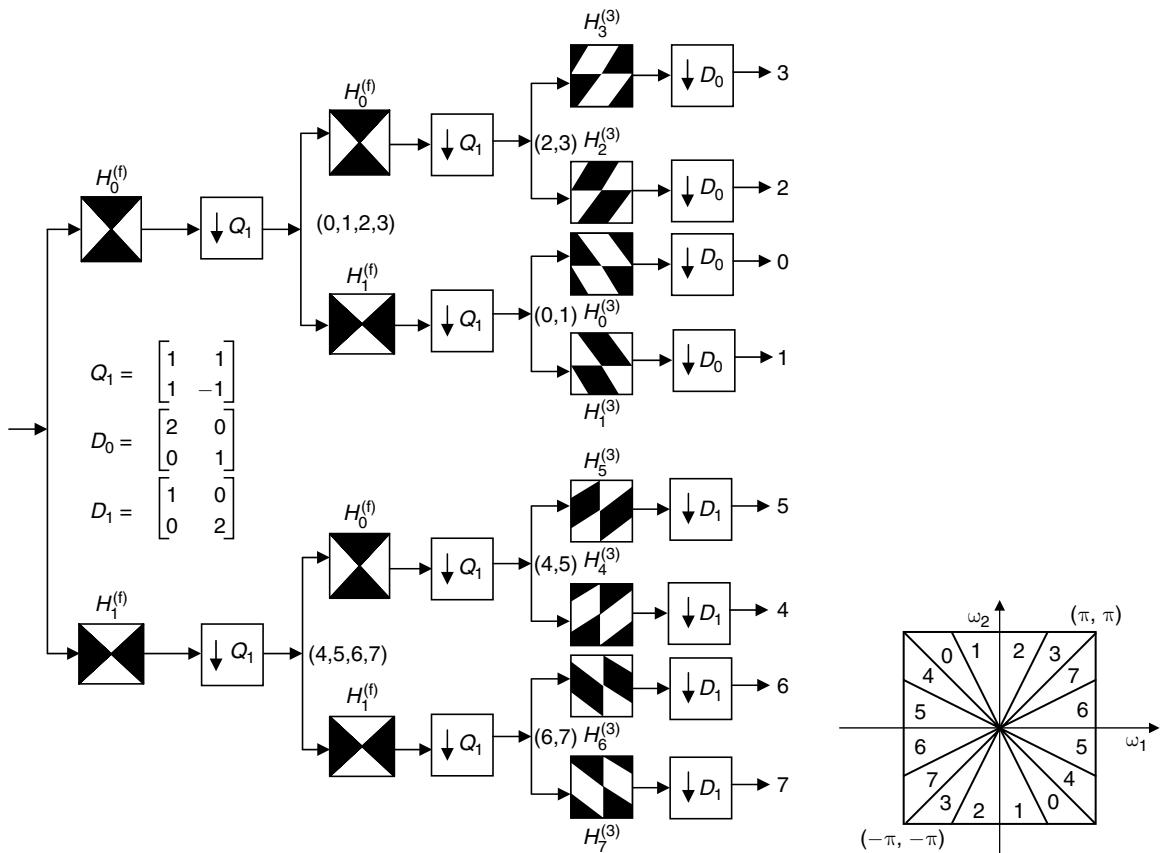


Fig. 12.50 Binary-tree structure directional filter bank

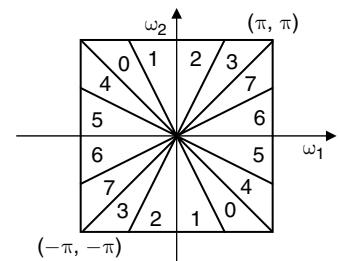


Fig. 12.51 Frequency partitioning of a directional filter bank

12.19 WAVELET-BASED DENOISING

An image is often corrupted by noise during its acquisition or transmission. Image denoising is used to remove the additive noise while retaining as much as possible the important image features. Wavelet denoising attempts to remove the noise present in the signal while preserving the signal characteristics, regardless of its frequency content. The wavelet-based methods mainly rely on thresholding the Discrete Wavelet transform (DWT) coefficients, which have been affected by Additive White Gaussian Noise (AWGN). Thresholding is a simple non-linear technique which operates on one wavelet coefficient at a time. In its most basic form, each coefficient is thresholded by comparing against the threshold. If the coefficient is smaller than threshold, it is set to zero; otherwise the coefficient is kept or modified.

Generally, smoothing removes high frequency and retains low frequency. Also, smoothing leads to blurring of the image. Deblurring increases the sharpness of the image by boosting the high frequencies, whereas denoising tries to remove whatever noise is present regardless of the spectral content of a noisy signal. Restoration is a kind of denoising that tries to retrieve the original signal with optimal balancing of deblurring and smoothing. Traditional smoothing filters such as mean, median and Gaussian filters are linear operators normally employed in the spatial domain, which smooth the signals with blurring effects. A frequency domain approach such as ‘inverse filtering’ is sensitive to noise. The Wiener filter executes an optimal trade-off between inverse filtering and noise smoothing. It is useful in restoration by removing the additive noise and inverting the blurring simultaneously. Wavelet-based denoising is widely popular due to properties such as sparsity and multi-resolution structure. Wavelet-based denoising methods mostly employ non-linear thresholding of wavelet coefficients in the time-scale transform domain.

Image denoising using DWT consists of three steps, namely, image decomposition, followed by thresholding of wavelet coefficients and image reconstruction.

12.19.1 Wavelet Shrinkage Denoising

Denoising by thresholding in wavelet domain was developed by Donoho. In wavelet domain, the large coefficients correspond to the signal, and small ones represent mostly noise. Wavelet-based denoising involves three steps: linear wavelet transform, non-linear thresholding step and a linear inverse wavelet transform.

Shrinkage Strategies The standard thresholding of wavelet coefficients is governed mainly by either ‘hard’ or ‘soft’ thresholding function as illustrated in Fig. 12.52.

From Fig. 12.52, the linear function is not useful for denoising as it does not alter the coefficients.

In hard thresholding, the wavelet coefficients below the threshold λ are made zero and coefficients above threshold are not changed. If x, y denote the input and the output then the hard threshold $T_h(y, \lambda)$ is given by

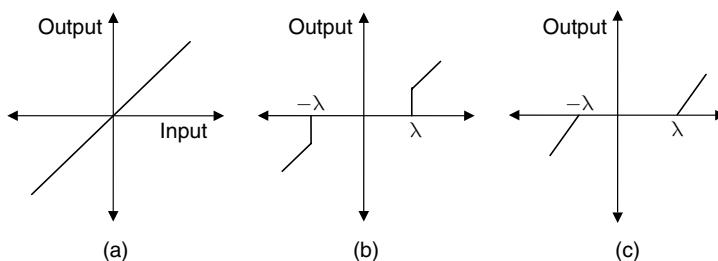


Fig. 12.52 Thresholding functions (a) Linear (b) Hard (c) Soft

$$\hat{X} = T_h(Y, \lambda) = \begin{cases} y, & \text{if } |y| \geq \lambda \\ 0, & \text{otherwise} \end{cases} \quad (12.58)$$

In soft thresholding, the wavelet coefficients are shrunk towards zero by an offset λ . Generally, soft thresholding gives fewer artifacts and preserves the smoothness. The soft thresholding operator $T_s(Y, \lambda)$ is given as

$$\hat{X} = T_s(Y, \lambda) = \begin{cases} y - \lambda, & \text{if } y \geq \lambda \\ y + \lambda, & \text{if } y \leq -\lambda \\ 0, & \text{otherwise} \end{cases} \quad (12.59)$$

The choice of the threshold plays a crucial role in image denoising.

12.19.2 Threshold Selection

A small threshold may yield a result close to the input, but the result may be still noisy. A large threshold produces a signal with a large number of zero coefficients. This leads to an overly smooth signal. The smoothness will suppress the details and edges of the original image, thus causing blurring and ringing artifacts.

Universal Threshold Originally, Donoho and Johnstone proposed universal threshold. The universal threshold is given by

$$\lambda_{univ} = \sqrt{2 \ln(M)} \times \sigma_w \quad (12.60)$$

where M is the signal size and σ_w^2 is the noise variance.

12.20 WAVELET-THRESHOLDING METHODS

The different types of wavelet thresholding methods are (i) VisuShrink, (ii) LevelShrink, (iii) SureShrink, and (iv) BayesShrink. The methods differ only in the selection of the threshold λ and the strategy employed in applying the threshold operator.

12.20.1 VisuShrink

The VisuShrink technique consists of applying the soft-thresholding operator using the universal threshold. The main feature of VisuShrink is that it guarantees a highly smoothed reconstruction of the noisy image, but in doing so it often compromises many of the important features of the image by setting the threshold conservatively large. Also, VisuShrink fails to adapt to the various statistical and structural properties of the wavelet tree. In VisuShrink, the universal threshold is applied uniformly throughout the wavelet tree. However, the use of different thresholds for different decomposition levels and subbands seems to be more reasonable.

12.20.2 LevelShrink

LevelShrink proposes the use of different thresholds for different levels of the wavelet tree. The content of various sub-bands varies from one level to the next, and the use of level-dependent thresholds seems to be more reasonable than the use of a uniform threshold. The threshold at the j^{th} decomposition level of the wavelet tree is given by

$$\lambda_j = \sqrt{2 \ln(M)} \times \sigma_w \times 2^{-(J-j)/2} = \lambda_{\text{univ}} \times 2^{-(J-j)/2} \text{ for } j = 1, 2, \dots, J$$

where J is the total number of decomposition levels and j is the scale level where the wavelet coefficient to be thresholded is located.

The LevelShrink thresholding method is more adaptive than VisuShrink since it adapts to the variability registered within the wavelet tree from one decomposition level to the next by using different thresholds for different levels. However, this level-dependent thresholding does not account for the inherent variability from one sub-band to another at the same decomposition level of the wavelet tree. In fact, the same threshold is used for the horizontal, vertical and diagonal sub-bands of the same decomposition level. In practice, the sub-bands contain different types of details of the image and they should be treated and thresholded differently.

12.20.3 SureShrink

Donoho and Johnstone developed an adaptive method of selecting a threshold that minimises the Stein Unbiased Risk Estimator (SURE), which is known as the SureShrink wavelet thresholding technique. The adaptivity of SureShrink is achieved by choosing distinct thresholds for each sub-band of each level of the wavelet tree using an efficient recursive process. This thresholding scheme attempts to select thresholds that adapt to the data as well as minimise an estimation of the mean squared error.

The threshold selection in SureShrink is determined by adopting the following procedure:

Let X_j^{sub} and Y_j^{sub} represent the wavelet coefficients corresponding to the original and noisy images of size M_j in the sub-band location $\text{sub} \in \{\text{horizontal, vertical, diagonal}\}$ and decomposition level $j \in \{1, 2, \dots, J\}$. In order to analyse the impact of choice of the threshold on the risk, let $R(X_j^{\text{sub}}, \lambda)$ denote the risk of a soft thresholding operator $T_s(\cdot, \lambda)$ calculated with a threshold λ . That is,

$$R(X, \lambda) = E \left[\left\| X_j^{\text{sub}} - T_s(Y_j^{\text{sub}}, \lambda) \right\|^2 \right] \quad (12.61)$$

The original image is generally not known, that is, X_j^{sub} is not known and the estimate $\hat{R}(X_j^{\text{sub}}, \lambda)$ of $R(X_j^{\text{sub}}, \lambda)$ is calculated from Y_j^{sub} and the best threshold level $\lambda_j^{(\text{sub})*}$ can be estimated by minimising $\hat{R}(X_j^{\text{sub}}, \lambda)$. The soft threshold operator in this case is given by

$$\hat{X}_j^{\text{sub}} = T_s(Y_j^{\text{sub}}, \lambda) \quad (12.62)$$

$$\text{where } T_s(Y_j^{\text{sub}}, \lambda) = \begin{cases} Y_{j,m}^{\text{sub}} - \lambda, & \text{if } Y_{j,m}^{\text{sub}} \geq \lambda \\ Y_{j,m}^{\text{sub}} + \lambda, & \text{if } Y_{j,m}^{\text{sub}} \leq -\lambda \\ 0, & \text{otherwise} \end{cases}$$

Let us consider two cases:

Case (i) If $|Y_{j,m}^{\text{sub}}| < \lambda$ then the soft thresholding operator sets this coefficient to zero, which produces a risk equal to $|X_{j,m}^{\text{sub}}|^2$ since

$$E \left[\left| Y_{j,m}^{\text{sub}} \right|^2 \right] = \left| X_{j,m}^{\text{sub}} \right|^2 + \sigma_w^2 \quad (12.63)$$

Also, one can estimate

$$\left|X_{j,m}^{\text{sub}}\right|^2 \approx \left|Y_{j,m}^{\text{sub}}\right|^2 - \sigma_w^2 \quad (12.64)$$

Case (ii) If $\left|Y_{j,m}^{\text{sub}}\right| \geq \lambda$, the soft thresholding subtracts λ from the amplitude $\left|Y_{j,m}^{\text{sub}}\right|$. The expected risk is the sum of the noise energy plus the bias introduced by the reduction of the amplitude of $Y_{j,m}^{\text{sub}}$ by λ . Thus, the expected risk associated with a wavelet coefficient $\left|Y_{j,m}^{\text{sub}}\right| < \lambda$ can be estimated by $\sigma^2 + \lambda^2$. The resulting total risk estimator of $R(X_j^{\text{sub}}, \lambda)$ is

$$\hat{R}(X_j^{\text{sub}}, \lambda) = \sum_{m=1}^{M_j} \Phi\left(\left|Y_{j,m}^{\text{sub}}\right|^2\right) \quad (12.65)$$

where

$$\sum_{m=1}^{M_j} \Phi\left(\left|Y_{j,m}^{\text{sub}}\right|^2\right) = \begin{cases} \left|Y_{j,m}^{\text{sub}}\right|^2 - \lambda^2, & \text{if } \left|Y_{j,m}^{\text{sub}}\right| \leq \lambda \\ \sigma^2 + \lambda^2, & \text{if } \left|Y_{j,m}^{\text{sub}}\right| > \lambda \end{cases}$$

Here, $\hat{R}(X_j^{\text{sub}}, \lambda)$ is an unbiased estimator of $R(X_j^{\text{sub}}, \lambda)$. This unbiased risk estimator $\hat{R}(X_j^{\text{sub}}, \lambda)$ is known as Stein's Unbiased Risk Estimator (SURE). To find an optimal threshold level $\lambda_j^{(\text{sub})*}$ that minimises the SURE estimator $\hat{R}(X_j^{\text{sub}}, \lambda)$, the following procedures have to be adopted:

- (i) The wavelet coefficients $Y_{j,m}^{\text{sub}}$, within each vertical, horizontal and diagonal sub-band at each level $j = 1, 2, \dots, J$ are sorted in decreasing magnitude order.
- (ii) Now, let $Y_j^{\text{sub}(r)}(k)$, $k = 1, 2, \dots, M_j$ be the ordered coefficient of order r , where M_j is the number of coefficients in a sub-band at the decomposition level j .
- (iii) The risk estimator is given by
$$\hat{R}(X_j^{\text{sub}}, \lambda) = \sum_{k=l}^{M_j} \left|Y_j^{\text{sub}(r)}(k)\right|^2 - (M-l)\sigma^2 + l(\sigma^2 + \lambda^2) \quad \text{where } l \text{ is an index such that } \left|Y_j^{\text{sub}(r)}(l)\right| \leq \left|Y_j^{\text{sub}(r)}(l+1)\right|.$$
- (iv) Since $\hat{R}(X_j^{\text{sub}}, \lambda)$ is an increasing function of λ , one must choose $\lambda_j^{\text{sub}*} = \left|Y_j^{\text{sub}(r)}(l)\right|$ to minimise $\hat{R}(X_j^{\text{sub}}, \lambda)$.

To find an optimal threshold λ_{sure}^* that minimises $\hat{R}(X_j^{\text{sub}}, \lambda)$, it is sufficient to compare the M_j possible values $\left\{Y_j^{\text{sub}(r)}(l)\right\}, l = 1, 2, \dots, M_j$, and choose the value of λ that yields the smallest risk estimator $\hat{R}(X_j^{\text{sub}}, \lambda)$.

The SureShrink thresholding method provides an adaptive thresholding strategy. The performance of the SureShrink method is dependent on estimating the statistics of the wavelet coefficients of the original image from the statistics in the wavelet transform of the noisy image.

Drawback of SureShrink-based Thresholding The SureShrink method has a drawback in certain situations, for example, when there is extreme sparsity of the wavelet coefficients. In such cases, the noise contributes to the SURE profile through the coordinates at which the signal is zero or close to zero. The wavelet coefficients of the noisy image at these locations, which correspond mainly to noise, will swamp the information contributed to the SURE profile by the few coordinates where the signal is non-zero.

12.20.4 BayesShrink

The BayesShrink wavelet thresholding adopts the Bayesian approach which assumes the knowledge of the probability distribution of the original signal and seeks to optimise the threshold operator for the purpose of minimising the expected risk. In particular, it is assumed that for the various sub-bands and decomposition levels, the wavelet coefficients of the original image follow approximately a Generalised Gaussian Distribution (GGD). The Generalised Gaussian distribution is given by

$$GG_{\sigma_{X_j^{\text{sub}, \beta(x)}}} = C(\sigma_{X_j^{\text{sub}, \beta}}) e^{-\left[\alpha(\sigma_{X_j^{\text{sub}, \beta}}) |x|\right]^2}, \text{ for } -\infty < x < \infty \text{ and } \beta > 0 \quad (12.66)$$

where $\alpha(\sigma_{X_j^{\text{sub}, \beta}}) = \sqrt{\frac{\Gamma(3/\beta)}{\Gamma(1/\beta)}}$ and $C(\sigma_{X_j^{\text{sub}, \beta}}) = \frac{\beta \alpha(\sigma_{X_j^{\text{sub}, \beta}})}{2\Gamma(1/\beta)}$. The parameter $\sigma_{X_j^{\text{sub}, \beta}}$ is the standard deviation and β is the shape parameter.

In wavelet domain, we have the relationship

$$Y_j^{\text{sub}} = X_j^{\text{sub}} + W_j^{\text{sub}}$$

Due to the independence assumption between the original signal ‘ x ’ and the noise ‘ w ’, the joint distribution of X_j^{sub} and W_j^{sub} is the product of the distribution of X_j^{sub} and W_j^{sub} . The conditional probability distribution of X_j^{sub} , given the observed noisy wavelet coefficients Y_j^{sub} , is called the posterior distribution. This posterior distribution can be used to construct a decision soft-thresholding operator that computes a denoised estimate $\hat{X}_j^{\text{sub}} = T(X_j^{\text{sub}}, \lambda)$ of X_j^{sub} from the noisy data Y_j^{sub} by minimising the Bayes risk.

Advantage of BayesShrink The Bayesian thresholding strategy is consistent with the human visual system which is less sensitive to the presence of noise in the vicinity of edges. However, the presence of noise in flat regions of the image is perceptually more noticeable.

12.21 COMPARISON OF DIFFERENT THRESHOLDING METHODS

The VisuShrink method uses universal threshold uniformly throughout the wavelet tree. But the universal threshold tends to be conservatively high, resulting in extra smoothing and visible degradation of the edges in the image.

The LevelShrink thresholding scheme uses thresholds that adapt only to the wavelet decomposition level. The LevelShrink method yields better results when compared to the SureShrink method.

The BayesShrink method performs denoising that is consistent with the human visual system and is less sensitive to the presence of noise in the vicinity of edges.

12.22 DIGITAL IMAGE WATERMARKING

The rapidly growing field of digitised images, video and audio has urged the need of copyright protection, which can be used to produce evidence against any illegal attempt to either reproduce or manipulate them in order to change their identity. Digital watermarking is a technique providing embedded copyright information in images.

12.22.1 Watermarking and Cryptography

Watermarking and cryptography are closely related but watermarking is distinct from encryption. In the digital watermarking system, information carrying the watermark is embedded in an original image. The watermarked image is transmitted or stored, and then decoded to be resolved by the receiver. Cryptography scrambles the image so that it cannot be understood. The general principle in cryptography is shown in Fig. 12.53.

The goal of watermarking is not to restrict access to the original image, but to ensure that embedded data remain recoverable. The general watermarking system is illustrated in Fig. 12.54.

Digital watermarking is intended by its developers as the solution for providing value added protection on top of data encryption and scrambling for content protection.

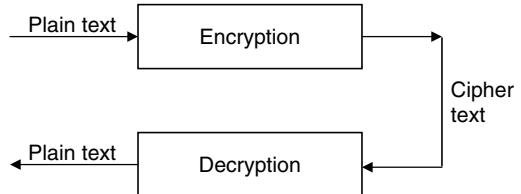


Fig. 12.53 Principle of cryptography

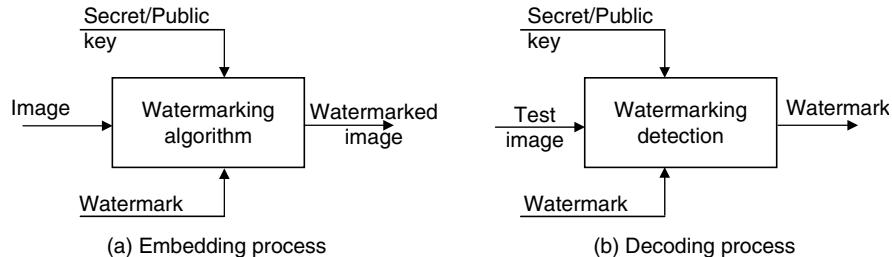


Fig. 12.54 Digital watermarking scheme

12.23 CLASSIFICATION OF WATERMARKING METHODS

Watermarking techniques can be divided into four categories according to the type of document to be watermarked. They are (i) text watermarking, (ii) image watermarking, (iii) audio watermarking, and (iv) video watermarking. In this chapter, the main focus is on image watermarking. Image watermarking can be done both in the spatial domain as well as in the frequency domain. The taxonomy of watermarking techniques is depicted in Fig. 12.55.

In images, the watermarking techniques can be broadly classified into three types (i) Visible watermark, (ii) Invisible fragile watermark, and (iii) Invisible robust watermark.

A *visible watermark* is a secondary translucent overlaid over the primary image. The watermark appears visible to a casual viewer on careful inspection. The invisible fragile watermark is embedded on the primary image in such a way that any manipulation or modification of the image would alter or destroy the watermark. The invisible robust watermark is embedded on the primary image in such a way that an alteration made to the pixel value is perceptually not noticeable and it can be recovered only with an appropriate decoding mechanism.

From the application point of view, a digital watermark could be either source-based or destination-based. Source-based watermarks are desirable for ownership identification or authentication where a unique watermark identifying the owner is introduced to all the copies of a particular image being distributed. A source-based watermark could be used for authentication and to determine whether a received image has been tampered with. The watermark could also be destination-based where each distributed copy gets a unique watermark, and it could be used to trace the buyer in the case of illegal reselling.

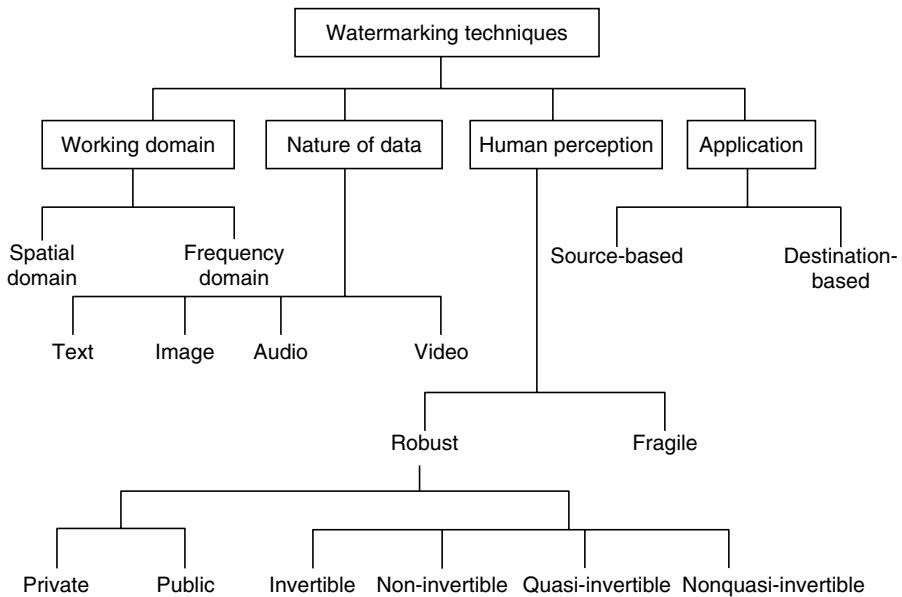


Fig. 12.55 Taxonomy of watermarking techniques

12.24 WATERMARKING IN THE SPATIAL DOMAIN

The most common and simplest watermarking technique in the spatial domain is the Least Significant Bit (LSB). The watermark to be embedded is placed in the LSB of the source image. Spatial domain methods are less complex as no transform is used, but are not robust against attacks.

12.24.1 Flow Chart of Watermarking in Spatial Domain

The idea of watermarking in the spatial domain is given in the form of a flow chart in Fig. 12.56.

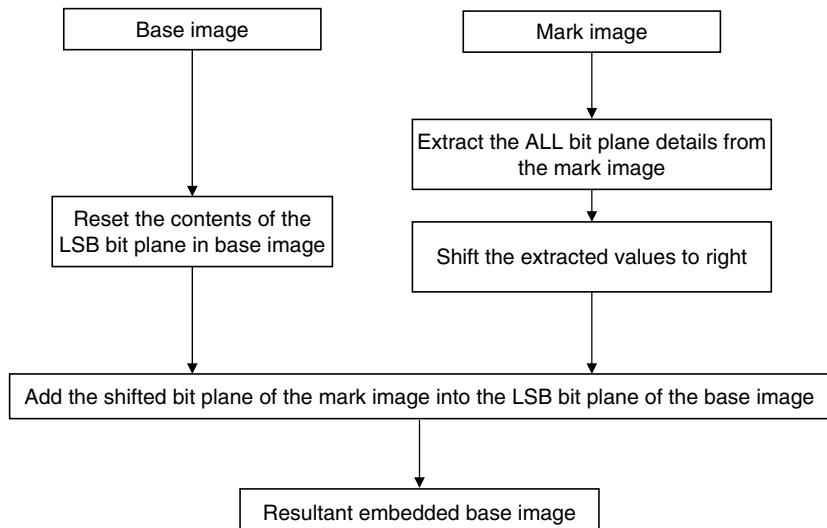


Fig. 12.56 Flow chart for watermarking in spatial domain

The MATLAB code to perform watermarking in the spatial domain as per the flow chart given in Fig. 12.56 is given in Fig. 12.57 and the corresponding output is given in Fig. 12.58.

Extraction of Watermark After embedding the watermark in the base image, it is necessary to extract the watermark from the base image. The procedure to extract the watermark from the base image is given in the form of a flow chart shown in Fig. 12.59.

The MATLAB code that performs the extraction of the watermark from the marked image is shown in Fig. 12.60. The result of the MATLAB code is illustrated in Fig. 12.61.

```

clear all
clc
close all
a=imread('cameraman.tif');
Fig., imshow(a),title('Base Image');
b=imresize(rgb2gray(imread('psg1.jpg')),[32 32]);
[m1 n1]=size(b);
Fig.,imshow(b),title('Mark Image');
[m n]=size(a);
i1=1;
j1=1;
p=1
c=a; i1=1; j1=1;
for ff=1:8,
    for i=1:32,
        j1=1;
        for j=j1:j1+n1-1,
            a(i,j)=bitand(a(i,j),254);% LSB of base image is set to
zero.
            temp=bitand(b(i,j1),2^(ff-1));%MSB of the mark is
extracted.
            temp=temp/(2^(ff-1));
            c(i,j)=bitor(a(i,j),temp);%MSB of mark is inserted into
the %LSB of the base.
            j1=j1+1;
        end
    end
    j1=j1+32;
end
Fig.,imshow(uint8(c)),title('Marked Image');
imwrite(c,'markimg.tif');

```

Fig. 12.57 MATLAB code to implement watermarking in spatial domain



Fig. 12.58 Results of watermark program given in Fig. 12.57

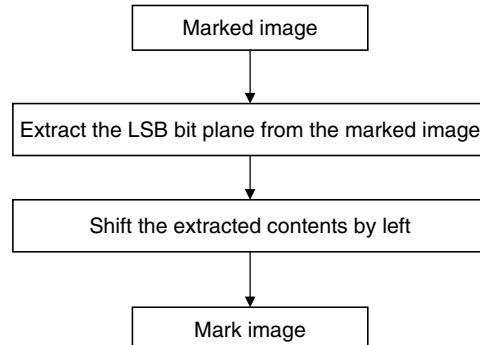


Fig. 12.59 Extraction of watermark from marked image

```

close all;
clear all;
clc
c1=imread('markimg.tif');
figure,imshow(uint8(c1)),title('Marked Image');
i1=1;
j1=1;
d=zeros(32,32);
[m1 n1]=size(d);
jjj=1;
  
```

Fig. 12.60 (Continued)

```

for ff=1:8,
    for i=1:32
        for j=j1:j1+n1-1,
            temp=bitand(c1(i,j),1);
            temp=temp*2^(ff-1);
            d(i,jjj)=d(i,jjj)+temp;
            jjj=jjj+1;
        end
        jjj=1;
    end
    j1=j1+32;
end
Fig.,imshow(uint8(d)),title(Extracted Mark);

```

Fig. 12.60 MATLAB code to extract the watermark from the marked image

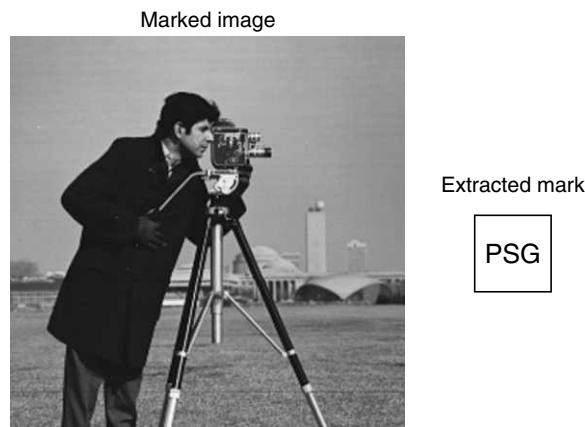


Fig. 12.61 Extracted watermark

12.25 WATERMARKING IN FREQUENCY DOMAIN

To obtain better imperceptibility as well as robustness, the addition of the watermark is done in a transform domain. DCT and DWT are popular transforms widely used in transform-based watermarking. Frequency-based techniques are very robust against attacks involving image compression and filtering, because the watermark is actually spread throughout the image, not just operating on an individual pixel.

DCT watermarking can be done for an entire image or block-wise. In both these methods, the image is transformed into its DCT coefficients, which arranges them based on the frequency, and the watermark is added to these coefficients. The watermarking is achieved by altering the transform coefficients of the image. Finally, the watermarked coefficients are inverse-transformed into the spatial domain thereby spreading the watermark throughout the image or the block of the image.

Wavelet-based watermarking methods exploit the frequency information and spatial information of the transformed data in multiple resolutions to gain robustness. The wavelet transform is closer to the human visual

system since it splits the input image into several frequency bands that can be processed independently. It is a multi-resolution transform that permits to locate image features such as smooth areas, edges or textured areas.

The block diagram of the watermarking scheme implemented is given in Fig. 12.62.

The MATLAB code that implements the wavelet-based watermarking scheme is shown in Fig. 12.63. The code performs the following task:

- (i) The wavelet transform of the input image is performed. Actually, only one level of decomposition is performed to get four sub-bands, namely, LL, LH, HL and HH respectively.

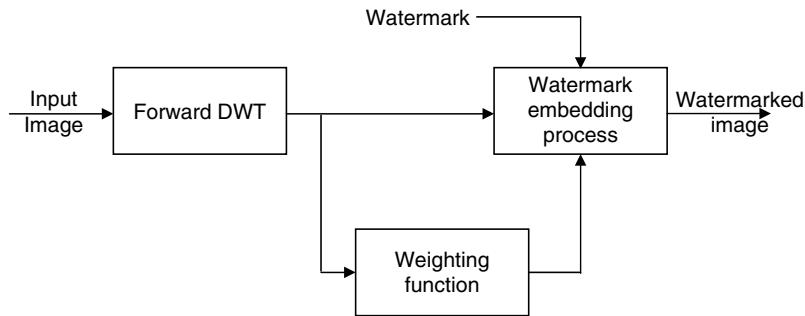


Fig. 12.62 Watermarking scheme

```

img = imread(baby.bmp);
img = rgb2gray(img);
img=imresize(img,[300 300]);
img = double(img);
c = 0.001; %Initialise the weight of Watermarking
Fig.,imshow(uint8(img)),title(Original Image);
[p q] = size(img);
p1=p;
q1=q;
%Generate the key
n = rgb2gray(imread(keyimg1.png));
key = imresize(double(n),[p q]);
Fig.,imshow(uint8(key)),title(Key);
[ca, ch, cv, cd] = dwt2(img, db1); %Compute 2D wavelet transform
%Perform the watermarking
y = [ca ch; cv cd];
Y = y + c*key;
p=p/2; q=q/2;
for i=1:p
    for j=1:q
        nca(i,j) = Y(i,j);
        ncv(i,j) = Y(i+p,j);
        nch(i,j) = Y(i,j+q);
        ncd(i,j) = Y(i+p,j+q);
    end
end
  
```

Fig. 12.63 (Continued)

```
%Display the Watermarked image
wimg = idwt2(nca,nch,ncv,ncd,db1);
Fig.,imshow(uint8(wimg)),title(Watermarked Image);
%Extraction of key from Watermarked image
[rca,rch,rcv,rcd] = dwt2(wimg,db1);%Compute 2D wavelet transform
n1=[rca,rch;rcv,rcd];
N1=n1-y;
Fig.,imshow(double(N1*4)),
title(Extract the key from watermarked image)
```

Fig. 12.63 MATLAB code to implement wavelet-based watermarking

- (ii) The key, which is the image to be embedded, is taken. The key is multiplied with the weighting function and it is then added to the sub-band information of the original image.
- (iii) The inverse wavelet transform is then taken to get the watermarked image.
- (iv) In order to retrieve the key, the forward wavelet transform of the watermarked image is taken and it is subtracted from the wavelet coefficients of the original image to retrieve the key image.



(a) Original image

Dr. S. Jayaraman, Mr. S. Esakkirajan,
Mr. T. Veerakumar & Mr. V. Senthil Murugan,
PSG College of Technology,
Coimbatore, India.

(b) Key



(c) Watermarked image

Dr. S. Jayaraman, Mr. S. Esakkirajan,
Mr. T. Veerakumar & Mr. V. Senthil Murugan,
PSG College of Technology,
Coimbatore, India.

(d) Extracted Key

Fig. 12.64 Output of Wavelet based Watermarking Scheme

12.26 APPLICATIONS OF DIGITAL WATERMARKING

Digital watermarking finds its application in the areas of (i) Copyright protection, (ii) tamper, assessment (iii) hidden annotations, and (iv) communications.

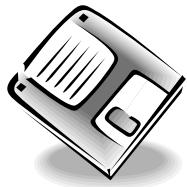
(i) Copyright Protection Digital watermarking is used to embed the copyright and authentication codes within media content. The large-scale distribution of multimedia data has created the need to protect digital information against illegal duplication and manipulation. For copyright-protection applications, the watermark must be recoverable even when the watermarked signal undergoes a reasonable level of distortion. In addition to embedding copyright information, the recipient information can be embedded in the original image to trace the image distribution.

(ii) Authentication Digital watermarks are useful in the field of electronic commerce and distribution of multimedia content to end users for the proof of authenticity of documents. Digital watermarks find a huge market in forensic applications.

(iii) Hidden Annotations Digital watermarks can create hidden labels and annotations in medical applications. In medical applications, watermarks can be used for identifying patient records. These records can be embedded directly into the image data associated with individual patients, speeding access to records and preventing the error of mismatching records and patients.

(iv) Secure Communications Digital watermarks find applications in the defense sector where it is a must to transmit the data secretly.

Summary



- A wavelet is an oscillatory functions of finite duration. Its spectrum resembles the transfer function of a bandpass filter.
- A set of basis functions for a wavelet transform can be generated from dilations and translations of a basic wavelet.
- The continuous wavelet transform represents a signal as a function of time and scale.
- The discrete wavelet transform represents an N -point signal with N coefficients. It represents an N -by- N image with N^2 coefficients.
 - Discrete Wavelet Transform (DWT) can be implemented through (a) filter bank, or (b) lifting scheme.
- Multi-resolution analysis is possible with wavelet transform. Through multi-resolution analysis, a signal can be analysed at different scales.
- Wavelets are widely used in the field of image compression, image denoising and image watermarking. The choice of the wavelet depends on the nature of the input image as well as the nature of application.
- Progressive image coding is possible through wavelet transform. The most commonly used technique to encode wavelet coefficients are (i) EZW, (ii) SPIHT, (iii) SPECK, and (iv) EBCOT.
- The different types of wavelet-thresholding methods for image denoising are (i) VisuShrink, (ii) LevelShrink, (iii) SureShrink, and (iv) BayesShrink.

Review Questions

1. What is the relationship between scale and frequency?

A higher scale corresponds to a more stretched wavelet and a low scale corresponds to a compressed wavelet. This implies that when the scale is high, the changes are slow and the frequency is low. When the scale is low, the changes are rapid and the frequency is high.

Mathematically, this can be proved as below:

If $\Psi(t)$ represents the mother wavelet, the daughter wavelet can be derived from the mother wavelet by means of scaling and shifting operation which is denoted by $\Psi_{a,b}(t)$ where

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}} \Psi\left(\frac{t-b}{a}\right)$$

where b is the shifting parameter and a is the scaling parameter. For convenience, let us ignore the normalisation constant and the shifting parameter. Then the expression for the daughter wavelet is given by

$$\Psi_{a,b}(t) = \Psi\left(\frac{t}{a}\right)$$

For convenience, let us consider the Haar wavelet. The reason for choosing the Haar wavelet is that it is perfectly symmetric.

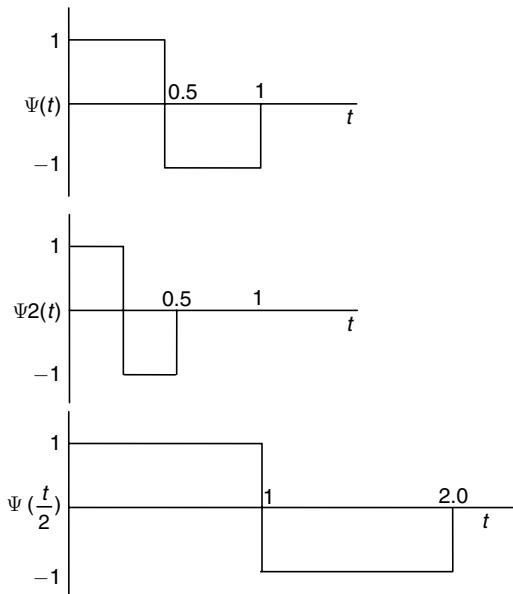


Fig. 12.65 Scaling of the mother wavelet

Case (i) In the expression for the daughter wavelet $\psi_{a,b}(t) = \psi\left(\frac{t}{a}\right)$, substitute the scaling parameter $a = \frac{1}{2}$ to get $\psi_{a,b}(t) = \psi(2t)$ which means compression of $\psi(t)$ by a factor of two. From Fig. 12.63, it is obvious that $\psi(2t)$ exhibits rapid change when compared to $\psi(t)$. Thus low scale corresponds to high frequency.

Case (ii) In the expression for the daughter wavelet $\psi_{a,b}(t) = \psi\left(\frac{t}{a}\right)$, substitute the scaling parameter $a = 2$ to get $\psi_{a,b}(t) = \psi\left(\frac{t}{2}\right)$ which means expansion of $\psi(t)$ by a factor of two. From Fig. 12.65, it is obvious that $\psi\left(\frac{t}{2}\right)$ exhibits less change when compared to $\psi(t)$. Thus, high scale corresponds to low frequency.

Thus, frequency and scale are inversely proportional to each other.

2. In STFT, the width of the window cannot be altered. What is the impact of choosing (i) smaller window, and (ii) a wider window?

A small window is effective to visualise sudden changes such as peaks or discontinuities. A wider window is effective for visualising slow variations. Depending upon the nature of the signal, the window has to be chosen in STFT. To analyse a low-frequency signal, a wider window is effective. To analyse a high-frequency signal, a smaller window is effective.

3. What are the advantages of DWT over DCT with respect to image compression?

- (i) DWT involves transformation of the whole image; hence there will not be any blocking artifact.
- (ii) DWT allows good localisation both in time and frequency domain. Also, multi-resolution is possible with DWT.
- (iii) In DCT, the basis function is fixed, whereas in DWT different basis functions are available depending upon the choice of the wavelet. Thus, DWT gives higher flexibility than DCT.

4. What is meant by progressive image compression?

Progressive image compression refers to the encoding of an image into a bit stream that can be parsed efficiently to obtain lower rates and lower resolution descriptions of the image.

5. Compare orthogonal wavelets with biorthogonal wavelets.

In the case of orthogonal filters, the wavelet filter (high-pass filter) and the scaling filter (low-pass filter) must be of the same length. This restriction is relaxed in biorthogonal systems. Parseval's theorem holds good in the case of orthogonal wavelet system, whereas it no longer holds in the biorthogonal wavelet systems. In the biorthogonal system, switching the roles of the primary and dual are valid.

6. What is a digital watermark and what is the need for digital watermark?

A digital watermark is an invisible mark embedded in a digital image which may be used for copyright protection.

Digital networks provide an efficient cost-effective means of distributing digital data. Unfortunately, digital networks afford virtually unprecedented opportunities to pirate copyrighted material. The idea of using a robust digital watermark is to detect and trace copyright violations.

7. Mention two differences between EZW and EBCOT coding.

The EZW coding exploits the interband dependencies of the wavelet coefficients, whereas in EBCOT coding the interband dependencies are not exploited. In EZW, there are two passes, namely, the *dominant pass* and the *refinement pass*; whereas in EBCOT coding there are three passes, namely, *significant pass*, *refinement pass* and *clean-up pass*.

Problems

- 12.1 Prove the Heisenberg uncertainty principle in signal processing.
 12.2 Calculate the Haar transform of the following image:

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

- 12.3 For the coefficients given in Fig. 12.66, find the bit stream generated by EZW coder. Also, decode the generated bit stream.

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

Fig. 12.66 Coefficients

- 12.4 For the coefficients shown in Fig. 12.66, find the bit stream generated by SPIHT coder. Also, decode the generated bit stream.
 12.5 What are the advantages of representing the signal in terms of contourlet basis?
 12.6 List the advantages of performing watermarking in the frequency domain.
 12.7 Derive the perfect reconstruction condition for a two-channel filter bank.
 12.8 Construct a fully populated approximation pyramid and corresponding prediction residual pyramid for the image

$$f(m, n) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}.$$

Use 2×2 block neighbourhood averaging for the approximation filter and omit the interpolation filter.

- 12.9 Compute the Haar transform $T = HFH^T$ of the 2×2 image $F = \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix}$. Compute also the inverse Haar transform $F = H^TTH$ of the obtained result.

References

Books

1. S Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, Second Edition, 1999
2. G Strang and TQ Nguyen, *Wavelets and Filter Banks*, Wellesley–Cambridge Press, Revised Edition, 1998
3. I Daubechies, *Ten Lectures on Wavelets*, SIAM, 1992
4. C Sidney Burrus, Ramesh A Gopinath, Haitao Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Upper Saddle River, NJ: Prentice Hall, 1998
5. A Akansu and R Haddad, *Multiresolution Signal Decomposition*, Academic Press, 1993
6. KP Soman, KI Ramachandran, *Insight into Wavelets from Theory to Practice*, Prentice Hall of India, Second Edition, 2006
7. Khalid Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers, 1996
8. David Solomon, *Data Compression—The Complete Reference*, Springer–Verlag, New York, 2004
9. Arne Jensen and Anders la Cour–Harbo, *Ripples in Mathematics—The Discrete Wavelet Transform*, Springer Verlag
10. GV Welland, *Beyond Wavelets*, Academic Press, 2003
11. Fritz Keinert, *Wavelets and Multiwavelets*, CRC Press, 2003
12. DS Taubman and MW Marcellin, *JPEG2000: Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Boston, 2002

Journal Articles

Introduction to Wavelet transform

1. Olivier Rioul and Martin Vetterli, *Wavelets and Signal Processing*, IEEE Signal Processing Magazine, pp. 14–38, October 1991

Embedded Quantisation Schemes

1. Shapiro JM *Embedded Image Coding using Zero Trees of Wavelet Coefficients*, IEEE Transactions on Signal Processing, vol. 41, No. 12, pp. 3445–3462, 1993
2. A Said and William A Pearlman, *A New, Fast, and Efficient Image coder based on Set Partitioning in Hierarchical Trees*, IEEE Transaction on Circuit and System for Video Technology, vol. 6, no. 3, pp. 243–250, June 1996

JPEG 2000 Standard

1. A Skodras, C Christopoulos, and T Ebrahimi, *The JPEG 2000 Still Image Compression Standard*, IEEE Signal Processing Magazine, vol. 18, pp. 36–58, 2001
2. D Taubman, *High-Performance Scalable Image Compression With EBCOT*, IEEE Transactions on Image Processing, vol. 9, July 2000

Image Pyramids

1. P Burt and E Adelson, *The Laplacian Pyramid as a Compact Image Code*, IEEE Trans. Comm., vol.31, no.5, pp. 532–540, 1983
2. MN Do and M Vetterli, *Framing Pyramids*, IEEE Trans. Signal Process., vol. 51, pp. 2329–2342, September 2003

Contourlet Transform

1. R H Bamberger and M J T Smith, *A Filter Bank for the Directional Decomposition of Images: Theory and Design*, IEEE Trans. Signal Process., vol. 40, no. 4, pp. 882–893, April 1992
2. M N Do and M Vetterli, *The Contourlet Transform: An Efficient Directional Multiresolution Image Representation*, IEEE Trans. Image Process., vol. 14, no. 12, pp. 2091–2106, December 2005

Denoising

1. DL Donoho, *Nonlinear Wavelet Methods for Recovery of Signals, Densities, and Spectra from Indirect and Noisy Data*, in Proc. of Symposia in Applied Mathematics, pp. 173–205, AMS, 1993
2. DL Donoho, *Denoising and Soft-thresholding*, IEEE Trans. Infor. Theory, vol. 41, pp. 613–627, 1995

Watermarking

1. C Voyatzin, N Nikolaidis and I Pitas, *Digital Watermarking: An Overview*, 9th European Signal Processing Conference (EUSIPCO 98).
2. N Kaewkamnerd and KR Rao, *Wavelet Based Image Watermarking Scheme*, EUSIPCO 2000, Tempere, Finland, September 2000

Web References

1. Robi Polikar's excellent introduction to the concepts of wavelet <http://users.rowan.edu/~polikar/>
2. <http://www.wavelet.org/> is a very good site to know the theory and application of wavelets.
3. William Pearlman's website is a treasure island for people working in the area of SPIHT: <http://www.cipr.rpi.edu/~pearlman/>
4. Minh Do's homepage gives rich information related to contourlets and directional decomposition: <http://www.ifp.uiuc.edu/~minhdo/publications/>
5. Martin Vetterli's homepage is a very useful source for a variety of wavelet information: <http://lcavwww.epfl.ch/~vetterli/>
6. Professor Deepa Kundur's research page contains good articles related to watermarking: <http://www.ece.tamu.edu/~deepa/pub.html>
7. For JPEG and JPEG2000 image compression standard the authors recommend the website www.jpeg.org/jpeg2000.

APPENDIX - I

Image Processing Related MATLAB Commands

MATLAB is a high-level technical computing environment suitable for solving scientific and engineering problems. There are many toolboxes available that extend the basic functions of MATLAB into different application areas. MATLAB contains a library of many useful functions for plotting and displaying the results of computation of various graphical forms. In this appendix, we have given a set of MATLAB commands that are commonly used in image processing.

Some useful MATLAB Image-Processing Commands

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	imread	Load an image
2	imwrite	Save an image
3	imshow	Display an image
4	imfinfo	Display image information
5	mean2	Compute global mean value of an image
6	std2	Compute the global standard deviation of an image
7	improfile	Select a line along which to take an intensity profile
8	imhist	Compute and display the image histogram
9	fspecial	Generate a predefined filter mask
10	filter2	Perform a 2D convolution
11	imadjust	Adjust the image intensity values

Commands related to image file information

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	dicominfo	Read metadata from a DICOM message
2	dicomread	Read a DICOM image
3	dicomwrite	Write a DICOM image
4	imfinfo	Return information about image file

Commands related to image type and type conversions

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	dither	Convert image using dithering
2	gray2ind	Convert intensity image to indexed image
3	grayslice	Create indexed image from intensity image by thresholding
4	graythresh	Compute global threshold using Otsu's method
5	im2bw	Convert image to binary image by thresholding
6	im2double	Convert image array to double precision
7	im2uint8	Convert image array to 8-bit unsigned integers

(Continued)

Commands related to image type and type conversions (Cont'd)

S.I. No.	Command	Function
8	im2uint16	Convert image array to 16-bit unsigned integers
9	ind2gray	Convert indexed image to intensity image
10	ind2rgb	Convert indexed image to RGB image
11	isbw	Return true for binary image
12	isgray	Return true for intensity image
13	isind	Return true for indexed image
14	isrgb	Return true for RGB image
15	mat2gray	Convert matrix to intensity image
16	rgb2gray	Convert RGB image to grayscale
17	rgb2ind	Convert RGB image to indexed image

Commands related to image arithmetic operations

S.I. No.	Command	Function
1	imabsdiff	Compute absolute difference of two images
2	imadd	Add a constant to image or adds two images
3	imcomplement	Complement image
4	imdivide	Divide two images, or divide image by constant
5	imlincomb	Compute linear combination of images
6	immultiply	Multiply two images, or multiply image by a constant
7	imsubtract	Subtract two images, or subtract constant from image

Commands related to Geometric Transformation

S.I. No.	Command	Function
1	checkerboard	Create checkerboard image
2	findbounds	Find output bounds for geometric transformation
3	fliptform	Flip the input and output roles of a TFORM struct
4	imcrop	Crop the image
5	imresize	Resize the input image
6	imrotate	Rotate the input image
7	imtransform	Apply geometric transformation to image
8	makeresampler	Create resampler structure
9	maketform	Create geometric transformation structure
10	tformarray	Apply geometric transformation to an ND array
11	tformfwd	Apply forward geometric transformation
12	tforminv	Apply inverse geometric transformation

Commands related to pixel values and statistics

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	corr2	Compute 2D correlation coefficient
2	imcontour	Create contour plot of image data
3	imhist	Display histogram of image data
4	impixel	Determine pixel colour values
5	improfile	Compute pixel value cross sections along line segments
6	mean2	Compute the mean of matrix elements
7	pixval	Display information about image pixels
8	std2	Compute standard deviation of matrix elements

Commands related to image transforms

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	fft2	Computes 2D fast Fourier transform of input matrix
2	fftshift	Reverse quadrants of output of fft
3	dftmtx	Creates discrete Fourier transform matrix
4	dctmtx	Creates discrete cosine transform matrix
5	dct2	Computes 2D discrete cosine transform of input matrix
6	idct2	Command to perform 2D inverse cosine transform
7	radon	Computes radon transform
8	iradon	Computes inverse radon transform
9	dwt2	Computes 2D wavelet transform of input image
10	idwt2	Computes the inverse wavelet transform

Commands related to image enhancement and restoration

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	histeq	Perform histogram equalisation of the input image
2	imadjust	Adjust the image intensity values
3	imnoise	Add noise to an image
4	medfilt2	Perform 2D median filtering
5	ordfilt2	Perform 2D order-statistic filter
6	stretchlim	Find limits to contrast stretch an image
7	wiener2	Perform 2D adaptive noise-removal filtering
8	fspecial	Create predefined filters
9	imfilter	Filter 2D images

Commands related to Morphological operations

S.I. No.	Command	Function
1	conndef	Default connectivity
2	imbothat	Perform bottom-hat filtering
3	imclearborder	Suppress light structures connected to image border
4	imclose	Close image
5	imdilate	Dilate image
6	imerode	Erode image
7	imextendedmax	Extended-maxima transform
8	imextendedmin	Extended-minima transform
9	imfill	Fill image regions and holes
10	imhmax	H-maxima transform
11	imhmin	H-minima transform
12	imimposemin	Impose minima
13	imopen	Open image
14	imreconstruct	Morphological reconstruction
15	imregionalmax	Regional maxima
16	imregionalmin	Regional minima
17	imtophat	Perform tophat filtering
18	watershed	Watershed transform

Useful commands related to Morphological Operations on Binary Image

S.I. No.	Command	Function
1	applylut	Perform neighbourhood operations using look-up table
2	bwarea	Compute area of objects in binary image
3	bwareaopen	Binary area open
4	bwdist	Compute distance transform of binary image
5	bweuler	Compute Euler number of binary image
6	bwhitmiss	Binary hit-miss operation
7	bwlabel	Label connected components in 2D binary image
8	bwlabeln	Label connected components in ND binary image
9	bwmorph	Perform morphological operations on binary image
10	bwpack	Pack binary image
11	bwperim	Determine perimeter of objects in binary image

(Continued)

Useful commands related to Morphological Operations on Binary Image (Cont'd)

S.I. No.	Command	Function
12	bwselect	Select objects in binary image
13	bwulterode	Ultimate erosion
14	bwunpack	Unpack binary image
15	makelut	Construct lookup table for use with applylut
16	strel	Create morphological structuring element

Commands related to region-based processing

S.I. No.	Command	Function
1	roicolor	Select region of interest, based on colour
2	roifill	Smoothly interpolate within arbitrary region
3	roifilt2	Filter a region of interest
4	roipoly	Select polygonal region of interest

Commands related to neighbourhood and block processing

S.I. No.	Command	Function
1	bestblk	Choose block size for block processing
2	blkproc	Implement distinct block processing for image
3	col2im	Rearrange matrix columns into blocks
4	colfilt	Columnwise neighbourhood operations
5	im2col	Rearrange image blocks into columns
6	nlfilter	Perform general sliding-neighbourhood operations

Commands related to colourmap manipulation

S.I. No.	Command	Function
1	brighten	Brighten or darken colourmap
2	cmpermute	Rearrange colours in colourmap
3	cmunique	Find unique colourmap colours and corresponding image
4	colormap	Set or get colour look-up table
5	imapprox	Approximate indexed image by one with few colours
6	rgbplot	Plot RGB colourmap components

Commands related to colourspace conversion

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	hsv2rgb	Convert HSV values to RGB colour space
2	ntsc2rgb	Convert NTSC values to RGB colour space
3	rgb2hsv	Convert RGB values to HSV colour space
4	rgb2ntsc	Convert RGB values to NTSC colour space
5	rgb2ycbcr	Convert RGB values to YCBCR colour space
6	ycbcr2rgb	Convert YCBCR values to RGB colour space

Useful commands related to filter banks and wavelets

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
1	dyaddown	Dyadic down sampling; if the input is a vector then the output is a down-sampled vector
2	dyadup	Dyadic upsampling. The output is an interpolated form of the input
3	qmfilter	Generates a quadrature mirror filter of the input
4	waveinfo	Gives information on wavelet families
5	wfilters	Computes the coefficients of the analysis and synthesis filters given a wavelet transform
6	dbwavf	This command computes the coefficients of the two-scale difference equation given a wavelet transform from the Daubechies family
7	orthfilt	Computes the coefficients of the analysis and the synthesis filters given the coefficients of the two-scale difference equation of an orthogonal wavelet
8	biorwavf	Computes the coefficients of the analysis and synthesis two-scale difference equations given the wavelet transform from the biorthogonal family
9	biorfilt	Computes the coefficients of the analysis and synthesis filters given the coefficients of the analysis and synthesis two-scale difference equations of a biorthogonal family
10	dwt	Single-stage discrete one-dimensional wavelet decomposition
11	idwt	Single-stage discrete one-dimensional wavelet reconstruction
12	dwtmode	Sets the type of signal extension at the signal boundaries for wavelet transform calculation

(Continued)

Useful commands related to filter banks and wavelets (Cont'd)

<i>S.I. No.</i>	<i>Command</i>	<i>Function</i>
13	wavedec	Performs multiple-stage one-dimensional wavelet decomposition
14	waverec	Performs multiple-stage one-dimensional wavelet reconstruction
15	upwlev	Performs single-stage reconstruction of a multiple-stage one-dimensional wavelet decomposition.
16	wavefun	Generates approximations of wavelets and scaling functions given a wavelet transform

References

1. Mathworks website: http://www.mathworks.com/products/product_listing/index.html
2. Rafael C Gonzalez, Richard E Woods and Steven L Eddins, "Digital Image Processing using MATLAB," Prentice Hall, 2004
3. Image Processing Toolbox: <http://www.mathworks.com/access/helpdesk/help/toolbox/images/>
4. The reader should not miss the content given in this website:
http://www.imageprocessingplace.com/root_files_V3/review_materials.htm

APPENDIX – II

Overview of Vector Space Concepts

Vector A vector is an ordered collection of N numbers. The numbers are called the components of the vector, and N is the dimensionality of the vector. The column vector is given by

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

Norm of a Vector Vector norm measures the size of the vector. In a vector space, the norm of a vector x is denoted by $|x|$, and has the following properties:

- (a) $|x| > 0$ (b) If a is a number then $|ax| = |a||x|$ (c) $|x + y| \leq |x| + |y|$

Linearly Independent Vectors The set of vectors $v_1, v_2, \dots, v_k \in R^n$ is linearly independent if

$$c_1 v_1 + c_2 v_2 + \dots + c_k v_k = 0$$

which implies that $c_1 = c_2 = \dots = c_k = 0$. Otherwise, this set is linearly dependent. In other words, a set of vectors are linearly independent if none of its elements can be written as a linear combination of the remaining ones.

Example 1: Determine whether the vectors $v_1 = [1 0 0]^T; v_2 = [0 1 0]^T; v_3 = [0 0 1]^T$ are linearly independent or not.

Solution First, form a matrix using the given vectors. Let A be the matrix formed using the three vectors v_1, v_2 and v_3 which is given by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, compute the determinant of the matrix A . If the determinant is not zero then the vectors are linearly independent. If the determinant is zero then the vectors are linearly dependent. In our case the determinant is given by

$$|A| = 1$$

The determinant is not equal to zero. Hence the vectors are linearly independent.

Example 2: Determine whether the vectors are linearly independent.

$$V = \left\{ \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 5 \\ 7 \end{bmatrix} \right\}$$

Solution In this example, one of the vectors is a null vector. Hence, the vectors are linearly dependent. This can be checked by determining the determinant value which will be equal to zero.

Inner Product The inner product between two vectors U and V is denoted by $\langle u, v \rangle = u^T v$. If the inner product $\langle U, V \rangle = 0$ then the vectors U and V are orthogonal to each other. The inner product should satisfy the following four properties:

- (i) $\langle u, v \rangle = \langle v, u \rangle$ for all $u, v \in V$
- (ii) $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$ for all $u, v, w \in V$
- (iii) $\langle cu, v \rangle = c \langle u, v \rangle = \langle u, cv \rangle$ for all $u, v \in V$ and all $c \in R$
- (iv) $\langle u, u \rangle \geq 0$ for $u \in V$, and $\langle u, u \rangle = 0$ iff $u = 0$

Example 3: Find the inner product between the two vectors $U = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $V = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and find whether they are orthogonal to each other or not.

Solution The inner product between U and V is denoted by $\langle U, V \rangle = U^T V$

In this problem, $\langle U, V \rangle = [1 \ 0] \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$. Since the inner product is equal to zero, the vectors are orthogonal to each other.

Norm from Inner Product

The relationship between the norm of the vector and the inner product is given by

$$\|v\| = \sqrt{\langle v, v \rangle}$$

Properties of Norm

- (i) $\|v\| \geq 0$ for all $v \in V$
- (ii) $\|v\| = 0$ if and only if $v = 0$
- (iii) $\|cv\| = |c| \|v\|$ for all $c \in R$ and all $v \in V$
- (iv) $\|u + v\| \leq \|u\| + \|v\|$ for all $u, v \in V$ which is termed the triangle inequality.

Zero Vector A vector having all components as zero is called a zero vector. In other words, a vector of zero length is termed a zero vector. Every linear space contains a zero vector.

Unit Vector A unit vector is a vector of unit length.

Orthogonal Vector Two vectors are orthogonal if their inner product is equal to zero.

Example 4: Let us consider two vectors $v_1 = \begin{bmatrix} \frac{1}{\sqrt{2}}, & \frac{1}{\sqrt{2}} \end{bmatrix}^T$ and $v_2 = \begin{bmatrix} \frac{1}{\sqrt{2}}, & -\frac{1}{\sqrt{2}} \end{bmatrix}^T$. Determine whether the vectors are orthogonal or not.

Solution First, we have to determine the inner product between the two vectors. The inner product is given by

$$\langle v_1, v_2 \rangle = v_1^T \times v_2 = \begin{bmatrix} \frac{1}{\sqrt{2}}, & \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = 0$$

The inner product is zero, and hence the vectors are orthogonal to each other.

Vector Space

A vector space V is a collection of vectors which obeys the following properties:

- (i) For $u, v \in V$, $u + v = v + u$. This property is known as commutative property.
- (ii) $u, v, w \in V$, $(u + v) + w = u + (v + w)$. This property is known as associative property.
- (iii) There exists a vector $0 \in V$, such that $u + 0 = u$ for all $u \in V$.
- (iv) For every $u \in V$, there is a $-u \in V$ such that $u + (-u) = 0$.
- (v) $1u = u$ for all $u \in V$.
- (vi) $\alpha(\beta u) = (\alpha\beta)u$ for all α, β belonging to the field of real numbers R or the field of complex numbers C .
- (vii) $(\alpha + \beta)u = \alpha u + \beta u$
- (viii) $\alpha(u + v) = \alpha u + \alpha v$

Dimension of the Vector Space The dimension of the vector space is the smallest number of elements necessary to span the vector space.

Subspace A subspace of a vector space is a non-empty subset of vectors that is closed under vector addition and scalar multiplication.

The following two constraints should be satisfied for a subspace:

- (a) The sum of any two vectors in the subspace remains in the subspace.
- (b) Multiplication of any vector by a scalar yields a vector in the subspace.

The different subspaces are (i) row space, (ii) column space, and (iii) null space.

Basis A linearly independent set of vectors in a vector space which spans the whole space is called the basis of the vector space.

Orthogonal and orthonormal bases A basis is orthogonal if its vectors are pairwise orthogonal. A basis is orthonormal if it is orthogonal and all vectors have unit length.

An example of orthonormal basis is $[e_1, e_2, e_3]$

where $e_1 = [1, 0, 0]^T$; $e_2 = [0, 1, 0]^T$; $e_3 = [0, 0, 1]^T$.

Schwarz Inequality Let V be an inner product space and $u, v \in V$. Then

$$|(u, v)| \leq \|u\| \|v\|$$

The equality condition holds good if and only if u, v are linearly independent.

Inner-Product Space

A vector space with an inner product defined on it is called an inner-product space. The following axioms are satisfied by the inner product for all vectors u, v and w in V .

(i) **Symmetry condition** The symmetry condition is given by

$$\langle u, v \rangle = \langle v, u \rangle$$

(ii) **Additivity** The additivity axiom is given by

$$\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$$

(iii) **Homogeneity condition** The homogeneity condition is mathematically represented as

$$\langle ku, v \rangle = k \langle u, v \rangle \text{ where } k \text{ is a scalar.}$$

(iv) **Positivity condition** The positivity condition is given by

$$\langle v, v \rangle \geq 0 \quad \text{and} \quad \langle v, v \rangle = 0 \text{ if and only if } v = 0$$

Hilbert space A Hilbert space is an inner-product space for which every Cauchy sequence in the norm converges to an element of that space.

References

1. Gilbert Strang, *Linear Algebra and its Applications*, Thomson Brooks/Cole international student edition, 2005
2. David C Lay, *Linear Algebra and its Applications*, Pearson Education, Third edition
3. Howard Anton, *Elementary Linear Algebra*, John Wiley & Sons, 2000

APPENDIX - III

Fundamentals of Matrices

This appendix presents an introduction to matrices to support the material of the textbook. The conventions followed in this appendix are given below:

Symbol	Meaning
*	Conjugate
,	Transpose
†	Conjugate transpose
-1	Inverse

Matrix A matrix is a two-dimensional array of real or complex numbers. An $m \times n$ matrix is generally represented as

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

The shorthand notation to represent the matrix is $A = [a_{ij}]$, where a_{ij} denotes the element in the i^{th} row and j^{th} column. Here, the first subscript indexes the row in which the element lies, and the second subscript indexes the column in which the element lies.

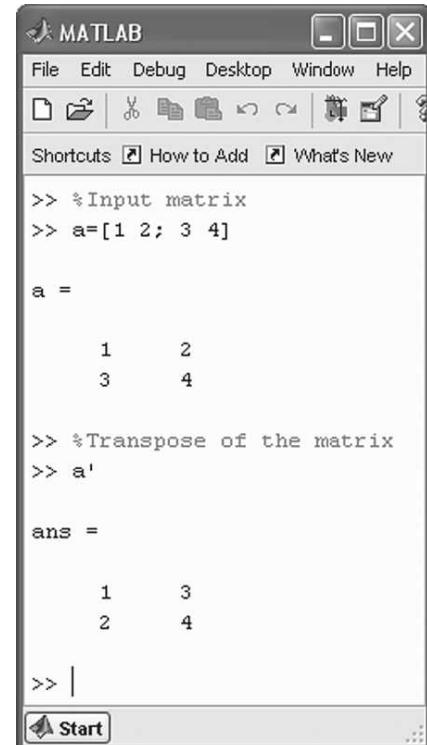
Basic Operations

(i) *Transpose of a Matrix* The transpose of a matrix is done by interchanging the rows and columns of the matrix. The transpose operation is represented as

$$[A^T]_{ij} = a_{ji} \text{ if } [A]_{ij} = a_{ij}$$

In MATLAB, the separator character (') is used to get the transpose of a matrix. The MATLAB command to get the transpose of a matrix and the corresponding output are shown in Fig. A3.1.

An image can be considered as a matrix; hence the transpose operation can be extended to images. The MATLAB command which is used to transpose an image is shown in Fig. A3.2 and the corresponding output is shown in Fig. A3.3.



```

MATLAB
File Edit Debug Desktop Window Help
File Edit Desktop Window Help
Shortcuts How to Add What's New
>> %Input matrix
>> a=[1 2; 3 4]

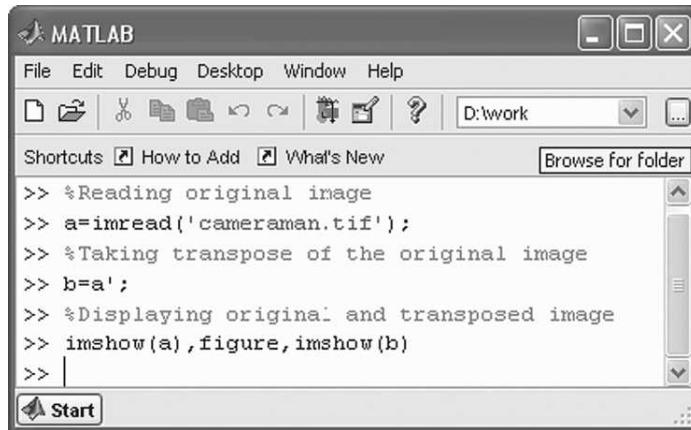
a =
1 2
3 4

>> %Transpose of the matrix
>> a'

ans =
1 3
2 4

```

Fig. A3.1 Transpose of a matrix



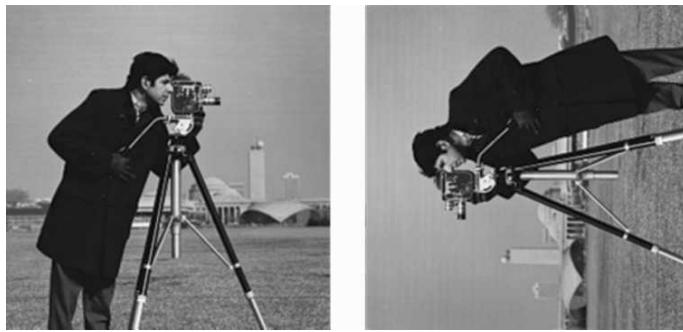
The image shows a screenshot of the MATLAB graphical user interface. The title bar says 'MATLAB'. The menu bar includes 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. Below the menu is a toolbar with various icons. The current directory is set to 'D:\work'. The command window contains the following MATLAB code:

```

>> %Reading original image
>> a=imread('cameraman.tif');
>> %Taking transpose of the original image
>> b=a';
>> %Displaying original and transposed image
>> imshow(a),figure,imshow(b)
>>

```

Fig. A3.2 MATLAB command to transpose an image



(a) Original image

(b) Image after transpose operation

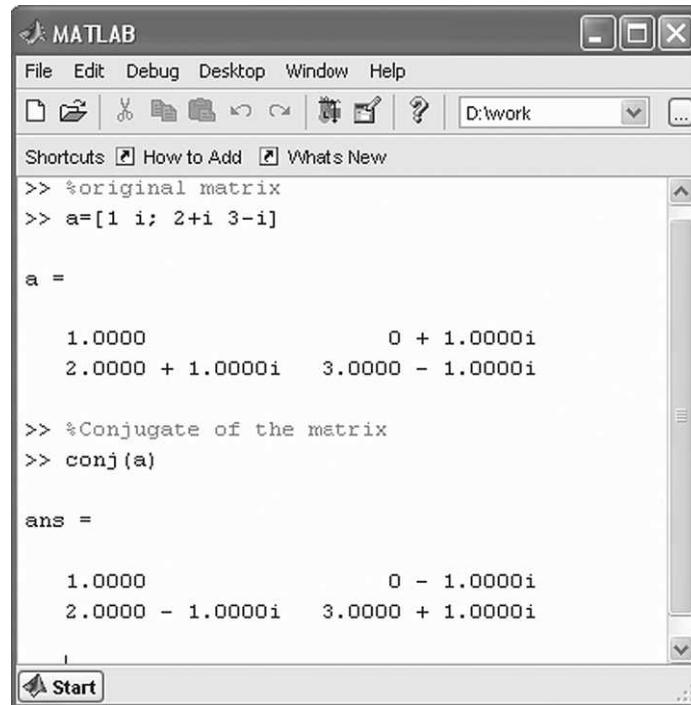
Fig. A3.3 Result of transpose operation

(ii) *Conjugate of a Matrix* The conjugate of a complex matrix is obtained by replacing each entry of the matrix by its conjugate. The conjugate of the matrix is mathematically represented by

$$\left[A^* \right]_{ij} = \bar{a}_{ij}$$

The MATLAB command which is used to obtain the conjugate of a matrix is given by `conjugate()`. The MATLAB command and the corresponding output are shown in Fig. A3.4.

(iii) *Conjugate of the Transpose Matrix (or) Hermitian Transpose* The conjugate transpose is obtained by combining matrix transposition with element-wise conjugation. The conjugate transpose is also referred as the Hermitian transpose.



The image shows a screenshot of the MATLAB graphical user interface. The title bar says 'MATLAB'. The menu bar includes 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. The toolbar has icons for file operations like Open, Save, and Print. The current directory is set to 'D:\work'. The command window displays the following MATLAB code and its output:

```

>> %original matrix
>> a=[1 i; 2+i 3-i]

a =

    1.0000          0 + 1.0000i
    2.0000 + 1.0000i  3.0000 - 1.0000i

>> %Conjugate of the matrix
>> conj(a)

ans =

    1.0000          0 - 1.0000i
    2.0000 - 1.0000i  3.0000 + 1.0000i

```

Fig. A3.4 Conjugate of a matrix

Example 1 Determine the Hermitian transpose of the matrix

$$A = \begin{bmatrix} -1+j3 & 4 \\ 0 & j \\ 1-j & 2 \\ 1+j5 & -j \end{bmatrix}$$

Solution The Hermitian transpose is obtained by first taking the transpose and then taking the conjugate.

Step 1 Transpose of the input matrix

The transpose is obtained by interchanging the rows and columns of the input matrix.

$$A^T = \begin{bmatrix} -1+j3 & 0 & 1-j & 1+j5 \\ 4 & j & 2 & -j \end{bmatrix}$$

Step 2 Conjugate of the matrix obtained in Step 1

$$(A^T)^* = \begin{bmatrix} -1-j3 & 0 & 1+j & 1-j5 \\ 4 & -j & 2 & j \end{bmatrix}$$

The conjugate of the real number is equal to the same number.

Note For $n \times n$ matrices A and B , the following relationship holds good.

$$(AB)^T = B^T A^T$$

(iv) *Trace of the Square Matrix* The trace of the square matrix is the sum of its diagonal elements which is mathematically represented as

$$tr(A) = \sum_{i=1}^n a_{ii}$$

Properties of Trace The following properties hold good with respect to trace of a matrix:

(i) $tr(AB) = tr(BA)$

(ii) $tr(A + B) = tr(A) + tr(B)$

(iii) $tr(A^T B) = \sum_{i=1}^n \sum_{j=1}^n [A]_{ij} [B]_{ij}$

Example 2 Let $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. Determine the trace of the matrix.

Solution The given matrix is a square matrix. The trace of the matrix is obtained by adding the diagonal elements of the matrix. In this case, the trace is $1 + 4 = 5$. The MATLAB command to obtain the trace of the matrix is shown in Fig. A3.5.

(v) *Determinant of a Square Matrix* The determinant of a square matrix A is denoted by $\det(A)$. The determinant of the 2×2 square matrix is given by

$$\det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

For an $n \times n$ square matrix, the determinant is given by

$$\det(A) = \sum_{k=1}^n (-1)^{i+k} a_{ik} \det(A_{ik}) = \sum_{k=1}^n (-1)^{k+j} a_{kj} \det(A_{kj})$$

for any $1 \leq i, j \leq n$ where A_{pq} is the $(n-1) \times (n-1)$ matrix resulting from the deletion of the p^{th} row and the q^{th} column of A .

The properties of determinants are given below:

- (i) $\det(AB) = \det(A) \det(B)$
- (ii) $\det(\alpha A) = \alpha^n \det(A)$ for any scalar α and an $n \times n$ matrix A
- (iii) $\det(A^T) = \det(A)$
- (iv) $\det(A^k) = (\det(A))^k$

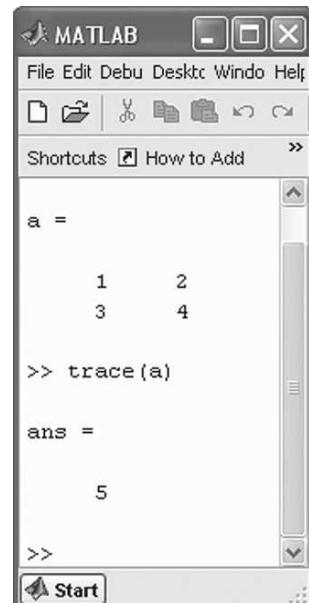


Fig. A3.5 Trace of a matrix

- (v) $\det(A) = 0$ if any row of A is a scalar multiple of another row
- (vi) $\det(A) = 0$ if any row of A is zero
- (vii) If B is obtained from A by interchanging two rows then $\det(B) = -\det(A)$.

Note A matrix is singular if the determinant value is equal to zero. A square matrix A is said to be non-singular if $\det(A) \neq 0$.

(vi) *Matrix Inversion* An $n \times n$ matrix A is invertible if there is another $n \times n$ matrix B that satisfies the relationship

$$AB = BA = I_n$$

In such cases, B is called the inverse of A^{-1} . If A has an inverse, the inverse is unique. If A has no inverse then A is called singular. The inverse of an invertible matrix is given by the formula

$$A^{-1} = \frac{\text{adj}(A)}{|A|}$$

The adjoint of A is given by

$$\text{adj}(A) = [C_{ij}]^T$$

where C_{ij} is the cofactor of the (i, j) th element of A .

Note It is not always possible to obtain the inverse of a matrix. The inverse of a matrix exists if and only if the matrix is non-singular. If the matrix happens to be a singular matrix then the choice is pseudo-inverse rather than the true inverse.

(vii) *Rank of a Matrix* The rank of an $m \times n$ matrix A , denoted by $\text{rank}(A)$, is the largest number of columns (or rows) of A that form a set of linearly independent vectors.

Example 3 Determine the rank of the matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 5 & 9 \\ 2 & 4 & 6 \end{bmatrix}$.

Solution The rank of the matrix is two, because the first and the last row are dependent on each other, whereas the first two rows are independent of each other. Hence the rank is two.

The MATLAB command to determine the rank of a matrix is `rank`. The MATLAB command and the corresponding output is shown in Fig. A3.6.

The rank of a matrix has the following properties:

- (i) An $n \times n$ matrix A is non-singular if and only if $\text{rank}(A) = n$.
- (ii) $\text{Rank}(AB) \leq \min\{\text{rank}(A), \text{rank}(B)\}$
- (iii) $\text{Rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$
- (iv) The number of singular values is equal to the rank of the matrix.

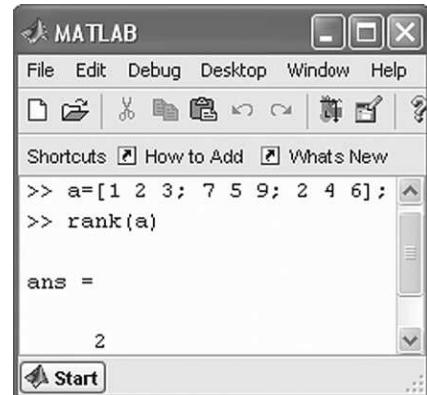


Fig. A3.6 Rank of the matrix

Kronecker Product Let $A = [a_{ij}]_{mn}$ and $B = [b_{kl}]_{pq}$. The Kronecker product of A and B denoted by $A \otimes B$ is an $mp \times nq$ matrix defined by $A \otimes B = [a_{ij}B]$.

Example 4 Let $A = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. Determine the Kronecker product between A and B .

$$A \otimes B = \begin{bmatrix} 2 & 4 & 4 & 8 & 6 & 12 \\ 6 & 8 & 12 & 16 & 18 & 24 \\ 1 & 2 & 3 & 6 & 5 & 10 \\ 3 & 4 & 9 & 12 & 15 & 20 \end{bmatrix}$$

The MATLAB command which is used to compute the Kronecker product is `kron`. The MATLAB command and the corresponding output is illustrated in Fig. A3.7.

Orthogonal condition An $n \times n$ square matrix is orthogonal if $A^{-1} = A^T$.

Example 5 Prove that the matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

is orthogonal.

Step 1 Determine the transpose of the matrix.

$$A^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 2 Determine the inverse of the matrix.

$$A^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From steps 1 and 2, we find that $A^{-1} = A^T$. Hence, the matrix A is orthogonal.

```

>> a
a =
2     4     6
1     3     5

>> b
b =
1     2
3     4

>> kron(a,b)
ans =
2     4     4     8     6     12
6     8     12    16    18    24
1     2     3     6     5     10
3     4     9     12    15    20

```

Fig. A3.7 Kronecker product

Special Matrices

(i) *Column Matrix* A matrix having only one column but more than one row is called a column matrix or column vector. An example of a column matrix is given below.

$$A = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix}$$

(ii) *Row Matrix* A matrix having only one row but more than one column is called a row matrix.

$$A = [a_{11} \ a_{12} \ \cdots \ a_{n1}]$$

(iii) *Upper Triangular Matrix* The upper triangular matrix has all zeros below the main diagonal. An example of an upper triangular matrix is given below.

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 0 & 1 & 3 \\ 0 & 0 & 7 \end{bmatrix}$$

(iv) *Lower Triangular Matrix* The lower triangular matrix has all zeros below the main diagonal. An example of a lower triangular matrix is given below.

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 3 & 0 \\ 3 & 2 & 5 \end{bmatrix}$$

Note The inverse of upper and lower triangular matrices are also upper and lower triangular in nature.

(v) *Square Matrix* A matrix that has an equal number of rows and columns is called a square matrix.

(vi) *Diagonal Matrix* A square matrix that has all elements except the diagonal elements equal to zero is called a diagonal matrix.

$$\text{An example of a diagonal matrix is } A = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}.$$

The MATLAB command that is used to obtain a diagonal matrix is `diag` which is shown in Fig. A3.8.

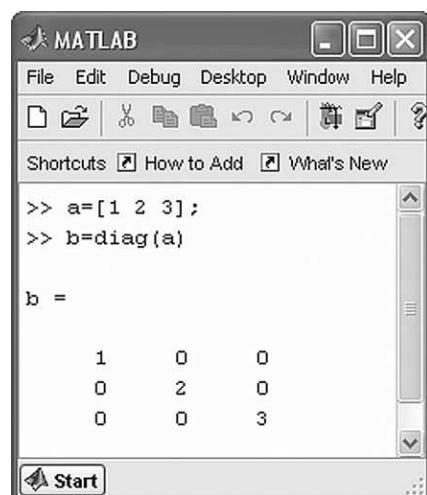
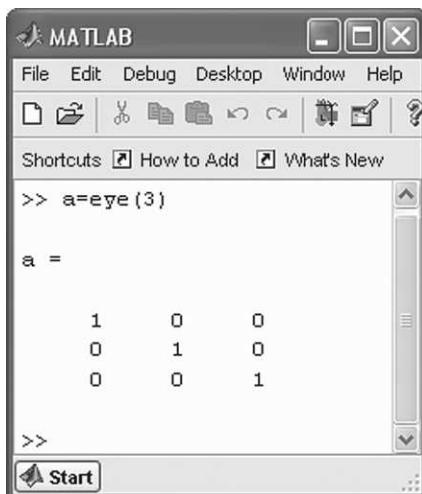


Fig. A3.8 *Diagonal matrix*

(vii) *Identity Matrix or Unit Matrix* A diagonal matrix that has all the diagonal elements equal to unity is called an identity matrix.

The MATLAB command to create an identity matrix is `eye` which is shown in Fig. A3.9.

The MATLAB command `eye` can be used to create an image which is shown in Fig. A3.10.



The image shows the MATLAB Command Window. The user has entered the command `>> a=eye(3)`. The resulting matrix `a` is displayed as:

```

a =
1 0 0
0 1 0
0 0 1

```

Fig. A3.9 Identity matrix

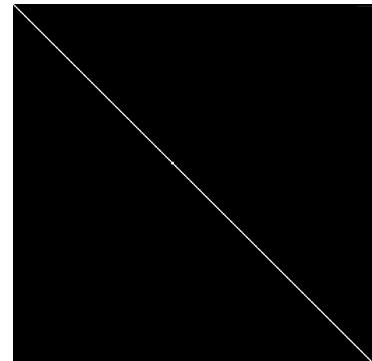


Fig. A3.10 Simple image

The MATLAB command that is used to create the image shown in Fig. A3.10 is shown in Fig. A3.11.

(viii) *Symmetric Matrix* An $n \times n$ square matrix is symmetric if $A^T = A$. In other words, $a_{ij} = a_{ji}$.

An example of a symmetric matrix is $A = \begin{bmatrix} 1 & 8 & 10 \\ 8 & 2 & 9 \\ 10 & 9 & 3 \end{bmatrix}$

In the matrix A , we find that $a_{ij} = a_{ji}$. Hence, the matrix is symmetric.

(ix) *Skew-symmetric Matrix* A matrix is skew-symmetric if $A^T = -A$. An example of a skew symmetric matrix is given below.

$$A = \begin{bmatrix} 0 & -5 & 4 \\ 5 & 0 & -9 \\ -4 & 9 & 0 \end{bmatrix}.$$

(x) *Hermitian Matrix* A matrix is Hermitian if it is the same as its adjoint.

`a=eye(256);
imshow(a)`

Fig. A3.11 MATLAB command to create a line image

(xi) *Hankel Matrix* A Hankel matrix has the property that the elements along every cross diagonal are equal. Thus,

$$a_{ij} = a_{i+j-N-1}$$

The Hankel matrix is represented by

$$A = \begin{bmatrix} a_{-3} & a_{-2} & a_{-1} & a_0 \\ a_{-2} & a_{-1} & a_0 & a_1 \\ a_{-1} & a_0 & a_1 & a_2 \\ a_0 & a_1 & a_2 & a_3 \end{bmatrix}$$

(xi) *Unitary Matrix* A matrix is unitary if its adjoint is the same as its inverse. The condition for a matrix to be unitary is

$$AA^\dagger = I$$

where I is the identity matrix and ‘ \dagger ’ indicates the conjugate transpose of A .

Note A unitary matrix always has a determinant value of one. Also, in order for the matrix A to be unitary, the columns must be orthonormal.

An example of unitary matrix is given below

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

All the entries in the matrix A are real. To prove that the matrix is unitary, take the transpose of A . Then multiply the transpose of the matrix A with the matrix A . If the result is an identity matrix then the matrix A is unitary.

Step 1 Transpose of the matrix ‘A’

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

Step 2 Multiplication of the matrix A with A^T

$$A \times A^T = I$$

$$A \times A^T = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Since the result is an identity matrix, the matrix A is a unitary matrix.

The properties of important matrices are summarised in Table 1.

Table 1 Properties of matrices

Matrix	Properties
Real matrix	$A^* = A$
Symmetric matrix	$A' = A$
Orthogonal matrix	$A' = A^{-1}$
Unitary matrix	$A^\dagger = A^{-1}$

(x) *Toeplitz Matrix* An $n \times n$ Toeplitz matrix is given by

$$[A]_{ij} = a_{i-j}$$

The Toeplitz matrix is given by

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \cdots & a_{-(n-2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix}$$

If $a_{-k} = a_k$, then A is symmetric Toeplitz.

The MATLAB command to generate a Toeplitz matrix is `toeplitz`. The MATLAB command and the corresponding output are shown in Fig. A3.12.

Note A Toeplitz matrix has the property that all the elements along each diagonal are identical.

(xi) *Circulant Matrix* The general form of a circulant matrix is given by

$$C_n = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-(n-1)} \\ t_{-(n-1)} & t_0 & t_{-1} & & \\ t_{-(n-2)} & t_{-(n-1)} & t_0 & & \\ \vdots & & & \ddots & \vdots \\ t_{-1} & t_{-2} & \cdots & & t_0 \end{bmatrix}$$

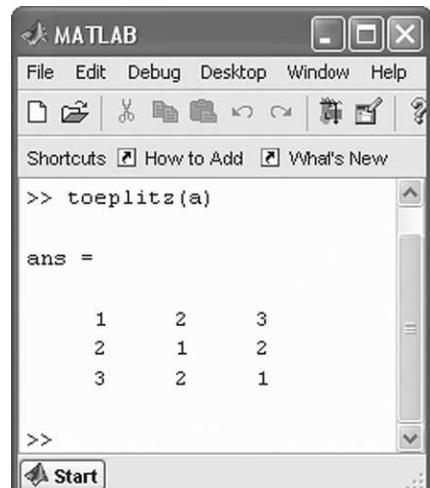


Fig. A3.12 Toeplitz matrix generation

A circulant matrix is a special kind of Toeplitz matrix where each column is obtained by doing wrap-around downshift of the previous column. A circulant matrix finds application in discrete Fourier transform, and study of cyclic codes.

Circulant matrices have the following properties:

- (i) The matrix C is circulant if and only if C^\dagger is circulant.
- (ii) If C_1 and C_2 are circulant then $C_1 C_2$ is circulant.
- (iii) Circulant matrices are normal matrices. That is, $C^T C = C C^T$.

Note Circulant matrices are also Toeplitz matrices.

(xii) *Vandermonde Matrix* An $n \times n$ Vandermonde matrix is a matrix of the form

$$V = \begin{bmatrix} 1 & x_1 & \dots & x_1^{n-2} & x_1^{n-1} \\ 1 & x_2 & \dots & x_2^{n-1} & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & \dots & x_n^{n-2} & x_n^{n-1} \end{bmatrix}$$

The matrix V depends on the elements x_1, \dots, x_n .

Note The transpose of a Vandermonde matrix is also a Vandermonde matrix. An example of a Vandermonde matrix is the discrete Fourier transform matrix.

Built-in Functions to Generate Matrices

The set of built-in functions to generate matrices is shown in Table 2.

Table 2 *Built-in MATLAB commands to generate matrices*

S.I. No.	Command	Description
1	eye	Identity matrix
2	zeros	Matrix of zeros
3	ones	Matrix of ones
4	diag	Create or extract diagonals
5	triu	Upper triangular part of matrix
6	tril	Lower triangular part of matrix
7	rand	Randomly generated matrix
8	hilb	Hilbert matrix
9	magic	Magic square
10	Toeplitz	Toeplitz matrix

References

1. R M Gray, *Toeplitz and Circulant Matrices: A Review*
2. R Bellman, *Introduction to Matrix Analysis*, McGraw Hill, New York, 1972

APPENDIX - IV

Objective Type Questions

1. The third bit-plane corresponding to the image $\begin{bmatrix} 4 & 3 \\ 5 & 2 \end{bmatrix}$ is
- (a) $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$
2. The 2D DFT of the image $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ is
- (a) $\begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} 0 & 4 \\ 0 & 0 \end{bmatrix}$ (c) $\begin{bmatrix} 0 & 0 \\ 4 & 0 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$
3. The transform which possesses the 'multi-resolution' property is
- (a) Fourier transform (b) short-time Fourier transform
(c) cosine transform (d) wavelet transform
4. The transform which is widely used to detect 'lines' in an image is
- (a) Fourier transform (b) Hough transform
(c) cosine transform (d) Haar transform
5. The transform which possesses the highest 'energy compaction' property is
- (a) Fourier transform (b) Walsh transform
(c) slant transform (d) KL transform
6. The condition for the transform A to be unitary is (*denotes conjugate and T denotes transpose)
- (a) $A^{-1} = \frac{1}{A}$ (b) $A^{-1} = A^{*T}$ (c) $A^{-1} = \frac{1}{A^{*T}}$ (d) $A^{-1} = \frac{1}{\det(A)}$
7. Statement 1: All prefix codes are uniquely decodable.
Statement 2: All uniquely decodable codes must be prefix codes.
- (a) Statement 1 and Statement 2 are always true.
(b) Statement 1 is always true whereas Statement 2 is not always true.
(c) Both statements 1 and 2 are wrong.
(d) Statement 2 is always true and Statement 1 is not always true.
8. Which one of the following is a lossy coding?
- (a) Run-length coding (b) Uniform quantiser
(c) Huffman coding (d) Predictive coding without quantiser
9. In a DPCM coder, which of the following needs to be quantised?
- (a) The reconstruction value (b) The difference between prediction value and the original value
(c) The prediction value (d) The transform coefficient
10. What does the definition of entropy tell us?
- (a) The lower bound to encode a source without distortion
(b) The upper bound to encode a source without distortion
(c) The average number of bits to encode a source without distortion
(d) The average number of bits to encode a source given a certain distortion
11. In an image compression system, 16384 bits are used to represent a 128×128 image with 256 gray levels. What is the compression ratio for this system?
- (a) 4 (b) 8 (c) 12 (d) 16

12. For an eight-bit image $x[m, n]$, the transformation $y[m, n] = 255 - x[m, n]$ will yield a/an
 (a) dark image (b) bright image
 (c) negative of the input image (d) output image same as the input image
13. In Huffman coding, the size of the codebook is L_1 , while the longest code word can have as many as L_2 bits. What is the relationship between L_1 and L_2 ?
 (a) $L_1 < L_2$ (b) $L_1 > L_2$ (c) $L_1 = L_2$ (d) No relationship
14. The relationship between current threshold T_i and the previous threshold T_{i-1} in different passes in EZW is given by
 (a) $T_i = \frac{1}{2}T_{i-1}$ (b) $T_i = \frac{1}{4}T_{i-1}$ (c) $T_i = \frac{1}{8}T_{i-1}$ (d) $T_i = \frac{1}{16}T_{i-1}$

15. A seven-level decomposition is shown below:

16	7	12	10
-7	8	8	4
4	-3	4	-3
2	-2	-2	0

The initial threshold using EZW algorithm is

- (a) 8 (b) 16 (c) 32 (d) 64
16. Statement 1: A histogram gives the frequency of occurrence of the gray level.
 Statement 2: A histogram is invariant to rotation.
 (a) Statements 1 and 2 are wrong. (b) Statement 1 is correct and Statement 2 is wrong.
 (c) Statements 1 and 2 are correct. (d) Statement 2 is correct and Statement 1 is wrong.
17. The transform used in JPEG image compression is
 (a) discrete cosine transform (b) Walsh transform
 (c) Hadamard transform (d) KL transform
18. The number of bits necessary to represent a 256×256 image with 256 gray level is
 (a) 524288 (b) 324288 (c) 224288 (d) 124288
19. The linear convolution between the matrices $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ results in
 (a) $\begin{bmatrix} 2 & 1 & 0 \\ 3 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & 2 & 0 \\ 4 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ (d) $\begin{bmatrix} 4 & 3 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
20. Comparing geometrical zonal coding with threshold coding, for the same number of transmitted samples, which of the following is not correct?
 (a) Threshold coding has more distortions.
 (b) Threshold coding mask gives a better choice of transmission samples.
 (c) Threshold coding needs more rates.
 (d) In threshold coding, the addresses of the transmitted samples have to be coded for every image block.

- 21. Statement 1: A median filter is effective in minimising salt-and-pepper noise in an image.**
Statement 2: A median filter is a linear filter.
- (a) Statement 1 is true whereas Statement 2 is wrong.
 (b) Statements 1 and 2 are true.
 (c) Statements 1 and 2 are false.
 (d) Statement 1 is false whereas Statement 2 is correct.
- 22. The wiener filter behaves like an inverse filter**
- (a) in the presence of noise
 (b) in the absence of noise
 (c) in the presence of noise and degradation
 (d) Wiener filter will not resemble inverse filter under any condition
- 23. The logarithm of a Fourier transform is generally termed**
- (a) spectrum (b) power spectrum
 (c) cepstrum (d) energy spectrum
- 24. Dilation followed by erosion operation leads to**
- (a) closing operation (b) opening operation
 (c) union operation (d) intersection operation
- 25. Erosion followed by dilation operation leads to**
- (a) opening operation (b) closing operation
 (c) intersection operation (d) union operation
- 26. The output of an STFT is generally termed**
- (a) mammogram (b) spectrogram
 (c) scalogram (d) hologram
- 27. The photoreceptors that are responsible for colour vision are**
- (a) rods (b) cylinders
 (c) ciliary muscles (d) cones
- 28. The technique in which uniform threshold is applied uniformly throughout the wavelet tree is**
- (a) VisuShrink (b) LevelShrink
 (c) SureShrink (d) BayesShrink
- 29. The transform that is effective in detecting lines in an image is**
- (a) Fourier (b) Hough
 (c) Wavelet (d) Hadamard
- 30. Below are the pixel values in a 5×5 gray-level image:**

1	2	3	1	2
4	5	2	3	3
3	3	5	4	4
1	3	2	3	5
2	1	3	1	3

What will be the value of the marked pixel after applying a 3×3 mode filter?

- (a) 2 (b) 3 (c) 4 (d) 5

31. Which of the following operations is idempotent?

 - (a) Dilation
 - (b) Erosion
 - (c) Convolution
 - (d) Closing

32. Which of the following operations is idempotent?

 - (a) Median filter
 - (b) Dilation
 - (c) Opening
 - (d) Erosion

33. Below are the pixel values in a 5×5 gray-level image:

1	2	3	1	2
4	5	2	3	3
3	3	5	4	4
1	3	2	3	5
2	1	3	1	3

What is the value of the marked pixel after applying a 3×3 median filter?

34.	1	1	2	2	4
	4	2	4	5	6
	3	3	3	1	4
	6	4	1	3	5
	4	2	2	1	2

The discrete Fourier transform is performed on the above image. What will be the result in $F(0,0)$?

35. Which of the following filters will in general have the best performance in enhancing edges in an image?

 - (a) Mean filter
 - (b) Median filter
 - (c) Laplace filter
 - (d) Mode filter

36. Statement 1: Entropy is a measure of average information.

Statement 2: Entropy is maximum if the symbols are equiprobable.

- (a) Statements 1 and 2 are wrong.
(b) Statement 1 is correct, Statement 2 is wrong.
(c) Statements 1 and 2 are true.
(d) Statement 2 is correct, Statement 1 is wrong.

37. In a black-and-white image, the probability of occurrence of black and white pixels is 0.5. Then the entropy of the image in bits per sample is

- (a) 0.25 (b) 0.5 (c) 0.75 (d) 1.0

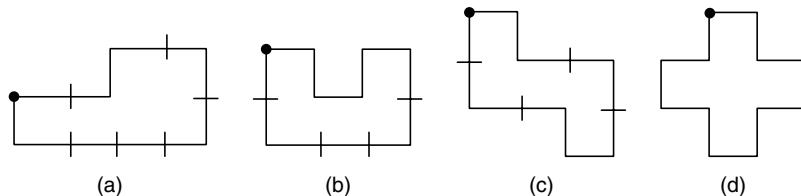
38. The conversion between RGB and CMY colour model is given by

$$\begin{array}{l} \text{(a)} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \\ \text{(b)} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \\ \text{(c)} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \\ \text{(d)} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \end{array}$$

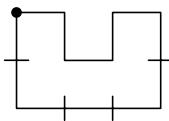
39. The operator which can be used to detect the edges in an image is

- (a) adder (b) integrator
(c) multiplier (d) differentiator

40. If the chain code is (0 3 0 0 3 3 2 1 2 2 1 1), the corresponding shape is



41. The shape is given below, and corresponding chain code is



- (a) 0 3 0 1 0 3 3 2 2 2 1 1
(b) 0 3 0 1 0 3 3 2 2 2 1 1
(c) 0 3 0 3 2 3 2 1 2 1 0 1
(d) 0 0 1 0 0 3 3 2 2 2 2 1

42. The Laplacian mask is given by

- (a) $\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$ (b) $\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$ (c) $\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

43. The colour model which is more relevant to a display system is the

- (a) RGB model (b) CMY model
(c) HSI model (d) YIQ model

44. The colour model which is more suitable for printing purposes is the

- (a) RGB model (b) CMY model
(c) HSI model (d) YIQ model

45. The image file format that can support simple animation is

- (a) GIF87a (b) GIF89a
(c) TIFF (d) JPEG

46. If the impulse response is given as

$h(n_1, n_2) = \delta(n_1 + 1, n_2) + \delta(n_1 - 1, n_2) + \delta(n_1, n_2 + 1) + \delta(n_1, n_2 - 1)$, the frequency response is given by

- (a) $(\cos \omega_1 + \cos \omega_2)$ (b) $\frac{1}{2}(\cos \omega_1 + \cos \omega_2)$
(c) $2(\cos \omega_1 + \cos \omega_2)$ (d) $\frac{1}{4}(\cos \omega_1 + \cos \omega_2)$

47. The Fourier transform of a spatial filter (sampling distance = 1 pixel) is given by $H(k, l) = 8 - 2 \cos(2\pi k) - 2\cos(2\pi l) - 2\cos(2\pi(k+l)) - 2\cos(2\pi(k-l))$. The filter will act like a

 - (a) low-pass filter
 - (b) high-pass filter
 - (c) band-pass filter
 - (d) band-reject filter

48. An image is given by the array
$$\begin{bmatrix} 3 & 1 & 0 & 2 \\ 5 & 4 & 3 & 2 \\ 7 & 5 & 6 & 2 \\ 4 & 6 & 3 & 2 \end{bmatrix}$$
. The minimum number of bits required to store each picture element is

49. You have an image $f(m, n)$. On taking two times the Fourier transform, the result will be
(a) $f(2m, 2n)$ (b) $f(m/2, n/2)$ (c) $1/2f(m, n)$ (d) $f(-m, -n)$

50. The minimum D4 and D8 distances between the marked pixels 1 and 5 are

$$\begin{bmatrix} 1 & 2 & 4 & 8 \\ 2 & 6 & 4 & 2 \\ 1 & 3 & 4 & 5 \\ 2 & 2 & 1 & 5 \end{bmatrix}$$

51. The parameter that may change if all the pixels in an image are shuffled is
(a) mean (b) entropy (c) histogram (d) covariance

52. The impulse response of a filter is given by

$$h(m, n) = 0.25[\delta(m+1, n) + \delta(m-1, n) + \delta(m, n+1) + \delta(m, n-1)]$$

The filter will act like a

53. Two images $x[m,n] = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $h[m,n] = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ are convolved to get the resultant image as

- $$(a) \begin{bmatrix} 5 & 16 & 12 \\ 22 & 60 & 40 \\ 21 & 52 & 32 \end{bmatrix}$$

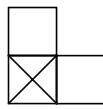
- (b)
$$\begin{bmatrix} 5 & 16 & 12 \\ 22 & 60 & 40 \\ 21 & 52 & 32 \end{bmatrix}$$

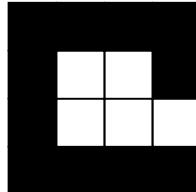
- $$(c) \begin{bmatrix} 5 & 16 & 12 \\ 22 & 60 & 40 \\ 21 & 52 & 32 \end{bmatrix}$$

- (d)
$$\begin{bmatrix} 5 & 16 & 12 \\ 22 & 60 & 40 \\ 21 & 52 & 32 \end{bmatrix}$$

Note: The circle denotes the origin.

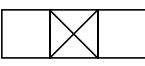
54. For the binary image shown below, calculate the number of white pixels left if it is eroded by the

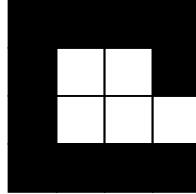
structuring element 



- (a) 1 (b) 2 (c) 4 (d) 8

55. For the binary image shown below, calculate the number of white pixels left if it is dilated by the

structuring element 



- (a) 1 (b) 2 (c) 4 (d) 8

56. The time-bandwidth product $\Delta t \times \Delta \omega = \frac{1}{2}$ for

- (a) rectangular window (b) triangular window
 (c) raised cosine window (d) Gaussian window

57. An example of a perfectly symmetric compactly supported orthonormal wavelet is the

- (a) Morlet wavelet (b) Mexican-hat wavelet
 (c) Lemarie and Battle wavelet (d) Haar wavelet

58. A 2×2 image $f(m, n) = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$ is passed through the linear filter $h(m, n) = \frac{1}{2} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$. Then the resultant image is (assume zero padding)

- (a) $\begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}$ (b) $\begin{bmatrix} 6 & 8 \\ 2 & 4 \end{bmatrix}$ (c) $\begin{bmatrix} 2 & 6 \\ 4 & 8 \end{bmatrix}$ (d) $\begin{bmatrix} 8 & 6 \\ 2 & 4 \end{bmatrix}$

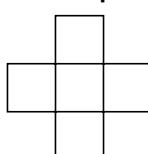
59. Which one of the following is a lossy coding?

- (a) Huffman coding (b) Run-length coding
 (c) Uniform quantiser (d) Predictive coding without quantiser

60. In a DPCM coder, which of the following needs to be quantised?

- (a) The prediction value (b) The transform coefficient
 (c) The reconstruction value (d) Difference between the prediction value and original value

61. The image compression scheme which exploits the self-similarity within an image is
 (a) wavelet-based compression (b) singular-value-decomposition-based compression
 (c) fractal-based compression (d) DPCM-based compression
62. The Fourier transform of a two-dimensional comb function is a two-dimensional
 (a) delta function (b) ramp function
 (c) step function (d) comb function
63. The application of a Laplacian operator to a 2D Gaussian function results in
 (a) Haar wavelet (b) Mexican-hat wavelet
 (c) Daubechies wavelet (d) Coiflet
64. The number of shades of gray in a six-bit image is
 (a) 256 (b) 128 (c) 64 (d) 32
65. The edge detector which uses non-maximum suppression to thin edges is the
 (a) Robert operator (b) Canny operator
 (c) Sobel operator (d) Prewitt operator
66. If one fixes the vector dimension to be 2 and the bit rate as 2 bits/dimension then the number of code vectors in the code book is
 (a) 8 (b) 16 (c) 32 (d) 64
67. An example of an additive colour space is
 (a) RGB (b) CMY (c) HSI (d) YUV
68. An example of a subtractive colour space is
 (a) RGB (b) CMY (c) HSI (d) YUV
69. The mask which can be used to perform smoothing operation of an input image is
- (a) $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & -1 & 1 \end{bmatrix}$ (d) $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
70. A 4×4 image is given by $\begin{bmatrix} 2 & 3 & 4 & 5 \\ 1 & 2 & 4 & 6 \\ 2 & 3 & 2 & 4 \\ 1 & 5 & 7 & 6 \end{bmatrix}$. If this image is filtered by a Min filter with a mask given below then the resultant image is (assume zero padding)



- (a) $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ (b) $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ (c) $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

- ## 71. The Fourier transform of a 2D Gaussian function results in a two-dimensional

72. What will be the value of the marked central pixel after applying a 5×5 mode filter?

0	1	2	1	4
4	1	4	5	6
3	3	3	0	4
6	4	1	3	5
4	2	2	1	2

73. The operator which can be used to detect edges in an image is

- (a) logarithm (b) exponential
(c) gradient (d) average

74. The transform whose kernel matrix contains elements which are either +1 or -1 is

- (a) discrete cosine transform (b) discrete sine transform
(c) hadamard transform (d) Haar transform

75. The transform which does not have a DC coefficient is

- The transform which does not have a DC coefficient is

 - (a) Discrete cosine transform
 - (b) Discrete sine transform
 - (c) KLT transform
 - (d) Haar transform

76. The Kronecker product between the two matrices $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ is

$$\begin{array}{l}
 \text{(a)} \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{bmatrix} \quad \text{(b)} \begin{bmatrix} 4 & 4 & 3 & 3 \\ 4 & 4 & 3 & 3 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \quad \text{(c)} \begin{bmatrix} 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \\ 1 & 2 & 1 & 2 \\ 3 & 3 & 3 & 4 \end{bmatrix} \quad \text{(d)} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}
 \end{array}$$

77. An example of a dictionary-based coding technique is

- All examples of a dictionary-based coding technique is

 - (a) Huffman coding
 - (b) run-length coding
 - (c) LZW coding
 - (d) predictive coding

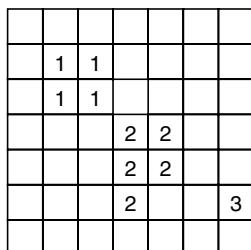
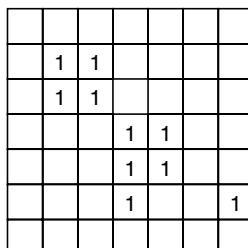
78. If the original image is $f(m, n) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 2 & 3 & 2 \\ 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 2 \end{bmatrix}$ and the reconstructed image is $R(m, n) = \begin{bmatrix} 2 & 2 & 4 & 4 \\ 6 & 2 & 4 & 2 \\ 2 & 4 & 4 & 6 \\ 2 & 2 & 6 & 2 \end{bmatrix}$

then the **Mean Absolute error (MAE)** between the original and the reconstructed image is

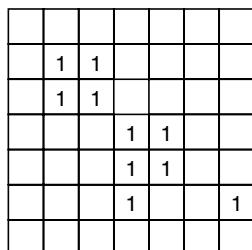
79. Let $f(m, n)$ and $h(m, n)$ represent 2D signals. If $h(m, n)$ is a two-dimensional impulse then the convolution of $f(m, n)$ with $h(m, n)$ will result in

- (a) $h(m, n)$ (b) $f(m, n)$ (c) $f(m, n) + h(m, n)$ (d) $f(m, n) = h(m, n)$

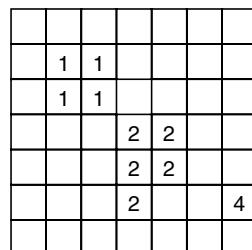
80. The term 'catchment basin' is associated with
- (a) LBG algorithm
 - (b) SPIHT algorithm
 - (c) Watershed algorithm
 - (d) SPECK algorithm
81. In multi-stage vector quantisation, except the first stage, the input to the successive stages are
- (a) input vectors
 - (b) error vectors from the previous stage
 - (c) quantised input vectors
 - (d) input vectors with some gain
82. If f represents the input image and B represents the structuring element then the morphological gradient is given by (Note: \oplus denotes dilation operation and \ominus denotes the erosion operation.)
- (a) $(f \oplus B) - (f \ominus B)$
 - (b) $(f \oplus B) + (f \ominus B)$
 - (c) $(f \oplus B) \times (f \ominus B)$
 - (d) $\frac{(f \oplus B)}{(f \ominus B)}$
83. If the Fourier transform of an image $f(m, n)$ is $F(k, l)$ and the Fourier transform of kernel $g(m, n)$ is $G(k, l)$ then the Fourier transform of $4f(m, n) + 5g(m, n)$ is
- (a) $F(4k, 4l) + G(5k, 5l)$
 - (b) $20F(k, l)G(k, l)$
 - (c) $4F(k, l) + 5G(k, l)$
 - (d) none of the above.
84. For the binary image shown below, the connected component labels assuming 4-neighbour connectivity is



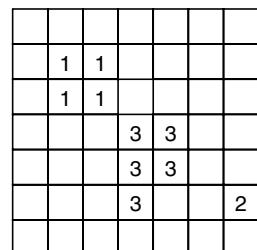
(a)



(b)

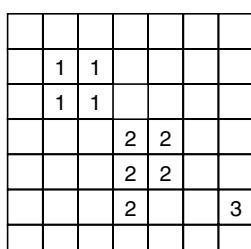


(c)

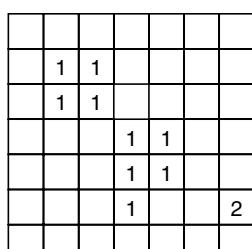


(d)

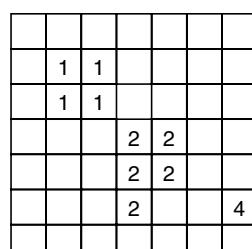
85. For the image given in Question 84, the connected component labels assuming 8-connectivity is



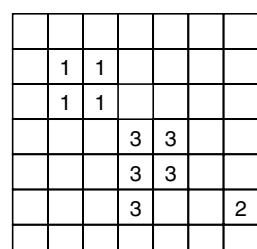
(a)



(b)



(c)



(d)

86. The singular values of the matrix $\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$ are
 (a) (1,0) (b) (2,0) (c) (3,0) (d) (4,0)
87. The watershed transform for image segmentation is executed on
 (a) the input image directly (b) gradient of the input image
 (c) input image after low pass filtering (d) input image after contrast stretching
88. The total number of bits required to represent a 256×256 image with 256 gray level is
 (a) 524288 (b) 65536 (c) 16777216 (d) 131072
89. The process of embedding one image in another image is termed
 (a) dithering (b) demosicing (c) watermarking (d) beamforming
90. The image-processing operation which is not commutative is
 (a) matrix addition (b) convolution (c) dilation (d) erosion
91. If $f(m, n)$ represents an 8-bit image then the transformation $g(m, n) = (m, n)$ will result in
 (a) low-pass filtered image (b) image negative
 (c) high-boost filtered image (d) gamma-corrected image
92. The transform which is recommended in JPEG2000 standard is
 (a) radon transform (b) discrete cosine transform
 (c) Walsh transform (d) wavelet transform
93. In image restoration, the blurred image is often modeled as
 (a) original image + PSF of blur (b) original image - PSF of blur
 (c) original image/ PSF of blur (d) convolution of original image with PSF of blur
94. The morphological operation which is commutative is
 (a) dilation (b) erosion (c) opening (d) closing
95. Which of the following statements is valid with respect to wavelets?
 (a) Wavelets are oscillatory functions of finite duration.
 (b) Wavelets are non-oscillatory functions of finite duration.
 (c) Wavelets are oscillatory functions of infinite duration.
 (d) Wavelets are non-oscillatory functions of infinite duration.

96. For the image patch $\begin{bmatrix} 2 & 3 & 4 & 5 & 6 \\ 3 & 1 & 2 & 3 & 8 \\ 5 & 3 & 2 & 1 & 3 \\ 4 & 1 & 2 & 3 & 2 \\ 3 & 2 & 1 & 4 & 2 \end{bmatrix}$, what will be the value of the marked central pixel after applying a 3×3 Gaussian filter given by $\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$?
 (a) 1 (b) 2 (c) 3 (d) 3.75
97. The quantiser in an image-compression system is a
 (a) lossless element which exploits the psychovisual redundancy
 (b) lossy element which exploits the psychovisual redundancy

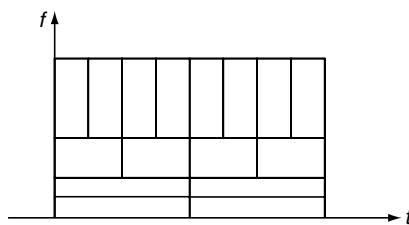


Fig. 1 Time-Frequency plane

- (c) Huffman code is a prefix code.
 (d) Huffman code is not uniquely decodable.
- 107. If the Fourier transform of the function $f(m, n)$ is $F(k, l)$, then the Fourier transform of the function $f(2m, 2n)$ is**
- (a) $\frac{1}{4}F\left(\frac{k}{2}, \frac{l}{2}\right)$ (b) $\frac{1}{4}F(2k, 2l)$ (c) $\frac{1}{4}F(k, l)$ (d) $\frac{1}{4}F\left(\frac{k}{4}, \frac{l}{4}\right)$
- 108. The singular value decomposition of the matrix A is given by $A = U\Sigma V^T$. Identify the false statement**
- (a) The number of singular values depends on the rank of the matrix A .
 (b) The singular values are the square roots of the eigen values of AA^T .
 (c) The singular values are the square roots of the eigen values of $A^T A$.
 (d) The singular values are the squares of the eigen values of $A^T A$.
- 109. The images at different levels of a Laplacian pyramid is similar to that of the output of a**
- (a) Low-pass filtered image (b) High-pass filtered image
 (c) Band-pass filtered image (d) Band-reject filtered image
- 110. Identify the statement that is wrong with respect to Laplacian pyramid.**
- (a) A Laplacian pyramid is a data structure composed of bandpass copies of an image.
 (b) Each level in a Laplacian pyramid represents the difference between successive levels of a Gaussian pyramid.
 (c) A Laplacian pyramid can be used as an image-transformation tool.
 (d) A Laplacian pyramid is non-redundant in nature.
- 111. The Agglomerative clustering technique comes under the category of**
- (a) one-to-one (b) one-to-many (c) many-to-one (d) many-to-many
- 112. The Divisive clustering technique can be considered in the category of**
- (a) one-to-one (b) one-to-many (c) many-to-one (d) many-to-many
- 113. The magnitude of the determinant of the unitary matrix is**
- (a) zero (b) unity (c) infinity (d) equal to the rank of the matrix
- 114. All the eigen values of the unitary matrix will have a magnitude of**
- (a) unity (b) zero (c) infinity (d) equal to the rank of the matrix
- 115. If the first row of a circulant matrix is given by [1 -2 4], then the complete matrix is given by**
- (a) $\begin{bmatrix} 1 & 4 & -2 \\ -2 & 1 & 4 \\ 4 & -2 & 1 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & -2 & -2 \\ -2 & 4 & 4 \\ 4 & 1 & 1 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & 4 & 1 \\ -2 & 1 & 4 \\ 4 & -2 & -2 \end{bmatrix}$ (d) $\begin{bmatrix} 1 & 1 & -2 \\ -2 & 4 & 4 \\ 4 & -2 & 1 \end{bmatrix}$
- 116. The matrix A can be decomposed into $A = U\Sigma V^T$ through singular value decomposition. If one attempts to plot the value of Σ , it will take the shape of a**
- (a) square (b) rectangle (c) straight line (d) parabola
- 117. The negative of an eight-bit image $f(m, n)$ is obtained using the relation $g(m, n) = 255 - f(m, n)$. This process is**
- (a) linear (b) shift-invariant
 (c) both linear and shift-invariant (d) neither linear nor shift invariant

118. The photosensitive 'detector' of the human eye is the
 (a) eyelens (b) iris (c) retina (d) cornea
119. A normal human eye has three types of cones that correspond to three ranges of the visible light which are
 (a) blue, yellow and red (b) blue, orange and green
 (c) red, violet and blue (d) blue, green and red
120. An example of an unsupervised neural network is
 (a) perceptron (b) backpropagation network
 (c) self-Organizing feature map (d) counter propagation network
121. An example of a multi-layer neural network is
 (a) McCulloch-Pitts (b) perceptron (c) ADALINE (d) MADALINE
122. Which of the following function is not possible to implement using a single-layer perceptron?
 (a) XOR (b) OR (c) NOT (d) AND
123. Bayes approach to pattern recognition fits into the category of
 (a) structural approach (b) statistical approach
 (c) template matching (d) neural-network approach
124. Which of the following statements is WRONG?
 (a) The covariance matrix is symmetric.
 (b) The eigen values of the covariance matrix are either positive or zero.
 (c) The eigen values of the covariance matrix are always negative.
 (d) The covariance matrix is a square matrix.
125. What should be the value of 'x' such that the mask acts as a high-pass filter?

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & x & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

- (a) 8 (b) 24 (c) 25 (d) 9

Answers

- | | | | |
|-------|-------|-------|-------|
| 1. a | 11. b | 21. a | 31. d |
| 2. a | 12. c | 22. b | 32. c |
| 3. d | 13. c | 23. c | 33. c |
| 4. b | 14. a | 24. a | 34. c |
| 5. d | 15. b | 25. a | 35. c |
| 6. b | 16. c | 26. b | 36. c |
| 7. b | 17. a | 27. d | 37. d |
| 8. b | 18. a | 28. a | 38. b |
| 9. b | 19. b | 29. b | 39. d |
| 10. a | 20. a | 30. b | 40. c |

41. a	63. b	85. b	107. a
42. d	64. c	86. b	108. d
43. a	65. b	87. b	109. c
44. b	66. b	88. a	110. d
45. b	67. a	89. c	111. c
46. c	68. b	90. d	112. b
47. b	69. a	91. b	113. b
48. c	70. b	92. d	114. a
49. d	71. c	93. d	115. a
50. b	72. d	94. a	116. c
51. d	73. c	95. a	117. b
52. a	74. c	96. b	118. c
53. b	75. b	97. b	119. d
54. b	76. a	98. c	120. c
55. d	77. c	99. a	121. d
56. d	78. b	100. a	122. a
57. d	79. b	101. a	123. b
58. a	80. c	102. c	124. c
59. c	81. b	103. b	125. b
60. d	82. a	104. c	
61. c	83. c	105. d	
62. d	84. a	106. d	

Glossary of Image Processing Terms

Archival image A digital image taken at the highest practicable resolution and stored securely

Access images A term used to denote low-resolution images (thumbnails, $\frac{1}{4}$ screen images) that are made available (usually at no cost) through the Internet

Adaptive filter A filter whose behaviour changes in response to variations in local image properties

Additive primary colour The colours red, green and blue which, when added in different combinations, can produce all the colours

Affine transformation A first-order geometric transformation that involves a combination of translation, scaling, rotation and skewing

Algorithm A computer program that will perform tasks, e.g., compression of file sizes

Aliasing A phenomenon which occurs when an image is undersampled; due to aliasing, the information with a high spatial frequency is incorrectly represented, manifesting itself as an artifact with a lower spatial frequency

Alpha-trimmed mean filter A filter that sorts pixel values from the neighbourhood into ascending order, discards a certain number of values at either end of the list and then outputs the mean of the remaining values

Amplitude spectrum A measure of how much of each frequency component is present in an image

Analog image An image characterised by a physical magnitude varying continuously in space

Analog-to-digital converter Used to convert an analog signal into digital form

Anti-aliasing The technique of minimising the distortion artifacts, known as aliasing, when representing a high-resolution signal at a lower resolution

Area array A common type of detector arrangement within a digital camera containing a fixed number of horizontal and vertical pixels

Artifacts Unwanted blemishes, which may have been introduced to an image by electrical noise during scanning or compression

Bandwidth A measure of data speed in bits per second in digital systems; a high bandwidth network is required for fast transfer of image files as they typically contain large amounts of data

Basis function Used to represent an image; the basis function should be linearly independent and it should span the space

Binary Computer data made up of a series of 0s or 1s; each individual character is referred as a bit

Binary image Binary image takes only two pixel values which are either '0' or '1'

Byte A collection of eight bits

Bit Depth Number of bits used to describe the colour of each pixel; greater bit depth allows more colours to be used in the colour palette for the image—8-bits per pixel will allow 256 colours; 8-bits per colour component in a RGB image will allow 16777216 colours ($256 \times 256 \times 256$)

Bitmapped image One of the many types of file formats for images stored in computerised form; the bitmap images are pixel-based which means that the location and colour information about the image is stored in individual pixels within a grid

Blocking artifact Common in block based coding and caused by the discontinuities that result from the rectangular windowing of the image data

Block Truncation Coding (BTC) A block-based image compression scheme developed by Delp and Mitchell which preserves the block mean and the block standard deviation

Brightness adaptation The human visual system has the ability to operate over a wide range of illumination levels. Dilatation and contraction of the iris of the eye can account for a change of only 16 times in the light intensity falling on the retina. The process which allows great extension of this range by changes in the sensitivity of the retina is called brightness adaptation

Camera A device for recording the reflectance characteristics of a scene on a sensitised medium

Capture Recording a subject in a digital form

Capture devices Include flatbed scanners, drum scanners, film scanners and digital cameras, and use electronic devices to capture images rather than photographic film

CCD Array (Charge Coupled Device) The two main types are linear array and area array, and are used in image scanners, digital cameras, video cameras

CGM (Computer Graphics Metafile) A standard vector graphics file format

Chain code A method of coding information about the position of the boundary of an object

Closing Dilatation operation followed by an erosion operation; closing operation fills the small holes and gaps in single pixel object and the operation is idempotent in nature

CMOS (Complementary Metal Oxide Semiconductor) A type of detector within a digital camera

CMY model (Cyan, Magenta and Yellow) They are complementary colours of RGB and can be used as subtractive primaries, CMY model is used mostly in printing devices

Coding redundancy In a digital image, the number of bites used for the representation of each pixel is constant for all the pixels, regardless of the value of the pixel and the frequency of occurrence of that value in the image. In general, coding redundancy is present in an image if the possible values are coded in such a way that one uses more code symbols than required. The most popular method for coding redundancy reduction is by employing variable length coding

Colour accuracy Fidelity of colour in the scanned image with that in the original

Colour look-Up table A table containing the RGB values for 256 colours. Storage space is saved as an 8-bit number which links each image pixel to a RGB value held in the table instead of each pixel holding a 24-bit description of its colour

Colour model Provides a standard way to specify a particular colour by defining a 3D coordinate system and a subspace that contains all constructible colours within a particular model

Compression An algorithm that is applied to a digital image to reduce its file size; compression can be either lossy or lossless

Compression ratio The ratio of the size of the input image to the size of the compressed bit stream; it is a measure of the performance of the compression algorithm

Cone A detector in the human eye that is sensitive to colour

Contrast adjustment Deals with adjusting the difference between the lightest and darkest areas of an image

Convolution Sums the products of pixel values from a neighbourhood and coefficients from the convolution kernel; can be used to filter an image

Correlation The relationship that exists between signals; the process of correlation measures the similarity between two signals; finding the similarity between two different signals is cross correlation

Covariance The joint variation of two variables about the common mean—an important statistic used to compute the basis of a KL transform

Cropping A method of discarding extraneous material from a digital image, e.g., borders created during the scanning process

Digital camera A camera that does not contain any film but records the image as a digital object. Which is then downloaded into a computer system

Digital image A matrix whose locations hold digital colour and/or brightness information which, when viewed at a suitable distance, form an image

Digital imaging A term used generally to describe the process of creating and manipulating digital images

Digital image processing The manipulation of digital data by computer programs in order to improve the appearance of an image

Digital watermark A visible or invisible watermark that is applied to a digital image so that ownership of the image is known

Dilation A morphological operation in which the binary image is expanded from its original shape

Discrete cosine transform The representation of the signal or image as a sum of sinusoids; in contrast to the Fourier transform, the DCT is real-valued and is a valuable tool in the field of image compression

Discrete Fourier transform The primary transform used for numerical computation in digital signal processing; widely used for spectrum analysis, fast convolution and many other applications

Dot gain An increase in the size of a printed dot, due to paper absorbency, ink type and temperature

DPI (Dots Per Inch) A measure of the spatial resolution of images

Drum scanner A ‘high-end’ device using photo-multiplier tube technology to capture images which are mounted on a cylinder

Dynamic range It is the difference between maximum and minimum value

Dynamic thresholding Automatic adjustment of brightness during a scan

Entropy A measure of the information content of a source

Erosion The counter process of dilation which shrinks an image

File format The order and conventions by which data is laid down on a storage medium

Film scanner Most commonly used for scanning 35-mm film by mounting film in a holder which may be automatically drawn through the device; may use a linear array CCD or an area array CCD

Filter A mathematical procedure that alters the digital numbers in an image

Flash media Generic term used to describe compact storage devices used in digital cameras

Flatbed scanner A digitisation device delivering scanned image data to a computer, the glass face on which the original is placed flat

Fourier transform The representation of a signal or image as a sum of complex-valued sinusoids that extend infinitely in time

Gamma correction Used to calibrate devices, smoothing out any irregularities between input and output signals

GIF (Graphic Interchange Format) A bitmap file format widely used on the Web; it has a limited colour palette (256 colours) which makes it more suited to graphics rather than photographs; can also be animated

Gray scale A number of grays ranging from black to white; an eight-bit grayscale image could have 254 grays between black and white

Hand Scanner A hand held, generally low-quality device for digitising images

Histogram Gives the frequency of occurrence of gray levels in an image

Histogram equalisation A contrast-enhancement technique with the objective to maintain a new enhanced image with a uniform histogram; it is achieved by using the normalised cumulative histogram as the gray-scale mapping function

Hit-or-Miss transform A transformation which is used for template matching; involves two template sets, B and $(W - B)$, which are disjoint

HSI model (Hue, Saturation and Intensity) Hue represents dominant colour as perceived by an observer, saturation refers to the relative purity or the amount of white light mixed with a hue, and intensity reflects the brightness

Huffman coding Huffman codes are variable codes that map one symbol to one code word. In Huffman coding, it is assumed that each pixel intensity has associated with it a certain probability of occurrence, and this probability is spatially invariant. Huffman coding assigns a binary code to each intensity value, with shorter codes going to intensities with higher probability

Idempotence A property in which the effect of multiple iterations of the operator is not different from the effect of a single iteration; opening and closing are examples of idempotent operators

Image archive Collection of images kept in secure storage

Image capture Process of obtaining a digital representation of an original through scanning, digital photography

Image database Place where image files are kept in an organised form. Software that facilitates organised storage and retrieval of digital images

Image longevity Length of time over which an image serves a purpose or length of time before degradation begins

Image processing The manipulation of images using image editing software

Image sharpening The operation that makes edges in the image more abrupt

Indexing The entry of data to assist in archiving and retrieval

Interpixel redundancy Related to interpixel correlation within an image

Jaggies Observed along a diagonal line of square pixels; especially apparent in low resolution graphical images

JFIF Also known as the JPEG File Interchange Format is the technical name for the file format more widely known as JPEG

JPEG (JPG) Named after the Joint Photographic Experts Group who devised it; the JPEG format compresses images but sacrifices image detail (lossy compression)

JPEG2000 A file format which uses wavelet-based image compression which offers high-compression ratios without the image degradation associated with standard JPEG compression

Laplacian A second-order derivative operator often used in edge detection

Logarithmic stretch A non-linear stretch that preferentially stretches the darker parts of a scene

Look-up table A method of mapping input pixel values to the output pixel values

Lossless compression Reduces the file size without any loss of information; the uncompressed image is identical to the original.

Lossy compression Reduces the file size by actually removing data from the image; the post-compressed image is different from the pre-compressed image, even though they may look identical (visually lossless)

LZW A form of lossless compression available in GIF and TIFF files; LZW compression is proprietary; the acronym LZW is derived from the names of its creators Lempel-Ziv and Welch

Mach band Describes an effect where the human mind subconsciously increases the contrast between two surfaces with different luminance

Mean or average filter Computes the sum of all pixels in the filter window and then divides the sum by the number of pixels in the filter window; can be used to minimise the noise at the expense of blurring of the image

Median filter Replaces a pixel by the median of all pixels in the neighbourhood; an effective method that can suppress isolated noise without blurring sharp edges

Mexican hat The Mexican Hat Wavelet is obtained by applying Laplacian operator to the 2D Gaussian function

Moire An interference pattern which may occur when scanning images with a halftone screen

Morphological operations Refers to simple non-linear operations performed on than image; examples of basic morphological operations are dilation and erosion

Network scanners Scanners accessed and operated over a computer network, shared by a number of users

Noise May appear as bright specks in dark areas of a scan due to electrical interference in the CCD sensor and associated circuitry

OCR (Optical Character Recognition) An application which can recognise scanned printed type and convert it into editable text on a computer

Opening Opening operation is done by first applying the erosion and then dilation operation on an image; opening smoothes the inside of the object contour, breaks narrow strips and eliminates thin portions of the image

Open Format A file format which can be used royalty free. The TIFF file format in its uncompressed state can be used freely while the LZW compression function is used under licence

PhotoShop Image-editing computer-application program widely used in imaging; published by Adobe, it is generally regarded as the industry standard

Photo CD Compact Disk type storage technology developed by Kodak in the early 1990s

Pixel Picture element, smallest element of a digital image

Pixelation When an image is displayed at a normal viewing magnification and the pixels are apparent it is said to be pixelated

PNG (Portable Network Graphics) An open source image file format, which supports 24-bit colour, transparency and lossless compression

Point-Spread Function (PSF) An image may be degraded due to motion blur, atmospheric turbulence blur, out-of-focus blur and electronic noises. In an image-restoration process, point-spread function is used to model the mentioned degradations and restoration activity is carried out with the filters based on the point-spread function of the degradation

Power law stretch A non-linear stretch that preferentially stretches the brighter parts of a scene

Power spectrum The square of the amplitude spectrum of the signal

Power spectral density Fourier transform of the autocorrelation function gives the power spectral density

Prefix code Set of words (codes) such that no word of the set is a prefix of another word in the set; prefix codes are uniquely decodable and an example of a prefix code is the Huffman code

Proprietary format A file format, which is protected by a patent, use of which may require the payment of royalties; the native Adobe Photoshop (PSD) format is an example of a proprietary format

Psychovisual redundancy Psychovisual redundancy is due to the fact that the human eye does not respond with equal intensity to all visual information. Psychovisual redundancy can be minimised by the quantisation process

Pyramid algorithm A fast algorithm that is used to compute the wavelet transform and involves a series of linear filtering operations in combination with down-sampling by two of the output

Quantisation Basically an approximation; a non-linear process which is irreversible. Two common types are (i) scalar quantisation, and (ii) vector quantisation

Quarter screen image An image occupying one quarter of the area of a Web page

Radon transform A collection of 1D projections around the object at angle intervals of phi

Raster Image An image that is composed of pixels represented with a matrix of dots

RAW A native file format offered by some of the more advanced digital cameras. While this format is proprietary it also offers higher quality than the standard TIFF and JPEG formats. Adobe is developing an open source format (DNG) file which will retain all the data stored in the original proprietary RAW image while making the file more widely supported

Redundancy Redundant information is basically additional information. Even if the additional information is deleted, the remaining information will convey the correct meaning. Basically, compression is achieved by minimising the redundant information. Strictly speaking, redundancy refers to statistical correlation or dependence between the samples of the signal

Region-of-convergence The term commonly used in Z transforms; It is the region in the z-plane that has a defined value

Resampling Changing the resolution of an image by increasing or decreasing the number of pixels

Resolution Normally expressed as the number of pixels per linear unit, e.g., 300 ppi (pixels per inch), sometimes dpi (dots per inch) or spi (samples per inch); for colour resolution, see Bit depth

Retina A multi-layered sensory tissue that lines the back of the eye; contains millions of photoreceptors that capture light rays and convert them into electrical impulses

RGB model In the RGB model, red, green and blue are used as the three primary colours; an additive model which is widely used in computer monitors

Rod The photosensors in the eye which detect the variations in brightness; not sensitive to colour

Run-length coding Provides a data representation where sequences of symbol values of the same value are coded as 'run-lengths' which indicates the character and the length of the run

Sampling The process of converting continuous values to discrete values

Scalar quantisation Each pixel value is quantised separately to produce an output

Scan area The maximum dimensions of the area, in a flatbed scanner, in which an original can be placed and imaged

Scanner A device which delivers a digital image of that which is placed in it

Scanning area array A hybrid of scanning linear array and area array for the arrangement of the detector within a digital camera

Scanning linear array Another way in which the detector within a digital camera can be arranged; gives high-resolution images

Scotopic vision The scientific term for human vision in the dark

Segmentation Image segmentation refers to the process of partitioning an image into groups of pixels which are homogeneous with respect to some criterion; different groups must not intersect with each other and adjacent groups must be heterogeneous

Spatial domain A signal having space or distance as the independent variable; image is an example of a signal that can be processed in the spatial domain

Spatial redundancy The correlation between neighboring pixels in an image

Spatial resolution Describes the finest detail visible to the human eye

Spectrogram Allows time-frequency analysis of a signal; the horizontal dimension represents time and the vertical dimension represents frequency

SPI Samples per inch, a measure of the resolution of a capture device

Subtractive colour A colour system based on reflected light; colour (CMYK) printing is based on the subtractive colour system

Surrogate image A digital image that has been derived from the archival image; usually not as high resolution as the archival image, surrogate images are usually used as access images

System Any process that produces an output signal in response to an input signal

System palette A look-up table containing information on a limited number of colours, normally 256; computer manufacturers' system palettes may differ

Thumbnail image Small, low-resolution preview, often hyperlinked to a high-resolution version of the same image

TIFF (Tagged Image File Format) A widely used file format particularly suited to the storage of high-quality archive images

Transform A procedure or algorithm that changes one group of data into another group of data; image transform is basically used to represent the image

Transform coding Used to convert spatial image pixel values to transform coefficient values; the number of coefficients produced is equal to the number of pixels transformed

Unsharp masking A technique used to sharpen an image

Vector Image An image that is composed of individual elements, e.g., arc, line, polygon, that have their own attributes. These attributes can be individually edited. A drawing-type package is usually required to display such vector images. File sizes tend to be smaller than raster images

Visually Lossless A compression technique that reduces the file size by removing data such that the eye does not notice. A visual comparison between the original file and the compressed file do not show any marked differences. A comparison of the binary code will determine the differences

Voxel An acronym for volume pixels

Wavelet Oscillatory functions of finite duration; useful for representing signals and images with discontinuities

Wavelet packet A waveform whose oscillations persist for several or many cycles, but are still finite; a wavelet packet has a location, a scale and an oscillation

Wavelet transform Basically the representation of an image using wavelet functions at different locations and scales

Wiener filter An optimal stationary linear filter for images degraded by additive noise and blurring

Zonal coding In zonal coding, the transform of the input image is taken first. Then coefficients within a specified region are coded. The zone shapes used are consistent with the observation that most of the energy in typical images is concentrated in the low-frequency regions

Zoom A technique used to examine a portion of an image in greater detail

Z-transform A tool to analyse discrete systems

Reference

http://www.tasi.ac.uk/glossary/images_glossary.html#Bit

Index

A

Activation function, 422
 Identity function, 422
 Step function, 422
 Sigmoidal function, 422

Active contour, 393

Affine transformation, 508

Aliasing, 8

Analog Colour TV Camera, 34

Arithmetic coding, 457

2D Autocorrelation

 Z transform method, 136

Automated object recognition, 409

B

Back Propagation Network, 428

Bartlett filter, 266

Bayes classifier, 416

Baye's shrink, 659

Best basis selection, 645

Bias, 421

Binary Image, 20

Bit plane slicing, 275

Blind deconvolution, 344

 Conjugate gradient method, 346

 Iterative blind deconvolution method, 345

 Priori blur identification method, 344

 Simulated annealing method, 347

Blind restoration, 324

Blocking artifact, 490

Block truncation coding, 511

Blur, 325

 Atmospheric turbulence blur, 327

 Gauss blur, 325

 Motion blur, 325

 Out-of-focus blur, 325

Blur Signal to Noise Ratio, 357

Boundary detection, 567

Boundary representation

Box filter, 263

Brightness, 16

Brightness modification, 245

C

Camcorder, 36

Canny edge detector, 389

Catchment basin, 395

Chain code, 397

Chessboard distance, 578

Chromaticity diagram, 590

2D Circular Convolution

 Matrix method, 122

2D Circular Correlation

 Matrix method, 127

 Transform method, 133

City-block distance, 578

Closing, 555

Clustering, 376

 Fuzzy clustering, 377

 Hierarchical clustering, 376

 K-means clustering, 377

 Partitional clustering, 376

CMY colour model, 587

Code book, 498

Colour, 585

Colour image, 21

Colour image filtering, 597

Colour image histogram, 594

Colour image segmentation, 605

Colour Models, 586

 RGB colour model, 586

 CMY colour model, 587

 HIS colour model, 589

YIQ colour model, 590
YCbCr colour model, 590
Colour quantization, 593
Competitive network, 424
Compression ratio, 493
Constrained least square filter, 340
Continuous Wavelet Transform, 615
Contrast, 17
 Simultaneous contrast, 17
Contrast adjustment, 246
Contourlet transform, 648
Convex hull, 571
2D Convolution, 55
 Graphical method, 85
 Matrix method, 119
 Z transform method, 115
2D Correlation, 127
 Matrix method, 137
2D Cross correlation
 Z transform method, 140
Cryptography, 660

D

Dark current, 23
Daubechies wavelet, 618
Decision-theoretic approach, 415
Delta modulation, 469
Delta rule, 426
2D DFT, 169
Dictionary based compression, 469
Difference of Gaussian, 388
Differential Pulse Code Modulation, 471
2D Digital Filter, 71
Dilation, 549
Directional filter bank, 652
Discrete Cosine Transform, 194
Discrete Wavelet Transform, 615
Discriminant function, 418
Distance transform-, 577
Dominant pass, 632
Dynamic range, 498

E

EBCOT, 641
Edge detection, 381
Edge types, 380
 Step edge, 380
 Line edge, 380
 Ramp edge, 380
 Roof edge, 381

Embedded image coding, 626
Embedded Zero tree Wavelet, 626
Energy compaction, 494
Erosion, 552
Euclidean distance, 577

F

Face recognition, 435
Feature Selection, 410
Fill factor, 23
Finger print, 435
Fingerprint recognition, 435
Fourier Transform, 155
Fourier Descriptor, 399
Fractal image compression, 507
Frei-Chen edge detector, 385
Frequency domain filtering, 278
 Low pass filtering, 278
 High pass filtering, 282

G

Gain shape vector quantization, 507
Gamut, 592
Gaussian filter, 267
Gaussian pyramid, 650
Global thresholding, 379,544
Gradient descent rule, 426
Gradient operator, 381
Grammar, 432
Grayscale image, 21

H

Haar transform, 182
Haar wavelet, 417
Hadamard transform, 181
Hebb rule, 425
Heisenberg uncertainty principle, 612
High-boost filter, 274
Histogram, 248
Histogram equalisation, 248
Hit-or-Miss transform, 565
Homomorphic filter, 292
Hopfield rule, 425
Hotelling transform, 202
Hough transform, 392
Huffman coding, 452
 Binary Huffman code, 452
 Non-binary Huffman code, 454
 Adaptive Huffman code, 456
Human Visual System, 14

I

- Image, 2
 - Analog Image, 2
 - Digital Image, 2
- Image Arithmetic, 311
 - Alpha blending, 300
 - Image addition, 297
 - Image division, 299
 - Image multiplication, 299
 - Image subtraction, 298
- Image compression, 444
- Image Enhancement, 243
- Image File Format, 38
 - Bit Map File format, 41
 - Encapsulated Post Script, 41
 - Graphics Interchange Format, 39
 - Joint Photographic Expert Group, 39
 - Joint Photographic Expert Group 2000(39
 - Portable Network Graphics, 40
 - PSD, 41
 - Scalable Vector Graphics, 41
 - Tagged Image File Format, 40
- Image Negative, 252
- Image pyramid, 650
- Image Restoration, 340
- Image Segmentation, 368
- Image Sensor, 23
 - CCD sensor, 23
 - CMOS sensor, 28
- Image Sharpening, 273
- Image Transform, 153
- Image watermarking, 659
- Improvement in Signal to Noise Ratio, 358
- Inverse filtering, 329
- Inverse Radon Transform, 224
- Iterated function system, 508
- Iterative restoration, 340

J

- JPEG, 488
- JPEG 2000, 639

K

- Karhunen-Loeve transform, 202
- Kohonen rule, 445

L

- Laplacian operator, 386
- Laplacian of Gaussian, 387

L

- Laplacian pyramid, 648
- Level shrink, 656
- Lifting scheme, 625
- Lossless compression, 447
- Lossy compression, 447
- Low pass filter, 260
- LZ77, 469
- LZ78, 469

M

- Machband effect, 19
- Mahalanobis distance, 378
- Mask operation, 244
- Mathematical morphology, 544
- Max-Min filter, 355
- Mean removed vector quantization, 507
- Median filter, 349
- Mexican hat wavelet, 618
- Momentum factor, 430
- Morlet wavelet, 618
- Morphology, 543
- Morphological operation
- Multichannel blind deconvolution, 348
- Multi-layer neural network, 424
- Multi-layer perceptron, 427
- Multiresolution, 616
- Multispectral image, 22
- Multistage vector quantization, 506
- Multiwavelet, 646

N

- Neighbors of a pixel
- Neural network, 419
- Noise
 - Additive noise, 363
 - Gaussian noise, 373
 - Impulse noise, 364
 - Multiplicative noise, 363
 - Poisson noise, 373
- Non-linear gray level transformation, 254
 - Exponential transformation, 258
 - Gamma correction, 259
 - Gray-level slicing, 255
 - Hard thresholding, 254
 - Logarithmic transformation, 257

O

- Object recognition, 409
- Opening, 555

P

Pattern, 411
Pattern class, 411
Pattern class representation
 String representation, 431
 Tree representation, 431
 Vector representation, 430
Perceptron, 426
Perfect reconstruction, 625
Photoconductive camera, 32
Photopic vision, 16
Pixel, 2
Pixelisation, 294
Point operation, 244
Point spread function, 328
Polygonal approximation, 400
Predictive coding, 469
Prefix code, 452
Prewitt operator, 384
Principal Component Analysis, 436
Properties of Morphological operation, 556
 Chain rule, 561
 Duality, 559
 Expansivity, 558
 Idempotency, 561
 Increasing, 556
Properties of 2D DFT, 159
 Conjugate symmetry, 167
 Convolution property, 65
 Correlation property, 165
 Multiplication by exponential, 168
 Orthogonality property, 167
 Periodicity property, 162
 Rotation property, 168
 Scaling property, 166
 Separable property, 159
 Spatial shift property, 161
Properties of Z transform, 64
 Linearity property, 64
 Convolution property, 65
 Shifting property, 65
 Conjugation property, 66
 Reflection property, 67
 Differentiation property, 67
 Modulation property, 68
Pseudo colour, 604
Pseudo inverse filter, 333
PSNR, 494

Q

Quantisation, 12
Quantum efficiency, 23

R

Radial basis function network, 431
Radon transform, 222
Range image, 22
Raster image, 19
Rate, 493
Rate-Distortion theory, 448
Recurrent network, 424
Redundancy, 445
 Psychovisual redundancy, 446
 Spatial redundancy, 446
 Statistical redundancy, 446
 Temporal redundancy, 446
Refinement pass, 632
Region based coding
Region filling, 569
Region growing, 369
Region splitting, 370
Region splitting and merging, 370
Reinforced learning, 425
Resolution, 13
 Gray level Resolution, 13
 Spatial Resolution, 13
Robert operator, 400
Runlength coding, 470

S

Sampling, 4
 Sampling theorem, 4
Scanner, 29
 Colour Scanner, 32
 Drum Scanner, 30
 Flatbed Scanner, 31
 Flying-spot Scanner, 30
Scotopic vision, 16
Sequency, 177
Set partitioning in hierarchical trees, 634
Set operation
Shannon-Fano coding, 450
Shannon wavelet, 619
Shape representation, 497
Short Time Fourier Transform, 613
2D Signal, 47

Single layer neural network, 424
 Singular Value Decomposition, 209
 Slant transform, 193
 Snake algorithm, 393
 Sobel Operator, 384
 SPIHT, 634
 Structuring element, 544
 Structural pattern recognition, 432
 Subband coding, 619
 Successive approximation quantization, 627
 Sure shrink, 657
 Supervised learning, 425
 Syntactic pattern recognition, 432
 2D System, 51
 Linear system, 51
 Shift-invariant system, 53
 Linear Shift Invariant system, 56
 Static system, 54

T
 Template matching, 418
 Thickening, 575
 Thinning, 573
 Threshold coding, 491
 Transform coding, 487
 Tree search vector quantization, 505
 Trimmed average filter, 354
 Two channel filter bank, 619

U
 Ultrasound, 36
 Unconstrained restoration, 329

Uniform scalar quantization, 495
 Midtread quantiser, 495
 Midrise quantiser, 495
 Unsharp masking, 274
 Unsupervised learning, 425

V

Vanishing moment, 644
 Variable length coding
 Vector image, 20
 Vector quantization, 497
 Visu shrink, 656
 Volume image, 22

W

Walsh transform, 175
 Watershed transformation, 394
 Wavelet, 613
 Wavelet shrinkage, 655
 Wavelet transform, 612
 Wavelet Packet Transform, 645
 Weight, 421
 Weighted average filter, 265
 Wiener filter, 335

Z

2D Z transform, 55
 Region of Convergence, 55
 2D Inverse Z transform, 71
 Zigzag scan, 488
 Zonal coding, 491
 Zoom, 293

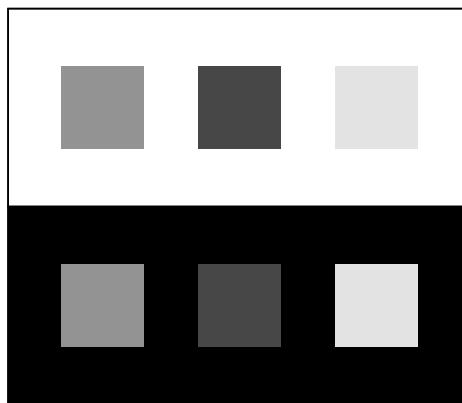


Fig. 1.19 Simultaneous contrast in a colour image

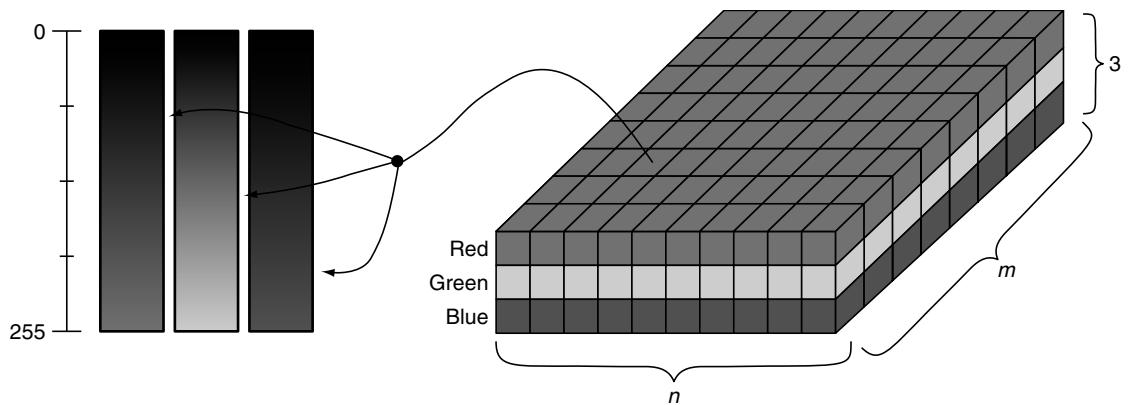


Fig. 1.27 RGB image representation



Fig. 1.28 Kalpana Chawla's colour image

Plate - 2

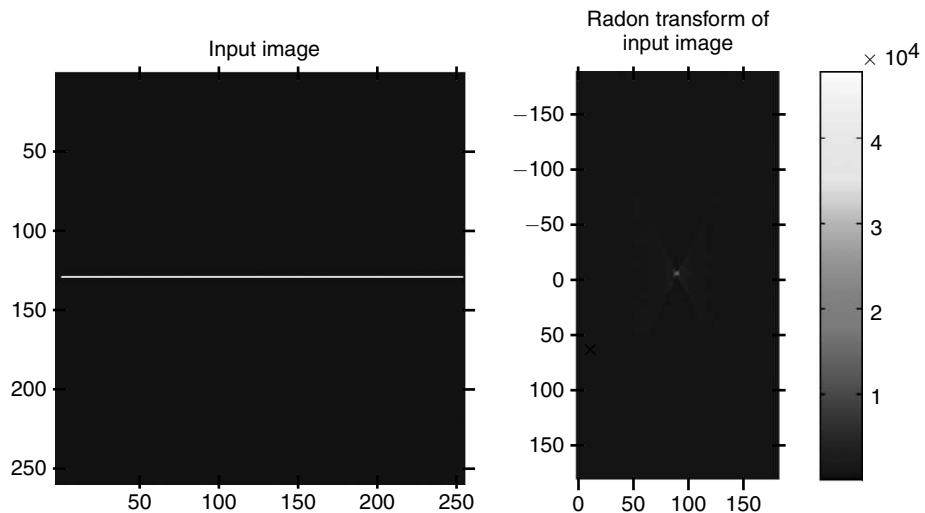


Fig. 4.40 Result of the MATLAB code given in Fig. 4.39

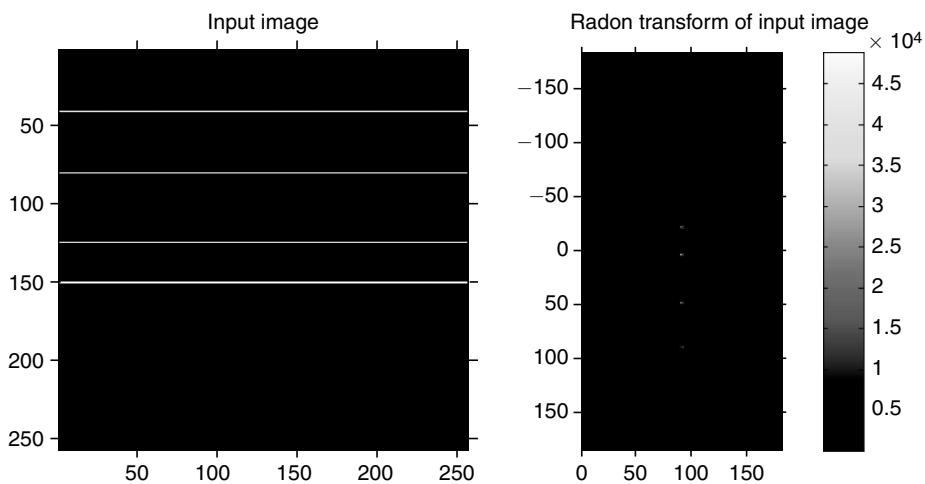


Fig. 4.42 Result of MATLAB code shown in Fig. 4.41

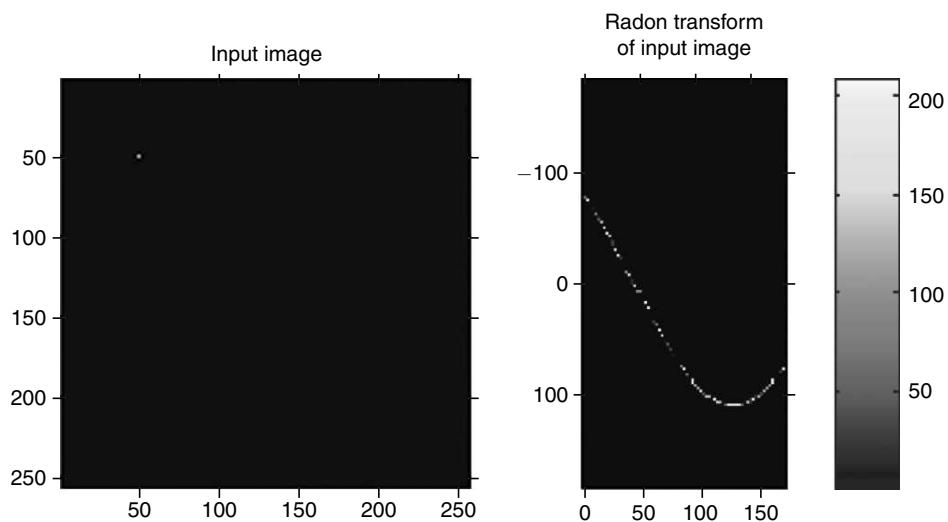


Fig. 4.44 Output of the MATLAB code given in Fig. 4.43

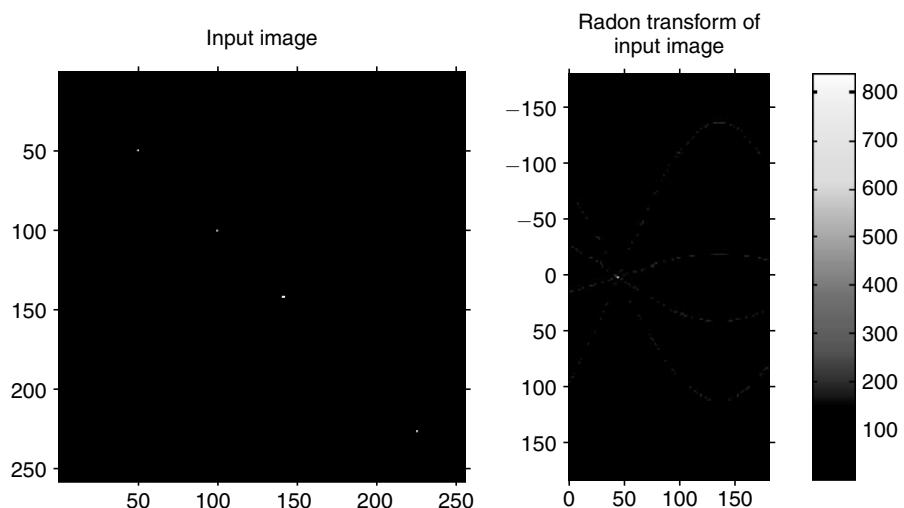
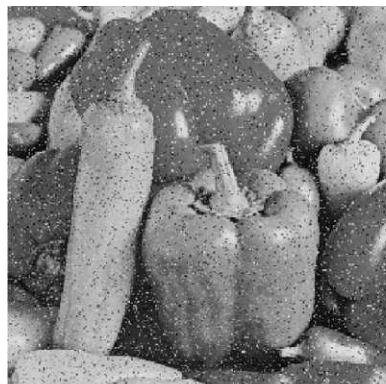


Fig. 4.46 Result of MATLAB code shown in Fig. 4.45

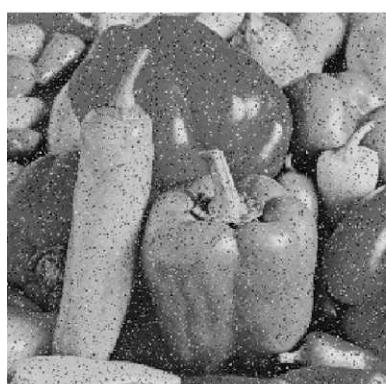
Plate - 4



(a) Colour image with salt-and-pepper noise



(b) 3×3 median-filtered image



(a) Corrupted image



(b) 5×5 median-filtered image

Fig. 6.22 Results of median filtering of colour image