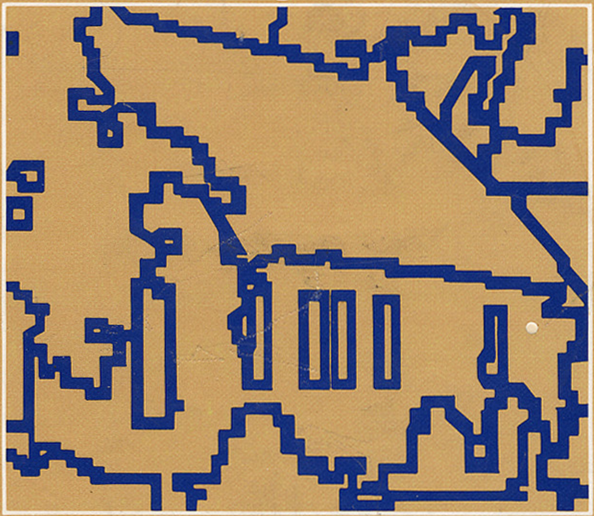
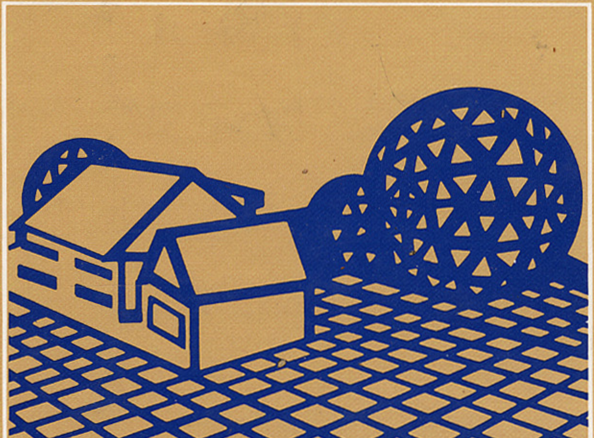
Table of Contents

Return to front page

COMPUTER VISION

DANA H. BALLARD • CHRISTOPHER M. BROWN





COMPUTER VISION

Dana H. Ballard
Christopher M. Brown

Department of Computer Science
University of Rochester
Rochester, New York

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

BALLARD, DANA HARRY.
Computer vision.

Bibliography: p.
Includes index.
1. Image processing. I. Brown, Christopher M.
II. Title.
TA1632.B34 621.38'0414 81-20974
ISBN 0-13-165316-4 AACR2

Cover design by Robin Breite

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-165316-4

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
WHITEHALL BOOKS LIMITED, *Wellington, New Zealand*

Preface

The dream of intelligent automata goes back to antiquity; its first major articulation in the context of digital computers was by Turing around 1950. Since then, this dream has been pursued primarily by workers in the field of *artificial intelligence,* whose goal is to endow computers with information-processing capabilities comparable to those of biological organisms. From the outset, one of the goals of artificial intelligence has been to equip machines with the capability of dealing with sensory inputs.

Computer vision is the construction of explicit, meaningful descriptions of physical objects from images. Image understanding is very different from image processing, which studies image-to-image transformations, not explicit description building. Descriptions are a prerequisite for recognizing, manipulating, and thinking about objects.

We perceive a world of coherent three-dimensional objects with many invariant properties. Objectively, the incoming visual data do not exhibit corresponding coherence or invariance; they contain much irrelevant or even misleading variation. Somehow our visual system, from the retinal to cognitive levels, understands, or imposes order on, chaotic visual input. It does so by using *intrinsic information* that may reliably be extracted from the input, and also through assumptions and *knowledge* that are applied at various levels in visual processing.

The challenge of computer vision is one of *explicitness.* Exactly what information about scenes can be extracted from an image using only very basic assumptions about physics and optics? Explicitly, what computations must be performed? Then, at what stage must domain-dependent, prior knowledge about the world be incorporated into the understanding process? How are world models and knowledge represented and used? This book is about the representations and mechanisms that allow image information and prior knowledge to interact in image understanding.

Computer vision is a relatively new and fast-growing field. The first experiments were conducted in the late 1950s, and many of the essential concepts have been developed during the last five years. With this rapid growth, crucial ideas have arisen in disparate areas such as artificial intelligence, psychology, computer graphics, and image processing. Our intent is to assemble a selection of this material in a form that will serve both as a senior/graduate-level academic text and as a useful reference to those building vision systems. This book has a strong artificial intelligence flavor, and we hope this will provoke thought. We believe that both the intrinsic image information and the internal model of the world are important in successful vision systems.

The book is organized into four parts, based on descriptions of objects at four different levels of abstraction.

1. Generalized images—images and image-like entities.
2. Segmented images—images organized into subimages that are likely to correspond to "interesting objects."
3. Geometric structures—quantitative models of image and world structures.
4. Relational structures—complex symbolic descriptions of image and world structures.

The parts follow a progression of increasing abstractness. Although the four parts are most naturally studied in succession, they are not tightly interdependent. Part I is a prerequisite for Part II, but Parts III and IV can be read independently.

Parts of the book assume some mathematical and computing background (calculus, linear algebra, data structures, numerical methods). However, throughout the book mathematical rigor takes a backseat to concepts. Our intent is to transmit a set of ideas about a new field to the widest possible audience.

In one book it is impossible to do justice to the scope and depth of prior work in computer vision. Further, we realize that in a fast-developing field, the rapid influx of new ideas will continue. We hope that our readers will be challenged to think, criticize, read further, and quickly go beyond the confines of this volume.

D. H. Ballard
C. M. Brown

Acknowledgments

Jerry Feldman and Herb Voelcker (and through them the University of Rochester) provided many resources for this work. One of the most important was a capable and forgiving staff (secretarial, technical, and administrative). For massive text editing, valuable advice, and good humor we are especially grateful to Rose Peet. Peggy Meeker, Jill Orioli, and Beth Zimmerman all helped at various stages.

Several colleagues made suggestions on early drafts: thanks to James Allen, Norm Badler, Larry Davis, Takeo Kanade, John Kender, Daryl Lawton, Joseph O'Rourke, Ari Requicha, Ed Riseman, Azriel Rosenfeld, Mike Schneier, Ken Sloan, Steve Tanimoto, Marty Tenenbaum, and Steve Zucker.

Graduate students helped in many different ways: thanks especially to Michel Denber, Alan Frisch, Lydia Hrechanyk, Mark Kahrs, Keith Lantz, Joe Maleson, Lee Moore, Mark Peairs, Don Perlis, Rick Rashid, Dan Russell, Dan Sabbah, Bob Schudy, Peter Selfridge, Uri Shani, and Bob Tilove. Bernhard Stuth deserves special mention for much careful and critical reading.

Finally, thanks go to Jane Ballard, mostly for standing steadfast through the cycles of elation and depression and for numerous engineering-to-English translations.

As Pat Winston put it: "A willingness to help is not an implied endorsement." The aid of others was invaluable, but we alone are responsible for the opinions, technical details, and faults of this book.

Funding assistance was provided by the Sloan Foundation under Grant 78-4-15, by the National Institutes of Health under Grant HL21253, and by the Defense Advanced Research Projects Agency under Grant N00014-78-C-0164.

The authors wish to credit the following sources for figures and tables. For complete citations given here in abbreviated form (as "from . . ." or "after . . ."), refer to the appropriate chapter-end references.

Fig. 1.2 from Shani, U., "A 3-D model-driven system for the recognition of abdominal anatomy from CT scans," TR77, Dept. of Computer Science, University of Rochester, May 1980.

Fig. 1.4 courtesy of Allen Hanson and Ed Riseman, COINS Research Project, University of Massachusetts, Amherst, MA.

Fig. 2.4 after Horn and Sjoberg, 1978.

Figs. 2.5, 2.9, 2.10, 3.2, 3.6, and 3.7 courtesy of Bill Lampeter.

Fig. 2.7a painting by Louis Condax; courtesy of Eastman Kodak Company and the Optical Society of America.

Fig. 2.8a courtesy of D. Greenberg and G. Joblove, Cornell Program of Computer Graphics.

Fig. 2.8b courtesy of Tom Check.

Table 2.3 after Gonzalez and Wintz, 1977.

Fig. 2.18 courtesy of EROS Data Center, Sioux Falls, SD.

Figs. 2.19 and 2.20 from Herrick, C.N., *Television Theory and Servicing: Black/White and Color,* 2nd Ed. Reston, VA: Reston, 1976.

Figs. 2.21, 2.22, 2.23, and 2.24 courtesy of Michel Denber.

Fig. 2.25 from Popplestone et al., 1975.

Fig. 2.26 courtesy of Production Automation Project, University of Rochester.

Fig. 2.27 from Waag and Gramiak, 1976.

Fig. 3.1 courtesy of Marty Tenenbaum.

Fig. 3.8 after Horn, 1974.

Figs. 3.14 and 3.15 after Frei and Chen, 1977.

Figs. 3.17 and 3.18 from Zucker, S.W. and R.A. Hummel, "An optimal 3-D edge operator," *IEEE Trans. PAMI 3,* May 1981, pp. 324-331.

Fig. 3.19 curves are based on data in Abdou, 1978.

Figs. 3.20, 3.21, and 3.22 from Prager, J.M., "Extracting and labeling boundary segments in natural scenes," *IEEE Tans. PAMI 12,* 1, January 1980. © 1980 IEEE.

Figs. 3.23, 3.28, 3.29, and 3.30 courtesy of Berthold Horn.

Figs. 3.24 and 3.26 from Marr, D. and T. Poggio, "Cooperative computation of stereo disparity," *Science,* Vol. 194, 1976, pp. 283-287. © 1976 by the American Association for the Advancement of Science.

Fig. 3.31 from Woodham, R.J., "Photometric stereo: A reflectance map technique for determining surface orientation from image intensity," *Proc. SPIE,* Vol. 155, August 1978.

Figs. 3.33 and 3.34 after Horn and Schunck, 1980.

Fig. 3.37 from Tanimoto, S. and T. Pavlidis, "A hierarchical data structure for picture processing," *CGIP 4,* 2, June 1975, pp. 104–119.

Fig. 4.6 from Kimme et al., 1975.

Figs. 4.7 and 4.16 from Ballard and Sklansky, 1976.

Fig. 4.9 courtesy of Dana Ballard and Ken Sloan.

Figs. 4.12 and 4.13 from Ramer, U., "Extraction of line structures from photgraphs of curved objects," *CGIP 4,* 2, June 1975, pp. 81–103.

Fig. 4.14 courtesy of Jim Lester, Tufts/New England Medical Center.

Fig. 4.17 from Chien, Y.P. and K.S. Fu, "A decision function method for boundary detection," *CGIP 3,* 2, June 1974, pp. 125–140.

Fig. 5.3 from Ohlander, R., K. Price, and D.R. Reddy, "Picture segmentation using a recursive region splitting method," *CGIP 8,* 3, December 1979.

Fig. 5.4 courtesy of Sam Kapilivsky.

Figs. 6.1, 11.16, and A1.13 courtesy of Chris Brown.

Fig. 6.3 courtesy of Joe Maleson and John Kender.

Fig. 6.4 from Connors, 1979. Texture images by Phil Brodatz, in Brodatz, *Textures.* New York: Dover, 1966.

Fig. 6.9 texture image by Phil Brodatz, in Brodatz, *Textures.* New York: Dover, 1966.

Figs. 6.11, 6.12, and 6.13 from Lu, S.Y. and K.S. Fu, "A syntactic approach to texture analysis," *CGIP 7,* 3, June 1978, pp. 303-330.

Fig. 6.14 from Jayaramamurthy, S.N., "Multilevel array grammars for generating texture scenes," *Proc. PRIP,* August 1979, pp. 391-398. © 1979 IEEE.
Fig. 6.20 from Laws, 1980.
Figs. 6.21 and 6.22 from Maleson et al., 1977.
Fig. 6.23 courtesy of Joe Maleson.
Figs. 7.1 and 7.3 courtesy of Daryl Lawton.
Fig. 7.2 after Prager, 1979.
Figs. 7.4 and 7.5 from Clocksin, W.F., "Computer prediction of visual thresholds for surface slant and edge detection from optical flow fields," Ph.D. dissertation, University of Edinburgh, 1980.
Fig. 7.7 courtesy of Steve Barnard and Bill Thompson.
Figs. 7.8 and 7.9 from Rashid, 1980.
Fig. 7.10 courtesy of Joseph O'Rourke.
Figs. 7.11 and 7.12 after Aggarwal and Duda, 1975.
Fig. 7.13 courtesy of Hans-Hellmut Nagel.
Fig. 8.1d after Requicha, 1977.
Figs. 8.2, 8.3, 8.21a, 8.22, and 8.26 after Pavlidis, 1977.
Figs. 8.10, 8.11, 9.6, and 9.16 courtesy of Uri Shani.
Figs. 8.12, 8.13, 8.14, 8.15, and 8.16 from Ballard, 1981.
Fig. 8.21 b from Preston, K., Jr., M.J.B. Duff; S. Levialdi, P.E. Norgren, and J-i. Toriwaki, "Basics of cellular logic with some applications in medical image processing," *Proc. IEEE,* Vol. 67, No. 5, May 1979, pp. 826–856.
Figs. 8.25, 9.8, 9.9, 9.10, and 11.3 courtesy of Robert Schudy.
Fig. 8.29 after Bribiesca and Guzman, 1979.
Figs. 9.1, 9.18, 9.19, and 9.27 courtesy of Ari Requicha.
Fig. 9.2 from Requicha, A.A.G., "Representations for rigid solids: theory, methods, systems," *Computer Surveys 12,* 4, December 1980.
Fig. 9.3 courtesy of Lydia Hrechanyk.
Figs. 9.4 and 9.5 after Baumgart, 1972.
Fig. 9.7 courtesy of Peter Selfridge.
Fig. 9.11 after Requicha, 1980.
Figs. 9.14 and 9.15b from Agin, G.J. and T.O. Binford, "Computer description of curved objects," *IEEE Trans. on Computers 25,* 1, April 1976.
Fig. 9.15a courtesy of Gerald Agin.
Fig. 9.17 courtesy of A. Christensen; published as frontispiece of *ACM SIGGRAPH 80 Proceedings.*
Fig. 9.20 from Marr and Nishihara, 1978.
Fig. 9.21 after Tilove, 1980.
Fig. 9.22b courtesy of Gene Hartquist.
Figs. 9.24, 9.25, and 9.26 from Lee and Requicha, 1980.
Figs. 9.28a, 9.29, 9.30, 9.31, 9.32, 9.35, and 9.37 and Table 9.1 from Brown, C. and R. Popplestone, "Cases in scene analysis," in *Pattern Recognition,* ed. B.G. Batchelor. New York: Plenum, 1978.
Fig. 9.28b from Guzman, A., "Decomposition of a visual scene into three-dimensional bodies," in *Automatic Interpretation and Classification of Images,* A. Grasseli, ed., New York: Academic Press, 1969.
Fig. 9.28c from Waltz, D., "Understanding line drawing of scenes with shadows," in *The Psychology of Computer Vision,* ed. P.H. Winston. New York: McGraw-Hill, 1975.
Fig. 9.28d after Turner, 1974.
Figs. 9.33, 9.38, 9.40, 9.42, 9.43, and 9.44 after Mackworth, 1973.

Figs. 9.39, 9.45, 9.46, and 9.47 and Table 9.2 after Kanade, 1978.
Figs. 10.2 and A2.1 courtesy of Dana Ballard.
Figs. 10.16, 10.17, and 10.18 after Russell, 1979.
Fig. 11.5 after Fischler and Elschlager, 1973.
Fig. 11.8 after Ambler et al., 1975.
Fig. 11.10 from Winston, P.H., "Learning structural descriptions from examples," in *The Psychology of Computer Vision,* ed. P.H. Winston. New York: McGraw-Hill, 1975.
Fig. 11.11 from Nevatia, 1974.
Fig. 11.12 after Nevatia, 1974.
Fig. 11.17 after Barrow and Popplestone, 1971.
Fig. 11.18 from Davis, L.S., "Shape matching using relaxation techniques," *IEEE Trans. PAMI 1,* 4, January 1979, pp. 60–72.
Figs. 12.4 and 12.5 from Sloan and Bajcsy, 1979.
Fig. 12.6 after Barrow and Tenenbaum, 1976.
Fig. 12.8 after Freuder, 1978.
Fig. 12.10 from Rosenfeld, A.R., A. Hummel, and S.W. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. SMC 6,* 6, June 1976, p. 420.
Figs. 12.11, 12.12, 12.13, 12.14, and 12.15 after Hinton, 1979.
Fig. 13.3 courtesy of Aaron Sloman.
Figs. 13.6, 13.7, and 13.8 from Garvey, 1976.
Fig. A1.11 after Duda and Hart, 1973.
Figs. A2.2 and A2.3 from Hanson, A.R. and E.M. Riseman, "VISIONS: A computer system for interpreting scenes," in *Computer Vision Systems,* ed. A.R. Hanson and E.M. Riseman. New York: Academic Press, 1978.

Mnemonics for Proceedings and Special Collections Cited in the References

CGIP

Computer Graphics and Image Processing

COMPSAC

IEEE Computer Society's 3rd International Computer Software and Applications Conference, Chicago, November 1979.

CVS

Hanson, A. R. and E. M. Riseman (Eds.). *Computer Vision Systems.* New York: Academic Press, 1978.

DARPA IU

Defense Advanced Research Projects Agency Image Understanding Workshop, Minneapolis, MN, April 1977.

Defense Advanced Research Projects Agency Image Understanding Workshop, Palo Alto, CA, October 1977.

Defense Advanced Research Projects Agency Image Understanding Workshop, Cambridge, MA, May 1978.

Defense Advanced Research Projects Agency Image Understanding Workshop, Carnegie-Mellon University, Pittsburgh, PA, November 1978.

Defense Advanced Research Projects Agency Image Understanding Workshop, University of Maryland, College Park, MD, April 1980.

IJCAI

2nd International Joint Conference on Artificial Intelligence, Imperial College, London, September 1971.

4th International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR, September 1975.

5th International Joint Conference on Artificial Intelligence, MIT, Cambridge, MA, August 1977.

6th International Joint Conference on Artificial Intelligence, Tokyo, August 1979.

IJCPR

2nd International Joint Conference on Pattern Recognition, Copenhagen, August 1974.

3rd International Joint Conference on Pattern Recognition, Coronado, CA, November 1976.

4th International Joint Conference on Pattern Recognition, Kyoto, November 1978.

5th International Joint Conference on Pattern Recognition, Miami Beach, FL, December 1980.

MI4

Meltzer, B. and D. Michie (Eds.). *Machine Intelligence 4.* Edinburgh: Edinburgh University Press, 1969.

MI5

Meltzer, B. and D. Michie (Eds.). *Machine Intelligence 5.* Edinburgh: Edinburgh University Press, 1970.

MI6

Meltzer, B. and D. Michie (Eds.). *Machine Intelligence 6.* Edinburgh: Edinburgh University Press, 1971.

M17

Meltzer, B. and D. Michie (Eds.). *Machine Intelligence 7.* Edinburgh: Edinburgh University Press, 1972.

PCV

Winston, P. H. (Ed.). *The Psychology of Computer Vision.* New York: McGraw-Hill, 1975.

PRIP

IEEE Computer Society Conference on Pattern Recognition and Image Processing, Chicago, August 1979.

COMPUTER VISION

DANA H. BALLARD ▪ CHRISTOPHER M. BROWN

What information about scenes can be extracted from an image using only basic assumptions about physics and optics?

How are images segmented into meaningful parts?

At what stage must domain-dependent, prior knowledge about the world be incorporated into the understanding process?

How are world models and conceptual knowledge represented and used?

These and many other questions, inherent in this relatively new and fast-growing field, are explored and answered in **Computer Vision** by **Dana H. Ballard** and **Christopher M. Brown.** The authors assemble crucial material from many disciplines including artificial intelligence, psychology, computer graphics, and image processing to form a practical text and reference for anyone involved in building vision systems.

Ballard and Brown write in their preface, "**Computer Vision** has a strong artificial intelligence flavor, and we hope this will provoke thought. The text shows how both intrinsic image information and internal models of the world are important in successful vision systems."

Divided into four parts, **Computer Vision** offers descriptions of objects at four levels of abstraction:

- Generalized images—images and image-like entities;
- Segmented images—images organized into sub-images that are likely to correspond to "interesting objects";
- Geometrical structures—quantitative models of image and world structures;
- Relational structures—complex symbolic descriptions of image and world structures.

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

ISBN 0-13-165316-4

Contents

PART II SEGMENTED IMAGES 115

Computer Vision 1

Computer Vision Issues

1.1 ACHIEVING SIMPLE VISION GOALS

Suppose that you are given an aerial photo such as that of Fig. 1.1a and asked to locate ships in it. You may never have seen a naval vessel in an aerial photograph before, but you will have no trouble predicting generally how ships will appear. You might reason that you will find no ships inland, and so turn your attention to ocean areas. You might be momentarily distracted by the glare on the water, but realizing that it comes from reflected sunlight, you perceive the ocean as continuous and flat. Ships on the open ocean stand out easily (if you have seen ships from the air, you know to look for their wakes). Near the shore the image is more confusing, but you know that ships close to shore are either moored or docked. If you have a map (Fig. 1.1b), it can help locate the docks (Fig. 1.1c); in a low-quality photograph it can help you identify the shoreline. Thus it might be a good investment of your time to establish the correspondence between the map and the image. A search parallel to the shore in the dock areas reveals several ships (Fig. 1.1d).

Again, suppose that you are presented with a set of computer-aided tomographic (CAT) scans showing "slices" of the human abdomen (Fig. 1.2a). These images are products of high technology, and give us views not normally available even with x-rays. Your job is to reconstruct from these cross sections the three-dimensional shape of the kidneys. This job may well seem harder than finding ships. You first need to know what to look for (Fig. 1.2b), where to find it in CAT scans, and how it looks in such scans. You need to be able to "stack up" the scans mentally and form an internal model of the shape of the kidney as revealed by its slices (Fig. 1.2c and 1.2d).

This book is about *computer vision.* These two example tasks are typical computer vision tasks; both were solved by computers using the sorts of knowledge and techniques alluded to in the descriptive paragraphs. Computer vision is the enterprise of automating and integrating a wide range of processes and representations used for vision perception. It includes as parts many techniques that are useful by themselves, such as *image processing* (transforming, encoding, and transmitting images) and *statistical pattern classification* (statistical decision theory applied to general patterns, visual or otherwise). More importantly for us, it includes techniques for geometric modeling and cognitive processing.

1.2 HIGH-LEVEL AND LOW-LEVEL CAPABILITIES

The examples of Section 1.1 illustrate vision that uses *cognitive processes*, *geometric models*, *goals*, and *plans*. These *high-level* processes are very important; our examples only weakly illustrate their power and scope. There surely would be some overall purpose to finding ships; there might be collateral information that there were submarines, barges, or small craft in the harbor, and so forth. CAT scans would be used with several diagnostic goals in mind and an associated medical history available. Goals and knowledge are high-level capabilities that can guide visual activities, and a visual system should be able to take advantage of them.

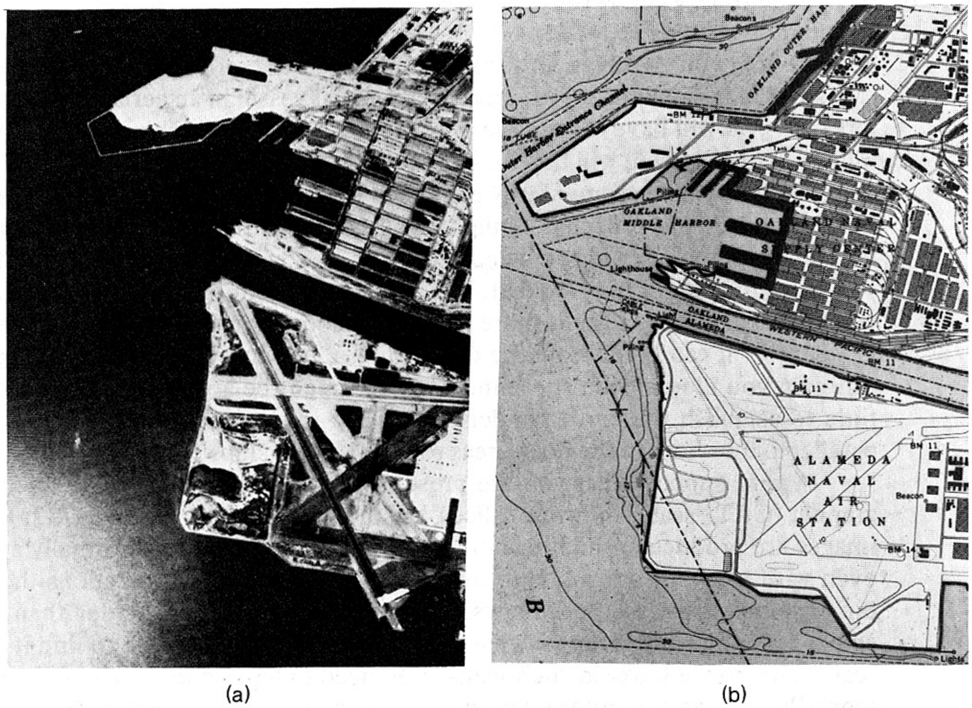
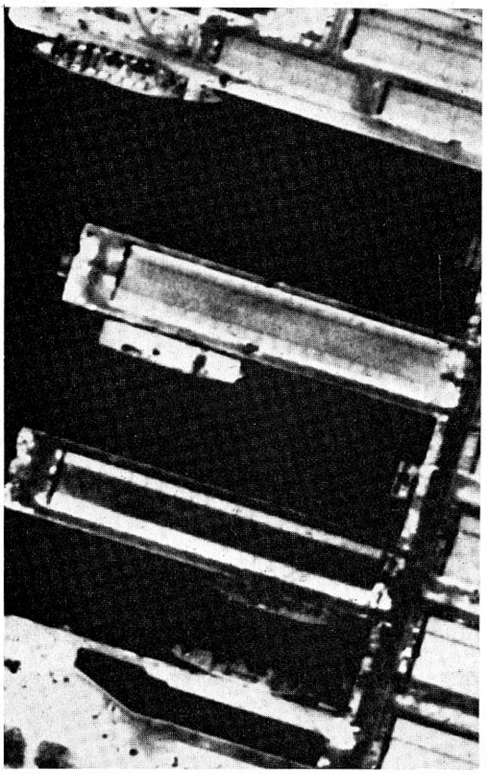


Fig. 1.1 Finding ships in an aerial photograph. (a) The photograph; (b) a corresponding map; (c) the dock area of the photograph; (d) registered map and image, with ship location.


(c)

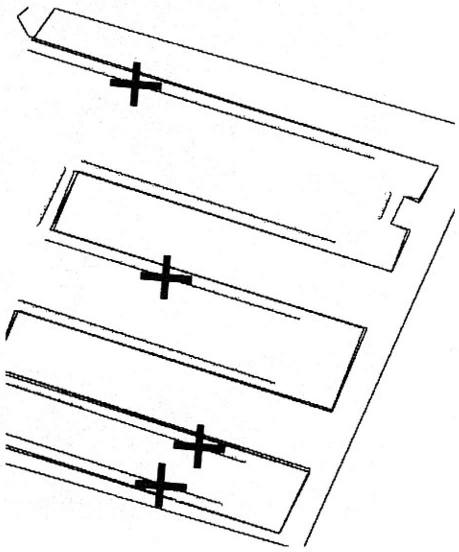

(d)

Fig. 1.1 (cont.)

Even such elaborated tasks are very special ones and in their way easier to think about than the commonplace visual perceptions needed to pick up a baby, cross a busy street, or arrive at a party and quickly "see" who you know, your host's taste in decor, and how long the festivities have been going on. All these tasks require judgment and large amounts of knowledge of objects in the world, how they look, and how they behave. Such high-level powers are so well integrated into "vision" as to be effectively inseparable.

Knowledge and goals are only part of the vision story. Vision requires many *low-level* capabilities we often take for granted; for example, our ability to extract *intrinsic images* of "lightness," "color," and "range." We perceive black as black in a complex scene even when the lighting is such that some black patches are reflecting more light than some white patches. Similarly, perceived colors are not related simply to the wavelengths of reflected light; if they were, we would consciously see colors changing with illumination. Stereo fusion (stereopsis) is a low-level facility basic to short-range three-dimensional perception.

An important low-level capability is *object perception*: for our purposes it does not really matter if this talent is innate, ("hard-wired"), or if it is developmental or even learned ("compiled-in"). The fact remains that mature biological vision systems are specialized and tuned to deal with the relevant objects in their environ-

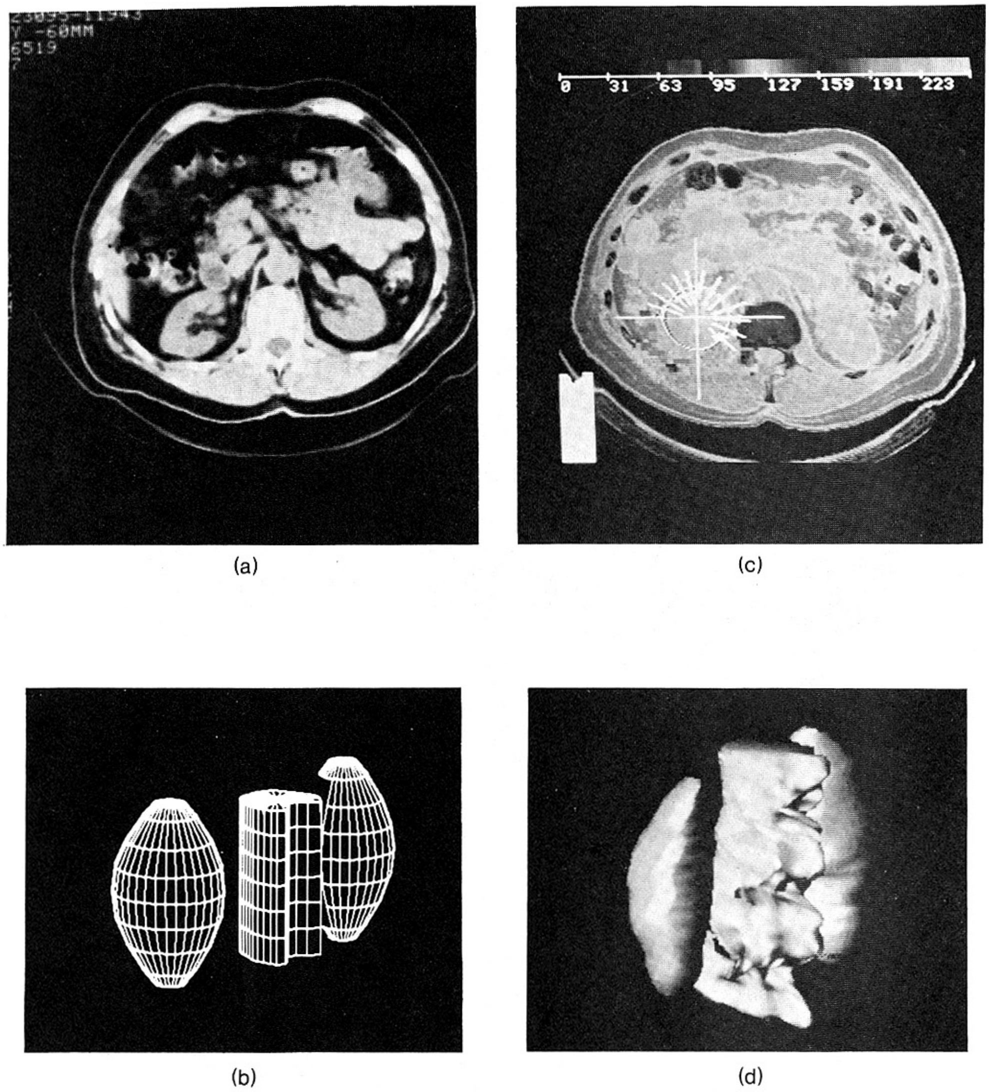


Fig. 1.2 Finding a kidney in a computer-aided tomographic scan. (a) One slice of scan data; (b) prototype kidney model; (c) model fitting; (d) resulting kidney and spinal cord instances.

ments. Further specialization can often be learned, but it is built on basic immutable assumptions about the world which underlie the vision system.

A basic sort of object recognition capability is the "figure/ground" discrimination that separates objects from the "background." Other basic organizational predispositions are revealed by the "Gestalt laws" of clustering, which demonstrate rules our vision systems use to form simple arrays of stimuli into more coherent spatial groups. A dramatic example of specialized object perception for human beings is revealed in our "face recognition" capability, which seems to occupy a large volume of brain matter. Geometric visual illusions are more surprising symptoms of nonintuitive processing that is performed by our vision systems, either for some direct purpose or as a side effect of its specialized architecture. Some other illusions clearly reflect the intervention of high-level knowledge. For instance, the familiar "Necker cube reversal" is grounded in our three-dimensional models for cubes.

Low-level processing capabilities are elusive; they are unconscious, and they are not well connected to other systems that allow direct introspection. For instance, our visual memory for images is quite impressive, yet our quantitative verbal descriptions of images are relatively primitive. The biological visual "hardware" has been developed, honed, and specialized over a very long period. However, its organization and functionality is not well understood except at extreme levels of detail and generality—the behavior of small sets of cat or monkey cortical cells and the behavior of human beings in psychophysical experiments.

Computer vision is thus immediately faced with a very difficult problem; it must reinvent, with general digital hardware, the most basic and yet inaccessible talents of specialized, parallel, and partly analog biological visual systems. Figure 1.3 may give a feeling for the problem; it shows two visual renditions of a familiar subject. The inset is a normal image, the rest is a plot of the intensities (gray levels) in the image against the image coordinates. In other words, it displays information

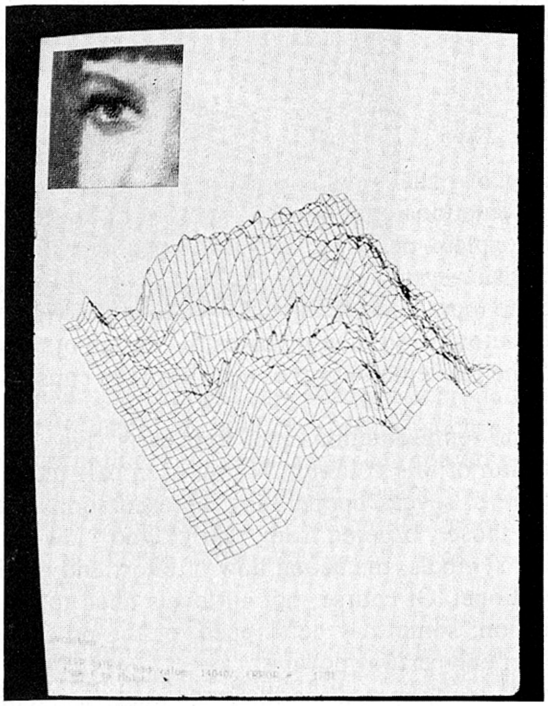


Fig. 1.3 Two representations of an image. One is directly accessible to our low-level processes; the other is not.

with "height" instead of "light." No information is lost, and the display is an image-like object, but we do not immediately see a face in it. The initial representation the computer has to work with is no better; it is typically just an array of numbers from which human beings could extract visual information only very painfully. Skipping the low-level processing we take for granted turns normally effortless perception into a very difficult puzzle.

Computer vision is vitally concerned with both low-level or "early processing" issues and with the high-level and "cognitive" use of knowledge. Where does vision leave off and reasoning and motivation begin? We do not know precisely, but we firmly believe (and hope to show) that powerful, cooperating, rich representations of the world are needed for any advanced vision system. Without them, no system can derive relevant and invariant information from input that is beset with ever-changing lighting and viewpoint, unimportant shape differences, noise, and other large but irrelevant variations. These representations can remove some computational load by predicting or assuming structure for the visual world.

Finally, if a system is to be successful in a variety of tasks, it needs some "meta-level" capabilities: it must be able to model and reason about its own goals and capabilities, and the success of its approaches. These complex and related models must be manipulated by cognitive-like techniques, even though introspectively the perceptual process does not always "feel" to us like cognition.

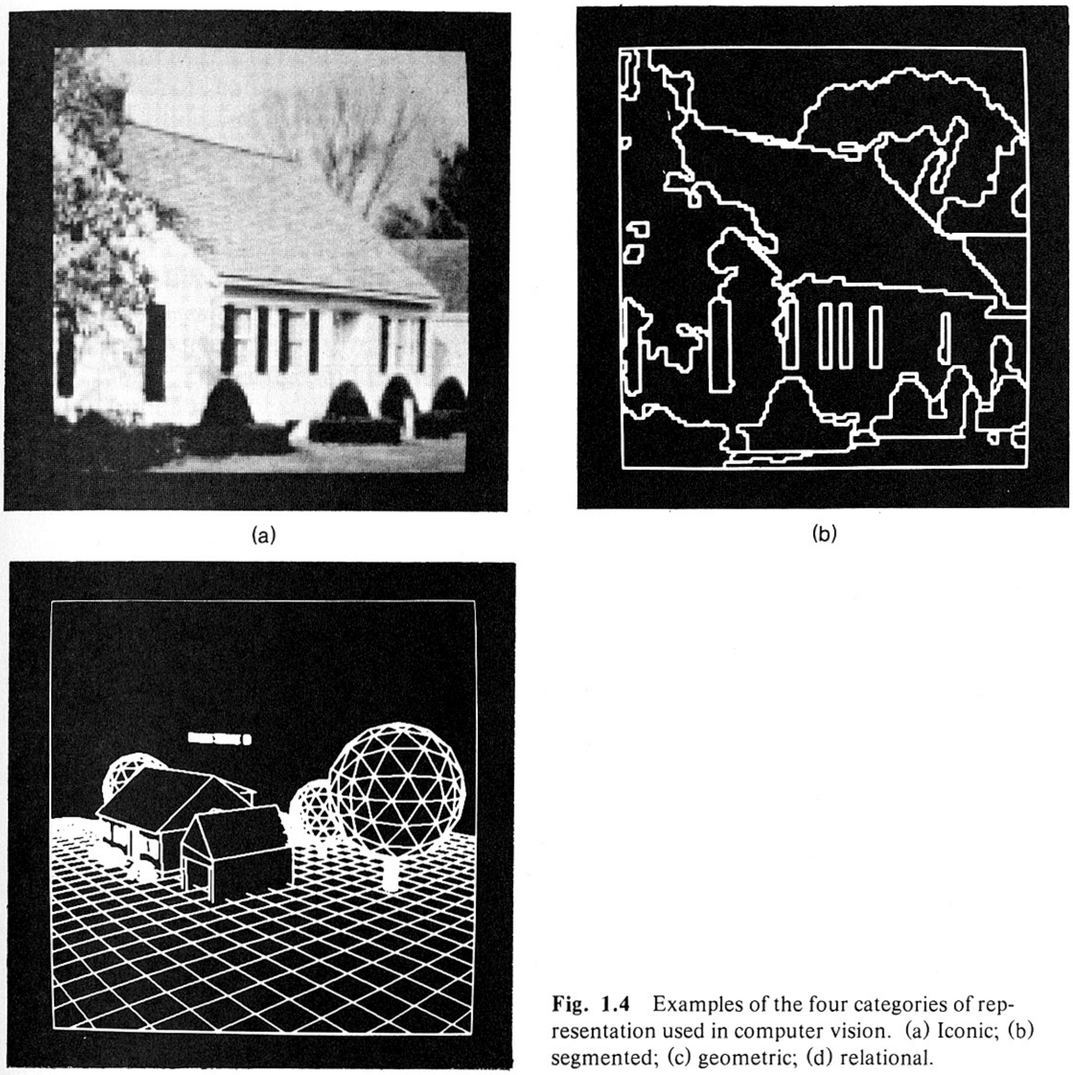
Computer Vision Systems

1.3 A RANGE OF REPRESENTATIONS

Visual perception is the relation of visual input to previously existing *models* of the world. There is a large representational gap between the image and the models ("ideas," "concepts") which explain, describe, or abstract the image information. To bridge that gap, computer vision systems usually have a (loosely ordered) *range of representations* connecting the input and the "output" (a final description, decision, or interpretation). Computer vision then involves the design of these intermediate representations and the implementation of algorithms to construct them and relate them to one another.

We broadly categorize the representations into four parts (Fig. 1.4) which correspond with the organization of this volume. Within each part there may be several layers of representation, or several cooperating representations. Although the sets of representations are loosely ordered from "early" and "low-level" *signals* to "late" and "*cognitive*" symbols, the actual flow of effort and information between them is not unidirectional. Of course, not all levels need to be used in each computer vision application; some may be skipped, or the processing may start partway up the hierarchy or end partway down it.

Generalized images (Part I) are *iconic* (image-like) and *analogical* representations of the input data. Images may initially arise from several technologies.


(a)

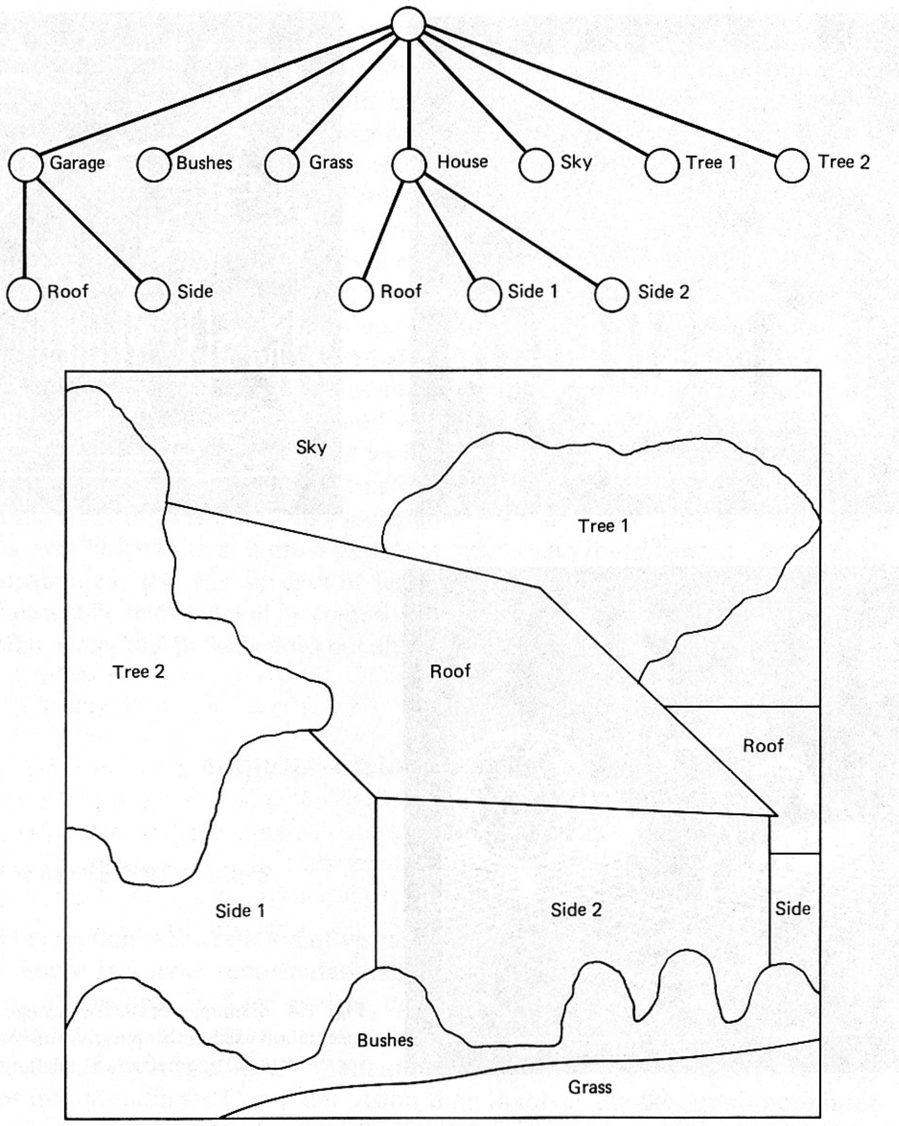
(b)

(c)

Fig. 1.4 Examples of the four categories of representation used in computer vision. (a) Iconic; (b) segmented; (c) geometric; (d) relational.

Domain-independent processing can produce other iconic representations more directly useful to later processing, such as arrays of *edge elements* (gray-level discontinuities). *Intrinsic images* can sometimes be produced at this level—they reveal physical properties of the imaged scene (such as surface orientations, range, or surface reflectance). Often *parallel processing* can produce generalized images. More generally, most "low-level" processes can be implemented with parallel computation.

Segmented images (Part II) are formed from the generalized image by gathering its elements into sets likely to be associated with meaningful *objects* in the scene. For instance, segmenting a scene of planar polyhedra (blocks) might result in a set of *edge segments* corresponding to polyhedral edges, or a set of two-



(d)

Fig. 1.4 (cont.)

dimensional *regions* in the image corresponding to polyhedral faces. In producing the segmented image, knowledge about the particular domain at issue begins to be important both to save computation and to overcome problems of *noise* and inadequate data. In the planar polyhedral example, it helps to know beforehand that the line segments must be straight. *Texture* and *motion* are known to be very important in segmentation, and are currently topics of active research; knowledge in these areas is developing very fast.

Geometric representations (Part III) are used to capture the all-important idea of two-dimensional and three-dimensional *shape*. Quantifying shape is as important as it is difficult. These geometric representations must be powerful enough to support complex and general processing, such as "simulation" of the effects of lighting and motion. Geometric structures are as useful for encoding previously acquired knowledge as they are for re-representing current visual input. Computer vision requires some basic mathematics; Appendix 1 has a brief selection of useful techniques.

Relational models (Part IV) are complex assemblages of representations used to support sophisticated high-level processing. An important tool in *knowledge representation* is *semantic nets*, which can be used simply as an organizational convenience or as a formalism in their own right. High-level processing often uses prior knowledge and models acquired prior to a perceptual experience. The basic mode of processing turns from *constructing* representations to *matching* them. At high levels, *propositional* representations become more important. They are made up of assertions that are true or false with respect to a model, and are manipulated by rules of *inference*. Inference-like techniques can also be used for *planning*, which models situations and actions through time, and thus must reason about temporally varying and hypothetical worlds. The higher the level of representation, the more marked is the flow of *control* (direction of attention, allocation of effort) downward to lower levels, and the greater the tendency of algorithms to exhibit *serial processing*. These issues of control are basic to complex information processing in general and computer vision in particular; Appendix 2 outlines some specific control mechanisms.

Figure 1.5 illustrates the loose classification of the four categories into analogical and propositional representations. We consider generalized and segmented images as well as geometric structures to be analogical models. Analogical models capture directly the relevant characteristics of the represented objects, and are manipulated and interrogated by simulation-like processes. Relational models are generally a mix of analogical and propositional representations. We develop this distinction in more detail in Chapter 10.

1.4 THE ROLE OF COMPUTERS

The computer is a congenial tool for research into visual perception.

- Computers are versatile and forgiving experimental subjects. They are easily and ethically reconfigurable, not messy, and their workings can be scrutinized in the finest detail.
- Computers are demanding critics. Imprecision, vagueness, and oversights are not tolerated in the computer implementation of a theory.
- Computers offer new metaphors for perceptual psychology (also neurology, linguistics, and philosophy). Processes and entities from computer science provide powerful and influential conceptual tools for thinking about perception and cognition.
- Computers can give precise measurements of the amount of processing they

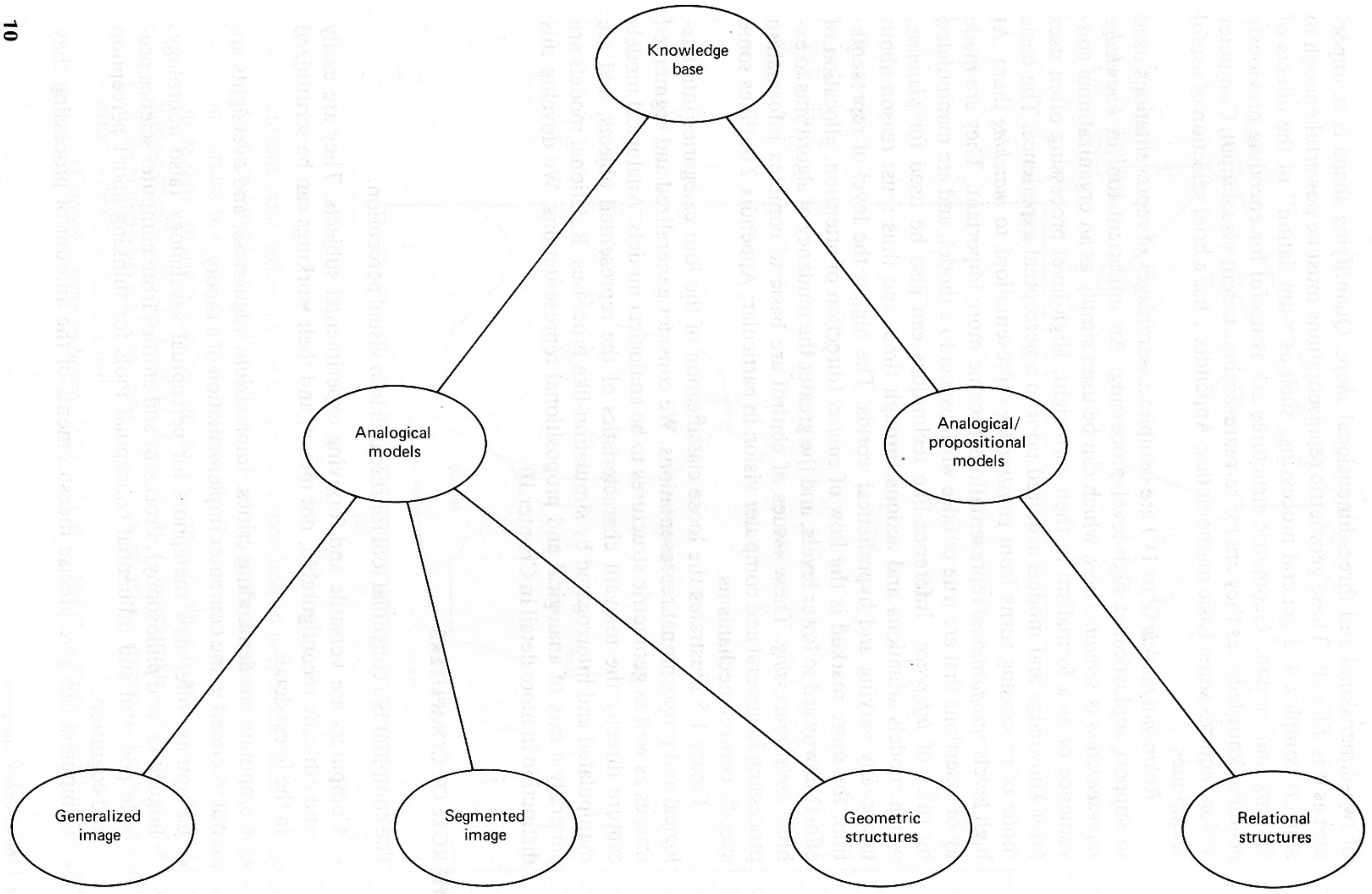


Fig. 1.5 The knowledge base of a complex computer vision system, showing four basic representational categories.

Table 1.1

EXAMPLES OF IMAGE ANALYSIS TASKS

Domain	*Objects*	*Modality*	*Tasks*	*Knowledge Sources*
Robotics	Three-dimensional outdoor scenes indoor scenes	Light X-rays	Identify or describe objects in scene Industrial tasks	Models of objects Models of the reflection of light from objects
	Mechanical parts	Light Structured light		
Aerial images	Terrain Buildings, etc.	Light Infrared Radar	Improved images Resource analyses Weather prediction Spying Missile guidance Tactical analysis	Maps Geometrical models of shapes Models of image formation
Astronomy	Stars Planets	Light	Chemical composition Improved images	Geometrical models of shapes
Medical Macro	Body organs	X-rays Ultrasound Isotopes Heat	Diagnosis of abnormalities Operative and treatment planning	Anatomical models Models of image formation
	Cells	Electronmicroscopy	Pathology, cytology	
Micro	Protein chains Chromosomes	Light	Karyotyping	Models of shape
Chemistry	Molecules	Electron densities	Analysis of molecular compositions	Chemical models Structured models
Neuroanatomy	Neurons	Light Electronmicroscopy	Determination of spatial orientation	Neural connectivity
Physics	Particle tracks	Light	Find new particles Identify tracks	Atomic physics

do. A computer implementation places an upper limit on the amount of computation necessary for a task.

- Computers may be used either to mimic what we understand about human perceptual architecture and processes, or to strike out in different directions to try to achieve similar ends by different means.
- Computer models may be judged either by their efficacy for applications and on-the-job performance or by their internal organization, processes, and structures—the theory they embody.

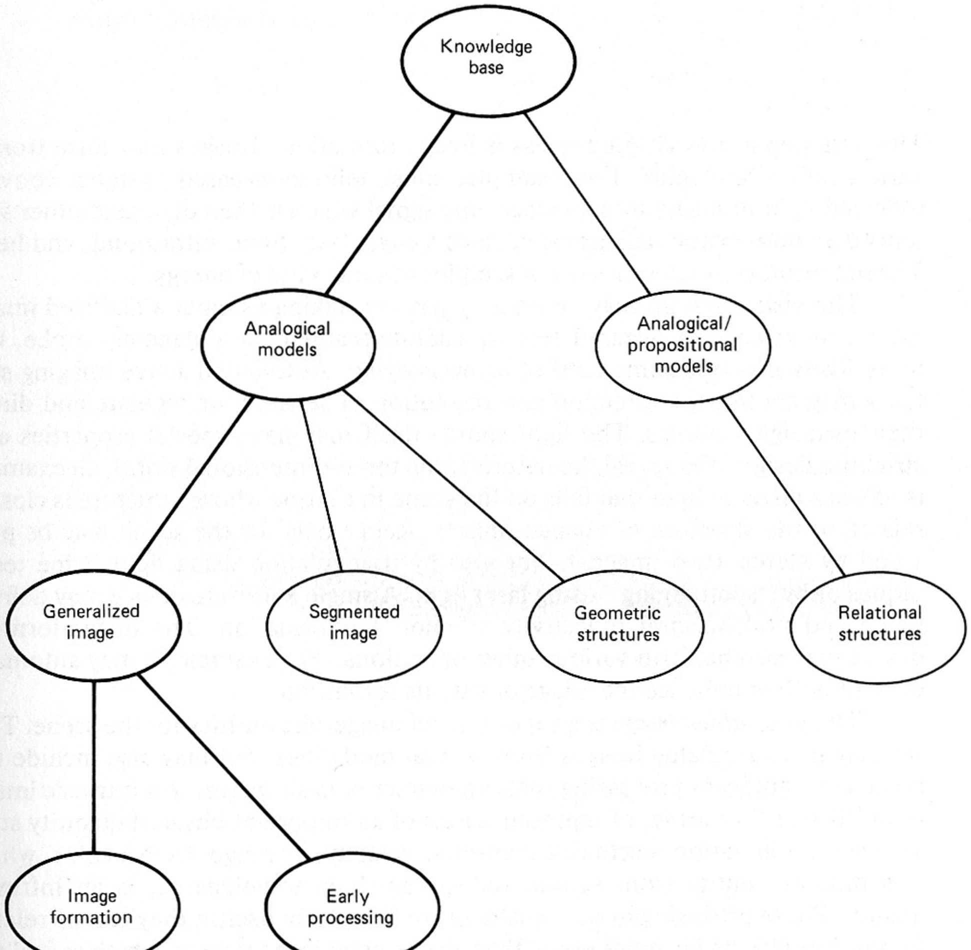
1.5 COMPUTER VISION RESEARCH AND APPLICATIONS

"Pure" computer vision research often deals with relatively domain-independent considerations. The results are useful in a broad range of contexts. Almost always such work is demonstrated in one or more applications areas, and more often than not an initial application problem motivates consideration of the general problem. Applications of computer vision are exciting, and their number is growing as computer vision becomes better understood. Table 1.1 gives a partial list of "classical" and current applications areas.

Within the organization outlined above, this book presents many specific ideas and techniques with general applicability. It is meant to provide enough basic knowledge and tools to support attacks on both applications and research topics.

GENERALIZED IMAGES

I



The first step in the vision process is image formation. Images may arise from a variety of technologies. For example, most television-based systems convert reflected light intensity into an electronic signal which is then digitized; other systems use more exotic radiations, such as x-rays, laser light, ultrasound, and heat. The net result is usually an array of samples of some kind of energy.

The vision system may be entirely passive, taking as input a digitized image from a microwave or infrared sensor, satellite scanner, or a planetary probe, but more likely involves some kind of *active imaging.* Automated active imaging systems may control the direction and resolution of sensors, or regulate and direct their own light sources. The light source itself may have special properties and structure designed to reveal the nature of the three-dimensional world; an example is to use a plane of light that falls on the scene in a stripe whose structure is closely related to the structure of opaque objects. Range data for the scene may be provided by stereo (two images), but also by triangulation using light-stripe techniques or by "spotranging" using laser light. A single hardware device may deliver range and multispectral reflectivity ("color") information. The image-forming device may also perform various other operations. For example, it may automatically smooth or enhance the image or vary its resolution.

The *generalized image* is a set of related image-like entities for the scene. This set may include related images from several modalities, but may also include the results of significant processing that can extract *intrinsic images.* An intrinsic image is an "image," or array, of representations of an important physical quantity such as surface orientation, occluding contours, velocity, or range. Object color, which is a different entity from sensed red—green—blue wavelengths, is an intrinsic quality. These intrinsic physical qualities are extremely useful; they can be related to physical objects far more easily than the original input values, which reveal the physical parameters only indirectly. An intrinsic image is a major step toward scene understanding and usually represents significant and interesting computations.

The information necessary to compute an intrinsic image is contained in the input image itself, and is extracted by "inverting" the transformation wrought by the imaging process, the reflection of radiation from the scene, and other physical processes. An example is the fusion of two stereo images to yield an intrinsic range image. Many algorithms to recover intrinsic images can be realized with *parallel* implementations, mirroring computations that may take place in the lower neurological levels of biological image processing.

All of the computations listed above benefit from the idea of *resolution pyramids.* A pyramid is a generalized image data structure consisting of the same image at several successively increasing levels of resolution. As the resolution increases, more samples are required to represent the increased information and hence the successive levels are larger, making the entire structure look like a pyramid. Pyramids allow the introduction of many different coarse-to-fine image-resolution algorithms which are vastly more efficient than their single-level, high-resolution-only counterparts.

Image Formation 2

2.1 IMAGES

Image formation occurs when a *sensor* registers *radiation* that has interacted with *physical objects.* Section 2.2 deals with mathematical models of images and image formation. Section 2.3 describes several specific image formation technologies.

The mathematical model of imaging has several different components.

1. An *image function* is the fundamental abstraction of an image.
2. A *geometrical model* describes how three dimensions are projected into two.
3. A *radiometrical model* shows how the imaging geometry, light sources, and reflectance properties of objects affect the light measurement at the sensor.
4. A *spatial* frequency model describes how spatial variations of the image may be characterized in a transform domain.
5. A *color model* describes how different spectral measurements are related to image colors.
6. A *digitizing model* describes the process of obtaining discrete samples.

This material forms the basis of much image-processing work and is developed in much more detail elsewhere, e.g., [Rosenfeld and Kak 1976; Pratt 1978]. Our goals are not those of image processing, so we limit our discussion to a summary of the essentials.

The wide range of possible sources of samples and the resulting different implications for later processing motivate our overview of specific imaging techniques. Our goal is not to provide an exhaustive catalog, but rather to give an idea of the range of techniques available. Very different analysis techniques may be needed depending on how the image was formed. Two examples illustrate this point. If the image is formed by reflected light intensity, as in a photograph, the image records both light from primary light sources and (more usually) the light reflected off physical surfaces. We show in Chapter 3 that in certain cases we can use these kinds of images together with knowledge about physics to derive the orientation of the surfaces. If, on the other hand, the image is a computed tomogram of the human body (discussed in Section 2.3.4), the image represents tissue density of internal organs. Here orientation calculations are irrelevant, but general segmentation techniques of Chapters 4 and 5 (the agglomeration of neighboring samples of similar density into units representing organs) are appropriate.

2.2 IMAGE MODEL

Sophisticated image models of a statistical flavor are useful in image processing [Jan 1981]. Here we are concerned with more geometrical considerations.

2.2.1 Image Functions

An *image function* is a mathematical representation of an image. Generally, an image function is a vector-valued function of a small number of arguments. A special case of the image function is the *digital (discrete) image function*, where the arguments to and value of the function are all integers. Different image functions may be used to represent the same image, depending on which of its characteristics are important. For instance, a camera produces an image on black-and-white film which is usually thought of as a real-valued function (whose value could be the density of the photographic negative) of two real-valued arguments, one for each of two spatial dimensions. However, at a very small scale (the order of the film grain) the negative basically has only two densities, "opaque" and "transparent."

Most images are presented by functions of two *spatial* variables $f(\mathbf{x}) = f(x, y)$, where $f(x, y)$ is the brightness of the gray level of the image at a spatial coordinate (x, y). A multispectral image $\mathbf{f}$ is a vector-valued function with components $(f_1 \ldots f_n)$. One special multispectral image is a color image in which, for example, the components measure the brightness values of each of three wavelengths, that is,

$$\mathbf{f}(\mathbf{x}) = \left\{ f_{\text{red}}(\mathbf{x}), f_{\text{blue}}(\mathbf{x}), f_{\text{green}}(\mathbf{x}) \right\}$$

Time-varying images $\mathbf{f}(\mathbf{x}, t)$ have an added temporal argument. For special three-dimensional images, $\mathbf{x} = (x, y, z)$. Usually, both the domain and range of $\mathbf{f}$ are bounded.

An important part of the formation process is the conversion of the image representation from a continuous function to a discrete function; we need some way of describing the images as samples at discrete points. The mathematical tool we shall use is the *delta function*.

Formally, the delta function may be defined by point. If the image is formed by reflected light intensity, as in a photograph, the image records both light from primary light sources and (more usually) the light reflected off physical surfaces. We show in Chapter 3 that in certain cases we can use these kinds of images together with knowledge about physics to derive the orientation of the surfaces. If, on the other hand, the image is a computed tomogram of the human body (discussed in Section 2.3.4), the image represents tissue density of internal organs. Here orientation calculations are irrelevant, but general segmentation techniques of Chapters 4 and 5 (the agglomeration of neighboring samples of similar density into units representing organs) are appropriate.

2.2 IMAGE MODEL

Sophisticated image models of a statistical flavor are useful in image processing [Jan 1981]. Here we are concerned with more geometrical considerations.

2.2.1 Image Functions

An *image function* is a mathematical representation of an image. Generally, an image function is a vector-valued function of a small number of arguments. A special case of the image function is the *digital (discrete) image function*, where the arguments to and value of the function are all integers. Different image functions may be used to represent the same image, depending on which of its characteristics are important. For instance, a camera produces an image on black-and-white film which is usually thought of as a real-valued function (whose value could be the density of the photographic negative) of two real-valued arguments, one for each of two spatial dimensions. However, at a very small scale (the order of the film grain) the negative basically has only two densities, "opaque" and "transparent."

Most images are presented by functions of two *spatial* variables $f(\mathbf{x}) = f(x, y)$, where $f(x, y)$ is the brightness of the gray level of the image at a spatial coordinate (x, y). A multispectral image $\mathbf{f}$ is a vector-valued function with components $(f_1 \ldots f_n)$. One special multispectral image is a color image in which, for example, the components measure the brightness values of each of three wavelengths, that is,

$$\mathbf{f}(\mathbf{x}) = \left\{ f_{\text{red}}(\mathbf{x}), f_{\text{blue}}(\mathbf{x}), f_{\text{green}}(\mathbf{x}) \right\}$$

Time-varying images $\mathbf{f}(\mathbf{x}, t)$ have an added temporal argument. For special three-dimensional images, $\mathbf{x} = (x, y, z)$. Usually, both the domain and range of $\mathbf{f}$ are bounded.

An important part of the formation process is the conversion of the image representation from a continuous function to a discrete function; we need some way of describing the images as samples at discrete points. The mathematical tool we shall use is the *delta function*.

Formally, the delta function may be defined by

$$\delta(x) = \begin{cases} 0 \text{ when } x \neq 0 \\ \infty \text{ when } x = 0 \end{cases} \tag{2.1}$$

$$\int_{-\infty}^{\infty} \delta(x)\, dx = 1$$

If some care is exercised, the delta function may be interpreted as the limit of a set of functions:

$$\delta(x) = \lim_{n \to \infty} \delta_n(x)$$

wh^e^re

$$\delta_n(x) = \begin{cases} n & \text{if } |x| < \dfrac{1}{2n} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

A useful property of the delta function is the *sifting property:*

$$\int_{-\infty}^{\infty} f(x)\, \delta(x-a)\, dx = f(a) \tag{2.3}$$

A continuous image may be multipled by a two-dimensional "comb," or array of delta functions, to extract a finite number of discrete *samples* (one for each delta function). This mathematical model of the sampling process will be useful later.

2.2.2 Imaging Geometry

Monocular Imaging

Point projection is the fundamental model for the transformation wrought by our eye, by cameras, or by numerous other imaging devices. To a first-order approximation, these devices act like a pinhole camera in that the image results from projecting scene points through a single point onto an *image plane* (see Fig. 2.1). In Fig. 2.1, the image plane is behind the point of projection, and the image is reversed. However, it is more intuitive to recompose the geometry so that the point of projection corresponds to a *viewpoint* behind the image plane, and the image occurs right side up (Fig. 2.2). The mathematics is the same, but now the viewpoint is $+f$ on the z axis, with $z = 0$ plane being the image plane upon which the image is projected. (f is sometimes called the *focal length* in this context. The use of f in this section should not be confused with the use of f for image function.) As the imaged object approaches the viewpoint, its projection gets bigger (try moving your hand toward your eye). To specify how its imaged size changes, one needs only the geometry of similar triangles. In Fig. 2.2b y', the projected height of the object, is related to its real height y, its position z, and the focal length f by

$$\frac{y}{f-z} = \frac{y'}{f} \tag{2.4}$$

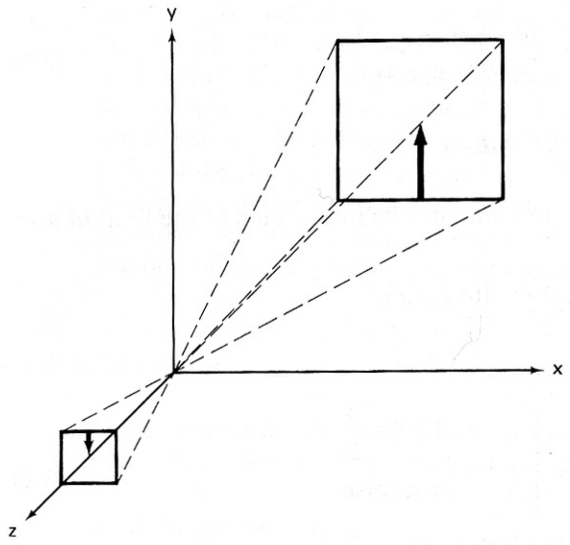


Fig. 2.1 A geometric camera model.

The case for x' is treated similarly:

$$\frac{x}{f-z} = \frac{x'}{f} \tag{2.5}$$

The projected image has $z = 0$ everywhere. However, projecting away the z component is best considered a separate transformation; the projective transform is usually thought to distort the z component just as it does the x and y. *Perspective distortion* thus maps (x, y, z) to

$$(x', y', z') = \left(\frac{fx}{f-z}, \frac{fy}{f-z}, \frac{fz}{f-z}\right) \tag{2.6}$$

The perspective transformation yields *orthographic projection* as a special case when the viewpoint is the *point at infinity* in the z direction. Then all objects are projected onto the viewing plane with no distortion of their x and y coordinates.

The perspective distortion yields a three-dimensional object that has been "pushed out of shape"; it is more shrunken the farther it is from the viewpoint. The z component is not available directly from a two-dimensional image, being identically equal to zero. In our model, however, the distorted z component has information about the distance of imaged points from the viewpoint. When this distorted object is projected orthographically onto the image plane, the result is a perspective picture. Thus, to achieve the effect of railroad tracks appearing to come together in the distance, the perspective distortion transforms the tracks so that they *do* come together (at a point at infinity)! The simple orthographic projection that projects away the z component unsurprisingly preserves this distortion. Several properties of the perspective transform are of interest and are investigated further in Appendix 1.

Binocular Imaging

Basic binocular imaging geometry is shown in Fig. 2.3a. For simplicity, we

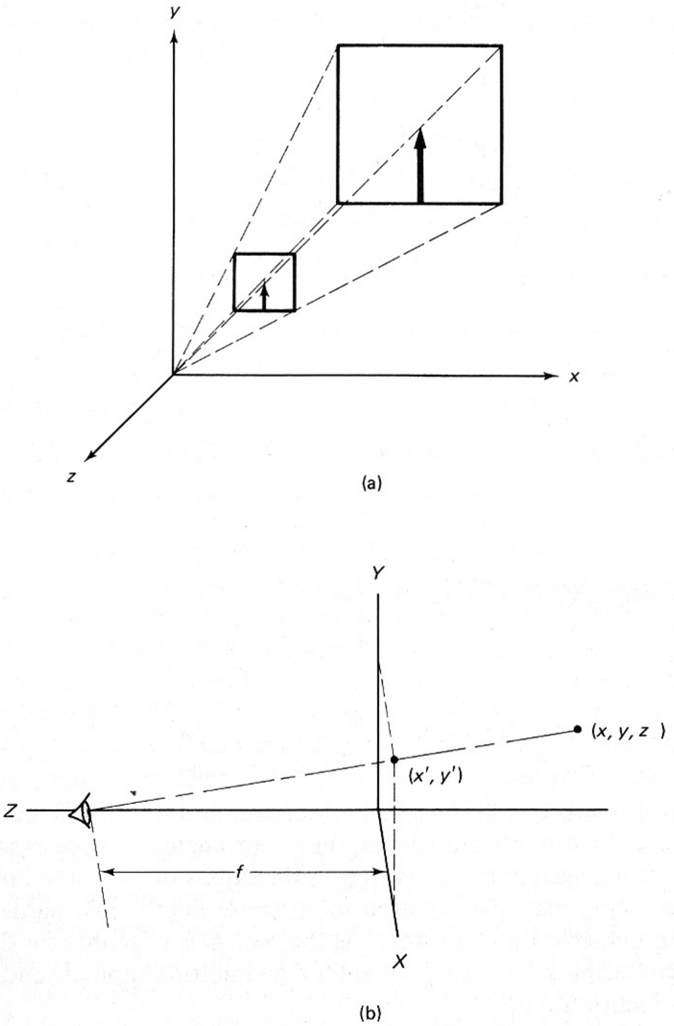


Fig. 2.2 (a) Camera model equivalent to that of Fig. 2.1; (b) definition of terms.

use a system with two viewpoints. In this model the eyes do not *converge*; they are aimed in parallel at the point at infinity in the $-z$ direction. The depth information about a point is then encoded only by its different positions (*disparity*) in the two image planes.

With the stereo arrangement of Fig. 2.3,

$$x' = \frac{(x-d)f}{f-z}$$

$$x'' = \frac{(x+d)f}{f-z}$$

where (x', y') and (x'', y'') are the retinal coordinates for the world point imaged

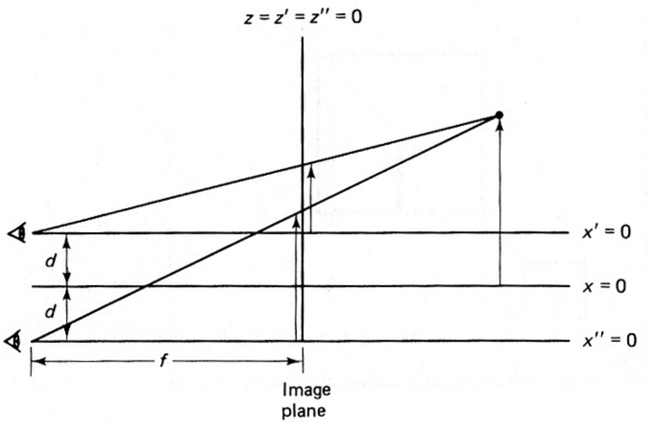


Fig. 2.3 A nonconvergent binocular imaging system.

through each eye. The *baseline* of the binocular system is $2d$. Thus

$$(f - z)x' = (x - d)f \tag{2.7}$$

$$(f - z)x'' = (x + d)f \tag{2.8}$$

Subtracting (2.7) from (2.8) gives

$$(f - z)(x'' - x') = 2df$$

or

$$z = f - \frac{2df}{x'' - x'} \tag{2.9}$$

Thus if points can be matched to determine the disparity $(x'' - x')$ and the baseline and focal length are known, the z coordinate is simple to calculate.

If the system can converge its directions of view to a finite distance, convergence angle may also be used to compute depth. The hardest part of extracting depth information from stereo is the *matching* of points for disparity calculations. "Light striping" is a way to maintain geometric simplicity and also simplify matching (Section 2.3.3).

2.2.3 Reflectance

Terminology

A basic aspect of the imaging process is the physics of the reflectance of objects, which determines how their "brightness" in an image depends on their inherent characteristics and the geometry of the imaging situation. A clear presentation of the mathematics of reflectance is given in [Horn and Sjoberg 1978; Horn 1977]. Light *energy flux* Φ is measured in watts; "brightness" is measured with respect to area and solid angle. The *radiant intensity* I of a source is the exitant flux per unit solid angle:

$$I = \frac{d\Phi}{d\omega} \qquad \text{watts/steradian} \tag{2.10}$$

Here $d\omega$ is an incremental solid angle. The solid angle of a small area dA measured perpendicular to a radius r is given by

$$d\omega = \frac{dA}{r^2} \tag{2.11}$$

in units of steradians. (The total solid angle of a sphere is 4π.)

The *irradiance* is flux incident on a surface element dA:

$$E = \frac{d\Phi}{dA} \qquad \text{watts/meter}^2 \tag{2.12}$$

and the flux exitant from the surface is defined in terms of the *radiance* L, which is the flux emitted per unit foreshortened surface area per unit solid angle:

$$L = \frac{d^2\Phi}{dA\ \cos\theta\, d\omega} \qquad \text{watts/(meter}^2\text{ steradian)} \tag{2.13}$$

where θ is the angle between the surface normal and the direction of emission.

Image irradiance f is the "brightness" of the image at a point, and is proportional to scene radiance. A "gray-level" is a quantized measurement of image irradiance. Image irradiance depends on the reflective properties of the imaged surfaces as well as on the illumination characteristics. How a surface reflects light depends on its micro-structure and physical properties. Surfaces may be *matte* (dull, flat), *specular* (mirrorlike), or have more complicated reflectivity characteristics (Section 3.5.1). The *reflectance* r of a surface is given quite generally by its Bidirectional Reflectance Distribution Function (BRDF) [Nicodemus et al. 1977]. The BRDF is the ratio of reflected radiance in the direction towards the viewer to the irradiance in the direction towards a small area of the source.

Effects of Geometry on an Imaging System

Let us now analyze a simple image-forming system shown in Fig. 2.4 with the objective of showing how the gray levels are related to the radiance of imaged objects. Following [Horn and Sjoberg 1978], assume that the imaging device is properly focused; rays originating in the infinitesimal area dA_o on the object's surface are projected into some area dA_p in the image plane and no rays from other portions of the object's surface reach this area of the image. The system is assumed to be an ideal one, obeying the laws of simple geometrical optics.

The energy flux/unit area that impinges on the sensor is defined to be E_p. To show how E_p is related to the scene radiance L, first consider the flux arriving at the lens from a small surface area dA_o. From (2.13) this is given as

$$d\Phi = dA_o \int L \cos\theta\, d\omega \tag{2.14}$$

This flux is assumed to arrive at an area dA_p in the imaging plane. Hence the irradiance is given by [using Eq. (2.12)]

$$E_p = \frac{d\Phi}{dA_p} \tag{2.15}$$

Now relate dA_o to dA_p by equating the respective solid angles as seen from the lens; that is [making use of Eq. (2.12)],

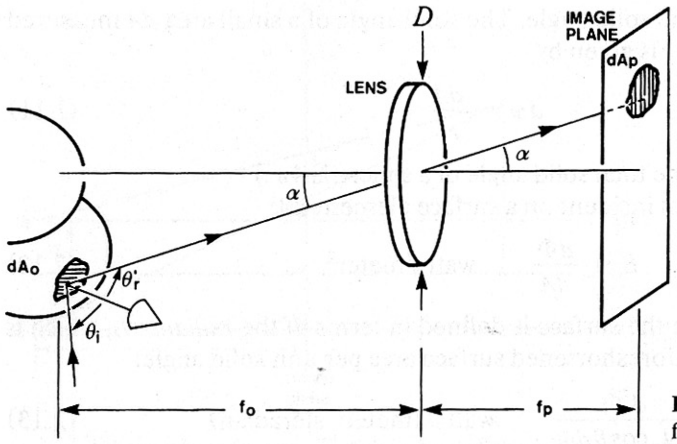


Fig. 2.4 Geometry of an image forming system.

$$dA_o \frac{\cos\theta}{f_o^2} = dA_p \frac{\cos\alpha}{f_p^2} \tag{2.16}$$

Substituting Eqs. (2.16) and (2.14) into (2.15) gives

$$E = \cos\alpha \left(\frac{f_o}{f_p} \right)^2 \int L d\omega \tag{2.17}$$

The integral is over the solid angle seen by the lens. In most instances we can assume that L is constant over this angle and hence can be removed from the integral. Finally, approximate $d\omega$ by the area of the lens foreshortened by $\cos\alpha$, that is, $(\pi/4)D^2 \cos\alpha$ divided by the distance $f_o/\cos\alpha$ squared:

$$d\omega = \frac{\pi}{4} D^2 \frac{\cos^3\alpha}{f_o^2} \tag{2.18}$$

so that finally

$$E = \frac{1}{4} \left(\frac{D}{f_p} \right)^2 \cos^4\alpha \, \pi L \tag{2.19}$$

The interesting results here are that (1) the image irradiance is proportional to the scene radiance L, and (2) the factor of proportionality includes the fourth power of the off-axis angle α. Ideally, an imaging device should be calibrated so that the variation in sensitivity as a function of α is removed.

2.2.4 Spatial Properties

The Fourier Transform

An image is a spatially varying function. One way to analyze spatial variations is the decomposition of an image function into a set of orthogonal functions, one such set being the Fourier (sinusoidal) functions. The Fourier transform may be used to transform the intensity image into the domain of *spatial frequency*. For notational convenience and intuition, we shall generally use as an example the continuous one-dimensional Fourier transform. The results can readily be extended to the discrete case and also to higher dimensions [Rosenfeld and Kak 1976]. In two dimensions we shall denote transform domain coordinates by (u, v). The one-dimensional Fourier transform, denoted $\mathfrak{F}$, is defined by

$$\mathfrak{F}\,[f(x)] = F(u)$$

where

$$F(u) = \int_{-\infty}^{+\infty} f(x) \exp(-j2\pi ux)\,dx \tag{2.20}$$

where $j = \sqrt{(-1)}$. Intuitively, Fourier analysis expresses a function as a sum of sine waves of different frequency and phase. The Fourier transform has an *inverse* $^{-1}[F(u)] = f(x)$. This inverse is given by

$$f(x) = \int_{-\infty}^{\infty} F(u) \exp(j2\pi ux)\,du \tag{2.21}$$

The transform has many useful properties, some of which are summarized in Table 2.1. Common one-dimensional Fourier transform pairs are shown in Table 2.2.

The transform F(u) is simply another representation of the image function. Its meaning can be understood by interpreting Eq. (2.21) for a specific value of x, say x_0:

$$f(x_0) = \int F(u) \exp(j2\pi ux_0)\,du \tag{2.22}$$

This equation states that a particular point in the image can be represented by a weighted sum of complex exponentials (sinusoidal patterns) at different spatial frequencies u. $F(u)$ is thus a *weighting function* for the different frequencies. Low-spatial frequencies account for the "slowly" varying gray levels in an image, such as the variation of intensity over a continuous surface. High-frequency components are associated with "quickly varying" information, such as edges. Figure 2.5 shows the Fourier transform of an image of rectangles, together with the effects of removing low- and high-frequency components.

The Fourier transform is defined above to be a continuous transform. Although it may be performed instantly by optics, a discrete version of it, the "fast Fourier transform," is almost universally used in image processing and computer vision. This is because of the relative versatility of manipulating the transform in the digital domain as compared to the optical domain. Image-processing texts, e.g., [Pratt 1978; Gonzalez and Wintz 1977] discuss the FFT in some detail; we content ourselves with an algorithm for it (Appendix 1).

The Convolution Theorem

Convolution is a very important image-processing operation, and is a basic operation of linear systems theory. The convolution of two functions f and g is a function h of a displacement y defined as

$$h(y) = f * g = \int_{-\infty}^{\infty} f(x) g(y - x)\,dx \tag{2.23}$$

Table 2.1

PROPERTIES OF THE FOURIER TRANSFORM

	Spatial Domain	*Frequency Domain*
	$f(x)$ $g(x)$	$F(u) = \mathfrak{F}[f(x)]$ $G(u) = \mathfrak{F}[g(x)]$
(1)	Linearity $c_1 f(x) + c_2 g(x)$ c_1, c_2 scalars	$c_1 F(u) + c_2 G(u)$
(2)	Scaling $f(ax)$	$\frac{1}{\lvert a \rvert} F\left(\frac{u}{a}\right)$
(3)	Shifting $f(x - x_0)$	$e^{-2\pi j x_0} F(u)$
(4)	Symmetry $F(x)$	$f(-u)$
(5)	Conjugation $f^*(x)$	$F^*(-u)$
(6)	Convolution $h(x) = f * g = \int_{-\infty}^{\infty} f(x') g(x - x')\, dx'$	$F(u) G(u)$
(7)	Differentiation $\frac{d^n f(x)}{dx^n}$	$(2\pi j u)^n F(u)$

Parseval's theorem:

$$\int_{-\infty}^{\infty} |f(x)|^2 dx = \int_{-\infty}^{\infty} |F(\xi)|^2 d\xi$$

$$\int_{-\infty}^{\infty} f(x) g^*(x)\, dx = \int_{-\infty}^{\infty} F(\xi) G^*(\xi)\, d\xi$$

$f(x)$	$F(\xi)$
Real(R)	Real part even (RE) Imaginary part odd (IO)
Imaginary (I)	RO,IE
RE,IO	R
RE,IE	I
RE	RE
RO	IO
IE	IE
IO	RO
Complex even (CE)	CE
CO	CO

Table 2.2

FOURIER TRANSFORM PAIRS

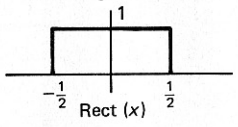
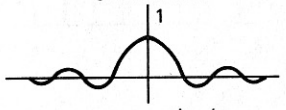
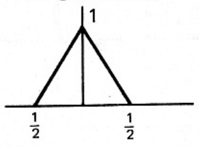
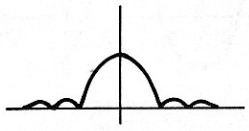
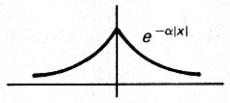
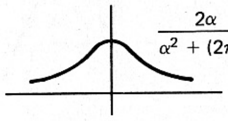
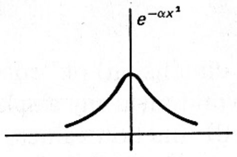
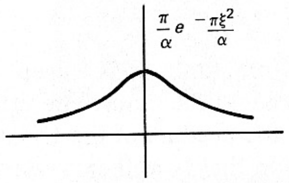
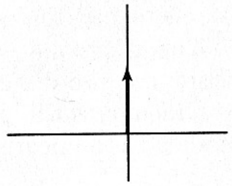
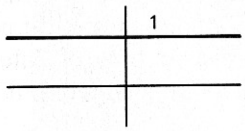
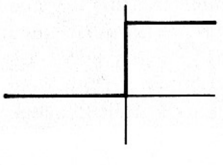
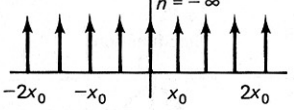
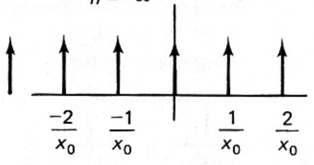
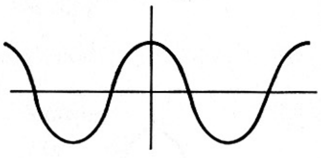
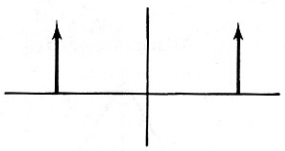
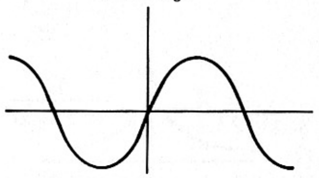
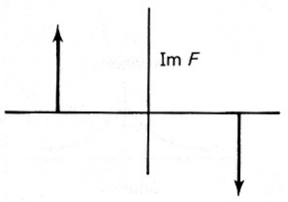
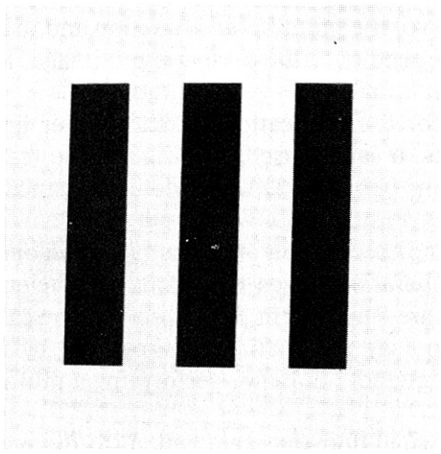
$f(x)$	$F(\xi)$
Rectangle function  Rect (x)	Sinc function  $\text{Sinc}(\xi) = \dfrac{\sin \pi\xi}{\pi\xi}$
Triangle function 	 $\text{Sinc}^2\ (\xi)$
Exponential  $e^{-\alpha\lvert x\rvert}$	 $\dfrac{2\alpha}{\alpha^2 + (2\pi\xi)^2}$
Gaussian  $e^{-\alpha x^2}$	 $\frac{\pi}{\alpha} e^{-\frac{\pi\xi^2}{\alpha}}$
Unit impulse $\delta(x)$ 	 1
Unit step 	$\frac{1}{2}\,\delta\,(\xi) + \dfrac{1}{2\pi j\xi}$

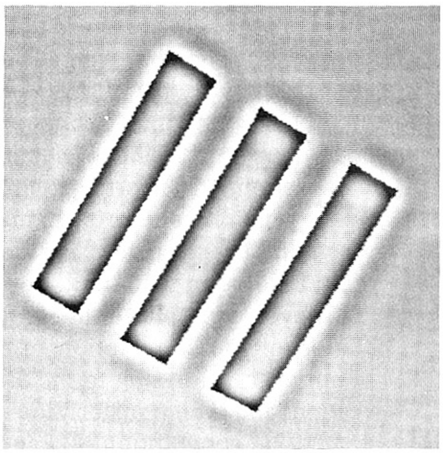
Table 2.2 (cont.)

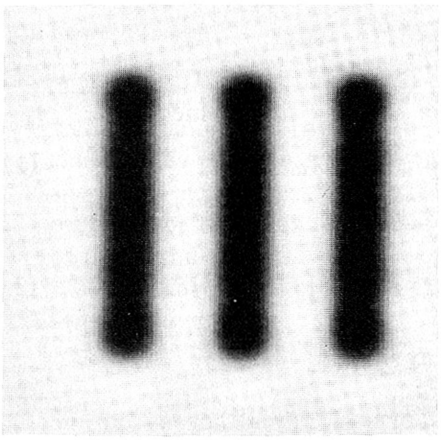
Comb function $\sum_{n=-\infty}^{\infty} \delta(x - nx_0)$ 	$\frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta(\xi - \frac{n}{x_0})$  $\frac{-2}{x_0}$ $\frac{-1}{x_0}$ $\frac{1}{x_0}$ $\frac{2}{x_0}$
$\cos 2\pi\omega_0 x$ 	$\frac{1}{2}[\delta(\xi - \omega_0) + \delta(\xi + \omega_0)]$ 
$\sin 2\pi\omega_0 x$ 	$\frac{1}{2} i [-\delta(\xi - \omega_0) + \delta(\xi + \omega_0)]$ 

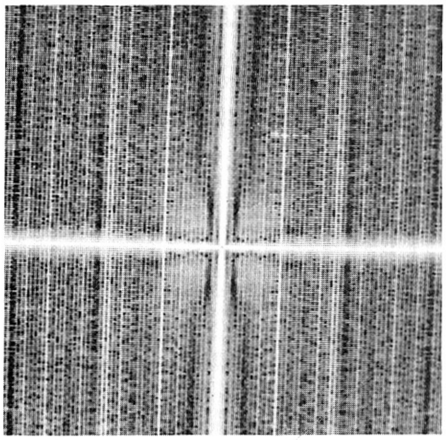
Intuitively, one function is "swept past" (in one dimension) or "rubbed over" (in two dimensions) the other. The value of the convolution at any displacement is the integral of the product of the (relatively displaced) function values. One common phenomenon that is well expressed by a convolution is the formation of an image by an optical system. The system (say a camera) has a "point-spread function," which is the image of a single point. (In linear systems theory, this is the "impulse response," or response to a delta-function input.) The ideal point-spread function is, of course, a point. A typical point-spread function is a two-dimensional Gaussian spatial distribution of intensities, but may include such phenomena as diffraction rings. In any event, if the camera is modeled as a linear system (ignor-

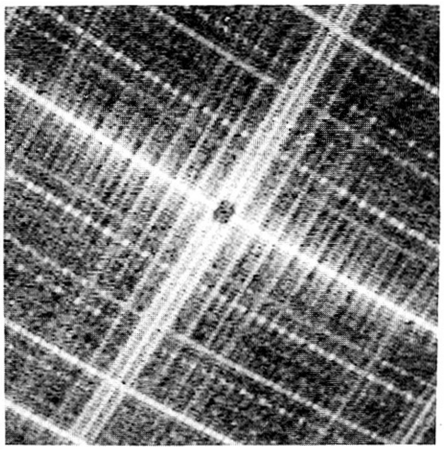
Fig. 2.5 (on facing page) (a) An image, $f(x, y)$. (b) A rotated version of (a), filtered to enhance high spatial frequencies. (c) Similar to (b), but filtered to enhance low spatial frequencies. (d), (e), and (f) show the logarithm of the power spectrum of (a), (b), and (c). The power spectrum is the log square modulus of the Fourier transform $F(u, v)$. Considered in polar coordinates (ρ, θ), points of small ρ correspond to low spatial frequencies ("slowly-varying" intensities), large ρ to high spatial frequencies contributed by "fast" variations such as step edges. The power at (ρ, θ) is determined by the amount of intensity variation at the frequency ρ occurring at the angle θ.

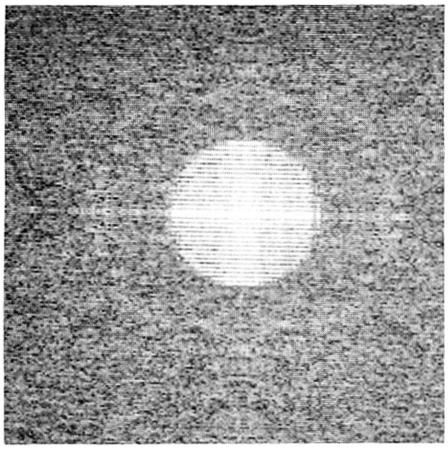

(a)


(b)


(c)


(d)


(e)


(f)

ing the added complexity that the point-spread function usually varies over the field of view), the image is the convolution of the point-spread function and the input signal. The point-spread function is rubbed over the perfect input image, thus blurring it.

Convolution is also a good model for the application of many other linear operators, such as line-detecting templates. It can be used in another guise (called correlation) to perform matching operations (Chapter 3) which detect instances of subimages or features in an image.

In the spatial domain, the obvious implementation of the convolution operation involves a shift–multiply–integrate operation which is hard to do efficiently. However, multiplication and convolution are "transform pairs," so that the calculation of the convolution in one domain (say the spatial) is simplified by first Fourier transforming to the other (the frequency) domain, performing a multiplication, and then transforming back.

The convolution of f and g in the spatial domain is equivalent to the pointwise product of F and G in the frequency domain,

$$\mathfrak{F}(f*g) = FG \tag{2.24}$$

We shall show this in a manner similar to [Duda and Hart 1973]. First we prove the *shift theorem.* If the Fourier transform of $f(x)$ is $F(u)$, defined as

$$F(u) = \int_x f(x) \exp\left[-j2\pi(ux)\right]dx \tag{2.25}$$

then

$$\mathfrak{F}\left[f(x-a)\right] = \int_x f(x-a) \exp\left[-j2\pi(ux)\right]dx \tag{2.26}$$

changing variables so that $x' = x - a$ and $dx = dx'$

$$= \int_{x'} f(x') \exp\{-j2\pi[u(x'+a)]\}dx' \tag{2.27}$$

Now $\exp[-j2\pi u(x'+a)] = \exp(-j2\pi ua)\exp(-j2\pi ux')$, where the first term is a constant. This means that

$$\mathfrak{F}\left[f(x-a)\right] = \exp(-j2\pi ua)F(u) \qquad \text{(shift theorem)}$$

Now we are ready to show that $\mathfrak{F}[f(x)*g(x)] = F(u)G(u)$.

$$\mathfrak{F}(f*g) = \int_y \{\int_x f(x)g(y-x)\} \exp(-j2\pi uy)\; dx\; dy \tag{2.28}$$

$$= \int_x f(x)\{\int_y g(y-x) \exp(-j2\pi uy)\; dy\}\; dx \tag{2.29}$$

Recognizing that the terms in braces represent $\mathfrak{F}[g(y-x)]$ and applying the shift theorem, we obtain

$$\mathfrak{F}(f*g) = \int_x f(x)\exp(-j2\pi ux)G(u)\; dx \tag{2.30}$$

$$= F(u)G(u) \tag{2.31}$$

2.2.5 Color

Not all images are monochromatic; in fact, applications using multispectral images are becoming increasingly common (Section 2.3.2). Further, human beings intuitively feel that color is an important part of their visual experience, and is useful or even necessary for powerful visual processing in the real world. Color vision provides a host of research issues, both for psychology and computer vision. We briefly discuss two aspects of color vision: color spaces and color perception. Several models of the human visual system not only include color but have proven useful in applications [Granrath 1981].

Color Spaces

Color spaces are a way of organizing the colors perceived by human beings. It happens that weighted combinations of stimuli at three principal wavelengths are sufficient to define almost all the colors we perceive. These wavelengths form a natural basis or coordinate system from which the color measurement process can be described. Color perception is not related in a simple way to color measurement, however.

Color is a perceptual phenomenon related to human response to different wavelengths in the visible *electromagnetic spectrum* [400 (blue) to 700 nanometers (red); a nanometer (nm) is 10^{-9} meter]. The sensation of color arises from the sensitivities of three types of neurochemical sensors in the retina to the visible spectrum. The relative response of these sensors is shown in Fig. 2.6. Note that each sensor responds to a range of wavelengths. The illumination source has its own spectral composition $f(\lambda)$ which is modified by the reflecting surface. Let $r(\lambda)$ be this reflectance function. Then the measurement R produced by the "red" sensor is given by

$$R = \int f(\lambda)\, r(\lambda)\, h_R(\lambda)\; d\lambda \tag{2.32}$$

So the sensor output is actually the integral of three different wavelength-dependent components: the source f, the surface reflectance r, and the sensor h_R.

Surprisingly, only weighted combinations of three delta-function approximations to the different $f(\lambda)h(\lambda)$, that is, $\delta(\lambda_R)$, $\delta(\lambda_G)$, and $\delta(\lambda_B)$, are necessary to

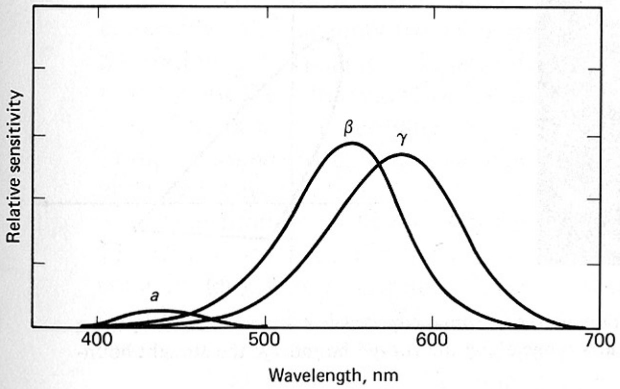


Fig. 2.6 Spectral response of human color sensors.

produce the sensation of nearly all the colors. This result is displayed on a *chromaticity diagram*. Such a diagram is obtained by first normalizing the three sensor measurements:

$$r = \frac{R}{R + G + B}$$
$$g = \frac{G}{R + G + B} \tag{2.33}$$
$$b = \frac{B}{R + G + B}$$

and then plotting perceived color as a function of any two (usually red and green). Chromaticity explicitly ignores intensity or brightness; it is a section through the three-dimensional color space (Fig. 2.7). The choice of $(\lambda_R, \lambda_G, \lambda_B) = (410, 530, 650)$ *nm* maximizes the realizable colors, but some colors still cannot be realized since they would require negative values for some of r, g, and b.

Another more intuitive way of visualizing the possible colors from the *RGB* space is to view these measurements as Euclidean coordinates. Here any color can be visualized as a point in the unit cube. Other coordinate systems are useful for different applications; computer graphics has proved a strong stimulus for investigation of different color space bases.

Color Perception

Color perception is complex, but the essential step is a transformation of three input intensity measurements into another basis. The coordinates of the new

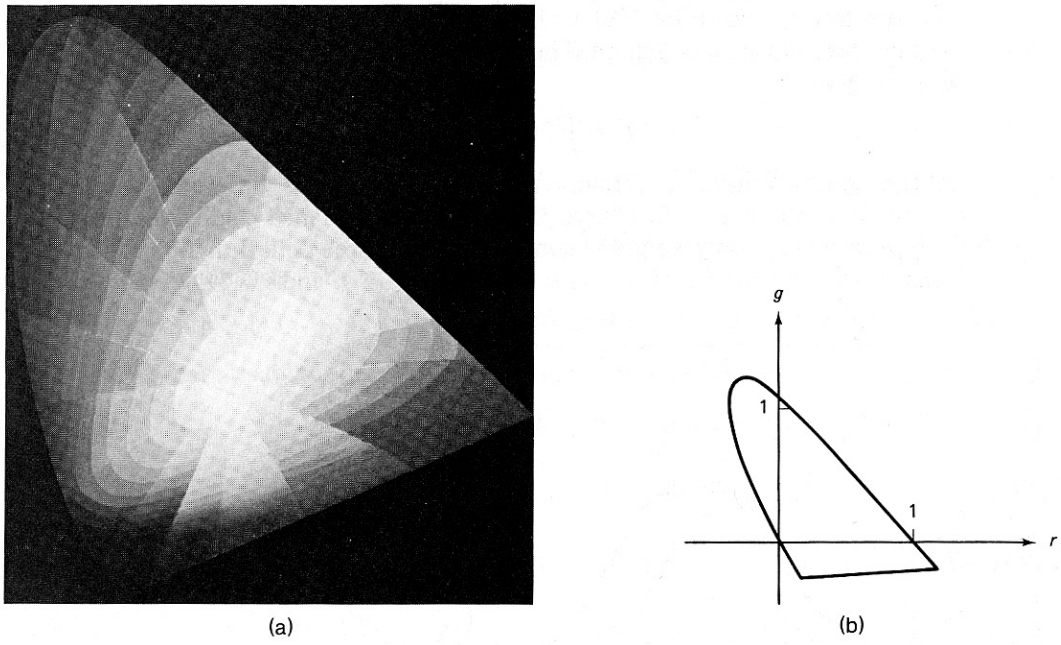


Fig. 2.7 (a) An artist's conception of the chromaticity diagram—*see color insert*; (b) a more useful depiction. Spectral colors range along the curved boundary; the straight boundary is the line of purples.

basis are more directly related to human color judgments.

Although the *RGB* basis is good for the acquisition or display of color information, it is not a particularly good basis to explain the perception of colors. Human vision systems can make good judgments about the relative surface reflectance $r(\lambda)$ despite different illuminating wavelengths; this reflectance seems to be what we mean by surface color.

Another important feature of the color basis is revealed by an ability to perceive in "black and white," effectively deriving intensity information from the color measurements. From an evolutionary point of view, we might expect that color perception in animals would be compatible with preexisting noncolor perceptual mechanisms.

These two needs—the need to make good color judgments and the need to retain and use intensity information—imply that we use a transformed, non-*RGB* basis for color space. Of the different bases in use for color vision, all are variations on this theme: Intensity forms one dimension and color is a two-dimensional subspace. The differences arise in how the color subspace is described. We categorize such bases into two groups.

1. *Intensity/Saturation/Hue (IHS).* In this basis, we compute intensity as

$$\text{intensity:} = R + G + B \tag{2.34}$$

The saturation measures the lack of whiteness in the color. Colors such as "fire engine" red and "grass" green are saturated; pastels (e.g., pinks and pale blues) are desaturated. Saturation can be computed from *RGB* coordinates by the formula [Tenenbaum and Weyl 1975]

$$\text{saturation:} = 1 - \frac{3 \min (R, G, B)}{\text{intensity}} \tag{2.35}$$

Hue is roughly proportional to the average wavelength of the color. It can be defined using *RGB* by the following program fragment:

$$\text{hue:} = \cos^{-1}\left\{\frac{\{\tfrac{1}{2}[(R-G)+(R-B)]\}}{\sqrt{(R-G)^2+(R-B)(G-B)}^{1/2}}\right\} \tag{2.36}$$

$$\text{If } B > G \text{ then hue:} = 2pi - \text{hue}$$

The IHS basis transforms the *RGB* basis in the following way. Thinking of the color cube, the diagonal from the origin to (1, 1, 1) becomes the intensity axis. Saturation is the distance of a point from that axis and hue is the angle with regard to the point about that axis from some reference (Fig. 2.8).

This basis is essentially that used by artists [Munsell 1939], who term saturation *chroma*. Also, this basis has been used in graphics [Smith 1978; Joblove and Greenberg 1978].

One problem with the IHS basis, particularly as defined by (2.34) through (2.36), is that it contains essential singularities where it is impossible to define the color in a consistent manner [Kender 1976]. For example, hue has an essential singularity for all values of (R, G, B), where $R = G = B$. This means that special care must be taken in algorithms that use hue.

2. *Opponent processes.* The opponent process basis uses Cartesian rather than

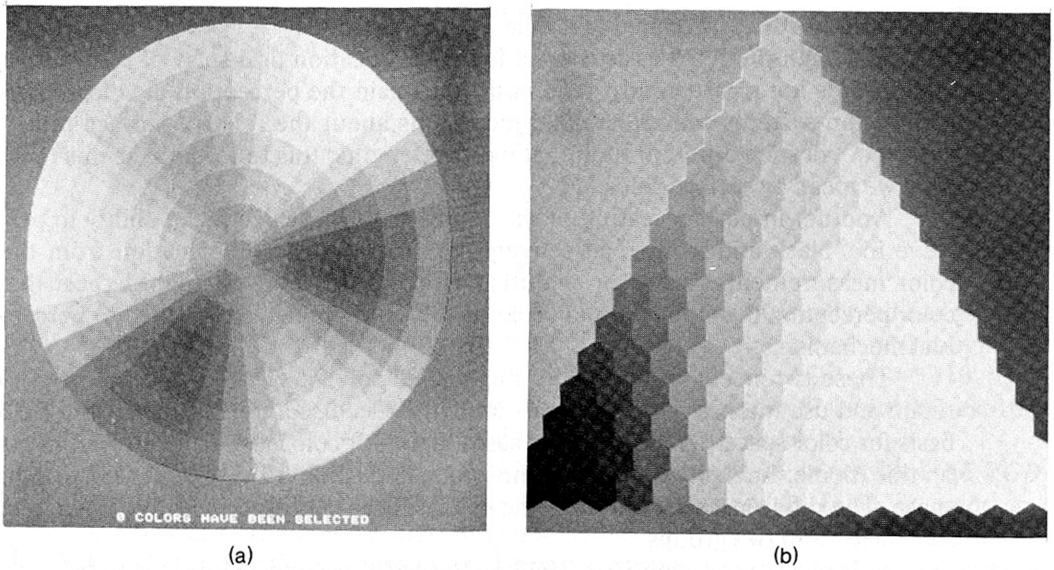


Fig. 2.8 An IHS Color Space. (a) Cross section at one intensity; (b) cross section at one hue—*see color inserts.*

cylindrical coordinates for the color subspace, and was first proposed by Hering [Teevan and Birney 1961]. The simplest form of basis is a linear transformation from R, G, B coordinates. The new coordinates are termed "$R - G$", "$Bl - Y$", and "$W - Bk$":

$$\begin{bmatrix} R - G \\ Bl - Y \\ W - Bk \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 \\ -1 & -1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The advocates of this representation, such as [Hurvich and Jameson 1957], theorize that this basis has neurological correlates and is in fact the way human beings represent ("name") colors. For example, in this basis it makes sense to talk about a "reddish blue" but not a "reddish green." Practical opponent process models usually have more complex weights in the transform matrix to account for psychophysical data. Some startling experiments [Land 1977] show our ability to make correct color judgments even when the illumination consists of only two principal wavelengths. The opponent process, at the level at which we have developed it, does not demonstrate how such judgments are made, but does show how stimulus at only two wavelengths will project into the color subspace. Readers interested in the details of the theory should consult the references.

Commercial television transmission needs an intensity, or "$W - Bk$" component for black-and-white television sets while still spanning the color space. The National Television Systems Committee (NTSC) uses a "YIQ" basis extracted from RGB via

$$\begin{bmatrix} I \\ Q \\ Y \end{bmatrix} = \begin{bmatrix} 0.60 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \\ 0.30 & 0.59 & 0.11 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This basis is a weighted form of

$$(I,\ Q,\ Y) = (\text{“}R\text{–cyan,” “magenta–green,” “}W\text{–}Bk\text{”})$$

2.2.6 Digital Images

The *digital images* with which computer vision deals are represented by m-vector discrete-valued image functions $\boldsymbol{f}(\mathbf{x})$, usually of one, two, three, or four dimensions.

Usually $m = 1$, and both the domain and range of $f(\mathbf{x})$ are discrete. The domain of f is finite, usually a rectangle, and the range of f is positive and bounded: $0 \leqslant f(\mathbf{x}) \leqslant M$ for some integer M. For all practical purposes, the image is a continuous function which is represented by measurements or *samples* at regularly spaced intervals. At the time the image is sampled, the intensity is usually *quantized* into a number of different *gray levels.* For a discrete image, $f(\mathbf{x})$ is an integer gray level, and $\mathbf{x} = (x, y)$ is a pair of *integer* coordinates representing a sample point in a two-dimensional image plane. Sampling involves two important choices: (1) the *sampling interval*, which determines in a basic way whether all the information in the image is represented, and (2) the *tesselation* or spatial pattern of sample points, which affects important notions of connectivity and distance. In our presentation, we first show qualitatively the effects of sampling and gray-level quantization. Second, we discuss the simplest kinds of tesselations of the plane. Finally, and most important, we describe the sampling theorem, which specifies how close the image samples must be to represent the image unambiguously.

The choice of integers to represent the gray levels and coordinates is dictated by limitations in sensing. Also, of course, there are hardware limitations in representing images arising from their sheer size. Table 2.3 shows the storage required for an image in 8-bit bytes as a function of m, the number of bits per sample, and N, the linear dimension of a square image.

For reasons of economy (and others discussed in Chapter 3) we often use images of considerably less spatial resolution than that required to preserve fidelity to the human viewer. Figure 2.9 provides a qualitative idea of image degradation with decreasing spatial resolution.

As shown in Table 2.3, another way to save space besides using less spatial resolution is to use fewer bits per gray level sample. Figure 2.10 shows an image represented with different numbers of bits per sample. One striking effect is the "contouring" introduced with small numbers of gray levels. This is, in general, a problem for computer vision algorithms, which cannot easily discount the false contours. The choice of spatial and gray-level resolution for any particular computer vision task is an important one which depends on many factors. It is typical in

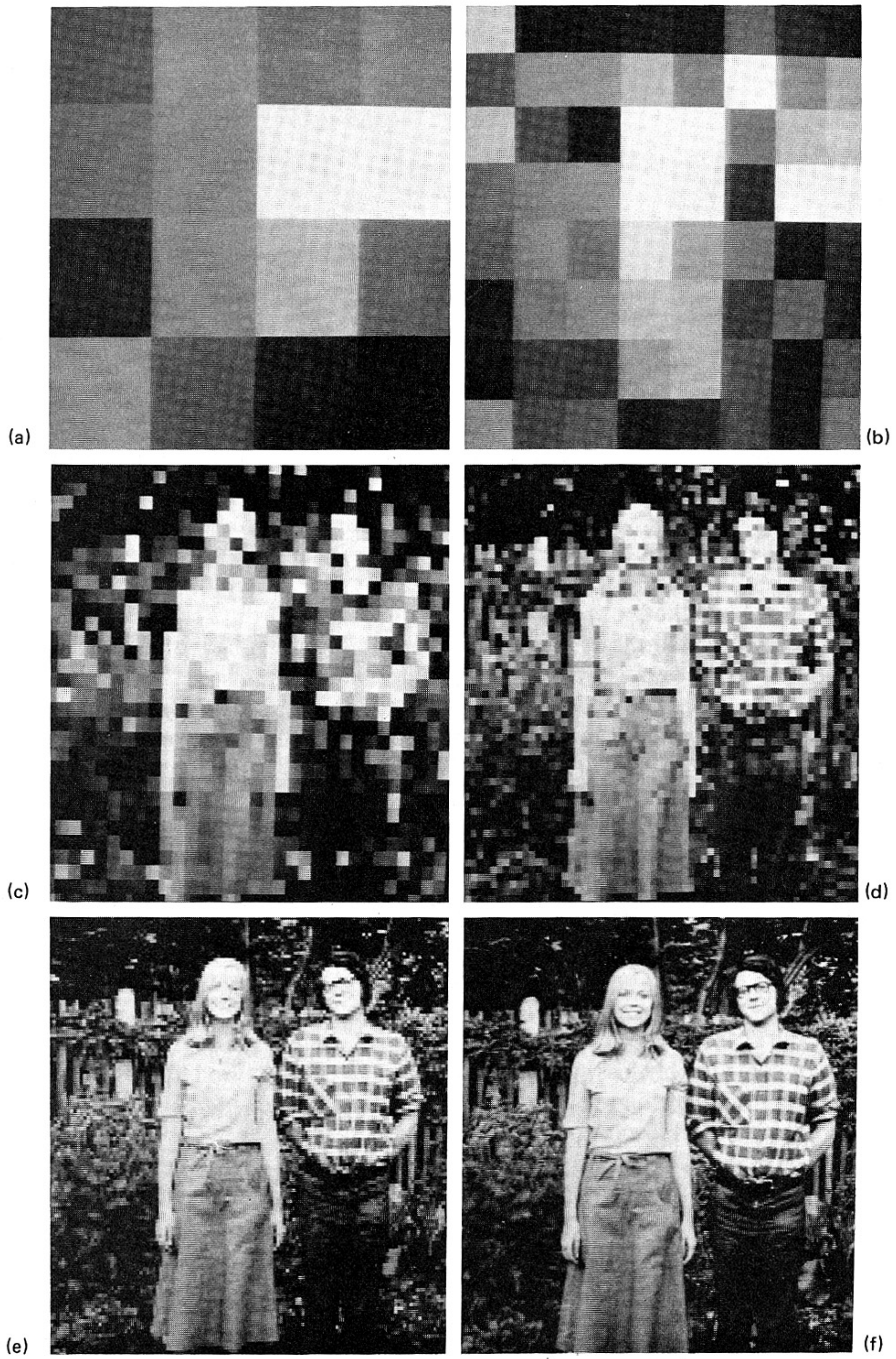


Fig. 2.9 Using different numbers of samples. (a) $N = 16$; (b) $N = 32$; (c) $N = 64$; (d) $N = 128$; (e) $N = 256$; (f) $N = 512$.

Table 2.3

NUMBER OF 8-BIT BYTES OF STORAGE FOR VARIOUS VALUES OF N AND M

N / m	32	64	128	256	512
1	128	512	2,048	8,192	32,768
2	256	1,024	4,096	16,384	65,536
3	512	2,048	8,192	32,768	131,072
4	512	2,048	8,192	32,768	131,072
5	1,024	4,096	16,384	65,536	262,144
6	1,024	4,096	16,384	65,536	262,144
7	1,024	4,096	16,384	65,536	262,144
8	1,024	4,096	16,384	65,536	262,144

computer vision to have to balance the desire for increased resolution (both gray scale and spatial) against its cost. Better data can often make algorithms easier to write, but a small amount of data can make processing more efficient. Of course, the image domain, choice of algorithms, and image characteristics all heavily influence the choice of resolutions.

Tesselations and Distance Metrics

Although the spatial samples for $f(\mathbf{x})$ can be represented as points, it is more satisfying to the intuition and a closer approximation to the acquisition process to think of these samples as finite-sized cells of constant gray-level partitioning the image. These cells are termed *pixels*, an acronym for *picture elements*. The pattern into which the plane is divided is called its *tesselation*. The most common regular tesselations of the plane are shown in Fig. 2.11.

Although rectangular tesselations are almost universally used in computer vision, they have a structural problem known as the "connectivity paradox." Given a pixel in a rectangular tesselation, how should we define the pixels to which it is connected? Two common ways are *four-connectivity* and *eight-connectivity*, shown in Fig. 2.12.

However, each of these schemes has complications. Consider Fig. 2.12c, consisting of a black object with a hole on a white background. If we use four-connectedness, the figure consists of four disconnected pieces, yet the hole is separated from the "outside" background. Alternatively, if we use eight-connectedness, the figure is one connected piece, yet the hole is now connected to the outside. This paradox poses complications for many geometric algorithms. Triangular and hexagonal tesselations do not suffer from connectivity difficulties (if we use three-connectedness for triangles); however, *distance* can be more difficult to compute on these arrays than for rectangular arrays.

The distance between two pixels in an image is an important measure that is fundamental to many algorithms. In general, a distance d is a *metric*. That is,

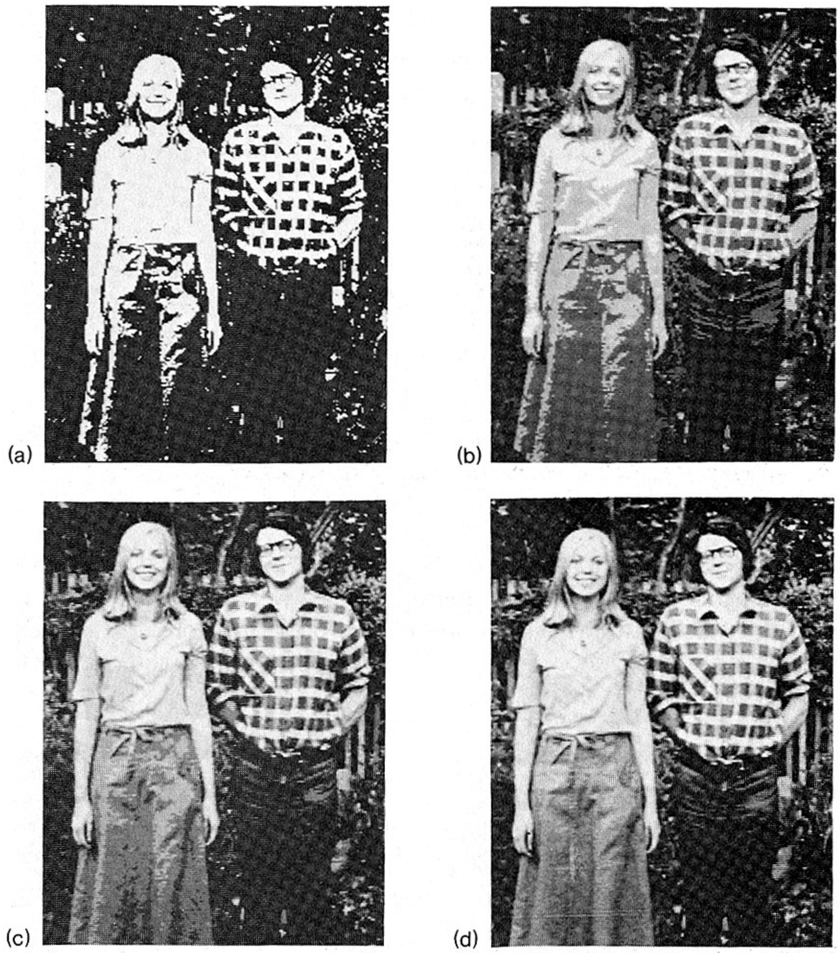


Fig. 2.10 Using different numbers of bits per sample. (a) $m = 1$; (b) $m = 2$; (c) $m = 4$; (d) $m = 8$.

(1) $d(\mathbf{x}, \mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$

(2) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$

(3) $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geqslant d(\mathbf{x}, \mathbf{z})$

For square arrays with unit spacing between pixels, we can use any of the following common distance metrics (Fig. 2.13) for two pixels $\boldsymbol{x} = (x_1,y_1)$ and $\boldsymbol{y} = (x_2,y_2)$.

Euclidean:

$$d_e(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.37}$$

City block:

$$d_{cb}(\mathbf{x}, \mathbf{y}) = |x_1 - x_2| + |y_1 - y_2| \tag{2.38}$$

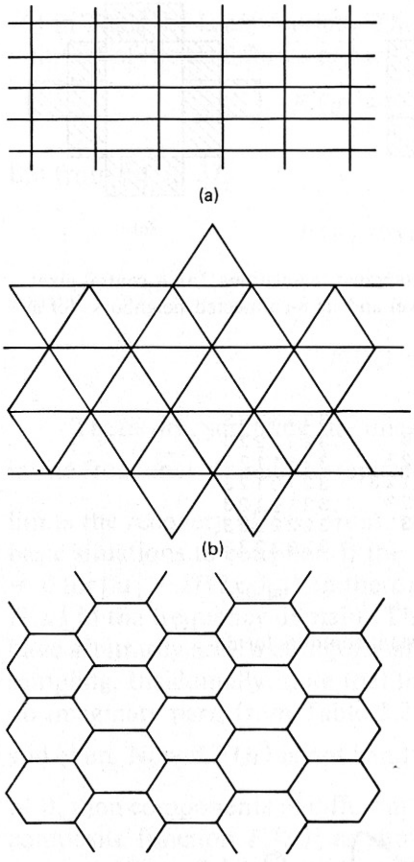


Fig. 2.11 Different tesselations of the image plane. (a) Rectangular; (b) triangular; (c) hexagonal.

Chessboard:

$$d_{ch}(\mathbf{x}, \mathbf{y}) = \max\left\{|x_1 - x_2|, |y_1 - y_2|\right\} \tag{2.39}$$

Other definitions are possible, and all such measures extend to multiple dimensions. The tesselation of higher-dimensional space into pixels usually is confined to (n-dimensional) cubical pixels.

The Sampling Theorem

Consider the one-dimensional "image" shown in Fig. 2.14. To digitize this image one must sample the image function. These samples will usually be separated at regular intervals as shown. How far apart should these samples be to allow reconstruction (to a given accuracy) of the underlying continuous image from its samples? This question is answered by the Shannon sampling theorem. An excellent rigorous presentation of the sampling theorem may be found in [Rosenfeld and Kak 1976]. Here we shall present a shorter graphical interpretation using the results of Table 2.2. For simplicity we consider the image to be periodic in order to avoid small edge effects introduced by the finite image domain. A more rigorous

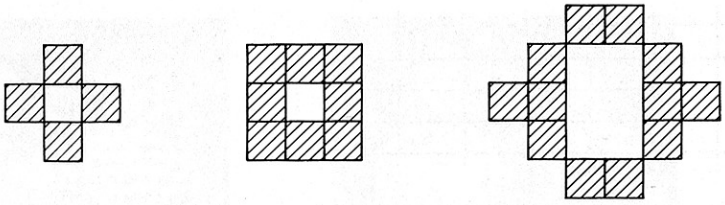


Fig. 2.12 Connectivity paradox for rectangular tesselations. (a) A central pixel and its 4-connected neighbors; (b) a pixel and its 8-connected neighbors; (c) a figure with ambiguous connectivity.

```
    2 3 2           3          3 3 3 3 3 3 3
  3 2 2 2 3       3 2 3        3 2 2 2 2 2 3
2 2 1 1 1 2 2   3 2 1 2 3      3 2 1 1 1 2 3
3 2 1 0 1 2 3 3 2 1 0 1 2 3    3 2 1 0 1 2 3
2 2 1 1 1 2 2   3 2 1 2 3      3 2 1 1 1 2 3
  3 2 2 2 3       3 2 3        3 2 2 2 2 2 3
    2 3 2           3          3 3 3 3 3 3 3

     (a)           (b)              (c)
```

Fig. 2.13 Equidistant contours for different metrics.

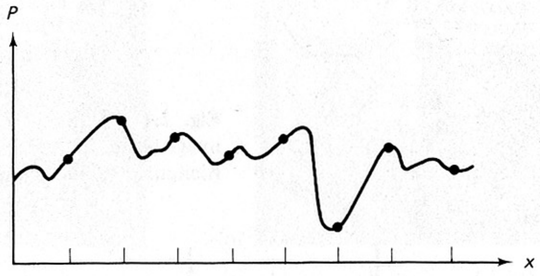


Fig. 2.14 One-dimensional image and its samples.

treatment, which considers these effects, is given in [Andrews and Hunt 1977].

Suppose that the image is sampled with a "comb" function of spacing x_0 (see Table 2.2). Then the sampled image can be modeled by

$$f_s(x) = f(x) \sum_n \delta(x - nx_0) \tag{2.40}$$

where the image function modulates the comb function. Equivalently, this can be written as

$$f_s(x) = \sum_n f(nx_0)\, \delta(x - nx_0) \tag{2.41}$$

The right-hand side of Eq. (2.40) is the product of two functions, so that property

(6) in Table 2.1 is appropriate. The Fourier transform of $f_s(x)$ is equal to the convolution of the transforms of each of the two functions. Using this result yields

$$F_s(u) = F(u) * \frac{1}{x_0} \sum_n \delta\left(u - \frac{n}{x_0}\right) \tag{2.42}$$

But from Eq. (2.3),

$$F(u) * \delta\left(u - \frac{n}{x_0}\right) = F\left(u - \frac{n}{x_0}\right) \tag{2.43}$$

so that

$$F_s(u) = \frac{1}{x_0} \sum_n F\left(u - \frac{n}{x_0}\right) \tag{2.44}$$

Therefore, sampling the image function $f(x)$ at intervals of x_0 is equivalent in the frequency domain to replicating the transform of f at intervals of $\frac{1}{x_0}$. This limits the recovery of $f(x)$ from its sampled representation, $f_s(x)$. There are two basic situations to consider. If the transform of $f(x)$ is *bandlimited* such that $F(u) = 0$ for $|u| > 1/(2x_0)$, then there is no overlap between successive replications of $F(u)$ in the frequency domain. This is shown for the case of Fig. 2.15a, where we have arbitrarily used a triangular-shaped image transform to illustrate the effects of sampling. Incidentally, note that for this transform $F(u) = F(-u)$ and that it has no imaginary part; from Table 2.2, the one-dimensional image must also be real and even. Now if $F(u)$ is not bandlimited, i.e., there are $u > \frac{1}{2x_0}$ for which $F(u) \neq 0$, then components of different replications of $F(u)$ will interact to produce the composite function $F_s(u)$, as shown in Fig. 2.15b. In the first case $f(x)$ can be recovered from $F_s(u)$ by multiplying $F_s(u)$ by a suitable $G(u)$:

$$G(u) = \begin{cases} 1 & |u| < \frac{1}{2x_0} \\ 0 & \text{otherwise} \end{cases} \tag{2.45}$$

Then

$$f(x) = \mathfrak{F}^{-1}[F_s(u)G(u)] \tag{2.46}$$

However, in the second case, $F_s(u)G(u)$ is very different from the original $F(u)$. This is shown in Fig. 2.15c. Sampling a $F(u)$ that is not bandlimited allows information at high spatial frequencies to interfere with that at low frequencies, a phenomenon known as *aliasing*.

Thus the sampling theorem has this very important result: As long as the image contains no spatial frequencies greater than one-half the sampling frequency, the underlying continuous image is unambiguously represented by its samples. However, lest one be tempted to insist on images that have been so sampled, note that it may be useful to sample at lower frequencies than would be required for total reconstruction. Such sampling is usually preceded by some form of blurring of

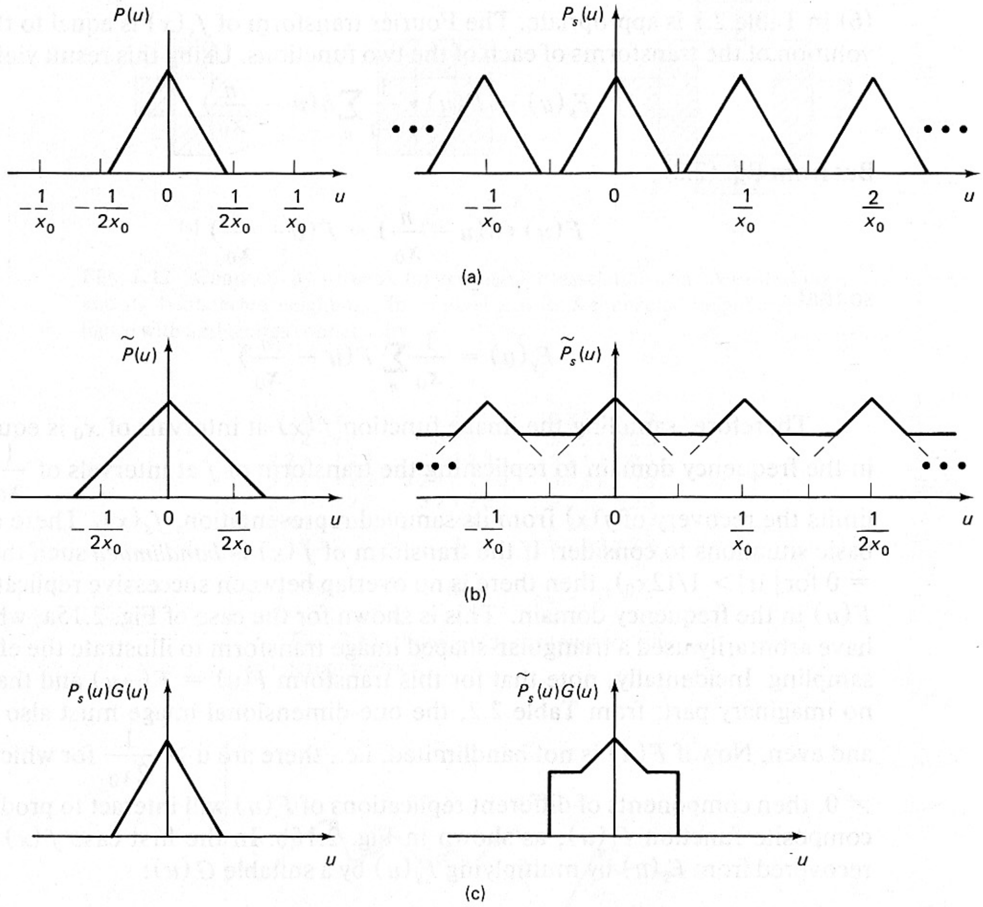


Fig. 2.15 (a) $F(u)$ bandlimited so that $F(u) = 0$ for $|u| > 1/2x_0$. (b) $F(u)$ not bandlimited as in (a). (c) reconstructed transform.

the image, or can be incorporated with such blurring (by integrating the image intensity over a finite area for each sample). Image blurring can bury irrelevant details, reduce certain forms of noise, and also reduce the effects of aliasing.

2.3 IMAGING DEVICES FOR COMPUTER VISION

There is a vast array of methods for obtaining a digital image in a computer. In this section we have in mind only "traditional" images produced by various forms of radiation impinging on a sensor after having been affected by physical objects.

Many sensors are best modeled as an *analog* device whose response must be *digitized* for computer representation. The types of imaging devices possible are limited only by the technical ingenuity of their developers; attempting a definitive

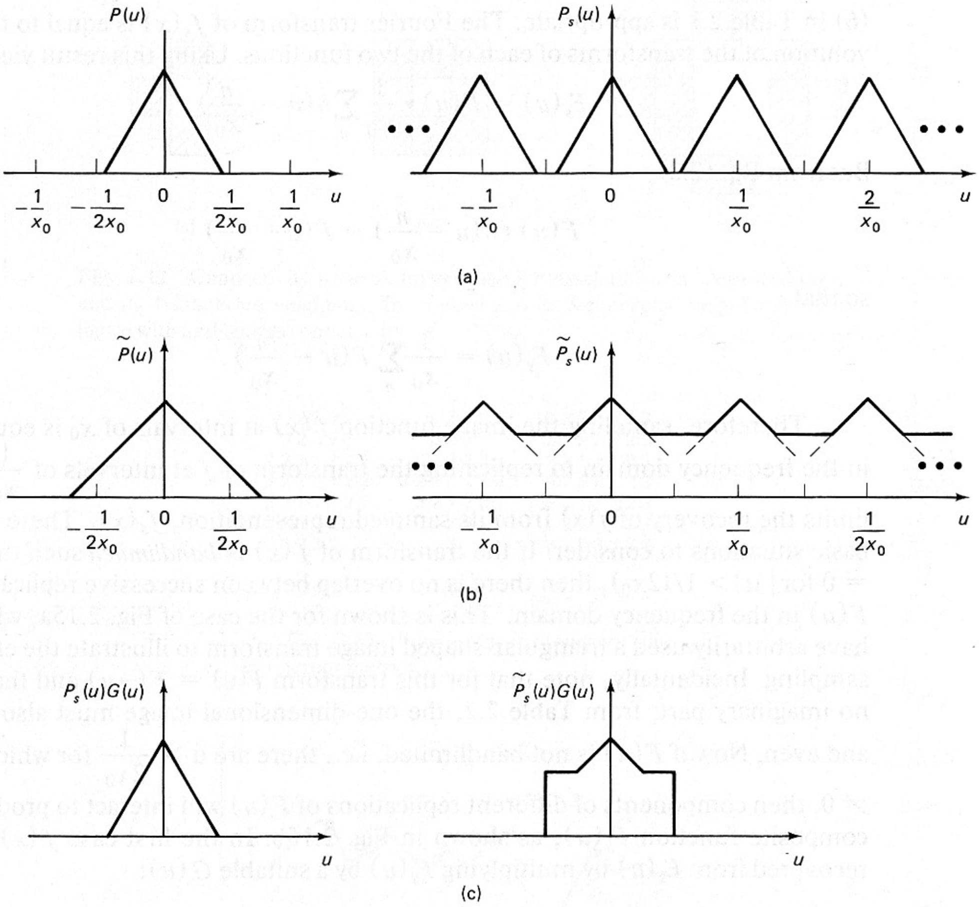


Fig. 2.15 (a) $F(u)$ bandlimited so that $F(u) = 0$ for $|u| > 1/2x_0$. (b) $F(u)$ not bandlimited as in (a). (c) reconstructed transform.

the image, or can be incorporated with such blurring (by integrating the image intensity over a finite area for each sample). Image blurring can bury irrelevant details, reduce certain forms of noise, and also reduce the effects of aliasing.

2.3 IMAGING DEVICES FOR COMPUTER VISION

There is a vast array of methods for obtaining a digital image in a computer. In this section we have in mind only "traditional" images produced by various forms of radiation impinging on a sensor after having been affected by physical objects.

Many sensors are best modeled as an *analog* device whose response must be *digitized* for computer representation. The types of imaging devices possible are limited only by the technical ingenuity of their developers; attempting a definitive

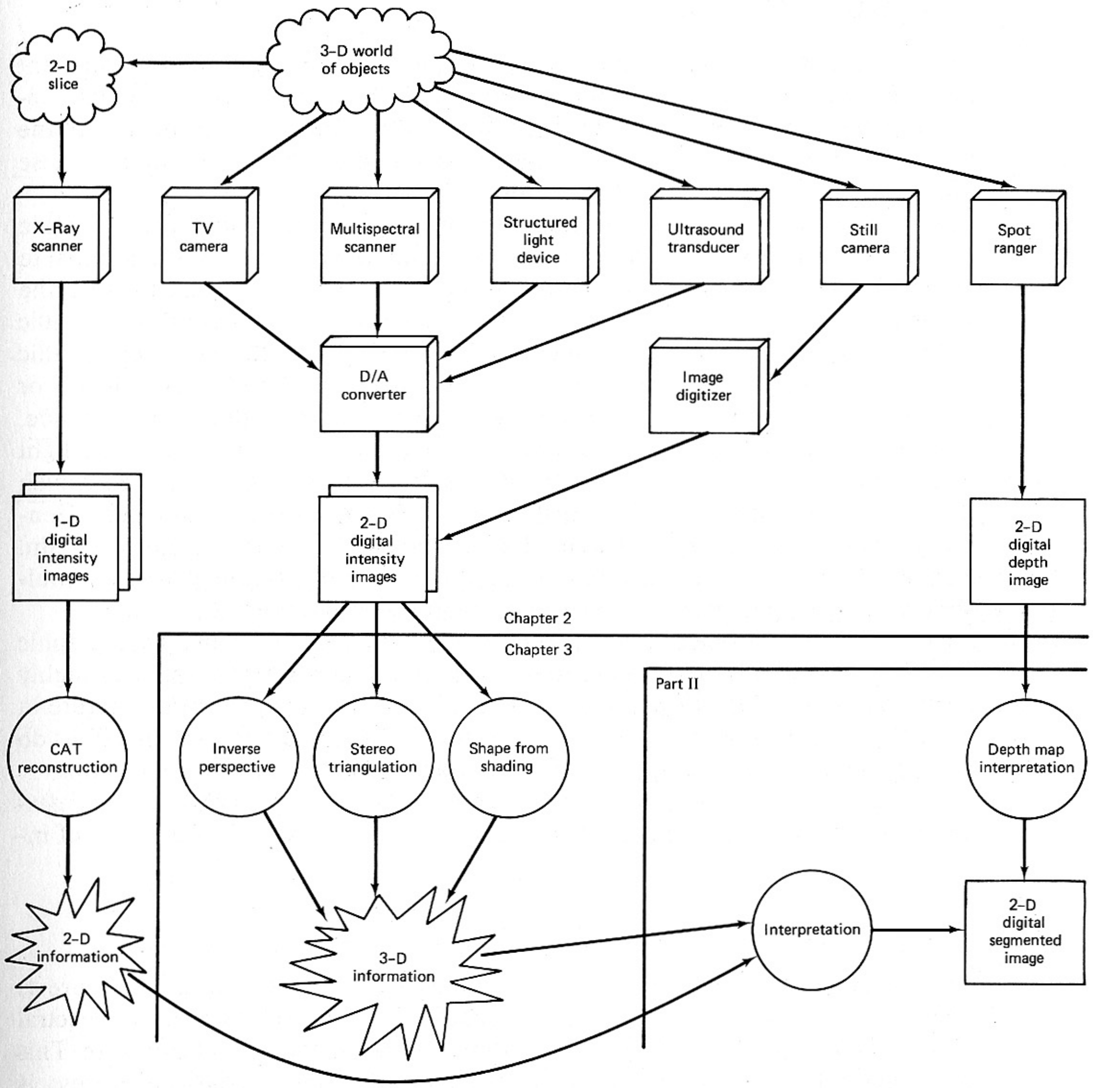


Fig. 2.16 Imaging devices (boxes), information structures (rectangles), and processes (circles).

taxonomy is probably unwise. Figure 2.16 is a flowchart of devices, information structures, and processes addressed in this and succeeding sections.

When the image already exists in some form, or physical considerations limit choice of imaging technology, the choice of digitizing technology may still be open. Most images are carried on a permanent medium, such as film, or at least are available in (essentially) analog form to a digitizing device. Generally, the relevant technical characteristics of imaging or digitizing devices should be foremost in mind when a technique is being selected. Such considerations as the signal-to-noise ratio of the device, its resolution, the speed at which it works, and its expense are important issues.

2.3.1 Photographic Imaging

The camera is the most familiar producer of optical images on a permanent medium. We shall not address here the multitudes of still- and movie-camera options; rather, we briefly treat the characteristics of the photographic film and of the digitizing devices that convert the image to machine-readable form. More on these topics is well presented in the References.

Photographic (black-and-white) film consists of an emulsion of silver halide crystals on a film base. (Several other layers are identifiable, but are not essential to an understanding of the relevant properties of film.) Upon exposure to light, the silver halide crystals form *development centers,* which are small grains of metallic silver. The photographic development process extends the formation of metallic silver to the entire silver halide crystal, which thus becomes a binary ("light" or "no light") detector. Subsequent processing removes undeveloped silver halide. The resulting film *negative* is dark where many crystals were developed and light where few were. The resolution of the film is determined by the *grain* size, which depends on the original halide crystals and on development techniques. Generally, the *faster* the film (the less light needed to expose it), the coarser the grain. Film exists that is sensitive to infrared radiation; x-ray film typically has two emulsion layers, giving it more gray-level range than that of normal film.

A repetition of the negative-forming process is used to obtain a photographic *print.* The negative is projected onto photographic paper, which responds roughly in the same way as the negative. Most photographic print paper cannot capture in one print the range of densities that can be present in a negative. Positive films do exist that do not require printing; the most common example is color slide film.

The response of film to light is not completely linear. The photographic *density* obtained by a negative is defined as the logarithm (base 10) of the ratio of incident light to transmitted light.

$$D = \log_{10}\left(\frac{I}{I_t}\right)$$

The *exposure* of a negative dictates (approximately) its response. Exposure is defined as the energy per unit area that exposed the film (in its sensitive spectral range). Thus exposure is the product of the *intensity* and the time of exposure. This mathematical model of the behavior of the photographic exposure process is correct for a wide operating range of the film, but *reciprocity failure* effects in the film keep one from being able always to trade light level for exposure time. At very low light levels, longer exposure times are needed than are predicted by the product rule.

The response of film to light is usually plotted in an "H&D curve" (named for Hurter and Driffield), which plots density versus exposure. The H&D curve of film displays many of its important characteristics. Figure 2.17 exhibits a typical H&D curve for a black and white film.

The *toe* of the curve is the lower region of low slope. It expresses reciprocity failure and the fact that the film has a certain bias, or *fog* response, which dominates its behavior at the lowest exposure levels. As one would expect, there is an upper limit to the density of the film, attained when a maximum number of silver

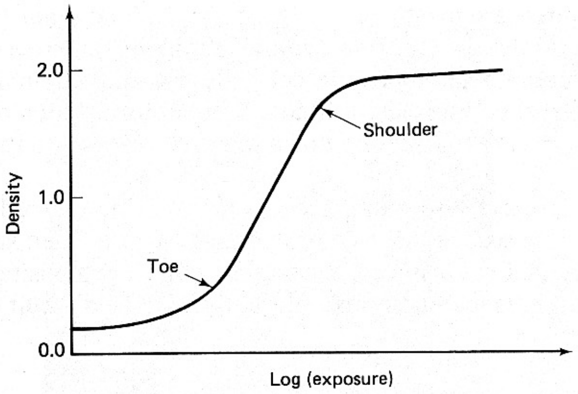


Fig. 2.17 Typical H & D curve.

halide crystals are rendered developable. Increasing exposure beyond this maximum level has little effect, accounting for the *shoulder* in the H&D curve, or its flattened upper end.

In between the toe and shoulder, there is typically a linear operating region of the curve. High-contrast films are those with high slope (traditionally called *gamma*); they respond dramatically to small changes in exposure. A high-contrast film may have a gamma between about 1.5 and 10. Films with gammas of approximately 10 are used in graphics arts to copy line drawings. General-purpose films have gammas of about 0.5 to 1.0.

The resolution of a general film is about 40 lines/mm, which means that a 1400 × 1400 image may be digitized from a 35mm slide. At any greater sampling frequency, the individual film grains will occupy more than a pixel, and the resolution will thus be grain-limited.

Image Digitizers (Scanners)

Accuracy and speed are the main considerations in converting an image on film into digital form. Accuracy has two aspects: spatial resolution, loosely the level of image spatial detail to which the digitizer can respond, and gray-level resolution, defined generally as the range of densities or reflectances to which the digitizer responds and how finely it divides the range. Speed is also important because usually many data are involved; images of 1 million samples are commonplace.

Digitizers broadly take two forms: mechanical and "flying spot." In a mechanical digitizer, the film and a sensing assembly are mechanically transported past one another while readings are made. In a flying-spot digitizer, the film and sensor are static. What moves is the "flying spot," which is a point of light on the face of a cathode-ray tube, or a laser beam directed by mirrors. In all digitizers a very narrow beam of light is directed through the film or onto the print at a known coordinate point. The light transmittance or reflectance is measured, transformed from analog to digital form, and made available to the computer through interfacing electronics. The location on the medium where density is being measured may also be transmitted with each reading, but it is usually determined by relative offset from positions transmitted less frequently. For example, a "new scan line" impulse is transmitted for TV output; the position along the current scan line yields an x position, and the number of scan lines yields a y position.

The mechanical scanners are mostly of two types, *flat-bed* and *drum.* In a flat-bed digitizer, the film is laid flat on a surface over which the light source and the sensor (usually a very accurate photoelectric cell) are transported in a raster fashion. In a drum digitizer, the film is fastened to a circular drum which revolves as the sensor and light source are transported down the drum parallel to its axis of rotation.

Color mechanical digitizers also exist; they work by using colored filters, effectively extracting in three scans three "color overlays" which when superimposed would yield the original color image. Extracting some "composite" color signal with one reading presents technical problems and would be difficult to do as accurately.

Satellite Imagery

LANDSAT and ERTS (Earth Resources Technology Satellites) have similar scanners which produce images of 2340 x 3380 7-bit pixels in four spectral bands, covering an area of 100 × 100 nautical miles. The scanner is mechanical, scanning six horizontal scan lines at a time; the rotation of the earth accounts for the advancement of the scan in the vertical direction.

A set of four images is shown in Fig. 2.18. The four spectral bands are numbered 4, 5, 6, and 7. Band 4 [0.5 to 0.6 μm (green)] accentuates sediment-laden water and shallow water, band 5 [0.6 to 0.7 μm (red)] emphasizes cultural features such as roads and cities, band 6 [0.7 to 0.8 μm (near infrared)] emphasizes vegetation and accentuates the contrast between land and water, band 7 [0.8 to 1.1 μm (near infrared)] is like band 6 except that it is better at penetrating atmospheric haze.

The LANDSAT images are available at nominal cost from the U.S. government (The EROS Data Center, Sioux Falls, South Dakota 57198). They are furnished on tape, and cover the entire surface of the earth (often the buyer has a choice of the amount of cloud cover). These images form a huge data base of multispectral imagery, useful for land-use and geological studies; they furnish something of an image analysis challenge, since one satellite can produce some 6 billion bits of image data per day.

Television Imaging

Television cameras are appealing devices for computer vision applications for several reasons. For one thing, the image is immediate; the camera can show events as they happen. For another, the image is already in electrical, if not digital form. "Television camera" is basically a nontechnical term, because many different technologies produce video signals conforming to the standards set by the FCC and NTSC. Cameras exist with a wide variety of technical specifications.

Usually, TV cameras have associated electronics which scan an entire "picture" at a time. This operation is closely related to broadcast and receiver standards, and is more oriented to human viewing than to computer vision. An entire image (of some 525 scan lines in the United States) is called a *frame*, and consists of two *fields*, each made up of alternate scan lines from the frame. These fields are generated and transmitted sequentially by the camera electronics. The transmitted image is thus *interlaced*, with all odd-numbered scan lines being "painted" on the

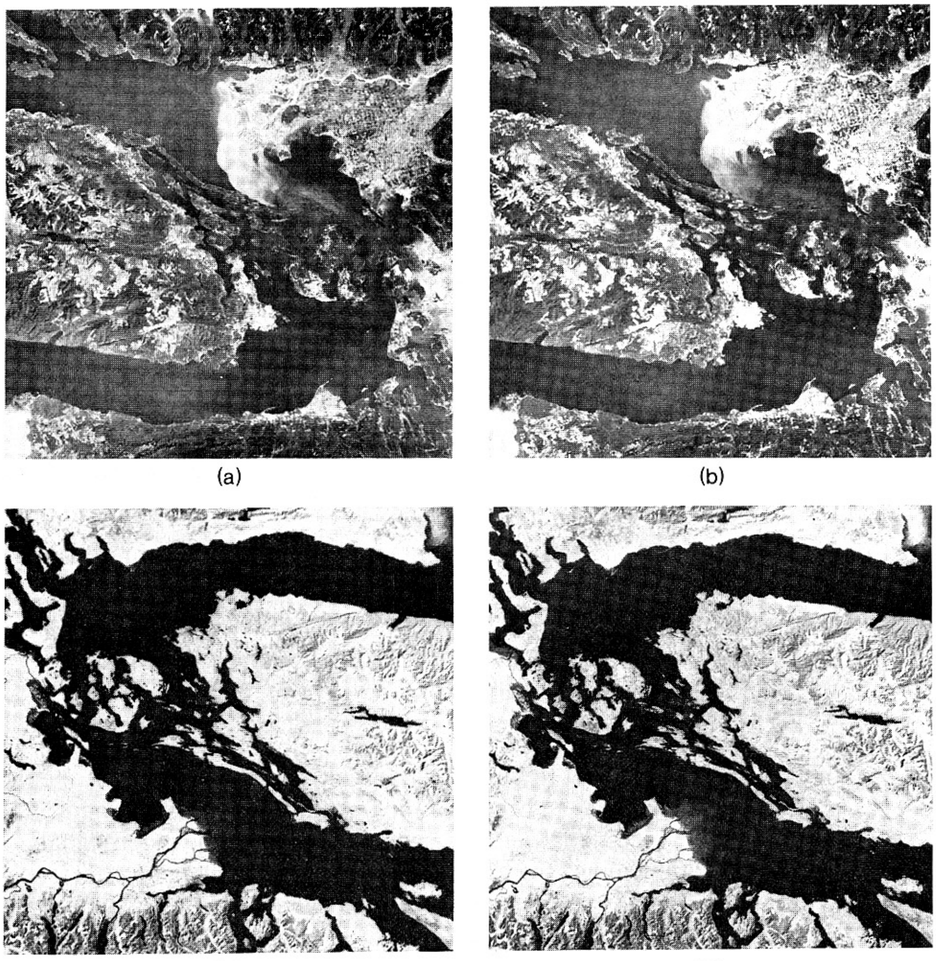


Fig. 2.18 The straits of Juan de Fuca as seen by the LANDSAT multispectral scanner. (a) Band 4; (b) band 5; (c) band 6; (d) band 7.

screen alternating with all even-numbered scan lines. In the United States, each field takes $\frac{1}{60}$ sec to scan, so a whole frame is scanned every $\frac{1}{30}$ sec. The interlacing is largely to prevent flickering of the image, which would become noticeable if the frame were painted from top to bottom only once in $\frac{1}{30}$ sec. These automatic scanning electronics may be replaced or overridden in many cameras, allowing "random access" to the image. In some technologies, such as the image dissector, the longer the signal is collected from any location, the better the signal-to-noise performance.

There are a number of different systems used to generate television images. We discuss five main methods below.

Image orthicon tube. This is one of the two main methods in use today (in addition to the vidicon). It offers very stable performance at all incident light levels and is widely used in commercial television. It is a storage-type tube, since it depends on the neutralization of positive charges by a scanning electron beam.

The image orthicon (Fig. 2.19) is divided into an imaging and readout section. In the imaging section, light from the scene is focused onto a semitransparent photocathode. This photocathode operates the same way as the cathode in a phototube. It emits electrons which are magnetically focused by a coil and are accelerated toward a positively charged target. The target is a thin glass disk with a fine-wire-mesh screen facing the photocathode. When electrons strike it, secondary emission from the glass takes place. As electrons are emitted from the photocathode side of the disk, positive charges build up on the scanning side. These charges correspond to the pattern of light intensity in the scene being viewed.

In the readout section, the back of the target is scanned by a low velocity electron beam from an electron gun at the rear of the tube. Electrons in this beam are absorbed by the target in varying amounts, depending on the charge on the target. The image is represented by the amplitude-modulated intensity of the returned beam.

Vidicon tube. The vidicon is smaller, lighter, and more rugged than the image orthicon, making it ideal for portable use. Here the target (the inner surface of the face plate) is coated with a transparent conducting film which forms a video signal electrode (Fig. 2.20). A thin photosensitive layer is deposited on the film, consisting of a large number of tiny resistive globules whose resistance decreases on illumination. This layer is scanned in raster fashion by a low velocity electron beam from the electron gun at the rear of the tube. The beam deposits electrons on the layer, thus reducing its surface potential. The two surfaces of the target essentially form a capacitor, and the scanning action of the beam produces a capacitive current at the video signal electrode which represents the video signal.

The plumbicon is essentially a vidicon with a lead oxide photosensitive layer. It offers the following advantages over the vidicon: higher sensitivity, lower dark current, and negligible persistence or lag.

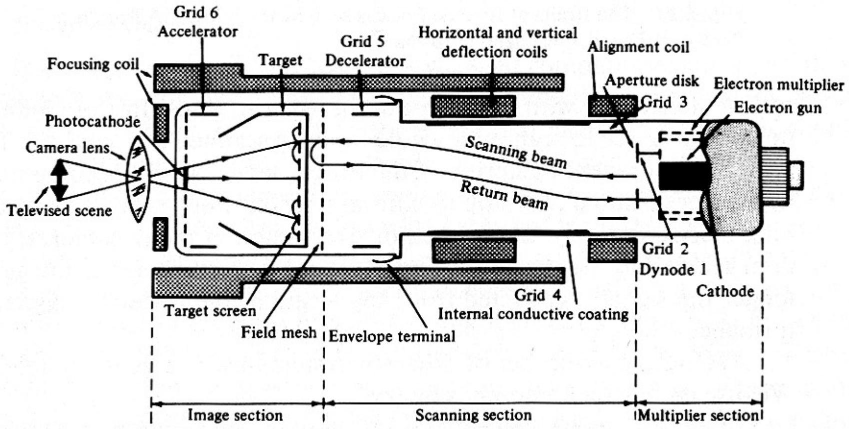
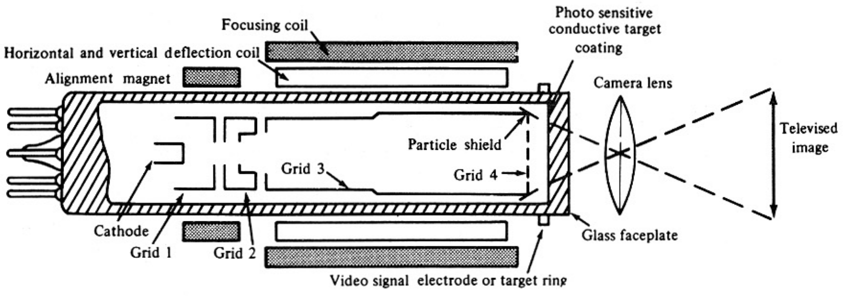


Fig. 2.19 The image orthicon.



Focusing coil
Horizontal and vertical deflection coil
Alignment magnet
Photo sensitive conductive target coating
Camera lens
Televised image
Particle shield
Grid 3
Grid 4
Cathode
Grid 1
Grid 2
Glass faceplate
Video signal electrode or target ring

Fig. 2.20 The vidicon.

Iconoscope tube. The iconoscope is now largely of historical interest. In it, an electron beam scans a target consisting of a thin mica sheet or mosaic coated with a photosensitive layer. In contrast to the vidicon and orthicon, the electron beam and the light both strike the same side of the target surface. The back of the mosaic is covered with a conductive film connected to an output load. The arrangement is equivalent to a matrix of small capacitors which discharge through a common lead.

Image dissector tube. The image dissector tube operates on instantaneous scanning rather than by neutralizing positive charges. Light from the scene is focused on a cathode coated with a photosensitive layer (Fig. 2.21). The cathode emits electrons in proportion to the amount of light striking it. These electrons are accelerated toward a target by the anode. The target is an electron multiplier covered by a small aperture which allows only a small part of the "electron image" emitted by the cathode to reach the target. The electron image is focused by a focusing coil that produces an axial magnetic field. The deflection coils then scan the electron image past the target aperture, where the electron multiplier produces a varying voltage representing the video signal. The image is thus "dissected" as it is scanned past the target, in an electronic version of a flat-bed digitizing process.

Charge transfer devices. A more recent development in image formation is that of solid-state image sensors, known as charge transfer devices (CTDs). There are two main classes of CTDs: charge-coupled devices (CCDs) and charge-injection devices (CIDs).

CCDs resemble MOSFETs (metal-oxide semiconductor field-effect transistor) in that they contain a "source" region and a "drain" region coupled by a depletion-region channel (Fig. 2.22). For imaging purposes, they can be considered as a monolithic array of closely spaced MOS capacitors forming a shift register (Fig. 2.23). Charges in the depletion region are transferred to the output by applying a series of clocking pulses to a row of electrodes between the source and the drain.

Photons incident on the semiconductor generate a series of charges on the CCD array. They are transferred to an output register either directly one line at a time (line transfer) or via a temporary storage area (frame transfer). The storage

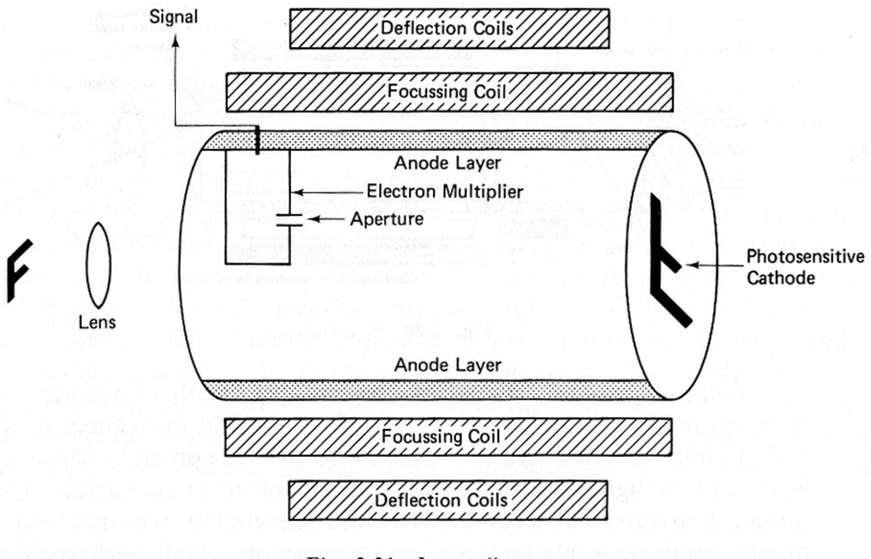


Fig. 2.21 Image dissector.

area is needed in frame transfer because the CCD array is scanned more rapidly than the output can be directly accommodated.

Charge injection devices (CIDs) resemble CCDs except that during sensing the charge is confined to the image site where it was generated (Fig. 2.24). The charges are read using an *X-Y* addressing technique similar to that used in computer memories. Basically, the stored charge is "injected" into the substrate and the resulting displacement current is detected to create the video signal.

CTD technology offers a number of advantages over conventional-tube-type cameras: light weight, small size, low power consumption, resistance to burn-in, low blooming, low dark current, high sensitivity, wide spectral and dynamic range, and lack of persistence. CIDs have the further advantages over CCDs of tolerance to processing defects, simple mechanization, avoidance of charge transfer losses, and minimized blooming. CTD cameras are now available commercially.

Analog-to-Digital Conversion

With current technology, the representation of an image as an analog electrical waveform is usually an unavoidable precursor to further processing. Thus the operation of deriving a digital representation of an analog voltage is basic to computer vision input devices.

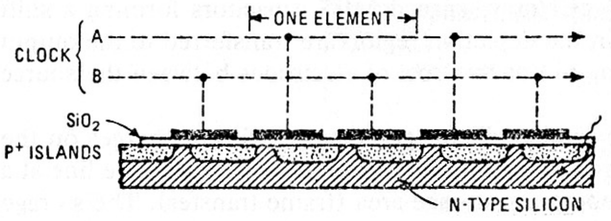


Fig. 2.22 Charge coupled device.

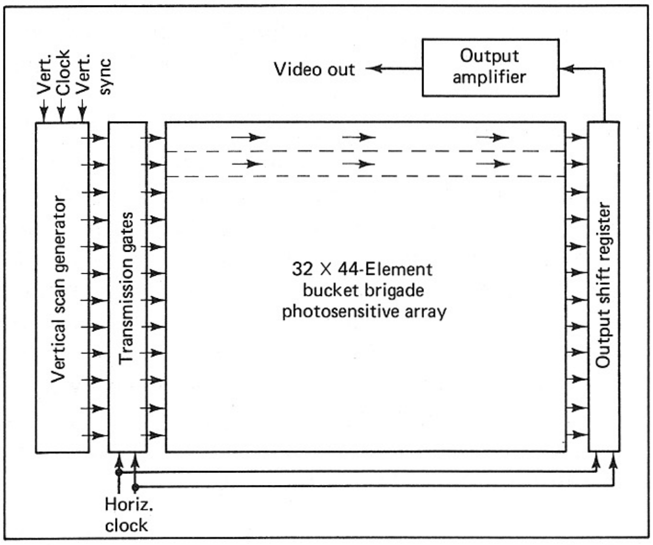


Fig. 2.23 A CCD array (line transfer).

The function of an analog-to-digital (A/D) converter is to take as input a voltage such as a video signal and to produce as output a representation of the voltage in digital memory, suitable for reading by an interface to a digital computer. The quality of an A/D converter is measured by its temporal resolution (the speed at which it can perform conversions) and the accuracy of its digital output. Analog-

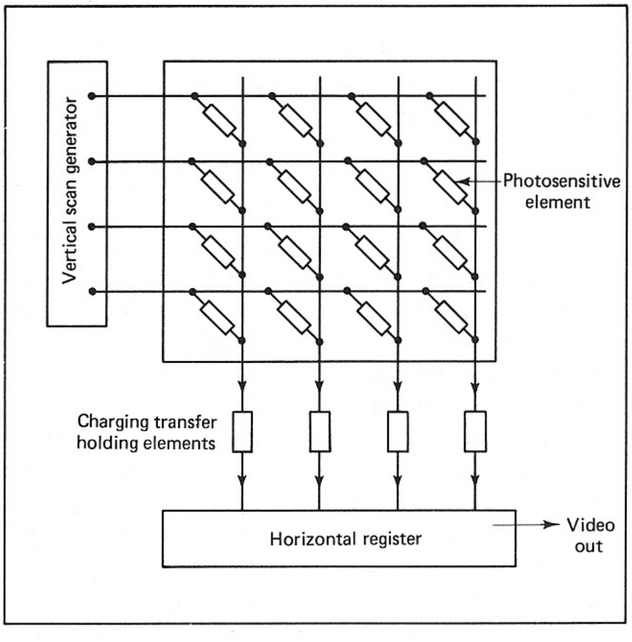


Fig. 2.24 A CID array.

to-digital converters are being produced as integrated circuit chips, but high-quality models are still expensive. The output precision is usually in the 8- to 12-bit range.

It is quite possible to digitize an entire frame of a TV camera (i.e., approximately 525 scan lines by 300 or so samples along a scan line) in a single frame time (1/30 sec in the United States). Several commercial systems can provide such fast digitization into a "frame buffer" memory, along with raster graphics display capabilities from the same frame buffer, and "video rate processing" of the digital data. The latter term refers to any of various low-level operations (such as averaging, convolution with small templates, image subtraction) which may be performed as fast as the images are acquired.

One inexpensive alternative to digitizing entire TV frames at once is to use an interface that acquires the TV signal for a particular point when the scan passes the requested location. With efficient programming, this point-by-point digitization can acquire an entire frame in a few seconds.

2.3.2 Sensing Range

The third dimension may be derived from binocular images by triangulation, as we saw earlier, or inferred from single monocular visual input by a variety of "depth cues," such as size and occlusion. Specialized technology exists to acquire "depth images" directly and reliably. Here we outline two such techniques: "light striping," which is based on triangulation, and "spot ranging," which is based on different principles.

Light Striping

Light striping is a particularly simple case of the use of *structured light* [Will and Pennington 1971]. The basic idea is to use geometric information in the illumination to help extract geometric information from the scene. The spatial frequencies and angles of bars of light falling on a scene may be clustered to find faces; randomly structured light may allow blank, featureless surfaces to be matched in stereo views; and so forth.

Many researchers [Popplestone et al. 1975; Agin 1972; Sugihara 1977] have used striping to derive three dimensions. In light striping, a single plane of light is projected onto a scene, which causes a stripe of light to appear on the scene (Fig. 2.25). Only the part of the scene illuminated by the plane is sensed by the vision system. This restricts the "image" to be an essentially one-dimensional entity, and simplifies matching corresponding points. The plane itself has a known position (equation in world coordinates), determinable by any number of methods involving either the measurement of the projecting device or the measurement of the final resulting plane of light. Every image point determines a single "line of sight" in three-space upon which the world point that produces the image point must lie. This line is determined by the focal point of the imaging system and the image point upon which the world point projects. In a light-striping system, any point that is sensed in the image is also guaranteed to lie on the light plane in three-space. But the light plane and the line of sight intersect in just one point (as long as

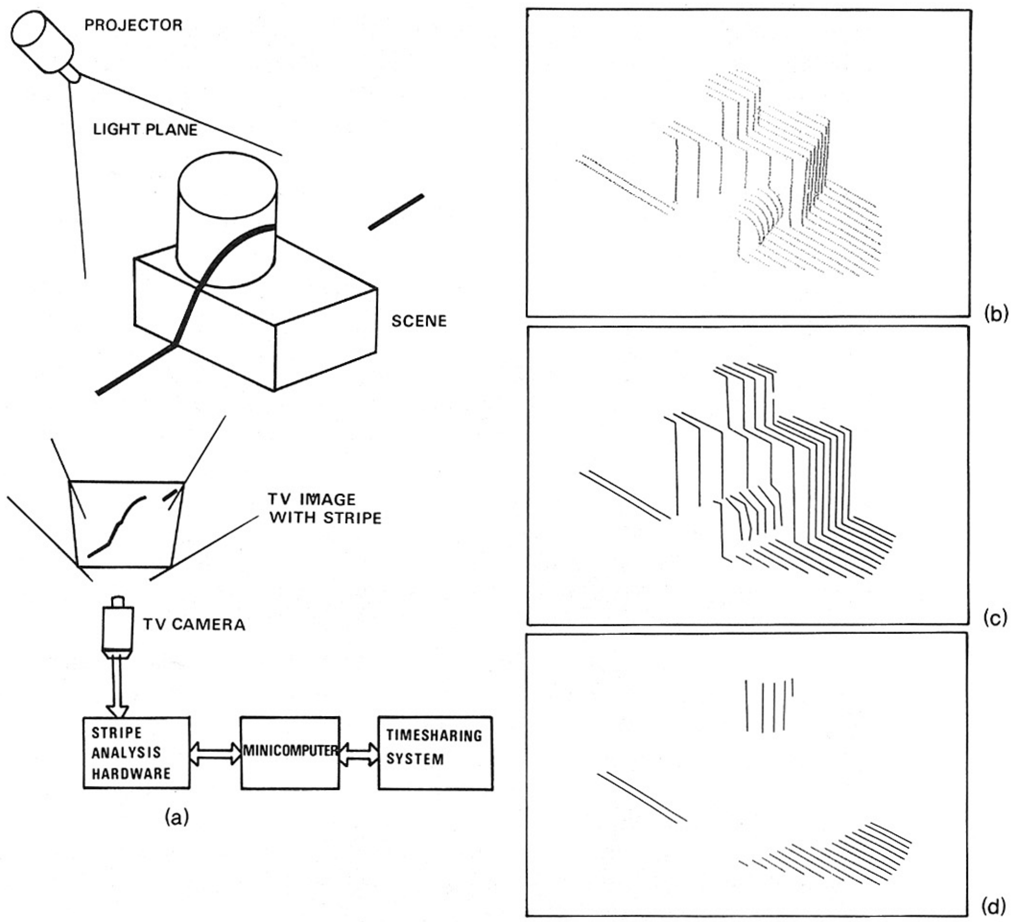


Fig. 2.25 Light striping. (a) A typical arrangement; (b) raw data; (c) data segmented into strips; (d) strips segmented into two surfaces.

the camera's focal point is not in the light plane). Thus by computation of the intersection of the line of sight with the plane of light, we derive the three-dimensional point that corresponds to any image point visible as part of a stripe.

The plane of light may result from a laser or from the projection of a slit. Only the light stripe should be visible to the imaging device; unless a laser is used, this implies a darkened room. If a camera is fitted with the proper filter, a laser-based system can be operated in normal light. Another advantage of the laser is that it can be focused into a narrower plane than can a slit image.

The only points whose three-dimensional coordinates can be computed are those that can be "seen" by both the light-stripe source and the camera at once. Since there must be a nonzero baseline if triangulation is to derive three-dimensional information, the camera cannot be too close to the projector, and thus concavities in the scene are potential trouble spots, since both the striper and the camera may not be able to "see" into them. Surfaces in the scene that are nearly parallel with the light plane will have a relatively small number of stripes projected onto them by any uniform stripe placement strategy. This problem is ameliorated by striping with two sets of parallel planes at right angles to each other [Agin 1972]. A major advantage of light striping over spot ranging is that (barring shadows) its continuity and discontinuity indicate similar conditions on the surface. It is easy to "segment" stripe images (Part II): Stripes falling on the same surface may easily be gathered together. This set of related stripes may be used in a number of ways to derive further information on the characteristics of the surface (Fig. 2.25b).

Spot Ranging

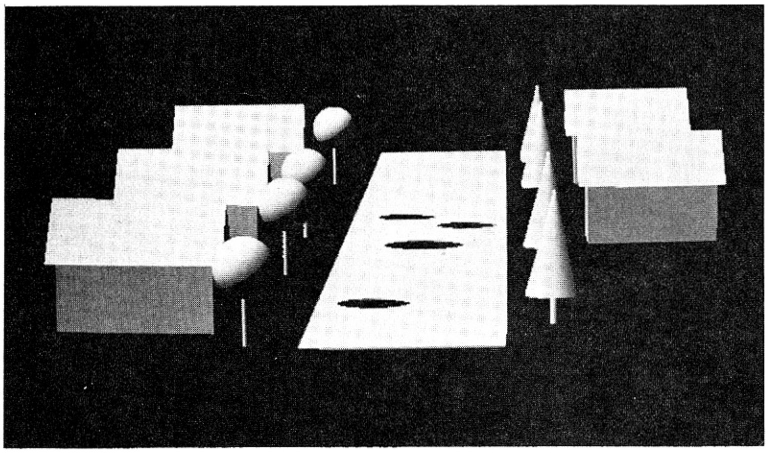
Civil engineers have used laser-based "spot range finders" for some time. In laboratory-size environments, they are a relatively new development. There are two basic techniques. First, one can emit a very sharp pulse and time its return ("lidar," the light equivalent of radar). This requires a sophisticated laser and electronics, since light moves 1 ft every billionth of a second, approximately. The second technique is to modulate the laser light in amplitude and upon its return compare the phase of the returning light with that of the modulator. The phase differences are related to the distance traveled [Nitzan et al. 1977]. A representative image is shown in Fig. 2.26.

Both these techniques produce results that are accurate to within about 1% of the range. Both of them allow the laser to be placed close to a camera, and thus "intensity maps" (images) and range maps may be produced from single viewpoints. The laser beam can easily poke into holes, and the return beam may be sensed close to the emitted one, so concavities do not present a serious problem. Since the laser beam is attenuated by absorption, it can yield intensity information as well. If the laser produces light of several wavelengths, it is possible to use filters and obtain multispectral reflectance information as well as depth information from the same device [Garvey 1976; Nitzan et al. 1977].

The usual mode of use of a spot ranging device is to produce a range map that corresponds to an intensity map. This has its advantages in that the correspondence may be close. The structural properties of light stripes are lost: It can be hard to "segment" the image into surfaces (to tell which "range pixels" are associated with the same surface). Range maps are amenable to the same sorts of segmentation techniques that are used for intensity images: Hough techniques, region growing, or differentiation-based methods of edge finding (Part II).

Ultrasonic Ranging

Just as light can be pulsed to determine range, so can sound and ultrasound (frequencies much higher than the audible range). Ultrasound has been used extensively in medicine to produce images of human organs (e.g., [Waag and Gramiak 1976]). The time between the transmitted and received signal determines range; the sound signal travels much slower than light, making the problem of timing the returning signal rather easier than it is in pulsed laser devices. However, the signal is severely attenuated as it travels through biological tissue, so that the detection apparatus must be very sensitive.


(a)

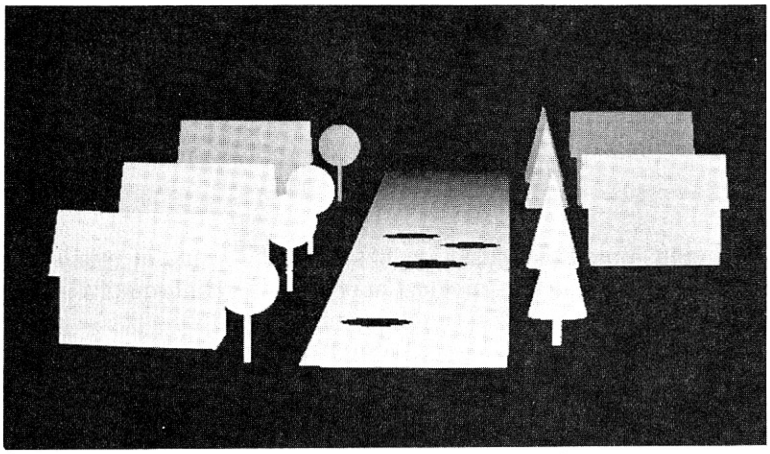

(b)

Fig. 2.26 Intensity and range images. (a) A (synthesized) intensity image of a street scene with potholes. The roofs all have the same intensity, which is different from the walls; (b) a corresponding range image. The wall and roof of each house have similar ranges, but the ranges differ from house to house.

One basic difference between sound and visible light ranging is that a light beam is usually reflected off just one surface, but that a sound beam is generally partially transmitted and partially reflected by "surfaces." The returning sound pulse has structure determined by the discontinuities in impedence to sound found in the medium through which it has passed. Roughly, a light beam returns information about a spot, whereas a sound beam can return information about the medium in the entire column of material. Thus, although sound itself travels relatively slowly, the data rate implicit in the returning structured sound pulse is quite high. Figure 2.27 shows an image made using the range data from ultrasound. The

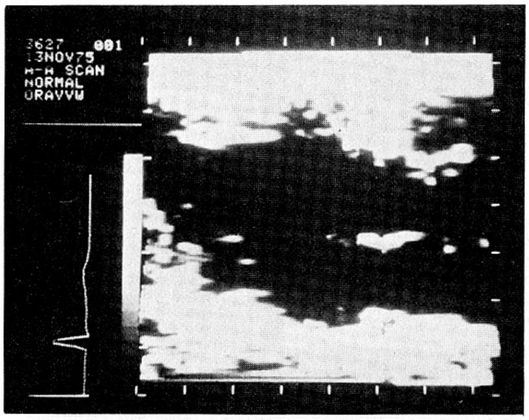


Fig. 2.27 Image made from ultrasound ranging.

sound pulses emanate from the top of the image and proceed toward the bottom, being partially reflected and transmitted along the way. In the figure, it is as if we were looking perpendicular to the beams, which are being displayed as brighter where strong reflectance is taking place. A single "scan line" of sound thus produces an image of an entire planar slice of medium.

2.3.3 Reconstruction Imaging

Two-dimensional reconstruction has been the focus of much research attention because of its important medical applications. High-quality images such as that shown in Fig. 1.2b can be formed by multiple images of x-ray projection data. This section contains the principles behind the most important reconstruction algorithms. These techniques are discussed in more detail with an expanded list of references in [Gordon and Herman 1974]. For a view of the many applications of two-dimensional reconstruction other than transmission scanning, the reader is referred to [Gordon et al. 1975].

Figure 2.28 shows the basic geometry to collect one-dimensional projections of two-dimensional data. (Most systems construct the image in a plane and repeat this technique for other planes; there are few true three-dimensional reconstruction systems that use planes of projection data simultaneously to construct volumes.)

In many applications sensors can measure the one-dimensional *projection* of two-dimensional image data. The projection $g(x')$ of an ideal image $f(x, y)$ in the direction θ is given by $\int f(x', y')\, dy'$ where $\mathbf{x}' = R_\theta \mathbf{x}$. If enough different projections are obtained, a good approximation to the image can be obtained with two-dimensional reconstruction techniques.

From Fig. 2.28, with the source at the first position along line AA', we can obtain the first projection datum from the detector at the first position along BB'. The line AB is termed a ray and the measurement at B a ray sum. Moving the source

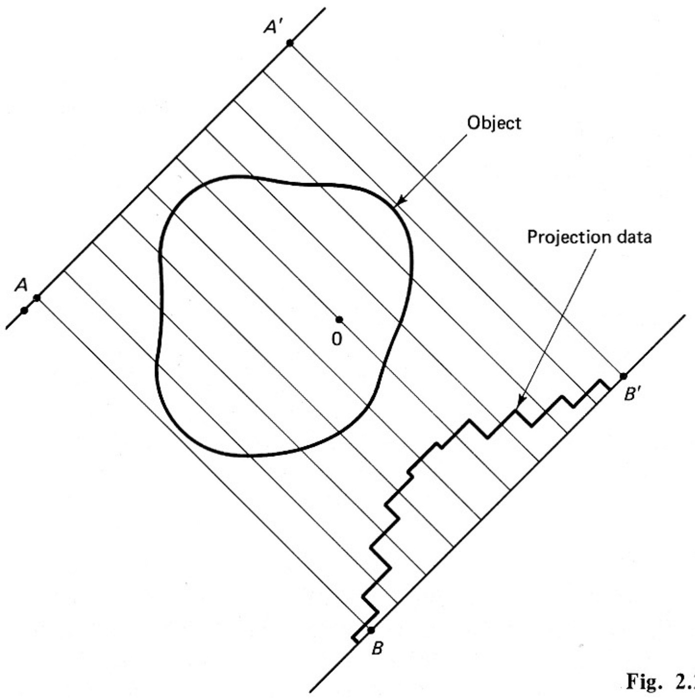


Fig. 2.28 Projection geometry.

and detector along lines AA' and BB' in synchrony allows us to obtain the entire data for projection 1. Now the lines AA' and BB' are rotated by a small angle $d\theta$ about 0 and the process is repeated. In the original x-ray systems $d\theta$ was 1° of angle, and 180 projections were taken. Each projection comprised 160 transmission measurements. The reconstruction problem is simply this: Given the projection data $g_k(x')$, $k = 0, \ldots, N-1$, construct the original image $f(\mathbf{x})$.

Systems in use today use a fan beam rather than the parallel rays shown. However, the mathematics is simpler for parallel rays and illustrates the fundamental ideas. We describe three related techniques: summation, Fourier interpolation, and convolution.

The Summation Method

The summation method is simple: Distribute every ray sum $g_k(x')$ over the image cells along the ray. Where there are N cells along a ray, each such cell is incremented by $\frac{1}{N}g(x')$. This step is termed *back projection*. Repeating this process for every ray results in an approximate version of the original [DeRosier 1971]. This technique is equivalent (within a scale factor) to blurring the image, or convolving it with a certain point-spread function. In the continuous case of infinitely many projections, this function is simply the radically symmetric $h(r) = 1/r$.

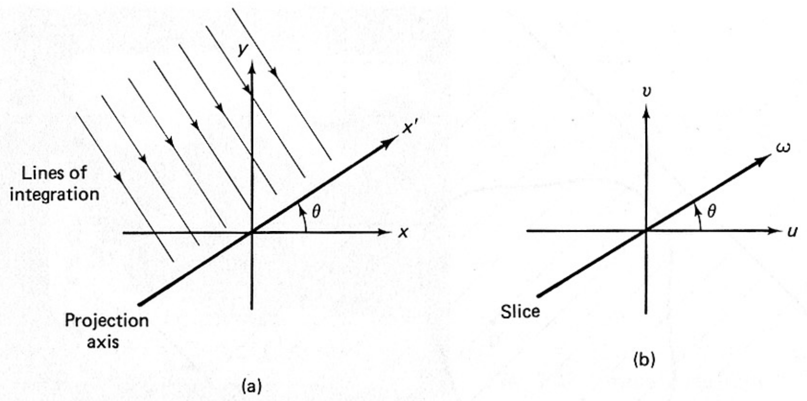


Fig. 2.29 Basis of Fourier techniques. (a) Projection axis x'; (b) corresponding axis in Fourier Space.

Fourier Algorithms

If a projection is Fourier-transformed, it defines a line through the origin in frequency space (Fig. 2.29). To show this formally, consider the expression for the two-dimensional transform

$$F(\mathbf{u}) = \int\int f(x, y) \exp[j2\pi(ux + vy)]\, dx\, dy \tag{2.47}$$

Now consider $y = 0$ (projection onto the x axis): $x' = x$ and

$$g_0(x') = \int f(x, y)\, dy \tag{2.48}$$

The Fourier transform of this equation is

$$\begin{aligned} \mathfrak{F}[g_0(x')] &= \int\int [f(x, y)\, dy] \exp j2\pi ux\; dx \\ &= \int\int f(x, y) \exp j2\pi ux\; dy\, dx \end{aligned} \tag{2.49}$$

which, by comparison with (2.47), is

$$\mathfrak{F}[g_0(x')] = F(u, 0) \tag{2.50}$$

Generalizing to any θ, the transform of an arbitrary $g(x')$ defines a line in the Fourier space representation of the cross section. Where $S_k(\omega)$ is the cross section of the Fourier transform along this line,

$$\begin{aligned} S_k(\omega) &= F(u\cos\theta,\ u\sin\theta) \\ &= \int g_k(x') \exp[-j2\pi u(x')]dx' \end{aligned} \tag{2.51}$$

Thus one way of reconstructing the original image is to use the Fourier transform of the projections to define points in the transform of $f(x)$, interpolate the undefined points of the transform from the known points, and finally take the inverse transform to obtain the reconstructed image.

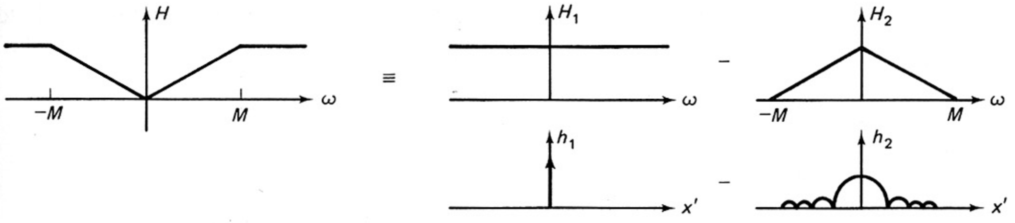


Fig. 2.30 Convolution method.

This technique can be applied with transforms other than the Fourier transform, and such methods are discussed in [DeRosier 1971; Crowther and Klug 1971].

The Convolution Method

The convolution method is the natural extension of the summation method. Since the summation method produces an image degraded from its convolution with some function h, one can remove the degradation by a "deconvolution." The straightforward way to accomplish this is to Fourier-transform the degraded image, multiply the result by an estimate of the transformed h^{-1}, and inverse-Fourier-transform the result. However, since all the operations are linear, a faster approach is to deconvolve the projections before performing the back projection. To show this formally, we use the inverse transform

$$f(\mathbf{x}) = \int\int F(u, v) \exp [j2\pi (ux + vy)] du\, dv \tag{2.52}$$

Changing to cylindrical coordinates (ω, θ) yields

$$f(\mathbf{x}) = \int\int F_\theta(\omega) \exp [j2\pi\omega (x \cos\theta + y \sin \theta)] |\omega| d\omega\, d\theta \tag{2.53}$$

Since $x' = x \cos\theta + y \sin\theta$, rewrite Eq. (2.53) as

$$f(\mathbf{x}) = \int \mathfrak{F}^{-1}\{F_\theta(\omega) H(\omega)\} d\theta \tag{2.54}$$

Since the image is bandlimited at some interval $(-\omega_m, \omega_m)$ one can define $H(\omega)$ arbitrarily outside of this interval. Therefore, $H(\omega)$ can be defined as a constant minus a triangular peak as shown in Fig. 2.30. Finally, the operation inside the integral in Eq. (2.54) is a convolution. Using the transforms shown in Fig. 2.30,

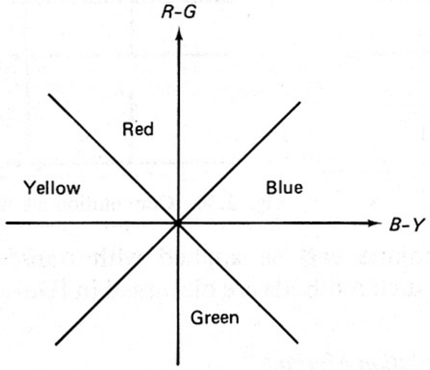
$$f(\mathbf{x}) = \int [f_\theta(x') - f_\theta(x')\omega_m \text{sinc}^2(\omega_m x')]\, d\theta \tag{2.55}$$

Owing to its speed and the fact that the deconvolutions can be performed while the data are being acquired, the convolution method is the method employed in the majority of systems.

EXERCISES

2.1 In a binocular animal vision system, assume a focal length f of an eye of 50 mm and a separation distance d of 5 cm. Make a plot of Δx vs. $-z$ using Eq. (2.9). If the resolution of each eye is on the order of 50 line pairs/mm, what is the useful range of the binocular system?

2.2 In an opponent-process color vision system, assume that the following relations hold:



For example, if the $(R-G, B-Y, W-Bk)$ components of the opponent-process system are $(0.5, 3, 4)$, the perceived color will be blue.

Work out the perceived colors for the following (R,G,B) measurements:

(a) $(0.2, 0.3, 0.4)$ (b) $(0.2, 0.3, 0)$ (c) $(7, 4, 1)$

2.3 Develop an indexing scheme for a hexagonal array and define a Euclidean distance measure between points in the array.

2.4 Assume that a one-dimensional image has the following form:

$$f(x) = \cos(2\pi u_o x)$$

and is sampled with $u_s = u_o$. Using the graphical method of Section 2.2.6, find an expression for $f(x)$ as given by Eq. (2.49). Is this expression equal to the original image? Explain.

2.5 A certain image has the following Fourier transform:

$$F(\mathbf{u}) = \begin{cases} \text{nonzero} & \text{inside a hexagonal domain} \\ 0 & \text{otherwise} \end{cases}$$

(a) What are the smallest values for u and v so that $F(\mathbf{u})$ can be reconstructed from $F_x(\mathbf{u})$?

(b) Suppose now that rectangular sampling is *not* used but that now the u and v directions subtend an angle of $\pi/3$. Does this change your answer as to the smallest u and v? Explain.

2.6 Extend the binocular imaging model of Fig. 2.3 to include convergence: Let the two imaging systems pivot in the $y = 0$ plane about the viewpoint. Let the system have a baseline of $2d$ and be converged at some angle θ such that a point (x, y, z) appears at the origin of each image plane.

(a) Solve for z in terms of r and θ.

(b) Solve for z in this situation for points with nonzero disparity.

2.7 Compute the convolution of two Rect functions, where

$$\text{Rect}(x) = \begin{cases} 1 & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

Show the steps in your calculations.

2.8

$$\text{Rect}(x) = \begin{cases} b & \text{for } |x| < a \\ 0 & \text{otherwise} \end{cases}$$

(a) What is $\text{Rect}(x) * \delta(x-a)$?

(b) What is the Fourier transform of $f(x)$ where $f(x) = \text{Rect}(x+c) + \text{Rect}(x-c)$ and $c >$ a?

2.9 A digitizer has a sampling interval of $\Delta x = \Delta y = \Delta$. Which of the following images can be represented unambiguously by their samples? (Assume that effects of a finite image domain can be neglected.)

(a) $(sin\,(\pi x/\Delta))/(\pi x/\Delta)$

(b) $cos\,(\pi/x/2\Delta)\cos(3\pi x/4\Delta)$

(c) $\text{Rect}(x)$ (see Problem 2.8)

(d) e^{-ax^2}

REFERENCES

AGIN, G. J. "Representation and description of curved objects" (Ph.D. dissertation). AIM-173, Stanford AI Lab, October 1972.

ANDREWS, H. C. and B. R. HUNT. *Digital Image Restoration.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1977.

CROWTHER, R. A. and A. KLUG. "ART and science, or, conditions for 3-d reconstruction from electron microscope images." *J. Theoretical Biology 32*, 1971.

DEROSIER, D. J. "The reconstruction of three-dimensional images from electron micrographs." *Contemporary Physics 12*, 1971.

DUDA, R. O. and P. E. HART. *Pattern Recognition and Scene Analysis.* New York: Wiley, 1973.

GARVEY, T. D. "Perceptual strategies for purposive vision." Technical Note 117, AI Center, SRI International, September 1976.

GONZALEZ, R. C. and P. WINTZ. *Digital Image Processing.* Reading, MA: Addison-Wesley, 1977.

GORDON, R. and G. T. HERMAN. "Three-dimensional reconstruction from projections: a review of algorithms." *International Review of Cytology 38*, 1974, 111–151.

GORDON, R., G. T. HERMAN, and S. A. JOHNSON. "Image reconstruction from projections." *Scientific American*, October 1975.

HERING, E. "Principles of a new theory of color sense." *In Color Vision*, R.C. Teevan and R.C. Birney (Eds.). Princeton, NJ: D. Van Nostrand, 1961.

HORN, B. K. P. "Understanding image intensities." *Artificial Intelligence 8*, 2, April 1977, 201–231.

HORN, B. K. P. and R. W. SJOBERG. "Calculating the reflectance map." *Proc.*, DARPA IU Workshop, November 1978, 115–126.

HURVICH, L. M. and D. JAMESON. "An opponent-process theory of color vision." *Psychological Review 64*, 1957, 384–390.

JAIN, A. K. "Advances in mathematical models for image processing." *Proc. IEEE 69*, 5, May 1981, 502-528.

JOBLOVE, G. H. and D. GREENBERG. "Color spaces for computer graphics." *Computer Graphics 12*, 3, August 1978, 20–25.

KENDER, J. R. "Saturation, hue, and normalized color: calculation, digitization effects, and use." Technical Report, Dept. of Computer Science, Carnegie-Mellon Univ., November 1976.

LAND, E. H. "The retinex theory of color vision." *Scientific American,* December 1977, 108–128.

MUNSELL, A. H. *A Color Notation,* 8th ed. Baltimore, MD: Munsell Color Co., 1939.

NICODEMUS, F. E., J. C. RICHMOND, J. J. HSIA, I. W. GINSBERG, and T. LIMPERIS. "Geometrical considerations and nomenclature for reflectance." NBS Monograph 160, National Bureau of Standards, U.S. Department of Commerce, Washington, DC, October 1977.

NITZAN, D., A. BRAIN, and R. DUDA. "The measurement and use of registered reflectance and range data in scene analysis." *Proc. IEEE 65,* 2, February 1977.

POPPLESTONE, R. J., C. M. BROWN, A. P. AMBLER, and G. F. CRAWFORD. "Forming models of plane-and-cylinder faceted bodies from light stripes." *Proc.,* 4th IJCAI, September 1975, 664-668.

PRATT, W. K. *Digital Image Processing.* New York: Wiley-Interscience, 1978.

ROSENFIELD A. and A. C. KAK. *Digital Picture Processing.* New York: Academic Press, 1976.

SMITH, A. R. "Color gamut transform pairs." *Computer Graphics 12,* 3, August 1978, 12–19.

SUGIHARA, K. "Dictionary-guided scene analysis based on depth information." In *Progress Report on 3-D Object Recognition.* Bionics Research Section, ETL, Tokyo, March 1977.

TENENBAUM, J. M. and S. WEYL. "A region-analysis subsystem for interactive scene analysis." *Proc.,* 4th IJCAI, September 1975, 682–687.

WAAG, R. B. and R. GRAMIAK. "Methods for ultrasonic imaging of the heart." *Ultrasound in Medicine and Biology 2,* 1976, 163–170.

WILL, P. M. and K. S. PENNINGTON. "Grid coding: a preprocessing technique for robot and machine vision." *Artificial Intelligence 2,* 3/4, Winter 1971, 319–329.

School of informatics

Some Former DAI Web Pages Temporarily Unavailable

Following a major fire on our premises at 80 South Bridge Edinburgh on Saturday December 7th, there has been some disruption to the Department of AI web service. Most of the service has been restored, but there is still a problem with the content you have just tried to access.

Further details are available at http://www.informatics.ed.ac.uk/emergency/.

If you wish to inform the web master of the missing content, please email webadmin @ inf.ed.ac.uk.

Early Processing 3

3.1 RECOVERING INTRINSIC STRUCTURE

The imaging process confounds much useful physical information into the gray-level array. In this respect, the imaging process is a collection of degenerate transformations. However, this information is not irrevocably lost, because there is much spatial redundancy: Neighboring pixels in the image have the same or nearly the same physical parameters. A collection of techniques, which we call *early processing*, exploits this redundancy in order to undo the degeneracies in the imaging process. These techniques have the character of transformations for changing the image into "parameter images" *or intrinsic images* [Barrow and Tenenbaum 1978; 1981] which reflect the spatial properties of the scene. Common intrinsic parameters are surface discontinuities, range, surface orientation, and velocity.

In this chapter we neglect high-level internal model information even though it is important and can affect early processing. Consider the case of the perceived central edge in Fig. 3.1a. As shown by Fig. 3.1b, which shows portions of the same image, the central edge of Fig. 3.1a is not present in the data. Nevertheless, the human perceiver "sees" the edge, and one reasonable explanation is that it is a product of an internal block model. Model-directed activity is taken up in later chapters. These examples show how high level models (e.g., circles) can affect low-level processors (e.g., edge finders). However, for the purposes of study it is often helpful to neglect these effects. These simplifications make it easier to derive the fundamental constraints between the physical parameters and gray levels. Once these are understood, they can be modified using the more abstract structures of later chapters.

Most early computer vision processing can be done with parallel computations whose inputs tend to be spatially localized. When computing intrinsic images

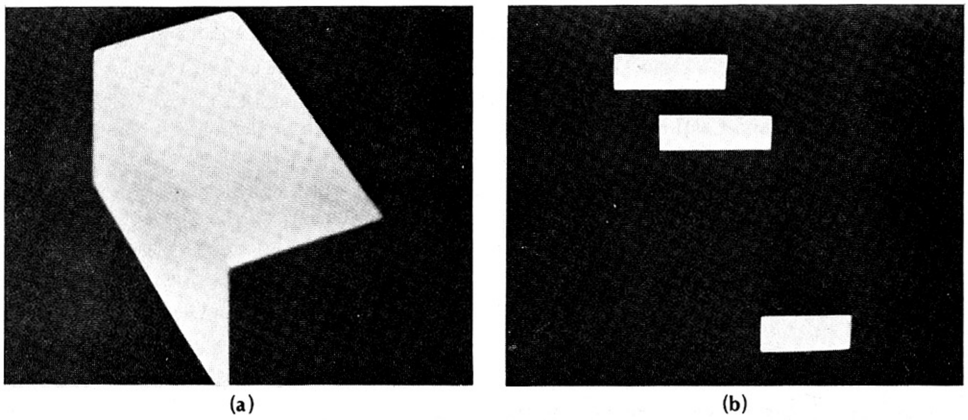


Fig. 3.1 (a) A perceived edge. (b) Portions of image in (a) showing the lack of image data.

the parallel computations are iterated until the intrinsic parameter measurements converge to a set of values. A computation that falls in the parallel-iterative category is known in computer vision as *relaxation* [Rosenfeld et al. 1976]. Relaxation is a very general computational technique that is useful in computer vision. Specific examples of relaxation computations appear throughout the book; general observations on relaxation appear in Chapter 12.

This chapter covers six categories of early processing techniques:

1. *Filtering* is a generic name for techniques of changing image gray levels to enhance the appearance of objects. Most often this means transformations that make the intensity discontinuities between regions more prominent. These transformations are often dependent on gross object characteristics. For example, if the objects of interest are expected to be relatively large, the image can be blurred to erase small intensity discontinuities while retaining those of the object's boundary. Conversely, if the objects are relatively small, a transformation that selectively removes large discontinuities may be appropriate. Filtering can also compensate for spatially varying illumination.

2. *Edge operators* detect and measure very local discontinuities in intensity or its gradient. The result of an edge operator is usually the magnitude and orientation of the discontinuity.

3. *Range transforms* use known geometry about stereo images to infer the distance of points from the viewer. These transforms make use of the inverse perspective transform to interpret how points in three-dimensional space project onto stereo pairs. A correspondence between points in two stereo images of known geometry determines the range of those points. Relative range may also be derived from local correspondences without knowing the imaging geometry precisely.

4. *Surface orientation* can be calculated if the source illumination and reflectance properties of the surface are known. This calculation is sometimes called

"shape from shading." Surface orientation is particularly simple to calculate when the source illumination can be controlled.

5. *Optical flow*, or velocity fields of image points, can be calculated from local temporal and spatial variations in sequences of gray-level images.

6. A *pyramid* is a general structure for representing copies of the image at multiple resolutions. A pyramid is a "utility structure" which can dramatically improve the speed and effectiveness of many early processing and later segmentation algorithms.

3.2 FILTERING THE IMAGE

Filtering is a very general notion of transforming the image intensities in some way so as to enhance or deemphasize certain features. We consider only transforms that leave the image in its original format: a spatial array of gray levels. Spurred on by the needs of planetary probes and aerial reconnaissance, filtering initially received more attention than any other area of image processing and there are excellent detailed reference works (e.g., [Andrews and Hunt 1977; Pratt 1978; Gonzalez and Wintz 1977]). We cannot afford to examine these techniques in great detail here; instead, our intent is to describe a set of techniques that conveys the principal ideas.

Almost without exception, the best time to filter an image is at the image formation stage, before it has been sampled. A good example of this is the way chemical stains improve the effectiveness of microscopic tissue analysis by changing the image so that diagnostic features are obvious. In contrast, filtering after sampling often emphasizes random variations in the image, termed *noise*, that are undesirable effects introduced in the sampling stage. However, for cases where the image formation process cannot be changed, digital filtering techniques do exist. For example, one may want to suppress low spatial frequencies in an image and sharpen its edges. An image filtered in this way is shown in Fig. 3.2.

Note that in Fig. 3.2 the work of recognizing real-world objects still has to be done. Yet the edges in the image, which constitute object boundaries, have been made more prominent by the filtering operation. Good filtering functions are not easy to define. For example, one hazard with Fourier techniques is that sharp edges in the filter will produce unwanted "ringing" in the spatial domain, as evidenced by Fig. 2.5. Unfortunately, it would be too much of a digression to discuss techniques of filter design. Instead, the interested reader should refer to the references cited earlier.

3.2.1 Template Matching

Template matching is a simple filtering method of detecting a particular feature in an image. Provided that the appearance of this feature in the image is known accu-

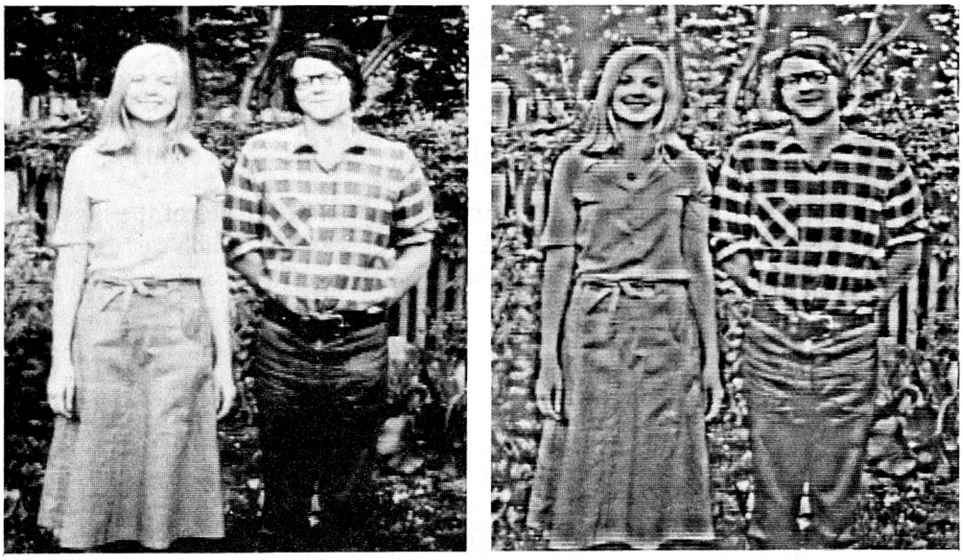

(a) (b)

Fig. 3.2 Effects of high frequency filtering. (a) Original image. (b) Filtered image.

rately, one can try to detect it with an operator called a *template*. This template is, in effect, a subimage that looks just like the image of the object. A similarity measure is computed which reflects how well the image data match the template for each possible template location. The point of maximal match can be selected as the location of the feature. Figure 3.3 shows an industrial image and a relevant template.

Correlation

One standard similarity measure between a function $f(\mathbf{x})$ and a template $t(\mathbf{x})$ is the Euclidean distance $d(\mathbf{y})$ squared, given by

$$d(\mathbf{y})^2 = \sum_{\mathbf{x}} [f(\mathbf{x}) - t(\mathbf{x} - \mathbf{y})]^2 \tag{3.1}$$

By $\sum_{\mathbf{x}}$ we mean $\sum_{x=-M}^{M} \sum_{y=-N}^{N}$, for some M, N which define the size of the template extent. If the image at point $\mathbf{y}$ is an exact match, then $d(\mathbf{y}) = 0$; otherwise, $d(\mathbf{y}) > 0$. Expanding the expression for d^2, we can see that

$$d^2(\mathbf{y}) = \sum_{\mathbf{x}} [f^2(\mathbf{x}) - 2f(\mathbf{x})t(\mathbf{x} - \mathbf{y}) + t^2(\mathbf{x} - \mathbf{y})] \tag{3.2}$$

Notice that $\sum_{\mathbf{x}} t^2(\mathbf{x} - \mathbf{y})$ is a constant term and can be neglected. When $\sum_{\mathbf{x}} f^2(\mathbf{x})$ is approximately constant it too can be discounted, leaving what is called the *cross correlation* between f and t.

$$R_{ft}(\mathbf{y}) = \sum_{\mathbf{x}} f(\mathbf{x})t(\mathbf{x} - \mathbf{y}) \tag{3.3}$$

This is maximized when the portion of the image "under" t is identical to t.

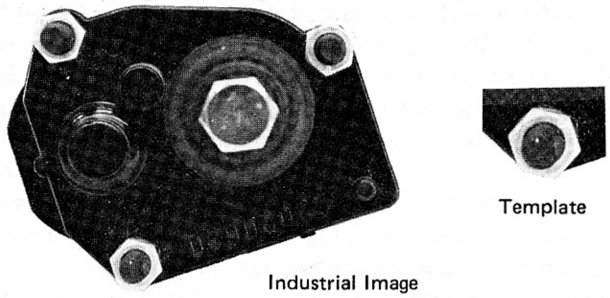


Fig. 3.3 An industrial image and template for a hexagonal nut.

One may visualize the template-matching calculations by imagining the template being *shifted* across the image to different offsets; then the superimposed values at this offset are *multiplied* together, and the products are *added*. The resulting sum of products forms an entry in the "correlation array" whose coordinates are the offsets attained by the source template.

If the template is allowed to take *all* offsets with respect to the image such that some overlap takes place, the correlation array is larger than either the template or the image. An $n \times n$ image with an $m \times m$ template yields an $(n + m - 1 \times n + m - 1)$ correlation array. If the template is not allowed to shift off the image, the correlation array is $(n - m + 1 \times n - m + 1)$; for $m < n$. Another form of correlation results from computing the offsets modulo the size of the image; in other words, the template "wraps around" the image. Being shifted off to the right, its right portion reappears on the left of the image. This sort of correlation is called *periodic* correlation, and those with no such wraparound properties are called *aperiodic*. We shall be concerned exclusively with aperiodic correlation. One can always modify the input to a periodic correlation algorithm by padding the outside with zeros so that the output is the aperiodic correlation.

Figure 3.4 provides an example of (aperiodic) "shift, add, multiply" template matching. This figure illustrates some difficulties with the simple correlation measure of similarity. Many of the advantages and disadvantages of this measure stem from the fact that it is linear. The advantages of this simplicity have mainly to do with the existence of algorithms for performing the calculation efficiently (in a transform domain) for the entire set of offsets. The disadvantages have to do with

Template	*Image*	*Correlation*
1 1 1	1 1 0 0 0	7 4 2 x x
1 1 1	1 1 1 0 0	5 3 2 x x
1 1 1	1 0 1 0 0	2 1 9 x x
	0 0 0 0 0	x x x x x
	0 0 0 0 8	x x x x x
		x = undefined

Fig. 3.4 (a) A simple template. (b) An image with noise. (c) The aperiodic correlation array of the template and image. Ideally peaks in the correlation indicate positions of good match. Here the correlation is only calculated for offsets that leave the template entirely within the image. The correct peak is the upper left one at 0, 0 offset. The "false alarm" at offset 2, 2 is caused by the bright "noise point" in the lower right of the image.

the fact that the metric is sensitive to properties of the image that may vary with the offset, such as its average brightness. Slight changes in the shape of the object, its size, orientation, or intensity values can also disturb the match.

Nonetheless, the idea of template matching is important, particularly if Eq. (3.3) is viewed as a *filtering* operation instead of an algorithm that does all the work of object detection. With this viewpoint one chooses one or more templates (filters) that transform the image so that certain features of an object are more readily apparent. These templates generally highlight subparts of the objects. One such class of templates is edge templates (discussed in detail in Section 3.3).

We showed in Section 2.2.4 that convolution and multiplication are Fourier transform pairs. Now note that the correlation operation in (3.3) is essentially the same as a convolution with a function $t'(\mathbf{x}) \equiv t(-\mathbf{x})$. Thus in a mathematical sense cross correlation and convolution are equivalent. Consequently, if the size of the template is sufficiently large, it is cheaper to perform the template matching operation in the spatial frequency domain, by the same transform techniques as for filtering.

Normalized Correlation

A crucial assumption in the development of Eq. (3.3) was that the image energy covered by the matching template at any offset was constant; this leads to a linear correlation matching technique. This assumption is approximately correct if the average image intensity varies slowly compared to the template size, but a bright spot in the image can heavily influence the correlation by affecting the sum of products violently in a small area (Fig. 3.4). Even if the image is well behaved, the range of values of the metric can vary with the size of the matching template. Are there ways of normalizing the correlation metric to make it insensitive to these variations?

There is a well-known treatment of the normalized correlation operation. It has been used for a variety of tasks involving registration and stereopsis of images [Quam and Hannah 1974]. Let us say that two input images are being matched to find the best offset that aligns them.

Let $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ be the images to be matched. q_2 is the patch of f_2 (possibly all of it) that is to be matched with a similar-sized patch of f_1. q_1 is the patch of f_1 that is covered by q_2 when q_2 is offset by $\mathbf{y}$.

Let $E()$ be the expectation operator. Then

$$\sigma(q_1) = [E(q_1^2) - (E(q_1))^2]^{1/2} \tag{3.4}$$

$$\sigma(q_2) = [E(q_2^2) - (E(q_2))^2]^{1/2} \tag{3.5}$$

give the standard deviations of points in patches q_1 and q_2. (For notational convenience, we have dropped the spatial arguments of q_1 and q_2.) Finally, the normalized correlation is

$$N(\mathbf{y}) = \frac{E(q_1 q_2) - E(q_1)E(q_2)}{\sigma(q_1)\sigma(q_2)} \tag{3.6}$$

and $E(q_1 q_2)$ is the expected value of the product of intensities of points that are superimposed by the translation by $\mathbf{y}$.

The normalized correlation metric is less dependent on the local properties of the reference and input images than is the unnormalized correlation, but it is sensitive to the signal-to-noise content of the images. High uncorrelated noise in the two images, or the image and the reference, decreases the value of the correlation. As a result, one should exercise some care in interpreting the metric. If the noise properties of the image are known, one indication of reliability is given by the "(signal + noise)-to-noise" ratio. For the normalized correlation to be useful, the standard deviation of the patches of images to be matched (i.e., of the areas of image including noise) should be significantly greater than that of the noise. Then a correlation value may be considered significant if it is approximately equal to the theoretically expected one. Consider uncorrelated noise of identical standard deviation, in a patch of true value $f(x, y)$. Let the noise component of the image be $n(x, y)$. Then the theoretical maximum correlation is

$$1 - \frac{\sigma^2(n)}{\sigma^2(f+n)} \tag{3.7}$$

In matching an idealized, noise-free reference pattern, the best expected value of the cross correlation is

$$\frac{\sigma(f)}{\sigma(f+n)} \tag{3.8}$$

If the noise and signal characteristics of the data are known, the patch size may be optimized by using that information and the simple statistical arguments above. However, such considerations leave out the effects of systematic, nonstatistical error (such as imaging distortions, rotations, and scale differences between images). These systematic errors grow with patch size, and may swamp the statistical advantages of large patches. In the worst case, they may vitiate the advantages of the correlation process altogether.

Since correlation is expensive, it is advantageous to ensure that there is enough information in the patches chosen for correlation before the operation is done. One way to do this is to apply a cheap "interest operator" before the relatively expensive correlation. The idea here is to make sure that the image varies enough to give a usable correlation image. If the image is of uniform intensity, even its correlation with itself (autocorrelation) is flat everywhere, and no information about where the image is registered with itself is derivable. The "interest operator" is a way of finding areas of image with high variance. In fact, a common and useful interest measure is exactly the (directional) variance over small areas of image. One directional variance algorithm works as follows.

The Moravec interest operator [Moravec 1977] produces candidate match points by measuring the distinctness of a local piece of the image from its surround. To explain the operator, we first define a variance measure at a pixel $(\mathbf{x})$ as

$$\text{var}(x, y) = \left\{ \sum_{k,\, l \text{ in } s} [f(x, y) - f(x+k, y+l)]^2 \right\}^{1/2} \tag{3.9}$$

$$s = \left\{ (0, a), (0, -a), (a, 0), (-a, 0) \right\}$$

where a is a parameter. Now the interest operator value is initially the minimum of itself and surrounding points:

$$\text{IntOpVal}(\mathbf{x}) := \min_{y<1} [\text{var}(\mathbf{x} + \mathbf{y})] \tag{3.10}$$

Next a check is made to see if the operator is a local maximum by checking neighbors again. Only local maxima are kept.

$$\begin{aligned} &\text{IntOpVal}(\mathbf{x}) := 0 \text{ if} \\ &\text{IntOpVal}(\mathbf{x}) \geqslant \text{IntOpVal}(\mathbf{x} + \mathbf{y}) \\ &\qquad \text{for } \mathbf{y} \leqslant 1 \end{aligned} \tag{3.11}$$

Finally, candidate points are chosen from the IntOpVal array by thresholding.

$$\mathbf{x} \text{ is a candidate point iff IntOpVal}(\mathbf{x}) > T \tag{3.12}$$

The threshold is chosen empirically to produce some fraction of the total image points.

3.2.2 Histogram Transformations

A gray-level histogram of an image is a function that gives the frequency of occurrence of each gray level in the image. Where the gray levels are quantized from 0 to n, the value of the histogram at a particular gray level p, denoted $h(p)$, is the number or fraction of pixels in the image with that gray level. Figure 3.5 shows an image with its histogram.

A histogram is useful in many different ways. In this section we consider the histogram as a tool to guide gray-level transformation algorithms that are akin to filtering. A very useful image transform is called *histogram equalization.* Histogram equalization defines a mapping of gray levels p into gray levels q such that the distribution of gray levels q is uniform. This mapping stretches contrast (expands the

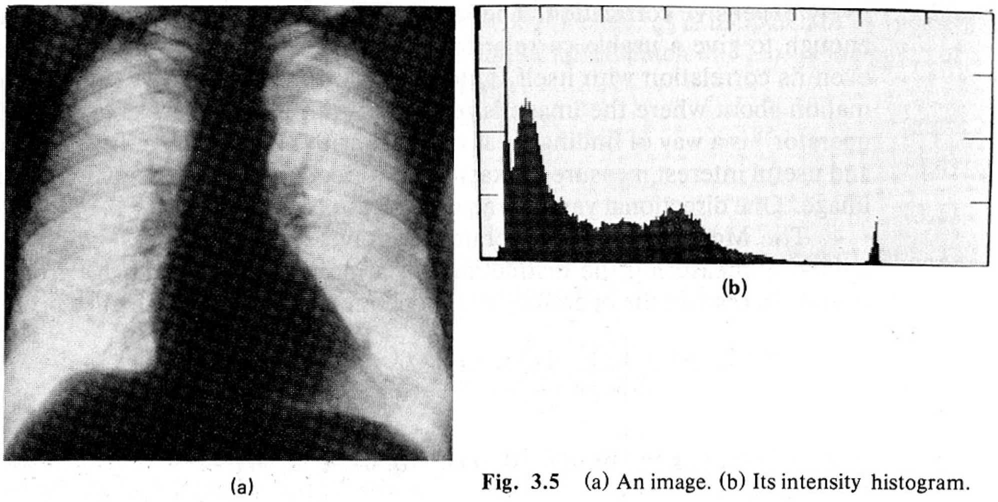

(a) (b)

Fig. 3.5 (a) An image. (b) Its intensity histogram.

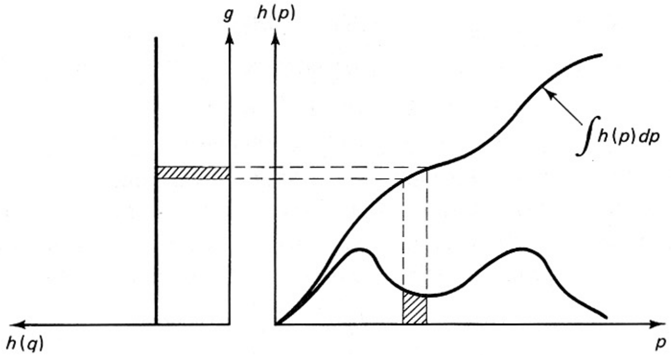
range of gray levels) for gray levels near histogram maxima and compresses contrast in areas with gray levels near histogram minima. Since contrast is expanded for most of the image pixels, the transformation usually improves the detectability of many image features.

The histogram equalization mapping may be defined in terms of the *cumulative* histogram for the image. To see this, consider Fig. 3.6a. To map a small interval of gray levels dp onto an interval dq in the general case, it must be true that

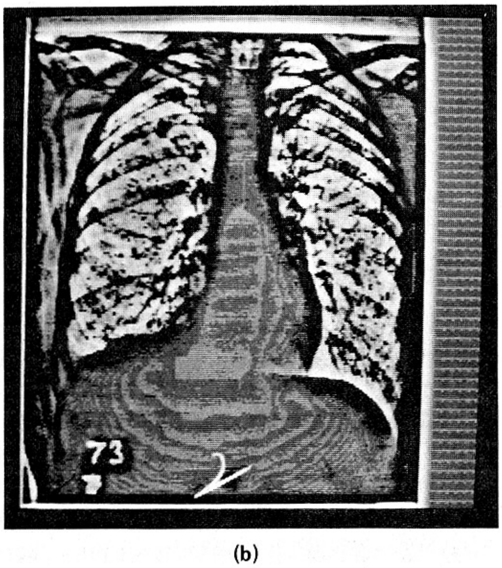
$$g(q)\,dq = h(p)\,dp \tag{3.13}$$

where $g(q)$ is the new histogram. If, in the histogram equalization case, $g(q)$ is to be uniform, then

$$g(q_2) = \frac{N^2}{M} \tag{3.14}$$



(a)



(b)

Fig. 3.6 (a) Basis for a histogram equalization technique. (b) Results of histogram equalization.

where N^2 is the number of pixels in the image and M is the number of gray levels. Thus combining Eqs. (3.13) and (3.14) and integrating, we have

$$g(q) = \frac{M}{N^2} \int_0^q h(p)\, dp \tag{3.15}$$

But Eq. (3.15) is simply the equation for the normalized cumulative histogram. Figure 3.6b shows the histogram-equalized image.

3.2.3 Background Subtraction

Background subtraction can be another important filtering step in early processing. Many images can have slowly varying background gray levels which are incidental to the task at hand. Examples of such variations are:

- Solution gradients in cell slides
- Lighting variations on surfaces in office scenes
- Lung images in a chest radiograph

Note that the last example is only a "background" in the context of looking for some smaller variations such as tumors or pneumoconiosis.

Background subtraction attempts to remove these variations by first approximating them (perhaps analytically) with a background image f_b and then subtracting this approximation from the original image. That is, the new image f_n is

$$f_n(\mathbf{x}) = f(\mathbf{x}) - f_b(\mathbf{x}) \tag{3.16}$$

Various functional forms have been tried for analytic representations of slowly varying backgrounds. In the simplest cases, $f_b(x)$ may be a constant,

$$f_b(\mathbf{x}) = c \tag{3.17}$$

or linear,

$$f_b(\mathbf{x}) = \mathbf{m}\cdot\mathbf{x} + c \tag{3.18}$$

A more sophisticated background model is to use a low-pass filtered variant of the original image:

$$f_b(\mathbf{x}) = \mathcal{F}^{-1}[H(\mathbf{u})\, F(\mathbf{u})] \tag{3.19}$$

where $H(\mathbf{u})$ is a low-pass filtering function. The problem with this technique is that it is global; one cannot count on the "best" effect in any local area since the filter treats all parts of the image identically. For the same reason, it is difficult to design a Fourier filter that works for a number of very different images.

A workable alternative is to approximate $f_b(\mathbf{x})$, using *splines*, which are piecewise polynomial approximation functions. The mathematics of splines is treated in Chapter 8 since they find more general application as representations of shape. The filtering application is important but specialized. The attractive feature of a spline approximation for filtering is that it is *variation diminishing* and *spatially variant.* The spline approximation is guaranteed to be "smoother" than the original function and will approximate the background differently in different parts of the image. The latter feature distinguishes the method from Fourier-domain techniques which are spatially invariant. Figure 3.7 shows the results of spline filtering.

3.2.4 Filtering and Reflectance Models

Leaving the effects of imaging geometry implicit (Section 2.2.2), the definitions in Section 2.2.3 imply that the image irradiance (gray level) at the image point x' is proportional to the product of the scene irradiance E and the *reflectance* r at its corresponding world point $\mathbf{x}$.

$$f(\mathbf{x}') = E(\mathbf{x})r(\mathbf{x}) \tag{3.20}$$

The irradiance at x is the sum of contributions from all illumination sources, and the reflectance is that portion of the irradiance which is reflected toward the observer (camera). Usually E changes slowly over a scene, whereas r changes quickly over edges, due to varying face angles, paint, and so forth. In many cases one would like to detect these changes in r while ignoring changes in E. One way of doing this is to filter the image $f(\mathbf{x}')$ to eliminate the slowly varying component. However, as f is the *product* of illumination and reflectance, it is difficult to define an operation that selectively diminishes E while retaining r. Furthermore, such an operation must retain the positivity of f. One solution is to take the logarithm of Eq. (3.20). Then

$$\log f = \log E + \log r \tag{3.21}$$

Equation (3.21) shows two desirable properties of the logarithmic transformation: (1) the logarithmic image is positive in sign, and (2) the image is a superposition of the irradiance component and reflectance component. Since reflectance is an in-

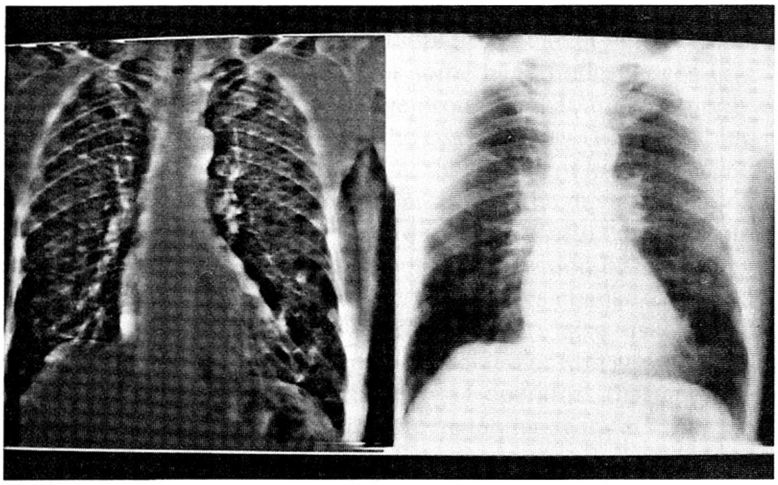


Fig. 3.7 The results of spline filtering to remove background variation.

trinsic characteristic of objects, the obvious goal of image analysis is to recognize the reflectance component under various conditions of illumination. Since the separation of two components is preserved under linear transformations and the irradiance component is usually of low spatial frequency compared to the reflectance component, filtering techniques can suppress the irradiance component of the signal relative to the reflectance component.

If the changes in r occur over very short distances in the images, r may be isolated by a three-step process [Horn 1974]. First, to enhance reflectance changes, the image function is differentiated (Section 3.3.1). The second step removes the low irradiance gradients by thresholding. Finally, the resultant image is integrated to obtain an image of perceived "lightness" or reflectance. Figure 3.8 shows these steps for the one-dimensional case.

A basic film parameter is density, which is proportional to the logarithm of transmitted intensity; the logarithmically transformed image is effectively a *density image.* In addition to facilitating the extraction of lightness, another advantage of the density image is that it is well matched to our visual experience. The ideas for many image analysis programs stem from our visual inspection of the image. However, the human visual system responds logarithmically to light intensity and also enhances high spatial frequencies [Stockham 1972]. Algorithms derived from

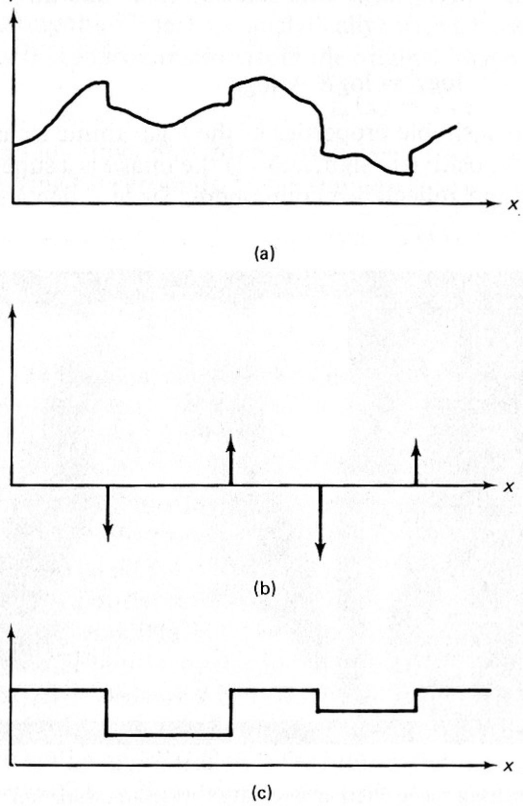


Fig. 3.8 Steps in processing an image to detect reflectance. (a) Original image. (b) Differentiation followed by thresholding. (c) Integration of function in (b).

introspective reasoning about the perceived image (which has been transformed by our visual system) will not necessarily be successful when applied to an unmodified intensity image. Thus one argument for using a density transformation followed by high spatial frequency emphasis filtering is that the computer is then "seeing" more like the human image analyzer.

3.3 FINDING LOCAL EDGES

Boundaries of objects tend to show up as intensity discontinuities in an image. Experiments with the human visual system show that boundaries in images are extremely important; often an object can be recognized from only a crude outline [Attneave 1954]. This fact provides the principal motivation for representing objects by their boundaries. Also, the boundary representation is easy to integrate into a large variety of object recognition algorithms.

One might expect that algorithms could be designed that find the boundaries of objects directly from the gray-level values in the image. But when the boundaries have complicated shapes, this is difficult. Much greater success has been obtained by first transforming the image into an intermediate image of *local* gray-level discontinuities, or edges, and then composing these into a more elaborate boundary. This strategy reflects the principle: When the gap between representations becomes too large, introduce intermediate representations. In this case, boundaries that are highly model-dependent may be decomposed into a series of local edges that are highly model-independent.

A local edge is a small area in the image where the local gray levels are changing rapidly in a simple (e.g., monotonic) way. An *edge operator* is a mathematical operator (or its computational equivalent) with a small spatial extent designed to detect the presence of a local edge in the image function.

It is difficult to specify a priori which local edges correspond to relevant boundaries in the image. Depending on the particular task domain, different local changes will be regarded as likely edges. Plots of gray level versus distance along the direction perpendicular to the edge for some hypothetical edges (Fig. 3.9a-e) demonstrate some different kinds of "edge profiles" that are commonly encountered. Of course, in most practical cases, the edge is noisy (Fig. 3.9d) and may appear as a composite of profile types. The fact that different kinds of edge operators perform best in different task domains has prompted the development of a variety of operators. However, the unifying feature of most useful edge operators is that they compute a *direction* which is aligned with the direction of maximal gray-level change, and a *magnitude* describing the severity of this change. Since edges are a high-spatial-frequency phenomenon, edge finders are also usually sensitive to high-frequency noise, such as "snow" on a TV screen or film grain.

Operators fall into three main classes: (1) operators that approximate the mathematical gradient operator, (2) template matching operators that use multiple templates at different orientations, and (3) operators that fit local intensities with parametric edge models. Representative examples from the first two of these categories appear in this section. The computer vision literature abounds with edge introspective reasoning about the perceived image (which has been transformed by our visual system) will not necessarily be successful when applied to an unmodified intensity image. Thus one argument for using a density transformation followed by high spatial frequency emphasis filtering is that the computer is then "seeing" more like the human image analyzer.

3.3 FINDING LOCAL EDGES

Boundaries of objects tend to show up as intensity discontinuities in an image. Experiments with the human visual system show that boundaries in images are extremely important; often an object can be recognized from only a crude outline [Attneave 1954]. This fact provides the principal motivation for representing objects by their boundaries. Also, the boundary representation is easy to integrate into a large variety of object recognition algorithms.

One might expect that algorithms could be designed that find the boundaries of objects directly from the gray-level values in the image. But when the boundaries have complicated shapes, this is difficult. Much greater success has been obtained by first transforming the image into an intermediate image of *local* gray-level discontinuities, or edges, and then composing these into a more elaborate boundary. This strategy reflects the principle: When the gap between representations becomes too large, introduce intermediate representations. In this case, boundaries that are highly model-dependent may be decomposed into a series of local edges that are highly model-independent.

A local edge is a small area in the image where the local gray levels are changing rapidly in a simple (e.g., monotonic) way. An *edge operator* is a mathematical operator (or its computational equivalent) with a small spatial extent designed to detect the presence of a local edge in the image function.

It is difficult to specify a priori which local edges correspond to relevant boundaries in the image. Depending on the particular task domain, different local changes will be regarded as likely edges. Plots of gray level versus distance along the direction perpendicular to the edge for some hypothetical edges (Fig. 3.9a-e) demonstrate some different kinds of "edge profiles" that are commonly encountered. Of course, in most practical cases, the edge is noisy (Fig. 3.9d) and may appear as a composite of profile types. The fact that different kinds of edge operators perform best in different task domains has prompted the development of a variety of operators. However, the unifying feature of most useful edge operators is that they compute a *direction* which is aligned with the direction of maximal gray-level change, and a *magnitude* describing the severity of this change. Since edges are a high-spatial-frequency phenomenon, edge finders are also usually sensitive to high-frequency noise, such as "snow" on a TV screen or film grain.

Operators fall into three main classes: (1) operators that approximate the mathematical gradient operator, (2) template matching operators that use multiple templates at different orientations, and (3) operators that fit local intensities with parametric edge models. Representative examples from the first two of these categories appear in this section. The computer vision literature abounds with edge

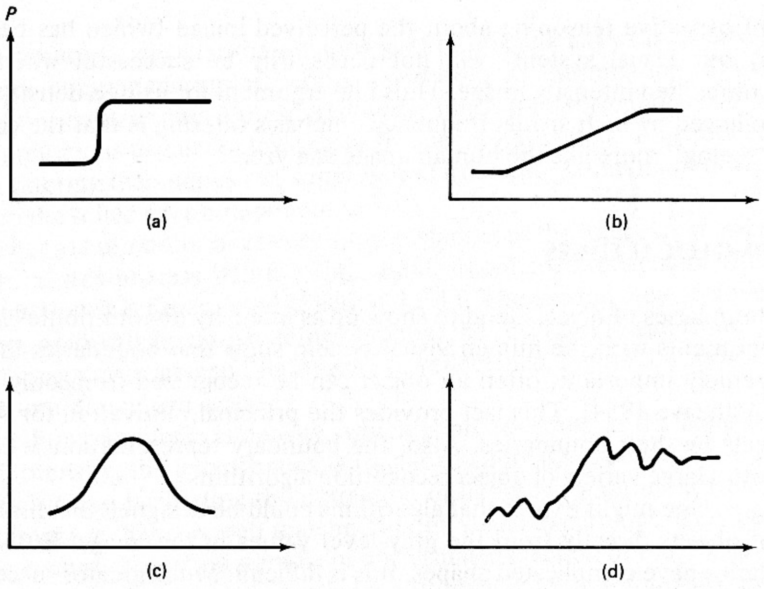


Fig. 3.9 Edge profiles.

operators, and we make no attempt to summarize them all here. For a guide to this literature, see [Rosenfeld and Kak 1976].

Parametric models generally capture more detailed edge structure than the two-parameter direction and magnitude vector; as a result, they can be more computationally complicated. For this reason and others discussed in Section 3.3.4, we shall omit a detailed discussion of these kinds of edge operators. One of the best known parametric models is Hueckel's [Hueckel 1971, 1973], but several others have been developed since [Mero and Vassy 1975; Nevatia 1977; Abdou 1978; Tretiak 1979].

3.3.1 Types of Edge Operators

Gradient and Laplacian

The most common and historically earliest edge operator is the gradient [Roberts 1965]. For an image function $f(\mathbf{x})$, the gradient magnitude $s(\mathbf{x})$ and direction $\phi(\mathbf{x})$ can be computed as

$$s(\mathbf{x}) = (\Delta_1^2 + \Delta_2^2)^{1/2} \tag{3.22}$$

$$\phi(\mathbf{x}) = \operatorname{atan}(\Delta_2, \Delta_1) \tag{3.23}$$

where

$$\Delta_1 = f(x+n, y) - f(x, y) \tag{3.24}$$
$$\Delta_2 = f(x, y+n) - f(x, y)$$

n is a small integer, usually unity, and atan (x, y) returns $\tan^{-1}(x/y)$ adjusted to the proper quadrant. The parameter n is called the "span" of the gradient. Roughly, n should be small enough so that the gradient is a good approximation to the local changes in the image function, yet large enough to overcome the effects of small variations in f.

Equation (3.24) is only one *difference operator*, or way of measuring gray-level intensities along orthogonal directions using Δ_1 and Δ_2. Figure 3.10 shows the gradient difference operators compared to other operators [Roberts 1965; Prewitt 1970]. The reason for the modified operators of Prewitt and Sobel is that the local averaging tends to reduce the effects of noise. These operators do, in fact, perform better than the Roberts operator for a step edge model.

One way to study an edge operator's performance is to use an ideal edge such as the step edge shown in Fig. 3.11. This edge has two gray levels: zero and h units. If the edge goes through the finite area associated with a pixel, the pixel is given a value between zero and h, depending on the proportion of its area covered. Comparative edge operator performance has been carried out [Abdou 1978]. In the case of the Sobel operator (Fig. 3.10c) the measured orientation ϕ' is given by

$$\Delta_1 \qquad \Delta_2$$

$$\begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline\end{array} \qquad \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array}$$

(a)

$$\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline\end{array}$$

(b)

$$\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$$

(c)

Fig. 3.10 Gradient operators.

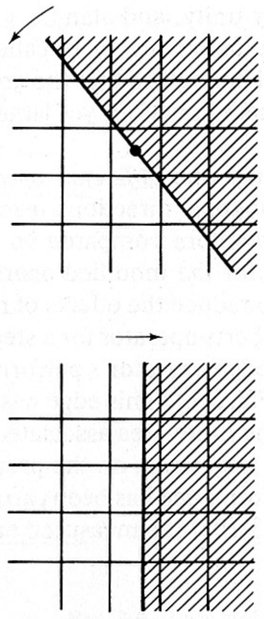


Fig. 3.11 Edge models for orientation and displacement sensitivity analyses.

$$\phi' = \begin{cases} \phi & \text{if } 0 \leqslant \phi \leqslant \tan^{-1}\left(\dfrac{1}{3}\right) \\ \tan^{-1}\left(\dfrac{7\tan^2\phi + 6\tan\phi - 1}{-9\tan^2\phi + 22\tan\phi - 1}\right) & \text{if } \tan^{-1}\left(\dfrac{1}{3}\right) \leqslant 0 \leqslant \phi \leqslant \pi/4 \end{cases} \qquad (3.25)$$

Arguments from symmetry show that only the $0 \leqslant \phi < \pi/4$ cases need be examined. Similar studies could be made using ramp edge models.

A rather specialized kind of gradient is that taken "between pixels." This scheme is shown in Fig. 3.12. Here a pixel may be thought of as having four *crack edges* surrounding it, whose directions of are fixed by the pixel to be multiples of $\pi/2$. The magnitude of the edge is determined by $|f(\mathbf{x}) - f(\mathbf{y})|$, where $\mathbf{x}$ and $\mathbf{y}$ are the coordinates of the pixels that have the edge in common. One advantage of this formulation is that it provides an effective way of separating regions and their boundaries. The disadvantage is that the edge orientation is crude.

The *Laplacian* is an edge detection operator that is an approximation to the mathematical Laplacian $\partial^2 f/\partial x^2 + \partial^2 f/\partial y^2$ in the same way that the gradient is an approximation to the first partial derivatives. One version of the discrete Laplacian is given by

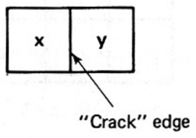


Fig. 3.12 "Crack" edge representation.

$$L(x, y) = f(x, y) - \tfrac{1}{4}[f(x, y+1) + f(x, y-1) + f(x+1, y) + f(x-1, y)] \tag{3.26}$$

The Laplacian has two disadvantages as an edge measure: (1) useful directional information is not available, and (2) the Laplacian, being an approximation to the second derivative, doubly enhances any noise in the image. Because of these disadvantages, the Laplacian has fallen into disuse, although some authors have used it as an adjunct to the gradient [Wechsler and Sklansky 1977; Akatsuka 1974] in the following manner: There is an edge at $\mathbf{x}$ with magnitude $g(\mathbf{x})$ and direction $\phi(\mathbf{x})$ if $g(\mathbf{x}) > T_1$ and $L(\mathbf{x}) > T_2$.

Edge Templates

The *Kirsch operator* [Kirsch 1971] is related to the edge gradient and is given by

$$S(\mathbf{x}) = \max\left[1, \max_k \sum_{k-1}^{k+1} f(\mathbf{x}_k)\right] \tag{3.27}$$

where $f(\mathbf{x}_k)$ are the eight neighboring pixels to $\mathbf{x}$ and where subscripts are computed modulo 8. A 3-bit direction can also be extracted from the value of k that yields the maximum in (3.27). In practice, "pure" template matching has replaced the use of (3.27). Four separate templates are matched with the image and the operator reports the magnitude and direction associated with the maximum match. As one might expect, the operator is sensitive to the magnitude of $f(\mathbf{x})$, so that in practice variants using large templates are generally used. Figure 3.13 shows Kirsch-motivated templates with different spans.

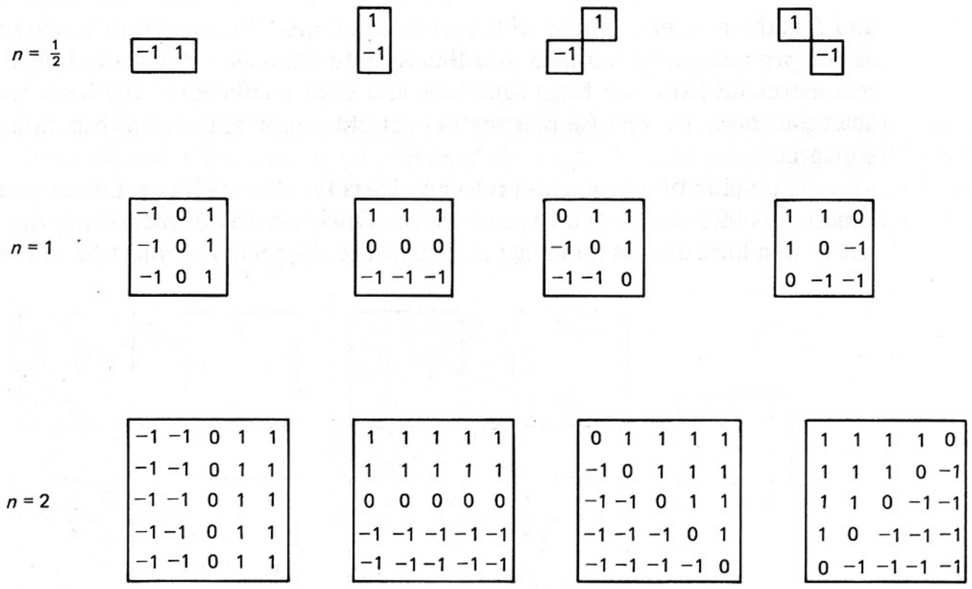


Fig. 3.13 Kirsch templates.

This brief discussion of edge templates should not be construed as a comment on their appropriateness or popularity. In fact, they are widely used, and the template-matching concept is the essence of the other approaches. There is also evidence that the mammalian visual system responds to edges through special low-level template-matching edge detectors [Hubel and Wiesel 1979].

3.3.2 Edge Thresholding Strategies

For most images there will be but few places where the gradient magnitude is equal to zero. Furthermore, in the absence of any special context, small magnitudes are most likely to be due to random fluctuations, or noise in the image function f. Thus in practical cases one may use the expedient of only reporting an edge element at $\mathbf{x}$ if $g(\mathbf{x})$ is greater than some threshold, in order to reduce these noise effects.

This strategy is computationally efficient but may not be the best. An alternative thresholding strategy [Frei and Chen 1977] views difference operators as part of a set of orthogonal basis functions analogous to the Fourier basis of Section 2.2.4. Figure 3.14 shows the nine Frei–Chen basis functions. Using this basis, the image near a point $\mathbf{x}_0$ can be represented as

$$f(\mathbf{x}) = \sum_{k=1}^{8} (f,\ h_k)\, h_k(\mathbf{x} - \mathbf{x}_0) \,/\, (h_k,\ h_k) \tag{3.28}$$

where the $(f,\ h_k)$ is the correlation operation given by

$$(f,\ h_k) = \sum_{D} f(\mathbf{x}_0)\, h_k(\mathbf{x} - \mathbf{x}_0) \tag{3.29}$$

and D is the nonzero domain of the basis functions. This operation is also regarded as the *projection* of the image into the basis function h_k. When the image can be reconstructed from the basis functions and their coefficients, the basis functions span the space. In the case of a smaller set of functions, the basis functions span a subspace.

The value of a projection into any basis function is highest when the image function is identical to the basis function. Thus one way of measuring the "edgeness" of a local area in an image is to measure the relative projection of the image

$$\boxed{\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}}$$

$$\boxed{\begin{matrix} -1 & \sqrt{2} & -1 \\ & & \\ 1 & \sqrt{2} & 1 \end{matrix}} \quad \boxed{\begin{matrix} & 1 & \\ -1 & & 1 \\ & -1 & \end{matrix}} \quad \boxed{\begin{matrix} & 1 & \\ -1 & & 1 \\ & -1 & \end{matrix}} \quad \boxed{\begin{matrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{matrix}}$$

$$\boxed{\begin{matrix} -1 & & 1 \\ \sqrt{2} & & \sqrt{2} \\ -1 & & 1 \end{matrix}} \quad \boxed{\begin{matrix} \sqrt{2} & -1 & \\ -1 & & 1 \\ & 1 & \sqrt{2} \end{matrix}} \quad \boxed{\begin{matrix} -1 & & 1 \\ & & \\ 1 & & -1 \end{matrix}} \quad \boxed{\begin{matrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{matrix}}$$

Fig. 3.14 Frei-Chen orthogonal basis.

into the edge basis functions. The relative projection into the particular "edge subspace" is given by

$$\cos\theta = \left(\frac{E}{S}\right)^{1/2} \tag{3.30}$$

where

$$E = \sum_{k=1}^{2} (f, h_k)^2$$

and

$$S = \sum_{k=0}^{8} (f, h_k)^2$$

Thus if $\theta < T$, report an edge; otherwise, not. Figure 3.15 shows the potential advantage of this technique compared to the technique of thresholding the gradient magnitude, using two hypothetical projections B_1 and B_2. Even though B_2 has a small magnitude, its relative projection into edge subspace is large and thus would be counted as an edge with the Frei–Chen criterion. This is not true for B_1.

Under many circumstances it is appropriate to use model information about the image edges. This information can affect the way the edges are interpreted after they have been computed or it may affect the computation process itself. As an example of the first case, one may still use a gradient operator, but vary the threshold for reporting an edge. Many versions of the second, more extreme strategies of using special spatially variant detection methods have been tried [Pingle and Tenenbaum 1971; Griffith 1973; Shirai 1975]. The basic idea is illustrated in Fig. 3.16. Knowledge of the orientation of an edge allows a special orientation-sensitive operator to be brought to bear on it.

3.3.3 Three-Dimensional Edge Operators

In many imaging applications, particularly medicine, the images are three-dimensional. Consider the examples of the reconstructed planes described in Sections 1.1 and 2.3.4. The medical scanner that acquires these data follows several parallel image planes, effectively producing a three-dimensional volume of data.

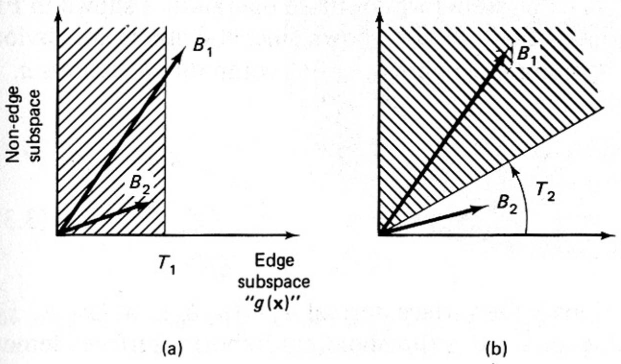


Fig. 3.15 Comparison of thresholding techniques.

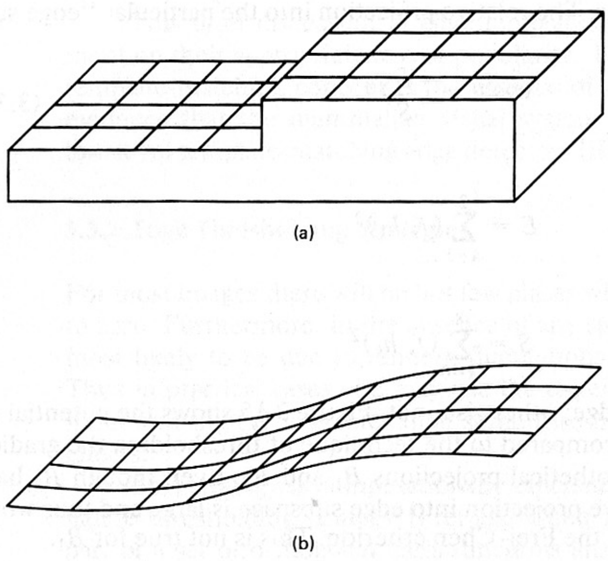


Fig. 3.16 Model-directed edge detection.

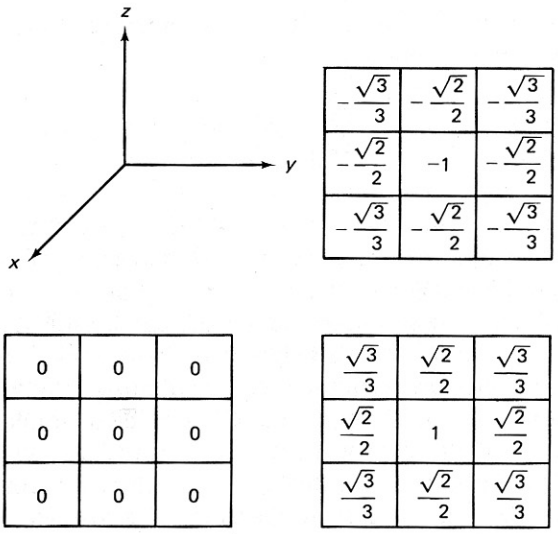
In three-dimensional data, boundaries of objects are surfaces. Edge elements in two dimensions become surface elements in three dimensions. The two-dimensional image gradient, when generalized to three dimensions, is the local surface normal. Just as in the two-dimensional case, many different basis operators can be used [Liu 1977; Zucker and Hummel 1979]. That of Zucker and Hummel uses an optimal basis assuming an underlying continuous model. We shall just describe the operator here; the proof of its correctness given the continuous image model may be found in the reference. The basis functions for the three-dimensional operator are given by

$$g_1(x, y, z) = \frac{x}{r} \tag{3.31}$$
$$g_2(x, y, z) = \frac{y}{r}$$
$$g_3(x, y, z) = \frac{z}{r}$$

where $r = (x^2 + y^2 + z^2)^{1/2}$. The discrete form of these operators is shown in Fig. 3.17 for a $3 \times 3 \times 3$ pixel domain D. Only g_1 is shown since the others are obvious by symmetry. To apply the operator at a point $x_0, , y_0, z_0$ compute projections a, b, and c, where

$$\begin{aligned} a &= (g_1, f) = \sum_D g_1(\mathbf{x}) f(\mathbf{x}-\mathbf{x}_0) \\ b &= (g_2, f) \\ c &= (g_3, f) \end{aligned} \tag{3.32}$$

The result of these computations is the surface normal $\boldsymbol{n} = (a, b, c)$ at (x_0, y_0, z_0). Surface thresholding is analogous to edge thresholding: Report a surface element



$-\frac{\sqrt{3}}{3}$	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{3}}{3}$
$-\frac{\sqrt{2}}{2}$	-1	$-\frac{\sqrt{2}}{2}$
$-\frac{\sqrt{3}}{3}$	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{3}}{3}$

0	0	0
0	0	0
0	0	0

$\frac{\sqrt{3}}{3}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{3}$
$\frac{\sqrt{2}}{2}$	1	$\frac{\sqrt{2}}{2}$
$\frac{\sqrt{3}}{3}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{3}$

Fig. 3.17 The $3 \times 3 \times 3$ edge basis function $g_1(x, y, z)$.

only if $s(x, y, z) = |\boldsymbol{n}|$ exceeds some threshold. Figure 3.18 shows the results of applying the operator to a synthetic three-dimensional image of a torus. The display shows small detected surface patches.

3.3.4 How Good are Edge Operators?

The plethora of edge operators is very difficult to compare and evaluate. For example, some operators may find most edges but also respond to noise; others may be

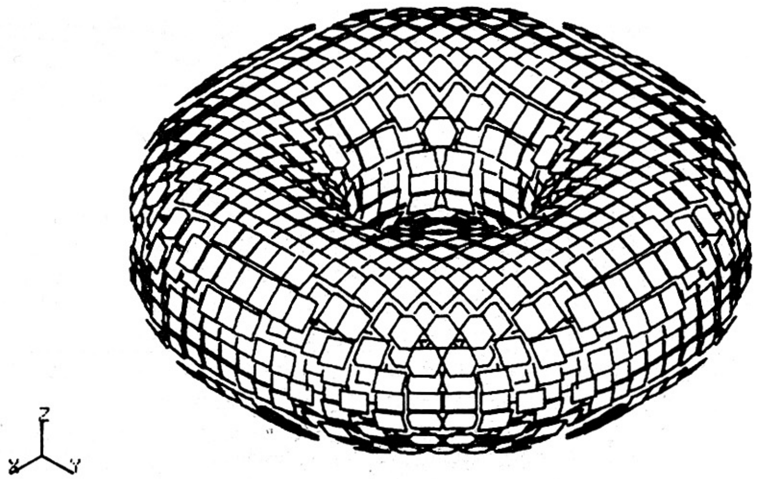


Fig. 3.18 Results of applying the Zucker-Hummel 3-D operator to synthetic image data in the shape of a torus.

noise-insensitive but miss some crucial edges. The following figure of merit [Pratt 1978] may be used to compare edge operators:

$$F = \frac{1}{\max(N_A, N_I)} \sum_{i=1}^{N_A} \frac{1}{1 + (ad_i^2)} \tag{3.33}$$

where N_A and N_I represent the number of actual and ideal edge points, respectively, a is a scaling constant, and d is the signed separation distance of an actual edge point normal to a line of ideal edge points. The term ad_i^2 penalizes detected edges which are offset from their true position; the penalty can be adjusted via a. Using this measure, all operators have surprisingly similar behaviors. Unsurprisingly, the performance of each deteriorates in the presence of noise [Abdou 1978]. (Pratt defines a signal-to-noise ratio as the square of the step edge amplitude divided by the standard deviation of Gaussian white noise.) Figure 3.19 shows some typical curves for different operators. To make this figure, the threshold for reporting an edge was chosen independently for each operator so as to maximize Eq. (3.33).

These comparisons are important as they provide a gross measure of differences in performance of operators even though each operator embodies a specific edge model and may be best in special circumstances. But perhaps the more important point is that since all real-world images have significant amounts of noise, all edge operators will generally produce imperfect results. This means that in considering the overall computer vision problem, that of building descriptions of objects, the efforts are usually best spent in developing methods that can use or improve the measurements from unreliable edges rather than in a search for the ideal edge detector.

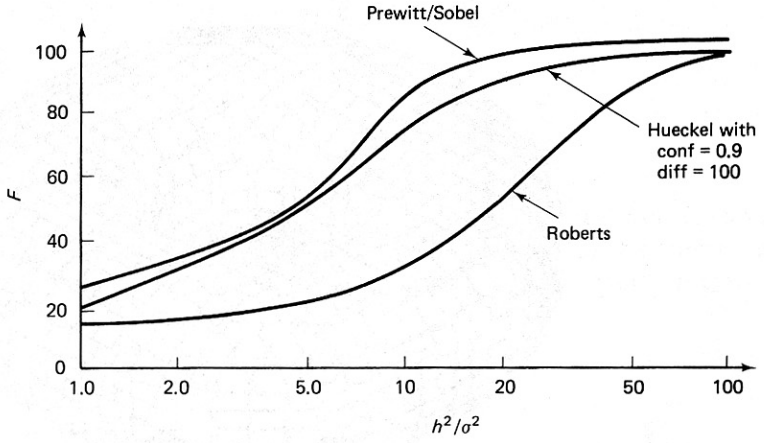


Fig. 3.19 Edge operator performance using Pratt's measure (Eq. 3.33).

3.3.5 Edge Relaxation

One way to improve edge operator measurements is to adjust them based on measurements of neighboring edges. This is a natural thing to want to do: If a weak horizontal edge is positioned between two strong horizontal edges, it should gain credibility. The edges can be adjusted based on local information using parallel-iterative techniques. This sort of process is related to more global analysis and is complementary to sequential approaches such as edge tracking (Chapter 4).

Early cooperative edge detection techniques used pairwise measurements between pixels [Zucker et al. 1977]. A later version [Prager 1980] allows for more complicated adjustment formulas. In describing the edge relaxation scheme, we essentially follow Prager's development and use the crack edges described at the end of the discussion on gradients (Sec. 3.31). The development can be extended to the other kinds of edges and the reader is invited to do just this in the Exercises.

The overall strategy is to recognize local edge patterns which cause the confidence in an edge to be modified. Prager recognizes three groups of patterns: patterns where the confidence of an edge can be increased, decreased, or left the same. The overall structure of the algorithm is as follows:

Algorithm 3.1 Edge Relaxation

0. Compute the initial confidence of each edge $C^0(e)$ as the normalized gradient magnitude normalized by the maximum gradient magnitude in the image.

1. $k = 1$;

2. Compute each edge type based on the confidence of edge neighbors;

3. Modify the confidence of each edge $C^k(e)$ based on its edge type and its previous confidence $C^{k-1}(e)$;

4. Test the $C^k(e)$'s to see if they have all converged to either 0 or 1. If so, stop; else, increment k and go to 2.

The two important parts of the algorithm are step 2, computing the edge type, and step 3, modifying the edge confidence.

The edge-type classification relies on the notation for edges (Fig. 3.20). The edge type is a concatenation of the left and right vertex types. Vertex types are computed from the strength of edges emanating from a vertex. Vertical edges are handled in the same way, exploiting the obvious symmetries with the horizontal case. Besides the central edge e, the left vertex is the end point for three other possible edges. Classifying these possible edges into "edge" and "no-edge" provides the underpinnings for the vertex types in Fig. 3.21.

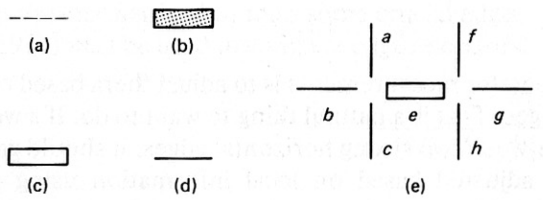


Fig. 3.20 Edge notation. (a) Edge position with no edge. (b) Edge position with edge. (c) Edge to be updated. (d) Edge of unknown strength. (e) Configuration of edges around a central edge e.

To compute vertex type, choose the maximum confidence vertex, i.e., the vertex is type j where j maximizes conf (j)

and

$$\begin{aligned}\text{conf}(0) &= (m-a)(m-b)(m-c)\\ \text{conf}(1) &= a(m-b)(m-c)\\ \text{conf}(2) &= ab(m-c)\\ \text{conf}(3) &= abc\end{aligned}$$

where

$m = \max(a, b, c, q)$
q is a constant (0.1 is about right)

and a, b, and c are the normalized gradient magnitudes for the three edges. Without loss of generality, $a \geqslant b \geqslant c$. The parameter m adjusts the vertex classification so that it is relative to the local maximum. Thus $(a, b, c) = (0.25, 0.01, 0.01)$ is a type 1 vertex. The parameter q forces weak vertices to type zero [e.g., (0.01, 0.001, 0.001) is type zero].

Once the vertex type has been computed, the edge type is simple. It is merely the concatenation of the two vertex types. That is, the edge type is (ij), where i and j are the vertex types. (From symmetry, only consider $i \geqslant j$.)

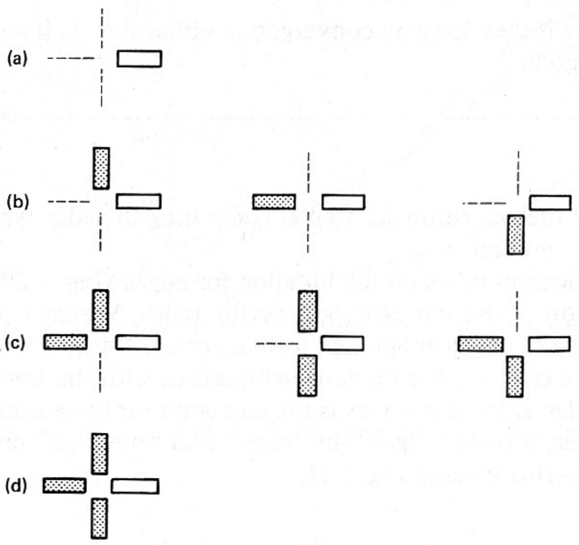


Fig. 3.21 Classification of vertex type of left-hand endpoint of edge e, Fig. 3.20.

Decisions in the second step of modifying edge confidence based on edge type appear in Table 3.1. The updating formula is:

increment: $C^{k+1}(e) = \min(1, C^k(e) + \delta)$
decrement: $C^{k+1}(e) = \max(0, C^k(e) - \delta)$
leave as is: $C^{k+1}(e) = C^k(e)$

where δ is a constant (values from 0.1 to 0.3 are appropriate). The result of using the relaxation scheme is shown in Fig. 3.22. The figures on the left-hand side show

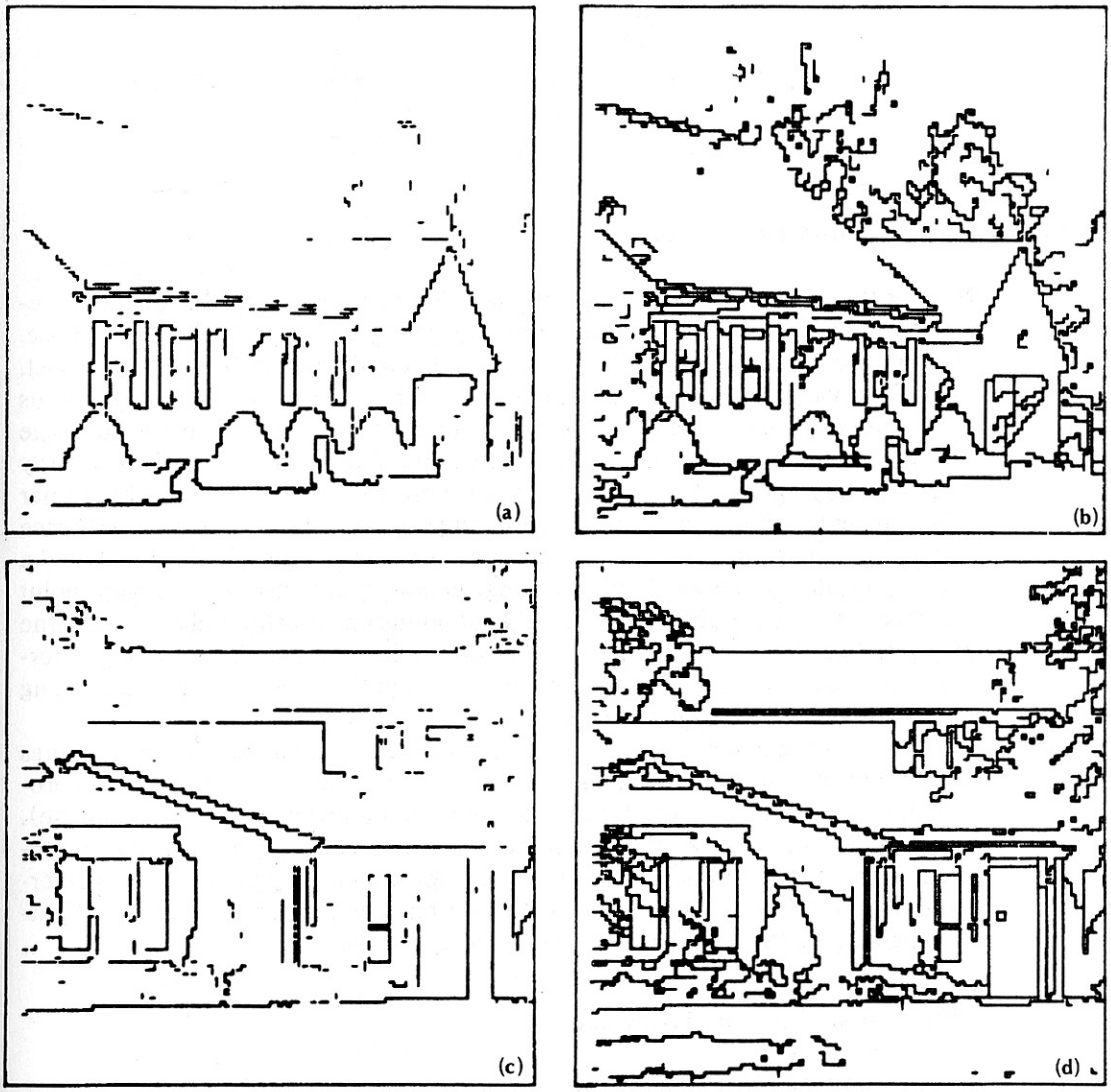


Fig. 3.22 Edge relaxation results. (a) Raw edge data. Edge strengths have been thresholded at 0.25 for display purposes only. (b) Results after five iterations of relaxation applied to (a). (c) Different version of (a). Edge strengths have been thresholded at 0.25 for display purposes only. (d) Results after five iterations of relaxation applied to (c).

the edges with normalized magnitudes greater than 0.25. Weak edges cause many gaps in the boundaries. The figures on the right side show the results of five iterations of edge relaxation. Here the confidence of the weak edges has been increased owing to the proximity of other edges, using the rules in Table 3.1.

Table 3.1

Decrement	*Increment*	*Leave as is*
0-0	*1-1*	*0-1*
0-2	*1-2*	*2-2*
0-3	*1-3*	*2-3*
		3-3

3.4 RANGE INFORMATION FROM GEOMETRY

Neither the perspective or orthogonal projection operations, which take the three-dimensional world to a two-dimensional image, is invertible in the usual sense. Since projection maps an infinite line onto a point in the image, information is lost. For a fixed viewpoint and direction, infinitely many continuous and discontinuous three-dimensional configurations of points could project on our retina in an image of, say, our grandmother. Simple cases are grandmothers of various sizes cleverly placed at varying distances so as to project onto the same area. An astronomer might imagine millions of points distributed perhaps through light-years of space which happen to line up into a "grandmother constellation." All that can be mathematically guaranteed by imaging geometry is that the image point corresponds to one of the infinite number of points on that three-dimensional line of sight. The "inverse perspective" transformation (Appendix 1) simply determines the equation of the infinite line of sight from the parameters of the imaging process modeled as a point projection.

However, a line and a plane not including it intersect in just one point. Lines of sight are easy to compute, and so it is possible to tell where any image point projects on to any known plane (the supporting ground or table plane is a favorite). Similarly, if two images from different viewpoints can be placed in correspondence, the intersection of the lines of sight from two matching image points determines a point in three-space. These simple observations are the basis of light-striping ranging (Section 2.3.3) and are important in stereo imaging.

3.4.1. Stereo Vision and Triangulation

One of the first ideas that occurs to one who wants to do three-dimensional sensing is the biologically motivated one of stereo vision. Two cameras, or one camera from two positions, can give relative depth or absolute three-dimensional location, depending on the elaboration of the processing and measurement. There has been the edges with normalized magnitudes greater than 0.25. Weak edges cause many gaps in the boundaries. The figures on the right side show the results of five iterations of edge relaxation. Here the confidence of the weak edges has been increased owing to the proximity of other edges, using the rules in Table 3.1.

Table 3.1

Decrement	*Increment*	*Leave as is*
0-0	*1-1*	*0-1*
0-2	*1-2*	*2-2*
0-3	*1-3*	*2-3*
		3-3

3.4 RANGE INFORMATION FROM GEOMETRY

Neither the perspective or orthogonal projection operations, which take the three-dimensional world to a two-dimensional image, is invertible in the usual sense. Since projection maps an infinite line onto a point in the image, information is lost. For a fixed viewpoint and direction, infinitely many continuous and discontinuous three-dimensional configurations of points could project on our retina in an image of, say, our grandmother. Simple cases are grandmothers of various sizes cleverly placed at varying distances so as to project onto the same area. An astronomer might imagine millions of points distributed perhaps through light-years of space which happen to line up into a "grandmother constellation." All that can be mathematically guaranteed by imaging geometry is that the image point corresponds to one of the infinite number of points on that three-dimensional line of sight. The "inverse perspective" transformation (Appendix 1) simply determines the equation of the infinite line of sight from the parameters of the imaging process modeled as a point projection.

However, a line and a plane not including it intersect in just one point. Lines of sight are easy to compute, and so it is possible to tell where any image point projects on to any known plane (the supporting ground or table plane is a favorite). Similarly, if two images from different viewpoints can be placed in correspondence, the intersection of the lines of sight from two matching image points determines a point in three-space. These simple observations are the basis of light-striping ranging (Section 2.3.3) and are important in stereo imaging.

3.4.1. Stereo Vision and Triangulation

One of the first ideas that occurs to one who wants to do three-dimensional sensing is the biologically motivated one of stereo vision. Two cameras, or one camera from two positions, can give relative depth or absolute three-dimensional location, depending on the elaboration of the processing and measurement. There has been considerable effort in this direction [Moravec 1977; Quam and Hannah 1974; Binford 1971; Turner 1974; Shapira 1974]. The technique is conceptually simple:

1. Take two images separated by a baseline.
2. Identify points between the two images.
3. Use the inverse perspective transform (Appendix 1) or simple triangulation (Section 2.2.2) to derive the two lines on which the world point lies.
4. Intersect the lines.

The resulting point is in three-dimensional world coordinates.

The hardest part of this method is step 2, that of identifying corresponding points in the two images. One way of doing this is to use correlation, or template matching, as described in Section 3.2.1. The idea is to take a patch of one image and match it against the other image, finding the place of best match in the second image, and assigning a related "disparity" (the amount the patch has been displaced) to the patch.

Correlation is a relatively expensive operation, its naive implementation requiring $0(n^2m^2)$ multiplications and additions for an $m \times m$ patch and $n \times n$ image. This requirement can be drastically improved by capitalizing on the idea of variable resolution; the improved technique is described in Section 3.7.2.

Efficient correlation is of technological concern, but even if it were free and instantaneous, it would still be inadequate. The basic problems with correlation in stereo imaging have to do with the fact that things can look significantly different from different points of view. It is possible for the two stereo views to be sufficiently different that corresponding areas may not be matched correctly. Worse, in scenes with much obscuration, very important features of the scene may be present in only one view. This problem is alleviated by decreasing the baseline, but of course then the accuracy of depth determinations suffers; at a baseline length of zero there is no problem, but no stereo either. One solution is to identify world features, not image appearance, in the two views, and match those (the nose of a person, the corner of a cube). However, if three-dimensional information is sought as a help in perception, it is unreasonable to have to do perception first in order to do stereo.

3.4.2 A Relaxation Algorithm for Stereo

Human *stereopsis*, or fusing the inputs from the eyes into a stereo image, does not necessarily involve being aware of features to match in either view. Most human beings can fuse quite efficiently stereo pairs which individually consist of randomly placed dots, and thus can perceive three-dimensional shapes without recognizing monocular clues in either image. For example, consider the stereo pair of Fig. 3.23. In either frame by itself, nothing but a randomly speckled rectangle can be perceived. All the stereo information is present in the relative displacement of dots in the two rectangles. To make the right-hand member of the stereo pair, a patch of

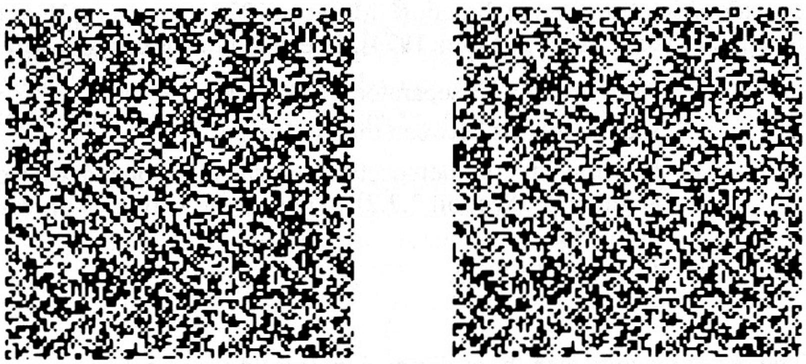


Fig. 3.23 A random-dot stereogram.

the randomly placed dots of the left-hand image is displaced sideways. The dots which are thus covered are lost, and the space left by displacing the patch is filled in with random dots.

Interestingly enough, a very simple algorithm [Marr and Poggio 1976] can be formulated that computes disparity from random dot stereograms. First consider the simpler problem of matching one-dimensional images of four points as depicted in Fig. 3.24. Although only one depth plane allows all four points to be placed in correspondence, lesser numbers of points can be matched in other planes.

The crux of the algorithm is the rules, which help determine, on a local basis, the appropriateness of a match. Two rules arise from the observation that most images are of opaque objects with smooth surfaces and depth discontinuities only at object boundaries:

1. Each point in an image may have only one depth value.
2. A point is almost sure to have a depth value near the values of its neighbors.

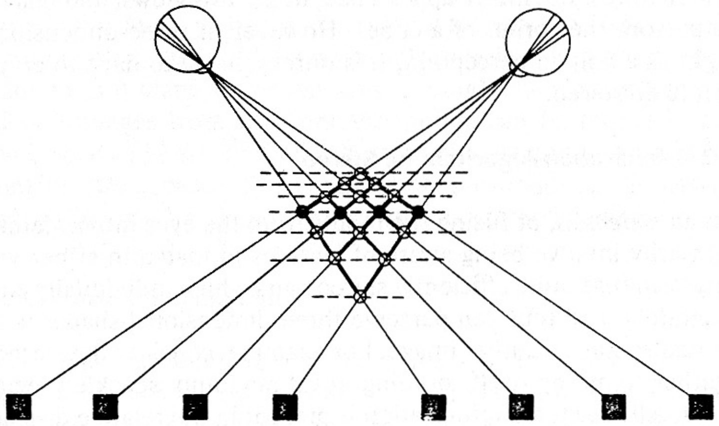


Fig. 3.24 The stereo matching problem.

Figure 3.24 can be viewed as a binary network where each possible match is represented by a binary state. Matches have value 1 and nonmatches value 0. Figure 3.25 shows an expanded version of Fig. 3.24. The connections of alternative matches for a point inhibit each other and connections between matches of equal depth reinforce each other. To extend this idea to two dimensions, use parallel arrays for different values of y where equal depth matches have reinforcing connections. Thus the extended array is modeled as the matrix $C(x, y, d)$ where the point x, y, d corresponds to a particular match between a point (x_1, y_1) in the right image and a point (x_2, y_2) in the left image. The stereopsis algorithm produces a series of matrices C_n which converges to the correct solution for most cases. The initial matrix $C_0(x, y, d)$ has values of one where x, y, d correspond to a match in the original data and has values of zero or otherwise.

Algorithm 3.2 [Marr and Poggio 1976]

Until C satisfies some convergence criterion, do

$$C_{n+1}(x, y, d) = \left\{ \sum_{x',y',d' \in S} C_n(x', y', d') - \sum_{x',y',d' \in \theta} C_n(x', y', d') + C_0(x, y, d) \right\} \tag{3.34}$$

where the term in braces is handled as follows:

$$\{t\} = \begin{cases} 1 & \text{if } t > T \\ 0 & \text{otherwise} \end{cases}$$

S = set of points x', y', d' such that $|\mathbf{x} - \mathbf{x}'| \leqslant 1$ and $d = d'$

θ = set of points x', y', d' such that $|\mathbf{x} - \mathbf{x}'| \leqslant 1$ and $|d - d'| = 1$

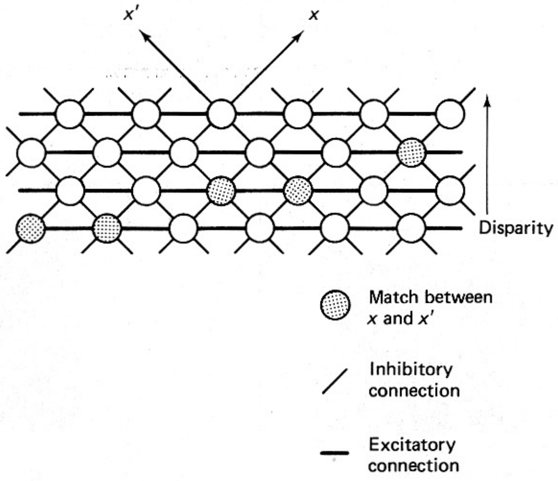


Fig. 3.25 Extension of stereo matching.

One convergence criterion is that the number of points modified on an iteration must be less than some threshold T. Fig. 3.26 shows the results of this computation; the disparity is encoded as a gray level and displayed as an image for different values of n.

A more general version of this algorithm matches image features such as edges rather than points (in the random-dot stereogram, the only features are

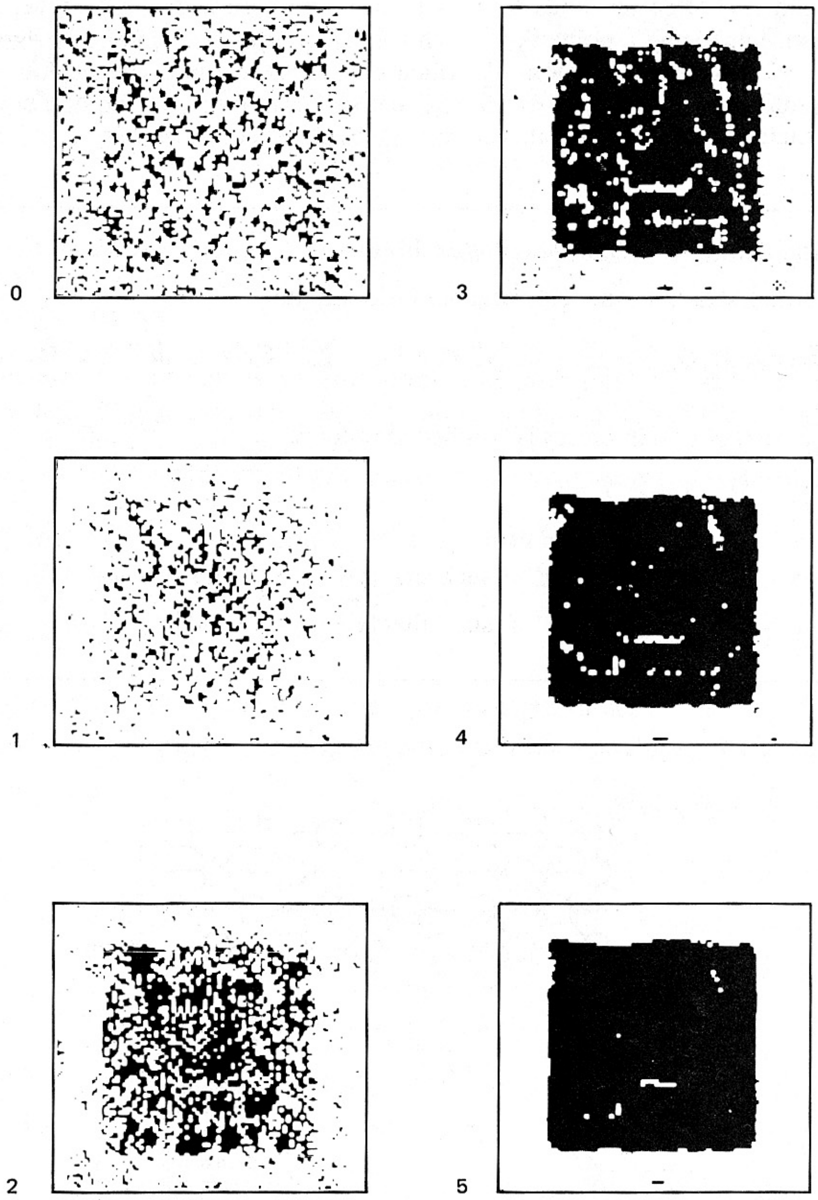


Fig. 3.26 The results of relaxation computations for stereo.

points), but the principles are the same. The extraction of features more complicated than edges or points is itself a thorny problem and the subject of Part II. It should be mentioned that Marr and Poggio have refined their stereopsis algorithm to agree better with psychological data [Marr and Poggio 1977].

3.5 SURFACE ORIENTATION FROM REFLECTANCE MODELS

The ordinary visual world is mostly composed of opaque three-dimensional objects. The intensity (gray level) of a pixel in a digital image is produced by the light reflected by a small area of surface near the corresponding point on the object.

It is easiest to get consistent shape (orientation) information from an image if the lighting and surface reflectance do not change from one scene location to another. Analytically, it is possible to treat such lighting as uniform illumination, a point source at infinity, or an infinite linear source. Practically, the human shape-from-shading transform is relatively robust. Of course, the perception of shape may be manipulated by changing the surface shading in calculated ways. In part, cosmetics work by changing the reflectivity properties of the skin and misdirecting our human shape-from-shading algorithms.

The recovery transformation to obtain information about surface orientation is possible if some information about the light source and the object's reflectivity is known. General algorithms to obtain and quantify this information are complicated but practical simplifications can be made [Horn 1975; Woodham 1978; Ikeuchi 1980]. The main complicating factor is that even with mathematically tractable object surface properties, a single image intensity does not uniquely define the surface orientation. We shall study two ways of overcoming this difficulty. The first algorithm uses intensity images as input and determines the surface orientation by using multiple light source positions to remove ambiguity in surface orientation. The second algorithm uses a single source but exploits constraints between neighboring surface elements. Such an algorithm assigns initial ranges of orientations to surface elements (actually to their corresponding image pixels) on the basis of intensity. The neighboring orientations are "relaxed" against each other until each converges to a unique orientation (Section 3.5.4).

3.5.1 Reflectivity Functions

For all these derivations, consider a distant point source of light impinging on a small patch of surface; several angles from this situation are important (Fig. 3.27).

A surface's reflectance is the fraction of a given incident energy flux (irradiance) it reflects in any given direction. Formally, the *reflectivity function* is defined as $r = \frac{dL}{dE}$, where L is exitant radiance and E is incident flux. In general, for anisotropic reflecting surfaces, the reflectivity function (hence L) is a function of all three angles i, e, and g. The quantity of interest to us is image irradiance, which is proportional to scene radiance, given by $L = \int r\, dE$. In general, the evaluation of this integral can be quite complicated, and the reader is referred to [Horn and

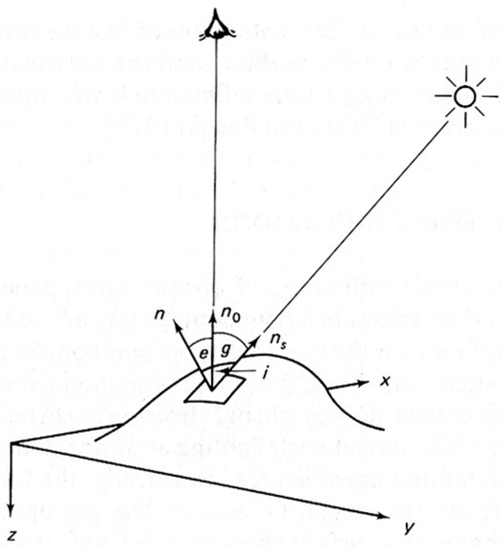


Fig. 3.27 Important reflectance angles: *i*, incidence; *e*, emittance; *g*, phase.

Sjoberg 1978] for a more detailed study. For our purposes we consider surfaces with simple reflectivity functions.

Lambertian surfaces, those with an ideal matte finish, have a very simple reflectivity function which is proportional only to the cosine of the incident angle. These surfaces have the property that under uniform or collimated illumination they look equally bright from any direction. This is because the amount of light reflected from a unit area goes down as the cosine of the viewing angle, but the amount of area seen in any solid angle goes up as the reciprocal of the cosine of the viewing angle. Thus the perceived intensity of a surface element is constant with respect to viewer position. Other surfaces with simple reflectivity functions are "dusty" and "specular" surfaces. An example of a dusty surface is the lunar surface, which reflects in all directions equally. Specular (purely mirror-like) surfaces such as polished metal reflect only at the angle of reflection = angle of incidence, and in a direction such that the incidence, normal, and emittance vectors are coplanar.

Most smooth things have a specular component to their reflection, but in general some light is reflected at all angles in decreasing amounts from the specular angle. One way to achieve this effect is to use the cosine of the angle between the predicted specular angle and the viewing angle, which is given by C where

$$C = 2\cos(i)\cos(e) - \cos(g)$$

This quantity is unity in the pure specular direction and falls off to zero at $\frac{\pi}{2}$ radians away from it. Convincing specular contributions of greater or less sharpness are produced by taking powers of C. A simple radiance formula that allows the simulation of both matte and specular effects is

$$L(i, e, g) = s(C)^n + (1-s)\cos(i) \tag{3.35}$$

Here s varies between 0 and 1 and determines the fraction of specularly reflected light; n determines the sharpness of specularity peaks. As n increases, the specular peak gets sharper and sharper. Computer graphics research is constantly extending the frontiers of realistic and detailed reflectance, refractance, and illumination calculations [Blinn 1978; Phong 1975; Whitted 1980].

3.5.2 Surface Gradient

The reflectance functions described above are defined in terms of angles measured with respect to a local coordinate frame. For our development, it is more useful to relate the reflectivity function to surface gradients measured with respect to a viewer-oriented coordinate frame.

The concept of *gradient space*, which is defined in a viewer-oriented frame [Horn 1975], is extremely useful in understanding the recovery transformation algorithm for the surface normal. This gradient refers to the orientation of a physical surface, *not* to local intensities. It must not be confused with the *intensity* gradients discussed in Section 3.3 and elsewhere in this book.

Gradient space is a two-dimensional space of slants of scene surfaces. It measures a basic "intrinsic" (three-dimensional) property of surfaces. Consider the point-projection imaging geometry of Fig. 2.2, with the viewpoint at infinity (far from the scene relative to the scene dimensions). The image projection is then orthographic, nor perspective.

The surface gradient is defined for a surface expressed as $-z = f(x, y)$. The gradient is a vector (p, q), where

$$p = \frac{\partial(-z)}{\partial x} \qquad (3.36)$$

$$q = \frac{\partial(-z)}{\partial y}$$

Any plane in the image (such as the face plane of a polyhedral face) may be expressed in terms of its gradient. The general plane equation is

$$Ax + By + Cz + D = 0 \qquad (3.37)$$

Thus

$$-z = \frac{A}{C}x + \frac{B}{C}y + \frac{D}{C} \qquad (3.38)$$

and from (3.36) the gradient may be related to the plane equation:

$$-z = px + qy + K \qquad (3.39)$$

Gradient space is thus the two-dimensional space of (p, q) vectors. The p and q axes are often considered to be superimposed on the x and y image plane coordinate axes. Then the (p, q) vector is "in the direction" of the surface slant of imaged surfaces. Any plane perpendicular to the viewing direction has a (p, q) vector of (0,0). Vectors on the q (or y) axis correspond to planes tilted about the x axis in an "upward" or "downward" ("$y$ward") direction (like the tilt of a dressing table mirror). The direction arctan (q/p) is the direction of fastest change of surface depth $(-z)$ as x and y change. $(p^2 + q^2)^{1/2}$ is the rate of this change. For instance, a vertical plane "edge on" to the viewer has a (p, q) of $(I\,\infty, 0)$.

The *reflectance map* $R(p, q)$ represents this variation of perceived brightness with surface orientation. $R(p, q)$ gives scene radiance (Section 2.2.3) as a function of surface gradient (in our usual viewer-centered coordinate system). (Figure 3.27 showed the situation and defined some important angles.) $R(p, q)$ is usually shown as contours of constant scene radiance (Fig. 3.28). The following are a few useful cases.

In the case of a Lambertian surface with the source in the direction of the viewer $(i = e)$, the gradient space image looks like Fig. 3.28. Remember that Lambertian surfaces have constant intensity for constant illumination angle; these constant angles occur on the concentric circles of Fig. 3.28, since the direction of tilt does not affect the magnitude of the angle. The brightest surfaces are those illuminated from a normal direction—they are facing the viewer and so their gradients are $(0, 0)$.

Working this out from first principles, the incident angle and emittance angle are the same in this case, since the light is near the viewer. Both are the angle between the surface normal and the view vector. Looking at the $x-y$ plane means a vector to the light source of $(0, 0, -1)$, and at a gradient point (p, q), the surface normal is $(p, q, -1)$. Also,

$$R = r_o \cos i \tag{3.40}$$

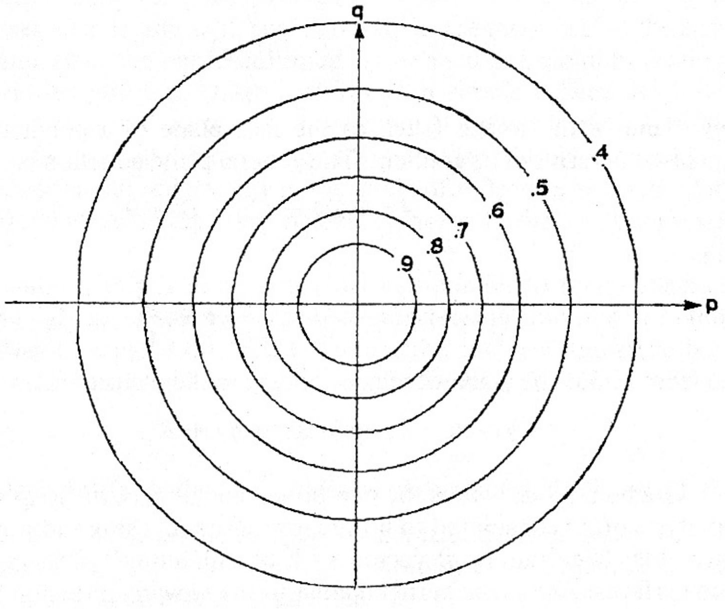


Fig. 3.28 Contours of constant radiance in gradient space for Lambertian surfaces; single light source near the viewpoint.

where r_o is a proportionality constant, and we conventionally use R to denote radiance in a viewer-centered frame. Let $\mathbf{n}_s$ and $\mathbf{n}$ be unit vectors in the source and surface normal directions. Since $\cos i = \mathbf{n}_s \cdot \mathbf{n}$

$$R = \frac{r_0}{(1 + p^2 + q^2)^{1/2}} \tag{3.41}$$

Thus cos (i) determines the image brightness, and so a plot of it is the gradient space image (Figs. 3.29 and 3.30).

For a more general light position, the mathematics is the same; if the light source is in the $(p_s, q_s, -1)$ direction, take the dot product of this direction and the surface normal.

$$R = r_o \mathbf{n} \cdot \mathbf{n}_s \tag{3.42}$$

Or, in other words,

$$R = \frac{r_o (p_s p + q_s q + 1)}{[(1 + p^2 + q^2)(1 + p_s^2 + q_s^2)]^{1/2}}$$

The phase angle g is constant throughout gradient space with orthographic projection (viewer distant from scene) and light source distant from scene.

Setting R constant to obtain contour lines gives a second-order equation, producing conic sections. In fact, the contours are produced by a set of cones of varying angles, whose axis is in the direction of the light source, intersecting a plane at unit distance from the origin. The resulting contours appear in Fig. 3.29. Here the dark line is the terminator, and represents all those planes that are edge-

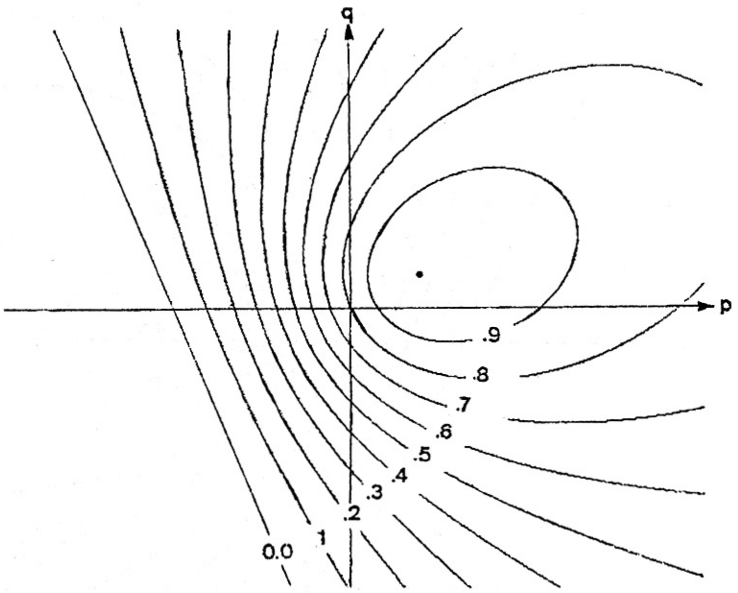


Fig. 3.29 Contours of constant radiance in gradient space. Lambertian surfaces; light not near viewpoint.

on to the light source; gradients on the back side of the terminator represent self-shadowed surfaces (facing away from the light). One intensity determines a contour and so gives a cone whose tangent planes all have that emittance. For a surface with specularity, contours of constant $I(i, e, g)$ could appear as in Fig. 3.30.

The point of specularity is between the matte component maximum brightness gradient and the origin. The brightest matte surface normal points at the light source and the origin points at the viewer. Pure specular reflection can occur if the vector tilts halfway toward the viewer maintaining the direction of tilt. Thus its gradient is on a line between the origin and the light-source direction gradient point.

3.5.3 Photometric Stereo

The reflectance equation (3.42) constrains the possible surface orientation to a locus on the reflectance map. Multiple light-source positions can determine the orientation uniquely [Woodham 1978]. Each separate light position gives a separate value for the intensity (proportional to radiance) at each point $f(\mathbf{x})$. If the surface reflectance r_o is unknown, three equations are needed to determine the reflectance together with the unit normal $\boldsymbol{n}$. If each source position vector is denoted by n_k, $k = 1, \ldots, 3$, the following equations result:

$$I_k(x, y) = r_o(\mathbf{n}_k \cdot \mathbf{n}), \qquad k = 1, \ldots, 3 \tag{3.43}$$

where I is normalized intensity. In matrix form

$$\mathbf{I} = r_o N \mathbf{n} \tag{3.44}$$

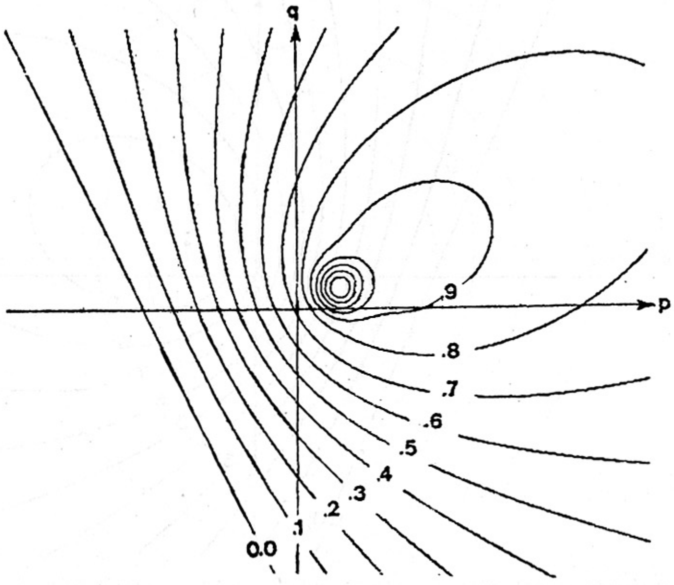


Fig. 3.30 Contours of constant radiance for a specular/matte surface.

where

$$\mathbf{I} = [I_1(x, y), I_2(x, y), I_3(x, y)]^T,$$

and

$$N = \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix} \tag{3.45}$$

and $I = fc$ where c is the appropriate normalization constant. If c is not known, it can be regarded as being part of r_o without affecting the normal direction calculation. As long as the three source positions $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{n}_3$ are not coplanar, the matrix N will have an inverse. Then solve for r_o and $\mathbf{n}$ by using (3.44), first using the fact that $\mathbf{n}$ is a unit vector to derive

$$r_0 = |N^{-1}\mathbf{I}| \tag{3.46}$$

and then solving for $\boldsymbol{n}$ to obtain

$$\mathbf{n} = \frac{1}{r_o} N^{-1}\mathbf{I} \tag{3.47}$$

Examples of a particular solution are shown in Fig. 3.31. Of course, a prerequisite for using this method is that the surface point not be in shadows for any of the sources.

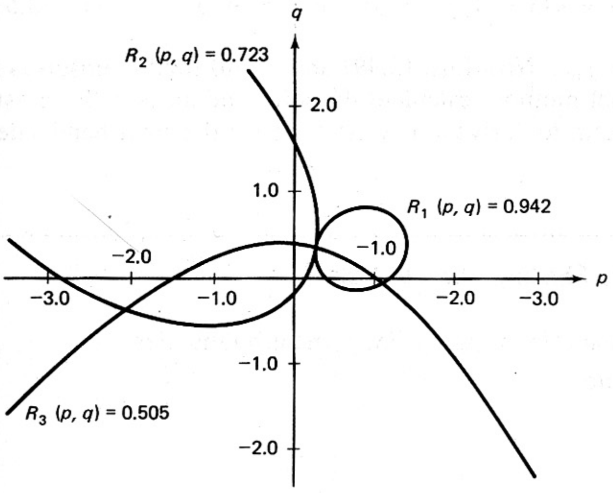


Fig. 3.31 A particular solution for photometric stereo.

3.5.4 Shape from Shading by Relaxation

Combining local information allows improved estimates for edges (Section 3.3.5) and for disparity (Section 3.4.2). In a similar manner local information can help in computing surface orientation [Ikeuchi 1980]. Basically, the reflectance equation provides one constraint on the surface orientation and another is provided by the heuristic requirement that the surface be smooth.

Suppose there is an estimate of the surface normal at a point $(p(x, y), q(x, y))$. If the normal is not accurate, the reflectivity equation $I(x, y) = R(p, q)$ will not hold. Thus it seems reasonable to seek p and q that minimize $(I - R)^2$. The other requirement is that $p(x, y)$ and $q(x, y)$ be smooth, and this can be measured by their Laplacians $\nabla^2 p$ and $\nabla^2 q$. For a smooth curve both of these terms should be small. The goal is to minimize the error at a point,

$$E(x, y) = [I(x, y) - R(p, q)]^2 + \lambda[(\nabla^2 p)^2 + (\nabla^2 q)^2] \tag{3.48}$$

where the Lagrange multiplier λ [Russell 1976] incorporates the smoothness constraint. Differentiating $E(x, y)$ with respect to p and q and approximating derivatives numerically gives the following equations for $p(x, y)$ and $q(x, y)$:

$$p(x, y) = p_{\text{av}}(x, y) + T(x, y, p, q)\frac{\partial R}{\partial q} \tag{3.49}$$

$$q(x, y) = q_{\text{av}}(x, y) + T(x, y, p, q)\frac{\partial R}{\partial q} \tag{3.50}$$

where

$$T(x, y, p, q) = (1/\lambda)[I(x, y) - R(p, q)]$$

using

$$p_{\text{av}}(x, y) = \frac{1}{4}[p(x + 1, y) + p(x - 1, y) + p(x, y + 1) + p(x,y - 1)] \tag{3.51}$$

and a similar expression for q_{av}. Now Eqs. (3.49) and (3.50) lend themselves to solution by the Gauss-Seidel method: calculate the left-hand sides with an estimate for p and q and use them to derive a new estimate for the right-hand sides. More formally,

---

Algorithm 3.3: Shape from Shading [Ikeuchi 1980].

Step 0. k = 0. Pick an initial $p^0(x, y)$ and $q^0(x, y)$ near boundaries.

Step 1. k = k + 1; compute

$$p^k = p_{\text{av}}^{k-1} + T\frac{\partial R}{\partial p}$$

$$q^k = q_{\text{av}}^{k-1} + T\frac{\partial R}{\partial p}$$

Step 2. If the sum of all the E's is sufficiently small, stop. Else, go to step 1.

A loose end in this algorithm is that boundary conditions must be specified. These are values of p and q determined a priori that remain constant throughout each iteration. The simplest place to specify a surface gradient is at an occluding contour (see Fig. 3.32) where the gradient is nearly 90° to the line of sight. Unfortunately, p and q are infinite at these points. Ikeuchi's elegant solution to this is to use a different coordinate system for gradient space, that of a Gaussian sphere (Appendix 1). In this system, the surface normal is described relative to where it intersects the sphere if the tail of the normal is at the sphere's origin. This is the point at which a plane perpendicular to the normal would touch the sphere if translated toward it (Fig. 3.32b).

In this system the radiance may be described in terms of the spherical coordinates θ, ϕ. For a Lambertian surface

$$R(\theta, \phi) = \cos\theta \ \cos\theta_s + \sin\theta \sin\theta_s \ \cos(\phi - \phi_s) \tag{3.52}$$

At an occluding contour $\phi = \pi/2$ and θ is given by $\tan^{-1}(\partial y / \partial x)$, where the derivatives are calculated at the occluding contour (Fig. 3.32c).

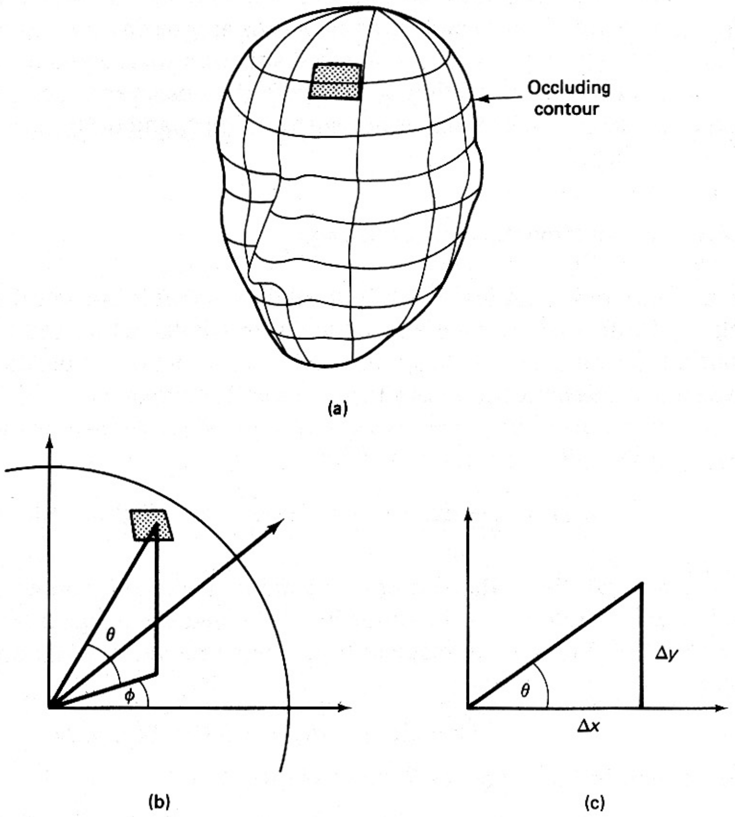


Fig. 3.32 (a) Occluding contour. (b) Gaussian sphere. (c) Calculating θ from occluding contour.

To use the (θ, ϕ) formulation instead of the (p, q) formulation is an easy matter. Simply substitute θ for p and ϕ for q in all instances of the formula in Algorithm 3.3.

3.6 OPTICAL FLOW

Much of the work on computer analysis of visual motion assumes a stationary observer and a stationary background. In contrast, biological systems typically move relatively continuously through the world, and the image projected on their retinas varies essentially continuously while they move. Human beings perceive smooth continuous motion as such.

Although biological visual systems are discrete, this quantization is so fine that it is capable of producing essentially continuous outputs. These outputs can mirror the continuous flow of the imaged world across the retina. Such continuous information is called *optical flow.* Postulating optical flow as an input to a perceptual system leads to interesting methods of motion perception.

The optical flow, or instantaneous velocity field, assigns to every point on the visual field a two-dimensional "retinal velocity" at which it is moving across the visual field. This section describes how approximations to instantaneous flow may be computed from the usual input situation in a sequence of discrete images. Methods of using optical flow to compute the observer's motion, a relative depth map, surface normals of his or her surroundings, and other useful information are given in Chapter 7.

3.6.1 The Fundamental Flow Constraint

One of the important features of optical flow is that it can be calculated simply, using local information. One way of doing this is to model the motion image by a continuous variation of image intensity as a function of position and time, then expand the intensity function $f(x, y, t)$ in a Taylor series.

$$\begin{aligned} &f(x + dx,\ y + dy,\ t + dt) = \\ &f(x, y, t) + \frac{\partial f}{\partial x}\,dx + \frac{\partial f}{\partial y}\,dy + \frac{\partial f}{\partial t}\,dt + \text{higher-order terms} \end{aligned} \tag{3.53}$$

As usual, the higher-order terms are henceforth ignored. The crucial observation to be exploited is the following: If indeed the image at some time $t + dt$ is the result of the original image at time t being moved translationally by dx and dy, then in fact

$$f(x + dx,\ y + dy,\ t + dt) = f(x, y, t) \tag{3.54}$$

Consequently, from Eqs. (3.53) and (3.54),

$$-\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} \tag{3.55}$$

To use the (θ, ϕ) formulation instead of the (p, q) formulation is an easy matter. Simply substitute θ for p and ϕ for q in all instances of the formula in Algorithm 3.3.

3.6 OPTICAL FLOW

Much of the work on computer analysis of visual motion assumes a stationary observer and a stationary background. In contrast, biological systems typically move relatively continuously through the world, and the image projected on their retinas varies essentially continuously while they move. Human beings perceive smooth continuous motion as such.

Although biological visual systems are discrete, this quantization is so fine that it is capable of producing essentially continuous outputs. These outputs can mirror the continuous flow of the imaged world across the retina. Such continuous information is called *optical flow*. Postulating optical flow as an input to a perceptual system leads to interesting methods of motion perception.

The optical flow, or instantaneous velocity field, assigns to every point on the visual field a two-dimensional "retinal velocity" at which it is moving across the visual field. This section describes how approximations to instantaneous flow may be computed from the usual input situation in a sequence of discrete images. Methods of using optical flow to compute the observer's motion, a relative depth map, surface normals of his or her surroundings, and other useful information are given in Chapter 7.

3.6.1 The Fundamental Flow Constraint

One of the important features of optical flow is that it can be calculated simply, using local information. One way of doing this is to model the motion image by a continuous variation of image intensity as a function of position and time, then expand the intensity function $f(x, y, t)$ in a Taylor series.

$$f(x + dx,\ y + dy,\ t + dt) = \qquad (3.53)$$
$$f(x, y, t) + \frac{\partial f}{\partial x}\,dx + \frac{\partial f}{\partial y}\,dy + \frac{\partial f}{\partial t}\,dt + \text{higher-order terms}$$

As usual, the higher-order terms are henceforth ignored. The crucial observation to be exploited is the following: If indeed the image at some time $t + dt$ is the result of the original image at time t being moved translationally by dx and dy, then in fact

$$f(x + dx,\ y + dy,\ t + dt) = f(x, y, t) \qquad (3.54)$$

Consequently, from Eqs. (3.53) and (3.54),

$$-\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} \qquad (3.55)$$

Now $\frac{\partial f}{\partial t}$, $\frac{\partial f}{\partial x}$, and $\frac{\partial f}{\partial y}$ are all measurable quantities, and $\frac{dx}{dt}$ and $\frac{dy}{dt}$ are estimates of what we are looking for—the velocity in the *x* and *y* directions. Writing

$$\frac{dx}{dt} = u, \qquad \frac{dy}{dt} = v$$

gives

$$-\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v \tag{3.56}$$

or equivalently,

$$-\frac{\partial f}{\partial t} = \nabla f \cdot \mathbf{u} \tag{3.57}$$

where ∇f is the spatial gradient of the image and $\mathbf{u} = (u, v)$ the velocity.

The implications of (3.57) are interesting. Consider a fixed camera with a scene moving past it. The equations say that the *time* rate of change in intensity of a point in the image is (to first order) explained as the *spatial* rate of change in the intensity of the scene multiplied by the *velocity* that points of the scene move past the camera.

This equation also indicates that the velocity (u, v) must lie on a line perpendicular to the vector (f_x, f_y) where f_x and f_y are the partial derivatives with respect to x and y, respectively (Fig. 3.33). In fact, if the partial derivatives are very accurate the magnitude component of the velocity in the direction (f_x, f_y) is (from 3.57):

$$\frac{-f_t}{[(f_x^2 + f_y^2)]^{1/2}}$$

3.6.2 Calculating Optical Flow by Relaxation

Equation (3.57) constrains the velocity but does not determine it uniquely. The development of Section 3.5.4 motivates the search for a solution that satisfies Eq.

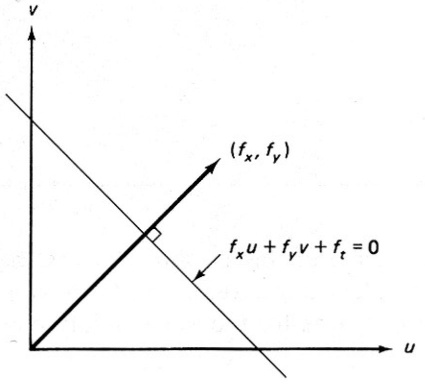


Fig. 3.33 Relation between (u, v) and (f_x, f_y).

(3.57) as closely as possible and also is locally smooth [Horn and Schunck 1980]. In this case as well, the Laplacians of the two velocity components, $\nabla^2 u$ and $\nabla^2 v$, can measure local smoothness.

Again using the method of Lagrange multipliers, minimize the flow error

$$E^2(x, y) = (f_x u + f_y v + f_t)^2 + \lambda^2[(\nabla^2 u)^2 + (\nabla^2 v)^2] \tag{3.58}$$

Differentiating this equation with respect to u and v provides equations for the change in error with respect to u and v, which must be zero for a minimum. Writing $\nabla^2 u$ as $u - u_{\text{av}}$ and $\nabla^2 v$ as $v - v_{\text{av}}$, these equations are

$$(\lambda^2 + f_x^2)u + f_x f_y v = \lambda^2 u_{\text{av}} - f_x f_t \tag{3.59}$$

$$f_x f_y u + (\lambda^2 + f_y^2) v = \lambda^2 v_{\text{av}} - f_y f_t \tag{3.60}$$

These equations may be solved for u and v, yielding

$$u = u_{\text{av}} - f_x \frac{P}{D} \tag{3.61}$$

$$v = v_{\text{av}} - f_y \frac{P}{D} \tag{3.62}$$

where

$$P = f_x u_{\text{av}} + f_y v_{\text{av}} + f_t$$
$$D = \lambda^2 + f_x^2 + f_y^2$$

To turn this into an iterative equation for solving $u(x, y)$ and $v(x, y)$, again use the Gauss-Seidel method.

Algorithm 3.4: Optical Flow [Horn and Schunck 1980].

```
k = 0.
Initialize all $u^k$ and $v^k$ to zero.
Until some error measure is satisfied, do
    $u^k = u_{\text{av}}^{k-1} - f_x \frac{P}{D}$
    $v^k = v_{\text{av}}^{k-1} - f_y \frac{P}{D}$
```

As Horn and Schunck demonstrate, this method derives the flow for two time frames, but it can be improved by using several time frames and using the final solution after one iteration at one time for the initial solution at the following time frame. That is:

Algorithm 3.5: Multiframe Optical Flow.

```
$t = 0$.
Initialize all $u(x, y, 0)$, $v(x, y, 0)$
*for* $t = 1$ *until* maxframes *do*
    $u(x, y, t) = u_{\text{av}}(x, y, t-1) - f_x \frac{P}{D}$
    $v(x, y, t) = v_{\text{av}}(x, y, t-1) - f_y \frac{P}{D}$
```

The results of using synthetic data from a rotating checkered sphere are shown in Fig. 3.34.

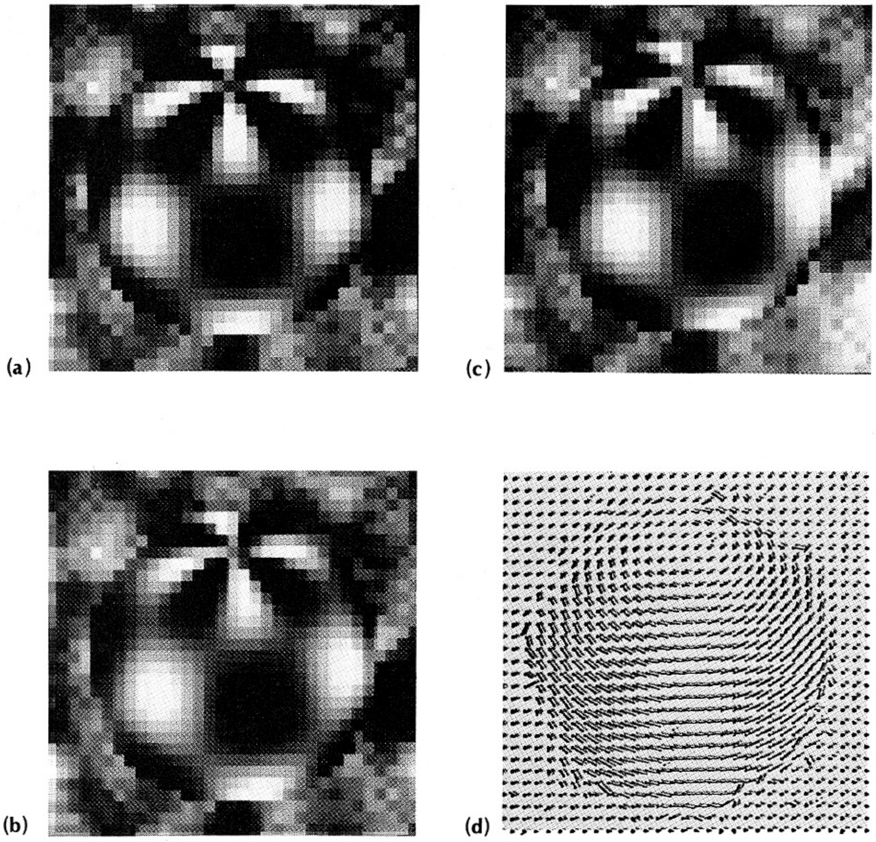


Fig. 3.34 Optical flow results. (a), (b) and (c) are three frames from the rotating sphere, (d) is the derived three-dimensional flow after 32 such time frames.

What is the best spatial resolution for an image? The sampling theorem states that the maximum spatial frequency in the image data must be less than half the sampling frequency in order that the sampled image represent the original unambiguously. However, the sampling theorem is not a good predictor of how easily objects can be recognized by computer programs. Often objects can be more easily recognized in images that have a very low sampling rate. There are two reasons for this. First, the computations are fewer because of the reduction in dimensionality. Second, confusing detail present in the high-resolution versions of the images may not appear at the reduced resolution. But even though some objects are more easily found at low resolutions, usually an object description needs detail only revealed at the higher resolutions. This leads naturally to the notion of a *pyramidal* image data structure in which the search for objects is begun at a low resolution, and refined at ever-increasing resolutions until one reaches the highest resolution of interest. Figure 3.35 shows the correspondence between pixels for the pyramidal structure.

In the next three sections, pyramids are applied to gray-level images and edge images. Pyramids, however, are a very general tool and can be used to represent any image at varying levels of detail.

3.7.1 Gray-level Consolidation

In some applications, redigitizing the image with a different sampling rate is a way to reduce the number of samples. However, most digitizer parameters are difficult to change, so that often computational means of reduction are needed. A straightforward method is to partition the digitized image into nonoverlapping

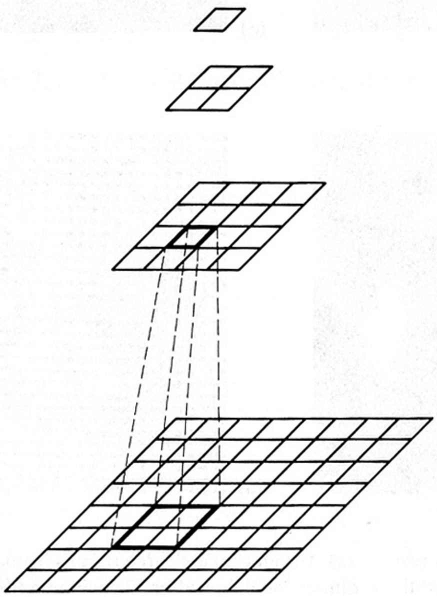


Fig. 3.35 Pyramidal image structure.

neighborhoods of equal size and shape and to replace each of those neighborhoods by the average pixel densities in that neighborhood. This operation is *consolidation*. For an $n \times n$ neighborhood, consolidation is equivalent to averaging the original image over the neighborhood followed by sampling at intervals n units apart.

Consolidation tends to offset the aliasing that would be introduced by sampling the sensed data at a reduced rate. This is due to the effects of the averaging step in the consolidation process. For the one-dimensional case where

$$f'(x) = \frac{1}{2}[f(x) + f(x + \Delta)] \tag{3.63}$$

the corresponding Fourier transform [Steiglitz 1974] is

$$H(u) = \frac{1}{2}\left(1 + e^{-j2\pi\Delta}\right)F(u) \tag{3.64}$$

which has magnitude $|H(u)| = \cos[\pi(u/u_o)]$ and phase $-\pi(u/u_o)$. The sampling frequency $u_o = 1/\Delta$ where Δ is the spacing between samples. Thus the averaging step has the effect of attenuating the higher frequencies of $F(u)$ as shown in Fig. 3.36. Since the higher frequencies are involved in aliasing, attenuating these frequencies reduces the aliasing effects.

3.7.2 Pyramidal Structures in Correlation

With correlation matching, the use of multiple resolution techniques can sometimes provide significant functional and computational advantages [Moravec 1977]. Binary search correlation uses pyramids of the input image and reference

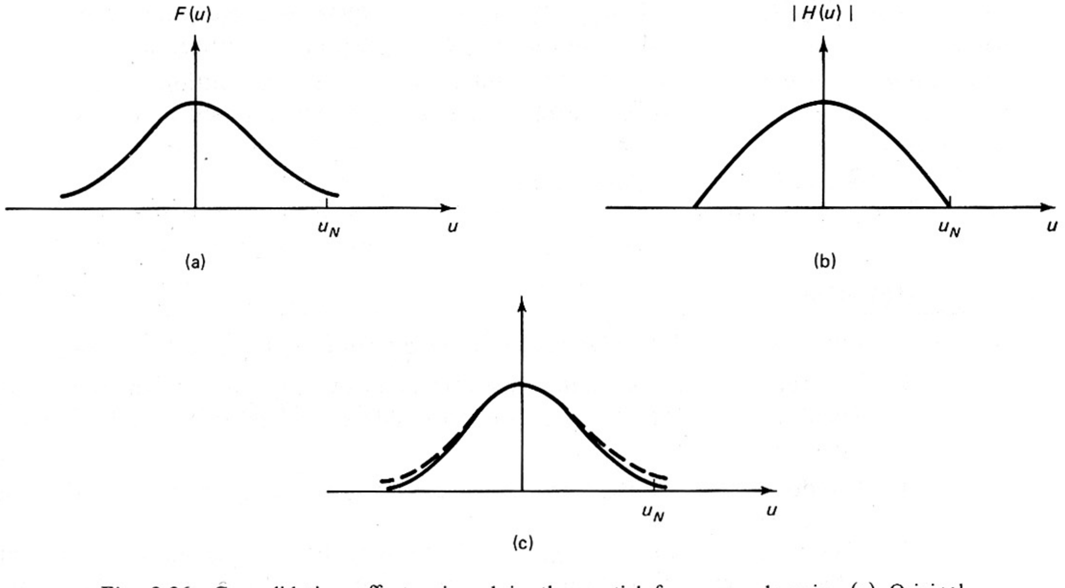


Fig. 3.36 Consolidation effects viewed in the spatial frequency domain. (a) Original transform. (b) Transform of averaging operator. (c) Transform of averaged image.

patterns. The algorithm partakes of the computational efficiency of binary (as opposed to linear) search [Knuth 1973]. Further, the low-resolution correlation operations at high levels in the pyramid ensure that the earlier correlations are on gross image features rather than details.

In binary search correlation a feature to be located is at some unknown location in the input image. The reference version of the feature originates in another image, the reference image. The feature in the reference image is contained in a window of $n \times n$ pixels. The task of the correlator is to find an $n \times n$ window in the input image that best matches the reference image window containing the feature. The details of the correlation processes are given in the following algorithm.

Algorithm 3.6: Binary Search Correlation Control Algorithm

Definitions

OrigReference:	an $N \times N$ image containing a feature centered at (FeatureX, FeatureY).
OrigInput:	an $M \times M$ array in which an instance of the Feature is to be located. For simplicity, assume that it is at the same resolution as OrigReference.
n:	a window size; an $n \times n$ window in OrigReference is large enough to contain the Feature.
Window:	an $n \times n$ array containing a varying-resolution subimage of OrigReference centered on the Feature.
Input:	a $2n \times 2n$ array containing a varying-resolution subimage of OrigInput, centered on the best match for the Feature.
Reference:	a temporary array.

Algorithm

1. Input := Consolidate OrigInput by a factor of $2n/M$ to size $2n \times 2n$.
2. Reference := Consolidate OrigReference by the same factor $2n/M$ to size $2nN/M \times 2nN/M$. This consolidation takes the Feature to a new (Feature$X$, Feature$Y$).
3. Window := $n \times n$ window from Reference centered on the new (Feature$X$, Feature$Y$).
4. Calculate the match metric of the window at the $(n+1)^2$ locations in Input at which it is wholly contained. Say that the best match occurs at (BestMatch$X$, BestMatch$Y$) in Input.

5. Input := $n \times n$ window from Input centered at (BestMatch$X$, BestMatch$Y$), enlarged by a factor of 2.
6. Reference := Reference enlarged by a factor of 2. This takes Feature to a new (Feature$X$, Feature$Y$).
7. *Go to* 3.

Through time, the algorithm uses a reference image for matching that is always centered on the feature to be matched, but that homes in on the feature by being increased in resolution and thus reduced in linear image coverage by a factor of 2 each time. In the input image, a similar homing-in is going on, but the search area is usually twice the linear dimension of the reference window. Further, the center of the search area varies in the input image as the improved resolution refines the point of best match.

Binary search correlation is for matching features with context. The template at low resolution possibly corresponds to much of the area around the feature, while the feature may be so small in the initial consolidated images as to be invisible. The coarse-to-fine strategy is perfect for such conditions, since it allows gross features to be matched first and to guide the later high-resolution search for best match. Such matching with context is less useful for locating several instances of a shape dotted at random around an image.

3.7.3 Pyramidal Structures in Edge Detection

As an example of the use of pyramidal structures in processing, consider the use of such structures in edge detection. This application, after [Tanimoto and Pavlidis 1975], uses two pyramids, one to store the image and another to store the image edges. The idea of the algorithm is that a neighborhood in the low-resolution image where the gray-level values are the same is taken to imply that in fact there is no gray-level change (edge) in the neighborhood. Of course, the low-resolution levels in the pyramid tend to blur the image and thus attenuate the gray-level changes that denote edges. Thus the starting level in the pyramid must be picked judiciously to ensure that the important edges are detected.

Algorithm 3.7: Hierarchical Edge Detection

```
recursive procedure refine $(k, x, y)$
  begin
  if $k <$ MaxLevel then
    for $dx = 0$ until 1 do
    for $dy = 0$ until 1 do
      if EdgeOp $(k, x + dx, y + dy) >$ Threshold$(x)$
      then refine $(k + 1, x + dx, y + dy)$
  end;
```

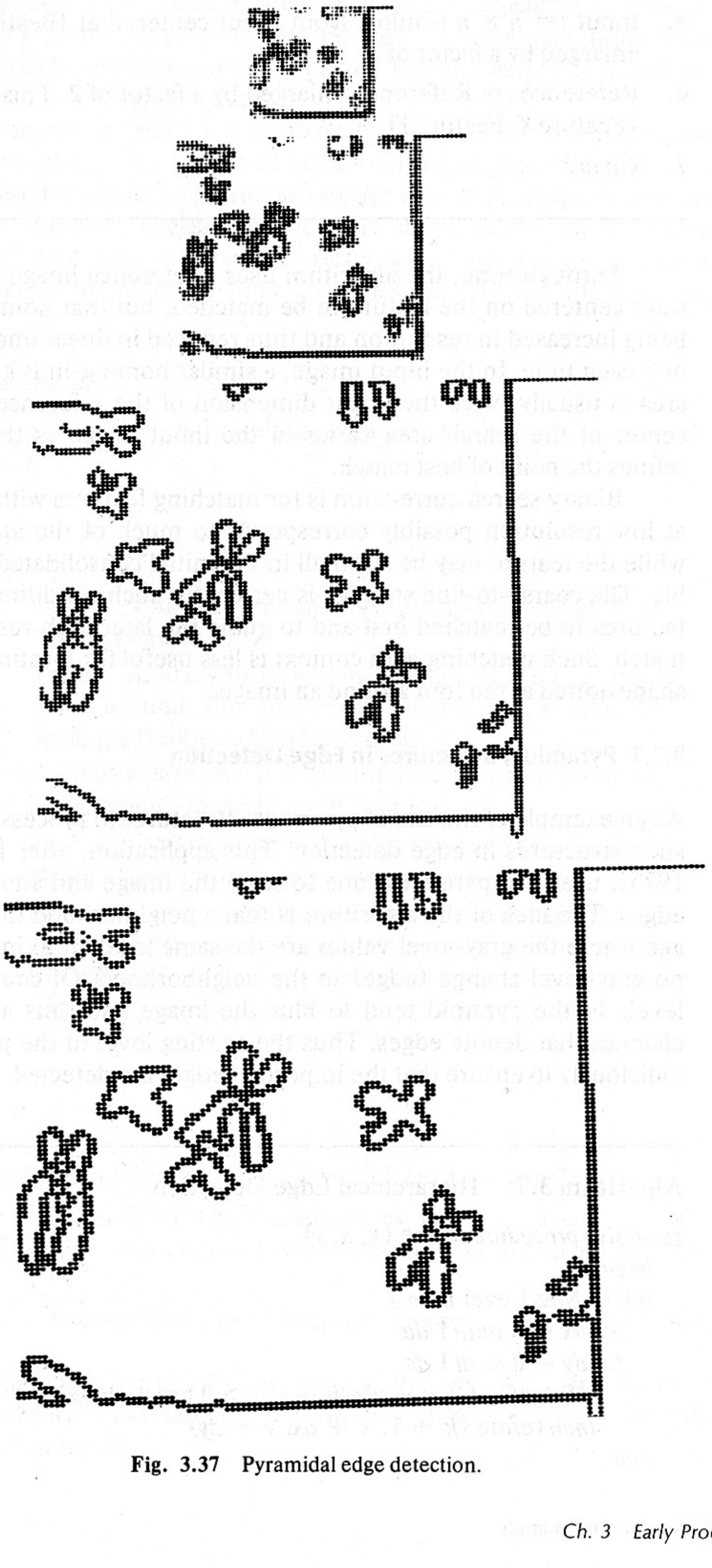


Fig. 3.37 Pyramidal edge detection.

```
procedure FindEdges:
  begin
  comment apply operator to every pixel in the
      starting level s, refining where necessary;
    for x := 0 until $2^S - 1$ do
    for y := 0 until $2^S - 1$ do
      if EdgeOp $(s, x, y) >$ Threshold$(s)$
      then refine $(s, x, y)$;
  end;
```

Figure 3.37 shows Tanimoto's results for a chromosome image. The table inset shows the computational advantage in terms of the calls to the edge operator as a function of the starting level s.

Similar kinds of edge detection strategies based on pyramids have been pursued by [Levine 1978; Hanson and Riseman 1978]. The latter effort is a little different in that processing within the pyramid is bidirectional; information from edges detected at a high-resolution level is projected to low-resolution levels of the pyramid.

EXERCISES

3.1 Derive an analytical expression for the response of the Sobel operator to a vertical step edge as a function of the distance of the edge to the center of the operator.

3.2 Use the formulas of Eqs. (3.31) to derive the digital template function for g_1 in a 5^3 pixel domain.

3.3 Specify a version of Algorithm 3.1 that uses the gradient edge operator instead of the "crack" edge operator.

3.4 In photometric stereo, three or more light source positions are used to determine a surface orientation. The dual of this problem uses surface orientations to determine light source position. What is the usefulness of the latter formulation? In particular, how does it relate to Algorithm 3.3?

3.5 Using any one of Algorithms 3.1 through 3.4 as an example, show how it could be modified to use pyramidal data structures.

3.6 Write a reflectance function to capture the "grazing incidence" phenomenon—surfaces become more mirror-like at small angles of incidence (and reflectance).

3.7 Equations 3.49 and 3.50 were derived by minimizing the local error. Show how these equations are modified when total error [i.e., $\sum_{x,y} E(x, y)$] is minimized.

REFERENCES

ABDOU, I. E. "Quantitative methods of edge detection." USCIPI Report 830, Image Processing Institute, Univ. Southern California, July 1978.

AKATSUKA, T., T. ISOBE, and O. TAKATANI. "Feature extraction of stomach radiograph." *Proc.*, 2nd IJCPR, August 1974, 324-328.

ANDREWS, H. C. and B. R. HUNT. *Digital Image Restoration.* Englewood Cliffs, NJ: Prentice-Hall, 1977.

ATTNEAVE, F. "Some informational aspects of visual perception." *Psychological Review 61*, 1954.

BARROW, H. G. and J. M. TENENBAUM. "Computational Vision." *Proc. IEEE 69*, 5, May 1981, 572-595

BARROW, H. G. and J. M. TENENBAUM. "Recovering intrinsic scene characteristics from images." Technical Note 157, AI Center, SRI International, April 1978.

BINFORD, T. O. "Visual perception by computer." *Proc.*, IEEE Conf. on Systems and Control, Miami, December 1971.

BLINN, J. E. "Computer display of curved surfaces." Ph.D. dissertation, Computer Science Dept., Univ. Utah, 1978.

FREI, W. and C. C. CHEN. "Fast boundary detection: a generalization and a new algorithm." *IEEE Trans. Computers 26*, 2, October 1977, 988-998.

GONZALEZ, R. C. and P. WINTZ. *Digital Image Processing.* Reading, MA: Addison-Wesley, 1977.

GRIFFITH, A. K. "Edge detection in simple scenes using a priori information." *IEEE Trans. Computers 22*, 4, April 1973.

HANSON, A. R. and E. M. RISEMAN (Eds.). *Computer Vision Systems (CVS).* New York: Academic Press, 1978.

HORN, B. K. P. "Determining lightness from an image." *CGIP 3*, 4, December 1974, 277-299.

HORN, B. K. P. "Shape from shading." In *PCV*, 1975.

HORN, B. K. P. and B. G. SCHUNCK. "Determining optical flow." AI Memo 572, AI Lab, MIT, April 1980.

HORN, B. K. P. and R. W. SJOBERG. "Calculating the reflectance map." *Proc.*, DARPA IU Workshop, November 1978, 115-126.

HUBEL, D. H. and T. N. WIESEL. "Brain mechanisms of vision." *Scientific American*, September 1979, 150-162.

HUECKEL, M. "An operator which locates edges in digitized pictures." *J. ACM 18*, 1, January 1971, 113-125.

HUECKEL, M. "A local visual operator which recognizes edges and lines." *J. ACM 20*, 4, October 1973, 634-647.

IKEUCHI, K. "Numerical shape from shading and occluding contours in a single view." AI Memo 566, AI Lab, MIT, revised February 1980.

KIRSCH, R. A. "Computer determination of the constituent structure of biological images." *Computers and Biomedical Research 4*, 3, June 1971, 315-328.

KNUTH, D. E. *The Art of Computer Programming.* Reading, MA: Addison-Wesley, 1973.

LEVINE, M. D. "A knowledge-based computer vision system." In *CVS*, 1978.

LIU, H. K. "Two- and three-dimensional boundary detection." *CGIP 6*, 2, 1977, 123-134.

MARR, D. and T. POGGIO. "Cooperative computation of stereo disparity." *Science 194*, 1976, 283-287.

MARR, D. and T. POGGIO. "A theory of human stereo vision." AI Memo 451, AI Lab, MIT, November 1977.

MERO, L. and Z. VASSY. "A simplified and fast version of the Hueckel operator for finding optimal edges in pictures." *Proc.*, 4th IJCAI, September 1975, 650-655.

MORAVEC, H. P. "Towards automatic visual obstacle avoidance." *Proc.*, 5th IJCAI, August 1977, 584.

NEVATIA, R. "Evaluation of a simplified Hueckel edge-line detector." Note, *CGIP 6*, 6, December 1977, 582-588.

PHONG, B-T. "Illumination for computer generated pictures." *Commun. ACM 18*, 6, June 1975, 311-317.

PINGLE, K. K. and J. M. TENENBAUM. "An accommodating edge follower." *Proc.*, 2nd IJCAI, September 1971, 1-7.

PRAGER, J. M. "Extracting and labeling boundary segments in natural scenes." *IEEE Trans. PAMI 2*, 1, January 1980, 16-27.

PRATT, W. K. *Digital Image Processing.* New York: Wiley-Interscience, 1978.

PREWITT, J. M. S. "Object enhancement and extraction." In *Picture Processing and Psychopictorics*, B. S. Lipkin and A. Rosenfeld (Eds.). New York: Academic Press, 1970.

QUAM, L. and M. J. HANNAH. "Stanford automated photogrammetry research." AIM-254, Stanford AI Lab, November 1974.

ROBERTS, L. G. "Machine perception of three-dimensional solids." In *Optical and Electro-optical Information Processing*, J. P. Tippett et al. (Eds.). Cambridge, MA: MIT Press, 1965.

ROSENFELD, A. and A. C. KAK. *Digital Picture Processing.* New York: Academic Press, 1976.

ROSENFELD, A., R. A. HUMMEL, and S. W. ZUCKER. "Scene labelling by relaxation operations." *IEEE Trans. SMC 6*, 1976, 430.

RUSSELL, D. L. (Ed.). *Calculus of Variations and Control Theory.* New York: Academic Press, 1976.

SHAPIRA, R. "A technique for the reconstruction of a straight-edge, wire-frame object from two or more central projections." *CGIP 3*, 4, December 1974, 318-326.

SHIRAI, V. "Analyzing intensity arrays using knowledge about scenes." In *PCV*, 1975.

STEIGLITZ, K. *An Introduction to Discrete Systems.* New York: Wiley, 1974.

STOCKHAM, T. J., Jr. "Image processing in the context of a visual model." *Proc. IEEE 60*, 7, July 1972, 828-842.

TANIMOTO, S. and T. PAVLIDIS. "A hierarchical data structure for picture processing." *CGIP 4*, 2, June 1975, 104-119.

TRETIAK, O. J. "A parameteric model for edge detection." *Proc.*, 3rd COMPSAC, November 1979, 884-887.

TURNER, K. J. "Computer perception of curved objects using a television camera." Ph.D. dissertation, Univ. Edinburgh, 1974.

WECHSLER, H. and J. SKLANSKY. "Finding the rib cage in chest radiographs." *Pattern Recognition 9*, 1977, 21-30.

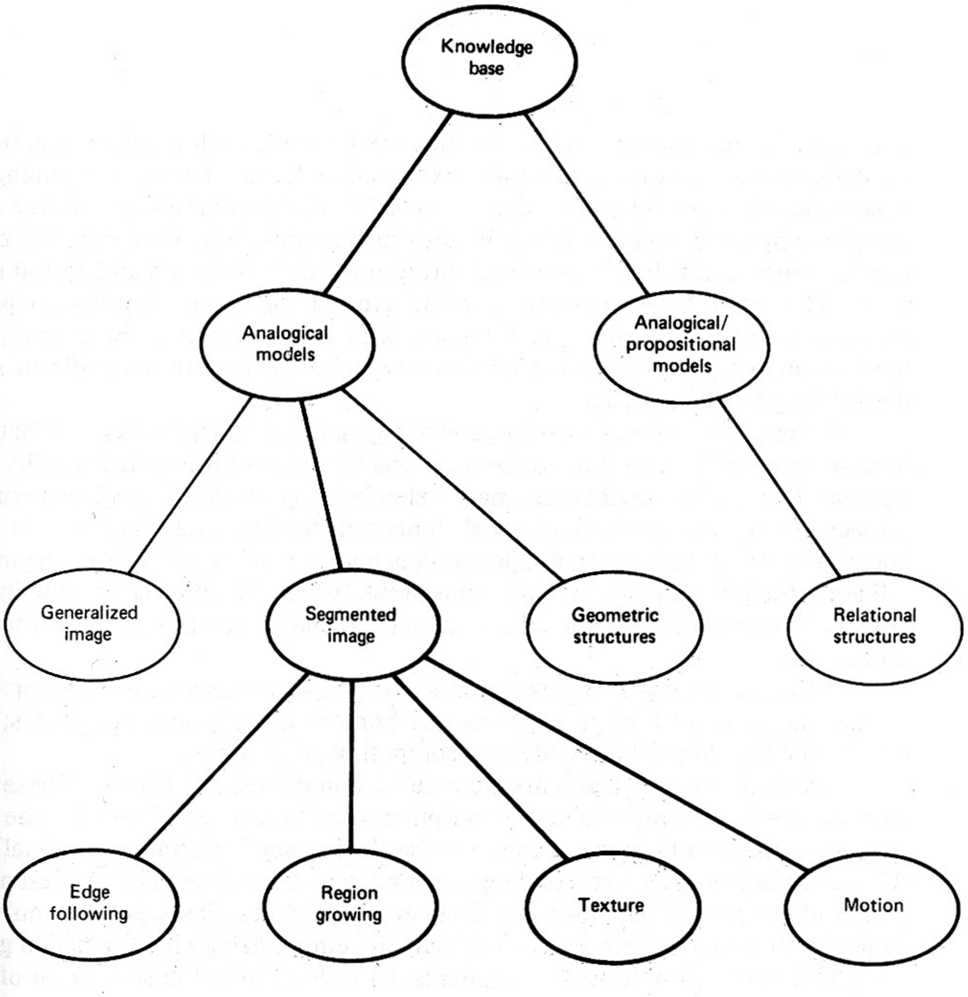
WHITTED, T. "An improved illumination model for shaded display." *Comm. ACM 23*, 6, June 1980, 343-349.

WOODHAM, R. J. "Photometric stereo: A reflectance map technique for determining surface orientation from image intensity." *Proc.*, 22nd International Symp., Society of Photo-optical Instrumentation Engineers, San Diego, CA, August 1978, 136-143.

ZUCKER, S. W. and R. A. HUMMEL. "An optimal three-dimensional edge operator." Report 79-10, McGill Univ., April 1979.

ZUCKER, S. W., R. A. HUMMEL, and A. ROSENFELD. "An application of relaxation labeling to line and curve enhancement." *IEEE Trans. Computers 26*, 1977. April N°4 pp394

SEGMENTED IMAGES II



The idea of segmentation has its roots in work by the Gestalt psychologists (e.g., Kohler), who studied the preferences exhibited by human beings in grouping or organizing sets of shapes arranged in the visual field. Gestalt principles dictate certain grouping preferences based on features such as proximity, similarity, and continuity. Other results had to do with figure/ground discrimination and optical illusions. The latter have provided a fertile ground for vision theories to post-Gestaltists such as Gibson and Gregory, who emphasize that these grouping mechanisms organize the scene into *meaningful units* that are a significant step toward image understanding.

In computer vision, grouping parts of a generalized image into units that are homogeneous with respect to one or more characteristics (or features) results in a *segmented image*. The segmented image extends the generalized image in a crucial respect: it contains the beginnings of domain-dependent interpretation. At this descriptive level the internal domain-dependent models of objects begin to influence the grouping of generalized image structures into units meaningful in the domain. For instance, the model may supply crucial parameters to segmentation procedures.

In the segmentation process there are two important aspects to consider: one is the data structure used to keep track of homogeneous groups of features; the other is the transformation involved in computing the features.

Two basic sorts of segments are natural: boundaries and regions. These can be used combined into a single descriptive structure, a set of nodes (one per region), connected by arcs representing the "adjacency" relation. The "dual" of this structure has arcs corresponding to boundaries connecting nodes representing points where several regions meet. Chapters 4 and 5 describe segmentation with respect to boundaries and regions respectively, emphasizing gray levels and gray-level differences as indicators of segments. Of course, from the standpoint of the algorithms involved, it is irrelevant whether the features are intensity gray levels or intrinsic image values perhaps representing motion, color, or range.

Texture and motion images are addressed in Chapters 6 and 7. Each has several computationally difficult aspects, and neither has received the attention given static, nontextured images. However, each is very important in the segmentation enterprise.

Boundary Detection 4

4.1 ON ASSOCIATING EDGE ELEMENTS

Boundaries of objects are perhaps the most important part of the hierarchy of structures that links raw image data with their interpretation [Marr 1975]. Chapter 3 described how various operators applied to raw image data can yield primitive edge elements. However, an image of only disconnected edge elements is relatively featureless; additional processing must be done to group edge elements into structures better suited to the process of interpretation. The goal of the techniques in this chapter is to perform a level of *segmentation*, that is, to make a coherent one-dimensional (*edge*) feature from many individual local edge elements. The feature could correspond to an object boundary or to any meaningful boundary between scene entities. The problems that edge-based segmentation algorithms have to contend with are shown by Fig. 4.1, which is an image of the local edge elements yielded by one common edge operator applied to a chest radiograph. As can be seen, the edge elements often exist where no meaningful scene boundary does, and conversely often are absent where a boundary is. For example, consider the boundaries of ribs as revealed by the edge elements. Missing edge elements and extra edge elements both tend to frustrate the segmentation process.

The methods in this chapter are ordered according to the amount of knowledge incorporated into the grouping operation that maps edge elements into boundaries. "Knowledge" means implicit or explicit constraints on the likelihood of a given grouping. Such constraints may arise from general physical arguments or (more often) from stronger restrictions placed on the image arising from domain-dependent considerations. If there is much knowledge, this implies that the global form of the boundary and its relation to other image structures is very constrained. Little prior knowledge means that the segmentation must proceed more on the basis of local clues and evidence and general (domain-dependent) assumptions with fewer expectations and constraints on the final resulting boundary.

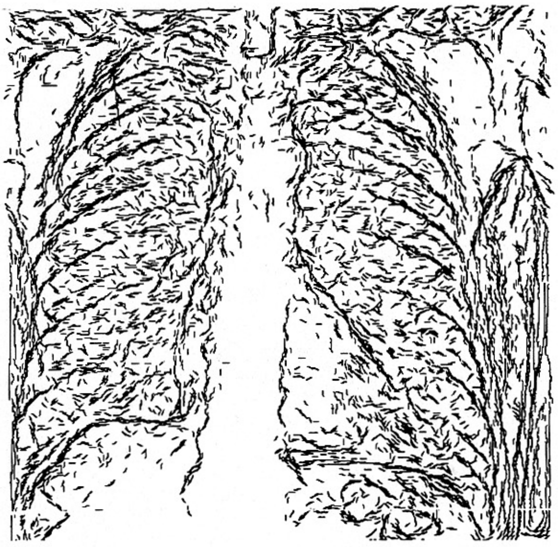


Fig. 4.1 Edge elements in a chest radiograph.

These constraints take many forms. Knowledge of where to expect a boundary allows very restricted searches to verify the edge. In many such cases, the domain knowledge determines the type of curve (its parameterization or functional form) as well as the relevant "noise processes." In images of polyhedra, only straight-edged boundaries are meaningful, and they will come together at various sorts of vertices arising from corners, shadows of corners, and occlusions. Human rib boundaries appear approximately like conic sections in chest radiographs, and radiographs have complex edge structures that can compete with rib edges. All this specific knowledge can and should guide our choice of grouping method.

If less is known about the specific image content, one may have to fall back on general world knowledge or heuristics that are true for most domains. For instance, in the absence of evidence to the contrary, the shorter line between two points might be selected over a longer line. This sort of general principle is easily built into evaluation functions for boundaries, and used in segmentation algorithms that proceed by methodically searching for such groupings. If there are no a priori restrictions on boundary shapes, a general contour-extraction method is called for, such as edge following or linking of edge elements.

The methods we shall examine are the following:

1. *Searching near an approximate location.* These are methods for refining a boundary given an initial estimate.
2. *The Hough transform.* This elegant and versatile technique appears in various guises throughout computer vision. In this chapter it is used to detect boundaries whose shape can be described in an analytical or tabular form.
3. *Graph searching.* This method represents the image of edge elements as a graph. Thus a boundary is a path through a graph. Like the Hough transform, these techniques are quite generally applicable.

4. *Dynamic programming.* This method is also very general. It uses a mathematical formulation of the globally best boundary and can find boundaries in noisy images.
5. *Contour following.* This hill-climbing technique works best with good image data.

4.2 SEARCHING NEAR AN APPROXIMATE LOCATION

If the approximate or a priori likely location of a boundary has been determined somehow, it may be used to guide the effort to refine that boundary [Kelly 1971]. The approximate location may have been found by one of the techniques below applied to a lower resolution image, or it may have been determined using high-level knowledge.

4.2.1 Adjusting A Priori Boundaries

This idea was described by [Bolles 1977] (see Fig. 4.2). Local searches are carried out at regular intervals along directions perpendicular to the approximate (a priori) boundary. An edge operator is applied to each of the discrete points along each of these perpendicular directions. For each such direction, the edge with the highest magnitude is selected from among those whose orientations are nearly parallel to the tangent at the point on the nearby a priori boundary. If sufficiently many elements are found, their locations are fit with an analytic curve such as a low-degree polynomial, and this curve becomes the representation of the boundary.

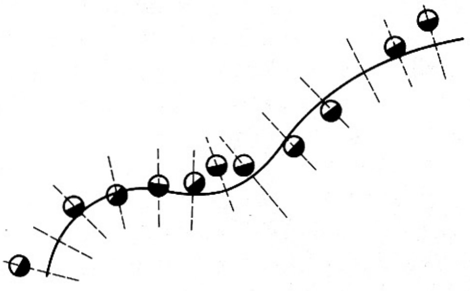


Fig. 4.2 Search orientations from an approximate boundary location.

4.2.2 Non-linear Correlation in Edge Space

In this correlation-like technique, the a priori boundary is treated as a rigid template, or piece of rigid wire along which edge operators are attached like beads. The a priori representation thus also contains relative locations at which the existence of edges will be tested (Fig. 4.3). An edge element returned by the edge-operator application "matches" the a priori boundary if its contour is tangent to the template and its magnitude exceeds some threshold. The template is to be moved around the image, and for each location, the number of matches is computed. If the number of matches exceeds a threshold, the boundary location is declared to

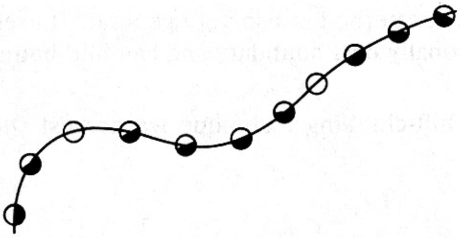


Fig. 4.3 A template for edge-operator application.

be the current template location. If not, the template is moved to a different image point and the process is repeated. Either the boundary will be located or there will eventually be no more image points to try.

4.2.3 Divide-and-Conquer Boundary Detection

This is a technique that is useful in the case that a low-curvature boundary is known to exist between two edge elements and the noise levels in the image are low (Algorithm 8.1). In this case, to find a boundary point in between the two known points, search along the perpendiculars of the line joining the two points. The point of maximum magnitude (if it is over some threshold) becomes a break point on the boundary and the technique is applied recursively to the two line segments formed between the three known boundary points. (Some fix must be applied if the maximum is not unique.) Figure 4.4 shows one step in this process. Divide-and-conquer boundary detection has been used to outline kidney boundaries on computed tomograms (these images were described in Section 2.3.4) [Selfridge et al. 1979].

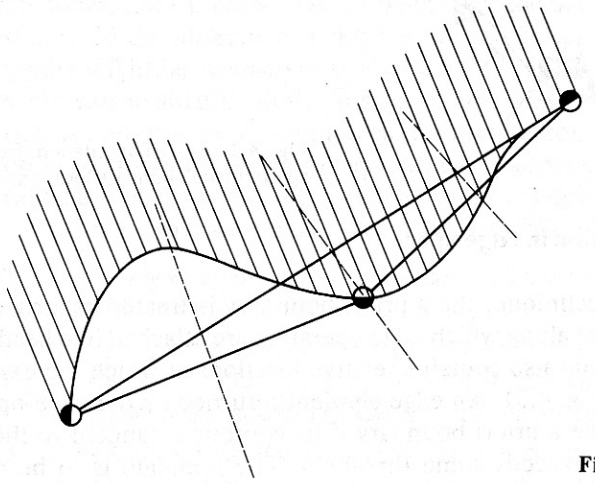


Fig. 4.4 Divide and conquer technique.

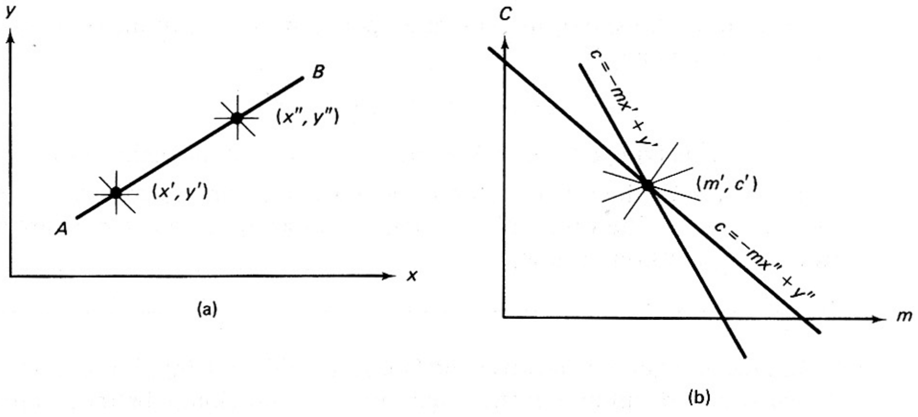


Fig. 4.5 A line (a) in image space; (b) in parameter space.

4.3 THE HOUGH METHOD FOR CURVE DETECTION

The classical Hough technique for curve detection is applicable if little is known about the location of a boundary, but its shape can be described as a parametric curve (e.g., a straight line or conic). Its main advantages are that it is relatively unaffected by gaps in curves and by noise.

To introduce the method [Duda and Hart 1972], consider the problem of detecting straight lines in images. Assume that by some process image points have been selected that have a high likelihood of being on linear boundaries. The Hough technique organizes these points into straight lines, basically by considering all possible straight lines at once and rating each on how well it explains the data.

Consider the point $\mathbf{x}'$ in Fig. 4.5a, and the equation for a line $y = mx + c$. What are the lines that could pass through $\mathbf{x}'$? The answer is simply all the lines with m and c satisfying $y' = mx' + c$. Regarding (x', y') as fixed, the last equation is that of a line in $m-c$ space, or parameter space. Repeating this reasoning, a second point $(\mathbf{x}'', \mathbf{y}'')$ will also have an associated line in parameter space and, furthermore, these lines will intersect at the point (m', c') which corresponds to the line AB connecting these points. In fact, all points on the line AB will yield lines in parameter space which intersect at the point (m', c'), as shown in Fig. 4.5b.

This relation between image space $\mathbf{x}$ and parameter space suggests the following algorithm for detecting lines:

Algorithm 4.1: Line Detection with the Hough Algorithm

1. Quantize parameter space between appropriate maximum and minimum values for c and m.
2. Form an accumulator array $A(c, m)$ whose elements are initially zero.
3. For each point (x,y) in a *gradient* image such that the strength of the gradient

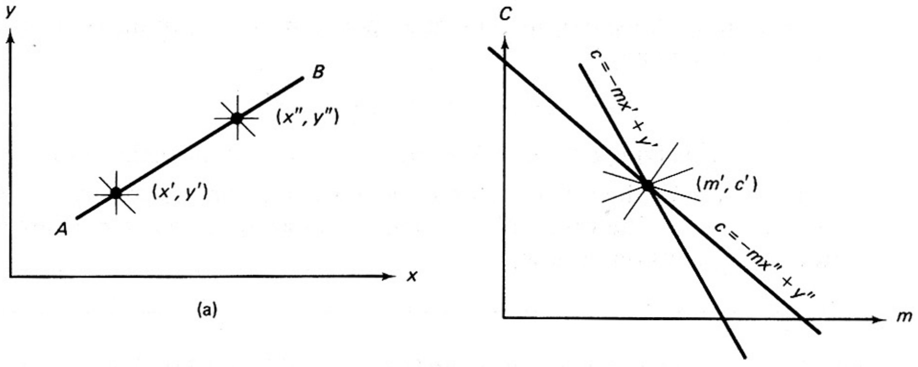


Fig. 4.5 A line (a) in image space; (b) in parameter space.

4.3 THE HOUGH METHOD FOR CURVE DETECTION

The classical Hough technique for curve detection is applicable if little is known about the location of a boundary, but its shape can be described as a parametric curve (e.g., a straight line or conic). Its main advantages are that it is relatively unaffected by gaps in curves and by noise.

To introduce the method [Duda and Hart 1972], consider the problem of detecting straight lines in images. Assume that by some process image points have been selected that have a high likelihood of being on linear boundaries. The Hough technique organizes these points into straight lines, basically by considering all possible straight lines at once and rating each on how well it explains the data.

Consider the point $\mathbf{x}'$ in Fig. 4.5a, and the equation for a line $y = mx + c$. What are the lines that could pass through $\mathbf{x}'$? The answer is simply all the lines with m and c satisfying $y' = mx' + c$. Regarding (x', y') as fixed, the last equation is that of a line in $m-c$ space, or parameter space. Repeating this reasoning, a second point $(\mathbf{x}'', \mathbf{y}'')$ will also have an associated line in parameter space and, furthermore, these lines will intersect at the point (m', c') which corresponds to the line AB connecting these points. In fact, all points on the line AB will yield lines in parameter space which intersect at the point (m', c'), as shown in Fig. 4.5b.

This relation between image space $\mathbf{x}$ and parameter space suggests the following algorithm for detecting lines:

Algorithm 4.1: Line Detection with the Hough Algorithm

1. Quantize parameter space between appropriate maximum and minimum values for c and m.
2. Form an accumulator array $A(c, m)$ whose elements are initially zero.
3. For each point (x,y) in a *gradient* image such that the strength of the gradient

exceeds some threshold, increment all points in the accumulator array along the appropriate line, i.e.,

$$A(c, m) := A(c, m) + 1$$

for m and c satisfying $c = -mx + y$ within the limits of the digitization.

4. Local maxima in the accumulator array now correspond to collinear points in the image array. The values of the accumulator array provide a measure of the number of points on the line.

This technique is generally known as the Hough technique [Hough 1962].

Since m may be infinite in the slope-intercept equation, a better parameterization of the line is $x \sin\theta + y \cos\theta = r$. This produces a sinusoidal curve in (r, θ) space for fixed x, y, but otherwise the procedure is unchanged.

The generalization of this technique to other curves is straightforward and this method works for any curve $f(\mathbf{x}, \mathbf{a}) = 0$, where a is a parameter vector. (In this chapter we often use the symbol f as various general functions unrelated to the image gray-level function.) In the case of a circle parameterized by

$$(x-a)^2 + (y-b)^2 = r^2 \tag{4.1}$$

for fixed $\mathbf{x}$, the modified algorithm 4.1 increments values of a, b, r lying on the surface of a cone. Unfortunately, the computation and the size of the accumulator array increase exponentially as the number of parameters, making this technique practical only for curves with a small number of parameters.

The Hough method is an efficient implementation of a generalized matched filtering strategy (i.e., a template-matching paradigm). For instance, in the case of a circle, imagine a template composed of a circle of 1's (at a fixed radius R) and 0's everywhere else. If this template is convolved with the gradient image, the result is the portion of the accumulator array $A(a, b, R)$.

In its usual form, the technique yields a set of parameters for a curve that best explains the data. The parameters may specify an infinite curve (e.g., a line or parabola). Thus, if a finite curve segment is desired, some further processing is necessary to establish end points.

4.3.1 Use of the Gradient

Dramatic reductions in the amount of computation can be achieved if the gradient direction is integrated into the algorithm [Kimme et al. 1975]. For example, consider the problem of detecting a circle of fixed radius R.

Without gradient information, all values a, b lying on the circle given by (4.1) are incremented. With the gradient direction, only the points near (a,b) in Fig. 4.6 need be incremented. From geometrical considerations, the point (a,b) is given by

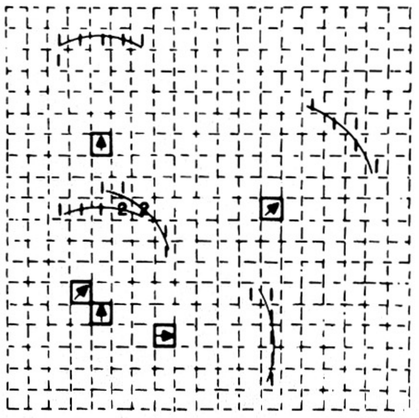

Contents of accumulator tray
Gradient direction information for artifact $\Delta\phi = 45$
□ Denotes a pixel in P(x) superimposed on accumulator tray
↗ Denotes the gradient direction

Fig 4.6 Reduction in computation with gradient information

$$a = x - r\sin\phi \tag{4.2}$$
$$b = y + r\cos\phi$$

where $\phi(x)$ is the gradient angle returned by an edge operator. Implicit in these equations is the assumption that the circle is the boundary of a disk that has gray levels greater than its surroundings. These equations may also be derived by differentiating (4.2), recognizing that $dy/dx = \tan\phi$, and solving for a and b between the resultant equation and (4.2). Similar methods can be applied to other conics. In each case, the use of the gradient saves one dimension in the accumulator array.

The gradient magnitude can also be used as a heuristic in the incrementing procedure. Instead of incrementing by unity, the accumulator array location may be incremented by a function of the gradient magnitude. This heuristic can balance the magnitude of brightness change across a boundary with the boundary length, but it can lead to detection of phantom lines indicated by a few bright points, or to missing dim but coherent boundaries.

4.3.2 Some Examples

The Hough technique has been used successfully in a variety of domains. Some examples include the detection of human hemoglobin fingerprints [Ballard et al. 1975], the detection of tumors in chest films [Kimme et al. 1975], the detection of storage tanks in aerial images [Lantz et al. 1978], and the detection of ribs in chest radiographs [Wechsler and Sklansky 1977]. Figure 4.7 shows the tumor-detection application. A section of the chest film (Fig. 4.7b) is searched for disks of radius 3 units. In Fig. 4.7c, the resultant accumulator array $A[a, b, 3]$ is shown in a pictoral fashion, by interpreting the array values as gray levels. This process is repeated for various radii and then a set of likely circles is chosen by setting a radius-dependent threshold for the accumulator array contents. This result is shown in Fig. 4.7d. The

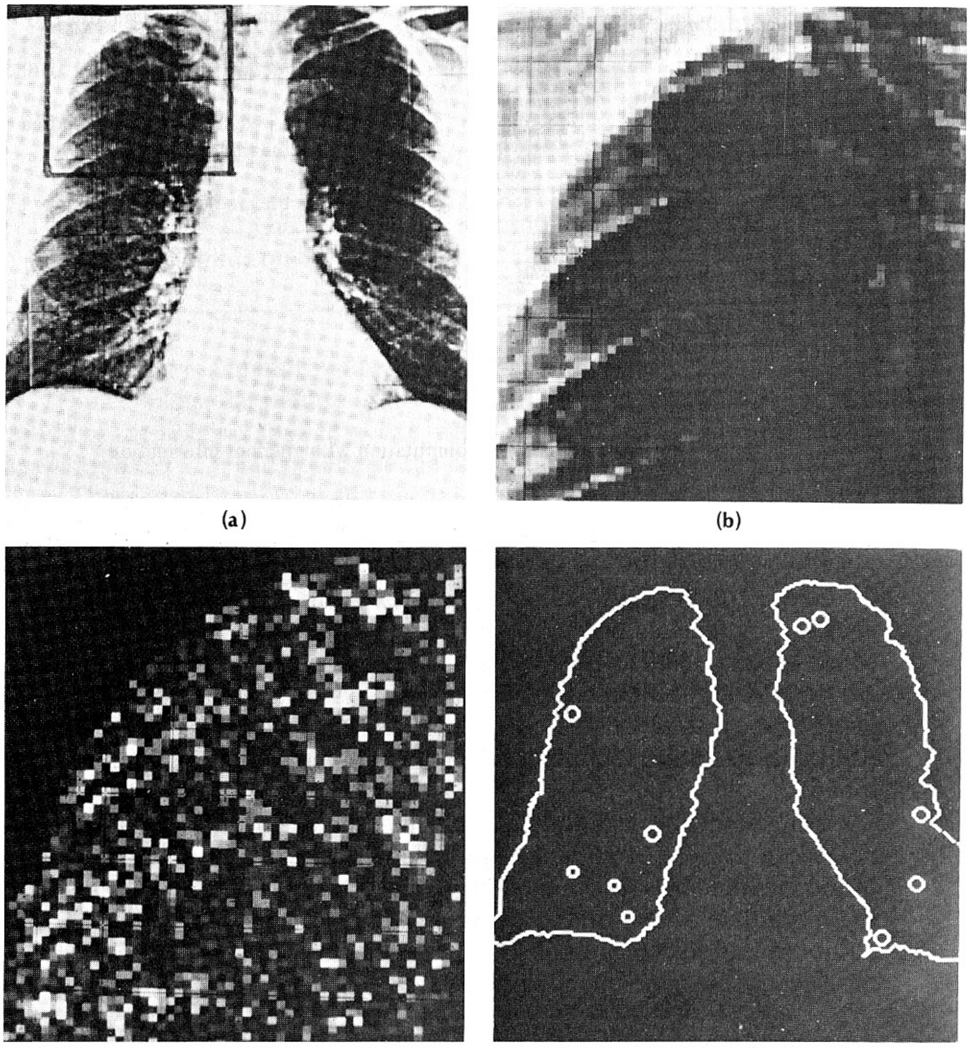

(a) (b) (c) (d)

Fig. 4.7 Using the Hough technique for circular shapes. (a) Radiograph. (b) Window. (c) Accumulator array for $r = 3$. (d) Results of maxima detection.

circular boundaries detected by the Hough technique are overlaid on the original image.

4.3.3 Trading Off Work in Parameter Space for Work in Image Space

Consider the example of detecting ellipses that are known to be oriented so that a principal axis is parallel to the x axis. These can be specified by four parameters. Using the equation for the ellipse together with its derivative, and substituting for the known gradient as before, one can solve for two parameters. In the equation

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1 \tag{4.3}$$

x is an edge point and x_0, y_0, a, and b are parameters. The equation for its derivative is

$$\frac{(x-x_0)}{a} + \frac{(y-y_0)^2}{b^2}\frac{dy}{dx} = 0 \tag{4.4}$$

where $dy/dx = \tan\phi(x)$. The Hough algorithm becomes:

Algorithm 4.2: Hough technique applied to ellipses

For each discrete value of x and y, increment the point in parameter space given by a, b, x_0, y_0, where

$$x = x_0 \pm \frac{a}{(1 + b^2/a^2 \tan^2\phi)^{1/2}} \tag{4.5}$$

$$y = y_0 \pm \frac{b}{(1 + a^2 \tan^2\phi/b^2)^{1/2}} \tag{4.6}$$

that is,

$$A(a, b, x_0, y_0) := A(a, b, x_0, y_0) + 1$$

For a and b each having m values the computational cost is proportional to m^2.

Now suppose that we consider all pairwise combinations of edge elements. This introduces two additional equations like (4.3) and (4.4), and now the four-parameter point can be determined exactly. That is, the following equations can be solved for a unique x_0, y_0, a, b.

$$\frac{(x_1-x_0)^2}{a^2} + \frac{(y_1-y_0)^2}{b^2} = 1 \tag{4.7a}$$

$$\frac{(x_2-x_0)^2}{a^2} + \frac{(y_2-y_0)^2}{b^2} = 1 \tag{4.7b}$$

$$\frac{x_1-x_0}{a^2} + \frac{y_1-y_0}{b^2}\frac{dy}{dx} = 0 \tag{4.7c}$$

$$\frac{x_2-x_0}{a^2} + \frac{y_2-y_0}{b^2}\frac{dy}{dx} = 0 \tag{4.7d}$$

$$\frac{dy}{dx} = \tan\phi \qquad \left(\frac{dy}{dx} \text{ is known from the edge operator}\right)$$

Their solution is left as an exercise. The amount of effort in the former case was proportional to the product of the number of discrete values of a and b, whereas this case involves effort proportional to the square of the number of edge elements.

4.3.4 Generalizing the Hough Transform

Consider the case where the object being sought has no simple analytic form, but has a particular silhouette. Since the Hough technique is so closely related to template matching, and template matching can handle this case, it is not surprising that the Hough technique can be generalized to handle this case also. Suppose for the moment that the object appears in the image with known shape, orientation, and scale. (If orientation and scale are unknown, they can be handled in the same way that additional parameters were handled earlier.) Now pick a reference point in the silhouette and draw a line to the boundary. At the boundary point compute the gradient direction and store the reference point as a function of this direction. Thus it is possible to precompute the location of the reference point from boundary points given the gradient angle. The set of all such locations, indexed by gradient angle, comprises a table termed the *R*-table [Ballard 1981]. Remember that the basic strategy of the Hough technique is to compute the possible loci of reference points in parameter space from edge point data in image space and increment the parameter points in an accumulator array. Figure 4.8 shows the relevant geometry and Table 4.1 shows the form of the *R*-table. For the moment, the reference point coordinates (x_c, y_c) are the only parameters (assuming that rotation and scaling have been fixed). Thus an edge point (x, y) with gradient orientation ϕ constrains the possible reference points to be at $\{x + r_1(\phi) \cos [\alpha_1(\phi)],\ y + r_1(\phi) \sin [\alpha_1(\phi)]\}$ and so on.

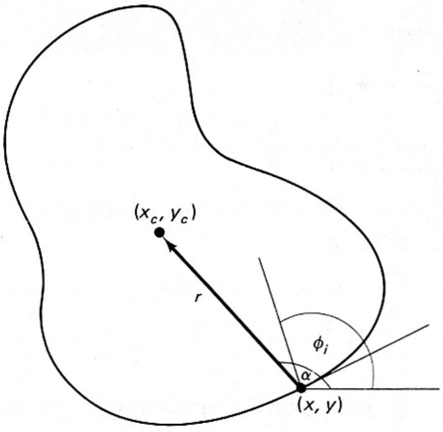


Fig. 4.8 Geometry used to form the *R*-Table.

Table 4.1

INCREMENTATION IN THE GENERALIZED HOUGH CASE

Angle measured from figure boundary to reference point	*Set of radii* $\{\mathbf{r}^k\}$ *where* $\mathbf{r} = (r, \alpha)$
ϕ_1	$\mathbf{r}_1^1, \mathbf{r}_2^1, \ldots, \mathbf{r}_{n_1}^1$
ϕ_2	$\mathbf{r}_1^2, \mathbf{r}_2^2, \ldots, \mathbf{r}_{n_2}^2$
.	.
.	.
.	.
ϕ_m	$\mathbf{r}_1^m, \mathbf{r}_2^m, \ldots, \mathbf{r}_{n_m}^m$

The generalized Hough algorithm may be described as follows:

Algorithm 4.3: Generalized Hough

```
Step 0.  Make a table (like Table 4.1) for the shape to be located.
Step 1.  Form an accumulator array of possible reference points
         $A(x_{c\min} : x_{c\max}, y_{c\min} : y_{c\max})$ initialized to zero.
Step 2.  For each edge point do the following:
    Step 2.1.   Compute $\phi(\mathbf{x})$
    Step 2.2a.  Calculate the possible centers; that is, for each table entry for
                $\phi$, compute
                    $x_c := x + r\,\phi \;\; \cos[\alpha(\phi)]$
                    $y_c := y + r\,\phi \;\; \sin[\alpha(\phi)]$
    Step 2.2b.  Increment the accumulator array
                    $A(x_c, y_c) := A(x_c, y_c) + 1$
Step 3.  Possible locations for the shape are given by maxima in array $A$.
```

The results of using this transform to detect a shape are shown in Fig. 4.9. Figure 4.9a shows an image of shapes. The *R*-table has been made for the middle shape. Figure 4.9b shows the Hough transform for the shape, that is, $A(x_c, y_c)$ displayed as an image. Figure 4.9c shows the shape given by the maxima of

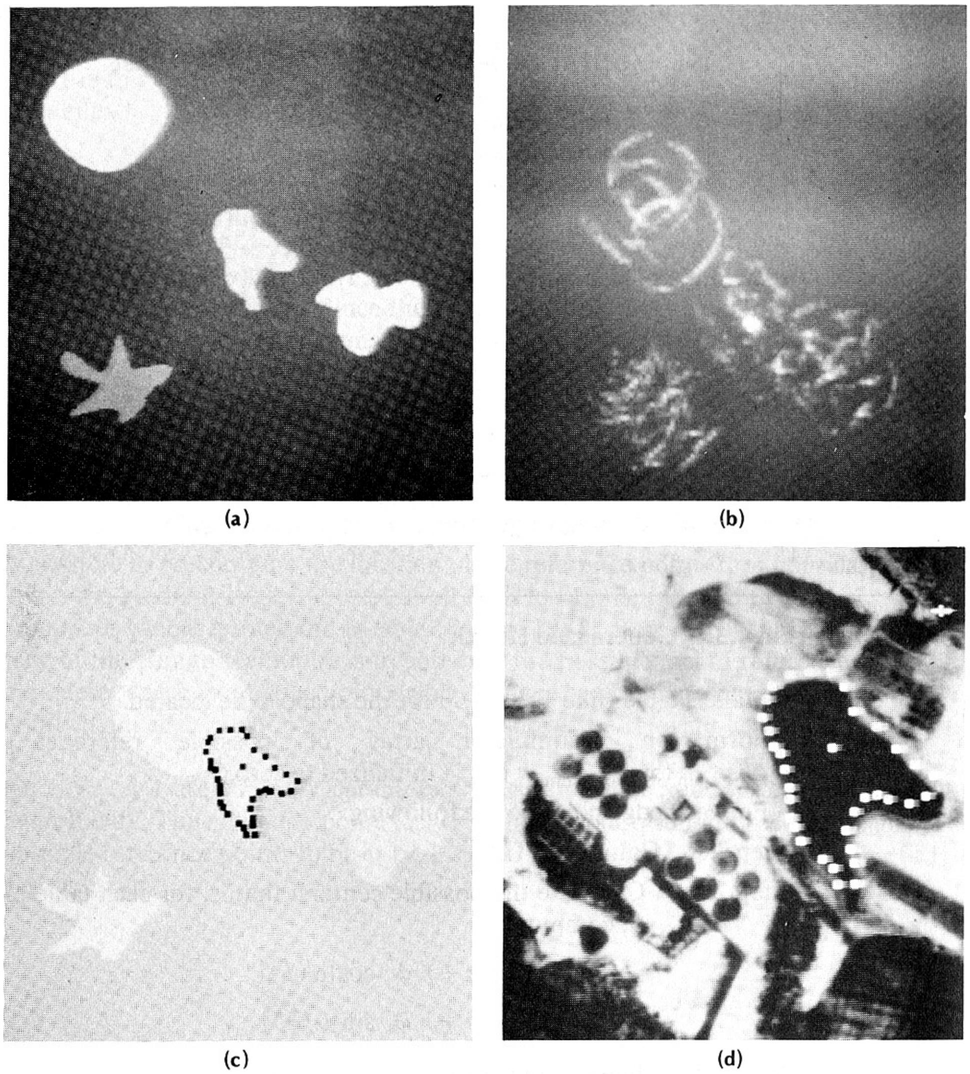


Fig. 4.9 Applying the Generalized Hough technique. (a) Synthetic image. (b) Hough Transform $A(x_c, y_c)$ for middle shape. (c) Detected shape. (d) Same shape in an aerial image setting.

$A(x_c, y_c)$ overlaid on top of the image. Finally, Fig. 4.9d shows the Hough transform used to detect a pond of the same shape in an aerial image.

What about the parameters of scale and rotation, S and θ? These are readily accommodated by expanding the accumulator array and doing more work in the incrementation step. Thus in step 1 the accumulator array is changed to

$$(x_{c\min} : x_{c\max}, y_{c\min} : y_{c\max}, S_{\min} : S_{\max}, \theta_{\min} : \theta_{\max})$$

and step 2.2a is changed to

```
for each table entry for $\phi$ do
    for each $S$ and $\theta$
        $x_c := x + r(\phi) S \cos[\alpha(\phi) + \theta]$
        $y_c := y + r(\phi) S \sin[\alpha(\phi) + \theta]$
```

Finally, step 2.2b is now

$$A(x_c, y_c, S, \theta) := A(x_c, y_c, S, \theta) + 1$$

4.4 EDGE FOLLOWING AS GRAPH SEARCHING

A graph is a general object that consists of a set of nodes $\{n_i\}$ and arcs between nodes $<n_i, n_j>$. In this section we consider graphs whose arcs may have numerical weights or *costs* associated with them. The search for the boundary of an object is cast as a search for the lowest-cost path between two nodes of a weighted graph.

Assume that a gradient operator is applied to the gray-level image, creating the magnitude image $s(\mathbf{x})$ and direction image $\phi(\mathbf{x})$. Now interpret the elements of the direction image $\phi(\mathbf{x})$ as nodes in a graph, each with a weighting factor $s(\mathbf{x})$. Nodes $\mathbf{x}_i$, $\mathbf{x}_j$ have arcs between them if the contour directions $\phi(\mathbf{x}_i)$, $\phi(\mathbf{x}_j)$ are appropriately aligned with the arc directed in the same sense as the contour direction. Figure 4.10 shows the interpretation. To generate Fig. 4.10b impose the following restrictions. For an arc to connect from $\mathbf{x}_i$ to $\mathbf{x}_j$, $\mathbf{x}_j$ must be one of the three possible eight-neighbors in front of the contour direction $\phi(\mathbf{x}_i)$ and, furthermore, $g(\mathbf{x}_i)$

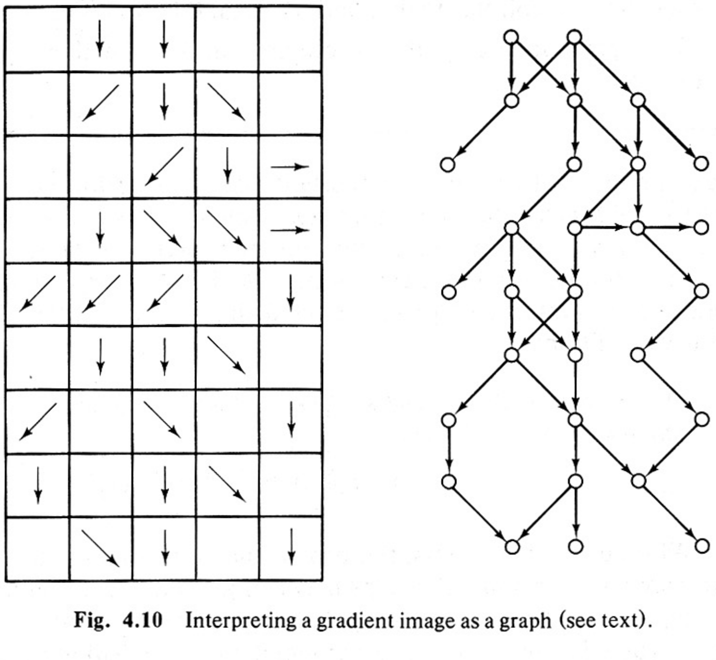


Fig. 4.10 Interpreting a gradient image as a graph (see text).

```
for each table entry for $\phi$ do
    for each $S$ and $\theta$
        $x_c := x + r(\phi) S \cos[\alpha(\phi) + \theta]$
        $y_c := y + r(\phi) S \sin[\alpha(\phi) + \theta]$
```

Finally, step 2.2b is now

$$A(x_c, y_c, S, \theta) := A(x_c, y_c, S, \theta) + 1$$

4.4 EDGE FOLLOWING AS GRAPH SEARCHING

A graph is a general object that consists of a set of nodes $\{n_i\}$ and arcs between nodes $<n_i, n_j>$. In this section we consider graphs whose arcs may have numerical weights or *costs* associated with them. The search for the boundary of an object is cast as a search for the lowest-cost path between two nodes of a weighted graph.

Assume that a gradient operator is applied to the gray-level image, creating the magnitude image $s(\mathbf{x})$ and direction image $\phi(\mathbf{x})$. Now interpret the elements of the direction image $\phi(\mathbf{x})$ as nodes in a graph, each with a weighting factor $s(\mathbf{x})$. Nodes $\mathbf{x}_i$, $\mathbf{x}_j$ have arcs between them if the contour directions $\phi(\mathbf{x}_i)$, $\phi(\mathbf{x}_j)$ are appropriately aligned with the arc directed in the same sense as the contour direction. Figure 4.10 shows the interpretation. To generate Fig. 4.10b impose the following restrictions. For an arc to connect from $\mathbf{x}_i$ to $\mathbf{x}_j$, $\mathbf{x}_j$ must be one of the three possible eight-neighbors in front of the contour direction $\phi(\mathbf{x}_i)$ and, furthermore, $g(\mathbf{x}_i)$

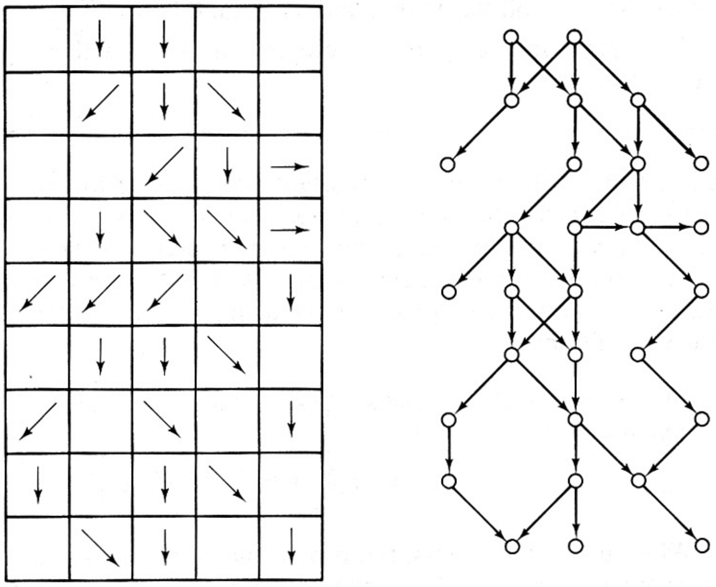


Fig. 4.10 Interpreting a gradient image as a graph (see text).

$> T$, $g(\mathbf{x}_j) > T$, where T is a chosen constant, and $|\{[\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)] \bmod 2\pi\}| < \pi/2$. (Any or all of these restrictions may be modified to suit the requirements of a particular problem.)

To generate a path in a graph from $\mathbf{x}_A$ to $\mathbf{x}_B$ one can apply the well-known technique of heuristic search [Nilsson 1971, 1980]. The specific use of heuristic search to follow edges in images was first proposed by [Martelli 1972]. Suppose:

1. That the path should follow contours that are directed from $\mathbf{x}_A$ to $\mathbf{x}_B$
2. That we have a method for generating the successor nodes of a given node (such as the heuristic described above)
3. That we have an evaluation function $f(\mathbf{x}_j)$ which is an estimate of the optimal cost path from $\mathbf{x}_A$ to $\mathbf{x}_B$ constrained to go through $\mathbf{x}_j$

Nilsson expresses $f(\mathbf{x}_i)$ as the sum of two components: $g(\mathbf{x}_i)$, the estimated cost of journeying from the *start node* $\mathbf{x}_A$ to $\mathbf{x}_i$, and $h(\mathbf{x}_i)$, the estimated cost of the path from $\mathbf{x}_i$ to $\mathbf{x}_B$, the *goal node.*

With the foregoing preliminaries, the heuristic search algorithm (called the A algorithm by Nilsson) can be stated as:

Algorithm 4.4: Heuristic Search (the A Algorithm)

1. "Expand" the start node (put the successors on a list called OPEN with pointers back to the start node).

2. Remove the node $\mathbf{x}_i$ of minimum f from OPEN. If $\mathbf{x}_i = \mathbf{x}_B$, then stop. Trace back through pointers to find optimal path. If OPEN is empty, fail.

3. Else expand node $\mathbf{x}_i$, putting successors on OPEN with pointers back to $\mathbf{x}_i$. Go to step 2.

The component $h(\mathbf{x}_i)$ plays an important role in the performance of the algorithm; if $h(\mathbf{x}_i) = 0$ for all i, the algorithm is a *minimum-cost search* as opposed to a *heuristic search.* If $h(\mathbf{x}_i) > h^*(\mathbf{x}_i)$ (the actual optimal cost), the algorithm may run faster, but may miss the minimum-cost path. If $h(\mathbf{x}_i) < h^*(\mathbf{x}_i)$, the search will always produce a minimum-cost path, provided that h also satisfies the following consistency condition:

If for any two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, $k(\mathbf{x}_i, \mathbf{x}_j)$ is the minimum cost of getting from $\mathbf{x}_i$ to $\mathbf{x}_j$ (if possible), then

$$k(\mathbf{x}_i, \mathbf{x}_j) \geqslant h^*(\mathbf{x}_i) - h^*(\mathbf{x}_j)$$

With our edge elements, there is no guarantee that a path can be found since there may be insurmountable gaps between $\mathbf{x}_A$ and $\mathbf{x}_B$. If finding the edge is crucial, steps should be taken to interpolate edge elements prior to the search, or gaps may be crossed by using the edge element definition of [Martelli 1972]. He defines edges on the image grid structure so that an edge can have a direction even though there is no local gray-level change. This definition is depicted in Fig. 4.11a.

4.4.1 Good Evaluation Functions

A good evaluation function has components specific to the particular task as well as components that are relatively task-independent. The latter components are discussed here.

1. *Edge strength.* If edge strength is a factor, the cost of adding a particular edge element at $\mathbf{x}$ can be included as

$$M - s(\mathbf{x}) \qquad \text{where } M = \max_{\mathbf{x}} s(\mathbf{x})$$

2. *Curvature.* If low-curvature boundaries are desirable, curvature can be measured as some monotonically increasing function of

$$diff[\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)]$$

where diff measures the angle between the edge elements at $\mathbf{x}_j$ and $\mathbf{x}_i$.

3. *Proximity to an approximation.* If an approximate boundary is known, boundaries near this approximation can be favored by adding:

$$d = dist(\mathbf{x}_i, B)$$

to the cost measure. The dist operator measures the minimum distance of the new point $\mathbf{x}_i$ to the approximate boundary B.

4. *Estimates of the distance to the goal.* If the curve is reasonably linear, points near the goal may be favored by estimating h as $d(\mathbf{x}_i, \mathbf{x}_{\text{goal}})$, where d is a distance measure.

Specific implementations of these measures appear in [Ashkar and Modestino 1978; Lester et al. 1978].

4.4.2 Finding All the Boundaries

What if the objective is to find *all* boundaries in the image using heuristic search? In one system [Ramer 1975] Hueckel's operator (Chapter 3) is used to obtain

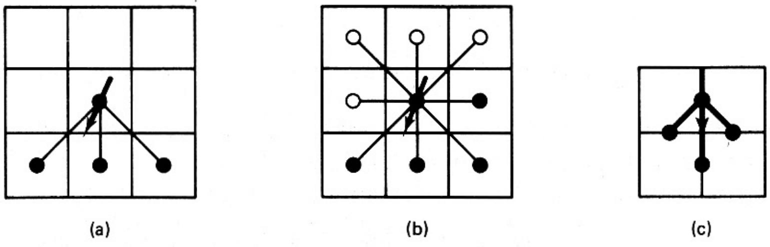


Fig. 4.11 Successor conventions in heuristic search (see text).

strokes, another name for the magnitude and direction of the local gray-level changes. Then these strokes are combined by heuristic search to form sequences of edge elements called *streaks.* Streaks are an intermediate organization which are used to assure a slightly broader coherence than is provided by the individual Hueckel edges. A bidirectional search is used with four eight-neighbors defined in front of the edge and four eight-neighbors behind the edge, as shown in Fig. 4.11b. The search algorithm is as follows:

1. Scan the stroke (edge) array for the most prominent edge.
2. Search in front of the edge until no more successors exist (i.e., a gap is encountered).
3. Search behind the edge until no more predecessors exist.
4. If the bidirectional search generates a path of 3 or more strokes, the path is a streak. Store it in a streak list and go to step 1.

Strokes that are part of a streak cannot be reused; they are marked when used and subsequently skipped.

There are other heuristic procedures for pruning the streaks to retain only *prime streaks.* These are shown in Fig. 4.12. They are essentially similar to the re-

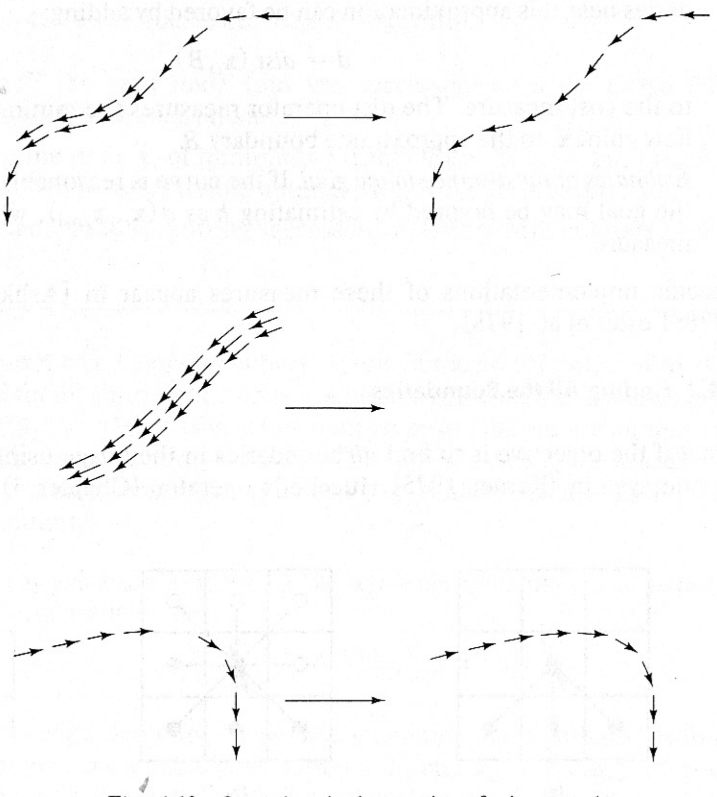


Fig. 4.12 Operations in the creation of prime streaks.

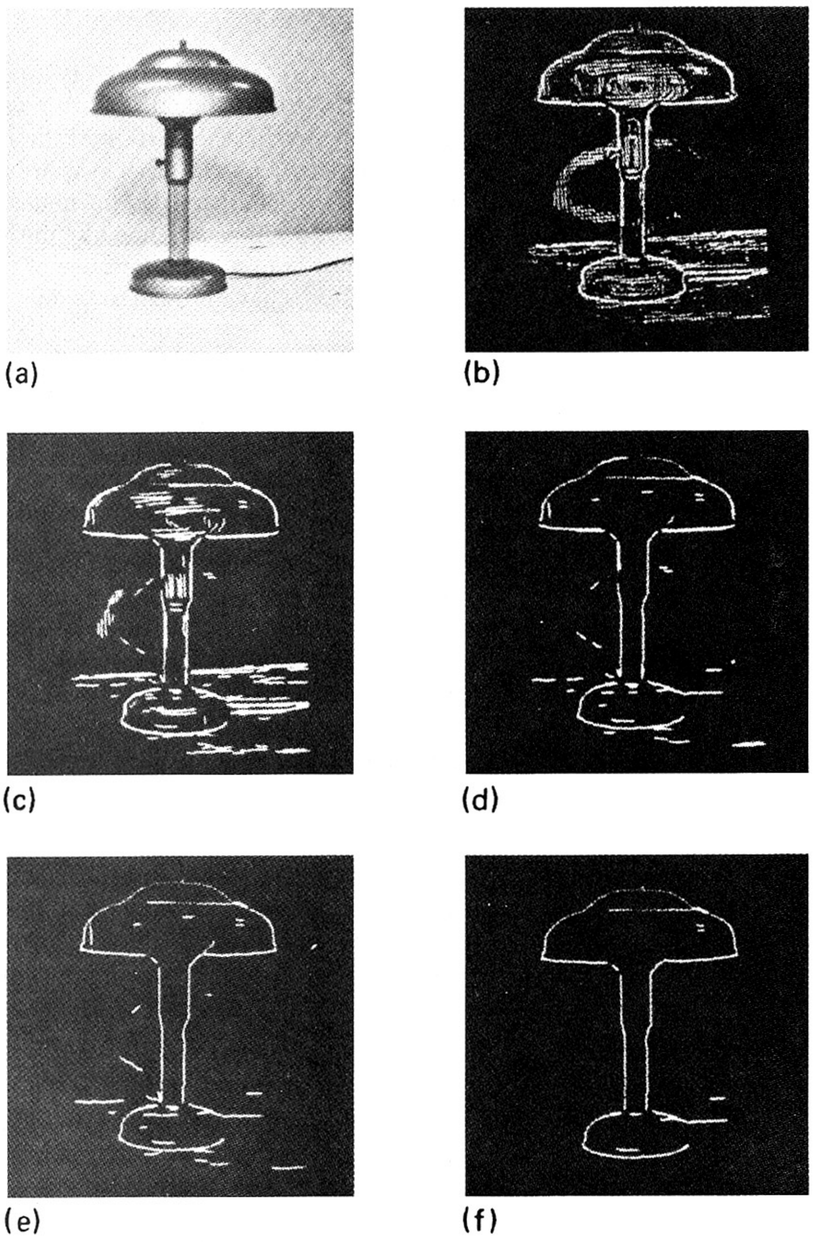


Fig. 4.13 Ramer's results.

laxation operations described in Section 3.3.5. The resultant streaks must still be analyzed to determine the objects they represent. Nevertheless, this method represents a cogent attempt to organize bottom-up edge following in an image. Fig. 4.13 shows an example of Ramer's technique.

4.4.3 Alternatives to the A Algorithm

The primary disadvantage with the heuristic search method is that the algorithm must keep track of a set of current best paths (nodes), and this set may become very large. These nodes represent tip nodes for the portion of the tree of possible paths that has been already examined. Also, since all the costs are nonnegative, a good path may eventually look expensive compared to tip nodes near the start node. Thus, paths from these newer nodes will be extended by the algorithm even though, from a practical standpoint, they are unlikely. Because of these disadvantages, other less rigorous search procedures have proven to be more practical, five of which are described below.

Pruning the Tree of Alternatives

At various points in the algorithm the tip nodes on the OPEN list can be pruned in some way. For example, paths that are short or have a high cost per unit length can be discriminated against. This pruning operation can be carried out whenever the number of alternative tip nodes exceeds some bound.

Modified Depth-First Search

Depth-first search is a meaningful concept if the search space is structured as a tree. Depth-first search means always evaluating the most recent expanded son. This type of search is performed if the OPEN list is structured as a stack in the A algorithm and the top node is always evaluated next. Modifications to this method use an evaluation function f to rate the successor nodes and expand the best of these. Practical examples can be seen in [Ballard and Sklansky 1976; Wechsler and Sklansky 1977; Persoon 1976].

Least Maximum Cost

In this elegant idea [Lester 1978], only the maximum-cost arc of each path is kept as an estimate of g. This is like finding a mountain pass at minimum altitude. The advantage is that g does not build up continuously with depth in the search tree, so that good paths may be followed for a long time. This technique has been applied to finding the boundaries of blood cells in optical microscope images. Some results are shown in Fig. 4.14.

Branch and Bound

The crux of this method is to have some upper bound on the cost of the path [Chien and Fu 1974]. This may be known beforehand or may be computed by actually generating a path between the desired end points. Also, the evaluation function must be monotonically increasing with the length of the path. With these conditions we start generating paths, excluding partial paths when they exceed the current bound.

Modified Heuristic Search

Sometimes an evaluation function that assigns negative costs leads to good results. Thus good paths keep getting better with respect to the evaluation function, avoiding the problem of having to look at all paths near the starting point.

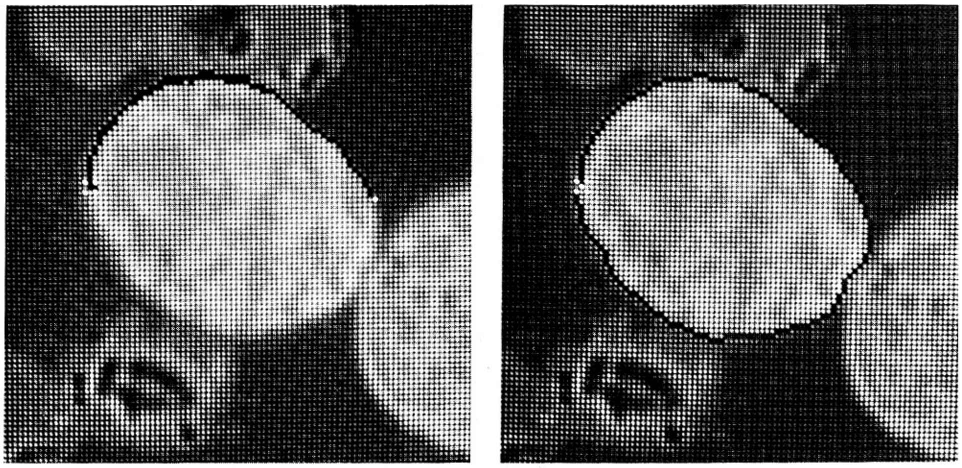

(a) (b)

Fig. 4.14 Using least maximum cost in heuristic search to find cell boundaries in microscope images. (a) A stage in the search process. (b) The completed boundary.

However, the price paid is the sacrifice of the mathematical guarantee of finding the least-cost path. This could be reflected in unsatisfactory boundaries. This method has been used in cineangiograms with satisfactory results [Ashkar and Modestino 1978].

4.5 EDGE FOLLOWING AS DYNAMIC PROGRAMMING

4.5.1 Dynamic Programming

Dynamic programming [Bellman and Dreyfus 1962] is a technique for solving optimization problems when not all variables in the evaluation function are interrelated simultaneously. Consider the problem

$$\max_{x_i} h(x_1, x_2, x_3, x_4) \tag{4.8}$$

If nothing is known about h, the only technique that guarantees a global maximum is exhaustive enumeration of all combinations of discrete values of $x_1, \ldots, x_4$. Suppose that

$$h(\cdot) = h_1(x_1, x_2) + h_2(x_2, x_3) + h_3(x_3, x_4) \tag{4.9}$$

x_1 only depends on x_2 in h_1. Maximize over x_1 in h_1 and tabulate the best value of $h_1(x_1, x_2)$ for each x_2:

$$f_1(x_2) = \max_{x_1} h_1(x_1, x_2) \tag{4.10}$$

Since the values of h_2 and h_3 do not depend on x_1, they need not be considered at

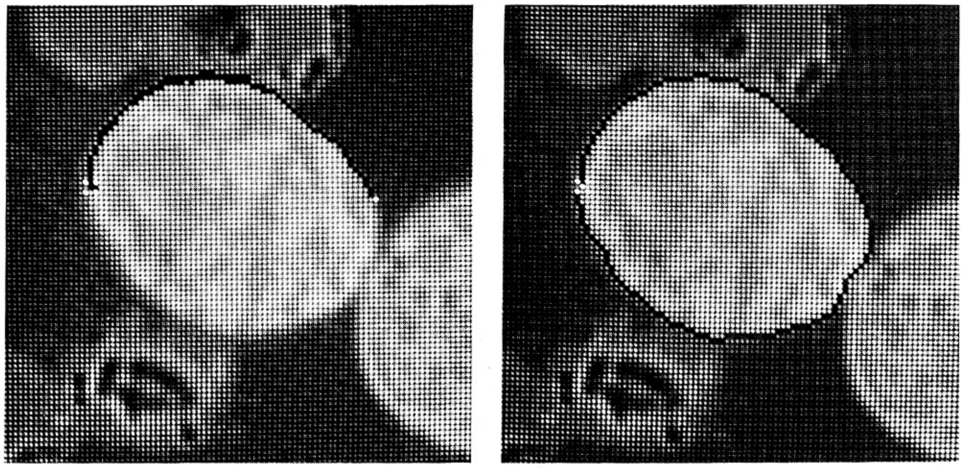

(a) (b)

Fig. 4.14 Using least maximum cost in heuristic search to find cell boundaries in microscope images. (a) A stage in the search process. (b) The completed boundary.

However, the price paid is the sacrifice of the mathematical guarantee of finding the least-cost path. This could be reflected in unsatisfactory boundaries. This method has been used in cineangiograms with satisfactory results [Ashkar and Modestino 1978].

4.5 EDGE FOLLOWING AS DYNAMIC PROGRAMMING

4.5.1 Dynamic Programming

Dynamic programming [Bellman and Dreyfus 1962] is a technique for solving optimization problems when not all variables in the evaluation function are interrelated simultaneously. Consider the problem

$$\max_{x_i} h(x_1, x_2, x_3, x_4) \tag{4.8}$$

If nothing is known about h, the only technique that guarantees a global maximum is exhaustive enumeration of all combinations of discrete values of $x_1, \ldots, x_4$. Suppose that

$$h(\cdot) = h_1(x_1, x_2) + h_2(x_2, x_3) + h_3(x_3, x_4) \tag{4.9}$$

x_1 only depends on x_2 in h_1. Maximize over x_1 in h_1 and tabulate the best value of $h_1(x_1, x_2)$ for each x_2:

$$f_1(x_2) = \max_{x_1} h_1(x_1, x_2) \tag{4.10}$$

Since the values of h_2 and h_3 do not depend on x_1, they need not be considered at this point. Continue in this manner and eliminate x_2 by computing $f_2(x_3)$ as

$$f_2(x_3) = \max_{x_2}[f_1(x_2) + h_2(x_2, x_3)] \tag{4.11}$$

and

$$f_3(x_4) = \max_{x_3}[f_2(x_3) + h_3(x_3, x_4)] \tag{4.12}$$

so that finally

$$\max_{x_i} h = \max_{x_4} f_3(x_4) \tag{4.13}$$

Generalizing the example to N variables, where $f_0(x_1) = 0$,

$$f_{n-1}(x_n) = \max_{x_{n-1}}[f_{n-2}(x_{n-1}) + h_{n-1}(x_{n-1}, x_n)] \tag{4.14}$$

$$\max_{x_i} h(x_i, \ldots, x_N) = \max_{x_N} f_{N-1}(x_N)$$

If each x_i took on 20 discrete values, then to compute $f_N(x_{N+1})$ one must evaluate the maximand for 20 different combinations of x_N and x_{N+1}, so that the resultant computational effort involves $(N-1)20^2 + 20$ such evaluations. This is a striking improvement over exhaustive evaluation, which would involve 20^N evaluations of h!

Consider the artificial example summarized in Table 4.2. In this example, each **x** can take on one of three discrete values. The h_i are completely described by their respective tables. For example, the value of $h_i(0, 1) = 5$. The solution steps are summarized in Table 4.3. In step 1, for each x_2 the value of x_1 that maximizes $h_1(x_1, x_2)$ is computed. This is the largest entry in each of the columns of h. Store the function value as $f_1(x_2)$ and the optimizing value of x_1 also as a function of x_2. In step 2, add $f_1(x_2)$ to $h_2(x_2, x_3)$. This is done by adding f_1 to each row of h_2, thus computing the quantity inside the braces of (4.11). Now to complete step 2, for each x_3, compute the x_2 that maximizes $h_2 + f_1$ by selecting the largest entry in each row of the appropriate table. The rest of the steps are straightforward once these are understood. The solution is found by tracing back through the tables. For example, for $x_4 = 2$ we see that the best x_3 is -1, and therefore the best x_2 is 3 and x_1 is 1. This step is denoted by arrows.

Table 4.2

DEFINITION OF *h*

x_1 \ x_2	1	2	3
0	5	7	3
1	2	1	8
2	6	3	3

h_1

x_2 \ x_3	−1	0	1
1	1	7	1
2	1	1	3
3	5	6	2

h_2

x_3 \ x_4	1	2	3
−1	7	9	8
0	2	3	6
1	5	4	1

h_3

Table 4.3

METHOD OF SOLUTION USING DYNAMIC PROGRAMMING

Step 1

x_2	f_1	x_1
1	6	2
2	7	0
3	8	1

Step 2

$x_2 \backslash x_3$	−1	0	1
1	7	13	7
2	8	8	10
3	13	14	

x_3	f_2	x_2
−1	13	3
0	14	3
1	10	2

Step 3

$x_3 \backslash x_4$	1	2	3
−1	20	22	21
0	16	17	20
1	15	14	11

x_4	f_3	x_3
1	20	−1
2	22	−1
3	21	−1

Step 4: Optimal x_i's are found by examing tables (dashed line shows the order in which they are recovered).

Solution: $h^* = 22$
$x_1^* = 1, x_2^* = 3, x_3^* = -1, x_4^* = 2$

4.5.2 Dynamic Programming for Images

To formulate the boundary-following procedure as dynamic programming, one must define an evaluation function that embodies a notion of the "best boundary" [Montanari 1971; Ballard 1976]. Suppose that a local edge detection operator is applied to a gray-level picture to produce edge magnitude and direction information. Then one possible criterion for a "good boundary" is a weighted sum of high cumulative edge strength and low cumulative curvature; that is, for an n-segment curve,

$$h(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{k=1}^{n} s(\mathbf{x}_k) + \alpha \sum_{k=1}^{n-1} q(\mathbf{x}_k, \mathbf{x}_{k+1}) \tag{4.16}$$

where the implicit constraint is that consecutive $\mathbf{x}_k$'s must be grid neighbors:

$$\|\mathbf{x}_k - \mathbf{x}_{k+1}\| \leqslant \sqrt{2} \tag{4.17}$$

$$q(\mathbf{x}_k, \mathbf{x}_{k+1}) = diff\,[\phi(\mathbf{x}_k), \phi(\mathbf{x}_{k+1})] \tag{4.18}$$

where α is negative. The function g we take to be edge strength, i.e., $g(x) = s(x)$. Notice that this evaluation function is in the form of (4.9) and can be optimized in stages:

$$f_0(\mathbf{x}_1) \equiv 0 \tag{4.19}$$

$$f_1(\mathbf{x}_2) = \max_{x_1} [s(\mathbf{x}_1) + \alpha q(\mathbf{x}_1, \mathbf{x}_2) + f_0(\mathbf{x}_1)] \tag{4.20}$$

$$f_k(\mathbf{x}_{k+1}) = \max_{x_k} [s(\mathbf{x}_k) + \alpha q(\mathbf{x}_k, \mathbf{x}_{k+1}) + f_{k-1}(\mathbf{x}_k)] \tag{4.21}$$

These equations can be put into the following steps:

Algorithm 4.5: Dynamic Programming for Edge Finding

1. Set $k = 1$.
2. Consider only $\mathbf{x}$ such that $s(\mathbf{x}) \geqslant T$. For each of these $\mathbf{x}$, define low-curvature pixels "in front of" the contour direction.
3. Each of these pixels may have a curve emanating from it. For $k = 1$, the curve is one pixel in length. Join the curve to $\mathbf{x}$ that optimizes the left-hand side of the recursion equation.
4. If $k = N$, pick the best f_{N-1} and stop. Otherwise, set $k = k + 1$ and go to step 2.

This algorithm can be generalized to the case of picking a *curve* emanating from $\mathbf{x}$ (that we have already generated): Find the end of that curve, and join the best of three curves emanating from the end of that curve. Figure 4.15 shows this process. The equations for the general case are

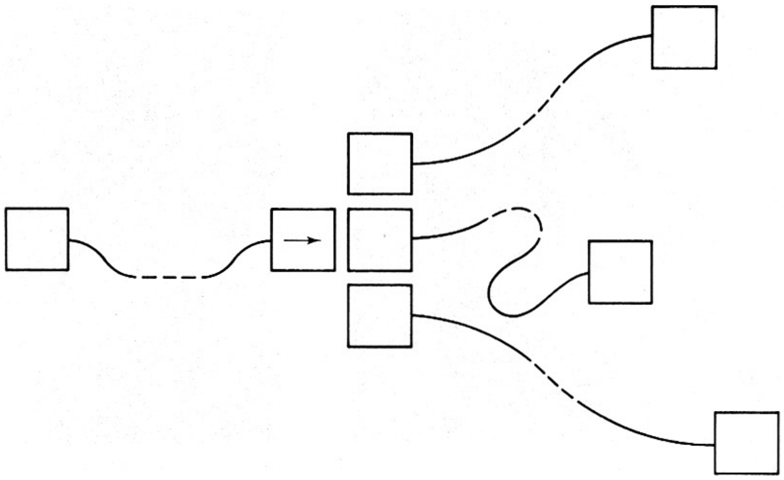


Fig. 4.15 DP optimization for boundary tracing.

$$
\begin{aligned}
f_0(\mathbf{x}_1) &\equiv 0 \\
f_l(\mathbf{x}_{k+1}) &= \max_{x_k}[s(\mathbf{x}_k) + \alpha q(\mathbf{x}_k,\, t(\mathbf{x}_{k+1})) \\
&\qquad + f_{l-1}(\mathbf{x}_k)]
\end{aligned}
\tag{4.22}
$$

where the curve length n is related to α by a building sequence $n(l)$ such that $n(1) = 1$, $n(L) = N$, and $n(l) - n(l-1)$ is a member of $\{n(k) | k = 1, ..., l-1\}$. Also, $t(\mathbf{x}_k)$ is a function that extracts the tail pixel of the curve headed by $\mathbf{x}_k$. Further details may be found in [Ballard 1976].

Results from the area of tumor detection in radiographs give a sense of this method's performance. Here it is known that the boundary inscribes an approximately circular tumor, so that circular cues can be used to assist the search. In Fig. 4.16, (a) shows the image containing the tumor, (b) shows the cues, and (c) shows the boundary found by dynamic programming overlaid on the image.

Another application of dynamic programming may be found in the pseudo-parallel road finder of Barrow [Barrow 1976].

4.5.3 Lower Resolution Evaluation Functions

In the dynamic programming formulation just developed, the components $g(\mathbf{x}_k)$ and $q(\mathbf{x}_k, \mathbf{x}_{k+1})$ in the evaluation function are very localized; the variables $\mathbf{x}$ for successive s and q are in fact constrained to be grid neighbors. This need not be the case: The $\mathbf{x}$ can be very distant from each other without altering the basic technique. Furthermore, the functions g and q need not be local gradient and absolute curvature, respectively, but can be any functions defined on permissible $\mathbf{x}$. This general formulation of the problem for images was first described by [Fischler and

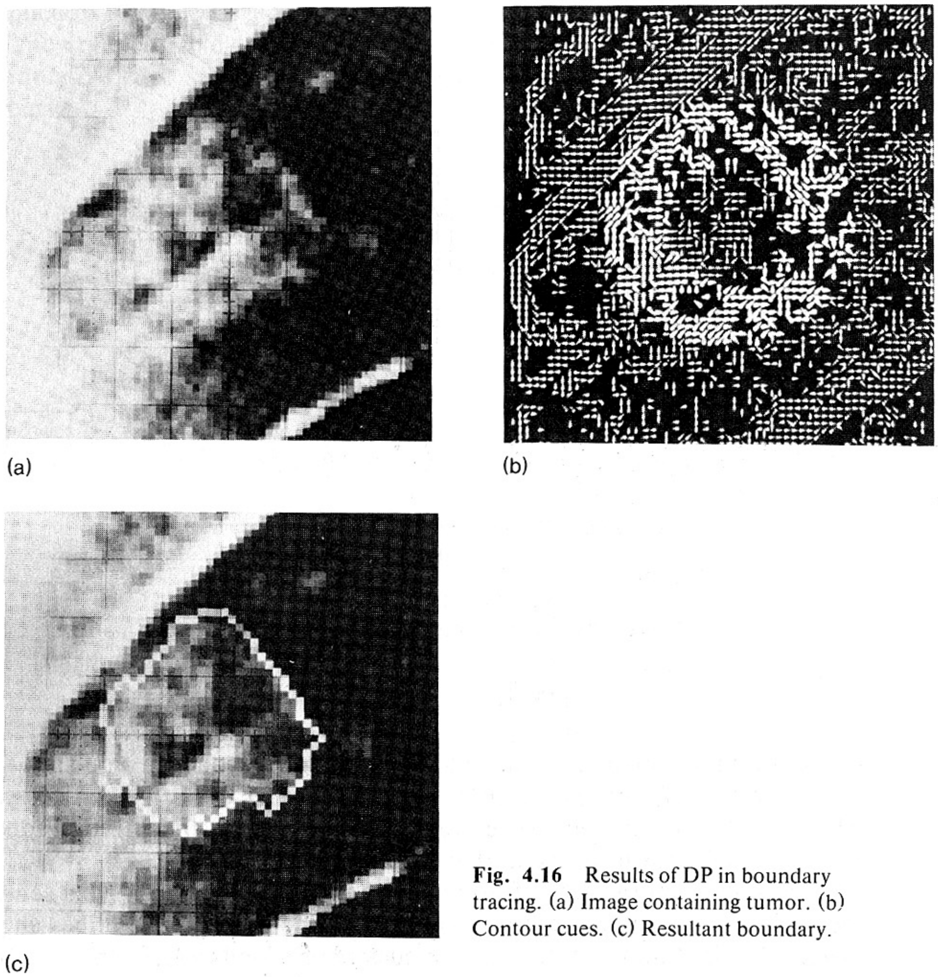


Fig. 4.16 Results of DP in boundary tracing. (a) Image containing tumor. (b) Contour cues. (c) Resultant boundary.

Elschlager 1973]. The Fischler and Elschlager formulation models an object as a set of parts and relations between parts, represented as a graph. Template functions, denoted by $g(\mathbf{x})$, measure how well a part of the model matches a part of the image at the point $\mathbf{x}$. (These local functions may be defined in any manner whatsoever.) "Relational functions," denoted by $q_{kj}(\mathbf{x}, \mathbf{y})$, measure how well the position of the match of the kth part at $(\mathbf{x})$ agrees with the position of the match of the jth part at $(\mathbf{y})$.

The basic notions are shown by a technique simplified from [Chien and Fu 1974] to find the boundaries of lungs in chest films. The lung boundaries are modeled with a polygonal approximation defined by the five key points. These points are the top of the lung, the two clavicle-lung junctions, and the two lower corners. To locate these points, local functions $g(\mathbf{x}_k)$ are defined which should be maximized when the corresponding point $\mathbf{x}_k$ is correctly determined. Similarly, $q(\mathbf{x}_k, \mathbf{x}_j)$ is a function relating points $\mathbf{x}_k$ and $\mathbf{x}_j$. In their case, Chien and Fu used the following functions:

$$T(\mathbf{x}) \equiv \text{template centered at } \mathbf{x} \text{ computed as an aggregate of a set of chest radiographs}$$

$$g(\mathbf{x}_k) = \sum_{\mathbf{x}} \frac{T(\mathbf{x} - \mathbf{x}_k) f(x)}{|T||f|}$$

and

$$\theta(\mathbf{x}_k, \mathbf{x}_j) = \text{expected angular orientation of } \mathbf{x}_k \text{ from } \mathbf{x}_j$$

$$q(\mathbf{x}_k\, \mathbf{x}_j) = \left| \theta(\mathbf{x}_k, \mathbf{x}_j) - \arctan \frac{y_k - y_j}{x_k - x_j} \right|$$

With this formulation no further modifications are necessary and the solution may be obtained by solving Eqs. (4.19) through (4.21), as before. For purposes of comparison, this method was formalized using a lower-resolution objective function. Figure 4.17 shows Chien and Fu's results using this method with five template functions.

4.5.4 Theoretical Questions about Dynamic Programming

The Interaction Graph

This graph describes the interdependence of variables in the objective function. In the examples the interaction graph was simple: Each variable depended on only two others, resulting in the graph of Fig. 4.18a. A more complicated case is the one in 4.18b, which describes an objective function of the following form:

$$h(\,) = h_1(x_1, x_2) + h_2(x_2, x_3, x_4) + h_3(x_{3,} x_4, x_5, x_6)$$

For these cases the dynamic programming technique still applies, but the computational effort increases exponentially with the number of interdependencies. For example, to eliminate x_2 in h_2, all possible combinations of x_3 and x_4 must be considered. To eliminate x_3 in h_3, all possible combinations of x_4, x_5, and x_6, and so forth.

Dynamic Programming versus Heuristic Search

It has been shown [Martelli 1976] that for finding a path in a graph between two points, which is an abstraction of the work we are doing here, heuristic search methods can be more efficient than dynamic programming methods. However, the point to remember about dynamic programming is that it efficiently builds paths from multiple starting points. If this is required by a particular task, then dynamic programming would be the method of choice, unless a very powerful heuristic were available.

4.6 CONTOUR FOLLOWING

If nothing is known about the boundary shape, but regions have been found in the image, the boundary is recovered by one of the simplest edge-following operations: "blob finding" in images. The ideas are easiest to present for binary images:

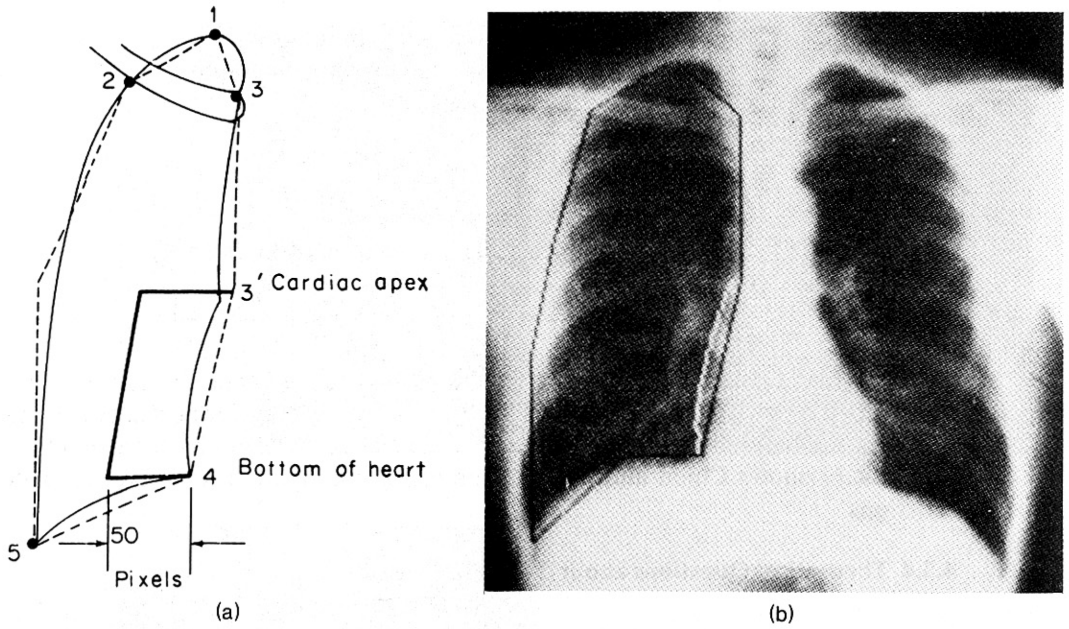


Fig. 4.17 Results of using local templates and global relations. (a) Model. (b) Results.

Given a binary image, the goal is find the boundaries of all distinct regions in the image.

This can be done simply by a procedure that functions like Papert's turtle [Papert 1973; Duda and Hart 1973]:

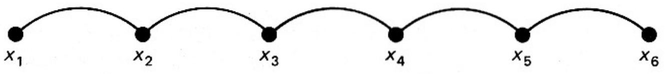
1. Scan the image until a region pixel is encountered.
2. If it is a region pixel, turn left and step; else, turn right and step.
3. Terminate upon return to the starting pixel.

Figure 4.19 shows the path traced out by the procedure. This procedure requires the region to be four-connected for a consistent boundary. Parts of an eight-connected region can be missed. Also, some bookkeeping is necessary to generate an exact sequence of boundary pixels without duplications.

A slightly more elaborate algorithm due to [Rosenfeld 1968] generates the boundary pixels exactly. It works by first finding a four-connected background pixel from a known boundary pixel. The next boundary pixel is the first pixel encountered when the eight neighbors are examined in a counter clockwise order from the background pixel. Many details have to be introduced into algorithms that follow contours of irregular eight-connected figures. A good exposition of these is given in [Rosenfeld and Kak 1976].

4.6.1 Extension to Gray-Level Images

The main idea behind contour following is to start with a point that is believed to be on the boundary and to keep extending the boundary by adding points in the contour directions. The details of these operations vary from task to task. The gen-



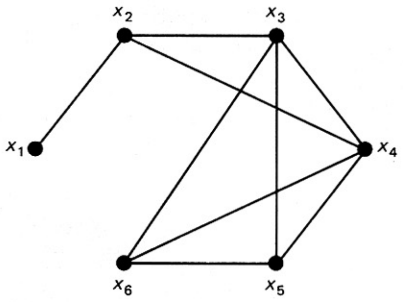


Fig. 4.18 Interaction graphs for DP (see text).

eralization of the contour follower to gray-level images uses local gradients with a magnitude $s(\mathbf{x})$ and direction $\phi(\mathbf{x})$ associated with each point $\mathbf{x}$. ϕ points in the direction of maximum change. If $\mathbf{x}$ is on the boundary of an image object, neighboring points on the boundary should be in the general direction of the contour directions, $\phi(\mathbf{x}) \pm \pi/2$, as shown by Fig. 4.20. A representative procedure is adapted from [Martelli 1976]:

1. Assume that an edge has been detected up to a point $\mathbf{x}_i$. Move to the point $\mathbf{x}_j$ adjacent to $\mathbf{x}_i$ in the direction perpendicular to the gradient of $\mathbf{x}_i$. Apply the gradient operator to $\mathbf{x}_j$; if its magnitude is greater than (some) threshold, this point is added to the edge.
2. Otherwise, compute the average gray level of the 3×3 array centered on $\mathbf{x}_j$, compare it with a suitably chosen threshold, and determine whether $\mathbf{x}_j$ is inside or outside the object.
3. Make another attempt with a point $\mathbf{x}_k$ adjacent to $\mathbf{x}_i$ in the direction perpendicular to the gradient at $\mathbf{x}_i$ plus or minus $(\pi/4)$, according to the outcome of the previous test.

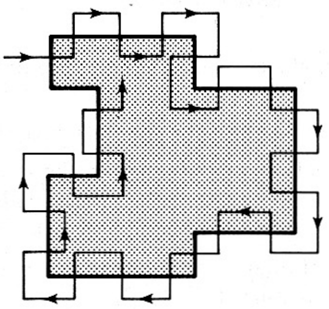


Fig. 4.19 Finding the boundary in a binary image.

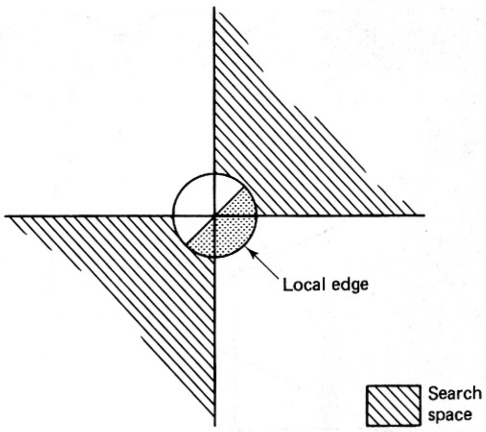


Fig. 4.20 Angular orientations for contour following.

4.6.2 Generalization to Higher-Dimensional Image Data

The generalization of contour following to higher-dimensional spaces is straightforward [Liu 1977; Herman and Liu 1978]. The search involved is, in fact, slightly more complex than contour following and is more like the graph searching methods described in Section 4.4. Higher-dimensional image spaces arise when the image has more than two spatial dimensions, is time-varying, or both. In these images the notion of a gradient is the same (a vector describing the maximum gray-level change and its corresponding direction), but the intuitive interpretation of the corresponding edge element may be difficult. In three dimensions, edge elements are primitive surface elements, separating volumes of differing gray level. The objective of contour following is to link together neighboring surface elements with high gradient modulus values and similar orientations into larger boundaries. In four dimensions, "edge elements" are primitive volumes; contour following links neighboring volumes with similar gradients.

The contour following approach works well when there is little noise present and no "spurious" boundaries. Unfortunately, if either of these conditions is present, the contour-following algorithms are generally unsatisfactory; they are easily thwarted by gaps in the data produced by noise, and readily follow spurious boundaries. The methods described earlier in this chapter attempt to overcome these difficulties through more elaborate models of the boundary structure.

EXERCISES

4.1 Specify a heuristic search algorithm that will work with "crack" edges such as those in Fig. 3.12.

4.2 Describe a modification of Algorithm 4.2 to detect parabolae in gray-level images.

4.3 Suppose that a relation $h(x_1, x_6)$ is added to the model described by Fig. 4.18a so that now the interaction graph is cyclical. Show formally how this changes the optimization steps described by Eqs. (4.11) through (4.13).

4.4 Show formally that the Hough technique without gradient direction information is equivalent to template matching (Chapter 3).

4.5 Extend the Hough technique for ellipses described by Eqs. (4.7a) through (4.7d) to ellipses oriented at an arbitrary angle θ to the x axis.

4.6 Show how to use the generalized Hough technique to detect hexagons.

REFERENCES

ASHKAR, G. P. and J. W. MODESTINO. "The contour extraction problem with biomedical applications." *CGIP 7*, 1978, 331–355.

BALLARD, D. H. *Hierarchic detection of tumors in chest radiographs.* Basel: Birkhäuser-Verlag (ISR-16), January 1976.

BALLARD, D. H. "Generalizing the Hough transform to detect arbitrary shapes." *Pattern Recognition 13*, 2, 1981, 111–122.

BALLARD, D. H. and J. SKLANSKY. "A ladder-structured decision tree for recognizing tumors in chest radiographs." *IEEE Trans. Computers 25*, 1976, 503-513.

BALLARD, D. H., M. MARINUCCI, F. PROIETTI-ORLANDI, A. ROSSI-MARI, and L. TENTARI. "Automatic analysis of human haemoglobin fingerprints." *Proc.*, 3rd Meeting, International Society of Haemotology, London, August 1975.

BARROW, H. G. "Interactive aids for cartography and photo interpretation." Semi-Annual Technical Report, AI Center, SRI International, December 1976.

BELLMAN, R. and S. DREYFUS. *Applied Dynamic Programming.* Princeton, NJ: Princeton University Press, 1962.

BOLLES, R. "Verification vision for programmable assembly." *Proc.*, 5th IJCAI, August 1977, 569-575.

CHIEN, Y. P. and K. S. FU. "A decision function method for boundary detection." *CGIP 3*, 2, June 1974, 125-140.

DUDA, R. O. and P. E. HART. "Use of the Hough transformation to detect lines and curves in pictures." *Commun. ACM 15*, 1, January 1972, 11–15.

DUDA, R. O. and P. E. HART. *Pattern Recognition and Scene Analysis.* New York: Wiley, 1973.

FISCHLER, M. A. and R. A. ELSCHLAGER. "The representation and matching of pictoral patterns." *IEEE Trans. Computers 22*, January 1973.

HERMAN, G. T. and H. K. LIU. "Dynamic boundary surface detection." *CGIP 7*, 1978, 130-138.

HOUGH, P. V. C. "Method and means for recognizing complex patterns." U.S. Patent 3,069,654; 1962.

KELLY, M.D. "Edge detection by computer using planning." In *MI6*, 1971.

KIMME, C., D. BALLARD, and J. SKLANSKY. "Finding circles by an array of accumulators." *Commun. ACM 18*, 2, 1975, 120-122.

LANTZ, K. A., C. M. BROWN and D. H. BALLARD. "Model-driven vision using procedure decription: motivation and application to photointerpretation and medical diagnosis." *Proc.*, 22nd International Symp., Society of Photo-optical Instrumentation Engineers, San Diego, CA, August 1978.

LESTER, J. M., H. A. WILLIAMS, B. A. WEINTRAUB, and J. F. BRENNER, "Two graph searching techniques for boundary finding in white blood cell images." *Computers in Biology and Medicine 8*, 1978, 293-308.

LIU, H. K. "Two- and three-dimensional boundary detection." *CGIP 6*, 2, April 1977, 123-134.

MARR, D. "Analyzing natural images; a computational theory of texture vision." Technical Report 334, AI Lab, MIT, June 1975.

MARTELLI, A. "Edge detection using heuristic search methods." *CGIP 1*, 2, August 1972, 169-182.

MARTELLI, A. "An application of heuristic search methods to edge and contour detection." *Commun. ACM 19*, 2, February 1976, 73–83.

MONTANARI, U. "On the optimal detection of curves in noisy pictures." *Commun. ACM 14*, 5, May 1971, 335-345.

NILSSON, N. J. *Problem-Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.

NILSSON, N. J. *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga, 1980.

PAPERT, S. "Uses of technology to enhance education." Technical Report 298, AI Lab, MIT, 1973.

PERSOON, E. "A new edge detection algorithm and its applications in picture processing." *CGIP 5*, 4, December 1976, 425-446.

RAMER, U. "Extraction of line structures from photographs of curved objects." *CGIP 4*, 2, June 1975, 81-103.

ROSENFELD, A. *Picture Processing by Computer.* New York: Academic Press, 1968.

ROSENFELD, A. and A. C. KAK. *Digital Picture Processing.* New York: Academic Press, 1976.

SELFRIDGE, P. G., J. M. S. PREWITT, C. R. DYER, and S. RANADE. "Segmentation algorithms for abdominal computerized tomography scans." *Proc.*, 3rd COMPSAC, November 1979, 571-577.

WECHSLER, H. and J. SKLANSKY. "Finding the rib cage in chest radiographs." *Pattern Recognition 9*, 1977, 21-30.

Region Growing 5

5.1 REGIONS

Chapter 4 concentrated on the linear features (discontinuities of image gray level) that often correspond to object boundaries, interesting surface detail, and so on. The "dual" problem to finding edges around regions of differing gray level is to find the regions themselves. The goal of region growing is to use image characteristics to map individual pixels in an input image to sets of pixels called *regions.* An image region might correspond to a world object or a meaningful part of one.

Of course, very simple procedures will derive a boundary from a connected region of pixels, and conversely can fill a boundary to obtain a region. There are several reasons why both region growing and line finding survive as basic segmentation techniques despite their redundant-seeming nature. Although perfect regions and boundaries are interconvertible, the processing to find them initially differs in character and applicability; besides, perfect edges or regions are not always required for an application. Region-finding and line-finding techniques can cooperate to produce a more reliable segmentation.

The geometric characteristics of regions depend on the domain. Usually, they are considered to be connected two-dimensional areas. Whether regions can be disconnected, non-simply connected (have holes), should have smooth boundaries, and so forth depends on the region-growing technique and the goals of the work. Ultimately, it is often the segmentation goal to partition the entire image into quasi-disjoint regions. That is, regions have no two-dimensional overlaps, and no pixel belongs to the interior of more than one region. However, there is no single definition of region—they may be allowed to overlap, the whole image may not be partitioned, and so forth.

Our discussion of region growers will begin with the most simple kinds and progress to the more complex. The most primitive region growers use only aggregates of properties of local groups of pixels to determine regions. More sophisticated techniques "grow" regions by *merging* more primitive regions. To do this in a structured way requires sophisticated representations of the regions and boundaries. Also, the merging *decisions* can be complex, and can depend on descriptions of the boundary structure separating regions in addition to the region semantics. A good survey of early techniques is [Zucker 1976].

The techniques we consider are:

1. *Local techniques.* Pixels are placed in a region on the basis of their properties or the properties of their close neighbors.
2. *Global techniques.* Pixels are grouped into regions on the basis of the properties of large numbers of pixels distributed throughout the image.
3. *Splitting and merging techniques.* The foregoing techniques are related to individual pixels or sets of pixels. State space techniques merge or split regions using graph structures to represent the regions and boundaries. Both local and global merging and splitting criteria can be used.

The effectiveness of region growing algorithms depends heavily on the application area and input image. If the image is sufficiently simple, say a dark blob on a light background, simple local techniques can be surprisingly effective. However, on very difficult scenes, such as outdoor scenes, even the most sophisticated techniques still may not produce a satisfactory segmentation. In this event, region growing is sometimes used conservatively to preprocess the image for more knowledgeable processes [Hanson and Riseman 1978].

In discussing the specific algorithms, the following definitions will be helpful. Regions R_k are considered to be sets of points with the following properties:

$\mathbf{x}_i$ in a region R is *connected* to $\mathbf{x}_j$ iff there is a sequence $\{\mathbf{x}_i, \ldots, \mathbf{x}_j\}$ such that $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ are connected and all the points are in R. (5.1)

R is a *connected region* if the set of points $\mathbf{x}$ in R has the property that every pair of points is connected. (5.2)

$$I, \text{ the entire image} = \bigcup_{k=1}^{m} R_k \tag{5.3}$$

$$R_i \cap R_j = \phi, \qquad i \neq j \tag{5.4}$$

A set of regions satisfying (5.2) through (5.4) is known as a *partition.* In segmentation algorithms, each region often is a unique, homogeneous area. That is, for some Boolean function $H(R)$ that measures region homogeneity,

$$H(R_k) = \text{true for all } k \tag{5.5}$$

$$H(R_i \cup R_j) = \text{false for } i \neq j \tag{5.6}$$

Note that R_i does not have to be connected. A weaker but still useful criterion is that neighboring regions not be homogeneous.

5.2 A LOCAL TECHNIQUE: BLOB COLORING

The counterpart to the edge tracker for binary images is the blob-coloring algorithm. Given a binary image containing four-connected blobs of 1's on a background of 0's, the objective is to "color each blob"; that is, assign each blob a different label. To do this, scan the image from left to right and top to bottom with a special L-shaped template shown in Fig. 5.1. The coloring algorithm is as follows.

Algorithm 5.1: Blob Coloring

Let the initial color, $k = 1$. Scan the image from left to right and top to bottom.

```
If $f(\mathbf{x}_C) = 0$ then continue
else
    begin

    if $(f(\mathbf{x}_U) = 1$ and $f(\mathbf{x}_L) = 0)$
    then color $(\mathbf{x}_C) :=$ color $(\mathbf{x}_U)$

    if $(f(\mathbf{x}_L) = 1$ and $f(\mathbf{x}_U) = 0)$
    then color $(\mathbf{x}_C) :=$ color $(\mathbf{x}_L)$

    if $(f(\mathbf{x}_L) = 1$ and $f(\mathbf{x}_U) = 1)$
    then begin
        color $(\mathbf{x}_C) :=$ color $(\mathbf{x}_L)$
        color $(\mathbf{x}_L)$ is equivalent to color $(\mathbf{x}_U)$
        end

  comment: two colors are equivalent.

    if $(f(\mathbf{x}_L) = 0$ and $f(\mathbf{x}_U) = 0)$
    then color $(\mathbf{x}_L) := k; k := k + 1$

    comment: new color

    end
```

After one complete scan of the image the color equivalences can be used to assure that each object has only one color. This binary image algorithm can be used as a simple region-grower for gray-level images with the following modifications. If in a

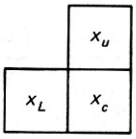


Fig. 5.1 L-shaped template for blob coloring.

gray-level image $f(\mathbf{x}_C)$ is approximately equal to $f(\mathbf{x}_U)$, assign $\mathbf{x}_C$ to the same region (blob) as $\mathbf{x}_U$. This is equivalent to the condition $f(\mathbf{x}_C) = f(\mathbf{x}_U) = 1$ in Algorithm 5.1. The modifications to the steps in the algorithm are straightforward.

5.3 GLOBAL TECHNIQUES: REGION GROWING VIA THRESHOLDING

This approach assumes an object-background image and picks a threshold that divides the image pixels into either object or background:

> $\mathbf{x}$ is part of the Object iff $f(\mathbf{x}) > T$
> Otherwise it is part of the Background

The best way to pick the threshold T is to search the histogram of gray levels, assuming it is bimodal, and find the minimum separating the two peaks, as in Fig. 5.2. Finding the right valley between the peaks of a histogram can be difficult when the histogram is not a smooth function. Smoothing the histogram can help but does not guarantee that the correct minimum can be found. An elegant method for treating bimodal images assumes that the histogram is the sum of two composite normal functions and determines the valley location from the normal parameters [Chow and Kaneko 1972].

The single-threshold method is useful in simple situations, but primitive. For example, the region pixels may not be connected, and further processing such as that described in Chapter 2 may be necessary to smooth region boundaries and remove noise. A common problem with this technique occurs when the image has a background of varying gray level, or when collections we would like to call regions vary smoothly in gray level by more than the threshold. Two modifications of the threshold approach to ameliorate the difficulty are: (1) high-pass filter the image to deemphasize the low-frequency background variation and then try the original technique; and (2) use a spatially varying threshold method such as that of [Chow and Kaneko 1972].

The Chow-Kaneko technique divides the image up into rectangular subimages and computes a threshold for each subimage. A subimage can fail to have a threshold if its gray-level histogram is not bimodal. Such subimages receive inter-

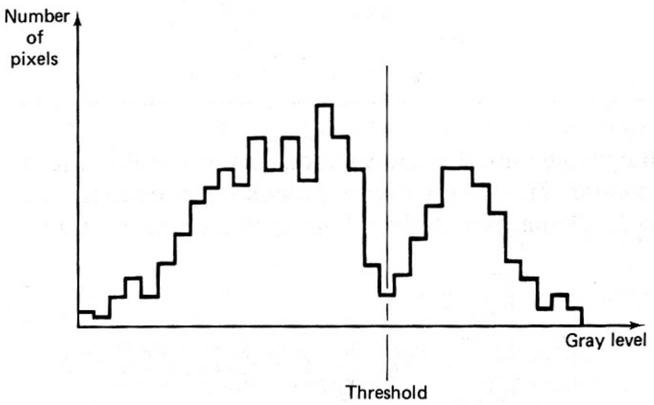


Fig. 5.2 Threshold determination from gray-level histogram.

polated thresholds from neighboring subimages that are bimodal, and finally the entire picture is thresholded by using the separate thresholds for each subimage.

5.3.1 Thresholding in Multidimensional Space

An interesting variation to the basic thresholding paradigm uses color images; the basic digital picture function is vector-valued with red, blue, and green components. This vector is augmented with possibly nonlinear combinations of these values so that the augmented picture vector has a number of components. The idea is to re-represent the color solid redundantly and hope to find color parameters for which thresholding does the desired segmentation. One implementation of this idea used the red, green, and blue color components; the intensity, saturation, and hue components; and the N.T.S.C. *Y*, *I*, *Q* components (Chapter 2) [Ohlander et al. 1979].

The idea of thresholding the components of a picture vector is used in a primitive form for multispectral LANDSAT imagery [Robertson et al. 1973]. The novel extension in this algorithm is the recursive application of this technique to nonrectangular subregions.

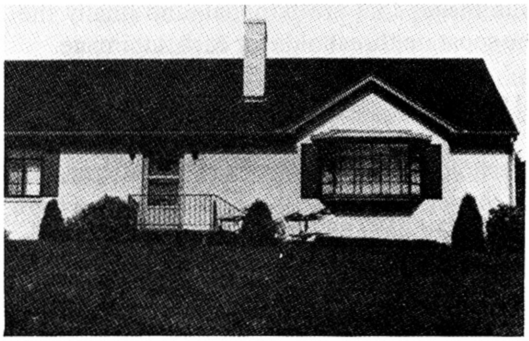
The region partitioning is then as follows:

Algorithm 5.2: Region Growing via Recursive Splitting

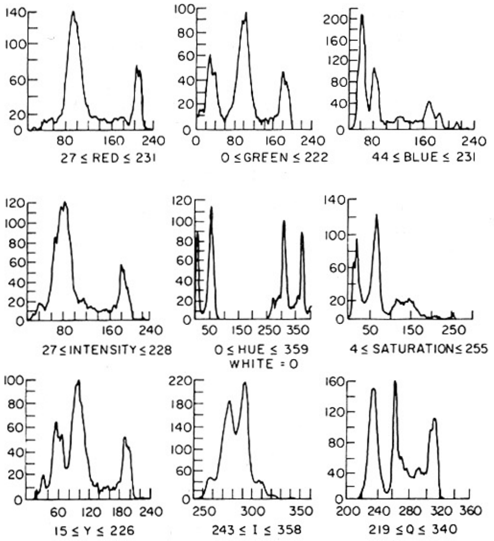
1. Consider the entire image as a region and compute histograms for each of the picture vector components.
2. Apply a peak-finding test to each histogram. If at least one component passes the test, pick the component with the most significant peak and determine two thresholds, one either side of the peak (Fig. 5.3). Use these thresholds to divide the region into subregions.
3. Each subregion may have a "noisy" boundary, so the binary representation of the image achieved by thresholding is smoothed so that only a single connected subregion remains. For binary smoothing see ch. 8 and [Rosenfeld and Kak 1976].
4. Repeat steps 1 through 3 for each subregion until no new subregions are created (no histograms have significant peaks).

A refinement of step 2 of this scheme is to create histograms in higher-dimensional space [Hanson and Riseman 1978]. Multiple regions are often in the same histogram peak when a single measurement is used. The advantage of the multimeasurement histograms is that these different regions are often separated into individual peaks, and hence the segmentation is improved. Figure 5.4 shows some results using a three-dimensional RGB color space.

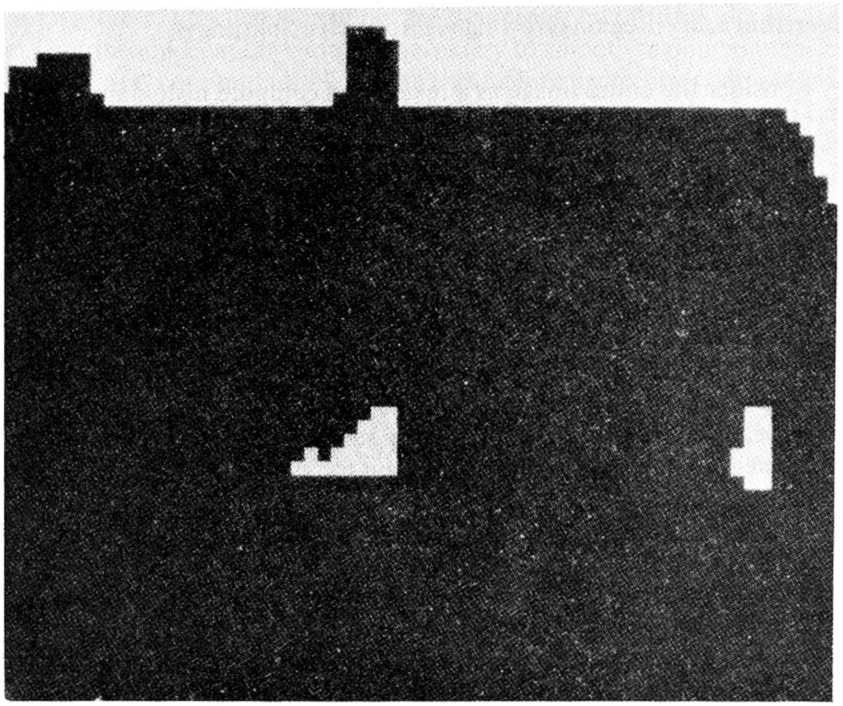
The figure shows the clear separation of peaks in the three-dimensional histogram that is not evident in either of the one-dimensional histograms. How many



(a)



(b)



(c)

Fig. 5.3 Peak detection and threshold determination. (a) Original image. (b) Histograms. (c) Image segments resulting from first histogram peak.

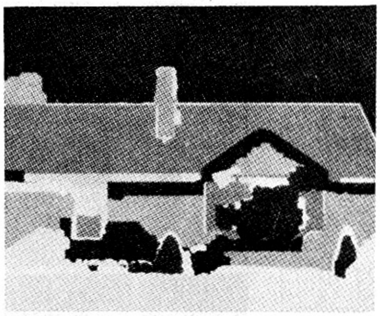

(d)

Fig. 5.3 (d) Final segments.

dimensions should be used? Obviously, there is a trade-off here: As the dimensionality becomes larger, the discrimination improves, but the histograms are more expensive to compute and noise effects may be more pronounced.

5.3.2 Hierarchical Refinement

This technique uses a pyramidal image representation (Section 3.7) [Harlow and Eisenbeis 1973]. Region growing is applied to a coarse resolution image. When the algorithm has terminated at one resolution level, the pixels near the boundaries of regions are disassociated with their regions. The region-growing process is then repeated for just these pixels at a higher-resolution level. Figure 5.5 shows this structure.

5.4 SPLITTING AND MERGING

Given a set of regions R_k, $k = 1,\ldots,m$, a low-level segmentation might require the basic properties described in Section 5.1 to hold. The important properties from the standpoint of segmentation are Eqs. (5.5) and (5.6).

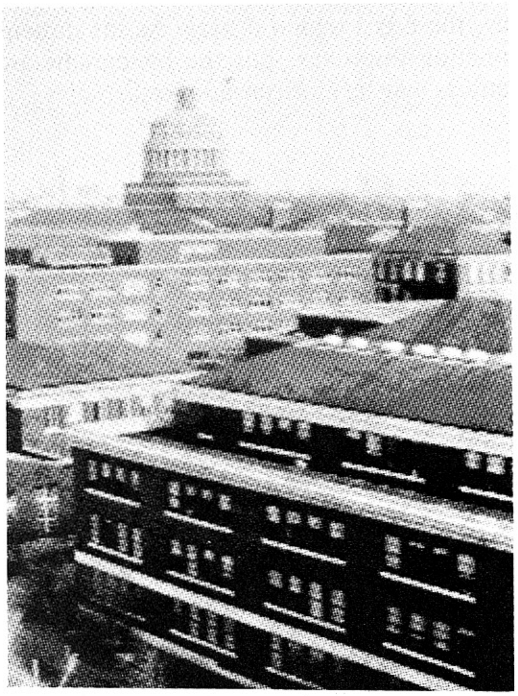
If Eq. (5.5) is not satisfied for some k, it means that that region is inhomogeneous and should be split into subregions. If Eq. (5.6) is not satisfied for some i and j, then regions i and j are collectively homogeneous and should be merged into a single region.

In our previous discussions we used

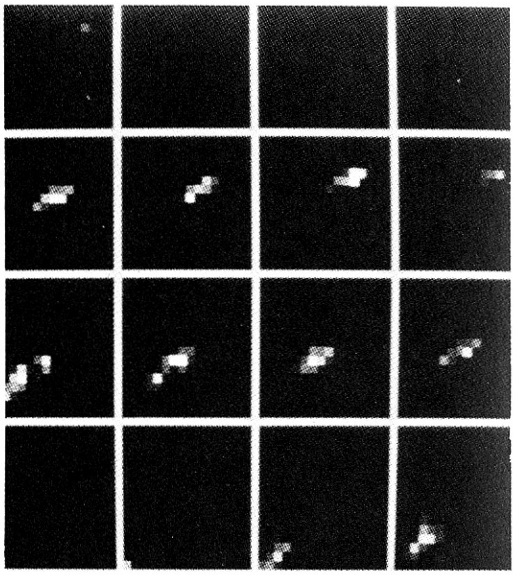
$$H(R) = \begin{cases} \text{true} & \text{if all neighboring pairs of points} \\ & \text{in } R \text{ are such that } f(\mathbf{x}) - f(\mathbf{y}) < T \\ \text{false} & \text{otherwise} \end{cases} \tag{5.7}$$

and

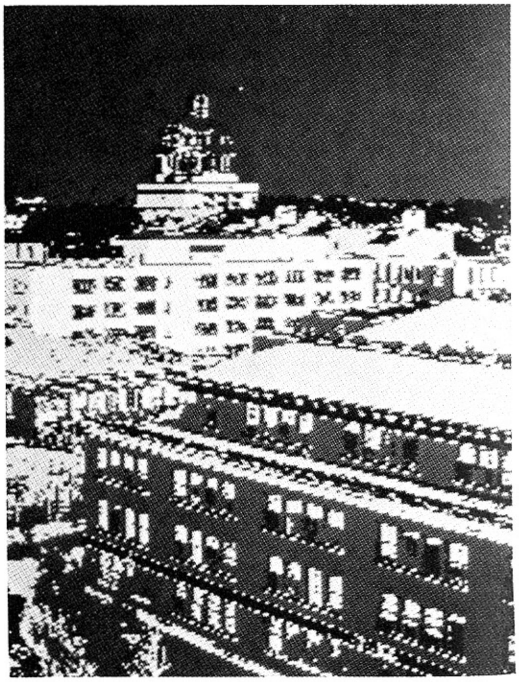
$$H(R) = \begin{cases} \text{true} & \text{if the points in } R \text{ pass a} \\ & \text{bimodality or peak test} \\ \text{false} & \text{otherwise} \end{cases} \tag{5.8}$$



(a)



(b)



(c)

Fig. 5.4 Multi-dimensional histograms in segmentation. (a) Image. (b) RGB histogram showing successive planes through a 16 × 16 × 16 color space. (c) Segments. (See color inserts.)

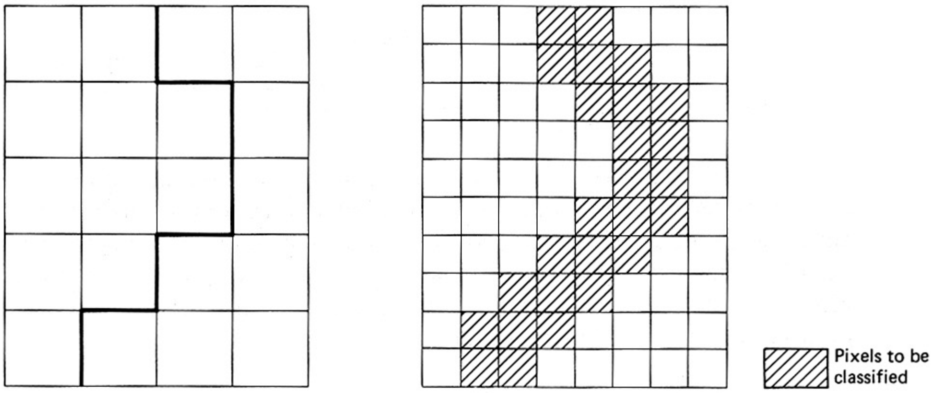


Fig. 5.5 Hierarchical region refinement.

A way of working toward the satisfaction of these homogeneity criteria is the split-and-merge algorithm [Horowitz and Pavlidis 1974]. To use the algorithm it is necessary to organize the image pixels into a pyramidal grid structure of regions. In this grid structure, regions are organized into groups of four. Any region can be split into four subregions (except a region consisting of only one pixel), and the appropriate groups of four can be merged into a single larger region. This structure is incorporated into the following region-growing algorithm.

Algorithm 5.3: Region Growing via Split and Merge [Horowitz and Pavlidis 1974]

1. Pick any grid structure, and homogeneity property H. If for any region R in that structure, $H(R) =$ false, split that region into four subregions. If for any four appropriate regions $R_{k1}, \ldots, R_{k4}$, $H(R_{k1} \bigcup R_{k2} \bigcup R_{k3} \bigcup R_{k4}) =$ true, merge them into a single region. When no regions can be further split or merged, stop.
2. If there are any neighboring regions R_i and R_j (perhaps of different sizes) such that $H(R_i \bigcup R_j) =$ true, merge these regions.

5.4.1 State-Space Approach to Region Growing

The "classical" state-space approach of artificial intelligence [Nilsson 1971, 1980] was first applied to region growing in [Brice and Fennema 1970] and significantly extended in [Feldman and Yakimovsky 1974]. This approach regards the initial two-dimensional image as a discrete state, where every sample point is a separate region. Changes of state occur when a boundary between regions is either removed or inserted. The problem then becomes one of searching allowable changes in state to find the best partition.

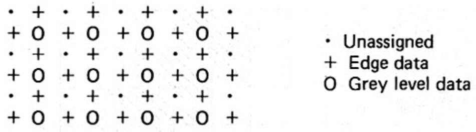


Fig. 5.6 Grid structure for region representation [Brice and Fennema 1970].

An important part of the state-space approach is the use of data structures to allow regions and boundaries to be manipulated as units. This moves away from earlier techniques, which labeled each individual pixel according to its region. The high-level data structures do away with this expensive practice by representing regions with their boundaries and then keeping track of what happens to these boundaries during split-and merge-operations.

5.4.2 Low-level Boundary Data Structures

A useful representation for boundaries allows the splitting and merging of regions to proceed in a simple manner [Brice and Fennema 1970]. This representation introduces the notion of a supergrid S to the image grid G. These grids are shown in Fig. 5.6, where · and + correspond to supergrid and O to the subgrid. The representation is assumed to be four-connected (i.e., **x**1 is a neighbor of **x**2 if $\|\mathbf{x}1 - \mathbf{x}2\| \leqslant 1$).

With this notation boundaries of regions are directed crack edges (see Sec. 3.1) at the points marked +. That is, if point $\mathbf{x}_k$ is a neighbor of $\mathbf{x}_j$ and $\mathbf{x}_k$ is in a different region than $\mathbf{x}_j$, insert two edges for the boundaries of the regions containing $\mathbf{x}_j$ and $\mathbf{x}_k$ at the point + separating them, such that each edge traverses its associated region in a counterclockwise sense. This makes merge operations very simple: To merge regions R_k and R_l, remove edges of the opposite sense from the boundary as shown in Fig. 5.7a. Similarly, to split a region along a line, insert edges of the opposite sense in nearby points, as shown in Fig. 5.7b.

The method of [Brice and Fennema 1970] uses three criteria for merging regions, reflecting a transition from local measurements to global measurements. These criteria use measures of boundary strength s_{ij} and w_{ij} defined as

$$s_{ij} = |f(\mathbf{x}_i) - f(\mathbf{x}_j)| \tag{5.9}$$

$$w_k = \begin{cases} 1 & \text{if } s_k < T_1 \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

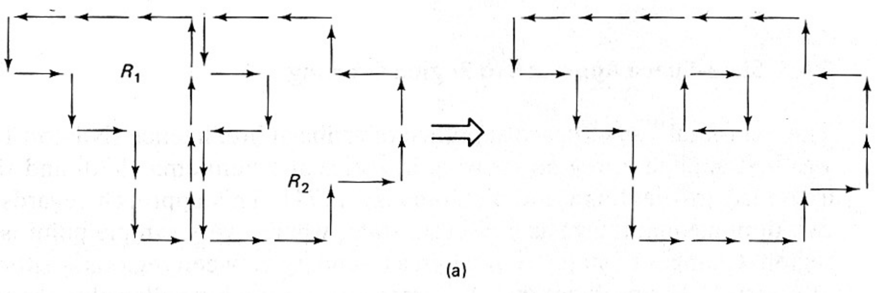


Fig. 5.7 Region operations on the grid structure of Fig. 5.6.

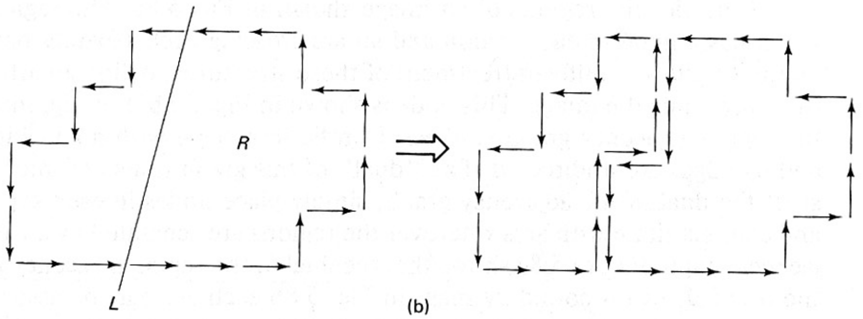


Fig. 5.7 (cont.)

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are assumed to be on either side of a crack edge (Chapter 3). The three criteria are applied sequentially in the following algorithm:

Algorithm 5.4: Region Growing via Boundary Melting (T_k, $k = 1, 2, 3$ are preset thresholds)

1. For all neighboring pairs of points, remove the boundary between $\mathbf{x}_i$ and $\mathbf{x}_j$ if $i \neq j$ and $w_{ij} = 1$. When no more boundaries can be removed, go to step 2.
2. Remove the boundary between R_i and R_j if

$$\frac{W}{\min [p_i, p_j]} \geqslant T_2 \tag{5.11}$$

   where W is the sum of the w_{ij} on the common boundary between R_i and R_j, that have perimeters p_i and p_j respectively. When no more boundaries can be removed, go to step 3.
3. Remove the boundary between R_i and R_j if

$$W \geqslant T_3 \tag{5.12}$$

5.4.3 Graph-Oriented Region Structures

The Brice–Fennema data structure stores boundaries explicitly but does not provide for explicit representation of regions. This is a drawback when regions must be referred to as units. An adjunct scheme of region representation can be developed using graph theory. This scheme represents both regions and their boundaries explicitly, and this facilitates the storing and indexing of their semantic properties.

The scheme is based on a special graph called the *region adjacency graph*, and its "dual graph." In the region adjacency graph, nodes are regions and arcs exist between neighboring regions. This scheme is useful as a way of keeping track of regions, even when they are inscribed on arbitrary nonplanar surfaces (Chapter 9).

Consider the regions of an image shown in Fig. 5.8a. The region adjacency graph has a node in each region and an arc crossing each separate boundary segment. To allow a uniform treatment of these structures, define an artificial region that surrounds the image. This node is shown in Fig. 5.8b. For regions on a plane, the region adjacency graph is *planar* (can lie in a plane with no arcs intersecting) and its edges are undirected. The "dual" of this graph is also of interest. To constuct the dual of the adjacency graph, simply place nodes in each separate region and connect them with arcs wherever the regions are separated by an arc in the adjacency graph. Figure 5.8c shows that the dual of the region adjacency graph is like the original region boundary map; in Fig. 5.8b each arc may be associated with a specific boundary segment and each node with a junction between three or more boundary segments. By maintaining both the region adjacency graph and its dual, one can merge regions using the following algorithm:

Algorithm 5.5: Merging Using the Region-Adjacency Graph and Its Dual

Task: Merge neighboring regions R_i and R_j.

Phase 1. Update the region-adjacency graph.

1. Place edges between R_i and all neighboring regions of R_j (excluding, of course, R_i) that do not already have edges between themselves and R_i.

2. Delete R_j and all its associated edges.

Phase 2. Take care of the dual.

1. Delete the edges in the dual corresponding to the borders between R_i and R_j.

2. For each of the nodes associated with these edges:

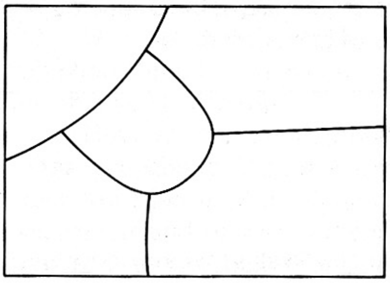
  (a) if the resultant degree of the node is less than or equal to 2, delete the node and join the two dangling edges into a single edge.

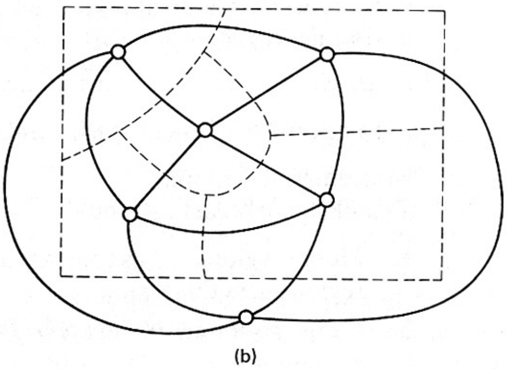
  (b) otherwise, update the labels of the edges that were associated with j to reflect the new region label i.

Figure 5.9 shows these operations.

5.5 INCORPORATION OF SEMANTICS

Up to this point in our treatment of region growers, domain-dependent "semantics" has not explicitly appeared. In other words, region-merging decisions were based on raw image data and rather weak heuristics of general applicability about the likely shape of boundaries. As in early processing, the use of domain-dependent knowledge can affect region finding. Possible interpretations of regions can affect the splitting and merging process. For example, in an outdoor scene possible region interpretations might be sky, grass, or car. This kind of knowledge is quite separate from but related to measurable region properties such as intensity


(a)


(b)

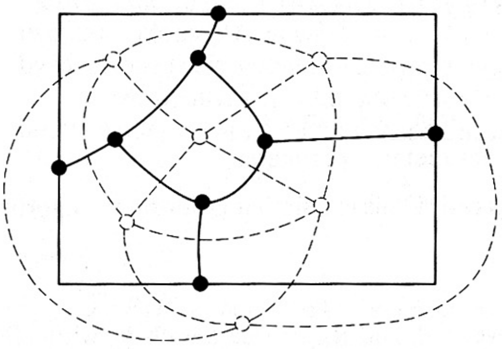
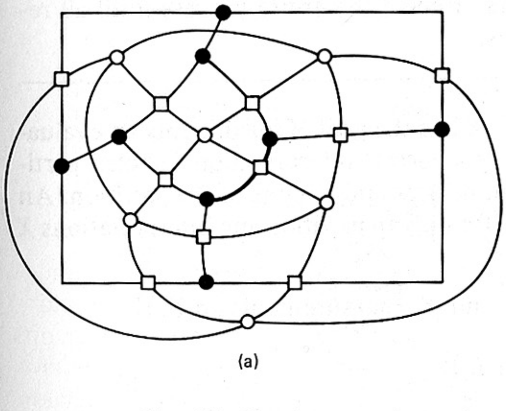

(c)

Fig. 5.8 (a) An image partition. (b) The region adjacency graph (solid lines). (c) The dual of the adjacency graph (solid lines).

and hue. An example shows how semantic labels for regions can guide the merging process. This approach was originally developed in [Feldman and Yakimovsky 1974]. it has found application in several complex vision systems [Barrow and Tenenbaum 1977; Hanson and Riseman 1978].

Early steps in the Feldman–Yakimovsky region grower used essentially the same steps as Brice–Fennema. Once regions attain significant size, semantic cri-


(a)

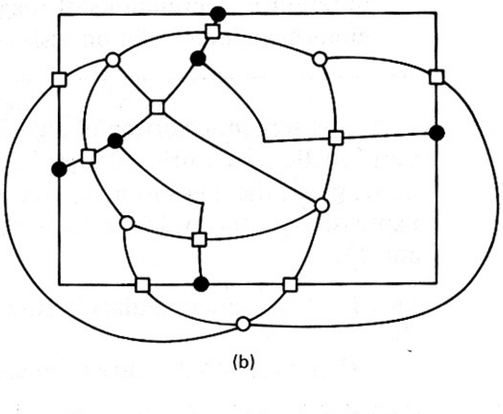

(b)

Fig. 5.9 Merging operations using the region adjacency graph and its dual. (a) Before merging regions separated by dark boundary line. (b) After merging.

teria are used. The region growing consists of four steps, as summed up in the following algorithm:

Algorithm 5.6 Semantic Region Growing

Nonsemantic Criteria
T_1 and T_2 are preset thresholds

1. Merge regions i, j as long as they have one weak separating edge until no two regions pass this test.
2. Merge regions i, j where $S(i, j) \leqslant T_2$ where

$$S(i, j) = \frac{c_1 + \alpha_{ij}}{c_2 + \alpha_{ij}}$$

   where c_1 and c_2 are constants,

$$\alpha ij = \frac{(\text{area}_i)^{1/2} + (\text{area}_j)^{1/2}}{\text{perimeter}_i \cdot \text{perimeter}_j}$$

   until no two regions pass this test. (This is a similar criterion to Algorithm 5.4, step 2.)

Semantic Criteria

3. Let B_{ij} be the boundary between R_i and R_j. Evaluate each B_{ij} with a Bayesian decision function that measures the (conditional) probability that B_{ij} separates two regions R_i and R_j of the *same interpretation*. Merge R_i and R_j if this conditional probability is less than some threshold. Repeat step 3 until no regions pass the threshold test.
4. Evaluate the interpretation of each region R_i with a Bayesian decision function that measures the (conditional) probability that an interpretation is the correct one for that region. Assign the interpretation to the region with the highest confidence of correct interpretation. Update the conditional probabilities for different interpretations of neighbors. Repeat the entire process until all regions have interpretation assignments.

The semantic portion of algorithm 5.6 had the goal of maximizing an evaluation function measuring the probability of a correct interpretation (labeled partition), given the measurements on the boundaries and regions of the partition. An expression for the evaluation function is (for a given partition and interpretations X and Y):

$$\max_{X,\,Y} \prod_{i,\,j} \{P[B_{ij} \text{ is a boundary between } X \text{ and } Y \mid \text{measurements on } B_{ij}]\}$$

$$\times \prod_{i} \{P[R_i \text{ is an } X \mid \text{measurements on } R_i]\}$$

$$\times \prod_{i} \{P[R_j \text{ is an } Y \mid \text{measurements on } R_j]\}$$

where P stands for probability and Π is the product operator.

How are these terms to be computed? Ideally, each conditional probability function should be known to a reasonable degree of accuracy; then the terms can be obtained by lookup.

However, the straightforward computation and representation of the conditional probability functions requires a massive amount of work and storage. An approximation used in [Feldman and Yakimovsky 1974] is to quantize the measurements and represent them in terms of a classification tree. The conditional probabilities can then be computed from data at the leaves of the tree. Figure 5.10 shows a hypothetical tree for the region measurements of intensity and hue, and interpretations ROAD, SKY, and CAR. Figure 5.11 shows the equivalent tree for two boundary measurements m and n and the same interpretations. These two figures indicate that $P[R_i \text{ is a CAR} \,|\, 0 \leqslant i < I, 0 \leqslant h < H_1] =$, and $P[B_{ij} \text{ divides two car regions} \,|\, M_k \leqslant m < M_{k+1}, N_l < n \leqslant N_{l+1} =$. These trees were created by laborious trials with correct segmentations of test images.

Now, finally, consider again step 3 of Algorithm 5.6. The probability that a boundary B_{ij} between regions R_i and R_j is false is given by

$$P_{\text{false}} = \frac{P_f}{P_t + P_f} \tag{5.13}$$

where

$$P_f = \sum \{P[B_{ij} \text{ is between two subregions } X \,|\, B_{ij}\text{'s measurements}]\} \times \{P[R_i \text{ is } X \,|\, \text{meas}]\} \times \{P[R_j \text{ is } X \,|\, \text{meas}]\} \tag{5.14a}$$

$$P_t = \sum_{x,y} \{P[B_{ij} \text{ is between } X \text{ and } Y \,|\, \text{meas}]\} \times \{P[R_i \text{ is } X \,|\, \text{meas}]\} \times \{P[R_j \text{ is } Y \,|\, \text{meas}]\} \tag{5.14b}$$

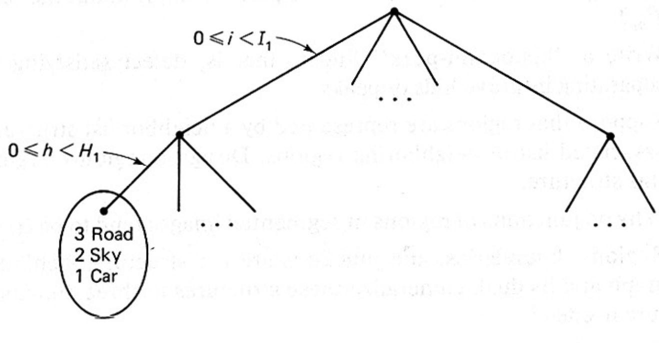


Fig. 5.10 Hypothetical classification tree for region measurements showing a particular branch for specific ranges of intensity and hue.

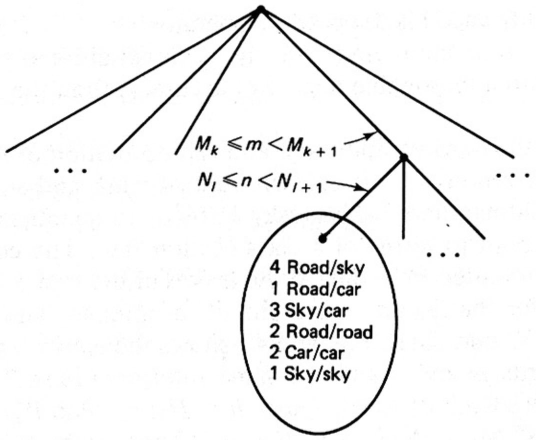


Fig. 5.11 Hypothetical classification tree for boundary measurements showing a specific branch for specific ranges of two measurements m and n.

And for step 4 of the algorithm,

$$\text{Confidence}_i = \frac{P[R_i \text{ is } X1 \mid \text{meas}]}{P[R_i \text{ is } X2 \mid \text{meas}]} \tag{5.15}$$

where $X1$, $X2$ are the first and second most likely interpretations, respectively. After the region is assigned interpretation $X1$, the neighbors are updated using

$$P[R_i \text{ is } X \mid \text{meas}] := Prob\ [Rj \text{ is } X \mid \text{meas}] \times P[B_{ij} \text{ is between } X \text{ and } X1 \mid \text{meas}] \tag{5.16}$$

EXERCISES

5.1 In Algorithm 5.1, show how one can handle the case where colors are equivalent. Do you need more than one pass over the image?

5.2 Show for the heuristic of Eq. (5.11) that

(a) $IT_2 \geqslant WT_2 > P_j$

(b) $P_m < P_i + I(1/T_2 - 2)$

where P_m is the perimeter of $R_i \bigcup R_j$, I is the perimeter common to both i and j and $P_m = \min(P_i, P_j)$. What does part (b) imply about the relation between T_2 and P_m?

5.3 Write a "histogram-peak" finder; that is, detect satisfying valleys in histograms separating intuitive hills or peaks.

5.4 Suppose that regions are represented by a neighbor list structure. Each region has an associated list of neighboring regions. Design a region-merging algorithm based on this structure.

5.5 Why do junctions of regions in segmented images tend to be trihedral?

5.6 Regions, boundaries, and junctions are the structures behind the region-adjacency graph and its dual. Generalize these structures to three dimensions. Is another structure needed?

5.7 Generalize the graph of Figure 5.8 to three dimensions and develop the merging algorithm analogous to Algorithm 5.5. (Hint: see Exercise 5.6.)

REFERENCES

BARROW, H. G. and J. M. TENENBAUM. "Experiments in model-driven scene segmentation." *Artificial Intelligence 8*, 3, June 1977, 241–274.

BRICE, C. and C. FENNEMA. "Scene analysis using regions." *Artificial Intelligence 1*, 3, Fall 1970, 205–226.

CHOW, C. K. and T. KANEKO. "Automatic boundary detection of the left ventricle from cineangiograms." *Computers and Biomedical Research 5*, 4, August 1972, 388–410.

FELDMAN, J. A. and Y. YAKIMOVSKY. "Decision theory and artificial intelligence: I. A semantics-based region analyzer." *Artificial Intelligence 5*, 4, 1974, 349–371.

HANSON, A. R. and E. M. RISEMAN. "Segmentation of natural scenes." In *CVS*, 1978.

HARLOW, C. A. and S. A. EISENBEIS. "The analysis of radiographic images." *IEEE Trans. Computers 22*, 1973, 678–688.

HOROWITZ, S. L. and T. PAVLIDIS. "Picture segmentation by a directed split-and-merge procedure." *Proc.*, 2nd IJCPR, August 1974, 424–433.

NILSSON, N. J. *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga, 1980.

NILSSON, N. J. *Problem-Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.

OHLANDER, R., K. PRICE, and D. R. REDDY. "Picture segmentation using a recursive region splitting method." *CGIP 8*, 3, December 1979.

ROBERTSON, T. V., P. H. SWAIN, and K. S. FU. "Multispectral image partitioning." TR-EE 73-26 (LARS Information Note 071373), School of Electrical Engineering, Purdue Univ., August 1973.

ROSENFELD, A. and A. C. KAK. *Digital Picture Processing.* New York: Academic Press, 1976.

ZUCKER, S. W. "Region growing: Childhood and adolescence." *CGIP 5*, 3, September 1976, 382–399.

Texture 6

6.1 WHAT IS TEXTURE?

The notion of texture admits to no rigid description, but a dictionary definition of texture as "something composed of closely interwoven elements" is fairly apt. The description of interwoven elements is intimately tied to the idea of texture resolution, which one might think of as the average amount of pixels for each discernable texture element. If this number is large, we can attempt to describe the individual elements in some detail. However, as this number nears unity it becomes increasingly difficult to characterize these elements individually and they merge into less distinct spatial patterns. To see this variability, we examine some textures.

Figure 6.1 shows "cane," "paper," "coffee beans," "brickwall," "coins," and "wire braid" after Brodatz's well-known book [Brodatz 1966]. Five of these examples are high-resolution textures: they show repeated primitive elements that exhibit some kind of variation. "Coffee beans," "brick wall" and "coins" all have obvious primitives (even if it is not so obvious how to extract these from image data). Two more examples further illustrate that one sometimes has to be creative in defining primitives. In "cane" the easiest primitives to deal with seem to be the physical holes in the texture, whereas in "wire braid" it might be better to model the physical relations of a loose weave of metallic wires. However, the paper texture does not fit nicely into this mold. This is not to say that there are not possibilities for primitive elements. One is regions of lightness and darkness formed by the ridges in the paper. A second possibility is to use the reflectance models described in Section 3.5 to compute "pits" and "bumps." However, the elements seem to be "just beyond our perceptual resolving power" [Laws 1980], or in our terms, the elements are very close in size to individual pixels.

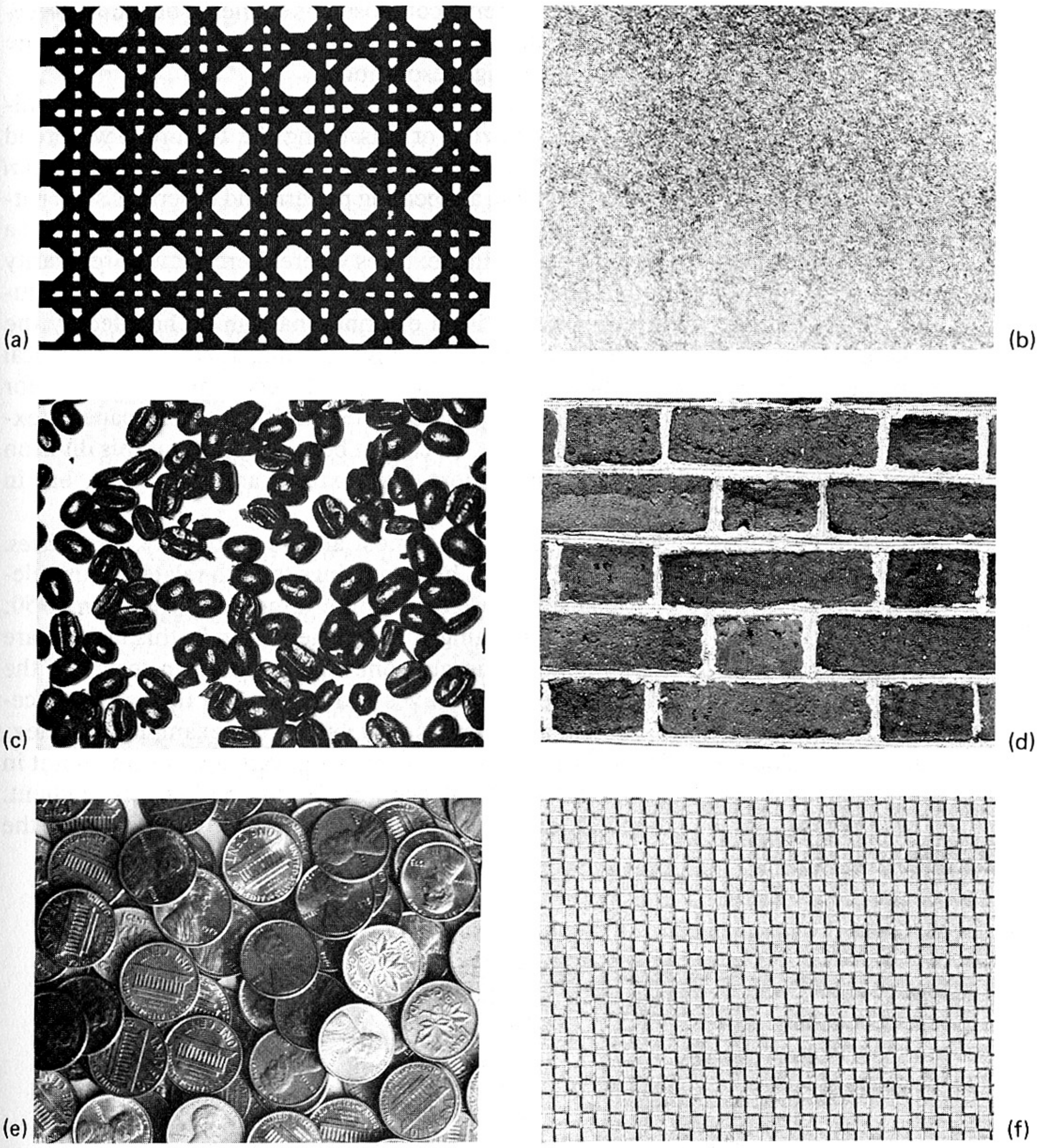


Fig. 6.1 Six examples of texture. (a) Cane. (b) Paper. (c) Coffee beans. (d) Brick wall. (e) Coins. (f) Wire braid.

The exposition of texture takes place under four main headings:

1. Texture primitives
2. Structural models
3. Statistical models
4. Texture gradients

We have already described texture as being composed of elements of *texture primitives.* The main point of additional discussion on texture primitives is to refine the idea of a primitive and its relation to image resolution.

The main work that is unique to texture is that which describes how primitives are related to the aim of recognizing or classifying the texture. Two broad classes of techniques have emerged and we shall study each in turn. The *structural* model regards the primitives as forming a repeating pattern and describes such patterns in terms of rules for generating them. Formally, these rules can be termed a grammar. This model is best for describing textures where there is much regularity in the placement of primitive elements and the texture is imaged at high resolution. The "reptile" texture in Fig. 6.9 is an example that can be handled by the structured approach. The *statistical* model usually describes texture by statistical rules governing the distribution and relation of gray levels. This works well for many natural textures which have barely discernible primitives. The "paper" texture is such an example. As we shall see, we cannot be too rigid about this division since statistical models can describe pattern-like textures and vice versa, but in general the dichotomy is helpful.

The examples suggest that texture is almost always a property of *surfaces.* Indeed, as the example of Fig. 6.2 shows, human beings tend to relate texture elements of varying size to a plausible surface in three dimensions [Gibson 1950; Stevens 1979]. Techniques for determining surface orientation in this fashion are termed texture *gradient* techniques. The gradient is given both in terms of the direction of greatest change in size of primitives and in terms of the spatial placement of primitives. The notion of a gradient is very useful. For example, if the texture is embedded on a flat surface, the gradient points toward a vanishing point in the image. The chapter concludes with algorithms for computing this gradient. The gradient may be computed directly or indirectly via the computation of the vanishing point.

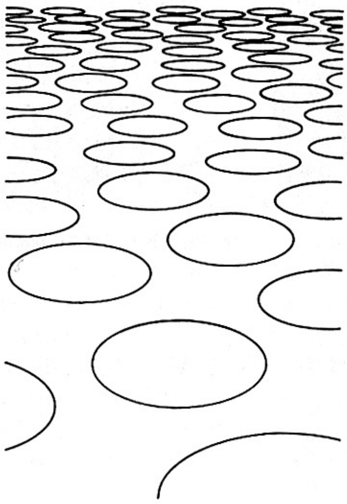


Fig. 6.2 Texture as a surface property.

6.2 TEXTURE PRIMITIVES

The notion of a primitive is central to texture. To highlight its importance, we shall use the appelation *texel* (for texture element) [Kender 1978]. A texel is (loosely) a visual primitive with certain invariant properties which occurs repeatedly in different positions, deformations, and orientations inside a given area. One basic invariant property of such a unit might be that its pixels have a constant gray level, but more elaborate properties related to shape are possible. (A detailed discussion of planar shapes is deferred until Chapter 8.) Figure 6.3 shows examples of two kinds of texels: (a) ellipses of approximately constant gray level and (b) linear edge segments. Interestingly, these are nearly the two features selected as texture primitives by [Julesz, 1981], who has performed extensive studies of human texture perception.

For textures that can be described in two dimensions, image-based descriptions are sufficient. Texture primitives may be pixels, or aggregates of pixels such as curve segments or regions. The "coffee beans" texture can be described by an image-based model: repeated dark ellipses on a lighter background. These models describe equally well an image of texture or an image of a picture of texture. The methods for creating these aggregates were discussed in Chapters 4 and 5. As with all image-based models, three-dimensional phenomena such as occlusion must be handled indirectly. In contrast, structural approaches to texture sometimes require knowledge of the three-dimensional world producing the texture image. One example of this is Brodatz's "coins" shown in Fig. 6.1. A three-dimensional model of the way coins can be stacked is needed to understand this texture fully.

An important part of the texel definition is that primitives must occur repeatedly inside a given area. The question is: How many times? This can be answered qualitatively by imagining a window that corresponds approximately to our field of view superimposed on a very large textured area. As this window is made smaller, corresponding to moving the viewpoint closer to the texture, fewer and fewer texels are contained in it. At some distance, the image in the window no longer

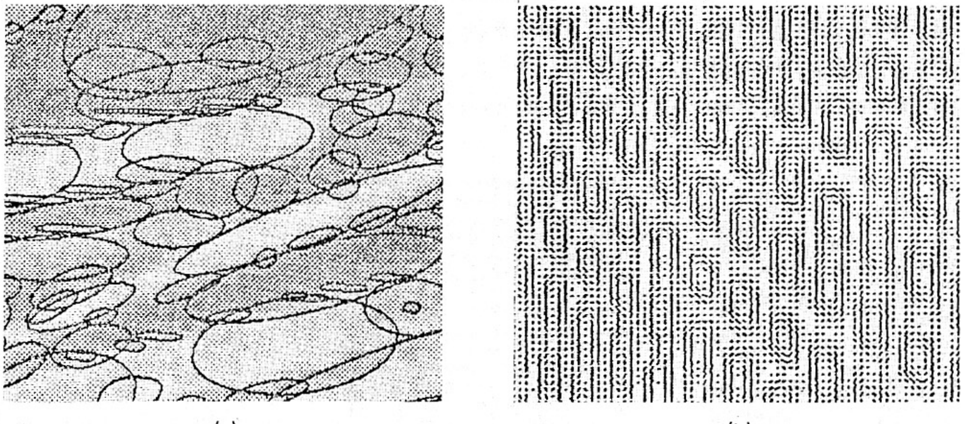


Fig. 6.3 Examples of texels. (a) Ellipses. (b) Linear segments.

appears textured, or if it does, translation of the window changes the perceived texture drastically. At this point we no longer have a texture. A similar effect occurs if the window is made increasingly larger, corresponding to moving the field of view farther away from the image. At some distance textural details are blurred into continuous tones and repeated elements are no longer visible as the window is translated. (This is the basis for halftone images, which are highly textured patterns meant to be viewed from enough distance to blur the texture.) Thus the idea of an appropriate *resolution*, or the number of texels in a subimage, is an implicit part of our qualitative definition of texture. If the resolution is appropriate, the texture will be apparent and will "look the same" as the field of view is translated across the textured area. Most often the appropriate resolution is not known but must be computed. Often this computation is simpler to carry out than detailed computations characterizing the primitives and hence has been used as a precursor to the latter computations. Figure 6.4 shows such a resolution-like computation, which examines the image for repeating peaks [Connors 1979].

Textures can be hierarchical, the hierarchies corresponding to different resolutions. The "brick wall" texture shows such a hierarchy. At one resolution, the highly structured pattern made by collections of bricks is in evidence; at higher resolution, the variations of the texture of each brick are visible.

6.3 STRUCTURAL MODELS OF TEXEL PLACEMENT

Highly patterned textures tesselate the plane in an ordered way, and thus we must understand the different ways in which this can be done. In a regular tesselation the

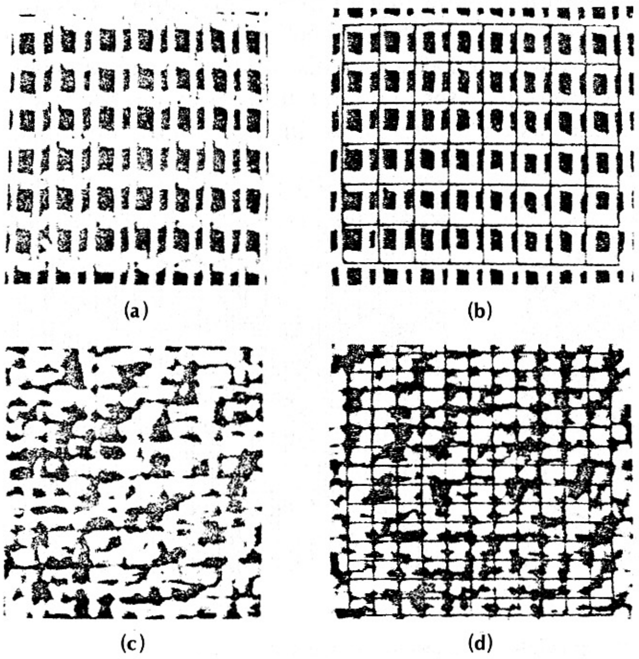
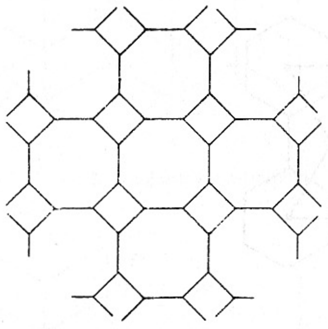
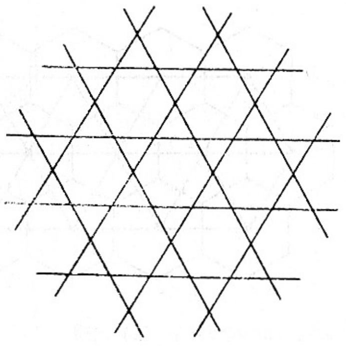
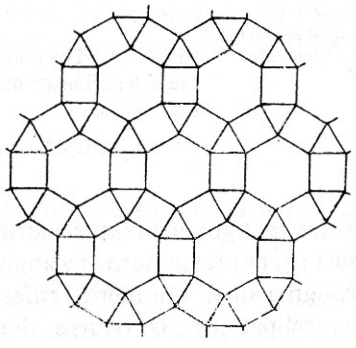


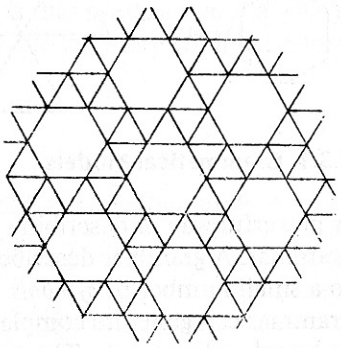
Fig. 6.4 Computing texture resolutions. (a) French canvas. (b) Resolution grid for canvas. (c) Raffia. (d) Grid for raffia.

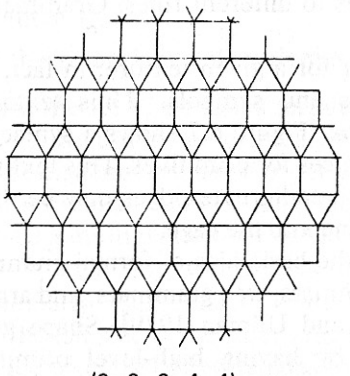
polygons surrounding a vertex all have the same number of sides. Semiregular tesselations have two kinds of polygons (differing in number of sides) surrounding a vertex. Figure 2.11 depicts the regular tesselations of the plane. There are eight semiregular tesselations of the plane, as shown in Fig. 6.5. These tesselations are conveniently described by listing in order the number of sides of the polygons sur-


(4, 8, 8)


(3, 6, 3, 6)


(3, 4, 6, 4)


(3, 3, 3, 3, 6)


(3, 3, 3, 4, 4)

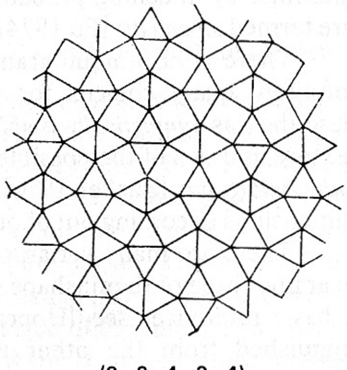

(3, 3, 4, 3, 4)

Fig. 6.5 Semiregular tesselations.

rounding each vertex. Thus a hexagonal tesselation is described by (6,6,6) and every vertex in the tesselation of Fig. 6.5 can be denoted by the list (3,12,12). It is important to note that the tesselations of interest are those which describe the *placement* of primitives rather than the primitives themselves. When the primitives define a tesselation, the tesselation describing the primitive placement will be the dual of this graph in the sense of Section 5.4. Figure 6.6 shows these relationships.

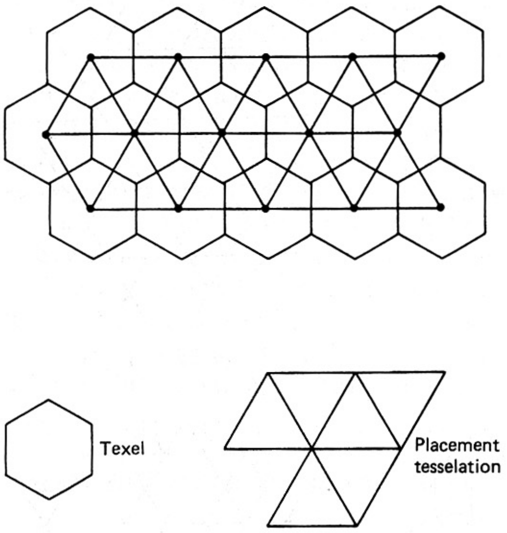


Fig. 6.6 The primitive placement tesselation as the dual of the primitive tesselation.

6.3.1 Grammatical Models

A powerful way of describing the rules that govern textural structure is through a grammar. A grammar describes how to generate patterns by applying *rewriting rules* to a small number of *symbols.* Through a small number of rules and symbols, the grammar can generate complex textural patterns. Of course, the symbols turn out to be related to texels. The mapping between the stored model prototype texture and an image of texture with real-world variations may be incorporated into the grammar by attaching probabilities to different rules. Grammars with such rules are termed *stochastic* [Fu 1974].

There is no unique grammar for a given texture; in fact, there are usually infinitely many choices for rules and symbols. Thus texture grammars are described as *syntactically ambiguous.* Figure 6.7 shows a syntactically ambiguous texture and two of the possible choices for primitives. This texture is also *semantically ambiguous* [Zucker 1976] in that alternate ridges may be thought of in three dimensions as coming out of or going into the page.

There are many variants of the basic idea of formal grammars and we shall examine three of them: shape grammars, tree grammars, and array grammars. For a basic reference, see [Hopcroft and Ullman 1979]. Shape grammars are distinguished from the other two by having high-level primitives that closely correspond to the shapes in the texture. In the examples of tree grammars and array grammars that we examine, texels are defined as pixels and this makes the

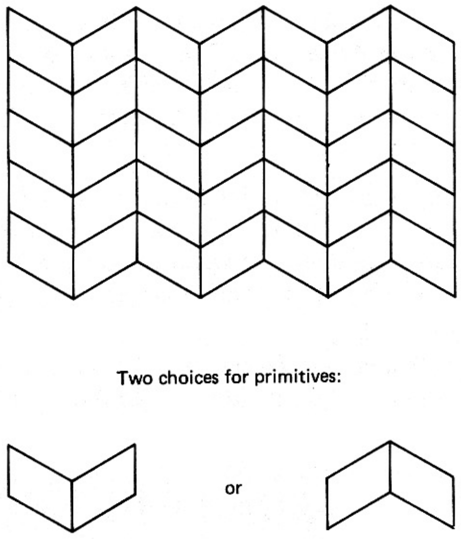


Fig. 6.7 Ambiguous texture.

grammars correspondingly more complicated. A particular texture that can be described in eight rules in a shape grammar requires 85 rules in a tree grammar [Lu and Fu 1978]. The compensating trade-off is that pixels are gratis with the image; considerable processing must be done to derive the more complex primitives used by the shape grammar.

6.3.2 Shape Grammars

A shape grammar [Stiny and Gips 1972] is defined as a four-tuple $< V_t, V_m, R, S>$ where:

1. V_t is a finite set of shapes
2. V_m is a finite set of shapes such that $V_t \cap V_m = \phi$
3. R is a finite set of ordered pairs (u, v) such that u is a shape consisting of elements of V_t^+ and v is a shape consisting of an element of V_t^* combined with an element of V_m^*
4. S is a shape consisting of an element of V_t^* combined with an element of V_m^*.

Elements of the set V_t are called terminal shape elements (or terminals). Elements of the set V_m are called nonterminal shape elements (or markers). The sets V_t and V_m must be disjoint. Elements of the set V_t^+ are formed by the finite arrangement of one or more elements of V_t in which any elements and/or their mirror images may be used a multiple number of times in any location, orientation, or scale. The set $V_t^* = V_t^+ \cup \{\Lambda\}$, where Λ is the empty shape. The sets V_m^+ and V_m^* are defined similarly. Elements (u, v) of R are called shape rules and are written $u\, v$. *u*is called the left side of the rule; *v* the right side of the rule. *u* and *v* usually are enclosed in identical dashed rectangles to show the correspondence between the two shapes. S is called the initial shape and normally contains a u such that there is a (u, v) which is an element of R.

A texture is generated from a shape grammar by beginning with the initial shape and repeatedly applying the shape rules. The result of applying a shape rule R to a given shape s is another shape, consisting of s with the right side of R substituted in S for an occurrence of the left side of R. Rule application to a shape proceeds as follows:

1. Find part of the shape that is geometrically similar to the left side of a rule in terms of both terminal elements and nonterminal elements (markers). There must be a one-to-one correspondence between the terminals and markers in the left side of the rule and the terminals and markers in the part of the shape to which the rule is to be applied.
2. Find the geometric transformations (scale, translation, rotation, mirror image) which make the left side of the rule identical to the corresponding part in the shape.
3. Apply those transformations to the right side of the rule.
4. Substitute the transformed right side of the rule for the part of the shape that corresponds to the left side of the rule.

The generation process is terminated when no rule in the grammar can be applied.

As a simple example, one of the many ways of specifying a hexagonal texture $\{V_t, V_m, R, S\}$ is

$$\begin{aligned} V_t &= \{⬡\} \\ V_m &= \{\ \cdot\ \} \\ R&: ⬡ \rightarrow ⬡⬡\,;\ ⬡⬡\ ;\text{etc.} \\ S &= \{⬡\} \end{aligned} \tag{6.1}$$

Hexagonal textures can be *generated* by the repeated application of the single rule in R. They can be *recognized* by the application of the rule in the opposite direction to a given texture until the initial shape, I, is produced. Of course, the rule will generate only hexagonal textures. Similarly, the hexagonal texture in Fig. 6.8a will be recognized but the variants in Fig. 6.8b will not.

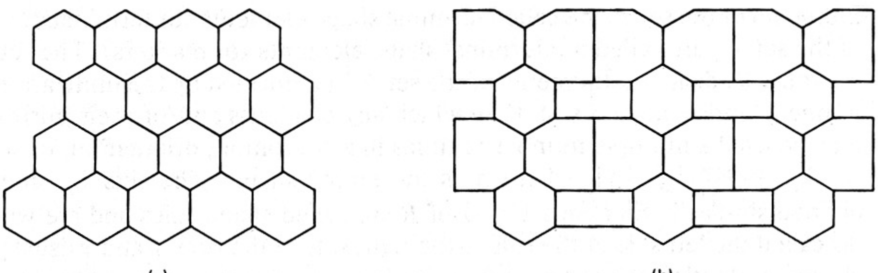


Fig. 6.8 Textures to be recognized (see text).

A more difficult example is given by the "reptile" texture. Except for the occasional new rows, a (3, 6, 3, 6) tesselation of primitives would model this texture exactly. As shown in Fig. 6.9, the new row is introduced when a seven-sided polygon splits into a six-sided polygon and a five-sided polygon. To capture this with a shape grammar, we examine the dual of this graph, which is the primitive placement graph, Fig. 6.9b. This graph provides a simple explanation of how the extra row is created; that is, the diamond pattern splits into two. Notice that the dual graph is composed solely of four-sided polygons but that some vertices are (4, 4, 4) and some are (4, 4, 4, 4, 4, 4). A shape grammar for the dual is shown in Fig. 6.10. The image texture can be obtained by forming the dual of this graph. One further refinement should be added to rules (6) and (7); so that rule (7) is used less often, the appropriate probabilities should be associated with each rule. This would make the grammar stochastic.

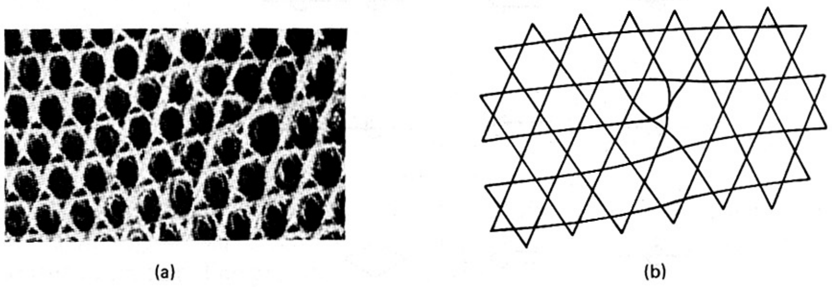


Fig. 6.9 (a) The reptile texture. (b) The reptile texture as a (3, 6, 3, 6) semiregular tesselation with local deformations.

6.3.3 Tree Grammars

The symbolic form of a tree grammar is very similar to that of a shape grammar. A grammar

$$G_t = (V_t,\ V_m,\ r,\ R,\ S)$$

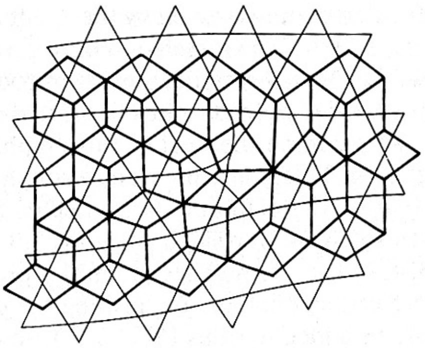
is a tree grammar if

V_t is a set of terminal symbols
V_m is a set of symbols such that
$V_m \bigcap V_t = \phi$
$r : V_t \rightarrow N$ (where N is the set of nonnegative integers)
is the rank associated with symbols in V_t
S is the start symbol
R is the set of rules of the form

$$\begin{tikzcd} X_0 \arrow[r] & x \arrow[dl, dash] \arrow[d, dash] \\ X_0 \cdots & X_{r(x)} \end{tikzcd} \qquad \text{or} \qquad X_0 \rightarrow x$$

with x in V_t and $X_0 \ldots X_{r(x)}$ in V_m

For a tree grammar to generate arrays of pixels, it is necessary to choose some way of embedding the tree in the array. Figure 6.11 shows two such embeddings.



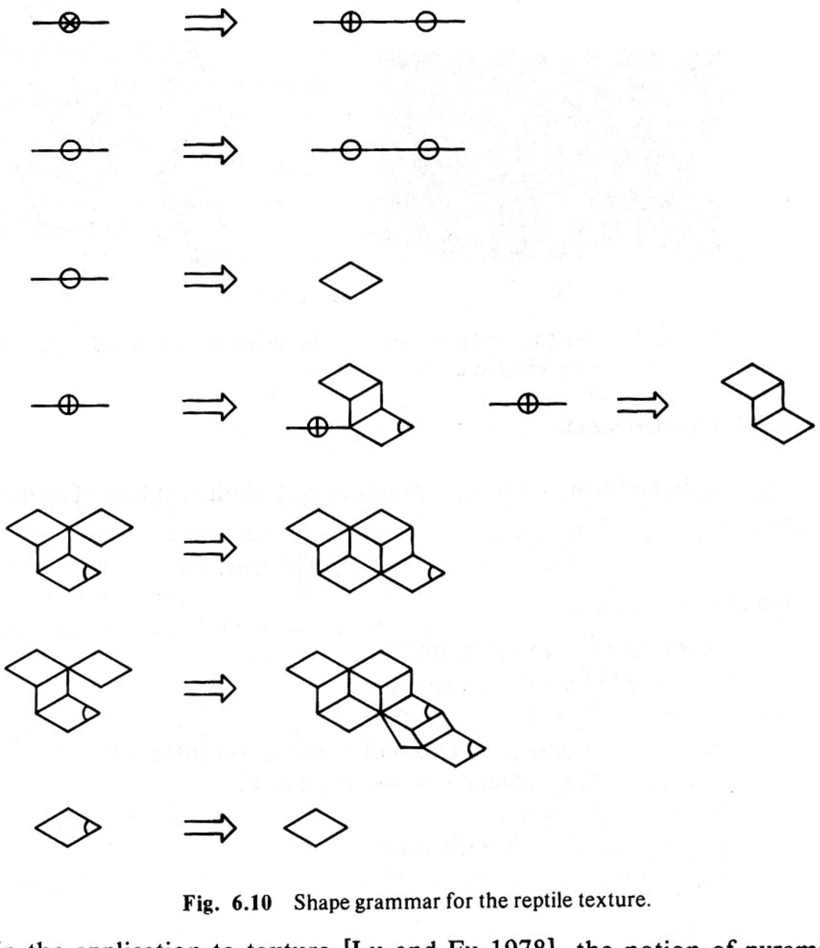


Fig. 6.10 Shape grammar for the reptile texture.

In the application to texture [Lu and Fu 1978], the notion of pyramids or hierarchical levels of resolution in texture is used. One level describes the placement of repeating patterns in texture windows—a rectangular texel placement tesselation—and another level describes texels in terms of pixels. We shall illus-

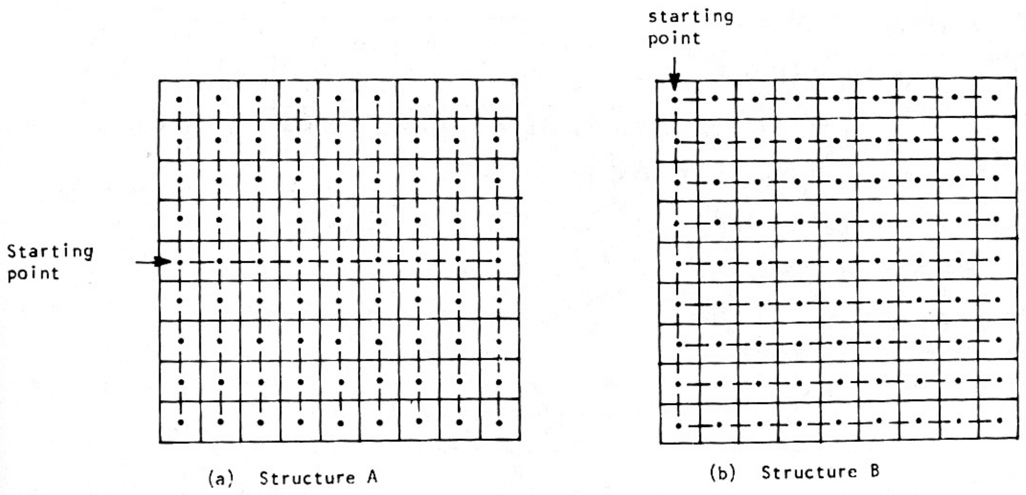


Fig. 6.11 Two ways of embedding a tree structure in an array.

trate these ideas with Lu and Fu's grammar for "wire braid." The texture windows are shown in Fig. 6.12a. Each of these can be described by a "sentence" in a second tree grammar. The grammar is given by:

$$G_w = (V_t, V_m, r, R, S)$$

where

$$\begin{aligned} V_t &= \{A_1, C_1\} \\ V_m &= \{X, Y, Z\} \\ r &= \{0, 1, 2\} \end{aligned} \tag{6.2}$$

$$R: X \rightarrow \begin{array}{c} A_1 \\ \diagup \quad \diagdown \\ X \qquad Y \end{array} \quad \text{or } \begin{array}{c} A_1 \\ | \\ Y \end{array}$$

$$Y \rightarrow \begin{array}{c} C_1 \\ | \\ Z \end{array} \quad \text{or } C_1$$

$$Z \rightarrow \begin{array}{c} A_1 \\ | \\ Y \end{array} \quad \text{or } A_1$$

and the first embedding in Fig. 6.11 is used. The pattern inside each of these windows is specified by another grammatical level:

$$G = (V_t, V_m, r, R, S)$$

where

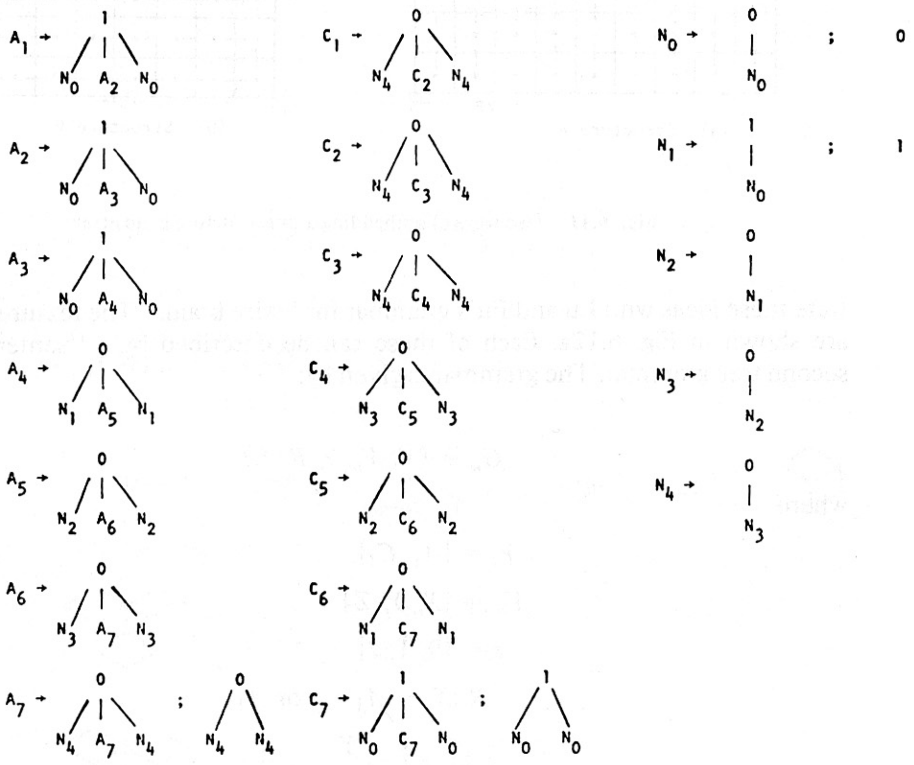
$$V_t = \{1, 0\}$$

$$V_m = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, C_1, C_2, C_3, C_4, C_5, C_6, C_7, N_0, N_1, N_2, N_3, N_4\}$$

$$r = \{0, 1, 2\}$$

$$S = \{A_1, C_1\}$$

R:



The application of these rules generates the two different patterns of pixels shown in Fig. 6.13.

6.3.4 Array Grammars

Like tree grammars, array grammars use hierarchical levels of resolution [Milgram and Rosenfeld 1971; Rosenfeld 1971]. Array grammars are different from tree grammars in that they do not use the tree-array embedding. Instead, prodigious use of a blank or null symbol is used to make sure the rules are applied in appropriate contexts. A simple array grammar for generating a checkerboard pattern is

$$G = \{V_t, V_n, R\}$$

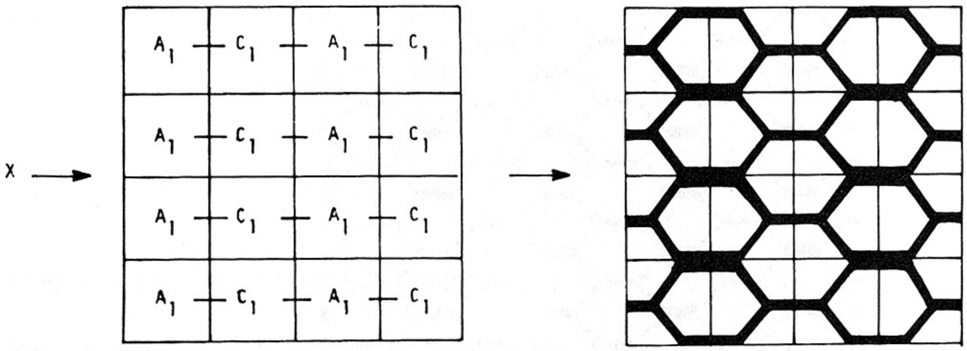


Fig. 6.12 Texture window and grammar (see text).

where

$$V_t = \{0, 1\} \text{ (corresponding to black and white pixels, respectively)}$$
$$V_n = \{b, S\}$$

b is a "blank" symbol used to provide context for the application of the rules. Another notational convenience is to use a subscript to denote the orientation of symbols. For example, when describing the rules R we use

$$0_x b \rightarrow 0_x 1 \qquad \text{where } x \text{ is one of } \{U, D, L, R\}$$

to summarize the four rules

$$\begin{matrix} 0 \\ b \end{matrix} \rightarrow \begin{matrix} 0 \\ 1 \end{matrix}, \quad \begin{matrix} b \\ 0 \end{matrix} \rightarrow \begin{matrix} 1 \\ 0 \end{matrix}, \quad 0b \rightarrow 01, \quad b0 \rightarrow 10$$

Thus the checkerboard rule set is given by

$$R: S \rightarrow 0 \text{ or } 1$$
$$0_x b \rightarrow 0_x 1 \qquad x \text{ in } \{U, D, L, R\}$$
$$1_x b \rightarrow 1_x 0$$

A compact encoding of textural patterns [Jayaramamurthy 1979] uses levels of array grammars defined on a pyramid. The terminal symbols of one layer are the start symbols of the next grammatical layer defined lower down in the pyramid. This corresponds nicely to the idea of having one grammar to generate primitives and another to generate the primitive placement tesselations.

As another example, consider the herringbone pattern in Fig. 6.14a, which is composed of 4×3 arrays of a particular placement pattern as shown in Fig. 6.14b. The following grammar is sufficient to generate the placement pattern.

$$G_w = \{V_t, V_m, R, S\}$$

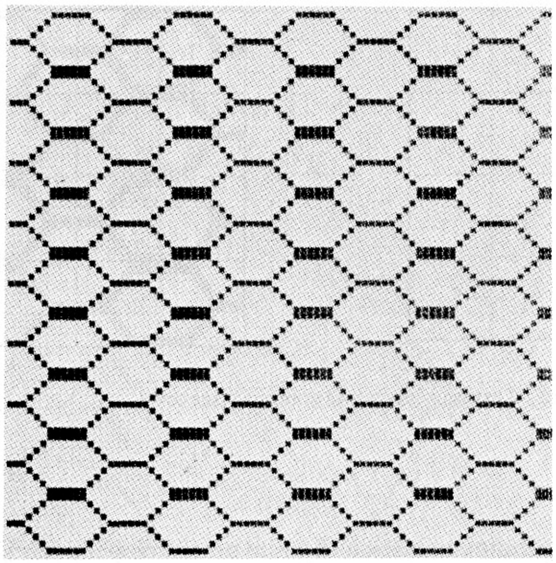


Fig. 6.13 Texture generated by tree grammar.

where

$$V_t = \{a\}$$
$$V_n = \{b, S\}$$
$$R: S \to a$$
$$a_x b \to a_x a \qquad x \text{ in } \{U, D, L, R\}$$

We have not been precise in specifying how the terminal symbol is projected onto the lower level. Assume without loss of generality that it is placed in the upper left-hand corner, the rest of the subarray being initially blank symbols. Thus a simple grammar for the primitive is

$$G_t = \{V_t, V_n, R, S\}$$

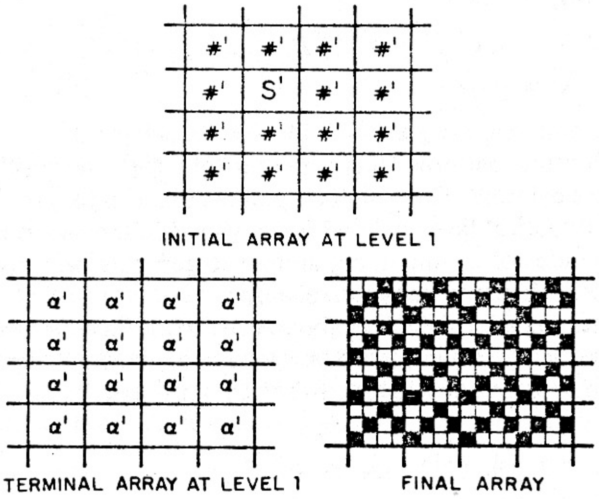


Fig. 6.14 Steps in generating a herringbone texture with an array grammar.

where

$$V_t = \{0, 1\}$$

$$V_n = \{a, b\}$$

$$R: \begin{array}{cccc} a & b & b & b \\ b & b & b & b \\ b & b & b & b \end{array} \rightarrow \begin{array}{cccc} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{array}$$

6.4 TEXTURE AS A PATTERN RECOGNITION PROBLEM

Many textures do not have the nice geometrical regularity of "reptile" or "wire braid"; instead, they exhibit variations that are not satisfactorily described by shapes, but are best described by statistical models. *Statistical pattern recognition* is a paradigm that can classify statistical variations in patterns. (There are other statistical methods of describing texture [Pratt et al. 1981], but we will focus on statistical pattern recognition since it is the most widely used for computer vision purposes.) There is a voluminous literature on pattern recognition, including several excellent texts (e.g., [Fu 1968; Tou and Gonzalez 1974; Fukunaga 1972], and the ideas have much wider application than their use here, but they seem particularly appropriate for low-resolution textures, such as those seen in aerial images [Weszka et al. 1976]. The pattern recognition approach to the problem is to classify instances of a texture in an image into a set of classes. For example, given the textures in Fig. 6.15, the choice might be between the classes "orchard," "field," "residential," "water."

The basic notion of pattern recognition is the *feature vector.* The feature vector $\mathbf{v}$ is a set of measurements $\{v_1 \cdots v_m\}$ which is supposed to condense the description of relevant properties of the textured image into a small, Euclidean *feature space* of m dimensions. Each point in feature space represents a value for the feature vector applied to a different image (or subimage) of texture. The measurement values for a feature should be correlated with its class membership. Figure 6.16 shows a two-dimensional space in which the features exhibit the desired correlation property. Feature vector values cluster according to the texture from which they were derived. Figure 6.16 shows a bad choice of features (measurements) which does not separate the different classes.

The pattern recognition paradigm divides the problem into two phases: training and test. Usually, during a training phase, feature vectors from known samples are used to partition feature space into regions representing the different classes. However, self teaching can be done; the classifier derives its own partitions. Feature selection can be based on parametric or nonparametric models of the distributions of points in feature space. In the former case, analytic solutions are sometimes available. In the latter, feature vectors are *clustered* into groups which are taken to indicate partitions. During a test phase the feature-space partitions are used to classify feature vectors from unknown samples. Figure 6.17 shows this process.

Given that the data are reasonably well behaved, there are many methods for clustering feature vectors [Fukunaga 1972; Tou and Gonzales 1974; Fu 1974].

where

$$V_t = \{0, 1\}$$

$$V_n = \{a, b\}$$

$$R: \begin{matrix} a & b & b & b \\ b & b & b & b \\ b & b & b & b \end{matrix} \rightarrow \begin{matrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{matrix}$$

6.4 TEXTURE AS A PATTERN RECOGNITION PROBLEM

Many textures do not have the nice geometrical regularity of "reptile" or "wire braid"; instead, they exhibit variations that are not satisfactorily described by shapes, but are best described by statistical models. *Statistical pattern recognition* is a paradigm that can classify statistical variations in patterns. (There are other statistical methods of describing texture [Pratt et al. 1981], but we will focus on statistical pattern recognition since it is the most widely used for computer vision purposes.) There is a voluminous literature on pattern recognition, including several excellent texts (e.g., [Fu 1968; Tou and Gonzalez 1974; Fukunaga 1972], and the ideas have much wider application than their use here, but they seem particularly appropriate for low-resolution textures, such as those seen in aerial images [Weszka et al. 1976]. The pattern recognition approach to the problem is to classify instances of a texture in an image into a set of classes. For example, given the textures in Fig. 6.15, the choice might be between the classes "orchard," "field," "residential," "water."

The basic notion of pattern recognition is the *feature vector.* The feature vector **v** is a set of measurements $\{v_1 \cdots v_m\}$ which is supposed to condense the description of relevant properties of the textured image into a small, Euclidean *feature space* of m dimensions. Each point in feature space represents a value for the feature vector applied to a different image (or subimage) of texture. The measurement values for a feature should be correlated with its class membership. Figure 6.16 shows a two-dimensional space in which the features exhibit the desired correlation property. Feature vector values cluster according to the texture from which they were derived. Figure 6.16 shows a bad choice of features (measurements) which does not separate the different classes.

The pattern recognition paradigm divides the problem into two phases: training and test. Usually, during a training phase, feature vectors from known samples are used to partition feature space into regions representing the different classes. However, self teaching can be done; the classifier derives its own partitions. Feature selection can be based on parametric or nonparametric models of the distributions of points in feature space. In the former case, analytic solutions are sometimes available. In the latter, feature vectors are *clustered* into groups which are taken to indicate partitions. During a test phase the feature-space partitions are used to classify feature vectors from unknown samples. Figure 6.17 shows this process.

Given that the data are reasonably well behaved, there are many methods for clustering feature vectors [Fukunaga 1972; Tou and Gonzales 1974; Fu 1974].

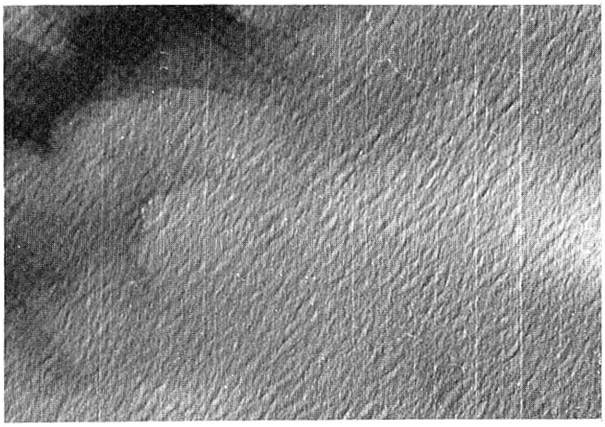
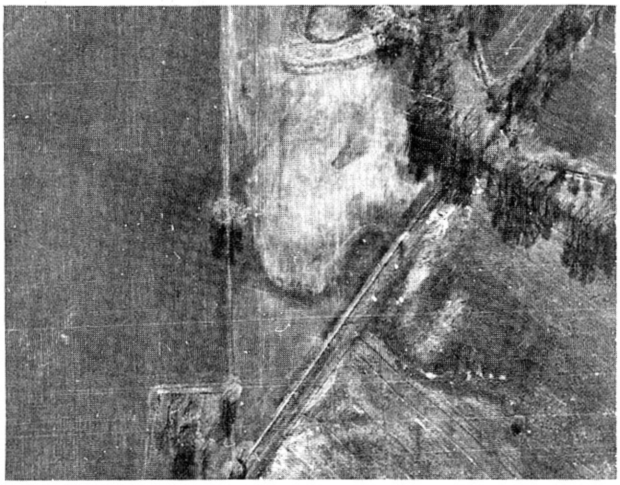
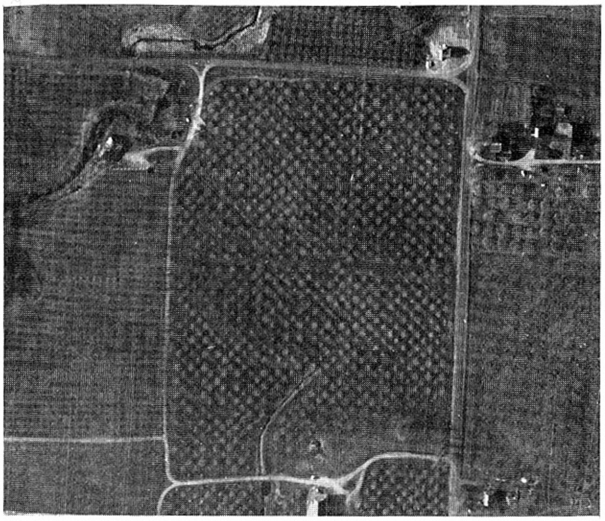




Fig. 6.15 Aerial image textures for discrimination.

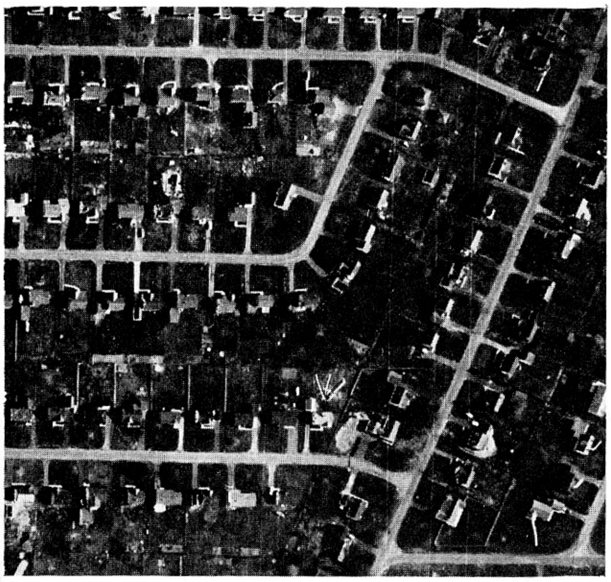


Fig. 6.15 (cont.)

One popular way of doing this is to use prototype points for each class and a *nearest-neighbor* rule [Cover 1968]:

assign $\mathbf{v}$ to class w_i if i minimizes
$\min_i d(\mathbf{v}, \mathbf{v}_{w_i})$

where $\mathbf{v}_{w_i}$ is the prototype point for class w_i.

Parametric techniques assume information about the feature vector probability distributions to find rules that maximize the likelihood of correct classification:

assign $\mathbf{v}$ to class w_i if i maximizes
$\max_i p(w_i|\mathbf{v})$

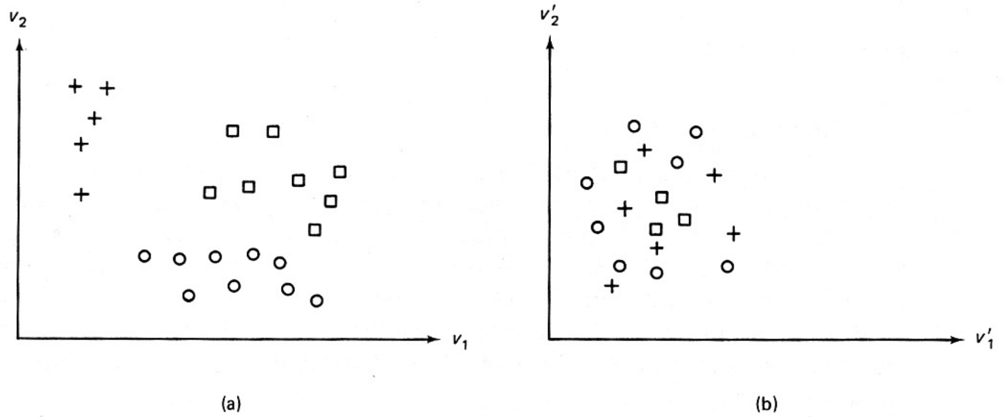


Fig. 6.16 Feature space for texture discrimination. (a) effective features (b) ineffective features.

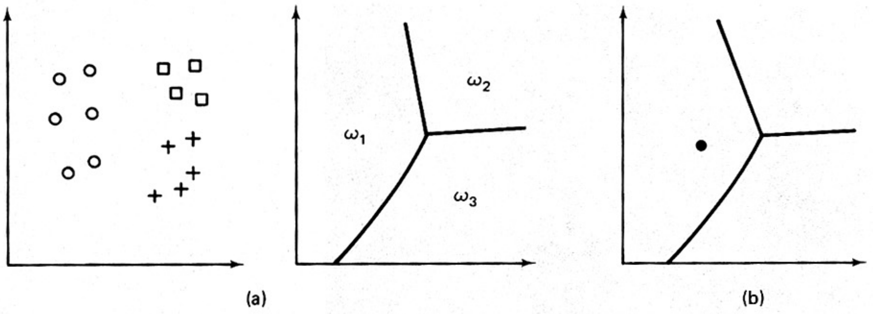


Fig. 6.17 Pattern recognition paradigm.

The distributions may also be used to formulate rules that minimize errors.

Picking good features is the essence of pattern recognition. No elaborate formalism will work well for bad features such as those of Fig. 6.15b. On the other hand, almost any method will work for very good features. For this reason, texture is a good domain for pattern recognition: it is fairly easy to define features that (1) cluster in feature space according to different classes, and (2) can separate texture classes.

The ensuing subsections describe features that have worked well. These subsections are in reverse order from those of Section 6.2 in that we begin with features defined on pixels—Fourier subspaces, gray-level dependencies—and conclude with features defined on higher-level texels such as regions. However, the lesson is the same as with the grammatical approach: hard work spent in obtaining high-level primitives can both improve and simplify the texture model. Space does not permit a discussion of many texture features; instead, we limit ourselves to a few representative samples. For further reading, see [Haralick 1978].

6.4.1 Texture Energy

Fourier Domain Basis

If a texture is at all spatially periodic or directional, its power spectrum will tend to have peaks for corresponding spatial frequencies. These peaks can form the basis of features of a pattern recognition discriminator. One way to define features is to search Fourier space directly [Bajcsy and Lieberman 1976]. Another is to partition Fourier space into bins. Two kinds of bins, radial and angular, are commonly used, as shown in Fig. 6.18. These bins, together with the Fourier power spectrum are used to define features. If F is the Fourier transform, the Fourier power spectrum is given by $|F|^2$.

Radial features are given by

$$v_{r_1 r_2} = \int\int |F(u, v)|^2 \, du \, dv \tag{6.5}$$

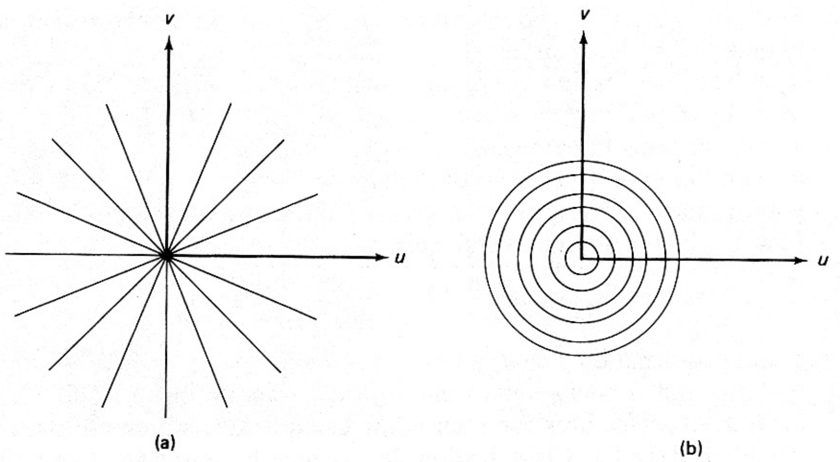


Fig. 6.18 Partitioning the Fourier domain into bins.

where the limits of integration are defined by

$$r_1^2 \leqslant u^2 + v^2 < r_2^2$$
$$0 \leqslant u,\ v < n-1$$

where $[r_1,\ r_2]$ is one of the radial bins and **v** is the vector (not related to v) defined by different values of r_1 and r_2. Radial features are correlated with texture coarseness. A smooth texture will have high values of $V_{r_1 r_2}$ for small radii, whereas a coarse, grainy texture will tend to have relatively higher values for larger radii.

Features that measure angular orientation are given by

$$\mathbf{v}_{\theta_1 \theta_2} = \int\int |F(u,\ v)|^2\, du\, dv \qquad (6.6)$$

where the limits of integration are defined by

$$\theta_1 \leqslant \tan^{-1}\left(\frac{v}{u}\right) < \theta_2$$
$$0 < u,\ v \leqslant n-1$$

where $[\theta_1,\ \theta_2)$ is one of the sectors and **v** is defined by different values of θ_1 and θ_2. These features exploit the sensitivity of the power spectrum to the directionality of the texture. If a texture has as many lines or edges in a given direction θ, $|F|^2$ will tend to have high values clustered around the direction in frequency space $\theta + \pi/2$.

Texture Energy in the Spatial Domain

From Section 2.2.4 we know that the Fourier approach could also be carried out in the image domain. This is the approach taken in [Laws 1980]. The advantage of this approach is that the basis is not the Fourier basis but a variant that is more matched to intuition about texture features. Figure 6.19 shows the most important of Laws' 12 basis functions.

The image is first histogram-equalized (Section 3.2). Then 12 new images are made by convolving the original image with each of the basis functions (i.e., $f_k' = f * h_k$ for basis functions $h_1, \ldots, h_{12}$). Then each of these images is transformed into an "energy" image by the following transformation: Each pixel in the convolved image is replaced by an average of the absolute values in a local window of 15 × 15 pixels centered over the pixel:

$$f_k''(x, y) = \sum_{x', y' \text{ in window}} (|f_k'(x', y')|) \tag{6.7}$$

The transformation $f \rightarrow f_k''$, $k = 1, \ldots 12$ is termed a "texture energy transform" by Laws and is analogous to the Fourier power spectrum. The f_k'', $k = 1, \ldots 12$ form a set of features for each point in the image which are used in a nearest-neighbor classifier. Classification details may be found in [Laws 1980]. Our interest is in the particular choice of basis functions used.

Figure 6.20 shows a composite of natural textures [Brodatz 1966] used in Laws's experiments. Each texture is digitized into a 128 × 128 pixel subimage. The texture energy transforms were applied to this composite image and each pixel was classified into one of the eight categories. The average classification accuracy was about 87% for interior regions of the subimages. This is a very good result for textures that are similar.

6.4.2 Spatial Gray-Level Dependence

Spatial gray-level dependence (SGLD) matrices are one of the most popular sources of features [Kruger et al. 1974; Hall et al. 1971; Haralick et al. 1973]. The SGLD approach computes an intermediate matrix of measures from the digitized image data, and then defines features as functions on this intermediate matrix. Given an image f with a set of discrete gray levels I, we define for each of a set of discrete values of d and θ the intermediate matrix $S(d, \theta)$ as follows:

> $S(i, j \mid d, \theta)$, an entry in the matrix, is the number of times gray level i is oriented with respect to gray level j such that where
>
> $$f(\mathbf{x}) = i \quad \text{and} \quad f(\mathbf{y}) = j \quad \text{then}$$
> $$\mathbf{y} = \mathbf{x} + (d\cos\theta,\ d\sin\theta)$$

$$\begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 16 & -24 & 16 & -4 \\ 6 & -24 & 36 & -24 & 6 \\ -4 & 16 & -24 & 16 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -2 & 0 & 4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 0 & -2 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

Fig. 6.19 Laws' basis functions (these are the low-order four of twelve actually used).

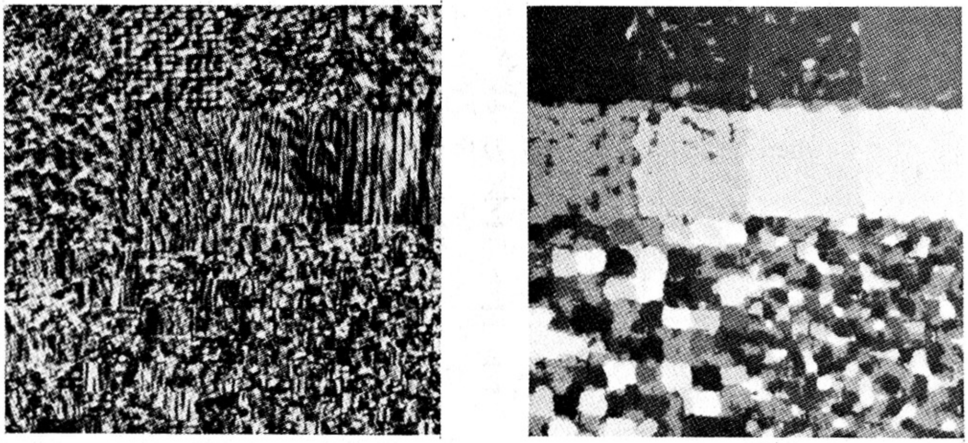

(a) (b)

Fig. 6.20 (a) Texture composite. (b) Classification.

Note that we the gray-level values appear as indices of the matrix S, implying that they are taken from some well-ordered discrete set 0, ..., K. Since

$$S(d, \theta) = S(d, \theta + \pi).$$

common practice is to restrict θ to multiples of $\pi/4$. Furthermore, information is not usually retained at both θ and $\theta + \pi$. The reasoning for the latter step is that for most texture discrimination tasks, the information is redundant. Thus we define

$$S(d, \theta) = ½\,[S(d, \theta) + S(d, \theta + \pi)]$$

The intermediate matrices S yield potential features. Commonly used features are:

1. *Energy*

$$E(d, \theta) = \sum_{i=0}^{K} \sum_{j=0}^{K} [S(i, j \mid d, \theta)]^2 \tag{6.8}$$

2. *Entropy*

$$H(d, \theta) = \sum_{i=0}^{K} \sum_{j=0}^{K} S(i, j \mid d, \theta) \log f(i, j \mid d, \theta) \tag{6.9}$$

3. *Correlation*

$$C(d, \theta) = \frac{\displaystyle\sum_{i=0}^{K} \sum_{j=0}^{K} (i - \mu_x)(j - \mu_y) S(i, j \mid d, \theta)}{\sigma_x \sigma_y} \tag{6.10}$$

4. *Inertia*

$$I(d, \theta) = \sum_{i=0}^{K} \sum_{j=0}^{K} (i - j)^2\, S(i, j \mid d, \theta) \tag{6.11}$$

5. *Local Homogeneity*

$$L(d, \theta) = \sum_{i=0}^{K} \sum_{j=0}^{K} \frac{1}{1 + (i-j)^2} S(i, j|d, \theta) \tag{6.12}$$

where $S(i, j|d, \theta)$ is the (i, j) th element of (d, θ), and

$$\mu_x = \sum_{i=0}^{K} i \sum_{j=0}^{K} S(i, j|d, \theta) \tag{6.13a}$$

$$\mu_y = \sum_{i=0}^{K} j \sum_{j=0}^{K} S(i, j|d, \theta) \tag{6.13b}$$

$$\sigma_x^2 = \sum_{i=0}^{K} (i-\mu_x)^2 \sum_{j=0}^{K} f(i, j|d, \theta) \tag{6.13c}$$

and

$$\sigma_y^2 = \sum_{i=0}^{K} (j-\mu_y)^2 \sum_{i=0}^{K} f(i, j|d, \theta) \tag{6.13d}$$

One important aspect of this approach is that the features chosen do not have psychological correlates [Tamura et al. 1978]. For example, none of the measures described would take on specific values corresponding to our notions of "rough" or "smooth." Also, the texture gradient is difficult to define in terms of SGLD feature values [Bajcsy and Lieberman 1976].

6.4.3 Region Texels

Region texels are an image-based way of defining primitives above the level of pixels. Rather than defining features directly as functions of pixels, a region segmentation of the image is created first. Features can then be defined in terms of the shape of the resultant regions, which are often more intuitive than the pixel-related features. Naturally, the approach of using edge elements is also possible. We shall discuss this in the context of texture gradients.

The idea of using regions as texture primitives was pursued in [Maleson et al. 1977]. In that implementation, all regions are ultimately modeled as ellipses and a corresponding five-parameter shape description is computed for each region. These parameters only define gross region shape, but the five-parameter primitives seem to work well for many domains. The texture image is segmented into regions in two steps. Initially, the modified version of Algorithm 5.1 that works for gray-level images is used. Figure 6.21 shows this example of the segmentation applied to a sample of "straw" texture. Next, parameters of the region grower are controlled so as to encourage convex regions which are fit with ellipses. Figure 6.22 shows the resultant ellipses for the "straw" texture. One set of ellipse parameters is $\mathbf{x}_0$, a, b, θ where $\mathbf{x}_0$ is the origin, a and b are the major and minor axis lengths and θ is the orientation of the major axis (Appendix 1). Besides these shape parameters, elliptical texels are also described by their average gray level. Figure 6.23 gives a qualitative indication of how ranges on feature values reflect different texels.

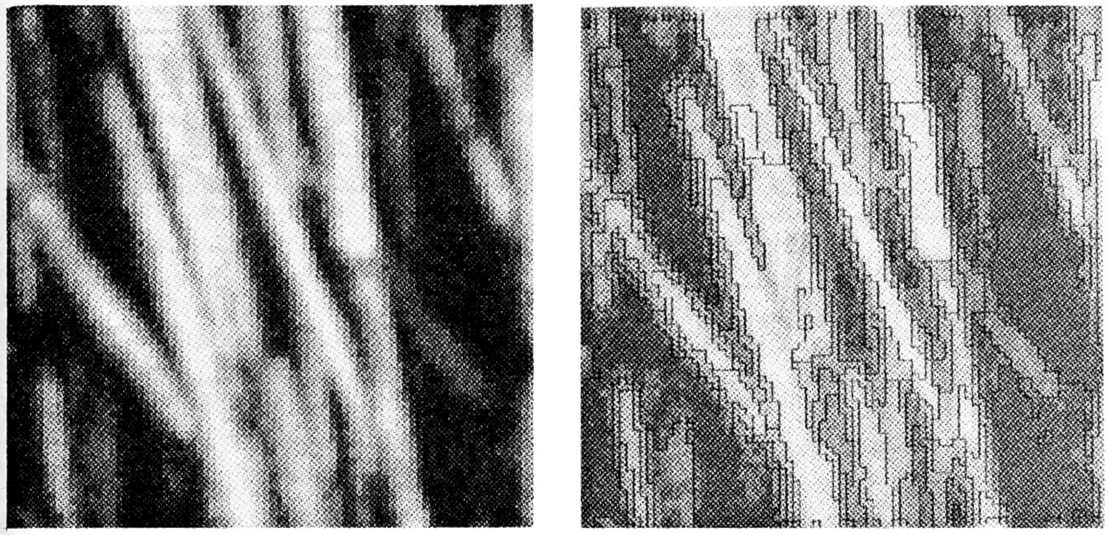

(a) Image (b) With Region Boundaries

Fig. 6.21 Region segmentation for straw texture.

6.5 THE TEXTURE GRADIENT

The importance of texture in determining surface orientation was described by Gibson [Gibson 1950]. There are three ways in which this can be done. These methods are depicted in Fig. 6.24. All these methods assume that the texture is embedded on a planar surface.

First, if the texture image has been segmented into primitives, the maximum rate of change of the projected size of these primitives constrains the orientation of

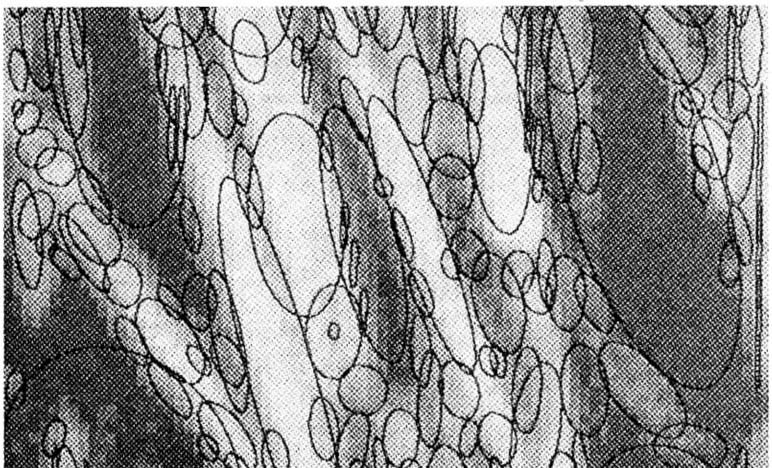


Fig. 6.22 Ellipses for straw texture.

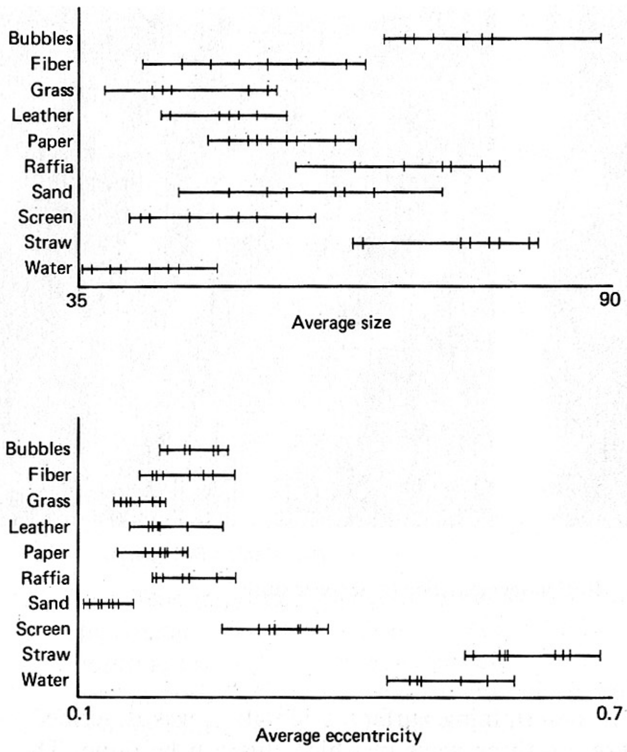


Fig. 6.23 Features defined on ellipses.

the plane in the following manner. The direction of maximum rate of change of projected primitive size is the direction of the *texture gradient.* The orientation of this direction with respect to the image coordinate frame determines how much the plane is rotated about the camera line of sight. The magnitude of the gradient can help determine how much the plane is tilted with respect to the camera, but knowledge about the camera geometry is also required. We have seen these ideas before in the form of gradient space; the rotation and tilt characterization is a polar coordinate representation of gradients.

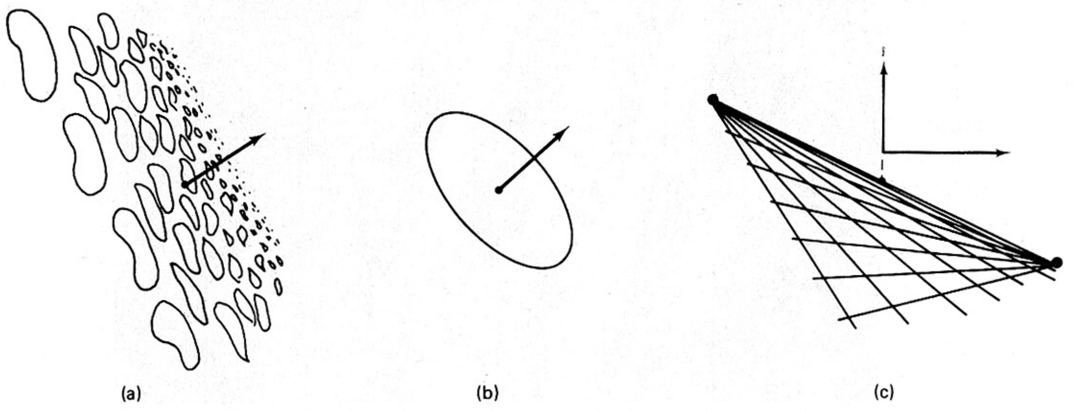


Fig. 6.24 Methods for calculating surface orientation from texture.

The second way to measure surface orientation is by knowing the shape of the texel itself. For example, a texture composed of circles appears as ellipses on the tilted surface. The orientation of the principal axes defines rotation with respect to the camera, and the ratio of minor to major axes defines tilt [Stevens 1979].

Finally, if the texture is composed of a regular grid of texels, we can compute vanishing points. For a perspective image, vanishing points on a plane P are the projection onto the image plane of the points at infinity in a given direction. In the examples here, the texels themselves are (conveniently) small line segments on a plane that are oriented in two orthogonal directions in the physical world. The general method applies whenever the placement tesselation defines lines of texels. Two vanishing points that arise from texels on the same surface can be used to determine orientation as follows. The line joining the vanishing points provides the orientation of the surface and the vertical position of the plane with respect to the z axis (i.e., the intersection of the line joining the vanishing points with $x = 0$) determines the tilt of the plane.

Line segment textures indicate vanishing points [Kender 1978]. As shown in Fig. 6.25, these segments could arise quite naturally from an urban image of the windows of a building which has been processed with an edge operator.

As discussed in Chapter 4, lines in images can be detected by detecting their parameters with a Hough algorithm. For example, by using the line parameterization

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

and by knowing the orientation of the line in terms of its gradient $\mathbf{g} = (\Delta x,\ \Delta y)$, a line segment $(x, y, \Delta x, \Delta y)$ can be mapped into r, θ space by using the relations

$$r = \frac{\Delta x x + \Delta y y}{\sqrt{\Delta x^2 + \Delta y^2}} \tag{6.14}$$

$$\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \tag{6.15}$$

These relationships can be derived by using Fig. 6.26 and some geometry. The Cartesian coordinates of the $r-\theta$ space vector are given by

$$\mathbf{a} = \left(\frac{\mathbf{g}\cdot\mathbf{x}}{\|\mathbf{g}\|^2}\right)\mathbf{g} \tag{6.16}$$

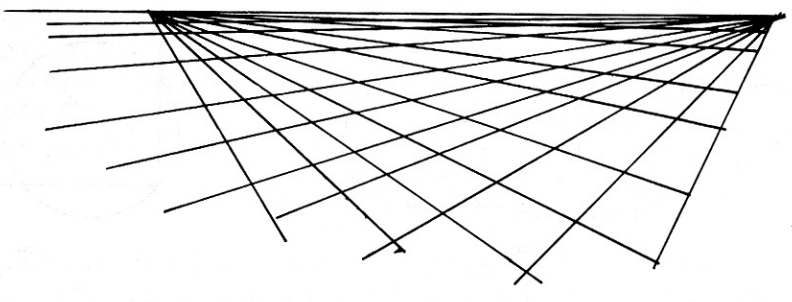


Fig. 6.25 Orthogonal line segments comprising a texture.

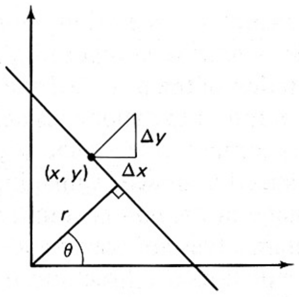


Fig. 6.26 r-θ transform.

Using this transformation, the set of line segments L_1 shown in Fig. 6.27 are all mapped into a single point in $r-\theta$ space. Furthermore, the set of lines L_2 which have the same vanishing point (x_v, y_v) project onto a circle in $r-\theta$ space with the line segment $((0, 0), (x_v, y_v))$ as a diameter. This scheme has two drawbacks: (1) vanishing points at infinity are projected into infinity, and (2) circles require some effort to detect. Hence we are motivated to use the transform $(x, y, \Delta x, \Delta y) \rightarrow \left(\frac{k}{r}, \theta\right)$ for some constant k. Now vanishing points at infinity are projected into the origin and the locus of the set of points L_2 is now a line. This line is perpendicular to the vector $\boldsymbol{x}_v$ and $\frac{k}{\|\mathbf{x}_v\|}$ units from the origin, as shown in Fig. 6.28. It can be detected by a second stage of the Hough transform; each point $\mathbf{a}$ is mapped into an $r'-\theta'$ space. For every $\mathbf{a}$, compute all the r', θ' such that

$$a \cos\theta' + b \sin\theta' = r' \tag{6.17}$$

and increment that location in the appropriate r', θ' accumulator array. In this second space a vanishing point is detected as

$$r' = \frac{k}{\|\mathbf{x}_v\|} \tag{6.18}$$

$$\theta' = \tan^{-1}\left(\frac{y_v}{x_v}\right) \tag{6.19}$$

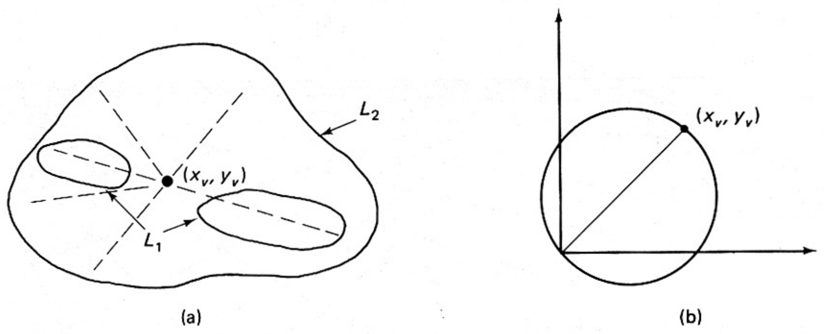


Fig. 6.27 Detecting the vanishing point with the Hough transform.

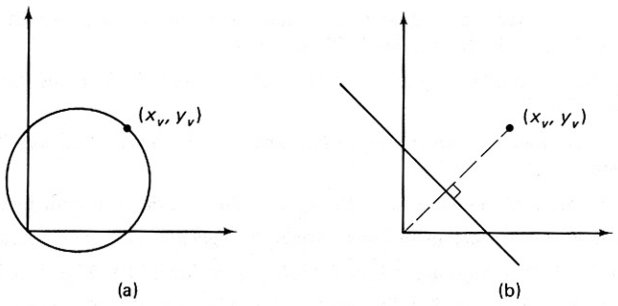


Fig. 6.28 Vanishing point loci.

In Kender's application the texels and their placement tesselation are similar in that the primitives are parallel to arcs in the placement tesselation graph. In a more general application the tesselation could be computed by connecting the centers of primitives.

EXERCISES

6.1 Devise a computer algorithm that, given a set of texels from each of a set of different "windows" of the textured image, checks to see of the resolution is appropriate. In other words, try to formalize the discussion of resolution in Section 6.2.

6.2 Are any of the grammars in Section 6.3 suitable for a parallel implementation (i.e., parallel application of rules)? Discuss, illustrating your arguments with examples or counterexamples from each of the three main grammatical types (shape, tree, and array grammars).

6.3 Are shape, array, and tree grammars context free or context-sensitive as defined? Can such grammars be translated into "traditional" (string) grammars? If not, how are they different; and if so, why are they useful?

6.4 Show how the generalized Hough transform (Section 4.3) could be applied to texel detection.

6.5 In an outdoors scene, there is the problem of different scales. For example, consider the grass. Grass that is close to an observer will appear "sharp" and composed of primitive elements, yet grass distant from an observer will be much more "fuzzy" and homogeneous. Describe how one might handle this problem.

6.6 The texture energy transform (Section 6.4.1) is equivalent to a set of Fourier-domain operations. How do the texture energy features compare with the ring and sector features?

6.7 The texture gradient is presumably a gradient in some aspect of texture. What aspect is it, and how might it be quantified so that texture descriptions can be made gradient independent?

6.8 Write a texture region grower and apply it to natural scenes.

REFERENCES

BAJCSY, R. and L. LIEBERMAN. "Texture gradient as a depth cue." *CGIP 5*, 1, March 1976, 52–67.

BRODATZ, P. *Textures: A Photographic Album for Artists and Designers.* Toronto: Dover Publishing Co., 1966.

CONNORS, R. "Towards a set of statistical features which measure visually perceivable qualities of textures." *Proc.*, PRIP, August 1979, 382–390.

COVER, T. M. "Estimation by the nearest neighbor rule." *IEEE Trans. Information Theory 14*, January 1968, 50–55.

FU, K. S. *Sequential Methods in Pattern Recognition and Machine Learning.* New York: Academic Press, 1968.

FU, K. S. *Syntactic Methods in Pattern Recognition.* New York: Academic Press, 1974.

FUKUNAGA, K. *Introduction to Statistical Pattern Recognition.* New York, Academic Press, 1972.

GIBSON, J. J. *The Perception of the Visual World.* Cambridge, MA: Riverside Press, 1950.

HALL, E. L, R. P. KRUGER, S. J. DWYER III, D. L. HALL, R. W. MCLAREN, and G. S. LODWICK. "A survey of preprocessing and feature extraction techniques for radiographic images." *IEEE Trans. Computers 20*, September 1971.

HARALICK, R. M. "Statistical and structural approaches to texture." *Proc.*, 4th IJCPR, November 1978, 45–60.

HARALICK, R. M., R. SHANMUGAM, and I. DINSTEIN. "Textural features for image classification." *IEEE Trans. SMC 3*, November 1973, 610–621.

HOPCROFT, J. E. and J. D. ULLMAN. *Introduction to Automata Theory, Languages and Computation.* Reading, MA: Addison-Wesley, 1979.

JAYARAMAMURTHY, S. N. "Multilevel array grammars for generating texture scenes." *Proc.*, PRIP, August 1979, 391–398.

JULESZ, B. "Textons, the elements of texture perception, and their interactions." *Nature 290*, March 1981, 91–97.

KENDER, J. R. "Shape from texture: a brief overview and a new aggregation transform." *Proc.*, DARPA IU Workshop, November 1978, 79–84.

KRUGER, R. P., W. B. THOMPSON, and A. F. TWINER. "Computer diagnosis of pneumoconiosis." *IEEE Trans. SMC 45*, 1974, 40–49.

LAWS, K. I. "Textured image segmentation." Ph.D. dissertation, Dept. of Engineering, Univ. Southern California, 1980.

LU, S. Y. and K. S. FU. "A syntactic approach to texture analysis." *CGIP 7*, 3, June 1978, 303–330.

MALESON, J. T., C. M. BROWN, and J. A. FELDMAN. "Understanding natural texture." *Proc.*, DARPA IU Workshop, October 1977, 19–27.

MILGRAM, D. L. and A. ROSENFELD. "Array automata and array grammars." *Proc.*, IFIP Congress 71, Booklet TA-2. Amsterdam: North-Holland, 1971, 166–173.

PRATT, W. K., O. D. FAUGERAS, and A. GAGALOWICZ. "Applications of Stochastic Texture Field Models to Image Processing." *Proc. of the IEEE.* Vol.69, No. 5, May 1981

ROSENFELD, A. "Isotonic grammars, parallel grammars and picture grammars." In *MI6*, 1971.

STEVENS, K.A. "Representing and analyzing surface orientation." In *Artificial Intelligence: An MIT Perspective,* Vol. 2, P. H. Winston and R. H. Brown (Eds.). Cambridge, MA: MIT Press, 1979.

STINY, G. and J. GIPS. *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts.* Berkeley, CA: University of California Press, 1972.

TAMURA, H., S. MORI, and T. YAMAWAKI. "Textural features corresponding to visual perception." *IEEE Trans. SMC 8*, 1978, 460–473.

TOU, J. T. and R. C. GONZALEZ. *Pattern Recognition Principles.* Reading, MA: Addison-Wesley, 1974.

WESZKA, J. S., C. R. DYER, and A. ROSENFELD. "A comparative study of texture measures for terrain classification." *IEEE Trans. SMC 6*, 4, April 1976, 269–285.

ZUCKER, S. W. "Toward a model of texture." *CGIP 5*, 2, June 1976, 190–202.