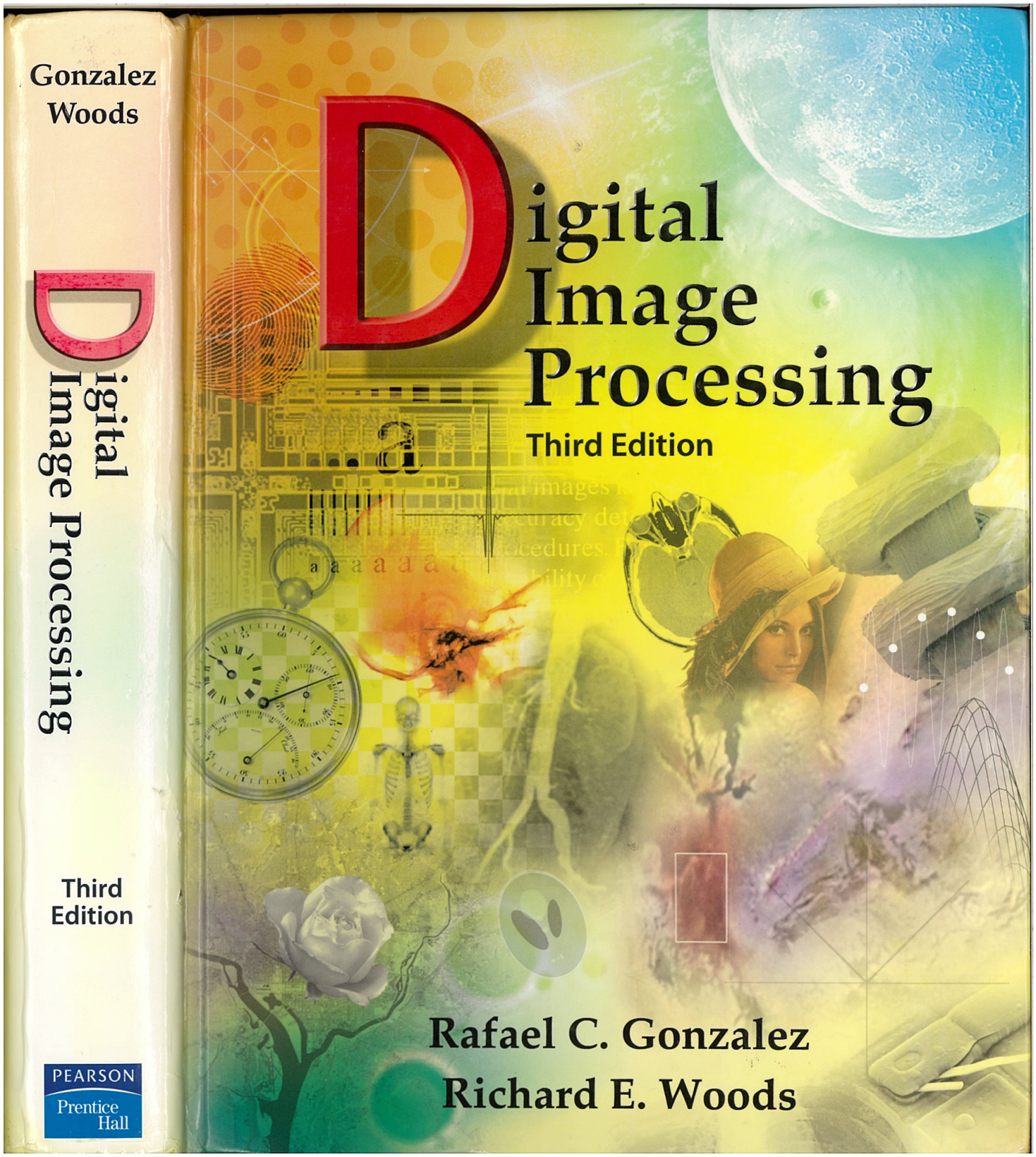


INTEL-1020
10,142,659

7949298576



Digital Image Processing

Third Edition

Rafael C. Gonzalez **Richard E. Woods**

Digital Image Processing has been the leading textbook in its field for more than thirty years. As the 1977 and 1987 editions by Gonzalez and Wintz, and the 1992 and 2002 editions by Gonzalez and Woods, the present edition was prepared with students and instructors in mind. The changes and upgrades resulting in this edition are based on a survey involving faculty, students, and independent readers of the book in 134 institutions from 32 countries. The new material is timely, highly readable, and illustrated with numerous examples of practical significance. Areas of image processing covered include intensity transformations and spatial filtering, the Fourier transform and filtering in the frequency domain, image enhancement, image restoration, computed tomography, fuzzy sets, color image processing, wavelets, image and video compression, morphological processing, image segmentation, and image description. Coverage concludes with a discussion on the fundamentals of object recognition.

New Features

- A revision of introductory concepts that provides readers with foundation material much earlier in the book than before.
- A revised and updated discussion of intensity transformation, spatial correlation, convolution, and their application to spatial filtering.
- New discussion of fuzzy sets and their application to image processing.
- A new chapter on the discrete Fourier transform and frequency domain processing.
- New coverage of computerized tomography.
- A major revision of the wavelets chapter.
- A new chapter on data compression, including new compression techniques, digital video compression, standards, and watermarking.
- New coverage of morphological reconstruction, gray-scale morphology, and advanced morphological algorithms.
- New coverage of the Marr-Hildreth and Canny edge detection algorithms.
- Expanded coverage of image thresholding.
- New examples and illustrations involving over 400 new images and more than 200 new drawings and tables.
- Expanded homework sets, including over 80 new problems.
- Updated bibliography.

Book Web Site: www.prenhall.com/gonzalezwoods

Although the book is completely self-contained, a newly redesigned companion web site provides additional support in the form of review material, answers to selected problems, laboratory project suggestions, classroom presentation slides, tutorials, and a score of other features. An Instructor's Manual is available to instructors who have adopted the book for classroom use.

ISBN-13: 978-0-13-168728-8
ISBN-10: 0-13-168728-X

EAN 9 780131 687288 90000

PEARSON Prentice Hall

Upper Saddle River, New Jersey 07458
www.prenhall.com

Digital Image Processing

Third Edition

Rafael C. Gonzalez
University of Tennessee

Richard E. Woods
MedData Interactive

PEARSON
Prentice Hall

Upper Saddle River, NJ 07458

Library of Congress Cataloging-in-Publication Data on File

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Executive Editor: *Michael McDonald*
Associate Editor: *Alice Dworkin*
Editorial Assistant: *William Opaluch*
Managing Editor: *Scott Disanno*
Production Editor: *Rose Kernan*
Director of Creative Services: *Paul Belfanti*
Creative Director: *Juan Lopez*
Art Director: *Heather Scott*
Art Editors: *Gregory Dulles* and *Thomas Benfatti*
Manufacturing Manager: *Alexis Heydt-Long*
Manufacturing Buyer: *Lisa McDowell*
Senior Marketing Manager: *Tim Galligan*



© 2008 by Pearson Education, Inc.
Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, New Jersey 07458

All rights reserved. No part of this book may be reproduced, in any form, or by any means, without permission in writing from the publisher.

Pearson Prentice Hall® is a trademark of Pearson Education, Inc.

The authors and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The authors and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America.
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-168728-x
978-0-13-168728-8

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Ltd., *Sydney*
Pearson Education Singapore, Pte., Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada, Inc., *Toronto*
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., *Upper Saddle River, New Jersey*

Contents

Preface xv
Acknowledgments xix
The Book Web Site xx
About the Authors xxi

1 Introduction 1

1.1 What Is Digital Image Processing? 1
1.2 The Origins of Digital Image Processing 3
1.3 Examples of Fields that Use Digital Image Processing 7
1.3.1 Gamma-Ray Imaging 8
1.3.2 X-Ray Imaging 9
1.3.3 Imaging in the Ultraviolet Band 11
1.3.4 Imaging in the Visible and Infrared Bands 12
1.3.5 Imaging in the Microwave Band 18
1.3.6 Imaging in the Radio Band 20
1.3.7 Examples in which Other Imaging Modalities Are Used 20
1.4 Fundamental Steps in Digital Image Processing 25
1.5 Components of an Image Processing System 28
Summary 31
References and Further Reading 31

2 Digital Image Fundamentals 35

2.1 Elements of Visual Perception 36
2.1.1 Structure of the Human Eye 36
2.1.2 Image Formation in the Eye 38
2.1.3 Brightness Adaptation and Discrimination 39
2.2 Light and the Electromagnetic Spectrum 43
2.3 Image Sensing and Acquisition 46
2.3.1 Image Acquisition Using a Single Sensor 48
2.3.2 Image Acquisition Using Sensor Strips 48
2.3.3 Image Acquisition Using Sensor Arrays 50
2.3.4 A Simple Image Formation Model 50
2.4 Image Sampling and Quantization 52
2.4.1 Basic Concepts in Sampling and Quantization 52
2.4.2 Representing Digital Images 55
2.4.3 Spatial and Intensity Resolution 59
2.4.4 Image Interpolation 65

2.5 Some Basic Relationships between Pixels 68
2.5.1 Neighbors of a Pixel 68
2.5.2 Adjacency, Connectivity, Regions, and Boundaries 68
2.5.3 Distance Measures 71
2.6 An Introduction to the Mathematical Tools Used in Digital Image Processing 72
2.6.1 Array versus Matrix Operations 72
2.6.2 Linear versus Nonlinear Operations 73
2.6.3 Arithmetic Operations 74
2.6.4 Set and Logical Operations 80
2.6.5 Spatial Operations 85
2.6.6 Vector and Matrix Operations 92
2.6.7 Image Transforms 93
2.6.8 Probabilistic Methods 96
Summary 98
References and Further Reading 98
Problems 99

3 *Intensity Transformations and Spatial Filtering* 104

3.1 Background 105
3.1.1 The Basics of Intensity Transformations and Spatial Filtering 105
3.1.2 About the Examples in This Chapter 107
3.2 Some Basic Intensity Transformation Functions 107
3.2.1 Image Negatives 108
3.2.2 Log Transformations 109
3.2.3 Power-Law (Gamma) Transformations 110
3.2.4 Piecewise-Linear Transformation Functions 115
3.3 Histogram Processing 120
3.3.1 Histogram Equalization 122
3.3.2 Histogram Matching (Specification) 128
3.3.3 Local Histogram Processing 139
3.3.4 Using Histogram Statistics for Image Enhancement 139
3.4 Fundamentals of Spatial Filtering 144
3.4.1 The Mechanics of Spatial Filtering 145
3.4.2 Spatial Correlation and Convolution 146
3.4.3 Vector Representation of Linear Filtering 150
3.4.4 Generating Spatial Filter Masks 151
3.5 Smoothing Spatial Filters 152
3.5.1 Smoothing Linear Filters 152
3.5.2 Order-Statistic (Nonlinear) Filters 156
3.6 Sharpening Spatial Filters 157
3.6.1 Foundation 158
3.6.2 Using the Second Derivative for Image Sharpening—The Laplacian 160

3.6.3 Unsharp Masking and Highboost Filtering 162
3.6.4 Using First-Order Derivatives for (Nonlinear) Image Sharpening—The Gradient 165
3.7 Combining Spatial Enhancement Methods 169
3.8 Using Fuzzy Techniques for Intensity Transformations and Spatial Filtering 173
3.8.1 Introduction 173
3.8.2 Principles of Fuzzy Set Theory 174
3.8.3 Using Fuzzy Sets 178
3.8.4 Using Fuzzy Sets for Intensity Transformations 186
3.8.5 Using Fuzzy Sets for Spatial Filtering 189
Summary 192
References and Further Reading 192
Problems 193

4 *Filtering in the Frequency Domain* 199
4.1 Background 200
4.1.1 A Brief History of the Fourier Series and Transform 200
4.1.2 About the Examples in this Chapter 201
4.2 Preliminary Concepts 202
4.2.1 Complex Numbers 202
4.2.2 Fourier Series 203
4.2.3 Impulses and Their Sifting Property 203
4.2.4 The Fourier Transform of Functions of One Continuous Variable 205
4.2.5 Convolution 209
4.3 Sampling and the Fourier Transform of Sampled Functions 211
4.3.1 Sampling 211
4.3.2 The Fourier Transform of Sampled Functions 212
4.3.3 The Sampling Theorem 213
4.3.4 Aliasing 217
4.3.5 Function Reconstruction (Recovery) from Sampled Data 219
4.4 The Discrete Fourier Transform (DFT) of One Variable 220
4.4.1 Obtaining the DFT from the Continuous Transform of a Sampled Function 221
4.4.2 Relationship Between the Sampling and Frequency Intervals 223
4.5 Extension to Functions of Two Variables 225
4.5.1 The 2-D Impulse and Its Sifting Property 225
4.5.2 The 2-D Continuous Fourier Transform Pair 226
4.5.3 Two-Dimensional Sampling and the 2-D Sampling Theorem 227
4.5.4 Aliasing in Images 228
4.5.5 The 2-D Discrete Fourier Transform and Its Inverse 235

4.6 Some Properties of the 2-D Discrete Fourier Transform 236
4.6.1 Relationships Between Spatial and Frequency Intervals 236
4.6.2 Translation and Rotation 236
4.6.3 Periodicity 237
4.6.4 Symmetry Properties 239
4.6.5 Fourier Spectrum and Phase Angle 245
4.6.6 The 2-D Convolution Theorem 249
4.6.7 Summary of 2-D Discrete Fourier Transform Properties 253
4.7 The Basics of Filtering in the Frequency Domain 255
4.7.1 Additional Characteristics of the Frequency Domain 255
4.7.2 Frequency Domain Filtering Fundamentals 257
4.7.3 Summary of Steps for Filtering in the Frequency Domain 263
4.7.4 Correspondence Between Filtering in the Spatial and Frequency Domains 263
4.8 Image Smoothing Using Frequency Domain Filters 269
4.8.1 Ideal Lowpass Filters 269
4.8.2 Butterworth Lowpass Filters 273
4.8.3 Gaussian Lowpass Filters 276
4.8.4 Additional Examples of Lowpass Filtering 277
4.9 Image Sharpening Using Frequency Domain Filters 280
4.9.1 Ideal Highpass Filters 281
4.9.2 Butterworth Highpass Filters 284
4.9.3 Gaussian Highpass Filters 285
4.9.4 The Laplacian in the Frequency Domain 286
4.9.5 Unsharp Masking, Highboost Filtering, and High-Frequency-Emphasis Filtering 288
4.9.6 Homomorphic Filtering 289
4.10 Selective Filtering 294
4.10.1 Bandreject and Bandpass Filters 294
4.10.2 Notch Filters 294
4.11 Implementation 298
4.11.1 Separability of the 2-D DFT 298
4.11.2 Computing the IDFT Using a DFT Algorithm 299
4.11.3 The Fast Fourier Transform (FFT) 299
4.11.4 Some Comments on Filter Design 303
Summary 303
References and Further Reading 304
Problems 304

5 *Image Restoration and Reconstruction* 311
5.1 A Model of the Image Degradation/Restoration Process 312
5.2 Noise Models 313
5.2.1 Spatial and Frequency Properties of Noise 313
5.2.2 Some Important Noise Probability Density Functions 314

5.2.3 Periodic Noise 318
5.2.4 Estimation of Noise Parameters 319
5.3 Restoration in the Presence of Noise Only—Spatial Filtering 322
5.3.1 Mean Filters 322
5.3.2 Order-Statistic Filters 325
5.3.3 Adaptive Filters 330
5.4 Periodic Noise Reduction by Frequency Domain Filtering 335
5.4.1 Bandreject Filters 335
5.4.2 Bandpass Filters 336
5.4.3 Notch Filters 337
5.4.4 Optimum Notch Filtering 338
5.5 Linear, Position-Invariant Degradations 343
5.6 Estimating the Degradation Function 346
5.6.1 Estimation by Image Observation 346
5.6.2 Estimation by Experimentation 347
5.6.3 Estimation by Modeling 347
5.7 Inverse Filtering 351
5.8 Minimum Mean Square Error (Wiener) Filtering 352
5.9 Constrained Least Squares Filtering 357
5.10 Geometric Mean Filter 361
5.11 Image Reconstruction from Projections 362
5.11.1 Introduction 362
5.11.2 Principles of Computed Tomography (CT) 365
5.11.3 Projections and the Radon Transform 368
5.11.4 The Fourier-Slice Theorem 374
5.11.5 Reconstruction Using Parallel-Beam Filtered Backprojections 375
5.11.6 Reconstruction Using Fan-Beam Filtered Backprojections 381
Summary 387
References and Further Reading 388
Problems 389

6 *Color Image Processing* 394

6.1 Color Fundamentals 395
6.2 Color Models 401
6.2.1 The RGB Color Model 402
6.2.2 The CMY and CMYK Color Models 406
6.2.3 The HSI Color Model 407
6.3 Pseudocolor Image Processing 414
6.3.1 Intensity Slicing 415
6.3.2 Intensity to Color Transformations 418
6.4 Basics of Full-Color Image Processing 424
6.5 Color Transformations 426
6.5.1 Formulation 426
6.5.2 Color Complements 430

6.5.3 Color Slicing 431
6.5.4 Tone and Color Corrections 433
6.5.5 Histogram Processing 438
6.6 Smoothing and Sharpening 439
6.6.1 Color Image Smoothing 439
6.6.2 Color Image Sharpening 442
6.7 Image Segmentation Based on Color 443
6.7.1 Segmentation in HSI Color Space 443
6.7.2 Segmentation in RGB Vector Space 445
6.7.3 Color Edge Detection 447
6.8 Noise in Color Images 451
6.9 Color Image Compression 454
Summary 455
References and Further Reading 456
Problems 456

7 *Wavelets and Multiresolution Processing* 461

7.1 Background 462
7.1.1 Image Pyramids 463
7.1.2 Subband Coding 466
7.1.3 The Haar Transform 474
7.2 Multiresolution Expansions 477
7.2.1 Series Expansions 477
7.2.2 Scaling Functions 479
7.2.3 Wavelet Functions 483
7.3 Wavelet Transforms in One Dimension 486
7.3.1 The Wavelet Series Expansions 486
7.3.2 The Discrete Wavelet Transform 488
7.3.3 The Continuous Wavelet Transform 491
7.4 The Fast Wavelet Transform 493
7.5 Wavelet Transforms in Two Dimensions 501
7.6 Wavelet Packets 510
Summary 520
References and Further Reading 520
Problems 521

8 *Image Compression* 525

8.1 Fundamentals 526
8.1.1 Coding Redundancy 528
8.1.2 Spatial and Temporal Redundancy 529
8.1.3 Irrelevant Information 530
8.1.4 Measuring Image Information 531
8.1.5 Fidelity Criteria 534

8.1.6 Image Compression Models 536
8.1.7 Image Formats, Containers, and Compression Standards 538
8.2 **Some Basic Compression Methods 542**
8.2.1 Huffman Coding 542
8.2.2 Golomb Coding 544
8.2.3 Arithmetic Coding 548
8.2.4 LZW Coding 551
8.2.5 Run-Length Coding 553
8.2.6 Symbol-Based Coding 559
8.2.7 Bit-Plane Coding 562
8.2.8 Block Transform Coding 566
8.2.9 Predictive Coding 584
8.2.10 Wavelet Coding 604
8.3 **Digital Image Watermarking 614**
Summary 621
References and Further Reading 622
Problems 623

9 *Morphological Image Processing 627*

9.1 **Preliminaries 628**
9.2 **Erosion and Dilation 630**
9.2.1 Erosion 631
9.2.2 Dilation 633
9.2.3 Duality 635
9.3 **Opening and Closing 635**
9.4 **The Hit-or-Miss Transformation 640**
9.5 **Some Basic Morphological Algorithms 642**
9.5.1 Boundary Extraction 642
9.5.2 Hole Filling 643
9.5.3 Extraction of Connected Components 645
9.5.4 Convex Hull 647
9.5.5 Thinning 649
9.5.6 Thickening 650
9.5.7 Skeletons 651
9.5.8 Pruning 654
9.5.9 Morphological Reconstruction 656
9.5.10 Summary of Morphological Operations on Binary Images 664
9.6 **Gray-Scale Morphology 665**
9.6.1 Erosion and Dilation 666
9.6.2 Opening and Closing 668
9.6.3 Some Basic Gray-Scale Morphological Algorithms 670
9.6.4 Gray-Scale Morphological Reconstruction 676
Summary 679
References and Further Reading 679
Problems 680

10 *Image Segmentation* 689

10.1 Fundamentals 690
10.2 Point, Line, and Edge Detection 692
10.2.1 Background 692
10.2.2 Detection of Isolated Points 696
10.2.3 Line Detection 697
10.2.4 Edge Models 700
10.2.5 Basic Edge Detection 706
10.2.6 More Advanced Techniques for Edge Detection 714
10.2.7 Edge Linking and Boundary Detection 725
10.3 Thresholding 738
10.3.1 Foundation 738
10.3.2 Basic Global Thresholding 741
10.3.3 Optimum Global Thresholding Using Otsu's Method 742
10.3.4 Using Image Smoothing to Improve Global Thresholding 747
10.3.5 Using Edges to Improve Global Thresholding 749
10.3.6 Multiple Thresholds 752
10.3.7 Variable Thresholding 756
10.3.8 Multivariable Thresholding 761
10.4 Region-Based Segmentation 763
10.4.1 Region Growing 763
10.4.2 Region Splitting and Merging 766
10.5 Segmentation Using Morphological Watersheds 769
10.5.1 Background 769
10.5.2 Dam Construction 772
10.5.3 Watershed Segmentation Algorithm 774
10.5.4 The Use of Markers 776
10.6 The Use of Motion in Segmentation 778
10.6.1 Spatial Techniques 778
10.6.2 Frequency Domain Techniques 782
Summary 785
References and Further Reading 785
Problems 787

11 *Representation and Description* 795

11.1 Representation 796
11.1.1 Boundary (Border) Following 796
11.1.2 Chain Codes 798
11.1.3 Polygonal Approximations Using Minimum-Perimeter Polygons 801
11.1.4 Other Polygonal Approximation Approaches 807
11.1.5 Signatures 808

11.1.6 Boundary Segments 810
11.1.7 Skeletons 812
11.2 Boundary Descriptors 815
11.2.1 Some Simple Descriptors 815
11.2.2 Shape Numbers 816
11.2.3 Fourier Descriptors 818
11.2.4 Statistical Moments 821
11.3 Regional Descriptors 822
11.3.1 Some Simple Descriptors 822
11.3.2 Topological Descriptors 823
11.3.3 Texture 827
11.3.4 Moment Invariants 839
11.4 Use of Principal Components for Description 842
11.5 Relational Descriptors 852
Summary 856
References and Further Reading 856
Problems 857

12 Object Recognition 861

12.1 Patterns and Pattern Classes 861
12.2 Recognition Based on Decision-Theoretic Methods 866
12.2.1 Matching 866
12.2.2 Optimum Statistical Classifiers 872
12.2.3 Neural Networks 882
12.3 Structural Methods 903
12.3.1 Matching Shape Numbers 903
12.3.2 String Matching 904
Summary 906
References and Further Reading 906
Problems 907

Appendix A 910

Bibliography 915

Index 943

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 introduced in Section 2.6.7 and discussed in more detail in Chapter 4, 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. As you will learn in this chapter, intensity transformations operate on single pixels of an image, principally for the purpose of contrast manipulation and image thresholding. Spatial filtering deals with performing operations, such as image sharpening, by working in a neighborhood of every pixel in an image. In the sections that follow, we discuss a number of "classical" techniques for intensity transformations and spatial filtering. We also discuss in some detail fuzzy techniques that allow us to incorporate imprecise, knowledge-based information in the formulation of intensity transformations and spatial filtering algorithms.

3.1 Background

3.1.1 The Basics of Intensity Transformations and Spatial Filtering

All the image processing techniques discussed in this section are implemented in the *spatial domain*, which we know from the discussion in Section 2.4.2 is simply the plane containing the pixels of an image. As noted in Section 2.6.7, 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. Generally, spatial domain techniques are more efficient computationally and require less processing resources to implement.

The spatial domain processes we discuss in this chapter can be denoted by the expression

$$g(x, y) = T[f(x, y)] \tag{3.1-1}$$

where $f(x, y)$ is the 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 apply to a single image (our principal focus in this chapter) or to a set of images, such as performing the pixel-by-pixel sum of a sequence of images for noise reduction, as discussed in Section 2.6.3. Figure 3.1 shows the basic implementation of Eq. (3.1-1) on a single image. The point (x, y) shown is an arbitrary location in the image, and the small region shown containing the point is a neighborhood of (x, y), as explained in Section 2.6.5. Typically, the neighborhood is rectangular, centered on (x, y), and much smaller in size than the image.

Other neighborhood shapes, such as digital approximations to circles, are used sometimes, but rectangular shapes are by far the most prevalent because they are much easier to implement computationally.

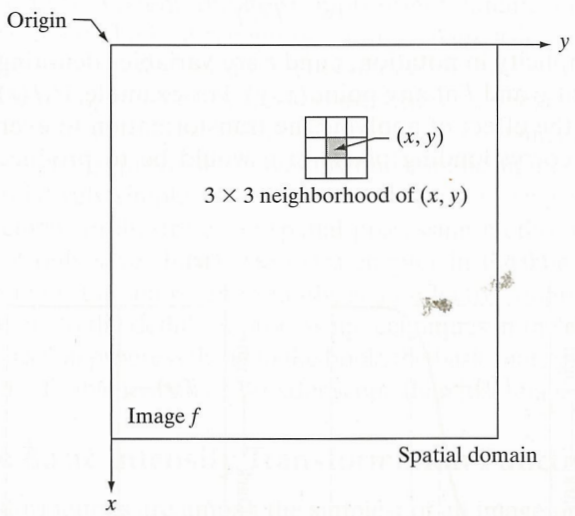


FIGURE 3.1 A 3 × 3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

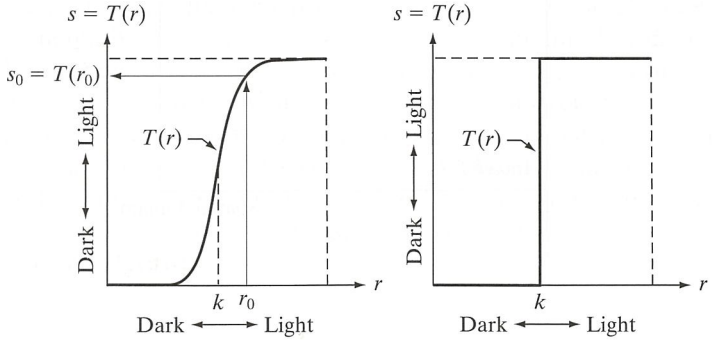
The process that Fig. 3.1 illustrates consists of moving the origin of the neighborhood from pixel to pixel and applying the operator T to the pixels in the neighborhood to yield the output at that location. Thus, for any specific location (x, y), the value of the output image g at those coordinates is equal to the result of applying T to the neighborhood with origin at (x, y) 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 neighborhood." Consider an arbitrary location in an image, say (100, 150). Assuming that the origin of the neighborhood is at its center, the result, $g(100, 150)$, at that location is computed as the sum of $f(100, 150)$ and its 8-neighbors, divided by 9 (i.e., the average intensity of the pixels encompassed by the neighborhood). The origin of the neighborhood is then moved to the next 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 scan, one row at a time. When the origin of the neighborhood is at the border of the image, part of the neighborhood will reside outside the image. The procedure is either to ignore the outside neighbors in the computations specified by T, or to pad the image with a border of 0s or some other specified intensity values. The thickness of the padded border depends on the size of the neighborhood. We will return to this issue in Section 3.4.1.

As we discuss in detail in Section 3.4, the procedure just described is called *spatial filtering*, in which the neighborhood, along with a predefined operation, is called a *spatial filter* (also referred to as a *spatial mask*, *kernel*, *template*, or *window*). The type of operation performed in the neighborhood determines the nature of the filtering process.

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. (3.1-1) becomes an *intensity* (also called *gray-level* or *mapping*) *transformation function* of the form

$$s = T(r) \tag{3.1-2}$$

where, for simplicity in notation, s and r are variables denoting, 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 effect of applying the transformation to every pixel of f to generate the corresponding pixels in g would be to produce an image of



a b

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

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 3.2.4), values of r lower than k are compressed by the transformation function into a narrow range 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 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 earlier in this section.

3.1.2 About the Examples in This Chapter

Although intensity transformations and spatial filtering 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 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 method works. When dealing with machine perception, a given technique 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 over another.

Regardless of the application or method used, however, image enhancement is one of the most visually appealing areas of image processing. By its very nature, 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 the details of processing techniques in the spatial domain. As you will see as you progress through the book, the basic material developed in this chapter is applicable to a much broader scope than just image enhancement.

3.2 Some Basic Intensity Transformation Functions

Intensity transformations are among the simplest of all image processing techniques. The values of pixels, before and after processing, will be denoted by r and s, respectively. As indicated in the previous section, these values are related

We summarize the preceding approach as follows. Let $f(x, y)$ represent the value of an image at any image coordinates (x, y), and let $g(x, y)$ represent the corresponding enhanced value at those coordinates. Then,

$$g(x, y) = \begin{cases} E \cdot f(x, y) & \text{if } m_{S_{xy}} \leq k_0 m_G \text{ AND } k_1 \sigma_G \leq \sigma_{S_{xy}} \leq k_2 \sigma_G \\ f(x, y) & \text{otherwise} \end{cases} \tag{3.3-24}$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$, where, as indicated above, E, k_0, k_1, and k_2 are specified parameters, 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. As usual, M and N are the row and column image dimensions.

Choosing the parameters in Eq. (3.3-24) generally requires a bit of experimentation to gain familiarity with a given image or class of images. In this case, the following values were selected: $E = 4.0$, $k_0 = 0.4$, $k_1 = 0.02$, and $k_2 = 0.4$. The relatively low value of 4.0 for E was chosen so that, when it was multiplied by the levels in the areas being enhanced (which are dark), the result would still tend toward the dark end of the scale, and thus preserve the general visual balance of the image. The value of k_0 was chosen as less than half the global mean because we can see by looking at the image that the areas that require enhancement definitely are dark enough to be below half the global mean. A similar analysis led to the choice of values for k_1 and k_2. Choosing these constants is not difficult in general, but their choice definitely must be guided by a logical analysis of the enhancement problem at hand. Finally, the size of the local area S_{xy} should be as small as possible in order to preserve detail and keep the computational burden as low as possible. We chose a region of size 3×3.

As a basis for comparison, we enhanced the image using global histogram equalization. Figure 3.27(b) shows the result. The dark area was improved but details still are difficult to discern, and the light areas were changed, something we did not want to do. Figure 3.27(c) shows the result of using the local statistics method explained above. In comparing this image with the original in Fig. 3.27(a) or the histogram equalized result in Fig. 3.27(b), we note the obvious detail that has been brought out on the right side of Fig. 3.27(c). Observe, for example, the clarity of the ridges in the dark filaments. It is noteworthy that the light-intensity areas on the left were left nearly intact, which was one of our initial objectives. ■

3.4 Fundamentals of Spatial Filtering

In this section, we introduce several basic concepts underlying the use of spatial filters for image processing. Spatial filtering is one of the principal tools used in this field for a broad spectrum of applications, so it is highly advisable that you develop a solid understanding of these concepts. As mentioned at the beginning of this chapter, the examples in this section deal mostly with the use of spatial filters for image enhancement. Other applications of spatial filtering are discussed in later chapters.

The name *filter* is borrowed from frequency domain processing, which is the topic of the next chapter, where "filtering" refers to accepting (passing) or rejecting certain frequency components. For example, a filter that passes low frequencies is called a *lowpass* filter. The net effect produced by a lowpass filter is to blur (smooth) an image. We can accomplish a similar smoothing directly on the image itself by using spatial filters (also called spatial *masks*, *kernels*, *templates*, and *windows*). In fact, as we show in Chapter 4, there is a one-to-one correspondence between linear spatial filters and filters in the frequency domain. However, spatial filters offer considerably more versatility because, as you will see later, they can be used also for nonlinear filtering, something we cannot do in the frequency domain.

See Section 2.6.2 regarding linearity.

3.4.1 The Mechanics of Spatial Filtering

In Fig. 3.1, we explained briefly that a spatial filter consists of (1) a *neighborhood*, (typically a small rectangle), and (2) a *predefined operation* that is performed on the image pixels encompassed by the neighborhood. Filtering creates a new pixel with coordinates equal to the coordinates of the center of the neighborhood, and whose value is the result of the filtering operation.[†] A processed (filtered) image is generated as the center of the filter visits each pixel in the input image. If the operation performed on the image pixels is linear, then the filter is called a *linear spatial filter*. Otherwise, the filter is *nonlinear*. We focus attention first on linear filters and then illustrate some simple nonlinear filters. Section 5.3 contains a more comprehensive list of nonlinear filters and their application.

Figure 3.28 illustrates the mechanics of linear spatial filtering using a 3×3 neighborhood. At any point (x, y) in the image, the response, $g(x, y)$, of the filter is the sum of products of the filter coefficients and the image pixels encompassed by the filter:

$$g(x, y) = w(-1, -1)f(x - 1, y - 1) + w(-1, 0)f(x - 1, y) + \ldots + w(0, 0)f(x, y) + \ldots + w(1, 1)f(x + 1, y + 1)$$

Observe that the center coefficient of the filter, $w(0, 0)$, aligns with the pixel at location (x, y). For a mask of size $m \times n$, we assume that $m = 2a + 1$ and $n = 2b + 1$, where a and b are positive integers. This means that our focus in the following discussion is on filters of odd size, with the smallest being of size 3×3. In general, linear spatial filtering of an image of size $M \times N$ with a filter of size $m \times n$ is given by the expression:

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

where x and y are varied so that each pixel in w visits every pixel in f.

It certainly is possible to work with filters of even size or mixed even and odd sizes. However, working with odd sizes simplifies indexing and also is more intuitive because the filters have centers falling on integer values.

[†] The 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 still is being performed.

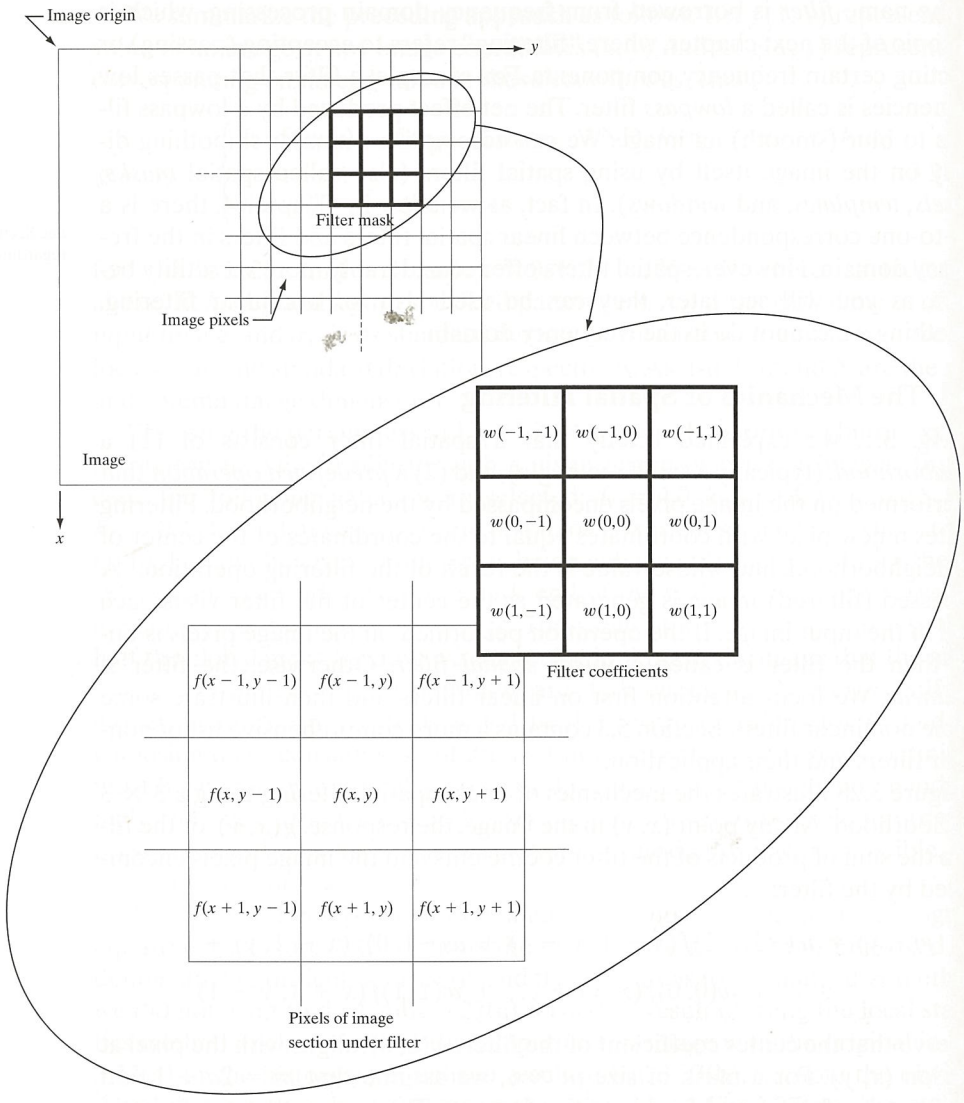


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

3.4.2 Spatial Correlation and Convolution

There are two closely related concepts that must be understood clearly when performing linear spatial filtering. One is *correlation* and the other is *convolution*. Correlation is the process of moving a filter mask over the image and computing the sum of products at each location, exactly as explained in the previous section. The mechanics of convolution are the same, except that the filter is first rotated by 180°. The best way to explain the differences between the two concepts is by example. We begin with a 1-D illustration.

Figure 3.29(a) shows a 1-D function, f, and a filter, w, and Fig. 3.29(b) shows the starting position to perform correlation. The first thing we note is that there

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 3.31 Another representation of a general 3 × 3 filter mask.

As an example, Fig. 3.31 shows a general 3 × 3 mask with coefficients labeled as above. In this case, Eq. (3.4-3) becomes

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 \\ &= \sum_{k=1}^{9} w_k z_k \\ &= \mathbf{w}^T \mathbf{z} \end{aligned} \tag{3.4-4}$$

where **w** and **z** are 9-dimensional vectors formed from the coefficients of the mask and the image intensities encompassed by the mask, respectively.

3.4.4 Generating Spatial Filter Masks

Generating an $m \times n$ *linear* spatial filter requires that we specify mn mask coefficients. In turn, these coefficients are selected based on what the filter is supposed to do, keeping in mind that all we can do with linear filtering is to implement a sum of products. For example, suppose that we want to replace the pixels in an image by the average intensity of a 3 × 3 neighborhood centered on those pixels. The average value at any location (x, y) in the image is the sum of the nine intensity values in the 3 × 3 neighborhood centered on (x, y) divided by 9. Letting z_i, $i = 1, 2, \ldots, 9$, denote these intensities, the average is

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

But this is the same as Eq. (3.4-4) with coefficient values $w_i = 1/9$. In other words, a linear filtering operation with a 3 × 3 mask whose coefficients are 1/9 implements the desired averaging. As we discuss in the next section, this operation results in image smoothing. We discuss in the following sections a number of other filter masks based on this basic approach.

In some applications, we have a continuous function of two variables, and the objective is to obtain a spatial filter mask based on that function. For example, a Gaussian function of two variables has the basic form

$$h(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

where σ is the standard deviation and, as usual, we assume that coordinates x and y are integers. To generate, say, a 3 × 3 filter mask from this function, we

sample it about its center. Thus, $w_1 = h(-1, -1), w_2 = h(-1, 0), \ldots, w_9 = h(1, 1)$. An $m \times n$ filter mask is generated in a similar manner. Recall that a 2-D Gaussian function has a bell shape, and that the standard deviation controls the "tightness" of the bell.

Generating a *nonlinear* filter requires that we specify the size of a neighborhood and the operation(s) to be performed on the image pixels contained in the neighborhood. For example, recalling that the max operation is nonlinear (see Section 2.6.2), a 5×5 max filter centered at an arbitrary point (x, y) of an image obtains the maximum intensity value of the 25 pixels and assigns that value to location (x, y) in the processed image. Nonlinear filters are quite powerful, and in some applications can perform functions that are beyond the capabilities of linear filters, as we show later in this chapter and in Chapter 5.

3.5 Smoothing Spatial Filters

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

3.5.1 Smoothing Linear Filters

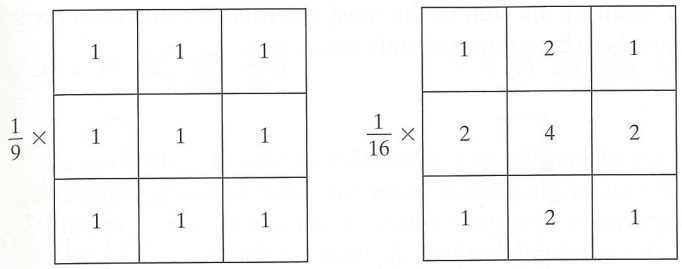
The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters sometimes are called *averaging filters*. As mentioned in the previous section, they also are referred to a *lowpass filters*.

The idea behind smoothing filters is straightforward. By replacing the value of every pixel in an image by the average of the intensity levels in the neighborhood defined by the filter mask, this process results in an image with reduced "sharp" transitions in intensities. Because random noise typically consists of sharp transitions in intensity levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp intensity transitions, so averaging filters have the undesirable side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result from using an insufficient number of intensity levels, as discussed in Section 2.4.3. A major use of averaging filters is in the reduction of "irrelevant" detail in an image. By "irrelevant" we mean pixel regions that are small with respect to the size of the filter mask. This latter application is illustrated later in this section.

Figure 3.32 shows two 3×3 smoothing filters. Use of the first filter yields the standard average of the pixels under the mask. This can best be seen by substituting the coefficients of the mask into Eq. (3.4-4):

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

which is the average of the intensity levels of the pixels in the 3×3 neighborhood defined by the mask, as discussed earlier. Note that, instead of being $1/9$,



a b

FIGURE 3.32 Two 3×3 smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to 1 divided by the sum of the values of its coefficients, as is required to compute an average.

the coefficients of the filter are all 1s. The idea here is that it is computationally more efficient to have coefficients valued 1. At the end of the filtering process the entire image is divided by 9. An $m \times n$ mask would have a normalizing constant equal to $1/mn$. A spatial averaging filter in which all coefficients are equal sometimes is called a *box filter*.

The second mask in Fig. 3.32 is a little more interesting. This mask yields a so-called *weighted average*, terminology used to indicate that pixels are multiplied by different coefficients, thus giving more importance (weight) to some pixels at the expense of others. In the mask shown in Fig. 3.32(b) the pixel at the center of the mask is multiplied by a higher value than any other, thus giving this pixel more importance in the calculation of the average. The other pixels are inversely weighted as a function of their distance from the center of the mask. The diagonal terms are further away from the center than the orthogonal neighbors (by a factor of $\sqrt{2}$) and, thus, are weighed less than the immediate neighbors of the center pixel. The basic strategy behind weighing the center point the highest and then reducing the value of the coefficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the smoothing process. We could have chosen other weights to accomplish the same general objective. However, the sum of all the coefficients in the mask of Fig. 3.32(b) is equal to 16, an attractive feature for computer implementation because it is an integer power of 2. In practice, it is difficult in general to see differences between images smoothed by using either of the masks in Fig. 3.32, or similar arrangements, because the area spanned by these masks at any one location in an image is so small.

With reference to Eq. (3.4-1), the general implementation for filtering an $M \times N$ image with a weighted averaging filter of size $m \times n$ (m and n odd) is given by the expression

$$g(x, y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)} \tag{3.5-1}$$

The parameters in this equation are as defined in Eq. (3.4-1). As before, it is understood that the complete filtered image is obtained by applying Eq. (3.5-1) for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. The denominator in

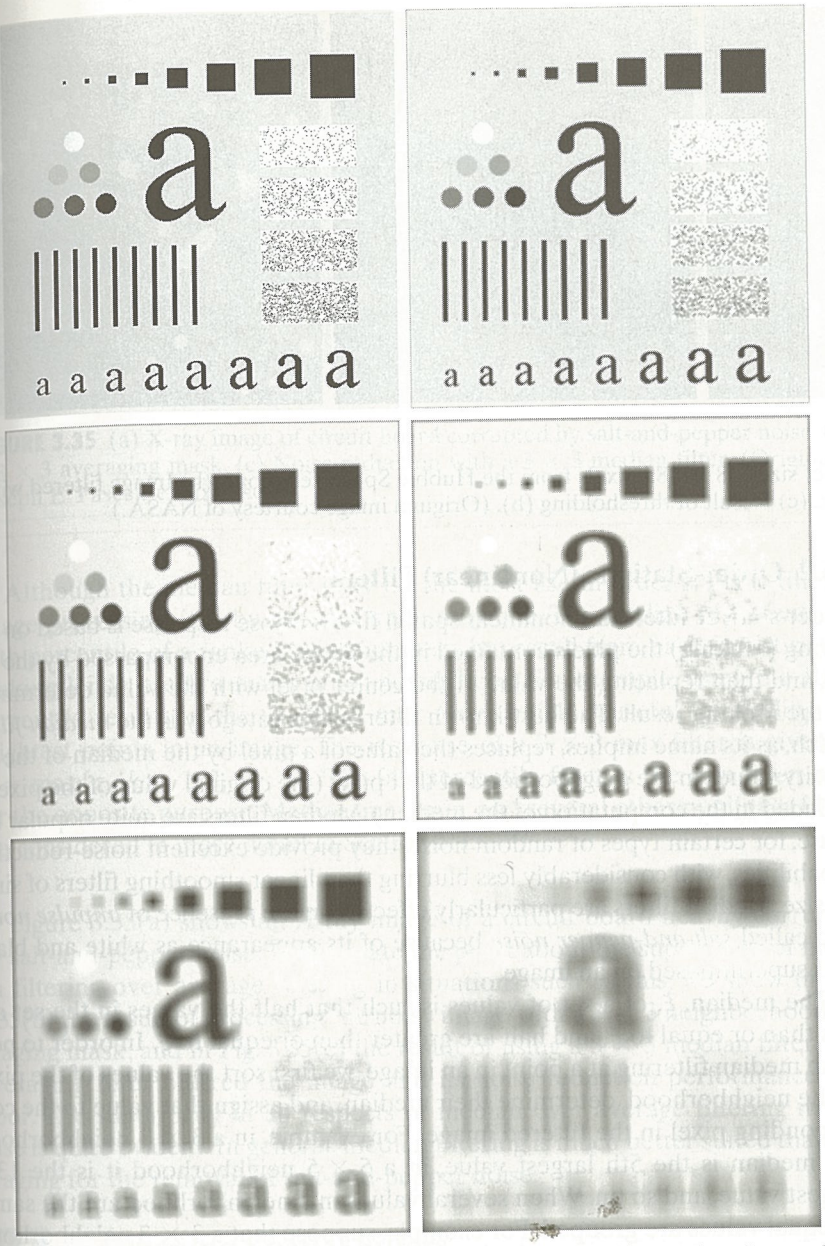
Eq. (3.5-1) is simply the sum of the mask coefficients and, therefore, it is a constant that needs to be computed only once.

EXAMPLE 3.13: Image smoothing with masks of various sizes.

■ The effects of smoothing as a function of filter size are illustrated in Fig. 3.33, which shows an original image and the corresponding smoothed results obtained using square averaging filters of sizes $m = 3$, 5, 9, 15, and 35 pixels, respectively. The principal features of these results are as follows: For $m = 3$, we note a general slight blurring throughout the entire image but, as expected, details that are of approximately the same size as the filter mask are affected considerably more. For example, the 3×3 and 5×5 black squares in the image, the small letter "a," and the fine grain noise show significant blurring when compared to the rest of the image. Note that the noise is less pronounced, and the jagged borders of the characters were pleasingly smoothed.

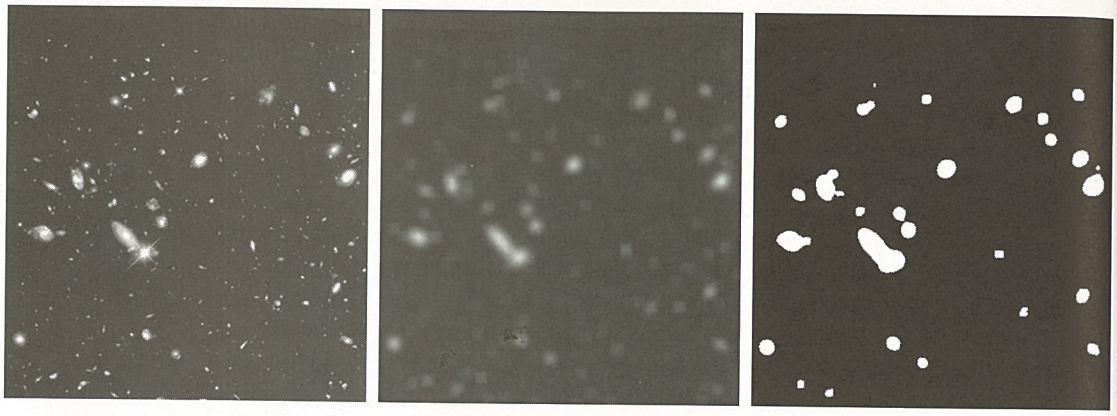
The result for $m = 5$ is somewhat similar, with a slight further increase in blurring. For $m = 9$ we see considerably more blurring, and the 20% black circle is not nearly as distinct from the background as in the previous three images, illustrating the blending effect that blurring has on objects whose intensities are close to that of its neighboring pixels. Note the significant further smoothing of the noisy rectangles. The results for $m = 15$ and 35 are extreme with respect to the sizes of the objects in the image. This type of aggresive blurring generally is used to eliminate small objects from an image. For instance, the three small squares, two of the circles, and most of the noisy rectangle areas have been blended into the background of the image in Fig. 3.33(f). Note also in this figure the pronounced black border. This is a result of padding the border of the original image with 0s (black) and then trimming off the padded area after filtering. Some of the black was blended into all filtered images, but became truly objectionable for the images smoothed with the larger filters. ■

As mentioned earlier, an important application of spatial averaging is to blur an image for the purpose of getting a gross representation of objects of interest, such that the intensity of smaller objects blends with the background and larger objects become "bloblike" and easy to detect. The size of the mask establishes the relative size of the objects that will be blended with the background. As an illustration, consider Fig. 3.34(a), which is an image from the Hubble telescope in orbit around the Earth. Figure 3.34(b) shows the result of applying a 15×15 averaging mask to this image. We see that a number of objects have either blended with the background or their intensity has diminished considerably. It is typical to follow an operation like this with thresholding to eliminate objects based on their intensity. The result of using the thresholding function of Fig. 3.2(b) with a threshold value equal to 25% of the highest intensity in the blurred image is shown in Fig. 3.34(c). Comparing this result with the original image, we see that it is a reasonable representation of what we would consider to be the largest, brightest objects in that image.



a b
c d
e f

FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.



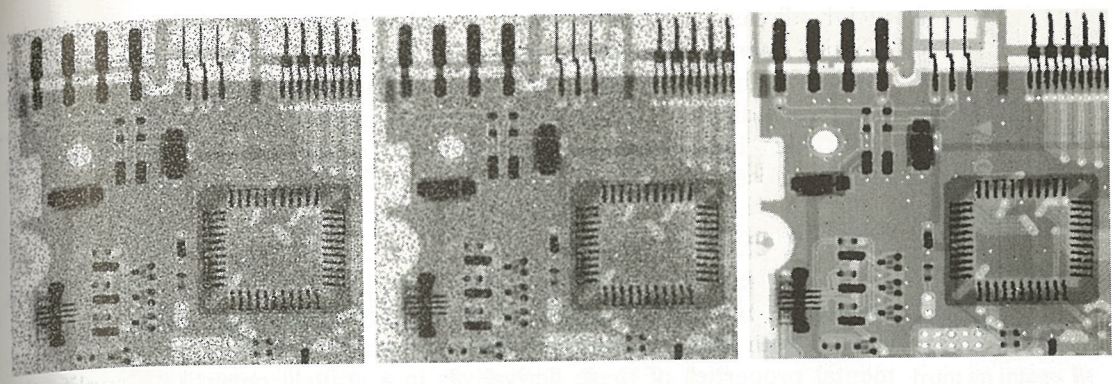
a b c

FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

3.5.2 Order-Statistic (Nonlinear) Filters

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

The median, 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 pixel in the neighborhood, determine their median, and assign that value to the corresponding pixel in the filtered image. 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 with distinct intensity levels to be more like their neighbors. In fact, isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $m^2/2$ (one-half the filter area), are eliminated by an $m \times m$ median filter. In this case "eliminated" means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.



a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Although the median filter is by far the most useful order-statistic filter in image processing, it is by no means the only one. The median represents the 50th percentile of a ranked set of numbers, but 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*, which is useful for finding the brightest points in an image. The response of a 3×3 max filter is given by $R = \max\{z_k | k = 1, 2, \ldots, 9\}$. The 0th percentile filter is the *min filter*, used for the opposite purpose. Median, max, min, and several other nonlinear filters are considered in more detail in Section 5.3.

See Section 10.3.5 regarding percentiles.

EXAMPLE 3.14: Use of median filtering for noise reduction.

■ Figure 3.35(a) shows an X-ray image of a circuit board heavily corrupted by salt-and-pepper noise. To illustrate the point about the superiority of median filtering over average filtering in situations such as this, we show in Fig. 3.35(b) the result of processing the noisy image with a 3×3 neighborhood averaging mask, and in Fig. 3.35(c) the result of using a 3×3 median filter. The averaging filter blurred the image and its noise reduction performance was poor. The superiority in all respects of median over average filtering in this case is quite evident. In general, median filtering is much better suited than averaging for the removal of salt-and-pepper noise. ■

3.6 Sharpening Spatial Filters

The principal objective of sharpening is to highlight transitions in intensity. Uses of image sharpening vary and include applications ranging from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems. In the last section, we saw that image blurring could be accomplished in the spatial domain by pixel averaging in a neighborhood. Because averaging is analogous to integration, it is logical to conclude that sharpening can be accomplished by spatial differentiation. This, in fact, is the case,

and the discussion in this section deals with various ways of defining and implementing operators for sharpening by digital differentiation. Fundamentally, the strength of the response of a derivative operator is proportional to the degree of intensity discontinuity of the image at the point at which the operator is applied. Thus, image differentiation enhances edges and other discontinuities (such as noise) and deemphasizes areas with slowly varying intensities.

3.6.1 Foundation

In the two sections that follow, we 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. The behavior of derivatives during transitions into and out of these image features also is of interest.

The 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; and (3) must be nonzero along ramps. Similarly, any definition of a *second derivative* (1) must be zero in constant areas; (2) must be nonzero at the onset *and* end of an intensity step or ramp; and (3) must be zero along ramps of constant slope. Because we are dealing with digital quantities whose values are finite, 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

$$\frac{\partial f}{\partial x} = f(x + 1) - f(x) \tag{3.6-1}$$

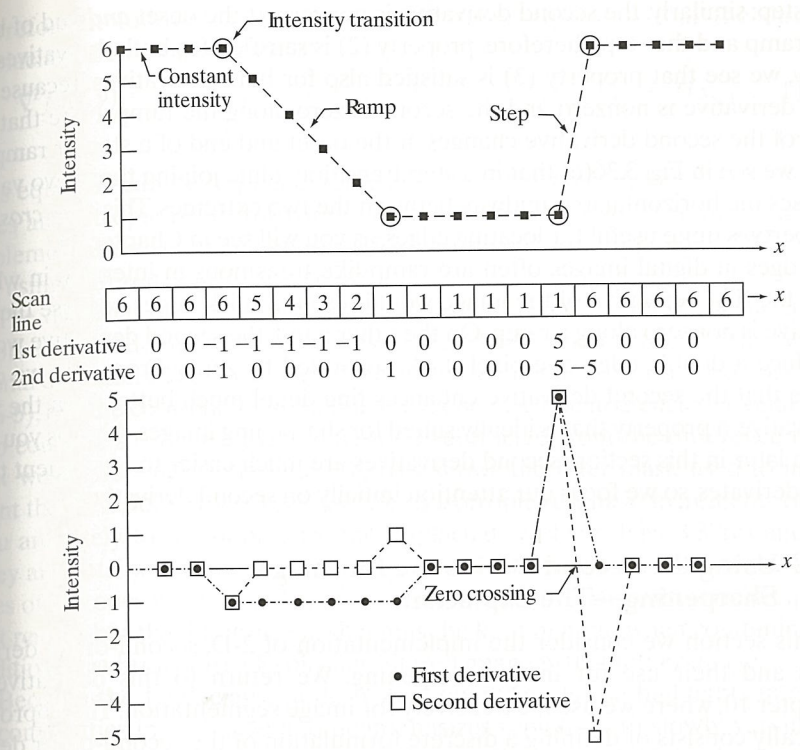
We return to Eq. (3.6-1) in Section 10.2.1 and show how it follows from a Taylor series expansion. For now, we accept it as a definition.

We used a partial derivative here in order to keep the notation the same as 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. Use of a partial derivative in the present discussion does not affect in any way the nature of what we are trying to accomplish. 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

$$\frac{\partial^2 f}{\partial x^2} = f(x + 1) + f(x - 1) - 2f(x) \tag{3.6-2}$$

It is easily verified that these two definitions satisfy the conditions stated above. To illustrate this, and to examine the similarities and differences between



a
b
c

FIGURE 3.36 Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

first- and second-order derivatives of a digital function, consider the example in Fig. 3.36.

Figure 3.36(b) (center of the figure) shows a section of a scan line (intensity profile). The values inside the small squares are the intensity values in the scan line, which are plotted as black dots above it in Fig. 3.36(a). The dashed line connecting the dots is included to aid visualization. As the figure shows, the scan line contains an intensity ramp, three sections of constant intensity, 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 included below the scan line in Fig. 3.36(b), and are plotted in Fig. 3.36(c). When computing the first derivative at a location x, we subtract the value of the function at that location from the next point. 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. 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. 3.36 from the second through the penultimate points in the sequence.

Let us consider the properties of the first and second derivatives as we traverse the profile from left to right. First, we encounter an area of constant intensity and, as Figs. 3.36(b) and (c) show, both derivatives are zero there, so condition (1) is satisfied for 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 for both derivatives. Finally, we see that property (3) is satisfied also for 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. 3.36(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 that is ideally suited for sharpening images. Also, as you will learn later in this section, second derivatives are much easier to implement than first derivates, so we focus our attention initially on second derivatives.

3.6.2 Using the Second Derivative for Image Sharpening—The Laplacian

In this section we consider the implementation of 2-D, second-order derivatives and their use for image sharpening. We return to this derivative in Chapter 10, where we use it extensively for image segmentation. The approach basically consists of defining a discrete formulation of the second-order derivative and then constructing a filter mask based on that formulation. We are interested in *isotropic* filters, whose response is independent of the direction of the discontinuities in the image to which the filter is applied. In other words, isotropic filters are *rotation invariant*, in the sense that rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.

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

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3.6-3}$$

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. (3.6-2), keeping in mind that we have to carry a second variable. In the x-direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x + 1, y) + f(x - 1, y) - 2f(x, y) \tag{3.6-4}$$

and, similarly, in the y-direction we have

$$\frac{\partial^2 f}{\partial y^2} = f(x, y + 1) + f(x, y - 1) - 2f(x, y) \tag{3.6-5}$$

Therefore, it follows from the preceding three equations that the discrete Laplacian of two variables is

$$\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y) \quad (3.6\text{-}6)$$

This equation can be implemented using the filter mask in Fig. 3.37(a), which gives an isotropic result for rotations in increments of 90°. The mechanics of implementation are as in Section 3.5.1 for linear smoothing filters. We simply are using different coefficients here.

The diagonal directions can be incorporated in the definition of the digital Laplacian by adding two more terms to Eq. (3.6-6), one for each of the two diagonal directions. The form of each new term is the same as either Eq. (3.6-4) or (3.6-5), but the coordinates are along the diagonals. Because each diagonal term also contains a $-2f(x, y)$ term, the total subtracted from the difference terms now would be $-8f(x, y)$. Figure 3.37(b) shows the filter mask used to implement this new definition. This mask yields isotropic results in increments of 45°. You are likely to see in practice the Laplacian masks in Figs. 3.37(c) and (d). They are obtained from definitions of the second derivatives that are the negatives of the ones we used in Eqs. (3.6-4) and (3.6-5). As such, they yield equivalent results, but the difference in sign must be kept in mind when combining (by addition or subtraction) a Laplacian-filtered image with another image.

Because the Laplacian is a derivative operator, its use highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be "recovered" while still preserving the sharpening

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a b
c d

FIGURE 3.37
(a) Filter mask used to implement Eq. (3.6-6).
(b) Mask used to implement an extension of this equation that includes the diagonal terms.
(c) and (d) Two other implementations of the Laplacian found frequently in practice.

effect of the Laplacian simply 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*, rather than add, the Laplacian image to obtain a sharpened result. Thus, the basic way in which we use the Laplacian for image sharpening is

$$g(x, y) = f(x, y) + c\left[\nabla^2 f(x, y)\right] \tag{3.6-7}$$

where $f(x, y)$ and $g(x, y)$ are the input and sharpened images, respectively. The constant is $c = -1$ if the Laplacian filters in Fig. 3.37(a) or (b) are used, and $c = 1$ if either of the other two filters is used.

EXAMPLE 3.15: Image sharpening using the Laplacian.

■ Figure 3.38(a) shows a slightly blurred image of the North Pole of the moon. Figure 3.38(b) shows the result of filtering this image with the Laplacian mask in Fig. 3.37(a). Large sections of this image are black because the Laplacian contains both positive and negative values, and all negative values are clipped at 0 by the display.

A typical way to scale a Laplacian image is to add to it its minimum value to bring the new minimum to zero and then scale the result to the full $[0, L - 1]$ intensity range, as explained in Eqs. (2.6-10) and (2.6-11). The image in Fig. 3.38(c) was scaled in this manner. Note that the dominant features of the image are edges and sharp intensity discontinuities. The background, previously black, is now gray due to scaling. This grayish appearance is typical of Laplacian images that have been scaled properly. Figure 3.38(d) shows the result obtained using Eq. (3.6-7) with $c = -1$. The detail in this image is unmistakably clearer and sharper than in the original image. Adding the original image to the Laplacian restored the overall intensity variations in the image, with the Laplacian increasing 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. 3.38(e) shows the result of repeating the preceding procedure with the filter in Fig. 3.37(b). Here, we note a significant improvement in sharpness over Fig. 3.38(d). This is not unexpected because using the filter in Fig. 3.37(b) provides additional differentiation (sharpening) in the diagonal directions. Results such as those in Figs. 3.38(d) and (e) have made the Laplacian a tool of choice for sharpening digital images. ■

3.6.3 Unsharp Masking and Highboost Filtering

A process that has been used for many years by the printing and publishing industry to sharpen images consists of subtracting an unsharp (smoothed) version of an image from the original image. 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.

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

Although significant effort was devoted in the previous chapter to spatial filtering, a thorough understanding of this area is impossible without having at least a working knowledge of how the Fourier transform and the frequency domain can be used for image filtering. You can develop a solid understanding of this topic without having to become a signal processing expert. The key lies in focusing on the fundamentals and their relevance to digital image processing. The notation, usually a source of trouble for beginners, is clarified significantly in this chapter by emphasizing the connection between image characteristics and the mathematical tools used to represent them. This chapter is concerned primarily with establishing a foundation for the Fourier transform and how it is used in basic image filtering. Later, in Chapters 5, 8, 10, and 11, we discuss other applications of the Fourier transform. 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. Next, we start from basic principles of function sampling and proceed step-by-step to derive the one- and two-dimensional discrete Fourier transforms, the basic staples 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 and the development of sections that parallel the spatial

smoothing and sharpening filtering techniques discussed in Chapter 3. We conclude the chapter with a discussion of issues related to implementing the Fourier transform in the context of image processing. Because the material in Sections 4.2 through 4.4 is basic background, readers familiar with the concepts of 1-D signal processing, including the Fourier transform, sampling, aliasing, and the convolution theorem, can proceed to Section 4.5, where we begin a discussion of the 2-D Fourier transform and its application to digital image processing.

4.1 Background

4.1.1 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 published in 1822 in his book, *La Théorie Analitique 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 now taken for granted 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 (Fig. 4.1), so it is not surprising that Fourier's ideas were met initially with skepticism.

Even 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 weighing 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" and then return to the original domain of the function without losing any information. Ultimately, it was the utility of the Fourier series and transform in solving practical problems that made 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 the 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 50 years, entire industries and academic disciplines have flourished as a result of Fourier's ideas. The advent of digital computers and the "discovery" of a fast Fourier transform (FFT) algorithm in the early 1960s (more about this later) revolutionized the field of signal processing. These two core technologies allowed for the first time practical processing of

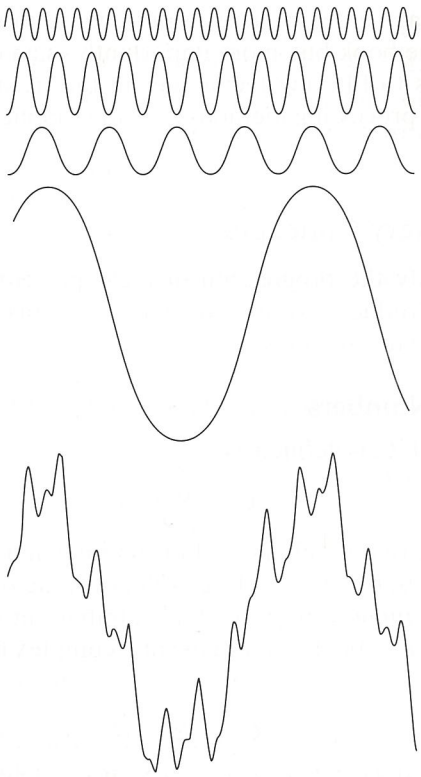


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

a host of signals of exceptional importance, ranging from medical monitors and scanners to modern electronic communications.

We will be dealing only with functions (images) of finite duration, so the Fourier transform is the tool in which we are interested. The material in the following section introduces the Fourier transform and the frequency domain. It is shown that Fourier techniques provide a meaningful and practical way to study and implement a host of image processing approaches. In some cases, these approaches are similar to the ones we developed in Chapter 3.

4.1.2 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

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 long white element, and 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 that can be made in general between the frequency and spatial domains. As we show later in this chapter, even these types of gross associations, coupled with the relationships mentioned previously between frequency content and rate of change of intensity levels in an image, can lead to some very useful results. In the next section, we show the effects of modifying various frequency ranges in the transform of Fig. 4.29(a).

4.7.2 Frequency Domain Filtering Fundamentals

Filtering in the frequency domain consists of modifying the Fourier transform of an image and then computing the inverse transform to obtain the processed result. Thus, given a digital image, $f(x, y)$, of size $M \times N$, the basic filtering equation in which we are interested has the form:

$$g(x, y) = \Im^{-1}[H(u, v)F(u, v)] \tag{4.7-1}$$

If H is real and symmetric and f is real (as is typically the case), then the IDFT in Eq. (4.7-1) should yield real quantities in theory. In practice, the inverse generally contains parasitic complex terms from roundoff and other computational inaccuracies. Thus, it is customary to take the real part of the IDFT to form g.

where $\Im^{-1}$ is the IDFT, $F(u, v)$ is the DFT of the input image, $f(x, y)$, $H(u, v)$ is a *filter function* (also called simply the *filter,* or the *filter transfer function*), and $g(x, y)$ is the filtered (output) image. Functions F, H, and g are arrays of size $M \times N$, the same as the input image. The product $H(u, v)F(u, v)$ is formed using array multiplication, as defined in Section 2.6.1. The filter function modifies the transform of the input image to yield a processed output, $g(x, y)$. Specification of $H(u, v)$ 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 4.6.3, this is accomplished by multiplying the input image by $(-1)^{x+y}$ prior to computing its transform.[†]

We are now in a position to consider the filtering process in some detail. One of the simplest filters we can construct is a filter $H(u, v)$ that is 0 at the center of the transform and 1 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 Eq. (4.6-21) 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 4.30 shows the result of this operation using Eq. (4.7-1). As expected, the image became much darker. (An average of zero

[†]Many 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 filters 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 practice, as long as consistency is maintained.

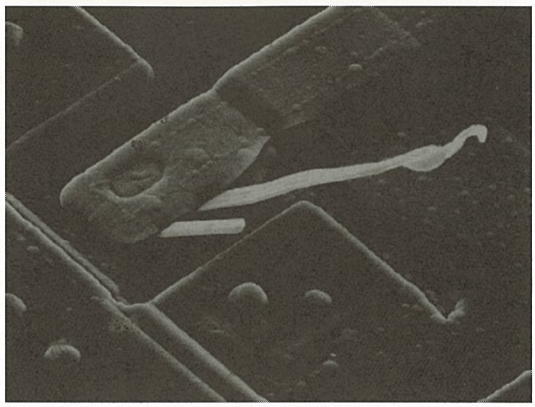


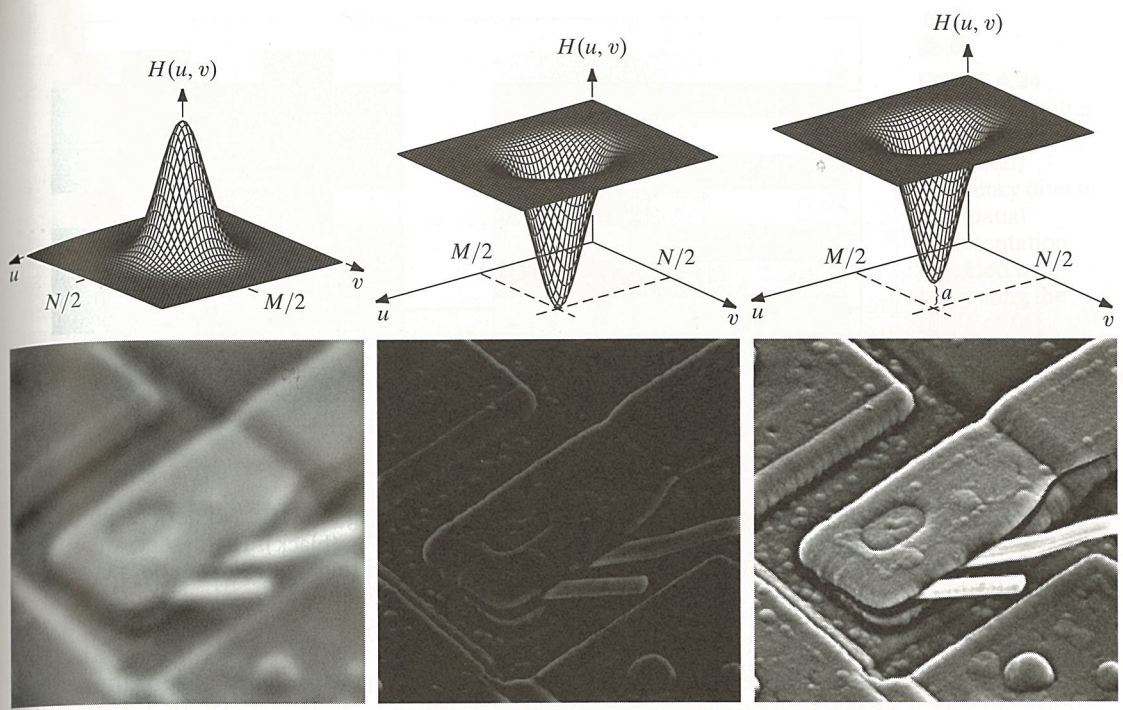
FIGURE 4.30 Result of filtering the image in Fig. 4.29(a) by setting to 0 the term $F(M/2, N/2)$ in the Fourier transform.

implies the existence of negative intensities. Therefore, although it illustrates the principle, Fig. 4.30 is not a true representation of the original, as all negative intensities were clipped (set to 0) for display purposes.)

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 filter $H(u, v)$ that attenuates high frequencies while passing low frequencies (appropriately called a *lowpass filter*) 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 4.31 illustrates these effects. Note the similarity between Figs. 4.31(e) and Fig. 4.30. The reason is that the highpass filter shown eliminates the dc term, resulting in the same basic effect that led to Fig. 4.30. 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, as Fig. 4.31(f) shows.

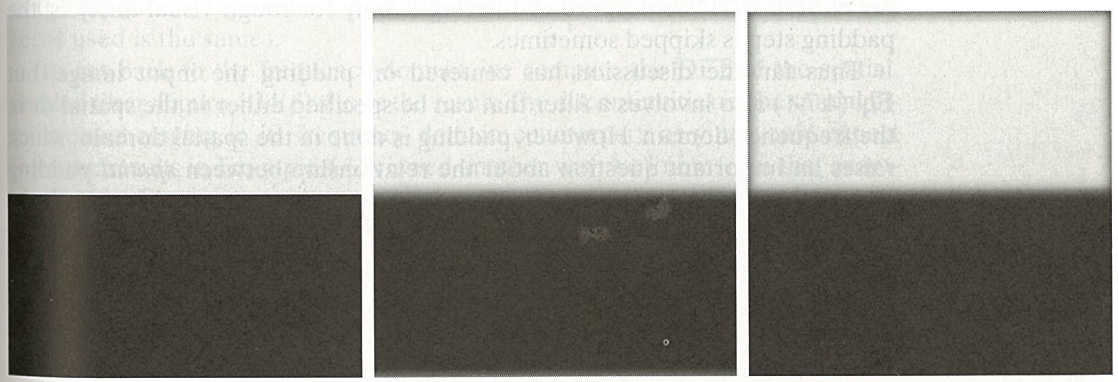
Equation (4.7-1) involves the product of two functions in the frequency domain which, by the convolution theorem, implies convolution in the spatial domain. We know from the discussion in Section 4.6.6 that if the functions in question are not padded we can expect wraparound error. Consider what happens when we apply Eq. (4.7-1) without padding. Figure 4.32(a) shows a simple image, and Fig. 4.32(b) is the result of lowpass filtering the image with a Gaussian lowpass filter of the form shown in Fig. 4.31(a). As expected, the image is blurred. However, the blurring is not uniform; the top white edge is blurred, but the side white edges are not. Padding the input image according to Eqs. (4.6-31) and (4.6-32) before applying Eq. (4.7-1) results in the filtered image in Fig. 4.32(c). This result is as expected.

Figure 4.33 illustrates the reason for the discrepancy between Figs. 4.32(b) and (c). The dashed areas in Fig. 4.33 correspond to the image in Fig. 4.32(a). Figure 4.33(a) shows the periodicity implicit in the use of the DFT, as explained in Section 4.6.3. Imagine convolving the *spatial* representation of the blurring filter with this image. When the filter is passing through the top of the



a b c
d e f

FIGURE 4.31 Top row: frequency domain filters. Bottom row: corresponding filtered images obtained using Eq. (4.7-1). We used $a = 0.85$ in (c) to obtain (f) (the height of the filter itself is 1). Compare (f) with Fig. 4.29(a).



a b c

FIGURE 4.32 (a) A simple image. (b) Result of blurring with a Gaussian lowpass filter without padding. (c) Result of lowpass filtering with padding. Compare the light area of the vertical edges in (b) and (c).

$H(u, v)$ as an image. As, expected, the function is odd, thus the antisymmetry about its center. Function $H(u, v)$ is used as any other frequency domain filter in the procedure outlined in Section 4.7.3.

Figure 4.39(c) is the result of using the filter just obtained in the procedure outlined in Section 4.7.3 to filter the image in Fig. 4.38(a). As expected from a derivative filter, edges are enhanced and all the constant intensity areas are reduced to zero (the grayish tone is due to scaling for display). Figure 4.39(d) shows the result of filtering the same image in the spatial domain directly, using $h(x, y)$ in the procedure outlined in Section 3.6.4. The results are identical. ■

4.8 Image Smoothing Using Frequency Domain Filters

The remainder of this chapter deals with various filtering techniques in the frequency domain. We begin 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 Butterworth filter has a parameter called the *filter order*. For high order values, the Butterworth filter approaches the ideal filter. For lower order values, the Butterworth filter is more like a Gaussian filter. Thus, the Butterworth filter may be viewed as providing a transition between two "extremes." All filtering in this section follows the procedure outlined in Section 4.7.3, so all filter functions, $H(u, v)$, are understood to be discrete functions of size $P \times Q$; that is, the discrete frequency variables are in the range $u = 0, 1, 2, \ldots, P - 1$ and $v = 0, 1, 2, \ldots, Q - 1$.

4.8.1 Ideal Lowpass Filters

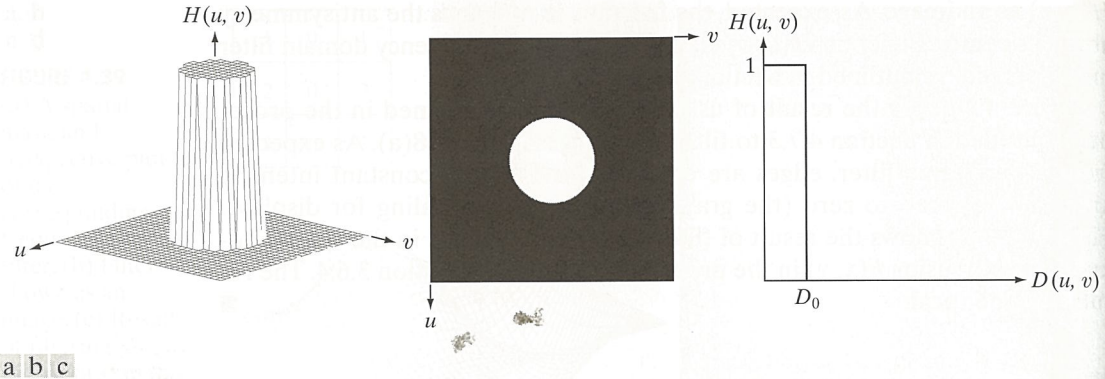
A 2-D lowpass filter that passes without attenuation all frequencies within a circle of radius D_0 from the origin and "cuts off" all frequencies outside this circle is called an *ideal lowpass filter* (ILPF); it is specified by the function

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases} \tag{4.8-1}$$

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 frequency rectangle; that is,

$$D(u, v) = \left[(u - P/2)^2 + (v - Q/2)^2\right]^{1/2} \tag{4.8-2}$$

where, as before, P and Q are the padded sizes from Eqs. (4.6-31) and (4.6-32). Figure 4.40(a) shows a perspective plot of $H(u, v)$ and Fig. 4.40(b) shows the filter displayed as an image. As mentioned in Section 4.3.3, the name *ideal* indicates that all frequencies on or inside a circle of radius D_0 are passed



a b c

FIGURE 4.40 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

without attenuation, whereas all frequencies outside the circle are completely attenuated (filtered out). The ideal lowpass filter is radially symmetric about the origin, which means that the filter is completely defined by a radial cross section, as Fig. 4.40(c) shows. Rotating the cross section by 360° yields the filter in 2-D.

For an ILPF cross section, the point of transition between $H(u, v) = 1$ and $H(u, v) = 0$ is called the *cutoff frequency*. In the case of Fig. 4.40, for example, the cutoff frequency is D_0. The sharp cutoff frequencies of an ILPF cannot be realized with electronic components, although they certainly can be simulated in a computer. The effects of using these "nonphysical" filters on a digital image are discussed later in this section.

The lowpass filters introduced in this chapter are compared by studying their behavior as a function of the same cutoff frequencies. One way to establish a set of standard cutoff frequency loci is to compute circles that enclose specified amounts of total image power P_T. This quantity is obtained by summing the components of the power spectrum of the padded images at each point (u, v), for $u = 0, 1, \ldots, P - 1$ and $v = 0, 1, \ldots, Q - 1$; that is,

$$P_T = \sum_{u=0}^{P-1} \sum_{v=0}^{Q-1} P(u, v) \tag{4.8-3}$$

where $P(u, v)$ is given in Eq. (4.6-18). 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

$$\alpha = 100\left[\sum_u \sum_v P(u, v)/P_T\right] \tag{4.8-4}$$

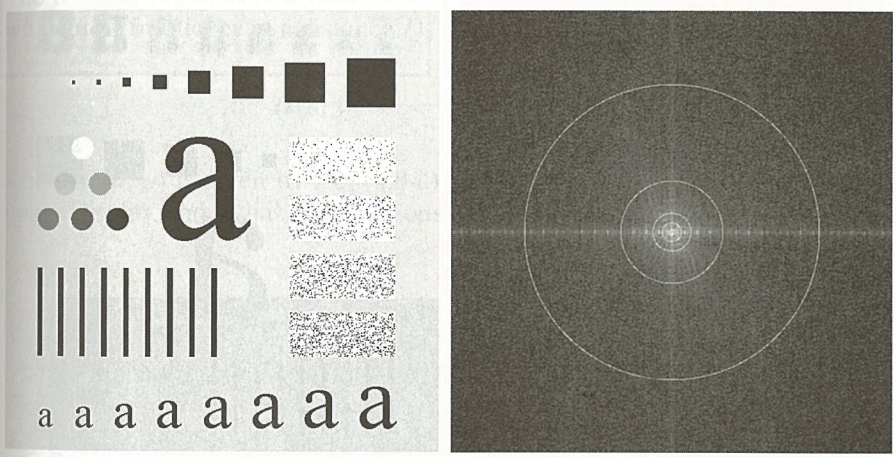
and the summation is taken over values of (u, v) that lie inside the circle or on its boundary.

Figures 4.41(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, respectively. These circles enclose α percent of the image power, for $\alpha = 87.0$, 93.1, 95.7, 97.8, and 99.2%, respectively. The spectrum falls off rapidly, with 87% of the total power being enclosed by a relatively small circle of radius 10.

EXAMPLE 4.16: Image smoothing using an ILPF.

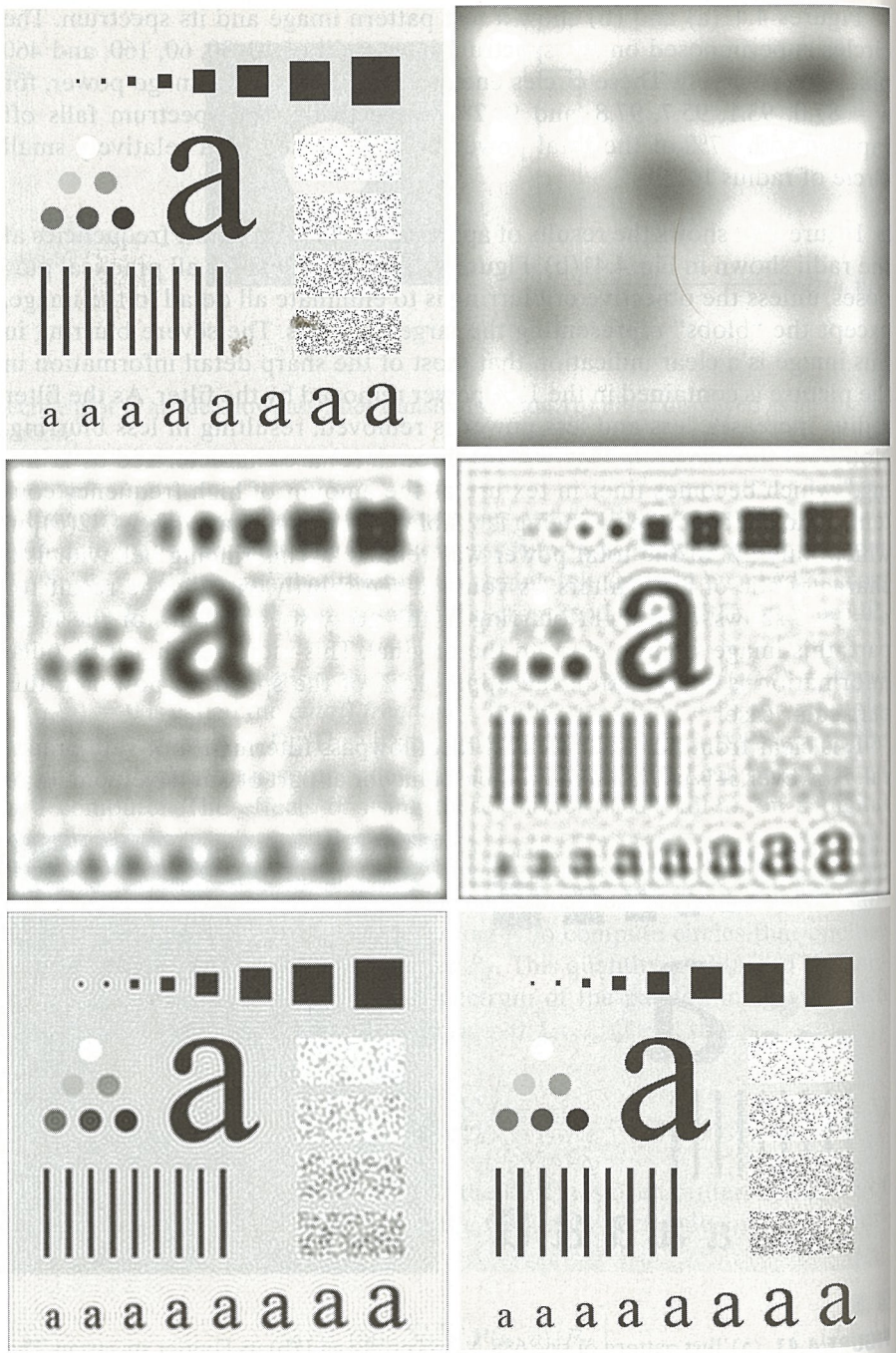
■ Figure 4.42 shows the results of applying ILPFs with cutoff frequencies at the radii shown in Fig. 4.41(b). Figure 4.42(b) is useless for all practical purposes, unless the objective of blurring is to eliminate all detail in the image, except the "blobs" representing the largest objects. The severe blurring in this image is a clear indication that most of the sharp detail information in the picture 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. 4.42(c) through (e) are characterized by "ringing," which becomes finer in texture as the amount of high frequency content removed decreases. Ringing is visible even in the image [Fig. 4.42(e)] in which only 2% of the total power was removed. This ringing behavior is a characteristic of ideal filters, as you will see shortly. Finally, the result for $\alpha = 99.2$ shows very slight blurring in the noisy squares but, for the most part, this image is quite close to the original. This indicates that little edge information is contained in the upper 0.8% of the spectrum power in this particular case.

It is clear from this example that ideal lowpass filtering is not very practical. However, it is useful to study their behavior as part of our development of



a b

FIGURE 4.41 (a) Test pattern of size 688 × 688 pixels, and (b) its Fourier spectrum. The spectrum is double the image size due to padding but is shown in half size so that it fits in the page. The superimposed circles have radii equal to 10, 30, 60, 160, and 460 with respect to the full-size spectrum image. These radii enclose 87.0, 93.1, 95.7, 97.8, and 99.2% of the padded image power, respectively.



a b
c d
e f

FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

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. ■

The blurring and ringing properties of ILPFs can be explained using the convolution theorem. Figure 4.43(a) shows the spatial representation, $h(x, y)$, of an ILPF of radius 10, and Fig. 4.43(b) shows the intensity profile of a line passing through the center of the image. Because a cross section of the ILPF in the frequency domain looks like a box filter, it is not unexpected that a cross section of the corresponding spatial filter has the shape of a sinc function. Filtering in the spatial domain is done by convolving $h(x, y)$ with the image. Imagine each pixel in the image being a discrete impulse whose strength is proportional to the intensity of the image at that location. Convolving a sinc with an impulse copies the sinc at the location of the impulse. The center lobe of the sinc is the principal cause of blurring, while the outer, smaller lobes are mainly responsible for ringing. Convolving the sinc with every pixel in the image provides a nice model for explaining the behavior of ILPFs. Because the "spread" of the sinc function is inversely proportional to the radius of $H(u, v)$, the larger D_0 becomes, the more the spatial sinc approaches an impulse which, in the limit, causes no blurring at all when convolved with the image. 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, which is an important objective in lowpass filtering.

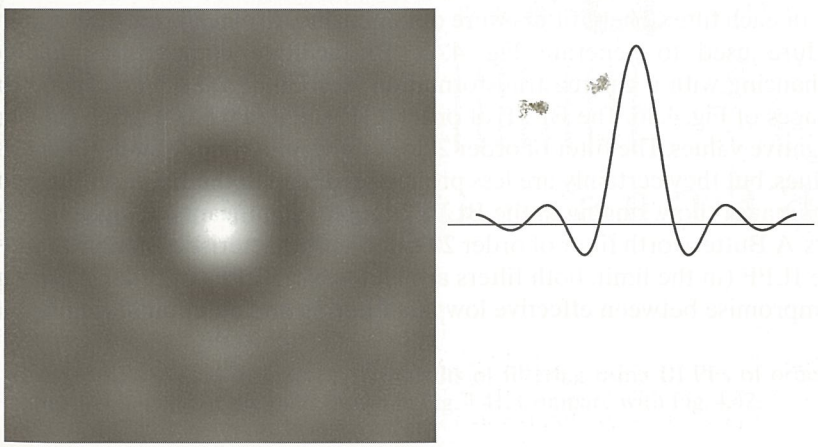
4.8.2 Butterworth Lowpass Filters

The transfer function of a Butterworth lowpass filter (BLPF) of order n, and with cutoff frequency at a distance D_0 from the origin, is defined as

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}} \tag{4.8-5}$$

where $D(u, v)$ is given by Eq. (4.8-2). Figure 4.44 shows a perspective plot, image display, and radial cross sections of the BLPF function.

The transfer function of the Butterworth lowpass filter normally is written as the square root of our expression. However, our interest here is in the basic *form* of the filter, so we exclude the square root for computational convenience.



a b

FIGURE 4.43
(a) Representation in the spatial domain of an ILPF of radius 5 and size 1000×1000. (b) Intensity profile of a horizontal line passing through the center of the image.

10 Image Segmentation

The whole is equal to the sum of its parts.

Euclid

The whole is greater than the sum of its parts.

Max Wertheimer

Preview

The material in the previous chapter began a transition from image processing methods whose inputs and outputs are images, to methods in which the inputs are images but the outputs are attributes extracted from those images (in the sense defined in Section 1.1). Segmentation is another major step in that direction.

Segmentation subdivides an image into its constituent regions or objects. The level of detail to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects or regions of interest in an application have been detected. For example, in the automated inspection of electronic assemblies, interest lies in analyzing images of products with the objective of determining the presence or absence of specific anomalies, such as missing components or broken connection paths. There is no point in carrying segmentation past the level of detail required to identify those elements.

Segmentation of nontrivial images is one of the most difficult tasks in image processing. Segmentation accuracy determines the eventual success or failure of computerized analysis procedures. For this reason, considerable care should be taken to improve the probability of accurate segmentation. In some situations, such as in industrial inspection applications, at least some measure of control over the environment typically is possible. The experienced image processing system designer invariably pays considerable attention to such opportunities. In other applications, such as autonomous target acquisition, the system designer has no control over the operating environment, and the usual

approach is to focus on selecting the types of sensors most likely to enhance the objects of interest while diminishing the contribution of irrelevant image detail. A good example is the use of infrared imaging by the military to detect objects with strong heat signatures, such as equipment and troops in motion.

Most of the segmentation algorithms in this chapter are based on one of two basic properties of intensity values: discontinuity and similarity. In the first category, the approach is to partition an image based on abrupt changes in intensity, such as edges. The principal approaches in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria. Thresholding, region growing, and region splitting and merging are examples of methods in this category. In this chapter, we discuss and illustrate a number of these approaches and show that improvements in segmentation performance can be achieved by combining methods from distinct categories, such as techniques in which edge detection is combined with thresholding. We discuss also image segmentation based on morphology. This approach is particularly attractive because it combines several of the positive attributes of segmentation based on the techniques presented in the first part of the chapter. We conclude the chapter with a brief discussion on the use of motion cues for segmentation.

See Sections 6.7 and 10.3.8 for a discussion of segmentation techniques based on more than just gray (intensity) values.

10.1 Fundamentals

Let R represent the entire spatial region occupied by an image. We may view image segmentation as a process that partitions R into n subregions, $R_1, R_2, \ldots, R_n$, such that

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

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

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

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

(e) $Q(R_i \cup R_j) =$ FALSE for any adjacent regions R_i and R_j.

See Section 2.5.2 regarding connected sets.

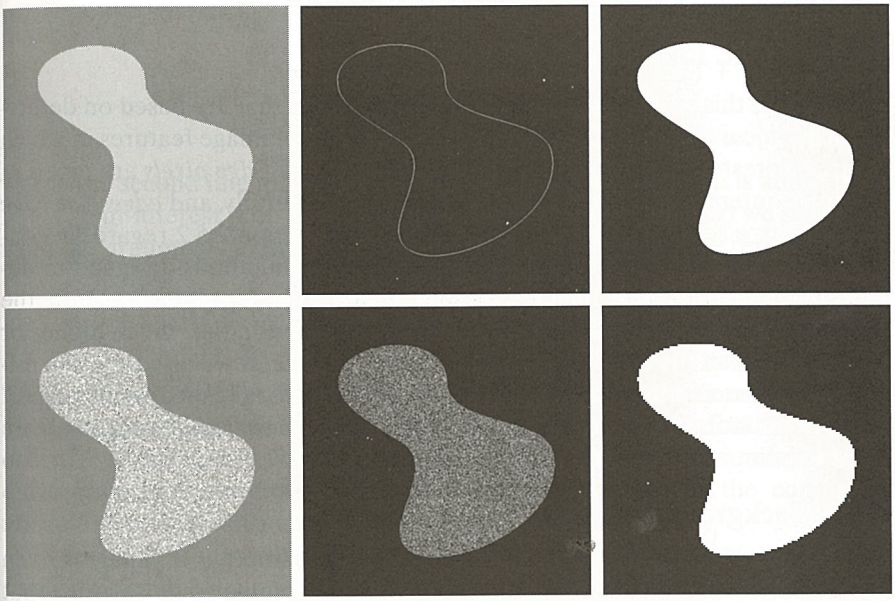
Here, $Q(R_k)$ is a logical predicate defined over the points in set R_k, and $\varnothing$ is the null set. The symbols $\cup$ and $\cap$ represent set union and intersection, respectively, as defined in Section 2.6.4. Two regions R_i and R_j are said to be *adjacent* if their union forms a connected set, as discussed in Section 2.5.2.

Condition (a) indicates that the segmentation must be complete; that is, every pixel must be in a region. Condition (b) requires that points in a region be connected in some predefined sense (e.g., the points must be 4- or 8-connected, as defined in Section 2.5.2). Condition (c) indicates that the regions must be disjoint. Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region—for example, $Q(R_i) =$ TRUE if all pixels in R_i have the same intensity level. Finally, condition (e) indicates that two adjacent regions R_i and R_j must be different in the sense of predicate Q.[†]

[†]In general, Q can be a compound expression such as, for example, $Q(R_i) =$ TRUE if the average intensity of the pixels in R_i is less than m_i, AND if the standard deviation of their intensity is greater than σ_i, where m_i and σ_i are specified constants.

Thus, we see that the fundamental problem in segmentation is to partition an image into regions that satisfy the preceding conditions. Segmentation algorithms for monochrome images generally are based on one of two basic categories dealing with properties of intensity values: discontinuity and similarity. In the first category, the assumption is that boundaries of regions are sufficiently different from each other and from the background to allow boundary detection based on local discontinuities in intensity. *Edge-based segmentation* is the principal approach used in this category. *Region-based segmentation* approaches in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria.

Figure 10.1 illustrates the preceding concepts. Figure 10.1(a) shows an image of a region of constant intensity superimposed on a darker background, also of constant intensity. These two regions comprise the overall image region. Figure 10.1(b) shows the result of computing the boundary of the inner region based on intensity discontinuities. Points on the inside and outside of the boundary are black (zero) because there are no discontinuities in intensity in those regions. To segment the image, we assign one level (say, white) to the pixels on, or interior to, the boundary and another level (say, black) to all points exterior to the boundary. Figure 10.1(c) shows the result of such a procedure. We see that conditions (a) through (c) stated at the beginning of this section are satisfied by



a b c
d e f

FIGURE 10.1 (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number of small edges that are connected to the original boundary, making it difficult to find a unique boundary using only edge information. (f) Result of segmentation based on region properties.

this result. The predicate of condition (d) is: If a pixel is on, or inside the boundary, label it white; otherwise label it black. We see that this predicate is TRUE for the points labeled black and white in Fig. 10.1(c). Similarly, the two segmented regions (object and background) satisfy condition (e).

The next three images illustrate region-based segmentation. Figure 10.1(d) is similar to Fig. 10.1(a), but the intensities of the inner region form a textured pattern. Figure 10.1(e) shows the result of computing the edges of this image. Clearly, the numerous spurious changes in intensity make it difficult to identify a unique boundary for the original image because many of the nonzero intensity changes are connected to the boundary, so edge-based segmentation is not a suitable approach. We note however, that the outer region is constant, so all we need to solve this simple segmentation problem is a predicate that differentiates between textured and constant regions. The standard deviation of pixel values is a measure that accomplishes this, because it is nonzero in areas of the texture region and zero otherwise. Figure 10.1(f) shows the result of dividing the original image into subregions of size 4×4. Each subregion was then labeled white if the standard deviation of its pixels was positive (i.e., if the predicate was TRUE) and zero otherwise. The result has a "blocky" appearance around the edge of the region because groups of 4×4 squares were labeled with the same intensity. Finally, note that these results also satisfy the five conditions stated at the beginning of this section.

10.2 Point, Line, and Edge Detection

The focus of this section is on segmentation methods that are based on detecting sharp, *local* changes in intensity. The three types of image features in which we are interested are isolated points, lines, and edges. *Edge pixels* are pixels at which the intensity of an image function changes abruptly, and *edges* (or *edge segments*) are sets of connected edge pixels (see Section 2.5.2 regarding connectivity). *Edge detectors* are local image processing methods designed to detect edge pixels. A line may be viewed as an edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels. In fact, as we discuss in the following section and in Section 10.2.4, lines give rise to so-called "roof edges." Similarly, an isolated point may be viewed as a line whose length and width are equal to one pixel.

When we refer to lines, we are referring to thin structures, typically just a few pixels thick. Such lines may correspond, for example, to elements of a digitized architectural drawing or roads in a satellite image.

10.2.1 Background

As we saw in Sections 2.6.3 and 3.5, local averaging smooths an image. Given that averaging is analogous to integration, it should come as no surprise that abrupt, local changes in intensity can be detected using derivatives. For reasons that will become evident shortly, first- and second-order derivatives are particularly well suited for this purpose.

Derivatives of a digital function are defined in terms of differences. There are various ways to approximate these differences but, as explained in Section 3.6.1, we require that any approximation used 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; and (3) must be nonzero at points along an intensity

ramp. Similarly, we require that an approximation used for 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; and (3) must be zero along intensity ramps. Because we are dealing with digital quantities whose values are finite, the maximum possible intensity change is also finite, and the shortest distance over which a change can occur is between adjacent pixels.

We obtain an approximation to the first-order derivative at point x of a one-dimensional function $f(x)$ by expanding the function $f(x + \Delta x)$ into a Taylor series about x, letting $\Delta x = 1$, and keeping only the linear terms (Problem 10.1). The result is the digital difference

Recall from Section 2.4.2 that increments between image samples are defined as unity for notational clarity, hence the use of $\Delta x = 1$ in the derivation of Eq. (10.2-1).

$$\frac{\partial f}{\partial x} = f'(x) = f(x + 1) - f(x) \tag{10.2-1}$$

We used a partial derivative here for consistency in notation 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 f is a function of only one variable.

We obtain an expression for the second derivative by differentiating Eq. (10.2-1) with respect to x:

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{\partial f'(x)}{\partial x} = f'(x + 1) - f'(x) \\ &= f(x + 2) - f(x + 1) - f(x + 1) + f(x) \\ &= f(x + 2) - 2f(x + 1) + f(x)\end{aligned}$$

where the second line follows from Eq. (10.2-1). This expansion is about point $x + 1$. Our interest is on the second derivative about point x, so we subtract 1 from the arguments in the preceding expression and obtain the result

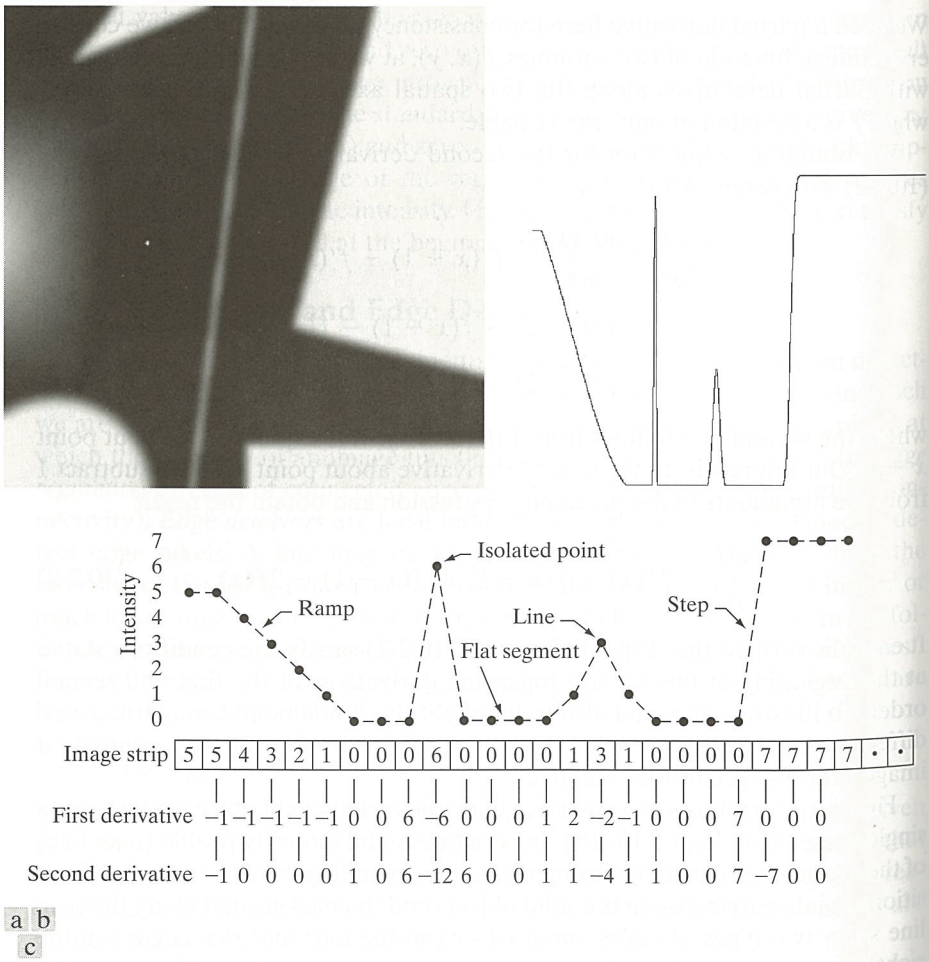
$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x + 1) + f(x - 1) - 2f(x) \tag{10.2-2}$$

It easily is verified that Eqs. (10.2-1) and (10.2-2) satisfy the conditions stated at the beginning of this section regarding derivatives of the first and second order. To illustrate this, and also to highlight the fundamental similarities and differences between first- and second-order derivatives in the context of image processing, consider Fig. 10.2.

Figure 10.2(a) shows an image that contains various solid objects, a line, and a single noise point. Figure 10.2(b) shows a horizontal intensity profile (scan line) of the image approximately through its center, including the isolated point. Transitions in intensity between the solid objects and the background along the scan line show two types of edges: *ramp edges* (on the left) and *step edges* (on the right). As we discuss later, intensity transitions involving thin objects such as lines often are referred to as *roof edges*. Figure 10.2(c) shows a simplification of the profile, with just enough points to make it possible for us to analyze numerically how the first- and second-order derivatives behave as they encounter a noise point, a line, and the edges of objects. In this simplified diagram the

transition in the ramp spans four pixels, the noise point is a single pixel, the line is three pixels thick, and the transition of the intensity step takes place between adjacent pixels. The number of intensity levels was limited to eight for simplicity.

Consider the properties of the first and second derivatives as we traverse the profile from left to right. Initially, we note that the first-order derivative is nonzero at the onset and along the entire intensity ramp, while the second-order derivative is nonzero only at the onset and end of the ramp. Because edges of digital images resemble this type of transition, we conclude that first-order derivatives produce "thick" edges and second-order derivatives much finer ones. Next we encounter the isolated noise point. Here, the magnitude of the response at the point is much stronger for the second- than for the first-order derivative. This is not unexpected, because a second-order derivative is much



a b
c

FIGURE 10.2 (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).

more aggressive than a first-order derivative in enhancing sharp changes. Thus, we can expect second-order derivatives to enhance fine detail (including noise) much more than first-order derivatives. The line in this example is rather thin, so it too is fine detail, and we see again that the second derivative has a larger magnitude. Finally, note in both the ramp and step edges that the second derivative has opposite signs (negative to positive or positive to negative) as it transitions into and out of an edge. This "double-edge" effect is an important characteristic that, as we show in Section 10.2.6, can be used to locate edges. The sign of the second derivative is used also to determine whether an edge is a transition from light to dark (negative second derivative) or from dark to light (positive second derivative), where the sign is observed as we move *into* the edge.

In summary, we arrive at the following conclusions: (1) First-order derivatives generally produce thicker edges in an image. (2) Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise. (3) Second-order derivatives produce a double-edge response at ramp and step transitions in intensity. (4) The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light.

The approach of choice for computing first and second derivatives at every pixel location in an image is to use spatial filters. For the 3×3 filter mask in Fig. 10.3, the procedure is to compute the sum of products of the mask coefficients with the intensity values in the region encompassed by the mask. That is, with reference to Eq. (3.4.3), the *response* of the mask at the center point of the region is

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 \\ &= \sum_{k=1}^{9} w_k z_k \end{aligned} \tag{10.2-3}$$

where z_k is the intensity of the pixel whose spatial location corresponds to the location of the kth coefficient in the mask. The details of implementing this operation over all pixels in an image are discussed in detail in Sections 3.4 and 3.6. In other words, computation of derivatives based on spatial masks is spatial filtering of an image with those masks, as explained in those sections.†

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 10.3
A general 3×3 spatial filter mask.

†As explained in Section 3.4.3, Eq. (10.2-3) is simplified notation either for spatial correlation, given by Eq. (3.4-1), or spatial convolution, given by Eq. (3.4-2). Therefore, when R is evaluated at all locations in an image, the result is an array. All spatial filtering in this chapter is done using correlation. In some instances, we use the term *convolving a mask with an image* as a matter of convention. However, we use this terminology only when the filter masks are symmetric, in which case correlation and convolution yield the same result.

10.2.2 Detection of Isolated Points

Based on the conclusions reached in the preceding section, we know that point detection should be based on the second derivative. From the discussion in Section 3.6.2, this implies using the Laplacian:

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{10.2-4}$$

where the partials are obtained using Eq. (10.2-2):

$$\frac{\partial^2 f(x, y)}{\partial x^2} = f(x + 1, y) + f(x - 1, y) - 2f(x, y) \tag{10.2-5}$$

and

$$\frac{\partial^2 f(x, y)}{\partial y^2} = f(x, y + 1) + f(x, y - 1) - 2f(x, y) \tag{10.2-6}$$

The Laplacian is then

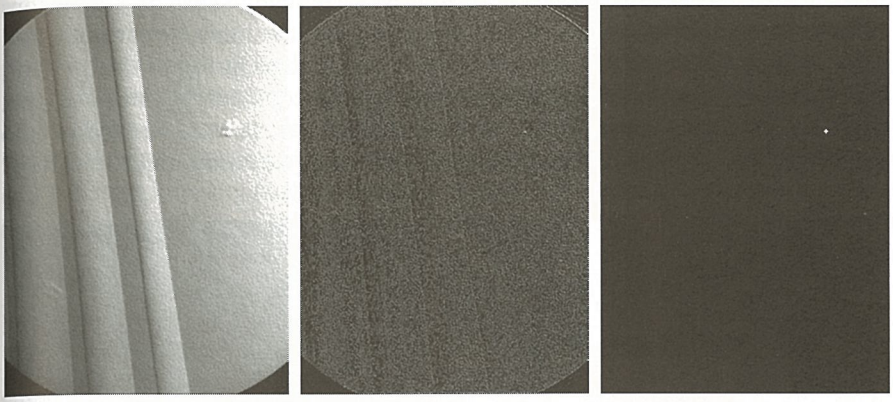
$$\begin{aligned} \nabla^2 f(x, y) = {} & f(x + 1, y) + f(x - 1, y) + f(x, y + 1) \\ & + f(x, y - 1) - 4f(x, y) \end{aligned} \tag{10.2-7}$$

As explained in Section 3.6.2, this expression can be implemented using the mask in Fig. 3.37(a). Also, as explained in that section, we can extend Eq. (10.2-7) to include the diagonal terms, and use the mask in Fig. 3.37(b). Using the Laplacian mask in Fig. 10.4(a), which is the same as the mask in Fig. 3.37(b), we say that a point has been detected at the location (x, y) on which the mask is centered if the absolute value of the response of the mask at that point exceeds a specified threshold. Such points are labeled 1 in the output image and all others are labeled 0, thus producing a binary image. In other words, the output is obtained using the following expression:

$$g(x, y) = \begin{cases} 1 & \text{if } |R(x, y)| \geq T \\ 0 & \text{otherwise} \end{cases} \tag{10.2-8}$$

where g is the output image, T is a nonnegative threshold, and R is given by Eq. (10.2-3). This formulation simply measures the weighted differences between a pixel and its 8-neighbors. Intuitively, the idea is that the intensity of an isolated point will be quite different from its surroundings and thus will be easily detectable by this type of mask. The only differences in intensity that are considered of interest are those large enough (as determined by T) to be considered isolated points. Note that, as usual for a derivative mask, the coefficients sum to zero, indicating that the mask response will be zero in areas of constant intensity.

1	1	1
1	−8	1
1	1	1



a
b c d

FIGURE 10.4
(a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

EXAMPLE 10.1: Detection of isolated points in an image.

■ We illustrate segmentation of isolated points in an image with the aid of Fig. 10.4(b), which is an X-ray image of a turbine blade from a jet engine. The blade has a porosity in the upper-right quadrant of the image, and there is a single black pixel embedded within the porosity. Figure 10.4(c) is the result of applying the point detector mask to the X-ray image, and Fig. 10.4(d) shows the result of using Eq. (10.2-8) with T equal to 90% of the highest absolute pixel value of the image in Fig. 10.4(c). The single pixel is clearly visible in this image (the pixel was enlarged manually to enhance its visibility). This type of detection process is rather specialized, because it is based on abrupt intensity changes at single-pixel locations that are surrounded by a homogeneous background in the area of the detector mask. When this condition is not satisfied, other methods discussed in this chapter are more suitable for detecting intensity changes. ■

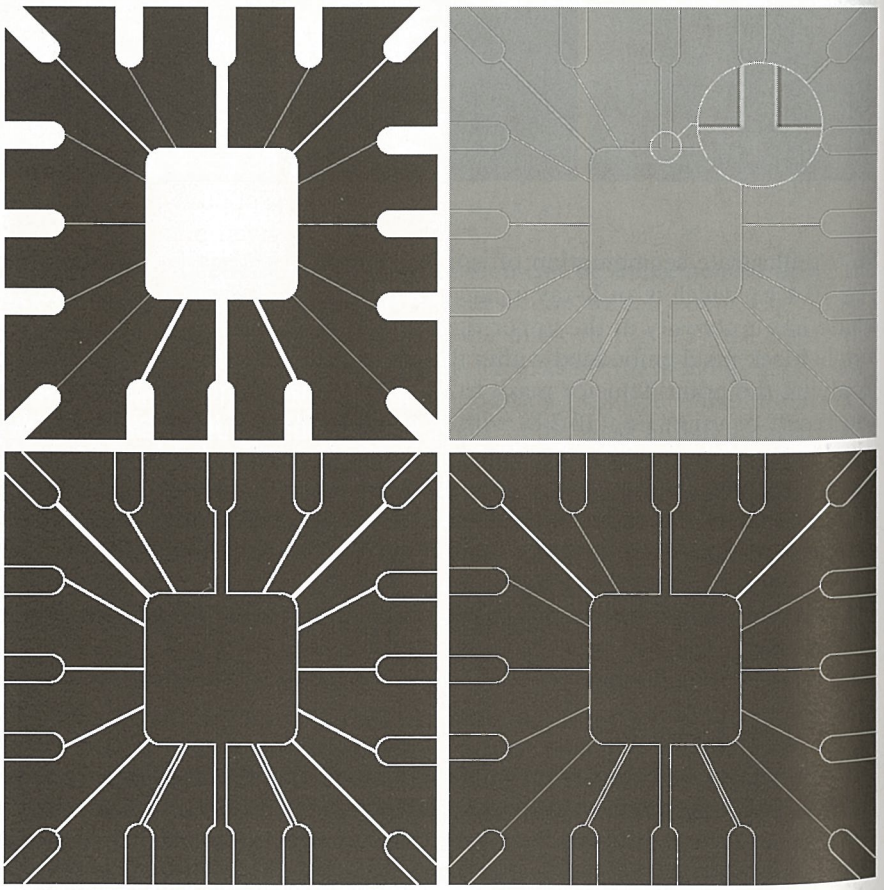
10.2.3 Line Detection

The next level of complexity is line detection. Based on the discussion in Section 10.2.1, we know that for line detection we can expect second derivatives to result in a stronger response and to produce thinner lines than first derivatives. Thus, we can use the Laplacian mask in Fig. 10.4(a) for line detection also, keeping in mind that the double-line effect of the second derivative must be handled properly. The following example illustrates the procedure.

EXAMPLE 10.2: Using the Laplacian for line detection.

■ Figure 10.5(a) shows a 486 × 486 (binary) portion of a wire-bond mask for an electronic circuit, and Fig. 10.5(b) shows its Laplacian image. Because the Laplacian image contains negative values,† scaling is necessary for display. As the magnified section shows, mid gray represents zero, darker shades of gray represent negative values, and lighter shades are positive. The double-line effect is clearly visible in the magnified region.

At first, it might appear that the negative values can be handled simply by taking the absolute value of the Laplacian image. However, as Fig. 10.5(c) shows, this approach doubles the thickness of the lines. A more suitable approach is to use only the positive values of the Laplacian (in noisy situations we use the values that exceed a positive threshold to eliminate random variations about zero caused by the noise). As the image in Fig. 10.5(d) shows, this approach results in thinner lines, which are considerably more useful. Note in Figs. 10.5(b) through (d) that when the lines are wide with respect to the size of the Laplacian mask, the lines are separated by a zero "valley."



a b
c d

FIGURE 10.5
(a) Original image. (b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian. (c) Absolute value of the Laplacian. (d) Positive values of the Laplacian.

†When a mask whose coefficients sum to zero is convolved with an image, the pixels in the resulting image will sum to zero also (Problem 3.16), implying the existence of both positive and negative pixels in the result. Scaling so that all values are nonnegative is required for display purposes.

This is not unexpected. For example, when the 3×3 filter is centered on a line of constant intensity 5 pixels wide, the response will be zero, thus producing the effect just mentioned. When we talk about line detection, the assumption is that lines are thin with respect to the size of the detector. Lines that do not satisfy this assumption are best treated as regions and handled by the edge detection methods discussed later in this section. ■

The Laplacian detector in Fig. 10.4(a) is isotropic, so its response is independent of direction (with respect to the four directions of the 3×3 Laplacian mask: vertical, horizontal, and two diagonals). Often, interest lies in detecting lines in *specified* directions. Consider the masks in Fig. 10.6. Suppose that an image with a constant background and containing various lines (oriented at 0°, ±45°, and 90°) is filtered with the first mask. The maximum responses would occur at image locations in which a horizontal line passed through the middle row of the mask. This is easily verified by sketching a simple array of 1s with a line of a different intensity (say, 5s) running horizontally through the array. A similar experiment would reveal that the second mask in Fig. 10.6 responds best to lines oriented at +45°; the third mask to vertical lines; and the fourth mask to lines in the −45° direction. The preferred direction of each mask is weighted with a larger coefficient (i.e., 2) than other possible directions. The coefficients in each mask sum to zero, indicating a zero response in areas of constant intensity.

Recall from Section 2.4.2 that the image axis convention has the origin at the top left, with the positive x-axis pointing down and the positive y-axis extending to the right. The angles of the lines discussed in this section are measured with respect to the positive x-axis. For example, a vertical line has an angle of 0°, and a +45° line extends downward and to the right.

Let R_1, R_2, R_3, and R_4 denote the responses of the masks in Fig. 10.6, from left to right, where the Rs are given by Eq. (10.2-3). Suppose that an image is filtered (individually) with the four masks. If, at a given point in the image, $|R_k| > |R_j|$, for all $j \neq k$, that point is said to be more likely associated with a line in the direction of mask k. For example, if at a point in the image, $|R_1| > |R_j|$ for $j = 2, 3, 4$, that particular point is said to be more likely associated with a horizontal line. Alternatively, we may be interested in detecting lines in a specified direction. In this case, we would use the mask associated with that direction and threshold its output, as in Eq. (10.2-8). In other words, if we are interested in detecting all the lines in an image in the direction defined by a given mask, we simply run the mask through the image and threshold the absolute value of the result. The points that are left are the strongest responses which, for lines 1 pixel thick, correspond closest to the direction defined by the mask. The following example illustrates this procedure.

Do not confuse our use of R to designate mask response with the same symbol to denote regions in Section 10.1.

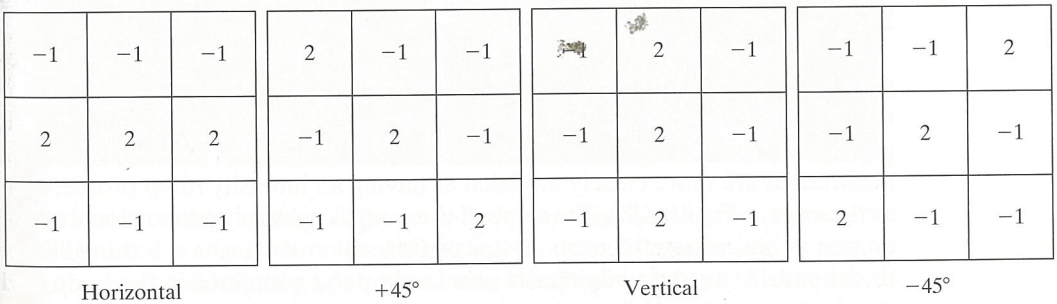


FIGURE 10.6 Line detection masks. Angles are with respect to the axis system in Fig. 2.18(b).

original image. The lines were obtained by joining all gaps not exceeding 20% of the image height (approximately 100 pixels). These lines clearly correspond to the edges of the runway of interest.

Note that the only key knowledge needed to solve this problem was the orientation of the runway and the observer's position relative to it. In other words, a vehicle navigating autonomously would know that if the runway of interest faces north, and the vehicle's direction of travel also is north, the runway should appear vertically in the image. Other relative orientations are handled in a similar manner. The orientations of runways throughout the world are available in flight charts, and direction of travel is easily obtainable using GPS (Global Positioning System) information. This information also could be used to compute the distance between the vehicle and the runway, thus allowing estimates of parameters such as expected length of lines relative to image size, as we did in this example. ■

10.3 Thresholding

Because of its intuitive properties, simplicity of implementation, and computational speed, image thresholding enjoys a central position in applications of image segmentation. Thresholding was introduced in Section 3.1.1, and we have used it in various discussions since then. In this section, we discuss thresholding in a more formal way and develop techniques that are considerably more general than what has been presented thus far.

10.3.1 Foundation

In the previous section, regions were identified by first finding edge segments and then attempting to link the segments into boundaries. In this section, we discuss techniques for partitioning images directly into regions based on intensity values and/or properties of these values.

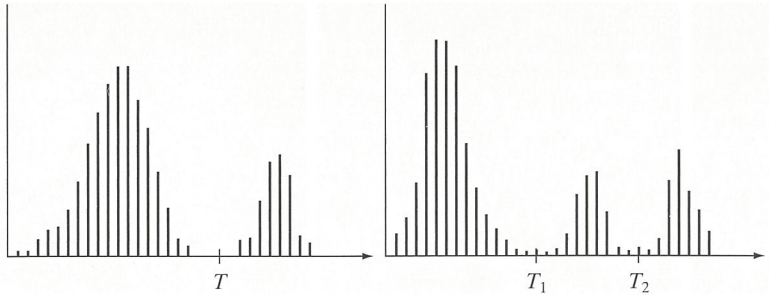
The basics of intensity thresholding

Suppose that the intensity histogram in Fig. 10.35(a) corresponds to an image, $f(x, y)$, composed of light objects on a dark background, in such a way that object and background pixels have intensity values grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold, T, that separates these modes. Then, any point (x, y) in the image at which $f(x, y) > T$ is called an *object point*; otherwise, the point is called a *background point*. In other words, the segmented image, $g(x, y)$, is given by

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} \tag{10.3-1}$$

Although we follow convention in using 0 intensity for the background and 1 for object pixels, any two distinct values can be used in Eq. (10.3-1).

When T is a constant applicable over an entire image, the process given in this equation is referred to as *global thresholding*. When the value of T changes over an image, we use the term *variable thresholding*. The term *local* or *regional thresholding* is used sometimes to denote variable thresholding in



a b

FIGURE 10.35 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.

which the value of T at any point (x, y) in an image depends on properties of a neighborhood of (x, y) (for example, the average intensity of the pixels in the neighborhood). If T depends on the spatial coordinates (x, y) themselves, then variable thresholding is often referred to as *dynamic* or *adaptive* thresholding. Use of these terms is not universal, and one is likely to see them used interchangeably in the literature on image processing.

Figure 10.35(b) shows a more difficult thresholding problem involving a histogram with three dominant modes corresponding, for example, to two types of light objects on a dark background. Here, *multiple thresholding* classifies a point (x, y) as belonging to the background if $f(x, y) \leq T_1$, to one object class if $T_1 < f(x, y) \leq T_2$, and to the other object class if $f(x, y) > T_2$. That is, the segmented image is given by

$$g(x, y) = \begin{cases} a & \text{if } f(x, y) > T_2 \\ b & \text{if } T_1 < f(x, y) \leq T_2 \\ c & \text{if } f(x, y) \leq T_1 \end{cases} \qquad (10.3\text{-}2)$$

where a, b, and c are any three distinct intensity values. We discuss dual thresholding in Section 10.3.6. Segmentation problems requiring more than two thresholds are difficult (often impossible) to solve, and better results usually are obtained using other methods, such as variable thresholding, as discussed in Section 10.3.7, or region growing, as discussed in Section 10.4.

Based on the preceding discussion, we may infer intuitively that the success of intensity thresholding is directly related to the width and depth of the valley(s) separating the histogram modes. In turn, the key factors affecting the properties of the valley(s) are: (1) the separation between peaks (the further apart the peaks are, the better the chances of separating the modes); (2) the noise content in the image (the modes broaden as noise increases); (3) the relative sizes of objects and background; (4) the uniformity of the illumination source; and (5) the uniformity of the reflectance properties of the image.

The role of noise in image thresholding

As an illustration of how noise affects the histogram of an image, consider Fig. 10.36(a). This simple synthetic image is free of noise, so its histogram consists of two "spike" modes, as Fig. 10.36(d) shows. Segmenting this image into two regions is a trivial task involving a threshold placed anywhere between the two