



Digital Image Processing

FOURTH EDITION

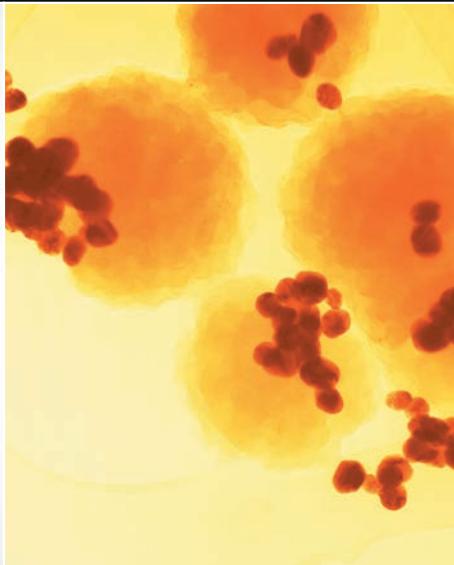
Rafael C. Gonzalez • Richard E. Woods



Pearson



www.BooksWorld.ir



Support Package for *Digital Image Processing*

Your new textbook provides access to support packages that may include reviews in areas like probability and vectors, tutorials on topics relevant to the material in the book, an image database, and more. Refer to the Preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for Rafael C. Gonzalez and Richard E. Woods' *Digital Image Processing*, Fourth Edition, Global Edition.

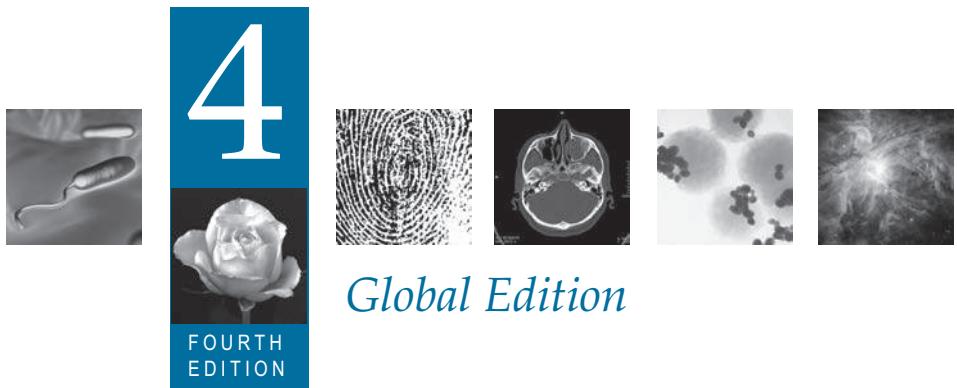
- 1. Go to www.ImageProcessingPlace.com**
- 2. Find the title of your textbook.**
- 3. Click Support Materials and follow the on-screen instructions to create a login name and password.**

Use the login name and password you created during registration to start using the digital resources that accompany your textbook.

IMPORTANT:

This serial code can only be used once. This subscription is not transferrable.

Digital Image Processing



Rafael C. Gonzalez

University of Tennessee

Richard E. Woods

Interaptics



Pearson

330 Hudson Street, New York, NY 10013

Senior Vice President Courseware Portfolio Management: Marcia J. Horton
Director, Portfolio Management: Engineering, Computer Science & Global Editions: Julian Partridge
Portfolio Manager: Julie Bai
Field Marketing Manager: Demetrius Hall
Product Marketing Manager: Yvonne Vannatta
Marketing Assistant: Jon Bryant
Content Managing Producer, ECS and Math: Scott Disanno
Content Producer: Michelle Bayman
Project Manager: Rose Kernan
Assistant Project Editor, Global Editions: Vikash Tiwari
Operations Specialist: Maura Zaldivar-Garcia
Manager, Rights and Permissions: Ben Ferrini
Senior Manufacturing Controller, Global Editions: Trudy Kimber
Media Production Manager, Global Editions: Vikram Kumar
Cover Designer: Lumina Datamatics
Cover Photo: CT image—©zhuravliki.123rf.com/Pearson Asset Library; Gram-negative bacteria—©royaltystockphoto.com/Shutterstock.com; Orion Nebula—©creativemarc/Shutterstock.com; Fingerprints—©Larysa Ray/Shutterstock.com; Cancer cells—©Greenshoots Communications/Alamy Stock Photo

MATLAB is a registered trademark of The MathWorks, Inc., 1 Apple Hill Drive, Natick, MA 01760-2098.

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

© Pearson Education Limited 2018

The rights of Rafael C. Gonzalez and Richard E. Woods to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Digital Image Processing, Fourth Edition, ISBN 978-0-13-335672-4, by Rafael C. Gonzalez and Richard E. Woods, published by Pearson Education © 2018.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-22304-9
ISBN 13: 978-1-292-22304-9

Typeset by Richard E. Woods

Printed and bound in Malaysia

*To Connie, Ralph, and Rob
and*

To Janice, David, and Jonathan

This page intentionally left blank

Contents

<i>Preface</i>	9
<i>Acknowledgments</i>	12
<i>The Book Website</i>	13
<i>The DIP4E Support Packages</i>	13
<i>About the Authors</i>	14

1 *Introduction* 17

What is Digital Image Processing?	18
The Origins of Digital Image Processing	19
Examples of Fields that Use Digital Image Processing	23
Fundamental Steps in Digital Image Processing	41
Components of an Image Processing System	44

2 *Digital Image Fundamentals* 47

Elements of Visual Perception	48
Light and the Electromagnetic Spectrum	54
Image Sensing and Acquisition	57
Image Sampling and Quantization	63
Some Basic Relationships Between Pixels	79
Introduction to the Basic Mathematical Tools Used in Digital Image Processing	83

3 *Intensity Transformations and Spatial Filtering* 119

Background	120
Some Basic Intensity Transformation Functions	122
Histogram Processing	133
Fundamentals of Spatial Filtering	153
Smoothing (Lowpass) Spatial Filters	164
Sharpening (Highpass) Spatial Filters	175
Highpass, Bandreject, and Bandpass Filters from Lowpass Filters	188
Combining Spatial Enhancement Methods	191

4 *Filtering in the Frequency Domain* 203

Background	204
Preliminary Concepts	207
Sampling and the Fourier Transform of Sampled Functions	215
The Discrete Fourier Transform of One Variable	225
Extensions to Functions of Two Variables	230
Some Properties of the 2-D DFT and IDFT	240
The Basics of Filtering in the Frequency Domain	260
Image Smoothing Using Lowpass Frequency Domain Filters	272
Image Sharpening Using Highpass Filters	284
Selective Filtering	296
The Fast Fourier Transform	303

5 *Image Restoration and Reconstruction* 317

A Model of the Image Degradation/Restoration process	318
Noise Models	318
Restoration in the Presence of Noise Only—Spatial Filtering	327
Periodic Noise Reduction Using Frequency Domain Filtering	340
Linear, Position-Invariant Degradations	348
Estimating the Degradation Function	352
Inverse Filtering	356
Minimum Mean Square Error (Wiener) Filtering	358
Constrained Least Squares Filtering	363
Geometric Mean Filter	367
Image Reconstruction from Projections	368

6 *Color Image Processing* 399

Color Fundamentals	400
Color Models	405
Pseudocolor Image Processing	420
Basics of Full-Color Image Processing	429
Color Transformations	430

Color Image Smoothing and Sharpening	442
Using Color in Image Segmentation	445
Noise in Color Images	452
Color Image Compression	455

7 *Wavelet and Other Image Transforms* 463

Preliminaries	464
Matrix-based Transforms	466
Correlation	478
Basis Functions in the Time-Frequency Plane	479
Basis Images	483
Fourier-Related Transforms	484
Walsh-Hadamard Transforms	496
Slant Transform	500
Haar Transform	502
Wavelet Transforms	504

8 *Image Compression and Watermarking* 539

Fundamentals	540
Huffman Coding	553
Golomb Coding	556
Arithmetic Coding	561
LZW Coding	564
Run-length Coding	566
Symbol-based Coding	572
Bit-plane Coding	575
Block Transform Coding	576
Predictive Coding	594
Wavelet Coding	614
Digital Image Watermarking	624

9 *Morphological Image Processing* 635

Preliminaries	636
Erosion and Dilation	638
Opening and Closing	644
The Hit-or-Miss Transform	648

Some Basic Morphological Algorithms	652
Morphological Reconstruction	667
Summary of Morphological Operations on Binary Images	673
Grayscale Morphology	674

10 *Image Segmentation* 699

Fundamentals	700
Point, Line, and Edge Detection	701
Thresholding	742
Segmentation by Region Growing and by Region Splitting and Merging	764
Region Segmentation Using Clustering and Superpixels	770
Region Segmentation Using Graph Cuts	777
Segmentation Using Morphological Watersheds	786
The Use of Motion in Segmentation	796

11 *Feature Extraction* 811

Background	812
Boundary Preprocessing	814
Boundary Feature Descriptors	831
Region Feature Descriptors	840
Principal Components as Feature Descriptors	859
Whole-Image Features	868
Scale-Invariant Feature Transform (SIFT)	881

12 *Image Pattern Classification* 903

Background	904
Patterns and Pattern Classes	906
Pattern Classification by Prototype Matching	910
Optimum (Bayes) Statistical Classifiers	923
Neural Networks and Deep Learning	931
Deep Convolutional Neural Networks	964
Some Additional Details of Implementation	987

Bibliography 995

Index 1009

Preface

When something can be read without effort, great effort has gone into its writing.

Enrique Jardiel Poncela

This edition of *Digital Image Processing* is a major revision of the book. As in the 1977 and 1987 editions by Gonzalez and Wintz, and the 1992, 2002, and 2008 editions by Gonzalez and Woods, this sixth-generation edition was prepared with students and instructors in mind. The principal objectives of the book continue to be to provide an introduction to basic concepts and methodologies applicable to digital image processing, and to develop a foundation that can be used as the basis for further study and research in this field. To achieve these objectives, we focused again on material that we believe is fundamental and whose scope of application is not limited to the solution of specialized problems. The mathematical complexity of the book remains at a level well within the grasp of college seniors and first-year graduate students who have introductory preparation in mathematical analysis, vectors, matrices, probability, statistics, linear systems, and computer programming. The book website provides tutorials to support readers needing a review of this background material.

One of the principal reasons this book has been the world leader in its field for 40 years is the level of attention we pay to the changing educational needs of our readers. The present edition is based on an extensive survey that involved faculty, students, and independent readers of the book in 150 institutions from 30 countries. The survey revealed a need for coverage of new material that has matured since the last edition of the book. The principal findings of the survey indicated a need for:

- Expanded coverage of the fundamentals of spatial filtering.
- A more comprehensive and cohesive coverage of image transforms.
- A more complete presentation of finite differences, with a focus on edge detection.
- A discussion of clustering, superpixels, and their use in region segmentation.
- Coverage of maximally stable extremal regions.
- Expanded coverage of feature extraction to include the Scale Invariant Feature Transform (SIFT).
- Expanded coverage of neural networks to include deep neural networks, back-propagation, deep learning, and, especially, deep convolutional neural networks.
- More homework exercises at the end of the chapters.

The new and reorganized material that resulted in the present edition is our attempt at providing a reasonable balance between rigor, clarity of presentation, and the findings of the survey. In addition to new material, earlier portions of the text were updated and clarified. This edition contains 241 new images, 72 new drawings, and 135 new exercises.

New to This Edition

The highlights of this edition are as follows.

Chapter 1: Some figures were updated, and parts of the text were rewritten to correspond to changes in later chapters.

Chapter 2: Many of the sections and examples were rewritten for clarity. We added 14 new exercises.

Chapter 3: Fundamental concepts of spatial filtering were rewritten to include a discussion on separable filter kernels, expanded coverage of the properties of low-pass Gaussian kernels, and expanded coverage of highpass, bandreject, and bandpass filters, including numerous new examples that illustrate their use. In addition to revisions in the text, including 6 new examples, the chapter has 59 new images, 2 new line drawings, and 15 new exercises.

Chapter 4: Several of the sections of this chapter were revised to improve the clarity of presentation. We replaced dated graphical material with 35 new images and 4 new line drawings. We added 21 new exercises.

Chapter 5: Revisions to this chapter were limited to clarifications and a few corrections in notation. We added 6 new images and 14 new exercises.

Chapter 6: Several sections were clarified, and the explanation of the CMY and CMYK color models was expanded, including 2 new images.

Chapter 7: This is a new chapter that brings together wavelets, several new transforms, and many of the image transforms that were scattered throughout the book. The emphasis of this new chapter is on the presentation of these transforms from a unified point of view. We added 24 new images, 20 new drawings, and 25 new exercises.

Chapter 8: The material was revised with numerous clarifications and several improvements to the presentation.

Chapter 9: Revisions of this chapter included a complete rewrite of several sections, including redrafting of several line drawings. We added 16 new exercises.

Chapter 10: Several of the sections were rewritten for clarity. We updated the chapter by adding coverage of finite differences, K -means clustering, superpixels, and graph cuts. The new topics are illustrated with 4 new examples. In total, we added 29 new images, 3 new drawings, and 6 new exercises.

Chapter 11: The chapter was updated with numerous topics, beginning with a more detailed classification of feature types and their uses. In addition to improvements in the clarity of presentation, we added coverage of slope change codes, expanded the explanation of skeletons, medial axes, and the distance transform, and added several new basic descriptors of compactness, circularity, and eccentricity. New material includes coverage of the Harris-Stephens corner detector, and a presentation of maximally stable extremal regions. A major addition to the chapter is a comprehensive discussion dealing with the Scale-Invariant Feature Transform (SIFT). The new material is complemented by 65 new images, 15 new drawings, and 12 new exercises.

Chapter 12: This chapter underwent a major revision to include an extensive rewrite of neural networks and deep learning, an area that has grown significantly since the last edition of the book. We added a comprehensive discussion on fully connected, deep neural networks that includes derivation of backpropagation starting from basic principles. The equations of backpropagation were expressed in “traditional” scalar terms, and then generalized into a compact set of matrix equations ideally suited for implementation of deep neural nets. The effectiveness of fully connected networks was demonstrated with several examples that included a comparison with the Bayes classifier. One of the most-requested topics in the survey was coverage of deep convolutional neural networks. We added an extensive section on this, following the same blueprint we used for deep, fully connected nets. That is, we derived the equations of backpropagation for convolutional nets, and showed how they are different from “traditional” backpropagation. We then illustrated the use of convolutional networks with simple images, and applied them to large image databases of numerals and natural scenes. The written material is complemented by 23 new images, 28 new drawings, and 12 new exercises.

Also for the first time, we have created student and faculty support packages that can be downloaded from the book website. The *Student Support Package* contains many of the original images in the book and answers to selected exercises. The *Faculty Support Package* contains solutions to all exercises, teaching suggestions, and all the art in the book in the form of modifiable PowerPoint slides. One support package is made available with every new book, free of charge.

The book website, established during the launch of the 2002 edition, continues to be a success, attracting more than 25,000 visitors each month. The site was upgraded for the launch of this edition. For more details on site features and content, see *The Book Website*, following the *Acknowledgments* section.

This edition of *Digital Image Processing* is a reflection of how the educational needs of our readers have changed since 2008. As is usual in an endeavor such as this, progress in the field continues after work on the manuscript stops. One of the reasons why this book has been so well accepted since it first appeared in 1977 is its continued emphasis on fundamental concepts that retain their relevance over time. This approach, among other things, attempts to provide a measure of stability in a rapidly evolving body of knowledge. We have tried to follow the same principle in preparing this edition of the book.

R.C.G.
R.E.W.

Acknowledgments

We are indebted to a number of individuals in academic circles, industry, and government who have contributed to this edition of the book. In particular, we wish to extend our appreciation to Hairong Qi and her students, Zhifei Zhang and Chengcheng Li, for their valuable review of the material on neural networks, and for their help in generating examples for that material. We also want to thank Ernesto Bribeasca Correa for providing and reviewing material on slope chain codes, and Dirk Padfield for his many suggestions and review of several chapters in the book. We appreciate Michel Kocher's many thoughtful comments and suggestions over the years on how to improve the book. Thanks also to Steve Eddins for his suggestions on MATLAB and related software issues.

Numerous individuals have contributed to material carried over from the previous to the current edition of the book. Their contributions have been important in so many different ways that we find it difficult to acknowledge them in any other way but alphabetically. We thank Mongi A. Abidi, Yongmin Kim, Bryan Morse, Andrew Oldroyd, Ali M. Reza, Edgardo Felipe Riveron, Jose Ruiz Shulcloper, and Cameron H.G. Wright for their many suggestions on how to improve the presentation and/or the scope of coverage in the book. We are also indebted to Naomi Fernandes at the MathWorks for providing us with MATLAB software and support that were important in our ability to create many of the examples and experimental results included in this edition of the book.

A significant percentage of the new images used in this edition (and in some cases their history and interpretation) were obtained through the efforts of individuals whose contributions are sincerely appreciated. In particular, we wish to acknowledge the efforts of Serge Beucher, Uwe Boos, Michael E. Casey, Michael W. Davidson, Susan L. Forsburg, Thomas R. Gest, Daniel A. Hammer, Zhong He, Roger Heady, Juan A. Herrera, John M. Hudak, Michael Hurwitz, Chris J. Johannsen, Rhonda Knighton, Don P. Mitchell, A. Morris, Curtis C. Ober, David. R. Pickens, Michael Robinson, Michael Shaffer, Pete Sites, Sally Stowe, Craig Watson, David K. Wehe, and Robert A. West. We also wish to acknowledge other individuals and organizations cited in the captions of numerous figures throughout the book for their permission to use that material.

We also thank Scott Disanno, Michelle Bayman, Rose Kernan, and Julie Bai for their support and significant patience during the production of the book.

R.C.G.

R.E.W.

The Book Website

www.ImageProcessingPlace.com

Digital Image Processing is a completely self-contained book. However, the companion website offers additional support in a number of important areas.

For the Student or Independent Reader the site contains

- Reviews in areas such as probability, statistics, vectors, and matrices.
- A Tutorials section containing dozens of tutorials on topics relevant to the material in the book.
- An image database containing all the images in the book, as well as many other image databases.

For the Instructor the site contains

- An Instructor's Manual with complete solutions to all the problems.
- Classroom presentation materials in modifiable PowerPoint format.
- Material removed from previous editions, downloadable in convenient PDF format.
- Numerous links to other educational resources.

For the Practitioner the site contains additional specialized topics such as

- Links to commercial sites.
- Selected new references.
- Links to commercial image databases.

The website is an ideal tool for keeping the book current between editions by including new topics, digital images, and other relevant material that has appeared after the book was published. Although considerable care was taken in the production of the book, the website is also a convenient repository for any errors discovered between printings.

The DIP4E Support Packages

In this edition, we created support packages for students and faculty to organize all the classroom support materials available for the new edition of the book into one easy download. The Student Support Package contains many of the original images in the book, and answers to selected exercises, The Faculty Support Package contains solutions to all exercises, teaching suggestions, and all the art in the book in modifiable PowerPoint slides. One support package is made available with every new book, free of charge. Applications for the support packages are submitted at the book website.

About the Authors

RAFAEL C. GONZALEZ

R. C. Gonzalez received the B.S.E.E. degree from the University of Miami in 1965 and the M.E. and Ph.D. degrees in electrical engineering from the University of Florida, Gainesville, in 1967 and 1970, respectively. He joined the Electrical and Computer Science Department at the University of Tennessee, Knoxville (UTK) in 1970, where he became Associate Professor in 1973, Professor in 1978, and Distinguished Service Professor in 1984. He served as Chairman of the department from 1994 through 1997. He is currently a Professor Emeritus at UTK.

Gonzalez is the founder of the Image & Pattern Analysis Laboratory and the Robotics & Computer Vision Laboratory at the University of Tennessee. He also founded Perceptics Corporation in 1982 and was its president until 1992. The last three years of this period were spent under a full-time employment contract with Westinghouse Corporation, who acquired the company in 1989.

Under his direction, Perceptics became highly successful in image processing, computer vision, and laser disk storage technology. In its initial ten years, Perceptics introduced a series of innovative products, including: The world's first commercially available computer vision system for automatically reading license plates on moving vehicles; a series of large-scale image processing and archiving systems used by the U.S. Navy at six different manufacturing sites throughout the country to inspect the rocket motors of missiles in the Trident II Submarine Program; the market-leading family of imaging boards for advanced Macintosh computers; and a line of trillion-byte laser disk products.

He is a frequent consultant to industry and government in the areas of pattern recognition, image processing, and machine learning. His academic honors for work in these fields include the 1977 UTK College of Engineering Faculty Achievement Award; the 1978 UTK Chancellor's Research Scholar Award; the 1980 Magnavox Engineering Professor Award; and the 1980 M.E. Brooks Distinguished Professor Award. In 1981 he became an IBM Professor at the University of Tennessee and in 1984 he was named a Distinguished Service Professor there. He was awarded a Distinguished Alumnus Award by the University of Miami in 1985, the Phi Kappa Phi Scholar Award in 1986, and the University of Tennessee's Nathan W. Dougherty Award for Excellence in Engineering in 1992.

Honors for industrial accomplishment include the 1987 IEEE Outstanding Engineer Award for Commercial Development in Tennessee; the 1988 Albert Rose National Award for Excellence in Commercial Image Processing; the 1989 B. Otto Wheeley Award for Excellence in Technology Transfer; the 1989 Coopers and Lybrand Entrepreneur of the Year Award; the 1992 IEEE Region 3 Outstanding Engineer Award; and the 1993 Automated Imaging Association National Award for Technology Development.

Gonzalez is author or co-author of over 100 technical articles, two edited books, and four textbooks in the fields of pattern recognition, image processing, and robotics. His books are used in over 1000 universities and research institutions throughout

the world. He is listed in the prestigious *Marquis Who's Who in America*, *Marquis Who's Who in Engineering*, *Marquis Who's Who in the World*, and in 10 other national and international biographical citations. He is the co-holder of two U.S. Patents, and has been an associate editor of the *IEEE Transactions on Systems, Man and Cybernetics*, and the *International Journal of Computer and Information Sciences*. He is a member of numerous professional and honorary societies, including Tau Beta Pi, Phi Kappa Phi, Eta Kappa Nu, and Sigma Xi. He is a Fellow of the IEEE.

RICHARD E. WOODS

R. E. Woods earned his B.S., M.S., and Ph.D. degrees in Electrical Engineering from the University of Tennessee, Knoxville in 1975, 1977, and 1980, respectively. He became an Assistant Professor of Electrical Engineering and Computer Science in 1981 and was recognized as a Distinguished Engineering Alumnus in 1986.

A veteran hardware and software developer, Dr. Woods has been involved in the founding of several high-technology startups, including Perceptics Corporation, where he was responsible for the development of the company's quantitative image analysis and autonomous decision-making products; MedData Interactive, a high-technology company specializing in the development of handheld computer systems for medical applications; and Interapptics, an internet-based company that designs desktop and handheld computer applications.

Dr. Woods currently serves on several nonprofit educational and media-related boards, including Johnson University, and was recently a summer English instructor at the Beijing Institute of Technology. He is the holder of a U.S. Patent in the area of digital image processing and has published two textbooks, as well as numerous articles related to digital signal processing. Dr. Woods is a member of several professional societies, including Tau Beta Pi, Phi Kappa Phi, and the IEEE.

This page intentionally left blank

1



Introduction

One picture is worth more than ten thousand words.

Anonymous

Preview

Interest in digital image processing methods stems from two principal application areas: improvement of pictorial information for human interpretation, and processing of image data for tasks such as storage, transmission, and extraction of pictorial information. This chapter has several objectives: (1) to define the scope of the field that we call image processing; (2) to give a historical perspective of the origins of this field; (3) to present an overview of the state of the art in image processing by examining some of the principal areas in which it is applied; (4) to discuss briefly the principal approaches used in digital image processing; (5) to give an overview of the components contained in a typical, general-purpose image processing system; and (6) to provide direction to the literature where image processing work is reported. The material in this chapter is extensively illustrated with a range of images that are representative of the images we will be using throughout the book.

Upon completion of this chapter, readers should:

- Understand the concept of a digital image.
- Have a broad overview of the historical underpinnings of the field of digital image processing.
- Understand the definition and scope of digital image processing.
- Know the fundamentals of the electromagnetic spectrum and its relationship to image generation.
- Be aware of the different fields in which digital image processing methods are applied.
- Be familiar with the basic processes involved in image processing.
- Be familiar with the components that make up a general-purpose digital image processing system.
- Be familiar with the scope of the literature where image processing work is reported.

1.1 WHAT IS DIGITAL IMAGE PROCESSING?

An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are *spatial* (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the *intensity* or *gray level* of the image at that point. When x , y , and the intensity values of f are all finite, discrete quantities, we call the image a *digital image*. The field of *digital image processing* refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are called *picture elements*, *image elements*, *pels*, and *pixels*. *Pixel* is the term used most widely to denote the elements of a digital image. We will consider these definitions in more formal terms in Chapter 2.

Vision is the most advanced of our senses, so it is not surprising that images play the single most important role in human perception. However, unlike humans, who are limited to the visual band of the electromagnetic (EM) spectrum, imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves. They can operate on images generated by sources that humans are not accustomed to associating with images. These include ultrasound, electron microscopy, and computer-generated images. Thus, digital image processing encompasses a wide and varied field of applications.

There is no general agreement among authors regarding where image processing stops and other related areas, such as *image analysis* and *computer vision*, start. Sometimes, a distinction is made by defining image processing as a discipline in which both the input and output of a process are images. We believe this to be a limiting and somewhat artificial boundary. For example, under this definition, even the trivial task of computing the average intensity of an image (which yields a single number) would not be considered an image processing operation. On the other hand, there are fields such as computer vision whose ultimate goal is to use computers to emulate human vision, including learning and being able to make inferences and take actions based on visual inputs. This area itself is a branch of *artificial intelligence* (AI) whose objective is to emulate human intelligence. The field of AI is in its earliest stages of infancy in terms of development, with progress having been much slower than originally anticipated. The area of image analysis (also called *image understanding*) is in between image processing and computer vision.

There are no clear-cut boundaries in the continuum from image processing at one end to computer vision at the other. However, one useful paradigm is to consider three types of computerized processes in this continuum: low-, mid-, and high-level processes. Low-level processes involve primitive operations such as image preprocessing to reduce noise, contrast enhancement, and image sharpening. A low-level process is characterized by the fact that both its inputs and outputs are images. Mid-level processing of images involves tasks such as segmentation (partitioning an image into regions or objects), description of those objects to reduce them to a form suitable for computer processing, and classification (recognition) of individual objects. A mid-level process is characterized by the fact that its inputs generally are images, but its outputs are attributes extracted from those images (e.g., edges, contours, and the identity of individual objects). Finally, higher-level processing

involves “making sense” of an ensemble of recognized objects, as in image analysis, and, at the far end of the continuum, performing the cognitive functions normally associated with human vision.

Based on the preceding comments, we see that a logical place of overlap between image processing and image analysis is the area of recognition of individual regions or objects in an image. Thus, what we call in this book *digital image processing* encompasses processes whose inputs and outputs are images and, in addition, includes processes that extract attributes from images up to, and including, the recognition of individual objects. As an illustration to clarify these concepts, consider the area of automated analysis of text. The processes of acquiring an image of the area containing the text, preprocessing that image, extracting (segmenting) the individual characters, describing the characters in a form suitable for computer processing, and recognizing those individual characters are in the scope of what we call digital image processing in this book. Making sense of the content of the page may be viewed as being in the domain of image analysis and even computer vision, depending on the level of complexity implied by the statement “making sense of.” As will become evident shortly, digital image processing, as we have defined it, is used routinely in a broad range of areas of exceptional social and economic value. The concepts developed in the following chapters are the foundation for the methods used in those application areas.

1.2 THE ORIGINS OF DIGITAL IMAGE PROCESSING

One of the earliest applications of digital images was in the newspaper industry, when pictures were first sent by submarine cable between London and New York. Introduction of the Bartlane cable picture transmission system in the early 1920s reduced the time required to transport a picture across the Atlantic from more than a week to less than three hours. Specialized printing equipment coded pictures for cable transmission, then reconstructed them at the receiving end. Figure 1.1 was transmitted in this way and reproduced on a telegraph printer fitted with typefaces simulating a halftone pattern.

Some of the initial problems in improving the visual quality of these early digital pictures were related to the selection of printing procedures and the distribution of



FIGURE 1.1 A digital picture produced in 1921 from a coded tape by a telegraph printer with special typefaces. (McFarlane.) [References in the bibliography at the end of the book are listed in alphabetical order by authors' last names.]

FIGURE 1.2

A digital picture made in 1922 from a tape punched after the signals had crossed the Atlantic twice. (McFarlane.)



intensity levels. The printing method used to obtain Fig. 1.1 was abandoned toward the end of 1921 in favor of a technique based on photographic reproduction made from tapes perforated at the telegraph receiving terminal. Figure 1.2 shows an image obtained using this method. The improvements over Fig. 1.1 are evident, both in tonal quality and in resolution.

The early Bartlane systems were capable of coding images in five distinct levels of gray. This capability was increased to 15 levels in 1929. Figure 1.3 is typical of the type of images that could be obtained using the 15-tone equipment. During this period, introduction of a system for developing a film plate via light beams that were modulated by the coded picture tape improved the reproduction process considerably.

Although the examples just cited involve digital images, they are not considered digital image processing results in the context of our definition, because digital computers were not used in their creation. Thus, the history of digital image processing is intimately tied to the development of the digital computer. In fact, digital images require so much storage and computational power that progress in the field of digital image processing has been dependent on the development of digital computers and of supporting technologies that include data storage, display, and transmission.

FIGURE 1.3

Unretouched cable picture of Generals Pershing (right) and Foch, transmitted in 1929 from London to New York by 15-tone equipment. (McFarlane.)



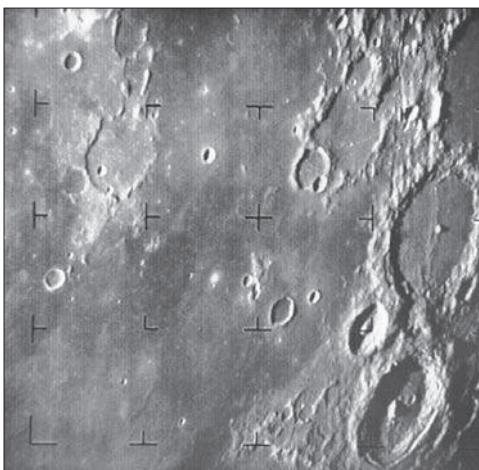
The concept of a computer dates back to the invention of the abacus in Asia Minor, more than 5000 years ago. More recently, there have been developments in the past two centuries that are the foundation of what we call a computer today. However, the basis for what we call a *modern* digital computer dates back to only the 1940s, with the introduction by John von Neumann of two key concepts: (1) a memory to hold a stored program and data, and (2) conditional branching. These two ideas are the foundation of a central processing unit (CPU), which is at the heart of computers today. Starting with von Neumann, there were a series of key advances that led to computers powerful enough to be used for digital image processing. Briefly, these advances may be summarized as follows: (1) the invention of the transistor at Bell Laboratories in 1948; (2) the development in the 1950s and 1960s of the high-level programming languages COBOL (Common Business-Oriented Language) and FORTRAN (Formula Translator); (3) the invention of the integrated circuit (IC) at Texas Instruments in 1958; (4) the development of operating systems in the early 1960s; (5) the development of the microprocessor (a single chip consisting of a CPU, memory, and input and output controls) by Intel in the early 1970s; (6) the introduction by IBM of the personal computer in 1981; and (7) progressive miniaturization of components, starting with large-scale integration (LI) in the late 1970s, then very-large-scale integration (VLSI) in the 1980s, to the present use of ultra-large-scale integration (ULSI) and experimental nonotechnologies. Concurrent with these advances were developments in the areas of mass storage and display systems, both of which are fundamental requirements for digital image processing.

The first computers powerful enough to carry out meaningful image processing tasks appeared in the early 1960s. The birth of what we call digital image processing today can be traced to the availability of those machines, and to the onset of the space program during that period. It took the combination of those two developments to bring into focus the potential of digital image processing for solving problems of practical significance. Work on using computer techniques for improving images from a space probe began at the Jet Propulsion Laboratory (Pasadena, California) in 1964, when pictures of the moon transmitted by *Ranger 7* were processed by a computer to correct various types of image distortion inherent in the on-board television camera. Figure 1.4 shows the first image of the moon taken by *Ranger 7* on July 31, 1964 at 9:09 A.M. Eastern Daylight Time (EDT), about 17 minutes before impacting the lunar surface (the markers, called *reseau marks*, are used for geometric corrections, as discussed in Chapter 2). This also is the first image of the moon taken by a U.S. spacecraft. The imaging lessons learned with *Ranger 7* served as the basis for improved methods used to enhance and restore images from the Surveyor missions to the moon, the *Mariner* series of flyby missions to Mars, the *Apollo* manned flights to the moon, and others.

In parallel with space applications, digital image processing techniques began in the late 1960s and early 1970s to be used in medical imaging, remote Earth resources observations, and astronomy. The invention in the early 1970s of *computerized axial tomography* (CAT), also called *computerized tomography* (CT) for short, is one of the most important events in the application of image processing in medical diagnosis. Computerized axial tomography is a process in which a ring of detectors

FIGURE 1.4

The first picture of the moon by a U.S. spacecraft. *Ranger 7* took this image on July 31, 1964 at 9:09 A.M. EDT, about 17 minutes before impacting the lunar surface. (Courtesy of NASA.)



encircles an object (or patient) and an X-ray source, concentric with the detector ring, rotates about the object. The X-rays pass through the object and are collected at the opposite end by the corresponding detectors in the ring. This procedure is repeated as the source rotates. Tomography consists of algorithms that use the sensed data to construct an image that represents a “slice” through the object. Motion of the object in a direction perpendicular to the ring of detectors produces a set of such slices, which constitute a three-dimensional (3-D) rendition of the inside of the object. Tomography was invented independently by Sir Godfrey N. Hounsfield and Professor Allan M. Cormack, who shared the 1979 Nobel Prize in Medicine for their invention. It is interesting to note that X-rays were discovered in 1895 by Wilhelm Conrad Roentgen, for which he received the 1901 Nobel Prize for Physics. These two inventions, nearly 100 years apart, led to some of the most important applications of image processing today.

From the 1960s until the present, the field of image processing has grown vigorously. In addition to applications in medicine and the space program, digital image processing techniques are now used in a broad range of applications. Computer procedures are used to enhance the contrast or code the intensity levels into color for easier interpretation of X-rays and other images used in industry, medicine, and the biological sciences. Geographers use the same or similar techniques to study pollution patterns from aerial and satellite imagery. Image enhancement and restoration procedures are used to process degraded images of unrecoverable objects, or experimental results too expensive to duplicate. In archeology, image processing methods have successfully restored blurred pictures that were the only available records of rare artifacts lost or damaged after being photographed. In physics and related fields, computer techniques routinely enhance images of experiments in areas such as high-energy plasmas and electron microscopy. Similarly successful applications of image processing concepts can be found in astronomy, biology, nuclear medicine, law enforcement, defense, and industry.

These examples illustrate processing results intended for human interpretation. The second major area of application of digital image processing techniques mentioned at the beginning of this chapter is in solving problems dealing with machine perception. In this case, interest is on procedures for extracting information from an image, in a form suitable for computer processing. Often, this information bears little resemblance to visual features that humans use in interpreting the content of an image. Examples of the type of information used in machine perception are statistical moments, Fourier transform coefficients, and multidimensional distance measures. Typical problems in machine perception that routinely utilize image processing techniques are automatic character recognition, industrial machine vision for product assembly and inspection, military recognition, automatic processing of fingerprints, screening of X-rays and blood samples, and machine processing of aerial and satellite imagery for weather prediction and environmental assessment. The continuing decline in the ratio of computer price to performance, and the expansion of networking and communication bandwidth via the internet, have created unprecedented opportunities for continued growth of digital image processing. Some of these application areas will be illustrated in the following section.

1.3 EXAMPLES OF FIELDS THAT USE DIGITAL IMAGE PROCESSING

Today, there is almost no area of technical endeavor that is not impacted in some way by digital image processing. We can cover only a few of these applications in the context and space of the current discussion. However, limited as it is, the material presented in this section will leave no doubt in your mind regarding the breadth and importance of digital image processing. We show in this section numerous areas of application, each of which routinely utilizes the digital image processing techniques developed in the following chapters. Many of the images shown in this section are used later in one or more of the examples given in the book. Most images shown are digital images.

The areas of application of digital image processing are so varied that some form of organization is desirable in attempting to capture the breadth of this field. One of the simplest ways to develop a basic understanding of the extent of image processing applications is to categorize images according to their source (e.g., X-ray, visual, infrared, and so on). The principal energy source for images in use today is the electromagnetic energy spectrum. Other important sources of energy include acoustic, ultrasonic, and electronic (in the form of electron beams used in electron microscopy). Synthetic images, used for modeling and visualization, are generated by computer. In this section we will discuss briefly how images are generated in these various categories, and the areas in which they are applied. Methods for converting images into digital form will be discussed in the next chapter.

Images based on radiation from the EM spectrum are the most familiar, especially images in the X-ray and visual bands of the spectrum. Electromagnetic waves can be conceptualized as propagating sinusoidal waves of varying wavelengths, or they can be thought of as a stream of massless particles, each traveling in a wavelike pattern and moving at the speed of light. Each massless particle contains a certain amount (or bundle) of energy. Each bundle of energy is called a *photon*. If spectral

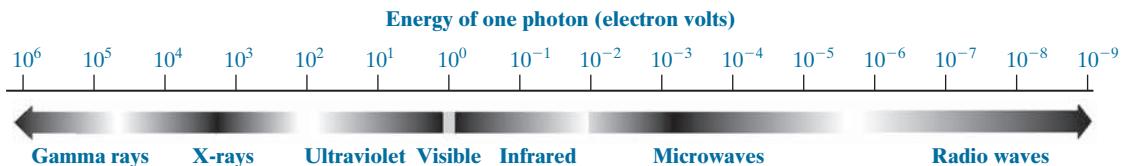


FIGURE 1.5 The electromagnetic spectrum arranged according to energy per photon.

bands are grouped according to energy per photon, we obtain the spectrum shown in Fig. 1.5, ranging from gamma rays (highest energy) at one end to radio waves (lowest energy) at the other. The bands are shown shaded to convey the fact that bands of the EM spectrum are not distinct, but rather transition smoothly from one to the other.

GAMMA-RAY IMAGING

Major uses of imaging based on gamma rays include nuclear medicine and astronomical observations. In nuclear medicine, the approach is to inject a patient with a radioactive isotope that emits gamma rays as it decays. Images are produced from the emissions collected by gamma-ray detectors. Figure 1.6(a) shows an image of a complete bone scan obtained by using gamma-ray imaging. Images of this sort are used to locate sites of bone pathology, such as infections or tumors. Figure 1.6(b) shows another major modality of nuclear imaging called *positron emission tomography* (PET). The principle is the same as with X-ray tomography, mentioned briefly in Section 1.2. However, instead of using an external source of X-ray energy, the patient is given a radioactive isotope that emits positrons as it decays. When a positron meets an electron, both are annihilated and two gamma rays are given off. These are detected and a tomographic image is created using the basic principles of tomography. The image shown in Fig. 1.6(b) is one sample of a sequence that constitutes a 3-D rendition of the patient. This image shows a tumor in the brain and another in the lung, easily visible as small white masses.

A star in the constellation of Cygnus exploded about 15,000 years ago, generating a superheated, stationary gas cloud (known as the Cygnus Loop) that glows in a spectacular array of colors. Figure 1.6(c) shows an image of the Cygnus Loop in the gamma-ray band. Unlike the two examples in Figs. 1.6(a) and (b), this image was obtained using the natural radiation of the object being imaged. Finally, Fig. 1.6(d) shows an image of gamma radiation from a valve in a nuclear reactor. An area of strong radiation is seen in the lower left side of the image.

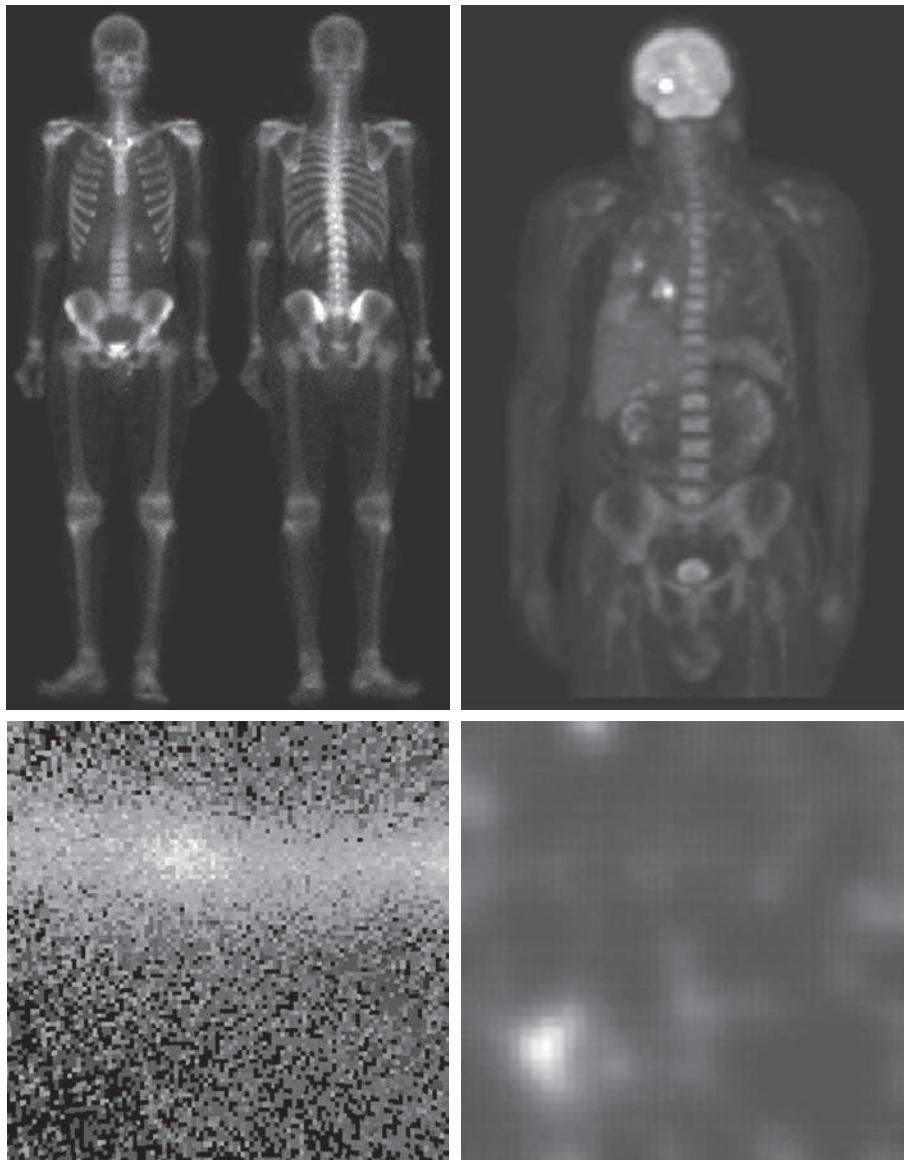
X-RAY IMAGING

X-rays are among the oldest sources of EM radiation used for imaging. The best known use of X-rays is medical diagnostics, but they are also used extensively in industry and other areas, such as astronomy. X-rays for medical and industrial imaging are generated using an X-ray tube, which is a vacuum tube with a cathode and anode. The cathode is heated, causing free electrons to be released. These electrons flow at high speed to the positively charged anode. When the electrons strike a

a b
c d

FIGURE 1.6

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

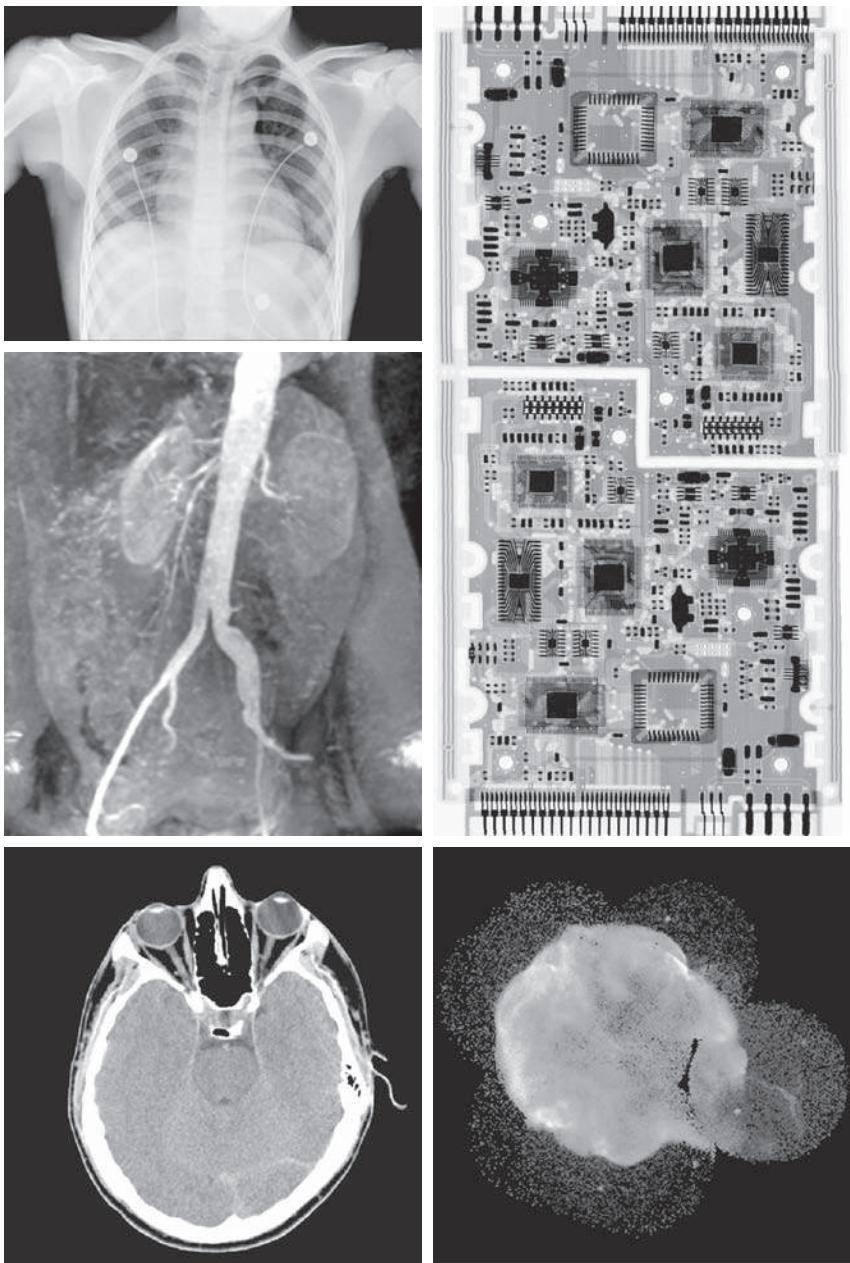


nucleus, energy is released in the form of X-ray radiation. The energy (penetrating power) of X-rays is controlled by a voltage applied across the anode, and by a current applied to the filament in the cathode. Figure 1.7(a) shows a familiar chest X-ray generated simply by placing the patient between an X-ray source and a film sensitive to X-ray energy. The intensity of the X-rays is modified by absorption as they pass through the patient, and the resulting energy falling on the film develops it, much in the same way that light develops photographic film. In digital radiography,

a	d
c	
b	e

FIGURE 1.7

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



digital images are obtained by one of two methods: (1) by digitizing X-ray films; or; (2) by having the X-rays that pass through the patient fall directly onto devices (such as a phosphor screen) that convert X-rays to light. The light signal in turn is captured by a light-sensitive digitizing system. We will discuss digitization in more detail in Chapters 2 and 4.

Angiography is another major application in an area called contrast enhancement radiography. This procedure is used to obtain images of blood vessels, called *angiograms*. A catheter (a small, flexible, hollow tube) is inserted, for example, into an artery or vein in the groin. The catheter is threaded into the blood vessel and guided to the area to be studied. When the catheter reaches the site under investigation, an X-ray contrast medium is injected through the tube. This enhances the contrast of the blood vessels and enables a radiologist to see any irregularities or blockages. Figure 1.7(b) shows an example of an aortic angiogram. The catheter can be seen being inserted into the large blood vessel on the lower left of the picture. Note the high contrast of the large vessel as the contrast medium flows up in the direction of the kidneys, which are also visible in the image. As we will discuss further in Chapter 2, angiography is a major area of digital image processing, where image subtraction is used to further enhance the blood vessels being studied.

Another important use of X-rays in medical imaging is computerized axial tomography (CAT). Due to their resolution and 3-D capabilities, CAT scans revolutionized medicine from the moment they first became available in the early 1970s. As noted in Section 1.2, each CAT image is a “slice” taken perpendicularly through the patient. Numerous slices are generated as the patient is moved in a longitudinal direction. The ensemble of such images constitutes a 3-D rendition of the inside of the body, with the longitudinal resolution being proportional to the number of slice images taken. Figure 1.7(c) shows a typical CAT slice image of a human head.

Techniques similar to the ones just discussed, but generally involving higher energy X-rays, are applicable in industrial processes. Figure 1.7(d) shows an X-ray image of an electronic circuit board. Such images, representative of literally hundreds of industrial applications of X-rays, are used to examine circuit boards for flaws in manufacturing, such as missing components or broken traces. Industrial CAT scans are useful when the parts can be penetrated by X-rays, such as in plastic assemblies, and even large bodies, such as solid-propellant rocket motors. Figure 1.7(e) shows an example of X-ray imaging in astronomy. This image is the Cygnus Loop of Fig. 1.6(c), but imaged in the X-ray band.

IMAGING IN THE ULTRAVIOLET BAND

Applications of ultraviolet “light” are varied. They include lithography, industrial inspection, microscopy, lasers, biological imaging, and astronomical observations. We illustrate imaging in this band with examples from microscopy and astronomy.

Ultraviolet light is used in fluorescence microscopy, one of the fastest growing areas of microscopy. Fluorescence is a phenomenon discovered in the middle of the nineteenth century, when it was first observed that the mineral fluorspar fluoresces when ultraviolet light is directed upon it. The ultraviolet light itself is not visible, but when a photon of ultraviolet radiation collides with an electron in an atom of a fluorescent material, it elevates the electron to a higher energy level. Subsequently, the excited electron relaxes to a lower level and emits light in the form of a lower-energy photon in the visible (red) light region. Important tasks performed with a fluorescence microscope are to use an excitation light to irradiate a prepared specimen, and then to separate the much weaker radiating fluorescent light from the brighter



a b c

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

excitation light. Thus, only the emission light reaches the eye or other detector. The resulting fluorescing areas shine against a dark background with sufficient contrast to permit detection. The darker the background of the nonfluorescing material, the more efficient the instrument.

Fluorescence microscopy is an excellent method for studying materials that can be made to fluoresce, either in their natural form (primary fluorescence) or when treated with chemicals capable of fluorescing (secondary fluorescence). Figures 1.8(a) and (b) show results typical of the capability of fluorescence microscopy. Figure 1.8(a) shows a fluorescence microscope image of normal corn, and Fig. 1.8(b) shows corn infected by “smut,” a disease of cereals, corn, grasses, onions, and sorghum that can be caused by any one of more than 700 species of parasitic fungi. Corn smut is particularly harmful because corn is one of the principal food sources in the world. As another illustration, Fig. 1.8(c) shows the Cygnus Loop imaged in the high-energy region of the ultraviolet band.

IMAGING IN THE VISIBLE AND INFRARED BANDS

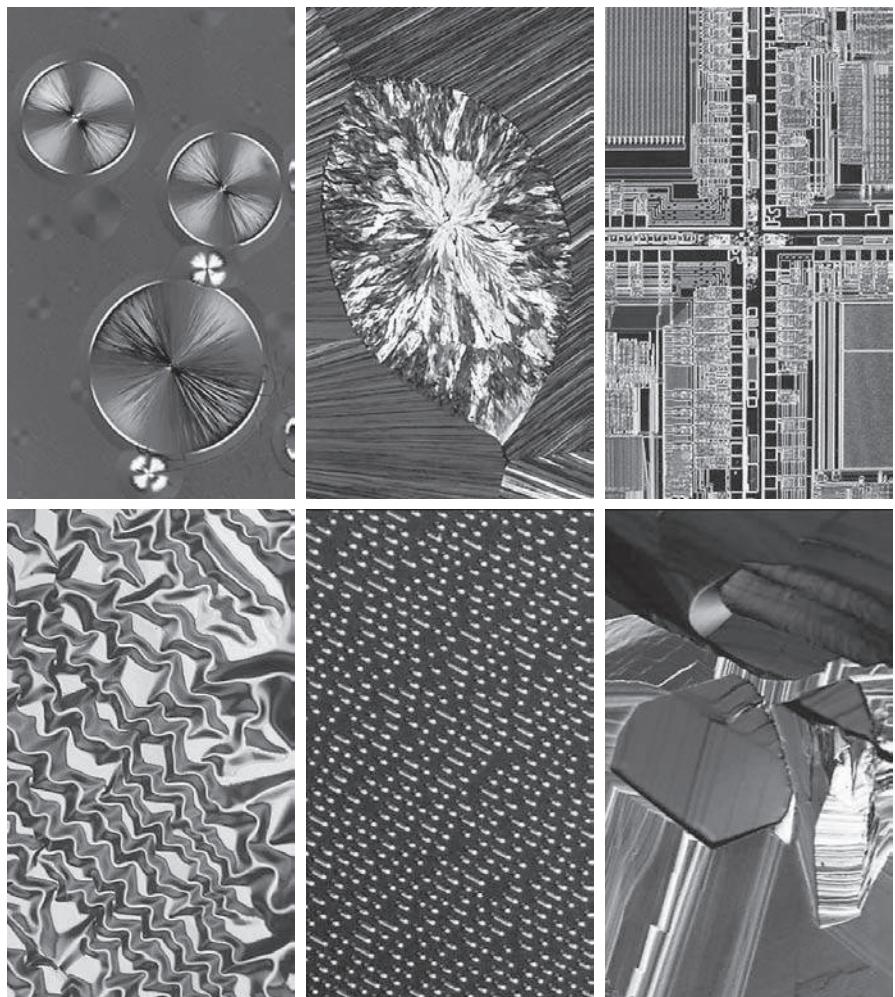
Considering that the visual band of the electromagnetic spectrum is the most familiar in all our activities, it is not surprising that imaging in this band outweighs by far all the others in terms of breadth of application. The infrared band often is used in conjunction with visual imaging, so we have grouped the visible and infrared bands in this section for the purpose of illustration. We consider in the following discussion applications in light microscopy, astronomy, remote sensing, industry, and law enforcement.

Figure 1.9 shows several examples of images obtained with a light microscope. The examples range from pharmaceuticals and microinspection to materials characterization. Even in microscopy alone, the application areas are too numerous to detail here. It is not difficult to conceptualize the types of processes one might apply to these images, ranging from enhancement to measurements.

a	b	c
d	e	f

FIGURE 1.9

Examples of light microscopy images.
 (a) Taxol (anticancer agent), magnified 250 ×.
 (b) Cholesterol—40 ×.
 (c) Microprocessor—60 ×.
 (d) Nickel oxide thin film—600 ×.
 (e) Surface of audio CD—1750 ×.
 (f) Organic superconductor—450 ×.
 (Images courtesy of Dr. Michael W. Davidson, Florida State University.)



Another major area of visual processing is remote sensing, which usually includes several bands in the visual and infrared regions of the spectrum. Table 1.1 shows the so-called *thematic bands* in NASA's LANDSAT satellites. The primary function of LANDSAT is to obtain and transmit images of the Earth from space, for purposes of monitoring environmental conditions on the planet. The bands are expressed in terms of wavelength, with $1\mu\text{m}$ being equal to 10^{-6} m (we will discuss the wavelength regions of the electromagnetic spectrum in more detail in Chapter 2). Note the characteristics and uses of each band in Table 1.1.

In order to develop a basic appreciation for the power of this type of multispectral imaging, consider Fig. 1.10, which shows one image for each of the spectral bands in Table 1.1. The area imaged is Washington D.C., which includes features such as buildings, roads, vegetation, and a major river (the Potomac) going though the city.

TABLE 1.1

Thematic bands of NASA's LANDSAT satellite.

Band No.	Name	Wavelength (μm)	Characteristics and Uses
1	Visible blue	0.45–0.52	Maximum water penetration
2	Visible green	0.53–0.61	Measures plant vigor
3	Visible red	0.63–0.69	Vegetation discrimination
4	Near infrared	0.78–0.90	Biomass and shoreline mapping
5	Middle infrared	1.55–1.75	Moisture content: soil/vegetation
6	Thermal infrared	10.4–12.5	Soil moisture; thermal mapping
7	Short-wave infrared	2.09–2.35	Mineral mapping

Images of population centers are used over time to assess population growth and shift patterns, pollution, and other factors affecting the environment. The differences between visual and infrared image features are quite noticeable in these images. Observe, for example, how well defined the river is from its surroundings in Bands 4 and 5.

Weather observation and prediction also are major applications of multispectral imaging from satellites. For example, Fig. 1.11 is an image of Hurricane Katrina, one of the most devastating storms in recent memory in the Western Hemisphere. This image was taken by a National Oceanographic and Atmospheric Administration (NOAA) satellite using sensors in the visible and infrared bands. The eye of the hurricane is clearly visible in this image.

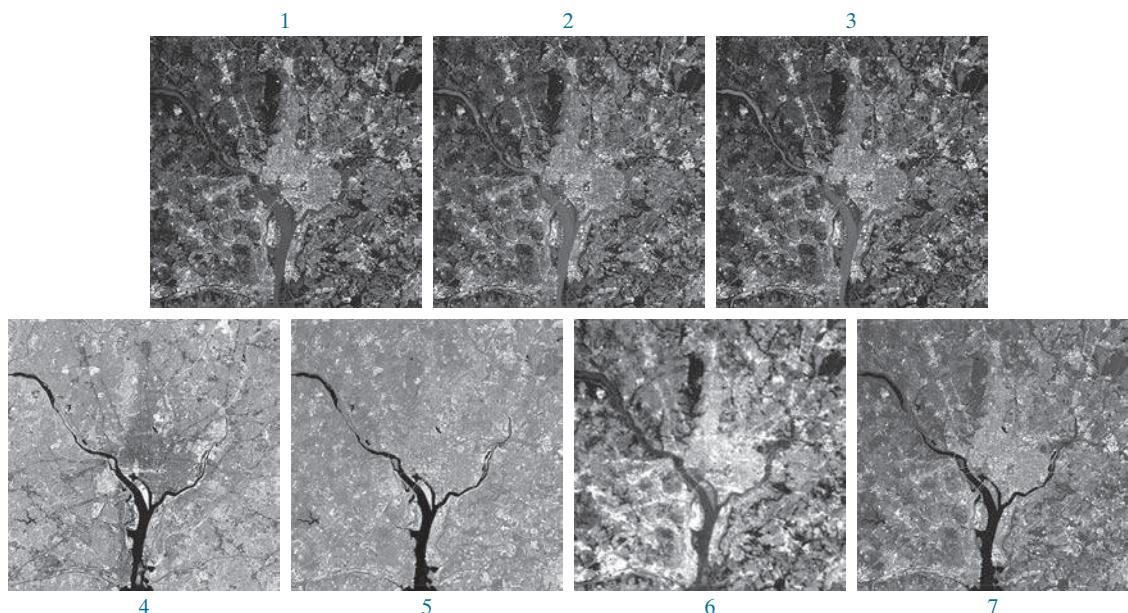
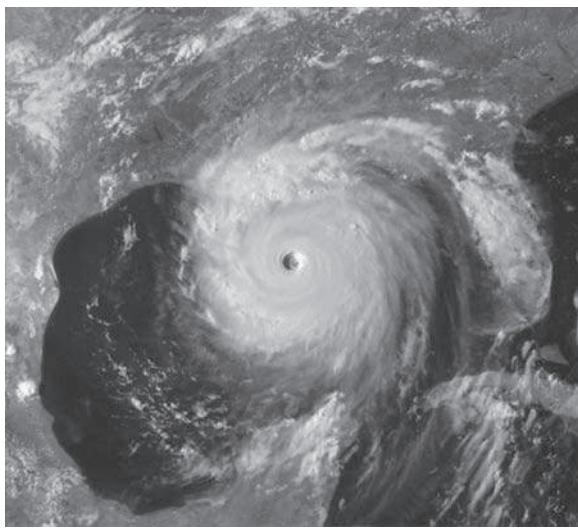


FIGURE 1.10 LANDSAT satellite images of the Washington, D.C. area. The numbers refer to the thematic bands in Table 1.1. (Images courtesy of NASA.)

FIGURE 1.11

Satellite image of Hurricane Katrina taken on August 29, 2005.

(Courtesy of NOAA.)



Figures 1.12 and 1.13 show an application of infrared imaging. These images are part of the Nighttime Lights of the World data set, which provides a global inventory of human settlements. The images were generated by an infrared imaging system mounted on a NOAA/DMSP (Defense Meteorological Satellite Program) satellite. The infrared system operates in the band 10.0 to 13.4 μm , and has the unique capability to observe faint sources of visible, near infrared emissions present on the Earth's surface, including cities, towns, villages, gas flares, and fires. Even without formal training in image processing, it is not difficult to imagine writing a computer program that would use these images to estimate the relative percent of total electrical energy used by various regions of the world.

A major area of imaging in the visible spectrum is in automated visual inspection of manufactured goods. Figure 1.14 shows some examples. Figure 1.14(a) is a controller board for a CD-ROM drive. A typical image processing task with products such as this is to inspect them for missing parts (the black square on the top, right quadrant of the image is an example of a missing component).

Figure 1.14(b) is an imaged pill container. The objective here is to have a machine look for missing, incomplete, or deformed pills. Figure 1.14(c) shows an application in which image processing is used to look for bottles that are not filled up to an acceptable level. Figure 1.14(d) shows a clear plastic part with an unacceptable number of air pockets in it. Detecting anomalies like these is a major theme of industrial inspection that includes other products, such as wood and cloth. Figure 1.14(e) shows a batch of cereal during inspection for color and the presence of anomalies such as burned flakes. Finally, Fig. 1.14(f) shows an image of an intraocular implant (replacement lens for the human eye). A “structured light” illumination technique was used to highlight deformations toward the center of the lens, and other imperfections. For example, the markings at 1 o'clock and 5 o'clock are tweezer damage. Most of the other small speckle detail is debris. The objective in this type of inspection is to find damaged or incorrectly manufactured implants automatically, prior to packaging.

FIGURE 1.12

Infrared satellite images of the Americas. The small shaded map is provided for reference.
(Courtesy of NOAA.)



Figure 1.15 illustrates some additional examples of image processing in the visible spectrum. Figure 1.15(a) shows a thumb print. Images of fingerprints are routinely processed by computer, either to enhance them or to find features that aid in the automated search of a database for potential matches. Figure 1.15(b) shows an image of paper currency. Applications of digital image processing in this area

FIGURE 1.13

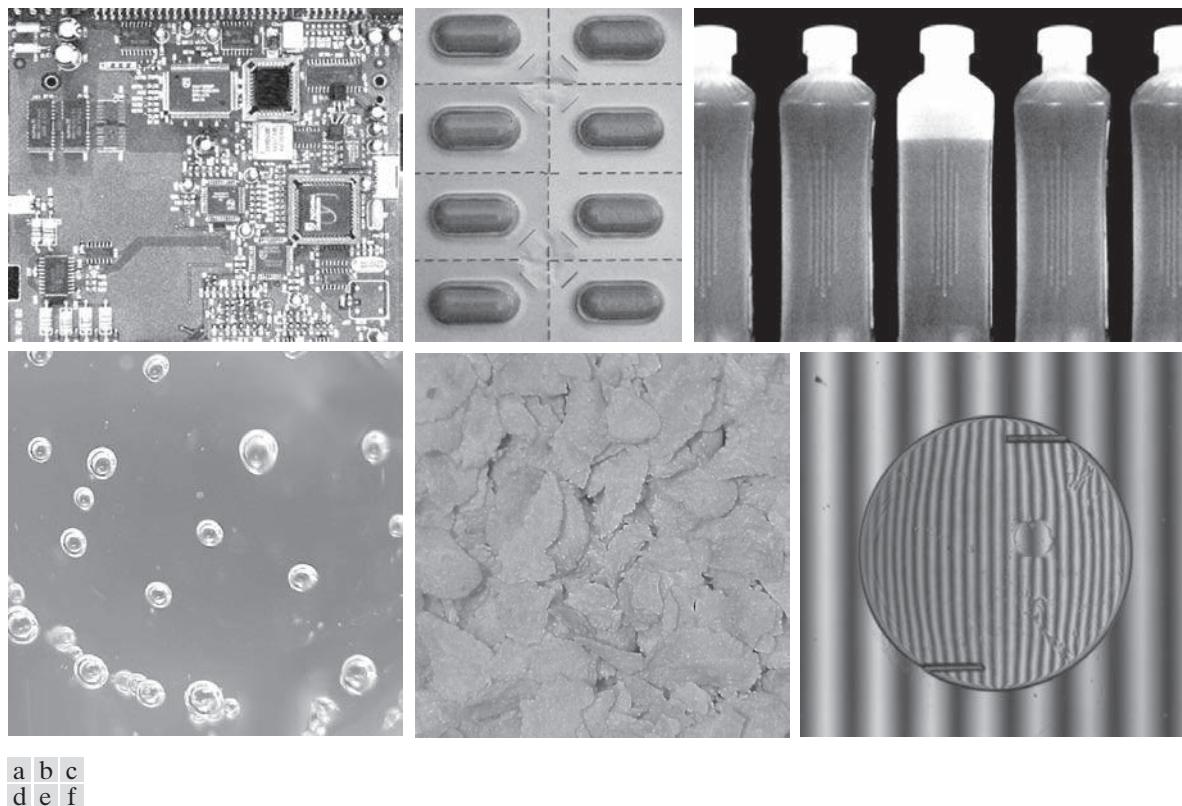
Infrared satellite images of the remaining populated parts of the world. The small shaded map is provided for reference.
(Courtesy of NOAA.)



include automated counting and, in law enforcement, the reading of the serial number for the purpose of tracking and identifying currency bills. The two vehicle images shown in Figs. 1.15(c) and (d) are examples of automated license plate reading. The light rectangles indicate the area in which the imaging system detected the plate. The black rectangles show the results of automatically reading the plate content by the system. License plate and other applications of character recognition are used extensively for traffic monitoring and surveillance.

IMAGING IN THE MICROWAVE BAND

The principal application of imaging in the microwave band is radar. The unique feature of imaging radar is its ability to collect data over virtually any region at any time, regardless of weather or ambient lighting conditions. Some radar waves can penetrate clouds, and under certain conditions, can also see through vegetation, ice, and dry sand. In many cases, radar is the only way to explore inaccessible regions of the Earth's surface. An imaging radar works like a flash camera in that it provides its own illumination (microwave pulses) to illuminate an area on the ground and



a b c
d e f

FIGURE 1.14 Some examples of manufactured goods checked using digital image processing. (a) Circuit board controller. (b) Packaged pills. (c) Bottles. (d) Air bubbles in a clear plastic product. (e) Cereal. (f) Image of intraocular implant. (Figure (f) courtesy of Mr. Pete Sites, Perceptics Corporation.)

take a snapshot image. Instead of a camera lens, a radar uses an antenna and digital computer processing to record its images. In a radar image, one can see only the microwave energy that was reflected back toward the radar antenna.

Figure 1.16 shows a spaceborne radar image covering a rugged mountainous area of southeast Tibet, about 90 km east of the city of Lhasa. In the lower right corner is a wide valley of the Lhasa River, which is populated by Tibetan farmers and yak herders, and includes the village of Menba. Mountains in this area reach about 5800 m (19,000 ft) above sea level, while the valley floors lie about 4300 m (14,000 ft) above sea level. Note the clarity and detail of the image, unencumbered by clouds or other atmospheric conditions that normally interfere with images in the visual band.

IMAGING IN THE RADIO BAND

As in the case of imaging at the other end of the spectrum (gamma rays), the major applications of imaging in the radio band are in medicine and astronomy. In medicine, radio waves are used in magnetic resonance imaging (MRI). This technique places a

a b
c
d

FIGURE 1.15

Some additional examples of imaging in the visible spectrum.
 (a) Thumb print.
 (b) Paper currency.
 (c) and (d) Automated license plate reading.
 (Figure (a) courtesy of the National Institute of Standards and Technology. Figures (c) and (d) courtesy of Dr. Juan Herrera, Perceptics Corporation.)



patient in a powerful magnet and passes radio waves through the individual's body in short pulses. Each pulse causes a responding pulse of radio waves to be emitted by the patient's tissues. The location from which these signals originate and their strength are determined by a computer, which produces a two-dimensional image of a section of the patient. MRI can produce images in any plane. Figure 1.17 shows MRI images of a human knee and spine.

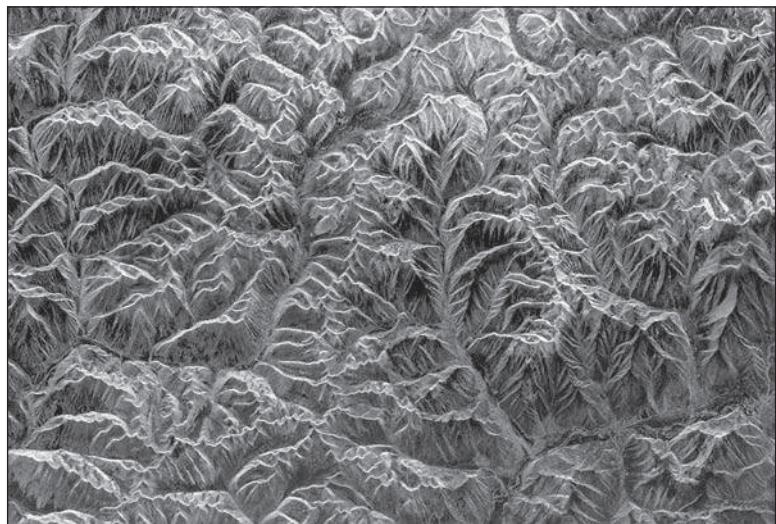
The rightmost image in Fig. 1.18 is an image of the Crab Pulsar in the radio band. Also shown for an interesting comparison are images of the same region, but taken in most of the bands discussed earlier. Observe that each image gives a totally different "view" of the pulsar.

OTHER IMAGING MODALITIES

Although imaging in the electromagnetic spectrum is dominant by far, there are a number of other imaging modalities that are also important. Specifically, we discuss

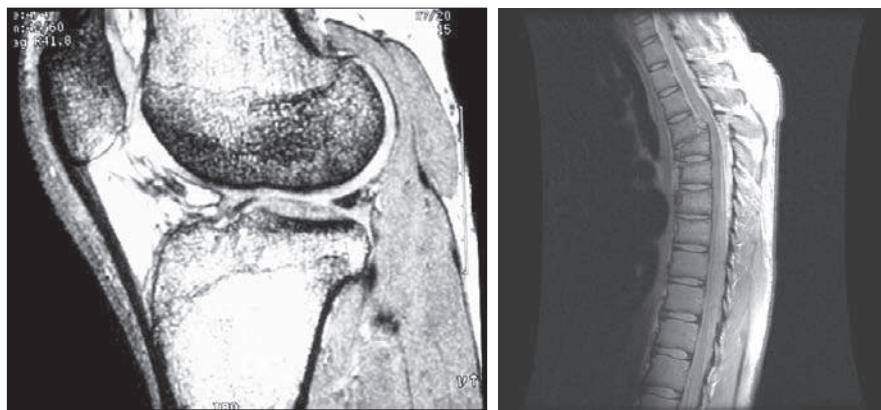
FIGURE 1.16

Spaceborne radar image of mountainous region in southeast Tibet. (Courtesy of NASA.)



in this section acoustic imaging, electron microscopy, and synthetic (computer-generated) imaging.

Imaging using “sound” finds application in geological exploration, industry, and medicine. Geological applications use sound in the low end of the sound spectrum (hundreds of Hz) while imaging in other areas use ultrasound (millions of Hz). The most important commercial applications of image processing in geology are in mineral and oil exploration. For image acquisition over land, one of the main approaches is to use a large truck and a large flat steel plate. The plate is pressed on the ground by



a b

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

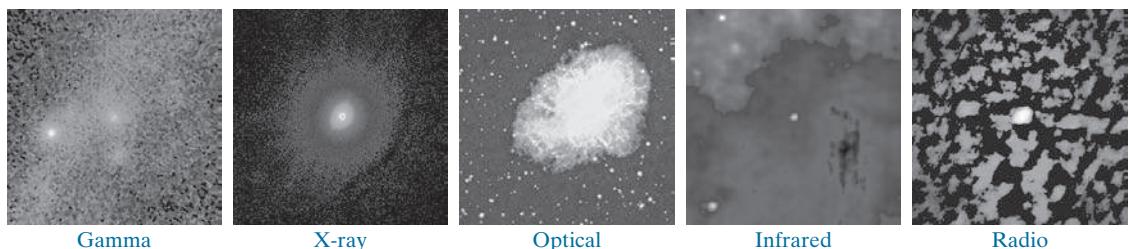


FIGURE 1.18 Images of the Crab Pulsar (in the center of each image) covering the electromagnetic spectrum. (Courtesy of NASA.)

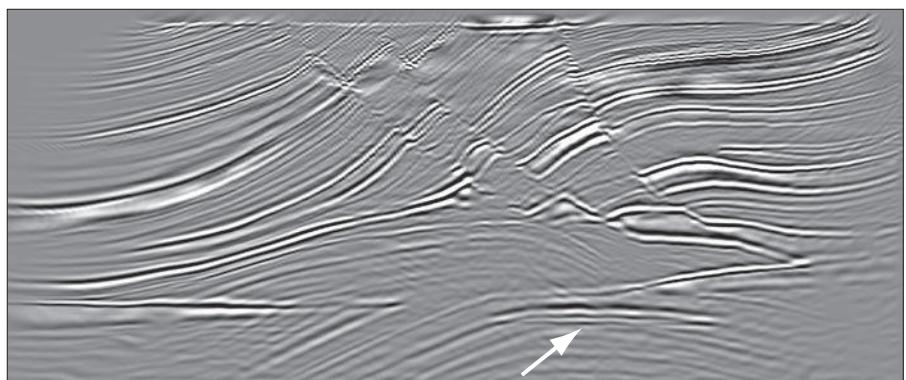
the truck, and the truck is vibrated through a frequency spectrum up to 100 Hz. The strength and speed of the returning sound waves are determined by the composition of the Earth below the surface. These are analyzed by computer, and images are generated from the resulting analysis.

For marine image acquisition, the energy source consists usually of two air guns towed behind a ship. Returning sound waves are detected by hydrophones placed in cables that are either towed behind the ship, laid on the bottom of the ocean, or hung from buoys (vertical cables). The two air guns are alternately pressurized to ~2000 psi and then set off. The constant motion of the ship provides a transversal direction of motion that, together with the returning sound waves, is used to generate a 3-D map of the composition of the Earth below the bottom of the ocean.

Figure 1.19 shows a cross-sectional image of a well-known 3-D model against which the performance of seismic imaging algorithms is tested. The arrow points to a hydrocarbon (oil and/or gas) trap. This target is brighter than the surrounding layers because the change in density in the target region is larger. Seismic interpreters look for these “bright spots” to find oil and gas. The layers above also are bright, but their brightness does not vary as strongly across the layers. Many seismic reconstruction algorithms have difficulty imaging this target because of the faults above it.

Although ultrasound imaging is used routinely in manufacturing, the best known applications of this technique are in medicine, especially in obstetrics, where fetuses are imaged to determine the health of their development. A byproduct of this

FIGURE 1.19
Cross-sectional image of a seismic model. The arrow points to a hydrocarbon (oil and/or gas) trap. (Courtesy of Dr. Curtis Ober, Sandia National Laboratories.)



examination is determining the sex of the baby. Ultrasound images are generated using the following basic procedure:

1. The ultrasound system (a computer, ultrasound probe consisting of a source, a receiver, and a display) transmits high-frequency (1 to 5 MHz) sound pulses into the body.
2. The sound waves travel into the body and hit a boundary between tissues (e.g., between fluid and soft tissue, soft tissue and bone). Some of the sound waves are reflected back to the probe, while some travel on further until they reach another boundary and are reflected.
3. The reflected waves are picked up by the probe and relayed to the computer.
4. The machine calculates the distance from the probe to the tissue or organ boundaries using the speed of sound in tissue (1540 m/s) and the time of each echo's return.
5. The system displays the distances and intensities of the echoes on the screen, forming a two-dimensional image.

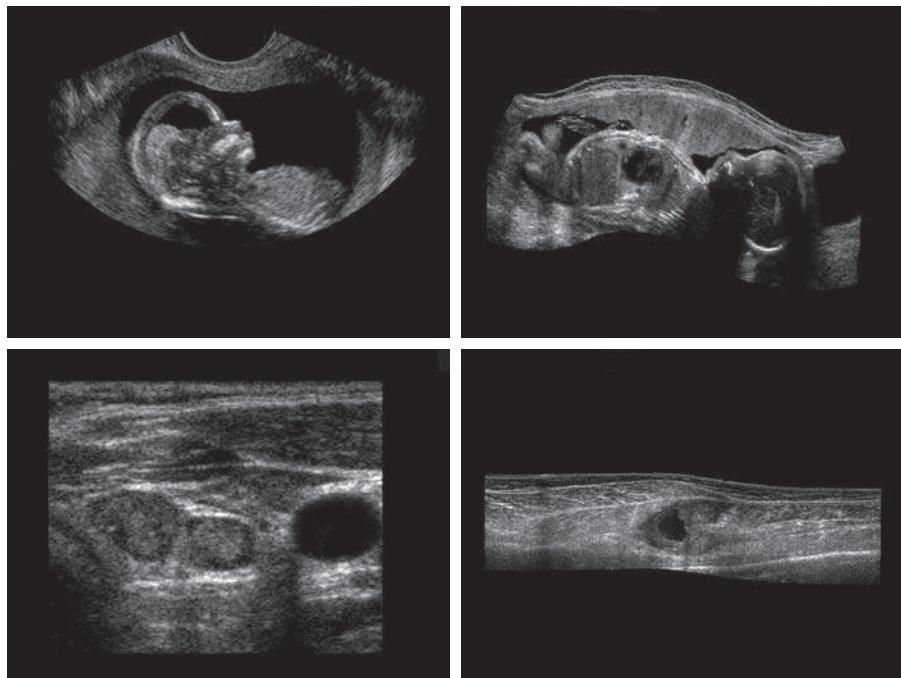
In a typical ultrasound image, millions of pulses and echoes are sent and received each second. The probe can be moved along the surface of the body and angled to obtain various views. Figure 1.20 shows several examples of medical uses of ultrasound.

We continue the discussion on imaging modalities with some examples of electron microscopy. Electron microscopes function as their optical counterparts, except

a
b
c
d

FIGURE 1.20

Examples of ultrasound imaging. (a) A fetus. (b) Another view of the fetus. (c) Thyroids. (d) Muscle layers showing lesion.
(Courtesy of Siemens Medical Systems, Inc., Ultrasound Group.)



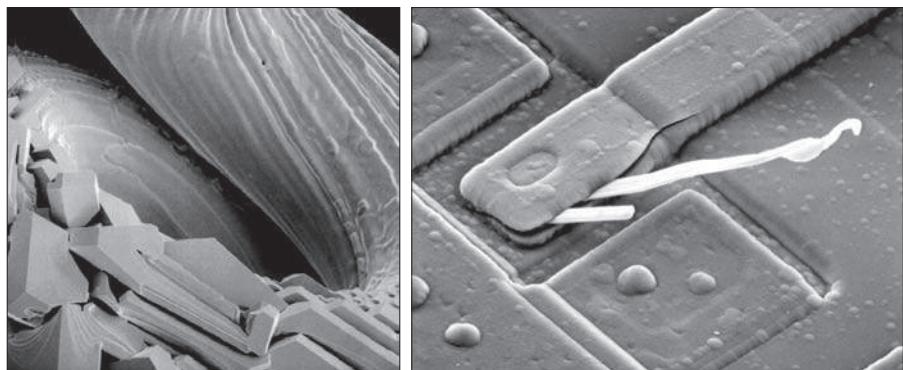
that they use a focused beam of electrons instead of light to image a specimen. The operation of electron microscopes involves the following basic steps: A stream of electrons is produced by an electron source and accelerated toward the specimen using a positive electrical potential. This stream is confined and focused using metal apertures and magnetic lenses into a thin, monochromatic beam. This beam is focused onto the sample using a magnetic lens. Interactions occur inside the irradiated sample, affecting the electron beam. These interactions and effects are detected and transformed into an image, much in the same way that light is reflected from, or absorbed by, objects in a scene. These basic steps are carried out in all electron microscopes.

A *transmission electron microscope* (TEM) works much like a slide projector. A projector transmits a beam of light through a slide; as the light passes through the slide, it is modulated by the contents of the slide. This transmitted beam is then projected onto the viewing screen, forming an enlarged image of the slide. TEMs work in the same way, except that they shine a beam of electrons through a specimen (analogous to the slide). The fraction of the beam transmitted through the specimen is projected onto a phosphor screen. The interaction of the electrons with the phosphor produces light and, therefore, a viewable image. A *scanning electron microscope* (SEM), on the other hand, actually scans the electron beam and records the interaction of beam and sample at each location. This produces one dot on a phosphor screen. A complete image is formed by a raster scan of the beam through the sample, much like a TV camera. The electrons interact with a phosphor screen and produce light. SEMs are suitable for “bulky” samples, while TEMs require very thin samples.

Electron microscopes are capable of very high magnification. While light microscopy is limited to magnifications on the order of $1000\times$, electron microscopes can achieve magnification of $10,000\times$ or more. Figure 1.21 shows two SEM images of specimen failures due to thermal overload.

We conclude the discussion of imaging modalities by looking briefly at images that are not obtained from physical objects. Instead, they are generated by computer. Fractals are striking examples of computer-generated images. Basically, a fractal is nothing more than an iterative reproduction of a basic pattern according to some mathematical rules. For instance, tiling is one of the simplest ways to generate a fractal image. A square can be subdivided into four square subregions, each of which can be further subdivided into four smaller square regions, and so on. Depending on the complexity of the rules for filling each subsquare, some beautiful tile images can be generated using this method. Of course, the geometry can be arbitrary. For instance, the fractal image could be grown radially out of a center point. Figure 1.22(a) shows a fractal grown in this way. Figure 1.22(b) shows another fractal (a “moonscape”) that provides an interesting analogy to the images of space used as illustrations in some of the preceding sections.

A more structured approach to image generation by computer lies in 3-D modeling. This is an area that provides an important intersection between image processing and computer graphics, and is the basis for many 3-D visualization systems (e.g., flight simulators). Figures 1.22(c) and (d) show examples of computer-generated images. Because the original object is created in 3-D, images can be generated in any



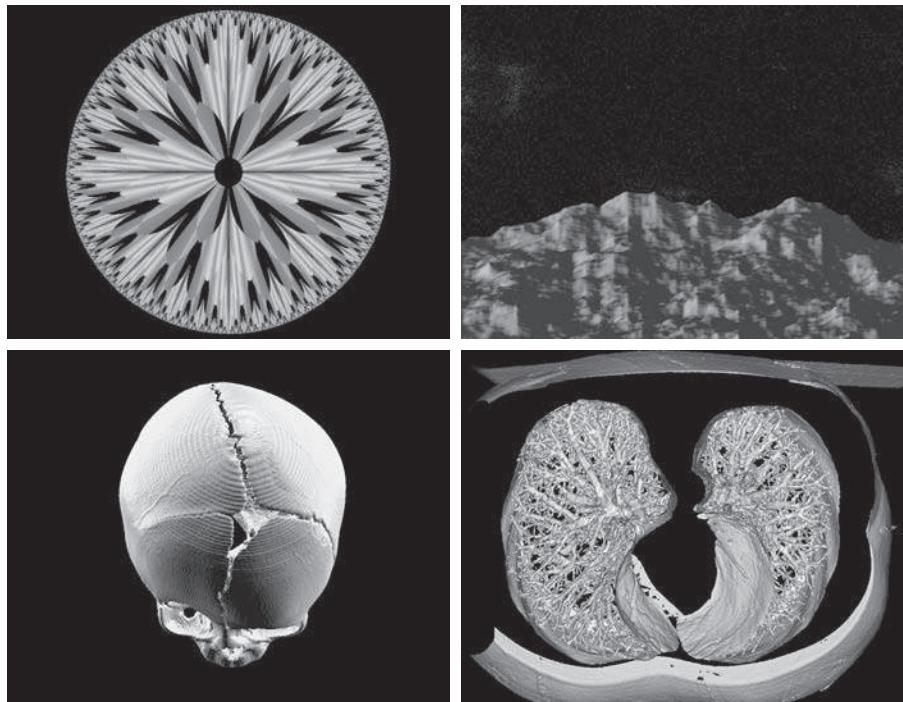
a b

FIGURE 1.21 (a) $250\times$ SEM image of a tungsten filament following thermal failure (note the shattered pieces on the lower left). (b) $2500\times$ SEM image of a damaged integrated circuit. The white fibers are oxides resulting from thermal destruction. (Figure (a) courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene; (b) courtesy of Dr. J. M. Hudak, McMaster University, Hamilton, Ontario, Canada.)

perspective from plane projections of the 3-D volume. Images of this type can be used for medical training and for a host of other applications, such as criminal forensics and special effects.

a b
c d

FIGURE 1.22
(a) and (b) Fractal images.
(c) and (d) Images generated from 3-D computer models of the objects shown.
(Figures (a) and (b) courtesy of Ms. Melissa D. Binde, Swarthmore College; (c) and (d) courtesy of NASA.)



1.4 FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING

It is helpful to divide the material covered in the following chapters into the two broad categories defined in Section 1.1: methods whose input and output are images, and methods whose inputs may be images, but whose outputs are attributes extracted from those images. This organization is summarized in Fig. 1.23. The diagram does not imply that every process is applied to an image. Rather, the intention is to convey an idea of all the methodologies that can be applied to images for different purposes, and possibly with different objectives. The discussion in this section may be viewed as a brief overview of the material in the remainder of the book.

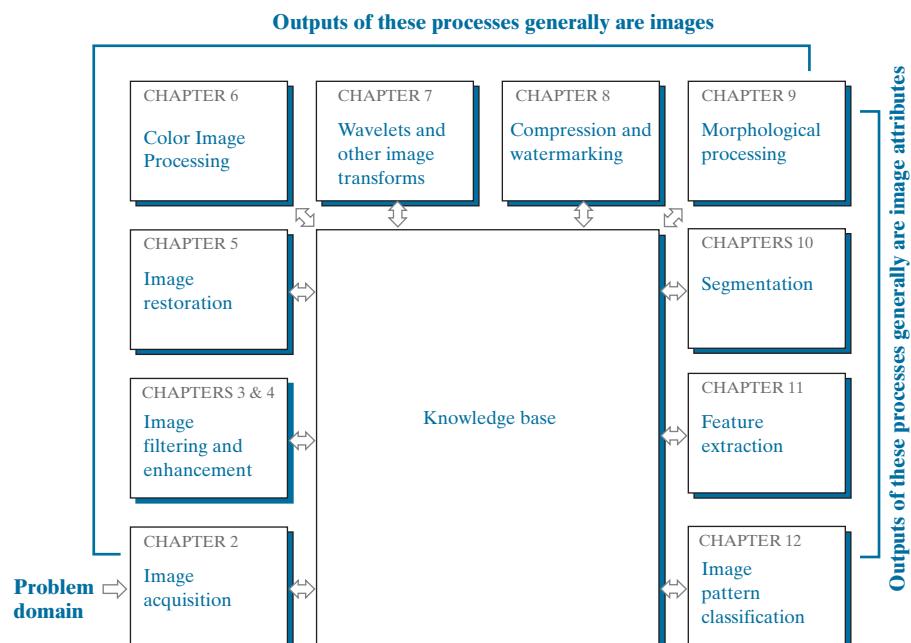
Image acquisition is the first process in Fig. 1.23. The discussion in Section 1.3 gave some hints regarding the origin of digital images. This topic will be considered in much more detail in Chapter 2, where we also introduce a number of basic digital image concepts that are used throughout the book. Acquisition could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves preprocessing, such as scaling.

Image enhancement is the process of manipulating an image so the result is more suitable than the original for a specific application. The word *specific* is important here, because it establishes at the outset that enhancement techniques are problem oriented. Thus, for example, a method that is quite useful for enhancing X-ray images may not be the best approach for enhancing satellite images taken in the infrared band of the electromagnetic spectrum.

There is no general “theory” of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular

FIGURE 1.23

Fundamental steps in digital image processing. The chapter(s) indicated in the boxes is where the material described in the box is discussed.



method works. Enhancement techniques are so varied, and use so many different image processing approaches, that it is difficult to assemble a meaningful body of techniques suitable for enhancement in one chapter without extensive background development. For this reason, and also because beginners in the field of image processing generally find enhancement applications visually appealing, interesting, and relatively simple to understand, we will use image enhancement as examples when introducing new concepts in parts of Chapter 2 and in Chapters 3 and 4. The material in the latter two chapters span many of the methods used traditionally for image enhancement. Therefore, using examples from image enhancement to introduce new image processing methods developed in these early chapters not only saves having an extra chapter in the book dealing with image enhancement but, more importantly, is an effective approach for introducing newcomers to the details of processing techniques early in the book. However, as you will see in progressing through the rest of the book, the material developed in Chapters 3 and 4 is applicable to a much broader class of problems than just image enhancement.

Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation. Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result.

Color image processing is an area that has been gaining in importance because of the significant increase in the use of digital images over the internet. Chapter 6 covers a number of fundamental concepts in color models and basic color processing in a digital domain. Color is used also as the basis for extracting features of interest in an image.

Wavelets are the foundation for representing images in various degrees of resolution. In particular, this material is used in the book for image data compression and for pyramidal representation, in which images are subdivided successively into smaller regions. The material in Chapters 4 and 5 is based mostly on the Fourier transform. In addition to wavelets, we will also discuss in Chapter 7 a number of other transforms that are used routinely in image processing.

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

Morphological processing deals with tools for extracting image components that are useful in the representation and description of shape. The material in this chapter begins a transition from processes that output images to processes that output image attributes, as indicated in Section 1.1.

Segmentation partitions an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image

processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually. On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely automated object classification is to succeed.

Feature extraction almost always follows the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region (i.e., the set of pixels separating one image region from another) or all the points in the region itself. Feature extraction consists of feature detection and feature description. *Feature detection* refers to finding the features in an image, region, or boundary. *Feature description* assigns quantitative attributes to the detected features. For example, we might detect corners in a region, and describe those corners by their orientation and location; both of these descriptors are quantitative attributes. Feature processing methods discussed in this chapter are subdivided into three principal categories, depending on whether they are applicable to boundaries, regions, or whole images. Some features are applicable to more than one category. Feature descriptors should be as insensitive as possible to variations in parameters such as scale, translation, rotation, illumination, and viewpoint.

Image pattern classification is the process that assigns a label (e.g., “vehicle”) to an object based on its feature descriptors. In the last chapter of the book, we will discuss methods of image pattern classification ranging from “classical” approaches such as *minimum-distance*, *correlation*, and *Bayes classifiers*, to more modern approaches implemented using *deep neural networks*. In particular, we will discuss in detail *deep convolutional neural networks*, which are ideally suited for image processing work.

So far, we have said nothing about the need for prior knowledge or about the interaction between the knowledge base and the processing modules in Fig. 1.23. *Knowledge* about a problem domain is coded into an image processing system in the form of a knowledge database. This knowledge may be as simple as detailing regions of an image where the information of interest is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base can also be quite complex, such as an interrelated list of all major possible defects in a materials inspection problem, or an image database containing high-resolution satellite images of a region in connection with change-detection applications. In addition to guiding the operation of each processing module, the knowledge base also controls the interaction between modules. This distinction is made in Fig. 1.23 by the use of double-headed arrows between the processing modules and the knowledge base, as opposed to single-headed arrows linking the processing modules.

Although we do not discuss image display explicitly at this point, it is important to keep in mind that viewing the results of image processing can take place at the output of any stage in Fig. 1.23. We also note that not all image processing applications require the complexity of interactions implied by Fig. 1.23. In fact, not even all those modules are needed in many cases. For example, image enhancement for human visual interpretation seldom requires use of any of the other stages in Fig. 1.23. In general, however, as the complexity of an image processing task increases, so does the number of processes required to solve the problem.

1.5 COMPONENTS OF AN IMAGE PROCESSING SYSTEM

As recently as the mid-1980s, numerous models of image processing systems being sold throughout the world were rather substantial peripheral devices that attached to equally substantial host computers. Late in the 1980s and early in the 1990s, the market shifted to image processing hardware in the form of single boards designed to be compatible with industry standard buses and to fit into engineering workstation cabinets and personal computers. In the late 1990s and early 2000s, a new class of add-on boards, called graphics processing units (GPUs) were introduced for work on 3-D applications, such as games and other 3-D graphics applications. It was not long before GPUs found their way into image processing applications involving large-scale matrix implementations, such as training deep convolutional networks. In addition to lowering costs, the market shift from substantial peripheral devices to add-on processing boards also served as a catalyst for a significant number of new companies specializing in the development of software written specifically for image processing.

The trend continues toward miniaturizing and blending of general-purpose small computers with specialized image processing hardware and software. Figure 1.24 shows the basic components comprising a typical general-purpose system used for digital image processing. The function of each component will be discussed in the following paragraphs, starting with image sensing.

Two subsystems are required to acquire digital images. The first is a physical *sensor* that responds to the energy radiated by the object we wish to image. The second, called a *digitizer*, is a device for converting the output of the physical sensing device into digital form. For instance, in a digital video camera, the sensors (CCD chips) produce an electrical output proportional to light intensity. The digitizer converts these outputs to digital data. These topics will be covered in Chapter 2.

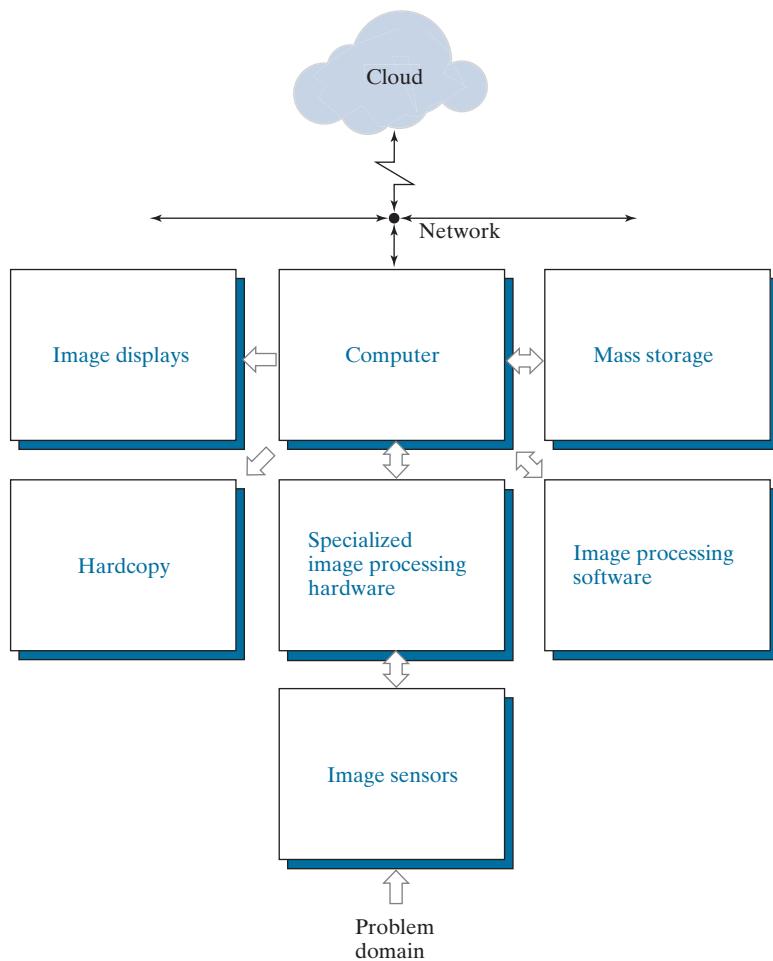
Specialized image processing hardware usually consists of the digitizer just mentioned, plus hardware that performs other primitive operations, such as an *arithmetic logic unit* (ALU), that performs arithmetic and logical operations in parallel on entire images. One example of how an ALU is used is in averaging images as quickly as they are digitized, for the purpose of noise reduction. This type of hardware sometimes is called a *front-end subsystem*, and its most distinguishing characteristic is speed. In other words, this unit performs functions that require fast data throughputs (e.g., digitizing and averaging video images at 30 frames/s) that the typical main computer cannot handle. One or more GPUs (see above) also are common in image processing systems that perform intensive matrix operations.

The *computer* in an image processing system is a general-purpose computer and can range from a PC to a supercomputer. In dedicated applications, sometimes custom computers are used to achieve a required level of performance, but our interest here is on general-purpose image processing systems. In these systems, almost any well-equipped PC-type machine is suitable for off-line image processing tasks.

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

FIGURE 1.24

Components of a general-purpose image processing system.



software packages allow the integration of those modules and general-purpose software commands from at least one computer language. Commercially available image processing software, such as the well-known MATLAB® Image Processing Toolbox, is also common in a well-equipped image processing system.

Mass storage is a must in image processing applications. An image of size 1024×1024 pixels, in which the intensity of each pixel is an 8-bit quantity, requires one megabyte of storage space if the image is not compressed. When dealing with image databases that contain thousands, or even millions, of images, providing adequate storage in an image processing system can be a challenge. Digital storage for image processing applications falls into three principal categories: (1) short-term storage for use during processing; (2) on-line storage for relatively fast recall; and (3) archival storage, characterized by infrequent access. Storage is measured in bytes (eight bits), Kbytes (10^3 bytes), Mbytes (10^6 bytes), Gbytes (10^9 bytes), and Tbytes (10^{12} bytes).

One method of providing short-term storage is computer memory. Another is by specialized boards, called *frame buffers*, that store one or more images and can be accessed rapidly, usually at video rates (e.g., at 30 complete images per second). The latter method allows virtually instantaneous image *zoom*, as well as *scroll* (vertical shifts) and *pan* (horizontal shifts). Frame buffers usually are housed in the specialized image processing hardware unit in Fig. 1.24. On-line storage generally takes the form of magnetic disks or optical-media storage. The key factor characterizing on-line storage is frequent access to the stored data. Finally, archival storage is characterized by massive storage requirements but infrequent need for access. Magnetic tapes and optical disks housed in “jukeboxes” are the usual media for archival applications.

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

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

Networking and *cloud* communication are almost default functions in any computer system in use today. Because of the large amount of data inherent in image processing applications, the key consideration in image transmission is *bandwidth*. In dedicated networks, this typically is not a problem, but communications with remote sites via the internet are not always as efficient. Fortunately, transmission bandwidth is improving quickly as a result of optical fiber and other broadband technologies. Image data compression continues to play a major role in the transmission of large amounts of image data.

Summary, References, and Further Reading

The main purpose of the material presented in this chapter is to provide a sense of perspective about the origins of digital image processing and, more important, about current and future areas of application of this technology. Although the coverage of these topics in this chapter was necessarily incomplete due to space limitations, it should have left you with a clear impression of the breadth and practical scope of digital image processing. As we proceed in the following chapters with the development of image processing theory and applications, numerous examples are provided to keep a clear focus on the utility and promise of these techniques. Upon concluding the study of the final chapter, a reader of this book will have arrived at a level of understanding that is the foundation for most of the work currently underway in this field.

In past editions, we have provided a long list of journals and books to give readers an idea of the breadth of the image processing literature, and where this literature is reported. The list has been updated, and it has become so extensive that it is more practical to include it in the book website: www.ImageProcessingPlace.com, in the section entitled *Publications*.

2



Those who wish to succeed must ask the right preliminary questions.

Aristotle

Preview

This chapter is an introduction to a number of basic concepts in digital image processing that are used throughout the book. Section 2.1 summarizes some important aspects of the human visual system, including image formation in the eye and its capabilities for brightness adaptation and discrimination. Section 2.2 discusses light, other components of the electromagnetic spectrum, and their imaging characteristics. Section 2.3 discusses imaging sensors and how they are used to generate digital images. Section 2.4 introduces the concepts of uniform image sampling and intensity quantization. Additional topics discussed in that section include digital image representation, the effects of varying the number of samples and intensity levels in an image, the concepts of spatial and intensity resolution, and the principles of image interpolation. Section 2.5 deals with a variety of basic relationships between pixels. Finally, Section 2.6 is an introduction to the principal mathematical tools we use throughout the book. A second objective of that section is to help you begin developing a “feel” for how these tools are used in a variety of basic image processing tasks.

Upon completion of this chapter, readers should:

- Have an understanding of some important functions and limitations of human vision.
- Be familiar with the electromagnetic energy spectrum, including basic properties of light.
- Know how digital images are generated and represented.
- Understand the basics of image sampling and quantization.
- Be familiar with spatial and intensity resolution and their effects on image appearance.
- Have an understanding of basic geometric relationships between image pixels.
- Be familiar with the principal mathematical tools used in digital image processing.
- Be able to apply a variety of introductory digital image processing techniques.

2.1 ELEMENTS OF VISUAL PERCEPTION

Although the field of digital image processing is built on a foundation of mathematics, human intuition and analysis often play a role in the choice of one technique versus another, and this choice often is made based on subjective, visual judgments. Thus, developing an understanding of basic characteristics of human visual perception as a first step in our journey through this book is appropriate. In particular, our interest is in the elementary mechanics of how images are formed and perceived by humans. We are interested in learning the physical limitations of human vision in terms of factors that also are used in our work with digital images. Factors such as how human and electronic imaging devices compare in terms of resolution and ability to adapt to changes in illumination are not only interesting, they are also important from a practical point of view.

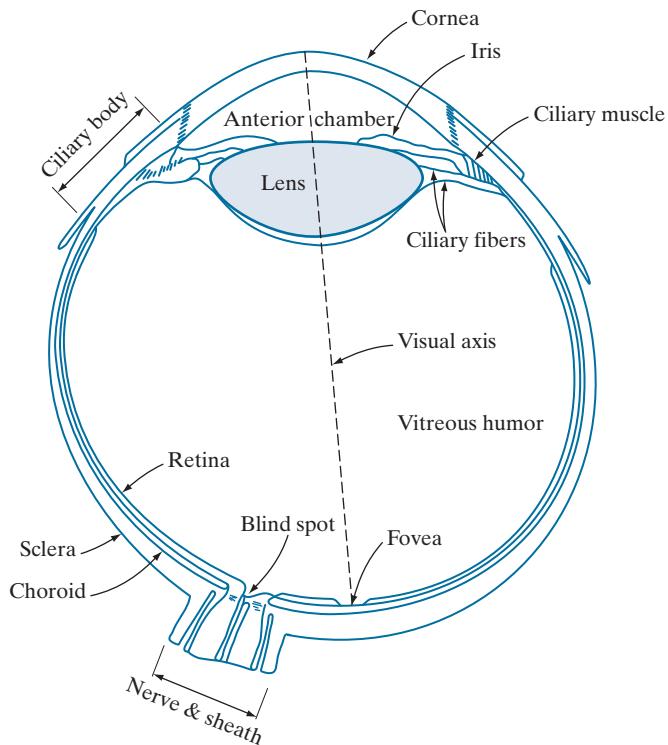
STRUCTURE OF THE HUMAN EYE

Figure 2.1 shows a simplified cross section of the human eye. The eye is nearly a sphere (with a diameter of about 20 mm) enclosed by three membranes: the *cornea* and *sclera* outer cover; the *choroid*; and the *retina*. The cornea is a tough, transparent tissue that covers the anterior surface of the eye. Continuous with the cornea, the sclera is an opaque membrane that encloses the remainder of the optic globe.

The choroid lies directly below the sclera. This membrane contains a network of blood vessels that serve as the major source of nutrition to the eye. Even superficial

FIGURE 2.1

Simplified diagram of a cross section of the human eye.



injury to the choroid can lead to severe eye damage as a result of inflammation that restricts blood flow. The choroid coat is heavily pigmented, which helps reduce the amount of extraneous light entering the eye and the backscatter within the optic globe. At its anterior extreme, the choroid is divided into the *ciliary body* and the *iris*. The latter contracts or expands to control the amount of light that enters the eye. The central opening of the iris (the *pupil*) varies in diameter from approximately 2 to 8 mm. The front of the iris contains the visible pigment of the eye, whereas the back contains a black pigment.

The *lens* consists of concentric layers of fibrous cells and is suspended by fibers that attach to the ciliary body. It is composed of 60% to 70% water, about 6% fat, and more protein than any other tissue in the eye. The lens is colored by a slightly yellow pigmentation that increases with age. In extreme cases, excessive clouding of the lens, referred to as *cataracts*, can lead to poor color discrimination and loss of clear vision. The lens absorbs approximately 8% of the visible light spectrum, with higher absorption at shorter wavelengths. Both infrared and ultraviolet light are absorbed by proteins within the lens and, in excessive amounts, can damage the eye.

The innermost membrane of the eye is the *retina*, which lines the inside of the wall's entire posterior portion. When the eye is focused, light from an object is imaged on the retina. Pattern vision is afforded by discrete light receptors distributed over the surface of the retina. There are two types of receptors: *cones* and *rods*. There are between 6 and 7 million cones in each eye. They are located primarily in the central portion of the retina, called the *fovea*, and are highly sensitive to color. Humans can resolve fine details because each cone is connected to its own nerve end. Muscles rotate the eye until the image of a region of interest falls on the fovea. Cone vision is called *photopic* or *bright-light* vision.

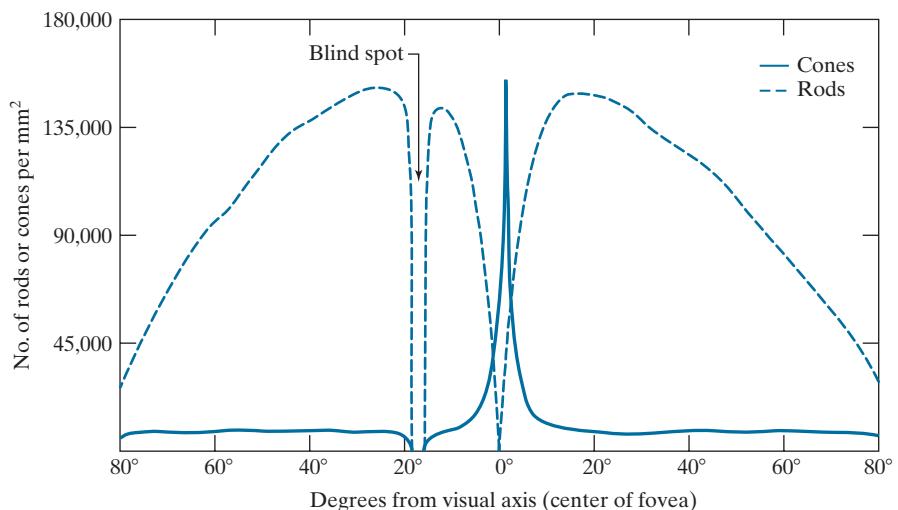
The number of rods is much larger: Some 75 to 150 million are distributed over the retina. The larger area of distribution, and the fact that several rods are connected to a single nerve ending, reduces the amount of detail discernible by these receptors. Rods capture an overall image of the field of view. They are not involved in color vision, and are sensitive to low levels of illumination. For example, objects that appear brightly colored in daylight appear as colorless forms in moonlight because only the rods are stimulated. This phenomenon is known as *scotopic* or *dim-light* vision.

Figure 2.2 shows the density of rods and cones for a cross section of the right eye, passing through the region where the optic nerve emerges from the eye. The absence of receptors in this area causes the so-called *blind spot* (see Fig. 2.1). Except for this region, the distribution of receptors is radially symmetric about the fovea. Receptor density is measured in degrees from the visual axis. Note in Fig. 2.2 that cones are most dense in the center area of the fovea, and that rods increase in density from the center out to approximately 20° off axis. Then, their density decreases out to the periphery of the retina.

The fovea itself is a circular indentation in the retina of about 1.5 mm in diameter, so it has an area of approximately 1.77 mm². As Fig. 2.2 shows, the density of cones in that area of the retina is on the order of 150,000 elements per mm². Based on these figures, the number of cones in the fovea, which is the region of highest acuity

FIGURE 2.2

Distribution of rods and cones in the retina.



in the eye, is about 265,000 elements. Modern electronic imaging chips exceed this number by a large factor. While the ability of humans to integrate intelligence and experience with vision makes purely quantitative comparisons somewhat superficial, keep in mind for future discussions that electronic imaging sensors can easily exceed the capability of the eye in resolving image detail.

IMAGE FORMATION IN THE EYE

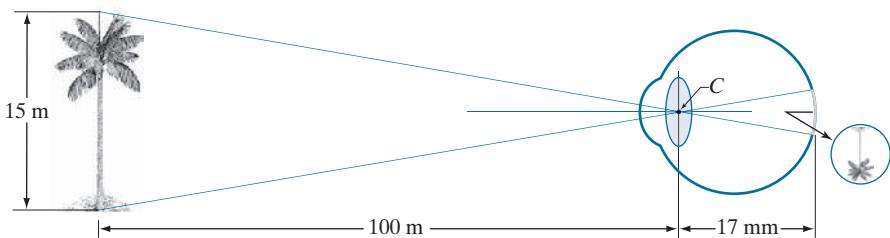
In an ordinary photographic camera, the lens has a fixed focal length. Focusing at various distances is achieved by varying the distance between the lens and the imaging plane, where the film (or imaging chip in the case of a digital camera) is located. In the human eye, the converse is true; the distance between the center of the lens and the imaging sensor (the retina) is fixed, and the focal length needed to achieve proper focus is obtained by varying the shape of the lens. The fibers in the ciliary body accomplish this by flattening or thickening the lens for distant or near objects, respectively. The distance between the center of the lens and the retina along the visual axis is approximately 17 mm. The range of focal lengths is approximately 14 mm to 17 mm, the latter taking place when the eye is relaxed and focused at distances greater than about 3 m. The geometry in Fig. 2.3 illustrates how to obtain the dimensions of an image formed on the retina. For example, suppose that a person is looking at a tree 15 m high at a distance of 100 m. Letting h denote the height of that object in the retinal image, the geometry of Fig. 2.3 yields $15/100 = h/17$ or $h = 2.5$ mm. As indicated earlier in this section, the retinal image is focused primarily on the region of the fovea. Perception then takes place by the relative excitation of light receptors, which transform radiant energy into electrical impulses that ultimately are decoded by the brain.

BRIGHTNESS ADAPTATION AND DISCRIMINATION

Because digital images are displayed as sets of discrete intensities, the eye's ability to discriminate between different intensity levels is an important consideration

FIGURE 2.3

Graphical representation of the eye looking at a palm tree. Point *C* is the focal center of the lens.

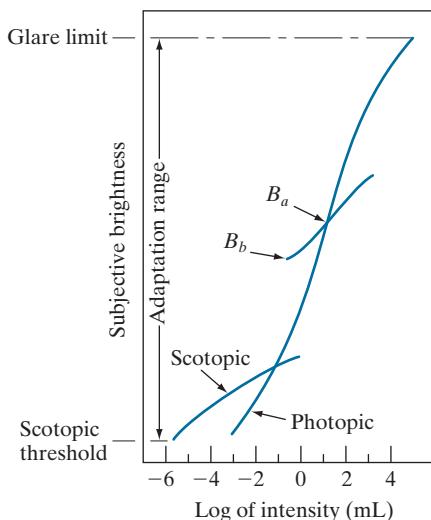


in presenting image processing results. The range of light intensity levels to which the human visual system can adapt is enormous—on the order of 10^{10} —from the scotopic threshold to the glare limit. Experimental evidence indicates that *subjective brightness* (intensity as perceived by the human visual system) is a logarithmic function of the light intensity incident on the eye. Figure 2.4, a plot of light intensity versus subjective brightness, illustrates this characteristic. The long solid curve represents the range of intensities to which the visual system can adapt. In photopic vision alone, the range is about 10^6 . The transition from scotopic to photopic vision is gradual over the approximate range from 0.001 to 0.1 millilambert (-3 to -1 mL in the log scale), as the double branches of the adaptation curve in this range show.

The key point in interpreting the impressive dynamic range depicted in Fig. 2.4 is that the visual system cannot operate over such a range *simultaneously*. Rather, it accomplishes this large variation by changing its overall sensitivity, a phenomenon known as *brightness adaptation*. The total range of distinct intensity levels the eye can discriminate simultaneously is rather small when compared with the total adaptation range. For a given set of conditions, the current sensitivity level of the visual system is called the *brightness adaptation level*, which may correspond, for example,

FIGURE 2.4

Range of subjective brightness sensations showing a particular adaptation level, B_a .



to brightness B_a in Fig. 2.4. The short intersecting curve represents the range of subjective brightness that the eye can perceive when adapted to *this* level. This range is rather restricted, having a level B_b at, and below which, all stimuli are perceived as indistinguishable blacks. The upper portion of the curve is not actually restricted but, if extended too far, loses its meaning because much higher intensities would simply raise the adaptation level higher than B_a .

The ability of the eye to discriminate between *changes* in light intensity at any specific adaptation level is of considerable interest. A classic experiment used to determine the capability of the human visual system for brightness discrimination consists of having a subject look at a flat, uniformly illuminated area large enough to occupy the entire field of view. This area typically is a diffuser, such as opaque glass, illuminated from behind by a light source, I , with variable intensity. To this field is added an increment of illumination, ΔI , in the form of a short-duration flash that appears as a circle in the center of the uniformly illuminated field, as Fig. 2.5 shows.

If ΔI is not bright enough, the subject says “no,” indicating no perceivable change. As ΔI gets stronger, the subject may give a positive response of “yes,” indicating a perceived change. Finally, when ΔI is strong enough, the subject will give a response of “yes” all the time. The quantity $\Delta I_c/I$, where ΔI_c is the increment of illumination discriminable 50% of the time with background illumination I , is called the *Weber ratio*. A small value of $\Delta I_c/I$ means that a small percentage change in intensity is discriminable. This represents “good” brightness discrimination. Conversely, a large value of $\Delta I_c/I$ means that a large percentage change in intensity is required for the eye to detect the change. This represents “poor” brightness discrimination.

A plot of $\Delta I_c/I$ as a function of $\log I$ has the characteristic shape shown in Fig. 2.6. This curve shows that brightness discrimination is poor (the Weber ratio is large) at low levels of illumination, and it improves significantly (the Weber ratio decreases) as background illumination increases. The two branches in the curve reflect the fact that at low levels of illumination vision is carried out by the rods, whereas, at high levels, vision is a function of cones.

If the background illumination is held constant and the intensity of the other source, instead of flashing, is now allowed to vary incrementally from never being perceived to always being perceived, the typical observer can discern a total of one to two dozen different intensity changes. Roughly, this result is related to the number of different intensities a person can see at any one *point or small area* in a monochrome image. This does not mean that an image can be represented by such a small number of intensity values because, as the eye roams about the image, the average

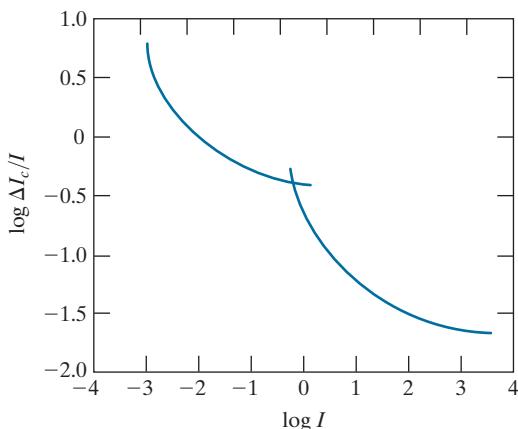
FIGURE 2.5

Basic experimental setup used to characterize brightness discrimination.



FIGURE 2.6

A typical plot of the Weber ratio as a function of intensity.



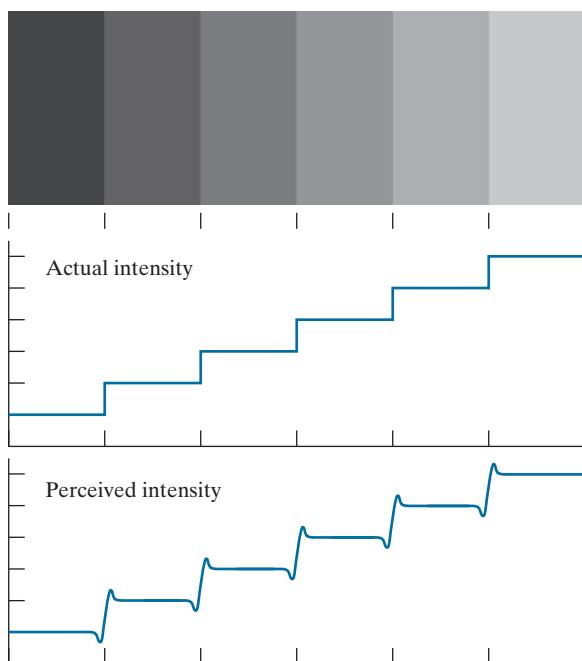
background changes, thus allowing a *different* set of incremental changes to be detected at each new adaptation level. The net result is that the eye is capable of a broader range of *overall* intensity discrimination. In fact, as we will show in Section 2.4, the eye is capable of detecting objectionable effects in monochrome images whose overall intensity is represented by fewer than approximately two dozen levels.

Two phenomena demonstrate that perceived brightness is not a simple function of intensity. The first is based on the fact that the visual system tends to undershoot or overshoot around the boundary of regions of different intensities. Figure 2.7(a) shows a striking example of this phenomenon. Although the intensity of the stripes

a
b
c

FIGURE 2.7

Illustration of the Mach band effect. Perceived intensity is not a simple function of actual intensity.



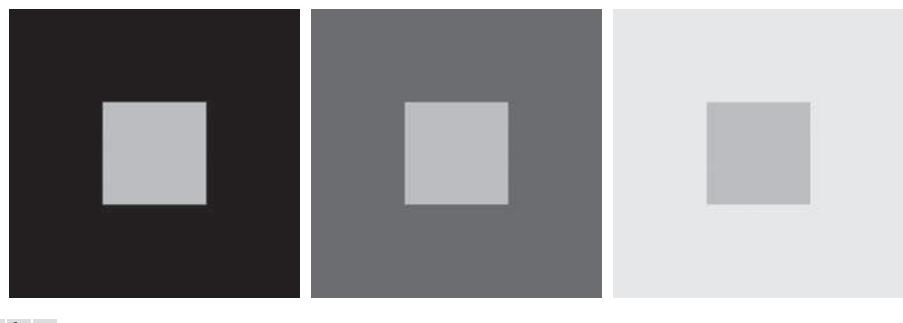


FIGURE 2.8 Examples of simultaneous contrast. All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

is constant [see Fig. 2.7(b)], we actually perceive a brightness pattern that is strongly scalloped near the boundaries, as Fig. 2.7(c) shows. These perceived scalloped bands are called *Mach bands* after Ernst Mach, who first described the phenomenon in 1865.

The second phenomenon, called *simultaneous contrast*, is that a region's perceived brightness does not depend only on its intensity, as Fig. 2.8 demonstrates. All the center squares have exactly the same intensity, but each appears to the eye to become darker as the background gets lighter. A more familiar example is a piece of paper that looks white when lying on a desk, but can appear totally black when used to shield the eyes while looking directly at a bright sky.

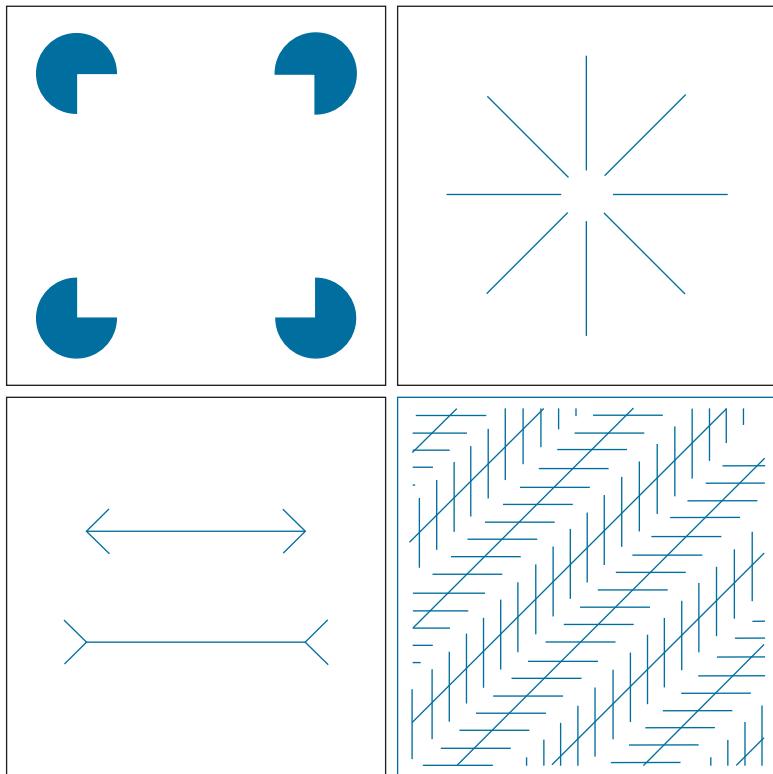
Other examples of human perception phenomena are *optical illusions*, in which the eye fills in nonexisting details or wrongly perceives geometrical properties of objects. Figure 2.9 shows some examples. In Fig. 2.9(a), the outline of a square is seen clearly, despite the fact that no lines defining such a figure are part of the image. The same effect, this time with a circle, can be seen in Fig. 2.9(b); note how just a few lines are sufficient to give the illusion of a complete circle. The two horizontal line segments in Fig. 2.9(c) are of the same length, but one appears shorter than the other. Finally, all long lines in Fig. 2.9(d) are equidistant and parallel. Yet, the crosshatching creates the illusion that those lines are far from being parallel.

2.2 LIGHT AND THE ELECTROMAGNETIC SPECTRUM

The electromagnetic spectrum was introduced in Section 1.3. We now consider this topic in more detail. In 1666, Sir Isaac Newton discovered that when a beam of sunlight passes through a glass prism, the emerging beam of light is not white but consists instead of a continuous spectrum of colors ranging from violet at one end to red at the other. As Fig. 2.10 shows, the range of colors we perceive in visible light is a small portion of the electromagnetic spectrum. On one end of the spectrum are radio waves with wavelengths billions of times longer than those of visible light. On the other end of the spectrum are gamma rays with wavelengths millions of times smaller than those of visible light. We showed examples in Section 1.3 of images in most of the bands in the EM spectrum.

a	b
c	d

FIGURE 2.9 Some well-known optical illusions.



The electromagnetic spectrum can be expressed in terms of wavelength, frequency, or energy. Wavelength (λ) and frequency (ν) are related by the expression

$$\lambda = \frac{c}{\nu} \quad (2-1)$$

where c is the speed of light (2.998×10^8 m/s). Figure 2.11 shows a schematic representation of one wavelength.

The energy of the various components of the electromagnetic spectrum is given by the expression

$$E = h\nu \quad (2-2)$$

where h is Planck's constant. The units of wavelength are meters, with the terms *microns* (denoted μm and equal to 10^{-6} m) and *nanometers* (denoted nm and equal to 10^{-9} m) being used just as frequently. Frequency is measured in *Hertz* (Hz), with one Hz being equal to one cycle of a sinusoidal wave per second. A commonly used unit of energy is the *electron-volt*.

Electromagnetic waves can be visualized as propagating sinusoidal waves with wavelength λ (Fig. 2.11), or they can be thought of as a stream of massless particles,

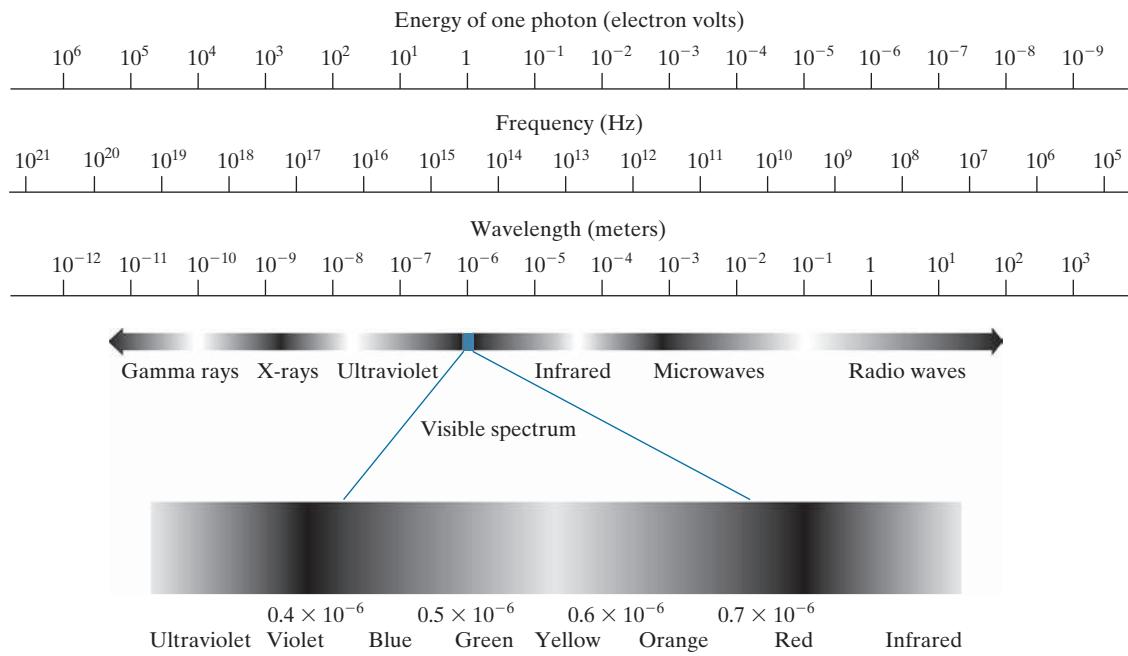
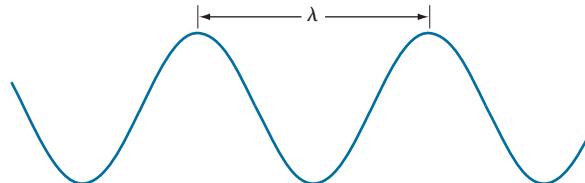


FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanations, but note that it encompasses a very narrow range of the total EM spectrum.

each traveling in a wavelike pattern and moving at the speed of light. Each massless particle contains a certain amount (or bundle) of energy, called a *photon*. We see from Eq. (2-2) that energy is proportional to frequency, so the higher-frequency (shorter wavelength) electromagnetic phenomena carry more energy per photon. Thus, radio waves have photons with low energies, microwaves have more energy than radio waves, infrared still more, then visible, ultraviolet, X-rays, and finally gamma rays, the most energetic of all. High-energy electromagnetic radiation, especially in the X-ray and gamma ray bands, is particularly harmful to living organisms.

Light is a type of electromagnetic radiation that can be sensed by the eye. The visible (color) spectrum is shown expanded in Fig. 2.10 for the purpose of discussion (we will discuss color in detail in Chapter 6). The visible band of the electromagnetic spectrum spans the range from approximately $0.43 \mu\text{m}$ (violet) to about $0.79 \mu\text{m}$ (red). For convenience, the color spectrum is divided into six broad regions: violet, blue, green, yellow, orange, and red. No color (or other component of the

FIGURE 2.11
Graphical representation of one wavelength.



electromagnetic spectrum) ends abruptly; rather, each range blends smoothly into the next, as Fig. 2.10 shows.

The colors perceived in an object are determined by the nature of the light *reflected* by the object. A body that reflects light relatively balanced in all visible wavelengths appears white to the observer. However, a body that favors reflectance in a limited range of the visible spectrum exhibits some shades of color. For example, green objects reflect light with wavelengths primarily in the 500 to 570 nm range, while absorbing most of the energy at other wavelengths.

Light that is void of color is called *monochromatic* (or *achromatic*) light. The only attribute of monochromatic light is its intensity. Because the intensity of monochromatic light is perceived to vary from black to grays and finally to white, the term *gray level* is used commonly to denote monochromatic intensity (we use the terms *intensity* and *gray level* interchangeably in subsequent discussions). The range of values of monochromatic light from black to white is usually called the *gray scale*, and monochromatic images are frequently referred to as *grayscale images*.

Chromatic (color) light spans the electromagnetic energy spectrum from approximately 0.43 to 0.79 μm , as noted previously. In addition to frequency, three other quantities are used to describe a chromatic light source: radiance, luminance, and brightness. *Radiance* is the total amount of energy that flows from the light source, and it is usually measured in watts (W). *Luminance*, measured in lumens (lm), gives a measure of the amount of energy an observer *perceives* from a light source. For example, light emitted from a source operating in the far infrared region of the spectrum could have significant energy (radiance), but an observer would hardly perceive it; its luminance would be almost zero. Finally, as discussed in Section 2.1, *brightness* is a subjective descriptor of light perception that is practically impossible to measure. It embodies the achromatic notion of intensity and is one of the key factors in describing color sensation.

In principle, if a sensor can be developed that is capable of detecting energy radiated in a band of the electromagnetic spectrum, we can image events of interest in that band. Note, however, that the wavelength of an electromagnetic wave required to “see” an object must be of the same size as, or smaller than, the object. For example, a water molecule has a diameter on the order of 10^{-10} m. Thus, to study these molecules, we would need a source capable of emitting energy in the far (high-energy) ultraviolet band or soft (low-energy) X-ray bands.

Although imaging is based predominantly on energy from electromagnetic wave radiation, this is not the only method for generating images. For example, we saw in Section 1.3 that sound reflected from objects can be used to form ultrasonic images. Other sources of digital images are electron beams for electron microscopy, and software for generating synthetic images used in graphics and visualization.

2.3 IMAGE SENSING AND ACQUISITION

Most of the images in which we are interested are generated by the combination of an “illumination” source and the reflection or absorption of energy from that source by the elements of the “scene” being imaged. We enclose *illumination* and *scene* in quotes to emphasize the fact that they are considerably more general than the

familiar situation in which a visible light source illuminates a familiar 3-D scene. For example, the illumination may originate from a source of electromagnetic energy, such as a radar, infrared, or X-ray system. But, as noted earlier, it could originate from less traditional sources, such as ultrasound or even a computer-generated illumination pattern. Similarly, the scene elements could be familiar objects, but they can just as easily be molecules, buried rock formations, or a human brain. Depending on the nature of the source, illumination energy is reflected from, or transmitted through, objects. An example in the first category is light reflected from a planar surface. An example in the second category is when X-rays pass through a patient's body for the purpose of generating a diagnostic X-ray image. In some applications, the reflected or transmitted energy is focused onto a photo converter (e.g., a phosphor screen) that converts the energy into visible light. Electron microscopy and some applications of gamma imaging use this approach.

Figure 2.12 shows the three principal sensor arrangements used to transform incident energy into digital images. The idea is simple: Incoming energy is transformed into a voltage by a combination of the input electrical power and sensor material that is responsive to the type of energy being detected. The output voltage waveform is the response of the sensor, and a digital quantity is obtained by digitizing that response. In this section, we look at the principal modalities for image sensing and generation. We will discuss image digitizing in Section 2.4.

IMAGE ACQUISITION USING A SINGLE SENSING ELEMENT

Figure 2.12(a) shows the components of a single sensing element. A familiar sensor of this type is the photodiode, which is constructed of silicon materials and whose output is a voltage proportional to light intensity. Using a filter in front of a sensor improves its selectivity. For example, an optical green-transmission filter favors light in the green band of the color spectrum. As a consequence, the sensor output would be stronger for green light than for other visible light components.

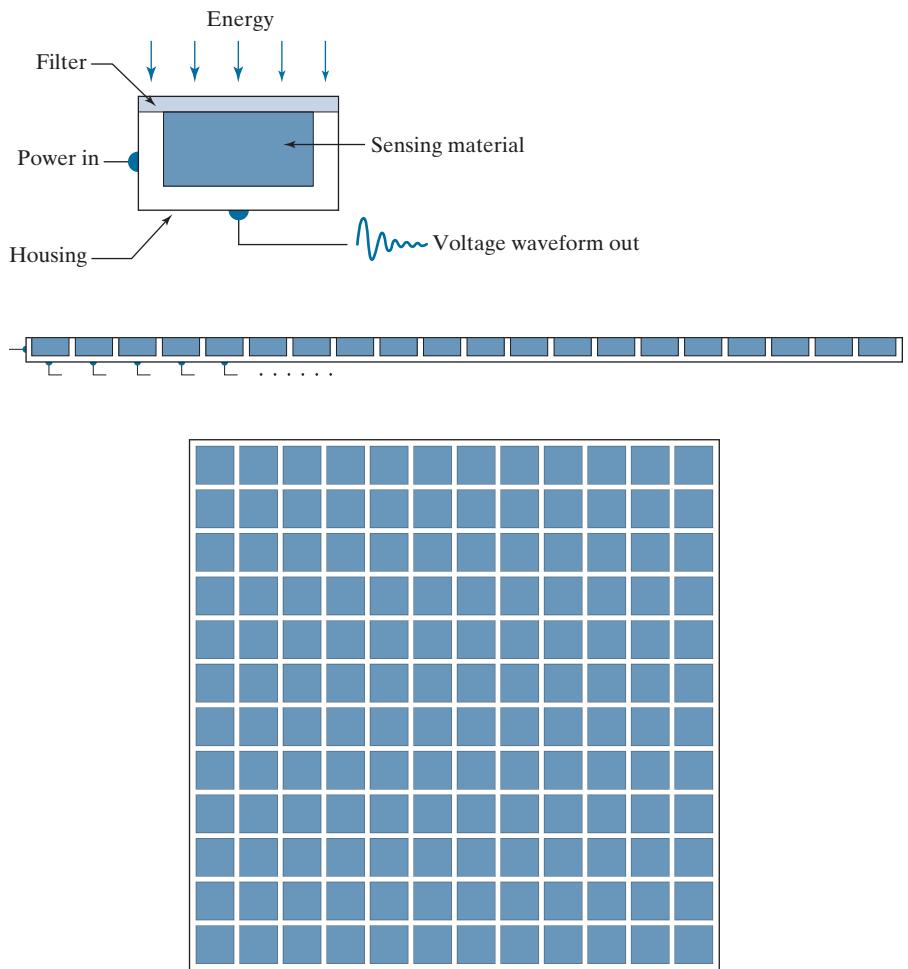
In order to generate a 2-D image using a single sensing element, there has to be relative displacements in both the x - and y -directions between the sensor and the area to be imaged. Figure 2.13 shows an arrangement used in high-precision scanning, where a film negative is mounted onto a drum whose mechanical rotation provides displacement in one dimension. The sensor is mounted on a lead screw that provides motion in the perpendicular direction. A light source is contained inside the drum. As the light passes through the film, its intensity is modified by the film density before it is captured by the sensor. This "modulation" of the light intensity causes corresponding variations in the sensor voltage, which are ultimately converted to image intensity levels by digitization.

This method is an inexpensive way to obtain high-resolution images because mechanical motion can be controlled with high precision. The main disadvantages of this method are that it is slow and not readily portable. Other similar mechanical arrangements use a flat imaging bed, with the sensor moving in two linear directions. These types of mechanical digitizers sometimes are referred to as *transmission microdensitometers*. Systems in which light is reflected from the medium, instead of passing through it, are called *reflection microdensitometers*. Another example of imaging with a single sensing element places a laser source coincident with the

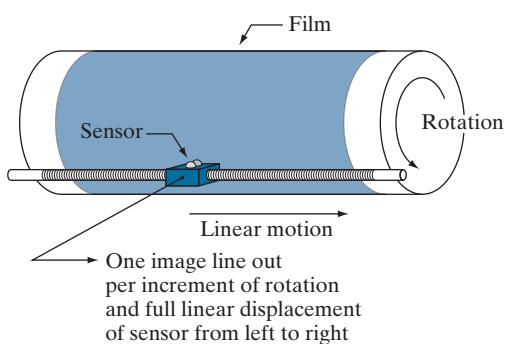
a
b
c

FIGURE 2.12

- (a) Single sensing element.
 (b) Line sensor.
 (c) Array sensor.

**FIGURE 2.13**

- Combining a single sensing element with mechanical motion to generate a 2-D image.



sensor. Moving mirrors are used to control the outgoing beam in a scanning pattern and to direct the reflected laser signal onto the sensor.

IMAGE ACQUISITION USING SENSOR STRIPS

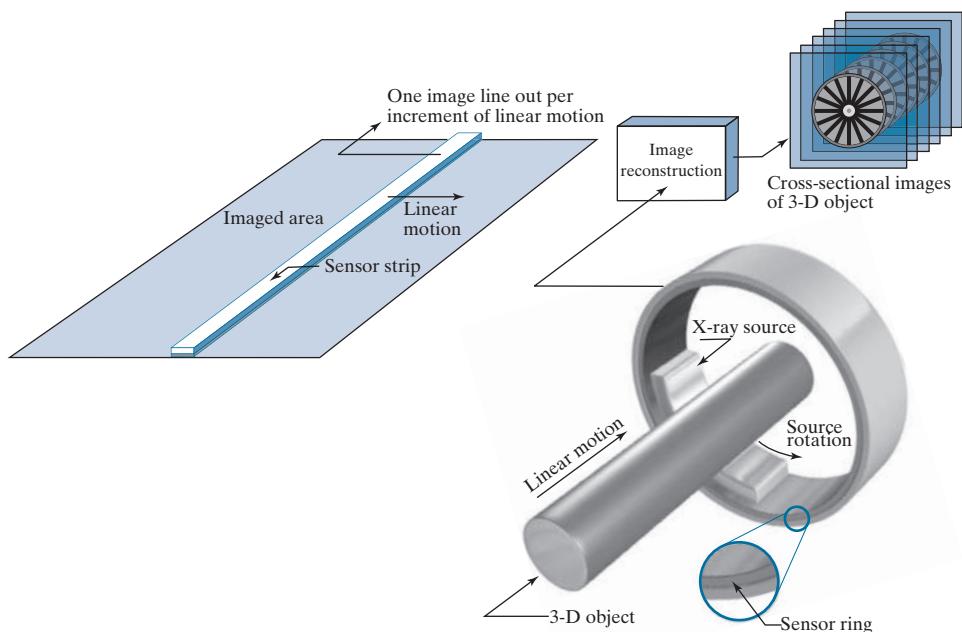
A geometry used more frequently than single sensors is an in-line sensor strip, as in Fig. 2.12(b). The strip provides imaging elements in one direction. Motion perpendicular to the strip provides imaging in the other direction, as shown in Fig. 2.14(a). This arrangement is used in most flat bed scanners. Sensing devices with 4000 or more in-line sensors are possible. In-line sensors are used routinely in airborne imaging applications, in which the imaging system is mounted on an aircraft that flies at a constant altitude and speed over the geographical area to be imaged. One-dimensional imaging sensor strips that respond to various bands of the electromagnetic spectrum are mounted perpendicular to the direction of flight. An imaging strip gives one line of an image at a time, and the motion of the strip relative to the scene completes the other dimension of a 2-D image. Lenses or other focusing schemes are used to project the area to be scanned onto the sensors.

Sensor strips in a ring configuration are used in medical and industrial imaging to obtain cross-sectional (“slice”) images of 3-D objects, as Fig. 2.14(b) shows. A rotating X-ray source provides illumination, and X-ray sensitive sensors opposite the source collect the energy that passes through the object. This is the basis for medical and industrial computerized axial tomography (CAT) imaging, as indicated in Sections 1.2 and 1.3. The output of the sensors is processed by reconstruction algorithms whose objective is to transform the sensed data into meaningful cross-sectional images (see Section 5.11). In other words, images are not obtained directly

a b

FIGURE 2.14

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



from the sensors by motion alone; they also require extensive computer processing. A 3-D digital volume consisting of stacked images is generated as the object is moved in a direction perpendicular to the sensor ring. Other modalities of imaging based on the CAT principle include magnetic resonance imaging (MRI) and positron emission tomography (PET). The illumination sources, sensors, and types of images are different, but conceptually their applications are very similar to the basic imaging approach shown in Fig. 2.14(b).

IMAGE ACQUISITION USING SENSOR ARRAYS

Figure 2.12(c) shows individual sensing elements arranged in the form of a 2-D array. Electromagnetic and ultrasonic sensing devices frequently are arranged in this manner. This is also the predominant arrangement found in digital cameras. A typical sensor for these cameras is a CCD (charge-coupled device) array, which can be manufactured with a broad range of sensing properties and can be packaged in rugged arrays of 4000×4000 elements or more. CCD sensors are used widely in digital cameras and other light-sensing instruments. The response of each sensor is proportional to the integral of the light energy projected onto the surface of the sensor, a property that is used in astronomical and other applications requiring low noise images. Noise reduction is achieved by letting the sensor integrate the input light signal over minutes or even hours. Because the sensor array in Fig. 2.12(c) is two-dimensional, its key advantage is that a complete image can be obtained by focusing the energy pattern onto the surface of the array. Motion obviously is not necessary, as is the case with the sensor arrangements discussed in the preceding two sections.

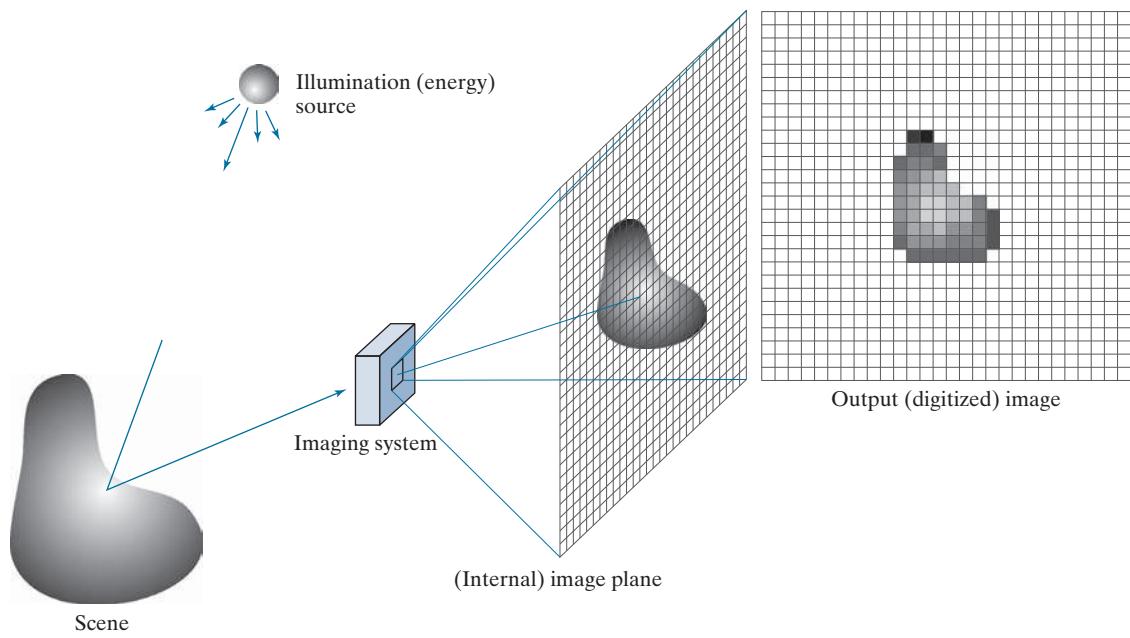
Figure 2.15 shows the principal manner in which array sensors are used. This figure shows the energy from an illumination source being reflected from a scene (as mentioned at the beginning of this section, the energy also could be transmitted through the scene). The first function performed by the imaging system in Fig. 2.15(c) is to collect the incoming energy and focus it onto an image plane. If the illumination is light, the front end of the imaging system is an optical lens that projects the viewed scene onto the focal plane of the lens, as Fig. 2.15(d) shows. The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor. Digital and analog circuitry sweep these outputs and convert them to an analog signal, which is then digitized by another section of the imaging system. The output is a digital image, as shown diagrammatically in Fig. 2.15(e). Converting images into digital form is the topic of Section 2.4.

In some cases, the source is imaged directly, as in obtaining images of the sun.

A SIMPLE IMAGE FORMATION MODEL

As introduced in Section 1.1, we denote images by two-dimensional functions of the form $f(x, y)$. The value of f at spatial coordinates (x, y) is a scalar quantity whose physical meaning is determined by the source of the image, and whose values are proportional to energy radiated by a physical source (e.g., electromagnetic waves). As a consequence, $f(x, y)$ must be nonnegative[†] and finite; that is,

[†] Image intensities can become negative during processing, or as a result of interpretation. For example, in radar images, objects moving toward the radar often are interpreted as having negative velocities while objects moving away are interpreted as having positive velocities. Thus, a velocity image might be coded as having both positive and negative values. When storing and displaying images, we normally scale the intensities so that the smallest negative value becomes 0 (see Section 2.6 regarding intensity scaling).



a c d e
b

FIGURE 2.15 An example of digital image acquisition. (a) Illumination (energy) source. (b) A scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

$$0 \leq f(x,y) < \infty \quad (2-3)$$

Function $f(x,y)$ is characterized by two components: (1) the amount of source illumination incident on the scene being viewed, and (2) the amount of illumination reflected by the objects in the scene. Appropriately, these are called the *illumination* and *reflectance* components, and are denoted by $i(x,y)$ and $r(x,y)$, respectively. The two functions combine as a product to form $f(x,y)$:

$$f(x,y) = i(x,y)r(x,y) \quad (2-4)$$

where

$$0 \leq i(x,y) < \infty \quad (2-5)$$

and

$$0 \leq r(x,y) \leq 1 \quad (2-6)$$

Thus, reflectance is bounded by 0 (total absorption) and 1 (total reflectance). The nature of $i(x,y)$ is determined by the illumination source, and $r(x,y)$ is determined by the characteristics of the imaged objects. These expressions are applicable also to images formed via transmission of the illumination through a medium, such as a

chest X-ray. In this case, we would deal with a *transmissivity* instead of a *reflectivity* function, but the limits would be the same as in Eq. (2-6), and the image function formed would be modeled as the product in Eq. (2-4).

EXAMPLE 2.1: Some typical values of illumination and reflectance.

The following numerical quantities illustrate some typical values of illumination and reflectance for visible light. On a clear day, the sun may produce in excess of $90,000 \text{ lm/m}^2$ of illumination on the surface of the earth. This value decreases to less than $10,000 \text{ lm/m}^2$ on a cloudy day. On a clear evening, a full moon yields about 0.1 lm/m^2 of illumination. The typical illumination level in a commercial office is about $1,000 \text{ lm/m}^2$. Similarly, the following are typical values of $r(x, y)$: 0.01 for black velvet, 0.65 for stainless steel, 0.80 for flat-white wall paint, 0.90 for silver-plated metal, and 0.93 for snow.

Let the intensity (gray level) of a monochrome image at any coordinates (x, y) be denoted by

$$\ell = f(x, y) \quad (2-7)$$

From Eqs. (2-4) through (2-6) it is evident that ℓ lies in the range

$$L_{\min} \leq \ell \leq L_{\max} \quad (2-8)$$

In theory, the requirement on L_{\min} is that it be nonnegative, and on L_{\max} that it be finite. In practice, $L_{\min} = i_{\min} r_{\min}$ and $L_{\max} = i_{\max} r_{\max}$. From Example 2.1, using average office illumination and reflectance values as guidelines, we may expect $L_{\min} \approx 10$ and $L_{\max} \approx 1000$ to be typical indoor values in the absence of additional illumination. The units of these quantities are lum/m^2 . However, actual units seldom are of interest, except in cases where photometric measurements are being performed.

The interval $[L_{\min}, L_{\max}]$ is called the *intensity (or gray) scale*. Common practice is to shift this interval numerically to the interval $[0, 1]$, or $[0, C]$, where $\ell = 0$ is considered black and $\ell = 1$ (or C) is considered white on the scale. All intermediate values are shades of gray varying from black to white.

2.4 IMAGE SAMPLING AND QUANTIZATION

The discussion of sampling in this section is of an intuitive nature. We will discuss this topic in depth in Chapter 4.

As discussed in the previous section, there are numerous ways to acquire images, but our objective in all is the same: to generate digital images from sensed data. The output of most sensors is a continuous voltage waveform whose amplitude and spatial behavior are related to the physical phenomenon being sensed. To create a digital image, we need to convert the continuous sensed data into a digital format. This requires two processes: *sampling* and *quantization*.

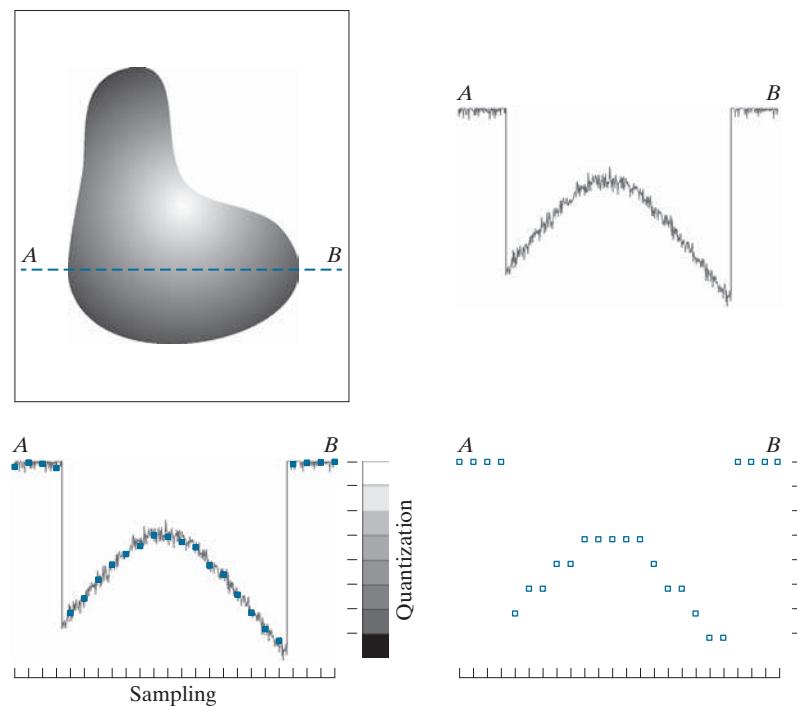
BASIC CONCEPTS IN SAMPLING AND QUANTIZATION

Figure 2.16(a) shows a continuous image f that we want to convert to digital form. An image may be continuous with respect to the x - and y -coordinates, and also in

a
b
c
d

FIGURE 2.16

- (a) Continuous image.
- (b) A scan line showing intensity variations along line AB in the continuous image.
- (c) Sampling and quantization.
- (d) Digital scan line. (The black border in (a) is included for clarity. It is not part of the image).



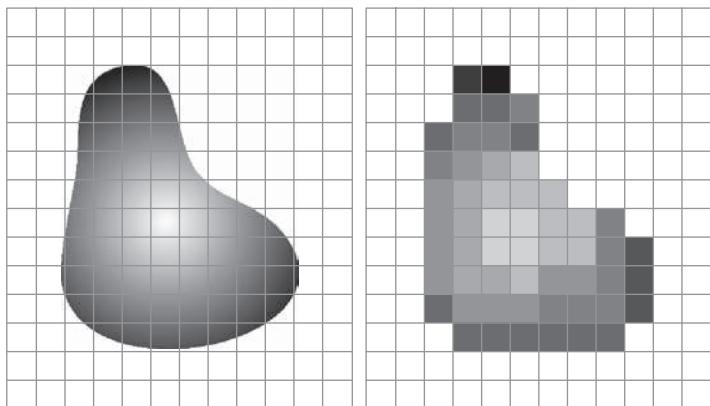
amplitude. To digitize it, we have to sample the function in both coordinates and also in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.

The one-dimensional function in Fig. 2.16(b) is a plot of amplitude (intensity level) values of the continuous image along the line segment AB in Fig. 2.16(a). The random variations are due to image noise. To sample this function, we take equally spaced samples along line AB, as shown in Fig. 2.16(c). The samples are shown as small dark squares superimposed on the function, and their (discrete) spatial locations are indicated by corresponding tick marks in the bottom of the figure. The set of dark squares constitute the *sampled* function. However, the *values* of the samples still span (vertically) a continuous range of intensity values. In order to form a digital function, the intensity values also must be converted (*quantized*) into *discrete* quantities. The vertical gray bar in Fig. 2.16(c) depicts the intensity scale divided into eight discrete intervals, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight intensity intervals. The continuous intensity levels are quantized by assigning one of the eight values to each sample, depending on the vertical proximity of a sample to a vertical tick mark. The digital samples resulting from both sampling and quantization are shown as white squares in Fig. 2.16(d). Starting at the top of the continuous image and carrying out this procedure downward, line by line, produces a two-dimensional digital image. It is implied in Fig. 2.16 that, in addition to the number of discrete levels used, the accuracy achieved in quantization is highly dependent on the noise content of the sampled signal.

a b

FIGURE 2.17

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



In practice, the method of sampling is determined by the sensor arrangement used to generate the image. When an image is generated by a single sensing element combined with mechanical motion, as in Fig. 2.13, the output of the sensor is quantized in the manner described above. However, spatial sampling is accomplished by selecting the number of individual mechanical increments at which we activate the sensor to collect data. Mechanical motion can be very exact so, in principle, there is almost no limit on how fine we can sample an image using this approach. In practice, limits on sampling accuracy are determined by other factors, such as the quality of the optical components used in the system.

When a sensing strip is used for image acquisition, the number of sensors in the strip establishes the samples in the resulting image in one direction, and mechanical motion establishes the number of samples in the other. Quantization of the sensor outputs completes the process of generating a digital image.

When a sensing array is used for image acquisition, no motion is required. The number of sensors in the array establishes the limits of sampling in both directions. Quantization of the sensor outputs is as explained above. Figure 2.17 illustrates this concept. Figure 2.17(a) shows a continuous image projected onto the plane of a 2-D sensor. Figure 2.17(b) shows the image after sampling and quantization. The quality of a digital image is determined to a large degree by the number of samples and discrete intensity levels used in sampling and quantization. However, as we will show later in this section, image content also plays a role in the choice of these parameters.

REPRESENTING DIGITAL IMAGES

Let $f(s, t)$ represent a *continuous* image function of two continuous variables, s and t . We convert this function into a *digital image* by sampling and quantization, as explained in the previous section. Suppose that we sample the continuous image into a digital image, $f(x, y)$, containing M rows and N columns, where (x, y) are discrete coordinates. For notational clarity and convenience, we use integer values for these discrete coordinates: $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$. Thus, for example, the value of the digital image at the origin is $f(0, 0)$, and its value at the next coordinates along the first row is $f(0, 1)$. Here, the notation $(0, 1)$ is used

to denote the second sample along the first row. It *does not* mean that these are the values of the physical coordinates when the image was sampled. In general, the value of a digital image at any coordinates (x, y) is denoted $f(x, y)$, where x and y are integers. When we need to refer to specific coordinates (i, j) , we use the notation $f(i, j)$, where the arguments are integers. The section of the real plane spanned by the coordinates of an image is called the *spatial domain*, with x and y being referred to as *spatial variables* or *spatial coordinates*.

Figure 2.18 shows three ways of representing $f(x, y)$. Figure 2.18(a) is a plot of the function, with two axes determining spatial location and the third axis being the values of f as a function of x and y . This representation is useful when working with grayscale sets whose elements are expressed as triplets of the form (x, y, z) , where x and y are spatial coordinates and z is the value of f at coordinates (x, y) . We will work with this representation briefly in Section 2.6.

The representation in Fig. 2.18(b) is more common, and it shows $f(x, y)$ as it would appear on a computer display or photograph. Here, the intensity of each point in the display is proportional to the value of f at that point. In this figure, there are only three equally spaced intensity values. If the intensity is normalized to the interval $[0, 1]$, then each point in the image has the value 0, 0.5, or 1. A monitor or printer converts these three values to black, gray, or white, respectively, as in Fig. 2.18(b). This type of representation includes color images, and allows us to view results at a glance.

As Fig. 2.18(c) shows, the third representation is an array (matrix) composed of the numerical values of $f(x, y)$. This is the representation used for computer processing. In equation form, we write the representation of an $M \times N$ numerical array as

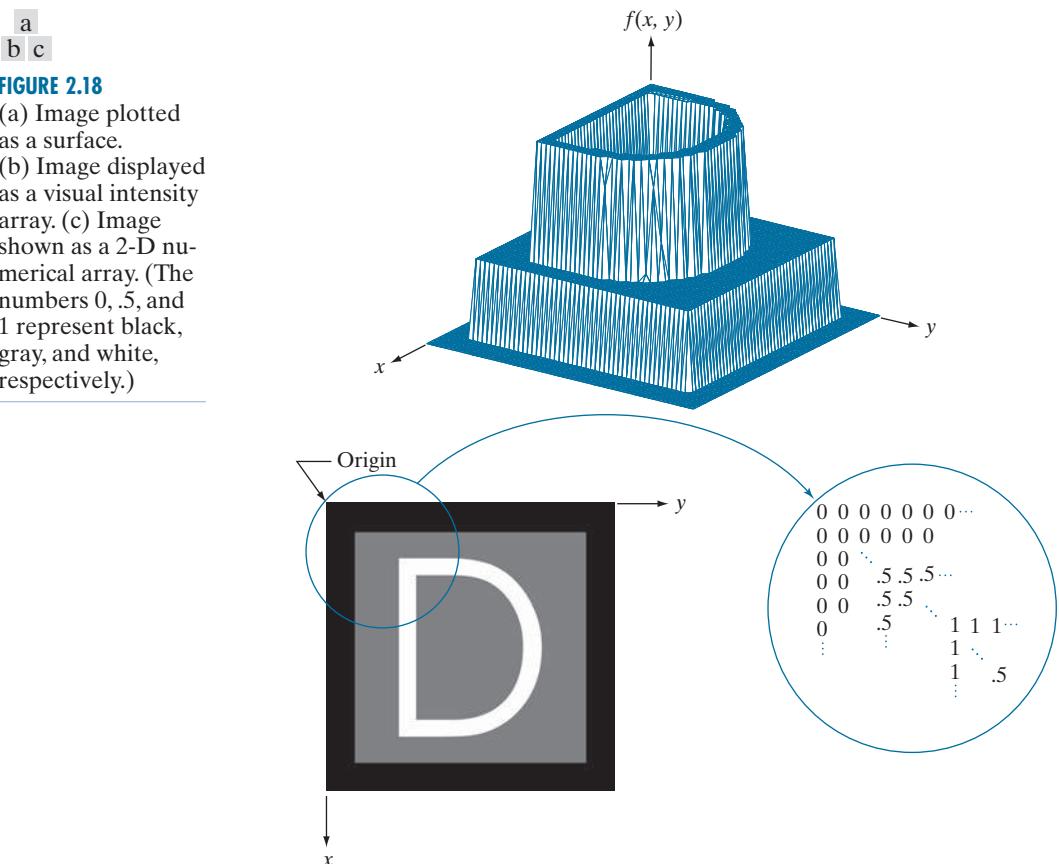
$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \quad (2-9)$$

The right side of this equation is a digital image represented as an array of real numbers. Each element of this array is called an *image element*, *picture element*, *pixel*, or *pel*. We use the terms *image* and *pixel* throughout the book to denote a digital image and its elements. Figure 2.19 shows a graphical representation of an image array, where the x - and y -axis are used to denote the rows and columns of the array. Specific pixels are values of the array at a fixed pair of coordinates. As mentioned earlier, we generally use $f(i, j)$ when referring to a pixel with coordinates (i, j) .

We can also represent a digital image in a traditional matrix form:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \quad (2-10)$$

Clearly, $a_{ij} = f(i, j)$, so Eqs. (2-9) and (2-10) denote identical arrays.



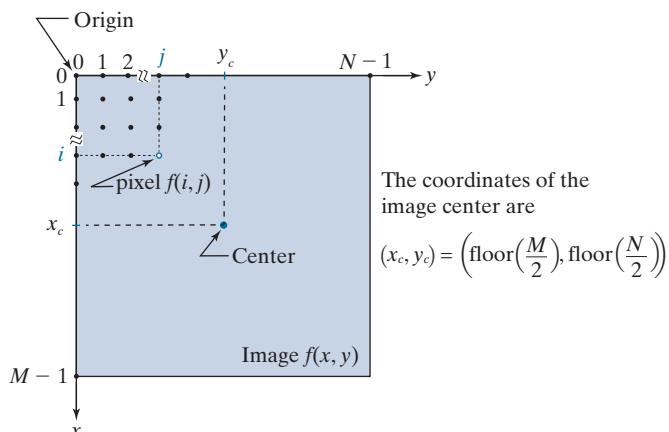
As Fig. 2.19 shows, we define the *origin* of an image at the top left corner. This is a convention based on the fact that many image displays (e.g., TV monitors) sweep an image starting at the top left and moving to the right, one row at a time. More important is the fact that the first element of a matrix is by convention at the top left of the array. Choosing the origin of $f(x, y)$ at that point makes sense mathematically because digital images in reality are matrices. In fact, as you will see, sometimes we use x and y interchangeably in equations with the *rows* (r) and *columns* (c) of a matrix.

It is important to note that the representation in Fig. 2.19, in which the positive x -axis extends downward and the positive y -axis extends to the right, is precisely the right-handed Cartesian coordinate system with which you are familiar,[†] but shown rotated by 90° so that the origin appears on the top, left.

[†]Recall that a right-handed coordinate system is such that, when the index of the right hand points in the direction of the positive x -axis and the middle finger points in the (perpendicular) direction of the positive y -axis, the thumb points up. As Figs. 2.18 and 2.19 show, this indeed is the case in our image coordinate system. In practice, you will also find implementations based on a left-handed system, in which the x - and y -axis are interchanged from the way we show them in Figs. 2.18 and 2.19. For example, MATLAB uses a left-handed system for image processing. Both systems are perfectly valid, provided they are used consistently.

FIGURE 2.19

Coordinate convention used to represent digital images. Because coordinate values are integers, there is a one-to-one correspondence between x and y and the rows (r) and columns (c) of a matrix.



The *floor* of z , sometimes denoted $\lfloor z \rfloor$, is the largest integer that is less than or equal to z . The *ceiling* of z , denoted $\lceil z \rceil$, is the smallest integer that is greater than or equal to z .

See Eq. (2-41) in Section 2.6 for a formal definition of the Cartesian product.

The *center* of an $M \times N$ digital image with origin at $(0,0)$ and range to $(M-1, N-1)$ is obtained by dividing M and N by 2 and rounding *down* to the nearest integer. This operation sometimes is denoted using the floor operator, $\lfloor \cdot \rfloor$, as shown in Fig. 2.19. This holds true for M and N even or odd. For example, the center of an image of size 1023×1024 is at $(511, 512)$. Some programming languages (e.g., MATLAB) start indexing at 1 instead of at 0. The center of an image in that case is found at $(x_c, y_c) = (\text{floor}(M/2) + 1, \text{floor}(N/2) + 1)$.

To express sampling and quantization in more formal mathematical terms, let Z and R denote the set of integers and the set of real numbers, respectively. The sampling process may be viewed as partitioning the xy -plane into a grid, with the coordinates of the center of each cell in the grid being a pair of elements from the Cartesian product Z^2 (also denoted $Z \times Z$) which, as you may recall, is the set of all ordered pairs of elements (z_i, z_j) with z_i and z_j being integers from set Z . Hence, $f(x, y)$ is a digital image if (x, y) are integers from Z^2 and f is a function that assigns an intensity value (that is, a real number from the set of real numbers, R) to each distinct pair of coordinates (x, y) . This functional assignment is the quantization process described earlier. If the intensity levels also are integers, then $R = Z$, and a digital image becomes a 2-D function whose coordinates and amplitude values are integers. This is the representation we use in the book.

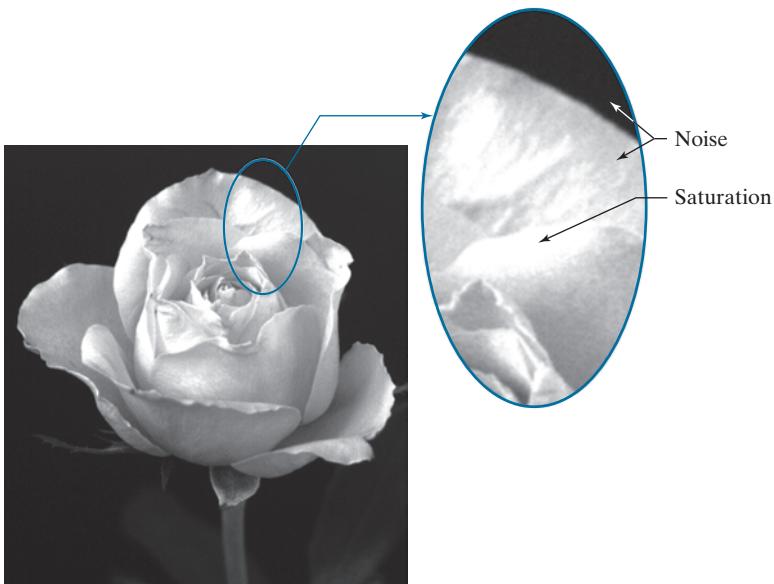
Image digitization requires that decisions be made regarding the values for M, N , and for the number, L , of discrete intensity levels. There are no restrictions placed on M and N , other than they have to be positive integers. However, digital storage and quantizing hardware considerations usually lead to the number of intensity levels, L , being an integer power of two; that is

$$L = 2^k \quad (2-11)$$

where k is an integer. We assume that the discrete levels are equally spaced and that they are integers in the range $[0, L-1]$.

FIGURE 2.20

An image exhibiting saturation and noise. Saturation is the highest value beyond which all intensity values are clipped (note how the entire saturated area has a high, constant intensity level). Visible noise in this case appears as a grainy texture pattern. The dark background is noisier, but the noise is difficult to see.



Sometimes, the range of values spanned by the gray scale is referred to as the *dynamic range*, a term used in different ways in different fields. Here, we define the dynamic range of an imaging system to be the ratio of the maximum measurable intensity to the minimum detectable intensity level in the system. As a rule, the upper limit is determined by *saturation* and the lower limit by *noise*, although noise can be present also in lighter intensities. Figure 2.20 shows examples of saturation and slight visible noise. Because the darker regions are composed primarily of pixels with the minimum detectable intensity, the background in Fig. 2.20 is the noisiest part of the image; however, dark background noise typically is much harder to see.

The dynamic range establishes the lowest and highest intensity levels that a system can represent and, consequently, that an image can have. Closely associated with this concept is *image contrast*, which we define as the difference in intensity between the highest and lowest intensity levels in an image. The *contrast ratio* is the ratio of these two quantities. When an appreciable number of pixels in an image have a high dynamic range, we can expect the image to have high contrast. Conversely, an image with low dynamic range typically has a dull, washed-out gray look. We will discuss these concepts in more detail in Chapter 3.

The number, b , of bits required to store a digital image is

$$b = M \times N \times k \quad (2-12)$$

When $M = N$, this equation becomes

$$b = N^2 k \quad (2-13)$$

FIGURE 2.21

Number of megabytes required to store images for various values of N and k .



Figure 2.21 shows the number of megabytes required to store square images for various values of N and k (as usual, one byte equals 8 bits and a megabyte equals 10^6 bytes).

When an image can have 2^k possible intensity levels, it is common practice to refer to it as a “ k -bit image,” (e.g., a 256-level image is called an *8-bit image*). Note that storage requirements for large 8-bit images (e.g., $10,000 \times 10,000$ pixels) are not insignificant.

LINEAR VS. COORDINATE INDEXING

The convention discussed in the previous section, in which the location of a pixel is given by its 2-D coordinates, is referred to as *coordinate indexing*, or *subscript indexing*. Another type of indexing used extensively in programming image processing algorithms is *linear indexing*, which consists of a 1-D string of nonnegative integers based on computing offsets from coordinates $(0,0)$. There are two principal types of linear indexing, one is based on a row scan of an image, and the other on a column scan.

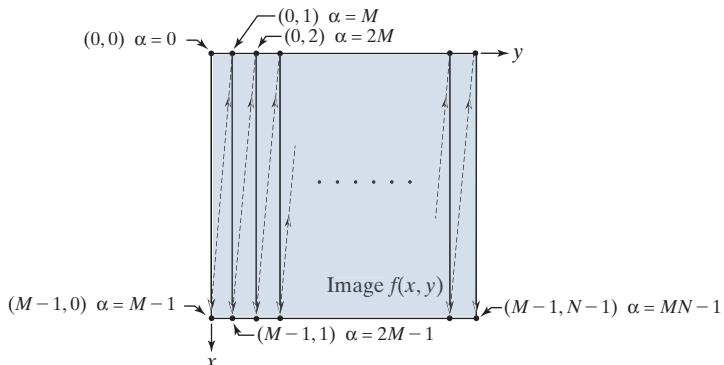
Figure 2.22 illustrates the principle of linear indexing based on a column scan. The idea is to scan an image column by column, starting at the origin and proceeding down and then to the right. The linear index is based on counting pixels as we scan the image in the manner shown in Fig. 2.22. Thus, a scan of the first (leftmost) column yields linear indices 0 through $M - 1$. A scan of the second column yields indices M through $2M - 1$, and so on, until the last pixel in the last column is assigned the linear index value $MN - 1$. Thus, a linear index, denoted by α , has one of MN possible values: $0, 1, 2, \dots, MN - 1$, as Fig. 2.22 shows. The important thing to notice here is that each pixel is assigned a linear index value that identifies it uniquely.

The formula for generating linear indices based on a column scan is straightforward and can be determined by inspection. For any pair of coordinates (x, y) , the corresponding linear index value is

$$\alpha = My + x \quad (2-14)$$

FIGURE 2.22

Illustration of column scanning for generating linear indices. Shown are several 2-D coordinates (in parentheses) and their corresponding linear indices.



Conversely, the coordinate indices for a given linear index value α are given by the equations[†]

$$x = \alpha \bmod M \quad (2-15)$$

and

$$y = (\alpha - x)/M \quad (2-16)$$

Recall that $\alpha \bmod M$ means “the remainder of the division of α by M .” This is a formal way of stating that row numbers repeat themselves at the start of every column. Thus, when $\alpha = 0$, the remainder of the division of 0 by M is 0, so $x = 0$. When $\alpha = 1$, the remainder is 1, and so $x = 1$. You can see that x will continue to be equal to α until $\alpha = M - 1$. When $\alpha = M$ (which is at the beginning of the second column), the remainder is 0, and thus $x = 0$ again, and it increases by 1 until the next column is reached, when the pattern repeats itself. Similar comments apply to Eq. (2-16). See Problem 2.11 for a derivation of the preceding two equations.

SPATIAL AND INTENSITY RESOLUTION

Intuitively, *spatial resolution* is a measure of the smallest discernible detail in an image. Quantitatively, spatial resolution can be stated in several ways, with *line pairs per unit distance*, and *dots (pixels) per unit distance* being common measures. Suppose that we construct a chart with alternating black and white vertical lines, each of width W units (W can be less than 1). The width of a *line pair* is thus $2W$, and there are $W/2$ line pairs per unit distance. For example, if the width of a line is 0.1 mm, there are 5 line pairs per unit distance (i.e., per mm). A widely used definition of image resolution is the largest number of *discernible* line pairs per unit distance (e.g., 100 line pairs per mm). Dots per unit distance is a measure of image resolution used in the printing and publishing industry. In the U.S., this measure usually is expressed as *dots per inch* (dpi). To give you an idea of quality, newspapers are printed with a

[†]When working with modular number systems, it is more accurate to write $x \equiv \alpha \bmod M$, where the symbol \equiv means *congruence*. However, our interest here is just on converting from linear to coordinate indexing, so we use the more familiar equal sign.

resolution of 75 dpi, magazines at 133 dpi, glossy brochures at 175 dpi, and the book page at which you are presently looking was printed at 2400 dpi.

To be meaningful, measures of spatial resolution must be stated with respect to spatial units. Image size by itself does not tell the complete story. For example, to say that an image has a resolution of 1024×1024 pixels is not a meaningful statement without stating the spatial dimensions encompassed by the image. Size by itself is helpful only in making comparisons between imaging capabilities. For instance, a digital camera with a 20-megapixel CCD imaging chip can be expected to have a higher capability to resolve detail than an 8-megapixel camera, assuming that both cameras are equipped with comparable lenses and the comparison images are taken at the same distance.

Intensity resolution similarly refers to the smallest *discernible* change in intensity level. We have considerable discretion regarding the number of spatial samples (pixels) used to generate a digital image, but this is not true regarding the number of intensity levels. Based on hardware considerations, the number of intensity levels usually is an integer power of two, as we mentioned when discussing Eq. (2-11). The most common number is 8 bits, with 16 bits being used in some applications in which enhancement of specific intensity ranges is necessary. Intensity quantization using 32 bits is rare. Sometimes one finds systems that can digitize the intensity levels of an image using 10 or 12 bits, but these are not as common.

Unlike spatial resolution, which must be based on a per-unit-of-distance basis to be meaningful, it is common practice to refer to the number of bits used to quantize intensity as the “*intensity resolution*.” For example, it is common to say that an image whose intensity is quantized into 256 levels has 8 bits of intensity resolution. However, keep in mind that *discernible* changes in intensity are influenced also by noise and saturation values, and by the capabilities of human perception to analyze and interpret details in the context of an entire scene (see Section 2.1). The following two examples illustrate the effects of spatial and intensity resolution on discernible detail. Later in this section, we will discuss how these two parameters interact in determining perceived image quality.

EXAMPLE 2.2: Effects of reducing the spatial resolution of a digital image.

Figure 2.23 shows the effects of reducing the spatial resolution of an image. The images in Figs. 2.23(a) through (d) have resolutions of 930, 300, 150, and 72 dpi, respectively. Naturally, the lower resolution images are smaller than the original image in (a). For example, the original image is of size 2136×2140 pixels, but the 72 dpi image is an array of only 165×166 pixels. In order to facilitate comparisons, all the smaller images were zoomed back to the original size (the method used for zooming will be discussed later in this section). This is somewhat equivalent to “getting closer” to the smaller images so that we can make comparable statements about visible details.

There are some small visual differences between Figs. 2.23(a) and (b), the most notable being a slight distortion in the seconds marker pointing to 60 on the right side of the chronometer. For the most part, however, Fig. 2.23(b) is quite acceptable. In fact, 300 dpi is the typical minimum image spatial resolution used for book publishing, so one would not expect to see much difference between these two images. Figure 2.23(c) begins to show visible degradation (see, for example, the outer edges of the chronometer

a
b
c
d

FIGURE 2.23

Effects of reducing spatial resolution. The images shown are at:
 (a) 930 dpi,
 (b) 300 dpi,
 (c) 150 dpi, and
 (d) 72 dpi.



case and compare the seconds marker with the previous two images). The numbers also show visible degradation. Figure 2.23(d) shows degradation that is visible in most features of the image. When printing at such low resolutions, the printing and publishing industry uses a number of techniques (such as locally varying the pixel size) to produce much better results than those in Fig. 2.23(d). Also, as we will show later in this section, it is possible to improve on the results of Fig. 2.23 by the choice of interpolation method used.

EXAMPLE 2.3: Effects of varying the number of intensity levels in a digital image.

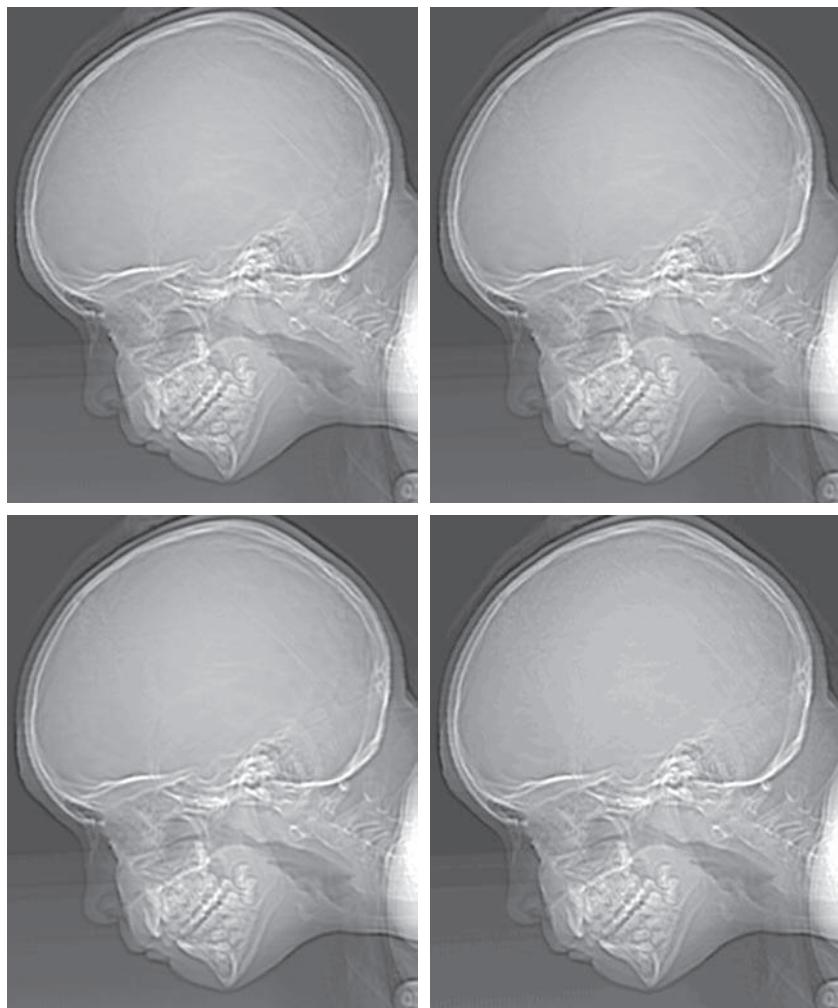
Figure 2.24(a) is a 774×640 CT projection image, displayed using 256 intensity levels (see Chapter 1 regarding CT images). The objective of this example is to reduce the number of intensities of the image from 256 to 2 in integer powers of 2, while keeping the spatial resolution constant. Figures 2.24(b) through (d) were obtained by reducing the number of intensity levels to 128, 64, and 32, respectively (we will discuss in Chapter 3 how to reduce the number of levels).

a	b
c	d

FIGURE 2.24

(a) 774×640 , 256-level image.
 (b)-(d) Image displayed in 128, 64, and 32 intensity levels, while keeping the spatial resolution constant.

(Original image courtesy of the Dr. David R. Pickens, Department of Radiology & Radiological Sciences, Vanderbilt University Medical Center.)



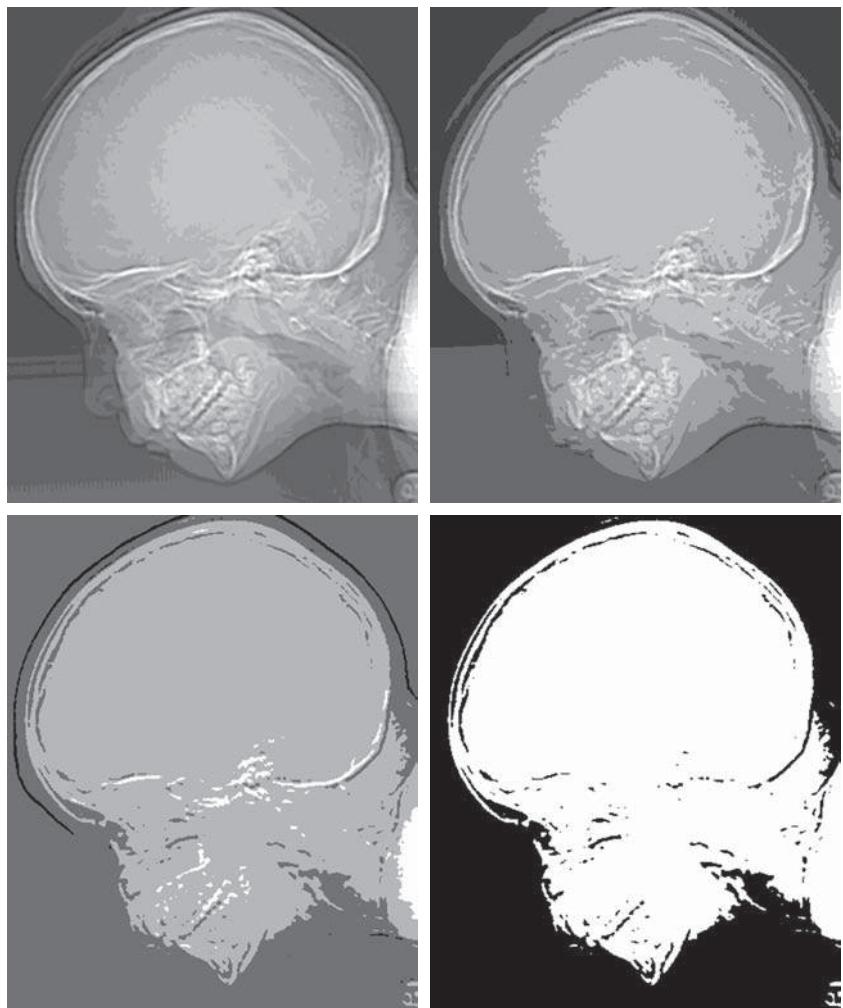
The 128- and 64-level images are visually identical for all practical purposes. However, the 32-level image in Fig. 2.24(d) has a set of almost imperceptible, very fine ridge-like structures in areas of constant intensity. These structures are clearly visible in the 16-level image in Fig. 2.24(e). This effect, caused by using an insufficient number of intensity levels in smooth areas of a digital image, is called *false contouring*, so named because the ridges resemble topographic contours in a map. False contouring generally is quite objectionable in images displayed using 16 or fewer uniformly spaced intensity levels, as the images in Figs. 2.24(e)-(h) show.

As a very rough guideline, and assuming integer powers of 2 for convenience, images of size 256×256 pixels with 64 intensity levels, and printed on a size format on the order of 5×5 cm, are about the lowest spatial and intensity resolution images that can be expected to be reasonably free of objectionable sampling distortions and false contouring.

e f
g h

FIGURE 2.24

(Continued)
(e)-(h) Image displayed in 16, 8, 4, and 2 intensity levels.



The results in Examples 2.2 and 2.3 illustrate the effects produced on image quality by varying spatial and intensity resolution independently. However, these results did not consider any relationships that might exist between these two parameters. An early study by Huang [1965] attempted to quantify experimentally the effects on image quality produced by the interaction of these two variables. The experiment consisted of a set of subjective tests. Images similar to those shown in Fig. 2.25 were used. The woman's face represents an image with relatively little detail; the picture of the cameraman contains an intermediate amount of detail; and the crowd picture contains, by comparison, a large amount of detail.

Sets of these three types of images of various sizes and intensity resolution were generated by varying N and k [see Eq. (2-13)]. Observers were then asked to rank



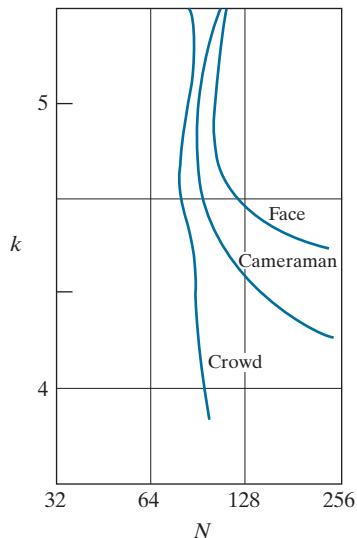
a b c

FIGURE 2.25 (a) Image with a low level of detail. (b) Image with a medium level of detail. (c) Image with a relatively large amount of detail. (Image (b) courtesy of the Massachusetts Institute of Technology.)

them according to their subjective quality. Results were summarized in the form of so-called *isopreference curves* in the Nk -plane. (Figure 2.26 shows average isopreference curves representative of the types of images in Fig. 2.25.) Each point in the Nk -plane represents an image having values of N and k equal to the coordinates of that point. Points lying on an isopreference curve correspond to images of equal subjective quality. It was found in the course of the experiments that the isopreference curves tended to shift right and upward, but their shapes in each of the three image categories were similar to those in Fig. 2.26. These results were not unexpected, because a shift up and right in the curves simply means larger values for N and k , which implies better picture quality.

FIGURE 2.26

Representative isopreference curves for the three types of images in Fig. 2.25.



Observe that isopreference curves tend to become more vertical as the detail in the image increases. This result suggests that for images with a large amount of detail only a few intensity levels may be needed. For example, the isopreference curve in Fig. 2.26 corresponding to the crowd is nearly vertical. This indicates that, for a fixed value of N , the perceived quality for this type of image is nearly independent of the number of intensity levels used (for the range of intensity levels shown in Fig. 2.26). The perceived quality in the other two image categories remained the same in some intervals in which the number of samples was increased, but the number of intensity levels actually decreased. The most likely reason for this result is that a decrease in k tends to increase the apparent contrast, a visual effect often perceived as improved image quality.

IMAGE INTERPOLATION

Interpolation is used in tasks such as zooming, shrinking, rotating, and geometrically correcting digital images. Our principal objective in this section is to introduce interpolation and apply it to image resizing (shrinking and zooming), which are basically image resampling methods. Uses of interpolation in applications such as rotation and geometric corrections will be discussed in Section 2.6.

Interpolation is the process of using known data to estimate values at unknown locations. We begin the discussion of this topic with a short example. Suppose that an image of size 500×500 pixels has to be enlarged 1.5 times to 750×750 pixels. A simple way to visualize zooming is to create an imaginary 750×750 grid with the same pixel spacing as the original image, then shrink it so that it exactly overlays the original image. Obviously, the pixel spacing in the shrunken 750×750 grid will be less than the pixel spacing in the original image. To assign an intensity value to any point in the overlay, we look for its closest pixel in the underlying original image and assign the intensity of that pixel to the new pixel in the 750×750 grid. When intensities have been assigned to all the points in the overlay grid, we expand it back to the specified size to obtain the resized image.

The method just discussed is called *nearest neighbor interpolation* because it assigns to each new location the intensity of its nearest neighbor in the original image (see Section 2.5 regarding neighborhoods). This approach is simple but, it has the tendency to produce undesirable artifacts, such as severe distortion of straight edges. A more suitable approach is *bilinear interpolation*, in which we use the four nearest neighbors to estimate the intensity at a given location. Let (x, y) denote the coordinates of the location to which we want to assign an intensity value (think of it as a point of the grid described previously), and let $v(x, y)$ denote that intensity value. For bilinear interpolation, the assigned value is obtained using the equation

$$v(x, y) = ax + by + cxy + d \quad (2-17)$$

Contrary to what the name suggests, bilinear interpolation is *not a linear operation* because it involves multiplication of coordinates (which is not a linear operation). See Eq. (2-17).

where the four coefficients are determined from the four equations in four unknowns that can be written using the *four* nearest neighbors of point (x, y) . Bilinear interpolation gives much better results than nearest neighbor interpolation, with a modest increase in computational burden.

The next level of complexity is *bicubic interpolation*, which involves the sixteen nearest neighbors of a point. The intensity value assigned to point (x, y) is obtained using the equation

$$v(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j \quad (2-18)$$

The sixteen coefficients are determined from the sixteen equations with sixteen unknowns that can be written using the sixteen nearest neighbors of point (x, y) . Observe that Eq. (2-18) reduces in form to Eq. (2-17) if the limits of both summations in the former equation are 0 to 1. Generally, bicubic interpolation does a better job of preserving fine detail than its bilinear counterpart. Bicubic interpolation is the standard used in commercial image editing applications, such as Adobe Photoshop and Corel Photopaint.

Although images are displayed with integer coordinates, it is possible during processing to work with *subpixel accuracy* by increasing the size of the image using interpolation to “fill the gaps” between pixels in the original image.

EXAMPLE 2.4: Comparison of interpolation approaches for image shrinking and zooming.

Figure 2.27(a) is the same as Fig. 2.23(d), which was obtained by reducing the resolution of the 930 dpi image in Fig. 2.23(a) to 72 dpi (the size shrank from 2136×2140 to 165×166 pixels) and then zooming the reduced image back to its original size. To generate Fig. 2.23(d) we used nearest neighbor interpolation both to shrink and zoom the image. As noted earlier, the result in Fig. 2.27(a) is rather poor. Figures 2.27(b) and (c) are the results of repeating the same procedure but using, respectively, bilinear and bicubic interpolation for both shrinking and zooming. The result obtained by using bilinear interpolation is a significant improvement over nearest neighbor interpolation, but the resulting image is blurred slightly. Much sharper results can be obtained using bicubic interpolation, as Fig. 2.27(c) shows.



a b c

FIGURE 2.27 (a) Image reduced to 72 dpi and zoomed back to its original 930 dpi using nearest neighbor interpolation. This figure is the same as Fig. 2.23(d). (b) Image reduced to 72 dpi and zoomed using bilinear interpolation. (c) Same as (b) but using bicubic interpolation.

It is possible to use more neighbors in interpolation, and there are more complex techniques, such as using *splines* or *wavelets*, that in some instances can yield better results than the methods just discussed. While preserving fine detail is an exceptionally important consideration in image generation for 3-D graphics (for example, see Hughes and Andries [2013]), the extra computational burden seldom is justifiable for general-purpose digital image processing, where bilinear or bicubic interpolation typically are the methods of choice.

2.5 SOME BASIC RELATIONSHIPS BETWEEN PIXELS

In this section, we discuss several important relationships between pixels in a digital image. When referring in the following discussion to particular pixels, we use lower-case letters, such as p and q .

NEIGHBORS OF A PIXEL

A pixel p at coordinates (x, y) has two horizontal and two vertical neighbors with coordinates

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

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

The four *diagonal* neighbors of p have coordinates

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

and are denoted $N_D(p)$. These neighbors, together with the 4-neighbors, are called the *8-neighbors* of p , denoted by $N_8(p)$. The set of image locations of the neighbors of a point p is called the *neighborhood* of p . The neighborhood is said to be *closed* if it contains p . Otherwise, the neighborhood is said to be *open*.

ADJACENCY, CONNECTIVITY, REGIONS, AND BOUNDARIES

Let V be the set of intensity values used to define adjacency. In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1. In a grayscale image, the idea is the same, but set V typically contains more elements. For example, if we are dealing with the adjacency of pixels whose values are in the range 0 to 255, set V could be any subset of these 256 values. We consider three types of adjacency:

1. *4-adjacency*. Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.
2. *8-adjacency*. Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.
3. *m-adjacency* (also called *mixed adjacency*). Two pixels p and q with values from V are m -adjacent if

We use the symbols \cap and \cup to denote set intersection and union, respectively. Given sets A and B , recall that their intersection is the set of elements that are members of both A and B . The union of these two sets is the set of elements that are members of A , of B , or of both. We will discuss sets in more detail in Section 2.6.

- (a) q is in $N_4(p)$, or
 - (b) q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V .

Mixed adjacency is a modification of 8-adjacency, and is introduced to eliminate the ambiguities that may result from using 8-adjacency. For example, consider the pixel arrangement in Fig. 2.28(a) and let $V = \{1\}$. The three pixels at the top of Fig. 2.28(b) show multiple (ambiguous) 8-adjacency, as indicated by the dashed lines. This ambiguity is removed by using m -adjacency, as in Fig. 2.28(c). In other words, the center and upper-right diagonal pixels are not m -adjacent because they do not satisfy condition (b).

A *digital path* (or *curve*) from pixel p with coordinates (x_0, y_0) to pixel q with coordinates (x_n, y_n) is a sequence of distinct pixels with coordinates

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

where points (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$. In this case, n is the *length* of the path. If $(x_0, y_0) = (x_n, y_n)$ the path is a *closed* path. We can define 4-, 8-, or m -paths, depending on the type of adjacency specified. For example, the paths in Fig. 2.28(b) between the top right and bottom right points are 8-paths, and the path in Fig. 2.28(c) is an m -path.

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

Let R represent a subset of pixels in an image. We call R a *region* of the image if R is a connected set. Two regions, R_i and R_j , are said to be *adjacent* if their union forms a connected set. Regions that are not adjacent are said to be *disjoint*. We consider 4- and 8-adjacency when referring to regions. For our definition to make sense, the type of adjacency used must be specified. For example, the two regions of 1's in Fig. 2.28(d) are adjacent only if 8-adjacency is used (according to the definition in the previous

$$\begin{array}{ccccccccc}
 & & & & \left. \begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix} \right\} R_i & & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 1 & 1 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
 & a & b & c & d & e & f & & & & & & & &
 \end{array}$$

FIGURE 2.28 (a) An arrangement of pixels. (b) Pixels that are 8-adjacent (adjacency is shown by dashed lines). (c) m -adjacency. (d) Two regions (of 1's) that are 8-adjacent. (e) The circled point is on the boundary of the 1-valued pixels only if 8-adjacency between the region and background is used. (f) The inner boundary of the 1-valued region does not form a closed path, but its outer boundary does.

paragraph, a 4-path between the two regions does not exist, so their union is not a connected set).

Suppose an image contains K disjoint regions, $R_k, k = 1, 2, \dots, K$, none of which touches the image border.[†] Let R_u denote the union of all the K regions, and let $(R_u)^c$ denote its complement (recall that the *complement* of a set A is the set of points that are not in A). We call all the points in R_u the *foreground*, and all the points in $(R_u)^c$ the *background* of the image.

The *boundary* (also called the *border* or *contour*) of a region R is the set of pixels in R that are adjacent to pixels in the complement of R . Stated another way, the border of a region is the set of pixels in the region that have at least one background neighbor. Here again, we must specify the connectivity being used to define adjacency. For example, the point circled in Fig. 2.28(e) is not a member of the border of the 1-valued region if 4-connectivity is used between the region and its background, because the only possible connection between that point and the background is diagonal. As a rule, adjacency between points in a region and its background is defined using 8-connectivity to handle situations such as this.

The preceding definition sometimes is referred to as the *inner border* of the region to distinguish it from its *outer border*, which is the corresponding border in the background. This distinction is important in the development of border-following algorithms. Such algorithms usually are formulated to follow the outer boundary in order to guarantee that the result will form a closed path. For instance, the inner border of the 1-valued region in Fig. 2.28(f) is the region itself. This border does not satisfy the definition of a closed path. On the other hand, the outer border of the region does form a closed path around the region.

If R happens to be an entire image, then its *boundary* (or *border*) is defined as the set of pixels in the first and last rows and columns of the image. This extra definition is required because an image has no neighbors beyond its border. Normally, when we refer to a region, we are referring to a subset of an image, and any pixels in the boundary of the region that happen to coincide with the border of the image are included implicitly as part of the region boundary.

The concept of an *edge* is found frequently in discussions dealing with regions and boundaries. However, there is a key difference between these two concepts. The boundary of a finite region forms a closed path and is thus a “global” concept. As we will discuss in detail in Chapter 10, edges are formed from pixels with derivative values that exceed a preset threshold. Thus, an edge is a “local” concept that is based on a measure of intensity-level discontinuity at a point. It is possible to link edge points into edge segments, and sometimes these segments are linked in such a way that they correspond to boundaries, but this is not always the case. The one exception in which edges and boundaries correspond is in binary images. Depending on the type of connectivity and edge operators used (we will discuss these in Chapter 10), the edge extracted from a binary region will be the same as the region boundary. This is

[†] We make this assumption to avoid having to deal with special cases. This can be done without loss of generality because if one or more regions touch the border of an image, we can simply pad the image with a 1-pixel-wide border of background values.

intuitive. Conceptually, until we arrive at Chapter 10, it is helpful to think of edges as intensity discontinuities, and of boundaries as closed paths.

DISTANCE MEASURES

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

- (a) $D(p, q) \geq 0$ ($D(p, q) = 0$ iff $p = q$),
- (b) $D(p, q) = D(q, p)$, and
- (c) $D(p, s) \leq D(p, q) + D(q, s)$.

The *Euclidean distance* between p and q is defined as

$$D_e(p, q) = \sqrt{(x - u)^2 + (y - v)^2} \quad (2-19)$$

For this distance measure, the pixels having a distance less than or equal to some value r from (x, y) are the points contained in a disk of radius r centered at (x, y) .

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

$$D_4(p, q) = |x - u| + |y - v| \quad (2-20)$$

In this case, pixels having a D_4 distance from (x, y) that is less than or equal to some value d form a diamond centered at (x, y) . For example, the pixels with D_4 distance ≤ 2 from (x, y) (the center point) form the following contours of constant distance:

$$\begin{matrix} & & & 2 \\ & 2 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 \\ & & 2 \end{matrix}$$

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

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

$$D_8(p, q) = \max(|x - u|, |y - v|) \quad (2-21)$$

In this case, the pixels with D_8 distance from (x, y) less than or equal to some value d form a square centered at (x, y) . For example, the pixels with D_8 distance ≤ 2 form the following contours of constant distance:

$$\begin{matrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{matrix}$$

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

Note that the D_4 and D_8 distances between p and q are independent of any paths that might exist between these points because these distances involve only the coordinates of the points. In the case of m -adjacency, however, the D_m distance between two points is defined as the shortest m -path between the points. In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors. For instance, consider the following arrangement of pixels and assume that p , p_2 , and p_4 have a value of 1, and that p_1 and p_3 can be 0 or 1:

$$\begin{matrix} & p_3 & p_4 \\ p_1 & & p_2 \\ & p & \end{matrix}$$

Suppose that we consider adjacency of pixels valued 1 (i.e., $V = \{1\}$). If p_1 and p_3 are 0, the length of the shortest m -path (the D_m distance) between p and p_4 is 2. If p_1 is 1, then p_2 and p will no longer be m -adjacent (see the definition of m -adjacency given earlier) and the length of the shortest m -path becomes 3 (the path goes through the points $pp_1p_2p_4$). Similar comments apply if p_3 is 1 (and p_1 is 0); in this case, the length of the shortest m -path also is 3. Finally, if both p_1 and p_3 are 1, the length of the shortest m -path between p and p_4 is 4. In this case, the path goes through the sequence of points $pp_1p_2p_3p_4$.

2.6 INTRODUCTION TO THE BASIC MATHEMATICAL TOOLS USED IN DIGITAL IMAGE PROCESSING

You may find it helpful to download and study the review material dealing with probability, vectors, linear algebra, and linear systems. The review is available in the Tutorials section of the book website.

This section has two principal objectives: (1) to introduce various mathematical tools we use throughout the book; and (2) to help you begin developing a “feel” for how these tools are used by applying them to a variety of basic image-processing tasks, some of which will be used numerous times in subsequent discussions.

ELEMENTWISE VERSUS MATRIX OPERATIONS

An *elementwise operation* involving one or more images is carried out on a *pixel-by-pixel* basis. We mentioned earlier in this chapter that images can be viewed equivalently as matrices. In fact, as you will see later in this section, there are many situations in which operations between images are carried out using matrix theory. It is for this reason that a clear distinction must be made between elementwise and matrix operations. For example, consider the following 2×2 images (matrices):

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

The *elementwise product* (often denoted using the symbol \odot or \otimes) of these two images is

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \odot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

The elementwise product of two matrices is also called the *Hadamard product* of the matrices.

The symbol \ominus is often used to denote *elementwise division*.

That is, the elementwise product is obtained by multiplying pairs of *corresponding* pixels. On the other hand, the *matrix product* of the images is formed using the rules of matrix multiplication:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

We assume elementwise operations throughout the book, unless stated otherwise. For example, when we refer to raising an image to a power, we mean that each individual pixel is raised to that power; when we refer to dividing an image by another, we mean that the division is between corresponding pixel pairs, and so on. The terms *elementwise addition* and *subtraction* of two images are redundant because these are elementwise operations by definition. However, you may see them used sometimes to clarify notational ambiguities.

LINEAR VERSUS NONLINEAR OPERATIONS

One of the most important classifications of an image processing method is whether it is linear or nonlinear. Consider a general operator, \mathcal{H} , that produces an output image, $g(x, y)$, from a given input image, $f(x, y)$:

$$\mathcal{H}[f(x, y)] = g(x, y) \quad (2-22)$$

Given two arbitrary constants, a and b , and two arbitrary images $f_1(x, y)$ and $f_2(x, y)$, \mathcal{H} is said to be a *linear operator* if

$$\begin{aligned} \mathcal{H}[af_1(x, y) + bf_2(x, y)] &= a\mathcal{H}[f_1(x, y)] + b\mathcal{H}[f_2(x, y)] \\ &= ag_1(x, y) + bg_2(x, y) \end{aligned} \quad (2-23)$$

This equation indicates that the output of a linear operation applied to the sum of two inputs is the same as performing the operation individually on the inputs and then summing the results. In addition, the output of a linear operation on a constant multiplied by an input is the same as the output of the operation due to the original input multiplied by that constant. The first property is called the property of *additivity*, and the second is called the property of *homogeneity*. By definition, an operator that fails to satisfy Eq. (2-23) is said to be *nonlinear*.

As an example, suppose that \mathcal{H} is the sum operator, Σ . The function performed by this operator is simply to sum its inputs. To test for linearity, we start with the left side of Eq. (2-23) and attempt to prove that it is equal to the right side:

$$\begin{aligned} \sum[af_1(x, y) + bf_2(x, y)] &= \sum af_1(x, y) + \sum bf_2(x, y) \\ &= a \sum f_1(x, y) + b \sum f_2(x, y) \\ &= ag_1(x, y) + bg_2(x, y) \end{aligned}$$

These are image summations, not the sums of all the elements of an image.

where the first step follows from the fact that summation is distributive. So, an expansion of the left side is equal to the right side of Eq. (2-23), and we conclude that the sum operator is linear.

On the other hand, suppose that we are working with the max operation, whose function is to find the maximum value of the pixels in an image. For our purposes here, the simplest way to prove that this operator is nonlinear is to find an example that fails the test in Eq. (2-23). Consider the following two images

$$f_1 = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \quad \text{and} \quad f_2 = \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}$$

and suppose that we let $a = 1$ and $b = -1$. To test for linearity, we again start with the left side of Eq. (2-23):

$$\begin{aligned} \max \left\{ (1) \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} + (-1) \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix} \right\} &= \max \left\{ \begin{bmatrix} -6 & -3 \\ -2 & -4 \end{bmatrix} \right\} \\ &= -2 \end{aligned}$$

Working next with the right side, we obtain

$$(1) \max \left\{ \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \right\} + (-1) \max \left\{ \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix} \right\} = 3 + (-1)7 = -4$$

The left and right sides of Eq. (2-23) are not equal in this case, so we have proved that the max operator is nonlinear.

As you will see in the next three chapters, linear operations are exceptionally important because they encompass a large body of theoretical and practical results that are applicable to image processing. The scope of nonlinear operations is considerably more limited. However, you will encounter in the following chapters several nonlinear image processing operations whose performance far exceeds what is achievable by their linear counterparts.

ARITHMETIC OPERATIONS

Arithmetic operations between two images $f(x, y)$ and $g(x, y)$ are denoted as

$$\begin{aligned} s(x, y) &= f(x, y) + g(x, y) \\ d(x, y) &= f(x, y) - g(x, y) \\ p(x, y) &= f(x, y) \times g(x, y) \\ v(x, y) &= f(x, y) \div g(x, y) \end{aligned} \tag{2-24}$$

These are elementwise operations which, as noted earlier in this section, means that they are performed between corresponding pixel pairs in f and g for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$. As usual, M and N are the row and column sizes of the images. Clearly, s , d , p , and v are images of size $M \times N$ also. Note that image arithmetic in the manner just defined involves images of the same size. The following examples illustrate the important role of arithmetic operations in digital image processing.

EXAMPLE 2.5: Using image addition (averaging) for noise reduction.

Suppose that $g(x, y)$ is a corrupted image formed by the addition of noise, $\eta(x, y)$, to a *noiseless* image $f(x, y)$; that is,

$$g(x, y) = f(x, y) + \eta(x, y) \quad (2-25)$$

where the assumption is that at every pair of coordinates (x, y) the noise is uncorrelated[†] and has zero average value. We assume also that the noise and image values are uncorrelated (this is a typical assumption for additive noise). The objective of the following procedure is to reduce the noise content of the output image by adding a set of noisy input images, $\{g_i(x, y)\}$. This is a technique used frequently for image enhancement.

If the noise satisfies the constraints just stated, it can be shown (Problem 2.26) that if an image $\bar{g}(x, y)$ is formed by averaging K different noisy images,

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^K g_i(x, y) \quad (2-26)$$

then it follows that

$$E\{\bar{g}(x, y)\} = f(x, y) \quad (2-27)$$

and

$$\sigma_{\bar{g}(x, y)}^2 = \frac{1}{K} \sigma_{\eta(x, y)}^2 \quad (2-28)$$

where $E\{\bar{g}(x, y)\}$ is the expected value of $\bar{g}(x, y)$, and $\sigma_{\bar{g}(x, y)}^2$ and $\sigma_{\eta(x, y)}^2$ are the variances of $\bar{g}(x, y)$ and $\eta(x, y)$, respectively, all at coordinates (x, y) . These variances are arrays of the same size as the input image, and there is a scalar variance value for each pixel location.

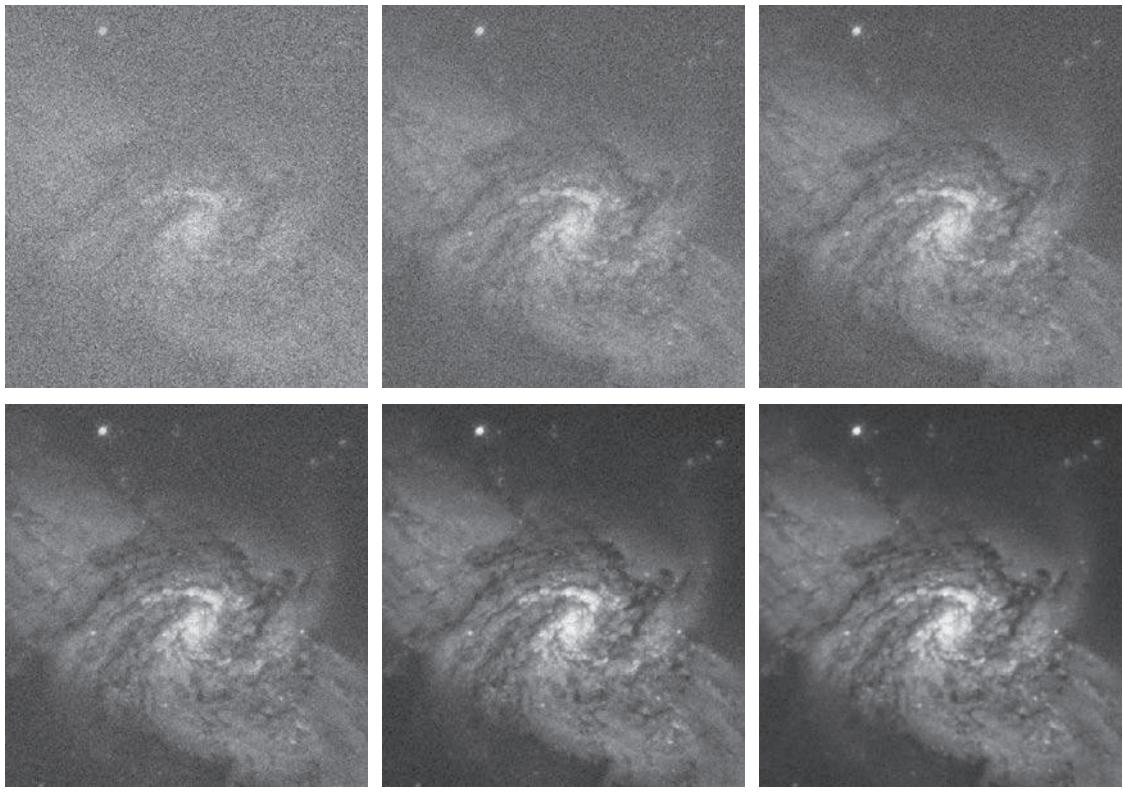
The standard deviation (square root of the variance) at any point (x, y) in the average image is

$$\sigma_{\bar{g}(x, y)} = \frac{1}{\sqrt{K}} \sigma_{\eta(x, y)} \quad (2-29)$$

As K increases, Eqs. (2-28) and (2-29) indicate that the variability (as measured by the variance or the standard deviation) of the pixel values at each location (x, y) decreases. Because $E\{\bar{g}(x, y)\} = f(x, y)$, this means that $\bar{g}(x, y)$ approaches the noiseless image $f(x, y)$ as the number of noisy images used in the averaging process increases. In order to avoid blurring and other artifacts in the output (average) image, it is necessary that the images $g_i(x, y)$ be *registered* (i.e., spatially aligned).

An important application of image averaging is in the field of astronomy, where imaging under very low light levels often cause sensor noise to render individual images virtually useless for analysis (lowering the temperature of the sensor helps reduce noise). Figure 2.29(a) shows an 8-bit image of the Galaxy Pair NGC 3314, in which noise corruption was simulated by adding to it Gaussian noise with zero mean and a standard deviation of 64 intensity levels. This image, which is representative of noisy astronomical images taken under low light conditions, is useless for all practical purposes. Figures 2.29(b) through (f) show the results of averaging 5, 10, 20, 50, and 100 images, respectively. We see from Fig. 2.29(b) that an average of only 10 images resulted in some visible improvement. According to Eq.

[†]The variance of a random variable z with mean \bar{z} is defined as $E[(z - \bar{z})^2]$, where $E[\cdot]$ is the expected value of the argument. The covariance of two random variables z_i and z_j is defined as $E[(z_i - \bar{z}_i)(z_j - \bar{z}_j)]$. If the variables are uncorrelated, their covariance is 0, and vice versa. (Do not confuse correlation and statistical independence. If two random variables are statistically independent, their correlation is zero. However, the converse is not true in general.)



a b c
d e f

FIGURE 2.29 (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)-(f) Result of averaging 5, 10, 20, 50, and 1,00 noisy images, respectively. All images are of size 566×598 pixels, and all were scaled so that their intensities would span the full [0, 255] intensity scale. (Original image courtesy of NASA.)

(2-29), the standard deviation of the noise in Fig. 2.29(b) is less than half ($1/\sqrt{5} = 0.45$) the standard deviation of the noise in Fig. 2.29(a), or $(0.45)(64) \approx 29$ intensity levels. Similarly, the standard deviations of the noise in Figs. 2.29(c) through (f) are 0.32, 0.22, 0.14, and 0.10 of the original, which translates approximately into 20, 14, 9, and 6 intensity levels, respectively. We see in these images a progression of more visible detail as the standard deviation of the noise decreases. The last two images are visually identical for all practical purposes. This is not unexpected, as the difference between the standard deviations of their noise level is only about 3 intensity levels. According to the discussion in connection with Fig. 2.5, this difference is below what a human generally is able to detect.

EXAMPLE 2.6: Comparing images using subtraction.

Image subtraction is used routinely for enhancing differences between images. For example, the image in Fig. 2.30(b) was obtained by setting to zero the least-significant bit of every pixel in Fig. 2.30(a). Visually, these images are indistinguishable. However, as Fig. 2.30(c) shows, subtracting one image from

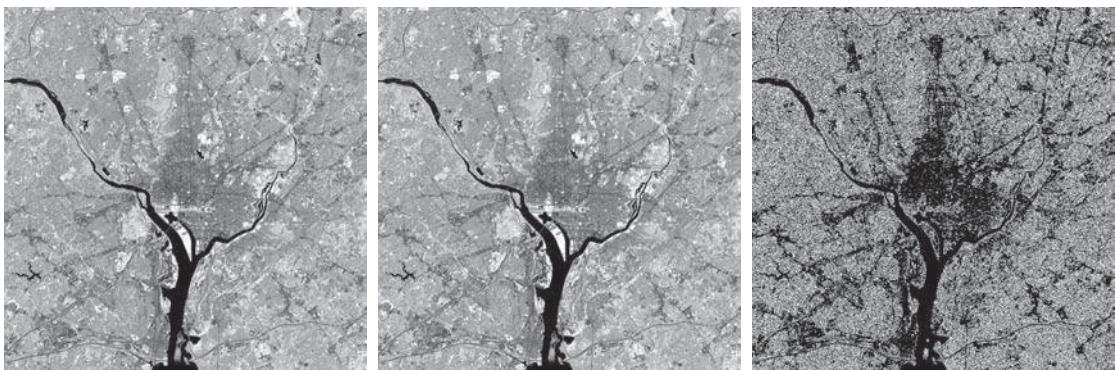


FIGURE 2.30 (a) Infrared image of the Washington, D.C. area. (b) Image resulting from setting to zero the least significant bit of every pixel in (a). (c) Difference of the two images, scaled to the range [0, 255] for clarity. (Original image courtesy of NASA.)

the other clearly shows their differences. Black (0) values in the difference image indicate locations where there is no difference between the images in Figs. 2.30(a) and (b).

We saw in Fig. 2.23 that detail was lost as the resolution was reduced in the chronometer image shown in Fig. 2.23(a). A vivid indication of image change as a function of resolution can be obtained by displaying the differences between the original image and its various lower-resolution counterparts. Figure 2.31(a) shows the difference between the 930 dpi and 72 dpi images. As you can see, the differences are quite noticeable. The intensity at any point in the difference image is proportional to the magnitude of the numerical difference between the two images at that point. Therefore, we can analyze which areas of the original image are affected the most when resolution is reduced. The next two images in Fig. 2.31 show proportionally less overall intensities, indicating smaller differences between the 930 dpi image and 150 dpi and 300 dpi images, as expected.



FIGURE 2.31 (a) Difference between the 930 dpi and 72 dpi images in Fig. 2.23. (b) Difference between the 930 dpi and 150 dpi images. (c) Difference between the 930 dpi and 300 dpi images.

As a final illustration, we discuss briefly an area of medical imaging called *mask mode radiography*, a commercially successful and highly beneficial use of image subtraction. Consider image differences of the form

$$g(x, y) = f(x, y) - h(x, y) \quad (2-30)$$

In this case $h(x, y)$, the *mask*, is an X-ray image of a region of a patient's body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source. The procedure consists of injecting an X-ray contrast medium into the patient's bloodstream, taking a series of images called *live images* [samples of which are denoted as $f(x, y)$] of the same anatomical region as $h(x, y)$, and subtracting the mask from the series of incoming live images after injection of the contrast medium. The net effect of subtracting the mask from each sample live image is that the areas that are different between $f(x, y)$ and $h(x, y)$ appear in the output image, $g(x, y)$, as enhanced detail. Because images can be captured at TV rates, this procedure outputs a video showing how the contrast medium propagates through the various arteries in the area being observed.

Figure 2.32(a) shows a mask X-ray image of the top of a patient's head prior to injection of an iodine medium into the bloodstream, and Fig. 2.32(b) is a sample of a live image taken after the medium was

a	b
c	d

FIGURE 2.32

Digital subtraction angiography.
 (a) Mask image.
 (b) A live image.
 (c) Difference between (a) and (b). (d) Enhanced difference image.
 (Figures (a) and (b) courtesy of the Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)



injected. Figure 2.32(c) is the difference between (a) and (b). Some fine blood vessel structures are visible in this image. The difference is clear in Fig. 2.32(d), which was obtained by sharpening the image and enhancing its contrast (we will discuss these techniques in the next chapter). Figure 2.32(d) is a “snapshot” of how the medium is propagating through the blood vessels in the subject’s brain.

EXAMPLE 2.7: Using image multiplication and division for shading correction and for masking.

An important application of image multiplication (and division) is *shading correction*. Suppose that an imaging sensor produces images that can be modeled as the product of a “perfect image,” denoted by $f(x, y)$, times a shading function, $h(x, y)$; that is, $g(x, y) = f(x, y)h(x, y)$. If $h(x, y)$ is known or can be estimated, we can obtain $f(x, y)$ (or an estimate of it) by multiplying the sensed image by the inverse of $h(x, y)$ (i.e., dividing g by h using elementwise division). If access to the imaging system is possible, we can obtain a good approximation to the shading function by imaging a target of constant intensity. When the sensor is not available, we often can estimate the shading pattern directly from a shaded image using the approaches discussed in Sections 3.5 and 9.8. Figure 2.33 shows an example of shading correction using an estimate of the shading pattern. The corrected image is not perfect because of errors in the shading pattern (this is typical), but the result definitely is an improvement over the shaded image in Fig. 2.33 (a). See Section 3.5 for a discussion of how we estimated Fig. 2.33 (b). Another use of image multiplication is in *masking*, also called *region of interest* (ROI) operations. As Fig. 2.34 shows, the process consists of multiplying a given image by a mask image that has 1’s in the ROI and 0’s elsewhere. There can be more than one ROI in the mask image, and the shape of the ROI can be arbitrary.

A few comments about implementing image arithmetic operations are in order before we leave this section. In practice, most images are displayed using 8 bits (even 24-bit color images consist of three separate 8-bit channels). Thus, we expect image values to be in the range from 0 to 255. When images are saved in a standard image format, such as TIFF or JPEG, conversion to this range is automatic. When image values exceed the allowed range, clipping or scaling becomes necessary. For example, the values in the difference of two 8-bit images can range from a minimum of -255



FIGURE 2.33 Shading correction. (a) Shaded test pattern. (b) Estimated shading pattern. (c) Product of (a) by the reciprocal of (b). (See Section 3.5 for a discussion of how (b) was estimated.)



a b c

FIGURE 2.34 (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

to a maximum of 255, and the values of the sum of two such images can range from 0 to 510. When converting images to eight bits, many software applications simply set all negative values to 0 and set to 255 all values that exceed this limit. Given a digital image g resulting from one or more arithmetic (or other) operations, an approach guaranteeing that the full range of g values is “captured” into a fixed number of bits is as follows. First, we perform the operation

$$g_m = g - \min(g) \quad (2-31)$$

These are elementwise subtraction and division.

which creates an image whose minimum value is 0. Then, we perform the operation

$$g_s = K [g_m / \max(g_m)] \quad (2-32)$$

which creates a scaled image, g_s , whose values are in the range $[0, K]$. When working with 8-bit images, setting $K = 255$ gives us a scaled image whose intensities span the full 8-bit scale from 0 to 255. Similar comments apply to 16-bit images or higher. This approach can be used for all arithmetic operations. When performing division, we have the extra requirement that a small number should be added to the pixels of the divisor image to avoid division by 0.

SET AND LOGICAL OPERATIONS

In this section, we discuss the basics of set theory. We also introduce and illustrate some important set and logical operations.

Basic Set Operations

A *set* is a collection of distinct objects. If a is an *element* of set A , then we write

$$a \in A \quad (2-33)$$

Similarly, if a is not an element of A we write

$$a \notin A \quad (2-34)$$

The set with no elements is called the *null* or *empty set*, and is denoted by \emptyset .

A set is denoted by the contents of two braces: $\{ \cdot \}$. For example, the expression

$$C = \{c \mid c = -d, d \in D\}$$

means that C is the set of elements, c , such that c is formed by multiplying each of the elements of set D by -1 .

If every element of a set A is also an element of a set B , then A is said to be a *subset* of B , denoted as

$$A \subseteq B \quad (2-35)$$

The *union* of two sets A and B , denoted as

$$C = A \cup B \quad (2-36)$$

is a set C consisting of elements belonging *either* to A , to B , or to *both*. Similarly, the *intersection* of two sets A and B , denoted by

$$D = A \cap B \quad (2-37)$$

is a set D consisting of elements belonging to *both* A and B . Sets A and B are said to be *disjoint* or *mutually exclusive* if they have no elements in common, in which case,

$$A \cap B = \emptyset \quad (2-38)$$

The *sample space*, Ω , (also called the *set universe*) is the set of all possible set elements in a given application. By definition, these set elements are members of the sample space for that application. For example, if you are working with the set of real numbers, then the sample space is the real line, which contains all the real numbers. In image processing, we typically define Ω to be the rectangle containing all the pixels in an image.

The *complement* of a set A is the set of elements that are not in A :

$$A^c = \{w \mid w \notin A\} \quad (2-39)$$

The *difference* of two sets A and B , denoted $A - B$, is defined as

$$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c \quad (2-40)$$

This is the set of elements that belong to A , but not to B . We can define A^c in terms of Ω and the set difference operation; that is, $A^c = \Omega - A$. Table 2.1 shows several important set properties and relationships.

Figure 2.35 shows diagrammatically (in so-called *Venn diagrams*) some of the set relationships in Table 2.1. The shaded areas in the various figures correspond to the set operation indicated above or below the figure. Figure 2.35(a) shows the sample set, Ω . As no earlier, this is the set of all possible elements in a given application. Figure 2.35(b) shows that the complement of a set A is the set of all elements in Ω that are not in A , which agrees with our earlier definition. Observe that Figs. 2.35(e) and (g) are identical, which proves the validity of Eq. (2-40) using Venn diagrams. This

TABLE 2.1

Some important set operations and relationships.

Description	Expressions
Operations between the sample space and null sets	$\Omega^c = \emptyset; \emptyset^c = \Omega; \Omega \cup \emptyset = \Omega; \Omega \cap \emptyset = \emptyset$
Union and intersection with the null and sample space sets	$A \cup \emptyset = A; A \cap \emptyset = \emptyset; A \cup \Omega = \Omega; A \cap \Omega = A$
Union and intersection of a set with itself	$A \cup A = A; A \cap A = A$
Union and intersection of a set with its complement	$A \cup A^c = \Omega; A \cap A^c = \emptyset$
Commutative laws	$A \cup B = B \cup A$ $A \cap B = B \cap A$
Associative laws	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
Distributive laws	$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$ $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$
DeMorgan's laws	$(A \cup B)^c = A^c \cap B^c$ $(A \cap B)^c = A^c \cup B^c$

is an example of the usefulness of Venn diagrams for proving equivalences between set relationships.

When applying the concepts just discussed to image processing, we let sets represent objects (regions) in a binary image, and the elements of the sets are the (x, y) coordinates of those objects. For example, if we want to know whether two objects, A and B , of a binary image overlap, all we have to do is compute $A \cap B$. If the result is not the empty set, we know that some of the elements of the two objects overlap. Keep in mind that the only way that the operations illustrated in Fig. 2.35 can make sense in the context of image processing is if the images containing the sets are binary, in which case we can talk about set membership based on coordinates, the assumption being that all members of the sets have the same intensity value (typically denoted by 1). We will discuss set operations involving binary images in more detail in the following section and in Chapter 9.

The preceding concepts are not applicable when dealing with grayscale images, because we have not defined yet a mechanism for assigning intensity values to the pixels resulting from a set operation. In Sections 3.8 and 9.6 we will define the union and intersection operations for grayscale values as the maximum and minimum of corresponding pixel pairs, respectively. We define the *complement* of a grayscale image as the pairwise differences between a constant and the intensity of every pixel in the image. The fact that we deal with corresponding pixel pairs tells us that grayscale set operations are elementwise operations, as defined earlier. The following example is a brief illustration of set operations involving grayscale images. We will discuss these concepts further in the two sections just mentioned.

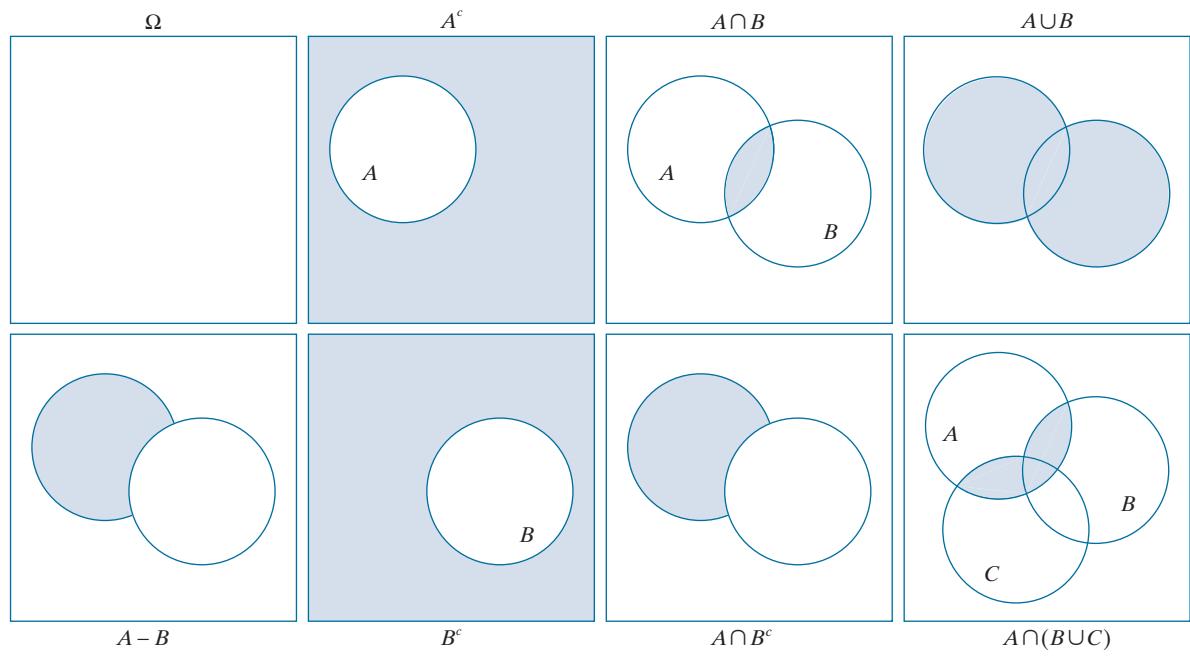


FIGURE 2.35 Venn diagrams corresponding to some of the set operations in Table 2.1. The results of the operations, such as A^c , are shown shaded. Figures (e) and (g) are the same, proving via Venn diagrams that $A - B = A \cap B^c$ [see Eq. (2-40)].

EXAMPLE 2.8: Illustration of set operations involving grayscale images.

Let the elements of a grayscale image be represented by a set A whose elements are triplets of the form (x, y, z) , where x and y are spatial coordinates, and z denotes intensity values. We define the *complement* of A as the set

$$A^c = \{(x, y, K - z) | (x, y, z) \in A\}$$

which is the set of pixels of A whose intensities have been subtracted from a constant K . This constant is equal to the maximum intensity value in the image, $2^k - 1$, where k is the number of bits used to represent z . Let A denote the 8-bit grayscale image in Fig. 2.36(a), and suppose that we want to form the negative of A using grayscale set operations. The negative is the set complement, and this is an 8-bit image, so all we have to do is let $K = 255$ in the set defined above:

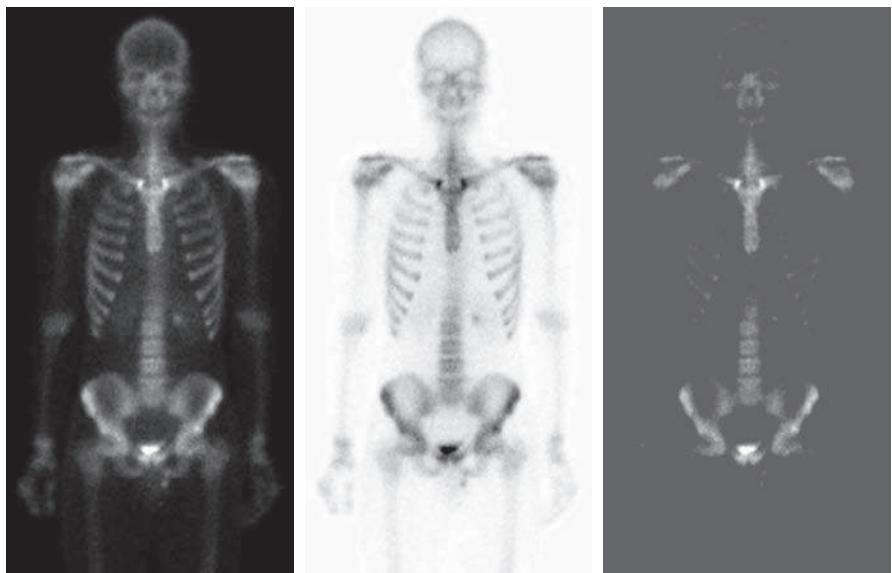
$$A^c = \{(x, y, 255 - z) | (x, y, z) \in A\}$$

Figure 2.36(b) shows the result. We show this only for illustrative purposes. Image negatives generally are computed using an intensity transformation function, as discussed later in this section.

a b c

FIGURE 2.36

Set operations involving grayscale images. (a) Original image. (b) Image negative obtained using grayscale set complementation. (c) The union of image (a) and a constant image. (Original image courtesy of G.E. Medical Systems.)



The *union* of two grayscale sets A and B with the same number of elements is defined as the set

$$A \cup B = \left\{ \max_z(a, b) \mid a \in A, b \in B \right\}$$

where it is understood that the max operation is applied to pairs of corresponding elements. If A and B are grayscale images of the same size, we see that their the union is an array formed from the maximum intensity between pairs of spatially corresponding elements. As an illustration, suppose that A again represents the image in Fig. 2.36(a), and let B denote a rectangular array of the same size as A , but in which all values of z are equal to 3 times the mean intensity, \bar{z} , of the elements of A . Figure 2.36(c) shows the result of performing the set union, in which all values exceeding $3\bar{z}$ appear as values from A and all other pixels have value $3\bar{z}$, which is a mid-gray value.

We follow convention in using the symbol \times to denote the Cartesian product. This is not to be confused with our use of the same symbol throughout the book to denote the size of an M -by- N image (i.e., $M \times N$).

Before leaving the discussion of sets, we introduce some additional concepts that are used later in the book. The *Cartesian product* of two sets X and Y , denoted $X \times Y$, is the set of *all* possible ordered pairs whose first component is a member of X and whose second component is a member of Y . In other words,

$$X \times Y = \left\{ (x, y) \mid x \in X \text{ and } y \in Y \right\} \quad (2-41)$$

For example, if X is a set of M equally spaced values on the x -axis and Y is a set of N equally spaced values on the y -axis, we see that the Cartesian product of these two sets define the coordinates of an M -by- N rectangular array (i.e., the coordinates of an image). As another example, if X and Y denote the specific x - and y -coordinates of a group of 8-connected, 1-valued pixels in a binary image, then set $X \times Y$ represents the region (object) comprised of those pixels.

A *relation* (or, more precisely, a *binary relation*) on a set A is a collection of ordered pairs of elements from A . That is, a binary relation is a subset of the Cartesian product $A \times A$. A binary relation between two sets, A and B , is a subset of $A \times B$.

A *partial order* on a set S is a relation \mathcal{R} on S such that \mathcal{R} is:

- (a) *reflexive*: for any $a \in S$, $a\mathcal{R}a$;
- (b) *transitive*: for any $a, b, c \in S$, $a\mathcal{R}b$ and $b\mathcal{R}c$ implies that $a\mathcal{R}c$;
- (c) *antisymmetric*: for any $a, b \in S$, $a\mathcal{R}b$ and $b\mathcal{R}a$ implies that $a = b$.

where, for example, $a\mathcal{R}b$ reads “ a is related to b .” This means that a and b are in set \mathcal{R} , which itself is a subset of $S \times S$ according to the preceding definition of a relation. A set with a partial order is called a *partially ordered set*.

Let the symbol \preceq denote an ordering relation. An expression of the form

$$a_1 \preceq a_2 \preceq a_3 \preceq \cdots \preceq a_n$$

reads: a_1 precedes a_2 or is the same as a_2 , a_2 precedes a_3 or is the same as a_3 , and so on. When working with numbers, the symbol \preceq typically is replaced by more traditional symbols. For example, the set of real numbers ordered by the relation “less than or equal to” (denoted by \leq) is a partially ordered set (see Problem 2.33). Similarly, the set of natural numbers, paired with the relation “divisible by” (denoted by \div), is a partially ordered set.

Of more interest to us later in the book are strict orderings. A *strict ordering* on a set S is a relation \mathcal{R} on S , such that \mathcal{R} is:

- (a) *antireflexive*: for any $a \in S$, $\neg a\mathcal{R}a$;
- (b) *transitive*: for any $a, b, c \in S$, $a\mathcal{R}b$ and $b\mathcal{R}c$ implies that $a\mathcal{R}c$.

where $\neg a\mathcal{R}a$ means that a is *not related* to a . Let the symbol \prec denote a strict ordering relation. An expression of the form

$$a_1 \prec a_2 \prec a_3 \prec \cdots \prec a_n$$

reads a_1 precedes a_2 , a_2 precedes a_3 , and so on. A set with a strict ordering is called a *strict-ordered set*.

As an example, consider the set composed of the English alphabet of lowercase letters, $S = \{a, b, c, \dots, z\}$. Based on the preceding definition, the ordering

$$a \prec b \prec c \prec \cdots \prec z$$

is strict because no member of the set can precede itself (antireflexivity) and, for any three letters in S , if the first precedes the second, and the second precedes the third, then the first precedes the third (transitivity). Similarly, the set of integers paired with the relation “less than ($<$)” is a strict-ordered set.

Logical Operations

Logical operations deal with TRUE (typically denoted by 1) and FALSE (typically denoted by 0) variables and expressions. For our purposes, this means binary images

composed of *foreground* (1-valued) pixels, and a *background* composed of 0-valued pixels.

We work with set and logical operators on binary images using one of two basic approaches: (1) we can use the *coordinates* of individual regions of foreground pixels in a single image as sets, or (2) we can work with one or more images of the same size and perform logical operations between corresponding pixels in those arrays.

In the first category, a binary image can be viewed as a Venn diagram in which the coordinates of individual regions of 1-valued pixels are treated as sets. The union of these sets with the set composed of 0-valued pixels comprises the set universe, Ω . In this representation, we work with single images using all the set operations defined in the previous section. For example, given a binary image with two 1-valued regions, R_1 and R_2 , we can determine if the regions overlap (i.e., if they have at least one pair of coordinates in common) by performing the set intersection operation $R_1 \cap R_2$ (see Fig. 2.35). In the second approach, we perform logical operations on the pixels of one binary image, or on the corresponding pixels of two or more binary images of the same size.

Logical operators can be defined in terms of truth tables, as Table 2.2 shows for two logical variables a and b . The logical AND operation (also denoted \wedge) yields a 1 (TRUE) only when both a and b are 1. Otherwise, it yields 0 (FALSE). Similarly, the logical OR (\vee) yields 1 when both a or b or *both* are 1, and 0 otherwise. The NOT (\sim) operator is self explanatory. When applied to two binary images, AND and OR operate on pairs of corresponding pixels between the images. That is, they are elementwise operators (see the definition of elementwise operators given earlier in this chapter) in this context. The operators AND, OR, and NOT are *functionally complete*, in the sense that they can be used as the basis for constructing any other logical operator.

Figure 2.37 illustrates the logical operations defined in Table 2.2 using the second approach discussed above. The NOT of binary image B_1 is an array obtained by changing all 1-valued pixels to 0, and vice versa. The AND of B_1 and B_2 contains a 1 at all spatial locations where the corresponding elements of B_1 and B_2 are 1; the operation yields 0's elsewhere. Similarly, the OR of these two images is an array that contains a 1 in locations where the corresponding elements of B_1 , or B_2 , or *both*, are 1. The array contains 0's elsewhere. The result in the fourth row of Fig. 2.37 corresponds to the set of 1-valued pixels in B_1 but not in B_2 . The last row in the figure is the XOR (exclusive OR) operation, which yields 1 in the locations where the corresponding elements of B_1 or B_2 , (but *not both*) are 1. Note that the logical

TABLE 2.2

Truth table defining the logical operators AND(\wedge), OR(\vee), and NOT(\sim).

a	b	$a \text{ AND } b$	$a \text{ OR } b$	$\text{NOT}(a)$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

FIGURE 2.37

Illustration of logical operations involving foreground (white) pixels. Black represents binary 0's and white binary 1's. The dashed lines are shown for reference only. They are not part of the result.



expressions in the last two rows of Fig. 2.37 were constructed using operators from Table 2.2; these are examples of the functionally complete nature of these operators.

We can arrive at the same results in Fig. 2.37 using the first approach discussed above. To do this, we begin by labeling the individual 1-valued regions in each of the two images (in this case there is only one such region in each image). Let A and B denote the set of coordinates of all the 1-valued pixels in images B_1 and B_2 , respectively. Then we form a single array by ORing the two images, while keeping the labels A and B . The result would look like the array $B_1 \text{ OR } B_2$ in Fig. 2.37, but with the two white regions labeled A and B . In other words, the resulting array would look like a Venn diagram. With reference to the Venn diagrams and set operations defined in the previous section, we obtain the results in the rightmost column of Fig. 2.37 using set operations as follows: $A^c = NOT(B_1)$, $A \cap B = B_1 \text{ AND } B_2$, $A \cup B = B_1 \text{ OR } B_2$, and similarly for the other results in Fig. 2.37. We will make extensive use in Chapter 9 of the concepts developed in this section.

SPATIAL OPERATIONS

Spatial operations are performed directly on the pixels of an image. We classify spatial operations into three broad categories: (1) single-pixel operations, (2) neighborhood operations, and (3) geometric spatial transformations.

Single-Pixel Operations

The simplest operation we perform on a digital image is to alter the intensity of its pixels individually using a transformation function, T , of the form:

$$s = T(z) \quad (2-42)$$

where z is the intensity of a pixel in the original image and s is the (mapped) intensity of the corresponding pixel in the processed image. For example, Fig. 2.38 shows the transformation used to obtain the *negative* (sometimes called the *complement*) of an 8-bit image. This transformation could be used, for example, to obtain the negative image in Fig. 2.36, instead of using sets.

Our use of the word “negative” in this context refers to the digital equivalent of a photographic negative, not to the numerical negative of the pixels in the image.

Neighborhood Operations

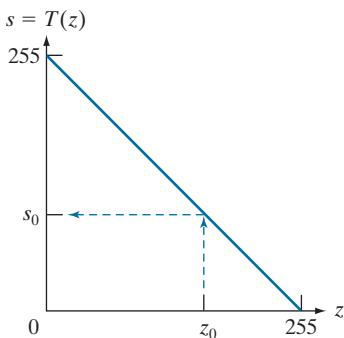
Let S_{xy} denote the set of coordinates of a neighborhood (see Section 2.5 regarding neighborhoods) centered on an arbitrary point (x, y) in an image, f . Neighborhood processing generates a corresponding pixel at the same coordinates in an output (processed) image, g , such that the value of that pixel is determined by a specified operation on the neighborhood of pixels in the input image with coordinates in the set S_{xy} . For example, suppose that the specified operation is to compute the average value of the pixels in a rectangular neighborhood of size $m \times n$ centered on (x, y) . The coordinates of pixels in this region are the elements of set S_{xy} . Figures 2.39(a) and (b) illustrate the process. We can express this averaging operation as

$$g(x, y) = \frac{1}{mn} \sum_{(r, c) \in S_{xy}} f(r, c) \quad (2-43)$$

where r and c are the row and column coordinates of the pixels whose coordinates are in the set S_{xy} . Image g is created by varying the coordinates (x, y) so that the center of the neighborhood moves from pixel to pixel in image f , and then repeating the neighborhood operation at each new location. For instance, the image in Fig. 2.39(d) was created in this manner using a neighborhood of size 41×41 . The

FIGURE 2.38

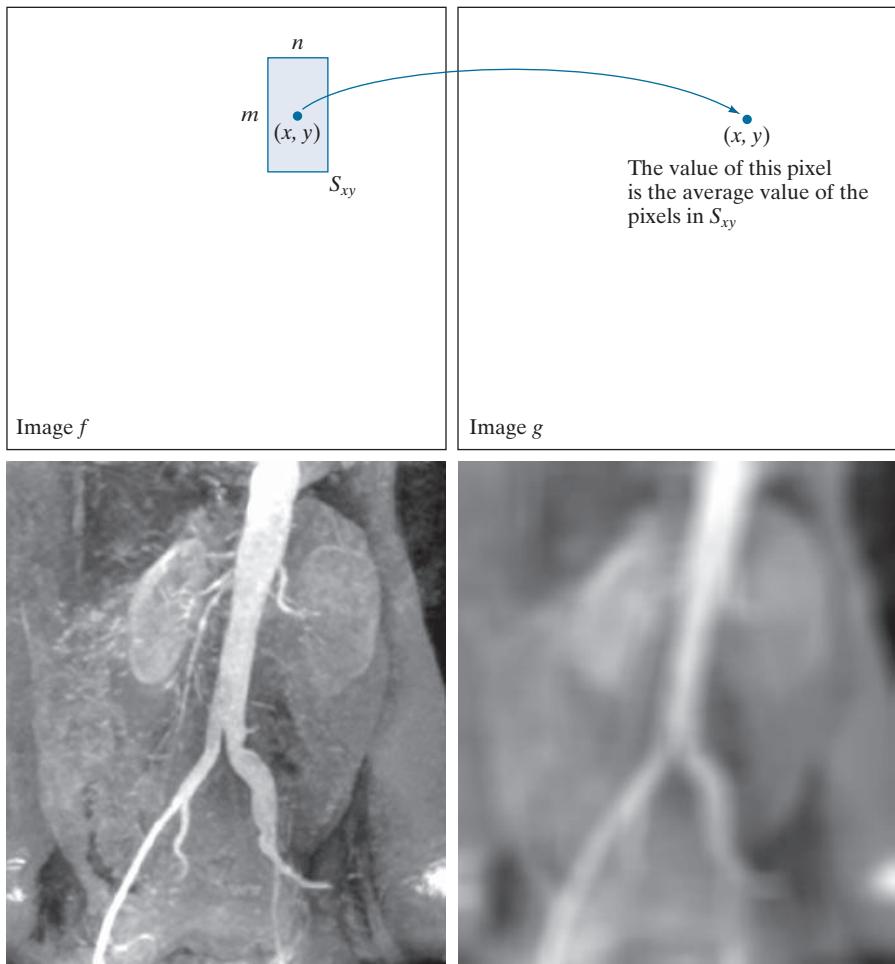
Intensity transformation function used to obtain the digital equivalent of photographic negative of an 8-bit image..



a	b
c	d

FIGURE 2.39

Local averaging using neighborhood processing. The procedure is illustrated in (a) and (b) for a rectangular neighborhood. (c) An aortic angiogram (see Section 1.3). (d) The result of using Eq. (2-43) with $m = n = 41$. The images are of size 790×686 pixels. (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)



net effect is to perform local blurring in the original image. This type of process is used, for example, to eliminate small details and thus render “blobs” corresponding to the largest regions of an image. We will discuss neighborhood processing in Chapters 3 and 5, and in several other places in the book.

Geometric Transformations

We use geometric transformations to modify the spatial arrangement of pixels in an image. These transformations are called *rubber-sheet transformations* because they may be viewed as analogous to “printing” an image on a rubber sheet, then stretching or shrinking the sheet according to a predefined set of rules. Geometric transformations of digital images consist of two basic operations:

1. Spatial transformation of coordinates.
2. Intensity interpolation that assigns intensity values to the spatially transformed pixels.

The transformation of coordinates may be expressed as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2-44)$$

where (x, y) are pixel coordinates in the original image and (x', y') are the corresponding pixel coordinates of the transformed image. For example, the transformation $(x', y') = (x/2, y/2)$ shrinks the original image to half its size in both spatial directions.

Our interest is in so-called *affine transformations*, which include scaling, translation, rotation, and shearing. The key characteristic of an affine transformation in 2-D is that it preserves points, straight lines, and planes. Equation (2-44) can be used to express the transformations just mentioned, except translation, which would require that a constant 2-D vector be added to the right side of the equation. However, it is possible to use homogeneous coordinates to express all four affine transformations using a single 3×3 matrix in the following general form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2-45)$$

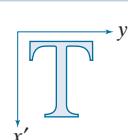
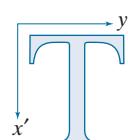
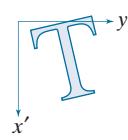
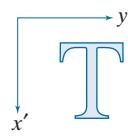
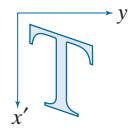
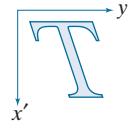
This transformation can *scale*, *rotate*, *translate*, or *shear* an image, depending on the values chosen for the elements of matrix \mathbf{A} . Table 2.3 shows the matrix values used to implement these transformations. A significant advantage of being able to perform all transformations using the unified representation in Eq. (2-45) is that it provides the framework for concatenating a sequence of operations. For example, if we want to resize an image, rotate it, and move the result to some location, we simply form a 3×3 matrix equal to the product of the scaling, rotation, and translation matrices from Table 2.3 (see Problems 2.36 and 2.37).

The preceding transformation moves the coordinates of pixels in an image to new locations. To complete the process, we have to assign intensity values to those locations. This task is accomplished using *intensity interpolation*. We already discussed this topic in Section 2.4. We began that discussion with an example of zooming an image and discussed the issue of intensity assignment to new pixel locations. Zooming is simply scaling, as detailed in the second row of Table 2.3, and an analysis similar to the one we developed for zooming is applicable to the problem of assigning intensity values to the relocated pixels resulting from the other transformations in Table 2.3. As in Section 2.4, we consider nearest neighbor, bilinear, and bicubic interpolation techniques when working with these transformations.

We can use Eq. (2-45) in two basic ways. The first, is a *forward mapping*, which consists of scanning the pixels of the input image and, at each location (x, y) , com-

puting the spatial location (x', y') of the corresponding pixel in the output image using Eq. (2-45) directly. A problem with the forward mapping approach is that two or more pixels in the input image can be transformed to the same location in the output image, raising the question of how to combine multiple output values into a single output pixel value. In addition, it is possible that some output locations may not be assigned a pixel at all. The second approach, called *inverse mapping*, scans the output pixel locations and, at each location (x', y') , computes the corresponding location in the input image using $(x, y) = A^{-1}(x', y')$. It then interpolates (using one of the techniques discussed in Section 2.4) among the nearest input pixels to determine the intensity of the output pixel value. Inverse mappings are more efficient to implement than forward mappings, and are used in numerous commercial implementations of spatial transformations (for example, MATLAB uses this approach).

TABLE 2.3
Affine transformations based on Eq. (2-45).

Transformation Name	Affine Matrix, \mathbf{A}	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to -1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x \cos \theta - y \sin \theta$ $y' = x \sin \theta + y \cos \theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

EXAMPLE 2.9: Image rotation and intensity interpolation.

The objective of this example is to illustrate image rotation using an affine transform. Figure 2.40(a) shows a simple image and Figs. 2.40(b)–(d) are the results (using inverse mapping) of rotating the original image by -21° (in Table 2.3, clockwise angles of rotation are negative). Intensity assignments were computed using nearest neighbor, bilinear, and bicubic interpolation, respectively. A key issue in image rotation is the preservation of straight-line features. As you can see in the enlarged edge sections in Figs. 2.40(f) through (h), nearest neighbor interpolation produced the most jagged edges and, as in Section 2.4, bilinear interpolation yielded significantly improved results. As before, using bicubic interpolation produced slightly better results. In fact, if you compare the progression of enlarged detail in Figs. 2.40(f) to (h), you can see that the transition from white (255) to black (0) is smoother in the last figure because the edge region has more values, and the distribution of those values is better balanced. Although the small intensity differences resulting from bilinear and bicubic interpolation are not always noticeable in human visual analysis, they can be important in processing image data, such as in automated edge following in rotated images.

The size of the spatial rectangle needed to contain a rotated image is larger than the rectangle of the original image, as Figs. 2.41(a) and (b) illustrate. We have two options for dealing with this: (1) we can crop the rotated image so that its size is equal to the size of the original image, as in Fig. 2.41(c), or we can keep the larger image containing the full rotated original, as Fig. 2.41(d). We used the first option in Fig. 2.40 because the rotation did not cause the object of interest to lie outside the bounds of the original rectangle. The areas in the rotated image that do not contain image data must be filled with some value, 0 (black) being the most common. Note that counterclockwise angles of rotation are considered positive. This is a result of the way in which our image coordinate system is set up (see Fig. 2.19), and the way in which rotation is defined in Table 2.3.

Image Registration

Image registration is an important application of digital image processing used to align two or more images of the same scene. In image registration, we have available an *input* image and a *reference* image. The objective is to transform the input image geometrically to produce an output image that is aligned (registered) with the reference image. Unlike the discussion in the previous section where transformation functions are known, the geometric transformation needed to produce the output, registered image generally is not known, and must be estimated.

Examples of image registration include aligning two or more images taken at approximately the same time, but using different imaging systems, such as an MRI (magnetic resonance imaging) scanner and a PET (positron emission tomography) scanner. Or, perhaps the images were taken at different times using the same instruments, such as satellite images of a given location taken several days, months, or even years apart. In either case, combining the images or performing quantitative analysis and comparisons between them requires compensating for geometric distortions caused by differences in viewing angle, distance, orientation, sensor resolution, shifts in object location, and other factors.

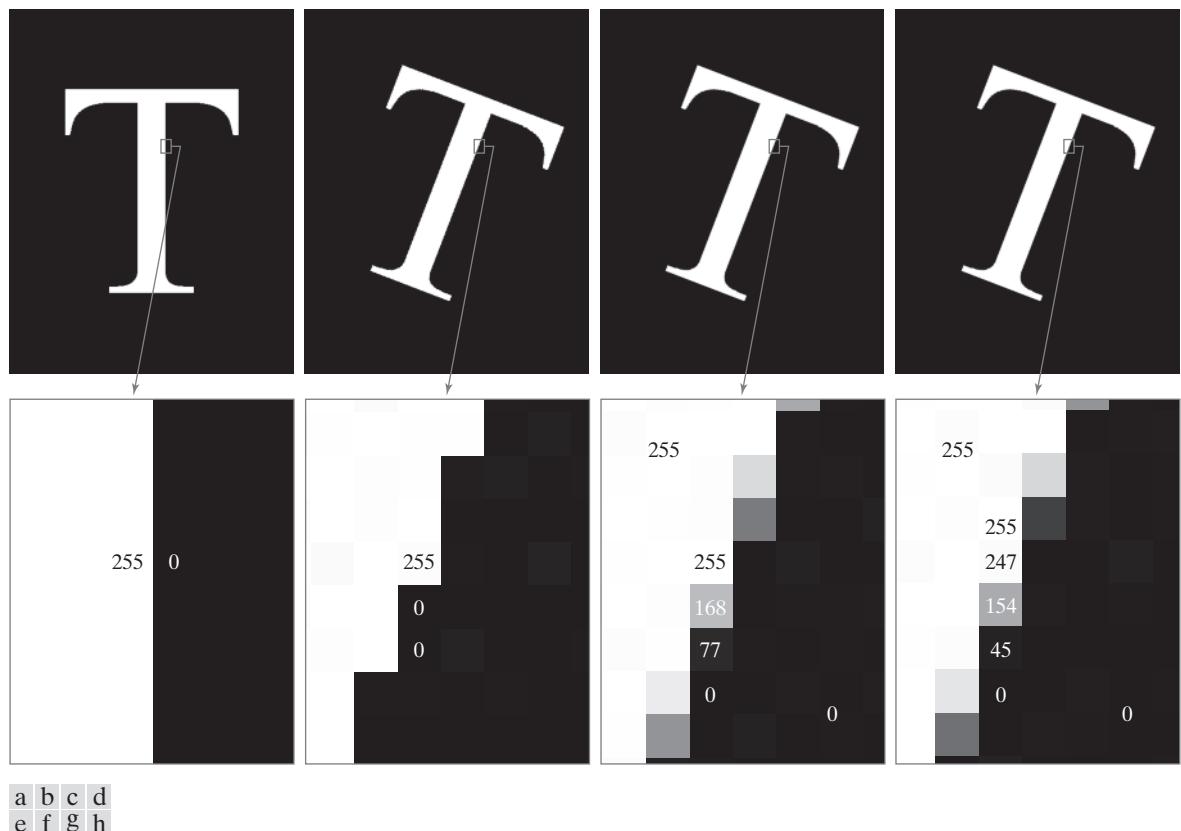


FIGURE 2.40 (a) A 541×421 image of the letter T. (b) Image rotated -21° using nearest-neighbor interpolation for intensity assignments. (c) Image rotated -21° using bilinear interpolation. (d) Image rotated -21° using bicubic interpolation. (e)-(h) Zoomed sections (each square is one pixel, and the numbers shown are intensity values).

One of the principal approaches for solving the problem just discussed is to use *tie points* (also called *control points*). These are corresponding points whose locations are known precisely in the input and reference images. Approaches for selecting tie points range from selecting them interactively to using algorithms that detect these points automatically. Some imaging systems have physical artifacts (such as small metallic objects) embedded in the imaging sensors. These produce a set of known points (called *reseau marks* or *fiducial marks*) directly on all images captured by the system. These known points can then be used as guides for establishing tie points.

The problem of estimating the transformation function is one of modeling. For example, suppose that we have a set of four tie points each in an input and a reference image. A simple model based on a bilinear approximation is given by

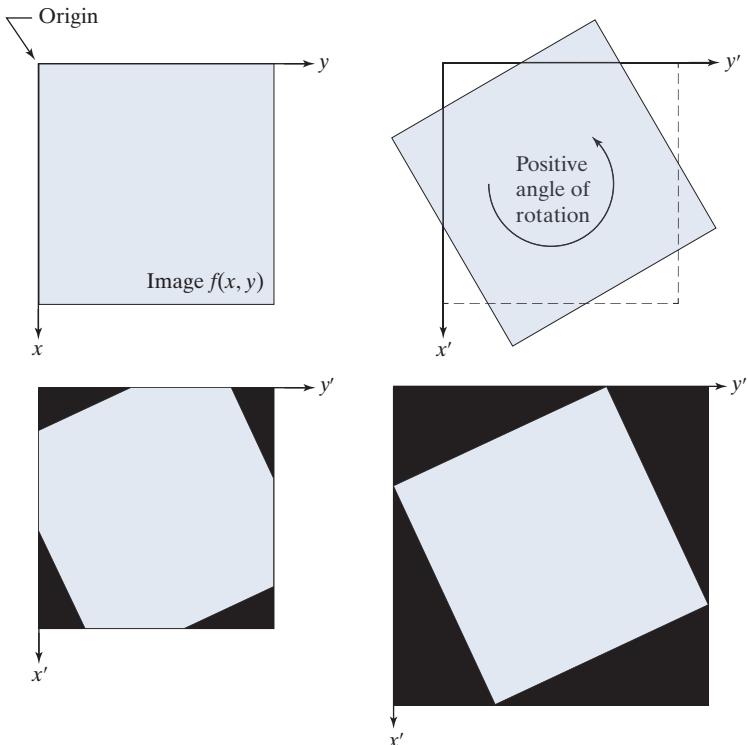
$$x = c_1v + c_2w + c_3vw + c_4 \quad (2-46)$$

and

a	b
c	d

FIGURE 2.41

- (a) A digital image.
- (b) Rotated image (note the counterclockwise direction for a positive angle of rotation).
- (c) Rotated image cropped to fit the same area as the original image.
- (d) Image enlarged to accommodate the entire rotated image.



$$y = c_5v + c_6w + c_7vw + c_8 \quad (2-47)$$

During the estimation phase, (v, w) and (x, y) are the coordinates of tie points in the input and reference images, respectively. If we have four pairs of corresponding tie points in both images, we can write eight equations using Eqs. (2-46) and (2-47) and use them to solve for the eight unknown coefficients, c_1 through c_8 .

Once we have the coefficients, Eqs. (2-46) and (2-47) become our vehicle for transforming all the pixels in the input image. The result is the desired registered image. After the coefficients have been computed, we let (v, w) denote the coordinates of each pixel in the input image, and (x, y) become the corresponding coordinates of the output image. The same set of coefficients, c_1 through c_8 , are used in computing all coordinates (x, y) ; we just step through all (v, w) in the input image to generate the corresponding (x, y) in the output, registered image. If the tie points were selected correctly, this new image should be registered with the reference image, within the accuracy of the bilinear approximation model.

In situations where four tie points are insufficient to obtain satisfactory registration, an approach used frequently is to select a larger number of tie points and then treat the quadrilaterals formed by groups of four tie points as subimages. The subimages are processed as above, with all the pixels within a quadrilateral being transformed using the coefficients determined from the tie points corresponding to that quadrilateral. Then we move to another set of four tie points and repeat the

procedure until all quadrilateral regions have been processed. It is possible to use more complex regions than quadrilaterals, and to employ more complex models, such as polynomials fitted by least squares algorithms. The number of control points and sophistication of the model required to solve a problem is dependent on the severity of the geometric distortion. Finally, keep in mind that the transformations defined by Eqs. (2-46) and (2-47), or any other model for that matter, only map the spatial coordinates of the pixels in the input image. We still need to perform intensity interpolation using any of the methods discussed previously to assign intensity values to the transformed pixels.

EXAMPLE 2.10: Image registration.

Figure 2.42(a) shows a reference image and Fig. 2.42(b) shows the same image, but distorted geometrically by vertical and horizontal shear. Our objective is to use the reference image to obtain tie points and then use them to register the images. The tie points we selected (manually) are shown as small white squares near the corners of the images (we needed only four tie points because the distortion is linear shear in both directions). Figure 2.42(c) shows the registration result obtained using these tie points in the procedure discussed in the preceding paragraphs. Observe that registration was not perfect, as is evident by the black edges in Fig. 2.42(c). The difference image in Fig. 2.42(d) shows more clearly the slight lack of registration between the reference and corrected images. The reason for the discrepancies is error in the manual selection of the tie points. It is difficult to achieve perfect matches for tie points when distortion is so severe.

VECTOR AND MATRIX OPERATIONS

Multispectral image processing is a typical area in which vector and matrix operations are used routinely. For example, you will learn in Chapter 6 that color images are formed in RGB color space by using red, green, and blue component images, as Fig. 2.43 illustrates. Here we see that *each* pixel of an RGB image has three components, which can be organized in the form of a column vector

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \quad (2-48)$$

where z_1 is the intensity of the pixel in the red image, and z_2 and z_3 are the corresponding pixel intensities in the green and blue images, respectively. Thus, an RGB color image of size $M \times N$ can be represented by three component images of this size, or by a total of MN vectors of size 3×1 . A general multispectral case involving n component images (e.g., see Fig. 1.10) will result in n -dimensional vectors:

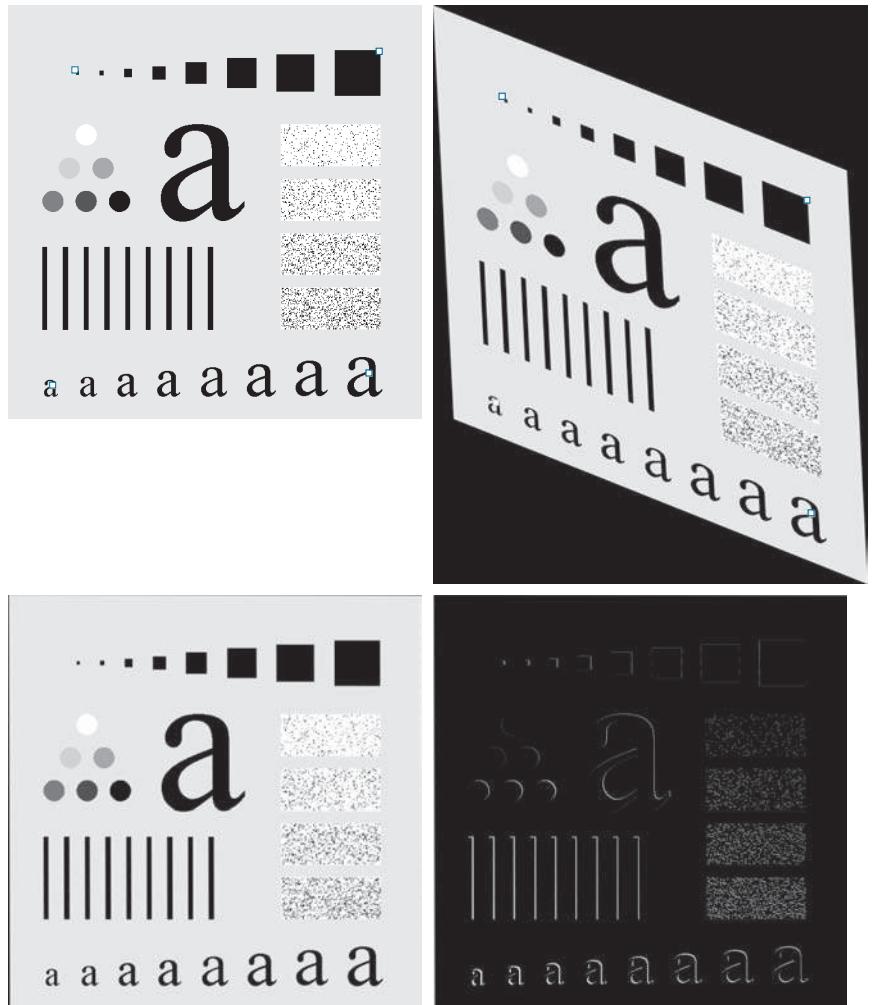
$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \quad (2-49)$$

Recall that an n -dimensional vector can be thought of as a point in n -dimensional Euclidean space.

a
b
c
d

FIGURE 2.42

Image registration.
(a) Reference image.
(b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners.
(c) Registered (output) image (note the errors in the border).
(d) Difference between (a) and (c), showing more registration errors.



We will use this type of vector representation throughout the book.

The *inner product* (also called the *dot product*) of two n -dimensional column vectors **a** and **b** is defined as

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &\triangleq \mathbf{a}^T \mathbf{b} \\ &= a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \\ &= \sum_{i=1}^n a_i b_i \end{aligned} \tag{2-50}$$

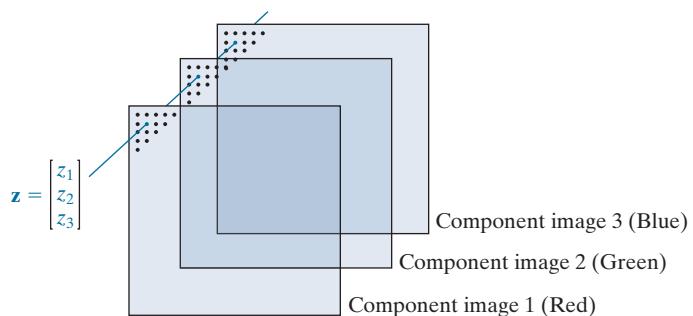
The product \mathbf{ab}^T is called the *outer product* of **a** and **b**. It is a matrix of size $n \times n$.

where T indicates the transpose. The *Euclidean vector norm*, denoted by $\|\mathbf{z}\|$, is defined as the square root of the inner product:

$$\|\mathbf{z}\| = (\mathbf{z}^T \mathbf{z})^{\frac{1}{2}} \tag{2-51}$$

FIGURE 2.43

Forming a vector from corresponding pixel values in three RGB component images.



We recognize this expression as the length of vector \mathbf{z} .

We can use vector notation to express several of the concepts discussed earlier. For example, the Euclidean distance, $D(\mathbf{z}, \mathbf{a})$, between points (vectors) \mathbf{z} and \mathbf{a} in n -dimensional space is defined as the Euclidean vector norm:

$$\begin{aligned} D(\mathbf{z}, \mathbf{a}) &= \|\mathbf{z} - \mathbf{a}\| = \left[(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a}) \right]^{\frac{1}{2}} \\ &= \left[(z_1 - a_1)^2 + (z_2 - a_2)^2 + \dots + (z_n - a_n)^2 \right]^{\frac{1}{2}} \end{aligned} \quad (2-52)$$

This is a generalization of the 2-D Euclidean distance defined in Eq. (2-19).

Another advantage of pixel vectors is in linear transformations, represented as

$$\mathbf{w} = \mathbf{A}(\mathbf{z} - \mathbf{a}) \quad (2-53)$$

where \mathbf{A} is a matrix of size $m \times n$, and \mathbf{z} and \mathbf{a} are column vectors of size $n \times 1$.

As noted in Eq. (2-10), entire images can be treated as matrices (or, equivalently, as vectors), a fact that has important implication in the solution of numerous image processing problems. For example, we can express an image of size $M \times N$ as a column vector of dimension $MN \times 1$ by letting the first M elements of the vector equal the first column of the image, the next M elements equal the second column, and so on. With images formed in this manner, we can express a broad range of linear processes applied to an image by using the notation

$$\mathbf{g} = \mathbf{Hf} + \mathbf{n} \quad (2-54)$$

where \mathbf{f} is an $MN \times 1$ vector representing an input image, \mathbf{n} is an $MN \times 1$ vector representing an $M \times N$ noise pattern, \mathbf{g} is an $MN \times 1$ vector representing a processed image, and \mathbf{H} is an $MN \times MN$ matrix representing a linear process applied to the input image (see the discussion earlier in this chapter regarding linear processes). It is possible, for example, to develop an entire body of generalized techniques for image restoration starting with Eq. (2-54), as we discuss in Section 5.9. We will mention the use of matrices again in the following section, and show other uses of matrices for image processing in numerous chapters in the book.

IMAGE TRANSFORMS

All the image processing approaches discussed thus far operate directly on the pixels of an input image; that is, they work directly in the spatial domain. In some cases, image processing tasks are best formulated by transforming the input images, carrying the specified task in a *transform domain*, and applying the inverse transform to return to the spatial domain. You will encounter a number of different transforms as you proceed through the book. A particularly important class of 2-D *linear transforms*, denoted $T(u,v)$, can be expressed in the general form

$$T(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) r(x,y,u,v) \quad (2-55)$$

where $f(x,y)$ is an input image, $r(x,y,u,v)$ is called a *forward transformation kernel*, and Eq. (2-55) is evaluated for $u = 0, 1, 2, \dots, M - 1$ and $v = 0, 1, 2, \dots, N - 1$. As before, x and y are spatial variables, while M and N are the row and column dimensions of f . Variables u and v are called the *transform variables*. $T(u,v)$ is called the *forward transform* of $f(x,y)$. Given $T(u,v)$, we can recover $f(x,y)$ using the *inverse transform* of $T(u,v)$:

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v) s(x,y,u,v) \quad (2-56)$$

for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$, where $s(x,y,u,v)$ is called an *inverse transformation kernel*. Together, Eqs. (2-55) and (2-56) are called a *transform pair*.

Figure 2.44 shows the basic steps for performing image processing in the linear transform domain. First, the input image is transformed, the transform is then modified by a predefined operation and, finally, the output image is obtained by computing the inverse of the modified transform. Thus, we see that the process goes from the spatial domain to the transform domain, and then back to the spatial domain.

The forward transformation kernel is said to be *separable* if

$$r(x,y,u,v) = r_1(x,u)r_2(y,v) \quad (2-57)$$

In addition, the kernel is said to be *symmetric* if $r_1(x,u)$ is functionally equal to $r_2(y,v)$, so that

$$r(x,y,u,v) = r_1(x,u)r_1(y,v) \quad (2-58)$$

Identical comments apply to the inverse kernel.

The nature of a transform is determined by its kernel. A transform of particular importance in digital image processing is the *Fourier transform*, which has the following forward and inverse kernels:

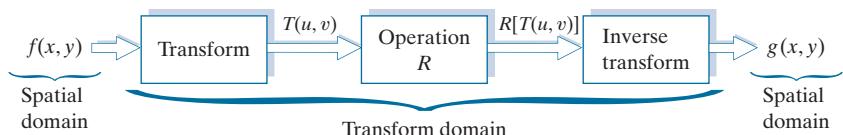
$$r(x,y,u,v) = e^{-j2\pi(ux/M + vy/N)} \quad (2-59)$$

and

$$s(x,y,u,v) = \frac{1}{MN} e^{j2\pi(ux/M + vy/N)} \quad (2-60)$$

FIGURE 2.44

General approach for working in the linear transform domain.



The exponential terms in the Fourier transform kernels can be expanded as sines and cosines of various frequencies. As a result, the domain of the Fourier transform is called the *frequency domain*.

respectively, where $j = \sqrt{-1}$, so these kernels are complex functions. Substituting the preceding kernels into the general transform formulations in Eqs. (2-55) and (2-56) gives us the *discrete Fourier transform pair*:

$$T(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)} \quad (2-61)$$

and

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v) e^{j2\pi(ux/M + vy/N)} \quad (2-62)$$

It can be shown that the Fourier kernels are separable and symmetric (Problem 2.39), and that separable and symmetric kernels allow 2-D transforms to be computed using 1-D transforms (see Problem 2.40). The preceding two equations are of fundamental importance in digital image processing, as you will see in Chapters 4 and 5.

EXAMPLE 2.11: Image processing in the transform domain.

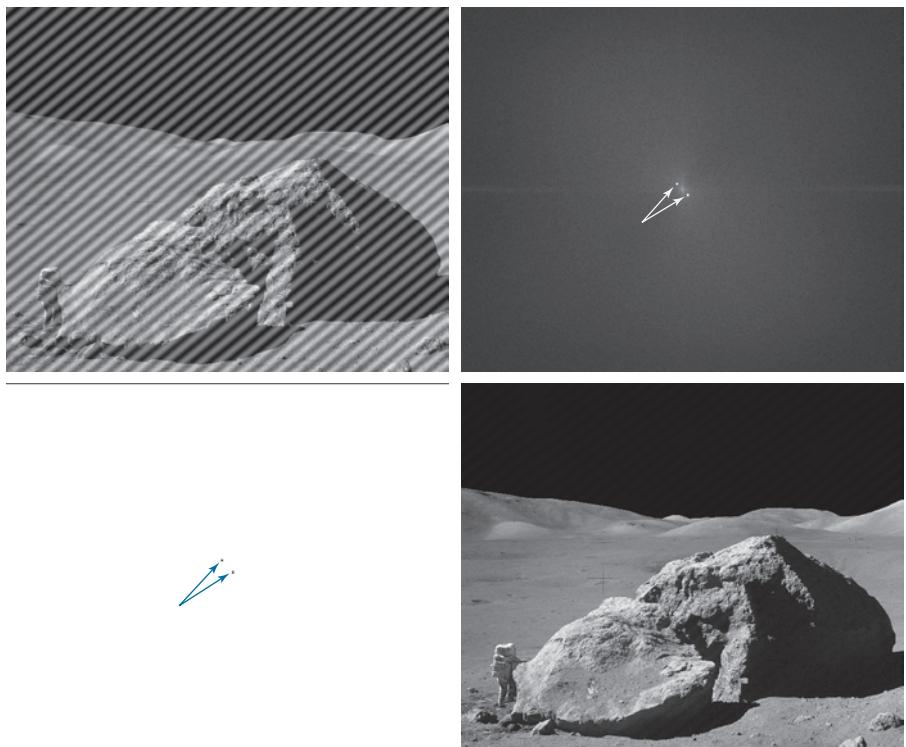
Figure 2.45(a) shows an image corrupted by periodic (sinusoidal) interference. This type of interference can be caused, for example, by a malfunctioning imaging system; we will discuss it in Chapter 5. In the spatial domain, the interference appears as waves of intensity. In the frequency domain, the interference manifests itself as bright bursts of intensity, whose location is determined by the frequency of the sinusoidal interference (we will discuss these concepts in much more detail in Chapters 4 and 5). Typically, the bursts are easily observable in an image of the magnitude of the Fourier transform, $|T(u,v)|$. With reference to the diagram in Fig. 2.44, the corrupted image is $f(x,y)$, the transform in the leftmost box is the Fourier transform, and Fig. 2.45(b) is $|T(u,v)|$ displayed as an image. The bright dots shown are the bursts of intensity mentioned above. Figure 2.45(c) shows a mask image (called a *filter*) with white and black representing 1 and 0, respectively. For this example, the operation in the second box of Fig. 2.44 is to multiply the filter by the transform to remove the bursts associated with the interference. Figure 2.45(d) shows the final result, obtained by computing the inverse of the modified transform. The interference is no longer visible, and previously unseen image detail is now made quite clear. Observe, for example, the fiducial marks (faint crosses) that are used for image registration, as discussed earlier.

When the forward and inverse kernels of a transform are separable and symmetric, and $f(x,y)$ is a square image of size $M \times M$, Eqs. (2-55) and (2-56) can be expressed in matrix form:

a	b
c	d

FIGURE 2.45

- (a) Image corrupted by sinusoidal interference.
 (b) Magnitude of the Fourier transform showing the bursts of energy caused by the interference (the bursts were enlarged for display purposes).
 (c) Mask used to eliminate the energy bursts.
 (d) Result of computing the inverse of the modified Fourier transform.
 (Original image courtesy of NASA.)



$$\mathbf{T} = \mathbf{A}\mathbf{F}\mathbf{A} \quad (2-63)$$

where \mathbf{F} is an $M \times M$ matrix containing the elements of $f(x, y)$ [see Eq. (2-9)], \mathbf{A} is an $M \times M$ matrix with elements $a_{ij} = r_1(i, j)$, and \mathbf{T} is an $M \times M$ transform matrix with elements $T(u, v)$, for $u, v = 0, 1, 2, \dots, M - 1$.

To obtain the inverse transform, we pre- and post-multiply Eq. (2-63) by an inverse transformation matrix \mathbf{B} :

$$\mathbf{B}\mathbf{T}\mathbf{B} = \mathbf{B}\mathbf{A}\mathbf{F}\mathbf{A}\mathbf{B} \quad (2-64)$$

If $\mathbf{B} = \mathbf{A}^{-1}$,

$$\mathbf{F} = \mathbf{B}\mathbf{T}\mathbf{B} \quad (2-65)$$

indicating that \mathbf{F} or, equivalently, $f(x, y)$, can be recovered completely from its forward transform. If \mathbf{B} is not equal to \mathbf{A}^{-1} , Eq. (2-65) yields an approximation:

$$\hat{\mathbf{F}} = \mathbf{B}\mathbf{A}\mathbf{F}\mathbf{A}\mathbf{B} \quad (2-66)$$

In addition to the Fourier transform, a number of important transforms, including the *Walsh*, *Hadamard*, *discrete cosine*, *Haar*, and *slant* transforms, can be expressed in the form of Eqs. (2-55) and (2-56), or, equivalently, in the form of Eqs. (2-63) and (2-65). We will discuss these and other types of image transforms in later chapters.

You may find it useful to consult the tutorials section in the book website for a brief review of probability.

IMAGE INTENSITIES AS RANDOM VARIABLES

We treat image intensities as random quantities in numerous places in the book. For example, let $z_i, i = 0, 1, 2, \dots, L - 1$, denote the values of all possible intensities in an $M \times N$ digital image. The probability, $p(z_k)$, of intensity level z_k occurring in the image is estimated as

$$p(z_k) = \frac{n_k}{MN} \quad (2-67)$$

where n_k is the number of times that intensity z_k occurs in the image and MN is the total number of pixels. Clearly,

$$\sum_{k=0}^{L-1} p(z_k) = 1 \quad (2-68)$$

Once we have $p(z_k)$, we can determine a number of important image characteristics. For example, the mean (average) intensity is given by

$$m = \sum_{k=0}^{L-1} z_k p(z_k) \quad (2-69)$$

Similarly, the variance of the intensities is

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k) \quad (2-70)$$

The variance is a measure of the spread of the values of z about the mean, so it is a useful measure of image contrast. In general, the n th central moment of random variable z about the mean is defined as

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k) \quad (2-71)$$

We see that $\mu_0(z) = 1$, $\mu_1(z) = 0$, and $\mu_2(z) = \sigma^2$. Whereas the mean and variance have an immediately obvious relationship to visual properties of an image, higher-order moments are more subtle. For example, a positive third moment indicates that the intensities are biased to values higher than the mean, a negative third moment would indicate the opposite condition, and a zero third moment would tell us that the intensities are distributed approximately equally on both sides of the mean. These features are useful for computational purposes, but they do not tell us much about the appearance of an image in general.

As you will see in subsequent chapters, concepts from probability play a central role in a broad range of image processing applications. For example, Eq. (2-67) is utilized in Chapter 3 as the basis for image enhancement techniques based on histograms. In Chapter 5, we use probability to develop image restoration algorithms, in Chapter 10 we use probability for image segmentation, in Chapter 11 we use it to describe texture, and in Chapter 12 we use probability as the basis for deriving optimum pattern recognition algorithms.

Summary, References, and Further Reading

The material in this chapter is the foundation for the remainder of the book. For additional reading on visual perception, see Snowden et al. [2012], and the classic book by Cornsweet [1970]. Born and Wolf [1999] discuss light in terms of electromagnetic theory. A basic source for further reading on image sensing is Trussell and Vrheil [2008]. The image formation model discussed in Section 2.3 is from Oppenheim et al. [1968]. The IES Lighting Handbook [2011] is a reference for the illumination and reflectance values used in that section. The concepts of image sampling introduced in Section 2.4 will be covered in detail in Chapter 4. The discussion on experiments dealing with the relationship between image quality and sampling is based on results from Huang [1965]. For further reading on the topics discussed in Section 2.5, see Rosenfeld and Kak [1982], and Klette and Rosenfeld [2004].

See Castleman [1996] for additional reading on linear systems in the context of image processing. The method of noise reduction by image averaging was first proposed by Kohler and Howell [1963]. See Ross [2014] regarding the expected value of the mean and variance of the sum of random variables. See Schröder [2010] for additional reading on logic and sets. For additional reading on geometric spatial transformations see Wolberg [1990] and Hughes and Andries [2013]. For further reading on image registration see Goshtasby [2012]. Bronson and Costa [2009] is a good reference for additional reading on vectors and matrices. See Chapter 4 for a detailed treatment of the Fourier transform, and Chapters 7, 8, and 11 for details on other image transforms. For details on the software aspects of many of the examples in this chapter, see Gonzalez, Woods, and Eddins [2009].

Problems

Solutions to the problems marked with an asterisk () are in the DIP4E Student Support Package (consult the book website: www.ImageProcessingPlace.com).*

- 2.1** If you use a sheet of white paper to shield your eyes when looking directly at the sun, the side of the sheet facing you appears black. Which of the visual processes discussed in Section 2.1 is responsible for this?
- 2.2*** Using the background information provided in Section 2.1, and thinking purely in geometrical terms, estimate the diameter of the smallest printed dot that the eye can discern if the page on which the dot is printed is 0.2 m away from the eyes. Assume for simplicity that the visual system ceases to detect the dot when the image of the dot on the fovea becomes smaller than the diameter of one receptor (cone) in that area of the retina. Assume further that the fovea can be modeled as a square array of dimension 1.5 mm on the side, and that the cones and spaces between the cones are distributed uniformly throughout this array.
- 2.3** Although it is not shown in Fig. 2.10, alternating current is part of the electromagnetic spectrum. Commercial alternating current in the United States has a frequency of 60 Hz. What is the wavelength in kilometers of this component of the spectrum?
- 2.4** You are hired to design the front end of an imaging system for studying the shapes of cells, bacteria, viruses, and proteins. The front end consists in this case of the illumination source(s) and corresponding imaging camera(s). The diameters of circles required to fully enclose individual specimens in each of these categories are 50, 1, 0.1, and 0.01 μm , respectively. In order to perform automated analysis, the smallest detail discernible on a specimen must be 0.001 μm .
 - (a)*** Can you solve the imaging aspects of this problem with a single sensor and camera? If your answer is yes, specify the illumination wavelength band and the type of camera needed. By “type,” we mean the band of the electromagnetic spectrum to which the camera is most sensitive (e.g., infrared).
 - (b)** If your answer in (a) is no, what type of illumination sources and corresponding imaging sensors would you recommend? Specify the light sources and cameras as requested in part (a). Use the minimum number of illumination sources and cameras needed to solve the problem. (*Hint:* From the discussion in

Section 2.2, the illumination required to “see” an object must have a wavelength the same size or smaller than the object.)

- 2.5** You are preparing a report and have to insert in it an image of size 2048×2048 pixels.

(a)* Assuming no limitations on the printer, what would the resolution in line pairs per mm have to be for the image to fit in a space of size 5×5 cm?

(b) What would the resolution have to be in dpi for the image to fit in 2×2 inches?

- 2.6*** A CCD camera chip of dimensions 7×7 mm and 1024×1024 sensing elements, is focused on a square, flat area, located 0.5 m away. The camera is equipped with a 35-mm lens. How many line pairs per mm will this camera be able to resolve? (*Hint:* Model the imaging process as in Fig. 2.3, with the focal length of the camera lens substituting for the focal length of the eye.)

- 2.7** An automobile manufacturer is automating the placement of certain components on the bumpers of a limited-edition line of sports cars. The components are color-coordinated, so the assembly robots need to know the color of each car in order to select the appropriate bumper component. Models come in only four colors: blue, green, red, and white. You are hired to propose a solution based on imaging. How would you solve the problem of determining the color of each car, keeping in mind that cost is the most important consideration in your choice of components?

- 2.8*** Suppose that a given automated imaging application requires a minimum resolution of 5 line pairs per mm to be able to detect features of interest in objects viewed by the camera. The distance between the focal center of the camera lens and the area to be imaged is 1 m. The area being imaged is 0.5×0.5 m. You have available a 200 mm lens, and your job is to pick an appropriate CCD imaging chip. What is the minimum number of sensing elements and square size, $d \times d$, of the CCD chip that will meet the requirements of this application? (*Hint:* Model the imaging process as in Fig. 2.3, and assume for simplicity that the imaged area is square.)

- 2.9** A common measure of transmission for digital data is the *baud rate*, defined as symbols (bits in our case) per second. As a minimum, transmission is accomplished in packets consisting of a start bit, a byte (8 bits) of information, and a stop bit. Using these facts, answer the following:

(a)* How many seconds would it take to transmit a sequence of 500 images of size 1024×1024 pixels with 256 intensity levels using a 3 M-baud (10^6 bits/sec) baud modem? (This is a representative medium speed for a DSL (Digital Subscriber Line) residential line.)

(b) What would the time be using a 30 G-baud (10^9 bits/sec) modem? (This is a representative medium speed for a commercial line.)

- 2.10*** High-definition television (HDTV) generates images with 1125 horizontal TV lines interlaced (i.e., where every other line is “painted” on the screen in each of two fields, each field being $1/60$ th of a second in duration). The width-to-height aspect ratio of the images is 16:9. The fact that the number of horizontal lines is fixed determines the vertical resolution of the images. A company has designed a system that extracts digital images from HDTV video. The resolution of each horizontal line in their system is proportional to vertical resolution of HDTV, with the proportion being the width-to-height ratio of the images. Each pixel in the color image has 24 bits of intensity, 8 bits each for a red, a green, and a blue component image. These three “primary” images form a color image. How many bits would it take to store the images extracted from a two-hour HDTV movie?

- 2.11** When discussing linear indexing in Section 2.4, we arrived at the linear index in Eq. (2-14) by inspection. The same argument used there can be extended to a 3-D array with coordinates x , y , and z , and corresponding dimensions M , N , and P . The linear index for any (x, y, z) is

$$s = x + M(y + Nz)$$

Start with this expression and

(a)* Derive Eq. (2-15).

(b) Derive Eq. (2-16).

- 2.12*** Suppose that a flat area with center at (x_0, y_0) is

illuminated by a light source with intensity distribution

$$i(x, y) = Ke^{-[(x-x_0)^2 + (y-y_0)^2]}$$

Assume for simplicity that the reflectance of the area is constant and equal to 1.0, and let $K = 255$. If the intensity of the resulting image is quantized using k bits, and the eye can detect an abrupt change of eight intensity levels between adjacent pixels, what is the highest value of k that will cause visible false contouring?

2.13 Sketch the image in Problem 2.12 for $k = 2$.

2.14 Consider the two image subsets, S_1 and S_2 in the following figure. With reference to Section 2.5, and assuming that $V = \{1\}$, determine whether these two subsets are:

- (a)* 4-adjacent.
- (b) 8-adjacent.
- (c) m -adjacent.

	S_1				S_2				
0	0	0	0	0	0	0	1	1	0
1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	1	1	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	1	1	1

2.15* Develop an algorithm for converting a one-pixel-thick 8-path to a 4-path.

2.16 Develop an algorithm for converting a one-pixel-thick m -path to a 4-path.

2.17 Refer to the discussion toward the end of Section 2.5, where we defined the background of an image as $(R_u)^c$, the complement of the union of all the regions in the image. In some applications, it is advantageous to define the background as the subset of pixels of $(R_u)^c$ that are not *hole* pixels (informally, think of holes as sets of background pixels surrounded by foreground pixels). How would you modify the definition to exclude hole pixels from $(R_u)^c$? An answer such as “the background is the subset of pixels of $(R_u)^c$ that are not hole pixels” is not acceptable. (*Hint:* Use the concept of connectivity.)

2.18 Consider the image segment shown in the figure that follows.

(a)* As in Section 2.5, let $V = \{0, 1\}$ be the set of intensity values used to define adjacency. Compute the lengths of the shortest 4-, 8-, and m -path between p and q in the following image. If a particular path does not exist between these two points, explain why.

3	1	2	1	(q)
2	2	0	2	
1	2	1	1	
(p)	1	0	1	2

(b) Repeat (a) but using $V = \{1, 2\}$.

2.19 Consider two points p and q .

(a)* State the condition(s) under which the D_4 distance between p and q is equal to the shortest 4-path between these points.

(b) Is this path unique?

2.20 Repeat problem 2.19 for the D_8 distance.

2.21 Consider two *one-dimensional* images f and g of the same size. What has to be true about the orientation of these images for the elementwise and matrix products discussed in Section 2.6 to make sense? Either of the two images can be first in forming the product.

2.22* In the next chapter, we will deal with operators whose function is to compute the sum of pixel values in a small subimage area, S_{xy} , as in Eq. (2-43). Show that these are linear operators.

2.23 Refer to Eq. (2-24) in answering the following:

(a)* Show that image summation is a linear operation.

(b) Show that image subtraction is a linear operation.

(c)* Show that image multiplication is a nonlinear operation.

(d) Show that image division is a nonlinear operation.

2.24 The median, ζ , of a set of numbers is such that

half the values in the set are below ζ and the other half are above it. For example, the median of the set of values $\{2, 3, 8, 20, 21, 25, 31\}$ is 20. Show that an operator that computes the median of a subimage area, S , is nonlinear. (*Hint:* It is sufficient to show that ζ fails the linearity test for a simple numerical example.)

- 2.25*** Show that image averaging can be done recursively. That is, show that if $a(k)$ is the average of k images, then the average of $k+1$ images can be obtained from the already-computed average, $a(k)$, and the new image, f_{k+1} .

- 2.26** With reference to Example 2.5:

- (a)* Prove the validity of Eq. (2-27).
 (b) Prove the validity of Eq. (2-28).

For part (b) you will need the following facts from probability: (1) the variance of a constant times a random variable is equal to the constant squared times the variance of the random variable. (2) The variance of the sum of uncorrelated random variables is equal to the sum of the variances of the individual random variables.

- 2.27** Consider two 8-bit images whose intensity levels span the full range from 0 to 255.

- (a)* Discuss the limiting effect of repeatedly subtracting image (2) from image (1). Assume that the results have to be represented also in eight bits.
 (b) Would reversing the order of the images yield a different result?

- 2.28*** Image subtraction is used often in industrial applications for detecting missing components in product assembly. The approach is to store a “golden” image that corresponds to a correct assembly; this image is then subtracted from incoming images of the same product. *Ideally*, the differences would be zero if the new products are assembled correctly. Difference images for products with missing components would be nonzero in the area where they differ from the golden image. What conditions do you think have to be met in practice for this method to work?

- 2.29** With reference to Eq. (2-32),

- (a)* Give a general formula for the value of K as a function of the number of bits, k , in an

image, such that K results in a scaled image whose intensities span the full k -bit range.

- (b)** Find K for 16- and 32-bit images.

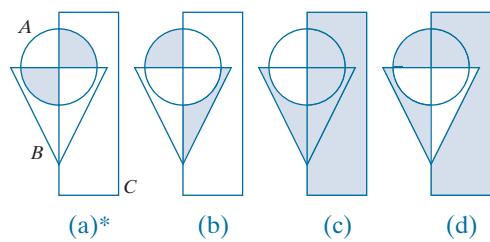
- 2.30** Give Venn diagrams for the following expressions:

- (a)* $(A \cap C) - (A \cap B \cap C)$.
 (b) $(A \cap C) \cup (B \cap C)$.
 (c) $B - [(A \cap B) - (A \cap B \cap C)]$
 (d) $B - B \cap (A \cup C)$; Given that $A \cap C = \emptyset$.

- 2.31** Use Venn diagrams to prove the validity of the following expressions:

- (a)* $(A \cap B) \cup [(A \cap C) - A \cap B \cap C] = A \cap (B \cup C)$
 (b) $(A \cup B \cup C)^c = A^c \cap B^c \cap C^c$
 (c) $(A \cup C)^c \cap B = (B - A) - C$
 (d) $(A \cap B \cap C)^c = A^c \cup B^c \cup C^c$

- 2.32** Give expressions (in terms of sets A , B , and C) for the sets shown shaded in the following figures. The shaded areas in each figure constitute one set, so give only one expression for each of the four figures.



- 2.33** With reference to the discussion on sets in Section 2.6, do the following:

- (a)* Let S be a set of real numbers ordered by the relation “less than or equal to” (\leq). Show that S is a partially ordered set; that is, show that the reflexive, transitive, and antisymmetric properties hold.
 (b)* Show that changing the relation “less than or equal to” to “less than” ($<$) produces a strict ordered set.
 (c) Now let S be the set of lower-case letters in the English alphabet. Show that, under ($<$), S is a strict ordered set.

- 2.34** For any nonzero integers m and n , we say that m

is divisible by n , written m/n , if there exists an integer k such that $kn = m$. For example, 42 (m) is divisible by 7 (n) because there exists an integer $k = 6$ such that $kn = m$. Show that the set of positive integers is a partially ordered set under the relation “divisible by.” In other words, do the following:

- (a)* Show that the property of reflectivity holds under this relation.
- (b) Show that the property of transitivity holds.
- (c) Show that anti symmetry holds.

2.35 In general, what would the resulting image, $g(x,y)$, look like if we modified Eq. (2-43), as follows:

$$g(x,y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} T[f(r,c)]$$

where T is the intensity transformation function in Fig. 2.38(b)?

2.36 With reference to Table 2.3, provide single, composite transformation functions for performing the following operations:

- (a)* Scaling and translation.
- (b)* Scaling, translation, and rotation.
- (c) Vertical shear, scaling, translation, and rotation.
- (d) Does the order of multiplication of the individual matrices to produce a single transformations make a difference? Give an example based on a scaling/translation transformation to support your answer.

2.37 We know from Eq. (2-45) that an affine transformation of coordinates is given by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where (x',y') are the transformed coordinates, (x,y) are the original coordinates, and the elements of \mathbf{A} are given in Table 2.3 for various types of transformations. The inverse transformation, \mathbf{A}^{-1} , to go from the transformed back to the original coordinates is just as important for performing inverse mappings.

(a)* Find the inverse scaling transformation.

(b) Find the inverse translation transformation.

(c) Find the inverse vertical and horizontal shearing transformations.

(d)* Find the inverse rotation transformation.

(e)* Show a composite inverse translation/rotation transformation.

2.38 What are the equations, analogous to Eqs. (2-46) and (2-47), that would result from using triangular instead of quadrilateral regions?

2.39 Do the following.

(a)* Prove that the Fourier kernel in Eq. (2-59) is separable and symmetric.

(b) Repeat (a) for the kernel in Eq. (2-60).

2.40* Show that 2-D transforms with separable, symmetric kernels can be computed by: (1) computing 1-D transforms along the individual rows (columns) of the input image; and (2) computing 1-D transforms along the columns (rows) of the result from step (1).

2.41 A plant produces miniature polymer squares that have to undergo 100% visual inspection. Inspection is semi-automated. At each inspection station, a robot places each polymer square over an optical system that produces a magnified image of the square. The image completely fills a viewing screen of size 80×80 mm. Defects appear as dark circular blobs, and the human inspector's job is to look at the screen and reject any sample that has one or more dark blobs with a diameter of 0.8 mm or greater, as measured on the scale of the screen. The manufacturing manager believes that if she can find a way to fully automate the process, profits will increase by 50%, and success in this project will aid her climb up the corporate ladder. After extensive investigation, the manager decides that the way to solve the problem is to view each inspection screen with a CCD TV camera and feed the output of the camera into an image processing system capable of detecting the blobs, measuring their diameter, and activating the accept/reject button previously operated by a human inspector. She is able to find a suitable system, provided that the smallest defect occupies an area of at least 2×2 pixels in the digital image. The manager hires you to help her specify the camera and lens

system to satisfy this requirement, using off-the-shelf components. Available off-the-shelf lenses have focal lengths that are integer multiples of 25 mm or 35 mm, up to 200 mm. Available cameras yield image sizes of 512×512 , 1024×1024 , or 2048×2048 pixels. The *individual* imaging elements in these cameras are squares measuring $8 \times 8 \mu\text{m}$, and the spaces between imaging elements are $2 \mu\text{m}$. For this application, the cameras

cost much more than the lenses, so you should use the lowest-resolution camera possible, consistent with a suitable lens. As a consultant, you have to provide a written recommendation, showing in reasonable detail the analysis that led to your choice of components. Use the imaging geometry suggested in Problem 2.6.

3

Intensity Transformations and Spatial Filtering

It makes all the difference whether one sees darkness through the light or brightness through the shadows.

David Lindsay

Preview

The term *spatial domain* refers to the image plane itself, and image processing methods in this category are based on direct manipulation of pixels in an image. This is in contrast to image processing in a transform domain which, as we will discuss in Chapters 4 and 6, involves first transforming an image into the transform domain, doing the processing there, and obtaining the inverse transform to bring the results back into the spatial domain. Two principal categories of spatial processing are intensity transformations and spatial filtering. Intensity transformations operate on single pixels of an image for tasks such as contrast manipulation and image thresholding. Spatial filtering performs operations on the neighborhood of every pixel in an image. Examples of spatial filtering include image smoothing and sharpening. In the sections that follow, we discuss a number of “classical” techniques for intensity transformations and spatial filtering.

Upon completion of this chapter, readers should:

- Understand the meaning of spatial domain processing, and how it differs from transform domain processing.
- Be familiar with the principal techniques used for intensity transformations.
- Understand the physical meaning of image histograms and how they can be manipulated for image enhancement.
- Understand the mechanics of spatial filtering, and how spatial filters are formed.
- Understand the principles of spatial convolution and correlation.
- Be familiar with the principal types of spatial filters, and how they are applied.
- Be aware of the relationships between spatial filters, and the fundamental role of lowpass filters.
- Understand how to use combinations of enhancement methods in cases where a single approach is insufficient.

3.1 BACKGROUND

All the image processing techniques discussed in this chapter are implemented in the spatial domain, which we know from the discussion in Section 2.4 is the plane containing the pixels of an image. Spatial domain techniques operate directly on the pixels of an image, as opposed, for example, to the frequency domain (the topic of Chapter 4) in which operations are performed on the Fourier transform of an image, rather than on the image itself. As you will learn in progressing through the book, some image processing tasks are easier or more meaningful to implement in the spatial domain, while others are best suited for other approaches.

THE BASICS OF INTENSITY TRANSFORMATIONS AND SPATIAL FILTERING

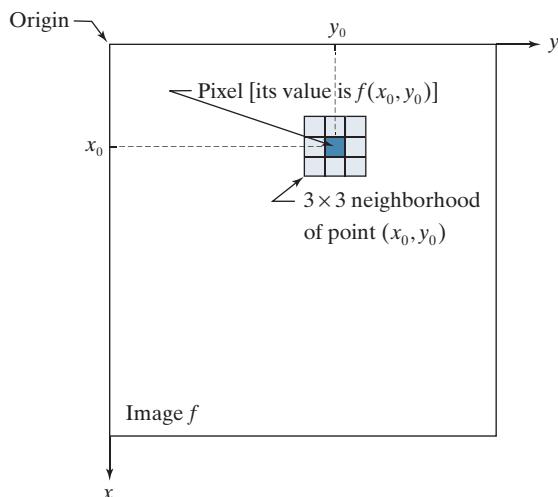
The spatial domain processes we discuss in this chapter are based on the expression

$$g(x, y) = T[f(x, y)] \quad (3-1)$$

where $f(x, y)$ is an input image, $g(x, y)$ is the output image, and T is an operator on f defined over a neighborhood of point (x, y) . The operator can be applied to the pixels of a single image (our principal focus in this chapter) or to the pixels of a set of images, such as performing the elementwise sum of a sequence of images for noise reduction, as discussed in Section 2.6. Figure 3.1 shows the basic implementation of Eq. (3-1) on a single image. The point (x_0, y_0) shown is an arbitrary location in the image, and the small region shown is a *neighborhood* of (x_0, y_0) , as explained in Section 2.6. Typically, the neighborhood is rectangular, centered on (x_0, y_0) , and much smaller in size than the image.

The process that Fig. 3.1 illustrates consists of moving the center of the neighborhood from pixel to pixel, and applying the operator T to the pixels in the neighborhood to yield an output value at that location. Thus, for any specific location (x_0, y_0) ,

FIGURE 3.1
A 3×3 neighborhood about a point (x_0, y_0) in an image. The neighborhood is moved from pixel to pixel in the image to generate an output image. Recall from Chapter 2 that the value of a pixel at location (x_0, y_0) is $f(x_0, y_0)$, the value of the image at that location.



the value of the output image g at those coordinates is equal to the result of applying T to the neighborhood with origin at (x_0, y_0) in f . For example, suppose that the neighborhood is a square of size 3×3 and that operator T is defined as “compute the average intensity of the pixels in the neighborhood.” Consider an arbitrary location in an image, say $(100, 150)$. The result at that location in the output image, $g(100, 150)$, is the sum of $f(100, 150)$ and its 8-neighbors, divided by 9. The center of the neighborhood is then moved to the next adjacent location and the procedure is repeated to generate the next value of the output image g . Typically, the process starts at the top left of the input image and proceeds pixel by pixel in a horizontal (vertical) scan, one row (column) at a time. We will discuss this type of neighborhood processing beginning in Section 3.4.

Depending on the size of a neighborhood and its location, part of the neighborhood may lie outside the image. There are two solutions to this:
 (1) to ignore the values outside the image, or
 (2) to pad image, as discussed in Section 3.4.
 The second approach is preferred.

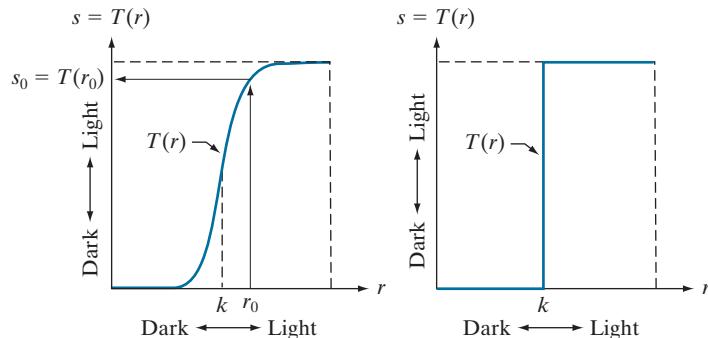
The smallest possible neighborhood is of size 1×1 . In this case, g depends only on the value of f at a single point (x, y) and T in Eq. (3-1) becomes an *intensity* (also called a *gray-level*, or *mapping*) *transformation function* of the form

$$s = T(r) \quad (3-2)$$

where, for simplicity in notation, we use s and r to denote, respectively, the intensity of g and f at any point (x, y) . For example, if $T(r)$ has the form in Fig. 3.2(a), the result of applying the transformation to every pixel in f to generate the corresponding pixels in g would be to produce an image of higher contrast than the original, by darkening the intensity levels below k and brightening the levels above k . In this technique, sometimes called *contrast stretching* (see Section 3.2), values of r lower than k reduce (darken) the values of s , toward black. The opposite is true for values of r higher than k . Observe how an intensity value r_0 is mapped to obtain the corresponding value s_0 . In the limiting case shown in Fig. 3.2(b), $T(r)$ produces a two-level (binary) image. A mapping of this form is called a *thresholding function*. Some fairly simple yet powerful processing approaches can be formulated with intensity transformation functions. In this chapter, we use intensity transformations principally for image enhancement. In Chapter 10, we will use them for image segmentation. Approaches whose results depend only on the intensity at a point sometimes are called *point processing* techniques, as opposed to the *neighborhood processing* techniques discussed in the previous paragraph.

a b

FIGURE 3.2
 Intensity transformation functions.
 (a) Contrast stretching function.
 (b) Thresholding function.



ABOUT THE EXAMPLES IN THIS CHAPTER

Although intensity transformation and spatial filtering methods span a broad range of applications, most of the examples in this chapter are applications to image enhancement. *Enhancement* is the process of manipulating an image so that the result is more suitable than the original for a specific application. The word *specific* is important, because it establishes at the outset that enhancement techniques are problem-oriented. Thus, for example, a method that is quite useful for enhancing X-ray images may not be the best approach for enhancing infrared images. There is no general “theory” of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works. When dealing with machine perception, enhancement is easier to quantify. For example, in an automated character-recognition system, the most appropriate enhancement method is the one that results in the best recognition rate, leaving aside other considerations such as computational requirements of one method versus another. Regardless of the application or method used, image enhancement is one of the most visually appealing areas of image processing. Beginners in image processing generally find enhancement applications interesting and relatively simple to understand. Therefore, using examples from image enhancement to illustrate the spatial processing methods developed in this chapter not only saves having an extra chapter in the book dealing with image enhancement but, more importantly, is an effective approach for introducing newcomers to image processing techniques in the spatial domain. As you progress through the remainder of the book, you will find that the material developed in this chapter has a scope that is much broader than just image enhancement.

3.2 SOME BASIC INTENSITY TRANSFORMATION FUNCTIONS

Intensity transformations are among the simplest of all image processing techniques. As noted in the previous section, we denote the values of pixels, before and after processing, by r and s , respectively. These values are related by a transformation T , as given in Eq. (3-2), that maps a pixel value r into a pixel value s . Because we deal with digital quantities, values of an intensity transformation function typically are stored in a table, and the mappings from r to s are implemented via table lookups. For an 8-bit image, a lookup table containing the values of T will have 256 entries.

As an introduction to intensity transformations, consider Fig. 3.3, which shows three basic types of functions used frequently in image processing: linear (negative and identity transformations), logarithmic (log and inverse-log transformations), and power-law (n th power and n th root transformations). The identity function is the trivial case in which the input and output intensities are identical.

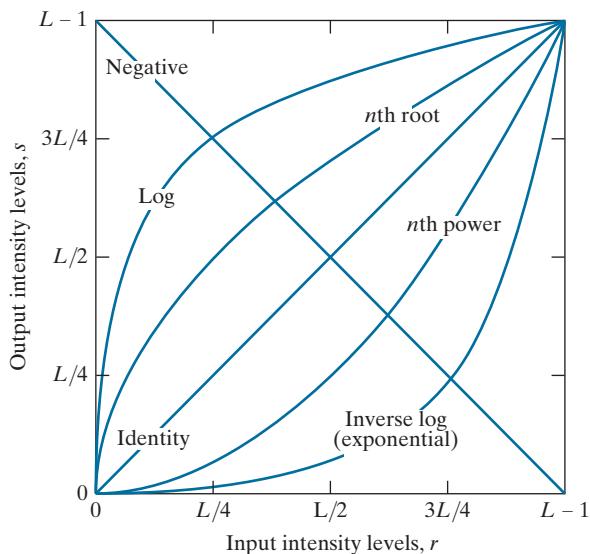
IMAGE NEGATIVES

The negative of an image with intensity levels in the range $[0, L - 1]$ is obtained by using the negative transformation function shown in Fig. 3.3, which has the form:

$$s = L - 1 - r \quad (3-3)$$

FIGURE 3.3

Some basic intensity transformation functions. Each curve was scaled independently so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.

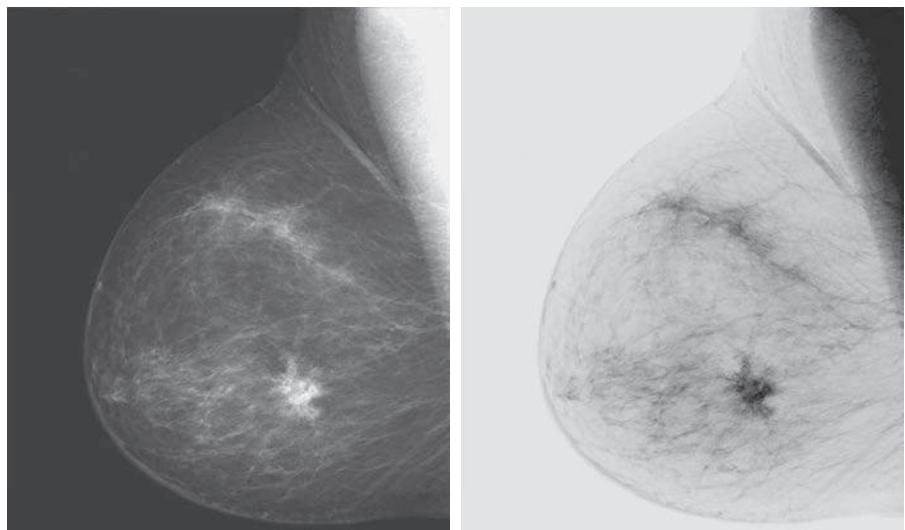


Reversing the intensity levels of a digital image in this manner produces the equivalent of a photographic negative. This type of processing is used, for example, in enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size. Figure 3.4 shows an example. The original image is a digital mammogram showing a small lesion. Despite the fact that the visual content is the same in both images, some viewers find it easier to analyze the fine details of the breast tissue using the negative image.

a b

FIGURE 3.4

(a) A digital mammogram.
 (b) Negative image obtained using Eq. (3-3).
 (Image (a) Courtesy of General Electric Medical Systems.)



LOG TRANSFORMATIONS

The general form of the log transformation in Fig. 3.3 is

$$s = c \log(1 + r) \quad (3-4)$$

where c is a constant and it is assumed that $r \geq 0$. The shape of the log curve in Fig. 3.3 shows that this transformation maps a narrow range of low intensity values in the input into a wider range of output levels. For example, note how input levels in the range $[0, L/4]$ map to output levels to the range $[0, 3L/4]$. Conversely, higher values of input levels are mapped to a narrower range in the output. We use a transformation of this type to expand the values of dark pixels in an image, while compressing the higher-level values. The opposite is true of the inverse log (exponential) transformation.

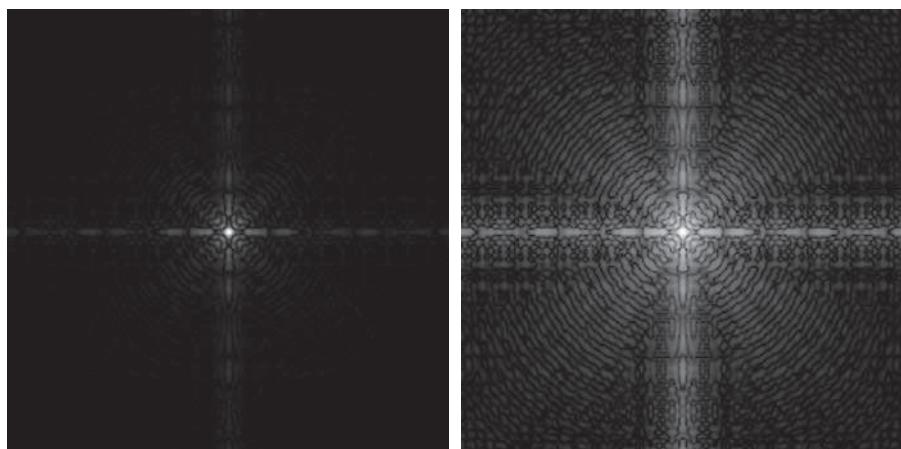
Any curve having the general shape of the log function shown in Fig. 3.3 would accomplish this spreading/compressing of intensity levels in an image, but the power-law transformations discussed in the next section are much more versatile for this purpose. The log function has the important characteristic that it compresses the dynamic range of pixel values. An example in which pixel values have a large dynamic range is the Fourier spectrum, which we will discuss in Chapter 4. It is not unusual to encounter spectrum values that range from 0 to 10^6 or higher. Processing numbers such as these presents no problems for a computer, but image displays cannot reproduce faithfully such a wide range of values. The net effect is that intensity detail can be lost in the display of a typical Fourier spectrum.

Figure 3.5(a) shows a Fourier spectrum with values in the range 0 to 1.5×10^6 . When these values are scaled linearly for display in an 8-bit system, the brightest pixels dominate the display, at the expense of lower (and just as important) values of the spectrum. The effect of this dominance is illustrated vividly by the relatively small area of the image in Fig. 3.5(a) that is not perceived as black. If, instead of displaying the values in this manner, we first apply Eq. (3-4) (with $c = 1$ in this case) to the spectrum values, then the range of values of the result becomes 0 to 6.2. Transforming values in this way enables a greater range of intensities to be shown on the display. Figure 3.5(b) shows the result of scaling the intensity range linearly to the

a b

FIGURE 3.5

- (a) Fourier spectrum displayed as a grayscale image.
- (b) Result of applying the log transformation in Eq. (3-4) with $c = 1$. Both images are scaled to the range [0, 255].



interval [0,255] and showing the spectrum in the same 8-bit display. The level of detail visible in this image as compared to an unmodified display of the spectrum is evident from these two images. Most of the Fourier spectra in image processing publications, including this book, have been scaled in this manner.

POWER-LAW (GAMMA) TRANSFORMATIONS

Power-law transformations have the form

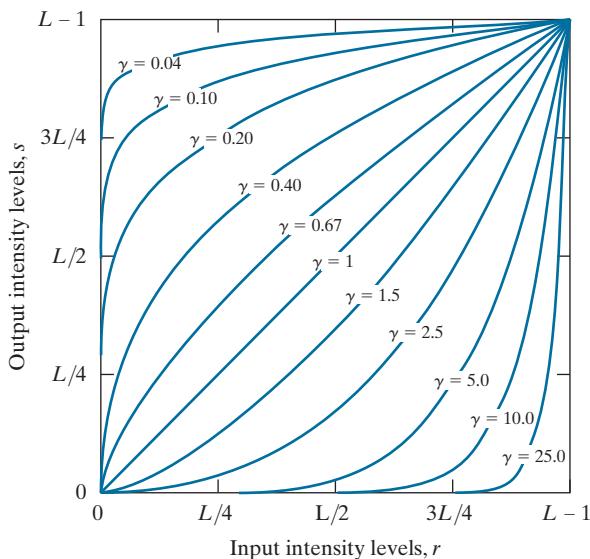
$$s = cr^\gamma \quad (3-5)$$

where c and γ are positive constants. Sometimes Eq. (3-5) is written as $s = c(r + \varepsilon)^\gamma$ to account for offsets (that is, a measurable output when the input is zero). However, offsets typically are an issue of display calibration, and as a result they are normally ignored in Eq. (3-5). Figure 3.6 shows plots of s as a function of r for various values of γ . As with log transformations, power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels. Note also in Fig. 3.6 that a family of transformations can be obtained simply by varying γ . Curves generated with values of $\gamma > 1$ have exactly the opposite effect as those generated with values of $\gamma < 1$. When $c = \gamma = 1$ Eq. (3-5) reduces to the identity transformation.

The response of many devices used for image capture, printing, and display obey a power law. By convention, the exponent in a power-law equation is referred to as *gamma* [hence our use of this symbol in Eq. (3-5)]. The process used to correct these power-law response phenomena is called *gamma correction* or *gamma encoding*. For example, cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with exponents varying from approximately 1.8 to 2.5. As the curve for $\gamma = 2.5$ in Fig. 3.6 shows, such display systems would tend to produce

FIGURE 3.6

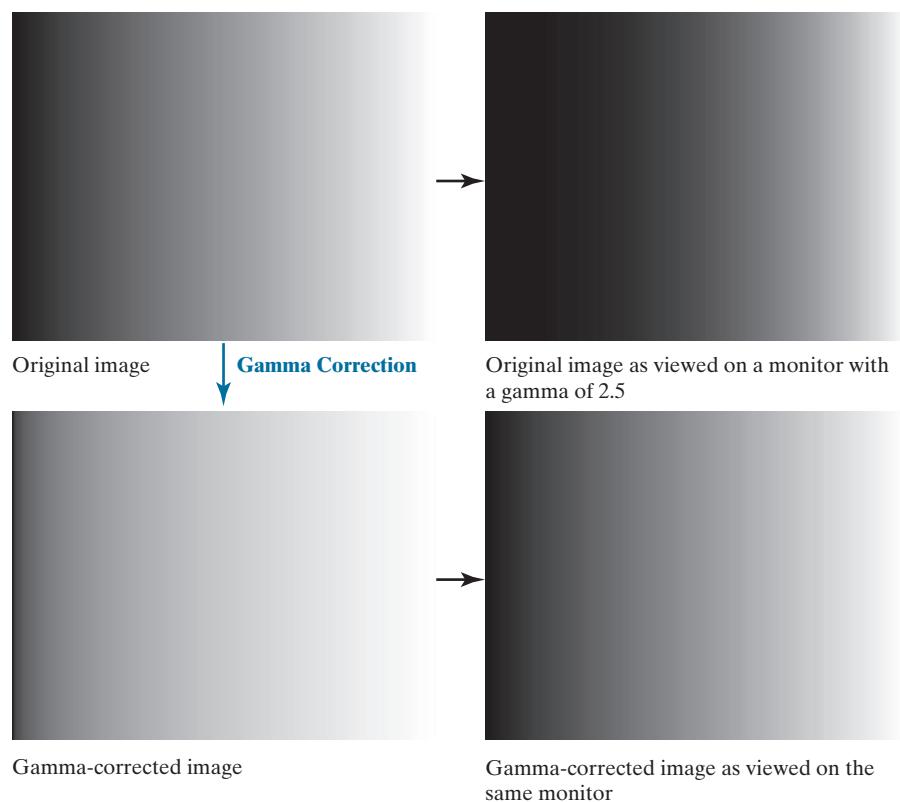
Plots of the gamma equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases). Each curve was scaled independently so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.



a	b
c	d

FIGURE 3.7

(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).



Sometimes, a higher gamma makes the displayed image look better to viewers than the original because of an increase in contrast. However, the objective of gamma correction is to produce a *faithful* display of an input image.

images that are darker than intended. Figure 3.7 illustrates this effect. Figure 3.7(a) is an image of an intensity ramp displayed in a monitor with a gamma of 2.5. As expected, the output of the monitor appears darker than the input, as Fig. 3.7(b) shows.

In this case, gamma correction consists of using the transformation $s = r^{1/2.5} = r^{0.4}$ to preprocess the image before inputting it into the monitor. Figure 3.7(c) is the result. When input into the same monitor, the gamma-corrected image produces an output that is close in appearance to the original image, as Fig. 3.7(d) shows. A similar analysis as above would apply to other imaging devices, such as scanners and printers, the difference being the device-dependent value of gamma (Poynton [1996]).

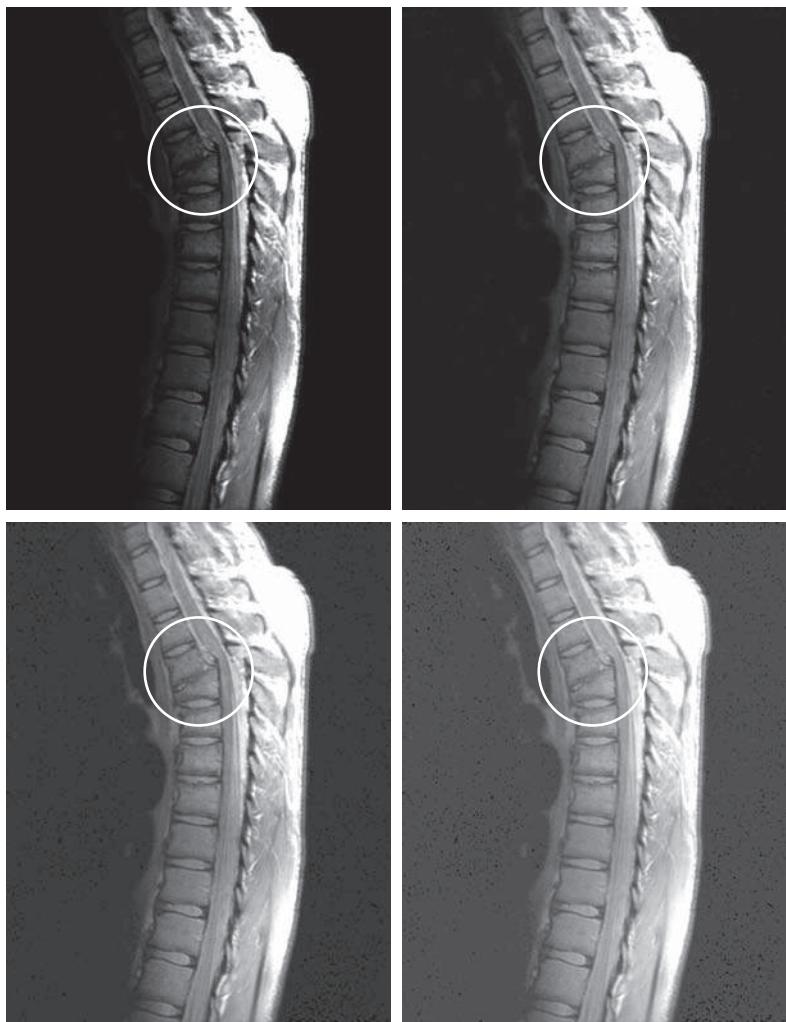
EXAMPLE 3.1: Contrast enhancement using power-law intensity transformations.

In addition to gamma correction, power-law transformations are useful for general-purpose contrast manipulation. Figure 3.8(a) shows a magnetic resonance image (MRI) of a human upper thoracic spine with a fracture dislocation. The fracture is visible in the region highlighted by the circle. Because the image is predominantly dark, an expansion of intensity levels is desirable. This can be accomplished using a power-law transformation with a fractional exponent. The other images shown in the figure were obtained by processing Fig. 3.8(a) with the power-law transformation function of Eq. (3-5). The values

a	b
c	d

FIGURE 3.8

(a) Magnetic resonance image (MRI) of a fractured human spine (the region of the fracture is enclosed by the circle).
 (b)–(d) Results of applying the transformation in Eq. (3-5) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3 , respectively.
 (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)



of gamma corresponding to images (b) through (d) are 0.6, 0.4, and 0.3, respectively ($c = 1$ in all cases). Observe that as gamma decreased from 0.6 to 0.4, more detail became visible. A further decrease of gamma to 0.3 enhanced a little more detail in the background, but began to reduce contrast to the point where the image started to have a very slight “washed-out” appearance, especially in the background. The best enhancement in terms of contrast and discernible detail was obtained with $\gamma = 0.4$. A value of $\gamma = 0.3$ is an approximate limit below which contrast in this particular image would be reduced to an unacceptable level.

EXAMPLE 3.2: Another illustration of power-law transformations.

Figure 3.9(a) shows the opposite problem of that presented in Fig. 3.8(a). The image to be processed

a	b
c	d

FIGURE 3.9

(a) Aerial image.
 (b)–(d) Results
 of applying the
 transformation
 in Eq. (3-5) with
 $\gamma = 3.0, 4.0,$ and
 $5.0,$ respectively.
 ($c = 1$ in all cases.)
 (Original image
 courtesy of
 NASA.)



now has a washed-out appearance, indicating that a compression of intensity levels is desirable. This can be accomplished with Eq. (3-5) using values of γ greater than 1. The results of processing Fig. 3.9(a) with $\gamma = 3.0, 4.0,$ and 5.0 are shown in Figs. 3.9(b) through (d), respectively. Suitable results were obtained using gamma values of 3.0 and 4.0. The latter result has a slightly more appealing appearance because it has higher contrast. This is true also of the result obtained with $\gamma = 5.0.$ For example, the airport runways near the middle of the image appears clearer in Fig. 3.9(d) than in any of the other three images.

PIECEWISE LINEAR TRANSFORMATION FUNCTIONS

An approach complementary to the methods discussed in the previous three sections is to use piecewise linear functions. The advantage of these functions over those discussed thus far is that the form of piecewise functions can be arbitrarily complex. In fact, as you will see shortly, a practical implementation of some important transformations can be formulated only as piecewise linear functions. The main disadvantage of these functions is that their specification requires considerable user input.

Contrast Stretching

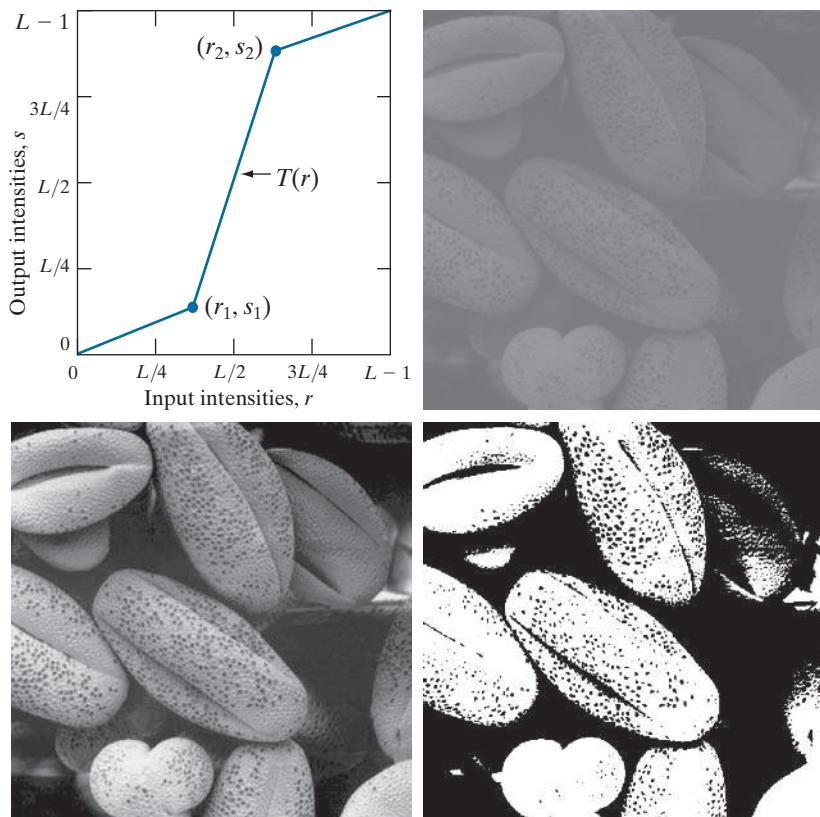
Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even the wrong setting of a lens aperture during image acquisition. *Contrast stretching* expands the range of intensity levels in an image so that it spans the ideal full intensity range of the recording medium or display device.

Figure 3.10(a) shows a typical transformation used for contrast stretching. The locations of points (r_1, s_1) and (r_2, s_2) control the shape of the transformation function. If $r_1 = s_1$ and $r_2 = s_2$ the transformation is a linear function that produces no changes in intensity. If $r_1 = r_2$, $s_1 = 0$, and $s_2 = L - 1$ the transformation becomes a *thresholding function* that creates a binary image [see Fig. 3.2(b)]. Intermediate values of (r_1, s_1) and (s_2, r_2) produce various degrees of spread in the intensity levels of the output image, thus affecting its contrast. In general, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed so that the function is single valued and monotonically increasing. This preserves the order of intensity levels, thus preventing the creation of intensity artifacts. Figure 3.10(b) shows an 8-bit image with low contrast. Figure 3.10(c) shows the result of contrast stretching, obtained by setting $(r_1, s_1) = (r_{\min}, 0)$ and $(r_2, s_2) = (r_{\max}, L - 1)$, where r_{\min} and r_{\max} denote the minimum and maximum intensity levels in the input image,

a
b
c
d

FIGURE 3.10

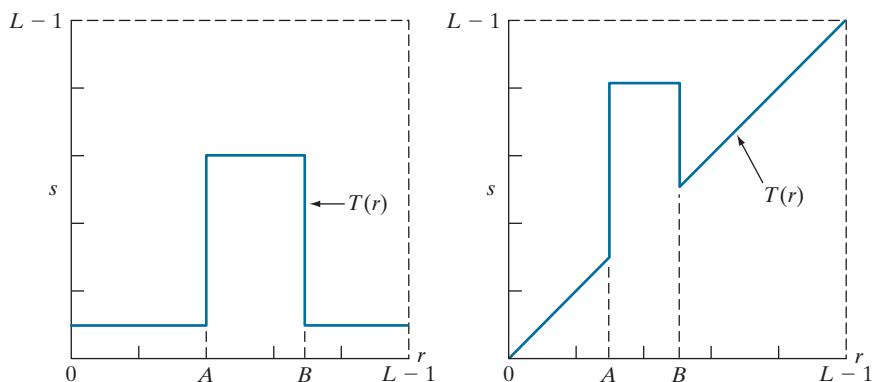
Contrast stretching.
(a) Piecewise linear transformation function. (b) A low-contrast electron microscope image of pollen, magnified 700 times.
(c) Result of contrast stretching.
(d) Result of thresholding.
(Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)



a b

FIGURE 3.11

- (a) This transformation function highlights range $[A, B]$ and reduces all other intensities to a lower level.
 (b) This function highlights range $[A, B]$ and leaves other intensities unchanged.



respectively. The transformation stretched the intensity levels linearly to the full intensity range, $[0, L - 1]$. Finally, Fig. 3.10(d) shows the result of using the thresholding function, with $(r_1, s_1) = (m, 0)$ and $(r_2, s_2) = (m, L - 1)$, where m is the mean intensity level in the image.

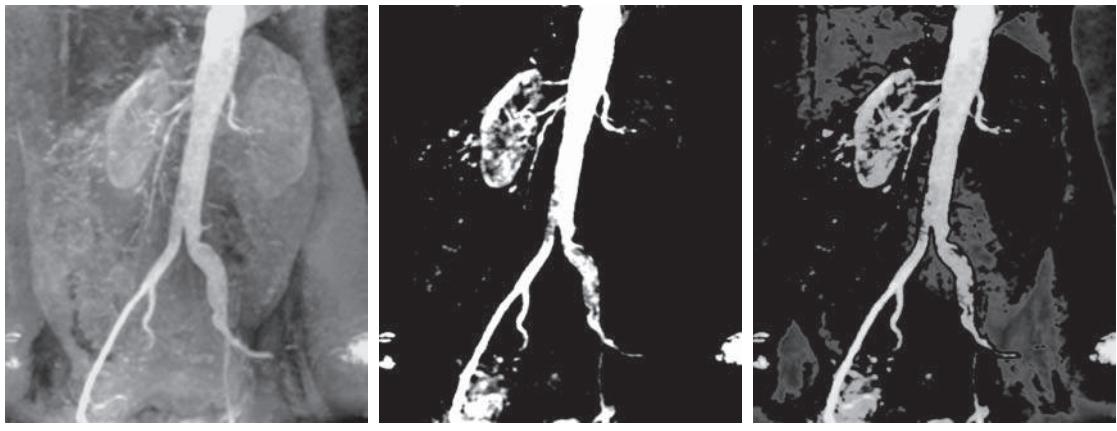
Intensity-Level Slicing

There are applications in which it is of interest to highlight a specific range of intensities in an image. Some of these applications include enhancing features in satellite imagery, such as masses of water, and enhancing flaws in X-ray images. The method, called *intensity-level slicing*, can be implemented in several ways, but most are variations of two basic themes. One approach is to display in one value (say, white) all the values in the range of interest and in another (say, black) all other intensities. This transformation, shown in Fig. 3.11(a), produces a binary image. The second approach, based on the transformation in Fig. 3.11(b), brightens (or darkens) the desired range of intensities, but leaves all other intensity levels in the image unchanged.

EXAMPLE 3.3: Intensity-level slicing.

Figure 3.12(a) is an aortic angiogram near the kidney area (see Section 1.3 for details on this image). The objective of this example is to use intensity-level slicing to enhance the major blood vessels that appear lighter than the background, as a result of an injected contrast medium. Figure 3.12(b) shows the result of using a transformation of the form in Fig. 3.11(a). The selected band was near the top of the intensity scale because the range of interest is brighter than the background. The net result of this transformation is that the blood vessel and parts of the kidneys appear white, while all other intensities are black. This type of enhancement produces a binary image, and is useful for studying the shape characteristics of the flow of the contrast medium (to detect blockages, for example).

If interest lies in the actual intensity values of the region of interest, we can use the transformation of the form shown in Fig. 3.11(b). Figure 3.12(c) shows the result of using such a transformation in which a band of intensities in the mid-gray region around the mean intensity was set to black, while all other intensities were left unchanged. Here, we see that the gray-level tonality of the major blood vessels and part of the kidney area were left intact. Such a result might be useful when interest lies in measuring the actual flow of the contrast medium as a function of time in a sequence of images.



a b c

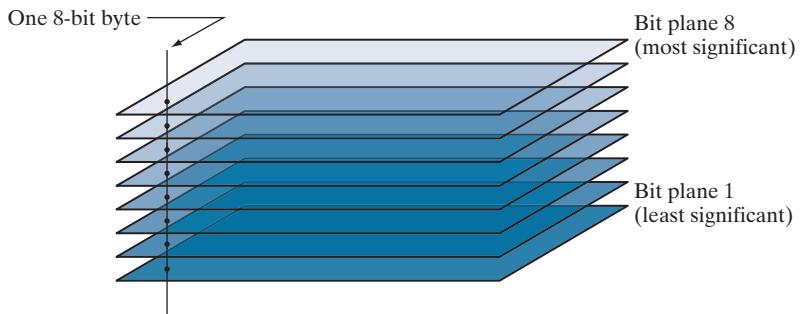
FIGURE 3.12 (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected range set near black, so that the grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)

Bit-Plane Slicing

Pixel values are integers composed of bits. For example, values in a 256-level grayscale image are composed of 8 bits (one byte). Instead of highlighting intensity-level ranges, as 3.3, we could highlight the contribution made to total image appearance by specific bits. As Fig. 3.13 illustrates, an 8-bit image may be considered as being composed of eight one-bit planes, with plane 1 containing the lowest-order bit of all pixels in the image, and plane 8 all the highest-order bits.

Figure 3.14(a) shows an 8-bit grayscale image and Figs. 3.14(b) through (i) are its eight one-bit planes, with Fig. 3.14(b) corresponding to the highest-order bit. Observe that the four higher-order bit planes, especially the first two, contain a significant amount of the visually-significant data. The lower-order planes contribute to more subtle intensity details in the image. The original image has a gray border whose intensity is 194. Notice that the corresponding borders of some of the bit

FIGURE 3.13
Bit-planes of an
8-bit image.



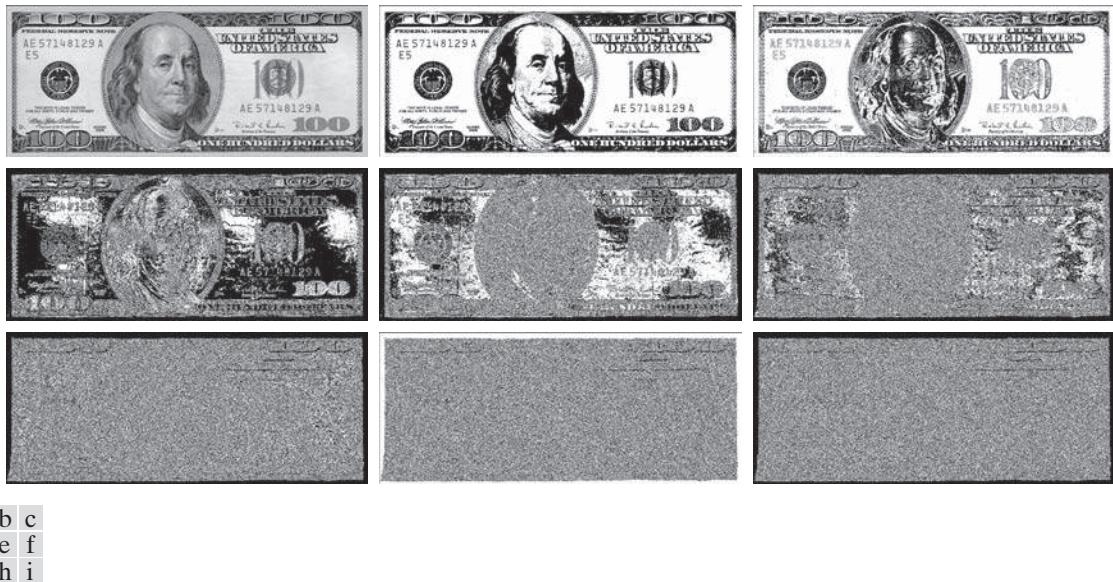


FIGURE 3.14 (a) An 8-bit gray-scale image of size 550×1192 pixels. (b) through (i) Bit planes 8 through 1, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image..

planes are black (0), while others are white (1). To see why, consider a pixel in, say, the middle of the lower border of Fig. 3.14(a). The corresponding pixels in the bit planes, starting with the highest-order plane, have values 1 1 0 0 0 0 1 0, which is the binary representation of decimal 194. The value of any pixel in the original image can be similarly reconstructed from its corresponding binary-valued pixels in the bit planes by converting an 8-bit binary sequence to decimal.

The binary image for the 8th bit plane of an 8-bit image can be obtained by thresholding the input image with a transformation function that maps to 0 intensity values between 0 and 127, and maps to 1 values between 128 and 255. The binary image in Fig. 3.14(b) was obtained in this manner. It is left as an exercise (see Problem 3.3) to obtain the transformation functions for generating the other bit planes.

Decomposing an image into its bit planes is useful for analyzing the relative importance of each bit in the image, a process that aids in determining the adequacy of the number of bits used to quantize the image. Also, this type of decomposition is useful for image compression (the topic of Chapter 8), in which fewer than all planes are used in reconstructing an image. For example, Fig. 3.15(a) shows an image reconstructed using bit planes 8 and 7 of the preceding decomposition. The reconstruction is done by multiplying the pixels of the n th plane by the constant 2^{n-1} . This converts the n th significant binary bit to decimal. Each bit plane is multiplied by the corresponding constant, and all resulting planes are added to obtain the grayscale image. Thus, to obtain Fig. 3.15(a), we multiplied bit plane 8 by 128, bit plane 7 by 64, and added the two planes. Although the main features of the original image were restored, the reconstructed image appears flat, especially in the background. This



a b c FIGURE 3.15 Image reconstructed from bit planes: (a) 8 and 7; (b) 8, 7, and 6; (c) 8, 7, 6, and 5.

is not surprising, because two planes can produce only four distinct intensity levels. Adding plane 6 to the reconstruction helped the situation, as Fig. 3.15(b) shows. Note that the background of this image has perceptible false contouring. This effect is reduced significantly by adding the 5th plane to the reconstruction, as Fig. 3.15(c) illustrates. Using more planes in the reconstruction would not contribute significantly to the appearance of this image. Thus, we conclude that, in this example, storing the four highest-order bit planes would allow us to reconstruct the original image in acceptable detail. Storing these four planes instead of the original image requires 50% less storage.

3.3 HISTOGRAM PROCESSING

Let r_k , for $k = 0, 1, 2, \dots, L - 1$, denote the intensities of an L -level digital image, $f(x, y)$. The *unnormalized histogram* of f is defined as

$$h(r_k) = n_k \quad \text{for } k = 0, 1, 2, \dots, L - 1 \quad (3-6)$$

where n_k is the number of pixels in f with intensity r_k , and the subdivisions of the intensity scale are called *histogram bins*. Similarly, the *normalized histogram* of f is defined as

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN} \quad (3-7)$$

where, as usual, M and N are the number of image rows and columns, respectively. Mostly, we work with normalized histograms, which we refer to simply as *histograms* or *image histograms*. The sum of $p(r_k)$ for all values of k is always 1. The components of $p(r_k)$ are estimates of the probabilities of intensity levels occurring in an image. As you will learn in this section, histogram manipulation is a fundamental tool in image processing. Histograms are simple to compute and are also suitable for fast hardware implementations, thus making histogram-based techniques a popular tool for real-time image processing.

Histogram shape is related to image appearance. For example, Fig. 3.16 shows images with four basic intensity characteristics: dark, light, low contrast, and high contrast; the image histograms are also shown. We note in the dark image that the most populated histogram bins are concentrated on the lower (dark) end of the intensity scale. Similarly, the most populated bins of the light image are biased toward the higher end of the scale. An image with low contrast has a narrow histo-

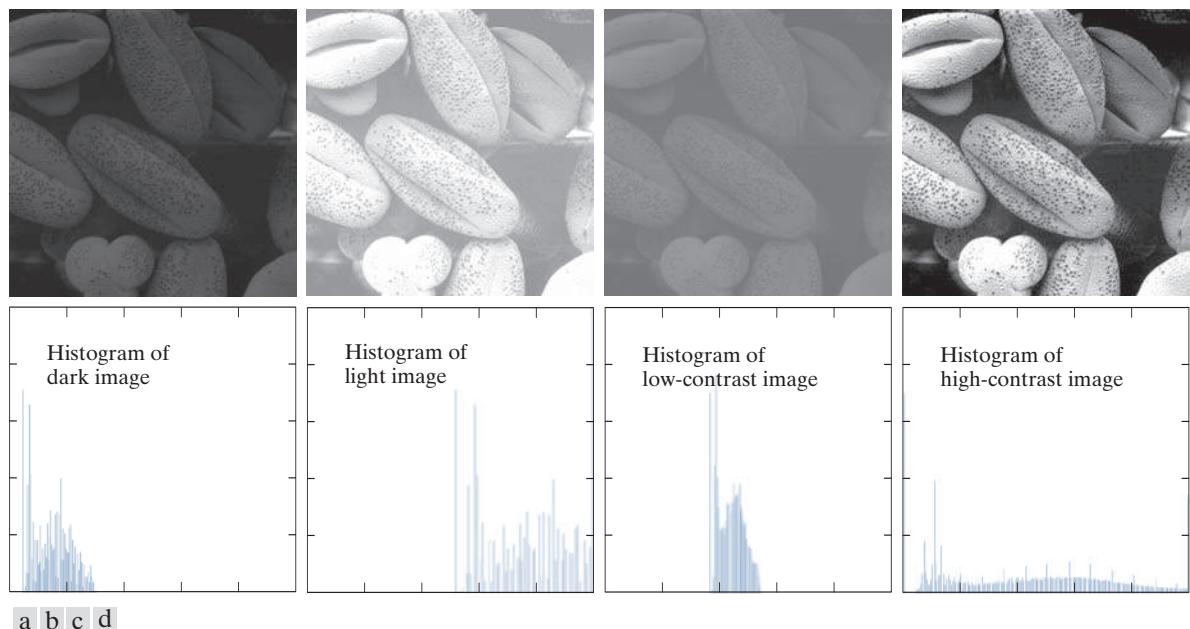


FIGURE 3.16 Four image types and their corresponding histograms. (a) dark; (b) light; (c) low contrast; (d) high contrast. The horizontal axis of the histograms are values of r_k and the vertical axis are values of $p(r_k)$.

gram located typically toward the middle of the intensity scale, as Fig. 3.16(c) shows. For a monochrome image, this implies a dull, washed-out gray look. Finally, we see that the components of the histogram of the high-contrast image cover a wide range of the intensity scale, and the distribution of pixels is not too far from uniform, with few bins being much higher than the others. Intuitively, it is reasonable to conclude that an image whose pixels tend to occupy the entire range of possible intensity levels and, in addition, tend to be distributed uniformly, will have an appearance of high contrast and will exhibit a large variety of gray tones. The net effect will be an image that shows a great deal of gray-level detail and has a high dynamic range. As you will see shortly, it is possible to develop a transformation function that can achieve this effect automatically, using only the histogram of an input image.

HISTOGRAM EQUALIZATION

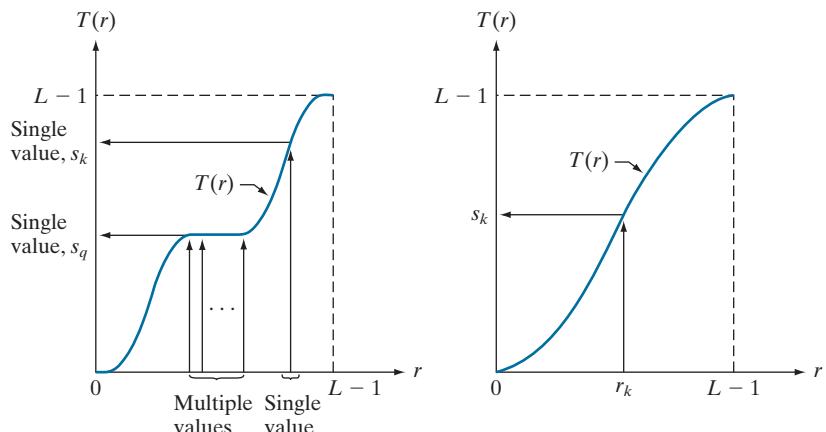
Assuming initially continuous intensity values, let the variable r denote the intensities of an image to be processed. As usual, we assume that r is in the range $[0, L - 1]$, with $r = 0$ representing black and $r = L - 1$ representing white. For r satisfying these conditions, we focus attention on transformations (intensity mappings) of the form

$$s = T(r) \quad 0 \leq r \leq L - 1 \quad (3-8)$$

a b

FIGURE 3.17

(a) Monotonic increasing function, showing how multiple values can map to a single value. (b) Strictly monotonic increasing function. This is a one-to-one mapping, both ways.



that produce an output intensity value, s , for a given intensity value r in the input image. We assume that

- (a) $T(r)$ is a monotonic[†] increasing function in the interval $0 \leq r \leq L - 1$; and
- (b) $0 \leq T(r) \leq L - 1$ for $0 \leq r \leq L - 1$.

In some formulations to be discussed shortly, we use the inverse transformation

$$r = T^{-1}(s) \quad 0 \leq s \leq L - 1 \quad (3-9)$$

in which case we change condition (a) to:

- (a') $T(r)$ is a strictly monotonic increasing function in the interval $0 \leq r \leq L - 1$.

The condition in (a) that $T(r)$ be monotonically increasing guarantees that output intensity values will never be less than corresponding input values, thus preventing artifacts created by reversals of intensity. Condition (b) guarantees that the range of output intensities is the same as the input. Finally, condition (a') guarantees that the mappings from s back to r will be one-to-one, thus preventing ambiguities.

Figure 3.17(a) shows a function that satisfies conditions (a) and (b). Here, we see that it is possible for multiple input values to map to a single output value and still satisfy these two conditions. That is, a monotonic transformation function performs a one-to-one or many-to-one mapping. This is perfectly fine when mapping from r to s . However, Fig. 3.17(a) presents a problem if we wanted to recover the values of r uniquely from the mapped values (inverse mapping can be visualized by reversing the direction of the arrows). This would be possible for the inverse mapping of s_k in Fig. 3.17(a), but the inverse mapping of s_q is a range of values, which, of course, prevents us in general from recovering the original value of r that resulted

[†]A function $T(r)$ is a *monotonic increasing* function if $T(r_2) \geq T(r_1)$ for $r_2 > r_1$. $T(r)$ is a *strictly monotonic increasing* function if $T(r_2) > T(r_1)$ for $r_2 > r_1$. Similar definitions apply to a monotonic decreasing function.

in s_q . As Fig. 3.17(b) shows, requiring that $T(r)$ be strictly monotonic guarantees that the inverse mappings will be *single valued* (i.e., the mapping is one-to-one in both directions). This is a theoretical requirement that will allow us to derive some important histogram processing techniques later in this chapter. Because images are stored using integer intensity values, we are forced to round all results to their nearest integer values. This often results in strict monotonicity not being satisfied, which implies inverse transformations that may not be unique. Fortunately, this problem is not difficult to handle in the discrete case, as Example 3.7 in this section illustrates.

The intensity of an image may be viewed as a random variable in the interval $[0, L - 1]$. Let $p_r(r)$ and $p_s(s)$ denote the PDFs of intensity values r and s in two different images. The subscripts on p indicate that p_r and p_s are different functions. A fundamental result from probability theory is that if $p_r(r)$ and $T(r)$ are known, and $T(r)$ is continuous and differentiable over the range of values of interest, then the PDF of the transformed (mapped) variable s can be obtained as

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| \quad (3-10)$$

Thus, we see that the PDF of the output intensity variable, s , is determined by the PDF of the input intensities and the transformation function used [recall that r and s are related by $T(r)$].

A transformation function of particular importance in image processing is

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw \quad (3-11)$$

where w is a dummy variable of integration. The integral on the right side is the *cumulative distribution function* (CDF) of random variable r . Because PDFs always are positive, and the integral of a function is the area under the function, it follows that the transformation function of Eq. (3-11) satisfies condition (a). This is because the area under the function cannot decrease as r increases. When the upper limit in this equation is $r = (L - 1)$ the integral evaluates to 1, as it must for a PDF. Thus, the maximum value of s is $L - 1$, and condition (b) is satisfied also.

We use Eq. (3-10) to find the $p_s(s)$ corresponding to the transformation just discussed. We know from Leibniz's rule in calculus that the derivative of a definite integral with respect to its upper limit is the integrand evaluated at the limit. That is,

$$\begin{aligned} \frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= (L - 1) \frac{d}{dr} \left[\int_0^r p_r(w) dw \right] \\ &= (L - 1)p_r(r) \end{aligned} \quad (3-12)$$

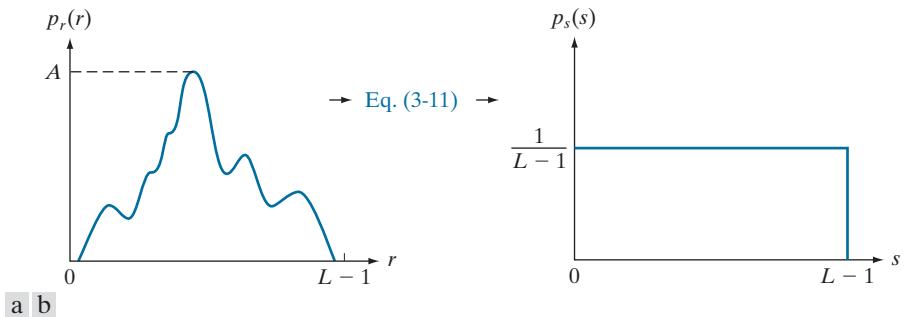


FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying Eq. (3-11) to the input PDF. The resulting PDF is always uniform, independently of the shape of the input.

Substituting this result for dr/ds in Eq. (3-10), and noting that all probability values are positive, gives the result

$$\begin{aligned}
 p_s(s) &= p_r(r) \left| \frac{dr}{ds} \right| \\
 &= p_r(r) \left| \frac{1}{(L-1)p_r(r)} \right| \\
 &= \frac{1}{L-1} \quad 0 \leq s \leq L-1
 \end{aligned} \tag{3-13}$$

We recognize the form of $p_s(s)$ in the last line of this equation as a *uniform* probability density function. Thus, performing the intensity transformation in Eq. (3-11) yields a random variable, s , characterized by a uniform PDF. What is important is that $p_s(s)$ in Eq. (3-13) will *always* be uniform, *independently* of the form of $p_r(r)$. Figure 3.18 and the following example illustrate these concepts.

EXAMPLE 3.4: Illustration of Eqs. (3-11) and (3-13).

Suppose that the (continuous) intensity values in an image have the PDF

$$p_r(r) = \begin{cases} \frac{2r}{(L-1)^2} & \text{for } 0 \leq r \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

From Eq. (3-11)

$$s = T(r) = (L-1) \int_0^r p_r(w) dw = \frac{2}{L-1} \int_0^r w dw = \frac{r^2}{L-1}$$

Suppose that we form a new image with intensities, s , obtained using this transformation; that is, the s values are formed by squaring the corresponding intensity values of the input image, then dividing them by $L - 1$. We can verify that the PDF of the intensities in the new image, $p_s(s)$, is uniform by substituting $p_r(r)$ into Eq. (3-13), and using the fact that $s = r^2/(L - 1)$; that is,

$$\begin{aligned} p_s(s) &= p_r(r) \left| \frac{dr}{ds} \right| = \frac{2r}{(L-1)^2} \left| \left[\frac{ds}{dr} \right]^{-1} \right| \\ &= \frac{2r}{(L-1)^2} \left| \left[\frac{d}{dr} \frac{r^2}{L-1} \right]^{-1} \right| = \frac{2r}{(L-1)^2} \left| \frac{(L-1)}{2r} \right| = \frac{1}{L-1} \end{aligned}$$

The last step follows because r is nonnegative and $L > 1$. As expected, the result is a uniform PDF.

For discrete values, we work with probabilities and summations instead of probability density functions and integrals (but the requirement of monotonicity stated earlier still applies). Recall that the probability of occurrence of intensity level r_k in a digital image is approximated by

$$p_r(r_k) = \frac{n_k}{MN} \quad (3-14)$$

where MN is the total number of pixels in the image, and n_k denotes the number of pixels that have intensity r_k . As noted in the beginning of this section, $p_r(r_k)$, with $r_k \in [0, L - 1]$, is commonly referred to as a normalized image histogram.

The discrete form of the transformation in Eq. (3-11) is

$$s_k = T(r_k) = (L-1) \sum_{j=0}^k p_r(r_j) \quad k = 0, 1, 2, \dots, L-1 \quad (3-15)$$

where, as before, L is the number of possible intensity levels in the image (e.g., 256 for an 8-bit image). Thus, a processed (output) image is obtained by using Eq. (3-15) to map each pixel in the input image with intensity r_k into a corresponding pixel with level s_k in the output image. This is called a *histogram equalization* or *histogram linearization* transformation. It is not difficult to show (see Problem 3.9) that this transformation satisfies conditions (a) and (b) stated previously in this section.

EXAMPLE 3.5: Illustration of the mechanics of histogram equalization.

It will be helpful to work through a simple example. Suppose that a 3-bit image ($L = 8$) of size 64×64 pixels ($MN = 4096$) has the intensity distribution in Table 3.1, where the intensity levels are integers in the range $[0, L - 1] = [0, 7]$. The histogram of this image is sketched in Fig. 3.19(a). Values of the histogram equalization transformation function are obtained using Eq. (3-15). For instance,

$$s_0 = T(r_0) = 7 \sum_{j=0}^0 p_r(r_j) = 7 p_r(r_0) = 1.33$$

TABLE 3.1

Intensity distribution and histogram values for a 3-bit, 64×64 digital image.

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

Similarly, $s_1 = T(r_1) = 3.08$, $s_2 = 4.55$, $s_3 = 5.67$, $s_4 = 6.23$, $s_5 = 6.65$, $s_6 = 6.86$, and $s_7 = 7.00$. This transformation function has the staircase shape shown in Fig. 3.19(b).

At this point, the s values are fractional because they were generated by summing probability values, so we round them to their nearest integer values in the range $[0, 7]$:

$$\begin{array}{llll} s_0 = 1.33 \rightarrow 1 & s_2 = 4.55 \rightarrow 5 & s_4 = 6.23 \rightarrow 6 & s_6 = 6.86 \rightarrow 7 \\ s_1 = 3.08 \rightarrow 3 & s_3 = 5.67 \rightarrow 6 & s_5 = 6.65 \rightarrow 7 & s_7 = 7.00 \rightarrow 7 \end{array}$$

These are the values of the equalized histogram. Observe that the transformation yielded only five distinct intensity levels. Because $r_0 = 0$ was mapped to $s_0 = 1$, there are 790 pixels in the histogram equalized image with this value (see Table 3.1). Also, there are 1023 pixels with a value of $s_1 = 3$ and 850 pixels with a value of $s_2 = 5$. However, both r_3 and r_4 were mapped to the same value, 6, so there are $(656 + 329) = 985$ pixels in the equalized image with this value. Similarly, there are $(245 + 122 + 81) = 448$ pixels with a value of 7 in the histogram equalized image. Dividing these numbers by $MN = 4096$ yielded the equalized histogram in Fig. 3.19(c).

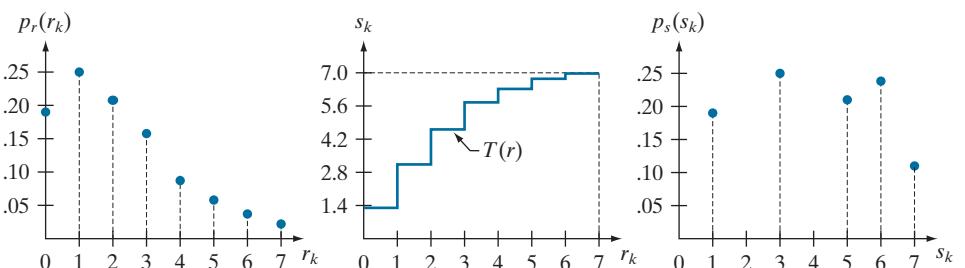
Because a histogram is an approximation to a PDF, and no new allowed intensity levels are created in the process, perfectly flat histograms are rare in practical applications of histogram equalization using the method just discussed. Thus, unlike its continuous counterpart, it cannot be proved in general that discrete histogram equalization using Eq. (3-15) results in a uniform histogram (we will introduce later in

a b c

FIGURE 3.19

Histogram equalization.

- (a) Original histogram.
- (b) Transformation function.
- (c) Equalized histogram.



this section an approach for removing this limitation). However, as you will see shortly, using Eq. (3-15) has the general tendency to spread the histogram of the input image so that the intensity levels of the equalized image span a wider range of the intensity scale. The net result is contrast enhancement.

We discussed earlier the advantages of having intensity values that span the entire gray scale. The method just derived produces intensities that have this tendency, and also has the advantage that it is fully automatic. In other words, the process of histogram equalization consists entirely of implementing Eq. (3-15), which is based on information that can be extracted directly from a given image, without the need for any parameter specifications. This automatic, “hands-off” characteristic is important.

The inverse transformation from s back to r is denoted by

$$r_k = T^{-1}(s_k) \quad (3-16)$$

It can be shown (see Problem 3.9) that this inverse transformation satisfies conditions (a') and (b) defined earlier *only if all* intensity levels are present in the input image. This implies that none of the bins of the image histogram are empty. Although the inverse transformation is not used in histogram equalization, it plays a central role in the histogram-matching scheme developed after the following example.

EXAMPLE 3.6: Histogram equalization.

The left column in Fig. 3.20 shows the four images from Fig. 3.16, and the center column shows the result of performing histogram equalization on each of these images. The first three results from top to bottom show significant improvement. As expected, histogram equalization did not have much effect on the fourth image because its intensities span almost the full scale already. Figure 3.21 shows the transformation functions used to generate the equalized images in Fig. 3.20. These functions were generated using Eq. (3-15). Observe that transformation (4) is nearly linear, indicating that the inputs were mapped to nearly equal outputs. Shown is the mapping of an input value r_k to a corresponding output value s_k . In this case, the mapping was for image 1 (on the top left of Fig. 3.21), and indicates that a dark value was mapped to a much lighter one, thus contributing to the brightness of the output image.

The third column in Fig. 3.20 shows the histograms of the equalized images. While all the histograms are different, the histogram-equalized images themselves are visually very similar. This is not totally unexpected because the basic difference between the images on the left column is one of contrast, not content. Because the images have the same content, the increase in contrast resulting from histogram equalization was enough to render any intensity differences between the equalized images visually indistinguishable. Given the significant range of contrast differences in the original images, this example illustrates the power of histogram equalization as an adaptive, autonomous contrast-enhancement tool.

HISTOGRAM MATCHING (SPECIFICATION)

As explained in the last section, histogram equalization produces a transformation function that seeks to generate an output image with a uniform histogram. When automatic enhancement is desired, this is a good approach to consider because the

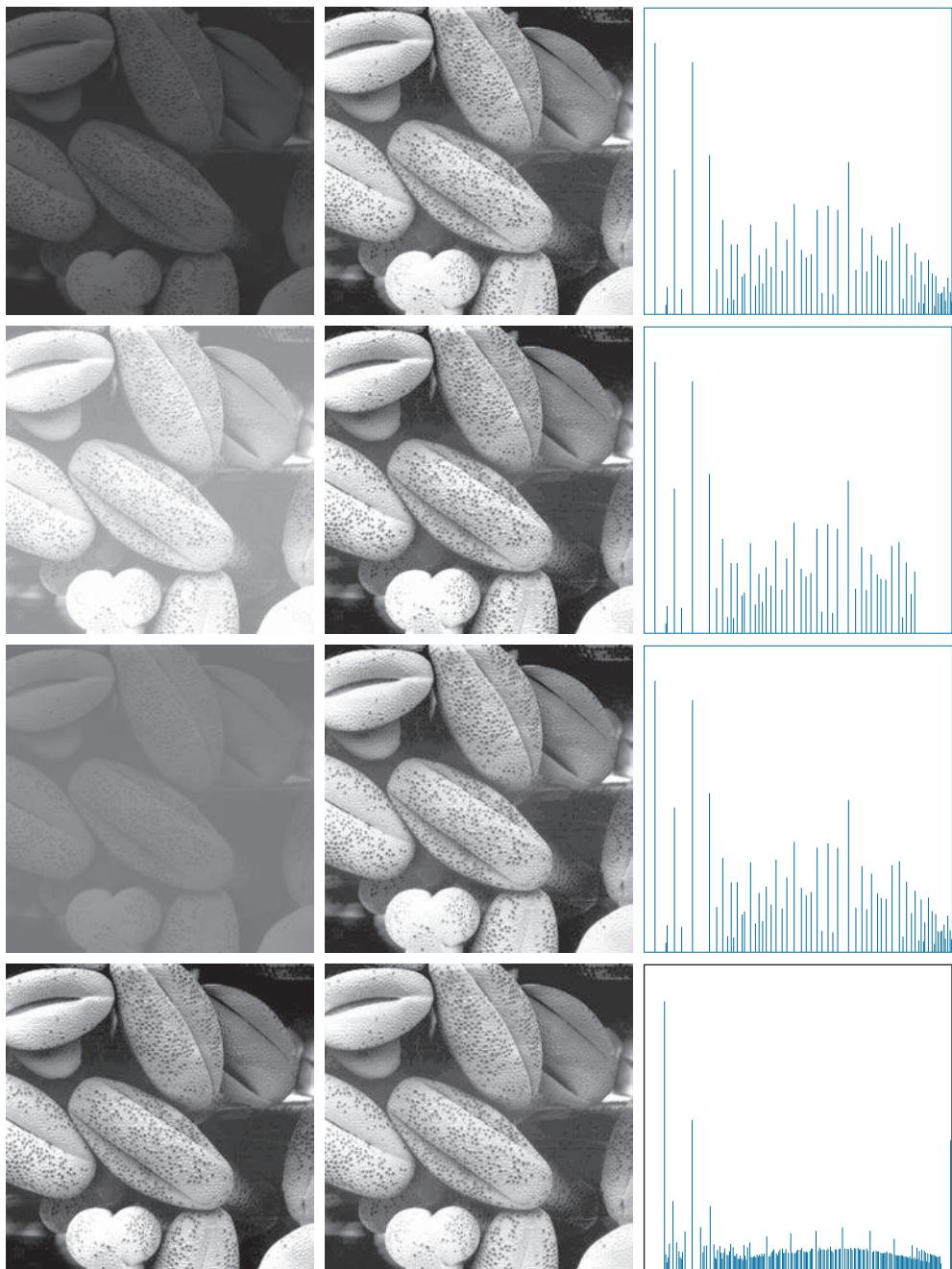
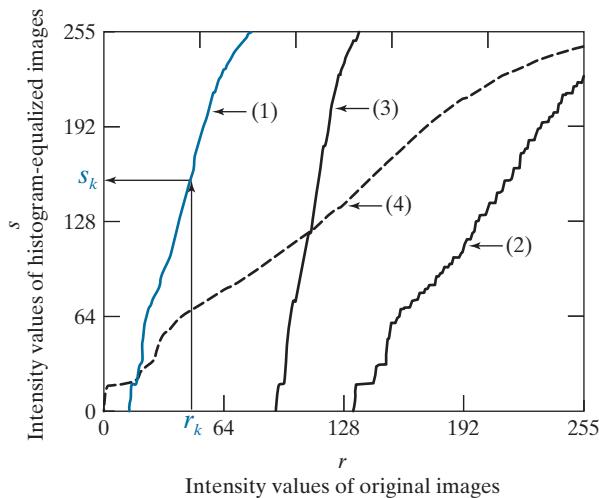


FIGURE 3.20 Left column: Images from Fig. 3.16. Center column: Corresponding histogram-equalized images. Right column: histograms of the images in the center column (compare with the histograms in Fig. 3.16).

FIGURE 3.21

Transformation functions for histogram equalization. Transformations (1) through (4) were obtained using Eq. (3-15) and the histograms of the images on the left column of Fig. 3.20. Mapping of one intensity value r_k in image 1 to its corresponding value s_k is shown.



results from this technique are predictable and the method is simple to implement. However, there are applications in which histogram equalization is not suitable. In particular, it is useful sometimes to be able to specify the shape of the histogram that we wish the processed image to have. The method used to generate images that have a specified histogram is called *histogram matching* or *histogram specification*.

Consider for a moment continuous intensities r and z which, as before, we treat as random variables with PDFs $p_r(r)$ and $p_z(z)$, respectively. Here, r and z denote the intensity levels of the input and output (processed) images, respectively. We can estimate $p_r(r)$ from the given input image, and $p_z(z)$ is the *specified* PDF that we wish the output image to have.

Let s be a random variable with the property

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw \quad (3-17)$$

where w is dummy variable of integration. This is the same as Eq. (3-11), which we repeat here for convenience.

Define a function G on variable z with the property

$$G(z) = (L - 1) \int_0^z p_z(v) dv = s \quad (3-18)$$

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

$$z = G^{-1}(s) = G^{-1}[T(r)] \quad (3-19)$$

The transformation function $T(r)$ can be obtained using Eq. (3-17) after $p_r(r)$ has been estimated using the input image. Similarly, function $G(z)$ can be obtained from Eq. (3-18) because $p_z(z)$ is given.

Equations (3-17) through (3-19) imply that an image whose intensity levels have a specified PDF can be obtained using the following procedure:

1. Obtain $p_r(r)$ from the input image to use in Eq. (3-17).
2. Use the specified PDF, $p_z(z)$, in Eq. (3-18) to obtain the function $G(z)$.
3. Compute the inverse transformation $z = G^{-1}(s)$; this is a mapping from s to z , the latter being the values that have the specified PDF.
4. Obtain the output image by first equalizing the input image using Eq. (3-17); the pixel values in this image are the s values. For each pixel with value s in the equalized image, perform the inverse mapping $z = G^{-1}(s)$ to obtain the corresponding pixel in the output image. When all pixels have been processed with this transformation, the PDF of the output image, $p_z(z)$, will be equal to the specified PDF.

Because s is related to r by $T(r)$, it is possible for the mapping that yields z from s to be expressed directly in terms of r . In general, however, finding analytical expressions for G^{-1} is not a trivial task. Fortunately, this is not a problem when working with discrete quantities, as you will see shortly.

As before, we have to convert the continuous result just derived into a discrete form. This means that we work with histograms instead of PDFs. As in histogram equalization, we lose in the conversion the ability to be able to guarantee a result that will have the exact specified histogram. Despite this, some very useful results can be obtained even with approximations.

The discrete formulation of Eq. (3-17) is the histogram equalization transformation in Eq. (3-15), which we repeat here for convenience:

$$s_k = T(r_k) = (L-1) \sum_{j=0}^k p_r(r_j) \quad k = 0, 1, 2, \dots, L-1 \quad (3-20)$$

where the components of this equation are as before. Similarly, given a specific value of s_k , the discrete formulation of Eq. (3-18) involves computing the transformation function

$$G(z_q) = (L-1) \sum_{i=0}^q p_z(z_i) \quad (3-21)$$

for a value of q so that

$$G(z_q) = s_k \quad (3-22)$$

where $p_z(z_i)$ is the i th value of the specified histogram. Finally, we obtain the desired value z_q from the inverse transformation:

$$z_q = G^{-1}(s_k) \quad (3-23)$$

When performed over all pixels, this is a mapping from the s values in the histogram-equalized image to the corresponding z values in the output image.

In practice, there is no need to compute the inverse of G . Because we deal with intensity levels that are integers, it is a simple matter to compute all the possible values of G using Eq. (3-21) for $q = 0, 1, 2, \dots, L - 1$. These values are rounded to their nearest integer values spanning the range $[0, L - 1]$ and stored in a lookup table. Then, given a particular value of s_k , we look for the closest match in the table. For example, if the 27th entry in the table is the closest value to s_k , then $q = 26$ (recall that we start counting intensities at 0) and z_{26} is the best solution to Eq. (3-23). Thus, the given value s_k would map to z_{26} . Because the z 's are integers in the range $[0, L - 1]$, it follows that $z_0 = 0$, $z_{L-1} = L - 1$, and, in general, $z_q = q$. Therefore, z_{26} would equal intensity value 26. We repeat this procedure to find the mapping from each value s_k to the value z_q that is its closest match in the table. These mappings are the solution to the histogram-specification problem.

Given an input image, a specified histogram, $p_z(z_i)$, $i = 0, 1, 2, \dots, L - 1$, and recalling that the s_k 's are the values resulting from Eq. (3-20), we may summarize the procedure for discrete histogram specification as follows:

1. Compute the histogram, $p_r(r)$, of the input image, and use it in Eq. (3-20) to map the intensities in the input image to the intensities in the histogram-equalized image. Round the resulting values, s_k , to the integer range $[0, L - 1]$.
2. Compute all values of function $G(z_q)$ using the Eq. (3-21) for $q = 0, 1, 2, \dots, L - 1$, where $p_z(z_i)$ are the values of the specified histogram. Round the values of G to integers in the range $[0, L - 1]$. Store the rounded values of G in a lookup table.
3. For every value of s_k , $k = 0, 1, 2, \dots, L - 1$, use the stored values of G from Step 2 to find the corresponding value of z_q so that $G(z_q)$ is closest to s_k . Store these mappings from s to z . When more than one value of z_q gives the same match (i.e., the mapping is not unique), choose the smallest value by convention.
4. Form the histogram-specified image by mapping every equalized pixel with value s_k to the corresponding pixel with value z_q in the histogram-specified image, using the mappings found in Step 3.

As in the continuous case, the intermediate step of equalizing the input image is conceptual. It can be skipped by combining the two transformation functions, T and G^{-1} , as Example 3.7 below shows.

We mentioned at the beginning of the discussion on histogram equalization that, in addition to condition (b), inverse functions (G^{-1} in the present discussion) have to be strictly monotonic to satisfy condition (a'). In terms of Eq. (3-21), this means that none of the values $p_z(z_i)$ in the specified histogram can be zero (see Problem 3.9). When this condition is not satisfied, we use the “work-around” procedure in Step 3. The following example illustrates this numerically.

EXAMPLE 3.7: Illustration of the mechanics of histogram specification.

Consider the 64×64 hypothetical image from Example 3.5, whose histogram is repeated in Fig. 3.22(a). It is desired to transform this histogram so that it will have the values specified in the second column of Table 3.2. Figure 3.22(b) shows this histogram.

The first step is to obtain the histogram-equalized values, which we did in Example 3.5:

$$s_0 = 1; s_1 = 3; s_2 = 5; s_3 = 6; s_4 = 6; s_5 = 7; s_6 = 7; s_7 = 7$$

In the next step, we compute the values of $G(z_q)$ using the values of $p_z(z_q)$ from Table 3.2 in Eq. (3-21):

$$\begin{aligned} G(z_0) &= 0.00 & G(z_2) &= 0.00 & G(z_4) &= 2.45 & G(z_6) &= 5.95 \\ G(z_1) &= 0.00 & G(z_3) &= 1.05 & G(z_5) &= 4.55 & G(z_7) &= 7.00 \end{aligned}$$

As in Example 3.5, these fractional values are rounded to integers in the range $[0, 7]$:

$$\begin{array}{ll} G(z_0) = 0.00 \rightarrow 0 & G(z_4) = 2.45 \rightarrow 2 \\ G(z_1) = 0.00 \rightarrow 0 & G(z_5) = 4.55 \rightarrow 5 \\ G(z_2) = 0.00 \rightarrow 0 & G(z_6) = 5.95 \rightarrow 6 \\ G(z_3) = 1.05 \rightarrow 1 & G(z_7) = 7.00 \rightarrow 7 \end{array}$$

These results are summarized in Table 3.3. The transformation function, $G(z_q)$, is sketched in Fig. 3.23(c). Because its first three values are equal, G is not strictly monotonic, so condition (a') is violated. Therefore, we use the approach outlined in Step 3 of the algorithm to handle this situation. According to this step, we find the smallest value of z_q so that the value $G(z_q)$ is the closest to s_k . We do this for every value of

a	b
c	d

FIGURE 3.22

- (a) Histogram of a 3-bit image.
- (b) Specified histogram.
- (c) Transformation function obtained from the specified histogram.
- (d) Result of histogram specification. Compare the histograms in (b) and (d).

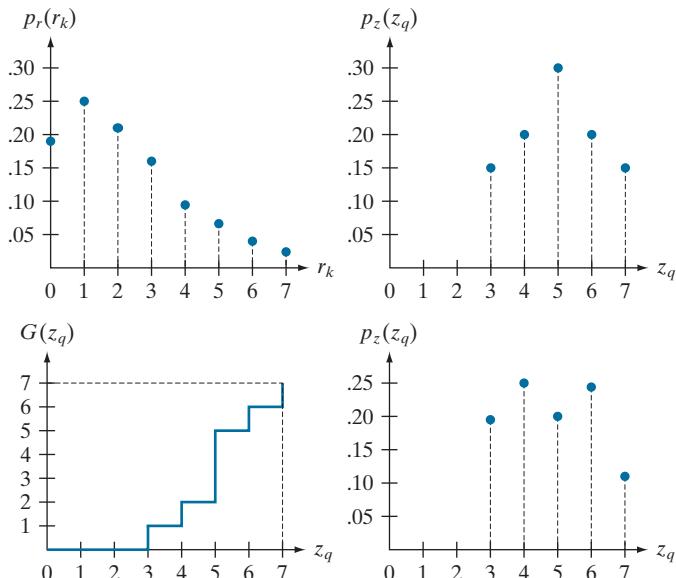


TABLE 3.2

Specified and actual histograms (the values in the third column are computed in Example 3.7).

z_q	Specified $p_z(z_q)$	Actual $p_z(z_q)$
$z_0 = 0$	0.00	0.00
$z_1 = 1$	0.00	0.00
$z_2 = 2$	0.00	0.00
$z_3 = 3$	0.15	0.19
$z_4 = 4$	0.20	0.25
$z_5 = 5$	0.30	0.21
$z_6 = 6$	0.20	0.24
$z_7 = 7$	0.15	0.11

s_k to create the required mappings from s to z . For example, $s_0 = 1$, and we see that $G(z_3) = 1$, which is a perfect match in this case, so we have the correspondence $s_0 \rightarrow z_3$. Every pixel whose value is 1 in the histogram equalized image would map to a pixel valued 3 in the histogram-specified image. Continuing in this manner, we arrive at the mappings in Table 3.4.

In the final step of the procedure, we use the mappings in Table 3.4 to map every pixel in the histogram equalized image into a corresponding pixel in the newly created histogram-specified image. The values of the resulting histogram are listed in the third column of Table 3.2, and the histogram is shown in Fig. 3.22(d). The values of $p_z(z_q)$ were obtained using the same procedure as in Example 3.5. For instance, we see in Table 3.4 that $s_k = 1$ maps to $z_q = 3$, and there are 790 pixels in the histogram-equalized image with a value of 1. Therefore, $p_z(z_3) = 790/4096 = 0.19$.

Although the final result in Fig. 3.22(d) does not match the specified histogram exactly, the general trend of moving the intensities toward the high end of the intensity scale definitely was achieved. As mentioned earlier, obtaining the histogram-equalized image as an intermediate step is useful for

TABLE 3.3

Rounded values of the transformation function $G(z_q)$.

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	0
$z_2 = 2$	0
$z_3 = 3$	1
$z_4 = 4$	2
$z_5 = 5$	5
$z_6 = 6$	6
$z_7 = 7$	7

TABLE 3.4

Mapping of values s_k into corresponding values z_q .

s_k	→	z_q
1	→	3
3	→	4
5	→	5
6	→	6
7	→	7

explaining the procedure, but this is not necessary. Instead, we could list the mappings from the r 's to the s 's and from the s 's to the z 's in a three-column table. Then, we would use those mappings to map the original pixels directly into the pixels of the histogram-specified image.

EXAMPLE 3.8: Comparison between histogram equalization and histogram specification.

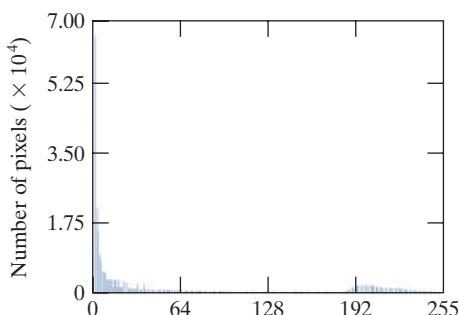
Figure 3.23(a) shows an image of the Mars moon, Phobos, taken by NASA's Mars Global Surveyor. Figure 3.23(b) shows the histogram of Fig. 3.23(a). The image is dominated by large, dark areas, resulting in a histogram characterized by a large concentration of pixels in the dark end of the gray scale. At first glance, one might conclude that histogram equalization would be a good approach to enhance this image, so that details in the dark areas become more visible. It is demonstrated in the following discussion that this is not so.

Figure 3.24(a) shows the histogram equalization transformation [Eq. (3-20)] obtained using the histogram in Fig. 3.23(b). The most relevant characteristic of this transformation function is how fast it rises from intensity level 0 to a level near 190. This is caused by the large concentration of pixels in the input histogram having levels near 0. When this transformation is applied to the levels of the input image to obtain a histogram-equalized result, the net effect is to map a very narrow interval of dark pixels into the

a b

FIGURE 3.23

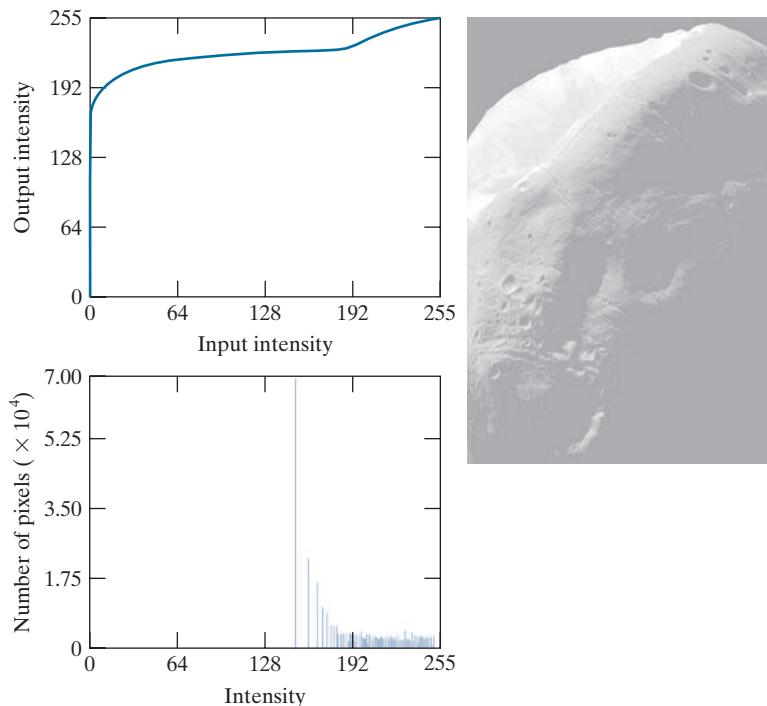
(a) An image, and
(b) its histogram.



a
b
c

FIGURE 3.24

- (a) Histogram equalization transformation obtained using the histogram in Fig. 3.23(b).
- (b) Histogram equalized image.
- (c) Histogram of equalized image.



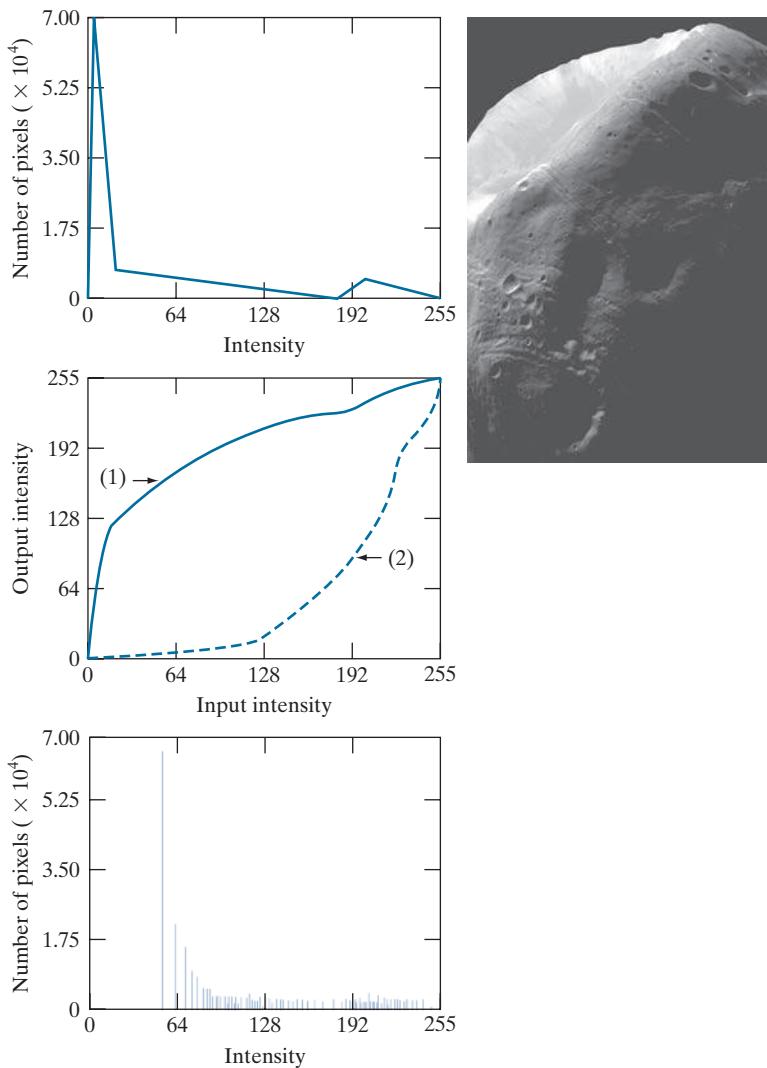
upper end of the gray scale of the output image. Because numerous pixels in the input image have levels precisely in this interval, we would expect the result to be an image with a light, washed-out appearance. As Fig. 3.24(b) shows, this is indeed the case. The histogram of this image is shown in Fig. 3.24(c). Note how all the intensity levels are biased toward the upper one-half of the gray scale.

Because the problem with the transformation function in Fig. 3.24(a) was caused by a large concentration of pixels in the original image with levels near 0, a reasonable approach is to modify the histogram of that image so that it does not have this property. Figure 3.25(a) shows a manually specified function that preserves the general shape of the original histogram, but has a smoother transition of levels in the dark region of the gray scale. Sampling this function into 256 equally spaced discrete values produced the desired specified histogram. The transformation function, $G(z_q)$, obtained from this histogram using Eq. (3-21) is labeled transformation (1) in Fig. 3.25(b). Similarly, the inverse transformation $G^{-1}(s_k)$, from Eq. (3-23) (obtained using the step-by-step procedure discussed earlier) is labeled transformation (2) in Fig. 3.25(b). The enhanced image in Fig. 3.25(c) was obtained by applying transformation (2) to the pixels of the histogram-equalized image in Fig. 3.24(b). The improvement of the histogram-specified image over the result obtained by histogram equalization is evident by comparing these two images. It is of interest to note that a rather modest change in the original histogram was all that was required to obtain a significant improvement in appearance. Figure 3.25(d) shows the histogram of Fig. 3.25(c). The most distinguishing feature of this histogram is how its low end has shifted right toward the lighter region of the gray scale (but not excessively so), as desired.

a c
b
d

FIGURE 3.25

Histogram specification.
 (a) Specified histogram.
 (b) Transformation $G(z_q)$, labeled (1), and $G^{-1}(s_k)$, labeled (2).
 (c) Result of histogram specification.
 (d) Histogram of image (c).



LOCAL HISTOGRAM PROCESSING

The histogram processing methods discussed thus far are *global*, in the sense that pixels are modified by a transformation function based on the intensity distribution of an entire image. This global approach is suitable for overall enhancement, but generally fails when the objective is to enhance details over small areas in an image. This is because the number of pixels in small areas have negligible influence on the computation of global transformations. The solution is to devise transformation functions based on the intensity distribution of pixel neighborhoods.

The histogram processing techniques previously described can be adapted to local enhancement. The procedure is to define a neighborhood and move its center from

pixel to pixel in a horizontal or vertical direction. At each location, the histogram of the points in the neighborhood is computed, and either a histogram equalization or histogram specification transformation function is obtained. This function is used to map the intensity of the pixel centered in the neighborhood. The center of the neighborhood is then moved to an adjacent pixel location and the procedure is repeated. Because only one row or column of the neighborhood changes in a one-pixel translation of the neighborhood, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible (see Problem 3.14). This approach has obvious advantages over repeatedly computing the histogram of all pixels in the neighborhood region each time the region is moved one pixel location. Another approach used sometimes to reduce computation is to utilize nonoverlapping regions, but this method usually produces an undesirable “blocky” effect.

EXAMPLE 3.9: Local histogram equalization.

Figure 3.26(a) is an 8-bit, 512×512 image consisting of five black squares on a light gray background. The image is slightly noisy, but the noise is imperceptible. There are objects embedded in the dark squares, but they are invisible for all practical purposes. Figure 3.26(b) is the result of global histogram equalization. As is often the case with histogram equalization of smooth, noisy regions, this image shows significant enhancement of the noise. However, other than the noise, Fig. 3.26(b) does not reveal any new significant details from the original. Figure 3.26(c) was obtained using local histogram equalization of Fig. 3.26(a) with a neighborhood of size 3×3 . Here, we see significant detail within all the dark squares. The intensity values of these objects are too close to the intensity of the dark squares, and their sizes are too small, to influence global histogram equalization significantly enough to show this level of intensity detail.

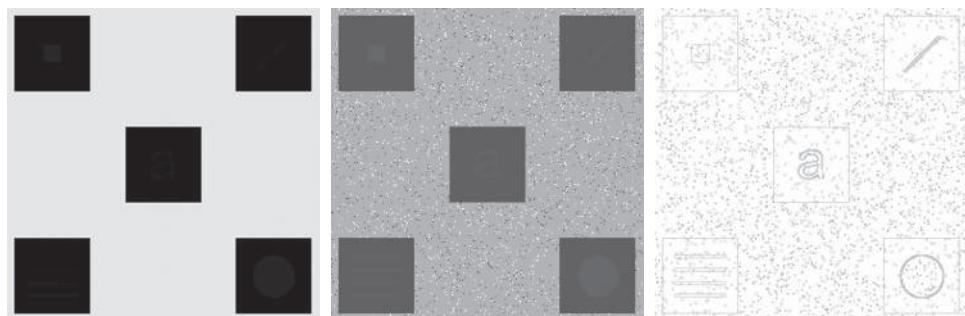
USING HISTOGRAM STATISTICS FOR IMAGE ENHANCEMENT

Statistics obtained directly from an image histogram can be used for image enhancement. Let r denote a discrete random variable representing intensity values in the range $[0, L - 1]$, and let $p(r_i)$ denote the normalized histogram component corresponding to intensity value r_i . As indicated earlier, we may view $p(r_i)$ as an estimate of the probability that intensity r_i occurs in the image from which the histogram was obtained.

a b c

FIGURE 3.26

- (a) Original image.
- (b) Result of global histogram equalization.
- (c) Result of local histogram equalization.



See the tutorials section in the book website for a review of probability.

For an image with intensity levels in the range $[0, L - 1]$, the n th moment of r about its mean, m , is defined as

$$\mu_n = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i) \quad (3-24)$$

where m is given by

$$m = \sum_{i=0}^{L-1} r_i p(r_i) \quad (3-25)$$

We follow convention in using m for the mean value. Do not confuse it with our use of the same symbol to denote the number of rows in an $m \times n$ neighborhood.

The mean is a measure of average intensity and the variance (or standard deviation, σ), given by

$$\sigma^2 = \mu_2 = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i) \quad (3-26)$$

is a measure of image contrast.

We consider two uses of the mean and variance for enhancement purposes. The *global* mean and variance [Eqs. (3-25) and (3-26)] are computed over an entire image and are useful for gross adjustments in overall intensity and contrast. A more powerful use of these parameters is in local enhancement, where the *local* mean and variance are used as the basis for making changes that depend on image characteristics in a neighborhood about each pixel in an image.

Let (x, y) denote the coordinates of any pixel in a given image, and let S_{xy} denote a neighborhood of specified size, centered on (x, y) . The mean value of the pixels in this neighborhood is given by the expression

$$m_{S_{xy}} = \sum_{i=0}^{L-1} r_i p_{S_{xy}}(r_i) \quad (3-27)$$

where $p_{S_{xy}}$ is the histogram of the pixels in region S_{xy} . This histogram has L bins, corresponding to the L possible intensity values in the input image. However, many of the bins will have 0 counts, depending on the size of S_{xy} . For example, if the neighborhood is of size 3×3 and $L = 256$, only between 1 and 9 of the 256 bins of the histogram of the neighborhood will be nonzero (the maximum number of possible *different* intensities in a 3×3 region is 9, and the minimum is 1). These non-zero values will correspond to the number of different intensities in S_{xy} .

The variance of the pixels in the neighborhood is similarly given by

$$\sigma_{S_{xy}}^2 = \sum_{i=0}^{L-1} (r_i - m_{S_{xy}})^2 p_{S_{xy}}(r_i) \quad (3-28)$$

As before, the local mean is a measure of average intensity in neighborhood S_{xy} , and the local variance (or standard deviation) is a measure of intensity contrast in that neighborhood.

As the following example illustrates, an important aspect of image processing using the local mean and variance is the flexibility these parameters afford in developing simple, yet powerful enhancement rules based on statistical measures that have a close, predictable correspondence with image appearance.

EXAMPLE 3.10: Local enhancement using histogram statistics.

Figure 3.27(a) is the same image as Fig. 3.26(a), which we enhanced using local histogram equalization. As noted before, the dark squares contain embedded symbols that are almost invisible. As before, we want to enhance the image to bring out these hidden features.

We can use the concepts presented in this section to formulate an approach for enhancing low-contrast details embedded in a background of similar intensity. The problem at hand is to enhance the low-contrast detail in the dark areas of the image, while leaving the light background unchanged.

A method used to determine whether an area is relatively light or dark at a point (x, y) is to compare the average local intensity, $m_{S_{xy}}$, to the average image intensity (the global mean), denoted by m_G . We obtain m_G using Eq. (3-25) with the histogram of the entire image. Thus, we have the first element of our enhancement scheme: We will consider the pixel at (x, y) as a candidate for processing if $k_0 m_G \leq m_{S_{xy}} \leq k_1 m_G$, where k_0 and k_1 are nonnegative constants and $k_0 < k_1$. For example, if our focus is on areas that are darker than one-quarter of the mean intensity, we would choose $k_0 = 0$ and $k_1 = 0.25$.

Because we are interested in enhancing areas that have low contrast, we also need a measure to determine whether the contrast of an area makes it a candidate for enhancement. We consider the pixel at (x, y) as a candidate if $k_2 \sigma_G \leq \sigma_{S_{xy}} \leq k_3 \sigma_G$, where σ_G is the global standard deviation obtained with Eq. (3-26) using the histogram of the entire image, and k_2 and k_3 are nonnegative constants, with $k_2 < k_3$. For example, to enhance a dark area of low contrast, we might choose $k_2 = 0$ and $k_3 = 0.1$. A pixel that meets all the preceding conditions for local enhancement is processed by multiplying it by a specified constant, C , to increase (or decrease) the value of its intensity level relative to the rest of the image. Pixels that do not meet the enhancement conditions are not changed.

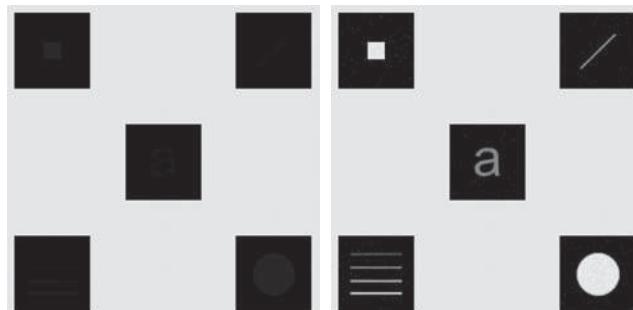
We summarize the preceding approach as follows. Let $f(x, y)$ denote the value of an image at any image coordinates (x, y) , and let $g(x, y)$ be the corresponding value in the enhanced image at those coordinates. Then,

$$g(x, y) = \begin{cases} Cf(x, y) & \text{if } k_0 m_G \leq m_{S_{xy}} \leq k_1 m_G \text{ AND } k_2 \sigma_G \leq \sigma_{S_{xy}} \leq k_3 \sigma_G \\ f(x, y) & \text{otherwise} \end{cases} \quad (3-29)$$

a b

FIGURE 3.27

(a) Original image. (b) Result of local enhancement based on local histogram statistics. Compare (b) with Fig. 3.26(c).



for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$, where, as indicated above, C , k_0 , k_1 , k_2 , and k_3 are specified constants, m_G is the global mean of the input image, and σ_G is its standard deviation. Parameters $m_{S_{xy}}$ and $\sigma_{S_{xy}}$ are the local mean and standard deviation, respectively, which change for every location (x, y) . As usual, M and N are the number of rows and columns in the input image.

Factors such as the values of the global mean and variance relative to values in the areas to be enhanced play a key role in selecting the parameters in Eq. (3-29), as does the range of differences between the intensities of the areas to be enhanced and their background. In the case of Fig. 3.27(a), $m_G = 161$, $\sigma_G = 103$, the maximum intensity values of the image and areas to be enhanced are 228 and 10, respectively, and the minimum values are 0 in both cases.

We would like for the maximum value of the enhanced features to be the same as the maximum value of the image, so we select $C = 22.8$. The areas to be enhanced are quite dark relative to the rest of the image, and they occupy less than a third of the image area; thus, we expect the mean intensity in the dark areas to be much less than the global mean. Based on this, we let $k_0 = 0$ and $k_1 = 0.1$. Because the areas to be enhanced are of very low contrast, we let $k_2 = 0$. For the upper limit of acceptable values of standard deviation we set $k_3 = 0.1$, which gives us one-tenth of the global standard deviation. Figure 3.27(b) is the result of using Eq. (3-29) with these parameters. By comparing this figure with Fig. 3.26(c), we see that the method based on local statistics detected the same hidden features as local histogram equalization. But the present approach extracted significantly more detail. For example, we see that all the objects are solid, but only the boundaries were detected by local histogram equalization. In addition, note that the intensities of the objects are not the same, with the objects in the top-left and bottom-right being brighter than the others. Also, the horizontal rectangles in the lower left square evidently are of different intensities. Finally, note that the background in both the image and dark squares in Fig. 3.27(b) is nearly the same as in the original image; by comparison, the same regions in Fig. 3.26(c) exhibit more visible noise and have lost their gray-level content. Thus, the additional complexity required to use local statistics yielded results in this case that are superior to local histogram equalization.

3.4 FUNDAMENTALS OF SPATIAL FILTERING

In this section, we discuss the use of spatial filters for image processing. Spatial filtering is used in a broad spectrum of image processing applications, so a solid understanding of filtering principles is important. As mentioned at the beginning of this chapter, the filtering examples in this section deal mostly with image enhancement. Other applications of spatial filtering are discussed in later chapters.

The name *filter* is borrowed from frequency domain processing (the topic of Chapter 4) where “filtering” refers to passing, modifying, or rejecting specified frequency components of an image. For example, a filter that passes low frequencies is called a *lowpass filter*. The net effect produced by a lowpass filter is to smooth an image by blurring it. We can accomplish similar smoothing directly on the image itself by using *spatial filters*.

Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbors. If the operation performed on the image pixels is linear, then the filter is called a *linear spatial filter*. Otherwise, the filter is a *nonlinear spatial filter*. We will focus attention first on linear filters and then introduce some basic nonlinear filters. Section 5.3 contains a more comprehensive list of nonlinear filters and their application.

See Section 2.6 regarding linearity.

THE MECHANICS OF LINEAR SPATIAL FILTERING

A linear spatial filter performs a sum-of-products operation between an image f and a *filter kernel*, w . The kernel is an array whose size defines the neighborhood of operation, and whose coefficients determine the nature of the filter. Other terms used to refer to a spatial filter kernel are *mask*, *template*, and *window*. We use the term *filter kernel* or simply *kernel*.

Figure 3.28 illustrates the mechanics of linear spatial filtering using a 3×3 kernel. At any point (x, y) in the image, the response, $g(x, y)$, of the filter is the sum of products of the kernel coefficients and the image pixels encompassed by the kernel:

$$\begin{aligned} g(x, y) = & w(-1, -1)f(x - 1, y - 1) + w(-1, 0)f(x - 1, y) + \dots \\ & + w(0, 0)f(x, y) + \dots + w(1, 1)f(x + 1, y + 1) \end{aligned} \quad (3-30)$$

As coordinates x and y are varied, the center of the kernel moves from pixel to pixel, generating the filtered image, g , in the process.[†]

Observe that the center coefficient of the kernel, $w(0, 0)$, aligns with the pixel at location (x, y) . For a kernel of size $m \times n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are nonnegative integers. This means that our focus is on kernels of odd size in both coordinate directions. In general, linear spatial filtering of an image of size $M \times N$ with a kernel of size $m \times n$ is given by the expression

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t)f(x + s, y + t) \quad (3-31)$$

where x and y are varied so that the center (origin) of the kernel visits every pixel in f once. For a fixed value of (x, y) , Eq. (3-31) implements the *sum of products* of the form shown in Eq. (3-30), but for a kernel of arbitrary odd size. As you will learn in the following section, this equation is a central tool in linear filtering.

It certainly is possible to work with kernels of even size, or mixed even and odd sizes. However, working with odd sizes simplifies indexing and is also more intuitive because the kernels have centers falling on integer values, and they are spatially symmetric.

SPATIAL CORRELATION AND CONVOLUTION

Spatial correlation is illustrated graphically in Fig. 3.28, and it is described mathematically by Eq. (3-31). Correlation consists of moving the center of a kernel over an image, and computing the sum of products at each location. The mechanics of *spatial convolution* are the same, except that the correlation kernel is rotated by 180°. Thus, when the values of a kernel are symmetric about its center, correlation and convolution yield the same result. The reason for rotating the kernel will become clear in the following discussion. The best way to explain the differences between the two concepts is by example.

We begin with a 1-D illustration, in which case Eq. (3-31) becomes

$$g(x) = \sum_{s=-a}^a w(s)f(x + s) \quad (3-32)$$

[†] A filtered pixel value typically is assigned to a corresponding location in a new image created to hold the results of filtering. It is seldom the case that filtered pixels replace the values of the corresponding location in the original image, as this would change the content of the image while filtering is being performed.

FIGURE 3.28

The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.

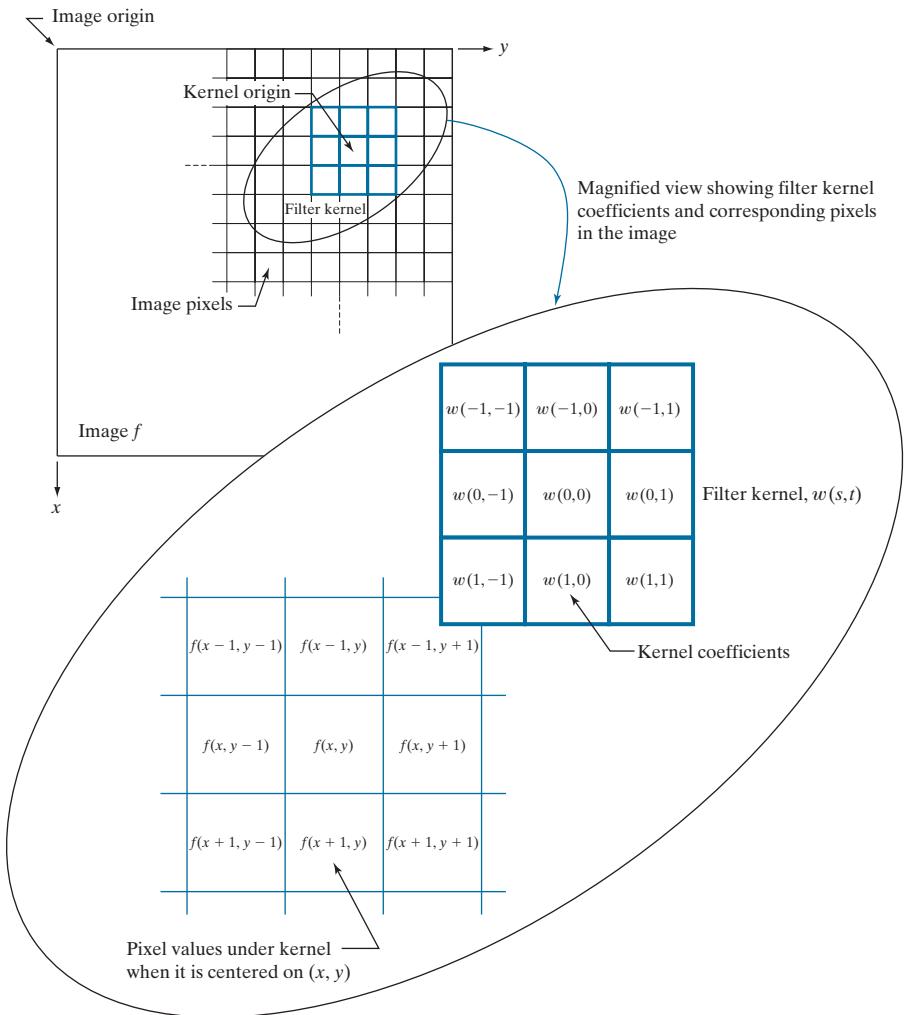


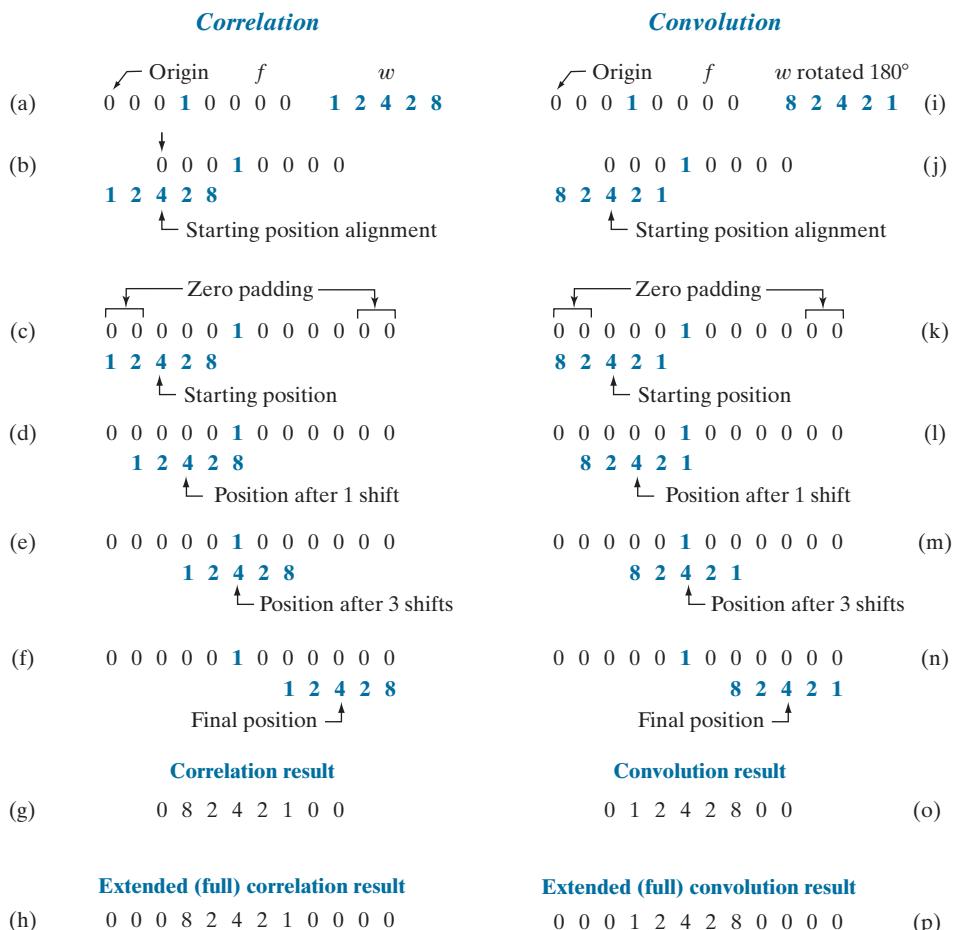
Figure 3.29(a) shows a 1-D function, f , and a kernel, w . The kernel is of size 1×5 , so $a = 2$ and $b = 0$ in this case. Figure 3.29(b) shows the starting position used to perform correlation, in which w is positioned so that its center coefficient is coincident with the origin of f .

The first thing we notice is that part of w lies outside f , so the summation is undefined in that area. A solution to this problem is to *pad* function f with enough 0's on either side. In general, if the kernel is of size $1 \times m$, we need $(m - 1)/2$ zeros on either side of f in order to handle the beginning and ending configurations of w with respect to f . Figure 3.29(c) shows a properly padded function. In this starting configuration, all coefficients of the kernel overlap valid values.

Zero padding is not the only padding option, as we will discuss in detail later in this chapter.

FIGURE 3.29

Illustration of 1-D correlation and convolution of a kernel, w , with a function f consisting of a discrete unit impulse. Note that correlation and convolution are functions of the variable x , which acts to *displace* one function with respect to the other. For the extended correlation and convolution results, the starting configuration places the right-most element of the kernel to be coincident with the origin of f . Additional padding must be used.



The first correlation value is the sum of products in this initial position, computed using Eq. (3-32) with $x = 0$:

$$g(0) = \sum_{s=2}^2 w(s)f(s+0) = 0$$

This value is in the leftmost location of the correlation result in Fig. 3.29(g).

To obtain the second value of correlation, we shift the relative positions of w and f one pixel location to the right [i.e., we let $x = 1$ in Eq. (3-32)] and compute the sum of products again. The result is $g(1) = 8$, as shown in the leftmost, nonzero location in Fig. 3.29(g). When $x = 2$, we obtain $g(2) = 2$. When $x = 3$, we get $g(3) = 4$ [see Fig. 3.29(e)]. Proceeding in this manner by varying x one shift at a time, we “build” the correlation result in Fig. 3.29(g). Note that it took 8 values of x (i.e., $x = 0, 1, 2, \dots, 7$) to fully shift w past f so the *center* coefficient in w visited *every* pixel in f . Sometimes, it is useful to have every element of w visit every pixel in f . For this, we have to start

with the rightmost element of w coincident with the origin of f , and end with the leftmost element of w being coincident the last element of f (additional padding would be required). Figure Fig. 3.29(h) shows the result of this *extended*, or *full*, correlation. As Fig. 3.29(g) shows, we can obtain the “standard” correlation by cropping the full correlation in Fig. 3.29(h).

There are two important points to note from the preceding discussion. First, correlation is a function of *displacement* of the filter kernel relative to the image. In other words, the first value of correlation corresponds to zero displacement of the kernel, the second corresponds to one unit displacement, and so on.[†] The second thing to notice is that correlating a kernel w with a function that contains all 0’s and a single 1 yields a *copy* of w , but *rotated* by 180°. A function that contains a single 1 with the rest being 0’s is called a *discrete unit impulse*. Correlating a kernel with a discrete unit impulse yields a *rotated* version of the kernel at the location of the impulse.

The right side of Fig. 3.29 shows the sequence of steps for performing convolution (we will give the equation for convolution shortly). The only difference here is that the kernel is *pre-rotated* by 180° prior to performing the shifting/sum of products operations. As the convolution in Fig. 3.29(o) shows, the result of pre-rotating the kernel is that now we have an *exact* copy of the kernel at the location of the unit impulse. In fact, a foundation of linear system theory is that convolving a function with an impulse yields a copy of the function at the location of the impulse. We will use this property extensively in Chapter 4.

The 1-D concepts just discussed extend easily to images, as Fig. 3.30 shows. For a kernel of size $m \times n$, we pad the image with a minimum of $(m - 1)/2$ rows of 0’s at the top and bottom and $(n - 1)/2$ columns of 0’s on the left and right. In this case, m and n are equal to 3, so we pad f with one row of 0’s above and below and one column of 0’s to the left and right, as Fig. 3.30(b) shows. Figure 3.30(c) shows the initial position of the kernel for performing correlation, and Fig. 3.30(d) shows the final result after the center of w visits every pixel in f , computing a sum of products at each location. As before, the result is a copy of the kernel, rotated by 180°. We will discuss the extended correlation result shortly.

For convolution, we pre-rotate the kernel as before and repeat the sliding sum of products just explained. Figures 3.30(f) through (h) show the result. You see again that convolution of a function with an impulse copies the function to the location of the impulse. As noted earlier, correlation and convolution yield the same result if the kernel values are symmetric about the center.

The concept of an impulse is fundamental in linear system theory, and is used in numerous places throughout the book. A *discrete impulse of strength (amplitude) A* located at coordinates (x_0, y_0) is defined as

$$\delta(x - x_0, y - y_0) = \begin{cases} A & \text{if } x = x_0 \text{ and } y = y_0 \\ 0 & \text{otherwise} \end{cases} \quad (3-33)$$

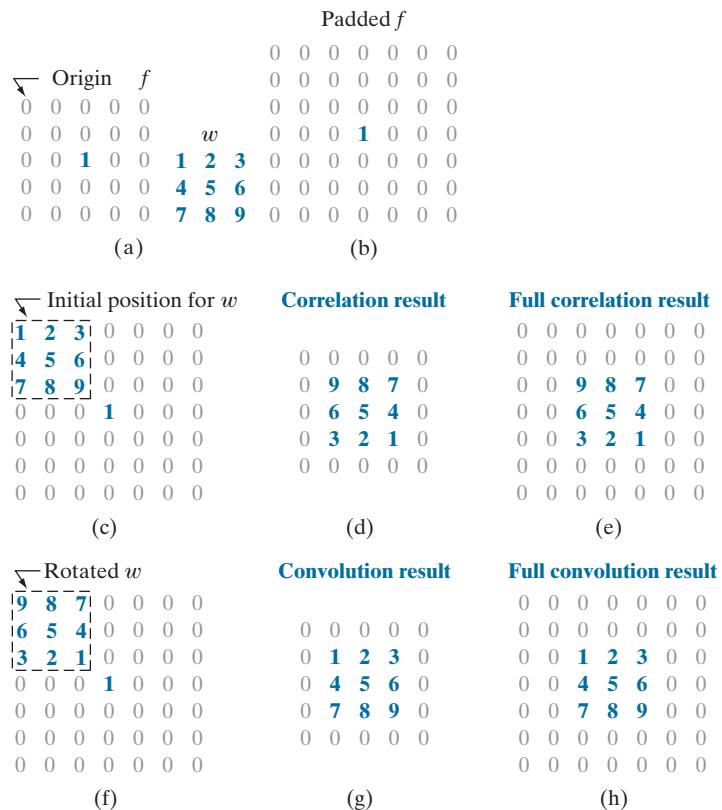
[†]In reality, we are shifting f to the left of w every time we increment x in Eq. (3-32). However, it is more intuitive to think of the smaller kernel moving right over the larger array f . The motion of the two is relative, so either way of looking at the motion is acceptable. The reason we increment f and not w is that indexing the equations for correlation and convolution is much easier (and clearer) this way, especially when working with 2-D arrays.

Rotating a 1-D kernel by 180° is equivalent to flipping the kernel about its axis.

In 2-D, rotation by 180° is equivalent to flipping the kernel about one axis and then the other.

FIGURE 3.30

Correlation (middle row) and convolution (last row) of a 2-D kernel with an image consisting of a discrete unit impulse. The 0's are shown in gray to simplify visual analysis. Note that correlation and convolution are functions of x and y . As these variable change, they displace one function with respect to the other. See the discussion of Eqs. (3-36) and (3-37) regarding full correlation and convolution.



Recall that $A = 1$ for a unit impulse.

For example, the unit impulse in Fig. 3.29(a) is given by $\delta(x - 3)$ in the 1-D version of the preceding equation. Similarly, the impulse in Fig. 3.30(a) is given by $\delta(x - 2, y - 2)$ [remember, the origin is at $(0,0)$].

Summarizing the preceding discussion in equation form, the correlation of a kernel w of size $m \times n$ with an image $f(x, y)$, denoted as $(w \star f)(x, y)$, is given by Eq. (3-31), which we repeat here for convenience:

$$(w \star f)(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t)f(x + s, y + t) \quad (3-34)$$

Because our kernels do not depend on (x, y) , we will sometimes make this fact explicit by writing the left side of the preceding equation as $w \star f(x, y)$. Equation (3-34) is evaluated for all values of the displacement variables x and y so that the center point of w visits every pixel in f [†] where we assume that f has been padded appropriately.

[†] As we mentioned earlier, the minimum number of required padding elements for a 2-D correlation is $(m - 1)/2$ rows above and below f , and $(n - 1)/2$ columns on the left and right. With this padding, and assuming that f is of size $M \times N$, the values of x and y required to obtain a complete correlation are $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$. This assumes that the starting configuration is such that the center of the kernel coincides with the origin of the image, which we have defined to be at the top, left (see Fig. 2.19).

As explained earlier, $a = (m - 1)/2$, $b = (n - 1)/2$, and we assume that m and n are odd integers.

In a similar manner, the *convolution* of a kernel w of size $m \times n$ with an image $f(x, y)$, denoted by $(w \star f)(x, y)$, is defined as

$$(w \star f)(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t)f(x - s, y - t) \quad (3-35)$$

where the minus signs align the coordinates of f and w when one of the functions is rotated by 180° (see Problem 3.17). This equation implements the sum of products process to which we refer throughout the book as *linear spatial filtering*. That is, linear spatial filtering and spatial convolution are synonymous.

Because convolution is commutative (see Table 3.5), it is immaterial whether w or f is rotated, but rotation of the kernel is used by convention. Our kernels do not depend on (x, y) , a fact that we sometimes make explicit by writing the left side of Eq. (3-35) as $w \star f(x, y)$. When the meaning is clear, we let the dependence of the previous two equations on x and y be implied, and use the simplified notation $w \star f$ and $w \star f$. As with correlation, Eq. (3-35) is evaluated for all values of the displacement variables x and y so that the center of w visits every pixel in f , which we assume has been padded. The values of x and y needed to obtain a full convolution are $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$. The size of the result is $M \times N$.

We can define correlation and convolution so that *every* element of w (instead of just its center) visits *every* pixel in f . This requires that the starting configuration be such that the right, lower corner of the kernel coincides with the origin of the image. Similarly, the ending configuration will be with the top left corner of the kernel coinciding with the lower right corner of the image. If the kernel and image are of sizes $m \times n$ and $M \times N$, respectively, the padding would have to increase to $(m - 1)$ padding elements above and below the image, and $(n - 1)$ elements to the left and right. Under these conditions, the size of the resulting full correlation or convolution array will be of size $S_v \times S_h$, where (see Figs. 3.30(e) and (h), and Problem 3.19),

$$S_v = m + M - 1 \quad (3-36)$$

and

$$S_h = n + N - 1 \quad (3-37)$$

Often, spatial filtering algorithms are based on correlation and thus implement Eq. (3-34) instead. To use the algorithm for correlation, we input w into it; for convolution, we input w rotated by 180° . The opposite is true for an algorithm that implements Eq. (3-35). Thus, either Eq. (3-34) or Eq. (3-35) can be made to perform the function of the other by rotating the filter kernel. Keep in mind, however, that the *order* of the functions input into a correlation algorithm *does* make a difference, because correlation is neither commutative nor associative (see Table 3.5).

TABLE 3.5

Some fundamental properties of convolution and correlation. A dash means that the property does not hold.

Because the values of these kernels are symmetric about the center, no rotation is required before convolution.

We could not write a similar equation for correlation because it is not commutative.

Property	Convolution	Correlation
Commutative	$f \star g = g \star f$	—
Associative	$f \star (g \star h) = (f \star g) \star h$	—
Distributive	$f \star (g + h) = (f \star g) + (f \star h)$	$f \star (g + h) = (f \star g) + (f \star h)$

Figure 3.31 shows two kernels used for smoothing the intensities of an image. To filter an image using one of these kernels, we perform a convolution of the kernel with the image in the manner just described. When talking about filtering and kernels, you are likely to encounter the terms *convolution filter*, *convolution mask*, or *convolution kernel* to denote filter kernels of the type we have been discussing. Typically, these terms are used in the literature to denote a spatial filter kernel, and not to imply necessarily that the kernel is used for convolution. Similarly, “convolving a kernel with an image” often is used to denote the sliding, sum-of-products process we just explained, and does not necessarily differentiate between correlation and convolution. Rather, it is used generically to denote either of the two operations. This imprecise terminology is a frequent source of confusion. In this book, when we use the term *linear spatial filtering*, we mean *convolving a kernel with an image*.

Sometimes an image is filtered (i.e., convolved) sequentially, in stages, using a different kernel in each stage. For example, suppose than an image f is filtered with a kernel w_1 , the result filtered with kernel w_2 , that result filtered with a third kernel, and so on, for Q stages. Because of the commutative property of convolution, this multistage filtering can be done in a single filtering operation, $w \star f$, where

$$w = w_1 \star w_2 \star w_3 \star \cdots \star w_Q \quad (3-38)$$

The size of w is obtained from the sizes of the individual kernels by successive applications of Eqs. (3-36) and (3-37). If all the individual kernels are of size $m \times n$, it follows from these equations that w will be of size $W_v \times W_h$, where

$$W_v = Q \times (m - 1) + m \quad (3-39)$$

and

$$W_h = Q \times (n - 1) + n \quad (3-40)$$

These equations assume that every value of a kernel visits every value of the array resulting from the convolution in the previous step. That is, the initial and ending configurations, are as described in connection with Eqs. (3-36) and (3-37).

a b

FIGURE 3.31

Examples of smoothing kernels:
(a) is a *box* kernel;
(b) is a *Gaussian* kernel.

$\frac{1}{9} \times$		$\frac{1}{4.8976} \times$	
----------------------	---	---------------------------	--

SEPARABLE FILTER KERNELS

As noted in Section 2.6, a 2-D function $G(x, y)$ is said to be *separable* if it can be written as the product of two 1-D functions, $G_1(x)$ and $G_2(y)$; that is, $G(x, y) = G_1(x)G_2(y)$. A spatial filter kernel is a matrix, and a separable kernel is a matrix that can be expressed as the outer product of two vectors. For example, the 2×3 kernel

$$w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

is separable because it can be expressed as the outer product of the vectors

$$\mathbf{c} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{r} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

That is,

$$\mathbf{c} \mathbf{r}^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix} [1 \ 1 \ 1] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = w$$

A separable kernel of size $m \times n$ can be expressed as the outer product of two vectors, \mathbf{v} and \mathbf{w} :

$$w = \mathbf{v} \mathbf{w}^T \quad (3-41)$$

where \mathbf{v} and \mathbf{w} are vectors of size $m \times 1$ and $n \times 1$, respectively. For a square kernel of size $m \times m$, we write

$$w = \mathbf{v} \mathbf{v}^T \quad (3-42)$$

It turns out that the product of a column vector and a row vector is the same as the 2-D convolution of the vectors (see Problem 3.24).

The importance of separable kernels lies in the computational advantages that result from the associative property of convolution. If we have a kernel w that can be decomposed into two simpler kernels, such that $w = w_1 \star w_2$, then it follows from the commutative and associative properties in Table 3.5 that

$$w \star f = (w_1 \star w_2) \star f = (w_2 \star w_1) \star f = w_2 \star (w_1 \star f) = (w_1 \star f) \star w_2 \quad (3-43)$$

This equation says that convolving a separable kernel with an image is the same as convolving w_1 with f first, and then convolving the result with w_2 .

We assume that the values of M and N include any padding of f prior to performing convolution.

For an image of size $M \times N$ and a kernel of size $m \times n$, implementation of Eq. (3-35) requires on the order of $MNmn$ multiplications and additions. This is because it follows directly from that equation that *each* pixel in the output (filtered) image depends on *all* the coefficients in the filter kernel. But, if the kernel is separable and we use Eq. (3-43), then the first convolution, $w_1 \star f$, requires on the order of MNm

To be strictly consistent in notation, we should use uppercase, bold symbols for kernels when we refer to them as matrices. However, kernels are mostly treated in the book as 2-D functions, which we denote in italics. To avoid confusion, we continue to use italics for kernels in this short section, with the understanding that the two notations are intended to be equivalent in this case.

multiplications and additions because w_1 is of size $m \times 1$. The result is of size $M \times N$, so the convolution of w_2 with the result requires MNn such operations, for a total of $MN(m + n)$ multiplication and addition operations. Thus, the *computational advantage* of performing convolution with a separable, as opposed to a nonseparable, kernel is defined as

$$C = \frac{MNmn}{MN(m+n)} = \frac{mn}{m+n} \quad (3-44)$$

For a kernel of modest size, say 11×11 , the computational advantage (and thus execution-time advantage) is a respectable 5.2. For kernels with hundreds of elements, execution times can be reduced by a factor of a hundred or more, which is significant. We will illustrate the use of such large kernels in Example 3.16.

We know from matrix theory that a matrix resulting from the product of a column vector and a row vector *always* has a rank of 1. By definition, a separable kernel is formed by such a product. Therefore, to determine if a kernel is separable, all we have to do is determine if its rank is 1. Typically, we find the rank of a matrix using a pre-programmed function in the computer language being used. For example, if you use MATLAB, function `rank` will do the job.

Once you have determined that the rank of a kernel matrix is 1, it is not difficult to find two vectors \mathbf{v} and \mathbf{w} such that their outer product, $\mathbf{v}\mathbf{w}^T$, is equal to the kernel. The approach consists of only three steps:

1. Find any nonzero element in the kernel and let E denote its value.
2. Form vectors \mathbf{c} and \mathbf{r} equal, respectively, to the column and row in the kernel containing the element found in Step 1.
3. With reference to Eq. (3-41), let $\mathbf{v} = \mathbf{c}$ and $\mathbf{w}^T = \mathbf{r}/E$.

The reason why this simple three-step method works is that the rows and columns of a matrix whose rank is 1 are linearly dependent. That is, the rows differ only by a constant multiplier, and similarly for the columns. It is instructive to work through the mechanics of this procedure using a small kernel (see Problems 3.20 and 3.22).

As we explained above, the objective is to find two 1-D kernels, w_1 and w_2 , in order to implement 1-D convolution. In terms of the preceding notation, $w_1 = \mathbf{c} = \mathbf{v}$ and $w_2 = \mathbf{r}/E = \mathbf{w}^T$. For circularly symmetric kernels, the column through the center of the kernel describes the entire kernel; that is, $w = \mathbf{v}\mathbf{v}^T/c$, where c is the value of the center coefficient. Then, the 1-D components are $w_1 = \mathbf{v}$ and $w_2 = \mathbf{v}^T/c$.

As we will discuss later in this chapter, the only kernels that are separable and whose values are circularly symmetric about the center are Gaussian kernels, which have a nonzero center coefficient (i.e., $c > 0$ for these kernels).

SOME IMPORTANT COMPARISONS BETWEEN FILTERING IN THE SPATIAL AND FREQUENCY DOMAINS

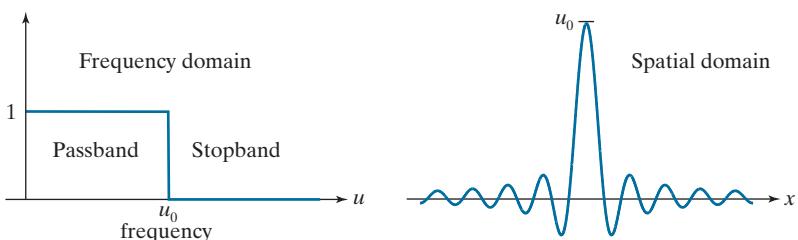
Although filtering in the frequency domain is the topic of Chapter 4, we introduce at this junction some important concepts from the frequency domain that will help you master the material that follows.

The tie between spatial- and frequency-domain processing is the *Fourier transform*. We use the Fourier transform to go from the spatial to the frequency domain;

a b

FIGURE 3.32

- (a) Ideal 1-D low-pass filter transfer function in the frequency domain.
 (b) Corresponding filter kernel in the spatial domain.



to return to the spatial domain we use the *inverse Fourier transform*. This will be covered in detail in Chapter 4. The focus here is on two fundamental properties relating the spatial and frequency domains:

1. Convolution, which is the basis for filtering in the spatial domain, is equivalent to multiplication in the frequency domain, and vice versa.
2. An impulse of strength A in the spatial domain is a constant of value A in the frequency domain, and vice versa.

See the explanation of Eq. (3-33) regarding impulses.

As explained in Chapter 4, a function (e.g., an image) satisfying some mild conditions can be expressed as the sum of sinusoids of different frequencies and amplitudes. Thus, the *appearance* of an image depends on the frequencies of its sinusoidal components—change the frequencies of those components, and you will change the appearance of the image. What makes this a powerful concept is that it is possible to associate certain frequency bands with image characteristics. For example, regions of an image with intensities that vary slowly (e.g., the walls in an image of a room) are characterized by sinusoids of low frequencies. Similarly, edges and other sharp intensity transitions are characterized by high frequencies. Thus, reducing the high-frequency components of an image will tend to blur it.

Linear filtering is concerned with finding suitable ways to modify the frequency content of an image. In the spatial domain we do this via convolution filtering. In the frequency domain we do it with multiplicative filters. The latter is a much more intuitive approach, which is one of the reasons why it is virtually impossible to truly understand spatial filtering without having at least some rudimentary knowledge of the frequency domain.

An example will help clarify these ideas. For simplicity, consider a 1-D function (such as an intensity scan line through an image) and suppose that we want to eliminate all its frequencies above a cutoff value, u_0 , while “passing” all frequencies below that value. Figure 3.32(a) shows a frequency-domain filter function for doing this. (The term *filter transfer function* is used to denote filter functions in the frequency domain—this is analogous to our use of the term “filter kernel” in the spatial domain.) Appropriately, the function in Fig. 3.32(a) is called a *lowpass* filter transfer function. In fact, this is an *ideal* lowpass filter function because it eliminates *all* frequencies above u_0 , while passing all frequencies below this value.[†] That is, the

As we did earlier with spatial filters, when the meaning is clear we use the term *filter* interchangeably with *filter transfer function* when working in the frequency domain.

[†]All the frequency domain filters in which we are interested are symmetrical about the origin and encompass both positive and negative frequencies, as we will explain in Section 4.3 (see Fig. 4.8). For the moment, we show only the right side (positive frequencies) of 1-D filters for simplicity in this short explanation.

transition of the filter between low and high frequencies is instantaneous. Such filter functions are not realizable with physical components, and have issues with “ringing” when implemented digitally. However, ideal filters are very useful for illustrating numerous filtering phenomena, as you will learn in Chapter 4.

To lowpass-filter a spatial signal in the frequency domain, we first convert it to the frequency domain by computing its Fourier transform, and then *multiply* the result by the filter transfer function in Fig. 3.32(a) to eliminate frequency components with values higher than u_0 . To return to the spatial domain, we take the inverse Fourier transform of the filtered signal. The result will be a blurred spatial domain function.

Because of the duality between the spatial and frequency domains, we can obtain the same result in the spatial domain by *convolving* the equivalent spatial domain filter kernel with the input spatial function. The equivalent spatial filter kernel is the inverse Fourier transform of the frequency-domain filter transfer function. Figure 3.32(b) shows the spatial filter kernel corresponding to the frequency domain filter transfer function in Fig. 3.32(a). The ringing characteristics of the kernel are evident in the figure. A central theme of digital filter design theory is obtaining faithful (and practical) approximations to the sharp cut off of ideal frequency domain filters while reducing their ringing characteristics.

A WORD ABOUT HOW SPATIAL FILTER KERNELS ARE CONSTRUCTED

We consider three basic approaches for constructing spatial filters in the following sections of this chapter. One approach is based on formulating filters based on mathematical properties. For example, a filter that computes the average of pixels in a neighborhood blurs an image. Computing an average is analogous to integration. Conversely, a filter that computes the local derivative of an image sharpens the image. We give numerous examples of this approach in the following sections.

A second approach is based on sampling a 2-D spatial function whose shape has a desired property. For example, we will show in the next section that samples from a Gaussian function can be used to construct a weighted-average (lowpass) filter. These 2-D spatial functions sometimes are generated as the inverse Fourier transform of 2-D filters specified in the frequency domain. We will give several examples of this approach in this and the next chapter.

A third approach is to design a spatial filter with a specified frequency response. This approach is based on the concepts discussed in the previous section, and falls in the area of digital filter design. A 1-D spatial filter with the desired response is obtained (typically using filter design software). The 1-D filter values can be expressed as a vector \mathbf{v} , and a 2-D separable kernel can then be obtained using Eq. (3-42). Or the 1-D filter can be rotated about its center to generate a 2-D kernel that approximates a circularly symmetric function. We will illustrate these techniques in Section 3.7.

3.5 SMOOTHING (LOWPASS) SPATIAL FILTERS

Smoothing (also called *averaging*) spatial filters are used to reduce sharp transitions in intensity. Because random noise typically consists of sharp transitions in

intensity, an obvious application of smoothing is noise reduction. Smoothing prior to image resampling to reduce aliasing, as will be discussed in Section 4.5, is also a common application. Smoothing is used to reduce irrelevant detail in an image, where “irrelevant” refers to pixel regions that are small with respect to the size of the filter kernel. Another application is for smoothing the false contours that result from using an insufficient number of intensity levels in an image, as discussed in Section 2.4. Smoothing filters are used in combination with other techniques for image enhancement, such as the histogram processing techniques discussed in Section 3.3, and unsharp masking, as discussed later in this chapter. We begin the discussion of smoothing filters by considering linear smoothing filters in some detail. We will introduce nonlinear smoothing filters later in this section.

As we discussed in Section 3.4, linear spatial filtering consists of convolving an image with a filter kernel. Convolving a smoothing kernel with an image blurs the image, with the degree of blurring being determined by the size of the kernel and the values of its coefficients. In addition to being useful in countless applications of image processing, lowpass filters are fundamental, in the sense that other important filters, including sharpening (highpass), bandpass, and bandreject filters, can be derived from lowpass filters, as we will show in Section 3.7.

We discuss in this section lowpass filters based on *box* and *Gaussian* kernels, both of which are separable. Most of the discussion will center on Gaussian kernels because of their numerous useful properties and breadth of applicability. We will introduce other smoothing filters in Chapters 4 and 5.

BOX FILTER KERNELS

The simplest, separable lowpass filter kernel is the *box kernel*, whose coefficients have the same value (typically 1). The name “box kernel” comes from a constant kernel resembling a box when viewed in 3-D. We showed a 3×3 box filter in Fig. 3.31(a). An $m \times n$ box filter is an $m \times n$ array of 1’s, with a normalizing constant in front, whose value is 1 divided by the sum of the values of the coefficients (i.e., $1/mn$ when all the coefficients are 1’s). This normalization, which we apply to all lowpass kernels, has two purposes. First, the average value of an area of constant intensity would equal that intensity in the filtered image, as it should. Second, normalizing the kernel in this way prevents introducing a *bias* during filtering; that is, the sum of the pixels in the original and filtered images will be the same (see Problem 3.31). Because in a box kernel all rows and columns are identical, the rank of these kernels is 1, which, as we discussed earlier, means that they are separable.

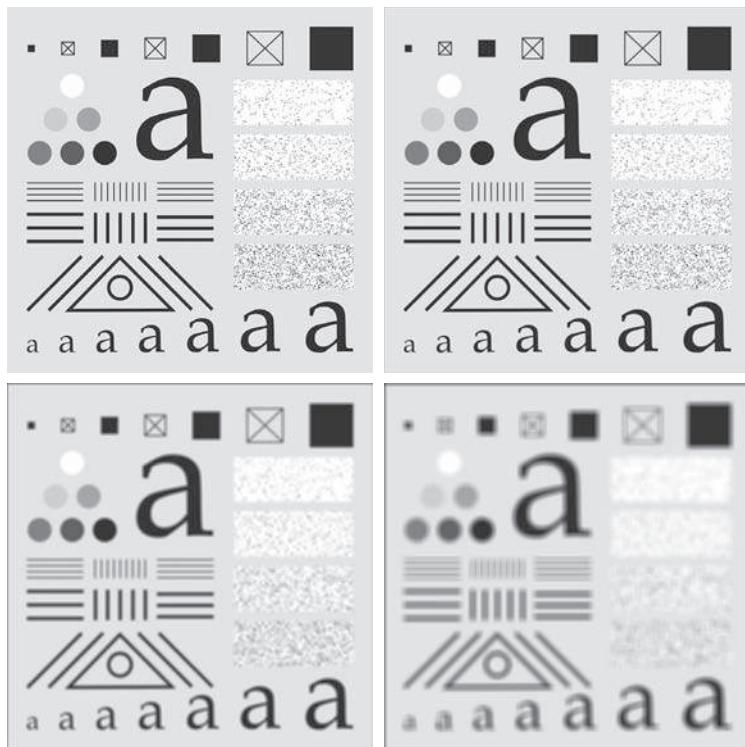
EXAMPLE 3.11: Lowpass filtering with a box kernel.

Figure 3.33(a) shows a test pattern image of size 1024×1024 pixels. Figures 3.33(b)-(d) are the results obtained using box filters of size $m \times m$ with $m = 3, 11$, and 21 , respectively. For $m = 3$, we note a slight overall blurring of the image, with the image features whose sizes are comparable to the size of the kernel being affected significantly more. Such features include the thinner lines in the image and the noise pixels contained in the boxes on the right side of the image. The filtered image also has a thin gray border, the result of zero-padding the image prior to filtering. As indicated earlier, padding extends the boundaries of an image to avoid undefined operations when parts of a kernel lie outside the border of

a	b
c	d

FIGURE 3.33

(a) Test pattern of size 1024×1024 pixels.
 (b)-(d) Results of lowpass filtering with box kernels of sizes 3×3 , 11×11 , and 21×21 , respectively.



the image during filtering. When zero (black) padding is used, the net result of smoothing at or near the border is a dark gray border that arises from including black pixels in the averaging process. Using the 11×11 kernel resulted in more pronounced blurring throughout the image, including a more prominent dark border. The result with the 21×21 kernel shows significant blurring of all components of the image, including the loss of the characteristic shape of some components, including, for example, the small square on the top left and the small character on the bottom left. The dark border resulting from zero padding is proportionally thicker than before. We used zero padding here, and will use it a few more times, so that you can become familiar with its effects. In Example 3.14 we discuss two other approaches to padding that eliminate the dark-border artifact that usually results from zero padding.

LOWPASS GAUSSIAN FILTER KERNELS

Because of their simplicity, box filters are suitable for quick experimentation and they often yield smoothing results that are visually acceptable. They are useful also when it is desired to reduce the effect of smoothing on edges (see Example 3.13). However, box filters have limitations that make them poor choices in many applications. For example, a defocused lens is often modeled as a lowpass filter, but box filters are poor approximations to the blurring characteristics of lenses (see Problem 3.33). Another limitation is the fact that box filters favor blurring along perpendicular directions. In applications involving images with a high level of detail,

or with strong geometrical components, the directionality of box filters often produces undesirable results. (Example 3.13 illustrates this issue.) These are but two applications in which box filters are not suitable.

The kernels of choice in applications such as those just mentioned are *circularly symmetric* (also called *isotropic*, meaning their response is independent of orientation). As it turns out, Gaussian kernels of the form

$$w(s, t) = G(s, t) = Ke^{-\frac{s^2 + t^2}{2\sigma^2}} \quad (3-45)$$

are the *only* circularly symmetric kernels that are also separable (Sahoo [1990]). Thus, because Gaussian kernels of this form are separable, Gaussian filters enjoy the same computational advantages as box filters, but have a host of additional properties that make them ideal for image processing, as you will learn in the following discussion. Variables s and t in Eq. (3-45), are real (typically discrete) numbers.

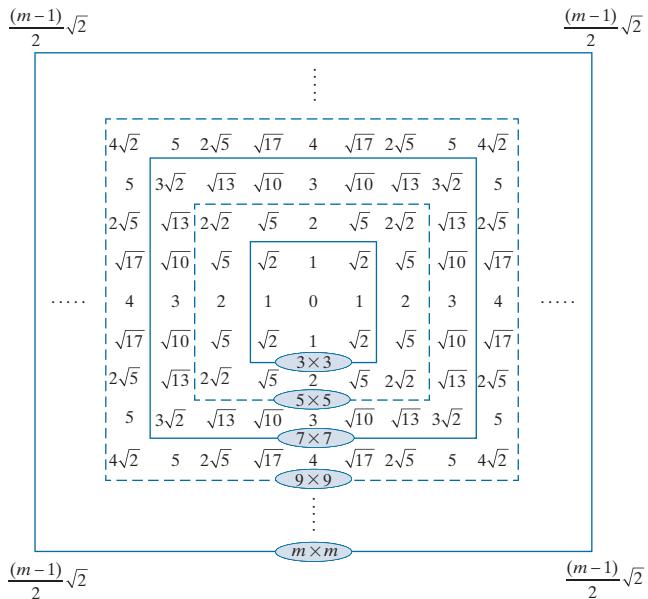
By letting $r = [s^2 + t^2]^{1/2}$ we can write Eq. (3-45) as

$$G(r) = Ke^{-\frac{r^2}{2\sigma^2}} \quad (3-46)$$

This equivalent form simplifies derivation of expressions later in this section. This form also reminds us that the function is circularly symmetric. Variable r is the distance from the center to any point on function G . Figure 3.34 shows values of r for several kernel sizes using integer values for s and t . Because we work generally with odd kernel sizes, the centers of such kernels fall on integer values, and it follows that all values of r^2 are integers also. You can see this by squaring the values in Fig. 3.34

FIGURE 3.34

Distances from the center for various sizes of square kernels.



(for a formal proof, see Padfield [2011]). Note in particular that the distance squared to the corner points for a kernel of size $m \times m$ is

$$r_{\max}^2 = \left[\frac{(m-1)}{2} \sqrt{2} \right]^2 = \frac{(m-1)^2}{2} \quad (3-47)$$

Small Gaussian kernels cannot capture the characteristic Gaussian bell shape, and thus behave more like box kernels. As we discuss below, a practical size for Gaussian kernels is on the order of $6\sigma \times 6\sigma$.

As we explained in Section 2.6, the symbols $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote the ceiling and floor functions. That is, the ceiling and floor functions map a real number to the smallest following, or the largest previous, integer, respectively.

Proofs of the results in Table 3.6 are simplified by working with the Fourier transform and the frequency domain, both of which are topics in Chapter 4.

The kernel in Fig. 3.31(b) was obtained by sampling Eq. (3-45) (with $K = 1$ and $\sigma = 1$). Figure 3.35(a) shows a perspective plot of a Gaussian function, and illustrates that the samples used to generate that kernel were obtained by specifying values of s and t , then “reading” the values of the function at those coordinates. These values are the coefficients of the kernel. Normalizing the kernel by dividing its coefficients by the sum of the coefficients completes the specification of the kernel. The reasons for normalizing the kernel are as discussed in connection with box kernels. Because Gaussian kernels are separable, we could simply take samples along a cross section through the center and use the samples to form vector \mathbf{v} in Eq. (3-42), from which we obtain the 2-D kernel.

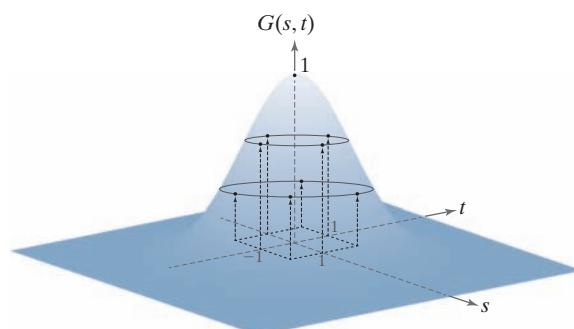
Separability is one of many fundamental properties of circularly symmetric Gaussian kernels. For example, we know that the values of a Gaussian function at a distance larger than 3σ from the mean are small enough that they can be ignored. This means that if we select the size of a Gaussian kernel to be $[6\sigma] \times [6\sigma]$ (the notation $\lceil c \rceil$ is used to denote the ceiling of c ; that is, the smallest integer not less than c), we are assured of getting essentially the same result as if we had used an arbitrarily large Gaussian kernel. Viewed another way, this property tells us that there is nothing to be gained by using a Gaussian kernel larger than $[6\sigma] \times [6\sigma]$ for image processing. Because typically we work with kernels of odd dimensions, we would use the smallest odd integer that satisfies this condition (e.g., a 43×43 kernel if $\sigma = 7$).

Two other fundamental properties of Gaussian functions are that the product and convolution of two Gaussians are Gaussian functions also. Table 3.6 shows the mean and standard deviation of the product and convolution of two 1-D Gaussian functions, f and g (remember, because of separability, we only need a 1-D Gaussian to form a circularly symmetric 2-D function). The mean and standard deviation

a | b

FIGURE 3.35

(a) Sampling a Gaussian function to obtain a discrete Gaussian kernel. The values shown are for $K = 1$ and $\sigma = 1$. (b) Resulting 3×3 kernel [this is the same as Fig. 3.31(b)].



$$\frac{1}{4.8976} \times$$

0.3679	0.6065	0.3679
0.6065	1.0000	0.6065
0.3679	0.6065	0.3679

TABLE 3.6 Mean and standard deviation of the product (\times) and convolution (\star) of two 1-D Gaussian functions, f and g . These results generalize directly to the product and convolution of more than two 1-D Gaussian functions (see Problem 3.25).

	f	g	$f \times g$	$f \star g$
Mean	m_f	m_g	$m_{f \times g} = \frac{m_f \sigma_g^2 + m_g \sigma_f^2}{\sigma_f^2 + \sigma_g^2}$	$m_{f \star g} = m_f + m_g$
Standard deviation	σ_f	σ_g	$\sigma_{f \times g} = \sqrt{\frac{\sigma_f^2 \sigma_g^2}{\sigma_f^2 + \sigma_g^2}}$	$\sigma_{f \star g} = \sqrt{\sigma_f^2 + \sigma_g^2}$

completely define a Gaussian, so the parameters in Table 3.6 tell us all there is to know about the functions resulting from multiplication and convolution of Gaussians. As indicated by Eqs. (3-45) and (3-46), Gaussian kernels have zero mean, so our interest here is in the standard deviations.

The convolution result is of particular importance in filtering. For example, we mentioned in connection with Eq. (3-43) that filtering sometimes is done in successive stages, and that the same result can be obtained by one stage of filtering with a composite kernel formed as the convolution of the individual kernels. If the kernels are Gaussian, we can use the result in Table 3.6 (which, as noted, generalizes directly to more than two functions) to compute the standard deviation of the composite kernel (and thus completely define it) without actually having to perform the convolution of all the individual kernels.

EXAMPLE 3.12: Lowpass filtering with a Gaussian kernel.

To compare Gaussian and box kernel filtering, we repeat Example 3.11 using a Gaussian kernel. Gaussian kernels have to be larger than box filters to achieve the same degree of blurring. This is because, whereas a box kernel assigns the same weight to all pixels, the values of Gaussian kernel coefficients (and hence their effect) decreases as a function of distance from the kernel center. As explained earlier, we use a size equal to the closest odd integer to $[6\sigma] \times [6\sigma]$. Thus, for a Gaussian kernel of size 21×21 , which is the size of the kernel we used to generate Fig. 3.33(d), we need $\sigma = 3.5$. Figure 3.36(b) shows the result of lowpass filtering the test pattern with this kernel. Comparing this result with Fig. 3.33(d), we see that the Gaussian kernel resulted in significantly less blurring. A little experimentation would show that we need $\sigma = 7$ to obtain comparable results. This implies a Gaussian kernel of size 43×43 . Figure 3.36(c) shows the results of filtering the test pattern with this kernel. Comparing it with Fig. 3.33(d), we see that the results indeed are very close.

We mentioned earlier that there is little to be gained by using a Gaussian kernel larger than $[6\sigma] \times [6\sigma]$. To demonstrate this, we filtered the test pattern in Fig. 3.36(a) using a Gaussian kernel with $\sigma = 7$ again, but of size 85×85 . Figure 3.37(a) is the same as Fig. 3.36(c), which we generated using the smallest odd kernel satisfying the $[6] \times [6]$ condition (43×43 , for $\sigma = 7$). Figure 3.37(b) is the result of using the 85×85 kernel, which is double the size of the other kernel. As you can see, not discernible additional

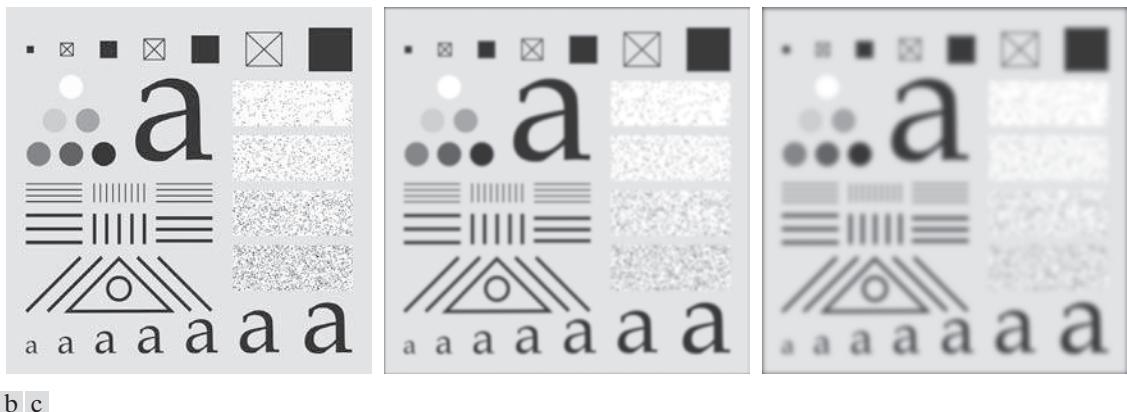


FIGURE 3.36 (a) A test pattern of size 1024×1024 . (b) Result of lowpass filtering the pattern with a Gaussian kernel of size 21×21 , with standard deviations $\sigma = 3.5$. (c) Result of using a kernel of size 43×43 , with $\sigma = 7$. This result is comparable to Fig. 3.33(d). We used $K = 1$ in all cases.

blurring occurred. In fact, the difference image in Fig 3.37(c) indicates that the two images are nearly identical, their maximum difference being 0.75, which is less than one level out of 256 (these are 8-bit images).

EXAMPLE 3.13: Comparison of Gaussian and box filter smoothing characteristics.

The results in Examples 3.11 and 3.12 showed little visual difference in blurring. Despite this, there are some subtle differences that are not apparent at first glance. For example, compare the large letter “a” in Figs. 3.33(d) and 3.36(c); the latter is much smoother around the edges. Figure 3.38 shows this type of different behavior between box and Gaussian kernels more clearly. The image of the rectangle was

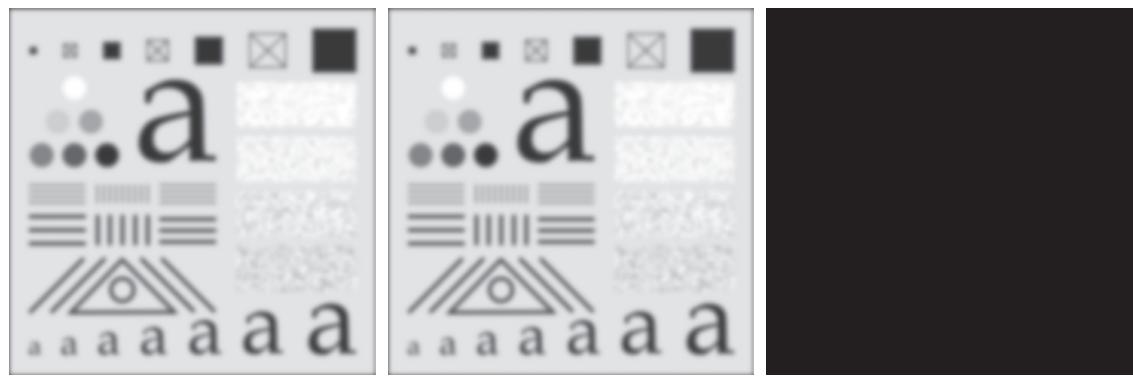


FIGURE 3.37 (a) Result of filtering Fig. 3.36(a) using a Gaussian kernels of size 43×43 , with $\sigma = 7$. (b) Result of using a kernel of 85×85 , with the same value of σ . (c) Difference image.

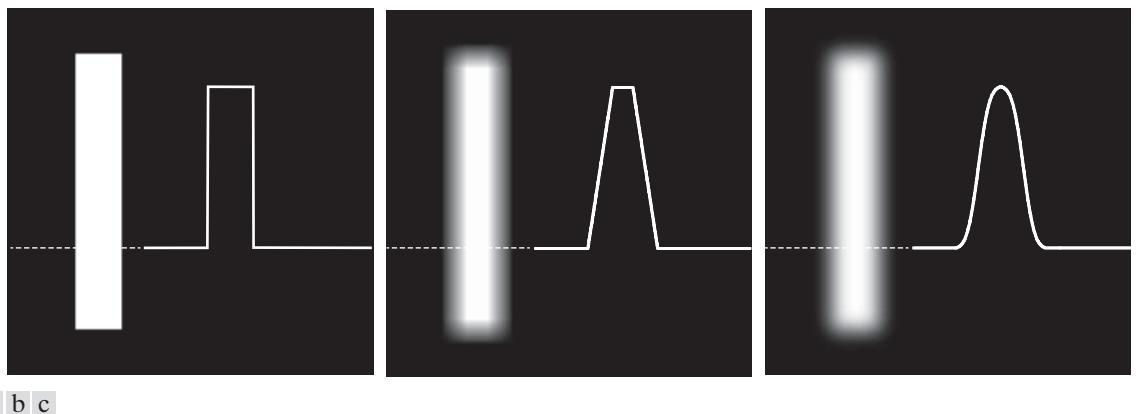


FIGURE 3.38 (a) Image of a white rectangle on a black background, and a horizontal intensity profile along the scan line shown dotted. (b) Result of smoothing this image with a box kernel of size 71×71 , and corresponding intensity profile. (c) Result of smoothing the image using a Gaussian kernel of size 151×151 , with $K = 1$ and $\sigma = 25$. Note the smoothness of the profile in (c) compared to (b). The image and rectangle are of sizes 1024×1024 and 768×128 pixels, respectively.

smoothed using a box and a Gaussian kernel with the sizes and parameters listed in the figure. These parameters were selected to give blurred rectangles of approximately the same width and height, in order to show the effects of the filters on a comparable basis. As the intensity profiles show, the box filter produced linear smoothing, with the transition from black to white (i.e., at an edge) having the shape of a ramp. The important features here are hard transitions at the onset and end of the ramp. We would use this type of filter when less smoothing of edges is desired. Conversely, the Gaussian filter yielded significantly smoother results around the edge transitions. We would use this type of filter when generally uniform smoothing is desired.

As the results in Examples 3.11, 3.12, and 3.13 show, zero padding an image introduces dark borders in the filtered result, with the thickness of the borders depending on the size and type of the filter kernel used. Earlier, when discussing correlation and convolution, we mentioned two other methods of image padding: *mirror* (also called *symmetric*) *padding*, in which values outside the boundary of the image are obtained by mirror-reflecting the image across its border; and *replicate* *padding*, in which values outside the boundary are set equal to the nearest image border value. The latter padding is useful when the areas near the border of the image are constant. Conversely, mirror padding is more applicable when the areas near the border contain image details. In other words, these two types of padding attempt to “extend” the characteristics of an image past its borders.

Figure 3.39 illustrates these padding methods, and also shows the effects of more aggressive smoothing. Figures 3.39(a) through 3.39(c) show the results of filtering Fig. 3.36(a) with a Gaussian kernel of size 187×187 elements with $K = 1$ and $\sigma = 31$, using zero, mirror, and replicate padding, respectively. The differences between the borders of the results with the zero-padded image and the other two are obvious,



FIGURE 3.39 Result of filtering the test pattern in Fig. 3.36(a) using (a) zero padding, (b) mirror padding, and (c) replicate padding. A Gaussian kernel of size 187×187 , with $K = 1$ and $\sigma = 31$ was used in all three cases.

and indicate that mirror and replicate padding yield more visually appealing results by eliminating the dark borders resulting from zero padding.

EXAMPLE 3.14: Smoothing performance as a function of kernel and image size.

The amount of relative blurring produced by a smoothing kernel of a given size depends directly on image size. To illustrate, Fig. 3.40(a) shows the same test pattern used earlier, but of size 4096×4096 pixels, four times larger in each dimension than before. Figure 3.40(b) shows the result of filtering this image with the same Gaussian kernel and padding used in Fig. 3.39(b). By comparison, the former image shows considerably less blurring for the same size filter. In fact, Fig. 3.40(b) looks more like the



FIGURE 3.40 (a) Test pattern of size 4096×4096 pixels. (b) Result of filtering the test pattern with the same Gaussian kernel used in Fig. 3.39. (c) Result of filtering the pattern using a Gaussian kernel of size 745×745 elements, with $K = 1$ and $\sigma = 124$. Mirror padding was used throughout.

image in Fig. 3.36(d), which was filtered using a 43×43 Gaussian kernel. In order to obtain results that are comparable to Fig. 3.39(b) we have to increase the size and standard deviation of the Gaussian kernel by four, the same factor as the increase in image dimensions. This gives a kernel of (odd) size 745×745 (with $K = 1$ and $\sigma = 124$). Figure 3.40(c) shows the result of using this kernel with mirror padding. This result is quite similar to Fig. 3.39(b). After the fact, this may seem like a trivial observation, but you would be surprised at how frequently not understanding the relationship between kernel size and the size of objects in an image can lead to ineffective performance of spatial filtering algorithms.

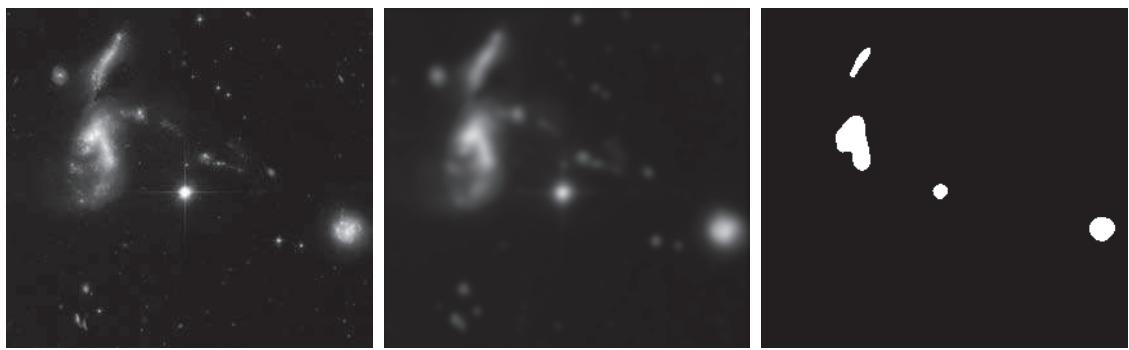
EXAMPLE 3.15: Using lowpass filtering and thresholding for region extraction.

Figure 3.41(a) is a 2566×2758 Hubble Telescope image of the *Hickson Compact Group* (see figure caption), whose intensities were scaled to the range $[0, 1]$. Our objective is to illustrate lowpass filtering combined with intensity thresholding for eliminating irrelevant detail in this image. In the present context, “irrelevant” refers to pixel regions that are small compared to kernel size.

Figure 3.41(b) is the result of filtering the original image with a Gaussian kernel of size 151×151 (approximately 6% of the image width) and standard deviation $\sigma = 25$. We chose these parameter values in order generate a sharper, more selective Gaussian kernel shape than we used in earlier examples. The filtered image shows four predominantly bright regions. We wish to extract only those regions from the image. Figure 3.41(c) is the result of thresholding the filtered image with a threshold $T = 0.4$ (we will discuss threshold selection in Chapter 10). As the figure shows, this approach effectively extracted the four regions of interest, and eliminated details deemed irrelevant in this application.

EXAMPLE 3.16: Shading correction using lowpass filtering.

One of the principal causes of image shading is nonuniform illumination. *Shading correction* (also called *flat-field correction*) is important because shading is a common cause of erroneous measurements, degraded performance of automated image analysis algorithms, and difficulty of image interpretation



a b c

FIGURE 3.41 (a) A 2566×2758 Hubble Telescope image of the *Hickson Compact Group*. (b) Result of lowpass filtering with a Gaussian kernel. (c) Result of thresholding the filtered image (intensities were scaled to the range $[0, 1]$). The Hickson Compact Group contains dwarf galaxies that have come together, setting off thousands of new star clusters. (Original image courtesy of NASA.)

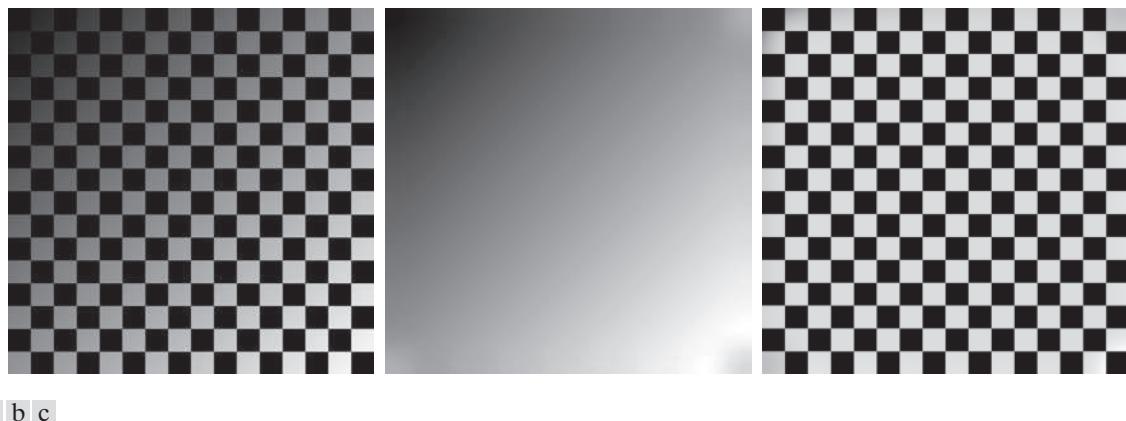


FIGURE 3.42 (a) Image shaded by a shading pattern oriented in the -45° direction. (b) Estimate of the shading patterns obtained using lowpass filtering. (c) Result of dividing (a) by (b). (See Section 9.8 for a morphological approach to shading correction).

by humans. We introduced shading correction in Example 2.7, where we corrected a shaded image by dividing it by the shading pattern. In that example, the shading pattern was given. Often, that is not the case in practice, and we are faced with having to estimate the pattern directly from available samples of shaded images. Lowpass filtering is a rugged, simple method for estimating shading patterns.

Consider the 2048×2048 checkerboard image in Fig. 3.42(a), whose inner squares are of size 128×128 pixels. Figure 3.42(b) is the result of lowpass filtering the image with a 512×512 Gaussian kernel (four times the size of the squares), $K = 1$, and $\sigma = 128$ (equal to the size of the squares). This kernel is just large enough to blur-out the squares (a kernel three times the size of the squares is too small to blur them out sufficiently). This result is a good approximation to the shading pattern visible in Fig. 3.42(a). Finally, Fig. 3.42(c) is the result of dividing (a) by (b). Although the result is not perfectly flat, it definitely is an improvement over the shaded image.

In the discussion of separable kernels in Section 3.4, we pointed out that the computational advantage of separable kernels can be significant for large kernels. It follows from Eq. (3-44) that the computational advantage of the kernel used in this example (which of course is separable) is 262 to 1. Thinking of computation time, if it took 30 sec to process a set of images similar to Fig. 3.42(b) using the two 1-D separable components of the Gaussian kernel, it would have taken 2.2 hrs to achieve the same result using a nonseparable lowpass kernel, or if we had used the 2-D Gaussian kernel directly, without decomposing it into its separable parts.

ORDER-STATISTIC (NONLINEAR) FILTERS

Order-statistic filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the region encompassed by the filter. Smoothing is achieved by replacing the value of the center pixel with the value determined by the ranking result. The best-known filter in this category is the *median filter*, which, as its name implies, replaces the value of the center pixel by the median of the intensity values in the neighborhood of that pixel (the value of the center pixel is included

in computing the median). Median filters provide excellent noise reduction capabilities for certain types of random noise, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of *impulse noise* (sometimes called *salt-and-pepper noise*, when it manifests itself as white and black dots superimposed on an image).

The *median*, ξ , of a set of values is such that half the values in the set are less than or equal to ξ and half are greater than or equal to ξ . In order to perform median filtering at a point in an image, we first sort the values of the pixels in the neighborhood, determine their median, and assign that value to the pixel in the filtered image corresponding to the center of the neighborhood. For example, in a 3×3 neighborhood the median is the 5th largest value, in a 5×5 neighborhood it is the 13th largest value, and so on. When several values in a neighborhood are the same, all equal values are grouped. For example, suppose that a 3×3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus, the principal function of median filters is to force points to be more like their neighbors. Isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $m^2/2$ (one-half the filter area), are forced by an $m \times m$ median filter to have the value of the median intensity of the pixels in the neighborhood (see Problem 3.36).

The median filter is by far the most useful order-statistic filter in image processing, but is not the only one. The median represents the 50th percentile of a ranked set of numbers, but ranking lends itself to many other possibilities. For example, using the 100th percentile results in the so-called *max filter*, which is useful for finding the brightest points in an image or for eroding dark areas adjacent to light regions. The response of a 3×3 max filter is given by $R = \max\{z_k \mid k = 1, 2, 3, \dots, 9\}$. The 0th percentile filter is the *min filter*, used for the opposite purpose. Median, max, min, and several other nonlinear filters will be considered in more detail in Section 5.3.

EXAMPLE 3.17: Median filtering.

Figure 3.43(a) shows an X-ray image of a circuit board heavily corrupted by salt-and-pepper noise. To illustrate the superiority of median filtering over lowpass filtering in situations such as this, we show in Fig. 3.43(b) the result of filtering the noisy image with a Gaussian lowpass filter, and in Fig. 3.43(c) the result of using a median filter. The lowpass filter blurred the image and its noise reduction performance was poor. The superiority in all respects of median over lowpass filtering in this case is evident.

3.6 SHARPENING (HIGHPASS) SPATIAL FILTERS

Sharpening highlights transitions in intensity. Uses of image sharpening range from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems. In Section 3.5, we saw that image blurring could be accomplished in the spatial domain by pixel averaging (smoothing) in a neighborhood. Because averaging is analogous to integration, it is logical to conclude that sharpening can be accomplished by spatial differentiation. In fact, this is the case, and the following discussion deals with various ways of defining and implementing operators for sharpening by digital differentiation. The strength of the response of

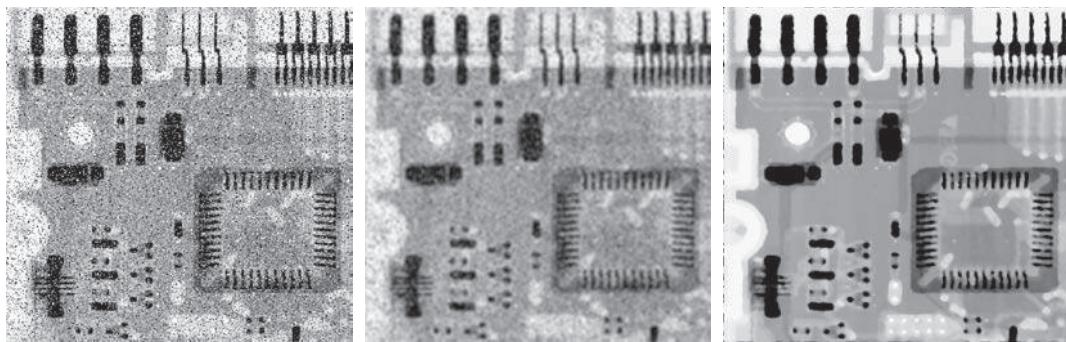


FIGURE 3.43 (a) X-ray image of a circuit board, corrupted by salt-and-pepper noise. (b) Noise reduction using a 19×19 Gaussian lowpass filter kernel with $\sigma = 3$. (c) Noise reduction using a 7×7 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

a derivative operator is proportional to the magnitude of the intensity discontinuity at the point at which the operator is applied. Thus, image differentiation enhances edges and other discontinuities (such as noise) and de-emphasizes areas with slowly varying intensities. As noted in Section 3.5, smoothing is often referred to as lowpass filtering, a term borrowed from frequency domain processing. In a similar manner, sharpening is often referred to as *highpass* filtering. In this case, high frequencies (which are responsible for fine details) are passed, while low frequencies are attenuated or rejected.

FOUNDATION

In the two sections that follow, we will consider in some detail sharpening filters that are based on first- and second-order derivatives, respectively. Before proceeding with that discussion, however, we stop to look at some of the fundamental properties of these derivatives in a digital context. To simplify the explanation, we focus attention initially on one-dimensional derivatives. In particular, we are interested in the behavior of these derivatives in areas of constant intensity, at the onset and end of discontinuities (*step* and *ramp discontinuities*), and along *intensity ramps*. As you will see in Chapter 10, these types of discontinuities can be used to model noise points, lines, and edges in an image.

Derivatives of a digital function are defined in terms of differences. There are various ways to define these differences. However, we require that any definition we use for a *first derivative*:

1. Must be zero in areas of constant intensity.
2. Must be nonzero at the onset of an intensity step or ramp.
3. Must be nonzero along intensity ramps.

Similarly, any definition of a *second derivative*

1. Must be zero in areas of constant intensity.
2. Must be nonzero at the onset *and* end of an intensity step or ramp.
3. Must be zero along intensity ramps.

We are dealing with digital quantities whose values are finite. Therefore, the maximum possible intensity change also is finite, and the shortest distance over which that change can occur is between adjacent pixels.

A basic definition of the *first-order* derivative of a one-dimensional function $f(x)$ is the difference

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \quad (3-48)$$

We used a partial derivative here in order to keep the notation consistent when we consider an image function of two variables, $f(x, y)$, at which time we will be dealing with partial derivatives along the two spatial axes. Clearly, $\partial f / \partial x = df / dx$ when there is only one variable in the function; the same is true for the second derivative.

We define the *second-order* derivative of $f(x)$ as the difference

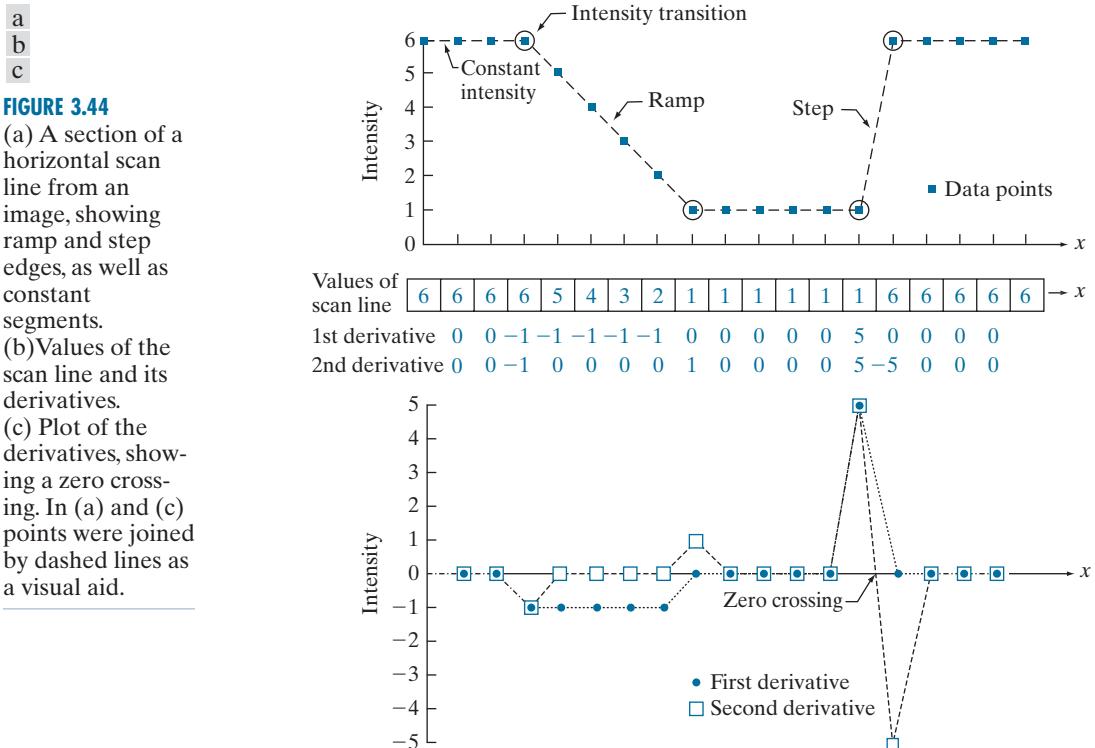
$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \quad (3-49)$$

These two definitions satisfy the conditions stated above, as we illustrate in Fig. 3.44, where we also examine the similarities and differences between first- and second-order derivatives of a digital function.

The values denoted by the small squares in Fig. 3.44(a) are the intensity values along a horizontal intensity profile (the dashed line connecting the squares is included to aid visualization). The actual numerical values of the scan line are shown inside the small boxes in 3.44(b). As Fig. 3.44(a) shows, the scan line contains three sections of constant intensity, an intensity ramp, and an intensity step. The circles indicate the onset or end of intensity transitions. The first- and second-order derivatives, computed using the two preceding definitions, are shown below the scan line values in Fig. 3.44(b), and are plotted in Fig. 3.44(c). When computing the first derivative at a location x , we subtract the value of the function at that location from the next point, as indicated in Eq. (3-48), so this is a “look-ahead” operation. Similarly, to compute the second derivative at x , we use the previous and the next points in the computation, as indicated in Eq. (3-49). To avoid a situation in which the previous or next points are outside the range of the scan line, we show derivative computations in Fig. 3.44 from the second through the penultimate points in the sequence.

As we traverse the profile from left to right we encounter first an area of constant intensity and, as Figs. 3.44(b) and (c) show, both derivatives are zero there, so condition (1) is satisfied by both. Next, we encounter an intensity ramp followed by a step, and we note that the first-order derivative is nonzero at the onset of the ramp and the step; similarly, the second derivative is nonzero at the onset and end of both the ramp and the step; therefore, property (2) is satisfied by both derivatives. Finally, we

We will return to Eq. (3-48) in Section 10.2 and show how it follows from a Taylor series expansion. For now, we accept it as a definition.



see that property (3) is satisfied also by both derivatives because the first derivative is nonzero and the second is zero along the ramp. Note that the sign of the second derivative changes at the onset and end of a step or ramp. In fact, we see in Fig. 3.44(c) that in a step transition a line joining these two values crosses the horizontal axis midway between the two extremes. This *zero crossing* property is quite useful for locating edges, as you will see in Chapter 10.

Edges in digital images often are ramp-like transitions in intensity, in which case the first derivative of the image would result in thick edges because the derivative is nonzero along a ramp. On the other hand, the second derivative would produce a double edge one pixel thick, separated by zeros. From this, we conclude that the second derivative enhances fine detail much better than the first derivative, a property ideally suited for sharpening images. Also, second derivatives require fewer operations to implement than first derivatives, so our initial attention is on the former.

USING THE SECOND DERIVATIVE FOR IMAGE SHARPENING—THE LAPLACIAN

We will return to the second derivative in Chapter 10, where we use it extensively for image segmentation.

In this section we discuss the implementation of 2-D, second-order derivatives and their use for image sharpening. The approach consists of defining a discrete formulation of the second-order derivative and then constructing a filter kernel based on

that formulation. As in the case of Gaussian lowpass kernels in Section 3.5, we are interested here in isotropic kernels, whose response is independent of the direction of intensity discontinuities in the image to which the filter is applied.

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

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (3-50)$$

Because derivatives of any order are linear operations, the Laplacian is a linear operator. To express this equation in discrete form, we use the definition in Eq. (3-49), keeping in mind that we now have a second variable. In the x -direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \quad (3-51)$$

and, similarly, in the y -direction, we have

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \quad (3-52)$$

It follows from the preceding three equations that the discrete Laplacian of two variables is

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \quad (3-53)$$

This equation can be implemented using convolution with the kernel in Fig. 3.45(a); thus, the filtering mechanics for image sharpening are as described in Section 3.5 for lowpass filtering; we are simply using different coefficients here.

The kernel in Fig. 3.45(a) is isotropic for rotations in increments of 90° with respect to the x - and y -axes. The diagonal directions can be incorporated in the definition of the digital Laplacian by adding four more terms to Eq. (3-53). Because each diagonal term would contain a $-2f(x, y)$ term, the total subtracted from the difference terms

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

a b c d

FIGURE 3.45 (a) Laplacian kernel used to implement Eq. (3-53). (b) Kernel used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other Laplacian kernels.

now would be $-8f(x, y)$. Figure 3.45(b) shows the kernel used to implement this new definition. This kernel yields isotropic results in increments of 45° . The kernels in Figs. 3.45(c) and (d) also are used to compute the Laplacian. They are obtained from definitions of the second derivatives that are the negatives of the ones we used here. They yield equivalent results, but the difference in sign must be kept in mind when combining a Laplacian-filtered image with another image.

Because the Laplacian is a derivative operator, it highlights sharp intensity transitions in an image and de-emphasizes regions of slowly varying intensities. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be “recovered” while still preserving the sharpening effect of the Laplacian by adding the Laplacian image to the original. As noted in the previous paragraph, it is important to keep in mind which definition of the Laplacian is used. If the definition used has a negative center coefficient, then we *subtract* the Laplacian image from the original to obtain a sharpened result. Thus, the basic way in which we use the Laplacian for image sharpening is

$$g(x, y) = f(x, y) + c[\nabla^2 f(x, y)] \quad (3-54)$$

where $f(x, y)$ and $g(x, y)$ are the input and sharpened images, respectively. We let $c = -1$ if the Laplacian kernels in Fig. 3.45(a) or (b) is used, and $c = 1$ if either of the other two kernels is used.

EXAMPLE 3.18: Image sharpening using the Laplacian.

Figure 3.46(a) shows a slightly blurred image of the North Pole of the moon, and Fig. 3.46(b) is the result of filtering this image with the Laplacian kernel in Fig. 3.45(a) directly. Large sections of this image are black because the Laplacian image contains both positive and negative values, and all negative values are clipped at 0 by the display.

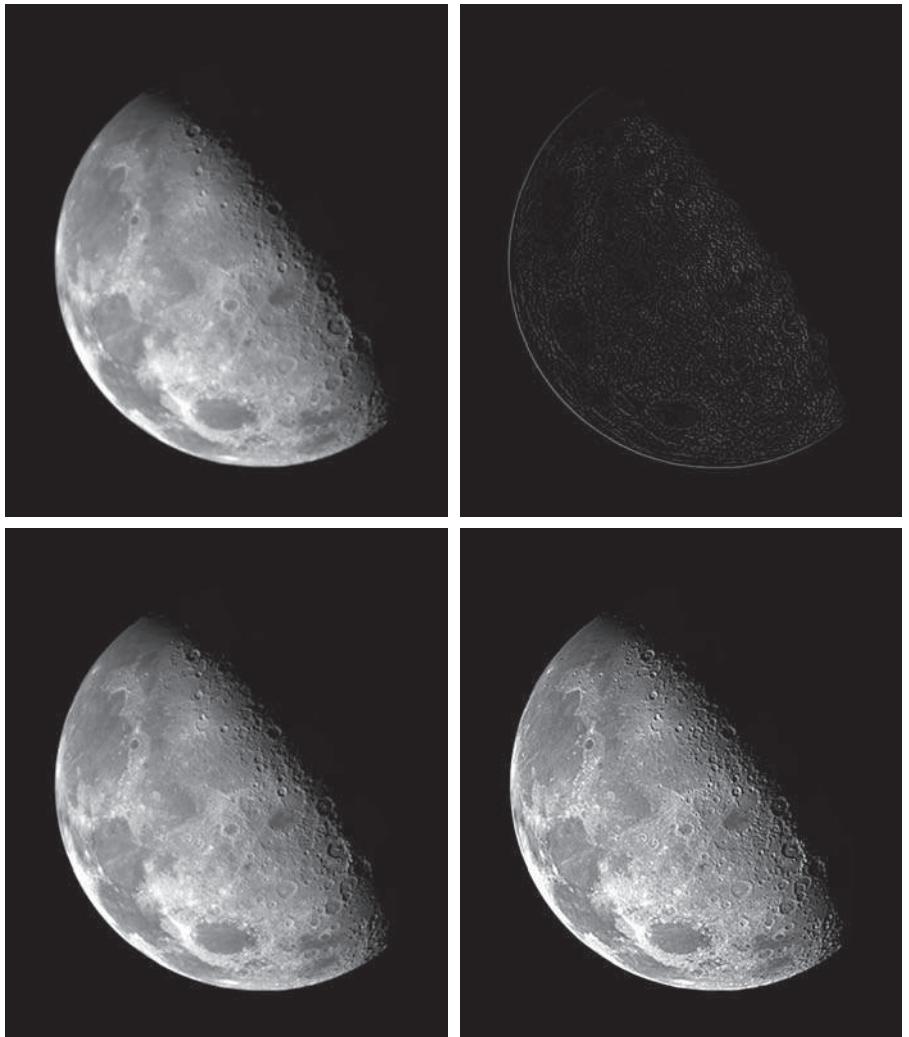
Figure 3.46(c) shows the result obtained using Eq. (3-54), with $c = -1$, because we used the kernel in Fig. 3.45(a) to compute the Laplacian. The detail in this image is unmistakably clearer and sharper than in the original image. Adding the Laplacian to the original image restored the overall intensity variations in the image. Adding the Laplacian increased the contrast at the locations of intensity discontinuities. The net result is an image in which small details were enhanced and the background tonality was reasonably preserved. Finally, Fig. 3.46(d) shows the result of repeating the same procedure but using the kernel in Fig. 3.45(b). Here, we note a significant improvement in sharpness over Fig. 3.46(c). This is not unexpected because using the kernel in Fig. 3.45(b) provides additional differentiation (sharpening) in the diagonal directions. Results such as those in Figs. 3.46(c) and (d) have made the Laplacian a tool of choice for sharpening digital images.

Because Laplacian images tend to be dark and featureless, a typical way to scale these images for display is to use Eqs. (2-31) and (2-32). This brings the most negative value to 0 and displays the full range of intensities. Figure 3.47 is the result of processing Fig. 3.46(b) in this manner. The dominant features of the image are edges and sharp intensity discontinuities. The background, previously black, is now gray as a result of scaling. This grayish appearance is typical of Laplacian images that have been scaled properly.

a	b
c	d

FIGURE 3.46

- (a) Blurred image of the North Pole of the moon.
 (b) Laplacian image obtained using the kernel in Fig. 3.45(a).
 (c) Image sharpened using Eq. (3-54) with $c = -1$.
 (d) Image sharpened using the same procedure, but with the kernel in Fig. 3.45(b).
 (Original image courtesy of NASA.)

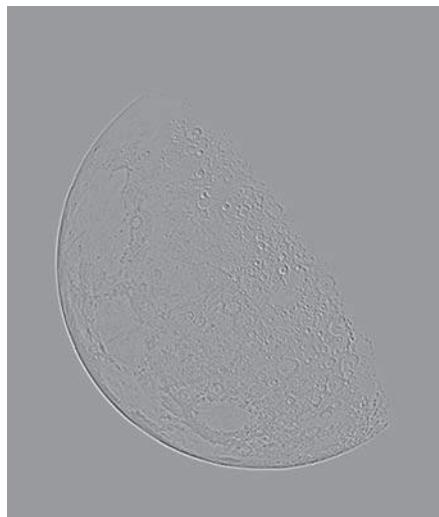


Observe in Fig. 3.45 that the coefficients of each kernel sum to zero. Convolution-based filtering implements a sum of products, so when a derivative kernel encompasses a constant region in a image, the result of convolution in that location must be zero. Using kernels whose coefficients sum to zero accomplishes this.

In Section 3.5, we normalized smoothing kernels so that the sum of their coefficients would be one. Constant areas in images filtered with these kernels would be constant also in the filtered image. We also found that the sum of the pixels in the original and filtered images were the same, thus preventing a bias from being introduced by filtering (see Problem 3.31). When convolving an image with a kernel

FIGURE 3.47

The Laplacian image from Fig. 3.46(b), scaled to the full [0, 255] range of intensity values. Black pixels correspond to the most negative value in the unscaled Laplacian image, grays are intermediate values, and white pixels corresponds to the highest positive value.



whose coefficients sum to zero, it turns out that the pixels of the filtered image *will sum to zero also* (see Problem 3.32). This implies that images filtered with such kernels will have negative values, and sometimes will require additional processing to obtain suitable visual results. Adding the filtered image to the original, as we did in Eq. (3-54), is an example of such additional processing.

UNSHARP MASKING AND HIGHBOOST FILTERING

The photographic process of unsharp masking is based on creating a blurred positive and using it along with the original negative to create a sharper image. Our interest is in the digital equivalent of this process.

Subtracting an unsharp (smoothed) version of an image from the original image is a process that has been used since the 1930s by the printing and publishing industry to sharpen images. This process, called *unsharp masking*, consists of the following steps:

- 1.** Blur the original image.
- 2.** Subtract the blurred image from the original (the resulting difference is called the *mask*.)
- 3.** Add the mask to the original.

Letting $\bar{f}(x, y)$ denote the blurred image, the mask in equation form is given by:

$$g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y) \quad (3-55)$$

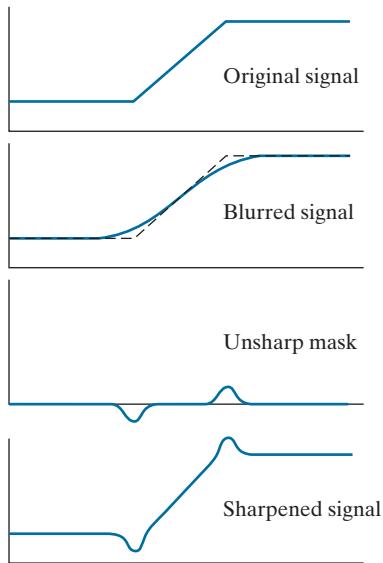
Then we add a weighted portion of the mask back to the original image:

$$g(x, y) = f(x, y) + k g_{\text{mask}}(x, y) \quad (3-56)$$

a
b
c
d

FIGURE 3.48

1-D illustration of the mechanics of unsharp masking.
(a) Original signal.
(b) Blurred signal with original shown dashed for reference.
(c) Unsharp mask.
(d) Sharpened signal, obtained by adding (c) to (a).



where we included a weight, k ($k \geq 0$), for generality. When $k = 1$ we have unsharp masking, as defined above. When $k > 1$, the process is referred to as *highboost filtering*. Choosing $k < 1$ reduces the contribution of the unsharp mask.

Figure 3.48 illustrates the mechanics of unsharp masking. Part (a) is a horizontal intensity profile across a vertical ramp edge that transitions from dark to light. Figure 3.48(b) shows the blurred scan line superimposed on the original signal (shown dashed). Figure 3.48(c) is the mask, obtained by subtracting the blurred signal from the original. By comparing this result with the section of Fig. 3.44(c) corresponding to the ramp in Fig. 3.44(a), we note that the unsharp mask in Fig. 3.48(c) is similar to what we would obtain using a second-order derivative. Figure 3.48(d) is the final sharpened result, obtained by adding the mask to the original signal. The points at which a change of slope occurs in the signal are now emphasized (sharpened). Observe that negative values were added to the original. Thus, it is possible for the final result to have negative intensities if the original image has any zero values, or if the value of k is chosen large enough to emphasize the peaks of the mask to a level larger than the minimum value in the original signal. Negative values cause dark halos around edges that can become objectionable if k is too large.

EXAMPLE 3.19: Unsharp masking and highboost filtering.

Figure 3.49(a) shows a slightly blurred image of white text on a dark gray background. Figure 3.49(b) was obtained using a Gaussian smoothing filter of size 31×31 with $\sigma = 5$. As explained in our earlier discussion of Gaussian lowpass kernels, the size of the kernel we used here is the smallest odd integer no less than $6\sigma \times 6\sigma$. Figure 3.49(c) is the unsharp mask, obtained using Eq. (3-55). To obtain the im-



FIGURE 3.49 (a) Original image of size 600×259 pixels. (b) Image blurred using a 31×31 Gaussian lowpass filter with $\sigma = 5$. (c) Mask. (d) Result of unsharp masking using Eq. (3-56) with $k = 1$. (e) Result of highboost filtering with $k = 4.5$.

age in Fig. 3.49(d) was used the unsharp masking expression room Eq. (3-56) with $k = 1$. This image is significantly sharper than the original image in Fig. 3.49(a), but we can do better, as we show in the following paragraph.

Figure 3.49(e) shows the result of using Eq. (3-56) with $k = 4.5$. This value is almost at the extreme of what we can use without introducing some serious artifacts in the image. The artifacts are dark, almost black, halos around the border of the characters. This is caused by the lower “blip” in Fig. 3.48(d) becoming negative, as we explained earlier. When scaling the image so that it only has positive values for display, the negative values are either clipped at 0, or scaled so that the most negative values become 0, depending on the scaling method used. In either case, the blips will be the darkest values in the image.

The results in Figs. 3.49(d) and 3.49(e) would be difficult to generate using the traditional film photography explained earlier, and it illustrates the power and versatility of image processing in the context of digital photography.

USING FIRST-ORDER DERIVATIVES FOR IMAGE SHARPENING—THE GRADIENT

We will discuss the gradient in more detail in Section 10.2. Here, we are interested only in using it for image sharpening.

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

$$\nabla f \equiv \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (3-57)$$

This vector has the important geometrical property that it points in the direction of the greatest rate of change of f at location (x, y) .

The *magnitude (length)* of vector ∇f , denoted as $M(x, y)$ (the vector norm notation $\|\nabla f\|$ is also used frequently), where

$$M(x, y) = \|\nabla f\| = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \quad (3-58)$$

is the *value* at (x, y) of the *rate of change* in the direction of the gradient vector. Note that $M(x, y)$ is an image of the same size as the original, created when x and y are allowed to vary over all pixel locations in f . It is common practice to refer to this image as the *gradient image* (or simply as the *gradient* when the meaning is clear).

Because the components of the gradient vector are derivatives, they are linear operators. However, the magnitude of this vector is not, because of the squaring and square root operations. On the other hand, the partial derivatives in Eq. (3-57) are not rotation invariant, but the magnitude of the gradient vector is.

In some implementations, it is more suitable computationally to approximate the squares and square root operations by absolute values:

The vertical bars denote absolute values.

$$M(x, y) \approx |g_x| + |g_y| \quad (3-59)$$

This expression still preserves the relative changes in intensity, but the isotropic property is lost in general. However, as in the case of the Laplacian, the isotropic properties of the discrete gradient defined in the following paragraph are preserved only for a limited number of rotational increments that depend on the kernels used to approximate the derivatives. As it turns out, the most popular kernels used to approximate the gradient are isotropic at multiples of 90° . These results are independent of whether we use Eq. (3-58) or (3-59), so nothing of significance is lost in using the latter equation if we choose to do so.

As in the case of the Laplacian, we now define discrete approximations to the preceding equations, and from these formulate the appropriate kernels. In order to simplify the discussion that follows, we will use the notation in Fig. 3.50(a) to denote the intensities of pixels in a 3×3 region. For example, the value of the center point, z_5 , denotes the value of $f(x, y)$ at an arbitrary location, (x, y) ; z_1 denotes the value of $f(x - 1, y - 1)$; and so on. As indicated in Eq. (3-48), the simplest approximations to a first-order derivative that satisfy the conditions stated at the beginning of this section are $g_x = (z_9 - z_5)$ and $g_y = (z_8 - z_6)$. Two other definitions, proposed by Roberts [1965] in the early development of digital image processing, use cross differences:

$$g_x = (z_9 - z_5) \quad \text{and} \quad g_y = (z_8 - z_6) \quad (3-60)$$

If we use Eqs. (3-58) and (3-60), we compute the gradient image as

$$M(x, y) = \left[(z_9 - z_5)^2 + (z_8 - z_6)^2 \right]^{1/2} \quad (3-61)$$

a
b
c

FIGURE 3.50

(a) A 3×3 region of an image, where the z s are intensity values.
 (b)–(c) Roberts cross-gradient operators.
 (d)–(e) Sobel operators. All the kernel coefficients sum to zero, as expected of a derivative operator.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	0	0	-1
0	1	1	0

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

If we use Eqs. (3-59) and (3-60), then

$$M(x, y) \approx |z_9 - z_5| + |z_8 - z_6| \quad (3-62)$$

where it is understood that x and y vary over the dimensions of the image in the manner described earlier. The difference terms needed in Eq. (3-60) can be implemented using the two kernels in Figs. 3.50(b) and (c). These kernels are referred to as the *Roberts cross-gradient operators*.

As noted earlier, we prefer to use kernels of odd sizes because they have a unique, (integer) center of spatial symmetry. The smallest kernels in which we are interested are of size 3×3 . Approximations to g_x and g_y using a 3×3 neighborhood centered on z_5 are as follows:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (3-63)$$

and

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (3-64)$$

These equations can be implemented using the kernels in Figs. 3.50(d) and (e). The difference between the third and first rows of the 3×3 image region approximates the partial derivative in the x -direction, and is implemented using the kernel in Fig. 3.50(d).

The difference between the third and first columns approximates the partial derivative in the y -direction and is implemented using the kernel in Fig. 3.50(e). The partial derivatives at all points in an image are obtained by convolving the image with these kernels. We then obtain the magnitude of the gradient as before. For example, substituting g_x and g_y into Eq. (3-59) yields

$$M(x, y) = \left[g_x^2 + g_y^2 \right]^{\frac{1}{2}} = \left[[(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)]^2 + [(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)]^2 \right]^{\frac{1}{2}} \quad (3-65)$$

This equation indicates that the value of M at any image coordinates (x, y) is given by squaring values of the convolution of the two kernels with image f at those coordinates, summing the two results, and taking the square root.

The kernels in Figs. 3.50(d) and (e) are called the *Sobel operators*. The idea behind using a weight value of 2 in the center coefficient is to achieve some smoothing by giving more importance to the center point (we will discuss this in more detail in Chapter 10). The coefficients in all the kernels in Fig. 3.50 sum to zero, so they would give a response of zero in areas of constant intensity, as expected of a derivative operator. As noted earlier, when an image is convolved with a kernel whose coefficients sum to zero, the elements of the resulting filtered image sum to zero also, so images convolved with the kernels in Fig. 3.50 will have negative values in general.

The computations of g_x and g_y are linear operations and are implemented using convolution, as noted above. The nonlinear aspect of sharpening with the gradient is the computation of $M(x, y)$ involving squaring and square roots, or the use of absolute values, all of which are nonlinear operations. These operations are performed after the linear process (convolution) that yields g_x and g_y .

EXAMPLE 3.20: Using the gradient for edge enhancement.

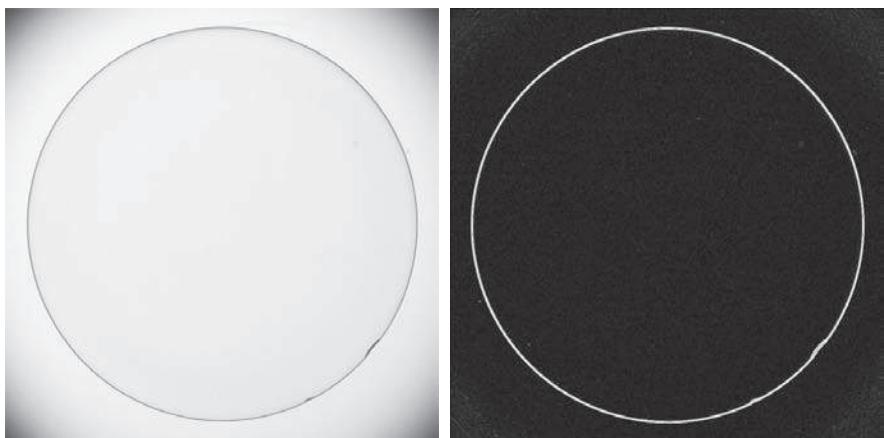
The gradient is used frequently in industrial inspection, either to aid humans in the detection of defects or, what is more common, as a preprocessing step in automated inspection. We will have more to say about this in Chapter 10. However, it will be instructive now to consider a simple example to illustrate how the gradient can be used to enhance defects and eliminate slowly changing background features.

Figure 3.51(a) is an optical image of a contact lens, illuminated by a lighting arrangement designed to highlight imperfections, such as the two edge defects in the lens boundary seen at 4 and 5 o'clock. Figure 3.51(b) shows the gradient obtained using Eq. (3-65) with the two Sobel kernels in Figs. 3.50(d) and (e). The edge defects are also quite visible in this image, but with the added advantage that constant or slowly varying shades of gray have been eliminated, thus simplifying considerably the computational task required for automated inspection. The gradient can be used also to highlight small specs that may not be readily visible in a gray-scale image (specs like these can be foreign matter, air pockets in a supporting solution, or minuscule imperfections in the lens). The ability to enhance small discontinuities in an otherwise flat gray field is another important feature of the gradient.

a b

FIGURE 3.51

- (a) Image of a contact lens (note defects on the boundary at 4 and 5 o'clock).
 (b) Sobel gradient. (Original image courtesy of Perceptics Corporation.)



3.7 HIGHPASS, BANDREJECT, AND BANDPASS FILTERS FROM LOW-PASS FILTERS

Spatial and frequency-domain linear filters are classified into four broad categories: lowpass and highpass filters, which we introduced in Sections 3.5 and 3.6, and *bandpass* and *bandreject* filters, which we introduce in this section. We mentioned at the beginning of Section 3.5 that the other three types of filters can be constructed from lowpass filters. In this section we explore methods for doing this. Also, we illustrate the third approach discussed at the end of Section 3.4 for obtaining spatial filter kernels. That is, we use a filter design software package to generate 1-D filter functions. Then, we use these to generate 2-D separable filters functions either via Eq.(3-42), or by rotating the 1-D functions about their centers to generate 2-D kernels. The rotated versions are approximations of circularly symmetric (isotropic) functions.

Figure 3.52(a) shows the transfer function of a 1-D ideal lowpass filter in the frequency domain [this is the same as Fig. 3.32(a)]. We know from earlier discussions in this chapter that lowpass filters attenuate or delete high frequencies, while passing low frequencies. A *highpass filter* behaves in exactly the opposite manner. As Fig. 3.52(b) shows, a highpass filter deletes or attenuates all frequencies below a cut-off value, u_0 , and passes all frequencies above this value. Comparing Figs. 3.52(a) and (b), we see that a highpass filter transfer function is obtained by subtracting a lowpass function from 1. This operation is in the frequency domain. As you know from Section 3.4, a constant in the frequency domain is an impulse in the spatial domain. Thus, we obtain a highpass filter kernel in the spatial domain by subtracting a lowpass filter kernel from a unit impulse with the same center as the kernel. An image filtered with this kernel is the same as an image obtained by subtracting a low-pass-filtered image from the original image. The unsharp mask defined by Eq. (3-55) is precisely this operation. Therefore, Eqs. (3-54) and (3-56) implement equivalent operations (see Problem 3.42).

Figure 3.52(c) shows the transfer function of a bandreject filter. This transfer function can be constructed from the sum of a lowpass and a highpass function with

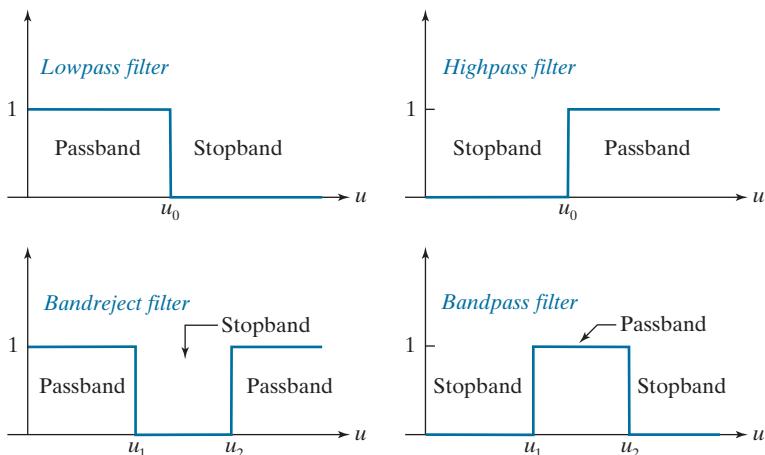
Recall from the discussion of Eq. (3-33) that a unit impulse is an array of 0's with a single 1.

a	b
c	d

FIGURE 3.52

Transfer functions of ideal 1-D filters in the frequency domain (u denotes frequency).

- (a) Lowpass filter.
- (b) Highpass filter.
- (c) Bandreject filter.
- (d) Bandpass filter.
(As before, we show only positive frequencies for simplicity.)



different cut-off frequencies (the highpass function can be constructed from a *different* lowpass function). The bandpass filter transfer function in Fig. 3.52(d) can be obtained by subtracting the bandreject function from 1 (a unit impulse in the spatial domain). Bandreject filters are also referred to as *notch* filters, but the latter tend to be more locally oriented, as we will show in Chapter 4. Table 3.7 summarizes the preceding discussion.

The key point in Fig. 3.52 and Table 3.7 is that all transfer functions shown can be obtained starting with a lowpass filter transfer function. This is important. It is important also to realize that we arrived at this conclusion via simple graphical interpretations in the frequency domain. To arrive at the same conclusion based on convolution in the spatial domain would be a much harder task.

EXAMPLE 3.21: Lowpass, highpass, bandreject, and bandpass filtering.

In this example we illustrate how we can start with a 1-D lowpass filter transfer function generated using a software package, and then use that transfer function to generate spatial filter kernels based on the concepts introduced in this section. We also examine the spatial filtering properties of these kernels.

TABLE 3.7

Summary of the four principal spatial filter types expressed in terms of low-pass filters. The centers of the unit impulse and the filter kernels coincide.

Filter type	Spatial kernel in terms of lowpass kernel, lp
Lowpass	$lp(x, y)$
Highpass	$hp(x, y) = \delta(x, y) - lp(x, y)$
Bandreject	$br(x, y) = lp_1(x, y) + hp_2(x, y)$ $= lp_1(x, y) + [\delta(x, y) - lp_2(x, y)]$
Bandpass	$bp(x, y) = \delta(x, y) - br(x, y)$ $= \delta(x, y) - [lp_1(x, y) + [\delta(x, y) - lp_2(x, y)]]$

FIGURE 3.53

A zone plate image of size 597×597 pixels.

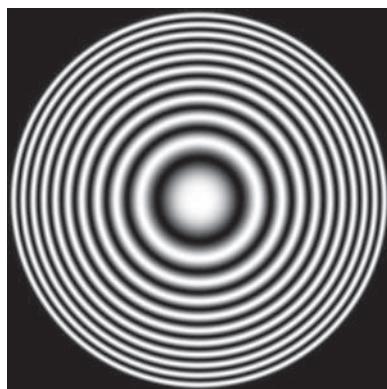


Figure 3.53 shows a so-called *zone plate* image that is used frequently for testing the characteristics of filtering approaches. There are various versions of zone plates; the one in Fig. 3.53 was generated using the equation

$$z(x, y) = \frac{1}{2} [1 + \cos(x^2 + y^2)] \quad (3-66)$$

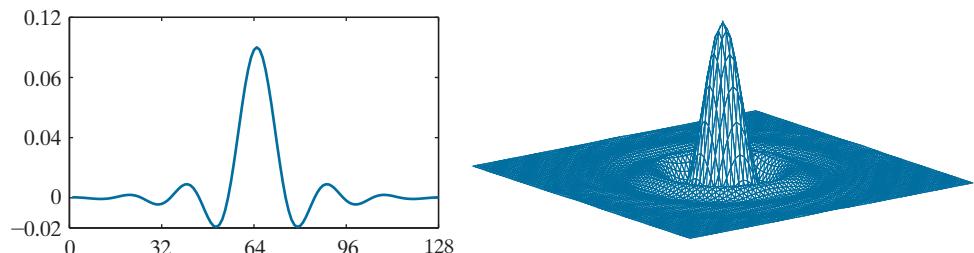
with x and y varying in the range $[-8.2, 8.2]$, in increments of 0.0275. This resulted in an image of size 597×597 pixels. The bordering black region was generated by setting to 0 all pixels with distance greater than 8.2 from the image center. The key characteristic of a zone plate is that its spatial frequency increases as a function of distance from the center, as you can see by noting that the rings get narrower the further they are from the center. This property makes a zone plate an ideal image for illustrating the behavior of the four filter types just discussed.

Figure 3.54(a) shows a 1-D, 128-element spatial lowpass filter function designed using MATLAB [compare with Fig. 3.32(b)]. As discussed earlier, we can use this 1-D function to construct a 2-D, separable lowpass filter kernel based on Eq. (3-42), or we can rotate it about its center to generate a 2-D, isotropic kernel. The kernel in Fig. 3.54(b) was obtained using the latter approach. Figures 3.55(a) and (b) are the results of filtering the image in Fig. 3.53 with the separable and isotropic kernels, respectively. Both filters passed the low frequencies of the zone plate while attenuating the high frequencies significantly. Observe, however, that the separable filter kernel produced a “squarish” (non-radially symmetric) result in the passed frequencies. This is a consequence of filtering the image in perpendicular directions with a separable kernel that is not isotropic. Using the isotropic kernel yielded a result that is uniform in all radial directions. This is as expected, because both the filter and the image are isotropic.

a b

FIGURE 3.54

(a) A 1-D spatial lowpass filter function. (b) 2-D kernel obtained by rotating the 1-D profile about its center.



a b

FIGURE 3.55

- (a) Zone plate image filtered with a separable lowpass kernel.
 (b) Image filtered with the isotropic lowpass kernel in Fig. 3.54(b).

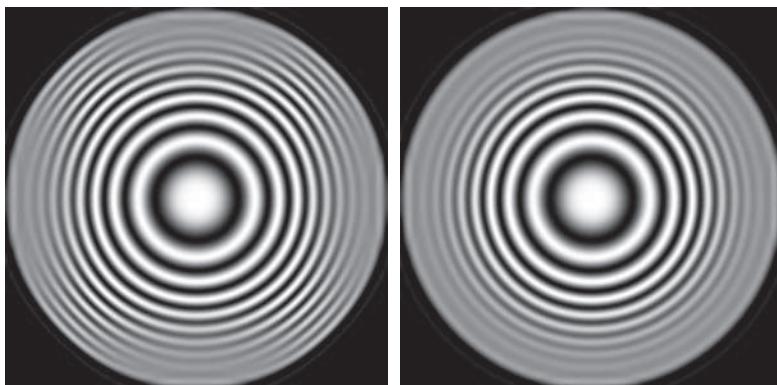


Figure 3.56 shows the results of filtering the zone plate with the four filters described in Table 3.7. We used the 2-D lowpass kernel in Fig. 3.54(b) as the basis for the highpass filter, and similar lowpass kernels for the bandreject filter. Figure 3.56(a) is the same as Fig. 3.55(b), which we repeat for convenience. Figure 3.56(b) is the highpass-filtered result. Note how effectively the low frequencies were filtered out. As is true of highpass-filtered images, the black areas were caused by negative values being clipped at 0 by the display. Figure 3.56(c) shows the same image scaled using Eqs. (2-31) and (2-32). Here we see clearly that only high frequencies were passed by the filter. Because the highpass kernel was constructed using the same lowpass kernel that we used to generate Fig. 3.56(a), it is evident by comparing the two results that the highpass filter passed the frequencies that were attenuated by the lowpass filter.

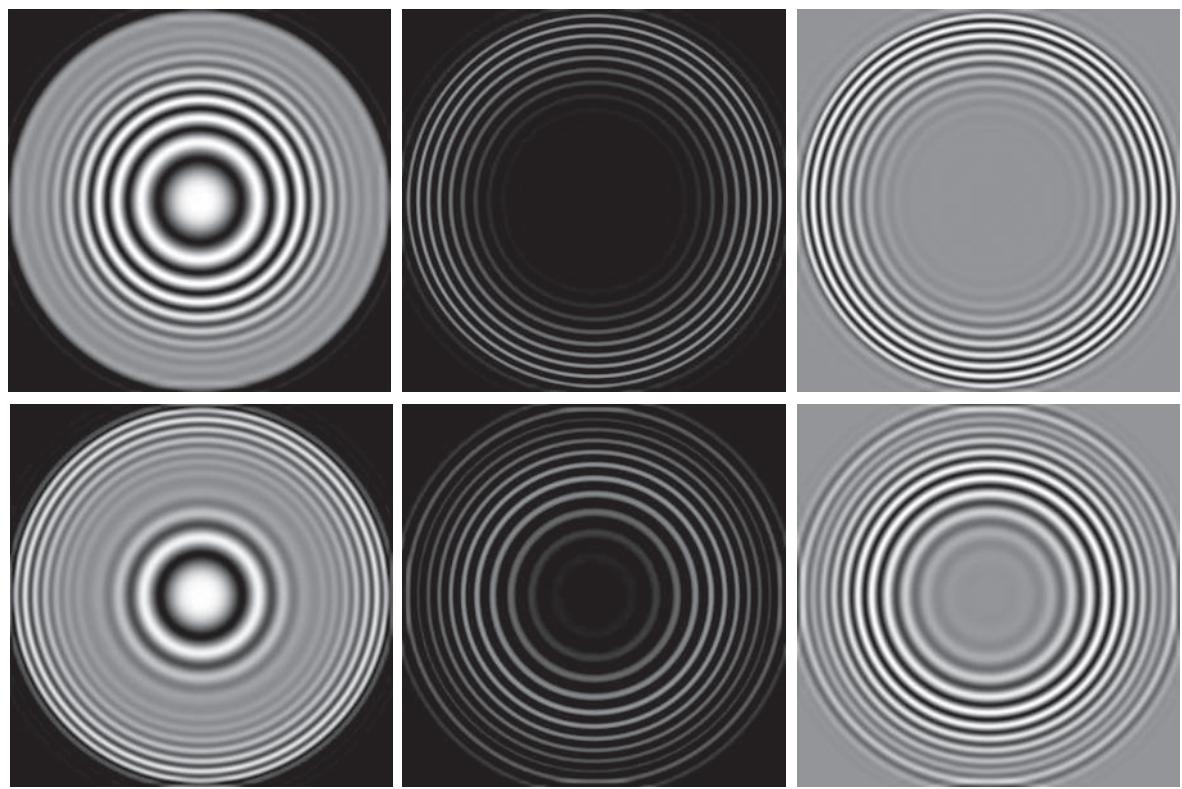
Figure 3.56(d) shows the bandreject-filtered image, in which the attenuation of the mid-band of frequencies is evident. Finally, Fig. 3.56(e) shows the result of bandpass filtering. This image also has negative values, so it is shown scaled in Fig. 3.56(f). Because the bandpass kernel was constructed by subtracting the bandreject kernel from a unit impulse, we see that the bandpass filter passed the frequencies that were attenuated by the bandreject filter. We will give additional examples of bandpass and bandreject filtering in Chapter 4.

3.8 COMBINING SPATIAL ENHANCEMENT METHODS

With a few exceptions, such as combining blurring with thresholding (Fig. 3.41), we have focused attention thus far on individual spatial-domain processing approaches. Frequently, a given task will require application of several complementary techniques in order to achieve an acceptable result. In this section, we illustrate how to combine several of the approaches developed thus far in this chapter to address a difficult image enhancement task.

The image in Fig. 3.57(a) is a nuclear whole body bone scan, used to detect diseases such as bone infections and tumors. Our objective is to enhance this image by sharpening it and by bringing out more of the skeletal detail. The narrow dynamic range of the intensity levels and high noise content make this image difficult to enhance. The strategy we will follow is to utilize the Laplacian to highlight fine detail, and the gradient to enhance prominent edges. For reasons that will be explained shortly, a smoothed version of the gradient image will be used to mask the Laplacian

In this context, masking refers to multiplying two images, as in Fig. 2.34. This is not to be confused with the mask used in unsharp masking.

**FIGURE 3.56**

Spatial filtering of the zone plate image. (a) Lowpass result; this is the same as Fig. 3.55(b). (b) Highpass result. (c) Image (b) with intensities scaled. (d) Bandreject result. (e) Bandpass result. (f) Image (e) with intensities scaled.

image. Finally, we will attempt to increase the dynamic range of the intensity levels by using an intensity transformation.

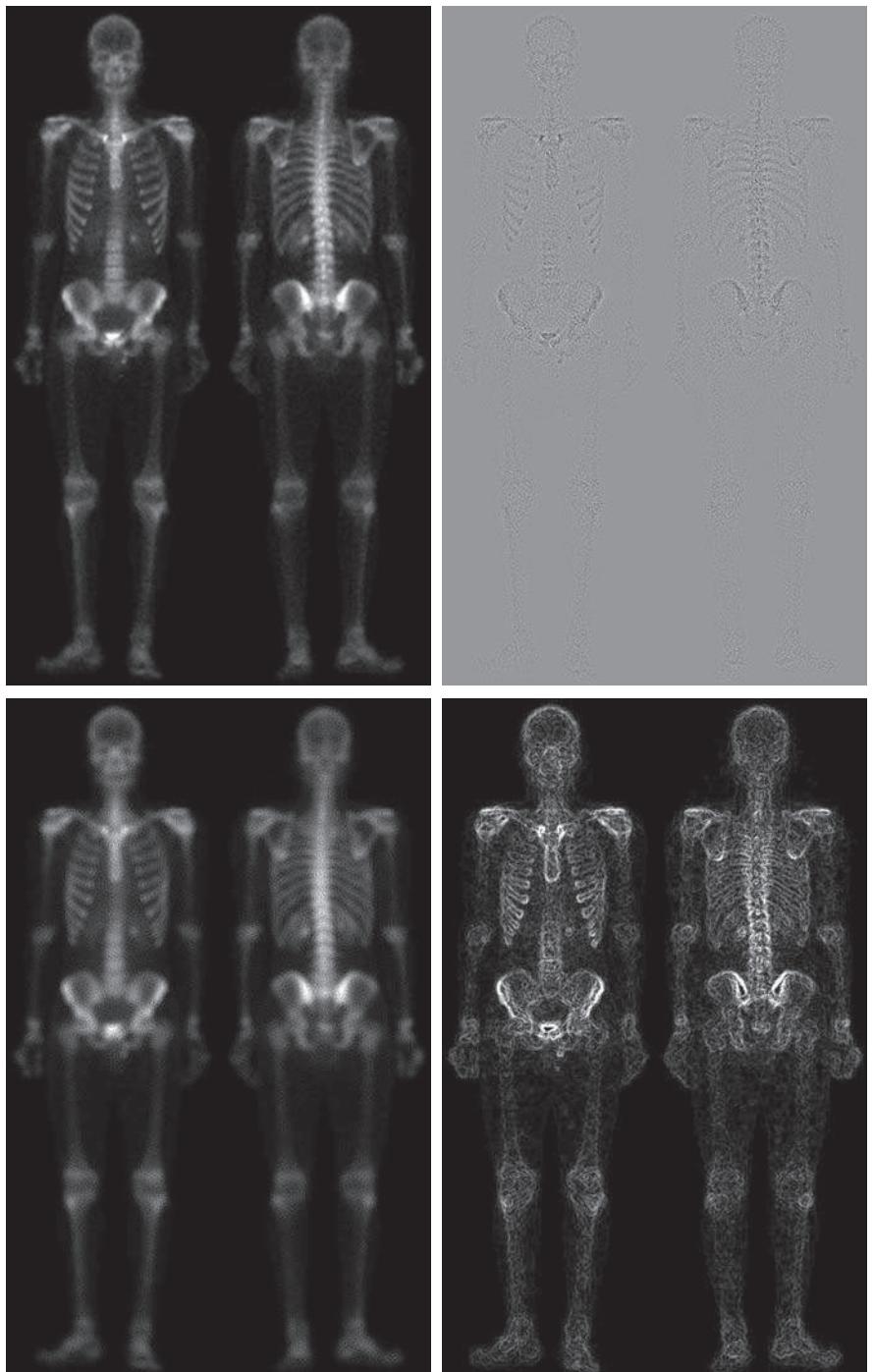
Figure 3.57(b) shows the Laplacian of the original image, obtained using the kernel in Fig. 3.45(d). This image was scaled (for display only) using the same technique as in Fig. 3.47. We can obtain a sharpened image at this point simply by adding Figs. 3.57(a) and (b), according to Eq. (3-54). Just by looking at the noise level in Fig. 3.57(b), we would expect a rather noisy sharpened image if we added Figs. 3.57(a) and (b). This is confirmed by the result in Fig. 3.57(c). One way that comes immediately to mind to reduce the noise is to use a median filter. However, median filtering is an aggressive nonlinear process capable of removing image features. This is unacceptable in medical image processing.

An alternate approach is to use a mask formed from a smoothed version of the gradient of the original image. The approach is based on the properties of first- and

a b
c d

FIGURE 3.57

- (a) Image of whole body bone scan.
(b) Laplacian of (a).
(c) Sharpened image obtained by adding (a) and (b).
(d) Sobel gradient of image (a). (Original image courtesy of G.E. Medical Systems.)



second-order derivatives we discussed when explaining Fig. 3.44. The Laplacian, is a second-order derivative operator and has the definite advantage that it is superior for enhancing fine detail. However, this causes it to produce noisier results than the gradient. This noise is most objectionable in smooth areas, where it tends to be more visible. The gradient has a stronger response in areas of significant intensity transitions (ramps and steps) than does the Laplacian. The response of the gradient to noise and fine detail is lower than the Laplacian's and can be lowered further by smoothing the gradient with a lowpass filter. The idea, then, is to smooth the gradient and multiply it by the Laplacian image. In this context, we may view the smoothed gradient as a mask image. The product will preserve details in the strong areas, while reducing noise in the relatively flat areas. This process can be interpreted roughly as combining the best features of the Laplacian and the gradient. The result is added to the original to obtain a final sharpened image.

Figure 3.57(d) shows the Sobel gradient of the original image, computed using Eq. (3-59). Components g_x and g_y were obtained using the kernels in Figs. 3.50(d) and (e), respectively. As expected, the edges are much more dominant in this image than in the Laplacian image. The smoothed gradient image in Fig. 3.57(e) was obtained by using a box filter of size 5×5 . The fact that Figs. 3.57(d) and (e) are much brighter than Fig. 3.57(b) is further evidence that the gradient of an image with significant edge content has values that are higher in general than in a Laplacian image.

Figure 3.57(f) shows the product of the Laplacian and smoothed gradient image. Note the dominance of the strong edges and the relative lack of visible noise, which is the reason for masking the Laplacian with a smoothed gradient image. Adding the product image to the original resulted in the sharpened image in Fig. 3.57(g). The increase in sharpness of detail in this image over the original is evident in most parts of the image, including the ribs, spinal cord, pelvis, and skull. This type of improvement would not have been possible by using the Laplacian or the gradient alone.

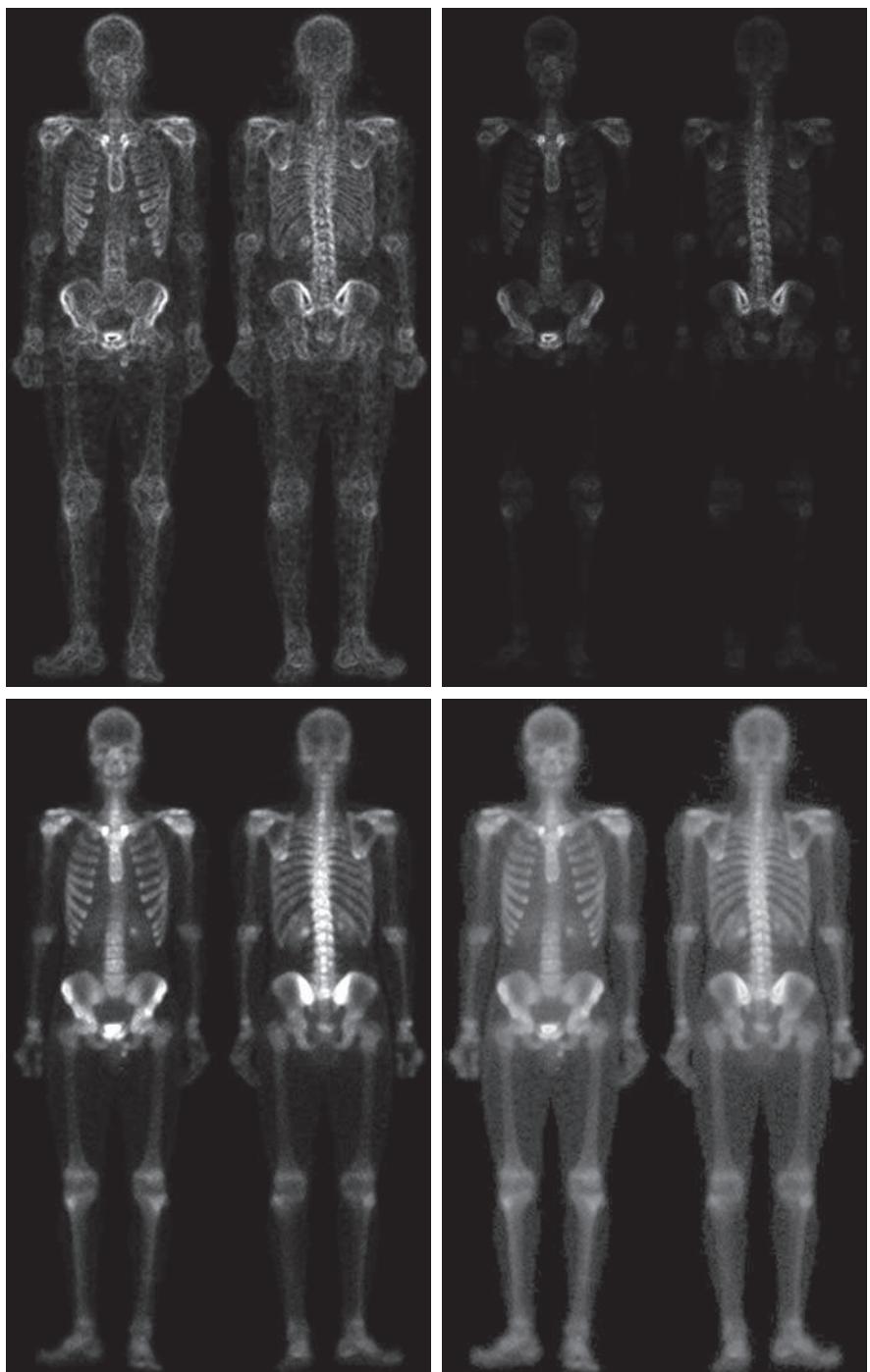
The sharpening procedure just discussed did not affect in an appreciable way the dynamic range of the intensity levels in an image. Thus, the final step in our enhancement task is to increase the dynamic range of the sharpened image. As we discussed in some detail in Sections 3.2 and 3.3, there are several intensity transformation functions that can accomplish this objective. Histogram processing is not a good approach on images whose histograms are characterized by dark and light components, which is the case here. The dark characteristics of the images with which we are dealing lend themselves much better to a power-law transformation. Because we wish to spread the intensity levels, the value of γ in Eq. (3-5) has to be less than 1. After a few trials with this equation, we arrived at the result in Fig. 3.57(h), obtained with $\gamma = 0.5$ and $c = 1$. Comparing this image with Fig. 3.57(g), we note that significant new detail is visible in Fig. 3.57(h). The areas around the wrists, hands, ankles, and feet are good examples of this. The skeletal bone structure also is much more pronounced, including the arm and leg bones. Note the faint definition of the outline of the body, and of body tissue. Bringing out detail of this nature by expanding the dynamic range of the intensity levels also enhanced noise, but Fig. 3.57(h) is a significant visual improvement over the original image.

e f
g h

FIGURE 3.57

(Continued)

- (e) Sobel image smoothed with a 5×5 box filter.
(f) Mask image formed by the product of (b) and (e).
(g) Sharpened image obtained by the adding images (a) and (f).
(h) Final result obtained by applying a power-law transformation to (g). Compare images (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)



Summary, References, and Further Reading

The material in this chapter is representative of current techniques used for intensity transformations and spatial filtering. The topics were selected for their value as fundamental material that would serve as a foundation in an evolving field. Although most of the examples used in this chapter deal with image enhancement, the techniques presented are perfectly general, and you will encounter many of them again throughout the remaining chapters in contexts unrelated to enhancement.

The material in Section 3.1 is from Gonzalez [1986]. For additional reading on the material in Section 3.2, see Schowengerdt [2006] and Poyton [1996]. Early references on histogram processing (Section 3.3) are Gonzalez and Fitts [1977], and Woods and Gonzalez [1981]. Stark [2000] gives some interesting generalizations of histogram equalization for adaptive contrast enhancement.

For complementary reading on linear spatial filtering (Sections 3.4-3.7), see Jain [1989], Rosenfeld and Kak [1982], Schowengerdt [2006], Castleman [1996], and Umbaugh [2010]. For an interesting approach for generating Gaussian kernels with integer coefficients see Padfield [2011]. The book by Pitas and Venetsanopoulos [1990] is a good source for additional reading on median and other nonlinear spatial filters.

For details on the software aspects of many of the examples in this chapter, see Gonzalez, Woods, and Eddins [2009].

Problems

Solutions to the problems marked with an asterisk () are in the DIP4E Student Support Package (consult the book website: www.ImageProcessingPlace.com).*

- 3.1** Give a single intensity transformation function for spreading the intensities of an image so the lowest intensity is 0 and the highest is $L - 1$.

- 3.2** Do the following:

(a)* Give a continuous function for implementing the contrast stretching transformation in Fig. 3.2(a). In addition to m , your function must include a parameter, E , for controlling the slope of the function as it transitions from low to high intensity values. Your function should be normalized so that its minimum and maximum values are 0 and 1, respectively.

(b) Sketch a family of transformations as a function of parameter E , for a fixed value $m = L/2$, where L is the number of intensity levels in the image..

- 3.3** Do the following:

(a)* Propose a set of intensity-slicing transformation functions capable of producing all the individual bit planes of an 8-bit monochrome image. For example, applying to an image a transformation function with the property $T(r) = 0$ if r is 0 or even, and $T(r) = 1$ if r is odd, produces an image of the least signifi-

cant bit plane (see Fig. 3.13). (*Hint:* Use an 8-bit truth table to determine the form of each transformation function.)

- (b) How many intensity transformation functions would there be for 16-bit images?
- (c) Is the basic approach in (a) limited to images in which the number of intensity levels is an integer power of 2, or is the method general for any number of integer intensity levels?
- (d) If the method is general, how would it be different from your solution in (a)?

- 3.4** Do the following:

- (a) Propose a method for extracting the bit planes of an image based on converting the value of its pixels to binary.
- (b) Find all the bit planes of the following 4-bit image:

0	1	8	6
2	2	1	1
1	15	14	12
3	6	9	10

- 3.5** In general:

- (a)* What effect would setting to zero the lower-

- order bit planes have on the histogram of an image?
- (b)** What would be the effect on the histogram if we set to zero the higher-order bit planes instead?
- 3.6** Explain why the discrete histogram equalization technique does not yield a flat histogram in general.
- 3.7** Suppose that a digital image is subjected to histogram equalization. Show that a second pass of histogram equalization (on the histogram-equalized image) will produce exactly the same result as the first pass.
- 3.8** Assuming continuous values, show by an example that it is possible to have a case in which the transformation function given in Eq. (3-11) satisfies conditions (a) and (b) discussed in Section 3.3, but its inverse may fail condition (a').
- 3.9** Do the following:
- (a)** Show that the discrete transformation function given in Eq. (3-15) for histogram equalization satisfies conditions (a) and (b) stated at the beginning of Section 3.3.
 - (b)*** Show that the inverse discrete transformation in Eq. (3-16) satisfies conditions (a') and (b) in Section 3.3 *only if* none of the intensity levels r_k , $k = 0, 1, 2, \dots, L - 1$, are missing in the original image.
- 3.10** Two images, $f(x, y)$ and $g(x, y)$ have unnormalized histograms h_f and h_g . Give the conditions (on the values of the pixels in f and g) under which you can determine the histograms of images formed as follows:
- (a)*** $f(x, y) + g(x, y)$
 - (b)** $f(x, y) - g(x, y)$
 - (c)** $f(x, y) \times g(x, y)$
 - (d)** $f(x, y) \div g(x, y)$
- Show how the histograms would be formed in each case. The arithmetic operations are element-wise operations, as defined in Section 2.6.
- 3.11** Assume continuous intensity values, and suppose that the intensity values of an image have the PDF $p_r(r) = 2r/(L-1)^2$ for $0 \leq r \leq L-1$, and $p_r(r) = 0$ for other values of r .
- (a)*** Find the transformation function that will map the input intensity values, r , into values, s , of a histogram-equalized image.
- (b)*** Find the transformation function that (when applied to the histogram-equalized intensities, s) will produce an image whose intensity PDF is $p_z(z) = 3z^2/(L-1)^3$ for $0 \leq z \leq L-1$ and $p_z(z) = 0$ for other values of z .
- (c)** Express the transformation function from (b) directly in terms of r , the intensities of the input image.
- 3.12** An image with intensities in the range $[0, 1]$ has the PDF, $p_r(r)$, shown in the following figure. It is desired to transform the intensity levels of this image so that they will have the specified $p_z(z)$ shown in the figure. Assume continuous quantities, and find the transformation (expressed in terms of r and z) that will accomplish this.
-
- 3.13*** In Fig. 3.25(b), the transformation function labeled (2) [$G^{-1}(s_k)$ from Eq. (3-23)] is the mirror image of (1) [$G(z_q)$ in Eq. (3-21)] about a line joining the two end points. Does this property always hold for these two transformation functions? Explain.
- 3.14*** The local histogram processing method discussed in Section 3.3 requires that a histogram be computed at each neighborhood location. Propose a method for updating the histogram from one neighborhood to the next, rather than computing a new histogram each time.
- 3.15** What is the behavior of Eq. (3-35) when $a = b = 0$? Explain.
- 3.16** You are given a computer chip that is capable of performing linear filtering in real time, but you are not told whether the chip performs correlation or convolution. Give the details of a test you would perform to determine which of the two operations the chip performs.
- 3.17*** We mentioned in Section 3.4 that to perform con-

volution we rotate the kernel by 180°. The rotation is “built” into Eq. (3-35). Figure 3.28 corresponds to correlation. Draw the part of the figure enclosed by the large ellipse, but with w rotated 180°. Expand Eq. (3-35) for a general 3×3 kernel and show that the result of your expansion corresponds to your figure. This shows graphically that convolution and correlation differ by the rotation of the kernel.

- 3.18** You are given the following kernel and image:

$$w = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a)* Give a sketch of the area encircled by the large ellipse in Fig. 3.28 when the kernel is centered at point (2,3) (2nd row, 3rd col) of the image shown above. Show specific values of w and f .

(b)* Compute the convolution $w \star f$ using the *minimum* zero padding needed. Show the details of your computations when the kernel is centered on point (2,3) of f , and then show the final full convolution result.

(c) Repeat (b), but for correlation, $w \star f$.

- 3.19*** Prove the validity of Eqs. (3-36) and (3-37).

- 3.20** The kernel, w , in Problem 3.18 is separable.

(a)* By inspection, find two kernels, w_1 and w_2 so that $w = w_1 \star w_2$.

(b) Using the image in Problem 3.18, compute $w_1 \star f$ using the *minimum* zero padding (see Fig. 3.30). Show the details of your computation when the kernel is centered at point (2,3) (2nd row, 3rd col) of f and then show the full convolution.

(c) Compute the convolution of w_2 with the result from (b). Show the details of your computation when the kernel is centered at point (3,3) of the result from (b), and then show the full convolution. Compare with the result in Problem 3.18(b).

- 3.21** Given the following kernel and image:

$$w = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad f = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(a) Give the convolution of the two.

(b) Does your result have a bias?

- 3.22** Answer the following:

(a)* If $\mathbf{v} = [1 \ 2 \ 1]^T$ and $\mathbf{w}^T = [2 \ 1 \ 1 \ 3]$, is the kernel formed by $\mathbf{v}\mathbf{w}^T$ separable?

(b) The following kernel is separable. Find w_1 and w_2 such that $w = w_1 \star w_2$.

$$w = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 6 & 2 \end{bmatrix}$$

- 3.23** Do the following:

(a)* Show that the Gaussian kernel, $G(s,t)$, in Eq. (3-45) is separable. (*Hint:* Read the first paragraph in the discussion of separable filter kernels in Section 3.4.)

(b) Because G is separable and circularly symmetric, it can be expressed in the form $G = \mathbf{v}\mathbf{v}^T$. Assume that the kernel form in Eq. (3-46) is used, and that the function is sampled to yield an $m \times m$ kernel. What is \mathbf{v} in this case?

- 3.24*** Show that the product of a column vector with a row vector is equivalent to the 2-D convolution of the two vectors. The vectors do not have to be of the same length. You may use a graphical approach (as in Fig. 3.30) to support the explanation of your proof.

- 3.25** Given K , 1-D Gaussian kernels, g_1, g_2, \dots, g_K , with arbitrary means and standard deviations:

(a)* Determine what the entries in the third column of Table 3.6 would be for the product $g_1 \times g_2 \times \dots \times g_K$.

(b) What would the fourth column look like for the convolution $g_1 \star g_2 \star \dots \star g_K$?

(*Hint:* It is easier to work with the variance; the standard deviation is just the square root of your result.)

- 3.26** The two images shown in the following figure are quite different, but their histograms are the same. Suppose that each image is blurred using a 3×3 box kernel.

(a)* Would the histograms of the blurred images still be equal? Explain.



(b) If your answer is no, either sketch the two histograms or give two tables detailing the histogram components.

- 3.27** An image is filtered four times using a Gaussian kernel of size 3×3 with a standard deviation of 1.0. Because of the associative property of convolution, we know that equivalent results can be obtained using a single Gaussian kernel formed by convolving the individual kernels.

(a)* What is the size of the single Gaussian kernel?

(b) What is its standard deviation?

- 3.28** An image is filtered with three Gaussian lowpass kernels of sizes 3×3 , 5×5 , and 7×7 , and standard deviations 1.5, 2, and 4, respectively. A composite filter, w , is formed as the convolution of these three filters.

(a)* Is the resulting filter Gaussian? Explain.

(b) What is its standard deviation?

(c) What is its size?

- 3.29*** Discuss the limiting effect of repeatedly filtering an image with a 3×3 lowpass filter kernel. You may ignore border effects.

- 3.30** In Fig. 3.42(b) the corners of the estimated shading pattern appear darker or lighter than their surrounding areas. Explain the reason for this.

- 3.31*** An image is filtered with a kernel whose coefficients sum to 1. Show that the sum of the pixel values in the original and filtered images is the same.

- 3.32** An image is filtered with a kernel whose coeffi-

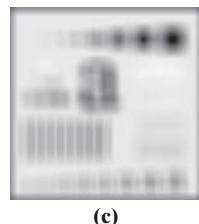
cients sum to 0. Show that the sum of the pixel values in the filtered image also is 0.

- 3.33** A single point of light can be modeled by a digital image consisting of all 0's, with a 1 in the location of the point of light. If you view a single point of light through a defocused lens, it will appear as a fuzzy blob whose size depends on the amount by which the lens is defocused. We mentioned in Section 3.5 that filtering an image with a box kernel is a poor model for a defocused lens, and that a better approximation is obtained with a Gaussian kernel. Using the single-point-of-light analogy, explain why this is so.

- 3.34** In the original image used to generate the three blurred images shown, the vertical bars are 5 pixels wide, 100 pixels high, and their separation is 20 pixels. The image was blurred using square box kernels of sizes 23, 25, and 45 elements on the side, respectively. The vertical bars on the left, lower part of (a) and (c) are blurred, but a clear separation exists between them.



(a) (b)



(c)

However, the bars have merged in image (b), despite the fact that the kernel used to generate this image is much smaller than the kernel that produced image (c). Explain the reason for this.

- 3.35** Consider an application such as in Fig. 3.41, in which it is desired to eliminate objects smaller than those enclosed by a square of size $q \times q$ pixels. Suppose that we want to reduce the average

intensity of those objects to one-tenth of their original average value. In this way, their intensity will be closer to the intensity of the background and they can be eliminated by thresholding. Give the (odd) size of the smallest box kernel that will yield the desired reduction in average intensity in only one pass of the kernel over the image.

- 3.36** With reference to order-statistic filters (see Section 3.5):

(a)* We mentioned that isolated clusters of dark or light (with respect to the background) pixels whose area is less than one-half the area of a median filter are forced to the median value of the neighbors by the filter. Assume a filter of size $n \times n$ (n odd) and explain why this is so.

(b) Consider an image having various sets of pixel clusters. Assume that all points in a cluster are lighter or darker than the background (but not both simultaneously in the same cluster), and that the area of each cluster is less than or equal to $n^2/2$. In terms of n , under what condition would one or more of these clusters cease to be isolated in the sense described in part (a)?

- 3.37** Do the following:

(a)* Develop a procedure for computing the median of an $n \times n$ neighborhood.

(b) Propose a technique for updating the median as the center of the neighborhood is moved from pixel to pixel.

- 3.38** In a given application, a smoothing kernel is applied to input images to reduce noise, then a Laplacian kernel is applied to enhance fine details. Would the result be the same if the order of these operations is reversed?

- 3.39*** Show that the Laplacian defined in Eq. (3-50) is isotropic (invariant to rotation). Assume continuous quantities. From Table 2.3, coordinate rotation by an angle θ is given by

$$x' = x \cos \theta - y \sin \theta \text{ and } y' = x \sin \theta + y \cos \theta$$

where (x, y) and (x', y') are the unrotated and rotated coordinates, respectively.

- 3.40*** You saw in Fig. 3.46 that the Laplacian with a -8

in the center yields sharper results than the one with a -4 in the center. Explain the reason why.

- 3.41*** Give a 3×3 kernel for performing unsharp masking in a single pass through an image. Assume that the average image is obtained using a box filter of size 3×3 .

- 3.42** Show that subtracting the Laplacian from an image gives a result that is proportional to the unsharp mask in Eq. (3-55). Use the definition for the Laplacian given in Eq. (3-53).

- 3.43** Do the following:

(a)* Show that the magnitude of the gradient given in Eq. (3-58) is an isotropic operation (see the statement of Problem 3.39).

(b) Show that the isotropic property is lost in general if the gradient is computed using Eq. (3-59).

- 3.44** Are any of the following highpass (sharpening) kernels separable? For those that are, find vectors \mathbf{v} and \mathbf{w} such that \mathbf{vw}^T equals the kernel(s).

(a) The Laplacian kernels in Figs. 3.45(a) and (b).

(b) The Roberts cross-gradient kernels shown in Figs. 3.50(b) and (c).

(c)* The Sobel kernels in Figs. 3.50(d) and (e).

- 3.45** In a character recognition application, text pages are reduced to binary using a thresholding transformation function of the form in Fig. 3.2(b). This is followed by a procedure that thins the characters until they become strings of binary 1's on a background of 0's. Due to noise, binarization and thinning result in broken strings of characters with gaps ranging from 1 to 3 pixels. One way to "repair" the gaps is to run a smoothing kernel over the binary image to blur it, and thus create bridges of nonzero pixels between gaps.

(a)* Give the (odd) size of the smallest box kernel capable of performing this task.

(b) After bridging the gaps, the image is thresholded to convert it back to binary form. For your answer in (a), what is the minimum value of the threshold required to accomplish this, without causing the segments to break up again?

- 3.46** A manufacturing company purchased an imaging system whose function is to either smooth or sharpen images. The results of using the system on the manufacturing floor have been poor, and the plant manager suspects that the system is not smoothing and sharpening images the way it should. You are hired as a consultant to determine if the system is performing these functions properly. How would you determine if the system is working correctly? (*Hint:* Study the statements of Problems 3.31 and 3.32).
- 3.47** A CCD TV camera is used to perform a long-term study by observing the same area 24 hours a day, for 30 days. Digital images are captured and transmitted to a central location every 5 minutes. The illu-

mination of the scene changes from natural daylight to artificial lighting. At no time is the scene without illumination, so it is always possible to obtain an acceptable image. Because the range of illumination is such that it is always in the linear operating range of the camera, it is decided not to employ any compensating mechanisms on the camera itself. Rather, it is decided to use image processing techniques to post-process, and thus normalize, the images to the equivalent of constant illumination. Propose a method to do this. You are at liberty to use any method you wish, but state clearly all the assumptions you made in arriving at your design.

This page intentionally left blank

4

Filtering in the Frequency Domain

Filter: A device or material for suppressing or minimizing waves or oscillations of certain frequencies.

Frequency: The number of times that a periodic function repeats the same sequence of values during a unit variation of the independent variable.

Webster's New Collegiate Dictionary

Preview

After a brief historical introduction to the Fourier transform and its importance in image processing, we start from basic principles of function sampling, and proceed step-by-step to derive the one- and two-dimensional discrete Fourier transforms. Together with convolution, the Fourier transform is a staple of frequency-domain processing. During this development, we also touch upon several important aspects of sampling, such as aliasing, whose treatment requires an understanding of the frequency domain and thus are best covered in this chapter. This material is followed by a formulation of filtering in the frequency domain, paralleling the spatial filtering techniques discussed in Chapter 3. We conclude the chapter with a derivation of the equations underlying the fast Fourier transform (FFT), and discuss its computational advantages. These advantages make frequency-domain filtering practical and, in many instances, superior to filtering in the spatial domain.

Upon completion of this chapter, readers should:

- Understand the meaning of frequency domain filtering, and how it differs from filtering in the spatial domain.
- Be familiar with the concepts of sampling, function reconstruction, and aliasing.
- Understand convolution in the frequency domain, and how it is related to filtering.
- Know how to obtain frequency domain filter functions from spatial kernels, and vice versa.
- Be able to construct filter transfer functions directly in the frequency domain.
- Understand why image padding is important.
- Know the steps required to perform filtering in the frequency domain.
- Understand when frequency domain filtering is superior to filtering in the spatial domain.
- Be familiar with other filtering techniques in the frequency domain, such as unsharp masking and homomorphic filtering.
- Understand the origin and mechanics of the fast Fourier transform, and how to use it effectively in image processing.

4.1 BACKGROUND

We begin the discussion with a brief outline of the origins of the Fourier transform and its impact on countless branches of mathematics, science, and engineering.

A BRIEF HISTORY OF THE FOURIER SERIES AND TRANSFORM

The French mathematician Jean Baptiste Joseph Fourier was born in 1768 in the town of Auxerre, about midway between Paris and Dijon. The contribution for which he is most remembered was outlined in a memoir in 1807, and later published in 1822 in his book, *La Théorie Analytique de la Chaleur* (*The Analytic Theory of Heat*). This book was translated into English 55 years later by Freeman (see Freeman [1878]). Basically, Fourier's contribution in this field states that any periodic function can be expressed as the sum of sines and/or cosines of different frequencies, each multiplied by a different coefficient (we now call this sum a *Fourier series*). It does not matter how complicated the function is; if it is periodic and satisfies some mild mathematical conditions, it can be represented by such a sum. This is taken for granted now but, at the time it first appeared, the concept that complicated functions could be represented as a sum of simple sines and cosines was not at all intuitive (see Fig. 4.1). Thus, it is not surprising that Fourier's ideas were met initially with skepticism.

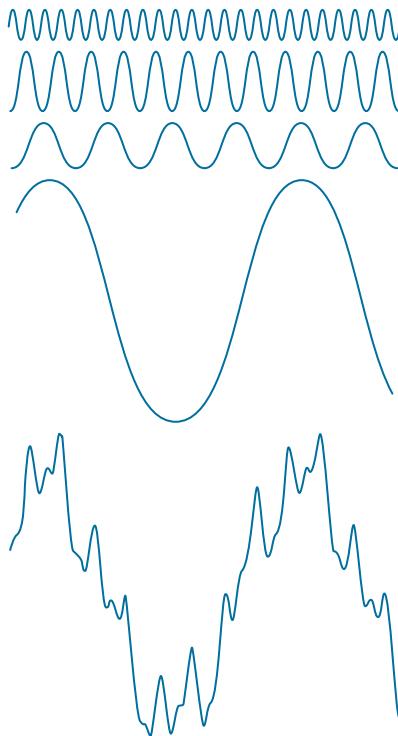
Functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weighting function. The formulation in this case is the *Fourier transform*, and its utility is even greater than the Fourier series in many theoretical and applied disciplines. Both representations share the important characteristic that a function, expressed in either a Fourier series or transform, can be reconstructed (recovered) completely via an inverse process, with no loss of information. This is one of the most important characteristics of these representations because it allows us to work in the *Fourier domain* (generally called the *frequency domain*) and then return to the original domain of the function without losing any information. Ultimately, it is the utility of the Fourier series and transform in solving practical problems that makes them widely studied and used as fundamental tools.

The initial application of Fourier's ideas was in the field of heat diffusion, where they allowed formulation of differential equations representing heat flow in such a way that solutions could be obtained for the first time. During the past century, and especially in the past 60 years, entire industries and academic disciplines have flourished as a result of Fourier's initial ideas. The advent of digital computers and the "discovery" of a fast Fourier transform (FFT) algorithm in the early 1960s revolutionized the field of signal processing. These two core technologies allowed for the first time practical processing of a host of signals of exceptional importance, ranging from medical monitors and scanners to modern electronic communications.

As you learned in Section 3.4, it takes on the order of $MNmn$ operations (multiplications and additions) to filter an $M \times N$ image with a kernel of size $m \times n$ elements. If the kernel is separable, the number of operations is reduced to $MN(m + n)$. In Section 4.11, you will learn that it takes on the order of $2MN \log_2 MN$ operations to perform the equivalent filtering process in the frequency domain, where the 2 in front arises from the fact that we have to compute a forward and an inverse FFT.

FIGURE 4.1

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



To get an idea of the relative computational advantages of filtering in the frequency versus the spatial domain, consider square images and kernels, of sizes $M \times M$ and $m \times m$, respectively. The *computational advantage* (as a function of kernel size) of filtering one such image with the FFT as opposed to using a nonseparable kernel is defined as

$$\begin{aligned} C_n(m) &= \frac{M^2 m^2}{2M^2 \log_2 M^2} \\ &= \frac{m^2}{4 \log_2 M} \end{aligned} \tag{4-1}$$

If the kernel is separable, the advantage becomes

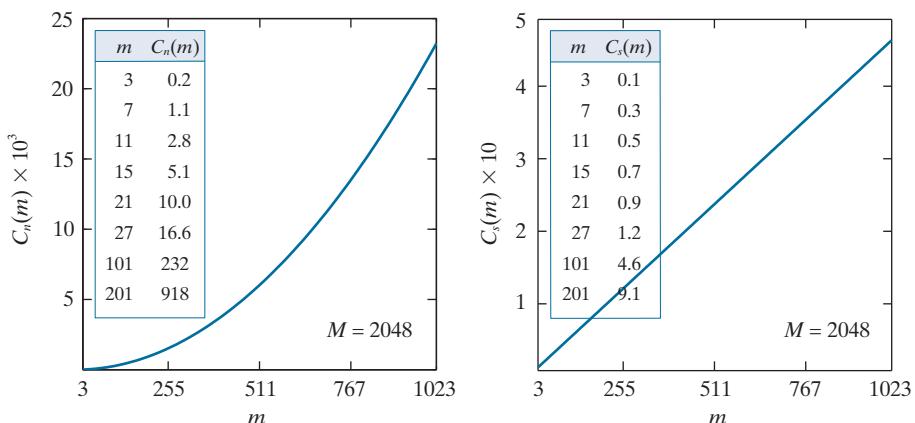
$$\begin{aligned} C_s(m) &= \frac{2M^2 m}{2M^2 \log_2 M^2} \\ &= \frac{m}{2 \log_2 M} \end{aligned} \tag{4-2}$$

In either case, when $C(m) > 1$ the advantage (in terms of fewer computations) belongs to the FFT approach; otherwise the advantage favors spatial filtering.

a b

FIGURE 4.2

- (a) Computational advantage of the FFT over non-separable spatial kernels.
 (b) Advantage over separable kernels. The numbers for $C(m)$ in the inset tables are not to be multiplied by the factors of 10 shown for the curves.



The computational advantages given by Eqs. (4-1) and (4-2) do not take into account the fact that the FFT performs operations between complex numbers, and other secondary (but small in comparison) computations discussed later in the chapter. Thus, comparisons should be interpreted only as guidelines.

Figure 4.2(a) shows a plot of $C_n(m)$ as a function of m for an image of intermediate size ($M = 2048$). The inset table shows a more detailed look for smaller kernel sizes. As you can see, the FFT has the advantage for kernels of sizes 7×7 and larger. The advantage grows rapidly as a function of m , being over 200 for $m = 101$, and close to 1000 for $m = 201$. To give you a feel for the meaning of this advantage, if filtering a bank of images of size 2048×2048 takes 1 minute with the FFT, it would take on the order of 17 hours to filter the same set of images with a nonseparable kernel of size 201×201 elements. This is a significant difference, and is a clear indicator of the importance of frequency-domain processing using the FFT.

In the case of separable kernels, the computational advantage is not as dramatic, but it is still meaningful. The “cross over” point now is around $m = 27$, and when $m = 101$ the difference between frequency- and spatial-domain filtering is still manageable. However, you can see that with $m = 201$ the advantage of using the FFT approaches a factor of 10, which begins to be significant. Note in both graphs that the FFT is an overwhelming favorite for large spatial kernels.

Our focus in the sections that follow is on the Fourier transform and its properties. As we progress through this chapter, it will become evident that Fourier techniques are useful in a broad range of image processing applications. We conclude the chapter with a discussion of the FFT.

ABOUT THE EXAMPLES IN THIS CHAPTER

As in Chapter 3, most of the image filtering examples in this chapter deal with image enhancement. For example, smoothing and sharpening are traditionally associated with image enhancement, as are techniques for contrast manipulation. By its very nature, beginners in digital image processing find enhancement to be interesting and relatively simple to understand. Therefore, using examples from image enhancement in this chapter not only saves having an extra chapter in the book but, more importantly, is an effective tool for introducing newcomers to filtering techniques in the frequency domain. We will use frequency domain processing methods for other applications in Chapters 5, 7, 8, 10, and 11.

4.2 PRELIMINARY CONCEPTS

We pause briefly to introduce several of the basic concepts that underlie the material in later sections.

COMPLEX NUMBERS

A complex number, C , is defined as

$$C = R + jI \quad (4-3)$$

where R and I are real numbers and $j = \sqrt{-1}$. Here, R denotes the *real part* of the complex number and I its *imaginary part*. Real numbers are a subset of complex numbers in which $I = 0$. The *conjugate* of a complex number C , denoted C^* , is defined as

$$C^* = R - jI \quad (4-4)$$

Complex numbers can be viewed geometrically as points on a plane (called the *complex plane*) whose abscissa is the *real axis* (values of R) and whose ordinate is the *imaginary axis* (values of I). That is, the complex number $R + jI$ is point (R, I) in the coordinate system of the complex plane.

Sometimes it is useful to represent complex numbers in polar coordinates,

$$C = |C|(\cos \theta + j \sin \theta) \quad (4-5)$$

where $|C| = \sqrt{R^2 + I^2}$ is the length of the vector extending from the origin of the complex plane to point (R, I) , and θ is the angle between the vector and the real axis. Drawing a diagram of the real and complex axes with the vector in the first quadrant will show that $\tan \theta = (I/R)$ or $\theta = \arctan(I/R)$. The \arctan function returns angles in the range $[-\pi/2, \pi/2]$. But, because I and R can be positive and negative independently, we need to be able to obtain angles in the full range $[-\pi, \pi]$. We do this by keeping track of the sign of I and R when computing θ . Many programming languages do this automatically via so called *four-quadrant arctangent functions*. For example, MATLAB provides the function `atan2(Imag, Real)` for this purpose.

Using *Euler's formula*,

$$e^{j\theta} = \cos \theta + j \sin \theta \quad (4-6)$$

where $e = 2.71828\dots$, gives the following familiar representation of complex numbers in polar coordinates,

$$C = |C| e^{j\theta} \quad (4-7)$$

where $|C|$ and θ are as defined above. For example, the polar representation of the complex number $1 + j2$ is $\sqrt{5}e^{j\theta}$, where $\theta = 63.4^\circ$ or 1.1 radians. The preceding equations are applicable also to complex functions. A complex function, $F(u)$, of a real variable u , can be expressed as the sum $F(u) = R(u) + jI(u)$, where $R(u)$ and $I(u)$ are the real and imaginary component functions of $F(u)$. As previously noted, the complex conjugate is $F^*(u) = R(u) - jI(u)$, the magnitude is $|F(u)| = [R(u)^2 + I(u)^2]^{1/2}$,

and the angle is $\theta(u) = \arctan[I(u)/R(u)]$. We will return to complex functions several times in the course of this and the next chapter.

FOURIER SERIES

As indicated in the previous section, a function $f(t)$ of a continuous variable, t , that is periodic with a period, T , can be expressed as the sum of sines and cosines multiplied by appropriate coefficients. This sum, known as a *Fourier series*, has the form

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j \frac{2\pi n}{T} t} \quad (4-8)$$

where

$$c_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j \frac{2\pi n}{T} t} dt \quad \text{for } n = 0, \pm 1, \pm 2, \dots \quad (4-9)$$

are the coefficients. The fact that Eq. (4-8) is an expansion of sines and cosines follows from Euler's formula, Eq. (4-6).

IMPULSES AND THEIR SIFTING PROPERTIES

Central to the study of linear systems and the Fourier transform is the concept of an impulse and its sifting property. A *unit impulse* of a continuous variable t , located at $t = 0$, and denoted $\delta(t)$, is defined as

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases} \quad (4-10)$$

and is constrained to satisfy the identity

$$\int_{-\infty}^{\infty} \delta(t) dt = 1 \quad (4-11)$$

Physically, if we interpret t as time, an impulse may be viewed as a spike of infinity amplitude and zero duration, having unit area. An impulse has the so-called *sifting property* with respect to integration,

$$\int_{-\infty}^{\infty} f(t) \delta(t) dt = f(0) \quad (4-12)$$

provided that $f(t)$ is continuous at $t = 0$, a condition typically satisfied in practice. Sifting simply yields the *value* of the function $f(t)$ at the *location* of the impulse (i.e., at $t = 0$ in the previous equation). A more general statement of the sifting property involves an impulse located at an arbitrary point, t_0 , denoted as, $\delta(t - t_0)$. In this case,

$$\int_{-\infty}^{\infty} f(t) \delta(t - t_0) dt = f(t_0) \quad (4-13)$$

An impulse is not a function in the usual sense. A more accurate name is a *distribution* or *generalized function*. However, one often finds in the literature the names *impulse function*, *delta function*, and *Dirac delta function*, despite the misnomer.

To *sift* means literally to separate, or to separate out, by putting something through a sieve.

which simply gives the value of the function at the location of the impulse. For example, if $f(t) = \cos(t)$, using the impulse $\delta(t - \pi)$ in Eq. (4-13) yields the result $f(\pi) = \cos(\pi) = -1$. The power of the sifting concept will become evident shortly.

Of particular interest later in this section is an *impulse train*, $s_{\Delta T}(t)$, defined as the sum of infinitely many impulses ΔT units apart:

$$s_{\Delta T}(t) = \sum_{k=-\infty}^{\infty} \delta(t - k\Delta T) \quad (4-14)$$

Figure 4.3(a) shows a single impulse located at $t = t_0$, and Fig. 4.3(b) shows an impulse train. Impulses for continuous variables are denoted by up-pointing arrows to simulate infinite height and zero width. For discrete variables the height is finite, as we will show next.

Let x represent a *discrete* variable. As you learned in Chapter 3, the unit discrete impulse, $\delta(x)$, serves the same purposes in the context of discrete systems as the impulse $\delta(t)$ does when working with continuous variables. It is defined as

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \quad (4-15)$$

Clearly, this definition satisfies the discrete equivalent of Eq. (4-11):

$$\sum_{x=-\infty}^{\infty} \delta(x) = 1 \quad (4-16)$$

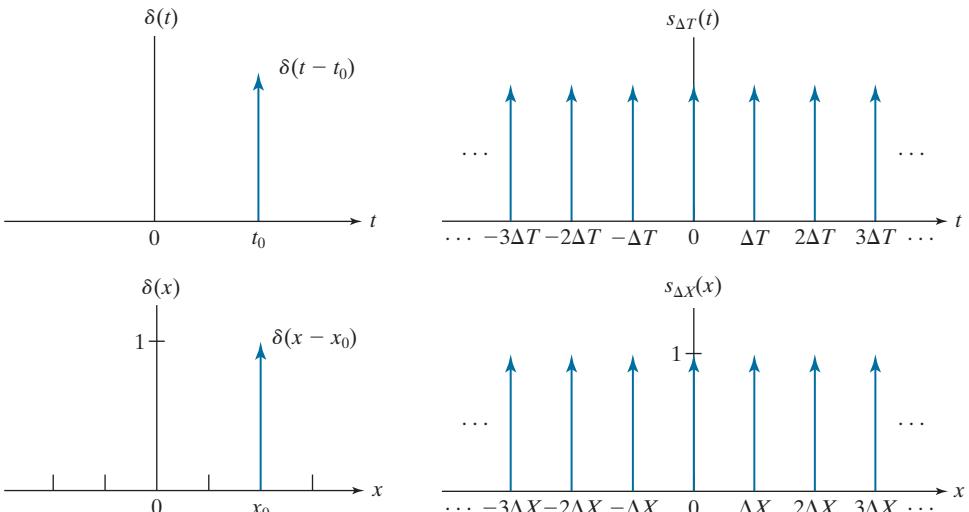
The sifting property for discrete variables has the form

$$\sum_{x=-\infty}^{\infty} f(x)\delta(x) = f(0) \quad (4-17)$$

a	b
c	d

FIGURE 4.3

- (a) Continuous impulse located at $t = t_0$.
- (b) An impulse train consisting of continuous impulses.
- (c) Unit discrete impulse located at $x = x_0$.
- (d) An impulse train consisting of discrete unit impulses.



or, more generally using a discrete impulse located at $x = x_0$ (see Eq. 3-33),

$$\sum_{x=-\infty}^{\infty} f(x)\delta(x - x_0) = f(x_0) \quad (4-18)$$

As before, we see that the sifting property yields the value of the function at the location of the impulse. Figure 4.3(c) shows the unit discrete impulse diagrammatically, and Fig. 4.3(d) shows a train of discrete unit impulses. Unlike its continuous counterpart, the discrete impulse is an ordinary function.

THE FOURIER TRANSFORM OF FUNCTIONS OF ONE CONTINUOUS VARIABLE

The *Fourier transform* of a continuous function $f(t)$ of a continuous variable, t , denoted $\mathfrak{F}\{f(t)\}$, is defined by the equation

$$\mathfrak{F}\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t} dt \quad (4-19)$$

where μ is a continuous variable also.[†] Because t is integrated out, $\mathfrak{F}\{f(t)\}$ is a function only of μ . That is $\mathfrak{F}\{f(t)\} = F(\mu)$; therefore, we write the Fourier transform of $f(t)$ as

$$F(\mu) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t} dt \quad (4-20)$$

Conversely, given $F(\mu)$, we can obtain $f(t)$ back using the *inverse Fourier transform*, written as

$$f(t) = \int_{-\infty}^{\infty} F(\mu)e^{j2\pi\mu t} d\mu \quad (4-21)$$

where we made use of the fact that variable μ is integrated out in the inverse transform and wrote simply $f(t)$, rather than the more cumbersome notation $f(t) = \mathfrak{F}^{-1}\{F(\mu)\}$. Equations (4-20) and (4-21) comprise the so-called *Fourier transform pair*, often denoted as $f(t) \Leftrightarrow F(\mu)$. The double arrow indicates that the expression on the right is obtained by taking the *forward Fourier transform* of the expression on the left, while the expression on the left is obtained by taking the *inverse Fourier transform* of the expression on the right.

Using Euler's formula, we can write Eq. (4-20) as

$$F(\mu) = \int_{-\infty}^{\infty} f(t)[\cos(2\pi\mu t) - j\sin(2\pi\mu t)]dt \quad (4-22)$$

Because t is integrated out in this equation, the only variable left is μ , which is the frequency of the sine and cosine terms.

[†]Conditions for the existence of the Fourier transform are complicated to state in general (Champeney [1987]), but a sufficient condition for its existence is that the integral of the absolute value of $f(t)$, or the integral of the square of $f(t)$, be finite. Existence is seldom an issue in practice, except for idealized signals, such as sinusoids that extend forever. These are handled using generalized impulses. Our primary interest is in the discrete Fourier transform pair which, as you will see shortly, is guaranteed to exist for all finite functions.

If $f(t)$ is real, we see that its transform in general is complex. Note that the Fourier transform is an expansion of $f(t)$ multiplied by sinusoidal terms whose frequencies are determined by the values of μ . Thus, because the only variable left after integration is frequency, we say that the domain of the Fourier transform is the *frequency domain*. We will discuss the frequency domain and its properties in more detail later in this chapter. In our discussion, t can represent any continuous variable, and the units of the frequency variable μ depend on the units of t . For example, if t represents time in seconds, the units of μ are cycles/sec or Hertz (Hz). If t represents distance in meters, then the units of μ are cycles/meter, and so on. In other words, the units of the frequency domain are cycles per unit of the independent variable of the input function.

EXAMPLE 4.1: Obtaining the Fourier transform of a simple continuous function.

The Fourier transform of the function in Fig. 4.4(a) follows from Eq. (4-20):

$$\begin{aligned} F(\mu) &= \int_{-\infty}^{\infty} f(t) e^{-j2\pi\mu t} dt = \int_{-W/2}^{W/2} A e^{-j2\pi\mu t} dt \\ &= \frac{-A}{j2\pi\mu} [e^{-j2\pi\mu t}]_{-W/2}^{W/2} = \frac{-A}{j2\pi\mu} [e^{-j\pi\mu W} - e^{j\pi\mu W}] \\ &= \frac{A}{j2\pi\mu} [e^{j\pi\mu W} - e^{-j\pi\mu W}] \\ &= AW \frac{\sin(\pi\mu W)}{(\pi\mu W)} \end{aligned}$$

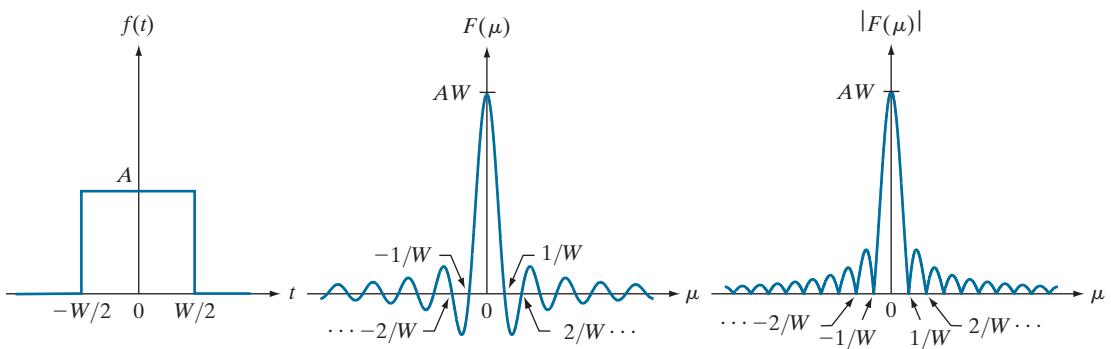


FIGURE 4.4 (a) A box function, (b) its Fourier transform, and (c) its spectrum. All functions extend to infinity in both directions. Note the inverse relationship between the width, W , of the function and the zeros of the transform.

where we used the trigonometric identity $\sin \theta = (e^{j\theta} - e^{-j\theta})/2j$. In this case, the complex terms of the Fourier transform combined nicely into a real sine function. The result in the last step of the preceding expression is known as the *sinc* function, which has the general form

$$\text{sinc}(m) = \frac{\sin(\pi m)}{(\pi m)} \quad (4-23)$$

where $\text{sinc}(0) = 1$ and $\text{sinc}(m) = 0$ for all other *integer* values of m . Figure 4.4(b) shows a plot of $F(\mu)$.

In general, the Fourier transform contains complex terms, and it is customary for display purposes to work with the magnitude of the transform (a real quantity), which is called the *Fourier spectrum* or the *frequency spectrum*:

$$|F(\mu)| = AW \left| \frac{\sin(\pi\mu W)}{(\pi\mu W)} \right|$$

Figure 4.4(c) shows a plot of $|F(\mu)|$ as a function of frequency. The key properties to note are (1) that the locations of the zeros of both $F(\mu)$ and $|F(\mu)|$ are inversely proportional to the width, W , of the “box” function; (2) that the height of the lobes decreases as a function of distance from the origin; and (3) that the function extends to infinity for both positive and negative values of μ . As you will see later, these properties are quite helpful in interpreting the spectra of two dimensional Fourier transforms of images.

EXAMPLE 4.2: Fourier transform of an impulse and an impulse train.

The Fourier transform of a unit impulse located at the origin follows from Eq. (4-20):

$$\Im\{\delta(t)\} = F(\mu) = \int_{-\infty}^{\infty} \delta(t) e^{-j2\pi\mu t} dt = \int_{-\infty}^{\infty} e^{-j2\pi\mu t} \delta(t) dt = e^{-j2\pi\mu}$$

where we used the sifting property from Eq. (4-12). Thus, we see that the Fourier transform of an impulse located at the origin of the spatial domain is a constant in the frequency domain (we discussed this briefly in Section 3.4 in connection with Fig. 3.30).

Similarly, the Fourier transform of an impulse located at $t = t_0$ is

$$\Im\{\delta(t - t_0)\} = F(\mu) = \int_{-\infty}^{\infty} \delta(t - t_0) e^{-j2\pi\mu t} dt = \int_{-\infty}^{\infty} e^{-j2\pi\mu t} \delta(t - t_0) dt = e^{-j2\pi\mu t_0}$$

where we used the sifting property from Eq. (4-13). The term $e^{-j2\pi\mu t_0}$ represents a unit circle centered on the origin of the complex plane, as you can easily see by using Euler’s formula to expand the exponential into its sine and cosine components.

In Section 4.3, we will use the Fourier transform of a periodic impulse train. Obtaining this transform is not as straightforward as we just showed for individual impulses. However, understanding how to derive the transform of an impulse train is important, so we take the time to derive it here. We start by noting that the only basic difference in the form of Eqs. (4-20) and (4-21) is the sign of the exponential. Thus, if a function $f(t)$ has the Fourier transform $F(\mu)$, then evaluating this function at t , $F(t)$, must have the transform $f(-\mu)$. Using this *symmetry* property and given, as we showed above, that the Fourier transform of an impulse $\delta(t - t_0)$ is $e^{-j2\pi\mu t_0}$, it follows that the function $e^{-j2\pi\mu t_0}$ has the transform

$\delta(-\mu - t_0)$. By letting $-t_0 = a$, it follows that the transform of $e^{j2\pi at}$ is $\delta(-\mu + a) = \delta(\mu - a)$, where the last step is true because δ is zero unless $\mu = a$, which is the same condition for either $\delta(-\mu + a)$ or $\delta(\mu - a)$.

The impulse train $s_{\Delta T}(t)$ in Eq. (4-14) is periodic with period ΔT , so it can be expressed as a Fourier series:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{\Delta T} t}$$

where

$$c_n = \frac{1}{\Delta T} \int_{-\Delta T/2}^{\Delta T/2} s_{\Delta T}(t) e^{-j\frac{2\pi n}{\Delta T} t} dt$$

With reference to Fig. 4.3(b), we see that the integral in the interval $[-\Delta T/2, \Delta T/2]$ encompasses only the impulse located at the origin. Therefore, the preceding equation becomes

$$c_n = \frac{1}{\Delta T} \int_{-\Delta T/2}^{\Delta T/2} \delta(t) e^{-j\frac{2\pi n}{\Delta T} t} dt = \frac{1}{\Delta T} e^0 = \frac{1}{\Delta T}$$

where we used the sifting property of $\delta(t)$. The Fourier series then becomes

$$s_{\Delta T}(t) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T} t}$$

Our objective is to obtain the Fourier transform of this expression. Because summation is a linear process, obtaining the Fourier transform of a sum is the same as obtaining the sum of the transforms of the individual components of the sum. These components are exponentials, and we established earlier in this example that

$$\Im\left\{e^{j\frac{2\pi n}{\Delta T} t}\right\} = \delta\left(\mu - \frac{n}{\Delta T}\right)$$

So, $S(\mu)$, the Fourier transform of the periodic impulse train, is

$$S(\mu) = \Im\{s_{\Delta T}(t)\} = \Im\left\{\frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T} t}\right\} = \frac{1}{\Delta T} \Im\left\{\sum_{n=-\infty}^{\infty} e^{j\frac{2\pi n}{\Delta T} t}\right\} = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right)$$

This fundamental result tells us that the Fourier transform of an impulse train with period ΔT is also an impulse train, whose period is $1/\Delta T$. This *inverse proportionality* between the periods of $s_{\Delta T}(t)$ and $S(\mu)$ is analogous to what we found in Fig. 4.4 in connection with a box function and its transform. This inverse relationship plays a fundamental role in the remainder of this chapter.

As in Section 3.4, the fact that convolution of a function with an impulse shifts the origin of the function to the location of the impulse is also true for continuous convolution. (See Figs. 3.29 and 3.30.)

CONVOLUTION

We showed in Section 3.4 that convolution of two functions involves flipping (rotating by 180°) one function about its origin and sliding it past the other. At each displacement in the sliding process, we perform a computation, which, for discrete variables, is a sum of products [see Eq. (3-35)]. In the present discussion, we are

interested in the convolution of two *continuous* functions, $f(t)$ and $h(t)$, of one continuous variable, t , so we have to use integration instead of a summation. The convolution of these two functions, denoted as before by the operator \star , is defined as

$$(f \star h)(t) = \int_{-\infty}^{\infty} f(\tau)h(t - \tau)d\tau \quad (4-24)$$

where the minus sign accounts for the flipping just mentioned, t is the *displacement* needed to slide one function past the other, and τ is a dummy variable that is integrated out. We assume for now that the functions extend from $-\infty$ to ∞ .

We illustrated the basic mechanics of convolution in Section 3.4, and we will do so again later in this chapter and in Chapter 5. At the moment, we are interested in finding the Fourier transform of Eq. (4-24). We start with Eq. (4-19):

$$\begin{aligned}\Im\{(f \star h)(t)\} &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\tau)h(t - \tau)d\tau \right] e^{-j2\pi\mu t} dt \\ &= \int_{-\infty}^{\infty} f(\tau) \left[\int_{-\infty}^{\infty} h(t - \tau) e^{-j2\pi\mu t} dt \right] d\tau\end{aligned}$$

The term inside the brackets is the Fourier transform of $h(t - \tau)$. We will show later in this chapter that $\Im\{h(t - \tau)\} = H(\mu)e^{-j2\pi\mu\tau}$, where $H(\mu)$ is the Fourier transform of $h(t)$. Using this in the preceding equation gives us

$$\begin{aligned}\Im\{(f \star h)(t)\} &= \int_{-\infty}^{\infty} f(\tau) \left[H(\mu)e^{-j2\pi\mu\tau} \right] d\tau \\ &= H(\mu) \int_{-\infty}^{\infty} f(\tau) e^{-j2\pi\mu\tau} d\tau \\ &= H(\mu)F(\mu) \\ &= (H \cdot F)(\mu)\end{aligned}$$

Remember, convolution is commutative, so the order of the functions in convolution expressions does not matter.

where “ \cdot ” indicates multiplication. As noted earlier, if we refer to the domain of t as the spatial domain, and the domain of μ as the frequency domain, the preceding equation tells us that the Fourier transform of the convolution of two functions in the spatial domain is equal to the product in the frequency domain of the Fourier transforms of the two functions. Conversely, if we have the product of the two transforms, we can obtain the convolution in the spatial domain by computing the inverse Fourier transform. In other words, $f \star h$ and $H \cdot F$ are a Fourier transform pair. This result is one-half of the *convolution theorem* and is written as

$$(f \star h)(t) \Leftrightarrow (H \cdot F)(\mu) \quad (4-25)$$

As noted earlier, the double arrow indicates that the expression on the right is obtained by taking the *forward* Fourier transform of the expression on the left, while

the expression on the left is obtained by taking the *inverse* Fourier transform of the expression on the right.

Following a similar development would result in the other half of the convolution theorem:

$$(f \bullet h)(t) \Leftrightarrow (H \star F)(\mu) \quad (4-26)$$

which states that convolution in the frequency domain is analogous to multiplication in the spatial domain, the two being related by the forward and inverse Fourier transforms, respectively. As you will see later in this chapter, the convolution theorem is the foundation for filtering in the frequency domain.

4.3 SAMPLING AND THE FOURIER TRANSFORM OF SAMPLED FUNCTIONS

In this section, we use the concepts from Section 4.2 to formulate a basis for expressing sampling mathematically. Starting from basic principles, this will lead us to the Fourier transform of sampled functions. That is, the discrete Fourier transform.

SAMPLING

Continuous functions have to be converted into a sequence of discrete values before they can be processed in a computer. This requires sampling and quantization, as introduced in Section 2.4. In the following discussion, we examine sampling in more detail.

Consider a continuous function, $f(t)$, that we wish to sample at uniform intervals, ΔT , of the independent variable t (see Fig. 4.5). We assume initially that the function extends from $-\infty$ to ∞ with respect to t . One way to model sampling is to multiply $f(t)$ by a sampling function equal to a train of impulses ΔT units apart. That is,

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T) \quad (4-27)$$

where $\tilde{f}(t)$ denotes the sampled function. Each component of this summation is an impulse weighted by the value of $f(t)$ at the location of the impulse, as Fig. 4.5(c) shows. The *value* of each sample is given by the “strength” of the weighted impulse, which we obtain by integration. That is, the value, f_k , of an arbitrary sample in the sampled sequence is given by

$$\begin{aligned} f_k &= \int_{-\infty}^{\infty} f(t)\delta(t - k\Delta T)dt \\ &= f(k\Delta T) \end{aligned} \quad (4-28)$$

where we used the sifting property of δ in Eq. (4-13). Equation (4-28) holds for any integer value $k = \dots, -2, -1, 0, 1, 2, \dots$. Figure 4.5(d) shows the result, which consists of equally spaced samples of the original function.

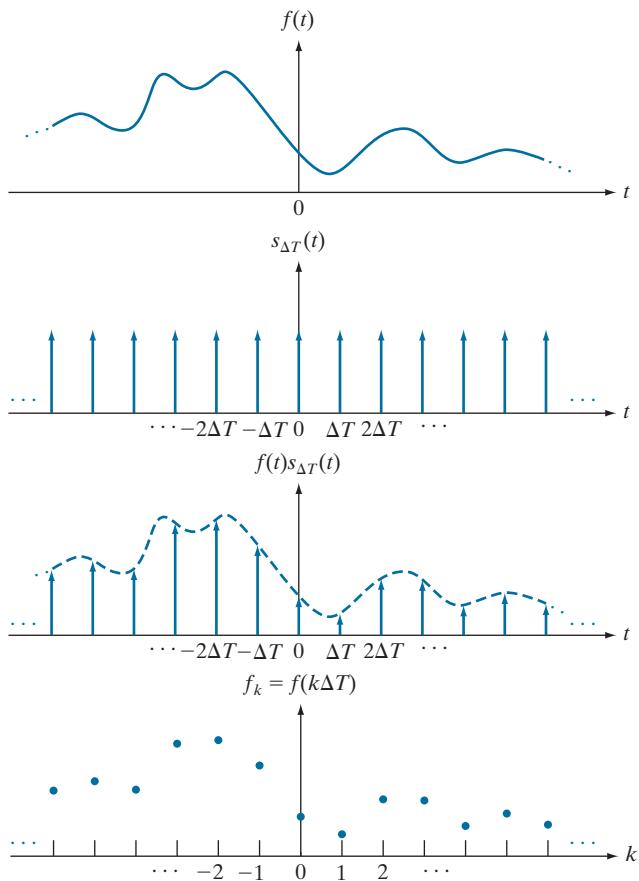
These two expressions also hold for discrete variables, with the exception that the right side of Eq. (4-26) is multiplied by $(1/M)$, where M is the number of discrete samples (see Problem 4.18).

Taking samples ΔT units apart implies a *sampling rate* equal to $1/\Delta T$. If the units of ΔT are seconds, then the sampling rate is in samples/s. If the units of ΔT are meters, then the sampling rate is in samples/m, and so on.

a
b
c
d

FIGURE 4.5

(a) A continuous function.
(b) Train of impulses used to model sampling.
(c) Sampled function formed as the product of (a) and (b).
(d) Sample values obtained by integration and using the sifting property of impulses. (The dashed line in (c) is shown for reference. It is not part of the data.)



THE FOURIER TRANSFORM OF SAMPLED FUNCTIONS

Let $F(\mu)$ denote the Fourier transform of a continuous function $f(t)$. As discussed in the previous section, the corresponding sampled function, $\tilde{f}(t)$, is the product of $f(t)$ and an impulse train. We know from the convolution theorem that the Fourier transform of the product of two functions in the spatial domain is the convolution of the transforms of the two functions in the frequency domain. Thus, the Fourier transform of the sampled function is:

$$\begin{aligned}\tilde{F}(\mu) &= \Im\{\tilde{f}(t)\} = \Im\{f(t)s_{\Delta T}(t)\} \\ &= (F \star S)(\mu)\end{aligned}\tag{4-29}$$

where, from Example 4.2,

$$S(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right)\tag{4-30}$$

is the Fourier transform of the impulse train $s_{\Delta T}(t)$. We obtain the convolution of $F(\mu)$ and $S(\mu)$ directly from the 1-D definition of convolution in Eq. (4-24):

$$\begin{aligned}
 \tilde{F}(\mu) &= (F \star S)(\mu) = \int_{-\infty}^{\infty} F(\tau)S(\mu - \tau)d\tau \\
 &= \frac{1}{\Delta T} \int_{-\infty}^{\infty} F(\tau) \sum_{n=-\infty}^{\infty} \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\
 &= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} F(\tau) \delta\left(\mu - \tau - \frac{n}{\Delta T}\right) d\tau \\
 &= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F\left(\mu - \frac{n}{\Delta T}\right)
 \end{aligned} \tag{4-31}$$

where the final step follows from the sifting property of the impulse, Eq. (4-13).

The summation in the last line of Eq. (4-31) shows that the Fourier transform $\tilde{F}(\mu)$ of the sampled function $\tilde{f}(t)$ is an *infinite, periodic* sequence of *copies* of the transform of the original, continuous function. The separation between copies is determined by the value of $1/\Delta T$. Observe that although $\tilde{f}(t)$ is a sampled function, its transform, $\tilde{F}(\mu)$, is *continuous* because it consists of copies of $F(\mu)$, which is a continuous function.

Figure 4.6 is a graphical summary of the preceding results.[†] Figure 4.6(a) is a sketch of the Fourier transform, $F(\mu)$, of a function $f(t)$, and Fig. 4.6(b) shows the transform, $\tilde{F}(\mu)$, of the sampled function, $\tilde{f}(t)$. As mentioned in the previous section, the quantity $1/\Delta T$ is the *sampling rate* used to generate the sampled function. So, in Fig. 4.6(b) the sampling rate was high enough to provide sufficient separation between the periods, and thus preserve the integrity (i.e., perfect copies) of $F(\mu)$. In Fig. 4.6(c), the sampling rate was just enough to preserve $F(\mu)$, but in Fig. 4.6(d), the sampling rate was below the minimum required to maintain distinct copies of $F(\mu)$, and thus failed to preserve the original transform. Figure 4.6(b) is the result of an *over-sampled* signal, while Figs. 4.6(c) and (d) are the results of *critically sampling* and *under-sampling* the signal, respectively. These concepts are the basis that will help you grasp the fundamentals of the sampling theorem, which we discuss next.

THE SAMPLING THEOREM

We introduced the idea of sampling intuitively in Section 2.4. Now we consider sampling formally, and establish the conditions under which a continuous function can be recovered uniquely from a set of its samples.

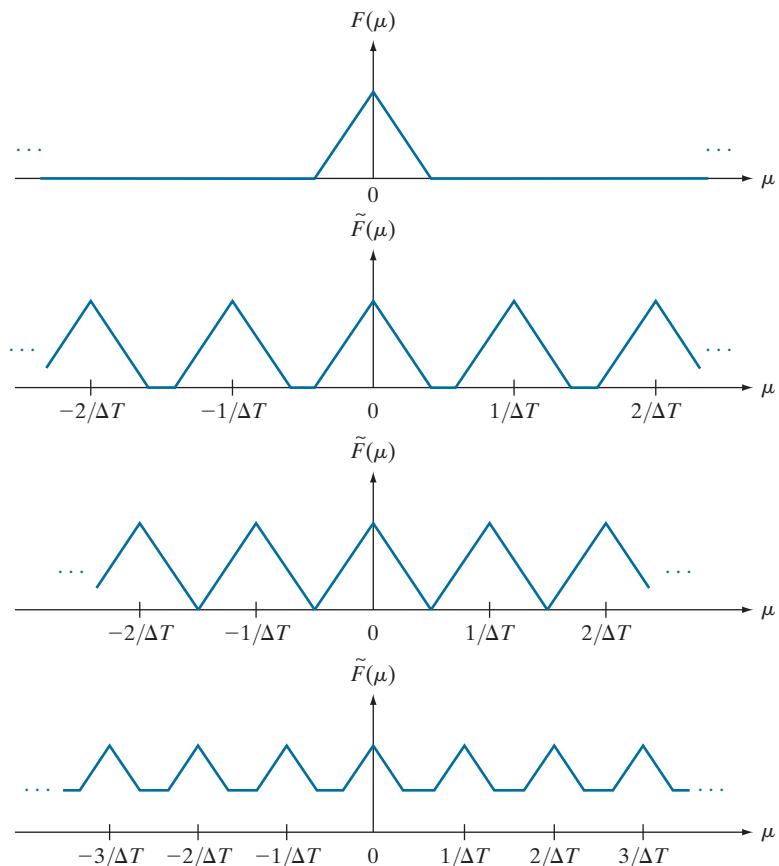
A function $f(t)$ whose Fourier transform is zero for values of frequencies outside a finite interval (band) $[-\mu_{\max}, \mu_{\max}]$ about the origin is called a *band-limited* function. Figure 4.7(a), which is a magnified section of Fig. 4.6(a), is such a function. Similarly, Fig. 4.7(b) is a more detailed view of the transform of the critically sampled

[†]For the sake of clarity in sketches of Fourier transforms in Fig. 4.6, and other similar figures in this chapter, we ignore the fact that Fourier transforms typically are complex functions. Our interest here is on concepts.

a
b
c
d

FIGURE 4.6

- (a) Illustrative sketch of the Fourier transform of a band-limited function.
 (b)–(d) Transforms of the corresponding sampled functions under the conditions of over-sampling, critically sampling, and under-sampling, respectively.



function [see Fig. 4.6(c)]. A higher value of ΔT would cause the periods in $\tilde{F}(\mu)$ to merge; a lower value would provide a clean separation between the periods.

We can recover $f(t)$ from its samples if we can isolate a single copy of $F(\mu)$ from the periodic sequence of copies of this function contained in $\tilde{F}(\mu)$, the transform of the sampled function $\tilde{f}(t)$. Recall from the discussion in the previous section that $\tilde{F}(\mu)$ is a *continuous, periodic* function with period $1/\Delta T$. Therefore, all we need is one complete period to characterize the entire transform. In other words, we can recover $f(t)$ from that single period by taking its inverse Fourier transform.

Extracting from $\tilde{F}(\mu)$ a single period that is equal to $F(\mu)$ is possible if the separation between copies is sufficient (see Fig. 4.6). In terms of Fig. 4.7(b), sufficient separation is guaranteed if $1/2\Delta T > \mu_{\max}$ or

$$\frac{1}{\Delta T} > 2\mu_{\max} \quad (4-32)$$

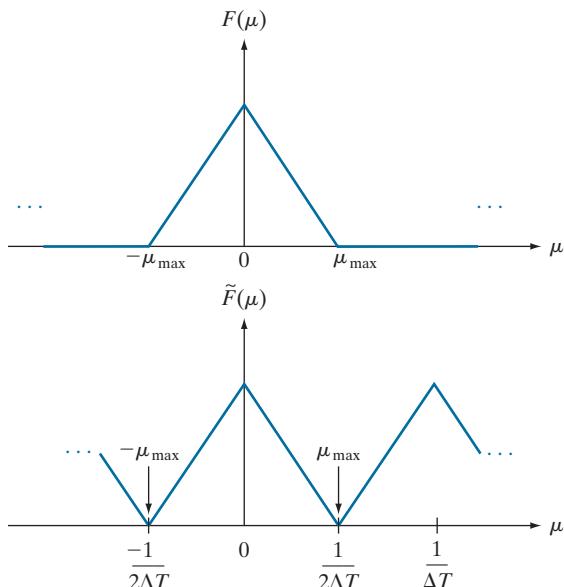
Remember, the sampling rate is the number of samples taken per unit of the independent variable.

This equation indicates that a continuous, band-limited function can be recovered completely from a set of its samples if the samples are acquired at a rate exceeding

a
b

FIGURE 4.7

- (a) Illustrative sketch of the Fourier transform of a band-limited function.
 (b) Transform resulting from critically sampling that band-limited function.



twice the highest frequency content of the function. This exceptionally important result is known as the *sampling theorem*.[†] We can say based on this result that no information is lost if a continuous, band-limited function is represented by samples acquired at a rate greater than twice the highest frequency content of the function. Conversely, we can say that the *maximum* frequency that can be “captured” by sampling a signal at a rate $1/\Delta T$ is $\mu_{\max} = 1/2\Delta T$. A sampling rate *exactly* equal to twice the highest frequency is called the *Nyquist rate*. Sampling at exactly the Nyquist rate sometimes is sufficient for perfect function recovery, but there are cases in which this leads to difficulties, as we will illustrate later in Example 4.3. This is the reason why the sampling theorem specifies that sampling must exceed the Nyquist rate.

Figure 4.8 illustrates the procedure for recovering $F(\mu)$ from $\tilde{F}(\mu)$ when a function is sampled at a rate higher than the Nyquist rate. The function in Fig. 4.8(b) is defined by the equation

The ΔT in Eq. (4-33)
cancels out the $1/\Delta T$ in
Eq. (4-31).

$$H(\mu) = \begin{cases} \Delta T & -\mu_{\max} \leq \mu \leq \mu_{\max} \\ 0 & \text{otherwise} \end{cases} \quad (4-33)$$

When multiplied by the periodic sequence in Fig. 4.8(a), this function isolates the period centered on the origin. Then, as Fig. 4.8(c) shows, we obtain $F(\mu)$ by multiplying $\tilde{F}(\mu)$ by $H(\mu)$:

[†]The sampling theorem is a cornerstone of digital signal processing theory. It was first formulated in 1928 by Harry Nyquist, a Bell Laboratories scientist and engineer. Claude E. Shannon, also from Bell Labs, proved the theorem formally in 1949. The renewed interest in the sampling theorem in the late 1940s was motivated by the emergence of early digital computing systems and modern communications, which created a need for methods dealing with digital (sampled) data.

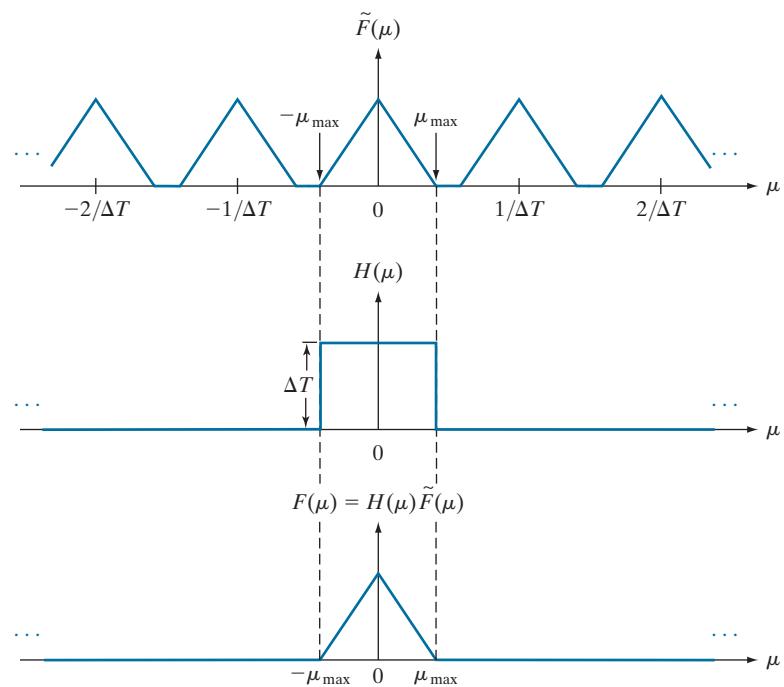
a
b
c

FIGURE 4.8

(a) Fourier transform of a sampled, band-limited function.

(b) Ideal lowpass filter transfer function.

(c) The product of (b) and (a), used to extract one period of the infinitely periodic sequence in (a).



$$F(\mu) = H(\mu)\tilde{F}(\mu) \quad (4-34)$$

Once we have $F(\mu)$, we can recover $f(t)$ using the inverse Fourier transform:

$$f(t) = \int_{-\infty}^{\infty} F(\mu)e^{j2\pi\mu t}d\mu \quad (4-35)$$

Equations (4-33) through (4-35) prove that, theoretically, it is possible to recover a band-limited function from samples obtained at a rate exceeding twice the highest frequency content of the function. As we will discuss in the following section, the requirement that $f(t)$ must be band-limited implies in general that $f(t)$ must extend from $-\infty$ to ∞ , a condition that cannot be met in practice. As you will see shortly, having to limit the duration of a function prevents perfect recovery of the function from its samples, except in some special cases.

Function $H(\mu)$ is called a *lowpass filter* because it passes frequencies in the low end of the frequency range, but it eliminates (filters out) higher frequencies. It is called also an *ideal lowpass filter* because of its instantaneous transitions in amplitude (between 0 and ΔT at location $-\mu_{\max}$ and the reverse at μ_{\max}), a characteristic that cannot be implemented physically in hardware. We can simulate ideal filters in software, but even then there are limitations (see Section 4.8). Because they are instrumental in recovering (reconstructing) the original function from its samples, filters used for the purpose just discussed are also called *reconstruction filters*.

In Fig. 3.32 we sketched the radial cross sections of filter transfer functions using only positive frequencies, for simplicity. Now you can see that frequency domain filter functions encompass both positive and negative frequencies.

ALIASING

Literally, the word *alias* means “a false identity.” In the field of signal processing, aliasing refers to sampling phenomena that cause different signals to become indistinguishable from one another after sampling; or, viewed another way, for one signal to “masquerade” as another.

Conceptually, the relationship between sampling and aliasing is not difficult to grasp. The foundation of aliasing phenomena as it relates to sampling is that we can describe a digitized function *only* by the values of its samples. This means that it is possible for two (or more) totally *different* continuous functions to coincide at the values of their respective samples, but we would have no way of knowing the characteristics of the functions between those samples. To illustrate, Fig. 4.9 shows two completely different sine functions sampled at the same rate. As you can see in Figs. 4.9(a) and (c), there are numerous places where the sampled values are the same in the two functions, resulting in identical sampled functions, as Figs. 4.9(b) and (d) show.

Two continuous functions having the characteristics just described are called an *aliased pair*, and such pairs are indistinguishable after sampling. Note that the reason these functions are aliased is because we used a sampling rate that is too coarse. That is, the functions were *under-sampled*. It is intuitively obvious that if sampling were refined, more and more of the differences between the two continuous functions would be revealed in the sampled signals. The principal objective of the following discussion is to answer the question: What is the minimum sampling rate required to avoid (or reduce) aliasing? This question has both a theoretical and a practical answer and, in the process of arriving at the answers, we will establish the conditions under which aliasing occurs.

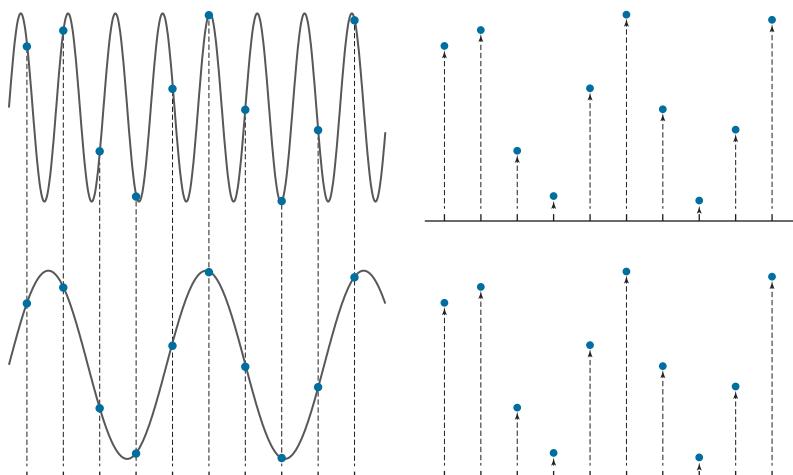
We can use the tools developed earlier in this section to formally answer the question we just posed. All we have to do is ask it in a different form: What happens

Although we show sinusoidal functions for simplicity, aliasing occurs between any arbitrary signals whose values are the same at the sample points.

a b
c d

FIGURE 4.9

The functions in (a) and (c) are totally different, but their digitized versions in (b) and (d) are identical. Aliasing occurs when the samples of two or more functions coincide, but the functions are different elsewhere.



if a band-limited function is sampled at less than the Nyquist rate (i.e., at less than twice its highest frequency)? This is precisely the under-sampled situation discussed earlier in this section and mentioned in the previous paragraph.

Figure 4.10(a) is the same as Fig. 4.6(d); it shows schematically the Fourier transform of an under-sampled, band-limited function. This figure illustrates that the net effect of lowering the sampling rate below the Nyquist rate is that the periods of the Fourier transform now overlap, and it becomes impossible to isolate a single period of the transform, regardless of the filter used. For instance, using the ideal lowpass filter in Fig. 4.10(b) would result in a transform that is corrupted by frequencies from adjacent periods, as Fig. 4.10(c) shows. The inverse transform would then yield a function, $f_a(t)$, different from the original. That is, $f_a(t)$ would be an aliased function because it would contain frequency components not present in the original. Using our earlier terminology, $f_a(t)$ would masquerade as a different function. It is possible for aliased functions to bear no resemblance whatsoever to the functions from which they originated.

Unfortunately, except in some special cases mentioned below, aliasing is always present in sampled signals. This is because, even if the original sampled function is band-limited, infinite frequency components are introduced the moment we limit the duration of the function, which we always have to do in practice. As an illustration, suppose that we want to limit the duration of a band-limited function, $f(t)$, to a finite interval, say $[0, T]$. We can do this by multiplying $f(t)$ by the function

$$h(t) = \begin{cases} 1 & 0 \leq t \leq T \\ 0 & \text{otherwise} \end{cases} \quad (4-36)$$

This function has the same basic shape as Fig. 4.4(a), whose Fourier transform, $H(\mu)$, has frequency components extending to infinity in both directions, as Fig. 4.4(b) shows. From the convolution theorem, we know that the transform of the product $h(t)f(t)$ is the convolution in the frequency domain of the transforms $F(\mu)$ and $H(\mu)$. Even if $F(\mu)$ is band-limited, convolving it with $H(\mu)$, which involves sliding one function across the other, will yield a result with frequency components extending to infinity in both directions (see Problem 4.12). From this we conclude that no function of *finite* duration can be band-limited. Conversely, a function that is band-limited must extend from $-\infty$ to ∞ .[†]

Although aliasing is an inevitable fact of working with sampled records of finite length, the effects of aliasing can be reduced by smoothing (lowpass filtering) the input function to attenuate its higher frequencies. This process, called *anti-aliasing*, has to be done *before* the function is sampled because aliasing is a sampling issue that cannot be “undone after the fact” using computational techniques.

[†]An important special case is when a function that extends from $-\infty$ to ∞ is band-limited *and* periodic. In this case, the function can be truncated and still be band-limited, *provided* that the truncation encompasses *exactly* an integral number of periods. A single truncated period (and thus the function) can be represented by a set of discrete samples satisfying the sampling theorem, taken over the truncated interval.

If we cannot isolate one period of the transform, we cannot recover the signal without aliasing,

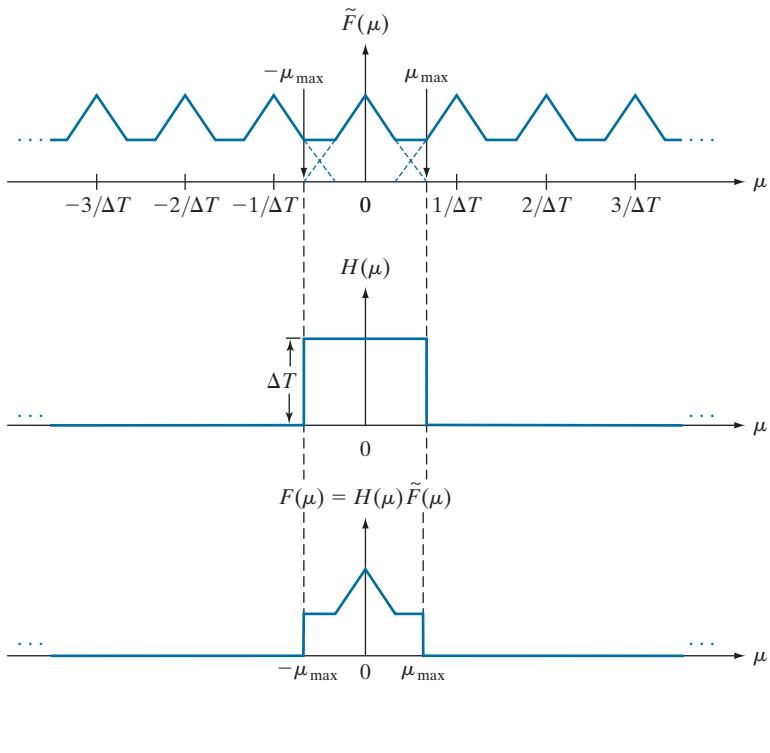


FIGURE 4.10 (a) Fourier transform of an under-sampled, band-limited function. (Interference between adjacent periods is shown dashed). (b) The same ideal lowpass filter used in Fig. 4.8. (c) The product of (a) and (b). The interference from adjacent periods results in aliasing that prevents perfect recovery of $F(\mu)$ and, consequently, of $f(t)$.

EXAMPLE 4.3: Aliasing.

Figure 4.11 shows a classic illustration of aliasing. A pure sine wave extending infinitely in both directions has a single frequency so, obviously, it is band-limited. Suppose that the sine wave in the figure (ignore the large dots for now) has the equation $f(t) = \sin(\pi t)$, and that the horizontal axis corresponds to time, t , in seconds. The function crosses the axis at $t = 0, \pm 1, \pm 2, \dots$.

Recall that a function $f(t)$ is *periodic* with period P if $f(t + P) = f(t)$ for all values of t . The period is the number (including fractions) of units of the independent variable that it takes for the function to complete one cycle. The *frequency* of a *periodic* function is the number of periods (cycles) that the function completes in one unit of the independent variable. Thus, the frequency of a periodic function is the *reciprocal* of the period. As before, the sampling rate is the number of samples taken per unit of the independent variable.

In the present example, the independent variable is time, and its units are seconds. The period, P , of $\sin(\pi t)$ is 2 s, and its frequency is $1/P$, or 1/2 cycles/s. According to the sampling theorem, we can recover this signal from a set of its samples if the sampling rate exceeds twice the highest frequency of the signal. This means that a sampling rate greater than 1 sample/s ($2 \times 1/2 = 1$) is required to

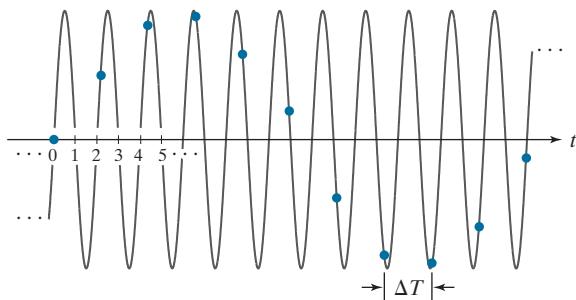


FIGURE 4.11 Illustration of aliasing. The under-sampled function (dots) looks like a sine wave having a frequency much lower than the frequency of the continuous signal. The period of the sine wave is 2 s, so the zero crossings of the horizontal axis occur every second. ΔT is the separation between samples.

recover the signal. Viewed another way, the separation, ΔT , between samples has to be less than 1 s. Observe that sampling this signal at *exactly* twice the frequency (1 sample/s), with samples taken at $t = 0, \pm 1, \pm 2, \dots$, results in $\dots \sin(-\pi), \sin(0), \sin(\pi) \dots$, all of which are 0. This illustrates the reason why the sampling theorem requires a sampling rate that exceeds twice the highest frequency of the function, as mentioned earlier.

The large dots in Fig. 4.11 are samples taken uniformly at a rate *below* the required 1 sample/s (i.e., the samples are taken *more* than 1 s apart; in fact, the separation between samples exceeds 2 s). The sampled signal *looks* like a sine wave, but its frequency is about one-tenth the frequency of the original function. This sampled signal, having a frequency well below anything present in the original continuous function, is an example of aliasing. If the signal had been sampled at a rate slightly exceeding the Nyquist rate, the samples would not look like a sine wave at all (see Problem 4.6).

Figure 4.11 also illustrates how aliasing can be extremely problematic in musical recordings by introducing frequencies not present in the original sound. In order to mitigate this, signals with frequencies above half the sampling rate *must* be filtered out to reduce the effect of aliased signals introduced into digital recordings. This is the reason why digital recording equipment contains lowpass filters specifically designed to remove frequency components above half the sampling rate used by the equipment.

If we were given just the samples in Fig. 4.11, another issue illustrating the seriousness of aliasing is that we would have no way of knowing that these samples are not a true representation of the original function. As you will see later in this chapter, aliasing in images can produce similarly misleading results.

FUNCTION RECONSTRUCTION (RECOVERY) FROM SAMPLED DATA

In this section, we show that reconstructing a function from a set of its samples reduces in practice to interpolating between the samples. Even the simple act of displaying an image requires reconstruction of the image from its samples by the display medium. Therefore, it is important to understand the fundamentals of sampled data reconstruction. Convolution is central to developing this understanding, demonstrating again the importance of this concept.

The discussion of Fig. 4.8 and Eq. (4-34) outlines the procedure for perfect recovery of a band-limited function from its samples using frequency domain methods.

Using the convolution theorem, we can obtain the equivalent result in the spatial domain. From Eq. (4-34), $F(\mu) = H(\mu)\tilde{F}(\mu)$, so it follows that

$$\begin{aligned} f(t) &= \mathcal{F}^{-1}\{F(\mu)\} \\ &= \mathcal{F}^{-1}\{H(\mu)\tilde{F}(\mu)\} \\ &= h(t) \star \tilde{f}(t) \end{aligned} \quad (4-37)$$

where, as before, $\tilde{f}(t)$ denotes the sampled function, and the last step follows from the convolution theorem, Eq. (4-25). It can be shown (see Problem 4.13), that substituting Eq. (4-27) for $\tilde{f}(t)$ into Eq. (4-37), and then using Eq. (4-24), leads to the following spatial domain expression for $f(t)$:

$$f(t) = \sum_{n=-\infty}^{\infty} f(n\Delta T) \text{sinc}[(t - n\Delta T)/\Delta T] \quad (4-38)$$

where the sinc function is defined in Eq. (4-23). This result is not unexpected because the inverse Fourier transform of the ideal (box) filter, $H(\mu)$, is a sinc function (see Example 4.1). Equation (4-38) shows that the perfectly reconstructed function, $f(t)$, is an infinite sum of sinc functions weighted by the sample values. It has the important property that the reconstructed function is identically equal to the sample values at multiple integer increments of ΔT . That is, for any $t = k\Delta T$, where k is an integer, $f(t)$ is equal to the k th sample, $f(k\Delta T)$. This follows from Eq. (4-38) because $\text{sinc}(0) = 1$ and $\text{sinc}(m) = 0$ for any other integer value of m . Between sample points, values of $f(t)$ are *interpolations* formed by the sum of the sinc functions.

See Section 2.4 regarding interpolation.

Equation (4-38) requires an infinite number of terms for the interpolations between samples. In practice, this implies that we have to look for approximations that are *finite* interpolations between the samples. As we discussed in Section 2.6, the principal interpolation approaches used in image processing are nearest-neighbor, bilinear, and bicubic interpolation. We will discuss the effects of interpolation on images in Section 4.5.

4.4 THE DISCRETE FOURIER TRANSFORM OF ONE VARIABLE

One of the principal goals of this chapter is the derivation of the *discrete Fourier transform* (DFT) starting from basic principles. The material up to this point may be viewed as the foundation of those basic principles, so now we have in place the necessary tools to derive the DFT.

OBTAINING THE DFT FROM THE CONTINUOUS TRANSFORM OF A SAMPLED FUNCTION

As we discussed in Section 4.3, the Fourier transform of a sampled, band-limited function extending from $-\infty$ to ∞ is a *continuous, periodic* function that also extends from $-\infty$ to ∞ . In practice, we work with a finite number of samples, and the objective of this section is to derive the DFT of such finite sample sets.

Equation (4-31) gives the transform, $\tilde{F}(\mu)$, of sampled data in terms of the transform of the original function, but it does not give us an expression for $\tilde{F}(\mu)$ in terms

of the sampled function $\tilde{f}(t)$ itself. We find that expression directly from the definition of the Fourier transform in Eq. (4-19):

$$\tilde{F}(\mu) = \int_{-\infty}^{\infty} \tilde{f}(t) e^{-j2\pi\mu t} dt \quad (4-39)$$

By substituting Eq. (4-27) for $\tilde{f}(t)$, we obtain

$$\begin{aligned} \tilde{F}(\mu) &= \int_{-\infty}^{\infty} \tilde{f}(t) e^{-j2\pi\mu t} dt = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(t) \delta(t - n\Delta T) e^{-j2\pi\mu t} dt \\ &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} f(t) \delta(t - n\Delta T) e^{-j2\pi\mu t} dt \\ &= \sum_{n=-\infty}^{\infty} f_n e^{-j2\pi\mu n\Delta T} \end{aligned} \quad (4-40)$$

The last step follows from Eq. (4-28) and the sifting property of the impulse. Although f_n is a discrete function, its Fourier transform, $\tilde{F}(\mu)$, is continuous and infinitely periodic with period $1/\Delta T$, as we know from Eq. (4-31). Therefore, all we need to characterize $\tilde{F}(\mu)$ is one period, and sampling one period of this function is the basis for the DFT.

Suppose that we want to obtain M equally spaced samples of $\tilde{F}(\mu)$ taken over the one period interval from $\mu = 0$ to $\mu = 1/\Delta T$ (see Fig. 4.8). This is accomplished by taking the samples at the following frequencies:

$$\mu = \frac{m}{M\Delta T} \quad m = 0, 1, 2, \dots, M-1 \quad (4-41)$$

Substituting this result for μ into Eq. (4-40) and letting F_m denote the result yields

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi mn/M} \quad m = 0, 1, 2, \dots, M-1 \quad (4-42)$$

This expression is the *discrete Fourier transform* we are seeking.[†] Given a set $\{f_m\}$ consisting of M samples of $f(t)$, Eq. (4-42) yields a set $\{F_m\}$ of M complex values corresponding to the discrete Fourier transform of the input sample set. Conversely,

[†]Referring back to Fig. 4.6(b), note that the interval $[0, 1/\Delta T]$ over which we sampled one period of $\tilde{F}(\mu)$ covers two adjacent half periods of the transform (but with the lowest half of period appearing at higher frequencies). This means that the data in F_m requires re-ordering to obtain samples that are ordered from the lowest to the highest frequency of the period. This is the price paid for the notational convenience of taking the samples at $m = 0, 1, 2, \dots, M-1$, instead of using samples on either side of the origin, which would require the use of negative notation. The procedure used to order the transform data will be discussed in Section 4.6.

given $\{F_m\}$, we can recover the sample set $\{f_n\}$ by using the *inverse discrete Fourier transform* (IDFT)

$$f_n = \frac{1}{M} \sum_{m=0}^{M-1} F_m e^{j2\pi mn/M} \quad n = 0, 1, 2, \dots, M-1 \quad (4-43)$$

It is not difficult to show (see Problem 4.15) that substituting Eq. (4-43) for f_n into Eq. (4-42) gives the identity $F_m \equiv f_m$. Similarly, substituting Eq. (4-42) into Eq. (4-43) for F_m yields $f_n \equiv f_n$. This implies that Eqs. (4-42) and (4-43) constitute a *discrete Fourier transform pair*. Furthermore, these identities indicate that the forward and inverse Fourier transforms exist for any set of samples whose values are finite. Note that neither expression depends explicitly on the sampling interval ΔT , nor on the frequency intervals of Eq. (4-41). Therefore, the DFT pair is applicable to *any* finite set of discrete samples taken uniformly.

We used m and n in the preceding development to denote discrete variables because it is typical to do so for derivations. However, it is more intuitive, especially in two dimensions, to use the notation x and y for image coordinate variables and u and v for frequency variables, where these are understood to be integers.[†] Then, Eqs. (4-42) and (4-43) become

$$F(u) = \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \quad u = 0, 1, 2, \dots, M-1 \quad (4-44)$$

and

$$f(x) = \frac{1}{M} \sum_{u=0}^{M-1} F(u) e^{j2\pi ux/M} \quad x = 0, 1, 2, \dots, M-1 \quad (4-45)$$

where we used functional notation instead of subscripts for simplicity. Comparing Eqs. (4-42) through (4-45), you can see that $F(u) \equiv F_m$ and $f(x) \equiv f_n$. From this point on, we use Eqs. (4-44) and (4-45) to denote the 1-D DFT pair. As in the continuous case, we often refer to Eq. (4-44) as the *forward DFT* of $f(x)$, and to Eq. (4-45) as the *inverse DFT* of $F(u)$. As before, we use the notation $f(x) \Leftrightarrow F(u)$ to denote a Fourier transform pair. Sometimes you will encounter in the literature the $1/M$ term in front of Eq. (4-44) instead. That does not affect the proof that the two equations form a Fourier transform pair (see Problem 4.15).

Knowledge that $f(x)$ and $F(u)$ are a transform pair is useful in proving relationships between functions and their transforms. For example, you are asked in Problem 4.17 to show that $f(x - x_0) \Leftrightarrow F(u)e^{-j2\pi ux_0/M}$ is a Fourier transform pair. That is, you have to show that the DFT of $f(x - x_0)$ is $F(u)e^{-j2\pi ux_0/M}$ and, conversely, that the *inverse DFT* of $F(u)e^{-j2\pi ux_0/M}$ is $f(x - x_0)$. Because this is done by substituting

[†]We have been careful in using t for *continuous* spatial variables and μ for the corresponding *continuous* frequency variables. From this point on, we will use x and u to denote 1-D *discrete* spatial and frequency variables, respectively. When working in 2-D, we will use (t, z) , and (μ, ν) , to denote continuous spatial and frequency domain variables, respectively. Similarly, we will use (x, y) and (u, v) to denote their discrete counterparts.

directly into Eqs. (4-44) and (4-45), and you will have proved already that these two equations constitute a Fourier transform pair (Problem 4.15), if you prove that one side of “ \Leftrightarrow ” is the DFT (IDFT) of the other, then it must be true the other side is the IDFT (DFT) of the side you just proved. It turns out that having the option to prove one side or the other often simplifies proofs significantly. This is true also of the 1-D continuous and 2-D continuous and discrete Fourier transform pairs.

It can be shown (see Problem 4.16) that both the forward and inverse discrete transforms are infinitely periodic, with period M . That is,

$$F(u) = F(u + kM) \quad (4-46)$$

and

$$f(x) = f(x + kM) \quad (4-47)$$

where k is an integer.

The discrete equivalent of the 1-D convolution in Eq. (4-24) is

$$f(x) \star h(x) = \sum_{m=0}^{M-1} f(m)h(x-m) \quad x = 0, 1, 2, \dots, M-1 \quad (4-48)$$

Because in the preceding formulations the functions are periodic, their convolution also is periodic. Equation (4-48) gives one period of the periodic convolution. For this reason, this equation often is referred to as *circular convolution*. This is a direct result of the periodicity of the DFT and its inverse. This is in contrast with the convolution you studied in Section 3.4, in which values of the displacement, x , were determined by the requirement of sliding one function completely past the other, and were not fixed to the range $[0, M - 1]$ as in circular convolution. We will discuss this difference and its significance in Section 4.6 and in Fig. 4.27.

Finally, we point out that the convolution theorem given in Eqs. (4-25) and (4-26) is applicable also to discrete variables, with the exception that the right side of Eq. (4-26) is multiplied by $1/M$ (Problem 4.18).

RELATIONSHIP BETWEEN THE SAMPLING AND FREQUENCY INTERVALS

If $f(x)$ consists of M samples of a function $f(t)$ taken ΔT units apart, the length of the record comprising the set $\{f(x)\}$, $x = 0, 1, 2, \dots, M - 1$, is

$$T = M\Delta T \quad (4-49)$$

The corresponding spacing, Δu , in the frequency domain follows from Eq. (4-41):

$$\Delta u = \frac{1}{M\Delta T} = \frac{1}{T} \quad (4-50)$$

The entire frequency range spanned by the M components of the DFT is then

$$R = M\Delta u = \frac{1}{\Delta T} \quad (4-51)$$

Thus, we see from Eqs. (4-50) and (4-51) that the resolution in frequency, Δu , of the DFT depends inversely on the length (duration, if t is time) of the record, T , over which the continuous function, $f(t)$, is sampled; and the range of frequencies spanned by the DFT depends on the sampling interval ΔT . Keep in mind these *inverse* relationships between Δu and ΔT .

EXAMPLE 4.4: The mechanics of computing the DFT.

Figure 4.12(a) shows four samples of a continuous function, $f(t)$, taken ΔT units apart. Figure 4.12(b) shows the samples in the x -domain. The values of x are 0, 1, 2, and 3, which refer to the number of the samples in sequence, counting up from 0. For example, $f(2) = f(t_0 + 2\Delta T)$, the third sample of $f(t)$.

From Eq. (4-44), the first value of $F(u)$ [i.e., $F(0)$] is

$$F(0) = \sum_{x=0}^3 f(x) = [f(0) + f(1) + f(2) + f(3)] = 1 + 2 + 4 + 4 = 11$$

The next value of $F(u)$ is

$$F(1) = \sum_{x=0}^3 f(x)e^{-j2\pi(1)x/4} = 1e^0 + 2e^{-j\pi/2} + 4e^{-j\pi} + 4e^{-j3\pi/2} = -3 + 2j$$

Similarly, $F(2) = -(1 + 0j)$ and $F(3) = -(3 + 2j)$. Observe that *all* values of $f(x)$ are used in computing each value of $F(u)$.

If we were given $F(u)$ instead, and were asked to compute its inverse, we would proceed in the same manner, but using the inverse Fourier transform. For instance,

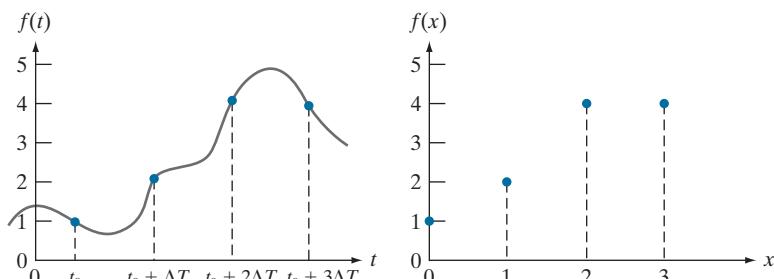
$$f(0) = \frac{1}{4} \sum_{u=0}^3 F(u)e^{j2\pi u(0)} = \frac{1}{4} \sum_{u=0}^3 F(u) = \frac{1}{4}[11 - 3 + 2j - 1 - 3 - 2j] = \frac{1}{4}[4] = 1$$

which agrees with Fig. 4.12(b). The other values of $f(x)$ are obtained in a similar manner.

a b

FIGURE 4.12

- (a) A continuous function sampled ΔT units apart.
- (b) Samples in the x -domain. Variable t is continuous, while x is discrete.



4.5 EXTENSIONS TO FUNCTIONS OF TWO VARIABLES

In the following discussion we extend to two variables the concepts introduced in the previous sections of this chapter.

THE 2-D IMPULSE AND ITS SIFTING PROPERTY

The impulse, $\delta(t, z)$, of two continuous variables, t and z , is defined as before:

$$\delta(t, z) = \begin{cases} 1 & \text{if } t = z = 0 \\ 0 & \text{otherwise} \end{cases} \quad (4-52)$$

and

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t, z) dt dz = 1 \quad (4-53)$$

As in the 1-D case, the 2-D impulse exhibits the *sifting property* under integration,

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) \delta(t, z) dt dz = f(0, 0) \quad (4-54)$$

or, more generally for an impulse located at (t_0, z_0) ,

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) \delta(t - t_0, z - z_0) dt dz = f(t_0, z_0) \quad (4-55)$$

As before, we see that the sifting property yields the value of the function at the location of the impulse.

For discrete variables x and y , the 2-D discrete unit impulse is defined as

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y = 0 \\ 0 & \text{otherwise} \end{cases} \quad (4-56)$$

and its sifting property is

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y) \delta(x, y) = f(0, 0) \quad (4-57)$$

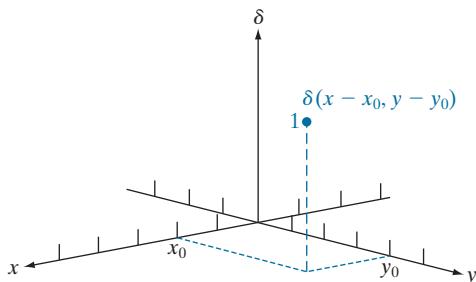
where $f(x, y)$ is a function of discrete variables x and y . For an impulse located at coordinates (x_0, y_0) (see Fig. 4.13) the sifting property is

$$\sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y) \delta(x - x_0, y - y_0) = f(x_0, y_0) \quad (4-58)$$

When working with an image of finite dimensions, the limits in the two preceding equations are replaced by the dimensions of the image.

FIGURE 4.13

2-D unit discrete impulse. Variables x and y are discrete, and δ is zero everywhere except at coordinates (x_0, y_0) , where its value is 1.



THE 2-D CONTINUOUS FOURIER TRANSFORM PAIR

Let $f(t, z)$ be a continuous function of two continuous variables, t and z . The two-dimensional, continuous Fourier transform pair is given by the expressions

$$F(\mu, \nu) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) e^{-j2\pi(\mu t + \nu z)} dt dz \quad (4-59)$$

and

$$f(t, z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\mu, \nu) e^{j2\pi(\mu t + \nu z)} d\mu d\nu \quad (4-60)$$

where μ and ν are the frequency variables. When referring to images, t and z are interpreted to be continuous *spatial* variables. As in the 1-D case, the domain of the variables μ and ν defines the *continuous frequency domain*.

EXAMPLE 4.5: Obtaining the Fourier transform of a 2-D box function.

Figure 4.14(a) shows the 2-D equivalent of the 1-D box function in Example 4.1. Following a procedure similar to the one used in that example gives the result

$$\begin{aligned} F(\mu, \nu) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, z) e^{-j2\pi(\mu t + \nu z)} dt dz = \int_{-T/2}^{T/2} \int_{-Z/2}^{Z/2} A e^{-j2\pi(\mu t + \nu z)} dt dz \\ &= ATZ \left[\frac{\sin(\pi\mu T)}{(\pi\mu T)} \right] \left[\frac{\sin(\pi\nu Z)}{(\pi\nu Z)} \right] \end{aligned}$$

Figure 4.14(b) shows a portion of the spectrum about the origin. As in the 1-D case, the locations of the zeros in the spectrum are inversely proportional to the values of T and Z . In this example, T is larger than Z , so the spectrum is the more “contracted” along the μ -axis.

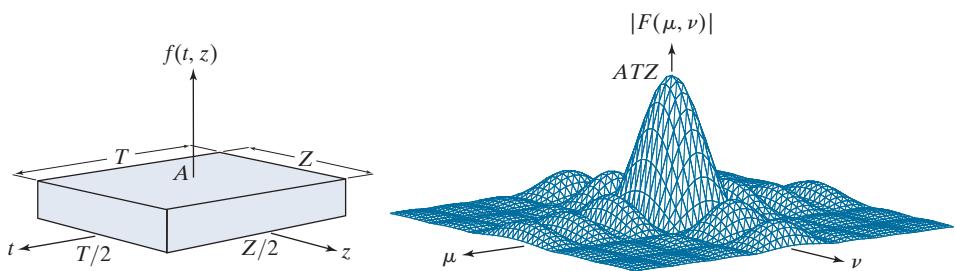
2-D SAMPLING AND THE 2-D SAMPLING THEOREM

In a manner similar to the 1-D case, sampling in two dimensions can be modeled using a sampling function (i.e., a 2-D impulse train):

a | b

FIGURE 4.14

(a) A 2-D function and (b) a section of its spectrum. The box is longer along the t -axis, so the spectrum is more contracted along the μ -axis.



$$s_{\Delta T \Delta Z}(t, z) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - m\Delta T, z - n\Delta Z) \quad (4-61)$$

where ΔT and ΔZ are the separations between samples along the t - and z -axis of the continuous function $f(t, z)$. Equation (4-61) describes a set of periodic impulses extending infinitely along the two axes (see Fig. 4.15). As in the 1-D case illustrated in Fig. 4.5, multiplying $f(t, z)$ by $s_{\Delta T \Delta Z}(t, z)$ yields the sampled function.

Function $f(t, z)$ is said to be *band limited* if its Fourier transform is 0 outside a rectangle established in the frequency domain by the intervals $[-\mu_{\max}, \mu_{\max}]$ and $[-\nu_{\max}, \nu_{\max}]$; that is,

$$F(\mu, \nu) = 0 \quad \text{for } |\mu| \geq \mu_{\max} \text{ and } |\nu| \geq \nu_{\max} \quad (4-62)$$

The *two-dimensional sampling theorem* states that a continuous, band-limited function $f(t, z)$ can be recovered with no error from a set of its samples if the sampling intervals are

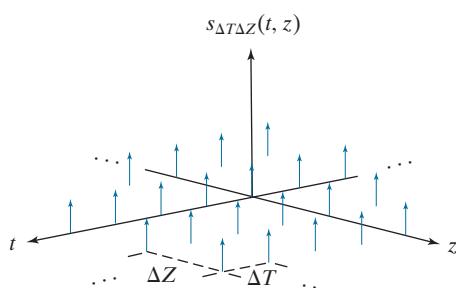
$$\Delta T < \frac{1}{2\mu_{\max}} \quad (4-63)$$

and

$$\Delta Z < \frac{1}{2\nu_{\max}} \quad (4-64)$$

or, expressed in terms of the sampling rate, if

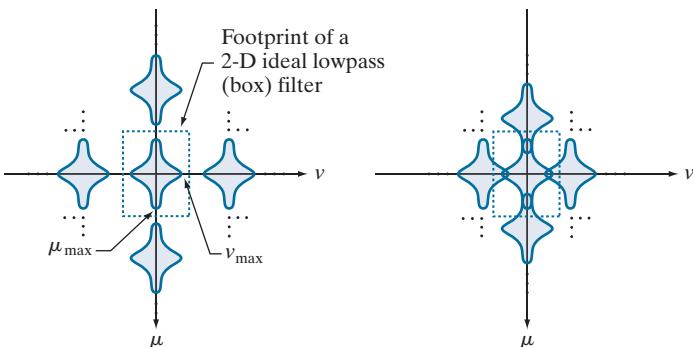
FIGURE 4.15
2-D impulse train.



a b

FIGURE 4.16

Two-dimensional Fourier transforms of (a) an over-sampled, and (b) an under-sampled, band-limited function.



$$\frac{1}{\Delta T} > 2\mu_{\max} \quad (4-65)$$

and

$$\frac{1}{\Delta Z} > 2\nu_{\max} \quad (4-66)$$

Stated another way, we say that no information is lost if a 2-D, band-limited, continuous function is represented by samples acquired at rates greater than twice the highest frequency content of the function in both the μ - and ν -directions.

Figure 4.16 shows the 2-D equivalents of Figs. 4.6(b) and (d). A 2-D ideal filter transfer function has the form illustrated in Fig. 4.14(a) (but in the frequency domain). The dashed portion of Fig. 4.16(a) shows the location of the filter function to achieve the necessary isolation of a single period of the transform for reconstruction of a band-limited function from its samples, as in Fig. 4.8. From Fig 4.10, we know that if the function is under-sampled, the periods overlap, and it becomes impossible to isolate a single period, as Fig. 4.16(b) shows. Aliasing would result under such conditions.

ALIASING IN IMAGES

In this section, we extend the concept of aliasing to images, and discuss in detail several aspects of aliasing related to image sampling and resampling.

Extensions from 1-D Aliasing

As in the 1-D case, a continuous function $f(t, z)$ of two continuous variables, t and z , can be band-limited in general only if it extends infinitely in both coordinate directions. The very act of limiting the spatial duration of the function (e.g., by multiplying it by a box function) introduces corrupting frequency components extending to infinity in the frequency domain, as explained in Section 4.3 (see also Problem 4.12). Because we cannot sample a function infinitely, aliasing is always present in digital images, just as it is present in sampled 1-D functions. There are two principal manifestations of aliasing in images: spatial aliasing and temporal aliasing. *Spatial aliasing* is caused by under-sampling, as discussed in Section 4.3, and tends to be more visible

(and objectionable) in images with repetitive patterns. *Temporal aliasing* is related to time intervals between images of a sequence of dynamic images. One of the most common examples of temporal aliasing is the “wagon wheel” effect, in which wheels with spokes in a sequence of images (for example, in a movie) appear to be rotating backwards. This is caused by the frame rate being too low with respect to the speed of wheel rotation in the sequence, and is similar to the phenomenon described in Fig. 4.11, in which under sampling produced a signal that appeared to be of much lower frequency than the original.

Our focus in this chapter is on spatial aliasing. The key concerns with spatial aliasing in images are the introduction of artifacts such as jaggedness in line features, spurious highlights, and the appearance of frequency patterns not present in the original image. Just as we used Fig. 4.9 to explain aliasing in 1-D functions, we can develop an intuitive grasp of the nature of aliasing in images using some simple graphics. The sampling grid in the center section of Fig. 4.17 is a 2-D representation of the impulse train in Fig. 4.15. In the grid, the little white squares correspond to the location of the impulses (where the image is sampled) and black represents the separation between samples. Superimposing the sampling grid on an image is analogous to multiplying the image by an impulse train, so the same sampling concepts we discussed in connection with the impulse train in Fig. 4.15 are applicable here. The focus now is to analyze graphically the interaction between sampling rate (the separation of the sampling points in the grid) and the frequency of the 2-D signals being sampled.

Figure 4.17 shows a sampling grid partially overlapping three 2-D signals (regions of an image) of low, mid, and high spatial frequencies (relative to the separation between sampling cells in the grid). Note that the level of spatial “detail” in the regions is proportional to frequency (i.e., higher-frequency signals contain more bars). The sections of the regions inside the sampling grip are rough manifestations of how they would appear after sampling. As expected, all three digitized regions

FIGURE 4.17

Various aliasing effects resulting from the interaction between the frequency of 2-D signals and the sampling rate used to digitize them. The regions outside the sampling grid are continuous and free of aliasing.

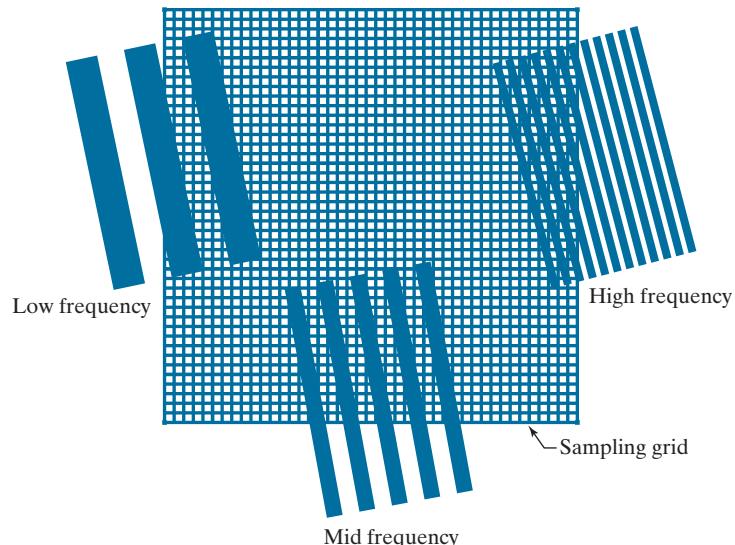


exhibit aliasing to some degree, but the effects are dramatically different, worsening as the discrepancy between detail (frequency) and sampling rate increases. The low-frequency region is rendered reasonably well, with some mild jaggedness around the edges. The jaggedness increases as the frequency of the region increases to the mid-range because the sampling rate is the same. This edge distortion (appropriately called *jaggies*) is common in images with strong line and/or edge content.

The digitized high-frequency region in the top right of Fig. 4.17 exhibits totally different and somewhat surprising behavior. Additional stripes (of lower frequency) appear in the digitized section, and these stripes are rotated significantly with respect to the direction of the stripes in the continuous region. These stripes are an alias of a totally different signal. As the following example shows, this type of behavior can result in images that appear “normal” and yet bear no relation to the original.

EXAMPLE 4.6: Aliasing in images.

Consider an imaging system that is perfect, in the sense that it is noiseless and produces an exact digital image of what it sees, but the number of samples it can take is fixed at 96×96 pixels. For simplicity, assume that pixels are little squares of unit width and length. We want to use this system to digitize checkerboard images of alternating black and white squares. Checkerboard images can be interpreted as periodic, extending infinitely in both dimensions, where one period is equal to adjacent black/white pairs. If we specify “valid” digitized images as being those extracted from an infinite sequence in such a way that the image contains an integer multiple of periods, then, based on our earlier comments, we know that properly sampled periodic images will be free of aliasing. In the present example, this means that the sizes of the squares must be such that dividing 96 by the size yields an even number. This will give an integer number of periods (pairs of black/white squares). The smallest size of squares under the stated conditions is 1 pixel.

The principal objective of this example is to examine what happens when checkerboard images with squares of sizes less than 1 pixel on the side are presented to the system. This will correspond to the undersampled case discussed earlier, which will result in aliasing. A horizontal or vertical scan line of the checkerboard images results in a 1-D square wave, so we can focus the analysis on 1-D signals.

To understand the capabilities of our imaging system in terms of sampling, recall from the discussion of the 1-D sampling theorem that, given the sampling rate, the maximum frequency allowed before aliasing occurs in the sampled signal has to be less than one-half the sampling rate. Our sampling rate is fixed, at one sample per unit of the independent variable (the units are pixels). Therefore, the maximum frequency our signal can have in order to avoid aliasing is $1/2$ cycle/pixel.

We can arrive at the same conclusion by noting that the most demanding image our system can handle is when the squares are 1 unit (pixel) wide, in which case the period (cycle) is two pixels. The frequency is the reciprocal of the period, or $1/2$ cycle/pixel, as in the previous paragraph.

Figures 4.18(a) and (b) show the result of sampling checkerboard images whose squares are of sizes 16×16 and 6×6 pixels, respectively. The frequencies of scan lines in either direction of these two images are $1/32$ and $1/6$ cycles/pixel. These are well below the $1/2$ cycles/pixel allowed for our system. Because, as mentioned earlier, the images are perfectly registered in the field of view of the system, the results are free of aliasing, as expected.

When the size of the squares is reduced to slightly less than one pixel, a severely aliased image results, as Fig. 4.18(c) shows (the squares used were approximately of size 0.95×0.95 pixels). Finally, reducing

a	b
c	d

FIGURE 4.18

Aliasing. In (a) and (b) the squares are of sizes 16 and 6 pixels on the side. In (c) and (d) the squares are of sizes 0.95 and 0.48 pixels, respectively. Each small square in (c) is one pixel. Both (c) and (d) are aliased. Note how (d) masquerades as a “normal” image.



the size of the squares to slightly less than 0.5 pixels on the side yielded the image in Fig. 4.18(d). In this case, the aliased result looks like a normal checkerboard pattern. In fact, this image would result from sampling a checkerboard image whose squares are 12 pixels on the side. This last image is a good reminder that aliasing can create results that may be visually quite misleading.

The effects of aliasing can be reduced by slightly defocusing the image to be digitized so that high frequencies are attenuated. As explained in Section 4.3, anti-aliasing filtering has to be done at the “front-end,” *before* the image is sampled. There are no such things as after-the-fact software anti-aliasing filters that can be used to reduce the effects of aliasing caused by violations of the sampling theorem. Most commercial digital image manipulation packages do have a feature called “anti-aliasing.” However, as illustrated in Example 4.8 below, this term is related to blurring a digital image to reduce additional aliasing artifacts caused by resampling. The term does not apply to reducing aliasing in the original sampled image. A significant number of commercial digital cameras have true anti-aliasing filtering built in, either in the lens or on the surface of the sensor itself. Even nature uses this approach to reduce the effects of aliasing in the human eye, as the following example shows.

EXAMPLE 4.7: Nature obeys the limits of the sampling theorem.

When discussing Figs. 2.1 and 2.2, we mentioned that cones are the sensors responsible for sharp vision. Cones are concentrated in the fovea, in line with the visual axis of the lens, and their concentration is measured in degrees off that axis. A standard test of visual acuity (the ability to resolve fine detail) in humans is to place a pattern of alternating black and white stripes in one degree of the visual field. If the total number of stripes exceeds 120 (i.e., a frequency of 60 cycles/degree), experimental evidence shows that the observer will perceive the image as a single gray mass. That is, the lens in the eye automatically lowpass filters spatial frequencies higher than 60 cycles/degree. Sampling in the eye is done by the cones, so, based on the sampling theorem, we would expect the eye to have on the order of 120 cones/degree in order to avoid the effects of aliasing. As it turns out, that is exactly what we have!

Image Resampling and Interpolation

As in the 1-D case, perfect reconstruction of a band-limited image function from a set of its samples requires 2-D convolution in the spatial domain with a sinc function. As explained in Section 4.3, this theoretically perfect reconstruction requires interpolation using infinite summations which, in practice, forces us to look for approximate interpolation methods. One of the most common applications of 2-D interpolation in image processing is in image resizing (zooming and shrinking). Zooming may be viewed as over-sampling, while shrinking may be viewed as under-sampling. The key difference between these two operations and the sampling concepts discussed in previous sections is that we are applying zooming and shrinking to digital images.

We introduced interpolation in Section 2.4. Our interest there was to illustrate the performance of nearest neighbor, bilinear, and bicubic interpolation. In this section, the focus is on sampling and anti-aliasing issues. Aliasing generally is introduced when an image is scaled, either by zooming or by shrinking. For example, a special case of nearest neighbor interpolation is zooming by *pixel replication*, which we use to increase the size of an image an integer number of times. To double the size of an image, we duplicate each column. This doubles the image size in the horizontal direction. Then, we duplicate each row of the enlarged image to double the size in the vertical direction. The same procedure is used to enlarge the image any integer number of times. The intensity level assignment of each pixel is predetermined by the fact that new locations are exact duplicates of old locations. In this crude method of enlargement, one of the principal aliasing effects is the introduction of jaggies on straight lines that are not horizontal or vertical. The effects of aliasing in image enlargement often are reduced significantly by using more sophisticated interpolation, as we discussed in Section 2.4. We show in the following example that aliasing can also be a serious problem in image shrinking.

EXAMPLE 4.8: Illustration of aliasing in resampled natural images.

The effects of aliasing generally are worsened when the size of a digital image is reduced. Figure 4.19(a) is an image containing regions purposely selected to illustrate the effects of aliasing (note the thinly spaced parallel lines in all garments worn by the subject). There are no objectionable aliasing artifacts in Fig. 4.19(a), indicating that the sampling rate used initially was sufficient to mitigate visible aliasing.

In Fig. 4.19(b), the image was reduced to 33% of its original size using row/column deletion. The effects of aliasing are quite visible in this image (see, for example, the areas around scarf and the subject's knees). Images (a) and (b) are shown in the same size because the reduced image was brought back to its original size by pixel replication (the replication did not alter appreciably the effects of aliasing just discussed).

The digital “equivalent” of the defocusing of continuous images mentioned earlier for reducing aliasing, is to attenuate the high frequencies of a *digital* image by smoothing it with a lowpass filter before resampling. Figure 4.19(c) was processed in the same manner as Fig. 4.19(b), but the original image was smoothed using a 5×5 spatial averaging filter (see Section 3.5) before reducing its size. The improvement over Fig. 4.19(b) is evident. The image is slightly more blurred than (a) and (b), but aliasing is no longer objectionable.



a b c

FIGURE 4.19 Illustration of aliasing on resampled natural images. (a) A digital image of size 772×548 pixels with visually negligible aliasing. (b) Result of resizing the image to 33% of its original size by pixel deletion and then restoring it to its original size by pixel replication. Aliasing is clearly visible. (c) Result of blurring the image in (a) with an averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)

The term *moiré* is a French word (not the name of a person) that appears to have originated with weavers, who first noticed what appeared to be interference patterns visible on some fabrics. The root of the word is from the word *mohair*, a cloth made from Angora goat hairs.

Aliasing and Moiré Patterns

In optics, a *moiré pattern* is a secondary, visual phenomenon produced, for example, by superimposing two gratings of approximately equal spacing. These patterns are common, everyday occurrences. For instance, we see them in overlapping insect window screens and on the interference between TV raster lines and striped or highly textured materials in the background, or worn by individuals. In digital image processing, moiré-like patterns arise routinely when sampling media print, such as newspapers and magazines, or in images with periodic components whose spacing is comparable to the spacing between samples. It is important to note that moiré patterns are more general than sampling artifacts. For instance, Fig. 4.20 shows the moiré effect using vector drawings that have not been digitized. Separately, the patterns are clean and void of interference. However, the simple acts of superimposing one pattern on the other creates a pattern with frequencies not present in either of the original patterns. Note in particular the moiré effect produced by two patterns of dots, as this is the effect of interest in the following discussion.

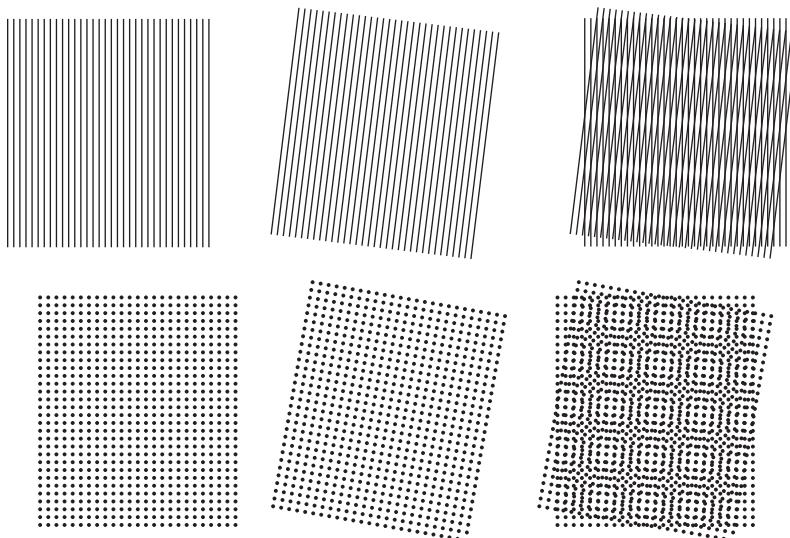
EXAMPLE 4.9: Sampling printed media.

Newspapers and other printed materials use so called *halftone dots*, which are black dots or ellipses whose sizes and various grouping schemes are used to simulate gray tones. As a rule, the following numbers are typical: newspapers are printed using 75 halftone dots per inch (dpi), magazines use 133 dpi, and

a	b	c
d	e	f

FIGURE 4.20

Examples of the moiré effect.
These are vector drawings, not digitized patterns.
Superimposing one pattern on the other is analogous to multiplying the patterns.



high-quality brochures use 175 dpi. Figure 4.21 shows what happens when a newspaper image is (under)sampled at 75 dpi. The sampling lattice (which is oriented vertically and horizontally) and dot patterns on the newspaper image (oriented at $\pm 45^\circ$) interact to create a uniform moiré-like pattern that makes the image look blotchy. (We will discuss a technique in Section 4.10 for reducing the effects of moiré patterns in under-sampled print media.)

FIGURE 4.21

A newspaper image digitized at 75 dpi. Note the moiré-like pattern resulting from the interaction between the $\pm 45^\circ$ orientation of the half-tone dots and the north-south orientation of the sampling elements used to digitized the image.



THE 2-D DISCRETE FOURIER TRANSFORM AND ITS INVERSE

A development similar to the material in Sections 4.3 and 4.4 would yield the following 2-D discrete Fourier transform (DFT):

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)} \quad (4-67)$$

where $f(x, y)$ is a digital image of size $M \times N$. As in the 1-D case, Eq. (4-67) must be evaluated for values of the discrete variables u and v in the ranges $u = 0, 1, 2, \dots, M - 1$ and $v = 0, 1, 2, \dots, N - 1$.[†]

Given the transform $F(u, v)$, we can obtain $f(x, y)$ by using the *inverse discrete Fourier transform* (IDFT):

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)} \quad (4-68)$$

for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$. As in the 1-D case, [Eqs. (4-44) and (4-45)], Eqs. (4-67) and (4-68) constitute a 2-D *discrete Fourier transform pair*, $f(x, y) \Leftrightarrow F(u, v)$. (The proof is a straightforward extension of the 1-D case in Problem 4.15.) The rest of this chapter is based on properties of these two equations and their use for image filtering in the frequency domain. The comments made in connection with Eqs. (4-44) and (4-45) are applicable to Eqs. (4-67) and (4-68); that is, knowing that $f(x, y)$ and $F(u, v)$ are a Fourier transform pair can be quite useful in proving relationships between functions and their transforms.

4.6 SOME PROPERTIES OF THE 2-D DFT AND IDFT

In this section, we introduce several properties of the 2-D discrete Fourier transform and its inverse.

RELATIONSHIPS BETWEEN SPATIAL AND FREQUENCY INTERVALS

The relationships between spatial sampling and the corresponding frequency domain intervals are as explained in Section 4.4. Suppose that a continuous function $f(t, z)$ is sampled to form a digital image, $f(x, y)$, consisting of $M \times N$ samples taken in the t - and z -directions, respectively. Let ΔT and ΔZ denote the separations between samples (see Fig. 4.15). Then, the separations between the corresponding discrete, frequency domain variables are given by

$$\Delta u = \frac{1}{M\Delta T} \quad (4-69)$$

[†]As mentioned in Section 4.4, keep in mind that in this chapter we use (t, z) and (μ, ν) to denote 2-D *continuous* spatial and frequency-domain variables. In the 2-D *discrete* case, we use (x, y) for spatial variables and (u, v) for frequency-domain variables, all of which are discrete.

and

$$\Delta v = \frac{1}{N \Delta Z} \quad (4-70)$$

respectively. Note the important property that the separations between samples in the frequency domain are inversely proportional both to the spacing between spatial samples *and* to the number of samples.

TRANSLATION AND ROTATION

The validity of the following Fourier transform pairs can be demonstrated by direct substitution into Eqs. (4-67) and (4-68) (see Problem 4.27):

$$f(x, y) e^{j2\pi(u_0x/M + v_0y/N)} \Leftrightarrow F(u - u_0, v - v_0) \quad (4-71)$$

and

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi(x_0u/M + y_0v/N)} \quad (4-72)$$

That is, multiplying $f(x, y)$ by the exponential shown shifts the origin of the DFT to (u_0, v_0) and, conversely, multiplying $F(u, v)$ by the negative of that exponential shifts the origin of $f(x, y)$ to (x_0, y_0) . As we illustrate in Example 4.13, translation has no effect on the magnitude (spectrum) of $F(u, v)$.

Using the polar coordinates

$$x = r \cos \theta \quad y = r \sin \theta \quad u = \omega \cos \varphi \quad v = \omega \sin \varphi$$

results in the following transform pair:

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0) \quad (4-73)$$

which indicates that rotating $f(x, y)$ by an angle θ_0 rotates $F(u, v)$ by the same angle. Conversely, rotating $F(u, v)$ rotates $f(x, y)$ by the same angle.

PERIODICITY

As in the 1-D case, the 2-D Fourier transform and its inverse are infinitely periodic in the u and v directions; that is,

$$F(u, v) = F(u + k_1 M, v) = F(u, v + k_2 N) = F(u + k_1 M, v + k_2 N) \quad (4-74)$$

and

$$f(x, y) = f(x + k_1 M, y) = f(x, y + k_2 N) = f(x + k_1 M, y + k_2 N) \quad (4-75)$$

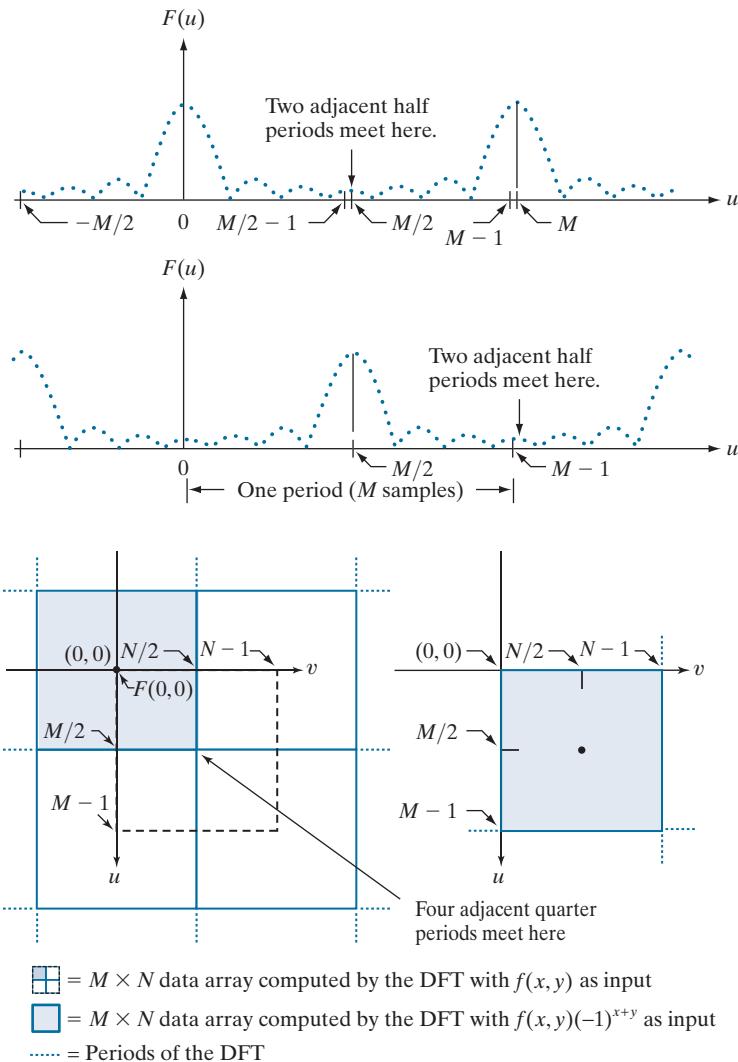
where k_1 and k_2 are integers.

The periodicities of the transform and its inverse are important issues in the implementation of DFT-based algorithms. Consider the 1-D spectrum in Fig. 4.22(a). As explained in Section 4.4 [see the footnote to Eq. (4-42)], the transform data in the interval from 0 to $M - 1$ consists of two half periods meeting at point $M/2$, but with

a
b
c d

FIGURE 4.22

Centering the Fourier transform.
 (a) A 1-D DFT showing an infinite number of periods. (b) Shifted DFT obtained by multiplying $f(x)$ by $(-1)^x$ before computing $F(u)$. (c) A 2-D DFT showing an infinite number of periods. The area within the dashed rectangle is the data array, $F(u, v)$, obtained with Eq. (4-67) with an image $f(x, y)$ as the input. This array consists of four quarter periods. (d) Shifted array obtained by multiplying $f(x, y)$ by $(-1)^{x+y}$ before computing $F(u, v)$. The data now contains one complete, centered period, as in (b).



the lower part of the period appearing at higher frequencies. For display and filtering purposes, it is more convenient to have in this interval a complete period of the transform in which the data are contiguous and ordered properly, as in Fig. 4.22(b). It follows from Eq. (4-71) that

$$f(x)e^{j2\pi(u_0x/M)} \Leftrightarrow F(u - u_0)$$

In other words, multiplying $f(x)$ by the exponential term shown shifts the transform data so that the origin, $F(0)$, is moved to u_0 . If we let $u_0 = M/2$, the exponential term becomes $e^{j\pi x}$, which is equal to $(-1)^x$ because x is an integer. In this case,

$$f(x)(-1)^x \Leftrightarrow F(u - M/2)$$

That is, multiplying $f(x)$ by $(-1)^x$ shifts the data so that $F(u)$ is centered on the interval $[0, M-1]$, which corresponds to Fig. 4.22(b), as desired.

In 2-D the situation is more difficult to graph, but the principle is the same, as Fig. 4.22(c) shows. Instead of two half periods, there are now four quarter periods meeting at the point $(M/2, N/2)$. As in the 1-D case, we want to shift the data so that $F(0,0)$ is at $(M/2, N/2)$. Letting $(u_0, v_0) = (M/2, N/2)$ in Eq. (4-71) results in the expression

$$f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2) \quad (4-76)$$

Using this equation shifts the data so that $F(0,0)$ is moved to the center of the *frequency rectangle* (i.e., the rectangle defined by the intervals $[0, M-1]$ and $[0, N-1]$ in the frequency domain). Figure 4.22(d) shows the result.

Keep in mind that in all our discussions, coordinate values in both the spatial and frequency domains are integers. As we explained in Section 2.4 (see Fig. 2.19) if, as in our case), the origin of an $M \times N$ image or transform is at $(0,0)$, then the center of that image or transform is at $(\text{floor}(M/2), \text{floor}(N/2))$. This expression is applicable to both even and odd values of M and N . For example, the center of an array of size 20×15 is at point $(10, 7)$. Because we start counting from 0, these are the 11th and 8th points in the first and second coordinate axes of the array, respectively.

SYMMETRY PROPERTIES

An important result from functional analysis is that any real *or* complex function, $w(x, y)$, can be expressed as the sum of an even and an odd part, each of which can be real or complex:

$$w(x, y) = w_e(x, y) + w_o(x, y) \quad (4-77)$$

where the *even* and *odd* parts are defined as

$$w_e(x, y) \triangleq \frac{w(x, y) + w(-x, -y)}{2} \quad (4-78)$$

and

$$w_o(x, y) \triangleq \frac{w(x, y) - w(-x, -y)}{2} \quad (4-79)$$

for all valid values of x and y . Substituting Eqs. (4-78) and (4-79) into Eq. (4-77) gives the identity $w(x, y) \equiv w(x, y)$, thus proving the validity of the latter equation. It follows from the preceding definitions that

$$w_e(x, y) = w_e(-x, -y) \quad (4-80)$$

and

$$w_o(x, y) = -w_o(-x, -y) \quad (4-81)$$

In the context of this discussion, the *locations* of elements in a sequence are denoted by integers. Therefore, the same observations made a few paragraphs back about the centers of arrays of even and odd *sizes* are applicable to sequences. But, do not confuse the concepts of even/odd *numbers* and even/odd *functions*.

To convince yourself that the samples of an odd function sum to zero, sketch one period of a 1-D sine wave about the origin or any other interval spanning one period.

Even functions are said to be *symmetric* and odd functions *antisymmetric*. Because all indices in the DFT and IDFT are nonnegative integers, when we talk about symmetry (antisymmetry) we are referring to symmetry (antisymmetry) about the *center point* of a sequence, in which case the definitions of even and odd become:

$$w_e(x, y) = w_e(M - x, N - y) \quad (4-82)$$

and

$$w_o(x, y) = -w_o(M - x, N - y) \quad (4-83)$$

for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$. As usual, M and N are the number of rows and columns of a 2-D array.

We know from elementary mathematical analysis that the product of two even or two odd functions is even, and that the product of an even and an odd function is odd. In addition, the only way that a discrete function can be odd is if all its samples sum to zero. These properties lead to the important result that

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} w_e(x, y) w_o(x, y) = 0 \quad (4-84)$$

for any two discrete even and odd functions w_e and w_o . In other words, because the argument of Eq. (4-84) is odd, the result of the summations is 0. The functions can be real or complex.

EXAMPLE 4.10: Even and odd functions.

Although evenness and oddness are visualized easily for continuous functions, these concepts are not as intuitive when dealing with discrete sequences. The following illustrations will help clarify the preceding ideas. Consider the 1-D sequence

$$f = \{f(0), f(1), f(2), f(3)\} = \{2, 1, 1, 1\}$$

in which $M = 4$. To test for evenness, the condition $f(x) = f(4 - x)$ must be satisfied for $x = 0, 1, 2, 3$. That is, we require that

$$f(0) = f(4), \quad f(1) = f(3), \quad f(2) = f(2), \quad f(3) = f(1)$$

Because $f(4)$ is outside the range being examined and can be any value, the value of $f(0)$ is immaterial in the test for evenness. We see that the next three conditions are satisfied by the values in the array, so the sequence is even. In fact, we conclude that *any* 4-point even sequence has to have the form

$$\{a, b, c, b\}$$

That is, only the second and last points must be equal in a 4-point even sequence. In general, when M is an even number, a 1-D even sequence has the property that the points at locations 0 and $M/2$ have

arbitrary values. When M is odd, the first point of an even sequence is still arbitrary, but the others form pairs with equal values.

Odd sequences have the interesting property that their first term, $w_o(0,0)$, is always 0, a fact that follows directly from Eq. (4-79). Consider the 1-D sequence

$$g = \{g(0), g(1), g(2), g(3)\} = \{0, -1, 0, 1\}$$

We can confirm that this is an odd sequence by noting that the terms in the sequence satisfy the condition $g(x) = -g(4-x)$ for $x = 1, 2, 3$. All we have to do for $x = 0$ is to check that $g(0) = 0$. We check the other terms using the definition. For example, $g(1) = -g(3)$. Any 4-point odd sequence has the form

$$\{0, -b, 0, b\}$$

In general, when M is an even number, a 1-D odd sequence has the property that the points at locations 0 and $M/2$ are always zero. When M is odd, the first term still has to be 0, but the remaining terms form pairs with equal value but opposite signs.

The preceding discussion indicates that evenness and oddness of sequences depend also on the length of the sequences. For example, we showed already that the sequence $\{0, -1, 0, 1\}$ is odd. However, the sequence $\{0, -1, 0, 1, 0\}$ is neither odd nor even, although the “basic” structure appears to be odd. This is an important issue in interpreting DFT results. We will show later in this section that the DFTs of even and odd functions have some very important characteristics. Thus, it often is the case that understanding when a function is odd or even plays a key role in our ability to interpret image results based on DFTs.

The same basic considerations hold in 2-D. For example, the 6×6 2-D array with center at location $(3,3)$, shown bold in the figure [remember, we start counting at $(0,0)$],

$$\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -2 & \mathbf{0} & 2 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

is odd, as you can prove using Eq. (4-83). However, adding another row or column of 0's would give a result that is neither odd nor even. In general, inserting a 2-D array of *even dimensions* into a larger array of zeros, *also* of even dimensions, preserves the symmetry of the smaller array, provided that the centers coincide. Similarly, a 2-D array of *odd dimensions* can be inserted into a larger array of zeros of *odd dimensions* without affecting the symmetry. Note that the inner structure of the preceding array is a Sobel kernel (see Fig. 3.50). We return to this kernel in Example 4.15, where we embed it in a larger array of zeros for filtering purposes.

Conjugate symmetry is also called *hermitian symmetry*. The term *antihermitian* is used sometimes to refer to conjugate antisymmetry.

Armed with the preceding concepts, we can establish a number of important symmetry properties of the DFT and its inverse. A property used frequently is that the Fourier transform of a *real* function, $f(x,y)$, is *conjugate symmetric*:

$$F^*(u, v) = F(-u, -v) \quad (4-85)$$

We show the validity of this equation as follows:

$$\begin{aligned} F^*(u, v) &= \left[\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)} \right]^* \\ &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f^*(x, y) e^{j2\pi(ux/M + vy/N)} \\ &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi([-u]x/M + [-v]y/N)} \\ &= F(-u, -v) \end{aligned}$$

where the third step follows from the fact that $f(x, y)$ is real. A similar approach can be used to prove that, if $f(x, y)$ is *imaginary*, its Fourier transform is conjugate *antisymmetric*; that is, $F^*(-u, -v) = -F(u, v)$.

Table 4.1 lists symmetries and related properties of the DFT that are useful in digital image processing. Recall that the double arrows indicate Fourier transform pairs; that is, for any row in the table, the properties on the right are satisfied by the Fourier transform of the function having the properties listed on the left, and vice versa. For example, entry 5 reads: The DFT of a real function $f(x, y)$, in which (x, y)

TABLE 4.1

Some symmetry properties of the 2-D DFT and its inverse. $R(u, v)$ and $I(u, v)$ are the real and imaginary parts of $F(u, v)$, respectively. Use of the word *complex* indicates that a function has nonzero real and imaginary parts.

	Spatial Domain[†]		Frequency Domain[†]
1)	$f(x, y)$ real	\Leftrightarrow	$F^*(u, v) = F(-u, -v)$
2)	$f(x, y)$ imaginary	\Leftrightarrow	$F^*(-u, -v) = -F(u, v)$
3)	$f(x, y)$ real	\Leftrightarrow	$R(u, v)$ even; $I(u, v)$ odd
4)	$f(x, y)$ imaginary	\Leftrightarrow	$R(u, v)$ odd; $I(u, v)$ even
5)	$f(-x, -y)$ real	\Leftrightarrow	$F^*(u, v)$ complex
6)	$f(-x, -y)$ complex	\Leftrightarrow	$F(-u, -v)$ complex
7)	$f^*(x, y)$ complex	\Leftrightarrow	$F^*(-u, -v)$ complex
8)	$f(x, y)$ real and even	\Leftrightarrow	$F(u, v)$ real and even
9)	$f(x, y)$ real and odd	\Leftrightarrow	$F(u, v)$ imaginary and odd
10)	$f(x, y)$ imaginary and even	\Leftrightarrow	$F(u, v)$ imaginary and even
11)	$f(x, y)$ imaginary and odd	\Leftrightarrow	$F(u, v)$ real and odd
12)	$f(x, y)$ complex and even	\Leftrightarrow	$F(u, v)$ complex and even
13)	$f(x, y)$ complex and odd	\Leftrightarrow	$F(u, v)$ complex and odd

[†]Recall that x, y, u , and v are *discrete* (integer) variables, with x and u in the range $[0, M - 1]$, and y and v in the range $[0, N - 1]$. To say that a complex function is *even* means that its real *and* imaginary parts are even, and similarly for an *odd* complex function. As before, “ \Leftrightarrow ” indicates a Fourier transform pair.

is replaced by $(-x, -y)$, is $F^*(u, v)$, the complex conjugate of the DFT of $f(x, y)$. Conversely, the IDFT of $F^*(u, v)$ is $f(-x, -y)$.

EXAMPLE 4.11: 1-D illustrations of the properties in Table 4.1.

The 1-D sequences (functions) and their transforms in Table 4.2 are short examples of the properties listed in Table 4.1. For example, in property 3 we see that a real function with elements $\{1, 2, 3, 4\}$ has a Fourier transform whose real part, $\{10, -2, -2, -2\}$, is even and whose imaginary part, $\{0, 2, 0, -2\}$, is odd. Property 8 tells us that a real even function has a transform that is real and even also. Property 12 shows that an even complex function has a transform that is also complex and even. The other listings in the table are analyzed in a similar manner.

EXAMPLE 4.12: Proving some of the DFT symmetry properties from Table 4.1.

In this example, we prove several of the properties in Table 4.1 to help you develop familiarity with manipulating these important properties, and to establish a basis for solving some of the problems at the end of the chapter. We prove only the properties on the right given the properties on the left. The converse is proved in a manner similar to the proofs we give here.

Consider property 3, which reads: If $f(x, y)$ is a real function, the real part of its DFT is even and the imaginary part is odd. We prove this property as follows: $F(u, v)$ is complex in general, so it can be expressed as the sum of a real and an imaginary part: $F(u, v) = R(u, v) + jI(u, v)$. Then, $F^*(u, v) = R(u, v) - jI(u, v)$. Also, $F(-u, -v) = R(-u, -v) + jI(-u, -v)$. But, as we proved earlier for Eq. (4-85), if $f(x, y)$ is real then $F^*(u, v) = F(-u, -v)$, which, based on the preceding two equations, means that $R(u, v) = R(-u, -v)$ and $I(u, v) = -I(-u, -v)$. In view of the definitions in Eqs. (4-80) and (4-81), this proves that R is an even function and that I is an odd function.

Next, we prove property 8. If $f(x, y)$ is real, we know from property 3 that the real part of $F(u, v)$ is even, so to prove property 8 all we have to do is show that if $f(x, y)$ is real and even then the imaginary part of $F(u, v)$ is 0 (i.e., F is real). The steps are as follows:

TABLE 4.2
1-D examples of
some of the prop-
erties in Table 4.1.

Property	$f(x)$	$F(u)$
3	$\{1, 2, 3, 4\}$	$\{(10 + 0j), (-2 + 2j), (-2 + 0j), (-2 - 2j)\}$
4	$\{1j, 2j, 3j, 4j\}$	$\{(0 + 2.5j), (.5 - .5j), (0 - .5j), (-.5 - .5j)\}$
8	$\{2, 1, 1, 1\}$	$\{5, 1, 1, 1\}$
9	$\{0, -1, 0, 1\}$	$\{(0 + 0j), (0 + 2j), (0 + 0j), (0 - 2j)\}$
10	$\{2j, 1j, 1j, 1j\}$	$\{5j, j, j, j\}$
11	$\{0j, -1j, 0j, 1j\}$	$\{0, -2, 0, 2\}$
12	$\{(4 + 4j), (3 + 2j), (0 + 2j), (3 + 2j)\}$	$\{(10 + 10j), (4 + 2j), (-2 + 2j), (4 + 2j)\}$
13	$\{(0 + 0j), (1 + 1j), (0 + 0j), (-1 - j)\}$	$\{(0 + 0j), (2 - 2j), (0 + 0j), (-2 + 2j)\}$

$$\begin{aligned}
\Im\{f(x,y)\} &= F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)} \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f_r(x,y)] e^{-j2\pi(ux/M + vy/N)} \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f_r(x,y)] e^{-j2\pi(ux/M)} e^{-j2\pi(vy/N)}
\end{aligned}$$

We can expand the last line of this expression in terms of even and odd parts

$$\begin{aligned}
F(u,v) &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even}][\text{even} - j\text{odd}][\text{even} - j\text{odd}] \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even}][\text{even} \cdot \text{even} - 2j\text{even} \cdot \text{odd} - \text{odd} \cdot \text{odd}] \\
&= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even} \cdot \text{even}] - 2j \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even} \cdot \text{odd}] - \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\text{even} \cdot \text{even}] \\
&= \text{real}.
\end{aligned}$$

The first step follows from Euler's equation, and the fact that the cos and sin are even and odd functions, respectively. We also know from property 8 that, in addition to being real, $f(x,y)$ is an even function. The only term in the penultimate line containing imaginary components is the second term, which is 0 according to Eq. (4-84). Therefore, if $f(x,y)$ is real and even, then $F(u,v)$ is real. As noted earlier, $F(u,v)$ is even also because $f(x,y)$ is real. This concludes the proof.

Finally, we demonstrate the validity of property 6. From the definition of the DFT,

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

We are not making a change of variable here. We are evaluating the DFT of $f(-x,-y)$, so we simply insert this function into the equation, as we would any other function. Because of periodicity, $f(-x,-y) = f(M-x, N-y)$. If we now define $m = M-x$ and $n = N-y$, then

$$\Im\{f(-x,-y)\} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j2\pi(u[M-m]/M + v[N-n]/N)}$$

To convince yourself that the summations are correct, try a 1-D transform and expand a few terms by hand. Because $\exp[-j2\pi(\text{integer})] = 1$, it follows that

$$\Im\{f(-x,-y)\} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{j2\pi(mu/M + vn/N)} = F(-u,-v)$$

This concludes the proof.

FOURIER SPECTRUM AND PHASE ANGLE

Because the 2-D DFT is complex in general, it can be expressed in polar form:

$$\begin{aligned} F(u, v) &= R(u, v) + jI(u, v) \\ &= |F(u, v)|e^{j\phi(u, v)} \end{aligned} \quad (4-86)$$

where the magnitude

$$|F(u, v)| = \left[R^2(u, v) + I^2(u, v) \right]^{1/2} \quad (4-87)$$

is called the *Fourier* (or *frequency*) *spectrum*, and

$$\phi(u, v) = \arctan \left[\frac{I(u, v)}{R(u, v)} \right] \quad (4-88)$$

is the *phase angle* or *phase spectrum*. Recall from the discussion in Section 4.2 that the arctan must be computed using a four-quadrant arctangent function, such as MATLAB's `atan2(imag, Real)` function.

Finally, the *power spectrum* is defined as

$$\begin{aligned} P(u, v) &= |F(u, v)|^2 \\ &= R^2(u, v) + I^2(u, v) \end{aligned} \quad (4-89)$$

As before, R and I are the real and imaginary parts of $F(u, v)$, and all computations are carried out for the discrete variables $u = 0, 1, 2, \dots, M - 1$ and $v = 0, 1, 2, \dots, N - 1$. Therefore, $|F(u, v)|$, $\phi(u, v)$, and $P(u, v)$ are arrays of size $M \times N$.

The Fourier transform of a real function is conjugate symmetric [see Eq. (4-85)], which implies that the spectrum has *even* symmetry about the origin:

$$|F(u, v)| = |F(-u, -v)| \quad (4-90)$$

The phase angle exhibits *odd* symmetry about the origin:

$$\phi(u, v) = -\phi(-u, -v) \quad (4-91)$$

It follows from Eq. (4-67) that

$$F(0, 0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

which indicates that the zero-frequency term of the DFT is proportional to the average of $f(x, y)$. That is,

$$\begin{aligned} F(0, 0) &= MN \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \\ &= MN \bar{f} \end{aligned} \quad (4-92)$$

where \bar{f} (a scalar) denotes the average value of $f(x, y)$. Then,

$$|F(0,0)| = MN |\bar{f}| \quad (4-93)$$

Because the proportionality constant MN usually is large, $|F(0,0)|$ typically is the largest component of the spectrum by a factor that can be several orders of magnitude larger than other terms. Because frequency components u and v are zero at the origin, $F(0,0)$ sometimes is called the *dc component* of the transform. This terminology is from electrical engineering, where “dc” signifies direct current (i.e., current of zero frequency).

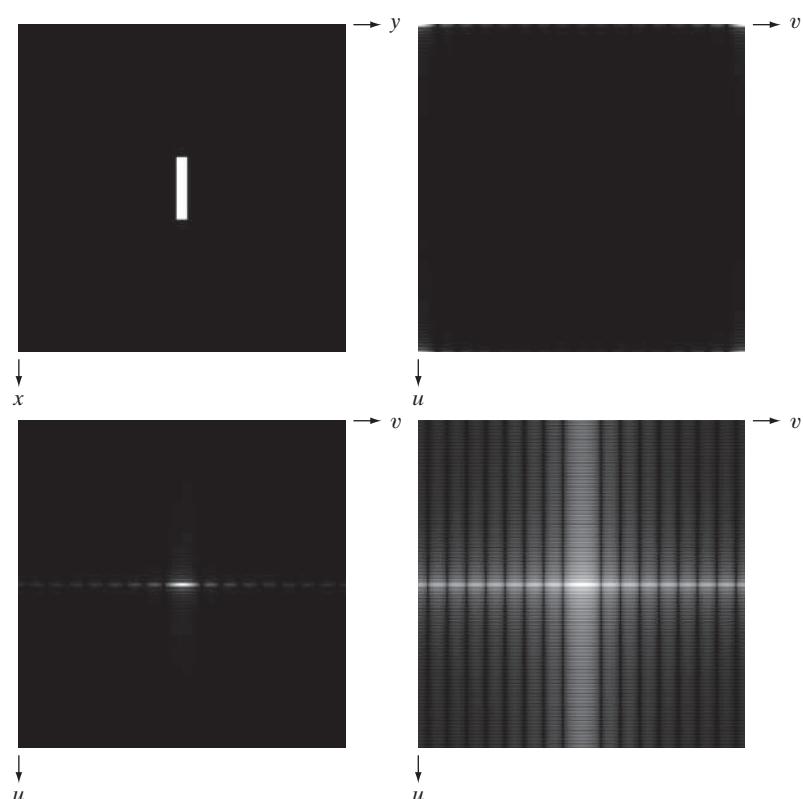
EXAMPLE 4.13: The spectrum of a rectangle.

Figure 4.23(a) shows an image of a rectangle and Fig. 4.23(b) shows its spectrum, whose values were scaled to the range [0, 255] and displayed in image form. The origins of both the spatial and frequency domains are at the top left. This is the right-handed coordinate system convention we defined in Fig. 2.19. Two things are apparent in Fig. 4.23(b). As expected, the area around the origin of the transform contains the highest values (and thus appears brighter in the image). However, note that the four corners

a	b
c	d

FIGURE 4.23

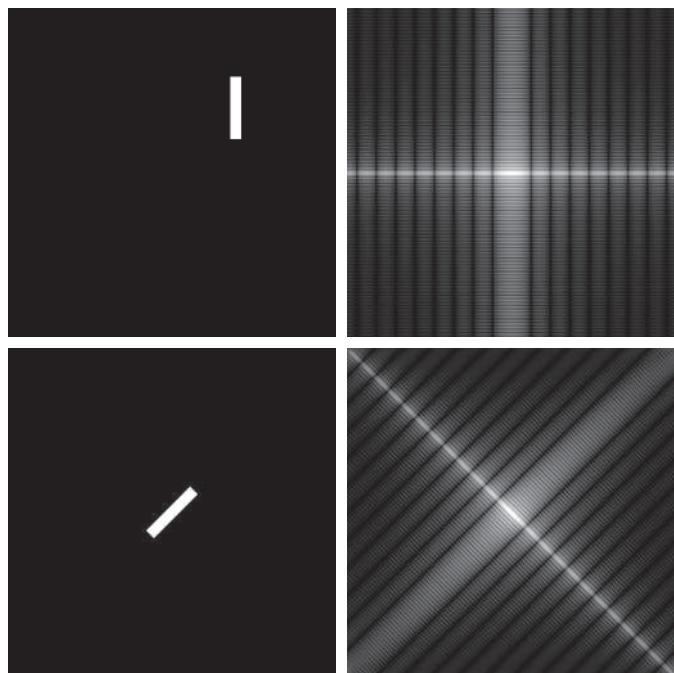
- (a) Image.
- (b) Spectrum, showing small, bright areas in the four corners (you have to look carefully to see them).
- (c) Centered spectrum.
- (d) Result after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The right-handed coordinate convention used in the book places the origin of the spatial and frequency domains at the top left (see Fig. 2.19).



a	b
c	d

FIGURE 4.24

- (a) The rectangle in Fig. 4.23(a) translated.
 (b) Corresponding spectrum.
 (c) Rotated rectangle.
 (d) Corresponding spectrum.
 The spectrum of the translated rectangle is identical to the spectrum of the original image in Fig. 4.23(a).



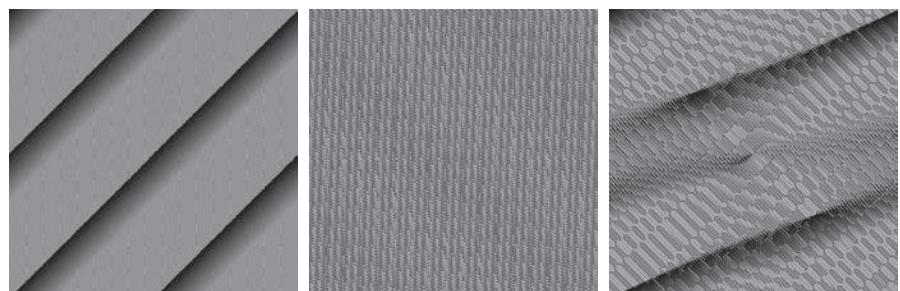
of the spectrum contain similarly high values. The reason is the periodicity property discussed in the previous section. To center the spectrum, we simply multiply the image in (a) by $(-1)^{x+y}$ before computing the DFT, as indicated in Eq. (4-76). Figure 4.23(c) shows the result, which clearly is much easier to visualize (note the symmetry about the center point). Because the dc term dominates the values of the spectrum, the dynamic range of other intensities in the displayed image are compressed. To bring out those details, we used the log transformation defined in Eq. (3-4) with $c = 1$. Figure 4.23(d) shows the display of $\log(1 + |F(u, v)|)$. The increased rendition of detail is evident. Most spectra shown in this and subsequent chapters are scaled in this manner.

It follows from Eqs. (4-72) and (4-73) that the spectrum is insensitive to image translation (the absolute value of the exponential term is 1), but it rotates by the same angle of a rotated image. Figure 4.24 illustrates these properties. The spectrum in Fig. 4.24(b) is identical to the spectrum in Fig. 4.23(d).

a	b	c
---	---	---

FIGURE 4.25

- Phase angle images of
 (a) centered,
 (b) translated,
 and (c) rotated rectangles.



Clearly, the images in Figs. 4.23(a) and 4.24(a) are different so, if their Fourier spectra are the same, then, based on Eq. (4-86), their phase angles must be different. Figure 4.25 confirms this. Figures 4.25(a) and (b) are the phase angle arrays (shown as images) of the DFTs of Figs. 4.23(a) and 4.24(a). Note the lack of similarity between the phase images, in spite of the fact that the only differences between their corresponding images is simple translation. In general, visual analysis of phase angle images yields little intuitive information. For instance, because of its 45° orientation, one would expect intuitively that the phase angle in Fig. 4.25(a) should correspond to the rotated image in Fig. 4.24(c), rather than to the image in Fig. 4.23(a). In fact, as Fig. 4.25(c) shows, the phase angle of the rotated image has a strong orientation that is much less than 45° .

The components of the spectrum of the DFT determine the amplitudes of the sinusoids that combine to form an image. At any given frequency in the DFT of an image, a large amplitude implies a greater prominence of a sinusoid of that frequency in the image. Conversely, a small amplitude implies that less of that sinusoid is present in the image. Although, as Fig. 4.25 shows, the contribution of the phase components is less intuitive, it is just as important. The phase is a measure of displacement of the various sinusoids with respect to their origin. Thus, while the magnitude of the 2-D DFT is an array whose components determine the intensities in the image, the corresponding phase is an array of angles that carry much of the information about where discernible objects are located in the image. The following example illustrates these ideas in more detail.

EXAMPLE 4.14: Contributions of the spectrum and phase angle to image formation.

Figure 4.26(b) shows as an image the phase-angle array, $\phi(u, v)$, of the DFT of Fig. 4.26(a), computed using Eq. (4-88). Although there is no detail in this array that would lead us by visual analysis to associate it with the structure of its corresponding image, the information in this array is crucial in determining shape features of the image. To illustrate this, we reconstructed the boy's image using only its phase angle. The reconstruction consisted of computing the inverse DFT of Eq. (4-86) using $\phi(u, v)$, but setting $|F(u, v)| = 1$. Figure Fig. 4.26(c) shows the result (the original result had much less contrast than is shown; to bring out details important in this discussion, we scaled the result using Eqs. (2-31) and (2-32), and then enhanced it using histogram equalization). However, even after enhancement, it is evident that much of the intensity information has been lost (remember, that information is carried by the spectrum, which we did not use in the reconstruction). However, the *shape* features in 4.26(c) are unmistakably from Fig. 4.26(a). This illustrates vividly the importance of the phase angle in determining shape characteristics in an image.

Figure 4.26(d) was obtained by computing the inverse DFT Eq. (4-86), but using only the spectrum. This means setting the exponential term to 1, which in turn implies setting the phase angle to 0. The result is not unexpected. It contains only intensity information, with the dc term being the most dominant. There is no shape information in the image because the phase was set to zero.

Finally, Figs. 4.26(e) and (f) show yet again the dominance of the phase in determining the spatial feature content of an image. Figure 4.26(e) was obtained by computing the inverse DFT of Eq. (4-86) using the spectrum of the rectangle from Fig. 4.23(a) and the phase angle from the boy's image. The boy's features clearly dominate this result. Conversely, the rectangle dominates Fig. 4.26(f), which was computed using the spectrum of the boy's image and the phase angle of the rectangle.



a	b	c
d	e	f

FIGURE 4.26 (a) Boy image. (b) Phase angle. (c) Boy image reconstructed using only its phase angle (all shape features are there, but the intensity information is missing because the spectrum was not used in the reconstruction). (d) Boy image reconstructed using only its spectrum. (e) Boy image reconstructed using its phase angle and the spectrum of the rectangle in Fig. 4.23(a). (f) Rectangle image reconstructed using its phase and the spectrum of the boy's image.

THE 2-D DISCRETE CONVOLUTION THEOREM

You will find it helpful to review Eq. (4-48), and the comments made there regarding circular convolution, as opposed to the convolution we studied in Section 3.4.

Extending Eq. (4-48) to two variables results in the following expression for 2-D *circular convolution*:

$$(f \star h)(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n) \quad (4-94)$$

for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$. As in Eq. (4-48), Eq. (4-94) gives one period of a 2-D periodic sequence. The 2-D convolution theorem is give by

$$(f \star h)(x, y) \Leftrightarrow (F \bullet H)(u, v) \quad (4-95)$$

and, conversely,

$$(f \bullet h)(x, y) \Leftrightarrow \frac{1}{MN} (F \star H)(u, v) \quad (4-96)$$

The function products are elementwise products, as defined in Section 2.6.

where F and H are the Fourier transforms of f and h , respectively, obtained using Eq. (4-67). As before, the double arrow is used to indicate that the left and right sides of the expressions constitute a Fourier transform pair. Our interest in the remainder of this chapter is in Eq. (4-95), which states that the Fourier transform of the spatial convolution of f and h , is the product of their transforms. Similarly, the inverse DFT of the product $(F \bullet H)(u, v)$ yields $(f \star h)(x, y)$.

Equation (4-95) is the foundation of linear filtering in the frequency domain and, as we will explain in Section 4.7, is the basis for all the filtering techniques discussed in this chapter. As you will recall from Chapter 3, spatial convolution is the foundation for spatial filtering, so Eq. (4-95) is the tie that establishes the equivalence between spatial and frequency-domain filtering, as we have mentioned several times before.

Ultimately, we are interested in the results of convolution in the spatial domain, where we analyze images. However, the convolution theorem tell us that we have two ways of computing the spatial convolution of two functions. We can do it directly in the spatial domain with Eq. (3-35), using the approach described in Section 3.4 or, according to Eq. (4-95), we can compute the Fourier transform of each function, multiply the transforms, and compute the inverse Fourier transform. Because we are dealing with discrete quantities, computation of the Fourier transforms is carried out using a DFT algorithm. This automatically implies periodicity, which means that when we take the inverse Fourier transform of the product of the two transforms we would get a circular (i.e., periodic) convolution, one period of which is given by Eq. (4-94). The question is: under what conditions will the direct spatial approach and the inverse Fourier transform method yield the same result? We arrive at the answer by looking at a 1-D example first, and then extending the results to two variables.

The left column of Fig. 4.27 implements convolution of two functions, f and h , using the 1-D equivalent of Eq. (3-35), which, because the two functions are of same size, is written as

$$(f \star h)(x) = \sum_{m=0}^{399} f(x)h(x-m)$$

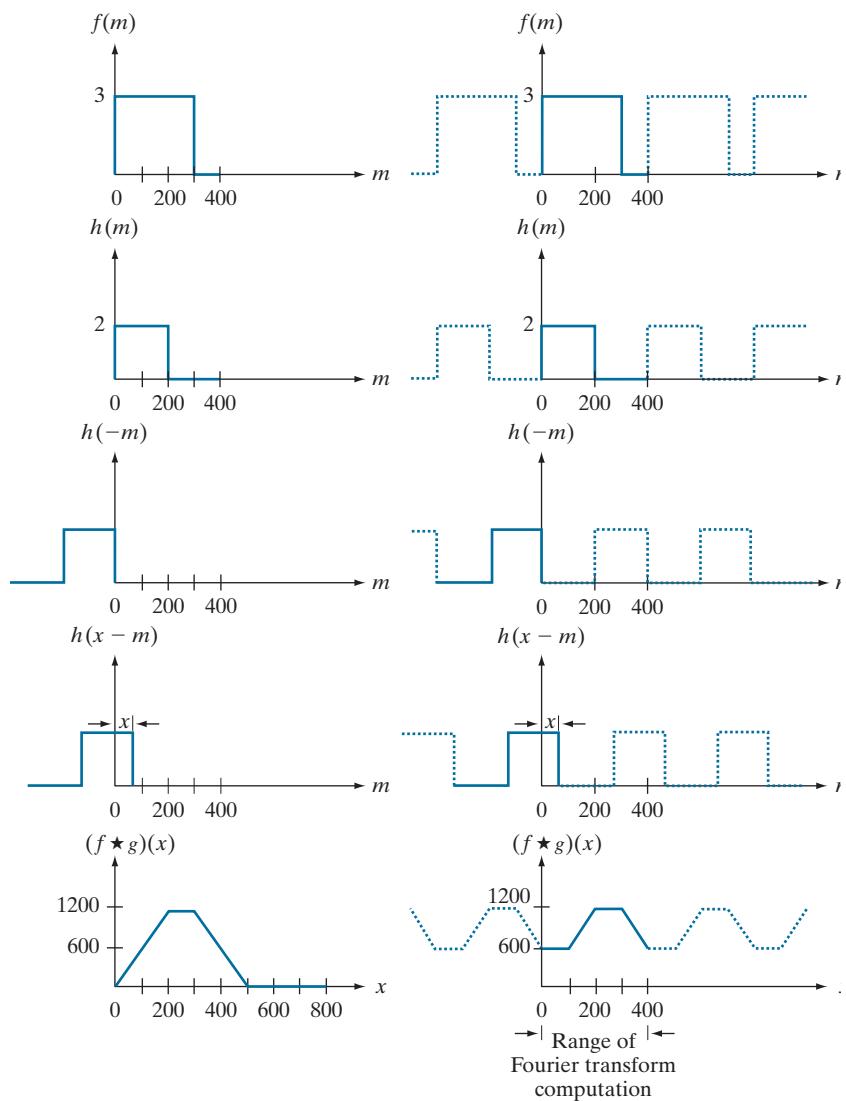
We will discuss efficient ways for computing the DFT in Section 4.11.

Recall from our explanation of Figs. 3.29 and 3.30 that the procedure consists of (1) rotating (flipping) h by 180°, [see Fig. 4.27(c)], (2) translating the resulting function by an amount x [Fig. 4.27(d)], and (3) for *each* value x of translation, computing the entire sum of products in the right side of the equation. In terms of Fig. 4.27, this means multiplying the function in Fig. 4.27(a) by the function in Fig. 4.27(d) for *each* value of x . The displacement x ranges over all values required to completely slide h across f . Figure 4.27(e) shows the convolution of these two functions. As you know, convolution is a function of the displacement variable, x , and the range of x required in this example to completely slide h past f is from 0 to 799.

a	f
b	g
c	h
d	i
e	j

FIGURE 4.27

Left column: Spatial convolution computed with Eq. (3-35), using the approach discussed in Section 3.4. Right column: Circular convolution. The solid line in (j) is the result we would obtain using the DFT, or, equivalently, Eq. (4-48). This erroneous result can be remedied by using zero padding.



If we use the DFT and the convolution theorem to try to obtain the same result as in the left column of Fig. 4.27, we must take into account the periodicity inherent in the expression for the DFT. This is equivalent to convolving the two periodic functions in Figs. 4.27(f) and (g) (i.e., as Eqs. (4-46) and (4-47) indicate, the functions their transforms have implied periodicity). The convolution procedure is the same as we just discussed, but the two functions now are periodic. Proceeding with these two functions as in the previous paragraph would yield the result in Fig. 4.27(j), which obviously is incorrect. Because we are convolving two periodic functions, the convolution itself is periodic. The closeness of the periods in Fig. 4.27 is such that

they interfere with each other to cause what is commonly referred to as *wraparound error*. According to the convolution theorem, if we had computed the DFT of the two 400-point functions, f and h , multiplied the two transforms, and then computed the inverse DFT, we would have obtained the erroneous 400-point segment of the periodic convolution shown as a solid line in Fig. 4.27(j) (remember the limits of the 1-D DFT are $u = 0, 1, 2, \dots, M - 1$). This is also the result we would obtain if we used Eq. (4-48) [the 1-D equivalent of Eq. (4-94)] to compute one period of the circular convolution.

Fortunately, the solution to the wraparound error problem is simple. Consider two functions, $f(x)$ and $h(x)$ composed of A and B samples, respectively. It can be shown (Brigham [1988]) that if we append zeros to both functions so that they have the same length, denoted by P , then wraparound is avoided by choosing

$$P \geq A + B - 1 \quad (4-97)$$

The padding zeros could be appended also at the beginning of the functions, or they could be divided between the beginning and end of the functions. It is simpler to append them at the end.

We use zero-padding here for simplicity. Recall from the discussion of Fig. 3.39 that replicate and mirror padding generally yield better results.

In our example, each function has 400 points, so the minimum value we could use is $P = 799$, which implies that we would append 399 zeros to the trailing edge of each function. This procedure is called *zero padding*, as we discussed in Section 3.4. As an exercise, you should convince yourself that if the periods of the functions in Figs. 4.27(f) and (g) were lengthened by appending to each period at least 399 zeros, the result would be a periodic convolution in which each period is identical to the correct result in Fig. 4.27(e). Using the DFT via the convolution theorem would result in a 799-point spatial function identical to Fig. 4.27(e). The conclusion, then, is that to obtain the same convolution result between the “straight” representation of the convolution equation approach in Chapter 3, and the DFT approach, functions in the latter must be padded prior to computing their transforms.

Visualizing a similar example in 2-D is more difficult, but we would arrive at the same conclusion regarding wraparound error and the need for appending zeros to the functions. Let $f(x, y)$ and $h(x, y)$ be two image arrays of sizes $A \times B$ and $C \times D$ pixels, respectively. Wraparound error in their circular convolution can be avoided by padding these functions with zeros, as follows:

$$f_p(x, y) = \begin{cases} f(x, y) & 0 \leq x \leq A - 1 \text{ and } 0 \leq y \leq B - 1 \\ 0 & A \leq x \leq P \text{ or } B \leq y \leq Q \end{cases} \quad (4-98)$$

and

$$h_p(x, y) = \begin{cases} h(x, y) & 0 \leq x \leq C - 1 \text{ and } 0 \leq y \leq D - 1 \\ 0 & C \leq x \leq P \text{ or } D \leq y \leq Q \end{cases} \quad (4-99)$$

with

$$P \geq A + C - 1 \quad (4-100)$$

and

$$Q \geq B + D - 1 \quad (4-101)$$

The resulting padded images are of size $P \times Q$. If both arrays are of the same size, $M \times N$, then we require that $P \geq 2M - 1$ and $Q \geq 2N - 1$. As a rule, DFT algorithms tend to execute faster with arrays of even size, so it is good practice to select P and Q as the smallest even integers that satisfy the preceding equations. If the two arrays are of the same size, this means that P and Q are selected as:

$$P = 2M \quad (4-102)$$

and

$$Q = 2N \quad (4-103)$$

Figure 4.31 in the next section illustrates the effects of wraparound error on images.

The two functions in Figs. 4.27(a) and (b) conveniently become zero before the end of the sampling interval. If one or both of the functions were not zero at the end of the interval, then a discontinuity would be created when zeros were appended to the function to eliminate wraparound error. This is analogous to multiplying a function by a box, which in the frequency domain would imply convolution of the original transform with a sinc function (see Example 4.1). This, in turn, would create so-called *frequency leakage*, caused by the high-frequency components of the sinc function. Leakage produces a blocky effect on images. Although leakage can never be totally eliminated, it can be reduced significantly by multiplying the sampled function by another function that tapers smoothly to near zero at both ends of the sampled record. This idea is to dampen the sharp transitions (and thus the high frequency components) of the box. This approach, called *windowing* or *apodizing*, is an important consideration when fidelity in image reconstruction (as in high-definition graphics) is desired.

A simple apodizing function is a triangle, centered on the data record, which tapers to 0 at both ends of the record. This is called a *Bartlett window*. Other common windows are the Gaussian, the *Hamming* and the *Hann* windows.

SUMMARY OF 2-D DISCRETE FOURIER TRANSFORM PROPERTIES

Table 4.3 summarizes the principal DFT definitions introduced in this chapter. We will discuss the separability property in Section 4.11, where we also show how to obtain the inverse DFT using a forward transform algorithm. Correlation will be discussed in detail Chapter 12.

Table 4.4 summarizes some important DFT pairs. Although our focus is on discrete functions, the last two entries in the table are Fourier transform pairs that can be derived only for continuous variables (note the use of continuous variable notation). We include them here because, with proper interpretation, they are quite useful in digital image processing. The differentiation pair can be used to derive the frequency-domain equivalent of the Laplacian defined in Eq. (3-50) (see Problem 4.52). The Gaussian pair is discussed in Section 4.7. Tables 4.1, 4.3 and 4.4 provide a summary of properties useful when working with the DFT. Many of these properties are key elements in the development of the material in the rest of this chapter, and some are used in subsequent chapters.

TABLE 4.3

Summary of DFT definitions and corresponding expressions.

	Name	Expression(s)
1)	Discrete Fourier transform (DFT) of $f(x,y)$	$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)}$
2)	Inverse discrete Fourier transform (IDFT) of $F(u,v)$	$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M+vy/N)}$
3)	Spectrum	$ F(u,v) = [R^2(u,v) + I^2(u,v)]^{1/2} \quad R = \text{Real}(F); I = \text{Imag}(F)$
4)	Phase angle	$\phi(u,v) = \tan^{-1} \left[\frac{I(u,v)}{R(u,v)} \right]$
5)	Polar representation	$F(u,v) = F(u,v) e^{j\phi(u,v)}$
6)	Power spectrum	$P(u,v) = F(u,v) ^2$
7)	Average value	$\bar{f} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) = \frac{1}{MN} F(0,0)$
8)	Periodicity (k_1 and k_2 are integers)	$\begin{aligned} F(u,v) &= F(u+k_1M, v) = F(u, v+k_2N) \\ &= F(u+k_1, v+k_2N) \\ f(x,y) &= f(x+k_1M, y) = f(x, y+k_2N) \\ &= f(x+k_1, y+k_2N) \end{aligned}$
9)	Convolution	$(f \star h)(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)h(x-m, y-n)$
10)	Correlation	$(f \star\!\! \star h)(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m,n)h(x+m, y+n)$
11)	Separability	The 2-D DFT can be computed by computing 1-D DFT transforms along the rows (columns) of the image, followed by 1-D transforms along the columns (rows) of the result. See Section 4.11.
12)	Obtaining the IDFT using a DFT algorithm	$MNf^*(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u,v) e^{-j2\pi(ux/M+vy/N)}$ This equation indicates that inputting $F^*(u,v)$ into an algorithm that computes the forward transform (right side of above equation) yields $MNf^*(x,y)$. Taking the complex conjugate and dividing by MN gives the desired inverse. See Section 4.11.

TABLE 4.4

Summary of DFT pairs. The closed-form expressions in 12 and 13 are valid only for continuous variables. They can be used with discrete variables by sampling the continuous expressions.

Name	DFT Pairs
1) Symmetry properties	$f(x,y) \Leftrightarrow F(u,v)$
2) Linearity	$a f_1(x,y) + b f_2(x,y) \Leftrightarrow a F_1(u,v) + b F_2(u,v)$
3) Translation (general)	$f(x,y) e^{j2\pi(u_0x/M + v_0y/N)} \Leftrightarrow F(u-u_0, v-v_0)$ $f(x-x_0, y-y_0) \Leftrightarrow F(u,v) e^{-j2\pi(ux_0/M + vy_0/N)}$
4) Translation to center of the frequency rectangle, $(M/2, N/2)$	$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2, v-N/2)$ $f(x-M/2, y-N/2) \Leftrightarrow F(u,v)(-1)^{u+v}$
5) Rotation	$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$ $r = \sqrt{x^2 + y^2} \quad \theta = \tan^{-1}(y/x) \quad \omega = \sqrt{u^2 + v^2} \quad \varphi = \tan^{-1}(v/u)$
6) Convolution theorem [†]	$(f \star h)(x,y) \Leftrightarrow (F \bullet H)(u,v)$ $(f \bullet h)(x,y) \Leftrightarrow (1/MN)[(F \star H)(u,v)]$
7) Correlation theorem [†]	$(f \star \hat{h})(x,y) \Leftrightarrow (F^* \bullet H)(u,v)$ $(F^* \bullet h)(x,y) \Leftrightarrow (1/MN)[(F \star \hat{H})(u,v)]$
8) Discrete unit impulse	$\delta(x,y) \Leftrightarrow 1$ $1 \Leftrightarrow MN\delta(u,v)$
9) Rectangle	$\text{rect}[a,b] \Leftrightarrow ab \frac{\sin(\pi ua)}{(\pi ua)} \frac{\sin(\pi vb)}{(\pi vb)} e^{-j\pi(ua+vb)}$
10) Sine	$\sin(2\pi u_0 x/M + 2\pi v_0 y/N) \Leftrightarrow \frac{jMN}{2} [\delta(u+u_0, v+v_0) - \delta(u-u_0, v-v_0)]$
11) Cosine	$\cos(2\pi u_0 x/M + 2\pi v_0 y/N) \Leftrightarrow \frac{1}{2} [\delta(u+u_0, v+v_0) + \delta(u-u_0, v-v_0)]$
12) Differentiation (the expressions on the right assume that $f(\pm\infty, \pm\infty) = 0$.)	$\left(\frac{\partial}{\partial t} \right)^m \left(\frac{\partial}{\partial z} \right)^n f(t,z) \Leftrightarrow (j2\pi\mu)^m (j2\pi\nu)^n F(\mu, \nu)$ $\frac{\partial^m f(t,z)}{\partial t^m} \Leftrightarrow (j2\pi\mu)^m F(\mu, \nu); \quad \frac{\partial^n f(t,z)}{\partial z^n} \Leftrightarrow (j2\pi\nu)^n F(\mu, \nu)$
13) Gaussian	$A 2\pi\sigma^2 e^{-2\pi^2\sigma^2(t^2+z^2)} \Leftrightarrow A e^{-(\mu^2+\nu^2)/2\sigma^2} \quad (A \text{ is a constant})$

[†] Assumes that $f(x,y)$ and $h(x,y)$ have been properly padded. Convolution is associative, commutative, and distributive. Correlation is distributive (see Table 3.5). The products are elementwise products (see Section 2.6).

4.7 THE BASICS OF FILTERING IN THE FREQUENCY DOMAIN

In this section, we lay the groundwork for all the filtering techniques discussed in the remainder of the chapter.

ADDITIONAL CHARACTERISTICS OF THE FREQUENCY DOMAIN

We begin by observing in Eq. (4-67) that *each* term of $F(u, v)$ contains *all* values of $f(x, y)$, modified by the values of the exponential terms. Thus, with the exception of trivial cases, it is usually impossible to make direct associations between specific components of an image and its transform. However, some general statements can be made about the relationship between the frequency components of the Fourier transform and spatial features of an image. For instance, because frequency is directly related to spatial rates of change, it is not difficult intuitively to associate frequencies in the Fourier transform with patterns of intensity variations in an image. We showed in Section 4.6 that the slowest varying frequency component ($u = v = 0$) is proportional to the average intensity of an image. As we move away from the origin of the transform, the low frequencies correspond to the slowly varying intensity components of an image. In an image of a room, for example, these might correspond to smooth intensity variations on the walls and floor. As we move further away from the origin, the higher frequencies begin to correspond to faster and faster intensity changes in the image. These are the edges of objects and other components of an image characterized by abrupt changes in intensity.

Filtering techniques in the frequency domain are based on modifying the Fourier transform to achieve a specific objective, and then computing the inverse DFT to get us back to the spatial domain, as introduced in Section 2.6. It follows from Eq. (4-87) that the two components of the transform to which we have access are the transform magnitude (spectrum) and the phase angle. We learned in Section 4.6 that visual analysis of the phase component generally is not very useful. The spectrum, however, provides some useful guidelines as to the gross intensity characteristics of the image from which the spectrum was generated. For example, consider Fig. 4.28(a), which is a scanning electron microscope image of an integrated circuit, magnified approximately 2500 times.

Aside from the interesting construction of the device itself, we note two principal features in this image: strong edges that run approximately at $\pm 45^\circ$, and two white, oxide protrusions resulting from thermally induced failure. The Fourier spectrum in Fig. 4.28(b) shows prominent components along the $\pm 45^\circ$ directions that correspond to the edges just mentioned. Looking carefully along the vertical axis in Fig. 4.28(b), we see a vertical component of the transform that is off-axis, slightly to the left. This component was caused by the edges of the oxide protrusions. Note how the angle of the frequency component with respect to the vertical axis corresponds to the inclination (with respect to the horizontal axis of the image) of the long white element. Note also the zeros in the vertical frequency component, corresponding to the narrow vertical span of the oxide protrusions.

These are typical of the types of associations we can make in general between the frequency and spatial domains. As we will show later in this chapter, even these types of gross associations, coupled with the relationships mentioned previously

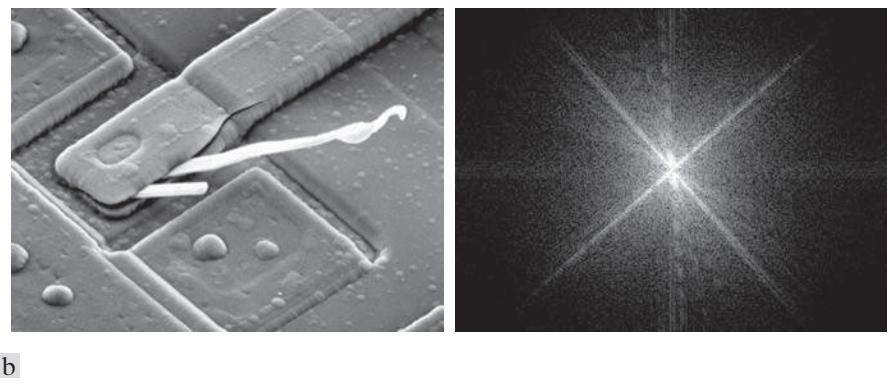


FIGURE 4.28 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

between frequency content and rate of change of intensity levels in an image, can lead to some very useful results. We will show in Section 4.8 the effects of modifying various frequency ranges in the transform of Fig. 4.28(a).

FREQUENCY DOMAIN FILTERING FUNDAMENTALS

Filtering in the frequency domain consists of modifying the Fourier transform of an image, then computing the inverse transform to obtain the spatial domain representation of the processed result. Thus, given (a padded) digital image, $f(x, y)$, of size $P \times Q$ pixels, the basic filtering equation in which we are interested has the form:

If H is real and symmetric and f is real (as is typically the case), then the IDFT in Eq. (4-104) should yield real quantities in theory. In practice, the inverse often contains parasitic complex terms from roundoff error and other computational inaccuracies. Thus, it is customary to take the real part of the IDFT to form g .

$$g(x, y) = \text{Real} \left\{ \mathfrak{F}^{-1} [H(u, v)F(u, v)] \right\} \quad (4-104)$$

where \mathfrak{F}^{-1} is the IDFT, $F(u, v)$ is the DFT of the input image, $f(x, y)$, $H(u, v)$ is a *filter transfer function* (which we often call just a *filter* or *filter function*), and $g(x, y)$ is the *filtered (output) image*. Functions F, H , and g are arrays of size $P \times Q$, the same as the padded input image. The product $H(u, v)F(u, v)$ is formed using elementwise multiplication, as defined in Section 2.6. The filter transfer function modifies the transform of the input image to yield the processed output, $g(x, y)$. The task of specifying $H(u, v)$ is simplified considerably by using functions that are symmetric about their center, which requires that $F(u, v)$ be centered also. As explained in Section 4.6, this is accomplished by multiplying the input image by $(-1)^{x+y}$ prior to computing its transform.[†]

[†]Some software implementations of the 2-D DFT (e.g., MATLAB) do not center the transform. This implies that filter functions must be arranged to correspond to the same data format as the uncentered transform (i.e., with the origin at the top left). The net result is that filter transfer functions are more difficult to generate and display. We use centering in our discussions to aid in visualization, which is crucial in developing a clear understanding of filtering concepts. Either method can be used in practice, provided that consistency is maintained.

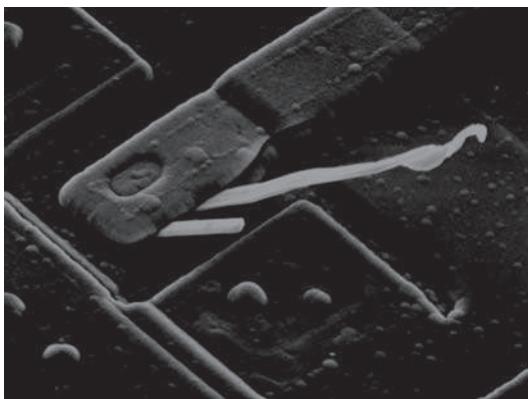
We are now in a position to consider filtering in detail. One of the simplest filter transfer functions we can construct is a function $H(u,v)$ that is 0 at the center of the (centered) transform, and 1's elsewhere. This filter would reject the dc term and "pass" (i.e., leave unchanged) all other terms of $F(u,v)$ when we form the product $H(u,v)F(u,v)$. We know from property 7 in Table 4.3 that the dc term is responsible for the average intensity of an image, so setting it to zero will reduce the average intensity of the output image to zero. Figure 4.29 shows the result of this operation using Eq. (4-104). As expected, the image became much darker. An average of zero implies the existence of negative intensities. Therefore, although it illustrates the principle, Fig. 4.29 is not a true representation of the original, as all negative intensities were clipped (set to 0) by the display.

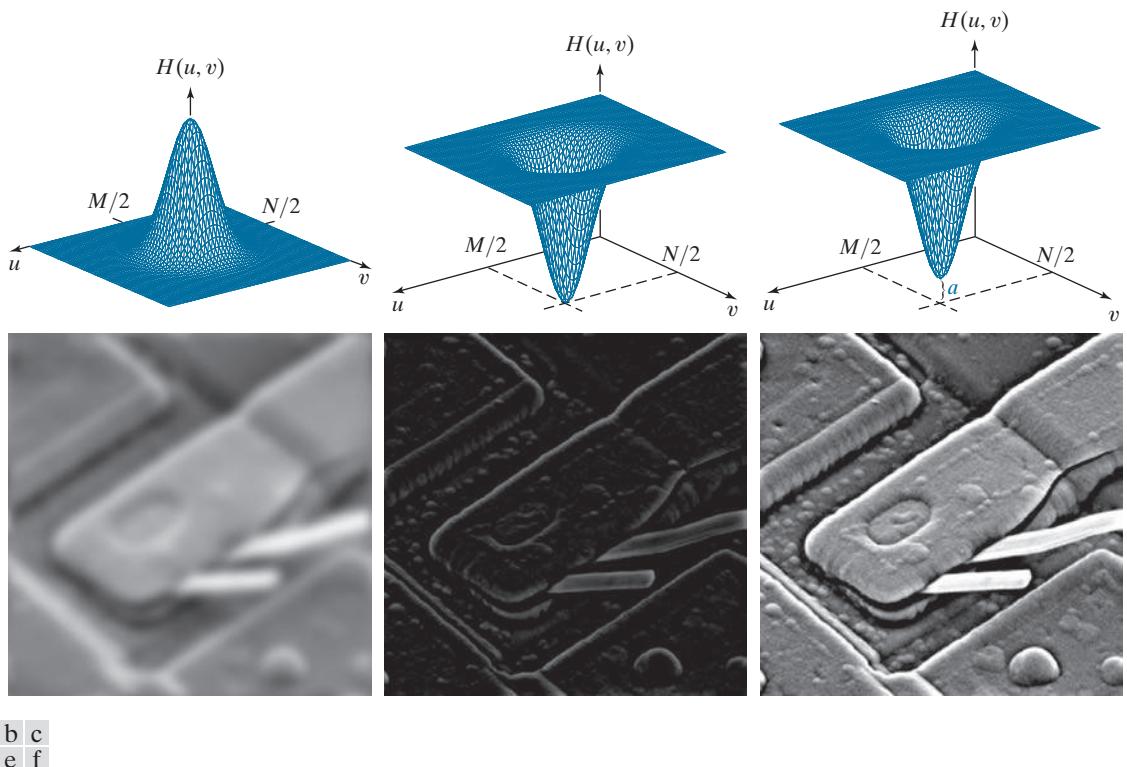
As noted earlier, low frequencies in the transform are related to slowly varying intensity components in an image, such as the walls of a room or a cloudless sky in an outdoor scene. On the other hand, high frequencies are caused by sharp transitions in intensity, such as edges and noise. Therefore, we would expect that a function $H(u,v)$ that attenuates high frequencies while passing low frequencies (called a *lowpass filter*, as noted before) would blur an image, while a filter with the opposite property (called a *highpass filter*) would enhance sharp detail, but cause a reduction in contrast in the image. Figure 4.30 illustrates these effects. For example, the first column of this figure shows a lowpass filter transfer function and the corresponding filtered image. The second column shows similar results for a highpass filter. Note the similarity between Figs. 4.30(e) and Fig. 4.29. The reason is that the highpass filter function shown eliminates the dc term, resulting in the same basic effect that led to Fig. 4.29. As illustrated in the third column, adding a small constant to the filter does not affect sharpening appreciably, but it does prevent elimination of the dc term and thus preserves tonality.

Equation (4-104) involves the product of two functions in the frequency domain which, by the convolution theorem, implies convolution in the spatial domain. We know from the discussion in Section 4.6 that we can expect wraparound error if the functions in question are not padded. Figure 4.31 shows what happens when

FIGURE 4.29

Result of filtering the image in Fig. 4.28(a) with a filter transfer function that sets to 0 the dc term, $F(P/2, Q/2)$, in the centered Fourier transform, while leaving all other transform terms unchanged.





a	b	c
d	e	f

FIGURE 4.30 Top row: Frequency domain filter transfer functions of (a) a lowpass filter, (b) a highpass filter, and (c) an offset highpass filter. Bottom row: Corresponding filtered images obtained using Eq. (4-104). The offset in (c) is $a = 0.85$, and the height of $H(u, v)$ is 1. Compare (f) with Fig. 4.28(a).



a	b	c
---	---	---

FIGURE 4.31 (a) A simple image. (b) Result of blurring with a Gaussian lowpass filter without padding. (c) Result of lowpass filtering with zero padding. Compare the vertical edges in (b) and (c).

we apply Eq. (4-104) without padding. Figure 4.31(a) shows a simple image, and Fig. 4.31(b) is the result of lowpass filtering the image with a Gaussian lowpass filter of the form shown in Fig. 4.30(a). As expected, the image is blurred. However, the blurring is not uniform; the top white edge is blurred, but the sides are not. Padding the input image with zeros according to Eqs. (4-98) and (4-99) before applying Eq. (4-104) resulted in the filtered image in Fig. 4.31(c). This result is as expected, with a uniform dark border resulting from zero padding (see Fig. 3.33 for an explanation of this effect).

Figure 4.32 illustrates the reason for the discrepancy between Figs. 4.31(b) and (c). The dashed area in Fig. 4.32(a) corresponds to the image in Fig. 4.31(a). The other copies of the image are due to the implied periodicity of the image (and its transform) implicit when we use the DFT, as explained in Section 4.6. Imagine convolving the spatial representation of the blurring filter (i.e., the corresponding spatial kernel) with this image. When the kernel is centered on the top of the dashed image, it will encompass part of the image and also part of the bottom of the periodic image immediately above it. When a dark and a light region reside under the filter, the result is a mid-gray, blurred output. However, when the kernel is centered on the top right side of the image, it will encompass only light areas in the image and its right region. Because the average of a constant value is that same value, filtering will have no effect in this area, giving the result in Fig. 4.31(b). Padding the image with 0's creates a uniform border around each image of the periodic sequence, as Fig. 4.32(b) shows. Convolving the blurring kernel with the padded “mosaic” of Fig. 4.32(b) gives the correct result in Fig. 4.31(c). You can see from this example that failure to pad an image prior to filtering can lead to unexpected results.

Thus far, the discussion has centered on padding the input image. However, Eq. (4-104) also involves a filter transfer function that can be specified either in the

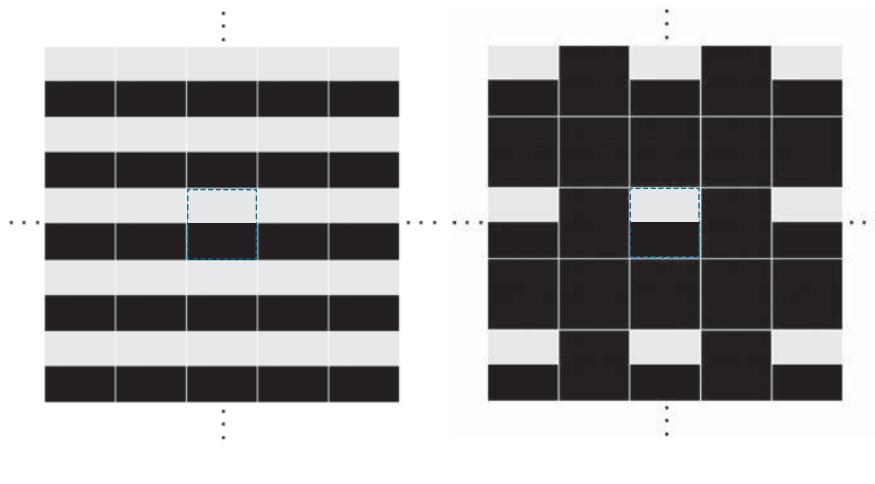


FIGURE 4.32 (a) Image periodicity without image padding. (b) Periodicity after padding with 0's (black). The dashed areas in the center correspond to the image in Fig. 4.31(a). Periodicity is inherent when using the DFT. (The thin white lines in both images are superimposed for clarity; they are not part of the data.)

spatial or in the frequency domain. But padding is done in the spatial domain, which raises an important question about the relationship between *spatial* padding and filter functions specified directly in the frequency domain.

It would be reasonable to conclude that the way to handle padding of a frequency domain transfer function is to construct the function the same size as the unpad-ded image, compute the IDFT of the function to obtain the corresponding spatial representation, pad that representation in the spatial domain, and then compute its DFT to return to the frequency domain. The 1-D example in Fig. 4.33 illustrates the pitfalls in this approach.

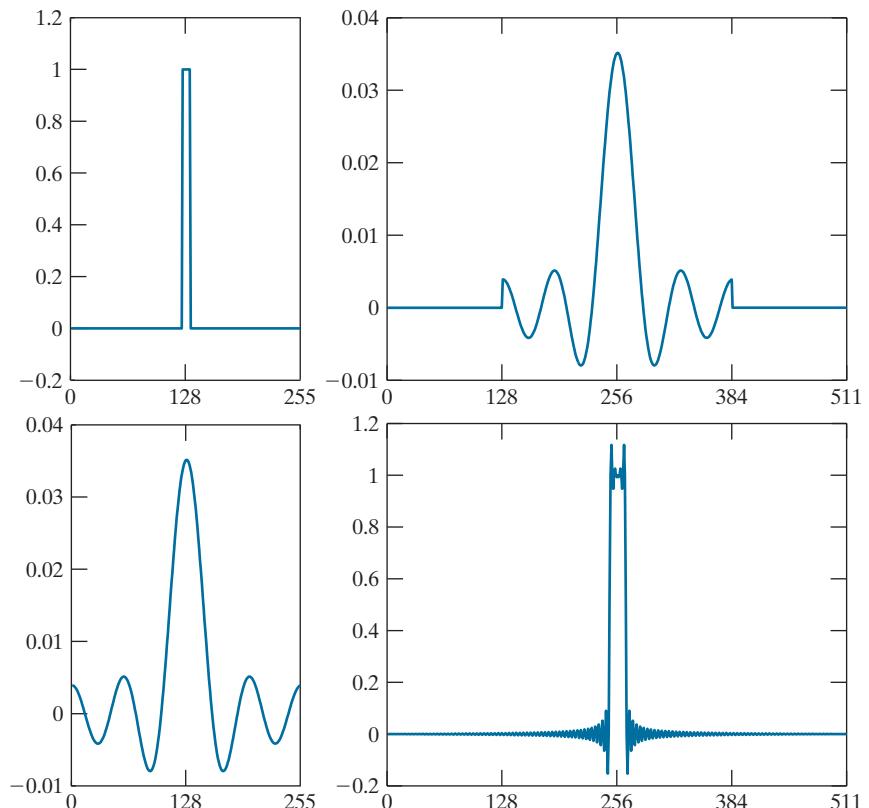
Figure 4.33(a) shows a 1-D ideal lowpass filter transfer function in the frequency domain. The function is real and has even symmetry, so we know from property 8 in Table 4.1 that its IDFT will be real and symmetric also. Figure 4.33(b) shows the result of multiplying the elements of the transfer function by $(-1)^u$ and computing its IDFT to obtain the corresponding spatial filter kernel. The result is shown in Fig. 4.33(b). It is evident in this figure that the extremes of this spatial function are not zero. Zero-padding the function would create two discontinuities, as Fig. 4.33(c) shows. To return to the frequency domain, we compute the forward DFT of the spatial, padded function. As Fig. 4.33(d) shows, the discontinuities in the padded function caused ringing in its frequency domain counterpart.

Padding the two ends of a function is the same as padding one end, provided that the total number of zeros is the same.

a c
b d

FIGURE 4.33

- (a) Filter transfer function specified in the (centered) frequency domain.
- (b) Spatial representation (filter kernel) obtained by computing the IDFT of (a).
- (c) Result of padding (b) to twice its length (note the discontinuities).
- (d) Corresponding filter in the frequency domain obtained by computing the DFT of (c). Note the ringing caused by the discontinuities in (c). Part (b) of the figure is below (a), and (d) is below (c).



The preceding results tell us that we cannot pad the spatial representation of a frequency domain transfer function in order to avoid wraparound error. Our objective is to work with specified filter shapes in the frequency domain without having to be concerned with truncation issues. An alternative is to pad images and then create the desired filter transfer function directly in the frequency domain, this function being of the same size as the padded images (remember, images and filter transfer functions must be of the same size when using the DFT). Of course, this will result in wraparound error because no padding is used for the filter transfer function, but this error is mitigated significantly by the separation provided by padding the image, and it is preferable to ringing. Smooth transfer functions (such as those in Fig. 4.30) present even less of a problem. Specifically, then, the approach we will follow in this chapter is to pad images to size $P \times Q$ and construct filter transfer functions of the same dimensions directly in the frequency domain. As explained earlier, P and Q are given by Eqs. (4-100) and (4-101).

We conclude this section by analyzing the phase angle of filtered images. We can express the DFT in terms of its real and imaginary parts: $F(u, v) = R(u, v) + jI(u, v)$. Equation (4-104) then becomes

$$g(x, y) = \mathfrak{F}^{-1}[H(u, v)R(u, v) + jH(u, v)I(u, v)] \quad (4-105)$$

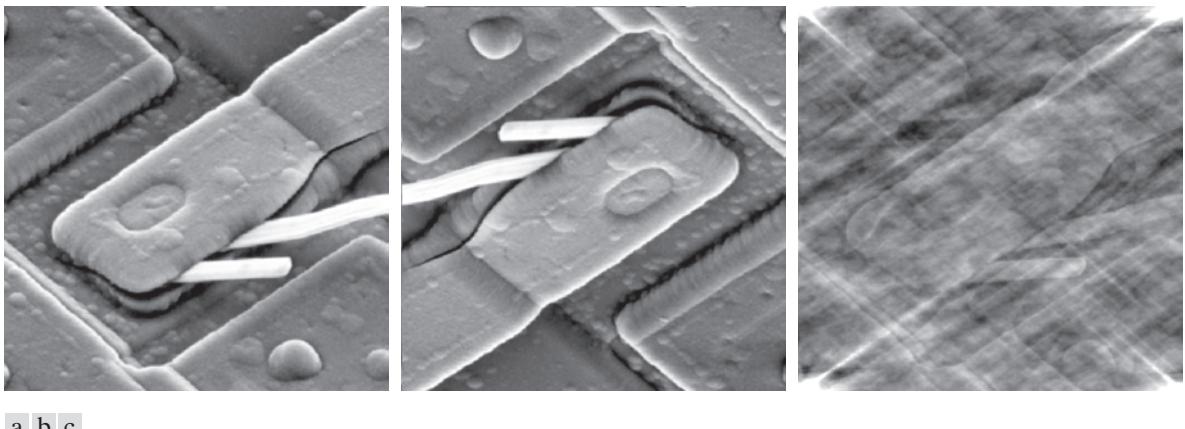
The phase angle is computed as the arctangent of the ratio of the imaginary and the real parts of a complex number [see Eq. (4-88)]. Because $H(u, v)$ multiplies both R and I , it will cancel out when this ratio is formed. Filters that affect the real and imaginary parts equally, and thus have no effect on the phase angle, are appropriately called *zero-phase-shift* filters. These are the only types of filters considered in this chapter.

The importance of the phase angle in determining the spatial structure of an image was vividly illustrated in Fig. 4.26. Thus, it should be no surprise that even small changes in the phase angle can have dramatic (and usually undesirable) effects on the filtered output. Figures 4.34(b) and (c) illustrate the effect of changing the phase angle array of the DFT of Fig. 4.34(a) (the $|F(u, v)|$ term was not changed in either case). Figure 4.34(b) was obtained by multiplying the phase angle, $\phi(u, v)$, in Eq. (4-86) by -1 and computing the IDFT. The net result is a reflection of every pixel in the image about both coordinate axes. Figure 4.34(c) was obtained by multiplying the phase term by 0.25 and computing the IDFT. Even a scale change rendered the image almost unrecognizable. These two results illustrate the advantage of using frequency-domain filters that do not alter the phase angle.

SUMMARY OF STEPS FOR FILTERING IN THE FREQUENCY DOMAIN

The process of filtering in the frequency domain can be summarized as follows:

- Given an input image $f(x, y)$ of size $M \times N$, obtain the padding sizes P and Q using Eqs. (4-102) and (4-103); that is, $P = 2M$ and $Q = 2N$.



a b c

FIGURE 4.34 (a) Original image. (b) Image obtained by multiplying the phase angle array by -1 in Eq. (4-86) and computing the IDFT. (c) Result of multiplying the phase angle by 0.25 and computing the IDFT. The magnitude of the transform, $|F(u,v)|$, used in (b) and (c) was the same.

2. Form a padded[†] image $f_p(x,y)$ of size $P \times Q$ using zero-, mirror-, or replicate padding (see Fig. 3.39 for a comparison of padding methods).
3. Multiply $f_p(x,y)$ by $(-1)^{x+y}$ to center the Fourier transform on the $P \times Q$ frequency rectangle.
4. Compute the DFT, $F(u,v)$, of the image from Step 3.
5. Construct a real, symmetric filter transfer function, $H(u,v)$, of size $P \times Q$ with center at $(P/2, Q/2)$.
6. Form the product $G(u,v) = H(u,v)F(u,v)$ using elementwise multiplication; that is, $G(i,k) = H(i,k)F(i,k)$ for $i = 0, 1, 2, \dots, M - 1$ and $k = 0, 1, 2, \dots, N - 1$.
7. Obtain the filtered image (of size $P \times Q$) by computing the IDFT of $G(u,v)$:

$$g_p(x,y) = \left(\text{real} \left[\mathfrak{F}^{-1} \{ G(u,v) \} \right] \right) (-1)^{x+y}$$

8. Obtain the final filtered result, $g(x,y)$, of the same size as the input image, by extracting the $M \times N$ region from the top, left quadrant of $g_p(x,y)$.

See Section 2.6 for a definition of elementwise operations.

We will discuss the construction of filter transfer functions (Step 5) in the following sections of this chapter. In theory, the IDFT in Step 7 should be real because $f(x,y)$ is real and $H(u,v)$ is real and symmetric. However, parasitic complex terms in the IDFT resulting from computational inaccuracies are not uncommon. Taking the real part of the result takes care of that. Multiplication by $(-1)^{x+y}$ cancels out the multiplication by this factor in Step 3.

[†]Sometimes we omit padding when doing “quick” experiments to get an idea of filter performance, or when trying to determine quantitative relationships between spatial features and their effect on frequency domain components, particularly in band and notch filtering, as explained later in Section 4.10 and in Chapter 5.

a	b	c
d	e	f
g	h	

FIGURE 4.35

- (a) An $M \times N$ image, f .
- (b) Padded image, f_p , of size $P \times Q$.
- (c) Result of multiplying f_p by $(-1)^{x+y}$.
- (d) Spectrum of F .
- (e) Centered Gaussian lowpass filter transfer function, H , of size $P \times Q$.
- (f) Spectrum of the product HF .
- (g) Image g_p , the real part of the IDFT of HF , multiplied by $(-1)^{x+y}$.
- (h) Final result, g , obtained by extracting the first M rows and N columns of g_p .

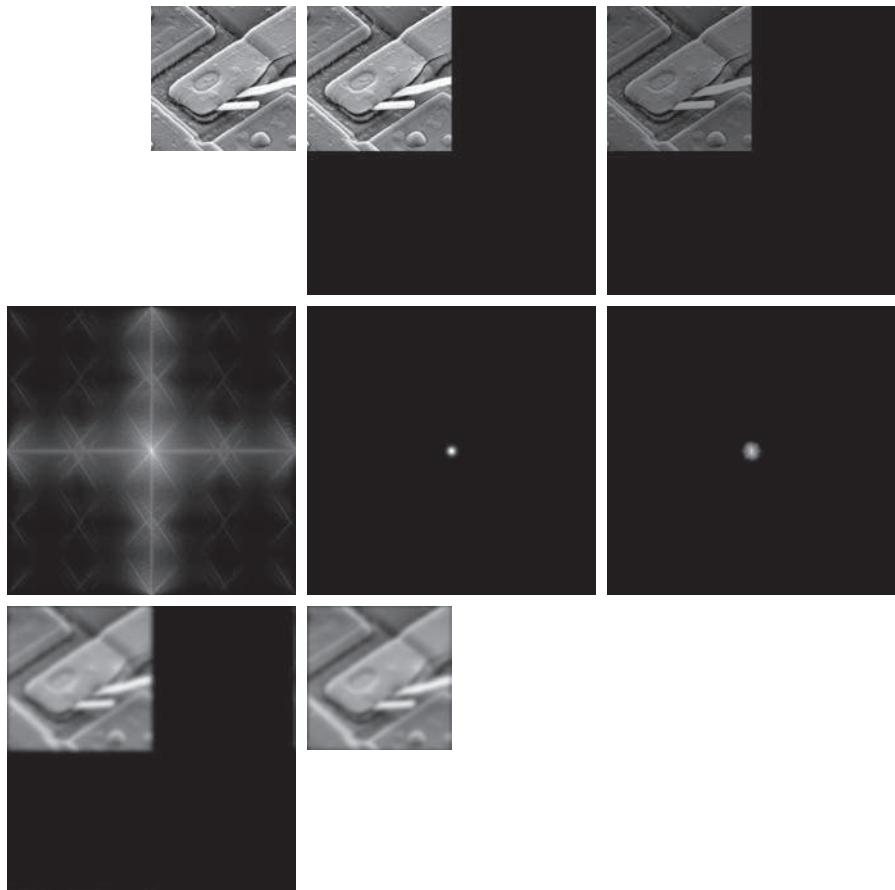


Figure 4.35 illustrates the preceding steps using zero padding. The figure legend explains the source of each image. If enlarged, Fig. 4.35(c) would show black dots interleaved in the image because negative intensities, resulting from the multiplication of f_p by $(-1)^{x+y}$, are clipped at 0 by the display. Note in Fig. 4.35(h) the characteristic dark border of lowpass filtered images obtained using zero padding.

CORRESPONDENCE BETWEEN FILTERING IN THE SPATIAL AND FREQUENCY DOMAINS

As mentioned several times before, the link between filtering in the spatial and frequency domains is the convolution theorem. Earlier in this section, we defined filtering in the frequency domain as the elementwise product of a filter transfer function, $H(u, v)$, and $F(u, v)$, the Fourier transform of the input image. Given $H(u, v)$, suppose that we want to find its equivalent kernel in the spatial domain. If we let $f(x, y) = \delta(x, y)$, it follows from Table 4.4 that $F(u, v) = 1$. Then, from Eq. (4-104), the filtered output is $\mathfrak{F}^{-1}\{H(u, v)\}$. This expression as the inverse transform of the frequency domain filter transfer function, which is the corresponding kernel in the

See Section 2.6 for a definition of elementwise operations.

spatial domain. Conversely, it follows from a similar analysis and the convolution theorem that, given a spatial filter kernel, we obtain its frequency domain representation by taking the forward Fourier transform of the kernel. Therefore, the two filters form a Fourier transform pair:

$$h(x, y) \Leftrightarrow H(u, v) \quad (4-106)$$

where $h(x, y)$ is the spatial kernel. Because this kernel can be obtained from the response of a frequency domain filter to an impulse, $h(x, y)$ sometimes is referred to as the *impulse response* of $H(u, v)$. Also, because all quantities in a discrete implementation of Eq. (4-106) are finite, such filters are called *finite impulse response* (FIR) filters. These are the only types of linear spatial filters considered in this book.

We discussed spatial convolution in Section 3.4, and its implementation in Eq. (3-35), which involved convolving functions of different sizes. When we use the DFT to compute the transforms used in the convolution theorem, it is implied that we are convolving periodic functions of the *same* size, as explained in Fig. 4.27. For this reason, as explained earlier, Eq. (4-94) is referred to as *circular convolution*.

When computational speed, cost, and size are important parameters, spatial convolution filtering using Eq. (3-35) is well suited for small kernels using hardware and/or firmware, as explained in Section 4.1. However, when working with general-purpose machines, frequency-domain methods in which the DFT is computed using a fast Fourier transform (FFT) algorithm can be hundreds of times faster than using spatial convolution, depending on the size of the kernels used, as you saw in Fig. 4.2. We will discuss the FFT and its computational advantages in Section 4.11.

Filtering concepts are more intuitive in the frequency domain, and filter design often is easier there. One way to take advantage of the properties of both domains is to specify a filter in the frequency domain, compute its IDFT, and then use the properties of the resulting, full-size spatial kernel as a guide for constructing smaller kernels. This is illustrated next (keep in mind that the Fourier transform and its inverse are linear processes (see Problem 4.24), so the discussion is limited to linear filtering). In Example 4.15, we illustrate the converse, in which a spatial kernel is given, and we obtain its full-size frequency domain representation. This approach is useful for analyzing the behavior of small spatial kernels in the frequency domain.

Frequency domain filters can be used as guides for specifying the coefficients of some of the small kernels we discussed in Chapter 3. Filters based on Gaussian functions are of particular interest because, as noted in Table 4.4, both the forward and inverse Fourier transforms of a Gaussian function are real Gaussian functions. We limit the discussion to 1-D to illustrate the underlying principles. Two-dimensional Gaussian transfer functions are discussed later in this chapter.

Let $H(u)$ denote the 1-D frequency domain Gaussian transfer function

$$H(u) = Ae^{-u^2/2\sigma^2} \quad (4-107)$$

where σ is the standard deviation of the Gaussian curve. The kernel in the spatial domain is obtained by taking the inverse DFT of $H(u)$ (see Problem 4.48):

$$h(x) = \sqrt{2\pi}\sigma A e^{-2\pi^2\sigma^2x^2} \quad (4-108)$$

As mentioned in Table 4.4, the forward and inverse Fourier transforms of Gaussians are valid only for continuous variables. To use discrete formulations, we sample the continuous forms.

These two equations are important for two reasons: (1) They are a Fourier transform pair, both components of which are Gaussian *and* real. This facilitates analysis because we do not have to be concerned with complex numbers. In addition, Gaussian curves are intuitive and easy to manipulate. (2) The functions behave reciprocally. When $H(u)$ has a broad profile (large value of σ), $h(x)$ has a narrow profile, and vice versa. In fact, as σ approaches infinity, $H(u)$ tends toward a constant function and $h(x)$ tends toward an impulse, which implies no filtering in either domain.

Figures 4.36(a) and (b) show plots of a Gaussian lowpass filter transfer function in the frequency domain and the corresponding function in the spatial domain. Suppose that we want to use the shape of $h(x)$ in Fig. 4.36(b) as a guide for specifying the coefficients of a small kernel in the spatial domain. The key characteristic of the function in Fig. 4.36(b) is that all its values are positive. Thus, we conclude that we can implement lowpass filtering in the spatial domain by using a kernel with all positive coefficients (as we did in Section 3.5). For reference, Fig. 4.36(b) also shows two of the kernels discussed in that section. Note the reciprocal relationship between the width of the Gaussian functions, as discussed in the previous paragraph. The narrower the frequency domain function, the more it will attenuate the low frequencies, resulting in increased blurring. In the spatial domain, this means that a larger kernel must be used to increase blurring, as we illustrated in Example 3.11.

As you know from Section 3.7, we can construct a highpass filter from a lowpass filter by subtracting a lowpass function from a constant. We working with Gaussian functions, we can gain a little more control over filter function shape by using a so-called *difference of Gaussians*, which involves two lowpass functions. In the frequency domain, this becomes

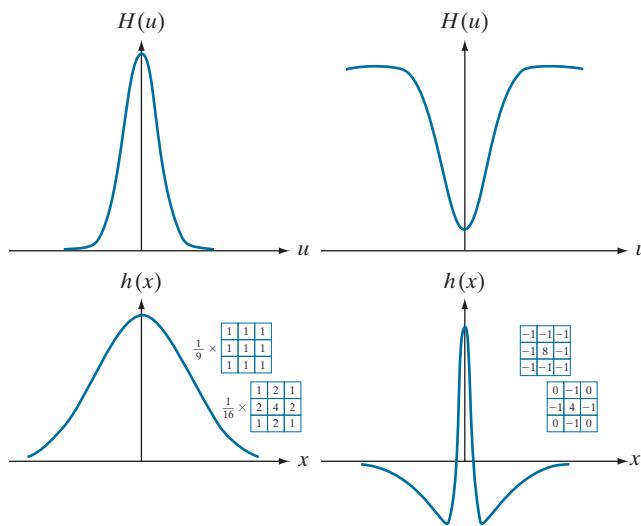
$$H(u) = A e^{-u^2/2\sigma_1^2} - B e^{-u^2/2\sigma_2^2} \quad (4-109)$$

with $A \geq B$ and $\sigma_1 > \sigma_2$. The corresponding function in the spatial domain is

a	c
b	d

FIGURE 4.36

- (a) A 1-D Gaussian lowpass transfer function in the frequency domain.
- (b) Corresponding kernel in the spatial domain.
- (c) Gaussian highpass transfer function in the frequency domain.
- (d) Corresponding kernel. The small 2-D kernels shown are kernels we used in Chapter 3.



$$h(x) = \sqrt{2\pi}\sigma_1 A e^{-2\pi^2\sigma_1^2 x^2} - \sqrt{2\pi}\sigma_2 B e^{-2\pi^2\sigma_2^2 x^2} \quad (4-110)$$

Figures 4.36(c) and (d) show plots of these two equations. We note again the reciprocity in width, but the most important feature here is that $h(x)$ has a positive center term with negative terms on either side. The small kernels shown in Fig. 4.36(d), which we used in Chapter 3 for sharpening, “capture” this property, and thus illustrate how knowledge of frequency domain filtering can be used as the basis for choosing coefficients of spatial kernels.

Although we have gone through significant effort to get here, be assured that it is impossible to truly understand filtering in the frequency domain without the foundation we have just established. In practice, the frequency domain can be viewed as a “laboratory” in which we take advantage of the correspondence between frequency content and image appearance. As will be demonstrated numerous times later in this chapter, some tasks that would be exceptionally difficult to formulate directly in the spatial domain become almost trivial in the frequency domain. Once we have selected a specific filter transfer function via experimentation in the frequency domain, we have the option of implementing the filter directly in that domain using the FFT, or we can take the IDFT of the transfer function to obtain the equivalent spatial domain function. As we showed in Fig. 4.36, one approach is to specify a small spatial kernel that attempts to capture the “essence” of the *full* filter function in the spatial domain. A more formal approach is to design a 2-D digital filter by using approximations based on mathematical or statistical criteria, as we discussed in Section 3.7.

EXAMPLE 4.15: Obtaining a frequency domain transfer function from a spatial kernel.

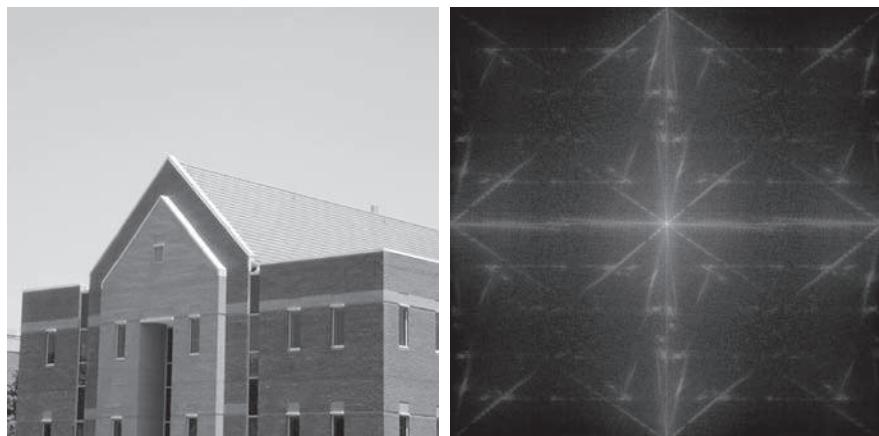
In this example, we start with a spatial kernel and show how to generate its corresponding filter transfer function in the frequency domain. Then, we compare the filtering results obtained using frequency domain and spatial techniques. This type of analysis is useful when one wishes to compare the performance of a given kernel against one or more “full” filter candidates in the frequency domain, or to gain a deeper understanding about the performance of a kernel in the spatial domain. To keep matters simple, we use the 3×3 vertical Sobel kernel from Fig. 3.50(e). Figure 4.37(a) shows a 600×600 -pixel image, $f(x, y)$, that we wish to filter, and Fig. 4.37(b) shows its spectrum.

Figure 4.38(a) shows the Sobel kernel, $h(x, y)$ (the perspective plot is explained below). Because the input image is of size 600×600 pixels and the kernel is of size 3×3 , we avoid wraparound error in the frequency domain by padding f and h with zeros to size 602×602 pixels, according to Eqs. (4-100) and (4-101). At first glance, the Sobel kernel appears to exhibit odd symmetry. However, its first element is not 0, as required by Eq. (4-81). To convert the kernel to the smallest size that will satisfy Eq. (4-83), we have to add to it a leading row and column of 0’s, which turns it into an array of size 4×4 . We can embed this array into a larger array of zeros and still maintain its odd symmetry if the larger array is of even dimensions (as is the 4×4 kernel) and their centers coincide, as explained in Example 4.10. The preceding comments are an important aspect of filter generation. If we preserve the odd symmetry with respect to the padded array in forming $h_p(x, y)$, we know from property 9 in Table 4.1 that $H(u, v)$ will be purely imaginary. As we show at the end of this example, this will yield results that are identical to filtering the image spatially using the original kernel $h(x, y)$. If the symmetry were not preserved, the results would no longer be the same.

a b

FIGURE 4.37

(a) Image of a building, and
(b) its Fourier spectrum.



The procedure used to generate $H(u, v)$ is: (1) multiply $h_p(x, y)$ by $(-1)^{x+y}$ to center the frequency domain filter; (2) compute the forward DFT of the result in (1) to generate $H(u, v)$; (3) set the real part of $H(u, v)$ to 0 to account for parasitic real parts (we know that H has to be purely imaginary because h_p is real and odd); and (4) multiply the result by $(-1)^{u+v}$. This last step reverses the multiplication of $H(u, v)$ by $(-1)^{u+v}$, which is implicit when $h(x, y)$ was manually placed in the center of $h_p(x, y)$. Figure 4.38(a) shows a perspective plot of $H(u, v)$, and Fig. 4.38(b) shows $H(u, v)$ as an image. Note the antisymmetry in this image about its center, a result of $H(u, v)$ being odd. Function $H(u, v)$ is used as any other frequency domain filter transfer function. Figure 4.38(c) is the result of using the filter transfer function just obtained to filter the image in Fig. 4.37(a) in the frequency domain, using the step-by-step filtering procedure outlined earlier. As expected from a derivative filter, edges were enhanced and all the constant intensity areas were reduced to zero (the grayish tone is due to scaling for display). Figure 4.38(d) shows the result of filtering the same image in the spatial domain with the Sobel kernel $h(x, y)$, using the procedure discussed in Section 3.6. The results are identical.

4.8 IMAGE SMOOTHING USING LOWPASS FREQUENCY DOMAIN FILTERS

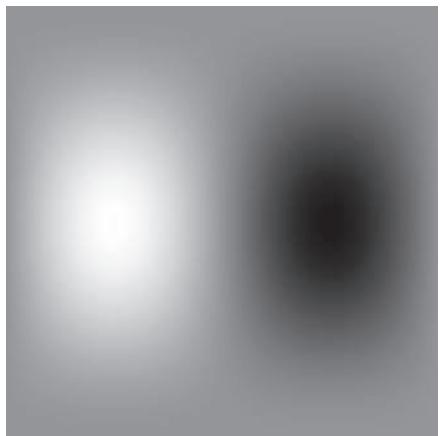
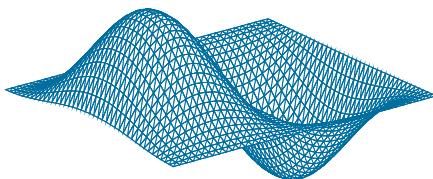
The remainder of this chapter deals with various filtering techniques in the frequency domain, beginning with lowpass filters. Edges and other sharp intensity transitions (such as noise) in an image contribute significantly to the high frequency content of its Fourier transform. Hence, smoothing (blurring) is achieved in the frequency domain by high-frequency attenuation; that is, by *lowpass* filtering. In this section, we consider three types of lowpass filters: *ideal*, *Butterworth*, and *Gaussian*. These three categories cover the range from very sharp (ideal) to very smooth (Gaussian) filtering. The shape of a Butterworth filter is controlled by a parameter called the *filter order*. For large values of this parameter, the Butterworth filter approaches the ideal filter. For lower values, the Butterworth filter is more like a Gaussian filter. Thus, the Butterworth filter provides a transition between two “extremes.” All filtering in this section follows the procedure outlined in the previous section, so all filter transfer functions, $H(u, v)$, are understood to be of size $P \times Q$; that is, the discrete

a	b
c	d

FIGURE 4.38

- (a) A spatial kernel and perspective plot of its corresponding frequency domain filter transfer function.
- (b) Transfer function shown as an image.
- (c) Result of filtering Fig. 4.37(a) in the frequency domain with the transfer function in (b).
- (d) Result of filtering the same image in the spatial domain with the kernel in (a). The results are identical.

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



frequency variables are in the range $u = 0, 1, 2, \dots, P - 1$ and $v = 0, 1, 2, \dots, Q - 1$, where P and Q are the padded sizes given by Eqs. (4-100) and (4-101).

IDEAL LOWPASS FILTERS

A 2-D lowpass filter that passes without attenuation all frequencies within a circle of radius from the origin, and “cuts off” all frequencies outside this circle is called an *ideal lowpass filter* (ILPF); it is specified by the transfer function

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases} \quad (4-111)$$

where D_0 is a positive constant, and $D(u, v)$ is the distance between a point (u, v) in the frequency domain and the center of the $P \times Q$ frequency rectangle; that is,

$$D(u, v) = \left[(u - P/2)^2 + (v - Q/2)^2 \right]^{1/2} \quad (4-112)$$

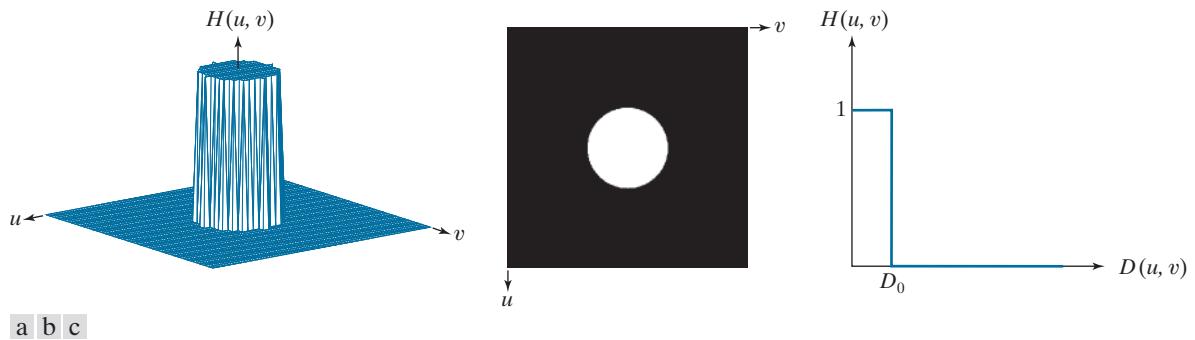


FIGURE 4.39 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Function displayed as an image. (c) Radial cross section.

where, as before, P and Q are the padded sizes from Eqs. (4-102) and (4-103). Figure 4.39(a) shows a perspective plot of transfer function $H(u,v)$ and Fig. 4.39(b) shows it displayed as an image. As mentioned in Section 4.3, the name *ideal* indicates that all frequencies on or inside a circle of radius D_0 are passed without attenuation, whereas all frequencies outside the circle are completely attenuated (filtered out). The ideal lowpass filter transfer function is radially symmetric about the origin. This means that it is defined completely by a radial cross section, as Fig. 4.39(c) shows. A 2-D representation of the filter is obtained by rotating the cross section 360° .

For an ILPF cross section, the point of transition between the values $H(u,v) = 1$ and $H(u,v) = 0$ is called the *cutoff frequency*. In Fig. 4.39, the cutoff frequency is D_0 . The sharp cutoff frequency of an ILPF cannot be realized with electronic components, although they certainly can be simulated in a computer (subject to the constraint that the fastest possible transition is limited by the distance between pixels).

The lowpass filters in this chapter are compared by studying their behavior as a function of the same cutoff frequencies. One way to establish standard cutoff frequency loci using circles that enclose specified amounts of total *image power* P_T , which we obtain by summing the components of the power spectrum of the padded images at each point (u,v) , for $u = 0, 1, 2, \dots, P - 1$ and $v = 0, 1, 2, \dots, Q - 1$; that is,

$$P_T = \sum_{u=0}^{P-1} \sum_{v=0}^{Q-1} P(u,v) \quad (4-113)$$

where $P(u,v)$ is given by Eq. (4-89). If the DFT has been centered, a circle of radius D_0 with origin at the center of the frequency rectangle encloses α percent of the power, where

$$\alpha = 100 \left[\sum_u \sum_v P(u,v) / P_T \right] \quad (4-114)$$

and the summation is over values of (u,v) that lie inside the circle or on its boundary.

Figures 4.40(a) and (b) show a test pattern image and its spectrum. The circles superimposed on the spectrum have radii of 10, 30, 60, 160, and 460 pixels,



a b

FIGURE 4.40 (a) Test pattern of size 688×688 pixels, and (b) its spectrum. The spectrum is double the image size as a result of padding, but is shown half size to fit. The circles have radii of 10, 30, 60, 160, and 460 pixels with respect to the full-size spectrum. The radii enclose 86.9, 92.8, 95.1, 97.6, and 99.4% of the padded image power, respectively.

respectively, and enclosed the percentages of total power listed in the figure caption. The spectrum falls off rapidly, with close to 87% of the total power being enclosed by a relatively small circle of radius 10. The significance of this will become evident in the following example.

EXAMPLE 4.16: Image smoothing in the frequency domain using lowpass filters.

Figure 4.41 shows the results of applying ILPFs with cutoff frequencies at the radii shown in Fig. 4.40(b). Figure 4.41(b) is useless for all practical purposes, unless the objective of blurring is to eliminate all detail in the image, except the “blobs” representing the largest objects. The severe blurring in this image is a clear indication that most of the sharp detail information in the image is contained in the 13% power removed by the filter. As the filter radius increases, less and less power is removed, resulting in less blurring. Note that the images in Figs. 4.41(c) through (e) contain significant “ringing,” which becomes finer in texture as the amount of high frequency content removed decreases. Ringing is visible even in the image in which only 2% of the total power was removed [Fig. 4.41(e)]. This ringing behavior is a characteristic of ideal filters, as we have mentioned several times before. Finally, the result for $\alpha = 99.4\%$ in Fig. 4.41(f) shows very slight blurring and almost imperceptible ringing but, for the most part, this image is close to the original. This indicates that little edge information is contained in the upper 0.6% of the spectrum power removed by the ILPF.

It is clear from this example that ideal lowpass filtering is not practical. However, it is useful to study the behavior of ILPFs as part of our development of filtering concepts. Also, as shown in the discussion that follows, some interesting insight is gained by attempting to explain the ringing property of ILPFs in the spatial domain.



FIGURE 4.41 (a) Original image of size 688×688 pixels. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.40(b). The power removed by these filters was 13.1, 7.2, 4.9, 2.4, and 0.6% of the total, respectively. We used mirror padding to avoid the black borders characteristic of zero padding, as illustrated in Fig. 4.31(c).

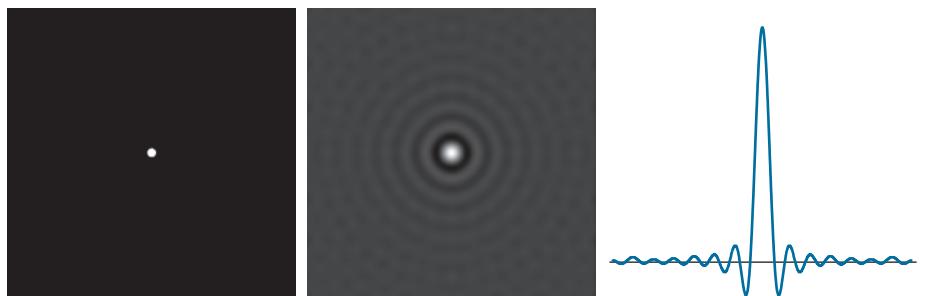
The blurring and ringing properties of ILPFs can be explained using the convolution theorem. Figure 4.42(a) shows an image of a frequency-domain ILPF transfer function of radius 15 and size 1000×1000 pixels. Figure 4.42(b) is the spatial representation, $h(x, y)$, of the ILPF, obtained by taking the IDFT of (a) (note the ringing). Figure 4.42(c) shows the intensity profile of a line passing through the center of (b). This profile resembles a sinc function.[†] Filtering in the spatial domain is done by convolving the function in Fig. 4.42(b) with an image. Imagine each pixel in an image as being a discrete impulse whose strength is proportional to the intensity of the image at that location. Convolving this sinc-like function with an impulse copies (i.e., shifts the origin of) the function to the location of the impulse. That is, convolution

[†]Although this profile resembles a sinc function, the transform of an ILPF is actually a Bessel function whose derivation is beyond the scope of this discussion. The important point to keep in mind is that the inverse proportionality between the “width” of the filter function in the frequency domain, and the “spread” of the width of the lobes in the spatial function, still holds.

a | b | c

FIGURE 4.42

- (a) Frequency domain ILPF transfer function.
 (b) Corresponding spatial domain kernel function.
 (c) Intensity profile of a horizontal line through the center of (b).



makes a copy of the function in Fig. 4.42(b) centered on each pixel location in the image. The center lobe of this spatial function is the principal cause of blurring, while the outer, smaller lobes are mainly responsible for ringing. Because the “spread” of the spatial function is inversely proportional to the radius of $H(u, v)$, the larger D_0 becomes (i.e., the more frequencies that are passed), the more the spatial function approaches an impulse which, in the limit, causes no blurring at all when convolved with the image. The converse happens as D_0 becomes smaller. This type of reciprocal behavior should be routine to you by now. In the next two sections, we show that it is possible to achieve blurring with little or no ringing, an important objective in lowpass filtering.

GAUSSIAN LOWPASS FILTERS

Gaussian lowpass filter (GLPF) transfer functions have the form

$$H(u, v) = e^{-D^2(u, v)/2\sigma^2} \quad (4-115)$$

where, as in Eq. (4-112), $D(u, v)$ is the distance from the center of the $P \times Q$ frequency rectangle to any point, (u, v) , contained by the rectangle. Unlike our earlier expressions for Gaussian functions, we do not use a multiplying constant here in order to be consistent with the filters discussed in this and later sections, whose highest value is 1. As before, σ is a measure of spread about the center. By letting $\sigma = D_0$, we can express the Gaussian transfer function in the same notation as other functions in this section:

$$H(u, v) = e^{-D^2(u, v)/2D_0^2} \quad (4-116)$$

where D_0 is the cutoff frequency. When $D(u, v) = D_0$, the GLPF transfer function is down to 0.607 of its maximum value of 1.0.

From Table 4.4, we know that the inverse Fourier transform of a frequency-domain Gaussian function is Gaussian also. This means that a spatial Gaussian filter kernel, obtained by computing the IDFT of Eq. (4-115) or (4-116), will have no ringing. As property 13 of Table 4.4 shows, the same inverse relationship explained earlier for ILPFs is true also of GLPFs. Narrow Gaussian transfer functions in the frequency domain imply broader kernel functions in the spatial domain, and vice

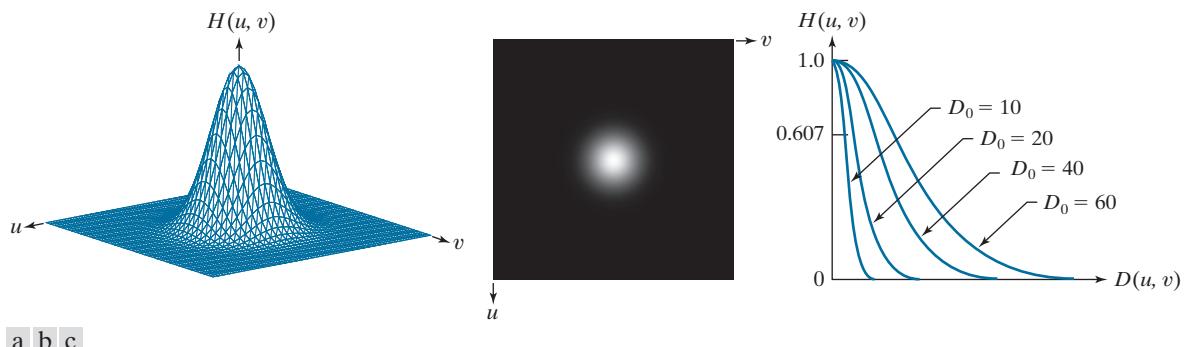


FIGURE 4.43 (a) Perspective plot of a GLPF transfer function. (b) Function displayed as an image. (c) Radial cross sections for various values of D_0 .

versa. Figure 4.43 shows a perspective plot, image display, and radial cross sections of a GLPF transfer function.

EXAMPLE 4.17: Image smoothing in the frequency domain using Gaussian lowpass filters.

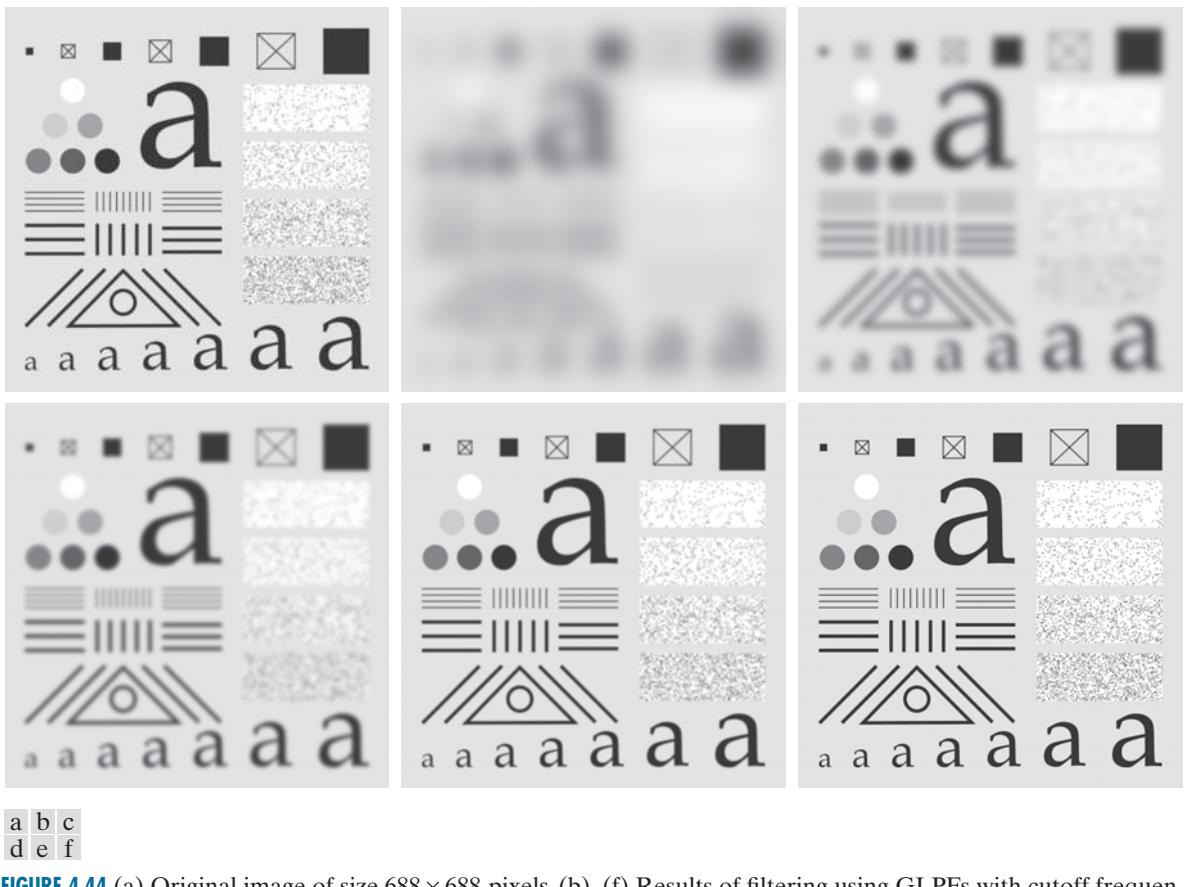
Figure 4.44 shows the results of applying the GLPF of Eq. (4-116) to Fig. 4.44(a), with D_0 equal to the five radii in Fig. 4.40(b). Compared to the results obtained with an ILPF (Fig. 4.41), we note a smooth transition in blurring as a function of increasing cutoff frequency. The GLPF achieved slightly less smoothing than the ILPF. The key difference is that we are assured of no ringing when using a GLPF. This is an important consideration in practice, especially in situations in which any type of artifact is unacceptable, as in medical imaging. In cases where more control of the transition between low and high frequencies about the cutoff frequency are needed, the Butterworth lowpass filter discussed next presents a more suitable choice. The price of this additional control over the filter profile is the possibility of ringing, as you will see shortly.

BUTTERWORTH LOWPASS FILTERS

The transfer function of a Butterworth lowpass filter (BLPF) of order n , with cutoff frequency at a distance D_0 from the center of the frequency rectangle, is defined as

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}} \quad (4-117)$$

where $D(u, v)$ is given by Eq. (4-112). Figure 4.45 shows a perspective plot, image display, and radial cross sections of the BLPF function. Comparing the cross section plots in Figs. 4.39, 4.43, and 4.45, we see that the BLPF function can be controlled to approach the characteristics of the ILPF using higher values of n , and the GLPF for lower values of n , while providing a smooth transition from low to high frequencies. Thus, we can use a BLPF to approach the sharpness of an ILPF function with considerably less ringing.



a	b	c
d	e	f

FIGURE 4.44 (a) Original image of size 688×688 pixels. (b)–(f) Results of filtering using GLPFs with cutoff frequencies at the radii shown in Fig. 4.40. Compare with Fig. 4.41. We used mirror padding to avoid the black borders characteristic of zero padding.

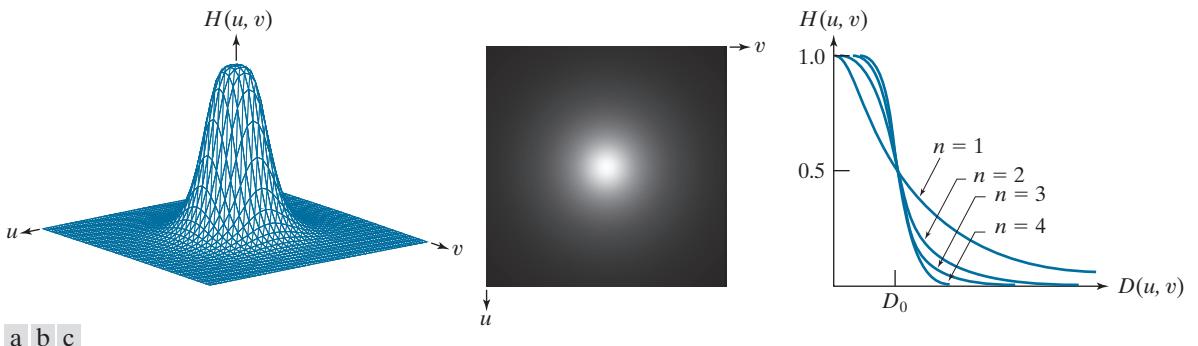


FIGURE 4.45 (a) Perspective plot of a Butterworth lowpass-filter transfer function. (b) Function displayed as an image. (c) Radial cross sections of BLPFs of orders 1 through 4.

EXAMPLE 4.18: Image smoothing using a Butterworth lowpass filter.

Figures 4.46(b)–(f) show the results of applying the BLPF of Eq. (4-117) to Fig. 4.46(a), with cutoff frequencies equal to the five radii in Fig. 4.40(b), and with $n = 2.25$. The results in terms of blurring are between the results obtained with using ILPFs and GLPFs. For example, compare Fig. 4.46(b), with Figs. 4.41(b) and 4.44(b). The degree of blurring with the BLPF was less than with the ILPF, but more than with the GLPF.

The kernels in Figs. 4.47(a) through (d) were obtained using the procedure outlined in the explanation of Fig. 4.42.

The spatial domain kernel obtainable from a BLPF of order 1 has no ringing. Generally, ringing is imperceptible in filters of order 2 or 3, but can become significant in filters of higher orders. Figure 4.47 shows a comparison between the spatial representation (i.e., spatial kernels) corresponding to BLPFs of various orders (using a cutoff frequency of 5 in all cases). Shown also is the intensity profile along



FIGURE 4.46 (a) Original image of size 688×688 pixels. (b)–(f) Results of filtering using BLPFs with cutoff frequencies at the radii shown in Fig. 4.40 and $n = 2.25$. Compare with Figs. 4.41 and 4.44. We used mirror padding to avoid the black borders characteristic of zero padding.

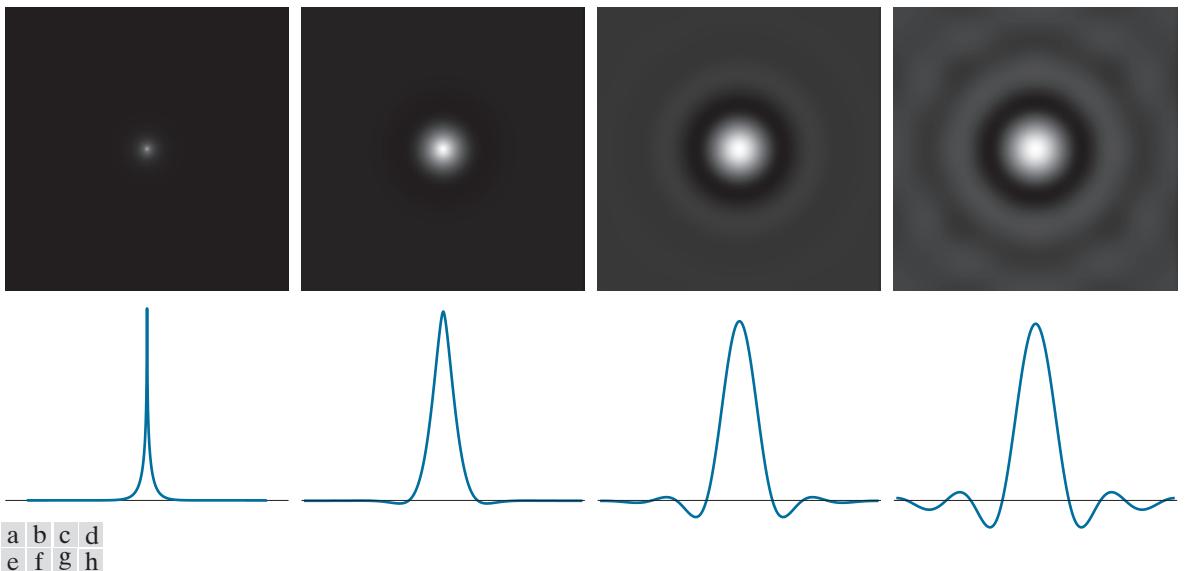


FIGURE 4.47 (a)–(d) Spatial representations (i.e., spatial kernels) corresponding to BLPF transfer functions of size 1000×1000 pixels, cut-off frequency of 5, and order 1, 2, 5, and 20, respectively. (e)–(h) Corresponding intensity profiles through the center of the filter functions.

a horizontal scan line through the center of each spatial kernel. The kernel corresponding to the BLPF of order 1 [see Fig. 4.47(a)] has neither ringing nor negative values. The kernel corresponding to a BLPF of order 2 does show mild ringing and small negative values, but they certainly are less pronounced than would be the case for an ILPF. As the remaining images show, ringing becomes significant for higher-order filters. A BLPF of order 20 has a spatial kernel that exhibits ringing characteristics similar to those of the ILPF (in the limit, both filters are identical). BLPFs of orders 2 to 3 are a good compromise between effective lowpass filtering and acceptable spatial-domain ringing. Table 4.5 summarizes the lowpass filter transfer functions discussed in this section.

ADDITIONAL EXAMPLES OF LOWPASS FILTERING

In the following discussion, we show several practical applications of lowpass filtering in the frequency domain. The first example is from the field of machine perception with application to character recognition; the second is from the printing and publishing industry; and the third is related to processing satellite and aerial images. Similar results can be obtained using the lowpass spatial filtering techniques discussed in Section 3.5. We use GLPFs in all examples for consistency, but similar results can be obtained using BLPFs. Keep in mind that images are padded to double size for filtering, as indicated by Eqs. (4-102) and (4-103), and filter transfer functions have to match padded-image size. The values of D_0 used in the following examples reflect this doubled filter size.

TABLE 4.5

Lowpass filter transfer functions. D_0 is the cutoff frequency, and n is the order of the Butterworth filter.

Ideal	Gaussian	Butterworth
$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$	$H(u,v) = e^{-D^2(u,v)/2D_0^2}$	$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$

Figure 4.48 shows a sample of text of low resolution. One encounters text like this, for example, in fax transmissions, duplicated material, and historical records. This particular sample is free of additional difficulties like smudges, creases, and torn sections. The magnified section in Fig. 4.48(a) shows that the characters in this document have distorted shapes due to lack of resolution, and many of the characters are broken. Although humans fill these gaps visually without difficulty, machine recognition systems have real difficulties reading broken characters. One approach for handling this problem is to bridge small gaps in the input image by blurring it. Figure 4.48(b) shows how well characters can be “repaired” by this simple process using a Gaussian lowpass filter with $D_0 = 120$. It is typical to follow the type of “repair” just described with additional processing, such as thresholding and thinning, to yield cleaner characters. We will discuss thinning in Chapter 9 and thresholding in Chapter 10.

Lowpass filtering is a staple in the printing and publishing industry, where it is used for numerous preprocessing functions, including unsharp masking, as discussed in Section 3.6. “Cosmetic” processing is another use of lowpass filtering prior to printing. Figure 4.49 shows an application of lowpass filtering for producing a smoother, softer-looking result from a sharp original. For human faces, the typical objective is to reduce the sharpness of fine skin lines and small blemishes. The magnified sections in Figs. 4.49(b) and (c) clearly show a significant reduction in fine skin lines around the subject’s eyes. In fact, the smoothed images look quite soft and pleasing.

Figure 4.50 shows two applications of lowpass filtering on the same image, but with totally different objectives. Figure 4.50(a) is an 808×754 segment of a very high

We will cover unsharp masking in the frequency domain in Section 4.9.

a b

FIGURE 4.48

(a) Sample text of low resolution (note the broken characters in the magnified view). (b) Result of filtering with a GLPF, showing that gaps in the broken characters were joined.

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



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





FIGURE 4.49 (a) Original 785×732 image. (b) Result of filtering using a GLPF with $D_0 = 150$. (c) Result of filtering using a GLPF with $D_0 = 130$. Note the reduction in fine skin lines in the magnified sections in (b) and (c).

resolution radiometer (VHRR) image showing part of the Gulf of Mexico (dark) and Florida (light) (note the horizontal sensor scan lines). The boundaries between bodies of water were caused by loop currents. This image is illustrative of remotely sensed images in which sensors have the tendency to produce pronounced scan lines along the direction in which the scene is being scanned. (See Example 4.24 for an

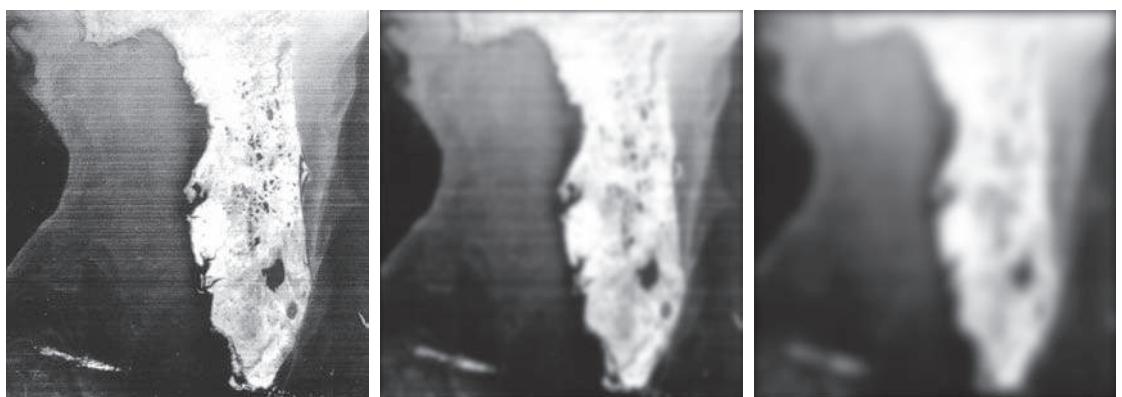


FIGURE 4.50 (a) 808×754 satellite image showing prominent horizontal scan lines. (b) Result of filtering using a GLPF with $D_0 = 50$. (c) Result of using a GLPF with $D_0 = 20$. (Original image courtesy of NOAA.)

illustration of imaging conditions that can lead for such degradations.) Lowpass filtering is a crude (but simple) way to reduce the effect of these lines, as Fig. 4.50(b) shows (we consider more effective approaches in Sections 4.10 and 5.4). This image was obtained using a GLFP with $D_0 = 50$. The reduction in the effect of the scan lines in the smoothed image can simplify the detection of macro features, such as the interface boundaries between ocean currents.

Figure 4.50(c) shows the result of significantly more aggressive Gaussian lowpass filtering with $D_0 = 20$. Here, the objective is to blur out as much detail as possible while leaving large features recognizable. For instance, this type of filtering could be part of a preprocessing stage for an image analysis system that searches for features in an image bank. An example of such features could be lakes of a given size, such as Lake Okeechobee in the lower eastern region of Florida, shown in Fig. 4.50(c) as a nearly round dark region surrounded by a lighter region. Lowpass filtering helps to simplify the analysis by averaging out features smaller than the ones of interest.

4.9 IMAGE SHARPENING USING HIGHPASS FILTERS

We showed in the previous section that an image can be smoothed by attenuating the high-frequency components of its Fourier transform. Because edges and other abrupt changes in intensities are associated with high-frequency components, image sharpening can be achieved in the frequency domain by highpass filtering, which attenuates low-frequencies components without disturbing high-frequencies in the Fourier transform. As in Section 4.8, we consider only zero-phase-shift filters that are radially symmetric. All filtering in this section is based on the procedure outlined in Section 4.7, so all images are assumed to be padded to size $P \times Q$ [see Eqs. (4-102) and (4-103)], and filter transfer functions, $H(u, v)$, are understood to be centered, discrete functions of size $P \times Q$.

In some applications of highpass filtering, it is advantageous to enhance the high-frequencies of the Fourier transform.

IDEAL, GAUSSIAN, AND BUTTERWORTH HIGHPASS FILTERS FROM LOWPASS FILTERS

As was the case with kernels in the spatial domain (see Section 3.7), subtracting a lowpass filter transfer function from 1 yields the corresponding highpass filter transfer function in the frequency domain:

$$H_{\text{HP}}(u, v) = 1 - H_{\text{LP}}(u, v) \quad (4-118)$$

where $H_{\text{LP}}(u, v)$ is the transfer function of a lowpass filter. Thus, it follows from Eq. (4-111) that an ideal highpass filter (IHPF) transfer function is given by

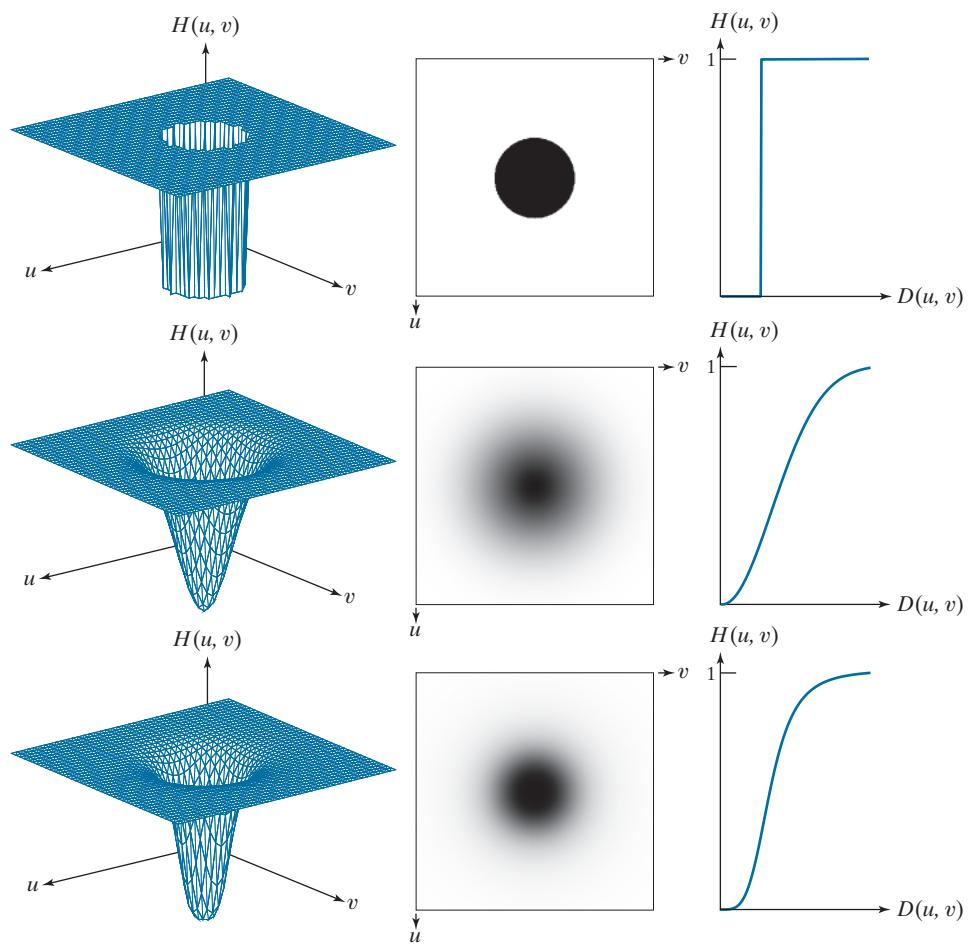
$$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases} \quad (4-119)$$

where, as before, $D(u, v)$ is the distance from the center of the $P \times Q$ frequency rectangle, as given in Eq. (4-112). Similarly, it follows from Eq. (4-116) that the transfer function of a Gaussian highpass filter (GHPF) transfer function is given by

a	b	c
d	e	f
g	h	i

FIGURE 4.51

Top row:
Perspective plot,
image, and, radial
cross section of
an IHPF transfer
function. Middle
and bottom
rows: The same
sequence for
GHPF and BHPF
transfer functions.
(The thin image
borders were
added for clarity.
They are not part
of the data.)



$$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2} \quad (4-120)$$

and, from Eq. (4-117), that the transfer function of a Butterworth highpass filter (BHPF) is

$$H(u, v) = \frac{1}{1 + [D_0/D(u, v)]^{2n}} \quad (4-121)$$

Figure 4.51 shows 3-D plots, image representations, and radial cross sections for the preceding transfer functions. As before, we see that the BHPF transfer function in the third row of the figure represents a transition between the sharpness of the IHPF and the broad smoothness of the GHPF transfer function.

It follows from Eq. (4-118) that the spatial kernel corresponding to a highpass filter transfer function in the frequency domain is given by

$$\begin{aligned}
 h_{\text{HP}}(x, y) &= \mathfrak{F}^{-1}[H_{\text{HP}}(u, v)] \\
 &= \mathfrak{F}^{-1}[1 - H_{\text{LP}}(u, v)] \\
 &= \delta(x, y) - h_{\text{LP}}(x, y)
 \end{aligned} \tag{4-122}$$

Recall that a unit impulse in the spatial domain is an array of 0's with a 1 in the center.

where we used the fact that the IDFT of 1 in the frequency domain is a unit impulse in the spatial domain (see Table 4.4). This equation is precisely the foundation for the discussion in Section 3.7, in which we showed how to construct a highpass kernel by subtracting a lowpass kernel from a unit impulse.

Figure 4.52 shows highpass spatial kernels constructed in just this manner, using Eq. (4-122) with ILPF, GLPF, and BLPF transfer functions (the values of M , N , and D_0 used in this figure are the same as those we used for Fig. 4.42, and the BLPF is of order 2). Figure 4.52(a) shows the resulting ideal highpass kernel obtained using Eq. (4-122), and Fig. 4.52(b) is a horizontal intensity profile through the center of the kernel. The center element of the profile is a unit impulse, visible as a bright dot in the center of Fig. 4.52(a). Note that this highpass kernel has the same ringing properties illustrated in Fig. 4.42(b) for its corresponding lowpass counterpart. As you will see shortly, ringing is just as objectionable as before, but this time in images sharpened with ideal highpass filters. The other images and profiles in Fig. 4.52 are for Gaussian and Butterworth kernels. We know from Fig. 4.51 that GHPF transfer functions in the frequency domain tend to have a broader “skirt” than Butterworth functions of comparable size and cutoff frequency. Thus, we would expect Butterworth spatial



FIGURE 4.52 (a)–(c): Ideal, Gaussian, and Butterworth highpass spatial kernels obtained from IHPF, GHPF, and BHPF frequency-domain transfer functions. (The thin image borders are not part of the data.) (d)–(f): Horizontal intensity profiles through the centers of the kernels.

TABLE 4.6

Highpass filter transfer functions. D_0 is the cutoff frequency and n is the order of the Butterworth transfer function.

Ideal	Gaussian	Butterworth
$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$	$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$	$H(u,v) = \frac{1}{1 + [D_0/D(u,v)]^{2n}}$

kernels to be “broader” than comparable Gaussian kernels, a fact that is confirmed by the images and their profiles in Figs. 4.52. Table 4.6 summarizes the three highpass filter transfer functions discussed in the preceding paragraphs.

EXAMPLE 4.19: Highpass filtering of the character test pattern.

The first row of Fig. 4.53 shows the result of filtering the test pattern in Fig. 4.37(a) using IHPF, GHPF, and BHPF transfer functions with $D_0 = 60$ [see Fig. 4.37(b)] and $n = 2$ for the Butterworth filter. We know from Chapter 3 that highpass filtering produces images with negative values. The images in Fig. 4.53 are not scaled, so the negative values are clipped by the display at 0 (black). The key objective of highpass filtering is to sharpen. Also, because the highpass filters used here set the DC term to zero, the images have essentially no tonality, as explained earlier in connection with Fig. 4.30.

Our main objective in this example is to compare the behavior of the three highpass filters. As Fig. 4.53(a) shows, the ideal highpass filter produced results with severe distortions caused by ringing. For example, the blotches inside the strokes of the large letter “a” are ringing artifacts. By comparison, neither Figs. 4.53(b) or (c) have such distortions. With reference to Fig. 4.37(b), the filters removed or attenuated approximately 95% of the image energy. As you know, removing the lower frequencies of an image reduces its gray-level content significantly, leaving mostly edges and other sharp transitions, as is evident in Fig. 4.53. The details you see in the first row of the figure are contained in only the upper 5% of the image energy.

The second row, obtained with $D_0 = 160$, is more interesting. The remaining energy of those images is about 2.5%, or half, the energy of the images in the first row. However, the difference in fine detail is striking. See, for example, how much cleaner the boundary of the large “a” is now, especially in the Gaussian and Butterworth results. The same is true for all other details, down to the smallest objects. This is the type of result that is considered acceptable when detection of edges and boundaries is important.

Figure 4.54 shows the images in the second row of Fig. 4.53, scaled using Eqs. (2-31) and (2-32) to display the full intensity range of both positive and negative intensities. The ringing in Fig. 4.54(a) shows the inadequacy of ideal highpass filters. In contrast, notice the smoothness of the background on the other two images, and the crispness of their edges.

EXAMPLE 4.20: Using highpass filtering and thresholding for image enhancement.

Figure 4.55(a) is a 962×1026 image of a thumbprint in which smudges (a typical problem) are evident. A key step in automated fingerprint recognition is enhancement of print ridges and the reduction of smudges. In this example, we use highpass filtering to enhance the ridges and reduce the effects of

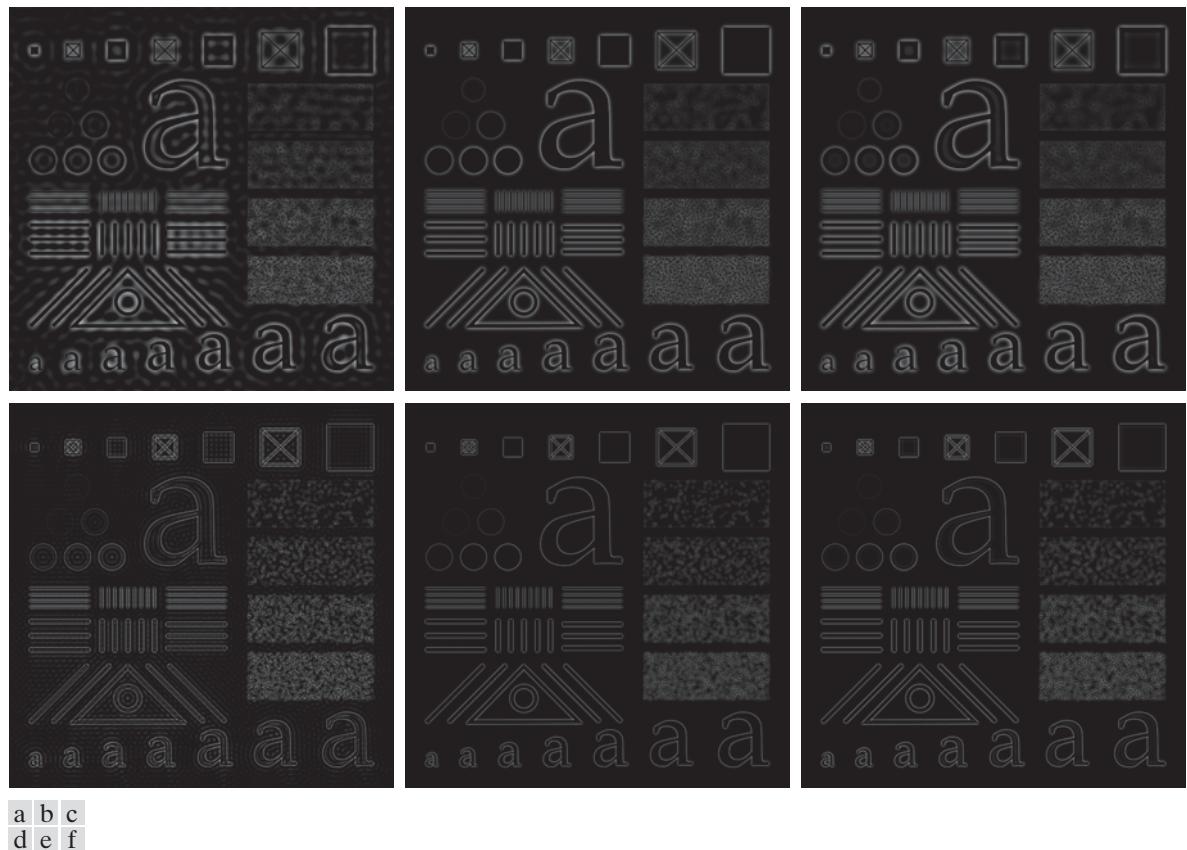


FIGURE 4.53 Top row: The image from Fig. 4.40(a) filtered with IHPF, GHPF, and BHPF transfer functions using $D_0 = 60$ in all cases ($n = 2$ for the BHPF). Second row: Same sequence, but using $D_0 = 160$.

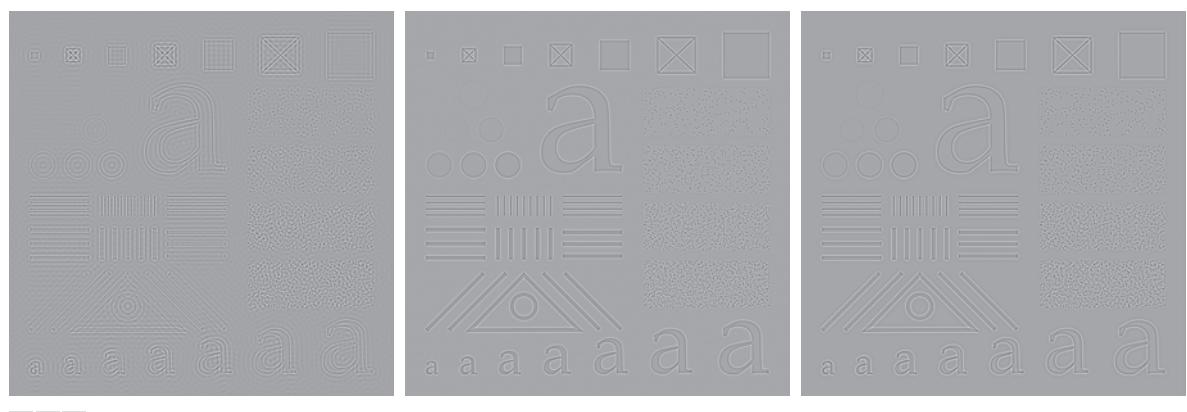


FIGURE 4.54 The images from the second row of Fig. 4.53 scaled using Eqs. (2-31) and (2-32) to show both positive and negative values.



a | b | c

FIGURE 4.55 (a) Smudged thumbprint. (b) Result of highpass filtering (a). (c) Result of thresholding (b). (Original image courtesy of the U.S. National Institute of Standards and Technology.)

smudging. Enhancement of the ridges is accomplished by the fact that their boundaries are characterized by high frequencies, which are unchanged by a highpass filter. On the other hand, the filter reduces low frequency components, which correspond to slowly varying intensities in the image, such as the background and smudges. Thus, enhancement is achieved by reducing the effect of all features except those with high frequencies, which are the features of interest in this case.

Figure 4.55(b) is the result of using a Butterworth highpass filter of order 4 with a cutoff frequency of 50. A fourth-order filter provides a sharp (but smooth) transition from low to high frequencies, with filtering characteristics between an ideal and a Gaussian filter. The cutoff frequency chosen is about 5% of the long dimension of the image. The idea is for D_0 to be close to the origin so that low frequencies are attenuated but not completely eliminated, except for the DC term which is set to 0, so that tonality differences between the ridges and background are not lost completely. Choosing a value for D_0 between 5% and 10% of the long dimension of the image is a good starting point. Choosing a large value of D_0 would highlight fine detail to such an extent that the definition of the ridges would be affected. As expected, the highpass filtered image has negative values, which are shown as black by the display.

A simple approach for highlighting sharp features in a highpass-filtered image is to threshold it by setting to black (0) all negative values and to white (1) the remaining values. Figure 4.55(c) shows the result of this operation. Note how the ridges are clear, and how the effect of the smudges has been reduced considerably. In fact, ridges that are barely visible in the top, right section of the image in Fig. 4.55(a) are nicely enhanced in Fig. 4.55(c). An automated algorithm would find it much easier to follow the ridges on this image than it would on the original.

THE LAPLACIAN IN THE FREQUENCY DOMAIN

In Section 3.6, we used the Laplacian for image sharpening in the spatial domain. In this section, we revisit the Laplacian and show that it yields equivalent results using frequency domain techniques. It can be shown (see Problem 4.52) that the Laplacian can be implemented in the frequency domain using the filter transfer function

$$H(u, v) = -4\pi^2(u^2 + v^2) \quad (4-123)$$

or, with respect to the center of the frequency rectangle, using the transfer function

$$\begin{aligned} H(u, v) &= -4\pi^2 \left[(u - P/2)^2 + (v - Q/2)^2 \right] \\ &= -4\pi^2 D^2(u, v) \end{aligned} \quad (4-124)$$

where $D(u, v)$ is the distance function defined in Eq. (4-112). Using this transfer function, the Laplacian of an image, $f(x, y)$, is obtained in the familiar manner:

$$\nabla^2 f(x, y) = \mathcal{F}^{-1}[H(u, v)F(u, v)] \quad (4-125)$$

where $F(u, v)$ is the DFT of $f(x, y)$. As in Eq. (3-54), enhancement is implemented using the equation

$$g(x, y) = f(x, y) + c\nabla^2 f(x, y) \quad (4-126)$$

Here, $c = -1$ because $H(u, v)$ is negative. In Chapter 3, $f(x, y)$ and $\nabla^2 f(x, y)$ had comparable values. However, computing $\nabla^2 f(x, y)$ with Eq. (4-125) introduces DFT scaling factors that can be several orders of magnitude larger than the maximum value of f . Thus, the differences between f and its Laplacian must be brought into comparable ranges. The easiest way to handle this problem is to normalize the values of $f(x, y)$ to the range $[0, 1]$ (before computing its DFT) and divide $\nabla^2 f(x, y)$ by its maximum value, which will bring it to the approximate range $[-1, 1]$. (Remember, the Laplacian has negative values.) Equation (4-126) can then be used.

We can write Eq. (4-126) directly in the frequency domain as

$$\begin{aligned} g(x, y) &= \mathcal{F}^{-1}\{F(u, v) - H(u, v)F(u, v)\} \\ &= \mathcal{F}^{-1}\{[1 - H(u, v)]F(u, v)\} \\ &= \mathcal{F}^{-1}\{[1 + 4\pi^2 D^2(u, v)]F(u, v)\} \end{aligned} \quad (4-127)$$

Although this result is elegant, it has the same scaling issues just mentioned, compounded by the fact that the normalizing factor is not as easily computed. For this reason, Eq. (4-126) is the preferred implementation in the frequency domain, with $\nabla^2 f(x, y)$ computed using Eq. (4-125) and scaled using the approach mentioned in the previous paragraph.

EXAMPLE 4.21: Image sharpening in the frequency domain using the Laplacian.

Figure 4.56(a) is the same as Fig. 3.46(a), and Fig. 4.56(b) shows the result of using Eq. (4-126), in which the Laplacian was computed in the frequency domain using Eq. (4-125). Scaling was done as described in connection with Eq. (4-126). We see by comparing Figs. 4.56(b) and 3.46(d) that the frequency-domain result is superior. The image in Fig. 4.56(b) is much sharper, and shows details that are barely visible in 3.46(d), which was obtained using the Laplacian kernel in Fig. 3.45(b), with a -8 in the center. The significant improvement achieved in the frequency domain is not unexpected. The spatial Laplacian kernel

a b

FIGURE 4.56

(a) Original, blurry image.
 (b) Image enhanced using the Laplacian in the frequency domain.
 Compare with Fig. 3.46(d).
 (Original image courtesy of NASA.)



encompasses a very small neighborhood, while the formulation in Eqs. (4-125) and (4-126) encompasses the entire image.

UNSHARP MASKING, HIGH-BOOST FILTERING, AND HIGH-FREQUENCY-EMPHASIS FILTERING

In this section, we discuss frequency domain formulations of the unsharp masking and high-boost filtering image sharpening techniques introduced in Section 3.6. Using frequency domain methods, the mask defined in Eq. (3-55) is given by

$$g_{\text{mask}}(x, y) = f(x, y) - f_{\text{LP}}(x, y) \quad (4-128)$$

with

$$f_{\text{LP}}(x, y) = \mathfrak{F}^{-1}[H_{\text{LP}}(u, v)F(u, v)] \quad (4-129)$$

where $H_{\text{LP}}(u, v)$ is a lowpass filter transfer function, and $F(u, v)$ is the DFT of $f(x, y)$. Here, $f_{\text{LP}}(x, y)$ is a smoothed image analogous to $\bar{f}(x, y)$ in Eq. (3-55). Then, as in Eq. (3-56),

$$g(x, y) = f(x, y) + kg_{\text{mask}}(x, y) \quad (4-130)$$

This expression defines *unsharp masking* when $k = 1$ and *high-boost filtering* when $k > 1$. Using the preceding results, we can express Eq. (4-130) entirely in terms of frequency domain computations involving a lowpass filter:

$$g(x, y) = \mathfrak{F}^{-1}\left\{\left(1 + k[1 - H_{\text{LP}}(u, v)]\right)F(u, v)\right\} \quad (4-131)$$

We can express this result in terms of a highpass filter using Eq. (4-118):

$$g(x, y) = \mathfrak{F}^{-1} \{ [1 + kH_{HP}(u, v)] F(u, v) \} \quad (4-132)$$

The expression contained within the square brackets is called a *high-frequency-emphasis filter transfer function*. As noted earlier, highpass filters set the dc term to zero, thus reducing the average intensity in the filtered image to 0. The high-frequency-emphasis filter does not have this problem because of the 1 that is added to the highpass filter transfer function. Constant k gives control over the proportion of high frequencies that influences the final result. A slightly more general formulation of high-frequency-emphasis filtering is the expression

$$g(x, y) = \mathfrak{F}^{-1} \{ [k_1 + k_2 H_{HP}(u, v)] F(u, v) \} \quad (4-133)$$

where $k_1 \geq 0$ offsets the value the transfer function so as not to zero-out the dc term [see Fig. 4.30(c)], and $k_2 > 0$ controls the contribution of high frequencies.

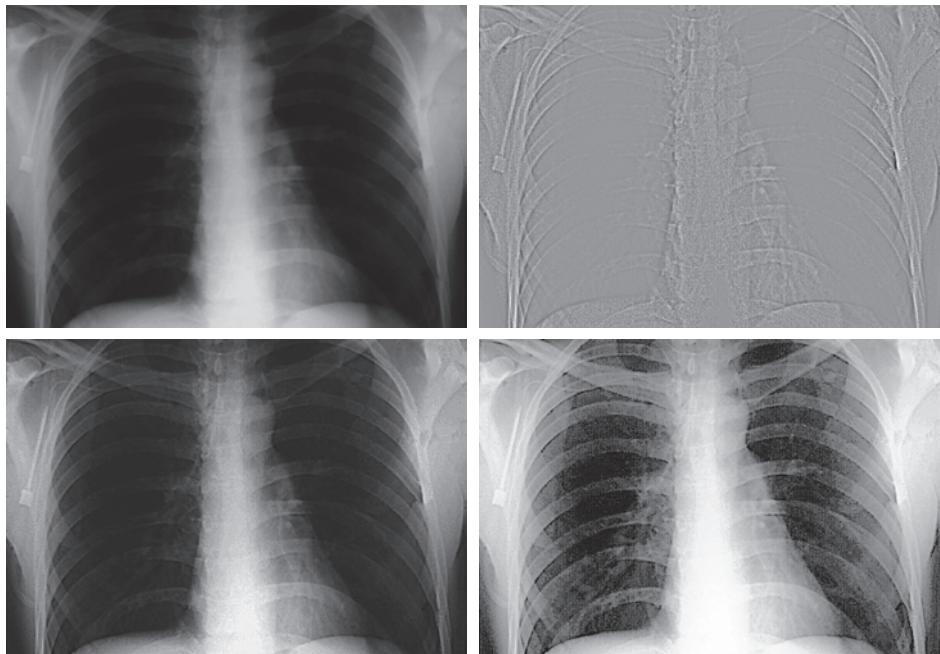
EXAMPLE 4.22: Image enhancement using high-frequency-emphasis filtering.

Figure 4.57(a) shows a 503×720 -pixel chest X-ray image with a narrow range of intensity levels. The objective of this example is to enhance the image using high-frequency-emphasis filtering. X-rays cannot be focused in the same manner that optical lenses can, and the resulting images generally tend to be slightly blurred. Because the intensities in this particular image are biased toward the dark end of the

a	b
c	d

FIGURE 4.57

- (a) A chest X-ray.
- (b) Result of filtering with a GHPF function.
- (c) Result of high-frequency-emphasis filtering using the same GHPF.
- (d) Result of performing histogram equalization on (c). (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)



gray scale, we also take this opportunity to give an example of how spatial domain processing can be used to complement frequency-domain filtering.

Image artifacts, such as ringing, are unacceptable in medical image processing, so we use a Gaussian highpass filter transfer function. Because the spatial representation of a GHPF function is Gaussian also, we know that ringing will not be an issue. The value chosen for D_0 should provide enough filtering to sharpen boundaries while at the same time not over-sharpening minute details (such as noise). We used $D_0 = 70$, approximately 10% of the long image dimension, but other similar values would work also. Figure 4.57(b) is the result of highpass filtering the original image (scaled as the images in Fig. 4.54). As expected, the image is rather featureless, but the important boundaries (e.g., the edges of the ribs) are clearly delineated. Figure 4.57(c) shows the advantage of high-frequency-emphasis filtering, where we used Eq. (4-133) with $k_1 = 0.5$ and $k_2 = 0.75$. Although the image is still dark, the gray-level tonality has been restored, with the added advantage of sharper features.

As we discussed in Section 3.3, an image characterized by intensity levels in a narrow range of the gray scale is an ideal candidate for histogram equalization. As Fig. 4.57(d) shows, this was indeed an appropriate method to further enhance the image. Note the clarity of the bone structure and other details that simply are not visible in any of the other three images. The final enhanced image is a little noisy, but this is typical of X-ray images when their gray scale is expanded. The result obtained using a combination of high-frequency-emphasis and histogram equalization is superior to the result that would be obtained by using either method alone.

HOMOMORPHIC FILTERING

The illumination-reflectance model introduced in Section 2.3 can be used to develop a frequency domain procedure for improving the appearance of an image by simultaneous intensity range compression and contrast enhancement. From the discussion in that section, an image $f(x, y)$ can be expressed as the product of its illumination, $i(x, y)$, and reflectance, $r(x, y)$, components:

$$f(x, y) = i(x, y)r(x, y) \quad (4-134)$$

This equation cannot be used directly to operate on the frequency components of illumination and reflectance because the Fourier transform of a product is not the product of the transforms:

$$\mathfrak{F}[f(x, y)] \neq \mathfrak{F}[i(x, y)]\mathfrak{F}[r(x, y)] \quad (4-135)$$

However, suppose that we define

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y) \end{aligned} \quad (4-136)$$

If $f(x, y)$ has any zero values, a 1 must be added to the image to avoid having to deal with $\ln(0)$. The 1 is then subtracted from the final result.

Then,

$$\begin{aligned} \mathfrak{F}[z(x, y)] &= \mathfrak{F}[\ln f(x, y)] \\ &= \mathfrak{F}[\ln i(x, y)] + \mathfrak{F}[\ln r(x, y)] \end{aligned} \quad (4-137)$$

or

$$Z(u, v) = F_i(u, v) + F_r(u, v) \quad (4-138)$$

where $F_i(u, v)$ and $F_r(u, v)$ are the Fourier transforms of $\ln i(x, y)$ and $\ln r(x, y)$, respectively.

We can filter $Z(u, v)$ using a filter transfer function $H(u, v)$ so that

$$\begin{aligned} S(u, v) &= H(u, v)Z(u, v) \\ &= H(u, v)F_i(u, v) + H(u, v)F_r(u, v) \end{aligned} \quad (4-139)$$

The filtered image in the spatial domain is then

$$\begin{aligned} s(x, y) &= \mathfrak{F}^{-1}[S(u, v)] \\ &= \mathfrak{F}^{-1}[H(u, v)F_i(u, v)] + \mathfrak{F}^{-1}[H(u, v)F_r(u, v)] \end{aligned} \quad (4-140)$$

By defining

$$i'(x, y) = \mathfrak{F}^{-1}[H(u, v)F_i(u, v)] \quad (4-141)$$

and

$$r'(x, y) = \mathfrak{F}^{-1}[H(u, v)F_r(u, v)] \quad (4-142)$$

we can express Eq. (4-140) in the form

$$s(x, y) = i'(x, y) + r'(x, y) \quad (4-143)$$

Finally, because $z(x, y)$ was formed by taking the natural logarithm of the input image, we reverse the process by taking the exponential of the filtered result to form the output image:

$$\begin{aligned} g(x, y) &= e^{s(x, y)} \\ &= e^{i'(x, y)}e^{r'(x, y)} \\ &= i_0(x, y)r_0(x, y) \end{aligned} \quad (4-144)$$

where

$$i_0(x, y) = e^{i'(x, y)} \quad (4-145)$$

and

$$r_0(x, y) = e^{r'(x, y)} \quad (4-146)$$

are the illumination and reflectance components of the output (processed) image.

Figure 4.58 is a summary of the filtering approach just derived. This method is based on a special case of a class of systems known as *homomorphic systems*. In this particular application, the key to the approach is the separation of the illumination

FIGURE 4.58

Summary of steps in homomorphic filtering.



and reflectance components achieved in the form shown in Eq. (4-138). The *homomorphic filter transfer function*, $H(u,v)$, then can operate on these components separately, as indicated by Eq. (4-139).

The illumination component of an image generally is characterized by slow spatial variations, while the reflectance component tends to vary abruptly, particularly at the junctions of dissimilar objects. These characteristics lead to associating the low frequencies of the Fourier transform of the logarithm of an image with illumination, and the high frequencies with reflectance. Although these associations are rough approximations, they can be used to advantage in image filtering, as illustrated in Example 4.23.

A good deal of control can be gained over the illumination and reflectance components with a homomorphic filter. This control requires specification of a filter transfer function $H(u,v)$ that affects the low- and high-frequency components of the Fourier transform in different, controllable ways. Figure 4.59 shows a cross section of such a function. If the parameters γ_L and γ_H are chosen so that $\gamma_L < 1$ and $\gamma_H \geq 1$, the filter function in Fig. 4.59 will attenuate the contribution made by the low frequencies (illumination) and amplify the contribution made by high frequencies (reflectance). The net result is simultaneous dynamic range compression and contrast enhancement.

The shape of the function in Fig. 4.59 can be approximated using a highpass filter transfer function. For example, using a slightly modified form of the GHPF function yields the homomorphic function

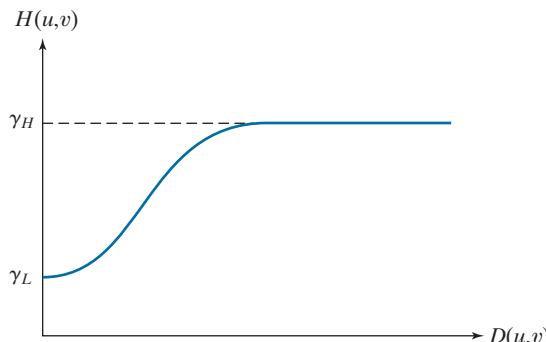
$$H(u,v) = (\gamma_H - \gamma_L) \left[1 - e^{-cD^2(u,v)/D_0^2} \right] + \gamma_L \quad (4-147)$$

where $D(u,v)$ is defined in Eq. (4-112) and constant c controls the sharpness of the slope of the function as it transitions between γ_L and γ_H . This filter transfer function is similar to the high-frequency-emphasis function discussed in the previous section.

A BHPF function would work well too, with the added advantage of more control over the sharpness of the transition between γ_L and γ_H . The disadvantage is the possibility of ringing for high values of n .

FIGURE 4.59

Radial cross section of a homomorphic filter transfer function..



a b

FIGURE 4.60

(a) Full body PET scan. (b) Image enhanced using homomorphic filtering. (Original image courtesy of Dr. Michael E. Casey, CTI Pet Systems.)

**EXAMPLE 4.23: Homomorphic filtering.**

Figure 4.60(a) shows a full body PET (Positron Emission Tomography) scan of size 1162×746 pixels. The image is slightly blurred and many of its low-intensity features are obscured by the high intensity of the “hot spots” dominating the dynamic range of the display. (These hot spots were caused by a tumor in the brain and one in the lungs.) Figure 4.60(b) was obtained by homomorphic filtering Fig. 4.60(a) using the filter transfer function in Eq. (4-147) with $\gamma_L = 0.4$, $\gamma_H = 3.0$, $c = 5$, and $D_0 = 20$. A radial cross section of this function looks just like Fig. 4.59, but with a much sharper slope, and the transition between low and high frequencies much closer to the origin.

Note in Fig. 4.60(b) how much sharper the hot spots, the brain, and the skeleton are in the processed image, and how much more detail is visible in this image, including, for example, some of the organs, the shoulders, and the pelvis region. By reducing the effects of the dominant illumination components (the hot spots), it became possible for the dynamic range of the display to allow lower intensities to become more visible. Similarly, because the high frequencies are enhanced by homomorphic filtering, the reflectance components of the image (edge information) were sharpened considerably. The enhanced image in Fig. 4.60(b) is a significant improvement over the original.

4.10 SELECTIVE FILTERING

The filters discussed in the previous two sections operate over the entire frequency rectangle. There are applications in which it is of interest to process specific bands of frequencies or small regions of the frequency rectangle. Filters in the first category

are called *band filters*. If frequencies in the band are filtered out, the band filter is called a *bandreject* filter; similarly, if the frequencies are passed, the filter is called a *bandpass* filter. Filters in the second category are called *notch filters*. These filters are further qualified as being *notch reject* or *notch pass* filters, depending on whether frequencies in the notch areas are rejected or passed.

BANDREJECT AND BANDPASS FILTERS

As you learned in Section 3.7, bandpass and bandreject filter transfer functions in the frequency domain can be constructed by combining lowpass and highpass filter transfer functions, with the latter also being derivable from lowpass functions (see Fig. 3.52). In other words, lowpass filter transfer functions are the basis for forming highpass, bandreject, and bandpass filter functions. Furthermore, a bandpass filter transfer function is obtained from a bandreject function in the same manner that we obtained a highpass from a lowpass transfer function:

$$H_{\text{BP}}(u, v) = 1 - H_{\text{BR}}(u, v) \quad (4-148)$$

Figure 4.61(a) shows how to construct an ideal bandreject filter (IBRF) transfer function. It consists of an ILPF and an IHPF function with different cutoff frequencies. When dealing with bandpass functions, the parameters of interest are the width, W , and the center, C_0 , of the band. An equation for the IBRF function is easily obtained by inspection from Fig. 4.61(a), as the leftmost entry in Table 4.7 shows. The key requirements of a bandpass transfer function are: (1) the values of the function must be in the range [0, 1]; (2) the value of the function must be zero at a distance C_0 from the origin (center) of the function; and (3) we must be able to specify a value for W . Clearly, the IBRF function just developed satisfies these requirements.

Adding lowpass and highpass transfer functions to form Gaussian and Butterworth bandreject functions presents some difficulties. For example, Fig. 4.61(b) shows a bandpass function formed as the sum of lowpass and highpass Gaussian functions with different cutoff points. Two problems are immediately obvious: we have no direct control over W , and the value of $H(u, v)$ is not 0 at C_0 . We could

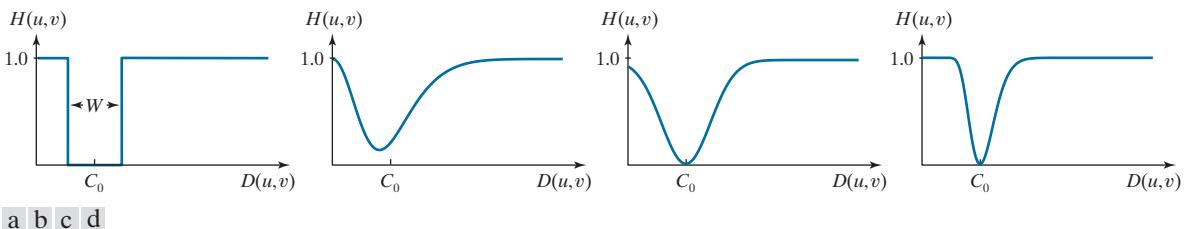


FIGURE 4.61 Radial cross sections. (a) Ideal bandreject filter transfer function. (b) Bandreject transfer function formed by the sum of Gaussian lowpass and highpass filter functions. (The minimum is not 0 and does not align with C_0 .) (c) Radial plot of Eq. (4-149). (The minimum is 0 and is properly aligned with C_0 , but the value at the origin is not 1.) (d) Radial plot of Eq. (4-150); this Gaussian-shape plot meets all the requirements of a bandreject filter transfer function.

TABLE 4.7

Bandreject filter transfer functions. C_0 is the center of the band, W is the width of the band, and $D(u,v)$ is the distance from the center of the transfer function to a point (u,v) in the frequency rectangle.

Ideal (IBRF)	Gaussian (GBRF)	Butterworth (BBRF)
$H(u,v) = \begin{cases} 0 & \text{if } C_0 - \frac{W}{2} \leq D(u,v) \leq C_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u,v) = 1 - e^{-\left[\frac{D^2(u,v) - C_0^2}{D(u,v)W}\right]^2}$	$H(u,v) = \frac{1}{1 + \left[\frac{D(u,v)W}{D^2(u,v) - C_0^2}\right]^{2n}}$

offset the function and scale it so that values fall in the range $[0, 1]$, but finding an analytical solution for the point where the lowpass and highpass Gaussian functions intersect is impossible, and this intersection would be required to solve for the cutoff points in terms of C_0 . The only alternatives are trial-and-error or numerical methods.

Fortunately, instead of adding lowpass and highpass transfer function, an alternative is to modify the expressions for the Gaussian and Butterworth highpass transfer functions so that they will satisfy the three requirements stated earlier. We illustrate the procedure for a Gaussian function. In this case, we begin by changing the point at which $H(u,v) = 0$ from $D(u,v) = 0$ to $D(u,v) = C_0$ in Eq. (4-120):

$$H(u,v) = 1 - e^{-\left[\frac{(D(u,v) - C_0)^2}{W^2}\right]} \quad (4-149)$$

A plot of this function [Fig. 4.61(c)] shows that, below C_0 , the function behaves as a lowpass Gaussian function, at C_0 the function will always be 0, and for values higher than C_0 the function behaves as a highpass Gaussian function. Parameter W is proportional to the standard deviation and thus controls the “width” of the band. The only problem remaining is that the function is not always 1 at the origin. A simple modification of Eq. (4-149) removes this shortcoming:

$$H(u,v) = 1 - e^{-\left[\frac{D^2(u,v) - C_0^2}{D(u,v)W}\right]^2} \quad (4-150)$$

The overall ratio in this equation is squared so that, as the distance increases, Eqs. (4-149) and (4-150) behave approximately the same.

Now, the exponent is infinite when $D(u,v) = 0$, which makes the exponential term go to zero and $H(u,v) = 1$ at the origin, as desired. In this modification of Eq. (4-149), the basic Gaussian shape is preserved and the three requirements stated earlier are satisfied. Figure 4.61(d) shows a plot of Eq. (4-150). A similar analysis leads to the form of a Butterworth bandreject filter transfer function shown in Table 4.7.

Figure 4.62 shows perspective plots of the filter transfer functions just discussed. At first glance the Gaussian and Butterworth functions appear to be about the same, but, as before, the behavior of the Butterworth function is between the ideal and Gaussian functions. As Fig. 4.63 shows, this is easier to see by viewing the three filter

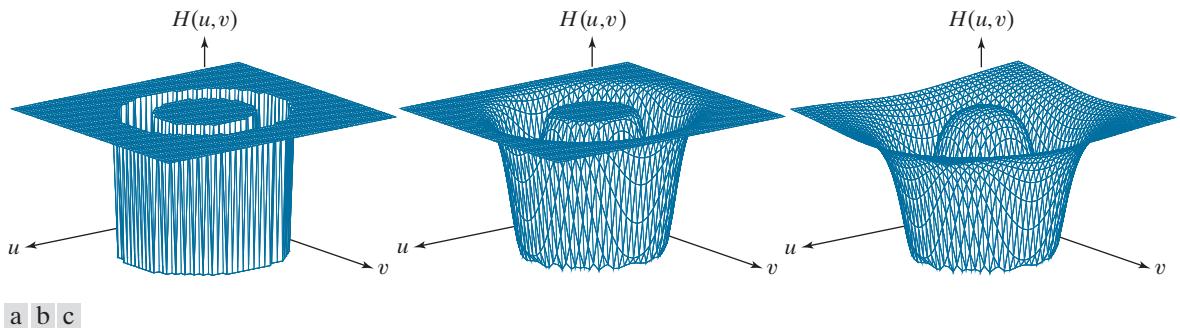


FIGURE 4.62 Perspective plots of (a) ideal, (b) modified Gaussian, and (c) modified Butterworth (of order 1) bandreject filter transfer functions from Table 4.7. All transfer functions are of size 512×512 elements, with $C_0 = 128$ and $W = 60$.

functions as images. Increasing the order of the Butterworth function would bring it closer to the ideal bandreject transfer function.

NOTCH FILTERS

Notch filters are the most useful of the selective filters. A notch filter rejects (or passes) frequencies in a predefined neighborhood of the frequency rectangle. Zero-phase-shift filters must be symmetric about the origin (center of the frequency rectangle), so a notch filter transfer function with center at (u_0, v_0) must have a corresponding notch at location $(-u_0, -v_0)$. *Notch reject* filter transfer functions are constructed as products of highpass filter transfer functions whose centers have been translated to the centers of the notches. The general form is:

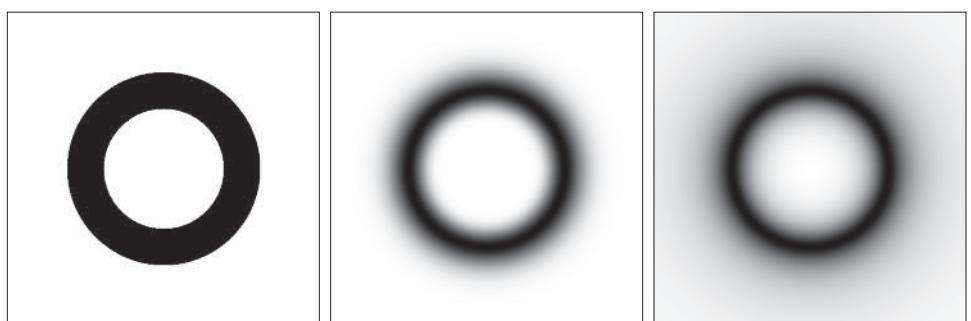
$$H_{\text{NR}}(u, v) = \prod_{k=1}^Q H_k(u, v) H_{-k}(u, v) \quad (4-151)$$

where $H_k(u, v)$ and $H_{-k}(u, v)$ are highpass filter transfer functions whose centers are at (u_k, v_k) and $(-u_k, -v_k)$, respectively. These centers are specified with respect to the center of the frequency rectangle, $(M/2, N/2)$, where, as usual, M and N are the



FIGURE 4.63

(a) The ideal,
(b) Gaussian, and
(c) Butterworth
bandpass transfer
functions from
Fig. 4.62, shown
as images. (The
thin border lines
are not part of the
image data.)



number of rows and columns in the input image. Thus, the distance computations for each filter transfer function are given by

$$D_k(u, v) = \left[(u - M/2 - u_k)^2 + (v - N/2 - v_k)^2 \right]^{1/2} \quad (4-152)$$

and

$$D_{-k}(u, v) = \left[(u - M/2 + u_k)^2 + (v - N/2 + v_k)^2 \right]^{1/2} \quad (4-153)$$

For example, the following is a Butterworth notch reject filter transfer function of order n , containing three notch pairs:

$$H_{\text{NR}}(u, v) = \prod_{k=1}^3 \left[\frac{1}{1 + [D_{0k}/D_k(u, v)]^n} \right] \left[\frac{1}{1 + [D_{0k}/D_{-k}(u, v)]^n} \right] \quad (4-154)$$

where $D_k(u, v)$ and $D_{-k}(u, v)$ are given by Eqs. (4-152) and (4-153). The constant D_{0k} is the same for each pair of notches, but it can be different for different pairs. Other notch reject filter functions are constructed in the same manner, depending on the highpass filter function chosen. As with the filters discussed earlier, a notch pass filter transfer function is obtained from a notch reject function using the expression

$$H_{\text{NP}}(u, v) = 1 - H_{\text{NR}}(u, v) \quad (4-155)$$

As the next two examples show, one of the principal applications of notch filtering is for selectively modifying local regions of the DFT. Often, this type of processing is done interactively, working directly with DFTs obtained without padding. The advantages of working interactively with actual DFTs (as opposed to having to “translate” from padded to actual frequency values) generally outweigh any wrap-around errors that may result from not using padding in the filtering process. If necessary, after an acceptable solution is obtained, a final result using padding can be generated by adjusting all filter parameters to compensate for the padded DFT size. The following two examples were done without padding. To get an idea of how DFT values change as a function of padding, see Problem 4.42.

EXAMPLE 4.24: Using notch filtering to remove moiré patterns from digitized printed media images.

Figure 4.64(a) is the scanned newspaper image used in Fig. 4.21, showing a prominent moiré pattern, and Fig. 4.64(b) is its spectrum. The Fourier transform of a pure sine, which is a periodic function, is a pair of conjugate symmetric impulses (see Table 4.4). The symmetric “impulse-like” bursts in Fig. 4.64(b) are a result of the near periodicity of the moiré pattern. We can attenuate these bursts by using notch filtering.

Figure 4.64(c) shows the result of multiplying the DFT of Fig. 4.64(a) by a Butterworth notch reject transfer function with $D_0 = 9$ and $n = 4$ for all notch pairs (the centers of the notches are coincide with the centers of the black circular regions in the figure). The value of the radius was selected (by visual inspection of the spectrum) to encompass the energy bursts completely, and the value of n was selected to produce notches with sharp transitions. The locations of the center of the notches were determined

a b
c d

FIGURE 4.64

- (a) Sampled newspaper image showing a moiré pattern.
(b) Spectrum.
(c) Fourier transform multiplied by a Butterworth notch reject filter transfer function.
(d) Filtered image.



interactively from the spectrum. Figure 4.64(d) shows the result obtained with this filter transfer function, using the filtering procedure outlined in Section 4.7. The improvement is significant, considering the low resolution and degree of degradation of the original image.

EXAMPLE 4.25: Using notch filtering to remove periodic interference.

Figure 4.65(a) shows an image of part of the rings surrounding the planet Saturn. This image was captured by *Cassini*, the first spacecraft to enter the planet's orbit. The nearly sinusoidal pattern visible in the image was caused by an AC signal superimposed on the camera video signal just prior to digitizing the image. This was an unexpected problem that corrupted some images from the mission. Fortunately, this type of interference is fairly easy to correct by postprocessing. One approach is to use notch filtering.

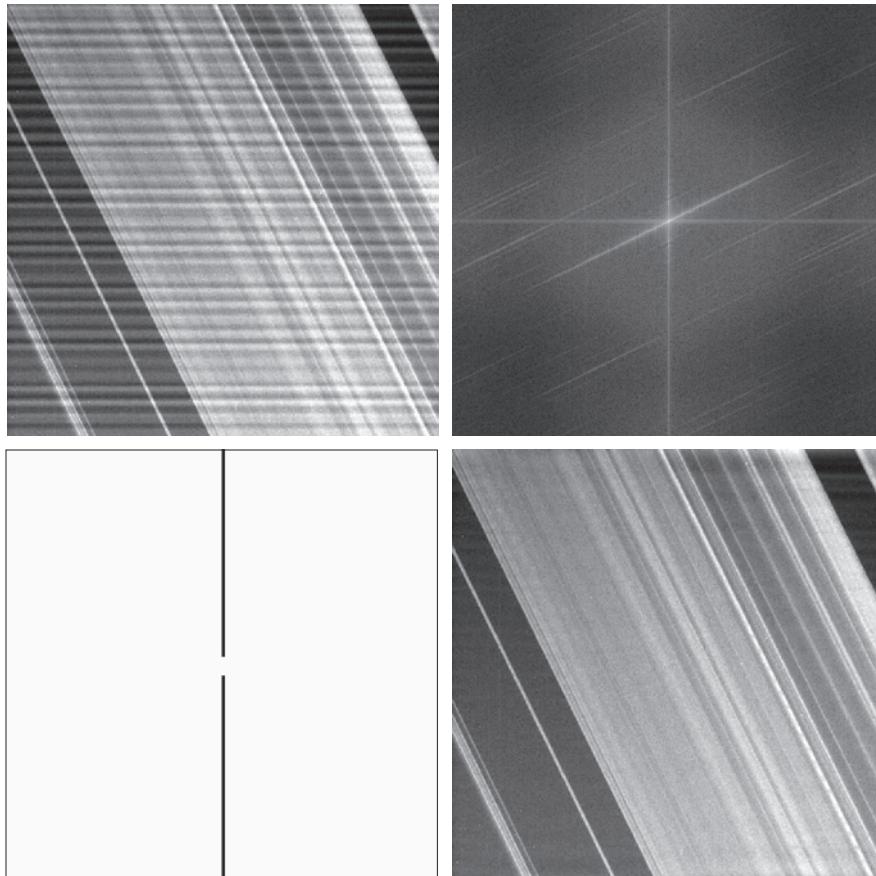
Figure 4.65(b) shows the DFT spectrum. Careful analysis of the vertical axis reveals a series of small bursts of energy near the origin which correspond to the nearly sinusoidal interference. A simple approach is to use a narrow notch rectangle filter starting with the lowest frequency burst, and extending for the remainder of the vertical axis. Figure 4.65(c) shows the transfer function of such a filter (white represents 1 and black 0). Figure 4.65(d) shows the result of processing the corrupted image with this filter. This result is a significant improvement over the original image.

To obtain an image of just the interference pattern, we isolated the frequencies in the vertical axis using a notch pass transfer function, obtained by subtracting the notch reject function from 1 [see Fig. 4.66(a)]. Then, as Fig. 4.66(b) shows, the IDFT of the filtered image is the spatial interference pattern.

a	b
c	d

FIGURE 4.65

- (a) Image of Saturn rings showing nearly periodic interference.
- (b) Spectrum. (The bursts of energy in the vertical axis near the origin correspond to the interference pattern).
- (c) A vertical notch reject filter transfer function.
- (d) Result of filtering. (The thin black border in (c) is not part of the data.) (Original image courtesy of Dr. Robert A. West, NASA/JPL.)



a b

FIGURE 4.66

- (a) Notch pass filter function used to isolate the vertical axis of the DFT of Fig. 4.65(a).
 (b) Spatial pattern obtained by computing the IDFT of (a).



4.11 THE FAST FOURIER TRANSFORM

We have focused attention thus far on theoretical concepts and on examples of filtering in the frequency domain. One thing that should be clear by now is that computational requirements in this area of image processing are not trivial. Thus, it is important to develop a basic understanding of methods by which Fourier transform computations can be simplified and speeded up. This section deals with these issues.

SEPARABILITY OF THE 2-D DFT

As mentioned in Table 4.3, the 2-D DFT is separable into 1-D transforms. We can write Eq. (4-67) as

$$\begin{aligned} F(u, v) &= \sum_{x=0}^{M-1} e^{-j2\pi ux/M} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi vy/N} \\ &= \sum_{x=0}^{M-1} F(x, v) e^{-j2\pi ux/M} \end{aligned} \quad (4-156)$$

where

$$F(x, v) = \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi vy/N} \quad (4-157)$$

For one value of x , and for $v = 0, 1, 2, \dots, N - 1$, we see that $F(x, v)$ is the 1-D DFT of *one row* of $f(x, y)$. By varying x from 0 to $M - 1$ in Eq. (4-157), we compute a set of 1-D DFTs for all rows of $f(x, y)$. The computations in Eq. (4-156) similarly are 1-D transforms of the columns of $F(x, v)$. Thus, we conclude that the 2-D DFT of $f(x, y)$ can be obtained by computing the 1-D transform of each row of $f(x, y)$ and then computing the 1-D transform along each column of the result. This is an important simplification because we have to deal only with one variable at a time. A similar development applies to computing the 2-D IDFT using the 1-D IDFT. However, as we show in the following section, we can compute the IDFT using an algorithm

We could have formulated the preceding two equations to show that a 2-D DFT can be obtained by computing the 1-D DFT of each *column* of the input image followed by 1-D computations on the rows of the result.

designed to compute the forward DFT, so all 2-D Fourier transform computations are reduced to multiple passes of a 1-D algorithm designed for computing the 1-D DFT.

COMPUTING THE IDFT USING A DFT ALGORITHM

Taking the complex conjugate of both sides of Eq. (4-68) and multiplying the results by MN yields

$$MNf^*(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M + vy/N)} \quad (4-158)$$

But, we recognize the form of the right side of this result as the DFT of $F^*(u, v)$. Therefore, Eq. (4-158) indicates that if we substitute $F^*(u, v)$ into an algorithm designed to compute the 2-D forward Fourier transform, the result will be $MNf^*(x, y)$. Taking the complex conjugate and dividing this result by MN yields $f(x, y)$, which is the inverse of $F(u, v)$.

Computing the 2-D inverse from a 2-D forward DFT algorithm that is based on successive passes of 1-D transforms (as in the previous section) is a frequent source of confusion involving the complex conjugates and multiplication by a constant, neither of which is done in the 1-D algorithms. The key concept to keep in mind is that we simply input $F^*(u, v)$ into whatever forward algorithm we have. The result will be $MNf^*(x, y)$. All we have to do with this result to obtain $f(x, y)$ is to take its complex conjugate and divide it by the constant MN . Of course, when $f(x, y)$ is real, as typically is the case, then $f^*(x, y) = f(x, y)$.

THE FAST FOURIER TRANSFORM (FFT)

Work in the frequency domain would not be practical if we had to implement Eqs. (4-67) and (4-68) directly. Brute-force implementation of these equations requires on the order of $(MN)^2$ multiplications and additions. For images of moderate size (say, 2048×2048 pixels), this means on the order of 17 trillion multiplications and additions for just one 2-D DFT, excluding the exponentials, which could be computed once and stored in a look-up table. Without the discovery of the *fast Fourier transform* (FFT), which reduces computations to the order of $MN \log_2 MN$ multiplications and additions, it is safe to say that the material presented in this chapter would be of little practical value. The computational reductions afforded by the FFT are impressive indeed. For example, computing the 2-D FFT of a 2048×2048 image would require on the order of 92 million multiplication and additions, which is a significant reduction from the one trillion computations mentioned above.

Although the FFT is a topic covered extensively in the literature on signal processing, this subject matter is of such significance in our work that this chapter would be incomplete if we did not provide an introduction explaining why the FFT works as it does. The algorithm we selected to accomplish this objective is the so-called *successive-doubling method*, which was the original algorithm that led to the birth of an entire industry. This particular algorithm assumes that the number of samples is an integer power of 2, but this is not a general requirement of other approaches

(Brigham [1988]). We know from the previous section that 2-D DFTs can be implemented by successive passes of the 1-D transform, so we need to focus only on the FFT of one variable.

In derivations of the FFT, it is customary to express Eq. (4-44) in the form

$$F(u) = \sum_{x=0}^{M-1} f(x) W_M^{ux} \quad (4-159)$$

for $u = 0, 1, 2, \dots, M - 1$, where

$$W_M = e^{-j2\pi/M} \quad (4-160)$$

and M is assumed to be of the form

$$M = 2^p \quad (4-161)$$

where p is a positive integer. Then it follows that M can be expressed as

$$M = 2K \quad (4-162)$$

with K being a positive integer also. Substituting Eq. (4-162) into Eq. (4-159) yields

$$\begin{aligned} F(u) &= \sum_{x=0}^{2K-1} f(x) W_{2K}^{ux} \\ &= \sum_{x=0}^{K-1} f(2x) W_{2K}^{u(2x)} + \sum_{x=0}^{K-1} f(2x+1) W_{2K}^{u(2x+1)} \end{aligned} \quad (4-163)$$

However, it can be shown using Eq. (4-160) that $W_{2K}^{2ux} = W_K^{ux}$, so Eq. (4-163) can be written as

$$F(u) = \sum_{x=0}^{K-1} f(2x) W_K^{ux} + \sum_{x=0}^{K-1} f(2x+1) W_K^{ux} W_{2K}^u \quad (4-164)$$

Defining

$$F_{\text{even}}(u) = \sum_{x=0}^{K-1} f(2x) W_K^{ux} \quad (4-165)$$

for $u = 0, 1, 2, \dots, K - 1$, and

$$F_{\text{odd}}(u) = \sum_{x=0}^{K-1} f(2x+1) W_K^{ux} \quad (4-166)$$

for $u = 0, 1, 2, \dots, K - 1$, reduces Eq. (4-164) to

$$F(u) = F_{\text{even}}(u) + F_{\text{odd}}(u) W_{2K}^u \quad (4-167)$$

Also, because $W_M^{u+K} = W_K^u$ and $W_{2K}^{u+K} = -W_{2K}^u$, it follows that

$$F(u+K) = F_{\text{even}}(u) - F_{\text{odd}}(u)W_{2K}^u \quad (4-168)$$

Analysis of Eqs. (4-165) through (4-168) reveals some important (and surprising) properties of these expressions. An M -point DFT can be computed by dividing the original expression into two parts, as indicated in Eqs. (4-167) and (4-168). Computing the first half of $F(u)$ requires evaluation of the two $(M/2)$ -point transforms given in Eqs. (4-165) and (4-166). The resulting values of $F_{\text{even}}(u)$ and $F_{\text{odd}}(u)$ are then substituted into Eq. (4-167) to obtain $F(u)$ for $u = 0, 1, 2, \dots, (M/2 - 1)$. The other half then follows directly from Eq. (4-168) *without* additional transform evaluations.

It is of interest to examine the computational implications of the preceding procedure. Let $m(p)$ and $\alpha(p)$ represent the number of complex multiplications and additions, respectively, required to implement the method. As before, the number of samples is 2^p , where p is a positive integer. Suppose first that $p = 1$ so that the number of samples is two. A two-point transform requires the evaluation of $F(0)$; then $F(1)$ follows from Eq. (4-168). To obtain $F(0)$ requires computing $F_{\text{even}}(0)$ and $F_{\text{odd}}(0)$. In this case $K = 1$ and Eqs. (4-165) and (4-166) are one-point transforms. However, because the DFT of a single sample point is the sample itself, no multiplications or additions are required to obtain $F_{\text{even}}(0)$ and $F_{\text{odd}}(0)$. One multiplication of $F_{\text{odd}}(0)$ by W_2^0 and one addition yields $F(0)$ from Eq. (4-167). Then $F(1)$ follows from Eq. (4-168) with one more addition (subtraction is considered to be the same as addition). Because $F_{\text{odd}}(0)W_2^0$ has been computed already, the total number of operations required for a two-point transform consists of $m(1) = 1$ multiplication and $\alpha(1) = 2$ additions.

The next allowed value for p is 2. According to the preceding development, a four-point transform can be divided into two parts. The first half of $F(u)$ requires evaluation of two two-point transforms, as given in Eqs. (4-165) and (4-166) for $K = 2$. A two-point transform requires $m(1)$ multiplications and $\alpha(1)$ additions. Therefore, evaluation of these two equations requires a total of $2m(1)$ multiplications and $2\alpha(1)$ additions. Two further multiplications and additions are necessary to obtain $F(0)$ and $F(1)$ from Eq. (4-167). Because $F_{\text{odd}}(u)W_{2K}^u$ has been computed already for $u = \{0, 1\}$, two more additions give $F(2)$ and $F(3)$. The total is then $m(2) = 2m(1) + 2$ and $\alpha(2) = 2\alpha(1) + 4$.

When p is equal to 3, two four-point transforms are needed to evaluate $F_{\text{even}}(u)$ and $F_{\text{odd}}(u)$. They require $2m(2)$ multiplications and $2\alpha(2)$ additions. Four more multiplications and eight more additions yield the complete transform. The total then is then $m(3) = 2m(2) + 4$ multiplication and $\alpha(3) = 2\alpha(2) + 8$ additions.

Continuing this argument for any positive integer p leads to recursive expressions for the number of multiplications and additions required to implement the FFT:

$$m(p) = 2m(p-1) + 2^{p-1} \quad p \geq 1 \quad (4-169)$$

and

$$\alpha(p) = 2\alpha(p-1) + 2^p \quad p \geq 1 \quad (4-170)$$

where $m(0) = 0$ and $\alpha(0) = 0$ because the transform of a single point does not require any multiplication or additions.

The method just developed is called the *successive doubling FFT algorithm* because it is based on computing a two-point transform from two one-point transforms, a four-point transform from two two-point transforms, and so on, for any M equal to an integer power of 2. It is left as an exercise (see Problem 4.63) to show that

$$m(p) = \frac{1}{2} M \log_2 M \quad (4-171)$$

and

$$\alpha(n) = M \log_2 M \quad (4-172)$$

where $M = 2^p$.

The computational advantage of the FFT over a direct implementation of the 1-D DFT is defined as

$$\begin{aligned} C(M) &= \frac{M^2}{M \log_2 M} \\ &= \frac{M}{\log_2 M} \end{aligned} \quad (4-173)$$

where M^2 is the number of operations required for a “brute force” implementation of the 1-D DFT. Because it is assumed that $M = 2^p$, we can write Eq. (4-173) in terms of p :

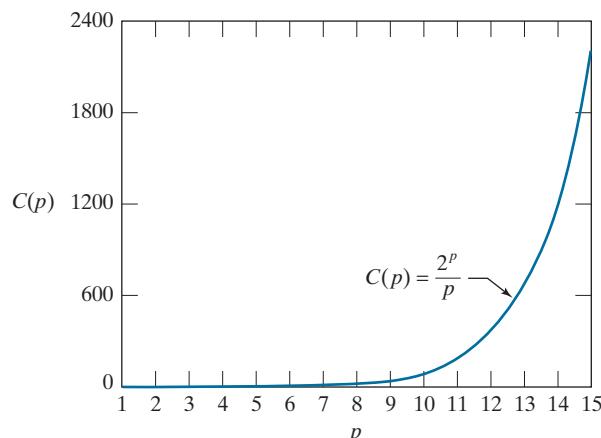
$$C(p) = \frac{2^p}{p} \quad (4-174)$$

A plot of this function (Fig. 4.67) shows that the computational advantage increases rapidly as a function of p . For example, when $p = 15$ (32,768 points), the FFT has nearly a 2,200 to 1 advantage over a brute-force implementation of the DFT. Thus, we would expect that the FFT can be computed nearly 2,200 times faster than the DFT on the same machine. As you learned in Section 4.1, the FFT also offers significant computational advantages over spatial filtering, with the cross-over between the two approaches being for relatively small kernels.

There are many excellent sources that cover details of the FFT so we will not dwell on this topic further (see, for example, Brigham [1988]). Most comprehensive signal and image processing software packages contain generalized implementations of the FFT that do not require the number of points to be an integer power

FIGURE 4.67

Computational advantage of the FFT over a direct implementation of the 1-D DFT. The number of samples is $M = 2^p$. The computational advantage increases rapidly as a function of p .



of 2 (at the expense of slightly less efficient computation). Free FFT programs also are readily available, principally over the internet.

Summary, References, and Further Reading

The material in this chapter is a progression from sampling to the Fourier transform, and then to filtering in the frequency domain. Some of the concepts, such as the sampling theorem, make very little sense if not explained in the context of the frequency domain. The same is true of effects such as aliasing. Thus, the material developed in the preceding sections is a solid foundation for understanding the fundamentals of 2-D digital signal processing. We took special care to develop the material starting with basic principles, so that any reader with a modest mathematical background would be in a position not only to absorb the material, but also to apply it.

For complementary reading on the 1-D and 2-D continuous Fourier transforms, see the books by Bracewell [1995, 2003]. These two books, together with Castleman [1996], Petrou and Petrou [2010], Brigham [1988], and Smith [2003], provide additional background for the material in Sections 4.2 through 4.6. Sampling phenomena such as aliasing and moiré patterns are topics amply illustrated in books on computer graphics, as exemplified by Hughes and Andries [2013]. For additional general background on the material in Sections 4.7 through 4.11 see Hall [1979], Jain [1989], Castleman [1996], and Pratt [2014]. For details on the software aspects of many of the examples in this chapter, see Gonzalez, Woods, and Eddins [2009].

Problems

Solutions to the problems marked with an asterisk (*) are in the DIP4E Student Support Package (consult the book website: www.ImageProcessingPlace.com)

4.1 Answer the following:

- (a)* Give an equation similar to Eq. (4-10), but for an impulse located at $t = t_0$.
- (b) Repeat for Eq. (4-15).
- (c)* Is it correct to say that $\delta(t - a) = \delta(a - t)$ in general? Explain.

4.2 Repeat Example 4.1, but using the function

$f(t) = A$ for $0 \leq t < T$ and $f(t) = 0$ for all other values of t . Explain the reason for any differences between your results and the results in the example.

4.3 What is the convolution of two, 1-D impulses:

- (a)* $\delta(t)$ and $\delta(t - t_0)$?
- (b) $\delta(t - t_0)$ and $\delta(t + t_0)$?

- 4.4*** Use the sifting property of the impulse to show that convolving a 1-D continuous function, $f(t)$, with an impulse located at t_0 shifts the function so that its origin is moved to the location of the impulse (if the impulse is at the origin, the function is not shifted).
- 4.5*** With reference to Fig. 4.9, give a graphical illustration of an aliased pair of functions that are not periodic.
- 4.6** With reference to Fig. 4.11:
- Redraw the figure, showing what the dots would look like for a sampling rate that exceeds the Nyquist rate slightly.
 - What is the *approximate* sampling rate represented by the large dots in Fig. 4.11?
 - Approximately, what would be the lowest sampling rate that you would use so that (1) the Nyquist rate is satisfied, and (2) the samples look like a sine wave?
- 4.7** A function, $f(t)$, is formed by the sum of three functions, $f_1(t) = A\sin(\pi t)$, $f_2(t) = B\sin(4\pi t)$, and $f_3(t) = C\cos(8\pi t)$.
- Assuming that the functions extend to infinity in both directions, what is the highest frequency of $f(t)$? (*Hint:* Start by finding the period of the sum of the three functions.)
 - What is the Nyquist rate corresponding to your result in (a)? (Give a numerical answer.)
 - At what rate would you sample $f(t)$ so that perfect recovery of the function from its samples is possible?
- 4.8*** Show that $\Im\{e^{j2\pi t_0 t}\} = \delta(\mu - t_0)$, where t_0 is a constant. (*Hint:* Study Example 4.2.)
- 4.9** Show that the following expressions are true. (*Hint:* Make use of the solution to Problem 4.8):
- $\Im\{\cos(2\pi\mu_0 t)\} = \frac{1}{2}[\delta(\mu - \mu_0) + \delta(\mu + \mu_0)]$
 - $\Im\{\sin(2\pi\mu_0 t)\} = \frac{1}{2j}[\delta(\mu - \mu_0) - \delta(\mu + \mu_0)]$
- 4.10** Consider the function $f(t) = \sin(2\pi n t)$, where n is an integer. Its Fourier transform, $F(\mu)$, is purely imaginary (see Problem 4.9). Because the transform, $\tilde{F}(\mu)$, of sampled data consists of periodic copies of $F(\mu)$, it follows that $\tilde{F}(\mu)$ will also be purely imaginary. Draw a diagram similar to Fig. 4.6, and answer the following questions based on your diagram (assume that sampling starts at $t = 0$).
- What is the period of $f(t)$?
 - What is the frequency of $f(t)$?
 - What would the sampled function and its Fourier transform look like in general if $f(t)$ is sampled at a rate higher than the Nyquist rate?
 - What would the sampled function look like in general if $f(t)$ is sampled at a rate lower than the Nyquist rate?
 - What would the sampled function look like if $f(t)$ is sampled at the Nyquist rate, with samples taken at $t = 0, \pm\Delta T, \pm 2\Delta T, \dots$?
- 4.11*** Prove the validity of the convolution theorem of one continuous variable, as given in Eqs. (4-25) and (4-26).
- 4.12** We explained in the paragraph after Eq. (4-36) that arbitrarily limiting the duration of a band-limited function by multiplying it by a box function would cause the function to cease being band limited. Show graphically why this is so by limiting the duration of the function $f(t) = \cos(2\pi\mu_0 t)$ [the Fourier transform of this function is given in Problem 4.9(a)]. (*Hint:* The transform of a box function is given in Example 4.1. Use that result in your solution, and also the fact that convolution of a function with an impulse shifts the function to the location of the impulse, in the sense discussed in the solution of Problem 4.4.)
- 4.13*** Complete the steps that led from Eq. (4-37) to Eq. (4-38).
- 4.14** Show that $\tilde{F}(\mu)$ in Eq. (4-40) is infinitely periodic in both directions, with period $1/\Delta T$.
- 4.15** Do the following:
- Show that Eqs. (4-42) and (4-43) are a Fourier transform pair: $f_n \Leftrightarrow F_m$.
 - Show that Eqs. (4-44) and (4-45) also are a Fourier transform pair: $f(x) \Leftrightarrow F(u)$.
- You will need the following orthogonality property in both parts of this problem:
- $$\sum_{x=0}^{M-1} e^{j2\pi r x/M} e^{-j2\pi u x/M} = \begin{cases} M & \text{if } r = u \\ 0 & \text{otherwise} \end{cases}$$

- 4.16** Show that both $F(u)$ and $f(x)$ in Eqs. (4-44) and (4-45) are infinitely periodic with period M ; that is, $F(u) = F(u + kM)$ and $f(x) = f(x + M)$, where k is an integer. [See Eqs. (4-46) and (4-47).]

- 4.17** Demonstrate the validity of the translation (shift) properties of the following 1-D, discrete Fourier transform pairs. (*Hint:* It is easier in part (b) to work with the IDFT.)

(a)* $f(x)e^{j2\pi u_0 x/M} \Leftrightarrow F(u - u_0)$

(b) $f(x - x_0) \Leftrightarrow F(u)e^{-j2\pi ux_0/M}$

- 4.18** Show that the 1-D convolution theorem given in Eqs. (4-25) and (4-26) also holds for discrete variables, but with the right side of Eq. (4-26) multiplied by $1/M$. That is, show that

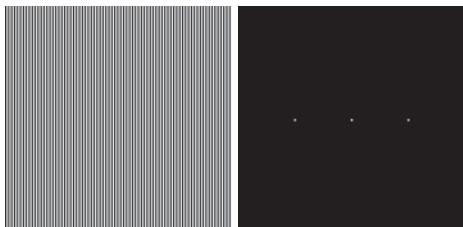
(a)* $(f \star h)(x) \Leftrightarrow (F \cdot H)(u)$, and

(b) $(f \cdot h)(x) \Leftrightarrow \frac{1}{M}(F \star H)(u)$

- 4.19*** Extend the expression for 1-D convolution [see Eq. (4-24)] to two continuous variables. Use t and z for the variables on the left side of the expression and α and β for the variables in the 2-D integral.

- 4.20** Use the sifting property of the 2-D impulse to show that convolution of a 2-D continuous function, $f(t, z)$, with an impulse shifts the function so that its origin is located at the location of the impulse. (If the impulse is at the origin, the function is copied exactly as it was.) (*Hint:* Study the solution to Problem 4.4).

- 4.21** The image on the left in the figure below consists of alternating stripes of black/white, each stripe



being two pixels wide. The image on the right is the Fourier spectrum of the image on the left, showing the dc term and the frequency terms corresponding to the stripes. (Remember, the spectrum is symmetric so all components, other than the dc term, appear in two symmetric locations.)

(a)* Suppose that the stripes of an image of the

same size are four pixels wide. Sketch what the spectrum of the image would look like, including only the dc term and the two highest-value frequency terms, which correspond to the two spikes in the spectrum above.

- (b)** Why are the components of the spectrum limited to the horizontal axis?

- (c)** What would the spectrum look like for an image of the same size but having stripes that are one pixel wide? Explain the reason for your answer.

- (d)** Are the dc terms in (a) and (c) the same, or are they different? Explain.

- 4.22** A high-technology company specializes in developing imaging systems for digitizing images of commercial cloth. The company has a new order for 1,000 systems for digitizing cloth consisting of repeating black and white vertical stripes, each of width 2 cm. Optical and mechanical engineers have already designed the front-end optics and mechanical positioning mechanisms so that you are guaranteed that every image your system digitizes starts with a complete black vertical stripe and ends with a complete white stripe. Every image acquired will contain exactly 250 vertical stripes. Noise and optical distortions are negligible. Having learned of your success in taking an image processing course, the company employs you to specify the resolution of the imaging chip to be used in the new system. The optics can be adjusted to project the field of view accurately onto the area defined by the size of the chip you specify. Your design will be implemented in hundreds of locations, so cost is an important consideration. What resolution chip (in terms of number of imaging elements per horizontal line) would you specify to avoid aliasing?

- 4.23*** We know from the discussion in Section 4.5 that zooming or shrinking a digital image generally causes aliasing. Give an example of an image that would be free of aliasing if it were zoomed by pixel replication.

- 4.24** With reference to the discussion on linearity in Section 2.6, demonstrate that

- (a)* The 2-D continuous Fourier transform is a linear operator.

- (b) The 2-D DFT is a linear operator also.

- 4.25** With reference to Eqs. (4-59) and (4-60), show the validity of the following translation (shift) properties of 2-D, *continuous* Fourier transform pairs. (*Hint:* Study the solutions to Problem 4.11.)
- (a)* $f(t, z)e^{j2\pi(\mu_0 t + \nu_0 z)} \Leftrightarrow F(\mu - \mu_0, \nu - \nu_0)$
- (b) $f(t - t_0, z - z_0) \Leftrightarrow F(\mu, \nu)e^{-j2\pi(t_0\mu + z_0\nu)}$
- 4.26** Show the validity of the following 2-D *continuous* Fourier transform pairs.
- (a)* $\delta(t, z) \Leftrightarrow 1$
- (b)* $1 \Leftrightarrow \delta(\mu, \nu)$
- (c)* $\delta(t - t_0, z - z_0) \Leftrightarrow e^{-j2\pi(t_0\mu + z_0\nu)}$
- (d) $e^{j2\pi(t_0 t + z_0 z)} \Leftrightarrow \delta(\mu - t_0, \nu - z_0)$
- (e)* $\cos(2\pi\mu_0 t + 2\pi\nu_0 z) \Leftrightarrow$
 $(1/2)[\delta(\mu - \mu_0, \nu - \nu_0) + \delta(\mu + \mu_0, \nu + \nu_0)]$
- (f) $\sin(2\pi\mu_0 t + 2\pi\nu_0 z) \Leftrightarrow$
 $(1/2j)[\delta(\mu - \mu_0, \nu - \nu_0) - \delta(\mu + \mu_0, \nu + \nu_0)]$
- 4.27** With reference to Eqs. (4-71) and (4-72), demonstrate the validity of the following translation (shifting) properties of 2-D, *discrete* Fourier transform pairs from Table 4.4. (*Hint:* Study the solutions to Problem 4.17.)
- (a) $f(x, y)e^{j2\pi(u_0 x/M + v_0 y/N)} \Leftrightarrow F(u - u_0, v - v_0)$
- (b)* $f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-j2\pi(x_0 u/M + y_0 v/N)}$
- 4.28** Show the validity of the following 2-D *discrete* Fourier transform pairs from Table 4.4:
- (a)* $\delta(x, y) \Leftrightarrow 1$
- (b)* $1 \Leftrightarrow MN\delta(u, v)$
- (c) $\delta(x - x_0, y - y_0) \Leftrightarrow e^{-j2\pi(ux_0/M + vy_0/N)}$
- (d)* $e^{j2\pi(u_0 x/M + v_0 y/N)} \Leftrightarrow MN\delta(u - u_0, v - v_0)$
- (e) $\cos(2\pi\mu_0 x/M + 2\pi\nu_0 y/N) \Leftrightarrow$
 $(MN/2)[\delta(u + \mu_0, v + \nu_0) + \delta(u - \mu_0, v - \nu_0)]$
- (f)* $\sin(2\pi\mu_0 x/M + 2\pi\nu_0 y/N) \Leftrightarrow$
 $(jMN/2)[\delta(u + \mu_0, v + \nu_0) - \delta(u - \mu_0, v - \nu_0)]$
- 4.29** You are given a “canned” program that computes the 2-D, DFT pair. However, it is not known in which of the two equations the $1/MN$ term is included or if it was split as two constants, $1/\sqrt{MN}$, in front of both the forward and inverse transforms. How can you find where the term(s) is (are) included if this information is not available in the documentation?
- 4.30** What is period and frequency of each of following digital sequences (*Hint:* Think of these as square waves.)
- (a)* 0 1 0 1 0 1 0 1 ...
- (b) 0 0 1 0 0 1 0 0 1 ...
- (c) 0 0 1 1 0 0 1 1 0 0 1 1 ...
- 4.31** With reference to the 1-D sequences in Example 4.10:
- (a)* When M is even, why is the point at $M/2$ in an even sequence always arbitrary?
- (b) When M is even, why is the point at $M/2$ in an odd sequence always 0?
- 4.32** We mentioned in Example 4.10 that embedding a 2-D array of even (odd) dimensions into a larger array of zeros of even (odd) dimensions keeps the symmetry of the original array, provided that the centers coincide. Show that this is true also for the following 1-D arrays (i.e., show that the larger arrays have the same symmetry as the smaller arrays). For arrays of even length, use arrays of 0's ten elements long. For arrays of odd lengths, use arrays of 0's nine elements long.
- (a)* $\{a, b, c, c, b\}$
- (b) $\{0, -b, -c, 0, c, b\}$
- (c) $\{a, b, c, d, c, b\}$
- (d) $\{0, -b, -c, c, b\}$
- 4.33** In Example 4.10 we showed a Sobel kernel embedded in a field of zeros. The kernel is of size 3×3 and its structure appears to be odd. However, its first element is -1 , and we know that in order to be odd, the first (top, left) element a 2-D array must be zero. Show the smallest field of zeros in which you can embed the Sobel kernel so that it satisfies the condition of oddness.
- 4.34** Do the following:
- (a)* Show that the 6×6 array in Example 4.10 is odd.
- (b) What would happen if the minus signs are changed to pluses?
- (c) Explain why, as stated at the end of the example, adding to the array another row of 0's on the top and column of 0's to the left would give a result that is neither even nor odd.
- (d) Suppose that the row is added to the bot-

tom and the column to the right? Would that change your answer in (c)?

- 4.35** The following problems are related to the properties in Table 4.1.

- (a)* Demonstrate the validity of property 2.
- (b)* Demonstrate the validity of property 4.
- (c) Demonstrate the validity of property 5.
- (d)* Demonstrate the validity of property 7.
- (e) Demonstrate the validity of property 9.

- 4.36** You know from Table 4.3 that the dc term, $F(0,0)$, of a DFT is proportional to the average value of its corresponding spatial image. Assume that the image is of size $M \times N$. Suppose that you pad the image with zeros to size $P \times Q$, where P and Q are given in Eqs. (4-102) and (4-103). Let $F_p(0,0)$ denote the dc term of the DFT of the padded function.

- (a)* What is the ratio of the average values of the original and padded images?
- (b) Is $F_p(0,0) = F(0,0)$? Support your answer mathematically.

- 4.37** Demonstrate the validity of the periodicity properties (entry 8) in Table 4.3.

- 4.38** With reference to the 2-D discrete convolution theorem in Eqs. (4-95) and (4-96) (entry 6 in Table 4.4), show that

- (a) $(f \star h)(x,y) \Leftrightarrow (F \bullet H)(u,v)$
 - (b)* $(f \bullet h)(x,y) \Leftrightarrow (1/MN)[(F \star H)(u,v)]$
- (Hint: Study the solution to Problem 4.18.)

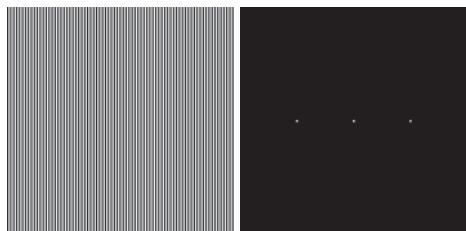
- 4.39** With reference to the 2-D discrete correlation theorem (entry 7 in Table 4.4), show that

- (a)* $(f \star\! h)(x,y) \Leftrightarrow (F^* \bullet H)(u,v)$
- (b) $(f^* \bullet h)(x,y) \Leftrightarrow (1/MN)[(F \star H)(u,v)]$

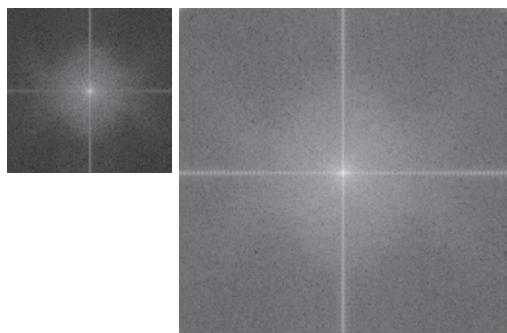
- 4.40*** Demonstrate validity of the differentiation pairs in entry 12 of Table 4.4.

- 4.41** We discussed in Section 4.6 the need for image padding when filtering in the frequency domain. We showed in that section that images could be padded by appending zeros to the ends of rows and columns in the image (see the following image, on the left). Do you think it would make a difference if we centered the image and surround-

ed it by a border of zeros instead (see image on the right), but without changing the total number of zeros used? Explain.



- 4.42*** The two Fourier spectra shown are of the same image. The spectrum on the left corresponds to the original image, and the spectrum on the right was obtained after the image was padded with zeros. Explain the significant increase in signal strength along the vertical and horizontal axes of the spectrum shown on the right.



- 4.43** Consider the images shown. The image on the right was obtained by: (a) multiplying the image on the left by $(-1)^{x+y}$; (b) computing the DFT; (c) taking the complex conjugate of the transform; (d) computing the inverse DFT; and (e) multiplying the real part of the result by $(-1)^{x+y}$. Explain (mathematically) why the image on the right appears as it does.



4.44* The image in Fig. 4.34(b) was obtained by multiplying by -1 the phase angle of the image in Fig. 4.34(a), and then computing the IDFT. With reference to Eq. (4-86) and entry 5 in Table 4.1, explain why this operation caused the image to be reflected about both coordinate axes.

4.45 In Fig. 4.34(b) we saw that multiplying the phase angle by -1 flipped the image with respect to both coordinate axes. Suppose that instead we multiplied the magnitude of the transform by -1 and then took the inverse DFT using the equation: $g(x,y) = \mathfrak{I}^{-1}\{-|F(u,v)|e^{j\phi(u,v)}\}$.

(a)* What would be the difference between the two images $g(x,y)$ and $f(x,y)$? [Remember, $F(u,v)$ is the DFT of $f(x,y)$.]

(b) Assuming that they are both 8-bit images, what would $g(x,y)$ look like in terms of $f(x,y)$ if we scaled the intensity values of $g(x,y)$ using Eqs. (2-31) and (2-32), with $K = 255$?

4.46 What is the source of the nearly periodic bright spots on the horizontal axis of Fig. 4.40(b)?

4.47* Consider a 3×3 spatial kernel that averages the four closest neighbors of a point (x,y) , but excludes the point itself from the average.

- (a) Find the equivalent filter transfer function, $H(u,v)$, in the frequency domain.
- (b) Show that your result is a lowpass filter transfer function.

4.48* A continuous Gaussian lowpass filter in the continuous frequency domain has the transfer function

$$H(\mu,\nu) = Ae^{-(\mu^2 + \nu^2)/2\sigma^2}$$

Show that the corresponding filter kernel in the continuous spatial domain is

$$h(t,z) = A2\pi\sigma^2 e^{-2\pi^2\sigma^2(t^2 + z^2)}$$

4.49 Given an image of size $M \times N$, you are asked to perform an experiment that consists of repeatedly lowpass filtering the image in the frequency domain using a Gaussian lowpass filter transfer function with a cutoff frequency, D_0 . You may ignore computational round-off errors.

(a)* Let K denote the number of applications of

the filter. Can you predict (without doing the experiment) what the result (image) will be for a sufficiently large value of K ? If so, what is that result?

(b) Let c_{\min} denote the smallest positive number representable in the machine in which the proposed experiment will be conducted (any number $< c_{\min}$ is automatically set to 0). Derive an expression (in terms of c_{\min}) for the minimum value of K that will guarantee the result that you predicted in (a).

4.50 As explained in Section 3.6, first-order derivatives can be approximated by the spatial differences $g_x = \partial f(x,y)/\partial x = f(x+1,y) - f(x,y)$ and $g_y = \partial f(x,y)/\partial y = f(x,y+1) - f(x,y)$.

- (a) Find the equivalent filter transfer functions $H_x(u,v)$ and $H_y(u,v)$ in the frequency domain.
- (b) Show that these are highpass filter transfer functions.

(Hint: Study the solution to Problem 4.47.)

4.51 Find the equivalent frequency-domain filter transfer function for the Laplacian kernel shown in Fig. 3.45(a). Show that your result behaves as a highpass filter transfer function. (Hint: Study the solution to Problem 4.47.)

4.52 Do the following:

- (a) Show that the Laplacian of a continuous function $f(t,z)$ of two continuous variables, t and z , satisfies the following Fourier transform pair:

$$\nabla^2 f(t,z) \Leftrightarrow -4\pi^2(\mu^2 + \nu^2)F(\mu,\nu)$$

(Hint: See Eq. (3-50) and study entry 12 in Table 4.4.)

(b)* The result in (a) is valid only for continuous variables. How would you implement the continuous frequency domain transfer function $H(\mu,\nu) = -4\pi^2(\mu^2 + \nu^2)$ for discrete variables?

- (c) As you saw in Example 4.21, the Laplacian result in the frequency domain was similar to the result in Fig. 3.46(d), which was obtained using a spatial kernel with a center coefficient equal to -8 . Explain why the frequency domain result was not similar instead to the

result in Fig. 3.46(c), which was obtained using a kernel with a center coefficient of -4 .

- 4.53*** Can you think of a way to use the Fourier transform to compute (or partially compute) the magnitude of the gradient [Eq. (3-58)] for use in image differentiation? If your answer is yes, give a method to do it. If your answer is no, explain why.
- 4.54** As explained in Eq. (4-118), it is possible to obtain the transfer function of a highpass filter from the transfer function of a lowpass filter by subtracting the latter from 1. What is the highpass spatial kernel corresponding to the lowpass Gaussian transfer function given in Problem 4.48?
- 4.55** Each spatial highpass kernel in Fig. 4.52 has a strong spike in the center. Explain the source of this spikes.
- 4.56*** Show how the Butterworth highpass filter transfer function in Eq. (4-121) follows from its low-pass counterpart in Eq. (4-117).
- 4.57** Consider the hand X-ray images shown below. The image on the right was obtained by lowpass



(Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

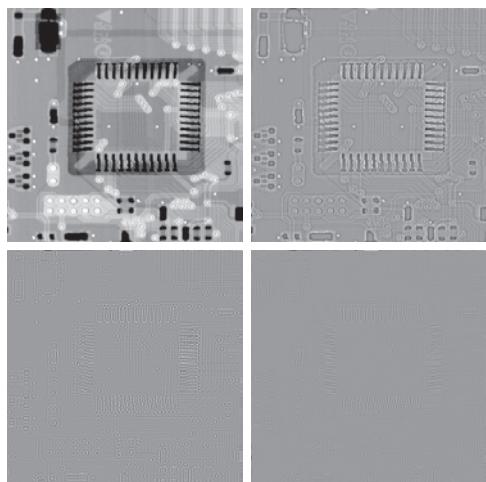
filtering the image on the left with a Gaussian lowpass filter, and then highpass filtering the result with a Gaussian highpass filter. The images are of size 420×344 pixels and $D_0 = 25$ was used for both filter transfer functions.

(a)* Explain why the center part of the finger ring in the figure on the right appears so bright and solid, considering that the dominant characteristic of the filtered image consists of edges of the fingers and wrist bones, with darker areas in between. In other words, would you not expect the highpass filter to render the constant area inside the ring as

dark, since a highpass filter eliminates the dc term and reduces low frequencies?

- (b)** Do you think the result would have been different if the order of the filtering process had been reversed?

- 4.58** Consider the sequence of images shown below. The image on the top left is a segment of an X-ray image of a commercial printed circuit board. The images following it are, respectively, the results of subjecting the image to 1, 10, and 100 passes of a Gaussian highpass filter with $D_0 = 30$. The images are of size 330×334 pixels, with each pixel being represented by 8 bits of gray. The images were scaled for display, but this has no effect on the problem statement.



(Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

- (a)** It appears from the images that changes will cease to take place after a finite number of passes. Show whether or not this is the case. You may ignore computational round-off errors. Let c_{\min} denote the smallest positive number representable in the machine in which the computations are conducted.

- (b)** If you determined in (a) that changes would cease after a finite number of iterations, determine the minimum value of that number.

(Hint: Study the solution to Problem 4.49.)

- 4.59** As illustrated in Fig. 4.57, combining high-frequency emphasis and histogram equalization is

an effective method for achieving edge sharpening and contrast enhancement.

(a)* Show whether or not it matters which process is applied first.

(b) If the order does matter, give a rationale for using one or the other method first.

4.60 Use a Butterworth highpass filter to construct a homomorphic filter transfer function that has the same general shape as the function in Fig. 4.59.

4.61 Suppose that you are given a set of images generated by an experiment dealing with the analysis of stellar events. Each image contains a set of bright, widely scattered dots corresponding to stars in a sparsely occupied region of the universe. The problem is that the stars are barely visible as a result of superimposed illumination from atmospheric dispersion. If these images are modeled as the product of a constant illumination component with a set of impulses, give an enhancement procedure based on homomorphic filtering designed to bring out the image components due to the stars themselves.

4.62 How would you generate an image of only the interference pattern visible in Fig. 4.64(a)?

4.63* Show the validity of Eqs. (4-171) and (4-172). (*Hint:* Use proof by induction.)

4.64 A skilled medical technician is assigned the job of inspecting a set of images generated by an electron microscope experiment. In order to simplify the inspection task, the technician decides to use digital image enhancement and, to this end, examines a set of representative images and finds the following problems: (1) bright, isolated dots that are of no interest; (2) lack of sharpness; (3) not enough contrast in some images; and (4) shifts in the average intensity to values other than A_0 , which is the average value required to perform correctly certain intensity measurements. The technician wants to correct these problems and then display in white all intensities in a band between intensities I_1 and I_2 , while keeping normal tonality in the remaining intensities. Propose a sequence of processing steps that the technician can follow to achieve the desired goal. You may use techniques from both Chapters 3 and 4.

This page intentionally left blank

5

Image Restoration and Reconstruction



Things which we see are not themselves what we see . . .
It remains completely unknown to us what the objects may be
by themselves and apart from the receptivity of our senses.
We know only but our manner of perceiving them.

Immanuel Kant

Preview

As in image enhancement, the principal goal of restoration techniques is to improve an image in some predefined sense. Although there are areas of overlap, image enhancement is largely a subjective process, while image restoration is for the most part an objective process. Restoration attempts to recover an image that has been degraded by using a priori knowledge of the degradation phenomenon. Thus, restoration techniques are oriented toward modeling the degradation and applying the inverse process in order to recover the original image. In this chapter, we consider linear, space invariant restoration models that are applicable in a variety of restoration situations. We also discuss fundamental techniques of image reconstruction from projections, and their application to computed tomography (CT), one of the most important commercial applications of image processing, especially in health care.

Upon completion of this chapter, readers should:

- Be familiar with the characteristics of various noise models used in image processing, and how to estimate from image data the parameters that define those models.
- Be familiar with linear, nonlinear, and adaptive spatial filters used to restore (denoise) images that have been degraded only by noise.
- Know how to apply notch filtering in the frequency domain for removing periodic noise in an image.
- Understand the foundation of linear, space invariant system concepts, and how they can be applied in formulating image restoration solutions in the frequency domain.
- Be familiar with direct inverse filtering and its limitations.
- Understand minimum mean-square-error (Wiener) filtering and its advantages over direct inverse filtering.
- Understand constrained, least-squares filtering.
- Be familiar with the fundamentals of image reconstruction from projections, and their application to computed tomography.

5.1 A MODEL OF THE IMAGE DEGRADATION/RESTORATION PROCESS

In this chapter, we model image degradation as an operator \mathcal{H} that, together with an additive noise term, operates on an input image $f(x, y)$ to produce a degraded image $g(x, y)$ (see Fig. 5.1). Given $g(x, y)$, some knowledge about \mathcal{H} , and some knowledge about the additive noise term $\eta(x, y)$, the objective of restoration is to obtain an estimate $\hat{f}(x, y)$ of the original image. We want the estimate to be as close as possible to the original image and, in general, the more we know about \mathcal{H} and η , the closer $\hat{f}(x, y)$ will be to $f(x, y)$.

We will show in Section 5.5 that, if \mathcal{H} is a linear, position-invariant operator, then the degraded image is given in the spatial domain by

$$g(x, y) = (h \star f)(x, y) + \eta(x, y) \quad (5-1)$$

where $h(x, y)$ is the spatial representation of the degradation function. As in Chapters 3 and 4, the symbol “ \star ” indicates convolution. It follows from the convolution theorem that the equivalent of Eq. (5-1) in the frequency domain is

$$G(u, v) = H(u, v)F(u, v) + N(u, v) \quad (5-2)$$

where the terms in capital letters are the Fourier transforms of the corresponding terms in Eq. (5-1). These two equations are the foundation for most of the restoration material in this chapter.

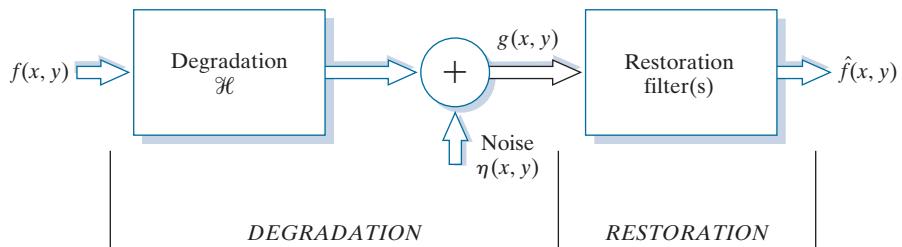
In the following three sections, we work only with degradations caused by noise. Beginning in Section 5.5 we look at several methods for image restoration in the presence of both \mathcal{H} and η .

5.2 NOISE MODELS

The principal sources of noise in digital images arise during image acquisition and/or transmission. The performance of imaging sensors is affected by a variety of environmental factors during image acquisition, and by the quality of the sensing elements themselves. For instance, in acquiring images with a CCD camera, light levels and sensor temperature are major factors affecting the amount of noise in the resulting image. Images are corrupted during transmission principally by interference in the transmission channel. For example, an image transmitted using a wireless network might be corrupted by lightning or other atmospheric disturbance.

FIGURE 5.1

A model of the image degradation/restoration process.



SPATIAL AND FREQUENCY PROPERTIES OF NOISE

Relevant to our discussion are parameters that define the spatial characteristics of noise, and whether the noise is correlated with the image. Frequency properties refer to the frequency content of noise in the Fourier (frequency) domain discussed in detail in Chapter 4. For example, when the Fourier spectrum of noise is constant, the noise is called *white noise*. This terminology is a carryover from the physical properties of white light, which contains all frequencies in the visible spectrum in equal proportions.

With the exception of spatially periodic noise, we assume in this chapter that noise is independent of spatial coordinates, and that it is uncorrelated with respect to the image itself (that is, there is no correlation between pixel values and the values of noise components). Although these assumptions are at least partially invalid in some applications (quantum-limited imaging, such as in X-ray and nuclear-medicine imaging, is a good example), the complexities of dealing with spatially dependent and correlated noise are beyond the scope of our discussion.

SOME IMPORTANT NOISE PROBABILITY DENSITY FUNCTIONS

You may find it helpful to take a look at the Tutorials section of the book website for a brief review of probability.

In the discussion that follows, we shall be concerned with the statistical behavior of the intensity values in the noise component of the model in Fig. 5.1. These may be considered random variables, characterized by a probability density function (PDF), as noted briefly as noted earlier. The noise component of the model in Fig. 5.1 is an image, $\eta(x, y)$, of the same size as the input image. We create a noise image for simulation purposes by generating an array whose intensity values are random numbers with a specified probability density function. This approach is true for all the PDFs to be discussed shortly, with the exception of salt-and-pepper noise, which is applied differently. The following are among the most common noise PDFs found in image processing applications.

Gaussian Noise

Because of its mathematical tractability in both the spatial and frequency domains, Gaussian noise models are used frequently in practice. In fact, this tractability is so convenient that it often results in Gaussian models being used in situations in which they are marginally applicable at best.

The PDF of a *Gaussian* random variable, z , is defined by the following familiar expression:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z - \bar{z})^2}{2\sigma^2}} \quad -\infty < z < \infty \quad (5-3)$$

where z represents intensity, \bar{z} is the mean (average) value of z , and σ is its standard deviation. Figure 5.2(a) shows a plot of this function. We know that for a Gaussian random variable, the probability that values of z are in the range $\bar{z} \pm \sigma$ is approximately 0.68; the probability is about 0.95 that the values of z are in the range $\bar{z} \pm 2\sigma$.

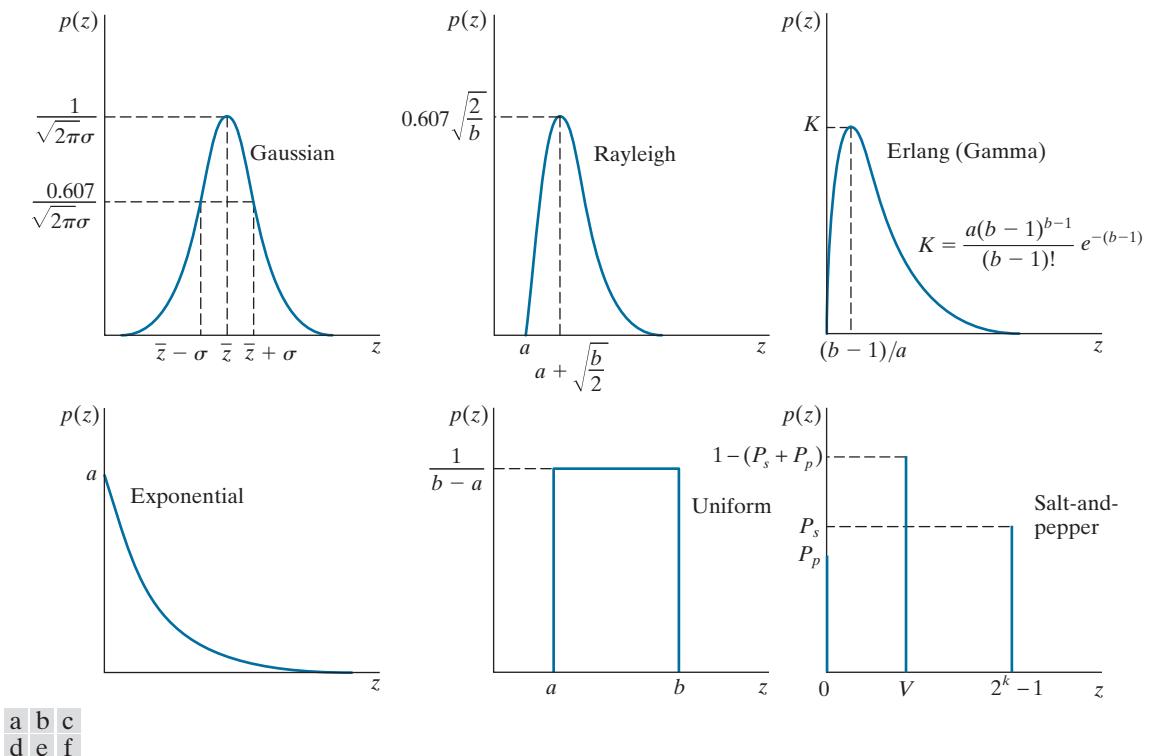


FIGURE 5.2 Some important probability density functions.

Rayleigh Noise

The PDF of *Rayleigh* noise is given by

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & z \geq a \\ 0 & z < a \end{cases} \quad (5-4)$$

The mean and variance of z when this random variable is characterized by a Rayleigh PDF are

$$\bar{z} = a + \sqrt{\pi b/4} \quad (5-5)$$

and

$$\sigma^2 = \frac{b(4-\pi)}{4} \quad (5-6)$$

Figure 5.2(b) shows a plot of the Rayleigh density. Note the displacement from the origin, and the fact that the basic shape of the density is skewed to the right. The Rayleigh density can be quite useful for modeling the shape of skewed histograms.

Erlang (Gamma) Noise

The PDF of Erlang noise is

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases} \quad (5-7)$$

where the parameters are such that $a > b$, b is a positive integer, and “!” indicates factorial. The mean and variance of z are

$$\bar{z} = \frac{b}{a} \quad (5-8)$$

and

$$\sigma^2 = \frac{b}{a^2} \quad (5-9)$$

Figure 5.2(c) shows a plot of this density. Although Eq. (5-9) often is referred to as the *gamma* density, strictly speaking this is correct only when the denominator is the gamma function, $\Gamma(b)$. When the denominator is as shown, the density is more appropriately called the *Erlang* density.

Exponential Noise

The PDF of *exponential* noise is given by

$$p(z) = \begin{cases} ae^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases} \quad (5-10)$$

where $a > 0$. The mean and variance of z are

$$\bar{z} = \frac{1}{a} \quad (5-11)$$

and

$$\sigma^2 = \frac{1}{a^2} \quad (5-12)$$

Note that this PDF is a special case of the Erlang PDF with $b = 1$. Figure 5.2(d) shows a plot of the exponential density function.

Uniform Noise

The PDF of *uniform* noise is

$$p(z) = \begin{cases} \frac{1}{b-a} & a \leq z \leq b \\ 0 & \text{otherwise} \end{cases} \quad (5-13)$$

The mean and variance of z are

$$\bar{z} = \frac{a + b}{2} \quad (5-14)$$

and

$$\sigma^2 = \frac{(b - a)^2}{12} \quad (5-15)$$

Figure 5.2(e) shows a plot of the uniform density.

Salt-and-Pepper Noise

If k represents the number of bits used to represent the intensity values in a digital image, then the range of possible intensity values for that image is $[0, 2^k - 1]$ (e.g., [0, 255] for an 8-bit image). The PDF of *salt-and-pepper* noise is given by

$$p(z) = \begin{cases} P_s & \text{for } z = 2^k - 1 \\ P_p & \text{for } z = 0 \\ 1 - (P_s + P_p) & \text{for } z = V \end{cases} \quad (5-16)$$

When image intensities are scaled to the range $[0, 1]$, we replace by 1 the value of salt in this equation. V then becomes a fractional value in the open interval $(0, 1)$.

where V is any integer value in the range $0 < V < 2^k - 1$.

Let $\eta(x, y)$ denote a salt-and-pepper noise image, whose intensity values satisfy Eq. (5-16). Given an image, $f(x, y)$, of the same size as $\eta(x, y)$, we corrupt it with salt-and-pepper noise by assigning a 0 to all locations in f where a 0 occurs in η . Similarly, we assign a value of $2^k - 1$ to all location in f where that value appears in η . Finally, we leave unchanged all location in f where V occurs in η .

If neither P_s nor P_p is zero, and especially if they are equal, noise values satisfying Eq. (5-16) will be white ($2^k - 1$) or black (0), and will resemble salt and pepper granules distributed randomly over the image; hence the name of this type of noise. Other names you will find used in the literature are *bipolar impulse noise* (*unipolar* if either P_s or P_p is 0), *data-drop-out noise*, and *spike noise*. We use the terms impulse and salt-and-pepper noise interchangeably.

The probability, P , that a pixel is corrupted by salt or pepper noise is $P = P_s + P_p$. It is common terminology to refer to P as the *noise density*. If, for example, $P_s = 0.02$ and $P_p = 0.01$, then $P = 0.03$ and we say that approximately 2% of the pixels in an image are corrupted by salt noise, 1% are corrupted by pepper noise, and the noise density is 3%, meaning that approximately 3% of the pixels in the image are corrupted by salt-and-pepper noise.

Although, as you have seen, salt-and-pepper noise is specified by the probability of each, and not by the mean and variance, we include the latter here for completeness. The mean of salt-and-pepper noise is given by

$$\bar{z} = (0)P_p + K(1 - P_s - P_p) + (2^k - 1)P_s \quad (5-17)$$

and the variance by

$$\sigma^2 = (0 - \bar{z})^2 P_p + (K - \bar{z})^2 (1 - P_s - P_p) + (2^k - 1)^2 P_s \quad (5-18)$$

where we have included 0 as a value explicit in both equations to indicate that the value of pepper noise is assumed to be zero.

As a group, the preceding PDFs provide useful tools for modeling a broad range of noise corruption situations found in practice. For example, Gaussian noise arises in an image due to factors such as electronic circuit noise and sensor noise caused by poor illumination and/or high temperature. The Rayleigh density is helpful in characterizing noise phenomena in range imaging. The exponential and gamma densities find application in laser imaging. Impulse noise is found in situations where quick transients, such as faulty switching, take place during imaging. The uniform density is perhaps the least descriptive of practical situations. However, the uniform density is quite useful as the basis for numerous random number generators that are used extensively in simulations (Gonzalez, Woods, and Eddins [2009]).

EXAMPLE 5.1: Noisy images and their histograms.

Figure 5.3 shows a test pattern used for illustrating the noise models just discussed. This is a suitable pattern to use because it is composed of simple, constant areas that span the gray scale from black to near white in only three increments. This facilitates visual analysis of the characteristics of the various noise components added to an image.

Figure 5.4 shows the test pattern after addition of the six types of noise in Fig. 5.2. Below each image is the histogram computed directly from that image. The parameters of the noise were chosen in each case so that the histogram corresponding to the three intensity levels in the test pattern would start to merge. This made the noise quite visible, without obscuring the basic structure of the underlying image.

We see a close correspondence in comparing the histograms in Fig. 5.4 with the PDFs in Fig. 5.2. The histogram for the salt-and-pepper example does not contain a specific peak for V because, as you will recall, V is used only during the creation of the noise image to leave values in the original image unchanged. Of course, in addition to the salt and pepper peaks, there are peaks for the other intensities in the image. With the exception of slightly different overall intensity, it is difficult to differentiate

FIGURE 5.3

Test pattern used to illustrate the characteristics of the PDFs from Fig. 5.2.



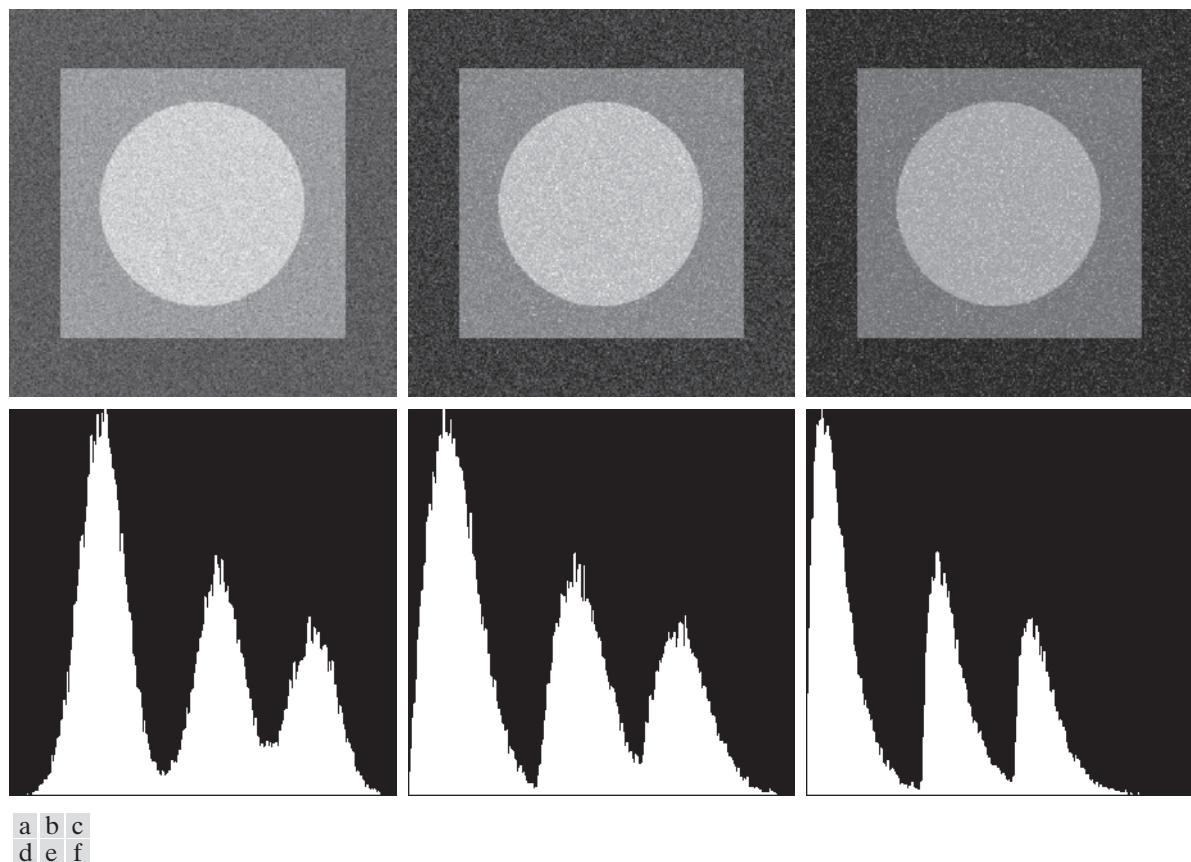


FIGURE 5.4 Images and histograms resulting from adding Gaussian, Rayleigh, and Erlanga noise to the image in Fig. 5.3.

visually between the first five images in Fig. 5.4, even though their histograms are significantly different. The salt-and-pepper appearance of the image in Fig. 5.4(i) is the only one that is visually indicative of the type of noise causing the degradation.

PERIODIC NOISE

Periodic noise in images typically arises from electrical or electromechanical interference during image acquisition. This is the only type of spatially dependent noise we will consider in this chapter. As we will discuss in Section 5.4, periodic noise can be reduced significantly via frequency domain filtering. For example, consider the image in Fig. 5.5(a). This image is corrupted by additive (spatial) sinusoidal noise. The Fourier transform of a pure sinusoid is a pair of conjugate impulses[†] located at

[†] Be careful not to confuse the term *impulse* in the frequency domain with the use of the same term in impulse noise discussed earlier, which is in the spatial domain.

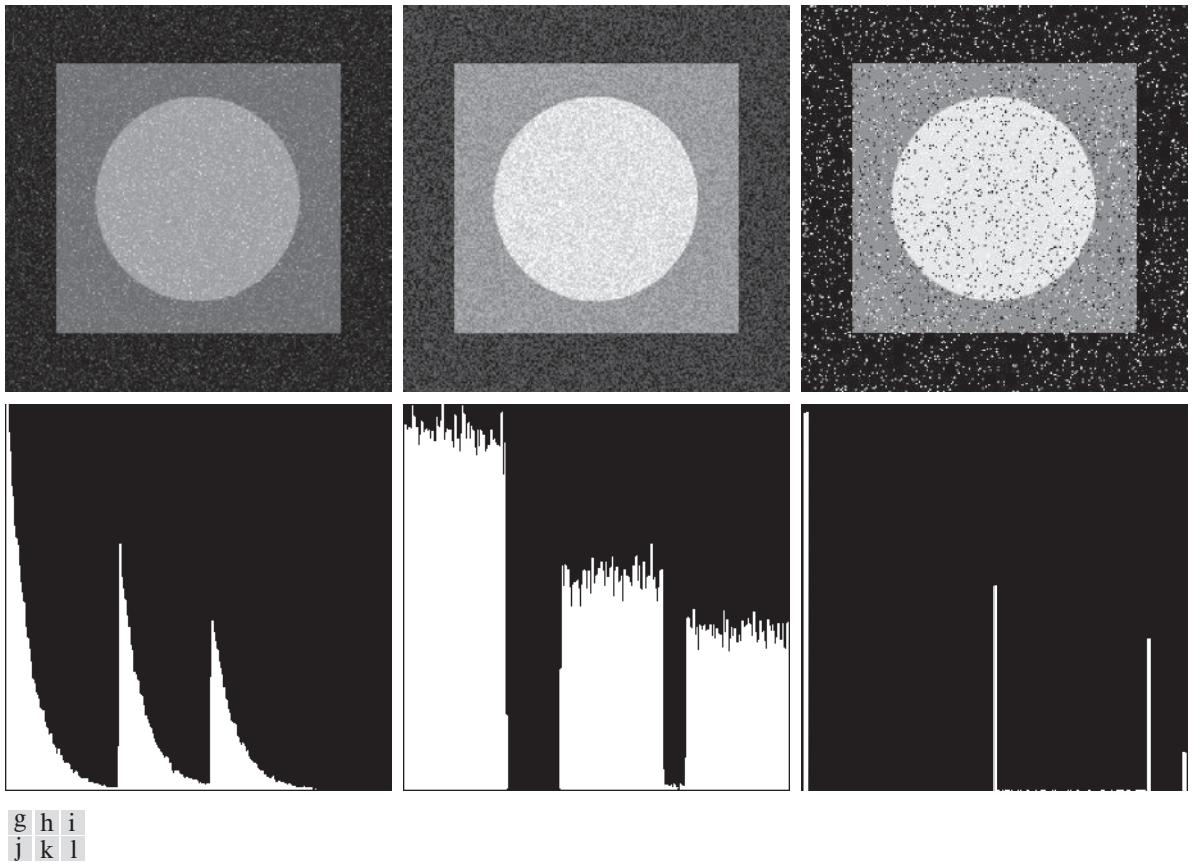


FIGURE 5.4 (continued) Images and histograms resulting from adding exponential, uniform, and salt-and-pepper noise to the image in Fig. 5.3. In the salt-and-pepper histogram, the peaks in the origin (zero intensity) and at the far end of the scale are shown displaced slightly so that they do not blend with the page background.

the conjugate frequencies of the sine wave (see Table 4.4). Thus, if the amplitude of a sine wave in the spatial domain is strong enough, we would expect to see in the spectrum of the image a pair of impulses for each sine wave in the image. As shown in Fig. 5.5(b), this is indeed the case. Eliminating or reducing these impulses in the frequency domain will eliminate or reduce the sinusoidal noise in the spatial domain. We will have much more to say in Section 5.4 about this and other examples of periodic noise.

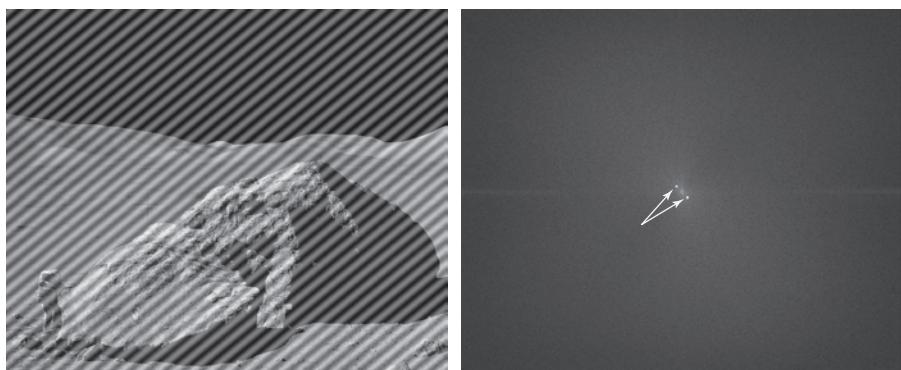
ESTIMATING NOISE PARAMETERS

The parameters of periodic noise typically are estimated by inspection of the Fourier spectrum. Periodic noise tends to produce frequency spikes that often can be detected even by visual analysis. Another approach is to attempt to infer the periodicity

a b

FIGURE 5.5

(a) Image corrupted by additive sinusoidal noise.
 (b) Spectrum showing two conjugate impulses caused by the sine wave.
 (Original image courtesy of NASA.)



of noise components directly from the image, but this is possible only in simplistic cases. Automated analysis is possible in situations in which the noise spikes are either exceptionally pronounced, or when knowledge is available about the general location of the frequency components of the interference (see Section 5.4).

The parameters of noise PDFs may be known partially from sensor specifications, but it is often necessary to estimate them for a particular imaging arrangement. If the imaging system is available, one simple way to study the characteristics of system noise is to capture a set of “flat” images. For example, in the case of an optical sensor, this is as simple as imaging a solid gray board that is illuminated uniformly. The resulting images typically are good indicators of system noise.

When only images already generated by a sensor are available, it is often possible to estimate the parameters of the PDF from small patches of reasonably constant background intensity. For example, the vertical strips shown in Fig. 5.6 were cropped from the Gaussian, Rayleigh, and uniform images in Fig. 5.4. The histograms shown were calculated using image data from these small strips. The histograms in Fig. 5.4 that correspond to the histograms in Fig. 5.6 are the ones in the middle of the group of three in Figs. 5.4(d), (e), and (k). We see that the shapes of these histograms correspond quite closely to the shapes of the corresponding histograms in Fig. 5.6. Their heights are different due to scaling, but the shapes are unmistakably similar.

The simplest use of the data from the image strips is for calculating the mean and variance of intensity levels. Consider a strip (subimage) denoted by S , and let $p_S(z_i)$, $i = 0, 1, 2, \dots, L - 1$, denote the probability estimates (normalized histogram values) of the intensities of the pixels in S , where L is the number of possible intensities in the entire image (e.g., 256 for an 8-bit image). As in Eqs. (2-69) and (2-70), we estimate the mean and variance of the pixel values in S as follows:

$$\bar{z} = \sum_{i=0}^{L-1} z_i p_S(z_i) \quad (5-19)$$

and

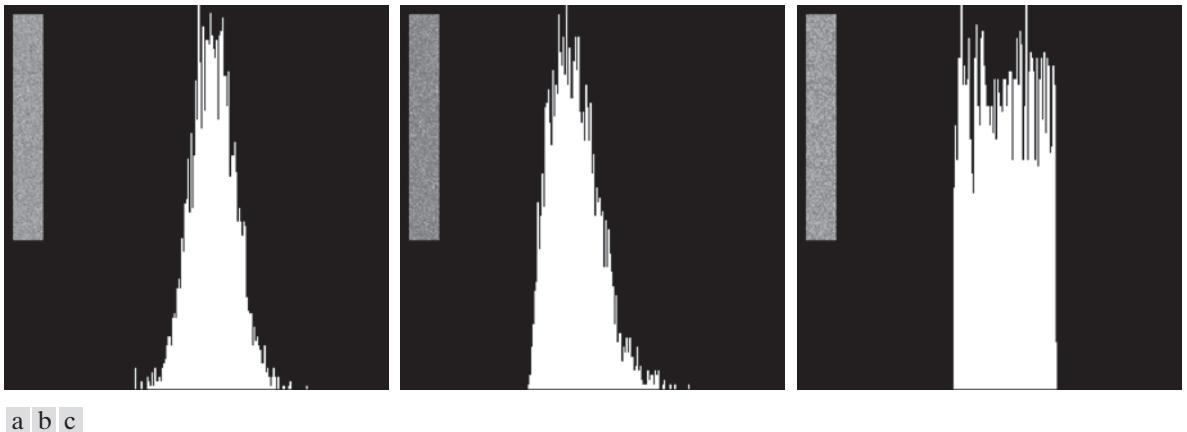


FIGURE 5.6 Histograms computed using small strips (shown as inserts) from (a) the Gaussian, (b) the Rayleigh, and (c) the uniform noisy images in Fig. 5.4.

$$\sigma^2 = \sum_{i=0}^{L-1} (z_i - \bar{z})^2 p_S(z_i) \quad (5-20)$$

The shape of the histogram identifies the closest PDF match. If the shape is approximately Gaussian, then the mean and variance are all we need because the Gaussian PDF is specified completely by these two parameters. For the other shapes discussed earlier, we use the mean and variance to solve for the parameters a and b . Impulse noise is handled differently because the estimate needed is of the actual probability of occurrence of white and black pixels. Obtaining this estimate requires that both black and white pixels be visible, so a mid-gray, relatively constant area is needed in the image in order to be able to compute a meaningful histogram of the noise. The heights of the peaks corresponding to black and white pixels are the estimates of P_a and P_b in Eq. (5-16).

5.3 RESTORATION IN THE PRESENCE OF NOISE ONLY—SPATIAL FILTERING

When an image is degraded only by additive noise, Eqs. (5-1) and (5-2) become

$$g(x,y) = f(x,y) + \eta(x,y) \quad (5-21)$$

and

$$G(u,v) = F(u,v) + N(u,v) \quad (5-22)$$

The noise terms generally are unknown, so subtracting them from $g(x,y)$ [$G(u,v)$] to obtain $f(x,y)$ [$F(u,v)$] typically is not an option. In the case of periodic noise,

sometimes it is possible to estimate $N(u, v)$ from the spectrum of $G(u, v)$, as noted in Section 5.2. In this case $N(u, v)$ can be subtracted from $G(u, v)$ to obtain an estimate of the original image, but this type of knowledge is the exception, rather than the rule.

Spatial filtering is the method of choice for estimating $f(x, y)$ [i.e., *denoising* image $g(x, y)$] in situations when only additive random noise is present. Spatial filtering was discussed in detail in Chapter 3. With the exception of the nature of the computation performed by a specific filter, the mechanics for implementing all the filters that follow are exactly as discussed in Sections 3.4 through 3.7.

MEAN FILTERS

In this section, we discuss briefly the noise-reduction capabilities of the spatial filters introduced in Section 3.5 and develop several other filters whose performance is in many cases superior to the filters discussed in that section.

Arithmetic Mean Filter

We assume that m and n are odd integers. The size of a mean filter is the same as the size of neighborhood S_{xy} ; that is, $m \times n$.

The *arithmetic mean filter* is the simplest of the mean filters (the arithmetic mean filter is the same as the box filter we discussed in Chapter 3). Let S_{xy} represent the set of coordinates in a rectangular subimage window (neighborhood) of size $m \times n$, centered on point (x, y) . The arithmetic mean filter computes the average value of the corrupted image, $g(x, y)$, in the area defined by S_{xy} . The value of the restored image \hat{f} at point (x, y) is the arithmetic mean computed using the pixels in the region defined by S_{xy} . In other words,

$$\hat{f}(x, y) = \frac{1}{mn} \sum_{(r, c) \in S_{xy}} g(r, c) \quad (5-23)$$

where, as in Eq. (2-43), r and c are the row and column coordinates of the pixels contained in the neighborhood S_{xy} . This operation can be implemented using a spatial kernel of size $m \times n$ in which all coefficients have value $1/mn$. A mean filter smooths local variations in an image, and noise is reduced as a result of blurring.

Geometric Mean Filter

An image restored using a *geometric mean filter* is given by the expression

$$\hat{f}(x, y) = \left[\prod_{(r, c) \in S_{xy}} g(r, c) \right]^{\frac{1}{mn}} \quad (5-24)$$

where Π indicates multiplication. Here, each restored pixel is given by the product of all the pixels in the subimage area, raised to the power $1/mn$. As Example 5.2 below illustrates, a geometric mean filter achieves smoothing comparable to an arithmetic mean filter, but it tends to lose less image detail in the process.

Harmonic Mean Filter

The *harmonic mean* filtering operation is given by the expression

$$\hat{f}(x, y) = \frac{mn}{\sum_{(r,c) \in S_{xy}} \frac{1}{g(r, c)}} \quad (5-25)$$

The harmonic mean filter works well for salt noise, but fails for pepper noise. It does well also with other types of noise like Gaussian noise.

Contraharmonic Mean Filter

The *contraharmonic mean filter* yields a restored image based on the expression

$$\hat{f}(x, y) = \frac{\sum_{(r,c) \in S_{xy}} g(r, c)^{Q+1}}{\sum_{(r,c) \in S_{xy}} g(r, c)^Q} \quad (5-26)$$

where Q is called the *order* of the filter. This filter is well suited for reducing or virtually eliminating the effects of salt-and-pepper noise. For positive values of Q , the filter eliminates pepper noise. For negative values of Q , it eliminates salt noise. It cannot do both simultaneously. Note that the contraharmonic filter reduces to the arithmetic mean filter if $Q = 0$, and to the harmonic mean filter if $Q = -1$.

EXAMPLE 5.2: Image denoising using spatial mean filters.

Figure 5.7(a) shows an 8-bit X-ray image of a circuit board, and Fig. 5.7(b) shows the same image, but corrupted with additive Gaussian noise of zero mean and variance of 400. For this type of image, this is a significant level of noise. Figures 5.7(c) and (d) show, respectively, the result of filtering the noisy image with an arithmetic mean filter of size 3×3 and a geometric mean filter of the same size. Although both filters did a reasonable job of attenuating the contribution due to noise, the geometric mean filter did not blur the image as much as the arithmetic filter. For instance, the connector fingers at the top of the image are sharper in Fig. 5.7(d) than in (c). The same is true in other parts of the image.

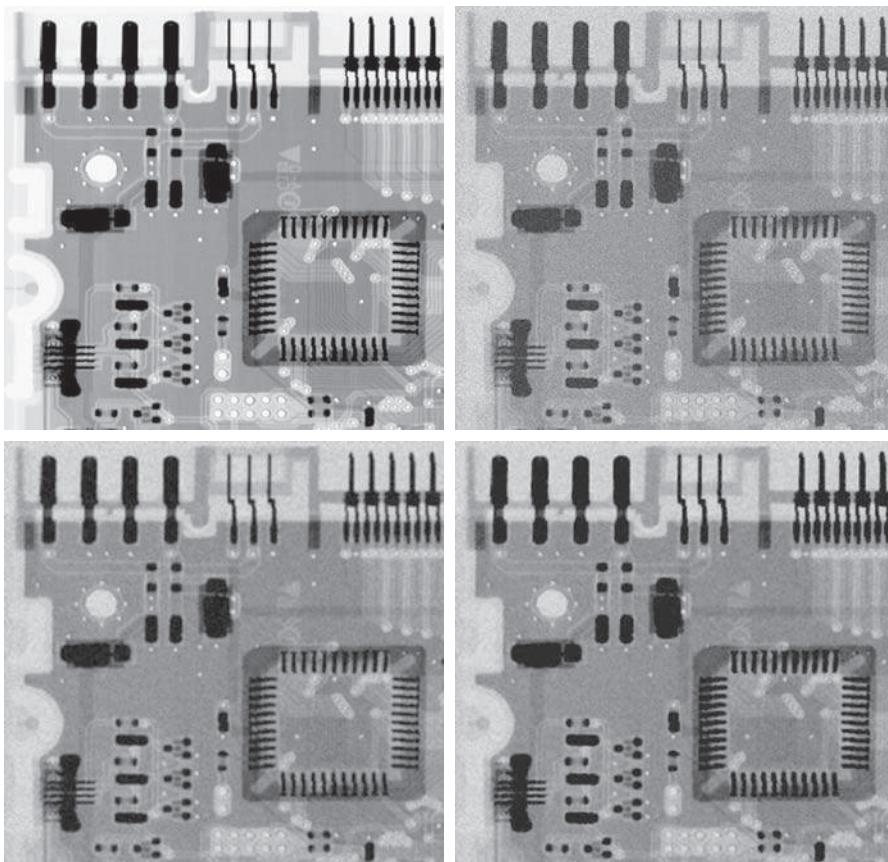
Figure 5.8(a) shows the same circuit image, but corrupted now by pepper noise with probability of 0.1. Similarly, Fig. 5.8(b) shows the image corrupted by salt noise with the same probability. Figure 5.8(c) shows the result of filtering Fig. 5.8(a) using a contraharmonic mean filter with $Q = 1.5$, and Fig. 5.8(d) shows the result of filtering Fig. 5.8(b) with $Q = -1.5$. Both filters did a good job of reducing the effect of the noise. The positive-order filter did a better job of cleaning the background, at the expense of slightly thinning and blurring the dark areas. The opposite was true of the negative order filter.

In general, the arithmetic and geometric mean filters (particularly the latter) are well suited for random noise like Gaussian or uniform noise. The contraharmonic filter is well suited for impulse noise, but it has the disadvantage that it must be known whether the noise is dark or light in order to select the proper sign for Q . The results of choosing the wrong sign for Q can be disastrous, as Fig. 5.9 shows. Some of the filters discussed in the following sections eliminate this shortcoming.

a	b
c	d

FIGURE 5.7

(a) X-ray image of circuit board.
 (b) Image corrupted by additive Gaussian noise.
 (c) Result of filtering with an arithmetic mean filter of size 3×3 .
 (d) Result of filtering with a geometric mean filter of the same size.
 (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)



ORDER-STATISTIC FILTERS

We introduced order-statistic filters in Section 3.6. We now expand the discussion in that section and introduce some additional order-statistic filters. As noted in Section 3.6, order-statistic filters are spatial filters whose response is based on ordering (ranking) the values of the pixels contained in the neighborhood encompassed by the filter. The ranking result determines the response of the filter.

Median Filter

The best-known order-statistic filter in image processing is the *median filter*, which, as its name implies, replaces the value of a pixel by the median of the intensity levels in a predefined neighborhood of that pixel:

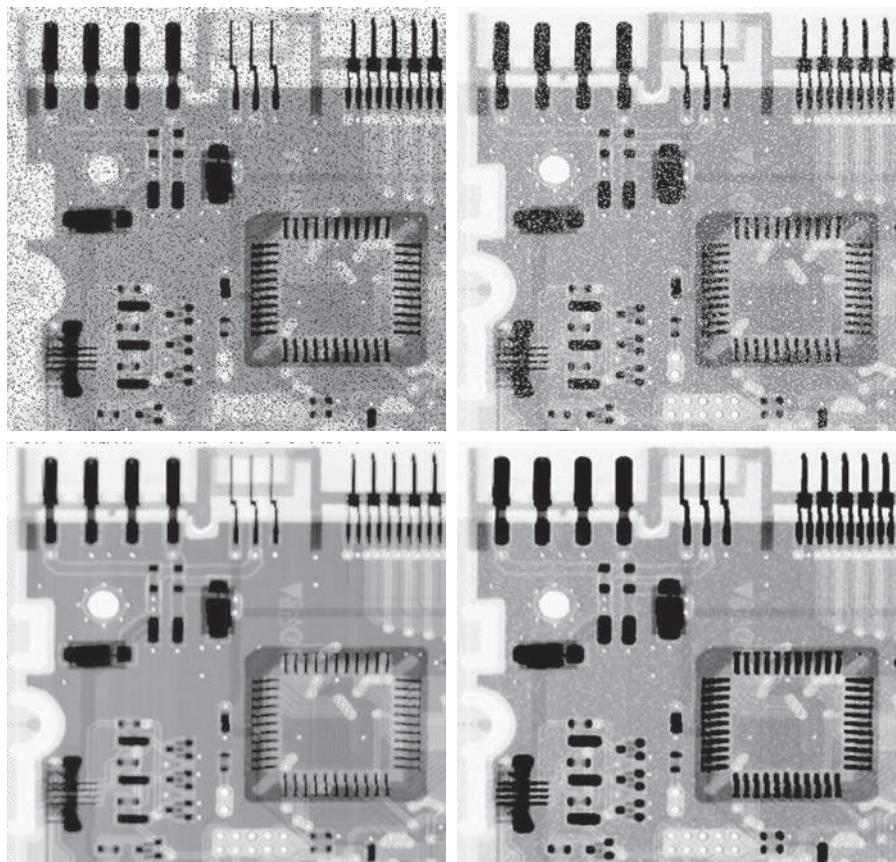
$$\hat{f}(x,y) = \text{median}_{(r,c) \in S_{xy}} \{g(r,c)\} \quad (5-27)$$

where, as before, S_{xy} is a subimage (neighborhood) centered on point (x,y) . The value of the pixel at (x,y) is included in the computation of the median. Median filters

a b
c d

FIGURE 5.8

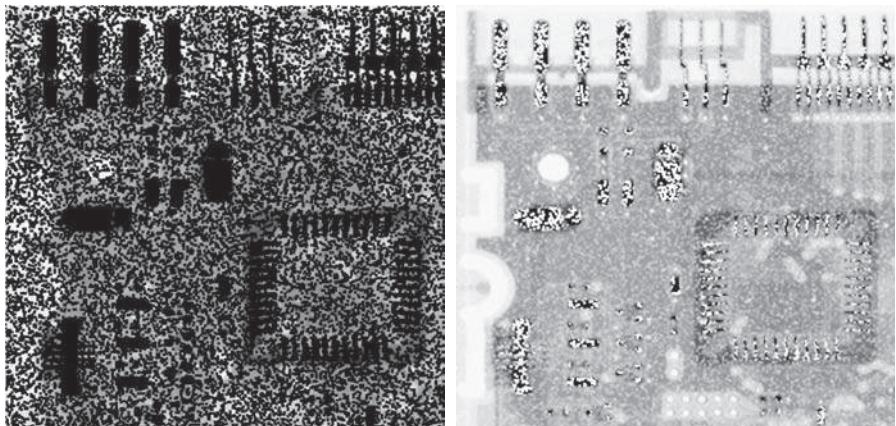
- (a) Image corrupted by pepper noise with a probability of 0.1. (b) Image corrupted by salt noise with the same probability. (c) Result of filtering (a) with a 3×3 contra-harmonic filter $Q = 1.5$. (d) Result of filtering (b) with $Q = -1.5$.



a b

FIGURE 5.9

- Results of selecting the wrong sign in contraharmonic filtering. (a) Result of filtering Fig. 5.8(a) with a contraharmonic filter of size 3×3 and $Q = -1.5$. (b) Result of filtering Fig. 5.8(b) using $Q = 1.5$.



are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of both bipolar and unipolar impulse noise, as Example 5.3 below shows. Computation of the median and implementation of this filter are discussed in Section 3.6.

Max and Min Filters

Although the median filter is by far the order-statistic filter most used in image processing, it is by no means the only one. The median represents the 50th percentile of a ranked set of numbers, but you will recall from basic statistics that ranking lends itself to many other possibilities. For example, using the 100th percentile results in the so-called *max filter*, given by

$$\hat{f}(x,y) = \max_{(r,c) \in S_{xy}} \{g(r,c)\} \quad (5-28)$$

This filter is useful for finding the brightest points in an image or for eroding dark regions adjacent to bright areas. Also, because pepper noise has very low values, it is reduced by this filter as a result of the max selection process in the subimage area S_{xy} .

The 0th percentile filter is the *min filter*:

$$\hat{f}(x,y) = \min_{(r,c) \in S_{xy}} \{g(r,c)\} \quad (5-29)$$

This filter is useful for finding the darkest points in an image or for eroding light regions adjacent to dark areas. Also, it reduces salt noise as a result of the min operation.

Midpoint Filter

The *midpoint filter* computes the midpoint between the maximum and minimum values in the area encompassed by the filter:

$$\hat{f}(x,y) = \frac{1}{2} \left[\max_{(r,c) \in S_{xy}} \{g(r,c)\} + \min_{(r,c) \in S_{xy}} \{g(r,c)\} \right] \quad (5-30)$$

Note that this filter combines order statistics and averaging. It works best for randomly distributed noise, like Gaussian or uniform noise.

Alpha-Trimmed Mean Filter

Suppose that we delete the $d/2$ lowest and the $d/2$ highest intensity values of $g(r,c)$ in the neighborhood S_{xy} . Let $g_R(r,c)$ represent the remaining $mn - d$ pixels in S_{xy} . A filter formed by averaging these remaining pixels is called an *alpha-trimmed mean filter*. The form of this filter is

$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(r,c) \in S_{xy}} g_R(r, c) \quad (5-31)$$

where the value of d can range from 0 to $mn - 1$. When $d = 0$ the alpha-trimmed filter reduces to the arithmetic mean filter discussed earlier. If we choose $d = mn - 1$, the filter becomes a median filter. For other values of d , the alpha-trimmed filter is useful in situations involving multiple types of noise, such as a combination of salt-and-pepper and Gaussian noise.

EXAMPLE 5.3: Image denoising using order-statistic filters.

Figure 5.10(a) shows the circuit board image corrupted by salt-and-pepper noise with probabilities $P_s = P_p = 0.1$. Figure 5.10(b) shows the result of median filtering with a filter of size 3×3 . The improvement over Fig. 5.10(a) is significant, but several noise points still are visible. A second pass [on the image in Fig. 5.10(b)] with the median filter removed most of these points, leaving only few, barely visible noise points. These were removed with a third pass of the filter. These results are good examples of the power of median filtering in handling impulse-like additive noise. Keep in mind that repeated passes of a median filter will blur the image, so it is desirable to keep the number of passes as low as possible.

Figure 5.11(a) shows the result of applying the max filter to the pepper noise image of Fig. 5.8(a). The filter did a reasonable job of removing the pepper noise, but we note that it also removed (set to a light intensity level) some dark pixels from the borders of the dark objects. Figure 5.11(b) shows the result of applying the min filter to the image in Fig. 5.8(b). In this case, the min filter did a better job than the max filter on noise removal, but it removed some white points around the border of light objects. These made the light objects smaller and some of the dark objects larger (like the connector fingers in the top of the image) because white points around these objects were set to a dark level.

The alpha-trimmed filter is illustrated next. Figure 5.12(a) shows the circuit board image corrupted this time by additive, uniform noise of variance 800 and zero mean. This is a high level of noise corruption that is made worse by further addition of salt-and-pepper noise with $P_s = P_p = 0.1$, as Fig. 5.12(b) shows. The high level of noise in this image warrants use of larger filters. Figures 5.12(c) through (f) show the results, respectively, obtained using arithmetic mean, geometric mean, median, and alpha-trimmed mean (with $d = 6$) filters of size 5×5 . As expected, the arithmetic and geometric mean filters (especially the latter) did not do well because of the presence of impulse noise. The median and alpha-trimmed filters performed much better, with the alpha-trimmed filter giving slightly better noise reduction. For example, note in Fig. 5.12(f) that the fourth connector finger from the top left is slightly smoother in the alpha-trimmed result. This is not unexpected because, for a high value of d , the alpha-trimmed filter approaches the performance of the median filter, but still retains some smoothing capabilities.

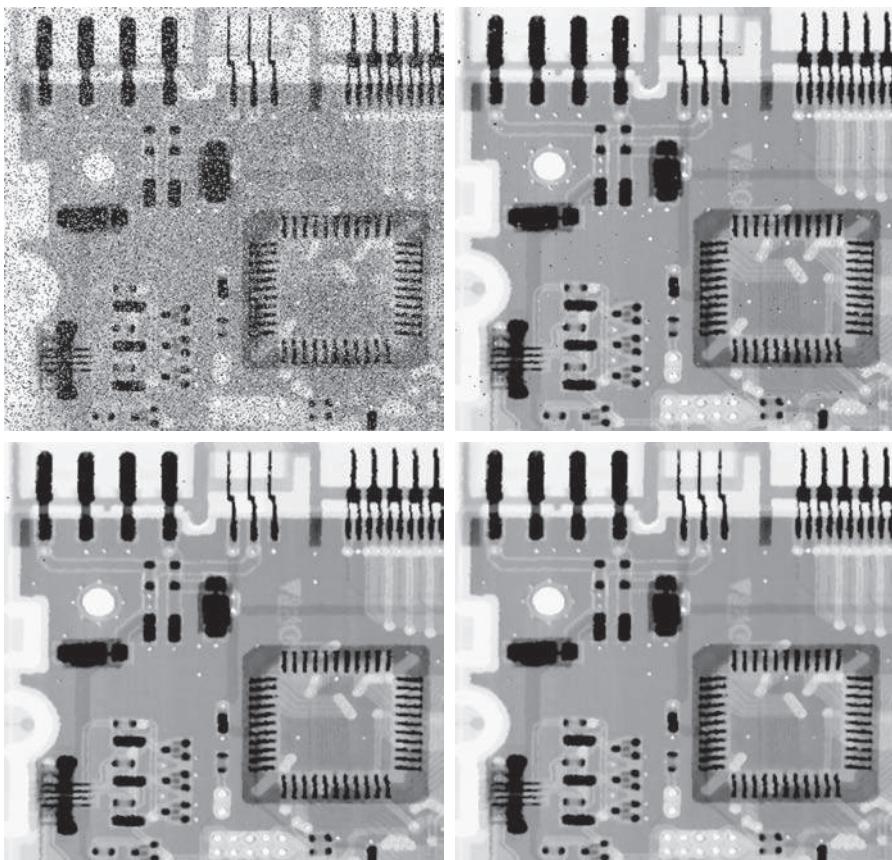
ADAPTIVE FILTERS

Once selected, the filters discussed thus far are applied to an image without regard for how image characteristics vary from one point to another. In this section, we take a look at two *adaptive* filters whose behavior changes based on statistical characteristics of the image inside the filter region defined by the $m \times n$ rectangular neighborhood S_{xy} . As the following discussion shows, adaptive filters are capable of performance superior to that of the filters discussed thus far. The price paid for

a
b
c
d

FIGURE 5.10

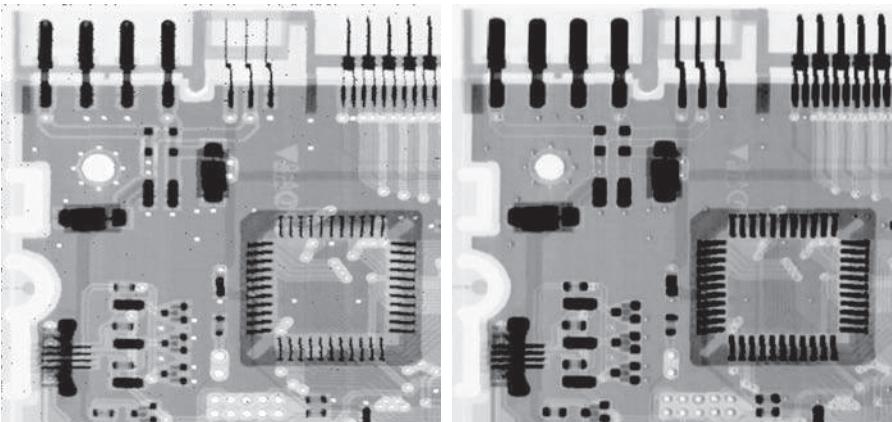
- (a) Image corrupted by salt-and-pepper noise with probabilities $P_s = P_p = 0.1$.
 (b) Result of one pass with a median filter of size 3×3 . (c) Result of processing (b) with this filter.
 (d) Result of processing (c) with the same filter.



a
b

FIGURE 5.11

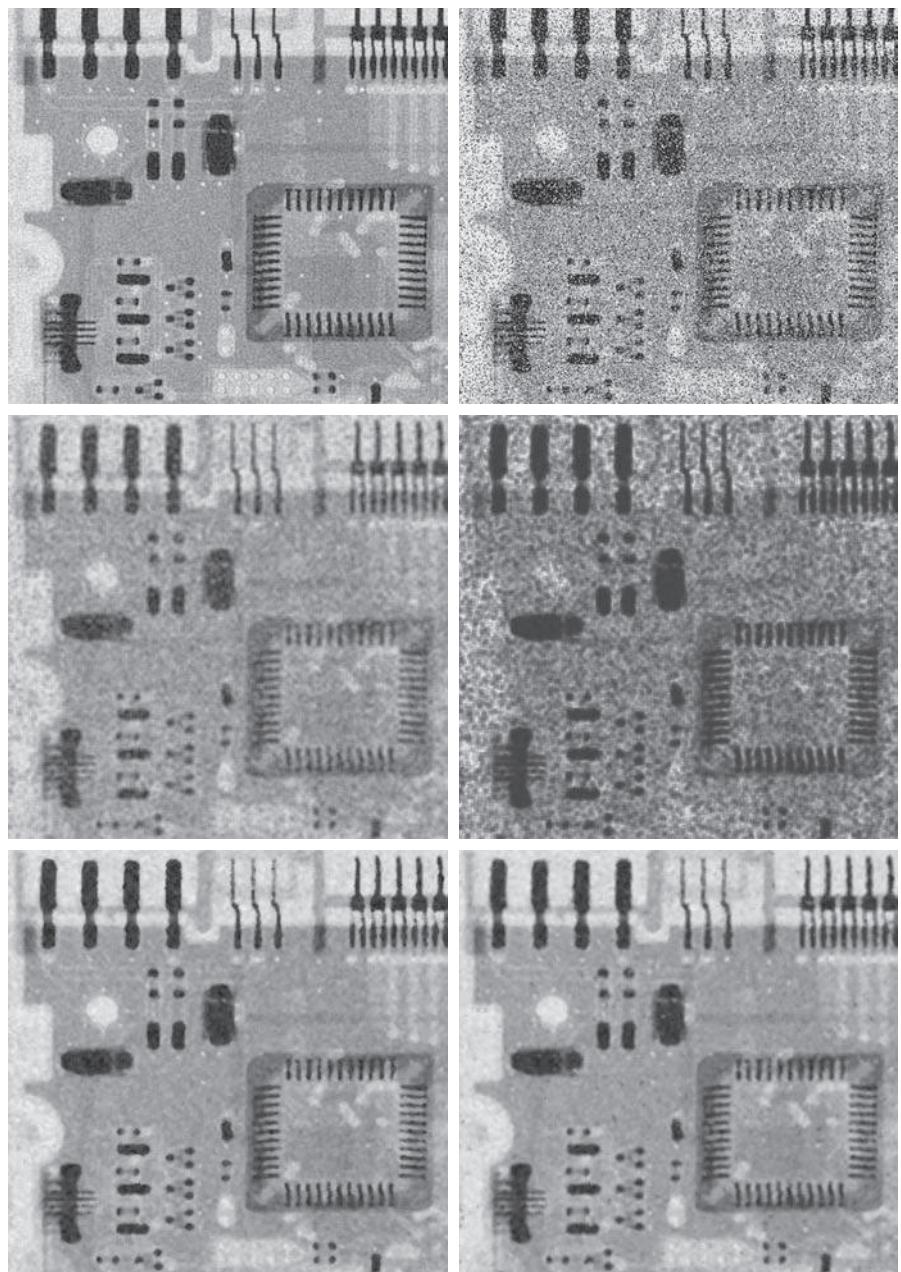
- (a) Result of filtering Fig. 5.8(a) with a max filter of size 3×3 .
 (b) Result of filtering Fig. 5.8(b) with a min filter of the same size.



a b
c d
e f

FIGURE 5.12

- (a) Image corrupted by additive uniform noise. (b) Image additionally corrupted by additive salt-and-pepper noise. (c)-(f) Image (b) filtered with a 5×5 :
- (c) arithmetic mean filter;
 - (d) geometric mean filter;
 - (e) median filter;
 - (f) alpha-trimmed mean filter, with $d = 6$.



improved filtering power is an increase in filter complexity. Keep in mind that we still are dealing with the case in which the degraded image is equal to the original image plus noise. No other types of degradations are being considered yet.

Adaptive, Local Noise Reduction Filter

The simplest statistical measures of a random variable are its mean and variance. These are reasonable parameters on which to base an adaptive filter because they are quantities closely related to the appearance of an image. The mean gives a measure of average intensity in the region over which the mean is computed, and the variance gives a measure of image contrast in that region.

Our filter is to operate on a neighborhood, S_{xy} , centered on coordinates (x, y) . The response of the filter at (x, y) is to be based on the following quantities: $g(x, y)$, the value of the noisy image at (x, y) ; σ_η^2 , the variance of the noise; $\bar{z}_{S_{xy}}$, the local average intensity of the pixels in S_{xy} ; and $\sigma_{S_{xy}}^2$, the local variance of the intensities of pixels in S_{xy} . We want the behavior of the filter to be as follows:

1. If σ_η^2 is zero, the filter should return simply the value of g at (x, y) . This is the trivial, zero-noise case in which g is equal to f at (x, y) .
2. If the local variance $\sigma_{S_{xy}}^2$ is high relative to σ_η^2 , the filter should return a value close to g at (x, y) . A high local variance typically is associated with edges, and these should be preserved.
3. If the two variances are equal, we want the filter to return the arithmetic mean value of the pixels in S_{xy} . This condition occurs when the local area has the same properties as the overall image, and local noise is to be reduced by averaging.

An adaptive expression for obtaining $\hat{f}(x, y)$ based on these assumptions may be written as

$$\hat{f}(x, y) = g(x, y) - \frac{\sigma_\eta^2}{\sigma_{S_{xy}}^2} [g(x, y) - \bar{z}_{S_{xy}}] \quad (5-32)$$

The only quantity that needs to be known a priori is σ_η^2 , the variance of the noise corrupting image $f(x, y)$. This is a constant that can be estimated from sample noisy images using Eq. (3-26). The other parameters are computed from the pixels in neighborhood S_{xy} using Eqs. (3-27) and (3-28).

An assumption in Eq. (5-32) is that the ratio of the two variances does not exceed 1, which implies that $\sigma_\eta^2 \leq \sigma_{S_{xy}}^2$. The noise in our model is additive and position independent, so this is a reasonable assumption to make because S_{xy} is a subset of $g(x, y)$. However, we seldom have exact knowledge of σ_η^2 . Therefore, it is possible for this condition to be violated in practice. For that reason, a test should be built into an implementation of Eq. (5-32) so that the ratio is set to 1 if the condition $\sigma_\eta^2 > \sigma_{S_{xy}}^2$ occurs. This makes this filter nonlinear. However, it prevents nonsensical results (i.e., negative intensity levels, depending on the value of $\bar{z}_{S_{xy}}$) due to a potential lack of knowledge about the variance of the image noise. Another approach is to allow the negative values to occur, and then rescale the intensity values at the end. The result then would be a loss of dynamic range in the image.

EXAMPLE 5.4: Image denoising using adaptive, local noise-reduction filtering.

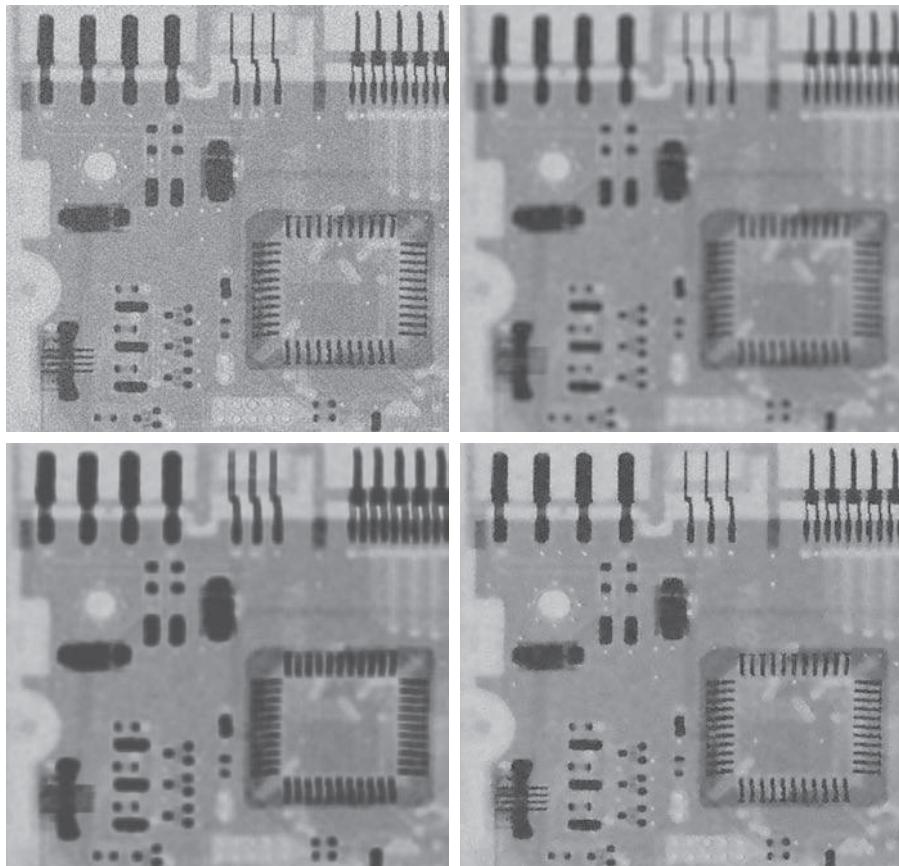
Figure 5.13(a) shows the circuit-board image, corrupted this time by additive Gaussian noise of zero mean and a variance of 1000. This is a significant level of noise corruption, but it makes an ideal test bed on which to compare relative filter performance. Figure 5.13(b) is the result of processing the noisy image with an arithmetic mean filter of size 7×7 . The noise was smoothed out, but at the cost of significant blurring. Similar comments apply to Fig. 5.13(c), which shows the result of processing the noisy image with a geometric mean filter, also of size 7×7 . The differences between these two filtered images are analogous to those we discussed in Example 5.2; only the degree of blurring is different.

Figure 5.13(d) shows the result of using the adaptive filter of Eq. (5-32) with $\sigma_\eta^2 = 1000$. The improvements in this result compared with the two previous filters are significant. In terms of overall noise reduction, the adaptive filter achieved results similar to the arithmetic and geometric mean filters. However, the image filtered with the adaptive filter is much sharper. For example, the connector fingers at the top of the image are significantly sharper in Fig. 5.13(d). Other features, such as holes and the eight legs of the dark component on the lower left-hand side of the image, are much clearer in Fig. 5.13(d). These results are typical of what can be achieved with an adaptive filter. As mentioned earlier, the price paid for the improved performance is additional filter complexity.

a b
c d

FIGURE 5.13

- (a) Image corrupted by additive Gaussian noise of zero mean and a variance of 1000.
- (b) Result of arithmetic mean filtering.
- (c) Result of geometric mean filtering.
- (d) Result of adaptive noise-reduction filtering. All filters used were of size 7×7 .



The preceding results used a value for σ_η^2 that matched the variance of the noise exactly. If this quantity is not known, and the estimate used is too low, the algorithm will return an image that closely resembles the original because the corrections will be smaller than they should be. Estimates that are too high will cause the ratio of the variances to be clipped at 1.0, and the algorithm will subtract the mean from the image more frequently than it would normally. If negative values are allowed and the image is rescaled at the end, the result will be a loss of dynamic range, as mentioned previously.

Adaptive Median Filter

The median filter in Eq. (5-27) performs well if the spatial density of the salt-and-pepper noise is low (as a rule of thumb, P_s and P_p less than 0.2). We show in the following discussion that adaptive median filtering can handle noise with probabilities larger than these. An additional benefit of the adaptive median filter is that it seeks to preserve detail while simultaneously smoothing non-impulse noise, something that the “traditional” median filter does not do. As in all the filters discussed in the preceding sections, the adaptive median filter also works in a rectangular neighborhood, S_{xy} . Unlike those filters, however, the adaptive median filter changes (increases) the size of S_{xy} during filtering, depending on certain conditions to be listed shortly. Keep in mind that the output of the filter is a single value used to replace the value of the pixel at (x, y) , the point on which region S_{xy} is centered at a given time.

We use the following notation:

- z_{\min} = minimum intensity value in S_{xy}
- z_{\max} = maximum intensity value in S_{xy}
- z_{med} = median of intensity values in S_{xy}
- z_{xy} = intensity at coordinates (x, y)
- S_{\max} = maximum allowed size of S_{xy}

The adaptive median-filtering algorithm uses two processing levels, denoted level A and level B , at each point (x, y) :

- | | |
|-------------|---|
| Level A : | If $z_{\min} < z_{\text{med}} < z_{\max}$, go to Level B |
| | Else, increase the size of S_{xy} |
| | If $S_{xy} \leq S_{\max}$, repeat level A |
| | Else, output z_{med} . |
| Level B : | If $z_{\min} < z_{xy} < z_{\max}$, output z_{xy} |
| | Else output z_{med} . |

where S_{xy} and S_{\max} are odd, positive integers greater than 1. Another option in the last step of level A is to output z_{xy} instead of z_{med} . This produces a slightly less blurred result, but can fail to detect salt (pepper) noise embedded in a constant background having the same value as pepper (salt) noise.

This algorithm has three principal objectives: to remove salt-and-pepper (impulse) noise, to provide smoothing of other noise that may not be impulsive, and to reduce distortion, such as excessive thinning or thickening of object boundaries. The values z_{\min} and z_{\max} are considered statistically by the algorithm to be “impulse-like” noise components in region S_{xy} , even if these are not the lowest and highest possible pixel values in the image.

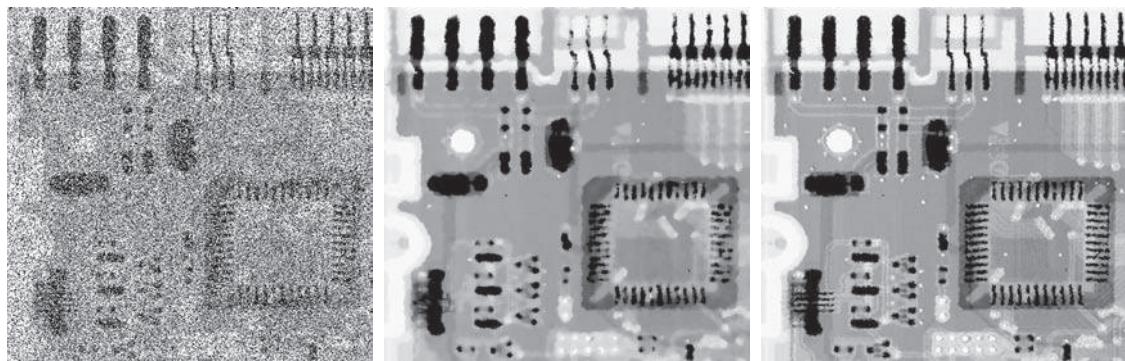
With these observations in mind, we see that the purpose of level *A* is to determine if the median filter output, z_{med} , is an impulse (salt *or* pepper) or not. If the condition $z_{\min} < z_{\text{med}} < z_{\max}$ holds, then z_{med} cannot be an impulse for the reason mentioned in the previous paragraph. In this case, we go to level *B* and test to see if the point in the center of the neighborhood is itself an impulse (recall that (x, y) is the location of the point being processed, and z_{xy} is its intensity). If the condition $z_{\min} < z_{xy} < z_{\max}$ is true, then the pixel at z_{xy} cannot be the intensity of an impulse for the same reason that z_{med} was not. In this case, the algorithm outputs the unchanged pixel value, z_{xy} . By not changing these “intermediate-level” points, distortion is reduced in the filtered image. If the condition $z_{\min} < z_{xy} < z_{\max}$ is false, then either $z_{xy} = z_{\min}$ or $z_{xy} = z_{\max}$. In either case, the value of the pixel is an extreme value and the algorithm outputs the median value, z_{med} , which we know from level *A* is not a noise impulse. The last step is what the standard median filter does. The problem is that the standard median filter replaces every point in the image by the median of the corresponding neighborhood. This causes unnecessary loss of detail.

Continuing with the explanation, suppose that level *A* *does* find an impulse (i.e., it fails the test that would cause it to branch to level *B*). The algorithm then increases the size of the neighborhood and repeats level *A*. This looping continues until the algorithm either finds a median value that is not an impulse (and branches to stage *B*), or the maximum neighborhood size is reached. If the maximum size is reached, the algorithm returns the value of z_{med} . Note that there is no guarantee that this value is not an impulse. The smaller the noise probabilities P_a and/or P_b are, or the larger S_{\max} is allowed to be, the less likely it is that a premature exit will occur. This is plausible. As the density of the noise impulses increases, it stands to reason that we would need a larger window to “clean up” the noise spikes.

Every time the algorithm outputs a value, the center of neighborhood S_{xy} is moved to the next location in the image. The algorithm then is reinitialized and applied to the pixels in the new region encompassed by the neighborhood. As indicated in Problem 3.37, the median value can be updated iteratively from one location to the next, thus reducing computational load.

EXAMPLE 5.5: Image denoising using adaptive median filtering.

Figure 5.14(a) shows the circuit-board image corrupted by salt-and-pepper noise with probabilities $P_s = P_p = 0.25$, which is 2.5 times the noise level used in Fig. 5.10(a). Here the noise level is high enough to obscure most of the detail in the image. As a basis for comparison, the image was filtered first using a 7×7 median filter, the smallest filter required to remove most visible traces of impulse noise in this case. Figure 5.14(b) shows the result. Although the noise was effectively removed, the filter caused significant



a b c

FIGURE 5.14 (a) Image corrupted by salt-and-pepper noise with probabilities $P_s = P_p = 0.25$. (b) Result of filtering with a 7×7 median filter. (c) Result of adaptive median filtering with $S_{\max} = 7$.

loss of detail in the image. For instance, some of the connector fingers at the top of the image appear distorted or broken. Other image details are similarly distorted.

Figure 5.14(c) shows the result of using the adaptive median filter with $S_{\max} = 7$. Noise removal performance was similar to the median filter. However, the adaptive filter did a much better job of preserving sharpness and detail. The connector fingers are less distorted, and some other features that were either obscured or distorted beyond recognition by the median filter appear sharper and better defined in Fig. 5.14(c). Two notable examples are the feed-through small white holes throughout the board, and the dark component with eight legs in the bottom, left quadrant of the image.

Considering the high level of noise in Fig. 5.14(a), the adaptive algorithm performed quite well. The choice of maximum allowed size for S_{xy} depends on the application, but a reasonable starting value can be estimated by experimenting with various sizes of the standard median filter first. This will establish a visual baseline regarding expectations on the performance of the adaptive algorithm.

5.4 PERIODIC NOISE REDUCTION USING FREQUENCY DOMAIN FILTERING

Periodic noise can be analyzed and filtered quite effectively using frequency domain techniques. The basic idea is that periodic noise appears as concentrated bursts of energy in the Fourier transform, at locations corresponding to the frequencies of the periodic interference. The approach is to use a selective filter (see Section 4.10) to isolate the noise. The three types of selective filters (bandreject, bandpass, and notch) were discussed in detail in Section 4.10. There is no difference between how these filters were used in Chapter 4, and the way they are used for image restoration. In restoration of images corrupted by periodic interference, the tool of choice is a notch filter. In the following discussion we will expand on the notch filtering approach introduced in Section 4.10, and also develop a more powerful optimum notch filtering method.

MORE ON NOTCH FILTERING

As explained in Section 4.10, notch reject filter transfer functions are constructed as products of highpass filter transfer functions whose centers have been translated to the centers of the notches. The general form of a notch filter transfer function is

$$H_{\text{NR}}(u, v) = \prod_{k=1}^Q H_k(u, v) H_{-k}(u, v) \quad (5-33)$$

where $H_k(u, v)$ and $H_{-k}(u, v)$ are highpass filter transfer functions whose centers are at (u_k, v_k) and $(-u_k, -v_k)$, respectively.[†] These centers are specified with respect to the center of the frequency rectangle, $[\text{floor}(M/2), \text{floor}(N/2)]$, where, as usual, M and N are the number of rows and columns in the input image. Thus, the distance computations for the filter transfer functions are given by

$$D_k(u, v) = [(u - M/2 - u_k)^2 + (v - N/2 - v_k)^2]^{1/2} \quad (5-34)$$

and

$$D_{-k}(u, v) = [(u - M/2 + u_k)^2 + (v - N/2 + v_k)^2]^{1/2} \quad (5-35)$$

For example, the following is a Butterworth notch reject filter transfer function of order n with three notch pairs:

$$H_{\text{NR}}(u, v) = \prod_{k=1}^3 \left[\frac{1}{1 + [D_{0k}/D_k(u, v)]^n} \right] \left[\frac{1}{1 + [D_{0k}/D_{-k}(u, v)]^n} \right] \quad (5-36)$$

Because notches are specified as symmetric pairs, the constant D_{0k} is the same for each pair. However, this constant can be different from one pair to another. Other notch reject filter functions are constructed in the same manner, depending on the highpass filter function chosen. As explained in Section 4.10, a notch pass filter transfer function is obtained from a notch reject function using the expression

$$H_{\text{NP}}(u, v) = 1 - H_{\text{NR}}(u, v) \quad (5-37)$$

where $H_{\text{NP}}(u, v)$ is the transfer function of the notch pass filter corresponding to the notch reject filter with transfer function $H_{\text{NR}}(u, v)$. Figure 5.15 shows perspective plots of the transfer functions of ideal, Gaussian, and Butterworth notch reject filters with one notch pair. As we discussed in Chapter 4, we see again that the shape of the Butterworth transfer function represents a transition between the sharpness of the ideal function and the broad, smooth shape of the Gaussian transfer function.

As we show in the second part of the following example, we are not limited to notch filter transfer functions of the form just discussed. We can construct notch

[†] Remember, frequency domain transfer functions are symmetric about the center of the frequency rectangle, so the notches are specified as symmetric pairs. Also, recall from Section 4.10 that we use unpadded images when working with notch filters in order to simplify the specification of notch locations.

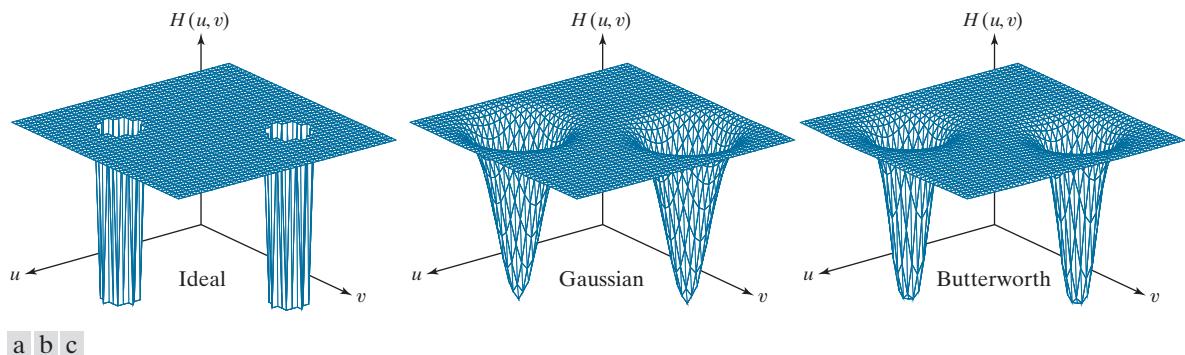


FIGURE 5.15 Perspective plots of (a) ideal, (b) Gaussian, and (c) Butterworth notch reject filter transfer functions.

filters of arbitrary shapes, provided that they are zero-phase-shift functions, as defined in Section 4.7.

EXAMPLE 5.6: Image denoising (interference reduction) using notch filtering.

Figure 5.16(a) is the same as Fig. 2.45(a), which we used in Section 2.6 to introduce the concept of filtering in the frequency domain. We now look in more detail at the process of denoising this image, which is corrupted by a single, 2-D additive sine wave. You know from Table 4.4 that the Fourier transform of a pure sine wave is a pair of complex, conjugate impulses, so we would expect the spectrum to have a pair of bright dots at the frequencies of the sine wave. As Fig. 5.16(b) shows, this is indeed the case. Because we can determine the location of these impulses accurately, eliminating them is a simple task, consisting of using a notch filter transfer function whose notches coincide with the location of the impulses.

Figure 5.16(c) shows an ideal notch reject filter transfer function, which is an array of 1's (shown in white) and two small circular regions of 0's (shown in black). Figure 5.16(d) shows the result of filtering the noisy image this transfer function. The sinusoidal noise was virtually eliminated, and a number of details that were previously obscured by the interference are clearly visible in the filtered image (see, for example, the thin fiducial marks and the fine detail in the terrain and rock formations). As we showed in Example 4.25, obtaining an image of the interference pattern is straightforward. We simply turn the reject filter into a pass filter by subtracting it from 1, and filter the input image with it. Figure 5.17 shows the result.

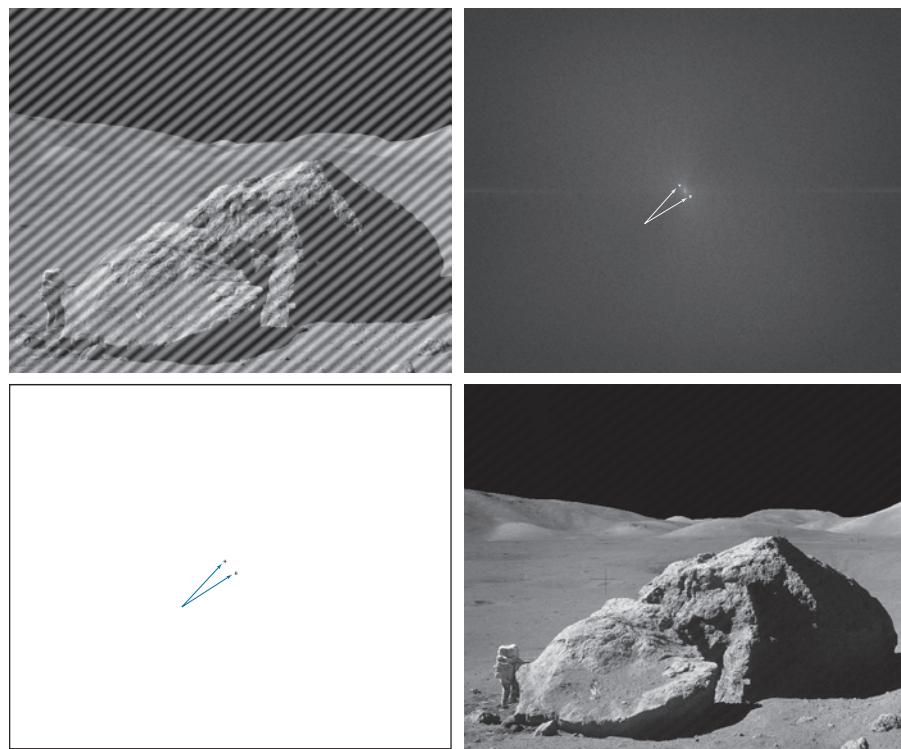
Figure 5.18(a) shows the same image as Fig. 4.50(a), but covering a larger area (the interference pattern is the same). When we discussed lowpass filtering of that image in Chapter 4, we indicated that there were better ways to reduce the effect of the scan lines. The notch filtering approach that follows reduces the scan lines significantly, without introducing blurring. Unless blurring is desirable for reasons we discussed in Section 4.9, notch filtering generally gives much better results.

Just by looking at the nearly horizontal lines of the noise pattern in Fig. 5.18(a), we expect its contribution in the frequency domain to be concentrated along the vertical axis of the DFT. However, the noise is not dominant enough to have a clear pattern along this axis, as is evident in the spectrum shown in Fig. 5.18(b). The approach to follow in cases like this is to use a narrow, rectangular notch filter function that extends along the vertical axis, and thus eliminates all components of the interference along that axis. We do not filter near the origin to avoid eliminating the dc term and low frequencies,

a b
c d

FIGURE 5.16

- (a) Image corrupted by sinusoidal interference.
- (b) Spectrum showing the bursts of energy caused by the interference. (The bursts were enlarged for display purposes.)
- (c) Notch filter (the radius of the circles is 2 pixels) used to eliminate the energy bursts. (The thin borders are not part of the data.)
- (d) Result of notch reject filtering. (Original image courtesy of NASA.)



which, as you know from Chapter 4, are responsible for the intensity differences between smooth areas. Figure 5.18(c) shows the filter transfer function we used, and Fig. 5.18(d) shows the filtered result. Most of the fine scan lines were eliminated or significantly attenuated. In order to get an image of the noise pattern, we proceed as before by converting the reject filter into a pass filter, and then filtering the input image with it. Figure 5.19 shows the result.

FIGURE 5.17

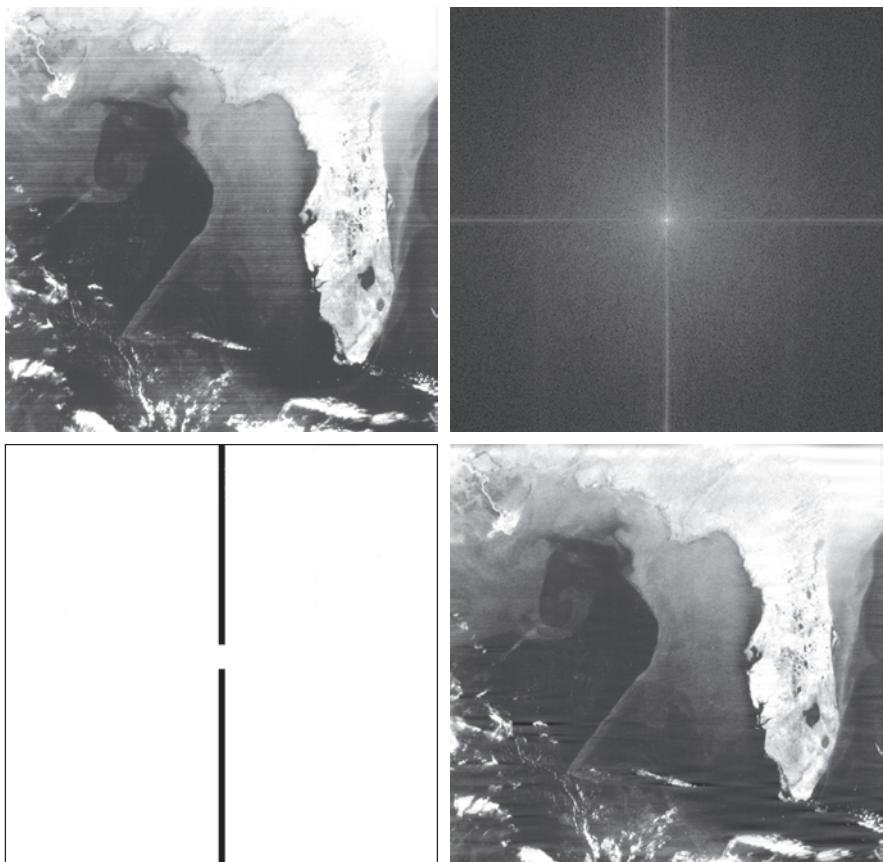
Sinusoidal pattern extracted from the DFT of Fig. 5.16(a) using a notch pass filter.



a	b
c	d

FIGURE 5.18

(a) Satellite image of Florida and the Gulf of Mexico. (Note horizontal sensor scan lines.) (b) Spectrum of (a). (c) Notch reject filter transfer function. (The thin black border is not part of the data.) (d) Filtered image. (Original image courtesy of NOAA.)

**FIGURE 5.19**

Noise pattern extracted from Fig. 5.18(a) by notch pass filtering.



OPTIMUM NOTCH FILTERING

In the examples of notch filtering given thus far, the interference patterns have been simple to identify and characterize in the frequency domain, leading to the specification of notch filter transfer functions that also are simple to define heuristically.

When several interference components are present, heuristic specifications of filter transfer functions are not always acceptable because they may remove too much image information in the filtering process (a highly undesirable feature when images are unique and/or expensive to acquire). In addition, the interference components generally are not single-frequency bursts. Instead, they tend to have broad skirts that carry information about the interference pattern. These skirts are not always easily detectable from the normal transform background. Alternative filtering methods that reduce the effect of these degradations are quite useful in practice. The method discussed next is optimum, in the sense that it minimizes local variances of the restored estimate $\hat{f}(x, y)$.

The procedure consists of first isolating the principal contributions of the interference pattern and then subtracting a variable, weighted portion of the pattern from the corrupted image. Although we develop the procedure in the context of a specific application, the basic approach is general and can be applied to other restoration tasks in which multiple periodic interference is a problem.

We begin by extracting the principal frequency components of the interference pattern. As before, we do this by placing a notch pass filter transfer function, $H_{NP}(u, v)$, at the location of each spike. If the filter is constructed to pass only components associated with the interference pattern, then the Fourier transform of the interference noise pattern is given by the expression

$$N(u, v) = H_{NP}(u, v)G(u, v) \quad (5-38)$$

where, as usual, $G(u, v)$ is the DFT of the corrupted image.

Specifying $H_{NP}(u, v)$ requires considerable judgment about what is or is not an interference spike. For this reason, the notch pass filter generally is constructed interactively by observing the spectrum of $G(u, v)$ on a display. After a particular filter function has been selected, the corresponding noise pattern in the spatial domain is obtained using the familiar expression

$$\eta(x, y) = \mathfrak{F}^{-1}\{H_{NP}(u, v)G(u, v)\} \quad (5-39)$$

Because the corrupted image is assumed to be formed by the addition of the uncorrupted image $f(x, y)$ and the interference, $\eta(x, y)$, if the latter were known completely, subtracting the pattern from $g(x, y)$ to obtain $f(x, y)$ would be a simple matter. The problem, of course, is that this filtering procedure usually yields only an approximation of the true noise pattern. The effect of incomplete components not present in the estimate of $\eta(x, y)$ can be minimized by subtracting from $g(x, y)$ a weighted portion of $\eta(x, y)$ to obtain an estimate of $f(x, y)$:

$$\hat{f}(x, y) = g(x, y) - w(x, y)\eta(x, y) \quad (5-40)$$

where, as before, $\hat{f}(x, y)$ is the estimate of $f(x, y)$ and $w(x, y)$ is to be determined. This function is called a *weighting* or *modulation function*, and the objective of the procedure is to select $w(x, y)$ so that the result is optimized in some meaningful way. One approach is to select $w(x, y)$ so that the variance of $\hat{f}(x, y)$ is minimized over a specified neighborhood of every point (x, y) .

Consider a neighborhood S_{xy} of (odd) size $m \times n$, centered on (x, y) . The “local” variance of $\hat{f}(x, y)$ at point (x, y) can be estimated using the samples in S_{xy} , as follows:

$$\sigma^2(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} [\hat{f}(r, c) - \bar{\hat{f}}]^2 \quad (5-41)$$

where $\bar{\hat{f}}$ is the average value of \hat{f} in neighborhood S_{xy} ; that is,

$$\bar{\hat{f}} = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} \hat{f}(r, c) \quad (5-42)$$

Points on or near the edge of the image can be treated by considering partial neighborhoods or by padding the border with 0's.

Substituting Eq. (5-40) into Eq. (5-41) we obtain

$$\sigma^2(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} \left\{ [g(r, c) - w(r, c)\eta(r, c)] - [\bar{g} - \bar{w}\bar{\eta}] \right\}^2 \quad (5-43)$$

where \bar{g} and $\bar{w}\bar{\eta}$ denote the average values of g and of the product $w\eta$ in neighborhood S_{xy} , respectively.

If we assume that w is approximately constant in S_{xy} we can replace $w(r, c)$ by the value of w at the center of the neighborhood:

$$w(r, c) = w(x, y) \quad (5-44)$$

Because $w(x, y)$ is assumed to be constant in S_{xy} , it follows that $\bar{w} = w(x, y)$ and, therefore, that

$$\bar{w}\bar{\eta} = w(x, y)\bar{\eta} \quad (5-45)$$

in S_{xy} , where $\bar{\eta}$ is the average value of η in the neighborhood. Using these approximations, Eq. (5-43) becomes

$$\sigma^2(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} \left\{ [g(r, c) - w(x, y)\eta(r, c)] - [\bar{g} - w(x, y)\bar{\eta}] \right\}^2 \quad (5-46)$$

To minimize $\sigma^2(x, y)$ with respect to $w(x, y)$ we solve

$$\frac{\partial \sigma^2(x, y)}{\partial w(x, y)} = 0 \quad (5-47)$$

for $w(x, y)$. The result is (see Problem 5.17):

$$w(x, y) = \frac{\bar{g\eta} - \bar{g}\bar{\eta}}{\bar{\eta}^2 - \bar{\eta}^2} \quad (5-48)$$

To obtain the value of the restored image at point (x, y) we use this equation to compute $w(x, y)$ and then substitute it into Eq. (5-40). To obtain the complete restored image, we perform this procedure at every point in the noisy image, g .

EXAMPLE 5.7: Denoising (interference removal) using optimum notch filtering.

Figure 5.20(a) shows a digital image of the Martian terrain taken by the Mariner 6 spacecraft. The image is corrupted by a semi-periodic interference pattern that is considerably more complex (and much more subtle) than those we have studied thus far. The Fourier spectrum of the image, shown in Fig. 5.20(b), has a number of “starlike” bursts of energy caused by the interference. As expected, these components are more difficult to detect than those we have seen before. Figure 5.21 shows the spectrum again, but without centering. This image offers a somewhat clearer view of the interference components because the more prominent dc term and low frequencies are “out of way,” in the top left of the spectrum.

Figure 5.22(a) shows the spectrum components that, in the judgement of an experienced image analyst, are associated with the interference. Applying a notch pass filter to these components and using Eq. (5-39) yielded the spatial noise pattern, $\eta(x, y)$, shown in Fig. 5.22(b). Note the similarity between this pattern and the structure of the noise in Fig. 5.20(a).

a b

FIGURE 5.20

(a) Image of the Martian terrain taken by Mariner 6.
 (b) Fourier spectrum showing periodic interference.
 (Courtesy of NASA.)

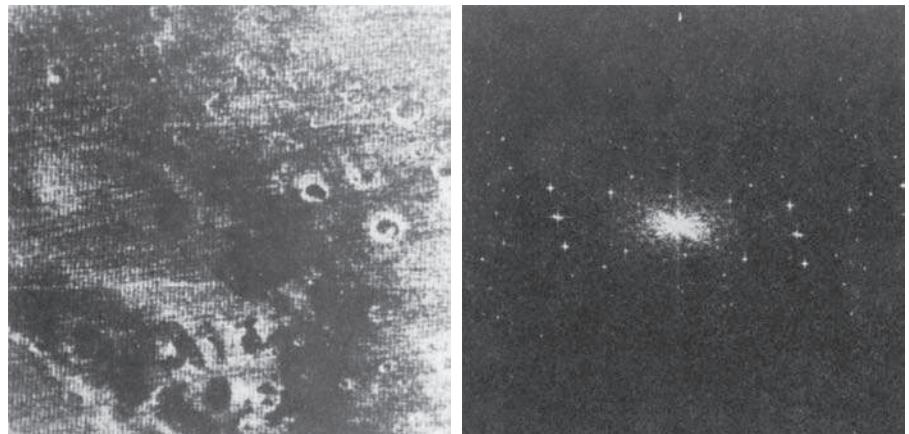
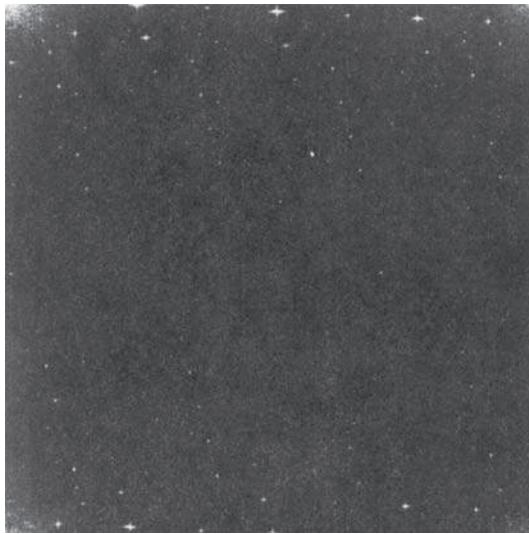


FIGURE 5.21

Uncentered Fourier spectrum of the image in Fig. 5.20(a). (Courtesy of NASA.)



Finally, Fig. 5.23 shows the restored image, obtained using Eq. (5-40) with the interference pattern just discussed. Function $w(x, y)$ was computed using the procedure explained in the preceding paragraphs. As you can see, the periodic interference was virtually eliminated from the noisy image in Fig. 5.20(a).

5.5 LINEAR, POSITION-INVARIANT DEGRADATIONS

The input-output relationship in Fig. 5.1 before the restoration stage is expressed as

$$g(x, y) = \mathcal{H}[f(x, y)] + \eta(x, y) \quad (5-49)$$

For the moment, let us assume that $\eta(x, y) = 0$ so that $g(x, y) = \mathcal{H}[f(x, y)]$. Based on the discussion in Section 2.6, \mathcal{H} is *linear* if

a b

FIGURE 5.22

(a) Fourier spectrum of $N(u, v)$, and
 (b) corresponding spatial noise interference pattern, $\eta(x, y)$. (Courtesy of NASA.)

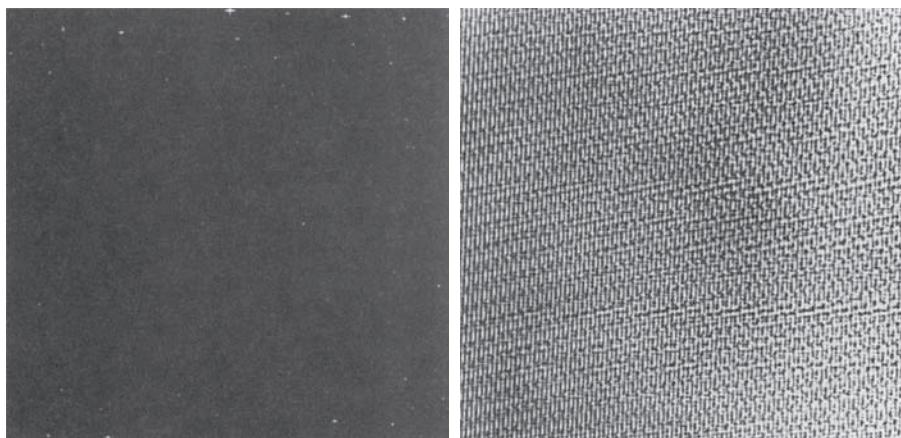


FIGURE 5.23

Restored image.
(Courtesy of
NASA.)



$$\mathcal{H}[af_1(x,y) + bf_2(x,y)] = a\mathcal{H}[f_1(x,y)] + b\mathcal{H}[f_2(x,y)] \quad (5-50)$$

where a and b are scalars and $f_1(x,y)$ and $f_2(x,y)$ are any two input images.

If $a = b = 1$, Eq. (5-50) becomes

$$\mathcal{H}[f_1(x,y) + f_2(x,y)] = \mathcal{H}[f_1(x,y)] + \mathcal{H}[f_2(x,y)] \quad (5-51)$$

which is called the property of *additivity*. This property says that, if \mathcal{H} is a linear operator, the response to a sum of two inputs is equal to the sum of the two responses.

With $f_2(x,y) = 0$, Eq. (5-50) becomes

$$\mathcal{H}[af_1(x,y)] = a\mathcal{H}[f_1(x,y)] \quad (5-52)$$

which is called the property of *homogeneity*. It says that the response to a constant multiple of any input is equal to the response to that input multiplied by the same constant. Thus, a linear operator possesses both the property of additivity and the property of homogeneity.

An operator having the input-output relationship $g(x,y) = \mathcal{H}[f(x,y)]$ is said to be *position* (or *space*) *invariant* if

$$\mathcal{H}[f(x - \alpha, y - \beta)] = g(x - \alpha, y - \beta) \quad (5-53)$$

for any $f(x,y)$ and any two scalars α and β . This definition indicates that the response at any point in the image depends only on the *value* of the input at that point, not on its *position*.

Using the sifting property of the 2-D continuous impulse [see Eq. (4-55)], we can write $f(x,y)$ as

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) \delta(x - \alpha, y - \beta) d\alpha d\beta \quad (5-54)$$

Assuming again that $\eta(x, y) = 0$, substituting this equation into Eq. (5-49) yields

$$g(x, y) = \mathcal{H}[f(x, y)] = \mathcal{H} \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) \delta(x - \alpha, y - \beta) d\alpha d\beta \right] \quad (5-55)$$

If \mathcal{H} is a linear operator and we extend the additivity property to integrals, then

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{H}[f(\alpha, \beta) \delta(x - \alpha, y - \beta)] d\alpha d\beta \quad (5-56)$$

Because $f(\alpha, \beta)$ is independent of x and y , and using the homogeneity property, it follows that

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) \mathcal{H}[\delta(x - \alpha, y - \beta)] d\alpha d\beta \quad (5-57)$$

The term

$$h(x, \alpha, y, \beta) = \mathcal{H}[\delta(x - \alpha, y - \beta)] \quad (5-58)$$

is called the *impulse response* of \mathcal{H} . In other words, if $\eta(x, y) = 0$ in Eq. (5-49), then $h(x, \alpha, y, \beta)$ is the response of \mathcal{H} to an impulse at coordinates (x, y) . In optics, the impulse becomes a point of light and $h(x, \alpha, y, \beta)$ is commonly referred to as the *point spread function* (PSF). This name is based on the fact that all physical optical systems blur (spread) a point of light to some degree, with the amount of blurring being determined by the quality of the optical components.

Substituting Eq. (5-58) into Eq. (5-57) we obtain the expression

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) h(x, \alpha, y, \beta) d\alpha d\beta \quad (5-59)$$

which is called the *superposition* (or *Fredholm*) *integral of the first kind*. This expression is a fundamental result that is at the core of linear system theory. It states that if the response of \mathcal{H} to an impulse is known, the response to *any* input $f(\alpha, \beta)$ can be calculated using Eq. (5-59). In other words, a linear system \mathcal{H} is *characterized completely* by its impulse response.

If \mathcal{H} is position invariant, then it follows from Eq. (5-53) that

$$\mathcal{H}[\delta(x - \alpha, y - \beta)] = h(x - \alpha, y - \beta) \quad (5-60)$$

In this case, Eq. (5-59) reduces to

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) h(x - \alpha, y - \beta) d\alpha d\beta \quad (5-61)$$

This expression is the *convolution integral* introduced for one variable in Eq. (4-24) and extended to 2-D in Problem 4.19. Equation (5-61) tells us that the output of a linear, position invariant system to *any* input, is obtained by convolving the input and the system's impulse response.

In the presence of additive noise, the expression of the linear degradation model [Eq. (5-59)] becomes

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) h(x, \alpha, y, \beta) d\alpha d\beta + \eta(x, y) \quad (5-62)$$

If \mathcal{H} is position invariant, then this equation becomes

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) h(x - \alpha, y - \beta) d\alpha d\beta + \eta(x, y) \quad (5-63)$$

The values of the noise term $\eta(x, y)$ are random, and are assumed to be independent of position. Using the familiar notation for convolution introduced in Chapters 3 and 4, we can write Eq. (5-63) as

$$g(x, y) = (h \star f)(x, y) + \eta(x, y) \quad (5-64)$$

or, using the convolution theorem, we write the equivalent result in the frequency domain as

$$G(u, v) = H(u, v)F(u, v) + N(u, v) \quad (5-65)$$

These two expressions agree with Eqs. (5-1) and (5-2). Keep in mind that, for discrete quantities, all products are elementwise products, as defined in Section 2.6.

In summary, the preceding discussion indicates that a linear, spatially invariant degradation system with additive noise can be modeled in the spatial domain as the convolution of an image with the system's degradation (point spread) function, followed by the addition of noise. Based on the convolution theorem, the same process can be expressed in the frequency domain as the product of the transforms of the image and degradation, followed by the addition of the transform of the noise. When working in the frequency domain, we make use of an FFT algorithm. However, unlike in Chapter 4, we do not use image padding in the implementation of any of the frequency domain restoration filters discussed in this chapter. The reason is that in restoration work we usually have access only to degraded images. For padding to be effective, it would have to be applied to images before they were degraded, a condition that obviously cannot be met in practice. If we had access to the original images, then restoration would be a mute point.

Many types of degradations can be approximated by linear, position-invariant processes. The advantage of this approach is that the extensive tools of linear system theory then become available for the solution of image restoration problems.

Nonlinear and position-dependent techniques, although more general (and usually more accurate), introduce difficulties that often have no known solution or are very difficult to solve computationally. This chapter focuses on linear, space-invariant restoration techniques. Because degradations are modeled as being the result of convolution, and restoration seeks to find filters that apply the process in reverse, the term *image deconvolution* is used frequently to signify linear image restoration. Similarly, the filters used in the restoration process often are called *deconvolution filters*.

5.6 ESTIMATING THE DEGRADATION FUNCTION

There are three principal ways to estimate the degradation function for use in image restoration: (1) observation, (2) experimentation, and (3) mathematical modeling. These methods are discussed in the following sections. The process of restoring an image by using a degradation function that has been estimated by any of these approaches sometimes is called *blind deconvolution*, to emphasize the fact that the true degradation function is seldom known completely.

ESTIMATION BY IMAGE OBSERVATION

Suppose that we are given a degraded image without any knowledge about the degradation function \mathcal{H} . Based on the assumption that the image was degraded by a linear, position-invariant process, one way to estimate \mathcal{H} is to gather information from the image itself. For example, if the image is blurred, we can look at a small rectangular section of the image containing sample structures, like part of an object and the background. In order to reduce the effect of noise, we would look for an area in which the signal content is strong (e.g., an area of high contrast). The next step would be to process the subimage to arrive at a result that is as unblurred as possible.

Let the observed subimage be denoted by $g_s(x, y)$, and let the processed subimage (which in reality is our estimate of the original image in that area) be denoted by $f_s(x, y)$. Then, assuming that the effect of noise is negligible because of our choice of a strong-signal area, it follows from Eq. (5-65) that

$$H_s(u, v) = \frac{G_s(u, v)}{\hat{F}_s(u, v)} \quad (5-66)$$

From the characteristics of this function, we then deduce the complete degradation function $H(u, v)$ based on our assumption of position invariance. For example, suppose that a radial plot of $H_s(u, v)$ has the approximate shape of a Gaussian curve. We can use that information to construct a function $H(u, v)$ on a larger scale, but having the same basic shape. We then use $H(u, v)$ in one of the restoration approaches to be discussed in the following sections. Clearly, this is a laborious process used only in very specific circumstances, such as restoring an old photograph of historical value.

ESTIMATION BY EXPERIMENTATION

If equipment similar to the equipment used to acquire the degraded image is available, it is possible in principle to obtain an accurate estimate of the degradation. Images similar to the degraded image can be acquired with various system settings

a b

FIGURE 5.24

Estimating a degradation by impulse characterization.
 (a) An impulse of light (shown magnified).
 (b) Imaged (degraded) impulse.



until they are degraded as closely as possible to the image we wish to restore. Then the idea is to obtain the impulse response of the degradation by imaging an impulse (small dot of light) using the same system settings. As noted in Section 5.5, a linear, space-invariant system is characterized completely by its impulse response.

An impulse is simulated by a bright dot of light, as bright as possible to reduce the effect of noise to negligible values. Then, recalling that the Fourier transform of an impulse is a constant, it follows from Eq. (5-65) that

$$H(u, v) = \frac{G(u, v)}{A} \quad (5-67)$$

where, as before, $G(u, v)$ is the Fourier transform of the observed image, and A is a constant describing the strength of the impulse. Figure 5.24 shows an example.

ESTIMATION BY MODELING

Degradation modeling has been used for many years because of the insight it affords into the image restoration problem. In some cases, the model can even take into account environmental conditions that cause degradations. For example, a degradation model proposed by Hufnagel and Stanley [1964] is based on the physical characteristics of atmospheric turbulence. This model has a familiar form:

$$H(u, v) = e^{-k(u^2 + v^2)^{5/6}} \quad (5-68)$$

where k is a constant that depends on the nature of the turbulence. With the exception of the $5/6$ power in the exponent, this equation has the same form as the Gaussian lowpass filter transfer function discussed in Section 4.8. In fact, the Gaussian LPF is used sometimes to model mild, uniform blurring. Figure 5.25 shows examples obtained by simulating blurring an image using Eq. (5-68) with values $k = 0.0025$

a	b
c	d

FIGURE 5.25

Modeling turbulence.

(a) No visible turbulence.

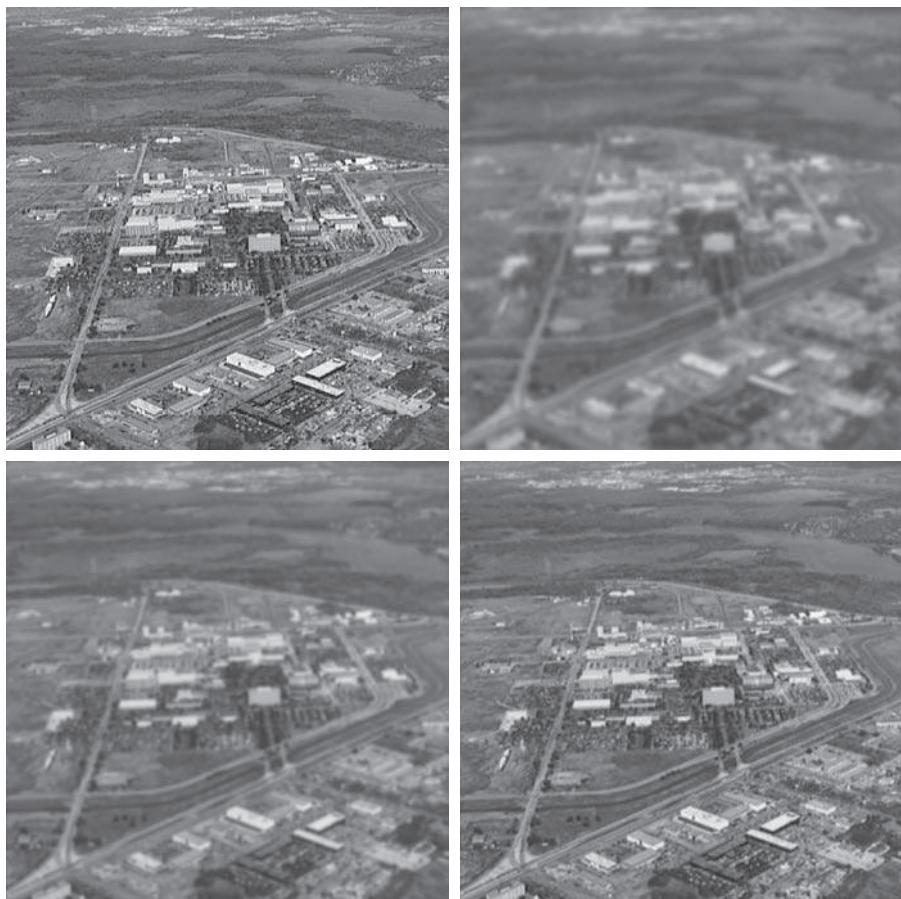
(b) Severe turbulence, $k = 0.0025$.

(c) Mild turbulence, $k = 0.001$.

(d) Low turbulence, $k = 0.00025$.

All images are of size 480×480 pixels.

(Original image courtesy of NASA.)



(severe turbulence), $k = 0.001$ (mild turbulence), and $k = 0.00025$ (low turbulence). We restore these images using various methods later in this chapter.

Another approach used frequently in modeling is to derive a mathematical model starting from basic principles. We illustrate this procedure by treating in some detail the case in which an image has been blurred by uniform linear motion between the image and the sensor during image acquisition. Suppose that an image $f(x, y)$ undergoes planar motion and that $x_0(t)$ and $y_0(t)$ are the time-varying components of motion in the x - and y -directions, respectively. We obtain the total exposure at any point of the recording medium (say, film or digital memory) by integrating the instantaneous exposure over the time interval during which the imaging system shutter is open.

Assuming that shutter opening and closing takes place instantaneously, and that the optical imaging process is perfect, lets us isolate the effects due to image motion. Then, if T is the duration of the exposure, it follows that

$$g(x, y) = \int_0^T f[x - x_0(t), y - y_0(t)] dt \quad (5-69)$$

where $g(x, y)$ is the blurred image.

The continuous Fourier transform of this expression is

$$G(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-j2\pi(ux + vy)} dx dy \quad (5-70)$$

Substituting Eq. (5-69) into Eq. (5-70) yields

$$G(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[\int_0^T f[x - x_0(t), y - y_0(t)] dt \right] e^{-j2\pi(ux + vy)} dx dy \quad (5-71)$$

Reversing the order of integration results in the expression

$$G(u, v) = \int_0^T \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f[x - x_0(t), y - y_0(t)] e^{-j2\pi(ux + vy)} dx dy \right] dt \quad (5-72)$$

The term inside the outer brackets is the Fourier transform of the displaced function $f[x - x_0(t), y - y_0(t)]$. Using entry 3 in Table 4.4 then yields the expression

$$\begin{aligned} G(u, v) &= \int_0^T F(u, v) e^{-j2\pi[ux_0(t) + vy_0(t)]} dt \\ &= F(u, v) \int_0^T e^{-j2\pi[ux_0(t) + vy_0(t)]} dt \end{aligned} \quad (5-73)$$

By defining

$$H(u, v) = \int_0^T e^{-j2\pi[ux_0(t) + vy_0(t)]} dt \quad (5-74)$$

we can express Eq. (5-73) in the familiar form

$$G(u, v) = H(u, v)F(u, v) \quad (5-75)$$

If the motion variables $x_0(t)$ and $y_0(t)$ are known, the transfer function $H(u, v)$ can be obtained directly from Eq. (5-74). As an illustration, suppose that the image in question undergoes uniform linear motion in the x -direction only (i.e., $y_0(t) = 0$), at a rate $x_0(t) = at/T$. When $t = T$, the image has been displaced by a total distance a . With $y_0(t) = 0$, Eq. (5-74) yields

$$\begin{aligned} H(u, v) &= \int_0^T e^{-j2\pi u x_0(t)} dt = \int_0^T e^{-j2\pi u a t/T} dt \\ &= \frac{T}{\pi u a} \sin(\pi u a) e^{-j\pi u a} \end{aligned} \quad (5-76)$$

a b

FIGURE 5.26

(a) Original image. (b) Result of blurring using the function in Eq. (5-77) with $a = b = 0.1$ and $T = 1$.



If we allow the y -component to vary as well, with the motion given by $y_0(t) = bt/T$, then the degradation function becomes

$$H(u, v) = \frac{T}{\pi(ua + vb)} \sin[\pi(ua + vb)] e^{-j\pi(ua + vb)} \quad (5-77)$$

To generate a discrete filter transfer function of size $M \times N$, we sample this equation for $u = 0, 1, 2, \dots, M - 1$ and $v = 0, 1, 2, \dots, N - 1$.

EXAMPLE 5.8: Image blurring caused by motion.

Figure 5.26(b) is an image blurred by computing the Fourier transform of the image in Fig. 5.26(a), multiplying the transform by $H(u, v)$ from Eq. (5-77), and taking the inverse transform. The images are of size 688×688 pixels, and we used $a = b = 0.1$ and $T = 1$ in Eq. (5-77). As we will discuss in Sections 5.8 and 5.9, recovery of the original image from its blurred counterpart presents some interesting challenges, particularly when noise is present in the degraded image. As mentioned at the end of Section 5.5, we perform all DFT computations without padding.

5.7 INVERSE FILTERING

The material in this section is our first step in studying restoration of images degraded by a degradation function \mathcal{H} , which is given, or is obtained by a method such as those discussed in the previous section. The simplest approach to restoration is direct inverse filtering, where we compute an estimate, $\hat{F}(u, v)$, of the transform of the original image by dividing the transform of the degraded image, $G(u, v)$, by the degradation transfer function:

$$\hat{F}(u, v) = \frac{G(u, v)}{H(u, v)} \quad (5-78)$$

The division is elementwise, as defined in Section 2.6 and in connection with Eq. (5-65). Substituting the right side of Eq. (5-2) for $G(u,v)$ in Eq. (5-78) yields

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)} \quad (5-79)$$

This is an interesting expression. It tells us that, even if we know the degradation function, we cannot recover the undegraded image [the inverse Fourier transform of $F(u,v)$] exactly because $N(u,v)$ is not known. There is more bad news. If the degradation function has zero or very small values, then the ratio $N(u,v)/H(u,v)$ could easily dominate the term $F(u,v)$. In fact, this is frequently the case, as you will see shortly.

One approach to get around the zero or small-value problem is to limit the filter frequencies to values near the origin. From the discussion of Eq. (4-92), we know that $H(0,0)$ is usually the highest value of $H(u,v)$ in the frequency domain. Thus, by limiting the analysis to frequencies near the origin, we reduce the likelihood of encountering zero values. The following example illustrates this approach.

EXAMPLE 5.9: Image deblurring by inverse filtering.

The image in Fig. 5.25(b) was inverse filtered with Eq. (5-78) using the exact inverse of the degradation function that generated that image. That is, the degradation function used was

$$H(u,v) = e^{-k[(u + M/2)^2 + (v - N/2)^2]^{5/6}}$$

with $k = 0.0025$. The $M/2$ and $N/2$ constants are offset values; they center the function so that it will correspond with the centered Fourier transform, as discussed in the previous chapter. (Remember, we do not use padding with these functions.) In this case, $M = N = 480$. We know that a Gaussian function has no zeros, so that will not be a concern here. However, despite this, the degradation values became so small that the result of full inverse filtering [Fig. 5.27(a)] is useless. The reasons for this poor result are as discussed in connection with Eq. (5-79).

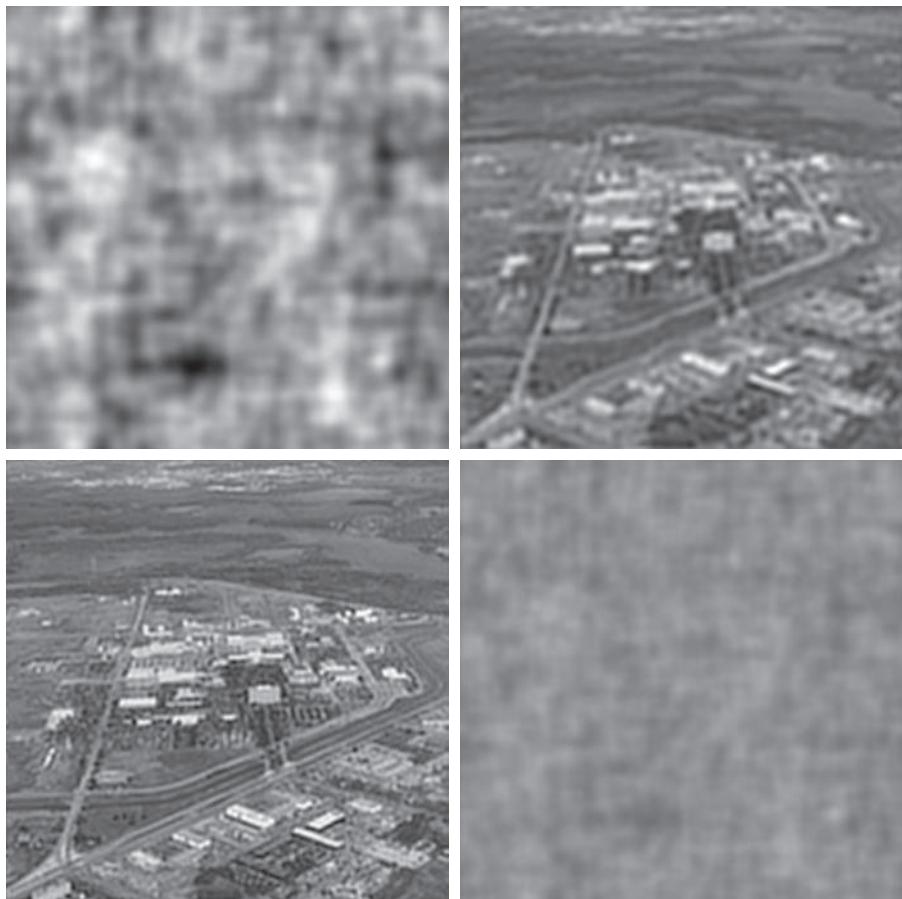
Figures 5.27(b) through (d) show the results of cutting off values of the ratio $G(u,v)/H(u,v)$ outside a radius of 40, 70, and 85, respectively. The cut off was implemented by applying to the ratio a Butterworth lowpass function of order 10. This provided a sharp (but smooth) transition at the desired radius. Radii near 70 yielded the best visual results [Fig. 5.27(c)]. Radii below 70 resulted in blurred images, as in Fig. 5.27(b), which was obtained using a radius of 40. Values above 70 started to produce degraded images, as illustrated in Fig. 5.27(d), which was obtained using a radius of 85. The image content is almost visible in this image behind a “curtain” of noise, but the noise definitely dominates the result. Further increases in radius values produced images that looked more and more like Fig. 5.27(a).

The results in the preceding example are illustrative of the poor performance of direct inverse filtering in general. The basic theme of the three sections that follow is how to improve on direct inverse filtering.

a	b
c	d

FIGURE 5.27

Restoring Fig. 5.25(b) using Eq. (5-78).
 (a) Result of using the full filter.
 (b) Result with H cut off outside a radius of 40.
 (c) Result with H cut off outside a radius of 70.
 (d) Result with H cut off outside a radius of 85.



5.8 MINIMUM MEAN SQUARE ERROR (WIENER) FILTERING

The inverse filtering approach discussed in the previous section makes no explicit provision for handling noise. In this section, we discuss an approach that incorporates both the degradation function and statistical characteristics of noise into the restoration process. The method is founded on considering images and noise as random variables, and the objective is to find an estimate \hat{f} of the uncorrupted image f such that the mean square error between them is minimized. This error measure is defined as

$$e^2 = E \left\{ (f - \hat{f})^2 \right\} \quad (5-80)$$

where $E\{\cdot\}$ is the expected value of the argument. We assume that the noise and the image are uncorrelated, that one or the other has zero mean, and that the intensity levels in the estimate are a linear function of the levels in the degraded image. Based

on these assumptions, the minimum of the error function in Eq. (5-80) is given in the frequency domain by the expression

$$\begin{aligned}\hat{F}(u, v) &= \left[\frac{H^*(u, v)S_f(u, v)}{S_f(u, v)|H(u, v)|^2 + S_\eta(u, v)} \right] G(u, v) \\ &= \left[\frac{H^*(u, v)}{|H(u, v)|^2 + S_\eta(u, v)/S_f(u, v)} \right] G(u, v) \\ &= \left[\frac{1}{H(u, v)} \frac{|H(u, v)|^2}{|H(u, v)|^2 + S_\eta(u, v)/S_f(u, v)} \right] G(u, v)\end{aligned}\quad (5-81)$$

where we used the fact that the product of a complex quantity with its conjugate is equal to the magnitude of the complex quantity squared. This result is known as the *Wiener filter*, after N. Wiener [1942], who first proposed the concept in the year shown. The filter, which consists of the terms inside the brackets, also is commonly referred to as the *minimum mean square error filter* or the *least square error filter*. We include references at the end of the chapter to sources containing detailed derivations of the Wiener filter. Note from the first line in Eq. (5-81) that the Wiener filter does not have the same problem as the inverse filter with zeros in the degradation function, unless the entire denominator is zero for the same value(s) of u and v .

The terms in Eq. (5-81) are as follows:

1. $\hat{F}(u, v)$ = Fourier transform of the estimate of the undegraded image.
2. $G(u, v)$ = Fourier transform of the degraded image.
3. $H(u, v)$ = degradation transfer function (Fourier transform of the spatial degradation).
4. $H^*(u, v)$ = complex conjugate of $H(u, v)$.
5. $|H(u, v)|^2 = H^*(u, v)H(u, v)$.
6. $S_\eta(u, v) = |N(u, v)|^2$ = power spectrum of the noise [see Eq. (4-89)][†]
7. $S_f(u, v) = |F(u, v)|^2$ = power spectrum of the undegraded image.

The restored image in the spatial domain is given by the inverse Fourier transform of the frequency-domain estimate $\hat{F}(u, v)$. Note that if the noise is zero, then the noise power spectrum vanishes and the Wiener filter reduces to the inverse filter. Also, keep in mind the discussion at the end of Section 5.5 regarding the fact that all transform work in this chapter is done without padding.

[†]The term $|N(u, v)|^2$ also is referred to as the *autocorrelation* of the noise. This term comes from the correlation theorem (first line of entry 7 in Table 4.4). When the two functions are the same, correlation becomes *autocorrelation* and the right side of that entry becomes $H^*(u, v)H(u, v)$, which is equal to $|H(u, v)|^2$. Similar comments apply to $|F(u, v)|^2$, which is the autocorrelation of the image. We will discuss correlation in more detail in Chapter 12.

A number of useful measures are based on the power spectra of noise and of the undegraded image. One of the most important is the *signal-to-noise ratio*, approximated using frequency domain quantities such as

$$\text{SNR} = \frac{\sum_{u=0}^{M-1} \sum_{v=0}^{N-1} |F(u,v)|^2}{\sum_{u=0}^{M-1} \sum_{v=0}^{N-1} |N(u,v)|^2} \quad (5-82)$$

This ratio gives a measure of the level of information-bearing signal power (i.e., of the original, undegraded image) to the level of noise power. An image with low noise would tend to have a high SNR and, conversely, the same image with a higher level of noise would have a lower SNR. This ratio is an important measure used in characterizing the performance of restoration algorithms.

The *mean square error* given in statistical form in Eq. (5-80) can be approximated also in terms of a summation involving the original and restored images:

$$\text{MSE} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x,y) - \hat{f}(x,y)]^2 \quad (5-83)$$

In fact, if one considers the restored image to be “signal” and the difference between this image and the original to be “noise,” we can define a signal-to-noise ratio in the spatial domain as

$$\text{SNR} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x,y) - \hat{f}(x,y)]^2} \quad (5-84)$$

The closer f and \hat{f} are, the larger this ratio will be. Sometimes the square root of the preceding two measures is used instead, in which case they are referred to as the *root-mean-square-error* and the *root-mean-square-signal-to-noise ratio*, respectively. As we have mentioned before, keep in mind that quantitative measures do not necessarily relate well to perceived image quality.

When dealing with white noise, the spectrum is a constant, which simplifies things considerably. However, the power spectrum of the undegraded image seldom is known. An approach frequently used when these quantities are not known, or cannot be estimated, is to approximate Eq. (5-81) by the expression

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + K} \right] G(u,v) \quad (5-85)$$

where K is a specified constant that is added to all terms of $|H(u,v)|^2$. The following examples illustrate the use of this expression.



a b c

FIGURE 5.28 Comparison of inverse and Wiener filtering. (a) Result of full inverse filtering of Fig. 5.25(b). (b) Radially limited inverse filter result. (c) Wiener filter result.

EXAMPLE 5.10: Comparison of deblurring by inverse and Wiener filtering.

Figure 5.28 illustrates the advantage of Wiener filtering over direct inverse filtering. Figure 5.28(a) is the full inverse-filtered result from Fig. 5.27(a). Similarly, Fig. 5.28(b) is the radially limited inverse filter result of Fig. 5.27(c). These images are duplicated here for convenience in making comparisons. Figure 5.28(c) shows the result obtained using Eq. (5-85) with the degradation function used in Example 5.9. The value of K was chosen interactively to yield the best visual results. The advantage of Wiener filtering over the direct inverse approach is evident in this example. By comparing Figs. 5.25(a) and 5.28(c), we see that the Wiener filter yielded a result very close in appearance to the original, undegraded image.

EXAMPLE 5.11: More deblurring examples using Wiener filtering.

The first row of Fig. 5.29 shows, from left to right, the blurred image of Fig. 5.26(b) heavily corrupted by additive Gaussian noise of zero mean and variance of 650; the result of direct inverse filtering; and the result of Wiener filtering. The Wiener filter of Eq. (5-85) was used, with $H(u, v)$ from Example 5.8, and with K chosen interactively to give the best possible visual result. As expected, direct inverse filtering produced an unusable image. Note that the noise in the inverse filtered image is so strong that it masks completely the content of the image. The Wiener filter result is by no means perfect, but it does give us a hint as to image content. The text can be read with moderate effort.

The second row of Fig. 5.29 shows the same sequence just discussed, but with the level of the noise variance reduced by one order of magnitude. This reduction had little effect on the inverse filter, but the Wiener results are considerably improved. For example, the text is much easier to read now. In the third row of Fig. 5.29, the noise variance was reduced more than five orders of magnitude from the first row. In fact, image in Fig. 5.29(g) has no visible noise. The inverse filter result is interesting in this case. The noise is still quite visible, but the text can be seen through a “curtain” of noise (see Problem 5.30). The Wiener filter result in Fig. 5.29(i) is excellent, being quite close visually to the original image in Fig.



FIGURE 5.29 (a) 8-bit image corrupted by motion blur and additive noise. (b) Result of inverse filtering. (c) Result of Wiener filtering. (d)–(f) Same sequence, but with noise variance one order of magnitude less. (g)–(i) Same sequence, but noise variance reduced by five orders of magnitude from (a). Note in (h) how the deblurred image is quite visible through a “curtain” of noise.

5.26(a). In practice, the results of restoration filtering are seldom this close to the original images. This example, and Example 5.12 in the next section, were idealized slightly to focus on the effects of noise on restoration algorithms.

5.9 CONSTRAINED LEAST SQUARES FILTERING

The problem of having to know something about the degradation function H is common to all methods discussed in this chapter. However, the Wiener filter presents an additional difficulty: the power spectra of the undegraded image and noise must be known also. We showed in the previous section that in some cases it is possible to achieve acceptable results using the approximation in Eq. (5-85), but a constant value for the ratio of the power spectra is not always a suitable solution.

The method discussed in this section requires knowledge of only the mean and variance of the noise. As discussed in Section 5.2, these parameters generally can be calculated from a given degraded image, so this is an important advantage. Another difference is that the Wiener filter is based on minimizing a statistical criterion and, as such, it is optimal in an average sense. The algorithm presented in this section has the notable feature that it yields an optimal result for each image to which it is applied. Of course, it is important to keep in mind that these optimality criteria, although they are comforting from a theoretical point of view, are not related to the dynamics of visual perception. As a result, the choice of one algorithm over the other will almost always be determined by the perceived visual quality of the resulting images.

By using the definition of convolution given in Eq. (4-94), and as explained in Section 2.6, we can express Eq. (5-64) in vector-matrix form:

$$\mathbf{g} = \mathbf{Hf} + \boldsymbol{\eta} \quad (5-86)$$

For example, suppose that $g(x, y)$ is of size $M \times N$. We can form the first N elements of vector \mathbf{g} by using the image elements in the first row of $g(x, y)$, the next N elements from the second row, and so on. The dimensionality of the resulting vector will be $MN \times 1$. These are also the dimensions of \mathbf{f} and $\boldsymbol{\eta}$, as these vectors are formed in the same manner. Matrix \mathbf{H} then has dimensions $MN \times MN$. Its elements are given by the elements of the convolution in Eq. (4-94).

It would be reasonable to arrive at the conclusion that the restoration problem can now be reduced to simple matrix manipulations. Unfortunately, this is not the case. For instance, suppose that we are working with images of medium size, say $M = N = 512$. Then the vectors in Eq. (5-86) would be of dimension $262,144 \times 1$ and matrix \mathbf{H} would be of dimension $262,144 \times 262,144$. Manipulating vectors and matrices of such sizes is not a trivial task. The problem is complicated further by the fact that \mathbf{H} is highly sensitive to noise (after the experiences we had with the effect of noise in the previous two sections, this should not be a surprise). The key advantage of formulating the restoration problem in matrix form is that it facilitates derivation of restoration algorithms.

Although we do not fully derive the method of constrained least squares that we are about to present, this method has its roots in a matrix formulation. We give

See Gonzalez and Woods [1992] for an entire chapter devoted to the topic of algebraic techniques for image restoration.

references at the end of the chapter to sources where derivations are covered in detail. Central to the method is the issue of the sensitivity of \mathbf{H} to noise. One way to reduce the effects of noise sensitivity, is to base optimality of restoration on a measure of smoothness, such as the second derivative of an image (our old friend, the Laplacian). To be meaningful, the restoration must be constrained by the parameters of the problems at hand. Thus, what is desired is to find the minimum of a criterion function, C , defined as

$$C = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\nabla^2 f(x, y)]^2 \quad (5-87)$$

subject to the constraint

$$\|\mathbf{g} - \mathbf{H}\hat{\mathbf{f}}\|^2 = \|\boldsymbol{\eta}\|^2 \quad (5-88)$$

where $\|\mathbf{a}\|^2 \triangleq \mathbf{a}^T \mathbf{a}$ is the Euclidean norm (see Section 2.6), and $\hat{\mathbf{f}}$ is the estimate of the undegraded image. The Laplacian operator ∇^2 is defined in Eq. (3-50).

The frequency domain solution to this optimization problem is given by the expression

$$\hat{F}(u, v) = \left[\frac{H^*(u, v)}{|H(u, v)|^2 + \gamma |P(u, v)|^2} \right] G(u, v) \quad (5-89)$$

The quantity in brackets is the transfer function of the constrained least squares filter. Note that it reduces to the inverse filter transfer function when $\gamma = 0$.

where γ is a parameter that must be adjusted so that the constraint in Eq. (5-88) is satisfied, and $P(u, v)$ is the Fourier transform of the function

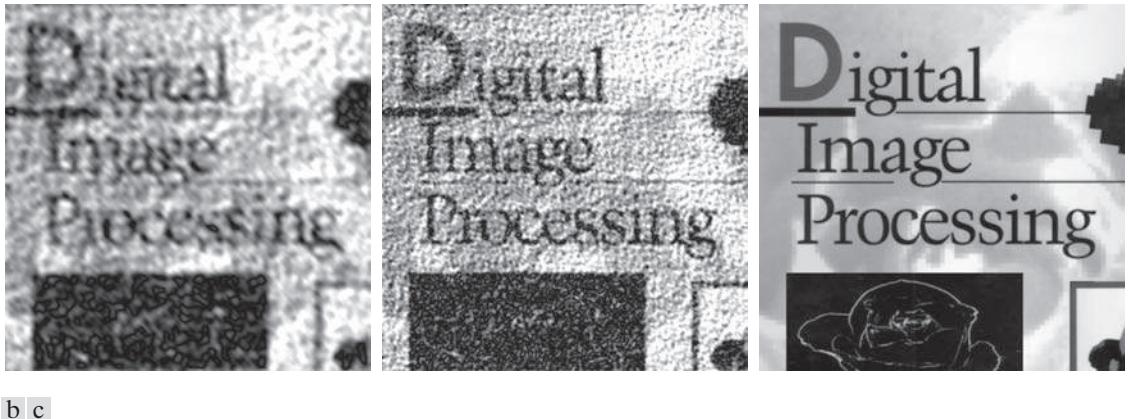
$$p(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (5-90)$$

We recognize this function as a Laplacian kernel from Fig. 3.45. Note that Eq. (5-89) reduces to inverse filtering if $\gamma = 0$.

Functions $P(u, v)$ and $H(u, v)$ must be of the same size. If H is of size $M \times N$, this means that $p(x, y)$ must be embedded in the center of an $M \times N$ array of zeros. In order to preserve the even symmetry of $p(x, y)$, M and N must be even integers, as explained in Examples 4.10 and 4.15. If a given degraded image from which H is obtained is not of even dimensions, then a row and/or column, as appropriate, must be deleted before computing H for use in Eq. (5-89).

EXAMPLE 5.12: Comparison of deblurring by Wiener and constrained least squares filtering.

Figure 5.30 shows the result of processing Figs. 5.29(a), (d), and (g) with constrained least squares filters, in which the values of γ were selected manually to yield the best visual results. This is the same procedure we used to generate the Wiener filter results in Fig. 5.29(c), (f), and (i). By comparing the constrained least squares and Wiener results, we see that the former yielded better results (especially in terms of noise reduction) for the high- and medium-noise cases, with both filters generating essentially



a b c

FIGURE 5.30 Results of constrained least squares filtering. Compare (a), (b), and (c) with the Wiener filtering results in Figs. 5.29(c), (f), and (i), respectively.

equal results for the low-noise case. This is not surprising because parameter γ in Eq. (5-89) is a true scalar, whereas the value of K in Eq. (5-85) is a scalar approximation to the ratio of two unknown frequency domain *functions* of size $M \times N$. Thus, it stands to reason that a result based on manually selecting γ would be a more accurate estimate of the undegraded image. As in Example 5.11, the results in this example are better than one normally finds in practice. Our focus here was on the effects of noise blurring on restoration. As noted earlier, you will encounter situations in which the restoration solutions are not quite as close to the original images as we have shown in these two examples.

As discussed in the preceding example, it is possible to adjust the parameter γ interactively until acceptable results are achieved. However, if we are interested in mathematical optimality, then this parameter must be adjusted so that the constraint in Eq. (5-88) is satisfied. A procedure for computing γ by iteration is as follows.

Define a “residual” vector \mathbf{r} as

$$\mathbf{r} = \mathbf{g} - \mathbf{H}\hat{\mathbf{f}} \quad (5-91)$$

From Eq. (5-89), we see that $\hat{F}(u, v)$ (and by implication $\hat{\mathbf{f}}$) is a function of γ . Then it follows that \mathbf{r} also is a function of this parameter. It can be shown (Hunt [1973], Gonzalez and Woods [1992]) that

$$\begin{aligned} \phi(\gamma) &= \mathbf{r}^T \mathbf{r} \\ &= \|\mathbf{r}\|^2 \end{aligned} \quad (5-92)$$

is a monotonically increasing function of γ . What we want to do is adjust γ so that

$$\|\mathbf{r}\|^2 = \|\boldsymbol{\eta}\|^2 \pm \alpha \quad (5-93)$$

where α is an accuracy factor. In view of Eq. (5-91), if $\|\mathbf{r}\|^2 = \|\boldsymbol{\eta}\|^2$, the constraint in Eq. (5-88) will be strictly satisfied.

Because $\phi(\gamma)$ is monotonic, finding the desired value of γ is not difficult. One approach is to

1. Specify an initial value of γ .
2. Compute $\|\mathbf{r}\|^2$.
3. Stop if Eq. (5-93) is satisfied; otherwise return to Step 2 after increasing γ if $\|\mathbf{r}\|^2 < (\|\boldsymbol{\eta}\|^2 - \alpha)$ or decreasing γ if $\|\mathbf{r}\|^2 > (\|\boldsymbol{\eta}\|^2 + \alpha)$. Use the new value of γ in Eq. (5-89) to recompute the optimum estimate $\hat{F}(u, v)$.

Other procedures, such as a Newton–Raphson algorithm, can be used to improve the speed of convergence.

In order to use this algorithm, we need the quantities $\|\mathbf{r}\|^2$ and $\|\boldsymbol{\eta}\|^2$. To compute $\|\mathbf{r}\|^2$, we note from Eq. (5-91) that

$$R(u, v) = G(u, v) - H(u, v)F(u, v) \quad (5-94)$$

from which we obtain $r(x, y)$ by computing the inverse Fourier transform of $R(u, v)$. Then, from the definition of the Euclidean norm, it follows that

$$\|\mathbf{r}\|^2 = \mathbf{r}^T \mathbf{r} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} r^2(x, y) \quad (5-95)$$

Computation of $\|\boldsymbol{\eta}\|^2$ leads to an interesting result. First, consider the variance of the noise over the entire image, which we estimate from the samples using the expression

$$\sigma_\eta^2 = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\eta(x, y) - \bar{\eta}]^2 \quad (5-96)$$

where

$$\bar{\eta} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \eta(x, y) \quad (5-97)$$

is the sample mean. With reference to the form of Eq. (5-95), we note that the double summation in Eq. (5-96) is proportional to $\|\boldsymbol{\eta}\|^2$. This leads to the expression

$$\|\boldsymbol{\eta}\|^2 = MN[\sigma_\eta^2 + \bar{\eta}^2] \quad (5-98)$$

This is a most useful result. It tells us that we can estimate the unknown quantity $\|\boldsymbol{\eta}\|^2$ by having knowledge of only the mean and variance of the noise. These quantities are not difficult to estimate (see Section 5.2), assuming that the noise and image

a b

FIGURE 5.31

(a) Iteratively determined constrained least squares restoration of Fig. 5.25(b), using correct noise parameters. (b) Result obtained with wrong noise parameters.



intensity values are not correlated. This is an assumption of all the methods discussed in this chapter.

EXAMPLE 5.13: Iterative estimation of the optimum constrained least squares filter.

Figure 5.31(a) shows the result obtained using the algorithm just described to estimate the optimum filter for restoring Fig. 5.25(b). The initial value used for γ was 10^{-5} , the correction factor for adjusting γ was 10^{-6} , and the value for α was 0.25. The noise parameters specified were the same used to generate Fig. 5.25(a): a noise variance of 10^{-5} , and zero mean. The restored result is comparable to Fig. 5.28(c), which was obtained by Wiener filtering with K manually specified for best visual results. Figure 5.31(b) shows what can happen if the wrong estimate of noise parameters are used. In this case, the noise variance specified was 10^{-2} and the mean was left at 0. The result in this case is considerably more blurred.

5.10 GEOMETRIC MEAN FILTER

It is possible to generalize slightly the Wiener filter discussed in Section 5.8. The generalization is in the form of the so-called *geometric mean filter*:

$$\hat{F}(u, v) = \left[\frac{H^*(u, v)}{|H(u, v)|^2} \right]^\alpha \left[\frac{H^*(u, v)}{\left| H(u, v) \right|^2 + \beta \left[\frac{S_\eta(u, v)}{S_f(u, v)} \right]} \right]^{1-\alpha} G(u, v) \quad (5-99)$$

where α and β are nonnegative, real constants. The geometric mean filter transfer function consists of the two expressions in brackets raised to the powers α and $1 - \alpha$, respectively.

When $\alpha = 1$ the geometric mean filter reduces to the inverse filter. With $\alpha = 0$ the filter becomes the so-called *parametric Wiener filter*, which reduces to the “standard”

Wiener filter when $\beta = 1$. If $\alpha = 1/2$, the filter becomes a product of the two quantities raised to the same power, which is the definition of the geometric mean, thus giving the filter its name. With $\beta = 1$, as α increases above $1/2$, the filter performance will tend more toward the inverse filter. Similarly, when α decreases below $1/2$, the filter will behave more like a Wiener filter. When $\alpha = 1/2$ and $\beta = 1$ the filter is commonly referred to as a *spectrum equalization filter*. Equation (5-99) is useful when implementing restoration filters because it represents a family of filters combined into a single expression.

5.11 IMAGE RECONSTRUCTION FROM PROJECTIONS

As noted in Chapter 1, the term *computerized axial tomography* (CAT) is used interchangeably to denote CT.

In the previous sections of this chapter we discussed techniques for restoring degraded images. In this section, we examine the problem of *reconstructing* an image from a series of projections, with a focus on X-ray *computed tomography* (CT). This is the earliest and still the most-widely used type of CT, and is currently one of the principal applications of digital image processing in medicine.

INTRODUCTION

The reconstruction problem is simple in principle, and can be explained qualitatively in a straightforward, intuitive manner, without using equations (we will deal with the math later in this section). To begin, consider Fig. 5.32(a), which consists of a single object on a uniform background. In order to bring physical meaning to the following explanation, suppose that this image is a *cross-section* of a 3-D region of a human body. Assume also that the background in the image represents soft, uniform tissue, while the round object is a tumor, also uniform, but with higher X-ray absorption characteristics.

Suppose next that we pass a thin, flat beam of X-rays from left to right (through the plane of the image), as Fig. 5.32(b) shows, and assume that the energy of the beam is absorbed more by the object than by the background, as typically is the case. Using a strip of X-ray absorption detectors on the other side of the region will yield the signal (*absorption profile*) shown, whose amplitude (intensity) is proportional to absorption.[†] We may view any point in the signal as the sum of the absorption values across the single ray in the beam corresponding spatially to that point (such a sum often is referred to as a *raysum*). At this juncture, all the information we have about the object is this 1-D absorption signal.

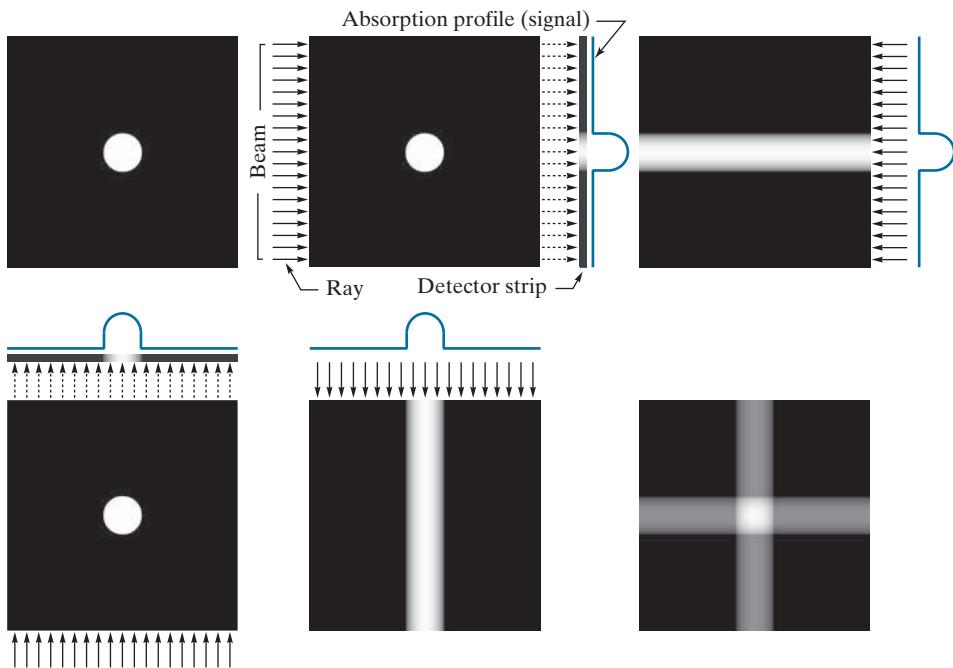
We have no way of determining from a single projection whether we are dealing with a single object, or a multitude of objects along the path of the beam, but we begin the reconstruction by creating an *image* based only on this information. The approach is to project the 1-D signal *back* in the opposite direction from which the beam came, as Fig. 5.32(c) shows. The process of backprojecting a 1-D signal across a 2-D area sometimes is referred to as *smearing* the projection back across the area. In

[†]A treatment of the physics of X-ray sources and detectors is beyond the scope of our discussion, which focuses on the image processing aspects of CT. See Prince and Links [2006] for an excellent introduction to the physics of X-ray image formation.

a b c
d e f

FIGURE 5.32

- (a) Flat region with a single object.
- (b) Parallel beam, detector strip, and profile of sensed 1-D absorption signal.
- (c) Result of back-projecting the absorption profile.
- (d) Beam and detectors rotated by 90° .
- (e) Backprojection.
- (f) The sum of (c) and (e), intensity-scaled. The intensity where the backprojections intersect is twice the intensity of the individual back-projections.



terms of digital images, this means duplicating the same 1-D signal across the image, perpendicularly to the direction of the beam. For example, Fig. 5.32(c) was created by duplicating the 1-D signal in all columns of the reconstructed image. For obvious reasons, the approach just described is called *backprojection*.

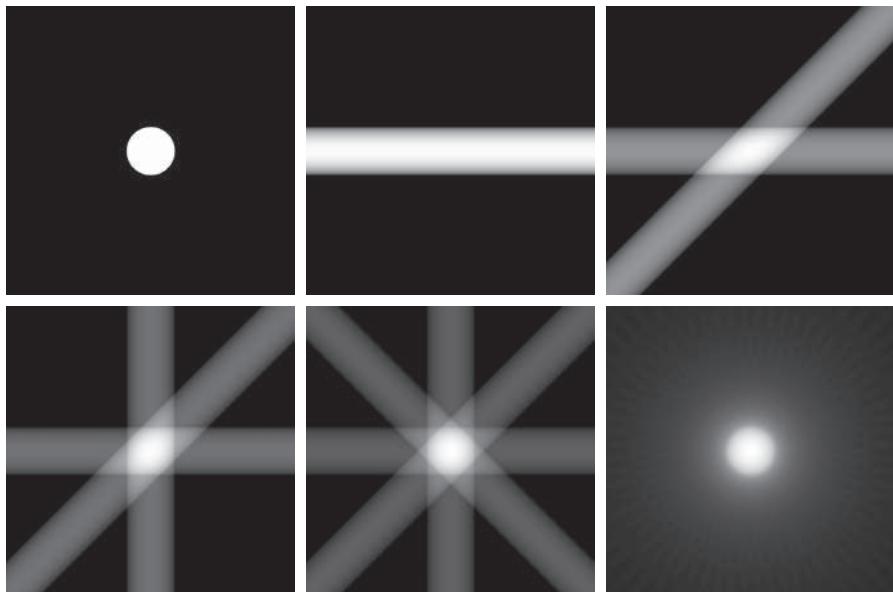
Next, suppose that we rotate the position of the source-detector pair by 90° , as in Fig. 5.32(d). Repeating the procedure explained in the previous paragraph yields a backprojection image in the vertical direction, as Fig. 5.32(e) shows. We continue the reconstruction by *adding* this result to the previous backprojection, resulting in Fig. 5.32(f). Now, we begin to suspect that the object of interest is contained in the square shown, whose amplitude is twice the amplitude of the individual backprojections because the signals were added. We should be able to learn more about the shape of the object in question by taking more views in the manner just described, as Fig. 5.33 shows. As the number of projections increases, the amplitude strength of non-intersecting backprojections decreases relative to the strength of regions in which multiple backprojections intersect. The net effect is that brighter regions will dominate the result, and backprojections with few or no intersections will fade into the background as the image is scaled for display.

Figure 5.33(f), which was formed from 32 backprojections, illustrates this concept. Note, however, that while this reconstructed image is a reasonably good approximation to the shape of the original object, the image is blurred by a “halo” effect, the formation of which can be seen in progressive stages in Fig. 5.33. For example, the halo in Fig. 5.33(e) appears as a “star” whose intensity is lower than that of the

a	b	c
d	e	f

FIGURE 5.33

- (a) Same as Fig. 5.32(a).
 (b)-(e) Reconstruction using 1, 2, 3, and 4 back-projections 45° apart.
 (f) Reconstruction with 32 backprojections 5.625° apart (note the blurring).



object, but higher than the background. As the number of views increases, the shape of the halo becomes circular, as in Fig. 5.33(f). Blurring in CT reconstruction is an important issue, whose solution is addressed later in this section. Finally, we conclude from the discussion of Figs. 5.32 and 5.33 that backprojections 180° apart are mirror images of each other, so we have to consider only angle increments halfway around a circle in order to generate all the backprojections required for reconstruction.

EXAMPLE 5.14: Backprojections of a planar region containing two objects.

Figure 5.34 illustrates reconstruction using backprojections on a region that contains two objects with different absorption properties (the larger object has higher absorption). Figure 5.34(b) shows the result of using one backprojection. We note three principal features in this figure, from bottom to top: a thin horizontal gray band corresponding to the unoccluded portion of the small object, a brighter (more absorption) band above it corresponding to the area shared by both objects, and an upper band corresponding to the rest of the elliptical object. Figures 5.34(c) and (d) show reconstruction using two projections 90° apart and four projections 45° apart, respectively. The explanation of these figures is similar to the discussion of Figs. 5.33(c) through (e). Figures 5.34(e) and (f) show more accurate reconstructions using 32 and 64 backprojections, respectively. The last two results are quite close visually, and they both show the blurring problem mentioned earlier.

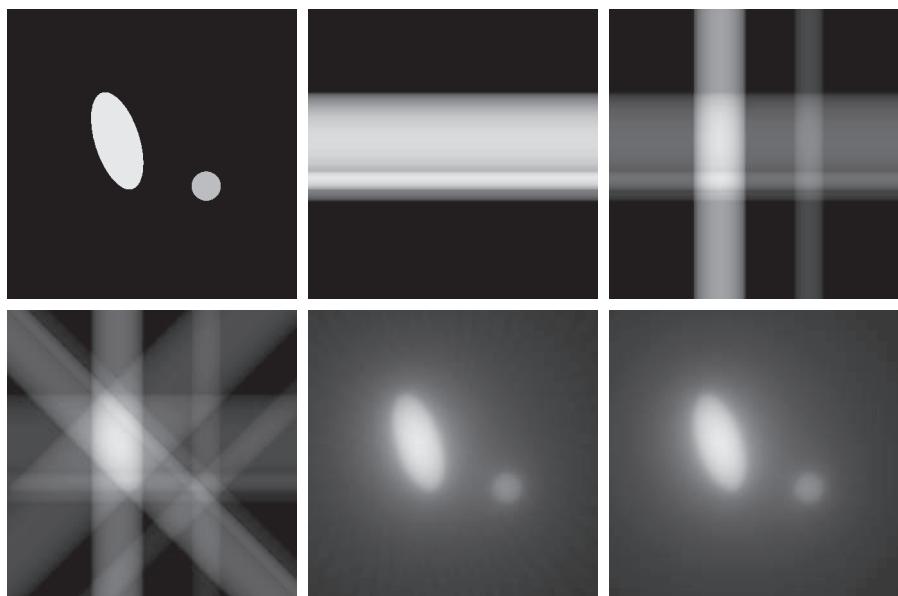
PRINCIPLES OF X-RAY COMPUTED TOMOGRAPHY (CT)

As with the Fourier transform discussed in the last chapter, the basic mathematical concepts required for CT were in place many years before the availability of digital

a	b	c
d	e	f

FIGURE 5.34

- (a) Two objects with different absorption characteristics.
 (b)–(d) Reconstruction using 1, 2, and 4 backprojections, 45° apart.
 (e) Reconstruction with 32 backprojections, 5.625° apart.
 (f) Reconstruction with 64 backprojections, 2.8125° apart.



computers made them practical. The theoretical foundation of CT dates back to Johann Radon, a mathematician from Vienna who derived a method in 1917 for projecting a 2-D object along parallel rays, as part of his work on line integrals (the method now is referred to as the *Radon transform*, a topic we will discuss shortly). Forty-five years later, Allan M. Cormack, a physicist at Tufts University, partially “rediscovered” these concepts and applied them to CT. Cormack published his initial findings in 1963 and 1964 and showed how his results could be used to reconstruct cross-sectional images of the body from X-ray images taken in different angular directions. He gave the mathematical formulae needed for the reconstruction and built a CT prototype to show the practicality of his ideas. Working independently, electrical engineer Godfrey N. Hounsfield and his colleagues at EMI in London formulated a similar solution and built the first medical CT machine. Cormack and Hounsfield shared the 1979 Nobel Prize in Medicine for their contributions to medical uses of tomography.

The goal of X-ray computed tomography is to obtain a 3-D representation of the internal structure of an object by X-raying the object from many different directions. Imagine a traditional chest X-ray, obtained by placing the subject against an X-ray sensitive plate and “illuminating” the individual with an X-ray beam in the form of a cone. The X-ray plate would produce an image whose intensity at a point would be proportional to the X-ray energy impinging on that point after it passed through the subject. This image is the 2-D equivalent of the projections we discussed in the previous section. We could back-project this entire image and create a 3-D volume. Repeating this process through many angles and adding the backprojections would result in 3-D rendition of the structure of the chest cavity. Computed tomography attempts to get that same information (or localized parts of it) by generating slices

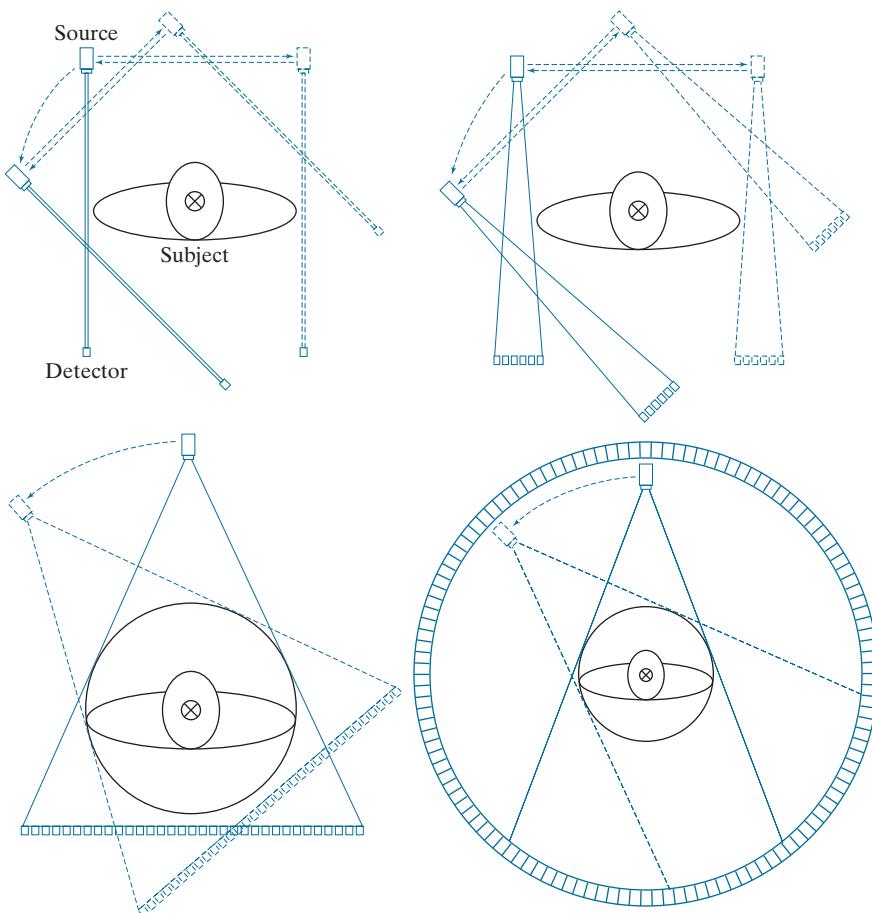
through the body. A 3-D representation then can be obtained by stacking the slices. A CT implementation is much more economical because the number of detectors required to obtain a high resolution slice is much smaller than the number of detectors needed to generate a complete 2-D projection of the same resolution. Computational burden and X-ray dosages are similarly reduced, making the 1-D projection CT a more practical approach.

First-generation (G1) CT scanners employ a “pencil” X-ray beam and a single detector, as Fig. 5.35(a) shows. For a given angle of rotation, the source/detector pair is translated incrementally along the linear direction shown. A projection (like the ones in Fig. 5.32), is generated by measuring the output of the detector at each increment of translation. After a complete linear translation, the source/detector assembly is rotated and the procedure is repeated to generate another projection at a different angle. The procedure is repeated for all desired angles in the range $[0^\circ, 180^\circ]$ to generate a complete set of projections images, from which one final cross-sectional image (a slice through the 3-D object) is obtained, as explained in the

a
b
c
d

FIGURE 5.35

Four generations of CT scanners. The dotted arrow lines indicate incremental linear motion. The dotted arrow arcs indicate incremental rotation. The cross-mark on the subject's head indicates linear motion perpendicular to the plane of the paper. The double arrows in (a) and (b) indicate that the source/detector unit is translated and then brought back into its original position.



previous section. A set of cross sectional images (slices) is generated by moving the subject incrementally (after each complete scan) past the source/detector plane (the cross-mark on the head of the subject indicates motion in a direction perpendicular to the plane of the source/detector pair). Stacking these images computationally produces a 3-D volume of a section of the body. G1 scanners are no longer manufactured for medical imaging, but, because they produce a parallel-ray beam (as in Fig. 5.32), their geometry is the one used predominantly for introducing the fundamentals of CT imaging, and serves as the starting point for deriving the equations necessary to implement image reconstruction from projections.

Second-generation (G2) CT scanners [Fig. 5.35(b)] operate on the same principle as G1 scanners, but the beam used is in the shape of a fan. This allows the use of multiple detectors, thus requiring fewer translations of the source/detector pair.

Third-generation (G3) scanners are a significant improvement over the earlier two generations of CT geometries. As Fig. 5.35(c) shows, G3 scanners employ a bank of detectors long enough (on the order of 1000 individual detectors) to cover the entire field of view of a wider beam. Consequently, each increment of angle produces an entire projection, eliminating the need to translate the source/detector pair, as in G1 and G2 scanners.

Fourth-generation (G4) scanners go a step further. By employing a circular ring of detectors (on the order of 5000 individual detectors), only the source has to rotate. The key advantage of G3 and G4 scanners is speed; key disadvantages are cost and greater X-ray scatter. The latter implies higher X-ray doses than G1 and G2 scanners to achieve comparable signal-to-noise characteristics.

Newer scanning modalities are beginning to be adopted. For example, *fifth-generation (G5) CT scanners*, also known as *electron beam computed tomography (EBCT) scanners*, eliminate all mechanical motion by employing electron beams controlled electromagnetically. By striking tungsten anodes that encircle the patient, these beams generate X-rays that are then shaped into a fan beam that passes through the patient and excites a ring of detectors, as in G4 scanners.

The conventional manner in which CT images are obtained is to keep the patient stationary during the scanning time required to generate one image. Scanning is then halted while the position of the patient is incremented in the direction perpendicular to the imaging plane, using a motorized table. The next image is then obtained and the procedure is repeated for the number of increments required to cover a specified section of the body. Although an image may be obtained in less than one second, there are procedures (e.g., abdominal and chest scans) that require patient to hold their breath during image acquisition. Completing these procedures for, say, 30 images, may require several minutes. An approach for which use is increasing is *helical CT*, sometimes referred to as *sixth-generation (G6) CT*. In this approach, a G3 or G4 scanner is configured using so-called *slip rings* that eliminate the need for electrical and signal cabling between the source/detectors and the processing unit. The source/detector pair then rotates continuously through 360° while the patient is moved at a constant speed along the axis perpendicular to the scan. The result is a continuous helical volume of data that is then processed to obtain individual slice images.

Seventh-generation (G7) scanners (also called *multislice CT scanners*) are emerging in which “thick” fan beams are used in conjunction with parallel banks of detectors to collect volumetric CT data simultaneously. That is, 3-D cross-sectional “slabs,” rather than single cross-sectional images are generated per X-ray burst. In addition to a significant increase in detail, this approach has the advantage that it utilizes X-ray tubes more economically, thus reducing cost and potentially reducing dosage.

In the following discussion, we develop the mathematical tools necessary for formulating image projection and reconstruction algorithms. Our focus is on the image-processing fundamentals that underpin all the CT approaches just discussed. Information regarding the mechanical and source/detector characteristics of CT systems is provided in the references cited at the end of the chapter.

PROJECTIONS AND THE RADON TRANSFORM

Next, we develop in detail the mathematics needed for image reconstruction in the context of X-ray computed tomography. The same basic principles apply to other CT imaging modalities, such as SPECT (single photon emission tomography), PET (positron emission tomography), MRI (magnetic resonance imaging), and some modalities of ultrasound imaging.

A straight line in Cartesian coordinates can be described either by its *slope-intercept* form, $y = ax + b$, or, as in Fig. 5.36, by its *normal* representation:

$$x \cos \theta + y \sin \theta = \rho \quad (5-100)$$

The projection of a *parallel-ray beam* can be modeled by a set of such lines, as Fig. 5.37 shows. An arbitrary *point* at coordinates (ρ_j, θ_k) in the projection profile is given by the raysum along the line $x \cos \theta_k + y \sin \theta_k = \rho_j$. Working with continuous quantities for the moment, the *raysum* is a line integral, given by

$$g(\rho_j, \theta_k) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta_k + y \sin \theta_k - \rho_j) dx dy \quad (5-101)$$

where we used the properties of the impulse, δ , discussed in Section 4.5. In other words, the right side of Eq. (5-101) is zero unless the argument of δ is zero, indicating

Throughout this section, we follow CT convention and place the origin of the xy -plane in the center, instead of at our customary top left corner (see Section 2.4). Both are right-handed coordinate systems, the only difference being that our image coordinate system has no negative axes. We can account for the difference with a simple translation of the origin, so both representations are interchangeable.

FIGURE 5.36

Normal representation of a line.

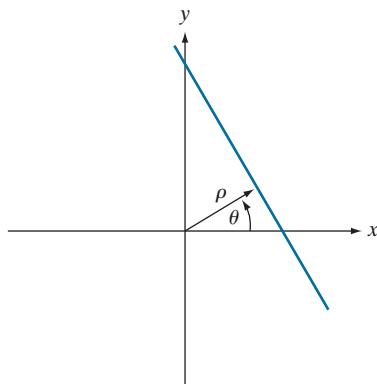
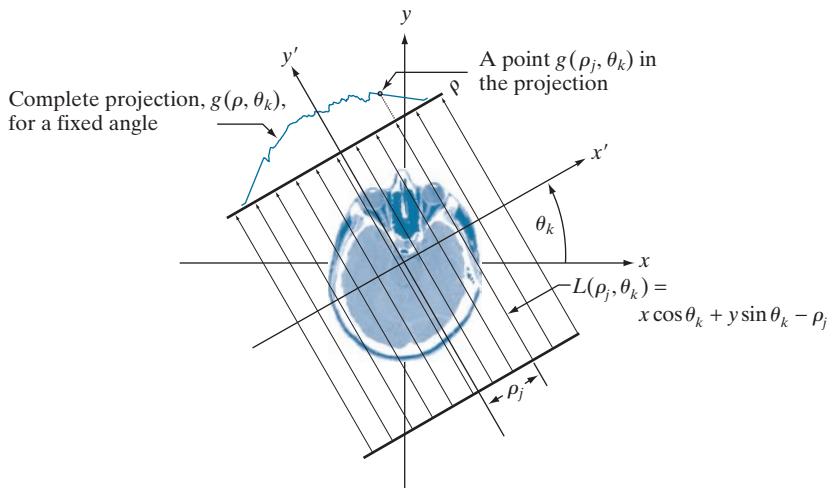


FIGURE 5.37

Geometry of a parallel-ray beam.



that the integral is computed only along the line $x \cos \theta_k + y \sin \theta_k = \rho_j$. If we consider all values of ρ and θ , the preceding equation generalizes to

$$g(\rho, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - \rho) dx dy \quad (5-102)$$

This equation, which gives the projection (line integral) of $f(x, y)$ along an arbitrary line in the xy -plane, is the *Radon transform* mentioned earlier. The notation $\mathfrak{R}\{f(x, y)\}$ or $\mathfrak{R}\{f\}$ is used sometimes in place of $g(\rho, \theta)$ in Eq. (5-102) to denote the Radon transform of $f(x, y)$, but the type of notation used in Eq. (5-102) is more customary. As will become evident in the discussion that follows, the Radon transform is the cornerstone of reconstruction from projections, with computed tomography being its principal application in the field of image processing.

In the discrete case,[†] the Radon transform of Eq. (5-102) becomes

$$g(\rho, \theta) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \delta(x \cos \theta + y \sin \theta - \rho) \quad (5-103)$$

where x , y , and ρ are now discrete variables, and M and N are the dimensions of a rectangular area over which the transform is applied. If we fix θ and allow ρ to vary, we see that (5-103) simply sums the pixels of $f(x, y)$ along the line defined by the specified values of these two parameters. Incrementing through all values of ρ

[†] In Chapter 4, we exercised great care in denoting continuous image coordinates by (t, z) and discrete coordinates by (x, y) . At that time, the distinction was important because we were developing basic concepts to take us from continuous to sampled quantities. In the present discussion, we go back and forth so many times between continuous and discrete coordinates that adhering to this convention is likely to generate unnecessary confusion. For this reason, and also to follow the published literature in this field (e.g., see Prince and Links [2006]), we let the context determine whether coordinates (x, y) are continuous or discrete. When they are continuous, you will see integrals; otherwise, you will see summations.

required to span the $M \times N$ area (with θ fixed) yields *one* projection. Changing θ and repeating this procedure yields another projection, and so forth. This is precisely how the projections in Figs. 5.32–5.34 were generated.

EXAMPLE 5.15: Using the Radon transform to obtain the projection of a circular region.

Before proceeding, we illustrate how to use the Radon transform to obtain an analytical expression for the projection of the circular object in Fig. 5.38(a):

$$f(x, y) = \begin{cases} A & x^2 + y^2 \leq r^2 \\ 0 & \text{otherwise} \end{cases}$$

where A is a constant and r is the radius of the object. We assume that the circle is centered on the origin of the xy -plane. Because the object is circularly symmetric, its projections are the same for all angles, so all we have to do is obtain the projection for $\theta = 0^\circ$. Equation (5-102) then becomes

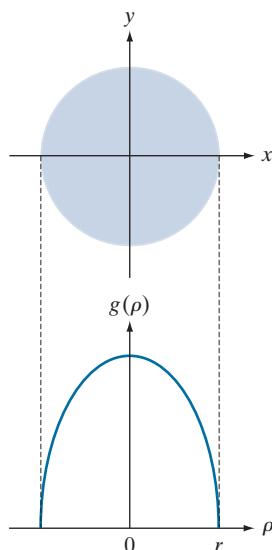
$$\begin{aligned} g(\rho, \theta) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x - \rho) dx dy \\ &= \int_{-\infty}^{\infty} f(\rho, y) dy \end{aligned}$$

where the second expression follows from Eq. (4-13). As noted earlier, this is a line integral (along the line $L(\rho, 0)$ in this case). Also, note that $g(\rho, \theta) = 0$ when $|\rho| > r$. When $|\rho| \leq r$ the integral is evaluated from $y = -(r^2 - \rho^2)^{1/2}$ to $y = (r^2 - \rho^2)^{1/2}$. Therefore,

a
b

FIGURE 5.38

(a) A disk and,
(b) a plot of its Radon transform, derived analytically. Here we were able to plot the transform because it depends only on one variable. When g depends on both ρ and θ , the Radon transform becomes an image whose axes are ρ and θ , and the intensity of a pixel is proportional to the value of g at the location of that pixel.



$$\begin{aligned} g(\rho, \theta) &= \int_{-\sqrt{r^2 - \rho^2}}^{\sqrt{r^2 - \rho^2}} f(\rho, y) dy \\ &= \int_{-\sqrt{r^2 - \rho^2}}^{\sqrt{r^2 - \rho^2}} A dy \end{aligned}$$

Carrying out the integration yields

$$g(\rho, \theta) = g(\rho) = \begin{cases} 2A\sqrt{r^2 - \rho^2} & |\rho| \leq r \\ 0 & \text{otherwise} \end{cases}$$

where we used the fact that $g(\rho, \theta) = 0$ when $|\rho| > r$. Figure 5.38(b) shows a plot of this result. Note that $g(\rho, \theta) = g(\rho)$; that is, g is independent of θ because the object is symmetric about the origin.

To generate arrays with rows of the same size, the minimum dimension of the ρ -axis in sinograms corresponds to the largest dimension encountered during projection. For example, the minimum size of a sinogram of a square of size $M \times M$ obtained using increments of 1° is $180 \times Q$ where Q is the smallest integer greater than $\sqrt{2} M$.

When the Radon transform, $g(\rho, \theta)$, is displayed as an image with ρ and θ as rectilinear coordinates, the result is called a *sinogram*, similar in concept to displaying the Fourier spectrum. Like the Fourier transform, a sinogram contains the data necessary to reconstruct $f(x, y)$. Unlike the Fourier transform, however, $g(\rho, \theta)$ is always a real function. As is the case with displays of the Fourier spectrum, sinograms can be readily interpreted for simple regions, but become increasingly difficult to “read” as the region being projected becomes more complex. For example, Fig. 5.39(b) is the sinogram of the rectangle on the left. The vertical and horizontal axes correspond to θ and ρ , respectively. Thus, the bottom row is the projection of the rectangle in the horizontal direction (i.e., $\theta = 0^\circ$), and the middle row is the projection in the vertical direction ($\theta = 90^\circ$). The fact that the nonzero portion of the bottom row is smaller than the nonzero portion of the middle row tells us that the object is narrower in the horizontal direction. The fact that the sinogram is symmetric in both directions about the center of the image tells us that we are dealing with an object that is symmetric and parallel to the x and y axes. Finally, the sinogram is smooth, indicating that the object has a uniform intensity. Other than these types of general observations, we cannot say much more about this sinogram.

Figure 5.39(c) is an image of the *Shepp-Logan phantom* (Shepp and Logan [1974]), a widely used synthetic image designed to simulate the absorption of major areas of the brain, including small tumors. The sinogram of this image is considerably more difficult to interpret, as Fig. 5.39(d) shows. We still can infer some symmetry properties, but that is about all we can say. Visual analyses of sinograms are of limited practical use, but they can be helpful in tasks such as algorithm development.

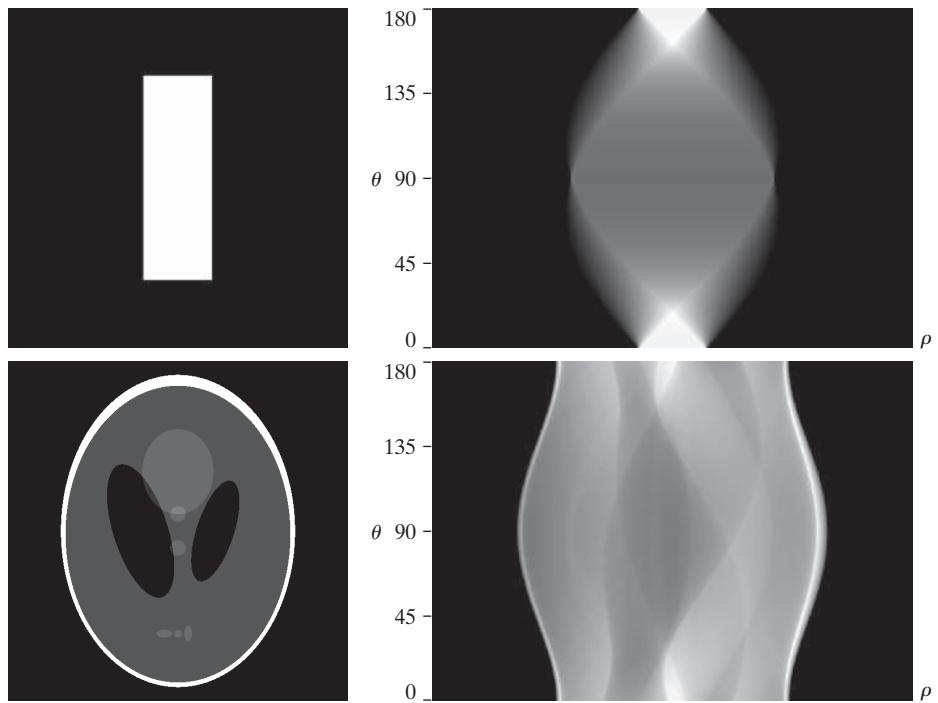
BACKPROJECTIONS

To obtain a formal expression for a backprojected image from the Radon transform, let us begin with a *single* point, $g(\rho_j, \theta_k)$, of the complete projection, $g(\rho, \theta_k)$, for a fixed value of rotation, θ_k (see Fig. 5.37). Forming part of an image by backprojecting this single point is nothing more than copying the line $L(\rho_j, \theta_k)$ onto the image,

a	b
c	d

FIGURE 5.39

Two images and their sinograms (Radon transforms). Each row of a sinogram is a projection along the corresponding angle on the vertical axis. (Note that the horizontal axis of the sinograms are values of ρ .) Image (c) is called the *Shepp-Logan phantom*. In its original form, the contrast of the phantom is quite low. It is shown enhanced here to facilitate viewing.



where the value (intensity) of each point in that line is $g(\rho_j, \theta_k)$. Repeating this process of all values of ρ_j in the projected signal (but keeping the value of θ fixed at θ_k) results in the following expression:

$$\begin{aligned} f_{\theta_k}(x, y) &= g(\rho, \theta_k) \\ &= g(x \cos \theta_k + y \sin \theta_k, \theta_k) \end{aligned}$$

for the image due to backprojecting the projection obtained with a fixed angle, θ_k , as in Fig. 5.32(b). This equation holds for an arbitrary value of θ_k , so we may write in general that the image formed from a *single* backprojection obtained at an angle θ is given by

$$f_\theta(x, y) = g(x \cos \theta + y \sin \theta, \theta) \quad (5-104)$$

We form the final image by integrating over all the backprojected images:

$$f(x, y) = \int_0^\pi f_\theta(x, y) d\theta \quad (5-105)$$

In the discrete case, the integral becomes a sum of all the backprojected images:

$$f(x, y) = \sum_{\theta=0}^{\pi} f_\theta(x, y) \quad (5-106)$$

where, x , y , and θ are now discrete quantities. As mentioned earlier, the projections at 0° and 180° are mirror images of each other, so the summations are carried out to the last angle increment before 180° . For example, if 0.5° increments are being used, the summation is from 0° to 179.5° in half-degree increments. A backprojected image formed in the manner just described sometimes is referred to as a *laminogram*. It is understood implicitly that a laminogram is only an approximation to the image from which the projections were generated, a fact that is illustrated in the following example.

EXAMPLE 5.16: Obtaining backprojected images from sinograms.

Equation (5-106) was used to generate the backprojected images in Figs. 5.32 through 5.34, from projections obtained with Eq. (5-103). Similarly, these equations were used to generate Figs. 5.40(a) and (b), which show the backprojected images corresponding to the sinograms in Figs. 5.39(b) and (d), respectively. As with the earlier figures, we note a significant amount of blurring, so it is obvious that a straight use of Eqs. (5-103) and (5-106) will not yield acceptable results. Early, experimental CT systems were based on these equations. However, as you will see later in our discussion, significant improvements in reconstruction are possible by reformulating the backprojection approach.

THE FOURIER-SLICE THEOREM

In this section, we derive a fundamental equation that establishes a relationship between the 1-D Fourier transform of a projection and the 2-D Fourier transform of the region from which the projection was obtained. This relationship is the basis for reconstruction methods capable of dealing with the blurring problems we have encountered thus far.

The 1-D Fourier transform of a projection with respect to ρ is

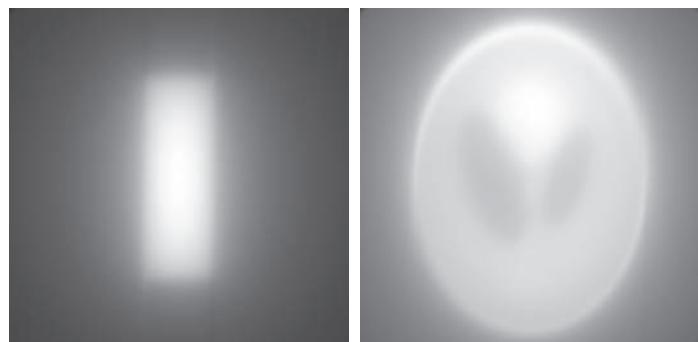
This equation has the same form as Eq. (4-20).

$$G(\omega, \theta) = \int_{-\infty}^{\infty} g(\rho, \theta) e^{-j2\pi\omega\rho} d\rho \quad (5-107)$$

a b

FIGURE 5.40

Backprojections of the sinograms in Fig. 5.39.



where ω is the frequency variable, and it is understood that this expression is based on a fixed value of θ . Substituting Eq. (5-102) for $g(\rho, \theta)$ we obtain

$$\begin{aligned} G(\omega, \theta) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos \theta + y \sin \theta - \rho) e^{-j2\pi\omega\rho} dx dy d\rho \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \left[\int_{-\infty}^{\infty} \delta(x \cos \theta + y \sin \theta - \rho) e^{-j2\pi\omega\rho} d\rho \right] dx dy \quad (5-108) \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi\omega(x \cos \theta + y \sin \theta)} dx dy \end{aligned}$$

where the last step follows from the sifting property of the impulse discussed in Chapter 4. By letting $u = \omega \cos \theta$ and $v = \omega \sin \theta$, we can write Eq. (5-108) as

$$G(\omega, \theta) = \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux + vy)} dx dy \right]_{u=\omega \cos \theta; v=\omega \sin \theta} \quad (5-109)$$

We recognize this expression as the 2-D Fourier transform of $f(x, y)$ [see Eq. (4-59)] evaluated at the values of u and v indicated. That is,

$$\begin{aligned} G(\omega, \theta) &= [F(u, v)]_{u=\omega \cos \theta; v=\omega \sin \theta} \quad (5-110) \\ &= F(\omega \cos \theta, \omega \sin \theta) \end{aligned}$$

where, as usual, $F(u, v)$ denotes the 2-D Fourier transform of $f(x, y)$.

The result in Eq. (5-110) is known as the *Fourier-slice theorem* (or the *projection-slice theorem*). It states that the Fourier transform of a projection is a *slice* of the 2-D Fourier transform of the region from which the projection was obtained. The reason for this terminology can be explained with the aid of Fig. 5.41. As this figure shows, the 1-D Fourier transform of an arbitrary projection is obtained by extracting the values of $F(u, v)$ along a line oriented at the same angle as the angle used in generating the projection.

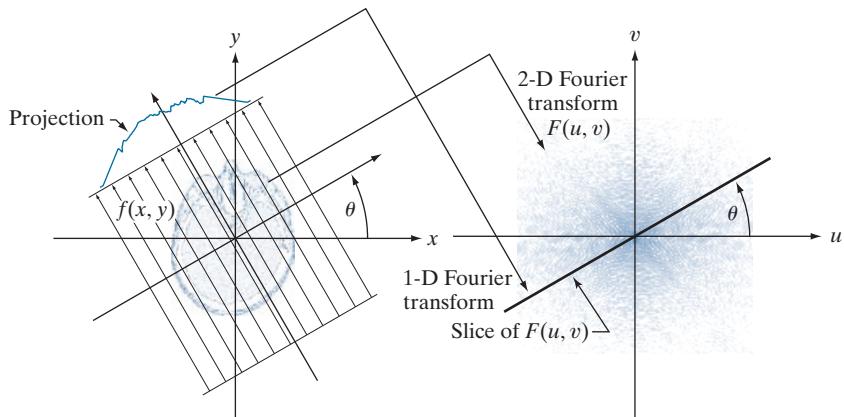
In principle, we could obtain $f(x, y)$ simply by obtaining the inverse Fourier transform of $F(u, v)$. However, this is expensive computationally, as it involves obtained the inverse of a 2-D transform. The approach discussed in the following section is much more efficient.

RECONSTRUCTION USING PARALLEL-BEAM FILTERED BACKPROJECTIONS

As we saw in Figs. 5.33, 5.34, and 5.40, obtaining backprojections directly yields unacceptably blurred results. Fortunately, there is a straightforward solution to this problem based simply on filtering the projections before computing the backprojections. From Eq. (4-60), the 2-D inverse Fourier transform of $F(u, v)$ is

FIGURE 5.41

Illustration of the Fourier-slice theorem. The 1-D Fourier transform of a projection is a slice of the 2-D Fourier transform of the region from which the projection was obtained. Note the correspondence of the angle θ in the two figures.



$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux + vy)} du dv \quad (5-111)$$

If, as in Eqs. (5-109) and (5-110), we let $u = \omega \cos \theta$ and $v = \omega \sin \theta$, then the differentials become $dudv = \omega d\omega d\theta$, and we can express Eq. (5-111) in polar coordinates:

$$f(x, y) = \int_0^{2\pi} \int_0^{\infty} F(\omega \cos \theta, \omega \sin \theta) e^{j2\pi\omega(x \cos \theta + y \sin \theta)} \omega d\omega d\theta \quad (5-112)$$

Then, using the Fourier slice theorem,

$$f(x, y) = \int_0^{2\pi} \int_0^{\infty} G(\omega, \theta) e^{j2\pi\omega(x \cos \theta + y \sin \theta)} \omega d\omega d\theta \quad (5-113)$$

By splitting this integral into two expressions, one for θ in the range 0° to 180° and the other in the range 180° to 360° , and using the fact that $G(\omega, \theta + 180^\circ) = G(-\omega, \theta)$ (see Problem 5.46), we can express Eq. (5-113) as

$$f(x, y) = \int_0^{\pi} \int_{-\infty}^{\infty} |\omega| G(\omega, \theta) e^{j2\pi\omega(x \cos \theta + y \sin \theta)} d\omega d\theta \quad (5-114)$$

The term $x \cos \theta + y \sin \theta$ is a constant with respect to ω , and we recognize it as ρ from Eq. (5-100). Therefore, we can write Eq. (5-114) as

$$f(x, y) = \int_0^{\pi} \left[\int_{-\infty}^{\infty} |\omega| G(\omega, \theta) e^{j2\pi\omega\rho} d\omega \right]_{\rho=x \cos \theta + y \sin \theta} d\theta \quad (5-115)$$

The ramp filter often is referred to as the *Ram-Lak filter*, after Ramachandran and Lakshminarayanan [1971] who generally are credited with having been first to suggest it.

The inner expression is in the form of an *inverse 1-D Fourier transform* [see Eq. (4-21)], with the added term $|\omega|$ which, based on the discussion in Section 4.7, we recognize as a 1-D filter transfer function. Observe that $|\omega|$ is a *ramp* function [see Fig. 5.42(a)]. This function is not integrable because its amplitude extends to $+\infty$ in both directions, so the inverse Fourier transform is undefined. Theoretically, this is handled by methods such as using so-called *generalized delta functions*. In practice, the approach is to *window* the ramp so that it becomes zero outside of a defined frequency interval. That is, a window *band-limits* the ramp filter transfer function.

The simplest approach to band-limit a function is to use a box in the frequency domain. However, as we saw in Fig. 4.4, a box has undesirable ringing properties. This is demonstrated by Figs. 5.42(b) and (c). The former shows a plot of the ramp transfer function after it was band-limited by a box window, and the latter shows its spatial domain representation, obtained by computing its inverse Fourier transform. As expected, the resulting windowed filter exhibits noticeable ringing in the spatial domain. We know from Chapter 4 that filtering in the frequency domain is equivalent to convolution in the spatial domain, so spatial filtering with a function that exhibits ringing will produce a result corrupted by ringing also. Windowing with a smooth function helps this situation. An M -point discrete window function used frequently for implementations with the 1-D FFT is given by

$$H(\omega) = \begin{cases} c + (c - 1)\cos \frac{2\pi\omega}{M} & 0 \leq \omega \leq (M - 1) \\ 0 & \text{otherwise} \end{cases} \quad (5-116)$$

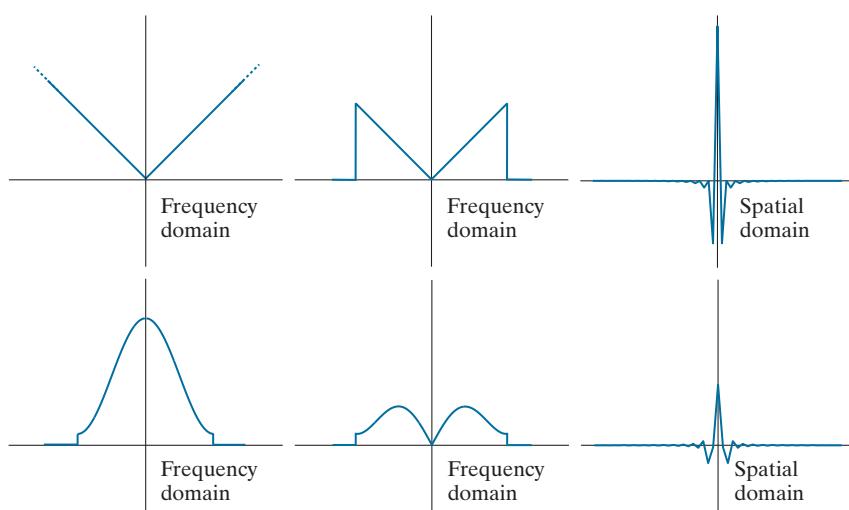
Sometimes the Hann window is referred to as the *Hanning window* in analogy to the Hamming window. However, this terminology is incorrect and is a frequent source of confusion.

When $c = 0.54$, this function is called the *Hamming window* (named after Richard Hamming) and, when $c = 0.5$ it is called the *Hann window* (named after Julius von Hann). The key difference between the Hamming and Hann windows is that the

a	b	c
d	e	f

FIGURE 5.42

- (a) Frequency domain ramp filter transfer function.
- (b) Function after band-limiting it with a box filter.
- (c) Spatial domain representation.
- (d) Hamming windowing function.
- (e) Windowed ramp filter, formed as the product of (b) and (d).
- (f) Spatial representation of the product. (Note the decrease in ringing.)



end points are zero in the latter. The difference between the two generally is visually imperceptible in image processing applications.

Figure 5.42(d) is a plot of the Hamming window, and Fig. 5.42(e) shows the product of this window and the band-limited ramp filter transfer function in Fig. 5.42(b). Figure 5.42(f) shows the representation of the product in the spatial domain, obtained as usual by computing the inverse FFT. It is evident by comparing this figure and Fig. 5.42(c) that ringing was reduced in the windowed ramp (the ratios of the peak to trough in Figs. 5.42(c) and (f) are 2.5 and 3.4, respectively). On the other hand, because the width of the central lobe in Fig. 5.42(f) is slightly wider than in Fig. 5.42(c), we would expect backprojections based on using a Hamming window to have less ringing, but be slightly more blurred. As Example 5.17 below shows, this is indeed the case.

Recall from Eq. (5-107) that $G(\omega, \theta)$ is the 1-D Fourier transform of $g(\rho, \theta)$, which is a *single* projection obtained at a fixed angle, θ . Equation (5-115) states that the *complete*, backprojected image $f(x, y)$ is obtained as follows:

1. Compute the 1-D Fourier transform of each projection.
2. Multiply each 1-D Fourier transform by the filter transfer function $| \omega |$ which, as explained above, has been multiplied by a suitable (e.g., Hamming) window.
3. Obtain the inverse 1-D Fourier transform of each resulting filtered transform.
4. Integrate (sum) all the 1-D inverse transforms from Step 3.

Because a filter function is used, this image reconstruction approach is appropriately called *filtered backprojection*. In practice, the data are discrete, so all frequency domain computations are carried out using a 1-D FFT algorithm, and filtering is implemented using the same basic procedure explained in Chapter 4 for 2-D functions. Alternatively, we can implement filtering in the spatial domain using convolution, as explained later.

The preceding discussion addresses the windowing aspects of filtered backprojections. As with any sampled data system, we also need to be concerned about sampling rates. We know from Chapter 4 that the selection of sampling rates has a profound influence on image processing results. In the present discussion, there are two sampling considerations. The first is the number of rays used, which determines the number of samples in each projection. The second is the number of rotation angle increments, which determines the number of reconstructed images (whose sum yields the final image). Under-sampling results in aliasing which, as we saw in Chapter 4, can manifest itself as artifacts in the image, such as streaks. We address CT sampling issues in more detail later in our discussion.

EXAMPLE 5.17: Image reconstruction using filtered backprojections.

The focus of this example is to show reconstruction using filtered backprojections, first with a box-limited ramp transfer function and then using a ramp limited by a Hamming window. These filtered backprojections are compared against the results of “raw” backprojections from Fig. 5.40. In order to focus on the difference due only to filtering, the results in this example were generated with 0.5° increments of rotation, the same we used to generate Fig. 5.40. The separation between rays was one pixel

in both cases. The images in both examples are of size 600×600 pixels, so the length of the diagonal is $\sqrt{2} \times 600 \approx 849$. Consequently, 849 rays were used to provide coverage of the entire region when the angle of rotation was 45° and 135° .

Figure 5.43(a) shows the rectangle reconstructed using a ramp function band-limited by a box. The most vivid feature of this result is the absence of any visually detectable blurring. However, as expected, ringing is present, visible as faint lines, especially around the corners of the rectangle. These lines are more visible in the zoomed section in Fig. 5.43(c). Using a Hamming window on the ramp helped considerably with the ringing problem, at the expense of slight blurring, as Figs. 5.43(b) and (d) show. The improvements (even with the box-windowed ramp) over Fig. 5.40(a) are evident. The phantom image does not have transitions that are as sharp and prominent as the rectangle so ringing, even with the box-windowed ramp, is imperceptible in this case, as you can see in Fig. 5.44(a). Using a Hamming window resulted in a slightly smoother image, as Fig. 5.44(b) shows. Both of these results are considerable improvements over Fig. 5.40(b), illustrating again the significant advantage inherent in the filtered backprojection approach.

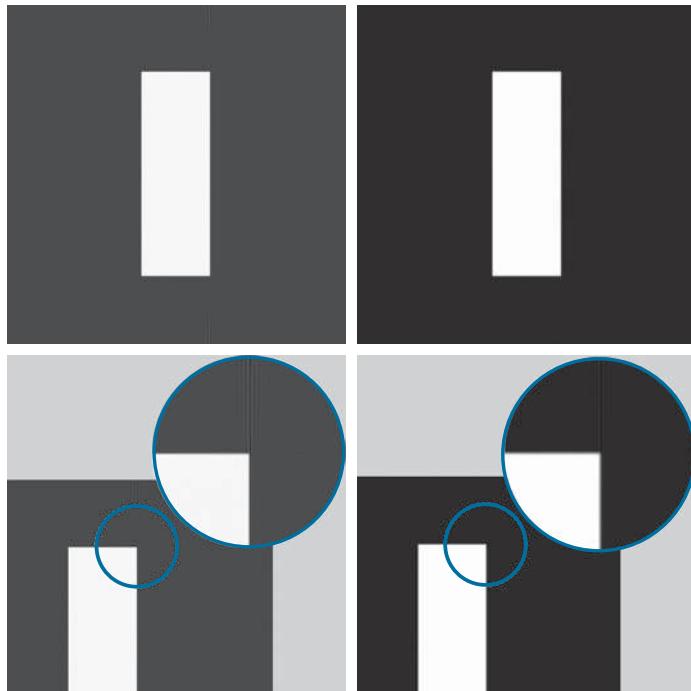
In most applications of CT (especially in medicine), artifacts such as ringing are a serious concern, so significant effort is devoted to minimizing them. Tuning the filtering algorithms and, as explained earlier, using a large number of detectors are among the design considerations that help reduce these effects.

The preceding discussion is based on obtaining filtered backprojections via an FFT implementation. However, we know from the convolution theorem in Chapter 4 that equivalent results can be obtained using spatial convolution. In particular, note

a	b
c	d

FIGURE 5.43

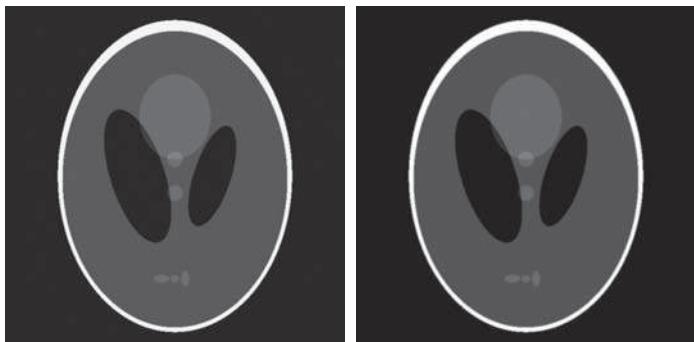
Filtered backprojections of the rectangle using (a) a ramp filter, and (b) a Hamming windowed ramp filter. The second row shows zoomed details of the images in the first row. Compare with Fig. 5.40(a).



a b

FIGURE 5.44

Filtered backprojections of the head phantom using (a) a ramp filter, and (b) a Hamming windowed ramp filter. Compare with Fig. 5.40(b)



that the term inside the brackets in Eq. (5-115) is the inverse Fourier transform of the product of two frequency domain functions which, according to the convolution theorem, we know to be equal to the convolution of the spatial representations (inverse Fourier transforms) of these two functions. In other words, letting $s(\rho)$ denote the inverse Fourier transform of $|\omega|^\dagger$, we write Eq. (5-115) as

$$\begin{aligned}
 f(x, y) &= \int_0^\pi \left[\int_{-\infty}^{\infty} |\omega| G(\omega, \theta) e^{j2\pi\omega\rho} d\omega \right]_{\rho=x\cos\theta + y\sin\theta} d\theta \\
 &= \int_0^\pi [s(\rho) \star g(\rho, \theta)]_{\rho=x\cos\theta + y\sin\theta} d\theta \\
 &= \int_0^\pi \left[\int_{-\infty}^{\infty} g(\rho, \theta) s(x\cos\theta + y\sin\theta - \rho) d\rho \right] d\theta
 \end{aligned} \tag{5-117}$$

where, as in Chapter 4, “ \star ” denotes convolution. The second line follows from the first for the reasons explained in the previous paragraph. The third line (including the $-\rho$) follows from the definition of convolution in Eq. (4-24).

The last two lines of Eq. (5-117) say the same thing: individual backprojections at an angle θ can be obtained by convolving the corresponding projection, $g(\rho, \theta)$, and the inverse Fourier transform of the ramp filter transfer function, $s(\rho)$. As before, the complete backprojected image is obtained by integrating (summing) all the individual backprojected images. With the exception of roundoff differences in computation, the results of using convolution will be identical to the results using the FFT. In actual CT implementations, convolution generally turns out to be more efficient computationally, so most modern CT systems use this approach. The Fourier transform does play a central role in theoretical formulations and algorithm development (for example, CT image processing in MATLAB is based on the FFT). Also, we note that there is no need to store all the backprojected images during reconstruction.

[†]If a windowing function, such as a Hamming window, is used, then the inverse Fourier transform is performed on the windowed ramp.

Instead, a single running sum is updated with the latest backprojected image. At the end of the procedure, the running sum will equal the sum total of all the backprojections.

Finally, we point out that, because the ramp filter (even when it is windowed) zeros the dc term in the frequency domain, each backprojection image will have zero average value (see Fig. 4.29). This means that the pixels in each backprojection image will have negative and positive values. When all the backprojections are added to form the final image, some negative locations may become positive and the average value may not be zero, but typically, the final image will still have negative pixels.

There are several ways to handle this problem. The simplest approach, when there is no knowledge regarding what the average values should be, is to accept the fact that negative values are inherent in the approach and scale the result using the procedure described in Eqs. (2-31) and (2-32). This is the approach followed in this section. When knowledge about what a “typical” average value should be is available, that value can be added to the filter transfer function in the frequency domain, thus offsetting the ramp and preventing zeroing the dc term [see Fig. 4.30(c)]. When working in the spatial domain with convolution, the very act of truncating the length of the spatial filter kernel (inverse Fourier transform of the ramp) prevents it from having a zero average value, thus avoiding the zeroing problem altogether.

RECONSTRUCTION USING FAN-BEAM FILTERED BACKPROJECTIONS

The discussion thus far has centered on parallel beams. Because of its simplicity and intuitiveness, this is the imaging geometry used traditionally to introduce computed tomography. However, more modern CT systems use a fan-beam geometry (see Fig. 5.35), which is the topic of the following discussion.

Figure 5.45 shows a basic fan-beam imaging geometry in which the detectors are arranged on a circular arc and the angular increments of the source are assumed to be equal. Let $p(\alpha, \beta)$ denote a fan-beam projection, where α is the angular position of a particular detector measured with respect to the *center ray*, and β is the angular displacement of the source, measured with respect to the *y-axis*, as shown in the figure. We also note in Fig. 5.45 that a ray in the fan beam can be represented as a line, $L(\rho, \theta)$, in normal form, which is the approach we used to represent a ray in the parallel-beam imaging geometry discussed earlier. This allows us to utilize parallel-beam results as the starting point for deriving the corresponding equations for the fan-beam geometry. We proceed to show this by deriving the fan-beam filtered backprojection based on convolution.[†]

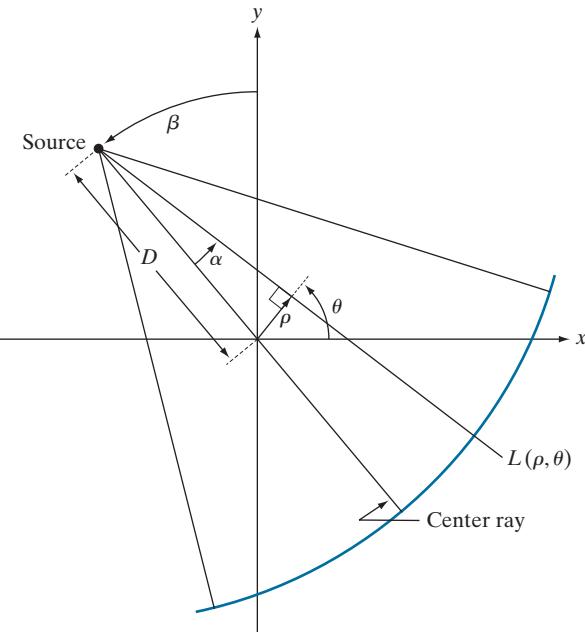
We begin by noticing in Fig. 5.45 that the parameters of line $L(\rho, \theta)$ are related to the parameters of a fan-beam ray by

$$\theta = \beta + \alpha \quad (5-118)$$

[†]The Fourier-slice theorem was derived for a parallel-beam geometry and is not directly applicable to fan beams. However, Eqs. (5-118) and (5-119) provide the basis for converting a fan-beam geometry to a parallel-beam geometry, thus allowing us to use the filtered parallel backprojection approach developed in the previous section, for which the slice theorem is applicable. We will discuss this in more detail at the end of this section.

FIGURE 5.45

Basic fan-beam geometry. The line passing through the center of the source and the origin (assumed here to be the center of rotation of the source) is called the *center ray*.



and

$$\rho = D \sin \alpha \quad (5-119)$$

where D is the distance from the center of the source to the origin of the xy -plane.

The convolution backprojection formula for the parallel-beam imaging geometry is given by Eq. (5-117). Without loss of generality, suppose that we focus attention on objects that are encompassed within a circular area of radius T about the origin of the xy -plane. Then $g(\rho, \theta) = 0$ for $|\rho| > T$ and Eq. (5-117) becomes

$$f(x, y) = \frac{1}{2} \int_0^{2\pi} \int_{-T}^T g(\rho, \theta) s(x \cos \theta + y \sin \theta - \rho) d\rho d\theta \quad (5-120)$$

where we used the fact mentioned earlier that projections 180° apart are mirror images of each other. In this way, the limits of the outer integral in Eq. (5-120) are made to span a full circle, as required by a fan-beam arrangement in which the detectors are arranged in a circle.

We are interested in integrating with respect to α and β . To do this, we change to polar coordinates, (r, φ) . That is, we let $x = r \cos \varphi$ and $y = r \sin \varphi$, from which it follows that

$$\begin{aligned} x \cos \theta + y \sin \theta &= r \cos \varphi \cos \theta + r \sin \varphi \sin \theta \\ &= r \cos(\theta - \varphi) \end{aligned} \quad (5-121)$$

Using this result we can express Eq. (5-120) as

$$f(x, y) = \frac{1}{2} \int_0^{2\pi} \int_{-T}^T g(\rho, \theta) s(r \cos(\theta - \varphi) - \rho) d\rho d\theta \quad (5-122)$$

This expression is nothing more than the parallel-beam reconstruction formula written in polar coordinates. However, integration still is with respect to ρ and θ . To integrate with respect to α and β requires a transformation of coordinates using Eqs. (5-118) and (5-119):

$$\begin{aligned} f(r, \varphi) &= \frac{1}{2} \int_{-\alpha}^{2\pi - \alpha} \int_{\sin^{-1}(-T/D)}^{\sin^{-1}(T/D)} g(D \sin \alpha, \alpha + \beta) \\ &\quad s(r \cos(\beta + \alpha - \varphi) - D \sin \alpha) D \cos \alpha d\alpha d\beta \end{aligned} \quad (5-123)$$

where we used $d\rho d\theta = D \cos \alpha d\alpha d\beta$ [see the explanation of Eq. (5-112)].

This equation can be simplified further. First, note that the limits $-\alpha$ to $2\pi - \alpha$ for variable β span the entire range of 360° . Because all functions of β are periodic with period 2π , the limits of the outer integral can be replaced by 0 and 2π , respectively. The term $\sin^{-1}(T/D)$ has a maximum value, α_m , corresponding to $|\rho| > T$, beyond which $g = 0$ (see Fig. 5.46), so we can replace the limits of the inner integral by $-\alpha_m$ and α_m , respectively. Finally, consider the line $L(\rho, \theta)$ in Fig. 5.45. A raysum of a fan beam along this line must equal the raysum of a parallel beam along the same line. This follows from the fact that a raysum is a sum of all values along a line, so the result must be the same for a given ray, regardless of the coordinate system in which it is expressed. This is true of any raysum for corresponding values of (α, β) and (ρ, θ) . Thus, letting $p(\alpha, \beta)$ denote a fan-beam projection, it follows that $p(\alpha, \beta) = g(\rho, \theta)$ and, from Eqs. (5-118) and (5-119), that $p(\alpha, \beta) = g(D \sin \alpha, \alpha + \beta)$. Incorporating these observations into Eq. (5-123) results in the expression

$$f(r, \varphi) = \frac{1}{2} \int_0^{2\pi} \int_{-\alpha_m}^{\alpha_m} p(\alpha, \beta) s[r \cos(\beta + \alpha - \varphi) - D \sin \alpha] D \cos \alpha d\alpha d\beta \quad (5-124)$$

This is the fundamental fan-beam *reconstruction formula* based on filtered backprojections.

Equation (5-124) can be manipulated further to put it in a more familiar convolution form. With reference to Fig. 5.47, it can be shown (see Problem 5.47) that

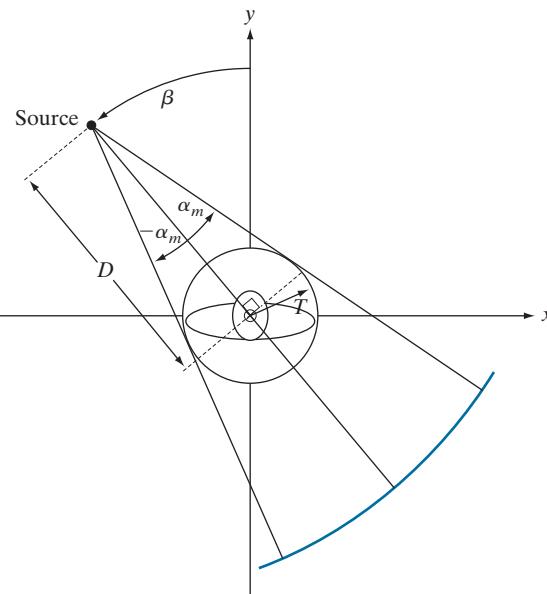
$$r \cos(\beta + \alpha - \varphi) - D \sin \alpha = R \sin(\alpha' - \alpha) \quad (5-125)$$

where R is the distance from the source to an arbitrary point in a fan ray, and α' is the angle between this ray and the center ray. Note that R and α' are determined by the values of r , φ , and β . Substituting Eq. (5-125) into Eq. (5-124) yields

$$f(r, \varphi) = \frac{1}{2} \int_0^{2\pi} \int_{-\alpha_m}^{\alpha_m} p(\alpha, \beta) s(R \sin[\alpha' - \alpha]) D \cos \alpha d\alpha d\beta \quad (5-126)$$

FIGURE 5.46

Maximum value of α needed to encompass a region of interest.



It can be shown (see Problem 5.48) that

$$s(R \sin \alpha) = \left[\frac{\alpha}{R \sin \alpha} \right]^2 s(\alpha) \quad (5-127)$$

Using this expression, we can write Eq. (5-126) as

$$f(r, \varphi) = \frac{1}{2} \int_0^{2\pi} \frac{1}{R^2} \left[\int_{-\alpha_m}^{\alpha_m} q(\alpha, \beta) h(\alpha' - \alpha) d\alpha \right] d\beta \quad (5-128)$$

where

$$h(\alpha) = \frac{1}{2} \left[\frac{\alpha}{\sin \alpha} \right]^2 s(\alpha) \quad (5-129)$$

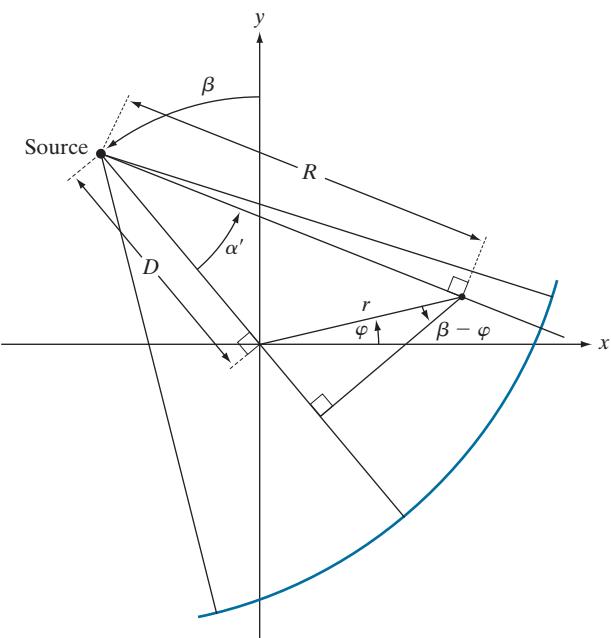
and

$$q(\alpha, \beta) = p(\alpha, \beta) D \cos \alpha \quad (5-130)$$

We recognize the inner integral in Eq. (5-128) as a convolution expression, thus showing that the image reconstruction formula in Eq. (5-124) can be implemented as the convolution of functions $q(\alpha, \beta)$ and $h(\alpha)$. Unlike the reconstruction formula for parallel projections, reconstruction based on fan-beam projections involves a term $1/R^2$, which is a weighting factor inversely proportional to the distance from the source. The computational details of implementing Eq. (5-128) are beyond the scope of the present discussion (see Kak and Slaney [2001] for a detailed treatment of this subject).

FIGURE 5.47

Polar representation of an arbitrary point on a ray of a fan beam.



Instead of implementing Eq. (5-128) directly, an approach used often, particularly in software simulations, is to: (1) convert a fan-beam geometry to a parallel-beam geometry using Eqs. (5-118) and (5-119), and (2) use the parallel-beam reconstruction approach developed earlier. We conclude this section with an example of how to do this. As noted earlier, a fan-beam projection, p , taken at angle β has a corresponding parallel-beam projection, g , taken at a corresponding angle θ and, therefore,

$$\begin{aligned} p(\alpha, \beta) &= g(\rho, \theta) \\ &= g(D \sin \alpha, \alpha + \beta) \end{aligned} \quad (5-131)$$

where the last line follows from Eqs. (5-118) and (5-119).

Let $\Delta\beta$ denote the angular increment between successive fan-beam projections, and let $\Delta\alpha$ be the angular increment between rays, which determines the number of samples in each projection. We impose the restriction that

$$\Delta\beta = \Delta\alpha = \gamma \quad (5-132)$$

Then, $\beta = m\gamma$ and $\alpha = n\gamma$ for some integer values of m and n , and we can write Eq. (5-131) as

$$p(n\gamma, m\gamma) = g(D \sin n\gamma, (m + n)\gamma) \quad (5-133)$$

This equation indicates that the n th ray in the m th radial projection is equal to the n th ray in the $(m + n)$ th parallel projection. The $D \sin n\gamma$ term on the right side of Eq. (5-133) implies that parallel projections converted from fan-beam projections

are not sampled uniformly, an issue that can lead to blurring, ringing, and aliasing artifacts if the sampling intervals $\Delta\alpha$ and $\Delta\beta$ are too coarse, as the following example illustrates.

EXAMPLE 5.18: Image reconstruction using filtered fan backprojections.

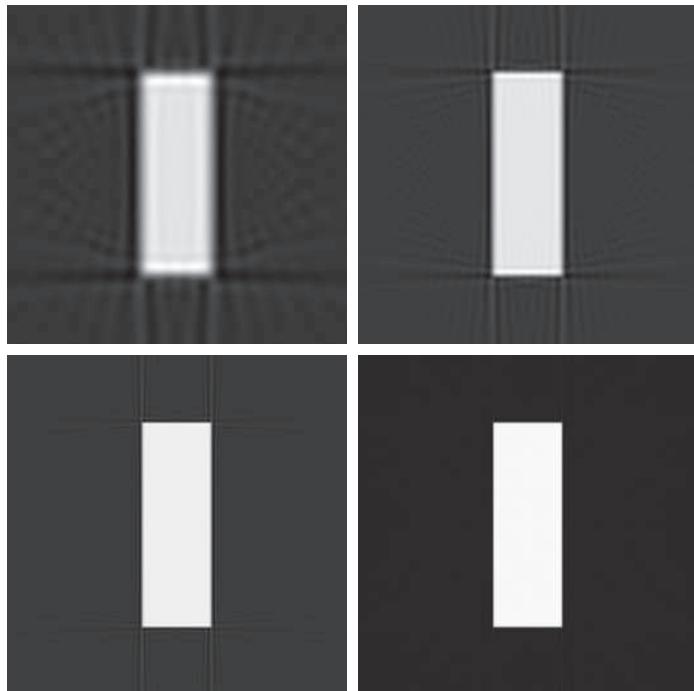
Figure 5.48(a) shows the results of: (1) generating fan projections of the rectangle image with $\Delta\alpha = \Delta\beta = 1^\circ$, (2) converting each fan ray to the corresponding parallel ray using Eq. (5-133), and (3) using the filtered backprojection approach developed earlier for parallel rays. Figures 5.48(b) through (d) show the results using 0.5° , 0.25° , and 0.125° increments of $\Delta\alpha$ and $\Delta\beta$. A Hamming window was used in all cases. We used this variety of angle increments to illustrate the effects of under-sampling.

The result in Fig. 5.48(a) is a clear indication that 1° increments are too coarse, as blurring and ringing are quite evident. The result in Fig. 5.48(b) is interesting, in the sense that it compares poorly with Fig. 5.43(b), which we generated using the same angle increment of 0.5° . In fact, as Fig. 5.48(c) shows, even with angle increments of 0.25° the reconstruction still is not as good as in Fig. 5.43(b). We have to use angle increments on the order of 0.125° before the two results become comparable, as Fig. 5.48(d) shows. This angle increment results in projections with $180 \times (1/0.125) = 1440$ samples, which is close to double the 849 rays used in the parallel projections of Example 5.17. Thus, it is not unexpected that the results are close in appearance when using $\Delta\alpha = 0.125^\circ$.

Similar results were obtained with the head phantom, except that aliasing in this case is much more visible as sinusoidal interference. We see in Fig. 5.49(c) that even with $\Delta\alpha = \Delta\beta = 0.25^\circ$ significant distortion still is present, especially in the periphery of the ellipse. As with the rectangle, using increments of 0.125° finally produced results that are comparable with the backprojected image of the head phantom

a b
c d

FIGURE 5.48
Reconstruction of the rectangle image from filtered fan backprojections.
(a) 1° increments of α and β .
(b) 0.5° increments.
(c) 0.25° increments.
(d) 0.125° increments.
Compare (d) with Fig. 5.43(b).



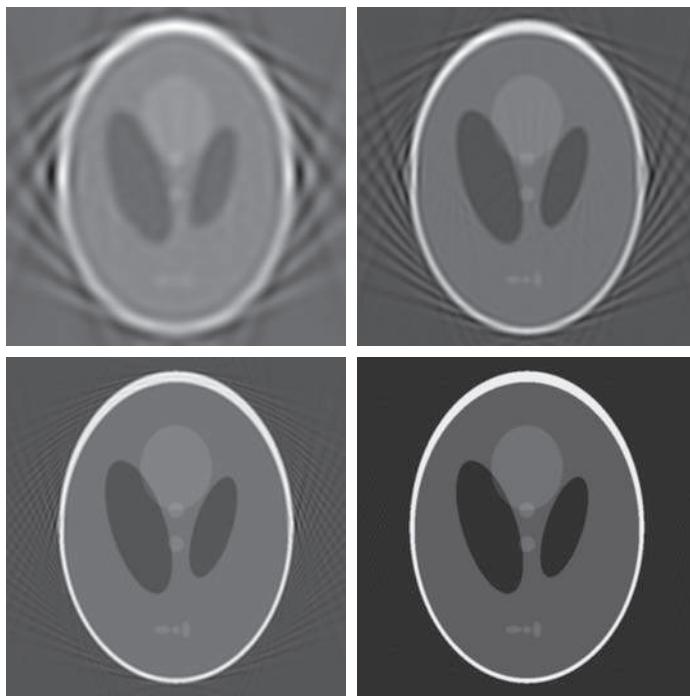
a	b
c	d

FIGURE 5.49

Reconstruction of the head phantom image from filtered fan backprojections. (a) 1° increments of α and β .

(b) 0.5° increments.
 (c) 0.25° increments.
 (d) 0.125° increments.

Compare (d) with Fig. 5.44(b).



in Fig. 5.44(b). These results illustrate one of the principal reasons why thousands of detectors have to be used in the fan-beam geometry of modern CT systems in order to reduce aliasing artifacts.

Summary, References, and Further Reading

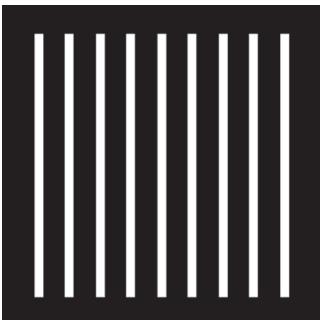
The restoration results in this chapter are based on the assumption that image degradation can be modeled as a linear, position invariant process followed by additive noise that is not correlated with image values. Even when these assumptions are not entirely valid, it is often possible to obtain useful results by using the methods developed in the preceding sections. Our treatment of image reconstruction from projections, though introductory, is the foundation for the image-processing aspects of this field. As noted in Section 5.11, computed tomography (CT) is the main application area of image reconstruction from projections. Although we focused on X-ray tomography, the principles established in Section 5.11 are applicable in other CT imaging modalities, such as SPECT (single photon emission tomography), PET (positron emission tomography), MRI (magnetic resonance imaging), and some modalities of ultrasound imaging.

For additional reading on the material in Section 5.1 see Pratt [2014]. The books by Ross [2014], and by Montgomery and Runger [2011], are good sources for a more in-depth discussion of probability density functions and their properties (Section 5.2). See Umbaugh [2010] for complementary reading on the material in Section 5.3, and Eng and Ma [2001, 2006] regarding adaptive median filtering. The filters in Section 5.4 are direct extensions of the material in Chapter 4. The material in Section 5.5 is fundamental linear system theory; for more advanced reading on this topic see Hespanha [2009]. The topic of estimating image degradation functions (Section 5.6) is fundamental in the field of image restoration. Some of the early techniques for estimating the degradation function are given in Andrews and Hunt [1977], Rosenfeld and Kak [1982]. More recent methods are discussed by Gunturk and Li [2013].

There are two major approaches to the methods developed in Sections 5.7–5.10. One is based on a general formulation using matrix theory, as introduced by Andrews and Hunt [1977] and by Gonzalez and Woods [1992]. This approach is elegant and general, but it tends to be difficult for first-time readers. Approaches based on frequency domain filtering (the approach we followed in this chapter) are easier to follow by newcomers to image restoration, but lack the unifying mathematical rigor of the matrix approach. Both approaches arrive at the same results, but our experience in teaching this material in a variety of settings indicates that students first entering this field favor the latter approach by a significant margin. Complementary readings for our coverage of these filtering concepts are Castleman [1996], Umbaugh [2010], Petrou and Petrou [2010] and Gunturk and Li [2013]. For additional reading on the material in Section 5.11 see Kak and Slaney [2001], Prince and Links [2006], and Buzug [2008]. For details on the software aspects of many of the examples in this chapter, see Gonzalez, Woods, and Eddins [2009].

Problems

Solutions to the problems marked with an asterisk () are in the DIP4E Student Support Package (consult the book website: www.ImageProcessingPlace.com).*

- 5.1*** The white bars in the test pattern shown are 7 pixels wide and 210 pixels high. The separation between bars is 17 pixels. What would this image look like after application of
- A 3×3 arithmetic mean filter?
 - A 7×7 arithmetic mean filter?
 - A 9×9 arithmetic mean filter?
- Note: This problem and the ones that follow it, related to filtering this image, may seem a bit tedious. However, they are worth the effort, as they help develop a real understanding of how these filters work. After you understand how a particular filter affects the image, your answer can be a brief verbal description of the result. For example, “the resulting image will consist of vertical bars 3 pixels wide and 206 pixels high.” Be sure to describe any deformation of the bars, such as rounded corners. You may ignore image border effects, in which the filter neighborhoods only partially contain image pixels.
- 
- 5.2** Repeat Problem 5.1 using a geometric mean filter.
- 5.3*** Repeat Problem 5.1 using a harmonic mean filter.
- 5.4** Repeat Problem 5.1 using a contraharmonic mean filter with $Q = 1$.
- 5.5*** Repeat Problem 5.1 using a contraharmonic mean filter with $Q = -1$.
- 5.6** Repeat Problem 5.1 using a median filter.
- 5.7*** Repeat Problem 5.1 using a max filter.
- 5.8** Repeat Problem 5.1 using a min filter.
- 5.9*** Repeat Problem 5.1 using a midpoint filter.
- 5.10** In answering the following, refer to the contraharmonic filter in Eq. (5-26) :
- Explain why the filter is effective in eliminating pepper noise when Q is positive.
 - Explain why the filter is effective in eliminating salt noise when Q is negative.
 - Explain why the filter gives poor results (such as the results in Fig. 5.9) when the wrong polarity is chosen for Q .
 - Discuss the expected behavior of the filter when $Q = -1$.
- 5.11** We mentioned when discussing Eq. (5-27)] that using median filters generally results in less blurring than using linear smoothing filters (e.g., box lowpass filters) of the same size. Explain why this is so. (*Hint:* In order to focus on the key difference between the filters, assume that noise is negligible, and consider the behavior of these filters in the neighborhood of a binary edge.)

- 5.12** With reference to the alpha-trimmed filter defined in Eq. (5-31)]:
- * Explain why setting $d = 0$ in the filter reduces it to an arithmetic mean filter.
 - Explain why setting $d = mn - 1$ turns the filter into a median filter.
- 5.13** With reference to the bandreject filter transfer functions in Table 4.7, obtain equations for the transfer functions of:
- An ideal bandpass filter.
 - * A Gaussian bandpass filter.
 - A Butterworth bandpass filter.
- 5.14** With reference to Eq. (5-33), obtain equations for:
- * An ideal notch filter transfer function.
 - A Gaussian notch filter transfer function.
 - A Butterworth notch filter transfer function.
- 5.15** Show that the Fourier transform of the 2-D discrete sine function
- $$f(x, y) = \sin(2\pi\mu_0 x/M + 2\pi\nu_0 y/N)$$
- for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$ is the pair of conjugate impulses
- $$F(u, v) = \frac{jMN}{2} [\delta(u + u_0, v + v_0) - \delta(u - u_0, v - v_0)]$$
- 5.16** With reference to $f(x, y)$ in Problem 5.15, answer the following:
- * If $v_0 = 0$, and u_0 and M are integers ($u_0 < M$), what would a plot of $f(x, y)$ look like along the x -axis for $x = 0, 1, 2, \dots, M - 1$?
 - * What would a plot of $F(u, v)$ look like for $u = 0, 1, 2, \dots, M - 1$?
 - If $v_0 = 0$, M is the same integer as before, but u_0 is no longer an integer ($u_0 < M$), how would a plot of $f(x, y)$ along the x -axis for $x = 0, 1, 2, \dots, M - 1$ be different from (a)?
- 5.17*** Start with Eq. (5-46) and derive Eq. (5-48).
- 5.18** An industrial plant manager has been promoted to a new position. His first responsibility is to characterize an image filtering system left by his predecessor. In reading the documentation, the manager discovers that his predecessor established that the system is linear and position invari-

ant. Furthermore, he learns that experiments conducted under negligible-noise conditions resulted in an impulse response that could be expressed analytically in the frequency domain as

$$H(u, v) = e^{-[u^2/150 + v^2/150]} + 1 - e^{-[(u - 50)^2/150 + (v - 50)^2/150]}$$

The manager is not a technical person, so he employs you as a consultant to determine what, if anything, he needs to do to complete the characterization of the system. He also wants to know the function that the system performs. What (if anything) does the manager need to do to complete the characterization of his system? What filtering function does the system perform?

- 5.19** A linear, space invariant system has the impulse response

$$h(x, y) = \delta(x - a, y - b)$$

where a and b are constants, and x and y are discrete quantities. Answer the following, assuming negligible noise in each case.

- * What is the system transfer function in the frequency domain?
- * What would the spatial domain system response be to a constant input, $f(x, y) = K$?
- What would the spatial domain system response be to an impulse input, $f(x, y) = \delta(x, y)$?

- 5.20*** Assuming now that x and y are continuous quantities, show how you would solve Problems 5.19(b) and (c) using Eq. (5-61) directly. [Hint: Take a look at the solution to Problem 4.1(c).]

- 5.21*** Consider a linear, position invariant image degradation system with impulse response

$$h(x, y) = e^{-[(x-\alpha)^2 + (y-\beta)^2]}$$

where x and y are continuous variables. Suppose that the input to the system is a binary image consisting of a white vertical line of infinitesimal width located at $x = a$, on a black background. Such an image can be modeled as $f(x, y) = \delta(x - a)$. Assume negligible noise and use Eq. (5-61) to find the output image, $g(x, y)$.

- 5.22** How would you solve Problem 5.21 if x and y were discrete quantities? You do not need to solve the problem. All you have to do is list the

steps you would take to solve it. (*Hint:* Refer to entry 13 in Table 4.4.)

- 5.23** The image shown consists of two infinitesimally thin white lines on a black background, intersecting at some point in the image. The image is input into a linear, position invariant system with the impulse response given in Problem 5.21. Assuming continuous variables and negligible noise, find an expression for the output image, $g(x,y)$. (*Hint:* Review linear operations in Section 2.6.)



- 5.24** Sketch (with arrow lines showing the direction of blur) what the image in Fig. 5.26(a) would look like if it were blurred using the transfer function in Eq. (5-77)

- (a)* With $a = -0.1$ and $b = 0.1$.
 (b) With $a = 0$ and $b = -0.1$.

- 5.25*** During acquisition, an image undergoes uniform linear motion in the vertical direction for a time T_1 . The direction of motion then switches to the horizontal direction for a time interval T_2 . Assuming that the time it takes the image to change directions is negligible, and that shutter opening and closing times are negligible also, give an expression for the blurring function, $H(u,v)$.

- 5.26** During acquisition, an image undergoes uniform linear motion in the vertical direction for a time T . The direction of motion then switches 180° in the opposite direction for a time T . Assume that the time it takes the image to change directions is negligible, and that shutter opening and closing times are negligible also. Is the final image blurred, or did the reversal in direction “undo” the first blur? Obtain the overall blurring function $H(u,v)$ first, and then use it as the basis for your answer.

- 5.27*** Consider image blurring caused by uniform acceleration in the x -direction. If the image is at rest at time $t = 0$ and accelerates with a uniform acceler-

ation $x_0(t) = at^2/2$ for a time T , find the blurring function $H(u,v)$. You may assume that shutter opening and closing times are negligible.

- 5.28** A space probe is designed to transmit images of a planet as it approaches it for landing. During the last stages of landing, one of the control thrusters fails, resulting in rotation of the craft about its vertical axis. The images sent during the last two seconds prior to landing are blurred as a consequence of this circular motion. The camera is located in the bottom of the probe, along its vertical axis, and pointing down. Fortunately, the rotation of the craft is also about its vertical axis, so the images are blurred by uniform rotational motion. During the acquisition time of each image, the craft rotation was $\pi/8$ radians. The image acquisition process can be modeled as an ideal shutter that is open only during the time the craft rotated $\pi/8$ radians. You may assume that the vertical motion was negligible during the image acquisition. Formulate a solution for restoring the images. You do not have to solve the problem, just give an outline of how you would solve it using the methods discussed in Section 5.6 through 5.9. (*Hint:* Consider using polar coordinates. The blur will then appear as one-dimensional, uniform motion blur along the θ -axis.)

- 5.29*** The image that follows is a blurred, 2-D projection of a volumetric rendition of a heart. It is known that each of the cross hairs on the right bottom part of the image was (before blurring) 3 pixels wide, 30 pixels long, and had an intensity value of 255. Provide a step-by-step procedure indicating how you would use the information just given to obtain the blurring function $H(u,v)$.



(Original image courtesy of GE Medical Systems.)

- 5.30** The image in Fig. 5.29(h) was obtained by inverse-filtering the image in Fig. 5.29(g), which is a blurred image that, in addition, is corrupted by additive Gaussian noise. The blurring itself is corrected by the inverse filter, as is evident in Fig. 5.29(h). However, the restored image has a strong streak pattern that is not apparent in Fig. 5.29(g) [for example, compare the area of constant white in the top right of Fig. 5.29(g) with the corresponding area in Fig. 5.29(h)]. Explain how this pattern originated.
- 5.31** A certain X-ray imaging geometry produces a blurring degradation that can be modeled as the convolution of the sensed image with the spatial, circularly symmetric function

$$h(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{-(x^2 + y^2)/2\sigma^2}$$

Assuming continuous variables, show that the degradation in the frequency domain is given by the expression

$$H(u, v) = -8\pi^4\sigma^2(u^2 + v^2)e^{-2\pi^2\sigma^2(u^2 + v^2)}$$

(Hint: Refer to the discussion of the Laplacian in Section 4.9, entry 13 in Table 4.4, and review Problem 4.52.)

- 5.32*** Using the transfer function in Problem 5.31, give the expression for a Wiener filter transfer function, assuming that the ratio of power spectra of the noise and undegraded images is a constant.

- 5.33** Given $p(x, y)$ in Eq. (5-90), show that

$$P(u, v) = 4 - 2\cos(2\pi u/M) - 2\cos(2\pi v/N)$$

(Hint: Study the solution to Problem 4.47.)

- 5.34** Show how Eq. (5-98) follows from Eqs. (5-96) and (5-97).

- 5.35** Using the transfer function in Problem 5.31, give the resulting expression for the constrained least squares filter transfer function.

- 5.36*** Assume that the model in Fig. 5.1 is linear and position invariant, and that the noise and image are uncorrelated. Show that the power spectrum of the output is

$$|G(u, v)|^2 = |H(u, v)|^2 |F(u, v)|^2 + |N(u, v)|^2$$

(Hint: Refer to Eqs. (5-65) and (4-89).)

- 5.37** Cannon [1974] suggested a restoration filter $R(u, v)$ satisfying the condition

$$|\hat{F}(u, v)|^2 = |R(u, v)|^2 |G(u, v)|^2$$

The restoration filter is based on the premise of forcing the power spectrum of the restored image, $|\hat{F}(u, v)|^2$, to equal the spectrum of the original image, $|F(u, v)|^2$. Assume that the image and noise are uncorrelated,

- (a)*** Find $R(u, v)$ in terms of $|F(u, v)|^2$, $|H(u, v)|^2$, and $|N(u, v)|^2$. (Hint: Take a look at Fig. 5.1, Eq. (5-65), and Problem 5.36.)

- (b)** Use your result from (a) to state a result in a form similar to the last line of Eq. (5-81), and using the same terms.

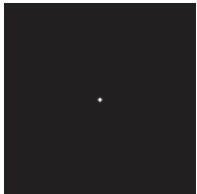
- 5.38** Show that, when $\alpha = 1$ in Eq. (5-99), the geometric mean filter reduces to the inverse filter.

- 5.39*** A professor of archeology doing research on currency exchange practices during the Roman Empire recently became aware that four Roman coins crucial to his research are listed in the holdings of the British Museum in London. Unfortunately, he was told after arriving there that the coins had been recently stolen. Further research on his part revealed that the museum keeps photographs of every item for which it is responsible. Unfortunately, the photos of the coins in question are blurred to the point where the date and other small markings are not readable. The cause of the blurring was the camera being out of focus when the pictures were taken. As an image processing expert and friend of the professor, you are asked as a favor to determine whether computer processing can be utilized to restore the images to the point where the professor can read the markings. You are told that the original camera used to take the photos is still available, as are other representative coins of the same era. Propose a step-by-step solution to this problem.

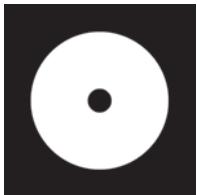
- 5.40** An astronomer is working with an optical telescope. The telescope lenses focus images onto a high-resolution, CCD imaging array, and the images are then converted by the telescope electronics into digital images. Working late one evening, the astronomer notices that her new images are noisy and blurry. The manufacturer tells the astronomer that the unit is operating within specifications. Trying to improve the situation by con-

ducting controlled lab experiments with the lenses and imaging sensors is not possible because of the size and weight of the telescope components. Having heard about your success in restoring the Roman coins, the astronomer calls you to help her formulate a digital image processing solution for sharpening her images. How would you go about solving this problem, given that the only images you can obtain are images of stellar bodies? (*Hint:* A single, bright star that appears as a point of light in the field of view can be used to approximate an impulse.)

- 5.41*** Sketch the Radon transform of the $M \times M$ binary image shown below, which consists of a single white pixel in the center of the image. Assume a parallel-beam geometry, and label quantitatively all the important elements of your sketch .



- 5.42*** Sketch a cross section of the Radon transform of the following white disk image containing a smaller black disk in its center. (*Hint:* Take a look at Fig. 5.38.)



- 5.43** Show that the Radon transform [Eq. (5-102)] of the Gaussian shape $f(x, y) = A \exp(-x^2 - y^2)$ is given by $g(\rho, \theta) = A\sqrt{\pi} \exp(-\rho^2)$. (*Hint:* Refer to Example 5.15, where we used symmetry to simplify integration.)

- 5.44** Do the following:

(a)* Show that the Radon transform [Eq. (5-102)] of the unit impulse $\delta(x, y)$ is a straight vertical line passing through the origin of the $\rho\theta$ -plane .

(b) Show that the radon transform of the impulse $\delta(x - x_0, y - y_0)$ is a sinusoidal curve in the $\rho\theta$ -plane.

- 5.45** Prove the validity of the following properties of the Radon transform [Eq. (5-102)]:

(a)* *Linearity:* The Radon transform is a linear operator. (See Section 2.6 regarding linearity.)

(b) *Translation property:* The radon transform of $f(x - x_0, y - y_0)$ is $g(\rho - x_0 \cos \theta - y_0 \sin \theta, \theta)$.

(c)* *Convolution property:* The Radon transform of the convolution of two functions is equal to the convolution of the Radon transforms of the two functions.

- 5.46** Provide the steps that lead from Eq. (5-113) to Eq. (5-114). [*Hint:* $G(\omega, \theta + 180^\circ) = G(-\omega, \theta)$.]

- 5.47*** Prove the validity of Eq. (5-125).

- 5.48** Prove the validity of Eq. (5-127).

This page intentionally left blank

6



It is only after years of preparation that the young artist should touch color—not color used descriptively, that is, but as a means of personal expression.

Henri Matisse

For a long time I limited myself to one color—as a form of discipline.

Pablo Picasso

Preview

Using color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades, compared to only about two dozen shades of gray. The latter factor is particularly important in manual image analysis. Color image processing is divided into two major areas: *pseudo-* and *full-color* processing. In the first category, the issue is one of assigning color(s) to a particular grayscale intensity or range of intensities. In the second, images typically are acquired using a full-color sensor, such as a digital camera, or color scanner. Until just a few years ago, most digital color image processing was done at the *pseudo-* or reduced-color level. However, because color sensors and processing hardware have become available at reasonable prices, full-color image processing techniques are now used in a broad range of applications. In the discussions that follow, it will become evident that some of the grayscale methods covered in previous chapters are applicable also to color images.

Upon completion of this chapter, readers should:

- Understand the fundamentals of color and the color spectrum.
- Be familiar with several of the color models used in digital image processing.
- Know how to apply basic techniques in pseudo-color image processing, including intensity slicing and intensity-to-color transformations.
- Be familiar with how to determine if a grayscale method is extendible to color images.
- Understand the basics of working with full-color images, including color transformations, color complements, and tone/color corrections.
- Be familiar with the role of noise in color image processing.
- Know how to perform spatial filtering on color images.
- Understand the advantages of using color in image segmentation.

6.1 COLOR FUNDAMENTALS

Although the process employed by the human brain in perceiving and interpreting color is a physiopsychological phenomenon that is not fully understood, the physical nature of color can be expressed on a formal basis supported by experimental and theoretical results.

In 1666, Sir Isaac Newton discovered that when a beam of sunlight passes through a glass prism, the emerging light is not white, but consists instead of a continuous spectrum of colors ranging from violet at one end to red at the other. As Fig. 6.1 shows, the color spectrum may be divided into six broad regions: violet, blue, green, yellow, orange, and red. When viewed in full color (see Fig. 6.2), no color in the spectrum ends abruptly; rather, each color blends smoothly into the next.

Basically, the colors that humans and some other animals perceive in an object are determined by the nature of the light reflected from the object. As illustrated in Fig. 6.2, visible light is composed of a relatively narrow band of frequencies in the electromagnetic spectrum. A body that reflects light that is balanced in all visible wavelengths appears white to the observer. However, a body that favors reflectance in a limited range of the visible spectrum exhibits some shades of color. For example, green objects reflect light with wavelengths primarily in the 500 to 570 nm range, while absorbing most of the energy at other wavelengths.

Characterization of light is central to the science of color. If the light is *achromatic* (void of color), its only attribute is its *intensity*, or amount. Achromatic light is what you see on movie films made before the 1930s. As defined in Chapter 2, and used numerous times since, the term *gray* (or *intensity*) *level* refers to a scalar measure of intensity that ranges from black, to grays, and finally to white.

Chromatic light spans the electromagnetic spectrum from approximately 400 to 700 nm. Three basic quantities used to describe the quality of a chromatic light source are: radiance, luminance, and brightness. *Radiance* is the total amount of energy that flows from the light source, and it is usually measured in watts (W). *Luminance*, measured in lumens (lm), is a measure of the amount of energy that an observer *perceives* from a light source. For example, light emitted from a source operating in the far infrared region of the spectrum could have significant energy (radiance), but an observer would hardly perceive it; its luminance would be almost zero. Finally, *brightness* is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of intensity, and is one of the key factors in describing color sensation.

FIGURE 6.1

Color spectrum seen by passing white light through a prism.
(Courtesy of the General Electric Co., Lighting Division.)

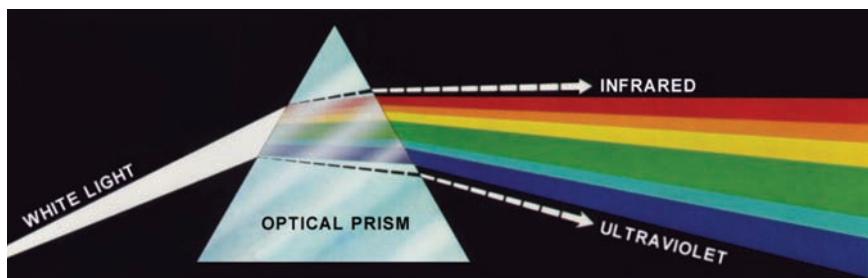


FIGURE 6.2

Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lighting Division.)

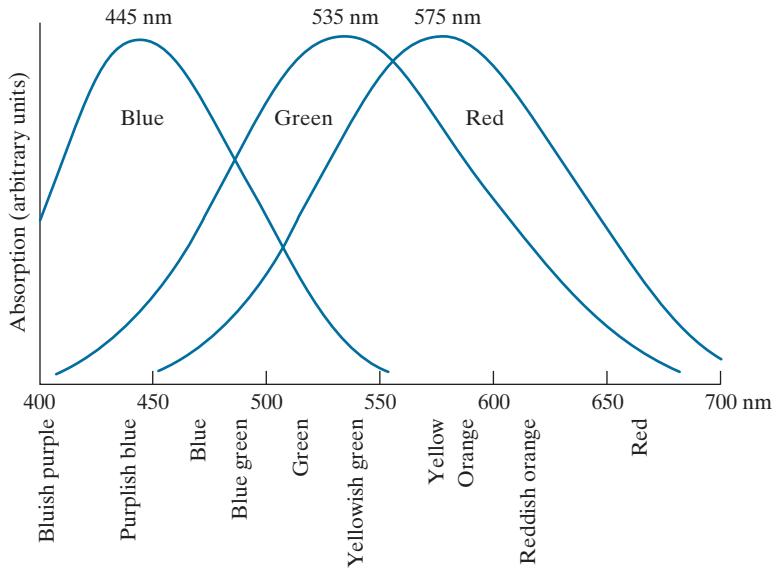


As noted in Section 2.1, cones are the sensors in the eye responsible for color vision. Detailed experimental evidence has established that the 6 to 7 million cones in the human eye can be divided into three principal sensing categories, corresponding roughly to red, green, and blue. Approximately 65% of all cones are sensitive to red light, 33% are sensitive to green light, and only about 2% are sensitive to blue. However, the blue cones are the most sensitive. Figure 6.3 shows average experimental curves detailing the absorption of light by the red, green, and blue cones in the eye. Because of these absorption characteristics, the human eye sees colors as variable combinations of the so-called *primary colors*: red (R), green (G), and blue (B).

For the purpose of standardization, the CIE (Commission Internationale de l'Eclairage—the International Commission on Illumination) designated in 1931 the following specific wavelength values to the three primary colors: blue = 435.8 nm, green = 546.1 nm, and red = 700 nm. This standard was set before results such as those in Fig. 6.3 became available in 1965. Thus, the CIE standards correspond only approximately with experimental data. It is important to keep in mind that defining three specific primary color wavelengths for the purpose of standardization does

FIGURE 6.3

Absorption of light by the red, green, and blue cones in the human eye as a function of wavelength.



not mean that these three fixed RGB components acting alone can generate all spectrum colors. Use of the word *primary* has been widely misinterpreted to mean that the three standard primaries, when mixed in various intensity proportions, can produce all visible colors. As you will see shortly, this interpretation is not correct unless the wavelength also is allowed to vary, in which case we would no longer have three fixed primary colors.

The primary colors can be added together to produce the *secondary* colors of light—*magenta* (red plus blue), *cyan* (green plus blue), and *yellow* (red plus green). Mixing the three primaries, or a secondary with its opposite primary color, in the right intensities produces white light. This result is illustrated in Fig. 6.4(a), which shows also the three primary colors and their combinations to produce the secondary colors of light.

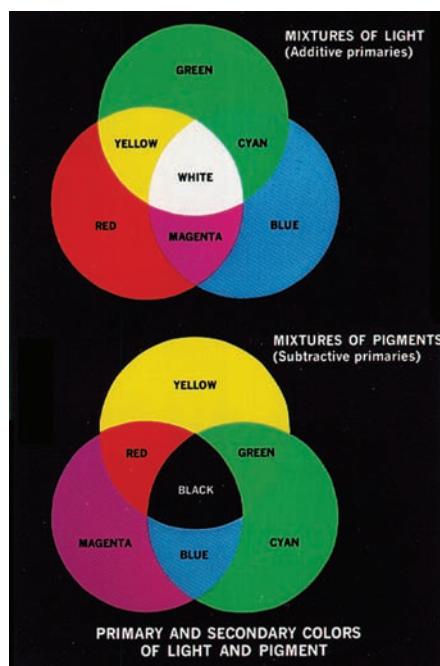
Differentiating between the primary colors of light and the primary colors of pigments or colorants is important. In the latter, a primary color is defined as one that subtracts or absorbs a primary color of light, and reflects or transmits the other two. Therefore, the primary colors of pigments are magenta, cyan, and yellow, and the secondary colors are red, green, and blue. These colors are shown in Fig. 6.4(b). A proper combination of the three pigment primaries, or a secondary with its opposite primary, produces black.

Color television reception is an example of the additive nature of light colors. The interior of CRT (cathode ray tube) color TV screens used well into the 1990s is composed of a large array of triangular dot patterns of electron-sensitive phosphor. When excited, each dot in a triad produces light in one of the primary colors. The

In practice, pigments seldom are pure. This results in a muddy brown instead of black when primaries, or primaries and secondaries, are combined. We will discuss this issue in Section 6.2

a
b

FIGURE 6.4
Primary and secondary colors of light and pigments.
(Courtesy of the General Electric Co., Lighting Division.)



intensity of the red-emitting phosphor dots is modulated by an electron gun inside the tube, which generates pulses corresponding to the “red energy” seen by the TV camera. The green and blue phosphor dots in each triad are modulated in the same manner. The effect, viewed on the television receiver, is that the three primary colors from each phosphor triad are received and “added” together by the color-sensitive cones in the eye and perceived as a full-color image. Thirty successive image changes per second in all three colors complete the illusion of a continuous image display on the screen.

CRT displays started being replaced in the late 1990s by flat-panel digital technologies, such as liquid crystal displays (LCDs) and plasma devices. Although they are fundamentally different from CRTs, these and similar technologies use the same principle in the sense that they all require three subpixels (red, green, and blue) to generate a single color pixel. LCDs use properties of polarized light to block or pass light through the LCD screen and, in the case of active matrix display technologies, thin film transistors (TFTs) are used to provide the proper signals to address each pixel on the screen. Light filters are used to produce the three primary colors of light at each pixel triad location. In plasma units, pixels are tiny gas cells coated with phosphor to produce one of the three primary colors. The individual cells are addressed in a manner analogous to LCDs. This individual pixel triad coordinate addressing capability is the foundation of digital displays.

The characteristics generally used to distinguish one color from another are brightness, hue, and saturation. As indicated earlier in this section, brightness embodies the achromatic notion of intensity. *Hue* is an attribute associated with the dominant wavelength in a mixture of light waves. Hue represents dominant color as perceived by an observer. Thus, when we call an object red, orange, or yellow, we are referring to its hue. *Saturation* refers to the relative purity or the amount of white light mixed with a hue. The pure spectrum colors are fully saturated. Colors such as pink (red and white) and lavender (violet and white) are less saturated, with the degree of saturation being inversely proportional to the amount of white light added.

Hue and saturation taken together are called *chromaticity* and, therefore, a color may be characterized by its brightness and chromaticity. The amounts of red, green, and blue needed to form any particular color are called the *tristimulus* values, and are denoted, X , Y , and Z , respectively. A color is then specified by its *trichromatic coefficients*, defined as

$$x = \frac{X}{X + Y + Z} \quad (6-1)$$

$$y = \frac{Y}{X + Y + Z} \quad (6-2)$$

and

$$z = \frac{Z}{X + Y + Z} \quad (6-3)$$

Our use of x , y , and z in this context follows convention. These should not be confused with our use of (x, y) throughout the book to denote spatial coordinates.

We see from these equations that

$$x + y + z = 1 \quad (6-4)$$

For any wavelength of light in the visible spectrum, the tristimulus values needed to produce the color corresponding to that wavelength can be obtained directly from curves or tables that have been compiled from extensive experimental results (Poynton [1996, 2012]).

Another approach for specifying colors is to use the CIE *chromaticity diagram* (see Fig. 6.5), which shows color composition as a function of x (red) and y (green). For any value of x and y , the corresponding value of z (blue) is obtained from Eq. (6-4) by noting that $z = 1 - (x + y)$. The point marked green in Fig. 6.5, for example, has approximately 62% green and 25% red content. It follows from Eq. (6-4) that the composition of blue is approximately 13%.

The positions of the various spectrum colors—from violet at 380 nm to red at 780 nm—are indicated around the boundary of the tongue-shaped chromaticity diagram. These are the pure colors shown in the spectrum of Fig. 6.2. Any point not actually on the boundary, but within the diagram, represents some mixture of the pure spectrum colors. The *point of equal energy* shown in Fig. 6.5 corresponds to equal fractions of the three primary colors; it represents the CIE standard for white light. Any point located on the boundary of the chromaticity chart is fully saturated. As a point leaves the boundary and approaches the point of equal energy, more white light is added to the color, and it becomes less saturated. The saturation at the point of equal energy is zero.

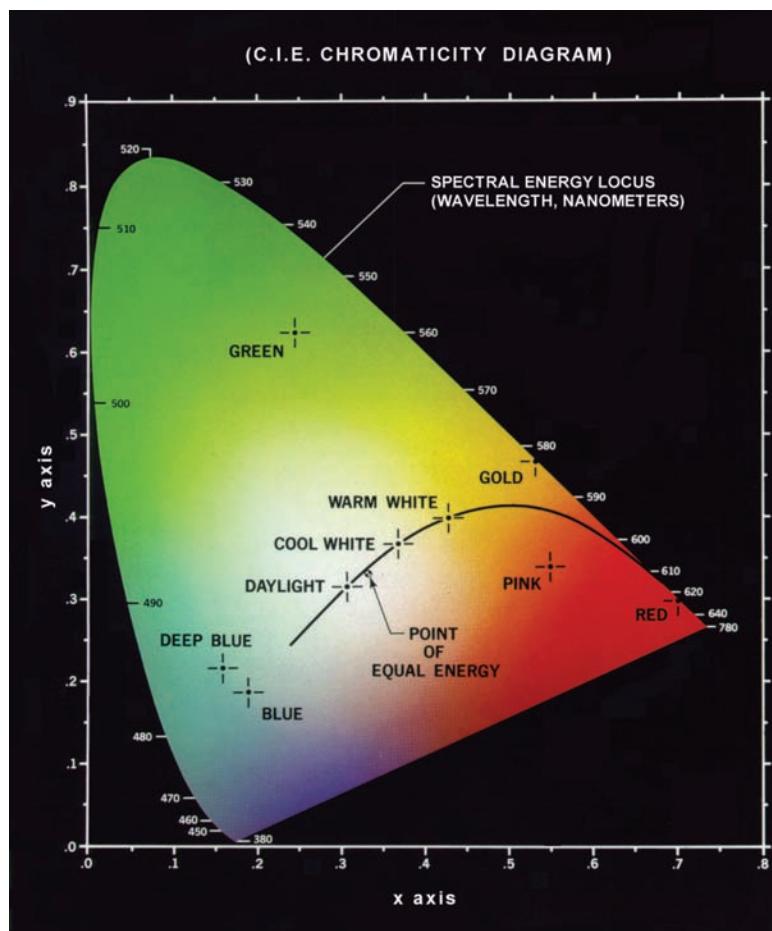
The chromaticity diagram is useful for color mixing because a straight-line segment joining any two points in the diagram defines all the different color variations that can be obtained by combining these two colors additively. Consider, for example, a straight line drawn from the red to the green points shown in Fig. 6.5. If there is more red than green light, the exact point representing the new color will be on the line segment, but it will be closer to the red point than to the green point. Similarly, a line drawn from the point of equal energy to any point on the boundary of the chart will define all the shades of that particular spectrum color.

Extending this procedure to three colors is straightforward. To determine the range of colors that can be obtained from any three given colors in the chromaticity diagram, we simply draw connecting lines to each of the three color points. The result is a triangle, and any color inside the triangle, or on its boundary, can be produced by various combinations of the three vertex colors. A triangle with vertices at any three fixed colors cannot enclose the entire color region in Fig. 6.5. This observation supports graphically the remark made earlier that not all colors can be obtained with three single, *fixed* primaries, because three colors form a triangle.

The triangle in Fig. 6.6 shows a representative range of colors (called the *color gamut*) produced by RGB monitors. The shaded region inside the triangle illustrates the color gamut of today's high-quality color printing devices. The boundary of the color printing gamut is irregular because color printing is a combination of additive and subtractive color mixing, a process that is much more difficult to control than

FIGURE 6.5

The CIE chromaticity diagram.
 (Courtesy of the General Electric Co., Lighting Division.)



that of displaying colors on a monitor, which is based on the addition of three highly controllable light primaries.

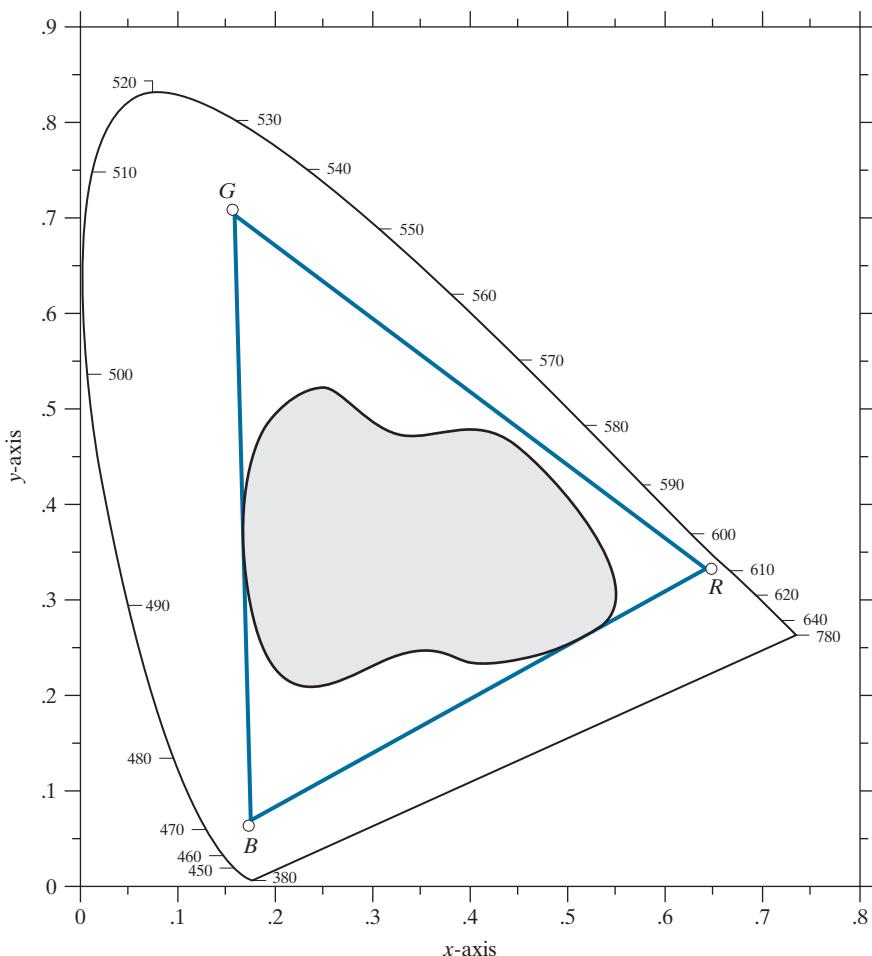
6.2 COLOR MODELS

The purpose of a *color model* (also called a *color space* or *color system*) is to facilitate the specification of colors in some standard way. In essence, a color model is a specification of (1) a coordinate system, and (2) a subspace within that system, such that each color in the model is represented by a single point contained in that subspace.

Most color models in use today are oriented either toward hardware (such as for color monitors and printers) or toward applications, where color manipulation is a goal (the creation of color graphics for animation is an example of the latter). In terms of digital image processing, the hardware-oriented models most commonly used in practice are the RGB (red, green, blue) model for color monitors and a

FIGURE 6.6

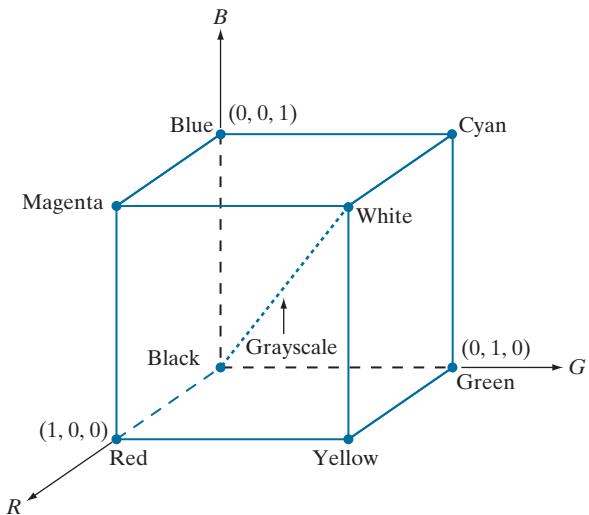
Illustrative color gamut of color monitors (triangle) and color printing devices (shaded region).



broad class of color video cameras; the CMY (cyan, magenta, yellow) and CMYK (cyan, magenta, yellow, black) models for color printing; and the HSI (hue, saturation, intensity) model, which corresponds closely with the way humans describe and interpret color. The HSI model also has the advantage that it decouples the color and gray-scale information in an image, making it suitable for many of the gray-scale techniques developed in this book. There are numerous color models in use today. This is a reflection of the fact that color science is a broad field that encompasses many areas of application. It is tempting to dwell on some of these models here, simply because they are interesting and useful. However, keeping to the task at hand, we focus attention on a few models that are representative of those used in image processing. Having mastered the material in this chapter, you will have no difficulty in understanding additional color models in use today.

FIGURE 6.7

Schematic of the RGB color cube. Points along the main diagonal have gray values, from black at the origin to white at point $(1, 1, 1)$.



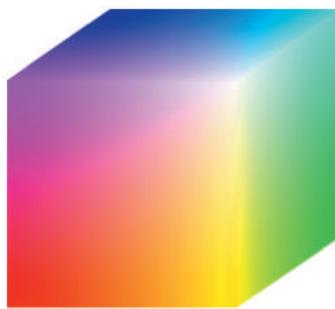
THE RGB COLOR MODEL

In the RGB model, each color appears in its primary spectral components of red, green, and blue. This model is based on a Cartesian coordinate system. The color subspace of interest is the cube shown in Fig. 6.7, in which RGB primary values are at three corners; the secondary colors cyan, magenta, and yellow are at three other corners; black is at the origin; and white is at the corner farthest from the origin. In this model, the grayscale (points of equal RGB values) extends from black to white along the line joining these two points. The different colors in this model are points on or inside the cube, and are defined by vectors extending from the origin. For convenience, the assumption is that all color values have been normalized so the cube in Fig. 6.7 is the unit cube. That is, all values of R, G, and B in this representation are assumed to be in the range $[0, 1]$. Note that the RGB primaries can be interpreted as unit vectors emanating from the origin of the cube.

Images represented in the RGB color model consist of three component images, one for each primary color. When fed into an RGB monitor, these three images combine on the screen to produce a composite color image, as explained in Section 6.1. The number of bits used to represent each pixel in RGB space is called the *pixel depth*. Consider an RGB image in which each of the red, green, and blue images is an 8-bit image. Under these conditions, each RGB *color pixel* [that is, a triplet of values (R, G, B)] has a depth of 24 bits (3 image planes times the number of bits per plane). The term *full-color* image is used often to denote a 24-bit RGB color image. The total number of possible colors in a 24-bit RGB image is $(2^8)^3 = 16,777,216$. Figure 6.8 shows the 24-bit RGB color cube corresponding to the diagram in Fig. 6.7. Note also that for digital images, the range of values in the cube are scaled to the

FIGURE 6.8

A 24-bit RGB color cube.



numbers representable by the number bits in the images. If, as above, the primary images are 8-bit images, the limits of the cube along each axis becomes [0, 255]. Then, for example, white would be at point [255, 255, 255] in the cube.

EXAMPLE 6.1: Generating a cross-section of the RGB color cube and its three hidden planes.

The cube in Fig. 6.8 is a solid, composed of the $(2^8)^3$ colors mentioned in the preceding paragraph. A useful way to view these colors is to generate color planes (faces or cross sections of the cube). This is done by fixing one of the three colors and allowing the other two to vary. For instance, a cross-sectional plane through the center of the cube and parallel to the GB-plane in Fig. 6.8 is the plane (127, G, B) for $G, B = 0, 1, 2, \dots, 255$. Figure 6.9(a) shows that an image of this cross-sectional plane is generated by feeding the three individual component images into a color monitor. In the component images, 0 represents black and 255 represents white. Observe that each component image into the monitor is a grayscale image. The monitor does the job of combining the intensities of these images to generate an RGB image. Figure 6.9(b) shows the three hidden surface planes of the cube in Fig. 6.8, generated in a similar manner.

Acquiring a color image is the process shown in Fig. 6.9(a) in reverse. A color image can be acquired by using three filters, sensitive to red, green, and blue, respectively. When we view a color scene with a monochrome camera equipped with one of these filters, the result is a monochrome image whose intensity is proportional to the response of that filter. Repeating this process with each filter produces three monochrome images that are the RGB component images of the color scene. In practice, RGB color image sensors usually integrate this process into a single device. Clearly, displaying these three RGB component images as in Fig. 6.9(a) would yield an RGB color rendition of the original color scene.

THE CMY AND CMYK COLOR MODELS

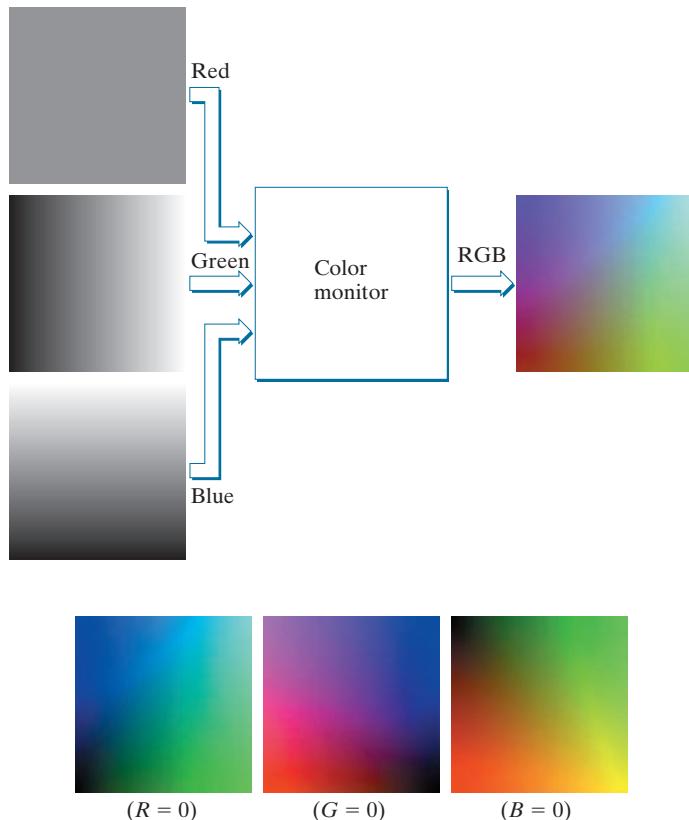
As indicated in Section 6.1, cyan, magenta, and yellow are the secondary colors of light or, alternatively, they are the primary colors of pigments. For example, when a surface coated with cyan pigment is illuminated with white light, no red light is reflected from the surface. That is, cyan subtracts red light from reflected white light, which itself is composed of equal amounts of red, green, and blue light.

Most devices that deposit colored pigments on paper, such as color printers and copiers, require CMY data input or perform an RGB to CMY conversion internally. This conversion is performed using the simple operation

a
b

FIGURE 6.9

- (a) Generating the RGB image of the cross-sectional color plane (127, G, B).
 (b) The three hidden surface planes in the color cube of Fig. 6.8.



Equation (6-5), as well as all other equations in this section, are applied on a pixel-by-pixel basis.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (6-5)$$

where the assumption is that all RGB color values have been normalized to the range [0, 1]. Equation (6-5) demonstrates that light reflected from a surface coated with pure cyan does not contain red (that is, $C = 1 - R$ in the equation). Similarly, pure magenta does not reflect green, and pure yellow does not reflect blue. Equation (6-5) also reveals that RGB values can be obtained easily from a set of CMY values by subtracting the individual CMY values from 1.

According to Fig. 6.4, equal amounts of the pigment primaries, cyan, magenta, and yellow, should produce black. In practice, because C, M, and Y inks seldom are pure colors, combining these colors for printing black produces instead a muddy-looking brown. So, in order to produce true black (which is the predominant color in printing), a fourth color, *black*, denoted by K , is added, giving rise to the CMYK color model. The black is added in just the proportions needed to produce true black. Thus,

when publishers talk about “four-color printing,” they are referring to the three CMY colors, plus a portion of black.

The conversion from CMY to CMYK begins by letting

$$K = \min(C, M, Y) \quad (6-6)$$

If $K = 1$, then we have pure black, with no color contributions, from which it follows that

$$C = 0 \quad (6-7)$$

$$M = 0 \quad (6-8)$$

$$Y = 0 \quad (6-9)$$

Otherwise,

$$C = (C - K)/(1 - K) \quad (6-10)$$

$$M = (M - K)/(1 - K) \quad (6-11)$$

$$Y = (Y - K)/(1 - K) \quad (6-12)$$

The C , M , and Y on the right side of Eqs. (6-6)-(6-12) are in the CMY color system. The C , M , and Y on the left of Eqs. (6-7)-(6-12) are in the CMYK system.

The C , M , Y , and K on the right side of Eqs. (6-13)-(6-15) are in the CMYK color system. The C , M , and Y on the left of these equations are in the CMY system.

where all values are assumed to be in the range $[0, 1]$. The conversions from CMYK back to CMY are:

$$C = C * (1 - K) + K \quad (6-13)$$

$$M = M * (1 - K) + K \quad (6-14)$$

$$Y = Y * (1 - K) + K \quad (6-15)$$

As noted at the beginning of this section, all operations in the preceding equations are performed on a pixel-by-pixel basis. Because we can use Eq. (6-5) to convert both ways between CMY and RGB, we can use that equation as a “bridge” to convert between RGB and CMYK, and vice versa.

It is important to keep in mind that all the conversions just presented to go between RGB, CMY, and CMYK are based on the preceding relationships as a group. There are many other ways to convert between these color models, so you cannot mix approaches and expect to get meaningful results. Also, colors seen on monitors generally appear much different when printed, unless these devices are calibrated (see the discussion of a device-independent color model later in this section). The same holds true in general for colors converted from one model to another. However, our interest in this chapter is not on color fidelity; rather, we are interested in using the properties of color models to facilitate image processing tasks, such as region detection.

THE HSI COLOR MODEL

As we have seen, creating colors in the RGB, CMY, and CMYK models, and changing from one model to the other, is straightforward. These color systems are ideally suited for hardware implementations. In addition, the RGB system matches nicely with the fact that the human eye is strongly perceptive to red, green, and blue primaries. Unfortunately, the RGB, CMY, and other similar color models are not well suited for describing colors in terms that are practical for human interpretation. For example, one does not refer to the color of an automobile by giving the percentage of each of the primaries composing its color. Furthermore, we do not think of color images as being composed of three primary images that combine to form a single image.

When humans view a color object, we describe it by its hue, saturation, and brightness. Recall from the discussion in Section 6.1 that hue is a color attribute that describes a pure color (pure yellow, orange, or red), whereas saturation gives a measure of the degree to which a pure color is diluted by white light. Brightness is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of *intensity* and is one of the key factors in describing color sensation. We do know that intensity (gray level) is a most useful descriptor of achromatic images. This quantity definitely is measurable and easily interpretable. The model we are about to present, called the *HSI* (hue, saturation, intensity) *color model*, decouples the intensity component from the color-carrying information (hue and saturation) in a color image. As a result, the HSI model is a useful tool for developing image processing algorithms based on color descriptions that are natural and intuitive to humans, who, after all, are the developers and users of these algorithms. We can summarize by saying that RGB is ideal for image color generation (as in image capture by a color camera or image display on a monitor screen), but its use for color description is much more limited. The material that follows provides an effective way to do this.

We know from Example 6.1 that an RGB color image is composed three grayscale intensity images (representing red, green, and blue), so it should come as no surprise that we can extract intensity from an RGB image. This becomes clear if we take the color cube from Fig. 6.7 and stand it on the black, $(0, 0, 0)$, vertex, with the white, $(1, 1, 1)$, vertex directly above it [see Fig. 6.10(a)]. As noted in our discussion of Fig. 6.7, the intensity (gray) scale is along the line joining these two vertices. In Figs. 6.10(a) and (b), the line (intensity axis) joining the black and white vertices is vertical. Thus, if we wanted to determine the intensity component of any color point in Fig. 6.10, we would simply define a plane that contains the color point and, at the same time, is perpendicular to the intensity axis. The intersection of the plane with the intensity axis would give us a point with intensity value in the range $[0, 1]$. A little thought would reveal that the saturation (purity) of a color increases as a function of distance from the intensity axis. In fact, the saturation of points on the intensity axis is zero, as evidenced by the fact that all points along this axis are gray.

Hue can be determined from an RGB value also. To see how, consider Fig. 6.10(b), which shows a plane defined by three points (black, white, and cyan). The fact that

a b

FIGURE 6.10

Conceptual relationships between the RGB and HSI color models.



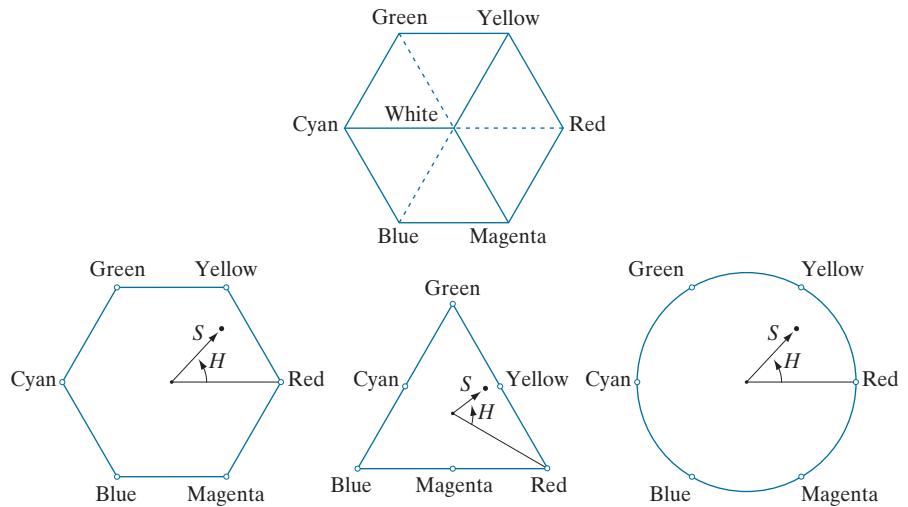
the black and white points are contained in the plane tells us that the intensity axis also is contained in the plane. Furthermore, we see that *all* points contained in the plane segment defined by the intensity axis and the boundaries of the cube have the *same* hue (cyan in this case). We could arrive at the same conclusion by recalling from Section 6.1 that all colors generated by three colors lie in the triangle defined by those colors. If two of those points are black and white, and the third is a color point, all points on the triangle would have the same hue, because the black and white components cannot change the hue (of course, the intensity and saturation of points in this triangle would be different). By rotating the shaded plane about the vertical intensity axis, we would obtain different hues. From these concepts, we arrive at the conclusion that the hue, saturation, and intensity values required to form the HSI space can be obtained from the RGB color cube. That is, we can convert any RGB point to a corresponding point in the HSI color space by working out the formulas that describe the reasoning outlined in the preceding discussion.

The key point regarding the cube arrangement in Fig. 6.10, and its corresponding HSI color space, is that the HSI space is represented by a vertical intensity axis, and the locus of color points that lie on planes perpendicular to that axis. As the planes move up and down the intensity axis, the boundaries defined by the intersection of each plane with the faces of the cube have either a triangular or a hexagonal shape. This can be visualized much more readily by looking at the cube straight down its grayscale axis, as shown in Fig. 6.11(a). We see that the primary colors are separated by 120° . The secondary colors are 60° from the primaries, which means that the angle between secondaries is 120° also. Figure 6.11(b) shows the same hexagonal shape and an arbitrary color point (shown as a dot). The hue of the point is determined by an angle from some reference point. Usually (but not always) an angle of 0° from the red axis designates 0 hue, and the hue increases counterclockwise from there. The saturation (distance from the vertical axis) is the length of the vector from the origin to the point. Note that the origin is defined by the intersection of the color plane with the vertical intensity axis. The important components of the HSI color space are the vertical intensity axis, the length of the vector to a color point, and the

a
b c d

FIGURE 6.11

Hue and saturation in the HSI color model. The dot is any color point. The angle from the red axis gives the hue. The length of the vector is the saturation. The intensity of all colors in any of these planes is given by the position of the plane on the vertical intensity axis.



angle this vector makes with the red axis. Therefore, it is not unusual to see the HSI planes defined in terms of the hexagon just discussed, a triangle, or even a circle, as Figs. 6.11(c) and (d) show. The shape chosen does not matter because any one of these shapes can be warped into one of the other two by a geometric transformation. Figure 6.12 shows the HSI model based on color triangles, and on circles.

Converting Colors from RGB to HSI

Given an image in RGB color format, the H component of each RGB pixel is obtained using the equation

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases} \quad (6-16)$$

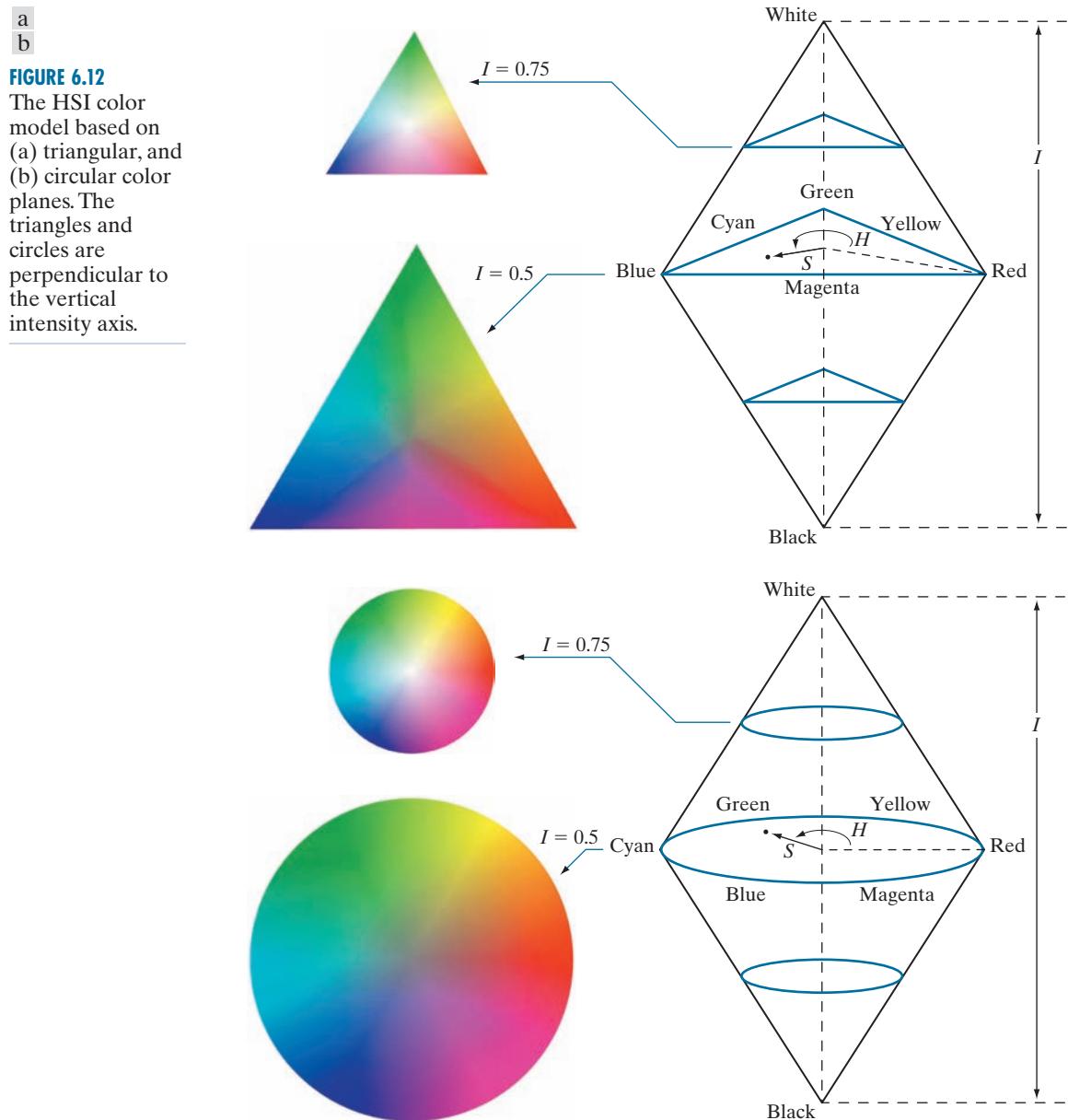
with[†]

$$\theta = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{[(R-G)^2 + (R-B)(G-B)]^{1/2}}} \right\} \quad (6-17)$$

The saturation component is given by

$$S = 1 - \frac{3}{(R+G+B)} [\min(R, G, B)] \quad (6-18)$$

[†] It is good practice to add a small number in the denominator of this expression to avoid dividing by 0 when $R = G = B$, in which case θ will be 90° . Note that when all RGB components are equal, Eq. (6-18) gives $S = 0$. In addition, the conversion from HSI back to RGB in Eqs. (6-20) through (6-30) will give $R = G = B = I$, as expected, because, when $R = G = B$, we are dealing with a grayscale image.



Finally, the intensity component is obtained from the equation

$$I = \frac{1}{3}(R + G + B) \quad (6-19)$$

These equations assume that the RGB values have been normalized to the range $[0, 1]$, and that angle θ is measured with respect to the red axis of the HSI space, as in Fig. 6.11. Hue can be normalized to the range $[0, 1]$ by dividing by 360° all values resulting from Eq. (6-16). The other two HSI components already are in this range if the given RGB values are in the interval $[0, 1]$.

The results in Eqs. (6-16) through (6-19) can be derived from the geometry in Figs. 6.10 and 6.11. The derivation is tedious and would not add significantly to the present discussion. You can find the proof for these equations (and for the equations that follow for HSI to RGB conversion) in the *Tutorials* section of the book website.

Converting Colors from HSI to RGB

Given values of HSI in the interval $[0, 1]$, we now want to find the corresponding RGB values in the same range. The applicable equations depend on the values of H . There are three sectors of interest, corresponding to the 120° intervals in the separation of primaries (see Fig. 6.11). We begin by multiplying H by 360° , which returns the hue to its original range of $[0^\circ, 360^\circ]$.

RG sector ($0^\circ \leq H < 120^\circ$): When H is in this sector, the RGB components are given by the equations

$$B = I(1 - S) \quad (6-20)$$

$$R = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right] \quad (6-21)$$

and

$$G = 3I - (R + B) \quad (6-22)$$

GB sector ($120^\circ \leq H < 240^\circ$): If the given value of H is in this sector, we first subtract 120° from it:

$$H = H - 120^\circ \quad (6-23)$$

Then, the RGB components are

$$R = I(1 - S) \quad (6-24)$$

$$G = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right] \quad (6-25)$$

and

$$B = 3I - (R + G) \quad (6-26)$$

BR sector ($240^\circ \leq H \leq 360^\circ$): Finally, if H is in this range, we subtract 240° from it:

$$H = H - 240^\circ \quad (6-27)$$

Then, the RGB components are

$$G = I(1 - S) \quad (6-28)$$

$$B = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right] \quad (6-29)$$

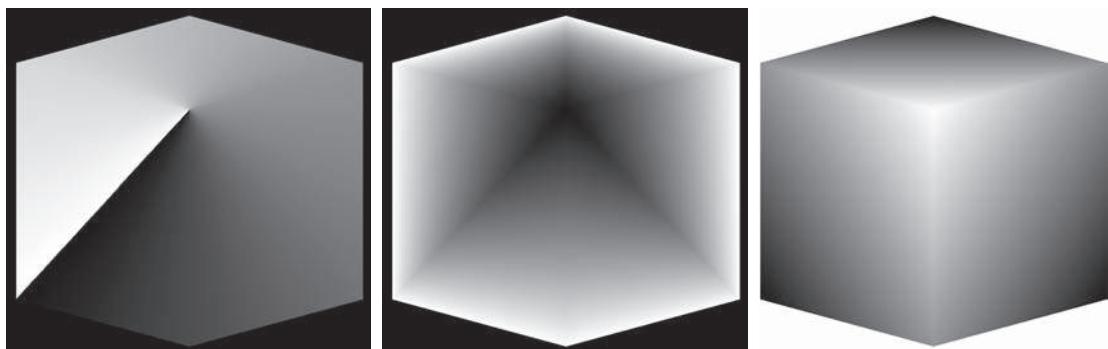
and

$$R = 3I - (G + B) \quad (6-30)$$

We discuss several uses of these equations in the following sections.

EXAMPLE 6.2: The HSI values corresponding to the image of the RGB color cube.

Figure 6.13 shows the hue, saturation, and intensity images for the RGB values in Fig. 6.8. Figure 6.13(a) is the hue image. Its most distinguishing feature is the discontinuity in value along a 45° line in the front (red) plane of the cube. To understand the reason for this discontinuity, refer to Fig. 6.8, draw a line from the red to the white vertices of the cube, and select a point in the middle of this line. Starting at that point, draw a path to the right, following the cube around until you return to the starting point. The major colors encountered in this path are yellow, green, cyan, blue, magenta, and back to red. According to Fig. 6.11, the values of hue along this path should increase from 0° to 360° (i.e., from the lowest to highest



a b c

FIGURE 6.13 HSI components of the image in Fig. 6.8: (a) hue, (b) saturation, and (c) intensity images.

possible values of hue). This is precisely what Fig. 6.13(a) shows, because the lowest value is represented as black and the highest value as white in the grayscale. In fact, the hue image was originally normalized to the range $[0, 1]$ and then scaled to 8 bits; that is, we converted it to the range $[0, 255]$, for display.

The saturation image in Fig. 6.13(b) shows progressively darker values toward the white vertex of the RGB cube, indicating that colors become less and less saturated as they approach white. Finally, every pixel in the intensity image shown in Fig. 6.13(c) is the average of the RGB values at the corresponding pixel in Fig. 6.8.

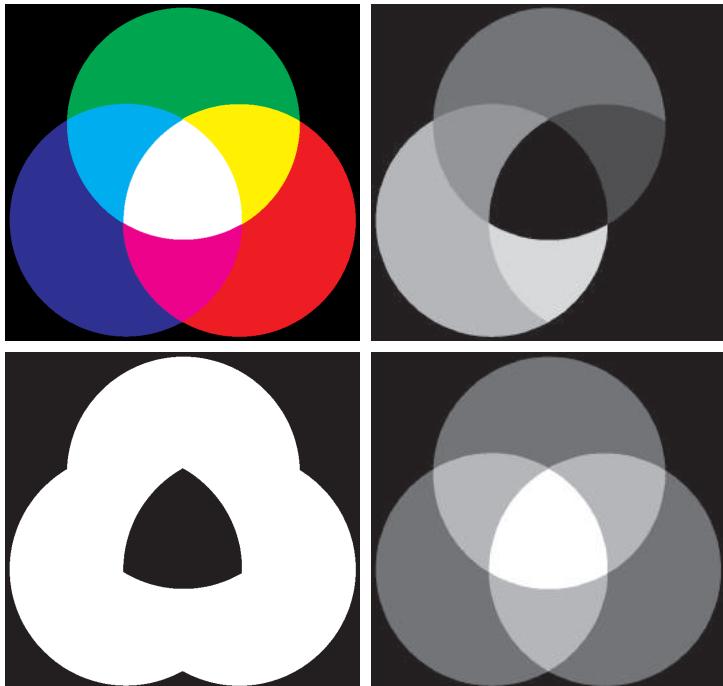
Manipulating HSI Component Images

In the following discussion, we take a look at some simple techniques for manipulating HSI component images. This will help you develop familiarity with these components, and deepen your understanding of the HSI color model. Figure 6.14(a) shows an image composed of the primary and secondary RGB colors. Figures 6.14(b) through (d) show the H, S, and I components of this image, generated using Eqs. (6-16) through (6-19). Recall from the discussion earlier in this section that the gray-level values in Fig. 6.14(b) correspond to angles; thus, for example, because red corresponds to 0° , the red region in Fig. 6.14(a) is mapped to a black region in the hue image. Similarly, the gray levels in Fig. 6.14(c) correspond to saturation (they were scaled to $[0, 255]$ for display), and the gray levels in Fig. 6.14(d) are average intensities.

To change the individual color of any region in the RGB image, we change the values of the corresponding region in the hue image of Fig. 6.14(b). Then we convert

a	b
c	d

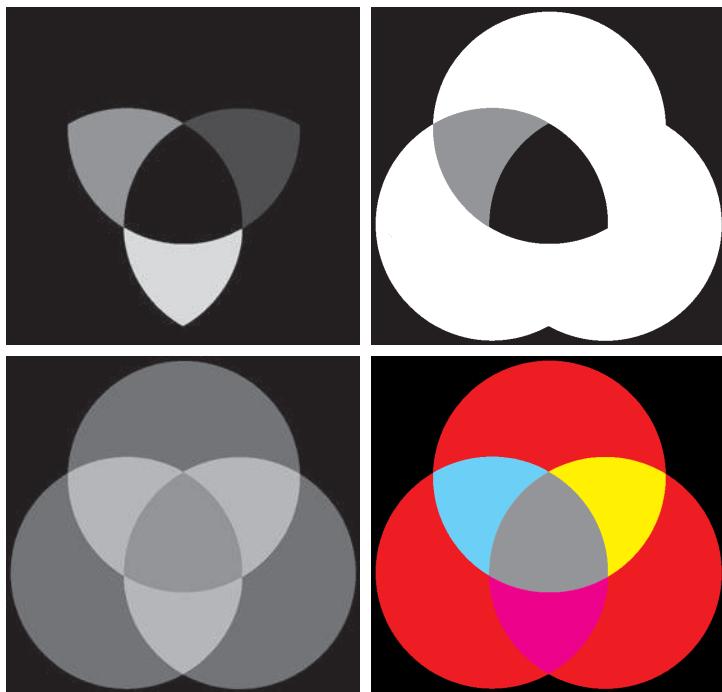
FIGURE 6.14
 (a) RGB image
 and the
 components of
 its corresponding
 HSI image:
 (b) hue,
 (c) saturation, and
 (d) intensity.



a	b
c	d

FIGURE 6.15

(a)-(c) Modified HSI component images.
 (d) Resulting RGB image. (See Fig. 6.14 for the original HSI images.)



the new H image, along with the unchanged S and I images, back to RGB using the procedure explained in Eqs. (6-20) through (6-30). To change the saturation (purity) of the color in any region, we follow the same procedure, except that we make the changes in the saturation image in HSI space. Similar comments apply to changing the average intensity of any region. Of course, these changes can be made simultaneously. For example, the image in Fig. 6.15(a) was obtained by changing to 0 the pixels corresponding to the blue and green regions in Fig. 6.14(b). In Fig. 6.15(b), we reduced by half the saturation of the cyan region in component image S from Fig. 6.14(c). In Fig. 6.15(c), we reduced by half the intensity of the central white region in the intensity image of Fig. 6.14(d). The result of converting this modified HSI image back to RGB is shown in Fig. 6.15(d). As expected, we see in this figure that the outer portions of all circles are now red; the purity of the cyan region was diminished, and the central region became gray rather than white. Although these results are simple, they clearly illustrate the power of the HSI color model in allowing independent control over hue, saturation, and intensity. These are quantities with which humans are quite familiar when describing colors.

A DEVICE INDEPENDENT COLOR MODEL

As noted earlier, humans see a broad spectrum of colors and color shades. However, color perception differs between individuals. Not only that, but color across devices such as monitors and printers can vary significantly unless these devices are properly calibrated.

Color transformations can be performed on most desktop computers. In conjunction with digital cameras, flatbed scanners, and ink-jet printers, they turn a personal computer into a *digital darkroom*. Also, commercial devices exist that use a combination of spectrometer measurements and software to develop color profiles that can then be loaded on monitors and printers to calibrate their color responses.

The effectiveness of the transformations examined in this section is judged ultimately in print. Because these transformations are developed, refined, and evaluated on monitors, it is necessary to maintain a high degree of color consistency between the monitors used and the eventual output devices. This is best accomplished with a device-independent color model that relates the color gamuts (see Section 6.1) of the monitors and output devices, as well as any other devices being used, to one another. The success of this approach depends on the quality of the color profiles used to map each device to the model, as well as the model itself. The model of choice for many color management systems (CMS) is the CIE $L^* a^* b^*$ model, also called CIELAB (CIE [1978], Robertson [1977]).

The $L^* a^* b^*$ color components are given by the following equations:

$$L^* = 116 \cdot h\left(\frac{Y}{Y_w}\right) - 16 \quad (6-31)$$

$$a^* = 500 \left[h\left(\frac{X}{X_w}\right) - h\left(\frac{Y}{Y_w}\right) \right] \quad (6-32)$$

and

$$b^* = 200 \left[h\left(\frac{Y}{Y_w}\right) - h\left(\frac{Z}{Z_w}\right) \right] \quad (6-33)$$

where

$$h(q) = \begin{cases} \sqrt[3]{q} & q > 0.008856 \\ 7.787q + 16 / 116 & q \leq 0.008856 \end{cases} \quad (6-34)$$

and X_w, Y_w , and Z_w are reference white tristimulus values—typically the white of a perfectly reflecting diffuser under CIE standard D65 illumination (defined by $x = 0.3127$ and $y = 0.3290$ in the CIE chromaticity diagram of Fig. 6.5). The $L^* a^* b^*$ color space is *colorimetric* (i.e., colors perceived as matching are encoded identically), *perceptually uniform* (i.e., color differences among various hues are perceived uniformly—see the classic paper by MacAdams [1942]), and *device independent*. While $L^* a^* b^*$ colors are not directly displayable (conversion to another color space is required), the $L^* a^* b^*$ gamut encompasses the entire visible spectrum and can represent accurately the colors of any display, print, or input device. Like the HSI system, the $L^* a^* b^*$ system is an excellent decoupler of intensity (represented by lightness L^*) and color (represented by a^* for red minus green and b^* for green minus blue), making it useful in both image manipulation (tone and contrast editing) and image compression applications. Studies indicate that the degree to which

the lightness information is separated from the color information in the $L^*a^*b^*$ system is greater than in any other color system (see Kasson and Plouffe [1972]). The principal benefit of calibrated imaging systems is that they allow tonal and color imbalances to be corrected interactively and independently—that is, in two sequential operations. Before color irregularities, like over- and under-saturated colors, are resolved, problems involving the image's tonal range are corrected. The tonal range of an image, also called its *key type*, refers to its general distribution of color intensities. Most of the information in high-key images is concentrated at high (or light) intensities; the colors of low-key images are located predominantly at low intensities; middle-key images lie in between. As in the monochrome case, it is often desirable to distribute the intensities of a color image equally between the highlights and the shadows. In Section 6.4, we give examples showing a variety of color transformations for the correction of tonal and color imbalances.

6.3 PSEUDOCOLOR IMAGE PROCESSING

Pseudocolor (sometimes called *false color*) image processing consists of assigning colors to gray values based on a specified criterion. The term pseudo or false color is used to differentiate the process of assigning colors to achromatic images from the processes associated with true color images, a topic discussed starting in Section 6.4. The principal use of pseudocolor is for human visualization and interpretation of grayscale events in an image or sequence of images. As noted at the beginning of this chapter, one of the principal motivations for using color is the fact that humans can discern thousands of color shades and intensities, compared to less than two dozen shades of gray.

INTENSITY SLICING AND COLOR CODING

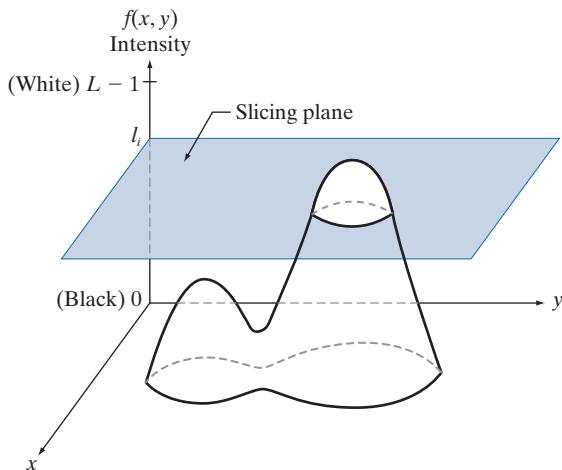
The techniques of *intensity* (sometimes called *density*) *slicing* and color coding are the simplest and earliest examples of pseudocolor processing of digital images. If an image is interpreted as a 3-D function [see Fig. 2.18(a)], the method can be viewed as one of placing planes parallel to the coordinate plane of the image; each plane then “slices” the function in the area of intersection. Figure 6.16 shows an example of using a plane at $f(x, y) = l_i$ to slice the image intensity function into two levels.

If a different color is assigned to each side of the plane in Fig. 6.16, any pixel whose intensity level is above the plane will be coded with one color, and any pixel below the plane will be coded with the other. Levels that lie on the plane itself may be arbitrarily assigned one of the two colors, or they could be given a third color to highlight all the pixels at that level. The result is a two- (or three-) color image whose relative appearance can be controlled by moving the slicing plane up and down the intensity axis.

In general, the technique for multiple colors may be summarized as follows. Let $[0, L - 1]$ represent the grayscale, let level l_0 represent black $[f(x, y) = 0]$, and level l_{L-1} represent white $[f(x, y) = L - 1]$. Suppose that P planes perpendicular to the intensity axis are defined at levels l_1, l_2, \dots, l_P . Then, assuming that $0 < P < L - 1$, the P planes partition the grayscale into $P + 1$ intervals, I_1, I_2, \dots, I_{P+1} . Intensity to color assignments at each pixel location (x, y) are made according to the equation

FIGURE 6.16

Graphical interpretation of the intensity-slicing technique.



$$\text{if } f(x, y) \in I_k, \text{ let } f(x, y) = c_k \quad (6-35)$$

where c_k is the color associated with the k th intensity interval I_k , defined by the planes at $l = k - 1$ and $l = k$.

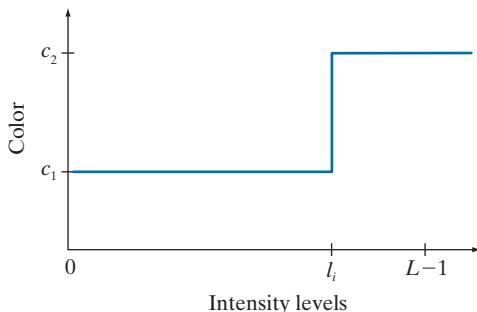
Figure 6.16 is not the only way to visualize the method just described. Figure 6.17 shows an equivalent approach. According to the mapping in this figure, any image intensity below level l_i is assigned one color, and any level above is assigned another. When more partitioning levels are used, the mapping function takes on a staircase form.

EXAMPLE 6.3: Intensity slicing and color coding.

A simple but practical use of intensity slicing is shown in Fig. 6.18. Figure 6.18(a) is a grayscale image of the Picker Thyroid Phantom (a radiation test pattern), and Fig. 6.18(b) is the result of intensity slicing this image into eight colors. Regions that appear of constant intensity in the grayscale image are actually quite variable, as shown by the various colors in the sliced image. For instance, the left lobe is a dull gray in the grayscale image, and picking out variations in intensity is difficult. By contrast, the color image

FIGURE 6.17

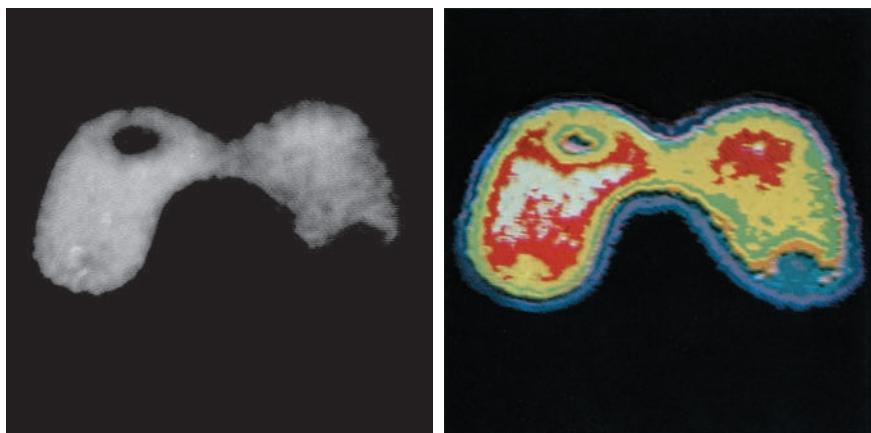
An alternative representation of the intensity-slicing technique.



a b

FIGURE 6.18

(a) Grayscale image of the Picker Thyroid Phantom.
 (b) Result of intensity slicing using eight colors.
 (Courtesy of Dr. J. L. Blankenship, Oak Ridge National Laboratory.)



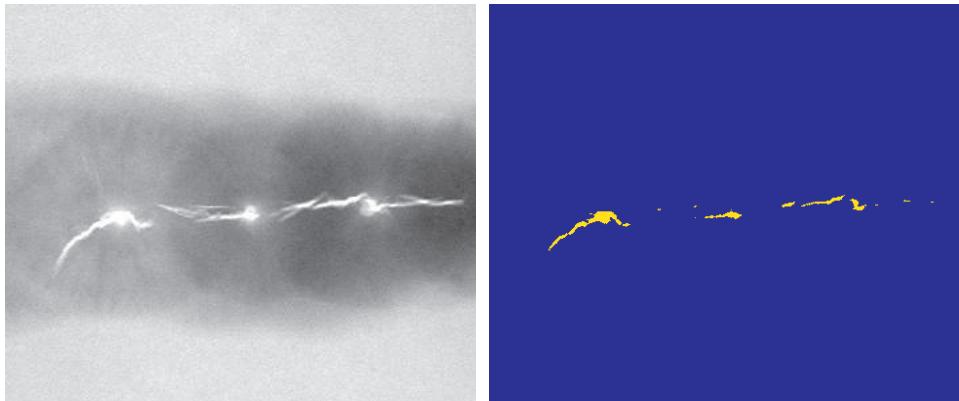
clearly shows eight different regions of constant intensity, one for each of the colors used. By varying the number of colors and the span of the intensity intervals, one can quickly determine the characteristics of intensity variations in a grayscale image. This is particularly true in situations such as the one shown here, in which the object of interest has uniform texture with intensity variations that are difficult to analyze visually. This example also illustrates the comments made in Section 6.1 about the eye's superior capability for detecting different color shades.

In the preceding simple example, the grayscale was divided into intervals and a different color was assigned to each, with no regard for the meaning of the gray levels in the image. Interest in that case was simply to view the different gray levels constituting the image. Intensity slicing assumes a much more meaningful and useful role when subdivision of the grayscale is based on physical characteristics of the image. For instance, Fig. 6.19(a) shows an X-ray image of a weld (the broad, horizontal dark region) containing several cracks and porosities (the bright streaks running horizontally through the middle of the image). When there is a porosity or crack in a weld, the full strength of the X-rays going through the object saturates the imaging sensor on the other side of the object. Thus, intensity values of 255 in an 8-bit image coming from such a system automatically imply a problem with the weld. If human visual analysis is used to inspect welds (still a common procedure today), a simple color coding that assigns

a b

FIGURE 6.19

(a) X-ray image of a weld.
 (b) Result of color coding. (Original image courtesy of X-TEK Systems, Ltd.)



one color to level 255 and another to all other intensity levels can simplify the inspector's job considerably. Figure 6.19(b) shows the result. No explanation is required to arrive at the conclusion that human error rates would be lower if images were displayed in the form of Fig. 6.19(b), instead of the form in Fig. 6.19(a). In other words, if an intensity value, or range of values, one is looking for is known, intensity slicing is a simple but powerful aid in visualization, especially if numerous images have to be inspected on a routine basis.

EXAMPLE 6.4: Use of color to highlight rainfall levels.

Measurement of rainfall levels, especially in the tropical regions of the Earth, is of interest in diverse applications dealing with the environment. Accurate measurements using ground-based sensors are difficult and expensive to acquire, and total rainfall figures are even more difficult to obtain because a significant portion of precipitation occurs over the ocean. One approach for obtaining rainfall figures remotely is to use satellites. The TRMM (Tropical Rainfall Measuring Mission) satellite utilizes, among others, three sensors specially designed to detect rain: a precipitation radar, a microwave imager, and a visible and infrared scanner (see Sections 1.3 and 2.3 regarding image sensing modalities).

The results from the various rain sensors are processed, resulting in estimates of average rainfall over a given time period in the area monitored by the sensors. From these estimates, it is not difficult to generate grayscale images whose intensity values correspond directly to rainfall, with each pixel representing a physical land area whose size depends on the resolution of the sensors. Such an intensity image is shown in Fig. 6.20(a), where the area monitored by the satellite is the horizontal band highlighted in the middle of the picture (these are tropical regions). In this particular example, the rainfall values are monthly averages (in inches) over a three-year period.

Visual examination of this picture for rainfall patterns is difficult and prone to error. However, suppose that we code intensity levels from 0 to 255 using the colors shown in Fig. 6.20(b). In this mode of intensity slicing, each slice is one of the colors in the color band. Values toward the blues signify low values of rainfall, with the opposite being true for red. Note that the scale tops out at pure red for values of rainfall greater than 20 inches. Figure 6.20(c) shows the result of color coding the grayscale image with the color map just discussed. The results are much easier to interpret, as shown in this figure and in the zoomed area of Fig. 6.20(d). In addition to providing global coverage, this type of data allows meteorologists to calibrate ground-based rain monitoring systems with greater precision than ever before.

INTENSITY TO COLOR TRANSFORMATIONS

Other types of transformations are more general, and thus are capable of achieving a wider range of pseudocolor enhancement results than the simple slicing technique discussed in the preceding section. Figure 6.21 shows an approach that is particularly attractive. Basically, the idea underlying this approach is to perform three independent transformations on the intensity of input pixels. The three results are then fed separately into the red, green, and blue channels of a color monitor. This method produces a composite image whose color content is modulated by the nature of the transformation functions.

The method for intensity slicing discussed in the previous section is a special case of the technique just described. There, piecewise linear functions of the intensity levels (see Fig. 6.17) are used to generate colors. On the other hand, the method

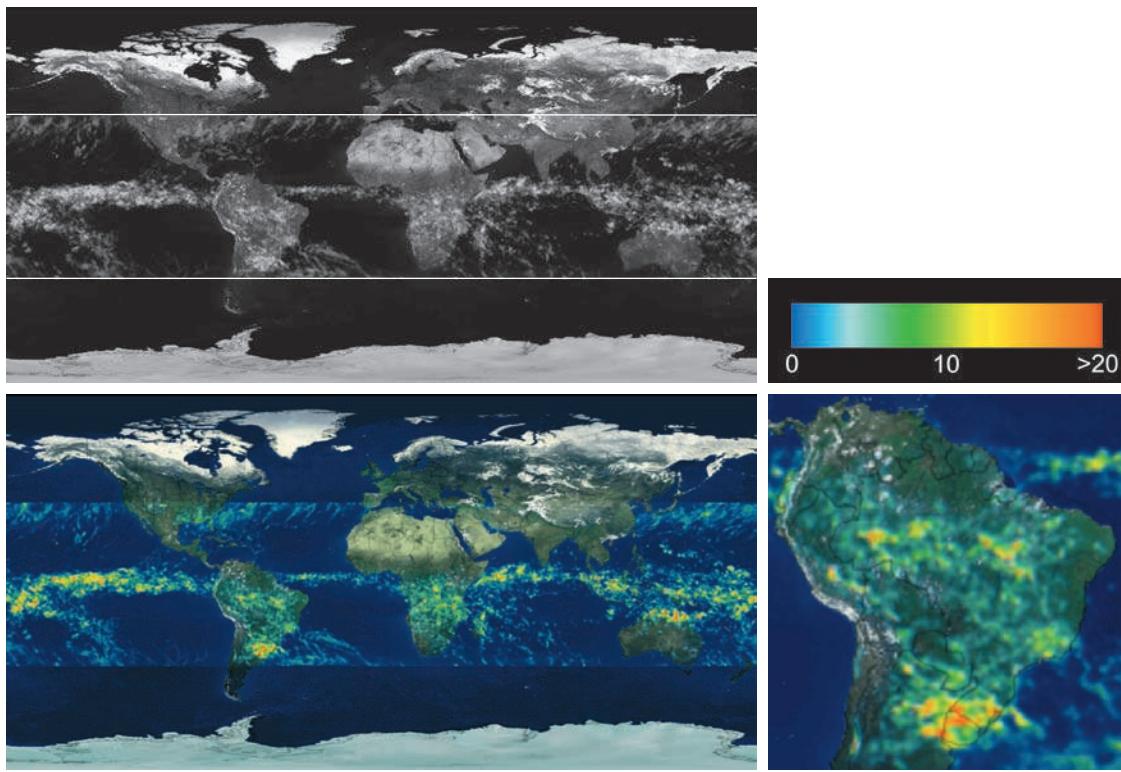
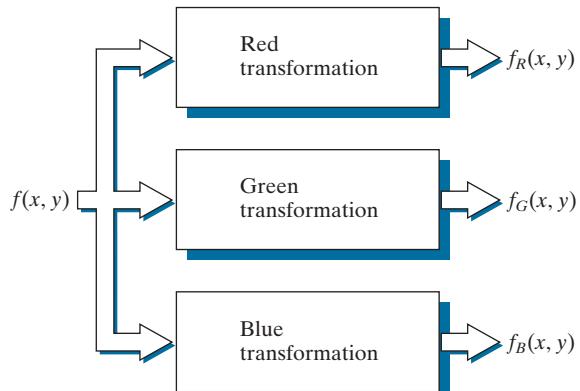


FIGURE 6.20 (a) Grayscale image in which intensity (in the horizontal band shown) corresponds to average monthly rainfall. (b) Colors assigned to intensity values. (c) Color-coded image. (d) Zoom of the South American region. (Courtesy of NASA.)

FIGURE 6.21
Functional block diagram for pseudocolor image processing. Images f_R , f_G , and f_B are fed into the corresponding red, green, and blue inputs of an RGB color monitor.



a
b c

FIGURE 6.22

Pseudocolor enhancement by using the gray level to color transformations in Fig. 6.23. (Original image courtesy of Dr. Mike Hurwitz, Westinghouse.)



discussed in this section can be based on smooth, nonlinear functions, which gives the technique considerable flexibility.

EXAMPLE 6.5: Using pseudocolor to highlight explosives in X-ray images.

Figure 6.22(a) shows two monochrome images of luggage obtained from an airport X-ray scanning system. The image on the left contains ordinary articles. The image on the right contains the same articles, as well as a block of simulated plastic explosives. The purpose of this example is to illustrate the use of intensity to color transformations to facilitate detection of the explosives.

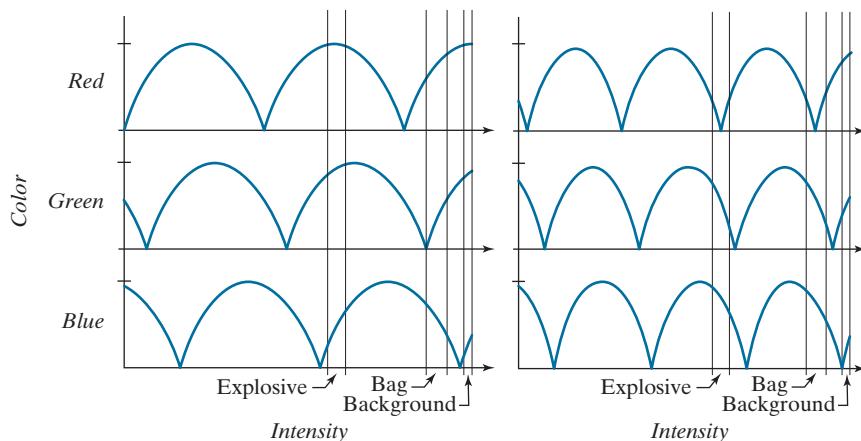
Figure 6.23 shows the transformation functions used. These sinusoidal functions contain regions of relatively constant value around the peaks as well as regions that change rapidly near the valleys. Changing the phase and frequency of each sinusoid can emphasize (in color) ranges in the grayscale. For instance, if all three transformations have the same phase and frequency, the output will be a grayscale image. A small change in the phase between the three transformations produces little change in pixels whose intensities correspond to peaks in the sinusoids, especially if the sinusoids have broad profiles (low frequencies). Pixels with intensity values in the steep section of the sinusoids are assigned a much stronger color content as a result of significant differences between the amplitudes of the three sinusoids caused by the phase displacement between them.

The image in Fig. 6.22(b) was obtained using the transformation functions in Fig. 6.23(a), which shows the gray-level bands corresponding to the explosive, garment bag, and background, respectively. Note that the explosive and background have quite different intensity levels, but they were both coded with approximately the same color as a result of the periodicity of the sine waves. The image in Fig. 6.22(c) was obtained with the transformation functions in Fig. 6.23(b). In this case, the explosives and garment bag intensity bands were mapped by similar transformations, and thus received essentially the same

a b

FIGURE 6.23

Transformation functions used to obtain the pseudocolor images in Fig. 6.22.



color assignments. Note that this mapping allows an observer to “see” through the explosives. The background mappings were about the same as those used for Fig. 6.22(b), producing almost identical color assignments for the two pseudocolor images.

The approach in Fig. 6.21 is based on a single grayscale image. Often, it is of interest to combine several grayscale images into a single color composite, as illustrated in Fig. 6.24. A frequent use of this approach is in multispectral image processing, where different sensors produce individual grayscale images, each in a different spectral band (see Example 6.6 below). The types of additional processing shown in Fig. 6.24 can be techniques such as color balancing and spatial filtering, as discussed later in this chapter. When coupled with background knowledge about the physical characteristics of each band, color-coding in the manner just explained is a powerful aid for human visual analysis of complex multispectral images.

EXAMPLE 6.6: Color coding of multispectral images.

Figures 6.25(a) through (d) show four satellite images of the Washington, D.C., area, including part of the Potomac River. The first three images are in the visible red (R), green (G), and blue (B) bands, and

FIGURE 6.24

A pseudocolor coding approach using multiple grayscale images. The inputs are grayscale images. The outputs are the three components of an RGB composite image.

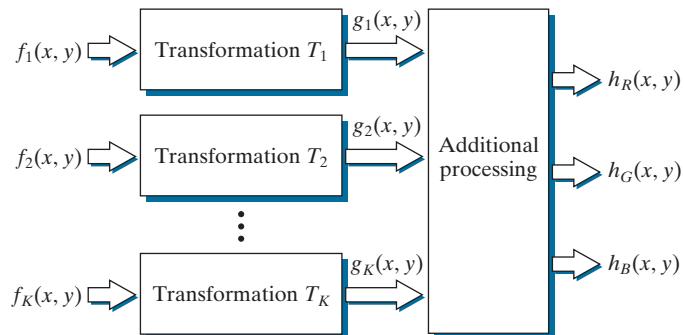




FIGURE 6.25 (a)–(d) Red (R), green (G), blue (B), and near-infrared (IR) components of a LANDSAT multispectral image of the Washington, D.C. area. (e) RGB color composite image obtained using the IR, G, and B component images. (f) RGB color composite image obtained using the R, IR, and B component images. (Original multispectral images courtesy of NASA.)

the fourth is in the near infrared (IR) band (see Table 1.1 and Fig. 1.10). The latter band is responsive to the biomass content of a scene, and we want to use this fact to create a composite RGB color image in which vegetation is emphasized and the other components of the scene are displayed in more muted tones.

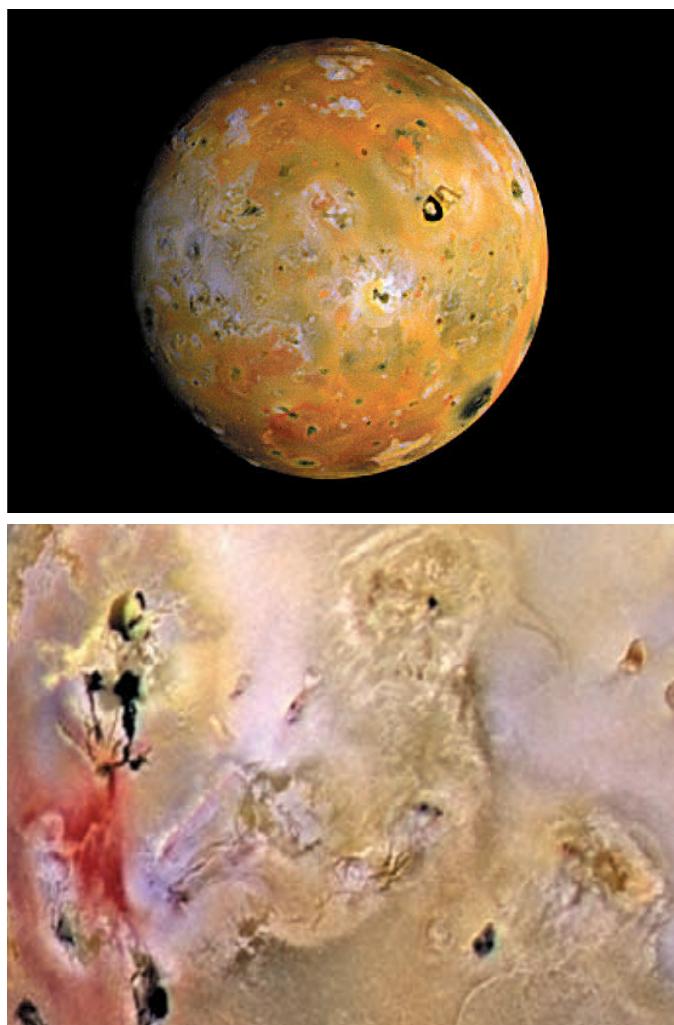
Figure 6.25(e) is an RGB composite obtained by replacing the red image by infrared. As you see, vegetation shows as a bright red, and the other components of the scene, which had a weaker response in the near-infrared band, show in pale shades of blue-green. Figure 6.25(f) is a similar image, but with the green replaced by infrared. Here, vegetation shows in a bright green color, and the other components of the scene show in purplish color shades, indicating that their major components are in the red and blue bands. Although the last two images do not introduce any new physical information, these images are much easier to interpret visually once it is known that the dominant component of the images are pixels of areas heavily populated by vegetation.

The type of processing just illustrated uses the physical characteristics of a single band in a multispectral image to emphasize areas of interest. The same approach can help visualize events of interest

a
b

FIGURE 6.26

(a) Pseudocolor rendition of Jupiter Moon Io.
(b) A close-up.
(Courtesy of NASA.)



in complex images in which the events are beyond human visual sensing capabilities. Figure 6.26 is an excellent illustration of this. These are images of the Jupiter moon Io, shown in pseudocolor by combining several of the sensor images from the Galileo spacecraft, some of which are in spectral regions not visible to the eye. However, by understanding the physical and chemical processes likely to affect sensor response, it is possible to combine the sensed images into a meaningful pseudocolor map. One way to combine the sensed image data is by how they show either differences in surface chemical composition or changes in the way the surface reflects sunlight. For example, in the pseudocolor image in Fig. 6.26(b), bright red depicts material newly ejected from an active volcano on Io, and the surrounding yellow materials are older sulfur deposits. This image conveys these characteristics much more readily than would be possible by analyzing the component images individually.

6.4 BASICS OF FULL-COLOR IMAGE PROCESSING

In this section, we begin the study of processing methods for full-color images. The techniques developed in the sections that follow are illustrative of how full-color images are handled for a variety of image processing tasks. Full-color image processing approaches fall into two major categories. In the first category, we process each grayscale component image individually, then form a composite color image from the individually processed components. In the second category, we work with color pixels directly. Because full-color images have at least three components, color pixels are vectors. For example, in the RGB system, each color point can be interpreted as a vector extending from the origin to that point in the RGB coordinate system (see Fig. 6.7).

Let \mathbf{c} represent an arbitrary vector in RGB color space:

$$\mathbf{c} = \begin{bmatrix} c_R \\ c_G \\ c_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (6-36)$$

This equation indicates that the components of \mathbf{c} are the RGB components of a color image at a point. We take into account the fact that the colors of the pixels in an image are a function of spatial coordinates (x, y) by using the notation

$$\mathbf{c}(x, y) = \begin{bmatrix} c_R(x, y) \\ c_G(x, y) \\ c_B(x, y) \end{bmatrix} = \begin{bmatrix} R(x, y) \\ G(x, y) \\ B(x, y) \end{bmatrix} \quad (6-37)$$

For an image of size $M \times N$, there are MN such vectors, $\mathbf{c}(x, y)$, for $x = 0, 1, 2, \dots, M - 1$ and $y = 0, 1, 2, \dots, N - 1$.

Equation (6-37) depicts a vector whose components are *spatial* variables x and y . This is a frequent source of confusion that can be avoided by focusing on the fact that our interest lies in spatial processes. That is, we are interested in image processing techniques formulated in x and y . The fact that the pixels are now color pixels introduces a factor that, in its easiest formulation, allows us to process a color image by processing each of its component images separately, using standard grayscale image processing methods. However, the results of individual color component processing are not always equivalent to direct processing in color vector space, in which case we must use approaches for processing the elements of color points directly. When these points have more than two components, we call them *voxels*. We use the terms vectors, points, and voxels interchangeably when the meaning is clear that we are referring to images composed of more than one 2-D image.

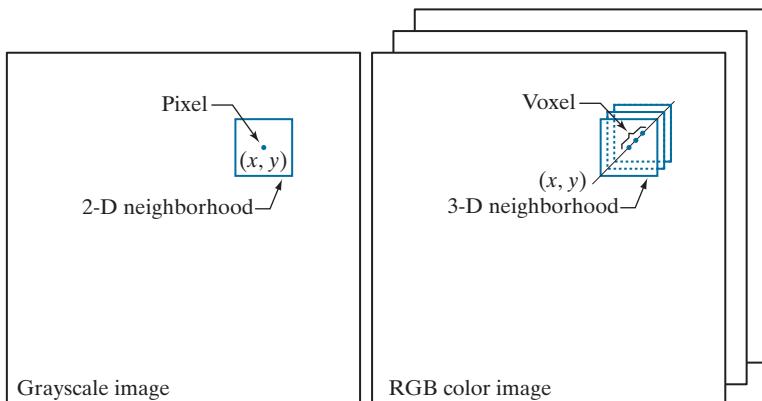
In order for per-component-image and vector-based processing to be equivalent, two conditions have to be satisfied: first, the process has to be applicable to both vectors and scalars; second, the operation on each component of a vector (i.e., each voxel) must be independent of the other components. As an illustration, Fig. 6.27 shows spatial neighborhood processing of grayscale and full-color images. Suppose

Although an RGB image is composed of three grayscale component images, pixels in all three images are registered spatially. That is, a single pair of spatial coordinates, (x, y) , addresses the same pixel location in all three images, as illustrated in Fig. 6.27(b) below.

a b

FIGURE 6.27

Spatial neighborhoods for grayscale and RGB color images. Observe in (b) that a single pair of spatial coordinates, (x, y) , addresses the same spatial location in all three images.



that the process is neighborhood averaging. In Fig. 6.27(a), averaging would be done by summing the intensities of all the pixels in the 2-D neighborhood, then dividing the result by the total number of pixels in the neighborhood. In Fig. 6.27(b), averaging would be done by summing all the voxels in the 3-D neighborhood, then dividing the result by the total number of voxels in the neighborhood. Each of the three component of the average voxel is the sum of the pixels in the single image neighborhood centered on that location. But the same result would be obtained if the averaging were done on the pixels of each image, independently, and then the sum of the three values were added for each. Thus, spatial neighborhood averaging can be carried out on a per-component-image or directly on RGB image voxels. The results would be the same. In the following sections we develop methods for which the per-component-image approach is suitable, and methods for which it is not.

6.5 COLOR TRANSFORMATIONS

The techniques described in this section, collectively called *color transformations*, deal with processing the components of a color image within the context of a *single* color model, as opposed to color transformations *between* color models, as in Section 6.2.

FORMULATION

As with the intensity transformation techniques of Chapter 3, we model color transformations for multispectral images using the general expression

$$s_i = T_i(r_i) \quad i = 1, 2, \dots, n \quad (6-38)$$

where n is the total number of component images, r_i are the intensity values of the input component images, s_i are the spatially corresponding intensities in the output component images, and T_i are a set of *transformation* or *color mapping functions* that operate on r_i to produce s_i . Equation (6-38) is applied individually to all pixels in the input image. For example, in the case of RGB color images, $n = 3$, r_1, r_2, r_3 are the intensities values at a point in the input components images, and s_1, s_2, s_3 are

the corresponding transformed pixels in the output image. The fact that i is also a subscript on T means that, in principle, we can implement a different transformation for each input component image.

As an illustration, the first row of Fig. 6.28 shows a full color CMYK image of a simple scene, and the second row shows its four component images, all normalized to the range $[0, 1]$. We see that the strawberries are composed of large amounts of magenta and yellow because the images corresponding to these two CMYK components are the brightest. Black is used sparingly and is generally confined to the coffee and shadows within the bowl of strawberries. The fourth row shows the equivalent RGB images obtained from the CMYK images using Eqs. (6-13)-(6-15). Here we see that the strawberries contain a large amount of red and very little (although some) green and blue. From the RGB images, we obtained the CMY images in the third row using Eq. (6-5). Note that these CMY images are slightly different from the CMY images in the row above them. This is because the CMY images in these two systems are different as a result of using K in one of them. The last row of Fig. 6.28 shows the HSI components, obtained from the RGB images using Eqs. (6-16)-(6-19). As expected, the intensity (I) component is a grayscale rendition of the full-color original. The saturation image (S) is as expected also. The strawberries are relatively pure in color; as a result, they show the highest saturation (least dilution by white light) values of any of the other elements of the image. Finally, we note some difficulty in interpreting the values of the hue (H) component image. The problem is that (1) there is a discontinuity in the HSI model where 0° and 360° meet [see Fig. 6.13(a)], and (2) hue is undefined for a saturation of 0 (i.e., for white, black, and pure grays). The discontinuity of the model is most apparent around the strawberries, which are depicted in gray level values near both black (0) and white (1). The result is an unexpected mixture of highly contrasting gray levels to represent a single color—red.

We can apply Eq. (6-38) to any of the color-space component images in Fig. 6.28. In theory, any transformation can be performed in any color model. In practice, however, some operations are better suited to specific models. For a given transformation, the effects of converting between representations must be factored into the decision regarding the color space in which to implement it. For example, suppose that we wish to modify the intensity of the full-color image in the first row of Fig. 6.28 by a constant value, k in the range $[0, 1]$. In the HSI color space we need to modify only the intensity component image:

$$s_3 = kr_3 \quad (6-39)$$

and we let $s_1 = r_1$ and $s_2 = r_2$. In terms of our earlier discussion note that we are using two different transformation functions: T_1 and T_2 are identity transformations, and T_3 is a constant transformation.

In the RGB color space we need to modify all three components by the same constant transformation:

$$s_i = kr_i \quad i = 1, 2, 3 \quad (6-40)$$



Full color image



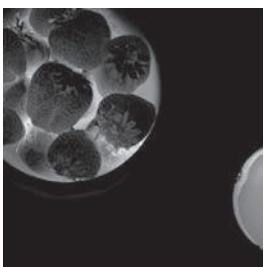
Cyan



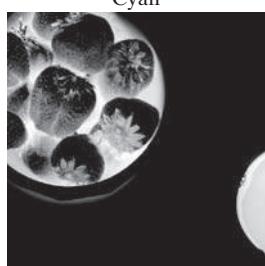
Magenta



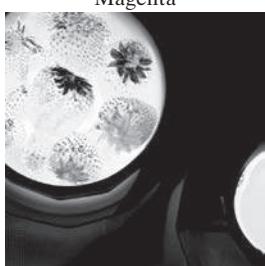
Yellow



Black



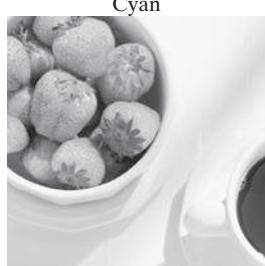
Cyan



Magenta



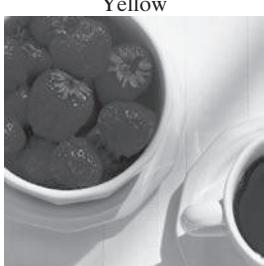
Yellow



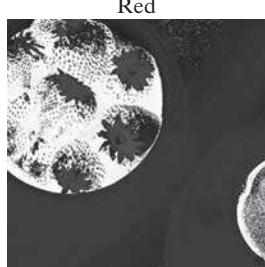
Red



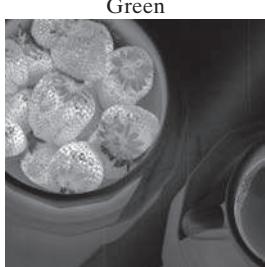
Green



Blue



Hue



Saturation



Intensity

FIGURE 6.28 A full-color image and its various color-space components. (Original image courtesy of MedData Interactive.)

The CMY space requires a similar set of linear transformations (see Problem 6.16):

$$s_i = kr_i + (1 - k) \quad i = 1, 2, 3 \quad (6-41)$$

Similarly, the transformations required to change the intensity of the CMYK image is given by

$$s_i = \begin{cases} r_i & i = 1, 2, 3 \\ kr_i + (1 - k) & i = 4 \end{cases} \quad (6-42)$$

This equation tells us that to change the intensity of a CMYK image, we only change the fourth (K) component.

Figure 6.29(b) shows the result of applying the transformations in Eqs. (6-39) through (6-42) to the full-color image of Fig. 6.28, using $k = 0.7$. The mapping functions themselves are shown graphically in Figs. 6.29(c) through (h). Note that the mapping function for CMYK consist of two parts, as do the functions for HSI; one of the transformations handles one component, and the other does the rest. Although

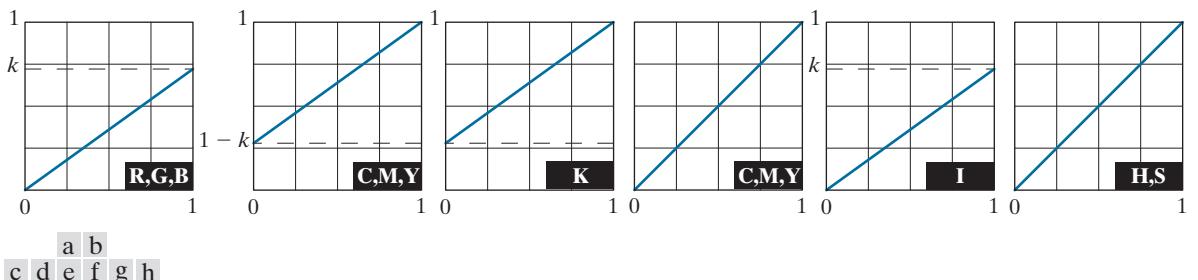


FIGURE 6.29 Adjusting the intensity of an image using color transformations. (a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting $k = 0.7$). (c) The required RGB mapping function. (d)–(e) The required CMY mapping functions. (f) The required CMY mapping function. (g)–(h) The required HSI mapping functions. (Original image courtesy of MedData Interactive.)

we used several different transformations, the net result of changing the intensity of the color by a constant value was the same for all.

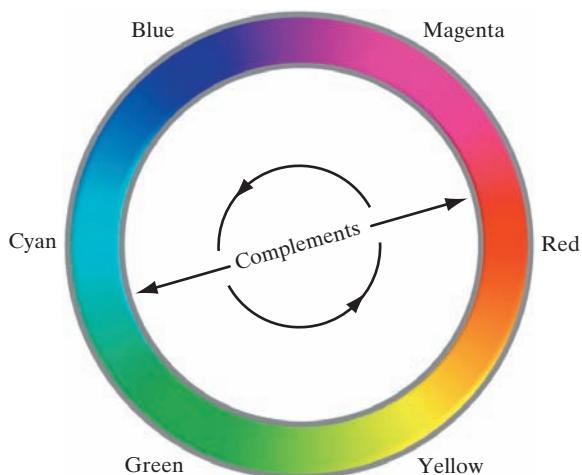
It is important to note that each transformation defined in Eqs. (6-39) through (6-42) depends only on one component within its color space. For example, the red output component, s_1 , in Eq. (6-40) is independent of the green (r_2) and blue (r_3) inputs; it depends only on the red (r_1) input. Transformations of this type are among the simplest and most frequently used color processing tools. They can be carried out on a per-color-component basis, as mentioned at the beginning of our discussion. In the remainder of this section, we will examine several such transformations and discuss a case in which the component transformation functions are dependent on all the color components of the input image and, therefore, cannot be done on an individual color-component basis.

COLOR COMPLEMENTS

The *color circle* (also called the *color wheel*) shown in Fig. 6.30 originated with Sir Isaac Newton, who in the seventeenth century created its first form by joining the ends of the color spectrum. The color circle is a visual representation of colors that are arranged according to the chromatic relationship between them. The circle is formed by placing the primary colors equidistant from each other. Then, the secondary colors are placed between the primaries, also in an equidistant arrangement. The net result is that hues directly opposite one another on the color circle are *complements*. Our interest in complements stems from the fact that they are analogous to the grayscale negatives we studied in Section 3.2. As in the grayscale case, color complements are useful for enhancing detail that is embedded in dark regions of a color image—particularly when the regions are dominant in size. The following example illustrates some of these concepts.

FIGURE 6.30

Color complements on the color circle.



EXAMPLE 6.7: Computing color image complements.

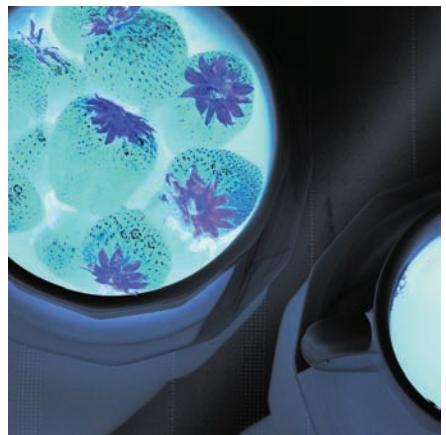
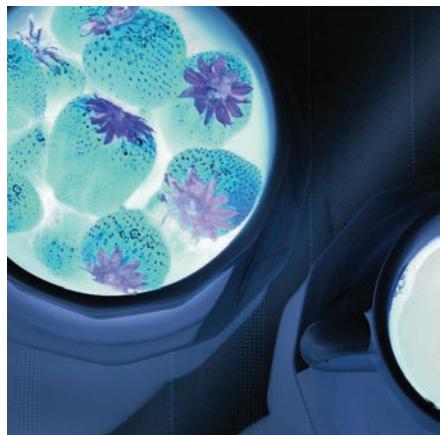
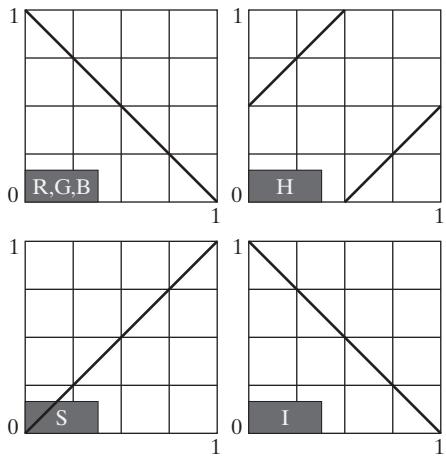
Figures 6.31(a) and (c) show the full-color image from Fig. 6.28 and its color complement. The RGB transformations used to compute the complement are plotted in Fig. 6.31(b). They are identical to the grayscale negative transformation defined in Section 3.2. Note that the complement is reminiscent of conventional photographic color film negatives. Reds of the original image are replaced by cyans in the complement. When the original image is black, the complement is white, and so on. Each of the hues in the complement image can be predicted from the original image using the color circle of Fig. 6.30, and each of the RGB component transforms involved in the computation of the complement is a function of only the corresponding input color component.

Unlike the intensity transformations of Fig. 6.29, the RGB complement transformation functions used in this example do not have a straightforward HSI equivalent. It is left as an exercise (see Problem 6.19) to show that the saturation component of the complement cannot be computed from the saturation component of the input image alone. Figure 6.31(d) shows an approximation of the complement using the hue, saturation, and intensity transformations in Fig. 6.31(b). The saturation component of the input image is unaltered; it is responsible for the visual differences between Figs. 6.31(c) and (d).

a
b
c
d

FIGURE 6.31

Color complement transformations.
 (a) Original image.
 (b) Complement transformation functions.
 (c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.



COLOR SLICING

Highlighting a specific range of colors in an image is useful for separating objects from their surroundings. The basic idea is either to: (1) display the colors of interest so that they stand out from the background; or (2) use the region defined by the colors as a mask for further processing. The most straightforward approach is to extend the intensity slicing techniques of Section 3.2. However, because a color pixel is an n -dimensional quantity, the resulting color transformation functions are more complicated than their grayscale counterparts in Fig. 3.11. In fact, the required transformations are more complex than the color component transforms considered thus far. This is because all practical color-slicing approaches require each pixel's transformed color components to be a function of all n original pixel's color components.

One of the simplest ways to “slice” a color image is to map the colors outside some range of interest into a nonprominent neutral color. If the colors of interest are enclosed by a cube (or *hypercube* for $n > 3$) of width W and centered at a prototypical (e.g., average) color with components (a_1, a_2, \dots, a_n) , the necessary set of transformations are given by

$$s_i = \begin{cases} 0.5 & \text{if } \left[\left| r_j - a_j \right| > \frac{W}{2} \right]_{\text{any } 1 \leq j \leq n} \\ r_i & \text{otherwise} \end{cases} \quad i = 1, 2, \dots, n \quad (6-43)$$

These transformations highlight the colors around the prototype by forcing all other colors to the midpoint of the reference color space (this is an arbitrarily chosen neutral point). For the RGB color space, for example, a suitable neutral point is middle gray or color $(0.5, 0.5, 0.5)$.

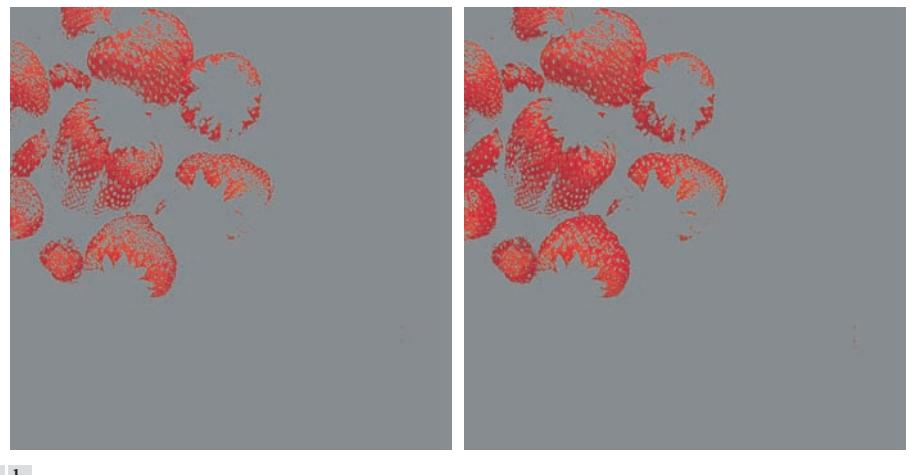
If a sphere is used to specify the colors of interest, Eq. (6-43) becomes

$$s_i = \begin{cases} 0.5 & \text{if } \sum_{j=1}^n (r_j - a_j)^2 > R_0^2 \\ r_i & \text{otherwise} \end{cases} \quad i = 1, 2, \dots, n \quad (6-44)$$

Here, R_0 is the radius of the enclosing sphere (or hypersphere for $n > 3$) and (a_1, a_2, \dots, a_n) are the components of its center (i.e., the prototypical color). Other useful variations of Eqs. (6-43) and (6-44) include implementing multiple color prototypes and reducing the intensity of the colors outside the region of interest—rather than setting them to a neutral constant.

EXAMPLE 6.8: Color slicing.

Equations (6-43) and (6-44) can be used to separate the strawberries in Fig. 6.29(a) from their sepals, cup, bowl, and other background elements. Figures 6.32(a) and (b) show the results of using both transformations. In each case, a prototype red with RGB color coordinate $(0.6863, 0.1608, 0.1922)$ was selected from the most prominent strawberry. Parameters W and R_0 were chosen so that the highlighted region would not expand to other portions of the image. The actual values used, $W = 0.2549$ and $R_0 = 0.1765$, were determined interactively. Note that the sphere-based transformation of Eq. (6-44) performed slightly better, in the sense that it includes more of the strawberries' red areas. A sphere of radius 0.1765



a b

FIGURE 6.32 Color-slicing transformations that detect (a) reds within an RGB cube of width $W = 0.2549$ centered at $(0.6863, 0.1608, 0.1922)$, and (b) reds within an RGB sphere of radius 0.1765 centered at the same point. Pixels outside the cube and sphere were replaced by color $(0.5, 0.5, 0.5)$.

does not completely enclose a cube of width 0.2549, but it is not small enough to be completely enclosed by the cube either. In Section 6.7, and later in Chapter 10, you will learn more advanced techniques for using color and other multispectral information to extract objects from their background.

TONE AND COLOR CORRECTIONS

Problems involving an image's tonal range need to be corrected before color irregularities, such as over- and under-saturated colors, can be resolved. The tonal range of an image, also called its *key type*, refers to its general distribution of color intensities. Most of the information in *high-key* images is concentrated at high (or light) intensities; the colors of *low-key* images are located predominantly at low intensities; and *middle-key* images lie in between. As in the grayscale case, it is often desirable to distribute the intensities of a color image equally between the highlights and the shadows. The following examples illustrate a variety of color transformations for the correction of tonal and color imbalances.

EXAMPLE 6.9: Tonal transformations.

Transformations for modifying image tones normally are selected interactively. The idea is to adjust experimentally the image's brightness and contrast to provide maximum detail over a suitable range of intensities. The colors themselves are not changed. In the RGB and CMY(K) spaces, this means mapping all the color components, except K , with the same transformation function (see Fig. 6.29); in the HSI color space, only the intensity component is modified, as noted in the previous section.

Figure 6.33 shows typical RGB transformations used for correcting three common tonal imbalances—flat, light, and dark images. The S-shaped curve in the first row of the figure is ideal for boosting contrast

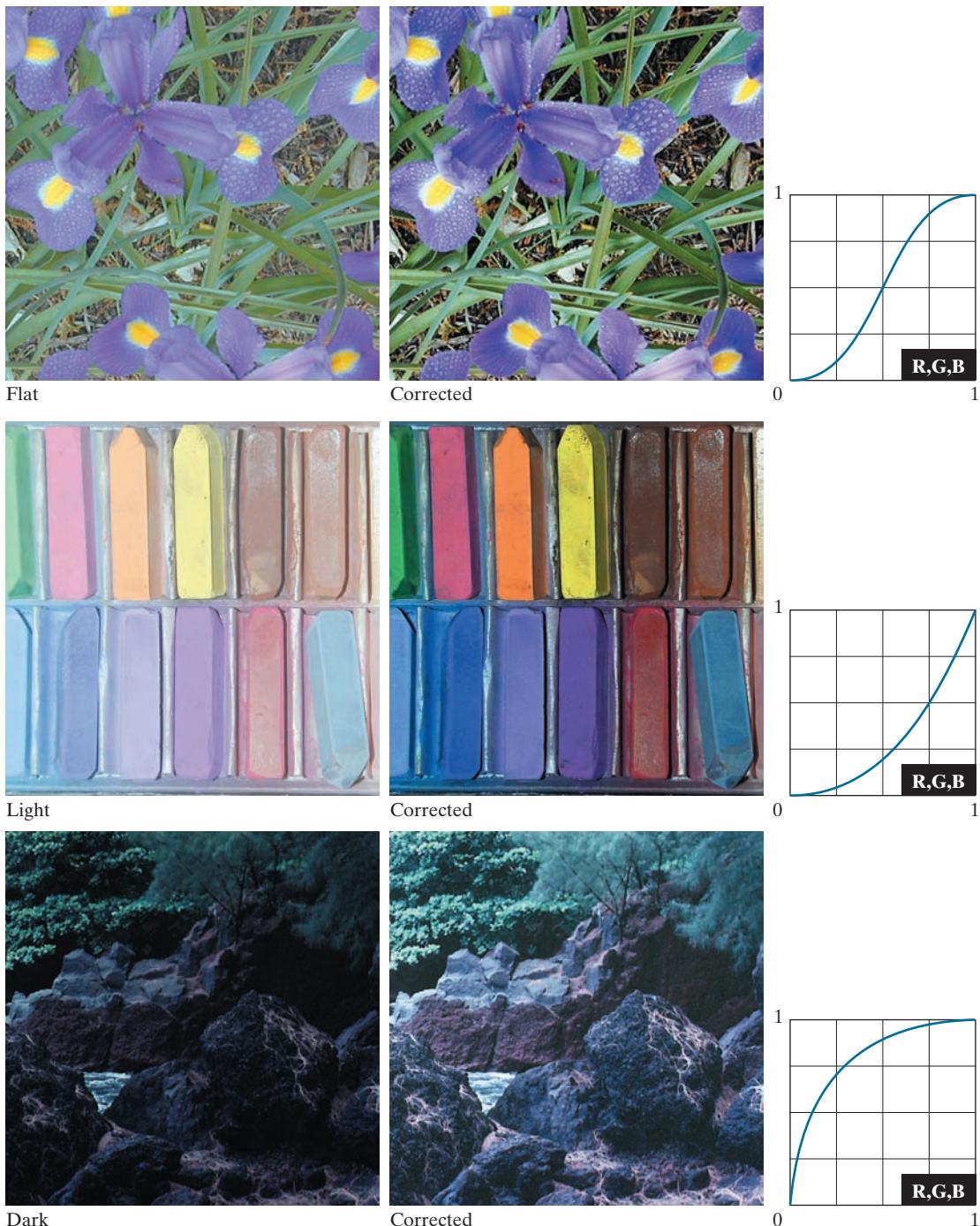


FIGURE 6.33 Tonal corrections for flat, light (high key), and dark (low key) color images. Adjusting the red, green, and blue components equally does not always alter the image hues significantly.

[see Fig. 3.2(a)]. Its midpoint is anchored so that highlight and shadow areas can be lightened and darkened, respectively. (The inverse of this curve can be used to correct excessive contrast.) The transformations in the second and third rows of the figure correct light and dark images, and are reminiscent of the power-law transformations in Fig. 3.6. Although the color components are discrete, as are the actual transformation functions, the transformation functions themselves are displayed and manipulated as continuous quantities—typically constructed from piecewise linear or higher order (for smoother mappings) polynomials. Note that the keys of the images in Fig. 6.33 are visually evident; they could also be determined using the histograms of the images' color components.

EXAMPLE 6.10: Color balancing.

Any color imbalances are addressed after the tonal characteristics of an image have been corrected. Although color imbalances can be determined directly by analyzing a known color in an image with a color spectrometer, accurate visual assessments are possible when white areas, where the RGB or CMY(K) components should be equal, are present. As Fig. 6.34 shows, skin tones are excellent subjects for visual color assessments because humans are highly perceptive of proper skin color. Vivid colors, such as bright red objects, are of little value when it comes to visual color assessment.

There are a variety of ways to correct color imbalances. When adjusting the color components of an image, it is important to realize that every action affects its overall color balance. That is, the perception of one color is affected by its surrounding colors. The color wheel of Fig. 6.30 can be used to predict how one color component will affect others. Based on the color wheel, for example, the proportion of any color can be increased by decreasing the amount of the opposite (or complementary) color in the image. Similarly, it can be increased by raising the proportion of the two immediately adjacent colors or decreasing the percentage of the two colors adjacent to the complement. Suppose, for instance, that there is too much magenta in an RGB image. It can be decreased: (1) by removing both red and blue, or (2) by adding green.

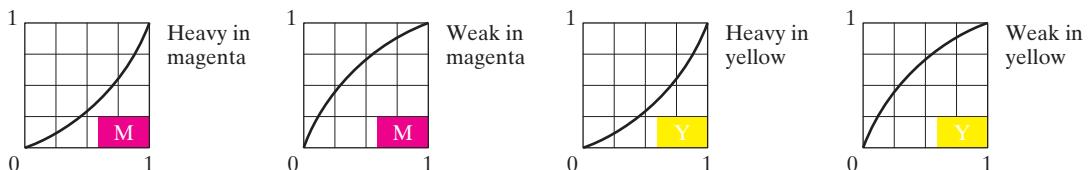
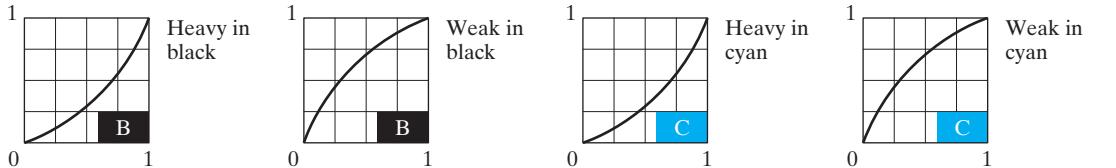
Figure 6.34 shows the transformations used to correct simple CMYK output imbalances. Note that the transformations depicted are the functions required for correcting the images; the inverses of these functions were used to generate the associated color imbalances. Together, the images are analogous to a color ring-around print of a darkroom environment and are useful as a reference tool for identifying color printing problems. Note, for example, that too much red can be due to excessive magenta (per the bottom left image) or too little cyan (as shown in the rightmost image of the second row).

HISTOGRAM PROCESSING OF COLOR IMAGES

Unlike the interactive enhancement approaches of the previous section, the gray-level histogram processing transformations of Section 3.3 can be applied to color images in an automated way. Recall that histogram equalization automatically determines a transformation that seeks to produce an image with a uniform histogram of intensity values. We showed in Section 3.3 that histogram processing can be quite successful at handling low-, high-, and middle-key images (for example, see Fig. 3.20). As you might suspect, it is generally unwise to histogram equalize the component images of a color image independently. This results in erroneous color. A more logical approach is to spread the color intensities uniformly, leaving the colors themselves (e.g., hues) unchanged. The following example shows that the HSI color space is ideally suited to this type of approach.

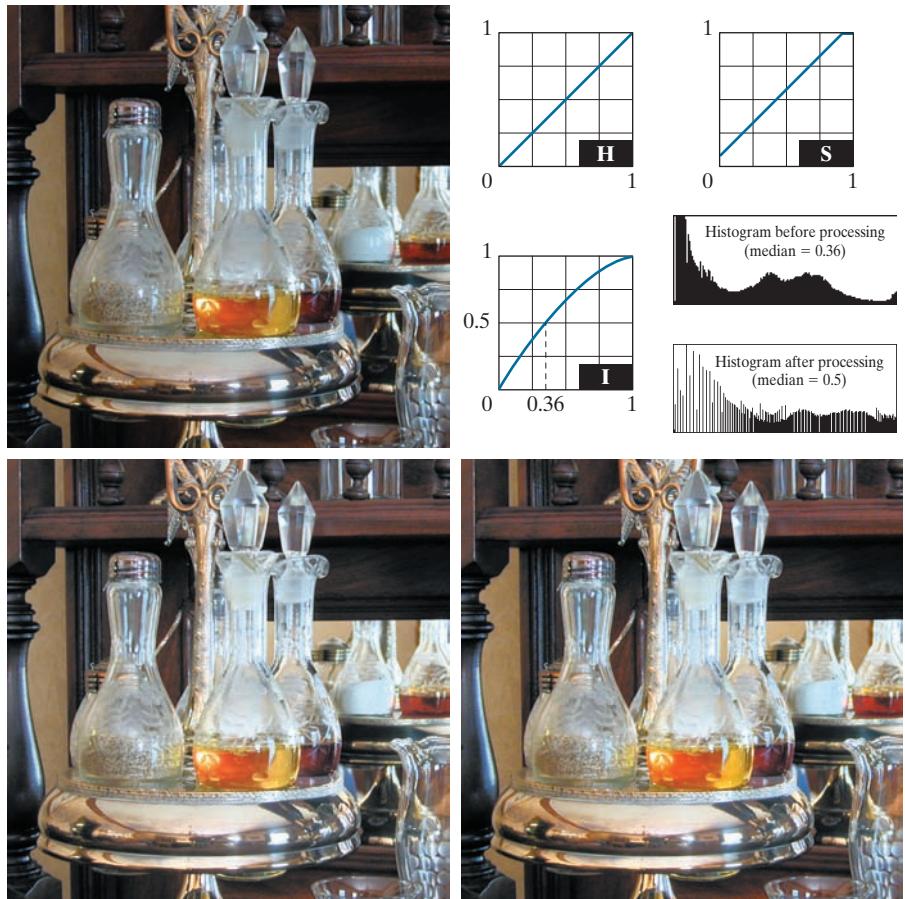


Original/Corrected

**FIGURE 6.34** Color balancing a CMYK image.

a	b
c	d

FIGURE 6.35
Histogram equalization (followed by saturation adjustment) in the HSI color space.



EXAMPLE 6.11: Histogram equalization in the HSI color space.

Figure 6.35(a) shows a color image of a caster stand containing cruets and shakers whose intensity component spans the entire (normalized) range of possible values, $[0, 1]$. As can be seen in the histogram of its intensity component prior to processing [see Fig. 6.35(b)], the image contains a large number of dark colors that reduce the median intensity to 0.36. Histogram equalizing the intensity component, without altering the hue and saturation, resulted in the image shown in Fig. 6.35(c). Note that the overall image is significantly brighter, and that several moldings and the grain of the wooden table on which the caster is sitting are now visible. Figure 6.35(d) shows the result of partially correcting this by increasing the image's saturation component, subsequent to histogram equalization, using the transformation in Fig. 6.35(b). This type of

Although intensity equalization did not alter the values of hue and saturation of the image, it did impact the overall color perception. Note, in particular, the loss of vibrancy in the oil and vinegar in the cruets. Figure 6.35(d) shows the result of partially correcting this by increasing the image's saturation component, subsequent to histogram equalization, using the transformation in Fig. 6.35(b). This type of

adjustment is common when working with the intensity component in HSI space because changes in intensity usually affect the relative appearance of colors in an image.

6.6 COLOR IMAGE SMOOTHING AND SHARPENING

The next step beyond transforming each pixel of a color image without regard to its neighbors (as in the previous section) is to modify its value based on the characteristics of the surrounding pixels. In this section, the basics of this type of neighborhood processing will be illustrated within the context of color image smoothing and sharpening.

COLOR IMAGE SMOOTHING

With reference to Fig. 6.27(a) and the discussion in Sections 3.4 and 3.5, grayscale image smoothing can be viewed as a spatial filtering operation in which the coefficients of the filtering kernel have the same value. As the kernel is slid across the image to be smoothed, each pixel is replaced by the average of the pixels in the neighborhood encompassed by the kernel. As Fig. 6.27(b) shows, this concept is easily extended to the processing of full-color images. The principal difference is that instead of scalar intensity values, we must deal with component vectors of the form given in Eq. (6-37).

Let S_{xy} denote the set of coordinates defining a neighborhood centered at (x, y) in an RGB color image. The average of the RGB component vectors in this neighborhood is

$$\bar{\mathbf{c}}(x, y) = \frac{1}{K} \sum_{(s,t) \in S_{xy}} \mathbf{c}(s, t) \quad (6-45)$$

It follows from Eq. (6-37) and the properties of vector addition that

$$\bar{\mathbf{c}}(x, y) = \begin{bmatrix} \frac{1}{K} \sum_{(s,t) \in S_{xy}} R(s, t) \\ \frac{1}{K} \sum_{(s,t) \in S_{xy}} G(s, t) \\ \frac{1}{K} \sum_{(s,t) \in S_{xy}} B(s, t) \end{bmatrix} \quad (6-46)$$

We recognize the components of this vector as the scalar images that would be obtained by independently smoothing each plane of the original RGB image using conventional grayscale neighborhood processing. Thus, we conclude that smoothing by neighborhood averaging can be carried out on a per-color-plane basis. The result is the same as when the averaging is performed using RGB color vectors.

a
b
c
d

FIGURE 6.36

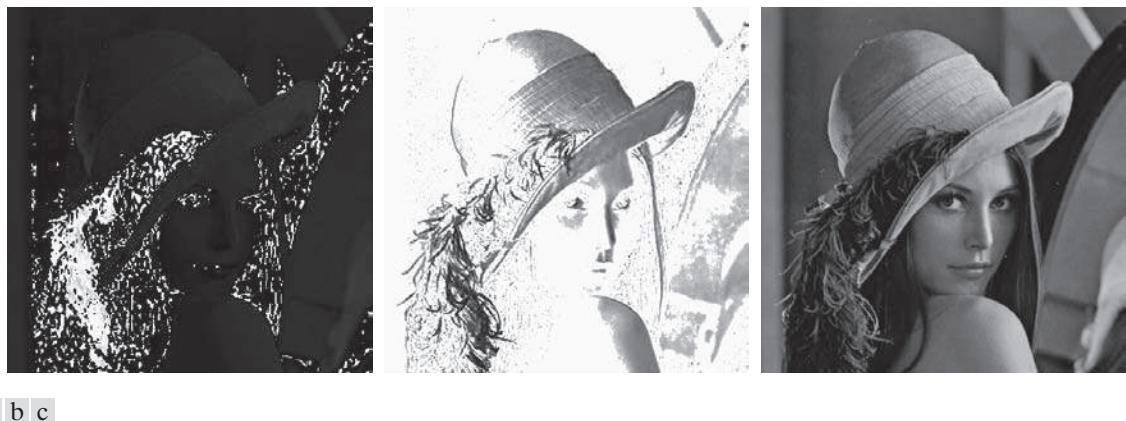
- (a) RGB image.
- (b) Red component image.
- (c) Green component.
- (d) Blue component.



EXAMPLE 6.12: Color image smoothing by neighborhood averaging.

Consider the RGB color image in Fig. 6.36(a). Its three component images are shown in Figs. 6.36(b) through (d). Figures 6.37(a) through (c) show the HSI components of the image. Based on the discussion in the previous paragraph, we smoothed each component image of the RGB image in Fig. 6.36 independently using a 5×5 averaging kernel. We then combined the individually smoothed images to form the smoothed, full-color RGB result in Fig. 6.38(a). Note that this image appears as we would expect from performing a spatial smoothing operation, as in the examples given in Section 3.5.

In Section 6.2, we mentioned that an important advantage of the HSI color model is that it decouples intensity and color information. This makes it suitable for many grayscale processing techniques and suggests that it might be more efficient to smooth only the intensity component of the HSI representation in Fig. 6.37. To illustrate the merits and/or consequences of this approach, we next smooth only the intensity component (leaving the hue and saturation components unmodified) and convert the processed result to an RGB image for display. The smoothed color image is shown in Fig. 6.38(b). Note



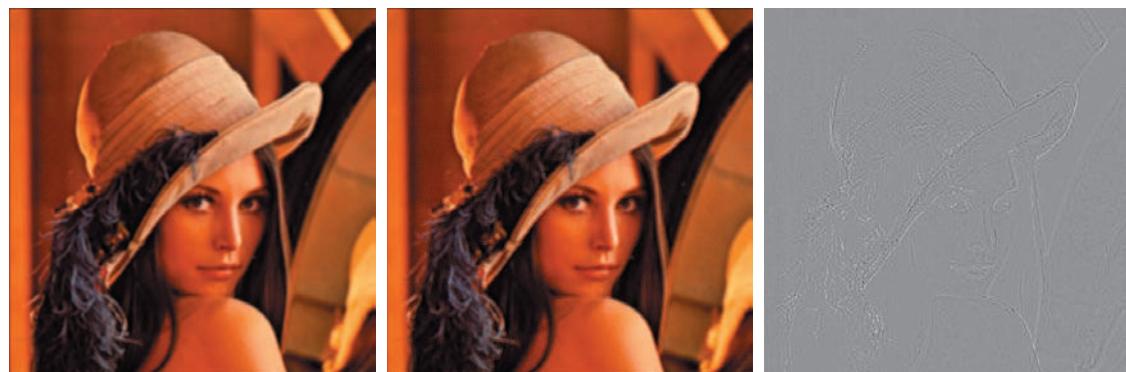
a b c

FIGURE 6.37 HSI components of the RGB color image in Fig. 6.36(a). (a) Hue. (b) Saturation. (c) Intensity.

that it is similar to Fig. 6.38(a), but, as you can see from the difference image in Fig. 6.38(c), the two smoothed images are not identical. This is because in Fig. 6.38(a) the color of each pixel is the average color of the pixels in the neighborhood. On the other hand, by smoothing only the intensity component image in Fig. 6.38(b), the hue and saturation of each pixel was not affected and, therefore, the pixel colors did not change. It follows from this observation that the difference between the two smoothing approaches would become more pronounced as a function of increasing kernel size.

COLOR IMAGE SHARPENING

In this section we consider image sharpening using the Laplacian (see Section 3.6). From vector analysis, we know that the Laplacian of a vector is defined as a vector



a b c

FIGURE 6.38 Image smoothing with a 5×5 averaging kernel. (a) Result of processing each RGB component image. (b) Result of processing the intensity component of the HSI image and converting to RGB. (c) Difference between the two results.

whose components are equal to the Laplacian of the individual scalar components of the input vector. In the RGB color system, the Laplacian of vector \mathbf{c} in Eq. (6-37) is

$$\nabla^2 [\mathbf{c}(x,y)] = \begin{bmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{bmatrix} \quad (6-47)$$

which, as in the previous section, tells us that we can compute the Laplacian of a full-color image by computing the Laplacian of each component image separately.

EXAMPLE 6.13: Image sharpening using the Laplacian.

Figure 6.39(a) was obtained using Eq. (3-54) and the kernel in Fig. 3.45(c) to compute the Laplacians of the RGB component images in Fig. 6.36. These results were combined to produce the sharpened full-color result. Figure 6.39(b) shows a similarly sharpened image based on the HSI components in Fig. 6.37. This result was generated by combining the Laplacian of the intensity component with the unchanged hue and saturation components. The difference between the RGB and HSI sharpened images is shown in Fig. 6.39(c). The reason for the discrepancies between the two images is as in Example 6.12.

6.7 USING COLOR IN IMAGE SEGMENTATION

Segmentation is a process that partitions an image into regions. Although segmentation is the topic of Chapters 10 and 11, we consider color segmentation briefly here for the sake of continuity. You will have no difficulty following the discussion.



a b c

FIGURE 6.39 Image sharpening using the Laplacian. (a) Result of processing each RGB channel. (b) Result of processing the HSI intensity component and converting to RGB. (c) Difference between the two results.

SEGMENTATION IN HSI COLOR SPACE

If we wish to segment an image based on color and, in addition, we want to carry out the process on individual planes, it is natural to think first of the HSI space because color is conveniently represented in the hue image. Typically, saturation is used as a masking image in order to isolate further regions of interest in the hue image. The intensity image is used less frequently for segmentation of color images because it carries no color information. The following example is typical of how segmentation is performed in the HSI color space.

EXAMPLE 6.14: Segmenting a color image in HSI color space.

Suppose that it is of interest to segment the reddish region in the lower left of the image in Fig. 6.40(a). Figures 6.40(b) through (d) are its HSI component images. Note by comparing Figs. 6.40(a) and (b) that the region in which we are interested has relatively high values of hue, indicating that the colors are on the blue-magenta side of red (see Fig. 6.11). Figure 6.40(e) shows a binary mask generated by thresholding the saturation image with a threshold equal to 10% of the maximum value in that image. Any pixel value greater than the threshold was set to 1 (white). All others were set to 0 (black).

Figure 6.40(f) is the product of the mask with the hue image, and Fig. 6.40(g) is the histogram of the product image (note that the grayscale is in the range [0, 1]). We see in the histogram that high values (which are the values of interest) are grouped at the very high end of the grayscale, near 1.0. The result of thresholding the product image with threshold value of 0.9 resulted in the binary image in Fig. 6.40(h). The spatial location of the white points in this image identifies the points in the original image that have the reddish hue of interest. This was far from a perfect segmentation because there are points in the original image that we certainly would say have a reddish hue, but that were not identified by this segmentation method. However, it can be determined by experimentation that the regions shown in white in Fig. 6.40(h) are about the best this method can do in identifying the reddish components of the original image. The segmentation method discussed in the following section is capable of yielding better results.

SEGMENTATION IN RGB SPACE

Although working in HSI space is more intuitive in the sense of colors being represented in a more familiar format, segmentation is one area in which better results generally are obtained by using RGB color vectors (see Fig. 6.7). The approach is straightforward. Suppose that the objective is to segment objects of a specified color range in an RGB image. Given a set of sample color points representative of the colors of interest, we obtain an estimate of the “average” color that we wish to segment. Let this average color be denoted by the RGB vector \mathbf{a} . The objective of segmentation is to classify each RGB pixel in a given image as having a color in the specified range or not. In order to perform this comparison, it is necessary to have a measure of similarity. One of the simplest measures is the Euclidean distance. Let \mathbf{z} denote an arbitrary point in RGB space. We say that \mathbf{z} is similar to \mathbf{a} if the distance between them is less than a specified threshold, D_0 . The Euclidean distance between \mathbf{z} and \mathbf{a} is given by

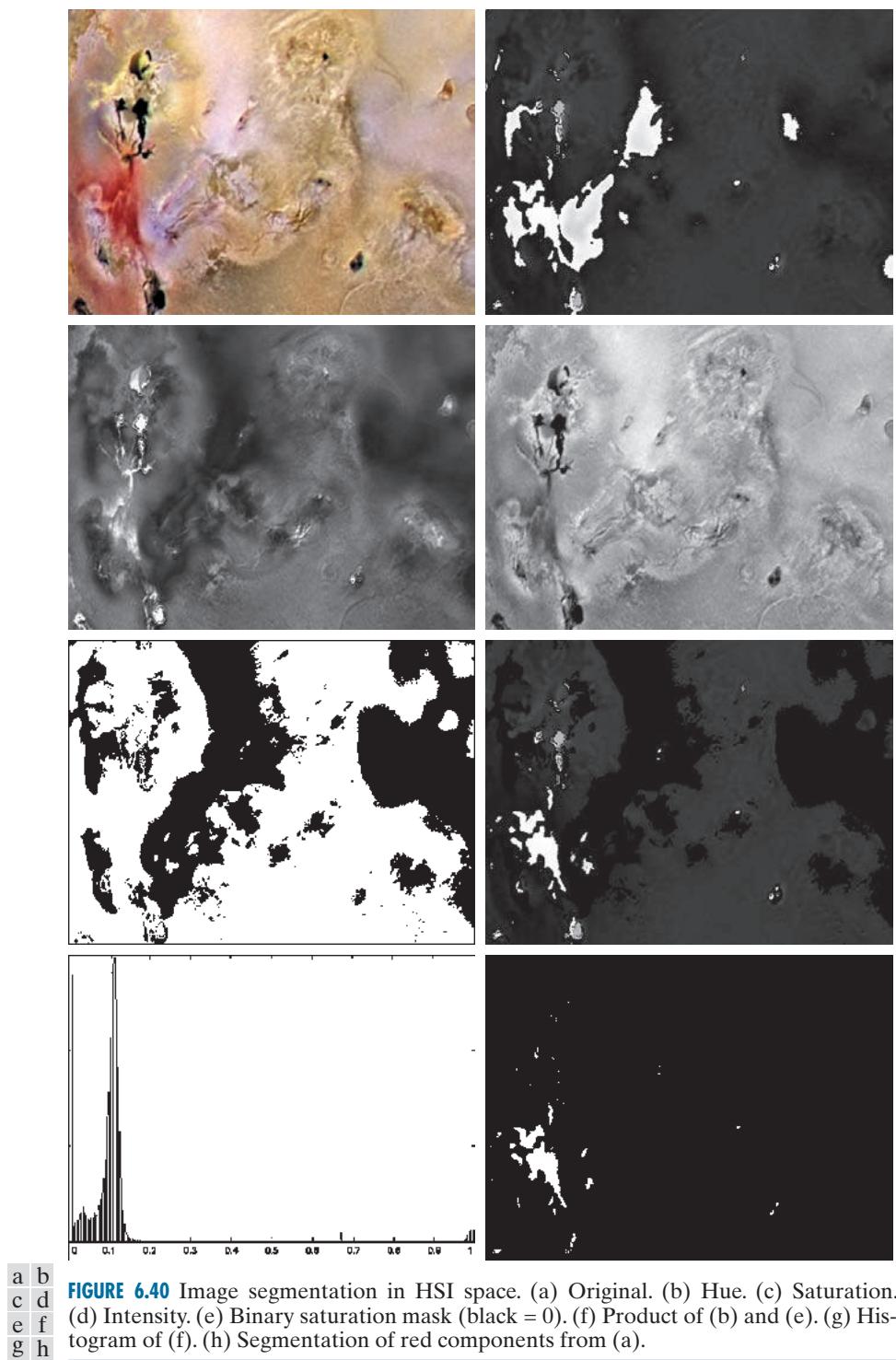
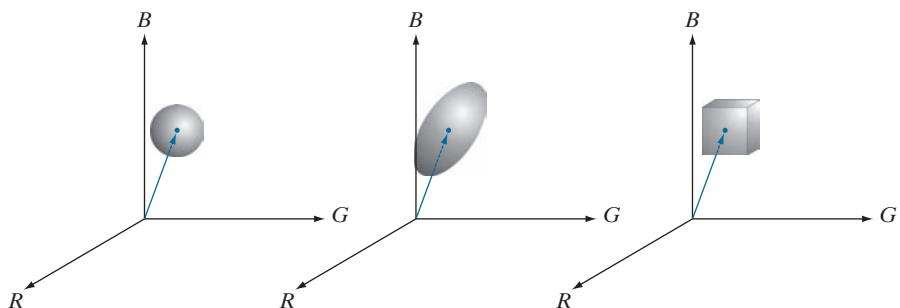


FIGURE 6.40 Image segmentation in HSI space. (a) Original. (b) Hue. (c) Saturation. (d) Intensity. (e) Binary saturation mask (black = 0). (f) Product of (b) and (e). (g) Histogram of (f). (h) Segmentation of red components from (a).

a	b	c
---	---	---

FIGURE 6.41

Three approaches for enclosing data regions for RGB vector segmentation.



$$\begin{aligned}
 D(\mathbf{z}, \mathbf{a}) &= \| \mathbf{z} - \mathbf{a} \| \\
 &= \left[(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a}) \right]^{\frac{1}{2}} \\
 &= \left[(\mathbf{z}_R - \mathbf{a}_R)^2 + (\mathbf{z}_G - \mathbf{a}_G)^2 + (\mathbf{z}_B - \mathbf{a}_B)^2 \right]^{\frac{1}{2}}
 \end{aligned} \tag{6-48}$$

where the subscripts R, G, and B denote the RGB components of vectors \mathbf{a} and \mathbf{z} . The locus of points such that $D(\mathbf{z}, \mathbf{a}) \leq D_0$ is a solid sphere of radius D_0 , as illustrated in Fig. 6.41(a). Points contained within the sphere satisfy the specified color criterion; points outside the sphere do not. Coding these two sets of points in the image with, say, black and white, produces a binary segmented image.

A useful generalization of Eq. (6-48) is a distance measure of the form

$$D(\mathbf{z}, \mathbf{a}) = \left[(\mathbf{z} - \mathbf{a})^T \mathbf{C}^{-1} (\mathbf{z} - \mathbf{a}) \right]^{\frac{1}{2}} \tag{6-49}$$

where \mathbf{C} is the covariance matrix (see Section 11.5) of the samples chosen to be representative of the color range we wish to segment. The locus of points such that $D(\mathbf{z}, \mathbf{a}) \leq D_0$ describes a solid 3-D elliptical body [Fig. 6.41(b)] with the important property that its principal axes are oriented in the direction of maximum data spread. When $\mathbf{C} = \mathbf{I}$, the 3×3 identity matrix, Eq. (6-49) reduces to Eq. (6-48). Segmentation is as described in the preceding paragraph.

Because distances are positive and monotonic, we can work with the distance squared instead, thus avoiding square root computations. However, implementing Eq. (6-48) or (6-49) is computationally expensive for images of practical size, even if the square roots are not computed. A compromise is to use a bounding box, as illustrated in Fig. 6.41(c). In this approach, the box is centered on \mathbf{a} , and its dimensions along each of the color axes is chosen proportional to the standard deviation of the samples along each of the axis. We use the sample data to compute the standard deviations, which are the parameters used for segmentation with this approach. Given an arbitrary color point, we segment it by determining whether or not it is on the surface or inside the box, as with the distance formulations. However, determining whether a color point is inside or outside a box is much simpler computationally

This equation is called the *Mahalanobis distance*. You are seeing it used here for *multivariate thresholding* (see Section 10.3 regarding thresholding).

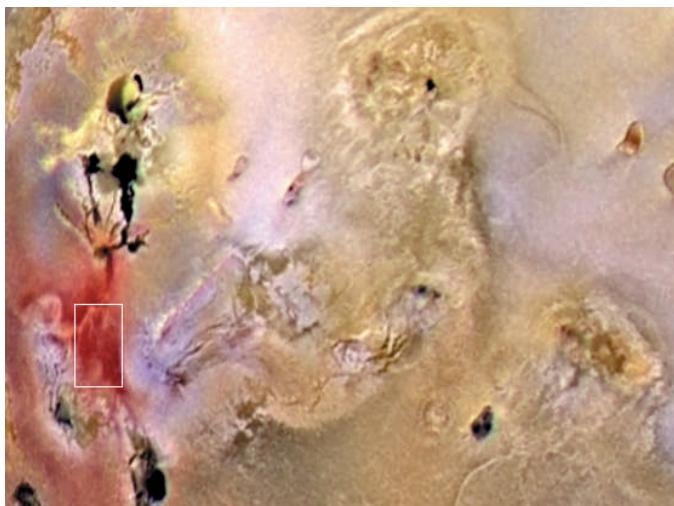
when compared to a spherical or elliptical enclosure. Note that the preceding discussion is a generalization of the color-slicing method introduced in Section 6.5.

EXAMPLE 6.15: Color segmentation in RGB color space.

The rectangular region shown Fig. 6.42(a) contains samples of reddish colors we wish to segment out of the color image. This is the same problem we considered in Example 6.14 using hue, but now we approach the problem using RGB color vectors. The approach followed was to compute the mean vector \mathbf{a} using the color points contained within the rectangle in Fig. 6.42(a), and then to compute the standard deviation of the red, green, and blue values of those samples. A box was centered at \mathbf{a} , and its dimensions along each of the RGB axes were selected as 1.25 times the standard deviation of the data along the corresponding axis. For example, let σ_R denote the standard deviation of the red components

a
b

FIGURE 6.42
Segmentation in
RGB space.
(a) Original image
with colors of
interest shown
enclosed by a
rectangle.
(b) Result of
segmentation
in RGB vector
space. Compare
with Fig. 6.40(h).



of the sample points. Then the dimensions of the box along the R-axis extended from $(a_R - 1.25\sigma_R)$ to $(a_R + 1.25\sigma_R)$, where a_R is the red component of average vector \mathbf{a} . Figure 6.42(b) shows the result of coding each point in the color image as white if it was on the surface or inside the box, and as black otherwise. Note how the segmented region was generalized from the color samples enclosed by the rectangle. In fact, by comparing Figs. 6.42(b) and 6.40(h), we see that segmentation in the RGB vector space yielded results that are much more accurate, in the sense that they correspond much more closely with what we would define as “reddish” points in the original color image. This result is not unexpected, because in the RGB space we used three color variables, as opposed to just one in the HSI space.

COLOR EDGE DETECTION

As we will discuss in Section 10.2, edge detection is an important tool for image segmentation. In this section, we are interested in the issue of computing edges on individual component images, as opposed to computing edges directly in color vector space.

We introduced edge detection by gradient operators in Section 3.6, when discussing image sharpening. Unfortunately, the gradient discussed there is not defined for vector quantities. Thus, we know immediately that computing the gradient on individual images and then using the results to form a color image will lead to erroneous results. A simple example will help illustrate the reason why.

Consider the two $M \times M$ color images (M odd) in Figs. 6.43(d) and (h), composed of the three component images in Figs. 6.43(a) through (c) and (e) through (g), respectively. If, for example, we compute the gradient image of each of the component images using Eq. (3-58), then add the results to form the two corresponding RGB gradient images, the value of the gradient at point $[(M+1)/2, (M+1)/2]$ would be the same in both cases. Intuitively, we would expect the gradient at that point to be stronger for the image in Fig. 6.43(d) because the edges of the R, G, and B images are in the same direction in that image, as opposed to the image in Fig. 6.43(h), in which only two of the edges are in the same direction. Thus we see from this simple example that processing the three individual planes to form a composite gradient image can yield erroneous results. If the problem is one of just detecting edges, then the individual-component approach can yield acceptable results. If accuracy is an issue, however, then obviously we need a new definition of the gradient applicable to vector quantities. We discuss next a method proposed by Di Zenzo [1986] for doing this.

The problem at hand is to define the gradient (magnitude and direction) of the vector \mathbf{c} in Eq. (6-37) at any point (x, y) . As we just mentioned, the gradient we studied in Section 3.6 is applicable to a *scalar* function $f(x, y)$; it is not applicable to vector functions. The following is one of the various ways in which we can extend the concept of a gradient to vector functions. Recall that for a scalar function $f(x, y)$, the gradient is a vector pointing in the direction of maximum rate of change of f at coordinates (x, y) .

Let \mathbf{r} , \mathbf{g} , and \mathbf{b} be unit vectors along the R, G, and B axis of RGB color space (see Fig. 6.7), and define the vectors



a	b	c	d
e	f	g	h

FIGURE 6.43 (a)–(c) R, G, and B component images, and (d) resulting RGB color image. (e)–(g) R, G, and B component images, and (h) resulting RGB color image.

$$\mathbf{u} = \frac{\partial R}{\partial x} \mathbf{r} + \frac{\partial G}{\partial x} \mathbf{g} + \frac{\partial B}{\partial x} \mathbf{b} \quad (6-50)$$

and

$$\mathbf{v} = \frac{\partial R}{\partial y} \mathbf{r} + \frac{\partial G}{\partial y} \mathbf{g} + \frac{\partial B}{\partial y} \mathbf{b} \quad (6-51)$$

Let the quantities g_{xx} , g_{yy} , and g_{xy} be defined in terms of the dot product of these vectors, as follows:

$$g_{xx} = \mathbf{u} \cdot \mathbf{u} = \mathbf{u}^T \mathbf{u} = \left| \frac{\partial R}{\partial x} \right|^2 + \left| \frac{\partial G}{\partial x} \right|^2 + \left| \frac{\partial B}{\partial x} \right|^2 \quad (6-52)$$

$$g_{yy} = \mathbf{v} \cdot \mathbf{v} = \mathbf{v}^T \mathbf{v} = \left| \frac{\partial R}{\partial y} \right|^2 + \left| \frac{\partial G}{\partial y} \right|^2 + \left| \frac{\partial B}{\partial y} \right|^2 \quad (6-53)$$

and

$$g_{xy} = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} = \frac{\partial R}{\partial x} \frac{\partial R}{\partial y} + \frac{\partial G}{\partial x} \frac{\partial G}{\partial y} + \frac{\partial B}{\partial x} \frac{\partial B}{\partial y} \quad (6-54)$$

Keep in mind that R, G, and B, and consequently the g 's, are functions of x and y . Using this notation, it can be shown (Di Zenzo [1986]) that the direction of maximum rate of change of $\mathbf{c}(x,y)$ is given by the angle

$$\theta(x,y) = \frac{1}{2} \tan^{-1} \left[\frac{2g_{xy}}{g_{xx} - g_{yy}} \right] \quad (6-55)$$

and that the value of the rate of change at (x,y) in the direction of $\theta(x,y)$ is given by

$$F_\theta(x,y) = \left\{ \frac{1}{2} \left[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy}) \cos 2\theta(x,y) + 2g_{xy} \sin 2\theta(x,y) \right] \right\}^{\frac{1}{2}} \quad (6-56)$$

Because $\tan(\alpha) = \tan(\alpha \pm \pi)$, if θ_0 is a solution to Eq. (6-55), so is $\theta_0 \pm \pi/2$. Furthermore, $F_\theta = F_{\theta+\pi}$, so F has to be computed only for values of θ in the half-open interval $[0,\pi)$. The fact that Eq. (6-55) gives two values 90° apart means that this equation associates with each point (x,y) a pair of orthogonal directions. Along one of those directions F is maximum, and it is minimum along the other. The derivation of these results is rather lengthy, and we would gain little in terms of the fundamental objective of our current discussion by detailing it here. Consult the paper by Di Zenzo [1986] for details. The Sobel operators discussed in Section 3.6 can be used to compute the partial derivatives required for implementing Eqs. (6-52) through (6-54).

EXAMPLE 6.16: Edge detection in RGB vector space.

Figure 6.44(b) is the gradient of the image in Fig. 6.44(a), obtained using the vector method just discussed. Figure 6.44(c) shows the image obtained by computing the gradient of each RGB component image and forming a composite gradient image by adding the corresponding values of the three component images at each coordinate (x,y) . The edge detail of the vector gradient image is more complete than the detail in the individual-plane gradient image in Fig. 6.44(c); for example, see the detail around the subject's right eye. The image in Fig. 6.44(d) shows the difference between the two gradient images at each point (x,y) . It is important to note that both approaches yielded reasonable results. Whether the extra detail in Fig. 6.44(b) is worth the added computational burden over the Sobel operator computations can only be determined by the requirements of a given problem. Figure 6.45 shows the three component gradient images, which, when added and scaled, were used to obtain Fig. 6.44(c).

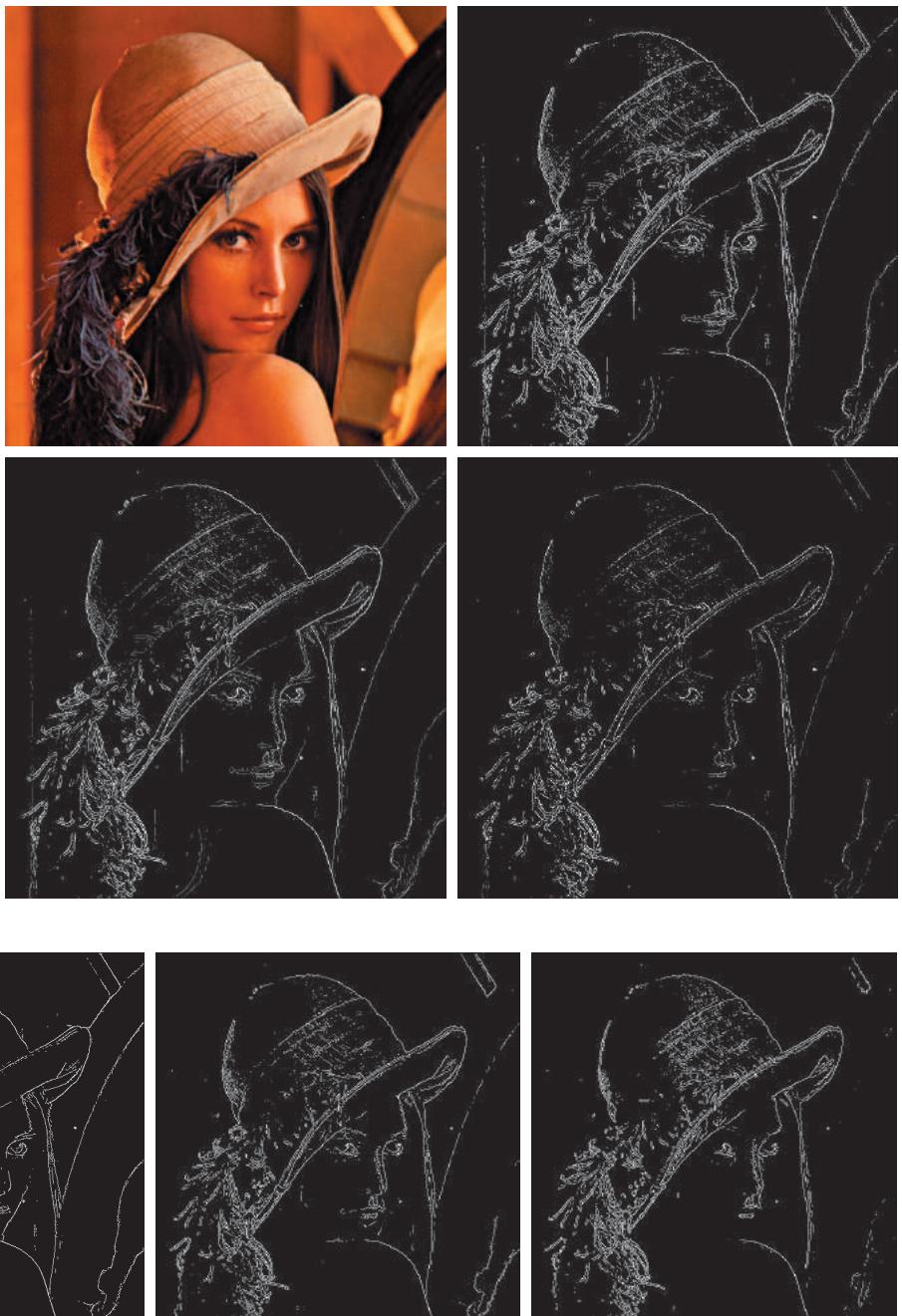
6.8 NOISE IN COLOR IMAGES

The noise models discussed in Section 5.2 are applicable to color images. Usually, the noise content of a color image has the same characteristics in each color channel, but it is possible for color channels to be affected differently by noise. One possibility is for the electronics of a particular channel to malfunction. However, different noise levels are more likely caused by differences in the relative strength of illumination available to each of the color channels. For example, use of a red filter in a CCD camera will reduce the strength of illumination detected by the red sensing elements. CCD sensors are noisier at lower levels of illumination, so the resulting red com-

a | b
c | d

FIGURE 6.44

- (a) RGB image.
- (b) Gradient computed in RGB color vector space.
- (c) Gradient image formed by the elementwise sum of three individual gradient images, each computed using the Sobel operators.
- (d) Difference between (b) and (c).



a | b | c

FIGURE 6.45 Component gradient images of the color image in Fig. 6.44. (a) Red component, (b) green component, and (c) blue component. These three images were added and scaled to produce the image in Fig. 6.44(c).

ponent of an RGB image would tend to be noisier than the other two component images in this situation.

EXAMPLE 6.17: Illustration of the effects of noise when converting noisy RGB images to HSI.

In this example, we take a brief look at noise in color images and how noise carries over when converting from one color model to another. Figures 6.46(a) through (c) show the three color planes of an RGB image corrupted by additive Gaussian noise, and Fig. 6.46(d) is the composite RGB image. Note that fine grain noise such as this tends to be less visually noticeable in a color image than it is in a grayscale image. Figures 6.47(a) through (c) show the result of converting the RGB image in Fig. 6.46(d) to HSI. Compare these results with the HSI components of the original image (see Fig. 6.37) and note how significantly degraded the hue and saturation components of the noisy image are. This was caused by the nonlinearity of the cos and min operations in Eqs. (6-17) and (6-18), respectively. On the other hand, the intensity component in Fig. 6.47(c) is slightly smoother than any of the three noisy RGB component images. This is because the intensity image is the average of the RGB images, as indicated in Eq. (6-19). (Recall the discussion in Section 2.6 regarding the fact that image averaging reduces random noise.)

a	b
c	d

FIGURE 6.46

(a)–(c) Red, green, and blue 8-bit component images corrupted by additive Gaussian noise of mean 0 and standard deviation of 28 intensity levels.
 (d) Resulting RGB image.
 [Compare (d) with Fig. 6.44(a).]





a b c

FIGURE 6.47 HSI components of the noisy color image in Fig. 6.46(d). (a) Hue. (b) Saturation. (c) Intensity.

In cases when, say, only one RGB channel is affected by noise, conversion to HSI spreads the noise to all HSI component images. Figure 6.48 shows an example. Figure 6.48(a) shows an RGB image whose green component image is corrupted by salt-and-pepper noise, with a probability of either salt or pepper equal to 0.05. The HSI component images in Figs. 6.48(b) through (d) show clearly how the noise spread from the green RGB channel to all the HSI images. Of course, this is not unexpected because computation of the HSI components makes use of all RGB components, as discussed in Section 6.2.

As is true of the processes we have discussed thus far, filtering of full-color images can be carried out on a per-image basis, or directly in color vector space, depending on the process. For example, noise reduction by using an averaging filter is the process discussed in Section 6.6, which we know gives the same result in vector space as it does if the component images are processed independently. However, other filters cannot be formulated in this manner. Examples include the class of order statistics filters discussed in Section 5.3. For instance, to implement a median filter in color vector space it is necessary to find a scheme for ordering vectors in a way that the median makes sense. While this was a simple process when dealing with scalars, the process is considerably more complex when dealing with vectors. A discussion of vector ordering is beyond the scope of our discussion here, but the book by Plataniotis and Venetsanopoulos [2000] is a good reference on vector ordering and some of the filters based on the concept of ordering.

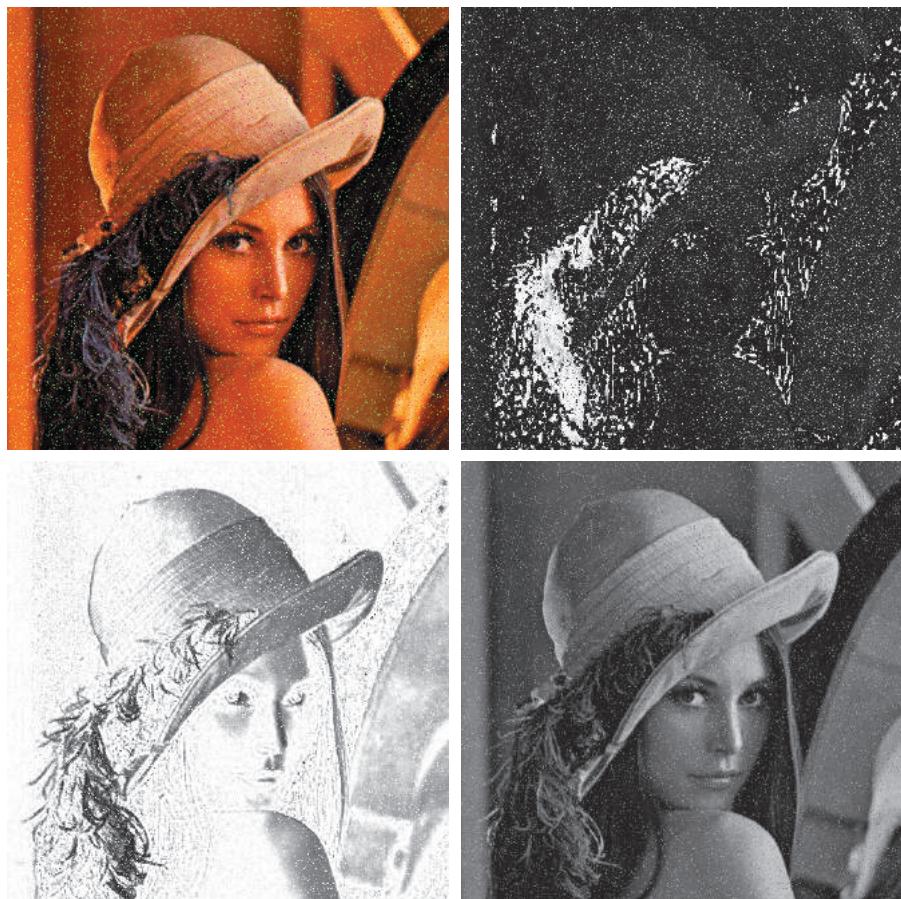
6.9 COLOR IMAGE COMPRESSION

Because the number of bits required to represent color is typically three to four times greater than the number employed in the representation of gray levels, data compression plays a central role in the storage and transmission of color images. With respect to the RGB, CMY(K), and HSI images of the previous sections, the data that are the object of any compression are the components of each color pixel (e.g., the red, green, and blue components of the pixels in an RGB image); they are

a	b
c	d

FIGURE 6.48

- (a) RGB image with green plane corrupted by salt-and-pepper noise.
- (b) Hue component of HSI image.
- (c) Saturation component.
- (d) Intensity component.



the means by which the color information is conveyed. Compression is the process of reducing or eliminating redundant and/or irrelevant data. Although compression is the topic of Chapter 8, we illustrate the concept briefly in the following example using a color image.

EXAMPLE 6.18: An example of color image compression.

Figure 6.49(a) shows a 24-bit RGB full-color image of an iris, in which 8 bits each are used to represent the red, green, and blue components. Figure 6.49(b) was reconstructed from a compressed version of the image in (a) and is, in fact, a compressed and subsequently decompressed approximation of it. Although the compressed image is not directly displayable—it must be decompressed before input to a color monitor—the compressed image contains only 1 data bit (and thus 1 storage bit) for every 230 bits of data in the original image (you will learn about the origin of these numbers in Chapter 8). Suppose that the image is of size $2000 \times 3000 = 6 \cdot 10^6$ pixels. The image is 24 bits/pixel, so its storage size is $144 \cdot 10^6$ bits.

a
b

FIGURE 6.49

Color image compression.
(a) Original RGB image.
(b) Result of compressing, then decompressing the image in (a).



Suppose that you are sitting at an airport waiting for your flight, and want to upload 100 such images using the airport's public WiFi connection. At a (relatively high) upload speed of $10 \cdot 10^6$ bits/sec, it would take you about 24 min to upload your images. In contrast, the compressed images would take about 6 sec to upload. Of course, the transmitted data would have to be decompressed at the other end for viewing, but the decompression can be done in a matter of seconds. Note that the reconstructed approximation image is slightly blurred. This is a characteristic of many lossy compression techniques; it can be reduced or eliminated by changing the level of compression. The JPEG 2000 compression algorithm used to generate Fig. 6.49(b) is described in detail in Section 8.2.

Summary, References, and Further Reading

The material in this chapter is an introduction to color image processing and covers topics selected to provide a solid background in the techniques used in this branch of image processing. Our treatment of color fundamentals and color models was prepared as foundation material for a field that is wide in technical scope and areas of application. In particular, we focused on color models that we felt are not only useful in digital image processing but provide also the tools necessary for further study in this area of image processing. The discussion of pseudocolor and full-color processing on an individual image basis provides a tie to techniques that were covered in some detail in Chapters 3 through 5. The material on color vector spaces is a departure from methods that we had studied before and highlights some important differences between grayscale and full-color processing. Our treatment of noise in color images also points out that the vector nature of the problem, along with the fact that color images are routinely transformed from one working space to another, has implications on the issue of how to reduce noise in these images. In some cases, noise filtering can be done on a per-image basis, but others, such as median filtering, require special treatment to reflect the fact that color pixels are vector quantities, as mentioned earlier. Although segmentation is the topic of Chapters 10 and 11, and image data compression is the topic of Chapter 8, we introduced them briefly in the context of color image processing.

For a comprehensive reference on the science of color, see Malacara [2011]. Regarding the physiology of color, see Snowden et al. [2012]. These two references, together with the book by Kuehni [2012], provide ample supplementary material for the discussion in Section 6.1. For further reading on color models (Section 6.2), see Fortner and Meyer [1997], Poynton [1996], and Fairchild [1998]. For a detailed derivation of the equations for the HSI model see the paper by Smith [1978] or consult the book website. The topic of pseudocolor (Section 6.3) is closely tied to the general area of image data visualization. Wolff and Yaeger [1993] is a good basic reference on the use of pseudocolor. See also Telea [2008]. For additional reading on the material in Sections 6.4 and 6.5, see Plataniotis and Venetsanopoulos [2000]. The material on color image filtering (Section 6.6) is based on the vector formulation introduced in Section 6.4 and on our discussion of spatial filtering in Chapter 3. The area of color image segmentation (Section 6.7) is of significant current interest. For an overview of current trends in this field see the survey by Vantaram and Saber [2012]. For more advanced color image processing techniques than those discussed in this chapter see Fernandez-Maloigne [2012]. The discussion in Section 6.8 is based on the noise models introduced in Section 5.2. References on color image compression (Section 6.9) are listed at the end of Chapter 8. For details of software implementation of many of the techniques discussed in this chapter, see Gonzalez, Woods, and Eddins [2009].

Problems

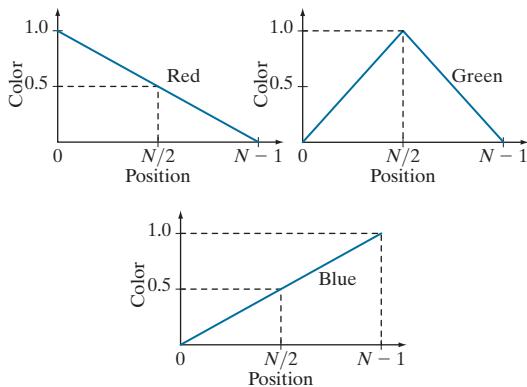
Solutions to the problems marked with an asterisk () are in the DIP4E Student Support Package (consult the book website: www.ImageProcessingPlace.com).*

- 6.1** Give the percentages of red (X), green (Y), and blue (Z) light required to generate the point labeled “warm white” in Fig. 6.5.
- 6.2*** Consider any two valid colors c_1 and c_2 with coordinates (x_1, y_1) and (x_2, y_2) in the chromaticity diagram of Fig. 6.5. Derive the necessary general expression(s) for computing the relative percentages of colors c_1 and c_2 composing any color that is known to lie on the straight line joining these two colors.
- 6.3** Consider any three valid colors 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. 6.5. Derive the necessary general expression(s) for computing the relative percentages of c_1 , c_2 , and c_3 composing a color that is known to lie within the triangle whose vertices are at the coordinates of c_1 , c_2 , and c_3 .

- 6.4*** In an automated assembly application, three types of parts are to be color-coded to simplify detection. However, only a monochrome TV camera is available to acquire digital images. Propose a technique for using this camera to detect the three different colors.

- 6.5** The R, G, and B component images of an RGB image have the horizontal intensity profiles shown in the following diagram. What color would a person see in the middle column of this image?



- 6.6*** Sketch the RGB components of the following image as they would appear on a monochrome monitor. All colors are at maximum intensity and saturation. In working this problem, consider the gray border as part of the image.



- 6.7** What is the maximum number of possible different shades of gray in an RGB image whose three component images are 8-bit images?

- 6.8** Consider the RGB cube in Fig. 6.8 and answer each of the following questions.

(a)* Describe how the gray levels vary in each of the R, G, and B primary images that make up the front face of the color cube (this is the face closer to you). Assume that each component image is an 8-bit image.

(b) Suppose that we replace every color in the

RGB cube by its CMY color. This new cube is displayed on an RGB monitor. Label with a color name the eight vertices of the new cube that you would see on the screen.

- (c) What can you say about the colors on the edges of the RGB color cube regarding saturation?

6.9

Do the following.

(a)* Sketch the CMY components of the image in Problem 6.6 as they would appear on a monochrome monitor.

(b) If the CMY components sketched in (a) are fed into the red, green, and blue inputs of a color monitor, respectively, describe the appearance of the resulting image.

6.10* Sketch the HSI components of the image in Problem 6.6 as they would appear on a monochrome monitor.

6.11 Propose a method for generating a color band similar to the one shown in the zoomed section entitled *Visible Spectrum* in Fig. 6.2. Note that the band starts at a dark purple on the left and proceeds toward pure red on the right. (*Hint:* Use the HSI color model.)

6.12* Propose a method for generating a color version of the image shown diagrammatically in Fig. 6.11(c). Give your answer in the form of a flow chart. Assume that the intensity value is fixed and given. (*Hint:* Use the HSI color model.)

6.13 Consider the following image composed of solid color squares. For discussing your answer, choose a gray scale consisting of eight shades of gray, 0 through 7, where 0 is black and 7 is white. Suppose that the image is converted to HSI color space. In answering the following questions, use specific numbers for the gray shades if using numbers makes sense. Otherwise, the relationships "same as," "lighter than," or "darker than" are sufficient. If you cannot assign a specific gray level or one of these relationships to the image you are discussing, give the reason.

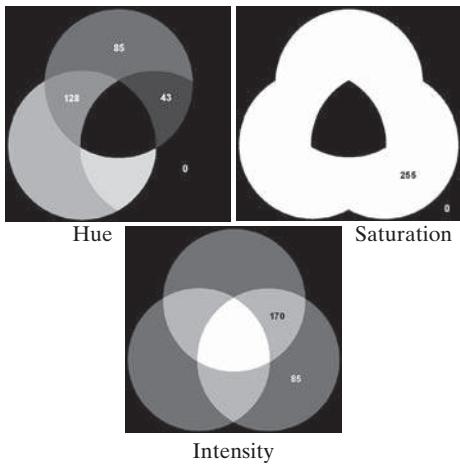
(a)* Sketch the hue image.

(b) Sketch the saturation image.

(c) Sketch the intensity image.



- 6.14** The following 8-bit images are the H, S, and I component images from Fig. 6.14. The numbers indicate gray-level values. Answer the following questions, explaining the basis for your answer in each. If it is not possible to answer a question based on the given information, state why you cannot do so.
- (a)* Give the gray-level values of all regions in the hue image.
 - (b) Give the gray-level value of all regions in the saturation image.
 - (c) Give the gray-level values of all regions in the intensity image.



- 6.15*** Compute the $L^* a^* b^*$ components of the image in Problem 6.6 assuming:

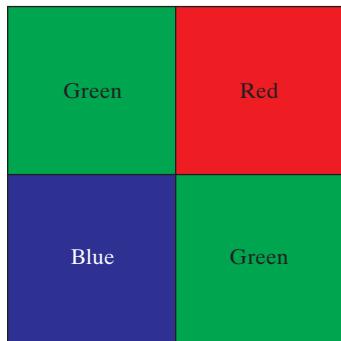
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.588 & 0.179 & 0.183 \\ 0.29 & 0.606 & 0.105 \\ 0 & 0.068 & 1.021 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This matrix equation defines the tristimulus values of the colors generated by the standard National Television System Committee (NTSC) color TV phosphors viewed under $D65$ standard illumination (Benson [1985]).

- 6.16*** Derive the CMY intensity mapping function of Eq. (6-41) from the RGB counterpart in Eq. (6-40). [Hint: Start with Eq. (6-5).]
- 6.17** Start with Eqs. (6-6)-(6-12) and derive Eq. (6-42). (Hint: The intensity of the CMYK image is changed by changing the K component only.)
- 6.18** Refer to Fig. 6.25 in answering the following:
- (a)* Why does the image in Fig. 6.25(e) exhibit predominantly red tones?
 - (b)* Suggest an automated procedure for coding the water in Fig. 6.25 in a bright-blue color.
 - (c) Suggest an automated procedure for coding the predominantly man-made components in a bright yellow color. [Hint: Work with Fig. 6.25(e).]
- 6.19*** Show that the saturation component of the complement of a color image cannot be computed from the saturation component of the input image alone.
- 6.20** Explain the shape of the hue transformation function for the image complement approximation in Fig. 6.31(b) using the HSI color model.
- 6.21*** Derive the CMY transformations to generate the complement of a color image.
- 6.22** Draw the general shape of the transformation functions used to correct excessive contrast in the RGB color space.
- 6.23*** Assume that the monitor and printer of an imaging system are imperfectly calibrated. An image that looks balanced on the monitor appears yellowish in print. Describe general transformations that might correct the imbalance. (Hints: Refer to the color wheel in Fig. 6.30 and the discussion of the $L^* a^* b^*$ color system in Section 6.2.)

6.24* Given an image in the RGB, CMY, or CMYK color system, how would you implement the color or equivalent of gray-scale histogram matching (specification) from Section 3.3?

6.25 Consider the following 500×500 RGB image, in which the squares are fully saturated red, green, and blue, and each of the colors is at maximum intensity. An HSI image is generated from this image. Answer the following questions.



(a) Describe the appearance of each HSI component image.

(b)* The saturation component of the HSI image is smoothed using an averaging kernel of size 125×125 . Describe the appearance of the result. (You may ignore image border effects in the filtering operation.)

(c) Repeat (b) for the hue image.

6.26 Answer the following.

(a)* Refer to the discussion in Section 6.7 about segmentation in the RGB color space. Give a procedure (in flow chart form) for deter-

mining whether a color vector (point) \mathbf{z} is inside a cube with sides W , centered at an average color vector \mathbf{a} . Distance computations are not allowed.

(b) If the box is aligned with the axes this process also can be implemented on an image-by-image basis. Show how you would do it.

6.27 Show that Eq. (6-49) reduces to Eq. (6-48) when $\mathbf{C} = \mathbf{I}$, the identity matrix.

6.28 Sketch the surface in RGB space for the points that satisfy the equation

$$D(\mathbf{z}, \mathbf{a}) = [(\mathbf{z} - \mathbf{a})^T \mathbf{C}^{-1} (\mathbf{z} - \mathbf{a})]^{\frac{1}{2}} = D_0$$

where D_0 is a positive constant. Assume that $\mathbf{a} = \mathbf{0}$, and that

$$\mathbf{C} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6.29 Refer to the discussion on color edge detection in Section 6.7. One might think that a logical approach for defining the gradient of an RGB image at any point (x, y) would be to compute the gradient vector (see Section 3.6) of each component image and then form a gradient vector for the color image by summing the three individual gradient vectors. Unfortunately, this method can at times yield erroneous results. Specifically, it is possible for a color image with clearly defined edges to have a zero gradient if this method were used. Give an example of such an image. (*Hint:* To simplify your analysis, set one of the color planes to a constant value.)

This page intentionally left blank

7

Wavelet and Other Image Transforms

Do not conform any longer to the pattern of this world, but be transformed by the renewing of your mind.

Romans 12:2

Preview

The discrete Fourier transform of Chapter 4 is a member of an important class of linear transforms that include the Hartley, sine, cosine, Walsh-Hadamard, Slant, Haar, and wavelet transforms. These transforms, which are the subject of this chapter, decompose functions into weighted sums of orthogonal or biorthogonal basis functions, and can be studied using the tools of linear algebra and functional analysis. When approached from this point of view, images are vectors in the vector space of all images. Basis functions determine the nature and usefulness of image transforms. Transforms are the coefficients of linear expansions. And for a given image and transform (or set of basis functions), both the orthogonality of the basis functions and the coefficients of the resulting transform are computed using inner products. All of an image's transforms are equivalent in the sense that they contain the same information and total energy. They are reversible and differ only in the way that the information and energy is distributed among the transform's coefficients.

Upon completion of this chapter, readers should:

- Understand image transforms in the context of series expansions.
- Be familiar with a variety of important image transforms and transform basis functions.
- Know the difference between orthogonal and biorthogonal basis functions.
- Be able to construct the transformation matrices of the discrete Fourier, Hartley, sine, cosine, Walsh-Hadamard, Slant, and Haar transforms.
- Be able to compute traditional image transforms, like the Fourier and Haar transforms, using elementary matrix operations.
- Understand the time-frequency plane and its relationship to wavelet transforms.
- Be able to compute 1-D and 2-D fast wavelet transforms (FWTs) using filter banks.
- Understand wavelet packet representations.
- Be familiar with the use of discrete orthogonal transforms in image processing.

7.1 PRELIMINARIES

Consult the Tutorials section of the book website for a brief tutorial on vectors and matrices.

In Chapter 2, the inner product of two column vectors, \mathbf{u} and \mathbf{v} , is denoted $\mathbf{u} \cdot \mathbf{v}$ [see Eq. (2-50)]. In this chapter, $\langle \mathbf{u}, \mathbf{v} \rangle$ is used to denote inner products within any inner product space satisfying conditions (a)–(d), including the Euclidean inner product space and real-valued column vectors of Chapter 2.

Euclidean space \mathbf{R}^N is an infinite set containing all real N -tuples.

A complex vector space with an inner product is called a *complex inner product space* or *unitary space*.

The notation $C[a, b]$ is also used in the literature.

Equations (7-4) through (7-15) are valid for all inner product spaces, including those defined by Eqs. (7-1) to (7-3).

In linear algebra and functional analysis, a *vector space* (or more formally an *abstract vector space*) is a set of mathematical objects or entities, called *vectors*, that can be added together and multiplied by *scalars*. An *inner product space* is an abstract vector space over a field of numbers, together with an *inner product function* that maps two vectors of the vector space to a scalar of the number field such that

- (a) $\langle u, v \rangle = \langle v, u \rangle^*$
- (b) $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$
- (c) $\langle \alpha u, v \rangle = \alpha \langle u, v \rangle$
- (d) $\langle v, v \rangle \geq 0$ and $\langle v, v \rangle = 0$ if and only if $v = 0$

where u , v , and w are vectors, α is a scalar, and $\langle \dots \rangle$ denotes the inner product operation. A simple example of a vector space is the set of directed line segments in two dimensions, where the line segments are represented mathematically as 2×1 column vectors, and the addition of vectors is the arithmetic equivalent of combining the line segments in a head to tail manner. An example of an inner product space is the set of real numbers \mathbf{R} combined with inner product function $\langle u, v \rangle = uv$, where the “vectors” are real numbers, the inner product function is multiplication, and axioms (a) through (d) above correspond to the commutative, distributive, associative, and “positivity of even powers” properties of multiplication, respectively.

Three inner product spaces are of particular interest in this chapter:

1. *Euclidean space* \mathbf{R}^N over real number field \mathbf{R} with *dot* or *scalar* inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} = u_0 v_0 + u_1 v_1 + \dots + u_{N-1} v_{N-1} = \sum_{i=0}^{N-1} u_i v_i \quad (7-1)$$

where \mathbf{u} and \mathbf{v} are $N \times 1$ column vectors.

2. *Unitary space* \mathbf{C}^N over complex number field \mathbf{C} with inner product function

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^{*T} \mathbf{v} = \sum_{i=0}^{N-1} u_i^* v_i = \langle \mathbf{v}, \mathbf{u} \rangle^* \quad (7-2)$$

where $*$ denotes the complex conjugate operation, and \mathbf{u} and \mathbf{v} are complex-valued $N \times 1$ column vectors.

3. Inner product space $C([a, b])$, where the vectors are continuous functions on the interval $a \leq x \leq b$ and the inner product function is the *integral inner product*

$$\langle f(x), g(x) \rangle = \int_a^b f^*(x) g(x) dx \quad (7-3)$$

In all three inner product spaces, the *norm* or *length* of vector z , denoted as $\|z\|$, is

$$\|z\| = \sqrt{\langle z, z \rangle} \quad (7-4)$$

While you must always take the context into account, we generally use the word “vector” for vectors in an abstract sense. A vector can be an $N \times 1$ matrix (i.e., column vector) or a continuous function.

Recall from linear algebra that a *basis* of a vector space is a set of linearly independent vectors for which any vector in the space can be written uniquely as a linear combination of basis vectors. The linear combinations are the *span* of the basis vectors. A set of vectors is *linearly independent* if no vector in the set can be written as a linear combination of the others.

While you must always take to the context into account, we often use the phrase “orthogonal basis” or “orthogonal transform” to refer to any basis or transform that is orthogonal, orthonormal, biorthogonal, or biorthonormal.

and the *angle* between two nonzero vectors z and w is

$$\theta = \cos^{-1} \frac{\langle z, w \rangle}{\|z\|\|w\|} \quad (7-5)$$

If the norm of z is 1, z is said to be *normalized*. If $\langle z, w \rangle = 0$ in Eq. (7-5), $\theta = 90^\circ$ and z and w are said to be *orthogonal*. A natural consequence of these definitions is that a set of nonzero vectors w_0, w_1, w_2, \dots is mutually or pairwise orthogonal if and only if

$$\langle w_k, w_l \rangle = 0 \quad \text{for } k \neq l \quad (7-6)$$

They are an *orthogonal basis* of the inner product space that they are said to *span*. If the *basis vectors* are normalized, they are an *orthonormal basis* and

$$\langle w_k, w_l \rangle = \delta_{kl} = \begin{cases} 0 & \text{for } k \neq l \\ 1 & \text{for } k = l \end{cases} \quad (7-7)$$

Similarly, a set of vectors w_0, w_1, w_2, \dots and a complementary set of *dual vectors* $\tilde{w}_0, \tilde{w}_1, \tilde{w}_2, \dots$ are said to be *biorthogonal* and a *biorthogonal basis* of the vector space that they span if

$$\langle \tilde{w}_k, w_l \rangle = 0 \quad \text{for } k \neq l \quad (7-8)$$

They are a *biorthonormal basis* if and only if

$$\langle \tilde{w}_k, w_l \rangle = \delta_{kl} = \begin{cases} 0 & \text{for } k \neq l \\ 1 & \text{for } k = l \end{cases} \quad (7-9)$$

As a mechanism for concisely describing an infinite set of vectors, the basis of an inner product space is one of the most useful concepts in linear algebra. The following derivation, which relies on the orthogonality of basis vectors, is foundational to the matrix-based transforms of the next section. Let $W = \{w_0, w_1, w_2, \dots\}$ be an orthogonal basis of inner product space V , and let $z \in V$. Vector z can then be expressed as the following linear combination of basis vectors

$$z = \alpha_0 w_0 + \alpha_1 w_1 + \alpha_2 w_2 + \dots \quad (7-10)$$

whose inner product with basis vector w_i is

$$\begin{aligned} \langle w_i, z \rangle &= \langle w_i, \alpha_0 w_0 + \alpha_1 w_1 + \alpha_2 w_2 + \dots \rangle \\ &= \alpha_0 \langle w_i, w_0 \rangle + \alpha_1 \langle w_i, w_1 \rangle + \dots + \alpha_i \langle w_i, w_i \rangle + \dots \end{aligned} \quad (7-11)$$

Since the w_i are mutually orthogonal, the inner products on the right side of Eq. (7-11) are 0 unless the subscripts of the vectors whose inner products are being

computed match [see Eq. (7-7)]. Thus, the only nonzero term is $\alpha_i \langle w_i, w_i \rangle$. Eliminating the zero terms and dividing both sides of the equation by $\langle w_i, w_i \rangle$ gives

$$\alpha_i = \frac{\langle w_i, z \rangle}{\langle w_i, w_i \rangle} \quad (7-12)$$

which reduces to

$$\alpha_i = \langle w_i, z \rangle \quad (7-13)$$

if the norms of the basis vectors are 1. A similar derivation, which is left as an exercise for the reader, yields

$$\alpha_i = \frac{\langle \tilde{w}_i, z \rangle}{\langle \tilde{w}_i, w_i \rangle} \quad (7-14)$$

and

$$\alpha_i = \langle \tilde{w}_i, z \rangle \quad (7-15)$$

for biorthogonal and biorthonormal basis vectors, respectively. Note when a basis and its dual are identical, biorthogonality reduces to orthogonality.

EXAMPLE 7.1: Vector norms and angles.

The norm of vector $f(x) = \cos x$ of inner product space $C([0, 2\pi])$ is

$$\|f(x)\| = \sqrt{\langle f(x), f(x) \rangle} = \left[\int_0^{2\pi} \cos^2 x \, dx \right]^{\frac{1}{2}} = \left[\frac{1}{2}x + \frac{1}{4}\sin(2x) \Big|_0^{2\pi} \right]^{\frac{1}{2}} = \sqrt{\pi}$$

The angle between vectors $\mathbf{z} = [1 \ 1]^T$ and $\mathbf{w} = [1 \ 0]^T$ of Euclidean inner product space \mathbf{R}^2 is

$$\theta = \cos^{-1} \left(\frac{\langle \mathbf{z}, \mathbf{w} \rangle}{\|\mathbf{z}\| \|\mathbf{w}\|} \right) = \cos^{-1} \left(\frac{1}{\sqrt{2}} \right) = 45^\circ$$

These results follow from Eqs. (7-1), (7-3), (7-4) and (7-5).

7.2 MATRIX-BASED TRANSFORMS

In mathematics, the word *transform* is used to denote a change in form without an accompanying change in value.

The 1-D discrete Fourier transform of Chapter 4 is one of a class of important transforms that can be expressed in terms of the general relation

$$T(u) = \sum_{x=0}^{N-1} f(x)r(x, u) \quad (7-16)$$

where x is a *spatial variable*, $T(u)$ is the transform of $f(x)$, $r(x,u)$ is a *forward transformation kernel*, and integer u is a *transform variable* with values in the range $0, 1, \dots, N - 1$. Similarly, the inverse transform of $T(u)$ is

$$f(x) = \sum_{u=0}^{N-1} T(u)s(x,u) \quad (7-17)$$

where $s(x,u)$ is an *inverse transformation kernel* and x takes on values in the range $0, 1, \dots, N - 1$. Transformation kernels $r(x,u)$ and $s(x,u)$ in Eqs. (7-16) and (7-17), which depend only on indices x and u and not on the values of $f(x)$ and $T(u)$, determine the nature and usefulness of the *transform pair* that they define.

Equation (7-17) is depicted graphically in Fig. 7.1. Note that $f(x)$ is a weighted sum of N inverse kernel functions (i.e., $s(x,u)$ for $u = 0, 1, \dots, N - 1$) and that $T(u)$ for $u = 0, 1, \dots, N - 1$ are the weights. All $N s(x,u)$ contribute to the value of $f(x)$ at every x . If we expand the right side of Eq. (7-17) to obtain

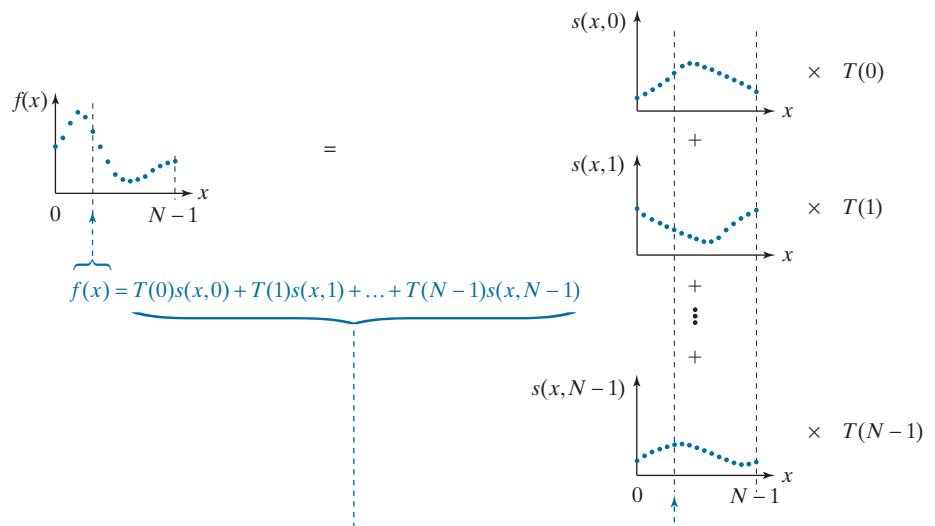
$$f(x) = T(0)s(x,0) + T(1)s(x,1) + \dots + T(N - 1)s(x,N - 1) \quad (7-18)$$

it is immediately apparent that the computation depicted in Fig. 7.1 is a *linear expansion* like that of Eq. (7-10)—with the $s(x,u)$ and $T(u)$ in Eq. (7-18) taking the place of the w_i (i.e., the basis vectors) and the α_i in Eq. (7-10). If we assume the $s(x,u)$ in Eq. (7-18) are orthonormal basis vectors of an inner product space, Eq. (7-13) tells us that

$$T(u) = \langle s(x,u), f(x) \rangle \quad (7-19)$$

and transform $T(u)$ for $u = 0, 1, \dots, N - 1$ can be computed via inner products.

FIGURE 7.1
A graphical illustration of Eq. (7-18).



We are now ready to express Eqs. (7-16) and (7-17) in matrix form. We begin by defining functions $f(x)$, $T(u)$, and $s(x,u)$ as column vectors

We will often use subscripts to denote the elements of a matrix or vector. Thus, f_0 denotes the first element of column vector \mathbf{f} , which is $f(0)$, and $s_{3,0}$ denotes the first element of column vector \mathbf{s}_3 , which is $s(0,3)$.

$$\mathbf{f} = \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix} \quad (7-20)$$

$$\mathbf{t} = \begin{bmatrix} T(0) \\ T(1) \\ \vdots \\ T(N-1) \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{N-1} \end{bmatrix} \quad (7-21)$$

and

$$\mathbf{s}_u = \begin{bmatrix} s(0,u) \\ s(1,u) \\ \vdots \\ s(N-1,u) \end{bmatrix} = \begin{bmatrix} s_{u,0} \\ s_{u,1} \\ \vdots \\ s_{u,N-1} \end{bmatrix} \text{ for } u = 0, 1, \dots, N-1 \quad (7-22)$$

and using them to rewrite Eq. (7-19) as

$$T(u) = \langle \mathbf{s}_u, \mathbf{f} \rangle \quad \text{for } u = 0, 1, \dots, N-1 \quad (7-23)$$

Combining the N basis vectors of the transform in an $N \times N$ transformation matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \vdots \\ \mathbf{s}_{N-1}^T \end{bmatrix} = [\mathbf{s}_0 \quad \mathbf{s}_1 \quad \dots \quad \mathbf{s}_{N-1}]^T \quad (7-24)$$

By employing Eq. (7-1), we assume the most common case of real-valued basis vectors. Equation (7-2) must be used for a complex inner product space.

we can then substitute Eq. (7-23) into Eq. (7-21) and use Eq. (7-1) to get

$$\begin{aligned} \mathbf{t} &= \begin{bmatrix} \langle \mathbf{s}_0, \mathbf{f} \rangle \\ \langle \mathbf{s}_1, \mathbf{f} \rangle \\ \vdots \\ \langle \mathbf{s}_{N-1}, \mathbf{f} \rangle \end{bmatrix} = \begin{bmatrix} s_{0,0}f_0 + s_{1,0}f_1 + \dots + s_{N-1,0}f_{N-1} \\ s_{0,1}f_0 + s_{1,1}f_1 + \dots + s_{N-1,1}f_{N-1} \\ \vdots \\ s_{0,N-1}f_0 + s_{1,N-1}f_1 + \dots + s_{N-1,N-1}f_{N-1} \end{bmatrix} \\ &= \begin{bmatrix} s_{0,0} & s_{1,0} & \dots & s_{N-1,0} \\ s_{0,1} & s_{1,1} & \ddots & s_{N-1,N-2} \\ \vdots & & & \\ s_{0,N-1} & s_{1,N-1} & \dots & s_{N-1,N-1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix} \end{aligned} \quad (7-25)$$

or

$$\mathbf{t} = \mathbf{Af} \quad (7-26)$$

The inverse of this equation follows from the observation that

$$\begin{aligned} \mathbf{AA}^T &= \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \vdots \\ \mathbf{s}_{N-1}^T \end{bmatrix} \begin{bmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \dots & \mathbf{s}_{N-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{s}_0^T \mathbf{s}_0 & \mathbf{s}_0^T \mathbf{s}_1 & \dots & \mathbf{s}_0^T \mathbf{s}_{N-1} \\ \mathbf{s}_1^T \mathbf{s}_0 & \mathbf{s}_1^T \mathbf{s}_1 & & \vdots \\ \vdots & \ddots & & \vdots \\ \mathbf{s}_{N-1}^T \mathbf{s}_0 & \dots & \mathbf{s}_{N-1}^T \mathbf{s}_{N-1} \end{bmatrix} \\ &= \begin{bmatrix} \langle \mathbf{s}_0, \mathbf{s}_0 \rangle & \langle \mathbf{s}_0, \mathbf{s}_1 \rangle & \dots & \langle \mathbf{s}_0, \mathbf{s}_{N-1} \rangle \\ \langle \mathbf{s}_1, \mathbf{s}_0 \rangle & \langle \mathbf{s}_1, \mathbf{s}_1 \rangle & & \vdots \\ \vdots & \ddots & & \vdots \\ \langle \mathbf{s}_{N-1}, \mathbf{s}_0 \rangle & \dots & \langle \mathbf{s}_{N-1}, \mathbf{s}_{N-1} \rangle \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & & \vdots \\ \vdots & \ddots & & \vdots \\ 0 & \dots & & 1 \end{bmatrix} = \mathbf{I} \end{aligned} \quad (7-27)$$

where the last two steps are a consequence of Eqs. (7-1) and (7-7), respectively. Since $\mathbf{AA}^T = \mathbf{I}$, premultiplying Eq. (7-26) by \mathbf{A}^T and simplifying gives $\mathbf{f} = \mathbf{A}^T \mathbf{t}$. Thus, Eqs. (7-16) and (7-17) become the matrix-based transform pair

$$\mathbf{t} = \mathbf{Af} \quad (7-28)$$

and

$$\mathbf{f} = \mathbf{A}^T \mathbf{t} \quad (7-29)$$

It is important to remember that, in the derivation of Eqs. (7-28) and (7-29), we assumed the N transform basis vectors (i.e., the \mathbf{s}_u for $u = 0, 1, \dots, N - 1$) of transformation matrix \mathbf{A} are real and orthonormal. In accordance with Eq. (7-7),

$$\langle \mathbf{s}_k, \mathbf{s}_l \rangle = \mathbf{s}_k^T \mathbf{s}_l = \delta_{kl} = \begin{cases} 0 & k \neq l \\ 1 & k = l \end{cases} \quad (7-30)$$

The assumed orthonormality allows forward transforms to be computed without explicit reference to a forward transformation kernel—that is, $\mathbf{t} = \mathbf{Af}$ where \mathbf{A} is a function of the inverse transformation kernel $s(x,u)$ alone. It is left as an exercise for the reader (see Problem 7.3) to show that for real orthonormal basis vectors, $r(x,u) = s(x,u)$.

Because the basis vectors of \mathbf{A} are real and orthonormal, the transform defined in Eq. (7-28) is called an *orthogonal transform*. It preserves inner products—i.e., $\langle \mathbf{f}_1, \mathbf{f}_2 \rangle = \langle \mathbf{t}_1, \mathbf{t}_2 \rangle = \langle \mathbf{Af}_1, \mathbf{Af}_2 \rangle$ —and thus the distances and angles between vectors before and after transformation. Both the rows and the columns of \mathbf{A} are orthonormal bases and $\mathbf{AA}^T = \mathbf{A}^T \mathbf{A} = \mathbf{I}$, so $\mathbf{A}^{-1} = \mathbf{A}^T$. The result is that Eqs. (7-28) and (7-29) are a *reversible transform pair*. Substituting Eq. (7-29) into (7-28) yields $\mathbf{t} = \mathbf{Af} = \mathbf{AA}^T \mathbf{t} = \mathbf{t}$, while substituting Eq. (7-28) into (7-29) gives $\mathbf{f} = \mathbf{A}^T \mathbf{t} = \mathbf{A}^T \mathbf{AF} = \mathbf{f}$.

For 2-D square arrays or images, Eqs. (7-16) and (7-17) become

Equations (7-31) and (7-32) are simplified versions of Eqs. (2-55) and (2-56) with $M = N$.

and

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) r(x,y,u,v) \quad (7-31)$$

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v) s(x,y,u,v) \quad (7-32)$$

where $r(x,y,u,v)$ and $s(x,y,u,v)$ are forward and inverse transformation kernels, respectively. Transform $T(u,v)$ and inverse transformation kernel $s(x,y,u,v)$ again can be viewed as weighting coefficients and basis vectors, respectively, with Eq. (7-32) defining a linear expansion of $f(x,y)$. As was noted in Chapter 2, forward transformation kernel $r(x,y,u,v)$ is *separable* if

$$r(x,y,u,v) = r_1(x,u)r_2(y,v) \quad (7-33)$$

and *symmetric* if r_1 is functionally equal to r_2 so

$$r(x,y,u,v) = r_1(x,u)r_1(y,v) \quad (7-34)$$

Substitute s for r in Eqs. (7-33) and (7-34) for separable and separable symmetric inverse kernels, respectively.

If the transformation kernels are real and orthonormal, and both r and s are separable and symmetric, the matrix equivalents of Eqs. (7-31) and (7-32) are

$$\mathbf{T} = \mathbf{AF}\mathbf{A}^T \quad (7-35)$$

and

$$\mathbf{F} = \mathbf{A}^T \mathbf{TA} \quad (7-36)$$

where \mathbf{F} is an $N \times N$ matrix containing the elements of $f(x,y)$, \mathbf{T} is its $N \times N$ transform, and \mathbf{A} is as previously defined in Eq. (7-24). The pre- and post-multiplications of \mathbf{F} by \mathbf{A} and \mathbf{A}^T in Eq. (7-35) compute the column and row transforms of \mathbf{F} , respectively. This, in effect, breaks the 2-D transform into two 1-D transforms, mirroring the process described in Section 4.11 for the 2-D DFT.

EXAMPLE 7.2: A simple orthogonal transformation.

Consider the 2-element basis vectors

$$\mathbf{s}_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{s}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

and note they are orthonormal in accordance with Eq. (7-30):

$$\begin{aligned}\langle \mathbf{s}_0, \mathbf{s}_1 \rangle &= \mathbf{s}_0^T \mathbf{s}_1 = \frac{1}{2} [1 \quad 1] \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{2} (1 - 1) = 0 \\ \langle \mathbf{s}_1, \mathbf{s}_0 \rangle &= \mathbf{s}_1^T \mathbf{s}_0 = \frac{1}{2} [1 \quad -1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{2} (1 - 1) = 0 \\ \langle \mathbf{s}_0, \mathbf{s}_0 \rangle &= \mathbf{s}_0^T \mathbf{s}_0 = \frac{1}{2} [1 \quad 1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{2} (1 + 1) = 1 \\ \langle \mathbf{s}_1, \mathbf{s}_1 \rangle &= \mathbf{s}_1^T \mathbf{s}_1 = \frac{1}{2} [1 \quad -1] \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{2} (1 + 1) = 1\end{aligned}$$

Substitution of \mathbf{s}_0 and \mathbf{s}_1 into Eq. (7-24) with $N = 2$ yields transformation matrix

$$\mathbf{A} = [\mathbf{s}_0 \quad \mathbf{s}_1]^T = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (7-37)$$

and the transform of 2×2 matrix

$$\mathbf{F} = \begin{bmatrix} 20 & 63 \\ 21 & 128 \end{bmatrix}$$

follows from Eq. (7-35):

$$\begin{aligned}\mathbf{T} &= \left(\frac{1}{\sqrt{2}} \right)^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 20 & 63 \\ 21 & 128 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^T \\ &= \frac{1}{2} \begin{bmatrix} 41 & 191 \\ -1 & -65 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 232 & -150 \\ -66 & 64 \end{bmatrix} \\ &= \begin{bmatrix} 116 & -75 \\ -33 & 32 \end{bmatrix}\end{aligned}$$

In accordance with Eq. (7-36), the inverse of transform \mathbf{T} is

$$\mathbf{F} = \left(\frac{1}{\sqrt{2}} \right)^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^T \begin{bmatrix} 116 & -75 \\ -33 & 32 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 83 & -43 \\ 149 & -107 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 20 & 63 \\ 21 & 128 \end{bmatrix}$$

Finally, we note \mathbf{A} is an orthogonal transformation matrix for which

$$\mathbf{AA}^T = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^T = \frac{1}{2} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}$$

and $\mathbf{A}^{-1} = \mathbf{A}^T$. It is also interesting to note Eq. (7-37) is the transformation matrix of the discrete *Fourier*, *Hartley*, *Cosine*, *Sin*, *Walsh-Hadamard*, *Slant*, and *Haar* transforms for 1- and 2-D inputs of size 2×1 and 2×2 , respectively. These transforms are discussed in detail in Sections 7.6 through 7.9.

Although formulated for real orthonormal bases and square arrays, Eqs. (7-35) and (7-36) can be modified to accommodate a variety of situations, including rectangular arrays, complex-valued basis vectors, and biorthonormal bases.

RECTANGULAR ARRAYS

When the arrays to be transformed are rectangular, as opposed to square, Eqs. (7-35) and (7-36) become

$$\mathbf{T} = \mathbf{A}_M \mathbf{F} \mathbf{A}_N^T \quad (7-38)$$

and

$$\mathbf{F} = \mathbf{A}_M^T \mathbf{T} \mathbf{A}_N \quad (7-39)$$

where \mathbf{F} , \mathbf{A}_M , and \mathbf{A}_N are of size $M \times N$, $M \times M$, and $N \times N$, respectively. Both \mathbf{A}_M and \mathbf{A}_N are defined in accordance with Eq. (7-24).

EXAMPLE 7.3: Computing the transform of a rectangular array.

A simple transformation in which M and N are 2 and 3, respectively, is

$$\begin{aligned} \mathbf{T} &= \mathbf{A}_2 \mathbf{F} \mathbf{A}_3^T = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 5 & 100 & 44 \\ 6 & 103 & 40 \end{bmatrix} \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0.366 & -1.366 \\ 1 & -1.366 & 0.366 \end{bmatrix}^T \\ &= \frac{1}{\sqrt{6}} \begin{bmatrix} 11 & 203 & 84 \\ -1 & -3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0.366 & -1.366 \\ 1 & -1.366 & 0.366 \end{bmatrix} = \begin{bmatrix} 121.6580 & -12.0201 & -96.1657 \\ 0 & -3.0873 & 1.8624 \end{bmatrix} \end{aligned}$$

where matrices \mathbf{F} , \mathbf{A}_2 , and \mathbf{A}_3 are as defined in the first step of the computation. As would be expected, 2×3 output transform \mathbf{T} is the same size as \mathbf{F} . It is left as an exercise for the reader (see Problem 7.5) to show that \mathbf{A}_3 is an orthogonal transformation matrix, and that the transformation is reversible using Eq. (7-39). The orthonormality of \mathbf{A}_2 was established in Example 7.2.

COMPLEX ORTHONORMAL BASIS VECTORS

Complex-valued basis vectors are orthonormal if and only if

$$\langle \mathbf{s}_k, \mathbf{s}_l \rangle = \langle \mathbf{s}_l, \mathbf{s}_k \rangle^* = \mathbf{s}_k^{*T} \mathbf{s}_l = \delta_{kl} = \begin{cases} 0 & k \neq l \\ 1 & k = l \end{cases} \quad (7-40)$$

where $*$ denotes the complex conjugate operation. When basis vectors are complex, as opposed to real-valued, Eqs. (7-35) and (7-36) become

$$\mathbf{T} = \mathbf{A}\mathbf{F}\mathbf{A}^T \quad (7-41)$$

and

$$\mathbf{F} = \mathbf{A}^{*T} \mathbf{T} \mathbf{A}^* \quad (7-42)$$

Orthogonal transforms are a special case of *unitary transforms* in which the expansion functions are real-valued. Both transforms preserve inner products.

respectively. Transformation matrix \mathbf{A} is then called a *unitary matrix* and Eqs. (7-41) and (7-42) are a *unitary transform* pair. An important and useful property of \mathbf{A} is that $\mathbf{A}^{*T} \mathbf{A} = \mathbf{A} \mathbf{A}^{*T} = \mathbf{A}^* \mathbf{A}^T = \mathbf{A}^T \mathbf{A}^* = \mathbf{I}$, so $\mathbf{A}^{-1} = \mathbf{A}^{*T}$. The 1-D counterparts of Eq. (7-41) and (7-42) are:

$$\mathbf{t} = \mathbf{A}\mathbf{f} \quad (7-43)$$

$$\mathbf{f} = \mathbf{A}^{*T} \mathbf{t} \quad (7-44)$$

EXAMPLE 7.4: A transform with complex-valued basis vectors.

Unlike orthogonal transformation matrices, where the inverse of the transformation matrix is its transpose, the inverse of unitary transformation matrix

$$\mathbf{A} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5 - j0.866 & -0.5 + j0.866 \\ 1 & -0.5 + j0.866 & -0.5 - j0.866 \end{bmatrix} \quad (7-45)$$

is its conjugate transpose. Thus,

$$\begin{aligned} \mathbf{A}^{*T} \mathbf{A} &= \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5 - j0.866 & -0.5 + j0.866 \\ 1 & -0.5 + j0.866 & -0.5 - j0.866 \end{bmatrix}^* \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5 - j0.866 & -0.5 + j0.866 \\ 1 & -0.5 + j0.866 & -0.5 - j0.866 \end{bmatrix} \\ &= \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5 + j0.866 & -0.5 - j0.866 \\ 1 & -0.5 - j0.866 & -0.5 + j0.866 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5 - j0.866 & -0.5 + j0.866 \\ 1 & -0.5 + j0.866 & -0.5 - j0.866 \end{bmatrix} \\ &= \frac{1}{3} \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} = \mathbf{I} \end{aligned}$$

where $j = \sqrt{-1}$ and matrix \mathbf{A} is a unitary matrix that can be used in Eqs. (7-41) through (7-44). It is easy

to show (see Problem 7.4) that when $\mathbf{A}^* \mathbf{A} = \mathbf{I}$, the basis vectors in \mathbf{A} satisfy Eq. (7-40) and are thus orthonormal.

BIORTHONORMAL BASIS VECTORS

Expansion functions $\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_{N-1}$ in Eq. (7-24) are *biorthonormal* if there exists a set of *dual expansion functions* $\tilde{\mathbf{s}}_0, \tilde{\mathbf{s}}_1, \dots, \tilde{\mathbf{s}}_{N-1}$ such that

$$\langle \tilde{\mathbf{s}}_k, \mathbf{s}_l \rangle = \delta_{kl} = \begin{cases} 0 & k \neq l \\ 1 & k = l \end{cases} \quad (7-46)$$

Neither the expansion functions nor their duals need be orthonormal themselves. Given a set of *biorthonormal expansion functions*, Eqs. (7-35) and (7-36) become

$$\mathbf{T} = \tilde{\mathbf{A}} \mathbf{F} \tilde{\mathbf{A}}^T \quad (7-47)$$

and

$$\mathbf{F} = \mathbf{A}^T \mathbf{T} \mathbf{A} \quad (7-48)$$

Transformation matrix \mathbf{A} remains as defined in Eq. (7-24); *dual transformation matrix* $\tilde{\mathbf{A}} = [\tilde{\mathbf{s}}_0 \ \tilde{\mathbf{s}}_1 \ \dots \ \tilde{\mathbf{s}}_{N-1}]^T$ is an $N \times N$ matrix whose rows are transposed dual expansion functions. When the expansion functions and their duals are identical—that is, when $\tilde{\mathbf{s}}_u = \mathbf{s}_u$ —Eqs. (7-47) and (7-48) reduce to Eqs. (7-35) and (7-36), respectively. The 1-D counterparts of Eqs. (7-47) and (7-48) are:

$$\mathbf{t} = \tilde{\mathbf{A}} \mathbf{f} \quad (7-49)$$

$$\mathbf{f} = \mathbf{A}^T \mathbf{t} \quad (7-50)$$

EXAMPLE 7.5: A biorthonormal transform.

Consider the real biorthonormal transformation matrices

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & 1 & 1 \\ -0.5303 & 0.5303 & -0.1768 & 0.1768 \\ -0.1768 & 0.1768 & -0.5303 & 0.5303 \end{bmatrix} \text{ and } \tilde{\mathbf{A}} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -0.25 & -0.25 & 0.25 & 0.25 \\ -1.0607 & 1.0607 & 0.3536 & -0.3536 \\ 0.3536 & -0.3536 & -1.0607 & 1.0607 \end{bmatrix}$$

It is left as an exercise for the reader (see Problem 7.16) to show that \mathbf{A} and $\tilde{\mathbf{A}}$ are biorthonormal. The transform of 1-D column vector $\mathbf{f} = [30 \ 11 \ 210 \ 6]^T$ is

$$\mathbf{t} = \tilde{\mathbf{A}} \mathbf{f} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -0.25 & -0.25 & 0.25 & 0.25 \\ -1.0607 & 1.0607 & 0.3536 & -0.3536 \\ 0.3536 & -0.3536 & -1.0607 & 1.0607 \end{bmatrix} \begin{bmatrix} 30 \\ 11 \\ 210 \\ 6 \end{bmatrix} = \begin{bmatrix} 128.5 \\ 43.75 \\ 51.9723 \\ -209.6572 \end{bmatrix}$$

Since

$$\langle \mathbf{f}, \mathbf{f} \rangle = \mathbf{f}^T \mathbf{f} = [30 \ 11 \ 210 \ 6] \begin{bmatrix} 30 \\ 11 \\ 210 \\ 6 \end{bmatrix} = 45,157$$

and $\langle \mathbf{t}, \mathbf{t} \rangle = \mathbf{t}^T \mathbf{t} = 65,084$, which is not equal to $\langle \mathbf{f}, \mathbf{f} \rangle$, the transformation does *not* preserve inner products. It is, however, reversible:

$$\mathbf{f} = \mathbf{A}^T \mathbf{t} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & 1 & 1 \\ -0.5303 & 0.5303 & -0.1768 & 0.1768 \\ -0.1768 & 0.1768 & -0.5303 & 0.5303 \end{bmatrix}^T \begin{bmatrix} 128.5 \\ 43.75 \\ 51.9723 \\ -209.6572 \end{bmatrix} = \begin{bmatrix} 30 \\ 11 \\ 210 \\ 6 \end{bmatrix}$$

Here, the forward and inverse transforms were computed using Eqs. (7-49) and (7-50), respectively.

Finally, we note the bulk of the concepts presented in this section can be generalized to continuous expansions of the form

$$f(x) = \sum_{u=-\infty}^{\infty} \alpha_u s_u(x) \quad (7-51)$$

where α_u and the $s_u(x)$ for $u = 0, \pm 1, \pm 2, \dots$ represent expansion coefficients and basis vectors of inner product space $C([a, b])$, respectively. For a given $f(x)$ and basis $s_u(x)$ for $u = 0, \pm 1, \pm 2, \dots$, the appropriate expansion coefficients can be computed from the definition of the integral inner product of $C([a, b])$ —i.e., Eq. (7-3)—and the general properties of all inner product spaces—i.e., Eqs. (7-10) through (7-15). Thus, for example, if $s_u(x)$ for $u = 0, \pm 1, \pm 2, \dots$ are orthonormal basis vectors of $C([a, b])$,

$$\alpha_u = \langle s_u(x), f(x) \rangle \quad (7-52)$$

Here, we have simply replaced i, z , and w_i in Eq. (7-13) with $u, f(x)$, and $s_u(x)$. In the next example, Eq. (7-52) will be used in the derivation of the continuous Fourier series.

EXAMPLE 7.6: The Fourier series and discrete Fourier transform.

Consider the representation of a continuous periodic function of period T as a linear expansion of orthonormal basis vectors of the form

$$s_u(x) = \frac{1}{\sqrt{T}} e^{j2\pi ux/T} \quad \text{for } u = 0, \pm 1, \pm 2, \dots \quad (7-53)$$

In accordance with Eqs. (7-51) and (7-52),

$$\begin{aligned} f(x) &= \sum_{u=-\infty}^{\infty} \alpha_u \left[\frac{1}{\sqrt{T}} e^{j2\pi ux/T} \right] \\ &= \frac{1}{\sqrt{T}} \sum_{u=-\infty}^{\infty} \alpha_u e^{j2\pi ux/T} \end{aligned} \quad (7-54)$$

and

$$\begin{aligned} \alpha_u &= \langle s_u(x), f(x) \rangle \\ &= \int_{-T/2}^{T/2} \left[\frac{1}{\sqrt{T}} e^{j2\pi ux/T} \right]^* f(x) dx \\ &= \frac{1}{\sqrt{T}} \int_{-T/2}^{T/2} f(x) e^{-j2\pi ux/T} dx \end{aligned} \quad (7-55)$$

With the exception of the variable names and normalization (i.e., the use of $1/\sqrt{T}$ in the above two equations as opposed to $1/T$ in only one of them), Eqs. (7-54) and (7-55) are the familiar Fourier series of Eqs. (4-8) and (4-9) in Chapter 4. An almost identical derivation, which is left as an exercise for the reader (see Problem 7.22), yields the following discrete counterparts of Eqs. (7-53) through (7-55):

$$s(x, u) = \frac{1}{\sqrt{N}} e^{j2\pi ux/N} \quad \text{for } u = 0, 1, \dots, N-1 \quad (7-56)$$

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} T(u) e^{j2\pi ux/N} \quad (7-57)$$

and

$$T(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) e^{-j2\pi ux/N} \quad (7-58)$$

The discrete complex basis vectors of Eq. (7-56) are an orthonormal basis of inner product space \mathbf{C}^N . Equations (7-58) and (7-57), except for the variable names and normalization, are the familiar discrete Fourier transform of Eqs. (4-44) and (4-45) in Chapter 4.

Now consider the use of Eqs. (7-55) and (7-58) in the computation of both the Fourier series and discrete Fourier transform of $f(x) = \sin(2\pi x)$ of period $T = 1$. In accordance with Eq. (7-55),

$$\begin{aligned} \alpha_1 &= \int_{-1/2}^{1/2} \left[\frac{1}{\sqrt{1}} e^{j2\pi(1)x/1} \right]^* \sin(2\pi x) dx \\ &= \int_{-1/2}^{1/2} e^{-j2\pi x} \sin(2\pi x) dx = \int_{-1/2}^{1/2} [\cos(2\pi x) - j \sin(2\pi x)] \sin(2\pi x) dx \\ &= \frac{1}{4\pi} \sin^2(2\pi x) - j \left[\frac{x}{2} - \frac{1}{8\pi} \sin(4\pi x) \right] \Big|_{-1/2}^{1/2} = -j0.5 \end{aligned}$$

and, in the same way, $\alpha_{-1} = j0.5$. Since all other coefficients are zero, the resulting Fourier series is

$$f(x) = j0.5e^{-j2\pi x} - j0.5e^{j2\pi x} \quad (7-59)$$

Equation (7-58) with $N = 8$ and $f(x) = \sin(2\pi x)$ for $x = 0, 1, \dots, 7$, on the other hand, yields

$$T(u) = \begin{cases} -j1.414 & u = 1 \\ +j1.414 & u = 7 \\ 0 & \text{otherwise} \end{cases} \quad (7-60)$$

Figure 7.2 depicts both computations as “matrix multiplications” in which continuous or discrete basis vectors (the rows of matrix \mathbf{A}) are multiplied by a continuous or discrete function (column vector \mathbf{f}) and integrated or summed to produce a set of discrete expansion or transform coefficients (column vector \mathbf{t}). For the Fourier series, the expansion coefficients are integral inner products of $\sin(2\pi x)$ and one of a potentially infinite set of continuous basis vectors. For the DFT, each transform coefficient is a discrete inner product of \mathbf{f} and one of eight discrete basis vectors using Eq. (7-2). Note since the DFT is based on complex orthonormal basis vectors, the transform can be computed as a matrix multiplication [in accordance with Eq. (7-43)]. Thus, the inner products that generate the elements of transform \mathbf{t} are embedded in matrix multiplication \mathbf{Af} . That is, each element of \mathbf{t} is formed by multiplying one row of \mathbf{A} —i.e., one discrete expansion function—element by element by \mathbf{f} and summing the resulting products.

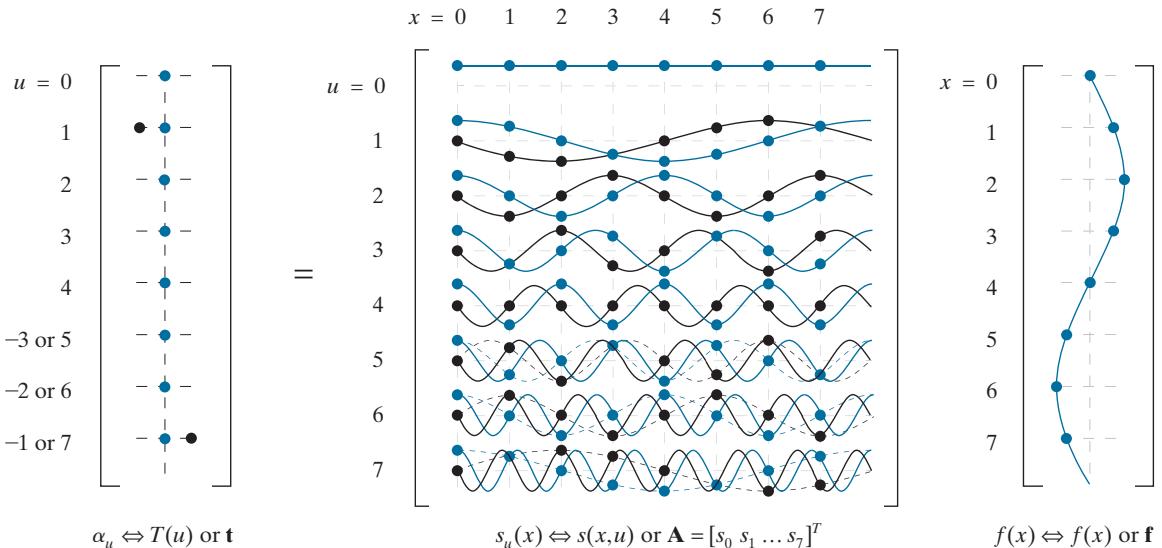


FIGURE 7.2 Depicting the continuous Fourier series and 8-point DFT of $f(x) = \sin(2\pi x)$ as “matrix multiplications.” The real and imaginary parts of all complex quantities are shown in blue and black, respectively. Continuous and discrete functions are represented using lines and dots, respectively. Dashed lines are included to show that $s_5 = s_3^*$, $s_6 = s_2^*$, and $s_7 = s_1^*$, effectively cutting the maximum frequency of the DFT in half. The negative indices to the left of \mathbf{t} are for the Fourier series computation alone.

7.3 CORRELATION

Example 7.6 highlights the role of inner products in the computation of orthogonal transform coefficients. In this section, we turn our attention to the relationship between those coefficients and correlation.

To be precise, we should use the term *cross-correlation* when $f(x) \neq g(x)$ and *auto-correlation* when $f(x) = g(x)$. Equation (7-61) is valid for both cases.

As the name *sliding inner product* suggests, visualize sliding one function over another, multiplying them together, and computing the area. As the area increases, the functions become increasingly similar.

Given two continuous functions $f(x)$ and $g(x)$, the *correlation* of f and g , denoted $f \star g(\Delta x)$, is defined as

$$\begin{aligned} f \star g(\Delta x) &= \int_{-\infty}^{\infty} f^*(x)g(x + \Delta x)dx \\ &= \langle f(x), g(x + \Delta x) \rangle \end{aligned} \quad (7-61)$$

where the final step follows from Eq. (7-3) with $a = -\infty$ and $b = \infty$. Sometimes called the *sliding inner product* of f and g , correlation measures the similarity of $f(x)$ and $g(x)$ as a function of their relative displacement Δx . If $\Delta x = 0$,

$$f \star g(0) = \langle f(x), g(x) \rangle \quad (7-62)$$

and Eq. (7-52), which defines the coefficients of the continuous orthonormal expansion in Eq. (7-51), can be alternately written as

$$\alpha_u = \langle f, s_u \rangle = f \star s_u(0) \quad (7-63)$$

Thus, the expansion coefficients are *single-point correlations* in which the displacement Δx is zero. Each α_u measures the similarity of $f(x)$ and one $s_u(x)$.

The discrete equivalents of Eqs. (7-61) through (7-63) are

$$\mathbf{f} \star \mathbf{g}(m) = \sum_{x=-\infty}^{\infty} f_n^* g_{n+m} \quad (7-64)$$

$$\mathbf{f} \star \mathbf{g}(0) = \langle \mathbf{f}, \mathbf{g} \rangle \quad (7-65)$$

and

$$T(u) = \langle \mathbf{s}_u, \mathbf{f} \rangle = \mathbf{s}_u \star \mathbf{f}(0) \quad (7-66)$$

The equation for 2-D discrete correlation is given in Table 4.3. In Eq. (7-64), n and m are integers, f_n denotes the n th element of \mathbf{f} , and g_{n+m} denotes the $(n+m)$ th element of \mathbf{g} . Equation (7-66) follows from Eqs. (7-65) and (7-23).

respectively. Comments similar to those made in regard to Eq. (7-63) and continuous series expansions also can be made with respect to Eq. (7-66) and discrete orthogonal transforms. Each element of an orthogonal transform [i.e., transform coefficient $T(u)$ of Eq. (7-23)] is a single-point correlation that measures the similarity of \mathbf{f} and vector \mathbf{s}_u . This powerful property of orthogonal transforms is the basis upon which the sinusoidal interference in Fig. 2.45(a) of Example 2.11 in Chapter 2 and Fig. 4.65(a) of Example 4.25 in Chapter 4 was identified and eliminated.

EXAMPLE 7.7: Correlation in the DFT of Example 7.6.

Consider again the 8-point DFT in Example 7.6 and note, in accordance with Eq. (7-56), the basis vectors are complex exponentials of the following harmonically related angular frequencies: $0, 2\pi, 4\pi, 6\pi, 8\pi, 6\pi, 4\pi$, and 2π (aliasing reduces the last three frequencies from $10\pi, 12\pi$, and 14π , respectively). Since discrete input $f(x) = \sin(2\pi x)$ is a single frequency sinusoid of angular frequency 2π , \mathbf{f} should be highly correlated with basis vectors \mathbf{s}_1 and \mathbf{s}_7 . As can be seen in Fig. 7.2, transform \mathbf{t} does indeed reach its maximum at $u = 1$ and 7 ; it is nonzero at these two frequencies alone.

7.4 BASIS FUNCTIONS IN THE TIME-FREQUENCY PLANE

Because transforms measure the degree to which a function resembles a selected set of basis vectors, we now turn our attention to the basis vectors themselves. In the following discussions, the terms basis vector and basis function are synonymous.

As can be seen in Fig. 7.3, where the basis vectors of some commonly encountered transforms are depicted, most orthogonal bases are mathematically related sets of sinusoids, square waves, ramps, and other small waves called *wavelets*. If $h(t)$ is a basis vector and $g(t)$ is the function being transformed, transform coefficient $g \star h(0)$, as noted in the previous section, is a measure of the similarity of g and h . Large values of $g \star h(0)$ indicate that g and h share important characteristics in time and frequency (e.g., shape and bandwidth). Thus, if h is the ramp-shaped basis function at $u = 1$ in Fig. 7.3(d), transform coefficient $g \star h(0)$ can be used to detect linear brightness gradients across a row of an image. If h is a sinusoidal basis function like those of Fig. 7.3(a), on the other hand, $g \star h(0)$ can be used to spot sinusoidal interference patterns. Plots like those of Fig. 7.3, together with a similarity measure like $g \star h(0)$, can reveal a great deal about the time and frequency characteristics of the function being transformed.

A purely objective descriptor of h , and thus of g for large values of $g \star h(0)$, is the location of h on the *time-frequency plane* of Fig. 7.4(a). Let $p_h(t) = |h(t)|^2 / \|h(t)\|^2$ be a probability density function with mean

$$\mu_t = \frac{1}{\|h(t)\|^2} \int_{-\infty}^{\infty} t |h(t)|^2 dt \quad (7-67)$$

and variance

$$\sigma_t^2 = \frac{1}{\|h(t)\|^2} \int_{-\infty}^{\infty} (t - \mu_t)^2 |h(t)|^2 dt \quad (7-68)$$

and let $p_H(f) = |H(f)|^2 / \|H(f)\|^2$ be a probability density function with mean

$$\mu_f = \frac{1}{\|H(f)\|^2} \int_{-\infty}^{\infty} f |H(f)|^2 df \quad (7-69)$$

and variance

$$\sigma_f^2 = \frac{1}{\|H(f)\|^2} \int_{-\infty}^{\infty} (f - \mu_f)^2 |H(f)|^2 df \quad (7-70)$$

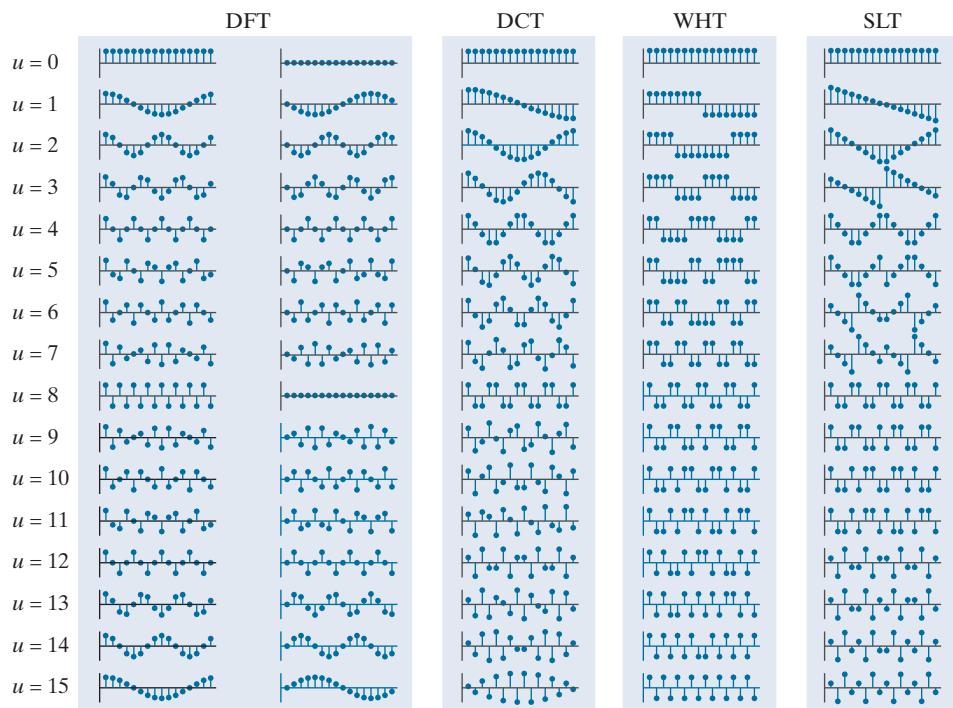
In our introduction to the *time-frequency plane*, independent variables t and f , rather than spatial variables x and u , are employed. Continuous functions $g(t)$ and $h(f)$ take the place of $f(x)$ and $s_u(x)$ in the previous sections. Though the concepts are presented using continuous functions and variables, they are equally applicable to discrete functions and variables.

In Eq. (7-67), each value of t is weighted by $p_h(t)$ to compute a weighted mean with respect to coordinate t .

a	b	c	d
e	f	g	h

FIGURE 7.3

Basis vectors (for $N = 16$) of some commonly encountered transforms:
 (a) Fourier basis (real and imaginary parts),
 (b) discrete Cosine basis,
 (c) Walsh-Hadamard basis,
 (d) Slant basis,
 (e) Haar basis,
 (f) Daubechies basis,
 (g) Biorthogonal B-spline basis and its dual, and
 (h) the standard basis, which is included for reference only (i.e., not used as the basis of a transform).



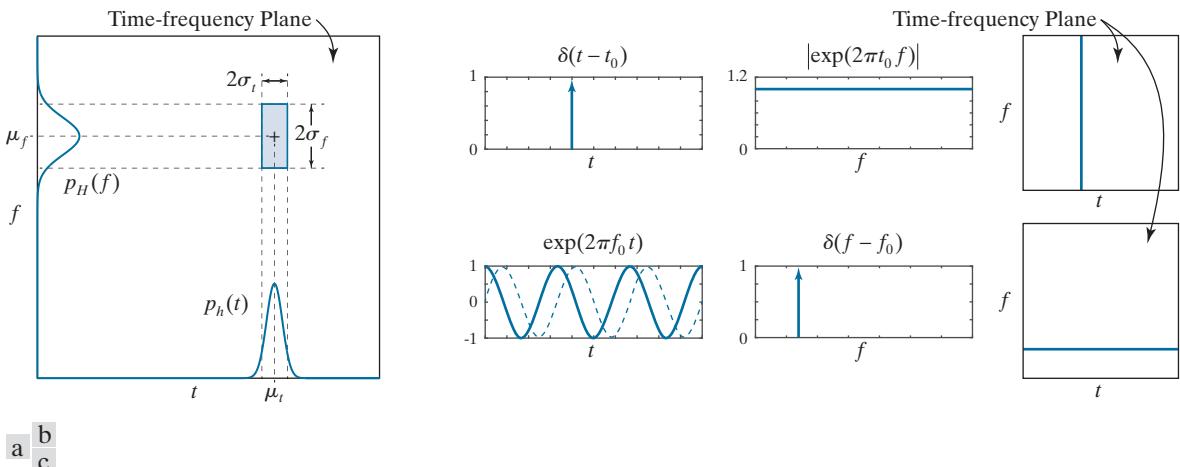


FIGURE 7.4 (a) Basis function localization in the time-frequency plane. (b) A standard basis function, its spectrum, and location in the time-frequency plane. (c) A complex sinusoidal basis function (with its real and imaginary parts shown as solid and dashed lines, respectively), its spectrum, and location in the time-frequency plane.

The constant on the right side of Eq. (7-71) is $\frac{1}{4}$ if stated in terms of angular frequency ω . Equality is possible, but only with a Gaussian basis function, whose transform is also a Gaussian function.

where f denotes frequency and $H(f)$ is the Fourier transform of $h(t)$. Then the energy[†] of basis function h , as illustrated in Fig. 7.4(a), is concentrated at (μ_t, μ_f) on the time-frequency plane. The majority of the energy falls in a rectangular region, called a *Heisenberg box* or *cell*, of area $4\sigma_t\sigma_f$ such that

$$\sigma_t^2\sigma_f^2 \geq \frac{1}{16\pi^2} \quad (7-71)$$

Since the *support* of a function can be defined as the set of points where the function is nonzero, Heisenberg's *uncertainty principle* tells us that it is impossible for a function to have finite support in both time and frequency. Equation (7-71), called the *Heisenberg-Gabor inequality*, places a lower bound on the area of the Heisenberg cell in Fig. 7.4(a), revealing that σ_t and σ_f cannot both be arbitrarily small. Thus, while basis function $\delta(t - t_0)$ in Fig. 7.4(b) is perfectly *localized* in time [that is, $\sigma_t = 0$ since the width of $\delta(t - t_0)$ is zero], its spectrum is nonzero on the entire f -axis. That is, since $\Im\{\delta(t - t_0)\} = \exp(-j2\pi f_0 t_0)$ and $|\exp(-j2\pi f_0 t_0)| = 1$ for all f , $\sigma_f = \infty$. The result is an infinitesimally narrow, infinitely high Heisenberg cell on the time-frequency plane. Basis function $\exp(2\pi f_0 t)$ of Fig. 7.4(c), on the other hand, is essentially nonzero on the entire time axis, but is perfectly localized in frequency. Because $\Im\{\exp(2\pi f_0 t)\} = \delta(f - f_0)$, spectrum $|\delta(f - f_0)|$ is zero at all frequencies other than $f = f_0$. The resulting Heisenberg cell is infinitely wide ($\sigma_t = \infty$) and infinitesimally small in height ($\sigma_f = 0$). As Figs. 7.4(b) and (c) illustrate, perfect localization in time is accompanied by a loss of localization in frequency and vice versa.

Returning again to Fig. 7.3, note the DFT basis in Fig. 7.3(a) and the standard basis in Fig. 7.3(h) are discrete examples (for $N = 16$) of the impulse and complex

[†]The energy of continuous function $h(t)$ is $\int_{-\infty}^{\infty} |h(t)|^2 dt$.

The DFT basis functions do not appear to be frequency ordered because of aliasing. See Example 7.6.

exponential functions in Figs. 7.4(c) and (b), respectively. Every other basis in the top half of Fig. 7.3 is both frequency ordered on index u and of width or support 16. For a given u , their locations in the time-frequency plane are similar. This is particularly evident when u is 8 and the basis functions are identical—as are their Heisenberg cells. For all other u , Heisenberg cell parameters μ_t , σ_t , μ_f , and σ_f are close in value, with small differences accounting for the distinctive shapes of the cosine, ramp, and square wave. In a similar manner, the basis functions in the bottom half of Fig. 7.3, with the exception of the standard basis already discussed, are also similar for a given u . These basis functions are scaled and shifted small waves, called *wavelets*, of the form

$$\psi_{s,\tau}(t) = 2^{s/2} \psi(2^s t - \tau) \quad (7-72)$$

where s and τ are integers and *mother wavelet* $\psi(t)$ is a real, square-integrable function with a *bandpass-like spectrum*. Parameter τ determines the position of $\psi_{s,\tau}(t)$ on the t -axis, s determines its width—that is, how broad or narrow it is along the t -axis, and $2^{s/2}$ controls its amplitude.

In conjunction with a properly designed mother wavelet, Eq. (7-72) generates a basis that is characterized by the Heisenberg cells on the right side of Fig. 7.5. Letting $\Psi(f)$ be the Fourier transform of $\psi(t)$, the transform of time-scaled wavelet $\psi(2^s t)$ is

$$\Im\{\psi(2^s t)\} = \frac{1}{|2^s|} \Psi\left(\frac{f}{2^s}\right) \quad (7-73)$$

and for positive values of s , the spectrum is stretched—shifting each frequency component higher by a factor of 2^s . As was the case for the rectangular pulse in Example 4.1, compressing time expands the spectrum. This is illustrated graphically in Figs. 7.5(b)–(d). Note the width of the basis function in Fig. 7.5(c) is half of that in (d), while the width of its spectrum is double that of (d). It is shifted higher in frequency by a factor of two. The same can be said for the basis function and spectrum in Fig. 7.5(b) when compared to (c). This halving of support in time and doubling of support in frequency produces Heisenberg cells of differing widths and heights, but of equal area. Moreover, each row of cells on the right of Fig. 7.5 represents a unique scale s and range of frequencies. The cells within a row are shifted with respect to one another in time. In accordance with Eq. (4-71) and Table 4.4 of Chapter 4, if $\psi(t)$ is shifted in time by τ ,

$$\Im\{\psi(t - \tau)\} = e^{-j2\pi\tau f} \Psi(f) \quad (7-74)$$

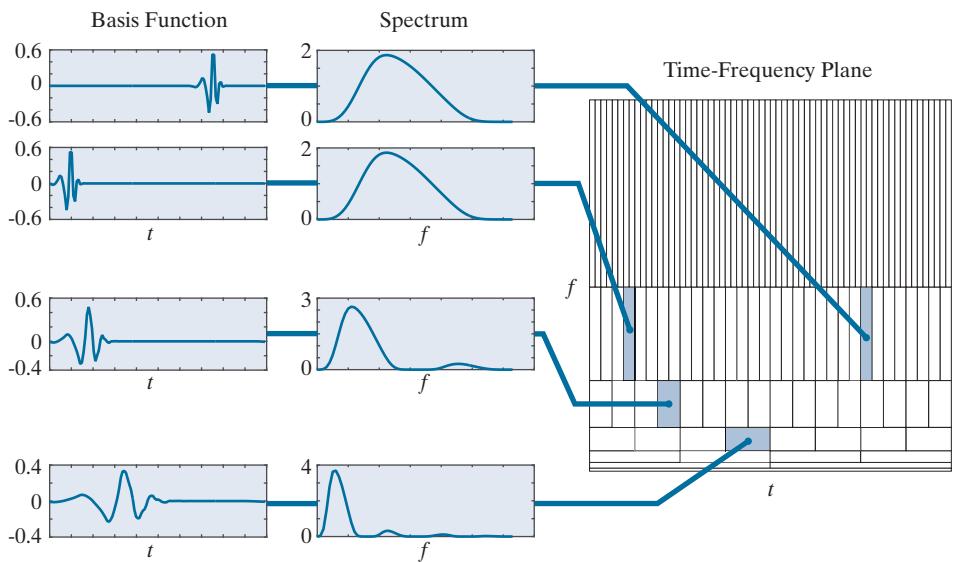
Thus, $|\Im\{\psi(t - \tau)\}| = |\Psi(f)|$ and the spectra of the time-shifted wavelets are identical. This is demonstrated by the basis functions in Figs. 7.5(a) and (b). Note their Heisenberg cells are identical in size and differ only in position.

A principle consequence of the preceding comments is that each wavelet basis function is characterized by a unique spectrum and location in time. Thus, the transform coefficients of a wavelet-based transform, as inner products measuring

a
b
c
d

FIGURE 7.5

Time and frequency localization of 128-point Daubechies basis functions.



the similarity of the function being transformed and the associated wavelet basis functions, provide both frequency and temporal information. They furnish the equivalent of a musical score for the function being transformed, revealing not only what notes to play but also *when* to play them. This is true for all the wavelet bases depicted in the bottom half of Fig. 7.3. The bases in the top half of the figure provide only the notes; temporal information is lost in the transformation process or is difficult to extract from the transform coefficients (e.g., from the phase component of a Fourier transform).

7.5 BASIS IMAGES

Since inverse transformation kernel $s(x, y, u, v)$ in Eq. (7-32) of Section 7.2 depends only on indices x, y, u, v , and not on the values of $f(x, y)$ or $T(u, v)$, Eq. (7-32) can be alternately written as the matrix sum

$$\mathbf{F} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v) \mathbf{S}_{u,v} \quad (7-75)$$

where \mathbf{F} is an $N \times N$ matrix containing the elements of $f(x, y)$ and

$$\mathbf{S}_{u,v} = \begin{bmatrix} s(0, 0, u, v) & s(0, 1, u, v) & \dots & s(0, N-1, u, v) \\ s(1, 0, u, v) & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ s(N-1, 0, u, v) & s(N-1, 1, u, v) & \dots & s(N-1, N-1, u, v) \end{bmatrix} \quad (7-76)$$

for $u, v = 0, 1, \dots, N - 1$. \mathbf{F} is then explicitly defined as a linear combination of N^2 matrices of size $N \times N$ —that is, the $\mathbf{S}_{u,v}$ for $u, v = 0, 1, \dots, N - 1$. If the underlying $s(x, y, u, v)$ are real-valued, separable, and symmetric,

$$\mathbf{S}_{u,v} = \mathbf{s}_u \mathbf{s}_v^T \quad (7-77)$$

where \mathbf{s}_u and \mathbf{s}_v are as previously defined by Eq. (7-22). In the context of digital image processing, \mathbf{F} is a 2-D image and the $\mathbf{S}_{u,v}$ are called *basis images*. They can be arranged in an $N \times N$ array, as shown in Fig. 7.6(a), to provide a concise visual representation of the 2-D basis functions they represent.

EXAMPLE 7.8: The basis images of the standard basis.

The basis in Fig. 7.3(h) is a specific instance (for $N = 16$) of *standard basis* $\{\mathbf{e}_0, \mathbf{e}_1, \dots, \mathbf{e}_{N-1}\}$, where \mathbf{e}_n is an $N \times 1$ column vector whose n th element is 1 and all other elements are 0. Because it is real and orthonormal, the corresponding orthogonal transformation matrix [see Eq. (7-24)] is $\mathbf{A} = \mathbf{I}$, while the corresponding 2-D transform [see Eq. (7-35)] is $\mathbf{T} = \mathbf{A}\mathbf{F}\mathbf{A}^T = \mathbf{I}\mathbf{F}\mathbf{I}^T = \mathbf{F}$. That is, the transform of \mathbf{F} with respect to the standard basis is \mathbf{F} —a confirmation of the fact that when a discrete function is written in vector form, it is represented implicitly with respect to the standard basis.

Figure 7.6(b) shows the basis images of a 2-D standard basis of size 8×8 . Like the 1-D basis vectors in Fig. 7.3(h), which are nonzero at only one instant of time (or value of x), the basis images in Fig. 7.6(b) are nonzero at only one point on the xy -plane. This follows from Eq. (7-77), since $\mathbf{S}_{u,v} = \mathbf{e}_u \mathbf{e}_v^T = \mathbf{E}_{u,v}$, where $\mathbf{E}_{u,v}$ is an $N \times N$ matrix of zeros with a 1 in the u th row and v th column. In the same way, the DFT basis images in Fig. 7.7 follow from Eq. (7-77), Eq. (7-22), and the defining equation of the 1-D DFT expansion functions [i.e., Eq. (7-56)]. Note the DFT basis image of maximum frequency occurs when u and v are 4, just as the 1-D DFT basis function of maximum frequency occurred at $u = 4$ in Fig. 7.2.

7.6 FOURIER-RELATED TRANSFORMS

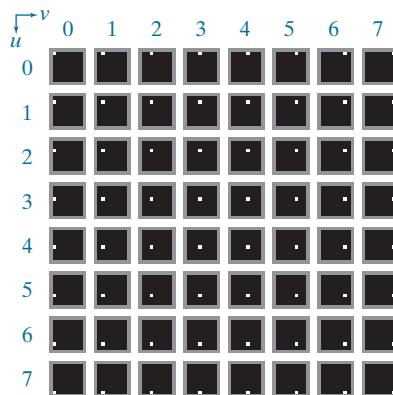
As was noted in Chapter 4, the Fourier transform of a real function is complex-valued. In this section, we examine three Fourier-related transforms that are real rather

a b

FIGURE 7.6

(a) Basis image organization and (b) a standard basis of size 8×8 . For clarity, a gray border has been added around each basis image. The origin of each basis image (i.e., $x = y = 0$) is at its top left.

$\mathbf{S}_{0,0}$	$\mathbf{S}_{0,1}$	$\mathbf{S}_{0,N-1}$
$\mathbf{S}_{1,0}$	⋮			⋮
⋮			⋮	
		⋮		
$\mathbf{S}_{N-1,0}$	$\mathbf{S}_{N-1,N-1}$



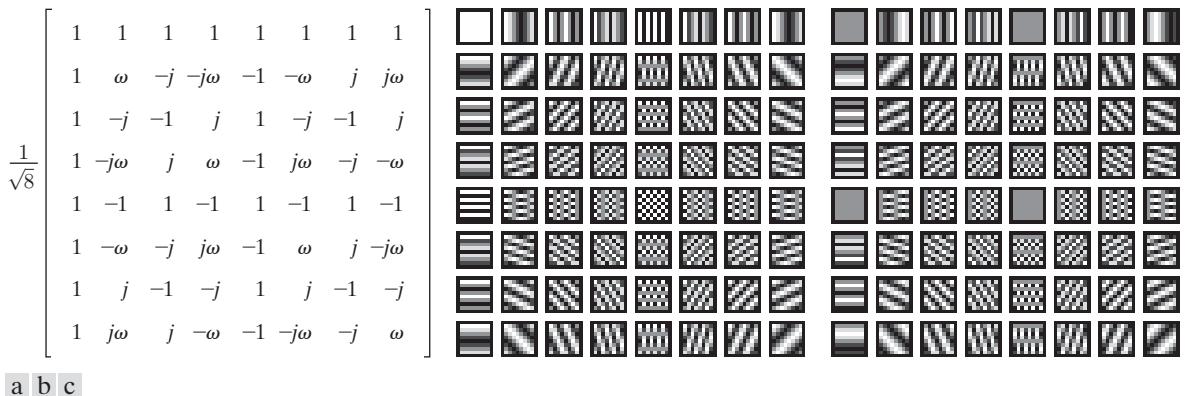


FIGURE 7.7 (a) Transformation matrix \mathbf{A}_F of the discrete Fourier transform for $N = 8$, where $\omega = e^{-j2\pi/8}$ or $(1 - j)/\sqrt{2}$. (b) and (c) The real and imaginary parts of the DFT basis images of size 8×8 . For clarity, a black border has been added around each basis image. For 1-D transforms, matrix \mathbf{A}_F is used in conjunction with Eqs. (7-43) and (7-44); for 2-D transforms, it is used with Eqs. (7-41) and (7-42).

than complex-valued—the *discrete Hartley transform*, *discrete cosine transform*, and *discrete sine transform*. All three transforms avoid the computational complexity of complex numbers and can be implemented via fast FFT-like algorithms.

THE DISCRETE HARTLEY TRANSFORM

The transformation matrix of the *discrete Hartley transform* (DHT) is obtained by substituting the inverse transformation kernel

Function cas, an acronym for the *cosine-and-sin* function, is defined as $\text{cas}(\theta) = \cos(\theta) + \sin(\theta)$.

$$\begin{aligned} s(x, u) &= \frac{1}{\sqrt{N}} \text{cas}\left(\frac{2\pi ux}{N}\right) \\ &= \frac{1}{\sqrt{N}} \left[\cos\left(\frac{2\pi ux}{N}\right) + \sin\left(\frac{2\pi ux}{N}\right) \right] \end{aligned} \quad (7-78)$$

We will not consider the non-separable form

$$s(x, y, u, v) = \frac{1}{N} \text{cas}\left(\frac{2\pi(ux + vy)}{N}\right).$$

$$s(x, y, u, v) = \left[\frac{1}{\sqrt{N}} \text{cas}\left(\frac{2\pi ux}{N}\right) \right] \left[\frac{1}{\sqrt{N}} \text{cas}\left(\frac{2\pi vy}{N}\right) \right] \quad (7-79)$$

into Eqs. (7-22) and (7-24). Since the resulting DHT transformation matrix—denoted \mathbf{A}_{HY} in Fig. 7.8—is real, orthogonal, and symmetric, $\mathbf{A}_{HY}^T = \mathbf{A}_{HY}^{-1} = \mathbf{A}_{HY}$ and \mathbf{A}_{HY} can be used in the computation of both forward and inverse transforms. For 1-D transforms, \mathbf{A}_{HY} is used in conjunction with Eqs. (7-28) and (7-29) of Section 7.2; for 2-D transforms, Eqs. (7-35) and (7-36) are used. Since \mathbf{A}_{HY} is symmetric, the forward and inverse transforms are identical.

Note the similarity of the harmonically related DHT basis functions in Fig. 7.8(a) and the real part of the DFT basis functions in Fig. 7.2. It is easy to show that

$$\begin{aligned}\mathbf{A}_{\text{HY}} &= \text{Real} \{ \mathbf{A}_F \} - \text{Imag} \{ \mathbf{A}_F \} \\ &= \text{Real} \{ (1+j)\mathbf{A}_F \}\end{aligned}\quad (7-80)$$

where \mathbf{A}_F denotes the unitary transformation matrix of the DFT. Furthermore, since the real part of the DFT kernel [see Eq. (7-56)] is

In Eqs. (7-81) and (7-82), subscripts _{HY} and _F are used to denote the Hartley and Fourier kernels, respectively.

$$\text{Re} \{ s_F(x, u) \} = \text{Re} \left\{ \frac{1}{\sqrt{N}} e^{j2\pi ux/N} \right\} = \frac{1}{\sqrt{N}} \cos \left(\frac{2\pi ux}{N} \right) \quad (7-81)$$

and trigonometric identity $\cos(\theta) = \sqrt{2} \cos(\theta - \pi/4)$ can be used to rewrite the discrete Hartley kernel [see Eq. (7-78)] as

$$s_H(x, u) = \sqrt{\frac{2}{N}} \cos \left(\frac{2\pi ux}{N} - \frac{\pi}{4} \right) \quad (7-82)$$

the basis functions of the discrete Fourier and Hartley transforms are scaled and shifted versions of one another—i.e., scaled by the $\sqrt{2}$ and shifted by $\pi/4$. The shift is clearly evident when comparing Figs. 7.2 and 7.8(a). Additionally, for a given value of N and sampling interval ΔT , the Fourier and Hartley transforms have the same frequency resolution $\Delta u = 1/(N\Delta T)$, same range of frequencies $0.5R = 0.5(1/\Delta T) = 1/(2\Delta T)$, and are both undersampled when $u > N/2$. Compare Figs. 7.2 and 7.8(a) for $u = 5, 6, 7$. Finally, we note the 8×8 basis images of the two transforms are also similar. As can be seen in Figs. 7.8(c) and 7.7(b), for example, the basis images of maximum frequency occur when u and v are $N/2$ or 4.

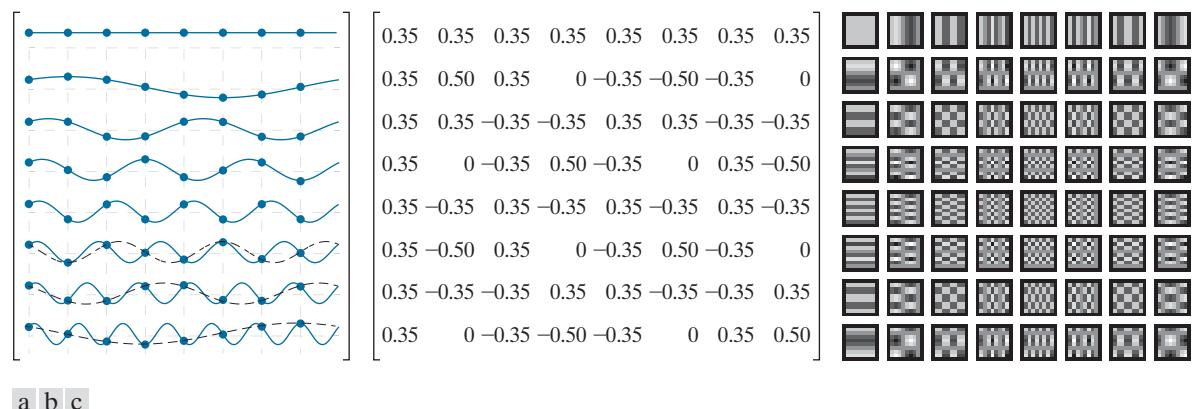


FIGURE 7.8 The transformation matrix and basis images of the discrete Hartley transform for $N = 8$: (a) Graphical representation of orthogonal transformation matrix \mathbf{A}_{HY} , (b) \mathbf{A}_{HY} rounded to two decimal places, and (c) 2-D basis images. For 1-D transforms, matrix \mathbf{A}_{HY} is used in conjunction with Eqs. (7-28) and (7-29); for 2-D transforms, it is used with Eqs. (7-35) and (7-36).

EXAMPLE 7.9: DHT and DFT reconstruction.

Consider discrete function $\mathbf{f} = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^T$ and its discrete Fourier transform

$$\mathbf{t}_F = [0.71 \ 0.6 - j0.25 \ 0.35 - j0.35 \ 0.1 - j0.25 \ 0 \ 0.1 + j0.25 \ 0.35 + j0.35 \ 0.6 + j0.25]^T$$

where $\mathbf{t}_F = \mathbf{A}_F \mathbf{f}$ and $\mathbf{A}_F = \mathbf{A}_{Fr} + j\mathbf{A}_{Fj}$ is the 8×8 unitary transformation matrix of Fig. 7.7(a). The real and imaginary parts of \mathbf{t}_F , denoted \mathbf{t}_{Fr} and \mathbf{t}_{Fj} , are

$$\mathbf{t}_{Fr} = [0.71 \ 0.60 \ 0.35 \ 0.10 \ 0 \ 0.10 \ 0.35 \ 0.60]^T$$

$$\mathbf{t}_{Fj} = [0 \ -0.25 \ -0.35 \ -0.25 \ 0 \ 0.25 \ 0.35 \ 0.25]^T$$

and discrete Hartley transform $\mathbf{t}_{HY} = \mathbf{A}_{HY} \mathbf{f} = (\mathbf{A}_{Fr} - \mathbf{A}_{Fj}) \mathbf{f} = \mathbf{A}_{Fr} \mathbf{f} - \mathbf{A}_{Fj} \mathbf{f} = \mathbf{t}_{Fr} - \mathbf{t}_{Fj}$ is

$$\mathbf{t}_{HY} = [0.71 \ 0.85 \ 0.71 \ 0.35 \ 0 \ -0.15 \ 0 \ 0.35]^T$$

In accordance with Eq. (7-17), \mathbf{f} can be written as

$$f(x) = \sum_{u=0}^7 T_{HY}(u) s_{HY}(x, u) \quad \text{for } x = 0, 1, \dots, 7$$

where $\mathbf{f} = [f(0) \ f(1) \ \dots \ f(7)]^T$ and $\mathbf{t}_{HY} = [T_{HY}(0) \ T_{HY}(1) \ \dots \ T_{HY}(7)]^T$. Thus, \mathbf{f} can be reconstructed from \mathbf{t}_{HY} as a sum of products involving the computed transform coefficients and corresponding basis functions. In Fig. 6.9(a), such a reconstruction is done progressively, beginning with the average or DC value of \mathbf{f} (for $u = 0$) at the top of the figure and converging to \mathbf{f} (for $u = 0, 1, \dots, 7$) at the bottom of the figure. As higher frequency basis functions are included in the sum, the reconstructed function becomes a better approximation of \mathbf{f} , with perfect reconstruction achieved when all eight weighted basis functions are summed to generate the equivalent of inverse discrete Hartley transform $\mathbf{f} = \mathbf{A}_{HY}^T \mathbf{t}_{HY}$. A similar progression is shown in Fig. 7.9(b) for the DFT.

THE DISCRETE COSINE TRANSFORM

The transformation matrix of the most commonly encountered form of the *discrete cosine transform* (DCT) is obtained by substituting the inverse transformation kernel

$$s(x, u) = \alpha(u) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \quad (7-83)$$

where

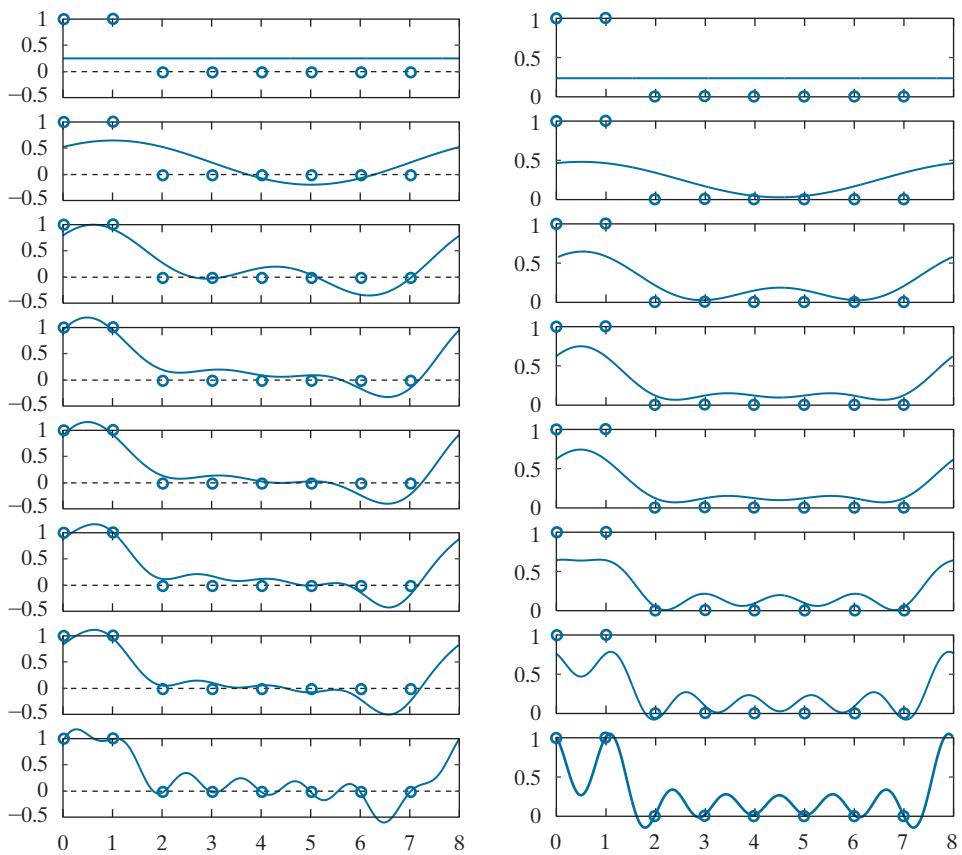
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, \dots, N-1 \end{cases} \quad (7-84)$$

There are eight standard DCT variants and they assume different symmetry conditions. For example, the input could be assumed to be even about a sample or about a point halfway between two samples.

a | b

FIGURE 7.9

Reconstructions of a discrete function by the addition of progressively higher frequency components: (a) DHT and (b) DFT.



into Eqs. (7-22) and (7-24). The resulting transformation matrix, denoted as \mathbf{A}_C in Fig. 7.10, is real and orthogonal, but not symmetric. The underlying basis functions are harmonically related cosines of frequency 0 to $R = [(N - 1)/N][1/(2\Delta T)]$; the spacing between adjacent frequencies (i.e., the frequency resolution) is $\Delta u = 1/(2N\Delta T)$. A comparison of Fig. 7.10(a) to either Figs. 7.8(a) or 7.2 reveals that the spectrum of a discrete cosine transform has roughly the same frequency range as that of the Fourier and Hartley transforms, but twice the frequency resolution. If $N = 4$ and $\Delta T = 1$, for example, the resulting DCT coefficients are at frequencies $\{0, 0.5, 1, 1.5\}$, while the DFT spectral components correspond to frequencies $\{0, 1, 2, 1\}$. Figures 7.10(c) and 7.8(c) further illustrate the point. Note that the DCT basis image of maximum frequency occurs when u and v are 7, as opposed to 4 for the DFT. Since 2-D DCTs are based on the separable inverse transformation kernel

$$s(x, y, u, v) = \alpha(u)\alpha(v)\cos\left(\frac{(2x + 1)u\pi}{2N}\right)\cos\left(\frac{(2y + 1)v\pi}{2N}\right) \quad (7-85)$$

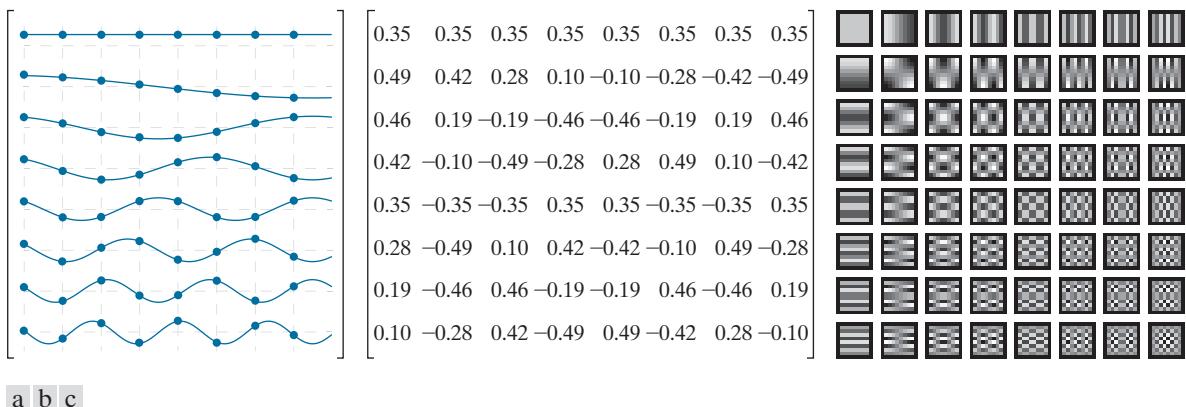


FIGURE 7.10 The transformation matrix and basis images of the discrete cosine transform for $N = 8$. (a) Graphical representation of orthogonal transformation matrix \mathbf{A}_C , (b) \mathbf{A}_C rounded to two decimal places, and (c) basis images. For 1-D transforms, matrix \mathbf{A}_C is used in conjunction with Eqs. (7-28) and (7-29); for 2-D transforms, it is used with Eqs. (7-35) and (7-36).

where $\alpha(u)$ and $\alpha(v)$ are defined in accordance with Eq. (7-84), transformation matrix \mathbf{A}_C can be used in the computation of both 1- and 2-D transforms (see the caption of Fig. 7.10 for the appropriate transform equations).

While sharing several attributes of the discrete Fourier transform, the discrete cosine transform imposes a entirely different set of assumptions on the functions being processed. Rather than N -point periodicity, the underlying assumption of the DFT, the discrete cosine transform assumes $2N$ -point periodicity *and* even symmetry. As can be seen in Fig. 7.11, while N -point periodicity can cause boundary discontinuities that introduce “artificial” high-frequency components into a transform, $2N$ -point periodicity and even symmetry minimize discontinuity, as well as the accompanying high-frequency artifact. As will be seen in Chapter 8, this is an important advantage of the DCT in image compression. In light of the above comments, it should come as no surprise that the DCT of N -point function $f(x)$ can be obtained from the DFT of a $2N$ -point symmetrically extended version of $f(x)$:

1. Symmetrically extend N -point discrete function $f(x)$ to obtain

$$g(x) = \begin{cases} f(x) & \text{for } 0 \leq x < N \\ f(2N - x - 1) & \text{for } N \leq x < 2N \end{cases} \quad (7-86)$$

where $\mathbf{f} = [f(0) \ f(1) \ \dots \ f(N-1)]^T$ and $\mathbf{g} = [g(0) \ g(1) \ \dots \ g(2N-1)]^T$.

2. Compute the $2N$ -point discrete Fourier transform of \mathbf{g} :

$$\mathbf{t}_F = \mathbf{A}_F \mathbf{g} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix} \quad (7-87)$$

where \mathbf{A}_F is the transformation matrix of the DFT and $2N$ -element transform \mathbf{t}_F is partitioned into two equal-length N -element column vectors, \mathbf{t}_1 and \mathbf{t}_2 .

3. Let N -element column vector $\mathbf{h} = [h(0) \ h(1) \ \dots \ h(N-1)]^T$ where

$$h(u) = e^{-j\pi u/2N} \quad \text{for } u = 0, 1, \dots, N-1 \quad (7-88)$$

and let $\mathbf{s} = [1/\sqrt{2} \ 1 \ 1 \ \dots \ 1]^T$.

4. The discrete cosine transform of \mathbf{f} is then

$$\mathbf{t}_C = \operatorname{Re}\{\mathbf{s} \circ \mathbf{h} \circ \mathbf{t}_1\} \quad (7-89)$$

where \circ denotes the *Hadamard product*, a matrix multiplication in which the corresponding elements of two vectors or matrices are multiplied together—for example, $[3 \ -0.5] \circ [2 \ 6] = [6 \ -3]$.

EXAMPLE 7.10: Computing a 4-point DCT from a 8-point DFT.

In this example, we use Eqs. (7-86) through (7-89) to compute the discrete cosine transform of 1-D function $f(x) = x^2$ for $x = 0, 1, 2, 3$.

1. Let $\mathbf{f} = [0 \ 1 \ 4 \ 9]^T$ and use Eq. (7-86) to create an 8-point extension of \mathbf{f} with even symmetry. Extended function $\mathbf{g} = [0 \ 1 \ 4 \ 9 \ 9 \ 4 \ 1 \ 0]^T$ is one period of an even symmetric function like the one in Fig. 7.11(b).
2. Substituting the 8×8 unitary transformation matrix from Fig. 7.7(a) into Eq. (7-87), the discrete Fourier transform of \mathbf{g} is

$$\mathbf{t}_F = \mathbf{A}_F \mathbf{g} = \begin{bmatrix} -9.9 \\ -6.18 - j2.56 \\ 1.41 + j1.41 \\ -0.18 - j0.44 \\ 0 \\ -0.18 + j0.44 \\ 1.41 - j1.41 \\ -6.18 + j2.56 \end{bmatrix} \text{ so } \mathbf{t}_1 = \begin{bmatrix} -9.9 \\ -6.18 - j2.56 \\ 1.41 + j1.41 \\ -0.18 - j0.44 \end{bmatrix} \text{ and } \mathbf{t}_2 = \begin{bmatrix} 0 \\ -0.18 + j0.44 \\ 1.41 - j1.41 \\ -6.18 + j2.56 \end{bmatrix}$$

3. In accordance with Eq. (7-88),

$$\mathbf{h} = \begin{bmatrix} 1 \\ e^{-j\pi/4} \\ e^{-j\pi/2} \\ e^{-j3\pi/4} \end{bmatrix} = \begin{bmatrix} 1 \\ 0.92 - j0.38 \\ 0.71 - j0.71 \\ 0.38 - j0.92 \end{bmatrix}$$

and $\mathbf{s} = \begin{bmatrix} 1/\sqrt{2} & 1 & 1 & 1 \end{bmatrix}^T = [0.71 \ 1 \ 1 \ 1]^T$.

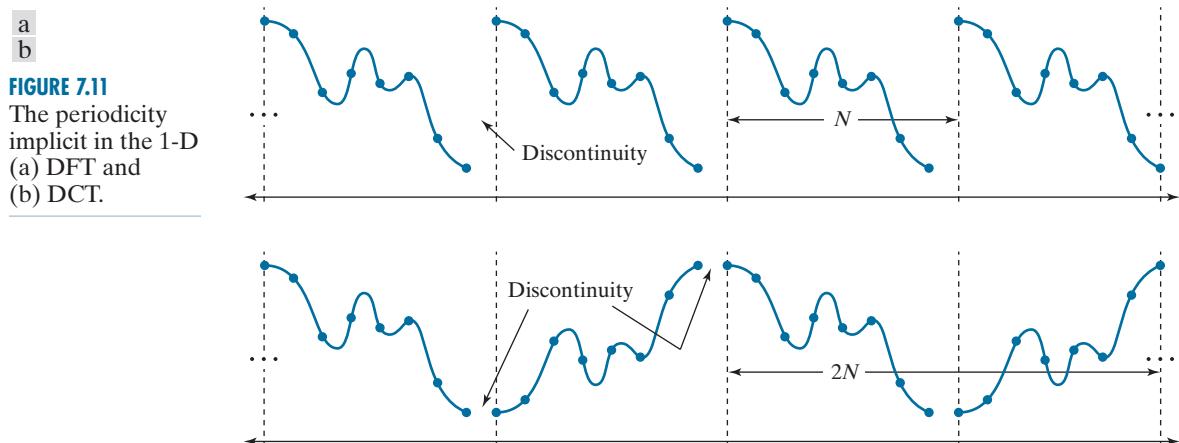
4. The discrete cosine transform of \mathbf{f} is then

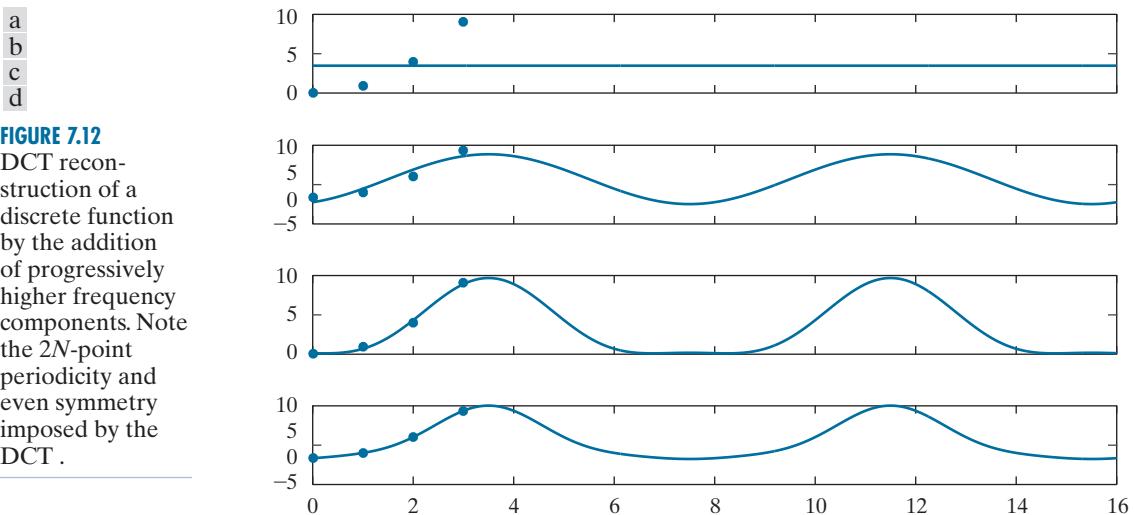
$$\mathbf{t}_C = \operatorname{Re}\{\mathbf{s} \circ \mathbf{h} \circ \mathbf{t}_1\} = \operatorname{Re} \left\{ \begin{bmatrix} 0.71 \\ 1 \\ 1 \\ 1 \end{bmatrix} \circ \begin{bmatrix} 1 \\ 0.92 - j0.38 \\ 0.71 - j0.71 \\ 0.38 - j0.92 \end{bmatrix} \circ \begin{bmatrix} -9.9 \\ -6.18 - j2.56 \\ 1.41 + j1.41 \\ -0.18 - j0.44 \end{bmatrix} \right\} = \begin{bmatrix} 7 \\ -6.69 \\ 2 \\ -0.48 \end{bmatrix}$$

To validate the result, we substitute Eq. (7-83) into Eqs. (7-22) and (7-24) with $N = 4$ and use the resulting 4×4 DCT transformation matrix in Eq. (7-28) to obtain

$$\mathbf{t}_C = \mathbf{A}_C \mathbf{f} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 4 \\ 9 \end{bmatrix} = \begin{bmatrix} 7 \\ -6.69 \\ 2 \\ -0.48 \end{bmatrix}$$

Figure 7.12 illustrates the reconstruction of \mathbf{f} by the inverse discrete cosine transform. Like the reconstructions in Fig. 7.9, the DC component at the top of the figure [i.e., Fig. 7.12(a)] is the average value of the discrete function—in this case, $(0+1+4+9)/4 = 3.5$. It is an initial but crude approximation of \mathbf{f} . As three additional cosines of increasing frequency are added in the (b), (c), and (d) parts of the figure, the accuracy of the approximation increases until a perfect reconstruction is achieved in (d). Note the x -axis has been extended to show that the resulting DCT expansion is indeed periodic with period $2N$ (in this case 8) and exhibits the even symmetry that is required of all discrete cosine transforms.



**FIGURE 7.12**

DCT reconstruction of a discrete function by the addition of progressively higher frequency components. Note the $2N$ -point periodicity and even symmetry imposed by the DCT.

Like the DCT, there are eight variants and they assume different symmetry conditions—for instance, is the input odd about a sample or about a point halfway between two samples?

THE DISCRETE SINE TRANSFORM

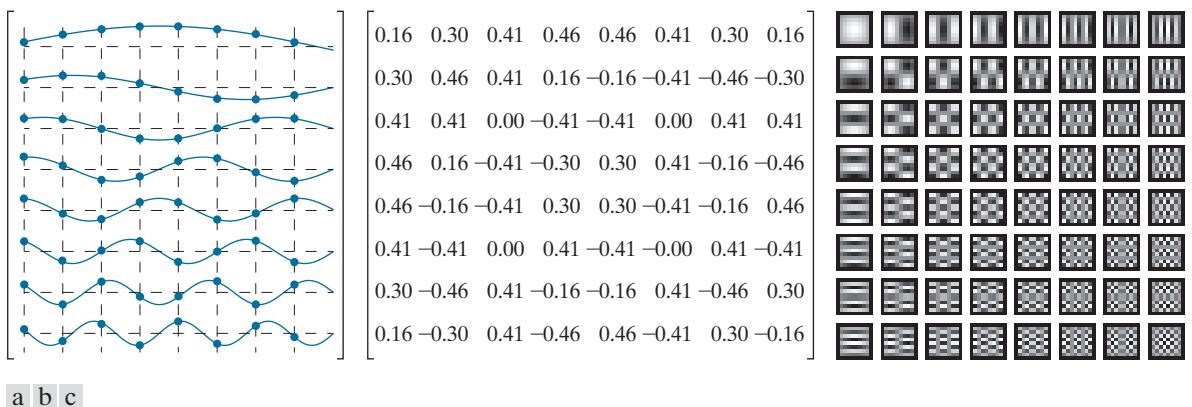
The transformation matrix of the *discrete sine transform* (DST) is obtained by substituting the inverse transformation kernel

$$s(x, u) = \sqrt{\frac{2}{N+1}} \sin\left(\frac{(x+1)(u+1)\pi}{N+1}\right) \quad (7-90)$$

whose separable 2-D counterpart is

$$s(x, y, u, v) = \frac{2}{N+1} \sin\left(\frac{(x+1)(u+1)\pi}{N+1}\right) \sin\left(\frac{(y+1)(v+1)\pi}{N+1}\right) \quad (7-91)$$

into Eqs. (7-22) and (7-24). The resulting transformation matrix, denoted as \mathbf{A}_S in Fig. 7.13, is real, orthogonal, and symmetric. As can be seen in the (a) part of the figure, the underlying basis functions are harmonically related sines of frequency $1/[2(N+2)\Delta T]$ to $N/[2(N+2)\Delta T]$; the frequency resolution or the spacing between adjacent frequencies is $\Delta u = 1/[2(N+2)\Delta T]$. Like the DCT, the DST has roughly the same frequency range as the DFT, but twice the frequency resolution. If $N = 4$ and $\Delta T = 1$, for example, the resulting DST coefficients are at frequencies $\{0.4, 0.8, 1.2, 1.6\}$. Note unlike both the DCT and DFT, the DST has no DC (at $u = 0$) component. This results from an underlying assumption that the function being transformed is $2(N+1)$ -point periodic and odd symmetric, making its average value zero. In contrast to the DCT, where the function is assumed to be even, the odd symmetry that is imposed by the DST does not reduce boundary discontinuity. This is clear in Fig. 6.14, where the result of computing the forward and inverse DCT of $f(x) = x^2$ for $x = 0, 1, 2, 3$ is shown. Note that the underlying continuous



a b c

FIGURE 7.13 The transformation matrix and basis images of the discrete sine transform for $N = 8$. (a) Graphical representation of orthogonal transformation matrix \mathbf{A}_S , (b) \mathbf{A}_S rounded to two decimal places, and (c) basis images. For 1-D transforms, matrix \mathbf{A}_S is used in conjunction with Eqs. (7-28) and (7-29); for 2-D transforms, it is used with Eqs. (7-35) and (7-36).

reconstruction, which was obtained by the same process that led to Fig. 6.12(d), exhibits the aforementioned periodicity, odd symmetry, and boundary discontinuity.

The discrete sine transform of an N -point function $f(x)$ can be obtained from the DFT of a $2(N + 1)$ -point symmetrically extended version of $f(x)$ with odd symmetry:

1. Symmetrically extend N -point function $f(x)$ to obtain

$$g(x) = \begin{cases} 0 & \text{for } x = 0 \\ f(x-1) & \text{for } 1 \leq x \leq N \\ 0 & \text{for } x = N+1 \\ -f(2N-x+1) & \text{for } N+2 \leq x \leq 2N+2 \end{cases} \quad (7-92)$$

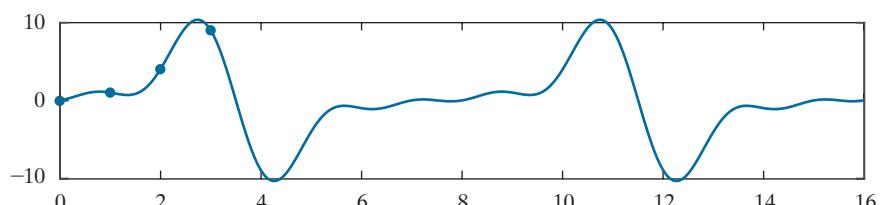
where $\mathbf{f} = [f(0) \ f(1) \ \dots \ f(N-1)]^T$ and $\mathbf{g} = [g(0) \ g(1) \ \dots \ g(2N+2)]^T$.

2. Compute the $2(N + 1)$ -point discrete Fourier transform of \mathbf{g} :

$$\mathbf{t}_F = \mathbf{A}_F \mathbf{g} = \begin{bmatrix} 0 \\ \mathbf{t}_1 \\ 0 \\ \mathbf{t}_2 \end{bmatrix} \quad (7-93)$$

FIGURE 7.14

A reconstruction of the DST of the function defined in Example 6.10.



where \mathbf{A}_F is the transformation matrix of the DFT and $2(N+1)$ -element transform \mathbf{t}_F is partitioned into two single-element zero vectors, $\mathbf{0} = [0]$, and two N -element column vectors \mathbf{t}_1 and \mathbf{t}_2 .

3. The discrete sine transform of \mathbf{f} , denoted \mathbf{t}_S , is then

$$\mathbf{t}_S = -\text{Imag}\{\mathbf{t}_1\} \quad (7-94)$$

EXAMPLE 7.11: Computing a 4-point DST from a 10-point DFT.

In this example, we use Eqs. (7-92) through (7-94) to find the DST of $\mathbf{f} = [0 \ 1 \ 4 \ 9]^T$ from Example 7.10:

1. Create a $2(N+1)$ -point extended version of \mathbf{f} with odd symmetry. In accordance with Eq. (7-92), $\mathbf{g} = [0 \ 0 \ 1 \ 4 \ 9 \ 0 \ -9 \ -4 \ -1]^T$.
2. Compute the discrete Fourier transform of \mathbf{g} using Eq. (7-93). Matrix \mathbf{A}_F is a unitary DFT transformation matrix of size 10×10 and the resulting transform is

$$\mathbf{t}_F = \mathbf{A}_F \mathbf{g} = [0 \ -j6.35 \ j6.53 \ -j3.56 \ j1.54 \ 0 \ j6.35 \ -j6.53 \ j3.56 \ -j1.54]^T$$

Note the real part of \mathbf{t}_F is zero and block \mathbf{t}_1 of \mathbf{t}_F is $[-j6.35 \ j6.53 \ -j3.56 \ j1.54]^T$.

3. In accordance with Eq. (7-94), the DST of \mathbf{f} is then

$$\mathbf{t}_S = -\text{Imag}\{\mathbf{t}_1\} = [6.35 \ -6.53 \ 3.56 \ -1.54]^T$$

Alternately, the DST can be computed directly as

$$\mathbf{t}_S = \mathbf{A}_S \mathbf{f} = \begin{bmatrix} 0.37 & 0.60 & 0.60 & 0.37 \\ 0.60 & 0.37 & -0.37 & -0.60 \\ 0.60 & -0.37 & -0.37 & 0.60 \\ 0.37 & -0.60 & 0.60 & -0.37 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 4 \\ 9 \end{bmatrix} = \begin{bmatrix} 6.35 \\ -6.53 \\ 3.56 \\ -1.54 \end{bmatrix}$$

where \mathbf{A}_S is obtained by substituting Eq. (7-90) into Eqs. (7-22) and (7-24) with $N = 4$.

EXAMPLE 7.12: Ideal lowpass filtering with Fourier-related transforms.

Figure 7.15 shows the results of applying an ideal lowpass filter to the test image that was used in Example 4.16 with all of the Fourier-related transforms that have been covered in this chapter. As in the Chapter 4 example, the test image shown in Fig. 7.15(a) is of size 688×688 and is padded to 1376×1376 before computing any transforms. For reference, the Fourier transform of the test image is shown in Fig. 7.15(b), where a blue overlay has been superimposed to show the lowpass filter function. Only the frequencies that are not shaded blue are passed by the filter. Since we are again using a cutoff frequency with a radius of 60, the filtered result in Fig. 7.15(c) is similar to that of Fig. 4.41(d), with any differences due to the use of zero padding rather than mirror padding. Note once more the blurring and ringing that was discussed in Example 4.16.

Figures 7.15(d)–(i) provide comparable results using the three Fourier-related transforms covered in this chapter. As was done for the Fourier transform in Fig. 6.15(b), Figs. 6.15(d)–(f) show the

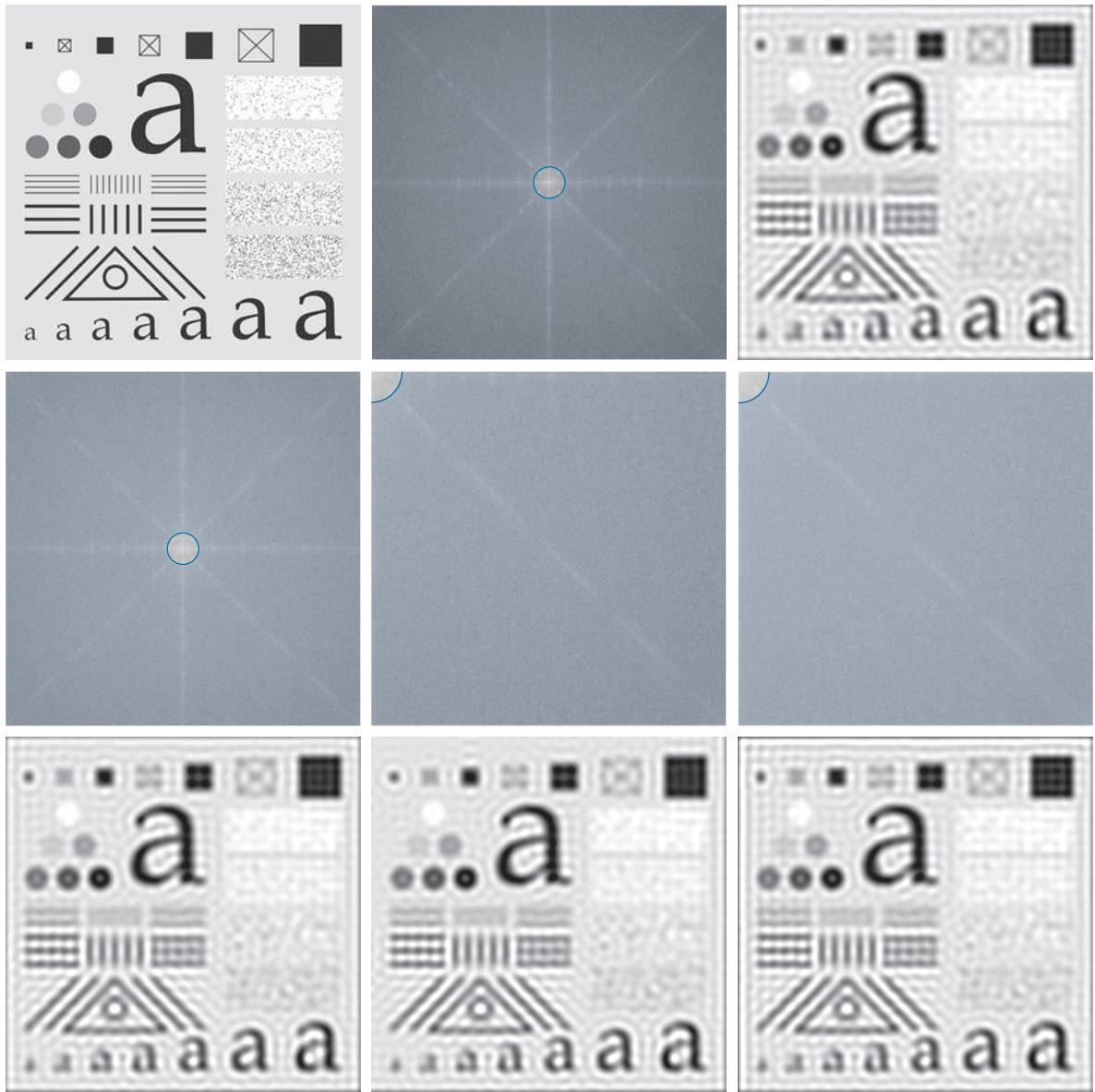


FIGURE 7.15 (a) Original image of the 688×688 test pattern from Fig. 4.41(a). (b) Discrete Fourier transform (DFT) of the test pattern in (a) after padding to size 1376×1376 . The blue overlay is an ideal lowpass filter (ILPF) with a radius of 60. (c) Result of Fourier filtering. (d)–(f) Discrete Hartley transform, discrete cosine transform (DCT), and discrete sine transform (DST) of the test pattern in (a) after padding. The blue overlay is the same ILPF in (b), but appears bigger in (e) and (f) because of the higher frequency resolution of the DCT and DST. (g)–(i) Results of filtering for the Hartley, cosine, and sine transforms, respectively.

discrete Hartley, cosine, and sine transforms of the test image in Fig. 7.15(a) after zero-padding to size 1376×1376 , respectively. Although the filter functions for the cosine and sine transforms, which are again superimposed in blue, appear to have twice the radii of the filters used with the Fourier and Hartley transforms, the same range of frequencies are passed by all filters. The apparent increase in size is due to the greater frequency resolution of the sine and cosine transforms, which has already been discussed. Note the spectra of these transforms do not need to be centered for easy interpretation, as is the case for the Fourier and Hartley spectra. Finally, we note for all practical purposes the filtered images in Figs. 7.15(g)–(i) are equivalent to the Fourier filtered result in Fig. 7.15(c).

To conclude the example, we note while Fourier-related transforms can be implemented in FFT-like algorithms or computed from the FFT itself, we used the matrix implementations that have been presented in this section to compute both the forward and inverse transforms. Using MATLAB®, Windows® 10, and a notebook PC with an Intel® i7-4600U processor at 2.1 GHz, the total times required to compute the Fourier-related transforms in this example were 2 to 5 times longer than the corresponding FFT computations. All computations, however, took less than a second.

7.7 WALSH-HADAMARD TRANSFORMS

Walsh-Hadamard transforms (WHTs) are non-sinusoidal transformations that decompose a function into a linear combination of rectangular basis functions, called *Walsh functions*, of value +1 and -1. The ordering of the basis functions within a Walsh-Hadamard transformation matrix determines the variant of the transform that is being computed. For *Hadamard ordering* (also called *natural ordering*), the transformation matrix is obtained by substituting the inverse transformation kernel

$$s(x, u) = \frac{1}{\sqrt{N}} \sum_{i=0}^{n-1} b_i(x) b_i(u) \quad (7-95)$$

into Eqs. (7-22) and (7-24), where the summation in the exponent of Eq. (7-95) is performed in modulo 2 arithmetic, $N = 2^n$, and $b_k(z)$ is the k th bit in the binary representation of z . For example, if $n = 3$ and $z = 6$ (110 in binary), $b_0(z) = 0$, $b_1(z) = 1$, and $b_2(z) = 1$. If $N = 2$, the resulting Hadamard-ordered transformation matrix is

$$\mathbf{A}_W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (7-96)$$

where the matrix on the right (without the scalar multiplier) is called a *Hadamard matrix* of order 2. Letting \mathbf{H}_N denote the Hadamard matrix of order N , a simple recursive relationship for generating Hadamard-ordered transformation matrices is

$$\mathbf{A}_W = \frac{1}{\sqrt{N}} \mathbf{H}_N \quad (7-97)$$

where

$$\mathbf{H}_{2N} = \begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix} \quad (7-98)$$

\mathbf{A}_W is used to denote the transformation matrix of the Hadamard- or natural-ordered WHT. Although of size 2×2 here, it is more generally of size $N \times N$, where N is the dimension of the discrete function being transformed.

and

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (7-99)$$

Thus, Eq. (7-96) follows from Eqs. (7-97) and (7-99). In the same way,

$$\begin{aligned} \mathbf{H}_4 &= \begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \end{aligned} \quad (7-100)$$

and

$$\begin{aligned} \mathbf{H}_8 &= \begin{bmatrix} \mathbf{H}_4 & \mathbf{H}_4 \\ \mathbf{H}_4 & -\mathbf{H}_4 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \end{aligned} \quad (7-101)$$

The corresponding Hadamard-ordered transformation matrices are obtained by substituting \mathbf{H}_4 and \mathbf{H}_8 into Eq. (7-97).

The number of sign changes along a row of a Hadamard matrix is known as the *sequency* of the row. Like frequency, sequency measures the rate of change of a function, and like the sinusoidal basis functions of the Fourier transform, every Walsh function has a unique sequency. Since the elements of a Hadamard matrix are derived from inverse kernal values, the sequency concept applies to basis functions $s(x, u)$ for $u = 0, 1, \dots, N - 1$ as well. For instance, the sequencies of the \mathbf{H}_4 basis vectors in Eq. (7-100) are 0, 3, 1, 2; the sequencies of the \mathbf{H}_8 basis vectors in Eq. (7-101) are 0, 7, 3, 4, 1, 6, 2, and 5. This arrangement of sequencies is the defining characteristic of a Hadamard-ordered Walsh-Hadamard transform.

Arranging the basis vectors of a Hadamard matrix so the sequency increases as a function of u is both desirable and common in signal and image processing

applications. The transformation matrix of the resulting sequency-ordered Walsh-Hadamard transform is obtained by substituting the inverse transformation kernal

Recall that $N = 2^n$, so
 $n = \log_2 N$.

$$s(x, u) = \frac{1}{\sqrt{N}} \sum_{i=0}^{n-1} b_i(x) p_i(u) \quad (7-102)$$

where

$$\begin{aligned} p_0(u) &= b_{n-1}(u) \\ p_1(u) &= b_{n-1}(u) + b_{n-2}(u) \\ p_2(u) &= b_{n-2}(u) + b_{n-3}(u) \\ &\vdots \\ p_{n-1}(u) &= b_1(u) + b_0(u) \end{aligned} \quad (7-103)$$

into Eqs. (7-22) and (7-24). As before, the summations in Eqs. (7-102) and (7-103) are performed in modulo 2 arithmetic. Thus, for example,

$$\mathbf{H}'_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \quad (7-104)$$

where the apostrophe (') has been added to indicate sequency ordering as opposed to Hadamard ordering. Note the sequences of the rows of \mathbf{H}'_8 match their row numbers—i.e., 0, 1, 2, 3, 4, 5, 6, and 7. An alternate way to generate \mathbf{H}'_8 is to rearrange the rows of Hadamard-ordered \mathbf{H}_8 , noting that row s of \mathbf{H}'_8 corresponds to the row of \mathbf{H}_8 that is the bit-reversed *gray code* of s . Since the n -bit gray code corresponding to $(s_{n-1} \dots s_2 s_1 s_0)_2$ can be computed as

$$\begin{aligned} g_i &= s_i \oplus s_{i+1} & \text{for } 0 \leq i \leq n-2 \\ g_{n-1} &= s_{n-1} & \text{for } i = n-1 \end{aligned} \quad (7-105)$$

where \oplus denotes the exclusive OR operation, row s of \mathbf{H}'_8 is the same as row $(g_0 g_1 g_2 \dots g_{n-1})_2$ of \mathbf{H}_8 . For example, row 4 or $(100)_2$ of \mathbf{H}'_8 , whose gray code is $(110)_2$, comes from row $(011)_2$ or 3 of \mathbf{H}_8 . Note row 4 of \mathbf{H}'_8 in Eq. (7-104) is indeed identical to row 3 of \mathbf{H}_8 in Eq. (7-101).

Figures 7.16(a) and (b) depict graphically and numerically the sequency-ordered WHT transformation matrix for the case of $N = 8$. Note the sequency of the discrete

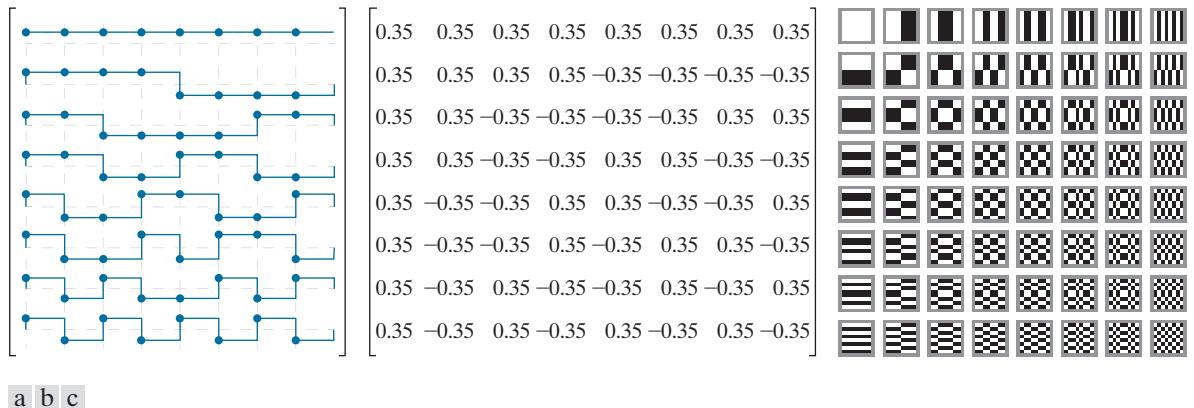


FIGURE 7.16 The transformation matrix and basis images of the sequency-ordered Walsh-Hadamard transform for $N = 8$. (a) Graphical representation of orthogonal transformation matrix $\mathbf{A}_{W'}$, (b) $\mathbf{A}_{W'}$ rounded to two decimal places, and (c) basis images. For 1-D transforms, matrix $\mathbf{A}_{W'}$ is used in conjunction with Eqs. (7-28) and (7-29); for 2-D transforms, it is used with Eqs. (7-35) and (7-36).

basis functions in Fig. 7.16(a) increase as u goes from 0 to 7, as does the sequency of the underlying square wave functions. Note also the transformation matrix in Fig. 7.16(b) is real, symmetric, and follows from Eqs. (7-105) and (7-97) as

$$\mathbf{A}_{W'} = \frac{1}{\sqrt{N}} \mathbf{H}'_8 \quad (7-106)$$

It is left as an exercise for the reader to show that it is orthogonal and that $\mathbf{A}_{W'} = \mathbf{A}_{W'}^T = \mathbf{A}_{W'}^{-1}$. Finally, note the similarity of the sequency-ordered basis images in Fig. 7.16(c), which are based on the separable 2-D inverse transformation kernel

$$s(x, y, u, v) = \frac{1}{N} (-1)^{\sum_{i=0}^{n-1} [b_i(x)p_i(u) + b_i(y)p_i(v)]} \quad (7-107)$$

to the basis images of the 2-D DCT in Fig. 7.10(c). Sequency increases as a function of both u and v , like frequency in the DCT basis images, but does not have as useful a physical interpretation.

EXAMPLE 7.13: A simple sequency-ordered Walsh-Hadamard transform.

To compute the sequency-ordered Walsh-Hadamard transform of the 1-D function $\mathbf{f} = [2 \ 3 \ 4 \ 5]^T$, we begin with the Hadamard-ordered Hadamard matrix \mathbf{H}_4 of Eq. (7-100) and use the procedure described in conjunction with Eq. (7-105) to reorder the basis vectors. The mapping of the Hadamard-ordered basis vectors of \mathbf{H}_4 to the sequency-ordered basis vectors of \mathbf{H}'_4 is computed as follows:

Row of \mathbf{H}'_4	Binary Code	Gray Code	Bit-Reversed Gray Code	Row of \mathbf{H}_4
0	00	00	00	0
1	01	01	10	2
2	10	11	11	3
3	11	10	01	1

Thus, in accordance with Eqs. (7-106), the sequency-ordered Walsh-Hadamard transformation matrix of size 4×4 is

$$\mathbf{A}_{W'} = \frac{1}{\sqrt{4}} \mathbf{H}_{W'} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

and the sequency-ordered transform is $\mathbf{t}_{W'} = \mathbf{A}_{W'} \mathbf{f} = [7 \ -2 \ 0 \ -1]^T$.

7.8 SLANT TRANSFORM

Many monochrome images have large areas of uniform intensity and areas of linearly increasing or decreasing brightness. With the exception of the discrete sine transform, all of the transforms that we have presented to this point include a basis vector (at frequency or sequency $u = 0$) for representing efficiently constant gray level areas, but none has a basis function that is targeted specifically at the representation of linearly increasing or decreasing intensity values. The transform considered in this section, called the *slant transform*, includes such a basis function. The transformation matrix of the slant transform of order $N \times N$ where $N = 2^n$ is generated recursively using

$$\mathbf{A}_{Sl} = \frac{1}{\sqrt{N}} \mathbf{S}_N \quad (7-108)$$

where *slant matrix*

$$\mathbf{S}_N = \begin{bmatrix} 1 & 0 & & 1 & 0 & & \mathbf{0} \\ a_N & b_N & & -a_N & b_N & & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{(N/2)-2} & & \mathbf{0} & \mathbf{I}_{(N/2)-2} & & \mathbf{I}_{(N/2)-2} \\ 0 & 1 & & 0 & -1 & & \mathbf{0} \\ & & \mathbf{0} & & & & \mathbf{0} \\ -b_N & a_N & & b_N & a_N & & -\mathbf{I}_{(N/2)-2} \\ \mathbf{0} & \mathbf{I}_{(N/2)-2} & & \mathbf{0} & & & \end{bmatrix} \begin{bmatrix} \mathbf{S}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{N/2} \end{bmatrix} \quad (7-109)$$

Note \mathbf{I}_1 is a 1×1 identity matrix [1] and \mathbf{I}_0 is the empty matrix of size 0×0 .

Here, \mathbf{I}_N is the identity matrix of order $N \times N$,

$$\mathbf{S}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (7-110)$$

and coefficients a_N and b_N are

$$a_N = \left[\frac{3N^2}{4(N^2 - 1)} \right]^{1/2} \quad (7-111)$$

and

$$b_N = \left[\frac{N^2 - 4}{4(N^2 - 1)} \right]^{1/2} \quad (7-112)$$

for $N > 1$. When $N \geq 8$, matrix \mathbf{S}_N is not sequency ordered, but can be made so using the procedure demonstrated in Example 6.13 for the WHT. An example of the use of Eqs. (7-108) through (7-112) is Slant transformation matrix

$$\mathbf{A}_{\text{Sl}} = \frac{1}{\sqrt{4}} \mathbf{S}_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} \\ 1 & -1 & -1 & 1 \\ \frac{1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} & \frac{3}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{bmatrix} \quad (7-113)$$

Since $N = 4$, the basis vectors of \mathbf{A}_{Sl} (and the rows of slant matrix of \mathbf{S}_4) are sequency ordered.

EXAMPLE 7.14: A simple 1-D slant transform.

Using Eqs. (7-28) and (7-113), the slant transform of function $\mathbf{f} = [2 \ 3 \ 4 \ 5]^T$ from Example 7.13 is $\mathbf{t}_{\text{Sl}} = \mathbf{A}_{\text{Sl}} \mathbf{f} = [7 \ -2.24 \ 0 \ 0]^T$. Note the transform contains only two nonzero terms, while the Walsh-Hadamard transform in the previous example had three nonzero terms. The slant transform represents \mathbf{f} more efficiently because \mathbf{f} is a linearly increasing function—that is, \mathbf{f} is highly correlated with the slant basis vector of sequency one. Thus, there are fewer terms in a linear expansion using slant basis functions as opposed to Walsh basis functions.

Figures 7.17(a) and (b) depict graphically and numerically the sequency-ordered slant transformation matrix for the case of $N = 8$. Just as apostrophes ('') were used to denote sequency ordering in Walsh-Hadamard transforms, \mathbf{S}'_8 and $\mathbf{A}_{\text{Sl}'}$ are used to denote sequency-ordered versions of Eqs. (7-108) and (7-109). Note the slant transformation matrix in Fig. 7.16(b) is real, but not symmetric. Thus, $\mathbf{A}_{\text{Sl}'}^{-1} = \mathbf{A}_{\text{Sl}'}^T$ but $\mathbf{A}_{\text{Sl}'}^T \neq \mathbf{A}_{\text{Sl}'}$. Matrix $\mathbf{A}_{\text{Sl}'}$ is also orthogonal and can be used in conjunction with

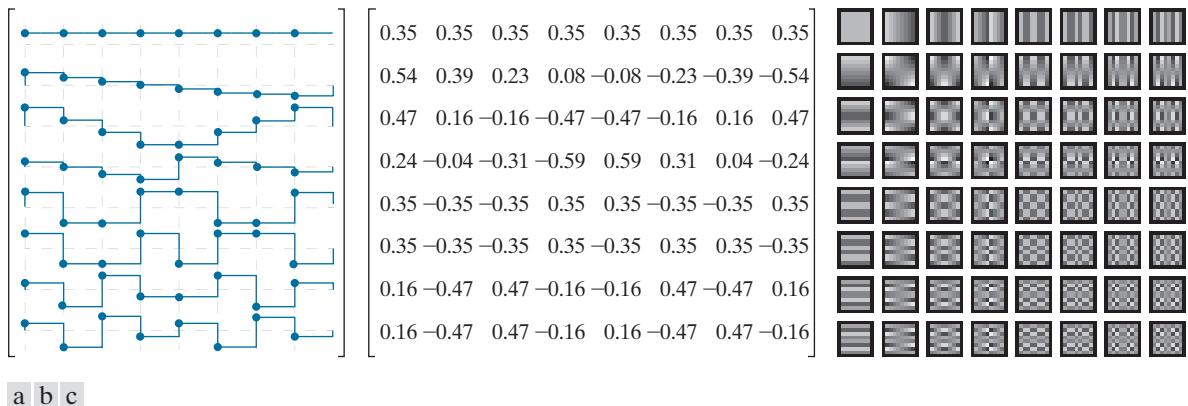


FIGURE 7.17 The transformation matrix and basis images of the slant transform for $N = 8$. (a) Graphical representation of orthogonal transformation matrix $\mathbf{A}_{Sl'}$, (b) $\mathbf{A}_{Sl'}$ rounded to two decimal places, and (c) basis images. For 1-D transforms, matrix $\mathbf{A}_{Sl'}$ is used in conjunction with Eqs. (7-28) and (7-29); for 2-D transforms, it is used with Eqs. (7-35) and (7-36).

Eqs. (7-35) and (7-36) to implement 2-D separable slant transforms. Figure 6.17(c) shows the 2-D slant basis images of size 8×8 . Note for $4 \leq u \leq 5$ and $4 \leq v \leq 5$, they are identical to the corresponding basis images of the WHT in Fig. 7.16(c). This is also evident in Figs. 7.16(a) and 7.17(a) when $4 \leq u \leq 5$. In fact, all of the slant basis vectors bear a striking resemblance to the basis vectors of the Walsh-Hadamard transform. Finally, we note slant matrices have the necessary properties to allow implementation of a fast slant transform algorithm similar to the FFT.

7.9 HAAR TRANSFORM

Discovered in 1910, the basis functions of the *Haar transform* (Haar [1910]) were later recognized to be the oldest and simplest orthonormal wavelets. We will look at Haar's functions in the context of wavelets in the next section. In this section, we approach Haar's transform as another matrix-based transformation that employs a set of rectangular-shaped basis functions.

The Haar transform is based on *Haar functions*, $h_u(x)$, that are defined over the continuous, half-open interval $x \in [0, 1]$. Variable u is an integer that for $u > 0$ can be decomposed uniquely as

$$u = 2^p + q \quad (7-114)$$

where p is the largest power of 2 contained in u and q is the remainder—that is, $q = 2^p - u$. The Haar basis functions are then

$$h_u(x) = \begin{cases} 1 & u = 0 \text{ and } 0 \leq x < 1 \\ 2^{p/2} & u > 0 \text{ and } q/2^p \leq x < (q+0.5)/2^p \\ -2^{p/2} & u > 0 \text{ and } (q+0.5)/2^p \leq x < (q+1)/2^p \\ 0 & \text{otherwise} \end{cases} \quad (7-115)$$

When u is 0, $h_0(x) = 1$ for all x ; the first Haar function is independent of continuous variable x . For all other values of u , $h_u(x) = 0$ except in the half-open intervals $[q/2^p, (q+0.5)/2^p)$ and $[(q+0.5)/2^p, (q+1)/2^p)$, where it is a rectangular wave of magnitude $2^{p/2}$ and $-2^{p/2}$, respectively. Parameter p determines the amplitude and width of both rectangular waves, while q determines their position along x . As u increases, the rectangular waves become narrower and the number of functions that can be represented as linear combinations of the Haar functions increases. Figure 7.18(a) shows the first eight Haar functions (i.e., the curves depicted in blue).

The transformation matrix of the *discrete Haar transform* can be obtained by substituting the inverse transformation kernel

$$s(x,u) = \frac{1}{\sqrt{N}} h_u(x/N) \quad \text{for } x = 0, 1, \dots, N-1 \quad (7-116)$$

for $u = 0, 1, \dots, N-1$, where $N = 2^n$, into Eqs. (7-22) and (7-24). The resulting transformation matrix, denoted \mathbf{A}_H , can be written as a function of the $N \times N$ *Haar matrix*

Do not confuse the Haar matrix with the Hadamard matrix of Section 7.7. Since the same variable is used for both, the proper matrix must be determined from the context of the discussion.

$$\mathbf{H}_N = \begin{bmatrix} h_0(0/N) & h_0(1/N) & \dots & h_0(N-1/N) \\ h_1(0/N) & h_1(1/N) & & \vdots \\ \vdots & & & \ddots \\ h_{N-1}(0/N) & \dots & & h_{N-1}(N-1/N) \end{bmatrix} \quad (7-117)$$

as

$$\mathbf{A}_H = \frac{1}{\sqrt{N}} \mathbf{H}_N \quad (7-118)$$

For example, if $N = 2$,

$$\mathbf{A}_H = \frac{1}{\sqrt{2}} \begin{bmatrix} h_0(0) & h_0(1/2) \\ h_1(0) & h_1(1/2) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (7-119)$$

In the computation of \mathbf{A}_H , x and u of Eq. (7-116) are 0 and 1, so Eqs. (7-114), (7-115), and (7-116) give $s(0,0) = h_0(0)/\sqrt{2} = 1/\sqrt{2}$, $s(1,0) = h_0(0.5)/\sqrt{2} = 1/\sqrt{2}$, $s(0,1) = h_1(0)/\sqrt{2} = 1/\sqrt{2}$, and $s(1,1) = h_1(0.5)/\sqrt{2} = -1/\sqrt{2}$. For $N = 4$, u , q , and p of Eq. (7-114) assume the values

When u is 0, $h_u(x)$ is independent of p and q .

u	p	q
1	0	0
2	1	0
3	1	1

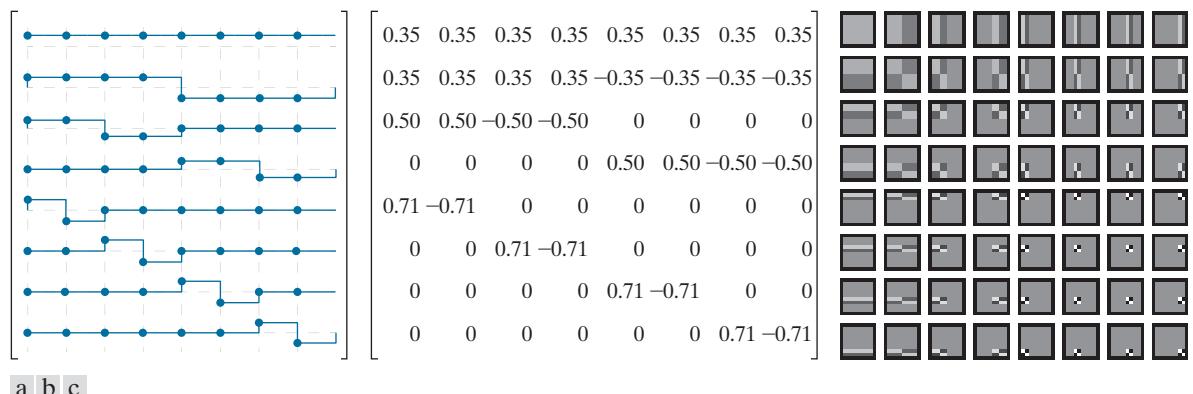


FIGURE 7.18 The transformation matrix and basis images of the discrete Haar transform for $N = 8$. (a) Graphical representation of orthogonal transformation matrix \mathbf{A}_H , (b) \mathbf{A}_H rounded to two decimal places, and (c) basis images. For 1-D transforms, matrix \mathbf{A}_H is used in conjunction with Eqs. (7-28) and (7-29); for 2-D transforms, it is used with Eqs. (7-35) and (7-36).

and the Haar transformation matrix of size 4×4 becomes

$$\mathbf{A}_H = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \quad (7-120)$$

The transformation matrix for $N = 8$ is shown in Fig. 7.18(b). \mathbf{A}_H is real, orthogonal, and sequency ordered. An important property of the Haar transformation matrix is that it can be decomposed into products of matrices with fewer nonzero entries than the original matrix. This is true of all of the transforms we have discussed to this point. They can be implemented in FFT-like algorithms of complexity $O(N \log_2 N)$. The Haar transformation matrix, however, has fewer nonzero entries before the decomposition process begins, making less complex algorithms on the order of $O(N)$ possible. As can be seen in Fig. 7.18(c), the basis images of the separable 2-D Haar transform for images of size 8×8 also have few nonzero entries.

7.10 WAVELET TRANSFORMS

As was noted in Section 7.1, wavelets are small waves with bandpass spectra as defined in Eq. (7-72).

In 1987, *wavelets* were shown to be the foundation of a powerful new approach to signal processing and analysis called *multiresolution* theory (Mallat [1987]). Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including subband coding from signal processing, quadrature mirror filtering from digital speech recognition, and pyramidal image processing. As its name implies, it is concerned with the representation and analysis of signals (or images) at more than one resolution. A *scaling function* is used to create a series of approximations of a function or image, each differing by a factor of 2 in resolution

The discrete wavelet transform, like all transforms considered in this chapter, generates linear expansions of functions with respect to sets of orthonormal or biorthonormal expansion functions.

The coefficients of a 1-D full-scale DWT with respect to Haar wavelets and a 1-D Haar transform are the same.

\mathbf{Z} is the set of integers.

Recall from Section 7.1 that the span of a basis is the set of functions that can be represented as linear combinations of the basis functions.

from its nearest neighboring approximations, and complementary functions, called *wavelets*, are used to encode the differences between adjacent approximations. The *discrete wavelet transform* (DWT) uses those wavelets, together with a single scaling function, to represent a function or image as a linear combination of the wavelets and scaling function. Thus, the wavelets and scaling function serve as an orthonormal or biorthonormal basis of the DWT expansion. The Daubechies and Biorthogonal B-splines of Figs. 7.3(f) and (g) and the Haar basis functions of the previous section are but three of the many bases that can be used in DWTS.

In this section, we present a mathematical framework for the interpretation and application of discrete wavelet transforms. We use the discrete wavelet transform with respect to Haar basis functions to illustrate the concepts introduced. As you proceed through the material, remember that the discrete wavelet transform of a function with respect to Haar basis functions is *not* the Haar transform of the function (although the two are intimately related).

SCALING FUNCTIONS

Consider the set of basis functions composed of all integer translations and binary scalings of the real, square-integrable *father scaling function* $\varphi(x)$ —that is, the set of scaled and translated functions $\{\varphi_{j,k}(x) \mid j, k \in \mathbf{Z}\}$ where

$$\varphi_{j,k}(x) = 2^{j/2} \varphi(2^j x - k) \quad (7-121)$$

In this equation, integer *translation* k determines the position of $\varphi_{j,k}(x)$ along the x -axis and *scale* j determines its shape—i.e., its width and amplitude. If we restrict j to some value, say $j = j_0$, then $\{\varphi_{j_0,k} \mid k \in \mathbf{Z}\}$ is the basis of the function space spanned by the $\varphi_{j,k}(x)$ for $j = j_0$ and $k = \dots, -1, 0, 1, 2, \dots$, denoted V_{j_0} . Increasing j_0 increases the number of representable functions in V_{j_0} , allowing functions with smaller variations and finer detail to be included in the space. As is demonstrated in Fig. 6.19 with Haar scaling functions, this is a consequence of the fact that as j_0 increases, the scaling functions used to represent the functions in V_{j_0} become narrower and separated by smaller changes in x .

EXAMPLE 7.15: The Haar scaling function.

Consider the unit-height, unit-width scaling function

$$\varphi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \quad (7-122)$$

and note it is the Haar basis function $h_0(x)$ from Eq. (7-115). Figure 7.19 shows a few of the pulse-shaped scaling functions that can be generated by substituting Eq. (7-122) into Eq. (7-121). Note when the scale is 1 [i.e., when $j = 1$ as in Figs. 7.19(d) and (e)], the scaling functions are half as wide as when the scale is 0 (i.e., when $j = 0$ as in Figs. 7.19(a) and (b)]. Moreover, for a given interval on x , there are

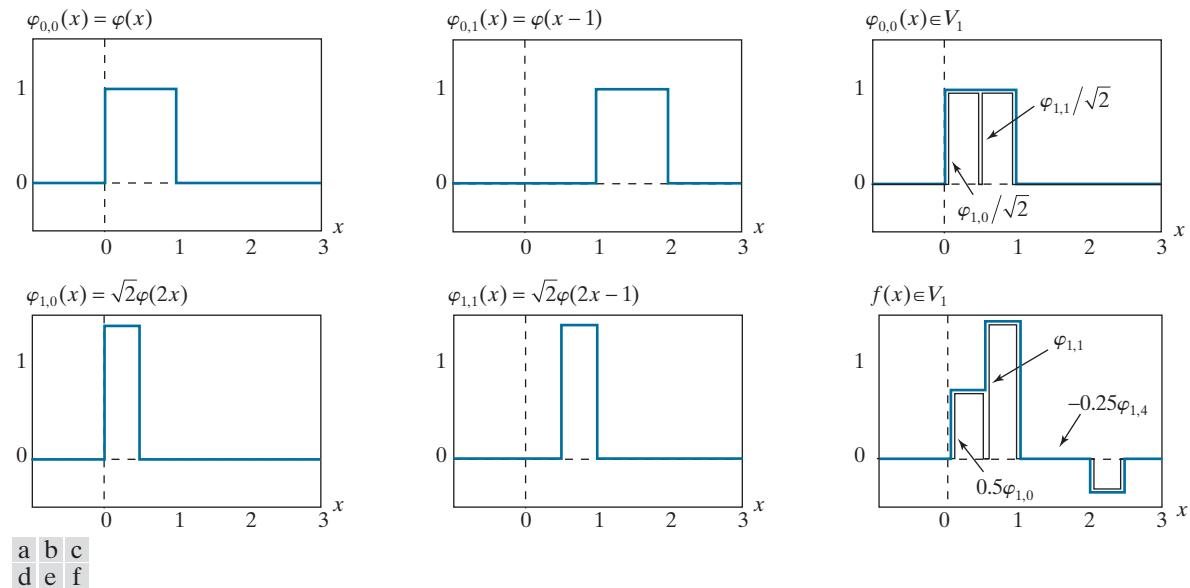


FIGURE 7.19 The Haar scaling function.

twice as many scale 1 as scale 0 scaling functions. For example, two V_1 scaling functions, $\varphi_{1,0}$ and $\varphi_{1,1}$, are located in interval $0 \leq x < 1$, while only one V_0 scaling function, $\varphi_{0,0}$, occupies the same interval.

Figure 7.19(f) shows a member of scaling space V_1 that does not belong in V_0 . The scaling functions in Figs. 7.19(a) and (b) are too coarse to represent it. Higher-resolution functions, like those in Figs. 7.19(d) and (e), are required. They can be used, as is shown in Fig. 7.19(f), to represent the function as the three-term expansion $f(x) = 0.5\varphi_{1,0}(x) + \varphi_{1,1}(x) - 0.25\varphi_{1,4}(x)$. In a similar manner, scaling function $\varphi_{0,0}$, which is both a basis function and member of V_0 , can be represented by a linear combination of V_1 scaling functions [see Fig. 7.19(c)] as follows:

$$\varphi_{0,k}(x) = \frac{1}{\sqrt{2}}\varphi_{1,2k}(x) + \frac{1}{\sqrt{2}}\varphi_{1,2k+1}(x)$$

The Haar scaling function of the preceding example, like the scaling functions of all discrete wavelet transforms, obeys the four fundamental requirements of *multi-resolution analysis* (Mallat [1989a]):

1. The scaling function is orthogonal to its integer translates.
2. The function spaces spanned by the scaling function at low scales are nested within those spanned at higher scales. That is,

$$V_{-\infty} \subset \dots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \dots \subset V_\infty \quad (7-123)$$

where \subset is used to denote “a subspace of.” The scaling functions satisfy the intuitive condition that if $f(x) \in V_j$, then $f(2x) \in V_{j+1}$.

3. The only function representable at every scale is $f(x) = 0$.
4. All measurable, square-integrable functions can be represented as a linear combination of the scaling function as $j \rightarrow \infty$. In other words,

$$V_\infty = L^2(\mathbf{R}) \quad (7-124)$$

Recall that \mathbf{R} is the set of real numbers.

where $L^2(\mathbf{R})$ is the set of measurable, square-integrable, 1-D functions.

Under the above conditions, $\varphi(x)$ can be expressed as a linear combination of double-resolution copies of itself:

$$\varphi(x) = \sum_{k \in \mathbf{Z}} h_\varphi(k) \sqrt{2} \varphi(2x - k) \quad (7-125)$$

Called the *refinement* or *dilation equation*, Eq. (7-125) defines a series expansion in which the expansion functions, in accordance with Eq. (7-121), are scaling functions from one scale higher than $\varphi(x)$ and the $h_\varphi(k)$ are expansion coefficients. The expansion coefficients, which can be collected into an ordered set $\{h_\varphi(k) | k = 0, 1, 2, \dots\} = \{h_\varphi(0), h_\varphi(1), \dots\}$, are commonly called *scaling function coefficients*. For orthonormal scaling functions, it follows from Eqs. (7-51) and (7-52) that

$$h_\varphi(k) = \langle \varphi(x), \sqrt{2} \varphi(2x - k) \rangle \quad (7-126)$$

EXAMPLE 7.16: Haar scaling function coefficients.

The coefficients of the Haar scaling function [i.e., Eq. (7-122)] are $\{h_\varphi(n) | n = 0, 1\} = \{1/\sqrt{2}, 1/\sqrt{2}\}$, the first row of Haar matrix \mathbf{A}_H for $N = 2$ in Eq. (7-119). It is left as an exercise for the reader (see Problem 7.33) to compute these coefficients using Eq. (7-126). Equation (7-125) then yields

$$\begin{aligned} \varphi(x) &= \frac{1}{\sqrt{2}} [\sqrt{2} \varphi(2x)] + \frac{1}{\sqrt{2}} [\sqrt{2} \varphi(2x - 1)] \\ &= \varphi(2x) + \varphi(2x - 1) \end{aligned}$$

This expansion is illustrated graphically in Fig. 7.19(c), where the bracketed terms of the preceding expression are seen to be $\varphi_{1,0}(x)$ and $\varphi_{1,1}(x)$.

WAVELET FUNCTIONS

Given a father scaling function that meets the MRA requirements of the previous section, there exists a *mother wavelet function* $\psi(x)$ whose integer translations and binary scalings,

$$\psi_{j,k}(x) = 2^{j/2} \psi(2^j x - k) \quad (7-127)$$

for all $j, k \in \mathbf{Z}$, span the difference between any two adjacent scaling spaces. If we let W_{j_0} denote the function space spanned by wavelet functions $\{\psi_{j_0,k} | k \in \mathbf{Z}\}$, then

$$V_{j_0+1} = V_{j_0} \oplus W_{j_0} \quad (7-128)$$

Scaling function coefficients can also be combined in a *scaling vector*.

The orthogonal complement of vector space $V \in W$ is the set of vectors in V that are orthogonal to every vector in W .

where \oplus denotes the union of function spaces (like the union of sets). The *orthogonal complement* of V_{j_0} in V_{j_0+1} is W_{j_0} , and the scaling functions that are the basis of V_{j_0} are orthogonal to the wavelet functions that are the basis of W_{j_0} :

$$\langle \varphi_{j_0,k}(x), \psi_{j_0,l}(x) \rangle = 0 \quad \text{for } k \neq l \quad (7-129)$$

Figure 7.20 illustrates graphically the relationship between scaling and wavelet spaces. Each oval in the figure is a scaling space that, in accordance with Eq. (7-123), is nested or contained within the next higher resolution scaling space. The difference between adjacent scaling spaces is a wavelet space. Since wavelet space W_j resides within scaling space V_{j+1} and $\psi_{j,k}(x) \in W_j \subset V_{j+1}$, wavelet function $\psi(x)$ —like its scaling function counterpart in Eq. (7-125)—can be written as a weighted sum of shifted, double-resolution scaling functions. That is, we can write

$$\psi(x) = \sum_k h_\psi(k) \sqrt{2} \varphi(2x - k) \quad (7-130)$$

Wavelet function coefficients can also be combined in a *wavelet vector*.

where the $h_\psi(k)$ coefficients, called *wavelet function coefficients*, can be combined into the ordered set $\{h_\psi(k) | k = 0, 1, 2, \dots\} = \{h_\psi(0), h_\psi(1), \dots\}$. Since integer wavelet translates are orthogonal to one another and to their complementary scaling functions, it can be shown (see, for example, Burrus, Gopinath, and Guo [1998]) that the $h_\psi(k)$ of Eq. (7-130) are related to the $h_\varphi(k)$ of Eq. (7-125) by

$$h_\psi(k) = (-1)^k h_\varphi(1-k) \quad (7-131)$$

EXAMPLE 7.17: The Haar wavelet function and coefficients.

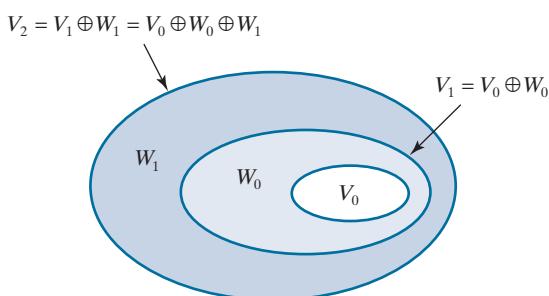
In the previous example, the Haar scaling coefficients were defined as $\{h_\varphi(n) | n = 0, 1\} = \{1/\sqrt{2}, 1/\sqrt{2}\}$. Using Eq. (7-131), the corresponding wavelet function coefficients are

$$\begin{aligned} h_\psi(0) &= (-1)^0 h_\varphi(1-0) = 1/\sqrt{2} \\ h_\psi(1) &= (-1)^1 h_\varphi(1-1) = -1/\sqrt{2} \end{aligned}$$

so $\{h_\psi(n) | n = 0, 1\} = \{1/\sqrt{2}, -1/\sqrt{2}\}$. These coefficients correspond to the second row of matrix \mathbf{A}_H for

FIGURE 7.20

The relationship between scaling and wavelet function spaces.



$N = 2$ in Eq. (7-119). Substituting these values into Eq. (7-130), we get $\psi(x) = \varphi(2x) - \varphi(2x - 1)$, which is plotted in Fig. 7.21(a). Thus, the Haar *mother wavelet function* is

$$\psi(x) = \begin{cases} 1 & 0 \leq x < 0.5 \\ -1 & 0.5 \leq x < 1 \\ 0 & \text{elsewhere} \end{cases} \quad (7-132)$$

Note it is also the Haar basis function $h_1(x)$ of Eq. (7-115). Using Eq. (7-127), we can now generate the universe of scaled and translated Haar wavelets. Two such wavelets, $\psi_{0,2}(x)$ and $\psi_{1,0}(x)$, are plotted in Figs. 7.21(b) and (c), respectively. Note wavelet $\psi_{1,0}(x) \in W_1$ is narrower than $\psi_{0,2}(x) \in W_0$ and as such can be used to represent functions of finer detail.

Figure 7.21(d) shows a member of function space V_1 that is not in V_0 . This function was considered in Example 7.15 [see Fig. 7.19(f)]. Although the function cannot be represented accurately in V_0 , Eq. (7-128) indicates that it can be written as a function of V_0 and W_0 scaling and wavelet functions. The resulting expansion is

$$f(x) = f_a(x) + f_d(x)$$

where

$$f_a(x) = \frac{3\sqrt{2}}{4} \varphi_{0,0}(x) - \frac{\sqrt{2}}{8} \varphi_{0,2}(x)$$

and

$$f_d(x) = \frac{-\sqrt{2}}{4} \psi_{0,0}(x) - \frac{\sqrt{2}}{8} \psi_{0,2}(x)$$

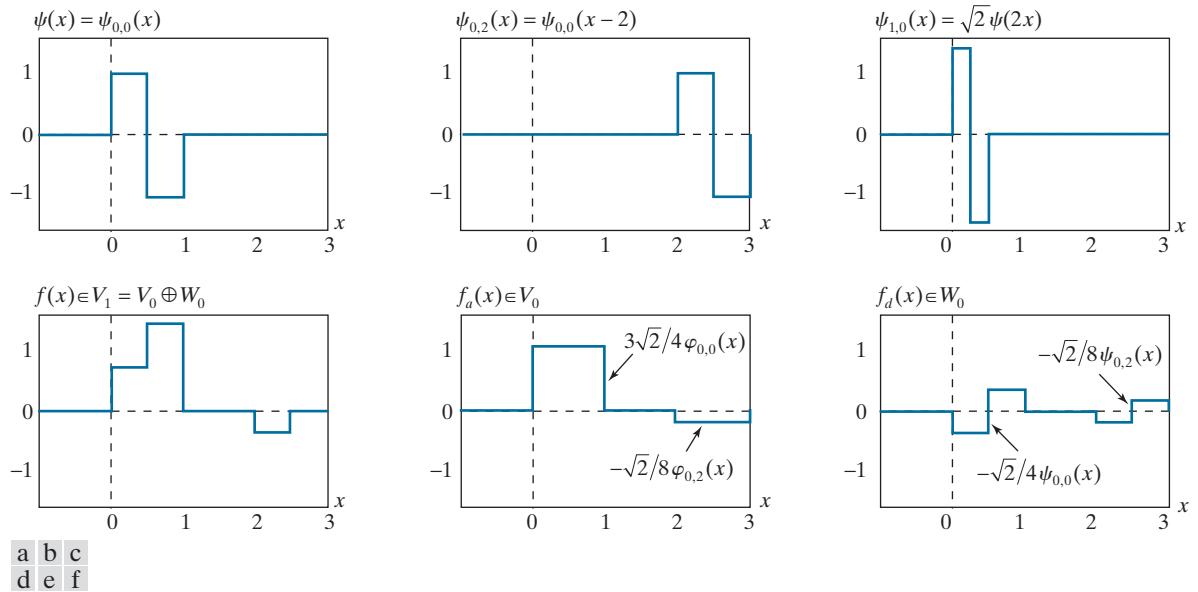


FIGURE 7.21 Haar wavelet functions.

Here, $f_a(x)$ is an approximation of $f(x)$ using V_0 scaling functions, while $f_d(x)$ is difference $f(x) - f_a(x)$ as a sum of W_0 wavelets. These approximations and differences, which are shown in Figs. 7.19(e) and (f), divide $f(x)$ in a manner similar to lowpass and highpass filtering. The low frequencies of $f(x)$ are captured in $f_a(x)$ —it assumes the average value of $f(x)$ in each integer interval—while the higher-frequency details are encoded in $f_d(x)$.

WAVELET SERIES EXPANSION

Combining Eqs. (7-124) and (7-128), the space of all measurable, square-integrable functions can be defined as $L^2(\mathbf{R}) = V_{j_0} \oplus W_{j_0} \oplus W_{j_0+1} \oplus \dots$, where j_0 is an arbitrary starting scale. We can then define the *wavelet series expansion* of function $f(x) \in L^2(\mathbf{R})$ with respect to wavelet $\psi(x)$ and scaling function $\varphi(x)$ as

$$f(x) = \sum_k c_{j_0}(k) \varphi_{j_0,k}(x) + \sum_{j=j_0}^{\infty} \sum_k d_j(k) \psi_{j,k}(x) \quad (7-133)$$

where c_{j_0} and d_j for $j \geq j_0$ are called *approximation* and *detail coefficients*, respectively. Any measurable, square-integrable, 1-D function can be expressed as a weighted sum of V_{j_0} scaling functions and W_j wavelets for $j \geq j_0$. The first sum in Eq. (7-133) produces an approximation of $f(x)$ from scale j_0 scaling functions; each successive scale of the second sum provides increasing detail as a sum of higher-resolution wavelets. If the scaling and wavlet functions are orthonormal,

$$c_{j_0} = \langle f(x), \varphi_{j_0,k}(x) \rangle \quad (7-134)$$

and

$$d_j = \langle f(x), \psi_{j,k}(x) \rangle \quad (7-135)$$

Here, we have used Eq. (7-13). If they are part of a biorthogonal basis, the φ and ψ terms must be replaced by their dual functions, $\tilde{\varphi}$ and $\tilde{\psi}$, respectively.

EXAMPLE 7.18: The Haar wavelet series expansion of $y = x^2$.

Consider the simple function

$$y = \begin{cases} x^2 & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

shown in Fig. 7.22(a). Using Haar wavelets—see Eqs. (7-122) and (7-132)—and starting scale $j_0 = 0$, Eqs. (7-134) and (7-135) can be used to compute the following expansion coefficients:

$$c_0(0) = \int_0^1 x^2 \varphi_{0,0}(x) dx = \int_0^1 x^2 dx = \frac{x^3}{3} \Big|_0^1 = \frac{1}{3}$$

$$d_0(0) = \int_0^1 x^2 \psi_{0,0}(x) dx = \int_0^{0.5} x^2 dx - \int_{0.5}^1 x^2 dx = -\frac{1}{4}$$

$$d_1(0) = \int_0^1 x^2 \psi_{1,0}(x) dx = \int_0^{0.25} x^2 \sqrt{2} dx - \int_{0.25}^{0.5} x^2 \sqrt{2} dx = -\frac{\sqrt{2}}{32}$$

$$d_1(1) = \int_0^1 x^2 \psi_{1,1}(x) dx = \int_{0.5}^{0.75} x^2 \sqrt{2} dx - \int_{0.75}^1 x^2 \sqrt{2} dx = -\frac{3\sqrt{2}}{32}$$

Substituting these values into Eq. (7-133), we get the wavelet series expansion

$$y = \underbrace{\frac{1}{3} \varphi_{0,0}(x)}_{V_0} + \underbrace{\left[-\frac{1}{4} \psi_{0,0}(x) \right]}_{W_0} + \underbrace{\left[-\frac{\sqrt{2}}{32} \psi_{1,0}(x) - \frac{3\sqrt{2}}{32} \psi_{1,1}(x) \right]}_{W_1} + \dots$$

$$\underbrace{V_1 = V_0 \oplus W_0}_{V_2 = V_1 \oplus W_1 = V_0 \oplus W_0 \oplus W_1}$$

The first term in this expansion employs $c_0(0)$ to generate a V_0 approximation of the function being expanded. This approximation is shown in Fig. 7.22(b) and is the average value of the original function. The second term uses $d_0(0)$ to refine the approximation by adding a level of detail from wavelet space W_0 . The added detail and resulting V_1 approximation are shown in Figs. 7.22(c) and (d), respectively. Another level of detail is formed from the products of $d_1(0)$ and $d_1(1)$ with the corresponding wavelets of W_1 . This additional detail is shown in Fig. 7.22(e), and the resulting V_2 approximation is depicted in Fig. 7.22(f). Note the expansion is now beginning to resemble the original function. As higher scales (greater levels of detail) are added, the approximation becomes a more precise representation of the function, realizing it in the limit as $j \rightarrow \infty$.

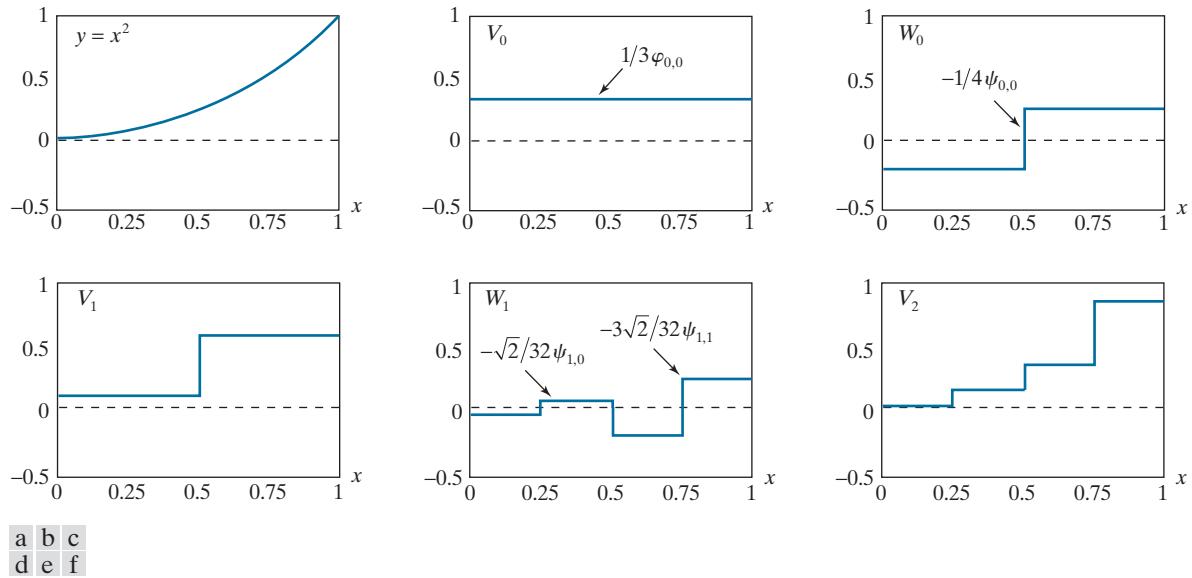


FIGURE 7.22 A wavelet series expansion of $y = x^2$ using Haar wavelets.

DISCRETE WAVELET TRANSFORM IN ONE DIMENSION

Like a Fourier series expansion, the wavelet series expansion of the previous section maps a function of a single continuous variable into a sequence of discrete coefficients. If the function being expanded is discrete, the coefficients of the expansion are its *discrete wavelet transform* (DWT) and the expansion itself is the function's *inverse discrete wavelet transform*. Letting $j_0 = 0$ in Eqs. (7-133) through (7-135) and restricting attention to N -point discrete functions in which N is a power of 2 (i.e., $N = 2^J$), we get

Remember that for discrete inputs, x is a discrete variable that takes on integer values between 0 and $N - 1$.

$$f(x) = \frac{1}{\sqrt{N}} \left[T_\varphi(0,0)\varphi(x) + \sum_{j=0}^{J-1} \sum_{k=0}^{2^j-1} T_\psi(j,k)\psi_{j,k}(x) \right] \quad (7-136)$$

where

$$T_\varphi(0,0) = \langle f(x), \varphi_{0,0}(x) \rangle = \langle f(x), \varphi(x) \rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x)\varphi^*(x) \quad (7-137)$$

and

$$T_\psi(j,k) = \langle f(x), \psi_{j,k}(x) \rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x)\psi_{j,k}^*(x) \quad (7-138)$$

for $j = 0, 1, \dots, J-1$ and $k = 0, 1, \dots, 2^j - 1$. The transform coefficients defined by Eqs. (7-137) and (7-138) are called *approximation* and *detail coefficients*, respectively. They correspond to the $c_{j_0}(k)$ and $d_j(k)$ of the wavelet series expansion in the previous section. Note the integrations of the series expansion have been replaced by summations in Eqs. (7-137) through (7-138). In the discrete case, inner products like those of Eqs. (7-1) and (7-2), as opposed to Eq. (7-3), are used. In addition, a $1/\sqrt{N}$ normalizing factor, reminiscent of the DFT in Example 7.6, has been added to both the forward and inverse transforms. This factor alternately could be incorporated into the forward or inverse alone as $1/N$. Finally, it should be remembered that Eqs. (7-137) through (7-138) are valid for orthonormal bases. If the scaling and wavelet functions are real-valued, the conjugations can be dropped. If the basis is biorthogonal, the φ and ψ terms in Eqs. (7-137) and (7-138) must be replaced by their duals, $\tilde{\varphi}$ and $\tilde{\psi}$, respectively.

EXAMPLE 7.19: A 1-D discrete wavelet transform.

To illustrate the use of Eqs. (7-137) through (7-138), consider a discrete function of four points in which $f(0) = 1$, $f(1) = 4$, $f(2) = -3$, and $f(3) = 0$. Since $N = 4$, J is 2 and the summations in Eqs. (7-136) through (7-138) are performed for $x = 0, 1, 2, 3$. When j is 0, k is 0; when j is 1, k is 0 or 1. If we use Haar scaling and wavelet functions and assume the four samples of $f(x)$ are distributed over the support of the scaling function, which is 1, Eq. (7-137) gives

$$T_\varphi(0,0) = \frac{1}{2} \sum_{x=0}^3 f(x)\varphi(x) = \frac{1}{2} [(1)(1) + (4)(1) + (-3)(1) + (0)(1)] = 1$$

Note we have employed uniformly spaced samples of the Haar scaling function for $j = k = 0$ —i.e., $\varphi(x) = 1$ for $x = 0, 1, 2, 3$. The sampled values match the elements of the first row of Haar transformation matrix \mathbf{A}_H in Eq. (7-120) of Section 7.9. Using Eq. (7-138) and similarly spaced samples of $\psi_{j,k}(x)$, which are the elements of rows 2, 3, and 4 of \mathbf{A}_H , we get

$$\begin{aligned}T_\psi(0,0) &= \frac{1}{2}[(1)(1) + (4)(1) + (-3)(-1) + (0)(-1)] = 4 \\T_\psi(1,0) &= \frac{1}{2}[(1)(\sqrt{2}) + (4)(-\sqrt{2}) + (-3)(0) + (0)(0)] = -1.5\sqrt{2} \\T_\psi(1,1) &= \frac{1}{2}[(1)(0) + (4)(0) + (-3)(\sqrt{2}) + (0)(-\sqrt{2})] = -1.5\sqrt{2}\end{aligned}$$

Thus, the discrete wavelet transform of our simple four-sample function relative to Haar scaling and wavelet functions is $\{1, 4, -1.5\sqrt{2}, -1.5\sqrt{2}\}$. Since the transform coefficients are a function of two variables—scale j and translation k —we combine them into an *ordered set*. The elements of this set turn out to be identical to the elements of the sequency-ordered Haar transform of the function:

$$\mathbf{t}_H = \mathbf{A}_H \mathbf{f} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ 4 \\ -3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ -1.5\sqrt{2} \\ -1.5\sqrt{2} \end{bmatrix}$$

Recall from the previous section that Haar transforms are a function of a single transform domain variable, denoted u .

Equation (7-136) enables the reconstruction of the original function from its wavelet transform coefficients. Expanding the summation, we get

$$f(x) = \frac{1}{2} [T_\varphi(0,0)\varphi(x) + T_\psi(0,0)\psi_{0,0}(x) + T_\psi(1,0)\psi_{1,0}(x) + T_\psi(1,1)\psi_{1,1}(x)]$$

for $x = 0, 1, 2, 3$. If $x = 0$, for instance,

$$f(0) = \frac{1}{2} [(1)(1) + (4)(1) + (-1.5\sqrt{2})(\sqrt{2}) + (-1.5\sqrt{2})(0)] = 1$$

As in the forward case, uniformly spaced samples of the scaling and wavelet functions are used in the computation of the inverse.

The Fast Wavelet Transform

The multiresolution refinement equation and its wavelet counterpart, Eqs. (7-125) and (7-130), make it possible to define the scaling and wavelet functions at any scale as a function of shifted, double-resolution copies of the scaling functions at the next higher scale. In the same way, the expansion coefficients of the wavelet series expan-

sion and discrete wavelet transform can be computed recursively (see Problem 7.35) using

$$c_j(k) = \sum_n h_\varphi(n-2k)c_{j+1}(n) \quad (7-139)$$

$$d_j(k) = \sum_n h_\psi(n-2k)c_{j+1}(n) \quad (7-140)$$

and

$$T_\varphi(j, k) = \sum_n h_\varphi(n-2k)T_\varphi(j+1, n) \quad (7-141)$$

$$T_\psi(j, k) = \sum_n h_\psi(n-2k)T_\varphi(j+1, n) \quad (7-142)$$

respectively. In contrast to Eqs. (7-133) and (7-136), where the only scaling coefficients that are needed in the computations are at scale j_0 , Eqs. (7-139) through (7-142) require the computation of all scaling coefficients up to the highest scale of interest. Comparing these equations to the equation defining discrete convolution [i.e., Eq. (4-48)], we see that n is a dummy variable of convolution and the remaining minus signs and $2k$ terms reverse the order of the h_φ and h_ψ coefficients and sample the convolution results at $n = 0, 2, 4, \dots$, respectively. Thus, for the discrete wavelet transform, we can rewrite Eqs. (7-141) and (7-142) as

$$T_\varphi(j, k) = T_\varphi(j+1, n) \star h_\varphi(-n) \quad (7-143)$$

$$T_\psi(j, k) = T_\varphi(j+1, n) \star h_\psi(-n) \quad (7-144)$$

where the convolutions are evaluated at instants $n = 0, 2, \dots, 2^{j+1} - 2$. As indicated in Fig. 7.23, evaluating convolutions at nonnegative, even indices is equivalent to filtering and *downsampling* by 2 (i.e., discarding every other convolved value). For a 1-D sequence of samples $y(n)$ for $n = 0, 1, 2, \dots$, downsampled sequence $y_{2\downarrow}(n)$ is defined as

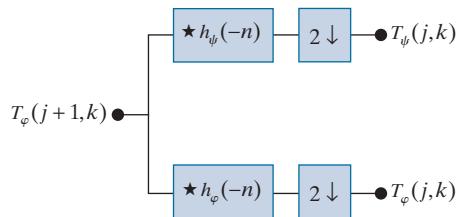
$$y_{2\downarrow}(n) = y(2n) \quad \text{for } n = 0, 1, \dots \quad (7-145)$$

Equations (7-143) and (7-144) are the defining equations of a computationally efficient form of the DWT called the *fast wavelet transform* (FWT). For an input sequence of length $N = 2^J$, the number of mathematical operations involved is on

Recall from Section 3.4 that the use of correlation or convolution in spatial filtering is a matter of personal preference.

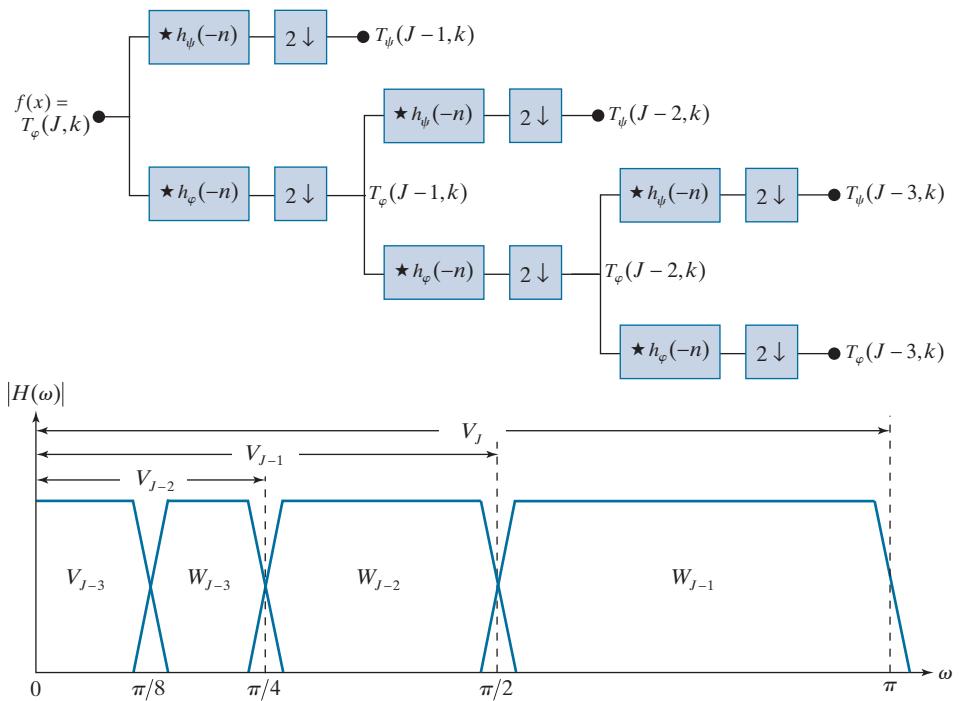
FIGURE 7.23

A FWT analysis filter bank for orthonormal filters. The \star and $2\downarrow$ denote convolution and downsampling by 2, respectively.



a
b**FIGURE 7.24**

(a) A three-stage or three-scale FWT analysis filter bank and (b) its frequency-splitting characteristics. Because of symmetry in the DFT of the filter's impulse response, it is common to display only the $[0, \pi]$ region.



the order of $O(N)$. That is, the number of multiplications and additions is linear with respect to the length of the input sequence—because the number of multiplications and additions involved in the convolutions performed by the FWT filter bank in Fig. 7.23 is proportional to the length of the sequences being convolved. Thus, the FWT compares favorably with the FFT algorithm, which requires on the order of $O(N \log_2 N)$ operations.

A P -scale FWT employs P filter banks to generate a P -scale transform at scales $J - 1, J - 2, \dots, J - P$, where $P \leq J$.

Figure 7.24(a) shows a three-scale filter bank in which the FWT *analysis filter* of Fig. 7.23 has been “iterated” three times to create a three-stage structure for computing transform coefficients at scales $J - 1, J - 2$, and $J - 3$. Note the highest scale coefficients are assumed to be samples of the function itself.[†] Otherwise, the approximation and detail coefficients at scale j are computed by convolving $T_\varphi(j+1, k)$, the scale $j+1$ approximation coefficients, with the order-reversed scaling and wavelet coefficients, $h_\varphi(-n)$ and $h_\psi(-n)$, and subsampling the results. If there are K scaling and wavelet function coefficients, the order reversed scaling and wavelet coefficients are $\{h_\varphi(K-1-m) | m = 0, 1, \dots, K-1\}$ and $\{h_\psi(K-1-m) | m = 0, 1, \dots, K-1\}$, respectively. For a discrete input of length $N = 2^J$, the filter bank in Fig. 7.23 can

[†]If function $f(x)$ is sampled above the Nyquist rate, as is usually the case, its samples are good approximations of the scaling coefficients at the sampling resolution and can be used as the starting high-resolution scaling coefficient inputs. In other words, no wavelet or detail coefficients are needed at the sampling scale. The highest-resolution scaling functions act as unit discrete impulse functions in Eqs. (7-141) and (7-142), allowing $f(x)$ to be used as the scaling (approximation) input to the first two-band filter bank (Odegard, Gopinath, and Burrus [1992]).

be iterated up to J times. In operation, the leftmost filter bank of Fig. 7.24(a) splits the input function into a lowpass *approximation* component that corresponds to scaling coefficients $T_\varphi(J-1, k)$ and a highpass *detail* component corresponding to coefficients $T_\psi(J-1, k)$. This is illustrated graphically in Fig. 7.24(b), where scaling space V_J is split into wavelet space W_{J-1} and scaling space V_{J-1} . The spectrum of the original function is split into two half-band components. The second filter bank in Fig. 7.24(a) splits the spectrum of scaling space V_{J-1} , the lower half-band of the first filter bank, into quarter-band spaces W_{J-2} and V_{J-2} and corresponding FWT coefficients $T_\psi(J-2, k)$ and $T_\varphi(J-2, k)$, respectively. Finally, the third filter bank generates eighth-band spaces W_{J-3} and V_{J-3} with FWT coefficients $T_\psi(J-3, k)$ and $T_\varphi(J-3, k)$. As was noted in connection with Eq. (7-73) of Section 7.4 and demonstrated in Fig. 7.5, as the scale of the wavelet functions increases, the spectra of the wavelets are stretched (i.e., their bandwidth is doubled and shifted higher by a factor of two). In Fig. 7.24(b), this is evidenced by the fact that the bandwidth of W_{J-1} is $\pi/2$, while the bandwidths of W_{J-2} and W_{J-3} are $\pi/4$ and $\pi/8$, respectively. For higher-scale transforms, the spectra of the wavelets would continue to decrease in bandwidth, but would never reach radian frequency $\omega = 0$. A lowpass scaling function is always needed to capture the frequencies around DC.

EXAMPLE 7.20: Computing a 1-D fast wavelet transform.

To illustrate the preceding concepts, consider the discrete function $f(x) = \{1, 4, -3, 0\}$ from Example 7.19. As in that example, we will compute its wavelet transform with respect to Haar scaling and wavelet functions. Here, however, we will not use the Haar basis functions directly. Instead, we will use the corresponding scaling and wavelet coefficients from Examples 7.16 and 7.17:

$$\{h_\varphi(n) | n = 0, 1\} = \{1/\sqrt{2}, 1/\sqrt{2}\} \quad (7-146)$$

and

$$\{h_\psi(n) | n = 0, 1\} = \{1/\sqrt{2}, -1/\sqrt{2}\} \quad (7-147)$$

Since the transform computed in Example 7.19 was the ordered set $\{T_\varphi(0, 0), T_\psi(0, 0), T_\psi(1, 0), T_\psi(1, 1)\}$, we will compute the corresponding two-scale FWT for scales $j = \{0, 1\}$. Recall from the previous example that $k = 0$ when $j = 0$, while k is 0 and 1 when $j = 1$. The transform will be computed using a two-stage filter bank that parallels the three-stage filter bank of Fig. 7.24(a). Figure 7.25 shows the resulting filter bank and the sequences that follow from the required FWT convolutions and downsamplings. Note input function $f(x)$ serves as the scaling (or approximation) input to the left most filter bank. To compute the $T_\psi(1, n)$ coefficients that appear at the end of the upper branch of Fig. 7.25, we first convolve $f(x)$ with $h_\psi(-n)$. For Haar scaling and wavelet coefficients, $K = 2$ and the order reversed wavelet coefficients are $\{h_\psi(K-1-m) | m = 0, 1, \dots, K-1\} = \{h_\psi(1-m) | m = 0, 1\} = \{-1/\sqrt{2}, 1/\sqrt{2}\}$. As explained in Section 3.4, convolution requires flipping one of the convolved functions about the origin, sliding it past the other, and computing the sum of the point-wise product of the two functions. Flipping order-reversed wavelet coefficients $\{-1/\sqrt{2}, 1/\sqrt{2}\}$ to get $\{1/\sqrt{2}, -1/\sqrt{2}\}$ and sliding them from left-to-right across input sequence $\{1, 4, -3, 0\}$, we get

$$\{-1/\sqrt{2}, -3/\sqrt{2}, 7/\sqrt{2}, -3/\sqrt{2}, 0\}$$

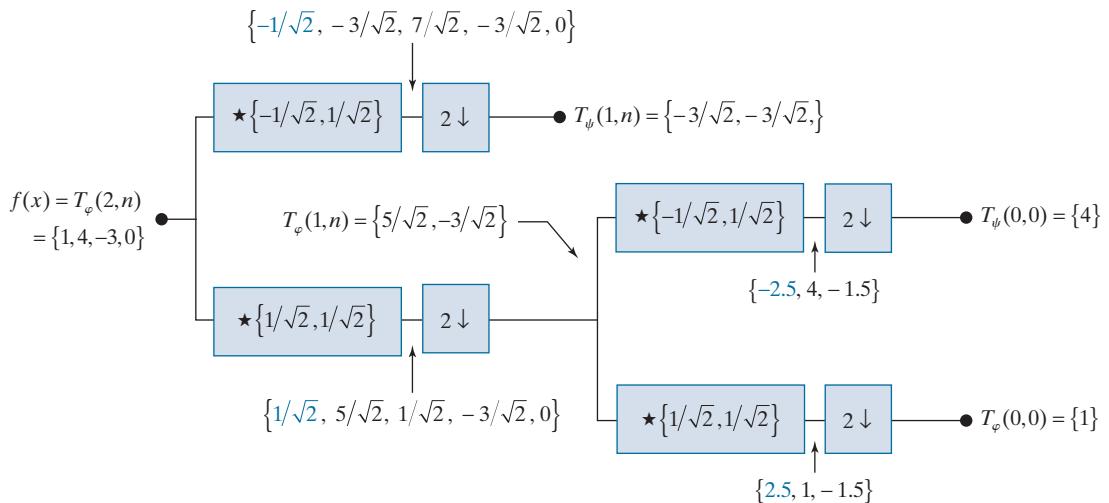


FIGURE 7.25 Computing a two-scale fast wavelet transform of sequence $\{1, 4, -3, 0\}$ using Haar scaling and wavelet coefficients.

where the first term corresponds to convolution index $n = -1$. In Fig. 7.25, convolution values that are associated with a negative dummy variable of convolution (i.e., $n < 0$) are denoted in blue. Since scale $j = 1$, the downsampled convolutions correspond to the even indices of n up to $2^{j+1} - 2$. Thus, $n = 0$ and 2 and $T_\psi(1, n) = \{-3/\sqrt{2}, -3/\sqrt{2}\}$. The remaining convolutions and downsamplings are performed in a similar manner.

The blocks containing a \star in Figs. 7.23 through 7.25 are FIR filters. FIR filters are also discussed in Section 4.7.

Note we use $h(n)$ for analysis or decomposition filters, which include one scaling filter and one wavelet filter, and $g(n)$ for synthesis or reconstruction filters, which also include a scaling and wavelet filter. The scaling filters are sometimes called approximation or lowpass filters and have a subscript of 0 in Fig. 7.26, while the wavelet filters are called detail or high-pass filters and have a subscript of 1.

In digital signal processing (DSP), filters like those in Figs. 7.23 through 7.25 are known as *finite impulse response* (FIR) filters. Their response to a unit impulse is a finite sequence of outputs that assumes the values of the filter coefficients. Figure 7.26(a) shows one well-known arrangement of real-coefficient, FIR filters that has been studied extensively in the literature. Called a two-band *subband coding* and *decoding* system, it is composed of two *analysis filters*, $h_0(n)$ and $h_1(n)$, and two *synthesis filters*, $g_0(n)$ and $g_1(n)$. The analysis filters decompose the input into two half-length sequences $f_0(n)$ and $f_1(n)$. As can be seen in Fig. 7.26(a), filter $h_0(n)$ is a lowpass filter whose output is an approximation of $f(x)$; filter $h_1(n)$ is a high-pass filter whose output is the difference between the lowpass approximation and $f(x)$. As Fig. 7.26(b) shows, the spectrum of the input sequence is split into two half-bands, $H_0(\omega)$ and $H_1(\omega)$. Synthesis bank filters $g_0(n)$ and $g_1(n)$ are then used to reconstruct $\hat{f}(x)$ from *upsampled* versions of $f_0(n)$ and $f_1(n)$. For a 1-D sequence of samples $y(n)$, upsampled sequence $y_{2\uparrow}(n)$ can be defined as

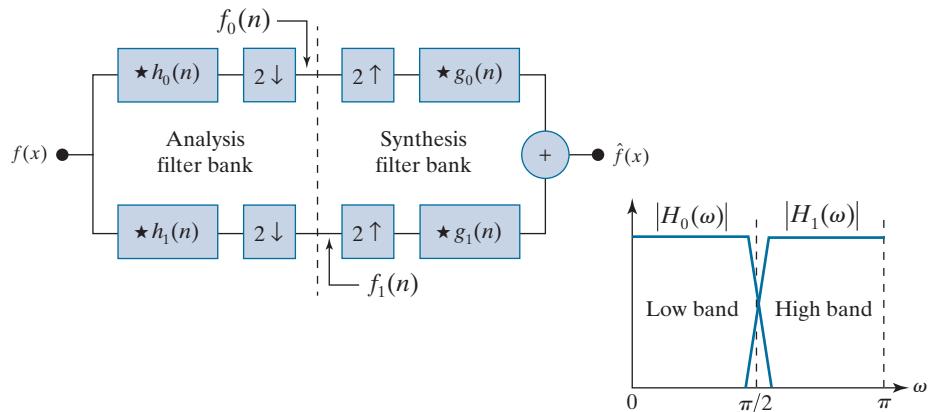
$$y_{2\uparrow}(n) = \begin{cases} y(n/2) & \text{if } n \text{ is even} \\ 0 & \text{otherwise} \end{cases} \quad (7-148)$$

where the upsampling is by a factor of 2. Upsampling by a factor of 2 can be thought of as inserting a 0 after every sample of $y(n)$.

a b

FIGURE 7.26

(a) A two-band digital filtering system for subband coding and decoding and (b) its spectrum-splitting properties.



The goal in subband coding is to choose the analysis and synthesis filters so $\hat{f}(x) = f(x)$. When this is accomplished, the system is said to employ *perfect reconstruction filters* and the filters are, up to some constant factors, related as follows:

Equations (7-149) and (7-151) are described in detail in the filter bank literature (see, for example, Vetterli and Kovacevic [1995]). For many biorthogonal filters, g_0 and g_1 are different in length, requiring the shorter filter to be zero-padded. In *causal* filters, $n \geq 0$ and the output depends only on current and past inputs.

and

$$g_0(n) = (-1)^n h_1(n) \quad (7-149)$$

$$g_1(n) = (-1)^n h_0(n) \quad (7-150)$$

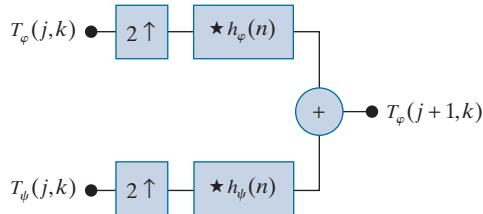
In these equations, $(-1)^n$ changes the signs of the odd-indexed analysis filter coefficients and is called *modulation*. Each synthesis filter is a modulated version of the analysis filter that opposes it diagonally in Fig. 7.26(a). Thus, the analysis and synthesis filters are said to be *cross-modulated*. Their impulse responses are biorthogonal. If they are also orthonormal and of length K , where K is divisible by 2, they satisfy the additional constraints that

$$\begin{aligned} g_1(n) &= (-1)^n g_0(K-1-n) \\ h_0(n) &= g_0(K-1-n) \\ h_1(n) &= g_1(K-1-n) \end{aligned} \quad (7-151)$$

Noting the similarity between the FWT analysis filter bank in Fig. 7.23 and the subband analysis filter bank of Fig. 7.26(a), we can postulate the inverse FWT *synthesis filter bank* of Fig. 7.27. For the case of orthonormal filters, Eq. (7-151) constrains the synthesis filters to be order-reversed versions of the analysis filters. Comparing the filters in Figs. 7.23 and 7.27, we see this is indeed the case. It must be remembered, however, that perfect reconstruction is also possible with biorthogonal analysis and synthesis filters, which are not order-reversed versions of one another. Biorthogonal analysis and synthesis filters are cross-modulated in accordance with Eqs. (7-149) and (7-150). Finally, we note the inverse filter bank of Fig. 7.27, like the forward FWT

FIGURE 7.27

An inverse FWT synthesis filter bank for orthonormal filters.



filter bank of Fig. 6.23, can be iterated for the computation of multiscale inverse FWTs. In the next example, a two-scale inverse FWT structure is considered. The coefficient combining process demonstrated there can be extended to any number of scales.

EXAMPLE 7.21: Computing a 1-D inverse fast wavelet transform.

Computation of the inverse fast wavelet transform mirrors its forward counterpart. Figure 7.28 illustrates the process for the sequence considered in Example 7.20. To begin the calculation, the level 0 approximation and detail coefficients are upsampled to yield $\{1, 0\}$ and $\{4, 0\}$, respectively. Convolution with filters $h_\varphi(n) = \{1/\sqrt{2}, 1/\sqrt{2}\}$ and $h_\psi(n) = \{1/\sqrt{2}, -1/\sqrt{2}\}$ produces $\{1/\sqrt{2}, 1/\sqrt{2}, 0\}$ and $\{4/\sqrt{2}, -4/\sqrt{2}, 0\}$, which when added give $T_\varphi(1, n) = \{5/\sqrt{2}, -3/\sqrt{2}\}$. Thus, the level 1 approximation of Fig. 7.28, which matches the computed approximation in Fig. 7.25, is reconstructed. Continuing in this manner, $f(x)$ is formed at the right of the second synthesis filter bank.

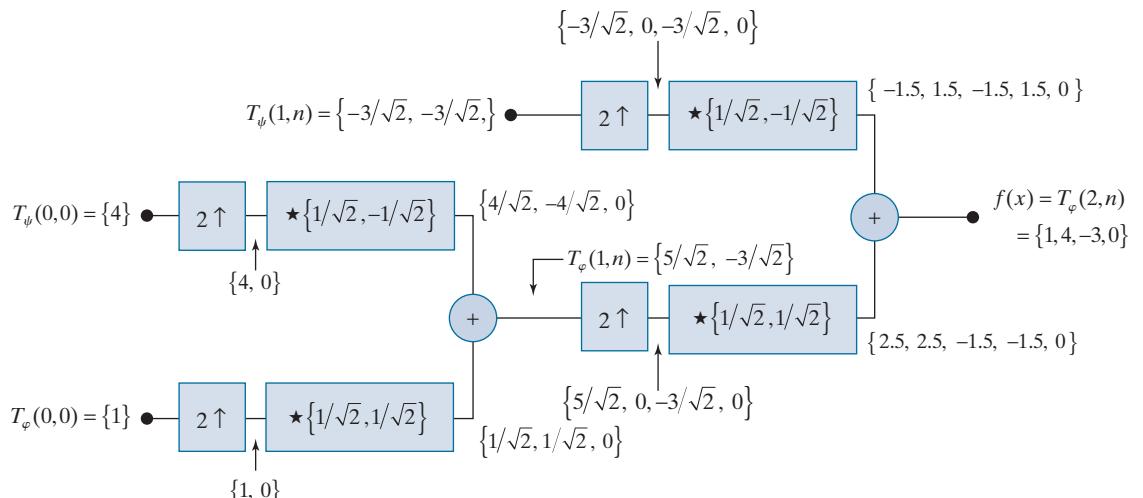


FIGURE 7.28 Computing a two-scale inverse fast wavelet transform of sequence $\{1, 4, -1.5\sqrt{2}, -1.5\sqrt{2}\}$ with Haar scaling and wavelet functions.

WAVELET TRANSFORMS IN TWO DIMENSIONS

The 1-D wavelet transform of the previous section is easily extended to 2-D functions such as images. In two dimensions, a two-dimensional scaling function, $\varphi(x, y)$, and three 2-D wavelets, $\psi^H(x, y)$, $\psi^V(x, y)$, and $\psi^D(x, y)$, are required. Each is the product of two 1-D functions. Excluding products that produce 1-D results, like $\varphi(x)\psi(x)$, the four remaining products produce the *separable* scaling function

$$1 \quad \int_{-\infty}^{\infty} \cdot \quad (7-152)$$

and separable, “directionally sensitive” wavelets

$$\psi^H(x, y) = \psi(x)\varphi(y) \quad (7-153)$$

$$\psi^V(x, y) = \varphi(x)\psi(y) \quad (7-154)$$

$$\psi^D(x, y) = \psi(x)\psi(y) \quad (7-155)$$

These wavelets measure functional variations—intensity changes in images—along different directions: ψ^H measures variations along columns (for example, horizontal edges), ψ^V responds to variations along rows (like vertical edges), and ψ^D corresponds to variations along diagonals. The directional sensitivity is a natural consequence of the separability in Eqs. (7-153) to (7-155); it does not increase the computational complexity of the 2-D transform discussed in this section.

Like the 1-D discrete wavelet transform, the 2-D DWT can be implemented using digital filters and downsamplers. With separable 2-D scaling and wavelet functions, we simply take the 1-D FWT of the rows of $f(x, y)$, followed by the 1-D FWT of the resulting columns. Figure 7.29(a) shows the process in block diagram form. Note, like its 1-D counterpart in Fig. 7.23, the 2-D FWT “filters” the scale $j+1$ approximation coefficients, denoted $T_\varphi(j+1, k, l)$ in the figure, to construct the scale j approximation and detail coefficients. In the 2-D case, however, we get three sets of detail coefficients—*horizontal details* $T_\psi^H(j, k, l)$, *vertical details* $T_\psi^V(j, k, l)$, and *diagonal details* $T_\psi^D(j, k, l)$.

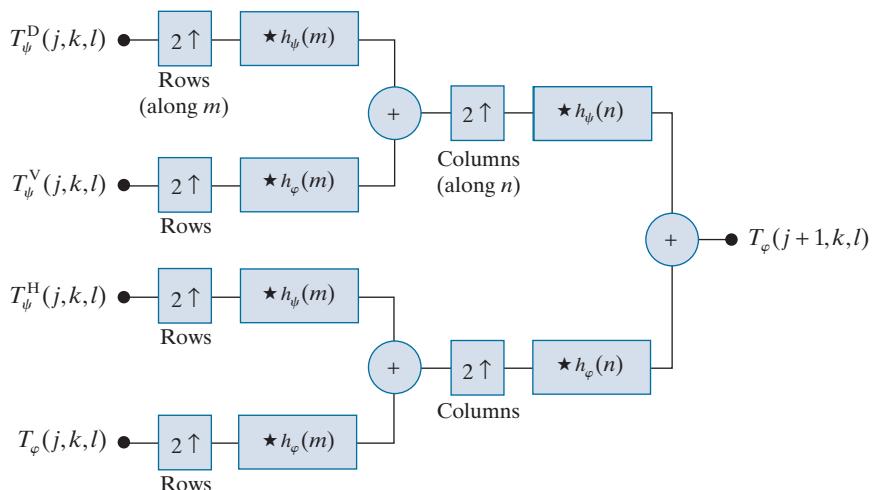
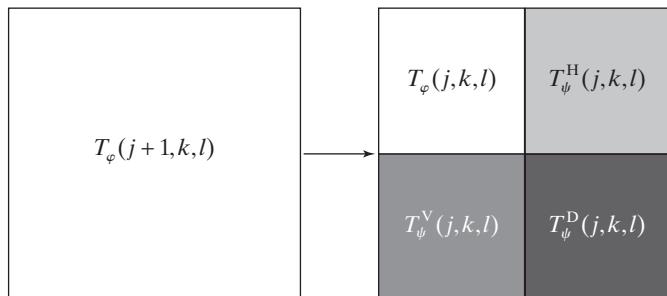
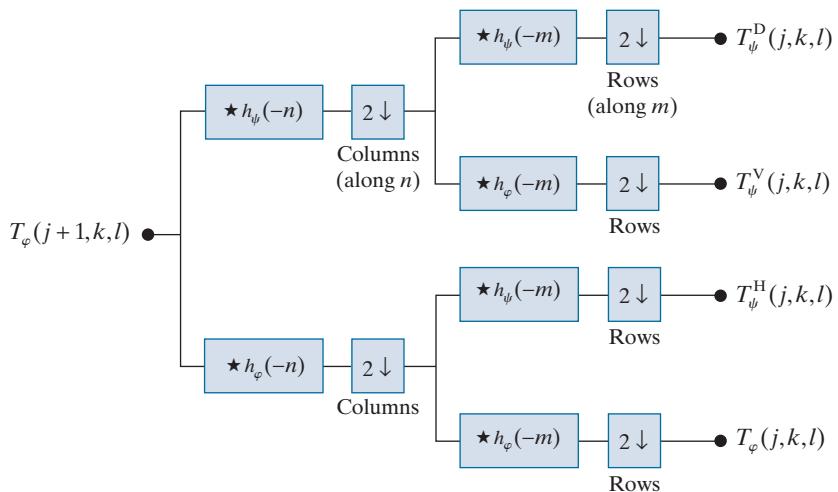
The single-scale filter bank of Fig. 7.29(a) can be “iterated” (by tying the approximation output to the input of another filter bank) to produce a $P \leq J$ scale transform in which scale j is equal to $J-1, J-2, \dots, J-P$. As in the 1-D case, image $f(x, y)$ is used as the $T_\varphi(J, k, l)$ input. Convolving its rows with $h_\varphi(-n)$ and $h_\psi(-n)$ and downsampling its columns, we get two subimages whose horizontal resolutions are reduced by a factor of 2. The highpass or detail component characterizes the image’s high-frequency information with vertical orientation; the lowpass, approximation component contains its low-frequency, vertical information. Both subimages are then filtered columnwise and downsampled to yield four quarter-size output subimages— T_φ , T_ψ^H , T_ψ^V , and T_ψ^D . These subimages, which are normally arranged as in Fig. 7.29(b), are the inner products of $f(x, y)$ and the two-dimensional scaling and

a
b
c

FIGURE 7.29

The 2-D fast wavelet transform: (a) the analysis filter bank; (b) the resulting decomposition; and (c) the synthesis filter bank.

Note m and n are dummy variables of convolution, while j , like in the 1-D case, is scale, and k and l are translations.



wavelet functions in Eqs. (7-152) through (7-155), followed by downsampling by two in each dimension.

Figure 7.29(c) shows the synthesis filter bank that reverses the process just described. As would be expected, the reconstruction algorithm is similar to the 1-D case. At each iteration, four-scale j approximation and detail subimages are upsampled and convolved with two 1-D filters—one operating on the subimages' columns and the other on its rows. Addition of the results yields the scale $j + 1$ approximation, and the process is repeated until the original image is reconstructed.

EXAMPLE 7.22: Computing 2-D fast wavelet transforms.

In this example, we compute a 2-D, multiscale FWT with respect to Haar basis functions and compare it to the traditional Haar transform of Section 7.9. Figures 7.30(a)–(d) show a 512×512 monochrome image of a vase on a windowsill, its one- and two-scale discrete wavelet transforms with respect to Haar basis functions, and its Haar transform, respectively. The computation of the wavelet transforms will be discussed shortly. The Haar transform in Fig. 7.30(d) is computed using a 512×512 Haar transformation matrix [see Eqs. (7-114) through (7-118)] and the matrix-based operations defined in Eq. (7-35). The detail coefficients in Figs. 7.30(b) and (c), as well as the Haar transform coefficients in Fig. 7.30(d), are scaled to make their underlying structure more visible. When the same area of any two transforms is shaded in blue, the corresponding pixels within those areas are identical in value.

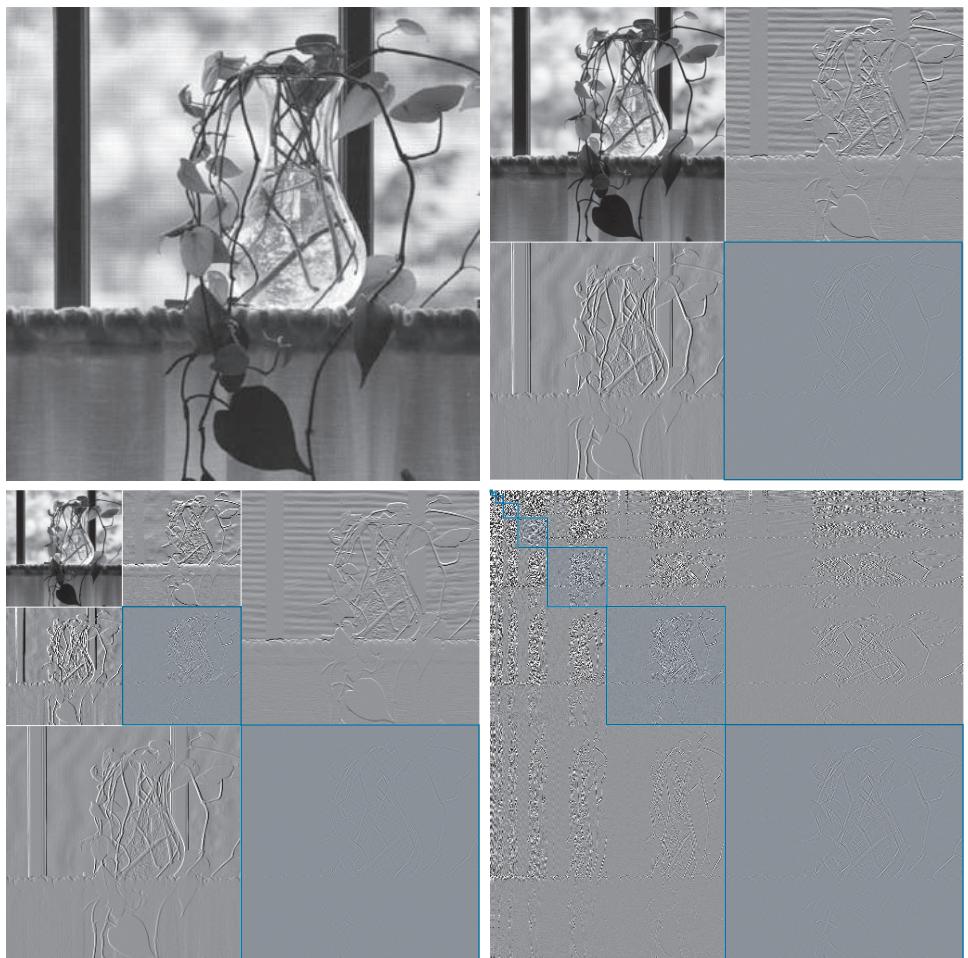
To compute the one-scale FWT of Fig. 7.30(b), the image in Fig. 7.30(a) is used as the input to a filter bank like that of Fig. 7.29(a). Since $J = \log_2 512 = 9$ and $P = 1$, $T_\varphi(9, k, l) = f(x, y)$ and the four resulting quarter-size decomposition outputs [i.e., approximation $T_\varphi(8, k, l)$ and horizontal, vertical, and diagonal details $T_\psi^H(8, k, l)$, $T_\psi^V(8, k, l)$, and $T_\psi^D(8, k, l)$] are then arranged in accordance with Fig. 7.29(b) to produce Fig. 7.30(b). A similar process is used to generate the two-scale transform in Fig. 7.30(c), but the input to the filter bank is a quarter-size approximation subimage $T_\varphi(8, k, l)$, from the upper left-hand corner of Fig. 7.30(b). As can be seen in Fig. 7.30(c), the quarter-size approximation subimage is then replaced by the four quarter-size (now 1/16th of the size of the original image) decomposition results that were generated by the second filtering pass. Each pass through the filter bank produces four quarter-size output images which are substituted for the input from which they were derived. The process is repeatable until $P = J = 9$, which produces a nine-scale transform.

Note the directional nature of the subimages associated with T_ψ^H , T_ψ^V , and T_ψ^D in Figs. 7.30(b) and (c). The diagonal details in these images (i.e., the T_ψ^D areas shaded in blue) are identical to the correspondingly shaded areas of the Haar transform in Fig. 7.30(d). In the 1-D case, as was demonstrated in Example 7.19, a J -scale 1-D FWT with respect to Haar basis functions is the same as its 1-D Haar transform counterpart. This is due to the fact that the basis functions of the two transforms are identical; both contain one scaling function and a series of scaled and translated wavelet functions. In the 2-D case, however, the basis images differ. The 2-D separable scaling and wavelet functions defined in Eqs. (7-153) through (7-155) introduce horizontal and vertical directionality that is not present in a traditional Haar transform. Figures 7.31(a) and (b), for example, are the basis images of an 8×8 Haar transform and three-scale FWT with respect to Haar basis functions. Note the blue highlighted regions along the main diagonals in which the basis images match. The same pattern occurs in Fig. 7.30(b) through (d). If a nine-scale wavelet transform of the vase were computed, it would match the Haar transform in Fig. 7.30(d) in all of its shaded areas.

a b
c d

FIGURE 7.30

(a) A 512×512 image of a vase; (b) a one-scale FWT; (c) a two-scale FWT; and (d) the Haar transform of the original image. All transforms have been scaled to highlight their underlying structure. When corresponding areas of two transforms are shaded in blue, the correspondent pixels are identical.



a b

FIGURE 7.31

(a) Haar basis images of size 8×8 [from Fig. 7.18(c)] and (b) the basis images of a three-scale 8×8 discrete wavelet transform with respect to Haar basis functions.



We conclude the section with a simple example that demonstrates the use of wavelets in image processing. As in the Fourier domain, the basic approach is to:

1. Compute the 2-D wavelet transform of an image with respect to a selected wavelet basis. Table 7.1 shows some representative bases, including their scaling and wavelet functions and corresponding filter coefficients. The filter coefficients are given in the context of Fig. 7.26. For orthonormal wavelets, lowpass synthesis coefficients are specified; the remaining filters must be computed using Eq. (7-151). For biorthonormal wavelets, two analysis filters are given and the synthesis filters must be computed using Eqs. (7-149) and (7-150).
2. Alter the computed transform to take advantage of the DWT's ability to (1) decorrelate image pixels, (2) reveal important frequency and temporal characteristics, and/or (3) measure the image's similarity to the transform's basis images. Modifications designed for image smoothing, sharpening, noise reduction, edge detection, and compression (to name only a few) are possible.
3. Compute the inverse wavelet transform.

Since the discrete wavelet transform decomposes an image into a weighted sum of spatially limited, bandlimited basis images, most Fourier-based imaging techniques have an equivalent “wavelet domain” counterpart.

EXAMPLE 7.23: Wavelet-based edge detection.

Figure 7.32 provides a simple illustration of the preceding three steps. Figure 7.32(a) shows a 128×128 computer-generated image of 2-D sine-shaped pulses on a black background. Figure 7.32(b) is the two-scale discrete wavelet transform of the image with respect to 4th-order *symlets*, short for “symmetrical wavelets.” Although they are not perfectly symmetrical, they are designed to have the least asymmetry and highest number of *vanishing moments*[†] for a given compact support (Daubechies [1992]). Row 4 of Table 7.1 shows the wavelet and scaling functions of the symlets, as well as the coefficients of the corresponding lowpass synthesis filter. The remaining filter coefficients are obtained using Eq. (7-151) with K , the number of filter coefficients, set to 8:

$$g_1(n) = (-1)^n g_0(7-n) = \{-0.0758, 0.0296, 0.4976, -0.8037, 0.2979, 0.0992, -0.0126, -0.0322\}$$

$$h_0(n) = g_0(7-n) = \{-0.0758, -0.0296, 0.4976, 0.8037, 0.2979, -0.0992, -0.0126, 0.0322\}$$

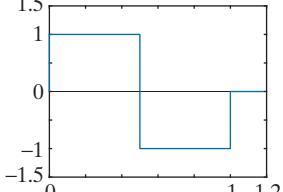
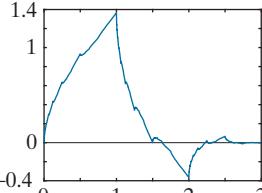
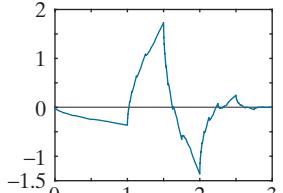
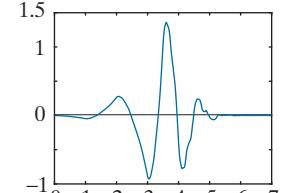
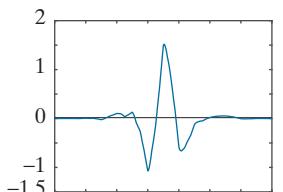
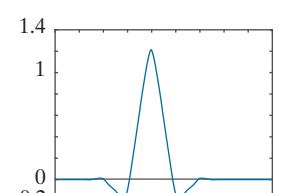
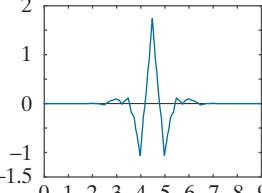
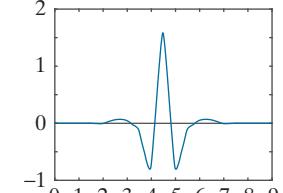
$$h_1(n) = g_1(7-n) = \{-0.0322, -0.0126, 0.0992, 0.2979, -0.8037, 0.4976, 0.0296, -0.0758\}$$

In Fig. 7.32(c), the approximation component of the discrete wavelet transform has been eliminated by setting its values to zero. As Fig. 7.32(d) shows, the net effect of computing the inverse transform using these modified coefficients is edge enhancement, reminiscent of the Fourier-based image sharpening results discussed in Section 4.9. Note how well the transitions between signal and background are delineated, despite the fact that they are relatively soft, sinusoidal transitions. By zeroing the horizontal details as well—see Figs. 7.32(e) and (f)—we can isolate vertical edges.

[†]The k th moment of wavelet $\psi(x)$ is $m(k) = \int x^k \psi(x) dx$. Vanishing moments impact the smoothness of the scaling and wavelet functions and our ability to represent them as polynomials. An order- N symlet has N vanishing moments.

TABLE 7.1

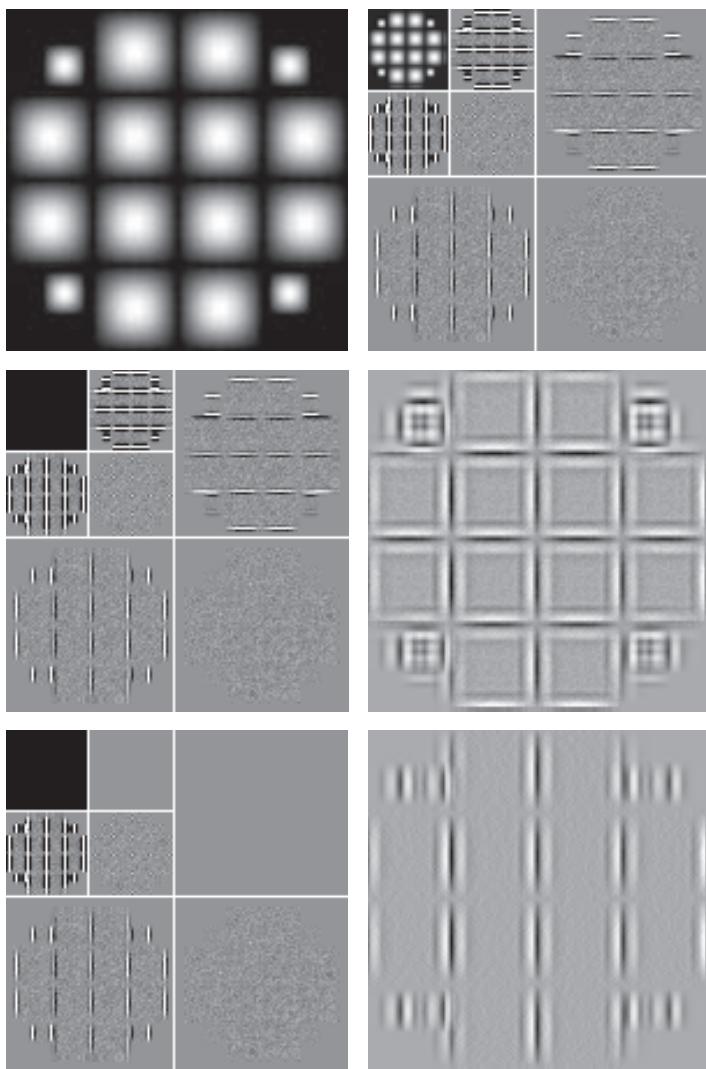
Some representative wavelets.

Wavelet Name or Family	Scaling Function	Wavelet Function	Filter Coefficients
Haar The oldest and simplest wavelets. Orthogonal and discontinuous.			$g_0(n) = \{1/\sqrt{2}, 1/\sqrt{2}\}$
Daubechies family Orthogonal with the most vanishing moments for a given support. Denoted db N , where N is the number of vanishing moments; db2 and db4 shown; db1 is the Haar of the previous row.			$g_0(n) = \{0.482963, 0.836516, 0.224144, -0.129410\}$
			$g_0(n) = \{0.230372, 0.714847, 0.630881, -0.027984, -0.187035, 0.030841, 0.032883, -0.010597\}$
Symlet family Orthogonal with the least asymmetry and most vanishing moments for a given support (sym4 or 4th order shown).			$g_0(n) = \{0.032231, -0.012604, -0.099220, 0.297858, 0.803739, 0.497619, -0.029636, -0.075766\}$
Cohen-Daubechies-Feauveau 9/7 Biorthogonal B-spline used in the irreversible JPEG2000 compression standard (see Chapter 8).			$h_0(n) = \{0.026749, -0.016864, -0.078223, 0.266864, 0.602949, 0.266864, -0.078223, -0.016864, 0.026749\}$
			$h_1(n) = \{-0.091271, -0.057544, 0.591272, -1.115087, 0.591272, 0.057544, -0.091271, 0\}$

a	b
c	d
e	f

FIGURE 7.32

Modifying a DWT for edge detection:
 (a) original image;
 (b) two-scale DWT with respect to 4th-order symlets; (c) modified DWT with the approximation set to zero; (d) the inverse DWT of (c); (e) modified DWT with the approximation and horizontal details set to zero; and (f) the inverse DWT of (e).
 (Note when the detail coefficients are zero, they are displayed as middle gray; when the approximation coefficients are zeroed, they display as black.)



WAVELET PACKETS

A fast wavelet transform decomposes a function into a sum of scaling and wavelet functions whose bandwidths are logarithmically related. That is, the low-frequency content of the function is represented using scaling and wavelet functions with narrow bandwidths, while the high-frequency content is represented using functions with wider bandwidths. This is apparent in Fig. 6.5. Each horizontal strip of constant height tiles, which are the basis functions of a single FWT scale, increases logarithmically in height as you move up the frequency axis. To obtain greater control over the partitioning of the time-frequency plane (e.g., to get smaller bandwidths for higher frequencies), the FWT must be generalized to yield a more flexible decomposition

called a *wavelet packet* (Coifman and Wickerhauser [1992]). The cost of this generalization is an increase in computational complexity from $O(N)$ for the FWT to $O(N \log_2 N)$ for a wavelet packet.

Consider again the three-scale filter bank of Fig. 7.24(a), but imagine the decomposition as a *binary tree*. Figure 7.33(a) details the structure of that tree, and links the appropriate FWT scaling and wavelet coefficients from Fig. 7.24(a) to the tree's *nodes*. The *root node* is assigned the highest-scale approximation coefficients, which are samples of the function itself, while the *leaves* inherit the transform's approximation and detail coefficient outputs. Two intermediate nodes, $T_\varphi(J-1,k)$ and $T_\psi(J-1,k)$, are filter-bank approximations that are subsequently filtered to become four additional leaf nodes. Note the coefficients of each node are the weights of a linear expansion that produces a bandlimited "piece" of root node $f(x)$. Because any such piece is an element of a known scaling or wavelet subspace, we can replace the generating coefficients in Fig. 7.33(a) by the corresponding subspace. The result is the *subspace analysis tree* of Fig. 7.33(b).

Analysis trees provide a compact and informative way of representing multiscale wavelet transforms. They are simple to draw, take less space than their corresponding filter and subsampler-based block diagrams, and make it relatively easy to detect valid decompositions. The three-scale analysis tree of Fig. 7.33(b), for example, suggests three possible expansion options:

Recall that \oplus denotes the union of spaces (like the union of sets). Equations (7-156) through (7-158) can be derived by the repeated application of Eq. (7-128).

$$V_J = V_{J-1} \oplus W_{J-1} \quad (7-156)$$

$$V_J = V_{J-2} \oplus W_{J-2} \oplus W_{J-1} \quad (7-157)$$

$$V_J = V_{J-3} \oplus W_{J-3} \oplus W_{J-2} \oplus W_{J-1} \quad (7-158)$$

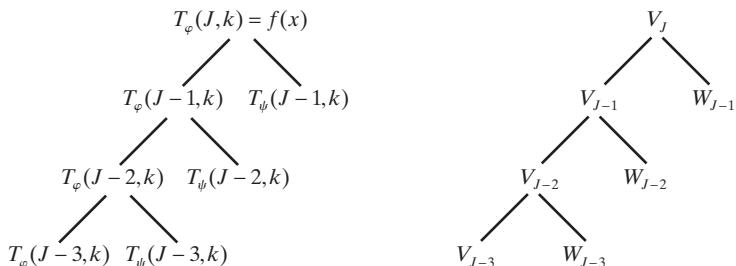
They correspond to the one-, two-, and three-scale FWT decompositions of a 1-D function. A valid decomposition requires one approximation term (or scaling subspace) and enough detail components (or wavelet subspaces) to cover the spectrum of Fig. 7.24(b). In general, a P -scale FWT analysis tree supports P unique decompositions.

Analysis trees are also an efficient mechanism for representing wavelet packets, which are nothing more than conventional wavelet transforms with the details filtered iteratively. Thus, the three-scale FWT analysis tree of Fig. 7.33(b) becomes the

a b

FIGURE 7.33

An (a) coefficient tree and (b) analysis tree for the two-scale FWT analysis bank of Fig. 7.24.



three-scale wavelet packet tree of Fig. 7.34. Note the additional subscripting that must be introduced. The first subscript of each double-subscripted node identifies the scale of the FWT parent node from which it is descended. The second, a variable length string of “A”’s and “D”’s, encodes the path from the parent node to the node being examined. An “A” designates approximation filtering, while a “D” indicates detail filtering. Subspace node $W_{J-1,DA}$, for example, is obtained by “filtering” the scale $J-1$ FWT coefficients (i.e., parent W_{J-1} in Fig. 7.34) through an additional detail filter (yielding $W_{J-1,D}$), followed by an approximation filter (giving $W_{J-1,DA}$). Figures 7.35(a) and (b) are the filter-bank and spectrum-splitting characteristics of the analysis tree in Fig. 7.34, respectively. Note the “naturally ordered” outputs of the filter bank in Fig. 7.35(a) have been reordered based on frequency content in Fig. 7.35(b) (see Problem 7.46 for more on “frequency ordered” wavelets).

The three-scale packet tree in Fig. 7.34 almost triples the number of decompositions (and associated time-frequency tilings) that are possible with the three-scale FWT tree. Recall that in a normal FWT, we split, filter, and downsample the lowpass bands alone. This creates a fixed logarithmic (base 2) relationship between the bandwidths of the scaling and wavelet spaces used in the representation of a function [see Figure 7.24(b)]. Thus, while the three-scale FWT analysis tree of Fig. 7.24(a) offers three possible decompositions—defined by Eqs. (7-156) to (7-158)—the wavelet packet tree of Fig. 7.34 supports 26 different decompositions. For instance, V_J and therefore function $f(x)$ can be expanded as

$$\begin{aligned} V_J = & V_{J-3} \oplus W_{J-3} \oplus W_{J-2,A} \oplus W_{J-2,D} \oplus W_{J-1,AA} \\ & \oplus W_{J-1,AD} \oplus W_{J-1,DA} \oplus W_{J-1,DD} \end{aligned} \quad (7-159)$$

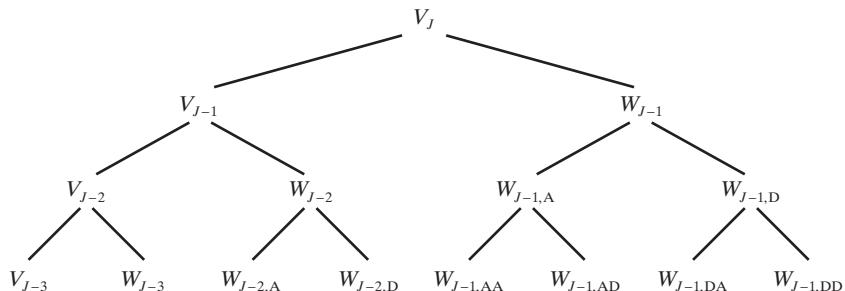
whose spectrum is shown in Fig. 7.35(b), or as

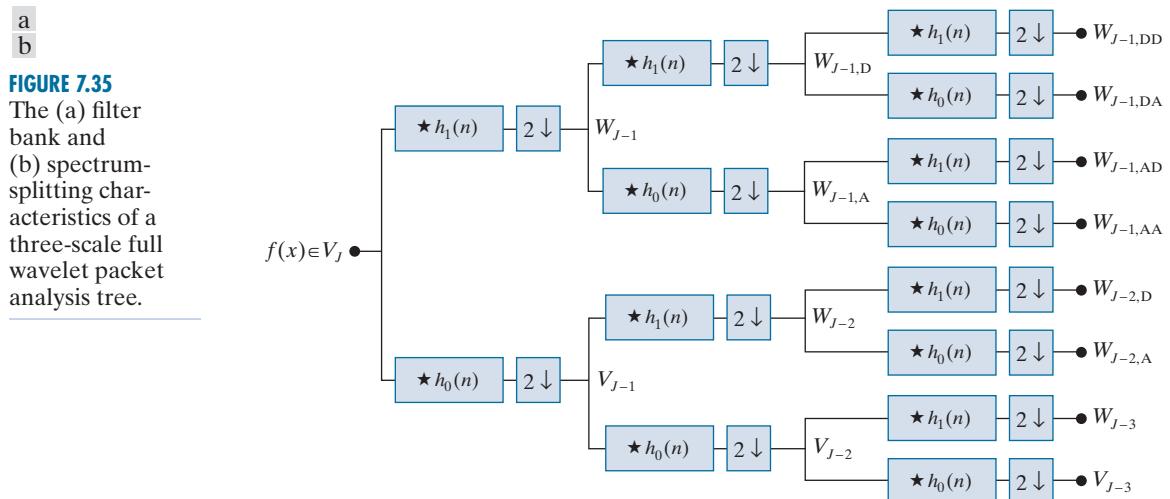
$$V_J = V_{J-1} \oplus W_{J-1,A} \oplus W_{J-1,DA} \oplus W_{J-1,DD} \quad (7-160)$$

whose spectrum is depicted in Fig. 7.36. Note the difference between this last spectrum and the full packet spectrum of Fig. 7.35(b), or the three-scale FWT spectrum

FIGURE 7.34

A three-scale wavelet packet analysis tree.



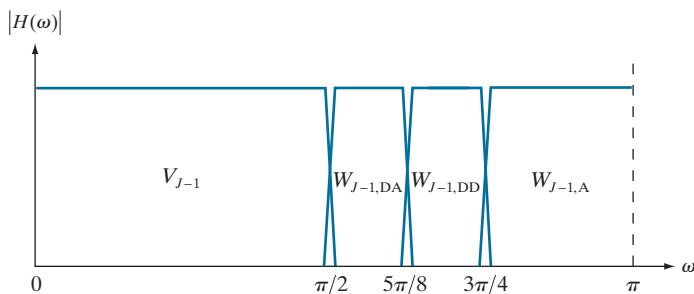


of Fig. 7.24(b). In general, P -scale, 1-D wavelet packet transforms (and associated $P + 1$ -level analysis trees) support

$$D(P+1) = [D(P)]^2 + 1 \quad (7-161)$$

unique decompositions, where $D(1) = 1$. With such a large number of valid expansions, packet-based transforms provide improved control over the partitioning of the

FIGURE 7.36
The spectrum of the decomposition in Eq. (7-160).



spectrum of the decomposed function. The cost of this control is an increase in computational complexity. Compare the filter bank in Fig. 7.35(a) to that of Fig. 7.24(a).

Now consider the 2-D, four-band filter bank of Fig. 7.29(a). As was noted earlier, it splits approximation $T_\varphi(j+1, k, l)$ into outputs $T_\varphi(j, k, l)$, $T_\psi^H(j, k, l)$, $T_\psi^V(j, k, l)$, and $T_\psi^D(j, k, l)$. As in the 1-D case, it can be “iterated” to generate P -scale transforms at scales $j = J - 1, J - 2, \dots, J - P$, with $T_\varphi(J, k, l) = f(x, y)$. The spectrum resulting from the first iteration, with $j + 1 = J$ in Fig. 7.29(a), is shown in Fig. 7.37(a). Note it divides the frequency plane into four equal areas. The low-frequency quarter-band in the center of the plane coincides with transform coefficients $T_\varphi(J - 1, k, l)$ and scaling space V_{J-1} . This nomenclature is consistent with the 1-D case. To accommodate the 2-D nature of the input, however, we now have three (rather than one) wavelet subspaces. They are denoted W_{J-1}^H , W_{J-1}^V , and W_{J-1}^D and correspond to coefficients $T_\psi^H(J - 1, k, l)$, $T_\psi^V(J - 1, k, l)$, and $T_\psi^D(J - 1, k, l)$, respectively. Figure 7.37(b) shows the resulting four-band, single-scale *quaternary FWT analysis tree*. Note the superscripts that link the wavelet subspace designations to their transform coefficient counterparts.

Figure 7.38 shows a portion of a three-scale, 2-D wavelet packet analysis tree. Like its 1-D counterpart in Fig. 6.34, the first subscript of every node that is a descendant of a conventional FWT detail node is the scale of the parent detail node. The second subscript, a variable length string of “A”s, “H”s, “V”s, and “D”s, encodes the path from the parent node to the node under consideration. The node labeled $W_{J-1,VD}^H$, for example, is obtained by “row/column filtering” the scale $J - 1$ FWT horizontal detail coefficients (i.e., parent W_{J-1}^H in Fig. 7.38) through an additional detail/approximation filter (yielding $W_{J-1,V}^H$), followed by a detail/detail filter (giving $W_{J-1,VD}^H$). A P -scale, 2-D wavelet packet tree supports

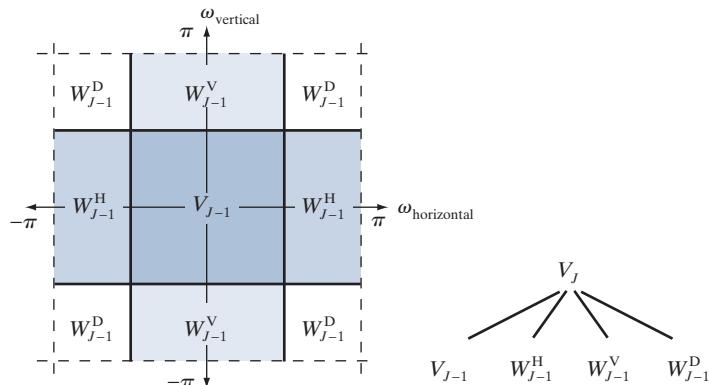
$$D(P + 1) = [D(P)]^4 + 1 \quad (7-162)$$

unique expansions, where $D(1) = 1$. Thus, the three-scale tree of Fig. 7.38 offers 83,522 possible decompositions. The problem of selecting among them is the subject of the next example.

a b

FIGURE 7.37

The first decomposition of a 2-D FWT: (a) the spectrum and (b) the subspace analysis tree.



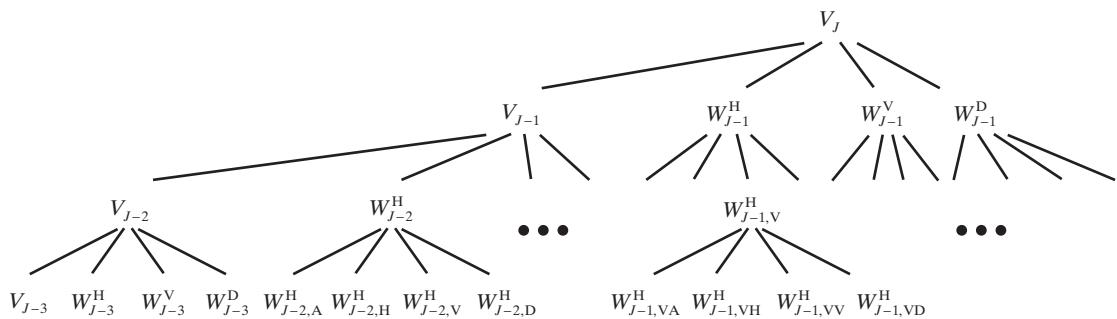


FIGURE 7.38 A three-scale, full wavelet packet decomposition tree. Only a portion of the tree is provided.

EXAMPLE 7.24: Two-dimensional wavelet packet decompositions.

As noted in the above discussion, a single wavelet packet tree presents numerous decomposition options. In fact, the number of possible decompositions is often so large that it is impractical, if not impossible, to enumerate or examine them individually. An efficient algorithm for finding optimal decompositions with respect to application specific criteria is highly desirable. As will be seen, classical *entropy*- and *energy-based cost functions* are applicable in many situations and are well-suited for use in binary and quaternary tree searching algorithms.

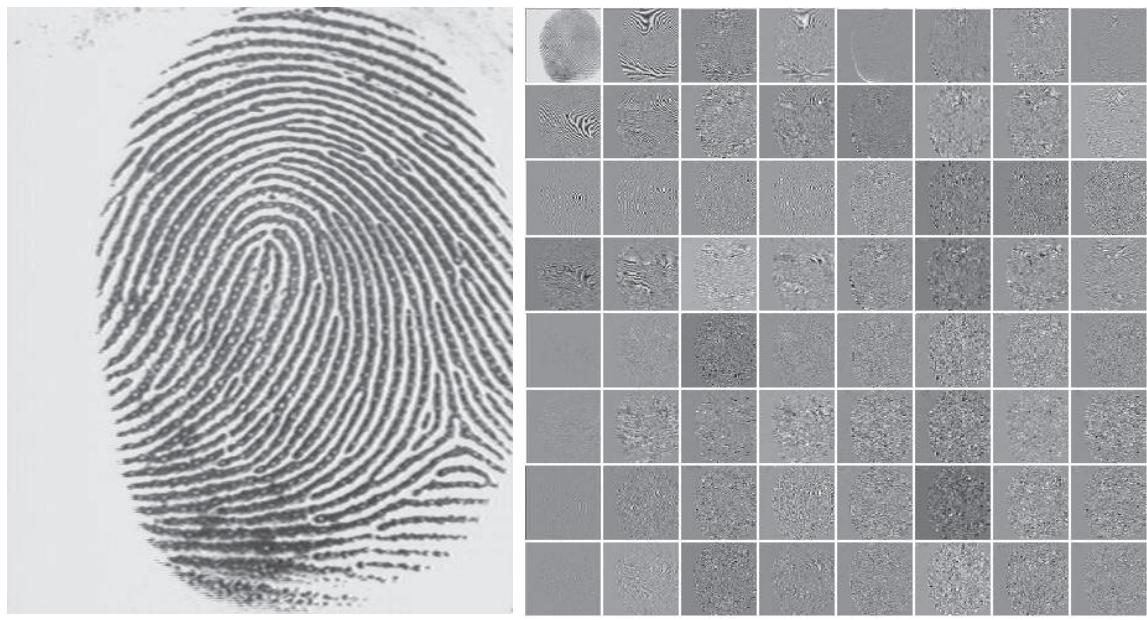
Consider the problem of reducing the amount of data needed to represent the 400×480 fingerprint image in Fig. 7.39(a). Image compression is discussed in detail in Chapter 8. In this example, we want to select the “best” three-scale wavelet packet decomposition as a starting point for the compression process. Using three-scale wavelet packet trees, there are 83,522 [per Eq. (7-162)] potential decompositions. Figure 7.39(b) shows one of them—a full wavelet packet, 64-leaf decomposition like the analysis tree of Fig. 7.38. Note the leaves of the tree correspond to the subbands of the 8×8 array of decomposed subimages in Fig. 7.39(b). The probability that this particular 64-leaf decomposition is in some way optimal for the purpose of compression, however, is relatively low. In the absence of a suitable optimality criterion, we can neither confirm nor deny it.

One reasonable criterion for selecting a decomposition for the compression of the image of Fig. 7.39(a) is the additive cost function

$$E(f) = \sum_{x,y} |f(x, y)| \quad (7-163)$$

This function provides one possible measure[†] of the energy content of 2-D function f . Under this measure, the energy of function $f(x, y) = 0$ for all x and y is 0. High values of E on the other hand, are indicative of functions with many nonzero values. Since most transform-based compression schemes work by truncating or thresholding the small coefficients to zero, a cost function that maximizes the number of near-zero values is a reasonable criterion for selecting a “good” decomposition from a compression point of view.

[†]Other possible energy measures include the sum of the squares of $f(x, y)$, the sum of the log of the squares, etc. Problem 7.48 defines one possible entropy-based cost function.



a b

FIGURE 7.39 (a) A scanned fingerprint and (b) its three-scale, full wavelet packet decomposition. Although the 64 subimages of the packet decomposition appear to be square (e.g., note the approximation subimage), this is merely an aberration of the program used to produce the result. (Original image courtesy of the National Institute of Standards and Technology.)

The cost function just described is both computationally simple and easily adapted to tree optimization routines. The optimization algorithm must use the function to minimize the “cost” of the leaf nodes in the decomposition tree. Minimal energy leaf nodes should be favored because they have more near-zero values, which leads to greater compression. Because the cost function of Eq. (7-163) is a local measure that uses only the information available at the node under consideration, an efficient algorithm for finding minimal energy solutions is easily constructed as follows:

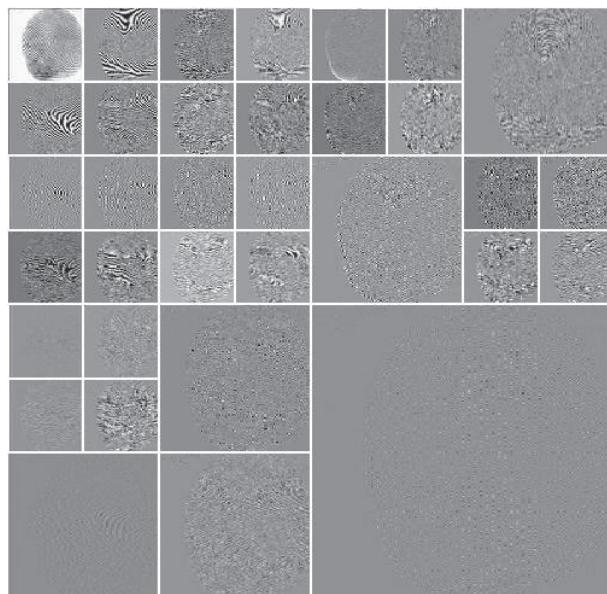
For each node of the analysis tree, beginning with the root and proceeding level by level to the leaves:

1. Compute both the energy of the node, denoted E_P (for parent energy), and the energy of its four offspring—denoted as E_A , E_H , E_V , and E_D . For two-dimensional wavelet packet decompositions, the parent is a two-dimensional array of approximation or detail coefficients; the offspring are the filtered approximation, horizontal, vertical, and diagonal details.
2. If the combined energy of the offspring is less than the energy of the parent (that is, $E_A + E_H + E_V + E_D < E_P$), include the offspring in the analysis tree. If the combined energy of the offspring is greater than or equal to that of the parent, prune the offspring, keeping only the parent. It is a leaf of the optimized analysis tree.

The preceding algorithm can be used to (1) prune wavelet packet trees or (2) design procedures for computing optimal trees from scratch. In the latter case, nonessential siblings—descendants of nodes that

FIGURE 7.40

An optimal wavelet packet decomposition for the fingerprint of Fig. 7.39(a).



would be eliminated in Step 2 of the algorithm—would not be computed. Figure 7.40 shows the optimized decomposition that results from applying the algorithm just described to the image of Fig. 7.39(a) with the cost function of Eq. (7-163). Note many of the original full packet decomposition's 64 subbands in Fig. 7.39(b) have been eliminated. In addition, the subimages that are not split (further decomposed) in Fig. 7.40 are relatively smooth and composed of pixels that are middle gray in value. Because all but the approximation subimage of this figure have been scaled so that gray level 128 indicates a zero-valued coefficient, these subimages contain little energy. There would be no overall decrease in energy realized by splitting them.

The preceding example is based on a real-world problem that was solved through the use of wavelets. The Federal Bureau of Investigation (FBI) currently maintains a large database of fingerprints, and has established a wavelet-based national standard for the digitization and compression of fingerprint images (FBI [1993]). Using Cohen-Daubechies-Feauveau (CDF) biorthogonal wavelets (Cohen, Daubechies, and Feauveau [1992]), the standard achieves a typical compression ratio of 15:1. Table 7.2 details the required analysis filter coefficients. Because the scaling and wavelet functions of the CDF family are symmetrical and have similar lengths, they are among the most widely used biorthogonal wavelets. The advantages of wavelet-based compression over the more traditional JPEG approach are examined in Chapter 8.

Summary, References, and Further Reading

The material in this chapter establishes a solid mathematical foundation for understanding and accessing the role of image transforms, including the discrete wavelet transform, in image processing. We approach transforms as series expansions in which the transform coefficients are inner products of a set of orthonormal or biorthonormal basis functions and the images being transformed. For many transforms, these inner products can be implemented

TABLE 7.2

Biorthogonal Cohen-Daubechies-Feauveau reconstruction and decomposition filter coefficients with 6 and 8 vanishing moments, respectively. (Cohen, Daubechies, and Feauveau [1992]).

n	$h_0(n)$	$h_1(n)$	n	$h_0(n)$	$h_1(n)$
0	0	0	9	0.825923	0.417849
1	0.001909	0	10	0.420796	0.040368
2	-0.001914	0	11	-0.094059	-0.078722
3	-0.016991	0.014427	12	-0.077263	-0.014468
4	0.011935	-0.014468	13	0.049733	0.0144263
5	0.049733	-0.078722	14	0.011935	0
6	-0.077263	0.040368	15	-0.016991	0
7	-0.094059	0.417849	16	-0.0019	0
8	0.420796	-0.758908	17	0.0019	0

as straightforward matrix operations. Further reading on the matrix formulation of image transforms is available in books like those of Andrews [1970] and Wang [2012], and in the original papers on the transforms themselves. See, for example, the original papers on the Haar transform (Haar [1910]), Walsh transform (Walsh [1923]), Hadamard transform (Hadamard [1893]), and the slant transform (Pratt, Chen, and Welch [1974]).

There are many good texts on wavelets and their application. Several complement our treatment and were relied upon during the development of the wavelet transform section of the chapter. Included among them are the books by Vetterli and Kovacevic [1995] and Burrus, Gopinath, and Guo [1998]. A partial listing of the imaging applications that have been approached from a wavelet point of view includes image matching, registration, segmentation, denoising, restoration, enhancement, compression (see Chapter 8), morphological filtering, and computed tomography. The history of wavelet analysis is recorded in a book by Hubbard [1998]. The early predecessors of wavelets were developed simultaneously in different fields and unified in a paper by Mallat [1987]. It brought a mathematical framework to the field. Much of the history of wavelets can be traced through the works of Meyer [1987] [1990] [1992a, 1992b] [1993], Mallat [1987] [1989a–c] [1998], and Daubechies [1988] [1990] [1992] [1993] [1996]. Finally, there have been a number of special issues devoted to wavelets, including a special issue on wavelet transforms and multiresolution signal analysis in the *IEEE Transactions on Information Theory* [1992], a special issue on wavelets and signal processing in the *IEEE Transactions on Signal Processing* [1993], and a special section on multiresolution representation in the *IEEE Transactions on Pattern Analysis and Machine Intelligence* [1989]. All of the examples in the chapter were done using MATLAB (see Gonzalez et al. [2004]).

Problems

Solutions to the problems marked with an asterisk (*) are in the DIP4E Student Support Package (consult the book website: www.ImageProcessingPlace.com).

7.1 Given column vectors

$$\mathbf{s}_0 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \mathbf{s}_1 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad \mathbf{s}_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

- (a) Prove that \mathbf{s}_0 , \mathbf{s}_1 , and \mathbf{s}_2 are orthogonal.
 (b)* Are they orthonormal? If not, normalize them to create a transformation matrix of

orthonormal vectors.

- (c) Using the result of (b), write an orthogonal transformation matrix for \mathbf{s}_0 , \mathbf{s}_1 , and \mathbf{s}_2 .
 (d) Compute the transform of column vector $\mathbf{f} = [3 \quad -6 \quad 5]$.
 (e) Compute the inverse transform of the result in (d).

- 7.2** Prove Eq. (7-23).
- 7.3*** Prove that $r(x,u) = s(x,u)$ in Eqs. (7-16) and (7-17) for real, orthonormal basis vectors.
- 7.4** Prove that if $\mathbf{A}^* \mathbf{T} \mathbf{A} = \mathbf{I}$, the associated expansion functions are orthonormal.
- 7.5** Prove that matrix \mathbf{A}_3 in Example 7.3 is an orthogonal transformation matrix.
- 7.6** Prove that orthogonal transformations preserve inner products.
- 7.7** Using Eqs. (7-4) and (7-5),
- Find the norm of $\mathbf{f} = [3 + j2 \quad 1 - j]^T$.
 - Find the norm of $\mathbf{g} = [0.707 \quad -0.707]^T$.
 - Find the angle between $\mathbf{h} = [0.707 \quad 0.707]^T$ and \mathbf{g} .
 - * Find the norm of $f(x) = \cos x$.
 - Find the angle between f from (d) and $g(x) = \sin x$.
 - Are f and g orthogonal to one another?
 - Are f and g orthonormal?
- 7.8** Using the results from Problem 7.1(c)–(e) and column vector $\mathbf{g} = [2 \quad 7 \quad 1]$:
- Compute the angle between \mathbf{f} and \mathbf{g} .
 - Compute the distance between \mathbf{f} and \mathbf{g} . Hint: The distance between vectors \mathbf{f} and \mathbf{g}
- $$d = \sqrt{\langle \mathbf{f} - \mathbf{g}, \mathbf{f} - \mathbf{g} \rangle}$$
- (c)* Show that angles and distances are preserved by this orthogonal transform.
- 7.9** Compute the inverse transform of \mathbf{T} in Example 7.3.
- 7.10** Prove that the set of sinusoidal expansion functions $\{1, \cos x, \sin x, \cos 2x, \sin 2x, \dots\}$ are orthogonal on the interval $[-\pi, \pi]$.
- 7.11** Compute the expansion coefficients of 2-tuple $[7 \quad 1]^T$ and write the corresponding expansions for the following bases:
- * $\mathbf{s}_0 = [0.707 \quad 0.707]^T$, $\mathbf{s}_1 = [0.707 \quad -0.707]^T$ on the set of real 2-tuples.
 - $\mathbf{s}_0 = [1 \quad 0]^T$, $\mathbf{s}_1 = [1 \quad 1]^T$ and the dual vectors $\tilde{\mathbf{s}}_0 = [1 \quad -1]^T$, $\tilde{\mathbf{s}}_1 = [0 \quad 1]^T$.
- 7.12** Are expansion functions

$$\mathbf{s}_0 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \quad \mathbf{s}_1 = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} \quad \mathbf{s}_2 = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \quad \mathbf{s}_3 = \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \end{bmatrix}$$

orthonormal? If so, write the corresponding orthogonal transformation matrix.

- 7.13** If $\mathbf{f} = [4 \quad -3 \quad 2 \quad 1]^T$, find the transform of \mathbf{f} using the transformation matrix of Problem 7.12. Then compute the inverse and show that the transform is reversible.

- 7.14** Given the 2-D matrix

$$\mathbf{F} = \begin{bmatrix} 4 & -4 & 4 & 0.5 \\ -3 & 1 & 5 & -0.5 \\ 2 & -4 & 8 & -0.5 \\ 1 & -3 & 3 & -1.5 \end{bmatrix}$$

- Compute the transform of \mathbf{F} with respect to the transformation matrix of Problem 7.12.
- * Using the 1-D transform computed in Problem 7.13, explain how a 2-D transform is computed as two 1-D transforms.
- Compute the 2-D inverse transform of the result from (a).

- 7.15** Prove that expansion functions

$$\mathbf{u}_0 = \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix} \quad \mathbf{u}_1 = \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}$$

$$\tilde{\mathbf{u}}_0 = \begin{bmatrix} \sqrt{2}/3 \\ 2\sqrt{2}/3 \end{bmatrix} \quad \tilde{\mathbf{u}}_1 = \begin{bmatrix} -2/3 \\ 2/3 \end{bmatrix}$$

are biorthonormal. Then show by example whether inner products, angles, and distances are preserved by the transform.

- 7.16** Prove that \mathbf{A} and $\tilde{\mathbf{A}}$ in Example 7.5 are biorthonormal.

- (a) Using biorthonormal matrices \mathbf{A} and $\tilde{\mathbf{A}}$ of Example 7.5, compute the transform of 4×4 array

$$\mathbf{F} = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

- (b)** Compute the inverse transform of the result in (a).
- 7.17*** Write a pair of 2-D transform matrix equations for rectangular arrays and complex orthonormal expansion functions.
- 7.18** Write a pair of 2-D transform matrix equations for complex biorthonormal expansion functions.
- 7.19** Show that Eq. (7-59) of Example 7.6 is equivalent to $\sin(2\pi x)$.
- 7.20*** Prove that the DFT expansion functions of Eq. (7-56) are orthonormal.
- 7.21** Prove Eq. (7-52).
- 7.22** Beginning with a series expansion of the expansion functions defined in Eq. (7-56), derive an expression for the discrete Fourier transform.
- 7.23** Given standard basis vectors $\mathbf{e}_0 = [1 \ 0]^T$ and $\mathbf{e}_1 = [0 \ 1]^T$ of inner product space \mathbf{R}^2 and an arbitrary vector \mathbf{r} of length r and angle θ , compute the single-point cross-correlation of \mathbf{r} with both \mathbf{e}_0 and \mathbf{e}_1 . When does \mathbf{r} resemble \mathbf{e}_0 more than \mathbf{e}_1 and vice versa?
- 7.24*** Prove that the Fourier transform of time-scaled wavelet $\psi(2^s t)$ is given by Eq. (7-73).
- 7.25** Prove Eq. (7-80).
- 7.26** Obtain the Hartley transformation matrix for $N = 4$.
- 7.27** Write a pair of discrete cosine transform equations of the form given in Eqs. (7-57) and (7-58) for the discrete Fourier transform.
- 7.28** Because the 2-D discrete cosine transform is separable, the 2-D DCT of an image can be computed by row and column passes with a 1-D DCT algorithm. In fact, an interesting property of the 1-D DCT is that it can be computed by using the FFT algorithm. Show in detail how this computation can be made.
- 7.29** Do the following:
- Compute the Fourier, sine, cosine and Hartley transformation matrices of size $N = 6$.
 - Compute the Hartley transform of the discrete function $f(x) = \{-2, -5, -3, 1, 0, 3\}$ using Eq. (7-28).
 - Compute the Hartley transform of the function in (b) from its discrete Fourier transform.
- (d)** Is it equal to the result in (b)?
- 7.30** Use Eqs. (7-86) through (7-89) to compute the DCT of $f(x) = [3, -6, 1]$.
- 7.31** Use Eq. (7-28) to compute the DST of the function in (b).
- 7.32** Compute the basis images of the Haar transform for $N = 2$.
- 7.33** Create a table mapping the rows of Hadamard-ordered transformation matrix \mathbf{H}_{16} to sequency-ordered Hadamard transformation matrix \mathbf{H}'_{16} .
- 7.34** Obtain the slant transformation matrix for $N = 8$.
- 7.35** Derive the Haar scaling coefficients from Eqs. (7-122) and (7-126).
- 7.36** Show that scaling function
- $$\varphi(x) = \begin{cases} 1 & 0.25 \leq x < 0.75 \\ 0 & \text{elsewhere} \end{cases}$$
- does not satisfy the second requirement of a multiresolution analysis.
- 7.37*** Draw wavelet $\psi_{3,3}(x)$ for the Haar wavelet function. Write an expression for $\psi_{3,3}(x)$ in terms of Haar scaling functions.
- 7.38** Suppose function $f(x)$ is a member of Haar scaling space V_3 —that is, $f(x) \in V_3$. Use Eq. (7-128) to express V_3 as a function of scaling space V_0 and any required wavelet spaces. If $f(x)$ is 0 outside the interval $[0, 1]$, sketch the scaling and wavelet functions required for a linear expansion of $f(x)$ based on your expression.
- 7.39** Compute the first four terms of the wavelet series expansion of the function used in Example 7.18 with starting scale $j_0 = 1$. Write the resulting expansion in terms of the scaling and wavelet functions involved. How does your result compare to the example, where the starting scale was $j_0 = 0$?
- 7.40** The DWT in Eqs. (7-137) and (7-138) is for a starting scale $j_0 = 0$.

- (a)* Rewrite these equations for any starting scale $j_0 \leq J$.
- (b) Recompute the 1-D DWT of function $f(x) = \{1, 4, -3, 0\}$ for $0 \leq x \leq 3$ in Example 7.19 with $j_0 = 1$ (rather than 0).
- (c) Use the result from (b) to compute $f(1)$ from the transform values.

7.41* Draw the FWT filter bank required to compute the transform in Problem 7.40. Label all inputs and outputs with the appropriate sequences.

7.42 The computational complexity of an N -point fast wavelet transform is $O(N)$. That is, the number of operations is proportional N . What determines the constant of proportionality?

7.43 Answer the following:

- (a)* If the input to the three-scale FWT filter bank of Fig. 7.24(a) is the Haar scaling function $\varphi(x) = 1$ for $n = 0, 1, \dots, 7$ and 0 elsewhere, what is the resulting transform with respect to Haar wavelets?
- (b) What is the transform if the input is the corresponding Haar wavelet function $\psi(x) = \{1, 1, 1, -1, -1, -1, -1\}$ for $n = 0, 1, \dots, 7$?
- (c) What input sequence produces transform $\{1, 0, 0, 0, 0, 0, B, 0\}$ with nonzero coefficient $T_\psi(2, 2) = B$?

7.44 Compute the 2-D wavelet transform with respect to Haar wavelets of the 2×2 image

$$\begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix}$$

Draw the required filter bank and label all inputs and outputs with the proper arrays.

7.45* In the Fourier domain

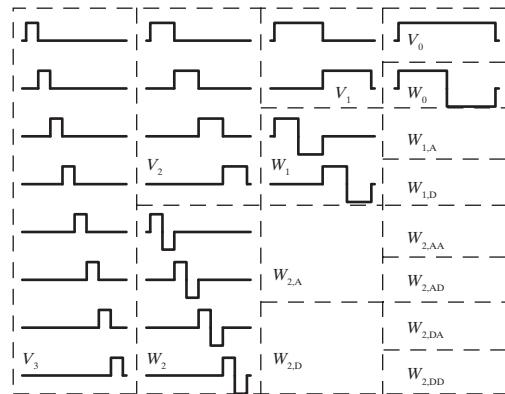
$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v)e^{-2\pi i(ux_0/M + vy_0/N)}$$

and translation does not affect the display of $|F(u, v)|$. Using the following sequence of images, explain the translation property of wavelet transforms. The top left image contains two 32×32 white squares centered on a 128×128 gray background. The top right image is its single-scale wavelet transform with respect to Haar wavelets.

The bottom left image is the wavelet transform of the original image after shifting its 32 pixels to the right and downward, and the final (bottom right) image is the wavelet transform of the original image after it has been shifted one pixel to the right and downward.



7.46 The following table shows the Haar wavelet and scaling functions for a four-scale fast wavelet transform. Sketch the additional basis functions needed for a full three-scale packet decomposition. Give the mathematical expression or expressions for determining them. Then order the basis functions according to frequency content and explain the results.

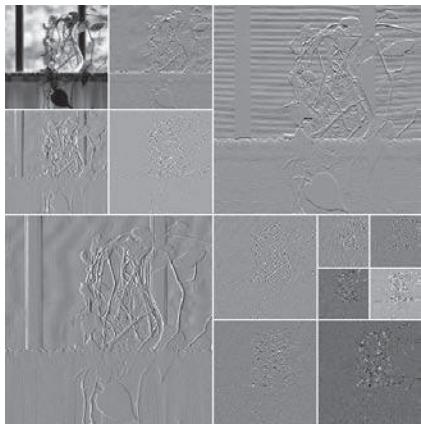


7.47 A wavelet packet decomposition of the vase from Fig. 7.30(a) is shown below.

- (a) Draw the corresponding decomposition anal-

ysis tree, labeling all nodes with the names of the proper scaling and wavelet spaces.

- (b) Draw and label the decomposition's frequency spectrum.



- 7.48** Using the Haar wavelet, determine the minimum entropy packet decomposition for the function for $f(x) = 0.25$ for $n = 0, 1, \dots, 15$. Employ the nonnormalized Shannon entropy

$$E[f(x)] = \sum_x f^2(x) \ln[f^2(x)]$$

as the minimization criterion. Draw the optimal tree, labeling the nodes with the computed entropy values.

8

Image Compression and Watermarking

But life is short and information endless ... Abbreviation is a necessary evil and the abbreviator's business is to make the best of a job which, although bad, is still better than nothing.

Aldous Huxley

The Titanic will protect itself.

Robert Ballard

Preview

Image compression, the art and science of reducing the amount of data required to represent an image, is one of the most useful and commercially successful technologies in the field of digital image processing. The number of images that are compressed and decompressed daily is staggering, and the compressions and decompressions themselves are virtually invisible to the user. Everyone who owns a digital camera, surfs the web, or streams the latest Hollywood movies over the Internet benefits from the algorithms and standards that will be discussed in this chapter. The material, which is largely introductory in nature, is applicable to both still-image and video applications. We will introduce both theory and practice, examining the most frequently used compression techniques, and describing the industry standards that make them useful. The chapter concludes with an introduction to *digital image watermarking*, the process of inserting visible and invisible data (such as copyright information) into images.

Upon completion of this chapter, students should:

- Be able to measure the amount of information in a digital image.
- Understand the main sources of data redundancy in digital images.
- Know the difference between lossy and error-free compression, and the amount of compression that is possible with each.
- Be familiar with the popular image compression standards, such as JPEG and JPEG-2000, that are in use today.
- Understand the principal image compression methods, and how and why they work.
- Be able to compress and decompress grayscale, color, and video imagery.
- Know the difference between visible, invisible, robust, fragile, public, private, restricted-key, and unrestricted-key watermarks.
- Understand the basics of watermark insertion and extraction in both the spatial and transform domain.