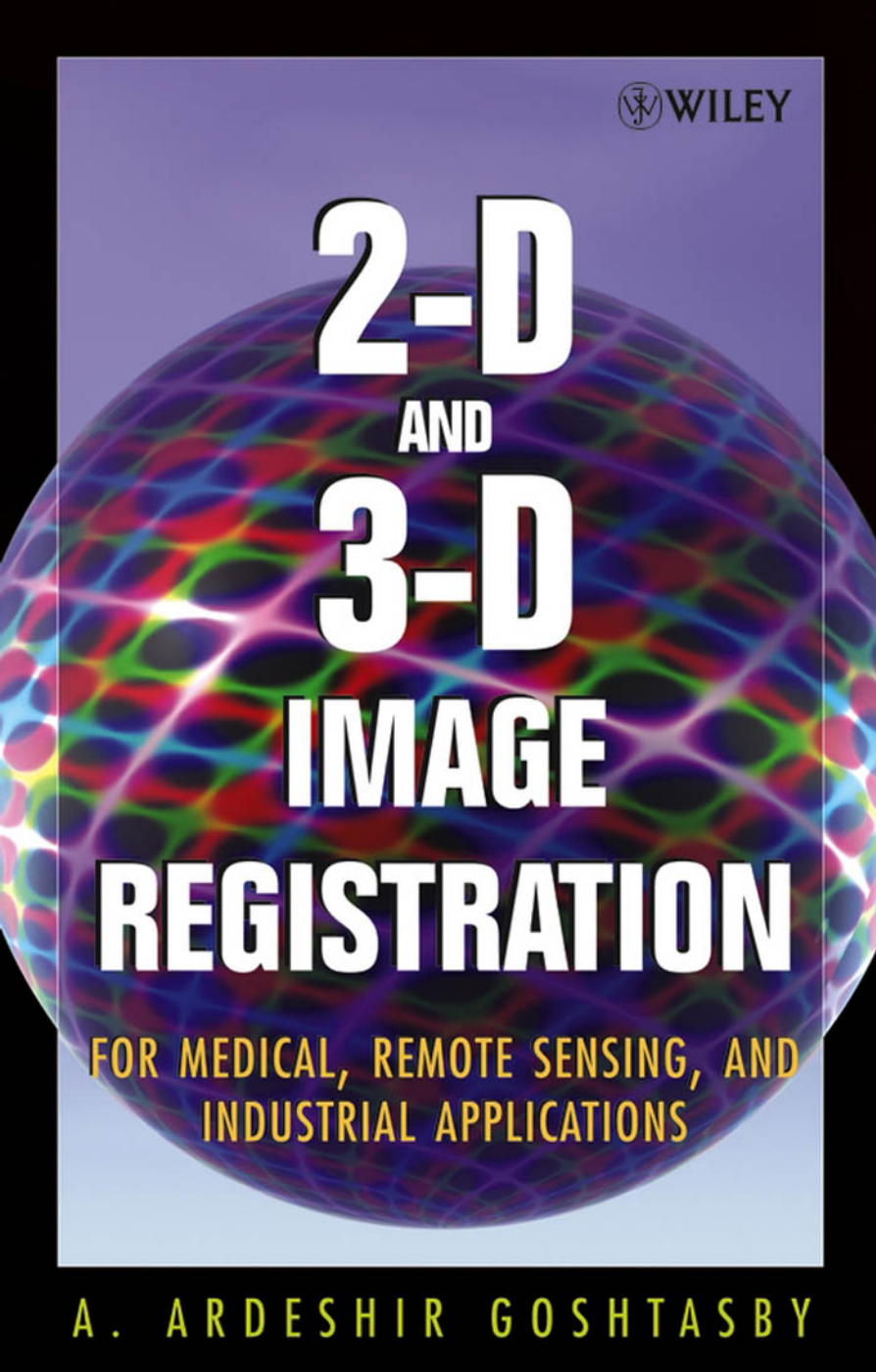

WILEY
2-D
AND
3-D
IMAGE
REGISTRATION
FOR MEDICAL, REMOTE SENSING, AND
INDUSTRIAL APPLICATIONS
A. ARDESHIR GOSHTASBY

2-D and 3-D
Image Registration

2-D and 3-D Image Registration

for Medical, Remote Sensing, and Industrial Applications

A. Ardeshir Goshtasby

WILEY-INTERSCIENCE

A John Wiley & Sons, Inc., Publication

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Goshtasby, Ardeshir.
2-D and 3-D image registration for medical, remote sensing, and industrial applications / A. Ardeshir Goshtasby.
p. cm.
"Wiley-Interscience publication."
Includes bibliographical references and index.
ISBN 0-471-64954-6 (cloth : alk. paper)
1. Image processing–Digital techniques. 2. Image analysis–Data processing. I. Title.

TA1637.G68 2005
621.36'7–dc22

2004059083

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To My Parents
and Mariko and Parviz

Contents

Preface

Image registration is the process of spatially aligning two or more images of a scene. This basic capability is needed in various image analysis applications. The alignment process will determine the correspondence between points in the images, enabling the fusion of information in the images and the determination of scene changes. If identities of objects in one of the images are known, by registering the images, identities of objects and their locations in another image can be determined. Image registration is a critical component of remote sensing, medical, and industrial image analysis systems.

This book is intended for image analysis researchers as well as graduate students who are starting research in image analysis. The book provides details of image registration, and each chapter covers a component of image registration or an application of it. Where applicable, implementation strategies are given and related work is summarized.

In Chapter 1, the main terminologies used in the book are defined, an example of image registration is given, and image registration steps are named. In Chapter 2, preprocessing of images to facilitate image registration is described. This includes image enhancement and image segmentation. Image enhancement is used to remove noise and blur from images and image segmentation is used to partition images into regions or extract region boundaries or edges for use in feature selection.

Chapters 3–5 are considered the main chapters in the book, covering the image registration steps. In Chapter 3, methods and algorithms for detecting points, lines, and regions are described, in Chapter 4, methods and algorithms for determining the correspondence between two sets of features are given, and in Chapter 5, transforma-

tion functions that use feature correspondences to determine a mapping function for image alignment are discussed.

In Chapter 6 resampling methods are given and in Chapter 7 performance evaluation measures, including accuracy, reliability, robustness, and speed are discussed. Chapters 8–10 cover applications of image registration. Chapter 8 discusses creation of intensity and range image mosaics by registering overlapping areas in the images, and Chapter 9 discusses methods for combining information in two or more registered images into a single highly informative image. In particular, fusion of multi-exposure and multi-focus images is discussed. Finally, Chapter 10 discusses registration of stereo images for depth perception. Camera calibration and correspondence algorithms are discussed in detail and examples are given.

Some of the discussions such as stereo depth perception apply to only 2-D images, but many of the topics covered in the book can be applied to both 2-D and 3-D images. Therefore, discussions on 2-D image registration and 3-D image registration continue in parallel. First the 2-D methods and algorithms are described and then their extensions to 3-D are provided.

This book represents my own experiences on image registration during the past twenty years. The main objective has been to cover the fundamentals of image registration in detail. Applications of image registration are not discussed in depth. A large number of application papers appear annually in *Proc. Computer Vision and Pattern Recognition*, *Proc. Int'l Conf. Computer Vision*, *Proc. Int'l Conf. Pattern Recognition*, *Proc. SPIE Int'l Sym. Medical Imaging*, and *Proc. Int'l Sym. Remote Sensing of Environment*. Image registration papers frequently appear in the following journals: *Int'l J. Computer Vision*, *Computer Vision and Image Understanding*, *IEEE Trans. Pattern Analysis and Machine Intelligence*, *IEEE Trans. Medical Imaging*, *IEEE Trans. Geoscience and Remote Sensing*, *Image and Vision Computing*, and *Pattern Recognition*.

The figures used in the book are available online and may be obtained by visiting the website *http://www.imgfsr.com/book.html*. The software implementing the methods and algorithms discussed in the book can be obtained by visiting the same site. Any typographical errors or errata found in the book will also be posted on this site. The site also contains other sources of information relating to image registration.

A. ARDESHIR GOSHTASBY

Dayton, Ohio, USA

Acknowledgments

I would like to thank NASA for providing the satellite images and Kettering Medical Center, Kettering, Ohio, for providing the medical images used in this book. I also would like to thank Shree Nayar of Columbia University for providing the multi-exposure images shown in Figs 8.2–8.5; Max Lyons for providing the multi-exposure images shown in Fig. 8.6; Paolo Favaro, Hailin Jin, and Stefano Saotto of University of California at Los Angeles for providing the multi-focus images shown in Fig. 8.7; Cody Benkelman of Positive Systems for providing the aerial images shown in Fig. 9.4; Yuichi Ohta of Tsukuba University for providing the stereo image pair shown in Fig. 10.10; and Daniel Scharstein of Middlebury College and Rick Szeliski of Microsoft Research for providing the stereo image pair shown in Fig. 10.11. My Ph.D. students, Lyubomir Zagorchev, Lijun Ding, and Marcel Jackowski, have contributed to this book in various ways and I appreciate their contributions. I also would like to thank Libby Stephens for editing the grammar and style of this book.

A. A. G.

Acronyms

CT	X-Ray Computed Tomography
FFT	Fast Fourier Transform
IMQ	Inverse Multiquadric
Landsat	Land Satellite
LoG	Laplacian of Gaussian
MAX	Maximum
MQ	Multiquadric
MR	Magnetic Resonance
MSS	Multispectral Scanner
PET	Positron Emission Tomography
RaG	Rational Gaussian
RMS	Root Mean Squared
TM	Thematic Mapper
TPS	Thin-Plate Spline

1

Introduction

Image registration is the process of determining the point-by-point correspondence between two images of a scene. By registering two images, the fusion of multimodality information becomes possible, the depth map of the scene can be determined, changes in the scene can be detected, and objects can be recognized.

An example of 2-D image registration is shown in Fig. 1.1. Figure 1.1a depicts a Landsat multispectral scanner (MSS) image and Fig. 1.1b shows a Landsat thematic mapper (TM) image of the same area. We will call Fig. 1.1a the *reference image* and Fig. 1.1b the *sensed image*. By resampling the sensed image to the geometry of the reference image, the image shown in Fig. 1.1c is obtained. Figure 1.1d shows overlaying of the resampled sensed image and the reference image. Image registration makes it possible to compare information in reference and sensed images pixel by pixel and determine image differences that are caused by changes in the scene. In the example of Fig. 1.1, closed-boundary regions were used as the features and the centers of corresponding regions were used as the corresponding points. Although ground cover appears differently in the two images, closed regions representing the lakes appear very similar with clear boundaries.

An example of a 3-D image registration is shown in Fig. 1.2. The top row shows orthogonal cross-sections of a magnetic resonance (MR) brain image, the second row shows orthogonal cross-sections of a positron emission tomography (PET) brain image of the same person, the third row shows overlaying of the orthogonal cross-sections of the images before registration, and the fourth row shows overlaying of the orthogonal cross-sections of the images after registration. MR images show anatomy well while PET images show function well. By registering PET and MR brain images, anatomical and functional information can be combined, making it possible to anatomically locate brain regions of abnormal function.

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

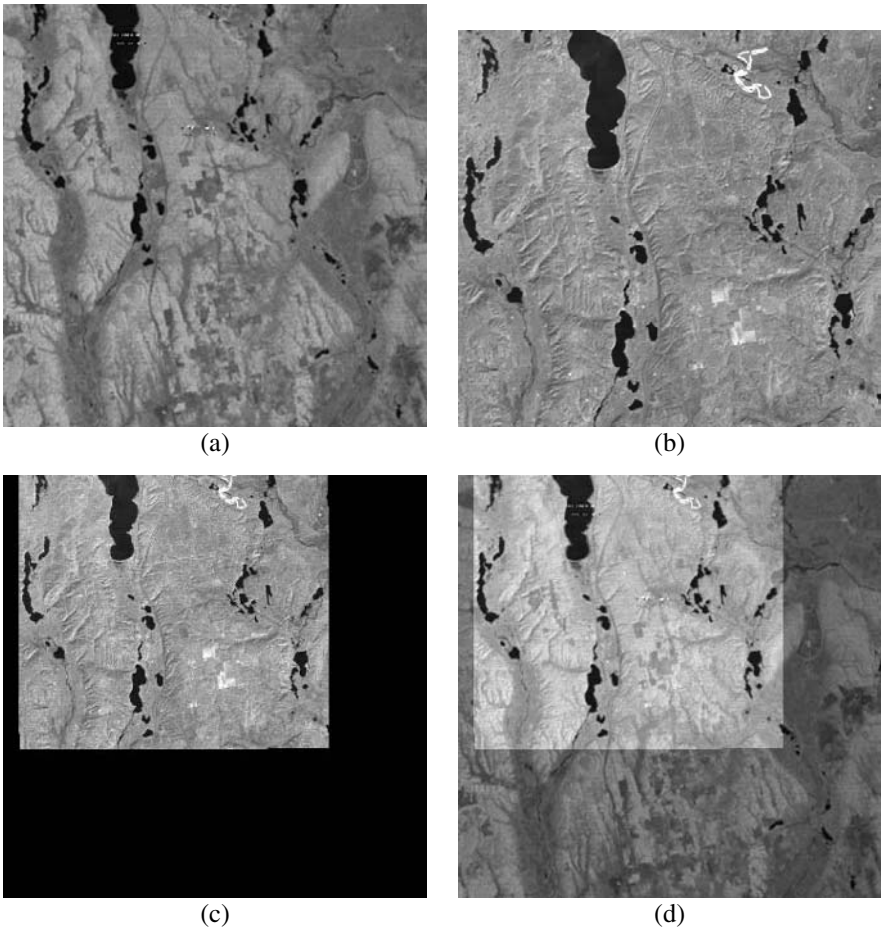


Fig. 1.1 (a) A Landsat MSS image used as the reference image. (b) A Landsat TM image used as the sensed image. (c) Resampling of the sensed image to register the reference image. (d) Overlaying of the reference and resampled sensed images.

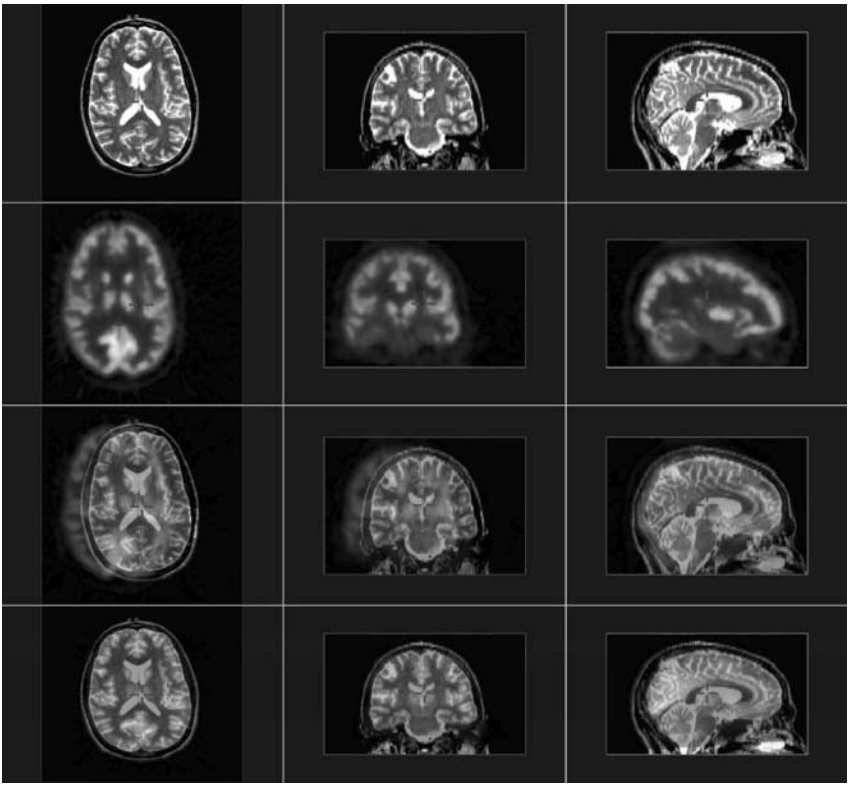


Fig. 1.2 Registration of MR and PET brain images. The first row shows the orthogonal cross-sections of the MR image, the second row shows orthogonal cross-sections of the PET image, the third row shows the images before registration, and the fourth row shows the images after registration.

1.1 TERMINOLOGIES

The following terminologies are used in this book.

1. **Reference Image:** One of the images in a set of two. This image is kept unchanged and is used as the reference. The reference image is also known as the **source image**.
2. **Sensed Image:** The second image in a set of two. This image is resampled to register the reference image. The sensed image is also known as the **target image**.
3. **Transformation Function:** The function that maps the sensed image to the reference image. It is determined using the coordinates of a number of corresponding points in the images.

Further terminologies are listed in the Glossary at the end of the book.

1.2 STEPS IN IMAGE REGISTRATION

Given two images of a scene, the following steps are usually taken to register the images.

1. **Preprocessing:** This involves preparing the images for feature selection and correspondence. Using methods such as scale adjustment, noise removal, and segmentation. When pixel sizes in the images to be registered are different but known, one image is resampled to the scale of the other image. This scale adjustment facilitates feature correspondence. If the given images are known to be noisy, they are smoothed to reduce the noise. Image segmentation is the process of partitioning an image into regions so that features can be extracted.
2. **Feature Selection:** To register two images, a number of features are selected from the images and correspondence is established between them. Knowing the correspondences, a transformation function is then found to resample the sensed image to the geometry of the reference image. The features used in image registration are corners, lines, curves, templates, regions, and patches. The type of features selected in an image depends on the type of image provided. An image of a man-made scene often contains line segments, while a satellite image often contains contours and regions. In a 3-D image, surface patches and regions are often present. Templates are abundant in both 2-D and 3-D images and can be used as features to register images.
3. **Feature Correspondence:** This can be achieved either by selecting features in the reference image and searching for them in the sensed image or by selecting features in both images independently and then determining the correspondence between them. The former method is chosen when the features contain considerable information, such as image regions or templates. The latter method is used when individual features, such as points and lines, do not contain sufficient information. If the features are not points, it is important that from each pair of corresponding features at least one pair of corresponding points is determined. The coordinates of corresponding points are used to determine the transformation parameters. For instance, if templates are used, centers of corresponding templates represent corresponding points; if regions are used, centers of gravity of corresponding regions represent corresponding points; if lines are used, intersections of corresponding line pairs represent corresponding points; and if curves are used, locally maximum curvature points on corresponding curves represent corresponding points.
4. **Determination of a Transformation Function:** Knowing the coordinates of a set of corresponding points in the images, a transformation function is determined to resample the sensed image to the geometry of the reference image. The type of transformation function used should depend on the type of geometric difference between the images. If geometric difference between the

images is not known, a transformation that can easily adapt to the geometric difference between the images should be used.

5. **Resampling:** Knowing the transformation function, the sensed image is resampled to the geometry of the reference image. This enables fusion of information in the images or detection of changes in the scene.

1.3 SUMMARY OF THE CHAPTERS TO FOLLOW

Chapter 2 covers the preprocessing operations used in image registration. This includes image restoration, image smoothing/sharpening, and image segmentation. Chapter 3 discusses methods for detecting corners, lines, curves, regions, templates, and patches. Chapter 4 discusses methods for determining the correspondence between features in the images, and Chapter 5 covers various transformation functions for registration of rigid as well as nonrigid images. Various image resampling methods are covered in Chapter 6 and evaluation of the performance of an image registration method is discussed in Chapter 7. Finally, three main applications of image registration are covered. Chapter 8 discusses image fusion, Chapter 9 discusses image mosaicking, and Chapter 10 covers stereo depth perception.

1.4 BIBLIOGRAPHICAL REMARKS

One of the first examples of image registration appeared in the work of Roberts [325]. By aligning projections of edges of model polyhedral solids with image edges, he was able to locate and recognize predefined polyhedral objects. The registration of entire images first appeared in remote sensing literature. Anuta [8, 9] and Barnea and Silverman [23] developed automatic methods for the registration of images with translational differences using the sum of absolute differences as the similarity measure. Leese *et al.* [237] and Pratt [315] did the same using the cross-correlation coefficient as the similarity measure. The use of image registration in robot vision was pioneered by Mori *et al.* [279], Levine *et al.* [241], and Nevatia [286]. Image registration found its way to biomedical image analysis as data from various scanners measuring anatomy and function became digitally available [20, 361, 397].

Image registration has been an active area of research for more than three decades. Survey and classification of image registration methods may be found in papers by Gerlot and Bizais [140], Brown [48], van den Elsen *et al.* [393], Maurer and Fitzpatrick [268], Maintz and Viergever [256], Lester and Arridge [239], Pluim *et al.* [311], and Zitova and Flusser [432].

A book covering various landmark selection methods and their applications is due to Rohr [331]. A collection of papers reviewing methods particularly suitable for registration of medical images has been edited into a book entitled *Medical Image Registration* by Hajnal *et al.* [175]. Separate collections of work covering methods for registration of medical images have been edited by Pernus *et al.* in a special issue of *Image and Vision Computing* [304] and by Pluim and Fitzpatrick in a special

issue of *IEEE Trans. Medical Imaging* [312]. A collection of work covering general methodologies in image registration has been edited by Goshtasby and LeMoigne in a special issue of *Pattern Recognition* [160] and a collection of work covering topics on nonrigid image registration has been edited by Goshtasby *et al.* in a special issue of *Computer Vision and Image Understanding* [166].

2
Preprocessing

The images to be registered often have scale differences and contain noise, motion blur, haze, and sensor nonlinearities. Pixel sizes in satellite and medical images are often known and, therefore, either image can be resampled to the scale of the other, or both images can be resampled to the same scale. This resampling facilitates the feature selection and correspondence steps. Depending on the features to be selected, it may be necessary to segment the images. In this chapter, methods for noise and blur reduction as well as methods for image segmentation are discussed.

2.1 IMAGE ENHANCEMENT

To facilitate feature selection, it may be necessary to enhance image intensities using smoothing or deblurring operations. Image smoothing reduces noise but blurs the image. Deblurring, on the other hand, reduces blur but enhances noise. The size of the filter selected for smoothing or deblurring determines the amount of smoothing or sharpening applied to an image.

2.1.1 Image smoothing

Image smoothing is intended to reduce noise in an image. Since noise contributes to high spatial frequencies in an image, a smoothing operation should reduce the magnitude of high spatial frequencies. Smoothing can be achieved by *convolution* or *filtering*. Given image $f(x, y)$ and a symmetric convolution operator h of size $(2k + 1) \times (2l + 1)$ with coordinates varying from $-k$ to k horizontally and from $-l$

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

to l vertically, the image value at (x, y) after convolution, $\bar{f}(x, y)$, is defined by

$$\bar{f}(x,y) = \sum_{i=-k}^{k} \sum_{j=-l}^{l} f(x+i, y+j)h(i,j). \tag{2.1}$$

Convolution, as can be observed, is a linear operation. For image filtering to be independent of the image orientation, the filter kernel should be radially symmetric.

To reduce *zero-mean* or *white* noise, *mean* or *Gaussian* filtering is performed, while to reduce *impulse* or *salt-and-pepper* noise, *median* filtering is performed. Median filtering involves finding the median value in a local neighborhood and is not a linear operation and, therefore, it cannot be computed by the convolution operation.

When carrying out Gaussian, mean, or median filtering, the size of the filter determines the amount of smoothing applied to an image. Larger filters reduce image noise more, but they blur the image more also. On the other hand, smaller filters do not blur the image as much, but they may leave considerable noise in the image. Filter size should be selected to provide a compromise between the amount of noise removed and the amount of detail retained in an image.

2.1.1.1 **Median filtering** Denoting the image before smoothing by f, the image after smoothing by $\bar{f}$, assuming image f contains $M \times N$ pixels, and the filter is of radius r pixels, median filtering is computed from

$$\bar{f}(i,j) = MEDIAN(f,i,j,r), \qquad i = 0,\ldots,M-1, \quad j = 0,\ldots,N-1, \tag{2.2}$$

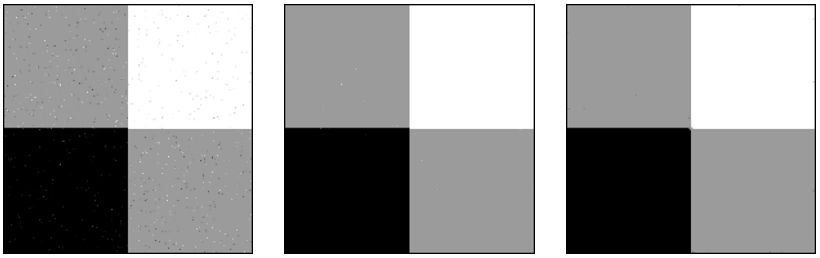
where $MEDIAN(f,i,j,r)$ is a function that returns the median intensity in image f in a circular window of radius r centered at (i, j). If a part of the window falls outside the image, intensities within the portion of the window falling inside the image are used to compute the median. As mentioned earlier, circular windows are used to make smoothing independent of image orientation.

The effect of filter size in median filtering is shown in Fig. 2.1. The image with impulse noise is shown in Fig. 2.1a. Results of median filtering using filters of radius 2 and 4 pixels are shown in Figs 2.1b and 2.1c, respectively. By increasing the filter size, more noise is removed. Since impulse noise usually presides over a small percentage of image pixels (in Fig. 2.1a only 2% of the pixels contain noise), a small window is often sufficient to remove it.

2.1.1.2 **Mean filtering** This is pure image averaging and the intensity at a pixel in the output is determined from the average of intensities in the input within a circular window centered at the pixel position in the output. Mean filtering is computed from

$$\bar{f}(i,j) = MEAN(f,i,j,r), \qquad i = 0,\ldots,M-1, \quad j = 0,\ldots,N-1, \tag{2.3}$$

where $MEAN(f,i,j,r)$ is a function that returns the average of intensities within the circular area of radius r centered at (i, j) in image f. The filter size determines the



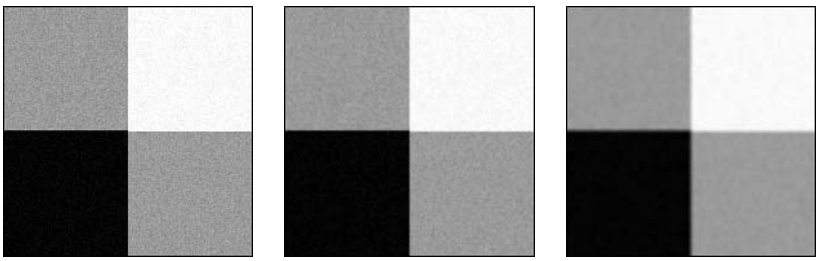
(a) (b) (c)

Fig. 2.1 (a) An image with 2% impulse noise. (b), (c) Noise reduction by median filtering using filters of radius 2 and 4 pixels, respectively.

neighborhood size where averaging is performed. As the window size is increased, more noise is removed, but the image is also more blurred. The example in Fig. 2.2 demonstrates this fact. Figure 2.2a shows an image containing zero-mean noise. Figures 2.2b and 2.2c show results of mean filtering the image using filters of radius 2 and 4 pixels, respectively.

Although mean filtering is very easy to implement, it is not the best filter for image smoothing. A smoothing filter should reduce the magnitude of higher spatial frequencies more. If we examine the frequency response of a mean filter by determining its Fourier transform, we observe that its frequency response does not monotonically decrease [55]. A mean filter allows some very high spatial frequencies to be reproduced while it completely removes some mid-frequencies, resulting in image artifacts. A filter, such as a Gaussian, which has a monotonically decreasing frequency response, is more suitable for image smoothing.

2.1.1.3 **Gaussian filtering** The Fourier transform of a Gaussian is also a Gaussian [55]. Therefore, Gaussian smoothing reduces higher spatial frequencies more than the lower spatial frequencies. Gaussian filters are also computationally very efficient



(a) (b) (c)

Fig. 2.2 (a) An image containing zero-mean noise. (b), (c) Noise reduction by mean filtering using filters of radius 2 and 4 pixels, respectively.

because they can be separated into 1-D Gaussians, enabling computations in 1-D:

$$\begin{aligned} G(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\} \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{x^2}{2\sigma^2}\right\} \times \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{y^2}{2\sigma^2}\right\} \\ &= G(x) \times G(y). \end{aligned} \tag{2.4}$$

Therefore, instead of smoothing an image with a 2-D Gaussian, the image is smoothed first with a 1-D Gaussian row-by-row and then, with the same 1-D Gaussian, column-by-column. If image size is $N \times N$, computation of the value at an output pixel using two 1-D convolutions requires $2N$ multiplications, while computation using a 2-D convolution requires N^2 multiplications.

Computations can be further speeded up if they are carried out in the frequency (Fourier) domain. For instance, to smooth a 1-D image by a Gaussian, the Fourier transform of the image and the Fourier transform of the Gaussian are determined and they are point-by-point multiplied. Then, the inverse Fourier transform of the result is computed to obtain the smoothed image. Assuming a 1-D image contains N pixels, computation of its Fourier transform by the fast Fourier transform (FFT) takes $N \log_2 N$ multiplications [46, 75], it takes N multiplications to compute the filtering operation in the Fourier domain, and finally, it takes $N \log_2 N$ multiplications to find the inverse Fourier transform of the result by the FFT algorithm. In total, therefore, it takes $N + 2N \log_2 N$ multiplications to carry out a smoothing. If smoothing is performed directly, computation of each smoothed pixel in the output requires N multiplications, and since there are N pixels in the 1-D image, N^2 multiplications are needed. For a 2-D image of size $N \times N$ pixels, computation using the Fourier transform requires on the order of $N^2 \log_2 N$ multiplications, while the direct method requires on the order of N^3 multiplications. Calculation of smoothing using the FFT algorithm is faster than that by direct calculation. However, the FFT algorithm requires that dimensions of the image be a power of 2, such as 128 or 256.

If dimensions of an image are not a power of 2, computations cannot be speeded up using the FFT algorithm but, since a Gaussian vanishes exponentially, it can be truncated 3 or 4 standard deviations away from its center without having any noticeable effect on the result. For instance, if smoothing is performed using a Gaussian of standard deviation 1 pixel, it is sufficient to find the weighted average of 9 pixels within a 1-D image to produce a pixel value in the output. For small standard deviations, this direct computation may be faster than computation using the FTT algorithm. One should also notice that Fourier transform assumes that an image is cyclic, that is, the first image row follows the last image row and the first image column is a continuation of the last image column. Therefore, if image intensities near the top and bottom or near the left and right image borders are different, computation of image smoothing by FFT results in artifacts. Therefore, depending on image content, image smoothing using direct computation may be more accurate and faster than computation by the FFT algorithm.

Results of image smoothing using Gaussians of standard deviation 2 and 4 pixels are shown in Fig. 2.3 using direct computation and computation using the FFT algorithm. A clean image is shown in Fig. 2.3a. After adding white noise to it, the image in Fig. 2.3d is obtained. Smoothing this image with Gaussians of standard deviation 2 and 4 pixels by direct computation the images in Figs 2.3b and 2.3c are obtained, and by the FFT algorithm the images in Figs 2.3e and 2.3f, are obtained. Inaccuracies near image borders are evident when computation is carried out in the Fourier domain.

Image smoothing in 3-D is the same as image smoothing in 2-D except that spherical kernels are used instead of circular kernels. If a separable filter, such as a Gaussian, is used, smoothing in 3-D can be achieved by a combination of 1-D filtering operations, first performed row-by-row, then column-by-column, and finally slice-by-slice.

2.1.2 Deblurring

Deblurring, also known as *inverse filtering*, is the process of reducing blur in an image. Deblurring is used to reduce image blur caused by camera defocus. Assuming $F(u, v)$ and $G(u, v)$ are the Fourier transforms of image $f(x, y)$ before and after blurring, and assuming $H(u, v)$ is the Fourier transform of the blurring source, deblurring is

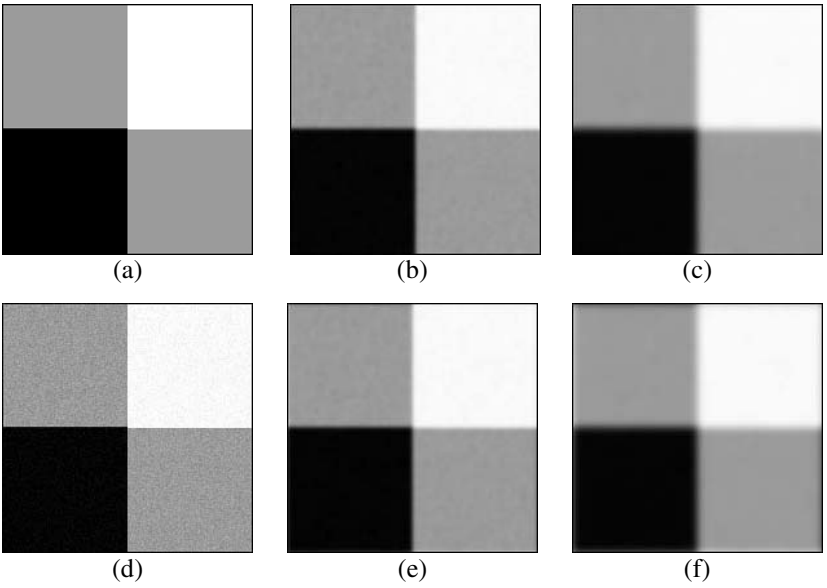


Fig. 2.3 (a) A noise-free image. (b), (c) Image smoothing by direct computation using Gaussians of standard deviation 2 and 4 pixels. (d) Image (a) after addition of zero-mean noise. (e), (f) Image smoothing using the FFT algorithm with Gaussians of standard deviation 2 and 4 pixels, respectively.

the process of estimating the image before it was blurred from

$$\hat{f}(x,y) = \mathcal{F}^{-1}\left\{\frac{G(u,v)}{H(u,v)}\right\}, \tag{2.5}$$

where $\hat{f}(x,y)$ denotes estimation of image $f(x,y)$ and $\mathcal{F}^{-1}$ denotes the inverse Fourier transform [142]. Therefore, if information about the blurring source is known, a blurred image can be sharpened by determining its Fourier transform, dividing it point-by-point by the Fourier transform of the blurring filter, and computing the inverse Fourier transform of the result. Note that inverse filtering is possible only when the Fourier transform of the blurring filter does not contain any zeros.

If the degradation source can be modeled by a rank-one filter, computation of inverse filtering can be achieved efficiently without the Fourier transform. Rank-one operators are those that can be separated into a combination of 1-D operators. For example, operator

$$\mathbf{T} = \begin{bmatrix} ac & a & ad \\ c & 1 & d \\ bc & b & bd \end{bmatrix} \tag{2.6}$$

can be separated into

$$\mathbf{r} = \begin{bmatrix} a \\ 1 \\ b \end{bmatrix} \tag{2.7}$$

and

$$\mathbf{s} = [c \;\; 1 \;\; d]\,. \tag{2.8}$$

Therefore $\mathbf{T}$ it is a rank-one operator.

Convolving an image with filter $\mathbf{T}$ is the same as convolving the image with filter $\mathbf{r}$ followed by filter $\mathbf{s}$. Similarly, inverse filtering an image with filter $\mathbf{T}$ is the same as inverse filtering the image with $\mathbf{s}$ and then with $\mathbf{r}$. In the following, an efficient algorithm for computing inverse filtering when the filter under consideration is rank-one is described. Computation of inverse filtering for filters of form $\mathbf{r}$ is discussed. Inverse filtering for filters of form $\mathbf{s}$ can be determined by inverse filtering the transpose of the image with the transpose of filter $\mathbf{s}$ and then transposing the obtained result.

Assuming $\mathbf{f}$ is an $M \times N$ image, convolving the image with filter $\mathbf{r}$ can be written as

$$\mathbf{g}(j) = \mathcal{F}^{-1}\left\{\mathcal{F}[\mathbf{f}(j)] \cdot \mathcal{F}(\mathbf{r})\right\} \qquad j = 0, \ldots, N-1 \tag{2.9}$$

where $\mathbf{f}(j)$ and $\mathbf{g}(j)$ are the jth columns of the image before and after filtering, respectively. The dot denotes point-by-point multiplication, and $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote Fourier and inverse Fourier transforms, respectively. Now, given the filtered (blurred) image $\mathbf{g}$ and the blurring filter $\mathbf{r}$, the image before blurring is computed from

$$\mathbf{f}(j) = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}[\mathbf{g}(j)]}{\mathcal{F}(\mathbf{r})}\right\} \qquad j = 0, \ldots, N-1 \tag{2.10}$$

where the division is again point-by-point. For this operation to be possible, none of the coefficients in the Fourier transform of $\mathbf{r}$ should be zero.

For rank-one filters, computation of inverse filtering does not require the use of Fourier transform. If by convolving image $\mathbf{f}$ of size $M \times N$ with filter $\mathbf{r}$ image $\mathbf{g}$ is obtained, then

$$g(x,y) = \sum_{i=-1}^{1} r(i) f(x, y+i); \qquad x = 0, \ldots, M-1; \quad y = 0, \ldots, N-1; \tag{2.11}$$

where $g(x,y)$ is the xyth entry in the convolved image, $r(-1) = a$, $r(0) = 1$, and $r(1) = b$. In this formula, $f(x,-1)$ and $f(x,N)$ are assumed to be zero for $x = 0, \ldots, M-1$. These assumptions will result in some inaccuracies in estimations of values at image borders. Formula (2.11) may be written in matrix form by $\mathbf{g} = \mathbf{Bf}$, where

$$\mathbf{B} = \begin{bmatrix} 1 & b & & & & & \\ a & 1 & b & & & & \\ & a & 1 & . & & & \\ & & a & . & b & & \\ & & & . & 1 & b & \\ & & & & a & 1 & \end{bmatrix}_{M \times M}. \tag{2.12}$$

Note that matrix $\mathbf{B}$ is completely determined when filter $\mathbf{r}$ is given. The problem in inverse filtering, therefore, is determining the original image $\mathbf{f}$ given the blurred image $\mathbf{g}$. Taking advantage of the special form of matrix $\mathbf{B}$, image $\mathbf{f}$ can be determined column by column. Assuming $\mathbf{f}(j)$ and $\mathbf{g}(j)$ are the jth columns of $\mathbf{f}$ and $\mathbf{g}$, respectively, $\mathbf{f}(j)$ can be determined by solving

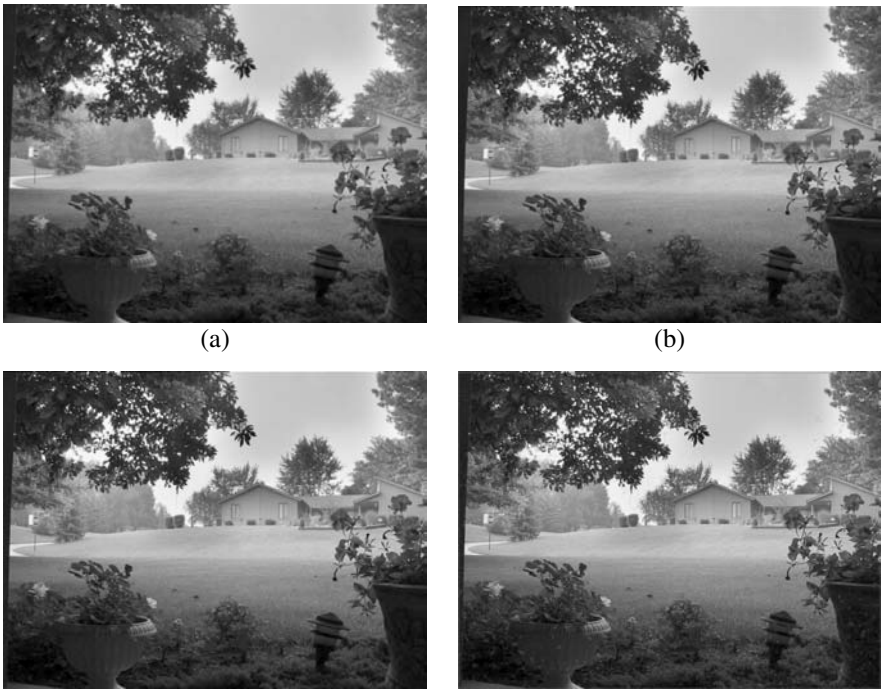
$$\mathbf{Bf}(j) = \mathbf{g}(j). \tag{2.13}$$

To solve equation (2.13), $\mathbf{B}$ is replaced by $b\mathbf{D}$, where

$$\mathbf{D} = \begin{bmatrix} \alpha & 1 & & & & & \\ \beta & \alpha & 1 & & & & \\ & \beta & \alpha & . & & & \\ & & \beta & . & 1 & & \\ & & & . & \alpha & 1 & \\ & & & & \beta & \alpha & \end{bmatrix}, \tag{2.14}$$

$\alpha = 1/b$, and $\beta = a/b$. Now, since matrix $\mathbf{D}$ can be easily factored into the product of $\mathbf{LU}$ [257] with

$$\mathbf{L} = \begin{bmatrix} 1 & & & & & \\ l_0 & 1 & & & & \\ & l_1 & 1 & . & & \\ & & l_2 & . & & \\ & & & . & 1 & \\ & & & & l_{M-2} & 1 \end{bmatrix}, \tag{2.15}$$



(a) (b) (c) (d)

Fig. 2.4 (a) An outdoor scene image. (b)–(d) Sharpening of the image by inverse filtering using filter $\mathbf{T}$ of (2.6) with $a = b = c = d = 0.16, 0.32$, and 0.48, respectively.

$$\mathbf{U} = \begin{bmatrix} u_0 & 1 & & & & \\ & u_1 & 1 & & & \\ & & u_2 & \ddots & & \\ & & & \ddots & 1 & \\ & & & \ddots & u_{M-2} & 1 \\ & & & & & u_{M-1} \end{bmatrix}, \tag{2.16}$$

$u_0 = \alpha$, $l_{i-1} = \beta/u_{i-1}$, and $u_i = \alpha - l_{i-1}$, for $i = 1, \ldots, M-1$, we can rewrite equation (2.13) by

$$b\mathbf{LU}\mathbf{f}(j) = \mathbf{g}(j). \tag{2.17}$$

In equation (2.17), by forward substitution an unknown vector $\mathbf{Y}$ is determined using $\mathbf{LY} = \mathbf{g}(j)$, and using $b\mathbf{U}\mathbf{f}(j) = \mathbf{Y}$, $\mathbf{f}(j)$ is determined by back substitution. Assuming $Y(i)$ is the ith element of $\mathbf{Y}$, $g(i, j)$ is the ijth element of $\mathbf{g}$, and $f(i, j)$ is the ijth element of $\mathbf{f}$, the following algorithm computes $\mathbf{f}$ given $\mathbf{g}$ and $\mathbf{r}$.

Algorithm 2.1: Direct computation of inverse filtering

1: Determine $\mathbf{L}$ and $\mathbf{U}$.
2: For $j = 0, \ldots, N-1$

2.1: Set $Y(0) = g(0, j)$.
2.2: Compute $Y(i) = g(i, j) - l_{i-1}Y(i-1)$ for $i = 1, \ldots, M-1$.
2.3: Set $f(M-1, j) = Y(M-1)/bu_{M-1}$.
2.4: Compute $f(i, j) = [Y(i)/[b - f(i+1, j)]/u_i$ for $i = (M-2), \ldots, 0$.

Computation of each element of matrix $\mathbf{f}$ requires only four multiplications and divisions. It has been shown [257] that the LU-decomposition for matrix $\mathbf{D}$ exists only when $a, b < 0.5$.

Computation of inverse filtering of an $N \times N$ image by the FFT algorithm takes in the order of $N^2 \log N$ multiplications. Computation of inverse filtering (for a 3×3 rank-one filter) by Algorithm 2.1 takes in the order of N^2 multiplications. For large values of N, this computational saving can be significant.

An example of inverse filtering is given in Fig. 2.4. Inverse filtering of image 2.4a using filter $\mathbf{T}$ with $a = b = c = d = 0.16, 0.32$, and 0.48 is shown in Figs 2.4b–d. When $a = b = c = d = 0.48$ we see that the amount of deblurring applied to the image is too much, resulting in artifacts.

2.2 IMAGE SEGMENTATION

Image segmentation is the process of partitioning an image into meaningful parts and is perhaps the most studied topic in image analysis. This can be attributed to the importance of segmentation in image analysis and the fact that a universal method does not exist that can segment all images. A method is usually developed taking into consideration the properties of a particular class of images.

Segmentation methods can be grouped into thresholding, boundary detection, and region growing. Thresholding methods assign pixels with intensities below a threshold value into one class and the remaining pixels into another class and form regions by connecting adjacent pixels of the same class. Thresholding methods work well on simple images where the objects and background have different intensity distributions.

Boundary extraction methods use information about intensity differences between adjacent regions to separate the regions from each other. If the intensities within a region vary gradually but the difference of intensities between adjacent regions remains large, boundary detection methods can successfully delineate the regions. Region growing methods form regions by combining pixels of similar properties.

The objective of this section is not to exhaustively review image segmentation methods, but rather to describe a few effective methods that can be used to prepare an image before feature selection.

2.2.1 Intensity thresholding

In image segmentation by thresholding, one or more threshold values are interactively or automatically selected and used to segment an image. When a single threshold value is used, image intensities equal to or greater than it are assigned to one class and the remaining intensities are assigned to another class. Then, regions are formed by connecting adjacent pixels of the same class. Intensity thresholding works well in

images where intensity distributions of the objects and the background are Gaussians and have different means.

Images containing two types of regions, one belonging to the objects and one belonging to the background, produce bimodal histograms. The threshold value is selected at the valley between the two modes. Since a histogram is usually noisy, to avoid selection of a local minimum as the threshold value, the histogram is smoothed before selecting the threshold value. If after smoothing, the valley contains a flat segment, the midpoint in the horizontal segment is taken as the threshold value. If the valley cannot be clearly located, a method to deepen the valley may be used [410, 411]. These methods either do not count pixels with gradient magnitudes above a threshold value, or they count higher-gradient pixels with smaller weights when creating the histogram.

Although some images may contain objects and background that have different Gaussian distributions, if the Gaussians are too close to each other a bimodal histogram may not be obtained. On the other hand, an image containing a complex scene with nonhomogeneous objects may produce a bimodal histogram. Pixels in an image can be randomly rearranged without changing the histogram of the image. Therefore, the intensity at the valley between the two modes in a bimodal histogram may not segment an image properly. A better approach will be to use gradient information to select pixels that belong to region boundaries and use intensities of those pixels to determine the threshold value. The average intensity of high gradient pixels may be used as the threshold value because high-gradient pixels represent the boundaries of objects or their parts. Selection of the right percentage of these high gradient pixels is critical when computing the threshold value [364].

Alternatively, the intensity at which the change in region size becomes minimum can be taken as the threshold value. Region boundaries have high gradients and changing the threshold value corresponding to an intensity at region boundaries will not change the region sizes significantly. To find the threshold value where change in pixel count becomes minimum, the number of pixels falling on one side of a threshold value must be determined and pixel count must be tracked as the threshold value is changed. This process is computationally more expensive than finding the average intensity of high gradient pixels, but the obtained threshold value will be optimal [421].

Examples of image segmentation by intensity thresholding are shown in Fig. 2.5. Figure 2.5a is the same as Fig. 1.1a except that it is smoothed by a Gaussian of standard deviation 1.5 pixels to reduce noisy details in the image. Figure 2.5b shows its histogram. The histogram has three peaks and two valleys. Thresholding this image using the intensity at the valley between the first two peaks results in the segmentation shown in Fig. 2.5c. Using the average intensity of pixels with the highest 5% gradients as the threshold value the segmentation shown in Fig. 2.5d is obtained. Changing the percentage of highest gradient pixels changes the threshold value and that changes the segmentation result. The best threshold value can be considered the one that is shared by most high gradient pixels in the image. Such a threshold value minimally changes the region sizes as the threshold value is changed.

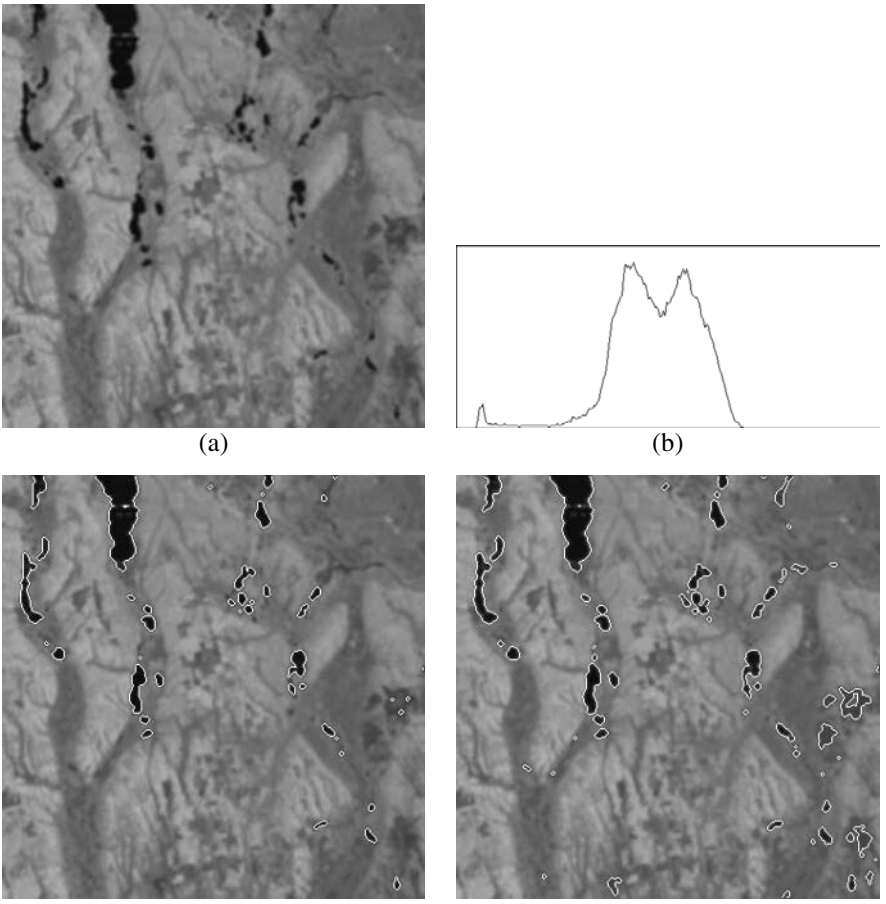


Fig. 2.5 (a) The Landsat MSS image of Fig. 1.1a after smoothing with a Gaussian of standard deviation 1.5 pixels. (b) The histogram of the image. (c) Segmentation using the threshold value corresponding to the valley between the first two modes in the histogram after the histogram is smoothed with a Gaussian of standard deviation 2. (d) Segmentation using a threshold value equal to the average intensity of the highest 5% gradient pixels in the image.

2.2.2 Boundary detection

Boundary contours or edges are significant image features that are needed in various image analysis applications. Edge detection is an efficient means of finding boundaries of objects or their parts in an image. Edges represent sharp changes in image intensities, which could be due to discontinuities in scene reflectance, surface orientation, or depth. Image pixels representing such discontinuities carry more information than pixels representing gradual change or no change in intensities.

Two main approaches to edge detection exist. One approach determines the zero-crossings of the second derivative of image intensities, while the second approach finds locally maximum gradient magnitudes of image intensities in the gradient direction. The zero-crossing method is easier to implement, but it detects a mixture of true and false edges, requiring removal of the false edges by a postprocessing operation. In the following, a number of edge detection methods are reviewed.

2.2.2.1 **The Laplacian of a Gaussian edge detector** Edges are considered image pixels where intensity change is locally maximum. Edges in a 1-D image can be found by computing the gradient (first derivative) of the image and locating pixels that have locally maximum gradient magnitudes. An alternative approach is to find the second derivative of the image intensities and locate the zero-crossings. In 2-D, the second derivative is computed by the Laplacian operator, and edges are obtained by finding the Laplacian of an image and locating the pixels that separate positive and negative regions.

The Laplacian operator is defined by $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$, and the Laplacian of image $f(x, y)$ is defined by $(\partial^2 f(x, y)/\partial x^2 + \partial^2 f(x, y)/\partial y^2)$. In the digital domain, the Laplacian can be approximated by

$$\mathbf{T} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{2.18}$$

Operator $\mathbf{T}$ is equivalent to the sum of operators $\mathbf{r} = [-1\ 2\ -1]^t$ and $\mathbf{s} = [-1\ 2\ -1]$, where t denotes transpose. Therefore, the Laplacian of an image is obtained by convolving the image with $\mathbf{r}$ and $\mathbf{s}$ separately and adding the convolved images together. Also, note that $[-1\ 2\ -1]$ is obtained by convolving the difference operator $\mathbf{d} = [1\ -1]$ with itself. $\mathbf{d}$ is a difference operator and computes the first derivative of an image horizontally, so $\mathbf{s}$ is a second-derivative operator horizontally. Similarly, we find that $\mathbf{r}$ represents a second-derivative operator vertically.

To avoid detection of noisy edges, an image is smoothed before its Laplacian is computed. Smoothing or convolving an image with a Gaussian and then determining its Laplacian is the same as convolving the image with the Laplacian of Gaussian (LoG). That is,

$$\begin{aligned} LoG[f(x, y)] &= \frac{\partial^2[f(x, y) \star G(x, y)]}{\partial x^2} + \frac{\partial^2[f(x, y) \star G(x, y)]}{\partial y^2} \\ &= f(x, y) \star \frac{\partial^2 G(x, y)}{\partial x^2} + f(x, y) \star \frac{\partial^2 G(x, y)}{\partial y^2} \end{aligned} \tag{2.19}$$

where $\star$ denotes convolution. Since a 2-D Gaussian can be separated into two 1-D Gaussians, to speed up the computations, in formula (2.19), $G(x, y)$ can be replaced with $G(x)G(y)$; therefore, the LoG of an image can be computed from

$$LoG[f(x, y)] = \frac{\partial^2 G(x)}{\partial x^2} \star G(y) \star f(x, y) + G(x) \star \frac{\partial^2 G(y)}{\partial y^2} \star f(x, y). \tag{2.20}$$

Edge detection by the LoG operator was proposed by Marr and Hildreth [261] in a pioneering paper on edge detection.

Zero-crossing edges always produce closed boundaries because they are formed by thresholding a LoG image at zero and finding the boundaries between positive and negative regions. Closed boundaries are very desirable because they often correspond to object boundaries. When a boundary contour breaks into pieces, it becomes difficult to delineate objects or their parts.

The zero-crossings of the second derivative of an image correspond to locally maximum as well as locally minimum gradients. Clark [67] has shown that the zero-crossing pixels that produce first and third derivatives of the same sign correspond to locally minimum gradients and zero-crossings that produce first and third derivatives of different signs correspond to locally maximum gradients. This provides a mechanism for distinguishing true from false edges among the detected zero-crossings.

As the standard deviation of the Gaussian smoother is increased, fewer edges are obtained, and the edge contours become rounder and displace from their true positions. A method known as edge focusing starts by finding edges at a coarse resolution (a rather high standard deviation of Gaussian). The standard deviation of the Gaussian smoother is then gradually reduced while tracking the edges from low to high resolution. The process allows edges to accurately position themselves while avoiding weaker edges entering the picture. It has been shown that if the standard deviation of Gaussians is changed by half a pixel, the edges move by less than a pixel, except near places where edge contours break into two or more contours [28]. Therefore, as the standard deviation of the Gaussian is reduced with half-pixel steps, it is only necessary to search for the new edge positions by searching within a ribbon of width three pixels centered at the old edge contours. An efficient method to track the edges from coarse to fine is described in [154].

An example of edge detection by the LoG operator is shown in Fig. 2.6. The zero-crossings of the image in Fig. 2.6a using a Gaussian of standard deviation 2.5 pixels are shown in Fig. 2.6b. Removing the false edges from among the zero-crossings results in the edges shown in Fig. 2.6c. The arteries, which are the objects of interest, have been detected but some edge contours have become disconnected after removal of the zero-crossings corresponding to locally minimum gradients. As we will see below some of the false edges that connect the true edges are needed to delineate the object boundaries. The edges determined by edge focusing are shown in Fig. 2.6d. Some critical edges are missed here also. The missing edges represent the false edges of the contour segments that break into new segments and displace by more than half a pixel, causing the tracking process to lose them. As a result, a contour is cut at the point it branches to new contours from low to high resolution.

2.2.2.2 **Canny edge detector** Canny has formulated a procedure that ensures 1) good detection, 2) good localization, and 3) a single response to a true edge [51]. Criterion one ensures than the edge detector has a low probability of missing a real edge and a low probability of detecting a false edge. Criterion two ensures that the

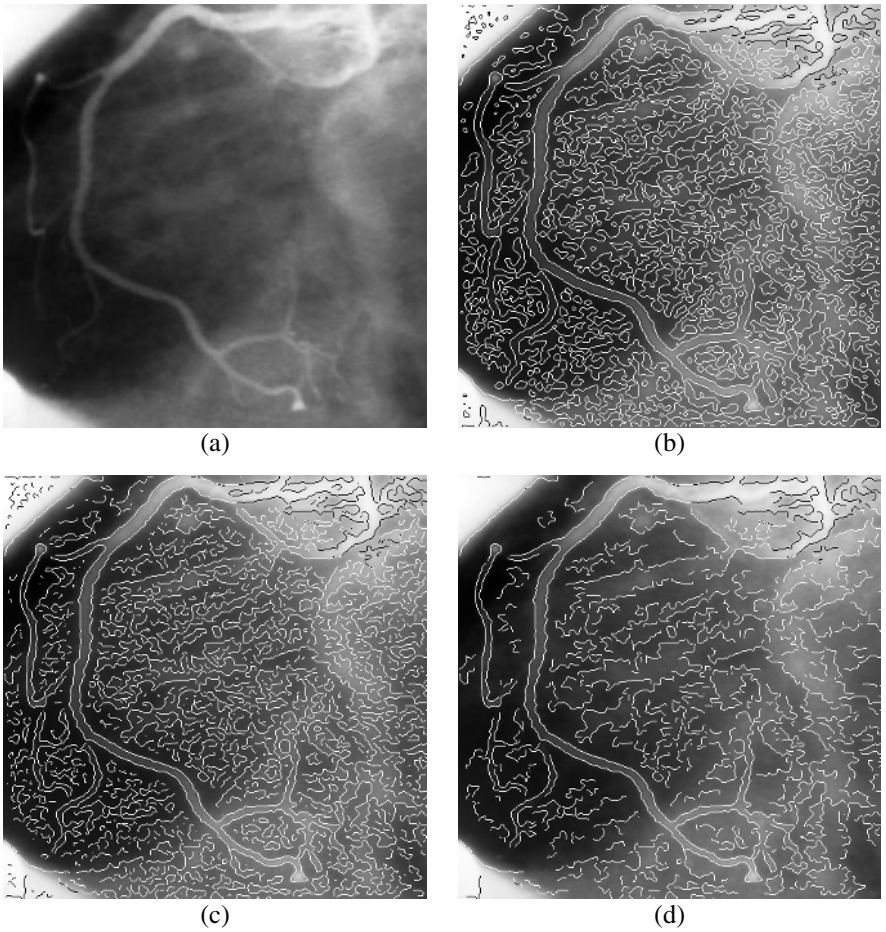


Fig. 2.6 (a) An X-ray angiogram. (b) Zero-crossing edges obtained by the LoG operator of standard deviation 2.5 pixels. (c) Zero-crossings after removal of the false edges. (d) Edges determined by edge focusing starting from $\sigma = 5$ pixels and stopping at $\sigma = 2.5$ pixels with half-pixel steps and after removal of false edges.

edge detector positions an edge as close as possible to the center of the real edge. Criterion three ensures that only one edge is obtained per a true edge. Canny finds that the edge detector satisfying the three criteria can be approximated by detecting locally maximum gradient magnitudes in the gradient direction. The Canny edges of Fig. 2.6a are shown in Fig. 2.7 using a Gaussian of standard deviation 2.5 pixels.

Representing gradient magnitudes in an image as elevations in a terrain scene, locally maximum gradient magnitudes correspond to the ridge points. All ridge points, however, do not correspond to locally maximum gradients in the gradient direction. Some correspond to locally minimum gradients. Note that the gradient

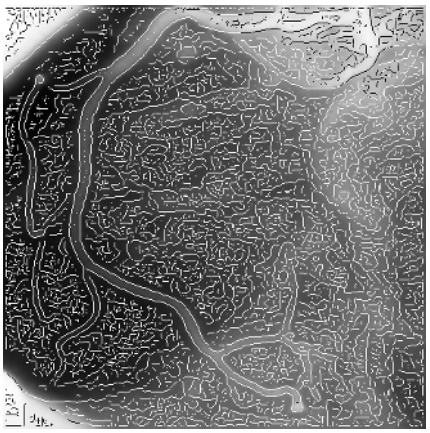


Fig. 2.7 Canny edges of Fig. 2.6a using a Gaussian smoother of standard deviation 2.5 pixels.

directions at two sides of a ridge point have opposite signs and the gradient magnitudes at ridge points vary rather slowly. When walking along a ridge contour, if change in the gradient of the ridge contour is greater than the gradient in the direction normal to it, the edge contour representing the ridge will not be detected and the edge contour will be fragmented. To avoid an edge contour from being fragmented, locally minimum gradients that are connected from both sides to locally maximum gradients should also be considered as edges and kept [95].

An example demonstrating this phenomenon is shown in Fig. 2.8. Suppose image 2.8a represents the gradient magnitudes of an image. Clearly, one closed edge contour should be obtained from this image. However, by labeling pixels with locally maximum gradients in the gradient direction as the edges, the edges shown in Fig. 2.8b are obtained. Representing the intensities as elevations, the 3-D elevation map shown in Fig. 2.8c is obtained. Although ridges in this image should be detected as edges, the ridge points do not represent locally maximum gradients in the gradient direction. Therefore, when only locally maximum gradients in the gradient direction are detected, some critical edges are missed, causing a boundary contour to break. By considering locally minimum gradients that are connected from both sides to locally maximum gradients as edges, the edge contour in Fig. 2.8d is obtained. This edge contour now represents the intended boundary more accurately than the fragmented edge contours shown in Fig. 2.8b.

2.2.2.3 **Edge detection by intensity ratios** A 2-D image represents the projection of a 3-D scene onto a plane, recording intensities proportional to brightnesses in the scene. Perceived brightness at a point depends on the illumination as well as the reflectance and orientation of the surface at the point. Therefore, recorded intensity

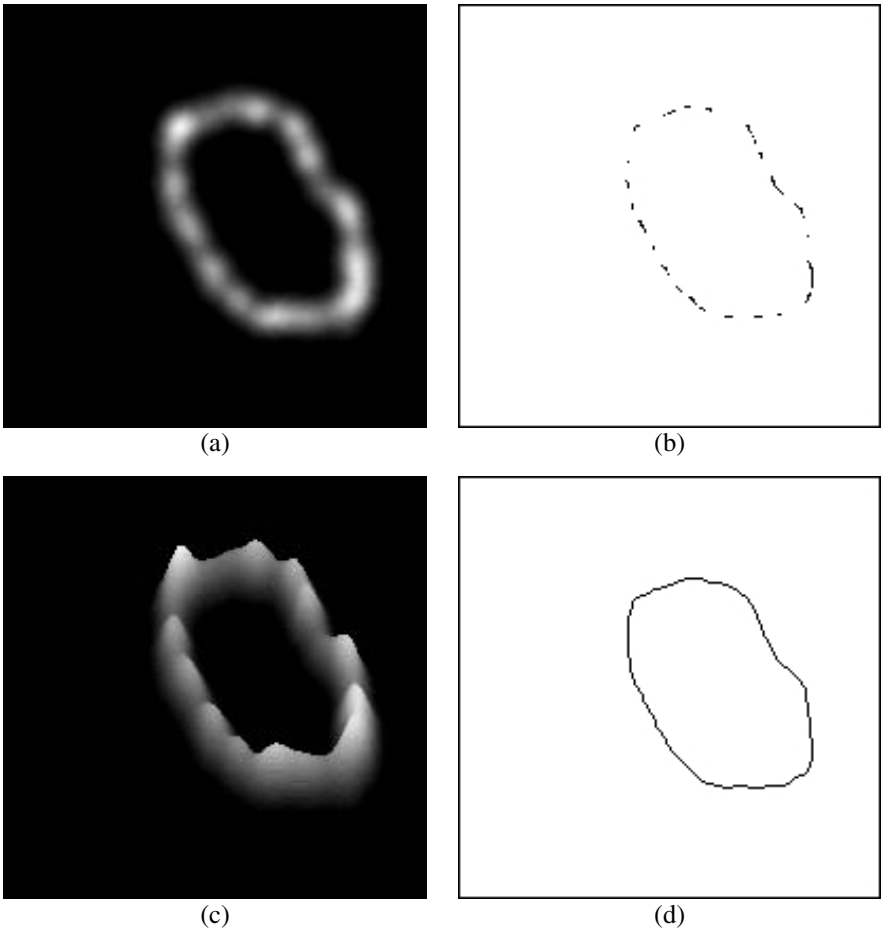


Fig. 2.8 (a) A synthetic gradient image with a clear region boundary. (b) The Canny edges, representing locally maximum gradients in the gradient direction. (c) Image (b) when intensities are considered as elevations. (d) The ridge contour of (c).

f at point (x, y) can be described by [192]

$$f(x, y) = i(x, y)r(x, y)\cos\theta(x, y), \tag{2.21}$$

where $i(x, y)$ is the illumination at the scene point whose projection in the image is point (x, y), and $r(x, y)$ and $\theta(x, y)$ are the reflectance and the angle of the surface normal with the direction of light. All these factors contribute to the recorded intensities; however, edges separating objects from each other and from the background, correspond to scene points where reflectance and surface normals change sharply. In the following, we will refer to metric $r(x, y)\cos\theta(x, y)$ as the *surface property* and

will denote it by $p(x, y)$. Therefore, formula (2.21) can be written as

$$f(x, y) = i(x, y)p(x, y), \tag{2.22}$$

where $f(x, y)$ is the recorded intensity at location (x, y) in an image, $i(x, y)$ is the scene illumination reaching the surface point whose projection in the image is (x, y), and $p(x, y)$ is the property of the surface at (x, y).

Intensity gradient in the x direction at point (x, y) can be computed from

$$\begin{aligned} D_x\left[f(x, y)\right] &= \frac{\partial f(x, y)}{\partial x} \\ &= \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}. \end{aligned} \tag{2.23}$$

The intensity gradient between two points separated by Δx is computed from the difference of intensities at the two points. Substituting relation (2.22) into relation (2.23) and letting $\Delta x = 1$, we obtain

$$D_x\left[f(x, y)\right] = i(x + 1, y)p(x + 1, y) - i(x, y)p(x, y). \tag{2.24}$$

Now, if two points separated by a small distance in an image receive about the same illumination, that is, if

$$i(x + 1, y) \approx i(x, y), \tag{2.25}$$

then relation (2.24) can be approximated by

$$D_x[f(x, y)] = i(x, y)[p(x + 1, y) - p(x, y)]. \tag{2.26}$$

If relation (2.26) is used to compute gradients at points x_1 and x_2 in an image, we obtain

$$D_x\left[f(x_1, y)\right] = i(x_1, y)\left[p(x_1 + 1, y) - p(x_1, y)\right] \tag{2.27}$$

and

$$D_x\left[f(x_2, y)\right] = i(x_2, y)\left[p(x_2 + 1, y) - p(x_2, y)\right], \tag{2.28}$$

respectively. When surface properties at (x_1, y) and (x_2, y) are the same and surface properties at $(x_1 + 1, y)$ and $(x_2 + 1, y)$ are the same also, we expect that gradients at x_1 and x_2 be the same. However, we see that if illuminations at (x_1, y) and (x_2, y) are different, gradients at (x_1, y) and (x_2, y) will be different too. If, instead of intensity

differences, intensity ratios are used as the metric to characterize scene changes, we will have

$$R_x\left[f(x_1,y)\right] = \frac{f(x_1+1,y)}{f(x_1,y)} = \frac{i(x_1+1,y)p(x_1+1,y)}{i(x_1,y)p(x_1,y)}. \tag{2.29}$$

In image areas where relation (2.25) holds, we can approximate relation (2.29) by

$$R_x\left[f(x_1,y)\right] = \frac{p(x_1+1,y)}{p(x_1,y)}. \tag{2.30}$$

Similarly, the property ratio at point (x_2, y) is determined from

$$R_x\left[f(x_2,y)\right] = \frac{p(x_2+1,y)}{p(x_2,y)}. \tag{2.31}$$

As can be observed, if surface property at (x_1, y) is the same as that at (x_2, y) and the surface property at $(x_1 + 1, y)$ is the same as that at $(x_2 + 1, y)$, we will find $R_x\left[f(x_1,y)\right] = R_x\left[f(x_2,y)\right]$. Therefore, metric R, which depends only on surface property (surface reflectance and surface normal), more accurately characterizes scene content than metric D, which depends on both surface property and scene lighting.

D_x is the intensity gradient in the x direction. We will refer to R_x as the property ratio in the x direction. A positive intensity gradient shows change in intensities in one direction, while a negative intensity gradient shows change in intensities in the opposite direction. A zero gradient shows no change in intensities. The property ratio, however, is always positive. Values greater than 1.0 show change in surface properties in one direction while values less than 1.0 show change in surface properties in the opposite direction. A ratio of 1.0 shows no change in surface properties. Locally maximum intensity gradients were called intensity edges. We will refer to locally maximum property ratios that are greater than 1.0 and locally minimum property ratios that are less than 1.0 as the property edges. In practice, we will replace property ratios that are less than 1.0 with their inverses and determine the local maxima of obtained values to detect the property edges, therefore, in the following we assume $R_x \geq 1$.

In many situations, intensity edges are the same as property edges, but they are not always the same. For instance, consider the 1-D image $[18\ \ 16\ \ 13\ \ 9\ \ 6\ \ 3\ \ 2\ \ 2]$. The gradient magnitudes of this image computed by intensity differences (with $\Delta x = 1$) are $[2\ \ 3\ \ 4\ \ 3\ \ 3\ \ 1\ \ 0]$, while the gradients of the image computed from intensity ratios are $[1.1\ \ 1.2\ \ 1.4\ \ 1.5\ \ 2\ \ 1.5\ \ 1]$. The position of the intensity edge is at the third point, while the position of the property edge is at the fifth point.

The intensity ratio method not only produces edges that are more representative of discontinuities in surface properties compared to the intensity difference method, but it also produces edges that are less sensitive to noise. Since noise is usually additive,

in a homogeneous area where intensity variations are mainly due to noise, the output from operator D_x will be mostly due to noise, while the output from operator R_x will depend on the noise as well as the signal.

Edges computed from intensity ratios partition an image into surfaces of different properties as they do not depend on scene illumination. However, on shadow boundaries where relation (2.25) does not hold, detected edges do not represent change in surface properties and represent the shadow boundaries. Gershon *et al.* [141] have developed a method for separating edges representing changes in surface material from edges representing changes in scene illumination (shadow boundaries) in color images.

Note that when metric R_x is greater than one, by taking the logarithm of both sides of relation (2.30) and replacing x_1 with x, we obtain

$$\log R_x\left[f(x,y)\right] = \log \frac{p(x+1,y)}{p(x,y)} \tag{2.32}$$

$$= \log \frac{i(x+1,y)p(x+1,y)}{i(x,y)p(x,y)} \tag{2.33}$$

$$\approx \log \frac{f(x+1,y)}{f(x,y)} \tag{2.34}$$

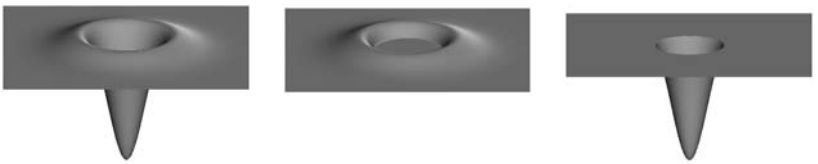
$$= \log f(x+1,y) - \log f(x,y) \tag{2.35}$$

$$= \frac{\partial\left[\log f(x,y)\right]}{\partial x}. \tag{2.36}$$

Since local maxima of $R_x\left[f(x,y)\right]$ are the same as local maxima of $\log R_x\left[f(x,y)\right]$ when $R_x > 1$, relation (2.36) states that edges determined by intensity ratios are the same as edges determined by the difference of the logarithm of the intensities.

Measurements on simple animal eyes have shown that relation between the light intensity input to visual receptors and the neural output level is approximately logarithmic [76, 367]. Experimental results on the human visual system have also shown a logarithmic sensitivity to light intensity [201, 418]. From relation (2.36) we can conclude that edges determined by intensity ratios are similar to edges perceived by biological vision systems.

The above discussions focused on computation of gradients in surface property in the x direction. Gradients in surface property in the y direction can be determined similarly, and gradients in surface property in an arbitrary direction will be a vector with x and y components, representing property changes along x and y directions. When an image is noisy or contains small details, the image should be smoothed before computing the gradients. Edges are located by finding the local maxima of property changes in the direction of maximum change. This procedure is similar to that of the Canny edge detector with the exception that, instead of using intensity differences, intensity ratios are used to compute gradients. To determine edges corresponding to the zero-crossings of the second derivative of surface properties, first, an image f is convolved with the positive part of the LoG operator (see Fig. 2.9b) to obtain an



(a) (b) (c)

Fig. 2.9 (a) A LoG operator. (b) The positive part of the operator. Negative values in the original operator are set to 0 while keeping all positive values as they are. (c) The negative part of the operator. Here all positive values in the original operator are set to zero and the negative values are kept.

image, which we denote by f_1. Then image f is convolved with the negative part of the LoG operator (see Fig. 2.9c). The negative part of LoG should be inverted so that all values become positive. Let's call the obtained image f_2. The logarithm of the ratio of f_1 and f_2 is computed at each pixel and the zero-crossings of the result are located to find the edges.

An example of edge detection by intensity ratios using the X-ray angiogram of Fig. 2.6a is shown in Fig. 2.10a. Edges detected by intensity differences are depicted in Fig. 2.10c. We see that arterial edges are more thoroughly detected by intensity ratios than by intensity differences. Removing the weakest 70% of edges in both cases, the edges shown in Figs 2.10b and 2.10d are obtained. We see that more edges in dark areas are detected by intensity ratios, while more edges in bright areas are detected by intensity differences. In this particular image, edges detected by intensity ratios delineate the objects of interest, which are the arteries, better than the edges detected by intensity differences.

2.2.2.4 **Edge detection by curve fitting** Curves are particularly useful if the ultimate objective in edge detection is to find the locally maximum curvature points or inflection points in region boundaries. In a method described by Goshtasby [155], gradient magnitude at each pixel is determined and connected pixels with gradient magnitudes above a threshold value are formed into regions. The regions are partitioned into elongated subregions and a curve is fitted to each subregion.

Consider a region that is formed by high-gradient pixels in an image. The region may correspond to a complex structure with many branches and loops. First, such a region is subdivided into simple ones, each containing only very short branches. Then, a curve is fitted to pixels within each simple region.

A region is reduced to a minimum-spanning tree (MST) [316] and tree branches that are longer than a threshold value are cut off. Each obtained branch is treated like a new tree and the subdivision is repeated until all trees have a trunk with branches that are shorter than a threshold value. A curve is then fitted to points in each simple tree trunk.

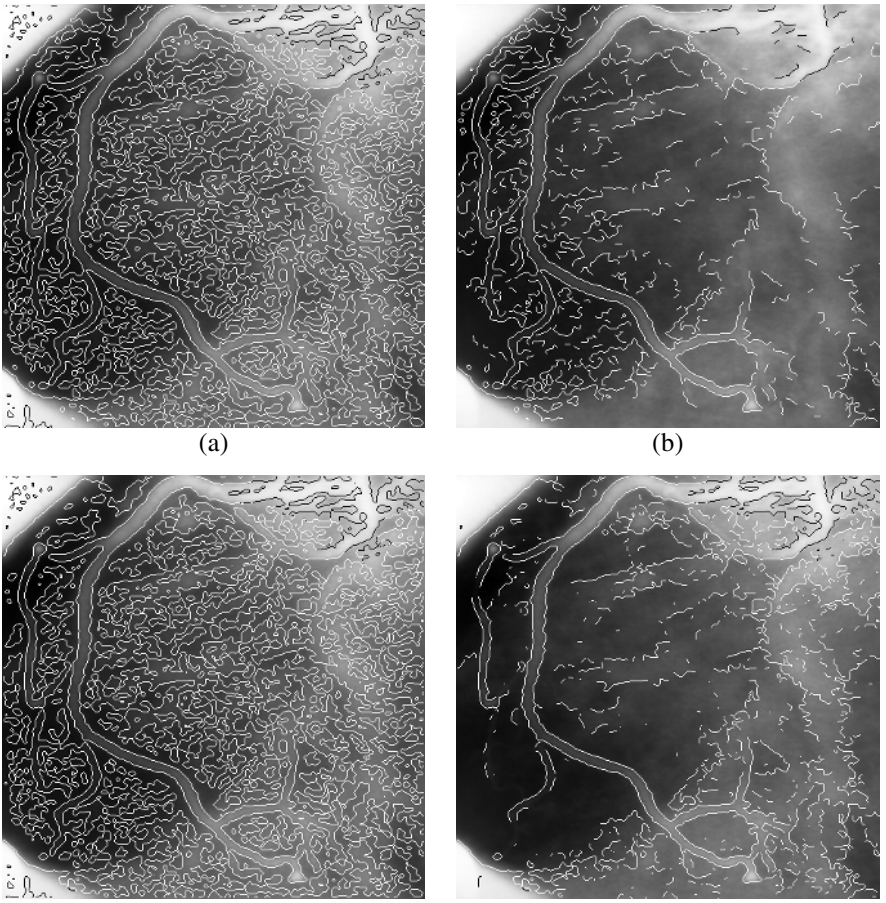


Fig. 2.10 (a) Zero-crossing edges of Fig. 2.6a determined by intensity ratios. (b) 30% strongest edges of (a). (c) Zero-crossing edges determined by intensity differences. (d) 30% strongest edges of (c).

Since the distance from a pixel to pixels to its left, right, above, and below is the same, many MSTs can be obtained from a single region. In order to obtain MSTs with trunks representing the spines of trees, the gradients and intensities of pixels are taken into consideration when choosing the MST edges. An MST is grown gradually, starting from a single pixel until all pixels in a region are included. At any stage, the procedure that selects the next pixel in a region among the unprocessed ones takes the pixel which is the closest to the MST. If two or more pixels can be selected as the next MST vertex, the pixel with the highest gradient magnitude is chosen. Choosing MST vertices in this manner will ensure that pixels with locally-maximum gradient magnitudes form the spine in a region.

If a pixel is of the same distance to two or more MST vertices, it could be connected to the MST in two or more different ways. In such a situation, the pixel is connected to the MST vertex that produces the largest absolute intensity difference between the two points. Selecting MST edges in this manner will ensure that branches are connected to the spine in the direction of the maximum gradient.

A complex region may have an MST that has many branches. In order to obtain longer curve segments, the maximal path (longest path) in the MST is determined. Then, branches connected to the maximal path that are longer than a threshold value are cut off. A branch is treated like a new tree, and if it has long branches, they are recursively cut until all trees have a trunk with branches that are shorter than a threshold value.

Given a set of pixels $\{\mathbf{V}_i : i = 1, \ldots, n\}$ representing a region, it is required that the order in which the curve approximates the points be determined. In a parametric curve, this ordering is provided by the nodes of the curve: $\{u_i : i = 1, \ldots, n\}$. The nodes of a curve show the positions of the curve's basis functions in the parameter space. Node u_i shows the parameter value at which the effect of the ith pixel is maximum on the curve. The nodes of a curve are determined such that the effect of a pixel is maximum at the curve point closest to it. If the curve is given, the curve point closest to a pixel can be determined. However, since a curve cannot be generated before its nodes are determined, the maximal path (spine) of the MST of a region is used as an approximation to the curve and the nodes are estimated. For points that lie on the maximal path, their distances to one end of the path are divided by the length of the path and used as the nodes. For points that are not on the maximal path, their projections to the path (points on the path closest to them) are determined and distances of the projections to the same end of the path are divided by the length of the maximal path to obtain the nodes. Estimating the nodes in this manner ensures that the effect of a pixel becomes maximum at the curve point closest to it.

Because an MST cannot contain loops, the procedure described above will replace a closed boundary by an open curve. In order to avoid that, whenever two branch end-points in an MST are adjacent, they are connected and a closed curve is fitted to it. Branches attached to the spine which are longer than a threshold value are again cut off and treated like new trees.

A curve formulation that can approximate a large number of points without solving a system of equations is the rational Gaussian (RaG) formulation defined by [153]

$$\mathbf{P}(u) = \sum_{i=1}^{n} \mathbf{V}_i g_i(u) \tag{2.37}$$

where $\mathbf{V}_i$ is the ith control point (pixel in a region),

$$g_i(u) = \frac{W_i G_i(u)}{\sum_{j=1}^{n} W_j G_j(u)} \tag{2.38}$$

is the ith blending function of the curve, W_i is the weight (gradient magnitude) associated with the ith pixel, and

$$G_i(u) = e^{-(u-u_i)^2/2\sigma^2} \tag{2.39}$$

is a Gaussian of height one, and u_i in formula (2.39), known as the ith node of the curve, is the parameter value at which the ith blending function is centered. σ is the standard deviation of the Gaussians and shows the smoothness of the curve. The standard deviation σ and parameter u are measured with the same unit.

Formulas (2.37)–(2.39) show an open curve. To obtain a closed curve, formula (2.39) is replaced with

$$G_i(u) = \sum_{j=-\infty}^{\infty} e^{-[u-(u_i+j)]^2/2\sigma^2}. \tag{2.40}$$

The infinity in this formula comes from the fact that a Gaussian extends from $-\infty$ to $+\infty$ and when a curve is closed, it makes infinite cycles. A Gaussian, however, approaches zero exponentially and in practice the infinity may be replaced by a small number. Assuming the accuracy of a computer is ϵ and the standard deviations of all Gaussians are equal to σ, we find

$$-\left\lceil \sigma\sqrt{-2\ln\epsilon} \right\rceil \leq j \leq \left\lceil \sigma\sqrt{-2\ln\epsilon} \right\rceil. \tag{2.41}$$

In digital images, it has been shown [155] that it is sufficient to vary j in the range [-5,5]. Using larger values of j will not change the curve fitting result.

A nice property of this curve is that it does not require the solution of a system of equations to obtain it. A curve is immediately obtained by substituting the coordinates of given points into relations (2.37)–(2.39). The standard deviation of Gaussians in this formula controls the smoothness of the obtained curve. This is because the Fourier transform of a Gaussian in the spatial domain is another Gaussian in the frequency domain [55] and, as the standard deviation of the Gaussian in the spatial domain is increased, the standard deviation of the Gaussian in the frequency domain decreases, showing that the obtained curve will have smaller high-spatial-frequency coefficients. Since a curve with smaller high-spatial-frequency coefficients implies a smoother curve, we can conclude that as the standard deviation of Gaussians increases, a smoother curve is generated. Hence, the ability to vary the standard deviation of Gaussians makes it possible to control the smoothness of generated edge contours.

The weight at a control point determines the degree of importance of that point with respect to others in defining a curve. The weight at a point may be set to the gradient magnitude at the point to ensure that the obtained curve passes closer to points with higher gradient magnitudes.

Since images come from a variety of scanners, have different levels of noise, and are in different resolutions, the employed edge detection method should have

parameters that can be adjusted to these image variations. This curve-fitting method has three parameters that can be varied to meet these needs.

1. The gradient threshold g, which selects the pixels in a region.
2. The maximum branch-length l, which ensures that a curve is fitted to a tree with only small branches. This threshold value also removes noisy regions that contain fewer than l pixels.
3. The standard deviation of Gaussians σ, which controls the smoothness of the obtained curves.

When selecting the gradient threshold value, the degree of detail in an image should be taken into consideration. In a noisy image, the threshold value should be increased to avoid detection of noisy edges. In a blurred image, the threshold value should be decreased to pick up low-gradient region boundaries. As the gradient threshold is decreased, the number of points used in curve fitting increases, slowing down the edge detection process. As the gradient threshold is increased, the process becomes faster but some of the edge contours become disconnected.

The maximum branch length in a tree determines the positional accuracy of an obtained curve. An MST with long branches will produce a curve that may not reconstruct local image details well. On the other hand, if very small branches are cut off, very short curve segments will be obtained. A textured image containing a lot of details requires a small threshold value so that image details are not averaged. A blurred image containing thick region boundaries requires a large threshold value so that multiple contours are not generated for the same region boundary.

The standard deviation of Gaussians determines the smoothness of obtained curves. If σ is too large, edge contours with sharp corners cannot be reproduced. On the other hand, if σ is too small, unwanted noisy details may appear in a generated contour. For a noisy image, a larger σ should be used to avoid detection of noisy edges. For a high-contrast image with very little noise, a smaller σ should be used to allow a curve to reproduce sharp corners.

An example of edge detection by curve fitting is shown in Fig. 2.11b. The gradient threshold value was 10, branches in a tree containing more than 10 pixels were cut off and treated as new trees, regions containing fewer than 10 pixels were considered noise and removed from the image, and the standard deviation of the Gaussian was 2 pixels. Edges obtained by the LoG operator with $\sigma = 2$ pixels and gradient magnitudes greater than ten are shown in Fig. 2.11d. Comparing edges determined by curve fitting and by the LoG operator, we see that edges detected by curve fitting are longer and smoother, while edges obtained by the LoG operator are better positioned. Since edge contours obtained by curve fitting are in continuous form, it is possible to generate them in an arbitrary scale.

2.2.2.5 **Edge detection by functional approximation** In this method an image is considered a single-valued surface and the local shape of the surface is approximated by a biquadratic patch. Gradients of the patches are then determined and edges are located at locally maximum gradients in the gradient direction. A biquadratic patch is fitted to intensities in a 3×3 neighborhood centered at each pixel by the least-squares

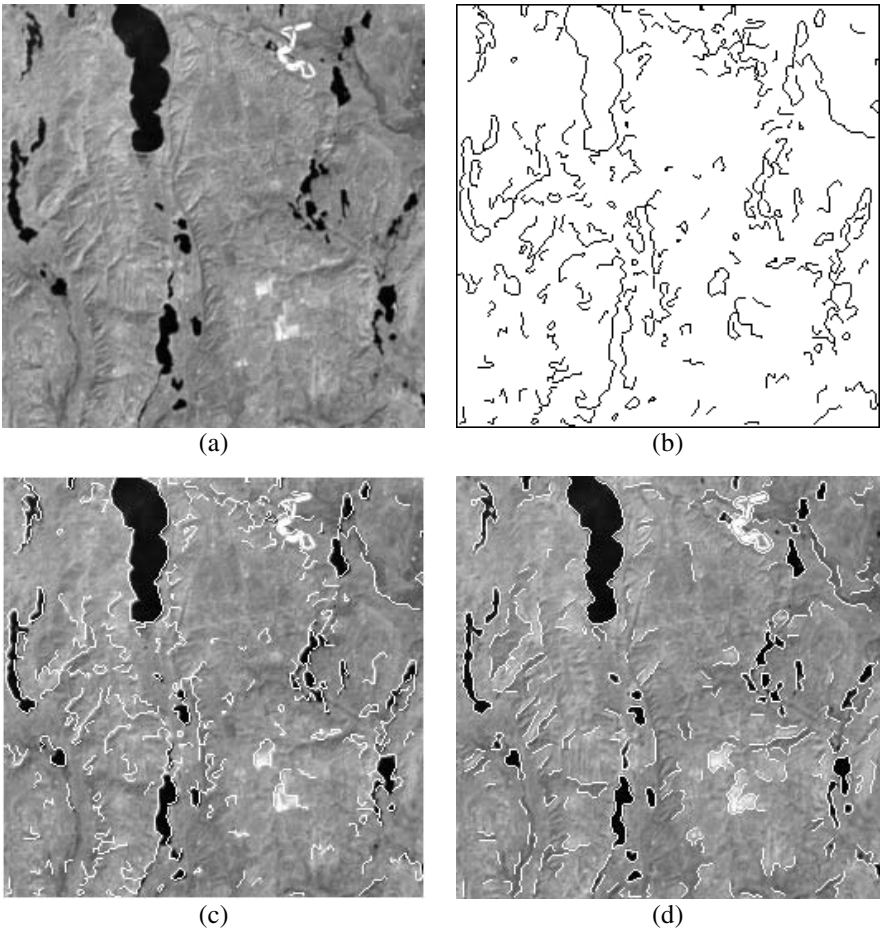


Fig. 2.11 (a) A Landsat TM image. (b) Edges detected by curve fitting using gradient threshold 10, minimum branch length 10 pixels, and $\sigma = 2$ pixels. (c) The edges are overlaid with the image to evaluate the quality of detected edges. (d) Edges detected by the LoG operator with gradient threshold 10 and $\sigma = 2$ pixels.

method. Since the shape of a patch is independent of the position of the 3×3 window and depends only on the intensity arrangements in it, the window is assumed to be centered at $(0, 0)$. So, biquadratic surface

$$S(x, y) = a + bx + cy + dxy + ex^2 + fy^2 \tag{2.42}$$

is fitted to

$$\{[x, y, f(x, y)] : x = -1, 0, 1; y = -1, 0, 1\} \tag{2.43}$$

by minimizing

$$E_2 = \sum_{x=-1}^{1}\sum_{y=-1}^{1}\left[S(x,y) - f(x,y)\right]^2$$
$$= \sum_{x=-1}^{1}\sum_{y=-1}^{1}\left[a + bx + cy + dxy + ex^2 + fy^2 - f(x,y)\right]^2, \quad (2.44)$$

where $f(x,y)$ denotes the intensity at (x,y). To find parameters a through f that minimize E_2, we find partial derivatives of E_2 with respect to the parameters, set them equal to zero, and solve the obtained system of linear equations. Since

$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} 1 = 9; \qquad \sum_{x=-1}^{1}\sum_{y=-1}^{1} x = 0;$$
$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} y = 0; \qquad \sum_{x=-1}^{1}\sum_{y=-1}^{1} xy = 0;$$
$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} xy^2 = 0; \qquad \sum_{x=-1}^{1}\sum_{y=-1}^{1} x^2y = 0;$$
$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} x^3 = 0; \qquad \sum_{x=-1}^{1}\sum_{y=-1}^{1} y^3 = 0;$$
$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} x^3y = 0; \qquad \sum_{x=-1}^{1}\sum_{y=-1}^{1} xy^3 = 0;$$
$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} x^2y^2 = 4; \qquad \sum_{x=-1}^{1}\sum_{y=-1}^{1} x^2 = 6;$$
$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} y^2 = 6; \qquad \sum_{x=-1}^{1}\sum_{y=-1}^{1} x^4 = 6;$$
$$\sum_{x=-1}^{1}\sum_{y=-1}^{1} y^4 = 6; \quad (2.45)$$

and letting

$$B_1 = \sum_{x=-1}^{1}\sum_{y=-1}^{1} f(x,y); \qquad B_2 = \sum_{x=-1}^{1}\sum_{y=-1}^{1} xf(x,y);$$
$$B_3 = \sum_{x=-1}^{1}\sum_{y=-1}^{1} yf(x,y); \qquad B_4 = \sum_{x=-1}^{1}\sum_{y=-1}^{1} xyf(x,y);$$

$$B_5 = \sum_{x=-1}^{1}\sum_{y=-1}^{1} x^2 f(x,y); \qquad B_6 = \sum_{x=-1}^{1}\sum_{y=-1}^{1} y^2 f(x,y); \tag{2.46}$$

we obtain

$$\begin{bmatrix} 9 & 0 & 0 & 0 & 6 & 6 \\ 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 6 & 0 & 0 & 0 & 6 & 4 \\ 6 & 0 & 0 & 0 & 4 & 6 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{bmatrix}, \tag{2.47}$$

which produces the following solution:

$$\begin{aligned} a = (5B_1 - 3B_5 - 3B_6)/9; \qquad & b = B_2/6; \\ c = B_3/6; \qquad & d = B_4/4; \\ e = B_5/2 - B_1/3; \qquad & f = -B_1/3 + B_6/2. \end{aligned} \tag{2.48}$$

The relations in (2.48) describe coefficients a–f in terms of the intensities of the pixels within a 3×3 window. The gradients of the patch in x and y directions are

$$S_x(x,y) = b + dy + 2ex, \tag{2.49}$$

$$S_y(x,y) = c + dx + 2fy. \tag{2.50}$$

At the center of the patch, gradients along x and y are obtained by setting $x = 0$ and $y = 0$ in equations (2.49) and (2.50). Doing so, we obtain $S_x = b$ and $S_y = c$. Therefore, gradient magnitude and gradient direction at the center of the patch are $\sqrt{b^2 + c^2}$ and $\tan^{-1}(c/b)$, respectively.

Now, since

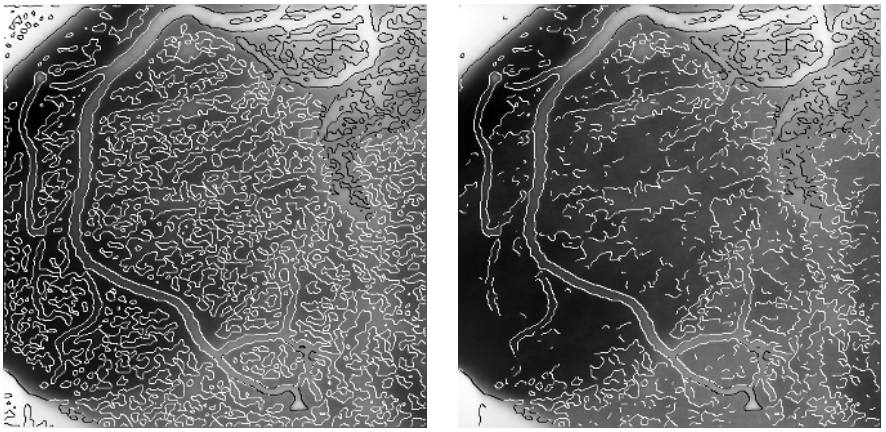
$$S_{xx}(x,y) = 2e, \tag{2.51}$$

$$S_{yy}(x,y) = 2f, \tag{2.52}$$

the Laplacian of the patch is obtained from

$$S_{xx}(x,y) + S_{yy}(x,y) = 2(e+f). \tag{2.53}$$

To find the zero-crossings of the Laplacian of the image, coefficients of the patch approximating each 3×3 neighborhood are computed and the zero-crossings of $(e+f)$ are located. To distinguish the zero-crossings that correspond to locally maximum gradients from the zero-crossings that correspond to locally minimum gradients, at each zero-crossing point, the gradient magnitudes at both sides of the zero-crossing in the gradient direction are determined. If the gradient at the zero-crossing is larger than those at its two sides, then the zero-crossing represents a locally maximum gradient;



(a) (b)

Fig. 2.12 (a) Edges of the image in Fig. 2.6a determined by functional approximation with a Gaussian smoother of standard deviation 2.5 pixels. (b) Strong edges of (a).

otherwise, it represents a locally minimum gradient. Since the neighborhood over which a patch is determined is very small, noisy edges may be obtained. To avoid detection of noisy edges, an image is first smoothed before approximating it locally by biquadratic functions. To avoid detection of edges corresponding to the discrete nature of image intensities, it is important that all computations, including smoothing, are performed using floating-point numbers.

Although it is required to fit a biquadratic surface to each 3×3 neighborhood, because the coefficients of the surface are determined by a small number of multiplications as demonstrated in (2.48), it takes in the order of N^2 multiplications to find the zero-crossing edges for an image of size $N \times N$ pixels. This computation does not include the time needed to smooth the image.

An example of edge detection by functional approximation is shown in Fig. 2.12. Figure 2.12a shows the zero-crossing edges of the X-ray angiogram in Fig. 2.6a obtained by functional approximation. The image was smoothed with a Gaussian of standard deviation 2.5 pixels before determining its edges. Removing the weak edges, the image shown in Fig. 2.12b is obtained. The quality of edges detected by functional approximation are similar to those detected by the LoG operator.

2.2.2.6 **Edge detection in 3-D images** The procedure for detecting edges in 3-D closely follows that in 2-D. The LoG operator in 3-D is computed from

$$\begin{aligned} LoG\left[f(x,y,z)\right] &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) G(x,y,z) \star f(x,y,z) \\ &= \frac{\partial^2 G(x)}{\partial x^2} \star G(y) \star G(z) \star f(x,y,z) \end{aligned}$$

$$+\frac{\partial^2 G(y)}{\partial y^2} \star G(x) \star G(z) \star f(x,y,z)$$

$$+\frac{\partial^2 G(z)}{\partial z^2} \star G(x) \star G(y) \star f(x,y,z). \tag{2.54}$$

Edges are considered to be voxels representing the boundary between positive and negative regions in the 3-D LoG image. To detect the edges, the LoG image is scanned in x, y, and z directions separately and changes in the sign of image values are detected and marked as edges. Since zero-crossings most often fall between image voxels, their positions can be determined with subvoxel accuracy. If subvoxel accuracy is not required, the magnitude of the LoG values of two adjacent voxels that have different signs are compared and the voxel with the smaller magnitude LoG is taken to represent the zero-crossing edge.

Zero-crossing edges correspond to locally maximum as well as locally minimum gradients and locally minimum gradients represent false edges. Since false edges usually have small gradient magnitudes, by removing edges with gradient magnitudes below a threshold value, false edges as well as weak edges can be removed from an image.

The Canny edge detector in 3-D requires the computation of the 3-D gradients and the location of voxels that have locally maximum gradient magnitudes in the gradient direction. Gradients in 3-D represent 3-D vectors with three components. Assuming an image should be smoothed with a 3-D Gaussian $G(x,y,z)$ before computing its gradient, the three components of a gradient vector are obtained from

$$\begin{aligned} f_x(x,y,z) &= \frac{\partial G(x,y,z) \star f(x,y,z)}{\partial x} \\ &= \frac{\partial G(x,y,z)}{\partial x} \star f(x,y,z) \\ &= \frac{\partial G(x)}{\partial x} \star G(y) \star G(z) \star f(x,y,z), \end{aligned} \tag{2.55}$$

$$f_y(x,y,z) = \frac{\partial G(y)}{\partial y} \star G(x) \star G(z) \star f(x,y,z), \tag{2.56}$$

$$f_z(x,y,z) = \frac{\partial G(z)}{\partial z} \star G(x) \star G(y) \star f(x,y,z). \tag{2.57}$$

The gradient magnitude at (x,y,z) is obtained from

$$M(x,y,z) = \left\{ [f_x(x,y,z)]^2 + [f_y(x,y,z)]^2 + [f_z(x,y,z)]^2 \right\}^{\frac{1}{2}} \tag{2.58}$$

and gradient direction at (x,y,z) is a unit vector with components

$$\mathbf{u}(x,y,z) = \left(\frac{f_x(x,y,z)}{M(x,y,z)}, \frac{f_y(x,y,z)}{M(x,y,z)}, \frac{f_z(x,y,z)}{M(x,y,z)} \right). \tag{2.59}$$

Note that convolutions in 2-D as well as in 3-D are performed by a combination of 1-D convolutions to achieve high speed.

Intensity ratio edges in 3-D are obtained by convolving the positive and negative parts of operator

$$\frac{\partial^2 G(x)}{\partial x^2} \star G(y) \star G(z) + \frac{\partial^2 G(y)}{\partial y^2} \star G(x) \star G(z) + \frac{\partial^2 G(z)}{\partial z^2} \star G(x) \star G(y) \quad (2.60)$$

separately with the image, creating two images, and then finding the ratio of the images point by point. The process closely follows that described for 2-D images.

Edge contours in 2-D become edge surfaces in 3-D, so edge detection by curve fitting becomes edge detection by surface fitting. Edges in 3-D, however, form complex surface structures, which cannot easily be represented by parametric surfaces. Edge detection in 3-D by surface fitting is possible if implicit surfaces are used [34, 292]. For images containing simple objects with spherical or cylindrical topologies, superquadrics [21], hyperquadrics [69], and rational Gaussian surfaces [165] have been used.

By considering a 3-D image as discrete samples from a continuous volume, the continuous volume within a $3 \times 3 \times 3$ neighborhood can be approximated by a triquadratic function by the least-squares method. The gradient magnitude and gradient direction at each image voxel can then be estimated from the gradient magnitude and gradient direction at $(x = 0, y = 0, z = 0)$ of the function. This process will find edges that correspond to image voxels with locally-maximum gradient magnitudes in the gradient direction. Note that since triquadratic functions are fitted to very small neighborhoods, to avoid detection of noisy edges, it is important that the image is smoothed with a Gaussian before carrying out functional approximation.

Examples of 3-D edges are shown in Fig. 2.13. Edges determined by the LoG operator with a Gaussian of standard deviation 2 voxels are shown in Fig. 2.13b. The weakest 70% of the edges have been removed. Canny edges obtained using a Gaussian of standard deviation 2 voxels and interactively removing the weak edges are shown in Fig. 2.13c. Figure 2.13d shows edges obtained by the intensity ratios. Again, the standard deviation of the Gaussian smoother was 2 pixels and the weak edges were interactively removed to keep the same number of edges as those found by the Canny method and the LoG operator. The majority of edges determined by the three methods are the same. There are some differences between the edges detected by the three methods.

2.2.2.7 **Edge detection in color images** In gray scale images, sharp changes in intensities are considered edges. In color images, sharp changes in color values are considered edges. In a color image, three color values are present at each pixel; therefore, edge detection requires the detection of sharp changes in a 3-D vector field over a plane. Denoting color gradients in x and y directions by $\mathbf{u}(x, y)$ and $\mathbf{v}(x, y)$, respectively, we have

$$\mathbf{u}(x, y) = \frac{\partial R(x, y)}{\partial x}\mathbf{r} + \frac{\partial G(x, y)}{\partial x}\mathbf{g} + \frac{\partial B(x, y)}{\partial x}\mathbf{b}, \quad (2.61)$$

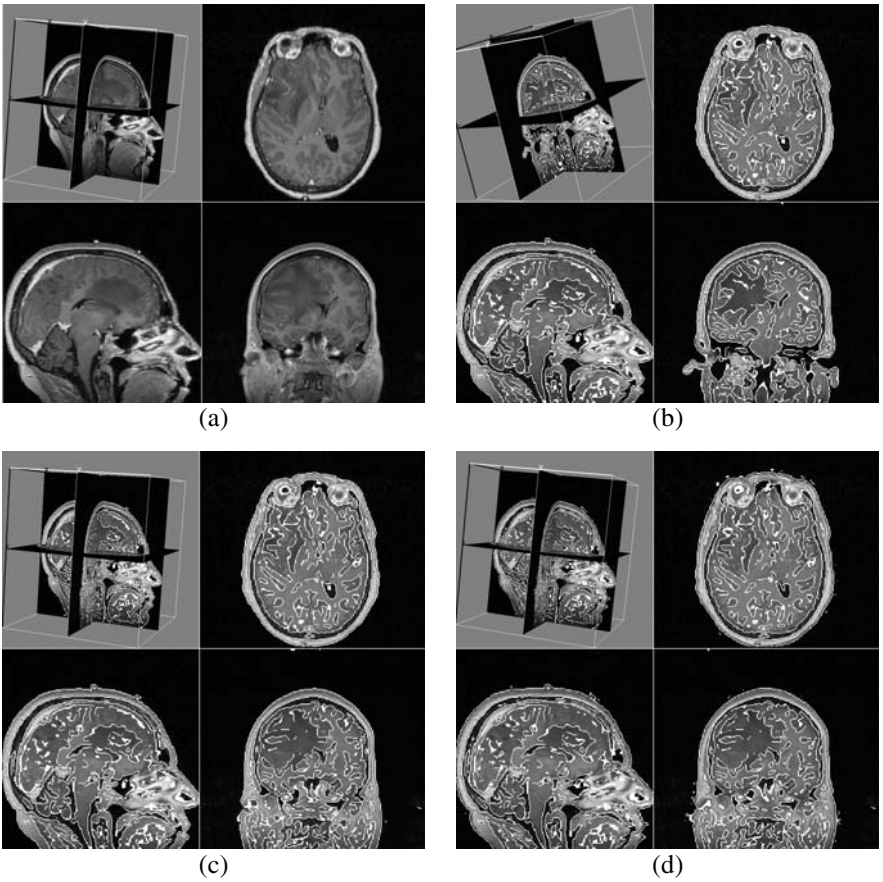


Fig. 2.13 (a) A volumetric MR brain image. The upper-left window shows three orthogonal cross-sections of the image in 3-D and the upper-right, lower-left, and lower-right windows show the three cross-sections in 2-D. (b)–(d) Edges detected by the LoG operator, the Canny method, and the intensity ratios, respectively. In all cases, the standard deviation of the Gaussian smoother was 2 pixels, and weak edges were interactively removed to keep about the same number of edges by the three methods.

$$\mathbf{v}(x,y) = \frac{\partial R(x,y)}{\partial y}\mathbf{r} + \frac{\partial G(x,y)}{\partial y}\mathbf{g} + \frac{\partial B(x,y)}{\partial y}\mathbf{b}, \tag{2.62}$$

where $R(x,y), G(x,y)$, and $B(x,y)$ are the red, green, and blue color values at pixel (x,y) and $\mathbf{r}, \mathbf{g}$, and $\mathbf{b}$ are unit vectors along red, green, and blue axes in the 3-D color space. Gradient direction at (x,y) is considered to be the direction $\theta(x,y)$ that maximizes [98]

$$F(x,y) = \left[\mathbf{u}(x,y)\cos\theta(x,y) + \mathbf{v}(x,y)\sin\theta(x,y)\right]^2. \tag{2.63}$$

The maximum $\theta(x, y)$ can be obtained by finding the derivative of $F(x, y)$ with respect to $\theta(x, y)$, setting it to zero, and solving for $\theta(x, y)$. Doing so,

$$\theta(x, y) = 0.5 \tan^{-1} \left(\frac{2\mathbf{u}(x, y) \cdot \mathbf{v}(x, y)}{\mathbf{u}(x, y) \cdot \mathbf{u}(x, y) - \mathbf{v}(x, y) \cdot \mathbf{v}(x, y)} \right) \tag{2.64}$$

is obtained, where $\cdot$ denotes inner product; that is,

$$\mathbf{u}(x, y) \cdot \mathbf{u}(x, y) = \left| \frac{\partial R(x, y)}{\partial x} \right|^2 + \left| \frac{\partial G(x, y)}{\partial x} \right|^2 + \left| \frac{\partial B(x, y)}{\partial x} \right|^2, \tag{2.65}$$

$$\mathbf{v}(x, y) \cdot \mathbf{v}(x, y) = \left| \frac{\partial R(x, y)}{\partial y} \right|^2 + \left| \frac{\partial G(x, y)}{\partial y} \right|^2 + \left| \frac{\partial B(x, y)}{\partial y} \right|^2, \tag{2.66}$$

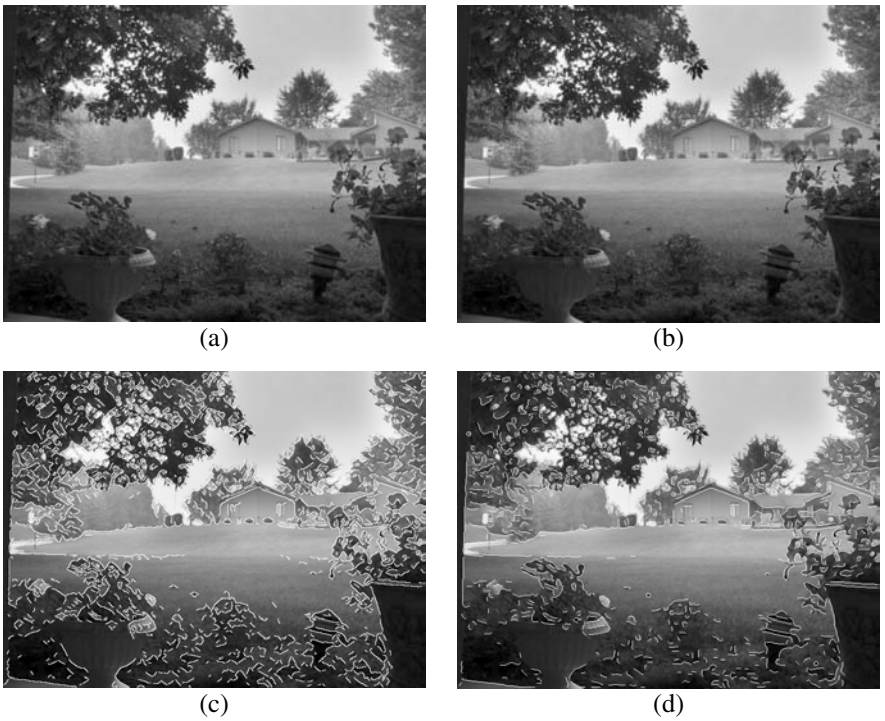


Fig. 2.14 (a) A color image of an outdoor scene. (b) The luminance component of the image. (c) Edges of the color image. (d) Edges of the luminance image. (See also color plate section.)

$$\mathbf{u}(x,y)\cdot\mathbf{v}(x,y) = \frac{\partial R(x,y)}{\partial x}\frac{\partial R(x,y)}{\partial y} + \frac{\partial G(x,y)}{\partial x}\frac{\partial G(x,y)}{\partial y} + \frac{\partial B(x,y)}{\partial x}\frac{\partial B(x,y)}{\partial y}. \tag{2.67}$$

Once the gradient direction at (x,y) is determined, the square-root of the quantity on the right hand side of equation (2.63) will show the gradient magnitude at (x,y). Knowing the gradient direction and gradient magnitude at each image pixel, the edges are detected by locating image pixels with locally maximum gradient magnitudes in the gradient direction.

An example of edges determined in this manner in a color image is shown in Fig. 2.14. The original image is shown in Fig. 2.14a and the obtained edges are shown in Fig. 2.14c. The luminance component of the color image obtained by averaging the red, green, and blue components is shown in Fig. 2.14b, and edges of the luminance image obtained by locating locally maximum gradients in the gradient direction are shown in Fig. 2.14d. We see that color edges pick up sharp changes in chroma in addition to sharp changes in luminance. Color edges separate adjacent regions with significant color differences.

2.3 SUMMARY

There is often a need to preprocess images before registration. Noise reduction and the removal of motion blur and haze in images improve feature selection. Image segmentation and edge detection also facilitate feature selection. It should be mentioned that although, customarily, image smoothing is used to reduce image noise, noise is a function of time and smoothing is a function of space. Image smoothing blurs image details and in the process it reduces the magnitude of noise. The right way to reduce noise in an image is to average the intensity at each pixel over time. This is, however, not possible if live camera images are not provided.

Several edge detection methods were covered in this chapter. The LoG operator finds edges that correspond to locally maximum as well as locally minimum gradient magnitudes. Edges corresponding to locally minimum gradient magnitudes can be removed by examining the signs of the first- and third-derivative image intensities. The Canny edge detector specifically looks for locally maximum gradients in the gradient direction, so it does not detect edges with locally minimum gradients. However, detecting only edges with locally maximum gradients may fragment a region boundary. It was shown that in order to produce closed region boundaries, it may be necessary to also detect those edges with locally minimum gradient magnitudes that are delimited by edges with locally maximum gradient magnitudes.

When lighting across an image varies, edge detection by intensity ratios is preferred over edge detection by intensity differences. Edge detection by intensity ratios responds to changes in surface reflectance and surface normal.

Among the edge detection methods discussed in this chapter, the curve fitting method is most expensive, but it enables selection of locally maximum curvature

points or inflection points as point features. The selection of such point features by traditional means requires fitting curves to the edge contours and locating the curvature maxima or inflection points.

3-D edge detection methods closely follow those of 2-D methods. Edge detection in color or multispectral images requires a different formulation than edge detection in intensity images because pixel values in these images represent d-dimensional vectors over a plane. In color images $d = 3$ and edges are considered those pixels where change in color magnitude in the color gradient direction is locally maximum.

2.4 BIBLIOGRAPHICAL REMARKS

Remote sensing images often require correction for motion blur, haze, noise, and sensor nonlinearities. Sawchuk describes a method for removing motion blur from images [338, 339], Richter describes a method for removing haze from aerial photographs and satellite images [323], and Horn and Woodham describe a method for removing geometric distortions from satellite images of the earth, using the curvature of the earth and the altitude and attitude of the satellite [193].

Geometric distortions due to lens nonlinearities can be corrected by placing a uniform grid in front of the camera and determining a transformation that maps the captured grid to the ideal grid [151, 205, 408]. Distortions in MR images occur due to the inhomogeneous magnetic field and the spatial nonlinearity of the field gradients and vary with tissue type. Distortions that are independent of the tissue type can be corrected by placing a uniform 3-D grid in the space of the scanner, obtaining an image of the grid, and determining a transformation function that maps the captured grid to the ideal grid [289, 340, 345].

A great deal of work has been done in image segmentation. A survey of thresholding methods is given by Bezdek *et al.* [32]. A survey and comparison of edge detection methods is provided by Heath *et al.* [186]. Edge detection by functional approximation has been described by Haralick [180] using bicubic surfaces. A class of iterative energy minimizing methods proposed by Kass *et al.* [220] delineates individual region boundaries. This class of methods, which has proven very effective in segmentation of medical images, has appeared in various forms in both 2-D and 3-D [11, 271, 285, 306, 337].

Many 2-D edge detectors can be easily extended to 3-D. Implementation of the LoG operator in 3-D is described by Bomans *et al.* [37]. Brejl and Sonka review various 3-D edge detection methods and describe an edge detection method that works with anisotropic data [45]. Boundary detection in 3-D using level sets has been described by Baillard *et al.* [15] and Lopéz *et al.* [248].

A relatively small number of methods exists for the detection of edges in color images. Nevatia [287] was among the first to detect edges in color images. He detected edges in red, green, and blue components by the Hueckel operator [199] and then combined the edges to obtain color edges. Cumani [80] developed a local measure of contrast in a color image and located points where the contrast in the gradient direction was locally maximum. Since color can be represented in different coor-

dinate systems, edges determined in different coordinate systems may be different. Robinson [326] compared color edges in different coordinate systems and found that edges determined in $CIELab$ color coordinates represented object boundaries better than edges determined by RGB color coordinates. This is not surprising since the evaluation was done visually, and the human visual system distinguishes more colors in the Lab color coordinate system than in the RGB coordinate system [188, 221].

3

Feature Selection

Image features are unique image properties that can be used to establish correspondence between two images. The most desired features are points, because their coordinates can be directly used to determine the parameters of a transformation function that registers the images. In some images it may not be possible to detect point features; however, lines or regions may be abundant. In such situations points are derived from the lines and regions. For example, the intersections of corresponding line pairs produce corresponding points and the centroids of corresponding regions produce corresponding points.

In this chapter, methods for extracting various image features from images are discussed. Methods for finding the correspondence between the features are discussed in the following chapter.

3.1 POINTS

Points are the most desired features. Knowing the coordinates of a number of corresponding points in two images, a transformation function can be determined to resample one image to the geometry of the other. Point features are also known as *interest points*, *point landmarks*, *corner points*, and *control points*. In this book the term *control point* will be used mostly. Control points represent centers of unique neighborhoods that contain considerable image information. Different methods can be used to measure uniqueness and information content in a neighborhood.

Neighborhoods with a large number of high-gradient pixels are highly informative, and if the high-gradient pixels form a unique pattern, such as a corner, the neighborhood becomes unique. Various methods for detecting corners in an image have been developed. Corners in an image are determined by locating points that have locally

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

maximum cornerness measures. Various cornerness measures have been proposed. Assuming $I_x(x,y)$ and $I_y(x,y)$ are gradients of image $I(x,y)$ with respect to x and y at (x,y), $\overline{I_x(x,y)}$ and $\overline{I_y(x,y)}$ are average gradients with respect to x and y in a small window centered at (x,y), and

$$\mathbf{C}(x,y) = \begin{bmatrix} \overline{I_x(x,y)}\,\overline{I_x(x,y)} & \overline{I_x(x,y)}\,\overline{I_y(x,y)} \\ \overline{I_y(x,y)}\,\overline{I_x(x,y)} & \overline{I_y(x,y)}\,\overline{I_y(x,y)} \end{bmatrix}, \tag{3.1}$$

then $det(\mathbf{C})$ can be used as a cornerness measure. $\mathbf{C}$ is called the inertia matrix. This cornerness measure has been used by Rohr [327], while the normalized version of it, $det(\mathbf{C})/tr(\mathbf{C})$, has been used by Förstner [125] to detect corners in an image. Here, $det()$ and $tr()$ denote the determinant and the trace of a matrix, respectively.

The eigenvalues of matrix $\mathbf{C}$ are indicators of the strength of the gradients in directions normal to each other in a window. If both eigenvalues are large, it can be concluded that a strong corner exists within the window. It is sufficient to examine the value of the smaller of the two eigenvalues to locate the corners. Tomasi and Kanade [387] find corners in an image in this manner. An algorithm based on this idea is given below. Although detection of image edges is not required, in order to speed up the process, image edges are used to limit the search space.

Algorithm 3.1: In this algorithm, INPUT is a list used to save all detected corners, OUTPUT is a list used to save the strong and widely dispersed corners, n shows the number of required corners, m shows the number of detected corners, L is an image where the value at an edge point represents the smaller of the two eigenvalues of matrix $\mathbf{C}$ at the point, and σ is the standard deviation of the Gaussian smoother used to detect the edges.

1: Find the image edges and compute gradients of the image with respect to x and y. Also, create empty INPUT and OUTPUT lists, and let $m = 0$.

2: At each edge point:
   2.1: Compute the inertia matrix $\mathbf{C}$ using image gradients in the circular area of radius 3σ. If a portion of the window falls outside the image, use only the portion that falls inside the image.

   2.2: Determine the eigenvalues of the matrix. Suppose they are λ_1 and λ_2, and $\lambda_1 > \lambda_2$. Find the eigenvalues for windows centered at all image edges and save their λ_2s in image L at the corresponding edge positions.

3: Find edge points that have locally maximum λ_2s by examining 3×3 neighborhoods in image L. Consider such edge points as the corners and save them in INPUT.

4: Sort INPUT according to λ_2 from the largest to the smallest.

5: Starting from the top of INPUT, move corners from INPUT to OUTPUT one at a time. After moving a corner, increment

m by 1 and remove all corners in INPUT that are within distance 7σ of it. Repeat the process until either all n required corners are found or no more corners remain in INPUT.

6: Return the OUTPUT list and m; m is the number of detected corners.

The window size used to compute the inertia matrix at an edge point is set to 3σ in step 2.1. This window size is arbitrary and the user may decrease it to make the measure more local or increase it to make the measure more global. The window size is set proportional to the standard deviation of the Gaussian smoother that is used to detect the edges and the image gradients. Therefore, a larger window is used when a larger smoothing filter is selected in order to detect the edges and provide sufficient image information in the computation of the inertia matrix. Once a corner is selected in step 5, the algorithm ensures that corners within distance 7σ of it are not selected. The number 7 is again arbitrary and can be changed to obtain desired spacing between the corners. Experiments on various images show that 3σ for neighborhood size and 7σ for spacing between corners produce strong and well dispersed corners suitable for image registration.

The eigenvalues of matrix $\mathbf{C}$ are determined as follows. If λ is an eigenvalue of $\mathbf{C}$, since it should satisfy

$$det(\mathbf{C} - \lambda\mathbf{I}) = 0, \tag{3.2}$$

where $\mathbf{I}$ is an identity matrix, we have

$$\begin{vmatrix} \overline{I_x}^2 - \lambda & \overline{I_x}\,\overline{I_y} \\ \overline{I_y}\,\overline{I_x} & \overline{I_y}^2 - \lambda \end{vmatrix} = 0, \tag{3.3}$$

which reduces to

$$\lambda^2 + B\lambda + C = 0, \tag{3.4}$$

having solutions

$$\lambda_1, \lambda_2 = (-B \pm \sqrt{B^2 - 4C})/2, \tag{3.5}$$

where

$$B = -(\overline{I_x}^2 + \overline{I_y}^2) \tag{3.6}$$

and

$$C = \overline{I_x}^2\,\overline{I_y}^2 - (\overline{I_x}\,\overline{I_y})^2. \tag{3.7}$$

Assuming λ_1 and λ_2 are the two eigenvalues of $\mathbf{C}$ and $k = \lambda_1/\lambda_2$, Harris and Stephens [184] define the cornerness measure by

$$R = \frac{k}{(k+1)^2}\left[det(\mathbf{C}) - tr(\mathbf{C})^2\right]. \tag{3.8}$$

The subtracted term in equation (3.8) makes sure that the corner detector does not respond to stair-casing of diagonal step edges. Harris and Stephens also suggested the use of Gaussian weighted averaging of the derivatives inside small windows when computing $\overline{I_x}$ and $\overline{I_y}$ rather than simple averaging.

These corner detectors compute the cornerness measure using the image gradients (first-derivative image intensities). The cornerness measure can be defined in terms of the second-derivative image intensities or a combination of the first- and second-derivative image intensities. Beaudet [26] used

$$I_{xx}I_{yy} - I_{xy}^2, \tag{3.9}$$

while Kitchen and Rosenfeld [225] used

$$\frac{I_{xx}I_y^2 - 2I_{xy}I_xI_y + I_{yy}I_x^2}{I_x^2 + I_y^2}, \tag{3.10}$$

as a cornerness measure to detect control points in images, where I_{xx} and I_{yy} are image second derivatives with respect to x and y, respectively, and I_{xy} is the derivative of image intensities with respect to both x and y. The first and second derivative image intensities with respect to x and y are obtained by convolving the image with the first and second derivative 1-D Gaussians in x and y directions, respectively. The Gaussian process will smooth the image while determining its derivatives. Image smoothing is required in order to reduce the effect of image noise on detected corners. Note that the absence of *overline* in the cornerness measures by Beaudet and Kitchen and Rosenfeld implies the use of the gradient information at a single pixel as opposed to a collection of pixels within a circular window for methods that use $\mathbf{C}$. However, since smoothing is performed in the computation of the gradients, in the method of Beaudet and the method of Kitchen and Rosenfeld, information about pixels in a small neighborhood of the pixel under consideration is used to calculate the cornerness measure.

Examples of corners detected by Algorithms 3.1 are shown in Fig. 3.1. The corners detected at two resolutions ($\sigma = 1.5$ and 2 pixels) are shown. Many corners have appeared in both resolutions although their positions are slightly shifted with respect to each other. There are also corners that have been detected in one resolution but not in the other. As image resolution is varied, some locally maximum cornerness measures cease to remain locally maximum, while points with locally maximum cornerness measures newly emerge. This can actually be used to filter out less stable corners and retain only the stable ones. Stable corners show neighborhoods that remain locally unique under a wide range of resolutions and will result in more

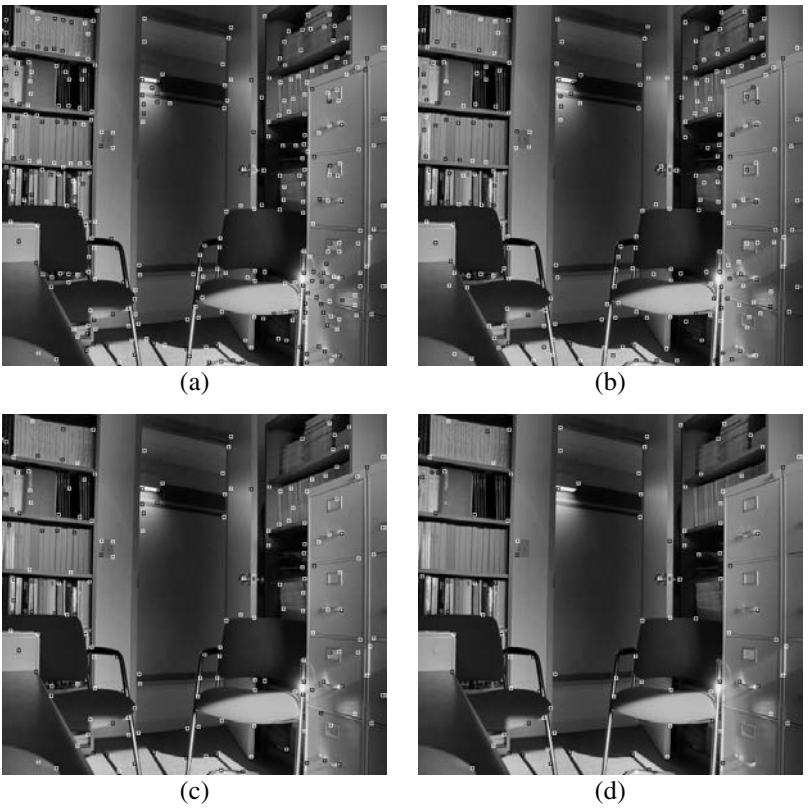


Fig. 3.1 Corners detected by Algorithm 3.1 using (a) $\sigma = 1.5$ pixels and (b) $\sigma = 2$ pixels. (c) Corners that are common in both (a) and (b). (d) Stable corners that persist from $\sigma_1 = 1.5$ pixels to $\sigma_2 = 3$ pixels. Since the position of a corner may shift as the resolution is changed, in this figure, corners whose positions did not shift by more than $\Delta\sigma = \sigma_2 - \sigma_1$ pixels were considered stable and selected.

reliable correspondences than corners that are not as unique. Figures 3.1c and 3.1d show examples of stable corners.

If the objective is to locate unique and highly informative image neighborhoods, uniqueness and information content may be explicitly used to locate the corners. The following algorithm first determines the image edges and then locates those edges that are locally unique and are centered at image windows with considerable image detail.

> **Algorithm 3.2:** In this algorithm, n is the number of required corners, m is the number of available corners, and D is a measure controlling the spacing between the obtained corners. INPUT is a list containing all edges that can potentially represent corners and OUTPUT is a list that contains the set of strong and widely dispersed corners.

1: Determine the image edges.

2: From among the detected edges, select those that are centered at highly informative circular windows of radius R and save in INPUT.

3: From among the edges in INPUT, keep only those that are centered at unique circular windows of radius R and are also widely dispersed as follows.

3.1: Order the edges in INPUT according to their uniqueness from the highest to the lowest.

3.2: Initially, clear OUTPUT to an empty list and let $m = 0$.

3.3: If $m = n$ or if the INPUT is empty, stop; otherwise, continue.

3.4: Remove the edge from the top of INPUT, append it to the OUTPUT, and increment m by 1.

3.5: Multiply the uniqueness of edges in INPUT by $H = 1 - exp(-d_i^2/D^2)$ and sort INPUT again, where d_i is the distance of the ith edge in INPUT to the edge just moved from INPUT to OUTPUT, and D is a parameter controlling the spacing of the selected control points. Go back to step 3.3.

Step 2 keeps only those edges that are centered at highly informative windows. A circular window is considered highly informative if it has a rather high entropy. Entropy is computed using the histogram of the window and is computed from

$$E = -\sum_{i=0}^{255} p_i \log_2(p_i), \tag{3.11}$$

where p_i is the probability that if a pixel is randomly selected in the window, it will have intensity i and is estimated from $h(i)/T$. Here, $h(i)$ is the number of pixels with intensity i and T is the total number of pixels in the window. Note that since the histogram of a circular window at (x, y) is independent of the orientation of the image, the same entropy is obtained for point (x, y) independent of the orientation of the image. Control points centered at highly informative neighborhoods are more likely to produce correct correspondences than control points that are located in rather uniform neighborhoods. In step 2, one may choose the most informative half, or one-third, of the edges as the candidate control points.

In addition to being informative, step 3 ensures that the selected edges are centered at unique windows. Locally unique control points will be more likely to produce correct correspondences than control points that are not locally unique. An edge point is considered locally unique if auto-correlation at the point becomes locally minimum when using circular windows of radius R in the computations.

Uniqueness at a pixel can be measured using the dissimilarity of the window centered at it with windows centered at pixels adjacent to it using the original image intensities. The circular window centered at a pixel is highly unique if it is not very similar to windows centered at the adjacent pixels. The correlation coefficient can be used as a means to determine the similarity between two windows. Denoting the window centered at (x, y) by $\mathbf{W}(x, y)$, the uniqueness measure $U(x, y)$ is defined by

$$U(x, y) = 1 - MAX\left\{CC[\mathbf{W}(x, y), \mathbf{W}(x + x', y + y')]\right\}, \tag{3.12}$$

where $x' = -1, 0, 1$; $y' = -1, 0, 1$; and x' and y' are not both 0. The correlation coefficient of two windows is denoted by CC. To determine the uniqueness of the window centered at (x, y), its correlation is determined with windows centered at the eight adjacent pixels and the difference of the largest similarity and the maximum possible similarity of 1 is determined. If at least one of the windows adjacent to (x, y) is similar to the window at (x, y) then the window at (x, y) will not be very unique and, therefore, U will be rather small. However, if correlations between $\mathbf{W}(x, y)$ and the windows adjacent to it are all rather small, then $\mathbf{W}(x, y)$ will be unique.

The cross-correlation coefficient between windows $\mathbf{W}_1$ and $\mathbf{W}_2$ is computed from

$$CC[\mathbf{W}_1, \mathbf{W}_2] = \frac{\mathbf{W}_1 \cdot \mathbf{W}_2}{||\mathbf{W}_1||\ ||\mathbf{W}_2||}, \tag{3.13}$$

where $\cdot$ denotes vector inner product and $||\mathbf{W}||$ denotes the magnitude of vector $\mathbf{W}$. If $\mathbf{W}$ is an image window, pixel intensities in the window form the elements of the vector. In order for the obtained correlation to fall in range $[-1, 1]$, the windows are normalized to have a mean of 0.

Parameter R in Algorithm 3.2 controls the neighborhood size over which the correlation, and thus the uniqueness measure, is computed. A smaller R will speed up the computations but it may not use sufficient image information to produce highly reliable correspondences. On the other hand, a larger R will detect control points that are unique and highly informative, thus finding more reliable correspondences, but it will slow down the feature selection process. Experiments show that values of R from 3σ to 5σ pixels are suitable compromises between speed and reliability.

In step 3.5, the uniqueness of edges remaining in INPUT are multiplied by H, a measure that is inversely proportional to the distance of the edge just moved from INPUT to OUTPUT to the edges remaining in INPUT. Since H approaches 0 exponentially, it will affect only a small number of edges near the edge just selected. In practice, it is necessary to revise the uniqueness of a point in INPUT and reposition it in INPUT, if necessary, when the distance of the point just selected to it is less than $5D$ pixels. Otherwise, the uniqueness and the position of the point are not changed. Step 3.5 ensures that edges that are very close to the edges already selected as control points are not selected again. Note that INPUT is sorted every time a feature point is moved from INPUT to OUTPUT, after multiplying the uniqueness measures by H. Alternatively, after selecting an edge point from INPUT, all edges within a certain distance of it, such as $5D$, can be removed from INPUT without sorting the list. This

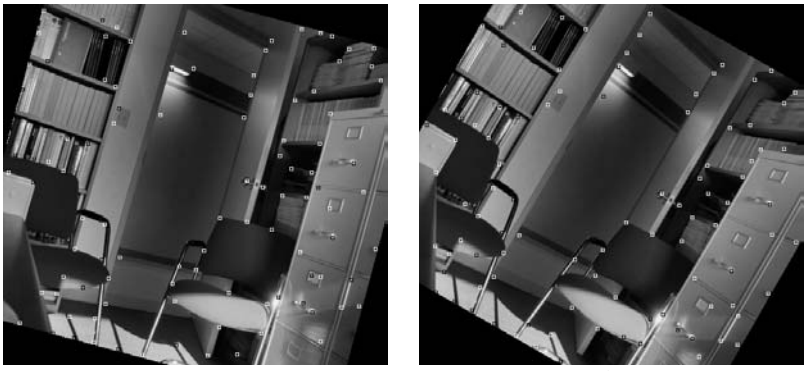
will speed up the feature selection process at the cost of selecting less optimal feature points.

Theoretically, the control points selected by Algorithms 3.1 and 3.2 are independent of the orientation of a given image. This is because the edge detector is rotationally invariant, the eigenvalues of matrix $\mathbf{C}$ are independent of the orientation of the image when circular windows are used, and the entropy of a circular window does not depend on the orientation of the window. Some differences may be observed in a 3×3 window when searching for the locally maximum cornerness measures due to the digital nature of images. Therefore, it is anticipated that some of the control points detected in two images that have rotational differences be different. Figures 3.2a and 3.2b show corners detected in the image of Fig. 3.1 after rotation by 15 and 30 degrees, respectively. As can be observed, the majority of corners in Figs 3.1d, 3.2a, and 3.2b are the same. There are corners, though, that appear in only one of the images.

Control points in 3-D are determined the same way they are determined in 2-D. In 3-D the inertia matrix is defined by

$$\mathbf{C} = \begin{bmatrix} \overline{I_x}^2 & \overline{I_x}\,\overline{I_y} & \overline{I_x}\,\overline{I_z} \\ \overline{I_y}\,\overline{I_x} & \overline{I_y}^2 & \overline{I_y}\,\overline{I_z} \\ \overline{I_z}\,\overline{I_x} & \overline{I_z}\,\overline{I_y} & \overline{I_z}^2 \end{bmatrix}. \tag{3.14}$$

Matrix $\mathbf{C}$ in 3-D will have three eigenvalues. Again, the smallest eigenvalue will be used to measure cornerness at an edge point. To find edges with locally maximum cornerness measures, a search is made in $3 \times 3 \times 3$ windows in step 3 of Algorithm 3.1. Everything else in the algorithm remains the same. The cornerness measure of



(a) (b)

Fig. 3.2 Stable corners detected by Algorithm 3.1 when rotating the image in Fig. 3.1 by (a) 15 degrees and (b) 30 degrees. The corners in each image show those persisting from $\sigma = 1.5$ to $\sigma = 3$ pixels.

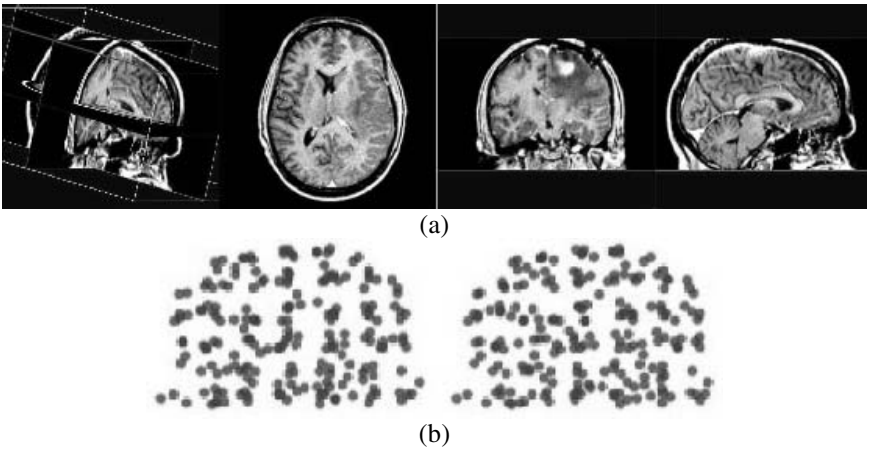


Fig. 3.3 (a) An MR brain image of size $256 \times 256 \times 128$ voxels. The leftmost plate shows three orthogonal cross-sections of the image in 3-D and the three plates to its right show the individual cross-sections. (b) The most informative, unique, and widely dispersed 200 control points detected by Algorithm 3.2 using $\sigma = 2.5$ pixels are shown in stereo.

Beaudet in 3-D becomes the determinant of the Hessian matrix $\mathbf{H}_g$ [185],

$$\mathbf{H}_g = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}, \tag{3.15}$$

and the cornerness measure of Kitchen and Rosenfeld becomes [185]

$$\frac{I_x^2(I_{yy}+I_{zz})+I_y^2(I_{xx}+I_{zz})+I_z^2(I_{xx}+I_{yy})-2(I_xI_yI_{xy}+I_xI_zI_{xz}+I_yI_zI_{yz})}{I_x^2+I_y^2+I_z^2}. \tag{3.16}$$

Algorithm 3.2 in 3-D primarily remains the same; however, a 3-D edge detector rather than a 2-D edge detector is used, spherical windows rather than circular windows are used to compute the entropies and correlation coefficients, and 26 neighbors of an edge point rather than 8 neighbors of an edge point are examined to determine whether or not the edge point has locally maximum uniqueness measure. An example of 3-D corners detected by Algorithm 3.2 is shown in Fig. 3.3. The number of detected corners is limited to 200, shown in stereo for 3-D viewing.

3.2 LINES

Images of man-made scenes contain abundant line features, which can be used in image registration. Various methods for detecting lines in images have been developed.

When the images are not noisy and contain a small number of lines, Hough transform may be used to detect them. If an image or a region in an image is known to contain a single line and the points on the line are known, the line can be determined by the least-squares method. Often, a large number of lines are present in an image and there is a need to find them. In the following section, the concepts of Hough transform and least-squares fitting are described and their uses in line detection and line fitting are given. Then, a general algorithm to detect a large number of lines in an image is described.

3.2.1 Line detection using the Hough transform

Hough transform is a patented methodology for detecting analytic lines and curves in images [196]. The idea, however, is general and can be used to detect arbitrary digital shapes in images [17]. A line in the xy space is defined by

$$y = mx + b. \tag{3.17}$$

Parameter m is the slope and parameter b is the y-intercept of the line. Note that a line in the xy space as defined by equation (3.17) corresponds to point (m, b) in the mb space. Conversely, line (3.17) in the mb space corresponds to point (x, y) in the xy space. If points $\{(x_i, y_i) : i = 1, \ldots, n\}$ on a line in the xy space are known, the line obtained from each such point should pass through the same point in the mb space, which is the point defining the line in the xy space. Therefore, to determine lines from points in an image, all that needs to be done is to initialize all entries of the mb space to 0 and increment an entry by 1 when the line representing a point in the xy space passes over it, and then find the entry in the mb space that has the highest count. If more than one line is to be detected, entries with locally peak counts in the mb space are located and their coordinates are used as the slopes and y-intercepts of the lines. To avoid detection of noisy lines, only locally peak counts that are larger than a prespecified threshold value may be used.

The process of line detection by Hough transform is depicted in Fig. 3.4. Figure 3.4a shows an image containing six points, five of which belong to a line and one that is noise. The image contains r rows and c columns. The parameter space is quantized into r rows and $C + 1$ columns, the columns representing the slopes of the lines from 0 to 1 with spacing of $1/C$. C is provided by the user. If a finer angular quantization is required, a wider array should be used. Initially, all entries of the array are set to 0 and entries falling on the line obtained from each point in the xy space are incremented by 1. For points in Fig. 3.4a, the lines in Fig. 3.4b are obtained. The entry with the maximum count shows the m and b parameters of the detected line.

The accumulator array in Fig. 3.4b can detect lines with slopes between 0 and 1. A similar accumulator array is needed to detect lines with slopes between -1 and 0. To determine lines with slopes greater than 1 and slopes less than -1, the image should be rotated by 90 degrees and processed again. This is done because slopes of nearly vertical lines are very large, requiring very large accumulator arrays. Alternatively,

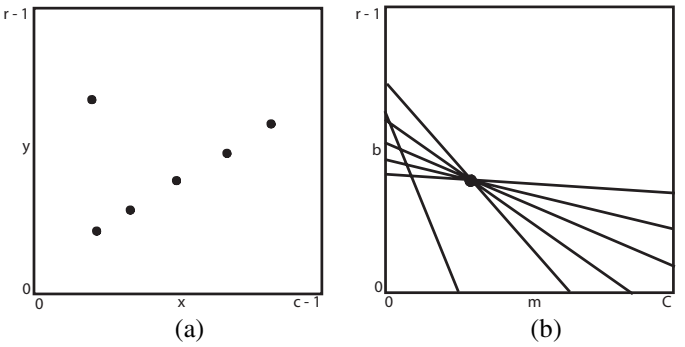


Fig. 3.4 (a) Points in an image. (b) Corresponding lines drawn in the parameter space (accumulator array). Points on a line in the image space produce lines that intersect at the same point in the parameter space, while random points in the image space produce random lines in the parameter space.

lines in the image space my be described in polar form;

$$\rho = (x - x_c)\cos(\theta) + (y - y_c)\sin(\theta), \tag{3.18}$$

where ρ is the distance of the line to the image center, (x_c, y_c) are the coordinates of the image center, and θ is the angle of the normal to the line with the x axis. Since ρ is limited to $\sqrt{r^2 + c^2}/2$ and θ is limited to 360 degrees, a rather small accumulator array is sufficient to find all lines in an image [101]. Note that if the polar equation of lines is used, for each point in an image, a sinusoidal curve rather than a line should be drawn in the accumulator array. Again, array entries with locally peak counts should be identified and used to detect the lines.

To detect lines in an image, the image edges are first determined and weak edges are removed. This will reduce the number of edge points to be processed and will also avoid the detection of weak and noisy lines. Hough transform by itself cannot tell which edge points contribute to a line, it simply gives the equation of the line. Further processing is needed to determine the edge points that contribute to a line. Hough transform is not sensitive to outliers if the number of outliers is much smaller than the number of points on the lines to be detected (see Fig. 3.4b). Hough transform is sensitive to noise though. If the majority of the given points are not noisy, noisy points and outliers will not affect the outcome of the Hough transform.

3.2.2 Least-squares line fitting

Knowing the coordinates of a number of points on a line, the coefficients of the line can be determined by the least-squares method. Least-squares is a mathematical tool for determining the unknown parameters of an analytic function from which a set of noisy samples are available [324]. In our case, the underlying function is a line and the parameters to be estimated are the line coefficients. The polar equation of a line

is given by (3.18), or equivalently by

$$Ax + By + C = 0, \tag{3.19}$$

where $A = \cos(\theta)$, $B = \sin(\theta)$, and $C = -(x_c \cos\theta + y_c \sin\theta + \rho)$. We will use line equation (3.19), keeping in mind that $A^2 + B^2 = 1$.

Assuming the given points are $\{(x_i, y_i) : i = 1, \ldots, n\}$ and the line to be estimated is defined by (3.19), the distance of point (x_i, y_i) to the line is given by

$$d_i = |Ax_i + By_i + C| \tag{3.20}$$

with the condition that $A^2 + B^2 = 1$. We would like to find the line where the sum of squared distances from the points to the line, that is

$$E_2 = \sum_{i=1}^{n} (Ax_i + By_i + C)^2, \tag{3.21}$$

is minimum. Figure 3.5 depicts the least-squares line fitting process. To find the A, B, and C that minimize E_2, we determine the partial derivatives of E_2 with respect to A, B, and C, set them to zero, and solve the equations for A, B, and C. This will result in;

$$A = \pm \frac{d\bar{x} - b\bar{y}}{\sqrt{(a\bar{y} - b\bar{x})^2 + (d\bar{x} - b\bar{y})^2}}, \tag{3.22}$$

$$B = A \frac{a\bar{y} - b\bar{x}}{d\bar{x} - b\bar{y}}, \tag{3.23}$$

$$C = -(A\bar{x} + B\bar{y}), \tag{3.24}$$

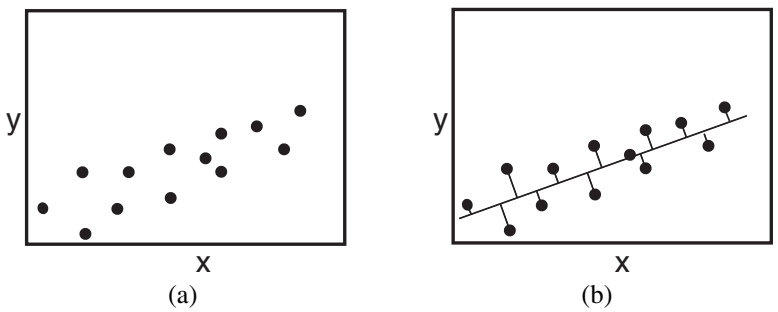


Fig. 3.5 (a) A set of noisy points and (b) the line fitting to it by the least-squares method. The process minimizes the sum of squared distances of the points to the line.

where $a = \sum_{i=1}^{n} x_i^2$, $b = \sum_{i=1}^{n} x_i y_i$, $d = \sum_{i=1}^{n} y_i^2$, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$. Consequently,

$$\theta = \tan^{-1}(B/A), \tag{3.25}$$

$$\rho = -(C + x_c \cos\theta + y_c \sin\theta). \tag{3.26}$$

Two solutions are obtained, one minimizing the sum of squared distances and one maximizing the sum of squared distances. Among the two solutions, the one producing the smaller E_2 is selected to represent the line.

Least-squares line fitting requires the coordinates of points on a line. If an edge image is available, one may trace the edge contours and fit lines to them as they are

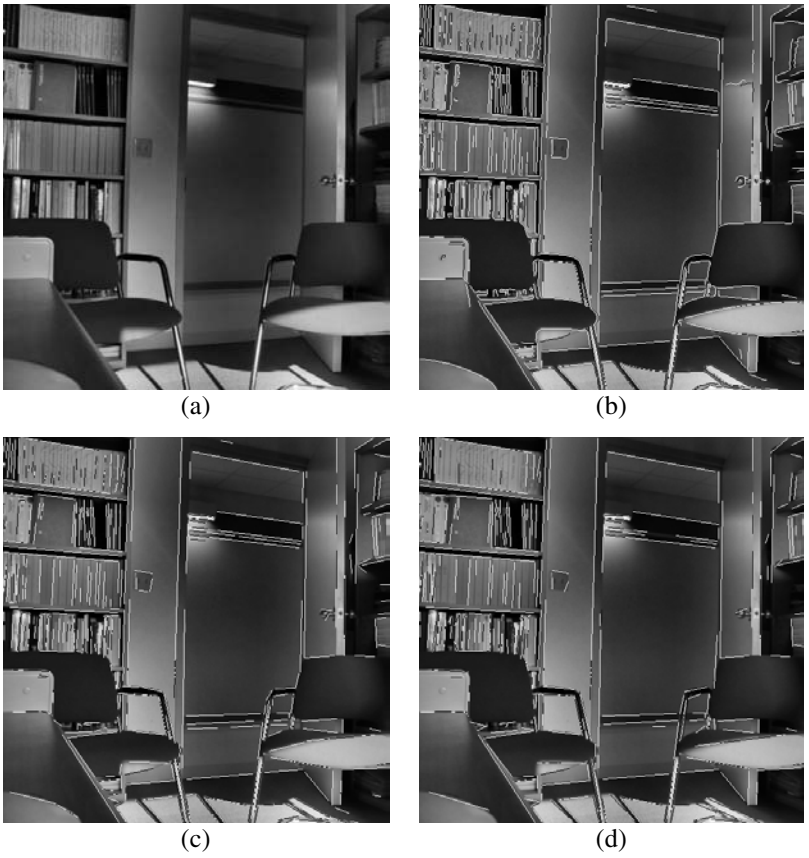


Fig. 3.6 (a) An office image. (b) Canny edges of the image obtained with a Gaussian smoother of standard deviation 1.5 pixels. The weak edges have been removed. (c), (d) Lines in the edge image detected by the least-squares line-fitting method with tolerances of 2 and 4 pixels, respectively. Very short line segments have been discarded.

traced. Once the error in line fitting reaches a desired tolerance, the contour segment traced so far is replaced with the line segment and the process is repeated until all edge contours are processed. This process, in effect, replaces edge contours with line segments. An example of line fitting in this manner is shown in Fig. 3.6. The Canny edges of Fig. 3.6a are shown in Fig. 3.6b. Fitting lines to the edge contours such that the maximum distance between the contours and the lines is less than or equal to 2 pixels the lines shown in Fig. 3.6c are obtained. Increasing the tolerance to 4 pixels, the lines shown in Fig. 3.6d are obtained. As the error tolerance is increased, the lines become longer but they also shift more from their true positions.

3.2.3 Line detection using image gradients

If the lines to be detected separate regions of different properties from each other, the lines can be found from the high-gradient pixels. Image gradients have magnitude and direction. An algorithm that detects lines in an image by partitioning the image into regions of different gradient directions and fitting a line to each region by the weighted least-squares method is described below.

Algorithm 3.3: n_t is the minimum region size used in line fitting and α is the directional accuracy of the lines being determined.

1: Determine the gradient magnitudes and directions of pixels in the image.

2: Partition the image into regions with gradient directions in ranges $[-\alpha, \alpha]$, $[\alpha, 3\alpha], \ldots, [\pi/2 - \alpha, \pi/2 + \alpha], \ldots$

3: Remove regions containing fewer than n_t pixels and fit a line to each remaining region by the weighted least-squares method.

4: For each line, draw the portion of the line falling inside the region from which it was obtained.

The smaller the α, the more accurate an obtained line direction will be, but because a smaller α will result in smaller regions, this will result in shorter line segments. Since man-made scenes often contain lines that are nearly vertical and horizontal, the angular quantization should be made such that the obtained regions include gradient directions $0, \pi/2, \pi, 3\pi/2$ and directions close to them. For instance, when $\alpha = \pi/16$, the ranges should be $[-\pi/16, \pi/16]$, $[\pi/16, 3\pi/16]$, $[3\pi/16, 5\pi/16]$, etc.

Detailed image regions or regions containing corners become fragmented into small regions because of the large variation in gradient directions. They are removed in step 3 of the algorithm before lines are fitted to them. When fitting a line to pixels in a region, we would like the pixels with larger gradient magnitudes to have larger roles in the obtained line than pixels with smaller gradient magnitudes. For that reason, weighted least-squares is used. The error measure to be minimized in

weighted least-squares is

$$W_2 = \sum_{i=1}^{n} w_i (Ax_i + By_i + C)^2 \tag{3.27}$$

where w_i is the gradient magnitude of pixel (x_i, y_i) in the region under consideration. The parameters of the line minimizing W_2 are again obtained from (3.22), (3.23),

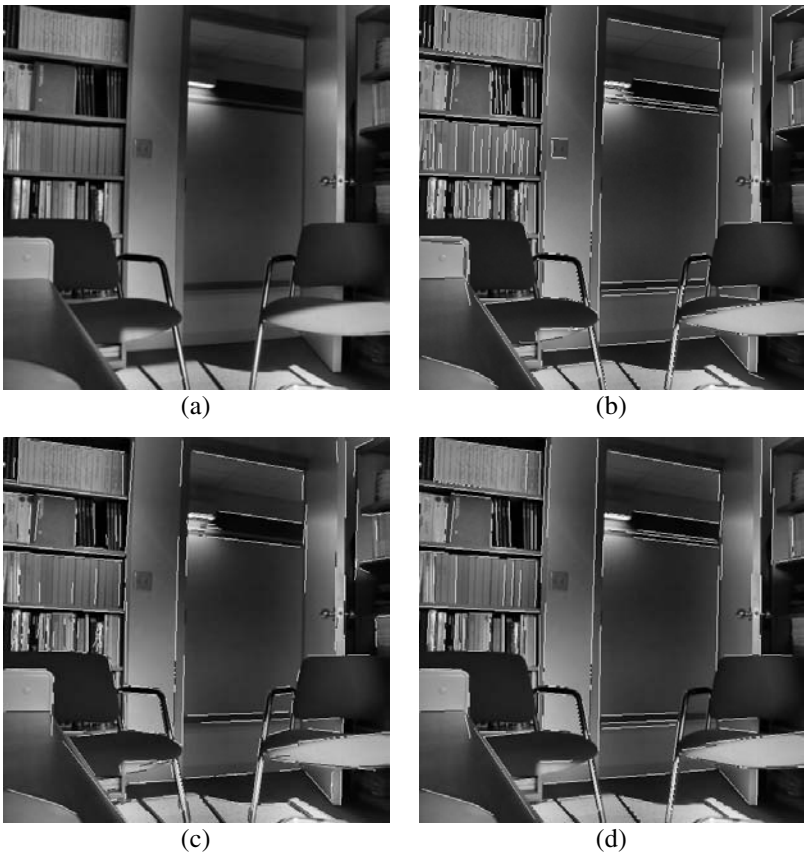


Fig. 3.7 (a) An office image. (b) Lines in the image determined by Algorithm 3.3 using a Gaussian smoother of standard deviation 1 pixel, grouping connected pixels with gradient magnitudes within $\pi/8$ of each other into regions starting from $-\pi/16$, removing pixels in a region with gradient magnitudes smaller than 3 and discarding regions containing fewer than 50 pixels. (c) Lines detected when changing the standard deviation of the Gaussian smoother to 2 pixels but keeping other parameters the same as those in (b). (d) Lines obtained when discarding regions containing fewer than 100 pixels, but keeping other parameters the same as those in (b).

and (3.24), but using $a = \sum_{i=1}^{n} w_i x_i^2$, $b = \sum_{i=1}^{n} w_i x_i y_i$, $d = \sum_{i=1}^{n} w_i y_i^2$, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} w_i x_i$, and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} w_i y_i$.

Examples of line detection by Algorithm 3.3 are shown in Fig. 3.7. Image 3.7a was smoothed with a Gaussian filter of standard deviation 1 pixel to reduce noise without significantly smoothing image details. Gradient magnitudes of the image were determined and the image was partitioned into regions with gradient magnitudes within $\pi/8$ of each other starting from $-\pi/16$. Pixels with gradient magnitudes smaller than 3 were then removed and, if more than 50 pixels remained in a region, a line was fitted to the region by the weighted least-squares method. This produced the lines shown in Fig. 3.7b. Smoothing the image with a Gaussian of standard deviation 2 pixels produced the lines shown in Fig. 3.7c. Smoothing suppresses the weaker lines. Using a Gaussian smoother of standard deviation 1 pixel but discarding regions containing fewer than 100 pixels, the lines shown in Fig. 3.7d are obtained. Compared to Fig. 3.7b, we see that longer lines are detected, suppressing the shorter line segments.

3.3 REGIONS

Remote sensing and medical images often contain clearly defined regions. For instance, lakes in satellite or aerial images appear homogeneous and have properties that are often quite different from those of surrounding land and thus can be easily extracted through a thresholding process. Bones in CT images represent rather homogeneous regions with intensities that are different from surrounding soft tissues and again can be isolated through intensity thresholding. Centroids of such regions are control points that are resistant to noise because they are obtained from the average of pixels on region boundaries and the averaging process reduces the effect of noise on the computed centroids. Elaborate image segmentation methods may be needed to extract nonhomogenous regions in images, some of which were discussed in section 2.2.

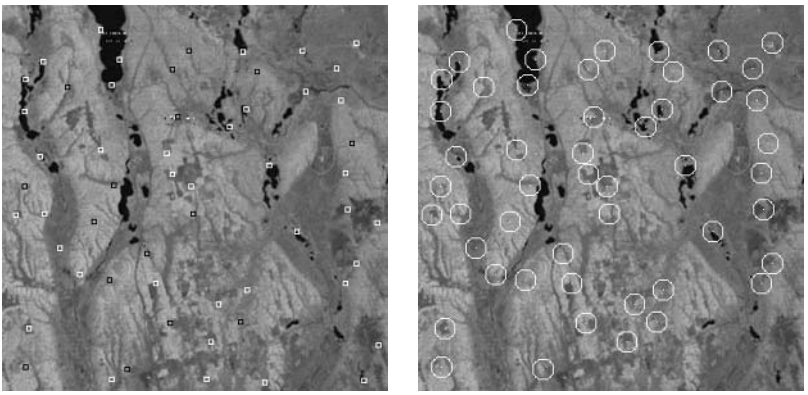
For the centroids of corresponding regions to correspond to each other, it is required that corresponding regions represent the same scene parts and geometric difference between images within the local neighborhoods representing the regions be negligible. The segmentation method that is used to find regions in two images should ensure that obtained regions are very similar. This may require changing the parameters of the segmentation, region by region. In a segmentation method described in Goshtasby *et al.* [164], first, regions in the reference and sensed images are extracted by intensity thresholding and correspondence is established between the regions using their shapes and relative distances. Then, for each region in the reference image, the corresponding region in the sensed image is iteratively revised by gradually changing the threshold value in such a way to increase the similarly between the region shapes. This process is repeated until the peak similarity is reached for corresponding regions. Finally, corresponding region centroids are determined with subpixel accuracy and used as corresponding points to determine the registration parameters.

3.4 TEMPLATES

Templates are unique image neighborhoods. For instance, neighborhoods contained in rectangular or circular windows can be used as templates. If two image windows correspond to each other, their centers will correspond to each other. Templates are abundant and are not limited to a particular type of image; therefore, they can be used to register all types of images.

When templates are selected from the reference image, template matching is carried out to find windows in the sensed image that contain similar patterns. Centers of corresponding templates and windows are then used as corresponding points to register the images. The template-matching process that finds this correspondence (see Chapter 4) produces more reliable matches if the selected templates are locally unique and highly informative. Algorithm 3.2, which was used to select highly informative and locally unique control points can, therefore, be used to select highly informative and locally unique templates in an image. Figure 3.8 shows an example. Figure 3.8a shows the detected control points and Fig. 3.8b shows the corresponding templates. Only templates that are fully included in the image are selected.

Circular templates are preferred over rectangular ones because two templates with the same center will contain the same image areas independent of the rotational difference between the images. Circular templates can be used in conjunction with invariant moments to produce rotationally invariant features that can register images with rotational differences [145].



(a) (b)

Fig. 3.8 (a) Control points and (b) templates selected in a satellite image.

3.5 SUMMARY

Algorithms for selecting points, lines, regions, and templates in images were described. Point features are ideal for image registration because their coordinates can be directly used to determine the parameters of the transformation function that registers the images. Point features are selected using corner detectors.

In man-made scenes, lines are often present and may be used as features to register images. Intersections of lines define points, which can be determined with subpixel accuracy and used to register the images. Line detection by Hough transform, least-squares, and weighted least-squares was discussed.

Closed-boundary regions often appear in satellite and aerial images and may be used as features to register images. Centroids of regions can be determined with subpixel accuracy and used as control points. Templates represent locally unique and highly informative image neighborhoods. Centers of closed-boundary regions and circular templates are rotationally invariant and may be used to register images with rotational differences.

Feature selection is the first and perhaps the most important step in image registration. The ability to extract a large number of features that are invariant to image orientation and are not sensitive to image noise and image resolution greatly reduces the burden on the correspondence step that follows feature selection. Algorithms 3.1 and 3.2 use circular windows, image gradients, eigenvalues of the inertia matrix, entropy, and autocorrelation, all of which are independent of the orientation of an image. Some minor dependency on image orientation may exist due to the digital nature of images. Keeping only the stable corners that persist over a wide range of image resolutions makes the process insensitive to image resolution. It also makes the process resistant to noise because the smoothing process that is employed to track a corner in a wide range of resolutions reduces the effect of image noise on selected corners.

3.6 BIBLIOGRAPHICAL REMARKS

Early methods registered images directly without finding their features. One image was shifted over another and, at each shift position, the similarity between the two was determined. The shift position producing the highest similarity was chosen to register the images [8, 23, 315].

To reduce the computation time subimages (templates) were introduced in image registration and, to speed up the process even further, multi-stage [159, 394] and coarse-to-fine [335] methods were developed. Circular templates have also been used to achieve rotational invariance in feature selection [145].

The importance of point features in image registration became evident when trying to register images with rotational differences. Rotationally invariant point features defined in terms of image derivatives are described by Beaudet [26], and Kitchen and Rosenfeld [225]. Image corners are used as point features in image registration by Rohr [328, 329, 330], Heitger *et al.* [187], Reisenfeld *et al.* [322], Trajkovic and

Hedley [388], Frantz *et al.* [131], Harris and Stephens [184], Zheng *et al.* [429], Rangarajan *et al.* [321], Wang and Pavlidis [402], and Harkens *et al.* [185]. Schmid *et al.* [343] compared various corner detectors against repeatability and information content and found the Harris and Stephens corner detector [184] producing the best repeatability and information content over a wide range of images.

Scale and affine invariant point feature detectors have also been developed. Crowley and Parker [78] used locally maximum or minimum points in the scale-space image as scale-invariant point features. Lowe [249] located locally maximum points in the scale-space pyramid built from the difference of Gaussian operators. He later improved the detector [250] so it would not respond to straight edges. An affine invariant point feature detector is described by Deriche and Giraudon [87], and Alvarez and Morales [6] where a chain of points is first detected through affine morphological muti-scale analysis and then the bisector of the chain is taken as the position and orientation of a point feature. Mikolajczyk and Schmid [275] combined the Harris and Stephens corner detector [184] with Laplacian-based scale detection [103]. The obtained detector was then made affine invariant.

Line detection by Hough transform has been studied extensively, a survey of which is given by Illingworth and Kittler [204]. In spite of considerable work, Hough transform has not emerged as a practical method in line detection. Rather, more generally applicable methods developed by Burns et al. [50], Mansouri *et al.* [258], and MacVicar-Whelan and Binford [252] have been used to detect lines in images.

Image regions and, in general, binary features have been extensively used in chamfer matching to locate objects of interest (binary templates) in an image or to align two binary images [39, 395]. The use of regions for registration of satellite images has been described by Goshtasby *et al.* [164]. Le Moigne *et al.* [238] have compared various image features in image registration.

4

Feature Correspondence

In the preceding chapter, methods for selecting features in an image were discussed. In this chapter, methods for determining corresponding features in sensed and reference images are discussed. Also, given two sets of features obtained from the reference and sensed images independently, methods for determining the correspondence between them are discussed. Using a set of corresponding point features in two images, methods for determining a transformation function that registers the images will be discussed in the next chapter.

4.1 POINT PATTERN MATCHING

The most desired features for image registration are points because their coordinates can be directly used to determine the parameters of a transformation function that registers the images. Point features are either determined directly or are determined from the intersections of lines, centroids of regions, or extrema of image contours. We assume that two images are available and a set of points are given in each image. The objective is to determine the correspondence between the two sets of points. Due to noise and other factors, it is possible that some points appear in only one of the sets. Such points are called *outliers*. It is also possible that some points shift from their true positions due to quantization and/or image noise. We would like to identify the outliers and determine the correspondence between remaining points in the images. In this section, we assume that the coordinates of the points are the only information available. In the following sections, we will consider cases where lines, regions, and templates are associated with the points and use the additional information to find the correspondences.

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

The problem to be solved is as follows. Given a set of points in the reference image, $\mathbf{P} = \{\mathbf{p}_i : i = 1, \ldots, n_r\}$, and a set of points in the sensed image, $\mathbf{Q} = \{\mathbf{q}_j : j = 1, \ldots, n_s\}$, where $\mathbf{p}_i = (x_i, y_i)$ and $\mathbf{q}_j = (X_j, Y_j)$, we would like to determine all index pairs (i, j) from the two point sets where $\mathbf{p}_i$ and $\mathbf{q}_j$ show the same point in the scene. Since only positional information is available about the points, constraints such as relative distance between points or relative orientation of lines connecting the points can be used to find the correspondences.

We assume that nonlinear geometric difference between the images is negligible. Therefore, corresponding points in the images can be related by affine or linear transformation

$$\begin{aligned} X &= ax + by + c, \\ Y &= dx + ey + f, \end{aligned} \tag{4.1}$$

where (X, Y) are the coordinates of points in the sensed image and (x, y) are the coordinates of points in the reference image. In 3-D, this linear relation becomes

$$\begin{aligned} X &= ax + by + cz + d, \\ Y &= ex + fy + gz + h, \\ Z &= ix + jy + kz + l. \end{aligned} \tag{4.2}$$

4.1.1 Matching using scene coherence

Because of scene coherence, if the correspondence between three non-colinear points in the two point sets is known, the remaining corresponding points in the two sets will be known by aligning the point sets at the three points. This, of course, requires that the three points correctly correspond to each other. If the points contain errors and only approximately correspond to each other, by aligning them, other points in the two sets will also approximately align. To determine the likelihood that the three selected points truly correspond to each other, the parameters of transformation (4.1) are determined from the three correspondences. The obtained transformation is then used to map points in set $\mathbf{Q}$ to points in set $\mathbf{P}$. Next, a measure of match between points in set $\mathbf{P}$ and points in transformed set $\mathbf{Q}$ is determined and, if the match-rating is sufficiently high, the three selected point pairs are considered corresponding points.

The match-rating between two point sets can be determined using the Hausdorff distance, named after Felix Hausdorff (1968–1942). The Hausdorff distance between sets $\mathbf{P}$ and $\mathbf{Q}$ is the maximum of minimum distances between points in $\mathbf{P}$ to points in $\mathbf{Q}$ [202]. That is,

$$h(\mathbf{P}, \mathbf{Q}) = \max_{\mathbf{p}\epsilon\mathbf{P}} \min_{\mathbf{q}\epsilon\mathbf{Q}} ||\mathbf{p} - \mathbf{q}||, \tag{4.3}$$

where $||\cdot||$ denotes some kind of a distance metric, such as the Euclidean distance. A small Hausdorff distance is evidence that the closest points in the two sets very likely correspond to each other. A large Hausdorff distance, on the other hand, is

evidence that closest points in the two sets are not likely to correspond to each other. Hausdorff distance produces a reliable match-rating between two points sets if outliers do not exist. Since the point sets obtained from two images often contain outliers, in order to use Hausdorff distance to compute the match rating between the two sets, correspondences whose distances are above a threshold value are considered outliers and are not used in the calculations. Remaining points in the two sets are then used to calculate the match-rating. Alternatively, the distances may be ranked from the smallest to the largest and the kth smallest distance may be used as the match rating [296], where k is smaller than both n_r and n_s. This process automatically removes the outliers.

Therefore, to find correspondence between two point sets, three non-colinear point pairs are selected from the images and hypothesized as corresponding points. The hypothesis is then verified by finding the number of other points in the two sets that fall within a small distance of each other with the transformation obtained from the three correspondences. The three point pairs producing the most correspondences will determine parameters a–f of the transformation and the points falling within a small distance of each other by the obtained transformation will be considered corresponding points. The following algorithm matches two point sets that are related by the affine transformation using scene coherence.

Algorithm 4.1: Given point sets $\mathbf{P} = \{\mathbf{p}_i = (x_i, y_i) : i = 1, \ldots, n_r\}$ and $\mathbf{Q} = \{\mathbf{q}_j = (X_j, Y_j) : j = 1, \ldots, n_s\}$, this algorithm finds the correspondence between the points and returns the result in list $\mathbf{L} = \{(i_k, j_k) : k = 1, \ldots, n\}$, where i_k and j_k show the indices of corresponding points in $\mathbf{P}$ and $\mathbf{Q}$. It is assumed that points in the two sets are related by the affine transformation given by (4.1), and the allowed error tolerance in the correspondences is D pixels.

1: Create empty list $\mathbf{L}$.

2: Select three points from set $\mathbf{P}$ and three points from set $\mathbf{Q}$ and let $n = 0$.

3: Determine parameters a-f of the affine transformation given by (4.1) using the coordinates of the three point pairs.

4: For $j = 1, \ldots, n_s$:
   4.1: Substitute point $\mathbf{q}_j$ into (4.1) and find the corresponding point $\mathbf{q'}_j$ in the reference image.

   4.2: Find the point in set $\mathbf{P}$ that is closest to $\mathbf{q'}_j$. Let $\mathbf{p}_i$ be such a point. If $d = |\mathbf{p}_i - \mathbf{q'}_j| < D$, save $[i, j]$ in entry n in list $\mathbf{L}$ and increment n by 1.

5: If n is sufficiently large, go to step 6. Otherwise, if no more choices of point triples in $\mathbf{P}$ and $\mathbf{Q}$ exist, report failure and stop. Otherwise, go to step 2.

6: From the coordinates of corresponding points in $\mathbf{L}$, determine the transformation parameters by the least-squares method and, using the newly obtained

transformation, map set $\mathbf{Q}$ to set $\mathbf{P}$ and determine the correspondences again as in step 4.

In step 2, a strategy to select three points from each image is needed. When the number of points in each set is small ($n \leq 10$), all combinations of point triples may be used. However, for very large point sets, an exhaustive search is prohibitive as the computational complexity of the algorithm will be in the order of $n_r^3 n_s^3$. Methods to reduce the search space considerably without significantly reducing the probability of missing the correct match have been developed. This includes using only points on the convex hulls [163] or the minimum-spanning-trees [427] of the two sets or using randomly selected subsets of the two point sets when determining the transformation parameters [117].

Once a sufficiently large number of correspondences is found, Algorithm 4.1 can be stopped. For instance, if by matching three points in the images, more than half of the points in the two sets match, the process may be stopped. The addition of step 6 to the algorithm is to reduce the effect of noise in the point triples that determined the correspondences in step 4. If the three point pairs truly correspond to each other but their positions are slightly shifted with respect to each other due to noise, by aligning them, there may be some corresponding points in the images that will not fall close enough to each other with the provided distance tolerance D. By recomputing the transformation using a larger number of points with the least-squared method the effect of noise is reduced, finding a larger number of correspondences.

If the images are related by the projective transformation, the transformation to be used is

$$X = \frac{ax + by + c}{dx + ey + 1}, \tag{4.4}$$

$$Y = \frac{fx + gy + h}{dx + ey + 1}. \tag{4.5}$$

Projective transformation should be used when the images are obtained from platforms that are sufficiently far from the scene and therefore their distances to the scene are much larger than depth variations in the scene. The images may be obtained with large view-angle differences. The projective transformation has eight parameters, which can be determined if the coordinates of four corresponding points in the images are known. Therefore, in Algorithm 4.1, four point pairs in the images are needed to determine parameters a–h. The correctness of the correspondences can then be verified by counting the number of other points in the images that also fall within distance D of each other by the obtained transformation.

The process in 3-D is the same as that in 2-D except that, instead of three non-colinear point pairs, four non-coplanar point pairs are used to calculate parameters a–l of transformation (4.2). The transformation obtained as a result is then used to determine the correspondence between remaining points in the images. The four-point pair producing a sufficiently large number of correspondences is used to find the

parameters of the transformation and determine the correspondence between remaining points in the two sets according to Algorithm 4.1.

4.1.2 Matching using clustering

In clustering, the transformation parameters are estimated through a voting process. For the affine imaging model, six 1-D accumulator arrays $A[\,], B[\,], C[\,], D[\,], E[\,], F[\,]$ are used to estimate parameters a–f of the transformation. The accumulator arrays are initially set to 0. Then, for each combination of three point pairs, parameters a–f are determined and corresponding entries in arrays $A[\,]$ through $F[\,]$ are incremented by 1. After testing a sufficiently large number of combinations, the entry showing the peak count in each array is used as the transformation parameter. The parameters obtained in this manner match the most points in the two sets. The obtained parameters are then used in relation (4.1) to map points in set $\mathbf{Q}$ to points in set $\mathbf{P}$. Points falling within a small distance of each other are then considered corresponding points. The following algorithm shows the steps of point-pattern matching by clustering.

Algorithm 4.2: Given point sets $\mathbf{P} = \{\mathbf{p}_i = (x_i, y_i) : i = 1, \ldots, n_r\}$ and $\mathbf{Q} = \{\mathbf{q}_j = (X_j, Y_j) : j = 1, \ldots, n_s\}$, this algorithm determines the correspondence between points in the two sets and returns the correspondences in list $\mathbf{L}$. N is the allowed number of three point pairs in the two sets that is tested to find the correspondence between the points.

1: Create arrays $A[\ ], B[\ ], \ldots, F[\ ]$ of size N and initialize entries of the arrays to 0. Also, initialize $\mathbf{L}$ to an empty list and let $n = 0$.

2: Find ranges of parameters a-f. Let these ranges be $[a_1, a_2]$, $[b_1, b_2], \ldots, [f_1, f_2]$.

3: Find a large number of three-point combinations in $\mathbf{P}$ and $\mathbf{Q}$ and for each such combination, say $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$, if $\varepsilon_1 < \frac{||\mathbf{p}_1 - \mathbf{p}_2||}{||\mathbf{q}_1 - \mathbf{q}_2||} \times \frac{||\mathbf{q}_1 - \mathbf{q}_3||}{||\mathbf{p}_1 - \mathbf{p}_3||} < \varepsilon_2$ do the following.

   3.1: Find parameters a-f by substituting the point coordinates into (4.1) and solving the obtained equations.
   3.2: Increment $A[i_a]$ by 1, where $i_a = \frac{(N-1)(a-a_1)}{a_2-a_1}$.
   3.3: In the same manner, increment entries of B, C, D, E, and F by 1.

4: If $A[i] > A[j]$ for all $j \neq i$, let $i_a = i$. Similarly, save indices of maximum entries in B, C, D, E, and F in i_b, i_c, i_d, i_e, and i_f, respectively.

5: Find a from i_a; $a = a_1 + i_a(a_2 - a_1)/(N-1)$. Similarly, from i_b, i_c, i_d, i_e and i_f find b, c, d, e and f.

6: Knowing parameters a-f, for $j = 1, \ldots, n_s$:
   6.1: Substitute point $\mathbf{q}_j$ into (4.1) and find the corresponding point $\mathbf{q}'_j$ in the reference image.

```
6.2:  Find the point in set P that is closest to q'_j.
      Let p_i be such a point.  If d = |p_i - q'_j| < D, save
      [i, j] in entry n in list L and increment n by 1.

7:  Return list L.
```

Since it is necessary to access the accumulator entries from the estimated parameters a–f computed by matching three point pairs in the two sets, it is required that the ranges of a–b be known. Therefore, step 3.1 may be carried out once to determine the ranges. Knowing the ranges, step 3.1 can then be carried out again to create the clusters. This, however, is not very efficient. A more efficient approach would be to estimate the ranges of the parameters by approximating the affine transformation with the transformation of the Cartesian coordinate system:

$$\begin{aligned} X &= (x - x_0)S\cos(\theta) + (y - y_0)S\sin(\theta) + h + X_0, \\ Y &= -(x - x_0)S\sin(\theta) + (y - y_0)S\cos(\theta) + k + Y_0. \end{aligned} \tag{4.6}$$

By using the possible ranges of the translational (h, k), rotational θ, and scaling S, the affine coefficients can be estimated. Coordinates (x_0, y_0) and (X_0, Y_0) are the coordinates of the image centers and are known. The scaling difference between the image can, for example, vary between 0.5 and 2, the rotational difference between the images can vary, for example, from $-\pi/4$ to $\pi/4$, and the translational difference between the images can vary horizontally and vertically, for example, by half the image width and height. These extreme translation, rotation, and scale differences can produce approximate minimum and maximum values for a–f.

Step 3.2 of the algorithm maps the maximum and minimum values of a–f to accumulator array entries 0 and $(N - 1)$. The size of the accumulator arrays is set to N. Very large accumulators will make the arrays very sparse, increasing the chance of missing the correct parameters. Very small accumulators, on the other hand, will produce very coarse transformation parameters, missing some correct correspondences, and incorrectly picking some correspondences. Setting the accumulator array size proportional to the number of point triples tested ensures that when more point combinations from the two sets are tested, a more accurate set of transformation parameters will be obtained.

In step 3, when three points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ are selected from $\mathbf{P}$ and three points $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ are selected from $\mathbf{Q}$, $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ may correspond to $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$, $\mathbf{q}_1\mathbf{q}_3\mathbf{q}_2$, $\mathbf{q}_2\mathbf{q}_1\mathbf{q}_3$, $\mathbf{q}_2\mathbf{q}_3\mathbf{q}_1$, $\mathbf{q}_3\mathbf{q}_1\mathbf{q}_2$, or $\mathbf{q}_3\mathbf{q}_2\mathbf{q}_1$. Therefore, each point triple from the two sets requires six matches, creating six votes, only one of which may be correct.

Step 3 maps the estimated parameters a–f for a particular triple of point correspondences to the accumulator array entries and votes by incrementing the entries by 1. As the point triples are tested, correct correspondences vote for the same parameters, while incorrect correspondences vote randomly. After trying a sufficiently large number of point combinations, accumulator entries corresponding to the correct transformation parameters become larger than other entries. The largest entry in each array is then located in step 4, and the indices of the peak entries are mapped back to

the parameter space in step 5. Step 6 then obtains the transformation parameters and finds the entire set of correspondences.

How much larger will entries corresponding to the correct parameters be compared to other entries in an accumulator array? To answer that, let's suppose that we only use points on the convex hulls of the two sets to determine the transformation parameters. This will reduce the computation time considerably by significantly reducing the number of hypotheses to be tested while maintaining sufficient points in the two sets to find the correct transformation parameters. Suppose that there are n points on the convex hull of each set, there are no outliers, and points in the two sets correctly correspond to each other. Under these perfect conditions, if we follow Algorithm 4.2, we will have $n(n-1)(n-2)/6$ combinations of three points in set $\mathbf{P}$. We will also have $n(n-1)(n-2)/6$ combinations of three points in set Q. Since ordered points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ may match $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ or $\mathbf{q}_3\mathbf{q}_2\mathbf{q}_1$, in total there will be $N_v = 2\left[n(n-1)(n-2)/6\right]^2$ cases to be considered, each producing a vote. Therefore, if an accumulator array contains N entries, each entry will receive on average $N_1 = N_v/N$ votes.

Among the N_v combinations of point triples from the two sets, $n(n-1)(n-2)/6$ of them will find the correct transformation parameters, and thus, vote for the same entries in the accumulators. Therefore, under perfect conditions, the entry corresponding to the correct parameter in each accumulator array will receive $N_2 = N_1 + n(n-1)(n-2)/6$ votes and all other entries will receive on average N_1 votes. When $n = 10$ and $N = 1000$, we find that $N_1 = 309$ and $N_2 = 429$, and thus, $N_2/N_1 = 1.38$. This means that under perfect conditions, the entry corresponding to the correct parameter in each accumulator receives only 38% more votes than other entries in the accumulator. Considering the fact that, in reality, not all points exist in both sets and points could be shifted due to quantization error or noise, it is quite possible to obtain an accumulator where the largest entry does not correspond to the correct transformation parameter. This has been verified by Grimson and Huttenlocher [170]. If used naively, clustering could often pick the wrong transformation parameters.

To reduce the number of votes that all entries of the accumulator arrays receive, we should not count votes that do not satisfy certain constraints. In step 3 of Algorithm 4.2, if the ratio of distances between $\mathbf{p}_1\mathbf{p}_2$ and $\mathbf{q}_1\mathbf{q}_2$ is much different than the ratio of distances between $\mathbf{p}_1\mathbf{p}_3$ and $\mathbf{q}_1\mathbf{q}_3$, it is concluded that the points cannot correspond to each other, and therefore, their vote is not counted. However, if the ratio of lengths of corresponding line segments in the two images are close, it is possible that the line segments, and thus the points, correspond to each other. In that case, their vote is counted. The terms ε_1 and ε_2 are numbers close to 1. For example, when $\varepsilon_1 = 0.75$ and $\varepsilon_2 = 1.5$, the affine transformation can stretch an image in one direction up to 1.5 times more than in another direction during registration. This constraint filters out a large proportion of the votes that stretch an image more than the specified amount and cannot contribute to the correct transformation parameters.

Although not included in Algorithm 4.2, it is a good idea to include one last test in step 6 before ending the algorithm. If the number of correspondences (n) is smaller than a threshold value, the correspondences can be wrong. In such a situation, the process needs to be continued from step 3. To increase the ratio of

correct correspondences with respect to the total number of tests carried out using three-point combinations, one may choose points on the convex hulls or minimum-spanning-trees of the two sets. Selection in this manner results in a more efficient search for the transformation parameters. Note that if points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ and $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ belong to the convex hulls of the two sets, there is only a need to match $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ with $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ and $\mathbf{q}_3\mathbf{q}_2\mathbf{q}_1$. Other point combinations are not possible to occur because points on a convex hull are ordered in either clockwise or counter-clockwise direction.

If the imaging geometry is a projective one as given by (4.5), Algorithm 4.2 remains unchanged but instead of six accumulator arrays, eight accumulator arrays corresponding to parameters a–h of the transformation should be used and, as a consequence, combinations of four points from each set should be used to estimate the transformation parameters. In 3-D, Algorithm 4.2 also remains unchanged except that combinations of four non-coplanar points in each set and twelve accumulator arrays are needed to estimate the parameters of transformation (4.2).

Clustering to estimate the values of six parameters of the affine transformation in 2-D generally requires a 6-D accumulator to count the votes. This, however, is not practical. If the 6-D space contains one dominant cluster as is the case in our situation, the cluster center can be determined equally well by projecting the 6-D space into the six axes and locating the cluster center on each axis. The reason for using six 1-D arrays as opposed to a 6-D array in Algorithm 4.2 is that when a dominant cluster is present, both methods produce the same result and use of six 1-D arrays is much more efficient than use of a 6-D array.

4.1.3 Matching using invariance

Knowing the positions of three non-colinear points $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ in the sensed image, the relation of other points in the image with respect to the three points can be written as

$$\mathbf{q} = \mathbf{q}_1 + \alpha_1(\mathbf{q}_2 - \mathbf{q}_1) + \alpha_2(\mathbf{q}_3 - \mathbf{q}_1). \tag{4.7}$$

If the relation between the reference and sensed images can be defined by an affine transformation, we can rewrite relation (4.1) by

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.8}$$

or

$$\mathbf{q} = \mathbf{A}\mathbf{p}. \tag{4.9}$$

Substituting (4.9) into (4.7), we obtain

$$\mathbf{A}\mathbf{p} = \mathbf{A}\left[\mathbf{p}_1 + \alpha_1(\mathbf{p}_2 - \mathbf{p}_1) + \alpha_2(\mathbf{p}_3 - \mathbf{p}_1)\right] \tag{4.10}$$

and by multiplying both sides by $\mathbf{A}^{-1}$, we obtain

$$\mathbf{p} = \mathbf{p}_1 + \alpha_1(\mathbf{p}_2 - \mathbf{p}_1) + \alpha_2(\mathbf{p}_3 - \mathbf{p}_1). \tag{4.11}$$

This shows that the same relation holds between the corresponding points in the reference image. Using this affine invariant property, combinations of three point pairs can be selected from the images and the relation between remaining points in the images with respect to the three points can be determined and compared. Comparison can be achieved by quantifying the relation obtained for each image into an array and measuring the similarity between the two arrays. If the three points selected in the images truly correspond to each other, relations of other points in the images with respect to the three points will be similar. The relations can be quantized into arrays and the similarity of the arrays can be used to verify the correctness of the three correspondences. The following algorithm outlines the steps of the process.

Algorithm 4.3: Given two sets of points, $\mathbf{P} = \{\mathbf{p}_i = (x_i, y_i) : i = 1, \ldots, n_r\}$ and $\mathbf{Q} = \{\mathbf{q}_j = (X_j, Y_j) : j = 1, \ldots, n_s\}$, this algorithm determines the correspondence between the points and returns the correspondences in list $\mathbf{L}$ using the affine invariance property. The affine parameters obtained for each image are quantized into an $m \times m$ parameter array. S_m is the minimum similarity between the parameter arrays from the two images for the correspondences to be considered correct.

1: Create two arrays $H_1[\ ]$ and $H_2[\ ]$ of size $m \times m$ and initialize their entries to 0. Also, create empty list $\mathbf{L}$ and let $n = 0$.

2: Select a combination of 3 points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ from $\mathbf{P}$. Then compute α_1 and α_2 for each additional point $\mathbf{p}$ in $\mathbf{P}$ using $\mathbf{p} = \mathbf{p}_1 + \alpha_1(\mathbf{p}_2 - \mathbf{p}_1) + \alpha_2(\mathbf{p}_3 - \mathbf{p}_1)$ and find ranges of α_1 and α_2. Let the ranges be $[\alpha_{11}, \alpha_{12}], [\alpha_{21}, \alpha_{22}]$.

3: For the same combination of three points, recalculate parameters α_1 and α_2 for each additional point, and let $i = \frac{(m-1)(\alpha_1 - \alpha_{11})}{(\alpha_{12} - \alpha_{11})}$ and $j = \frac{(m-1)(\alpha_2 - \alpha_{21})}{(\alpha_{22} - \alpha_{21})}$. Then increment entry $[i, j]$ of H_1 by 1.

4: For a combination of 3 points $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ in $\mathbf{Q}$, compute α_1 and α_2 for each additional point $\mathbf{q}$ in $\mathbf{Q}$ using $\mathbf{q} = \mathbf{q}_1 + \alpha_1(\mathbf{q}_2 - \mathbf{q}_1) + \alpha_2(\mathbf{q}_3 - \mathbf{q}_1)$. Then compute $i = \frac{(m-1)(\alpha_1 - \alpha_{11})}{(\alpha_{12} - \alpha_{11})}$ and $j = \frac{(m-1)(\alpha_2 - \alpha_{21})}{(\alpha_{22} - \alpha_{21})}$ and increment entry $[i, j]$ of H_2 by 1.

5: Find the *similarity* between $H_1[\ ]$ and $H_2[\ ]$. If the *similarity* $> S_m$, go to step 8.

6: If another combination of 3 points in $\mathbf{Q}$ is available or can be afforded, go to step 4.

7: If another combination of 3 points in $\mathbf{P}$ is available or can be afforded, go to step 2. Otherwise, report failure.

8: From the obtained corresponding points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ and $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ compute coefficients a-f of the transformation.

9: Transform set $\mathbf{Q}$ with the obtained transformation. Let the transformed set be $\mathbf{Q}' = \{\mathbf{q}'_j : j = 1, \ldots, n_s\}$.

10: For each point $\mathbf{q}'_j$ in $\mathbf{Q}'$, find point in $\mathbf{P}$ closest to it. Let $\mathbf{p}_i$ be the point and d_j be the distance between $\mathbf{p}_i$ and $\mathbf{q}'_j$. Then, if $d_j < D$, save $[i, j]$ at entry n in list $\mathbf{L}$ and increment n by 1.

11: Return list $\mathbf{L}$.

The size of arrays $H_1[\,]$ and $H_2[\,]$ determines the matching coarseness. Very large arrays produce sparse arrays that, when matched, could miss the correct correspondences, especially when the points are noisy, thus contributing to the false negative error. On the other hand, very small arrays may count points that do not correspond to each other as corresponding points, contributing to the false positive error. Array size should be selected taking into consideration the noisiness of the point sets. Experiments on data sets with small amounts of noise (up to a pixel positional error) show that if the reference and sensed images are of size $m \times m$, arrays H_1 and H_2 should be from $\frac{m}{4} \times \frac{m}{4}$ to $m \times m$. The smaller arrays should be used when smaller point sets are available and the larger arrays should be used when larger point sets are given.

In steps 2, 4, 6, and 7 of the algorithm, combinations of three points are selected from the point sets and tested for correspondences. Since the number of such combinations can be prohibitively large when the given data sets are large, a mechanism to choose the points should be devised. For instance, points on the convex hulls or the minimum-spanning-trees of the two sets may be used. If points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ in $\mathbf{P}$ and points $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ in $\mathbf{Q}$ belong to the convex hulls of the points taken clockwise or counter-clockwise, points $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ should be matched with $\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ once and then with $\mathbf{q}_3\mathbf{q}_2\mathbf{q}_1$. Array $H_2[\,]$ determined in each case should be compared with array $H_1[\,]$ in step 5 to find the point triples in $\mathbf{P}$ and $\mathbf{Q}$ that correspond to each other.

Step 2 is repeated again in step 3 because there is a need to find the ranges of parameters α_1 and α_2. The ranges of the parameters are needed to effectively map the obtained parameters into the indices of the arrays.

For small points sets (containing fewer than m points), arrays $H_1[\,]$ and $H_2[\,]$ will be nearly binary and the similarity between the two arrays can be determined by a measure such as $similarity = COUNT(H_1.OR.H_2)/n$, where $H = H_1.OR.H_2$ is an array with an entry having value 0 when H_1 and H_2 are both 0 at that entry, and having value 1 otherwise. $COUNT(H)$ is a function that returns the number of entries in H that have value 1, and n is the smaller of n_s and n_r. The measure, $similarity$, therefore, will be a value between 0 and 1. A $similarity$ value as low as 0.2 can trigger a correct matching configuration as it implies that 20% of the points in the sensed image match with those in the reference image under the 3-point configuration that is being tested. Therefore, S_m in Algorithm 4.3 can be as low as

0.2 and still find the correspondences. Use of a very large similarity threshold value such as 0.8 or 0.9 could miss the correspondences.

For each combination of three non-colinear points in $\mathbf{P}$ and three non-colinear points in $\mathbf{Q}$, images $H_1[\,]$ and $H_2[\,]$ are generated. Although there are many such combinations, whenever the similarity between $H_1[\,]$ and $H_2[\,]$ reaches a threshold similarity S_m, the algorithm is stopped. It is sufficient to find three points in $\mathbf{P}$ that correspond to three points in $\mathbf{Q}$ in order to find the correspondence between remaining points in the two sets.

Although not mentioned in Algorithm 4.3, it is a good idea to recalculate the transformation parameters from the obtained correspondences by the least-squares method just like in step 6 of Algorithm 4.1 and recompute the correspondences from the refined transformation. This is necessary so that, if the initial transformation parameters are obtained from three noisy corresponding points, the transformation parameters are refined using a larger number of correspondences. The least-squares method will reduce the effect of noise and improve the correspondence reliability.

If the images are related by the projective transformation, we use transformation (4.5) to represent the relation between the two point sets. Transformation (4.5) in matrix form can be written as

$$\begin{bmatrix} WX \\ WY \\ W \end{bmatrix} = \begin{bmatrix} a & b & c \\ f & g & h \\ d & e & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.12}$$

or,

$$W\mathbf{q} = \mathbf{B}\mathbf{p}. \tag{4.13}$$

Under the projective imaging geometry, the following two relations remain invariant between five non-colinear points [68].

$$I_1 = \frac{det[\mathbf{m}_{431}]det[\mathbf{m}_{521}]}{det[\mathbf{m}_{421}]det[\mathbf{m}_{531}]}, \tag{4.14}$$

$$I_2 = \frac{det[\mathbf{m}_{421}]det[\mathbf{m}_{532}]}{det[\mathbf{m}_{432}]det[\mathbf{m}_{521}]} \tag{4.15}$$

where

$$det[\mathbf{m}_{123}] = \begin{vmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{vmatrix}. \tag{4.16}$$

Substituting (4.13) into (4.14) and (4.15) and simplifying the relations, we obtain

$$I_1 = \frac{det[\mathbf{n}_{431}]det[\mathbf{n}_{521}]}{det[\mathbf{n}_{421}]det[\mathbf{n}_{531}]}, \tag{4.17}$$

$$I_2 = \frac{det[\mathbf{n}_{421}]det[\mathbf{n}_{532}]}{det[\mathbf{n}_{432}]det[\mathbf{n}_{521}]} \tag{4.18}$$

where

$$det[\mathbf{n}_{123}] = \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}. \tag{4.19}$$

Measures I_1 and I_2 remain unchanged under the projective transformation. To revise Algorithm 4.3 so it will use the projective invariance, instead of combinations of three points, combinations of four points are used, and for each additional point, I_1 and I_2 are used instead of α_1 and α_2 to create arrays $H_1[\,]$ and $H_2[\,]$. A sufficiently high similarity between the two arrays will again identify a corresponding set of four points, which can then be used to determine the transformation parameters. The obtained transformation can be used to determine the correspondence between remaining points in the two sets.

Among the point-pattern matching algorithms discussed above, Algorithm 4.1 is the simplest to implement and Algorithm 4.2 is the slowest as it requires a large number of tests to build the 1-D clusters, the centers of which determine the parameters of the transformation. There is a possibility that Algorithm 4.1 or Algorithm 4.3 find the correspondences at the first attempt. It is also possible that they may not find the correspondences after a large number of attempts. The order in which the tests are performed greatly affects the speed of these algorithms.

Algorithms 4.1 and 4.3 find the transformation parameters through an averaging process and if noise is not zero-mean, the averaging process could shift the obtained transformation parameters from the true values. The clustering method selects the transformation parameters through a voting process, without averaging. If most points contain very little noise, Algorithm 4.2 finds the transformation parameters without averaging and, thus, is capable of finding the registration parameters more accurately than Algorithms 4.1 and 4.3.

The difficulty in Algorithms 4.2 and 4.3 is in the selection of the accumulator arrays and the determination of the ranges of parameters the accumulator arrays are mapped to. Optimal values of the accumulator sizes and ranges require knowledge about the images being registered. Absence of such knowledge requires experimentation to find the parameters of the algorithm through a training process. Overall, Algorithm 4.1 is the easiest to implement and is the most versatile as it does not have any free parameters to be determined.

4.2 LINE MATCHING

Line segments have position, orientation, and length. Usually, the orientation of a line is least influenced by noise and its length is most influenced by noise. If the reference and sensed images from which the lines are extracted have translational, rotational, and scaling differences, the rotational difference between the lines is first determined and then the translational and scaling differences between the lines are found.

When two images have translational, rotational, and scaling differences, the relation between them can be written by the transformation of the Cartesian coordinate system:

$$X = S[x\cos(\theta) + y\sin(\theta)] + t_x,$$
$$Y = S[-x\sin(\theta) + y\cos(\theta)] + t_y. \tag{4.20}$$

Let's suppose two sets of lines $\mathbf{K} = \{k_i : i = 1, \ldots, n_r\}$ and $\mathbf{L} = \{l_j : j = 1, \ldots, n_s\}$ are available and it is required to determine the correspondence between the lines as well as the translational, rotational, and scaling differences between them. Since a line in one image may break into two or more lines in another image, it is possible that two or more lines in one set correspond to the same line in another set. It is also possible that some lines appear in only one of the sets.

To determine the rotational difference between lines in the two sets, two lines are selected, one from each set (say, k_i and l_j), and hypothesized as corresponding lines. The correctness of the hypothesis is then verified using information about other lines in the two sets. The angular difference between two directed lines is a value between 0 and 359 degrees. However, the provided lines may not contain directional information. In other words, it may not be known which endpoint of a line is the beginning and which endpoint is the end in an image. We will consider the endpoint that has the smaller y-coordinate as the beginning but we understand that when the angular difference between two lines is determined, it may be off by 180 degrees. Therefore, if the rotational difference between two lines is found to be θ degrees, we also consider $\theta + 180$ a possibility and find the correct angle among the two using further evidence among the lines.

If $\angle(k_i)$ and $\angle(l_j)$ are the angles of lines k_i and l_j with the x-axis, the angle between the two lines is $\theta_{ij} = \angle(l_j) - \angle(k_i)$. If lines k_i and l_j truly correspond to each other and θ_{ij} is their true rotational difference, by rotating the lines in set $\mathbf{L}$ by $-\theta_{ij}$, all corresponding lines in the two sets will have the same or very close orientations. The following algorithm takes advantage of this coherence property to determine the translational, rotational, and scaling differences between two images from their lines.

Algorithm 4.4: Given two sets of lines $\mathbf{K} = \{k_i : i = 1, \ldots, n_r\}$ and $\mathbf{L} = \{l_j : j = 1, \ldots, n_s\}$, this algorithm determines the translational, rotational, and scaling differences between the two line sets.

1: Create two accumulator arrays $A[\ ]$ and $B[\ ]$, each with 181 entries and initialize the entries to 0.

2: Find the angles the lines in $\mathbf{K}$ and $\mathbf{L}$ make with the x-axis in degrees and save them with the lines. Round the angles so the values will be between 0 and 180 (see Note 1 below).

3: For each line in $\mathbf{K}$, if the angle of the line is i, increment $A[i]$ by 1.

4: For each line in $\mathbf{L}$, if the angle of the line is j, increment $B[j]$ by 1.

5: Find S_θ by changing θ from 0 to 180 by 1-degree increments (see Note 2) and find $\theta_m = MAX\{S_\theta\}$, for $\theta =$ 0 to 180 with 1-degree increments. Either θ_m or $\theta_m + 180$ is the rotational difference between the images.

6. Let the index of the entry with the largest value in $A[\ ]$ be i_m. Take a line in $\mathbf{K}$ that makes angle i_m with the x-axis and call the line k.
   a. Take a line in $\mathbf{L}$ that makes angle $i_m + \theta_m$ with the x-axis. Call the line l and find the translational and scaling differences between the two lines (see Note 3).
   b. Determine the likelihood that the obtained scaling and translational parameters are correct (see Note 4). Repeat steps 6a and 6b for all combinations of lines in $\mathbf{K}$ that have angle i_m and all lines in $\mathbf{L}$ that have angle $i_m + \theta_m$. Choose the line pair producing the highest likelihood as matching lines and determine the registration parameters from them (see Note 5).

Note 1. If a line is defined by end points $\mathbf{p}_1$ and $\mathbf{p}_2$, denote the end point with the smaller y-coordinate $\mathbf{p}_1$. Then, the angle between $\mathbf{p}_1\mathbf{p}_2$ and the x-axis will be a value between 0 and 180 degrees.

Note 2. Denoting the cyclic rotation of array $B[\,]$ by θ degrees by $B_\theta[\,]$, the similarity between $A[\,]$ and $B_\theta[\,]$ can be defined by

$$S_\theta = 1 - \sum_{i=0}^{180} \left| \frac{1}{n_r} A[i] - \frac{1}{n_s} B_\theta[i] \right|. \tag{4.21}$$

where $B_\theta[i] = B[i + \theta]$. Note that when $i + \theta > 180$, $B_\theta[i] = B[i + \theta - 180]$.

Note 3. The relation between two lines that have translational, rotational, and scaling differences is given by formula (4.20). Assuming (x, y) and (X, Y) are coordinates of line end points in $\mathbf{K}$ and $\mathbf{L}$, respectively, the scaling S between a line in $\mathbf{L}$ with respect to a line in $\mathbf{K}$ can be determined from the ratio of the length of line l over the length of line k. Knowing S and θ, translation parameters (t_x, t_y), can be determined by substituting the coordinates of the midpoints of lines k and l into (4.20) and solving for t_x and t_y.

Note 4. Knowing θ, S, t_x, t_y, substitute coordinates of line midpoints in $\mathbf{K}$ into (4.20) and see how many of them fall within a pixel or two of lines in $\mathbf{L}$. If among the m line midpoints in $\mathbf{K}$, m' fall within a pixel or two of the lines in $\mathbf{L}$, the likelihood will be m'/m. Note that if a line is defined in polar form $\rho = x\cos\theta - y\sin\theta$, the distance of point (x_i, y_i) to the line is $|\rho - x_i\cos\theta - y_i\sin\theta|$.

Note 5. The reason for selecting the angle that corresponds to the largest entry in $A[\,]$ is to allow more tests in finding the correct answer. If the lines are rather short, corresponding lines may be off by a few degrees due to noise and digital nature of images. Therefore in addition to testing lines with angle i_m in $\mathbf{K}$ and lines with angle $i_m + \theta_m$ in $\mathbf{L}$, tests should be performed on lines in $\mathbf{K}$ with angles in the range

$[i_m - \epsilon, i_m + \epsilon]$ and lines in $\mathbf{L}$ with angles in the range $[i_m + \theta_m - \epsilon, i_m + \theta_m + \epsilon]$, where ϵ is a small angle, such as one or two degrees.

Note that in steps 6a and 6b, if θ_m is found not to be the answer, $180 + \theta_m$ should be tested for a possible answer. Also note that in step 6a if $i_m + \theta_m > 180$, $i_m + \theta_m$ should be replaced with $i_m + \theta_m - 180$ so that when $i_m + \theta_m + 180$ is tested as a possible rotational difference between images, the angle used in the test will be less than 360 degrees.

Parameters (S, t_x, t_y) may be refined after finding the line correspondences by the above algorithm. Assuming $\mathbf{K}'$ represents $\mathbf{K}$ after the lines are transformed according to (4.20) with the estimated (S, θ, t_x, t_y), then, for each corresponding line pair, the midpoint of the line in $\mathbf{L}$ and the point closest to it on the corresponding line in $\mathbf{K}'$ are considered corresponding points. The coordinates of corresponding points are then used to determined parameters (S, t_x, t_y) by the least-squares method using (4.20). The midpoints of corresponding lines are not used as corresponding points and instead the midpoint of one line and the point closest to it on the corresponding line are considered corresponding points because a line segment in one image can break into two or more line segments in another image and one or more lines in one set may correspond to the same line in another set. Therefore, a midpoint of a line in one set may not correspond to the midpoint of the corresponding line in another set.

Note that θ is not recomputed in this refinement. Since line midpoints may contain errors, we do not want such errors to affect the estimation of θ. Also note that, once parameters (S, θ, t_x, t_y) are determined, relation (4.20) is used to resample the sensed image to the reference image. This involves finding the corresponding point (X, Y) in the sensed image for each point (x, y) in the reference image using relation (4.20), reading the intensity at (X, Y) in the sensed image and saving at (x, y) in the reference image, thereby mapping the sensed image point-by-point to the reference image. Algorithm 4.4 assumes that the images from which the lines are extracted have only translation, rotation, and scaling differences. Determination of transformation parameters when the images are related by the affine or projective transformation is an open problem.

4.3 REGION MATCHING

Region matching can be considered point matching if the centroids of the regions are used. In region matching, in addition to the positional information, information about the size and shape of the regions is available. This additional information can be used to find corresponding points in the images more reliably. An example of two sets of regions to be matched is shown in Fig. 4.1. Some regions may appear in only one of the images, two or more regions in one image may merge and form a single region in another image, and the shapes of some regions may change from one image to another due to segmentation errors. Most regions are, however, assumed to have correspondences and most corresponding regions are assumed to have similar shapes.

Using information about the shapes of regions, an initial guess can be made about the region correspondences. This is achieved via a shape-matching algorithm. Errors made in assigning the initial correspondences is then refined by an iterative (relax-

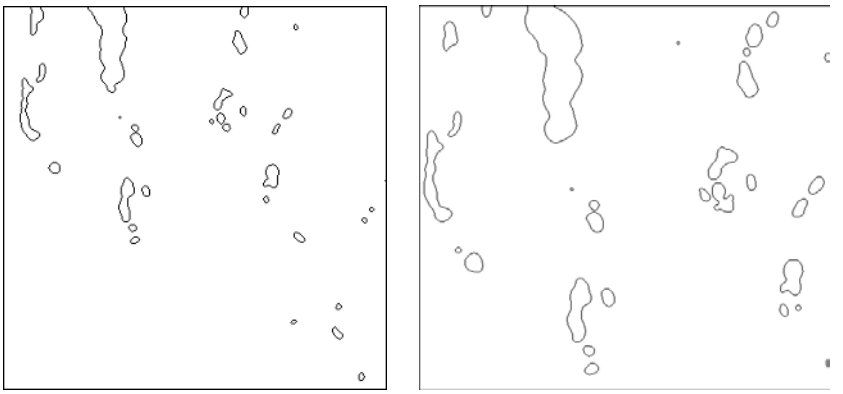


Fig. 4.1 (a), (b) Regions representing lakes in two satellite images.

ation) process that uses information about the relative positions and/or orientations of the regions. In the following, first, methods for shape matching and relaxation labeling are described. Then, their uses in region-matching are discussed.

4.3.1 Shape matching

Digital shapes can be matched by determining and comparing their various properties. The Fourier transform measures the spatial frequency characteristics of a shape. Using the Fourier transform coefficients, two shapes can be compared by their spatial frequency characteristics. Invariant moments measure the geometric layout of shapes independent of their positions, orientations, and scales and can also be used to compare shapes. Shape matrices represent shapes independent of their positions, orientations, and scales and can be used to compare shapes too. In the following, the use of these shape descriptors in shape matching is examined.

4.3.1.1 Fourier descriptors A digital shape represents a region boundary in an image and, therefore, it is an ordered sequence of pixels. Let's suppose the pixels defining a shape are

$$\{(x_i, y_i) : i = 0, \ldots, N-1\}. \tag{4.22}$$

The choice of the first pixel (x_0, y_0) is arbitrary. The sequence is cyclic, that is, starting from any pixel the same cyclic sequence is obtained. Denoting the real part of a complex number z_i by x_i and the imaginary part of z_i by y_i, we have $z_i = x_i + jy_i$, where $j = \sqrt{-1}$. The discrete Fourier transform of the complex sequence $\{z_i : i = 0, \ldots, N-1\}$ is

$$c_k = \frac{1}{N} \sum_{i=0}^{N-1} z_i \exp(-j2\pi ki/N), \qquad k = 0, \ldots, N-1. \tag{4.23}$$

The complex coefficients c_k are called the *Fourier descriptors* of the shape [167, 305]. The Fourier transform assumes that the given sequence represents uniformly spaced samples from one period of a periodic signal. In our case, pixels forming a closed digital shape form a cyclic sequence, representing uniformly spaced samples from a periodic signal. The Fourier transform describes the spatial frequency characteristics of a periodic signal and represents the signal in terms of its frequency characteristics. Lower order coefficients show the magnitude of lower spatial frequencies and higher order coefficients show the magnitude of higher frequencies in the signal. A very smooth signal (shape) will have very small high-order coefficients, while a noisy shape or a shape with a lot of details will have large high-frequency coefficients. The inverse Fourier transform restores a shape from its Fourier transform. That is,

$$z_i = \sum_{k=0}^{N-1} c_k \exp(j2\pi ki/N), \qquad i = 0, \ldots, N-1. \tag{4.24}$$

The magnitude of Fourier transform coefficients is invariant of the shift of z_i, and therefore is independent of the choice of the start point. It also means that if a shape is rotated, the magnitude of its Fourier transform coefficients will not change. This is a very desirable property and may be used to compare shapes independent of their orientations. Use of only the magnitude of the coefficients, however, implies discarding the phase information, and that reduces the matching reliability.

Since higher-order Fourier transform coefficients correspond to high-frequency details in a shape, which could be due to noise, the effect of noise in shape matching can be reduced by using only the lower half or two-third of the coefficients in shape matching. The correlation coefficient between the magnitude Fourier descriptors of two shapes can be used to determine the similarity between the shapes.

If a shape is normalized so that it has a fixed perimeter, such as 1, the obtained Fourier descriptor will be invariant of the position and scale of the shape, and the magnitude of the Fourier descriptors will be invariant of the position, orientation, and scale of the shape. Zahn ad Roskies [428] parametrized points along a shape from 0 to 1, creating a real sequence that represented cumulative angular bends of points along the shape. The sequence was then used to compute the Fourier descriptors of the shape.

4.3.1.2 Invariant moments

The $(p+q)$th order moment of a digital shape composed of points given in (4.22) is defined by [198],

$$m_{pq} = \sum_{i=0}^{N-1} x_i^p y_i^q f_i, \tag{4.25}$$

where f_i is the intensity at point (x_i, y_i). Generally, we have a binary shape and therefore, $f_i = 1$ for all i. However, shape points could have values representing the intensity, gradient, or other image properties.

Measure m_{pq} changes if the shape is translated. To make m_{pq} invariant of the shape's position, central moments defined by

$$u_{pq} = \sum_{i=0}^{N-1} (x_i - \bar{x})^p (y_i - \bar{y})^q f_i \tag{4.26}$$

are used, where

$$\bar{x} = \sum_{i=0}^{N-1} x_i f_i \Big/ \sum_{i=0}^{N-1} f_i = m_{10}/m_{00}, \tag{4.27}$$

$$\bar{y} = \sum_{i=0}^{N-1} y_i f_i \Big/ \sum_{i=0}^{N-1} f_i = m_{01}/m_{00}. \tag{4.28}$$

Here, u_{pq} is called the $(p+q)$th order central moment of shape f and is invariant of the position of the shape. The value of u_{pq} still varies with respect to the orientation of shape f. The following set of second-order and third-order moments are invariant of the orientation of a shape [198].

$$a_1 = u_{20} + u_{02} \tag{4.29}$$

$$a_2 = (u_{20} - u_{02})^2 + 4u_{11}^2 \tag{4.30}$$

$$a_3 = (u_{30} - 3u_{12})^2 + (3u_{21} + u_{03})^2 \tag{4.31}$$

$$a_4 = (u_{30} + u_{12})^2 + (u_{21} + u_{03})^2 \tag{4.32}$$

$$\begin{aligned} a_5 = {} & (u_{30} - 3u_{12})(u_{30} + u_{12})[(u_{30} + u_{12})^2 - 3(u_{21} + u_{03})^2] \\ & + (3u_{21} - u_{03})(u_{21} + u_{03})[3(u_{30} + u_{12})^2 - (u_{21} + u_{03})^2] \end{aligned} \tag{4.33}$$

$$\begin{aligned} a_6 = {} & (u_{20} - u_{02})[(u_{30} + u_{12})^2 - (u_{21} + u_{03})^2] \\ & + 4u_{11}(u_{30} + u_{12})(u_{21} + u_{03}) \end{aligned} \tag{4.34}$$

$$\begin{aligned} a_7 = {} & (3u_{21} - u_{03})(u_{30} + u_{12})[(u_{30} + u_{12})^2 - 3(u_{21} + u_{03})^2] \\ & - (u_{30} - 3u_{12})(u_{21} + u_{03})[3(u_{30} + u_{12})^2 - (u_{21} + u_{03})^2]. \end{aligned} \tag{4.35}$$

Usually more than one moment invariant is used in shape matching [102]. One way to measure the similarity between two shapes is to compute the distance between two vectors of moments representing the shapes. The smaller the distance, the more similar the shapes. This, however, requires that features in the vectors be of the same scale. Moments of different orders have different scales. The correlation of the logarithm of the moments has been used to measure this similarity [415]. Taking the logarithm of the moments does not completely remove the scale difference between different-order moments, and moments of higher orders always appear larger than moments of lower orders. The correlation of two vectors works well when the features forming each vector are of the same scale. When features of different scales are used, the feature with the largest scale will dominate the correlation value.

Garrett *et al.* [136] proposed the concept of pairing functions where the similarity between two sets of features is determined by quantizing the features and counting the number of quantized features that are equal. In this method, since features in a set are quantized into the same number of levels, it is as if the features are measured with the same scale. This alleviates the problem of scale difference between features but it causes the loss of some information in the quantization process. In a method outlined by Goshtasby [145], the image coordinates are mapped to values with average 1. To achieve this, x coordinates are replaced with $x/\bar{x}$ and y coordinates are replaced with $y/\bar{y}$, where $\bar{x}$ and $\bar{y}$ are the averages of the x and y coordinates of the shape points, respectively. This will keep all moments close to 1, making correlations of invariant moments more effective in shape matching.

4.3.1.3 Shape matrices Regions in two images to be registered often have translational, rotational, and scaling differences. Due to the local nature of images, if images do not represent a deformable scene, local nonlinear geometric difference between the images can often be ignored. To find correspondence between regions in two images, it is required to find the similarity between regions independent of their translational, rotational, and scaling differences. A shape matrix is a polar quantization of a shape as shown in Fig. 4.2. Consider a coordinate system with origin at the center of gravity of the shape and x-axis pointing from the center to the farthest point on the shape. If the shape is resampled at uniform angular and radial steps an array of numbers will be obtained that is independent of the position and orientation of the shape. If we let the radial increments be a function of the maximum radius of the shape, that is, always quantizing the maximum radius of the shape to the same number of intervals, the obtained representation will be independent of the scale of the shape. An array obtained in this manner is called a *shape matrix* [146].

Note that a shape matrix contains information about the boundary as well as the interior of a region. Therefore, it can represent regions with holes. Regions obtained by image segmentation methods often contain smaller regions. Shape matrices preserve such information although they standardize the information with respect to the

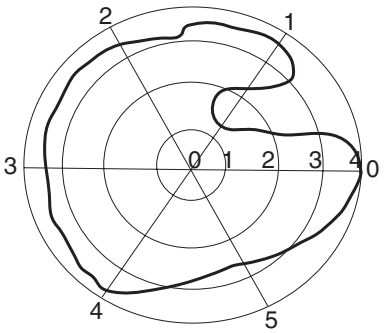


Fig. 4.2 (a) A shape and (b) its shape matrix.

position, orientation, and scale of the shape. Given two shape matrices M_1 and M_2 of size $m \times n$, because the matrices are binary, the similarity between them can be determined from

$$S = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \frac{1}{mn} \left[(M_1[i,j] \& M_2[i,j]) | (!M_1[i,j] \& !M_2[i,j]) \right], \tag{4.36}$$

where '&', '|', and '!' denote logical AND, OR, and NOT operations, respectively. When $S = 1$, the two shapes perfectly match and, as S decreases and approaches 0, the shapes become more dissimilar. If a shape is sampled finely when creating its shape matrix, the original shape can be reconstructed from the obtained shape matrix [146].

If a shape is quantized radially with logarithmic steps rather than with uniform steps, the difference between scales of two shapes will transform into translational difference horizontally in the log polar coordinate system and if, instead of starting from the maximum radius of a shape, we start from an arbitrary point on the shape when quantizing the shape angularly, a log polar quantization will be obtained that is translated vertically [44, 266, 405]. Log polar mapping transforms rotational and scaling differences between two shapes into translational differences, simplifying shape matching.

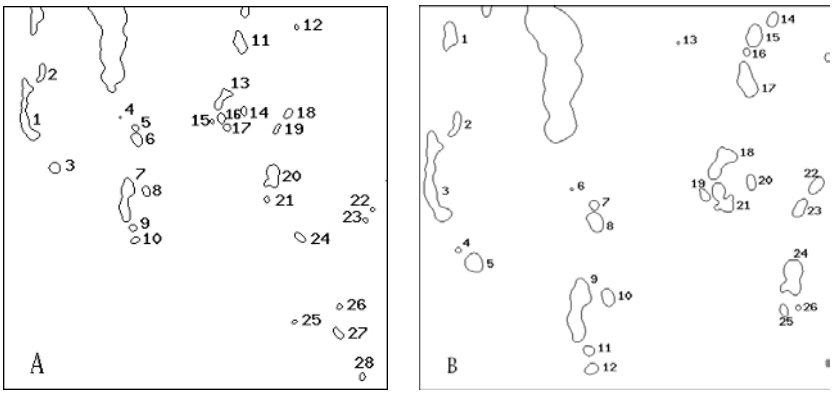
4.3.2 Region matching by relaxation labeling

Relaxation techniques have been around for quite some time in mathematics, but their use in computer vision and image analysis was popularized by Rosenfeld and colleagues [334]. Given a set of objects $\mathbf{A} = \{a_1, a_2, \ldots, a_m\}$, and a set of labels $\mathbf{B} = \{b_0, b_1, \ldots, b_n\}$, it is required to assign labels to the objects to best satisfy the world knowledge. Objects $a_1, a_2, \ldots, a_m$ can be considered regions in the sensed image and labels $b_1, b_2, \ldots, b_n$ can be considered regions in the reference image. By assigning labels to the objects, in a way, correspondence is established between the regions in the images. If a region in the sensed image does not have a correspondence in the reference image, label b_0 is assigned to it.

Labels are initially assigned to the objects using information about the shapes of regions. The labels are then iteratively examined and, if necessary, revised using information about distances between regions. Assuming $P_i(b_j)$ denotes the probability that object a_i has label b_j, the probabilities are initially computed using region shapes and are iteratively updated using

$$P_i^{k+1}(b_j) = \frac{P_i^k(b_j) \left[1 + q_i^k(b_j)\right]}{\sum_{j=0}^{n} P_i^k(b_j) \left[1 + q_i^k(b_j)\right]}, \quad i = 1, \ldots, m, \tag{4.37}$$

where $P_i^k(b_j)$ is the probability that object a_i has label b_j (region b_j corresponds to region a_i) at the kth iteration and $q_i^k(b_j)$ is the neighbor support for object a_i to have



(a) (b)

Fig. 4.3 (a) Regions in the reference image. They represent the labels. (b) Regions in the sensed image. They represent the objects. By assigning unique labels to the objects, correspondence is established between the regions in the images. Regions that are partially outside an image are not used in the labeling process.

label b_j in the kth iteration and is in the range $[-1, 1]$. The labeling probabilities at each iteration should satisfy

$$\sum_{j=0}^{n} P_i^k(b_j) = 1, \qquad i = 1, \ldots, m. \tag{4.38}$$

The iterative formula (4.37) requires that the initial probabilities, $P_i^0(b_j)$, and the initial neighbor contribution factors, $q_i^0(b_j)$, be known.

The initial probability that region a_i has label b_j is computed using the similarity between regions a_i and b_j. If the images have translational and rotational differences, we may use the cross-correlation coefficient of the magnitude of the Fourier transform coefficients as their similarity. If the regions have translational, rotational, and scaling differences, we may use the correlation of the invariant moments of the regions as the similarity measure, or use the shape matrices of the regions according to (4.36) to determine the initial probabilities. Therefore, if S_{ij} denotes the similarity between regions a_i and b_j, from the similarities we determine the initial probabilities as follows,

$$P_i^0(b_j) = \frac{S_{ij}}{\sum_{j=1}^{n} S_{ij}}. \tag{4.39}$$

When determining the initial probabilities, we assume all regions in the sensed image are present in the reference image, and therefore, the probability of an object having label b_0 is zero. As iterations proceed, if region a_i does not have a correspondence in the reference image, its probability of having label b_0 will gradually increase and ultimately converge to 1.

The neighbor contribution factors update the labels in such a way that the assignments become consistent with world knowledge. World knowledge can be relative sizes of the regions, distances of the regions from each other, and relative positions of the regions. We represent this knowledge in matrix form and call it knowledge matrix $V[\,]$. An entry of the matrix, $V[b, b']$, may show:

1. The ratio of the perimeter or area of regions b and b'.
2. The distance between regions b and b'.
3. Relative position of region b with respect to region b', for example, the angle of the line connecting the centroids of b and b' with the x-axis.
4. The shape similarity of regions b and b'.

To find the compatibility of label b_j of object a_i with labels of other objects, we continue as follows. Let $\mathbf{V}[b_j] = \{V[b_j, b_l] : l = 1, \ldots, n\}$ show the distances between region b_j and other regions in the reference image. We also determine the distances of a_i to other regions in the sensed image. Let $\mathbf{D}[a_i] = \{D[a_i, a_l] : l = 1, \ldots, m\}$ denote the distances. Every region in the sensed image has a set of labels, each with its own probability. Assuming $c_i \in \mathbf{B}$ is the label of a_i that has the largest probability, we then determine another set of distances $\mathbf{U}^k[b_j] = \{V[b_j, c_i] : i = 1, \ldots, m\}$, where $V[b_j, c_i]$ denotes the distance between b_j and c_i in the reference image. Then the cross-correlation between $\mathbf{D}[a_i]$ and $\mathbf{U}^k[b_j]$,

$$q_i^k(b_j) = \frac{E\{\mathbf{U}^k[b_j]\mathbf{D}[a_i]\} - E\{\mathbf{U}^k[b_j]\}E\{\mathbf{D}[a_i]\}}{\sigma\{\mathbf{U}^k[b_j]\}\sigma\{\mathbf{D}[a_i]\}}, \tag{4.40}$$

shows the neighbor support for object a_i having label b_j at the kth iteration, where

$$E\{\mathbf{U}^k[b_j]\} = \frac{1}{m}\sum_{i=1}^{m} V[b_j, c_i], \tag{4.41}$$

$$E\{\mathbf{D}[a_i]\} = \frac{1}{m}\sum_{l=1}^{m} D[a_i, a_l], \tag{4.42}$$

$$E\{\mathbf{U}^k[b_j]\mathbf{D}[a_i]\} = \frac{1}{m}\sum_{l=1}^{m} \{V[b_j, c_l]D[a_i, a_l]\}, \tag{4.43}$$

$$\sigma\{\mathbf{U}^k[b_j]\} = \left\{\frac{1}{m}\sum_{i=1}^{m} \left\{U^k[b_j, c_i] - E\left[\mathbf{U}^k[b_j]\right]\right\}^2\right\}^{1/2}, \tag{4.44}$$

$$\sigma\{\mathbf{D}[a_i]\} = \left\{\frac{1}{m}\sum_{l=1}^{m} \{D[a_i, a_l] - E\left[\mathbf{D}[a_i]\right]\}^2\right\}^{1/2}. \tag{4.45}$$

Here, $q_i^k(b_j)$ is a measure in the range $[-1, 1]$ and, if region a_i truly corresponds to region b_j, then, regardless of some mistakes in the labels of other regions, $q_i^k(b_j)$

will be high. On the other hand, if region a_i truly does not correspond to region b_j then, even though all other regions in the sensed image are labeled correctly, the value of $q_i^k(b_j)$ will be low. This is a very desirable property and we will use $q_i^k(b_j)$ as the neighbor contribution factor in (4.37).

The neighbor contribution factors for the undefined region labels $q_i^k(b_0)$ cannot be determined in this manner because correlation of distances that do not exist cannot be computed and so $\mathbf{V}[b_0]$ is undefined. To determine the neighbor support for object a_i having label b_0 (the neighbor support for region a_i in the sensed image having no correspondence in the reference image), we observe that if object a_i truly has label b_0, then assigning label b_0 to it should increase the label probabilities of other objects. If b_0 is not the true label of a_i, assigning b_0 to it will decrease the label probabilities of the other objects. Assuming that most of the highest-probability labels of the objects show the true labels of the objects, we, first, determine the highest probability label of each object when b_0 is assigned to a_i (except when the largest probability label of an object is already b_0) and, second, determine the highest probability label of objects by assigning the highest probability label of a_i to a_i (again, except when the highest probability label of the object is b_0). Then, if there are M objects where their highest label probabilities increase in case 1 and there are N objects where their highest label probabilities increase in case 2, we:

- increase the label probability for object a_i having label b_0 if $M > N$;
- decrease the label probability for object a_i having label b_0 if $M < N$;
- leave the label probability for object a_i having label b_0 unchanged, if $M = N$.

If T is the number of objects whose highest probability labels are not b_0, then $(M - N)/T$ shows the degree of neighbor support for object a_i having label b_j and we will use $(M - N)/T$ as the neighbor contribution factor, $q_i^k(b_0)$. This measure also shows that if, by assigning label b_0 to a_i, the highest probability for all object labels increases, then $q_i^k(b_0) = 1$, while if the highest probability of all object labels decreases, then $q_i^k(b_0) = -1$. This measure characterizes the neighbor contribution factor for an object having the undefined-region label well. In the above, distances between regions were used as the world knowledge to estimate the neighbor contribution factors. Other information, such as relative sizes of regions or relative positions of regions, can also be used for this purpose.

For images with translational, rotational, and scaling differences we should select a knowledge matrix that makes the neighbor contribution factors $q_i^k(b_j)$ independent of the translational, rotational, and scaling differences between images. For instance, the angles of lines connecting region pairs in an image with the x-axis are not invariant with respect to the orientation of the image, but the ratio of distances between regions is independent of the position, orientation, and scale of images.

Once correspondence is established between regions in two images by any of the above methods, the similarity between corresponding regions can be increased through a region refinement process. This refinement will produce very similar regions with centroids that correspond to each other with subpixel accuracy. The capability to determine corresponding region centroids with subpixel accuracy makes it

possible to determine a transformation that can register the images with subpixel accuracy. Coupling image segmentation and image registration has proven very effective in image registration [162, 164].

4.4 CHAMFER MATCHING

Chamfer matching is the process of finding the position of an object (subimage) in an image where both the subimage and the image are binary. The sum of distances between corresponding object and image points is used as the match-rating. When an object is an entire image, the process will find the amount of shift that is needed to align the two images. The amount of shift where the images best align is determined using the sum of distances between closest points in the images. Barrow *et al.* [25] first used this idea to find the model of a coastline in a segmented aerial image. The idea has since been widely used in image matching [41, 246].

Chamfer matching can be used to determine the positions of regions in the sensed image with respect to the reference image when local geometric differences between the images are small, although global geometric difference between the images can be large. Given a template and an image, the process of locating the template within the image involves shifting the template within the image and at each shift position determining the sum of distances of closest object points in the template and the image. The smaller the sum, the closer the template is considered to be from the true match position in the image. The process, therefore, involves starting from an initial position in the image, determining the sum of distances of points in the template to points in the image closest to them and shifting the template in the image in the gradient-descent direction of the sum until a minimum is reached in the sum. If the template perfectly matches the image at a shift position, the sum of distances obtained as a result will be zero. Due to noise and segmentation errors, however, this rarely occurs even when a perfect match is possible. The smaller the sum of distances, the closer the template is considered to be to its true position in the image.

Depending on the starting position of the template in the image and the contents of the image, the process may converge to a local minimum rather than a global one and miss the true match position. If an exhaustive search can be afforded, it should be taken rather than the gradient-descent approach. Since most of the computation time is spent on finding the sum of distances at various shift positions, considerable effort has gone into finding ways to speed up the computation of distances.

If the distances are computed ahead of time and reused when calculating the sum of distances, considerable computational savings can be achieved. Given a binary (edge) image, first the *distance transform* of the image is computed. The value at an entry in the distance transform image shows the distance of that entry to the edge point closest to it. Once the distance transform of an image is determined, instead of shifting the region within the image, the region is shifted within the distance transform image and at each shift position the sum of entries in the distance transform image corresponding to the region edges is computed. This can be achieved very quickly, requiring only n additions for a region containing n edge points. Each shift position, therefore,

involves n additions. If the distance transform is not used in the calculations, for each region point the edge point closest to it in the image should be found, which would require N comparisons if N edge points are present in the image. Therefore, computation time reduces from nN when direct computation is used to n when the distance transform is used.

4.4.1 Distance transform

Computation of the distance transform of an image goes as far back as 1966 [333]. Given a binary image with 0 and 1 values, 0 showing the object points and 1 showing the background points, the distance of a background point $[i, j]$ to the closest object point in the image is computed and saved at $[i, j]$ in the distance transform image. The objective in finding the distances in most applications is to determine the object point closest to a given point in an image. For that reason, actual Euclidean distances are not necessary and any distance measure that provides this property can be used. Rosenfeld and Pfaltz [333] used city-block distances and suggested the following two-step algorithm to compute the distance transform $f[\,]$ of binary image $a[\,]$ of size $m \times n$ pixels.

1. In forward raster sequence ($i = 0, \ldots, m-1$; $j = 0, \ldots, n-1$):
   (a) set $f[i, j]$ to 0 if $a[i, j] = 0$;
   (b) set $f[i, j]$ to $\min(a[i-1, j] + 1, a[i, j-1] + 1)$ if $a[i, j] = 1$ and $[i, j] \neq [0, 0]$;
   (c) set $f[i, j]$ to $m + n$ if $a[i, j] = 1$ and $[i, j] = [0, 0]$.
2. In reverse raster sequence ($i = m-1, \ldots, 0$; $j = n-1, \ldots, 0$), set $f[i, j]$ to $\min(a[i, j], a[i+1, j] + 1, a[i, j+1] + 1)$.

Since no two points in an image can be at a distance greater than $m + n$, all background points in the image have distances less than $m + n$ from an object point. This algorithm uses the city block distance, which is not invariant of an image's orientation. That is, rotating an image will change the computed distances. Nevertheless, the computed distances are sufficient in decision making in many applications.

To obtain distances that do not depend on the orientation of an image, the Euclidean distance or a measure that monotonically increases with increasing Euclidean distances is needed. Algorithms that approximately produce Euclidean distances are given by Montanari [276], Danielsson [83], and Borgefors [40]. The algorithm proposed by Borgefors starts with image $a[\,]$ where object pixels are marked with 0 and background pixels are marked with a very large number and proceeds as follows.

1. In forward raster sequence ($i = 0, \ldots, m-1$; $j = 0, \ldots, n-1$),
   set $a[i, j] = \min_{k,l}(a[i + k, j + l] + d)$,
   where $k = -1, \ldots, 1$; $l = k, \ldots, 1$.
2. In reverse raster sequence ($i = m-1, \ldots, 0$; $j = n-1, \ldots, 0$),
   set $a[i, j] = \min_{k,l}(a[i + k, j + l] + d)$,
   where $k = 1, \ldots, -1$; $l = k, \ldots, -1$,

where $d = 2$ if $|k| + |l| = 1$ and $d = 3$ if $|k| + |l| = 2$. In both forward and reverse scans, a 3×3 window is centered at each pixel $[i, j]$ and d is added to values in the upper half or the lower half of the window entries in the image, selecting the smallest value and saving at $[i, j]$. If $|k| + |l| = 1$, the selected value is to the left, to the right, above, or below $[i, j]$ and, therefore, its distance to $[i, j]$ is 1. If $|k| + |l| = 2$, the selected value is at an entry diagonally positioned with respect to $[i, j]$. Therefore, its distance to $[i, j]$ is $\sqrt{2}$. Since integer distances are used, the actual distances are multiplied by two and rounded, that is, when $|k| + |l| = 1$, d will be 2 and when $|k| + |l| = 2$, d will be 3. In-place computation is performed to propagate the distance values first from top to bottom and then from bottom to top.

Approximation errors caused by rounding the distances and adding them together iteratively could accumulate and become large. An example of the distance transform computed by this algorithm for an image containing a single point at its center is shown in Fig. 4.4a. Isovalued distances at 10-unit intervals are shown in Fig. 4.4b. The transform is not circularly symmetric and, therefore, the distances are not invariant of the orientation of the image. In many applications, these approximation errors may not be large enough to affect the decision-making process. The fact that computed distances are not independent of an image's orientation means that reliability in chamfer matching will depend on the orientation of the template and/or image. Exact Euclidean distances, as shown in Fig. 4.4c and 4.4d, are rotationally invariant.

An algorithm developed by Cuisenaire and Macq [79] computes exact Euclidean distances by first computing an approximate distance transform and then iteratively improving the approximation. Shih and Wu [355] improve the computational speed of this algorithm by proposing a two-pass process without any iterations. Both algorithms have computational complexity mn for an image of size $m \times n$, although the latter has a smaller coefficient than the former.

Exact Euclidean distances are invariant of an image's orientation and are desired in chamfer matching. In the following, an algorithm for the exact calculation of Euclidean distances is described with computational complexity that is comparable to the method of Cuisenaire and Macq [79] but, in addition to computing the distances of background pixels to the object pixels, it constructs a pointer list showing a pointer

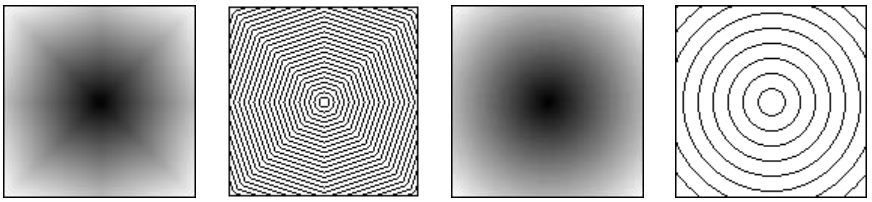


Fig. 4.4 (a) Distance transform of an image containing a single object point at its center by the method of Borgefors [40]. (b) Isovalued contours of the distance transform image with 10-unit intervals. (c) True Euclidean distances of the image with a single object point at the center of the image. Displayed intensities show scaled distances for better viewing. (d) Isovalued contours of the Euclidean distance transform with 10-unit intervals.

from each background pixel to the object pixel closest to it. Therefore, once the best-match position of an object in a binary image is determined, the correspondence between image and object pixels will be immediately known. This correspondence is needed to determine the transformation function that registers the images.

Suppose an image $a[\,]$ of type integer and size $m \times n$ pixels is given. An image $f[\,]$ of type float and of the same size is created with entries ultimately showing the exact Euclidean distances of background pixels to the object pixels closest to them in $a[\,]$. The algorithm revises image $a[\,]$ such that an entry will show the label of the object pixel closest to it. The details of the algorithm follow.

Algorithm 4.5: Given binary image $a[\,]$ of size $m \times n$, this algorithm calculates the Euclidean distance transform of the image and saves in $f[\,]$. It also changes image $a[\,]$ such that an entry will show the label of the object point closest to it. It is assumed that there are N object pixels in the image. R and C are two lists of size N keeping the row and column numbers of the object pixels in $a[\,]$.

1: Scan image $a[\ ]$ and save unique labels 1 through N at the object pixels and save 0 at the background pixels. Also, create lists R and C of size N and save the row and column numbers of an object pixel with label l at R_l and C_l.

2: Create image $f[\ ]$ of size $m \times n$ and let $f[i,j] = 0$ if $a[i,j] \neq 0$; for $i = 0, \ldots, m-1$ and $j = 0, \ldots, n-1$.

3: For $i = 0, \ldots, m-1$ and $j = 0, \ldots, n-1$, if $a[i,j] = 0$ and there are nonzero entries adjacent to $a[i,j]$, for each such entry $[i', j']$, let $l = a[i', j']$ and compute distance of $[i,j]$ to $[R_l, C_l]$ and let the label of the neighbor with the smallest distance be k and the smallest distance be d_k. Then, let $f[i,j] = d_k$ and $a[i,j] = k$.

4: Repeat step 3 until no more zeros remain in image $a[\ ]$.

Step 1 assigns labels to the object points and enters the coordinates of the object points into lists R and C, making a link between the coordinates of the object points and their labels. Step 2 creates the distance transform image and initializes the entries at the object pixels to 0. At this point, the object pixels in image $a[\,]$ have nonzero labels and the background pixels have value 0. The nonzero pixels in image $a[\,]$ are dialated by one pixel at step 3 and the process is repeated in step 4 until the labels propagate to all pixels in image $a[\,]$. Upon completion of Algorithm 4.5, $a[i,j]$ will show the label of the object pixel closest to it and $f[i,j]$ will show the distance of the object pixel in image $a[\,]$ closest to entry $[i,j]$ in the image.

Algorithm 4.5 finds the exact Euclidean distance between each image pixel $[i,j]$ and the object pixel closest to it. First, it finds the exact Euclidean distance because all distances between $[i,j]$ and an object pixel computed in step 3 are Euclidean distances. Second, it finds the distance between $[i,j]$ and the object pixel closest to it because if a closer object pixel was present, it would have propagated to it earlier. Since that did not happen, the current distance should be the shortest. At each iteration

in step 4, distances of farther away background pixels to object pixels closest to them are determined. *For the propagation to proceed symmetrically, in-place computation should be avoided. That is, the distances computed in step 3 should be saved in a buffer and upon completion of the iteration they should be copied to the image. Entering the calculated distances to the image gradually may result in asymmetric propagation of the distances and may result in inaccurate distances.*

Chamfer matching using the distance transform image is sensitive to noise, especially when searching for the best-match position of an object in an image using the steepest descent algorithm. Noisy object pixels considerably change the contents of a distance transform image and can make the process converge to a local minimum rather than the global minimum. Figure 4.5a shows a circle, Fig. 4.5b shows the same circle with five additional random points, and Figs 4.5c and 4.5d show the corresponding distance transforms. As can be observed, a small number of noisy object points could drastically change a distance transform image, and potentially change the matching result.

In chamfer matching, the exact Euclidean distances are not needed and as long as the distances are proportional to Euclidean distances and are rotationally invariant, the same matching results will be obtained. If we convolve an image with a monotonically increasing radial function, an image will be obtained that functions like a distance transform image. The inverse of a Gaussian may be used as the monotonically increasing radial function. Therefore, to obtain the distance transform of an image, the image is convolved with a Gaussian and the intensities of the convolved image are inverted. Computation of the distance transform in this manner makes the obtained distances less sensitive to noise. This is demonstrated in an example in Fig. 4.6. Figures 4.6a and 4.6b show distance transforms of images 4.5a and 4.5b, respectively, computed by Gaussian convolution. Compared to the Euclidean distance transform, the distance transform computed by Gaussian convolution is less sensitive to noise. The isovalued distances of Fig. 4.6b are shown in Figs 4.6c. In contrast, the isovalued distances of image 4.5d as shown in Fig. 4.6d, are greatly influenced by noise, destroying monotonicity in distance values. Noise has very little effect on the distance transform computed by Gaussian convolution, implying that the process will less likely produce mismatches compared to when the Euclidean

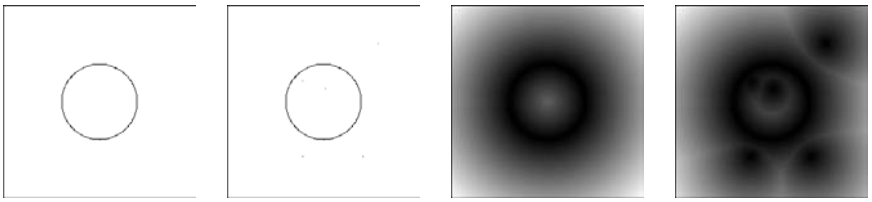


Fig. 4.5 (a) An image containing a circle. (b) Same as (a) but with five added randomly points. (c), (d) Exact Euclidean distance transforms of (a) and (b), respectively.

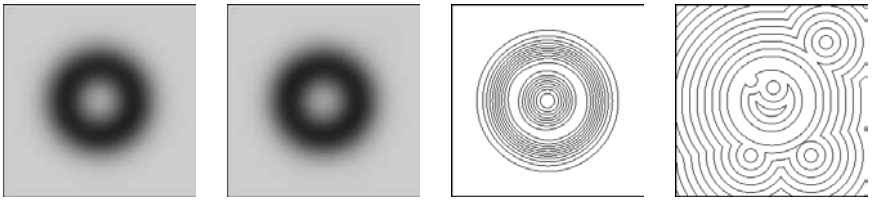


Fig. 4.6 (a), (b) Distance transforms of images 4.5a and 4.5b by Gaussian convolution. (c), (d) Isovalued distances of 4.6b and 4.5d, respectively.

distance transform is used. The Gaussian convolution in effect smoothes noise in an image while calculating its distance transform.

The computational complexity of the distance transform when computed by Gaussian convolution is somewhat higher than that computed by Borgefors' method [40]. Since a 2-D Gaussian convolution can be implemented by a 1-D Gaussian convolution row-by-row and then column-by-column, computation of the distance value at an image point requires $m + n$ additions and multiplications, making the computational complexity of Gaussian convolution in the order of $mn(m + n)$, where m and n are the dimensions of the image. This computation can be reduced to $mn(\log m + \log n)$ if the FFT algorithm [46] is used.

The standard deviation of the Gaussian used to calculate the distance transform of an image depends on the noise level in the image. The smaller the standard deviation of the Gaussian, the smaller will be the smoothing effect, retaining some of the noisy details in the obtained transform. The larger the standard deviation of the Gaussian, the smaller will be the influence of noise on the computed distance transform, but this also may smooth the image details that are essential in finding the best-match position accurately. A very large standard deviation of the Gaussian may shift the best-match position from its true position, with the amount of shift depending on the magnitude of the standard deviation. The larger the standard deviation, the smaller will be the likelihood of the matching process converging to a local minimum during a steepest descent search and missing the correct match, but the larger will be the amount of shift between the best-match position and the correct match position. The standard deviation should be selected by making a compromise between the localization accuracy and the likelihood of obtaining a mismatch.

Chamfer matching in 3-D closely follows that in 2-D except that the search is made in 3-D for the best-match position using 3-D distance transforms. Computation of distance transform in 3-D also follows that in 2-D. The algorithm of Rosenfeld and Pfaltz [333] in 3-D is virtually the same as in 2-D except that 3-D instead of 2-D city-block distances are used. In the 3-D version of the Borgefors' algorithm [42], distances at the $3 \times 3 \times 3$ neighborhood centered at each image point are examined. The smallest distance is selected and 3, 4, or 5 is added to it and saved at the point under consideration depending on whether the sum of absolute differences between the coordinates of the point under consideration and the coordinates of the neighbor

with the smallest distance is 1, $\sqrt{2}$, or $\sqrt{3}$. Here, the actual distances are multiplied by 3 and rounded to obtain d. The 3-D version of Algorithm 4.5 will dialate the distances in 3-D symmetrically starting from distance 0 at object points. Distance transform by Gaussian convolution in 3-D involves 3-D smoothing as opposed to 2-D smoothing.

Variants of chamfer matching are the iterative head-and-hat matching [240, 300] and the iterative closest point (ICP) matching [31, 114]. These algorithms iteratively change the rotation and translation of one image with respect to the other in the gradient descent direction until the two images best match according to a distance measure.

4.5 TEMPLATE MATCHING

Template matching is the process of locating a template in an image. The template can be considered a subimage from the reference image, and the image can be considered the sensed image. By locating templates from the reference image in the sensed image, the objective is to establish correspondence between the reference and sensed image points. The template-matching process involves shifting the template in the sensed image and at each shift position determining the similarity between the template and the window matching it in the sensed image and determining the shift position where the similarity between the template and the window is maximum. It is assumed that the reference and sensed images have only translational differences and, thus, scaling, rotation, and local geometric differences between the images are negligible.

Different metrics have been used to determine the similarity between a template and a window of the same size. The metrics compare different properties of the template and the window to determine their similarity. Most widely used similarity measures are described below.

4.5.1 Similarity measures

Various similarity measures have been used in template matching. They are, sum of absolute differences [23], cross-correlation coefficient [315], moments [5], Fourier transform coefficients [9, 53, 123], Mellin transform coefficients [54], Haar transform coefficients [58], Walsh-Hadamard transform coefficients [351], K-S test [108], and, most recently, mutual information [253, 399].

4.5.1.1 Similarity using raw image intensities This includes the sum of absolute differences and the cross-correlation coefficient. The sum of absolute differences is the Minkowski metric of order one [90] and is defined by

$$D[x,y] = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} |f_t[i,j] - f_w[i+x, j+y]|\,, \tag{4.46}$$

where $x = 0, \ldots, M-m+1$ and $y = 0, \ldots, N-n+1$ when template is of size $m \times n$ pixels, image is of size $M \times N$ pixels, $f_t[\,]$ shows the template, and $f_w[\,]$ is the

window in the sensed image being matched with the template. Position $[x, y]$ shows the position of the template in the sensed image. By changing x and y, the template is shifted in the image and at each shift position the sum of absolute differences between corresponding pixels in the template and the window is determined. This metric actually measures the dissimilarity between the template and the window. The smaller the D, the more similar will be the template and the window. A study carried out by Svedlow *et al.* [374] found that instead of raw image intensities, gradient of intensities produce a more reliable matching.

At each shift position mn additions and subtractions are needed; therefore, the computational complexity of sum of absolute differences is mn. A sequential similarity measure proposed by Barnea and Silverman [23] reduces the computation time by stopping the process when the sum reaches a value larger than a previously reached value since the search is for the smallest sum. To speed up the search, Dewdney [92] used a steepest descent approach to guide the search for the best-match position without searching the entire image. To speed up the template matching process, Vanderburg and Rosenfeld [394] used a two-stage process where a part of the template was first used to find the candidate match positions and then the entire template was used to find the best-match position among the candidates. These methods increase speed at the cost of increasing the mismatch probability.

The cross-correlation coefficient is a distance metric of order two and is defined by

$$S[x,y] = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} g_t[i,j]g_w[i+x,j+y]}{\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} g_t^2[i,j]\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} g_w^2[i+x,j+y]\right\}^{1/2}}, \quad (4.47)$$

where $g_t[i,j] = f_t[i,j] - \bar{f}_t$, $g_w[i,j] = f_w[i,j] - \bar{f}_w$, and $\bar{f}_t$ and $\bar{f}_w$ are the template and window averages, respectively. The closer the intensities in the window vary with those in the template, the higher the value for S will be. The cross-correlation coefficient varies between -1 and 1. The larger the coefficient, the more similar the template and the window will be.

Computation of the cross-correlation coefficient involves on the order of mn additions and multiplication, but the coefficient of mn is much larger than that for the sum of absolute differences. Note that as $[x, y]$ is changed, the numerator of (4.47) becomes a convolution operation and fast Fourier transform may be used to speed up the computation [9]. Speed up is also achievable by using a two-step search [159], first a subset of pixels in the template is used to find the candidate match positions and then the entire template is used to find the best-match position from among the candidates.

4.5.1.2 Similarity using spatial properties

Similarity between two images can be measured using their spatial-frequency characteristics. Spatial frequencies in an image can be characterized using the Fourier transform, the Hadamard transform, the Haar transform, or other transform coefficients. Low-order transform coefficients measure low-frequency contents in an image and high-order coefficients reflect high-spatial frequencies present in an image. By comparing the transform coefficients in

two images, we can determine the similarity between spatial-frequency characteristics of the images. Two images, however, may have very different geometric layouts but still have very similar spatial frequency characteristics. Correlation of the transform coefficients from two images may be used to measure the similarity between the spatial frequency characteristics of the images. If the Fourier transform is used, rather than finding the correlation of the magnitude of the coefficients, the correlation of both real and imaginary parts of the coefficients should be used. Using the magnitudes alone will result in loss of the phase information that the coefficients carry. It has been shown that the phase in the Fourier transform carries substantial information that represents significant details in an image [297]. Images with translational differences have been registered using this similarity measure [9, 123].

4.5.1.3 Similarity using geometric properties

Metrics that capture the geometric layout of intensities in a template or window are moments and major-axis parameters. The $(p+q)$th order moment of template $f[\,]$ of size $m \times n$ is defined by [198]

$$M_{pq} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} i^p j^q f[i,j]. \tag{4.48}$$

If two images do not have scaling and rotational differences, the geometric similarity between them can be measured using the correlation of their moments. Moments from order 0 to 3 may be used to determine this similarity. Higher order moments measure small variations in an image and are greatly influenced by noise.

The major axis or the axis of minimum inertia of a nonsymmetric template $f[\,]$ of size $m \times n$ is a unique axis defined by $j = ai + b$, where i and j denote image coordinates horizontally and vertically and a and b are computed from [299]

$$a = \frac{\sum_i \sum_j ijf[i,j] \sum_i \sum_j f[i,j] - \sum_i \sum_j if[i,j] \sum_i \sum_j jf[i,j]}{\sum_i \sum_j i^2 f[i,j] \sum_i \sum_j f[i,j] - \left[\sum_i \sum_j f[i,j]\right]^2}, \tag{4.49}$$

$$b = \frac{\sum_i \sum_j jf[i,j] - a \sum_i \sum_j if[i,j]}{\sum_i \sum_j f[i,j]}, \tag{4.50}$$

where i varies from 0 to $m-1$ and j varies from 0 to $n-1$.

When the major axis is nearly vertical, parameters a and b may involve large errors. If $a > 1$, the process should be repeated after switching the row and column numbers of the template. In that case, the equation of the major axis of the template will be $i = aj + b$. Alternatively, the polar equation of the line

$$\rho = i \cos\theta + j \sin\theta \tag{4.51}$$

may be used. In that case, parameters ρ and θ of the axis of minimum inertia are obtained from [207]

$$\theta = 0.5 \tan^{-1}\left(\frac{b}{a-c}\right), \tag{4.52}$$

$$\rho = (\bar{i}\cos\theta + \bar{j}\sin\theta), \tag{4.53}$$

where

$$\bar{i} = \frac{1}{\bar{f}}\sum_i\sum_j i f[i,j], \tag{4.54}$$

$$\bar{j} = \frac{1}{\bar{f}}\sum_i\sum_j j f[i,j], \tag{4.55}$$

$$a = \sum_i\sum_j (i-\bar{i})^2 f[i,j], \tag{4.56}$$

$$b = 2\sum_i\sum_j (i-\bar{i})(j-\bar{j}) f[i,j], \tag{4.57}$$

$$c = \sum_i\sum_j (j-\bar{j})^2 f[i,j], \tag{4.58}$$

and $\bar{f} = \sum_i \sum_j f[i,j]$. Parameters ρ and θ are obtained by finding the weighted sum of squared distances of points in a template to its major axis and minimizing the obtained sum, the weights being the intensities of the pixels.

The major axis of template $f[\,]$ depends on the geometric layout of intensities in the template. Similarity between a template and a window can be measured by the angle between the major axes of the template and the window. This can be used to quickly filter out the impossible matches and keep record of the possible matches. Other measures, such as cross-correlation coefficient, may then be used to determine the best match among the possible ones. Major axis parameters work well when the template contains a nonsymmetric pattern. The computational complexity of this metric is on order of mn multiplications and additions.

4.5.1.4 Similarity using algebraic properties

Any $n \times n$ image $\mathbf{f}$ may be represented in a space defined by orthonormal matrices $\mathbf{U}$ and $\mathbf{V}$ by [7]

$$\mathbf{f} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^t, \tag{4.59}$$

where t denotes transpose and $\boldsymbol{\Sigma}$ is a diagonal matrix whose elements are the singular values of $\mathbf{f}$ (that is, $\boldsymbol{\Sigma} = diag(\sigma_0, \sigma_1, \ldots, \sigma_{n-1})$). $\mathbf{U}$ and $\mathbf{V}$ are, correspondingly, the row eigenvector system and the column eigenvector system of $\mathbf{f}$. That is,

$$\mathbf{U} = [\mathbf{u}_0 \;\; \mathbf{u}_1 \;\; \ldots \;\; \mathbf{u}_{n-1}], \tag{4.60}$$

$$\mathbf{V} = [\mathbf{v}_0 \;\; \mathbf{v}_1 \;\; \ldots \;\; \mathbf{v}_{n-1}], \tag{4.61}$$

where $\mathbf{u}_i$ is the ith row eigenvector and $\mathbf{v}_i$ is the ith column eigenvector of $\mathbf{f}$. Therefore, relation (4.59) may be written as

$$\mathbf{f} = [\mathbf{u}_0 \ \mathbf{u}_1 \ \ldots \mathbf{u}_{n-1}] \, \mathbf{\Sigma} \, [\mathbf{v}_0 \ \mathbf{v}_1 \ \ldots \ \mathbf{v}_{n-1}]^t \tag{4.62}$$

or

$$\mathbf{f} = \sum_{i=0}^{n-1} \sigma_i \mathbf{u}_i \mathbf{v}_i^t. \tag{4.63}$$

Note that (4.63) shows $\mathbf{f}$ as a summation of n orthogonal matrices $\mathbf{u}_i \mathbf{v}_i^t$. If $\mathbf{f}$ is not singular, there would be n orthogonal matrices. If the rank of $\mathbf{f}$ is k, k of the n singular values of $\mathbf{f}$ will be nonzero and that implies that image $\mathbf{f}$ will be composed of k orthogonal matrices. Therefore, the singular values of an image are good indicators of the number of orthogonal components the image carries. If all singular values of an image are large, it implies that the image is composed of n strong orthogonal components. If some of the singular values are small, the corresponding orthogonal components are weak and may not contribute to template matching. The similarity between a template and a window can be computed from the correlation of the singular values of the two. The computation time to find the singular values of an $n \times n$ image is proportional to n^3. Among the similarity measures discussed so far, singular-value decomposition is the most expensive one.

4.5.1.5 Similarity using statistical properties

If random samples are taken from a template and a window, a test that determines whether the samples come from the same distribution, and thus represent the same scene parts, is the Kolmogorov-Smirnov (K-S) test. In the K-S test, the empirical distribution functions of random samples from two images are compared to determine the similarity between the images. Similarity is measured by the maximum distance between the empirical distribution functions of random samples from the images [73]. The smaller this maximum, the more likely it will be that the two samples came from the same distribution. The assumptions are that the samples are taken randomly from unknown distributions and independence holds within and between samples. When an image is homogeneous, the assumption of independence within samples will be violated but if an image contains high intensity variations and the sample points are taken widely spread, then the samples may be considered random.

In the K-S test, the null hypothesis is that the two samples come from the same distribution and the alternative hypothesis is that the two samples do not belong to the same distribution. In this test, the decision rule is as follows. Let $S_1(I)$ and $S_2(I)$ be the empirical distributions of samples from a template and a window, respectively, where I shows that the intensities of the pixels are used to determine the empirical distribution functions. Suppose T denotes the greatest vertical distance between the two distribution functions,

$$T = Sup_I \left| S_1(I) - S_2(I) \right|, \tag{4.64}$$

then, the null hypothesis is rejected at a level of significance α if T exceeds its $1-\alpha$ quantile. The quantiles are computed and tabulated in statistical textbooks (see for example Table A20 in [73]).

In template matching using the K-S test, the template from the reference image is shifted in the sensed image and at each shift position the similarity between samples from the template and the corresponding window in the sensed image is computed using test statistic T. The shift position which produces the smallest test statistic is taken to be the best-match position of the template in the image. The best-match position is taken to be the correct match with 95% confidence if the corresponding test statistics falls below its 0.95 quantile.

The computational complexity of K-S test involves the time to order the points, which involves $N \log N$ comparisons when sample size is N, and the time to determine the maximum difference between the empirical distributions, which is in average about $2N$ comparisons. Overall, computational complexity of the method is $N \log N$.

4.5.1.6 Similarity using mutual information

To determine the similarity between template $f_t[\,]$ and window $f_w[\,]$ of size $m \times n$ pixels, assuming $f_t[\,]$ and $f_w[\,]$ are random variables and $P_t(a)$ is the probability that the intensity at a pixel in $f_t[\,]$ is a and $P_w(b)$ is the probability that the intensity at a pixel in $f_w[\,]$ is b, if we overlay the template and the window, the probability that intensity a in the template lies on top of intensity b in the window will be equal to their joint probability, $P_{tw}(a,b)$. If the template and the window truly correspond to each other, their intensities will be highly dependent and they will produce high joint probabilities. However, if the template and the window do not correspond to each other, they will produce small joint probabilities. If intensities in the template and the window are completely independent, the joint probabilities will be $P_t(a)P_w(b)$. Given these probabilities, mutual information is computed from [253, 392, 399]

$$I(t,w) = \sum_{a=0}^{255}\sum_{b=0}^{255} P_{tw}(a,b)\log\frac{P_{tw}(a,b)}{P_t(a)P_w(b)}. \tag{4.65}$$

The joint probabilities $P_{tw}(a,b)$ for different values of a and b can be estimated from the histogram of intensities of corresponding pixels in the matching template and window. To obtain the histogram, a 256×256 array is allocated and all its entries are initialized to zero. It is assumed that intensities in the template and the window vary between 0 and 255. If at a particular pixel the template shows intensity a and the window shows intensity b, entry (a,b) in the array is incremented by one. After processing all pixels in the template and the window, an array will be obtained whose entry (a,b) shows the number of pixels in the template with intensity a where corresponding positions in the window have intensity b. To obtain the probabilities, contents of the histogram array are divided by the sum of the entries. Note that the sum of the entries is mn if the template and the window are $m \times n$. Entries of the obtained array correspond to the values of $P_{tw}(a,b)$ for different values of a and b.

Assuming intensity a represents the column numbers and intensity b represents the row numbers in $P_{tw}(a, b)$, $P_t(a)$ and $P_w(b)$ can be estimated from

$$P_t(a) = \sum_{b=0}^{255} P_{tw}(a, b) \tag{4.66}$$

and

$$P_w(b) = \sum_{a=0}^{255} P_{tw}(a, b), \tag{4.67}$$

respectively. If a unique mapping exists between intensities in the images, a high mutual information will be obtained. Compared to cross-correlation coefficient and sum of absolute differences, mutual information is more sensitive to image noise. Intensity a in the template should always correspond to intensity b in the window to produce a high similarity. This shows that images from different sensors can be registered using mutual information as long as a unique mapping exists between intensities in the images. Noise in one or both images will quickly degrade the similarity measure. A study carried out by Penney *et al.* [302] in registration of same modality images found that mutual information did not perform as well as the cross-correlation coefficient or the sum of absolute differences using medical images. A study carried out by Cole-Rhodes *et al.* [70] found that mutual information produced a more accurate registration than correlation coefficient when using remote sensing data. This could be because remote sensing images are usually not as noisy as medical images. Cole-Rhodes *et al.* found that mutual information produces a sharper peak at the best-match position, thus, being more suitable for sub-pixel registration of images than correlation coefficient.

The computational complexity of mutual information at each shift position is on the order of $256^2 + mn$ multiplications, additions, and comparisons to find the 2-D histogram and compute the mutual information. Although in many of the similarity measures discussed above m and n can be much smaller than 256 and still work well, in computation of mutual information, an m and n that is much smaller than 256 will produce a sparse 2-D histogram, making computation of the mutual information unreliable. Use of templates of size 256×256 or larger, however, will not only make the template-matching process very slow, it will lose its effectiveness for being a local operator and so will be affected by global geometric difference between images, even when such geometric differences are locally small and negligible.

4.5.1.7 Rotationally invariant template matching Among the similarity measures discussed above, the K-S test determines the similarity between a template and a window independent of their rotational differences if the template and window are circular. When the centers of two circular subimages correspond to each other, they will contain the same scene parts and will have the same statistical properties independent of their orientations. Another statistical property that can be used in

template matching is the probability density of a template or window. The histogram of a template contains information about the probabilities of different intensities in the template and does not depend on the orientation of the template, again if the template is circular. Therefore, the similarity between the histograms of a template and window can be used as their similarity. The correlation coefficient of two histograms may be used as the similarity of the images from which the histograms were obtained.

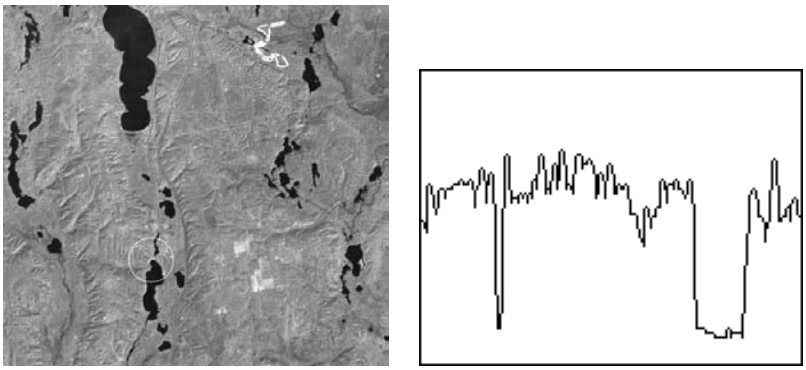
Another measure that determines the similarity between a template and a window independent of the rotational difference between the two is the invariant moments. If both template and window are circular and their centers correspond to each other, they will cover the same scene parts and thus will contain the same pattern independent of their rotational differences. If invariant moments of the template and window are computed according to the formulations given in section 4.3.1.2, similar moments will be obtained when centers of the template and window correspond to each other, independent of their rotational differences. The correlation of the invariant moments after the necessary normalization may be used as the similarity between a template and a window. This method was used in [155] to register satellite images with rotational differences.

Instead of matching circular regions in images, intensities along rings (perimeters of circular regions) may be used in matching. When the centers of two rings of the same radius correspond to each other, the same sequence of intensities will form along the rings independent of their orientations. The orientation of a ring can be normalized with respect to its major axis as computed by (4.49) and (4.50) so that the start of the sequence can be taken to always be the point where the major axis intersects the ring. Since the major axis intersects the circle at two points, two sequences representing the window should be tested against the sequence representing the template. Computation of the major axes of rings can be avoided if the 1-D arrays representing the rings are matched n times for a ring composed of n pixels. One sequence is shifted over the other and the similarity between the two is determined by the cross-correlation coefficient or the sum of absolute differences. Figure 4.7 shows an example of a ring and the sequence of intensities obtained from it.

To achieve a more reliable matching, the rings should be unique and highly informative. A criterion similar to that used to select templates in the reference image will be needed to select the rings. Rings that are centered at unique and highly informative landmarks are expected to be highly informative and unique because landmarks are selected as the centers of highly informative and unique regions. The landmarks and the rings can be combined to achieve a higher positional accuracy when searching for corresponding points in two images. For instance, the result of the point-pattern matching algorithms discussed in this chapter can be further refined by selecting rings centered at the landmarks in the reference image and searching for them in the neighborhoods of the corresponding landmarks in the sensed image.

4.5.2 Gaussian-weighted templates

Template matching as discussed above treats all pixels in a template or a window similarly. If the images to be registered have linear or nonlinear geometric differ-



(a) (b)

Fig. 4.7 (a) A ring centered at a landmark. (b) The sequence of intensities forming the ring counter-clockwise starting from the rightmost pixel on the ring. The horizontal axis shows the pixel positions along the ring and the vertical axis shows the intensities of the pixels.

ences, a template and a window whose centers correspond to each other may have considerable linear or nonlinear geometric differences away from their centers and such differences could increase as one moves away from the center points. To reduce the effect of these differences in template matching, the pixel intensities can be multiplied by weights that decrease from the centers. If the centers of a template and a window correspond to each other, when the pixels are weighted in this manner, more information will be used from pixels at and near the centers of the template and window where differences are smaller. By scaling down intensities away from the centers of corresponding template and window, the effect of geometric difference between them is reduced.

Very small standard deviations will use information only from a small region at the center of the template and window and the information may not be sufficient to accurately find their similarity. On the other hand, if a very large standard deviation is used, intensities in the template and window will hardly change, producing matching results similar to those obtained without Gaussian weighting. Typically, the standard deviation of the Gaussian is set equal to the radius of the template and window. Gaussian weighting is especially effective when registering stereo images where geometric differences between a template and window pair increase as one moves away from the centers of the template and window.

4.5.3 Template size

A template that contains very little information because it represents a rather homogeneous area may produce a false match. One way to avoid false matches is to set the size of a template proportional to the information content in the template. In rather homogeneous areas, therefore, larger templates are needed, and in detailed areas smaller templates are sufficient. This has proven very effective in the matching

of stereo images [293]. If the local geometric difference between images is not large, a small number of landmark correspondences is sufficient to register the images. Therefore, selection of landmarks or templates in rather homogeneous image areas can be avoided altogether.

To achieve highly reliable matches, it is required that the selected templates be unique and highly informative. Algorithm 3.2 described steps for determining control points that were centered at unique and highly informative regions. The same algorithm can be used to select highly informative and unique templates. The area size used to select a template size should depend on the information present in the area. In that way, all selected templates will be locally unique and have sufficient information to produce highly reliable matches.

4.5.4 Coarse-to-fine methods

Coarse-to-fine methods are used to speed up the matching process or to increase the registration accuracy. The coarse-to-fine idea can be implemented at image level, at template level, or during the correspondence search.

4.5.4.1 At image level If the given images have nonlinear geometric differences, as the scale of the images is reduced, the geometric difference between the images reduces, making the images more similar and increasing the registration accuracy. The transformation function that registers the images at a lower scale, however, will only register the images approximately at a higher scale. Therefore, after approximately registering the images at a scale, the process should be repeated to find more corresponding landmarks in the images at a higher scale and compute a more elaborate transformation that can compensate more accurately for the local geometric difference between the images.

The coarse-to-fine approach is particularly useful when high-resolution images with large local geometric differences need to be registered. At a coarse resolution (small scale), the global geometric difference between the images is determined and the sensed image is resampled to globally align the reference image. As the scale is increased, more landmarks are selected to better represent the local geometric differences between the images. The process is repeated until images at full scale are registered.

Coarse-to-fine matching at image level is found particularly effective when the images represent different views of a scene. Quam [319] developed a coarse-to-fine hierarchical matching algorithm that was shown to be very effective in the registration of multi-view terrain images. The method used disparity estimates from a low scale to change the geometry of the sensed image at one level higher scale to register the reference image. Since this warping makes the sensed image more similar to the reference image, the reliability of the correlation matching increases. Correlation results obtained at a level are used to estimate disparities between the reference and sensed images at that level. The disparities are then used to warp the sensed image at one level higher scale and the process is repeated until the images register at full scale.

To reduce the scale of an image, the image is first convolved with a Gaussian to smooth small and noisy details. The smoothed image is then resampled to an image at a smaller scale. Alternatively, local averaging may be carried out to reduce the scale of an image. For instance, to reduce the scale of an image by a factor of two, the average of 2×2 neighborhoods in the image are used as the values of pixels in the lower-scale image. Lowering the scale by a factor of four will require averaging pixels in 4×4 neighborhoods and using the values as intensities of pixels in the lower-scale image.

If images $f_1[\,]$ and $g_1[\,]$ are produced from images $f_0[\,]$ and $g_0[\,]$, respectively, by reducing their scales by a factor of 2 and, assuming images $f_1[\,]$ and $g_1[\,]$ are registered with functions $X = T_x(x, y)$ and $Y = T_y(x, y)$, the same functions may be used to approximately register images $f_0[\,]$ and $g_0[\,]$ using $X = 2T_x(x/2, y/2); Y = 2T_y(x/2, y/2)$. The approximately registered images can then be more finely registered by finding correspondence between more landmarks in the images and using the correspondences to compute a more elaborate transformation function that can compensate for more local geometric differences between the images.

4.5.4.2 At template level The scale of the images may be kept unchanged, but smaller templates may be initially used to find likely match positions very quickly. Then, larger templates may be used to find the best-match position among the likely ones [335]. Alternatively, a subset of a template may be used to find the likely matches and then use the entire template to find the best match among the likely ones [159]. The template size can be kept unchanged, but initially a fast similarity measure may be used to identify the likely match positions and then a more expensive similarity measure may be used to locate the best-match position among the likely ones. For example, initially, the average intensity of pixels in the template or the histogram of the template may be used to determine the likely match positions. Then, the cross-correlation coefficient may be used to find the best-match position among the likely ones.

After locating a template in an image at the full scale, one may go one step further and determine the position of the template with subpixel accuracy [104, 384]. To achieve this, a biquadratic polynomial is fitted to the similarities in the 3×3 neighborhood centered at the best-match position by the least-squares method. The peak of the polynomial is then determined with subpixel accuracy. Assuming $f(x, y) = 0$ is the function that fits to the similarities by the least-squares method, the best-match position is computed with subpixel accuracy by solving $\partial f(x, y)/\partial x = 0$ and $\partial f(x, y)/\partial y = 0$ for x and y.

4.5.4.3 During the search A search can also be carried out from coarse to fine. For instance, when searching for the position of a template in an image, the search can be first carried out at large intervals. When a template of radius r pixels is used, the search is initially performed at every other pixel or every nth pixel. In the neighborhoods where a sufficiently high match rating is obtained, the search is then carried out at every pixel to accurately find the match position.

In registration methods where the two sets of landmarks are selected independently, first, a subset of the landmarks from each set is matched and the transformation

function mapping the two sets to each other is estimated. The correctness of the transformation is then verified using all landmarks in the two sets. The subsets used in the initial match may be taken randomly [117] or selectively by choosing those that form the minimum-spanning-tree [427] or the convex hull of the landmarks [163].

4.6 SUMMARY

In this chapter, methods for determining the correspondence between two sets of points, lines, and regions were given. Also discussed were methods for finding windows in the sensed image that correspond to templates selected in the reference image. Similarity measures for template matching were reviewed and coarse-to-fine approaches to image registration were presented. Further reading on these topics is provided below.

4.7 BIBLIOGRAPHICAL REMARKS

In addition to point-pattern matching using the minimum-spanning-tree [427] and convex hull vertices [163], point-pattern matching using relaxation labeling [290, 226] has been achieved. An algorithm for matching composite sets of point patterns using a tree search algorithm is given by Umeyama [391]. This algorithm allows point patterns from an object or scene with flexible parts to be matched. A method described by Chui and Rangarajan [66] uses thin-plate splines as the transformation function to detect the outliers and finds the correspondence between points in two images with nonlinear geometric differences.

Hausdorff distance has also been used in point-pattern matching [202, 281]. Although Hausdorff distance is sensitive to the outliers, once the outliers are eliminated through a filtering process, it can be used effectively to determine the match-rating between two points sets. Hausdorff distance can be used to locate objects in images if the objects and images are both defined by points. This is achieved by finding the transformation parameters that minimize the Hausdorff distance between the object and the image [336]. If a second image is taken as the object, the process will in effect find the transformation parameters that register the images.

Use of clustering in image registration was pioneered by Stockman *et al.* [369]. Clustering has been used as a means to determine the locations of objects in images and to recognize them [59, 111, 368, 370]. It is a general methodology for determining the consensus among evidences available about the parameters that relate two images or a model and an image. The challenge in clustering is to set up an accumulator in a high-dimensional space that has minimal coverage but still contains the desired cluster in the parameter space. In this chapter, instead of a 6-D accumulator, six 1-D accumulators were used because only one cluster is anticipated in the parameter space. A second challenge in clustering is to reduce the number of point combinations that are used to find the clusters. In this chapter, the use of points on the convex hulls of the point sets was suggested. Olson [295] suggested the use of randomized points similar to random sample consensus [117] for this purpose. This step is needed to make the

computation time manageable while still finding the required cluster or clusters in the parameter space. Clustering is a generalization of the Hough transform [17, 170], which does not perform well under noise. However, it performs well under outliers if they represent a small percentage of the overall data. For further discussions on pose clustering and its applications, see the work of Dhome and Kasvand [93], Linnainmaa *et al.* [245], Silberberg *et al.* [359], and Thompson and Mundy [383].

In addition to invariants of five co-planar points as given in (4.14) and (4.15), the cross ratio of four distinct points on a line, the relation between two points and two lines in a plane, and the relation between two coplanar conics remain unchanged under projective transformation and may be used to register two images. Extensive discussions of invariants can be found in a book on invariance edited by Mundy and Zisserman [283].

Matching of directed lines in images by clustering is described by Stockman *et al.* [369]. 4-D clustering was performed to determine parameters (S, θ, h, k). To make the process efficient, clustering was performed in two stages. In the first stage, the approximate cluster center was determined by coarsely quantizing the parameter space. Once the approximate cluster center was determined, finer clustering was performed within a small neighborhood of the estimated cluster center. Line matching via minimization of a distance measure [112] and using the dual quaternion representation of lines [401] have also been proposed. In [218] the point that is overall closest to all the lines is determined and used as the origin of the coordinate system for the lines. The representations for the lines are then revised with respect to the new origin. The process, in effect, repositions the lines in each set before matching them, making the process independent of the coordinate systems of the images from which the lines are extracted.

Line-matching in stereo images has been achieved via hypothesis-and-test [13]. The relation between lines in stereo images is no longer a fixed affine or projective transformation, so relations (4.1) and (4.5) can no longer be used as constraints to align images and verify the correctness of the correspondences. However, epipolar constraint, continuity constraint, and smoothness constraint [211] may be used to eliminate or reduce the number of incorrect correspondences.

Various forms of discrete and continuous relaxation techniques have appeared in the literature [178, 179, 200, 230, 334] in various computer vision applications, including image and shape matching [84, 224], mostly in the late 70s and early 80s. An excellent review of relaxation labeling techniques and their applications in computer vision and image analysis is provided by Kittler and Illingworth [227].

Among the similarity measures discussed for template matching, mutual information has been studied the most. The strength of mutual information is in its ability to find similarity between images with considerable intensity differences. If a function exists that uniquely maps intensities in the sensed image to intensities in the reference image, a high similarity will be obtained between the images when using mutual information. This similarity measure is especially useful in the registration of medical images. For example, although MR and CT scanners assign different values to a tissue type, the same tissue type will receive the same value in an imaging modality. Therefore, a template taken from a CT image will have high mutual information

with the corresponding window in the MR image. Different implementations of mutual information have been proposed [253, 372, 381, 399] and the performance of algorithms using mutual information as the similarity measure has been determined [119, 409] and compared with performances of the sum of absolute differences and the cross-correlation coefficient in image registration [302]. An extensive review of mutual information in image registration has been provided by Pluim *et al.* [311].

5

Transformation Functions

In the preceding chapters, methods for determining point features in images and methods for determining correspondence between them were discussed. In this chapter, methods for determining a transformation function that uses the coordinates of corresponding points in the images to find the correspondence between all points in the images are discussed. Linear and nonlinear transformations are covered. Among the nonlinear transformations, thin-plate splines, multiquadrics, weighted mean, piecewise linear, and weighted linear methods are presented.

The problem to be solved is as follows. Given the coordinates of N corresponding points in the reference and sensed images,

$$\{(x_i, y_i), (X_i, Y_i) : i = 1, \ldots, N\}, \tag{5.1}$$

we would like to determine a transformation function $\mathbf{f}(x, y)$ with components $f_x(x, y)$ and $f_y(x, y)$ that satisfies

$$X_i = f_x(x_i, y_i), \tag{5.2}$$
$$Y_i = f_y(x_i, y_i), \qquad i = 1, \ldots, N, \tag{5.3}$$

or

$$X_i \approx f_x(x_i, y_i), \tag{5.4}$$
$$Y_i \approx f_y(x_i, y_i), \qquad i = 1, \ldots, N. \tag{5.5}$$

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

Once $\mathbf{f}(x, y)$ is determined, the coordinates of the corresponding point in the sensed image can be determined, given the coordinates of a point (x, y) in the reference image.

If we rearrange the coordinates of corresponding points in the images as follows.

$$\{(x_i, y_i, X_i) : i = 1, \ldots, N\}, \tag{5.6}$$
$$\{(x_i, y_i, Y_i) : i = 1, \ldots, N\}, \tag{5.7}$$

we see that the components of the transformation defined by equations (5.2) and (5.3) represent two single-valued surfaces that interpolate data points given in (5.6) and (5.7), and the components of the transformation defined by (5.4) and (5.5) represent single-valued surfaces approximating data points in (5.6) and (5.7). The two components of a transformation have the same form and can be determined in the same manner. Therefore, we will consider the general problem of finding a single-valued function that interpolates or approximates a set of irregularly spaced data in the plane,

$$\{(x_i, y_i, f_i) : i = 1, \ldots, N\}. \tag{5.8}$$

That is, we find $f(x, y)$ such that

$$f_i = f(x_i, y_i), \qquad i = 1, \ldots, N, \tag{5.9}$$

or

$$f_i \approx f(x_i, y_i), \qquad i = 1, \ldots, N. \tag{5.10}$$

In volumetric images, the control points are in 3-D and the set of corresponding points can be written as

$$\{(x_i, y_i, z_i), (X_i, Y_i, Z_i) : i = 1, \ldots, N\}. \tag{5.11}$$

If we rearrange the coordinates of corresponding points into

$$\{(x_i, y_i, z_i, X_i) : i = 1, \ldots, N\}, \tag{5.12}$$
$$\{(x_i, y_i, z_i, Y_i) : i = 1, \ldots, N\}, \tag{5.13}$$
$$\{(x_i, y_i, z_i, Z_i) : i = 1, \ldots, N\}, \tag{5.14}$$

the single-valued functions interpolating the data,

$$X_i = f_x(x_i, y_i, z_i), \tag{5.15}$$
$$Y_i = f_y(x_i, y_i, z_i), \tag{5.16}$$
$$Z_i = f_z(x_i, y_i, z_i), \qquad i = 1, \ldots, N, \tag{5.17}$$

or approximating the data,

$$X_i \approx f_x(x_i, y_i, z_i), \tag{5.18}$$
$$Y_i \approx f_y(x_i, y_i, z_i), \tag{5.19}$$
$$Z_i \approx f_z(x_i, y_i, z_i), \qquad i = 1, \ldots, N, \tag{5.20}$$

will represent the components of a transformation function that register the images. Again, because the three components of the transformation are similar, they can be determined in the same manner. Therefore, we will consider the general problem of finding a single-valued function that interpolates or approximates

$$\{(x_i, y_i, z_i, f_i) : i = 1, \ldots, N\}. \tag{5.21}$$

A component of a transformation function can be represented by a variety of functions. The type of function selected should depend on the type and severity of the geometric difference between the images, the accuracy of the point correspondences, and the density and organization of the points. In the following, we will examine the properties of different transformation functions and provide a guide to their selection using information about the images.

Because of the nonlinear nature of the image acquisition process and, sometimes, the deformable nature of the scene, the images to be registered often have nonlinear geometric differences. Sometimes, the nonlinear geometric difference between two images is small and negligible and a linear transformation can be used to register them. In other cases, the geometric difference between the images is quite large and complex, and a composite of local transformations is needed to register them. The problem becomes even more difficult when the images to be registered contain rigid as well as nonrigid bodies. The transformation function for the registration of such images should be able to rigidly register parts of the images while nonrigidly registering the rest.

Knowing a set of corresponding control points in two images, many transformation functions can be used to accurately map the control points to each other. A proper transformation function will accurately align the remaining points in the images as well. Some transformation functions may map control points in the sensed image to corresponding control points in the reference image accurately, but they warp the sensed image too much, causing large errors in the registration away from the control points. Also, since a transformation is computed from the point correspondences, error in the correspondences will carry over to the transformation function. A transformation is needed that can smooth the noise and small inaccuracies in the correspondences. When noise and inaccuracies are present, approximation methods are preferred over interpolating methods in image registration.

Various transformations have been used in image registration. In the following sections, the properties of these transformations are examined and a guide to them is provided. To determine the behavior of transformation functions and to find their accuracies in image registration, a number of test images are prepared, as shown in

Fig. 5.1. Figure 5.1a is an image of size 512×512 pixels containing a 32×32 uniform grid, and Fig. 5.1b shows the same grid after being translated by (5,8) pixels. Figure 5.1c shows counterclockwise rotation of the grid by 0.1 radians about the image center, Fig. 5.1d shows scaling of the grid by 1.1 with respect to the image center, Fig. 5.1e shows the grid after linear transformation by

$$X = 0.7x - y + 3, \tag{5.22}$$

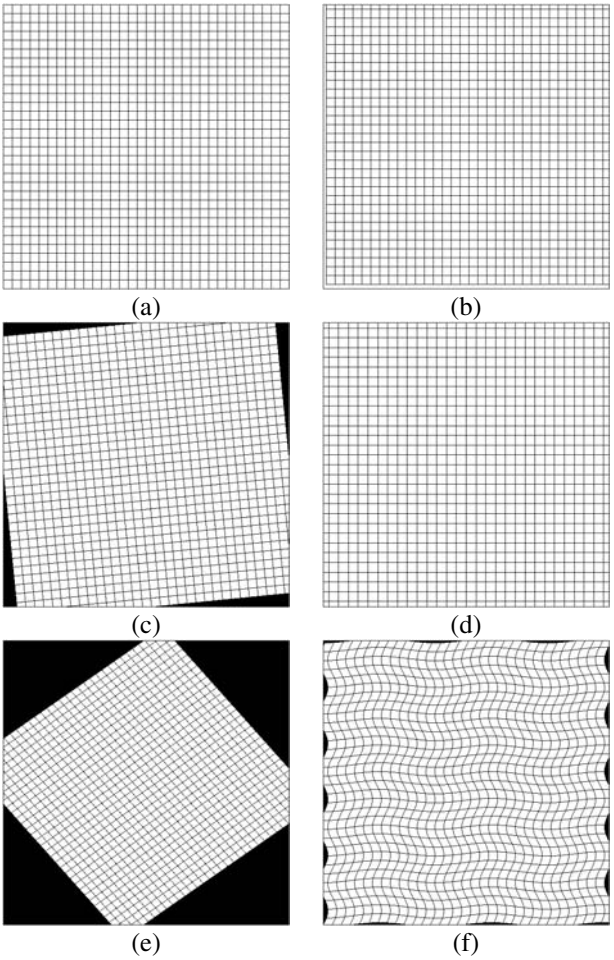


Fig. 5.1 (a) A 32×32 uniform grid in an image of size 512×512 pixels. (b) The grid after translation by (5,8) pixels. (c) The grid after rotation by 0.1 radians counterclockwise about the image center. (d) The grid after scaling by 1.1 with respect to the image center. (e) The grid after the linear transformation defined by (5.22) and (5.23). (f) The grid after the nonlinear transformation defined by (5.24) and (5.25).

$$Y = 0.9x + 0.8y + 5, \tag{5.23}$$

and Fig. 5.1f shows nonlinear transformation of the grid by sinusoidal function

$$X = x - 8\sin(y/16), \tag{5.24}$$
$$Y = y + 4\cos(x/32). \tag{5.25}$$

We will use the coordinates of all, or a subset, of the corresponding grid points in the images to estimate these transformations.

Different densities and organization of point correspondences will be used to estimate the geometric difference between Fig. 5.1a and Figs 5.1b–f. Figure 5.2a shows all the grid points in Fig. 5.1a, Fig. 5.2b shows a uniformly spaced subset of the grid points, Fig. 5.2c shows a random subset of the grid points, and Fig. 5.2d shows a subset of the grid points with variation in the density of the points. Although uniformly spaced points are rarely obtained in images, they are used here to determine the influence of the spacing and organization of the points on registration accuracy. The five geometric transformations are used to represent the geometric difference between the reference and sensed images and four sets of corresponding points are used to estimate the transformation in each case. Overall, there will be twenty test cases to consider.

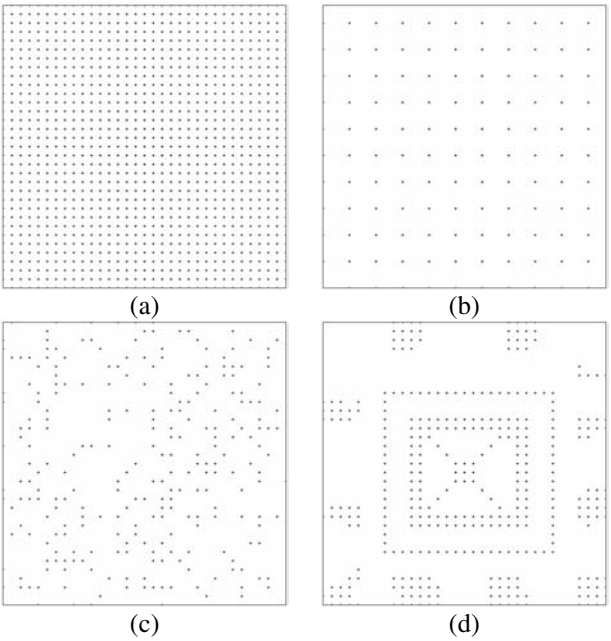


Fig. 5.2 Various densities and organization of data points. (a) A grid of uniformly spaced data. (b) A uniform grid with fewer data points. (c) A set of randomly spaced data points. (d) A set of data points with a varying density.

The points in the reference image are the grid points shown in Figs 5.2a–d. The points in the sensed image are obtained by:

1. translating the points in the reference image by (5,8) pixels;
2. rotating the points by 0.1 radians about the image center and rounding the obtained coordinates;
3. scaling the points by 1.1 with respect to the image center and rounding the coordinates;
4. transforming the points using the linear transformation given in (5.22) and (5.23) and rounding the coordinates;
5. transforming the points using the sinusoidal transformation given by (5.24) and (5.25) and rounding the coordinates.

From the coordinates of corresponding points, the transformations are estimated and compared to the actual ones. In these experiments, although mismatches do not exist, corresponding control points contain rounding errors.

5.1 SIMILARITY TRANSFORMATION

The similarity transformation or the transformation of the Cartesian coordinate system represents global translational, rotational, and scaling differences between two images and is defined by

$$X = S[x\cos\theta + y\sin\theta] + h, \tag{5.26}$$
$$Y = S[-x\sin\theta + y\cos\theta] + k, \tag{5.27}$$

where S, θ, and (h, k) are scaling, rotational, and translational differences between the images, respectively. These four parameters can be determined if the coordinates of two corresponding points in the images are known. The rotational difference between the images is determined from the angle between the lines connecting the two points in the images. The scaling difference between the images is determined from the ratio of distances between the points in the images. Knowing the scale s and the rotation θ, the translation parameters (h, k) are then determined by substituting the coordinates of midpoints of lines connecting the points into equations (5.26) and (5.27) and solving for h and k.

If the correspondences are known to be noisy or inaccurate, more than two correspondences should be used to determine the parameters by least-squares [324] or clustering [369]. Least-squares is preferred when the inaccuracies can be modeled by zero-mean noise, while clustering is preferred when large errors (outliers) exist among the correspondences.

In 3-D, the similarity transformation can be written as

$$\mathbf{P} = S\mathbf{R}\mathbf{p} + \mathbf{T}, \tag{5.28}$$

where $\mathbf{P}$ and $\mathbf{p}$ represent corresponding points in the sensed and reference images, S is the scaling, $\mathbf{R}$ is an orthonormal matrix representing the rotational difference, and $\mathbf{T}$ is a vector representing the translational difference between the images. First, the scaling difference between the images is determined from the average ratio of distances between corresponding point pairs in the images. Knowing the scaling parameter S, the translation vector $\mathbf{T}$ and the rotation matrix $\mathbf{R}$ are determined as follows. Assuming coordinates of corresponding points in the images after correction for scale are $\{\mathbf{p}_i = (x_i, y_i, z_i), \mathbf{P}_i = (X_i, Y_i, Z_i) : i = 1, \ldots, N\}$, the relation between corresponding points in the images can be written as

$$\mathbf{P}_i = \mathbf{R}\mathbf{p}_i + \mathbf{T}, \qquad i = 1, \ldots, N. \tag{5.29}$$

Vector $\mathbf{T}$ and matrix $\mathbf{R}$ are determined by minimizing

$$E^2 = \sum_{i=1}^{N} ||\mathbf{P}_i - (\mathbf{R}\mathbf{p}_i + \mathbf{T})||^2. \tag{5.30}$$

An efficient algorithm for determining $\mathbf{R}$ and $\mathbf{T}$ has been described by Arun *et al.* [10]. In this algorithm, the rotation matrix is obtained first by minimizing

$$E_R^2 = \sum_{i=1}^{p} ||\mathbf{Q}_i - \mathbf{R}\mathbf{q}_i||^2, \tag{5.31}$$

and then the translation vector $\mathbf{T}$ is determined from

$$\mathbf{T} = \mathbf{P} - \mathbf{R}\mathbf{p}. \tag{5.32}$$

In equation (5.31), $\mathbf{Q}_i = \mathbf{P}_i - \bar{\mathbf{P}}$, $\mathbf{q}_i = \mathbf{p}_i - \bar{\mathbf{p}}$, and $\bar{\mathbf{P}}$ and $\bar{\mathbf{p}}$ are the centers of gravity of the points in the sensed and reference images, respectively.

If the correspondences are known to be accurate, the translational and rotational differences between the images can be determined from the coordinates of three non-colinear corresponding points in the images. If quantization noise is present, more than three corresponding points are needed to reduce the effect of noise and determine the registration parameters by the least-squares method as outlined above. If outliers are also present, parameters determined from combinations of three correspondences at a time should be clustered to obtain the transformation parameters. Note that if three corresponding points are used, determination of the registration parameters involves the solution of a nonlinear system of equations, while if four or more points are used, the transformation parameters can be determined by solving a linear system of equations [97].

If the images do not have a scaling difference, the similarity transformation becomes the rigidity transformation. Therefore, the rigidity constraint can be used to find the parameters of the transformation. If noise is not zero-mean, or outliers exist,

the least squares method may fail to find the correct transformation parameters. To find the transformation parameters by specifically using the rigidity constraint, in Ding and Goshtasby [96], about a dozen correspondences that had the highest match rating and were widely spread over the image domain were selected from a large set of corresponding points. Then, transformation parameters were determined from combinations of four points at a time and the combination that produced a linear transformation closest to a rigid transformation was used to register the images. The idea is to use a few correspondences that are very accurate rather than using a large number of correspondences, some of which are inaccurate.

Similarity transformation can be used to register aerial and satellite images where the scene is rather flat and the platform is very far from, and looking directly down at, the scene. In medical images, bony structures can be registered with this transformation. Similarity transformation is widely used in the registration of multimodality brain images since the brain is contained in a rigid skull and images of the brain taken a short time apart do not have nonlinear geometric differences (if sensor nonlinearities do not exist). An example of brain image registration using the similarity transformation is shown in Fig. 5.3. The left column shows orthogonal slices of an MR image, the middle column shows orthogonal slices of a fluorodeoxyglucose (FDG) PET image exhibiting brain glucose metabolism taken within a week of the MR acquisition, and the right column shows orthogonal slices of the registered MR and PET images. In this example, control points were selected as locally unique and highly informative

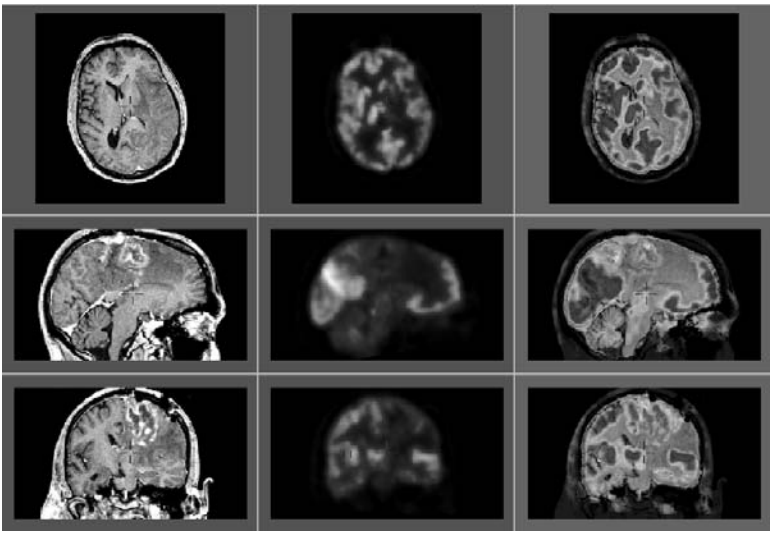


Fig. 5.3 Registration of MR (left) and the PET (middle) brain images by the similarity transformation. The right column shows the registered images. The PET image reveals the cerebral glucose metabolism. The intensities of the fused image are reassigned to enhance areas of high activity.

points, and correspondences were determined by template matching using mutual information as the similarity measure.

5.2 PROJECTIVE AND AFFINE TRANSFORMATIONS

Image acquisition is a projective process and if lens and sensor nonlinearities do not exist, the relation between two images of a rather flat scene can be described by the projective transformation

$$X = \frac{ax + by + c}{dx + ey + 1}, \tag{5.33}$$

$$Y = \frac{fx + gy + h}{dx + ey + 1}. \tag{5.34}$$

Here, a–h are the eight unknown parameters of the transformation, which can be determined if the coordinates of four non-colinear corresponding points in the images are known. If correspondences are known to contain noise, more than four correspondences should be used in the least-squares method to obtain the transformation parameters. When the scene is very far from the camera, the projective transformation can be approximated by the affine transformation

$$X = ax + by + c, \tag{5.35}$$

$$Y = dx + ey + f. \tag{5.36}$$

An affine transformation has six parameters, which can be determined if the coordinates of at least three non-colinear corresponding points in the images are known. Affine transformation is a weak perspective transformation and has been widely used in the registration of satellite and aerial images [29, 366]. Figure 5.4 shows an example of image registration by the affine transformation. Figures 5.4a and 5.4b represent two images of the lunar surface obtained by Apollo 17. The images have a large overlap. Registration of the images by the affine transformation is shown in Fig. 5.4c. To show the accuracy of the registration, the registered images are subtracted as shown in Fig. 5.4d. Completely dark areas in the subtracted image correspond to the flat areas on the lunar surface. By a careful examination of the subtracted image, we see some misregistration by the affine transformation in the lower part of the image due to the presence of a highly elevated terrain. To more accurately register these images, a nonlinear transformation is needed.

The projective transformation requires that straight lines in the reference image remain straight in the sensed image. If parallel lines remain parallel, affine transformation may be used instead, and if angles between corresponding lines are preserved, the transformation of the Cartesian coordinate system may be used to register the images. Images that can be registered by the transformation of the Cartesian coordinate system can be registered by the affine transformation, and images that can be registered by the affine transformation can be registered by the projective transformation.

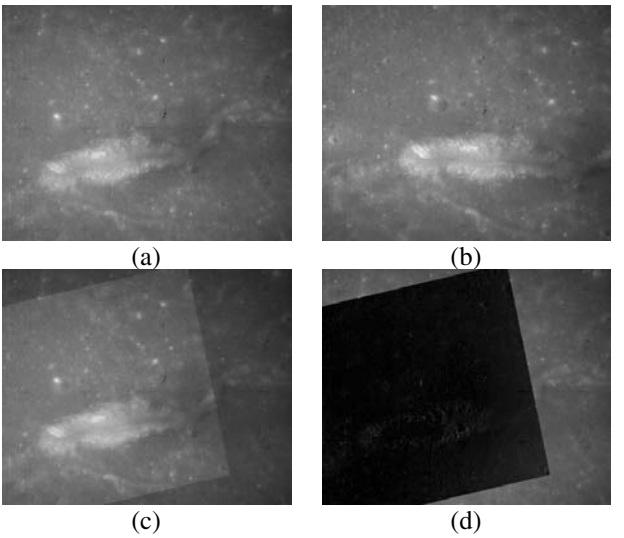


Fig. 5.4 (a), (b) Two images of the lunar surface obtained by Apollo 17. The images show a rather flat area with a mountain range in the middle(the elongated white region). (c) Registration of the images by the affine transformation. (d) Subtraction of the registered images.

If a straight line in the reference image maps to a curve in the sensed image, a nonlinear transformation is needed to register the images. Nonlinear transformations are discussed next.

5.3 THIN-PLATE SPLINE

Thin-plate spline (TPS) or surface spline [181, 273] is perhaps the most widely used transformation function in the registration of images with nonlinear geometric differences. It was first used in the nonrigid registration of remote sensing images [149] and then in the nonrigid registration of medical images [38]. Given a set of single-valued points in the plane as defined by (5.8), the thin-plate spline interpolating the points is defined by

$$f(x,y) = A_1 + A_2x + A_3y + \sum_{i=1}^{N} F_i r_i^2 \ln r_i^2, \tag{5.37}$$

where $r_i^2 = (x-x_i)^2 + (y-y_i)^2 + d^2$. TPS is an interpolating radial basis function. It is defined by a linear term and a weighted sum of radially symmetric logarithmic functions. TPS represents the equation of a plate of infinite extent deforming under loads centered at $\{(x_i, y_i) : i = 1, \ldots, N\}$. The plate deflects under the imposition of loads to take values $\{f_i : i = 1, \ldots, N\}$. Parameter d^2 acts like a stiffness parameter. As d^2 approaches zero, the loads approach point loads. As d^2 increases, the loads

become more widely distributed producing a stiffer (smoother) surface. Equation (5.37) contains $N + 3$ unknowns. By substituting the coordinates of N points as described by (5.8) into (5.37), N relations are obtained. Three more relations are obtained from the following constraints:

$$\sum_{i=1}^{N} F_i = 0, \tag{5.38}$$

$$\sum_{i=1}^{N} x_i F_i = 0, \tag{5.39}$$

$$\sum_{i=1}^{N} y_i F_i = 0. \tag{5.40}$$

Constraint (5.38) ensures that the sum of the loads applied to the plate is 0 so that the plate will remain stationary. Constraints (5.39) and (5.40) ensure that moments with respect to x and y axes are zero, so the surface will not rotate under the imposition of the loads. A minimum of three points are needed to obtain a thin-plate spline. When seven points with coordinates $(1, 1, 10)$, $(1, 20, 10)$, $(20, 1, 10)$, $(20, 20, 10)$, $(11, 8, 15)$, $(7, 17, 25)$, and $(16, 12, 20)$ are used and $d^2 = 0$, we find $A_1 = 61.622$, $A_2 = 0.079$, $A_3 = -0.323$, $F_1 = 0.0001$, $F_2 = 0.0313$, $F_3 = 0.0695$, $F_4 = -0.0498$, $F_5 = 0.0344$, $F_6 = -0.0014$, and $F_7 = -0.0215$.

Using thin-plate splines to map Figs 5.1b–f to Fig. 5.1a with the point arrangements shown in Fig. 5.2, the results shown in Table 5.1 are obtained. The errors in each entry of the table are obtained by using the coordinates of corresponding points to determine the transformation parameters. The obtained transformation is then used to deform the sensed image to overlay the reference image. This transformation maps the control points in the sensed image exactly to the corresponding control points in the reference image. The grid points shown in Figs 5.2a–d were used to estimate the transformation functions and the grid points shown in Fig. 5.1a were used to estimate the maximum (MAX) error and the root-mean-squared (RMS) error in each case. All errors in the

Table 5.1 MAX and RMS errors of TPS in image registration under different densities and organization of the points as well as the geometric difference between the images.

		Translation	Rotation	Scaling	Linear	Nonlinear
Uniform Dense	MAX	0.0	0.0	0.0	0.0	0.0
	RMS	0.0	0.0	0.0	0.0	0.0
Uniform Sparse	MAX	0.0	1.4	1.4	1.4	27.0
	RMS	0.0	0.6	0.6	0.6	5.8
Random	MAX	0.0	1.4	1.4	1.4	14.1
	RMS	0.0	0.7	0.6	0.6	3.5
Varying Density	MAX	0.0	2.2	2.0	1.4	20.0
	RMS	0.0	0.8	0.7	0.7	6.7

first row in Table 5.1 are zero because TPS is an interpolating function and it maps corresponding control points in the images exactly to each other. Since the control points used to compute the transformation and the points used to measure the error are the same, no errors are obtained. Figure 5.5a shows the resampling of Fig. 5.1f to align with Fig. 5.1a using all the correspondences. Although corresponding control points map to each other exactly, errors are visible away from the control points, especially around image borders.

When the images have translational differences, no errors are observed in the registration independent of the density and organization of the points. This can be attributed to the linear term in TPS and the fact that there is no rounding error between corresponding points in the images. When images have rotational, scaling, or linear differences, TPS produces small errors that are primarily due to rounding error between corresponding points in the images. Errors slightly increase when the density of points varies across the image domain. When images have nonlinear geometric differences, errors are quite large as shown in the last column in Table 5.1. Although corresponding points are mapped to each other, at non control points, errors increase as one moves away from the control points. Figures 5.5a–d show the resampling of image 5.1f to align with image 5.1a using the point arrangements shown in Figs 5.2a–d,

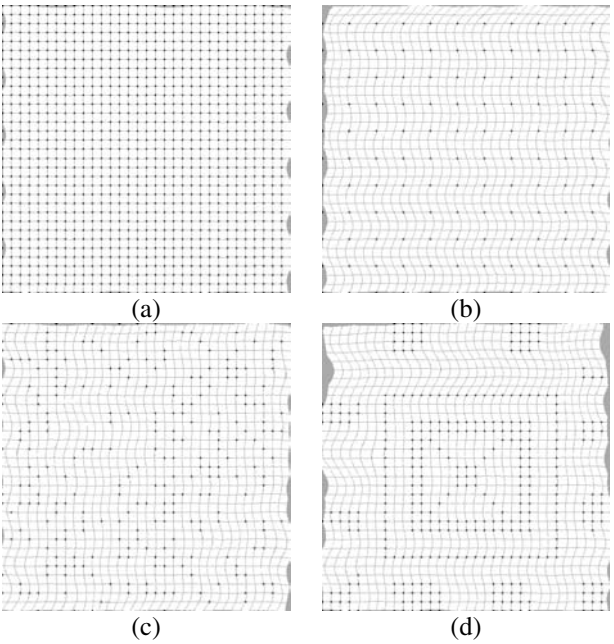


Fig. 5.5 The effect of density and organization of control points in the registration accuracy of TPS. (a)–(d) The resampling of Fig. 5.1f using the control points shown in Figs 5.2a–d, respectively, to align with Fig. 5.1a.

respectively. The control points in the reference image are overlaid with the resampled images for qualitative assessment of the registration accuracy.

From the examples shown in Fig. 5.5, we see that thin-plate spline is not suitable for the registration of images with local geometric differences. In Figs 5.5c and 5.5d although reasonably good accuracy is achieved at and near the control points, errors are large away from the control points. This can be attributed to the fact that logarithmic basis functions, which are radially symmetric, are used to define the transformation. When the arrangement of the points is nonsymmetric, large errors are obtained in areas where large gaps exist between control points. One should also note that as the number of control points increases, since the matrix of coefficients becomes increasingly large, the numerical stability of the method decreases.

The stiffness parameter d^2 changes the shape of the interpolating surface and that changes the registration accuracy. With the point arrangements given in Fig. 5.2, best accuracy is obtained when the stiffness parameter is zero. Larger stiffness parameters increase registration errors in these test cases. The errors reported in Table 5.1 are for a stiffness parameter of 0. To avoid the creation of high fluctuations away from the control points, Rohr *et al.* [332, 400] added a smoothing term to the interpolating spline while letting $d^2 = 0$ to obtain a surface that contained smaller fluctuations but approximated the points. As the smoothness term is increased, the obtained surface becomes smoother and fluctuations become smaller but the surface moves away from some of the control points. If interaction with the system is allowed, one may gradually change the smoothness parameter while observing the components of the transformation and stop the process when the surfaces get sufficiently close to the points. When the points are noisy and/or are irregularly spaced, approximating splines should produce better results than interpolating splines. This, however, necessitates interaction with the system to ensure that the surfaces are not overly smoothed.

Thin-plate splines can be extended to volume splines for the registration of volumetric images [110, 400]. A component of a transformation function for the registration of volumetric images will be a single-valued hypersurface in 4-D,

$$f(x, y, z) = A_1 + A_2 x + A_3 y + A_4 z + \sum_{i=1}^{N} F_i r_i^2 \ln r_i^2, \tag{5.41}$$

where $r_i^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 + d^2$. Parameters A_1, A_2, A_3, and A_4 as well as F_i for $i = 1, \ldots, N$ are determined by solving a system of $N + 4$ linear equations. N equations are obtained by substituting coordinates (5.21) into (5.41) and letting $f(x_i, y_i, z_i) = f_i$. Four more equations are obtained from constraints

$$\sum_{i=1}^{N} F_i = 0, \tag{5.42}$$

$$\sum_{i=1}^{N} x_i F_i = 0, \tag{5.43}$$

$$\sum_{i=1}^{N} y_i F_i = 0, \tag{5.44}$$

$$\sum_{i=1}^{N} z_i F_i = 0. \tag{5.45}$$

These constraints ensure that the spline will not translate or rotate under the loads.

5.4 MULTIQUADRIC

Multiquadric (MQ) is another interpolating radial basis function. Radial basis functions are, in general, defined by

$$f(x, y) = \sum_{i=1}^{N} F_i R_i(x, y). \tag{5.46}$$

Parameters $\{F_i : i = 1, \ldots, N\}$ are determined by letting $f(x_i, y_i) = f_i$ for $i = 1, \ldots, N$ and solving the obtained system of linear equations. In 2-D, $R_i(x, y)$ is a radial function whose value is proportional or inversely proportional to the distance between (x, y) and (x_i, y_i). A surface point, therefore, is obtained from a weighted sum of the radial basis functions. Powell [314] provides an excellent review of radial basis functions.

When

$$R_i(x, y) = [(x - x_i)^2 + (y - y_i)^2 + d^2]^{\frac{1}{2}}, \tag{5.47}$$

$f(x, y)$ is called a multiquadric [182, 183]. As d^2 is increased, a smoother surface is obtained. When basis functions are logarithmic, the obtained transformation becomes a thin-plate spline without the linear term. In a comparative study carried out by Franke [127], it was found that multiquadric followed by thin-plate spline produced the best accuracy in the interpolation of randomly spaced data.

When monotonically increasing radial basis functions are used, the farther a data point is from a surface point, the larger the influence of that data point will be on the surface point. This means that if radial basis functions with monotonically increasing basis functions are used in image registration, control points farther away will have a larger influence on the transformation at a particular image point than nearer control points. Radial basis functions that are monotonically decreasing seem more appropriate for nonrigid image registration because a local deformation affects the registration in the neighborhood of the deformation.

Examples of radial basis functions that monotonically decrease are Gaussians [161, 341],

$$R_i(x,y) = \exp\left\{-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma_i^2}\right\} \tag{5.48}$$

and inverse multiquadrics (IMQ) [52, 127, 183],

$$R_i(x,y) = \left[(x-x_i)^2+(y-y_i)^2+d^2\right]^{-\frac{1}{2}}. \tag{5.49}$$

Franke [127] has shown through extensive experimentation that monotonically decreasing radial basis functions do not perform as well as monotonically increasing radial basis functions when data are uniformly or randomly spaced. When applied to image registration, errors from radial basis functions defined in terms of inverse distances, such as IMQ and Gaussians, are especially large around image borders where the sum of the basis functions approaches zero abruptly.

Errors in registering images 5.1b–f with image 5.1a using multiquadrics with the control points shown in Fig. 5.2 are summarized in Table 5.2. Registration examples are shown in Fig. 5.6. Comparing these results with those obtained in Table 5.1, we see that TPS generally performs better than MQ, especially when the transformation function to be estimated is linear. This can be attributed to the linear term in TPS. However, when the transformation function is nonlinear, MQ performs slightly better. It should be mentioned that a linear term may also be added to the equation of MQ. However, since MQ is used in nonrigid registration, the addition of a linear term will not change the registration result noticeably.

Monotonically decreasing radial basis functions can be defined locally so that the coordinates of a small number of control points will be sufficient to find a functional value. Such functions are called compactly supported radial basis functions. Wendland [407] describes a family of compactly supported radial basis functions that are positive definite and guarantee a solution. For instance, function

Table 5.2 MAX and RMS registration errors of MQ using the control points in Fig. 5.2 and the deformations shown in Figs 5.1b–f in comparison with Fig. 5.1a. Numbers inside parentheses show errors when optimal d^2 was used, while numbers not in parentheses are errors obtained when $d^2 = 0$.

		Translation	Rotation	Scaling	Linear	Nonlinear
Uniform Dense	MAX	0.0	0.0	0.0	0.0	0.0
	RMS	0.0	0.0	0.0	0.0	0.0
Uniform Sparse	MAX	12.6 (1.4)	36.7 (6.0)	36.8 (2.8)	22.4 (5.1)	35.4 (29.2)
	RMS	1.5 (0.2)	3.6 (0.9)	3.2 (0.6)	3.3 (1.0)	5.6 (5.5)
Random	MAX	33.5 (3.2)	33.4 (14.0)	21.6 (6.0)	25.3 (8.0)	66.1 (23.1)
	RMS	3.0 (0.2)	3.3 (2.1)	1.7 (1.3)	3.4 (1.4)	7.7 (2.9)
Varying Density	MAX	74.2 (46.8)	63.8 (36.7)	65.5 (37.6)	25.1 (11.3)	70.2 (41.3)
	RMS	8.4 (4.4)	7.2 (3.5)	7.1 (3.4)	3.7 (1.9)	9.6 (2.3)

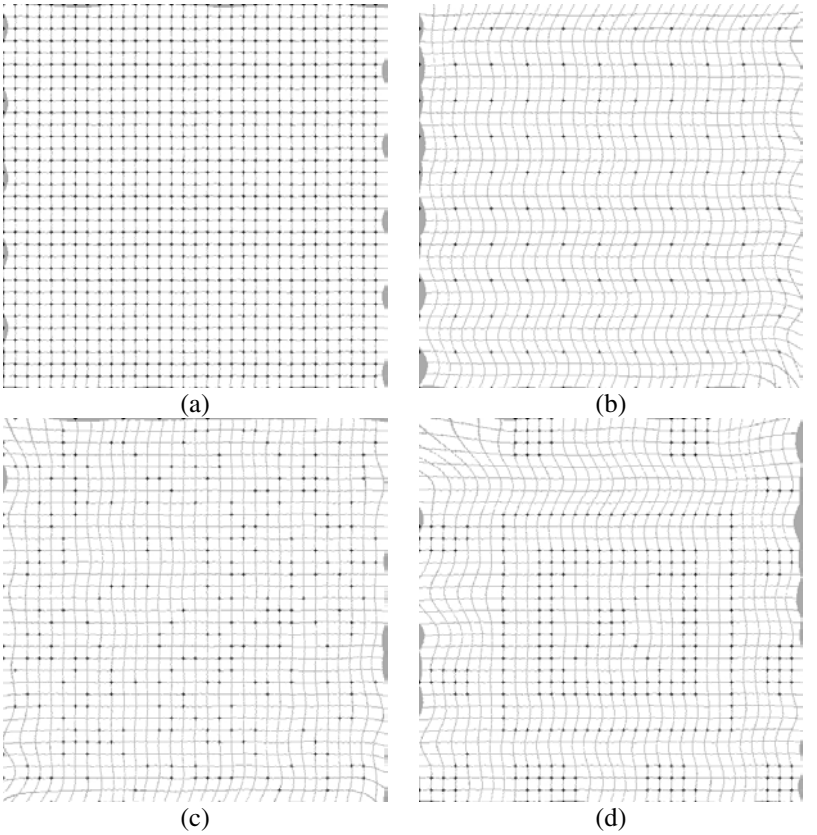


Fig. 5.6 Registration results using multiquadrics as the transformation function. (a)–(d) Resampling Fig. 5.1f to overlay Fig. 5.1a using the control points shown in Figs 5.2a–d.

$$R_i(x,y) = \begin{cases} (x-x_i)^2 + (y-y_i)^2 - a^2, & 0 \le \left[(x-x_i)^2 + (y-y_i)^2\right]^{\frac{1}{2}} \le a, \\ 0, & \left[(x-x_i)^2 + (y-y_i)^2\right]^{\frac{1}{2}} > a, \end{cases} \tag{5.50}$$

is not only 0 at distance a from its center, but its gradient is also 0 there. Therefore, the basis functions smoothly vanish at distance a from their centers and a weighted sum of them will be a smooth function everywhere in the approximation domain. Parameter a should be selected in such a way that within each region of radius a, at least a few points are obtained. Parameters $\{F_i : i = 1, \ldots, N\}$ are again determined by solving a system of linear equations. Note that although the basis functions have local support, a global system of equations has to be solved to find the F_is. Since the sum of the weights varies across the image domain, registration accuracy will greatly depend on the organization of the points.

A comparison of globally defined radial basis functions and compactly supported radial basis functions in image registration is provided by Fornefett *et al.* [122]. Improved accuracy is reported for compactly supported radial basis functions when compared to globally defined radial basis functions in the registration of intra-operative brain images using control points whose density does not vary across the image domain.

Because radial basis functions are radially symmetric, when points are irregularly spaced, they cannot model the geometric difference between two images well. Registration accuracy of radial basis functions depends on the organization of the control points. Functions that can adapt to the organization of control points will be more appropriate for registering images with irregularly spaced control points. Such basis functions are discussed next.

5.5 WEIGHTED MEAN METHODS

Radial basis functions interpolate the given data. They map corresponding control points in the images exactly to each other. Methods that map corresponding control points approximately to each other are called approximation methods. An approximation method is defined by a weighted sum of given data, with the weights having a sum of 1 everywhere in the approximation domain. Approximation of the data points in (5.8) by a weighted mean method is defined by

$$f(x,y) = \sum_{i=1}^{N} f_i b_i(x,y), \tag{5.51}$$

where

$$b_i(x,y) = \frac{R_i(x,y)}{\sum_{i=1}^{N} R_i(x,y)} \tag{5.52}$$

and $R_i(x,y)$ is a monotonically decreasing radial basis function.

When $R_i(x,y)$ is a Gaussian,

$$R_i(x,y) = \exp\left\{-\frac{(x-x_i)^2 + (y-y_i)^2}{2\sigma_i^2}\right\}, \tag{5.53}$$

the weights are called rational Gaussians or RaGs. Although Gaussians are symmetric, RaGs are nonsymmetric because they have to ensure that the sum of the weights is 1 everywhere in the approximation domain [156]. This property makes the weight functions stretch toward the gaps. Use of RaG weights in the correction of the local geometry of deformed images has been demonstrated [205].

When $R_i(x,y)$ is an inverse multiquadric function, we have

$$R_i(x,y) = [(x-x_i)^2 + (y-y_i)^2 + d_i^2]^{-1/2}. \tag{5.54}$$

As d_i approaches 0, $f(x_i, y_i)$ gets closer to f_i. When $d_i = 0$ for all i, the function will interpolate the points. When Gaussians are used, as the standard deviations are reduced, the same effect is observed and at the limit when $\sigma_i = 0$ for all i, the function interpolates the points. At very small d_i's or σ_i's, flat spots will appear in function f at and surrounding the data points. This is demonstrated in a simple example in Fig. 5.7. Consider the following five data points: $(0, 0, 0)$, $(1, 0, 0)$, $(1, 1, 0)$, $(0, 1, 0)$, and $(0.5, 0.5, 0.5)$. The single-valued RaG surface approximating the points with increasing σ_i's are shown in Figs 5.7a–d. For small σ_i's, flat horizontal spots are obtained around the data points. This means, for large variations in x and y, only small variations will be obtained in f near the data points. The implication of this is that when such a function is used to represent a component of a transformation function, well-separated and uniformly spaced image points surrounding a control point in the reference image will map densely to image points in the neighborhood of the corresponding control point in the sensed image, even when no geometric difference between the images exists.

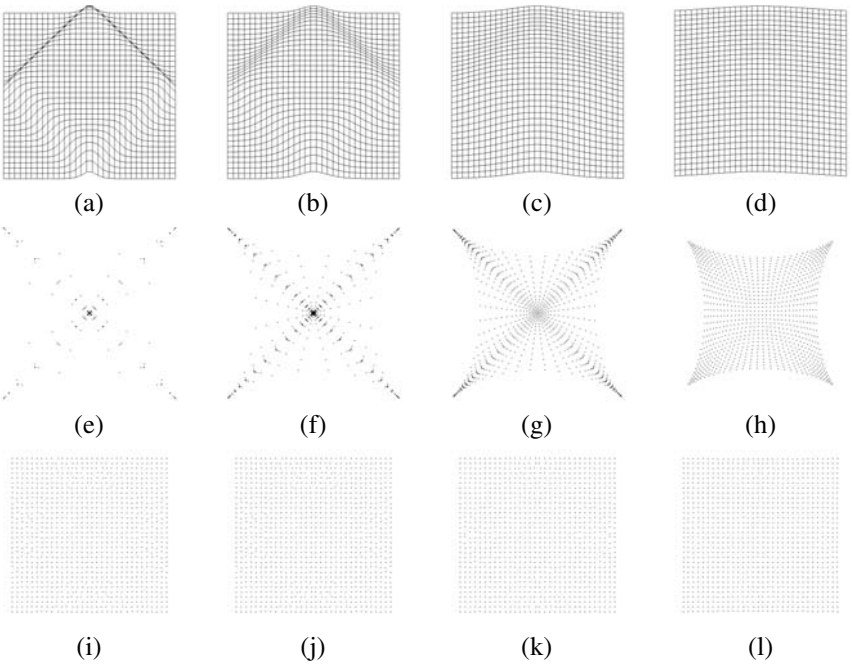


Fig. 5.7 (a)–(d) A component of a transformation represented by a RaG surface using 5 data points, $\sigma_i = \sigma$ for $i = 1, \ldots, 5$, at increasing values of σ. (e)–(h) A transformation function with components represented by single-valued RaG surfaces at increasing values of σ. Shown here are image points in the sensed image mapping to uniformly spaced image points in the reference image. (i)–(l) The same as in (e)–(h) but using a parametric surface for each component of the transformation.

Consider the following 5 correspondences.

$$[(0,0)(0,0)];[(1,0)(1,0)];[(1,1)(1,1)];[(0,1)(0,1)];[(0.5,0.5)(0.5,0.5)]. \quad (5.55)$$

The functions mapping points in the reference image to points in the sensed image for increasing σ_i's are depicted in Figs 5.7e–h. The images show points in the sensed image mapping to uniformly spaced points in the reference image. As σ_i's are reduced, the density of points increases near the control points and the function more closely approximates the data. As σ_i's are increased, spacing between the points becomes more uniform, but the function moves further away from the data, increasing the approximation error. Ideally, we want a transformation function that maps corresponding control points more closely when σ_i's are decreased without increasing the density of the points near the control points. Some variation in the density of resampled points is inevitable if the images to be registered have nonlinear geometric differences. Nonlinear geometric differences change the local density of resampled points in order to stretch some parts of the sensed image while shrinking others, as needed, to make its geometry resemble that of the reference image.

To obtain rather uniformly spaced points in the sensed image when uniformly spaced points are provided in the reference image, it is required to represent each component of the transformation by a parametric RaG surface. Using the corresponding points given in (5.1), we first compute a RaG surface with components $x = x(u,v), y = y(u,v)$, and $f = f(u,v)$ that approximate

$$\{(x_i, y_i, f_i) : i = 1, \ldots, N\}, \quad (5.56)$$

with nodes $\{(u_i = x_i, v_i = y_i) : i = 1, \ldots, N\}$. Given pixel coordinates (x,y) in the reference image, the corresponding parameter coordinates (u,v) are determined from $x = x(u,v)$ and $y = y(u,v)$. This requires the solution of two nonlinear equations. Then, knowing (u,v), $f = f(u,v)$ is evaluated. The obtained value will, in effect, represent the X or the Y coordinate of the image point in the sensed image corresponding to image point (x,y) in the reference image. Figures 5.7i–l show the resampling result achieved in this manner. As can be observed, this transformation function now maps uniformly spaced points in the reference image to almost uniformly spaced points in the sensed image while mapping corresponding control points approximately to each other. The nonlinear equations can be solved by initially letting $u = x$ and $v = y$ and iteratively revising u and v by Newton's method [234].

To determine the accuracy of this approximation method in nonrigid registration, the control points shown in Fig. 5.2 and the deformations shown in Fig. 5.1 are used. MAX and RMS errors for RaG weights are shown in Table 5.3. Registration errors vary with the standard deviations of Gaussians, which are the widths of the weight functions. As the standard deviations of Gaussians are increased, the obtained transformation becomes smoother but errors in registration increase. As the standard deviations of Gaussians are decreased, the transformation becomes less smooth, producing sharp edges and corners, but it maps corresponding control points to each

Table 5.3 MAX and RMS registration errors of the weighted mean method using RaG weights, the control points in Fig. 5.2 and the geometric deformations shown in Fig. 5.1. The errors are for $n = 9$. When n is reduced to 5, the errors shown in parentheses are obtained.

		Translation	Rotation	Scaling	Linear	Nonlinear
Uniform	MAX	0.0 (0.0)	1.0 (0.0)	0.0 (0.0)	1.0 (1.0)	2.2 (1.4)
Dense	RMS	0.0 (0.0)	0.1 (0.0)	0.0 (0.0)	0.1 (0.1)	1.2 (0.6)
Uniform	MAX	0.0 (0.0)	1.4 (0.0)	1.0 (0.0)	1.4 (1.0)	9.0 (7.1)
Sparse	RMS	0.0 (0.0)	0.3 (0.0)	0.1 (0.0)	0.3 (0.1)	4.0 (1.2)
Random	MAX	0.0 (0.0)	1.4 (1.4)	1.4 (1.4)	1.4 (1.4)	10.2 (9.0)
	RMS	0.0 (0.0)	0.6 (0.4)	0.6 (0.4)	0.6 (0.3)	3.7 (1.3)
Varying	MAX	0.0 (0.0)	1.4 (1.4)	1.4 (1.4)	1.4 (1.4)	15.1 (15.0)
Density	RMS	0.0 (0.0)	0.5 (0.2)	0.6 (0.3)	0.5 (0.2)	4.0 (1.8)

other more closely. By decreasing the standard deviations of Gaussians, corresponding points are mapped to each other more closely, but other points in the images may not accurately map to each other. There is a set of standard deviations that produces the least error for a given pair of images. In general, if local geometric difference between images varies sharply, smaller standard deviations should be used than when the geometric difference between images varies smoothly.

The standard deviations of Gaussians are set inversely proportional to the density of the points. This will produce narrow Gaussians where the density of points is high and wide Gaussians where the density of points is low, covering large gaps between the control points. The process can be made totally automatic by setting σ_i equal to the radius r of a circular region surrounding the ith control point in the reference image that contains $n - 1$ other control points ($n \geq 2$). Here, n can be considered a smoothness parameter. The larger its value, the smoother the obtained transformation will be. This is because as n is increased the widths of Gaussians increase, producing a smoother surface, and as n is decreased the widths of Gaussians decrease, producing a more detailed surface and more accurately reproducing local geometric differences between the images.

The results shown in Table 5.3 in parentheses are obtained using $n = 5$, and the results outside parentheses are obtained when $n = 9$. Both MAX and RMS errors are smaller than those shown in Tables 5.1 and 5.2, except for errors in the first row. Since registration is done by approximation, corresponding control points are not mapped to each other exactly, so the errors are not zero; however, they are very small. Examples of image registration by approximation using RaG weights are shown in Fig. 5.8. The irregular boundaries of the resampled images are due to the fact that RaG weights approach zero exponentially and, when a point exists in only one of the images, it cannot be used in the calculation of the transformation, so resampling is not possible near that point. This is in contrast to monotonically increasing radial basis functions such as TPS and MQ where resampling is possible even outside the image domain. Approximation with RaG weights, however, has produced more accurate results in areas where resampling is possible compared to TPS and MQ.

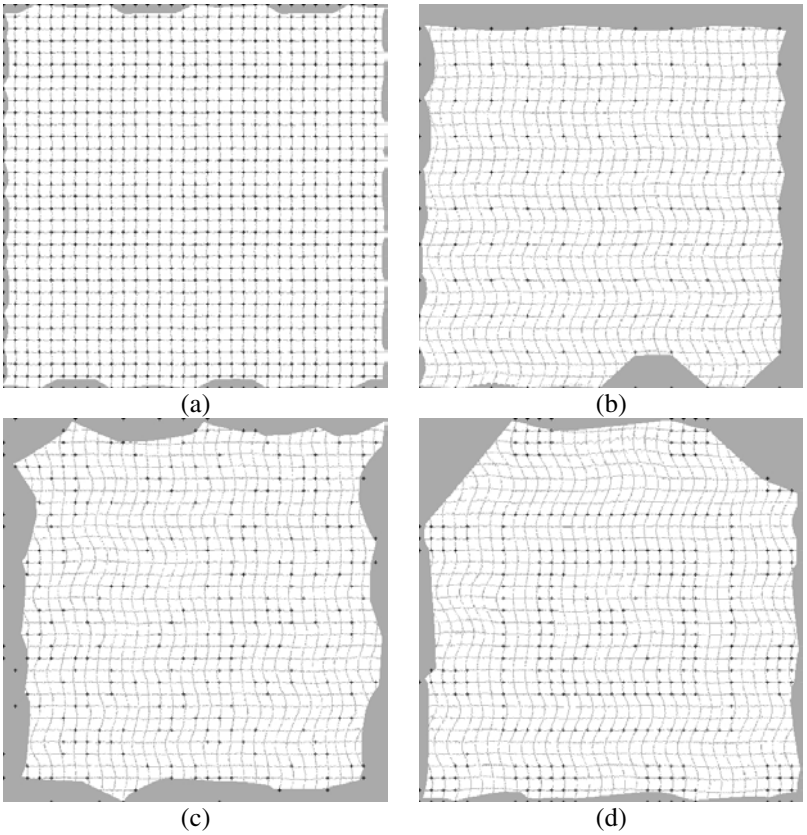


Fig. 5.8 (a)–(d) Resampling of Fig. 5.1f to overlay Fig. 5.1a using RaG weights and the control points shown in Figs 5.2a–d. The control points in the reference image and the resampled sensed image are overlaid to show the quality of registration.

Experiment results show that when $n = 2$ or σ_i is equal to the distance of the ith point to the point closest to it, local deformations, including noise in the correspondences, are reproduced. As n is increased, noise in the correspondences is smoothed more, producing a smoother transformation. Increasing n too much, however, will make it impossible to reproduce sharp geometric differences between the images. n should be selected using information about geometric differences between the images as well as the density of the control points. If no information about the geometric differences between images is provided, information about the density of the control points should be used to select n. Since, usually, more control points are found at and near region boundaries, the method will align region boundaries where dense feature points are available more accurately than region interiors where only sparse control points are available.

Formula (5.51) defines a component of a transformation function in terms of a weighted sum of data (a component of the points in the sensed image). Computationally, this weighted mean approach is more stable than TPS and MQ because it does not require the solution of a system of equations; therefore, it can handle a very large set of control points and in any arrangement. When TPS or MQ is used, the system of equations to be solved may become unstable for a particular arrangement of control points.

Methods have been proposed by Maude [267] and McLain [272] that can make the weighted mean local using rational weights of the form given in (5.52) with

$$R_i(x,y) = \begin{cases} 1 - 3r_i^2 + 2r_i^3, & 0 \le r_i \le 1, \\ 0, & r_i > 1, \end{cases} \tag{5.57}$$

and

$$r_i = [(x - x_i)^2 + (y - y_i)^2]^{\frac{1}{2}} / r_n. \tag{5.58}$$

Here, r_n is the distance of point (x_i, y_i) to its $(n-1)$st closest point in the reference image. Note that since

$$\left[\frac{dR_i(x,y)}{dr_i}\right]_{r_i=1} = 0 \tag{5.59}$$

not only does $R_i(x,y)$ vanish at $r_i = 1$, it vanishes smoothly. Therefore, $f(x,y)$ will be continuous and smooth everywhere in the approximation domain. Having the weight function at a point, next the polynomial interpolating that point and the $(n-1)$ points closest to it is determined. The weighted sum of the polynomial is then used to represent a component of the transformation. Therefore, the formula for a local weighted mean is

$$f(x,y) = \frac{\sum_{i=1}^{N} R_i(x,y) p_i(x,y)}{\sum_{i=1}^{N} R_i(x,y)} \tag{5.60}$$

where $p_i(x,y)$ is the polynomial interpolating data at (x_i, y_i) and the $(n-1)$ data points closest to it. Note that this requires the solution of a small system of equations to find each of the local polynomials. The polynomials encode information about geometric differences between images in small neighborhoods. Therefore, the method is suitable for the registration of images where a large number of points is available and sharp geometric differences exist between the images. When the density of points is rather uniform, local polynomials of degree two will suffice. However, for very irregularly spaced points, a polynomial of degree two may create holes in the transformation function in areas where large gaps exist between the control points. Polynomials of higher degrees are needed to widen the local functions in order to cover the holes. Polynomials of high degrees, however, are known to produce fluctuations away from the interpolating points. Therefore, when the density of points

varies across the image domain, the local weighted mean may not be a suitable transformation for registration of the images and the global weighted mean is preferred. A comparison of local and global approximation methods in image registration is provided by Goshtasby [150].

The process of finding the transformation functions for the registration of 3-D images is the same as that for the registration of 2-D images except that in the equations, $[(x - x_i)^2 + (y - y_i)^2]$ is replaced with $[(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2]$.

5.6 PIECEWISE LINEAR

If control points in the reference image are triangulated, by knowing the correspondence between the control points in the sensed and reference images, corresponding triangles in the sensed image can be determined. Piecewise methods are those that map corresponding regions in the images to each other. If a linear transformation function is used to map a region in the sensed image to the corresponding region in the reference image, the transformation becomes piecewise linear. The transformation will be continuous but it will not be smooth. When the regions are small or when local geometric difference between the images is small, piecewise linear may produce satisfactory results. However, if local deformations are large, gradients of the transformation at the two sides of a region boundary may be quite different, resulting in an inaccurate registration. Usually triangles are used as regions in piecewise linear.

The choice of triangulation will affect the registration result. As a general rule, elongated triangles with acute angles should be avoided. Algorithms that maximize the minimum angle in triangles is know as Delaunay triangulation [168, 233]. A better approximation accuracy can be achieved if 3-D points obtained from the control points in the reference image and the X or the Y coordinate of corresponding control points in the sensed image are used to create the triangles [30, 350].

In order to provide the same tangent at both sides of a triangle edge, corresponding triangular regions in the images need to be mapped to each other by polynomials of degree two or higher. Various methods for fitting piecewise smooth surfaces over triangles have been developed [59, 65, 74, 344, 357].

Using the coordinates of control points in the reference image and the X or the Y coordinate of the corresponding control points in the sensed image, 3-D triangle meshes are obtained. Fast subdivision techniques have been developed as a means to efficiently fit piecewise smooth surfaces to 3-D triangle meshes. Subdivision methods use corner-cutting rules to produce a limit smooth surface by recursively cutting off corners in the mesh obtained from the points [255, 308, 365]. Subdivision surfaces contain B-spline, piecewise Bézier, and non-uniform B-spline (NURBS) as special cases [348]. Therefore, each component of a transformation can be considered a B-spline, a piecewise Bézier, or a NURBS surface. Loop [247] describes an algorithm for approximating a triangular mesh through recursive subdivision, while Dyn *et al.* [106] describe an algorithm for interpolating a triangular mesh by recursive subdivision. A subdivision-based surface approximation of quadrilateral meshes is described by Catmull and Clark [56], while a surface interpolation of a quadrilateral

mesh is described by Kobbelt [228]. Doo [99], and Doo and Sabin [100] describe a subdivision scheme that can approximate a mesh with triangular, quadrilateral, and in general n-sided faces.

Piecewise linear [147] and piecewise cubic [148] transformation functions have been used in the registration of images with nonlinear geometric difference. The methods are efficient and are easy to implement. However, good accuracy is achieved only within the convex hull of the points where triangles are obtained. For areas outside the convex hull of the points, extrapolation has to be performed, which often results in large errors, especially when higher order polynomials are used to represent the patches.

Table 5.4 shows the results of mapping the grids in Figs 5.1b–f to the grid in Fig. 5.1a with the control points shown in Fig. 5.2 with piecewise linear transformations. Errors are slightly worse than those obtained by the weighted mean method using RaG weights, but they are much better than those for TPS and MQ. To obtain these results, the control points in the reference image were triangulated by Delaunay triangulation as shown in Fig. 5.9. Then, corresponding triangles were obtained in the sensed image and a linear transformation function was found to map a triangle in the sensed image to the corresponding triangle in the reference image. Figure 5.10 shows resampling of Fig. 5.1f to overlay Fig. 5.1a using the control points shown in Fig. 5.2a–d.

Table 5.4 MAX and RMS errors of the piecewise linear transformation registering the images in Fig. 5.1b–f with the image in Fig. 5.1a using the control points in Fig. 5.2.

		Translation	Rotation	Scaling	Linear	Nonlinear
Uniform Dense	MAX	0.0	0.0	0.0	0.0	0.0
	RMS	0.0	0.0	0.0	0.0	0.0
Uniform Sparse	MAX	0.0	1.4	1.4	1.4	15.0
	RMS	0.0	0.5	0.5	0.5	4.0
Random	MAX	0.0	1.4	1.4	1.4	16.1
	RMS	0.0	0.6	0.6	0.6	4.0
Varying Density	MAX	0.0	1.4	1.4	1.4	16.0
	RMS	0.0	0.5	0.4	0.6	5.8

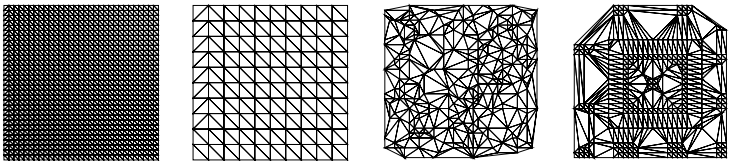


Fig. 5.9 (a)–(d) Delaunay triangulation of the control points in Figs 5.2a–d.

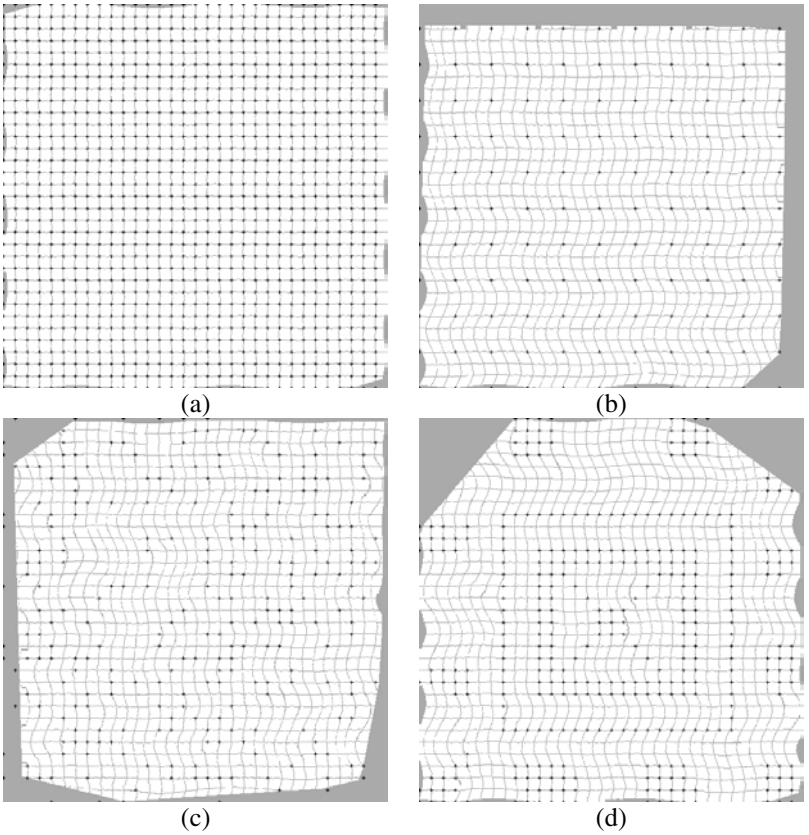


Fig. 5.10 (a)–(d) Piecewise linear resampling of image 5.1f to register with image 5.1a using the control points shown in Figs 5.2a–d.

5.7 WEIGHTED LINEAR

Although two images may have considerable global geometric differences, when compared locally, their geometric differences could be approximated by linear functions. A weighted sum of linear functions should, therefore, be able to register images with nonlinear geometric differences.

A linear transformation that maps a small patch centered at (x_i, y_i) in the reference image to a small patch centered at (X_i, Y_i) in the sensed image has two components. Each component of this linear transformation can be written as

$$f_i(x, y) = a_i x + b_i y + c_i. \tag{5.61}$$

A weighted sum of such linear functions then becomes

$$f(x, y) = \sum_{i=1}^{N} f_i(x, y) b_i(x, y), \tag{5.62}$$

where $b_i(x, y)$ is the ith weight function as described by (5.52), and $f_i(x, y)$ represents the plane passing through (x_i, y_i, f_i). Here, f_i is either X_i or Y_i depending on whether the X- or the Y-component of the transformation is computed. If the linear transformations are not given, each can be determined by fitting a plane to (x_i, y_i, f_i) and $n \geq 2$ other points in its neighborhood.

In 3-D the process is the same except that the planes become hyperplanes, the ith passing through (x_i, y_i, z_i, f_i), for $i = 1, \ldots, N$. Again, f_i is set to X_i, Y_i, or Z_i,

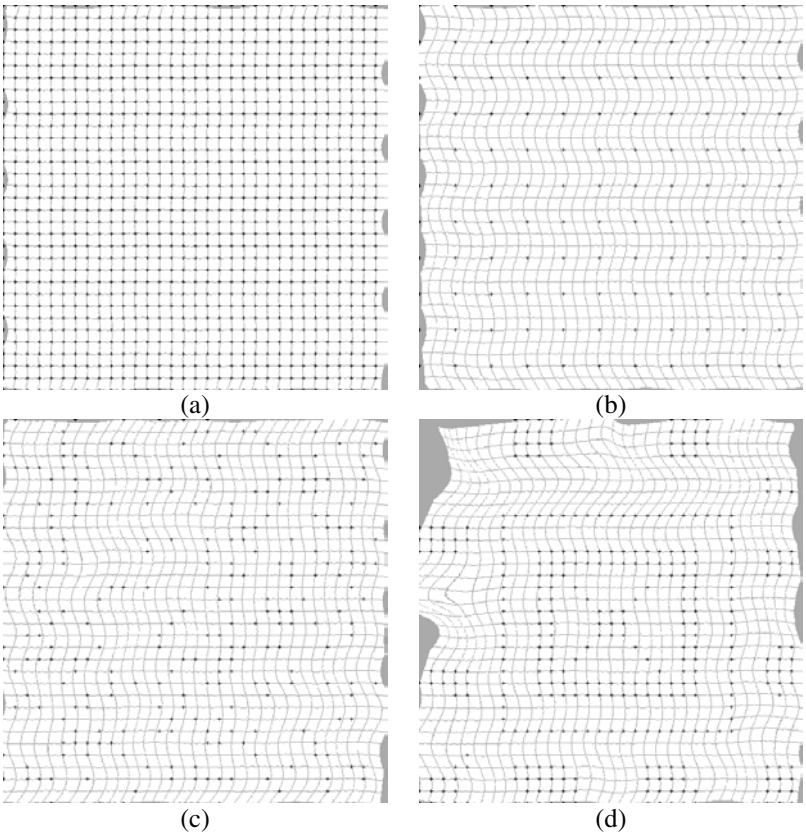


Fig. 5.11 (a)–(d) Resampling of Fig. 5.1f by weighted linear approximation to align with Fig. 5.1a using the control points shown in Figs 5.2a–d.

depending on whether the X-, the Y-, or the Z-component of the transformation is computed.

Examples of image registration by weighted linear transformation are shown in Fig. 5.11. Figures 5.11a–d show resampling of Fig. 5.1f to take the geometry of Fig. 5.1a using the control points shown in Figs 5.2a–d. Results of registering Figs 5.1b–f to Fig. 5.1a using the control points of Figs 5.2a–d are summarized in Table 5.5. Compared to the weighted mean with the same weight functions, we see that, overall, errors have reduced. RaG weights are used and the coefficients of a plane are computed by fitting it to $n = 9$ data points by the least-squares method and σ_i is set equal to the radius of the smallest circle centered at (x_i, y_i) containing 9 control points.

If the density of control points varies across the image domain and a fixed number of correspondences is used to compute the planes, the neighborhood sizes used in the computations will vary from point to point depending on the density of the points within a neighborhood. Assuming that distance of point (x_i, y_i) to the $(n-1)$st closest point is r_i, the standard deviation of the Gaussian associated with the ith point is set to r_i. The larger the value of n, the smoother the transformation will appear. In Table 5.5, the standard deviation of the Gaussian associated with a data point was set to the distance of that data point to the 8th data point closest to it in the xy domain. Neighborhood size should be chosen taking into consideration the noisiness of the point correspondences and the degree of local geometric differences between the images. If the point correspondences are noisy or if local geometric difference between the images is not large, a larger neighborhood size is needed compared to when the point correspondences are accurate or when the images have large local geometric differences.

Note that when a weighted sum of planes as opposed to a weighted sum of points is used, flat horizontal spots will no longer appear in a components of the transformation. This is because the planes are no longer horizontal. Therefore, intermediate parameters u and v are not needed.

Table 5.5 MAX and RMS errors of the weighted linear method using the control points shown in Figs 5.2a–d and the deformations shown in Figs 5.1b–f when compared to Fig. 5.1a. The standard deviation of the Gaussian associated with a control point was set to the distance of that control point to the 8th closest control point in the reference image.

		Translation	Rotation	Scaling	Linear	Nonlinear
Uniform Dense	MAX	0.0	1.0	0.0	1.0	3.6
	RMS	0.0	0.1	0.0	0.1	0.7
Uniform Sparse	MAX	0.0	1.4	1.4	1.4	14.8
	RMS	0.0	0.5	0.5	0.5	4.4
Random	MAX	0.0	1.4	1.4	1.4	16.1
	RMS	0.0	0.6	0.6	0.6	4.0
Varying Density	MAX	0.0	1.4	1.0	1.4	27.0
	RMS	0.0	0.5	0.4	0.6	7.4

5.8 COMPUTATIONAL COMPLEXITY

Assuming the images being registered are of size $n \times n$ and N corresponding control points are available, the time to register the images consists of the time to compute the transformation and the time to resample the sensed image to the coordinate system of the reference image. In the following, the computational complexities of the transformations discussed above are determined and compared.

If the correspondences are known to be accurate, only two corresponding points are sufficient to determine the parameters of a similarity transformation. If the correspondences are known to contain small errors, least-squares is needed to determine the transformation. This requires on the order of N multiplications. If the correspondences are known to contain large errors, first the inaccurate correspondences should be identified and removed. This can be achieved by clustering the parameters of the transformation using combinations of two-point correspondences at a time [369]. Alternatively, combinations of three point correspondences at a time can be used to determine the parameters of linear transformations, and the three correspondences that produce the linear transformation closest to a similarity transformation can be used to define the transformation [96]. Once the translational, rotational, and scaling differences between the images are determined, the sensed image can be resampled to the coordinate system of the reference image, which takes on the order of n^2 operations. Therefore, the computational complexity of the similarity transformation is at least $O(N) + O(n^2)$.

To find the parameters of an affine transformation, the coordinates of at least three correspondences are needed. If the correspondences are known to be accurate, three correspondences are sufficient. Otherwise, all correspondences should be used to determine the parameters by the least-squares method. Once the transformation is obtained, it should be evaluated at each pixel in the reference image to find the corresponding pixel in the sensed image. Therefore, overall, the computational complexity of the affine transformation is $O(N) + O(n^2)$. Computational complexity of the projective transformation is similar to that of affine transformation.

To determine a component of a TPS, a system of $N + 3$ linear equations needs to be solved. This requires on the order of N^2 multiplications. Because TPS is a global method, resampling of a pixel in the sensed image into the space of the reference image requires use of all N correspondences. Therefore, resampling by TPS requires on the order of n^2N operations. Overall, the computational complexity of TPS is $O(N^2) + O(n^2N)$. The computational complexity of MQ and other monotonically increasing radial basis functions is also $O(N^2) + O(n^2N)$. If compactly supported radial basis functions are used, the process still requires the solution of a system of N linear equations, although many of the entries in the matrix of coefficients will be 0. The time needed to resample a sensed image point to the space of the reference image requires only a subset of the points in the neighborhood of the point under consideration. Therefore, the computation complexity of compactly supported radial basis functions is $O(N^2) + O(n^2)$.

In the weighted mean method with RaG weights, the coefficients of the weight functions are the coordinates of corresponding control points, which are known.

Therefore, a transformation is immediately obtained without solving a system of equations. Mapping of each pixel in the sensed image to the space of the reference image takes on the order of N multiplications because theoretically coordinates of all correspondences are used in the calculations. Therefore, overall, the computational complexity of the method is $O(n^2N)$. In practice, however, RaG functions approach zero abruptly and it is sufficient to use only those control points that are in the vicinity of the pixel under consideration. Since digital accuracy is sufficient, use of control points farther than a certain distance from the pixel under consideration will not affect the result. To efficiently choose the control points that affect a pixel, the control points should be binned in a 2-D array so that given a pixel in the sensed image, the control points surrounding it could be determined without examining all the control points. If local weights are used, computation of a resampled point requires use of a very small subset of the control points that are in the vicinity of the pixel under consideration. Therefore, the computational complexity of the local weighted mean is $O(n^2)$.

Piecewise methods first require the time to triangulate the points in the reference image. This requires on the order of $N \log N$ comparisons. Once the triangles are obtained, there is a need to find a transformation for the corresponding triangles. This requires on the order of N operations. Resampling of each point in the sensed image to the space of the reference image takes a small number of operations. Therefore, overall, the computational complexity of piecewise linear is $O(N \log N) + O(n^2)$. Piecewise quadratic, cubic, and higher degree functions require a little more time in each operation, but the number of operations to be performed is the same. Therefore, the computational complexity of piecewise methods is $O(N \log N) + O(n^2)$. Subdivision methods have the same computational complexity but with smaller coefficients.

Finally, to compute a transformation by the weighted linear method, it is required to compute two planes for each control point in the reference image. This requires on the order of N multiplications. Once the planes are determined, the transformation is immediately obtained. Resampling of each pixel in the sensed image to the space of the reference image takes on the order of N operations when RaG weights are used. This computational complexity is $O(n^2N)$. Therefore, the overall computational complexity of the weighted linear method is $O(n^2N)$. However, since Gaussians approach zero exponentially, in practice, it is sufficient to use control points within a small distance of the pixel under consideration to carry out the computations. Therefore, for very small standard deviations, the computational complexity will be $O(n^2) + O(N)$. If rational weights with local support are used, a small subset of the control points that are in the vicinity of the pixel under consideration is used in the calculations. Therefore, the computational complexity of the method is $O(n^2) + O(N)$.

Table 5.6 summarizes the computational complexities of the transformation functions discussed above for 2-D images. In 3-D, similar results are obtained with the exception of replacing n^2 with n^3.

Table 5.6 Computational complexities of the transformation functions when N corresponding control points are used and the images are of size $n \times n$ pixels.

Type of Transformation	Computational Complexity
Similarity	$O(N) + O(n^2)$
Affine and Projective	$O(N) + O(n^2)$
Thin-Plate Splines (TPS)	$O(N^2) + O(n^2N)$
Multiquadrics (MQ)	$O(N^2) + O(n^2N)$
Compactly Supported RBF	$O(N^2) + O(n^2)$
Weighted Mean	$O(n^2N)$
Local Weighted Mean	$O(n^2)$
Piecewise Methods	$O(N \log N) + O(n^2)$
Weighted Linear	$O(n^2N)$
Local Weighted Linear	$O(n^2) + O(N)$

5.9 PROPERTIES OF THE TRANSFORMATION FUNCTIONS

A transformation function contains information about geometric differences between two images. This information is sometimes crucial in understanding the contents of the underlying scene. The presence of sharp geometric differences could be due to the local motion or deformation of the scene. Figures 5.12a and 5.12b show the X and Y components of the transformation used in the registration shown in Fig. 5.11a. The sinusoidal geometric difference between Figs 5.1f and 5.1a are clearly reflected in the components of the transformation.

If some information about geometric differences between two images is known, that information together with the obtained transformation can be used to identify the inaccurate correspondences. For instance, if the images are known to have only linear geometric differences, $f_x(x, y)$ and $f_y(x, y)$ will be planar. However, planes are obtained only when all the correspondences are accurate. Inaccuracies in the correspondences will result in small dents and bumps in the planes. The geometric difference between Figs 5.1a and 5.1f is sinusoidal as demonstrated in Figs 5.12a and 5.12b. This is observed when all the correspondences are correct.

In an experiment, two of the points in Fig. 5.2a were horizontally displaced, two were vertically displaced, and one was displaced both horizontally and vertically by 15 pixels. The displacements are reflected in the obtained transformation as depicted in Figs 5.12c and 5.12d. The dark and bright spots in these image are centered at the points that were displaced. A bright spot shows a positive displacement while a dark spot shows a negative displacement. When displacements are caused by inaccuracies in the correspondences, these small dents and bumps (dark and bright spots) in the components of a transformation identify the inaccurate correspondences.

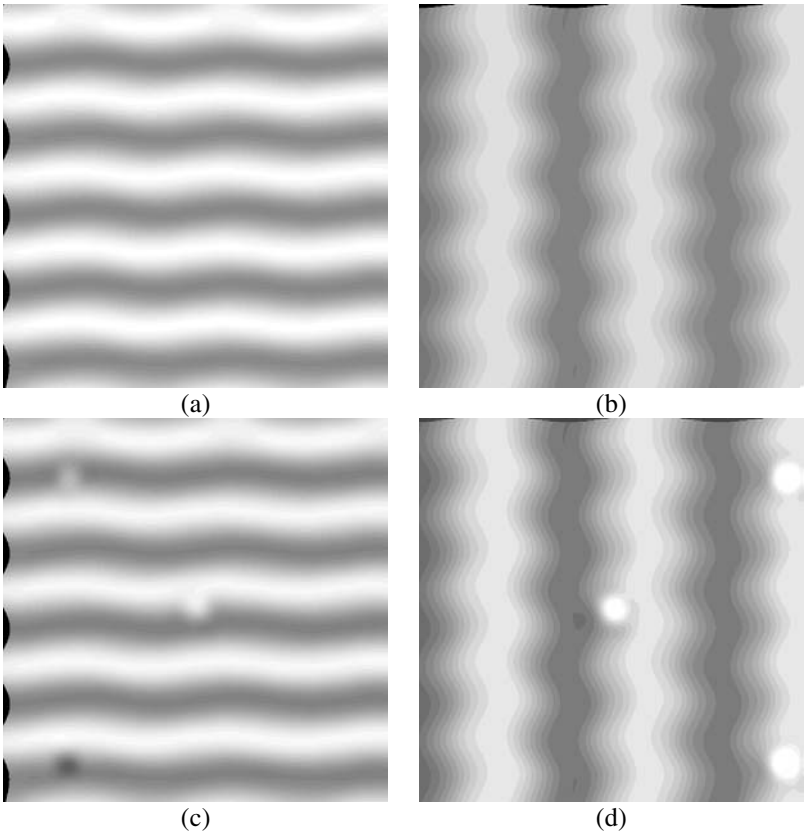


Fig. 5.12 (a), (b) The X and Y components of the transformation for mapping Fig. 5.1f to Fig. 5.1a (shown here are actually $f_x(x, y) - x$ and $f_y(x, y) - y$ and the values are mapped to 0–255 for enhanced viewing). Brighter points show higher functional values. (c), (d) The X and Y components of the transformation after displacing five of the control points in the sensed image. Errors in the correspondences have resulted in small bumps and dents in the surfaces representing the components of the transformation. The bumps and dents are centered at the control points in the reference image that have inaccurate correspondences.

Note that horizontal displacements appear only in the X-component of the transformation and vertical displacements appear only in the Y-component of the transformation. An inaccurate correspondence, therefore, could appear in one or both components of a transformation. A transformation not only provides information about geometric differences between two images, it also contains valuable information about the correctness of the correspondences, which can be used to identify and remove the inaccurate correspondences. If local geometric differences between two images are known to be small, large gradients in the components of a transformation will point to the locations of the incorrect correspondences.

A transformation function determines the relation between all points in two images. Given the coordinates of a point in the reference image, it finds the coordinates of the same point in the sensed image. This information enables the creation of an image flow, quantifying the local deformation or motion of the sensed image with respect to the reference image. This information is very valuable in certain image analysis applications. An example is given in Fig. 5.13.

Figure 5.13a shows MR brain images before (the top row) and after (the middle row) a surgery to remove a brain tumor. The bottom row shows the results of a rigid registration of the images. The misregistration is quite obvious near the ventricles, highlighted within a window of interest. The rigid registration has been unable to correct for the relative movement of brain tissues due to the brain surgery. Performing a nonrigid registration using 10,000 corresponding control points and using

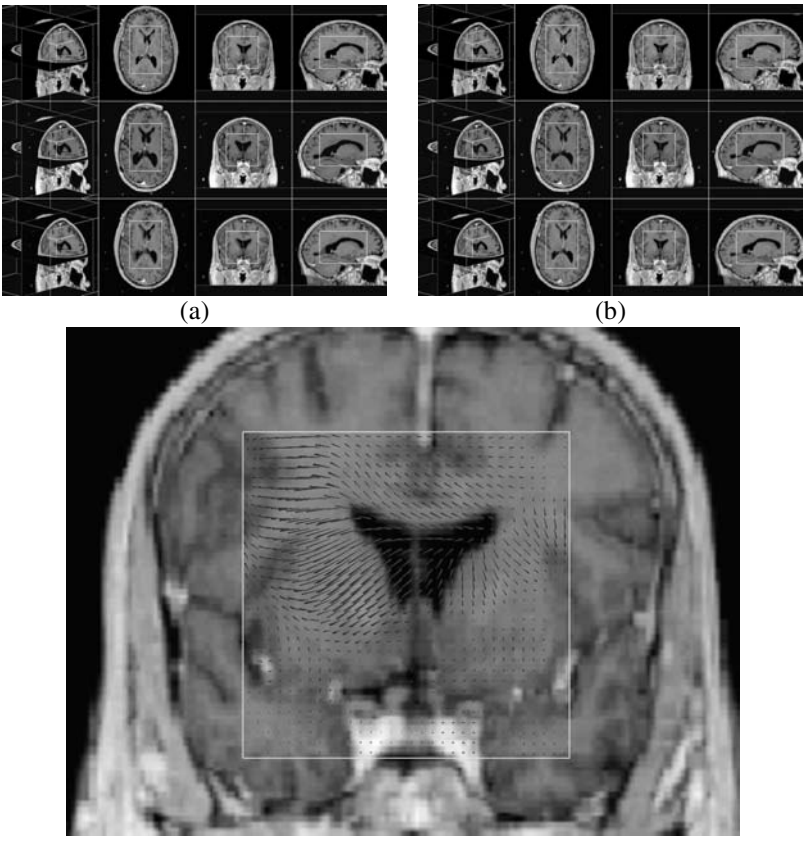


Fig. 5.13 (a) Rigid registration. (b) Nonrigid registration. (c) Flow diagram showing the local motion of voxels in an image slice in the sensed image with respect to the corresponding voxels in the reference image.

the weighted linear with RaG weights as the transformation, the results depicted in Fig. 5.13b are obtained. The nonrigid registration has corrected the relative movement of brain tissues, deforming the sensed image to register the reference image.

A transformation function contains information about the displacement of individual pixels or voxels in the sensed image with respect to the corresponding pixels or voxels in the reference image. Figure 5.13c shows this information when depicted as a flow diagram in one of the image slices. One end of a flow line shows a voxel in the sensed image, and the other end shows the corresponding voxel in the reference image. The flow diagram depicts the brain shift following resection of the sensed (post-surgical) image with respect to the reference (pre-surgical) image.

5.10 SUMMARY

In this chapter, transformation functions for both rigid and nonrigid registration were discussed. If information about the geometric difference between two images is available, that information should be used to select the transformation. For instance, if it is known that the images are obtained by a distant platform of a rather flat scene, an affine transformation is sufficient to register the images. Use of a nonlinear transformation will not only require more computation time, but it may actually worsen the registration accuracy. If no information about the geometric difference between two images is available, a transformation that can adapt to the local geometric differences between the images should be used.

Use of radial basis functions, weighted mean, weighted linear, and piecewise methods in the registration of images with nonlinear geometric differences was discussed. In a set of experiments on various transformations, it was found that radial basis functions, in general, produce the worst accuracy among the methods tested. This is believed to be due to three main reasons. First, the basis functions from which a component of a transformation is determined are radially symmetric and, when spacing between the control points is not symmetric, large errors are obtained in areas where the arrangement of control points is nonuniform. Second, because the basis functions used in TPS or MQ are monotonically increasing, errors in the correspondences or local image deformations spread over the entire resampled image. Also, the addition of a new control point to the set of control points requires recomputation of the transformation and resampling of the entire image. Third, a system of equations has to be solved to find each component of a transformation and, when the number of correspondences is very large, this becomes impractical or impossible. If a small number of widely spread points is available and the local geometric difference between the images is not very large, radial basis functions are effective. This has been demonstrated in the registration of aerial and satellite images [149].

Piecewise methods are attractive because the effect of a control point is limited to a small neighborhood surrounding the point. This property not only makes the computations more stable, it keeps inaccuracies in the correspondences local without spreading them to the entire image domain. Recent subdivision schemes provide efficient means for computing smooth piecewise transformation functions.

Weighted mean and weighted linear methods are preferred over radial basis functions in nonrigid image registration due to four main reasons. First, they do not require the solution of large systems of equations. A component of a transformation is obtained immediately from the given points in weighted mean and only small equations need to be solved in weighted linear. Second, because they do not map corresponding control points exactly to each other, digital errors in the correspondences as well as small mismatch errors are smoothed. Third, the rational weight functions adapt to the density and organization of the control points. The weight functions automatically extend to large gaps between control points and their widths increase or decrease with the spacing between the points. This makes it possible to register images where the density of control points varies greatly in the image domain. Fourth, the width of all weight functions can be controlled globally to control the smoothness or the rigidity of the transformation.

5.11 BIBLIOGRAPHICAL REMARKS

Each component of a transformation function in 2-D is a single-valued function approximating or interpolating a set of irregularly spaced data in the plane. Approximation/interpolation to irregularly spaced data has been studied extensively in Approximation Theory. Excellent surveys of the methods are provided by Schumaker [349], Barnhill [24], Franke [127, 128], Franke and Schumaker [130], Grosse [171], and Weiss *et al.* [406].

Approximation methods can be categorized into:

- mesh-based methods;
- methods using radial basis functions;
- weighted mean methods.

Mesh-based methods subdivide the domain of approximation into polygons and fit a patch over each polygon in such a way that adjacent patches join with a required degree of continuity [30, 59, 74, 307]. These methods work well when the given data are sparse and not noisy. They are completely local and the functional value at a point is obtained from data in a small neighborhood of the point. Fast subdivision algorithms have recently been developed to compute surfaces approximating polygon meshes [81, 174, 255, 308, 318, 365, 396, 433].

Methods using radial basis functions are global; therefore, they use information about all of the data to estimate the value at a point [49, 161, 183, 206, 274]. These methods do not smooth noise and require that the density of the data be rather uniform. Otherwise, high fluctuations may be observed in the obtained surface. If density across a data set varies moderately, Floater and Iske [121] suggest partitioning the data into a hierarchy of smaller sets with increasing density but with rather uniform density at each level. Starting from the set with the lowest density, a surface is obtained. The residuals between the surface and data at the next level are used to determine a residual surface. This process is repeated until a surface is obtained that fits the residuals at the highest density. The surface at each level is defined as the sum of the

surface at the lowest level and the residual surfaces obtained so far. This method is effective when the variation in density across a data set is not very high.

Weighted mean methods [126, 143, 272, 354] work well on dense and noisy data, but they produce horizontal spots at, and in the neighborhood of, the data points when the widths of the weight functions are small. Weighted linear methods do not produce horizontal spots when the widths of the weight functions are small and, thus, are ideal for approximation of irregularly spaced data. Weighted sum of polynomials has been used to approximate irregularly spaced data. Maude [267] fitted a polynomial to a point and $n-1$ of its nearest points and assigned a weight function to the polynomial that vanished at the $(n-1)$st nearest point. Use of gradients in weighted mean methods has been suggested by Franke [127] as a means to eliminate flat spots at interpolating points in a surface. An alternative method proposed by Franke and Nielson [129] finds a weighted sum of bivariate quadratic polynomials, each polynomial obtained by weighted least-squares to approximate data in a small neighborhood. In both methods, local weight functions are used. When the density of the data varies greatly, these methods may produce holes in the approximation.

Also related to the weighted linear method is the Catmull-Rom [57] spline, which is obtained from a blending of polynomials. In the formulation of the Catmull-Rom spline, if RaGs are used as the blending functions and equations of planes are used as the polynomials, the weighted linear method discussed in this chapter will be obtained. Since all polynomials and weight functions extend over the entire image domain, no holes will be obtained in the approximation even when the density of the data, or the spacing between data points, varies greatly across the approximation domain.

6

Resampling

The transformation

$$X = f(x, y) \tag{6.1}$$

$$Y = g(x, y) \tag{6.2}$$

relates coordinates of points in the reference image to the coordinates of the corresponding points in the sensed image. Given the (x, y) coordinates of a point in the reference image, relations (6.1) and (6.2) determine the (X, Y) coordinates of the corresponding point in the sensed image. By reading the intensity at (X, Y) in the sensed image and saving at (x, y) in a new image, the sensed image is point-by-point resampled to the geometry of the reference image. Therefore, to resample the sensed image, the reference image is scanned and, for each pixel location, the corresponding location is determined in the sensed image. Although (x, y) are integers, (X, Y) are floating-point numbers. Since intensities at only integer coordinates are available in the sensed image, the intensity at point (X, Y) has to be estimated from the intensities of a small number of surrounding pixels.

If the sensed image were a continuous image, the intensity of any point (X, Y) in the image would be known. However, a sensed image $I(u, v)$ contains only uniformly spaced samples of a continuous image $C(X, Y)$. A resampling method has to estimate the intensity at (X, Y) from intensities at discrete locations surrounding it. Figure 6.1 depicts the resampling process. Pixel a in the reference image maps to point A in the continuous sensed image. To estimate the intensity at A, intensities in a small neighborhood of A in the discrete sensed image are used. Different methods to achieve this estimation have been developed. In the following sections, nearest-

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

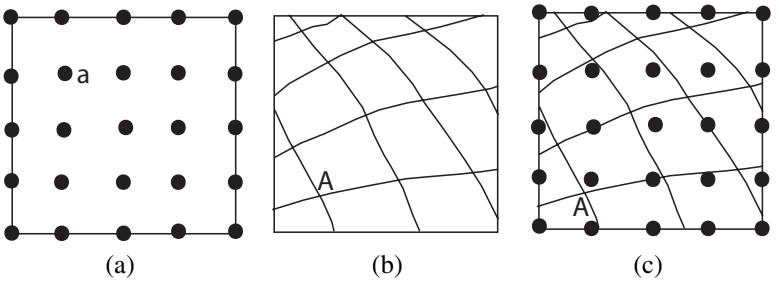


Fig. 6.1 The resampling process. (a) Pixels in the reference image. (b) A continuous sensed image. The grid points in this image correspond to the pixels in the reference image. (c) Overlaying of the continuous and discrete sensed images. The continuous sensed image is determined from intensities in the discrete sensed image. Resampling involves scanning the reference image and, for each pixel a, determining the intensity of the corresponding grid point A in the continuous sensed image. Intensity at A in the continuous image is estimated from the intensities of a small number of pixels surrounding A in the discrete sensed image.

neighbor, bilinear interpolation, cubic convolution, cubic spline interpolation, and radially symmetric resampling methods are discussed.

6.1 NEAREST NEIGHBOR

Given the coordinates (X, Y) of a point, where X and Y are floating-point numbers, and assuming u is the integer part of X and v is the integer part of Y, the rectangular neighborhood defined by pixels $(u, v), (u, v + 1), (u + 1, v), (u + 1, v + 1)$ contain point (X, Y). Among the four pixels, the one closest to (X, Y) is determined and its intensity is used as the intensity at (X, Y). There is actually no need to find the distances to achieve this. If $X - u < 0.5$, u is considered to be the X-coordinate of the pixel to be sampled, otherwise it is considered to be $u+1$ and, if $Y - v < 0.5$, the Y-coordinate of the pixel to be sampled is considered v, otherwise it is considered to be $v + 1$. This is actually a rounding operation. Therefore, the intensity at pixel $[round(X), round(Y)]$ is taken as the intensity at (X, Y).

Nearest neighbor resampling preserves the image intensities. Therefore, the histograms of an image before and after resampling are very similar. If certain intensities do not exist in an image, they will not be produced in the resampled image. The process does not blur an image, but it may produce aliasing effects. Horizontal and vertical edges in an image may appear jagged after rotation of the image by angles that are not a multiple of 90 degrees. Figure 6.2 shows a resampling example by the nearest-neighbor method. Figure 6.2a is an image of a penny and Fig. 6.2b is the penny after being rotated by 10 degrees counterclockwise about the image center and resampled by the nearest-neighbor method. The jagged appearance of the penny, especially along its boundary, is quite evident in the resampled image.

(a) (b)

Fig. 6.2 A penny (a) before and (b) after rotation by 10 degrees counterclockwise and resampling by the nearest-neighbor method.

In 3-D, the intensity at (X, Y, Z) is set to the intensity at $[round(X), round(Y), round(Z)]$. The computational complexity of nearest-neighbor resampling in 2-D is on the order of n^2 comparisons for an image of size $n \times n$ pixels and in 3-D is on the order of n^3 comparisons for an image of size $n \times n \times n$ voxels.

6.2 BILINEAR INTERPOLATION

In bilinear interpolation, the intensity at a point is determined from the weighted sum of intensities at four pixels closest to it. Therefore, given location (X, Y) and assuming u is the integer part of X and v is the integer part of Y, the intensity at (X, Y) is estimated from the intensities at $(u, v), (u + 1, v), (u, v + 1), (u + 1, v + 1)$. This resampling involves first finding the intensity at (X, v) from the linear interpolation of intensities at (u, v) and $(u+1, v)$. Let this intensity be $I(X, v)$. Then, the intensity at $(X, v + 1)$ is determined from the linear interpolation of intensities at $(u, v + 1)$ and $(u + 1, v + 1)$. Let this intensity be $I(X, v + 1)$. Then, the intensity at (X, Y) is computed from the linear interpolation of intensities at (X, v) and $(X, v + 1)$. This can be summarized as

$$\begin{aligned} I(X,Y) = {} & W_{u,v}I(u,v) + W_{u+1,v}I(u+1,v) + W_{u,v+1}I(u,v+1) \\ & + W_{u+1,v+1}I(u+1,v+1), \end{aligned} \tag{6.3}$$

where

$$W_{u,v} = (u + 1 - x)(v + 1 - y), \tag{6.4}$$

$$W_{u+1,v} = (x-u)(v+1-y), \tag{6.5}$$

$$W_{u,v+1} = (u+1-x)(y-v), \tag{6.6}$$

$$W_{u+1,v+1} = (x-u)(y-v), \tag{6.7}$$

and $I(u,v)$, $I(u+1,v)$, $I(u,v+1,)$, and $I(u+1,v+1)$ are intensities at (u,v), $(u+1,v)$, $(u,v+1)$, and $(u+1,v+1)$, respectively. Figure 6.3a depicts this estimation process. Fig. 6.3b is obtained by rotating Fig. 6.2a counterclockwise by 10 degrees about the image center and resampling it by bilinear interpolation. Compared to Fig. 6.2b, we see that the aliasing effect has mostly disappeared. One should also note that this anti-aliasing is at the cost of blurring the image. Therefore, if repeated resampling of an image is required, the process will gradually smooth image details. Bilinear interpolation changes image intensities and, therefore, the histogram of the image changes.

In 3-D, the intensity at point (X,Y,Z) is estimated from the trilinear interpolation of intensities of the $2\times2\times2$ neighborhood containing the point. First, linear interpolation is carried out along slices to produce intensities at the desired Z plane. Then, linear interpolation is carried out along the rows to estimate the intensities at column X. Finally, linear interpolation is carried out along column X to find the intensity at (X,Y,Z).

Computationally, 2-D resampling by bilinear interpolation requires on the order of n^2 multiplications for an image of size $n\times n$ pixels, and 3-D resampling requires on the order of n^3 multiplications for an image of size $n\times n\times n$ voxels. Therefore, nearest-neighbor and bilinear interpolation have the same computational complexity, although nearest-neighbor is several times faster than bilinear interpolation.

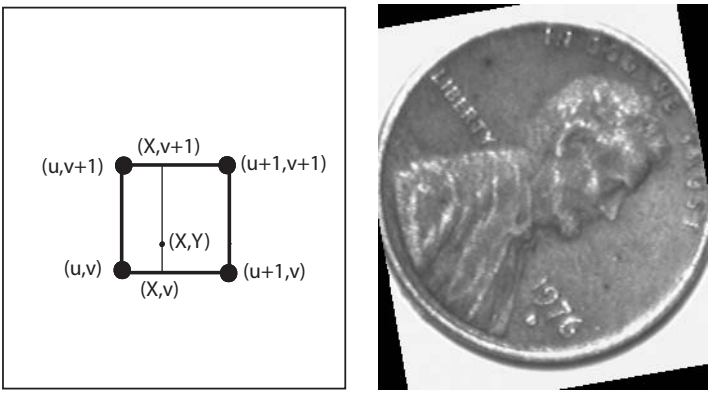


Fig. 6.3 (a) Estimating the intensity at (X,Y) from intensities at (u,v), $(u+1,v)$, $(u,v+1)$, and $(u+1,v+1)$ by bilinear interpolation. (b) The penny shown in Fig 6.2a after rotation by 10 degrees counterclockwise about the image center and resampling by bilinear interpolation.

6.3 CUBIC CONVOLUTION

In cubic convolution, the intensity at point (X, Y) is estimated from the intensities of the 16 pixels closest to it as depicted in Fig. 6.4a. Just like a separable 2-D convolution that can be performed via 1-D convolutions row-by-row and then column-by-column, interpolation by cubic convolution can be carried out via 1-D interpolation along the rows and then separately along columns. Therefore, first, 1-D interpolation is carried out along the four image rows in the 4×4 neighborhood of (X, Y). This will determine values at $(X, v-1), (X, v), (X, v+1), (X, v+2)$. Then interpolation is carried out along column X to find the intensity at (X, Y).

If u denotes the integer part of Xin a 1-D image, and assuming intensities at $u-1, u, u+1$, and $u+2$ are $I(u-1), I(u), I(u+1)$, and $I(u+2)$, respectively, a function $f(t)$ that interpolates the four intensities should satisfy $f(t_i) = I(t_i)$, where t_i is $u-1, u, u+1$, or $u+2$. To ensure that the function approaches zero two pixels away from the point under consideration, function f is defined by a weighted sum of four local functions, with the weights representing the intensities at the pixels. That is,

$$f(X) = I(u-1)f_{-1} + I(u)f_0 + I(u+1)f_1 + I(u+2)f_2, \tag{6.8}$$

where

$$f_{-1} = -\frac{1}{2}t^3 + t^2 - \frac{1}{2}t, \tag{6.9}$$

$$f_0 = \frac{3}{2}t^3 - \frac{5}{2}t^2 + 1, \tag{6.10}$$

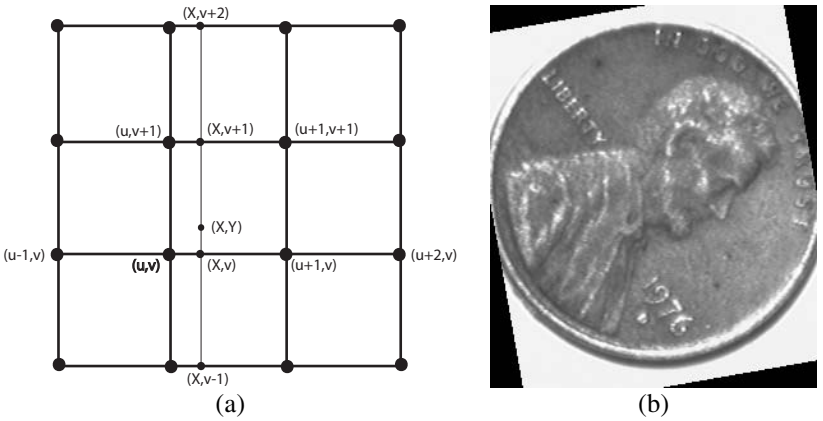


Fig. 6.4 (a) Cubic convolution uses intensities in a 4×4 neighborhood to estimate the intensity at a point. To determine the intensity at (X, Y), first 1-D interpolation is carried out along the 4 rows to determine intensities at X in each row. Then 1-D interpolation is carried out along column X to determine the intensity at (X, Y). (b) Resampling of Fig. 6.2a by cubic convolution.

$$f_1 = -\frac{3}{2}t^3 + 2t^2 + \frac{1}{2}t, \tag{6.11}$$

$$f_2 = \frac{1}{2}t^3 - \frac{1}{2}t^2, \tag{6.12}$$

and $t = X - u$. Note that the sum of the local functions for any t in the range from 0 to 1 is 1, $f(X)$ evaluates to $I(u)$ when $X = u$ and it evaluates to $I(u+1)$ when $X = u+1$. In a 1-D image with n pixels, the pixel positions are $u = 0, 1, \ldots, n-1$ with corresponding intensities $I(u)$. At the two image borders it is assumed that $I(-1) = 3I(0) - 3I(1) + I(2)$ and $I(n) = 3I(n-1) - 3I(n-2) + I(n-3)$. Cubic convolution in 2-D involves cubic convolution in 1-D first along the rows to find the values at the desired column and then interpolation along the obtained column. Figure 6.4a depicts this computation. An example of image resampling by cubic convolution is given in Fig. 6.4b.

Although resampling by cubic convolution appears visually similar to that of bilinear interpolation, by a closer examination it becomes apparent that intensities estimated by the two methods are not the same. An example is given in Fig. 6.5. The image of Fig. 6.2a was rotated by 10 degrees 36 times. This is expected to produce the original image if no resampling errors existed. However, due to resampling errors, the obtained image is different from the original one. The absolute difference between intensities of the obtained image and the intensities of the original image show the magnitude of resampling error.

Figures 6.5a–c show absolute errors when nearest-neighbor, bilinear interpolation, and cubic convolution are, respectively, used. Only errors within the circle touching the image borders can be computed. As the image is rotated areas near image corners move outside the image, causing them to be cut off. Consequently, after 360-degree rotation, only pixels within the largest circle that can be contained in the image remain. Pixels outside this circle will all assume value zero. Therefore, when

(a) (b) (c)

Fig. 6.5 (a)–(c) Resampling errors when rotating the image in Fig. 6.2a 36 times with 10-degree increments and computing the absolute different between corresponding pixels in the original and resampled images using nearest-neighbor, bilinear interpolation, and cubic convolution, respectively. Higher intensities show larger errors.

subtracting the resampled image from the original image, intensities of pixels outside this circle become equal to the intensities of the original image. This experiment reveals that among the three methods, cubic convolution is the most accurate method while nearest-neighbor is the least accurate method.

Cubic convolution at (X, Y, Z) in a volumetric image involves carrying out interpolation along slices to obtain intensities of pixels at plane Z, then carrying out cubic convolution in the obtained 2-D image to find the intensity at (X, Y) within plane Z.

Computational complexity of cubic convolution in 2-D is $O(n^2)$ multiplications for an image of size $n \times n$ pixels, and in 3-D it is $O(n^3)$ multiplications for an image of size $n \times n \times n$ voxels. Cubic convolution in 2-D is several times slower than bilinear interpolation.

6.4 CUBIC SPLINE

Given intensities $\{I_i : i = -1, 0, 1, 2\}$ of pixels $\{u_i = i : i = -1, 0, 1, 2\}$ in a 1-D image, the intensity at point $0 \leq u < 1$ in the image can be estimated using a B-spline curve of order four (degree three) from:

$$f(u) = \sum_{i=-1}^{2} I_i b_i(u), \tag{6.13}$$

where

$$b_{-1}(u) = (-u^3 + 3u^2 - 3u + 1)/6, \tag{6.14}$$

$$b_0(u) = (3u^3 - 6u^2 + 4)/6, \tag{6.15}$$

$$b_1(u) = (-3u^3 + 3u^2 + 3u + 1)/6, \tag{6.16}$$

$$b_2(u) = u^3/6, \tag{6.17}$$

are the B-spline basis functions of order 4 and I_is are the control points of the curve. Note that the sum of the B-spline basis functions is 1 everywhere in the range $0 \leq u \leq 1$. The basis functions, when evaluated at a particular point u, show the contributions of the control points in the computation of the intensity at that point. The intensity at point u in a 1-D image is, therefore, a weighted sum of the intensities of the four pixels closest to it.

Formula (6.13) shows an approximating curve, and thus $f(u)$ will not evaluate to the intensities at the pixels. In order for the curve to interpolate the intensities, it is required to find new intensities such that, when they are used as the control points, the obtained B-spline curve will evaluate to the intensities at the pixels. That is, $\{I_j^{'} : j = -1, \ldots, 2\}$ should be determined such that

$$I_i = \sum_{j=-1}^{2} I_j^{'} b_j(u_i), \qquad i = -1 \ldots, 2. \tag{6.18}$$

Since two adjacent B-spline segments of order four share three control points (pixel intensities in our case), we cannot determine the control points of an interpolating B-spline by repeated use of formula (6.18). Instead, it is required to determine the entire set of control points collectively. This would require the solution of a system of n equations if a 1-D image of size n pixels is given.

In 2-D, estimation of the intensity at point (u, v) requires use of intensities at 16 pixels (the 4×4 neighborhood) of the point. To make this possible, first, new intensities should be determined so that when used as the control points in a bicubic B-spline surface, the obtained surface will interpolate the given intensities. Computation of the control points of a bicubic B-spline surface interpolating the intensities in an $n \times n$ image requires the solution of a system of n^2 equations, which involves on the order of n^4 multiplications. A parallel algorithm to calculate the control points of an interpolating B-spline surface with a smaller computational complexity is given by Cheng *et al.* [63].

Noticing that a B-spline curve acts like a digital filter when intensities of uniformly spaced pixels in a 1-D image are used as its control points; the control points of the B-spline curve interpolating the intensities can be obtained through an inverse filtering operation [158]. This involves finding the Fourier transform of the image and dividing it point-by-point by the Fourier transform of the B-spline filter. The inverse Fourier transform of the result will produce the control points of the interpolating B-spline. If FFT algorithm is used to compute the Fourier transform coefficients, it will take on the order of $n \log n$ multiplications to find the control points of the interpolating B-spline curve. This is a reduction in computation time from n^2 to $n \log n$ multiplications. Since the basis functions for B-spline surfaces can be separated into basis functions for curves, to obtain an interpolating surface, first, inverse filtering is performed along the image rows and then along the image columns. This would require on the order of $n^2 \log n$ multiplications.

In 3-D, the system of equations to be solved becomes very large even for very small images; therefore, inverse filtering is the method of choice to find the control points of the interpolating B-spline. Since B-spline volumes have separable basis functions, inverse filtering can be carried out first along the image slices, then along the image rows, and finally along the image columns to obtain the interpolating B-spline volume. The computational complexity of resampling by B-splines in 3-D when using inverse-filtering is $O(n^3 \log n)$ multiplications.

The cubic splines method is computationally more expensive than cubic convolution, but it has been found to produce more accurate results [223].

6.5 RADIALLY SYMMETRIC KERNELS

Cubic spline, cubic convolution, and bilinear interpolation are not radially symmetric functions. In general, locally defined separable functions are not radially symmetric because they approximate rectangular domains. The effect of such functions on an image depends on the orientation of the image. For a resampling method to be independent of an image's orientation, the function used in resampling should be

radially symmetric. Consider centering the radial function,

$$f(r_i) = \begin{cases} 1 - 3r_i^2 + 2r_i^3, & 0 \leq r_i \leq 1, \\ 0, & r_i > 1, \end{cases} \tag{6.19}$$

at pixel (X_i, Y_i), where

$$r_i = \frac{\sqrt{(X - X_i)^2 + (Y - Y_i)^2}}{a} \tag{6.20}$$

and a is the radius of the domain where the function is nonzero (see Fig. 6.6a). The function $f(r_i)$ will then show the influence of the pixel centered at (X_i, Y_i) on point (X, Y). Note that since $df(r_i)/dr_i = 0$ at $r_i = 1$, not only does $f(r_i)$ vanish at $r_i = 1$, it vanishes smoothly. Therefore, if radial function f is centered at each pixel and a weighted sum of the functions is computed, the obtained surface will be smooth everywhere in the image domain. For the surface to interpolate the intensities, the weights of the functions should be computed by solving a linear system of equations. The weights can be considered new image intensities. Then, the intensity at (X, Y) can be computed from the newly obtained intensities falling within the circular area of radius a centered at (X, Y) (see Fig. 6.6b). That is,

$$I(X, Y) = \sum_{r_i < a} I_i' f(r_i) \tag{6.21}$$

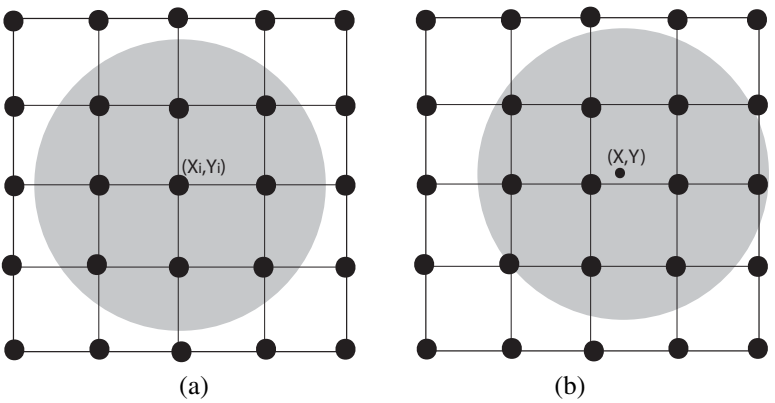


Fig. 6.6 (a) A radial function with local support of radius 1.8 pixels centered at pixel (X_i, Y_i). This function keeps the influence of pixel (X_i, Y_i) on the interpolating function within the shaded area shown. (b) The intensity at (X, Y) is estimated from the intensities of pixels whose distances to (X, Y) are less than a. The pixels used in the calculation are those falling in the shaded area in this figure when $a = 1.8$ pixels.

where $I_i^{'}$ is the weight of the radial function centered at (X_i, Y_i). Only pixels within the circular area of radius a centered at (X, Y) as depicted in Fig. 6.6b are needed to calculate the intensity at (X, Y).

Global radial functions, such as Gaussians, which approach zero exponentially, can be used for this purpose also. When a Gaussian is used, computation of new image intensities, which after convolution with the Gaussian produces the original intensities, can be achieved by inverse filtering. Given intensities $\{I(i, j) : i = 0, \ldots, m-1; j = 0, \ldots, n-1\}$, first, new intensities $\{I'(i, j) : i = 0, \ldots, m-1; j = 0, \ldots, n-1\}$ are computed by inverse filtering. The standard deviation of the Gaussian used in inverse filtering should be a small number such as 1 pixel to make the process local. Knowing the intensities of the inversely filtered image, the intensity at point (X, Y) in the original image can be computed by centering the Gaussian at each pixel in the circular neighborhood centered at (X, Y) and evaluating the weighted sum of the Gaussians, the weights being the $I_i^{'}$s. A circular neighborhood of radius 3 standard deviations centered at (X, Y) is sufficient to find the value at (X, Y). When $\sigma = 1$ pixel, circular neighborhoods of radius larger than 3 pixels will not change the resampled intensities because the intensities have to be quantized and accuracy beyond a decimal place will not change a computed intensity.

Computationally, radial functions with local support require the solution of a system of n^2 equations for an image of size $n \times n$ pixels. This requires on the order of n^4 multiplications, which is prohibitively large. If Gaussians are used instead, computation of image intensities by inverse filtering requires on the order of $n^2 \log n$ multiplications.

Radial functions in 3-D with local support will be nonzero within a spherical region. Therefore, we will have:

$$r_i = \frac{\sqrt{(X - X_i)^2 + (Y - Y_i)^2 + (Z - Z_i)^2}}{a} \tag{6.22}$$

If Gaussians are used, since a 3-D Gaussian can be separated into three 1-D Gaussians, computation of new image intensities from which resampled intensities are calculated would require inverse filtering along image slices, then along image rows, and finally along image columns. This requires on the order of $n^3 \log n$ multiplications. Again, the standard deviation of the Gaussian used in the calculations should be small, such as 1 pixel. A larger Gaussian produces very small high-spatial frequency coefficients. Since, in inverse filtering, the Fourier transform of the image is point-by-point divided by the Fourier transform of the filter, very small Fourier transform coefficients in the denominator will result in large errors in highly detailed areas in an image. For a very sharp or highly detailed image, a small standard deviation, such as 0.7 pixels, should be used. If radial functions with local support are used and a is large, the system of equations to be solved may become ill-conditioned or singular, making resampling impossible. For a very detailed image, the neighborhood size used in the calculations should have a rather small radius, such as $a = 0.7$ pixels.

6.6 SUMMARY

Nearest-neighbor, bilinear interpolation, cubic convolution, cubic spline, and radially symmetric kernels for image resampling were discussed. The computational complexities of these methods are summarized in Table 6.1. Nearest-neighbor is the fastest method and radially symmetric functions are the slowest.

Nearest-neighbor resampling does not change the image intensities, it only displaces them. Since nearest-neighbor resampling preserves image intensities, it preserves sharpness in an image. However, this sharpness could be the source of aliasing. Bilinear interpolation, only slightly smoothes an image, but that is sufficient to reduce aliasing effects considerably. Among all resampling methods, bilinear interpolation is perhaps the best compromise between speed and accuracy. Cubic convolution requires several times more computation time than bilinear interpolation, but it produces more accurate results. Cubic spline is considerably slower than cubic convolution, especially when the control points of the spline are determined by solving a system of equations, but resampling accuracy is found to be better than that of cubic convolution [223].

In this chapter, use of radially symmetric kernels in image resampling was also discussed. No experimental results on this resampling method are available but, considering the fact that the neighborhood size used in this method is adjustable, this resampling can adjust to the smoothness/detailedness of an image. Cubic spline uses a fixed neighborhood size and, if the image being resampled is very sharp, the matrix of coefficients may become singular. If radial functions with local or global support are used, since radius a of the local functions or standard deviation σ of the Gaussians can be reduced as desired, the resampling can be made to succeed even on very sharp images.

Table 6.1 Computational complexities of nearest-neighbor, bilinear, cubic convolution, cubic spline, radial functions with local and global supports in image resampling. The image being resampled is assumed to be of size $n \times n$ pixels.

Type of Resampling	Computational Complexity
Nearest-Neighbor	$O(n^2)$
Bilinear Interpolation	$O(n^2)$
Cubic Convolution	$O(n^2)$
Cubic Spline, Direct Computation	$O(n^4)$
Cubic Spline, Using FFT	$O(n^2 \log n)$
Radial Functions with Local Support	$O(n^4)$
Gaussian, Using FFT	$O(n^2 \log n)$

6.7 BIBLIOGRAPHICAL REMARKS

Use of cubic convolution in image resampling is discussed by Key [223], while use of interpolating cubic B-spline in image resampling is described by Hou and Andrews [195]. Properties of B-splines as filters have been explored by Ferrari *et al.* [116], and a fast algorithm to find interpolating B-spline surfaces is described by Cheng *et al.* [63]. Computation of the control points of interpolating B-splines by inverse filtering is outlined by Goshtasby *et al.* [158].

7

Performance Evaluation

Common criteria for the evaluation of image registration performance are *accuracy, reliability, robustness,* and *computational complexity* [137, 208]. *Accuracy* refers to the difference between true and estimated values. The smaller this difference, the more accurate the estimate will be. In image registration, accuracy refers to the mean, median, maximum, or root-mean-squared distance between points in the reference image and corresponding points in the sensed image after they have been resampled to the space of the reference image. Accuracy can be measured using synthetic or simulation images where the coordinates of true correspondences are known. Alternatively, fiducial markers may be placed in the scene and the locations of the fiducials may be used to evaluate the registration accuracy. Accuracy is measured in pixels/voxels.

Reliability refers to the number of times an algorithm succeeds in finding a satisfactory answer compared to the total number of tests performed. In image registration, if n pairs of images are tested by an algorithm and m out of n tests produce satisfactory results and, if n is sufficiently large and the images are representative of images in the input, then m/n represents the reliability of the algorithm. The closer this ratio is to 1, the more reliable the algorithm will be. The behavior of a reliable algorithm is predictable.

Robustness refers to the degree of stability of the accuracy or the reliability of an algorithm under variations in one or more of its input parameters. For instance, if an algorithm can register a pre-operative image with an intra-operative image containing surgical instruments that hide some tissues, the algorithm is considered robust with respect to occlusion. Robustness can be measured with respect to noise, intensity/geometric difference between images, percentage of dissimilar regions in images, etc. The robustness of an algorithm can be determined by finding the degree of sta-

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

bility of the accuracy or the reliability of the algorithm as one or more of its input parameters is changed. The degree of stability of accuracy or reliability can be quantified using the standard deviation of the accuracy or reliability as an input parameter is varied. The smaller this standard deviation, the more robust the algorithm will be. If there are many input parameters, each affecting the accuracy or reliability of the algorithm, the robustness of the algorithm with respect to each input parameter can be determined. An algorithm may be robust with respect to noise but not with respect to the geometric difference between images. Saying that an algorithm is, in general, robust implies that the performance of the algorithm does not change noticeably as all its input parameters are changed. An algorithm that has a consistently low accuracy or reliability is not considered robust. It should have a consistently high accuracy or reliability to be robust.

Computational complexity determines the speed of an algorithm and shows the practicality of the algorithm in real situations. A registration algorithm that requires a few hours to register two brain images is not of much value in image-guided neurosurgery. To be useful, it should be able to register a pre-operative image that is used to plan the surgery with an intra-operative image that shows the state of the surgery at a particular time in a few seconds. An algorithm that registers the map of a region to aerial images of the region captured by a moving platform is useful for scene intervention if the images can be registered in milliseconds. Computational complexity is measured in number of additions/multiplications expressed as a function of the image dimensions. A registration algorithm is desired to have a computational complexity that is a linear function of the image dimensions.

Most image registration algorithms are composed of three components:

- feature selection;
- feature correspondence;
- transformation function.

The performance of an image registration algorithm is dependent on the performances of its components. In the following sections, first, performances of individual components of an image registration algorithm are discussed. Then, the performance of a registration algorithm from the performances of its components is discussed.

7.1 FEATURE SELECTION PERFORMANCE

In Chapter 3, algorithms for selecting points, lines, regions, and templates from images were discussed. Points are the most widely used features in image registration. Certain points represent meaningful landmarks in images. For instance, if a corner detector identifies road intersections, distances between true road intersections and the detected ones measure the accuracy of the corner detector. If the corner detector of Algorithm 3.1 is applied to image of Fig. 7.1a with the Gaussian smoother of standard deviation 1 pixel, the corners shown in Fig. 7.1b are obtained. Let's suppose the objective is to find points where lines or edges meet. We see that some of the detected corners are displaced from their true positions. The maximum distance

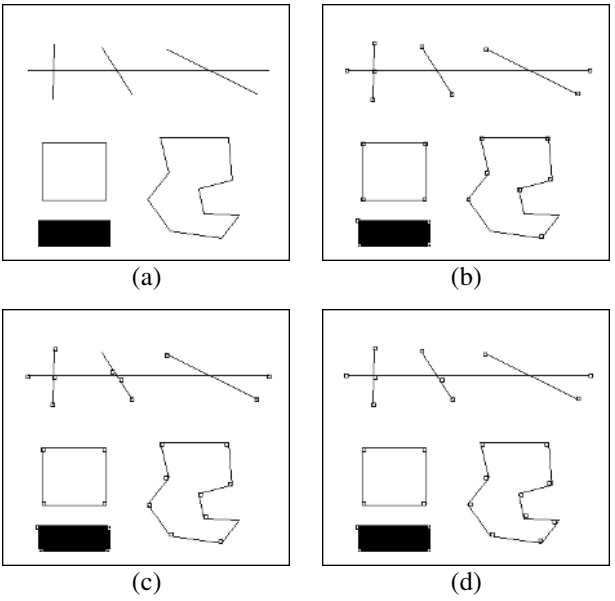


Fig. 7.1 (a) A synthetic image. (b)–(d) Corners detected by Algorithm 3.1 when standard deviation of the Gaussian smoother was 1, 2, and 3 pixels, respectively.

between the true and detected corners is 1.4 pixels and the average distance between true and estimated corners is 0.6 pixels. These measures characterize the accuracy of Algorithm 3.1 in detecting corners in the image of Fig. 7.1a.

The percentage of true corners detected in an image defines the reliability of the algorithm. If an algorithm finds all corners, but the detected corners are displaced with respect to the true corners, that algorithm is highly reliable but has a poor accuracy. On the other hand, if an algorithm finds a small number of corners but the detected corners are very close to the true ones, that algorithm is highly accurate but is not very reliable. In Fig. 7.1b, there are 21 true corners among which 16 have been found. Therefore, the reliability of Algorithm 3.1 in detecting corners in Fig. 7.1a is 16/21. The algorithm has also classified line end points as corners. If the objective was to find locally unique points, the algorithm would have had reliability 24/29 as line end points and line intersections are both locally unique image features. But if only points where lines or edges meet are considered corners, then the algorithm has reliability 16/21 for the image in Fig. 7.1a. In order to speak about the reliability of an algorithm in absolute terms, it is required to perform tests on a sufficiently large number of images that are representative of the kind of images the algorithm may encounter. The number of corners detected over the total number of corners in the images defines the reliability of the algorithm.

The reliability measure computed in this manner does not take into consideration the number of falsely detected corners. This will be fine if false corners do not

harm a registration. However, if false corners are harmful to a registration algorithm, the number of false corners should also be taken as a measure when describing the reliability of an algorithm. In Fig. 7.1b, 5 true corners are missed while 8 false corners are detected. Therefore, false-negative probability is 5/21 and false-positive probability is 8/21. A very reliable algorithm will have a very small false-negative probability and a very small false-positive probability.

If the behavior of an algorithm remains stable when it is tested on various images, the algorithm becomes robust. For instance, a robust corner detector finds the same set of corners independent of the orientation, scale, and a reasonable amount of noise in input images. Robustness of a corner detector can be characterized by plotting the accuracy or reliability of the detector against variation in one or more of its input parameters. Figures 7.1b–d show corners detected when the standard deviation of the Gaussian smoother in Algorithm 3.1 was 1, 2, and 3 pixels, respectively. This can be considered the change in the resolution of the input. The numbers of correctly detected corners in these images are 16, 18, 20. The reliability of the algorithm slightly increases as the standard deviation of the Gaussian smoother is increased from 1 to 3 pixels. This increased reliability is, however, at the cost of decreased accuracy. A wider Gaussian reduces noise but it makes the corners rounder, shifting their positions. Therefore, corners obtained at lower resolutions are not localized well. Thus, they are less accurate than corners detected at higher resolutions.

Images obtained by changing other input parameters are shown in Fig. 7.2. Figure 7.2a shows the rotation of the image of Fig. 7.1a by 90 degrees and the determination of its corners. 17 corners have been found, of which 14 are the same as those found in Fig. 7.1b. Figure 7.2b shows the rotation of Fig. 7.1a by 12 degrees and the determination of its corners. 15 out of the 17 corners are the same as those detected in Fig. 7.1b. Figure 7.2c shows the scaling of Fig. 7.1a by 0.8 and the determination of its corners. 15 out of the 17 corners are the same as those detected in Fig. 7.1b. Finally, Fig. 7.2d shows the scaling of Fig. 7.1a by 1.2 and the determination of its corners. 14 out of the 17 corners are the same as those detected in Fig. 7.1b. In all these cases, the standard deviation of the Gaussian smoother in Algorithm 3.1 was 1 pixel. The reliability of the algorithm varies only slightly with respect to variations in these input parameters; therefore, the algorithm can be considered relatively robust when tested on the images shown in Figs 7.1 and 7.2. In order to make a general statement about the robustness of this corner detector, it is required to test it on a sufficiently large number of images that represent possible inputs to the detector.

Compared to point features, which have only position, line features have position, orientation, and length. The position of a line feature is considered to be the position of its midpoint. The performance of a line detector is determined from the performances of its attributes.

Regions have one purpose to serve in image registration and that is to provide unique control points with subpixel positional accuracy. Since the centroid of a region is obtained from the average of pixels defining the region, random noise on the region boundary cancels, producing highly accurate center coordinates. Unlike random noise, inaccuracy in region shape will affect the accuracy of the center coordinates.

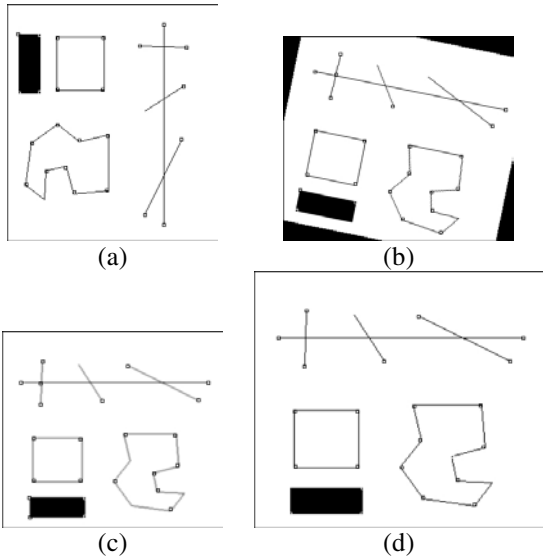


Fig. 7.2 Corners detected in the image of Fig. 7.1a after (a) rotation by 90 degrees clockwise, (b) rotation by 12 degrees clockwise, (c) scaling by 0.9, and (d) scaling by 1.2 using Algorithm 3.1 with a Gaussian smoother of standard deviation 1 pixel.

When a segment of a region is missing, the centroid of the region displaces by an amount proportional to the size of the missing segment.

The objective of template selection and template matching in image registration is to provide a means to establish the correspondence between control points in two images. By selecting templates in the reference image and locating their correspondences in the sensed image, a set of corresponding points representing the centroids of corresponding regions is obtained. For the correspondence process to succeed, it is required that the templates be highly informative and unique. The accuracy, reliability, and robustness of a template selection algorithm can be determined in the same manner that the accuracy, reliability, and robustness of a corner detector was determined.

In order to determine the performance of a feature detector, it is required to know the true positions of features in a set of representative images. True positions of image features can be determined through interactive means. Accuracy is computed from the positional difference between features selected interactively and features detected automatically. Reliability is the percentage of features the algorithm correctly detects. Robustness represents the behavior of the algorithm under variations in one or more of its input parameters such as image orientation, scale, and noise. A feature detector may be robust with respect to the variation in one input parameter but not with respect to variation in another input parameter. For example, the behavior of a feature detector may remain relatively unchanged under variation in the orientation of an image but it may drastically change with variation in the scale of the image.

7.2 FEATURE CORRESPONDENCE PERFORMANCE

In Chapter 4, various algorithms for determining the correspondence between features in two images were discussed. Point pattern matching algorithms use information about the positions of the points to find the correspondences. Line-matching algorithms use information about the orientation as well as the position of the lines to determine the correspondences. Region-matching algorithms use positions as well as the shapes of the regions to find the correspondences, and template matching algorithms use information within small circular neighborhoods in images to find the correspondences.

A feature-matching algorithm finds a correspondence either correctly or incorrectly. It, therefore, does not make sense to talk about the accuracy of a feature matching algorithm. The reliability of such algorithms can be considered to be the number of correct correspondences found over the total number of correspondences existing in the tested images.

A single wrong correspondence could force a whole registration to fail. Therefore, in addition to the true-positive probability, false-positive probability should be used to describe the reliability of a feature correspondence algorithm. Assuming M features are present in the reference image and N features are present in the sensed image and n is the smaller of M and N, then, if an algorithm finds m_1 correct correspondences and m_2 wrong correspondences, m_1/n represents the true-positive probability and m_2/n represents the false-positive probability for that pair of images. True-positive is 0 when no correct correspondences are found and is 1 when the maximum number of possible correspondences is found. False-positive is 0 when no wrong correspondences are found and is 1 when the maximum number of possible false correspondences is found. To determine these measures, a feature in the sensed image should not be allowed to correspond to more than one feature in the reference image to ensure that $m_1 + m_2 = n$. We want to increase the true-positive probability while decreasing the false-positive probability. Ideally, we want to find the largest number of correct correspondences that it is possible to find without obtaining any wrong correspondences. The true-positive and false-positive probabilities are determined by testing an algorithm on a sufficiently large number of images that are representative of inputs to the algorithm.

Robustness in a feature correspondence algorithm refers to the variation in the reliability (true positive or false positive) as one or more of the input parameters are varied. In addition to the outliers, control points in the sensed image may be locally displaced with respect to corresponding control points in the reference image. The ability of an algorithm to consistently find a high number of correct correspondences under variations in image parameters makes the algorithm robust. Robustness shows the stability of the true-positive or false-positive probability under variation in one or more of the input parameters.

In template matching, given a set of templates in the reference image, it is required to find windows in the sensed image that correspond to them. A particular match is either correct or incorrect. Therefore, characterizing a template-matching algorithm in terms of its accuracy does not make sense. Reliability of template-matching

algorithms is, however, meaningful. What ratio of the correspondences is found? What ratio of the correspondences is missed? Reliability can be described in terms of these measures. Assuming M templates are selected in the reference image and among the M templates only N exist in the sensed image ($M - N$ templates fall outside the sensed image so they do not have correspondences) then, if a particular algorithm finds m_1 correct correspondences and m_2 incorrect correspondences, where $m_1 + m_2 = N$, the true-positive probability is m_1/N and the false-positive probability is m_2/N. For the probabilities to be meaningful, the algorithm should be tested on a sufficiently large number of images that represent the possible inputs. Computation of the true-positive and false-positive probabilities requires knowledge about the true correspondences, which if not known should be determined visually.

7.3 TRANSFORMATION FUNCTION PERFORMANCE

A transformation function uses the coordinates of corresponding points in two images to spatially align the images. The coordinates of corresponding points used to determine a transformation function may contain inaccuracies. Evaluation of the performance of a transformation function requires the determination of the behavior of the transformation under:

- digital noise and small positional error in the correspondences;
- variation in the spacing between points;
- variation in the density of points;
- local geometric differences between images.

Even when corresponding control points in the images are correctly found, due to the digital nature of images, corresponding points could be off by up to half a pixel. The local geometric difference between images may cause a template-matching algorithm to find correspondences that are off by a pixel or two. A transformation function should be able to smooth noise and small positional inaccuracies in the correspondences. Therefore, when noise or small inaccuracies in the correspondences exist, methods that are based on approximation are preferred over methods that are based on interpolation. The presence of a wrong correspondence should not affect an entire resampled image, but rather, should influence the neighborhood where the mistake has occurred. Therefore, transformation functions that are based on monotonically decreasing basis or weight functions are preferred over transformation functions that use monotonically increasing basis or weight functions.

Control points detected in an image rarely have a uniform spacing. The basis or weight functions used to represent a transformation function should, therefore, be able to adjust to the irregular spacing of the points.

A set of points that are irregularly spaced may have a uniform density. If spacing between the control points varies greatly from one area to another in an image, the density of the points varies across the image domain. For instance, in a very detailed area in an image, a much higher density of points is obtained than in a rather homogenous area. Some of the control points in high-density areas may be removed

to produce an overall uniform density of points but this will discard information that may be critical for accurate registration of the images. A transformation function should be able to handle large variations in the density of the control points. Also, a transformation function should be able to handle a large number of correspondences, typically in the order of thousands, to accurately register volumetric images that have local geometric differences. For that reason, algorithms that do not require the solution of a system of equations are preferred over algorithms that require the solution of a system of equations where the number of equations is equal to the number of control points.

Most importantly, a transformation function should be able to accurately represent the geometric difference between two images. If the geometric difference between two images is linear and that is reflected in the point correspondences, the obtained transform should represent a linear function or one that is very close to it. A transformation function should use information present in the correspondences to correctly represent the local geometric difference between images. Transformation functions with monotonically decreasing basis or weight functions keep local deformations local and are preferred over transformation functions that have monotonically increasing basis or weight functions.

Evaluation of the accuracy of a transformation function requires knowledge about the correspondence between all points in the images. The correspondence between all points in two images will be known if one image is generated from the other using a known transformation. To evaluate the performance of a transformation function, real reference images may be used to create simulated sensed images. Average, median, maximum, or root-mean-squared distances between estimated and true correspondences can be used as a metric to quantify the accuracy of a transformation function. For the accuracy to be meaningful, the functions used to generate the sensed images should truly represent the kind of geometric differences observed in the input. Various realistic densities and organizations of control points, as well as realistic noise among the correspondences, should be used in this evaluation.

A transformation function always registers two images. Therefore, speaking about the reliability of a transformation is meaningless. It is the accuracy of a transformation that is of interest. One may talk about the robustness of a transformation, which involves determining changes in registration accuracy as the density or organization of the control points is changed or the local geometric difference between images is varied. The ability of a transformation to consistently produce a highly accurate registration under variations in one or more of its input parameters makes the transformation robust. A highly robust transformation can accurately register images with varying density and organization of control points, it is not sensitive to a reasonable amount of random noise among the correspondences, and it can handle a reasonable amount of local geometric differences between images.

7.4 REGISTRATION PERFORMANCE

If an image registration algorithm is being evaluated during the design stage, access to each component of the registration is available. Therefore, each component of the algorithm can be individually evaluated. However, a purchaser of image registration software usually does not have access to the individual registration components. Consequently, the entire software has to be evaluated in one piece.

If the performances of the components of a registration algorithm can be individually evaluated, the performances can be combined to produce the performance of the registration. Overall accuracy will depend on the accuracy with which control points are selected and the accuracy with which points in the sensed image are resampled to align with corresponding points in the reference image. If, for example, average error in selecting feature positions for a particular class of images is E, point correspondences with average error E should be generated with different densities, organizations, and geometric differences and used as inputs to the transformation to determine the average error in registration.

The reliability of a registration algorithm depends on the reliability of the selected control points and the reliability of the correspondences. If the feature selector has false-positive probability p, the correspondence algorithm should be tested with control points that have false-positive probability p under various density, organization, and geometric difference between images. If the false-positive probability in the obtained correspondences is q, $1 - q$ will be the reliability of the registration.

Robustness involves characterizing the accuracy or reliability of a registration under variations in its input parameters. Input parameters can represent image noise, intensity difference between images, geometric difference between images, and the percentage of dissimilar regions in images. The robustness of a registration algorithm can be defined by the product of the robustness of the components of the algorithm.

If the user is faced with evaluating the performance of a black box, simulation data where the correspondence between all points in the images is known may be used. Creation of images that contain realistic intensity and geometric differences between images, however, is a difficult task. Alternatively, fiducial markers whose positions in the images can be accurately determined may be used. After obtaining images of scenes containing fiducials, the fiducials may be digitally removed and the images registered without them. Knowing the coordinates of the fiducials in the images, errors at the fiducials can then be determined. Although this enables use of realistic images in the evaluation, the error measurements are limited to the positions of the fiducials. Because of limitations in the placement of fiducials in the scene, this form of input data cannot determine registration errors everywhere or at least at uniformly spaced points in the image domain that is essential in the evaluation of nonrigid registration algorithms.

7.5 SUMMARY

Measurement of the accuracy, reliability, and robustness of image registration algorithms and their components was discussed. Accuracy shows the distance between corresponding pixels/voxels in the images after registration. Reliability shows the likelihood that an obtained registration is satisfactory, and robustness shows the stability of the behavior of an algorithm under variations in its input parameters. If access to each component of a registration algorithm is available, evaluation should be carried out on the components individually to find the weak components. This makes it possible to improve the performances of the weak components by perhaps changing the free parameters of the components to adjust them to the particular type of images being registered.

A registration algorithm that produces consistently high accuracy when tested on various types of images is ideal. However, this is rarely possible. An algorithm that performs well on a particular class of images uses information about that class of images. When a different class of images is used, the knowledge base or the constraints that the algorithm is based on are violated and as a result the algorithm does not perform well. Typically, an algorithm that performs well on a particular class of images performs poorly in general, and an algorithm that is designed to handle various image types does not perform well on a particular class of images. An algorithm may be designed to have free parameters that can be varied to adapt to different types of images. For such algorithms, rather than providing a single performance measure, a chart should be provided showing the image type, the values of the free parameters optimally performing under an image type, and the corresponding performance measure.

7.6 BIBLIOGRAPHICAL REMARKS

The performance of registration algorithms is most rigorously evaluated in medical applications. This could be because the loss incurred as a result of a misregistration in a medical application is much higher than in other applications. In industrial or remote sensing applications, if necessary, a human operator can intervene and assist a registration. Although this is also possible in medical applications, visual examination of large volumetric images to determine registration errors in local neighborhoods is very time consuming and tiring.

Various algorithms have been developed for the rigid registration of medical images. These algorithms shift one image over the other and at each shift position determine the similarity between the two. This is especially true for algorithms that use mutual information as the similarity measure [254, 372] because large overlapping areas are needed to achieve high accuracy in registration. These algorithms have a very simple but effective mechanism that can register multimodality images with translational and small rotational differences.

Fitzpatrick and colleagues [118, 120, 409] have prepared fiducial-based images for the evaluation of rigid registration algorithms. The prepared data sets make it

possible to determine the accuracy and reliability of a registration algorithm on a number of representative multimodality brain images. Accuracy can be determined for only image points corresponding to the fiducials though. Because the fiducials could not be placed within the brain, registration accuracy is limited to points outside the brain. However, because the data sets are for the evaluation of rigid registration algorithms, accuracy at the fiducials may be extrapolated to the entire image domain. Various types and spacing of fiducial markers have been used in the evaluation of rigid registration algorithms [107, 251, 269, 398]. Fiducial markers allow the use of realistic images in the evaluation, but the obtained accuracy is only for those image locations that represent the fiducials. Therefore, use of fiducials is limited to evaluation of rigid registration algorithms. For evaluation of nonrigid registration algorithms, simulation data is preferred.

Simulation data have been used in the evaluation of image registration algorithms before [71, 172, 346]. Simulation images have the ability to represent all types of geometric difference between images and provide exact correspondence between all points in the images. Therefore, they can be used to evaluate rigid as well as nonrigid registration algorithms. Creation of realistic data, however, requires realistic models of noise, intensity, and geometric differences between images, which may be difficult to find.

Anatomic landmarks have also been used to evaluate the accuracy of registration algorithms [27, 113, 185, 189, 330, 371, 373]. If simulation data, fiducial markers, or anatomic landmarks are not available, registration accuracy may be evaluated qualitatively through visual examination [416]. Fitzpatrick *et al.* [119] have shown that 2 mm errors can be visually detected in a brain MR and CT registration. Wong *et al.* [414] have found that similar registration errors can be achieved visually even when using lower resolution PET images.

8

Image Fusion

Image fusion is the process of combining information in two or more images of a scene to enhance viewing or understanding of the scene. If the images are registered, information can be combined at pixel level. If the images are not registered, information from images should be determined individually and then combined. In this chapter, discussions will be limited to fusion of registered images. Specifically, fusion of multi-exposure and multi-focus images will be covered.

Intensities across a 3-D scene vary depending on the orientation of scene surfaces with respect to light sources and also with respect to the camera if the scene has specular surfaces. Depending on the scene geometry, scene lighting, and camera position, intensities could greatly vary across an image of the scene. Rarely in an image do all scene points appear well-exposed. No matter what shutter speed is used, some scene areas may appear over- or under-exposed. A method for combining images captured at different exposures into one image where all areas are well-exposed is described.

The distance between scene points and the camera can also vary greatly. When capturing an image, the camera is focused on a small area in the scene that is at a particular distance to the camera. Only scene points within the focus plane will appear sharp. Scene points behind or in front of the focus plane will appear blurred. To capture detailed information about an entire scene, it is necessary to combine images captured at different focus levels. The blending of images captured at different focus levels of a scene into a single image where all points appear sharp is also discussed.

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

8.1 FUSING MULTI-EXPOSURE IMAGES

When perceived brightness across a scene varies, there will always be some over- or under-exposed areas in an image of the scene, no matter what shutter speed is used. Given a set of images of a scene obtained at different shutter speeds (exposure levels), the image domain is subdivided into small blocks and for each block the image containing the most information is selected. The images selected in this manner are then blended together to create a single image that is well-exposed everywhere. A monotonically decreasing blending function is centered at each block within the selected image. The blending functions are defined such that their sum is 1 everywhere in the image domain. Within a block, the well-exposed image is given a higher weight than an under- or over-exposed image. This image blending will create an image that is well-exposed and is highly informative everywhere.

When images at multiple exposures of a scene are obtained, a scene area in one image may appear the most informative of all the images. Under- and over-exposed areas in an image carry less information than the well-exposed areas. We will select the "best-exposed" image for each local area and by properly blending the selected images create the output. An image is considered best-exposed within a local area if it carries more information than other images within that area.

8.1.1 Image blending

The problem to be solved is as follows. Given N registered images of a scene obtained at different exposure levels, it is required to blend the images in such a way that the obtained image has the maximum possible information. Image information will be measured using image entropy [55, 132]. The higher the entropy of an image, the more informative the image will be. In image processing, maximization of image entropy has been used as a means to enhance images [55, 176]. To maximize entropy, intensities in an image are mapped to new intensities in such a way that the obtained image has a flat histogram. A flat histogram produces intensities that are equally likely to appear in the image, maximizing image entropy.

In the method described here, the image domain is partitioned into blocks and within each block the image providing the most information is selected. The selected images are then blended together to produce the output. When brightness across a scene varies greatly, images captured at various exposures of the scene are needed to create an image that is well-exposed everywhere. The main problem to be solved is to identify the image that contains the most information within each image block. An image that is over- or under-exposed within a block does not carry as much information as an image that is well-exposed in that block.

In a gray scale image, entropy is defined by

$$E_g = \sum_{i=0}^{255} -p_i \log(p_i), \tag{8.1}$$

where p_i is the probability that if a pixel is randomly selected from the image, it will have intensity i. To estimate $\{p_i : i = 0, \ldots, 255\}$, first, the histogram of the image is computed. If the number of pixels having intensity i is n_i and the image contains n pixels, then $p_i = n_i/n$.

The human visual system is capable of discriminating far more colors than luminances [177]. Therefore, when the images are in color, image information should be computed using the color values rather than the luminances. Since we are interested in maximizing perceived information, the RGB colors may be transformed into $CIELab$ colors [72] and image information may be computed using Lab color values. This transformation nonlinearly maps colors in such a way that most colors distinguishable by the human visual system are obtained [188, 221]. To determine the entropy of a color image, a 3-D histogram that represents the frequency of colors appearing in the image is needed. Since such a histogram may have more elements than the number of pixels in the image itself, instead of using all colors in an image, the colors are clustered to the 256 most frequently observed colors $\{\mathbf{C}_i : i = 0, \ldots, 255\}$. An efficient algorithm for achieving this has been described by Xiang [419]. The color at each pixel is approximated by the color of the cluster center closest to it, and entropy of a color image is approximated by

$$E_c = \sum_{i=0}^{255} -p_i^c \log(p_i^c) \tag{8.2}$$

where p_i^c is the number of pixels within cluster i (with cluster center $\mathbf{C}_i$) divided by the total number of pixels in the image.

To determine the image that is best exposed within a local neighborhood, first the images are divided into blocks of size $d \times d$ pixels. Here, d will be one of the unknown parameters of the method to be determined by maximizing information content in the output. Image entropy E_g or E_c is determined for each block and the image that provides the highest entropy for a block is selected to represent that block in the output.

If we simply create an image from the composition of image blocks obtained in this manner, we will obtain an image that may have sharp discontinuities across image blocks. Although each block will be highly informative, the image created from them will be awkward to view. An example is given in Fig. 8.1. Figures 8.1a–e show a set of images of an office taken at five different exposure levels. These images are of size 640×480 pixels. Dividing the images into blocks of size 160×160 pixels, we obtain an array of 3×4 blocks. If we calculate the entropy within each block for the five input images, cut out the block from the image that contains the highest entropy, and put the blocks together, we will obtain the image shown in Fig. 8.1f.

To remove discontinuities across image blocks, instead of cutting and pasting the blocks in an image, the images representing the blocks are smoothly blended together. A monotonically decreasing blending function is centered at each block and image colors are multiplied by corresponding blending function values. Therefore, the color at a pixel in the output is obtained from a weighted sum of colors at the same pixel in the given images. A blending function assigns the maximum weight to the pixel at the

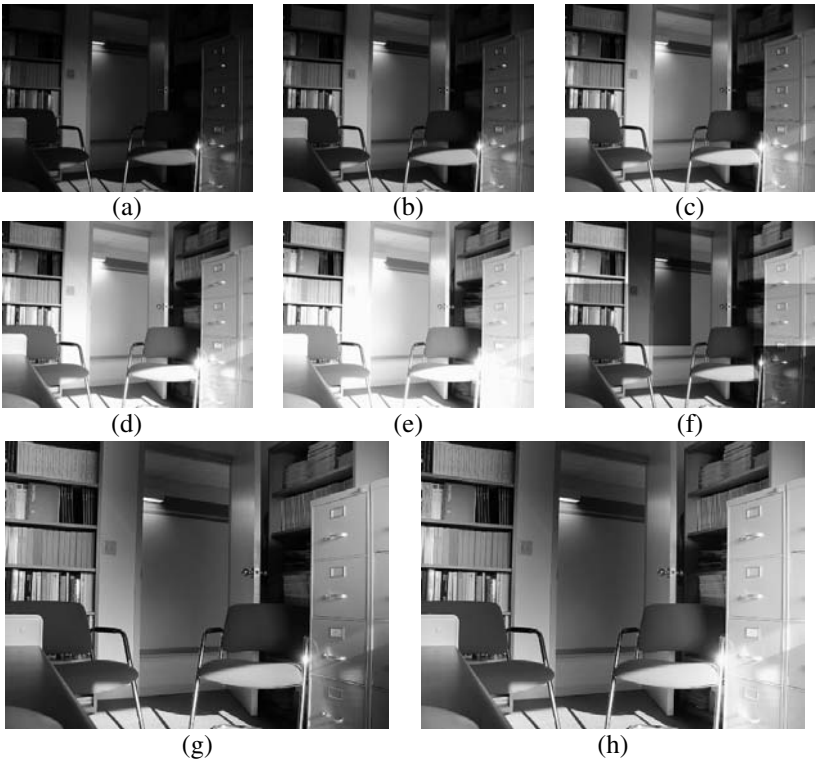


Fig. 8.1 (a)–(e) Images representing different exposures of an office room. (f) The image obtained by composing the twelve blocks cut out of the five images. (g) The image produced by centering the blending functions at the selected blocks in the images, multiplying the blending functions by image colors, and adding the weighted colors together. (h) The most informative image constructed from images (a)–(e). These images are of size 640×480 pixels. Optimal values found for d and σ were 128 and 96 pixels, respectively. (See also color plate section.)

center of a block and assigns smaller weights to other pixels inversely proportional to their distances to the center of the block.

Assuming N images are given and each image has been subdivided into an array of $n_r \times n_c$ blocks, and assuming j and k denote the row and column indices of a block, and also letting $\mathbf{I}_{jk}$ denote the image that has the highest entropy among the N images for block jk, we compute output image $\mathbf{O}(x, y)$ from

$$\mathbf{O}(x, y) = \sum_{j=1}^{n_r} \sum_{k=1}^{n_c} W_{jk}(x, y)\mathbf{I}_{jk}(x, y), \tag{8.3}$$

where $W_{jk}(x, y)$ is the value of the blending function centered at the jkth block at location (x, y), and $\mathbf{I}_{jk}(x, y)$ is a vector representing the Lab color coordinates of the

image representing block jk at location (x, y). Note that the number of images being blended is equal to the number of blocks and not the number of given images. When the blocks are small, many blocks may be selected from the same image. Also note that, rather than using image values within only the selected blocks, entire images are blended together, although the contribution of far away blocks on a pixel in the output may be very small. This process smoothly blends the images, avoiding the creation of discontinuities across blocks.

Since the blocks form a uniform grid, tensor-product blending functions, such as B-splines [280], may be used. When B-spline blending functions are centered at the blocks, the blending functions will smoothly blend the images. This works for interior blocks but not the border blocks because the blending functions do not extend to the image borders. Rational Gaussian (RaG) blending functions [153] extend over the entire image domain and have a sum of 1 everywhere. RaG blending functions are defined by

$$W_{jk}(x, y) = \frac{G_{jk}(x, y)}{\sum_{m=1}^{n_r} \sum_{n=1}^{n_c} G_{mn}(x, y)}, \tag{8.4}$$

where n_r and n_c denote the number of image blocks vertically and horizontally, and $G_{jk}(x, y)$ represents the Gaussian of height 1 centered at the jkth block when evaluated at (x, y);

$$G_{jk}(x, y) = \exp\left\{-\frac{(x - x_{jk})^2 + (y - y_{jk})^2}{2\sigma^2}\right\}. \tag{8.5}$$

Here, (x_{jk}, y_{jk}) are the coordinates of the center of the jkth block, and σ is the standard deviation or the width of the Gaussians. In the following, we will refer to σ as the *width* of the blending functions.

Figure 8.1g shows the image obtained by centering blending functions of standard deviation 80 pixels at blocks of size 160×160 pixels in the selected images and multiplying color values at image pixels with weights given by the blending functions and adding the weighted colors together. This is the kind of output we will obtain. Here, σ is one other unknown parameter that has to be determined to maximize information content in the output. Since more than one image block in the output may point to the same input image, an input image may be used more than once in relation (8.3). The blending functions blend the proper amounts of the images to create the output.

There are two parameters to be determined; d, image block size, and σ, the width of the blending functions. As these parameters are varied, different images are obtained. We have to determine the d and σ that maximize image entropy as computed by relation (8.1) or (8.2). The optimal parameters are found using a gradient-ascent approach.

The following algorithm summarizes the steps of the image blending process.

Algorithm 8.1: Fusing multi-exposure images.

```
1:  Set block size d and width of blending functions σ to
```

some initial guesses and let Δ be the increment in d and σ.

2: Determine the entropy of block jk in each image using formula (8.1) or (8.2) for $j = 1, \ldots, n_r$ and $k = 1, \ldots, n_c$ and let the image having the highest entropy within block jk be $\mathbf{I}_{jk}$.

3: Find the entropy of the image obtained by blending the images selected in step 2.

4: Increment or decrement d and σ by Δ in the gradient-ascent direction and repeat steps 2 and 3 until the highest entropy is reached. Let the d and σ producing the highest entropy be d_{max} and σ_{max}, respectively.

5: Select the images identified in step 2 when setting $d = d_{max}$, and create a new image by blending the selected images with blending functions of width σ_{max}.

The initial values for parameters d and σ are set to the averages of optimal values obtained for these parameters in a number of test runs of the algorithm. Applying this algorithm to the images in Figs 8.1a–e, the image shown in Fig. 8.1h is obtained. As initial values, $d = \sigma = 32$ pixels were used and Δ was 32 pixels. The values of d_{max} and σ_{max} obtained by the algorithm were 128 and 96 pixels, respectively. Smaller increments produce higher optimal entropies at a higher computational cost. This parameter should be chosen as a compromise between speed and accuracy.

8.1.2 Examples

Examples of image fusion by Algorithm 8.1 are shown in Figs 8.2–8.6. Entropies of the original images and the resultant images are shown in the figure captions. Initial values of d and σ for these images were both 32 pixels and Δ was 8 pixels.

Some of the input images in Figs 8.2–8.5 have large well-exposed areas. Therefore, entropies of some of the images are quite high but the entropies of the newly created images are even higher. To compare the automatic method with manual photographic methods, an example is given in Fig. 8.6. Results obtained manually by an expert photographer and by Algorithm 8.1 are virtually indistinguishable, producing very close entropies.

Optimal parameters d and σ vary from image to image; d and σ depend on the detailedness of given images. Images representing scenes with highly varying albedos, highly varying surface orientations, and highly varying environmental factors such as shadows and specularities, produce smaller optimal d's and σ's. Scenes with smoothly varying albedos and smoothly varying surface orientations produce larger optimal d's and σ's. The optimal d depends on the size of well-exposed regions and the optimal σ depends on the rate the well-exposed regions transition to under- or over-exposed regions.

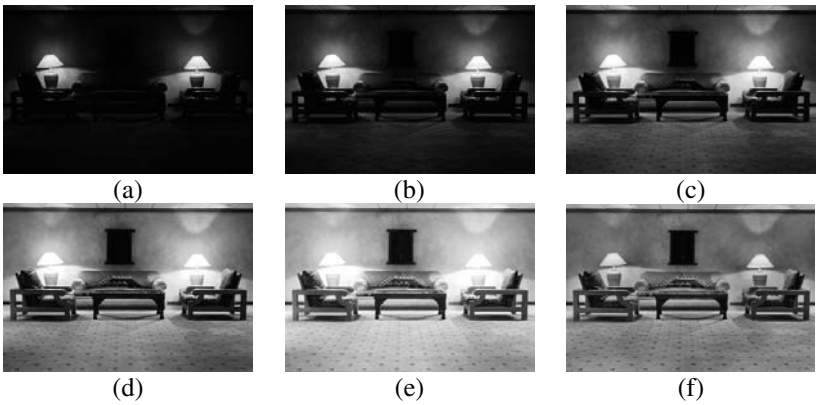


Fig. 8.2 (a)–(e) Images representing different exposures of a waiting-room scene. Entropies of the images are 2.82, 3.36, 4.21, 4.78, and 4.40, respectively. (f) The image obtained by fusing the five images. Entropy of the fused image is 5.29. These images are of size 343×231 pixels. Optimal parameters found are $d = 32$ pixels and $\sigma = 32$ pixels. (See also color plate section.)

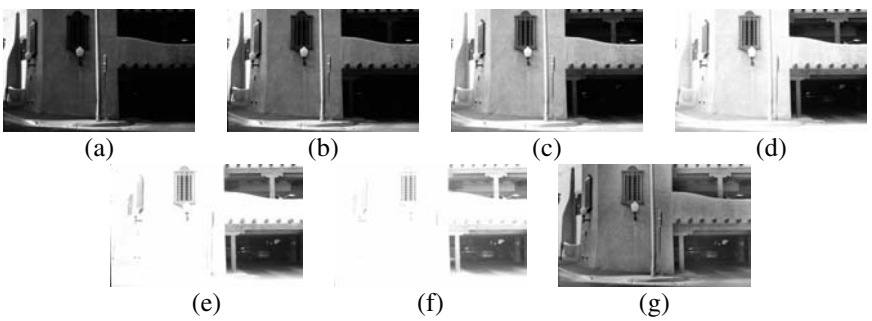


Fig. 8.3 (a)–(f) Images of a garage scene obtained at different exposures. (g) The image obtained by fusing the six images. Entropies of images (a) through (g) are 3.02, 3.29, 5.04, 4.70, 4.03, 3.90, and 5.27, respectively. These images are of size 348×222 pixels. The optimal $d = 8$ pixels and the optimal $\sigma = 16$ pixels. (See also color plate section.)

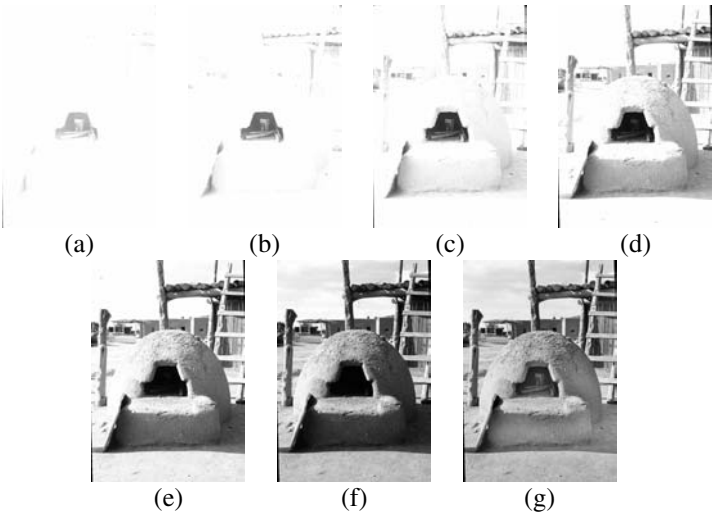


Fig. 8.4 (a)–(f) Images of an igloo scene obtained at multiple exposures. (g) The image obtained by blending the six images. Entropies of these images are 2.92, 3.02, 3.97, 4.39, 4.86, 5.18, and 5.38, respectively. The images are of size 236×341 pixels. The optimal values of d and σ are both 16 pixels. (See also color plate section.)

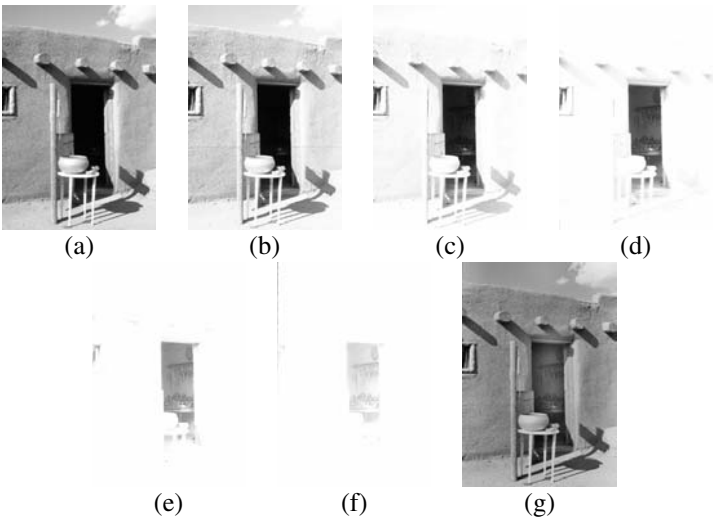


Fig. 8.5 (a)–(f) Images of a door to a dark room obtained at different exposures. (g) Image obtained by fusing the six images. Entropies of the images are 5.09, 4.81, 3.86, 3.17, 3.04, 2.81, and 5.23, respectively. These images are of size 231×338 pixels. Optimal $d = 8$ pixels and optimal $\sigma = 16$ pixels. (See also color plate section.)

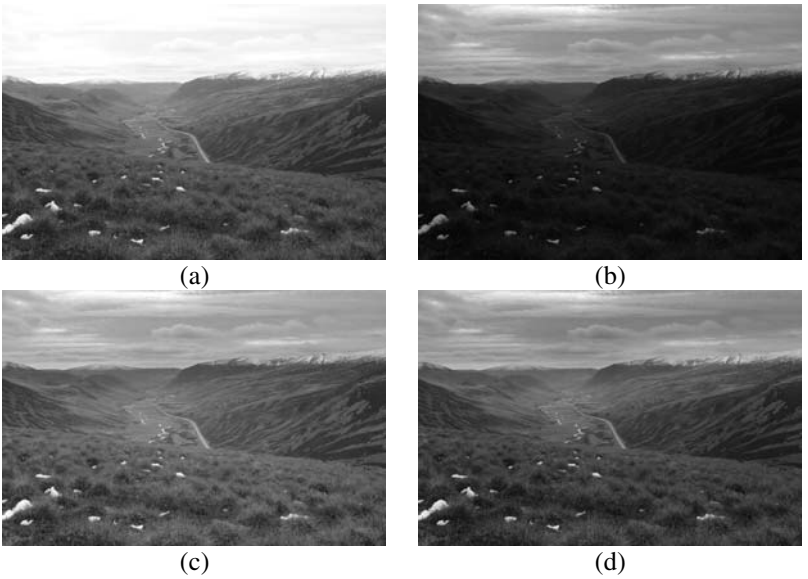


Fig. 8.6 (a), (b) Two images of the Scottish Highlands representing different exposure levels. Entropies of these images are 4.86 and 4.34. (c) The image created by manual photographic techniques. (d) The image constructed by fusing (a) and (b). Entropies of (c) and (d) are 5.22 and 5.20, respectively. These images are of size 1536×1024 pixels. Optimal values of d and σ are both 64 pixels. (See also color plate section.)

8.2 FUSING MULTI-FOCUS IMAGES

Often due to great variations in a scene's depth, it is not possible to capture an image of the scene in such a way that all scene areas appear sharp. Only scene areas that are at the focus plane will appear sharp and areas in front of and behind the focus plane will appear blurred. The ability to create a single image where all scene areas appear sharp is not only desired in digital photography, it is needed in various machine vision applications.

We cannot use image information, as we did when fusing multi-exposure images, to fuse multi-focus images. A sharp image may have a smaller entropy than a blurred version of it. We can use image gradient as a means to measure sharpness. Specifically, the sum of squared gradient magnitudes in each block may be taken as the sharpness measure. Squared gradient magnitude rather than gradient magnitude is used to avoid a sharp image with a small number of high-gradient pixels having a sharpness measure that is smaller than a blurred version of it. Square gradient in a color image is determined from the sum of the squared gradients of the red, green, and blue bands (or, alternatively, the Lab bands). Therefore, as in Algorithm 8.1, the image domain is subdivided into blocks and the image appearing the sharpest

within each block is selected and blended together to create an image that is sharp everywhere.

Figures 8.7a and 8.7b depict two images, obtained at two different focus levels, of a statue showing dancing children. Fusing the images produces the image shown in Fig. 8.7c. In this image, both front and rear children are in focus. A second example is given in Fig. 8.8. Figures 8.8a and 8.8b show two images of a newspaper scene obtained at different focus levels. More than half of each image is blurred. Fusing the images, the image shown in Fig. 8.8c is obtained. Scene depth gradually varies in this example. Near the center of the scene, both images are rather blurred and, as a result, the center part of the combined image is also blurred. An example where there is a sharp change in scene depth is shown in Fig. 8.9. The scene contains two books that are at different distances to the camera. The image in Fig. 8.9a was captured by focusing the camera on the front book and the image in Fig. 8.9b was captured by focusing the camera on the rear book. Figure 8.9c shows the result of fusing the two images. Both books now appear in focus.

(a) (b) (c)

Fig. 8.7 Images obtained by focusing the camera on (a) front children and (b) rear children. (c) Image obtained by fusing the two images according to Algorithm 8.1 using image gradient as the sharpness measure.

(a) (b) (c)

Fig. 8.8 (a), (b) Two images obtained at different focus levels of a newspaper scene. About half of each image appears sharp. (c) The image obtained by fusing the two images.

(a) (b) (c)

Fig. 8.9 (a), (b) Two images of a book scene, one captured by focusing the camera on the front book and another captured by focusing the camera on the rear book. (c) The image obtained by fusing the images captured at the two focus levels.

8.3 SUMMARY

An image fusion algorithm should have two inherent characteristics. First, it should derive the most relevant information in the input and present them in the output and, second, it should not present information in the output that is non-existent in the input. If the objective is to create a sharp image from a number of images that are partially blurred, the created image should not contain areas that are sharper than the same areas in the input. A fusion algorithm should not overdo a process. It should be truthful to the information present in the input and create an output that is an accurate representation of the input.

In this chapter, a method was described for fusing multi-exposure images of a scene into a highly informative image by image blending. The method divides the image domain into small blocks and selects the image that provides the most information within each block. The selected images are then blended together to produce the output. The block size and the width of the blending function are determined using a gradient-ascent algorithm to maximize information content in the output.

The described method preserves scene highlights if color information within a highlight area is quite high. A characteristic of the method is that it does not change the local color and contrast in the input. Therefore, contrast in an image area in the output cannot be higher than the contrast in the best-exposed image in the input. If further contrast enhancement is needed, traditional methods such as inverse filtering, as discussed in Chapter 2, should be used.

Also discussed in this chapter was fusion of multi-focus images. Gradient magnitude was used to quantify sharpness. An algorithm similar to that used for fusion of multi-exposure images was used to fuse multi-focus images.

8.4 BIBLIOGRAPHICAL REMARKS

Image fusion methods can be grouped into high-level, mid-level, and low-level methods. High-level methods receive image descriptions from two or more images and

combine the descriptions to a richer one. The descriptions could be in the form of relational graphs [353, 412]. These methods require that objects of interest in each image be identified and their relations be determined by a vision system. The process uses minimum observer interaction and can automatically detect events in image sequences.

Mid-level techniques work with segmented images to improve segmentation of stationary objects or track moving objects. Although recognition of objects and events is left to the observer, the system can detect certain simple objects or events. Lewis *et al.* [243] describe a method that uses segmentation results from two or more images to create a more robust segmentation. Piella [310] discusses image fusion methods based on a multi-resolution segmentation method and Nikolov *et al.* [288] discuss a fusion method using multiscale edges.

If the images are accurately registered, pixel-level information in the images can be combined. Information integration can be achieved in the frequency domain or in the spatial domain. Wang *et al.* [403] find the wavelet transforms of the images, and fuse the transform coefficients. The largest coefficients or a weighted sum of the coefficients of the transformed images are combined to create the fused transform image. The inverse transform of the fused transform image is then computed to create the output image. Jones and Nikolov [212] describe a fusion technique in the spatial domain which finds the intensity at a point in the output using a weighted sum of intensities of corresponding pixels in the input. Petrović and Xydeas [309] use image gradients to fuse images. A multiresolution approach is taken where, at each resolution, input images are represented by gradient maps and the gradient maps are combined to a new gradient map. Then, a reconstruction process similar to the wavelet approach is taken to create the output from the combined gradient map.

Although increased redundancy in images simplifies the fusion process, image fusion is most effective when the least redundancy is present in the given images. Multimodal images contain different information about a scene and by fusing them an image will be obtained that is a rich source of information about the scene. Alparone *et al.* [4] discuss fusion of multimodal satellite images, Jones and Nikolov [212] discuss fusion of muiltimodal medical images, Toet *et al.* [385, 386] discuss fusion of multimodal video images, and Yaroslavsky *et al.* [424] discuss fusion of both temporal and multimodal video images. Temporal images represent images in the same modality that are taken at different times. By fusing such images objects can be tracked and events can be detected. Snidaro *et al.* [363] describe a method for tracking persons in a video and Chen *et al.* [62] describe a method for tracking automobiles in a video of a highway scene.

Fusion of multi-exposure images has appeared as high dynamic range reduction in the literature. Intensities in an image are considered intensities of a high dynamic range image between two threshold values. Dynamic range is reduced by decreasing the global contrast while maintaining or increasing the local contrast. This is achieved by nonlinearly mapping intensities in the high dynamic range image to intensities in the output using image gradients. Excellent surveys of these methods are provided by DiCarlo and Wandell [94], and Devlin *et al.* [91].

Mechanisms to capture multi-exposure images have been developed. A camera system that captures n images on the same sensor but with $\frac{1}{n}$th of the sensor resolution is described by Nayar and Mitsunaga [284]. A camera system that uses a beam splitter to send incoming light to n different sensors, thus capturing n images at different exposure levels is described by Aggarwal and Ahuja [1]. With some modification, the beam-splitter mechanism used to capture images at multiple exposures can be used to capture images at multiple focus levels.

9

Image Mosaicking

Image mosaicking is the process of seamlessly stitching together or blending a set of overlapping images of a scene into one large image. This process is needed in various remote sensing, computer vision, and computer graphics applications. It is used in map building by piecing together georectified images [115]. It is used in object tracking by subtracting registered images and tracking image changes [197]. It is used in the creation of panoramic images by integrating a sequence of overlapping images [376]. It is also used in 3-D scene reconstruction by integrating multiple-view range images [82].

Image mosaicking is the process of finding a global transformation that resamples a set of images of a scene into a common coordinate system. Depending on the scene content, the distance of the camera to the scene, and the position and orientation of the camera capturing the images, different transformations may be needed to register and integrate the images. For instance, if a sequence of images is obtained by a camera with a fixed lens center and a horizontal optical axis while rotating the camera about a vertical axis passing through the lens center and, assuming the camera is sufficiently far from the scene, adjacent images in the sequence will have only translational differences. Such images can be registered by shifting one over the other and locating the position where maximum similarity is obtained between the images. If the camera is not very far from the scene, the images obtained during the rotation of the camera can be mapped to a cylindrical surface whose axis is parallel to the axis of rotation. The mapped images will then have only translational differences and can be registered easily. If the camera is not fixed and is not far from the scene, but the scene is flat, images of the scene can be registered by the projective transformation. This requires a minimum of four corresponding points in the images. If the scene is not flat and the

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

camera is not very far from the scene, the images will have local geometric differences, requiring a nonlinear transformation to register them.

If the camera parameters in each image acquisition are known, the camera models may be used to relate the captured images and align them. Often, however, either the camera parameters are not known or the provided parameters are not very accurate. Therefore, from information within the images, it is required to find a transformation that seamlessly combines the images. The discussions below will focus on a general methodology where no information about the camera position, camera orientation, and scene content is given. Some limited information about the scene or the camera may be available. For instance, it may be known that the scene is flat or the camera is very far from the scene.

9.1 PROBLEM DESCRIPTION

Two main problems have to be solved in image mosaicking. First, a global transformation that spatially aligns all images need to be determined and, second, intensities within overlapping areas should be blended to provide a smooth transition from one image to the next.

Problem 1. Determining the global transformation
Given n images with some overlap between adjacent images and knowing that at least three corresponding control points exist between adjacent images, determine a global transformation that can spatially align and combine all images into a single image where the scene area covered by the combined image is the union of the scene areas covered by the input images.

Problem 2. Blending image intensities
Given n spatially aligned images, blend intensities in overlapping areas in adjacent images such that the images seamlessly merge into one large image. It is understood that intensities of corresponding pixels in overlapping images may be different.

9.2 DETERMINING THE GLOBAL TRANSFORMATION

Let's first consider the case where the scene area within each image is rather flat. That is, variation in scene elevation in an image compared to the distance of the camera to the scene is negligible. Let's also suppose the field of view of the camera capturing each image is small so that the relation between image and scene points can be written by a linear transformation. The area between two adjacent images, therefore, can be registered by a linear transformation. Later, the discussions will be extended to include scenes that are not flat or when the field of view of the camera is not small.

It is understood that, although the relation between adjacent images may be linear, the relation between images that are not adjacent may be nonlinear. To satisfy this condition, the global transformation that registers all the images is taken to be

nonlinear. To satisfy the local linearity condition, the global transformation is taken such that it approximates to the linear function aligning two overlapping images. The weighted linear transformation that was discussed in section 5.7 provides this capability. Therefore, each component of the global transformation will be represented by

$$f(x,y) = \sum_{i=0}^{N} f_i(x,y) b_i(x,y), \tag{9.1}$$

where N is the number of overlaps produced by the images, $f_i(x,y)$ is the linear transformation that registers two adjacent images producing the ith overlap, and $b_i(x,y)$ is a monotonically decreasing weight function that is centered at the ith overlap. The weight functions are chosen such that the influence of a linear transformation registering two overlapping images gradually vanishes as one moves away from the center of the overlap area. The term $f_0(x,y)$ represents the relation of the reference image with itself and, therefore, is the unit transformation. It is required that the sum of the weights at each point in the mosaic be one, that is,

$$\sum_{i=0}^{N} b_i(x,y) = 1. \tag{9.2}$$

The weight function $b_i(x,y)$ can be represented by rational Gaussian (RaG) weights [153], defined by

$$b_i(x,y) = \frac{G_i(x,y)}{\sum_{i=0}^{N} G_i(x,y)}, \tag{9.3}$$

where

$$G_i(x,y) = \exp\{-[(x-x_i)^2 + (y-y_i)^2]/2\sigma_i^2\}, \tag{9.4}$$

(x_i, y_i) is the center of the ith overlap area, and σ_i is proportional to the width of the overlap area. The width of an area can be considered the diameter of the largest circle enclosed in that area. Therefore, $\sigma_i = sr_i$, where r_i is the radius of the largest circle enclosed in the ith overlap area and s is the proportionality term. Parameter s may be varied globally to control the smoothness of the entire transformation. As s is increased, the influence of a local transformation extends to farther points while, as s is decreased, the process becomes more local and the influence of a local transformation remains more local.

Figure 9.1 demonstrates this property. Consider creating a mosaic from five images. Consider one component of the transformation as shown in Fig. 9.1. The center plane in Fig. 9.1a shows the unit transformation, that is $X = x$ for the X-component and $Y = y$ for the Y-component. The unit transformation means there is no transformation so the reference image is unchanged. The four surrounding planes show the linear transformations needed to register images to the left, to the right,

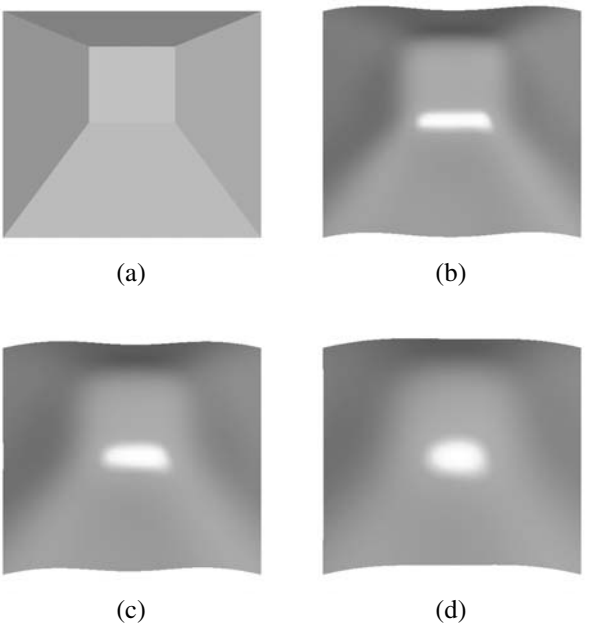


Fig. 9.1 (a) Five local linear transformations. (b)–(d) The global transformation obtained by a weighted sum of the linear transformations at increasing values of parameter s.

above, and below the reference image with the reference image. The weighted sum of the transformation functions at three different values of parameter s are shown in Figs 9.1b–d. When examined locally within an overlap area, the global transformation approximates the local transformation. The global transformation deforms the local transformations at their boundaries so they smoothly join and create smooth transitions between adjacent local transformations.

A transformation function transforms the coordinate system of an arbitrary image to the coordinate system of the reference image. It is best to take the center image as the reference image and transform the coordinate system of the remaining images to that of the reference image. This will reduce the maximum registration error. If one of the corner images is taken as the reference image, errors could accumulate and become large when reaching the opposite corner image in a mosaic. When the center image is taken as the reference, fewer steps are needed to get from the reference image to a boundary image, reducing the maximum registration error.

For instance, consider the nine images shown in Fig. 9.2. If image 5 is taken as the reference image, eight local transformations will be needed to register images 1&5, 2&5, 3&5, 4&5, 6&5, 7&5, 8&5, and 9&5. There will also be ten local transformations registering images 1&2, 2&3, 1&4, 2&4, 2&6, 3&6, 4&7, 4&8, 6&8, and 6&9. Therefore, in equation (9.1), $N = 18$ and $\mathbf{L}_1$–$\mathbf{L}_{18}$ will be the 18

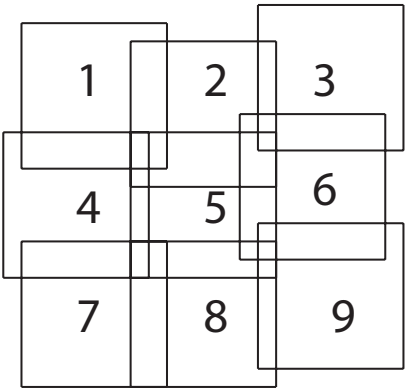


Fig. 9.2 Creating a mosaic from nine overlapping images.

local transformations linearly registering adjacent images. Note that more than two adjacent images may overlap in an area. Formula (9.1) smoothly blends all local transformations into one global transformation.

If the field of view of the camera is not small but the scene is flat, instead of using the linear transformation for local transformations, projective transformation should be used. Therefore, in formula (9.1), $f_i(x, y)$ will become a projective transformation (see section 5.2). If the scene is not flat and the camera is not very far from the scene, adjacent images cannot be registered by a linear or a projective transformation and a nonlinear transformation is needed to register them. Assuming $f_i(x, y)$ is the nonlinear transformation registering the images that create the ith overlap, in formula (9.1), the local functions become such nonlinear functions.

When the overlap area between two images is not large, local nonlinear functions may deform an image away from the overlap areas too much. A local transformation uses the coordinates of corresponding points only within the overlap area of two images and it does not use any information about the non-overlap areas in the images. Since an obtained transformation is used to resample the overlap as well as the non-overlap areas in two images, when an overlap area is too small, large errors may be obtained away from the overlap area. To minimize such errors, the local transformation functions should be chosen such that outside an overlap area, least deformation is obtained. To achieve this, a weighted linear transformation (section 5.7) can be used as a local transformation so that outside an overlap area the deformation becomes linear. If done so, the global transformation will be a weighted sum of local transformations, each of which is a weighted linear function.

9.3 BLENDING IMAGE INTENSITIES

After determining the spatial relationship between the images and standardizing the geometry of all images with respect to that of the reference image, it is required

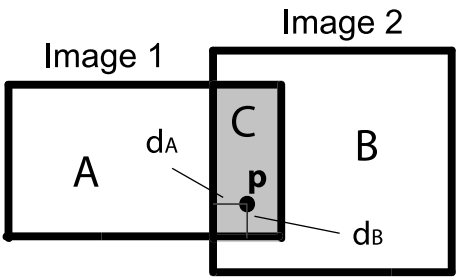


Fig. 9.3 Distance-based blending of image intensities.

to blend intensities of the images within the overlap areas. This is needed so that if overlapping areas in adjacent images have different intensities, intensities in one image gradually convert to intensities in the other image, creating a seamless boundary between the images.

In Fig. 9.3 intensities of pixels in area A are the intensities of pixels in image 1, intensities of pixels in area B are those of pixels in image 2, and intensities of pixels in area C are a blending of intensities of corresponding pixels in images 1 and 2. To make sure that the images merge seamlessly, intensities in C are determined such that they gradually convert to intensities in A and B at the C borders. If the weights of pixels in area C in images 1 and 2 are set inversely proportional to the distances of the pixels to the closest border pixels in the two images, a continuous blending of intensities will be obtained. For instance, in Fig. 9.3, if the distance of pixel $\mathbf{p}$ to the closest pixel in area A is d_A and its closest distance to area B is d_B, the intensity at $\mathbf{p}$ will be

$$I_p = \frac{I_p^A d_B + I_p^B d_A}{d_A + d_B}, \tag{9.5}$$

where I_p^A and I_p^B are intensities of images 1 and 2 at $\mathbf{p}$, respectively.

Distances d_A and d_B are found by computing the distance transforms (see section 4.4.1) of the images using the border pixels as the object pixels. If more than two images overlap, the weighted sum of all overlapping images is used in the same manner to blend the intensities.

9.4 EXAMPLES

In the first example, the nine images shown in the top three rows of Fig. 9.4 are used. These are near IR images of a rather flat urban area. The relation between the images is depicted in Fig. 9.2. Using the projective transformation as the local transformation, the mosaic shown in the bottom in Fig. 9.4 is obtained. Projective transformation is used rather than the linear transformation because the field of view of the camera is not small. Projective transformation is preferred over nonlinear transformations

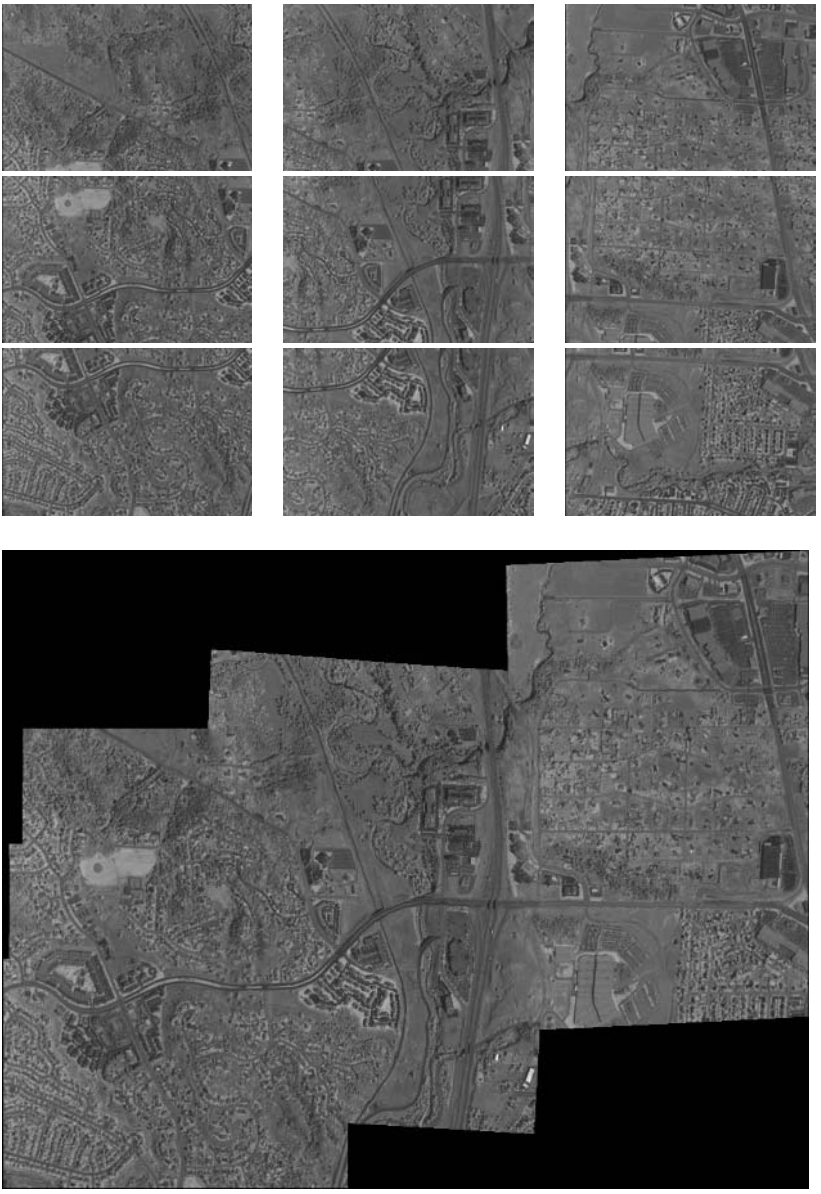


Fig. 9.4 (Top) Nine near IR images of an urban area. (Bottom) The mosaic created from the nine images using a weighted sum of local projective transformations and distance-based intensity blending.

such as thin-plate splines because the scene is relatively flat. Use of a more complex nonlinear transformation will not only slow down the registration process, but it may also deform the images unnecessarily. Projective transformation has eight unknown parameters which can be determined if the coordinates of at least four corresponding points in adjacent images are known. If more than four correspondences are available, the parameters of the transformation are determined by the least-squares method.

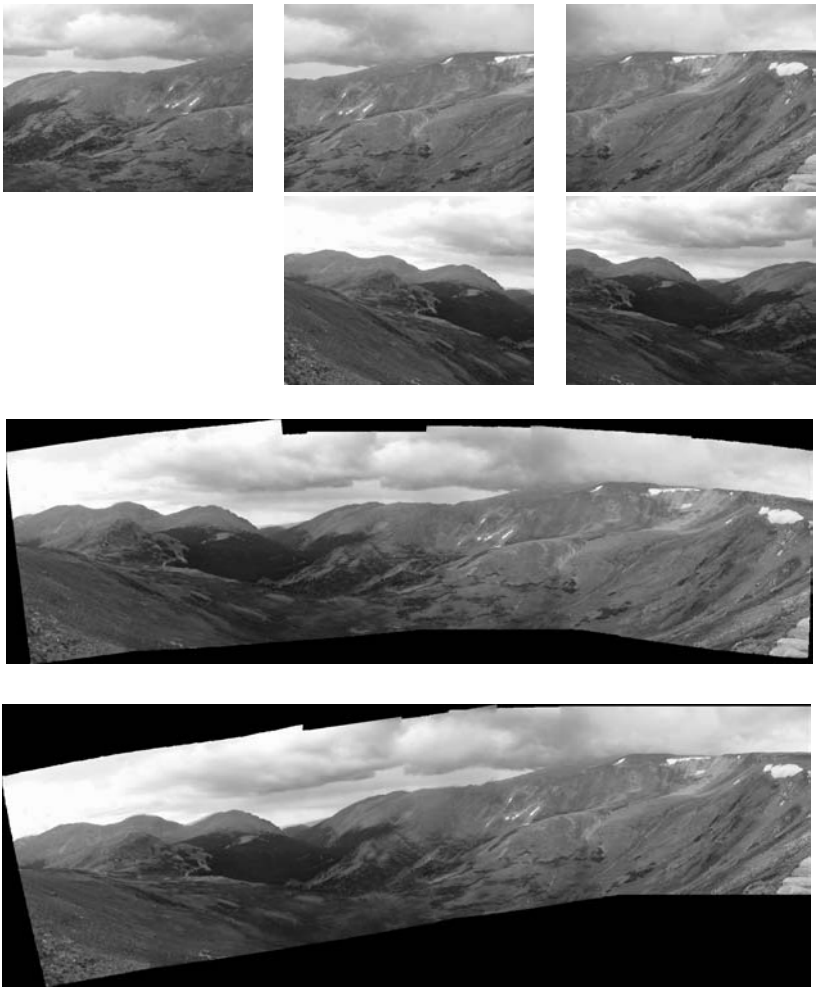


Fig. 9.5 In the first row from right to left are images 1, 2, and 3. In the second row from right to left are images 4 and 5. The mosaic obtained by using image 3 as the reference is shown in the third row. The mosaic obtained by using image 1 as the reference is shown in the bottom.

If a sufficient number of correspondences is not found between adjacent images, the local transformation for registering the images is not used in formula (9.1). For instance, in the above example, sufficient corresponding points were not found between images 1&5, 3&5, 7&5, and 9&5. Therefore, the transformations representing registration of those images were not included in formula (9.1). The process was still able to combine the images because the remaining local transformations were able to relate all images to the reference image. If too many local transformations are not included in the global transformation, some images may become disconnected from the rest, making the creation of a mosaic from the images impossible. In such a situation, images with small overlaps may be manually aligned or corresponding points in overlap areas may be manually selected to assist the process.

As a second example, the five images shown in the top two rows in Fig. 9.5 are used. Because considerable variations exist in scene depth and the camera is not very far from the scene, nonlinear local transformations are needed to register adjacent images. Here, image 3 is taken as the reference image and images 1, 2, 4, and 5 are transformed to register with image 3. The mosaic obtained by blending the local transformations according to (9.1) is shown in the third row of Fig. 9.5.

Since each overlap area in the images in Fig. 9.5 is shared by only two images, the mosaic can be created gradually like making a mosaic of a video sequence. This does not require determining a global transformation. Transformations are determined locally and the images are merged one by one starting from image 1 until all images are merged. The mosaic obtained in this manner is shown in Fig. 9.5 in the bottom row. Compared to the mosaic obtained by using image 3 as the reference and using a global transformation, we see that when the mosaic is created from only local transformations, local errors accumulate and propagate at a faster pace than when the center image is used as the reference and a global transformation is used to integrate the images. These images exhibit considerable intensity differences, due to the fast moving clouds, and change in scene illumination as a result. The intensity blending process has gradually converted intensities in one image to intensities in images adjacent to it.

9.5 MOSAICKING RANGE IMAGES

If the provided images carry range information, the range information alone or together with the intensity information may be used to globally register the images and create a mosaic. An algorithm that is widely used in registration of range images is the iterative closest point (ICP) [31]. Two images are initially approximately aligned by manual or automatic means. The registration is then refined by determining the sum of distances of points in one image to the points closest to them in another image. One image is shifted over the other in the direction that maximally reduces the sum of distances. The process is repeated until a minimum is reached in the sum.

The ICP algorithm requires that the images be from a rigid scene and that camera distortions be negligible so that the two images can be registered by a rigid transformation. The transformation that preserves rigidity is the transformation of the

Cartesian coordinate system (see section 5.1) with scale factor of 1. When determining the distances, those distances that are obtained by mapping points in one image to boundary points in another image are not used in the calculations. This is done to limit the sum of distances to only those that fall within the overlap area.

Errors in registration of two images could propagate to other images and become larger as new images are added to the mosaic. The rigid transformation between two overlapping range images can be written as

$$\mathbf{P} = \mathbf{AQ}, \tag{9.6}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are corresponding points in the images and $\mathbf{A}$ is the rigid transformation. Transformation $\mathbf{A}$ can be determined by the ICP algorithm for any two images that have some overlap.

To make a mosaic of a set of overlapping range images, one of the images is taken as the reference and the coordinate systems of all other images are transformed to the coordinate system of the reference image. This process is similar to mosaicking intensity images with the exception that in range images the points are in 3-D and have three coordinates, while in intensity images the points are in 2-D and have two coordinates. First, images adjacent to the reference image are registered and combined with the reference image. Let's suppose the transformation for registering image i, that is adjacent to the reference image, with the reference image is $\mathbf{A}_i$ (see Fig. 9.6). After this registration, the coordinate system of image i is changed so that points coinciding in the reference image and in image i will have the same coordinates.

Next, images that are adjacent to the images just processed are considered. For instance, if image j is adjacent to image i, the coordinate system of image j is transformed so that coinciding points in image j and image i have the same coordinates. The registration process transforms the coordinate system of image j to the coordinate system of image i and, since the coordinate system of image i is already transformed to the coordinate system of the reference image, the process, in effect, transforms the

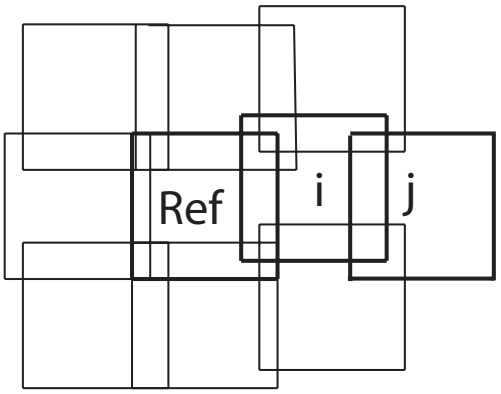


Fig. 9.6 Rigid registration of range images.

coordinate system of image j to the coordinate system of the reference image. In this manner, the coordinate systems of all images are transformed to that of the reference image.

Each local transformation may involve errors and after a few steps errors may accumulate and become large. To minimize propagation of registration errors from one image to the next, it is required to minimize the distance of individual images to the reference image. To achieve this, instead of one reference image, two reference images may be taken. Images closer to one reference image are combined with that reference image and the process is repeated until all images are merged either with one reference image or the other. Then, the mosaics obtained from the two reference images are combined to create a larger mosaic. This process can be generalized using a larger number of reference images and reduce the distance between an image and the closest reference image.

If m images are given, every other image may be initially taken as the reference image. Each reference image can then be combined with the image adjacent to it to obtain a larger image. This will reduce the number of images from m to $m/2$. Again, every other image can be considered a reference image and combined with the image adjacent to it to produce $m/4$ larger images. The process can be repeated until all images are merged into a single mosaic. Each step produces larger images, creating larger overlaps and producing more accurate matches. Note that this pairwise registration does not require a global transformation to register the images. At each step, adjacent images are registered, two at a time, producing larger images until all images are merged into one large image. The merging process has the structure of a binary tree, the leaves being the individual images and the root being the mosaic.

Unlike mosaicking of intensity images where image intensities in overlapping areas are blended, in mosaicking of range images the process resamples the range points in their individual coordinate systems to a common coordinate system. Therefore, in an overlap area, multiple points or a thick layer of points may be obtained. Also, unlike intensity images that do not contain holes, range images may contain holes due to self occlusion, the dark texture of the object during laser scanning, or for other reasons. Therefore, even after combining images from different views, holes may appear in the combined image. To fill in the holes and create a thin layer of points in the output, a surface is fitted to the combined range data set.

An example of image mosaicking using range images is shown in Fig. 9.7. The individual range images are shown in Figs 9.7a–e. These are range scans of a tooth taken from different views. First, images 9.7b and 9.7c and images 9.7d and 9.7e are combined to obtain larger images. The two newly obtained images are then registered to create a surround image of the tooth. The surround image is then registered with image 9.7a to obtain the mosaic shown in Fig. 9.7f. The obtained mosaic contains some holes due to the presence of holes in the original images. It also contains multiple coinciding points and areas where thick layers of points exist. Fitting a RaG surface [153] to the combined data set produces the tooth model shown in Fig. 9.7g. Surface fitting fills in the holes and creates a surface that can be resampled to a thin layer of uniformly spaced points at a desired resolution for rendering and other purposes.

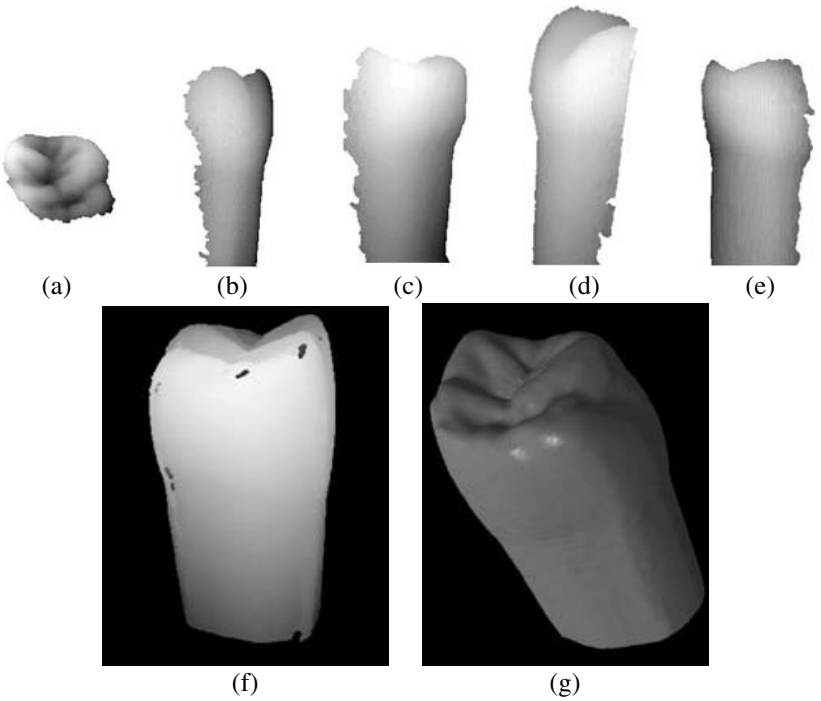


Fig. 9.7 (a)–(e) Range images showing different views of a tooth. (f) The mosaic obtained by registering and integrating the range images. (g) Surface fitting and 3-D reconstruction of the tooth.

9.6 EVALUATION

The performance of an image mosaicking algorithm depends on the performance of the local registration algorithm and the performance of the transformation function that combines all the local transformations. The performance of the local registration algorithm can be evaluated as outlined in Chapter 7, which involves determining its accuracy, reliability, robustness, and speed. The robustness measure should particularly consider the different amounts of overlap between images, characterizing accuracy and reliability as a function of the amount of overlap between images.

The performance of the global transformation involves determining the accuracy of the overall registration and the change in accuracy as the image content, size of overlap areas, and the geometric difference between images are changed. There also is a need to evaluate the speed of the algorithm as a function of the size of images and the number of overlaps. It is desired that the computational complexity of the overall algorithm be a linear function of the number of overlaps and the size of the constructed mosaic.

To evaluate the accuracy of the global transformation, test images should be prepared where the absolute coordinates of a number of control points in each image are known. After registering the images using some of the control points, the accuracy of the global transformation can then be determined using the coordinates of the remaining control points in the images. Is the error obtained by the global process larger than the errors produced by the individual local transformations? The average, root-mean-squared, or maximum global errors can be used to measure the accuracy of a mosaicking algorithm.

9.7 SUMMARY

In this chapter mosaicking of intensity images and range images was discussed. For intensity images, depending on the camera parameters and the scene content, different transformations will be needed to register the images.

- If the scene is flat, the camera is far from the scene, and the images are obtained by rotating the camera about an axis that passes through the lens center and is normal to the optical axis of the camera, adjacent images have only translational differences and can be registered by shifting one image over the other and determining the best-match position.
- If the scene is flat and the field of view of the camera is small but the camera is not far from the scene, adjacent images should be registered using the affine or linear transformation.
- If the scene is flat but the field of view of the camera is not small, adjacent images should be registered by the projective transformation.
- If the scene is not flat, adjacent images should be registered by a nonlinear transformation.

Although a rigid, affine, or projective transformation may be sufficient to register two adjacent images, it is generally understood that, when a large number of images is given, images that are not adjacent may not register well by a rigid, affine, or projective transformation and a more elaborate transformation is needed. Formula (9.1) describes a global transformation in terms of a weighted sum of local transformations, with the sum of the weights being equal to 1 everywhere in the mosaic. The weights are monotonically decreasing, ensuring that the global transformation approximates the local transformation in a local neighborhood. To ensure that minimum error is obtained in the overall registration, the center image should be taken as the reference image. This will reduce the maximum distance between all images and the reference image.

To avoid the creation of seams between adjacent images, the intensities of the images in an overlap area are blended with weights that are inversely proportional to the distances of points in the overlap area to the closest boundary points.

When registering range images, all the local transformations and the global transformation are rigid transformations. Therefore, adjacent images are rigidly registered to obtain a larger image and the process is repeated until all images merge into one

large image. A surface is then fitted to the combined data set to fill in the holes and create a continuous surface, which can then be quantized to a regular grid at a desired resolution for display and other purposes.

9.8 BIBLIOGRAPHICAL REMARKS

One of the obvious applications of mosaicking is in map building, creating a large image by piecing together small aerial images. The main focus of mosaicking methods that have appeared in the photogrammetry literature has been on finding the best seam between two images within the overlap area so the images can be cut and pasted together [115, 356]. The images are often already georectified [18] and, therefore, they approximately register. There is a need to refine the registration through correlation-based matching, however.

Kerschner [222] used two snakes, each positioned in the overlap area in an image. The snakes are made to attract to each other and position themselves to a common path where the sum of the color differences between corresponding points on the snakes is minimum. The snakes make it possible to combine two images along the same irregular path. This has the advantage that, if two images at the overlap area are not registered well, the constructed mosaic will not become blurred or show double images. It has the disadvantage that if the intensities of the images in the overlap area are different, the intensities will not gradually change from one image to the next, but rather a sharp intensity change will be observed across the seams. In order to make the seams invisible, radiometric corrections [404] may be applied to one of the images or both so corresponding points in the images have similar intensities.

In spite of the fact that projective transformation is rarely used in image registration, in image mosaicking, the overwhelming majority of the methods use the projective transformation to register images. Contrary to methods developed in photogrammetric engineering, which cut and paste image pieces to create a mosaic, in the computer vision and graphics literature, image intensities in the overlapped areas are blended to create a mosaic. This can be attributed to the fact that in photogrammetry, the parameters of the sensors and the platform are often known; therefore, not only can the images be radiometrically standardized, but their geometries can also be standardized through a rectification process [404]. Chavez *et al.* [61] used sensor specific algorithms to correct both the geometry and the intensity of images to create mosaics. Garcias and Santos-Victor [135] used camera parameters to integrate images from a rotating camera.

Bao and Xu [19] described a registration algorithm based on wavelets that is invariant of scale difference between images. The process makes it possible to register and combine images taken by different cameras or from different distances of a scene. Multi-resolution approaches to mosaicking have also been developed. Chiba *et al.* [64] first used a patch-based optical flow estimation to obtain the point correspondences. Then the points were triangulated and triangular regions were merged. In a coarse-to-fine approach, Hsieh [197] first found the translational difference between

images through edge alignment and then refined the registration by minimizing errors at corresponding features.

Cortelazzo and Lucchese [77] developed a two-step process to create image mosaics where, in the first step, the images were approximately aligned using the affine transformation and, in the second step, the registration was refined using the projective transformation. Zhou *et al.* [430] and Hoshino and Kourogi [194] created an image mosaic using only the affine transformation. Shum and Szeliski [358] developed a mosaicking method where, after aligning the adjacent image pairs, a global process is used to refine the local registrations by minimizing the sum of squared intensity differences between overlapping areas. Marzotto *et al.* [265] achieved global alignment by exploiting the topological structure of the overlaps and a bundle adjustment step, which minimizes the total misalignment of a pre-defined set of grid points on the mosaic.

Algorithms for mosaicking range images can be categorized into feature-based methods and methods that work with raw range values. Examples of features are ridge curves [301] and local peak points in single-view range images [86]. Matching of range features is achieved through subgraph isomorphism [85]. Ramalingam and Lodha [320] registered range images using point features obtained by a corner detector. They then refined the registration using all range points in an ICP algorithm [31]. Gunadi *et al.* [173] segmented range images into low curvature regions and, by determining the correspondence between the regions, aligned the images approximately. The registration was then refined by the ICP algorithm using all range points.

Many improvements to the original ICP algorithm [31] have been proposed. Steps have been taken to make the ICP algorithm faster and more resistant to outliers. To speed up the ICP algorithm, coarse-to-fine approaches have been proposed by Jost and Hügli [215] and Schönfeld *et al.* [347]. To improve the geometric stability of the ICP algorithm, Gelfand *et al.* [138] devised a point selection strategy that samples those points in the input that provide the best convergence of the algorithm to the correct pose. The sampling strategy estimates the transformation that produces unstable sliding of one surface over another and picks those points that best constrain this sliding. To avoid convergence to a local minimum, Silva *et al.* [360] searched in the parameter space for the best-match position with a genetic algorithm. Contrary to the ICP algorithm, a genetic algorithm does not require pre-alignment of the images.

To reduce registration errors, Tubić *et al.* [389] incrementally captured images that had large overlaps and small view-angle differences. El-Hakim *et al.* [109] suggested user interaction during the registration process when the scene is very large and the overlaps between adjacent scans are small. To achieve a higher matching accuracy, Wyngaerd and Gool [417] as well as Smith and Elstrom [362] suggested use of both the color and range information in matching. This is especially helpful when the surfaces to be registered are geometrically smooth but contain highly detailed textures. In a hierarchical method described by Ikemoto *et al.* [203], linear features that are less sensitive to noise and warping are used to register the images. To avoid accumulation of registration errors when a large number of range images are present, Pulli [317] used a method that obtains a consensus registration among all images, rather than between two images, averaging errors among the images.

10

Stereo Depth Perception

When two images of a scene are obtained by two similar cameras that are horizontally displaced, the images possess certain constraints that simplify registration of the images. If a single camera is horizontally moved while obtaining the images, the process is called *motion stereo* [286]. The main objective in (motion) stereo image registration is to recover the 3-D geometry of the scene. When stereo images are used in a registration, the left image is usually taken as the reference image and the right image is taken as the sensed image. Stereo image registration is often referred to as stereo correspondence.

Although, in image registration, occlusion is generally considered nonexistent, when dealing with stereo images occlusion is often present. The scene being imaged may have depth discontinuities with scene parts that appear in only one of the images. Since overlaying of stereo images does not serve a purpose, there is no need to find a transformation function that aligns the images. Instead, an interpolation function is needed to estimate depth values not determined due to occlusion and other reasons.

Stereo vision is perhaps the most studied subject in computer vision. This can be attributed to two facts. First, binocular stereo is the most versatile method for recovering a scene's geometry without using special lighting. Second, The correspondence problem has not yet been fully solved, although algorithms that find stereo correspondences have become more robust.

In this chapter, the relation between scene depth and corresponding points in stereo images is described, a camera calibration method that relates the scene coordinates to the image coordinates is outlined, a rectification procedure that converts images obtained by cameras with converging optical axes to images with parallel optical axes is discussed, various stereo constraints and correspondence algorithms are reviewed,

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

and scene reconstruction from scattered scene points obtained by stereo correspondence is described.

10.1 STEREO CAMERA GEOMETRY

The simplest stereo camera geometry has optical axes that are parallel and are normal to the *baseline*, the line connecting the lens centers of the cameras. This geometry is depicted in Fig. 10.1. The image of an object point (X, Y, Z) is formed at (x_L, y) in the left image and at (x_R, y) in the right image. Let's suppose the coordinates of the object point when measured with respect to the left camera coordinate system are (X_L, Y, Z) and when measured with respect to the right camera coordinate system are (X_R, Y, Z). The stereo-camera coordinate system is considered to be midway between the left and right camera coordinate systems. Assuming idealistic (pin-hole) cameras, the relation between the object point when measured with respect to the left camera coordinate system and the stereo-camera coordinate system can be written by

$$X_L = X + \frac{B}{2}, \tag{10.1}$$

$$Y_L = Y, \tag{10.2}$$

$$Z_L = Z, \tag{10.3}$$

where B is the baseline length (the distance between the lens centers of the cameras). Similarly, the relation between the coordinates of a point measured with respect to the right camera coordinate system and the stereo-camera coordinate system can be written by

$$X_R = X - \frac{B}{2}, \tag{10.4}$$

$$Y_R = Y, \tag{10.5}$$

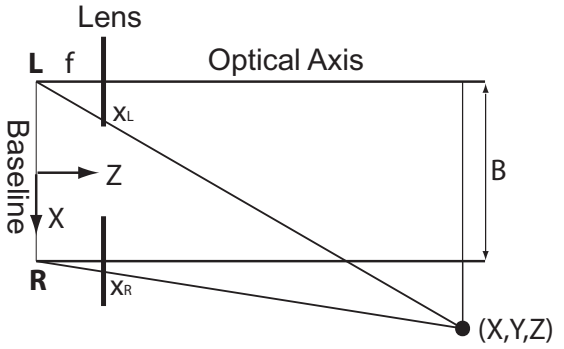


Fig. 10.1 A simple stereo camera geometry with parallel optical axes.

$$Z_R = Z. \tag{10.6}$$

From similar triangles in Fig. 10.1, for the left camera we have

$$\frac{x_L}{f} = \frac{X_L}{Z_L} = \frac{X + B/2}{Z}, \tag{10.7}$$

and similarly for the right camera we have

$$\frac{x_R}{f} = \frac{X_R}{Z_R} = \frac{X - B/2}{Z}. \tag{10.8}$$

These can be written as

$$X + \frac{B}{2} = x_L \frac{Z}{f}, \tag{10.9}$$

$$X - \frac{B}{2} = x_R \frac{Z}{f}. \tag{10.10}$$

Removing X from (10.9) and (10.10), we obtain

$$Z = \frac{Bf}{x_L - x_R}. \tag{10.11}$$

The quantity $(x_L - x_R)$, which is the horizontal distance between corresponding points in the images, is known as *disparity*, and f represents the focal length of the cameras. Relation (10.11) shows that depth is inversely proportional to disparity. The larger the disparity the closer the object point will be to the camera baseline, and the smaller the disparity the more distant the object point will be from the baseline. When disparity is zero, the object point is very far from the viewer. Relation (10.11) also shows that for the same depth, a larger disparity is obtained when a camera system with a larger focal length or a larger baseline length is used. A larger disparity allows measurement of depth with finer increments and, consequently, with a higher accuracy.

Measurement of depth from formula (10.11) requires knowledge about the coordinates of corresponding points in the images. In the pin-hole camera set up shown in Fig. 10.1, corresponding points are horizontally displaced. This implies that to determine the correspondence to a point in the left image a search is needed only along the same scanline in the right image. A one-dimensional search, therefore, is sufficient to find the correspondences. This is in contrast to matching non-stereo images, which requires two-dimensional searches to find the correspondences.

Note that in formula (10.11) coordinates of points in an image are measured with respect to a coordinate system whose origin is at the intersection of the optical axis with the image plane. If it is necessary to measure the image coordinates with respect

to the lower left corner of the images, x_L and x_R should be replaced with $x_L - x_{CL}$ and $x_R - x_{CR}$, respectively, where x_{CL} and x_{CR} are the x-coordinates of the left and right *piercing points* (the intersections of the optical axes of the cameras with the image planes). Also note that formula (10.11) uses the same unit of measurement to describe both the object and the image coordinates. In practice, however, object coordinates are measured with units such as *cm* and *mm* while image coordinates are measured in pixels. To relate image coordinates to object coordinates, a calibration process is needed. Camera calibration is discussed in the next section.

When the optical axes of the cameras are parallel, the overlap between the left and right images decreases as objects get closer to the baseline. To increase this overlap, a camera setup with converging optical axes is needed. By letting the intersection point of the optical axes of the cameras with the image planes lie near the center of the object of interest, the most overlap can be obtained between images of the object. The geometry of a stereo camera system with converging optical axes is depicted in Fig. 10.2.

The relation between the stereo-camera coordinate system XYZ and the coordinate system of the left camera $X_L Y_L Z_L$ can be written by a rotation and a translation. First, rotate the left-camera coordinate system about the Y_L-axis by θ so that the Z_L-axis becomes parallel to the Z-axis. Let's denote this rotation by $R_Y(\theta)$. Second, translate the obtained coordinate system by $B/2$ so it will align with the stereo-camera coordinate system. Let's denote this translation by $T_X(B/2)$. The transformation that brings the left-camera coordinate system to the stereo-camera coordinate system can, therefore, be written as $R_Y(\theta)T_X(B/2)$, or

$$\begin{bmatrix} X_L \\ Y_L \\ Z_L \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & B/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (10.12)$$

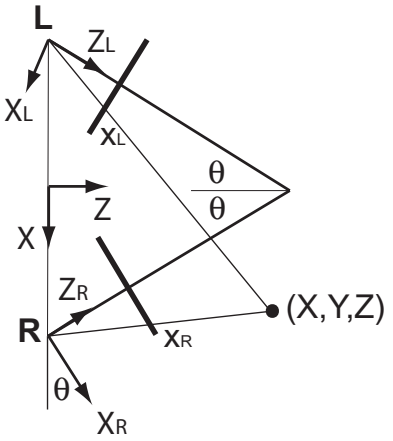


Fig. 10.2 A stereo camera geometry with converging optical axes.

which after simplification becomes

$$X_L = X\cos\theta - Z\sin\theta + \frac{B}{2}\cos\theta, \tag{10.13}$$

$$Y_L = Y, \tag{10.14}$$

$$Z_L = X\sin\theta + Z\cos\theta + \frac{B}{2}\sin\theta. \tag{10.15}$$

From the pin-hole camera geometry given by formula (10.7) and using X_L and Z_L from formulas (10.13) and (10.15), we obtain

$$\frac{x_L}{f} = \frac{X_L}{Z_L} = \frac{(X + B/2)\cos\theta - Z\sin\theta}{(X + B/2)\sin\theta + Z\cos\theta}. \tag{10.16}$$

Similarly, the relation between the stereo-camera coordinate system XYZ and the right-camera coordinate system $X_RY_RZ_R$ involves a rotation and a translation. First, rotate the right-camera coordinate system about the Y_R-axis by $-\theta$ so the Z_R-axis becomes parallel to the Z-axis: $R_Y(-\theta)$. Second, translate the obtained coordinate system by $-B/2$ so it will align with the stereo-camera coordinate system: $T_X(-B/2)$. The overall transformation, therefore, is $R_Y(-\theta)T_X(-B/2)$, or

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -B/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{10.17}$$

which, after simplification and use of the pin-hole camera geometry given by (10.8), reduces to

$$\frac{x_R}{f} = \frac{X_R}{Z_R} = \frac{(X - B/2)\cos\theta + Z\sin\theta}{-(X - B/2)\sin\theta + Z\cos\theta}. \tag{10.18}$$

If x_L, x_R, f, B, and θ are known, the coordinates X and Z of an object point can be determined from relations (10.16) and (10.18). Note that when $\theta = 0$, equations (10.16) and (10.18) become

$$\frac{x_L}{f} = \frac{X + B/2}{Z}, \tag{10.19}$$

$$\frac{x_R}{f} = \frac{X - B/2}{Z}, \tag{10.20}$$

and, by eliminating X from these equations,

$$Z = \frac{Bf}{x_L - x_R}, \tag{10.21}$$

is obtained, which is the same as equation (10.11), which was obtained for a camera system with parallel optical axes.

To relate the Y-coordinate of an object point to its image coordinates, note that from the pin-hole camera geometry we have

$$\frac{y_L}{f} = \frac{Y_L}{Z_L} = \frac{Y}{(X + B/2)\sin\theta + Z\cos\theta}, \tag{10.22}$$

$$\frac{y_R}{f} = \frac{Y_R}{Z_R} = \frac{Y}{-(X - B/2)\sin\theta + Z\cos\theta}, \tag{10.23}$$

or

$$Y = \frac{y_L}{f}\left[(X + B/2)\sin\theta + Z\cos\theta\right], \tag{10.24}$$

$$Y = \frac{y_R}{f}\left[-(X - B/2)\sin\theta + Z\cos\theta\right]. \tag{10.25}$$

By knowing the coordinates of corresponding points (x_L, y_L) and (x_R, y_R) and by using either (10.24) or (10.25), the Y-coordinate of the object point can be determined.

Parameters f, B, and θ are determined through a calibration process by using a number of known points in the scene and their images in the stereo images. This is explained in the next section.

10.2 CAMERA CALIBRATION

If lens distortions do not exist, the relation between points in a scene and their projections in an image can be written by a projective transformation. If lens distortions exist, the images should be corrected before using them. The procedure to correct image deformations from lens distortion is described by Goshtasby [151] and Fryer and Brown [133]. The relation between a scene point (X, Y, Z) and its image (x_L, y_L) in the left camera may be written by

$$x_L = \frac{m_1X + m_2Y + m_3Z + m_4}{m_5X + m_6Y + m_7Z + 1}, \tag{10.26}$$

$$y_L = \frac{m_8X + m_9Y + m_{10}Z + m_{11}}{m_5X + m_6Y + m_7Z + 1}, \tag{10.27}$$

or

$$m_1X + m_2Y + m_3Z + m_4 + m_5(-x_LX) + m_6(-x_LY) + m_7(-x_LZ) = x_L, \tag{10.28}$$

$$m_5(-y_LX) + m_6(-y_LY) + m_7(-y_LZ) + m_8X + m_9Y + m_{10}Z + m_{11} = y_L. \tag{10.29}$$

Equations (10.28) and (10.29) are linear in parameters $m_1 \ldots m_{11}$ and can be solved if the coordinates of at least 5.5 points (5 points and the x or the y coordinate of

the 6th point) in the scene and in the left image are known. If more than 5.5 points are available, parameters $m_1 \ldots m_{11}$ can be determined by the least-squares method. Once parameters $m_1 \ldots m_{11}$ are determined, relations (10.26) and (10.27) can be used to determine the positions of points in the left image from the positions of the object points.

Similarly, the relation between scene points and points in the right image can be written as

$$x_R = \frac{n_1 X + n_2 Y + n_3 Z + n_4}{n_5 X + n_6 Y + n_7 Z + 1}, \tag{10.30}$$

$$y_R = \frac{n_8 X + n_9 Y + n_{10} Z + n_{11}}{n_5 X + n_6 Y + n_7 Z + 1}, \tag{10.31}$$

and parameters $n_1 \ldots n_{11}$ can be determined using the coordinates of at least 5.5 points in the scene and in the right image. Relations (10.30) and (10.31) determine the coordinates of a point in the right image from the coordinates of an object point in 3-D. Given an image point, however, there are many points in 3-D that satisfy (10.30) and (10.31). Any point along the ray connecting the image point to the lens center qualifies as the answer. To determine a unique object point for each image point, coordinates of corresponding points in both left and right images must be used. Three of the four relations (10.26), (10.27), (10.30), and (10.31) can be used to uniquely relate points in 3-D to points in the two images. For instance, from the first three relations we have

$$X(m_5 x_L - m_1) + Y(m_6 x_L - m_2) + Z(m_7 x_L - m_3) = m_4 - x_L, \tag{10.32}$$

$$X(m_5 y_L - m_8) + Y(m_6 y_L - m_9) + Z(m_7 y_L - m_{10}) = m_{11} - y_L, \tag{10.33}$$

$$X(n_5 x_R - n_1) + Y(n_6 x_R - n_2) + Z(n_7 x_R - n_3) = n_4 - x_R. \tag{10.34}$$

Given the coordinates of corresponding points (x_L, y_L) and (x_R, y_R) in the images, and knowing the calibration parameters $m_1 \ldots m_{11}$ and $n_1 \ldots n_7$, the coordinates of the point in 3-D can be determined by solving (10.32) – (10.34). More accurate coordinates can be obtained if X, Y, and Z are determined by the least squares method using all four relations (10.26), (10.27), (10.30), and (10.31).

Calibration parameters $m_1 \ldots m_{11}$ and $n_1 \ldots n_{11}$ contain information about the focal length f of each camera, the baseline length B, and the *vergence angle* (the angle between the optical axes of the cameras). The calibration process that was outlined above actually allows the cameras to have different focal lengths, different aspect ratios, and different piercing points. Also, the orientational difference between the cameras is not limited to the Y axis but can be with respect to the XYZ axes as well.

To determine parameters $m_1 \ldots m_{11}$ and $n_1 \ldots n_{11}$, the coordinates of at least 5.5 points in the scene and in the images are needed. For a given camera set up, these parameters have to be determined only once. Figure 10.3 shows a calibration box prepared for this purpose. Clearly identifiable marks are made on three orthogonal

(a) (b)

Fig. 10.3 (a), (b) Images of a calibration box obtained by a stereo camera system.

planes that form a corner. The sides of the box sharing the corner are used as the axes of the world coordinate system. Coordinates of the marks are carefully measured in this coordinate system and recorded. The images of the marks in the left and right images are also determined. Each time one or more of the camera parameters is changed, the calibration box is placed in front of the stereo camera and a pair of images is obtained. The coordinates of the marks in 3-D and in the images are then used to determine the camera parameters.

Corresponding marks in stereo images can be selected interactively. A template-matching or line-fitting process can accurately locate a desired mark and determine its position with subpixel accuracy. Instead of crosses, circular regions may also be used. Centers of gravity of elliptic regions obtained in the images make determination of the center of gravity of the marks in any view possible with subpixel accuracy.

10.3 IMAGE RECTIFICATION

Rectification is the process of transforming the geometry of images obtained by cameras with converging optical axes to images obtained by cameras with parallel optical axes. Consider the images before and after rectification shown in Fig. 10.4. A ray from an object point $\mathbf{P}$ intersects the left image plane before and after rectification at (x, y) and (X, Y), respectively. Each point in the image before rectification can be connected to the lens center and extended to intersect the image after rectification. In this manner, for each point in the image before rectification, the corresponding point in the image after rectification can be determined. The images before and after rectification can be considered images obtained by two cameras with coinciding lens centers but different image planes. The relation between such images is a projective transformation:

$$x = \frac{a_1 X + a_2 Y + a_3}{a_4 X + a_5 Y + 1}, \tag{10.35}$$

$$y = \frac{a_6 X + a_7 Y + a_8}{a_4 X + a_5 Y + 1}. \tag{10.36}$$

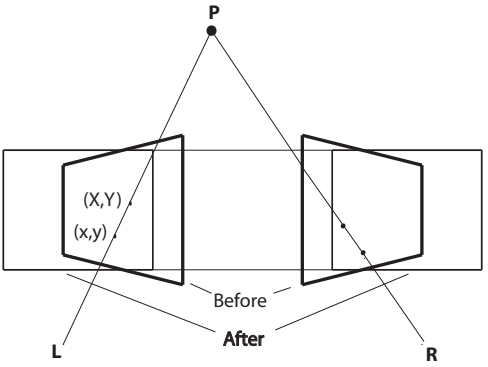


Fig. 10.4 Using two projective transformations, images obtained by converging optical axes are transformed to images as if obtained by cameras with parallel optical axes.

Parameters $a_1 \ldots a_8$ are determined using the coordinates of a minimum of four corresponding points in the images before and after rectification. Since the main objective in image rectification is to produce horizontal epipolar lines (intersections of a plane formed by the baseline and a point in the scene with the image planes), all that needs to be done is to select two epipolar lines in the image before rectification and map them to two horizontal lines in the image after rectification as shown in Fig. 10.5. For instance, consider the following correspondences for the left image before and after rectification.

$$X_1 = x_1; \qquad Y_1 = \frac{y_1 + y_2}{2}; \tag{10.37}$$

$$X_2 = x_2; \qquad Y_2 = \frac{y_1 + y_2}{2}; \tag{10.38}$$

$$X_3 = x_3; \qquad Y_3 = \frac{y_3 + y_4}{2}; \tag{10.39}$$

$$X_4 = x_4; \qquad Y_4 = \frac{y_3 + y_4}{2}. \tag{10.40}$$

These correspondences will keep the width of the images before and after rectification the same, but they will scale the images vertically to map non-horizontal epipolar lines

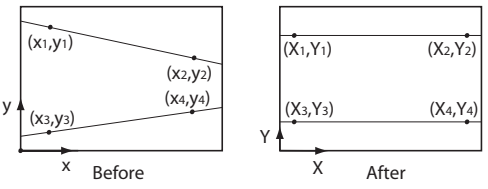


Fig. 10.5 Images before and after rectification.

to horizontal epipolar lines. To create a rectified image, for each pixel (X, Y) in the rectified image, the corresponding pixel (x, y) in the image before rectification is found using relations (10.35) and (10.36). Then the intensity at (x, y) is read and saved at (X, Y).

The process is repeated in the same manner for the right image. In order for corresponding epipolar lines in the rectified left and right images to represent the same scanline, it is required that corresponding epipolar lines in the images before rectification map to the same scanline in images after rectification. Therefore, the Y-coordinates used in (10.37) – (10.40) should be used when rectifying both the left and the right images.

An example of image rectification is given in Fig. 10.6. Figures 10.6a and 10.6b show images obtained by a stereo camera system with converging optical axes. Horizontal lines were drawn on a cardboard and placed vertically in front of and parallel to the baseline of the stereo camera system, and a pair of images were obtained. Corresponding lines in these images are actually corresponding epipolar lines. In order to make the epipolar lines appear horizontal, two pairs of points are selected on two epipolar lines according to (10.37) – (10.40) in the left image, the projective transformation defined by (10.35) and (10.36) is determined, and the left image is resampled. This is shown in Fig. 10.6c. The same is repeated for the right image to obtain Fig. 10.6d. Corresponding epipolar lines in the rectified images now show the same scanline in both left and right images. Once the parameters of the projec-

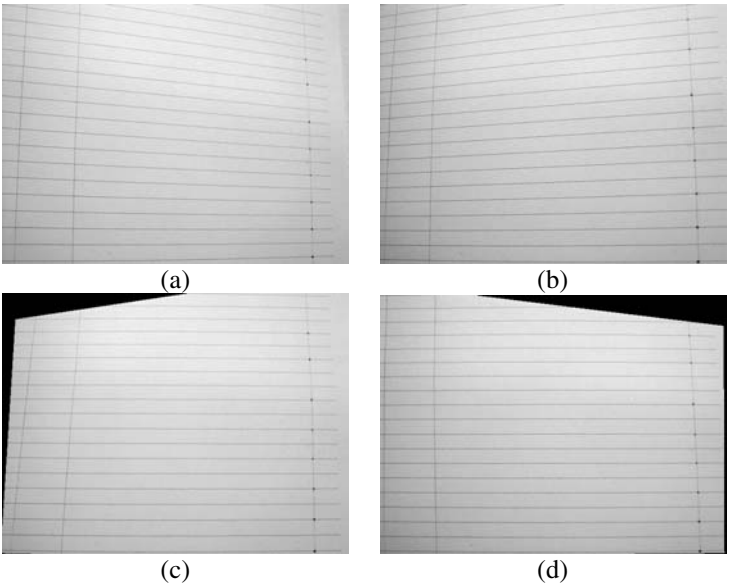


Fig. 10.6 (a), (b) Images of horizontal lines in 3-D parallel to the baseline obtained by a stereo camera system with converging optical axes. (c), (b) Rectification of the images so that corresponding epipolar lines fall on the same scanline in the images.

tive transformations that rectify the left and right images are determined for a stereo camera system, they can be used to rectify all images obtained by that camera system.

Note that after finding corresponding points in the rectified images, the coordinates of the points should be transformed back to the points before rectification by substituting the (X, Y) coordinates of each point in the left rectified image into (10.35) and (10.36) and finding the (x, y) coordinates of the point in the left image before rectification. This should be done similarly for the right image. Corresponding points obtained in the images before rectification should then be used in equations (10.32) – (10.34) to find the 3-D object coordinates.

10.4 THE CORRESPONDENCE PROCESS

If the stereo camera system is rather far from the scene or the vergence angle is small, occlusions can be considered negligible. However, if the camera system is not far from the scene or the vergence angle is not small, occlusions may be inevitable and local geometric difference between some areas in the images may be large, making determination of the correspondences difficult. The constraints holding between stereo images, however, may be used to assist the correspondence process.

10.4.1 Constraints in stereo

The constraints used in stereo correspondence describe the:

- epipolar geometry;
- uniqueness of correspondences;
- ordering of correspondences;
- continuity of disparity;
- photometric compatibility of corresponding points;
- geometric compatibility of corresponding features.

10.4.1.1 Epipolar constraint The plane obtained from the camera lens centers and a point in the scene is called an *epipolar plane*. The intersections of an epipolar plane with the image planes are *epipolar lines* (see Fig. 10.7). Let's suppose an object point $\mathbf{A}$ in the scene produces images $\mathbf{a}_L$ and $\mathbf{a}_R$ in the left and right images. The geometry in Fig. 10.7 shows that, to find the correspondence to point $\mathbf{a}_L$, it is sufficient to search in 1-D along line $\mathbf{e}_R\mathbf{a}_R$. If camera parameters are known, the intersection of the plane passing through the lens centers and point $\mathbf{a}_L$ with the right image plane can be determined. This is the line along which the search should be performed. The line passing through $\mathbf{e}_L$ and $\mathbf{a}_L$ and the line passing through $\mathbf{e}_R$ and $\mathbf{a}_R$ form corresponding epipolar lines. The intersections of the baseline with the image planes are called *epipoles*. Note that all epipolar planes share the baseline, and therefore, all epipolar lines in an image pass through the epipole in that image. Also note that when the image planes are parallel to the baseline, the epipoles will be at infinity and all epipolar lines will become parallel to the baseline. Therefore, taking

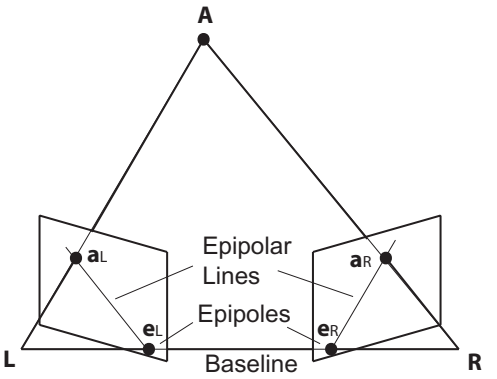


Fig. 10.7 The epipolar constraint in stereo images.

a point in the left image, the point corresponding to it lies on the same scanline in the right image.

Given a point $\mathbf{a}_L = (x_L, y_L)$ in the left image, from relations (10.26) and (10.27), coordinates X and Y of the object point can be determined in terms of Z and substituted into relations (10.30) and (10.31). This will produce two equations relating x_R and y_R to Z. Eliminating Z from the two equations will produce a linear equation in x_R and y_R that shows the epipolar line along which a search should be made to find the point corresponding to (x_L, y_L). In the preceding section, it was shown how to rectify the images so that corresponding epipolar lines appear on the same scanline in both left and right images. Therefore, taking a point in the left image, there will be a need to search for the point along the same scanline in the right image. Epipolar constraint reduces 2-D searches into 1-D searches, not only speeding up the correspondence process but also reducing the likelihood of finding wrong correspondences.

10.4.1.2 Uniqueness constraint For each point in the left image there exists a maximum of one point in the right image and for each point in the right image there exists a maximum of one point in the left image. In reality, due to view-angle difference between the cameras and the 3-D nature of a scene, corresponding patches in the images may have different sizes, implying that in the digital space it is possible to find a point in one image that corresponds to more than one point in another image. Physically, however, only one of the correspondences is possible. Therefore, if many to one correspondence is obtained, only one (the middle) point should be selected and the rest should be discarded. This may produce holes in the created depth map, which will be later filled by interpolation.

It is possible to have a point in one image that is occluded in the other image. For occluded points, a correspondence does not exist. Occluded points produce holes in the created depth map. A robust stereo-correspondence algorithm should detect the occluded points and avoid finding correspondences for them.

10.4.1.3 Ordering constraint If points along an epipolar line belong to a rigid object, the order of points along corresponding epipolar lines in the images will be the same. Therefore, after finding corresponding points along corresponding epipolar lines, this constraint can be used to eliminate some of the mismatches. If the order of corresponding points along corresponding epipolar lines is the same, it does not mean that all correspondences are correct. However, if the order of correspondences is not the same, we can say for sure that some of the correspondences are wrong. This constraint can, therefore, be used to detect potential mismatches and eliminate them. It is always better to have fewer correspondences, all of which are correct, than to have many correspondences, some of which are incorrect. The depth values at points where correspondences are not found can be estimated from the depth values of nearby points by interpolation. Errors in estimation of missing depth values are usually much smaller than errors in depth values obtained from incorrect correspondences.

10.4.1.4 Continuity constraint If the scene is rigid, points in the scene near each other appear near each other in both images. Therefore, depth values or disparities along an epipolar line vary continuously, except at points where each side of the point represents a different object. If the difference of disparities of adjacent points along an epipolar line become larger than a threshold value, either the point represents a depth discontinuity or the point represents a wrong correspondence. This is an indication that a mismatch might have occurred. The continuity constraint is not limited to points along an epipolar line. Adjacent points belonging to different epipolar lines should also satisfy this constraint. Correspondence algorithms that match individual epipolar lines do not take advantage of this constraint fully. There should be interaction between the matching of adjacent epipolar lines to ensure that the continuity constraint is satisfied not only along epipolar lines but also across them.

10.4.1.5 Photometric compatibility constraint A correspondence pair of points in stereo images represents the same object point in 3-D. Therefore, if the object point does not belong to a specular surface, corresponding image points will have very similar intensities. This is the basis for correlation-based template-matching methods. This constraint can also be used to detect the incorrect correspondences. When two matching windows have similar intensities it is not a guarantee that a correct correspondence is obtained. However, if two matching windows do not have similar intensities, it can be said with great certainty that the windows do not correspond to each other and, therefore, their centers will not represent the same scene point. Correspondences that produce similarities that are not high enough can be considered questionable and removed from the set of correspondences. The holes obtained as a result can be filled later by interpolation.

10.4.1.6 Geometric compatibility constraint Shapes of geometric features do not change greatly from one image to another. For instance, if small regions and line intersections are the features used in matching, regions in the images should be matched separately from line intersections. This constraint makes it possible to select different types of features and establish correspondence between each feature

type separately, reducing the combination of features that are tested. The geometric compatibility constraint is also known as the *feature compatibility constraint*.

In addition to the above constraints, depending on the scene geometry and the vergence angle of the stereo camera system, the distance between corresponding points in the images and the orientation of corresponding lines in the images may be used as constraints to filter out possible wrong correspondences.

10.4.2 Correspondence algorithms

Stereo-correspondence algorithms often take into consideration one or more of the constraints described above to reduce ambiguities in correspondences and arrive at unique solutions. Stereo-correspondence algorithms match image features or image intensities. Feature-based algorithms use edges, corners, lines, and curves, and intensity-based methods use intensities or gradients in small neighborhoods to find the correspondences. Feature-based methods are more robust since they are less sensitive to noise and intensity difference between images. However, they produce sparse disparities and require an interpolation method that can estimate the missing disparities. Intensity-based methods, on the other hand, are not as robust, since they use raw image intensities, but they can produce a dense disparity map, minimizing the role of the interpolation method. One may actually use both image features and image intensities to find the correspondences [16].

Edge contours representing silhouettes of smooth objects may produce inaccurate depth values. This is because silhouettes of smooth objects in stereo images do not represent the same object points in 3-D. In algorithms that use image intensities, the windows containing different objects can also produce inaccurate matches. In Goshtasby [144], the images were first segmented and a window falling on a region boundary was partitioned into two: the part belonging to the foreground object and the part belonging to the background object. Matching was then performed independently between the foreground and background parts of the window. In a method developed by Agrawal and Davis [2], a window was considered to have two possible disparities, one for the object in the foreground and one for the object in the background. A global optimization was then used to compute the compatibility of disparity values between neighboring pixels, creating a final disparity map that preserved depth discontinuities.

In the following, a number of well-known stereo-correspondence algorithms are reviewed.

10.4.2.1 Cooperative stereo A class of stereo-matching algorithms is modeled after human stereopsis and based on Julesz's proposal that stereo matching is a cooperative process [89, 216, 262, 263]. In a version of the algorithm implemented by Marr and Poggio [264], matching is performed at multi-resolution, propagating information from low to high resolution. In particular, four different resolutions are used and, at each resolution, zero-crossing edges are grouped into twelve different orientations, and edges with the same sign and orientation are matched. Since, at a lower resolution, fewer edges are obtained in an image, matching can be performed more reliably. As the resolution is increased, more edges are obtained but, since the

approximate match positions are known from matching edges at one level lower resolution, the area of search is restricted, producing fewer false matches. Additionally, to reduce the false-match probability, continuity and uniqueness constraints are used. The correspondence algorithm of Marr and Poggio follows the steps below.

1. Filter each image with oriented bar masks of four different sizes and twelve different orientations.
2. Find zero-crossing edges of each filtered image.
3. Match edges of the same sign, same resolution, and same orientation in the stereo images, starting from the lowest resolution edges and limiting the search space to the width of the mask used to obtain the edges.
4. Use the correspondences obtained at one resolution to estimate the correspondences at one level higher resolution. Then, refine this estimate using the lower-resolution edges in the images.
5. Repeat step 4 until correspondence is established between highest-resolution edges in the images.

An extension of this algorithm has been proposed by Mayhew and Frisby [270], which considers use of all resolutions simultaneously in the correspondence process. Experimental results suggest that stereo correspondence improves when information about all resolutions is used simultaneously rather than in sequence. A further improvement of the cooperative algorithm is proposed by Zitnick and Kanade [431], specifically looking for occlusions and using that information to guide the correspondence process.

10.4.2.2 Prediction and verification In the cooperative algorithm outlined above, the matching result obtained at a resolution was used to guide matching at one level higher resolution. In contrast, in the prediction and verification algorithm [13, 12, 279], information from one match is used to guide matching in its immediate neighborhood. The prediction and verification process is gradually propagated from one neighborhood to the next until all image features are matched.

A version of the prediction and verification algorithm implemented by Ayache and Faugeras [12] follows the steps below.

1. Detect line segments in the images and remove the very short segments.
2. For each line segment in the left image, choose a window in the right image centered at that line. Then,
   (a) find the line in the right image that corresponds to the line in the left image within the selected window using the coordinates of the midpoint, the orientation, the contrast, and the average gradient at the lines;
   (b) in case of a conflict, choose the correspondence that minimizes a weighted sum of the feature differences. If the weighted sum of the feature differences of the best-match is above a threshold value, discard the correspondence and consider the line in the left image having no correspondence in the right image.

The above algorithm may also be used to match points, regions, and templates by appropriately replacing the similarity measure. Matching can be carried out at multi-resolution to achieve a higher speed. First correspondence is established between features at the lowest resolution where fewest features are available. Matching results obtained at one resolution are then used to predict the match positions at one level higher resolution. The match positions are then refined using information in the higher resolution images.

10.4.2.3 Dynamic programming This algorithm finds a minimal cost path connecting pixels on corresponding epipolar lines [291]. The process is depicted in Fig. 10.8. The horizontal axis shows pixels along (a segment of) an epipolar line in the left image and the vertical axis shows pixels along (a segment of) the corresponding epipolar line in the right image. Suppose each entry obtained in the matrix formed by pixels along the epipolar lines in the images represents a node in a graph that is connected to the nodes to its right and above. If no occlusions exist along the segment, the correspondence problem becomes that of finding the minimal cost path that connects a node on the left border of the matrix to a node on the right border. An example path is shown in the Fig. 10.8. Note that the path is always monotonically increasing due to the ordering constraint in stereo. If the absolute intensity difference between corresponding pixels is used as the correspondence cost, then

$$E(x_L, y, d_{x_L}) = |I_L(x_L, y) - I_R(x_L + d_{x_L}, y)|\,, \tag{10.41}$$

and the total cost for a particular path would be

$$E_y = \sum_{x_L=1}^{w} E(x_L, y, d_{x_L}), \tag{10.42}$$

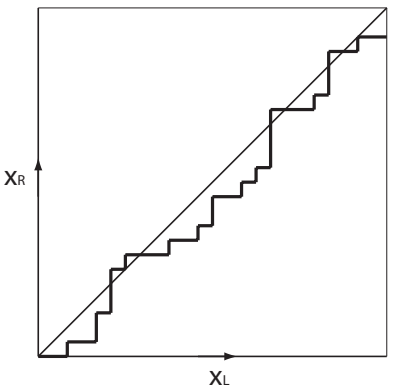


Fig. 10.8 Stereo correspondence by dynamic programming. The diagonal line shows the zero-disparity path, and the path shown by the contour represents a typical correspondence.

where I_L and I_R are intensities of corresponding pixels in the left and right images, respectively, d_{x_L} is the disparity of the pixel at x_L in the left image and its corresponding pixel in the right image, w is the number of pixels along the left epipolar line (segment) where matching is performed, $E(x_L, y, d_{x_L})$ is the cost of pixel (x_L, y) in the left image corresponding to pixel $(x_L + d_{x_L}, y)$ in the right image, and E_y is the total cost of matching (corresponding segments of) the yth epipolar lines in the images. If the images are rectified so that corresponding epipolar lines represent the same scanline, E_y will be the cost of matching (corresponding segments of) the yth scanline in the images. Dynamic programming finds the path that has the smallest cost among all possible paths. This algorithm is of iterative nature and is often time consuming. Forstmann et al [124] have proposed a coarse-to-fine matching approach to speed up the process.

Occlusion disrupts the correspondence process in dynamic programming. Birchfield and Tomasi [33] have made a special effort to detect occlusions and use them in the correspondence process. A cost function is used to measure the likelihood that a particular correspondence is not correct. The cost function is designed to prefer piecewise constant disparities, therefore producing constant disparities for foreground and background objects and enabling separation of objects at different depths. In an algorithm designed by Aguado and Montiel [3], the scene is modeled by a piecewise linear surface, considering the disparity in a neighborhood linear. Epipolar lines are recursively subdivided by explicitly looking for breaks in disparity. As long as disparity within a segment is linear, the segment is kept together. A search is carried out to explicitly look for breaks in the disparity, dividing a segment into two at the break. This method produces longer segments compared to the method of Birchfield and Tomasi, where epipolar lines are divided into segments with constant disparities. Bobick and Intille [35] have used intensity edges to detect occlusion boundaries.

10.4.2.4 Template-matching methods Template matching methods use image intensities, or gradients, in small windows to establish correspondence between points in the images as outlined in section 4.5. Due to occlusion and local geometric difference between images, larger windows may not necessarily produce more reliable matches. Okutomi and Kanade [293] suggested setting window size proportional to gradient at the center of the window. Larger gradient magnitudes often imply larger local geometric differences between images, thus requiring smaller windows to find the correspondences. Smaller gradient magnitudes imply a more homogeneous area, requiring larger windows in matching. The sum of absolute intensity differences and the cross-correlation coefficient are typical similarity measures used in the matching of stereo images. Multi-scale matching has been suggested as a means of achieving more reliable matches. Initially, the size of the images is reduced in order to reduce the local geometric differences between them and also to reduce the size of homogeneous areas in the images. Matching is performed at the lowest size first and the results are used to guide the matching of images at higher sizes. As image size is increased, the local geometric difference between images increases but, from matching information at lower sizes, smaller search areas are selected to achieve more reliable matches. Lane *et al.* [232] proposed the concept of stretch correlation, where

a window is stretched in a manner to maximize correlation. Information about image edges and disparity gradient is used to stretch the windows.

Cross-checking is a simple process that can be used by both feature-matching and intensity-matching algorithms to detect and eliminate the incorrect correspondences. Cross-checking involves switching the left and right images, repeating the process, and taking only those correspondences that are obtained in both forward and reverse steps. Although there is no guarantee that a correct match is obtained when both forward and reverse steps produce the same result, we can be sure that, if the forward and reverse steps do not produce the same result, the forward correspondence, the backward correspondence, or both are wrong.

10.4.2.5 Trinocular stereo The constraints used in stereo matching can be made more powerful if, instead of a binocular setup, a trinocular setup [294] is used. In Fig. 10.9, the geometry of a trinocular setup is depicted. If $\mathbf{e}_{11}$ and $\mathbf{e}_{21}$ represent corresponding epipolar lines in images from cameras 1 and 2 then, to find the correspondence to edge point $\mathbf{a}$, it is sufficient to search along $\mathbf{e}_{21}$ for an edge. Suppose two edge points $\mathbf{b}$ and $\mathbf{c}$ exist along $\mathbf{e}_{21}$ and we do not know which one to choose. Now, in images from cameras 2 and 3, suppose the epipolar lines passing through $\mathbf{b}$ and $\mathbf{c}$ in the image from camera 2 are $\mathbf{e}_{22}$ and $\mathbf{e}_{23}$ and the corresponding epipolar lines in the image from camera 3 are $\mathbf{e}_{32}$ and $\mathbf{e}_{33}$. If we consider the stereo images from cameras 1 and 3, if $\mathbf{e}_{11}$ and $\mathbf{e}_{31}$ are corresponding epipolar lines and there is only one edge point along $\mathbf{e}_{31}$ that lies at the intersection of $\mathbf{e}_{31}$ and $\mathbf{e}_{32}$, then it can be concluded that points $\mathbf{a}$ and $\mathbf{b}$ in the images from cameras 1 and 2 correspond to each other because both $\mathbf{a}$ and $\mathbf{b}$ correspond to the same point $\mathbf{d}$ in image 3. The constraint provided by the third camera has, in effect, been able to remove the ambiguity of not knowing whether $\mathbf{b}$ or $\mathbf{c}$ corresponds to $\mathbf{a}$. The reliability of the correspondence algorithms can be increased if images from more than two cameras are used.

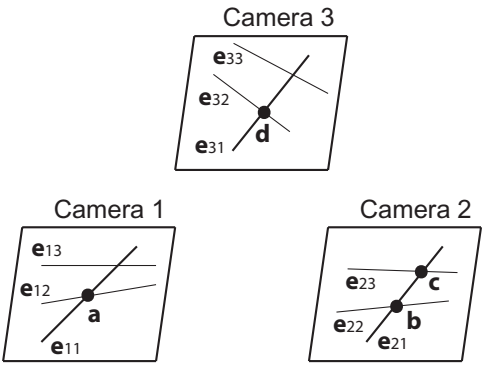


Fig. 10.9 The trinocular stereo geometry.

10.4.2.6 Voxel coloring Kutulakos and Seitz [231] and Seitz and Dyer [352] have taken a completely different approach from the approaches described above. They quantize the 3-D space into volume elements, or voxels, and scan the voxels from back to front to find corresponding points in the images. The method requires a camera set up that is accurately calibrated. As each voxel is visited, pixels in the stereo images corresponding to it are located and their intensities or colors are compared. The voxel position and the intensity or color similarity of corresponding pixels is associated with the pixel in the left images. As the voxels are visited from back to front, the information associated with a pixel is revised if the newly obtained correspondence has a similarity measure that is higher than that previously obtained. The process is repeated until all voxels are processed. As the correspondences are revised, the intensity or color of the pixel in the left image is associated with the voxel just processed, in a way, painting the voxel with the image intensity or color. Once all the voxels are processed, those pixels in the left image with associating similarity measures that are below a threshold value are considered occluded points and removed from the set of correspondences. This algorithm does not have to satisfy any constraints and, therefore, it can reconstruct very detailed scenes with sharp depth discontinuities. However, for the same reason, noise can greatly affect the matching outcome. Images from highly textured scenes produce more reliable correspondences than images from textureless scenes.

10.4.2.7 Examples Stereo correspondence produces a disparity map from which the geometry of the underlying scene can be guessed. Examples of disparity maps generated by three stereo correspondence algorithms are shown in Fig. 10.10. The results are obtained using the cooperative algorithm of Zitnick and Kanade [431], the dynamic programming with specific occlusion detection of Bobick and Intille [35], and a template matching algorithm, with sum of squared intensity differences as the similarity measure, as implemented by Scharstein and Szeliski [342]. Figures 10.10a and 10.10b show a pair of stereo images of a "head and lamp" scene. The disparity ground truth is shown in Fig. 10.10c. Disparities computed by the cooperative, the dynamic programming, and the template matching algorithms are depicted in Figs 10.10d–f. All disparities have been multiplied by 16 for enhanced viewing. The percentage of bad disparities (incorrect correspondences) by the cooperative, dynamic programming, and template matching algorithms are 3.49, 4.12, and 5.23, respectively. Only non-occluded pixels are considered and image regions within 10 pixels of image borders are not used in the computation of the percentages.

A second example is provided in Fig. 10.11. The scene is made up of a few planar segments with a "sawtooth" geometry. The disparity ground truth as well as the disparities computed by the three algorithms are depicted in Fig. 10.11. The percentage of incorrect correspondences found by the cooperative, dynamic programming, and template matching algorithms are 2.03, 4.84, and 2.21, respectively. The parameters of an algorithm were fixed when testing on both head-and-lamp and sawtooth images. Both cooperative and template-matching algorithms have produced better results for the sawtooth scene than the head-and-lamp scene. This can be attributed to the fact that the sawtooth scene contains large planar segments and the continuity constraint

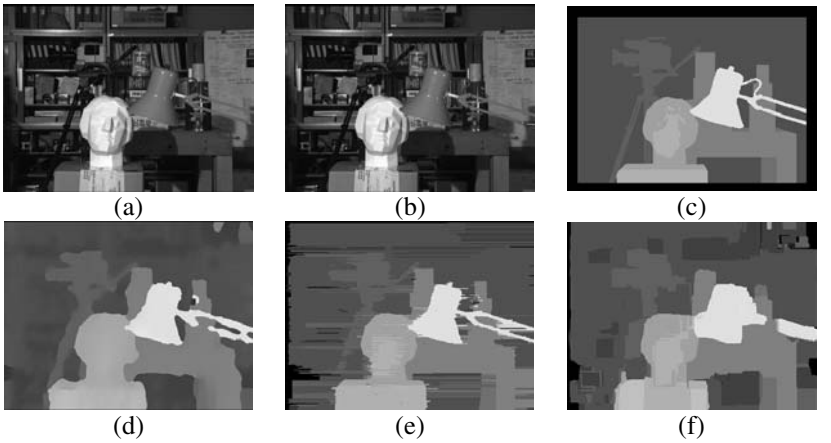


Fig. 10.10 (a), (b) The head-and-lamp stereo image pair. (c) The disparity ground truth. (d)–(f) Disparity maps obtained by an implementation of the cooperative algorithm, dynamic programming, and template matching, respectively.

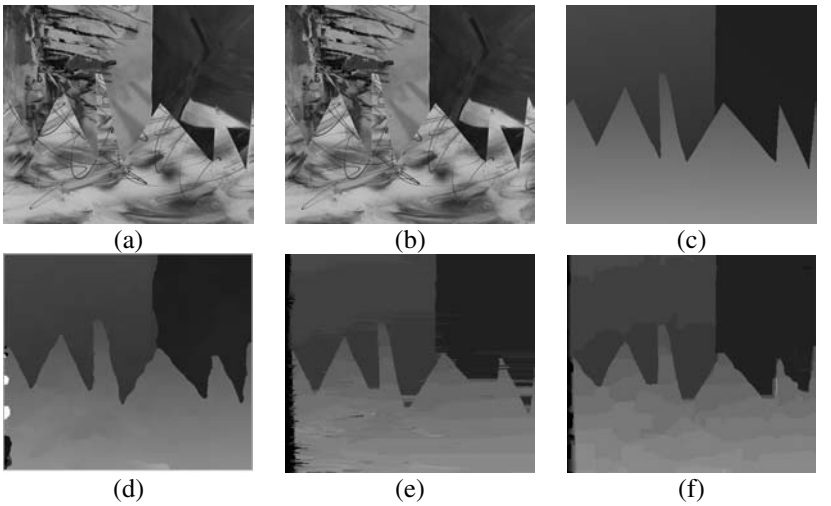


Fig. 10.11 (a), (b) The sawtooth stereo image pair. (c) Disparity ground truth. (d)–(f) Disparity maps obtained by the cooperative, dynamic programming, and template matching algorithms, respectively.

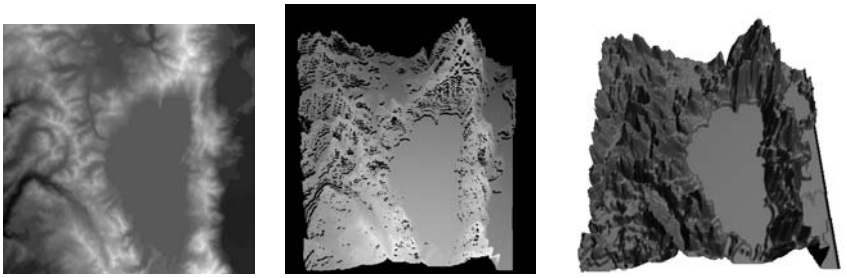
used in these algorithms has been able to recover the scene geometry effectively. The dynamic programming algorithm has performed better on the head-and-lamp scene because it is more detailed than the sawtooth scene. The images and results reported in this section are courtesy of Scharstein and Szeliski [342].

These results are from state-of-the-art methods. As can be observed, there is still room for improvement. It is very difficult to partition these disparity images into objects that can be recognized. Stereo correspondence will produce cleaner and more useful results when coupled with object models.

10.5 INTERPOLATION

Knowing the coordinates of corresponding points in the images, coordinates of scene points can be determined if the camera parameters $m_1 \dots m_{11}$ and $n_1 \dots n_{11}$ are known. Scene points computed from stereo correspondence create an irregularly spaced set of 3-D points. Even when no occlusions exist and correspondence for all points in the left image is determined, when the points are mapped to 3-D gaps will appear in areas where direction of view and surface normal make relatively large angles. Such gaps can be filled by interpolation. An example of scene reconstruction from a computed depth map is shown in Fig. 10.12. Figure 10.12a shows a depth map of a terrain scene. Brighter pixels show scene points that are closer to the camera baseline. When the same information is displayed in 3-D, as shown in Fig. 10.12b, gaps will appear in areas where discontinuities in depth exist. To reconstruct the scene, it is required to create a continuous surface that interpolates the scene points, as shown in Fig. 10.12c.

Suppose a set of points in the image domain $\{(x_i, y_i) : i = 1, \dots, N\}$, with associated depth values $\{f_i : i = 1, \dots, N\}$, is available. Let's call (x_i, y_i, f_i) the ith data point. The problem to be solved is to find a surface that approximates or interpolates the N data points. Since N can be very large, it is desired to find the surface without solving a system of equations. Depending on the contents of stereo



(a) (b) (c)

Fig. 10.12 (a) The top view of a depth map showing distances of scene points to the camera baseline; brighter pixels show scene points that are closer to the baseline. (b) The same depth map shown from the side. (c) Reconstruction of the scene by surface fitting.

images, the given points may be very dense in some areas while sparsely spaced in other areas. Therefore, the surface to be estimated should automatically adjust itself to the density of points. We also would like the surface to have a free smoothness parameter which can be increased when the given data are noisy in order to smooth the noise and decreased when there is a need to reproduce details in the depth map.

Consider the weighted linear function described in section 5.7, where $f_i(x, y)$ is a plane fitting the ith data point and $n-1$ data points closest to it. Then, a surface point will be determined from a weighted sum of the planes (polynomials of degree 1), each approximating the data in a small neighborhood. Instead of planes, polynomials of degree two or higher may be used. Polynomials of degree higher than two, however, may produce fluctuations away from data points. Polynomials of degrees 1 and 2 are sufficient to create a surface that approximates irregularly spaced points. Because monotonically decreasing weights are used, the effect of a local polynomial vanishes with a required accuracy beyond a certain distance. Moreover, the widths of the weight functions can be adjusted to the local density of points, making it possible to preserve local details in the reconstructed surface. Note also that, since a system of equations need not be solved, a solution is guaranteed even when a very large set of points with a highly varying density is given.

Scene recovery from scattered points is generally an underconstrained problem. Given a set of points in 3-D, an infinite number of surfaces can be found that can fit the points. To obtain a unique surface, smoothness constraints that minimize quadratic variation [169] or curvature [380] have been used. Analytic surfaces formulated by these methods are very difficult to find. Therefore, the methods often iteratively find surface values at discrete image coordinates. Iterative methods are easy to implement but are very time consuming. To speed up the process, various multiresolution and hierarchical methods have been proposed [303, 375, 423].

Radial basis functions [105, 183, 314] find surfaces in closed form. If the number of points is large, surface recovery by these methods requires the solution of a large system of equations, which may not always be possible. Since these surfaces do not satisfy the convex-hull property, when the density of points varies across the image domain, overshoots and fluctuations may appear away from the points. To avoid the solution of a large system of equations when a large number of correspondences is available, iterative methods that estimate a surface at discrete points may be used. Using thin-plate splines, Turk and O'Brien [390] developed an implicit surface interpolation based on signed distances that could find a discrete surface interpolating a large number of points. In a method developed by DeRose, Hoppe, and others [88, 191], first, the topological type of a point set is determined and a crude estimation of its geometry is made. Then, the quality of the fit is improved through an optimization process by revising the triangles locally to achieve desired shapes, for instance, producing sharp edges. Through a subdivision scheme, they later developed a method that could produce a smooth surface fitting a triangulated mesh [191]. In Lee *et al.* [236], a multi-resolution approach is described where, at the coarsest resolution, predefined templates are used to obtain an initial piecewise linear approximation to the data and then, through a subdivision scheme, the approximation is refined.

In the interpolation scheme described by Ohtake *et al.* [292], a quadratic surface is fitted to each local neighborhood in such a way that the functional value becomes 0 at each data point. A weighted sum of such functions is computed and the zero-crossings of the sum are determined in 3-D to recover the surface. Compactly supported weights, such as B-spline bases, are used and the basis functions are normalized to have a sum of one everywhere in the approximation domain. When the density of the data varies, holes may appear in the constructed surface model. The holes can be covered by a hierarchical approach where, through an octree subdivision, larger neighborhoods are used higher in the hierarchy to estimate the surface everywhere. Based on the error obtained, smaller neighborhoods are used at lower hierarchies to create more detailed surfaces where denser data are available.

A class of analytic methods that work with large data sets and provide the convex-hull property is based on the weighted mean idea. These methods can be categorized into global [143] and local [126, 267]. In global methods, a weighted sum of the points is used, with the weights normalized such that their sum is equal to one everywhere in the image domain. In the local methods, polynomials are fitted to local points and the weights are taken in such a way that they are nonzero only in local neighborhoods of the points. The global methods are easy to implement but they are time consuming because a surface point is obtained from all data points. The local methods are more difficult to implement but they are faster because a small number of points is needed to find a surface point. The local methods produce holes where large gaps exist between the data points.

Approximation of irregularly spaced data by parametric surfaces is described by Goshtasby [152, 157]. A globally defined parametric surface is used to approximate an entire scene geometry. A smoothness parameter is used to control the level of detail of the generated surface. A larger smoothness measure is used to produce an overall smoother surface by smoothing out noisy details, while a smaller smoothness measure is used to reproduce more details.

10.6 SUMMARY

To recover the geometry of a scene from a pair of stereo images, it is required to:

1. calibrate the camera system and determine the relation between scene points and corresponding points in the images. This involves placing a calibration box of the kind shown in Fig. 10.3 in front of the camera system, obtaining a pair of images, and using the coordinates of corresponding points in the images and on the box to determine parameters $m_1 \ldots m_{11}$ and $n_1 \ldots n_{11}$;
2. determine the parameters of a projective transformation as defined by (10.35) and (10.36) for each camera so that, after the transformation (rectification), corresponding epipolar lines map to the same scanline in the images;
3. determine the correspondence between calibrated and rectified images through feature matching or template matching. Correspondences that are believed to be unreliable should be discarded, keeping only highly reliable correspondences;

4. determine the 3-D coordinates of scene points by substituting the coordinates of corresponding points in the image into (10.32) – (10.34) and solving for X, Y, and Z;
5. fit a surface to recovered 3-D scene points and estimate the missing scene points.

By far the most difficult step in stereo depth perception is the correspondence step. Considerable progress has been made in this area and the methods have gradually become more intelligent and reliable. An algorithm that works as well as the human visual system may never be realized. However, algorithms that work relatively well on a well-defined class of images may not be out of reach. Use of domain knowledge and the right constraints could lead to very robust algorithms that can make robots see and navigate in a factory environment, or a desert scene, where the objects being viewed are known ahead of time.

10.7 BIBLIOGRAPHICAL REMARKS

Work on stereo depth perception started when trying to understand depth perception in the human visual system. Julesz and Miller [217], Marr and Poggio [264], and Mayhew and Frisby [270] used the multi-channel model of the human visual system to find depth from stereo correspondence. The earliest computer algorithms for stereo depth perception are reported by Levine *et al.* [241], Mori *et al.* [279], Thompson [382], Nevatia [286], and Gennery [139].

Although stereo often implies images from two cameras that are horizontally displaced, if the scene is static, the same information can be captured by a single camera while horizontally shifting it. This is known as motion stereo. Depth perception by motion stereo has been studied by Bolles *et al.* [36], Jiang and Bunke [210], Williams [413], and Yip [426]. Most studies in stereo depth perception have used gray scale images. Such algorithms can be easily extended to color images. The specific role of color in depth perception has been the subject of a number of studies [47, 213, 214, 229, 282, 420].

Excellent reviews of stereo depth perception algorithms are provided by Barnard and Fischler [22], Levine and Shefner [242], Poggio and Poggio [313], and Scharstein and Szeliski [342]. Although almost all stereo correspondence algorithms that use image features use edge points, a method described by Ayache and Faverjan [14] finds correspondence between lines and recovers scene geometry from such correspondences.

Obvious applications of stereo depth perception are in autonomous vehicle navigation [139, 277, 278, 422] and in photogrammetry for construction of digital elevation maps [235, 298]. When equipped with a laser line generator, the problem of stereo correspondence becomes considerably simpler since correspondences need to be established between one laser profile (or a small number of laser profiles) in an image pair. Stereo depth perception when assisted by special lighting is known as active stereo and is the basis for many range scanners [43, 219, 229, 259, 425].

In spite of considerable progress in stereo depth perception, a robust and universal algorithm that works on all types of images does not exist. Each algorithm is effective

in determining the correspondence between a particular class of images. Jenkin and Kolers [209] point out some of the problems with existing algorithms. First, a typical algorithm uses discrete images and consequently produces discrete scenes. In reality, this is not the case because the scene is continuous. Second, correspondence is assumed to be a two-dimensional problem and is handled without regard to the underlying 3-D structure of a scene. In depth perception in the human visual system, 3-D interpretation influences the correspondence process. Algorithms that come close to incorporating 3-D information in the correspondence process use model priors, for instance assuming the scene is locally planar [134]; use a segmentation of at least one of the images to guess the occluded boundaries [144]; know that the scene is made up of smooth surfaces that can be represented by spline patches [244]; have some information about the geometry of the scene, for example knowing that the scene is polyhedral [378]; or know that the geometric difference between the images can be modeled by splines [377].

Glossary

accuracy The difference between the true value and an estimated value.

affine transformation The transformation obtained by parallel projection. Under affine transformation, parallel lines remain parallel.

anatomic landmark A landmark that has anatomic meaning, such as the tip of a person's nose or the point where a blood vessel branches.

auto-correlation Correlation of two overlapping image windows.

baseline The line connecting the lens centers of two cameras.

baseline length The distance between the lens centers of two cameras.

bilinear interpolation A resampling method that sets the intensity at a point equal to the weighted sum of intensities of four pixels closest to it according to formula (6.3).

control point A point landmark centered at a locally unique image neighborhood, such as the centroid of a region or the centroid of a locally unique circular window.

convolution A linear shift-invariant operation applied to an image. Given image $f(x, y)$ and symmetric operator $h(x, y)$, the convolved image is obtained from $g(x, y) = \sum_{x'} \sum_{y'} h(x - x', y - y') f(x', y')$.

corner An image point appearing as the intersection of two orthogonal edges.

cross-correlation coefficient The correlation coefficient of windows in two images. Correlation coefficient is a measure quantifying the dependence between two sets of numbers (such as intensities in two images).

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

cubic convolution resampling A resampling method that sets the intensity at a point to the weighted sum of the intensities of the pixels within a 4×4 window centered at the point according to formula (6.8).

deblurring An image operation used to sharpen an image. Deblurring has the inverse effect of the smoothing operation.

discrete Fourier transform Given uniformly spaced samples $\{f_i : i = 0, \ldots, n-1\}$ from a periodic signal, the discrete Fourier transform of the samples is defined by

$$F_k = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} f_i e^{-j2\pi ik/n} \qquad k = 0, \ldots, n,$$

where $j = \sqrt{-1}$. Note that discrete Fourier transform coefficients are complex numbers.

disparity The horizontal distance between corresponding points in stereo images.

distance transform image An image where the value at an object point is 0 and the value at a background point is proportional to the distance of that point to the object point closest to it.

entropy A measure used to quantify information content in an image or a subimage.

epipolar constraint Corresponding points in stereo images lie on corresponding epipolar lines.

epipolar line The intersection of an epipolar plane and an image plane.

epipolar plane The plane obtained from the lens centers of a stereo camera system and an object point.

epipole The point where the baseline intersects an image plane.

false-negative probability The probability that saying something is not present but it, in fact, being present.

false-positive probability The probability that saying something is present but it, in fact, being absent.

fast Fourier transform A fast computational method for determining the discrete Fourier transform of a discrete signal.

fiducial marker An external marker which, when placed in the scene, is clearly identifiable, in the images of the scene, either visually or by image analysis algorithms.

filtering A convolution operation.

Fourier transform A mathematical method for expressing a signal in terms of its spatial frequencies. The Fourier transform of periodic signal $f(x)$ is defined by

$$F(u) = \int_{-\infty}^{+\infty} f(x) e^{-j2\pi ux} dx = F_r(u) + jF_i(u).$$

A coefficient corresponding to frequency u has a real part $F_r(u)$ and an imaginary part $F_i(u)$. $j = \sqrt{-1}$.

Gaussian filtering A filtering operation where the filter kernel is a Gaussian. The intensity at a point in the output is determined from the weighted sum of intensities in the neighborhood of the point in the input, with the weights being the Gaussian values.

georectification Identifying Ground Control Points (GCP) in an image and resampling the image so that corresponding GCPs in the image and in a standard data set align.

graph isomorphism Finding a one-to-one mapping between vertices in two graphs such that, if there is an edge between two vertices in one graph, there will be an edge between the corresponding vertices in the other graph.

Hough transform A methodology for determining the unknown parameters of an analytic function through a voting process. The parameters receiving the most votes from the available image features, such as points, are selected as the parameters of the function.

image feature A quantifiable image property.

image fusion The process of combining information in two or more images of a scene.

image segmentation The process of partitioning an image into meaningful parts.

impulse noise Random noise that affects a small percentage of pixels in an image.

intensity edge A sharp change in image intensities. An edge has magnitude and direction. Edge direction at a pixel is the direction of maximum intensity change and edge magnitude is the magnitude of intensity change in that direction.

invariant moments A class of image moments that are invariant under image rotation, scale, and translation.

inverse filtering Inverse filtering has the inverse effect of filtering. For example, filtering an image with a Gaussian and then inverse filtering the result with a Gaussian should produce the original image.

landmark A visible image feature, such as a corner.

least-squares fitting A mathematical method for approximating (fitting) a function to a set of image features in such a way that the sum of squared distances between the given features and the function is minimum.

linear transformation The transformation defined by $X = ax + by + c$, $Y = dx + ey + f$.

log-polar mapping A polar quantization where radial steps are taken logarithmically rather than uniformly.

mean filtering The averaging operation. The intensity at a point in output is computed from the average of intensities in a small neighborhood centered at the point in input.

median filtering A nonlinear operation where the intensity at a point in output is set to the median of intensities in a small neighborhood centered at the point in input.

nearest-neighbor resampling Setting the intensity at a point to the intensity of the pixel closest to it.

outlier An image feature with considerably different properties than other features under consideration.

performance evaluation Evaluating the accuracy, reliability, robustness, and computational complexity of an algorithm or a system.

piercing point The point where the optical axis of a camera intersects its image plane.

projective transformation The transformation obtained by the perspective projection. Under this transformation, straight lines remain straight.

rectification Standardization of an image. In remote sensing, it is the process of standardizing an image with respect to a map. In stereo depth perception, it is the process of standardizing images obtained by cameras with converging optical axes to images with parallel optical axes.

reference image One of the two images to be registered. The reference image is the image that is kept unchanged and used as the reference.

relaxation labeling An algorithm that assigns labels to objects initially based on available information and then iteratively revises the labels to satisfy a set of constraints until labels consistent with world knowledge are obtained.

reliability The rate of success of an algorithm or system.

resampling Mapping one image geometry to another. This involves fitting a function to the image intensities and evaluating the function at image locations determined by a transformation function.

rigid transformation The transformation obtained by a combination of translations and rotations.

robustness The degree of stability of an algorithm under variations in one or more of its input parameters. An algorithm that consistently produces high accuracy or reliability in what it does is considered robust.

salt-and-pepper noise Impulse noise.

sensed image The second image in a set of two to be registered. This image is resampled through the registration process to overlay the reference image.

shape matrix Quantization of a shape into a uniform grid of measurements that is independent of the position, orientation, and scale of the shape.

similarity transformation The transformation defining the orthographic projection. Under this transformation, angles are preserved.

smoothing A filtering operation that reduces the magnitude of high spatial frequency coefficients in an image. An example of image smoothing is Gaussian filtering.

stable corner A corner that persists over a wide range of image resolutions.

stereo images Two images of a scene, most often obtained from about the same distance but from slightly different views of the scene.

subgraph isomorphism Finding whether there is a subgraph of one graph that is isomorphic to another graph (see also graph isomorphism).

template Depending on the context, this means a pattern, the contents of a window, or a neighborhood in the reference image.

template matching Taking a template in the reference image and finding its location in the sensed image.

transformation function A function relating the geometry of the sensed image to the geometry of the reference image. Given the coordinates of a point in the reference image, a transformation function determines the coordinates of the corresponding point in the sensed image.

true-positive probability The probability that saying something is present and it truly being present.

unit transformation No transformation; that is $X = x$ and $Y = y$.

vergence angle The angle between the optical axes of the left and right cameras in a stereo camera system.

white noise Random noise that has a mean of zero.

zero-mean noise White noise.

References

1. Aggarwal, M. and N. Ahuja, Split aperture imaging for high dynamic range, *Int'l J. Computer Vision*, **58**(1):7–17 (2004).

2. Agrawal, M. and L. S. Davis, Window-based, discontinuity preserving stereo, *Proc. Conf. Pattern Recognition and Image Processing*, Washington, D.C., I67–I73 (2004).

3. Aguado, A. S. and E. Montiel, Progressive linear search for stereo matching and its application to interframe interpolation, *Computer Vision and Image Understanding*, **81**:46–71 (2001).

4. Alparone, L., L. Facheris, S. Baronti, A. Garzelli, and F. Nencini, Fusion of multispectral and SAR images by intensity modulation, *Proc. 7th Int'l Conf. Information Fusion (Fusion 2004)*, Stockholm, Sweden, June 28 – July 1, 637–643 (2004).

5. Alt, F. L., Digital pattern recognition by moments, *Optical Character Recognition*, G. L. Fischer, *et al.* (Eds.), Spartan Books Inc., 240–258 (1962).

6. Alvarez, L. and F. Morales, Affine morphological multiscale analysis of corners and multiple junctions, *Int'l J. Computer Vision*, **2**(25):95–107 (1997).

7. Andrews, H. C. and C. L. Patterson, Singular value decomposition image coding, *IEEE Trans. Communications*, 425–432 (1976).

8. Anuta, P. E., Registration of multispectral video imagery, *Society Photo-Optical Instrum. Eng. J.*, **7**:168–175 (1969).

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

9. Anuta, P. E., Spatial registration of multispectral and multitemporal digital imagery using fast Fourier transform techniques, *IEEE Trans. Geoscience Electronics*, **8**(4):353–368 (1970).
10. Arun, K. S., T. S. Huang, and S. D. Blostein, Least-squares fitting of two 3-D point sets, *IEEE Trans. Pattern Analysis and Machine Intelligence,* **9**(5):698–700 (1987).
11. Aubert, G. and L. Blanc-Feraud, Some remarks on the equivalence between 2D and 3D classical snakes and geodesic active contours, *Int'l J. Computer Vision*, **34**(1):19–28 (1999).
12. Ayache, N. and O. D. Faugeras, HYPER: A new approach for the recognition and positioning of two-dimensional objects, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **8**(1):44–54 (1986).
13. Ayache, N. and B. Faverjon, Fast stereo matching of edge segments using prediction and verification of hypothesis, *Proc. Computer Vision and Pattern Recognition*, 662–664 (1985).
14. Ayache, N. and B. Faverjon, Efficient registration of stereo images by matching graph descriptions of edge segments, *Int'l J. Computer Vision*, 107–131 (1987).
15. Baillard, C., P. Hellier, and C. Barillot, Segmentation of brain 3D MR images using level sets and dense registration, *Medical Image Analysis*, **5**:185–194 (2001).
16. Baker, H. H. and T. O. Binford, Depth from edge and intensity based stereo, *Int'l J. Conf. Artificial Intelligence*, 631–636 (1981).
17. Ballard, D. H., Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition*, **13**(2):111–122 (1981).
18. Baltsavias, E. and Ch. Käser, DTM and orthoimage generation — a thorough analysis and comparison of four digital photogrammetric systems, *Int'l Arch. Photogramm. Remote Sensing*, **32**(4):42–51 (1998).
19. Bao, P. and D. Xu, Complex wavelet-based image mosaics using edge-preserving visual perception modeling, *Computers & Graphics*, **23**:309–321 (1999).
20. Barber, D. C., Automatic alignment of radionuclide images, *Physics in Medicine and Biology*, **27**:387–396 (1982).
21. Bardinet, E., L. D. Cohen, and N. Ayache, Superquadrics and free-form deformations: A global model to fit and track 3D medical data, *Proc. 1st Int'l Conf. Computer Vision, Virtual Reality, and Robotics in Medicine*, 319–326 (1995).
22. Barnard, S. T. and M. A. Fischler, Computational stereo, *Computing Surveys*, **14**(4):553–572 (1992).
23. Barnea, D. I. and H. F. Silverman, A class of algorithms for fast digital image registration, *IEEE Trans. Computers*, **21**(2):179–186 (1972).
24. Barnhill, R. E., Representation and approximation of surfaces, *Mathematical Software III*, J. Rice (Ed.), Academic Press, 69–120 (1977).
25. Barrow, D., J. M. Tennebaum, R. C. Bolles, and H. C. Wolf, Parametric correspondence and chamfer matching: Two new techniques for image matching, *Int'l J. Conf. Artificial Intelligence*, 659–663 (1977).

26. Beaudet, P. R., Rotationally invariant image operators, *Proc. Int'l Conf. Pattern Recognition*, 579–583 (1978).
27. Beil, W., K. Rohr, and H. S. Steihl, Investigation of approaches for the localization of anatomical landmarks in 3-D medical images, *Computer Assisted Radiology and Surgery*, H. U. Lemke, *et al.* (Eds.), Elsevier Science, 265–270 (1997).
28. Bergholm, F., Edge focusing, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **19**:726–741 (1987).
29. Bernstein, R., Digital image processing of earth observation sensor data, *IBM J. Research and Development,* 40–67 (1976).
30. Bertram, M., J. C. Barnes, B. Hamann, K. I. Joy, H. Pottmann, and D. Wushour, Piecewise optimal triangulation for the approximation of scattered data in the plane, *Computer Aided Geometric Design,* **17**:767–787 (2000).
31. Besl, P. J. and N. D. McKay, A method for registration of 3-D shapes, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **14**(2):239–256 (1992).
32. Bezdek, J., L. Hall, and L. Clarke, Review of MR image segmentation techniques using pattern recognition, *Medical Physics*, **20**:1033–1048 (1993).
33. Birchfield, S. and C. Tomasi, Depth discontinuities by pixel-to-pixel stereo, *Int'l J. Computer Vision*, **35**(3):269–293 (1999).
34. Bloomenthal, J., *Introduction to Implicit Surfaces*, Morgan Kaufmann, San Francisco, CA (1997).
35. Bobick, A. F. and S. S. Intille, Large occlusion stereo, *Int'l J. Computer Vision*, **33**(3):181–200 (1999).
36. Bolles, R. C., H. H. Baker, and D. H. Marimont, Epipolar-plane image analysis: An approach to determining structure from motion, *Int'l J. Computer Vision*, **1**:7–55 (1987).
37. Bomans, M., K.-H. Hohne, U. Tiede, and M. Riemer, 3-D segmentation of MR images of the dead for 3-D display, *IEEE Transactions on Medical Imaging*, **9**(2):177-183 (1990).
38. Bookstein, F. L., Principal warps: Thin-plate splines and the decomposition of deformations, *IEEE Trans. Pattern Analysis and Machine Intelligence,* **11**(6): 567–585 (1989).
39. Borgefors, G., An improved version of the chamfer matching algorithm, *Int'l J. Conf. Pattern Recognition*, **2**:1175–1177 (1986).
40. Borgefors, G., Distance transforms in digital images, *Computer Vision, Graphics, and Image Processing*, **34**: 344–371 (1986).
41. Borgefors, G., Hierarchical chamfer matching, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**(6):849–865 (1988).
42. Borgefors, G., On digital distance transforms in three dimensions, *Computer Vision and Image Understanding*, **64**(3):368–376 (1996).
43. Boyer, K. L. and A. C. Kak, Color-encoded structured light for rapid active ranging, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **9**(1):14–28 (1987).

44. Braccini, C., G. Gambardella, G. Sandini, and V. Tagliasco, A model of the early stages of the human visual system: Functional and topological transformations performed in the peripheral visual field, *Biological Cybernetics*, **44**:47–58 (1982).
45. Brejl, M. and M. Sonka, Directional 3-D edge detection in anisotropic data: Detector design and performance assessment, *Computer Vision and Image Understanding*, **77**:84–110 (2000).
46. Brigham, E. O. and R. E. Morrow, The fast Fourier transform, *IEEE Spectrum*, 63–70 (1967).
47. Brockelbank, D. C. and Y. H. Yang, An experimental investigation in the use of color in computational stereopsis, *IEEE Trans. Systems, Man, and Cybernetics*, **19**(6):1365–1383 (1989).
48. Brown, L. G. A survey of image registration techniques, *ACM Computing Surveys*, **24**(4):325–276 (1992).
49. Buhmann, M. and N. Dyn, Spectral convergence of multiquadric interpolation, *Proc. Edinburgh Mathematical Society*, **36**:319–333 (1993).
50. Burns, J. B., A. R. Hanson, and E. M. Riseman, Extracting straight lines, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **8**(4):425–455 (1986).
51. Canny, J., A computational approach to edge detection, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **8**:679–714 (1986).
52. Carlson, R. E., The parameter R^2 in multiquadric interpolation, *Computers Math. Applic.*, **21**(9):29–42 (1991).
53. Casasent, D. and D. Psaltis, Position, rotation, and scale invariant optical correlation, *Applied Optics*, **15**(7):1795–1799 (1976).
54. Casasent, D. and D. Psaltis, Scale invariant optical transform, *Optical Engineering*, **15**(3):258–261 (1976).
55. Castleman, K. R., *Digital Image Processing,* Prentice-Hall, Englewood Cliffs, NJ, 161–163, 187–188, 200 (1996).
56. Catmull, E. and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design*, 350–355 (1978).
57. Catmull, E. and R. Rom, A class of local interpolation splines, in *Computer Aided Geometric Design*, R. Barnhill and R. Riesenfeld (Eds.), Academic Press, 317–326 (1974).
58. Chandra, D. V., J. S. Boland, H. S. Ranganath, and W. W. Malcolm, Feature matching multiple image registration using Haar coefficients, *SouthEastcon*, 549–552 (1982).
59. Chang, L. H. T. and H. B. Said, A C^2 Ttriangular patch for the interpolation of functional scattered data, *Computer-Aided Design*, **29**(6):407–412 (1997).
60. Chang, S.-H., F.-H. Cheng, W.-H. Hsu, and G.-H. Wu, Fast algorithm for point pattern matching: Invariant to translation, rotation, and scale changes, *Pattern Recognition*, **30**(2):311–320 (1997).
61. Chavez, P. S., J. Isbrecht, P. Galanis, G. L. Gabel, S. C. Sides, D. L. Soltesz, S. L. Ross, M. G. Velasco, Processing, mosaicking and management of the

Monterey Bay digital sidescan-sonar images, *Marine Geology*, **181**:305–315 (2002).

62. Chen, Y., C. Han, X. Kang, M. Wang, Background information fusion and its application in video target tracking, *Proc. 7th Int'l Conf. Information Fusion (Fusion 2004)*, Stockholm, Sweden, June 28 – July 1, 747–753 (2004).
63. Cheng, F. and A. Goshtasby, A parallel B-spline surface fitting algorithm, *ACM Trans. Graphics*, **8**(1):41–50 (1989).
64. Chiba, N., H. Kano, M. Minoh, and M. Yasuda, Feature-based image mosaicing, *Systems and Computers in Japan*, **31**(7):1–9 (2000).
65. Chui, C. K. and M.-J. Lai, Filling polygonal holes using C^1 cubic triangular spline patches, *Computer Aided Geometric Design*, **17**:297–307 (2000).
66. Chui, H. and A. Rangarajan, A new point matching algorithm for non-rigid registration, *Computer Vision and Image Understanding*, **89**(2/3):114–141 (2003).
67. Clark, J. J., Authenticating edges produced by zero-crossing algorithms, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **11**(1):43–57 (1989).
68. Coelho, C., A. Heller, J. L. Mundy, D. A. Forsyth, and A. Zisserman, An experimental evaluation of projective invariants, in *Geometric Invariance in Computer Vision*, J. L. Mundy and A. Zisserman (Eds.), The MIT Press, Cambridge, MA, 87–104 (1992).
69. Cohen, I., A hybrid hyperquadric model for 2-D and 3-D data fitting, *Computer Vision and Image Understanding*, **63**(3):527–541 (1996).
70. Cole-Rhodes, A. A., K. L. Johnson, J. LeMoigne, and Ilya Zavorin, Multiresolution registration of remote sensing imagery by optimization of mutual information using a stochastic gradient, *IEEE Trans. Image Processing*, **12**(12):1495–1511 (2003).
71. Collins, D. L., A. P. Zijdenbos, V. Killokian, J. G. Sled, N. J. Kabani, C. J. Holmes, and A. C. Evans, Design and construction of a realistic digital brain phantom, *IEEE Trans. Medical Imaging*, **17**:463–468 (1998).
72. Connolly, C. and T. Fliess, A study of efficiency and accuracy in the transformation from RGB to $CIELAB$ color space, *IEEE Trans. Image Processing,* **6**(7):1046–1052 (1997).
73. Conover, W. J., *Practical Nonparametric Statistics*, John Wiley & Sons, 344–386 (1980).
74. Constantini, P. and C. Manni, On a class of polynomial triangular macroelements, *Computational and Applied Mathematics*, **73**:45–64 (1996).
75. Cooley, J. W. and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Mathematics of Computation*, **19**:297–301 (1965).
76. Cornsweet, T. N., *Visual Perception,* Academic Press, New York, NY, 247–251 (1970).
77. Cortelazzo, G. M. and L. Lucchese, A new method of image mosaicking and its application to cultural heritage representation, *Proc. Eurographics '99*, P. Brunet and R. Scopigno (Eds.), **18**(3) (1999).

78. Crowley, J. and A. Parker, A representation for shape based on peaks and ridges in the difference of low-pass transform, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **6**(2):156–170 (1984).

79. Cuisenaire, O. and B. Macq, Fast Euclidean distance transformation by propagation using multiple neighborhoods, *Computer Vision and Image Understanding*, **76**(2):163–172 (1999).

80. Cumani, A., Edge detection in multispectral images, *CVGIP: Graphical Models and Image Processing*, **53**(1):40–51, (1991).

81. Dahmen, W. and C. A. Micchelli, Subdivision algorithms for the generation of box spline surfaces, *Computer Aided Geometric Design*, **1**:115–129 (1984).

82. Dalley, G. and P. Flynn, Pair-wise range image registration: A survey in outlier classification, *Computer Vision and Image Understanding*, **87**:104–115 (2002).

83. Danielsson, P.-E., Euclidean distance mapping, *Computer Graphics and Image Processing*, **14**:227–248 (1980).

84. Davis, L. S., Shape matching using relaxation techniques, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **1**(1):70–72 (1979).

85. DePiero, F. W., M. M. Trivedi, and S. Serbin, Graph matching using a direct classification of node attendance, *Pattern Recognition*, **29**(6):1031–1048 (1996).

86. DePiero, F., Fast landmark-based registration via deterministic and efficient processing, some preliminary results, *Proc. 1st Int'l Sym. 3-D Data Processing Visualization and Transmission*, Padova, Italy, June 19–21, 544–548 (2002).

87. Deriche, R. and G. Giraudon, A computational approach for corner and vertex detection, *Int'l J. Computer Vision*, **10**(2):101–124 (1993).

88. DeRose, T., H. Hoppe, T. Duchamp, J. A. McDonald, and W. Stuetzle, Fitting of surfaces to scattered data, *SPIE Curves and Surfaces in Computer Vision and Graphics III*, **1830**:212–220 (1992).

89. Dev, P., Perception of depth surfaces in random-dot stereograms: A neural model, *Int'l J. Man Machine Studies*, **7**:511–528 (1975).

90. Devijer, P. A. and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall, 232 (1982).

91. Devlin, K., A. Chalmers, A. Wilkie, and W. Purgathofer, Tone reproduction and physically based spectral rendering, *Eurographics, STAR*, 101–123 (2002).

92. Dewdney, A. K., Analysis of a steepest-descent image-matching algorithm, *Pattern Recognition*, **10**:31–39 (1978).

93. Dhome, M. and T. Kasvand, Polyhedra recognition by hypothesis accumulation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **9**(3):429–438 (1987).

94. DiCarlo, J. M. and B. A. Wandell, Rendering high dynamic range images, *Proceedings of the SPIE: Image Sensors*, **3965**:392–401 (2002).

95. Ding, L. and A. Goshtasby, On the Canny edge detector, *Pattern Recognition*, **34**:721–725 (2001).

96. Ding, L. and A. Goshtasby, Registration of multi-modal brain images using the rigidity constraint, *2nd IEEE Int'l Sym. Bioinformatics & Bioengineering,* 1–6 (2001).

97. Ding, L., A. Goshtasby, and M. Satter, Volume image registration by template matching, *Image and Vision Computing,* **19**(12):821–832 (2001).

98. Di Zenzo, S., A note on the gradient of a multi-image, *Computer Vision, Graphics, and Image Processing,* **33**:116–125 (1986).

99. Doo, D. W. H., A subdivision algorithm for smoothing down irregular shaped polyhedrons, *Proc. Interactive Techniques in Computer Aided Design*, **1**:157–165 (1978).

100. Doo, D. and M. Sabin, Behavior of recursive division surfaces near extraordinary points, *Computer Aided Design*, 356–360 (1978).

101. Duda, R. O. and P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures, *Communications of the ACM*, **15**(1):11–15 (1972).

102. Dudani, S. A., K. J. Breeding, and R. B. McGhee, Aircraft identification by moment invariants, *IEEE Transactions on Computers*, **26**(1):39–45 (1977).

103. Dufournaud, Y., C. Schmid, and R. Horaud, Matching images with different resolutions, *Proc. Conf. Computer Vision and Pattern Recognition*, Hilton Head Island, SC, 612–618 (2000).

104. Dvorchenko, V. N., Bounds on (deterministic) correlation functions with applications to registration, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **5**(2):206–213 (1983).

105. Dyn, N., Interpolation of scattered data by radial basis functions, *Topics in Multiquadric Approximation*, Academic Press, New York, NY, 47–61 (1987).

106. Dyn, N., D. Levin, and J. A. Gregory, A butterfly subdivision scheme for surface interpolation with tension control, *ACM Trans. Graphics*, **9**(2):160–169 (1990).

107. Edwards, P. J., D. L. G. Hill, J. A. Little, and D. J. Hawkes, A three-component deformation model for image-guided surgery, *Medical Image Analysis*, **2**:355–367 (1998).

108. Eghbali, H. J., K-S test for detecting changes from Landsat imagery data, *IEEE Trans. Systems, Man, and Cybernetics,* **9**(1):17–23 (1979).

109. El-Hakim, S., J. A. Beraldin, and J. F. Lapointe, Toward automatic modeling of monuments and towers, *Proc. 1st Int'l Sym. 3-D Data Processing Visualization and Transmission*, Padova, Italy, June 19–21, 526–531 (2002).

110. Evans, A. C., W. Dai, L. Collins, P. Neelin, and S. Marrett, Warping of a computerized 3-D atlas to match brain image volumes for quantitative neuroanatomical and functional analysis, *SPIE Image Processing,* **1445**:236–247 (1991).

111. Fang, T. J., Z. H. Huang, L. N. Kanal, B. D. Lavine, G. Stockman, and F. L. Xiong, Three-dimensional object recognition using a transformation clustering technique, *6th Int'l Conf. Pattern Recognition*, 678–681 (1982).

112. Faugeras, O. D. and M. Herbert, The representation, recognition, and locating of 3-D objects, *Int'l J. Robotics Research*, **5**(3):27–52 (1986).

113. Fei, B., A. Wheaton, Z. Lee, J. L. Duerk, and D. L. Wilson, Automatic MR volume registration and its evaluation for the pelvis and prostate, *Physics in Medicine and Biology*, **47**: 823–838 (2002).

114. Feldmar, J., *et al.*, Extension of the ICP algorithm to nonrigid intensity-based registration of 3-D volumes, *Computer Vision and Image Understanding*, **66**(2): 193–206 (1997).

115. Fernández, E. and R. Marti, GRASP for seam drawing in mosaicking of aerial photographic maps, *J. Heuristics*, **5**:181–197 (1999).

116. Ferrari, L. A., P. V. Sankar, J. Sklansky, and S. Leeman, Efficient two-dimensional filters using B-spline functions, *Computer Vision, Graphics, and Image Processing*, **35**:152–169 (1986).

117. Fischler, M. A. and R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, **24**(6):381–395 (1981).

118. Fitzpatrick, J. M., Detecting failure, assessing success, in *Medical Image Registration*, J. V. Hajnal, D. L. G. Hill, and D. J. Hawkes (Eds.), CRS Press, 117–139 (2001).

119. Fitzpatrick, J. M., D. L. G. Hill, Y. Shyr, J. West, C. Studholme, and C. R. Maurer, Jr., Visual assessment of the accuracy of retrospective registration of MR and CT images of the brain, *IEEE Transactions on Medical Imaging*, **17**(4):571–585 (1998).

120. Fitzpatrick, J. M. and J. West, A blinded evaluation and comparison of image registration methods, in *Empirical Evaluation Techniques in Computer Vision*, K. Bowyer and P. J. Phillips (Eds.), IEEE Computer Society Press, Los Alamitos, 12–27 (2000).

121. Floater, M. S. and A. Iske, Multistep scattered data interpolation using compactly supported radial basis functions, *J. Computational and Applied Mathematics*, **73**, 65–78 (1996).

122. Fornefett, M., K. Rohr, and H. S. Stiehl, Radial basis functions with compact support for elastic registration of medical images, *Image and Vision Computing,* **19**:87–96 (2001).

123. Foroosh, H. and W. S. Hoge, Motion information in the phase domain, in *Video Registration*, M. Shah and R. Kumar (Eds.), Kluwer Academic Publishers, Boston, MA 36–71 (2003).

124. Forstmann, S., Y. Kanou, J. Ohya, S. Thuering, and A. Schmitt, Real-time stereo by using dynamic programming, *CVPR Workshop: Real-Time 3-D Sensors and Their Use* (2004).

125. Förstner, W., A feature based correspondence algorithm for image matching, *Int'l Arch. Photogramm. Remote Sensing*, **26**:150–166 (1986).

126. Franke, R., Locally determined smooth interpolation at irregularly spaced points in several variables, *J. Inst. Maths. Applics.*, **19**, 471–482 (1977).

127. Franke, R., Scattered data interpolation: Tests of some methods, *Mathematics of Computation,* **38**(157):181–200 (1982).

128. Franke, R., Recent advances in the approximation of surfaces from scattered data, *Topics in Multivariate Approximation*, C. Chui, L. Shumaker, and F. Utrerus (Eds.), Academic Press, 79–98 (1987).
129. Franke, R. and G. Nielson, Smooth interpolation of large sets of scattered data, *Int'l J. Numerical Methods in Engineering*, **15**:1691–1704 (1980).
130. Franke, R. and L. L. Schumaker, A bibliography of multivariate approximation, *Topics in Multivariate Approximation*, Academic Press, 275–335 (1987).
131. Frantz, S., K. Rohr, and H. Siegfried, Multi-step differential approaches for the localization of 3-D point landmarks in medical images, *J. Computing and Information Technology*, **6**:435–447 (1998).
132. Frieden, B. R., Restoring with maximum likelihood and maximum entropy, *J. Optical Society America*, **62**:511–518 (1972).
133. Fryer, J. G. and D. C. Brown, Lens distortion for close-range photogrammetry, *Photogrammetric Engineering and Remote Sensing*, **52**(1):51–58 (1986).
134. Fua, P., Reconstructing complex surfaces from multiple stereo views, *Proc. Computer Vision and Pattern Recognition*, 1078–1085 (1995).
135. Garcias, N. and J. Santos-Victor, Underwater video mosaics as visual navigation maps, *Computer Vision and Image Understanding*, **79**:66–91 (2000).
136. Garrett, G. S., E. L. Reagh, and E. B. Hibbs, Jr., Detection threshold estimation for digital area correlation, *IEEE Trans. Systems, Man, and Cybernetics*, **1**:65–70 (1976).
137. Gee, J. C., Performance evaluation of medical image processing algorithms, *Medical Imaging 2000, Image Processing*, K. M. Hanson (Ed.), *Proc. SPIE*, **3979**:19–27 (2000).
138. Gelfand, N., L. Ikemoto, S. Rusinkiewicz, and M. Levoy, Geometrically stable sampling for the ICP algorithm, *Proc. 4th Int'l Conf. 3-D Digital Imaging and Modeling*, Banff, Alberta, Canada, Oct. 6–10, 260–267 (2003).
139. Gennery, D. B., A stereo vision system for an autonomous vehicle, *5th Int'l J. Conf. Artificial Intelligence*, 576–582 (1977).
140. Gerlot, P. and Y. Bizais, Image registration: A review and a strategy for medical applications, *Information Processing in Medical Imaging*, C. N. de Graaf and M. A. Viergever (Eds.), Plenum Press, 81–89 (1988).
141. Gershon, R., A. D. Jepson, and J. K. Tsotsos, Ambient illumination and the determination of material changes, *J. Opt. Soc. Am. A*, **3**:10, 1700–1707 (1986).
142. Gonzalez, R. C. and P. Wintz, *Digital Image Processing*, Second Edition, Addison-Wesley, Reading, MA, 221–224 (1987).
143. Gordon, W. J. and J. A. Wixom, Shepard's method of "metric interpolation" to bivariate and multivariate interpolation, *Mathematics of Computation*, **32**:253–264 (1978).
144. Goshtasby, A., A refined technique for stereo depth perception, *Proc. IEEE Computer Society Workshop on Computer Vision*, 125–129 (1984).
145. Goshtasby, A., Template matching in rotated images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **7**(3):338–344 (1985).

146. Goshtasby, A., Description and discrimination of planar shapes using shape matrices, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **7**(6):738–743 (1985).

147. Goshtasby, A., Piecewise linear mapping functions for image registration, *Pattern Recognition,* **19**(6):459–466 (1986).

148. Goshtasby, A., Piecewise cubic mapping functions for image registration, *Pattern Recognition*, **20**(5):525–533 (1987).

149. Goshtasby, A., Registration of image with geometric distortion, *IEEE Trans. Geoscience and Remote Sensing,* **26**(1):60–64 (1988).

150. Goshtasby, A., Image registration by local approximation methods, *Image and Vision Computing,* **6**(4):255–261 (1988).

151. Goshtasby, A., Correction of image deformation from lens distortion using Bézier patches, *Computer Vision, Graphics, and Image Processing*, **47**:385–394 (1989).

152. Goshtasby, A., Surface reconstruction from scattered measurements, *SPIE Curves and Surfaces in Computer Vision and Graphics III*, **1830**:247–255 (1992).

153. Goshtasby, A., Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces, *Int. J. Computer Vision*, **10**(3):233–256 (1993).

154. Goshtasby, A., On edge focusing, *Image and Vision Computing*, **12**(4):247–256 (1994).

155. Goshtasby, A., Edge detection by curve fitting, *Image and Vision Computing,* **13**(3):169–177 (1995).

156. Goshtasby, A., Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces, *Int'l J. Computer Vision,* **10**(3):233–256 (1995).

157. Goshtasby, A., Three-dimensional model construction from multiview range images: Survey with new results, *Pattern Recognition*, **31**(11):1705–1714 (1998).

158. Goshtasby, A., F. Cheng, and B. A. Barksy, B-spline curves and surface viewed as digital filters, *Computer Vision, Graphics, and Image Processing*, **52**:264–275 (1990).

159. Goshtasby, A., S. H. Gage, and J. F. Bartholic, A two-stage cross correlation approach to template matching, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **6**(3):374–378 (1984).

160. Goshtasby, A. and J. LeMoigne, *Image Registration*, a special issue of *Pattern Recognition*, Jan. 1999.

161. Goshtasby, A. and W. D. O'Neill, Surface fitting to scattered data by a sum of Gaussians, *Computer-Aided Geometric Design,* **10**:143–15 (1993).

162. Goshtasby, A. and C. Page, A multiple image segmentation technique with subpixel accuracy, *Proc. Computer Vision and Pattern Recognition*, 157–158 (1983).

163. Goshtasby, A. and G. C. Stockman, Point pattern matching using convex-hull edges, *IEEE Trans. Systems, Man, and Cybernetics*, **15**(5):631–637 (1985).

164. Goshtasby, A., G. Stockman, and C. Page, A region-based approach to digital image registration with subpixel accuracy, *IEEE Trans. Geoscience and Remote Sensing*, **24**(3):390–399 (1986).

165. Goshtasby, A. and D. A. Turner, Segmentation of cardiac cine MR images for extraction of right and left ventricular chambers, *IEEE Trans. Medical Imaging*, **14**(1):56–64, (1995).

166. Goshtasby, A., L. Staib, C. Studholme, and D. Terzopoulos, *Non-Rigid Image Registration*, a special issue of *Computer Vision and Image Understanding*, Feb. 2003.

167. Granlund, G. H., Fourier preprocessing for hand print character recognition, *IEEE Trans. Computers*, 195–201 (1972).

168. Green, P. J. and R. Sibson, Computing Dirichlet tessellation in the plane, *Computer Journal,* **21**:168–173 (1978).

169. Grimson, W. E. L., An implementation of a computational theory of visual surface interpolation, *Computer Vision, Graphics, and Image Processing*, **22**:39–69 (1983).

170. Grimson, W. E. L. and D. P. Huttenlocher, On the sensitivity of the Hough transform for object recognition, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**(3):255–276 (1990).

171. Grosse, E., A catalogue of algorithms for approximation, *Algorithms for Approximation II*, J. Mason, and M. Cox (Eds.), Chapman and Hall, 479–514 (1990).

172. Grova, C., P. Jannin, A. Biraben, I. Buvat, H. Benali, A. M. Bernard, B. Gibaud, and J. M. Scarabin, Validation of MRI/SPECT similarity-based registration methods using realistic simulations of normal and pathological SPECT data, *Computer Assisted Radiology and Surgery (CARS 2002)*, H. U. Lemke, M. V. Vannier, K. Inamura, A. G. Farman, K. Doi, and J. H. C. Reiber (Eds.), Springer, 450–455 (2002).

173. Gunadi, C. R., H. Shimizu, K. Kodama, and K. Aizawa, Construction of large-scale virtual environment by fusing range data, texture images, and airborne altimetry data, *Proc. 1st Int'l Sym. 3-D Data Processing Visualization and Transmission*, Padova, Italy, June 19–21, 772–775 (2002).

174. Habib, A. and J. Warren, Edge and vertex insertion for a class of C1 subdivision surfaces, *Computer Aided Geometric Design*, **16**:223–247 (1999).

175. Hajnal, J. V., D. L. G. Hill, and D. J. Hawkes, *Medical Image Registration*, CRS Press (2001).

176. Hall, E. L., Almost uniform distributions for computer image enhancement, *IEEE Trans. Computers*, **23**(2):207–208 (1974).

177. Hall, E. L., *Computer Image Processing and Recognition*, Academic Press, 29–30 (1979).

178. Haralick, R. M. and L. G. Shapiro, The consistent labeling problem: Part I, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **1**(2):173–184 (1979).

179. Haralick, R. M. and L. G. Shapiro, The consistent labeling problem: Part II, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **2**(3):193–203 (1980).

180. Haralick, R. M., Ridges and valleys on digital images, *Computer Vision, Graphics, and Image Processing*, **22**:28–38 (1983).

181. Harder, R. L. and R. N. Desmarais, Interpolation using surface splines, *J. Aircraft,* **9**(2):189–191 (1972).

182. Hardy, R. L., Multiquadric equations of topography and other irregular surfaces, *Journal of Geophysical Research,* **76**(8):1905–1915 (1971).

183. Hardy, R. L., Theory and applications of the multiquadric-biharmonic method – 20 years of discovery – 1969-1988, *Computers Math. Applic.*, **19**(8/9):163–208 (1990).

184. Harris, C. and M. Stephens, A combined corner and edge detector, *Alvey Vision Conference*, 147–151 (1988).

185. Hartkens, T., K. Rohr, and H. S. Stiehl, Evaluation of 3-D operators for the detection of anatomical point landmarks in MR and CR images, *Computer Vision and Image Understanding*, **18**:118–136 (2002).

186. Heath, M. D., S. Sarkar, T. Sanocki, and K. W. Bowyer, A robust visual method for assessing the relative performance of edge-detection algorithms, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **19**(12):1338–1359 (1997).

187. Heitger, F., L. Rosenthaler, von der Heydt, E. Peterhans, and O. Kuebler, Simulation of neural contour mechanism: From simple to end-stopped cells, *Vision Research*, **32**(5):963–981 (1992).

188. Hill, B., Th. Roger, and F. W. Vorhagen, Comparative analysis of the quantization of color spaces on the basis of the CIELAB color-difference formula, *ACM Trans. Graphics*, **11**(4):373–405 (1992).

189. Hill, D. L. G., D. J. Hawkes, J. E. Crossman, M. J. Gleeson, T. C. S. Fox, E. C. Bracey, A. J. Strong, and P. Graves, Registration of MR and CT images for skull base surgery using point-like anatomical features, *The British Journal of Radiology*, **64**(767):1030–1035 (1991).

190. Hoppe, H., T. DeRose, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, *Proceedings SIGGRAPH*, 71–78 (1992).

191. Hoppe, H., T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, Piecewise smooth surface reconstruction, *Proc. SIGGRAPH*, 295–302 (1994).

192. Horn, B. K. P., *Robot Vision,* McGraw-Hill, New York, NY, 187, 205 (1986).

193. Horn, B. K. P. and R. J. Woodham, *Proc. Machine Processing of Remotely Sensed Data*, 59–68 (1979).

194. Hoshino, J. and M. Kourogi, Fast panoramic image mosaicing using one-dimensional flow estimation, *Real-Time Imaging*, **8**:95–103 (2002).

195. Hou, H. S. and H. C. Andrews, Cubic splines for image interpolation and digital filtering, *IEEE Trans. Acoustics, Speech, and Signal Processing*, **26**(6):508–517 (1978).

196. Hough, P. V. C., *Method and means for recognizing complex patterns*, U. S. Patent 3,069, 654, Dec. 18, 1962.

197. Hsieh, J.-W., Fast stitching algorithm for moving object detection and mosaic construction, *Image and Vision Computing*, **22**:291–306 (2004).

198. Hu, M.-K., Visual pattern recognition by moment invariants, *Trans. Information Theory*, 179–187 (1962).

199. Hueckel, M., An operator which locates edges in digital pictures, *J. ACM*, **18**(1):113–125 (1971).

200. Hummel, R. A. and S. W. Zucker, On the foundations of relaxation labeling procedures, *Pattern Recognition and Image Processing*, 50–53 (1980).

201. Hurvick, L. M. and D. Jameson, *The Perception of Brightness and Darkness,* Allyn and Bacon, Boston, MA, 5–9 (1966).

202. Huttenlocher, D. P., G. A. Klanderman, and W. J. Rucklidge, Comparing images using the Hausdorff distance, *IEEE Trans. Pattern Intelligence and Machine Intelligence*, **9**:850–863 (1993).

203. Ikemoto, L., N. Gelfand, and M. Levoy, A hierarchical method for aligning warped meshes, *Proc. 4th Int'l Conf. 3-D Digital Imaging and Modeling*, Banff, Alberta, Canada, Oct. 6–10, 434–441 (2003).

204. Illingworth, J. and J. Kittler, A survey of the Hough transform, *Computer Vision, Graphics, and Image Processing*, **44**:87–116 (1988).

205. Jackowski, M., A. Goshtasby, S. Bines, D. Roseman, and C. Yu, Correcting the geometry and color of digital images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **19**(10): 1152–1158 (1997).

206. Jackson, I. R. H., Convergence properties of radial basis functions, *Constructive Approximation*, **4**:243–264 (1988).

207. Jain, R., R. Katsuri, and B. G. Schnuck, *Machine Vision*, 33–35 (1995).

208. Jannin, P., J. M. Fitzpatrick, D. J. Hawkes, X. Pennec, R. Shahidi, and M. W. Vannier, Validation of medical image processing in image-guided therapy, *IEEE Trans. Medical Imaging*, **21**(12):1445–1449 (2002).

209. Jenkin, M. and P. A. Kolers, Some problems with correspondence, in *Motion Understanding: Robot and Human Vision*, W. N. Martin and J. K. Aggarwal (Eds.), Kluwer Academic Publishing, 269–295 (1988).

210. Jiang, X. Y. and H. Bunke, Line segment based axial motion stereo, *Pattern Recognition*, **28**(4):553–562 (1995).

211. Jones, G. A., Constraint, optimization, and hierarchy: Reviewing stereoscopic correspondence of complex features, *Computer Vision and Image Understanding*, **65**(1):57–78 (1997).

212. Jones, M. G., S. G. Nikolov, Dual-modality two-stage direct volume rendering, *Proc. 6th Int'l Conf. Information Fusion (Fusion 2003)*, Cairns, Queensland, Australia, July 8–11, 845–851 (2003).

213. Jordan, J. R., III and A. C. Bovik, Using chromatic information in edge-based stereo correspondence, *CVGIP: Image Understanding*, **54**(1):98–118 (1991).

214. Jordan, J. R., III, and A. C. Bovik, Using chromatic information in dense stereo correspondence, *Pattern Recognition*, **25**(4):367–383 (1992).

215. Jost, T. and H. Hügli, A multi-resolution ICP with heuristic closest point search for fast and robust 3-D registration of range images, *Proc. 4th Int'l Conf. 3-D Digital Imaging and Modeling*, Banff, Alberta, Canada, Oct. 6–10, 427–433 (2003).

216. Julesz, B. and J. J. Chang, Interaction between tools of binocular disparity detectors tuned to different disparities, *Biological Cybernetics*, **22**:107–120 (1976).

217. Julesz, B. and J. E. Miller, Independent spatial-frequency-tuned channels in binocular fusion and rivalry, *Perception*, **4**:125–143 (1975).

218. Kamgar-Parsi, B. and B. Kamgar-Parsi, An invariant, closed form solution for matching sets of 3-D lines, *Proc. Conf. Computer Vision and Pattern Recognition*, II431–II436 (2004).

219. Kang, S. B., J. A. Webb, C. L. Zitnick, and T. Kanade, A multi-baseline stereo system with active illumination and real-time image acquisition, *Proc. 5th Int'l Conf. Computer Vision*, 88–93 (1995).

220. Kass, M., A. Witkin, and D. Terzopoulos, Snakes: Active contour models, *Int'l J. Computer Vision*, 321-331 (1988).

221. Kasson, J. K. and W. Plouffe, An analysis of selected computer interchange color spaces, *ACM Trans. Graphics,* **11**(4):373–405 (1992).

222. Kerschner, M., Seamline detection in color ortho-image mosaicking by use of twin snakes, *Photogrammetry and Remote Sensing*, **56**:53–64 (2001).

223. Keys, E. G., Cubic convolution interpolation for digital image processing, *IEEE Trans. Acoustics, Speech, and Signal Processing,* **29**(6):1153–1160 (1981).

224. Kitchen, L., Relaxation applied to matching quantitative relational structures, *IEEE Trans. Systems, Man, and Cybernetics*, **10**(2):96–101 (1980).

225. Kitchen, L. and A. Rosenfeld, Gray-level corner detection, *Pattern Recognition Letters*, **1**:95–102 (1982).

226. Kitchen, L. J., Relaxation for point-pattern matching: What it really computes? *Proc. Computer Vision and Pattern Recognition*, 405–407 (1985).

227. Kittler, J. and J. Illingworth, Relaxation labeling algorithms – A review, *Image and Vision Computing*, **3**(4):206–216 (1985).

228. Kobbelt, L., Interpolatory subdivision on open quadrilateral nets with arbitrary topology, in *Proc. Eurographics '96, Computer Graphics Forum*, 409–420 (1996).

229. A. Koschan, V. Rodehorst, and K. Spiller, Color stereo vision using hierarchical block matching and active color illumination, *Proc. 13th Int'l Conf. Pattern Recognition*, Vienna, Austria, **1**:835–839 (1996).

230. Kuschel, S. A. and C. V. Page, Augmented relaxation labeling and dynamic relaxation Labeling, *Pattern Recognition and Image Processing*, 441–448 (1981).

231. Kutulakos, K. N. and S. M. Seitz, *A Theory of Shape by Space Carving*, University of Rochester, Computer Science Department, TR 692, 1–19 (1998).

232. Lane, R. A., N. A. Thacker, and N. L. Seed, Stretch-correlation as a real-time alternative to feature-based stereo matching algorithms, *Image and Vision Computing*, **12**(4):203–212 (1994).

233. Lawson, C. L., Software for C^1 surface interpolation, in *Mathematical Software III*, J. R. Rice (Ed.), Academic Press, London, 161–194 (1977).

234. Leader, J. L., *Numerical Analysis and Scientific Computation*, Addison-Wesley, New York, NY, 2005.

235. Lee, H-Y, T. Kim, W. Park, and H. K. Lee, Extraction of digital elevation models from satellite stereo images through stereo matching based on epipolarity and scene geometry, *Image and Vision Computing*, **21**:789–7906 (2003).

236. Lee, S-M, A. L. Abbott, and D. L. Schmoldt, Wavelet-based hierarchical surface approximation from height fields, *Proc. Conf. Computer Vision and Pattern Recognition*, I299–I305 (2004).

237. Leese, J. A., G. S. Novak, and B. B. Clark, An automatic technique for obtaining cloud motion from geosynchronous satellite data using cross correlation, *A. Applied Meteorology*, **10**:110–132 (1971).

238. Le Moigne, J., W. Xie, S. Chettri, T. El-Ghazawi, E. Kaymaz, B-T Lerner, M. Mareboyana, N. Natanyahu, J. Pierce, S. Raghavan, J. C. Tilton, W. J. Campbell, and R. F. Cromp, Toward an intercomparison of automated registration algorithms for multiple source remote sending data, *Proc. Image Registration Workshop*, Goddard Space Flight Center, Greenbelt, MD, U.S.A., Nov. 20–21, 307–316 (1997).

239. Lester, H. and S. R. Arridge, A survey of hierarchical non-linear medical image registration, *Pattern Recognition*, **32**:129–149 (1999).

240. Levin, D. N., C. A. Pelizzari, G. T. Y. Chen, C.-T. Chen, and M. D. Cooper, Retrospective geometric correlation of MR, CT, and PET images, *Radiology*, **169**(3):817–823 (1988).

241. Levine, M. D., D. O. O'Handley, and G. M. Yagi, Computer determination of depth maps, *Computer Graphics and Image Processing*, **2**:131–150 (1973).

242. Levine, M. W. and J. M. Shefner, *Depth Perception, Fundamentals of Sensation and Perception*, Addison-Wesley Publishing Co., 250–274 (1981).

243. Lewis, J. J., R. J. O'Callaghan, S. G. Nikolov, D. R. Bull, C. N. Canagarajah, Region-based image fusion using complex wavelets, *Proc. 7th Int'l Conf. Information Fusion (Fusion 2004)*, Stockholm, Sweden, June 28 – July 1, 555–562 (2004).

244. Lin, M. H. and C. Tomasi, Surfaces with occlusions from layered stereo, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **26**(8):1073–1078 (2004).

245. Linnainmaa, S., D. Harwood, and L. S. Davis, Pose determination of a three-dimensional object using triangle pairs, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**(5):634–647 (1988).

246. Liu, H. C. and M. D. Srinath, Partial shape classification using contour matching in distance transformation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**:1072–1079 (1990).

247. Loop, C., *Smooth subdivision surfaces based on triangles*, Master's thesis, Department of Mathematics, University of Utah, 1987.

248. López, A. M., D. Lloret, J. Serrat, and J. J. Villanueva, Multilocal creaseness based on the level set extrinsic curvature, *Computer Vision and Image Understanding*, **77**:111–144 (2000).

249. Lowe, D. G., Object recognition from local scale-invariant features, *Proc. 7th Int'l Conf. Computer Vision*, Kerkyra, Greece, 1150–1157 (1999).

250. Lowe, D. G., Distinctive image features from scale-invariant keypoints, *Int'l J. Computer Vision*, **60**(2):91–110 (2004).

251. Ma, B., R. E. Ellis, Robust registration for computer-integrated orthopedic surgery: Laboratory validation and clinical experience, *Medical Image Analysis*, **7**:237–250 (2003).

252. MacVicar-Whelan, P. J. and T. O. Binford, Line finding with subpixel precision, *Proceedings DARPA Image Understanding Workshop*, 26–31 (1981).

253. Maes, F., A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, Multimodality image registration by maximization of mutual information, *IEEE Trans. Medical Imaging*, **16**(2):187–198 (1997).

254. Maes, F., D. Vandermeulen, and P. Suetens, Comparative evaluation of multiresolution optimization strategies for multimodality image registration by maximization of mutual information, *Medical Image Analysis*, **3**(4):373–386 (1999).

255. Maillot, J. and J. Stam, A unified subdivision scheme for polygonal modeling, *EUROGRAPHICS*, A. Chalmers and T.-M. Rhyne (Eds.), **20**(3), (2001).

256. Maintz, J. B. A. and M. A. Viergever, A survey of medical image registration, *Medical Image Analysis*, **2**(1):1–36 (1988).

257. Malcolm, M. A. and J. Palmer, A fast method for solving a class of tridiagonal linear systems, *Communications of the ACM*, **17**:14–17 (1974).

258. Mansouri, A. R., A. S. Malowany, and M. D. Levine, Line detection in digital pictures: A hypothesis prediction/verification paradigm, *Computer Vision, Graphics, and Image Processing*, **40**:95–114 (1987).

259. Marchand E. and F. Chaumette, An autonomous active vision system for complete and accurate 3-D scene reconstruction, *Int'l J. Computer Vision*, **32**(3):171–194 (1999).

260. Mardia, K., J. T. Kent, C. R. Goodall, and J. Little, Kriging and splines with derivative information, *Biometrika*, **83**(1):207–221 (1996).

261. Marr D. and E. Hildreth, Theory of edge detection, *Proc. R. Soc. Lond.*, **207**:187–217 (1980).

262. Marr, D., G. Palm, and T. Poggio, Analysis of a cooperative stereo algorithm, *Biological Cybernetics*, **28**:223–239 (1978).

263. Marr, D. and T. Poggio, Cooperative computation of stereo disparity, *Science*, **194**:283–287 (1976).

264. Marr, D. and T. Poggio, A computational theory of human stereo vision, *Proc. R. Soc. Lond.*, **204**:301–328 (1979).

265. Marzotto, R. A. Fusiello, and V. Murino, High resolution video mosaicing with global alignment, *Proc. Conf. Computer Vision and Pattern Recognition*, I692–I698 (2004).
266. Massone, L., G. Sandini, and V. Tagliasco, "Form-Invariant" topological mapping strategy for 2-D shape recognition, *Computer Vision, Graphics, and Image Processing*, **30**:169–188 (1985).
267. Maude, A. D., Interpolation–mainly for graph plotters, *The Computer Journal,* **16**(1):64–65 (1973).
268. Maurer, C. R., Jr. and J. M. Fitzpatrick, A review of medical image registration, *Interactive Image-Guided Neurosurgery*, 17–44 (1993).
269. Maurer, C. R., Jr., J. M. Fitzpatrick, M. Y. Wang, R. L. Galloway, R. M. Maciunas, and G. S. Allen, Registration of head volume images using implantable fiducial markers, *IEEE Trans. Medical Imaging*, **16**(4):447–462 (1997).
270. Mayhew, J. E. W. and J. P. Frisby, Psychophysical and computational studies towards a theory of human stereopsis,*Artificial Intelligence*, **17**:349–385 (1981).
271. McInerney, I. and D. Terzopoulos, T-snakes: Topology adaptive snakes, *Medical Image Analysis*, **4**:73-91 (2000).
272. McLain, D. H., Two-dimensional interpolation from random data, *The Computer Journal*, **19**:178–181 (1976).
273. Meinguet, J., An intrinsic approach to multivariate spline interpolation at arbitrary points, in *Polynomial and Spline Approximation,* B. N. Sahney (Ed.), D. Reidel Publishing Company, 163–190 (1979).
274. Micchelli, C. A., Interpolation of scattered data: Distance matrices and computationally positive definite functions, *Constr. Approx.*, **2**:11–22 (1986).
275. Mikolajczyk, K. and C. Schmid, Scale and affine invariant interest point detectors, *Int'l J. Computer Vision*, **60**(1):63–86 (2004).
276. Montanari, U., A method for obtaining skeletons using a quasi-Euclidean distance, *Journal of the Association for Computing Machinery*, **15**(4):600–624 (1968).
277. Moravec, H. P., Towards automatic visual obstacle avoidance, *5th Int'l J. Conf. Artificial Intelligence*, 584 (1977).
278. Moravec, H. P., Rover visual obstacle avoidance, *Int'l J. Conf. Artificial Intelligence*, 785–790 (1981).
279. Mori, K. I., M. Kidode, and H. Asada, An iterative prediction and correction method for automatic stereocomparison, *Computer Graphics and Image Processing*, **2**:393–401 (1973).
280. Mortenson, M. E., *Geometric Modeling*, Wiley Computer Publishing, Second Edition (1997).
281. Mount, D. M., N. S. Netanyahu, and J. Le Moigne, Efficient algorithms for robust feature matching, *Pattern Recognition*, **32**:17–38 (1999).
282. Mühlmann, K., D. Maier, J. Hesser, and R. Männer, Calculating dense disparity maps from color stereo images, an efficient implementation, *Int'l J. Computer Vision*, **47**(1/2/3):79–88 (2002).

283. Mundy, J. L. and A. Zisserman, *Geometric Invariance in Computer Vision*, The MIT Press, Cambridge, MA (1992).

284. Nayar, S. K. and T. Mitsunaga, High dynamic range imaging: Spatially varying pixel exposures, *IEEE Conf. Computer Vision and Pattern Recognition*, **1**:472–479 (2000).

285. Neuenschwander, W. M., P. Fua, L. Iverson, G. Szekely, and O. Kubler, Ziploc Snakes, *Int'l J. Computer Vision*, **25**(3):191–201 (1997).

286. Nevatia, R., Depth measurement by motion stereo, *Computer Graphics and Image Processing*, **15**:203–214 (1976).

287. Nevatia, R., A color edge detector and its use in scene segmentation, *IEEE Trans. Systems, Man, and Cybernetics*, **7**(11):820–826 (1977).

288. Nikolov, S. G., D. R. Bull, C. N. Canagarajah, M. Halliwell, and P. N. T. Wells, 2-D image fusion by multiscale edge graph combination, *Proc. 3rd Int'l Conf. Information Fusion*, Paris, France, **1**: MoD3-16–22 (2000).

289. O'Donnell, M. and W. A. Edelstein, NMR imaging in the presence of magnetic field inhomogeneities and gradient field nonlinearities, *Medical Physics*, **12**(1):20–26 (1985).

290. Ogawa, H. Labeled point-pattern matching by fuzzy relaxation, *Pattern Recognition*, **17**:569–573 (1984).

291. Ohta, Y. and T. Kanade, Stereo by intra- and inter-scanline search, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **7**(2):139–154 (1985).

292. Ohtake, Y., A. Belyaeu, M. Alexa, G. Turk, and H-P Seidel, Multi-level partition of unity implicits, *ACM SIGGRAPH*, 463–473 (2003).

293. Okutomi, M. and T. Kanade, A locally adaptive window for signal matching, *Int'l J. Computer Vision*, **7**(2):143–162 (1992).

294. Okutomi, M. and T. Kanade, A multiple-baseline stereo system, *IEEE Trans. Pattern Analysis and Machine Intelligence,* **15**(4):353–363 (1993).

295. Olson, C. F., Efficient pose clustering using a randomized algorithm, *Int'l J. Computer Vision*, **23**(2):131–147 (1997).

296. Orrite, C. and J. E. Herrero, Shape matching of partially occluded curves invariant under projective transformation, *Computer Vision and Image Understanding*, **93**:34–64 (2004).

297. Oppenheim, A. V. and J. S. Lim, The importance of phase in signals, *Proc. IEEE*, **69**(5):529–541 (1981).

298. Panton, D. J., A flexible approach to digital stereo mapping, *Photogrammetric Engineering and Remote Sensing*, **44**(12):1499–1512 (1978).

299. Parui, S. K. and D. D. Majumder, A new definition of shape similarity, *Pattern Recognition Letters*, **1**:37–42 (1982).

300. Pelizzari, C. A., G. T. Y. Chen, D. R. Spelbring, R. R. Weichselbaum, and C.-T. Chen, Accurate three-dimensional registration of CT, PET, and/or MR images of the brain, *J. Computer Assisted Tomography*, **13**(1):20–26 (1989).

301. Pennec, X., N. Ayache, and J-P. Thirion, Landmark-based registration using features identified through differential geometry, *Handbook of Medical Imaging,* I. N. Bankman (Ed.), Academic Press, 499–513 (2000).
302. Penney, G. P., J. Weese, J. A. Little, P. Desmedt, D. L. G. Hill, and D. Hawkes, A comparison of similarity measures for use in 2-D and 3-D medical image registration, *IEEE Trans. Medical Imaging,* **17**(4):586–595 (1998).
303. Pentland, A. P., Interpolation using wavelet bases, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **16**(4):410–414 (1994).
304. Pernus, F., S. H. Siegfried, and M. Vierveger, *Biomedical Image Registration*, a special issue of *Image and Vision Computing*, January 2001.
305. Persoon, E. and K.-S. Fu, Shape discrimination using Fourier descriptors, *IEEE Transactions on Systems, Man, and Cybernetics*, **7**(3):170–179 (1977).
306. Peterfreund, N. The velocity snake: Deformable contour for tracking in spatio-velocity space, *Computer Vision and Image Understanding*, **73**(3):346–356 (1999).
307. Peters, J., Local cubic and bicubic surface interpolation with linearly varying boundary normal, *Computer Aided Geometric Design*, **7**:499–516 (1990).
308. Peters, J. and U. Reif, The simplest subdivision scheme for smoothing polyhedra, *ACM Trans. Graphics*, **16**(4):420–431 (1997).
309. Petrović, V. S. and C. S. Xydeas, Gradient-based multiresolution image fusion, *IEEE Trans. Image Processing*, **13**(2):228–237 (2004).
310. Piella, P., A region-based multiresolution image fusion algorithm, *Proc. 5th Int'l Conf. Information Fusion (Fusion 2002)*, Annapolis, MD (Washington DC Area), USA, July 8–11, 1557–1564 (2002).
311. Pluim, J. P. W., J. B. A. Maintz, and M. A. Viergever, Mutual-information-based image registration of medical images: A survey, *IEEE Trans. Medical Imaging*, **22**(8):986–1004 (2003).
312. Pluim, J. P. W. and J. M. Fitzpatrick, *Image Registration*, a special issue of *IEEE Trans. Medical Imaging*, Nov. 2003.
313. Poggio, G. F. and T. Poggio, The analysis of stereopsis, *Ann. Rev. Neurosci.*, **7**:379–412 (1984).
314. Powell, M. J. D., Radial basis functions for multivariate interpolation: A review, in *Algorithms for Approximation*, J. C. Mason and M. G. Cox (Eds.), Clarendon Press, Oxford, 143–167 (1987).
315. Pratt, W. K., Correlation techniques for image registration, *IEEE Trans. Aerospace and Electronic Systems*, **10**(3):353–358 (1974).
316. Preparata, F. P. and M. I. Shamos, *Computational Geometry: An Introduction,* Springer-Verlag, New York, NY, 219–233 (1985).
317. Pulli, K., Multiview registration for large data sets, *Proc. 2nd Int'l Conf. 3-D Digital Imaging and Modeling*, Ottawa, Canada, Oct. 4–8, 160–168 (1999).
318. Qu, R. and R. P. Agarwal, Smooth surface interpolation to scattered data using interpolatory subdivision algorithms, *Computers Math. Applic.*, **32**(3):93–110 (1996).

319. Quam, L. H., Hierarchical warp stereo, *Proc. DARPA Image Understanding Workshop*, New Orleans, LA, 149–15 (1984).

320. Ramalingam, S. and S. K. Lodha, Adaptive enhancement of 3-D scenes using hierarchical registration of texture-mapped 3-D models, *Proc. 2nd Int'l Conf. 3-D Digital Imaging and Modeling*, Ottawa, Canada, Oct. 4–8, 203–210 (1999).

321. Rangarajan, K., M. Shah, and D. Van Brackle, Optimal corner detection, *Computer Vision, Graphics, and Image Processing*, **48**:230–245 (1989).

322. Reisfeld, D., H. Wolfson, and Y. Yeshurun, Context-free attentional operators: The generalized symmetry transform, *Int'l J. Computer Vision*, **14**(2):119–130 (1995).

323. Richter, R., Atmospheric correction of satellite data with haze removal including a haze/clear transition region, *Computers and Geoscience*, **22**(6):675–681 (1996).

324. Rivlin, T. J., *An Introduction to the Approximation of Functions*, Blaisdell Publishing, 48–65 (1969).

325. Roberts, L. G., *Machine Perception of 3-D Solids*, Ph.D. Thesis, MIT (1963).

326. Robinson, G. S., Color edge detection, *Optical Engineering*, **16**(5):479-484 (1977).

327. Rohr, K., Modeling and identification of characteristic intensity variations, *Image and Vision Computing*, **10**:66–76 (1992).

328. Rohr, K., Localization properties of direct corner detectors, *J. Mathematical Imaging and Vision*, **4**:139–150 (1994).

329. Rohr, K., On 3D differential operators for detecting point landmarks, *Image and Vision Computing*, **15**:219–233 (1997).

330. Rohr, K., Extraction of 3D anatomical point landmarks based on invariance principles, *Pattern Recognition*, **32**:3–15 (1999).

331. Rohr, K., *Landmark-Based Image Analysis: Using Geometric and Intensity Models*, Kluwer Academic Publishers, Boston, MA, 2001.

332. Rohr, K., H. S. Stiehl, R. Sprengel, T. M. Buzug, J. Weese, and M. H. Kuhn, Landmark-based elastic registration using approximating thin-plate splines, *IEEE Trans. Medical Imaging,* **20**(6):526–534 (2001).

333. Rosenfeld, A. and J. L. Pfaltz, Sequential operations in digital picture processing, *Journal of the Association for Computing Machinery*, **13**(4):471–494 (1966).

334. Rosenfeld, A., R. A. Hummel, and S. W. Zucker, Scene labeling by relaxation operations, *IEEE Trans. Systems, Man, and Cybernetics*, **6**(6):420–433 (1976).

335. Rosenfeld, A. and G. J. Vanderbrug, Coarse-fine template matching, *IEEE Trans. Systems, Man, and Cybernetics*, 104–107 (1977).

336. Rucklidge, W. J., Efficiently locating objects using the Hausdorff distance, *Int'l J. Computer Vision*, **24**(3):251–270 (1997).

337. Sapiro, G. Color snakes, *Computer Vision and Image Understanding,* **68**(2):247–253 (1997).

338. Sawchuk, A. A., Space-variant image motion degradation and restoration, *Proc. IEEE*, **60**:854–861 (1972).

339. Sawchuk, A. A. and M. J. Peyrovian, Space-variant image restoration by coordinate transformation, *J. Optical Society of America*, **64**(2):138–144 (1974).

340. Schad, L., S. Lott, F. Schmitt, V. Sturm, and W. J. Lorenz, Correction of spatial distortion in MR imaging: A prerequisite for accurate stereotraxy, *J. Computer Assisted Tomography*, **11**(3):499–505 (1987).

341. Schagen, I. P., The use of stochastic processes in interpolation and approximation, *Intern. J. Computer Math.*, **8**, Section B:63–76 (1980).

342. Scharstein, D. and R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, *Int'l J. Conf. Computer Vision*, **47**(1/2/3):1–35 (2002).

343. Schmid, C., R. Mohr and C. Bauckhage, Evaluation of interest point detectors, *Int'l J. Computer Vision*, **37**(2):151–172 (2000).

344. Schmidt, J. W., Scattered data interpolation applying regional C^1 splines on refined triangulations, *Math. Mech.*, **80**(1):27–33 (2000).

345. Schmitt, F., Correction of geometrical distortions in MR-images, *Computer Assisted Radiology*, 15–23 (1985).

346. Schnabel, J. A., C. Tanner, A. D. Castellano-Smith, A. Degenhard, M. O. Leach, D. R. Hose, D. L. G. Hill, and D. J. Hawkes, Validation of nonrigid image registration using finite-element methods: Application to breast MR images, *IEEE Trans. Medical Imaging*, **22**(5):238–247 (2003).

347. Schönfeld, H., G. Häusler, and S. Karbacher, Reverse engineering using optical 3-D sensors, *3-D Image Capture and Applications, Proc. SPIE*, 3313, (1998).

348. Shröder, P. and D. Zorin, Subdivision for modeling and animation, *SIGGRAPH Course No. 36 Notes* (1998).

349. Schumaker, L. L., Fitting surfaces to scattered data, *Approximation Theory II*, G. Lorentz, C. Chui, and L. Shumaker (Eds.), Academic Press, 203–268 (1976).

350. Schumaker, L. L., Computing optimal triangulations using simulated annealing, *Computer Aided Geometric Design,* **10**:329–345 (1993).

351. Schutte, H., S. Frydrychowicz, and J. Schroder, Scene matching with translation invariant transforms, *5th Int'l J. Conf. Pattern Recognition*, 1980, 195–198 (1980).

352. Seitz, S. M. and C. R. Dyer, Photorealistic scene reconstruction by voxel coloring, *Int'l J. Computer Vision*, **35**(2):151–173 (1999).

353. Shapiro, L. G., Organization of relational models, *Int'l Conf. Pattern Recognition*, 360–365 (1982).

354. Shepard, D., A two-dimensional interpolation function for irregularly spaced data, *Proc. 23rd Nat'l Conf. ACM*, 517–524 (1968).

355. Shih, F. Y. and Wu, Y-T, Fast Euclidean distance transformation in two scans using a 3×3 neighborhood, *Computer Vision and Image Understanding*, **93**:195–205 (2004).

356. Shiren, Y., L. Li, and G. Peng, Two-dimensional seam-point searching in digital image matching, *Photogrammetric Engineering and Remote Sensing,* **55**:49–53 (1989).

357. Shirman, L. A. and C. H. Sequin, Local surface interpolation with shape parameters between adjoining gregory patches, *Computer Aided Geometric Design*, **7**:375–388 (1990).

358. Shum, H.-Y. and R. Szeliski, Construction of panoramic image mosaics with global and local alignment, *Int'l J. Computer Vision*, **36**(2):101–130 (2000).

359. Silberberg, T. M., D. A. Harwood, and L. S. Davis, Object recognition using oriented model points, *Computer Vision, Graphics, and Image Processing*, **35**:47–71 (1986).

360. Silva, L., O. R. P. Bellon, and K. L. Boyer, Enhanced, robust genetic algorithms for multiview range image registration, *Proc. 4th Int'l Conf. 3-D Digital Imaging and Modeling*, Banff, Alberta, Canada, Oct. 6–10, 268–275 (2003).

361. Singh, M., W. Frei, T. Shibata, and G. C. Huth, A digital technique for accurate change detection in nuclear medical images with application to myocardial perfusion studies using Thallium-201, *IEEE Trans. Nuclear Science*, **26**:565–575 (1979).

362. Smith, P. W. and M. D. Elstrom, Automatic feature correspondence for scene reconstruction from multiple views, *Proc. 2nd Int'l Conf. 3-D Digital Imaging and Modeling*, Ottawa, Canada, Oct. 4–8, 463–472 (1999).

363. Snidaro, L., G. L. Foresti, R. Niu, P. K. Varshney, Sensor fusion for video surveillance, *Proc. 7th Int'l Conf. Information Fusion (Fusion 2004)*, Stockholm, Sweden, June 28 – July 1, 739–746 (2004).

364. Snyder, W., G. Bilbro, A. Logenthiran, and S. Rajala, Optimal thresholding: A new approach, *Pattern Recognition Letters*, **11**:803–810 (1990).

365. Stam, J., On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree, *Computer Aided Geometric Design*, **18**:383–396 (2001).

366. Steiner, D. and M. E. Kirby, Geometric referencing of LANDSAT images by affine transformation and overlaying of map data, *Photogrammetria*, **33**:41–75 (1977).

367. Stockham, G., Image processing in the context of a visual model, *Proc. IEEE,* **60**:828–841, (1972).

368. Stockman, G. Object recognition and localization via pose clustering, *Computer Vision, Graphics, and Image Processing*, **40**:361–387 (1987).

369. Stockman, G., S. Kopstein, and S. Benett, Matching images to models for registration and object detection via clustering, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **4**(3):229–241 (1982).

370. Stockman, G. and J. C. Esteva, Use of geometrical constraints and clustering to determine 3-D object pose, *7th International Conference Pattern Recognition*, **2**:742–744 (1984).

371. Strasters, K. C., J. A. Little, J. Buurman, D. L. G. Hill, and D. J. Hawkes, Anatomical landmark image registration: Validation and comparison, *Lecture*

Notes in Computer Science, J. Troccaz, E. Grimson, and R. Mosges (Eds.), **1205**:161–170 (1997).

372. Studholme, C., D. L. G. Hill, and D. J. Hawkes, Automated 3-D registration of MR and CT images of the head, *Medical Image Analysis*, **1**(2):163–175 (1996).
373. Sugano, N., T. Sasama, Y. Sato, Y. Nakajima, T. Nishii, K. Yoneobu, S. Tamura, and T. Ochi, Accuracy evaluation of surface-based registration methods in a computer navigation system for hip surgery performance through a posterolateral approach, *Computer Aided Surgery*, **6**:195–203 (2001).
374. Svedlow, M., C. D. McGillem, and P. E. Anuta, Experimental examination of similarity measures and preprocessing methods used for image registration, *Symposium Machine Processing of Remotely Sensed Data*, 9–17 (1976).
375. Szeliski, R., Fast surface interpolation using hierarchical basis functions, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**(6):513–528 (1990).
376. Szeliski, R., Video mosaics for virtual environments, *IEEE Computer Graphics and Applications*, **16**(2):22–30 (1996).
377. Szeliski, R. and J. Coughlan, Hierarchical spline based image registration, *Proc. Computer Vision and Pattern Recognition*, 194–201 (1994).
378. Taylor, C. J., P. E. Debevec, and J. Malik, Reconstructing polyhedral models of architectural scenes from photographs, *Proc. European Conf. Computer Vision*, **2**:659–668 (1996).
379. Terzopoulos, D., Multi-level computational processes for visual surface reconstruction, *Computer Vision, Graphics, and Image Processing*, **24**:52–96 (1983).
380. Terzopoulos, D., Regularization of inverse visual problems involving discontinuities, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **8**(4):413–424 (1986).
381. Thévenaz, P. and M. Unser, Optimization of mutual information for multiresolution image registration, *IEEE Trans. Image Processing*, **9**:2083–2099 (2000).
382. Thompson, C., *Depth Perception in Stereo Computer Vision*, Stanford Artificial Intelligence Laboratory, Memo AIM-268, Computer Science Department, Report no. STAN-CS-75-521, 1–15 (1975).
383. Thompson, D. W. and J. L. Mundy, Three-dimensional model matching from an unconstrained viewpoint, in *Proc. IEEE Conf. Robotics and Automation*, 208–220 (1987).
384. Tian, Q. and M. N. Huhns, Algorithms for subpixel registration, *Computer Vision, Graphics and Image Processing*, **35**:220–223 (1986).
385. Toet, A., J. K. I. Jspeert, A. M. Waxman, and M. Aguilar, Fusion of visible and thermal imagery improves situational awareness, *Displays*, **18**:85–95 (1997).
386. Toet, A., N. Schoumans, and J. K. I. Jspeert, Perceptual evaluation of different nighttime imaging modalities, *Proc. 3rd Int'l Conf. Information Fusion (Fusion 2000)*, Paris, France, July 10–13, **1**:TuD3–17 (2000).
387. Tomasi, C. and T. Kanade, Shape and motion from image streams under orthography: A factorization method, *Int'l J. Computer Vision*, **9**:137–154 (1992).

388. Trajkovic, M. and M. Hedley, Fast corner detection, *Image and Vision Computing*, **16**:75–87 (1998).

389. Tubić, D., P. Hébert, and D. Laurendeau, A volumetric approach for interactive 3-D modeling, *Proc. 1st Int'l Sym. 3-D Data Processing Visualization and Transmission*, Padova, Italy, June 19–21, 150–158 (2002).

390. Turk, G. and J. F. O'Brien, Modeling with implicit surfaces that interpolate, *ACM Trans. Graphics*, **21**(4):855–973 (2002).

391. Umeyama, S., Parameterized point pattern matching and its application to recognition of object families, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **15**(2):136–144 (1993).

392. Vajda, I., *Theory of Statistical Inference and Information,* Kluwer (1989).

393. van den Elsen, P. A., E.-J. D. Pol, and M. A. Viergever, Medical image matching: A review with classification, *IEEE Engineering in Medicine and Biology*, 26–39 (1993).

394. Vanderbrug, G. J. and A. Rosenfeld, Two-stage template matching, *IEEE Trans. Computers*, **26**(4):384–393 (1977).

395. van Herk, M. and H. M. Kooy, Automatic three-dimensional correlation of CT-CT, CT-MRI, and CT-SPECT using chamfer matching, *Medical Physics*, **21**(7):1163–1178 (1994).

396. Velho, L. and D. Zorin, 4-8 Subdivision, *Computer Aided Geometric Design*, **18**:397–427 (2001).

397. Venot, A. and V. Leclerc, Automated correlation of patient motion and gray values prior to subtraction in digitized angiography, *IEEE Trans. Medical Imaging*, **8**(3):179–186 (1984).

398. Vinas, F. C., L. Zamorano, R. Buciuc, Q. H. Li, F. Shamsa, Z. Jiang, and F. G. Diaz, Approximation accuracy study of semipermanent fiducial system for frameless stereotaxis, *Computer Aided Surgery*, **3**:257–263 (1997).

399. Viola, P. and W. M. Wells III, Alignment by maximization of mutual information, *Int'l J. Computer Vision*, **24**(2):137–154 (1997).

400. Wahba, G., *Spline Models for Observation Data*, Philadelphia, PA, SIAM Press, 1990.

401. Walker, M. W., L. Shao, and R. V. Volz, Estimating 3-D location parameters using dual number quaternions, *CVGIP: Image Understanding*, **54**:358–367 (1991).

402. Wang, L. and T. Pavlidis, Detection of curved and straight segments from gray scale topography, *CVGIP: Image Understanding*, **58**(3):352–365 (1993).

403. Wang, Y., Z. You, J. Wang, SAR and optical images fusion algorithm based on wavelet transform, *Proc. 7th Int'l Conf. Information Fusion (Fusion 2004)*, Stockholm, Sweden, June 28 – July 1, 644–648 (2004).

404. Weidner, U., Practical aspects of digital orthophoto production, *OEEPE Off. Publ.*, **37**:307–314 (1999).

405. Weiman, C. F. and G. Chaikin, Logarithmic spiral grids for image processing and display, *Computer Graphics and Image Processing*, **11**:197–226 (1979).

406. Weiss, V., L. Andor, G. Renner, and T. Varady, Advanced surface fitting techniques, *Computer Aided Geometric Design*, **19**:19–42 (2002).

407. Wendland, H., Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, *Advances in Computational Mathematics,* **4**:389–396 (1995).

408. Weng, J., P. Cohen, and M. Hernious, Camera calibration for distortion models and accuracy evaluation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **14**(10):965–980 (1992).

409. West, J., J. M. Fitzpatrick *et al.* Comparison and evaluation of retrospective intermodality brain image registration techniques, *J. Computer Assisted Tomography*, **21**(4):554–566 (1997).

410. Weszka, J. S., A survey of threshold selection techniques, *Computer Graphics and Image Processing*, **7**:259–265 (1978).

411. Weszka, J. S. and A. Rosenfeld, Histogram modification for threshold selection, *IEEE Trans. Systems, Man, Cybernetics*, **9**(1):38–52 (1979).

412. Williams, M. L., R. C.Wilson, and E. R.Hancock, Deterministic search for relational graph matching, *Pattern Recognition*, **32**:1255–1271 (1999).

413. Williams, T. D., Depth from camera motion in a real world scene, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **2**(6):511–516 (1980).

414. Wong, J. C. H., C. Studhomes, D. J. Hawkes, and M. N. Maisey, Evaluation of the limits of visual detection of image misregistration in a brain fluorine-18 fluorodeoxyglucose PET-MRI study, *European J. Nuclear Medicine*, **24**:642–650 (1997).

415. Wong, R. Y. and E. L. Hall, Scene matching with invariant moments, *Computer Graphics and Image Processing*, **8**:16–24 (1978).

416. Woods, R. P., Validation of registration accuracy, in *Handbook of Medical Imaging: Processing & Analysis*, Academic Press, 491–497 (2000).

417. Wyngaerd, J. V. and L. V. Gool, Combining texture and shape for automatic crude patch registration, *Proc. 4th Int'l Conf. 3-D Digital Imaging and Modeling*, Banff, Alberta, Canada, Oct. 6–10, 179–186 (2003).

418. Wyszecki, G. and W. S. Stiles, *Color Science,* John Wiley & Sons, New York, NY, 514–581 (1982).

419. Xiang, Z., Color image quantization by minimizing the maximum intercluster distance, *ACM Trans. Graphics,* **16**(3):260–276 (1997).

420. Xie, M., Automatic feature matching in uncalibrated stereo vision through the use of colors, *Robotics and Autonomous Systems*, **21**:355–364 (1997).

421. Xu, L., M. Jackowski, A. Goshtasby, D. Roseman, S. Bines, C. Yu, A. Dhawan, and A. Huntley, Segmentation of skin cancer images, *Image and Vision Computing*, **17**:65–74 (1999).

422. Yakimovsky, Y. and R. Cunningham, A system for extracting three-dimensional measurements from a stereo pair of TV cameras, *Computer Graphics and Image Processing*, **7**:195–210 (1978).

423. Yaou, M-H and W-T Chang, Fast surface interpolation using multiresolution wavelet transform, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **16**(7):673–688 (1994).

424. Yaroslavsky, L. P., B. Fishbain, A. Shteinman, S. Gepshtein, Processing and fusion of thermal and video sequences for terrestrial long range observation systems, *Proc. 7th Int'l Conf. Information Fusion (Fusion 2004)*, Stockholm, Sweden, June 28 – July 1, 848–855 (2004).

425. Yau, W. Y. and H. Wang, Fast relative depth computation for an active stereo vision system, *Real Time Imaging*, **5**:189–202 (1999).

426. Yip, R. K. K., A multi-level dynamic programming method for line segment matching in axial motion stereo, *Pattern Recognition*, **31**(11):1653–1668 (1998).

427. Zahn, C. T., Jr., An algorithm for noisy template matching, *IFIP Congress*, 698–701, Aug. 5–10, 1974.

428. Zahn, C. T., Jr. and R. Z. Roskies, Fourier descriptions for plane closed curves, *IEEE Transactions on Computers*, **21**(3):269–281 (1972).

429. Zheng, Z., H. Wang, and E. K. Teoh, Analysis of gray level corner detection, *Pattern Recognition Letters*, **20**:149–162 (1999).

430. Zhou, Y., H. Xue, and M. Wan, Inverse image alignment method for image mosaicing and video stabilization in fundus indocyanine green angiography under confocal scanning laser ophthalmoscope, *Computerized Medical Imaging and Graphics*, **27**:513–523 (2003).

431. Zitnick, C. L. and T. Kanade, A cooperative algorithm for stereo matching and occlusion detection, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **22**(7):675–684 (2000).

432. Zitova, B. and J. Flusser, Image registration methods: A survey, *Image and Vision Computing*, **21**: 977–1000 (2003).

433. Zorin, D. and P. Schroder, A unified framework for primal/dual quadrilateral subdivision schemes, *Computer Aided Geometric Design*, **18**:429–454 (2001).

Index

2-D and 3-D Image Registration, by A. Ardeshir Goshtasby
ISBN 0-471-64954-6

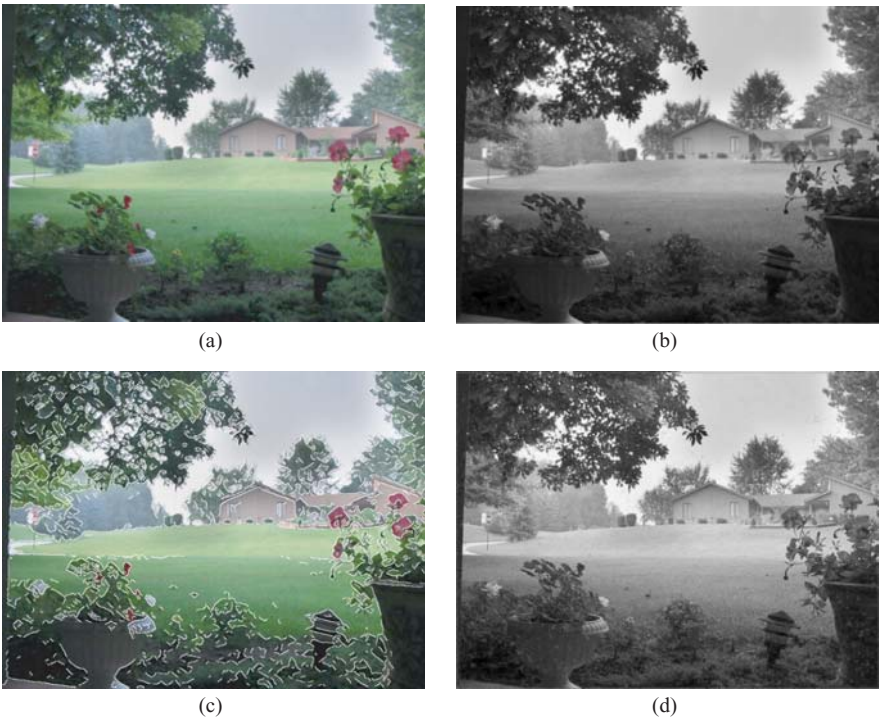


Fig. 2.14 (a) A color image of an outdoor scene. (b) The luminance component of the image. (c) Edges of the color image. (d) Edges of the luminance image.

Fig. 8.1 (a)–(e) Images representing different exposures of an office room. (f) The image obtained by composing the twelve blocks cut out of the five images. (g) The image produced by centering the blending functions at the selected blocks in the images, multiplying the blending functions by image colors, and adding the weighted colors together. (h) The most informative image constructed from images (a)–(e). These images are of size 640×480 pixels. Optimal values found for d and σ were 128 and 96 pixels, respectively.

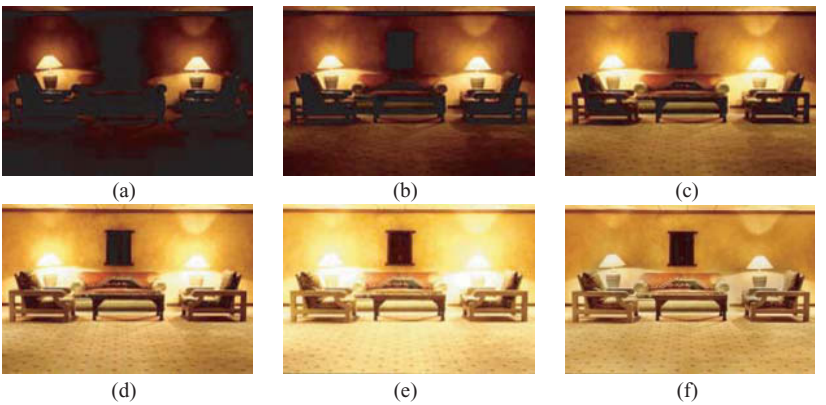


Fig. 8.2 Images representing different exposures of a waiting-room scene. Entropies of the images are 2.82, 3.36, 4.21, 4.78, and 4.40, respectively. (f) The image obtained by fusing the five images. Entropy of the fused image is 5.29. These images are of size 343×231 pixels. Optimal parameters found are $d = 32$ pixels and $\sigma = 32$ pixels.

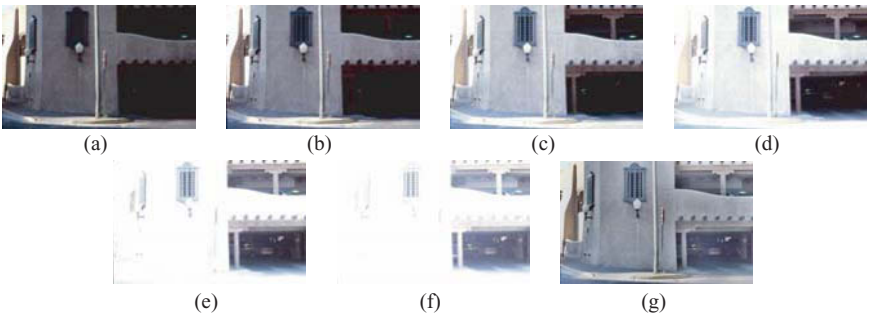


Fig. 8.3 (a)–(f) Images of a garage scene obtained at different exposures. (g) The image obtained by fusing the six images. Entropies of images (a) through (g) are 3.02, 3.29, 5.04, 4.70, 4.03, 3.90, and 5.27, respectively. These images are of size 348×222 pixels. The optimal $d = 8$ pixels and the optimal $\sigma = 16$ pixels.

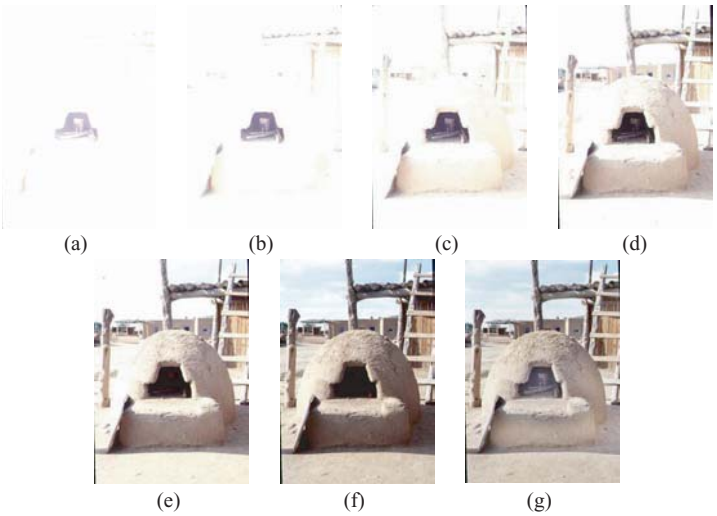


Fig. 8.4 (a)–(f) Images of an igloo scene obtained at multiple exposures. (g) The image obtained by blending the six images. Entropies of these images are 2.92, 3.02, 3.97, 4.39, 4.86, 5.18, and 5.38, respectively. The images are of size 236×341 pixels. The optimal values of d and σ are both 16 pixels.

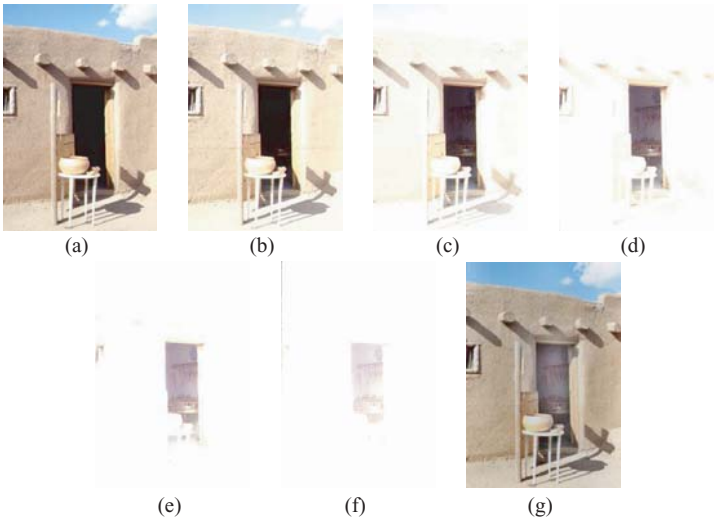


Fig. 8.5 (a)–(f) Images of a door to a dark room obtained at different exposures. (g) Image obtained by fusing the six images. Entropies of the images are 5.09, 4.81, 3.86, 3.17, 3.04, 2.81, and 5.23, respectively. These images are of size 231 × 338 pixels. Optimal $d = 8$ pixels and optimal $\sigma = 16$ pixels.

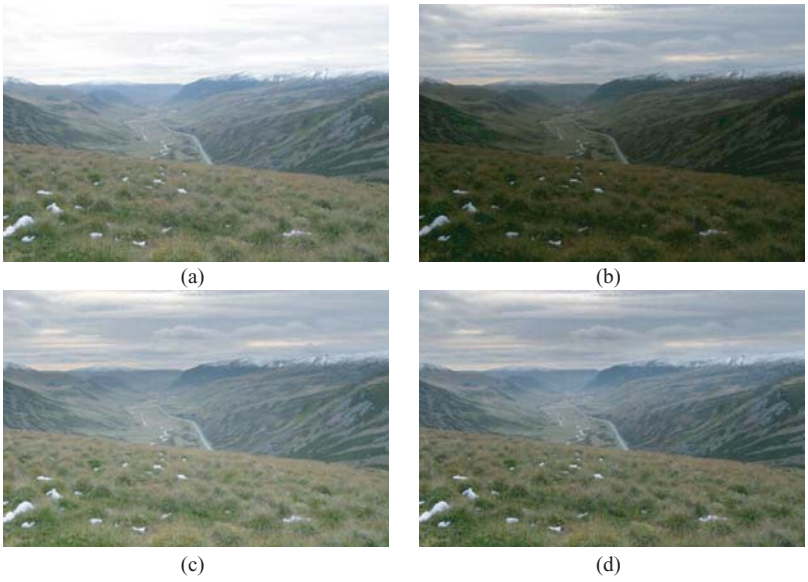


Fig. 8.6 (a), (b) Two images of the Scottish Highlands representing different exposure levels. Entropies of these images are 4.86 and 4.34. (c) The image created by manual photographic techniques. (d) The image constructed by fusing (a) and (b). Entropies of (c) and (d) are 5.22 and 5.20, respectively. These images are of size 1536 × 1024 pixels. Optimal values of d and σ are both 64 pixels.