

CP5095 COMPUTER VISION

UNIT I IMAGE PROCESSING FOUNDATIONS

Review of image processing techniques – classical filtering operations – thresholding techniques – edge detection techniques – corner and interest point detection – mathematical morphology – texture.

UNIT II SHAPES AND REGIONS

Binary shape analysis – connectedness – object labeling and counting – size filtering – distance functions – skeletons and thinning – deformable shape analysis – boundary tracking procedures – active contours – shape models and shape recognition – centroidal profiles – handling occlusion – boundary length measures – boundary descriptors – chain codes – Fourier descriptors – region descriptors – moments.

UNIT III HOUGH TRANSFORM

Line detection – Hough Transform (HT) for line detection – foot-of-normal method – line localization – line fitting – RANSAC for straight line detection – HT based circular object detection – accurate center location – speed problem – ellipse detection – Case study: Human Iris location – hole detection – generalized Hough Transform (GHT) – spatial

matched filtering – GHT for ellipse detection – object location – GHT for feature collation.

UNIT IV 3D VISION AND MOTION

Methods for 3D vision – projection schemes – shape from shading – photometric stereo – shape from texture – shape from focus – active range finding – surface representations – point-based representation – volumetric representations – 3D object recognition – 3D reconstruction – introduction to motion – triangulation – bundle adjustment – translational alignment – parametric motion – spline-based motion – optical flow – layered motion.

UNIT V APPLICATIONS

Application: Photo album – Face detection – Face recognition – Eigen faces – Active appearance and 3D shape models of faces Application: Surveillance – foreground-background separation – particle filters – Chamfer matching, tracking, and occlusion – combining views from multiple cameras – human gait analysis Application: In-vehicle vision system: locating roadway – road markings – identifying road signs – locating pedestrians.

Images and Imaging Operations

2

*pi'x'ēllātēd, a. picture broken into a regular tiling
pi'x'īlātēd, a. pixie-like, crazy, deranged*

Images are at the core of vision, and there are many ways—from simple to sophisticated—for processing and analyzing them. This chapter concentrates on simple algorithms, which nevertheless need to be treated carefully as there are important subtleties to be learnt. Above all, the chapter aims to show that quite a lot can be achieved with such algorithms, which can readily be programmed and tested by the reader.

Look out for:

- the different types of image—binary, grayscale, and color.
- a compact notation for presenting image processing operations.
- basic pixel operations—clearing, copying, inverting, thresholding.
- basic window operations—shifting, shrinking, expanding.
- grayscale brightening and contrast-stretching operations.
- binary edge location and noise removal operations.
- multi-image and convolution operations.
- the distinction between sequential and parallel operations, and complications that can arise in the sequential case.
- problems that arise around the edge of the image.

Although elementary, this chapter actually provides basic methodology for the whole of Part 1 and much of Part 2 of the book, and its importance should not be underestimated: nor should the subtleties be ignored. Full understanding at this stage will save many complications later, while programming more sophisticated algorithms.

2.1 INTRODUCTION

This chapter is concerned with images and simple image processing operations. It is intended to lead on to more advanced image analysis operations that are of use for machine vision in an industrial environment. Perhaps the main purpose of the chapter is to introduce the reader to some basic techniques and notations that will be of use throughout the book. However, the image processing algorithms introduced here are of value in their own right in disciplines ranging from remote sensing to medicine, and from forensic to military and scientific applications.

This chapter deals with images that have already been obtained from suitable sensors: sensors are covered in Chapter 25. Typical of such images is that shown in Fig. 2.1(a). This is a gray-tone image that at first sight appears to be a normal “black and white” photograph. However, closer inspection shows that it is composed of a large number of individual picture cells, or “pixels.” In fact, the image is a 128×128 array of pixels. To get a better feel for the limitations of such a digitized image, Fig. 2.1(b) shows a 42×42 section that has been subjected to a three-fold magnification so that the pixels can be examined individually.

It is not easy to see that these gray-tone images are digitized into a gray scale containing just 64 gray levels. To some extent high spatial resolution compensates for lack of grayscale resolution, and as a result it is difficult to see the difference between an individual shade of gray and the shade it would have had in an ideal picture. In addition, when we look at the magnified section of image in Fig. 2.1(b), it is difficult to understand the significance of the individual pixel intensities—the whole is becoming lost in a mass of small parts. Early TV cameras typically gave a grayscale resolution that was accurate only to about 1 part in 50, corresponding to about 6 bits of useful information per pixel. Modern solid-state cameras commonly give less noise and may allow 8 or even 9 bits of information per pixel. However, there are many occasions when it is not worthwhile to aim for such high grayscale resolutions, particularly when the result will not be visible to the human eye, or when there is an enormous amount of other data that a robot can use to locate objects within the field of view. Note that if the human eye can see an object in a digitized image of particular spatial and grayscale resolution, it is in principle possible to devise a computer algorithm to do the same thing.

Nevertheless, there is a range of applications for which it is valuable to retain good grayscale resolution, so that highly accurate measurements can be made from a digital image. This is the case in many robotic applications, where high-accuracy checking of components is critical. More will be said about this later. In addition, it will be seen in Part 2 that certain techniques for locating components efficiently require local edge orientation to be estimated to better than 1° , and this can be achieved only if at least 6 bits of grayscale information are available per pixel.

(a)

(b)

FIGURE 2.1

Typical grayscale image: (a) grayscale image digitized into a 128×128 array of pixels; (b) section of image shown in (a) subjected to three-fold linear magnification: the individual pixels are now clearly visible.

2.1.1 Gray Scale Versus Color

Returning now to the image of Fig. 2.1(a), we might reasonably ask whether it would be better to replace the gray scale with color, using an RGB color camera

and three digitizers for the three main colors. There are two aspects of color that are important for the present discussion. One is the *intrinsic value* of color in machine vision and the other is the *additional storage and processing penalty* it might bring. It is tempting to say that the latter aspect is of no great importance given the cheapness of modern computers which have both high storage and high speed. On the other hand, high-resolution images can arrive from a collection of CCTV cameras at huge data rates, and it will be many years before it will be possible to analyze *all* the data arriving from such sources as they come in. Hence, if color adds substantially to the storage and processing load, this will need to be justified.

Against this, the *potential* of color for helping with many aspects of inspection, surveillance, control, and a wide variety of other applications including medicine (color playing a crucial role in images taken during surgery) is enormous. This is illustrated with regard to robot navigation and driving in [Figs. 2.2 and 2.3](#); and

FIGURE 2.2

Value of color for segmentation and recognition. In natural outdoor scenes such as this, color helps with segmentation and with recognition. While it may have been important to the early human when discerning sources of food in the wild, robot drones may benefit by using color to aid navigation (see also color Plate 2).

for food inspection in [Figs. 2.4 and 2.5](#); for color filtering see Figs. 3.12 and 3.13. Note that some of these images almost have color for color's sake (especially in [Figs. 2.4 and 2.5](#)), although none of them are artificially generated. In others the color is more subdued ([Fig. 2.3](#)), and in [Fig. 2.5](#) (excluding the tomatoes), it is quite subtle. The point to be made here is that for color to be useful it need not be garish, but can be subtle as long as it brings the right sort of information to bear on the task in hand. Suffice it to say that in some of the simpler inspection applications, where mechanical components are scrutinized on a conveyor or workbench, it is quite likely to be the shape that is in question rather than the color of the object or its parts. On the other hand, if an automatic fruit picker is to be devised, it will probably be more crucial to check color than specific shape. We leave it to the reader to imagine when and where color is particularly useful, or merely an unnecessary luxury.

FIGURE 2.3

Value of color in the built environment. Color plays an important role for the human in managing the built environment. In a vehicle, a plethora of bright lights, road signs, and markings (such as yellow lines) are coded to help the driver: they may likewise help a robot to drive more safely by the provision of crucial information (see also color Plate 3).

FIGURE 2.4

Value of color for food inspection. Much food is brightly colored, as with this Japanese meal. While this may be attractive to the human, it could also help the robot to check quickly for foreign bodies or toxic substances (see also color Plate 4).

Next, it is useful to consider the processing aspect of color. In many cases, good color discrimination is required to separate and segment two types of object from each other. Typically this will mean not using one or other specific color channel,¹ but subtracting two, or combining three in such a way as to foster discrimination. In the worst case of combining three color channels by simple arithmetic processing in which each pixel is treated identically, the processing load will be very light. In contrast, the amount of processing required to *determine* the optimal means of combining the data from the color channels, and to carry out different operations dynamically on different parts of the image, may be far from negligible, and some care will be needed in the analysis. These problems arise because color signals are inhomogeneous: this contrasts with the situation for grayscale images, where the bits representing the gray scale are all of the same type and take the form of a number representing the pixel intensity: they can thus be processed as a single entity on a digital computer.

¹Here we use the term “channel” not just to refer to the red, green, or blue channel, but any derived channel obtained by combining the colors into a single-color dimension.

FIGURE 2.5

Subtle shades of color in food inspection. While much food is brightly colored, as for the tomatoes in this picture, green salad leaves show much more subtle combinations of color and may indeed provide the only reliable means of identification. This could be important for inspection both of the raw product and its state when it reaches the warehouse or the supermarket.

2.2 IMAGE PROCESSING OPERATIONS

In what follows, the images of Figs. 2.1(a) and 2.7(a) are considered in some detail, examining some of the many image processing operations that can be performed on them. The resolution of these images reveals a considerable amount of detail and at the same time shows how it relates to the more “meaningful” global information. This should help to make it clear how simple imaging operations contribute to image interpretation.

When performing image processing operations, we start with an image in one storage area and generate a new processed image in another storage area. In practice, these storage areas may either be in a special hardware unit called a frame store that is interfaced to the computer or they may be in the main memory of the computer or on one of its disks. In the past a special frame store was required to store images, since each image contains a good fraction of a megabyte of information and this amount of space was not available for normal users in the computer main memory. Nowadays this is less of a problem, but for image

acquisition and display a frame store is still required. However, we shall not worry about such details here. Instead it will be assumed that all images are inherently visible and that they are stored in various image “spaces” P, Q, R, etc. Thus, we might start with an image in space P and copy it to space Q, for example.

2.2.1 Some Basic Operations on Grayscale Images

Perhaps the simplest of imaging operations is that of clearing an image or setting the contents of a given image space to a constant level. We need some way of arranging this, and accordingly the following C++ routine may be written for implementing it:²

```
for (j = 0; j <= 127; j++)
    for (i = 0; i <= 127; i++)
        P[j][i] = alpha;
```

(2.1)

In this routine the local pixel intensity value is expressed as $P[j][i]$, since P-space is taken to be a two-dimensional array of intensity values (Table 2.1). In what follows, it will be advantageous to rewrite such routines in the more succinct form:

```
for all pixels in image do {P0 = alpha;}
```

(2.2)

as this will aid understanding by removing irrelevant programming detail. The reason for calling the pixel intensity $P0$ will become clear later.

Another simple imaging operation is to copy an image from one space to another. This is achieved, without changing the contents of the original space P, by the routine:

```
for all pixels in image do {Q0 = P0;}
```

(2.3)

A more interesting operation is that of inverting the image, as in the process of converting a photographic negative to a positive. This process is represented as follows:

```
for all pixels in image do {Q0 = 255 - P0;}
```

(2.4)

In this case, it is assumed that pixel intensity values lie within the range 0–255, as is commonly true for frame stores that represent each pixel as one byte of information. Note that such intensity values are commonly unsigned and this is assumed generally in what follows.

There are many operations of these types. Some other simple operations are those that shift the image left, right, up, down, or diagonally. They are easy to implement if the new local intensity is made identical to that at a neighboring location in the original image. It is evident how this would be expressed in the double suffix notation used in the original C++ routine. In the new shortened

²Readers who are unfamiliar with C++ or Java, which is similar at this level of programming, should refer to Stroustrup (1991) and Schildt (1995).

Table 2.1 C++ Notation

Notation	Meaning
<code>++</code>	increment the preceding variable.
<code>[]</code>	add array index after a variable.
<code>[][]</code>	add two array indices after a variable; the last is the faster running index.
<code>(int)</code>	changes the following variable to integer type.
<code>(float)</code>	changes the following variable to floating point.
<code>{ }</code>	encloses a sequence of instructions.
<code>if () { };</code>	basic conditional statement: <code>()</code> encloses the condition; <code>{ }</code> encloses the instructions to be executed.
<code>if () { }; else if () { }; ... ; else { };</code>	the most general type of conditional statement.
<code>while () { }</code>	common type of iterated loop.
<code>do { } while ();</code>	another common type of iterated loop.
<code>do { } until ();</code>	“until” means the same as “while not.” This is often a convenient notation, although it is not strict C++ .
<code>for (; ;) { };</code>	here the conditional statement <code>()</code> has three arguments separated by semicolons: they are respectively the initial condition; the terminating condition; and the incrementation operation.
<code>=</code>	forces equality (literally: “takes the value”).
<code>==</code>	tests for equality in a conditional expression.
<code><=</code>	\leq
<code>>=</code>	\geq
<code>!=</code>	\neq
<code>!</code>	logical not.
<code>&&</code>	logical and.
<code> </code>	logical or.
<code>//</code>	indicates that the remainder of the line is a comment.
<code>/* ... */</code>	brackets enclosing a comment.
<code>A0 ... A8</code> <code>B0 ... B8</code> <code>C0 ... C8</code>	bit image variables in 3×3 window. ^a
<code>P0 ... P8</code> <code>Q0 ... Q8</code> <code>R0 ... R8</code>	byte image variables in 3×3 window. ^a
<code>P[0], ...</code>	equivalent to <code>P0, ...</code>

^aThese predefined variables denote special syntax not available in C++, but useful for simplifying the image processing algorithms presented in Chapters 2, 3, 4, 7 and 9.

Note: The purpose of this table is to show what is meant by the various C++ commands and instructions used in this book. It is not intended to be comprehensive. The aim is merely to be helpful to the reader. In general, only notation that differs between C++ and other commonly used languages such as Pascal is included, in order to eliminate possible ambiguity or confusion.

26 CHAPTER 2 Images and Imaging Operations

notation, it is necessary to name neighboring pixels in some convenient way, and we here employ the following simple scheme:

P4	P3	P2
P5	P0	P1
P6	P7	P8

with a similar scheme for other image spaces. With this notation, it is easy to express a left shift of an image as follows:

```
for all pixels in image do {Q0 = P1;}
```

 (2.5)

Similarly, a shift down to the bottom right is expressed as:

```
for all pixels in image do {Q0 = P4;}
```

 (2.6)

It will now be clear why P0 and Q0 were chosen for the basic notation of pixel intensity: the “0” denotes the central pixel in the “neighborhood” or “window,” and corresponds to zero shift when copying from one space to another. However, the type of window operation presented above is much more powerful than single-pixel operations, and we shall see many examples of it in what follows. Meanwhile, note that it can give rise to difficulties around the boundaries of the image: we shall return to this point in [Section 2.4](#).

There is a whole range of possible operations associated with modifying images in such a way as to match them to the requirements of a human viewer. For example, adding a constant intensity makes the image brighter:

```
for all pixels in image do {Q0 = P0 + beta;}
```

 (2.7)

and the image can be made darker in the same way. A more interesting operation is to stretch the contrast of a dull image:

```
for all pixels in image do {Q0 = P0 * gamma + beta;}
```

 (2.8)

where $\text{gamma} > 1$. In practice (as for [Fig. 2.6](#)), it is necessary to ensure that intensities do not result that are outside the normal range, e.g., by using an operation of the form:

```
for all pixels in image do {
    QQ = P0 * gamma + beta;
    if (QQ < 0) Q0 = 0;
    else if (QQ > 255) Q0 = 255;
    else Q0 = QQ;
}
```

 (2.9)

Most practical situations demand more sophisticated transfer functions—either nonlinear or piecewise linear—but such complexities are ignored here.

FIGURE 2.6

Contrast stretching: effect of increasing the contrast in the image of Fig. 2.1(a) by a factor of two and adjusting the mean intensity level appropriately. The interior of the jug can now be seen more easily. Note, however, that there is no additional information in the new image.

A further simple operation that is often applied to grayscale images is that of thresholding to convert to a binary image. This topic is covered in more detail later, since it is widely used to detect objects in images. However, our purpose here is to look on it as another basic imaging operation. It can be implemented using the routine:³

```
for all pixels in image do {  
    if ( $P_0 > \text{thresh}$ )  $A_0 = 1$ ; else  $A_0 = 0$ ;  
}
```

(2.10)

If, as very often happens, objects appear as dark objects on a light background, it is easier to visualize the subsequent binary processing operations by inverting the thresholded image using a routine such as:

```
for all pixels in image do { $A_0 = 1 - A_0$ };
```

(2.11)

³The first few letters of the alphabet (A, B, C, ...) are used consistently to denote binary image spaces, and later letters of the alphabet (P, Q, R, ...) to denote grayscale images (Table 2.1). In software, these variables are assumed to be predeclared, and in hardware (e.g., frame store) terms they are taken to refer to dedicated memory spaces containing only the necessary 1 or 8 bits per pixel. The intricacies of data transfer between variables of different types are important considerations that are not addressed in detail here. It is sufficient to assume that both $A_0 = P_0$ and $P_0 = A_0$ correspond to a single-bit transfer, except that in the latter case the top 7 bits are assigned the value 0.

(a) (b)

FIGURE 2.7

Thresholding of grayscale images: (a) 128×128 pixel grayscale image of a collection of parts; (b) effect of thresholding the image.

However, it would be more usual to combine the two operations into a single routine of the form:

```
for all pixels in image do {
    if( $P_0 > \text{thresh}$ )  $A_0 = 0$ ; else  $A_0 = 1$ ;
}
```

(2.12)

To display the resulting image in a form as close as possible to the original, it can be reinverted and given the full range of intensity values (intensity values 0 and 1 being scarcely visible):

```
for all pixels in image do { $R_0 = 255 * (1 - A_0)$ };
```

(2.13)

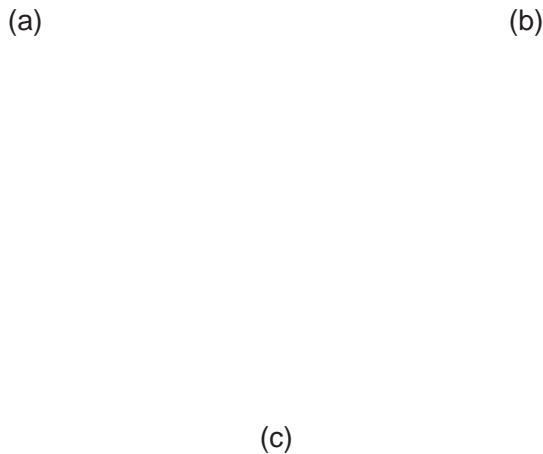
[Figure 2.7](#) shows the effect of these two operations.

2.2.2 Basic Operations on Binary Images

Once the image has been thresholded, a wide range of binary imaging operations become possible. Only a few such operations are covered here, with the aim of being instructive rather than comprehensive. With this in mind, a routine may be written for shrinking dark-thresholded objects ([Fig. 2.8\(a\)](#)) that are here represented by a set of 1's in a background of 0's:

```
for all pixels in image do {
     $\sigma = A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8$ ;
    if ( $A_0 == 0$ )  $B_0 = 0$ ;
    else if ( $\sigma < 8$ )  $B_0 = 0$ ;
    else  $B_0 = 1$ ;
}
```

(2.14)

**FIGURE 2.8**

Simple operations applied to binary images: (a) effect of shrinking the dark-thresholded objects appearing in Fig. 2.7(b); (b) effect of expanding these dark objects; (c) result of applying an edge location routine. Note that the shrink, expand, and edge routines are applied to the *dark* objects: this implies that the intensities are initially inverted as part of the thresholding operation and then reinverted as part of the display operation (see text).

In fact, the logic of this routine can be simplified to give the following more compact version:

```
for all pixels in image do {  
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;  
    if (sigma < 8) B0 = 0; else B0 = A0;  
}
```

(2.15)

Note that the process of shrinking dark objects also expands light objects, including the light background. It also expands holes in dark objects. The opposite

30 CHAPTER 2 Images and Imaging Operations

process, that of expanding dark objects (or shrinking light ones), is achieved ([Fig. 2.8\(b\)](#)) with the routine:⁴

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    if(sigma > 0) B0 = 1; else B0 = A0;
}
```

(2.16)

Each of these routines employs the same technique for interrogating neighboring pixels in the original image: as will be apparent on numerous occasions in this book, the sigma value is a useful and powerful descriptor for 3×3 pixel neighborhoods. Thus, “if ($\text{sigma} > 0$)” can be taken to mean “if next to a dark object” and the consequence can be read as “then expand it.” Similarly, “if ($\text{sigma} < 8$)” can be taken to mean “if next to a light object” or “if next to light background,” and the consequence can be read as “then expand the light background into the dark object.”

The process of finding the edge of a binary object has several possible interpretations. Clearly, it can be assumed that an edge point has a sigma value in the range 1–7 inclusive. However, it may be defined as being within the object, within the background or in either position. Taking the definition that the edge of an object has to lie within the object ([Fig. 2.8\(c\)](#)), the following edge-finding routine for binary images results:

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    if(sigma == 8) B0 = 0; else B0 = A0;
}
```

(2.17)

This strategy amounts to canceling out object pixels that are not on the edge. For this and a number of other algorithms (including the shrink and expand algorithms already encountered), a thorough analysis of exactly which pixels should be set to 1 and 0 (or which should be retained and which eliminated) involves drawing up tables of the form:

sigma			
0–7		8	
A0	0	0	0
	1	1	0

This reflects the fact that algorithm specification includes a recognition phase and an action phase, i.e., it is necessary first to locate situations within an image where (for example) edges are to be marked or noise eliminated, and then action must be taken to implement the change.

⁴The processes of shrinking and expanding are widely known by the respective terms “erosion” and “dilation.” (See also Chapter 7.)

Another function that can usefully be performed on binary images is the removal of “salt and pepper” noise, i.e., noise which appears as a light spot on a dark background or a dark spot on a light background. The first problem to be solved is that of recognizing such noise spots; the second is the simpler one of correcting the intensity value. For the first of these tasks the sigma value is again useful. To remove salt noise (which has binary value 0 in our convention), we arrive at the following routine:

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    if(sigma == 8) B0 = 1; else B0 = A0;
}
```

(2.18)

which can be read as leaving the pixel intensity unchanged unless it is proven to be a salt noise spot. The corresponding routine for removing pepper noise (binary value 1) is:

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    if(sigma == 0) B0 = 0; else B0 = A0;
}
```

(2.19)

Combining these two routines into one operation ([Fig. 2.9\(a\)](#)) gives:

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    if(sigma == 0) B0 = 0;
    else if(sigma == 8) B0 = 1;
    else B0 = A0;
}
```

(2.20)

(a)

(b)

FIGURE 2.9

Simple binary noise removal operations: (a) result of applying a “salt and pepper” noise removal operation to the thresholded image in [Fig. 2.7\(b\)](#). (b) Result of applying a less stringent noise removal routine: this is effective in cutting down the jagged spurs that appear on some of the objects.

The routine can be made less stringent in its specification of noise pixels, so that it removes spurs on objects and background: this is achieved ([Fig. 2.9\(b\)](#)) by a variant such as:

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    if(sigma < 2) B0 = 0;
    else if(sigma > 6) B0 = 1;
    else B0 = A0;
}
```

(2.21)

As before, if there is any doubt about the algorithm, its specification should be set up rigorously—as in the following table:

		sigma		
		0 or 1	2–6	7 or 8
A0	0	0	0	1
	1	0	1	1

There are many other simple operations that can usefully be applied to binary images and some of them are dealt with in Chapter 9.

2.3 CONVOLUTIONS AND POINT SPREAD FUNCTIONS

Convolution is a powerful and widely used technique in image processing and other areas of science. It appears in many applications throughout this book and it is therefore useful to introduce it at an early stage. We start by defining the convolution of two functions $f(x)$ and $g(x)$ as the integral:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(u)g(x-u) du \quad (2.22)$$

The action of this integral is normally described as the result of applying a point spread function $g(x)$ to all points of a function $f(x)$ and accumulating the contributions at every point. It is significant that if the point spread function (PSF) is very narrow,⁵ then the convolution is identical to the original function $f(x)$. This makes it natural to think of the function $f(x)$ as having been spread out under the influence of $g(x)$. This argument may give the impression that convolution necessarily blurs the original function but this is not always so if, for example, the PSF has a distribution of positive and negative values.

When convolution is applied to digital images, the above formulation changes in two ways: (i) a double integral must be used in respect of the two dimensions

⁵Formally, it can be a delta function, which is infinite at one point and zero elsewhere while having an integral of unity.

and (ii) integration must be changed into discrete summation. The new form of the convolution is:

$$F(x, y) = f(x, y) * g(x, y) = \sum_i \sum_j f(i, j)g(x - i, y - j) \quad (2.23)$$

where g is now referred to as a spatial convolution mask. The fact that the mask has to be inverted before it is applied is inconvenient for visualizing the process of convolution—particularly when matching operations are involved, e.g., for corner location (see Chapter 6). In this book, we therefore present only pre-inverted masks of the form:

$$h(x, y) = g(-x, -y) \quad (2.24)$$

Convolution can then be calculated using the more intuitive formula:

$$F(x, y) = \sum_i \sum_j f(x + i, y + j)h(i, j) \quad (2.25)$$

which involves multiplying corresponding values in the modified mask and the neighborhood under consideration. Re-expressing this result for a 3×3 neighborhood and writing the mask coefficients in the form:

$$\begin{bmatrix} h4 & h3 & h2 \\ h5 & h0 & h1 \\ h6 & h7 & h8 \end{bmatrix}$$

the algorithm can be obtained in terms of our earlier notation:

```
for all pixels in image do {
    Q0 = P0 * h0 + P1 * h1 + P2 * h2 + P3 * h3 + P4 * h4
        + P5 * h5 + P6 * h6 + P7 * h7 + P8 * h8;
}
```

(2.26)

We are now in a position to apply convolution to a real situation. At this stage we attempt to suppress noise by averaging over nearby pixels. A simple way of achieving this is to use the convolution mask:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

where the number in front of the mask weighs all the coefficients in the mask and is inserted to ensure that applying the convolution does not alter the mean intensity in the image. As hinted above, this particular convolution has the effect of blurring the image as well as reducing the noise level (Fig. 2.10). More will be said about this in the next chapter.

FIGURE 2.10

Noise suppression by neighborhood averaging achieved by convolving the original image of Fig. 2.1(a) with a uniform mask within a 3×3 neighborhood. Note that noise is suppressed only at the expense of introducing significant blurring.

The above discussion makes it clear that convolutions are linear operators. In fact, they are the most general spatially invariant linear operators that can be applied to a signal such as an image. Note that linearity is often of interest in that it permits mathematical analysis to be performed that would otherwise be intractable.

2.4 SEQUENTIAL VERSUS PARALLEL OPERATIONS

It will be noticed that most of the operations defined so far have started with an image in one space and finished with an image in a different space. Unfortunately, many of the operations will not work satisfactorily if we do not use separate input and output spaces in this way. This is because they are inherently “parallel processing” routines. This term is used as these are the types of process that would be performed by a parallel computer possessing a number of processing elements equal to the number of pixels in the image, so that all the pixels are processed simultaneously. If a serial computer is to *simulate* the operation of a parallel computer, then it must have separate input and output image spaces and rigorously work in such a way that it uses the original image values to compute the output

pixel values. This means that an operation such as the following cannot be an ideal parallel process:

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    if (sigma < 8) A0 = 0; else A0 = A0;
}
```

(2.27)

This is so because, when the operation is half completed, the output pixel intensity will depend not only on some of the unprocessed pixel values but also on some that have already been processed. For example, if the computer makes a normal (forward) TV raster scan through the image, the situation at a general point in the scan will be

✓	✓	✓
✓	✗	✗
✗	✗	✗

where the ticked pixels have already been processed and the others have not. As a result, the above operation will shrink all objects to nothing.

A much simpler illustration of this is obtained by attempting to shift an image to the right using the following routine:

```
for all pixels in image do {P0 = P5;}
```

(2.28)

In fact, all this achieves is to fill up the image with values corresponding to those off its left edge,⁶ whatever they are assumed to be. Thus, we have shown that the shifting process is inherently parallel.

It will be seen in Chapter 9 that there are some processes that are inherently sequential—i.e., the processed pixel has to be returned immediately to the *original* image space. Meanwhile, note that not all of the routines described so far need to be restricted rigorously to parallel processing. In particular, all single-pixel routines (essentially those that only refer to the single pixel in a 1×1 neighborhood) can validly be performed as if they were sequential in nature. Such routines include the following intensity adjustment and thresholding operations:

```
for all pixels in image do {P0 = P0 * gamma + beta;}
```

(2.29)

```
for all pixels in image do {if (P0 > thresh) P0 = 1; else P0 = 0;}
```

(2.30)

These remarks are intended to act as a warning. In general, it is safest to design algorithms that are exclusively parallel processes unless there is a definite need to make them sequential. It will be seen later how this need can arise.

⁶Note that when the computer is performing a 3×3 (or larger) window operation, it has to assume some value for off-image pixel intensities: usually whatever value is selected will be inaccurate, and so the final processed image will contain a border that is also inaccurate. This will be so whether the off-image pixel addresses are trapped in software or in specially designed circuitry in the frame store.

2.5 CONCLUDING REMARKS

This chapter has introduced a compact notation for representing imaging operations and has demonstrated some basic parallel processing routines. The following chapter extends this work to see how noise suppression can be achieved in grayscale images. This leads on to more advanced image analysis work that is directly relevant to machine vision applications. In particular, Chapter 4 studies in more detail the thresholding of grayscale images, building on the work of [Section 2.2.1](#), while Chapter 9 studies object shape analysis in binary images.

Pixel–pixel operations can be used to make radical changes in digital images. However, this chapter has shown that window–pixel operations are far more powerful, and capable of performing all manner of size- and shape-changing operations, as well as eliminating noise. But caveat emptor—sequential operations can have some odd effects if adventitiously applied.

2.6 BIBLIOGRAPHICAL AND HISTORICAL NOTES

Since the aim of this chapter is not to cover the most recent material but to provide a succinct overview of basic techniques, it will not be surprising that most of the topics discussed were developed well over 20 years ago and have been used by a large number of workers in many areas. For example, thresholding of grayscale images was first reported at least as long ago as 1960, while shrinking and expanding of binary picture objects date from a similar period. Discussion of the origins of other techniques is curtailed: for further detail the reader is referred to the texts by, e.g., Gonzalez and Woods (2008), Nixon and Aguado (2008), Petrou and Petrou (2010), and Sonka et al. (2007). We also refer to two texts that cover programming aspects of image processing in some depth: Parker (1994), which covers C programming, and Whelan and Molloy (2001), which covers Java programming. More specialized texts will be referred to in the following chapters.

2.7 PROBLEMS

- Derive an algorithm for finding the edges of binary picture objects by applying a shrink operation and combining the result with the original image. Is the result the same as that obtained using the edge-finding routine (Eq. 2.17)? Prove your statement rigorously by drawing up suitable algorithm tables as in [Section 2.2.2](#).

2. In a certain frame store, each off-image pixel can be taken to have either the value 0 or the intensity of the nearest image pixel. Which of the two will give the more meaningful results for (a) shrinking, (b) expanding, and (c) a blurring convolution?
3. Suppose the noise elimination routines of equations [\(2.20\)](#) and [\(2.21\)](#) were reimplemented as sequential algorithms. Show that the action of the first would be unchanged, whereas the second would produce very odd effects on some binary images.

3

Basic Image Filtering Operations

Image filtering involves the application of window operations that achieve useful effects, such as noise removal or image enhancement. This chapter is concerned particularly with what can be achieved with quite basic approaches, such as application of local mean, median, or mode filters to digital images. The focus is on grayscale images, although some aspects of color processing are also covered.

Look out for:

- what can be achieved by low-pass filtering in the spatial frequency domain.
- how the same process can be carried out by convolution in the spatial domain.
- the problem of impulse noise and what can be achieved with a limiting filter.
- the value of median, mode, and rank order filters.
- how computational load can be reduced.
- the distinction between image enhancement and image restoration.
- the distortions produced by standard filters—mean, Gaussian, median, mode, and rank order filters.

This chapter takes pain to delve into the properties of a variety of standard types of filter, because it is necessary to know both what they can achieve and what their limitations are. In fact, the edge shifts produced by most of these filters are small but predictable, and therefore correctable in principle. The exception is the rank order filter, for which the shifts can be large—but this is the *advantage* of this type of filter, which is at the core of mathematical morphology (see Chapter 7).

3.1 INTRODUCTION

Chapter 2 is concerned with simple imaging operations, including such problems as thresholding grayscale images and suppressing noise in binary images. In this chapter, the discussion is extended to noise suppression and enhancement in

grayscale images. Although these types of operation can for the most part be avoided in industrial applications of vision, it is useful to examine them in some depth because of their wide use in a variety of other image processing applications and because they set the scene for much of what follows. In addition, some fundamental issues come to light which are of vital importance.

It has already been seen that noise can arise in real images and it is hence necessary to have sound techniques for suppressing it. Commonly, in electrical engineering applications, noise is removed by means of low-pass or other filters that operate in the frequency domain (Rosie, 1966). Applying these filters to 1-D time-varying analog signals is straightforward, since it is necessary only to place them at suitable stages in the sequence of black boxes through which the signals pass. For digital signals the situation is more complicated, since the frequency transform of the signal must first be computed, then the low-pass filter applied, and finally the signal obtained from the modified transform by converting back to the time domain. Thus, two Fourier transforms have to be computed, although modifying the signal while it is in the frequency domain is a straightforward task (Fig. 3.1). In fact, the amount of processing involved in computing the discrete Fourier transform of a signal represented by N samples is of order N^2 (we shall write this as $O(N^2)$), although the amount of computation can be cut down to $O(N \log_2 N)$ by employing the fast Fourier transform (FFT) (Gonzalez and Woods, 1992). This then becomes a practical approach for the elimination of noise.

When applying these ideas to images we must first note that the signal is a spatial rather than a time-varying quantity and must be filtered in the spatial frequency domain. Mathematically this makes no real difference, but there are nevertheless significant problems. First, there is no satisfactory analog shortcut and the whole process has to be carried out digitally (we here ignore optical processing methods despite their obvious power, speed, and high resolution, because they are by no means trivial to marry with digital computer technology). Second,

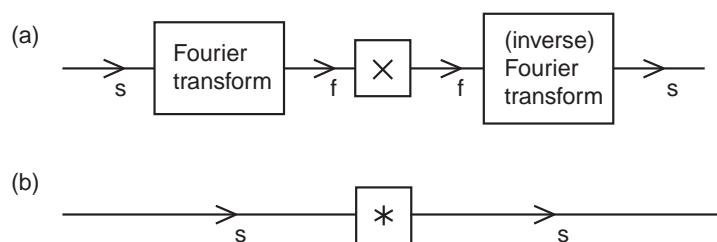


FIGURE 3.1

Low-pass filtering for noise suppression: s , spatial domain; f , spatial frequency domain; \times , multiplication by low-pass characteristic; $*$, convolution with Fourier transform of low-pass characteristics. (a) Low-pass filtering achieved most simply, by a process of multiplication in the (spatial) frequency domain; (b) low-pass filtering achieved by a process of convolution. Note that (a) may require more computation overall, because of the two Fourier transforms that have to be performed.

for an $N \times N$ pixel image, the number of operations required to compute a Fourier transform is $O(N^3)$ and the FFT only reduces this to $O(N^2 \log_2 N)$, so the amount of computation is quite considerable (here it is assumed that the 2-D transforms are implemented by successive passes of 1-D transforms: see Gonzalez and Woods, 1992). Note also that two Fourier transforms are required for the purpose of noise suppression (Fig. 3.1). Nevertheless, in many imaging applications it is worth proceeding in this way, because not only noise can be removed but also TV scan lines and other artifacts can be filtered out. This situation applies particularly in remote sensing and space technology. However, in industrial applications the emphasis is always on real-time processing, so in many cases it is not practicable to remove noise by spatial frequency domain operations. A further problem is that low-pass filtering is suited to removing Gaussian noise, but distorts the image if it is used to remove impulse noise.

Section 3.2 discusses Gaussian smoothing, in both the spatial frequency and the spatial domains. The subsequent three sections introduce median filters, mode filters, and then general rank order filters, and contrast their main properties and uses. In Section 3.6, consideration is given to reducing computational load, with particular reference to the median filter. Section 3.7 introduces the sharp–unsharp masking technique, which provides a rather simple route to image enhancement. Then follow a number of sections that concentrate on the edge shifts produced by the various filters. In the case of the median filter, the discrete theory (Section 3.9) is much more exact than the continuum model (Section 3.8). All edge shifts are quite small, except for rank order filters (Section 3.12): these are treated fairly fully because of their relevance to widely used morphological operators (Chapter 7) where the shifts are turned to advantage. Finally, Section 3.14 gives a brief discussion on the application of filters to color images.

3.2 NOISE SUPPRESSION BY GAUSSIAN SMOOTHING

Low-pass filtering is normally thought of as the elimination of signal components with high spatial frequencies, and it is therefore natural to carry it out in the spatial frequency domain. Nevertheless, it is possible to implement it directly in the spatial domain. That this is possible is due to the well-known fact (Rosie, 1966) that multiplying a signal by a function in the spatial frequency domain is equivalent to convolving it with the Fourier transform of the function in the spatial domain (Fig. 3.1). If the final convolving function in the spatial domain is sufficiently narrow, then the amount of computation involved will not be excessive: in this way a satisfactory implementation of the low-pass filter can be sought. It now remains to find a suitable convolving function.

If the low-pass filter is to have a sharp cut-off, then its transform in image space will be oscillatory: an extreme example of this is the sinc ($\sin x/x$) function, which is the spatial transform of a low-pass filter of rectangular profile (Rosie,

1966). Oscillatory convolving functions are unsatisfactory since they can introduce halos around objects, hence distorting the image quite grossly. Marr and Hildreth (1980) suggested that the right types of filter to apply to images are those that are well-behaved (nonoscillatory) both in the frequency and in the spatial domain. Gaussian filters are able to fulfill this criterion optimally: they have identical forms in the spatial and spatial frequency domains. In 1-D, these forms are:

$$f(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \quad (3.1)$$

$$F(\omega) = \exp\left(-\frac{1}{2}\sigma^2\omega^2\right) \quad (3.2)$$

Thus, the type of spatial convolving operator required for the purpose of noise suppression by low-pass filtering is one that approximates to a Gaussian profile. Many such approximations appear in the literature: these vary with the size of the neighborhood chosen and in the precise values of the convolution mask coefficients.

One of the most common is the following mask, first introduced in Chapter 2, which is used more for simplicity of computation than for its fidelity to a Gaussian profile:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Another commonly used mask, which approximates more closely to a Gaussian profile, is the following:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

In both cases, the coefficients that precede the mask are used to weight all the mask coefficients: as mentioned in Section 2.3, these weights are chosen so that applying the convolution to an image does not affect the average image intensity. These two convolution masks probably account for over 80% of all discrete approximations to a Gaussian. Note that as they operate within a 3×3 neighborhood, they are reasonably narrow and hence incur a relatively small computational load.

Let us next study the properties of this type of operator, deferring for now consideration of Gaussian operators in larger neighborhoods. First, imagine such an operator is applied to a noisy image whose intensity is inherently uniform. Then clearly noise is suppressed, as it is now averaged over nine pixels. This averaging model is obvious for the first of the two masks above but in fact applies equally to the second mask, once it is accepted that the averaging effect is differently distributed in accordance with the improved approximation to a Gaussian profile.

Although this example shows that noise is suppressed, it is clear that the signal will also be affected. This problem arises only where the signal is initially non-uniform: indeed, if the image intensity is constant, or if the intensity map approximates to a plane, there is again no problem. However, if the signal is uniform over one part of a neighborhood and rises in another part of it, as is bound to occur adjacent to the edge of an object, then the object will make itself felt at the center of the neighborhood in the filtered image (see Fig. 3.2). As a result, the edges of objects become somewhat blurred. Looking at the operator as a “mixing operator” that forms a new picture by mixing together the intensities of pixels fairly close to each other, it is intuitively obvious why blurring occurs.

It is also apparent from a spatial frequency viewpoint why blurring should occur. Basically, we are aiming to give the signal a sharp cut-off in the spatial frequency domain, and as a result it will become slightly blurred in the spatial domain. Clearly, the blurring effect can be reduced by using the narrowest possible approximation to a Gaussian convolution filter, but at the same time the noise suppression properties of the filter are lessened. Assuming that the image was initially digitized at roughly the correct spatial resolution, it will not normally be appropriate to smooth it using convolution masks larger than 3×3 or at most 5×5 pixels (here we ignore methods of analyzing images that use a number of versions of the image with different spatial resolutions: see for example, Babaud et al., 1986).

Overall, low-pass filtering and Gaussian smoothing are not appropriate for the applications considered here because of the blurring effects they introduce. Note also that where interference occurs, it can give rise to impulse or “spike” noise (corresponding to a number of individual pixels having totally the wrong intensities): merely averaging this noise over a larger neighborhood can make the situation worse, since the spikes will be smeared over a sizeable number of pixels and will distort the intensity values of all of them. This consideration is important as it leads naturally to the concepts of limit and median filtering.

/

x

FIGURE 3.2

Blurring of object edges by simple Gaussian convolutions. The simple Gaussian convolution can be regarded as a grayscale neighborhood “mixing” operator, hence explaining why blurring arises.

3.3 MEDIAN FILTERS

The idea explored here is to locate the pixels in the image that have extreme and therefore highly improbable intensities and to ignore their actual intensities, replacing them with more suitable values. This is akin to drawing a graph through a set of plots and ignoring the plots that are evidently a long way from the best fit curve. An obvious way of achieving this is to apply a “limit” filter that prevents any pixel having an intensity outside the intensity range of its neighbors:

```
for all pixels in image do {
    minP = min (P1, P2, P3, P4, P5, P6, P7, P8);
    maxP = max(P1, P2, P3, P4, P5, P6, P7, P8);
    if (P0 < minP) Q0 = minP;
    else if (P0 > maxP) Q0 = maxP;
    else Q0 = P0;
}
```

(3.3)

To develop this technique, it is necessary to examine the local intensity distribution within a particular neighborhood. Points at the extremes of the distribution are quite likely to have arisen from impulse noise. So it is sensible not only to eliminate these points, as in the limit filter, but also to try taking the process further, removing equal areas at either end of the distribution and ending with the median. Thus, we arrive at the median filter, which takes all the local-intensity distributions and generates a new image corresponding to the set of median values. As the preceding argument indicates, the median filter is excellent at impulse noise suppression and this is amply confirmed in practice (see Fig. 3.3).

FIGURE 3.3

Effect of applying a 3×3 median filter to the image of Fig. 2.1(a). Note the slight loss of fine detail and the rather “softened” appearance of the whole image.

In view of the blurring caused by Gaussian smoothing operators, it is pertinent to ask whether the median filter also induces blurring. In fact, Fig. 3.3 shows that any blurring is only marginal, although there is some slight loss of fine detail that can give the resulting pictures a “softened” appearance. Theoretical discussion of this point is deferred for now; the lack of blur makes good the main deficiency of the Gaussian smoothing filter and results in the median filter being perhaps the most widely used filter in general image processing applications.

There are many ways of implementing the median filter: Table 3.1 reproduces only an obvious algorithm that essentially implements the above description. The notation of Chapter 2 is used but is augmented in order to permit the nine pixels in a 3×3 neighborhood to be accessed in turn with a running suffix (specifically, P0 to P8 are written as $P[m]$ where m runs from 0 to 8).

The operation of the algorithm is as follows: first, the histogram array is cleared and the image is scanned, generating a new image in Q-space; then, for each neighborhood, the histogram of intensity values is constructed; then the median is found; and, finally, points in the histogram array that have been incremented are cleared. This last feature eliminates the need to clear the whole histogram and hence saves computation. Unlike the general situation in which the median of a distribution is being located, only one (half) scan through the distribution is required, since the total area is known in advance (in this case it is 9).

As is clear from the above discussion, methods of computing the median involve pixel intensity sorting operations. If a bubble sort (Gonnet, 1984) were used for this purpose, then up to $O(n^4)$ operations would be required for an $n \times n$ neighborhood, compared with some 256 operations for the histogram method described above. Thus, sorting methods such as the bubble sort are faster for small neighborhoods where n is 3 or 4 but not for neighborhoods where n is greater than 5, or where pixel intensity values are more restricted.

Much of the discussion of the median filter in the literature is concerned with saving computation (Narendra, 1978; Huang et al., 1979; Danielsson, 1981). In

Table 3.1 An Implementation of the Median Filter

```

for (i = 0; i <= 255; i++) hist[i] = 0;
for all pixels in image do {
    for (m = 0; m <= 8; m++) hist[P[m]]++;
    i = 0; sum = 0;
    while (sum < 5) {
        sum = sum + hist[i];
        i = i + 1;
    }
    Q0 = i - 1;
    for (m = 0; m <= 8; m++) hist[P[m]] = 0;
}

```

particular, it has been noticed that, on proceeding from one neighborhood to the next, relatively few new pixels are encountered: this means that the new median value can be found by updating the old value rather than starting from scratch (Huang et al., 1979).

3.4 MODE FILTERS

Having considered the mean and the median of the local intensity distribution as candidate intensity values for noise smoothing filters, it also seems relevant to consider the mode of the distribution. Indeed, we might imagine that this is if anything more important than the mean or the median, since the mode represents the most probable value of any distribution.

However, a tedious problem arises as soon as we attempt to apply this idea. The local intensity distribution is calculated from relatively few pixel intensity values (Fig. 3.4). This means that instead of a smooth intensity distribution whose mode is easily located, we are almost certain to have a multimodal distribution whose highest point does not indicate the position of the *underlying* mode. Clearly the distribution needs to be smoothed out considerably before the mode is computed. Another tedious problem is that the width of the distribution varies widely from neighborhood to neighborhood (e.g., from close to zero to close to 256), so that it is difficult to know quite how much to smooth the distribution in any instance. For these reasons, it is likely to be better to choose an indirect measure of the position of the mode rather than to attempt to measure it directly.

In fact, the position of the mode can be estimated with reasonable accuracy once the median has been located (Davies, 1984a, 1988c). To understand the technique, it is necessary to consider how local intensity distributions of various sorts arise in practical situations. At most positions in an image, variations in pixel intensity are generated by steady changes in background illumination, or by steady variations in surface orientation, or else by noise. Thus, a symmetrical unimodal local intensity distribution is to be expected. It is well known that the mean, median, and mode are coincident in such cases. More problematic is what

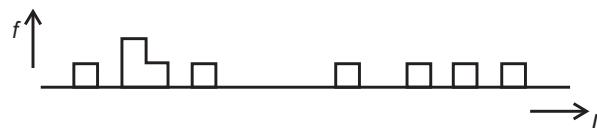


FIGURE 3.4

The sparse nature of the local intensity histogram for a small neighborhood. This situation clearly causes significant problems for estimation of the mode. It also has definite implications for rigorous estimation of the underlying median, assuming that the observed intensities are only noisy samples of the ideal intensity pattern (see Section 3.8.3).

Source: © IEEE 1984

FIGURE 3.5

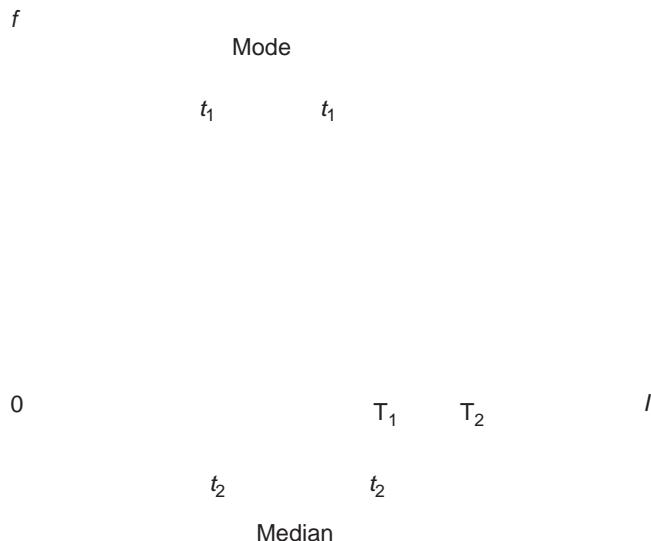
Local models of image data near the edge of an object: (a) cross-sections of an edge falling in the vicinity of a filter neighborhood; (b) corresponding local intensity distributions when very little image noise is present; (c) situation when the noise level is increased.

Source: © IEEE 1984

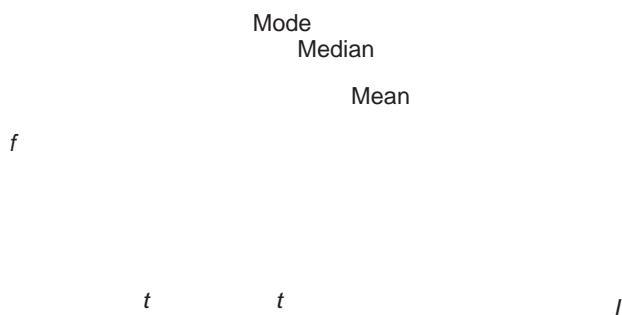
happens to the intensity variation near the edge of an object in the image. Here the local intensity distribution is unlikely to be symmetrical and, more important, it may not even be unimodal. In fact, near an edge the distribution is in general inherently *bimodal*, since the neighborhood contains pixels with intensities corresponding to the values they would have on either side of the edge (Fig. 3.5). Considering the image as a whole, this will be the most likely alternative to a symmetrical unimodal distribution, any further possibilities such as trimodal distributions being rare and of varied causes (e.g., odd glints on the edges of metal objects) which are outside the scope of the present discussion.¹

If the neighborhood straddles an edge and the local intensity distribution is bimodal, the larger peak position should clearly be selected as the most probable intensity value. A good strategy for finding the larger peak is to eliminate the smaller peak. If we knew the position of the mode, we could find where to truncate the smaller peak by first finding which extreme of the distribution was closer to the mode, and then moving an equal distance to the opposite side of the mode

¹Here we are ignoring the effects of noise and just considering the underlying image signal.

**FIGURE 3.6**

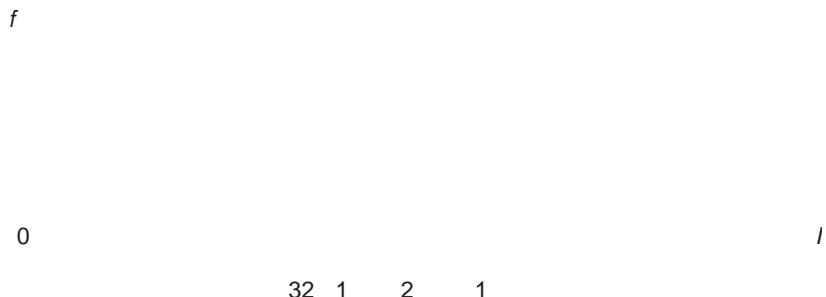
Rationale for the method of truncation. The obvious position at which to truncate the distribution is T_1 . Since the position of the mode is not initially known, it is suboptimal but safe to truncate instead at T_2 .

**FIGURE 3.7**

Relative positions of the mode, median, and mean for a typical unimodal distribution. This ordering is unchanged for a bimodal distribution, as long as it can be approximated by two Gaussian distributions of similar width.

Source: © IEEE 1984

(Fig. 3.6). Since we start off *not* knowing the position of the mode, one option is to use the position of the median as an estimator of the position of the mode, and then to use that position to find where to truncate the distribution. Since it invariably happens that the three means take the order mean, median, and mode (see Fig. 3.7), except when distributions are badly behaved or multimodal, this method

**FIGURE 3.8**

Iterative truncation of the local intensity distribution. Here the median converges on the mode within three iterations of the truncation procedure. This is possible since at each stage the mode of the new truncated distribution remains the same as that of the previous distribution.

FIGURE 3.9

Effect of a single application of 3×3 truncated median filter to the image of Fig. 2.1(a).

is cautious in the sense that it truncates less of the distribution than the required amount: this makes it a safe method to use. When we now find the median of the truncated distribution, the position is much closer to the mode than the original median was, a good proportion of the second peak being removed ([Fig. 3.8](#)). Iteration could be used to find an even closer approximation to the position of the mode. However, the method gives a marked enhancement in the image even when this is not done ([Fig. 3.9](#)).

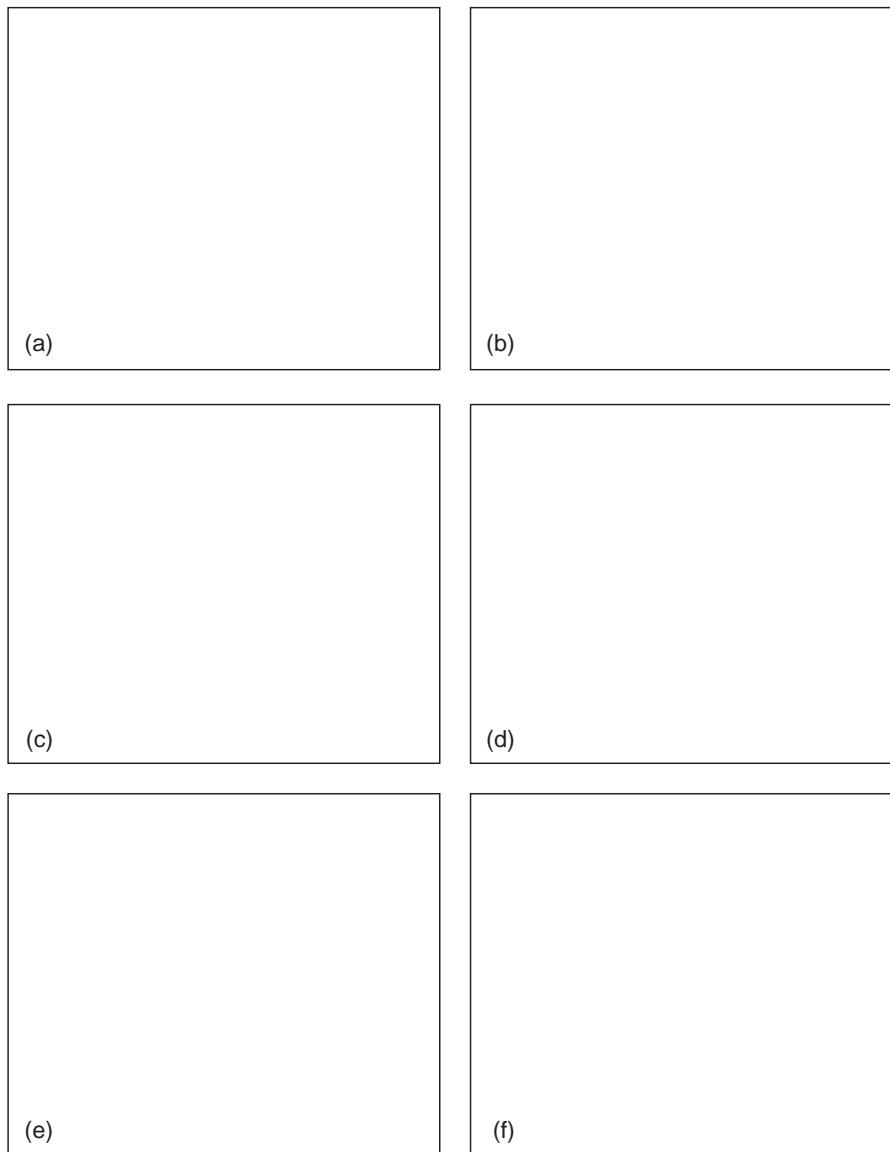
We next examine more closely the properties of the “truncated median filter” (TMF) described above. The median filter is highly successful at removing noise,

**FIGURE 3.10**

Image enhancement performed by the mode filter. Here the onset of the edge is pushed laterally by the action of the mode filter within one neighborhood; since the same happens from the other side within an adjacent neighborhood, the actual position of the edge is unchanged in first order. The overall effect is to sharpen the edge.

whereas the TMF not only removes noise but also enhances the image so that edges become sharper. [Figure 3.10](#) makes it clear why this should happen. Basically, at a location even very slightly to one side of an edge, a majority of the pixel intensities contribute to the larger peak and the TMF ignores the pixel intensities contributing to the smaller peak. Thus, the TMF makes an informed binary choice about which side of the edge it is on. At first this seems to mean that it pushes a nearby edge further away. However, it must be remembered that it actually “pushes the edge away” from both sides, and the result is that its sides are made sharper and object outlines are crispened up. Particularly striking is the effect of applying the TMF to an image a number of times, when objects start to become segmented into regions of fairly uniform intensity ([Fig. 3.11](#)). The complete algorithm for achieving this is outlined in [Table 3.2](#).

This problem has been dealt with at some length for a number of reasons. First, the mode filter has not hitherto received the attention it deserves. Second, the median filter seems to be used fairly universally, often without very much justification or thought. Third, all these filters show what markedly different characteristics are available merely by analyzing the contents of the local intensity distribution and ignoring totally where in the neighborhood the different intensities appear: it is perhaps remarkable that there is sufficient information in the local intensity distribution for this to be possible. All this shows the danger of applying operators that have been derived in an *ad hoc* manner without first making a specification of what is required and then designing an operator with the required characteristics. In fact, the situation appears to be that if we want a filter that has maximum impulse noise suppression capability, then we should use a median filter; and if we want a filter that enhances images by sharpening edges,

**FIGURE 3.11**

Results of repeated action of the truncated median filter: (a) the original, moderately noisy picture; (b) effect of a 3×3 median filter; (c)–(f) effect of 1–4 passes of the basic truncated median filter, respectively.

Source: © IEEE 1984

then we should use a mode filter or TMF (note that the TMF should be an improvement on the mode filter in that it is more cautious very close to an edge transition, where noise prevents an exact judgement being made as to which side of the edge a pixel is on: see Davies (1984a, 1988c)).

Table 3.2 Outline of Algorithm for Implementing the Truncated Median Filter

```

do { // as many passes over image as necessary
    for all pixels in image do {
        compute local intensity distribution;
        do { // iterate to improve estimate of mode
            find minimum, median, and maximum intensity values;
            decide from which end local intensity distribution should
            be truncated;
            deduce where local intensity distribution should be
            truncated;
            truncate local intensity distribution;
            find median of truncated local intensity distribution;
        } until median sufficiently close to mode of local distribution;
        transfer estimate of mode to output image space;
    }
} until sufficient enhancement of image;

```

Comments:

- (i) *The outermost and innermost loops can normally be omitted (i.e., they need to be executed once only).*
- (ii) *The final estimate of the position of the mode can be performed by simple averaging instead of computing the median: this has been found to save computation with negligible loss of accuracy.*
- (iii) *Instead of the minimum and maximum intensity values, the positions of the outermost octiles (for example) may be used to give more stable estimates of the extremes of the local intensity distribution.*

While considering enhancement, attention has been restricted to filters based on the local intensity distribution: there are many filters that enhance images without the aid of the local intensity distribution (Lev et al., 1977; Nagao and Matsuyama, 1979), but they are not within the scope of this chapter. Note that the method of “sharp–unsharp masking” (Section 3.7) performs an enhancement function, although its main purpose is to restore images that have inadvertently become blurred, e.g., by a hazy atmosphere or defocussed camera.

Finally, while this section has concentrated on the grayscale properties of mode filters, Charles and Davies (2003a, 2004) have shown how to devise versions of the TMF that operate on color images. Typical results are shown in Fig. 3.12. In addition, Fig. 3.13 shows that the TMF has the useful property of being able to eliminate very large amounts of impulse noise from images—significantly more than a median filter—in spite of being designated as an image enhancement filter.

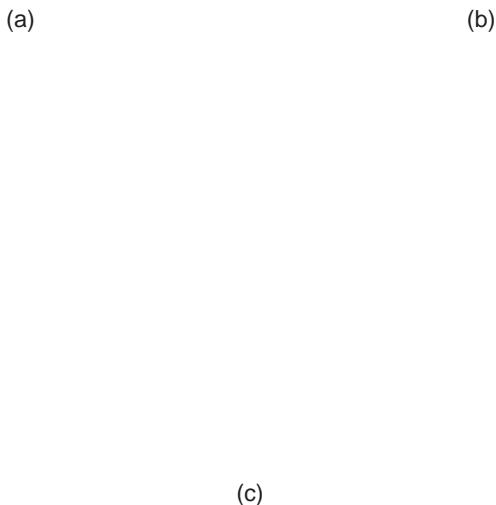
**FIGURE 3.12**

Color filtering of brightly colored objects. (a) Original color image of some sweets. (b) Vector median filtered version. (c) Vector mode filtered version. (d) Version to which a mode filter has been applied to each color channel separately. Note that (b) and (c) show no evidence of color bleeding, although it is strongly evident in (d). It is most noticeable as isolated pink pixels, plus a few green pixels, around the yellow sweets. For further details on color bleeding, see Section 3.14 (see also color Plate 5).

Source: © RPS 2004

3.5 RANK ORDER FILTERS

The principle employed in rank order filters is to take all the intensity values in a given neighborhood, to place these in order of increasing value, and finally to select the r th of the n values and return this value as the filter local output value. Clearly, n rank order filters can be specified in terms of the value r that is used, but these filters are intrinsically nonlinear, i.e., the output intensity cannot be expressed as a linear sum of the component intensities within the neighborhood. In particular, the median filter (for which $r = (n + 1)/2$, and which is only defined

**FIGURE 3.13**

Color filtering of images containing substantial impulse noise. (a) Version of the Lena image containing 70% random color impulse noise. (b) Effect of applying a vector median filter, and (c) effect of applying a vector mode filter. While the mode filter is designed more for enhancement than for noise suppression, it has been found to perform remarkably well at this task when the noise level is very high (see also color Plate 6).

Source: © RPS 2004

if n is odd)² does not normally give the same output image as a mean filter; indeed, it is well known that the mean and median of a distribution are in general only coincident for symmetrical distributions. Note that minimum and maximum filters (corresponding to $r = 1$ and $r = n$, respectively) are also often classed as morphological filters (see Chapter 7).

²If n is even, it is usual to take the mean of the central two values in the distribution as representing the median.

3.6 REDUCING COMPUTATIONAL LOAD

Significant efforts have been made to speedup the operation of the Gaussian filter since implementations in large neighborhoods require considerable amounts of computation (Wiejak et al., 1985). For example, smoothing an image of 256×256 pixels using a 30×30 Gaussian convolution mask involves 64 million basic operations. For such a basic operation as smoothing, this is unacceptable. However, it is possible to cut down the amount of computation drastically, since a 2-D Gaussian convolution can be factorized into two 1-D Gaussian convolutions, which can be applied in turn:

$$\exp\left(-\frac{r^2}{2\sigma^2}\right) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \exp\left(-\frac{y^2}{2\sigma^2}\right) \quad (3.4)$$

It is important to realize that the decomposition is rigorously provable and is not an approximation: we shall refer to this below in the context of the median filter. Meanwhile, the decompositions for the two 3×3 Gaussian filters we discussed earlier are:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad (3.5)$$

and

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \quad (3.6)$$

Overall, this approach replaces a single $n \times n$ operator whose load is $O(n^2)$ with two operators of load $O(n)$, and ignoring scanning and other overheads, the saving factor is $n/2$. Hence, for $n > 2$, there will always be a useful saving.

In fact, it is not possible to decompose the median filter in the same way without making approximations. However, it is quite common to try to perform a similar function by applying two 1-D median filters in turn (Narendra, 1978). Although the effect is similar in its outlier rejection properties to the standard 2-D median filter, the following example confirms that the two are not exactly equivalent:

Original image segment:

$$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

After applying a 3×3 filter:

$$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

After applying a 3×1 and then a 1×3 filter:

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

After applying a 3×1 and then a 1×3 filter:

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

By chance, the final results in the last two cases are the same, but this is a relatively rare occurrence.

In general, spots and streaks not more than one pixel wide are eliminated quite effectively by the original or by the separated forms of the filter. Larger filters should effectively eliminate wider spots and streaks, although exact functional equivalence between the original and its separated forms is not to be expected, as has been indicated.

Finally, the problem of inexact decomposition is not an exclusive property of nonlinear filters; many linear filters cannot be decomposed exactly either: this is evident because the number of independent coefficients in an $n \times n$ mask is n^2 , which is much greater than the total number in an $n \times 1$ and a $1 \times n$ component mask.

3.7 SHARP–UNSHARP MASKING

When images are blurred either before or as part of the process of acquisition, it is possible frequently to restore them substantially to their ideal state. Properly, this is achieved by making a model of the blurring process and applying an inverse transformation that is intended to cancel the blurring. This is a complex task to carry out rigorously, but in some cases a rather simple method called sharp–unsharp masking is able to produce significant improvement (Gonzalez and Woods, 1992). As indicated in Fig. 3.14, this technique involves first obtaining an even more blurred version of the image (e.g., with the aid of a Gaussian filter) and then subtracting this image from the original. Note that the amount of

**FIGURE 3.14**

The principle of sharp–unsharp masking: (a) cross-section of an idealized edge; (b) observed edge; (c) artificially blurred version of (b); (d) result of subtracting a proportion of (c) from (b).

artificial blurring to apply and the proportion of the blurred image to subtract are rather arbitrary quantities that are normally adjusted by eye. Thus, the method is better categorized under the heading “enhancement” than “restoration,” as it is not the precise mathematical technique normally understood by the latter term. Of such enhancement techniques, Hall (1979) states: “Much of the art of enhancement is knowing when to stop.”

3.8 SHIFTS INTRODUCED BY MEDIAN FILTERS

Despite knowing the main characteristics of the different types of filter there are still some unknown factors. In particular, it is often important (especially when making precision measurements on manufactured components) to ensure that noise is removed in such a way that object locations and sizes are unchanged. However, at this point the following two problems arise.

First, it has been assumed that the intensity profile of an edge is symmetrical. If this is so, then the mean, median, and mode of the local intensity distribution will be coincident and there will clearly be no overall bias for any of them. However, when the edge profile is asymmetrical, it will not be obvious in the absence of a detailed model of the situation what the result will be for any of the

**FIGURE 3.15**

Variation in local intensity distribution with position of neighborhood: (a) neighborhood of radius a overlapping a dark circular object of radius b ; (b)–(d) intensity distributions f when the separations of the centers are respectively less than, equal to, or greater than the distance d for which the object bisects the area of the neighborhood.

filters. The situation is even more involved when significant noise is present (Yang and Huang, 1981; Bovik et al., 1987). Since the problem is so data-dependent, it is not profitable to consider it further here.

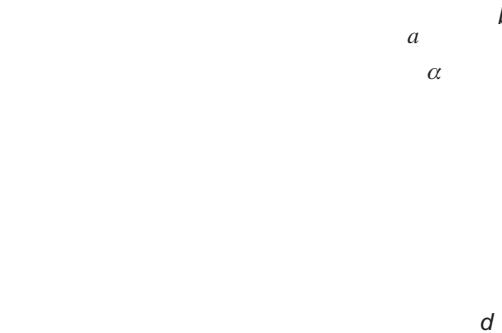
The second problem concerns the situation for a curved edge. In this case, there is again a variety of possibilities, and filters employing the different means will modify the edge position in ways that depend markedly on its shape. In robot vision applications, the median filter is the one we are most likely to use because its main purpose is to suppress noise without introducing blurring. Hence, the bias produced by this type of filter is worth considering in some detail. This is done in the next subsection.

3.8.1 Continuum Model of Median Shifts

This section takes the case of a continuous image (i.e., a nondiscrete lattice), assuming (i) that the image is binary, (ii) that neighborhoods are exactly circular, and (iii) that images are noise-free. To proceed we notice that binary edges have symmetrical cross-sections, while straight edges extend this symmetry into 2-D: hence applying a median filter in a (symmetrical) circular neighborhood cannot pull a straight edge to one side or the other.

Now consider what happens when the filter is applied to an edge that is not straight. If, for example, the edge is circular, the local intensity distribution will contain two peaks whose relative sizes will vary with the precise position of the neighborhood (Fig. 3.15). At some position the sizes of the two peaks will be identical. Clearly, this happens when the center of the neighborhood is situated at a point where the output of the median filter changes from dark to light (or *vice versa*). Thus, the median filter produces an inward shift toward the center of a circular object (or the center of curvature), whether the object is dark on a light background or light on a dark background.

To calculate the magnitude of this effect, we need to find at what distance d from the center of a circular object (of radius b) the area of a circular neighborhood (of radius a) is bisected by the object boundary.

**FIGURE 3.16**

Geometry for calculating neighborhood and object overlap.

From Fig. 3.16 the area of the sector of angle 2β is βb^2 , while the area of the triangle of angle 2β is $b^2 \sin\beta \cos\beta$. Hence, the area of the segment shown shaded is:

$$B = b^2(\beta - \sin\beta \cos\beta) \quad (3.7)$$

Making a similar calculation of the area A of a circular segment of radius a and angle 2α , the area of overlap (Fig. 3.16) between the circular neighborhood of radius a and the circular object of radius b may be deduced as:

$$C = A + B \quad (3.8)$$

For a median filter this is equal to $\pi a^2 / 2$. Hence:

$$F = a^2(\alpha - \sin\alpha \cos\alpha) + b^2(\beta - \sin\beta \cos\beta) - \frac{\pi a^2}{2} = 0 \quad (3.9)$$

where

$$a^2 = b^2 + d^2 - 2bd \cos\beta \quad (3.10)$$

and

$$b^2 = a^2 + d^2 - 2ad \cos\alpha \quad (3.11)$$

To solve this set of equations, we take a given value of d , deduce values of α and β , calculate the value of F , and then adjust the value of d until $F = 0$. Since d is the modified value of b obtained after filtering, the shift produced by the filtering process is:

$$D = b - d \quad (3.12)$$

The results of doing this computation numerically have been found by Davies (1989b). As expected, $D \rightarrow 0$ as $b \rightarrow \infty$ or $a \rightarrow 0$. Conversely, the shift becomes very large as a first approaches and then exceeds b . Note, however, that when

$a > \sqrt{2}b$ the object is ignored, being small enough to be regarded as irrelevant noise by the filter: beyond this point it has no effect at all on the final image. The maximum edge shift before the object finally disappears is $(2 - \sqrt{2})b \approx 0.586b$.

It is instructive to approximate the above equations for the case when edge curvature is small, i.e., $a \ll b$. Under these conditions, β is small, $\alpha \approx \pi/2$ and $d \approx b$. Hence, we find:

$$\beta \approx \frac{a}{b} \quad (3.13)$$

After some manipulation the edge shift D is obtained in the form:

$$D \approx \frac{a^2}{6b} = \frac{\kappa a^2}{6} \quad (3.14)$$

$\kappa = 1/b$ being the local curvature. In Chapter 6, this equation is found to be useful for estimating the signals from a median-based corner detector.

3.8.2 Generalization to Grayscale Images

To extend these results to grayscale images, first consider the effect of applying a median filter near a smooth step edge in 1-D. Here the median filter gives zero shift, since for equal distances from the center to either end of the neighborhood there are equal numbers of higher and lower intensity values and hence equal areas under the corresponding portions of the intensity histogram. Clearly this is always valid where the intensity increases monotonically from one end of the neighborhood to the other—a property first pointed out by Gallagher and Wise (1981) [for more recent discussions on related “root” (invariance) properties of signals under median filtering, see Fitch et al. (1985) and Heinonen and Neuvo (1987)].

Next, it is clear that for 2-D images, the situation is again unchanged in the vicinity of a straight edge, since the situation remains highly symmetrical. Hence the median filter gives zero shift, as in the binary case.

For curved boundaries, the situation has to be considered carefully for grayscale edges, which, unlike binary edges, have finite slope. When boundaries are roughly circular, contours of constant intensity often appear as in Fig. 3.17. To find how a median filter acts, we merely need to identify the contour of median intensity (in 2-D the median intensity value labels a whole contour) that divides the area of the neighborhood into two equal parts. The geometry of the situation is identical to that already examined in Section 3.8.1: the main difference here is that for every position of the neighborhood, there is a corresponding median contour with its own particular value of shift depending on the curvature. Intriguingly, the formulae already deduced may immediately be applied for calculating the shift for each contour. Figure 3.17 shows an idealized case in which the contours of constant intensity have similar curvature, so that they are all moved

FIGURE 3.17

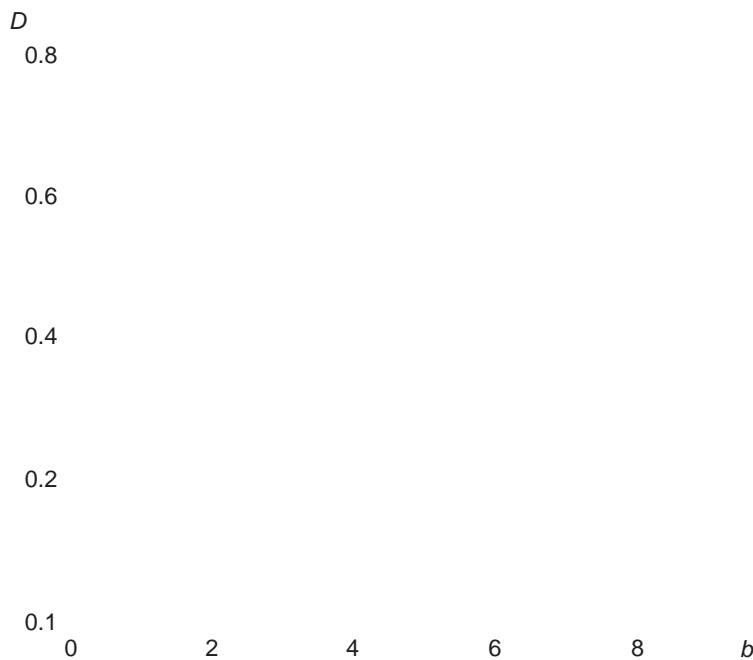
Contours of constant intensity on the edge of a large circular object, as seen within a small circular neighborhood.

inward by similar amounts. This means that, to a first approximation, the edges of the object retain their cross-sectional profile as it becomes smaller.

For grayscale images, the shifts predicted by this theory (with certain additional corrections: see Davies (1989b)) agree with experimental shifts within approximately 10% for a large range of circle sizes in a discrete lattice (see Fig. 3.18). Paradoxically, the agreement is less perfect for binary images, since circles of certain sizes show stability effects (akin to median root behavior): these effects tend to average out for grayscale images, owing to the presence of many contours of different sizes at different gray levels. Overall, the edge shifts obtained with median filters are now quite well understood. Figures 3.19 and 3.20 give some indication of the magnitudes of these shifts in practical situations. Note that once image detail such as a small hole or screw thread has been eliminated by a filter, it is not possible to apply any edge shift correction formula to recover it, although for larger features such formulae are useful for deducing true edge positions.

3.8.3 Problems with Statistics

Thus far it has been seen that computations of the position of the mode are made more difficult because of the sparse statistics of the local intensity distribution. In fact this also affects the median calculation. Suppose the median value happens to be well spaced near the center of the distribution (Fig. 3.4). Then a small error in

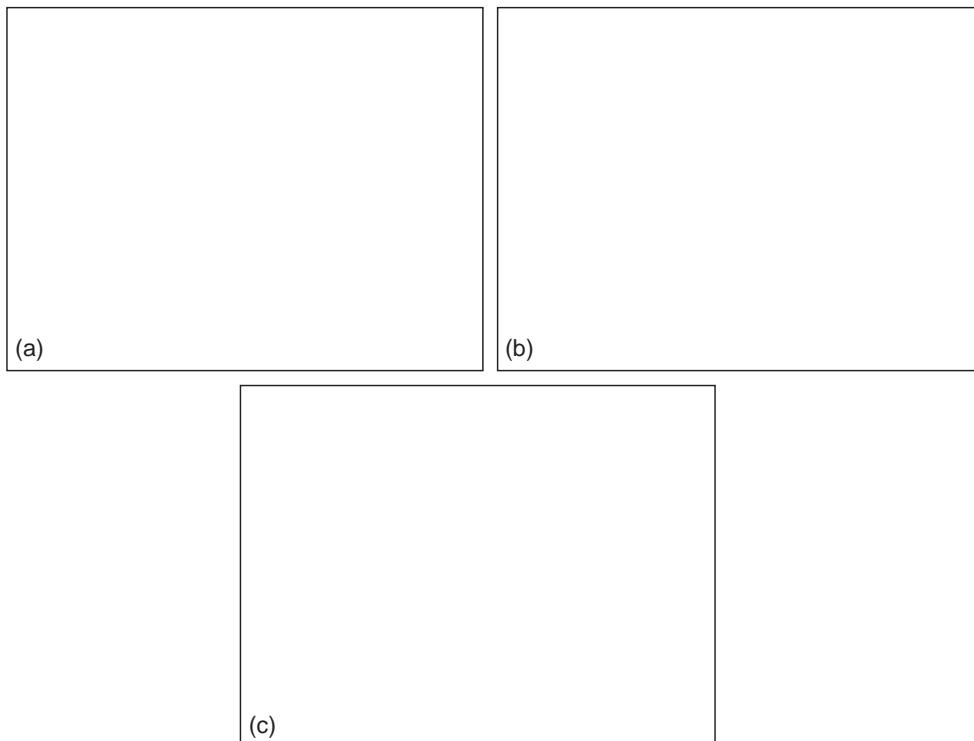
**FIGURE 3.18**

Edge shifts for 5×5 median filter applied to a grayscale image. The upper set of plots represents the experimental results and the upper continuous curve is derived from the theory of [Section 3.8.1](#). The lower continuous curve is derived from a more accurate model (Davies, 1989b). The lower set of plots represents the much reduced shifts obtained with the “detail-preserving” type of filter (see [Section 3.16](#)).

this one intensity value is immediately reflected in full when calculating the median: i.e., the poor statistics have biased the median in a particular way. Ideally, what is required is a stable estimator of the median of the underlying distribution. Thus, the distribution should be made smoother before arriving at a specific value for the median. In practice, this procedure adds significant computational load to the filter calculation and is commonly not carried out. As a consequence the median filter tends to result in runs of constant intensity, thereby giving images the “softened” appearance noted earlier. This is apparent on studying the following 1-D example:

Original:	0	0	1	0	0	1	1	2	1	2	2	1	2	3	3	4	3	2	2	3
Filtered:	?	0	0	0	0	1	1	1	2	2	2	2	2	3	3	3	3	2	2	?

Although histogram smoothing is not commonly carried out, some workers have felt it necessary to adjust the relative weights of the various pixels in the neighborhood according to their distance from the central pixel (Akey and Mitchell, 1984). This mimics what happens for a Gaussian filter and is theoretically

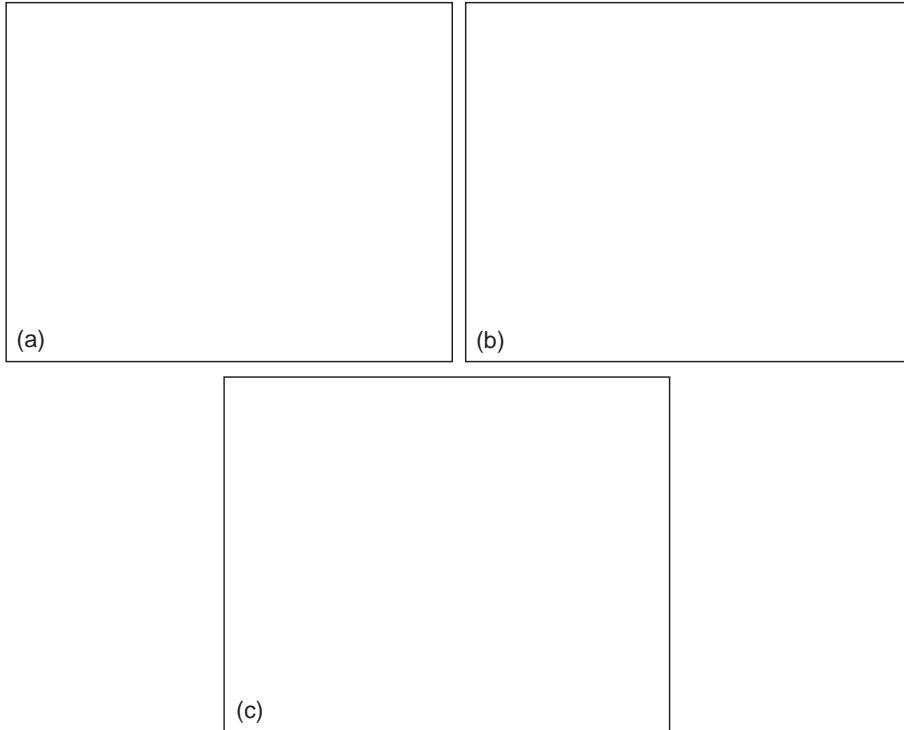
**FIGURE 3.19**

Edge smoothing property of the median filter: (a) Original image; (b) median filter smoothing of irregularities, in particular those around the boundaries (note how the threads on the screws are virtually eliminated although detail larger in scale than half the filter area is preserved), using a 21-element filter operating within a 5×5 neighborhood on a 128×128 pixel image of 6-bit gray scale; (c) effect of the detail-preserving filter (see Section 3.16).

necessary, although it is not generally implemented. However, there have recently been further developments on this front (e.g., see Charles and Davies, 2003b).

3.9 DISCRETE MODEL OF MEDIAN SHIFTS

To produce a discrete model of median shifts we need to recognize explicitly the positions of the pixels within the chosen neighborhood. We approximate by assuming that the intensity of any pixel is the mean intensity over the whole pixel and is represented by a sample positioned at the center of the pixel. We start by examining the case of a 3×3 neighborhood, and proceed by taking the underlying analog intensity variation to have contours of curvature κ , as shown in

**FIGURE 3.20**

Circular holes in metal objects before and after filtering: (a) original 128×128 pixel image with 6-bit gray scale; (b) 5×5 median-filtered image: the diminution in size of the hole is clearly visible and such distortions would have to be corrected for when taking measurement from real filtered images of this type; (c) result using a detail-preserving filter: some distortions are present although the overall result is much better than in (b).

Fig. 3.17. Following what happened in the continuum case, it will not matter whether the contours of constant intensity are those of a step edge or those of a slowly varying slant edge: it is what happens at the median contour that determines the shift that arises.

The starting point is that zero shift occurs for $\kappa = 0$. Next, if κ is even minutely greater than zero, the center pixel will not necessarily be the median pixel. Consider the case when the circular median intensity contour passes close to the center of the neighborhood at a small angle θ to the positive x -axis ([Figs. 3.21 and 3.22\(a\)](#)). In that case, the filter will produce a definite shift, whose value is:

$$D_\theta \approx \frac{1}{2} \kappa a_0^2 - a_0 \theta = \frac{1}{2} \kappa - \theta \quad (3.15)$$

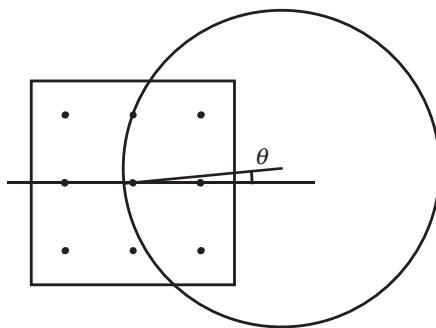


FIGURE 3.21

Geometry for calculation of median shifts on the discrete model.

Source: © IEE 1999

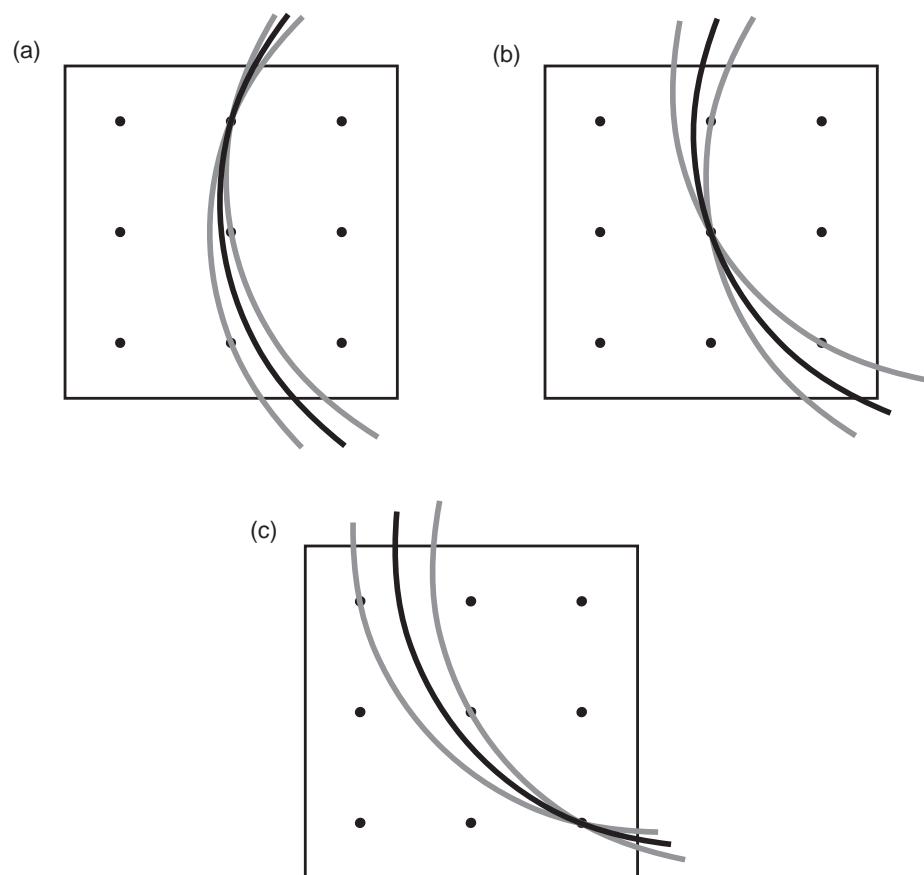
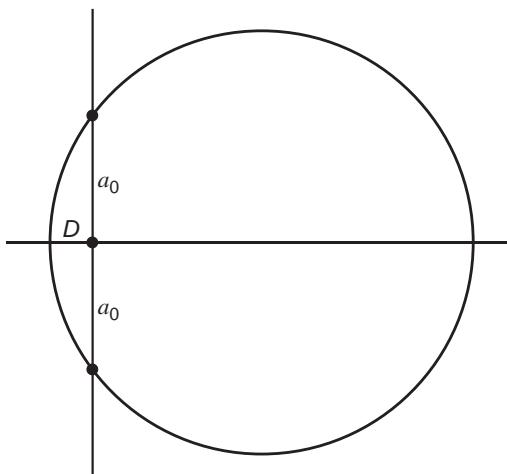


FIGURE 3.22

Geometry for calculation of median shifts at low κ . These three diagrams show the positions of the median pixels and the ranges of orientations of circular intensity contours for which they apply, (a) for low θ , (b) for intermediate θ , and (c) for high θ .

Source: © IEE 1999

**FIGURE 3.23**

Geometry for calculation of shift when the median contour passes through the centers of two pixels.

where a_0 is the inter-pixel separation—here taken as unity. The first term arises from the following simple result for the geometry of the circle of radius $b = 1/\kappa$ in Fig. 3.23:

$$a_0^2 = D \times (2b - D) \approx 2Db \quad (3.16)$$

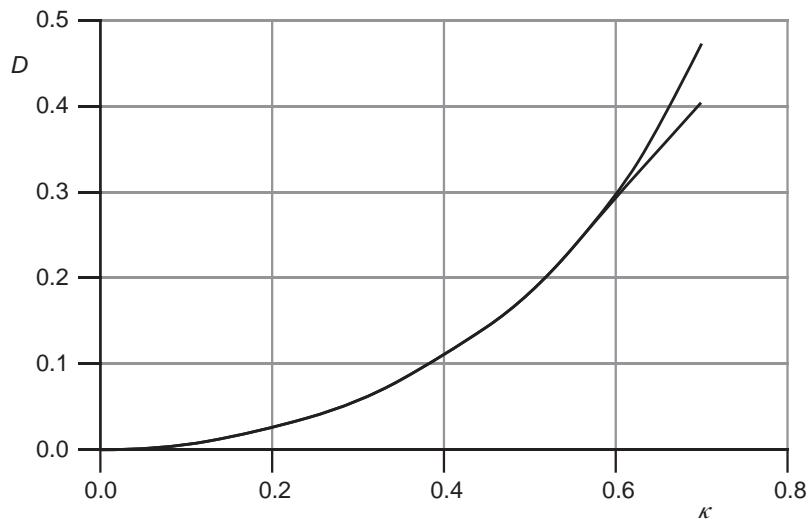
It is too tedious to recount here the complete analysis for the variation in D_θ for all θ . Suffice it to say that when it has been carried out, it is necessary to average it over all θ for each value of κ . When this is done (Davies, 1999f) the agreement between theory and experiment is essentially exact over a wide range of values of κ , as shown in Fig. 3.24: the reason for the discrepancy of high values of κ is due to the limited intensity gradients that occur at edges in grayscale images.

Overall, the problem of median shifts is now well understood, and is fully explained using the discrete model. The continuum model turns out to be capable of giving accurate results only in the limiting case where a and b ($= 1/\kappa$) are many pixels in size.

3.10 SHIFTS INTRODUCED BY MODE FILTERS

In this section we consider the shifts produced by mode filters in continuous images. As in the cases of median filters, straight edges with symmetrical profiles cannot be shifted by mode filters, because of symmetry. We proceed to the two paradigm cases—step edges and slant edges with circular boundaries. Again, the effects of noise will be ignored as we are considering the intrinsic rather than the noise-induced behavior of the mode filtering operation.

The situation for a curved step edge can again be understood by appealing to Fig. 3.15. The result for the mode also has to be identical to that for the median,

**FIGURE 3.24**

Comparisons of 3×3 median shifts. The lower solid curve shows the nonapproximated results of the discrete model (Davies, 1999f): the upper solid curve shows the results of experiments on grayscale circles.

Source: © IEE 1999

because the local intensity distribution is exactly symmetric and bimodal at the point where the median filter switches from a left-hand to a right-hand decision: at that point the mode must give the same result, since the median and the mode are coincident for a symmetric distribution. Hence, we conclude that the mode also gives a shift of $1/6 \kappa a^2$ for a curved step edge.

Next, we calculate edge shifts in the case where smoothly varying intensity functions exist—or within the confines of a small neighborhood—appear to be smoothly varying. In this case the calculation is especially simple (Davies, 1997a). Using the geometry of Fig. 3.17, we consider the intensity pattern within a circular neighborhood C. Of all the circular intensity contours appearing within C, the one possessing the most frequently occurring intensity, as selected by a mode filter, is the longest. Clearly, this is the one (M) whose ends are at opposite ends of a diameter of C. To estimate the shift in this case, all we need to do is to calculate the position of M, and determine its distance from the center of C. To proceed, we use the well-known formula relating the lengths of parts of intersecting chords of a circle, which gives:³

$$a^2 = D(2b - D) \approx 2Db \quad (3.17)$$

³This equation is a more general form of equation (3.16), as a can have any value; the proof follows similarly, on replacing a_0 by a in Fig. 3.23.

Table 3.3 Summary of Edge Shifts for Neighborhood Averaging Filters

Edge Type	Filter		
	Mean	Median	Mode
Step	$\frac{1}{6}\kappa a^2$	$\frac{1}{6}\kappa a^2$	$\frac{1}{6}\kappa a^2$
Intermediate	$\sim \frac{1}{7}\kappa a^2$	$\frac{1}{6}\kappa a^2$	$\frac{1}{2}\kappa a^2$
Linear	$\frac{1}{8}\kappa a^2$	$\frac{1}{6}\kappa a^2$	$\frac{1}{2}\kappa a^2$

© IEE 1999.

Hence:

$$D = \frac{1}{2}\kappa a^2 \quad (3.18)$$

i.e., there is a *right* shift of the contour, toward the local center of curvature, of $1/2\kappa a^2$. If we regard this set of contours as forming part of a grayscale edge profile, then the mode filter shifts the edge through $1/2\kappa a^2$ toward the center of curvature.

Some comments on the marked difference between the cases of step edges and linear intensity profiles are called for. This is all the more interesting as the median filter produces identical shifts, of $1/6\kappa a^2$, for the two profiles (see Table 3.3). In fact, of all the cases listed in Table 3.3, the outstanding one is the large shift for a mode filter operating on a linear intensity profile: what is special in this case is that the result relies on a single extreme contour length rather than an average of lengths amounting to an area measure. Hence, it is not surprising that the mode filter gives an exceptionally large shift in this case.

Finally, we note that edge shifts are not avoided merely by choosing an alternative method of neighborhood averaging, but rather that they are intrinsic to the averaging process, and can be avoided only by specially designed operators (e.g., see Greenhill and Davies, 1994a).

3.11 SHIFTS INTRODUCED BY MEAN AND GAUSSIAN FILTERS

In this section, we consider the shifts produced by mean and Gaussian filters in continuous images. As in the cases of median and mode filters, straight edges with symmetrical profiles cannot be shifted by mean and Gaussian filters, because of symmetry. We again consider the two paradigm cases—step edges and slant edges with circular boundaries. Again, we will ignore the effects of noise as we are considering the intrinsic rather than the noise-induced behavior of the filters.

The situation for a curved step edge can again be understood by appealing to Fig. 3.15, and is identical to that for the median and mode filters, and follows because of the symmetry of the local intensity distribution at the point where the filter switches from a left-hand to a right-hand decision: at that point the mean filter must give the same result, since all three statistics coincide for a symmetric distribution. Hence, it also gives a shift of $1/6 \kappa a^2$ for a curved step edge.

In the case of a smoothly varying slant edge, the result for the mean filter has to be calculated by integrating over the area of the neighborhood. The results cannot be obtained by intuitive or simple geometric or intuitive arguments, and here we merely quote the shift for the mean filter as being $1/8 \kappa a^2$.

These considerations complete the entries in Table 3.3. The results for Gaussian filters do not differ substantially from those for mean filters, but have to be obtained by integration, taking account of the Gaussian weighting function (Davies, 1991b). A general point is that all such filters have similar shifting effects because they all incorporate a measure of signal averaging: the shifting effect is not avoided simply by employing a different central-seeking statistic to perform the averaging.

3.12 SHIFTS INTRODUCED BY RANK ORDER FILTERS

This section is particularly concerned with rank order filters (Bovik et al., 1983), which form a whole family of filters that can be applied to digital images—often in combination with other filters of the family—in order to give a variety of effects (Goetcherian, 1980; Hodgson et al., 1985): other notable members of the family are max and min filters. Because rank order filters generalize the concept of the median filter, it is relevant to study the types of distortion they produce on straight and curved intensity contours. It should also be pointed out that these filters are of central importance in the design of filters for morphological image analysis and measurement. In addition, it has been pointed out that they have some advantages when used for this purpose in that they help to suppress noise (Harvey and Marshall, 1995) (although the effect vanishes in the special cases of max and min filters).

Section 3.12.1 examines the reasons underlying the shifts produced by rank order filters and makes calculations of their extent for rectangular neighborhoods. It then generalizes these results to circular neighborhoods and goes on to examine the extent to which the theoretical predictions are borne out in practice by measurements of the shifts produced by 5×5 rank order filters on circular disks of varying sizes. It will be taken as axiomatic that the application of rank order filters produces edge shifts on real images (they are well attested in the case of max, min, and median filters): the main question to be answered here is the exact numerical extent of these shifts and how they may be modeled for general rank order filters.

3.12.1 Shifts in Rectangular Neighborhoods

In common with previous work in this area we here concentrate on the ideal noiseless case, in which the filter operates within a small neighborhood, over which the signal is basically a monotonically increasing intensity function in some direction. The most complex intensity variation that will be considered is that in which the intensity contours are curved with curvature κ . In spite of this simplified configuration, valuable statements can still be made about the level of distortion likely to be produced in practice by rank order filters.

Because of the complexity of the calculations that arise in the case of rank order filters, which involve an additional parameter *vis-à-vis* the median filter, it is worth studying their properties first for the simple case of rectangular neighborhoods (Davies, 2000f). Let us presume that a rank order filter is being applied in a situation in which straight intensity contours are aligned parallel to the short sides of a rectangular neighborhood that we initially take to be a $1 \times n$ array of pixels (Fig. 3.25). In this case, we can assume without loss of generality that the successive pixels within the neighborhood will have *increasing* values of intensity. We next take the basic property of the rank order filter (effectively or in fact) to construct an intensity histogram of the local intensity distribution and return the value of the r th of the n intensity values within the neighborhood. This means that the rank order filter selects an intensity that has physical separation B from the lowest intensity pixel of the neighborhood and C from the highest intensity pixel, where:

$$B = r - 1 \quad (3.19)$$

$$C = n - r \quad (3.20)$$

$$A = B + C = n - 1 \quad (3.21)$$

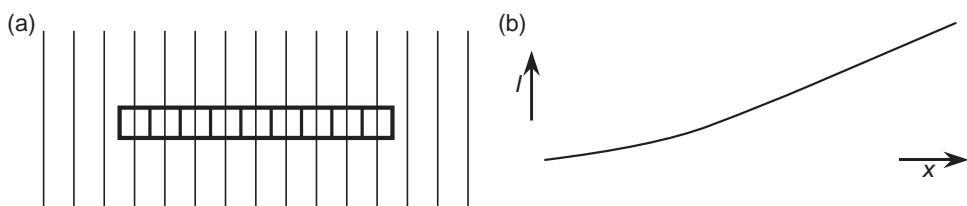


FIGURE 3.25

Basic situation for a rank order filter in a rectangular neighborhood. This figure illustrates the problem of applying a rank order filter within a rectangular neighborhood consisting of a $1 \times n$ array of pixels. The intensity is taken to increase monotonically from left to right, as in (b); the intensity contours in (a) are taken to be parallel to the short sides of the neighborhood.

Source: © RPS 2000

Table 3.4 Properties of the Three Paradigm Filters

Filter	r	η	B	C	D
Median	$\frac{1}{2}(n+1)$	0	$\frac{1}{2}A$	$\frac{1}{2}A$	0
Max	n	-1	A	0	$-\frac{1}{2}(n-1)$
Min	1	1	0	A	$\frac{1}{2}(n-1)$

© RPS 2000.

These definitions underline that a rank order filter will in general produce a D -pixel shift, whose value is:

$$D = \frac{1}{2}(n+1) - r \quad (3.22)$$

Before proceeding further, it will be useful to introduce a parameter η that is more symmetric than r , and has value +1 at $r=1$ and -1 at $r=n$:

$$\eta = (n - 2r + 1)/(n - 1) \quad (3.23)$$

Using this parameter in preference to r , we can write down new formulae for B , C , D :

$$B = \frac{1}{2}A(1 - \eta) \quad (3.24)$$

$$C = \frac{1}{2}A(1 + \eta) \quad (3.25)$$

$$D = \frac{1}{2}\eta(n - 1) \quad (3.26)$$

The properties of the three paradigm filters are summarized in [Table 3.4](#) in terms of these parameters.

We now proceed to a continuum model, assuming a large number of pixels in any neighborhood (i.e., $n \rightarrow \infty$). The main difference will be that we shall specify distance in terms of the half-length a of the neighborhood rather than in terms of numbers of pixels:

$$D = \eta a \quad (3.27)$$

Next note that this formulation is independent of the width of the neighborhood, so long as the latter is rectangular. We now generalize the situation by taking the neighborhood to be rectangular and of dimensions $2a$ by $2\tilde{a}$ ([Fig. 3.26](#)).

The next task is to determine the result of a curvature $\kappa = 1/b$ in the intensity contours. Here we approximate the equation of a circle of radius b , with its diameter on the positive x -axis and passing through the origin, as:

$$x = \frac{y^2}{2b} \quad (3.28)$$

We can integrate the area under an intensity contour (see Fig. 3.26) as follows:

$$\begin{aligned} K &= \int_{-\tilde{a}}^{\tilde{a}} x \, dy = \left(\frac{1}{2b} \right) \int_{-\tilde{a}}^{\tilde{a}} y^2 \, dy = \left(\frac{1}{2b} \right) \left[\frac{y^3}{3} \right]_{-\tilde{a}}^{\tilde{a}} \\ &= \frac{\tilde{a}^3}{3b} = \frac{1}{3} \kappa \tilde{a}^3 \end{aligned} \quad (3.29)$$

We deduce that the shift D is given by:

$$B = 2\tilde{a}(a - D) + \frac{1}{3} \kappa \tilde{a}^3 \quad (3.30)$$

$$C = 2\tilde{a}(a + D) - \frac{1}{3} \kappa \tilde{a}^3 \quad (3.31)$$

$$\eta = \frac{(C - B)}{A} = \frac{4\tilde{a}D - (2/3)\kappa \tilde{a}^3}{A} \quad (3.32)$$

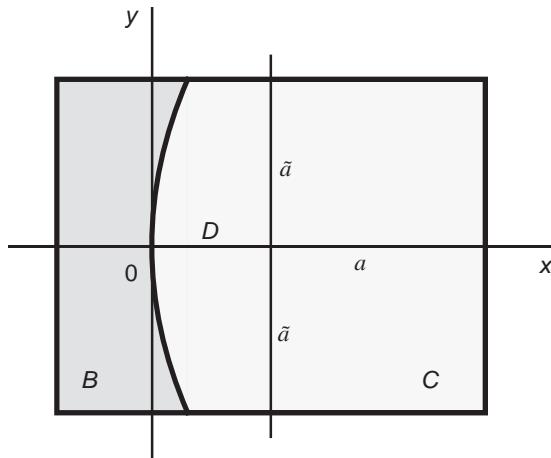


FIGURE 3.26

Geometry of a rectangular neighborhood with curved intensity contours. Here the neighborhood is a general rectangular neighborhood of dimensions $2a \times 2\tilde{a}$. Again, the intensity is taken to increase monotonically from left to right; the intensity contours are taken to be parallel and in this case are curved with identical curvature κ . x and y axes needed for area calculations are also shown. B and C represent the areas of the two shaded regions on either side of the thick intensity contour.

Source: © RPS 2000

where

$$A = 4a\tilde{a} \quad (3.33)$$

Hence:

$$D = \frac{\eta A}{4\tilde{a}} + \frac{1}{6}\kappa\tilde{a}^2 = \eta a + \frac{1}{6}\kappa\tilde{a}^2 \quad (3.34)$$

What is important about this equation is that it shows that the effects of rank order and curvature can be calculated and summed separately, the first term being that obtained above for the case of zero curvature and the second term being exactly that calculated for a median filter when the intensity contour is of length $2\tilde{a}$ (the earlier calculation (Davies, 1989b) related to a circular neighborhood). Thus, in principle we merely need to recompute the first term for any appropriate shape of neighborhood. However, various complications arise, particularly in the case of high curvature contours. These have been dealt with successfully, with the results shown in Figs. 3.27 and 3.28 (Davies, 2000f).

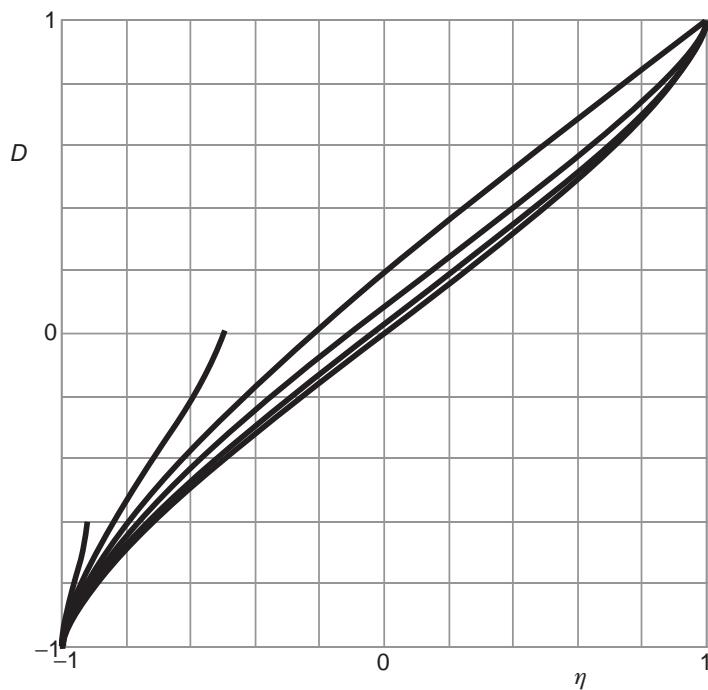


FIGURE 3.27

Graphs of shift D against rank order parameter η for various κ . This diagram summarizes the operation of rank order filters, with graphs, bottom to top, respectively, for $\kappa = 0, 0.2/a, 0.5/a, 1/a, 2/a, 5/a$. Note that graphs for which $b < a$ ($\kappa > 1/a$) apply for restricted ranges of η and D (see Section 3.12.1). A multiplier of a must be included in the D -values.

Source: © RPS 2000

As the above theory is based on a continuum model, it is not perfect, and it does indicate only the main features of the practical situation. In particular, when the curvatures are very high, they may arise from spots that are entirely within the neighborhood, and then there is the possibility that they will be completely eliminated by the rank order filter (note that noise points are entirely eliminated by a median filter, which indeed is the prime practical use of that type of filter). Correspondingly, the assumptions made in the model break down when there is no intersection of the circular neighborhood and the intensity contour of radius $b = 1/\kappa$.

Some of the conclusions of this work are quite important. In particular, the result for a median filter is the special case that arises when $\eta = 0$ and is in agreement

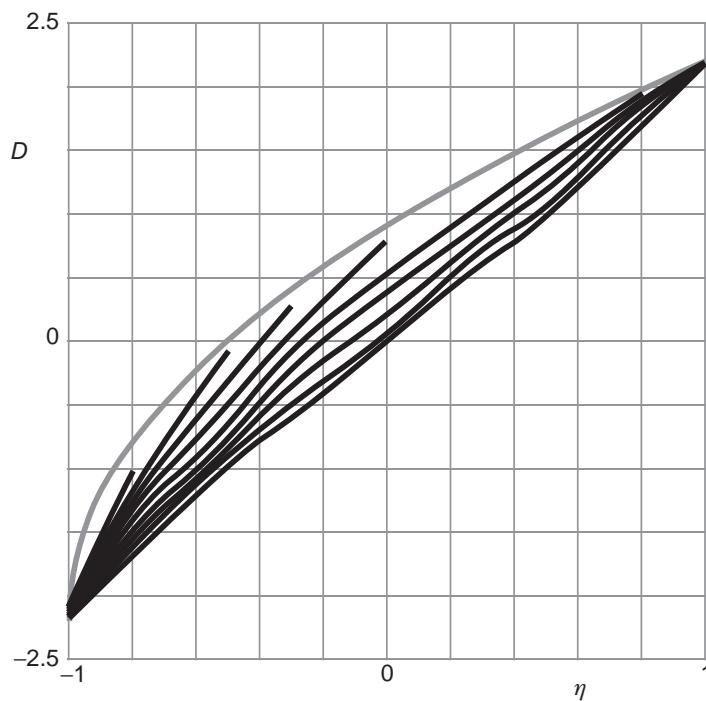


FIGURE 3.28

Shifts obtained for a typical discrete neighborhood. These shifts were obtained for rank order filters operating within a truncated 5×5 neighborhood when applied to eight discrete circular disks with radii ranging from 10.0 down to 1.25 pixels, the mean curvatures being 0.1–0.8 in steps of 0.1; the lowest curve was obtained by averaging the responses from circular disks of radius ± 20.0 pixels, with curvatures ± 0.05 , and to the given scale are indistinguishable from the result that would be obtained with zero curvature. The uppermost curve represents the theoretical limiting value. However, because of the directional effects that occur in the discrete case, the upper limit is actually lower than indicated by this curve (see text).

Source: © RPS 2000

with the calculations of [Section 3.8](#). Next, the max and min filters are also special cases and occur for $\eta = -1$ and 1 , respectively. In these limiting cases, the shifts are $D = -a$ and a , respectively, the results being independent of κ : this is as might be expected since the value of \tilde{a} is zero in each case. Between the max and min filters and the median filter, there is a continuous gradation of performance, with very significant but opposite shifts for the max and min filters, and the two basic effects canceling out for median filters—although the cancellation is only exact for straight contours. The full situation is summarized in [Fig. 3.27](#).

3.13 THE ROLE OF FILTERS IN INDUSTRIAL APPLICATIONS OF VISION

It has been shown above how the median filter can successfully remove noise and artifacts such as spots and streaks from images. Unfortunately, many useful features such as fine lines and important points and holes are effectively indistinguishable from spots and streaks. In addition, it has been seen that the median filter “softens” pictures by removing fine detail. It is also found to clip corners of objects—another generally undesirable trait (but see Chapter 6). Finally, although it does not blur edges, it can still shift them slightly. In fact, shifting of curved edges seems to be a general characteristic of noise suppression filters.

Such distortions are quite alarming and mitigate against the indiscriminate use of filters. If applied in situations where accurate measurements are to be made on images, particular care must be taken to test whether the data are being biased in any way. Although it is possible to make suitable corrections to the data, it seems a good general policy to employ noise removal filters only where they are absolutely essential for object visibility. The alternative is to employ edge detection and other operators that automatically suppress noise as an integral part of their function. This is the general approach taken in subsequent chapters: indeed, it is one of the principles underlined in this book that algorithms should be “robust” against noise or other artifacts that might upset measurements. There is quite significant scope for the design of robust algorithms, since images contain so much information that it is normally possible to arrange for erroneous information to be ignored.

3.14 COLOR IN IMAGE FILTERING

In Chapter 2, it was indicated that color often adds to the complexity of image analysis algorithms, and could also add to the associated computational costs. From these points of view color might, except for applications such as assessing the ripeness of fruit, be regarded as an irrelevant luxury. Nevertheless, in the field of image processing and image filtering, where good quality images have to be

presented to human operators, it is a vital concern. In fact, in recent years much effort has been devoted to the development of effective color filtering algorithms. Here we shall consider mainly median and related impulse noise filtering procedures.

Perhaps the first point to note is that median filtering is defined in terms of sorting operations and is thus undefined in the color domain, which normally contains three dimensions. However, a simple solution is to apply a standard median filter to each of the color channels, and then to reassemble the color image. Unfortunately, this approach leads to certain problems, the most obvious one being that of color “bleeding” (Fig. 3.12). This occurs when an impulse noise point appears in just one of the channels and is situated near an edge or other image feature. The case of an impulse noise point near an edge is hereby illustrated in simplified form:

$$\begin{array}{ll} \text{Original:} & 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \\ \text{Filtered:} & ? \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ ? \end{array}$$

We see that a 3-element median filter eliminates the impulse noise point but at the same time moves the edge toward it. The end result for a color image is that the edge will be tinted with the color of the impulse noise point.

Fortunately, there is a standard solution to this problem. First, note that it is possible to express single-channel median filtering as the minimization of a distance metric, and this metric is trivially extendible to three color channels (or indeed any number of channels). The relevant single channel metric is:⁴

$$\text{median} = \arg \min_i \sum_j |d_{ij}| \quad (3.35)$$

where d_{ij} is the distance between sample points i and j in the single-channel (gray scale) space. In the three-color domain, the metric is readily extended to:

$$\text{median} = \arg \min_i \sum_j |\tilde{d}_{ij}| \quad (3.36)$$

where \tilde{d}_{ij} is the generalized distance between sample points i and j , and we typically take the L_2 norm to define the distance measure for three colors:

$$\tilde{d}_{ij} = \left[\sum_{k=1}^3 (I_{i,k} - I_{j,k})^2 \right]^{1/2} \quad (3.37)$$

Here \mathbf{I}_j , \mathbf{I}_i are RGB vectors and $I_{i,k}$, $I_{j,k}$ ($k = 1, 2, 3$) are their color components.

⁴“arg min” is a standard mathematical term that means the argument (here pixel intensity) corresponding to the index (here i) giving rise to the minimum value of the expression in Eq. (3.36) (here $\sum_j |d_{ij}|$).

While the resulting vector median filter (VMF) no longer treats the individual color components separately, it is by no means guaranteed to eliminate color bleeding completely. In fact, like the standard median, it replaces any noisy intensity I_n , (including color) by the intensity I_j of another pixel that exists in the same window—rather than by an ideal intensity I . Hence, color bleeding is only reduced, but not eliminated. If indeed there is a confluence of colors at any one point in an image, even in the absence of any impulse noise there is the possibility that these sorts of algorithms will become confused and inadvertently introduce small amounts of color bleeding: ultimately, the effect is due to the increased dimensionality of the data, which means that the algorithm has to contend with a greatly increased number of possible outcomes in spite of being an *ad hoc* procedure that does not embody specific understanding of images.

[Figure 3.12](#) demonstrates the nature of color bleeding, albeit in the case of mode filtering: this figure shows vector median and vector mode filters to be remarkably free from color bleeding, but the same does not apply to scalar mode filters—for similar reasons to those indicated above for median filters.

3.15 CONCLUDING REMARKS

Although this chapter has dwelt on the implementation of noise suppression and image enhancement operators based on the local intensity distribution, it has made certain other points. In particular, it has shown the need to make a specification of the required imaging process and only then to work out the algorithm design strategy. Not only does this ensure that the algorithm will perform its function effectively, but also it should make it possible to optimize the algorithm for various practical criteria including speed, storage, and other parameters of interest. In addition, this chapter has demonstrated that any undesirable properties of the particular design strategy chosen (such as the inadvertent shifting of edges) should be sought and dealt with. Next, it has demonstrated a number of fundamental problems to do with imaging in discrete lattices—not least being problems of statistics that arise with small pixel neighborhoods. Finally, the large edge shifts of certain types of rank order filter are particularly important because they are turned to advantage in morphological operators (Chapter 7).

The next chapter moves on to a particularly vital problem in machine vision—that of segmenting images in order to find where objects are situated. This work builds on what has been learnt in the present chapter about edge profiles and how they are “seen” by neighborhood operators.

Median filters have long been used to eliminate impulse noise without blurring edges. However, this chapter has shown that significant shifting of edges can result from use of median filters, and this property extends to mode filters and a fortiori to rank order filters—so much so that the latter form the basis of morphological processing.

3.16 BIBLIOGRAPHICAL AND HISTORICAL NOTES

Much of the work of this chapter has built on a paper by the author (Davies, 1988c), which rests on considerable earlier work on Gaussian, median, and other rank order filters (Hodgson et al., 1985; Duin et al., 1986). Note that the edge shifts that occur for median filters are not limited to this type of filter but apply almost equally to mean filters (Davies, 1991b). In addition, other inaccuracies have been found with median filters and methods have been found to correct them (Davies, 1992e).

The early literature hardly mentions mode filters, presumably because of the difficulty of finding simple mode estimators that are not unduly confused by noise and which still operate rapidly. Indeed, only one early reference has been found (Coleman and Andrews, 1979), although it has been backed up by later work (e.g., Evans and Nixon, 1995; Griffin, 2000). Other work referred to here is that on decomposing Gaussian and median filters (Narendra, 1978; Wiejak et al., 1985), and the many papers on fast implementation of median filters (e.g., Narendra, 1978; Huang et al., 1979; Danielsson, 1981; Davies, 1992a).

Considerable efforts have been devoted to studying the “root” behavior of the median filter, i.e., the result of applying median filtering operations until no further change occurs. In fact, much of this work has been carried out on 1-D signals, including cardiac and speech waveforms, rather than on images (Gallagher and Wise, 1981; Fitch et al., 1985; Heinonen and Neuvo, 1987). Root behavior is of interest as it relates to the underlying structure of signals, although its realization involves considerable amounts of processing. Some of the work on filtering aims to improve on rather than to emulate the median filter. Work of this type includes the detail-preserving filters of Heinonen and others (Niemenen et al., 1987) and relates to the lower set of plots in Fig. 3.18. See also the neural network approach to this topic (e.g., Greenhill and Davies, 1994a). More recent work on nonlinear filtering appears in Marshall et al. (1998); see Marshall (2004) for a new design method for weighted order statistics filters.

The author (Davies, 1987c) has reported methods of optimizing linear smoothing filters in small neighborhoods by minimizing the total error in fitting them to a continuous Gaussian function: a balance has to be struck between subpixel errors within the neighborhood and errors that arise from the proportion of the distribution that lies outside the neighborhood (Fig. 3.29).

With the advent of extremely low cost color frame grabbers on PCs, and the widespread use of digital cameras, digital color images have become ubiquitous, and this has extended to (or even necessitated) much research on color filtering. A useful summary of work in this area up to 1998 appears in Sangwine and Horne (1998). More recent work on vector (color) filtering includes that of Lukac (2003). Charles and Davies (2003b) describe new distance-weighted median filters and their application to color images. They also extend the author’s earlier mode filter work to color images (Charles and Davies, 2003a, 2004). Davies’s (2000e) theorem shows that restricting a multichannel (color) filter output to the

FIGURE 3.29

Approximating a discrete to a continuous Gaussian. This diagram shows how a balance needs to be struck between subpixel errors and those arising from the truncated part of the function.

vector value of one of the input sample points (i.e., from the current window in the image) will increase the inaccuracy present in the final image, for a large proportion of pixels: since this represents the usual vector median strategy that is employed to minimize color bleeding, the effectiveness of the current generation of color filter algorithms needs to be looked at further.

Davies has further analyzed the distortions and edge shifts produced by a range of rank order, mean, and mode filters, and has produced a unified review of the subject (Davies, 2003e). In the case of median filters, it proved possible, and necessary for high accuracy, to produce a discrete model of the situation (Davies, 2003c), rather than extending the continuum model described much earlier (Davies, 1989b).

3.16.1 More Recent Developments

The 2000s have seen a new approach to filtering via “switched” types of filter that judge whether or not any pixel is corrupted by impulse noise: if the latter, they use a method such as the median or vector-median filter to eliminate it; if the former, they adopt a policy of zero change by using the original pixel intensity or color. The zero change policy is useful because it helps maintain image sharpness and fidelity. An early example of this approach was the work of Eng and Ma (2001): see Chen et al. (2009) and Smolka (2010) for recent, more sophisticated versions of this concept (Smolka’s version falls in the category of a “peer group switching filter”).

Davies (2007b) has studied the properties of the generalized (nonvector) median filter that has the capability for eliminating even more noise than the VMF while not being targeted so specifically at eliminating color bleeding. He demonstrates ways of implementing the filter so that it runs sufficiently rapidly to make it a viable alternative to the VMF.

Celebi (2009) has shown how to reduce the computational needs of directional vector filters based on order statistics without significant loss of accuracy. At another end of the scale, Rabbani and Gazor (2010) have shown how to reduce

additive Gaussian noise by using local mixture models; they find that of the wavelet types of local representation, the discrete complex wavelet transform is preferable in terms of both peak noise performance and computational cost.

3.17 PROBLEMS

1. Draw up a table showing the numbers of operations required to implement a median filter in various sizes of the neighborhood. Include in your table (i) results for a straight bubble sort of all n^2 pixels, (ii) results for bubble sorts in separated $1 \times n$ and $n \times 1$ neighborhoods, and (iii) results for the histogram method of [Section 3.3](#). Discuss the results, taking account of possible computational overheads.
2. Show how to perform a median filtering operation on a binary image. Show also that if a set of binary images is formed by thresholding a grayscale image at various levels, and each of these binary images is median filtered, then a grayscale image can be reconstructed that is a median filtered version of the original grayscale image. Consider to what extent the reduced amount of computation in filtering a binary image compensates for the number of separate thresholded images to be filtered.
3. An “extremum” filter is an image-parallel operation that assigns every pixel the intensity value closer to the two extreme values in its local intensity distribution. Show that it should be possible to use such a filter to enhance images. What would be the *disadvantage* of such a filter?
4. Under what conditions is a 1-D signal that has been filtered once by a median filter a root signal? What truth is there in the statement that a straight edge in an image is neither shifted nor blurred by a median filter, whatever its cross-section?
5. **a.** Explain the action of the following median filtering algorithm:

```

for all pixels in image do{
    for (i = 0; i <= 255; i++) hist[i] = 0;
    for (m = 0; m <= 8; m++) hist[P[m]]++;
    i = 0; sum = 0;
    while (sum < 5) {
        sum = sum + hist[i];
        i = i + 1;
    }
    Q0 = i - 1;
}

```

- b.** Show how this algorithm can be speeded up (i) by a more efficient histogram clearing technique and (ii) by calculating the minimum intensity in each 3×3 window. In each case, estimate approximately how much the algorithm will be speeded up.
- c.** Explain why a median filter is able to smooth images without introducing blurring.

- d. A 1-D cross-section of an image has the following intensity profile:

1 2 1 1 2 3 0 2 2 3 1 1 2 2 9 2 2 8 8 8 8 7 8 7 9 9 9 9

Apply (i) a 3-element median filter and (ii) a 5-element median filter to this profile. With the aid of these examples, show that median filters tend to produce “runs” of constant values in 1-D profiles. Show also that under some circumstances an edge in the profile can be shifted by a nearby spike: give a rule showing when this is likely to occur for an n -element median filter in one dimension.

6. a. A *mode* filter is defined as one in which the new pixel intensity at any pixel takes the most probable value in the local intensity distribution of a window placed around that pixel in the original image space. Show for a grayscale image that a mode filter will, if anything, sharpen the image, while a *mean* filter will tend to blur the image.

b. A *max* filter is one that takes the maximum value of the local intensity distribution in a window around each pixel. Explain what will be seen when a max filter is applied to an image. Consider whether any similar effects are liable to happen when a mode filter is applied to an image.

c. Explain the purpose of a *median* filter. Why are 2-D median filters sometimes implemented as two 1-D median filters applied in sequence?

d. Contrast the behavior of 5-element 1-D mean, max, and median filters as applied to the following waveform:⁵

0 1 1 2 3 2 2 0 2 3 9 3 2 4 4 6 5 6 7 0 8 8 9 1 1 8 9

- e. Work out what would happen if the 1-D median filter were applied many times, starting with this waveform.

- 7. a.** Determine the effect of applying (i) a 3×3 median filter and (ii) a 5×5 median filter to the portion of an image shown in Fig. 3.30.

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	0	0	9	9	9	9	9
0	0	0	0	9	9	8	9	9
0	0	1	0	9	8	9	9	7
0	0	0	0	7	9	9	8	9
0	1	0	8	9	9	9	9	9
0	0	0	0	9	9	9	9	9
0	0	0	0	8	9	9	9	9

FIGURE 3.30

Portion of image for tests of median filter.

⁵For the mean filter, give the nearest integer value in each case.

- b. Show that it should be possible to develop a corner detector based on the properties of these median filters. What advantages or disadvantages might result from employing this design strategy?
8. a. Distinguish between *mean* and *median filtering*. Explain why a mean filter would be expected to blur an image, while a median filter would not have this effect. Illustrate your answer by showing what happens in the following 1-D case with a window of size 1×3 :

1 1 1 1 2 1 1 2 3 4 4 0 4 4 4 5 6 7 6 5 4 3 3

- b. Give a complete median filter algorithm based on histograms and operating within a 3×3 window. Explain why it operates relatively slowly.
- c. A computer language has the $\max(a, b)$ operation as standard. Show how it may be used to find the maximum intensity within a 3×3 window. Show also how it may be used to find the median by successively replacing the maximum values by zeros. If the $\max(a, b)$ operation is about the same speed as the $a + b$ operation, determine whether the median can be found any faster by this method.
- d. Discuss whether splitting a 3×3 median operation into 1×3 and 3×1 median operations is likely to be effective at eliminating impulse noise in images. How would the speed of this approach be affected by use of the $\max(a, b)$ operation?
9. a. Determine the result of applying a 3-element median filter to the following 1-D signals:

- i. 0 0 0 0 0 1 0 1 1 1 1 1 1
ii. 2 1 2 3 2 1 2 2 3 2 4 3 3 4
iii. 1 1 2 3 3 4 5 8 6 6 7 8 9 9

- b. What general lessons can be learnt from the results? In the first case, consider also the corresponding situation for a grayscale edge in a 2-D image.
- c. 2-D median filters are sometimes implemented as two 1-D median filters applied in sequence in order to improve the speed of processing. Estimate the gain in speed that could be achieved in this way for (i) a 3×3 median filter, (ii) a 7×7 median filter, and (iii) in the general case.

4

Thresholding Techniques

One of the important practical aims of image processing is the demarcation of objects appearing in digital images. This process is called segmentation, and a good approximation to it can often be achieved by thresholding. Broadly, this involves separating the dark and light regions of the image, and thus identifying dark objects on a light background (or vice versa). This chapter discusses the effectiveness of this idea and the means for achieving it.

Look out for:

- the segmentation, region-growing and thresholding concepts.
- the problem of threshold selection.
- the limitations of global thresholding.
- problems in the form of shadows or glints (highlights).
- the possibility of modeling the image background.
- the idea of adaptive thresholding.
- the rigorous Chow and Kaneko approach.
- what can be achieved with simple local adaptive thresholding algorithms.
- more thoroughgoing variance, entropy-based, and maximum likelihood methods.
- the possibility of modeling images by multilevel thresholding.
- the value of the global valley transformation.
- how thresholds can be found in unimodal distributions.

Thresholding is limited in what it can achieve, and there are severe difficulties in automatically estimating the optimum threshold—as evidenced by the many available techniques that have been devised for the purpose. In fact, segmentation is an ill-posed problem, and it is misleading that the human eye appears to perform thresholding reliably. Nevertheless, there are instances where the task can be simplified, for example, by suitable lighting schemes, so that thresholding becomes effective. Hence, it is a useful technique that needs to be included in the toolbox of available algorithms for use when appropriate. However, edge detection (Chapter 5) provides an alternative highly effective means to key into complex image data.

4.1 INTRODUCTION

One of the first tasks to be undertaken in vision applications is to segment objects from their backgrounds. When objects are large and do not possess very much surface detail, segmentation can be imagined as splitting the image into a number of regions each having a high level of uniformity in some parameter such as brightness, color, texture or even motion. Hence, it should be straightforward to separate objects from one another and from their background, and also to discern the different facets of solid objects such as cubes.

Unfortunately, the concept of segmentation presented above is an idealization that is sometimes reasonably accurate, but more often in the real world, it is an invention of the human mind, generalized inaccurately from certain simple cases. This problem arises because of the ability of the eye to understand real scenes at a glance, and hence to segment and perceive objects within images in the form they are known to have. Introspection is not a good way of devising vision algorithms, and it must not be overlooked that segmentation is actually one of the central and most difficult practical problems of machine vision.

Thus, the common view of segmentation as looking for regions possessing some degree of uniformity is to a large extent invalid. There are many examples of this in the world of 3-D objects: one is a sphere lit from one direction, the brightness in this case changes continuously over the surface so that there is no distinct region of uniformity; another is a cube where the direction of the lighting may lead to several of the facets having equal brightness values so that it is impossible from intensity data alone to segment the image completely as desired.

Nevertheless, there is sufficient correctness in the concept of segmentation by uniformity measures for it to be worth pursuing for practical applications. The reason is that in many (especially industrial) applications, only a very restricted range and number of objects are involved, and in addition it is possible to have almost complete control over the lighting and the general environment. The fact that a particular method may not be completely general need not be problematic, since by employing tools that are appropriate for the task in hand, a cost-effective solution will have been achieved in that case at least. However, in practical situations, there is clearly a tension between simple cost-effective solutions and general-purpose but more computationally expensive solutions; this tension must always be kept in mind in severely practical subjects such as machine vision.

4.2 REGION-GROWING METHODS

The segmentation idea outlined in Section 4.1 leads naturally to the region-growing technique (Zucker, 1976b). Here, pixels of like intensity (or other suitable property) are successively grouped together to form larger and larger

regions until the whole image has been segmented. Clearly, there have to be rules about not combining adjacent pixels that differ too much in intensity, while permitting combinations for which intensity changes gradually because of variations in background illumination over the field of view. However, this is not enough to make a viable strategy, and in practice the technique has to include the facility not only to merge regions together but also to split them if they become too large and inhomogeneous (Horowitz and Pavlidis, 1974). Particular problems are noise and sharp edges and lines that form disconnected boundaries, and for which it is difficult to formulate simple criteria to decide whether they form true region boundaries. In remote sensing applications, for example, it is often difficult to separate fields rigorously when hedges are broken and do not give continuous lines: in such applications, segmentation may have to be performed interactively, with a human operator helping the computer. Hall (1979) found that in practice regions tend to grow too far,¹ so that to make the technique work well it is necessary to limit their growth with the aid of edge detection schemes.

Thus, the region-growing approach to segmentation turns out to be quite complex to apply in practice. In addition, region-growing schemes usually operate iteratively, gradually refining hypotheses about which pixels belong to which regions. The technique is complicated because, carried out properly, it involves global as well as local image operations. Thus, each pixel intensity will in principle have to be examined many times, and as a result the process tends to be quite computation intensive. For this reason, it is not considered further here, since we are often more interested in methods involving low computational load that are amenable to real-time implementation.

4.3 THRESHOLDING

If background lighting is arranged so as to be fairly uniform, and we are looking for rather flat objects that can be silhouetted against a contrasting background, segmentation can be achieved simply by thresholding the image at a particular intensity level. This possibility was apparent from Fig. 2.2. In such cases, the complexities of the region-growing approach are bypassed. The process of thresholding has already been covered in Chapter 2, the basic result being that the initial grayscale image is converted into a binary image in which objects appear as black figures on a white background, or as white figures on a black background. Further analysis of the image then devolves into analysis of the shapes and dimensions of the figures: at this stage, object identification should be straightforward. Chapter 9 concentrates on such tasks. Meanwhile, there is one outstanding problem—how to devise an automatic procedure for determining the optimum thresholding level.

¹Clearly, there is a danger that even one small break could join two regions into a single larger one.

4.3.1 Finding a Suitable Threshold

One simple technique for finding a suitable threshold arises in situations such as optical character recognition (OCR) where the proportion of the background that is occupied by objects (i.e., print) is relatively constant in a variety of conditions. A preliminary analysis of relevant picture statistics then permits subsequent thresholds to be set by insisting on a fixed proportion of dark and light in a sequence of images (Doyle, 1962). In practice, a series of experiments is performed in which the thresholded image is examined as the threshold is adjusted, and the best result ascertained by eye: at that stage, the proportions of dark and light in the image are measured. Unfortunately, any changes in noise level following the original measurement will upset such a scheme, since they will affect the relative amounts of dark and light in the image. However, this is frequently a useful technique in industrial applications, especially when particular details within an object are to be examined: typical examples of this are holes in mechanical components such as brackets (note that the mark–space ratio for objects may well vary substantially on a production line, but the proportion of hole area *within* the object outline would not be expected to vary).

The technique that is most frequently employed for determining thresholds involves analyzing the histogram of intensity levels in the digitized image (Fig. 4.1): if a significant minimum is found, it is interpreted as the required threshold value (Weska, 1978). Clearly, the assumption being made here is that the peak on the left of the histogram corresponds to dark objects, and the peak on the right corresponds to light background (here it is assumed that, as in many industrial applications, objects appear dark on a light background).

This method is subject to the following major difficulties:

1. the valley may be so broad that it is difficult to locate a significant minimum.
2. there may be a number of minima because of the type of detail in the image, and selecting the most significant one will be difficult.
3. noise within the valley may inhibit location of the optimum position.
4. there may be no clearly visible valley in the distribution because noise may be excessive or because the background lighting may vary appreciably over the image.
5. either of the major peaks in the histogram (usually due to the background) may be much larger than the other and this will then bias the position of the minimum.
6. the histogram may be inherently multimodal, making it difficult to determine which is the relevant thresholding level.

Perhaps the worst of these problems is the last point: that is, if the histogram is inherently multimodal, and we are trying to employ a single threshold, then we are applying what is essentially an *ad hoc* technique to obtain a meaningful result. In general, such efforts are unlikely to succeed, and this is clearly a case where full image interpretation must be performed before we could be sure that the results are valid. Ideally, thresholding rules have to be formed after many images

FIGURE 4.1

Idealized histogram of pixel intensity levels in an image. The large peak on the right results from the light background; the smaller peak on the left is due to dark foreground objects. The minimum of the distribution provides a convenient intensity value to use as a threshold.

have been analyzed. In what follows such problems of meaningfulness are eschewed and attention is concentrated on how best to find a genuine single threshold when its position is obscured as indicated by problems 1–5 above (which can be ascribed to image “clutter,” noise, and lighting variations).

4.3.2 Tackling the Problem of Bias in Threshold Selection

This section considers problem 5 of Section 4.3.1—that of eliminating the bias in the selection of thresholds that arises when one peak in the histogram is larger than the other. First, note that if the relative heights of the peaks are known, this effectively eliminates the problem, since the “fixed proportion” method of threshold selection outlined above can be used. However, this is not normally possible. A more useful approach is to prevent bias by weighting down the extreme values of the intensity distribution and weighting up the intermediate values in some way. To achieve this, note that the intermediate values are special in that they correspond to object edges. Hence, a good basic strategy is to find positions in the image where there is a significant intensity gradient—corresponding to pixels in the regions of edges—and to analyze the intensity values of these locations while ignoring other points in the image.

One way of dealing with this is to construct “scattergrams” in which pixel properties are plotted on a 2-D map with intensity variation along one axis and

**FIGURE 4.2**

Scattergram showing the frequency of occurrence of various combinations of pixel intensity I and intensity gradient magnitude g in an idealized image. There are three main populated regions of interest: (1) a low- I , low- g region; (2) a high- I , low- g region; and (3) a medium- I , high- g region. Analysis of the scattergram sometimes provides useful information on how to segment the image.

intensity gradient magnitude variation along the other. As indicated in Fig. 4.2, there are three main populated regions on the map: (1) a low-intensity, low-gradient region corresponding to the dark objects; (2) a high-intensity, low-gradient region corresponding to the background; and (3) a medium-intensity, high-gradient region corresponding to object edges (Panda and Rosenfeld, 1978). By analyzing how these regions merge into each other, it is sometimes possible to obtain better results than can be obtained using simple thresholding. In particular, by examining the situation for moderate values of gradient, bias may be reduced, as indicated above. However, instead of constructing a scattergram, we can try weighting the plots in the intensity histogram in such a way as to minimize threshold bias: this possibility is discussed in the following section.

4.3.2.1 Methods Based on Finding a Valley in the Intensity Distribution

This section considers how to weight the intensity distribution using a parameter other than the intensity gradient, in order to locate accurately the valley in the intensity distribution. A simple strategy is first to locate all pixels that have a significant intensity gradient, and then to find the intensity histogram not only of these pixels but also of nearby pixels. This means that the two main modes in the intensity distribution are still attenuated very markedly and hence the bias in the valley position is significantly reduced. Indeed, the numbers of background and foreground pixels that are now being examined are very similar, so the bias

from the relatively large number of background pixels is virtually eliminated (note that if the modes are modeled as two Gaussian distributions of equal widths and they also have equal heights, then the minimum lies exactly halfway between them).

Although obvious, this approach clearly includes the edge pixels themselves, which tend to fill the valley between the two modes. For the best results, the points of highest gradient must actually be removed from the intensity histogram. A well-attested way of achieving this is to weight pixels in the intensity histogram according to their response to a Laplacian filter (Weska et al., 1974). Since such a filter gives an isotropic estimate of the second derivative of the image intensity (i.e., the magnitude of the first derivative of the intensity gradient), it is zero where intensity gradient magnitude is high: hence, it gives such locations zero weight, but it nevertheless weights up those locations on the shoulders of edges. It has been found that this approach is very good at estimating where to place a threshold within a wide valley in the intensity histogram (Weska et al., 1974).

4.3.3 Summary

It has been shown that available techniques are able to provide values at which intensity thresholding can be applied, but they do not themselves solve the problems caused by uneven lighting. They are even less capable of coping with glints, shadows and image clutter. Unfortunately, these artifacts are common in most real situations (Figs. 4.3–4.5) and are only eliminated with difficulty in practice. Indeed, in industrial applications where shiny metal components are involved, glints are the rule rather than the exception, while shadows can seldom be avoided with any sort of object. Even flat objects are liable to have quite strong shadow contours around them because of the particular placement of lights. Lighting problems are studied in detail in Chapter 25. Meanwhile, note that glints and shadows can only be allowed for properly in a two-stage image analysis system, where tentative assignments are made first, and these are firmed up by exact explanation of all pixel intensities. We now return to the problem of making the most of the thresholding technique, by finding how variations in background lighting can be allowed for.

4.4 ADAPTIVE THRESHOLDING

The problem that arises when illumination is not sufficiently uniform may be tackled by permitting the threshold to vary adaptively (or “dynamically”) over the whole image. In principle, there are several ways of achieving this. One involves modeling the background within the image. Another is to work out a local threshold value for each pixel by examining the range of intensities in its neighborhood. A third approach is to split the image into subimages and deal

(a) (b)

FIGURE 4.3

Histogram for the image shown in Fig. 2.7(a). Note that the histogram is not particularly close to the ideal form of [Fig. 4.1](#). Hence, the threshold obtained from (a) (indicated by the short line beneath the scale) does not give ideal results with all the objects in the binarized image (b). Nevertheless, the results are better than for the arbitrarily thresholded image of Fig. 2.7(b).

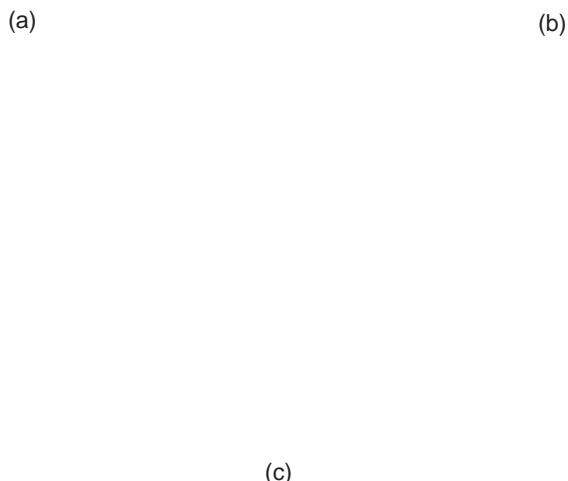
(a) (b)

FIGURE 4.4

Histogram for the image shown in Fig. 2.1(a). The histogram is not at all close to the idealized form, and the results of thresholding (b) are not a particularly useful aid to interpretation.

with them independently. Although “obvious,” the last method will clearly run into problems at the boundaries between subimages, and by the time these problems have been solved, it will look more like one of the other two methods.

The problem can sometimes be solved rather neatly in the following way. On some occasions—such as in automated assembly applications—it is possible to

**FIGURE 4.5**

A picture with more ideal properties. (a) Image of a plug that has been lit fairly uniformly. The histogram (c) approximates to the ideal form, and the result of thresholding (b) is acceptable. However, much of the structure of the plug is lost during binarization.

obtain an image of the background in the absence of any objects. This appears to solve the problem of adaptive thresholding in a rigorous manner, since the tedious task of modeling the background has already been carried out. However, caution is needed because objects bring with them not only shadows (which can in some sense be regarded as part of the objects) but also an additional effect due to the reflections they cast over the background and other objects. This additional effect is nonlinear in the sense that it is necessary to add not only the difference between the object and the background intensity in each case but also an intensity that depends on the products of the reflectances of pairs of objects. These considerations mean that using the no-object background as the equivalent background when several objects are present is ultimately invalid. However, as a first approximation, it is frequently possible to assume an equivalence. If this proves impractical, there is no option but to model the background from the actual image to be segmented.

On other occasions, the background intensity may be rather slowly varying, in which case it may be possible to model it by the following technique (this is a form of Hough transform—see Chapter 11). First, an equation is selected, which can act as a reasonable approximation to the intensity function, for example, a quadratic variation:

$$I = a + bx + cy + dx^2 + exy + fy^2 \quad (4.1)$$

Next, a parameter space for the six variables a, b, c, d, e, f is constructed; then each pixel in the image is taken in turn and all sets of values of the parameters that could have given rise to the pixel intensity value are accumulated in parameter space. Finally, a peak is sought in parameter space, which represents an optimal fit to the background model. So far it appears that this has been carried out only for a linear variation, the analysis being simplified initially by considering only the differences in intensities of pairs of points in image space (Nixon, 1985). Note that a sufficient number of pairs of points must be considered so that the peak in parameter space resulting from background pairs is sufficiently well populated.

4.4.1 The Chow and Kaneko Approach

As early as 1972, Chow and Kaneko introduced what is widely recognized as the standard technique for dynamic thresholding: the technique performs a thoroughgoing analysis of the background intensity variation, making few compromises to save computation (Chow and Kaneko, 1972). In this method, the image is divided into a regular array of overlapping subimages and individual intensity histograms are constructed for each one. Those that are unimodal are ignored since they are assumed not to provide any useful information that can help in modeling the background intensity variation. However, the bimodal distributions are well suited to this task: these are individually fitted to pairs of Gaussian distributions of adjustable height and width and the threshold values are located. Thresholds are then found, by interpolation, for the unimodal distributions. Finally, a second stage of interpolation is necessary to find the correct thresholding value at each pixel.

One problem with this approach is that if the individual subimages are made very small in an effort to model the background illumination more exactly, the statistics of the individual distributions become worse, their minima become less well defined and the thresholds deduced from them are no longer statistically significant. This means that it does not pay to make subimages too small and that ultimately only a certain level of accuracy can be achieved in modeling the background in this way. Clearly, the situation is highly data dependent, but it might be expected that little would be gained by reducing the subimage size below 32×32 pixels. Chow and Kaneko employed 256×256 pixel images and divided these into a 7×7 array of 64×64 pixel subimages with 50% overlap.

Overall, this approach involves considerable computation, and in real-time applications it may well not be viable for this reason.

4.4.2 Local Thresholding Methods

The other approach mentioned earlier is particularly useful for finding local thresholds. It involves analyzing intensities in the neighborhood of each pixel to determine the optimum local thresholding level. Ideally, the Chow and Kaneko histogramming technique would be repeated at each pixel, but this would significantly increase the computational load of this already computationally intensive technique. Thus, it is necessary to obtain the vital information by an efficient sampling procedure. One simple means for achieving this is to take a suitably computed function of nearby intensity values as the threshold: often the mean of the local intensity distribution is taken because this is a simple statistic and gives good results in some cases. For example, in astronomical images, stars have been thresholded in this way. Niblack (1985) reported a case in which a proportion of the local standard deviation was added to the mean to give a more suitable threshold value, the reason (presumably) being to help suppress noise (clearly, addition is appropriate where bright objects such as stars are to be located, whereas subtraction is more appropriate in the case of dark objects).

Another statistic that is frequently used is the mean of the maximum and minimum values in the local intensity distribution. The justification for this is that whatever the sizes of the two main peaks of the distribution, this statistic often gives a reasonable estimate of the position of the histogram minimum. The theory presented earlier shows that this method will only be accurate if (a) the intensity profiles of object edges are symmetrical, (b) noise acts uniformly everywhere in the image so that the widths of the two peaks of the distribution are similar, and (c) the heights of the two distributions do not differ markedly. Sometimes these assumptions are definitely invalid—e.g., when looking for (dark) cracks in eggs or other products. In such cases, the mean and maximum of the local intensity distribution can be found and a threshold deduced using the statistic

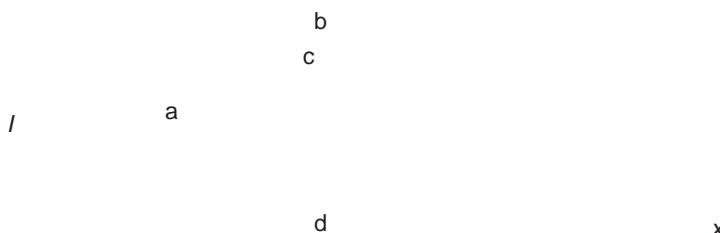
$$T = \text{mean} - (\text{maximum} - \text{mean}) \quad (4.2)$$

where the strategy is to estimate the lowest intensity in the bright background assuming the distribution of noise is symmetrical (Fig. 4.6): use of the mean here is realistic only if the crack is narrow and does not affect the value of the mean significantly. If it does, then the statistic can be adjusted by use of an *ad hoc* parameter:

$$T = \text{mean} - k(\text{maximum} - \text{mean}) \quad (4.3)$$

where k may be as low as 0.5 (Plummer and Dale, 1984).

This method is essentially the same as that of Niblack (1985), but the computational load in estimating the standard deviation is minimized. Each of the last two techniques relies on finding local extrema of intensity. Using these measures helps save computation, but they are clearly somewhat unreliable because of the effects of noise. If this is a serious problem, quartiles or other statistics of the distribution may be used. The alternative of prefiltering the image to remove noise is unlikely to work for crack thresholding, since cracks will almost certainly be removed at the same time as the noise. A better strategy is to form an image of

**FIGURE 4.6**

Method for thresholding the crack in an egg. a, Intensity profile of an egg in the vicinity of a crack: the crack is assumed to appear dark (e.g., under oblique lighting); b, local maximum of intensity on the surface of the egg; c, local mean intensity. Eq. (4.2) gives a useful estimator T of the thresholding level d .

T -values obtained using Eq. (4.2) or (4.3): smoothing this image should then permit the initial image to be thresholded effectively.

Unfortunately, all these methods work well only if the size of the neighborhood selected for estimating the required threshold is large enough to span a significant amount of foreground and background. In many practical cases, this is not possible and the method then adjusts itself erroneously, for example, so that it finds darker spots within dark objects as well as segmenting the dark objects themselves. However, there are certain applications where there is little risk of this occurring. One notable case is that of OCR. Here the widths of character limbs are likely to be known in advance and should not vary substantially. If this is so, then a neighborhood size can be chosen to span or at least sample both character and background, and it is thus possible to threshold the characters highly efficiently using a simple functional test of the type described above. The effectiveness of this procedure (Table 4.1) is demonstrated in Fig. 4.7.

Finally, before leaving this topic, note that hysteresis thresholding is a type of adaptive thresholding—effectively permitting the threshold value to vary locally: this topic is investigated in Section 5.10.

4.5 MORE THOROUGHGOING APPROACHES TO THRESHOLD SELECTION

At this point, we return to global threshold selection and describe some important approaches that have a rigorous mathematical basis. The first of these is variance-based thresholding, the second is entropy-based thresholding, and the third is maximum likelihood thresholding. All three are widely used, the second having achieved an increasingly wide following over the past 20–30 years, and the third is a more broad-based technique that has its roots in statistical pattern recognition—a subject that is covered in Chapter 24.

Table 4.1 A Simple Algorithm for Adaptively Thresholding Print

```

minrange = 255 / 5;
/* minimum likely difference in intensity between print and background:
this parameter can be preset manually or “learnt” by a previous routine */
for all pixels in image do {
    find minimum and maximum of local intensity distribution;
    range = maximum - minimum;
    if (range > minrange)
        T = (minimum + maximum)/2; // print is visible in neighborhood
    else T = maximum - minrange/2; // neighborhood is all white
    if (P0 > T) Q0 = 255; else Q0 = 0; // now binarize print
}

```

(a)

(b)

(c)

FIGURE 4.7

Effectiveness of local thresholding on printed text. Here, a simple local thresholding procedure (Table 4.1), operating within a 3×3 neighborhood, is used to binarize the image of a piece of printed text (a). Despite the poor illumination, binarization is performed quite effectively (b). Note the complete absence of isolated noise points in (b), while by contrast the dots on all the i's are accurately reproduced. The best that could be achieved by uniform thresholding is shown in (c).

4.5.1 Variance-Based Thresholding

The standard approach to thresholding outlined earlier involved finding the neck of the global image intensity histogram. However, this is impracticable when the dark peak of the histogram is minuscule in size, as it will then be hidden among the noise in the histogram and it will not be possible to extract it with the usual algorithms.

A good many investigators have studied this sort of problem (e.g., Otsu, 1979; Kittler et al., 1985; Sahoo et al., 1988; Abutaleb, 1989): among the most well-known approaches are the variance-based methods. In these methods, the image intensity histogram is analyzed to find where it can best be partitioned to optimize criteria based on ratios of the within-class, between-class, and total variance. The simplest approach (Otsu, 1979) is to calculate the between-class variance, as will now be described.

First, we assume that the image has a grayscale resolution of L gray levels. The number of pixels with gray level i is written as n_i , so the total number of pixels in the image is $N = n_1 + n_2 + \dots + n_L$. Thus, the probability of a pixel having gray level i is:

$$p_i = \frac{n_i}{N} \quad (4.4)$$

where

$$p_i \geq 0 \quad \sum_{i=1}^L p_i = 1 \quad (4.5)$$

For ranges of intensities up to and above the threshold value k , we can now calculate the between-class variance σ_B^2 and the total variance σ_T^2 :

$$\sigma_B^2 = \pi_0(\mu_0 - \mu_T)^2 + \pi_1(\mu_1 - \mu_T)^2 \quad (4.6)$$

$$\sigma_T^2 = \sum_{i=1}^L (i - \mu_T)^2 p_i \quad (4.7)$$

where

$$\pi_0 = \sum_{i=1}^k p_i \quad \pi_1 = \sum_{i=k+1}^L p_i = 1 - \pi_0 \quad (4.8)$$

$$\mu_0 = \sum_{i=1}^k ip_i / \pi_0 \quad \mu_1 = \sum_{i=k+1}^L ip_i / \pi_1 \quad \mu_T = \sum_{i=1}^L ip_i \quad (4.9)$$

Making use of the latter definitions, the formula for the between-class variance can be simplified to:

$$\sigma_B^2 = \pi_0\pi_1(\mu_1 - \mu_0)^2 \quad (4.10)$$

For a single threshold, the criterion to be maximized is the ratio of the between-class variance to the total variance:

$$\eta = \frac{\sigma_B^2}{\sigma_T^2} \quad (4.11)$$

However, the total variance is constant for a given image histogram, so maximizing η simplifies to maximizing the between-class variance.

The method can readily be extended to the dual threshold case $1 \leq k_1 \leq k_2 \leq L$, where the resultant classes, C_0, C_1 , and C_2 , have respective gray-level ranges of $[1, \dots, k_1]$, $[k_1 + 1, \dots, k_2]$, and $[k_2 + 1, \dots, L]$.

In some situations (e.g., Hannah et al., 1995), this approach is still not sensitive enough to cope with histogram noise, and more sophisticated methods must be used. One such technique is that of entropy-based thresholding, which has become firmly embedded in the subject (Pun, 1980; Kapur et al., 1985; Abutaleb, 1989; Brink, 1992). For further insight into the performance of the between-class variance method (BCVM), see Section 4.7.

4.5.2 Entropy-Based Thresholding

Entropy measures of thresholding are based on the concept of entropy. The entropy statistic is high if a variable is well distributed over the available range, and low if it is well ordered and narrowly distributed: specifically, entropy is a measure of disorder, and is zero for a perfectly ordered system. The concept of entropy thresholding is to threshold at an intensity for which the sum of the entropies of the two intensity probability distributions thereby separated is maximized. The reason for this is to obtain the greatest reduction in entropy—i.e., the greatest increase in order—by applying the threshold: in other words, the most appropriate threshold level is the one that imposes the greatest order on the system, and thus leads to the most meaningful result.

To proceed, the intensity probability distribution is again divided into two classes—those with gray levels up to the threshold value k and those with gray levels above k (Kapur et al., 1985). This leads to two probability distributions A and B:

$$A: \quad \frac{p_1}{P_k}, \frac{p_2}{P_k}, \dots, \frac{p_k}{P_k} \quad (4.12)$$

$$B: \quad \frac{p_{k+1}}{1-P_k}, \frac{p_{k+2}}{1-P_k}, \dots, \frac{p_L}{1-P_k} \quad (4.13)$$

where

$$P_k = \sum_{i=1}^k p_i \quad 1 - P_k = \sum_{i=k+1}^L p_i \quad (4.14)$$

The entropies for each class are given by:

$$H(A) = - \sum_{i=1}^k \frac{p_i}{P_k} \ln \frac{p_i}{P_k} \quad (4.15)$$

$$H(B) = - \sum_{i=k+1}^L \frac{p_i}{1-P_k} \ln \frac{p_i}{1-P_k} \quad (4.16)$$

and the total entropy is:

$$H(k) = H(A) + H(B) \quad (4.17)$$

Substitution leads to the final formula:

$$H(k) = \ln \left(\sum_{i=1}^k p_i \right) + \ln \left(\sum_{i=k+1}^L p_i \right) - \frac{\sum_{i=1}^k p_i \ln p_i}{\sum_{i=1}^k p_i} - \frac{\sum_{i=k+1}^L p_i \ln p_i}{\sum_{i=k+1}^L p_i} \quad (4.18)$$

and it is this parameter that has to be maximized.

This approach can give very good results—see, e.g., Hannah et al. (1995). Again, it is straightforwardly extended to dual thresholds, but we shall not go into the details here (Kapur et al., 1985). In fact, probabilistic analysis to find mathematically ideal dual thresholds may not be the best approach in practical situations: an alternative technique for determining dual thresholds sequentially has been devised by Hannah et al. (1995), and applied to an X-ray inspection task—as described in Chapter 20.

4.5.3 Maximum Likelihood Thresholding

When dealing with distributions such as intensity histograms, it is important to compare the actual data with the data that might be expected from a previously constructed model based on a training set: this is in agreement with the methods of statistical pattern recognition (see Chapter 24), which takes full account of prior probabilities. For this purpose, one option is to model the training set data using a known distribution function such as a Gaussian. The latter has many advantages, including its accessibility to relatively straightforward mathematical analysis. In addition, it is specifiable in terms of two well-known parameters—the mean and standard deviation—which are easily measured in practical situations. Indeed, for any Gaussian distribution, we have:

$$p_i(x) = \frac{1}{(2\pi\sigma_i^2)^{1/2}} \exp \left[-\frac{(x - \mu_i)^2}{2\sigma_i^2} \right] \quad (4.19)$$

where the suffix i refers to a specific distribution, and of course when thresholding is being carried out, there is a supposition that two such distributions are involved. Applying the respective *a priori* class probabilities P_1, P_2 (Chapter 24), careful analysis (Gonzalez and Woods, 1992) shows that the condition $p_1(x) = p_2(x)$ reduces to the form:

$$x^2 \left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2} \right) - 2x \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2} \right) + \left(\frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_2^2}{\sigma_2^2} \right) + 2 \log \left(\frac{P_2 \sigma_1}{P_1 \sigma_2} \right) = 0 \quad (4.20)$$

Note that, in general, this equation has two solutions,² implying the need for two thresholds, although when $\sigma_1 = \sigma_2$ there is a single solution:

$$x = \frac{1}{2} (\mu_1 + \mu_2) + \frac{\sigma^2}{\mu_1 - \mu_2} \ln \left(\frac{P_2}{P_1} \right) \quad (4.21)$$

In addition, when the prior probabilities for the two classes are equal, the equation reduces to the altogether simpler and more obvious form:

$$x = \frac{1}{2} (\mu_1 + \mu_2) \quad (4.22)$$

Of all the methods described in this chapter, only the maximum likelihood method makes use of *a priori* probabilities. While this makes it look as if it is the only rigorous method, and indeed that all other methods are automatically erroneous and biased in their estimations, this is not the actual position. The reason lies in the fact that the other methods incorporate actual frequencies of sample data, which embody the *a priori* probabilities (see Section 24.4). Hence, the other methods should give correct results. Nevertheless, it is refreshing to see *a priori* probabilities brought in explicitly, as this gives a greater confidence of getting unbiased results in any doubtful situations.

4.6 THE GLOBAL VALLEY APPROACH TO THRESHOLDING

An important disadvantage of the many approaches to threshold estimation, including particularly entropy thresholding and its variants, is that it is often unclear how they will react to unusual or demanding situations, such as where multiple thresholds have to be found in the same image (Kapur et al., 1985;

²The reason for the existence of two solutions is that one solution represents a threshold in the area of overlap between the two Gaussians; the other solution is mathematically unavoidable, and lies at either very high or very low intensities. It is this latter solution that disappears when the two Gaussians have equal variance, as the distributions clearly never cross again. In any case, it seems unlikely that the distributions being modeled would in practice approximate so well to Gaussians that the non-central solution could ever be important—i.e. it is essentially a mathematical fiction that needs to be eliminated from consideration.

(a) (b)

FIGURE 4.8

Result of applying global minimization algorithm to 1-D data sets. (a) A basic two-peak structure. (b) A basic multimode structure. Top trace: original 1-D data sets. Middle trace: results from Eq. (4.23). Bottom trace: results from Eq. (4.24).

Source: © IET 2008

Hannah et al., 1995; Tao et al., 2003; Wang and Bai, 2003; Sezgin and Sankur, 2004). Added to this, there is the risk that the more complex approaches will miss important aspects of the original data. The global valley approach (Davies, 2007a) aimed to provide a rigorous means of going back to basics to find global valleys of intensity histograms in such a way as to embody the intrinsic meaning of the data.

The top trace of Fig. 4.8(a) shows the basic situation—where thresholding is effective and the optimum threshold should be simple to locate. However, the intensity histogram often contains such a welter of peaks and valleys that even the human eye, with its huge capability for analysis “at a glance,” can be confused—especially when it is necessary to identify global valley positions rather than local minima of lesser significance. The situation is made clearer by the example shown in the top trace of Fig. 4.8(b). Here valley 1 (numbering from the left) is lower than valley 3, but valley 3 is deeper in the sense that it has two high peaks immediately around it; however, valley 1 also lies between the highest two peaks, and in that sense it is the *globally* deepest valley in the distribution.

Clearly, to judge global valley deepness, we need a mathematical criterion so that comparisons between all the valleys can be carried out unambiguously. To proceed, for any potential global valley point (call it point j), we need to look at all points (i) on the left of it to find the highest peak to the left and all points (k) on the right of it to find the highest peak to the right, before we can construct a suitable criterion value for point j . Hence, we need to take the maximum over all points i and the maximum over all points k . Furthermore, we need to do this for all points j , and for each of them we need to consider only points i ($i < j$) and points k ($k > j$), and take account of the corresponding heights h_i , h_j , h_k in the distribution. The maximum must then be taken for a criterion function C_j of general

form $\max_{i,k} \{Q(h_i - h_j, h_k - h_j)\}$. An obvious criterion function of this form employs the arithmetic mean. However, to avoid complications from negative heights, we introduce a sign function $s(\cdot)$ such that $s(u) = u$ if $u > 0$ and $s(u) = 0$ if $u \leq 0$. The result is the following function:

$$F_j = \max_{i,k} \left\{ \frac{1}{2} [s(h_i - h_j) + s(h_k - h_j)] \right\} \quad (4.23)$$

When this is applied to the top trace of Fig. 4.8(a), the result is a distribution (middle trace of Fig. 4.8(a)) that has a maximum at the required valley position. In addition, the values of i and k corresponding to this maximum are the first and third peak positions in the original intensity distribution. The sign function $s(\cdot)$ has the effect of preventing negative responses that would complicate the situation unnecessarily.

While the function F used above is straightforward to apply and employs linear expressions that are often attractive in permitting in-depth analysis, it results in pedestals at either end of the output distribution: these could complicate the situation when there are many peaks and valleys. Fortunately, the geometric mean is not subject to this problem, and so it is the one that is adopted in the global valley method (GVM). Thus, we use the following function instead of F_j :

$$K_j = \max_{i,k} \left\{ [s(h_i - h_j)s(h_k - h_j)]^{1/2} \right\} \quad (4.24)$$

Note that the arithmetic and geometric means are very similar when the two arguments are nearly equal, but deviate a lot when the two arguments are dissimilar: it is the dissimilar case that applies at the ends of the distribution, where it is required to suppress a potential valley that has only one peak near to it, and the geometric mean then offers a sound advantage over the arithmetic mean. These ideas are further made clear in Fig. 4.8(b).

Overall, the rationale for this approach is that we are looking for the most significant valley in an intensity distribution, corresponding to an optimum discriminating point between, for example, dark objects and light background in the original image. While in some cases the situation is obvious (Fig. 4.8(a)), in general it is difficult to sort out a confusing set of peaks and valleys and in particular to identify global valleys. So the concept embodied in Eq. (4.24) is that of aiming to guarantee an optimal global solution by automatic means. Clearly, by analysis of the output distribution, it is also possible to find a whole range of maxima corresponding to global valley positions in the input distribution: to this extent, the method is able to cope with multimode distributions and to find multiple threshold positions.

With all histogramming methods, it is necessary to take due account of local noise in the distribution, as it could lead to inaccurate results. Hence, the K distribution is smoothed before proceeding with further analysis to locate thresholds.

Another important factor is the amount of computation required for this approach. While it at first appears that a computationally intensive scan over all possible sets of sampling points i, j, k is required to obtain the optimal solution, it turns out that with care the computational load can be reduced from $O(N^3)$ to $O(N)$, where N is the number of gray levels in the intensity distribution.

4.7 PRACTICAL RESULTS OBTAINED USING THE GLOBAL VALLEY METHOD

The ideas presented in Section 4.6 are next tested using Fig. 4.9(a). Starting with this image, the following sequence of operations is applied: (a) an intensity histogram is generated (top trace in Fig. 4.9(d)); (b) the function K is applied (middle trace in Fig. 4.9(d)); (c) the output distribution is smoothed (bottom trace in Fig. 4.9(d)); (d) peaks are located (see the short vertical lines at the bottom of Fig. 4.9(d)); (e) the most significant peaks are chosen as threshold levels (here all eight are selected); (f) a new image is generated by applying the mean of the adjacent threshold intensity levels. The result (Fig. 4.9(b)) is a reasonably segmented likeness of the original image, albeit with clear limitations in the cloud regions—simply because accurate renditions of these would require a rather full range of gray levels, and thresholding is not appropriate in such regions. However, what is significant is the ease with which the approach automatically incorporates multi-level thresholding of multimode intensity distributions—a point that has been a difficulty with entropy thresholding, for example (Hannah et al., 1995). Finally, Fig. 4.9(c) gives a comparison with the maximum BCVM of Otsu (1979), which has recently undergone something of a resurgence of popularity and use, partly as a result of the ease with which it can be used for the systematic generation of multi-level thresholds (Liao et al., 2001; Otsu, 1979).

The reconstructability of the method (in the sense that much of the image is reconstructed so well that it is difficult to distinguish from the original) is an indication of success in that it is clear that the information removed was by no means arbitrary, but was actually redundant and unhelpful. This property is also evident in Fig. 4.10, which shows the application of the method to the well-known Lena image.

The basic criterion used for smoothing is that of reducing noise as far as possible without eliminating relevant thresholding points. To achieve this, repeated convolutions of the K distributions are made with a three-element $\frac{1}{4}[1 \ 2 \ 1]$ kernel until an appropriate amount of smoothing is obtained. Note that the GVM peaks are by no means static. In particular, as smoothing progresses, they gradually move and then merge, as can be seen in the bottom traces in Fig. 4.10(f–h). Just before merger, there is often a rapid movement to align the merging peaks. To cope with this and to find suitable thresholding levels, a useful heuristic was to move one quarter of the way from the merged position to the next merger position



FIGURE 4.9

Result of applying the global valley algorithm to a multimode intensity distribution. (a) Original grayscale image. (b) Reconstituted image after multiple thresholding using the eight peaks in the output distribution. (d) Top: original intensity histogram for (a). Middle: result of applying the global valley transformation. Bottom: result of smoothing. The eight short vertical lines at the very bottom indicate the peak positions. In (d), the intensity scale is 0–255; the vertical scale is normalized to a maximum height indicated by the height of the vertical axis. *Note:* the three traces are computed 25 times more accurately than the rounded values displayed, so the peak locations are determined as accurately as indicated. For comparison, (c) shows the result of applying the between-class variance method to the same image: the eight thresholds are indicated by vertical lines in the top trace of (d).

Source: © IET 2008

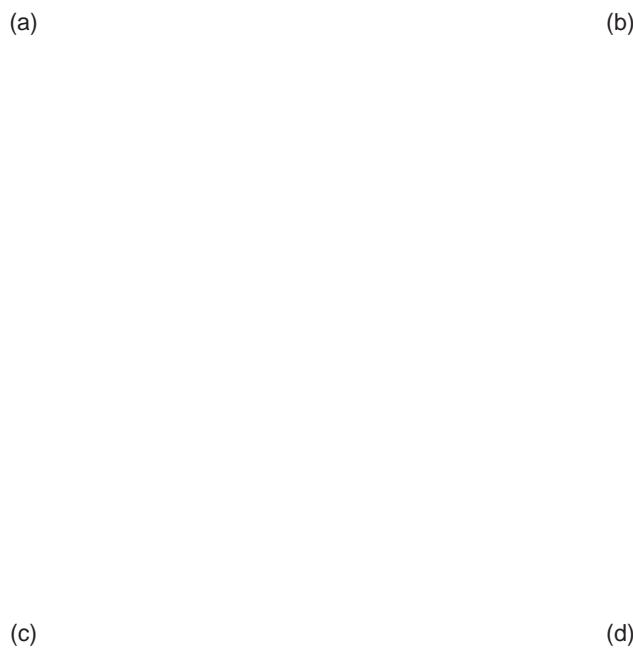


FIGURE 4.10

Multilevel thresholding of the Lena image. For the original Lena grayscale image, see ‘Miscellaneous’ at the USC-SIPI Image Database*. (a) Result of applying the between-class variance method (BCVM) to original image. (b)–(d) Results of applying the global valley method to original image, producing, respectively, bi-level, tri-level, and five-level images. (e) Top: intensity histogram of original image: the vertical line indicates the bi-level threshold selected by the BCVM. Bottom: the resulting K distribution. (f)–(h) The upper traces show smoothed versions of the K distribution, with short vertical lines indicating, respectively, one, two, or four threshold positions; the lower traces show threshold positions resulting from progressive smoothing of the K distribution: note that these are scaled and some are truncated as indicated by the horizontal gray line at the top; the horizontal dotted lines show how sets of threshold values are selected automatically (see text).

Source: © IET 2008

*<http://sipi.usc.edu/database/database.php> (website accessed 13 December 2011).

**FIGURE 4.10**

(Continued)

(see horizontal dotted lines in Fig. 4.10(f–h)). To clarify the process, the basic GVM algorithm is given in Fig. 4.11.

Figure 4.10(f–h) gives three examples of smoothing until 1, 2 or 4 thresholding points are produced (these give bi-level, tri-level, and five-level thresholding).³ These lead to the images shown in Fig. 4.10(b–d) (note particularly that the light shaded region on Lena’s nose is very stable and noise-free).

We concentrate next on a specific advantage of the GVM: that it produces robust judgments of minority intensities at the ends of the intensity range. Effectively, it amplifies such regions of the distribution and provides highly stable image segmentations: see, in particular, the under-vehicle shadows located in Fig. 23.1(d) and the ergot contaminant located in Fig. 21.2(d).⁴ That the

³There is a potential confusion here: as smoothing proceeds, the number of GVM thresholds progressively *decreases*. Hence the ordering of the respective images and traces in subfigures (f)–(h) appears inverted from this point of view. However, it is the logical order for the BCVM for which computation increases approximately exponentially with the number of thresholds.

⁴Note that these represent important vehicle guidance and inspection tasks: (1) use of under-vehicle shadows is a promising technique for locating vehicles on the road ahead (Liu et al., 2007); (2) ergot is poisonous and it is important to locate it amongst wheat or other grains that are to be used for human consumption (Davies, 2003b).

```

scan = 0;
do {
    numberofpeaks = 0;
    for (all intensity values in distribution) {
        if (peak found) {
            peakposition[scan, numberofpeaks] = intensity;
            numberofpeaks++;
        }
    }
    if (numberofpeaks == requirednumber) {
        if (previousnumberofpeaks > numberofpeaks) lowestscan = scan;
        else highestscan = scan;
    }
    previousnumberofpeaks = numberofpeaks;
    apply incremental smoothing kernel to distribution;
    scan++;
} while (numberofpeaks > 0);

optimumscan = (lowestscan*3 + highestscan)/4;
for (all peaks up to requirednumber)
    bestpeakposition[peak] = peakposition[optimumscan, peak];

```

FIGURE 4.11

Basic global valley algorithm. This version of the algorithm assumes that the required number of peaks (*required number*) is known in advance, although the optimum amount of smoothing is unknown. Here, the latter is estimated by taking a weighted mean of the lowest and highest numbers of smoothing scans that yield the required number of peaks. The final line of the algorithm gives the required number of peaks in the best positions. While this form of the algorithm obtains positions for a specific required number of peaks, the underlying process also maps out a complete set of stability graphs because it proceeds until the number of peaks is zero. For further details, see Section 4.7.

Source: © IET 2008

GVM is able to make sense of the exceptionally noisy K distribution shown in Fig. 21.2(d) seems rather remarkable.

Comparing the GVM results with those of the BCVM (see Fig. 4.10(a, e)), we see that the bi-level BCVM threshold appears to lie in an *a priori* quite reasonable position in the intensity histogram: however, closer examination shows that the performance of the BCVM approximates to splitting the active area of the histogram into equal parts, corresponding to finding an approximate median. This means that for nearly unimodal histograms, it has much less chance of leading to optimal segmentations. This view of its operation is supported by tests (Fig. 4.12) made on idealized histograms, which show that it is unable to locate the bottom of the valley. It is also noteworthy that, unlike the GVM, the multilevel BCVM sometimes misses thresholds at the ends of the range of intensities (see, e.g., the vertical lines in the top trace of Fig. 4.9(d)).

Overall, it has been found that the GVM produces significantly more stable thresholds than the BCVM, that it is less prone to producing noisy boundaries in the thresholded images, and that its results tend to be more meaningful.

**FIGURE 4.12**

Results of applying the between-class variance method (BCVM) and concavity analysis in idealized cases. Applying the BCVM to (a) a triangular histogram and (b) a parabolic histogram. The vertical lines indicate the bi-level threshold selected by the BCVM: note that in each case it lies well away from the obvious global minimum of the histogram. (c) Finding thresholds by concavity analysis. The technique forms the convex hull of the distribution, takes each joining line, and uses the foot of the longest normal as an indicator of the threshold position. This approach is often highly effective but tends to give a result closer to the main peak than the optimum minimum location.

Source: © IET 2008

In fact, the BCVM tends to split intensity distributions rather blindly into approximately equal areas: although its mathematical formulation does not explicitly aim at this, it often seems to have essentially this effect.

4.8 HISTOGRAM CONCAVITY ANALYSIS

In this section, we briefly consider previous work on histogram concavity analysis. Rosin (2001) described how a simple geometrical construction (Fig. 4.12(c)) could be used to identify a suitable bi-level threshold. The technique depends on the histogram having a “corner,” which is then easily identified, but when the corner is less well defined, bias can creep in and it becomes necessary to model the histogram distributions to obtain systematic corrections to the thresholding point. The approach will work for true unimodal distributions (including those produced by grey-scale edge images) or for “nearly” unimodal distributions where there is a very weak mode in addition to the main mode. For true unimodal distributions, the GVM will not work because one of the component signals in function K is zero: in such cases, it is imperative to use a method such as that described by Rosin—although others have been described over a long period—by Rosenfeld and de la Torre (1983), Tsai (1995), and others. For nearly unimodal distributions, the Rosin approach gives some intrinsic bias, as indicated in Fig. 4.12(c): but it is presumably possible in many applications to perform modeling to overcome this problem. However, the need for modeling does not seem to arise with the GVM—as has already been demonstrated (see particularly Figs. 21.2 and 23.1).

4.9 CONCLUDING REMARKS

Sections 4.3 and 4.4 have revealed a number of factors that are crucial to the process of thresholding. First, the need to avoid bias in threshold selection by arranging roughly equal populations in the dark and light regions of the intensity histogram; second, the need to work with a small subimage (or neighborhood) size so that the intensity histogram has a well-defined valley despite variations in illumination; and third, the need for subimages to be sufficiently large so that statistics are reliable, permitting the valley to be located accurately.

Unfortunately, these conditions are not compatible and compromises are needed in practical situations. In particular, it is generally not possible to find a neighborhood size that can be applied everywhere in an image, on all occasions yielding roughly equal populations of dark and light pixels. Indeed, if the chosen size is small enough to span edges ideally, hence yielding unbiased local thresholds, it will be valueless inside large objects. Attempting to avoid this situation by resorting to alternative methods of threshold calculation does not solve the problem since inherent to such methods is a built-in region size. It is therefore not surprising that a number of workers have opted for variable resolution and hierarchical techniques in an attempt to make thresholding more effective (Wu et al., 1982; Wermser et al., 1984; Kittler et al., 1985).

At this stage, we call into question the complications involved in such thresholding procedures—which become even worse when intensity distributions start to become multimodal. Note that the overall procedure is to find local intensity gradients in order to obtain accurate, unbiased estimates of thresholds so that it then becomes possible to take a horizontal slice through a grayscale image and hence, ultimately, find “vertical” (i.e., spatial) boundaries within the image. Why not use the gradients *directly* to estimate the boundary positions? Such an approach, for example, leads to no problems from large regions where intensity histograms are essentially unimodal, although it would be foolish to pretend that there are no other problems (see Chapters 5 and 10).

On the whole, the author takes the view that many approaches (region-growing, thresholding, edge detection, etc.), *taken to the limits of approximation*, will give equally good results. After all, they are all limited by the same physical effects—image noise, variability of lighting, presence of shadows, etc. However, some methods are easier to coax into working well, or need minimal computation, or have other useful properties such as robustness. Thus, thresholding can be a highly efficient means of aiding the interpretation of certain types of image: but as soon as image complexity rises above a certain critical level, it suddenly becomes more effective and considerably less complicated to rely on edge detection. This is studied in the next chapter. Meanwhile, we must not overlook the possibility of easing the thresholding task by optimizing the lighting system and ensuring that any worktable or conveyor is kept clean and white: this turns out to be a viable approach in a surprisingly large number of industrial applications.

The end result of thresholding is a set of silhouettes representing the shapes of objects: these constitute a “binarized” version of the original image. Many techniques exist for performing binary shape analysis, and some of these are described in Chapter 9. Meanwhile, note that many features of the original scene—e.g., texture, grooves or other surface structure—will not be present in the binarized image. Although the use of multiple thresholds to generate a number of binarized versions of the original image can preserve relevant information present in the original image, this approach tends to be clumsy and impracticable, and sooner or later one may be forced to return to the original grayscale image for the required data.

Thresholding is among the simplest of image processing operations and is an intrinsically appealing way of performing segmentation. While the approach is clearly limited, it would be a mistake to ignore it and its recent developments, which provide useful tools for the programmer’s toolkit.

4.10 BIBLIOGRAPHICAL AND HISTORICAL NOTES

Segmentation by thresholding started many years ago from simple beginnings, and in recent years has been refined into a set of mature procedures. Among the notable early methods is the paradigm but computation-intensive Chow and Kaneko method (1972), which has been outlined in Section 4.4.1. Nakagawa and Rosenfeld (1979) studied the method and developed it for cases of trimodal distributions but without improving computational load.

Fu and Mui (1981) provided a useful general survey on image segmentation: which was updated by Haralick and Shapiro (1985). These papers review many topics that could not be covered in this chapter due to space reasons—which also applies for Sahoo et al.’s (1988) valuable survey of thresholding techniques. Nevertheless, it is worth emphasizing the point made by Fu and Mui (1981) that “All the region extraction techniques process the pictures in an iterative manner and usually involve a great expenditure in computation time and memory.”

As hinted in Section 4.4, thresholding (particularly local adaptive thresholding) has had many applications in optical character recognition. Among the earliest were the algorithms described by Bartz (1968) and Ullmann (1974): also two highly effective algorithms have been described by White and Rohrer (1983).

During the 1980s, the entropy approach to automatic thresholding evolved (e.g., Pun, 1981; Kapur et al., 1985; Abutaleb, 1989; Pal and Pal, 1989): this approach (Section 4.5.2) proved highly effective, and its development continued during the 1990s (e.g., Hannah et al., 1995).

In the 2000s, the entropy approach to threshold selection has remained important, in respect both of conventional region location and ascertaining the transition

region between objects and background to make the segmentation process more reliable (Yan et al., 2003). In one instance, it was found useful to employ fuzzy entropy and genetic algorithms (Tao et al., 2003). Wang and Bai (2003) have shown how threshold selection may be made more reliable by clustering the intensities of boundary pixels, while ensuring that a continuous rather than a discrete boundary is considered (the problem is that in images that approximate to binary images over restricted regions, the edge points will lie preferentially in the object or the background, not neatly between both). However, in complex outdoor scenes and for many medical images such as brain scans, thresholding alone will not be sufficient, and resort may even have to be made to graph matching (Chapter 14) to produce the best results—reflecting the important fact that segmentation is necessarily a high-level rather than a low-level process (Wang and Siskind, 2003). In rather less demanding cases, deformable model-guided split-and-merge techniques may, on the other hand, still be sufficient (Liu and Sclaroff, 2004).

4.10.1 More Recent Developments

Sezgin and Sankur (2004) give a thorough review and assessment of work on thresholding prior to 2004. More recently, there has been continued interest in thresholding in the case of unimodal (Coudray et al., 2010; Medina-Carnicer et al., 2011) and near-unimodal histograms (Davies, 2007a, 2008b): the latter case is covered fairly fully in Sections 4.6 and 4.7. In the case of Coudray et al. (2010), the aim is to threshold intensity gradient histograms in order to locate edges reliably: the approach taken is to model the contribution from noise as a Rayleigh distribution and then to devise heuristics for analyzing the overall distribution. With the same aim, Medina-Carnicer et al. (2011) show that applying a histogram transformation improves the performance of the Otsu (1979) and Rosin (2001) methods. Li et al. (2011) adopt the novel approach of constraining the gray-level ranges considered by the thresholding algorithm in such a way as to weaken gray-level changes in both foreground and background, thus simplifying the original image and making the intensity histogram more closely bimodal. After that several thresholding methods are found to operate more reliably. Ng (2006) describes a revised version of the Otsu (1979) method that operates well for unimodal distributions, and which is useful for defect detection. This “valley emphasis” method works by applying a weight to the Otsu threshold calculation. Overall, several of the recent developments can be construed as applying transformations or other improvements to older methods to make them more sophisticated and accurate: in fact none is highly complex in any theoretical way. Finally, it may seem somewhat surprising that, after so many decades, thresholding is still something of a “hot” subject: the driving force for this is its extreme simplicity and high level of utility.

4.11 PROBLEMS

1. Using the ideas outlined in Section 4.3.2, model the intensity distribution obtained by finding all the edge pixels in an image and including also all pixels adjacent to these pixels. Show that while this gives a sharper valley than for the original intensity distribution, it is not as sharp as for pixels located by the Laplacian operator.
2. Consider whether it is more accurate to estimate a suitable threshold for a bimodal, dual-Gaussian distribution by (a) finding the position of the minimum, or (b) finding the mean of the two peak positions. What corrections could be made by taking account of the magnitudes of the peaks?
3. Obtain a complete derivation of Eq. (4.20). Show that, in general (as stated in Section 4.5.3), it has two solutions. What is the physical reason for this? How can it have only one solution when $\sigma_1 = \sigma_2$?
4. Prove the statement made in Section 4.6 that the computational load of the histogram analysis for the global value method can be reduced from $O(N^3)$ to $O(N)$. Show also that the number of passes over the histogram required to achieve this is at most 2.

Edge Detection

5

Edge detection provides an intrinsically more rigorous means than thresholding for initiating image segmentation. However, there is a large history of *ad hoc* edge detection algorithms, and this chapter aims to distinguish what is principled from what is *ad hoc* and to provide theoretical and practical knowledge underpinning available techniques.

Look out for:

- the variety of template matching operators that have been used for edge detection—e.g., the Prewitt, Kirsch, and Robinson operators.
- the differential gradient approach to edge detection—exemplified by the Roberts, Sobel, and Frei–Chen operators.
- theory explaining the performance of the template matching operators.
- methods for the optimal design of differential gradient operators and the value of “circular” operators.
- tradeoffs between resolution, noise suppression capability, location accuracy, and orientation accuracy.
- the distinction between edge enhancement and edge detection.
- outlines of more modern operators—the Canny and Laplacian-based operators.
- the use of active contour models (snakes) for modeling object boundaries.
- the “graph cut” approach to object segmentation.

In discussing the process of edge detection, this chapter shows that it is possible to estimate edge orientation with surprising accuracy within a small window—the secret being the considerable information residing in the grayscale values. High orientation accuracy turns out to be of particular value when using the Hough transform to locate extended objects in digital images—as will be seen in several chapters in Part 2 of this book.

5.1 INTRODUCTION

In Chapter 4, segmentation has been tackled by the general approach of finding regions of uniformity in images—on the basis that the areas found in this way would have a fair likelihood of coinciding with the surfaces and facets of objects. The most computationally efficient means of following this approach was that of thresholding but for real images. This turns out to be failure-prone or quite difficult to implement satisfactorily. Indeed, to make it work well seems to require a multiresolution or hierarchical approach, coupled with sensitive measures for obtaining suitable local thresholds. Such measures have to take account of local intensity gradients as well as pixel intensities, and the possibility of proceeding more simply—by taking account of intensity gradients alone—was suggested.

In fact, edge detection has long been an alternative path to image segmentation and is the method pursued in this chapter. Whichever way is inherently the better approach, edge detection has the additional advantage in that it immediately reduces by a large factor (typically around 100) the considerable redundancy of most image data: this is useful because it significantly reduces both the space needed to store the information and the amount of processing subsequently required to analyze it.

Edge detection has gone through an evolution spanning well over 30 years. Two main methods of edge detection have been apparent over this period, the first of these being the template matching (TM) approach and the second being the differential gradient (DG) approach. In either case the aim is to find where the intensity gradient magnitude g is sufficiently large to be taken as a reliable indicator of the edge of an object. Then g can be thresholded in a similar way in which intensity has been thresholded in Chapter 4 (in fact, we shall see that it is possible to look for local maxima of g instead of, or as well as, thresholding it). The TM and DG methods differ mainly in how they proceed to estimate g locally; however, there are also important differences in how they determine local edge orientation, which is an important variable in certain object detection schemes.

Later in the chapter we look at the Canny operator, which was much more rigorously designed than previous edge detectors. Then we consider Laplacian-based operators before moving on to study active contour models or “snakes.” Finally, we outline the “graph cut” approach to object segmentation: this makes use of intensity gradient information to zone in on object regions, thereby in a sense embodying both the edge detection and the region growing paradigms and ending up with ideal, provably unique solutions.

Before proceeding to discuss the performance of the various edge detection operators, note that there are a variety of types of edge, including in particular the “sudden step” edge, the “slanted step” edge, the “planar” edge, and various intermediate edge profiles (see Fig. 5.1). This chapter considers edges of the types shown in Fig. 5.1(a)–(d): edges of the types shown in Fig. 5.1(e) and (f) are much rarer; an example being shown in Fig. 11.4(a).

**FIGURE 5.1**

Edge models: (a) sudden step edge; (b) slanted step edge; (c) smooth step edge; (d) planar edge; (e) roof edge; and (f) line edge. The effective profiles of edge models are nonzero only within the stated neighborhood. The slanted step and the smooth step are approximations to realistic edge profiles: the sudden step and the planar edge are extreme forms that are useful for comparisons (see text).

5.2 BASIC THEORY OF EDGE DETECTION

Both DG and TM operators estimate local intensity gradients with the aid of suitable convolution masks. In the case of the DG type of operator, only two such masks are required—for the x and y directions. In the TM case, it is usual to employ up to 12 convolution masks capable of estimating local components of gradient in different directions (Prewitt, 1970; Kirsch, 1971; Robinson, 1977; Abdou and Pratt, 1979).

In the TM approach, the local edge gradient magnitude (for short, the edge “magnitude”) is approximated by taking the maximum of the responses for the component masks:

$$g = \max (g_i : i = 1, \dots, n) \quad (5.1)$$

where n is usually 8 or 12.

In the DG approach, the local edge magnitude may be computed vectorially using the nonlinear transformation:

$$g = \left(g_x^2 + g_y^2 \right)^{1/2} \quad (5.2)$$

To save computational effort, it is common practice (Abdou and Pratt, 1979) to approximate this formula by one of the simpler forms:

$$g = |g_x| + |g_y| \quad (5.3)$$

or

$$g = \max(|g_x|, |g_y|) \quad (5.4)$$

which are, on average, equally accurate (Föglein, 1983).

In the TM approach, edge orientation is estimated simply as that of the mask giving rise to the largest value of gradient in Eq. (5.1). In the DG approach, it is estimated vectorially by the more complex equation:

$$\theta = \arctan \frac{g_y}{g_x} \quad (5.5)$$

Clearly, DG equations ((5.2) and (5.5)) require considerably more computation than TM equation (5.1), although they are more accurate. However, in some situations orientation information is not required; in addition, image contrast may vary widely, so there may appear to be little gain from thresholding a more accurate estimate of g . This may explain why so many workers have employed the TM instead of the DG approach. Since both approaches essentially involve estimation of local intensity gradients, it is not surprising that TM masks often turn out to be identical to DG masks (see Tables 5.1 and 5.2).

Table 5.1 Masks of Well-known Differential Edge Operators

(a) Masks for the Roberts 2×2 operator

$$R_{x'} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad R_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

(b) Masks for the Sobel 3×3 operator

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(c) Masks for the Prewitt 3×3 “smoothed gradient” operator

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

In this table masks are presented in an intuitive format (viz. coefficients increasing in the positive x and y directions) by rotating the normal convolution format through 180° .

This convention is employed throughout this chapter. The Roberts 2×2 operator masks (a) can be taken as being referred to axes x' and y' at 45° to the usual x and y axes.

5.3 THE TEMPLATE MATCHING APPROACH

Table 5.2 shows four sets of well-known TM masks for edge detection. These masks were originally (Prewitt, 1970; Kirsch, 1971; Robinson, 1977) introduced on an intuitive basis, starting in two cases from the DG masks shown in [Table 5.1](#). In all cases the eight masks of each set are obtained from a given mask by permuting the mask coefficients cyclically. By symmetry, this is a good strategy for even permutations, but symmetry alone does not justify it for odd permutations: the situation is explored in more detail below.

Note first that four of the “3-level” and four of the “5-level” masks can be generated from the other four of their set by sign inversion. This means that in either case only four convolutions need to be performed at each pixel neighborhood, thereby saving computation. This is an obvious procedure if the basic idea of the TM approach is regarded as one of comparing intensity gradients in the eight directions. The two operators that do not employ this strategy were developed much earlier on some unknown intuitive basis.

Before proceeding, we note the rationale behind the Robinson “5-level” masks. These were intended (Robinson, 1977) to emphasize the weights of diagonal edges in order to compensate for the characteristics of the human eye, which tends to enhance vertical and horizontal lines in images. Normally, image analysis is concerned with computer interpretation of images, and an isotropic set of responses is required. Thus, the “5-level” operator is a special-purpose one that need not be discussed further.

Table 5.2 Masks of Well-known 3×3 Template Matching Edge Operators

	0°	45°
(a) Prewitt masks	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$
(b) Kirsch masks	$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}$	$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$
(c) Robinson “3-level” masks	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 1 \end{bmatrix}$
(d) Robinson “5-level” masks	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}$

The table illustrates only two of the eight masks in each set; the remaining masks can in each case be generated by symmetry operations. For the 3-level and 5-level operators, four of the eight available masks are inverted versions of the other four (see text).

These considerations show that the four template operators mentioned above have limited theoretical justification. It is therefore worth studying the situation in more depth (see [Section 5.4](#)).

5.4 THEORY OF 3×3 TEMPLATE OPERATORS

In what follows, it is assumed that eight masks are to be used, with angles differing by 45° . In addition, four of the masks differ from the others only in sign, since this seems unlikely to result in any loss of performance. Symmetry requirements then lead to the following masks for 0° and 45° , respectively.

$$\begin{bmatrix} -A & 0 & A \\ -B & 0 & B \\ -A & 0 & A \end{bmatrix} \quad \begin{bmatrix} 0 & C & D \\ -C & 0 & C \\ -D & -C & 0 \end{bmatrix}$$

It is clearly of great importance to design masks so that they give consistent responses in different directions. To find how this affects the mask coefficients, we employ the strategy of ensuring that intensity gradients follow the rules of vector addition. If the pixel intensity values within a 3×3 neighborhood are:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

the above masks will give the following estimates of gradient in the 0° , 90° , and 45° directions:

$$g_0 = A(c + i - a - g) + B(f - d) \quad (5.6)$$

$$g_{90} = A(a + c - g - i) + B(b - h) \quad (5.7)$$

$$g_{45} = C(b + f - d - h) + D(c - g) \quad (5.8)$$

If vector addition is to be valid, then:

$$g_{45} = \frac{g_0 + g_{90}}{\sqrt{2}} \quad (5.9)$$

Equating coefficients of a, b, \dots, i leads to the self-consistent pair of conditions:

$$C = \frac{B}{\sqrt{2}} \quad (5.10)$$

$$D = A\sqrt{2} \quad (5.11)$$

A further requirement is for the 0° and 45° masks to give equal responses at 22.5° . This can be shown to lead to the formula:

$$\frac{B}{A} = \sqrt{2} \frac{9t^2 - (14 - 4\sqrt{2})t + 1}{t^2 - (10 - 4\sqrt{2})t + 1} \quad (5.12)$$

where $t = \tan 22.5^\circ$, so that:

$$\frac{B}{A} = \frac{13\sqrt{2} - 4}{7} = 2.055 \quad (5.13)$$

We can now summarize our findings with regard to the design of TM masks. First, obtaining sets of masks by permuting coefficients “cyclically” in a square neighborhood is *ad hoc* and cannot be relied upon to produce useful results. Next, following the rules of vector addition and the need to obtain consistent responses in different directions, we have shown that ideal TM masks need to closely match the Sobel coefficients; we have also rigorously derived an accurate value for the ratio B/A .

Having obtained some insight into the process of designing TM masks for edge detection, we next move on to study the design of DG masks.

5.5 THE DESIGN OF DIFFERENTIAL GRADIENT OPERATORS

This section studies the design of DG operators. These include the Roberts 2×2 operator and the Sobel and Prewitt 3×3 operators (Roberts, 1965; Prewitt, 1970; for the Sobel operator see Pringle, 1969, Duda and Hart, 1973, p. 271) (see Table 5.1). The Prewitt or “gradient smoothing” type of operator has been extended to larger pixel neighborhoods by Prewitt (1970) and others (Brooks, 1978; Haralick, 1980) (see Table 5.3). In these instances the basic rationale is to model local edges by the best fitting plane over a convenient size of neighborhood. Mathematically, this amounts to obtaining suitably weighted averages to estimate slope in the x and y directions. As pointed out by Haralick (1980), the use of equally weighted averages to measure slope in a given direction is incorrect: the proper weightings to use are given by the masks listed in Table 5.3. Thus, the Roberts and Prewitt operators are apparently optimal, whereas the Sobel operator is not. This point is discussed in more detail below.

A full discussion of the edge detection problem involves consideration of the accuracy with which edge magnitude and orientation can be estimated when the local intensity pattern cannot be assumed to be planar. In fact, there have been a number of analyses of the angular dependencies of edge detection operators for a step edge approximation. In particular, O’Gorman (1978) considered the variation of estimated versus actual angle resulting from a step edge observed within a

Table 5.3 Masks for Estimating Components of Gradient in Square Neighborhoods

	\mathbf{M}_x	\mathbf{M}_y
(a) 2×2 neighborhood	$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$
(b) 3×3 neighborhood	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
(c) 4×4 neighborhood	$\begin{bmatrix} -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \end{bmatrix}$	$\begin{bmatrix} 3 & 3 & 3 & 3 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -3 & -3 & -3 & -3 \end{bmatrix}$
(d) 5×5 neighborhood	$\begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -2 & -2 & -2 & -2 & -2 \end{bmatrix}$

The masks provided in this table can be regarded as extended Prewitt masks. The 3×3 masks are Prewitt masks, included in this table for completeness. In all cases weighting factors have been omitted in the interests of simplicity, as they are throughout this chapter.

square neighborhood (see also Brooks, 1978): note that the case considered was that of a continuum rather than a discrete lattice of pixels. This was found to lead to a smooth variation with angular error varying from zero at 0° and 45° to a maximum of 6.63° at 28.37° (where the estimated orientation was 21.74°), the variation for angles outside this range being replicated by symmetry. Abdou and Pratt (1979) obtained similar variations for the Sobel and Prewitt operators in a discrete lattice, the respective maximum angular errors being 1.36° and 7.38° (Davies, 1984b). It seems that the Sobel operator has angular accuracy that is close to optimal because it is close to being a “truly circular” operator. This point is discussed in more detail in [Section 5.6](#).

5.6 THE CONCEPT OF A CIRCULAR OPERATOR

It has been stated above that when step edge orientation is estimated in a square neighborhood, an error of up to 6.63° can result. Such an error does not arise with a planar edge approximation, since fitting of a plane to a planar edge profile within a square window can be carried out exactly. Errors appear only when the edge profile differs from the ideal planar form, within the square neighborhood—with the step edge probably being something of a “worst case.”

One way to limit errors in the estimation of edge orientation might be to restrict observation of the edge to a circular neighborhood. In the continuous case this is sufficient to reduce the error to zero for all orientations, since symmetry dictates that there is only one way of fitting a plane to a step edge within a circular neighborhood, assuming that all planes pass through the same central point; the estimated orientation θ is then equal to the actual angle φ . A rigorous calculation along the lines indicated by Brooks (1976), which results in the following formula for a square neighborhood (O’Gorman, 1978):

$$\tan \theta = \frac{2 \tan \varphi}{3 - \tan^2 \varphi} \quad 0^\circ \leq \varphi \leq 45^\circ \quad (5.14)$$

leads to the following formula:

$$\tan \theta = \tan \varphi, \quad \text{i.e., } \theta = \varphi \quad (5.15)$$

for a circular neighborhood (Davies, 1984b). Similarly, zero angular error results from fitting a plane to an edge of *any* profile within a circular neighborhood, in the continuous approximation. Indeed, for an edge surface of arbitrary shape, the only problem is whether the mathematical best fit plane coincides with one that is subjectively desirable (and, if not, a fixed angular correction will be required). Ignoring such cases, the basic problem is how to approximate a circular neighborhood in a digitized image of small dimensions, containing typically 3×3 or 5×5 pixels.

To proceed systematically, we first recall a fundamental principle stated by Haralick (1980):

the fact that the slopes in two orthogonal directions determine the slope in any direction is well known in vector calculus. However, it seems not to be so well known in the image processing community.

Essentially, appropriate estimates of slopes in two orthogonal directions permit the slope in any direction to be computed. For this principle to apply, appropriate estimates of the slopes have first to be made: if the components of slope are inappropriate, they will not act as components of true vectors and the resulting estimates of edge orientation will be in error. This appears to be the main source of error with the Prewitt and other operators—it is not so much that the components of slope are in any instance incorrect, but rather that they are inappropriate for the purpose of vector computation since *they do not match one another adequately in the required way* (Davies, 1984b).

Following the arguments for the continuous case discussed earlier, slopes must be rigorously estimated within a circular neighborhood. Then the operator design problem devolves into determining how best it is to simulate a circular neighborhood on a discrete lattice so that errors are minimized. To carry this out, it is necessary to apply a close to circular weighting while computing the masks, so that correlations between the gradient weighting and circular weighting factors are taken properly into account.

5.7 DETAILED IMPLEMENTATION OF CIRCULAR OPERATORS

In practice, the task of computing angular variations and error curves has to be tackled numerically, dividing each pixel in the neighborhood into arrays of suitably small subpixels. Each subpixel is then assigned a gradient weighting (equal to the x or y displacement) and a neighborhood weighting (equal to 1 for inside and 0 for outside a circle of radius r). Clearly, the angular accuracy of “circular” differential gradient edge detection operators must depend on the radius of the circular neighborhood. In particular, poor accuracy would be expected for small values of r and reasonable accuracy for large values of r , as the discrete neighborhood approaches a continuum.

The results of such a study are presented in Fig. 5.2. The variations depicted represent RMS angular errors (Fig. 5.2(a)) and maximum angular errors (Fig. 5.2(b)) in the estimation of edge orientation. The structures on each variation are surprisingly smooth: they are so closely related and systematic that they can only represent statistics of the arrangement of pixels in neighborhoods of various sizes. Details of these statistics are discussed in the next section.

<u>r</u>	<u>rms error</u>
1.0	6.04
1.1	4.45
1.2	3.08
1.3	1.87
1.4	0.89
1.5	0.60
1.6	1.11
1.7	1.28
1.8	1.17
1.9	0.94
2.0	0.66
2.1	0.47
2.2	0.52
2.3	0.57
2.4	0.53
2.5	0.48
2.6	0.46
2.7	0.40
2.8	0.28
2.9	0.20
3.0	0.25
3.1	0.30
3.2	0.31
3.3	0.27
3.4	0.21
3.5	0.16
3.6	0.18
3.7	0.18

a

b

FIGURE 5.2

Variations in angular error as a function of radius r : (a) RMS angular error and (b) maximum angular error.

Overall, three features of Fig. 5.2 are noteworthy. First, as expected, there is a general trend to zero angular error as r tends to infinity. Second, there is a very marked periodic variation, with particularly good accuracy resulting where the circular operators best match the tessellation of the digital lattice. The third feature of interest is the fact that errors do not vanish for any finite value of r —clearly, the constraints of the problem do not permit more than the minimization of errors. These curves show that it is possible to generate a family of optimal operators (at the minima of the error curves), the first of which corresponds closely to an operator (the Sobel operator) that is known to be nearly optimal.

The variations shown in Fig. 5.2 can be explained (Davies, 1984b) as pixel centers lying in well-packed or “closed” bands approximating to continua—indicated by the low error points in Fig. 5.2—between which centers would be more loosely packed. Thus, we get the “closed band” operators listed in Table 5.4; their angular variations appear in Table 5.5. It is seen that the Sobel operator, which is already the most accurate of the 3×3 edge gradient operators suggested previously, can be made some 30% more accurate by adjusting its coefficients to make it more circular. In addition, the closed bands idea indicates that the corner pixels of 5×5 or larger operators are best removed altogether: not only does this require less computation, but also it actually improves performance. It also seems likely that this situation would apply for many other operators and would not be specific to edge detection.

Table 5.4 Masks of “Closed Band” Differential Gradient Edge Operators

(a) Band containing shells a–c (effective radius = 1.500)

$$\begin{bmatrix} -0.464 & 0.000 & 0.464 \\ -0.959 & 0.000 & 0.959 \\ -0.464 & 0.000 & 0.464 \end{bmatrix}$$

(b) Band containing shells a–e (effective radius = 2.121)

$$\begin{bmatrix} 0.000 & -0.294 & 0.000 & 0.294 & 0.000 \\ -0.582 & -1.000 & 0.000 & 1.000 & 0.582 \\ -1.085 & -1.000 & 0.000 & 1.000 & 1.085 \\ -0.582 & -1.000 & 0.000 & 1.000 & 0.582 \\ 0.000 & -0.294 & 0.000 & 0.294 & 0.000 \end{bmatrix}$$

(c) Band containing shells a–h (effective radius = 2.915)

$$\begin{bmatrix} 0.000 & 0.000 & -0.191 & 0.000 & 0.191 & 0.000 & 0.000 \\ 0.000 & -1.085 & -1.000 & 0.000 & 1.000 & 1.085 & 0.000 \\ -0.585 & -2.000 & -1.000 & 0.000 & 1.000 & 2.000 & 0.585 \\ -1.083 & -2.000 & -1.000 & 0.000 & 1.000 & 2.000 & 1.083 \\ -0.585 & -2.000 & -1.000 & 0.000 & 1.000 & 2.000 & 0.585 \\ 0.000 & -1.085 & -1.000 & 0.000 & 1.000 & 1.085 & 0.000 \\ 0.000 & 0.000 & -0.191 & 0.000 & 0.191 & 0.000 & 0.000 \end{bmatrix}$$

In all cases only the x-mask is shown: the y-mask may be obtained by a trivial symmetry operation. Mask coefficients are accurate to ~ 0.003 but would in normal practical applications be rounded to one- or two-figure accuracy.

Table 5.5 Angular Variations for the Best Operators Tested

Actual Angle (°)	Estimated Angle (°) ^a					
	Prew	Sob	a–c	circ	a–e	a–h
0	0.00	0.00	0.00	0.00	0.00	0.00
5	3.32	4.97	5.05	5.14	5.42	5.22
10	6.67	9.95	10.11	10.30	10.81	10.28
15	10.13	15.00	15.24	15.52	15.83	14.81
20	13.69	19.99	20.29	20.64	20.07	19.73
25	17.72	24.42	24.73	25.10	24.62	25.00
30	22.62	28.86	29.14	29.48	29.89	30.02
35	28.69	33.64	33.86	34.13	35.43	34.86
40	35.94	38.87	39.00	39.15	40.30	39.71
45	45.00	45.00	45.00	45.00	45.00	45.00
RMS error	5.18	0.73	0.60	0.53	0.47	0.19

Prew, Prewitt; Sob, Sobel; a–c, theoretical optimum—closed band containing shells a–c; circ, actual optimum circular operator (as defined by the first minimum in Fig. 5.2); a–e, theoretical optimum—closed band containing shells a–e; a–h, theoretical optimum—closed band containing shells a–h.

^aValues are accurate to within $\sim 0.02^\circ$ in each case.

Before leaving this topic, note that the optimal 3×3 masks obtained above numerically by consideration of circular operators are very close to those obtained purely analytically in [Section 5.4](#), for TM masks, following the rules of vector addition. In the latter case a value of 2.055 was obtained for the ratio of the two mask coefficients, whereas for circular operators the value $0.959/0.464 = 2.067 \pm 0.015$ is obtained. Clearly this is no accident, and it is very satisfying that a coefficient that was formerly regarded as *ad hoc* (Kittler, 1983) is in fact optimizable and can be obtained in closed form (see [Section 5.4](#)).

5.8 THE SYSTEMATIC DESIGN OF DIFFERENTIAL EDGE OPERATORS

The family of “circular” differential gradient edge operators studied in [Sections 5.6](#) and [5.7](#) incorporates only one design parameter—the radius r . Only a limited number of values of this parameter permit optimum accuracy for estimation of edge orientation to be attained.

It is worth considering what additional properties this one parameter can control and how it should be adjusted during operator design. In fact, it affects signal-to-noise ratio, resolution, measurement accuracy, and computational load. To understand this, note first that signal-to-noise ratio varies linearly with the radius of the circular neighborhood, since signal is proportional to area and

Gaussian noise is proportional to the square root of area. Likewise, the measurement accuracy is determined by the number of pixels over which averaging occurs and hence is proportional to operator radius. Resolution and “scale” also vary with radius, since relevant linear properties of the image are averaged over the active area of the neighborhood. Finally, computational load, and the associated cost of hardware for speeding up the processing, is generally at least in proportion to the number of pixels in the neighborhood, and hence proportional to r^2 .

Overall, the fact that four important parameters vary in a fixed way with the radius of the neighborhood means that there are exact tradeoffs between them and that improvements in some are only obtained by losses to others: from an engineering point of view, compromises between them will have to be made according to circumstances.

5.9 PROBLEMS WITH THE ABOVE APPROACH—SOME ALTERNATIVE SCHEMES

Although the above ideas may be interesting, they have their own inherent problems. In particular, they take no account of the displacement E of the edge from the center of the neighborhood, or of the effects of noise in biasing the estimates of edge magnitude and orientation. In fact, it is possible to show that a Sobel operator gives zero error in the estimation of step edge orientation under the following condition:

$$|\theta| \leq \arctan\left(\frac{1}{3}\right) \quad \text{and} \quad |E| \leq \frac{(\cos \theta - 3 \sin |\theta|)}{2} \quad (5.16)$$

Furthermore, for a 3×3 operator of the form

$$\begin{bmatrix} -1 & 0 & 1 \\ -B & 0 & B \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & B & 1 \\ 0 & 0 & 0 \\ -1 & -B & -1 \end{bmatrix}$$

applied to the edge

a	$a + h(0.5 - E \sec \theta + \tan \theta)$	$a + h$
a	$a + h(0.5 - E \sec \theta)$	$a + h$
a	$a + h(0.5 - E \sec \theta - \tan \theta)$	$a + h$

Lyvers and Mitchell (1988) found that the estimated orientation is:

$$\varphi = \arctan\left[\frac{2B \tan \theta}{B + 2}\right] \quad (5.17)$$

which immediately shows why the Sobel operator should give zero error for a specific range of θ and E . However, this is somewhat misleading, since considerable errors arise outside this region. Not only do they arise when $E = 0$, as assumed in the foregoing sections, but also they vary strongly with E . Indeed, the maximum

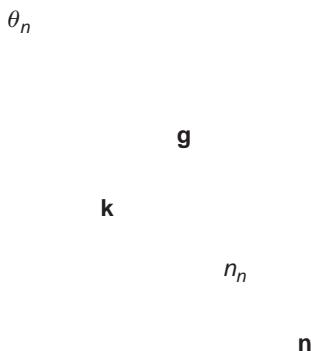
errors for the Sobel and Prewitt operators rise to 2.90° and 7.43° , respectively in this more general case (the corresponding RMS errors are 1.20° and 4.50°). Hence, a full analysis should be performed to determine how to reduce the maximum and average errors. Lyvers and Mitchell (1988) carried out an empirical analysis and constructed a lookup table with which to correct the orientations estimated by the Sobel operator, the maximum error being reduced to 2.06° .

Another scheme that reduces the error is the moment-based operator of Reeves et al. (1983). This leads to Sobel-like 3×3 masks, which are essentially identical to the 3×3 masks of Davies (1984b), both having $B = 2.067$ (for $A = 1$). However, the moment method can also be used to estimate the edge position E if additional masks are used to compute second-order moments of intensity. Hence, it is possible to make a very significant improvement in performance by using a 2-D lookup table to estimate orientation: the result is that the maximum error is reduced from 2.83° to 0.135° for 3×3 masks and from 0.996° to 0.0042° for 5×5 masks.

However, Lyvers and Mitchell (1988) found that much of this additional accuracy is lost in the presence of noise, and RMS standard deviations of edge orientation estimates are already around 0.5° for 3×3 operators at 40 dB signal-to-noise ratios. The reasons for this are quite simple. Each pixel intensity has a noise component that induces errors in its weighted mask components; the combined effects of these errors can be estimated assuming that they arise independently, so that their variances add (Davies, 1987c). Thus, noise contributions to the x and y components of gradient can be computed. These provide estimates for the components of noise along and perpendicular to the edge gradient vector (Fig. 5.3): the edge orientation for a Sobel operator turns out to be affected by an amount $\sqrt{12}\sigma/4h$ radians, where σ is the standard deviation on the pixel intensity values and h is the edge contrast. This explains the angular errors given by Lyvers and Mitchell, if Pratt's (2001) definition of signal-to-noise ratio (in dB) is used:

$$S/N = 20 \log_{10} \left(\frac{h}{\sigma} \right) \quad (5.18)$$

A totally different approach to edge detection was developed by Canny (1986). He used functional analysis to derive an optimal function for edge detection, starting with three optimization criteria—good detection, good localization, and only one response per edge under white noise conditions. The analysis is too technical to be discussed in detail here. However, the 1-D function found by Canny is accurately approximated by the derivative of a Gaussian: this is then combined with a Gaussian of identical σ in the perpendicular direction, truncated at 0.001 of its peak value, and split into suitable masks. Underlying this method is the idea of locating edges at local maxima of gradient magnitude for a Gaussian-smoothed image. In addition, the Canny implementation employs a hysteresis operation (Section 5.10) on edge magnitude in order to make edges

**FIGURE 5.3**

Calculating angular errors arising from noise: **g**, intensity gradient vector; **n**, noise vector; **k**, resultant of intensity gradient and noise vector; n_n , normal component of noise; θ_n , noise-induced orientation error.

reasonably connected. Finally, a multiple-scale method is employed to analyze the output of the edge detector. It is discussed in more detail below. Lyvers and Mitchell (1988) tested the Canny operator and found it to be significantly less accurate for orientation estimation than for the moment and IDD operators described above. In addition, it needed to be implemented using 180 masks and hence took enormous computation time, although many practical implementations of this operator are much faster than this early paper indicates.¹ One such implementation is described in [Section 5.11](#).

An operator that has been of great historical importance is that of Marr and Hildreth (1980). The motivation for the design of this operator was the modeling of certain psychophysical processes in mammalian vision. The basic rationale is to find the Laplacian of the Gaussian-smoothed ($\nabla^2 G$) image and then to obtain a “raw primal sketch” as a set of zero-crossing lines. The Marr–Hildreth operator does not use any form of threshold since it merely assesses where the $\nabla^2 G$ image passes through zero. This feature is attractive, since working out threshold values is a difficult and unreliable task. However, the Gaussian smoothing procedure can be applied at a variety of scales, and in one sense the scale is a new parameter that substitutes for the threshold. In fact, a major feature of the Marr–Hildreth approach, which has been very influential in later work (Witkin, 1983; Bergholm,

¹In fact, it is nowadays necessary to ask “Which Canny?”, as there are a great many implementations of it, and this leads to problems for any realistic comparison between operators.

1986), is the fact that zero crossings can be obtained at several scales, giving the potential for more powerful semantic processing: clearly, this necessitates finding means for combining all the information in a systematic and meaningful way. This may be carried out by a bottom-up or top-down approach, and there has been much discussion in the literature about methods for carrying out these processes. However, it is worth remarking that in many (especially industrial inspection) applications, one is interested in working at a particular resolution, and considerable savings in computation can then be made. It is also noteworthy that the Marr–Hildreth operator is reputed to require neighborhoods of at least 35×35 for proper implementation (Brady, 1982). Nevertheless, other workers have implemented the operator in much smaller neighborhoods, down to 5×5 . Wiejak et al. (1985) showed how to implement the operator using linear smoothing operations to save computation. Lyvers and Mitchell (1988) reported orientation accuracies using the Marr–Hildreth operator that are not especially high (2.47° for a 5×5 operator and 0.912° for a 7×7 operator, in the absence of noise).

It has been noted above that those edge detection operators that are applied at different scales lead to different edge maps at different scales. In such cases, certain edges that are present at lower scales disappear at larger scales; in addition, edges that are present at both low and high scales appear shifted or merged at higher scales. Bergholm (1986) demonstrated the occurrence of elimination, shifting, and merging, whereas Yuille and Poggio (1986) showed that edges that are present at low resolution should not disappear at some higher resolution. These aspects of edge location are by now well understood.

In what follows, we first consider hysteresis thresholding, a process already mentioned with regard to the Canny operator. In [Section 5.11](#) we give a fuller appraisal of the Canny operator and show detailed results on real images. Then in [Section 5.12](#) we consider the Laplacian type of operator. In [Sections 5.13 and 5.14](#), we show how the well-known active contour (snake) concept can be used to lead to connected object boundaries. In [Section 5.15](#) we outline the level set approach. Finally, in [Section 5.16](#) we discuss the aims and methodology of the graph cut approach to producing connected object boundaries: it should be noticed that this method essentially dispenses with the usual ideas of edge detection and regional analysis and aims to give an integrated, generalized methodology for segmentation.

5.10 HYSTERESIS THRESHOLDING

The concept of hysteresis thresholding is a general one and can be applied in a range of applications, including both image and signal processing. In fact, the Schmitt trigger is a very widely used electronic circuit for converting a varying voltage into a pulsed (binary) waveform. In the latter case there are two thresholds, and the input has to rise above the upper threshold before the output is allowed to switch on, and has to fall below the lower threshold before the output is allowed to switch off. This gives considerable immunity against noise in the input waveform—far more than

where the difference between the upper and lower switching thresholds is zero (the case of zero hysteresis), since then a small amount of noise can cause an undue amount of switching between the upper and lower output levels.

When the concept is applied in image processing it is usually with regard to edge detection, in which case there is an exactly analogous 1-D waveform to be negotiated around the boundary of an object—although, as we shall see, some specifically 2-D complications arise. The basic rule is to threshold the edge at a high level, and then to allow extension of the edge down to a lower level threshold, but only adjacent to points that have already been assigned edge status.

Figure 5.4 shows the results of making tests on the edge gradient image in Fig. 7.4(b). Figure 5.4(a) and (b) shows the result of thresholding at the upper and lower hysteresis levels respectively and Fig. 5.4(c) shows the result of hysteresis thresholding using these two levels. For comparison, Fig. 5.4(d) shows the effect of thresholding at a suitably chosen intermediate level. Note that isolated edge points within the object boundaries are ignored by hysteresis thresholding, although noise spurs can occur and are retained. We can envision the process of hysteresis thresholding in an edge image as the location of points that:

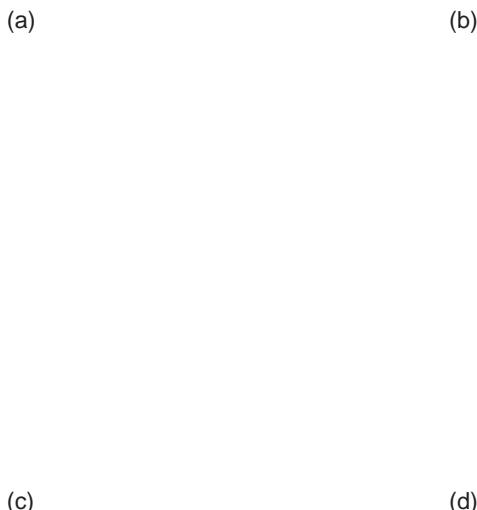
1. form a superset of the upper threshold edge image.
2. form a subset of the lower threshold edge image.
3. form that subset of the lower threshold image that is connected to points in the upper threshold image via the usual rules of connectedness (Chapter 9).

Clearly, edge points survive only if they are seeded by points in the upper threshold image.

Although the result in Fig. 5.4(c) is better than in Fig. 5.4(d), in that gaps in the boundaries are eliminated or reduced in length, in a few cases noise spurs are introduced. Nevertheless, the aim of hysteresis thresholding is to obtain a better balance between false positives and false negatives by exploiting connectedness in the object boundaries. Indeed, if managed correctly, the additional parameter will normally lead to a net (average) reduction in boundary pixel classification error. However, there are few simple guidelines for selection of hysteresis thresholds, apart from the following:

1. Use a pair of hysteresis thresholds that provides immunity against the known range of noise levels.
2. Choose the lower threshold to limit the possible extent of noise spurs (in principle, the lowest threshold subset that contains *all* true boundary points).
3. Select the upper threshold to guarantee as far as possible the seeding of important boundary points (in principle, the highest threshold subset that is connected to *all* true boundary points).

Unfortunately, in the limit of high signal variability, rules 2 and 3 appear to suggest eliminating hysteresis altogether! Ultimately, this means that the only rigorous way of treating the problem is to perform a complete statistical analysis of false positives and false negatives for a large number of images in any new application.

**FIGURE 5.4**

Effectiveness of hysteresis thresholding. This figure shows tests made on the edge gradient image of Fig. 7.4(b). (a) Effect of thresholding at the upper hysteresis level. (b) Effect of thresholding at the lower hysteresis level. (c) Effect of hysteresis thresholding. (d) Effect of thresholding at an intermediate level.

5.11 THE CANNY OPERATOR

Since its publication in 1986 the Canny operator (Canny, 1986) has become one of the most widely used edge detection operators—and for good reason, as it seeks to get away from a tradition of mask-based operators, many of which can hardly be regarded as “designed”, into one that is entirely principled and fully integrated. Intrinsic to the method is that of carefully specifying the spatial bandwidth within which it is expected to work, and also the exclusion of unnecessary thresholds, while permitting thin line structures to emerge and ensuring that they

are connected together as far as possible and indeed are meaningful at the particular scale and bandwidth. As a result of these considerations, the method involves a number of stages of processing:

1. Low-pass spatial frequency filtering
2. Application of first-order differential masks
3. Nonmaximum suppression involving subpixel interpolation of pixel intensities
4. Hysteresis thresholding

In principle, low-pass filtering needs to be carried out by Gaussian convolution operators for which the standard deviation (or spatial bandwidth) σ is known and prespecified. Then first-order differential masks need to be applied: for this purpose, the Sobel operator is acceptable. In this context note that the Sobel operator masks can be regarded as convolutions (\otimes) of a basic $\begin{bmatrix} -1 & 1 \end{bmatrix}$ type of mask with a $\begin{bmatrix} 1 & 1 \end{bmatrix}$ smoothing mask. Thus, taking the Sobel x -derivative we have:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad (5.19)$$

where

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} \quad (5.20)$$

and

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} \quad (5.21)$$

These equations make it clear that the Sobel operator itself includes a considerable amount of low-pass filtering, so the amount of additional filtering needed in stage 1 can reasonably be reduced. Another thing to bear in mind is that low-pass filtering can itself be carried out by a smoothing mask of the type shown in Fig. 5.5(b), and it is interesting how close this mask is to the full 2-D Gaussian shown in Fig. 5.5(a). Note also that the bandwidth of the mask in Fig. 5.5(b) is exactly known (it is 0.707), and when combined with that of the Sobel the overall bandwidth becomes almost exactly 1.0.

Next, we turn our attention to stage 3—that of nonmaximum suppression. For this purpose we need to determine the local edge normal direction using Eq. (5.5), and move either way along the normal to determine whether the current location is or is not a local maximum along it. If it is not, we suppress the edge output at the current location, only retaining edge points that are proven local maxima along the edge normal. Since only one point along this direction should be a local maximum, this procedure will necessarily thin the grayscale edges to unit width. Here a slight problem arises in that the edge normal direction will in general not pass through the centers of the adjacent pixels, and the Canny method requires the intensities along the normal to be estimated by interpolation. In a

0.000 0.000 0.004 0.008 0.004 0.000 0.000	
0.000 0.016 0.125 0.250 0.125 0.016 0.000	
0.004 0.125 1.000 2.000 1.000 0.125 0.004	1 2 1
0.008 0.250 2.000 4.000 2.000 0.250 0.008	2 4 2
0.004 0.125 1.000 2.000 1.000 0.125 0.004	1 2 1
0.000 0.016 0.125 0.250 0.125 0.016 0.000	
0.000 0.000 0.004 0.008 0.004 0.000 0.000	

(a)

(b)

FIGURE 5.5

Exactness of the well-known 3×3 smoothing kernel. This figure shows the Gaussian-based smoothing kernel (a) that is closest to the well-known 3×3 smoothing kernel (b) over the central (3×3) region. For clarity, neither is normalized by the factor $1/16$. The larger Gaussian envelope drops to 0.000 outside the region shown and integrates to 18.128 rather than 16. Hence, the kernel in (b) can be said to approximate a Gaussian within $\sim 13\%$. Its actual standard deviation is 0.707 compared with 0.849 for the Gaussian.

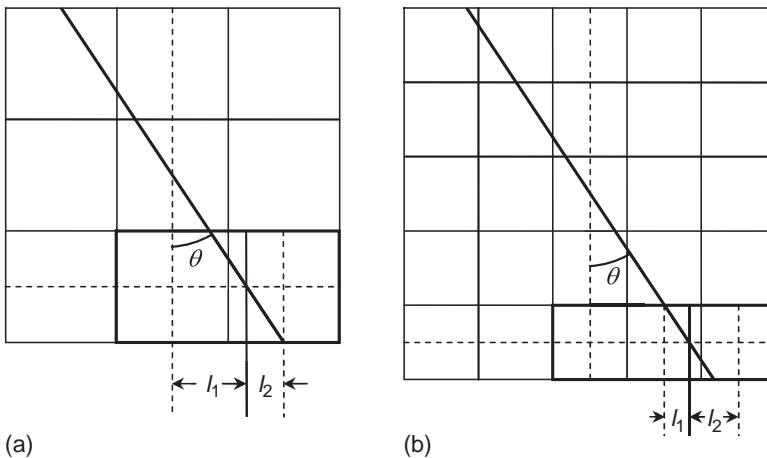
3×3 neighborhood this is simply achieved, as the edge normal in any octant will have to lie within a given pair of pixels, as shown in Fig. 5.6(a). In a larger neighborhood, interpolation can take place between several pairs of pixels. For example, in a 5×5 neighborhood, it will have to be determined which one of the two pairs is relevant (Fig. 5.6(b)), and an appropriate interpolation formula applied. However, it could be construed that there is no need to use larger neighborhoods, as a 3×3 neighborhood will contain all the relevant information, and given enough presmoothing in stage 1, negligible loss of accuracy will result. Of course, if impulse noise is present, this could lead to serious error, but low-pass filtering is in any case not guaranteed to eliminate impulse noise, so no special loss results from using the smaller neighborhood for nonmaximum suppression. Such considerations need to be examined carefully in the light of the particular image data and the noise it contains. Figure 5.6 shows the two distances l_1 and l_2 that have to be determined. The pixel intensity along the edge normal is given by weighting the corresponding pixel intensities in *inverse* proportion to the distances:

$$I = \frac{l_2 I_1 + l_1 I_2}{l_1 + l_2} = (1 - l_1)I_1 + l_1 I_2 \quad (5.22)$$

where

$$l_1 = \tan \theta \quad (5.23)$$

This brings us to the final stage, that of hysteresis thresholding. By this point as much as possible has been achieved without applying thresholds, and it becomes necessary to take this final step. However, by applying the two hysteresis thresholds, it is intended to limit the damage that can be caused by a single threshold and repair it with another: that is to say, select the upper threshold to ensure capturing edges that are reliable and then select other points that have high likelihood of being viable edge points because they are adjacent to edge points of

**FIGURE 5.6**

Pixel interpolation in the Canny operator. (a) Interpolation between the two highlighted pixels at the bottom right in a 3×3 neighborhood. (b) Interpolation in a 5×5 neighborhood: note that two possibilities exist for interpolating between pairs of adjacent pixels, the relevant distances being marked for the one on the right.

known reliability. In fact, this is still somewhat *ad hoc*, but in practice it gives quite good results. A simple rule for choice of the lower threshold is that it should be about half the upper threshold. Again, this is only a rule of thumb, and it has to be examined carefully in the light of the particular image data.

Figures 5.7 and 5.8 show results for the Canny operator at various stages. They also show comparisons for various thresholds. In particular, both figures show the effects of (e) hysteresis thresholding, (f) single thresholding at the lower level, and (g) single thresholding at the upper level. The evidence is that hysteresis thresholding is usually more reliable and more coherent than single-level thresholding, in the sense of giving fewer false or misleading results.

5.12 THE LAPLACIAN OPERATOR

An edge detector such as the Sobel is a first derivative operator, whereas the Laplacian is a second derivative operator, and as such it is sensitive only to changes in intensity gradient. In 2-D its standard (mathematical) definition is given by:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad (5.24)$$

Localized masks for computing Laplacian output can be derived by taking difference of Gaussian (DoG) kernels using two Gaussians of different bandwidths; for details of this procedure, see Section 6.7.3. This gives them an isotropic 2-D

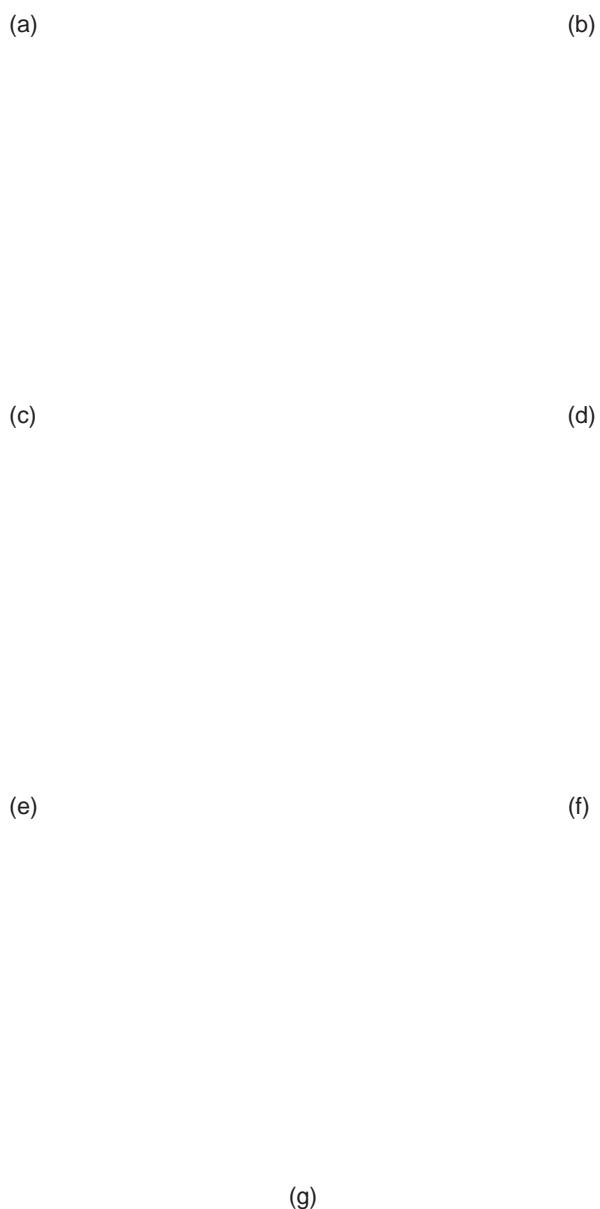


FIGURE 5.7

Application of the Canny edge detector. (a) Original image. (b) Smoothed image. (c) Result of applying Sobel operator. (d) Result of nonmaximum suppression. (e) Result of hysteresis thresholding. (f) Result of thresholding only at the lower threshold level. (g) Result of thresholding at the upper threshold level. Note that there are fewer false or misleading outputs in (e) than would result from using a single threshold.

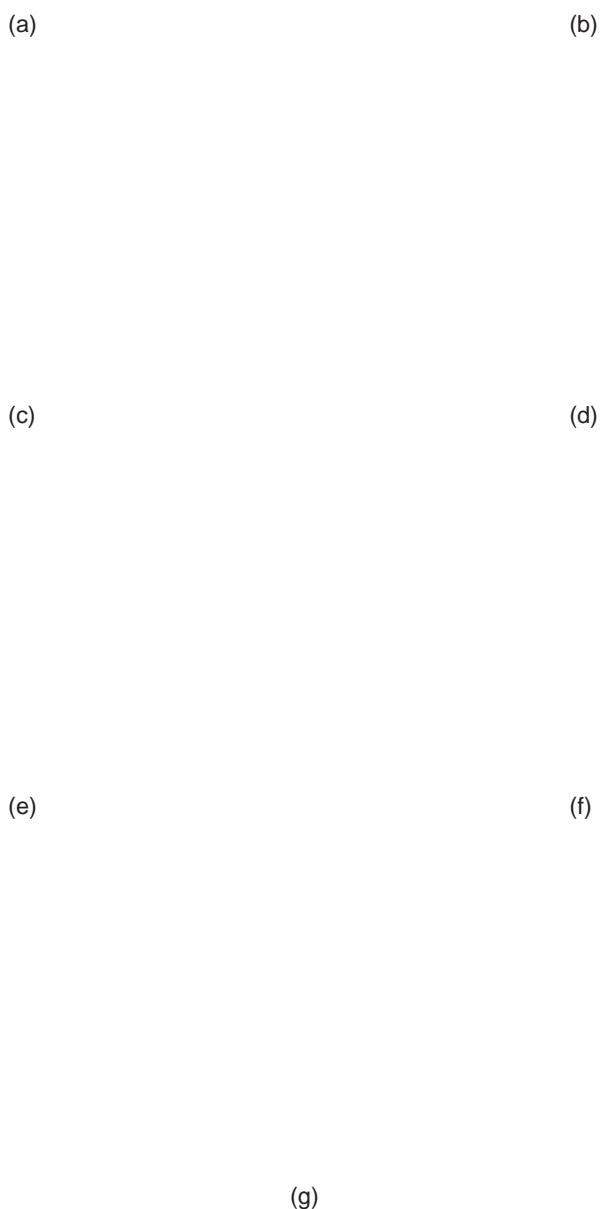


FIGURE 5.8

Another application of the Canny edge detector. (a) Original image. (b) Smoothed image. (c) Result of applying Sobel operator. (d) Result of nonmaximum suppression. (e) Result of hysteresis thresholding. (f) Result of thresholding only at the lower threshold level. (g) Result of thresholding at the upper threshold level. Again there are fewer false or misleading outputs in (e) than would result from using a single threshold.

profile, with a positive center and a negative surround. This shape can be approximated in 3×3 windows by masks such as the following:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (5.25)$$

Clearly, this mask is far from isotropic: nevertheless it exhibits many of the properties of larger masks, such as DoG kernels, that are much more accurately isotropic.

Here we present only an outline of the properties of this type of operator. These can be seen in Fig. 5.9. First, note that the Laplacian output ranges from positive to negative: hence, in Fig. 5.9(c) it is presented on a medium-gray background, which indicates that on the exact edge of an object the Laplacian output is actually zero, as stated earlier. This is made clearer in Fig. 5.9(d), where the magnitude of the Laplacian output is shown. It is seen that edges are highlighted by strong signals just inside and just outside the edge locations that are located by a Sobel or Canny operator (see Fig. 5.9(b)). Ideally this effect is symmetrical, and if the Laplacian is to be used for edge detection, zero crossings of the output will have to be located. However, in spite of preliminary smoothing of the image (Fig. 5.9(a)), the background in Fig. 5.9(d) has a great deal of noise in the background, and attempting to find zero crossings will therefore lead to a lot of noise being detected in addition to the edge points: in fact, it is well known that differentiation (especially double differentiation, as here) tends to accentuate noise. Nevertheless, this approach has been used highly successfully, usually with DoG operators working in much larger windows. Indeed, with much larger windows there will be a good number of pixels lying very near the zero crossings, and it will be possible to discriminate much more successfully between them and the pixels merely having low Laplacian output. A particular advantage of using Laplacian zero crossings is that theoretically they are bound to lead to closed contours around objects (albeit noise signals will also have their own separate closed contours).

5.13 ACTIVE CONTOURS

Active contour models (also known as “deformable contours” or “snakes”) are widely used for systematically refining object contours. The basic concept is to obtain a complete and accurate outline of an object that may be ill-defined in places, whether through lack of contrast, or noise or fuzzy edges. A starting approximation is made, either by instituting a large contour that may be shrunk to size, or a small contour that may be expanded suitably, until its shape matches that of the object. In principle, the initial boundary can be rather arbitrary, whether mostly outside or within the object in question. Then its shape is made to evolve subject to an energy minimization process: on the one hand it is desired to minimize the *external*

(a)

(b)

(c)

(d)

FIGURE 5.9

Comparison of Sobel and Laplacian outputs. (a) Pre-smoothed version of original image. (b) Result of applying Sobel operator. (c) Result of applying Laplacian operator. Because the Laplacian output can be positive or negative, the output in (c) is displayed relative to a medium (128)-gray-level background. (d) Absolute magnitude Laplacian output. For clarity, (c) and (d) have been presented at increased contrast. Note that the Laplacian output in (d) gives double edges—one just inside and one just outside the edge position indicated by a Sobel or Canny operator. (To find edges using a Laplacian, zero crossings have to be located.) Both the Sobel and the Laplacian used here operate within a 3×3 window.

energy corresponding to imperfections in the degree of fit, and on the other hand it is desired to minimize the *internal* energy, so that the shape of the snake does not become unnecessarily intricate, e.g., by taking on any of the characteristics of image noise. There are also model constraints that are represented in the formulation as contributions to the external energy: typical of such constraints is that of preventing the snake from moving into prohibited regions, such as beyond the image boundary, or, for a moving vehicle, off the region of the road.

The snake's internal energy includes elastic energy, which might be needed to extend or compress it, and bending energy. If no bending energy terms were included, sharp corners and spikes in the snake would be free to occur with no restriction. Similarly, if no elastic energy terms were included, the snake would be permitted to grow or shrink without penalty.

The image data are normally taken to interact with the snake via three main types of image feature—lines, edges, and terminations (the last can be line terminations or corners). Various weights can be given to these features according to the behavior required of the snake. For example, it might be required to hug edges and go around corners, and only to follow lines in the absence of edges: so the line weights would be made much lower than the edge and corner weights.

These considerations lead to the following breakdown of the snake energy:

$$\begin{aligned} E_{\text{snake}} &= E_{\text{internal}} + E_{\text{external}} \\ &= E_{\text{internal}} + E_{\text{image}} + E_{\text{constraints}} \\ &= E_{\text{stretch}} + E_{\text{bend}} + E_{\text{line}} + E_{\text{edge}} + E_{\text{term}} + E_{\text{repel}} \end{aligned} \quad (5.26)$$

The energies are written down in terms of small changes in position $\mathbf{x}(s) = (x(s), y(s))$ of each point on the snake, the parameter s being the arc length distance along the snake boundary. Thus, we have:

$$E_{\text{stretch}} = \int \kappa(s) \|\mathbf{x}_s(s)\|^2 ds \quad (5.27)$$

and

$$E_{\text{bend}} = \int \lambda(s) \|\mathbf{x}_{ss}(s)\|^2 ds \quad (5.28)$$

where the suffices s and ss imply first- and second-order differentiation, respectively. Similarly, E_{edge} is calculated in terms of the intensity gradient magnitude $|\text{grad } I|$, leading to:

$$E_{\text{edge}} = - \int \mu(s) \|\text{grad } I\|^2 ds \quad (5.29)$$

where $\mu(s)$ is the edge weighting factor.

The overall snake energy is obtained by summing the energies for all positions on the snake: a set of simultaneous differential equations is then set up to minimize the total energy. This process is not discussed in detail due to the space limitation. Suffice it to say that the equations cannot be solved analytically, and recourse has to be made to iterative numerical solution, during which the shape of

the snake evolves from some high-energy initialization state to the final low-energy equilibrium state, defining the contour of interest in the image.

In the general case, there are several possible complications to be tackled:

1. Several snakes may be required to locate an initially unknown number of relevant image contours.
2. Different types of snake will need different initialization conditions.
3. Snakes will sometimes have to split up as they approach contours that turn out to be fragmented.

There are also procedural problems. The intrinsic snake concept is that of well-behaved differentiability. However, lines, edges, and terminations are usually highly localized, so there is no means by which a snake even a few pixels away could be expected to learn about them and hence to move toward them. In these circumstances the snake would “thrash around” and fail to systematically zone in on a contour representing a global minimum of energy. To overcome this problem, smoothing of the image is required, so that edges can communicate with the snake some distance away, and the smoothing must gradually be reduced as the snake nears its target position. Ultimately, the problem is that the algorithm has no high-level appreciation of the overall situation, but merely reacts to a conglomerate of local pieces of information in the image: this makes segmentation using snakes somewhat risky despite the intuitive attractiveness of the concept.

In spite of these potential problems, a valuable feature of the snake concept is that, if set up correctly, the snake can be rendered insensitive to minor discontinuities in a boundary: it is important as this makes the snake capable of negotiating practical situations such as fuzzy or low contrast edges, or places where small artifacts get in the way (this may happen with resistor leads, for example); this capability is possible because the snake energy is set up *globally*—quite unlike the situation for boundary tracking where error propagation can cause wild deviations from the desired path. The reader is referred to the abundant literature on the subject, not only to clarify the basic theory (Kass and Witkin, 1987; Kass et al., 1988), but also to find how it may be made to work well in real situations.

5.14 PRACTICAL RESULTS OBTAINED USING ACTIVE CONTOURS

In this section we briefly explore a simple implementation of the active contour concept. Arguably, the implementation chosen is among the simplest that will work in practical situations while still adhering to the active contour concept. To make it work without undue complication or high levels of computation, a “greedy” algorithm is used, i.e., one that makes local optimizations (energy minimizations) in the expectation that this will result in global optimization. Naturally, it could lead to

solutions that do not correspond to absolute minima of the energy function, although this is by no means a problem that is caused solely by using a greedy algorithm, as almost all forms of iterative energy minimization method can fall into this trap.

The first thing to do when devising such an algorithm is to interpret the theory in practical terms. Thus, we rewrite the snake stretch function (Eq. (5.27)) in the discrete form:

$$E_{\text{stretch}} = \sum_{i=1}^N \kappa \| \mathbf{x}_i - \mathbf{x}_{i+1} \|^2 \quad (5.30)$$

where there are N snake points \mathbf{x}_i , $i = 1, \dots, N$: note that this set must be accessed cyclically. In addition, when using a greedy algorithm and updating the position of the i th snake point, the following local form of Eq. (5.30) has to be used:

$$\varepsilon_{\text{stretch},i} = \kappa \left(\| \mathbf{x}_i - \mathbf{x}_{i-1} \|^2 + \| \mathbf{x}_i - \mathbf{x}_{i+1} \|^2 \right) \quad (5.31)$$

Unfortunately, although this function causes the snake to be tightened, it can also result in clustering of snake points. To avoid this, the following alternative form can be useful:

$$\varepsilon_{\text{stretch},i} = \kappa \left[(d - \| \mathbf{x}_i - \mathbf{x}_{i-1} \ |)^2 + (d - \| \mathbf{x}_i - \mathbf{x}_{i+1} \ |)^2 \right] \quad (5.32)$$

where d is a fixed number representing the smallest likely value of the mean distance between adjacent pairs of snake points for the given type of target object. In the implementation used in Fig. 5.10, d had the noncritical value of 8 pixels; interestingly, this also resulted in faster convergence toward the final form of the snake, as it was encouraged to move further to minimize the magnitudes of the terms in round brackets.

The contour shown in Fig. 5.10 fills the concavity at the top right, but hardly moves into the concavity at the bottom because of a low contrast shadow edge: note that more or less influence by weak edges can readily be obtained by adjusting the grad^2 coefficient μ in Eq. (5.29). Elsewhere the snake ends up with almost exact adherence to the object boundary. The snake shown in the figure employs $p = 40$ points, and $r = 60$ iterations are needed to bring it to its final position. In each iteration, the greedy optimization for each snake point is over an $n \times n$ pixel region with $n = 11$. Overall, the computation time is controlled by and essentially proportional to the quantity prn^2 .

The final contour in Fig. 5.10(d) shows the result of using an increased number of initialization points and joining the final locations to give a connected boundary: some of the remaining deficiencies could be reduced by fitting with splines or other means instead of simply joining the dots.

As indicated earlier, this was a simple implementation—so much so that no attempt was made to take corners and bends into account, although in the case shown in Fig. 5.10 no disadvantages or deviations can be seen, except in Fig. 5.10 (d). Clearly, a suitable redesign involving additional energy terms would have to be included to cope with more complex image data. It is interesting that so much

(a)

(b)

(c)

(d)

FIGURE 5.10

Generation of active contour models (snakes). (a) Original picture with snake initialization points near the image boundary; the final snake locations hug the outside of the object but hardly penetrate the large concavity at the bottom: they actually lie approximately along a weak shadow edge. (b) Result of smoothing and application of Sobel operator to (a); the snake algorithm used this image as its input. The snake output is superimposed in black on (b), so that the high degree of co-location with the edge maxima can readily be seen. (c) Intermediate result, after half (30) the total number of iterations (60): this illustrates that after one edge point has been captured, it becomes much easier for other such points to be captured. (d) Result of using an increased number of initialization points and joining the final locations to give a connected boundary: some remanent deficiencies are evident.

can be achieved by just two terms, *viz.* the stretch and edge terms in Eq. (5.26). However, an important factor in getting the greedy algorithm to work optimally one snake point at a time is the need to include the energies for *both* adjacent links (as in Eqs. (5.31) and (5.32)) so as to limit bias and other complications.

5.15 THE LEVEL SET APPROACH TO OBJECT SEGMENTATION

Although the active contour approach described in the previous two sections can be effective in many situations, it nevertheless has several drawbacks (Cremers et al., 2007):

1. There is the possibility of snake self-intersection.
2. Topological changes like splitting or merging of the evolving contour are not allowed.
3. The algorithm is highly dependent on the initialization, and this can result in the snake being biased or getting stuck in local minima.
4. Snakes lack meaningful probabilistic interpretation, so generalizing their action to cover color, texture, or motion is not straightforward.

The level set approach is intended to remedy these deficiencies. The basic approach is to work with whole regions rather than edges, and to evolve an “embedding function” in which contours are represented implicitly rather than directly. In fact, the embedding function is a function $\varphi(\mathbf{x}, t)$ and the contour is defined as the zero level of this function:

$$C(t) = \{\mathbf{x} | \varphi(\mathbf{x}, t) = 0\} \quad (5.33)$$

For a contour that evolves (by gradient descent) along each local normal \mathbf{n} with a speed F , we have:

$$\varphi(C(t), t) = 0 \quad (5.34)$$

which leads to:

$$\frac{d}{dt} \varphi(C(t), t) = \nabla \varphi \frac{\partial C}{\partial t} + \frac{\partial \varphi}{\partial t} = F \nabla \varphi \cdot \mathbf{n} + \frac{\partial \varphi}{\partial t} = 0 \quad (5.35)$$

Substituting for \mathbf{n} using:

$$\mathbf{n} = \frac{\nabla \varphi}{|\nabla \varphi|} \quad (5.36)$$

we obtain:

$$\frac{\partial \varphi}{\partial t} = - |\nabla \varphi| F \quad (5.37)$$

Next we need to substitute for F . Following Caselles et al. (1997), we have:

$$\frac{\partial \varphi}{\partial t} = |\nabla \varphi| \operatorname{div} \left(g(I) \frac{\nabla \varphi}{|\nabla \varphi|} \right) \quad (5.38)$$

where $g(I)$ is a generalized version of $|\nabla \varphi|$ in the snake potential.

Note that because the contour C is not mentioned explicitly, the updating takes place over all pixels, thereby involving a great many useless calculations: thus, the “narrow band” method was devised to overcome this problem, and involves updating only in a narrow strip around the current contour. However, the need to continually update this strip means that the computational load remains considerable. An alternative approach is the “fast marching” method, which essentially propagates a solution rapidly along an active wavefront, while leaving pixel values frozen behind it. As a result, this method involves maintaining the sign of the speed values F . The Hermes algorithm of Paragios and Deriche (2000) seeks to combine the two approaches. It aims at a *final* solution where all the necessary constraints are fulfilled while maintaining these constraints only loosely at intermediate stages. The overall front propagation algorithm overcomes the four problems mentioned above: in particular, it is able to track nonrigid objects, copes with splitting and merging, and has low computational cost. The paper confirms these claims by showing traffic scenes in which vehicles and pedestrians are successfully tracked.

5.16 THE GRAPH CUT APPROACH TO OBJECT SEGMENTATION

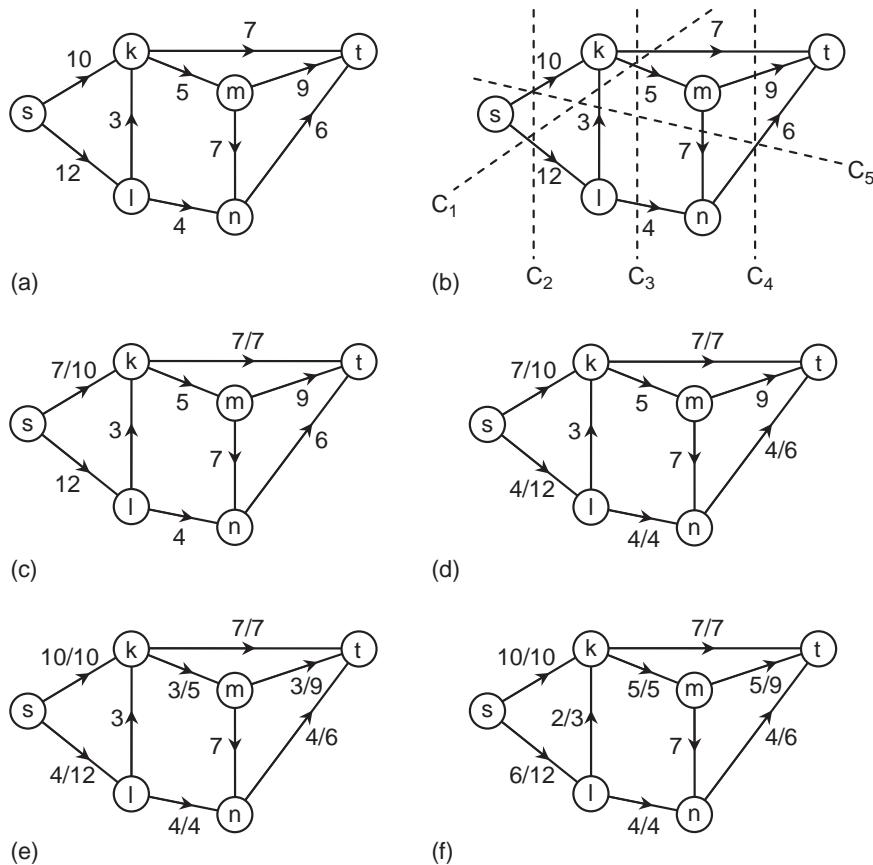
It has been stated several times in both Chapter 4 and this chapter that image segmentation tends to be an unreliable *ad hoc* process if simple uniformity measures are used for locating the extents of regions. For example, region growing techniques are prone to problems such as leaking through weak points in object boundaries, whereas thresholding techniques are sensitive to problems of variable illumination, so again one region will elide into another. Edge-based methods can also become confused at breaks, and edge linking (using hysteresis thresholding or more sophisticated techniques) cannot be relied upon to improve the situation dramatically. The reason for the widespread use of such techniques is their simplicity and speed. However, modern computers are nowadays capable of handling much more powerful segmentation algorithms, and it is key that such algorithms should be robust, effective, and accurate even if they involve increased computation. Hence, there is a trend for segmentation and other vision algorithms to be designed to optimize carefully selected energy functions. We have already seen this with the active contour formalism covered in Section 5.13. The snake algorithm employed in Section 5.14 is “greedy,” so it involves many local processes and therefore is not guaranteed to find a single global optimum: however, to a lesser extent this also applies with more rigorously conceived active contour

models. Neural network techniques (whether used for segmentation, recognition, or other purposes) normally operate by minimizing energy functions, but again have a tendency to be trapped by local energy minima.

In this context, the graph cut approach to segmentation has become an especially attractive one because its aim is to guarantee exact convergence to the minimum of a global energy function. This is because, in a carefully chosen mathematical milieu, it can be proven that only a single global solution exists. Of course, it is also necessary to relate the particular mathematical milieu to the type of reality arising for practical segmentation tasks.

To achieve this, we first describe an ideal traffic flow problem where every road has an exact, known maximum flow capacity. (It will be most people's experience that any road has a fairly well-defined maximum flow capacity.) Taking a network of roads joining a source s to a sink t , we aim to determine the maximum flow rate between these two terminals. It turns out that there is a theorem (Ford and Fulkerson, 1962) that gives a unique answer to this. We first define a "cut," which is a line passing across the roads between s and t and partitioning the junctions between them into two sets S and T . We then work out the capacity of the cut, this being defined as the sum of the (s to t) capacities of the roads intersected by the cut (Fig. 5.11(a) and (b)). Next, we assess which of all possible cuts has the minimum capacity c : this is called the "minimum cut" C . Interestingly, c is equal to the maximum flow capacity f allowed by the network between s and t . To understand why this is so, suppose first that $f > c$. Then there must be another route between s and t , which is able to carry additional flow. But by definition, any cut has to partition *all* the junctions into two sets S and T , so it has to cross *all possible routes* between s and t . Hence, the premise is violated and the maximum network capacity f cannot be greater than the minimum cut capacity c . Suppose now that $f < c$, then there is no reason why the flow through at least one road cut by C —in particular one that is not saturated—cannot be augmented until the condition no longer holds. (Note that increasing the flow will violate the condition for C first as, by definition, C is the minimum cut.) Hence, by the end of this process, f will be equal to c . Thus, we finally conclude that max flow = min cut. Fig. 5.11(c) and (d) shows how the maximum flow rates can be worked out intuitively by identifying a suitable sequence of paths: the process can be carried out more systematically using the augmenting paths approach outlined below.

Since the max flow–min cut theorem is guaranteed to give a unique global solution, it is worth trying to map it onto the object segmentation task. To achieve this we take all neighbor links (n-links) between pixels (we can envisage these as being nearest neighbor links between pixels, which is highly practical but not essential) and see whether we can identify the minimum cut with the boundary of an object. First, it is clearly necessary to place s inside the object and t outside it, in each case with suitable terminal links (t-links) between the terminal (s or t) and a number of pixels inside and outside the object. It also has to be arranged for the links between pixels to be weighted in such a way as to regulate the flow pattern. Denoting the i th pixel intensity by I_i , the flow capacity can be minimized at edges

**FIGURE 5.11**

Achieving maximum flow in a network. (a) A network of roads with a source s , a sink t , and a set of junctions k , l , m , and n . Each road is marked with a number representing its maximum flow capacity. (b) Five cuts C_1 – C_5 are defined, each of which intersects a set of roads and partitions the junctions into two sets S and T . The respective cuts have capacities 24, 22, 16, 22, and 19, making C_3 the minimum cut, with capacity $c = 16$. (c) A flow of 7 is initiated via path s – k – t : this is the maximum possible as the capacity of road kt is 7 (the notation “ u/v ” next to a road means that the actual flow is u and the capacity is v). (d) It is easy to augment the total flow by initiating a maximal flow of 4 in path s – l – n – t . (e) Further thought shows that dual use of road sk is permissible, allowing an additional flow of 3 in path s – k – m – t . (f) A further, longer augmenting path s – l – k – m – t is possible, involving dual use of roads sl , km , and mt . This increases the net flow through the network to 16, as predicted. The actual flow through each of the five cuts shown in (b) is seen to be 16: in case C_1 the result is not 18 because we are now considering actual flows rather than capacities (conservation of flow applies to actual flows, but not to capacities). Note that road mn is not useful for optimizing flow.

where $|I_i - I_j|$ is large and maximized where $I_i \approx I_j$. For this purpose a convenient energy function (Boykov and Jolly, 2001) is:

$$E_{ij} = \frac{1}{d_{ij}} \exp\left[\frac{-(I_i - I_j)^2}{2\sigma^2}\right] \quad (5.39)$$

where d_{ij} is the distance between pixels i and j , and is introduced to give greater relevance for pairs of pixels that are closer together. Note that E_{ij} tends to zero for high-intensity gradients, but is low for pairs of pixels with similar intensities. In fact, one of the main requirements is to prevent noise from influencing the locations of object boundaries, so σ^2 can be taken to correspond to the local Gaussian noise energy.

Overall, the idea is to penalize the formation of boundaries in regions of uniformity and to encourage it at locations of high-intensity gradient. Note that the mapping between the max flow–min cut milieu and the optimization of object boundaries is slightly arbitrary, and corresponds to an *analogy*—albeit with a sensible choice of intensity mapping function, a single global solution is still guaranteed to exist (this is mediated more by the algorithm than by the data). In fact, there is another problem as well: that finding a fast algorithm to determine the min cut is nontrivial, so it is possible that approximations might have to be made that are incompatible with identifying an exact global minimum.

The classic Ford and Fulkerson (1962) algorithm worked by starting with a single complete path P from s to t , and determining the n-link (p, q) with the minimum capacity, and then incrementing the flow along path P until link (p, q) was saturated and became the bottleneck for the whole path P . Then “augmenting” paths from s to t were sought in turn, each time incrementing the flow until another bottleneck link became saturated. At each stage the algorithm is most conveniently presented in the form of a *residual network* R consisting of all the unsaturated links in the network (this is essentially a subset of the original network N) and an augmenting path is a path within R . Clearly, one link in the residual network R becomes saturated and is lost to R in each successful iteration: eventually R will become disconnected (no paths from s to t will exist in R) and the total flow from s to t will be a maximum. (Here we ignore the mathematical complication that R will have acquired oppositely orientated (reverse) flows in many of its links: these arise because the residual network is a representation in which the flows in the saturated links are canceled by oppositely directed flows.)

The Ford and Fulkerson (1962) algorithm was long known to be quite slow, and not to be guaranteed to run in polynomial time, except for integer flows. However, two routes for speeding it up came to light later on—first the Dinic (1970) algorithm and second the push-relabel algorithm (Goldberg and Tarjan, 1988). The former uses an augmentation strategy similar to that of the Ford and Fulkerson algorithm, but uses breadth-first search to tackle short paths first. The latter aims to push flows as far as possible from s toward t , taking account of the fact that various links will become saturated. The two approaches give similar flow rates, so here we concentrate on the former.

In fact, the shortest path strategy of the Dinic algorithm is important in permitting it to achieve a worst-case running time complexity of $O(mn^2)$, where n is the number of junctions and m is the number of links in the network. Boykov and Jolly (2001), and others (Boykov and Kolmogorov, 2004; Boykov and Funka-Lea, 2006) developed an even faster algorithm based on the augmented paths approach. Although it has essentially the same worst-case complexity as the Dinic algorithm, it is found to be far faster in practice (at least when applied in typical vision applications). This is because the search trees it uses in each iteration do not have to be developed from scratch but on the contrary capitalize on their previous forms—albeit incorporating novelties such as pruning “orphan” nodes, which turn out to be efficient in practice. This means that graph cut algorithms have now achieved a high degree of efficiency for practical energy minimization tasks. Nevertheless, they have not yet attained absolute supremacy for image segmentation—which may be due partly to the fact that seeds have to be selected to act as terminal nodes, thereby making the approach best suited for interactive use. (Interactive use is not necessarily overly disadvantageous, e.g., when analyzing medical data such as that from MRI brain scans.)

5.17 CONCLUDING REMARKS

The above sections make it clear that the design of edge detection operators has by now been taken to quite an advanced stage, so that edges can be located to subpixel accuracy and orientated to fractions of a degree. In addition, edge maps may be made at several scales and the results correlated to aid image interpretation. Unfortunately, some of the schemes that have been devised to achieve these things (and especially that outlined in the previous section) are fairly complex and tend to consume considerable computation. In many applications this complexity may not be justified because the application requirements are, or can reasonably be made, quite restricted. Furthermore, there is often the need to save computation for real-time implementation. For these reasons, it will often be useful to explore what can be achieved using a single high-resolution detector such as the Sobel operator, which provides a good balance between computational load and orientation accuracy. Indeed, several of the examples in Part 2 of the book have been implemented using this type of operator, which is able to estimate edge orientation to within about 1° . This does not in any way invalidate the latest methods, particularly those involving studies of edges at various scales: such methods come into their own in applications such as general scene analysis, where vision systems are required to cope with largely unconstrained image data.

This chapter has completed another facet of the task of low-level image segmentation. Later chapters move on to consider the shapes of objects that have been found by the thresholding and edge detection schemes discussed in the last two chapters. In particular, Chapter 9 studies shapes by analysis of the regions over which objects extend, whereas Chapter 10 studies shapes by considering their boundary patterns.

Edge detection is perhaps the most widely used means of locating and identifying objects in digital images. Although different edge detection strategies vie with each other for acceptance, this chapter has shown that they obey fundamental laws, such as sensitivity, noise suppression capability, and computation cost all increasing with footprint size.

5.18 BIBLIOGRAPHICAL AND HISTORICAL NOTES

As seen in the first few sections of this chapter, early attempts at edge detection tended to employ numbers of template masks that could locate edges at various orientations. Often these masks were *ad hoc* in nature, and after 1980 this approach finally gave way to the differential gradient approach that had already existed in various forms for a considerable period (see the influential paper by Haralick, 1980).

The Frei–Chen approach is of interest in that it takes a set of nine 3×3 masks forming a complete set within this size of neighborhood—of which one test for brightness, four test for edges, and four test for lines (Frei and Chen, 1977). Although interesting, the Frei–Chen edge masks do not correspond to those devised for optimal edge detection: Lacroix (1988) makes further useful remarks about the approach.

Meanwhile, psychophysical work by Marr (1976), Wilson and Giese (1977), and others provided another line of development for edge detection. This led to the well-known paper by Marr and Hildreth (1980), which was highly influential in the following few years. This spurred others to think of alternative schemes, and the Canny (1986) operator emerged from this vigorous milieu. In fact, the Marr–Hildreth operator was among the first to preprocess images in order to study them at different scales—a technique that has expanded considerably (see, e.g., Yuille and Poggio, 1986) and which will be considered in more depth in Chapter 6. The computational problems of the Marr–Hildreth operator kept others thinking along more traditional lines, and the work by Reeves et al. (1983), Haralick (1984), and Zuniga and Haralick (1987) fell into this category. Lyvers and Mitchell (1988) reviewed many of these papers and made their own suggestions. Another study (Petrou and Kittler, 1988) carried out further work on operator optimization. The work of Sjöberg and Bergholm (1988), which found rules for discerning shadow edges from object edges, is also of interest.

More recently, there was a move to achieving greater robustness and confidence in edge detection by careful elimination of local outliers: in Meer and Georgescu's (2001) method, this was achieved by estimating the gradient vector, suppressing nonmaxima, performing hysteresis thresholding, and integrating with a confidence measure to produce a more general robust result; in fact, each pixel was assigned a confidence value *before* the final two steps of the algorithm. Kim et al. (2004) took this technique a step further and eliminated the need for setting a threshold by using a fuzzy reasoning approach. Similar sentiments were

expressed by Yitzhaky and Peli (2003), and they aimed to find an optimal parameter set for edge detectors by ROC and chi-square measures, which actually gave very similar results. Prieto and Allen (2003) designed a similarity metric for edge images, which could be used to test the effectiveness of a variety of edge detectors. They pointed to the fact that metrics need to allow slight latitude in the positions of edges, in order to compare the similarity of edges reliably. They reported a new approach that took into account both displacement of edge positions and edge strengths in determining the similarity between edge images.

Not content with hand-crafted algorithms, Suzuki et al. (2003) devised a back-propagation neural edge enhancer, which undergoes supervised learning on model data to permit it to cope well (in the sense of giving clear, continuous edges) with noisy images: it was found to give results superior to those of conventional algorithms (including Canny, Heuckel, Sobel, and Marr–Hildreth) in similarity tests relative to the desired edges. The disadvantage was a long learning time, although the final execution time was short.

5.18.1 More Recent Developments

Among the most recent developments, Shima et al. (2010) have described the design of more accurate gradient operators on hexagonal lattices. Although the latter are not commonly used, there has long been a special interest in this area because of the greater number of nearest neighbors at equal distances from a given pixel in a hexagonal lattice: this makes certain types of window operation and algorithm more accurate and efficient, and is particularly useful for edge detection and thinning. Ren et al. (2010) have described an improved edge detection algorithm that operates via the fusion of intensity and chromatic difference, thereby making better use of inter-component information in color images.

Cosío et al. (2010) used simplex search in active shape models for improved boundary segmentation: this involves fast numerical optimization to find the most suitable values of nonlinear functions without the need to calculate function derivatives. Their approach typically employs 4 pose parameters and 10 shape parameters for defining a shape such as the prostate. The method significantly increases the range of object poses, and thus results in more accurate boundary segmentation. Chiverton et al. (2008) describe a method that is closely related to the active contour concept: it zones in on objects using parameters relating to foreground similarity and background dissimilarity, and employs a new variational logistic maximum a posteriori (MAP) contextual modeling schema. In this case the (achieved) aim is to permit tracking of moving objects by iterative adaptive matching. Mishra et al. (2011) identify five basic limitations of preexisting active contour methods. Their solution is to decouple the internal and external active contour energies and to update for each of them separately. The method is shown to be faster and to have at least comparable segmentation accuracy to five earlier methods.

Papadakis and Bugeau (2011) underline the power of the graph cut approach (see [Section 5.16](#)) by showing that it can be applied to tracking partially occluded objects if predictions to allow for this are included in the formalism. They make the key point that the advantages of the graph cut approach are “its low computational cost and the fact that it converges to the global minimum without getting stuck in local minima.”

5.19 PROBLEMS

1. Prove Eqs. [\(5.12\)](#) and [\(5.13\)](#).
2. Check the results quoted in [Section 5.9](#) giving the conditions under which the Sobel operator leads to zero error in the estimation of edge orientation. Proceed to prove Eq. [\(5.17\)](#).

Corner and Interest Point Detection

6

Corner detection is valuable for locating complex objects and for tracking them in 2-D or 3-D. This chapter discusses this detection problem and considers methods that are best suited for the task.

Look out for:

- the ways in which corner features are useful.
- the variety of methods available for corner detection—template matching, the second-order derivative method, the median-based method, the Harris interest point detector.
- where the corner signal is a maximum: how detector bias arises.
- how corner orientation may be estimated.
- why invariant feature detectors are needed: the hierarchy of relevant types of invariance.
- how feature detectors may be made invariant to similarity and affine transformations.
- the need for invariant detectors to embody multiparameter descriptors to help with subsequent matching tasks.
- what criteria can be developed for measuring the performance of conventional and invariant types of feature and feature detector.

Note the variety of methods available for performing interrelated detection tasks. However, different methods have different speeds, accuracies, sensitivities, and degrees of robustness: this chapter aims to bring out all these aspects of the problem.

6.1 INTRODUCTION

This chapter is concerned with the efficient detection of corners. It has been noted in previous chapters that objects are generally located most efficiently from their features. Prominent features include straight lines, circles, arcs, holes, and corners. Corners are particularly important since they may be used to locate and orientate objects and to provide measures of their dimensions; for example, knowledge about orientation will be vital if a robot is to find the best way of picking up an object, while dimensional measurement will be necessary in most inspection applications. Hence efficient, accurate corner detectors are of great relevance in machine vision.

We start this chapter by considering what is perhaps the most obvious detection scheme—that of template matching. Then we move on to other types of detectors, based on the second-order derivatives of the local intensity function; subsequently, we find that median filters can lead to useful corner detectors, with properties similar to those for the second-order derivative based detectors. Next, we consider detectors based on the second moments of the *first* derivatives of the local intensity function. While this will complete the traditional approach to corner detection, it opens the door for consideration of the highly important invariant local feature detectors that have in the past decade or so been developed for matching widely separated views of 3-D scenes, including those containing rapidly moving objects. By the end of the chapter the vital task of considering performance criteria for the various types of corner and feature detector will also be undertaken.

6.2 TEMPLATE MATCHING

Following our experience with template matching methods for edge detection (Chapter 5), it would appear to be straightforward to devise suitable templates for corner detection. These would have the general appearance of corners, and in a 3×3 neighborhood would take forms such as the following:

$$\begin{bmatrix} -4 & 5 & 5 \\ -4 & 5 & 5 \\ -4 & -4 & -4 \end{bmatrix} \quad \begin{bmatrix} 5 & 5 & 5 \\ -4 & 5 & -4 \\ -4 & -4 & -4 \end{bmatrix}$$

The complete set of eight templates would be generated by successive 90° rotations of the first two shown. An alternative set of templates was suggested by Bretschneider (1981). As for edge detection templates, the mask coefficients are made to sum to zero, so that corner detection is insensitive to absolute changes in light intensity. Ideally, this set of templates should be able to locate all corners and to estimate their orientation to within 22.5° .

Unfortunately, corners vary very much in a number of their characteristics, including in particular their degree of pointedness,¹ internal angle and the intensity gradient at the boundary. Hence it is quite difficult to design optimal corner detectors. In addition, corners are generally insufficiently pointed for good results to be obtained with the 3×3 template masks shown above. Another problem is that in larger neighborhoods, not only do the masks become larger but also more of them are needed to obtain optimal corner responses, and it rapidly becomes clear that the template matching approach is likely to involve excessive computation for practical corner detection. The alternative is to approach the problem analytically, somehow deducing the ideal response for a corner at any arbitrary orientation, and thereby bypassing the problem of calculating many individual responses to find the one that gives maximum signal. The methods described in the remainder of this chapter embody this alternative philosophy.

6.3 SECOND-ORDER DERIVATIVE SCHEMES

Second-order differential operator approaches have been used widely for corner detection and to mimic the first-order operators used for edge detection. Indeed, the relationship lies deeper than this. By definition, corners in grayscale images occur in regions of rapidly changing intensity levels. By this token they are detected by the same operators that detect edges in images. However, corner pixels are much rarer² than edge pixels—by one definition, they arise where two relatively straight-edged fragments intersect. Thus, it is useful to have operators that detect corners *directly*, i.e. *without unnecessarily locating edges*. To achieve this sort of discriminability it is clearly necessary to consider local variations in image intensity up to at least second order. Hence, the local intensity variation is expanded as follows:

$$I(x, y) = I(0, 0) + I_x x + I_y y + I_{xx} \frac{x^2}{2} + I_{xy} xy + I_{yy} \frac{y^2}{2} + \dots \quad (6.1)$$

where the suffices indicate partial differentiation with respect to x and y and the expansion is performed about the origin $X_0 (0,0)$. The symmetrical matrix of second derivatives is:

$$\mathbf{I}_{(2)} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix}, \quad \text{where } I_{xy} = I_{yx} \quad (6.2)$$

¹The term “pointedness” is used as the opposite to “bluntness,” the term “sharpness” being reserved for the total angle η through which the boundary turns in the corner region, i.e. π minus the internal angle.

²We might imagine a 256×256 image of 64K pixels, of which 1000 ($\sim 2\%$) lie on edges and a mere 30 ($\sim 0.06\%$) are situated at corner points.

This gives information on the local curvature at X_0 . In fact, a suitable rotation of the coordinate system transforms $\mathbf{I}_{(2)}$ into diagonal form:

$$\tilde{\mathbf{I}}_{(2)} = \begin{bmatrix} I_{\tilde{x}\tilde{x}} & 0 \\ 0 & I_{\tilde{y}\tilde{y}} \end{bmatrix} = \begin{bmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{bmatrix} \quad (6.3)$$

where appropriate derivatives have been reinterpreted as principal curvatures at X_0 .

We are particularly interested in rotationally invariant operators and it is significant that the trace and determinant of a matrix such as $\mathbf{I}_{(2)}$ are invariant under rotation. Thus, we obtain the Beaudet (1978) operators:

$$\text{Laplacian} = I_{xx} + I_{yy} = \kappa_1 + \kappa_2 \quad (6.4)$$

and

$$\text{Hessian} = \det(\mathbf{I}_{(2)}) = I_{xx}I_{yy} - I_{xy}^2 = \kappa_1\kappa_2 \quad (6.5)$$

It is well known that the Laplacian operator gives significant responses along lines and edges and hence is not particularly suitable as a corner detector. On the other hand, Beaudet's "DET" operator does not respond to lines and edges but gives significant signals in the vicinity of corners: it should therefore form a useful corner detector. However, DET responds with one sign on one side of a corner and with the opposite sign on the other side of the corner: at the point of real interest—on the point of the corner—it gives a null response. Hence, rather more complicated analysis is required to deduce the presence and exact position of each corner (Dreschler and Nagel, 1981; Nagel, 1983). The problem is clarified in Fig. 6.1. Here the dotted line shows the path of maximum horizontal curvature



FIGURE 6.1

Sketch of an idealized corner, taken to give a smoothly varying intensity function. The dotted line shows the path of maximum horizontal curvature for various intensity values up the slope. The DET operator gives maximum responses at P and Q, and it is required to find the ideal corner position C where DET gives a null response.

for various intensity values up the slope. The DET operator gives maximum response at positions P and Q on this line, and the parts of the line between P and Q must be explored to find the “ideal” corner point C where DET is zero.

Perhaps to avoid rather complicated procedures of this sort, Kitchen and Rosenfeld (1982) examined a variety of strategies for locating corners, starting from the consideration of local variation in the directions of edges. They found a highly effective operator that estimates the projection of the local rate of change of gradient direction vector along the horizontal edge tangent direction, and showed that it is mathematically identical to calculating the horizontal curvature κ of the intensity function I . To obtain a realistic indication of the strength of a corner they multiplied κ by the magnitude of the local intensity gradient g :

$$\begin{aligned} C &= \kappa g = \kappa(I_x^2 + I_y^2)^{1/2} \\ &= \frac{I_{xx}I_y^2 - 2I_{xy}I_xI_y + I_{yy}I_x^2}{I_x^2 + I_y^2} \end{aligned} \quad (6.6)$$

Finally, they used the heuristic of nonmaximum suppression along the edge normal direction to localize the corner positions further.

In 1983, Nagel was able to show that the Kitchen and Rosenfeld (KR) corner detector using nonmaximum suppression is mathematically virtually identical to the Dreschler and Nagel (DN) corner detector. A year later, Shah and Jain (1984) studied the Zuniga and Haralick (ZH) corner detector (1983) based on a bicubic polynomial model of the intensity function: they showed that this is essentially equivalent to the KR corner detector. However, the ZH corner detector operates rather differently in that it thresholds the intensity gradient and then works with the subset of edge points in the image, only at that stage applying the curvature function as a corner strength criterion. By making edge detection explicit in the operator, the ZH detector eliminates a number of false corners that would otherwise be induced by noise.

The inherent near-equivalence of these three corner detectors need not be overly surprising, since in the end the different methods would be expected to reflect the same underlying physical phenomena (Davies, 1988d). However, it is gratifying that the ultimate result of these rather mathematical formulations is interpretable by something as easy to visualize as horizontal curvature multiplied by intensity gradient.

6.4 A MEDIAN FILTER-BASED CORNER DETECTOR

An entirely different strategy for detecting corners was developed by Paler et al. (1984). It adopts an initially surprising and rather nonmathematical approach based on the properties of the median filter. The technique involves applying a median filter to the input image, and then forming another image that is the difference between the input and the filtered images. This difference image contains a set of signals that are interpreted as local measures of corner strength.

Clearly, it seems risky to apply such a technique since its origin suggests that, far from giving a correct indication of corners, it may instead unearth all the noise in the original image and present this as a set of “corner” signals. Fortunately, analysis shows that these worries may not be too serious. First, in the absence of noise, strong signals are not expected in areas of background; nor are they expected near straight edges, since median filters do not shift or modify such edges significantly (see Chapter 3). However, if a window is moved gradually from a background region until its central pixel is just over a convex object corner, there is no change in the output of the median filter: hence, there is a strong difference signal indicating a corner.

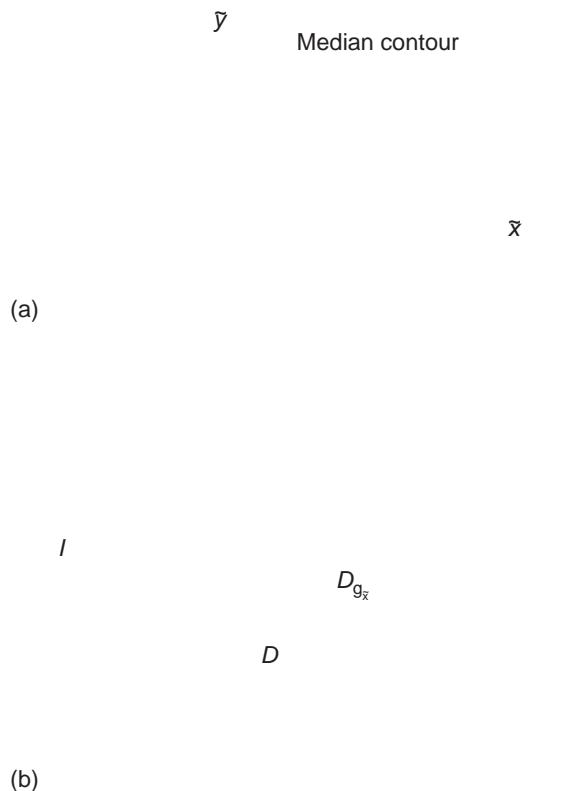
Paler et al. (1984) analyzed the operator in some depth and concluded that the signal strength obtained from it is proportional to (a) the local contrast, and (b) the “sharpness” of the corner. The definition of sharpness they used was that of Wang et al. (1983), meaning the angle η through which the boundary turns. Since it is assumed here that the boundary turns through a significant angle (perhaps the whole angle η) within the filter neighborhood, the difference from the second-order intensity variation approach is a major one. Indeed, it is an implicit assumption in the latter approach that first- and second-order coefficients describe the local intensity characteristics reasonably rigorously, the intensity function being inherently continuous and differentiable. Thus, the second-order methods may give unpredictable results with pointed corners where directions change within the range of a few pixels. Although there is some truth in this, it is worth looking at the similarities between the two approaches to corner detection before considering the differences. We proceed with this in the next subsection.

6.4.1 Analyzing the Operation of the Median Detector

This subsection considers the performance of the median corner detector under conditions where the grayscale intensity varies by only a small amount within the median filter neighborhood region. This permits the performance of the corner detector to be related to low-order derivatives of the intensity variation, so that comparisons can be made with the second-order corner detectors mentioned earlier.

To proceed we assume a continuous analog image and a median filter operating in an idealized circular neighborhood. For simplicity, since we are attempting to relate signal strengths and differential coefficients, noise is ignored. Next, recall (Chapter 3) that for an intensity function that increases monotonically with distance in some arbitrary direction \tilde{x} but which does not vary in the perpendicular direction \tilde{y} , the median within the circular window is equal to the value at the center of the neighborhood. This means that the median corner detector gives zero signal if the horizontal curvature is locally zero.

If there is a small horizontal curvature κ , the situation can be modeled by envisaging a set of constant-intensity contours of roughly circular shape and approximately equal curvature, within a circular window of radius a (Fig. 6.2). Consider the contour having the median intensity value. The center of this

**FIGURE 6.2**

(a) Contours of constant intensity within a small neighborhood: ideally, these are parallel, circular and of approximately equal curvature (the contour of median intensity does not pass through the center of the neighborhood); (b) cross-section of intensity variation, indicating how the displacement D of the median contour leads to an estimate of corner strength.

Source: © Springer 1988

contour does not pass through the center of the window but is displaced to one side along the negative \tilde{x} -axis. Furthermore, the signal obtained from the corner detector depends on this displacement. If the displacement is D , it is easy to see that the corner signal is $Dg_{\tilde{x}}$ because $g_{\tilde{x}}$ allows the intensity change over the distance D to be estimated (Fig. 6.2). The remaining problem is to relate D to the horizontal curvature κ . A formula giving this relation has already been obtained in Chapter 3. The required result is:

$$D = \frac{1}{6} \kappa a^2 \quad (6.7)$$

so the corner signal is

$$C = Dg_{\tilde{x}} = \frac{1}{6} \kappa g_{\tilde{x}} a^2 \quad (6.8)$$

Note that C has the dimensions of intensity (contrast), and that the equation may be re-expressed in the form:

$$C = \frac{1}{12} (g_{\tilde{x}} a) \cdot (2a\kappa) \quad (6.9)$$

so that, as in the formulation of Paler et al. (1984), corner strength is closely related to corner contrast and corner sharpness.

To summarize, the signal from the median-based corner detector is proportional to horizontal curvature and to intensity gradient. Thus, this corner detector gives an identical response to the three second-order intensity variation detectors discussed in [Section 6.3](#), the closest practically being the KR detector. However, this comparison is valid only when second-order variations in intensity give a complete description of the situation. Clearly, the situation might be significantly different where corners are so pointed that they turn through a large proportion of their total angle within the median neighborhood. In addition, the effects of noise might be expected to be rather different in the two cases, as the median filter is particularly good at suppressing impulse noise. Meanwhile, for small horizontal curvatures, there ought to be no difference in the positions at which median and second-order derivative methods locate corners, and accuracy of localization should be identical in the two cases.

6.4.2 Practical Results

Experimental tests with the median approach to corner detection have shown that it is a highly effective procedure (Paler et al., 1984; Davies, 1988d). Corners are detected reliably and signal strength is indeed roughly proportional both to local image contrast and to corner sharpness (see [Fig. 6.3](#)). Noise is more apparent for 3×3 implementations and this makes it better to use 5×5 or larger neighborhoods to give good corner discrimination. However, the fact that median operations are slow in large neighborhoods, and that background noise is still evident even in 5×5 neighborhoods, means that the basic median-based approach gives poor performance by comparison with the second-order methods. However, both of these disadvantages are virtually eliminated by using a “skimming” procedure, in which edge points are first located by thresholding the edge gradient, and the edge points are then examined with the median detector to locate the corner points (Davies, 1988d). With this improved method, performance is found to be generally superior to that for the KR method in that corner signals are better localized and accuracy is enhanced. Indeed, the second-order methods appear to give rather fuzzy and blurred signals that contrast with the sharp signals obtained with the improved median approach ([Fig. 6.4](#)).

At this stage the reason for the more blurred corner signals obtained using the second-order operators is not clear. Basically, there is no valid rationale for

(a)

(b)

FIGURE 6.3

(a) Original off-camera 128×128 6-bit grayscale image; (b) result of applying the median-based corner detector in a 5×5 neighborhood. Note that corner signal strength is roughly proportional both to corner contrast and to corner sharpness.

(a)

(b)

(c)

(d)

FIGURE 6.4

Comparison of the median and KR corner detectors: (a) original 128×128 grayscale image; (b) result of applying a median detector; (c) result of including a suitable gradient threshold; (d) result of applying a KR detector. The considerable amount of background noise is saturated out in (a) but is evident from (b). To give a fair comparison between the median and KR detectors, 5×5 neighborhoods are employed in each case, and nonmaximum suppression operations are not applied: the same gradient threshold is used in (c) and (d).

Source: © Springer 1988

applying second-order operators to pointed corners, since higher derivatives of the intensity function will become important and will at least in principle interfere with their operation. However, it is evident that the second-order methods will probably give strong corner signals when the tip of a pointed corner appears anywhere in their neighborhood, so there is likely to be a minimum blur region of radius a for any corner signal. This appears to explain the observed results adequately. However, note that the sharpness of signals obtained by the KR method may be improved by nonmaximum suppression (Kitchen and Rosenfeld, 1982; Nagel, 1983). Furthermore, this technique can also be applied to the output of median-based corner detectors: hence, the fact remains that the median-based method gives inherently better localized signals than the second-order methods.

Overall, the inherent deficiencies of the median-based corner detector can be overcome by incorporating a skimming procedure, and then the method becomes superior to the second-order approaches in giving better localization of corner signals. The underlying reason for the difference in localization properties appears to be that the median-based signal is ultimately sensitive only to the particular few pixels whose intensities fall near the median contour within the window, whereas the second-order operators use typical convolution masks that are in general sensitive to the intensity values of all the pixels within the window. Thus, the KR operator tends to give a strong signal when the tip of a pointed corner is present anywhere in the window.

6.5 THE HARRIS INTEREST POINT OPERATOR

Earlier in this chapter, we considered the second-order derivative type of corner detector that was designed on the basis that corners are ideal, smoothly varying differentiable intensity profiles. We also examined median filter-based detectors: these had a totally different modus operandi and were found to be suitable for processing curved step edges whose profiles were quite likely *not* to be smoothly varying and differentiable. At this point we consider what other strategies are available for corner detection. An important one that has become extremely widely used is the Harris operator. Far from being a second-order derivative type of detector, the Harris operator only takes account of *first*-order derivatives of the intensity function. Thus, there is a question of how it can acquire enough information to detect corners. In this section, we construct a model of its operation in order to throw light on this crucial question.

The Harris operator is defined very simply, in terms of the local components of intensity gradient I_x, I_y in an image. The definition requires a window region to be defined and averages $\langle \cdot \rangle$ are taken over this whole window. We start by computing the following matrix:

$$\Delta = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \quad (6.10)$$

where the suffixes indicate partial differentiation of the intensity I ; we then use the determinant and trace to estimate the corner signal:

$$C = \frac{\det\Delta}{\text{trace}\Delta} \quad (6.11)$$

While this definition involves averages, we shall find it more convenient to work with sums of quadratic products of intensity gradients:

$$\Delta = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \quad (6.12)$$

To understand the operation of the detector, first consider its response for a single edge (Fig. 6.5(a)). In fact:

$$\det\Delta = 0 \quad (6.13)$$

because I_x is zero over the whole window region. Note that there is no loss in generality from selecting a horizontal edge, as $\det\Delta$ and $\text{trace}\Delta$ are invariant under rotation of axes.

Next consider the situation in a corner region (Fig. 6.5(b)). Here:

$$\Delta = \begin{bmatrix} l_2 g^2 \sin^2 \theta & l_2 g^2 \sin \theta \cos \theta \\ l_2 g^2 \sin \theta \cos \theta & l_2 g^2 \cos^2 \theta + l_1 g^2 \end{bmatrix} \quad (6.14)$$

where l_1 , l_2 are the lengths of the two edges bounding the corner, and g is the edge contrast, assumed constant over the whole window. We now find:

$$\det\Delta = l_1 l_2 g^4 \sin^2 \theta \quad (6.15)$$

and

$$\text{trace}\Delta = (l_1 + l_2)g^2 \quad (6.16)$$

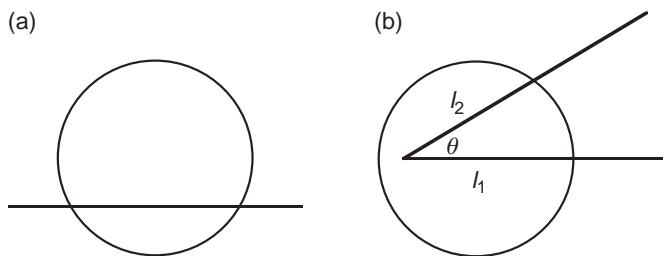


FIGURE 6.5

Case of a straight edge and a general corner. (a) A single straight edge appearing in a circular window. (b) A general corner appearing in a circular window. Circular windows are taken as ideal, in that they will not favor any direction over any other.

so

$$C = \frac{l_1 l_2}{l_1 + l_2} g^2 \sin^2 \theta \quad (6.17)$$

which may be interpreted as the product of (1) a strength factor λ , which depends on the edge lengths within the window, (2) a contrast factor g^2 , and (3) a shape factor $\sin^2 \theta$, which depends on the edge “sharpness” θ . Clearly, C is zero for $\theta = 0$ and $\theta = \pi$, and is a maximum for $\theta = \pi/2$, all these results being intuitively correct and appropriate.

There is a useful theorem about the sets of lengths l_1, l_2 for which the strength factor λ , and thus C , is a maximum. Suppose we set $L = l_1 + l_2 = \text{constant}$. Then $l_1 = L - l_2$, and substituting for l_1 we find:

$$\lambda = \frac{l_1 l_2}{l_1 + l_2} = \left[\frac{Ll_2 - l_2^2}{L} \right] \quad (6.18)$$

$$\therefore \frac{d\lambda}{dl_2} = \frac{1 - 2l_2}{L} \quad (6.19)$$

which is zero for $l_2 = L/2$, at which point $l_1 = l_2$. This means that the best way of obtaining maximum corner signal is to place the corner symmetrically within the window, following which the signal can be increased further by moving the corner so that L is maximized (Fig. 6.6).

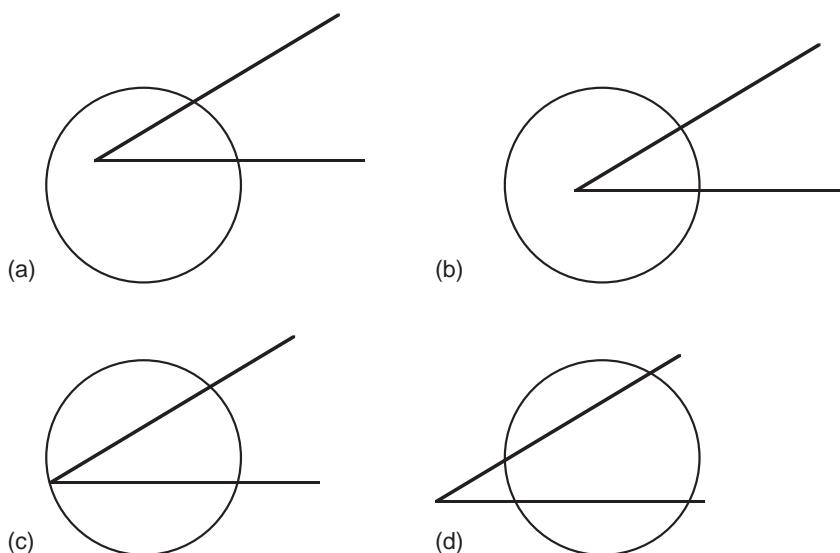


FIGURE 6.6

Possible geometries for a sharp corner being sampled by a circular window. (a) General case. (b) Symmetrical placement with $l_1 = l_2$ (see notation in Fig. 6.5(b)). (c) Case of maximum signal. (d) Case where the signal is reduced in size as the tip of the corner goes outside the window.

Source: © IET 2005

Next, if l_i is small (for either value of i), the corner signal at first increases linearly with l_i , and as noted earlier, the corner detector will ignore a single straight edge on its own.

Finally, the fact that we are exploring the properties of a symmetric matrix, which can be represented using any convenient set of orthogonal axes, means that we can find the eigenvalues and eigenvectors. However, it is illuminating to note that these arise automatically when a symmetrically aligned set of axes is selected along the corner bisectors, as then the off-diagonal elements of the modified Δ matrix acquire two components $(L/2)g^2 \sin(\theta/2) \cos(\theta/2)$ of opposite sign and therefore cancel out. The on-diagonal elements are thus the eigenvalues themselves, and are $(L/2)g^2 \times 2 \cos^2(\theta/2)$, $(L/2)g^2 \times 2 \sin^2(\theta/2)$. Again, if $\theta = 0$ or π , one or the other eigenvalue is zero, so the determinant is zero and the corner signal vanishes; also, the maximum signal occurs for $\theta = \pi/2$.

6.5.1 Corner Signals and Shifts for Various Geometric Configurations

In this section, we seek the conditions for maximum corner signal for corners of different degrees of sharpness. We shall make use of the observation made in the previous section that maximum signal requires that $l_1 = l_2 = L/2$.

First, we take the case when $\theta = 0$: we have already seen that this leads to $C = 0$.

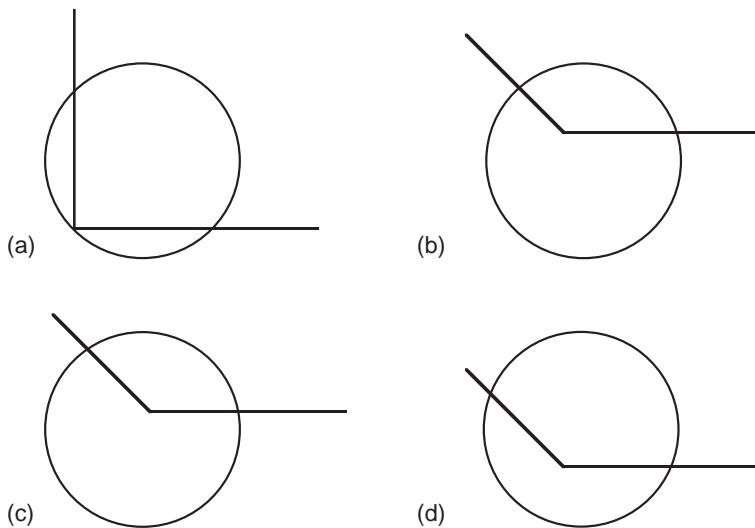
Next, when θ is small, i.e. less than $\pi/2$, we can go on increasing L by moving the corner symmetrically. The optimum is reached exactly as the tip of the corner reaches the far side of the window (Fig. 6.6). We could envisage the corner moving even further, but then the portions of the sides that lie within the window will be moved laterally, so they will become shorter, and the signal will fall (Fig. 6.6(d)).

Now take the case $\theta = \pi/2$. Then we can again proceed as above, and the optimum will still occur when the tip of the corner lies on the far side of the window (Fig. 6.7(a)). However, further increase of θ will result in a different optimum condition (Fig. 6.7(b–d)). In that case the optimum occurs for a reduced shift of the tip of the corner, and occurs when the visible ends of the edges are exactly at opposite ends of a window diameter (Fig. 6.7(d)). Formally, we can see this in the symmetrical case ($l_1 = l_2$) from the following equation:

$$\lambda_{\text{sym}} = \frac{L^2/4}{L} = \frac{L}{4} \quad (6.20)$$

so reduction of L will reduce λ_{sym} and C will fall. This situation continues until $\theta = \pi$, at which point C again falls to zero.

We can now calculate the corner shift produced by the Harris detector. Specifically, the detector places the maximum output signal at the center of the window in the cases where the signal is stated to be “optimum” above. The shift produced has a size equal to radius a of the window for small corner angles, as

**FIGURE 6.7**

Possible geometries for right-angle and obtuse corners. (a) Optimum case for a right-angle corner. This is the right-angle case corresponding to that shown in Fig. 6.6(c). (b) General case for an obtuse corner. (c) Symmetrical placement with $l_1 = l_2$. (d) Case of maximum signal. In (d) the edges bounding the corner cross the boundary of the circular window at opposite ends of a diameter.

Source: © IET 2005

then the tip of the corner is symmetrically placed on the boundary of the window. When θ rises above $\pi/2$, simple geometry (Fig. 6.8(a)) shows that the shift is given by:

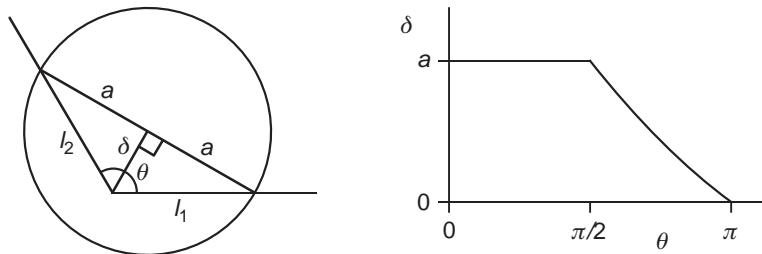
$$\delta = a \cot\left(\frac{\theta}{2}\right) \quad (6.21)$$

Hence δ starts with value a at $\theta=\pi/2$, and falls to zero as $\theta \rightarrow \pi$ (see Fig. 6.8(b)).

6.5.2 Performance with Crossing Points and Junctions

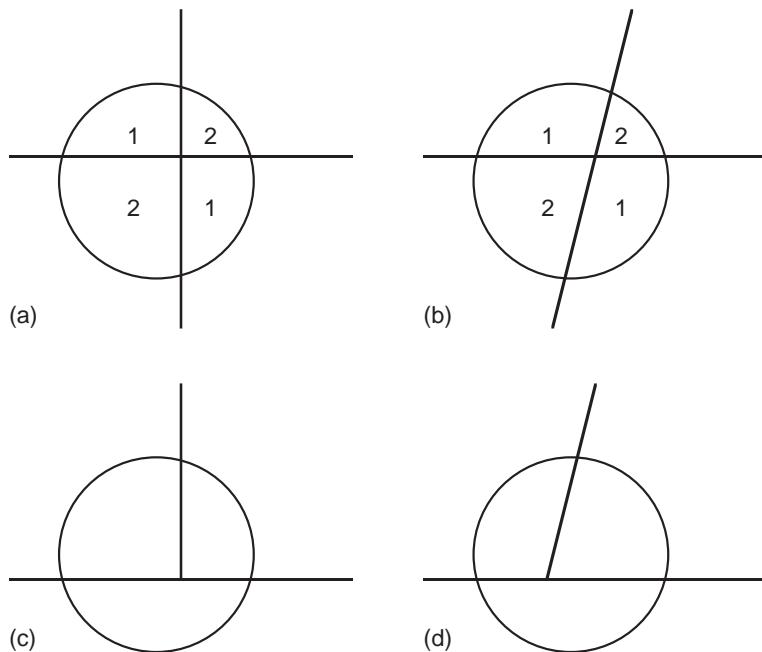
In this section we consider the performance of the Harris operator on other types of feature, which are not normally classed as simple corners. Examples are shown in Figs. 6.9 and 6.10. It turns out that the Harris operator picks these out with much the same efficiency as for corners. We start by considering crossing points.

One of the most important points to note is that many of the same equations apply as for corners, and in particular Eq. (6.17) still applies. However, l_1, l_2 must now be taken as the sum of the edge lengths in each of the two main directions. Here, there is an important point to note that along the two edge directions the signs of the contrast values both reverse at the crossing point. Nevertheless, this does not alter the response, because in Eq. (6.17) the contrast g is squared. So,


FIGURE 6.8

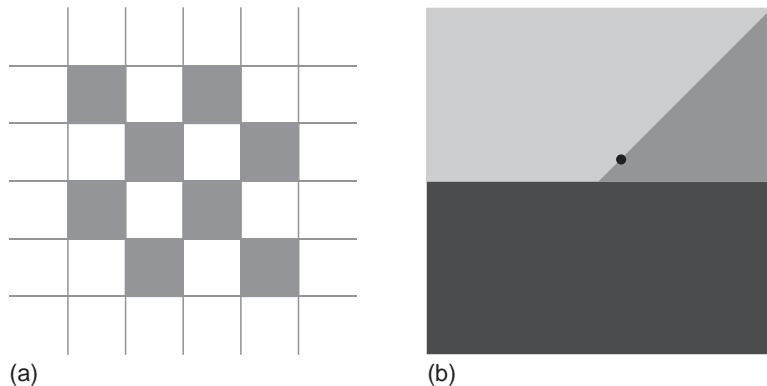
Geometry for calculating obtuse corner shifts and actual results. (a) Detailed geometry for calculating corner shift for the case shown in Fig. 6.7(d). (b) Graph showing corner shift δ as a function of corner sharpness θ . The left of the graph corresponds to the constant shift of a obtained for sharp corners, while the right shows the varying results for obtuse angles.

Source: © IET 2005


FIGURE 6.9

Other types of interest point. The types of interest point shown in this figure are those that cannot be classed as simple corner points. (a) Crossing point. (b) Oblique crossing point. (c) T-junction. (d) Oblique T-junction. In (a) and (b) the numbers indicate regions of equal, or different, intensity.

Source: © IET 2005

**FIGURE 6.10**

Effect of the Harris corner detector. (a) A checker-board pattern that gives high responses at each of the edge crossover points. Because of symmetry, the location of the peaks is exactly at the crossover locations. (b) Example of a T-junction. The black dot shows a typical peak location: in this case there is no symmetry to dictate that the peak must occur exactly at the T-junction.

when the window is centered at the crossing point, which is the tip of both constituent corners, the values of l_1, l_2 are doubled.

Another relevant factor is that the corner configuration is now symmetric about the crossing point, so by symmetry this must also be the position of maximum signal. In fact, the global maximum signal must occur when there is a maximum length of both edges within the window, and they must therefore be closely aligned along window diameters. These remarks apply both for $\pi/2$ and for oblique crossovers (Fig. 6.9(a and b) and 6.10(a)).

We now consider another case that arises fairly often—the T-junction interest point. This can be either a $\pi/2$ or an oblique junction. Such cases are more general than corners and the crossing point junctions discussed above, in that they are mediated by three regions with three different intensities (Figs. 6.9(c and d) and 6.10(b)). A complete analysis of the situation for all these cases cannot be undertaken here. Instead, we consider the interesting case of a high contrast edge that is reached but not crossed by a low contrast edge. In this case, the additional intensity breaks the symmetry of the junction, so that not only does the corner peak not lie on the junction point, but also there will be a small lateral movement of the peak. However, if the low contrast edge has much lower contrast than the other two, the lateral shift will be minimal. To calculate the corner signal, we first generalize Eq. (6.17) to take into account the fact that one line will have higher contrast than the other:

$$C' = \frac{l_1 l_2 g_1^2 g_2^2}{l_1 g_1^2 + l_2 g_2^2} \sin^2 \theta \quad (6.22)$$

where l_1 is taken as the straight edge with high contrast g_1 and l_2 is the straight edge with low contrast g_2 . Proceeding as before we find that the optimal signal occurs where $l_1|g_1| = l_2|g_2|$. Interestingly, this can mean that the maximum signal occurs on the low contrast edge, in a highly asymmetric way (Fig. 6.10(b)). Part of the motivation of this study was the observation that the Harris operator peaks shown in the literature (e.g. Shen and Wang, 2002) often seem to be localized at such points, though oddly this does not seem to have been remarked upon before 2005 when the author noted and explained the phenomenon (Davies, 2005). While apparently trivial it is actually important, as measurement bias can mislead and/or be the cause of error in subsequent algorithms. However, here the bias is known, systematic and calculable, and can be allowed for when the operator is used in practice.

Note that the Harris operator³ is often called an “interest” operator, as it detects not only corners but also other interesting points such as crossovers and T-junctions: and we have seen that there is good reason why this happens. Indeed, it is difficult to imagine that a second-order derivative signal would give sizeable signals in these other cases, as the coherence of the second derivative would be largely absent, or even identically zero in the case of a crossover.

6.5.3 Different Forms of the Harris Operator

In this section we consider the different forms the Harris operator can take. The form in Eq. (6.11) is due to Noble (1988) who actually gave the inverse of this expression and included a small positive constant in the denominator to prevent divide-by-zero situations. However, the original Harris operator had the rather different form:

$$C = \det\Delta - k(\text{trace}\Delta)^2 \quad (6.23)$$

where $k \approx 0.04$. Ignoring the constant, we find that the analysis presented above remains virtually unchanged, particularly concerning the optimal signal and the localization bias. The term involving k was added by Harris and Stevens (1988) in order to limit the number of false positives due to prominent edges. In principle, isolated edges should have no such effect, because as shown earlier, they lead to $\det\Delta = 0$. However, noise or clutter can affect this by introducing short extraneous edges that interact with any existing strong edges to constitute pseudo-corners:⁴ so k is adjusted empirically to minimize the number of false positives. Searching the literature shows that in practice workers almost invariably give k a value close to 0.04 or 0.05: in fact, Rocket has investigated this, and has found that: (a) making k equal to 0.04 rather than zero drastically cuts down the number

³Note also that the Harris operator often used to be called the Plessey operator, after the company at which it was originally developed.

⁴In the absence of explanations in the literature, this seems to be the most reasonable interpretation of the situation.

of false positives due to edges; (b) there appears to be an optimum value for k that is actually much closer to 0.05 than to 0.04, but definitely below 0.06, the k response function being a smoothly varying curve (Rocket, 2003). Nevertheless, we must expect the optimum value of k to vary with the image data.

Interestingly, in tests carried out using the Harris operator, the form given in Eq. (6.11) was used without any attempt to introduce a term in k (though divide-by-zero was taken care of), with the results shown in Fig. 6.11. Excessive numbers of false positives due to edges were not evident, though possibly this was so because of the lack of sensitivity to that effect with this particular type of data.

Finally, it should be pointed out that, when making direct theoretical comparisons between the Harris and other operators (such as the second-order derivative and median-based operators), the square roots of the expressions in Eqs. (6.11) and (6.17) will need to be taken to ensure that the result is directly proportional to edge contrast g .

6.6 CORNER ORIENTATION

This chapter has so far considered the problem of corner detection as relating merely to corner location. However, of the possible point features by which objects might be detected, corners differ from holes in that they are not isotropic, and hence are able to provide orientation information. Such information can be used by procedures that collate the information from various features in order to deduce the presence and positions of objects containing them. In Chapter 14, it will be seen that orientation information is valuable in immediately eliminating a large number of possible interpretations of an image, and hence of quickly narrowing down the search problem and saving computation.

Clearly, when corners are not particularly pointed (Fig. 6.12), or are detected within rather small neighborhoods, the accuracy of orientation will be somewhat restricted.⁵ However, orientation errors will seldom be worse than 45° , and will generally be less than 20° . Although these accuracies are far worse than those (around 1°) for edge orientation (see Chapter 5), nonetheless they provide valuable constraints on possible interpretations of an image.

Here we consider only simple means of estimating corner orientation. Basically, once a corner has been located accurately, it is a rather trivial matter to estimate its orientation from that of the intensity gradient at that location. This estimate can be made more accurate by finding the mean intensity gradient over a small region surrounding the estimated corner position, i.e. using the components $\langle I_x \rangle$ and $\langle I_y \rangle$.

⁵Clearly, accuracy of corner location will also suffer. However, a way of overcoming this problem will be described in Chapter 13, by making use of the generalized Hough transform.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

FIGURE 6.11

Application of the Harris interest point detector. (a) Original image. (b) Interest point feature strength. (c) Map of interest points showing only those giving greatest response over a distance of 5 pixels: (d) their placement in the original image. (e and f) Corresponding results for interest points giving greatest response

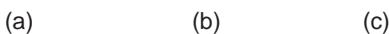


FIGURE 6.12

Types of corner: (a) pointed; (b) rounded; (c) chipped. Corners of type (a) are normal with metal components, those of type (b) are usual with biscuits and other food products, whereas those of type (c) are common with food products but rarer with metal parts.

Source: © IEE 1988

6.7 LOCAL INVARIANT FEATURE DETECTORS AND DESCRIPTORS

The discussion in the foregoing sections covered corner and interest point detectors useful for general-purpose object location, i.e. finding objects from their features. The specifications of the detectors were that they should be sensitive, reliable and accurate, so that there would be little chance of missing any object containing them, and so that object location would be accurate. In the context of the object inference schemes described in Part 2—and particularly in Chapter 14—it does not matter if some features are missing or whether additional noise or clutter features arise, as the inferential schemes are sufficiently robust to be able to find the objects in spite of this. However, the whole context was essentially the 2-D situation where it was good enough to imagine that the objects were nearly flat, or had nearly flat faces, so that 3-D perspective types of distortion could be avoided. Even so, in 3-D, corners appear as corners from almost any viewpoint, so robust inference algorithms should still be able to perform object location. However, when viewing objects from quite different directions in

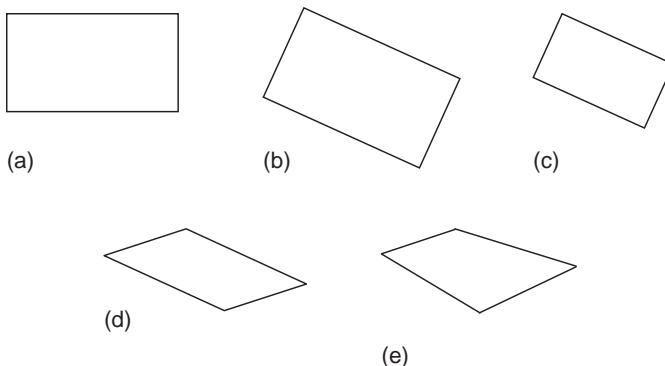
FIGURE 6.11 (Continued)

over a distance of 7 pixels. (g and h) Later frames in the sequence (also using maximum responses over a distance of 7 pixels), showing a high consistency of feature identification, which is important for tracking purposes. Note that interest points really do indicate locations of interest—corners, people's feet, ends of white road markings, and castle window and battlement features. Also, the greater the significance as measured by the pixel suppression range, the greater relevance the feature tends to have.

3-D, appearance can change dramatically, so it becomes extremely difficult to recognize them, even if all the features are present in the images. Thus, we arrive at the concept of viewing over wide baselines. In the case of binocular vision that takes two views over quite a narrow baseline (~ 7 cm for human eyes), the difference between the views is necessary in order to convey depth information, but it is rarely so great that features recognized in one view cannot be re-identified in the second view (though when huge numbers of similar, e.g. textural, features occur, as when viewing a piece of material, this may not apply). On the other hand, when objects are viewed on a wide baseline, as happens after significant motion has occurred, the angular separation between the views may be as large as 50° . If even larger angular separations occur, there will be much less possibility of recognition. While this sort of situation can be tackled by memorizing sequences of views of objects, here we concentrate primarily on what can be discerned from local features that are seen in wide baseline views of up to $\sim 50^\circ$.

At this point we have established the need to be able to recognize local features from wide baseline views as far apart as 50° so that objects can be recognized and tracked or found in databases without especial difficulty. Clearly, the corner and interest point detectors described thus far have no special provision for this. To achieve this aim, additional criteria have to be fulfilled. The first is that feature detection must be consistent and repeatable in spite of substantial change of viewpoint. The second is that features must embody descriptions of their localities so that there is high probability that the same physical feature will be positively identified in each of the views. Imagine that each image contains ~ 1000 corner features. Then there will be ~ 1 million potential feature matches between two views. While a robust inference scheme could perform the match in the case of flat objects, the situation becomes so much more demanding for general views of 3-D objects that matching might not be possible, either at all, or more likely, within a reasonable time—or without large numbers of ambiguities occurring. So, it is highly important to minimize the feature matching task. Indeed, ideally, if a rich enough descriptor is provided for each feature, feature matching might be reducible to one-to-one between views. At this stage we are extremely far from this possibility, as the corners that we have detected can so far only be characterized on the basis of their enclosed angle and intensity or color (and note that the first of these parameters will generally be substantially changed by the altered viewing angle). In what follows we consider in turn the two requirements of consistent, repeatable feature detection, and feature description.

Broadly speaking, obtaining consistent, repeatable feature detection involves allowing for and normalizing the variations between views. The obvious candidates for normalization are scale, affine distortion and perspective distortion. Of these, the first is straightforward, the second difficult, and the third impractical to implement. This is because of the number of parameters that need to be estimated for each feature. Bearing in mind that local features are necessarily small, the accuracy with which the parameters can be estimated decreases rapidly with increase in their number. [Figure 6.13](#) shows how various transformations affect a

**FIGURE 6.13**

Effects of various transformations on a convex 2-D shape. (a) Original shape. (b) Effect of Euclidean transform (translation + rotation). (c) Effect of similarity transform (change of scale). (d) Effect of affine transform (stretch + shear). (e) Effect of perspective transform. Note that in (d) parallel lines still remain parallel: this is not in general the case after a projective transform, as indicated in (e). Overall, each of the transforms illustrated is a generalization of the previous one. The respective numbers of degrees of freedom are 3, 4, 6, 8: in the last case each of the four points is independent and has two degrees of freedom, though there are constraints, such as convexity having to be maintained.

2-D shape. The following equations respectively define Euclidean, similarity (scale variation), and affine transformations:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \quad (6.24)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} sr_{11} & sr_{12} \\ sr_{21} & sr_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \quad (6.25)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \quad (6.26)$$

where rotation takes place through an angle theta, and the rotation matrix is:

$$\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (6.27)$$

Euclidean transformations allow translation and rotation operations and have three degrees of freedom (DoF); *additionally*, similarity transformations include scaling operations and have four DoF; *additionally*, affine transformations include stretching and shearing operations, have six DoF, and are the most complex of the transformations that make parallel lines transform into parallel lines; projective transformations are much more complex, have eight DoF, and include operations that (a) make parallel lines nonparallel, and (b) change *ratios* of lengths on straight

lines. The steady increase in the number of parameters is what mitigates against estimation of perspective distortions in the feature points: in fact, it also tends to reduce accuracy for the scale parameter when estimating full affine distortion.

6.7.1 Harris Scale and Affine-Invariant Detectors and Descriptors

Before proceeding to consider the above ideas in more detail, note that feature detectors such as the Harris operator already estimate location and orientation, so normalization for translation and rotation is already allowed for. This leaves scale as the next candidate for normalization. Here, the basic concept is to apply a given feature detector at various scales, using larger and larger masks. In the case of the Harris operator, there are two relevant scales: one is the edge detection (differentiation) scale σ_D and the other is the overall feature (integration) scale σ_I . In practice, these need to be linked together (this involves little loss of generality) so that $\sigma_I = \gamma\sigma_D$, where γ has a suitable value in the range 0–1 (typically ~ 0.5). σ_I then represents the scale of the overall operator. The approach is now to vary σ_I and to find the value that provides the best match of the operator to the local image data: the best match (extremum value) is the one representing the local image structure: it is intended to be independent of image resolution, which is arbitrary. In fact, the resulting “scale-adapted” Harris operator rarely attains true maxima over scales in such a (“scale-space”) representation (Mikolajczyk and Schmid, 2004); this is because a corner appears as a corner over a wide range of scales (Tuytelaars and Mikolajczyk, 2008). To achieve an optimal scale for matching, a totally different approach is applied: that is to use the Harris operator to locate a suitable feature point, and then to examine its surroundings to find the ideal scale, using a Laplacian operator. The scale of the latter is then adjusted to determine, in a matched filter (i.e. optimum signal-to-noise ratio) way, when the profile of the Laplacian most accurately matches the local image structure (Fig. 6.14). The required operator is called a Laplacian of Gaussian (LoG). It corresponds to smoothing the image using a Gaussian and then applying the Laplacian $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$ (see Chapter 5), and results in the following combined isotropic convolution operator:⁶

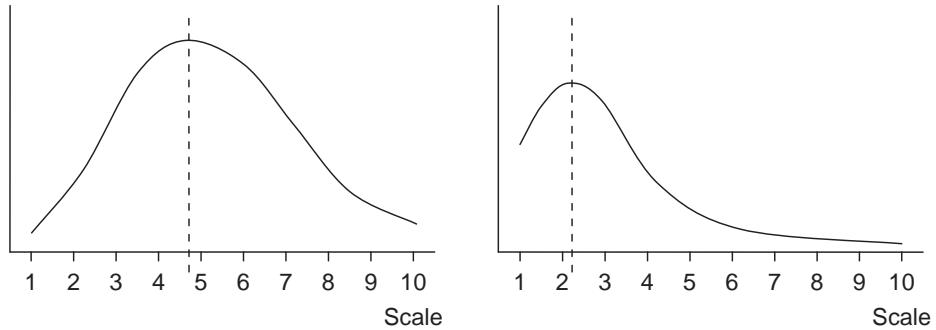
$$LoG = \frac{(r^2 - 2\sigma^2)}{\sigma^4(2\pi\sigma^2)} \exp\left(-\frac{r^2}{2\sigma^2}\right) = \frac{(r^2 - 2\sigma^2)}{\sigma^4} G(\sigma) \quad (6.28)$$

where

$$G(\sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad (6.29)$$

Having optimized this operator, we know the scale of the corner, and also its location and 2-D orientation. This means that when comparing two such corner

⁶Note that convolution (\otimes) is associative, so we have $\nabla^2 \otimes (G \otimes I) = (\nabla^2 \otimes G) \otimes I = LoG \otimes I$.

**FIGURE 6.14**

Scaling graphs for two objects that are being matched. The scaling graph on the left has an extremum at 4.7, and the one on the right has an extremum at 2.2. This shows that the best match occurs when the ratios of their scaling factors are approximately 2.14:1. The vertical scales of the graphs do not come into the optimization calculation.

features we can maintain translation, rotation, and scale invariance. To obtain affine invariance we estimate the affine shape of the corner neighborhood. Examining the Harris matrix Eq. (6.10), we rewrite it in the scale-adapted form:

$$\Delta = \sigma_D^2 G(\sigma_I) \otimes \begin{bmatrix} I_x^2(\sigma_D) & I_x(\sigma_D)I_y(\sigma_D) \\ I_x(\sigma_D)I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix} \quad (6.30)$$

where

$$I_x(\sigma_D) = \frac{\partial}{\partial x} G(\sigma_D) \otimes I \quad (6.31)$$

and similarly for $I_y(\sigma_D)$. These equations take full account of the differentiation and integration scales σ_D , σ_I . Then for each scale of the scale-adapted Harris operator, we repeat the process that was applied while determining the scale using the Laplacian, this time iteratively determining the best-fit ellipse (rather than circle) profile that fits the local intensity pattern. In fact, in spite of starting at separate scales, the resulting elliptic fits are, for well-defined corner structures, highly consistent and a robust average can be selected. The corresponding ellipse will (compared with the original circle) be stretched by different amounts in two perpendicular directions: the degrees of stretch and skew are the output affine parameters.

The final step is to normalize the feature by transforming it so that the elliptic profile becomes isotropic, circular and therefore affine invariant (i.e. the affine deformation is nullified). This corresponds to equalizing the eigenvalues of the optimum scale-adapted second-order matrix, Eq. (6.30).

When comparing two corners we require invariant parameters. To obtain these descriptors, it is necessary to determine Gaussian derivatives of the local neighborhood of the interest points, computed on the transformed isotropic feature profile. Clearly, the Gaussian derivatives have to be adjusted for a standardized

isotropic profile size, and they have to be normalized to intensity variations by dividing the higher order derivatives by the first-order derivative (i.e. the average intensity gradient in the neighborhood). In the work of Mikolajczyk and Schmid (2004), descriptors of dimension 12 were obtained by using derivatives up to fourth order. (There are two first-order derivatives, three second order, four third order, and five fourth order: excluding the first-order derivatives, this leaves a total of 12 up to fourth order.) This set of descriptors proved highly effective for identifying corresponding pairs of features in widely different views of up to $\sim 40^\circ$ angular separation with better than 40% repeatability, and in the affine case up to $\sim 70^\circ$ with up to 40% repeatability. In addition, the localization accuracy for Harris–Laplace dropped off more or less linearly with angular separation, becoming excessive above 40° , whereas that for Harris–Affine remained at an acceptable level (~ 1.5 pixel error). The Harris–Laplace was described as having a breakdown point at a viewpoint change of 40° .

6.7.2 Hessian Scale and Affine-Invariant Detectors and Descriptors

Over the same period that scale and affine-invariant detectors and descriptors based on the Harris operator were developed, investigations of similar operators based on the Hessian operator were being undertaken. Here it is useful to recall that the Harris operator is defined in terms of first derivatives of the intensity function I , while the Hessian operator (see Eq. (6.5)) is defined in terms of the second derivatives of I . Thus, we can consider the Harris operator as being edge-based, and the Hessian operator as being blob-based. This matters for two reasons. One is that the two types of operator might, and do, bring in different information about objects and hence to some extent they are complementary. The other is that the Hessian is better matched than the Harris to the Laplacian scale estimator: indeed, the Hessian arises from the determinant and the Laplacian from the trace of the matrix of second-order derivatives (Eq. (6.2)). The better matching of the Hessian to the Laplacian results in improved scale selection accuracy for this operator (Mikolajczyk and Schmid, 2005). The other details of the Hessian–Laplacian and Hessian–Affine operators are similar to those for the corresponding Harris operators and will not be discussed in more detail here. However, it is worth remarking that in all four cases there are typically 200–3000 detected regions per image depending on the content (Mikolajczyk and Schmid, 2005).

6.7.3 The SIFT Operator

Lowe's scale invariant feature transform (widely known as “SIFT”) was first introduced in 1999, a much fuller account being given by Lowe (2004). While being restricted to a scale invariant version, it is important for two reasons: (1) for impressing on the vision community the existence, importance, and value of

invariant types of detector; and (2) for demonstrating the richness that feature descriptors can bring to feature matching. For estimating scale, the SIFT operator uses the same basic principle as for the Harris and Hessian-based operators outlined above. However, it differs in using the Difference of Gaussians (DoG) instead of the Laplacian of Gaussians (LoG), in order to save computation. This possibility is seen by differentiating G with respect to σ in Eq. (6.29):

$$\frac{\partial G}{\partial \sigma} = \left(\frac{r^2}{\sigma^3} - \frac{2}{\sigma} \right) G(\sigma) = \sigma \text{LoG} \quad (6.32)$$

which means that we can approximate LoG as the difference of Gaussians of two scales:

$$\text{LoG} \approx \frac{G(\sigma') - G(\sigma)}{\sigma(\sigma' - \sigma)} = \frac{G(k\sigma) - G(\sigma)}{(k - 1)\sigma^2} \quad (6.33)$$

where use of the constant scale factor k permits scale normalization to be carried out easily between scales.

In fact, it is in the design of the descriptors that SIFT is particularly different from the Harris and Hessian-based detectors. Here the operator divides the support region, at each scale, into a 16×16 sample array and estimates the intensity gradient orientations for each of these. They are then grouped into sets of sixteen 4×4 sub-arrays and orientation histograms are generated for each of these, the directions being restricted to one of eight directions. The final output is a 4×4 array of histograms each containing entries for eight directions—amounting to a total output dimensionality of $4 \times 4 \times 8 = 128$.

The overall detector is found (Mikolajczyk, 2002) to be more repeatable than Harris–Affine and to retain a final matching accuracy above 50% out to a 50° angular separation. However, because of the limited stability of Harris–Affine, Lowe (2004) recommends the approach of Pritchard and Heidrich (2003) of including additional SIFT features with 60° viewpoint separation during training. We defer further discussion of the performance of this detector to Section 6.7.6.

6.7.4 The SURF Operator

The development of SIFT stimulated efforts to produce an effective invariant feature detector that was also highly efficient and required a smaller descriptor than the large one employed by SIFT. An important operator in this mold was the speeded-up robust features (SURF) method of Bay et al. (2006, 2008). This was based on the Hessian–Laplace operator. In order to increase speed, several measures were taken: (1) the integral image approach was used to perform rapid computation of the Hessian and was also used during scale-space analysis; (2) the DoG was used in place of the LoG for assessing scale; (3) sums of Haar wavelets were used in place of gradient histograms, resulting in a descriptor dimensionality

of 64—half that of SIFT; (4) the sign of the Laplacian was used at the matching stage; (5) various reduced forms of the operator were used to adapt it to different situations, notably an “upright” version capable of recognizing features within $\pm 15^\circ$ of those pertaining to an upright stance, as occurs for outdoor buildings and other objects. By maintaining a rigorous, robust design, the operator was described as outperforming SIFT, and also proved capable of estimating 3-D object orientation within fractions of a degree and certainly more accurately than SIFT, Harris–Laplace, and Hessian–Laplace.

Of some importance for this implementation is the integral image approach (Simard et al., 1999): this was brought prominently to light by Viola and Jones (2001), but maybe not utilized as much as one might expect since then. This is extremely simple, yet radical in the levels of speedup it can bring. It involves computing an integral image I_Σ , which is an image that retains sums of all pixel intensities encountered so far in a single scan over the input image:

$$I_\Sigma(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \quad (6.34)$$

This not only permits any pixel intensity in the original image to be recovered:

$$I(i, j) = I_\Sigma(i, j) - I_\Sigma(i - 1, j) - I_\Sigma(i, j - 1) + I_\Sigma(i - 1, j - 1) \quad (6.35)$$

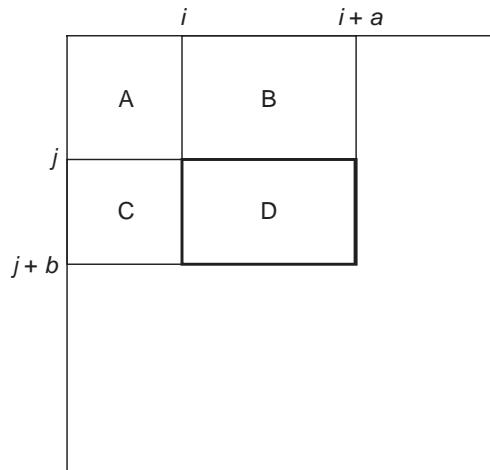
but also allows the sum of the pixel intensities in any upright rectangular block, such as those ranging from $x = i$ to $i + a$ and $y = j$ to $j + b$ within block D in Fig. 6.15, to be utilized:

$$\begin{aligned} \sum_D I &= \sum_A I - \sum_{A,B} I - \sum_{A,C} I + \sum_{A,B,C,D} I \\ &= I_\Sigma(i, j) - I_\Sigma(i + a, j) - I_\Sigma(i, j + b) + I_\Sigma(i + a, j + b) \end{aligned} \quad (6.36)$$

The method is exceptionally well adapted to computing Haar filters that typically consist of arrays containing blocks of identical values, for example:

$$\begin{bmatrix} -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \end{bmatrix}$$

Note that once the integral image has been computed, it permits summations to be made over any block merely by performing four additions—taking a minuscule time that is independent of the size of the block. A simple generalization to the 3-D box filter (Simard et al., 1999) is also possible, and this is used in computing within scale-space in the SURF implementation.

**FIGURE 6.15**

The integral image concept. Here block D can be considered as made up by taking block A + B + C + D, then subtracting block A + B and block C, in the latter case by subtracting A + C and adding A: see text for an exact mathematical treatment.

6.7.5 Maximally Stable Extremal Regions

There is one class of invariant feature that does not fall into the pattern covered in the preceding sections—the invariant region type of feature. Among the most important examples of this type of feature is the maximally stable extremal region (MSER). The method analyzes regions with increasing ranges of intensity, and aims to determine those that are extremal in a particularly stable way. (Recall that finding extrema is a powerful general method for locating invariant features, as we have already seen in [Section 6.7.1](#).)

The method (Matas et al., 2002) starts by taking pixels of zero intensity and progressively adding pixels with higher intensity levels, at each stage monitoring the regions that form. At each stage largest connected regions or “connected components” will represent extremal regions.⁷ As more and more gray levels are added, the connected component regions will grow and some initially separate ones will merge. Maximally stable extremal regions are those connected components that are close to stable (as conveniently measured by their area) over a range of intensities, i.e. each MSER is represented by the position of a local intensity minimum in the rate of change of the area function. Interestingly, relative area change is an affine-invariant property, so finding MSER regions guarantees both scale and affine

⁷See [Section 9.3](#) for a full explanation of connected components and their computation. Meanwhile, assume that a “connected component” means a region *containing everything that is connected to any part of that region*. Hence, by definition, a connected component is an extremal region.

invariance. In fact, because of the way intensities are handled in this method, the results are also independent of monotonic transformation of image intensities.

While every MSER can be regarded as a connected component of a threshold image, global thresholding is not carried out, and optimality is judged on the basis of the stability of the connected components that are located. Intrinsically, MSERs have arbitrary shapes, though for matching purposes they can be converted into ellipses of appropriate areas, orientations and moments. Perhaps surprisingly for affine features, they can be computed highly efficiently in times that are nearly linear in the number of pixels; they also have good repeatability, though they are quite sensitive to image blur. This last problem is to be expected, given the dependence on individual gray levels and the precision with which connected components analysis is carried out; in fact, this difficulty has been addressed in a recent extension of the work (Perdoch et al., 2007).

6.7.6 Comparison of the Various Invariant Feature Detectors

While there are many more invariant feature detectors than we have been able to cover here (salient regions, IBR, FAST, SFOP, ...) and many variants of them (GLOH, PCA-SIFT, ...), we next concentrate on comparisons between them. In fact, most of the papers describing new detectors make comparisons with older detectors, but often with limited datasets. Here we outline the conclusions of Ehsan et al. (2010), who compared SIFT, SURF, Harris–Laplace, Harris–Affine, Hessian–Laplace, and Hessian–Affine, using the following datasets: Bark, Bikes, Boat, Graffiti, Leuven, Trees, UBC, and Wall (viz. eight sequences of six images).⁸ Table 6.1 presents the results in a modified form with SURF placed after Hessian–Laplace, as it is based on the latter.

Apart from Table 6.1, Ehsan et al. (2010) showed the results of using three different criteria for judging repeatability of feature detector performance. The first was the standard repeatability criterion:

$$C_0 = \frac{N_{\text{rep}}}{\min(N_1, N_2)} \quad (6.37)$$

where N_1 is the total number of points detected in the first image, N_2 is the total number of points detected in the second image, and N_{rep} is the number of repeated points.

They emphasized that it has been remarked (Tuytelaars and Mikolajczyk, 2008) that repeatability “does not guarantee high performance in a given application.” They reasoned that this was due in part to comparing features within adjacent pairs of images rather than over whole image sequences: specifically, they

⁸See Oxford DataSets at: <http://www.robots.ox.ac.uk/~vgg/research/affine/> (website accessed 19 April 2011).

Table 6.1 Comparison of Invariant Feature Detectors

Datasets	SIFT	Harris—Laplace	Hessian—Laplace	SURF	Harris—Affine	Hessian—Affine	Total
Bark	—	—	—	—	—	—	9
Bikes	—	—	—	—	—	—	14
Boat	—	—	—	—	—	—	12
Graffiti	—	—	—	—	—	—	10
Leuven	—	—	—	—	—	—	12
Trees	—	—	—	—	—	—	13
UBC	—	—	—	—	—	—	17
Wall	—	—	—	—	—	—	14
total	16	14	18	20	15	18	101

The totals give some indication of the overall capabilities of the detectors, and of the complexity of the individual datasets. However, the detector totals must be interpreted in the light of the highest level of invariance achievable—viz. scale or affine.

recommended that each image should be compared taking the first frame of the sequence as a reference and using the following criterion:

$$C_1 = \frac{N_{\text{rep}}}{N_{\text{ref}}} \quad (6.38)$$

Nevertheless, they also proposed a more symmetric measure of repeatability:

$$C_2 = \frac{N_{\text{rep}}}{(N_{\text{ref}} + N_c)} \quad (6.39)$$

where N_c is the total number of points detected in the current frame. This proved to be a less harsh and more realistic criterion when compared with the trends of the observed ground truth for an image sequence. Further evidence for moving away from the standard repeatability criterion C_0 is that it rewards failure to detect features (because decrease in N_1 or N_2 will, if anything, *raise* the value of C_0). This suggests altering C_0 to use the maximum instead of the minimum. However, using either the maximum or the minimum tends to emphasize extreme results, leading to nonrobust measures. From this point of view the most appropriate measure has to be C_2 . In fact, this criterion gave optimal results and low error probability measures when run against ground truth using Pearson's correlation coefficients (Ehsan et al., 2010). Using C_2 , an important result was the dominance of the Hessian-based detectors, which is already very evident in Table 6.1 (derived from their Table 2), where the three Hessian-based totals are 18, 18, 20, vis-à-vis 14, 15, 16 for the others (interestingly, the situation is even more polarized in favor of Hessian-based detectors when the datasets giving the best and worst results—Bark and UBC—are ignored). Note that Tuytelaars and Mikolajczyk (2008) did not come out so strongly in favor of the Hessian-based

Table 6.2 Performance Evaluation of Various Feature Detectors

Detector	Invariance	Repeatability	Accuracy	Robustness	Efficiency	Total
Harris	Rotation	—	—	—	—	11
Hessian	Rotation	—	—	—	—	7
SIFT	Scale	—	—	—	—	8
Harris–Laplace	Scale	—	—	—	—	9
Hessian–Laplace	Scale	—	—	—	—	10
SURF	Scale	—	—	—	—	9
Harris–Affine	Affine	—	—	—	—	10
Hessian–Affine	Affine	—	—	—	—	11
MSER	Affine	—	—	—	—	11

The totals give some indication of the overall capabilities of the detectors: however, they must be interpreted in the light of the highest level of invariance achievable (column 2).

detectors, but this was probably because their analysis of datasets was not so extensive; also, Ehsan et al. (2010) were the first to look at image sequences so rigorously using C_2 .

The review by Tuytelaars and Mikolajczyk (2008) is of great value in evaluating performance using several disparate criteria, viz. repeatability, localization accuracy, robustness and efficiency. Some of their results are shown in Table 6.2—notably those for all the feature detectors covered in Table 6.1 and those for the single-scale Harris and Hessian, and for the MSER detector (Matas et al., 2002) mentioned earlier. They also make the following valuable observations:

1. Scale invariant operators can normally be dealt with adequately by a robustness capability for viewpoint changes of less than 30° , as affine deformations only rise above those due to variations in object appearance beyond that level.
2. In different applications, different feature properties may be important, and thus success depends largely on appropriate selection of features.
3. Repeatability may not always be the most important feature performance characteristic: not only is it hard to define and measure but robustness to small appearance variations matters more.
4. There is a need for work focussing on complementarity of features, leading either to complementary detectors or to detectors providing complementary features.

In the last respect, note that some ground work has recently been carried out by Ehsan et al. (2011) to measure the coverage of interest point detectors. They identify the recent SFOP scale invariant feature transform (Förstner et al., 2009) as the most outstanding detector in this respect, either used on its own or in

conjunction with others. Ehsan et al.'s new criterion for coverage C is based on the harmonic mean so as not to overemphasize nearby features:

$$C = \frac{N(N - 1)}{\sum_{i=1}^{N-1} \sum_{j>1}^N (1/d_{ij})} \quad (6.40)$$

(In this formula d_{ij} is the Euclidean distance between feature points i and j .)

Note that a detector which has high coverage is not guaranteed to be sound: after all, a detector giving a random selection of feature points might fare well on this count. Hence, a coverage criterion can only come into its own when it is used to select complementary types of feature and feature detector, or a detector that provides a good mix of types of feature, for which the output selection is known to be sound on other counts (repeatability, robustness, and so on).

6.8 CONCLUDING REMARKS

Corner detection provides a useful start to the process of object location, and to this end is often used in conjunction with the abstract pattern matching approaches discussed in Chapter 14. Apart from the obvious template matching procedure, which is of limited applicability, three main approaches have been described. The first was the second-order derivative approach that includes the KR, DN, and ZH methods—all of which embody the same basic schema; the second was the median-based method, which turned out to be equivalent to the second-order derivative methods in situations where corners have smoothly varying intensity functions; and the third was the Harris detector which is based on the matrix of second moments of the *first* derivatives of the intensity function. Perhaps surprisingly, the latter is able to extract much the same information as the other two approaches, though there are differences, in that the Harris detector is better described as an interest point detector than as a corner detector. In fact, the Harris detector has probably been the most widely used corner and interest point detector of all, and for general purpose (non-3-D) operation this still seems to be the case—in spite of the advent of the SUSAN detector (Smith and Brady, 1997), which is known to be faster and more efficient, but somewhat less resistant to noise.

Interestingly, the situation presented above started changing radically from about 1998, when workers started looking for approaches to object location that were not merely robust to noise, distortion, partial occlusion, and extraneous features, but were able to overcome problems of gross distortion due to viewing the same scene from widely different directions. This “wide baseline” problem, which is prominent with 3-D and motion applications, including tracking moving objects, became the driving force for radical new thinking and development. As we have seen, attempts were made to adapt the Harris operator to this scenario, making it invariant to similarity (scale) and affine transformations, though in the

end somewhat more success was achieved by returning to the Hessian operator discussed early in the chapter. Alternative approaches included the MSER approach, which is by no means based on the location of any sort of corner or interest point. Indeed, it harks back to the thresholding methods of Chapter 4. But this is all to the good, as the underlying task is that of segmentation coupled with recognition and identification/matching: division of the subject into watertight topics, such as thresholding, edge detection, and corner detection, has limited validity or at least it is too restrictive to offer the best solutions to the real problems of the subject. In this context it is relevant that the newly evolved feature detectors embody multiparameter descriptors, making them far better suited not only to detection *per se* but also to the more exacting task of wide baseline 3-D matching.

Overall, in this chapter we have seen the corner detector approach transmogrify itself to overcome the problems of viewing objects from directions as far apart as 70°—and with a great deal of success. Remarkably, all this was achieved in little more than a decade—evidence that progress in this subject has been accelerating. Importantly, the old computer adage “garbage in—garbage out” is relevant, because feature detection forms a crucial link between the original pictures and their high-level interpretation.

This chapter has studied how objects may be detected and located from their corners and interest points. It has developed both the classic approach to detector design and the more recent invariant approaches, which result in multiparameter feature descriptors to aid matching between widely separated views of objects.

6.9 BIBLIOGRAPHICAL AND HISTORICAL NOTES

The subject of corner detection has been developing for over three decades. The scene was set for the development of parallel corner detection algorithms by Beaudet’s (1978) work on rotationally invariant image operators. This was soon followed by Dreschler and Nagel’s (1981) more sophisticated second-order corner detector: the motivation for this research was to map the motion of cars in traffic scenes, corners providing the key to unambiguous interpretation of image sequences. One year later, Kitchen and Rosenfeld (1982) had completed their study of corner detectors based mainly on edge orientation, and had developed the second-order KR method described in Section 6.3. Years 1983 and 1984 saw the development of the second-order ZH detector and the median-based detector (Zuniga and Haralick, 1983; Paler et al., 1984). Subsequently, the author published work on the detection of blunt corners (see Chapter 13) and on analyzing and improving the median-based detector (Davies, 1988a,d, 1992a). Meanwhile, other methods had been developed, such as the Harris algorithm (Harris and Stephens, 1988; see also Noble, 1988). The Smith and Brady (1997) “SUSAN”

algorithm marked a further turning point, needing no assumptions on the corner geometry, as it works by making simple comparisons of local gray levels: this is one of the most cited of all corner detection algorithms.

In the year 2000s further corner detectors were developed. Lüdtke et al. (2002) designed a detector based on a mixture model of edge orientation: in addition to being effective in comparison with the Harris and SUSAN operators, particularly at large opening angles, the method provides accurate angles and strengths for the corners. Olague and Hernández (2002) worked on a unit step edge function (USEF) concept, which is able to model complex corners well: this resulted in adaptable detectors that are able to detect corners with subpixel accuracy. Shen and Wang (2002) described a Hough transform-based detector: as this works in a 1-D parameter space it is fast enough for real-time operation; a useful feature of the paper is the comparison with, and between, the Wang and Brady detector, the Harris detector, and the SUSAN detector. The several example images show that it is difficult to be sure exactly what one is looking for in a corner detector (i.e. corner detection is an ill-posed problem), and that even the well-known detectors sometimes inexplicably fail to find corners in obvious places. Golightly and Jones (2003) present a practical problem in outdoor country scenery: they discuss not only the incidence of false positives and false negatives but also the probability of correct association in corner matching, e.g. during motion.

Rocket (2003) gives a performance assessment of three corner detection algorithms—the KR detector, the median-based detector, and the Harris detector: the results are complex, and the three detectors are found to have very different characteristics. The paper is valuable in showing how to optimize the three methods (not least showing that the Harris detector parameter k should be ~ 0.05), and also because it concentrates on careful research rather than “selling” a new detector. Tissainayagam and Suter (2004) gave an assessment of the performance of corner detectors, with vitally important coverage of point feature (motion) tracking applications. Interestingly, it finds that, in image sequence analysis, the Harris detector is more robust to noise than the SUSAN detector, a possible explanation being that it “has a built-in smoothing function as part of its formulation.” Finally, Davies (2005) analyzed the localization properties of the Harris operator: see [Section 6.5](#) for the main results of this work.

While the above discussion covers many of the developments on corner detection, it is not the whole story. This is because in many applications it is not specific corner detectors that are needed but “interest point” detectors, which are capable of detecting any characteristic patterns of intensity that can be used as reliable feature points. In fact, the Harris detector is often called an interest point detector—with good reason, as indicated in [Section 6.5.2](#). Moravec (1977) was among the first to refer to interest points and was followed by Schmid et al. (2000) and many others. However, Sebe and Lew (2003) and Sebe et al. (2003) call them salient points—a term more often reserved for points that attract the attention of the *human* visual system. Overall, it is probably safest to use the Haralick and Shapiro (1993) definition: a point being “interesting” if it is both

distinctive and invariant—i.e. it stands out and is invariant to geometric distortions such as might result from moderate changes in scale or viewpoint (note that Haralick and Shapiro also list other desirable properties—stability, uniqueness, and interpretability). The invariance aspect is taken up by Kenney et al. (2003) who show how to remove ill-conditioned points from consideration, to make matching more reliable.

The subject of invariant feature detectors and descriptors has taken little over a decade to develop and over that period it has come a long way. It started with papers by Lindeberg (1998) and Lowe (1999) that indicated the way forward and provided basic techniques. It arose largely because of difficulties in wide baseline stereo work, and with tracking object features over many video frames—because features change their appearance over time and correspondences are easily lost. To proceed, it was necessary first to eliminate the relatively simple problem of features changing in size, thereby necessitating scale invariance (it being implicit that translation and rotation invariance have already been dealt with). Later, improvements became necessary to cope with affine invariance. Thus, Lindeberg’s pioneering theory (1998) was soon followed by Lowe’s work (1999, 2004) on scale invariant feature transforms (SIFT). This was followed by affine-invariant methods developed by Tuytelaars and Van Gool (2000), Mikolajczyk and Schmid (2002, 2004), Mikolajczyk et al. (2005) and others. In parallel with these developments, work was published on maximally stable extremal regions (Matas et al., 2002) and other extremal methods (e.g. Kadir and Brady, 2001; Kadir et al., 2004).

Much of this work capitalized on the interest point work of Harris and Stephens (1988), and was underpinned by careful in-depth experimental investigations and comparisons (Schmid et al., 2000; Mikolajczyk and Schmid, 2005; Mikolajczyk et al., 2005). Next, the tide turned in other directions, in particular the design of feature detectors that aim at real-time operation—as in the case of the speeded-up robust features (SURF) approach (Bay et al., 2006, 2008).

A review article summarizing the main approaches was published by Tuytelaars and Mikolajczyk in 2008. However, that was by no means the end of the story. As outlined in [Section 6.7.6](#), Ehsan et al. (2010) briefly reviewed the status quo on repeatability of the main features and detectors, and reported on experiments to assess it: their work included two new repeatability criteria, which more realistically reflected the underlying requirements for invariant detectors. In addition, they presented new work (Ehsan et al., 2011) on the coverage of invariant feature detectors, reflecting poignant remarks made in Tuytelaars and Mikolajczyk’s (2008) review. It is clear that, with the passing of the first decade of the 2000s, an even more exacting phase of development is under way, with more rigorous performance evaluation: it will no longer be sufficient to produce new invariant feature detectors; instead it will be necessary to integrate them much more fully with the target applications, following rigorous design to ensure that all relevant criteria are being met, and that the tradeoffs between the criteria are much more transparent.

6.9.1 More Recent Developments

Since the Tuytelaars and Mikolajczyk's (2008) review, further relevant work on feature detectors and descriptors has emerged. Rosten et al. (2010) have presented the FAST family of corner detectors, which is designed on a new heuristic to be especially fast while at the same time to be highly repeatable; they also review methods for comparing feature detectors and call for less concentration on how a feature detector should do its job than on what performance measure it is required to optimize. Cai et al. (2011) work on a “linear discriminant projections” procedure for reducing the dimensionality of local image descriptors, and manage to bring the SIFT tally down from 128 to just 30. However, they warn that this seems to be achievable only by making the projections specific to the type of image data. With a similar motivation, Teixeira and Corte-Real (2009) quantize the SIFT descriptor to form visual words using a predefined vocabulary, though in this case the vocabulary is structured in the form of a tree; it is constructed using a generic dataset related to the type of object tracking being performed. van de Sande et al. (2010) discuss the generation of color object descriptors. They find that the choice of a single color descriptor for all categories of data is suboptimal: but for unknown data, “OpponentSIFT” (using three sets of SIFT features for the three opponent colors) showed the highest degree of invariance with respect to photometric variations. Zhou et al. (2011) proposed a method to perform descriptor combination and classifier fusion. They cast the problem of object classification into a learning setting, which again means that the method is adaptive and not applicable to new data without retraining. Overall, we see that trying to reduce the original 128 SIFT features (or the equivalent) tends to make such methods specific to particular training data.

6.10 PROBLEMS

1. By examining suitable binary images of corners, show that the median corner detector gives a maximal response within the corner boundary rather than half-way down the edge outside the corner. Show how the situation is modified for grayscale images. How will this affect the value of the gradient noise-skimming threshold to be used in the improved median detector?
2. Prove Eq. (6.6), starting with the following formula for curvature:

$$\kappa = \frac{d^2y/dx^2}{[1 + (dy/dx)^2]^{3/2}}$$

Hint: First express dy/dx in terms of the components of intensity gradient, remembering that the intensity gradient vector (I_x, I_y) is oriented along the edge normal; then replace the x, y variation by I_x, I_y variation in the formula for κ .

Mathematical Morphology

7

Historically, the study of shape took place over a long period of time and resulted in a highly variegated set of algorithms and methods. Over the past 30 years the formalism of mathematical morphology was set up, and provided a background theory into which many of the individual advances could be slotted. This chapter takes a journey through this interesting subject, but aims to steer an intuitive path between the many mathematical theorems, concentrating on finding practically useful results.

Look out for:

- how the concepts of expanding and shrinking are transformed into the more general concepts of dilation and erosion.
- how dilation and erosion operations may be combined to form more complex operations whose properties may be predicted mathematically.
- how the concepts of closing and opening are defined, and how they are used to find defects in binary object shapes, via residue (or “top hat”) operations.
- how mathematical morphology is generalized to cover grayscale processing.
- how noise affects morphological grouping operations.

This theory in the present chapter is especially valuable because of the way in which it integrates a range of topics. Once the methodology has been learnt, morphology should be of distinct value in taking the earlier ideas forward and optimizing algorithms that use them.

7.1 INTRODUCTION

In Chapter 2, we have discussed the operations of erosion and dilation: in Chapter 9 we will apply them to the filtering of binary images, and will show that with suitable combinations of these operators it is possible to eliminate certain types of object from images, and also to locate other objects. These possibilities are not fortuitous, but on the contrary reflect fundamental properties of shape, which are dealt with in the subject known as mathematical morphology. This

subject has grown up over the past two or three decades, and over the past few years knowledge in this area has become consolidated and is now understood in considerable depth. It is the purpose of this chapter to give some insight into this vital area of study. Note that mathematical morphology is especially important because it provides a backbone for the whole study of shape and thus is able to unify techniques as disparate as noise suppression, shape analysis, feature recognition, skeletonization, convex hull formation, and a host of other topics.

[Section 7.2](#) starts the discussion by extending the concepts of expanding and shrinking first encountered in Section 2.2. [Section 7.3](#) then develops the theory of mathematical morphology, arriving at many important results—with emphasis deliberately being placed on understanding of concepts rather than mathematical rigor. [Section 7.4](#) goes on to show how morphology can be generalized to cope with grayscale images. The chapter also includes a discussion ([Section 7.5](#)) on the noise behavior of morphological grouping operations and arrives at a formula explaining the shifts introduced by noise.

7.2 DILATION AND EROSION IN BINARY IMAGES

7.2.1 Dilation and Erosion

As we have seen in Chapter 2, dilation expands objects into the background and is able to eliminate “salt” noise within an object. It can also be used to remove cracks in objects that are less than 3 pixels in width.

In contrast, erosion shrinks binary picture objects, and has the effect of removing “pepper” noise. It also removes thin object “hairs” whose widths are less than 3 pixels.

As we shall see in more detail below, erosion is strongly related to dilation, in that a dilation acting on the inverted input image acts as an erosion, and vice versa.

7.2.2 Cancellation Effects

An obvious question is whether erosions cancel out dilations, or vice versa. We can easily answer this question: for if a dilation has been carried out, salt noise and cracks will have been removed, and once they are gone, erosion cannot bring them back; hence, exact cancellation will not occur in general. Thus, for the set S of object pixels in a general image I , we may write:

$$\text{erode}(\text{dilate}(S)) \neq S \quad (7.1)$$

equality only occurring for certain specific types of image (these will lack salt noise, cracks, and fine boundary detail). Similarly, pepper noise or hairs that are eliminated by erosion will not in general be restored by dilation:

$$\text{dilate}(\text{erode}(S)) \neq S \quad (7.2)$$

Overall, the most general statements that can be made are:

$$\text{erode}(\text{dilate}(S)) \supseteq S \quad (7.3)$$

$$\text{dilate}(\text{erode}(S)) \subseteq S \quad (7.4)$$

We may note, however, that large objects will be made 1 pixel larger all round by dilation, and will be reduced by 1 pixel all round by erosion, so a considerable amount of cancellation will normally take place when the two operations are applied in sequence. This means that sequences of erosions and dilations provide a good basis for filtering noise and unwanted detail from images.

7.2.3 Modified Dilation and Erosion Operators

It sometimes happens that images contain structures that are aligned more or less along the image axes' directions, and in such cases it is useful to be able to process these structures differently. For example, it might be useful to eliminate fine vertical lines, without altering broad horizontal strips. In that case the following “vertical erosion” operator will be useful:

```
for all pixels in image do {
    sigma = A1 + A5;
    if (sigma < 2) B0 = 0; else B0 = A0;
}
```

although it will be necessary to follow it with a compensating dilation operator¹ so that horizontal strips are not shortened:

```
for all pixels in image do {
    sigma = A1 + A5;
    if (sigma > 0) B0 = 1; else B0 = A0;
}
```

This example demonstrates some of the potential for constructing more powerful types of image filter. To realize these possibilities, we next develop a more general mathematical morphology formalism.

7.3 MATHEMATICAL MORPHOLOGY

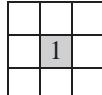
7.3.1 Generalized Morphological Dilation

The basis of mathematical morphology is the application of set operations to images and their operators. We start by defining a generalized dilation mask as a set of locations within a 3×3 neighborhood. When referred to the center of the neighborhood as origin, each of these locations causes a shift of the image in the direction defined by the vector from the origin to the location. When several

¹Here and elsewhere in this chapter, any operations required to restore the image to the original image space are not considered or included.

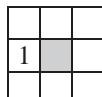
shifts are prescribed by a mask, the 1 locations in the various shifted images are combined by a set union operation.

The simplest example of this type is the identity operation I , which leaves the image unchanged:

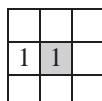


(Note that we leave the 0s out of this mask, as we are now focussing on the set of elements at the various locations, and set elements are either present or absent.)

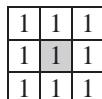
The next operation to consider is:



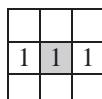
which is a left shift, equivalent to the one discussed in Section 2.2. Combining the above two operations into a single mask:



leads to a horizontal thickening of all objects in the image, by combining it with its left-shifted version. An isotropic thickening of all objects is achieved by the operator:



(clearly, this is equivalent to the dilation operator discussed in [Sections 2.2 and 7.2](#)), whereas a symmetrical horizontal thickening operation (see [Section 7.2.3](#)) is achieved by the mask:



A rule of such operations is that if we want to guarantee that all the original object pixels are included in the output image, then we must include a 1 at the center (origin) of the mask.

Finally, there is no compulsion for all masks to be 3×3 . Indeed, all but one of those listed above are effectively smaller than 3×3 , and in more complex cases larger masks could be used. To emphasize this point, and to allow for asymmetrical masks in which the full 3×3 neighborhood is not given, we shall shade the origin—as shown in the above cases.

7.3.2 Generalized Morphological Erosion

We now move on to describe erosion in terms of set operations. The definition is somewhat peculiar in that it involves reverse shifts, but the reason for this

will become clear as we proceed. Here the masks define directions as before, but in this case we shift the image in the reverse of each of these directions and perform intersection operations to combine the resulting images. For masks with a single element (as for the identity and shift left operators in [Section 7.3.1](#)), the intersection operation is improper and the final result is as for the corresponding dilation operator, but with a reverse shift. For more complex cases, the intersection operation results in objects being reduced in size. Thus, the mask:

1	1	

has the effect of stripping away the left sides of objects (the object is moved right and *anded* with itself). Similarly, the mask:

1	1	1
1	1	1
1	1	1

results in an isotropic stripping operation, and is hence identical to the erosion operation described in [Section 7.2.1](#).

7.3.3 Duality Between Dilation and Erosion

We shall write the dilation and erosion operations formally as $A \oplus B$ and $A \ominus B$, respectively, where A is an image and B is the mask of the relevant operation:

$$A \oplus B = \cup_{b \in B} A_b \quad (7.7)$$

$$A \ominus B = \cap_{b \in B} A_{-b} \quad (7.8)$$

In these equations, A_b indicates a basic shift operation in the direction of element b of B and A_{-b} indicates the reverse shift operation.

We next prove an important theorem relating the dilation and erosion operations:

$$(A \ominus B)^c = A^c \oplus B^r \quad (7.9)$$

where A^c represents the complement of A , and B^r represents the reflection of B in its origin. We first note that:²

$$x \in A^c \Leftrightarrow x \notin A \quad (7.10)$$

and

$$b \in B^r \Leftrightarrow -b \in B \quad (7.11)$$

²The sign “ \Leftrightarrow ” means “if and only if,” i.e., the statements so connected are equivalent.

We now have:³

$$\begin{aligned}
 x \in (A \oplus B)^c &\Leftrightarrow x \notin A \oplus B \\
 &\Leftrightarrow \exists b \text{ such that } x \notin A_{-b} \\
 &\Leftrightarrow \exists b \text{ such that } x + b \notin A \\
 &\Leftrightarrow \exists b \text{ such that } x + b \in A^c \\
 &\Leftrightarrow \exists b \text{ such that } x \in (A^c)_{-b} \\
 &\Leftrightarrow x \in \cup_{b \in B} (A^c)_{-b} \\
 &\Leftrightarrow x \in \cup_{b \in B^r} (A^c)_b \\
 &\Leftrightarrow x \in A^c \oplus B^r
 \end{aligned} \tag{7.12}$$

This completes the proof. The related theorem:

$$(A \oplus B)^c = A^c \ominus B^r \tag{7.13}$$

is proved similarly.

The fact that there are two such closely related theorems, following the related union and intersection definitions of dilation and erosion given above, indicates an important duality between the two operations. Indeed, as stated earlier, erosion of the objects in an image corresponds to dilation of the background, and vice versa. However, the two theorems indicate that this relation is not absolutely trivial, on account of the reflections of the masks required in the two cases. It is perhaps curious that in contrast with the case of the de Morgan rule for complementation of an intersection:

$$(P \cap Q)^c = P^c \cup Q^c \tag{7.14}$$

the effective complementation of the dilating or eroding mask is its reflection rather than its complement *per se*, while that for the operator is the alternate operator.

7.3.4 Properties of Dilation and Erosion Operators

Dilation and erosion operators have some very important and useful properties. First, note that successive dilations are associative:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) \tag{7.15}$$

whereas successive erosions are not. In fact, the corresponding relation for erosions is:

$$(A \ominus B) \ominus C = A \ominus (B \oplus C) \tag{7.16}$$

Clearly, the apparent symmetry between the two operators is more subtle than their simple origins in expanding and shrinking might indicate.

Next, the property:

$$X \oplus Y = Y \oplus X \tag{7.17}$$

³This proof is based on that of Haralick et al. (1987). The sign “ \exists ” means “there exists,” and in this context should be interpreted as “there is a value of.” The symbol “ \in ” means “is a member of the following set”: the symbol “ \notin ” means “is *not* a member of the following set.”

means that the order in which dilations of an image are carried out does not matter, and the same applies to the order in which erosions are carried out:

$$(A \oplus B) \oplus C = (A \oplus C) \oplus B \quad (7.18)$$

$$(A \ominus B) \ominus C = (A \ominus C) \ominus B \quad (7.19)$$

In addition to the above relations, which use only the morphological operators \oplus and \ominus , there are many more relations that involve set operations. In the examples that follow, great care must be exercised to note which particular distributive operations are actually valid:

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C) \quad (7.20)$$

$$A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C) \quad (7.21)$$

$$(A \cap B) \ominus C = (A \ominus C) \cap (B \ominus C) \quad (7.22)$$

In certain other cases, where equality might *a priori* have been expected, the strongest statements that can be made are typified by the following:

$$A \ominus (B \cap C) \supseteq (A \ominus B) \cup (A \ominus C) \quad (7.23)$$

Note that the associative relations are of value in showing how large dilations and erosions might be factorized so that they can be implemented more efficiently as two smaller dilations and erosions applied in sequence. Similarly, the distributive relations show that a large mask may be split into two separate masks, which may then be applied separately and the resulting images *ored* together to create the same final image. These approaches can be useful for providing efficient implementations, especially in cases where very large masks are involved. For example, we could dilate an image horizontally and vertically by two separate operations, which would then be merged together—as in the following instance:

$$\begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline 1 & 1 & 1 \\ \hline & & \\ \hline \end{array} \text{ followed by } \begin{array}{|c|c|c|} \hline & 1 & \\ \hline & 1 & \\ \hline & 1 & \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Next, let us consider the importance of the identity operation I , which corresponds to a mask with a single 1 at the central ($A0$) position:

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

By way of example, we take Eqs. (7.20) and (7.21) and replace C by I in each of them. If we write the union of B and I as D , so that mask D is bound to contain a central 1 (i.e., $D \supseteq I$), we have:

$$A \oplus D = A \oplus (B \cup I) = (A \oplus B) \cup (A \oplus I) = (A \oplus B) \cup A \quad (7.24)$$

which always contains A :

$$A \oplus D \supseteq A \quad (7.25)$$

Similarly:

$$A \ominus D = A \ominus (B \cup I) = (A \ominus B) \cap (A \ominus I) = (A \ominus B) \cap A \quad (7.26)$$

which is always contained within A :

$$A \ominus D \subseteq A \quad (7.27)$$

Operations (such as dilation by a mask containing a central 1) which give outputs that are guaranteed to contain the inputs are termed *extensive*, whereas those (such as erosion by a mask containing a central 1) for which the outputs are guaranteed to be contained by the inputs are termed *antiextensive*. Clearly, extensive operations extend objects and antiextensive operations contract them, or in either case, leave them unchanged.

Another important type of operation is the *increasing* type of operation. An increasing operation is one, such as union, which preserves order in the size of the objects on which it operates. If object F is small enough to be contained within object G , then applying erosions or dilations will not affect the situation, even though the objects change their sizes and shapes considerably. We can write these conditions in the form:

$$\text{if } F \subseteq G \quad (7.28)$$

$$\text{then } F \oplus B \subseteq G \oplus B \quad (7.29)$$

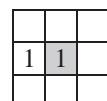
$$\text{and } F \ominus B \subseteq G \ominus B \quad (7.30)$$

Next, we note that erosion can be used for locating the boundaries of objects in binary images:⁴

$$P = A - (A \ominus B) \quad (7.31)$$

There are many practical applications of dilation and erosion, which follow particularly from using them together, as we shall see below.

Finally, we explore why the morphological definition of erosion involves a reflection. The idea is so that dilation and erosion are able, under the right circumstances, to cancel each other out. Take the left-shift dilation operation and the right-shift erosion operation. These are both achieved via the mask:



but in the erosion operation it is applied in its reflected form, thereby producing the right shift required to erode the left edge of any object. This makes it clear why an operation of the type $(A \oplus B) \ominus B$ has a chance of canceling to give A .

⁴Technically, we are here dealing with sets, and the appropriate set operation is the *andnot* function / rather than minus. However, the latter admirably conveys the required meaning without ambiguity.

More specifically, there must be shifts in opposite directions as well as appropriate subtractions produced by *anding* instead or *oring* in order for cancellation to be possible. Of course, in many cases the dilation mask will have 180° rotation symmetry, and then the distinction between B^r and B will be purely academic.

7.3.5 Closing and Opening

Dilation and erosion are basic operators from which many others can be derived. Earlier, we were interested in the possibility of an erosion canceling a dilation and vice versa. Hence, it is an obvious step to define two new operators that express the degree of cancellation: the first is called *closing* since it often has the effect of closing gaps between objects and the other is called *opening* because it often has the effect of opening gaps (Fig. 7.1). Closing (\bullet) and opening (\circ) are formally defined by the formulae:

$$A \bullet B = (A \oplus B) \ominus B \quad (7.32)$$

$$A \circ B = (A \ominus B) \oplus B \quad (7.33)$$

Closing is able to eliminate salt noise, narrow cracks or channels, and small holes or concavities.⁵ Opening is able to eliminate pepper noise, fine hairs, and small protrusions. Thus, these operators are extremely important for practical applications. Furthermore, by subtracting the derived image from the original image, it is possible to locate many sorts of defect, including those cited above as being eliminated by opening and closing: this possibility makes the two operations even more important. For example, we might use the following operation to locate all the fine hairs in an image:

$$Q = A - A \circ B \quad (7.34)$$

This operator and its dual using opening:

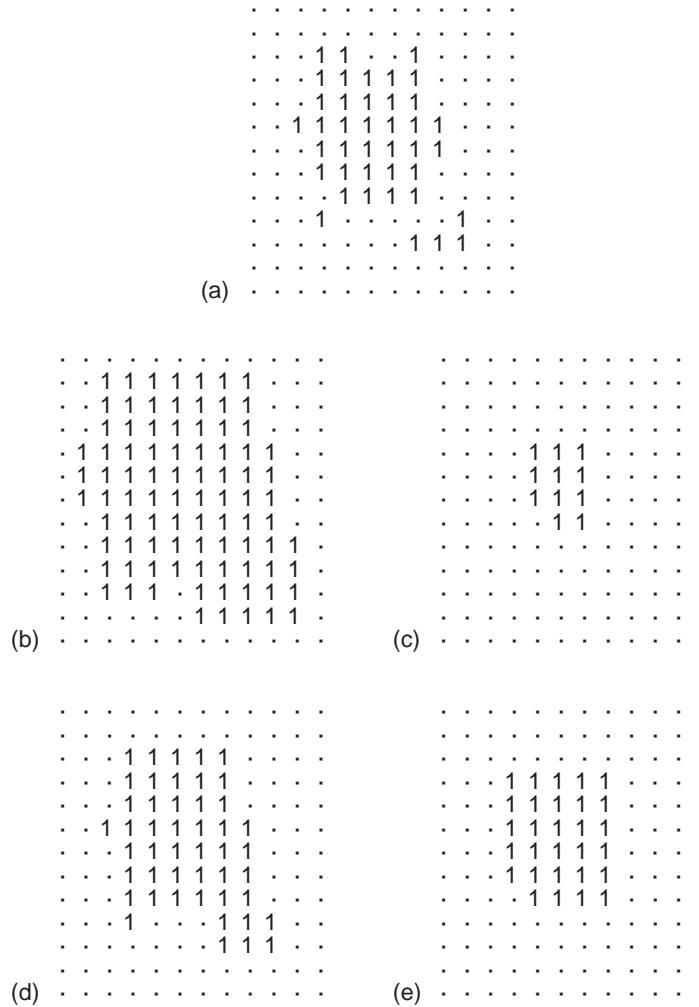
$$R = A \bullet B - A \quad (7.35)$$

are extremely important for defect detection tasks. They are often, respectively, called the white and black top-hat operators.⁶ (Practical applications of these two operators include location of solder bridges and cracks in printed circuit board tracks.)

Closing and opening have the interesting property that they are idempotent: this means that repeated application of either operation has no further effect.

⁵Here we continue to take the convention that dark objects have become 1s in binary images, and light background or other features have become 0s.

⁶It is dubious whether “top hat” is a very appropriate name for this type of operator: *a priori*, the term “residue function” (or simply “residue”) would appear to be better, as it conjures up the right functional connotations.

**FIGURE 7.1**

Results of morphological operations. (a) The original image, (b) the dilated image, (c) the eroded image, (d) the closed image, and (e) the opened image.

Source: © World Scientific 2000

(This property contrasts strongly with what happens when dilation and erosion are applied a number of times.) We can write these results formally as follows:

$$(A \bullet B) \bullet B = A \bullet B \quad (7.36)$$

$$(A \circ B) \circ B = A \circ B \quad (7.37)$$

From a practical point of view these properties are to be expected, since any hole or crack that has been filled in remains filled in, and there is no point in

repeating the operation. Similarly, once a hair or protrusion has been removed, it cannot again be removed without first recreating it. Not quite so obvious is the fact that the combined closing and opening operation is idempotent:

$$\{[(A \bullet B) \circ C] \bullet B\} \circ C = (A \bullet B) \circ C \quad (7.38)$$

The same applies to the combined opening and closing operation. A simpler result is the following:

$$(A \oplus B) \circ B = (A \oplus B) \quad (7.39)$$

which shows that there is no point in opening with the same mask that has already been used for dilation: essentially, the first dilation produces some effects that are not reversed by the erosion (in the opening operation), and the second dilation then merely reverses the effects of the erosion. The dual of this result is also valid:

$$(A \ominus B) \bullet B = (A \ominus B) \quad (7.40)$$

There are a number of other properties of closing and opening; among the most important ones are the following set containment properties, which apply when $D \supseteq I$:

$$A \oplus D \supseteq A \bullet D \supseteq A \quad (7.41)$$

$$A \ominus D \subseteq A \circ D \subseteq A \quad (7.42)$$

Thus, closing an image will, if anything, increase the sizes of objects, while opening an image will, if anything, make objects smaller, although there are clear limits on how much change closing and opening operations can induce.

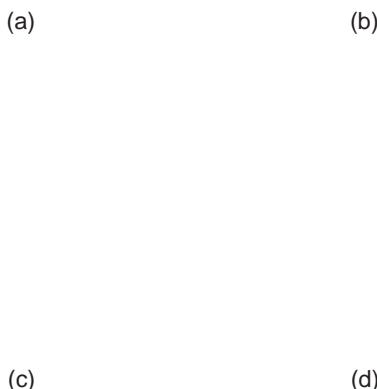
Finally, note that closing and opening are subject to the same duality as for dilation and erosion:

$$(A \bullet B)^c = A^c \circ B^r \quad (7.43)$$

$$(A \circ B)^c = A^c \bullet B^r \quad (7.44)$$

7.3.6 Summary of Basic Morphological Operations

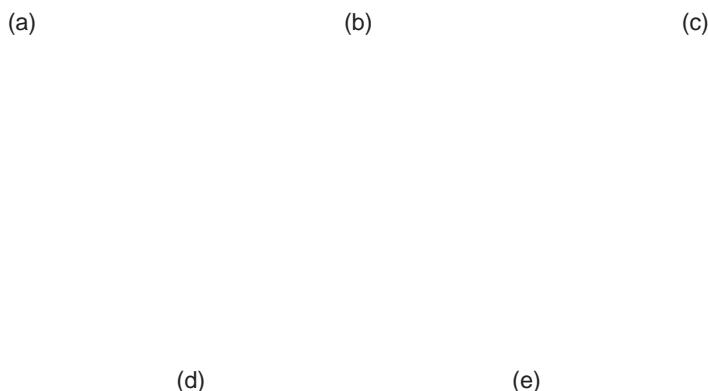
The past few sections have by no means exhausted the properties of the morphological operations, dilate, erode, close, and open. However, these sections have outlined some of their properties and have demonstrated some of the practical results obtained using them. Perhaps the main aim of including the mathematical analysis has been to show that these operations are not *ad hoc* and that their properties are mathematically provable. Furthermore, the analysis has also indicated (a) how sequences of operations can be devised for a number of eventualities and (b) how sequences of operations can be analyzed to save computation (for instance) by taking care not to use idempotent operations repeatedly and by breaking masks down into smaller more efficient ones.

**FIGURE 7.2**

Use of the closing operation. (a) A peppercorn image, (b) the result of thresholding, (c) the result of applying a 3×3 dilation operation to the object shapes, and (d) the effect of subsequently applying a 3×3 erosion operation. The overall effect of the two operations is a “closing” operation. In this case closing is useful for eliminating the small holes in the objects: this would, e.g., be useful for helping to prevent misleading loops from appearing in skeletons. For this picture, extremely large window operations would be required to group peppercorns into regions.

Source: © World Scientific 2000

Overall, the operations devised here can help to eliminate noise and irrelevant artifacts from images so as to obtain more accurate recognition of shapes; they can also help to identify defects on objects by locating specific features of interest. In addition, they can perform grouping functions such as locating regions of images where small objects such as seeds may reside (Section 7.5). In general, elimination of artifacts is carried out by operations such as closing and opening, while location of such features is carried out by finding how the results of these operations differ from the original image (cf. Eqs. (7.20) and (7.21)); and locating regions where clusters of small objects occur may be achieved by larger scale closing operations. Clearly, care in the choice of scales and mask sizes is of vital importance in the design of complete algorithms for all these tasks. Figures 7.2 and 7.3 illustrate some of these possibilities in the case of a peppercorn image: some of the interest in this image relates to the presence of a twiglet and how it is eliminated from consideration and/or identified.

**FIGURE 7.3**

Use of the opening operation. (a) A thresholded peppercorn image. (b) The result of applying a 7×7 erosion operation to the object shapes. (c) The effect of subsequently applying a 7×7 dilation operation. The overall effect of the two operations is an “opening” operation. In this case, opening is useful for eliminating the twiglet. (d) and (e) The same respective operations when applied within an 11×11 window. Here some size filtering of the peppercorns has been achieved and all the peppercorns have been separated—thereby helping with subsequent counting and labeling operations.

Source: © World Scientific 2000

7.4 GRAYSCALE PROCESSING

The generalization of morphology to grayscale images can be achieved in a number of ways. A particularly simple approach is to employ “flat” structuring elements. These perform morphological processing in the same way for each of the gray levels, acting as if the shapes at each level were separate, independent binary images. If dilation is carried out in this way, the result turns out to be identical to the effect of applying a maximum intensity operation of the same shape: i.e., we replace set inclusion by a magnitude comparison; needless to say, this is mathematically identical in action for a normal binary image, but when applied to a grayscale image it neatly generalizes the dilation concept. Similarly, erosion can

be carried out by applying a minimum intensity structuring element of the same shape as the original binary structuring element. This discussion assumes that we focus on light objects against dark backgrounds⁷: these will be dilated when the maximum intensity operation is applied, and eroded when the minimum intensity operation is applied; we could of course reverse the convention, depending on what type of objects we are concentrating on at any moment, or in any application. We can summarize the situation as follows:

$$A \oplus B = \max_{b \in B} A_b \quad (7.45)$$

$$A \ominus B = \min_{b \in B} A_b \quad (7.46)$$

There are other more complex grayscale analogs of dilation and erosion: these take the form of 3-D structuring elements whose output at any gray level depends not only on the shape of the image at that gray level, but also on the shapes at a number of nearby gray levels. Although such “nonflat” structuring elements are useful, for a good many applications they are not necessary, as flat structuring elements already embody a very considerable amount of generalization relative to the binary case.

Next we consider how edge detection is carried out using grayscale morphology.

7.4.1 Morphological Edge Enhancement

In Section 2.2.2, we have shown how edge detection can be carried out in binary images. We have defined edge detection asymmetrically, in the sense that the edge is the part of the object that is next to the background. This is useful because including the part of the background next to the object would merely have served to make the boundary wider and less precise. However, edge detection in grayscale images does not need to embody such an asymmetry,⁸ because it starts by performing edge enhancement and then carrying out a thresholding type of operation—the width being controlled largely by the manner of thresholding and whether nonmaximum suppression or other factors are brought to bear. Here we start by formulating the original binary edge detector in morphological form. Then we make it more symmetric. Finally we generalize it to grayscale operation.

The original binary edge detector may be written in the form (cf. Eq. (7.31)):

$$E = A - (A \ominus B) \quad (7.47)$$

⁷In fact, this is the opposite convention to that employed in Chapter 2, but, as we shall see below, in grayscale processing it is probably more general to focus on intensities rather than on specific objects.

⁸Indeed, any asymmetry would lead to an unnecessary bias and hence inaccuracy in the location of edges.

(a) (b)

FIGURE 7.4

Determination of the morphological gradient of an image. The original image is that of Fig. 7.2(a). (a) The morphological gradient, obtained using 3×3 window operations. (b) The result for a Sobel operator: note that the latter gives less diffuse responses.

Source: © World Scientific 2000

Making it symmetrical merely involves adding the background edge $((A \oplus B) - A)$:

$$G = (A \oplus B) - (A \ominus B) \quad (7.48)$$

To convert to grayscale operation involves employing maximum and minimum operations in place of dilation and erosion. In this case we are concentrating on intensity *per se*, and so these respective assignments of dilation and erosion are used (the alternate arrangement would result in negative edge contrast). Thus, we here use Eqs. (7.45) and (7.46) to *define* dilation and erosion for grayscale processing, so Eq. (7.48) already represents morphological edge enhancement for grayscale images. The argument G is often called the morphological gradient of an image (Fig. 7.4). Note that it is not accompanied by an accurate edge orientation value, although approximate orientations can, of course, be computed by determining which part of the structuring element gives rise to the maximum signal.

7.4.2 Further Remarks on the Generalization to Grayscale Processing

In the previous subsection we found that reinterpreting Eq. (7.48) permitted edge detection to be generalized immediately from binary to gray scale. This is a consequence of the extremely powerful *umbra homomorphism theorem*. This starts with the knowledge that intensity I is a single-valued function of position within the image. This means that it represents a surface in a 3-D (grayscale) space. However, as we have seen, it is useful to take into account the individual gray levels. In particular, we note that the set of pixels of gray level g_i is a subset of the set of pixels of gray level g_{i-1} , where $g_i \geq g_{i-1}$. The important step forward is to interpret the 3-D volume containing all these gray levels under the intensity

surface as constituting an umbra—a 3-D shadow region for the relevant part of the surface. In fact, we write the umbra volume of I as $U(I)$, and clearly we also have $I = T(U(I))$, where the operator $T(\cdot)$ recalculates the top surface.

The umbra homomorphism theorem then states that a dilation has to be defined and interpreted as an operation on the umbras:

$$U(I \oplus K) = U(I) \oplus U(K) \quad (7.49)$$

To find the relevant intensity function we merely need to apply the top-surface operator to the umbra:

$$I \oplus K = T(U(I \oplus K)) = T(U(I) \oplus U(K)) \quad (7.50)$$

A similar statement applies for erosion.

The next step is to note that the generalization from binary to grayscale dilation using flat structuring elements involved applying a maximum operation in place of a set union operation. The operation can, for the simple case of a 1-D image, be rewritten in the form:

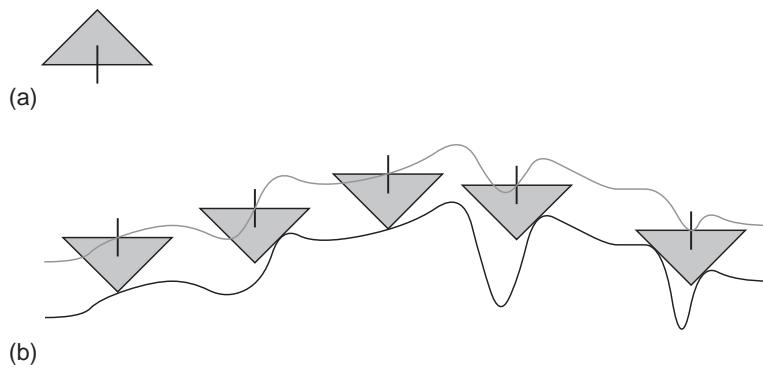
$$(I \oplus K)(x) = \max(I(x - z) + K(z)) \quad (7.51)$$

where $K(z)$ takes value 0 or 1 only, and $x - z$, z must lie within the domains of I and K . However, another vital step is to notice that this form generalizes to nonbinary $K(z)$, whose values can, e.g., be the integer gray-level values. This makes the dilation operation considerably more powerful, yielding the nonflat structuring element concept—which will clearly also work for 2-D images with full gray scale.

It can be tedious working out the responses of this sort of operation, but it is susceptible to a neat geometric interpretation. If the function $K(z)$ is inverted and turned into a template, this may be run over the image $I(x)$ in such a way as to remain just in contact with it. Thus, the origin of the inverted template will trace out the top surface of the dilated image. The process is depicted in Fig. 7.5 for the case of a triangular structuring element in a 1-D image.

Similar relations apply for erosion, closing, opening, and a variety of set functions. This means that the standard binary morphological relations, Eqs. (7.15)–(7.23), apply for grayscale images as well as for binary images. Furthermore, the dilation–erosion and closing–opening dualities (Eqs. (7.9), (7.13), (7.43), and (7.44)) also apply for grayscale images. These are extremely powerful results, and allow one to apply morphological concepts in an intuitive manner. In that case the practically important factor devolves into choosing the right grayscale structuring element for the application.

Another interesting factor is the possibility of using morphological operations instead of convolutions in the many places where the latter are employed throughout image analysis. We have already seen how edge enhancement and detection can be performed using morphology in place of convolution. In addition, noise suppression by Gaussian smoothing can be replaced by opening and closing operations. However, it must always be borne in mind that convolutions are linear operations, and are thereby grossly restricted, whereas morphological operations

**FIGURE 7.5**

Dilation of 1-D grayscale image by triangular structuring element. (a) The structuring element, with the vertical line at the bottom indicating the origin of coordinates. (b) The original image (continuous black line), several instances of the inverted structuring element being applied, and the output image (continuous gray line). This geometric construction automatically takes account of the maximum operation in Eq. (7.51). Note that as no part of the structuring element is below the origin in (a), the output intensity is increased at every point in the image.

are highly nonlinear, their very structure embodying multiple “if” statements, so outward appearances of similarity are bound to hide deep differences of operation, effectiveness, and applicability. Consider, e.g., the optimality of the mean filter for suppressing Gaussian noise and the optimality of the median filter (a morphological operator) for eliminating impulse noise.

One example of this is worth including here: when performing edge enhancement prior to edge detection, it is possible to show that if both a differential gradient (e.g., Sobel) operator and a morphological gradient operator are applied to a noise-free image with a steady intensity gradient, the results will be identical, within a constant factor. However, if one impulse noise pixel arises, the maximum or minimum operations of the morphological gradient operator will select this value in calculating the gradient, whereas the differential gradient operator will average its effect over the window, giving significantly less error.

Space prevents grayscale morphological processing from being considered in more detail here (see, e.g., Haralick and Shapiro (1992) and Soille (2003)).

7.5 EFFECT OF NOISE ON MORPHOLOGICAL GROUPING OPERATIONS

Texture analysis is an important area of machine vision, and is relevant not only for segmenting one region of an image from another (as in many remote sensing

applications), but also for characterizing regions absolutely—as is necessary when performing surface inspection (e.g., when assessing the paint finish on automobiles). Many methods have been employed for texture analysis. These range from the widely used gray-level co-occurrence matrix approach to Law's texture energy approach and from the use of Markov random fields to fractal modeling (Chapter 8).

In fact, there are approaches that involve even less computation and which are applicable when the textures are particularly simple and the shapes of the basic texture elements are not especially critical. For example, if it is required to locate regions containing small objects, simple morphological operations applied to thresholded versions of the image are often appropriate (Fig. 7.6) (Haralick and Shapiro, 1992; Bangham and Marshall, 1998). Such approaches can be used for locating regions containing seeds, grains, nails, sand, or other materials, either for assessing the overall quantity or spread or for determining whether there are regions that have not yet been covered. The basic operation to be applied is the dilation operation, which combines the individual particles into fully connected regions. This method is suitable not only for connecting individual particles but also for separating regions containing high and low densities of such particles. The expansion characteristic of the dilation operation can be largely canceled by a subsequent erosion operation, using the same morphological kernel. Indeed, if the particles are always convex and well separated, the erosion should exactly cancel the dilation, although in general the combined closing operation is not a null operation, and this is relied upon in the above connecting operation.

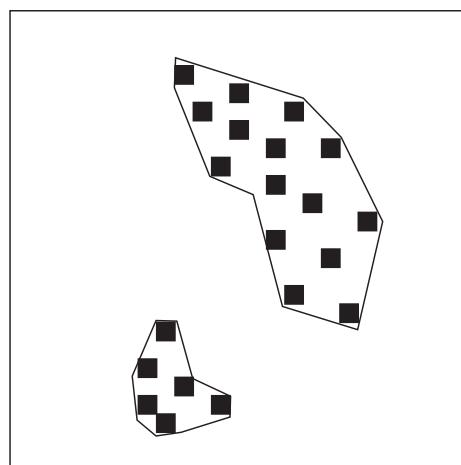


FIGURE 7.6

Idealized grouping of small objects into regions such as might be attempted using closing operations.

Source: © IEE 2000

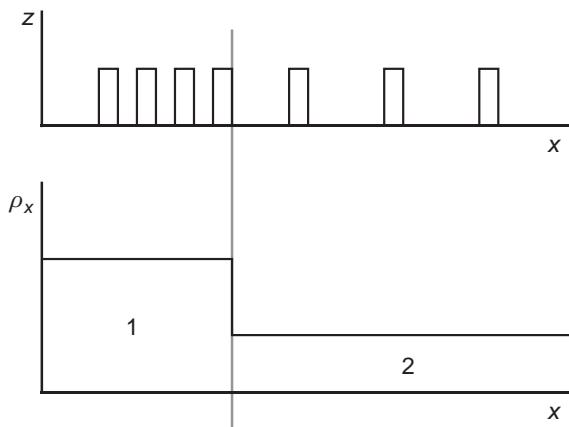
Closing operations have been applied to images of cereal grains containing dark rodent droppings in order to consolidate the droppings (which contain significant speckle—and therefore holes when the images are thresholded) and thus to make them more readily recognizable from their shapes (Davies et al., 1998a). However, the initial result was rather unsatisfactory as dark patches on the grains tend to combine with the dark droppings: this has the effect of distorting the shapes and also makes the objects larger. This problem was partially overcome by a subsequent erosion operation so that the overall procedure is dilate + erode + erode (for further details, see Chapter 21). Originally, adding this final operation seemed to be an *ad hoc* procedure, but on analysis it was found that the size increase actually applies quite generally when segmentation of textures containing different densities of particles is carried out. It is this general effect that we now consider.

7.5.1 Detailed Analysis

Let us take two regions containing small particles with occurrence densities ρ_1 and ρ_2 , where $\rho_1 > \rho_2$. In region 1, the mean distance between particles will be d_1 and in region 2, the mean distance will be d_2 , where $d_1 < d_2$. If we dilate using a kernel of radius a , where $d_1 < 2a < d_2$, this will tend to connect the particles in region 1 but should leave the particles in region 2 separate. To ensure connecting the particles in region 1, we can make $2a$ larger than $\frac{1}{2}(d_1+d_2)$, but this may risk connecting the particles in region 2 (the risk will be reduced when the subsequent erosion operation is taken into account). Selecting an optimum value of a clearly depends not only on the mean distances d_1 and d_2 but also on their distributions. This is not discussed in detail due to space limitation: here we assume that a suitable selection of a is made, and that it is effective. The problem that is tackled next is whether the size of the final regions matches the *a priori* desired segmentation, i.e., whether any size distortion takes place. We start by taking this to be an essentially 1-D problem, which can be modeled as in Fig. 7.7 (in what follows, the 1-D particle densities are given an x suffix).

Suppose first that $\rho_{2x} = 0$. Then in region 2, the initial dilation will be counteracted exactly (in 1-D) by the subsequent erosion. Next take $\rho_{2x} > 0$: when dilation occurs, a number of particles in region 2 will be enveloped, and the erosion process will not exactly reverse the dilation. If a particle in region 2 is within $2a$ of an outermost particle in region 1, it will merge with region 1, and will remain merged when erosion occurs. The probability P that this will happen is the integral over a distance $2a$ of the probability density for particles in region 2. In addition, when the particles are well separated we can take the probability density as being equal to the mean particle density ρ_{2x} . Hence:

$$P = \int_0^{2a} \rho_{2x} dx = 2a\rho_{2x} \quad (7.52)$$

**FIGURE 7.7**

1-D particle distribution. z indicates the presence of a particle, and ρ_x shows the densities in the two regions.

Source: © IEE 2000

If such an event occurs, then region 1 will be expanded by amounts ranging from a to $3a$, or 0 to $2a$ after erosion, although these figures must be increased by b for particles of width b . Thus, the *mean* increase in size of region 1 after dilation + erosion is $2a\rho_{2x} \times (a + b)$, where we have assumed that the particle density in region 2 remains uniform right up to region 1.

Next we consider what additional erosion operation will be necessary to cancel this increase in size. In fact, we just make the radius $\tilde{a}_{1\text{-D}}$ of the erosion kernel equal to the increase in size:

$$\tilde{a}_{1\text{-D}} = 2a\rho_{2x}(a + b) \quad (7.53)$$

Finally, we must recognize that the required process is 2-D rather than 1-D, and take y to be the lateral axis, normal to the original (1-D) x -axis. For simplicity, we assume that the dilated particles in region 2 are separated laterally, and are not touching or overlapping (Fig. 7.8). As a result, the change of size of region 1 given in Eq. (7.53) will be diluted relative to the 1-D case by the reduced density along the direction (y) of the border between the two regions: i.e., we must multiply the right-hand side of Eq. (7.53) by $b\rho_{2y}$. We now obtain the relevant 2-D equation:

$$\tilde{a}_{2\text{-D}} = 2ab\rho_{2x}\rho_{2y}(a + b) = 2ab\rho_2(a + b) \quad (7.54)$$

where we have finally reverted to the appropriate 2-D *area* particle density ρ_2 .

Clearly, for low values of ρ_2 an additional erosion will not be required, whereas for high values of ρ_2 substantial erosion will be necessary, particularly if b is comparable to or larger than a . If $\tilde{a}_{2\text{-D}} < 1$, it will be difficult to provide an accurate correction by applying an erosion operation, and all that can be done is

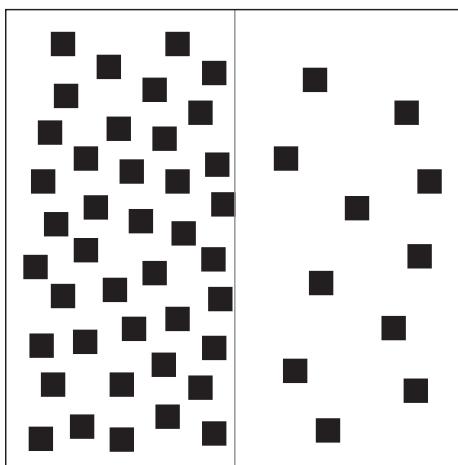


FIGURE 7.8

Model of the incidence of particles in two regions. Region 2 (on the right) has sufficiently low density that the dilated particles will not touch or overlap.

Source: © IEE 2000

to bear in mind that any measurements made from the image will require correction. (Note that if, as often happens, $a \gg 1$, \tilde{a}_{2-D} could well be at least 1.)

7.5.2 Discussion

The work of the author described above (Davies, 2000c) was motivated by analysis of cereal grain images containing rodent droppings, which had to be consolidated by dilation operations to eliminate speckle, followed by erosion operations to restore size.⁹ It has been found that if the background contains a low density of small particles that tend, upon dilation, to increase the sizes of the foreground objects, additional erosion operations will in general be required to accurately represent the sizes of the regions. The effect would be similar if impulse noise were present, although theory shows what is observed in practice—that the effect is enhanced if the particles in the background are not negligible in size. The increases in size are proportional to the occurrence density of the particles in the background, and the kernel for the final erosion operation is calculable, the overall process being a necessary measure rather than an *ad hoc* technique.

7.6 CONCLUDING REMARKS

Binary images contain all the data needed to analyze the shapes, sizes, positions, and orientations of objects in two dimensions, and thereby to recognize them

⁹For further background on this application, see Chapter 21.

and even to inspect them for defects. As we shall see in Chapters 9 and 10, many simple small neighborhood operations exist for processing binary images and moving toward the goals stated earlier. At first sight these may appear a somewhat random set, reflecting historical development rather than systematic analytic tools. However, in the past few years, mathematical morphology has emerged as a unifying theory of shape analysis: we have aimed to give the flavor of the subject in this chapter. In fact, mathematical morphology, as its name suggests, is mathematical in nature, and this can be a source of difficulty,¹⁰ but there are a number of key theorems and results, which are worth remembering: a few of these have been considered here and placed in context. For example, generalized dilation and erosion have acquired a central importance, since further vital concepts and constructs are based on them—closing, opening, template matching, and even connectedness properties (although space has prevented a detailed discussion of its application to the last two topics). For further information on grayscale morphological processing, see Haralick and Shapiro (1992) and Soille (2003).

Mathematical morphology is one of the standard methodologies that has evolved over the past few decades. This chapter has demonstrated how the mathematical aspects make the subject of shape analysis rigorous and less ad hoc. Its extensions to grayscale image processing are interesting and useful, although at this stage they have not ousted more traditional approaches.

7.7 BIBLIOGRAPHICAL AND HISTORICAL NOTES

The book by Serra (1982) was an important early landmark in the development of morphology. Many subsequent papers helped to lay the mathematical foundations, perhaps the most important and influential being that by Haralick et al. (1987); see also Zhuang and Haralick (1986) for methods for decomposing morphological operators, and Crimmins and Brown (1985) for more practical aspects of shape recognition. The papers by Dougherty and Giardina (1988), Heijmans (1991), and Dougherty and Sinha (1995a, 1995b) were important in the development of methods for grayscale morphological processing, while the work of Huang and Mitchell (1994) on grayscale morphology decomposition and that of Jackway and Deriche (1996) on multiscale morphological operators gave further impetus to the subject.

¹⁰The rigor of mathematics is a cause for celebration, but at the same time it can make the arguments and the results obtained from them less intuitive. On the other hand, the real benefit of mathematics is to leapfrog what is possible by intuition alone and to arrive at results that are new and unexpected.

One problem is that it is by no means obvious how to decide on the sequence of morphological operations that is required in any application. This is an area where genetic algorithms have contributed to the systematic generation of complete systems (see, e.g., Harvey and Marshall, 1994).

Work up to 1998 is reviewed in a useful tutorial paper by Bangham and Marshall (1998); more recently Soille (2003) produced a thoroughgoing volume on the subject. Gil and Kimmel (2002) address the problem of rapid implementation of dilation, erosion, opening, and closing algorithms, and arrive at a new approach based on deterministic calculations: these give low computational complexity for calculating max and min functions, and similar complexities for the four cited filters and other derived filters. The paper goes on to state some open problems, and to suggest how they might be tackled: it is clear that some apparently simple tasks that ought perhaps to have been dealt with before the 2000s are still unsolved—such as how to compute the median in better than $O(\log^2 p)$ time per filtered point (in a $p \times p$ window) or how to *optimally* extend 1-D morphological operations to circular rather than square window (2-D) operations. Note that the new implementation is immediately applicable to determining the morphological gradient of an image.

7.7.1 More Recent Developments

More recently, Bai and Zhou (2010) have designed a top-hat selection transformation for locating and enhancing small dim infrared targets typified by aircraft in the sky. The selection transformation is based on the classical top-hat (residue) operator. A necessary parameter in the analysis is the value of n , the minimum difference in intensity between the target and the background, and methods are given for estimating it. Jiang et al. (2007) also use a residue operator to find thin low-contrast edges. The method uses five basic 5×5 masks to detect edges of the right widths. Very high resistance to noise is demonstrated by the particular combination of techniques applied in this approach. Soille and Vogt (2009) show how binary images may be segmented to identify a range of different types of pattern. These include the following mutually exclusive foreground categories: core, islet, connector (loop and bridge), boundary (perforation and edge), branch, and segmented binary pattern. Lézoray and Charrier (2009) describe a new approach to color image segmentation, by analysis of color projections in 2-D histograms to find the dominant colors: the important factor is that clustering in 2-D histograms can proceed very effectively using standard image processing techniques, including morphological processing. Valero et al. (2010) use directional mathematical morphology to detect roads in remote sensing images. The paper starts by taking roads to be linear connected paths; however, curved road segments and other network details can be dealt with using “path openings” and “path closings” in order to obtain the required structural information.

7.8 PROBLEM

1. A morphological gradient binary edge enhancement operator is defined by the formula:

$$G = (A \oplus B) - (A \ominus B)$$

Using a 1-D model of an edge, or otherwise, shows that this will give wide edges in binary images. If grayscale dilation (\oplus) is equated to taking a local *maximum* of the intensity function within a 3×3 window, and grayscale erosion (\ominus) is equated to taking a local *minimum* within a 3×3 window, sketch the result of applying the operator G . Show that it is similar in effect to a Sobel edge enhancement operator, if edge orientation effects are ignored by taking the Sobel magnitude:

$$g = (g_x^2 + g_y^2)^{1/2}$$

Texture

8

It is quite easy to understand what a texture is, although somewhat less easy to define it. For many reasons it is useful to be able to classify textures and to distinguish them from one another; it is also useful to be able to determine the boundaries between different textures, as they often signify the boundaries of real objects. This chapter studies the means for achieving these aims.

Look out for:

- basic measures by which textures can be classified—such as regularity, randomness, and directionality.
- problems that arise with “obvious” texture analysis methods, such as autocorrelation.
- the long-standing graylevel co-occurrence matrix method.
- Laws’ method and Adé’s generalization of it.
- the fact that textures have to be analyzed statistically, because of the random element in their construction.

Texture analysis is a core element in the vision repertoire, just as textures are core components of most images. It therefore seemed most appropriate to include this topic in Part 1 of the book.

8.1 INTRODUCTION

In the foregoing chapters, many aspects of image analysis and recognition have been studied. At the core of these matters has been the concept of segmentation, which involves the splitting of images into regions that have some degree of

uniformity, whether in intensity, color, texture, depth, motion, or other relevant attributes. Care has been taken in Chapter 4 to emphasize that such a process will be largely *ad hoc*, since the boundaries produced will not necessarily correspond to those of real objects. Nevertheless, it is important to make the attempt, either as a preliminary to more accurate or iterative demarcation of objects and their facets, or as an end in itself—e.g., to judge the quality of surfaces.

In this chapter, we move on to the study of texture and its measurement. Texture is a difficult property to define: indeed, in 1979, Haralick reported that no satisfactory definition of it had up till then been produced. Perhaps we should not be surprised by this, as the concept has rather separate meanings in the contexts of vision, touch, and taste: furthermore, the ways in which different people understand the terms are highly individual and subjective. Nevertheless, we require a working definition of texture, and in vision the particular aspect we focus on is the variation in intensity of a particular surface or region of an image. Even with this statement we are being indecisive about whether it is the physical object being observed which is being described or the image derived from it. This reflects the fact that it is the roughness of the surface or the structure or composition of the material that originally gives rise to its visual properties. However, in this chapter, we are mainly interested in the interpretation of images, and so we define texture as the characteristic variation in intensity of a region of an image, which should allow us to recognize and describe it and to outline its boundaries ([Fig. 8.1](#)).

This definition of texture implies that texture is nonexistent in a surface of uniform intensity and does not say anything about how the intensity might be expected to vary or how we might recognize and describe it. In fact, there are many ways in which intensity might vary, but if the variation does not have sufficient uniformity, the texture may not be characterized sufficiently close to permit recognition or segmentation.

We next consider ways in which intensity might vary. Clearly, it can vary rapidly or slowly, markedly or with low contrast, with a high or low degree of directionality, and with greater or lesser degrees of regularity. This last characteristic is often taken as key: either the textural pattern is regular as for a piece of cloth, or it is random as for a sandy beach or a pile of grass cuttings. However, this ignores the fact that a regular textural pattern is often not wholly regular (again, as for a piece of cloth), or not wholly random (as for a mound of potatoes of similar size). Thus, the degrees of randomness and of regularity will have to be measured and compared when characterizing a texture.

There are more profound things to say about the textures described earlier in this section. Often textures are derived from tiny objects or components that are themselves similar, but that are placed together in ways ranging from purely random to purely regular—be they bricks in a wall, grains of sand, blades of grass, strands of material, stripes on a shirt, wickerwork on a basket, or a host of other items. In texture analysis it is useful to have a name for the similar textural

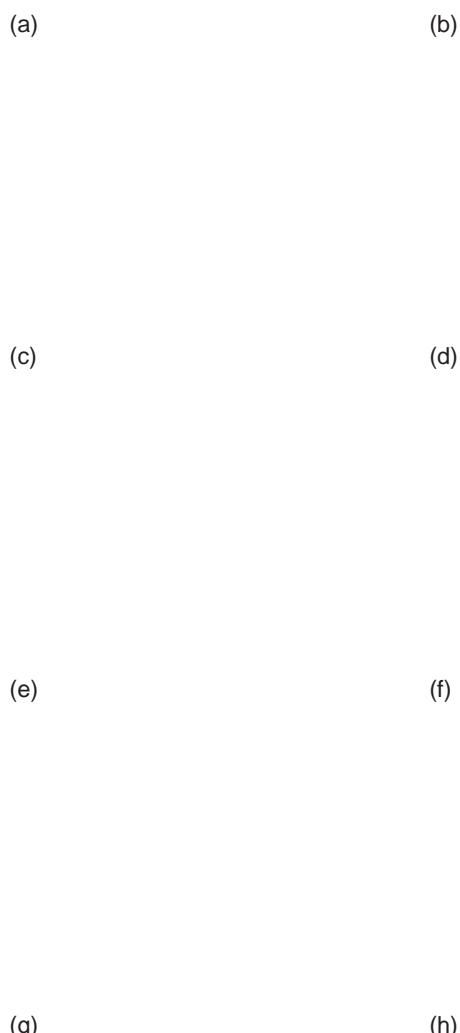


FIGURE 8.1

The variety of textures obtained from real objects. (a) Bark, (b) wood grain, (c) fir leaves, (d) chick peas, (e) carpet, (f) fabric, (g) stone chips, (h) water. These textures demonstrate the wide variety of familiar textures that are easily recognized from their characteristic intensity patterns.

elements that are replicated over a region of the image: such textural elements are called *texels*. These considerations lead us to characterize textures in the following ways:

1. The texels will have various sizes and degrees of uniformity.
2. The texels will be orientated in various directions.
3. The texels will be spaced at varying distances in different directions.
4. The contrast will have various magnitudes and variations.
5. Various amounts of background may be visible between texels.
6. The variations composing the texture may each have varying degrees of regularity *vis-à-vis* randomness.

It is quite clear from this discussion that a texture is a complicated entity to measure. The reason is primarily that many parameters are likely to be required to characterize it: in addition, when so many parameters are involved, it is difficult to disentangle the available data and measure the individual values or decide the ones that are most relevant for recognition. And of course, the statistical nature of many of the parameters is by no means helpful. However, we have so far only attempted to show how complex the situation can be. In what follows, we attempt to show that quite simple measures can be used to recognize and segment textures in practical situations.

Before proceeding, it is useful to recall that in the analysis of shape, there is a dichotomy between available analysis methods. We could, e.g., use a set of measures such as circularity, aspect ratio, and so on, which would permit a description of the shape, but which would not allow it to be reconstructed; or else we could use descriptors such as skeletons with distance function values, or moments, which would permit full and accurate reconstruction—although the set of descriptors might have been curtailed so that only limited but predictable accuracy was available. In principle, such a reconstruction criterion should be possible with texture. However, in practice there are two levels of reconstruction. In the first, we could reproduce a pattern which, to human eyes, would be indistinguishable from the off-camera texture until one compared the two on a pixel-by-pixel basis. In the second, we could reproduce a textured pattern exactly. The point is that textures are normally partially statistical in nature, so it will not be easy to obtain a pixel-by-pixel match in intensities: nor, in general, will it be worth aiming to do so. Thus, texture analysis generally only aims at obtaining accurate statistical descriptions of textures, from which *apparently* identical textures can be reproduced, if desired.

Very many workers have contributed to, and used, a wide range of approaches for texture analysis over a period of 40 years. The sheer weight of the available material and the statistical nature of it can be daunting for many. Note that [Section 8.4](#) is particularly relevant to practitioners because it describes the Laws' texture energy approach which is intuitive, straightforward to apply in both software and hardware, and highly effective in many application areas. However, [Section 8.3](#) on graylevel co-occurrence matrices (which were important historically) can be omitted on a first reading.

8.2 SOME BASIC APPROACHES TO TEXTURE ANALYSIS

In [Section 8.1](#), texture was defined as the characteristic variation in intensity of a region of an image that should allow us to recognize and describe it and to outline its boundaries. In view of the likely statistical nature of textures, this prompts us to characterize texture by the variance in intensity values taken over the whole region of the texture.¹ However, such an approach will not give a rich enough description of the texture for most purposes and will certainly not provide any possibility of reconstruction: it will also be especially unsuitable in cases where the texels are well defined, or where there is a high degree of periodicity in the texture. On the other hand, for highly periodic textures such as that arise with many textiles, it is natural to consider the use of Fourier analysis. Indeed, in the early days of image analysis, this approach was tested thoroughly, although the results were not always encouraging.

Bajcsy (1973) used a variety of ring and orientated strip filters in the Fourier domain to isolate texture features—an approach that was found to work successfully on natural textures such as grass, sand, and trees. However, there is a general difficulty in using the Fourier power spectrum in that the information is more scattered than might be expected at first. In addition, strong edges and image boundary effects can prevent accurate texture analysis by this method. Perhaps more important is the fact that the Fourier approach is a global one that is difficult to apply successfully to an image that is to be segmented by texture analysis (Weszka et al., 1976).

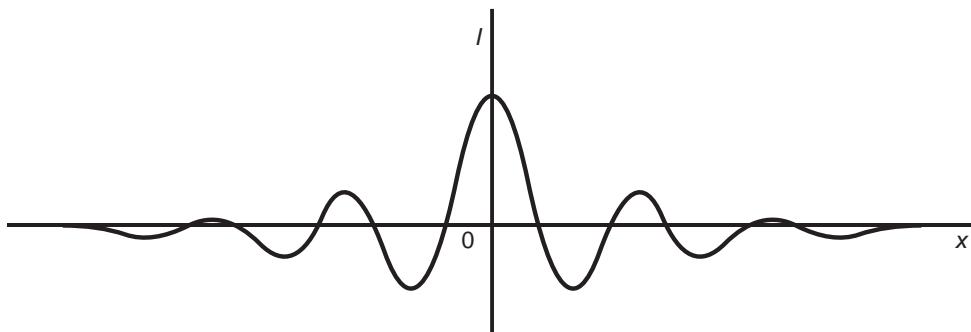
Autocorrelation is another obvious approach to texture analysis, since it should show up both local intensity variations and also the repeatability of the texture (see [Fig. 8.2](#)). An early study was carried out by Kaizer (1955). He examined how many pixels an image has to shift before the autocorrelation function drops to $1/e$ of its initial value and produced a subjective measure of coarseness on this basis. However, Rosenfeld and Troy (1970a, 1970b) later showed that autocorrelation is not a satisfactory measure of coarseness. In addition, autocorrelation is not a very good discriminator of isotropy in natural textures. Hence, workers were quick to take up the co-occurrence matrix approach introduced by Haralick et al. (1973): in fact, this approach not only replaced the use of autocorrelation but during the 1970s also became to a large degree the “standard” approach to texture analysis.

8.3 GRAYLEVEL CO-OCCURRENCE MATRICES

The graylevel co-occurrence matrix approach² is based on studies of the statistics of pixel intensity distributions. As hinted above with regard to the variance in

¹We defer for now the problem of finding the region of a texture so that we can compute its characteristics in order to perform a segmentation function. However, some preliminary training of a classifier may clearly be used to overcome this problem for supervised texture segmentation tasks.

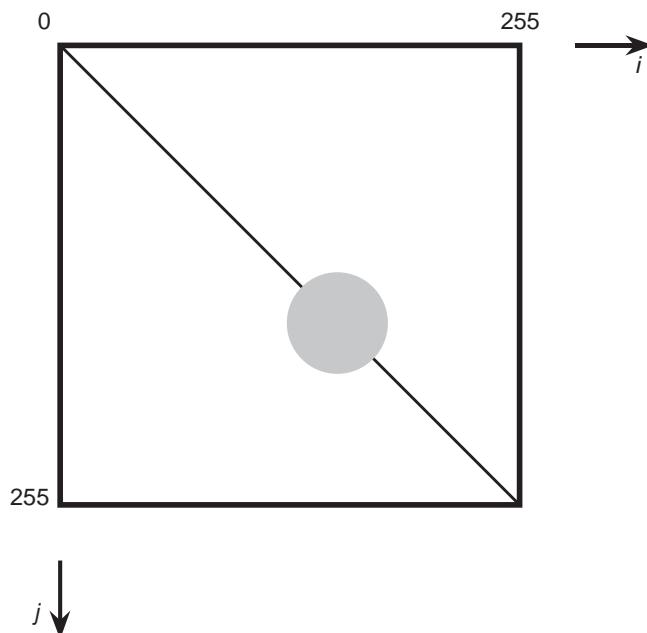
²This is also frequently called the spatial graylevel dependence matrix (SGLDM) approach.

**FIGURE 8.2**

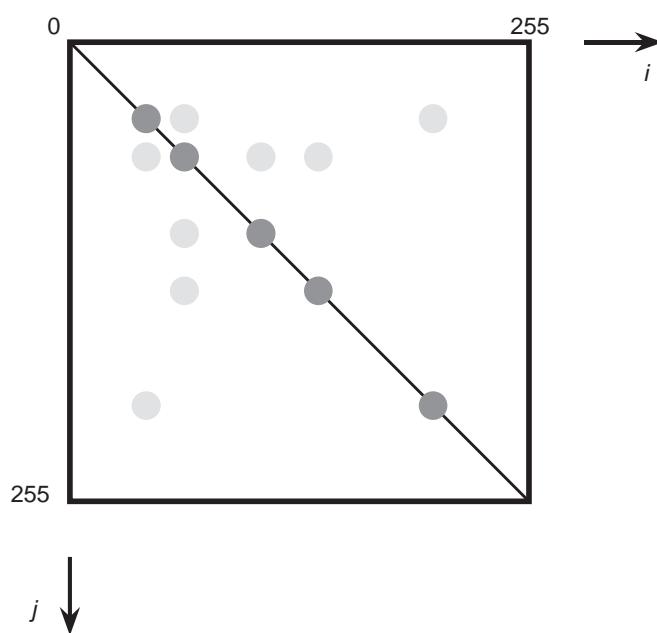
Use of autocorrelation function for texture analysis. This diagram shows the possible 1-D profile of the autocorrelation function for a piece of material in which the weave is subject to significant spatial variation: note that the periodicity of the autocorrelation function is damped down over quite a short distance.

pixel intensity values, single-pixel statistics do not provide rich enough descriptions of textures for practical applications. Thus, it is natural to consider second-order statistics obtained by considering *pairs* of pixels in certain spatial relations to each other. Hence, co-occurrence matrices are used, which express the relative frequencies (or probabilities) $P(i, j|d, \theta)$ with which two pixels having relative polar coordinates (d, θ) appear with intensities i, j . The co-occurrence matrices provide raw numerical data on the texture, although this data must be condensed to relatively few numbers before it can be used to classify the texture. The early paper by Haralick et al. (1973) gave 14 such measures, and these were used successfully for classification of many types of material (including, e.g., wood, corn, grass, and water). However, Connors and Harlow (1980a) found that only five of these measures were normally used, *viz.* “energy,” “entropy,” “correlation,” “local homogeneity,” and “inertia” (note that these names do not provide much indication of the modes of operation of the respective operators).

To obtain a more detailed idea of the operation of the technique, consider the co-occurrence matrix shown in Fig. 8.3. This corresponds to a nearly uniform image containing a single region in which the pixel intensities are subject to an approximately Gaussian noise distribution, the attention being on pairs of pixels at a constant vector distance $\mathbf{d} = (d, \theta)$ from each other. Next, consider the co-occurrence matrix shown in Fig. 8.4, which corresponds to an almost noiseless image with several nearly uniform image regions. In this case, the two pixels in each pair may correspond either to the same image regions or to different ones, although if d is small they will only correspond to adjacent image regions. Thus, we have a set of N on-diagonal patches in the co-occurrence matrix, but only a limited number L of the possible number M of off-diagonal patches linking them, where $M = {}^N C_2$ and $L \leq M$ (typically L will be of order N rather than N^2). With textured images, if the texture is not too strong, it may be modeled as noise, and

**FIGURE 8.3**

Co-occurrence matrix for a nearly uniform grayscale image with superimposed Gaussian noise. Here the intensity variation is taken to be almost continuous: normal convention is followed by making the j index increase downward, as for a table of discrete values (cf. Fig. 8.4).

**FIGURE 8.4**

Co-occurrence matrix for an image with several distinct regions of nearly constant intensity. Again, the leading diagonal of the diagram is from top left to bottom right (cf. Figs. 8.2 and 8.5).

the $N + L$ patches in the image will be larger but still not overlapping. However, in more complex cases, the possibility of segmentation using the co-occurrence matrices will depend on the extent to which \mathbf{d} can be chosen to prevent the patches from overlapping. Since many textures are directional, careful choice of θ will clearly help with this task, although the optimum value of d will depend on several other characteristics of the texture.

As a further illustration, we consider the small image shown in Fig. 8.5(a). To produce the co-occurrence matrices for a given value of \mathbf{d} , we merely need to calculate the numbers of cases for which pixels, a distance \mathbf{d} apart, have intensity values i and j . Here, we content ourselves with the two cases $\mathbf{d} = (1, 0)$ and $\mathbf{d} = (1, \pi/2)$. We thus obtain the matrices shown in Fig. 8.5(b) and (c).

This simple example demonstrates that the amount of data in the matrices is liable to be many times more than in the original image—a situation which is exacerbated in more complex cases by the number of values of d and θ that are required to accurately represent the texture. In addition, the number of gray levels will normally be closer to 256 than to 6, and the amount of matrix data varies as the square of this number. Finally, we should notice that the co-occurrence

	0	0	0	1
1	1	1	1	
2	2	2	2	3
3	3	3	4	5

(a)

	0	1	2	3	4	5
0	2	1	0	0	0	0
1	1	3	0	0	0	0
2	0	0	2	1	0	0
3	0	0	1	1	1	0
4	0	0	0	1	0	1
5	0	0	0	0	1	0

(b)

	0	1	2	3	4	5
0	0	3	0	0	0	0
1	3	1	3	1	0	0
2	0	3	0	2	1	0
3	0	1	2	0	0	1
4	0	0	1	0	0	0
5	0	0	0	1	0	0

(c)

FIGURE 8.5

Co-occurrence matrices for a small image. (a) The original image; (b) the resulting co-occurrence matrix for $\mathbf{d} = (1, 0)$; and (c) the matrix for $\mathbf{d} = (1, \pi/2)$. Note that even in this simple case, the matrices contain more data than the original image.

matrices merely provide a new representation: they do not themselves solve the recognition problem.

These factors mean that the gray scale has to be compressed into a much smaller set of values and careful choice of specific sample d , θ values must be made: in most cases, it is not at all obvious how such a choice should be made, and it is even more difficult to arrange for it to be made automatically. In addition, various functions of the matrix data must be tested before the texture can be properly characterized and classified.

These problems with the co-occurrence matrix approach have been tackled in many ways: just two are mentioned here. The first is to ignore the distinction between opposite directions in the image, thereby reducing storage by 50%. The second is to work with *differences* between gray levels; this amounts to performing a summation in the co-occurrence matrices along axes parallel to the main diagonal of the matrix. The result is a set of *first-order difference* statistics. While these modifications have given some additional impetus to the approach, the 1980s saw a highly significant diversification of methods for the analysis of textures. Of these, Laws' approach (1979, 1980a, 1980b) is important in that it has led to other developments which provide a systematic, adaptive means of tackling texture analysis. This approach is covered in [Section 8.4](#).

8.4 LAWS' TEXTURE ENERGY APPROACH

In 1979 and 1980 Laws presented his novel texture energy approach to texture analysis (Laws, 1979, 1980a, 1980b). This involved the application of simple filters to digital images. The basic filters he used were common Gaussian, edge detector, and Laplacian-type filters, and were designed to highlight points of high "texture energy" in the image. By identifying these high energy points, smoothing the various filtered images, and pooling the information from them, he was able to characterize textures highly efficiently. As remarked earlier, Laws' approach has strongly influenced much subsequent work and it is therefore worth considering it here in some detail.

The Laws' masks are constructed by convolving together just three basic 1×3 masks:

$$L3 = [1 \ 2 \ 1] \quad (8.1)$$

$$E3 = [-1 \ 0 \ 1] \quad (8.2)$$

$$S3 = [-1 \ 2 \ -1] \quad (8.3)$$

The initial letters of these masks indicate *Local averaging*, *Edge detection*, and *Spot detection*. In fact, these basic masks span the entire 1×3 subspace and form

Table 8.1 The Nine 3×3 Laws Masks

$L3^T L3$	$L3^T E3$	$L3^T S3$
1 2 1 2 4 2 1 2 1	-1 0 1 -2 0 2 -1 0 1	-1 2 -1 -2 4 -2 -1 2 -1
$E3^T L3$	$E3^T E3$	$E3^T S3$
-1 -2 -1 0 0 0 1 2 1	1 0 -1 0 0 0 -1 0 1	1 -2 1 0 0 0 -1 2 -1
$S3^T L3$	$S3^T E3$	$S3^T S3$
-1 -2 -1 2 4 2 -1 -2 -1	1 0 -1 -2 0 2 1 0 -1	1 -2 1 -2 4 -2 1 -2 1

a complete set. Similarly, the 1×5 masks obtained by convolving pairs of these 1×3 masks together form a complete set:³

$$L5 = [1 \ 4 \ 6 \ 4 \ 1] \quad (8.4)$$

$$E5 = [-1 \ -2 \ 0 \ 2 \ 1] \quad (8.5)$$

$$S5 = [-1 \ 0 \ 2 \ 0 \ -1] \quad (8.6)$$

$$R5 = [1 \ -4 \ 6 \ -4 \ 1] \quad (8.7)$$

$$W5 = [-1 \ 2 \ 0 \ -2 \ 1] \quad (8.8)$$

In Eqs. (8.7) and (8.8), the initial letters indicate Ripple detection and Wave detection. We can also use matrix multiplication (see also Section 3.6) to combine the 1×3 and a similar set of 3×1 masks to obtain nine 3×3 masks—e.g.:

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix} \quad (8.9)$$

The resulting set of masks also forms a complete set (Table 8.1): note that two of these masks are identical to the Sobel operator masks. The corresponding 5×5 masks are entirely similar but are not considered in detail here as all relevant principles are illustrated by the 3×3 masks.

All such sets of masks include one whose components do not average to zero. Thus, it is less useful for texture analysis since it will give results dependent more on image intensity than on texture. The remainder are sensitive to edge points, spots, lines, and combinations of these.

³In principle nine masks can be formed in this way, but only five of them are distinct.

Having produced images that indicate, e.g., local edginess, the next stage is to deduce the local magnitudes of these quantities. These magnitudes are then smoothed over a fair-sized region rather greater than the basic filter mask size (e.g., Laws used a 15×15 smoothing window after applying his 3×3 masks): the effect of this is to smooth over the gaps between the texture edges and other microfeatures. At this point the image has been transformed into a vector image, each component of which represents energy of a different type. Although Laws (1980b) used both squared magnitudes and absolute magnitudes to estimate texture energy, the former correspond to true energy and give a better response, while the latter are useful in requiring less computation:

$$E(l, m) = \sum_{i=l-p}^{l+p} \sum_{j=m-p}^{m+p} |F(i, j)| \quad (8.10)$$

where $F(i, j)$ is the local magnitude of a typical microfeature, which is smoothed at a general scan position (l, m) in a $(2p + 1) \times (2p + 1)$ window.

A further stage is required to combine the various energies in a number of different ways, providing several outputs that can be fed into a classifier to decide upon the particular type of texture at each pixel location (Fig. 8.6). If necessary, principal components analysis is used at this point to help select a suitable set of intermediate outputs.

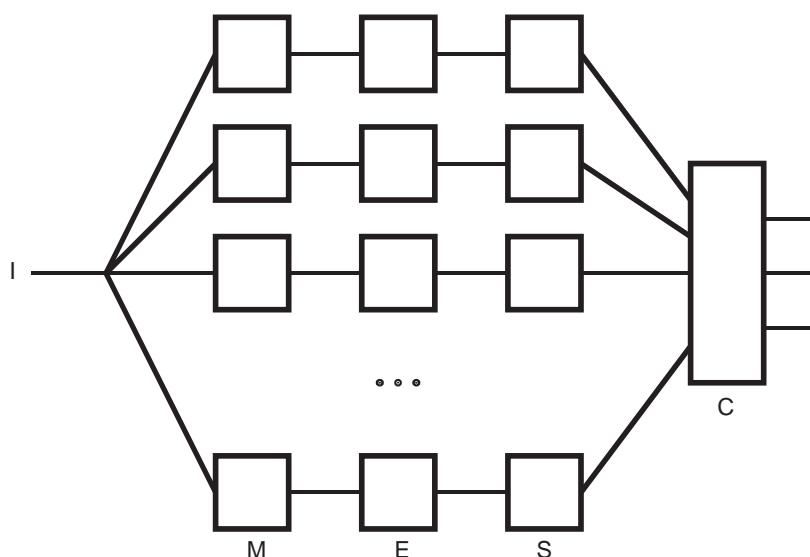


FIGURE 8.6

Basic form for a Laws' texture classifier. Here, I is the incoming image, M represents the microfeature calculation, E the energy calculation, S the smoothing, and C the final classification.

Laws' method resulted in excellent classification accuracy quoted at, e.g., 87% compared with 72% for the co-occurrence matrix method, when applied to a composite texture image of grass, raffia, sand, wool, pigskin, leather, water, and wood (Laws, 1980b). He also found that the histogram equalization normally applied to images to eliminate first-order differences in texture field grayscale distributions gave little improvement in this case.

Research was undertaken by Pietikäinen et al. (1983) to determine whether the precise coefficients used in the Laws' masks are responsible for the performance of his method. They found that so long as the general forms of the masks were retained, performance did not deteriorate, and could in some instances be improved. They were able to confirm that Laws' texture energy measures are more powerful than measures based on pairs of pixels (i.e., co-occurrence matrices).

8.5 ADE'S EIGENFILTER APPROACH

In 1983, Ade investigated the theory underlying the Laws' approach and developed a revised rationale in terms of eigenfilters.⁴ He took all possible pairs of pixels within a 3×3 window, and characterized the image intensity data by a 9×9 covariance matrix. He then determined the eigenvectors required to diagonalize this matrix. These correspond to filter masks similar to the Laws' masks, i.e., use of these "eigenfilter" masks produces images that are principal component images for the given texture. Furthermore, each eigenvalue gives the part of the variance of the original image that can be extracted by the corresponding filter. Essentially, the variances give an exhaustive description of a given texture in terms of the texture of the images from which the covariance matrix was originally derived. Clearly, the filters that give rise to low variances can be taken to be relatively unimportant for texture recognition.

It will be useful to illustrate the technique for a 3×3 window. Here, we follow Ade (1983) in numbering the pixels within a 3×3 window in scan order:

1	2	3
4	5	6
7	8	9

This leads to a 9×9 covariance matrix for describing relationships between pixel intensities within a 3×3 window, as stated above. At this point, we recall that we are describing a texture and assuming that its properties are not synchronous with the pixel tessellation, we would expect various coefficients of the covariance matrix \mathbf{C} to be equal: e.g., C_{24} should equal C_{57} ; in addition, C_{57} must equal C_{75} .

⁴Before reading further, the reader may find it helpful to refer to Section 24.10, where principal components analysis and eigenvalue problems are discussed.

Table 8.2 Spatial Relationships Between Pixels in a 3×3 Window

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>	<i>m</i>
9	6	6	4	4	3	3	1	1	2	2	2	2

This table shows the number of occurrences of the spatial relationships between pixels in a 3×3 window. Note that *a* is the diagonal element of the covariance matrix \mathbf{C} , and that all others appear twice as many times in \mathbf{C} as indicated in the table.

It is worth pursuing this matter, as a reduced number of parameters will lead to increased accuracy in determining the remaining ones. In fact, there are $9 C_2 = 36$ ways of selecting pairs of pixels, but there are only 12 distinct spatial relationships between pixels if we disregard translations of whole pairs—or 13 if we include the null vector in the set (see Table 8.2). Thus, the covariance matrix (see Section 24.10), whose components include the 13 parameters *a*–*m*, takes the form:

$$\mathbf{C} = \begin{bmatrix} a & b & f & c & d & k & g & m & h \\ b & a & b & e & c & d & l & g & m \\ f & b & a & j & e & c & i & l & g \\ c & e & j & a & b & f & c & d & k \\ d & c & e & b & a & b & e & c & d \\ k & d & c & f & b & a & j & e & c \\ g & l & i & c & e & j & a & b & f \\ m & g & l & d & c & e & b & a & b \\ h & m & g & k & d & c & f & b & a \end{bmatrix} \quad (8.11)$$

\mathbf{C} is symmetric, and the eigenvalues of a real symmetric covariance matrix are real and positive, and the eigenvectors are mutually orthogonal (see Section 24.10). In addition, the eigenfilters thus produced reflect the proper structure of the texture being studied and are ideally suited to characterizing it. For example, for a texture with a prominent highly directional pattern, there will be one or more high energy eigenvalues with eigenfilters having strong directionality in the corresponding direction.

8.6 APPRAISAL OF THE LAWS AND ADE APPROACHES

At this point, it will be worthwhile to compare the Laws and Ade approaches more carefully. In the Laws approach, standard filters are used, texture energy images are produced, and then principal component analysis may be applied to lead to recognition, whereas in the Ade approach, special filters (the eigenfilters) are applied, incorporating the results of principal component analysis, following

which texture energy measures are calculated, and a suitable number of these are applied for recognition.

The Ade approach is superior to the extent that it permits low-value energy components to be eliminated early on, thereby saving computation. For example, in Ade's application, the first five of the nine components contain 99.1% of the total texture energy, so the remainder can be ignored. In addition, it would appear that another two of the components containing respectively 1.9% and 0.7% of the energy could also be ignored, with little loss of recognition accuracy. However, in some applications textures could vary continually, and it may well be inadvisable to fine-tune a method to the particular data pertaining at any one time.⁵

In 1986, Unser developed a more general version of the Ade technique that also covered the methods of Faugeras (1978), Granlund (1980), and Wermser and Liedtke (1982). In this approach, performance is optimized not only for texture classification, but also for discrimination between two textures by simultaneous diagonalization of two covariance matrices. The method was developed further by Unser and Eden (1989, 1990): this work makes a careful analysis of the use of nonlinear detectors. As a result, two levels of nonlinearity are employed, one immediately after the linear filters are designed (by employing a specific Gaussian texture model) to feed the smoothing stage with genuine variance or other suitable measures, and the other after the spatial smoothing stage to counteract the effect of the earlier filter and aiming to provide a feature value that is in the same units as the input signal. In practical terms, this means having the capability for providing an RMS texture signal from each of the linear filter channels.

Overall, the originally intuitive Laws approach emerged during the 1980s as a serious alternative to the co-occurrence matrix approach. It is as well to note that alternative methods that are potentially superior have also been devised—see e.g., the local rank correlation method of Harwood et al. (1985), and the forced-choice method of Vistnes (1989) for finding edges between different textures, which apparently has considerably better accuracy than the Laws approach. Vistnes's (1989) investigation concludes that the Laws approach is limited by (a) the small scale of the masks that can miss larger-scale textural structures and (b) the fact that the texture energy smoothing operation blurs the texture feature values across the edge. The latter finding (or even the worse situation where a third class of texture appears to be located in the region of the border between two textures) has also been noted by Hsiao and Sawchuk (1989, 1990) who applied an improved technique for feature smoothing; they also used probabilistic relaxation for enforcing spatial organization on the resulting data.

⁵For example, these remarks apply (1) to textiles, for which the degree of stretch will vary continuously during manufacture, (2) to raw food products, such as beans, whose sizes will vary with the source of supply, and (3) to processed food products, such as cakes, for which the crumbliness will vary with cooking temperature and water vapor content.

8.7 CONCLUDING REMARKS

In this chapter, we have seen the difficulties of analyzing textures: these arise from the potential, and in many cases the frighteningly real complexities of textures—not least from the fact that their properties are often largely statistical in nature. The erstwhile widely used grayscale co-occurrence matrix approach has been seen to have distinct computational shortcomings. First, many co-occurrence matrices are in principle required (with different values of d and θ) in order to adequately describe a given texture; second, the co-occurrence matrices can be very large and, paradoxically, may hold more data than the images they are characterizing—especially if the range of grayscale values is large. In addition, many sets of co-occurrence matrices may be needed to allow for variation of the texture over the image, and if necessary, to initiate segmentation. Hence, co-occurrence matrices need to be significantly compressed, although in most cases it is not at all obvious *a priori* how this should be achieved, and it is even more difficult to arrange for it to be carried out automatically. This probably explains why attention shifted during the 1980s to other approaches, including particularly Laws' technique and its variations (especially that of Ade). Other developments were fractal-based measures, Markov approaches and the Gabor filter technique, although space has prevented a discussion of these methods here: see [Section 8.8](#) for further reading on these topics.

Textures are recognized and segmented by humans with the same apparent ease as for plain objects. This chapter has shown that texture analysis needs to be sensitive to microstructures and then pulled into macrostructures—with PCA (principal components analysis) being a natural means of finding the optimum structure. The subject has great importance for new applications, such as iris recognition.

8.8 BIBLIOGRAPHICAL AND HISTORICAL NOTES

Early work on texture analysis was carried out by Haralick et al. (1973) and in 1976, Weska and Rosenfeld applied textural analysis to materials' inspection. The area was reviewed by Zucker (1976a) and Haralick (1979), and excellent accounts appear in the books by Ballard and Brown (1982) and Levine (1985).

At the end of the 1970s, the Laws technique (1979, 1980a, 1980b) arrived upon the scene (which had up till then been dominated by the co-occurrence matrix approach) and led to the principal components approach of Ade (1983), which was further developed by Dewaele et al. (1988), Unser and Eden (1989, 1990), and others. The direction taken by Laws was particularly valuable as it showed how texture analysis could be implemented straightforwardly and in a manner consistent with real-time applications such as inspection.

The 1980s also saw other new developments, such as the fractal approach led by Pentland (1984), and a great amount of work on Markov random field models of texture. Here the work of Hansen and Elliott (1982) was very formative, although the names G.R. Cross, H. Derin, D. Geman, S. Geman, and A.K. Jain come up repeatedly in this context. Bajcsy and Liebermann (1976), Witkin (1981), and Kender (1983) pioneered the *shape from texture* concept, which has received considerable attention ever since. Later, much work appeared on the application of neural networks to texture analysis, e.g., Greenhill and Davies (1993) and Patel et al. (1994). A number of reviews and useful comparative studies have been made including Van Gool et al. (1985), du Buf et al. (1990), Ohanian and Dubes (1992), and Reed and du Buf (1993). For further work on texture analysis related to inspection of faults and foreign objects, see Chapter 20.

More recent developments include further work with automated visual inspection in mind (Davies, 2000c; Tsai and Huang, 2003; Ojala et al., 2002; Manthalkar et al., 2003; Pun and Lee, 2003), although several of these papers also cite medical, remote sensing, and other applications. Of these papers, the last three are specifically aimed at rotation invariant texture classification and the last one also aims at scale invariance. In previous years, there has not been quite enough emphasis on rotation invariance, although it was by no means a new topic. Other work (Clerc and Mallat, 2002) was concerned with recovering shape from texture *via* a texture gradient equation, while Ma et al. (2003) were particularly concerned with person identification based on iris textures. Mirmehdi and Petrou (2000) described an in-depth investigation of color texture segmentation. In this context, the importance of “wavelets”⁶ as an increasingly used technique of texture analysis with interesting applications (such as, human iris recognition) should be noted (e.g., Daugman, 1993, 2003).

Then, in a particularly exciting advance, Spence et al. (2004) managed to eliminate texture by using photometric stereo to find the underlying surface shape (or “bump map”), following which they were able to perform impressive reconstructions, including texture, from a variety of viewpoints; McGunnigle and Chantler (2003) have shown that this sort of technique is also able to reveal hidden writing on textured surfaces, where only pen pressure marks have been made. Similarly, Pan et al. (2004) have shown how texture can be eliminated from ancient tablets (in particular those made of lead and wood) to reveal clear images of the writing underneath.

8.8.1 More Recent Developments

Over the 2000s, the trend to scale and rotation invariant texture analysis mentioned above has continued, the paper by Janney and Geers (2010) describing an

⁶Wavelets are directional filters reminiscent of the Laws edges, bars, waves, and ripples, but have more rigorously defined shapes and envelopes, and are defined in multiresolution sets (Mallat, 1989).

“invariant features of local textures” approach, using a strictly circular 1-D array of sampling, positions around any given position. The method employs Haar wavelets and as a result is computationally efficient. It is applied at multiple scales in order to achieve scale invariance; in addition, intensity normalization is used to make the method illumination as well as scale and rotation invariant.

Two new books have recently been published on this rather specialist subject—by Petrou and Sevilla (2006) and Mirmehdi et al. (2008). The first is a very sound textbook, starting from a low level and progressing through topics not covered in the present volume, such as fractals, Markov random fields, Gibbs distributions, Gabor functions, wavelets, and the Wigner distribution. The second is an edited volume containing chapters by various researchers and providing much new information—as indicated by some of the more novel chapter titles: “TEXEMS: random texture representation and analysis,” “3-D texture analysis,” “Texture for appearance models,” “From dynamic texture to dynamic shape and appearance models,” “Divide-and-texture: Hierarchical feature description,” “Practical implementation of the trace transform,” and “Face analysis using local binary patterns.”

Binary Shape Analysis

9

While binary images contain much less information than their grayscale counterparts, they embody shape and size information that is highly relevant for object recognition. However, this information resides in a digital lattice of pixels, and this results in intricacies appearing in the geometry. This chapter resolves these problems and explores a number of important algorithms for processing shapes.

Look out for:

- the connectedness paradox and how it is resolved.
- object labeling and how labeling conflicts are resolved.
- problems related to measurement in binary images.
- size filtering techniques.
- the convex hull as a means of characterizing shape, and methods for determining it.
- distance functions and how they are obtained using parallel and sequential algorithms.
- the skeleton and how it is found by thinning; the crucial role played by the crossing number, both in determining the skeleton and in analyzing it.
- simple measures for shape recognition, including circularity, and aspect ratio.
- more rigorous measures of shape, including moments and boundary descriptors.

In reality, this chapter almost exclusively covers area-based methods of shape analysis, leaving boundary-based procedures to Chapter 10—though circularity measures and boundary tracking are both covered. However, chapter boundaries cannot be completely exclusive, as any method requires “hooks” that have been laid down in a variety of places, and indeed, it is often valuable to meet a concept before finding out in detail how to put flesh on it.

Returning to the present chapter, it is interesting to note how intricate some of the algorithmic processes are: connectedness, in particular, pervades the whole subject of digital shape analysis and comes with a serious health warning.

9.1 INTRODUCTION

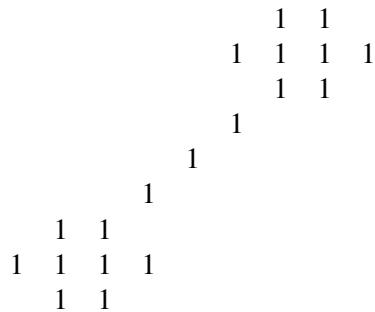
Over the past few decades 2-D shape analysis has provided the main means by which objects are recognized and located in digital images. Fundamentally, 2-D shape has been important because it uniquely characterizes many types of object, from keys to characters, from gaskets to spanners, and from fingerprints to chromosomes, while in addition it can be represented by patterns in simple binary images. Chapter 1 showed how the template matching approach leads to a combinatorial explosion even when fairly basic patterns are to be found, so preliminary analysis of images to find features constitutes a crucial stage in the process of efficient recognition and location. Thus, the capability for binary shape analysis is a very basic requirement for practical visual recognition systems.

In fact, 40 years of progress have provided an enormous range of shape analysis techniques and a correspondingly large range of applications. Clearly, it will be impossible to cover the whole field within the confines of a single chapter—so completeness will not even be attempted (the alternative of a catalog of algorithms and methods, all of which are covered only in brief outline, is eschewed). At one level, the main topics covered are examples with their own intrinsic interest and practical application, and at another level they introduce matters of fundamental principle. Recurring themes are the central importance of connectedness for binary images; the contrasts between local and global operations on images and between different representations of image data; the need to optimize accuracy and computational efficiency; and the compatibility of algorithms and hardware. The chapter starts with a discussion of how connectedness is measured in binary images.

9.2 CONNECTEDNESS IN BINARY IMAGES

This section begins with the assumption that objects have been segmented, by thresholding or other procedures, into sets of 1's in a background of 0's (see Chapters 2–4). At this stage it is important to realize that a second assumption is already being made implicitly—that it is easy to demarcate the boundaries between objects in binary images. However, in an image that is represented digitally in rectangular tessellation, a problem arises with the definition of connectedness. Consider a dumbbell-shaped object:¹

¹All unmarked image points are taken to have the binary value 0.



At its center, this object has a segment of the form

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

which separates two regions of background. At this point diagonally adjacent 1's are regarded as being connected, whereas diagonally adjacent 0's are regarded as disconnected—a set of allocations that seems inconsistent. However, we can hardly accept a situation where a connected diagonal pair of 0's crosses a connected diagonal pair of 1's without causing a break in either case. Similarly, we cannot accept a situation in which a disconnected diagonal pair of 0's crosses a disconnected diagonal pair of 1's without there being a join in either case. Hence, a symmetrical definition of connectedness is not possible and it is conventional to regard diagonal neighbors as connected only if they are foreground, i.e. the foreground is “8-connected” and the background is “4-connected”. This convention is followed in the subsequent discussion.

9.3 OBJECT LABELING AND COUNTING

Now we have a consistent definition of connectedness, we can unambiguously demarcate all objects in binary images and should be able to devise algorithms for labeling them uniquely and counting them. Labeling may be achieved by scanning the image sequentially until a 1 is encountered on the first object; a note is then made of the scanning position, and a “propagation” routine is initiated to label the whole of the object with a 1; since the original image space is already in use, a separate image space has to be allocated for labeling. Next, the scan is resumed, ignoring all points already labeled, until another object is found; this is labeled with a 2 in the separate image space. This procedure is continued until the whole image has been scanned and all the objects have been labeled (Fig. 9.1). Implicit in this procedure is the possibility of propagating through a connected object. Suppose at this stage no method is available for limiting the

FIGURE 9.1

A process in which all binary objects are labeled.

field of the propagation routine, so that it has to scan the whole image space. Then the propagation routine takes the form:

```
do {
    for all points in image
        if point is in an object
            and next to a propagating region labelled N
            assign it the label N
    } until no further change;
```

(9.1)

the kernel of the do–until loop being expressed more explicitly as:

```
//original image in A-space; labels to be inserted in P-space
for all pixels in image do {
    if((A0 == 1)
        &&((P1 == N)|| (P2 == N)|| (P3 == N)|| (P4 == N)
        ||(P5 == N)|| (P6 == N)|| (P7 == N)|| (P8 == N)))
        P0 = N;
}
```

(9.2)

At this stage a fairly simple type of algorithm for object labeling is obtained, as shown in [Table 9.1](#): the *for forward scan over image do {…}* notation denotes a *sequential* forward raster scan over the image.

Note that the above object counting and labeling routine requires a *minimum* of $2N + 1$ passes over the image space, and in practice the number will be closer to $NW/2$, where W is the average width of the objects: hence, the algorithm is inherently rather inefficient. This prompts us to consider how the number of passes over the image could be reduced to save computation. One possibility would be to scan forward through the image, propagating new labels through objects as they are discovered. While this would work mostly straightforwardly with convex objects, problems would be encountered with objects possessing concavities—e.g. “U” shapes—since different parts of the same object would end with different labels, and also means would have to be devised for coping with “collisions” of labels (e.g. the largest local label could be propagated through the remainder of the object: see [Fig. 9.2](#)). Then inconsistencies could be resolved by a reverse scan through the image. However, this procedure will not resolve all problems that can arise, as in the case of more complex (e.g. spiral) objects. In such cases a general parallel propagation, repeatedly applied until no further

Table 9.1 A Simple Algorithm for Object Labeling

```

//start with binary image containing objects in A-space
//clear label space
for all pixels in image do {P0 = 0;}
//start with no objects
N = 0;
/* look for objects using a sequential scan and propagate
   labels through them*/
do { //search for an unlabeled object
    found = false;
    for forward scan over image do {
        if ((A0 == 1)&&(P0 == 0)&&not found) {
            N = N + 1;
            P0 = N;
            found = true;
        }
    }
    if (found) //label the object just found
    do {
        finished = true;
        for all pixels in image do {
            if ((A0 == 1)&&(P0 == 0)
                &&((P1 == N)||((P2 == N)||((P3 == N)||((P4 == N)
                    ||(P5 == N)||((P6 == N)||((P7 == N)||((P8 == N)))) {
                    P0 = N;
                    finished = false;
                }
            }
        } until finished;
    } until not found;//i.e. no (more) objects
//N is the number of objects found and labeled

```

FIGURE 9.2

Labeling U-shaped objects: a problem that arises in labeling certain types of object if a simple propagation algorithm is used. Some provision has to be made to accept “collisions” of labels although the confusion can be removed by a subsequent stage of processing.

labeling occurs, might be preferable—though as we have seen, such a process is inherently rather computation intensive. However, it is implemented very conveniently on certain types of parallel SIMD processor (see Chapter 26).

Ultimately, the least computationally intensive procedures for propagation involve a different approach: objects and parts of objects are labeled on a *single* sequential pass through the image, at the same time noting which labels coexist on objects. Then the labels are sorted separately, in a stage of abstract information processing, to determine how the initially rather *ad hoc* labels should be interpreted. Finally, the objects are relabeled appropriately in a second pass over the image (in fact, this latter pass is sometimes unnecessary, since the image data are merely labeled in an overcomplex manner and what is needed is simply a key to interpret them). The improved labeling algorithm now takes the form shown in [Table 9.2](#). Clearly this algorithm with its single sequential scan is intrinsically far more efficient than the previous one, although the presence of particular dedicated hardware or a suitable SIMD processor might alter the situation and justify the use of alternative procedures.

Table 9.2 The Improved Algorithm for Object Labeling

```
//clear label space
for all pixels in image do {P0 = 0;}
//start with no objects
N = 0;
//clear the table that is to hold the label coexistence data
for(i = 1; i <= Nmax; i++)
    for(j = 1; j <= Nmax; j++)
        coexist[i][j] = false;
//label objects in a single sequential scan
for forward scan over image do {
    if((A0 == 1)
        if((P2 == 0)&&(P3 == 0)&&(P4 == 0)&&(P5 == 0)) {
            N = N + 1;
            P0 = N;
        }
        else {
            P0 = max(P2, P3, P4, P5);
            //now note which labels coexist in objects
            coexist[P0][P2] = true;
            coexist[P0][P3] = true;
            coexist[P0][P4] = true;
            coexist[P0][P5] = true;
        }
    }
    analyze the coexist table and decide ideal labeling scheme;
    relabel image if necessary;
```

It will be clear that minor amendments to the above algorithms permit the areas and perimeters of objects to be determined: thus, objects may be labeled by their areas or perimeters instead of by numbers representing their order of appearance in the image. More important, the availability of propagation routines means that objects can be considered in turn in their entirety—if necessary by transferring them individually to separate image spaces or storage areas ready for unencumbered independent analysis. Evidently, if objects appear in individual binary spaces, maximum and minimum spatial coordinates are trivially measurable, centroids can readily be found and more detailed calculations of moments (see below) and other parameters can easily be undertaken.

9.3.1 Solving the Labeling Problem in a More Complex Case

In this section, we add substance to the all too facile statement at the end of [Table 9.2](#)—“analyze the coexist table and decide ideal labeling scheme.” First, we have to make the task nontrivial by providing a suitable example. [Figure 9.3](#) shows an example image, in which sequential labeling has been carried out in line with the algorithm of [Table 9.2](#). However, one variation has been adopted—of using a minimum rather than a maximum labeling convention, so that the values are in general slightly closer to the eventual ideal labels. (This also serves to demonstrate that there is not just one way of designing a suitable labeling algorithm.) The algorithm itself indicates that the coexist table should now appear as in [Table 9.3](#).

		1		2	2		
		1		2	2		
		1		2	2		
		1	1	1			
		3	3	1	1	1	
	4	4	3	3	1	1	1
	4	4	3	3	1	1	1
4	4	3	3	1	1	1	
		5				6	
	5		7	7		8	8
5			7	7		8	8
	5	5			7	7	6

FIGURE 9.3

Solving the labeling problem in a more complex case.

Table 9.3 Coexist Table for the Image of Fig. 9.3

	1	2	3	4	5	6	7	8
1		✓	✓					
2	✓							
3	✓			✓				
4			✓					
5							✓	
6								✓
7					✓			✓
8						✓	✓	

The ticks correspond to clashes of labels.

Table 9.4 Coexist Table with Additional Numerical Information

	1	2	3	4	5	6	7	8
1	1	1	1					
2	1	2						
3	1		3	3				
4			3	4				
5					5		5	
6						6		6
7					5		7	7
8						6	7	8

This coexist table is an enhanced version of Table 9.3.

Technically, the numbers along, and below, the leading diagonal are redundant, but nevertheless they speed up the subsequent computation.

However, the whole process of calculating ideal labels can be made more efficient by inserting numbers instead of ticks, and also adding the right numbers along the leading diagonal, as in Table 9.4; for the same reason, the numbers below the leading diagonal, which are technically redundant, are retained here.

The next step is to minimize the entries along the individual rows of the table, as in Table 9.5. Then we minimize along the individual columns (Table 9.6). Then we minimize along rows again (Table 9.7). This process is iterated to

completion, which has already happened here after three stages of minimization. We can now read off the final result from the leading diagonal. Note that a further stage of computation is needed to make the resulting labels consecutive integers, starting with unity. However, the procedure needed to achieve this is much more basic and does not need to manipulate a 2-D table of data: this will be left as a simple programming task for the reader.

At this point, some comment on the nature of the process described above will be appropriate. What has happened is that the original image data has effectively been condensed into the minimum space required to express the labels—namely just one entry per original clash. This explains why the table retains the 2-D format

	1	2	3	4	5	6	7	8
1	1	1	1					
2	1	1						
3	1		1	1				
4			3	3				
5					5		5	
6						6		6
7					5		5	5
8						6	6	6
<i>At this stage the table is no longer symmetric.</i>								

	1	2	3	4	5	6	7	8
1	1	1	1					
2	1	1						
3	1		1	1				
4			1	1				
5					5		5	
6						6		5
7					5		5	5
8						6	5	5

Table 9.7 Coexist Table Redrawn Yet Again with Minimized Rows								
	1	2	3	4	5	6	7	8
1	1	1	1					
2	1	1						
3	1		1	1				
4			1	1				
5					5		5	
6						5		5
7					5		5	5
8						5	5	5

At this stage the table is in its final form, and is once again symmetric.

of the original image: lower dimensionality would not permit the image topology to be represented properly. It also explains why minimization has to be carried out, to completion, in two orthogonal directions. On the other hand, the particular implementation, including both above- and below-diagonal elements, is able to minimize computational overheads and finalize the operation in remarkably few iterations.

Finally, it might be felt that too much attention has been devoted to finding connected components of binary images. In fact, this is a highly important topic in practical applications such as industrial inspection, where it is crucial to locate all the objects unambiguously before they can individually be identified and scrutinized. In addition, Fig. 9.3 makes it clear that it is not only U-shaped objects that give problems, but also those that have shape subtleties—as is seen at the left of the upper object in this figure.

9.4 SIZE FILTERING

Before proceeding to study size filtering, we draw attention to the fact that the 8-connected and 4-connected definitions of connectedness lead to the following measures of distance (or “metrics”) that apply to pairs of pixels, labeled i and j , in a digital lattice:

$$d_8 = \max(|x_i - x_j|, |y_i - y_j|) \quad (9.3)$$

and

$$d_4 = |x_i - x_j| + |y_i - y_j| \quad (9.4)$$

While the use of the d_4 and d_8 metrics is bound to lead to certain inaccuracies, there is a need to see what can be achieved with the use of local operations in

(a)

(b)

FIGURE 9.4

The effect of a simple size filtering procedure. When size filtering is attempted by a set of N shrink and N expand operations, larger objects are restored approximately to their original size but their shapes frequently become distorted or even fragmented. In this example, (b) shows the effect of applying two shrink and then two expand operations to the image in (a).

binary images. This section studies how simple size filtering operations can be carried out, using merely local (3×3) operations. The basic idea is that small objects may be eliminated by applying a series of shrink operations. In fact, N shrink operations will eliminate an object (or those parts of an object) that are $2N$ or fewer pixels across their narrowest dimension. Of course this process shrinks *all* the objects in the image, but in principle a subsequent N expand operations will restore the larger objects to their former size.

If complete elimination of small objects is required but perfect retention of larger objects, this will, however, not be achieved by the above procedure, since in many cases the larger objects will be distorted or fragmented by these operations (Fig. 9.4). To recover the larger objects in their *original* form, the proper

Table 9.8 Algorithm for Recovering Original Forms of Shrunken Objects

```

//save original image
for all pixels in image do {C0 = A0;}
//now shrink the original objects N times
for (i = 1; i <= N; i++) {
    for all pixels in image do {
        sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
        if(sigma < 8) B0 = 0; else B0 = A0;
    }
    for all pixels in image do {A0 = B0;}
}
//next propagate the shrunken objects using the original image
do {
    finished = true;
    for all pixels in image do {
        sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
        if ((A0 == 0)&&(sigma > 0)&&(C0 == 1)) {
            A0 = 1;
            finished = false;
        }
    }
} until finished;

```

approach is to use the shrunken versions as “seeds” from which to grow the originals via a propagation process. The algorithm of **Table 9.8** is able to achieve this.

Having seen how to remove whole (connected) objects that are everywhere narrower than a specified width, it is possible to devise algorithms for removing any subset of objects that are characterized by a given range of widths: large objects may be filtered out by first removing lesser-sized objects and then performing a logical masking operation with the original image, while intermediate-sized objects may be filtered out by removing a larger subset and then restoring small objects that have previously been stored in a separate image space. Ultimately, all these schemes depend on the availability of the propagation technique, which in turn depends on the internal connectedness of individual objects.

Finally, note that expand operations followed by shrink operations may be useful for joining nearby objects, filling in holes, and so on. Numerous refinements and additions to these simple techniques are possible. A particularly interesting one is to separate the silhouettes of touching objects such as chocolates by a shrinking operation: this then permits them to be counted reliably ([Fig. 9.5](#)).

9.5 DISTANCE FUNCTIONS AND THEIR USES

The distance function of an object is a very simple and useful concept in shape analysis. Essentially, each pixel in the object is numbered according to its

(a) (b)

FIGURE 9.5

Separation of touching objects by shrink operations. Here objects (chocolates) in (a) are shrunk (b) in order to separate them so that they may be counted reliably.

FIGURE 9.6

The distance function of a binary shape: the value at every pixel is the distance (in the d_8 metric) from the background.

distance from the background. As usual, background pixels are taken as 0's; then edge pixels are counted as 1's; object pixels next to 1's become 2's; next to those are the 3's; and so on throughout all the object pixels (Fig. 9.6).

The parallel algorithm of Table 9.9 finds the distance function of binary objects by propagation. Note that this algorithm performs a final pass in which nothing happens; this is inevitable if we are to be certain that the process will run to completion.

It is possible to perform the propagation of a distance function with far fewer operations if sequential processing is used. In 1-D, the basic idea would be to build up ramps within objects using a routine like the following:

```
for all pixels in a row of pixels do {Q0 = A0 * 255;}
for forward scan over the row of pixels do
    if(Q0 > Q5 + 1) Q0 = Q5 + 1;
```

Table 9.9 A parallel Algorithm for Propagating Distance Functions

```

//Start with binary image containing objects in A-space
for all pixels in image do{Q0 = A0 * 255;}
N = 0;
do {
    finished = true;
    for all pixels in image do {
        if((Q0 == 255) //in object and no answer yet
           &&((Q1 == N)|| (Q2 == N)|| (Q3 == N)|| (Q4 == N)
           ||(Q5 == N)|| (Q6 == N)|| (Q7 == N)|| (Q8 == N))) {
            //next to an N
            Q0 = N + 1;
            finished = false;//some action has been taken
        }
    }
    N = N + 1;
} until finished;

```

Next, we need to insist on double-sided ramps within objects, both horizontally and vertically. This is elegantly achieved using two sequential operations, one being a normal forward raster scan and the other being a reverse raster scan:

```

for all pixels in image do {Q0 = A0 * 255;}
for forward scan over image do {
    minplusone = min(Q2, Q3, Q4, Q5) + 1;
    if(Q0 > minplusone) Q0 = minplusone;
}
for reverse scan over image do {
    minplusone = min(Q6, Q7, Q8, Q1) + 1;
    if(Q0 > minplusone) Q0 = minplusone;
}

```

(9.5)

Note the compact notation being used to distinguish between forward and reverse raster scans over the image: *for forward scan over image do {…}* denotes a forward raster scan, while *for reverse scan over image do {…}* denotes a reverse raster scan. A more succinct version of this algorithm is the following:

```

for all pixels in image do {Q0 = A0 * 255;}
for forward scan over image do {
    Q0 = min(Q0 - 1, Q2, Q3, Q4, Q5) + 1;
}
for reverse scan over image do {
    Q0 = min(Q0 - 1, Q6, Q7, Q8, Q1) + 1;
}

```

(9.6)

Before moving on, it will be useful to emphasize the value of sequential processing for propagating distance functions. In fact, when this sequential algorithm is run on a serial computer, it will be $O(N)$ times *faster* than the corresponding parallel algorithm running on a serial computer, but $O(N)$ times *slower* than the

FIGURE 9.7

Local maxima of the distance function of the shape shown in Fig. 9.6, the remainder of the shape being indicated by dots and the background being blank. Notice that the local maxima group themselves into clusters each containing points of equal distance function value, while clusters of different values are clearly separated.

same parallel algorithm running on a parallel computer, for an $N \times N$ image. While this statement is specific to propagation of distance functions, similar statements can be made about a good many other operations. (For further discussion relating to parallel computers such as SIMD machines, see Chapter 26.)

9.5.1 Local Maxima and Data Compression

An interesting application of distance functions is that of data compression. To achieve this, operations are carried out to locate the pixels that are local maxima of the distance function (Fig. 9.7), since storing these pixel values and positions permits the original image to be regenerated by a process of downward propagation, as shown at the end of this section. Note that although finding the local maxima of the distance function provides the basic information for data compression, the actual compression occurs only when the data are stored as a list of points rather than in the original picture format. In order to locate the local maxima, the following parallel routine may be employed:

```
for all pixels in image do {
    maximum = max(Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8);
    if((Q0 > 0)&&(Q0 >= maximum)) B0 = 1; else B0 = 0;
}
```

(9.7)

Alternatively, the compressed data can be transferred to a single image space:

```
for all pixels in image do {
    maximum = max(Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8);
    if((Q0 > 0)&&(Q0 >= maximum)) P0 = Q0; else P0 = 0;
}
```

(9.8)

Table 9.10 A Parallel Algorithm for Recovering Objects From Local Maxima of the Distance Functions

```
//assume that input image is in Q-space, and that non-maximum
values have value 0
do {
    finished = true;
    for all pixels in image do {
        maxminusone = max(Q0 + 1, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8) - 1;
        if(Q0 < maxminusone) {
            Q0 = maxminusone;
            finished = false; //some action has been taken
        }
    }
} until finished;
```

Note that the local maxima that are retained for the purpose of data compression are not absolute maxima but are maximal in the sense of not being adjacent to larger values. If this were not so, insufficient numbers of points would be retained for completely regenerating the original object. As a result of this, it is found that the local maxima group themselves into clusters of connected points, each cluster having a common distance value and being separated from points of different distance values (Fig. 9.7). Thus, the set of local maxima of an object is not a connected subset. This fact has an important bearing on skeleton formation (see Section 9.6.3).

Having seen how data compression may be performed by finding local maxima of the distance function, it is relevant to consider a parallel downward propagation algorithm (Table 9.10) for recovering the shapes of objects from an image into which the values of the local maxima have been inserted. Note again that if it can be assumed that at most N passes are needed to propagate through objects of known maximum width, then the algorithm becomes simply:

```
for(i = 1; i <= N; i++)
    for all pixels in image do {
        Q0 = max(Q0 + 1, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8) - 1;
    }
```

(9.9)

9.6 SKELETONS AND THINNING

The skeleton is a powerful analog concept that may be employed for the analysis and description of shapes in binary images. A skeleton may be defined as a connected set of medial lines along the limbs of a figure: for example, in the case of thick hand-drawn characters the skeleton may be supposed to be the path actually traveled by the pen. In fact, the basic idea of the skeleton is that of eliminating redundant information while retaining only the topological information concerning

the shape and structure of the object that can help with recognition. In the case of hand-drawn characters, the thickness of the limbs is taken to be irrelevant: it may be constant and therefore carry no useful information, or it may vary randomly and again be of no value for recognition (Fig. 1.2).

The definition presented above leads to the idea of finding the loci of the centers of maximal disks inserted within the object boundary. First, suppose the image space to be a continuum. Then the disks are circles and their centers form loci that may be modeled very conveniently when object boundaries are approximated by linear segments. In fact, sections of the loci fall into three categories:

1. They may be angle bisectors, i.e. lines which bisect corner angles and reach right up to the apexes of corners.
2. They may be lines that lie half-way between boundary lines.
3. They may be parabolas that are equidistant from lines and from the nearest points of other lines—namely, corners where two lines join.

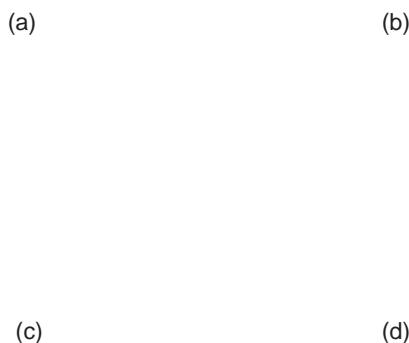
Clearly, categories 1 and 2 are special forms of a more general case.

These ideas lead to unique skeletons for objects with linear boundaries, and the concepts are easily generalizable to curved shapes. In fact, this approach tends to give rather more detail than is commonly required, even the most obtuse corner having a skeleton line going into its apex (Fig. 9.8). Hence, a thresholding scheme is often employed such that skeleton lines only reach into corners having a specified minimum degree of sharpness.

We now have to see how the skeleton concept will work in a digital lattice. Here we are presented with an immediate choice: which metric should we employ? If we select the Euclidean metric (i.e. lattice distance is measured as the Euclidean distance between pairs of pixels), there may be a considerable computational load. If we select the d_8 metric we will immediately lose accuracy but the computational requirements should be more modest (we do not here consider the d_4 metric, since we are dealing with the shapes of foreground objects). In what follows we concentrate on the d_8 metric.

At this stage some thought shows that the application of maximal disks in order to locate skeleton lines amounts essentially to finding the positions of local maxima of the distance function. Unfortunately, as seen in the previous section, the set of local maxima does not form a connected graph within a given object: nor is it necessarily composed of thin lines, and indeed it may at places be 2 pixels wide. Thus, problems arise in trying to use this approach to obtain a connected unit-width skeleton that can conveniently be used to represent the shape of the object. We shall return to this approach again in Section 9.6.3. Meanwhile, however, we pursue an alternative idea—that of thinning.

Thinning is perhaps the simplest approach to skeletonization. It may be defined as the process of systematically stripping away the outermost layers of a figure until only a connected unit-width skeleton remains (see, for example, Fig. 9.9). A number of algorithms are available to implement this process, with

**FIGURE 9.8**

Four shapes whose boundaries consist entirely of straight line segments. The idealized skeletons go right to the apex of each corner, however obtuse. In certain parts of shapes (b)–(d), the skeleton segments are parts of parabolas rather than straight lines. As a result, the detailed shape of the skeleton (or the approximations produced by most algorithms operating in discrete images) is not exactly what might initially be expected or what would be preferred in certain applications.

FIGURE 9.9

Typical result of a thinning algorithm operating in a discrete lattice.

varying degrees of accuracy, and we discuss below how a specified level of precision can be achieved and tested for. First, however, it is necessary to discuss the mechanism by which points on the boundary of a figure may validly be removed in thinning algorithms.

0 0 0	0 0 0	1 0 0	1 0 0	1 0 1	1 0 0	1 0 1
0 1 0	0 1 1	0 1 1	0 1 0	0 1 0	0 1 0	0 1 0
0 0 0	1 1 1	1 1 1	0 1 0	1 1 1	1 0 1	1 0 1
0	2	4	4	6	6	8

FIGURE 9.10

Some examples of the crossing number values associated with given pixel neighborhood configurations (0, background; 1, foreground).

9.6.1 Crossing Number

The exact mechanism for examining points to determine whether they can be removed in a thinning algorithm must now be considered. This may be decided by reference to the *crossing number* χ (chi) for the 8 pixels around the outside of a particular 3×3 neighborhood. χ is defined as the total number of 0-to-1 and 1-to-0 transitions on going once round the outside of the neighborhood: this number is in fact twice the number of potential connections joining the remainder of the object to the center of the neighborhood (Fig. 9.10). Unfortunately, the formula for χ is made more complex by the 8-connectedness criterion. Basically, we would expect²:

$$\begin{aligned} \text{badchi} = & (\text{int})(A1 != A2) + (\text{int})(A2 != A3) + (\text{int})(A3 != A4) \\ & + (\text{int})(A4 != A5) + (\text{int})(A5 != A6) + (\text{int})(A6 != A7) \\ & + (\text{int})(A7 != A8) + (\text{int})(A8 != A1); \end{aligned} \quad (9.10)$$

However, this is incorrect because of the 8-connectedness criterion. For example, in the case

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

the formula gives the value 4 for χ instead of 2. The reason is that the isolated 0 in the top right-hand corner does not prevent the adjacent 1's from being joined. It is therefore tempting to use the modified formula:

$$\text{wrongchi} = (\text{int})(A1 != A3) + (\text{int})(A3 != A5) + (\text{int})(A5 != A7) + (\text{int})(A7 != A1); \quad (9.11)$$

However, this too is wrong, since in the case

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

²In this section, as in the case of C++, the “(int)” construct is used to convert logical outcomes true and false to integer outcomes 1 and 0.

it gives the answer 2 instead of 4. It is therefore necessary to add four extra terms to deal with isolated 1's in the corners:

$$\begin{aligned} \text{chi} = & (\text{int})(A1 != A3) + (\text{int})(A3 != A5) + (\text{int})(A5 != A7) \\ & + (\text{int})(A7 != A1) \\ & + 2 * ((\text{int})((A2 > A1) \& \& (A2 > A3)) + (\text{int})((A4 > A3) \& \& (A4 > A5)) \\ & + (\text{int})((A6 > A5) \& \& (A6 > A7)) + (\text{int})((A8 > A7) \& \& (A8 > A1))); \end{aligned} \quad (9.12)$$

This (now correct) formula for crossing number gives values 0, 2, 4, 6 or 8 in different cases (Fig. 9.10). The rule for removing points during thinning is that points may only be removed if they are at those positions at the boundary of an object where χ is 2. When χ is greater than 2, the point *must* be retained, as it forms a vital connected point between two or more parts of the object; in addition, when it is 0 it must be retained since removing it would create a hole.

Finally, there is one more condition that must be fulfilled before a point can be removed during thinning—that the sum σ (sigma) of the eight pixel values around the outside of the 3×3 neighborhood (see Chapter 2) must not be equal to 1. The reason for this is to preserve line ends, as in the following cases:

$$\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{array}$$

Clearly, if line ends are eroded as thinning proceeds, the final skeleton will not represent the shape of an object (including the relative dimensions of its limbs) at all accurately (however, it is possible that we might sometimes wish to shrink an object while preserving connectedness, in which case this extra condition need not be implemented). Having covered these basics, we are now in a position to devise complete thinning algorithms.

9.6.2 Parallel and Sequential Implementations of Thinning

Thinning is “essentially sequential” in that it is easiest to ensure that connectedness is maintained by arranging that only one point may be removed at a time. As indicated above, this is achieved by checking before removing a point that it has a crossing number of 2. Now imagine applying the “obvious” sequential algorithm of Table 9.11 to a binary image. Assuming a normal forward raster scan, the result of this process is to produce a highly distorted skeleton, consisting of lines along the right-hand and bottom edges of objects. It may now be seen that the $\chi = 2$ condition is necessary but not sufficient, since it says nothing about the order in which points are removed. To produce a skeleton that is unbiased, giving a set of truly medial lines, it is necessary to remove points as evenly as possible around the object boundary. A scheme that helps with this involves a novel processing sequence: mark edge points on the first pass over an image; on the second pass, strip points sequentially as in the above algorithm, *but only where they have already been marked*; then mark a new set of edge points; then perform another stripping pass; then repeat this marking and stripping sequence until no further

Table 9.11 An “Obvious” Sequential Thinning Algorithm

```

do {
    finished = true;
    for forward scan over image do {
        sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
        chi = (int)(A1 != A3) + (int)(A3 != A5) + (int)(A5 != A7)
            + (int)(A7 != A1)
            + 2 * ((int)((A2 > A1) && (A2 > A3)) + (int)((A4 > A3) && (A4 > A5))
            + (int)((A6 > A5) && (A6 > A7)) + (int)((A8 > A7) && (A8 > A1)));
        if((A0 == 1) && (chi == 2) && (sigma != 1)) {
            A0 = 0;
            finished = false; //some action has been taken
        }
    }
} until finished;

```

change occurs. An early algorithm working on this principle is that of Beun (1973).

While the novel sequential thinning algorithm described above can be used to produce a reasonable skeleton, it would be far better if the stripping action could be performed symmetrically around the object, thereby removing any possible skeletal bias. In this respect a parallel algorithm should have a distinct advantage. However, parallel algorithms result in several points being removed at once: this means that lines 2 pixels wide will disappear (since masks operating in a 3×3 neighborhood cannot “see” enough of the object to judge whether a point may validly be removed or not), and as a result shapes can become disconnected. The general principle for avoiding this problem is to strip points lying on different parts of the boundary in different passes, so that there is no risk of causing breaks. In fact, there is a very large number of ways of achieving this, by applying different masks and conditions to characterize different parts of the boundary. If boundaries were always convex the problem would no doubt be reduced; however, boundaries can be very convoluted and are subject to quantization noise, so the problem is a complex one. With so many potential solutions to the problem, we concentrate here on one that can conveniently be analyzed and which gives acceptable results.

The method discussed is that of removing north, south, east, and west points cyclically until thinning is complete. North points are defined as the following:

×	0	×
×	1	×
×	1	×

where \times means either a 0 or a 1: south, east, and west points are defined similarly. It is easy to show that all north points for which $\chi = 2$ and $\sigma \neq 1$ may be removed in parallel without any risk of causing a break in the skeleton—and similarly for south, east, and west points. Thus, a possible format for a parallel thinning algorithm in rectangular tessellation is the following:

```
do {
    strip appropriate north points;
    strip appropriate south points;
    strip appropriate east points;
    strip appropriate west points;
} until no further change; (9.13)
```

where the basic parallel routine for stripping “appropriate” north points is:

```
for all pixels in image do {
    sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8;
    chi = (int)(A1 != A3) + (int)(A3 != A5) + (int)(A5 != A7)
        + (int)(A7 != A1)
        + 2 * ((int)((A2 > A1) && (A2 > A3)) + (int)((A4 > A3) && (A4 > A5))
        + (int)((A6 > A5) && (A6 > A7)) + (int)((A8 > A7) && (A8 > A1)));
    if((A3 == 0) && (A0 == 1) && (A7 == 1) //north point
        && (chi == 2) && (sigma != 1))
        B0 = 0;
    else B0 = A0;
} (9.14)
```

(but extra code needs to be inserted to detect whether any changes have been made in a given pass over the image).

Algorithms of the above type can be highly effective, although their design tends to be rather intuitive and *ad hoc*. In a survey made by the author in 1981 (Davies and Plummer, 1981), a great many such algorithms exhibited problems. Ignoring cases where the algorithm design was insufficiently rigorous to maintain connectedness, four other problems were evident:

1. the problem of skeletal bias;
2. the problem of eliminating skeletal lines along certain limbs;
3. the problem of introducing “noise spurs”;
4. the problem of slow speed of operation.

In fact, problems 2 and 3 are opposites in many ways: if an algorithm is designed to suppress noise spurs, it is liable to eliminate skeletal lines in some circumstances; contrariwise, if an algorithm is designed never to eliminate skeletal lines, it is unlikely to be able to suppress noise spurs. This situation arises since the masks and conditions for performing thinning are intuitive and *ad hoc*, and therefore have no basis for discriminating between valid and invalid skeletal lines: ultimately this is because it is difficult to build overt global models of reality into purely local operators. In a similar way, algorithms that proceed with caution, i.e. which do not remove object points in the fear of making an error or causing bias, tend to be slower in operation than they might otherwise be. Again,

it is difficult to design algorithms that can make correct global decisions rapidly via intuitively designed local operators. Hence, a totally different approach is needed if solving one of the above problems is not to cause difficulties with the others. Such an alternative approach is discussed in the next section.

9.6.3 Guided Thinning

This section returns to the ideas of [Section 9.5.1](#), where it was found that the local maxima of the distance function do not form an ideal skeleton because they appear in clusters and are not connected. In addition, the clusters are often two pixels wide. On the plus side, the clusters are accurately in the correct positions and should therefore not be subject to skeletal bias. Hence, an ideal skeleton should result if (a) the clusters could be reconnected appropriately and (b) the resulting structure could be reduced to unit width—though, of course, a unit-width skeleton can only be perfectly unbiased where the object is an odd number of pixels wide.

A simple means of reconnecting the clusters is to use them to guide a conventional thinning algorithm (see [Section 9.6.2](#)). As a first stage, thinning is allowed to proceed normally but with the proviso that no cluster points may be removed. This gives a connected graph which at certain places is 2 pixels wide. Then a routine is applied to strip the graph down to unit width. At this stage an unbiased skeleton (within 1/2 pixel) should result. The main problem here is the presence of noise spurs. The opportunity now exists to eliminate these systematically by applying suitable global rules. A simple rule is that of eliminating lines on the skeletal graph that terminate in a local maximum of value (say) 1 (or, better, stripping them back to the next local maximum), since such a local maximum corresponds to fairly trivial detail on the boundary of the object. Thus, the level of detail that is ignored can be programmed into the system (Davies and Plummer, 1981). The whole guided thinning process is shown in [Fig. 9.11](#).

9.6.4 A Comment on the Nature of the Skeleton

At the beginning of [Section 9.6](#), the case of character recognition was taken as an example and it was stated that the skeleton may be supposed to be the path traveled by the pen in drawing out the character. However, in one important respect this is not valid. The reason is seen both in the analog reasoning and from the results of thinning algorithms. Take the case of a letter K. The vertical limb on the left of the skeleton theoretically consists of two linear segments joined by two parabolic segments leading into the junction ([Fig. 9.8](#)). This limb will only become straight if a higher level model is used to constrain the result.

9.6.5 Skeleton Node Analysis

Skeleton node analysis may be carried out very simply with the aid of the crossing number concept. Points in the middle of a skeletal line have a crossing

**FIGURE 9.11**

Results of a guided thinning algorithm: (a) distance function on the original shape; (b) set of local maxima; (c) set of local maxima now connected by a simple thinning algorithm; (d) final thinned skeleton. The effect of removing noise spurs systematically, by cutting limbs terminating in a 1 back to the next local maximum, is easily discernible from the result in (d): the general shape of the object is not perturbed by this process.

number of 4; points at the end of a line have crossing number 2; points at skeletal “T” junctions have a crossing number of 6; and points at skeletal “X” junctions have a crossing number of 8. However, there is a situation to beware of—places which look like a “+” junction:

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

In such places the crossing number is actually 0 (see formula), although the pattern is definitely that of a cross. At first the situation seems to be that there is insufficient resolution in a 3×3 neighborhood to identify a “+” cross, the best option being to look for this particular pattern of 0’s and 1’s and use a more sophisticated construct than the 3×3 crossing number to check whether or not a

cross is present. The problem is that of distinguishing between two situations such as:

$$\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & 1 & 1 & 1 & 0 & \text{and} \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & \end{array} \quad \begin{array}{cccccc} 0 & 0 & 1 & 0 & 0 & \\ 1 & 0 & 1 & 0 & 0 & \\ 1 & 1 & 1 & 1 & 1 & \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 0 & \end{array}$$

However, further analysis shows that the first of these two cases would be thinned down to a dot (or a short bar), so that if a “+” node appears on the final skeleton (as in the second case) it actually signifies that a cross is present despite the contrary value of χ . Davies and Celano (1993) have shown that the proper measure to use in such cases is the *modified* crossing number $\chi_{\text{skel}} = 2\sigma$, this crossing number being different from χ because it is required not to test whether points can be eliminated from the skeleton, but to ascertain the meaning of points that are at that stage *known* to lie on the final skeleton. Note that χ_{skel} can have values as high as 16—it is not restricted to the range 0–8!

Finally, note that sometimes insufficient resolution really is a problem, in that a cross with a shallow crossing angle appears as two “T” junctions:

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \end{array}$$

Clearly, resolution makes it impossible to recognize an asterisk or more complex figure from its crossing number, within a 3×3 neighborhood. Probably, the best solution is to label junctions tentatively, then to consider all the junction labels in the image, and to analyze whether a particular local combination of junctions should be reinterpreted—e.g. two “T” junctions may be deduced to form a cross. This is especially important in view of the distortions that appear on a skeleton in the region of a junction (see [Section 9.6.4](#)).

9.6.6 Application of Skeletons for Shape Recognition

Shape analysis may be carried out simply and conveniently by analysis of skeleton shapes and dimensions. Clearly, study of the nodes of a skeleton (points for which there are other than two skeletal neighbors) can lead to the classification of simple shapes but not, for example, discrimination of all block capitals from each other. Many classification schemes exist which can complete the analysis, in terms of limb lengths and positions, and methods for achieving this are touched on in later chapters.

A similar situation exists for the analysis of the shapes of chromosomes, which take the form of a cross or a “V”. For small industrial components, more

detailed shape analysis is called for; this can still be approached with the skeleton technique, by examination of distance function values along the lines of the skeleton. In general, shape analysis using the skeleton proceeds by examination in turn of nodes, limb lengths and orientations, and distance function values, until the required level of characterization is obtained.

The particular importance of the skeleton as an aid in the analysis of connected shapes is not only that it is invariant under translations and rotations but also that it embodies what is for many purposes a highly convenient representation of the figure that (with the distance function values) essentially carries all the original information. If the original shape of an object can be deduced exactly from a representation, this is generally a good sign since it means that it is not merely an *ad hoc* descriptor of shape but that considerable reliance may be placed on it (compare other methods such as the circularity measure—see [Section 9.7](#)).

9.7 OTHER MEASURES FOR SHAPE RECOGNITION

There are many simple tests of shape that can be made to confirm the identity of objects or to check for items such as defects. These include measurements of product area and perimeter, length of maximum dimension, moments relative to the centroid, number and area of holes, area and dimensions of the convex hull (see below) and enclosing rectangle, number of sharp corners, number of intersections with a check circle and angles between intersections ([Fig. 9.12](#)), and numbers and types of skeleton nodes.

The list would not be complete without a mention of the widely used shape measure $C = \text{area}/\text{perimeter}^2$. This quantity is often called “circularity” or “compactness,” since it has a maximum value of $1/4\pi$ for a circle, decreases as shapes become more irregular, and approaches zero for long narrow objects; alternatively, its reciprocal is sometimes used, being called “complexity” since it increases in size as shapes become more complex. Note that both measures are dimensionless so that they are independent of size and are therefore sensitive only

C

FIGURE 9.12

Rapid product inspection by polar checking.

to the shape of an object. Other dimensionless measures of this type include rectangularity and aspect ratio.

All these measures have the property of characterizing a shape but not of describing it uniquely. Thus, it is easy to see that there are in general many different shapes having the same values of parameters such as circularity. Hence, these rather *ad hoc* measures are on the whole less valuable than approaches such as skeletonization (Section 9.6) or moments (see below) that can be used to represent and reproduce a shape to any required degree of precision. Nevertheless, rigorous checking of even one measured number to high precision often permits a machined part to be identified positively.

The use of moments for shape analysis was mentioned above: these are widely used and should be covered in more detail. In fact, moment approximations provide a rigorous means of describing 2-D shapes, and take the form of series expansions of the type:

$$M_{pq} = \sum_x \sum_y x^p y^q f(x, y) \quad (9.15)$$

for a picture function $f(x, y)$; such a series may be curtailed when the approximation is sufficiently accurate. By referring axes to the centroid of the shape, moments can be constructed that are position-invariant: they can also be normalized so that they are invariant under rotation and change of scale (Hu, 1962; see also Wong and Hall, 1978). The main value of using moment descriptors is that in certain applications the number of parameters may be made small without significant loss of precision—although the number required may not be clear without tests being made on a range of relevant shapes. Moments can prove particularly valuable in describing shapes such as cams and other fairly round objects, although they have also been used in a variety of other applications including aeroplane silhouette recognition (Dudani et al., 1977).

The convex hull was also mentioned above and has also been used as the basis for sophisticated, complete descriptions of shapes. The *convex hull* is defined as the smallest convex shape that contains the original shape (it may be envisaged as the shape contained by an elastic band placed around the original shape). The *convex deficiency* is defined as the shape that has to be added to a given shape to create the convex hull (Fig. 9.13). The convex hull may be used as a simple approximation providing a rapid indication of the extent of an object. A fuller description of the shape of an object may be obtained by means of *concavity trees*: here the convex hull of an object is first obtained with its convex deficiencies, then the convex hulls and deficiencies of the obtained convex deficiencies are found, then the convex hulls and deficiencies of these convex deficiencies—and so on until all the derived shapes are convex, or until an adequate approximation to the original shape is obtained. Thus, a tree is formed that can be used for systematic shape analysis and recognition (Fig. 9.14). We shall not dwell on this approach beyond noting its inherent utility and that at its core is the need for a reliable means of determining the convex hull of a shape.

FIGURE 9.13

Convex hull and convex deficiency. The convex hull is the shape enclosed on placing an elastic band around an object. The shaded portion is the convex deficiency that is added to the shape to create the convex hull.

(a) (b)

FIGURE 9.14

A simple shape and its concavity tree. The shape in (a) has been analyzed by repeated formation of convex hulls and convex deficiencies until all the constituent regions are convex (see text). The tree representing the entire process as shown in (b): at each node, the branch on the left is the convex hull and the branches on the right are convex deficiencies.

A simple strategy for obtaining the convex hull is to repeatedly fill in the center pixel of all neighborhoods that exhibit a concavity, including each of the following:

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{array} \quad \begin{array}{ccc} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{array}$$

until no further change occurs. In fact, the shapes obtained by the above approach are larger than ideal convex hulls and approximate to octagonal (or degenerate octagonal) shapes. Hence more complex algorithms are required to generate convex hulls, a useful approach involving the use of boundary tracking to search for positions on the boundary that have common tangent lines.

9.8 BOUNDARY TRACKING PROCEDURES

The preceding sections have described methods of analyzing shape on the basis of body representations such as skeletons and moments. However, an important approach has so far been omitted—the use of boundary pattern analysis. This approach has the potential advantage of requiring considerably reduced computation, since the number of pixels to be examined is equal to the number of pixels on the boundary of any object rather than the much larger number of pixels within the boundary. Before proper use can be made of boundary pattern analysis techniques, means must be found for tracking systematically around the boundaries of all the objects in an image: in addition, care must be taken not to ignore any holes that are present or any objects within holes.

In one sense the problem has been analyzed already, in that the object labeling algorithm of [Section 9.3](#) systematically visits and propagates through all objects in the image. All that is required now is some means of tracking round object boundaries once they have been encountered. Quite clearly it will be useful to mark in a separate image space all points that have been tracked: alternatively, an object boundary image may be constructed and the tracking performed in this space, all tracked points being eliminated as they are passed.

In the latter procedure, objects having unit width in certain places may become disconnected. Hence, we ignore this approach and adopt the previous one. There is still a problem when objects have unit-width sections, since these can cause badly designed tracking algorithms to choose a wrong path, going back around the previous section instead of on to the next ([Fig. 9.15](#)). To avoid this circumstance it is best to adopt the following strategy:

1. track round each boundary, keeping to the left path consistently;
2. stop the tracking procedure only when passing through the starting point in the original direction (or passing through the first two points in the same order).

Apart from necessary initialization at the start, a suitable tracking procedure is given in [Table 9.12](#).

Having seen how to track around the boundaries of objects in binary images, we are now in a position to embark on boundary pattern analysis. This is done in Chapter 10.

9.9 CONCLUDING REMARKS

This chapter has concentrated on rather traditional methods of performing image analysis—using image processing techniques. This has led naturally to area representations of objects, including for example moment and convex hull-based

FIGURE 9.15

A problem with an over-simple boundary tracking algorithm: the boundary tracking procedure takes a short-cut across a unit-width boundary segment instead of continuing and keeping to the left path at all times.

Table 9.12 Basic Procedure for Tracking Around a Single Object

```

do {
    //find direction to move next
    start with current tracking direction;
    reverse it;
    do {
        rotate tracking direction clockwise
    } until the next 1 is met on outer pixels of 3×3 neighborhood;
    record this as new current direction;
    move one pixel along this direction;
    increment boundary index;
    store current position in boundary list;
} until (position == original position)&&(direction == original direction)

```

schemes, although the skeleton approach appeared as a rather special case in that it converts objects into graphical structures. An alternative schema is to represent shapes by their boundary patterns, after applying suitable tracking algorithms: this latter approach is considered in the following chapter. Meanwhile, connectedness has been an underlying theme in the present chapter, objects being separated from each other by regions of background, thereby permitting objects to be considered and characterized individually. Connectedness has been seen to involve rather more intricate computations than might have been expected, and this necessitates great care in the design of algorithms: this must partly explain why after so many

years, new thinning algorithms are still being developed (e.g. Kwok, 1989; Choy et al., 1995) (ultimately, these complexities arise because global properties of images are being computed by purely local means).

Although it will turn out that boundary pattern analysis is in certain ways more attractive than region pattern analysis, this comparison cannot be completely divorced from considerations of the hardware the algorithms have to run on. In this respect, note that many of the algorithms of this chapter can be performed efficiently on SIMD processors, which have one processing element per pixel (see Chapter 26), whereas boundary pattern analysis will be seen to match better the capabilities of more conventional serial computers.

Shape analysis can be attempted by boundary or region representations. Both are deeply affected by connectedness and related metric issues for a digital lattice of pixels. This chapter has shown that these issues are only solved by carefully incorporating global knowledge alongside local information—e.g. by use of distance transforms.

9.10 BIBLIOGRAPHICAL AND HISTORICAL NOTES

The development of shape analysis techniques has been particularly extensive: hence, only a brief perusal of the history is attempted here. The all-important theory of connectedness and the related concept of adjacency in digital images was developed largely by Rosenfeld (see for example, Rosenfeld, 1970). The connectedness concept led to the idea of a distance function in a digital picture (Rosenfeld and Pfaltz, 1966, 1968), and the related skeleton concept (Pfaltz and Rosenfeld, 1967). However, the basic idea of a skeleton dates from the classic work by Blum (1967)—see also Blum and Nagel (1978). Important work on thinning has been carried out by Arcelli et al. (1975, 1981), Arcelli and di Baja (1985) and parallels work by Davies and Plummer (1981). The latter paper demonstrates possibilities for limb pruning, and a rigorous method for testing the results of *any* thinning algorithm, however generated, and in particular for detecting skeletal bias. More recently, Arcelli and Ramella (1995) have reconsidered the problem of skeletons in grayscale images. There have also been important developments to generalize the distance function concept and to make distance functions uniform and isotropic: see for example Huttenlocher et al. (1993). The design of a modified crossing number χ_{skel} for the analysis of skeletal shape dates from the same period: as pointed out in Section 9.6.5, χ_{skel} is different from χ since it evaluates the *remaining* (i.e. skeletal) points rather than points that *might* be eliminated from the skeleton (Davies and Celano, 1993).

Sklansky has carried out much work on convexity and convex hull determination (see for example, Sklansky, 1970; Sklansky et al., 1976), while Batchelor (1979) developed concavity trees for shape description. Haralick et al. (1987) have generalized the underlying mathematical (morphological) concepts,

including the case of grayscale analysis. Use of invariant moments for pattern recognition dates from the two seminal papers by Hu (1961, 1962). Pavlidis has drawn attention to the importance of unambiguous (“asymptotic”) shape representation schemes (Pavlidis, 1980)—as distinct from *ad hoc* sets of shape measures.

In the early 2000s, skeletons maintained their interest and utility, becoming if anything more precise by reference to exact analog shapes (Kégl and Krzyżak, 2002), and giving rise to the concept of a shock graph, which characterizes the result of the much earlier grass-fire transformation (Blum, 1967) more rigorously (Giblin and Kimia, 2003). Wavelet transforms have also been used to implement skeletons more accurately in the discrete domain (Tang and You, 2003). In contrast, shape matching has been carried out using self-similarity analysis coupled with tree representation—an approach that has been especially valuable for tracking articulated object shapes, including human profiles and hand motions (Geiger et al., 2003). It is interesting to see graphical analysis of skeletonized hand-written character shapes performed taking account of catastrophe theory (Chakravarty and Kompella, 2003): this is relevant because (a) critical points—where points of inflection exist—can be deformed into pairs of points each corresponding to a curvature maximum plus a minimum; (b) crossing of t’s can be actual or non-crossing; and (c) loops can turn into cusps or corners (many other possibilities also exist). The point is that methods are needed for mapping between *variations* of shapes rather than making snap judgements as to classification (this corresponds to the difference between scientific understanding of process and *ad hoc* engineering).

9.10.1 More Recent Developments

More recently, increased attention has been devoted to processing skeletons and using them for object matching and classification. Bai and Latecki (2008) discuss how to prune skeletons meaningfully, by ensuring that endpoints of skeleton branches correspond to visual parts of objects (such as all the legs of a horse). Once this has been achieved, it should be possible to match objects (such as horses) in spite of any articulations or contour deformations that may have taken place. The method is found to permit much more efficient matching and to be more resistant to partial occlusion: this is because meaningfulness is built into the final skeleton, while minor intricacies (which may originally have been due to noise forming tiny holes in the object) will have been eliminated. This approach is potentially useful for tracking, stereo matching, and database matching. Ward and Hamarneh (2010) attend to the order in which skeleton branches should be pruned. They report on several pruning algorithms and quantify their performance in terms of denoising, classification, and within-class skeleton similarity measures. The work is important because of the well-known fact that the medial axis transform is unstable with respect to minor perturbations on the boundary of a shape: this means that before skeletons can be used reliably, noise spurs need to be pruned so that they correspond to the underlying shapes.

9.11 PROBLEMS

1. Write the full C++ routine required to sort the lists of labels, to be inserted at the end of the algorithm of [Table 9.2](#).
2. Show that, as stated in [Section 9.6.2](#) for a parallel thinning algorithm, all north points may be removed in parallel without any risk of causing a break in the skeleton.
3. Describe methods for locating, labeling and counting objects in binary images. You should consider whether methods based on propagation from a “seed” pixel, or those based on progressively shrinking a skeleton to a point, would provide the more efficient means for achieving the stated aims. Give examples for objects of various shapes.
4.
 - a. Give a simple one-pass algorithm for labeling the objects appearing in a binary image, making clear the role played by connectedness. Give examples showing how this basic algorithm goes wrong with real objects: illustrate your answer with clear pixel diagrams, which show the numbers of labels that can appear on objects of different shapes.
 - b. Show how a table-orientated approach can be used to eliminate multiple labels in objects. Make clear how the table is set up and what numbers have to be inserted into it. Are the number of iterations needed to analyze the table similar to the number that would be needed in a multi-pass labeling algorithm taking place entirely within the original image? Consider how the *real* gain in using a table to analyze the labels arises.
5.
 - a. Using the following notation for a 3×3 window:

A4	A3	A2
A5	A0	A1
A6	A7	A8

work out the effect of the following algorithm on a binary image containing small foreground objects of various shapes:

```

do{
    for all pixels in image do {
        sum=(int)(A1 + A3 == 2)+(int)(A3 + A5 == 2)
        + (int)(A5 + A7 == 2)+(int)(A7 + A1 == 2);
        if(sum>0) B0 = 1; else B0 = A0;
    }
    for all pixels in image do {A0 = B0;}
} until no further change;

```

6.
 - b. Show in detail how to implement the *do ... until no further change* function in this algorithm.
 - a. Give a simple algorithm operating in a 3×3 window for generating a *rectangular* convex hull around each object in a binary image. Include

in your algorithm any necessary code for implementing the required *do ... until no further change* function.

- b.** A more sophisticated algorithm for finding accurate convex hulls is to be designed. Explain why this would employ a boundary tracking procedure. State the general strategy of an algorithm for tracking around the boundaries of objects in binary images and write a program for implementing it.
 - c.** Suggest a strategy for designing the complete convex hull algorithm and indicate how rapidly you would expect it to operate, in terms of the size of the image.
- 7. a.** Explain the meaning of the term *distance function*. Give examples of the distance functions of simple shapes, including that shown in Fig. 9.16.
- b.** Rapid image transmission is to be performed by sending only the coordinates and values of the local maxima of the distance functions. Give a complete algorithm for finding the local maxima of the distance functions in an image, and devise a further algorithm for reconstructing the original binary image.
 - c.** Discuss which of the following sets of data would give more compressed versions of the binary picture object shown in Fig. 9.16:
 - i.** The list of local maxima coordinates and values.
 - ii.** A list of the coordinates of the boundary points of the object.
 - iii.** A list consisting of one point on the boundary and the relative directions (each expressed as a 3-bit code) between each *pair* of boundary points on tracking round the boundary.

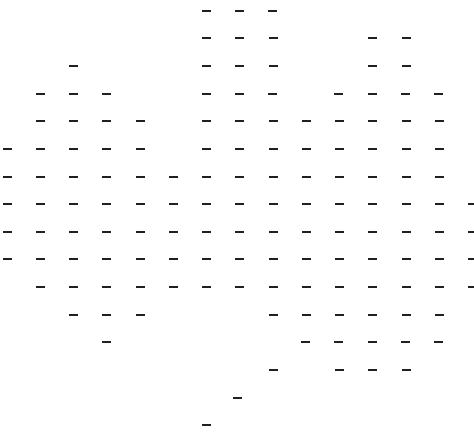


FIGURE 9.16

Binary picture object for shape analysis tests.

- 8. a.** What is the *distance function* of a binary image? Illustrate your answer for the case where a 128×128 image P contains just the object shown in Fig. 9.17. How many passes of (i) a parallel algorithm and (ii) a sequential algorithm would be required to find the distance function of the image?

- b. Give a complete parallel *or* sequential algorithm for finding the distance function, and explain how it operates.
 - c. Image P is to be transmitted rapidly by determining the coordinates of locations where the distance function is locally maximum. Indicate the positions of the local maxima, and explain how the image could be reconstructed at the receiver.
 - d. Determine the compression factor η if image P is transmitted in this way.

FIGURE 9.17

Binary picture object for distance function analysis.

Show that η can be increased by eliminating some of the local maxima before transmission, and estimate the resulting increase in η .

9. a. The local maxima of the distance function can be defined in the following two ways:

 - Pixels whose values are greater than those of all the neighboring pixels.
 - Pixels whose values are greater than *or equal to* the values of all the neighboring pixels. Which definition of the local maxima would be more useful for reproducing the original object shapes? Why is this?

b. Give an algorithm that is capable of reproducing the original object shapes from the local maxima of the distance function, and explain how it operates.

c. Explain the *run-length encoding* approach to image compression. Compare the run-length encoding and local maxima methods for compressing binary images. Explain why the one method would be expected to lead to a greater degree of compression with some types of image while the other method would be expected to be better with other types of image.

10. a. Explain how propagation of a distance function may be carried out using a parallel algorithm. Give in full a simpler algorithm that operates using two sequential passes over the image.

- b.** It has been suggested that a four-pass sequential algorithm will be even faster than the two-pass algorithm, as each pass can use just a 1-D window involving at most three pixels. Write down the code for *one* typical pass of the algorithm.
- c.** Estimate the approximate speeds of these three algorithms for computing the distance function, in the case of an $N \times N$ pixel image. Assume a conventional serial computer is used to perform the computation.
- 11.** Small dark insects are to be located among cereal grains. The insects approximate to rectangular bars of dimensions 20×7 pixels, and the cereal grains are approximately elliptical with dimensions 40×24 pixels. The proposed algorithm design strategy is: (i) apply an edge detector which will mark all the edge points in the image as 0's in a 1's background, (ii) propagate a distance function in the *background* region, (iii) locate the local maxima of the distance function, (iv) analyze the values of the local maxima, and (v) carry out necessary further processing to identify the nearly parallel sides of the insects. Explain how to design stages (iv) and (v) of the algorithm in order to identify the insects, ignoring the cereal grains. Assume that the image is not so large that the distance function will overflow the byte limit. Determine how robust the method will be if the edge is broken in a few places.
- 12.** Give the general strategy of an algorithm for tracking around the boundaries of objects in binary images. If the tracker has reached a boundary point with crossing number $\chi = 2$ and neighborhood

0	0	1
0	1	1
1	1	1

decide in which direction it should now proceed. Hence, give a complete procedure for determining the direction code of the next position on the boundary for cases where $\chi = 2$. Take the direction codes starting *from* the current pixel (*) as being specified by the following diagram:

4	3	2
5	*	1
6	7	8

How should the procedure be modified for cases where $\chi \neq 2$?

- 13. a.** Explain the principles involved in tracking around the boundaries of objects in binary images to produce reliable outlines. Outline an algorithm which can be used for this purpose, assuming it is to get its information from a normal 3×3 window.
- b.** A binary image is obtained and the data in it is to be compressed as much as possible. The following range of algorithms is to be tested for this purpose:

- i. the boundary image;
 - ii. the skeleton image;
 - iii. the image of the local maxima of the distance function;
 - iv. the image of a suitably chosen subset of the local maxima of the distance function;
 - v. a set of run-length data, i.e. a series of numbers obtained by counting runs of 0's, then runs of 1's, then runs of 0's, and so on, in a continuous line-by-line scan over the image.
- c. In (i) and (ii) the lines may be encoded using chain code, i.e. giving the *coordinates* of the first point met, and the *direction* of each subsequent point using the direction codes 1–8 defined relative to the current position C by:
- | | | |
|---|---|---|
| 4 | 3 | 2 |
| 5 | C | 1 |
| 6 | 7 | 8 |
- d. With the aid of suitably chosen examples, discuss which of the methods of data compression should be most suitable for different types of data. Wherever possible, give numerical estimates to support your arguments.
 - e. Indicate what you would expect to happen if noise were added to initially noise-free input images.
14. Test the two-mask strategy outlined in Section 9.7 for obtaining the convex hulls of binary picture objects. Confirm that it operates consistently, and give a geometrical construction that predicts the final shapes it produces. What happens if either the first or the second mask is used on its own? Show that the two-mask strategy will operate both as a sequential and as a parallel algorithm. Devise a version of the algorithm that does not permit nearby shapes to be merged.

10 Boundary Pattern Analysis

Recognition of binary objects by boundary pattern analysis should be a straightforward process, but this chapter shows that there are a number of problems to be overcome. In particular, any boundary distortions such as those due to breakage or several objects being in contact may result in total failure of the matching process. This chapter discusses the problems and their solution.

Look out for:

- the centroidal profile approach and its limitations.
- how the method may be speeded up.
- how recognition based on the (s, ψ) boundary plot is significantly more robust.
- how the (s, ψ) plot leads on to the more convenient (s, κ) plot.
- the relation between κ and ψ .
- more rigorous ways of dealing with the occlusion problem.
- discussion of the accuracy of boundary length measures.

Disparate ways are available for representing object boundaries and for measuring and recognizing objects using them. All the methods are subject to the same ultimate difficulties—particularly that of managing occlusion (which necessarily removes relevant data), and that of inaccuracy in the pixel-based description. Sound ways of managing occlusion are indicated in [Section 10.6](#). This work presages later chapters where methods such as the Hough transform are widely employed for robust object recognition.

10.1 INTRODUCTION

Chapter 4 has shown how thresholding may be used to binarize grayscale images and hence to present objects as 2-D shapes. However, that method of segmenting objects is successful only when considerable care is taken with lighting and when the object can be presented conveniently, e.g., as dark patches on a light

background. In other cases, adaptive thresholding may help to achieve the same end: as an alternative, edge detection schemes can be applied, which are generally rather more resistant to problems of lighting. Nevertheless, thresholding of edge-enhanced images still gives certain problems: in particular, edges may peter out in some places and may be thick in others (Fig. 10.1). For many purposes, the output of an edge detector is ideally a connected unit-width line around the periphery of an object and steps need to be taken to convert edges to this form—if this has not already been achieved using a Canny or other operator employing nonmaximum suppression (see Chapter 5).



FIGURE 10.1

Some problems with edges. The edge-enhanced image (b) from an original image (a) is thresholded as in (c). The edges so detected are found to peter out in some places and to be thick in other places. A thinning algorithm is able to reduce the edges to unit thickness (d) but *ad hoc* (i.e., not model-based) linking algorithms are liable to produce erroneous results (not shown).

Thinning algorithms can be used to reduce edges to unit thickness while maintaining connectedness (Fig. 10.1(d)). Many algorithms have been developed for this purpose, and the main problems here are: (a) slight bias and inaccuracy due to uneven stripping of pixels, especially in view of the fact that even the best algorithm can only produce a line that is locally within 1/2 pixel of the ideal position; and (b) introduction of a certain number of noise spurs. The first problem can be minimized by using grayscale edge thinning algorithms, which act directly on the original grayscale edge-enhanced image (e.g., Paler and Kittler, 1983). Noise spurs around object boundaries can be eliminated quite efficiently by removing lines that are shorter than (say) 3 pixels. Overall, the major problem to be dealt with is that of reconnecting boundaries that have become fragmented during thresholding.

A number of rather *ad hoc* schemes are available for relinking broken boundaries: e.g., line ends may be extended along their existing directions—a very limited procedure since there are (at least for binary edges) only eight possible directions, and it is quite possible for the extended line ends not to meet. Another approach is to join line ends that are sufficiently close together and point in similar directions, both to each other *and* to the direction of the vector between the two ends. In fact, this approach can be made quite credible in principle, but in practice it can lead to all sorts of problems as it is still *ad hoc* and not model driven. Hence, adjacent lines that arise from genuine surface markings and shadows may be arbitrarily linked together by such algorithms. In many situations, it is therefore best if the process is model driven—e.g., by finding the best fit to some appropriate idealized boundary such as an ellipse. Yet another approach is that of relaxation labeling, which iteratively enhances the original image, progressively making decisions as to where the original gray levels reinforce each other. Thus, edge linking is permitted only where evidence is available in the original image that this is appropriate. A similar but computationally more efficient line of attack is the hysteresis thresholding method described in Chapter 5. Here intensity gradients above a certain upper threshold are taken to give definite indication of edge positions, whereas those above a second, lower threshold are taken to indicate edges only if they are adjacent to positions that have already been accepted as edges (for a more detailed analysis, see Section 5.10).

It may be thought that the Marr–Hildreth and related (Laplacian-based) edge detectors do not run into these problems, because they give edge contours that are necessarily connected. However, the result of using methods that force connectedness is that sometimes (e.g., when edges are diffuse, or of low contrast, so that image noise is an important factor) parts of a contour will lack meaning; indeed, a contour may meander over such regions following noise rather than useful object boundaries. Furthermore, it is as well to note that the problem is not merely one of pulling low-level signals out of noise, but rather that sometimes there is no signal at all present that could be enhanced to a meaningful level. Reasons for this include lighting being such as to give zero contrast (as, e.g., when a cube is lit in such a way that two faces have equal brightness) and occlusion. Lack of spatial resolution can also cause problems by merging together several lines on an object.

In what follows, it is assumed that all of these problems have been overcome by sufficient care with the lighting scheme, appropriate digitization, and other means. It is also assumed that suitable thinning and linking algorithms have been applied so that all objects are outlined by connected unit-width boundary lines. At this stage, it should be possible to locate the objects from the boundary image, and to identify and orientate them accurately.

10.2 BOUNDARY TRACKING PROCEDURES

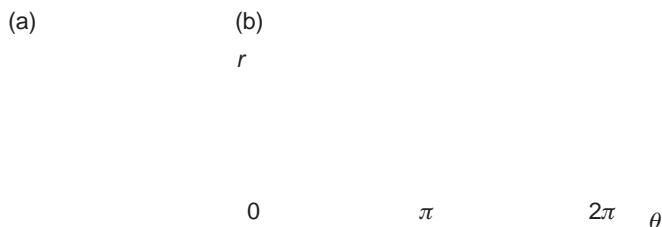
Before objects can be matched with their boundary patterns, means must be found for tracking systematically around the boundaries of all the objects in an image. Means have already been demonstrated for achieving this in the case of regions such as those that result from intensity thresholding routines (Chapter 9). However, if a connected unit-width boundary is formed by an alternative process such as edge detection, the problem of tracking is much simpler, since it is necessary only to move repeatedly to the next edge pixel. Clearly, it is necessary to ensure that we (a) never reverse direction, (b) know when we have been round the whole boundary once, and (c) record which object boundaries have been encountered. As when tracking around regions, we must ensure that in each case we end by passing through the starting point in the same direction.

10.3 CENTROIDAL PROFILES

The substantial matching problems that occur with 2-D template matching make it attractive to attempt to locate objects in a less demanding search space. In fact, it is possible to achieve this very simply by matching the boundary of each object in a single dimension. Perhaps the most obvious such scheme uses an (r, θ) plot. Here the centroid of the object is first located,¹ and then a polar coordinate system is set up relative to this point and the object boundary is plotted as an (r, θ) graph—often called a “centroidal profile” (Fig. 10.2). Next, the 1-D graph so obtained is matched against the corresponding graph for an idealized object of the same type. Since objects generally have arbitrary orientation, it is necessary to “slide” the idealized graph along that obtained from the image data until the best match is obtained. The match for each possible orientation α_j of the object is commonly tested by measuring the differences in radial distance between the boundary graph B and the template graph T for various values of θ and summing their squares to give a difference measure D_j for the quality of the fit:

$$D_j = \sum_i [r_B(\theta_i) - r_T(\theta_i + \alpha_j)]^2 \quad (10.1)$$

¹Note that the position of the centroid is deducible directly from the list of boundary pixel coordinates—there is no need to start with a region-based description of the object for this purpose.

**FIGURE 10.2**

Centroidal profiles for object recognition and scrutiny: (a) hexagonal nut shape in which one corner has been damaged; (b) centroidal profile, which permits both straightforward identification of the object and detailed scrutiny of its shape.

Alternatively, the absolute magnitudes of the differences are used:

$$D_j = \sum_i |r_B(\theta_i) - r_T(\theta_i + \alpha_j)| \quad (10.2)$$

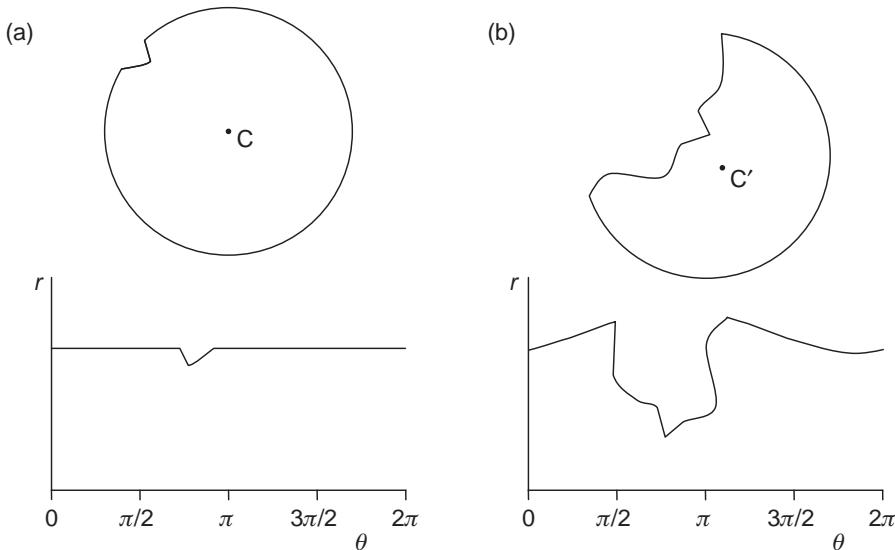
The latter measure has the advantage of being easier to compute and less biased by extreme or erroneous difference values. Note that the basic 2-D matching operation has now been reduced to 1-D, and if we need work in 1° steps, the orientation indices i and j will each have to range over 360 values. The result is that the number of operations that are required to test each object drops to around 360^2 (i.e., $\sim 100,000$), so boundary pattern analysis should give a very substantial saving in computation.

The 1-D boundary pattern matching approach described above is able to identify objects and also to find their orientations. In fact, initial location of the centroid of the object also solves one other part of the problem as specified at the end of [Section 10.1](#). At this stage, it may be noted that the matching process also leads to the possibility of inspecting the object's shape as an inherent part of the process ([Fig. 10.2](#)). In principle, this combination of capabilities makes the centroidal profile technique quite powerful.

Finally, note that the method is able to cope with objects of identical shapes but different sizes. This is achieved by using the maximum value of r to normalize the profile, giving a variation (ρ, θ) where $\rho = r/r_{\max}$.

10.4 PROBLEMS WITH THE CENTROIDAL PROFILE APPROACH

In practice, there are several problems with the procedure outlined in [Section 10.3](#). First, any major defect or occlusion of the object boundary can cause the centroid to be moved away from its true position, and the matching process will be largely spoiled ([Fig. 10.3](#)). Thus, instead of concluding that this is an object of type X with a specific part of its boundary damaged, the algorithm will most probably not recognize it at all. Such behavior would be inadequate in many

**FIGURE 10.3**

Problems with the centroidal profile descriptor. (a) A circular object with a minor defect on its boundary; its centroidal profile appears beneath it. (b) The same object, this time with a gross defect: because the centroid is shifted to C' , the *whole* of the centroidal profile is grossly distorted.

automated inspection applications, where positive identification and fault-finding are required, and the object would have to be rejected without a satisfactory diagnosis being made.

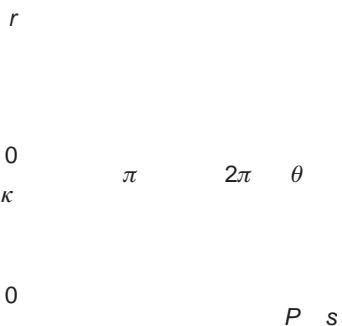
Second, the (r, θ) plot will be multivalued for a certain class of object (Fig. 10.4). This has the effect of making the matching process partly 2-D and leads to complication and excessive computation.

Third, the very variable spacing of the pixels when plotted in (r, θ) space is a source of complication. It leads to the requirement for considerable smoothing of the 1-D plots, especially in regions where the boundary comes close to the centroid—as for elongated objects such as spanners or screwdrivers (Fig. 10.5); however, in other places accuracy will be greater than necessary and the overall process will be wasteful. The problem arises because quantization should ideally be uniform along the θ -axis so that the two templates can be moved conveniently relative to one another to find the orientation of best match.

Finally, computation times can still be quite significant, so some timesaving procedure is required.

10.4.1 Some Solutions

All four of the above-described problems can be tackled in one way or another, with varying degrees of success. The first problem, that of coping with occlusions

**FIGURE 10.4**

Boundary pattern analysis via (r, θ) and (s, κ) plots.

FIGURE 10.5

A problem in obtaining a centroidal profile for elongated objects. This figure highlights the pixels around the boundary of an elongated object—a spanner—showing that it will be difficult to obtain an accurate centroidal profile for the region near the centroid.

and gross defects, is probably the most fundamental and the most resistant to satisfactory solution. For the method to work successfully, a stable reference point must be found within the object. The centroid is naturally a good candidate for this since the averaging inherent in its location tends to eliminate most forms of

noise or minor defect: however, major distortions such as those arising from breakages or occlusions are bound to affect it adversely. The centroid of the boundary is no better, and may also be less successful at suppressing noise. Other possible candidates are the positions of prominent features such as corners, holes, centers of arcs, and so on. In general, the smaller such a feature, the more likely it is to be missed altogether in the event of a breakage or occlusion, although the larger such a feature, the more likely it is to be affected by the defect. In fact, circular arcs can be located accurately (at their centers) even if they are partly occluded (see Chapter 12), so these features are very useful for leading to suitable reference points. A set of symmetrically placed holes may sometimes be suitable, since even if one of them is obscured, one of the others is likely to be visible and can act as a reference point.

Clearly, such features can help the method to work adequately but their presence also calls into question the value of the 1-D boundary pattern matching procedure, since they make it likely that superior methods can be used for object recognition (see later chapters in Part 2). For the present, we therefore accept that (a) severe complications arise when part of an object is missing or occluded and (b) it may be possible to provide some degree of resistance to such problems by using a prominent feature as a reference point instead of the centroid. Indeed, the only significant *further* change that is required to cope with occlusions is that difference ($r_B - r_T$) values greater than (say) 3 pixels should be ignored, and the best match then becomes one for which the greatest number of values of θ gives good agreement between B and T.

The second problem, of multivalued (r, θ) plots, is solved very simply by employing the heuristic of taking the smallest value of r for any given θ and then proceeding with matching as normal (here it is assumed that the boundaries of any holes present within the boundary of the object are dealt with separately, information about any object and its holes being collated at the end of the recognition process). This *ad hoc* procedure should in fact be acceptable when making a preliminary match of the object to its 1-D template, and may be discarded at a later stage when the orientation of the object is known accurately.

The third problem described above arises because of uneven spacing of the pixel boundaries along the θ dimension of the (r, θ) graph. To some extent this problem can be avoided by deciding in advance on the permissible values of θ and querying a list of boundary points to find which has the closest θ to each permissible value. Some local smoothing of the ordered set of boundary points can be undertaken but this is in principle unnecessary, since for a connected boundary, there will always be 1 pixel, which is closest to a line from the centroid at a given value of θ .

The two-stage approach to matching hinted in [Section 10.3](#) can also be used to help with the last of the problems mentioned above—the need to speed up the processing. First, a coarse match is obtained between the object and its 1-D template by taking θ in relatively large steps of (say) 5° and ignoring intermediate angles in both the image data and the template, and then a better match is

obtained by making fine adjustments to the orientations, obtaining a match to within 1° . In this way, the coarse match is obtained perhaps 20 times faster than the previous full match, whereas the final fine match takes a relatively short time, since very few distinct orientations have to be tested.

This two-stage process can be optimized by making a few simple calculations. The coarse match is given by increasing the θ incrementation step to $\delta\theta$, so the computational load is proportional to $(360/\delta\theta)^2$, whereas the load for the fine match is proportional to $360\delta\theta$, giving a total load of:

$$\lambda = \left(\frac{360}{\delta\theta} \right)^2 + 360\delta\theta \quad (10.3)$$

This should be compared with the original load of:

$$\lambda_0 = 360^2 \quad (10.4)$$

Hence, the load is reduced (and the algorithm speeded up) by the factor:

$$\eta = \frac{\lambda_0}{\lambda} = \frac{1}{(1/\delta\theta)^2 + (\delta\theta/360)} \quad (10.5)$$

This is a maximum for $d\eta/d\delta\theta = 0$, giving:

$$\delta\theta = \sqrt[3]{2 \times 360} \approx 9^\circ \quad (10.6)$$

In practice, this value of $\delta\theta$ is rather large and there is a risk that the coarse match will give such a poor fit that the object will not be identified: hence, values of $\delta\theta$ in the range $2\text{--}5^\circ$ are more usual (see, e.g., Berman et al., 1985). Note that the optimum value of η is 26.8 and this reduces only to 18.6 for $\delta\theta = 5^\circ$, although it goes down to 3.9 for $\delta\theta = 2^\circ$.

Another way of approaching the problem is to search the (r, θ) graph for some characteristic feature such as a sharp corner (this step constituting the coarse match), and then to perform a fine match around the object orientation so deduced. Clearly, there are possibilities of error here, in situations where objects have several similar features—as in the case of a rectangle: however, the individual trials are relatively inexpensive and so it is worth invoking this procedure if the object possesses appropriate well-defined features. Note that it is possible to use the position of the maximum value, r_{\max} , as an orientating feature, but this is frequently inappropriate because a smooth maximum gives a relatively large angular error.

10.5 THE (s, ψ) PLOT

It can be seen from the above considerations that boundary pattern analysis should usually be practicable except when problems from occlusions and gross defects can be expected. However, these latter problems do give motivation for

employing alternative methods if these can be found. In fact, the (s, ψ) graph has proved particularly popular since it is inherently better suited than the (r, θ) graph to situations where defects and occlusions may occur. In addition, it does not suffer from the multiple values encountered by the (r, θ) method.

The (s, ψ) graph does not require prior estimation of the centroid or some other reference point since it is computed directly from the boundary, in the form of a plot of the tangential orientation ψ as a function of boundary distance s . The method is not without its problems and, in particular, distance along the boundary needs to be measured accurately. The commonly used approach is to count horizontal and vertical steps as unit distance and to take diagonal steps as distance $\sqrt{2}$; in fact, this idea must be regarded as a rather *ad hoc* solution, and the situation is discussed further in [Section 10.7](#).

When considering application of the (s, ψ) graph for object recognition, it will immediately be noticed that the graph has a ψ value that increases by 2π for each circuit of the boundary, i.e., $\psi(s)$ is not periodic in s . The result is that the graph becomes essentially 2-D, i.e., the shape has to be matched by moving the ideal object template both along the s -axis and along the ψ -axis directions. Ideally, the template could be moved diagonally along the direction of the graph. However, noise and other deviations of the actual shape relative to the ideal shape mean that in practice the match must be at least partly 2-D, hence adding to the computational load.

One way of tackling this problem is to make a comparison with the shape of a circle of the same boundary length P . Thus, an $(s, \Delta\psi)$ graph is plotted, which reflects the difference $\Delta\psi$ between the ψ expected for the shape and that expected for a circle of circumference P :

$$\Delta\psi = \psi - \left(\frac{2\pi s}{P} \right) \quad (10.7)$$

This expression helps to keep the graph 1-D, since $\Delta\psi$ automatically resets itself to its initial value after one circuit of the boundary (i.e., $\Delta\psi$ is periodic in s).

Next note that the $\Delta\psi(s)$ variation depends on the starting position where $s = 0$ and this is randomly sited on the boundary. It is useful to eliminate this dependence, and this may be achieved by subtracting from $\Delta\psi$ its mean value μ . This gives the new variable:

$$\tilde{\psi} = \psi - \left(\frac{2\pi s}{P} \right) - \mu \quad (10.8)$$

At this stage, the graph is completely 1-D and is also periodic, being similar in these respects to an (r, θ) graph. Matching should now reduce to the straightforward task of sliding the template along the $\tilde{\psi}(s)$ graph until a good fit is achieved.

At this point, there should be no problems so long as (a) the scale of the object is known and (b) occlusions or other disturbances cannot occur. Suppose next that the scale is unknown: then the perimeter P may be used to normalize the value of s . If, however, occlusions *can* occur, then no reliance can be placed on P for

normalizing s and hence the method cannot be guaranteed to work. This problem does not arise if the scale of the object is known, since a standard perimeter P_T can be assumed. However, the possibility of occlusion gives further problems, which are discussed in the [Section 10.6](#).

Another way in which the problem of nonperiodic $\psi(s)$ can be solved is by replacing ψ by its derivative $d\psi/ds$. Then the problem of constantly expanding ψ (which results in its increase by 2π after each circuit of the boundary) is eliminated—the addition of 2π to ψ does not affect $d\psi/ds$ locally, since $d(\psi + 2\pi)/ds = d\psi/ds$. Note that $d\psi/ds$ is actually the local curvature function $\kappa(s)$ (see [Fig. 10.4](#)), so the resulting graph has a simple physical interpretation. Unfortunately, this version of the method has its own problems in that κ approaches infinity at any sharp corner. For industrial components, which frequently have sharp corners, this is a genuine practical difficulty and may be tackled by approximating adjacent gradients and ensuring that κ integrates to the correct value in the region of a corner (Hall, 1979).

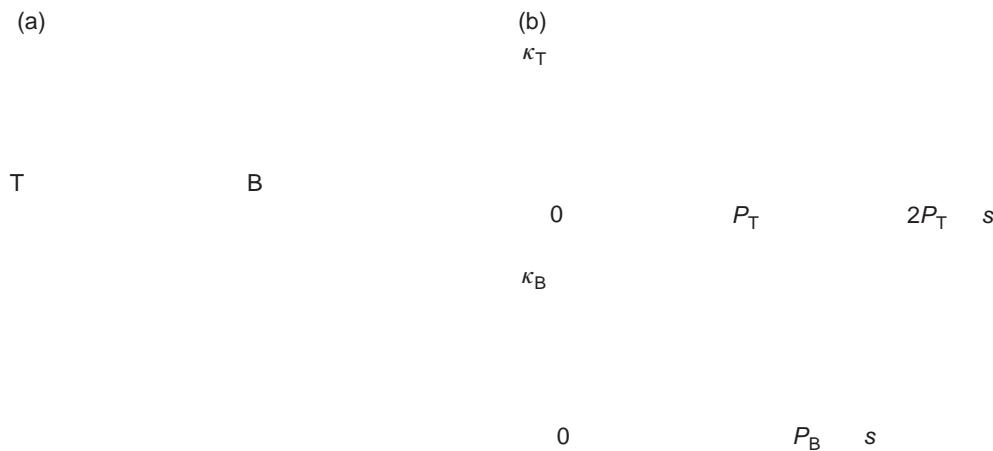
Many workers take the (s, κ) graph idea further and expand $\kappa(s)$ as a Fourier series:

$$\kappa(s) = \sum_{n=-\infty}^{\infty} c_n \exp\left(\frac{2\pi ins}{P}\right) \quad (10.9)$$

This results in the well-known Fourier descriptor method. In this method, shapes are analyzed in terms of a series of Fourier descriptor components, which are truncated to zero after a sufficient number of terms. Unfortunately, the amount of computation involved in this approach is considerable and there is a tendency to approximate curves with relatively few terms. In industrial applications, where computations have to be performed in real time, this can generate problems, so it is often more appropriate to match to the basic (s, κ) graph. In this way, critical measurements between object features can be made with adequate accuracy in real time.

10.6 TACKLING THE PROBLEMS OF OCCLUSION

Whatever means are used for tackling the problem of continuously increasing ψ , problems still arise when occlusions occur. However, the approach is not immediately invalidated by missing sections of boundary as it is for the basic (r, θ) method. As noted in [Section 10.5](#), a major effect of occlusions is that the perimeter of the object is altered, so P can no longer be used to indicate its scale. Hence, the latter has to be known in advance: this is assumed in what follows. Another practical result of occlusions is that certain sections correspond correctly to parts of the object, whereas other sections correspond to parts of occluding objects; alternatively, they may correspond to unpredictable boundary segments where damage has occurred. Note that if the overall boundary is that of two overlapping objects, the observed perimeter P_B will be greater than the ideal perimeter P_T .

**FIGURE 10.6**

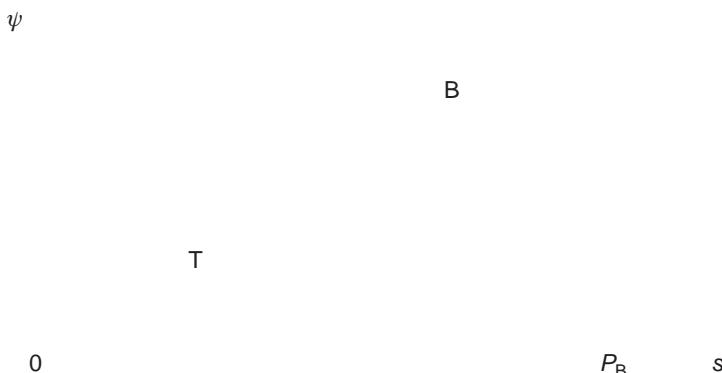
Matching a template against a distorted boundary. When a boundary B is broken (or partly occluded) but continuous, it is necessary to attempt to match between B and a template T that is doubled to length $2P_T$, to allow for T being severed at any point: (a) the basic problem and (b) matching in (s, κ) space.

Segmenting the boundary between relevant and irrelevant sections is, *a priori*, a difficult task. However, a useful strategy is to start by making positive matches wherever possible and to ignore irrelevant sections—i.e., try to match as usual, ignoring any section of the boundary that is a bad fit. We can imagine achieving a match by sliding the template T along the boundary B . However, a problem arises since T is periodic in s and should not be cut off at the ends of the range $0 \leq s \leq P_T$. As a result, it is necessary to attempt to match over a length $2P_T$. At first sight, it might be thought that the situation ought to be symmetrical between B and T . However, T is known in advance, whereas B is partly unknown, there being the possibility of one or more breaks in the ideal boundary into which foreign boundary segments have been included. Indeed, the positions of the breaks are unknown, so it is necessary to try matching the whole of T at all positions on B . Taking a length $2P_T$ in testing for a match effectively permits the required break to arise in T at any relevant position: (see Fig. 10.6).

When carrying out the match, we basically use the difference measure:

$$D_{jk} = \sum_i [\psi_B(s_i) - (\psi_T(s_i + s_k) + \alpha_j)]^2 \quad (10.10)$$

where j and k are the match parameters for orientation and boundary displacement, respectively. Note that the resulting D_{jk} is roughly proportional to the length L of the boundary over which the fit is reasonable. Unfortunately, this means that the measure D_{jk} appears to *improve* as L decreases; hence, when variable occlusions can occur, the best match must be taken as the one for which the greatest length L gives good agreement between B and T (this may be measured

**FIGURE 10.7**

Matching a short template to part of a boundary. A short template T , corresponding to part of an idealized boundary, is matched against the observed boundary B . Strictly speaking, matching in (s, ψ) space is 2-D, although there is very little uncertainty in the vertical (orientation) dimension.

as the greatest number of values of s in the sum of Eq. (10.10), which gives good agreement between B and T , i.e., the sum over all i such that the difference in square brackets is numerically less than, say, 5° .

If the boundary is occluded in more than one place, then L is at most the largest single length of unoccluded boundary (not the total length of unoccluded boundary), since the separate segments will in general be “out of phase” with the template. This is a disadvantage when trying to obtain an accurate result, since extraneous matches add noise, which degrades the fit that is obtainable—hence adding to the risk that the object will not be identified and reducing accuracy of registration. This suggests that it might be better to use only short sections of the boundary template for matching. Indeed, this strategy can be advantageous since speed is enhanced and registration accuracy can be retained or even improved by careful selection of salient features (note that nonsalient features such as smooth curved segments could have originated from many places on object boundaries and are not very helpful for identifying and accurately locating objects: hence, it is reasonable to ignore them). In this version of the method, we now have $P_T < P_B$, and it is necessary to match over a length P_T rather than $2P_T$, since T is no longer periodic (Fig. 10.7). Once various segments have been located, the boundary can be reassembled as closely as possible into its full form, and at that stage defects, occlusions, and other distortions can be recognized overtly and recorded. Reassembly of the object boundaries can be performed by techniques such as the Hough transform and relational pattern matching techniques (see Chapters 13 and 14). Work of this type has been carried out by Turney et al. (1985), who found that the salient features should be short boundary segments where corners and other distinctive “kinks” occur.

Before leaving this topic, note that $\tilde{\psi}$ can no longer be used when occlusions are present, since although the perimeter can be assumed to be known, the mean value of $\Delta\psi$ (Eq. (10.8)) cannot be deduced. Hence, the matching task reverts to

a 2-D search (although, as stated earlier, very little unrestrained search in the ψ direction need be made). However, in the case when small salient features are being sought, it is a reasonable working assumption that no occlusion occurs in any of the individual cases—a feature is either entirely present or entirely absent. Hence, the average slope $\bar{\psi}$ over T can validly be computed (Fig. 10.7) and this again reduces the search to 1-D (Turney et al., 1985).

Overall, missing sections of object boundaries necessitate a fundamental rethink as to how boundary pattern analysis should be carried out. For quite small defects the (r, θ) method is sufficiently robust but in less trivial cases it is vital to use some form of the (s, ψ) approach, while for really gross occlusions it is not particularly useful to try to match for the full boundary: rather it is better to attempt to match small salient features. This sets the scene for the Hough transform and relational pattern matching techniques of later chapters.

10.7 ACCURACY OF BOUNDARY LENGTH MEASURES

Next we examine the accuracy of the idea expressed earlier, that adjacent pixels on an 8-connected curve should be regarded as separated by 1 pixel if the vector joining them is aligned along the major axes and by $\sqrt{2}$ pixels if the vector is in a diagonal orientation. In general, this estimator overestimates the distance along the boundary. The reason for this is quite simple to see by appealing to the following pair of situations:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1								
...																												
										1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1								
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1								
...																												

In either case we are considering only the top of the object. In the first example, the boundary length along the top of the object is exactly that given by the rule. However, in the second case the estimated length is increased by amount $\sqrt{2} - 1$ because of the step. Now as the length of the top of the object tends to large values, say p pixels, the actual length approximates to p , whereas the estimated length is $p + \sqrt{2} - 1$; thus, a definite error exists. Indeed, this error initially increases in importance as p decreases, since the actual length of the top of the object (when there is one step) is still:

$$L = (1 + p^2)^{1/2} \approx p \quad (10.11)$$

so the fractional error is:

$$\xi \approx \frac{\sqrt{2} - 1}{p} \quad (10.12)$$

which increases as p becomes smaller.

This result can be construed as meaning that the fractional error ξ in estimating boundary length increases initially as the boundary orientation ψ increases from zero. A similar effect occurs as the orientation decreases from 45° . Thus, the ξ variation possesses a maximum between 0° and 45° . This systematic overestimation of boundary length may be eliminated by employing an improved model in which the length per pixel is s_m along the major axis directions and s_d in diagonal directions. A complete calculation (Kulpa, 1977; see also Dorst and Smeulders, 1987) shows that:

$$s_m = 0.948 \quad (10.13)$$

and

$$s_d = 1.343 \quad (10.14)$$

It is perhaps surprising that this solution corresponds to a ratio s_d/s_m that is still equal to $\sqrt{2}$, although the arguments given above make it obvious that s_m should be less than unity.

Unfortunately, an estimator that has just two free parameters can still permit quite large errors in estimating the perimeters of individual objects. To reduce this problem, it is necessary to perform more detailed modeling of the step pattern around the boundary (Koplowitz and Bruckstein, 1989), which seems certain to increase the computational load significantly.

It is important to underline that the basis of this work is to estimate the length of the original continuous boundary rather than that of the digitized boundary: furthermore, it must be noted that the digitization process loses information, so the best that can be done is to obtain the best estimate of the original boundary length. Thus, employing the values 0.948 and 1.343 given above, rather than the values 1 and $\sqrt{2}$, reduces the estimated errors in boundary length measurement from 6.6% to 2.3%—but then only under certain assumptions about correlations between orientations at neighboring boundary pixels (Dorst and Smeulders, 1987).

10.8 CONCLUDING REMARKS

This chapter is concerned with boundary pattern analysis. The boundary patterns were imagined to arise from edge detection operations that have been processed to make them connected and of unit width. However, if intensity thresholding methods were employed for segmenting images, boundary tracking procedures would also permit the boundary pattern analysis methods of this chapter to be used. Conversely, if edge detection operations led to the production of connected boundaries, these could be filled in by suitable algorithms (which are more tricky to devise than might at first be imagined) (Ali and Burge, 1988) and converted to regions to which the binary shape analysis methods of Chapter 9 could be applied. Hence, shapes are representable in region or boundary form: if they initially arise in one representation, they may be converted to the alternate representation. This means that boundary or regional means may be employed for shape analysis, as appropriate.

An important factor here is that a positive advantage is often gained by employing boundary pattern analysis, since computation should be inherently lower (in proportion to the numbers of pixels that are required to describe the shapes in the two representations). Another important determining factor that is discussed in the present chapter is that of occlusion. If occlusions are present, then several of the methods described in Chapter 9 will operate incorrectly—as also happens for the basic centroidal profile method described in [Section 10.3](#). The (s, ψ) method then provides a good starting point. As has been seen, this is best applied to detect small salient boundary features, which can then be reassembled into whole objects by relational pattern matching techniques (see especially Chapter 14).

A variety of boundary representations is available for shape analysis. However, this chapter has shown that intuitive schemes raise fundamental robustness issues: these will only be resolved later on by forgoing deduction in favor of inference. Underlying analog shape estimation in a digital lattice is also an issue.

10.9 BIBLIOGRAPHICAL AND HISTORICAL NOTES

Many of the techniques described in this chapter have been known since the early days of image analysis. Boundary tracking has been known since 1961 when Freeman introduced his chain code. Indeed, Freeman (1974) is responsible for much subsequent work in this area. Freeman (1978) introduced the notion of segmenting boundaries at “critical points” in order to facilitate matching: suitable critical points were corners (discontinuities in curvature), points of inflection, and curvature maxima. This work is clearly strongly related to that of Turney et al. (1985). Early work on Fourier boundary descriptors using the (r, θ) and (s, ψ) approaches was carried out by Rutovitz (1970), Barrow and Popplestone (1971), and Zahn and Roskies (1972); another notable paper in this area is by Persoon and Fu (1977). In an interesting development, Lin and Chellappa (1987) were able to classify partial (i.e., nonclosed) 2-D curves using Fourier descriptors.

At the beginning of the chapter it was noted that there are significant problems in obtaining a thin connected boundary for every object in an image. Since 1988, the concept of active contour models (or “snakes”) solved many of these problems. See Chapter 5 for an introduction to snakes and Chapter 22 for their application to vehicle location.

It is worth remarking on the increased attention to accuracy evident over the past 20–30 years: this is seen, e.g., in the length estimators for digitized boundaries discussed in [Section 10.7](#) (see Kulpa, 1977; Dorst and Smeulders, 1987; Beckers and Smeulders, 1989; Koplowitz and Bruckstein, 1989; Davies, 1991). For a later update on the topic, see Coeurjolly and Klette (2004).

In recent times, there has been an emphasis on characterizing and classifying *families* of shapes rather than just individual isolated shapes: (see in particular Cootes et al. (1992), Amit (2002), and Jacinto et al. (2003)). Klassen et al. (2004)

provide a further example of this in their analysis of planar (boundary) shapes using geodesic paths between the various shapes of the family. In their work, they employ the Surrey fish database (Mokhtarian et al., 1996). The same general idea is also manifest in the self-similarity analysis and matching approach of Geiger et al. (2003), which they used for human profile and hand motion analysis. Horng (2003) describes an adaptive smoothing approach for fitting digital planar curves with line segments and circular arcs. The motivation for this approach is to obtain significantly greater accuracy than can be achieved with the widely used polygonal approximation, yet with lower computational load than the spline fitting type of approach. It can also be imagined that any fine accuracy restriction imposed by a line plus circular arc model will have little relevance in a discrete lattice of pixels. da Gama Leitão and Stolfi (2002) have developed a multiscale contour-based method for matching and reassembling 2-D fragmented objects. Although this method is targeted at reassembly of pottery fragments in archeology, the authors imply that it is also likely to be of value in forensic science, in art conservation and in assessing the causes of failure of mechanical parts following fatigue and the like.

Two useful books are available that cover the subjects of shape and shape analysis in rather different ways: one is by Costa and Cesar (2000) and the other is by Mokhtarian and Bober (2003). The former is fairly general in coverage, but emphasizes Fourier methods, wavelets, and multiscale methods. The latter sets up a scale-space (especially curvature scale-space) representation (which is multiscale in nature), and develops the subject quite widely from there.

10.9.1 More Recent Developments

Ghosh and Petkov (2005) have described problems relating to the robust interpretation of incomplete object boundaries. They discuss the problems in relation to the ICR test—*viz.* assessing recognition rate performance as a function of the percentage of the contour retained, where deletions may occur either as segment deletions, or as occlusions, or as random pixel deletions. Experiments showed that occlusion was the most, and random pixel deletion the least, serious problem. Mori et al. (2005) considered problems relating to 3-D shape recognition from multiple 2-D views. They found that “shape contexts” were particularly important for efficient matching in such situations, shape contexts corresponding to representing shapes by a set of n samples on an object and examining the distribution of relative positions. This technique permitted shape matching to take place efficiently in two stages—fast pruning of possibilities followed by detailed matching.

10.10 PROBLEMS

1. Devise a program for finding a thinned (8-connected) boundary of an object in a binary image.

11

Line Detection

Detection of macroscopic features is an important part of image analysis and visual pattern recognition. Of particular interest is the identification of straight lines in images, as these are ubiquitous—appearing both on manufactured parts and in the built environment. The Hough transform (HT) provides the means for locating these features highly robustly in digital images. This chapter describes the processes and principles needed to achieve this.

Look out for:

- the basic HT technique for locating straight lines in images.
- alternative parametrizations of the HT.
- how lines can be localized along their length.
- how final line fitting can be made more accurate.
- why the HT is robust against noise and background clutter.
- how speed of processing can be improved.
- the RANSAC approach to line fitting.
- the relative efficiencies of RANSAC and the HT.
- how laparoscopic tools may be located.

While this chapter provides interesting methods for detecting line features in images, they may appear somewhat specialized. However, later chapters will show that both the HT and RANSAC (RANdom SAMpling Consensus) have much wider applicability than such arguments might suggest.

11.1 INTRODUCTION

Straight edges are among the most common features of the modern world, arising in perhaps the majority of manufactured objects and components, not least in the very buildings in which we live. Yet it is arguable whether true straight lines ever arise in the natural state: possibly the only example of their appearance in virgin

outdoor scenes is the horizon—although even this is clearly seen from space as a circular boundary. The surface of water is essentially planar, although it is important to realize that this is a deduction. The fact remains that straight lines seldom appear in completely natural scenes. Be all this as it may, it is clearly vital both in city pictures and in the factory to have effective means of detecting straight edges. This chapter studies available methods for locating these important features.

Historically, the Hough transform (HT) has been the main means of detecting straight edges, and since the method was originally invented (Hough, 1962) it has been developed and refined for this purpose. Hence this chapter concentrates on this particular technique; it also prepares the ground for applying the HT to the detection of circles, ellipses, corners, etc. in the next few chapters. We start by examining the original Hough scheme, even though it is now seen to be wasteful in computation, since important principles are involved. By the end of the chapter we shall see that the HT is not alone in its capabilities for line detection: RANSAC is also highly capable in this direction. In fact, both approaches have their advantages and limitations, as the discussion in [Section 11.6](#) will show.

11.2 APPLICATION OF THE HOUGH TRANSFORM TO LINE DETECTION

The basic concept involved in locating lines by the HT is point–line duality. A point P can be defined either as a pair of coordinates or in terms of the set of lines passing through it. The concept starts to make sense if we consider a set of collinear points P_i , then list the sets of lines passing through each of them, and finally note that there is just one line that is common to all these sets. Thus, it is possible to find the line containing all the points P_i merely by eliminating those that are not multiple hits. Indeed, it is easy to see that if a number of noise points Q_j are intermingled with the signal points P_i , the method will be able to discriminate the collinear points from among the noise points at the same time as finding the line containing them, merely by searching for multiple hits. Thus, the method is inherently robust against noise, as indeed it is in discriminating against currently unwanted signals such as circles.

In fact, the duality goes further. For just as a point can define (or be defined by) a set of lines, so a line can define (or be defined by) a set of points, as is obvious from the above argument. This makes the above approach to line detection a mathematically elegant one and it is perhaps surprising that the Hough detection scheme was first published as a patent (Hough, 1962) of an electronic apparatus for detecting the tracks of high-energy particles, rather than as a paper in a learned journal.

The form in which the method was originally applied involves parametrizing lines using the slope–intercept equation:

$$y = mx + c \quad (11.1)$$

Every point on a straight edge is then plotted as a line in (m, c) space corresponding to all the (m, c) values consistent with its coordinates, and lines are detected in this space. The embarrassment of unlimited ranges of the (m, c) values (near-vertical lines require near-infinite values of these parameters) is overcome by using two sets of plots, the first corresponding to slopes of less than 1.0 and the second to slopes of 1.0 or more; in the latter case, Eq. (11.1) is replaced by the form:

$$x = \tilde{m}x + \tilde{c} \quad (11.2)$$

where

$$\tilde{m} = \frac{1}{m} \quad (11.3)$$

The need for this rather wasteful device was removed by the Duda and Hart (1972) approach, which replaces the slope–intercept formulation with the so-called “normal” (θ, ρ) form for the straight line (see Fig. 11.1):

$$\rho = x \cos \theta + y \sin \theta \quad (11.4)$$

To apply the method using this form, the set of lines passing through each point P_i is represented as a set of sine curves in (θ, ρ) space: for example, for point $P_1(x_1, y_1)$, the sine curve has equation:

$$\rho = x_1 \cos \theta + y_1 \sin \theta \quad (11.5)$$

Then multiple hits in (θ, ρ) space indicate, via their (θ, ρ) values, the presence of lines in the original image.

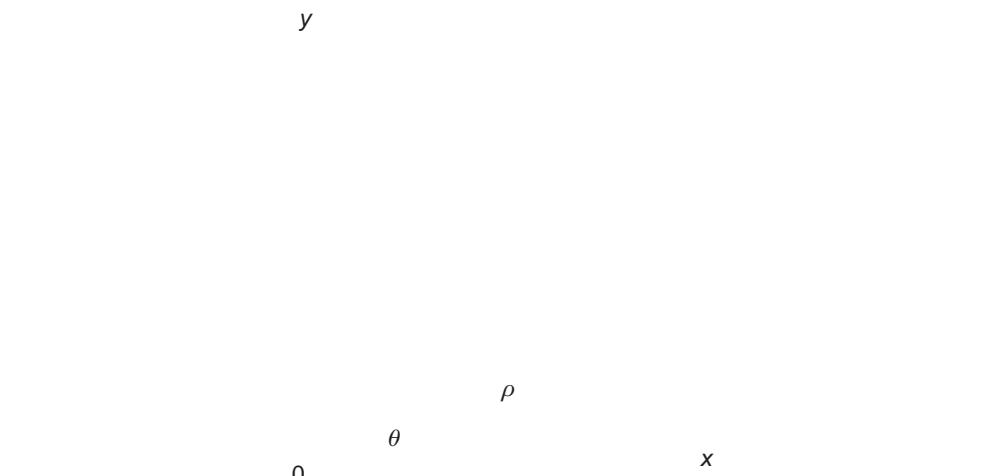


FIGURE 11.1

Normal (θ, ρ) parametrization of a straight line.

Each of the methods described above has the feature that it employs an “abstract” parameter space in which multiple hits are sought. Above we talked about “plotting” points in parameter space, but in fact the means of looking for hits is to seek peaks that have been built by *accumulation* of data from various sources. Although it might be possible to search for hits by logical operations such as use of the AND function, the Hough method gains considerably by *accumulating evidence* for events by a *voting scheme*. It will be seen below that this is the source of the method’s high degree of robustness.

Although the methods described above are mathematically elegant and are capable of detecting lines (or sets of collinear points—which may be completely isolated from each other) amid considerable interfering signals and noise, they are subject to considerable computational problems. The reason for this is that every prominent point¹ in the original image gives rise to a great many votes in parameter space, so for a 256×256 image the (m, c) parametrization requires 256 votes to be accumulated, while the (θ, ρ) parametrization requires a similar number—360 if the θ quantization is to be fine enough to resolve 1° changes in line orientation.

A vital key to overcoming this problem was discovered by O’Gorman and Clowes (1976), who noted that points on lines are usually not isolated but instead are joined in fragments that are sufficiently long that their directions can be measured. Supposing that direction is known with high accuracy, it is then sufficient to accumulate just one vote in parameter space for every potential line point (in fact, if the local gradient is known with lesser accuracy then parameter space can be quantized more coarsely—say in 10° steps (O’Gorman and Clowes, 1976)—and again a single vote per point can be accumulated). Clearly, this method is much more modest in its computational requirements and it was soon adopted by other workers.

However, the computational load is still quite substantial: not only is a large two-dimensional (2-D) storage area needed but this must be searched carefully for significant peaks—a tedious task if short line segments are being sought. Various means have been tried for cutting down computation further. Dudani and Luk (1978) tackled the problem by trying to separate out the θ and ρ estimations. They accumulated votes first in a 1-D parameter space for θ —i.e., a histogram of θ values (it must not be forgotten that such a histogram is itself a simple form of HT).² Having found suitable peaks in the θ histogram, they then built a ρ histogram for all the points that contributed votes to a given θ peak, and repeated this for all θ peaks. Thus, two 1-D spaces replace the original 2-D parameter space, with very significant savings in storage and load. However, two-stage methods of

¹For the present purpose it does not matter in what way these points are prominent. They may in fact be edge points, dark specks, centers of holes, and so on. Later we shall consistently take them to be edge points.

²It is now common for any process to be called an HT if it involves accumulating votes in a parameter space, with the intention of searching for significant peaks to find properties of the original data.

this type tend to be less accurate since the first stage is less selective: biased θ values may result from pairs of lines that would be well separated in a 2-D space. In addition, any error in estimating θ values is propagated to the ρ determination stage, making values of ρ even less accurate. For this reason Dudani and Luk added a final least-squares fitting stage to complete the accurate analysis of straight edges present in the image.

From a practical point of view, to proceed with either of the above methods of line detection, it is first necessary to obtain the local components of intensity gradient, and then to deduce the gradient magnitude g and threshold it to locate each edge pixel in the image. θ may be estimated using the arctan function in conjunction with the local edge gradient components g_x, g_y :

$$\theta = \arctan\left(\frac{g_y}{g_x}\right) \quad (11.6)$$

As the arctan function has period π , $\pm\pi$ may have to be added to obtain a principal value in the range $-\pi$ to $+\pi$: this can be decided from the signs of g_x and g_y .³ Once θ is known, ρ can be found from Eq. (11.4).

Finally, note that straight lines and straight edges are different and need to be detected differently. (Straight *edges* are probably more common and appear as object boundaries, whereas straight *lines* are typified by telephone wires in outdoor scenes.) In fact, we have concentrated above on using the HT to locate straight edges, starting with edge detectors. Straight line segments may be located using Laplacian-type operators and their orientations are defined over a range 0 – 180° rather than 0 – 360° : this makes HT design subtly different. For concreteness, in the remainder of this chapter, we concentrate on straight *edge* detection.

11.3 THE FOOT-OF-NORMAL METHOD

An alternative means of saving computation (Davies, 1986b) eliminates the use of trigonometric functions such as arctan by employing a different parametrization scheme. As noted earlier, the methods so far described all employ abstract parameter spaces in which points bear no immediately obvious visual relation to image space. In the alternative scheme, the parameter space is a second image space, which is congruent⁴ to image space.

This type of parameter space is obtained in the following way. First, each edge fragment in the image is produced much as required previously so that ρ can be measured, but this time the foot of the normal from the origin is itself taken as a voting

³Note that in C++, the basic arctan function is *atan*, with a single argument, which should be g_y/g_x used as indicated above. However, the C++ *atan2* function has two arguments, and if g_y and g_x are used respectively for these, the function automatically returns an angle in the range $-\pi$ to π .

⁴That is, parameter space is like image space, and each point in parameter space holds information that is immediately relevant to the corresponding point in image space.

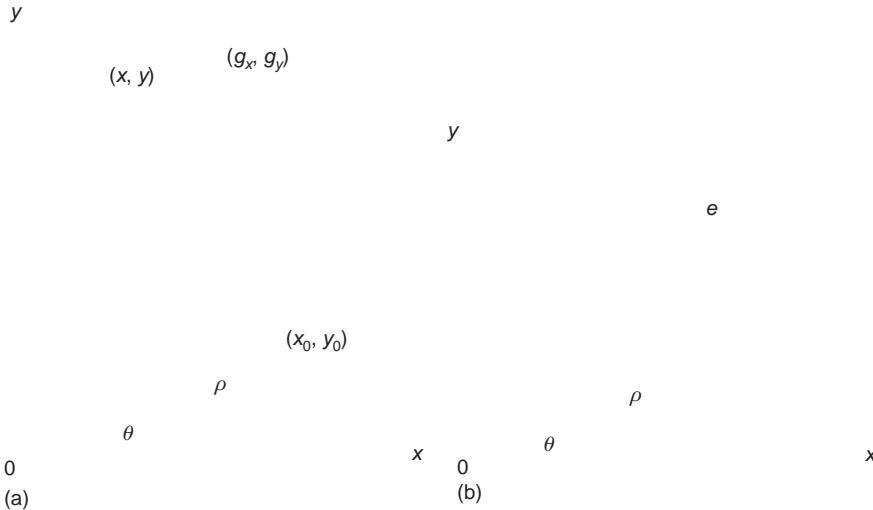

FIGURE 11.2

Image space parametrization of a straight line: (a) parameters involved in the calculation (see text); (b) buildup of foot-of-normal positions in parameter space for a more practical situation, where the line is not exactly straight: e is a typical edge fragment leading to a single vote in the parameter space.

Source: (b) © Unicom 1988

position in parameter space (Fig. 11.1). Clearly, the foot-of-normal position embodies all the information previously carried by the ρ and θ values, and mathematically the methods are essentially equivalent. However, the details differ, as will be seen.

The detection of straight edges starts with the analysis of (a) local pixel coordinates (x, y) and (b) the corresponding local components of intensity gradient (g_x, g_y) for each edge pixel. Taking (x_0, y_0) as the foot of the normal from the origin to the relevant line (produced if necessary—see Fig. 11.2), it is found that

$$\frac{g_y}{g_x} = \frac{y_0}{x_0} \quad (11.7)$$

$$(x - x_0)x_0 + (y - y_0)y_0 = 0 \quad (11.8)$$

These two equations are sufficient to compute the two coordinates (x_0, y_0) . Solving for x_0 and y_0 gives

$$x_0 = vg_x \quad (11.9)$$

$$y_0 = vg_y \quad (11.10)$$

where

$$v = \frac{xg_x + yg_y}{g_x^2 + g_y^2} \quad (11.11)$$

Note that these expressions involve only additions, multiplications, and just one division, so voting can be carried out efficiently using this formulation.

11.3.1 Application of the Foot-of-Normal Method

Although the foot-of-normal method is mathematically similar to the (θ, ρ) method, it is unable to determine line orientation directly with quite the same degree of accuracy. This is because the orientation accuracy depends on the fractional accuracy in determining ρ —which in turn depends on the absolute magnitude of ρ . Hence, for small ρ the orientation of a line that is predicted from the position of the peak in parameter space will be relatively inaccurate, even though the *position* of the foot-of-normal is known accurately. However, accurate values of line orientation can always be found by identifying the points that contributed to a given peak in the foot-of-normal parameter space and making them contribute to a θ histogram, from which line orientation may be determined accurately.

Typical results with the above method are shown in Fig. 11.3. Here, it was applied in subimages of size 64×64 within 128×128 images. Clearly, some of the objects in these pictures are grossly overdetermined by their straight edges, so low ρ values are not a major problem. For those peaks where $\rho > 10$, line orientation is estimated within approximately 2° ; as a result, these objects are located within 1 pixel and orientated within 1° by this technique, without the necessity for θ histograms. Figure 11.4(a) and (b) contain some line segments that are not detected. This is due partly to their relatively low contrast, higher noise levels, above average fuzziness, or short length. However, it is also due to the thresholds set on the initial edge detector and on the final peak detector: when these were set at lower values, additional lines were detected but other noise peaks also became prominent in parameter space, and each of these needed to be checked in detail to confirm the presence of the corresponding line in the image. This is one aspect of a general problem that arises right through the field of image analysis.

11.4 LONGITUDINAL LINE LOCALIZATION

The preceding sections have provided a variety of means for locating lines in digital images and finding their orientations. However, these methods are insensitive to where along an infinite idealized line an observed segment appears. The reason for this is that the fit includes only two parameters. There is some advantage to be gained in this, in that partial occlusion of a line does not prevent its detection. Indeed, if several segments of a line are visible, they can all contribute to the peak in parameter space, hence improving sensitivity. On the other hand, for full image interpretation, it is useful to have information about the longitudinal placement of line segments.

This is achieved by a further stage of processing. The additional stage involves finding which points contributed to each peak in the main parameter space and carrying out connectivity analysis in each case. Dudani and Luk (1978)

(a)

(b)

FIGURE 11.3

Results of image space parametrization of mechanical parts. The dot at the center of each quadrant is the origin used for computing the image–space transform. The crosses are the positions of peaks in parameter space that mark the individual straight edges (produced if necessary). For further explanation, see text.

Source: (a) © Unicom 1988

called this process “xy–grouping.” It is not vital that the line segments should be 4- or 8-connected—just that there should be sufficient points on them so that adjacent points are within a threshold distance apart, i.e., groups of points are merged if they are within the prespecified distance (typically, 5 pixels). Finally, segments shorter than a certain minimum length (also typically ~ 5 pixels) can be ignored as too insignificant to help with image interpretation.

**FIGURE 11.4**

Straight line location using the RANSAC technique. (a) An original grayscale image with various straight edges located using the RANSAC technique; (b) the edge points fed to RANSAC for (a). These were isolated points that were local maxima of the gradient image. (c) The straight edges of a pair of laparoscopic tools—a cutter and a gripper—which have been located by RANSAC. (d) The points fed to RANSAC for (c). In (a) three edges of the icosahedron are missed. This is because they are roof edges with low-contrast and low-intensity gradient. RANSAC missed a fourth edge because of a lower limit placed on the level of support (see text).

11.5 FINAL LINE FITTING

Dudani and Luk (1978) found it useful to include a least-squares fitting stage to complete the analysis of the straight edges present in an image. This is carried out by taking the x , y coordinates of points on the line as input data and minimizing the sum of squares of distances from the data points to the best fit line. Their reason for adding such a stage was to eliminate local edge orientation accuracy as

a source of error. This motivation is generally reasonable if the highest possible accuracy is required (e.g., obtaining line orientation to significantly better than 1°). However, many workers have criticized the use of the least-squares technique, since it tends to weight up the contribution of less accurate points—including those points that have nothing to do with the line in question, but which arise from image “clutter.” This criticism is probably justified, although the least-squares technique is convenient in yielding to mathematical analysis and in this case in giving explicit equations for θ and ρ in terms of the input data (Dudani and Luk, 1978). Dudani and Luk obtained the endpoints by reference to the best-fit line thus obtained. To understand the limitations of the least-squares technique, see the Appendix A.

11.6 USING RANSAC FOR STRAIGHT LINE DETECTION

RANSAC is an alternative model-based search schema that can often be used instead of the HT. In fact, it is highly effective when used for line detection, which is why the method is introduced here. The strategy can be construed as a voting scheme, but it is used in a different way from that in the HT. The latter operates by building up the evidence for instances of target objects in the form of votes in parameter space, and then making decisions about their existence (or by making hypotheses about their existence that are then finally checked out). RANSAC operates by making a sequence of hypotheses about the target objects, and determines the support for each of them by counting how many data points agree with them. As might be expected, for any potential target object, only the hypotheses with the maximum support are retained at each stage. This results in more compact information storage than for the HT: i.e., for RANSAC a list of hypotheses is held in current memory, whereas for the HT a whole parameter space, which is usually only sparsely populated, is held in memory. Thus, the RANSAC data are abstract lists, whereas the HT data can often be viewed as pictures in parameter space—as in the case of the foot-of-normal line detector. None of this prevents the RANSAC output being displayed (e.g., as straight lines) in image space, nor does it prevent the HT being accumulated using a list representation.

To explain RANSAC in more detail, we take the case of line detection. As for the HT, we start by applying an edge detector and locating all the edge points in the image. As we shall see, RANSAC operates best with a limited number of points, so it is useful to find the edge points that are local maxima of the intensity gradient image. (This does not correspond to the type of nonmaximum suppression used in the Canny operator, which produces thin connected *strings* of edge points but to individual *isolated* points: we shall return to this point below.) Next, to form a straight line hypothesis, all that is necessary is to take any pair of edge points from the list of N that remain after applying the local maximum operation. For each hypothesis we run through the list of N points finding how many points M support the hypothesis. Then we take more hypotheses (more pairs of edge

Table 11.1 Basic RANSAC Algorithm for Finding the Line with Greatest Support

```

Mmax = 0;
for all pairs of edge points do {
    find equation of line defined by the two points i, j;
    M = 0;
    for all N points in list do
        if (point k is within threshold distance d of line) M++;
    if (M > Mmax) {
        Mmax = M;
        imax = i;
        jmax = j;
        //this records the hypothesis giving the maximum support so far
    }
}
/* if Mmax > 0, (x[imax], y[imax]) and (x[jmax], y[jmax]) will be the coordinates of
the points defining the line having greatest support */

```

This algorithm only returns one line: in fact it returns the specific line model that has greatest support, for the line that has greatest support. Lines with less support are in the end ignored.

points) and at each stage retain only the one giving maximum support M_{\max} . This process is shown in [Table 11.1](#).

The algorithm in [Table 11.1](#) corresponds to finding the center of the highest peak in parameter space in the case of the HT. To find all the lines in the image, the most obvious strategy is the following: find the first line, then eliminate all the points that gave it support; then find the next line and eliminate all the points that gave it support; and so on until all the points have been eliminated from the list. The process may be written more compactly in the form:

```

repeat {
    find line;
    eliminate support;
}
until no data points remain;

```

Such a strategy carries the problem that if lines cross each other, support for the second line (which will necessarily be the weaker one) could be lesser support than it deserves. However, this should only be a serious disadvantage if the image is severely cluttered with lines. Nevertheless, the process is sequential and as such the results (i.e., the exact line locations) will depend on the order in which lines are eliminated, as the support regions will be minutely altered at each stage. Overall, the interpretation of complex images almost certainly has to proceed sequentially, and there is significant evidence that the human eye–brain system interprets images in this way, following early cues in order to progressively make sense of the data. Interestingly, the HT seems to escape from this by the potential capability for *parallel*

identification of peaks. While for simple images this may well be true, for complicated images containing many overlapping edges, there will again be the need for sequential analysis of the type envisaged above (e.g., see the back-projection method of Gerig and Klein, described in Section 13.11). The point is that with the particular list representation employed by RANSAC, we are *immediately* confronted with the problem of how to identify multiple targets, whereas for the reasons given above this does not immediately happen with the HT. Finally, on the plus side, successive elimination of support points necessarily makes it progressively easier and less computation intensive to find subsequent target objects. But the process itself is not cost-free, as the whole RANSAC procedure in Table 11.1 has to be run twice per line in order to identify the support points that have to be eliminated.

Next, we consider the computational load of the RANSAC process. If there are N edge points, the number of potential lines will be ${}^N C_2$, corresponding to a computational load of $O(N^2)$. However, finding the support for each line will involve $O(N)$ operations, so the overall computational load will be $O(N^3)$. In addition, the need to eliminate the support points for each line found will require computation proportional to the number of lines n , amounting to only $O(nN)$, and this will have little effect on the overall computational load.

One point that has not yet been made is that all N edge points will not arise from straight lines: some will arise from lines, some from curves, some from general background clutter, and some from noise. To limit the number of false positives, it will be useful to set a support threshold M_{thr} such that potential lines for which $M > M_{\text{thr}}$ are most likely to be true straight lines, while others are most likely to be artifacts, such as parts of curves or noise points. Thus, the RANSAC procedure can be terminated when M_{max} drops below M_{thr} . Of course, it may be required to retain only “significant” lines, e.g., those having length greater than L pixels. In that case, analysis of each line could allow many more points to be eliminated as the RANSAC algorithm proceeds. Another factor is whether hypotheses corresponding to pairs whose points are too close together should be taken into account. In particular, it might be considered that points closer together than 5 pixels would be superfluous as they would be likely to have much reduced chance of pointing along the direction of a line. However, it turns out that RANSAC is fail-safe in this respect, and there is some gain from keeping pairs with quite small separations, as some of the resulting hypotheses can actually be more accurate than any others. Overall, restricting pairs by their separations can be a useful way of reducing computational load, bearing in mind that $O(N^3)$ is rather high. Here, we should recall that the load for the HT is $O(N^2)$ during voting, if pairs of points are used, or $O(N)$, if single edge points and their gradients are used instead.

As we have just seen, RANSAC does not compare well with the HT regarding computational load, so it is better to employ RANSAC when N can be reduced in some way. This is why it is useful to use N local maxima rather than a full list of edge points in the form of strings of edge points generated by nonmaximum suppression or *a fortiori*, those existing before nonmaximum suppression. Indeed,

there is much to be gained by repeated random sampling⁵ from the full list until sufficient hypotheses have been tested to be confident that all significant lines have been detected. In this way, the computational load is reduced from $O(N^3)$ to $O(N^2)$ or even $O(N)$ (it is difficult to predict the resulting computational complexity: in any case, the achievable computational load will be highly data dependent). Confidence that all significant lines have been detected can be obtained by estimating the risk that a significant line will be missed because no representative pair of points lying on the line was considered. This aspect of the problem will be examined more fully in Section A.6 of the appendix on Robust Statistics.

Before proceeding further, we shall briefly consider another way of reducing computational load. That is, by using the connected strings of edge points resulting from nonmaximum suppression, but eliminating those that are very short and coding the longer ones as isolated points every p pixels, where $p \approx 10$. In this way, there would be far fewer points than for our earlier paradigm, and also those that are employed would have increased coherence and probability of lying on straight edges; hence there would be a concentration on high quality data, while the $O(N^3)$ computation factor would be dramatically reduced. Clearly, in high noise situations this would not work well. It is left to the reader to judge how well it would work for the sets of edges located by the Canny operator in Figs. 5.7 and 5.8.

We are now in a position to consider actual results obtained by applying RANSAC to straight line detection. In the tests described, pairs of points were employed as hypotheses, and all edge points were local maxima of the intensity gradient. The cases shown in Fig. 11.4 correspond to detection of a block of wood in the shape of an icosahedron, and a pair of laparoscopic tools with parallel sides. Note that one line on the right of Fig. 11.4(a) was missed because a lower limit had to be placed on the level of support for each line. This was necessary because below this level of support the number of chance collinearities rose dramatically even for the relatively small number of edge points shown in Fig. 11.4(b), leading to a sharp rise in the number of false-positive lines. Figs. 23.2 and 23.3 show RANSAC being used to locate road lane markings. The same version of RANSAC was used in all the above cases, albeit in the case of Fig. 23.3 a refinement was added to allow improved elimination of points on lines that had already been located (this point will be clarified at the end of this section). Overall, this set of examples shows that RANSAC is a highly important contender for location of straight lines in digital images. Not discussed here is the fact that RANSAC is useful for obtaining robust fits to many other types of shape in 2-D and in 3-D.

It should be mentioned that one characteristic of RANSAC is that it is less influenced by aliasing along straight lines than the HT. This is because HT peaks tend to be fragmented by aliasing, so the best hypotheses can be difficult to obtain

⁵This corresponds to the original reason for the term RANSAC, which stands for RANdom SAMpling Consensus, “consensus” indicating that any hypothesis has to form a consensus with the available support data.

without excessive smoothing of the image. The reason why RANSAC wins in this context is that it does not rely on individual hypotheses being accurate: rather it relies on enough hypotheses easily being generatable, and by the same token, discardable.

Finally, we return to the above comment about obtaining improved deletion of points on lines that have already been located. Suppose the cross-section of a line is characterized by a (lateral) Gaussian distribution of edge points. As a true Gaussian extends to infinity in either direction, the support region is not well defined, but for high accuracy it is reasonable to take it as the region within $\pm\sigma$ of the centerline of the line. However, if just these points are eliminated, the remaining points near the line could later on give rise to alternative lines or combine with other points to lead to false positives. It is therefore better (Mastorakis and Davies, 2011) to make the “delete distance” d_d larger than the “fit distance” d_f that is used for support during detection, e.g., $d_d = 2\sigma$ or 3σ , where $d_f = \sigma$. ($d_d \approx 3\sigma$ can be considered to be close to optimal because 99.9% of the samples in a Gaussian distribution lie within $\pm 3\sigma$.) Figure 23.3 shows instances in which the fit distance is 3 pixels and the delete distance has values of 3, 6, 10, and 11 pixels, showing the advantages that can be gained by making d_d significantly greater than d_f . Figure 23.4 shows a flowchart of the algorithm used in this case.

11.7 LOCATION OF LAPAROSCOPIC TOOLS

Section 11.6 showed how RANSAC can provide a highly efficient means for locating straight edges in digital pictures, and gave an example of its use to locate the handles of laparoscopic tools. These are used for various forms of “keyhole” surgery: specifically, one tool (e.g., a cutter) might be inserted through one incision and another (e.g., a gripper) through a second incision. Additional incisions are needed for viewing *via* a laparoscope that employs optical fiber technology, and for inflating the cavity—e.g., the abdominal or chest cavity. In this section, we consider what information can be obtained *via* the laparoscope.

Figure 11.4(c) shows laparoscopic gripper and cutter tools being located in a simulated flesh background. The latter will normally be a wet surface that is largely red and will exhibit many regions that are close to being specularly reflective. The large variations in intensity that occur under these conditions make the scene quite difficult to interpret. While the surgeon who is in control of the instruments can learn a lot about the scene through tactile feedback and thus bolster his understanding of it, other people viewing the scene, e.g., on a TV monitor or computer, are liable to find it highly confusing. The same will apply for any computer attempting to interpret, analyze, or record the progress of an operation. These latter tasks are potentially important for logging operations, for training other doctors, for communicating with specialists elsewhere, or for analyzing the progress of the operation during any subsequent debriefing. It would therefore be useful if the exact locations,

FIGURE 11.5

Tips of laparoscopic tools located from the parts highlighted in gray.

orientations, and other parameters of the tools could be determined at least relative to the frame of reference of the laparoscope. To this end, RANSAC has provided important 2-D data on the location of the tool handles. Assessing the vanishing points of the pairs of lines from the handles also provides 3-D information. Clearly, further information can be obtained from the coordinates of the ends of the tools.

To identify the ends of the tools, the ends of the handles are first located. This is a straightforward task requiring knowledge of the exact ends of the RANSAC support regions for the tool handle edges. The remainder of the tool ends can now be located by initial approximate prediction, adaptive thresholding, and connected components analysis (Fig. 11.5), special attention being paid to accurate location of the tips of the tool ends. In Fig. 11.5, this is achieved within ~ 1 pixel for the gripper on the left, and slightly less accurately for the closed cutter on the right. If the gripper had been open, accuracy would have been similar to that for the gripper. Note that, because of the complex intensity patterns in the background, it would have been difficult to locate the tool ends without first identifying the tool handles.

Each of the laparoscopic tools referred to above has (X, Y, Z) position coordinates, together with rotations ψ within the image plane (x, y) , θ away from the image plane (toward the Z -axis), and φ about the axis of the handle; in addition, each tool end has an opening angle α (Fig. 11.6). It is bound to be difficult to obtain all seven parameters with any great accuracy from a single monocular view. However, in principle, using an exact CAD model of the tool end, this should be possible with $\sim 15\text{--}20^\circ$ accuracy for the angles. The 2-D information about the centerlines and widths of the handles, the convergence of the handle edges, and the exact positions of the tips of the tools, should together permit such a 3-D analysis to be carried out. Here, we have concentrated on the 2-D analysis: details of the relevant 3-D background theory needed to proceed further can be found in Part 3.

**FIGURE 11.6**

Orientation parameters for a laparoscopic tool. (a) A gripper tool with closed jaws. (b) Gripper with jaws separated by an angle α . (c) Gripper rotated through an angle φ about a horizontal axis. (d) Gripper tipped through an angle θ away from the image plane. (e) Gripper rotated through an angle ψ about the camera optical axis.

11.8 CONCLUDING REMARKS

This chapter has described a variety of techniques for finding straight lines and straight edges in digital images. Several of these were based on the HT, which is important because it permits systematic extraction of global data from images and has the capability to ignore “local” problems, due for example to occlusions and noise. This is what is required for “intermediate-level” processing, as will be seen repeatedly in later chapters.

The specific techniques covered have involved various parametrizations of a straight line, and means for improving efficiency and accuracy. In particular, speed is improved by using a two-stage line finding procedure—a method that is useful in other applications of the HT, as will be seen in later chapters. Accuracy tends to be cut down by such two-stage processing because of error propagation and because the first stage is liable to be subject to too many interfering signals. However, it is possible to improve the accuracy of approximate solutions by using least-squares refinement procedures.

In fact, by the end of the chapter it became clear that the RANSAC approach also has line fitting capabilities, which are for some purposes superior to those of the HT, although RANSAC tends to be more computation intensive (with N edge points it has a computational load of $O(N^3)$ rather than $O(N^2)$). Suffice it to say that the choice of which approach to use will depend on the exact type of image data, including levels of noise and background clutter.

The Hough transform is one way of inferring the presence of objects from their feature points, and RANSAC is another. Both methods can be said to use voting schemes to select best-fit lines, although only the HT employs a parameter space representation; and both methods are highly robust as they focus only on positive evidence for the existence of objects.

11.9 BIBLIOGRAPHICAL AND HISTORICAL NOTES

The HT was developed in 1962 (Hough, 1962) with the aim of finding (straight) particle tracks in high-energy nuclear physics and was brought into the mainstream image analysis literature much later by Rosenfeld (1969). Duda and Hart (1972) developed the method further and applied it to the detection of lines and curves in digital pictures. O’Gorman and Clowes (1976) soon developed a Hough-based scheme for finding lines efficiently, by making use of edge orientation information, at much the same time that Kimme et al. (1975) applied the same method (apparently independently) to the efficient location of circles. Many of the ideas for fast effective line finding described in this chapter arose in a paper by Dudani and Luk (1978). The author’s foot-of-normal method (Davies, 1986b) was developed much later. During the 1990s, work in this area progressed further—see for example Atiquzzaman and Akhtar’s (1994) method for the efficient determination of lines together with their end coordinates and lengths; Lutton et al.’s (1994) application of the transform to the determination of vanishing points; and Kamat-Sadekar and Ganesan’s (1998) extensions of the technique to cover the reliable detection of multiple line segments, particularly in relation to road scene analysis. Other applications of the HT are covered in the next two chapters.

Some mention should be made of the related Radon transform. This is formed by integrating the picture function $I(x, y)$ along infinitely thin straight strips of the image, with normal coordinate parameters (θ, ρ) , and recording the results in a (θ, ρ) parameter space. The Radon transform is a generalization of the Hough transform for line detection (Deans, 1981). In fact, for straight lines the Radon transform reduces to the Duda and Hart (1972) form of the HT that, as remarked earlier, involves considerable computation. For this reason the Radon transform is not covered in depth in this book. The transforms of real lines have a characteristic “butterfly” shape (a pencil of segments passing through the corresponding peak) in parameter space. This phenomenon has been investigated by Leavers and Boyce (1987), who have devised special 3×3 convolution filters for sensitively detecting these peaks.

Contrary to the view of some that the HT is completely worked over and no longer a worthwhile topic for research, there has been strong continuing interest in it. Indeed, the computational difficulties of the method reflect underlying matching problems that are inescapable in computer vision, so development of methods must continue. Thus, Schaffalitsky and Zisserman (2000) carried out an interesting extension of earlier ideas on vanishing lines and points by considering the case of repeated lines, such as those occurring on certain types of fences and

brick buildings; Song et al. (2002) developed HT methods for coping with the problems of fuzzy edges and noise in large-sized images; and Guru et al. (2004) demonstrated viable alternatives to the HT, based for example on heuristic search achieved by small eigenvalue analysis.

11.9.1 More Recent Developments

In 2010, advances were still being made in the application of the HT. In particular, Chung et al. (2010) developed an orientation-based elimination strategy that they have shown to be more efficient than previous line-determination methods based on the HT. It operates by dividing edge pixels into sets with small (typically 10°) ranges of orientation, and for each of these, it carries out the process of line detection. Since this process involves a parameter space of reduced size, both storage and search times are reduced.

The RANSAC procedure was published by Fischler and Bolles in 1981. This must be one of the most cited papers in computer vision, and the method must be one of the most used (more even than the HT, because it only relies on the existence of suitable hypotheses, *however* obtained). The original paper used it for tackling the full perspective n -point fitting problem in 3-D (see Chapter 16). Clarke et al. (1996) used it for locating and tracking straight lines. Borkar et al. (2009) used it for locating lane markings on roads, and Mastorakis and Davies (2011) developed it further for the same purpose. Interestingly, Borkar et al. used a low-resolution HT to feed RANSAC and followed it by least-squares fitting of the inliers. The paper does not report on how much was gained by this three-stage approach, either in accuracy or in reliability. (If enough hypotheses are employed—and there is certainly no lack of these in such an application—both the HT and least-squares fitting might be avoided, but here optimization for speed may make the inclusion of least squares essential.) For further discussion of RANSAC, see Chapter 23 and Appendix A.

11.10 PROBLEMS

1.
 - a. In the foot-of-normal HT, straight edges are found by locating the foot of the normal $F(x_f, y_f)$ from an origin $O(0, 0)$ at the center of the image to an extended line containing each edge fragment $E(x, y)$, and placing a vote at F in a separate image space.
 - b. By examining the possible positions of lines within a square image and the resulting foot-of-normal positions, determine the exact extent of the parameter space that is theoretically required to form the HT.
 - c. Would this form of the HT be expected to be (i) more or less robust and (ii) more or less computation intensive than the (ρ, θ) HT for line location?
2.
 - a. Why is it sometimes stated that a HT generates *hypotheses* rather than actual solutions for object location? Is this statement justified?

- b. A new type of HT is to be devised for detecting straight lines. It will take every edge fragment in the image and extend it in either direction until it meets the boundary of the image, and then accumulate a vote at each position. Thus *two* peaks should result for every line. Explain why finding these peaks will require *less* computation than for the standard HT, but that deducing the presence of lines will then require *extra* computation. How will these amounts of computation be likely to vary with (i) the size of the image and (ii) the number of lines in the image?
- c. Discuss whether this approach would be an improvement on the standard approach for straight line location, and whether it would have any disadvantages.
3. a. Describe the *Hough transform* approach to object location. Explain its advantages relative to the *centroidal* (r, θ) *plot* approach. Illustrate your answer with reference to location of circles of known radius R .
- b. Describe how the Hough transform may be used to locate straight edges. Explain what is seen in the parameter space if many curved edges also appear in the original image.
- c. Explain what happens if the image contains a square object of arbitrary size and *nothing* else. How would you deduce from the information in parameter space that a square object is present in the image? Give the main features of an algorithm to decide that a square object is present and to locate it.
- d. Examine in detail whether an algorithm using the strategy described in (c) would become confused if (i) parts of some sides of the square were occluded; (ii) one or more sides of the square were missing; (iii) several squares appeared in the image; (iv) several of these complications occurred together.
- e. How important is it to this type of algorithm to have edge detectors that are capable of accurately determining edge orientation? Describe a type of edge detector that is capable of achieving this.

Circle and Ellipse Detection

12

As in the case of straight lines, circle features occur very widely in digital images of manufactured objects. In fact, they can conveniently be located using the Hough transform approach. This also applies for ellipses, which may appear in their own right or as oblique views of circular objects. However, locating ellipses is more complicated than locating circles because of the greater number of parameters that are involved.

Look out for:

- the basic Hough transform technique for locating circular objects in images.
- how the method can be adapted when circle radius is unknown.
- how accuracy of center location can be improved.
- how speed of processing can be increased.
- the basic diameter bisection method for ellipse detection.
- the chord–tangent method for ellipse detection.
- means of testing a shape to confirm that it is an ellipse.
- how small holes may be detected.
- how the human iris may be located.

This chapter shows how the Hough transform is able to provide a useful means for detecting objects of selected shape in digital images. Again the method is seen to rely on the accumulation of votes in a parameter space and the robustness of the technique to result from concentration on *positive* evidence for the objects.

12.1 INTRODUCTION

Location of round objects is important in many areas of image analysis but it is especially important in industrial applications such as automatic inspection and assembly. In the food industry alone, a very sizable number of products are round—biscuits, cakes, pizzas, pies, jellies, oranges, and so on (Davies, 1984c). In the automotive industry, many circular components are used—washers, wheels, pistons, heads of bolts, and so on, while round holes of various sizes appear in such items as casings and cylinder blocks. In addition, buttons and many other everyday objects are round. Of course, when round objects are viewed obliquely, they appear elliptical; furthermore, certain other objects are *actually* elliptical. This makes it clear that we need algorithms that are capable of finding both circles and ellipses. Finally, objects can frequently be located by their holes, so finding round holes or features is part of the larger problem: this chapter addresses various aspects of this problem.

An important facet of this work is how well object location algorithms cope in the presence of artifacts such as shadows and noise. In particular, the paradigm represented by [Table 12.1](#) has been shown in Chapter 10 to be insufficiently robust to cope in such situations. This chapter shows that the Hough transform (HT) technique is able to overcome many of these problems. Indeed, it is found to be particularly good at dealing with all sorts of difficulties, including quite severe occlusions. It achieves this not by adding robustness but by having robustness built in as an integral part of the technique.

The application of the HT to circle detection is one of the most straightforward uses of the technique. However, there are several enhancements and adaptations that can be applied in order to improve accuracy and speed of operation, and in addition to make the method work efficiently when detecting circles with a range of sizes. These modifications are studied after covering the basic HT technique. Versions of the Hough transform that can perform ellipse detection are then considered. Finally, after a short section on an important application—that of human iris location—the topic of hole detection is considered.

Table 12.1 Procedure for Finding Objects Using (r, θ) Boundary Graphs

1. Locate edges within the image
2. Link broken edges
3. Thin thick edges
4. Track around object outlines
5. Generate a set of (r, θ) plots
6. Match (r, θ) plots to standard templates

This procedure is not sufficiently robust with many types of real data, e.g., in the presence of noise, distortions in product shape, and so on: in fact, it is quite common to find the tracking procedure veering off and tracking around shadows or other artifacts.

12.2 HOUGH-BASED SCHEMES FOR CIRCULAR OBJECT DETECTION

In the original HT method for finding circles (Duda and Hart, 1972), the intensity gradient is first estimated at all locations in the image and then thresholded to give the positions of significant edges. Then the positions of all possible center locations—namely all points a distance R away from every edge pixel—are accumulated in parameter space, R being the anticipated circle radius. Parameter space can be a general storage area but when looking for circles it is convenient to make it congruent to image space: in that case, possible circle centers are accumulated in a new plane of image space. Finally, parameter space is searched for peaks that correspond to the centers of circular objects. Since edges have nonzero width and noise will always interfere with the process of peak location, accurate center location requires the use of suitable averaging procedures (Davies, 1984c; Brown, 1984).

This approach clearly requires a very large number of points to be accumulated in parameter space and so a revised form of the method has now become standard: in this approach, locally available edge orientation information at each edge pixel is used to enable the exact positions of circle centers to be estimated (Kimme et al., 1975). This is achieved by moving a distance R along the edge normal at each edge location. Thus, the number of points accumulated is equal to the number of edge pixels in the image:¹ this represents a significant saving in computational load. For this to be possible, the edge detection operator that is employed must be highly accurate. Fortunately, the Sobel operator is able to estimate edge orientation to 1° and is very simple to apply (Chapter 5). Thus, the revised form of the transform is viable in practice.

As was seen in Chapter 5, once the Sobel convolution masks have been applied, the local components of intensity gradient g_x and g_y are available, and the magnitude and orientation of the local intensity gradient vector can be computed using the formulae:

$$g = \left(g_x^2 + g_y^2 \right)^{1/2} \quad (12.1)$$

and

$$\theta = \arctan\left(\frac{g_y}{g_x}\right) \quad (12.2)$$

However, use of the arctan operation is not necessary when estimating center location coordinates (x_c, y_c) since the trigonometric functions can be made to cancel out:

$$x_c = x - R \left(\frac{g_x}{g} \right) \quad (12.3)$$

¹We assume that objects are known to be *either* lighter *or* darker than the background, so that it is only necessary to move along the edge normal in one direction.

$$y_c = y - R \left(\frac{g_y}{g} \right) \quad (12.4)$$

the values of $\cos \theta$ and $\sin \theta$ being given by:

$$\cos \theta = \frac{g_x}{g} \quad (12.5)$$

$$\sin \theta = \frac{g_y}{g} \quad (12.6)$$

In addition, the usual edge thinning and edge linking operations—which normally require considerable amounts of processing—can be avoided if a little extra smoothing of the cluster of candidate center points is performed (Davies, 1984c) (Table 12.2). Thus, this Hough-based approach can be a very efficient one for locating the centers of circular objects, virtually all the superfluous operations having been eliminated, leaving only edge detection, location of candidate center points, and center point averaging to be carried out. In addition, the method is highly robust, so if part of the boundary of an object is obscured or distorted, the object center is still located accurately. In fact, the results are often quite impressive (see, e.g., Figs. 12.1 and 12.2). The reason for this useful property is clear from Fig. 12.3.

The efficiency of the above technique means that it takes slightly less time to perform the actual HT part of the calculation than to evaluate and threshold the intensity gradient over the whole image. Part of the reason for this is that the edge detector operates within a 3×3 neighborhood and necessitates some 12 pixel accesses, four multiplications, eight additions, two subtractions, and an operation for the evaluation of the square root of sum of squares (Eq. (12.1)). As seen in Chapter 5, the latter type of operation is commonly approximated by taking a sum or maximum of absolute magnitudes of the component intensity gradients in order to estimate the magnitude of the local intensity gradient vector.

Table 12.2 A Hough-Based Procedure for Locating Circular Objects

1. Locate edges within the image
2. Link broken edges
3. Thin thick edges
4. For every edge pixel, find a candidate center point
5. Locate all clusters of candidate centers
6. Average each cluster to find accurate center locations

This procedure is particularly robust. It is largely unaffected by shadows, image noise, shape distortions, and product defects. Note that stages 1–3 of the procedure are identical to stages 1–3 in Table 12.1. However, in the Hough-based method, computation can be saved, and accuracy actually increased, by omitting stages 2 and 3.

FIGURE 12.1

Location of broken and overlapping biscuits, showing the robustness of the center location technique. Accuracy is indicated by the black dots, which are each within 1/2 pixel of the radial distance from the center.

Source: © IFS Publications Ltd 1984

FIGURE 12.2

Location of a biscuit with a distortion, showing a chocolate-coated biscuit with excess chocolate on one edge. Note that the computed center has not been “pulled” sideways by the protuberances. For clarity, the black dots are marked 2 pixels outside the normal radial distance.

Source: © IFS Publications Ltd 1984

FIGURE 12.3

Robustness of the Hough transform when locating the center of a circular object. The circular part of the boundary gives candidate center points that focus on the true center, whereas the irregular broken boundary gives candidate center points at random positions.

Source: © Unicom 1988

However, this type of shortcut is not advisable in the present context, since an accurate value for the magnitude of this vector is required in order to compute the position of the corresponding candidate center location with sufficient precision.

Overall, the dictates of accuracy imply that candidate center location requires significant computation. However, substantial increases in speed are still possible by software means alone, as will be seen later in the chapter. Meanwhile, we consider the problems that arise when images contain circles of many different radii, or for one reason or another radii are not known in advance.

12.3 THE PROBLEM OF UNKNOWN CIRCLE RADIUS

There are a number of situations where circle radius is initially unknown. One such situation is where a number of circles of various sizes are being sought—as in the case of coins, or different types of washer. Another is where the circle size is variable—as for food products such as biscuits—so that some tolerance must be built into the system. In general, all circular objects have to be found and their radii measured. In such cases, the standard technique is to accumulate candidate center points simultaneously in a number of parameter planes in a suitably augmented parameter space, each plane corresponding to one possible radius value. The centers of the peaks detected in parameter space give not only the location of

each circle in two dimensions but also its radius. Although this scheme is entirely viable in principle, there are several problems in practice:

1. Many more points have to be accumulated in parameter space.
2. Parameter space requires much more storage.
3. Significantly greater computational effort is involved in searching parameter space for peaks.

To some extent this is to be expected, since the augmented scheme enables more objects to be detected directly in the original image.

It is shown below that the last two problems may largely be eliminated. This is achieved by using just one parameter plane to store all the information for locating circles of different radii, i.e., accumulating not just one point per edge pixel but a whole line of points along the direction of the edge normal in this one plane. In practice, the line need not be extended indefinitely in either direction but only over the restricted range of radii over which circular objects or holes might be expected.

Even with this restriction, a large number of points are being accumulated in a single parameter plane, and it might be thought initially that this would lead to such a proliferation of points that almost any “blob” shape would lead to a peak in parameter space, which might be interpreted as a circle center. However, this is not so and significant peaks normally result only from genuine circles and some closely related shapes.

To understand the situation, consider how a sizeable peak can arise at a particular position in parameter space. This can happen only when a large number of radial vectors from this position meet the boundary of the object normally. In the absence of discontinuities in the boundary, a contiguous set of boundary points can only be normal to radius vectors if they lie on the arc of a circle (indeed, a circle could be *defined* as a locus of points that are normal to the radius vector and form a thin connected line). If a limited number of discontinuities are permitted in the boundary, it may be deduced that shapes like that of a fan² will also be detected using this scheme. Since it is in any case useful to have a scheme that can detect such shapes, the main problem is that there will be some ambiguity in interpretation—i.e., does peak P in parameter space arise from a circle or a fan? In practice, it is quite straightforward to distinguish between such shapes with relatively little additional computation, the really important problem being to cut down the amount of computation needed to key into the initially unstructured image data. Indeed, it is often a good strategy to prescreen the image to eliminate most of it from further detailed consideration and then to analyze the remaining data with tools having much greater discrimination: this two-stage template matching procedure frequently leads to significant savings in computation (VanderBrug and Rosenfeld, 1977; Davies, 1988g).

²A four-blade fan shape bounded by two concentric circles with eight equally spaced radial lines joining them.

A more significant problem arises because of errors in the measurement of local edge orientation. As stated earlier, edge detection operators such as the Sobel introduce an inherent inaccuracy of about 1° . Image noise typically adds a further 1° error, and for objects of radius 60 pixels, the result is an overall uncertainty of about 2 pixels in the estimation of center location. This makes the scheme slightly less specific in its capability for detecting objects.

Overall, the scheme is likely to accept the following object shapes during its prescreening stage:

1. Circles of various sizes
2. Shapes such as fans, which contain arcs of circles
3. Partly occluded or broken versions of these shapes

Sometimes, objects of type 2 will be known *not* to be present. Alternatively, they may readily be identified, e.g., with the aid of corner detectors. Hence, problems of ambiguity may be nonexistent or easy to eliminate in practice. As the situation is highly application-dependent, we will eschew further discussion of this point here.

Next, note that the information on radial distance has been lost by accumulating all votes in a single parameter plane. Hence, a further stage of analysis is needed to measure object radius. This extra stage of analysis normally involves negligible additional computation, because the search space has been narrowed down so severely by the initial circle location procedure. The radial histogram technique (Chapter 20) can be used to measure the radius: in addition, it can be used to perform a final object inspection function.

12.3.1 Some Practical Results

The method described above works much as expected, the main problems arising with small circular objects (of radii less than about 20 pixels) at low resolution (Davies, 1988b). Essentially, the problems are those of lack of discrimination in the precise shapes of small objects (see [Figs. 12.4 and 12.5](#)), as anticipated above. As suggested earlier, this can frequently be turned to advantage in that the method becomes a circular feature detector for small radii (see [Fig. 12.5](#), where a wing nut is located).

As required, objects are detected reliably even when they are partly occluded. However, occlusions can result in the centers of small objects being “pulled” laterally ([Fig. 12.4](#)). More generally, it is clear from [Figs. 12.4 and 12.5](#) that high accuracy of center location cannot be expected when a single parameter plane is used to detect objects over a large range of sizes: hence, it is best to cut down the voting range as far as possible.

Overall, there is a tradeoff between speed and accuracy with this approach. However, the results confirm that it is possible to locate objects of various radii within a significantly conflated parameter space, thereby making substantial

(a) (b)

FIGURE 12.4

(a) Simultaneous location of coins and a key with the modified Hough scheme: the various radii range from 10 to 17 pixels and (b) transform used to compute the centers indicated in (a). Detection efficiency is unimpaired by partial occlusions, shape distortions, and glints. However, displacements of some centers are apparent; in particular, one coin (top left) has only two arcs showing the fact that one of these is distorted, giving a lower curvature, leads to a displacement of the computed center. The shape distortions are due to rather uneven illumination.

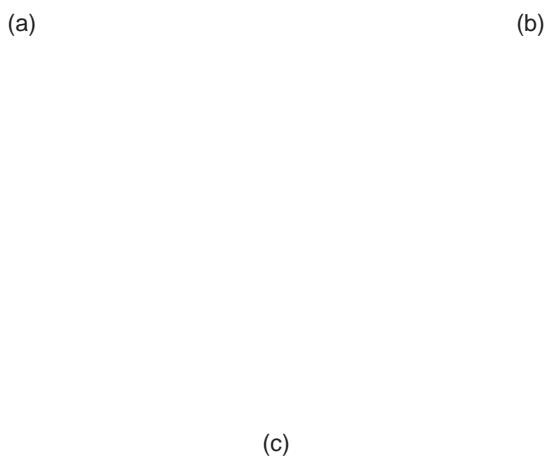
savings in storage and computation—even though the total number of votes that have to be accumulated in parameter space is not itself reduced.

12.4 THE PROBLEM OF ACCURATE CENTER LOCATION

Section 12.3 analyzed the problem of how to cope efficiently with images containing circles of unknown or variable size. This section examines how centers of circles may be located with high (preferably subpixel) accuracy. This problem is relevant since the alternative of increased resolution would entail the processing of many more pixels. Hence, it will be of advantage if high accuracy can be attained with images of low or moderate resolution.

There are a number of causes of inaccuracy in locating the centers of circles using the HT:

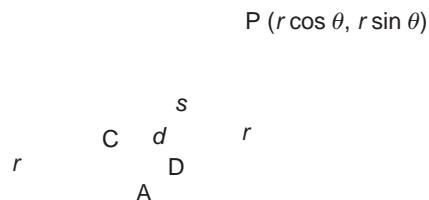
1. Natural edge width may add errors to any estimates of the location of the center.
2. Image noise may cause the radial position of the edge to become modified.
3. Image noise may cause variation in the estimates of local edge orientation.
4. The object itself may be distorted, in which case the most accurate and robust estimate of circle center is what is required (i.e., the algorithm should be able to ignore the distortion and take a useful average from the remainder of the boundary).

**FIGURE 12.5**

(a) Accurate simultaneous detection of a lens cap and a wing nut when radii are assumed to range from 4 to 17 pixels; (b) response in parameter space that arises with such a range of radii: note the overlap of the transforms from the lens cap and the bracket; (c) hole detection in the image of (a) when radii are assumed to fall in the range -26 to -9 pixels (negative radii are used since holes are taken to be objects of negative contrast): clearly, in *this* image a smaller range of negative radii could have been employed.

5. The object may appear distorted because of inadequacies in the method of illumination.
6. The edge orientation operator will have a significant inherent inaccuracy of its own, which contributes to inaccuracy in the estimation of center location.

Evidently, it is necessary to minimize the effects of all these potential sources of error. In applying the HT, it is usual to bear in mind that all possible center locations that could have given rise to the currently observed edge pixel should be accumulated in parameter space. Therefore, we should accumulate in parameter space not a single candidate center point corresponding to the given edge pixel, but a point spread function (PSF), which may be approximated by a Gaussian error function: this will generally have different radial and transverse

**FIGURE 12.6**

Arrangement for obtaining improved center approximation.

standard deviations. The radial standard deviation will commonly be 1 pixel—corresponding to the expected width of the edge—whereas the transverse standard deviation will frequently be at least 2 pixels, and for objects of moderate (60 pixels) radius the intrinsic orientation accuracy of the edge detection operator contributes at least 1 pixel to this standard deviation.

When the center need only be estimated to within 1 pixel, these considerations hardly matter. However, when it is desired to locate center coordinates to within 0.1 pixel, each PSF will have to contain 100 points, all of which will have to be accumulated in a parameter space of much increased spatial resolution, or its equivalent,³ if the required accuracy is to be attained. However, the following section outlines a technique that can cut down the amount of computation from this sort of level, without significant loss of accuracy.

12.4.1 A Solution Requiring Minimal Computation

A potential key to increasing accuracy of center location arises from the observation that most of the inaccuracy in calculating the position of the center is due to transverse rather than radial errors. Hence, it is reasonable to concentrate on eliminating transverse errors.

This immediately leads to the following strategy: find a point D in the region of the center and use it to obtain a better approximation A to the center by moving from the current edge pixel P a distance equal to the expected radius r in the direction of D (Fig. 12.6). Then repeat the process 1 edge pixel at a time until all edge pixels have been taken into account. Although intuition indicates that this strategy will lead to a good final approximation to the center, the edge points need to be taken in random order to prevent the bias of the initial center approximation from

³For example, some abstract list structure might be employed, which effectively builds up to high resolution only where needed (see also the adaptive HT scheme described in Chapter 13).

permanently affecting the final result.⁴ In addition, some averaging of the intermediate results is needed to minimize the effects of boundary noise, and edge points that give extreme predictions for the center approximation (e.g., deviations of more than ~ 2 pixels) have to be discounted.

Overall, there are two main stages of calculation in the whole algorithm:

1. Find the position of the center to within ~ 1 pixel by the usual Hough technique.
2. Use the iterative procedure to obtain the final result, with an accuracy of 0.1 pixels.

Tests of this algorithm on accurately made circular objects led to accuracies of center location in the region of 0.1 pixels (Davies, 1988e). It proved difficult to increase the accuracy further because of the problem of setting up lighting systems of sufficient uniformity, i.e., the limit was set by practicalities of illumination rather than by the algorithm itself.

Finally, a by-product of the approach is that radius r can be obtained with exceptionally high accuracy—generally within 0.05 pixels (Davies, 1988e).

12.5 OVERCOMING THE SPEED PROBLEM

Section 12.4 studied how the accuracy of the HT circle detection scheme could be improved substantially, with modest additional computational cost. This section examines how circle detection may be carried out with significant improvement in speed. To achieve this, two methods are tried: (1) sampling the image data and (2) using a simpler edge detector. The most appropriate strategy for (1) appears to be to look only at every n th line in the image, whereas that for (2) involves using a small two-element neighborhood while searching for edges (Davies, 1987f). Although this approach will lose the capability for estimating edge orientation, it will still permit horizontal and vertical chords of a circle to be bisected, thereby leading to values for the center coordinates x_c , y_c . It also involves much less computation, the multiplications and square root calculations, and most of the divisions being eliminated or replaced by two-element differencing operations, thereby giving a further gain in speed.

12.5.1 More Detailed Estimates of Speed

To help understand the situation, this section estimates the gain in speed that should result by applying the strategy described above. First, the amount of computation involved in the original Hough-based approach is modeled by:

$$T_0 = N^2 s + St_0 \quad (12.7)$$

⁴This problem arises because the method is intrinsically sequential rather than parallel, and it is necessary to compensate for this.

where T_0 is the total time taken to run the algorithm on an $N \times N$ pixel image; s is the time taken per pixel to compute the Sobel operator and to threshold the intensity gradient g ; S is the number of edge pixels that are detected; and t_0 is the time taken to compute the position of a candidate center point.

Next, the amount of computation in the basic chord bisection approach is modeled as:

$$T = 2(N^2 q + Qt) \quad (12.8)$$

where T is the total time taken to run the algorithm; q is the time taken per pixel to compute a two-element x or y edge detection operator and to perform thresholding; Q is the number of edge pixels that are detected in one or other scan direction; and t is the time taken to compute the position of a candidate center point coordinate (either x_c or y_c). In Eq. (12.8), the factor 2 results because scanning occurs in both horizontal and vertical directions. If the image data are now sampled by examining only a proportion α of all possible horizontal and vertical scan lines, the overall gain in speed from using the chord bisection scheme should be:

$$G = \frac{N^2 s + St_0}{2\alpha(N^2 q + Qt)} \quad (12.9)$$

Typical values of relevant parameters for (say) a biscuit of radius 32 pixels in a 128×128 pixel image are listed below:

$N^2 = 16,384$	$t_0/s \approx 1$
$S \approx Q \approx 200$	$s/q \approx 6$
$\alpha \approx 1/3$	$t_0/t \approx 5$

Hence

$$G \approx \frac{s}{2\alpha q} \approx 9 \quad (12.10)$$

Broadly, this corresponds to a gain ~ 3 from sampling and a further gain ~ 3 from applying a much simplified edge detector. However, if the sampling factor $1/\alpha$ could be increased further, greater gains in speed could be obtained—in principle without limit, but in practice the situation is governed by how robust the algorithm really is.

12.5.2 Robustness

Robustness can be considered relative to two factors. The first is the amount of noise in the image and the second is the amount of signal distortion that can be tolerated. Fortunately, both the original HT and the chord bisection approach lead to peak finding situations, and if there is any distortion of the object shape, then points are thrown into relatively random locations in parameter space and consequently do not have a significant direct impact on the accuracy of peak location.

However, they do have an indirect impact in that the signal-to-noise ratio is reduced, so that accuracy is impaired. In fact, if a fraction β of the original signal is removed, leaving a fraction $\gamma = 1 - \beta$, either due to such distortions or occlusions or else by the deliberate sampling procedures already outlined, then the number of independent measurements of the center location drops to a fraction γ of the optimum. This means that the accuracy of estimation of the center location drops to a fraction $\sqrt{\gamma}$ of the optimum. Since noise affects the optimum accuracy directly, we have shown the result of both major factors governing robustness.

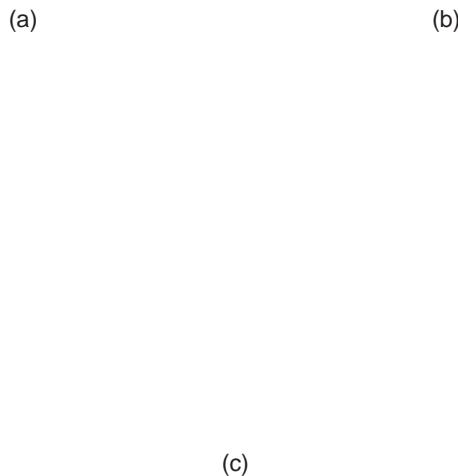
What is important here is that the effect of sampling is substantially the same as that of signal distortion, so that the more distortion that must be tolerated, the higher the value α has to have. This principle applies both to the original Hough approach and to the chord bisection algorithm. However, the latter does its peak finding in a different way—*via* 1-D rather than 2-D averaging processes. As a result, it turns out to be somewhat more robust than the standard HT in its capability for accepting a high degree of sampling.

This gain in capability for accepting sampling carries with it a set of related disadvantages: highly textured objects may not easily be located by the method; high noise levels may confuse it via the production of too many false edges; and (since it operates by specific x and y scanning mechanisms) there must be a sufficiently small amount of occlusion and other gross distortion that a significant number of scans (both horizontally and vertically) pass through the object. Ultimately, this means that the method will not tolerate more than about one-quarter of the circumference of the object being absent.

12.5.3 Practical Results

Tests (Davies, 1987f) with the image in Fig. 12.7 show that gains in speed of more than 25 can be obtained, with values of α down to less than 0.1 (i.e., every 10th horizontal and vertical line scanned). The results for broken circular products (Figs. 12.8 and 12.9) are self-explanatory; they indicate the limits to which the method can be taken. An outline of the complete algorithm is given in Table 12.3 (note the relatively straightforward problem of disambiguating the results if there happen to be several peaks).

Figure 12.10 shows the effect of adjusting the threshold in the two-element edge detector. The result of setting it too low is seen in Fig. 12.10(a). Here the surface texture of the object has triggered the edge detector, and the chord midpoints give rise to a series of false estimates of the center coordinates. Figure 12.10(b) shows the result of setting the threshold at too high a level, so that the number of estimates of center coordinates is reduced and sensitivity suffers. Although the images in Fig. 12.7 were obtained with the threshold adjusted intuitively, a more rigorous approach can be taken by optimizing a suitable criterion function. There are two obvious functions: (1) the number of accurate center predictions n and (2) the speed–sensitivity product. The latter can be written in the form \sqrt{n}/T , where T is the execution time. The two methods of optimization make little difference in the

**FIGURE 12.7**

Successful object location using the chord bisection algorithm for the same initial image, using successive step sizes of 2, 4, and 8 pixels. The black dots show the positions of the horizontal and vertical chord bisectors, and the white dot shows the position found for the center.

example shown in Fig. 12.10. However, had there been a strong regular texture on the object, the situation would have been rather different.

12.5.4 Summary

The center location procedure described above is more than an order of magnitude faster than the standard Hough-based approach and often as much as 25 times faster. This could be quite important in exacting real-time applications. Robustness is so good that the method tolerates at least one-quarter of the circumference of an object being absent, making it adequate for many real applications. In addition, it is entirely clear what types of image data would be likely to confuse the algorithm.

FIGURE 12.8

Successful location of a broken object using the chord bisection algorithm: only about one-quarter of the ideal boundary is missing.

FIGURE 12.9

A test on overlapping and broken biscuits: the overlapping objects are successfully located, albeit with some difficulty, but there is no chance of finding the center of the broken biscuit since over one-half of the ideal boundary is missing.

Table 12.3 Outline of the Fast Center-Finding Algorithm

```

y = 0;
do {
    scan horizontal line y looking for start and end of each object;
    calculate midpoints of horizontal object segments;
    accumulate midpoints in 1-D parameter space (x space);
    // note that the same space, x space, is used for all lines y
    y = y + d;
} until y > ymax;

x = 0;
do {
    scan vertical line x looking for start and end of each object;
    calculate midpoints of vertical object segments;
    accumulate midpoints in 1-D parameter space (y space);
    // note that the same space, y space, is used for all lines x
    x = x + d;
} until x > xmax;

find peaks in x space;
find peaks in y space;
test all possible object centres arising from these peaks;
// the last step is necessary only if  $\exists > 1$  peak in each space
// d is the horizontal and vertical step-size ( $= 1/\alpha$ )

```

(a)

(b)

FIGURE 12.10

Effect of misadjustment of the gradient threshold: (a) effect of setting the threshold too low, so that surface texture confuses the algorithm and (b) loss of sensitivity on setting the threshold too high.

Not only is robustness built into the algorithm in such a way as to emulate the standard Hough-based approach, but it is possible to interpret the method as being a Hough-based approach, since the center coordinates are each accumulated in their own 1-D “parameter spaces.” These considerations give further insight into the robustness of the standard Hough technique.

12.6 ELLIPSE DETECTION

The problem of detecting ellipses may seem only marginally more complex than that of detecting circles—as eccentricity is only a single parameter. However, eccentricity destroys the symmetry of the circle, so the direction of the major axis also has to be defined. As a result, five parameters are required to describe an ellipse, and ellipse detection has to take account of this, either explicitly or implicitly. In spite of this, one method for ellipse detection is especially simple and straightforward to implement, i.e., the diameter bisection method, which is described next.

12.6.1 The Diameter Bisection Method

The diameter bisection method of Tsuji and Matsumoto (1978) is very simple in concept. First, a list is compiled of all the edge points in the image. Then, the list is sorted to find those that are antiparallel, so that they could lie at opposite ends of ellipse diameters; next, the positions of the center points of the connecting lines for all such pairs are taken as voting positions in parameter space (Fig. 12.11). As for circle location, the parameter space that is used for this purpose is congruent to image space. Finally, the positions of significant peaks in parameter space are located to identify possible ellipse centers.

FIGURE 12.11

Principle of the diameter bisection method. A pair of points is located for which the edge orientations are antiparallel. If such a pair of points lies on an ellipse, the midpoint of the line joining the points will be situated at the center of the ellipse.

Naturally, in an image containing many ellipses and other shapes, there will be very many pairs of antiparallel edge points and for most of these the center points of the connecting lines will lead to nonuseful votes in parameter space. Clearly, such clutter leads to wasted computation. However, it is a principle of the HT that votes must be accumulated in parameter space at all points that *could in principle* lead to correct object center location: it is left to the peak finder to find the voting positions that are most likely to correspond to object centers.

Not only does clutter lead to wasted computation, but the method itself also is computationally expensive. This is because it examines all *pairs* of edge points, and there are many more such pairs than there are edge points (m edge points lead to ${}^m C_2 \approx m^2/2$ pairs of edge points). Indeed, since there are likely to be at least 1000 edge points in a typical image, the computational problems can be formidable.

Interestingly, the basic method is not particularly discriminating about ellipses. It picks out many symmetrical shapes—any indeed that possess 180° rotation symmetry, including rectangles, ellipses, circles, or superellipses (these have equations of the form $x^s/a^s + y^s/b^s = 1$, ellipses being a special case). In addition, the basic scheme sometimes gives rise to a number of false identifications even in an image in which only ellipses are present (Fig. 12.12). However, Tsuji and Matsumoto (1978) also proposed a technique by which true ellipses can be distinguished. The basis of the technique is the property of an ellipse that the lengths of perpendicular semidiameters OP , OQ obey the relation:

$$\frac{1}{OP^2} + \frac{1}{OQ^2} = \frac{1}{R^2} = \text{constant} \quad (12.11)$$

To proceed, the set of edge points that contribute to a given peak in parameter space is used to construct a histogram of R values (the latter being obtained from

FIGURE 12.12

Result of using the basic diameter bisection method. The larger dots show true ellipse centers found by the method, whereas the smaller dots show positions at which false alarms commonly occur. Such false alarms are eliminated by applying the test described in the text.

FIGURE 12.13

Limitations of the diameter bisection method: of the three ellipses shown, only the lowest one cannot be located by the diameter bisection method.

Source: © Unicom 1988

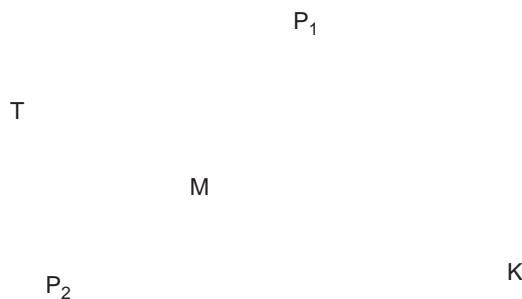
Eq. (12.11)). If a significant peak is found in this histogram, then there is clear evidence of an ellipse at the specified location in the image. If two or more such peaks are found, then there is evidence of a corresponding number of concentric ellipses in the image. If, however, no such peaks are found, then a rectangle, superellipse, or other symmetrical shape may be present and each of these would need its own identifying test.

The method obviously relies on there being an appreciable number of pairs of edge points on an ellipse lying at opposite ends of diameters: hence, there are strict limits on the amount of the boundary that must be visible (Fig. 12.13). Finally, it should not go unnoticed that the method wastes the signal available from unmatched edge points. These considerations have led to a search for further methods of ellipse detection.

12.6.2 The Chord–Tangent Method

The chord–tangent method was devised by Yuen et al. (1988) and makes use of another simple geometric property of the ellipse. Again pairs of edge points are taken in turn, and for each point of the pair, tangents to the ellipse are constructed and found to cross at T, the midpoint of the connecting line is found at M, and then the equation of line TM is calculated and all points that lie on the portion MK of this line are accumulated in parameter space (Fig. 12.14) (clearly, T and the center of the ellipse lie on the opposite sides of M). Finally, peak location proceeds as before.

The proof that this method is correct is trivial. Symmetry ensures that the method works for circles, and projective properties then ensure that it also works for ellipses: under projection, straight lines project into straight lines, midpoints into midpoints, tangents into tangents, and circles into ellipses; in addition, it is

**FIGURE 12.14**

Principle of the chord–tangent method. The tangents at P_1 and P_2 meet at T and the midpoint of P_1P_2 is M . The center C of the ellipse lies on the line TM produced. Note that M lies between C and T . Hence, the transform for points P_1 and P_2 need only include the portion MK of this line.

always possible to find a viewpoint such that a circle can be projected into a given ellipse.

Unfortunately, this method suffers from significantly increased computation, since so many points have to be accumulated in parameter space. This is obviously the price to be paid for greater applicability. However, computation can be minimized in at least three ways: (1) cutting down the lengths of the lines of votes accumulated in parameter space by taking account of the expected sizes and spacings of ellipses; (2) not pairing edge points initially if they are too close together or too far apart; and (3) eliminating edge points once they have been identified as belonging to a particular ellipse.

12.6.3 Finding the Remaining Ellipse Parameters

Although the methods described above are designed to locate the center coordinates of ellipses, a more formal approach is required to determine other ellipse parameters. Accordingly, we write the equation of an ellipse in the form:

$$Ax^2 + 2Hxy + By^2 + 2Gx + 2Fy + C = 0 \quad (12.12)$$

an ellipse being distinguished from a hyperbola by the additional condition:

$$AB > H^2 \quad (12.13)$$

This condition guarantees that A can never be zero and that the ellipse equation may without loss of generality be rewritten with $A = 1$. This leaves five parameters, which can be related to the position of the ellipse, its orientation, and its size and shape (or eccentricity).

Having located the center of the ellipse, we may select a new origin of coordinates at its center (x_c, y_c) ; the equation then takes the form:

$$x'^2 + 2Hx'y' + By'^2 + C' = 0 \quad (12.14)$$

where

$$x' = x - x_c; \quad y' = y - y_c \quad (12.15)$$

It now remains to fit to Eq. (12.14) the edge points that gave evidence for the ellipse center under consideration. The problem will normally be vastly overdetermined. Hence, an obvious approach is the method of least squares. Unfortunately, this technique tends to be very sensitive to outlier points and is therefore liable to be inaccurate. An alternative is to employ some form of Hough transformation. Here we follow Tsukane and Goto (1983) by differentiating Eq. (12.14):

$$x' + \frac{By'}{dx'} + H\left(y' + \frac{x'dy'}{dx'}\right) = 0 \quad (12.16)$$

Then dy'/dx' can be determined from the local edge orientation at (x', y') and a set of points accumulated in the new (H, B) parameter space. When a peak is eventually located in (H, B) space, the relevant data (a subset of a subset of the original set of edge points) can be used with Eq. (12.14) to obtain a histogram of C' values, from which the final parameter for the ellipse can be obtained.

The following formulae are needed to determine the orientation θ and semi-axes a and b of an ellipse in terms of H , B , and C' :

$$\theta = \frac{1}{2} \arctan\left(\frac{2H}{1-B}\right) \quad (12.17)$$

$$a^2 = \frac{-2C'}{(B+1) - [(B-1)^2 + 4H^2]^{1/2}} \quad (12.18)$$

$$b^2 = \frac{-2C'}{(B+1) + [(B-1)^2 + 4H^2]^{1/2}} \quad (12.19)$$

Mathematically, θ is the angle of rotation that diagonalizes the second-order terms in Eq. (12.14); having performed this diagonalization, the ellipse is then essentially in the standard form $\tilde{x}^2/a^2 + \tilde{y}^2/b^2 = 1$, so a and b are determined.

Note that the above method finds the five ellipse parameters in *three* stages: first the positional coordinates are obtained, then the orientation, and finally the size and eccentricity.⁵ This three-stage calculation involves less computation but compounds any errors—in addition, edge orientation errors, although low, become a limiting factor. For this reason, Yuen et al. (1988) tackled the problem by speeding up the HT procedure itself rather than by avoiding a direct assault on

⁵Strictly, the eccentricity is $e = (1 - b^2/a^2)^{1/2}$, but in most cases we are more interested in the ratio of semiminor to semimajor axes, b/a .

Eq. (12.14): i.e., they aimed at a fast implementation of a thoroughgoing second stage, which finds all the parameters of Eq. (12.14) in one 3-D parameter space.

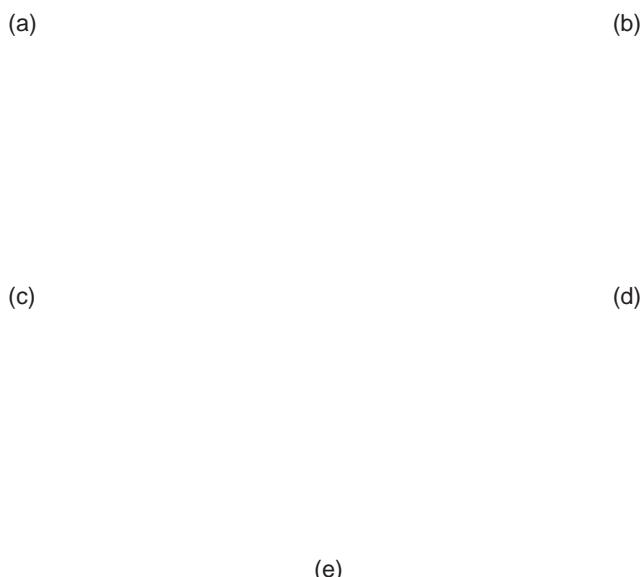
It is now clear that reasonably optimal means are available for finding the orientation and semi-axes of an ellipse once its position is known: the weak point in the process appears to be that of finding the ellipse initially. Indeed, the two approaches for achieving this that have been described above are particularly computation intensive, mainly because they examine all pairs of edge points; a possible alternative is to apply the generalized Hough transform (GHT), which locates objects by taking edge points singly: this possibility will be considered in Chapter 13.

12.7 HUMAN IRIS LOCATION

Human iris location is an important application of computer vision for three reasons: (1) it provides a useful cue for human face analysis; (2) it can be used for the determination of gaze direction; and (3) it is useful in its own right for biometric purposes, that is to say, for identifying individuals almost uniquely. The latter possibility has already been noted in Chapter 8 where textural methods for iris recognition are outlined and some key references are given. More details of human face location and analysis will be given in Chapter 17. Here we concentrate on iris location using the Hough transform.

In fact, we can tackle the iris location and recognition task reasonably straightforwardly. First, if the head has been located with reasonable accuracy, then it can form a region of interest, inside which the iris can be sought. In a front view with the eyes looking ahead, the iris will be seen as a round often high-contrast object, and can be located straightforwardly with the aid of a HT (Ma et al., 2003). In some cases, this will be less easy because the iris is relatively light and the color may not be distinctive—although a lot will depend on the quality of the illumination. Perhaps more important, in some human subjects, the eyelid and lower contour of the eye may partially overlap the iris (Fig. 12.15(a)), making it more difficult to identify, although, as confirmed below, HTs are capable of coping with a fair degree of occlusion.

Note that the iris will appear elliptical if the eyes are not facing directly ahead; in addition, the shape of the eye is far from spherical and the horizontal diameter is larger than the vertical diameter—again making the iris appear elliptical (Wang and Sung, 2001). In either case, the iris can still be detected using a Hough transform. Furthermore, once this has been done, it should be possible to estimate the direction of gaze with a reasonable degree of accuracy (Gong et al., 2000), thereby taking us further than mere recognition. (The fact that measurement of ellipse eccentricity would lead to an ambiguity in the gaze direction can be offset by measuring the position of the ellipse on the eyeball.) Finally, Toennies et al. (2002) showed that the HT can be used to localize the irises for real-time applications, in spite of quite substantial partial occlusion by the eyelid and lower contour of the eye.

**FIGURE 12.15**

Iris location using the Hough transform. (a) Original image of the eye region of the face with irises located by a uniformly weighted Hough transform (HT). (b) Irises located using a gradient-weighted HT. (c) and (d) The respective HTs used to locate the irises in (a) and (b). (e) The Canny operator (incorporating smoothing, nonmaximum suppression, and hysteresis) is used to obtain the initial edge image in both cases. The sharper peaks obtained using the gradient weighting permit the irises to be located significantly more robustly and accurately. Note the number of additional edges in (e) that are able to produce substantive numbers of additional votes, which interfere with those from the irises.

A number of the points made above are illustrated by the example in Fig. 12.15. Far from being a trivial application of the HT, there are a surprisingly large number of edges in the eye region: these produce significant numbers of additional votes, which interfere with those from the irises. These arise from the upper and lower eyelids and the folds of skin around them. Hence, the accuracy of the method is not assured: this makes gradient weighting (see Section 13.6) especially valuable. The radii of the irises shown in Fig. 12.15 are about 17.5 pixels, and there is no particular evidence of ellipticity in their shape. However, more accurate location of the iris and measurement of ellipticity in order to estimate orientation (e.g., to determine the angle of gaze) require considerably greater resolution, with iris radii approaching 100 pixels, whereas pictures that analyze iris texture patterns for biometric purposes require even larger radii.

12.8 HOLE DETECTION

At this stage, an obvious question is whether the usual methods for circular object detection can be applied to hole detection. In principle, this is undoubtedly practicable, in many cases the only change arising because holes have negative contrast. This means that if the HT technique is used, votes will have to be accumulated on moving a distance $-R$ along the direction of the local edge normal. There is no difficulty with this for large holes, but as the holes become smaller, complications tend to occur because of poor lighting inside the holes. As a result, large regions of shadow are likely to occur, and the contrast is liable to be weak and variable. We shall not consider the general situation further as it is so dependent on size, illumination, and other factors, such as 3-D shape and the material within which the hole appears.

Nevertheless, for small holes a few pixels across, detection can be important, as they constitute point features, and these can be valuable for precise object location, as happens for biscuits, hinges, brackets, and a host of other manufactured parts (see Chapter 14). In fact, circular holes may even be preferable to corners for object location, as they are, ideally, isotropic and thus less liable to bias when used for measurement.

Very small holes can be detected using template matching. This involves applying a Laplacian type of operator, and in a 3×3 neighborhood this may take the form of one of the following convolution masks:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} -2 & -3 & -2 \\ -3 & 20 & -3 \\ -2 & -3 & -2 \end{bmatrix}$$

Having applied the convolution, the resulting image is thresholded to indicate regions where holes might be present. Note that the coefficients in each of the above masks sum to zero so that they are insensitive to varying levels of background illumination.

If the holes in question are larger than 1–2 pixels in diameter, the above masks will prove inadequate. Clearly, larger masks are required if larger holes are to be detected efficiently at their centers. For much larger holes, the masks that are required become impractically large, and the HT approach becomes a more attractive option.

12.9 CONCLUDING REMARKS

This chapter has described techniques for circle and ellipse detection, starting with the HT approach. Although the HT is found to be effective and highly robust against occlusions, noise, and other artifacts, it incurs considerable storage and computation—especially if it is required to locate circles of unknown radius or if high accuracy is required. Techniques have been described whereby the latter two problems can be tackled much more efficiently; in addition, a method has been

described for markedly reducing the computational load involved in circle detection. The general circle location scheme is a type of HT but although the other two methods are related to the HT, they are distinct methods, which draw on the HT for inspiration. As with the HT, these methods achieve robustness as an integral part of their design—i.e., robustness is not included as an afterthought—so they achieve known levels of robustness.

Two HT-based schemes for ellipse detection have also been described—the diameter bisection method and the chord–tangent method. A further approach to ellipse detection, based on the generalized HT, will be covered in Chapter 13. At that point, further lessons will be drawn on the efficacies of the various methods.

As in the case of line detection, a trend running through the design of circle and ellipse detection schemes is the deliberate splitting of algorithms into two or more stages. This is useful for keying into the important and relevant parts of an image prior to finely discriminating one type of object or feature from another, or prior to measuring dimensions or other characteristics accurately. Indeed, the concept can be taken further, in that the efficiencies of all the algorithms discussed in this chapter have been improved by searching first for edge features in the image. The concept of two-stage template matching is therefore deep seated in the methodology of the subject and is developed further in later chapters. Although two-stage template matching is a standard means of increasing efficiency (VanderBrug and Rosenfeld, 1977; Davies, 1988g), it is not obvious that efficiency can always be increased in this way. It appears to be in the nature of the subject that ingenuity is needed to discover means of achieving this.

The Hough transform is very straightforwardly applied to circle detection, for which it achieves an impressive level of robustness. It is also easily applied to ellipse detection. Practical issues include accuracy, speed, and storage requirements—some of which can be improved by employing parameter spaces of reduced dimension, although this can affect the specificity that can be achieved.

12.10 BIBLIOGRAPHICAL AND HISTORICAL NOTES

The Hough transform was developed in 1962 and first applied to circle detection by Duda and Hart (1972). However, the now standard HT technique, which makes use of edge orientation information to reduce computation, only emerged 3 years later (Kimme et al., 1975). The author's work on circle detection for automated inspection required real-time implementation and also high accuracy. This spurred the development of the three main techniques described in [Sections 12.3–12.5](#) (Davies, 1987f, 1988b, 1988e). In addition, the author has considered the effect of noise on edge orientation computations, showing in particular their effect in reducing the accuracy of center location (Davies, 1987e).

Yuen et al. (1989) reviewed various existing methods for circle detection using the HT. In general, their results confirmed the efficiency of the method described in [Section 12.3](#) when circle radius is unknown, although they found that the two-stage process that was involved can sometimes lead to slight loss of robustness. It appears that this problem can be reduced in some cases by using a modified version of the algorithm of Gerig and Klein (1986). Note that the Gerig and Klein approach is itself a two-stage procedure: it is discussed in detail in Chapter 13. More recently, Pan et al. (1995) have increased the speed of computation of the HT by prior grouping of edge pixels into arcs, for an underground pipe inspection application.

The two-stage template matching technique and related approaches for increasing search efficiency in digital images were known by 1977 (Nagel and Rosenfeld, 1972; Rosenfeld and VanderBrug, 1977; VanderBrug and Rosenfeld, 1977) and have undergone further development since then—especially in relation to particular applications such as those described in this chapter (Davies, 1988g).

Later, Atherton and Kerbyson (1999) (see also Kerbyson and Atherton, 1995) showed how to find circles of arbitrary radius in a single parameter space using the novel procedure of coding radius as a phase parameter and then performing accumulation with the resulting phase-coded annular kernel. Using this approach, they attained higher accuracy with noisy images; Goulermas and Liatsis (1998) showed how the HT could be fine tuned for the detection of fuzzy circular objects such as overlapping bubbles by using genetic algorithms. In effect, the latter are able to sample the solution space with very high efficiency and hand over cleaner data to the following HT.

The ellipse detection sections are based particularly on the work of Tsuji and Matsumoto (1978), Tsukune and Goto (1983), and Yuen et al. (1988); for a fourth method (Davies, 1989a) using the GHT idea of Ballard (1981) in order to save computation, see Chapter 13. The contrasts between these methods are many and intricate, as this chapter has shown. In particular, the idea of saving dimensionality in the implementation of the GHT appears also in a general circle detector (Davies, 1988b). At that point in time, the necessity for a multistage approach to determination of ellipse parameters seemed proven, although somewhat surprisingly the optimum number of such stages was just two.

Later algorithms represented moves to greater degrees of robustness with real data by explicit inclusion of errors and error propagation (Ellis et al., 1992); increased attention was subsequently given to the verification stage of the Hough approach (Ser and Siu, 1995). In addition, work was carried out on the detection of superellipses, which are shapes intermediate in shape between ellipses and rectangles, although the technique used (Rosin and West, 1995) was that of segmentation trees rather than HTs (nonspecific detection of superellipses can of course be achieved by the diameter bisection method (see [Section 12.6.1](#)); see also Rosin (2000)).

For cereal grain inspection, with typical flow rates in excess of 300 grains per second, ultrafast algorithms were needed and the resulting algorithms were

limiting cases of chord-based versions of the HT (Davies, 1999b, 1999d); a related approach was adopted by Xie and Ji (2002) for their efficient ellipse detection method; Lei and Wong (1999) employed a method that was based on symmetry, and this was found to be able to detect parabolas and hyperbolas as well as ellipses:⁶ it was also reported as being more stable than other methods since it does not have to calculate tangents or curvatures; the latter advantage has also been reported by Sewisy and Leberl (2001). The fact that even in the 2000s, basic new ellipse detection schemes are being developed says something about the science of image analysis: even today the toolbox of algorithms is incomplete, and the science of how to choose between items in the toolbox, or how, *systematically*, to develop new items for the toolbox, is immature. Further, although all the parameters for specification of such a toolbox may be known, knowledge about the possible tradeoffs between them is still limited.

12.10.1 More Recent Developments

Much work has recently been carried out on iris detection using the HT. Jang et al. (2008) were particularly concerned with overlap of the iris region by the upper and lower eyelids, and used a parabolic version of the HT to accurately locate their boundaries, taking special care to limit the computational load. Li et al. (2010) used a circular HT to locate the iris and a RANSAC-like technique for locating the upper and lower eyelids, again using a parabolic model for the latter: their approach was designed to cope with very noisy iris images. Chen et al. (2010) used a circular HT to locate the iris and a straight line HT to locate up to two line segment approximations to the boundaries of each of the eyelids. Cauchie et al. (2008) produced a new version of the HT to accurately locate common circle centers from circular or partial circle segments, and demonstrated its value for iris location. Min and Park (2009) used a circular HT for iris detection, a parabolic HT for eyelid detection, and followed this with eyelash detection using thresholding.

Finally, we summarize the work carried out by Guo et al. (2009) to overcome the problems of dense sets of edges in textured regions. To reduce the impact of such edges, a measure of isotropic surround suppression was introduced: the resulting algorithm gave small weights to edges in texture regions and large weights to edges on strong and clear boundaries when accumulating votes in Hough space. The approach gave good results when locating straight lines in scenes containing man-made structures such as buildings.

⁶Note that while this is advantageous in some applications, the lack of discrimination could prove to be a disadvantage in other applications.

12.11 PROBLEMS

1. Prove the result of [Section 12.4.1](#) that as D approaches C and d approaches zero, the shape of the locus becomes a circle on DC as diameter.
2.
 - a. Describe the use of the Hough transform for circular object detection, assuming that object size is known in advance. Show also how a method for detecting ellipses could be adapted for detecting circles of unknown size.
 - b. A new method is suggested for circle location that involves scanning the image both horizontally and vertically. In each case, the midpoints of chords are determined and their x or y coordinates are accumulated in separate 1-D histograms. Show that these can be regarded as simple types of Hough transform, from which the positions of circles can be deduced. Discuss whether any problems would arise with this approach; consider also whether it would lead to any advantages relative to the standard Hough transform for circle detection.
 - c. A further method is suggested for circle location. This again involves scanning the image horizontally, but in this case, for every chord that is found, an estimate is immediately made of the *two* points at which the center could lie, and votes are placed at those locations. Work out the geometry of this method, and determine whether this method is faster than the method outlined in (b). Determine whether the method has any disadvantages compared with method described in (b)?
3. Determine which of the methods described in this chapter will detect (a) hyperbolas, (b) curves of the form $Ax^3 + By^3 = 1$, and (c) curves of the form $Ax^4 + Bx + Cy^4 = 1$.
4. Prove Eq. [\(12.11\)](#) for an ellipse. *Hint:* Write the coordinates of P and Q in suitable parametric forms, and then use the fact that $OP \perp OQ$ to eliminate one of the parameters from the left-hand side of the equation.
5. Describe the *diameter bisection* and *chord–tangent* methods for the location of ellipses in images, and compare their properties. Justify the use of the chord–tangent method by proving its validity for circle detection and then extending the proof to cover ellipse detection.
6. Round coins of a variety of sizes are to be located, identified, and sorted in a vending machine. Discuss whether the chord–tangent method should be used for this purpose instead of the usual form of Hough transform circle location scheme operating within a 3D (x, y, r) parameter space.
7. Outline each of the following methods for locating ellipses using the Hough transform: (a) the *diameter bisection* method and (b) the *chord–tangent* method. Explain the principles on which these methods rely. Determine which is more robust and compare the amounts of computation each requires.
8. For the diameter bisection method, searching through lists of edge points with the right orientations can take excessive computation. It is suggested that a

two-stage approach might speed up the process: (a) load the edge points into a table, which may be addressed by orientation and (b) look up the right edge points by feeding appropriate orientations into the table. Estimate how much this would be likely to speed up the diameter bisection method.

9. It is found that the diameter bisection method sometimes becomes confused when several ellipses appear in the same image, and generates false “centers” that are not situated at the centers of any ellipses. It is also found that certain other shapes are detected by the diameter bisection method. Ascertain in each case quite what the method is sensitive to, and consider ways in which these problems might be overcome.

The Hough Transform and Its Nature

13

It has already been seen that the Hough transform can be used to locate straight line, circle, and ellipse features in digital images. It would be useful to know whether the method can be generalized to cover all shapes and whether it is always as robust as it is for the original three examples. This chapter discusses these questions, showing that the method can be generalized and is broadly able to retain its robustness properties.

Look out for:

- the generalized Hough transform technique.
- its relation to spatial matched filtering.
- how sensitivity is optimized by gradient rather than uniform weighting.
- use of the generalized HT for ellipse detection.
- how speed can be improved by the use of a universal lookup table.
- how the computational loads of the various HT techniques can be estimated.
- the value of the Gerig and Klein back-projection technique in cutting down the effects of extraneous clutter.

This chapter describes the generalized Hough transform and generalizes our view of the HT as a generic computer vision technique. It also makes order calculations of computational load for three HT-based methods of ellipse detection.

13.1 INTRODUCTION

In Chapters 11 and 12, it has been seen that the Hough transform (HT) is of great importance for the detection of features such as lines, circles, and ellipses, and for finding relevant image parameters. This makes it worthwhile to see the extent to which the method can be generalized so that it can detect arbitrary shapes. The

works of Merlin and Farber (1975) and Ballard (1981) were crucial historically and led to the development of the generalized Hough transform (GHT). The GHT is studied in this chapter, showing first how it is implemented and then examining how it is optimized and adapted to particular types of image data. This requires us to go back to first principles, taking spatial matched filtering as a starting point.

Having developed the relevant theory, it is applied to the important case of ellipse detection, showing in particular how computational load may be minimized. Finally, the computational problems of the GHT and HT are examined more generally.

13.2 THE GENERALIZED HOUGH TRANSFORM

This section shows how the standard Hough technique is generalized so that it can detect arbitrary shapes. In principle, it is trivial to achieve this. First, we need to select a localization point L within a template of the idealized shape. Then, we need to arrange such that, instead of moving from an edge point a *fixed* distance R directly along the local edge normal to arrive at the center, as for circles, we move an appropriate *variable* distance R in a variable direction φ so as to arrive at L : R and φ are now functions of the local edge normal direction θ (Fig. 13.1). Under these circumstances, votes will peak at the preselected object localization point L . The functions $R(\theta)$ and $\varphi(\theta)$ can be stored analytically in the computer algorithm, or for completely arbitrary shapes they may be stored as lookup tables. In either case, the scheme is beautifully simple in principle but two complications arise in practice. The first arises because some shapes have features such as concavities and holes, so several values of R and φ are required for certain values of θ (Fig. 13.2). The second arises because we are going from an isotropic shape (a circle) to an anisotropic shape, which may be in a completely arbitrary orientation.

To cope with the first of these complications, the lookup table (usually called the “ R -table”) must contain a list of the positions \mathbf{r} , relative to L , of all points on

$$\begin{array}{ccc} \theta & R & L \\ & \varphi & \end{array}$$

FIGURE 13.1

Computation of the generalized Hough transform.

**FIGURE 13.2**

A shape exhibiting a concavity: certain values of θ correspond to several points on the boundary and hence require several values of R and φ —as for points: P_1 and P_2 .

the boundary of the object for each possible value of edge orientation θ (or a similar effect must be achieved analytically). Then, on encountering an edge fragment in the image whose orientation is θ , estimates of the position of L may be obtained by moving a distance (or distances) $\mathbf{R} = -\mathbf{r}$ from the given edge fragment. Clearly, if the R -table has multivalued entries (i.e., several values of \mathbf{r} for certain values of θ), only one of these entries (for given θ) can give a correct estimate of the position of L. However, at least the method is guaranteed to give optimum sensitivity, since all relevant edge fragments contribute to the peak at L in parameter space. This property of optimal sensitivity reflects the fact that the GHT is a form of spatial matched filter: it is analyzed in more detail below.

The second complication arises because any shape other than circle is anisotropic. Since in most applications (including industrial applications such as automated assembly) object orientations are initially unknown, the algorithm has to obtain its own information on object orientation. This means adding an extra dimension in parameter space (Ballard, 1981). Then each edge point contributes a vote in each plane in parameter space at a position given by that expected for an object of given shape and orientation. Finally, the whole of parameter space is searched for peaks, the highest points indicating both the locations of objects and their orientations. Clearly, if object size is also a parameter, the problem becomes far worse but this complication is ignored here (although the method described in Section 12.3 is clearly relevant).

The changes made in proceeding to the GHT leave it just as robust as the HT circle detector described previously. This gives an incentive to improve the GHT so as to limit the computational problems in practical situations. In particular, the size of the parameter space must be cut down drastically both to save storage and to curtail the associated search task. Considerable ingenuity has been devoted to devising alternative schemes and adaptations to achieve this. Important cases are those of ellipse detection and polygon detection, and in each of these, definite advances have been made: ellipse detection is covered in Chapter 12 and for polygon detection, see Davies (1989a). Here we proceed with some more basic studies of the GHT.

13.3 SETTING UP THE GENERALIZED HOUGH TRANSFORM— SOME RELEVANT QUESTIONS

The next few sections explore the theory underpinning the GHT, with the aim of clarifying how to optimize it systematically for specific circumstances. It is relevant to ask what is happening when a GHT is being computed. Although the HT has been shown to be equivalent to template matching (Stockman and Agrawala, 1977) and also to spatial matched filtering (Sklansky, 1978), further clarification is required. In particular, the following three problems (Davies, 1987a) need to be addressed:

1. *The parameter space weighting problem:* In introducing the GHT, Ballard mentioned the possibility of weighing points in parameter space according to the magnitudes of the intensity gradients at the various edge pixels. But when should gradient weighting be used in preference to uniform weighting?
2. *The threshold selection problem:* When using the GHT to locate an object, edge pixels are detected and used to compute candidate positions for the localization point L (see [Section 13.2](#)). To achieve this, it is necessary to threshold the edge gradient magnitude. How should the threshold be chosen?
3. *The sensitivity problem:* Optimum sensitivity in detecting objects does not automatically provide optimum sensitivity in locating objects, and vice versa. How should the GHT be optimized for these two criteria?

To understand the situation and solve these problems, it is necessary to go back to first principles. Section 13.4 starts discussion on this.

13.4 SPATIAL MATCHED FILTERING IN IMAGES

To discuss the questions posed in [Section 13.3](#), it is necessary to analyze the process of spatial matched filtering. In principle, this is the ideal method of detecting objects, since it is well known (Rosie, 1966) that a filter that is matched to a given signal detects it with optimum signal-to-noise ratio under white noise¹ conditions (North, 1943; Turin, 1960). (For a more recent discussion of this topic, see Davies (1993).)

Mathematically, using a matched filter is identical to correlation with a signal (or “template”) of the same shape as the one to be detected (Rosie, 1966). Here “shape” is a general term meaning the amplitude of the signal as a function of time or spatial location.

¹White noise is noise that has equal power at all frequencies. In image science, white noise is understood to have equal power at all *spatial* frequencies. The significance of this is that noise at different pixels is completely uncorrelated but is subject to the same grayscale probability distribution, i.e., it has potentially the same range of amplitudes at all pixels.

When applying correlation in image analysis, changes in background illumination cause large changes in signal from one image to another and from one part of an image to another. The varying background level prevents straightforward peak detection in convolution space. The matched filter optimizes signal-to-noise ratio only in the presence of white noise. This is likely to be a good approximation in the case of radar signals, whereas this is not generally true in the case of images. For ideal detection, the signal should be passed through a “noise-whitening filter” (Turin, 1960), which in the case of objects in images is usually some form of high-pass filter: this must be applied prior to correlation analysis. However, this is likely to be a computationally expensive operation.

If we are to make correlation work with near optimal sensitivity but without introducing a lot of computation, other techniques must be employed. In the template matching context, the following possibilities suggest themselves:

1. Adjust templates so that they have a mean value of zero to suppress the effects of varying levels of illumination in first order.
2. Break up templates into a number of smaller templates each having a zero mean. Then as the sizes of subtemplates approach zero, the effects of varying levels of illumination will tend to zero in second order.
3. Apply a threshold to the signals arising from each of the subtemplates so as to suppress those that are less than the expected variation in signal level.

If these possibilities fail, only two further strategies appear to be available:

1. Readjust the lighting system—an important option in industrial inspection applications, although it may give little improvement when a number of objects can cast shadows or reflect light over each other.
2. Use a more “intelligent” (e.g., context sensitive) object detection algorithm, although this will almost certainly be computation intensive.

13.5 FROM SPATIAL MATCHED FILTERS TO GENERALIZED HOUGH TRANSFORMS

To proceed, we note that items 1–3 listed in [Section 13.3](#) essentially amount to a specification of the GHT. First, breaking up the templates into small subtemplates each having a zero mean and then thresholding them are analogous, and in many cases identical, to a process of edge detection (see, e.g., the templates used in the Sobel and similar operators). Next, locating objects by peak detection in parameter space clearly corresponds to the process of reconstructing whole template information from the subtemplate (edge location) data. What is important here is that these ideas reveal how the GHT is related to the spatial matched filter. Basically, *the GHT can be described as a spatial matched filter that has been modified, with the effect of including integral noise whitening, by breaking down*

the main template into small zero-mean templates and thresholding the individual responses before object detection.

Small templates do not permit edge orientation to be estimated as accurately as large ones. Although the Sobel edge detector is in principle accurate to about 1° (see Chapter 5), there is a deleterious effect if the edge of the object is fuzzy. In such a case, it is not possible to make the subtemplates very small, and an intermediate size should be chosen that gives a suitable compromise between accuracy and sensitivity.

Employing zero-mean templates results in the absolute signal level being reduced to zero and only local relative signal levels being retained. Thus, the GHT is not a true spatial matched filter: in particular, it suppresses the signal from the bulk of the object, retaining only that which is near its boundary. As a result, the GHT is highly sensitive to object position but is not optimized for object detection.

Thresholding of subtemplate responses has much the same effect as employing zero-mean templates, although it may remove a small proportion of the signal giving positional information. This makes the GHT even less like an ideal spatial matched filter and further reduces the sensitivity of object detection. The thresholding process is particularly important in the present context since it provides a means of saving computational effort *without losing significant positional information*. On its own this characteristic of the GHT would correspond to a type of perimeter template around the outside of an object (see Fig. 13.3). This must not be taken as excluding all of the interior of the object, since any high-contrast edges within the object will facilitate location.

(a) (b)

0 0

FIGURE 13.3

The idea of a perimeter template: both the original spatial matched filter template (a) and the corresponding “perimeter template” (b) have a zero mean (see text). The lower illustrations show the cross-sections along the dashed lines.

13.6 GRADIENT WEIGHTING VERSUS UNIFORM WEIGHTING

The first problem described in [Section 13.3](#) is that of how best to weight plots in parameter space in relation to the respective edge gradient magnitudes. To find an answer to this problem, it should now only be necessary to go back to the spatial matched filter case to find the ideal solution, and then to determine the corresponding solution for the GHT in the light of the discussion in [Section 13.4](#). First, note that the responses to the subtemplates (or to the perimeter template) are proportional to edge gradient magnitude. With a spatial matched filter, signals are detected optimally by templates of the same shape. Each contribution to the spatial matched filter response is then proportional to the local magnitude of the signal and to that of the template. In view of the correspondence between (a) using a spatial matched filter to locate objects by looking for peaks in convolution space and (b) using a GHT to locate objects by looking for peaks in parameter space, we should use weights proportional to the gradients of the edge points *and* the *a priori* edge gradients.

There are two ways in which the choice of weighting is important. First, the use of uniform weighting implies that all edge pixels whose gradient magnitudes are above threshold will effectively have them reduced to the threshold value, so that the signal will be curtailed: this can mean that the signal-to-noise ratio of high-contrast objects will be reduced significantly. Second, the widths of edges of high-contrast objects will be broadened in a crude way by uniform weighting (see [Fig. 13.4](#)) but under gradient weighting this broadening will be controlled, giving a roughly Gaussian edge profile. Thus, the peak in parameter space will be narrower and more rounded, and the object reference point L can be located more easily and with greater accuracy. This effect is visible in [Fig. 13.5](#), which also shows the relatively increased noise level that results from uniform weighting.

Note also that low gradient magnitudes correspond to edges of poorly known location, whereas high values correspond to sharply defined edges. Thus, the accuracy of the information relevant to object location is proportional to the magnitude of the gradient at each of the edge pixels, and appropriate weighting should therefore be used.

13.6.1 Calculation of Sensitivity and Computational Load

The aim of this subsection is to underline the above-described ideas by working out formulae for sensitivity and computational load. It is assumed that p objects of size around $n \times n$ are being sought in an image of size $N \times N$.

Correlation requires N^2n^2 operations to compute the convolutions for all possible positions of the object in the image. Using the perimeter template, the number of basic operations is reduced to $\sim N^2n$, corresponding to the reduced number of pixels in the template. The GHT requires $\sim N^2$ operations to locate the edge pixels, plus a further $\sim pn$ operations to accumulate the points in parameter space.

**FIGURE 13.4**

Effective gradient magnitude as a function of position within a section across an object of moderate contrast, thresholded at a fairly low level: (a) gradient magnitude for original image data and gradient thresholding level; (b) uniform weighting: the effective widths of edges are broadened rather crudely, adding significantly to the difficulty of locating the peak in parameter space; (c) gradient weighting: the position of the peak in parameter space can be estimated in a manner that is basically limited by the shape of the gradient profile for the original image data.

The situation for sensitivity is rather different. With correlation, the results for n^2 pixels are summed, giving a signal proportional to n^2 , although the noise (assumed to be independent at every pixel) is proportional to n : this is because of the well-known result that the noise powers of various independent noise components are additive (Rosie, 1966). Overall, this results in the signal-to-noise ratio being proportional to n . The perimeter template possesses only $\sim n$ pixels, and here the overall result is that the signal-to-noise ratio is proportional to \sqrt{n} . The situation for the GHT is inherently identical to that for the perimeter template method, so long as plots in parameter space are weighted proportional to edge gradient g multiplied by *a priori* edge gradient G . It is now necessary to compute the constant of proportionality α . Take s as the average signal, equal to the intensity (assumed to be roughly uniform) over the body of the object, and S as the magnitude of a full matched filter template. In the same units, g (and G) is the magnitude of the signal within the perimeter template. Then $\alpha = 1/sS$. This means that the perimeter template method and the GHT method lose sensitivity for two reasons—first they look at less of the available signal and second they look where the signal is low. For a high value of gradient magnitude, which occurs for a step edge (where most of the variation in intensity takes place within the range of 1 pixel), the values of g and G saturate out, so that they are nearly equal to s and S (see Fig. 13.6). Under these conditions, the perimeter template method and the GHT have sensitivities that depend only on the value of n .

(a)

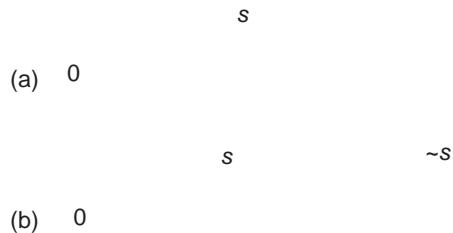
(b)

(c)

FIGURE 13.5

Results of applying the two types of weighting to a real image: (a) original image; (b) results in parameter space for uniform weighting; and (c) results for gradient weighting. The peaks (which arise from the outer edges of the washer) are normalized to the same levels in the two cases: the increased level of noise in (b) is readily apparent. In this example, the gradient threshold is set at a low level (around 10% of the maximum level) so that low-contrast objects can also be detected.

Table 13.1 summarizes the situation discussed above. The oft-quoted statement that the computational load of the GHT is proportional to the number of perimeter pixels, rather than to the much greater number of pixels within the body of an object, is only an approximation. In addition, this saving is not obtained without cost: in particular, the sensitivity (signal-to-noise ratio) is reduced (at best) as the square root of object area/perimeter (note that area and perimeter are measured in the same units, so it is valid to find their ratio).

**FIGURE 13.6**

Effect of edge gradient on perimeter template signal: (a) low edge gradient: signal is proportional to gradient and (b) high edge gradient: signal saturates at value of s .

Table 13.1 Formulae for Computational Load and Sensitivity^a

	Template Matching	Perimeter Template Matching	Generalized Hough Transform
Number of operations	$O(N^2n^2)$	$O(N^2n)$	$O(N^2) + O(pn)$
Sensitivity	$O(n)$	$O\left(\frac{\sqrt{n}gG}{sS}\right)$	$O\left(\frac{\sqrt{n}gG}{sS}\right)$
Maximum sensitivity ^b	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$

^aThis table gives formulae for computational load and sensitivity when p objects of size $n \times n$ are sought in an image of size $N \times N$. The intensity of the image within the whole object template is taken as s and the value for the ideal template is taken as S : corresponding values for intensity gradient within the perimeter template are g and G .

^bMaximum sensitivity refers to the case of a step edge, for which $g \approx s$ and $G \approx S$ (see Fig. 13.6).

Finally, the absolute sensitivity for the GHT varies as gG . As contrast changes so that $g \rightarrow g'$, we see that $gG \rightarrow g'G$, i.e., sensitivity changes by a factor of g'/g . Hence, theory predicts that sensitivity is proportional to contrast. Although this result might have been anticipated, we now see that it is valid only under conditions of gradient weighting.

13.7 SUMMARY

The above sections examined the GHT and found a number of factors involved in optimizing it, as follows.

1. Each point in parameter space should be weighted in proportion to the intensity gradient at the edge pixel giving rise to it, and in proportion to the a

priori gradient, if sensitivity is to be optimized, particularly for objects of moderate-to-high contrast.

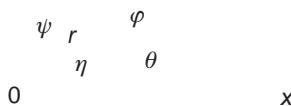
2. The ultimate reason for using the GHT is to save computation. The main means by which this is achieved is by ignoring pixels having low magnitudes of intensity gradient. If the threshold of gradient magnitude is set too high, fewer objects are in general detected; if it is set too low, computational savings are diminished. Suitable means are required for setting the threshold but little reduction in computation is possible if the highest sensitivity in a low-contrast image is to be retained.
3. The GHT is inherently optimized for the location of objects in an image but is not optimized for the detection of objects. This means that it may miss low-contrast objects, which are detectable by other methods that take the whole area of an object into account. However, this consideration is often unimportant in applications where signal-to-noise ratio is less of a problem than finding objects quickly in an uncluttered environment.

Overall, it is clear that the GHT is a spatial matched filter only in a particular sense, and as a result it has suboptimal sensitivity. The main advantage of the technique is that it is highly efficient, overall computational load in principle being proportional to the relatively few pixels on the perimeters of objects rather than to the much greater numbers of pixels within them. In addition, by concentrating on the boundaries of objects, the GHT retains its power to locate objects accurately. It is thus important to distinguish clearly between sensitivity in *detecting* objects and sensitivity in *locating* them.

13.8 USE OF THE GHT FOR ELLIPSE DETECTION

It has already been seen that when the GHT is used to detect anisotropic objects, there is an intrinsic need to employ a large number of planes in parameter space. However, it is shown below that by accumulating the votes for all possible orientations in a *single* plane in parameter space, significant savings in computation can sometimes be made. Basically, the idea is largely to reduce the considerable storage requirements of the GHT by using only one instead of 360 planes in parameter space while significantly reducing the computation involved in the final search for peaks. Such a scheme could have concomitant disadvantages such as the production of spurious peaks, and this aspect will have to be examined carefully.

To achieve these aims, it is necessary to analyze the shape of the point spread function (PSF) to be accumulated for each edge pixel. To demonstrate this, we take the case of ellipses of unknown orientation. We start by taking a general edge fragment at a position defined by ellipse parameter ψ and deducing the bearing of the center of the ellipse relative to the local edge normal (Fig. 13.7).

y**FIGURE 13.7**

Geometry of an ellipse and its edge normal.

Working first in an ellipse-based axis system, for an ellipse with semimajor and semiminor axes a and b , respectively, it is clear that:

$$x = a \cos \psi \quad (13.1)$$

$$y = b \sin \psi \quad (13.2)$$

Hence

$$\frac{dx}{d\psi} = -a \sin \psi \quad (13.3)$$

$$\frac{dy}{d\psi} = b \cos \psi \quad (13.4)$$

giving

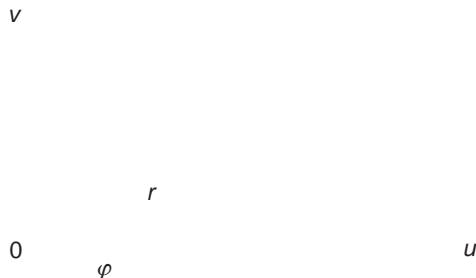
$$\frac{dy}{dx} = -\left(\frac{b}{a}\right) \cot \psi \quad (13.5)$$

Hence, the orientation of the edge normal is given by:

$$\tan \theta = \left(\frac{a}{b}\right) \tan \psi \quad (13.6)$$

At this point, we wish to deduce the bearing of the center of the ellipse relative to the local edge normal. From Fig. 13.7:

$$\varphi = \theta - \eta \quad (13.7)$$

**FIGURE 13.8**

Geometry for finding the PSF for ellipse detection by forming the locus of the centers of ellipses touching a given edge fragment.

where

$$\tan \eta = \frac{y}{x} = \left(\frac{b}{a} \right) \tan \psi \quad (13.8)$$

and

$$\begin{aligned} \tan \varphi &= \tan(\theta - \eta) \\ &= \frac{\tan \theta - \tan \eta}{1 + \tan \theta \tan \eta} \end{aligned} \quad (13.9)$$

Substituting for $\tan \theta$ and $\tan \eta$, and then rearranging, gives:

$$\tan \varphi = \frac{(a^2 - b^2)}{2ab} \sin 2\psi \quad (13.10)$$

In addition:

$$r^2 = a^2 \cos^2 \psi + b^2 \sin^2 \psi \quad (13.11)$$

To obtain the PSF for an ellipse of unknown orientation, we now simplify matters by taking the current edge fragment to be at the origin and orientated with its normal along the u -axis (Fig. 13.8). The PSF is then the locus of all possible positions of the center of the ellipse. To find its form, it is merely required to eliminate ψ between Eqs. (13.10) and (13.11). This is facilitated by re-expressing r^2 in double angles (the significance of double angles lies in the 180° rotation symmetry of an ellipse):

$$r^2 = \frac{a^2 + b^2}{2} + \frac{a^2 - b^2}{2} \cos 2\psi \quad (13.12)$$

After some manipulation, the locus is obtained as:

$$r^4 - r^2(a^2 + b^2) + a^2b^2 \sec^2 \varphi = 0 \quad (13.13)$$

which can, in the edge-based coordinate system, also be expressed in the form:

$$v^2 = (a^2 + b^2) - u^2 - a^2b^2/u^2 \quad (13.14)$$

In fact, this is a complex and variable shape, as shown in Fig. 13.9, although for ellipses of low eccentricity, the PSF approximates to an ellipse. This is seen by defining two new parameters:

$$c = \frac{(a + b)}{2} \quad (13.15)$$

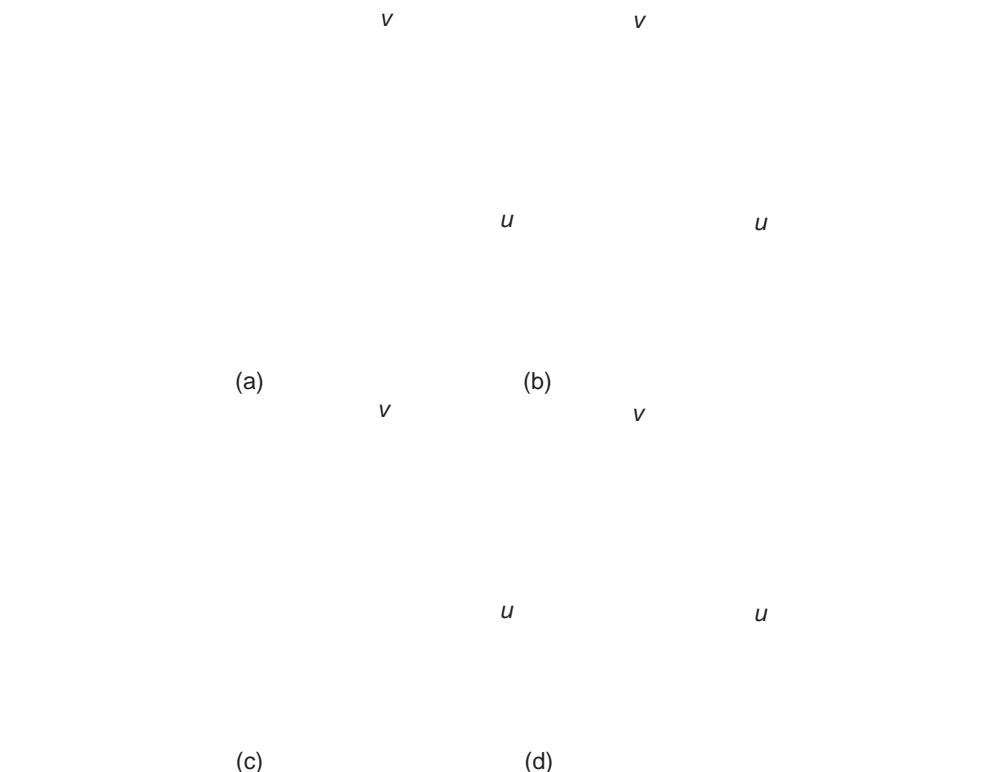


FIGURE 13.9

Typical PSF shapes for detection of ellipses with various eccentricities: (a) ellipse with $a/b = 21.0$ and $c/d = 1.1$, (b) ellipse with $a/b = 5.0$, $c/d = 1.5$, (c) ellipse with $a/b = 2.0$ and $c/d = 3.0$, (d) ellipse with $a/b = 1.4$ and $c/d = 6.0$. Note how the PSF shape approaches a small ellipse of aspect ratio 2.00 as eccentricity tends to zero. The semimajor and semiminor axes of the PSF are $2d$ and d , respectively.

$$d = \frac{(a - b)}{2} \quad (13.16)$$

and taking an approximation of small d , the locus is obtained in the form:

$$\frac{(u + c)^2}{d^2} + \frac{v^2}{4d^2} = 1 \quad (13.17)$$

As stated above, this approximates to an ellipse, and has semimajor axis $2d$ and semiminor axis d . However, this approximation has restricted validity, applying only when $d \gtrsim 0.1c$, so that $a \gtrsim 1.2b$. However, in practice, where ellipses are small and the PSF is only a few pixels across, it is a reasonable approximation to insist only that $a < 2b$.

Implementation is simplified since a universal lookup table (ULUT) for ellipse detection can be compiled, which is independent of the size and eccentricity of the ellipse to be detected so long as the eccentricity is not excessive. Since ellipses are common features in industrial and other applications—arising both from elliptical objects and from oblique views of circles—this factor should be an important consideration in many applications. Thus, a single ULUT is compiled and stored and it then needs only to be scaled and positioned to produce the PSF in a given instance of ellipse detection.

13.8.1 Practical Details

Having constructed a ULUT for ellipse detection, the detection algorithm has to scale it, position it, and rotate it so that points can be accumulated in parameter space. *A priori*, it would be imagined that a considerable amount of trigonometric computation is involved in this process. However, it is possible to avoid calculating angles directly (e.g., using the arctan function) by always working with sines and cosines; this is rendered possible partly because such edge orientation operators as the Sobel give two components (g_x, g_y) for the intensity gradient vector (this has already been seen to happen in several line and circle detection schemes—see Chapters 11 and 12). Hence, a lot of computation can be saved.

[Figure 13.10](#) shows the result of testing the above scheme on an image of some O-rings lying on a slope of arbitrary direction, whereas [Fig. 13.11](#) shows the result obtained for an elliptical object; the two cases used PSFs containing 50 and 100 votes, respectively. In [Fig. 13.10](#), the O-rings are found accurately and with a fair degree of robustness, i.e., despite overlapping and partial occlusion (up to 40% in one case). In several cases, incidental transforms from points on the inner edges of the O-rings overlap other transforms from points on the outer edges, although only the latter are actually employed usefully here for peak finding. Hence, the scheme is able to overcome problems resulting from additional clutter in parameter space.

[Figure 13.10](#) also shows the arrangement of points in parameter space that results from applying the PSF to every edge point on the boundary of an

(a) (b)

FIGURE 13.10

Applying the PSF to detection of tilted circles: (a) off-camera 128×128 image of a set of circular O-rings on a 45° slope of arbitrary direction; (b) transform in parameter space: note the peculiar shape of the ellipse transform, which is close to a “four-leaf clover” pattern. (a) also indicates the positions of the centers of the O-rings as located from (b): accuracy is limited by the presence of noise, shadows, clutter, and available resolution, to an overall standard deviation of about 0.6 pixels.

(a) (b)

FIGURE 13.11

Applying the PSF to detection of elliptical objects: (a) off-camera 128×128 image of an elliptical bar of soap of arbitrary orientation; (b) transform in parameter space: in this case, the clover-leaf pattern is better resolved. Accuracy of location is limited partly by distortions in the shape of the object but the peak location procedure results in an overall standard deviation of the order of 0.5 pixels.

ellipse: the pattern is somewhat clearer in Fig. 13.11. In either case, it is seen to contain a high degree of structure (curiously, the votes seem to form approximate “four-leaved clover” patterns). For an ideal transform, there would be no structure apart from the main peak, and all points on the PSF *not* falling on the peak at the center of the ellipse would be randomly distributed nearby. Nevertheless, the peak at the center is very well defined and shows that this compressed form of GHT represents a viable solution.

13.9 COMPARING THE VARIOUS METHODS

This section briefly compares the computational loads for the methods of ellipse detection discussed in [Section 13.8](#) and in Chapter 12. To make fair comparisons, we concentrate on ellipse detection *per se* and ignore any additional procedures concerned with (a) finding other ellipse parameters, (b) distinguishing ellipses from other shapes, or (c) separating concentric ellipses. We start by examining the GHT method and the diameter bisection method.

First, suppose that an $N \times N$ pixel image contains p ellipses of identical size given by the parameters a , b , c , and d defined in [Section 13.8](#). By ignoring noise and general background clutter, we shall be favoring the diameter bisection method, as will be seen below. Next, the discussion is simplified by supposing that the computational load resides mainly in the calculation of the positions at which votes should be accumulated in parameter space—the effort involved in locating edge pixels and in locating peaks in parameter space is much smaller.

Under these circumstances, the load for the GHT method may be approximated by the product of the number of edge pixels and the number of points per edge pixel that have to be accumulated in parameter space, the latter being equal to the number of points on the PSF. Hence, the load is proportional to:

$$\begin{aligned} L_G &\approx \frac{p \times 2\pi c \times 2\pi(2d + d)}{2} = 6\pi^2 pcd \\ &\approx 60pcd \end{aligned} \quad (13.18)$$

where the ellipse has been taken to have relatively low eccentricity so that the PSF itself approximates to an ellipse of semiaxes $2d$ and d .

For the diameter bisection method, the actual voting is a minor part of the algorithm—as indeed it is in the GHT method (see the snippet of code listed in [Table 13.1](#)). In either case, most of the computational load concerns edge orientation calculations or comparisons. Assuming that these calculations and comparisons involve similar inherent effort, it is fair to assess the load for the diameter bisection method as:

$$L_D \approx p \times 2\pi c C_2 \approx \frac{(2\pi pc)^2}{2} \approx 20p^2 c^2 \quad (13.19)$$

Hence

$$\frac{L_D}{L_G} \approx \frac{pc}{3d} \quad (13.20)$$

when a is close to b , as for a circle, $L_G \rightarrow 0$ and then the diameter bisection method becomes a poor option. However, in some cases, it is found that a is close to $2b$, so that c is close to $3d$. The ratio of the loads then becomes:

$$\frac{L_D}{L_G} \approx p \quad (13.21)$$

It is possible that p will be as low as 1 in some cases: however, such cases are likely to be rare and are offset by applications where there is significant background image clutter and noise, or where all p ellipses have other edge detail giving irrelevant signals that can be considered as a type of self-induced clutter (see the O-ring example in Fig. 13.10).

It is also possible that some of the pairs of edge points in the diameter bisection method can be excluded before they are considered, e.g., by giving every edge point a range of interaction related to the size of the ellipses. This would tend to reduce the computational load by a factor of the order of (but not as small as) p . However, the computational overhead required for this would not be negligible.

Overall, the GHT method should be significantly faster than the diameter bisection method in most real applications, the diameter bisection method being at a definite disadvantage when image clutter and noise are strong. By comparison, the chord–tangent method always requires more computation than the diameter bisection method, since not only does it examine every pair of edge points but also it generates a *line* of votes in parameter space for each pair.

Against these computational limitations, the different characteristics of the methods must be noted. First, the diameter bisection method is not particularly discriminating, in that it locates many symmetrical shapes, as remarked earlier. The chord–tangent method is selective for ellipses but is not selective about their size or eccentricity. The GHT method is selective about all of these factors. These types of discriminability, or lack of it, can turn out to be advantageous or disadvantageous, depending on the application: hence, we do no more here than draw attention to these different properties. It is also relevant that the diameter bisection method is rather less robust than the other methods. This is so since if one edge point of an antiparallel pair is not detected, then the other point of the pair cannot contribute to detection of the ellipse—a factor that does not apply for the other two methods since they take all edge information into account.

13.10 FAST IMPLEMENTATIONS OF THE HOUGH TRANSFORM

The foregoing sections have shown that the GHT requires considerable computation. The problem arises particularly in respect of the number of planes needed in parameter space to accommodate transforms for different object orientations and sizes. Clearly, significant improvements in speed are needed before the GHT can achieve its potential in practical instances of arbitrary shapes. This section considers some important developments in this area.

The source of the computational problems in the HT is the huge size the parameter space can take. Typically, a single plane parameter space has much the same size as an image plane (this will normally be so in those instances where parameter space is congruent to image space), but when many planes are required

to cope with various object orientations and sizes, the number of planes is likely to be multiplied by a factor of around 300 for each extra dimension. Hence, the total storage area will then involve some 10,000 million accumulator cells. Clearly, reducing the resolution might just make it possible to bring this down to 100 million cells, although accuracy will start to suffer. Further, when the HT is stored in such a large space, the search problem involved in locating significant peaks becomes formidable.

Fortunately, data are not stored at all uniformly in parameter space and this provides a key to solving both the storage problem and the subsequent search problem. Indeed, the fact that parameter space is to be searched for the most prominent peaks—in general, the highest and the sharpest ones—means that the process of detection can start during accumulation. Furthermore, initial accumulation can be carried out at relatively low resolution, and then resolution can be increased locally as necessary in order to provide both the required accuracy and the separation of nearby peaks. In this context, it is natural to employ a binary splitting procedure, i.e., to repeatedly double the resolution locally in each of the dimensions of parameter space (e.g., the x dimension, the y dimension, the orientation dimension, and the size dimension, where these exist) until resolution is sufficient: a tree structure may conveniently be used to keep track of the situation.

Such a method (the “fast” HT) was developed by Li and Lavin (1986) and Li et al. (1985). Illingworth and Kittler (1987) found this method to be insufficiently flexible in dealing with real data and produced a revised version (the “adaptive” HT), which permits each dimension in parameter space to change its resolution locally in tune with whatever the data demand, rather than insisting on some previously devised rigid structure. In addition, they employed a 9×9 accumulator array at each resolution rather than the theoretically most efficient 2×2 array, since this was found to permit better judgments to be made on the nature of the local data. This approach seemed to work well with fairly clean images but later, doubts were cast on its effectiveness with complex images (Illingworth and Kittler, 1988). The most serious problem to be overcome here is that at coarse resolutions, extended patterns of votes from several objects can overlap, giving rise to spurious peaks in parameter space. Since all of these peaks have to be checked at all relevant resolutions, the whole process can consume more computation than it saves. Clearly, optimization in multiresolution peak-finding schemes is complex² and data-dependent, and so discussion is curtailed here. The reader is

²Ultimately, the problem of system optimization for analysis of complex images is a difficult one, since in conditions of low signal-to-noise ratio even the eye may find it difficult to interpret an image and may “lock on” to an interpretation that is incorrect. Note that in general image interpretation work, there are many variables to be optimized—sensitivity, efficiency/speed, storage, accuracy, robustness, and so on—and it is seldom valid to consider any of these individually. Often tradeoffs between just two such variables can be examined and optimized but in real situations multivariable tradeoffs should be considered. This is a complex task and it is one of the purposes of this book to show clearly the serious nature of these types of optimization problem, although at the same time it can only guide the reader through a limited number of basic optimization processes.

referred to the original research papers (Li et al., 1985; Li and Lavin, 1986; Illingworth and Kittler, 1987) for implementation details.

An alternative scheme is the hierarchical HT (Princen et al., 1989a); so far it has been applied only to line detection. The scheme can most easily be envisaged by considering the foot-of-normal method of line detection described in Section 11.3. Rather small subimages of just 16×16 pixels are taken first of all, and the foot-of-normal positions determined. Then each foot-of-normal is tagged with its orientation and an identical HT procedure is instigated to generate the foot-of-normal positions for line segments in 32×32 subimages, this procedure being repeated as many times as necessary until the whole image is spanned at once. The paper by Princen et al. discusses the basic procedure in detail and also elaborates necessary schemes for systematically grouping separate line segments into full-length lines in a hierarchical context. An interesting detail is that successful operation of the method requires subimages with 50% overlap to be employed at each level. The overall scheme appears to be as accurate and reliable as the basic HT method but does appear to be faster and to require significantly less storage.

13.11 THE APPROACH OF GERIG AND KLEIN

The Gerig and Klein approach was first demonstrated in the context of circle detection but was only mentioned in passing in Chapter 12. This is because it is an important *approach* that has much wider application than merely to circle detection. The motivation for it has already been noted in Section 13.10—namely, the problem of extended patterns of votes from several objects giving rise to spurious peaks in parameter space.

Ultimately, the reason for the extended pattern of votes is that each edge point in the original image can give rise to a large number of votes in parameter space. The tidy case of detection of circles of known radius is somewhat unusual, as will be seen particularly in Chapters 12 and 17. Hence, in general *most* of the votes in parameter space are in the end unwanted and serve only to confuse. Ideally, we would like a scheme in which each edge point gives rise only to the single vote corresponding to the localization point of the particular boundary on which it is situated. Although this ideal is not initially realizable, it can be engineered by the “back-projection” technique of Gerig and Klein (1986). Here all peaks and other positions in parameter space to which a given edge point contributes are examined, and a new parameter space is built in which only the vote at the strongest of these peaks is retained (there is the greatest probability, but no certainty, that it belongs to the *largest* such peak). This second parameter space thus contains no extraneous clutter and weak peaks are hence found much more easily: this gives objects with highly fragmented or occluded boundaries much more chance of being detected. Overall, the method avoids many of the problems associated with setting arbitrary thresholds on peak height—in principle, no thresholds are required in this approach.

The scheme can be applied to any HT detector that throws multiple votes for each edge point. Thus, it appears to be widely applicable and is capable of improving robustness and reliability at an intrinsic expense of approximately doubling computational effort (however, set against this is the relative ease with which peaks can be located—a factor which is highly data-dependent). Note that the method is another example in which a two-stage process is used for effective recognition.

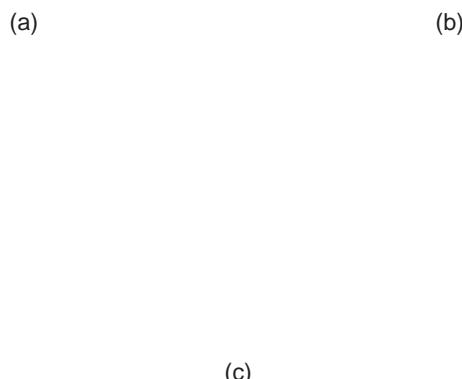
Other interesting features of the Gerig and Klein method must be omitted here for reasons of space, except to note that, rather oddly, the published scheme ignores edge orientation information as a means of reducing computation.

13.12 CONCLUDING REMARKS

The Hough transform was introduced in Chapter 11 as a line detection scheme and then used in Chapter 12 for detecting circles and ellipses. In both chapters, it appeared as a rather cunning method for aiding object detection; although it was seen to offer various advantages, particularly in its robustness in the face of noise and occlusion, there appeared to be no real significance in its rather novel voting scheme. The present chapter has shown that, far from being a trick method, the HT is much more general an approach than originally supposed: indeed, it embodies the properties of the spatial matched filter and is therefore capable of close-to-optimal sensitivity for object detection. However, this does not prevent its implementation from entailing considerable computational load, and significant effort and ingenuity have been devoted to overcoming this problem, both in general and in specific cases. The general case is tackled by the schemes discussed in Sections 13.9 and 13.10. It is important not to underestimate the value of specific solutions, both because such shapes as lines, circles, ellipses, and polygons cover a large proportion of (or approximations to) manufactured objects and because methods for coping with specific cases have a habit (as for the original HT) of becoming more general as workers see possibilities for developing the underlying techniques.

Finally, to further underline the generality of the GHT, it has also been used for optimal location of lines of known length, by emulating a spatial matched filter detector; this result has been applied to the optimal detection of polygons and of corners: for an example of the latter, see Fig. 13.12 (Davies, 1988a, 1989a). For further discussion and critique of the whole HT and GHT approach, see Chapter 27.

Although the Hough transform may appear to have a somewhat arbitrary design, this chapter has shown that it has solid roots in matched filtering, which in turn implies that votes should be gradient weighted for optimal sensitivity. The chapter also contrasts three methods for ellipse detection, showing how computational load may be estimated and minimized.

**FIGURE 13.12**

Example of the generalized Hough transform approach to corner detection. (a) Original image of a biscuit (128×128 pixels, 64 gray levels). (b) Transform with lateral displacement around 22% of the shorter side. (c) Image with transform peaks located (white dots) and idealized corner positions deduced (black dots). The lateral displacement employed here is close to the optimum for this type of object.

Source: (a,b) © IEE 1988, (c) © Unicom 1988

13.13 BIBLIOGRAPHICAL AND HISTORICAL NOTES

Although the HT was introduced as early as 1962, a number of developments—including especially those of Merlin and Farber (1975) and Kimme et al. (1975)—were required before the GHT was developed (Ballard, 1981). By that time, the HT was already known to be formally equivalent to template matching (Stockman and Agrawala, 1977) and to spatial matched filtering (Sklansky, 1978). However, the questions posed in [Section 13.3](#) were only answered much later (Davies, 1987a), the necessary analysis being reproduced in [Sections 13.4–13.7](#). The author's work (Davies, 1987b, 1987d, 1989b) on line detection by the GHT (not covered here) was aimed particularly at optimizing sensitivity of line detection, although deeper issues of tradeoffs between sensitivity, speed, and accuracy are also involved.

By 1985, the computational load of the HT became the critical factor preventing its more general use—particularly as it could be used for most types of

arbitrary shape detection, with well-attested sensitivity and considerable robustness. Following preliminary work by Brown (1984), with the emphasis on hardware implementations of the HT, Li et al. (1985) and Li and Lavin (1986) showed the possibility of much faster peak location by using nonuniformly quantized parameter spaces. This work was developed further by Illingworth and Kittler (1987) and others (see, e.g., Illingworth and Kittler, 1988; Princen et al., 1989a, 1989b; Davies, 1992g). An important development has been the randomized Hough transform (RHT), pioneered by Xu and Oja (1993) among others: it involves casting votes until specific peaks in parameter space become evident, thereby saving unnecessary computation.

Accurate peak location remains an important aspect of the HT approach. Properly, this is the domain of robust statistics, which handles the elimination of outliers (see Appendix A). Davies (1992f) has shown a computationally efficient means of accurately locating HT peaks, and has found why peaks sometimes appear narrower than *a priori* considerations would indicate (Davies, 1992b). Kiryati and Bruckstein (1991) have tackled aliasing effects, which can arise with the HT, and which have the effect of cutting down accuracy.

Over time, the GHT approach has been broadened by geometric hashing, structural indexing, and other approaches (e.g., Lamdan and Wolfson, 1988; Gavrila and Groen, 1992; Califano and Mohan, 1994). At the same time, a probabilistic approach to the subject has been developed (Stephens, 1991), which puts it on a firmer footing. Grimson and Huttenlocher (1990) warn (possibly over-pessimistically) against the blithe use of the GHT for complex object recognition tasks, because of the false peaks that can appear in such cases. For further review of the state of the subject up to 1993, see Leavers (1993).

In various chapters of Part 2, the statement has been made that the HT³ carries out a search leading to hypotheses that should be checked before a final decision about the presence of an object can be made. However, Princen et al. (1994) show that the performance of the HT can be improved if it is itself regarded as a hypothesis testing framework: this is in line with the concept that the HT is a model-based approach to object location. Other studies have been made about the nature of the HT. In particular, Aguado et al. (2000) consider the intimate relationship between the HT and the principle of duality in shape description: the existence of this relationship underlines the importance of the HT and provides a means for a more general definition of it. Kadyrov and Petrou (2001) have developed the trace transform, which can be regarded as a generalized form of the Radon transform—itself closely related to the Hough transform.

Other workers have used the HT for affine-invariant search: Montiel et al. (2001) made an improvement to reduce the incidence of erroneous evidence in the gathered data, whereas Kimura and Watanabe (2002) made an extension for 2-D shape detection that is less sensitive to the problems of occlusion and broken

³A similar statement can be made in the case of graph matching methods such as the maximal clique approach to object location (see Chapter 14).

boundaries. Kadyrov and Petrou (2002) have adapted the trace transform to cope with affine parameter estimation.

In a generalization of the work of Atherton and Kerbyson (1999) and of Davies (1987a) on gradient weighting (see Section 13.6), Anil Bharath and his colleagues have examined how to optimize the sensitivity of the HT (private communication, 2004). Their method is particularly valuable in solving the problems of early threshold setting that limit many HT techniques. Similar sentiments come out in a different way in the work of Kesidis and Papamarkos (2000), which maintains the grayscale information throughout the transform process, thereby leading to more exact representations of the original images.

Olson (1999) has shown that localization accuracy can be improved efficiently by transferring local error information into the HT and handling it rigorously. An important finding is that the HT can be subdivided into many sub-problems without decrease in performance. This finding is elaborated in a 3-D model-based vision application where it is shown to lead to reduced false positive rates (Olson, 1998). Wu et al. (2002) extend the 3-D possibilities further by using a 3-D HT to find glasses: first a set of features are located that lie on the same plane, and this is then interpreted as the glasses' rim plane. This approach allows the glasses to be separated from the face, and then they can be located in their entirety.

van Dijck and van der Heijden (2003) develop the geometric hashing method of Lamdan and Wolfson (1988) to perform 3-D correspondence matching using full 3-D hashing. This is found to have advantages in that knowledge of 3-D structure can be used to reduce the number of votes and spurious matches. Tuytelaars et al. (2003) describe how invariant-based matching and HTs can be used to identify regular repetitions in planes appearing within visual (3-D) scenes in spite of perspective skew. The overall system has the ability to reason about consistency and is able to cope with periodicities, mirror symmetries, and reflections about a point.

13.13.1 More Recent Developments

Among the most recent developments are the following. Aragon-Camarasa and Siebert (2010) considered using the GHT for clustering SIFT feature matches. However, it turned out that a *continuous* rather than discretized HT space was needed for this application. This meant that each matched point had to be stored at the full machine precision in a Hough space consisting of a list data structure. Therefore, peak location had to take the form of standard unsupervised clustering algorithms. This was an interesting case where the intended GHT could not follow the standard voting and accumulating procedure. Assheton and Hunter (2011) also deviated sharply from the standard GHT approach when performing pedestrian detection and tracking: they used a shape-based voting algorithm based on Gaussian mixture models. The algorithm was stated to be highly effective for detecting pedestrians based on the silhouette shape. Chung et al. (2010) studied the problem of information retrieval from databases. They produced a region-based

The Three-Dimensional World

15

3-D VISION

Humans are able to employ 3-D vision with consummate ease, and according to conventional wisdom, binocular vision is the key to this success. The truth is more complex than this, and this chapter demonstrates why.

Look out for:

- what can be achieved using binocular vision.
- how the shading of surfaces can be used in place of binocular vision to achieve similar ends.
- how these basic methods provide dimensional information for 3-D scenes but do not immediately lead to object recognition.
- how the process of 3-D object recognition can be tackled by studies of 3-D geometry.

Note that this is an introductory chapter on 3-D vision, designed to give the flavor of the subject and to show its origins in human vision. It will be followed by the other four chapters (Chapters 16–19) that comprise Part 3 of this volume.

At a more detailed level, notice the importance of the epipolar line approach in solving the correspondence problem. The concept is deservedly taken considerably further in Chapter 18, in conjunction with the required mathematical formulation.

15.1 INTRODUCTION

In the foregoing chapters, it has generally been assumed that objects are essentially flat and are viewed from above in such a way that there are only three

degrees of freedom—namely, the two associated with position and a further one concerned with orientation. While this approach was adequate for carrying out many useful visual tasks, it is inadequate for interpreting outdoor or factory scenes or even for helping with quite simple robot assembly and inspection tasks. Indeed, over the past few decades a considerable amount of quite sophisticated theory has been developed and backed up by experiment, to find how scenes composed of real 3-D objects can be understood in detail.

In general, this means attempting to interpret scenes in which objects may appear in totally arbitrary positions and orientations—corresponding to six degrees of freedom. Interpreting such scenes, and deducing the translation and orientation parameters of arbitrary sets of objects, takes a substantial amount of computation—partly because of the inherent ambiguity in inferring 3-D information from 2-D images.

A variety of approaches are now available for proceeding with 3-D vision. A single chapter will be unable to describe all of them but the intention here is to provide an overview, outlining the basic principles and classifying the methods according to generality, applicability, and so on. While computer vision need not necessarily mimic the capabilities of the human eye–brain system, much research on 3-D vision has been aimed at biological modeling. This type of research shows that the human visual system makes use of a number of different methods simultaneously, taking appropriate cues from the input data and forming hypotheses about the content of a scene, progressively enhancing these hypotheses until a useful working model of what is present is produced. Thus, individual methods are not expected to work in isolation: rather, they need to provide the model generator with whatever data become available. Clearly, biological machinery of various types will lie idle for much of the time until triggered by specific input stimuli. Computer vision systems are currently less sophisticated than this and tend to be built on specific processing models, so that they can be applied efficiently to more restricted types of image data. In this chapter, we adopt the pragmatic view that particular methods need to be (or have been) developed for specific types of situation, and that they should be used only when appropriate—although some care is taken to elucidate what the appropriate types of applications are.

15.2 3-D VISION—THE VARIETY OF METHODS

One of the most obvious characteristics of the human visual system is that it employs two eyes, and it is well known to the layman that binocular (or “stereo”) vision permits depth to be discerned within a scene. However, the loss of vision that results when one eye is shut is relatively insignificant and is by no means a disqualification from driving a car or even an aeroplane. On the contrary, depth can readily be deduced in monocular vision from a plethora of cues that are buried in an image. Naturally, to achieve this, the eye–brain system is able to call

on a huge amount of pre-stored data about the physical world and about the types of object in it, be they man-made or natural entities. For example, the size of any car being viewed is strongly constrained; likewise, most objects have highly restricted sizes, both absolutely and in their depths relative to their frontal dimensions. Nevertheless, in a single view of a scene, it is normally impossible to deduce absolute sizes—all the objects and their depths can be scaled up or down by arbitrary factors and this cannot be discerned from a monocular view.

While it is clear that the eye–brain system makes use of a huge database relating to the physical world, there is much that can be learned with negligible prior knowledge, even from a single monocular view. The main key to this is the “shape from shading” concept. For 3-D shape to be deducible from shading information (i.e., from the grayscale intensities in an image), something has to be known about how the scene is lit—the simplest situation being when the scene is illuminated by a single point light source at a known position. Note that indoors a single overhead tungsten light is still the most usual illuminator, while outdoors the sun performs a similar function. In either case, an obvious result is that a single source will illuminate one part of an object and not another—which then remains in shadow—and parts that are orientated in various ways relative to the source and the observer appear with different brightness values, so that orientation can in principle be deduced. In fact, as will be seen below, deduction of orientation and position is not at all trivial and may even be ambiguous. Nevertheless, successful methods have been developed for carrying out this task. One problem that often arises is that the position of the light source is unknown but this information can generally be extracted (at least by the eye) from the scene being examined, so a bootstrapping procedure is then able to unlock the image data gradually and proceed to an interpretation.

While these methods enable the eye to interpret real scenes, it is difficult to say quite to what degree of precision they are carried out. With computer vision, the required precision levels are liable to be higher, although the machine will be aided by knowing exactly where the source of illumination is. However, with computer vision, we can go further and arrange artificial lighting schemes that would not appear in nature, so the computer can acquire an advantage over the human visual system. In particular, a set of light sources can be applied in sequence to the scene—an approach known as photometric stereo—which can in certain cases help the computer to interpret the scene more rigorously and efficiently. In other cases, structured light may be applied. This means projecting onto the scene a pattern of spots or stripes, or even a grid of lines, and measuring their positions in the resulting image. By this means the depth information can be obtained much as for pairs of stereo images.

Finally, a number of methods have been developed for analyzing images on the basis of readily identifiable sets of features. These methods are the 3-D analogs of the graph matching and GHT approaches of Chapter 14. However, they are significantly more complex because they generally involve six degrees of freedom in place of the three assumed throughout Chapter 14. It should also be

noted that such methods make strong assumptions about the particular objects to be located within the scene. In general situations, it is unlikely that such assumptions could be made, and so initial analysis of any images must be made on the basis that the entire scene must be mapped out in 3-D, then 3-D models built up, and finally deductions must be made by noticing what relation one part of the scene bears to another part. Note that if a scene is composed from an entirely new set of objects, all that can be done is to *describe* what is present and say perhaps what the set most closely *resembles*: recognition *per se* cannot be performed. Note that scene analysis is—at least from a single monocular image—an inherently ambiguous process: every scene can have a number of possible interpretations and there is evidence that the eye looks for the simplest and most probable explanation rather than an absolute interpretation. Indeed, it is underlined by the many illusions to which the eye–brain system is subject, that decisions must repeatedly be made concerning the most likely interpretation of a scene and that there is some risk that its internal model builder will lock on to an interpretation or part-interpretation that is suboptimal (see the paintings of Escher).

This section has indicated that methods of 3-D vision can be categorized according to whether they start by mapping out the shapes of objects in 3-D space and then attempt to interpret the resulting shapes, or whether they try to identify objects directly from their features. In either case, a knowledge base is ultimately called for. It has also been seen that methods of mapping objects in real space include monocular and binocular methods, although structured lighting can help to offset the deficiencies of employing a single “eye.” Laser scanning and ranging techniques must also be included in methods of 3-D mapping, although space precludes detailed discussion of these techniques in this book.

15.3 PROJECTION SCHEMES FOR THREE-DIMENSIONAL VISION

It is common in engineering drawings to provide three views of an object to be manufactured—the plan, the side view, and the elevation. Traditionally these views are simple orthographic (nondistorting) projections of the object—i.e., they are made by taking sets of parallel lines from points on the object to the flat plane on which it is being projected.

However, when objects are viewed by eye or from a camera, rays converge to the lens and so images formed in this way are subject not only to change of scale but also to perspective distortions (Fig. 15.1). This type of projection is called perspective projection, although it includes orthographic projection as the special case of viewing from a distant point. Unfortunately, perspective projections have the disadvantage that they tend to make objects appear more complex than they really are by destroying simple relationships between their features. Thus, parallel edges no longer appear parallel and midpoints no longer appear as such (although

(a) (b)

FIGURE 15.1

(a) Image of a rectangular box taken using orthographic projection; (b) the same box taken using perspective projection. In (b), note that parallel lines no longer appear parallel, although paradoxically the box appears more realistic.

many useful geometric properties still hold—e.g., a tangent line remains a tangent line and the order of points on a straight line remains unchanged).

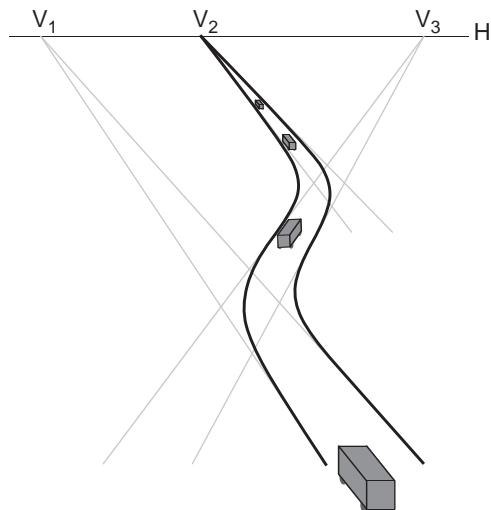
In outdoor scenes, it is very common to see lines that are known to be parallel apparently converging toward a vanishing point on the horizon line (Fig. 15.2). In fact, the horizon line is the projection onto the image plane of the line at infinity on the ground plane G: it is the set of all possible vanishing points for parallel lines on G. In general, the vanishing points of a plane P are the projections onto the image plane corresponding to points at infinity in different directions on P. Thus, any plane Q within the field of view may have vanishing points in the image plane, and these will lie on a vanishing line, which is the analog of the horizon line for Q.

Figure 15.3(a) shows how an image is projected into the image plane by a convex (eye or camera) lens at the origin. It is inconvenient to have to consider inverted images and it is a commonly used convention in image analysis to set the center of the lens at the origin (0, 0, 0) and to imagine the image plane to be the plane $Z=f$, f being the focal length of the lens; with this simplified geometry (Fig. 15.3(b)), images in the image plane appear noninverted. Taking a general point in the scene as (X, Y, Z) , which appears in the image as (x_1, y_1) , perspective projection now gives:

$$(x_1, y_1) = \left(\frac{fX}{Z}, \frac{fY}{Z} \right) \quad (15.1)$$

15.3.1 Binocular Images

Figure 15.4 shows the situation when two lenses are used to obtain a stereo pair of images. In general, the two optical systems do not have parallel optical axes but exhibit a “vergence” (which may be variable, as it is for human eyes), so that they intersect at some point within the scene. Then a general point (X, Y, Z) in the scene has two different pairs of coordinates, (x_1, y_1) and (x_2, y_2) , in its two images, which differ both because of the vergence between the optical axes and

**FIGURE 15.2**

Vanishing points and the horizon line. This figure shows how parallel lines on the ground plane appear, under perspective projection, to meet at vanishing points V_i on the horizon line H . (Note that V_i and H lie in the *image plane*.) If two parallel lines do not lie on the ground plane, their vanishing point will lie on a different vanishing line. Hence, it should be possible to determine whether any roads are on an incline by computing all the vanishing points for the scene.



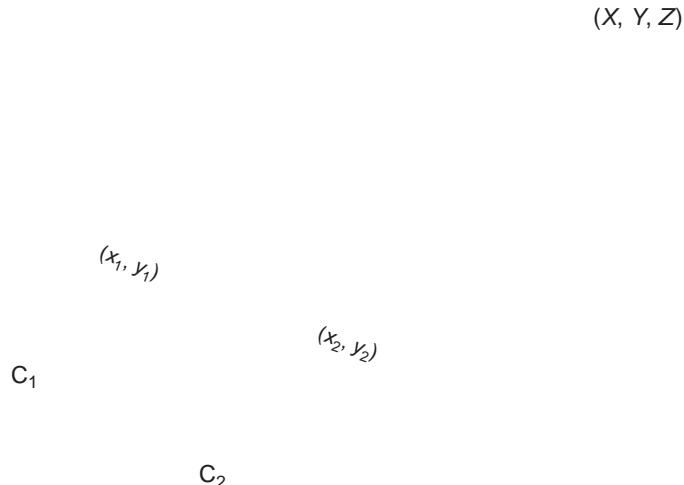
(a)



(b)

FIGURE 15.3

(a) Projection of an image into the image plane by a convex lens; note that a single image plane only brings objects at a single distance into focus but that for far-off objects the image plane may be taken to be the focal plane, a distance F from the lens;
 (b) a commonly used convention that imagines the projected image to appear noninverted at a focal plane F in front of the lens. The center of the lens is said to be the center of projection for image formation.

**FIGURE 15.4**

Stereo imaging using two lenses. The axes of the optical systems are parallel, i.e., there is no “vergence” between the optical axes.

because the baseline b between the lenses causes relative displacement or “disparity” of the points in the two images.

For simplicity, we now take the vergence to be zero, i.e., the optic axes are parallel. Then, with suitable choice of Z -axis on the perpendicular bisector of the baseline b , we obtain two equations:

$$x_1 = \frac{(X + b/2)f}{Z} \quad (15.2)$$

$$x_2 = \frac{(X - b/2)f}{Z} \quad (15.3)$$

so that the disparity is

$$D = x_1 - x_2 = \frac{bf}{Z} \quad (15.4)$$

Rewriting this equation in the form:

$$Z = \frac{bf}{x_1 - x_2} \quad (15.5)$$

now permits the depth Z to be calculated. In fact, computation of Z only requires the disparity for a stereo pair of image points to be found and parameters of the optical systems to be known. However, confirming that both points in a stereo pair actually correspond to the same point in the original scene is in general not at all trivial, and much of the computation in stereo vision is devoted to this task. In addition, to obtain good accuracy in the determination of depth, a large baseline b is required. Unfortunately as b is increased, the correspondence between the images decreases, so it becomes more difficult to find matching points.

15.3.2 The Correspondence Problem

There are two important approaches to finding pairs of points that match in the two images of a stereo pair. One is that of “light striping” (one form of structured lighting), which encodes the two images so that it is easy to see pairs of corresponding points. If a single vertical stripe is used, for every value of y there is in principle only one light stripe point in each image and so the matching problem is solved. We return to this problem in a later section.

The second important approach is to employ epipolar lines. To understand this approach, imagine that we have located a distinctive point in the first image and that we are marking all possible points in the object field which could have given rise to it. This will mark out a line of points at various depths in the scene and, when viewed in the second image plane, a locus of points can be constructed in that plane. This locus is the *epipolar line* corresponding to the original image point in the alternate image (Fig. 15.5). If we now search along the epipolar line for a similarly distinctive point in the second image, the chance of finding the correct match is significantly enhanced. This method has the advantage not only of cutting down the amount of computation required to find corresponding points, but also of reducing significantly the chance of false alarms. Note that the concept of an epipolar line applies to both images—a point in one image gives an epipolar line in the other image. Note also that in the simple geometry of Fig. 15.4, all epipolar lines are parallel to the x -axis, although this is not so in general (in fact, the general situation is that all epipolar lines in one image plane pass through the point that is the image of the projection point of the alternate image plane).

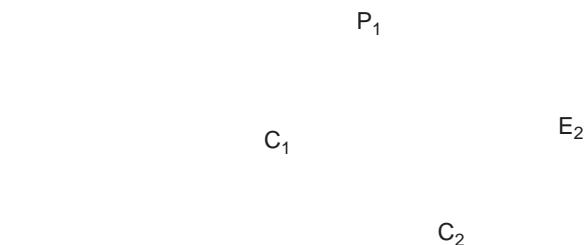


FIGURE 15.5

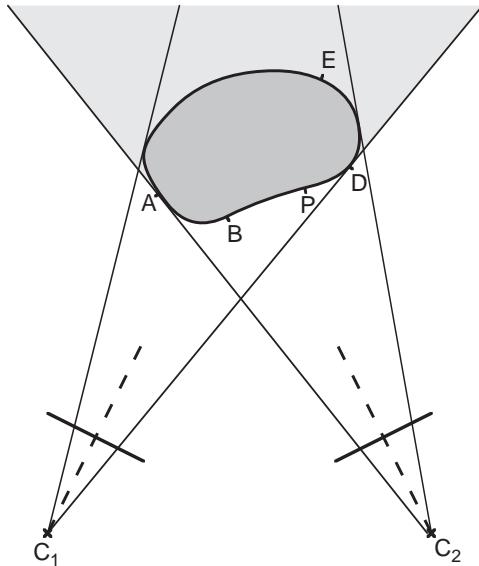
Geometry of epipolar lines. A point P_1 in one image plane may have arisen from any one of a line of points in the scene, and may appear in the alternate image plane at any point on the so-called epipolar line E_2 .

The correspondence problem is rendered considerably more difficult by the fact that there will be points in the scene that give rise to points in one image but not in the other. Such points are either occluded in the one image, or else are so distorted as not to give a recognizable match in the two images (e.g., the different background might mask a corner point in one image while permitting it to stand out in the other). Any attempt to match such points can then only lead to false alarms. Thus, it is necessary to search for consistent sets of solutions in the form of continuous object surfaces in the scene. For this reason, iterative “relaxation” schemes are widely used to implement stereo matching.

Broadly speaking, correspondences are sought by two methods: one is the matching of near-vertical edge points in the two images (near-horizontal edge points do not give the required precision); the other is the matching of local intensity patterns using correlation techniques. Correlation is an expensive operation and in this case is relatively unreliable—principally because intensity patterns frequently appear significantly foreshortened¹ in one or other image and hence are difficult to match reliably. In such cases, the most practical solution is to reduce the baseline; as noted earlier, this has the effect of reducing the accuracy of depth measurement. Further details of these techniques are to be found in Shirai (1987).

Before leaving this topic, we consider in slightly more detail how the problems of visibility mentioned above arise. Figure 15.6 shows a situation in which an object is being observed by two cameras giving stereo images. Clearly, much of the object will not be visible in either image because of self-occlusion, while some feature points will only be visible in one or the other image. Now, consider the order in which the points appear in the two images (Fig. 15.7). The points that are visible appear in the same order as in the scene, and the points that are just going out of sight are those for which the order between the scene and the image is just about to change. Points that provide information about the front surface of the object can thus only bear a simple geometrical relation to each other: in particular, for points not to obscure, or be obscured by, a given point P, they must not lie within a double angular sector defined by P and the centers of projection C_1, C_2 of the two cameras. This region is shown shaded in Fig. 15.7. A surface passing through P for which full depth information can be retrieved must lie entirely within the nonshaded region. (Of course, a new double sector must be considered for each point on the surface being viewed.) Note that the possibility of objects containing holes, or having transparent sections, must not be forgotten (such cases can be detected from differences in the ordering of feature points in the two views—see Fig. 15.7); neither must it be ignored that the foregoing figures represent a single horizontal cross-section of an object that can have totally different shapes and depths in different cross-sections.

¹That is, distorted by the effects of perspective.

**FIGURE 15.6**

Visibility of feature points in two stereo views. Here an object is viewed from two directions. Only feature points that appear in both views are of value for depth estimation. This eliminates all points in the shaded region, such as E, from consideration.

15.4 SHAPE FROM SHADING

It was mentioned in [Section 15.2](#) that it is possible to analyze the pattern of intensities in a single (monocular) image and to deduce the shapes of objects from the shading information. The principle underlying this technique is that of modeling the reflectance of objects in the scene as a function of the angles of incidence i and emergence e of light from their surfaces. In fact, a third angle is also involved, and it is called the “phase” g ([Fig. 15.8](#)).

A general model of the situation gives the radiance I (light intensity in the image) in terms of the irradiance E (energy per unit area falling on the surface of the object) and the reflectance R :

$$I(x_1, y_1) = E(x, y, z)R(\mathbf{n}, \mathbf{s}, \mathbf{v}) \quad (15.6)$$

It is well known that a number of matt surfaces approximate reasonably well to an ideal Lambertian surface whose reflectance function depends only on the angle of incidence i —i.e., the angles of emergence and phase are immaterial:

$$I = \left(\frac{1}{\pi}\right)E \cos i \quad (15.7)$$

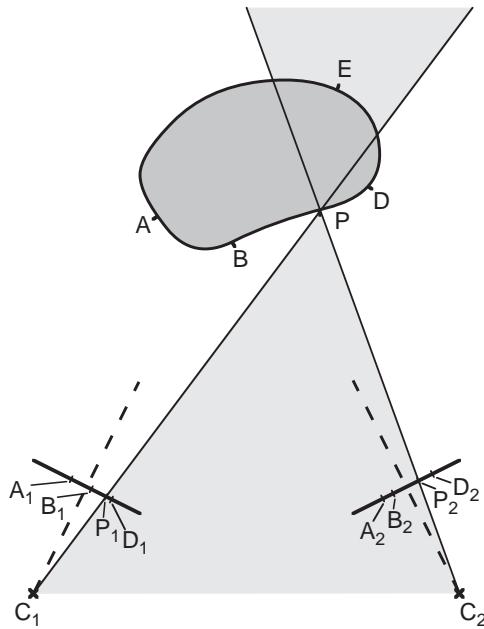


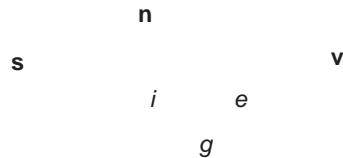
FIGURE 15.7

Ordering of feature points on an object. In the two views of the object shown here, the feature points all appear in the same order A, B, P, D as on the surface of the object. Points for which this would not be valid, such as E, are behind the object and are obscured from view. Relative to a given visible feature P, there is a double sector (shaded) in which feature points must not appear if they are not to obscure the feature under consideration. An exception to these rules might be if the object had a semi-transparent window through which an additional feature T were visible: in that case interpretation would be facilitated by noting that the orderings of the features seen in the two views were different—e.g., A₁, T₁, B₁, P₁, D₁ and A₂, B₂, T₂, P₂, D₂.

For the present purpose, *E* is regarded as a constant and is combined with other constants for the camera and the optical system (including, e.g., the *f*-number). In this way, a normalized reflectance is obtained, which in this case is:

$$\begin{aligned}
 R &= R_0 \cos i = R_0 \mathbf{s} \cdot \mathbf{n} \\
 &= \frac{R_0(1 + pp_s + qq_s)}{(1 + p^2 + q^2)^{1/2}(1 + p_s^2 + q_s^2)^{1/2}}
 \end{aligned} \tag{15.8}$$

where we have used the standard convention of writing orientations in 3-D in terms of *p* and *q* values. These are not direction cosines but correspond to the coordinates of the point (*p*, *q*, 1) at which a particular direction vector from the origin meets the plane *z* = 1: hence they need suitable normalization, as in Eq. (15.8).

**FIGURE 15.8**

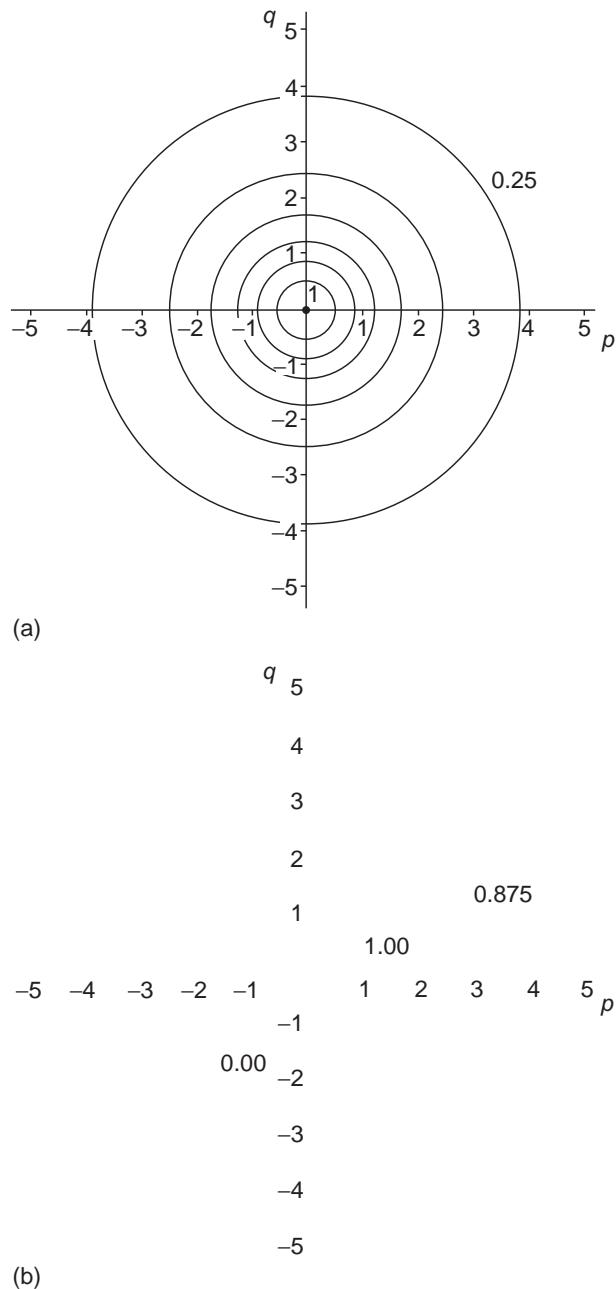
Geometry of reflection. An incident ray from source direction \mathbf{s} is reflected along the viewer direction \mathbf{v} by an element of the surface whose local normal direction is \mathbf{n} ; i , e , and g are defined respectively as the incident, emergent, and phase angles.

[Equation \(15.8\)](#) gives a reflectance map in gradient (p, q) space. We now temporarily set the absolute reflectance value R_0 equal to unity. The reflectance map can be drawn as a set of contours of equal brightness, starting with a point having $R = 1$ at $\mathbf{s} = \mathbf{n}$, and going down to zero for \mathbf{n} perpendicular to \mathbf{s} . When $\mathbf{s} = \mathbf{v}$, so that the light source is along the viewing direction (here taken to be the direction $p = q = 0$), zero brightness occurs only for infinite distances on the reflectance map ($(p^2 + q^2)^{1/2}$ approaching infinity) ([Fig. 15.9\(a\)](#)). In a more general case, when $\mathbf{s} \neq \mathbf{v}$, zero brightness occurs along a straight line in gradient space ([Fig. 15.9\(b\)](#)). To find the exact shapes of the contours, we can set R at a constant value a , which results in:

$$a(1 + p^2 + q^2)^{1/2}(1 + p_s^2 + q_s^2)^{1/2} = 1 + pp_s + qq_s \quad (15.9)$$

Squaring this equation clearly gives a quadratic in p and q , which could be simplified by a suitable change of axes. Thus, the contours must be curves of conic section, namely, circles, ellipses, parabolas, hyperbolas, lines, or points (the case of a point arises only when $a = 1$, when we get $p = p_s$, $q = q_s$; and that of a line only if $a = 0$, when we get the equation $1 + pp_s + qq_s = 0$: both of these solutions were implied above).

Unfortunately, object reflectances are not all Lambertian, and an obvious exception is for surfaces that approximate to pure specular reflection. In that case, $e = i$ and $g = i + e$ (\mathbf{s} , \mathbf{n} , \mathbf{v} are coplanar); the only nonzero reflectance position in gradient space is the point representing the bisector of the angle between the

**FIGURE 15.9**

Reflectance maps for Lambertian surfaces: (a) contours of constant intensity plotted in gradient (p, q) space for the case where the source direction \mathbf{s} (marked by a black dot) is along the viewing direction $\mathbf{v} (0, 0)$ (the contours are taken in steps of 0.125 between the values shown); (b) the contours that arise where the source direction (p_s, q_s) is at a point (marked by a black dot) in the positive quadrant of (p, q) space: note that there is a well-defined region, bounded by the straight line $1 + pp_s + qq_s = 0$, for which the intensity is zero (the contours are again taken in steps of 0.125).

source direction \mathbf{s} (p, q) and the viewing direction \mathbf{v} (0, 0)—i.e., \mathbf{n} is along $\mathbf{s} + \mathbf{v}$ —and very approximately:

$$p \approx \frac{p_s}{2} \quad (15.10)$$

$$q \approx \frac{q_s}{2} \quad (15.11)$$

For less perfect specularity, a peak is obtained around this position. A good approximation to the reflectance of many real surfaces is obtained by modeling them as basically Lambertian but with a strong additional reflectance near the specular reflectance position. Using the Phong (1975) model for the latter component gives:

$$R = R_0 \cos i + R_1 \cos^m \theta \quad (15.12)$$

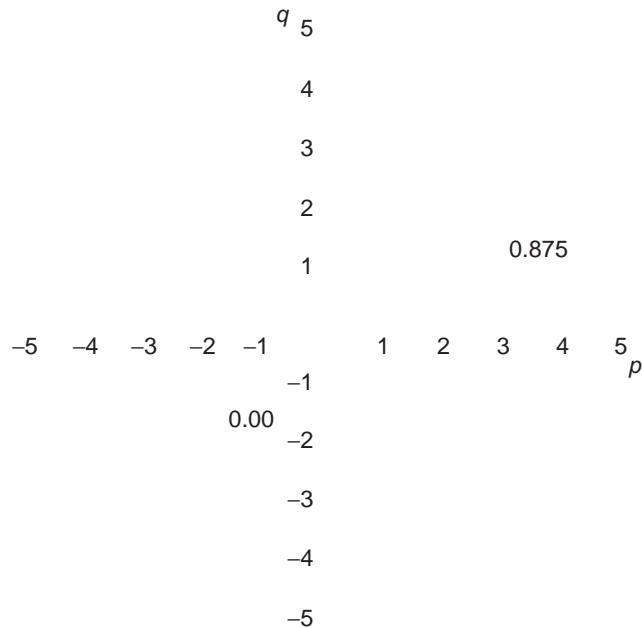
θ being the angle between the actual emergence direction and the ideal specular reflectance direction.

The resulting contours now have two centers around which to peak: the first is the ideal specular reflection direction ($p \approx p_s/2, q \approx q_s/2$) and the second is that of the source direction ($p = p_s, q = q_s$). When objects are at all shiny—such as metal, plastic, liquid, or even wood surfaces—the specular peak is quite sharp and rather intense: casual observation may not even indicate the presence of another peak, since Lambertian reflection is so diffuse (Fig. 15.10). In other cases, the specular peak can broaden and become more diffuse: hence it may merge with the Lambertian peak and effectively disappear.

Some remarks should be made about the Phong model employed above. First, it is adapted to different materials by adjusting the values of R_0 , R_1 , and m . Phong remarks that R_1 typically lies between 10% and 80%, while m is in the range 1–10. However, Rogers (1985) indicates that m may be as high as 50. Note that there is no physical significance in these numbers—the model is simply a phenomenological one. This being so, care should be taken to prevent the $\cos^m \theta$ term from contributing to reflectance estimates when $|\theta| > 90^\circ$. The Phong model is reasonably accurate but has been improved by Cook and Torrance (1982). This is important in computer graphics applications, but the improvement is difficult to apply in computer vision, because of lack of data concerning the reflectances of real objects and because of variability in the current state (cleanliness, degree of polish, etc.) of a given surface. However, the method of photometric stereo gives some possibility of overcoming these problems.

15.5 PHOTOMETRIC STEREO

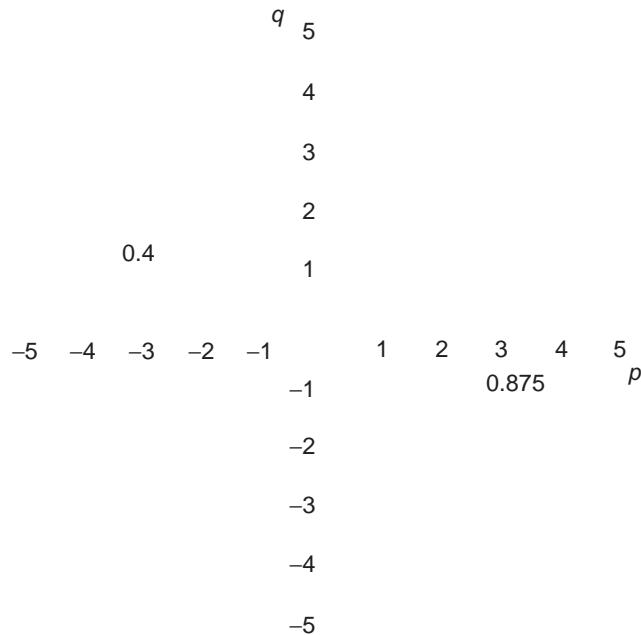
Photometric stereo is a form of structured lighting that increases the information available from surface reflectance variations. Basically, instead of taking a single monocular image of a scene illuminated from a single source, several images are

**FIGURE 15.10**

Reflectance map for a non-Lambertian surface: a modified form of Fig. 15.9(b) for the case where the surface has a marked specular component ($R_0 = 1.0$, $R_1 = 0.8$). Note that the specular peak can have very high intensity (much greater than the maximum value of unity for the Lambertian component). In this case, the specular component is modeled with a $\cos^8\theta$ variation (the contours are again taken in steps of 0.125).

taken, from the same vantage point, with the scene illuminated in turn by separate light sources. These light sources are ideally point sources some distance away in various directions, so that there is in each case a well-defined light source direction from which to measure surface orientation.

The basic idea of photometric stereo is that of cutting down the number of possible positions in gradient space for a given point on the surface of an object. It has already been seen that, for known absolute reflectance R_0 , a constant brightness in one image permits the surface orientation to be limited to a curve of conic cross-section in gradient space. This would also be true for a second such image, the curve being a new one if the illuminating source is different. In general, two such conic curves meet at two points, so there is now only a single ambiguity in the gradient of the surface at any given point in the image. To resolve this ambiguity a third source of illumination can be employed (this must not be in the plane containing the first two and the surface point being examined), and the third image gives another curve in gradient space that should pass through the appropriate crossing point of the first two curves (Fig. 15.11). If a third source of

**FIGURE 15.11**

Obtaining a unique surface orientation by photometric stereo. Three contours of constant intensity arise for different light sources of equal strength: all three contours pass through a single point in (p, q) space and result in a unique solution for the local gradient.

illumination cannot be used, it is sometimes possible to arrange that the inclination of each of the sources is so high that $(p^2 + q^2)^{1/2}$ on the surface is always lower than $(p_s^2 + q_s^2)^{1/2}$ for each of the sources, so that only one interpretation of the data is possible. This method is prone to difficulty, however, since it means that parts of the surface could be in shadow, thereby preventing the gradient for these parts of the surface from being measured. Another possibility is to assume that the surface is reasonably smooth, so that p and q vary continuously over it. This itself ensures that ambiguities are resolved over most of the surface.

However, there are other advantages to be gained from using more than two sources of illumination. One is that information on the absolute surface reflectance can be obtained. Another is that the assumption of a Lambertian surface can be tested. Thus, three sources of illumination ensure that the remaining ambiguity is resolved *and* permit absolute reflectivity to be measured: this is obvious, since if the three contours in gradient space do not pass through the same point, then the absolute reflectivity cannot be unity, so corresponding contours should be sought that do pass through the same point. In practice, the calculation is normally carried out by defining a set of nine matrix components of irradiance, s_{ij} being the j th component of light source vector \mathbf{s}_i . Then, in matrix notation:

$$\mathbf{E} = R_0 S \mathbf{n} \quad (15.13)$$

where

$$\mathbf{E} = (E_1, E_2, E_3)^T \quad (15.14)$$

and

$$S = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{32} & s_{33} \end{bmatrix} \quad (15.15)$$

Provided that the three vectors $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ are not coplanar, so that S is not a singular matrix, R_0 and \mathbf{n} can now be determined from the formulas:

$$R_0 = |S^{-1}\mathbf{E}| \quad (15.16)$$

$$\mathbf{n} = \frac{S^{-1}\mathbf{E}}{R_0} \quad (15.17)$$

An interesting special case arises if the three source directions are mutually perpendicular; taking them to be aligned along the respective major axes directions, S is now the unit matrix, so that:

$$R_0 = (E_1^2 + E_2^2 + E_3^2)^{1/2} \quad (15.18)$$

and

$$\mathbf{n} = \frac{(E_1, E_2, E_3)^T}{R_0} \quad (15.19)$$

If four or more images are obtained using further illumination sources, more information can be obtained: e.g., the coefficient of specular reflectance, R_1 . In practice, this coefficient varies somewhat randomly with the cleanliness of the surface and it may not be relevant to determine it accurately. More probably, it will be sufficient to check whether significant specularity is present, so that the corresponding region of the surface can be ignored for absolute reflectance calculations. Nonetheless, finding the specularity peak can itself give important surface orientation information, as will be clear from Section 15.4. Note that, although the information from several illumination sources should ideally be collated using least-squares analysis, this method requires significant computation. Hence, it seems better to use the images resulting from further illumination sources as confirmatory—or, instead, to select the three that exhibit the least evidence of specularity as giving the most reliable information on local surface orientation.

15.6 THE ASSUMPTION OF SURFACE SMOOTHNESS

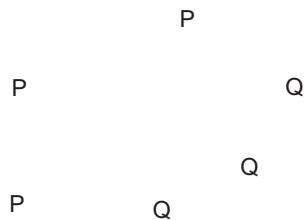
It was hinted above that the assumption of a reasonably smooth surface permits ambiguities to be removed in situations where there are two illuminating sources.

In fact, this method can be used to help analyze the brightness map even for situations where a single source is employed: indeed, the fact that the eye can perform this feat of interpretation indicates that it should be possible to find computer methods for achieving it. Much research has been carried out on this topic and a set of methods is available, although the calculations are complex, iterative, computation intensive procedures. For this reason, they are not studied in depth here: the reader is referred to the volume by Horn (1986) for detailed information on this topic. However, one or two remarks are in order.

First, consider the representation to be employed for this type of analysis. In fact, normal gradient (p, q) space is not very appropriate for the purpose. In particular, it is necessary to average gradient (i.e., the \mathbf{n} -values) locally within the image; however, (p, q) space is not “linear”, in that a simple average of (p, q) values within a window would give biased results. It turns out that a conformal representation of gradient (i.e., one which preserves small shapes) is closer to the ideal, in that the distances between points in such a representation provide better approximations to the relative orientations of surface normals: averaging in such a representation gives reasonably accurate results. The required representation is obtained by a stereographic projection, which maps the unit (Gaussian) sphere onto a plane ($z = 1$) through its north pole but this time using as a projection point not its center but its south pole. This projection has the additional advantage that it projects all possible orientations of a surface onto the plane, not merely those from the northern hemisphere. Hence, backlit objects can be represented conveniently in the same map as used for frontlit objects.

Second, the relaxation methods used to estimate surface orientation have to be provided with accurate boundary conditions: in principle, the more correct the orientations that are presented initially to such procedures, the more quickly and accurately the iterations proceed. There are normally two sets of boundary conditions that can be applied in such programs. One is the set of positions in the image where the surface normal is perpendicular to the viewing direction. The other is the set of positions in the image where the surface normal is perpendicular to the direction of illumination: this set of positions corresponds to the set of shadow edges (Fig. 15.12). Careful analysis of the image must be undertaken to find each set of positions, but once they have been located they provide valuable cues for unlocking the information content of the monocular image, and mapping out surfaces in detail.

Finally, all shapes from shading techniques provide information that initially takes the form of surface orientation maps. Dimensions are not obtainable directly but these can be computed by integration across the image from known starting points. In practice this tends to mean that absolute dimensions are unknown and that dimensional maps are obtainable only if the size of an object is given or if its depth within the scene is known.

**FIGURE 15.12**

The two types of boundary condition that can be used in shape-from-shading computations of surface orientation: (i) positions P where the surface normal is perpendicular to the viewing direction; (ii) positions Q where the surface normal is perpendicular to the direction of illumination (i.e., shadow boundaries).

15.7 SHAPE FROM TEXTURE

Texture can be very helpful to the human eye in permitting depth to be perceived. Although textured patterns can be very complex, even the simplest textural elements can carry depth information. Ohta et al. (1981) showed how circular patches on a flat surface viewed more and more obliquely in the distance become first elliptical and then progressively flatter and flatter. At infinite distance, on the horizon line (here defined as the line at infinity in the given plane), they would clearly become very short line segments. To disentangle such textured images sufficiently to deduce depths within the scene, it is first necessary to find the horizon line reliably. This is achieved by taking all pairs of texture elements and deducing from their areas where the horizon line would have to be. To proceed, we make use of the rule:

$$\frac{d_1^3}{d_2^3} = \frac{A_1}{A_2} \quad (15.20)$$

which applies since circles at various depths would give a square law, although the progressive eccentricity also reduces the area linearly in proportion to the depth. This information is accumulated in a separate image space and a line is then fitted to these data: false alarms are eliminated automatically by this Hough-based procedure.

At this stage the original data—the ellipse areas—provide direct information on depth, although some averaging is required to obtain accurate results. Although this type of method has been demonstrated in certain instances, it is in practice highly restricted unless very considerable amounts of computation are performed. Hence it is doubtful whether it can be of general practical use in machine vision applications.

15.8 USE OF STRUCTURED LIGHTING

Structured lighting has already been considered briefly in [Section 15.2](#) as an alternative to stereo for mapping out depth in scenes. Basically, a pattern of light stripes, or other arrangement of light spots or grids, is projected onto the object field. Then these patterns are enhanced in a (generally) single monocular image and analyzed to extract the depth information. To obtain the maximum information the light pattern must be close-knit and the received images must be of very high resolution. When shapes are at all complex, the lines can in places appear so close together that they are unresolvable. It then becomes necessary to separate the elements in the projected pattern, trading resolution and accuracy for reliability of interpretation. Even so, if parts of the objects are along the line of sight, the lines can merge and even cross back and fore, so unambiguous interpretation is never assured. In fact, this is part of a larger problem, in which parts of the object will be obscured from the projected pattern by occluding bodies or by self-occlusion. The method has this feature in common with the shape from shading technique and with stereo vision, which relies on *both* cameras being able to view various parts of the objects simultaneously. Hence, the structured light approach is subject to similar restrictions to those found for other methods of 3-D vision and is not a panacea. Nevertheless, it is a useful technique that is generally simple to set up so as to acquire specific 3-D information that can enable a computer to start the process of cueing into complex images.

Light spots provide perhaps the most obvious form of structured light. However, they are restricted because for each spot, an analysis has to be performed to determine which spot is being viewed: connected lines, in contrast, carry a large amount of coding information with them so that ambiguities are less likely to arise. Grids of lines carry even more coding information but do not necessarily give any more depth information. Indeed, if a pattern of light stripes can be projected, e.g., from the left of the camera so that they are parallel to the y -axis in the observed image, then there is no point in projecting another set of lines parallel to the x -axis, since these merely replicate information that is already available from the rows of pixels in the image—all the depth information is carried by the vertical lines and their horizontal displacements in the image. This analysis assumes that the camera and projected beams are carefully aligned and that no perspective or other distortions are present. In fact, most practical structured

**FIGURE 15.13**

Three of the structures that are observed when a light stripe is incident on even quite simple shapes: bends (B), jumps (J), and discontinuities (D).

lighting systems in current use employ light stripe patterns rather than spot patterns or full grid patterns.

This section ends with an analysis of the situations that can arise when a single stripe is incident on objects as simple as rectangular blocks. [Figure 15.13](#) shows three types of structure in observed stripes: (a) the effect of a sharp angle being encountered; (b) the effect of “jump edges” at which light stripes jump horizontally and vertically at the same time; and (c) the effect of discontinuous edges at which light stripes jump horizontally but not vertically. The reasons for these circumstances will be obvious from [Fig. 15.13](#). Basically, the problem to be tackled with jump and discontinuous edges is to find whether a given stripe end marks an occluding edge or an occluded edge. The importance of this distinction is that occluding edges mark actual edges of the object being observed, whereas occluded edges may be merely edges of shadow regions and are then not *directly* significant.² A simple rule is that, if stripes are projected from the left, the left-hand component of a discontinuous edge will be the occluding edge and the right-hand component will be the occluded edge. Angle edges are located by

²More precisely, they involve interactions of light with two objects rather than with one, and are therefore more complex to interpret.

applying a Laplacian type of operator that detects the change in orientation of the light stripe.

The ideas outlined above correspond to possible 1-D operators that interpret light stripe information to locate nonvertical edges of objects. The method provides no direct information concerning vertical edges. To obtain such information it is necessary to analyze the information from sets of light stripes. For this purpose 2-D edge operators are required, which collect sufficient data from at least two or three adjacent light stripes. Further details are beyond the scope of this chapter.

Overall, light stripes provide a very useful means of recognizing planes forming the faces of polyhedra and other types of manufactured object. The characteristic sets of parallel lines can be found and demarcated relatively easily, and the fact that the lines usually give rather strong signals means that line tracking techniques can be applied and that algorithms can operate quite rapidly. However, whole-scene interpretation, including inferring the presence and relative positions of different objects, remains a more complex task, as will be seen below.

15.9 THREE-DIMENSIONAL OBJECT RECOGNITION SCHEMES

The methods described so far in this chapter employ various means for finding depth at all places in a scene, and are hence able to map out 3-D surfaces in a fair amount of detail. However, they do not give any clue as to what these surfaces represent. In some situations it may be clear that certain planar surfaces are parts of the background, e.g., the floor and the walls of a room, but in general individual objects will not be inherently identifiable. Indeed, objects tend to merge with each other and with the background, so specific methods are needed to segment the 3-D space map³ and finally recognize the objects, giving detailed information on their positions and orientations.

Before proceeding to study this problem, notice that further general processing can be carried out to analyze the 3-D shapes. Agin and Binford (1976) and others have developed techniques for likening 3-D shapes to “generalized cylinders,” these being like normal (right circular) cylinders but with additional degrees of freedom so that the axes can bend and the cross-sections can vary, both in size and in detailed shape: even an animal like a sheep can be likened to a distorted cylinder. On the whole, this approach is elegant but may not be well adapted to describe many industrial objects, and it is therefore not pursued further here. A simpler approach may be to model the 3-D surfaces as planar, quadratic, cubic, and quartic, and then to try to understand these model surfaces in terms of what

³This may be defined as an imagined 3-D map showing, without interpretation, the surfaces of all objects in the scene and incorporating all the information from depth or range images. Note that it will generally include only the front surfaces of objects seen from the vantage point of the camera.

is known about existing objects. This approach was adopted by Hall et al. (1982) and was found to be viable, at least for certain quite simple objects such as cups. Shirai (1987) has taken the approach even further so that a whole range of objects can be found and identified in quite complex indoor scenes.

We next consider what we are trying to achieve regarding recognition. First, can recognition be carried out *directly* on the mapped out 3-D surfaces, just as it could for the 2-D images of earlier chapters? Second, if we can bypass the 3-D modeling process, and still recognize objects, might it not be possible to save even more computation and omit the stage of mapping out 3-D surfaces, instead identifying 3-D objects directly in 2-D images? It might even be possible to locate 3-D objects from a single 2-D image.

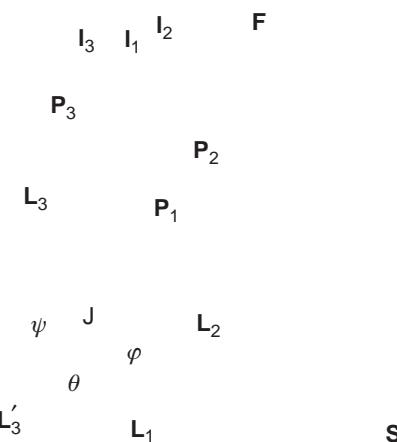
Consider the first of these problems. When we studied 2-D recognition, many instances were found where the HT approach was of great help. It turned out to give trouble in more complex cases, particularly when attempts were made to find objects where there were more than two or at most three degrees of freedom. Here, however, we have situations where objects normally have six degrees of freedom—three degrees of freedom for translation and another three for rotation. This doubling of the number of free parameters on going from 2-D to 3-D makes the situation far worse, since the search space is proportional in size not to the number of degrees of freedom, but to its exponent: e.g., if each degree of freedom in translation or rotation can have 256 values, the number of possible locations in parameter space changes from 256^3 in 2-D to 256^6 in 3-D. This will be seen to have a very profound effect on object location schemes and tends to make the HT technique difficult to implement. In Section 15.10, we study an interesting approach to the 3-D recognition problem, which uses a subtle combination of 2-D and 3-D techniques.

15.10 HORAUD'S JUNCTION ORIENTATION TECHNIQUE⁴

Horauad's (1987) technique is special in that it uses as its starting point 2-D images of 3-D scenes and “backprojects” them into the scene, with the aim of making interpretations in 3-D rather than 2-D frames of reference. This has the initial effect of increasing mathematical complexity, although in the end useful, more accurate results emerge.

Initially the boundaries of planar surfaces on objects are backprojected. Each boundary line is thus transformed into an “interpretation plane” defined by the center of the camera projection system and the boundary line in the image plane: clearly, the interpretation plane must contain the line that originally projected into the boundary line in the image. Similarly, angles between boundary lines in the image are backprojected into two interpretation planes, which must contain the original two object lines. Finally, junctions between three boundary lines are

⁴This and related techniques are sometimes referred to as “shape from angle.”

**FIGURE 15.14**

Geometry for backprojection from junctions: a junction of three lines in an image may be backprojected into three planes, from which the orientation in space of the original corner **J** may be deduced.

backprojected into three interpretation planes which must contain a corner in the space map (Fig. 15.14). The paper focusses on the backprojection of junctions and shows how measurements of the junction angles in the image relate to those of the original corner; it also shows how the space orientation of the corner can be computed. In fact, it is interesting that the orientation of an object in 3-D can in general be deduced from the appearance of just one of its corners in a single image. This is a powerful result and in principle permits objects to be recognized and located from extremely sparse data.

To understand the method, the mathematics first needs to be set up with some care. Assume that lines **L₁**, **L₂**, **L₃** meet at a junction in an object, and appear as lines **I₁**, **I₂**, **I₃** in the image (Fig. 15.14). Take respective interpretation planes containing the three lines and label them by unit vectors **P₁**, **P₂**, **P₃** along their normals, so that:

$$\mathbf{P}_1 \cdot \mathbf{L}_1 = 0 \quad (15.21)$$

$$\mathbf{P}_2 \cdot \mathbf{L}_2 = 0 \quad (15.22)$$

$$\mathbf{P}_3 \cdot \mathbf{L}_3 = 0 \quad (15.23)$$

In addition, take the space plane containing \mathbf{L}_1 and \mathbf{L}_2 , and label it by a unit vector \mathbf{S} along its normal, so that:

$$\mathbf{S}_1 \cdot \mathbf{L}_1 = 0 \quad (15.24)$$

$$\mathbf{S}_2 \cdot \mathbf{L}_2 = 0 \quad (15.25)$$

Since \mathbf{L}_1 is perpendicular to \mathbf{S} and \mathbf{P}_1 , and \mathbf{L}_2 is perpendicular to \mathbf{S} and \mathbf{P}_2 , it is found that:

$$\mathbf{L}_1 = \mathbf{S} \times \mathbf{P}_1 \quad (15.26)$$

$$\mathbf{L}_2 = \mathbf{S} \times \mathbf{P}_2 \quad (15.27)$$

Note that \mathbf{S} is not in general perpendicular to \mathbf{P}_1 and \mathbf{P}_2 , so \mathbf{L}_1 and \mathbf{L}_2 are not in general unit vectors. Defining φ as the angle between \mathbf{L}_1 and \mathbf{L}_2 , we now have:

$$\mathbf{L}_1 \cdot \mathbf{L}_2 = L_1 L_2 \cos \varphi \quad (15.28)$$

which can be re-expressed in the form:

$$(\mathbf{S} \times \mathbf{P}_1) \cdot (\mathbf{S} \times \mathbf{P}_2) = |\mathbf{S} \times \mathbf{P}_1| |\mathbf{S} \times \mathbf{P}_2| \cos \varphi \quad (15.29)$$

Next we need to consider the junction between \mathbf{L}_1 , \mathbf{L}_2 , \mathbf{L}_3 . To proceed, it is necessary to specify the relative orientations in space of the three lines. θ is the angle between \mathbf{L}_1 and the projection \mathbf{L}'_3 of \mathbf{L}_3 on plane \mathbf{S} , while ψ is the angle between \mathbf{L}'_3 and \mathbf{L}_3 (Fig. 15.14). Thus, the structure of the junction J is described completely by the three angles φ , θ , ψ . \mathbf{L}_3 can now be found in terms of other quantities:

$$\mathbf{L}_3 = \mathbf{S} \sin \psi + \mathbf{L}_1 \cos \theta \cos \psi + (\mathbf{S} \times \mathbf{L}_1) \sin \theta \cos \psi \quad (15.30)$$

Applying Eq. (15.23), we find:

$$\mathbf{S} \cdot \mathbf{P}_3 \sin \psi + \mathbf{L}_1 \cdot \mathbf{P}_3 \cos \theta \cos \psi + (\mathbf{S} \times \mathbf{L}_1) \cdot \mathbf{P}_3 \sin \theta \cos \psi = 0 \quad (15.31)$$

Substituting for \mathbf{L}_1 from Eq. (15.26), and simplifying, we finally obtain:

$$\begin{aligned} & (\mathbf{S} \cdot \mathbf{P}_3) |\mathbf{S} \times \mathbf{P}_1| \sin \psi + \mathbf{S} \cdot (\mathbf{P}_1 \times \mathbf{P}_3) \cos \theta \cos \psi \\ & + (\mathbf{S} \cdot \mathbf{P}_1) (\mathbf{S} \cdot \mathbf{P}_3) \sin \theta \cos \psi = (\mathbf{P}_1 \cdot \mathbf{P}_3) \sin \theta \cos \psi \end{aligned} \quad (15.32)$$

Equations (15.31) and (15.32) now exclude the unknown vectors \mathbf{L}_1 , \mathbf{L}_2 , \mathbf{L}_3 but they retain \mathbf{S} , \mathbf{P}_1 , \mathbf{P}_2 , \mathbf{P}_3 and the three angles φ , θ , ψ . \mathbf{P}_1 , \mathbf{P}_2 , \mathbf{P}_3 are known from the image geometry, and the angles φ , θ , ψ are presumed to be known from the object geometry; in addition, only two components (α , β) of the unit vector \mathbf{S} are independent, so the two equations should be sufficient to determine the orientation of the space plane \mathbf{S} . Unfortunately, the two equations are highly nonlinear and it is necessary to solve them numerically. Horaud (1987) achieved this by re-expressing the formulas in the forms:

$$\cos \varphi = f(\alpha, \beta) \quad (15.33)$$

$$\begin{aligned} \sin \theta \cos \psi &= g_1(\alpha, \beta) \sin \psi + g_2(\alpha, \beta) \cos \theta \cos \psi \\ &\quad + g_3(\alpha, \beta) \sin \theta \cos \psi \end{aligned} \quad (15.34)$$

For each image junction, \mathbf{P}_1 , \mathbf{P}_2 , \mathbf{P}_3 are known and it is possible to evaluate f , g_1 , g_2 , g_3 . Then, assuming a particular interpretation of the junction, values are assigned to φ , θ , ψ and curves giving the relation between α and β are plotted for each equation. Possible orientations for the space plane \mathbf{S} are then given by positions in (α, β) space where the curves cross. Horaud showed that, in general, 0, 1, or 2 solutions are possible. The case of no solutions corresponds to trying to make an impossible match between a corner and an image junction when totally the wrong angles φ , θ , ψ are assumed; one solution is the normal situation; and two solutions arise in the interesting special case when orthographic or near-orthographic projection permits perceptual reversals—i.e., a convex corner is interpreted as a concave corner or *vice versa*. In fact, under orthographic projection the image data from a single corner are insufficient, taken on their own, to give a unique interpretation. In this situation, even the human visual system makes mistakes—as in the case of the well-known Necker cube illusion (see Chapter 16). However, when such cases arise in practical situations, it may be better to take the convex rather than the concave corner interpretation as a working assumption, as it has slightly greater likelihood of being correct.

Horaud has shown that such ambiguities are frequently resolved if the space plane orientation is estimated simultaneously for all the junctions bordering the object face in question, by plotting the α and β values for all such junctions on the same α, β graph. For example, with a cube face on which there are three such junctions, nine curves are coincident at the correct solution, and there are nine points where only two curves cross, indicating false solutions. On the other hand, if the same cube is viewed under conditions approximating very closely to orthographic projection, two solutions with nine coincident curves appear and the situation remains unresolved, as before.

Overall, this technique is important in showing that although lines and angles individually lead to virtually unlimited numbers of possible interpretations of 3-D scenes, junctions lead individually to at most two solutions and any remaining ambiguity can normally be eliminated if junctions on the same face are considered together. As has been seen, the exception to this rule occurs when projection is accurately orthographic, although this is a situation that can often be avoided in practice.

So far we have considered only how a given hypothesis about the scene may be tested: nothing has been said about how assignments of the angles φ , θ , ψ are made to the observed junctions. Horaud's paper discussed this aspect of the work in some depth. In general, the approach is to use a depth-first search technique in which a match is “grown” from the initial most promising junction assignment. In fact, considerable preprocessing of sample data is carried out to find how to rank image features for their utility during depth-first search interpretation. The idea is

to order possible alternatives such as linear or circular arcs, convex or concave junctions, and short or long lines. In this way, the tree search becomes more planned and efficient at run time. Generally, the more frequently occurring types of feature should be weighted down in favor of the rarer types of feature, for greater search efficiency. In addition, remember that hypothesis generation is relatively expensive in that it demands a stage of back-projection, as described above. Ideally, this stage need be employed only once for each object (in the case that only a single corner is, initially, considered). Subsequent stages of processing then involve hypothesis verification in which other features of the object are predicted and their presences are sought in the image: if found they are used to refine the existing match; if the match at any stage becomes worse, then the algorithm backtracks and eliminates one or more features and proceeds with other ones. This process is unavoidable, since more than one image feature may be present near a predicted feature.

One of the factors that has been found to make the method converge quickly is the use of grouped rather than individual features, since this tends to decrease the combinatorial explosion in the size of the search. In the present context, this means that attempts should be made to match first all junctions or angles bordering a given object face, and further that a face should be selected that has the greatest number of matchable features around it.

In summary, this approach is successful since it backprojects from the image and then uses geometrical constraints and heuristic assumptions for matching in 3-D space. It is suitable for matching objects that possess planar faces and straight line boundaries, hence giving angle and junction features. However, extending the backprojection technique to situations where object faces are curved and have curved boundaries could be significantly more difficult.

15.11 AN IMPORTANT PARADIGM—LOCATION OF INDUSTRIAL PARTS

In this section, we consider the location of a common class of industrial part: this constitutes an important example that has to be solved in one way or another. Here we go along with the Bolles and Horaud (1986) approach as it leads to sensible solutions and embodies a number of useful didactic lessons. The method starts with a depth map of the scene (obtained in this case using structured lighting).

Figure 15.15 shows in simplified form the type of industrial part being sought in the images. In typical scenes several of these parts may appear jumbled on a worktable, with perhaps three or four being piled on top of each other in some places. In such cases, it is vital that the matching scheme be highly robust if most of the parts are to be found, since even when a part is unoccluded, it appears against a highly cluttered and confusing background. However, the parts themselves have reasonably simple shapes and possess certain salient features. In the

**FIGURE 15.15**

The essential features of the industrial components located by the 3DPO system of Bolles and Horaud (1986). S, C and T indicate respectively straight and circular dihedral edges and straight tangential edges, all of which are searched for by the system.

particular problem cited, each has a cylindrical base with a concentric cylindrical head, and also a planar shelf is attached symmetrically to the base. To locate such objects, it is natural to attempt to search for circular and straight dihedral edges. In addition, because of the type of data being used, it is useful to search for straight tangential edges, which appear where the sides of curved cylinders are viewed obliquely.

In general, circular dihedral edges appear elliptical, and parameters for five of the six degrees of freedom of the part can be determined by analyzing these edges. The parameter that cannot be determined in this way corresponds to rotation about the axis of symmetry of the cylinder.

Straight dihedral edges also permit five free parameters to be determined, since location of one plane eliminates three degrees of freedom and location of an adjacent plane eliminates a further two degrees of freedom. The parameter that remains undetermined is that of linear motion along the direction of the edge. However, there is also a further ambiguity in that the part may appear either way around on the dihedral edge.

Straight tangential edges determine only four free parameters, since the part is free to rotate about the axis of the cylinder and can also move along the tangential edge. Note that these edges are the most difficult to locate accurately, since range data are subject to greater levels of noise as surfaces curve away from the sensor.

All three of these types of edge are planar. They also provide useful additional information that can help to identify where they are on a part. For example, straight and curved dihedral edges both provide information on the size of the

included angle, and the curved edges also give radius values. In fact, curved dihedral edges provide significantly more parametric information about a part than either of the other two types of edge, and therefore they are of most use to form initial hypotheses about the pose (position and orientation) of a part. Having found such an edge, it is necessary to try out various hypotheses about which edge it is, e.g., by searching for other circular dihedral edges at specific relative positions: this is a vital hypothesis verification step. Next, the problem of how to determine the remaining free parameter is solved by searching for the linear straight dihedral edge features from the planar shelf on the part.

At this stage hypothesis generation is complete and the part is essentially found, but hypothesis verification is required (a) to confirm that the part is genuine and not an accidental grouping of independent features in the image, (b) to refine the pose estimate, and (c) to determine the “configuration” of the part, i.e., to what extent it is buried under other parts (making it difficult for a robot to pick it up). When the most accurate pose has been obtained, the overall degree of fit can be considered and the hypothesis rejected if some relevant criterion is not met.

In common with other researchers (Faugeras and Hebert, 1983; Grimson and Lozano-Perez, 1984), Bolles and Horaud took a depth-first tree search as the basic matching strategy. Their scheme uses a minimum number of features to key into the data, first generating hypotheses and then taking care to ensure verification (note that Bolles and Cain (1982) had earlier used this technique in a 2-D part location problem). This contrasts with much work (especially that based on the HT) that makes hypotheses but does not check them. (Note that forming the initial hypotheses is the difficult and computation intensive part of the work. Researchers will therefore write about this aspect of their work and perhaps not state the minor amount of computation that went into confirming that objects had indeed been located. Note also that in much 2-D work, images can be significantly simpler and the size of the peak in parameter space can be so large as to make it virtually certain that an object has been located—thus rendering verification unnecessary.)

15.12 CONCLUDING REMARKS

To the layman, 3-D vision is an obvious and automatic result of the fact that the human visual system is binocular, and presumes both that binocular vision is the only way to arrive at depth maps and that once they have been obtained the subsequent recognition process is trivial. However, what this chapter has actually demonstrated is that neither of these commonly held views is valid. First, there are a good many ways of arriving at depth maps, and some of them are available using monocular vision. Second, the complexity of the mathematical calculations involved in locating objects and the amount of abstract reasoning involved in obtaining robust solutions—plus the need to ensure that the latter are not

ambiguous—are taxing even in simple cases, including those where the objects have well-defined salient features.

Despite the diversity of methods covered in this chapter, there are certain important themes: the use of “trigger” features, the value of combining features into groups that are analyzed together, the need for working hypotheses to be generated at an early stage, the use of depth-first heuristic search (combined where appropriate with more rigorous breadth-first evaluation of the possible interpretations), and the detailed verification of hypotheses. All these can be taken as parts of current methodology; *details*, however, vary with the dataset. More specifically, if a new type of industrial part is to be considered, some study must be made of its most salient features: then this causes not only the feature detection scheme to vary but also the heuristics of the search employed—and also the mathematics of the hypothesis mechanism. The reader is referred to the following chapter for further discussion of object recognition under perspective projection.

While the previous two sections have concentrated on object recognition and have perhaps tended to eschew the value of range measurements and depth maps, it is possible that this might give a misleading impression of the situation. In fact, there are many situations where recognition is largely irrelevant but where it is mandatory to map out 3-D surfaces in great detail. Turbine blades, automobile body parts, or even food products such as fruit may need to be measured accurately in 3-D. In such cases, it is known in advance what object is in what position, but some inspection or measurement function has to be carried out and a diagnosis made. In such instances, the methods of structured lighting, stereopsis, or photometric stereo come into their own and are highly effective methods. Ultimately also, one might expect that a robot vision system will have to use all the tricks of the human visual system if it is to be as adaptable and useful when operating in an unconstrained environment rather than at a particular worktable.

This has been a preliminary chapter on 3-D vision, setting the scene for Parts 3 and 4. In particular, Chapter 16 will be devoted to a careful analysis of the distinction between weak and full perspective projection and how this affects the object recognition process; Chapter 17 will aim to show something of the elegance and value of invariants in providing short cuts around some of the complexities of full perspective projection; Chapter 18 will consider camera calibration and will also consider how recent research on interrelating multiple views of a scene has allowed some of the tedium of camera calibration to be by-passed; and Chapter 19 will introduce the topic of motion analysis in 3-D scenes.

Conventional wisdom indicates that binocular vision is the key to understanding the 3-D world. This chapter has shown that the correspondence problem makes the practice of binocular vision tedious, while the solutions it provides are only depth maps and require further intricate analysis before the 3-D world can fully be understood.

Motion

19

Motion is another aspect of 3-D vision that humans are able to interpret with ease. This chapter studies the basic theoretical concepts. It is left to Chapters 22 and 23 to apply them to real problems where motion is crucial, including the monitoring of traffic flow and the tracking of people.

Look out for:

- the basic concepts of optical flow, and its limitations.
- the idea of a focus of expansion, and how it leads to the possibility of “structure from motion.”
- how motion stereo is achieved.
- the important status of the Kalman filter in motion applications.
- the ways in which invariant features may be used for wide baseline matching.

Note that this introductory chapter on 3-D motion leads to important methods for performing vital surveillance tasks—as will be seen in Chapters 22 and 23.

19.1 INTRODUCTION

This chapter is concerned with the analysis of motion in digital images. For space reasons, it will not be possible to cover the whole subject comprehensively. Instead, the aim is to give the flavor of the subject, airing some of the principles that have proved important over the past two or three decades. Over much of the time, optical flow has been topical. It is appropriate to study it in fair detail because of its importance for surveillance and other applications. Later in the chapter, the use of the Kalman filter for tracking moving objects is discussed, and the use of invariant features such as the scale-invariant feature transform (SIFT) for wide baseline matching, also relevant to motion tracking, is covered.

19.2 OPTICAL FLOW

When scenes contain moving objects, analysis is necessarily more complex than for scenes where everything is stationary, since temporal variations in intensity have to be taken into account. However, intuition suggests that it should be possible—even straightforward—to segment moving objects by virtue of their motion. Image differencing over successive pairs of frames should permit this to be achieved. More careful consideration shows that things are not quite so simple, as illustrated in Fig. 19.1. The reason is that regions of constant intensity give no sign of motion, and edges parallel to the direction of motion give the appearance of not moving. Only edges with a component normal to the direction of motion carry information about the motion. In addition, there is some ambiguity in the direction of the velocity vector. This arises partly because there is too little information available within a small aperture to permit the full velocity vector to be computed (Fig. 19.2). This is hence called the *aperture problem*.

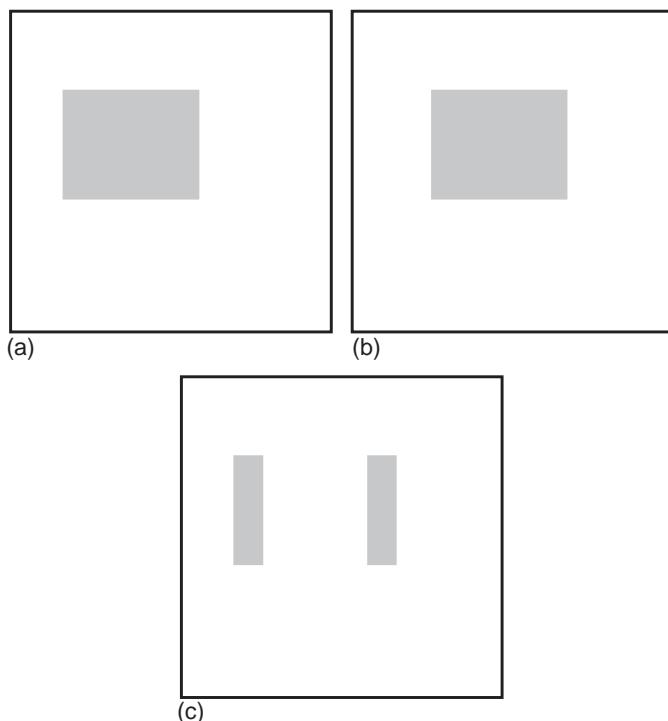


FIGURE 19.1

Effect of image differencing. The figure shows an object that has moved between frames (a) and (b). (c) The result of performing an image differencing operation. Note that the edges parallel to the direction of motion do not show up in the difference image. Also, regions of constant intensity give no sign of motion.

These elementary ideas can be taken further, and they lead to the notion of optical flow, wherein a local operator which is applied at all pixels in the image will lead to a motion vector field that varies smoothly over the whole image. The attraction lies in the use of a local operator, with its limited computational burden. Ideally, it would have an overhead comparable to an edge detector in a normal intensity image—although clearly it will have to be applied locally to pairs of images in an image sequence.

We start by considering the intensity function $I(x, y, t)$ and expanding it in a Taylor series:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + I_x dx + I_y dy + I_t dt + \dots \quad (19.1)$$

where second- and higher order terms have been ignored. In this equation, I_x , I_y , and I_t denote respective partial derivatives with respect to x , y , and t .

We next set the local condition that the image has shifted by amount (dx, dy) in time dt so that it is functionally identical at $(x + dx, y + dy, t + dt)$ and (x, y, t) :

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \quad (19.2)$$

Hence, we can deduce:

$$I_t = -(I_x \dot{x} + I_y \dot{y}) \quad (19.3)$$

Writing the local velocity \mathbf{v} in the form:

$$\mathbf{v} = (v_x, v_y) = (\dot{x}, \dot{y}) \quad (19.4)$$

we find:

$$I_t = -(I_x v_x + I_y v_y) = -\nabla I \cdot \mathbf{v} \quad (19.5)$$

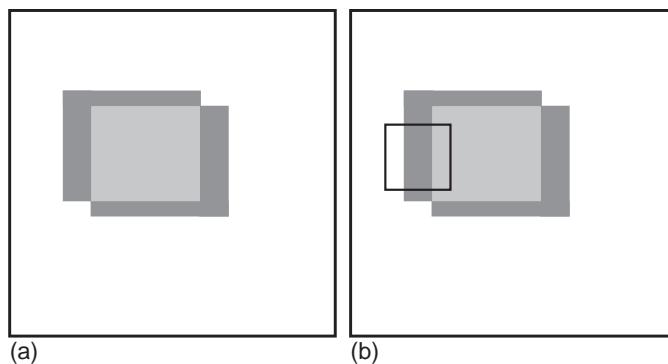


FIGURE 19.2

The aperture problem. The figure illustrates the aperture problem. (a) (Dark gray) regions of motion of an object whose central uniform region (light gray) gives no sign of motion. (b) Depiction of how little is visible in a small aperture (black border), thereby leading to ambiguity in the deduced direction of motion of the object.

I_t can be measured by subtracting pairs of images in the input sequence, while ∇I can be estimated by Sobel or other gradient operators. Hence, it should be possible to deduce the velocity field $\mathbf{v}(x, y)$ using the above equation. Unfortunately, this equation is a scalar equation and will not suffice for determining the two local components of the velocity field as we require. There is a further problem with this equation—that the velocity value will depend on the values of both I_t and ∇I , and these quantities are only estimated approximately by the respective differencing operators. In both cases, significant noise will arise, and this will be exacerbated by taking the ratio in order to calculate \mathbf{v} .

Let us now return to the problem of computing the full velocity field $\mathbf{v}(x, y)$. All we know about \mathbf{v} is that its components lie on the following line in (v_x, v_y) -space (Fig. 19.3):

$$I_x v_x + I_y v_y + I_t = 0 \quad (19.6)$$

This line is normal to the direction (I_x, I_y) , and has a distance from the (velocity) origin that is equal to:

$$|\mathbf{v}| = \frac{-I_t}{(I_x^2 + I_y^2)^{1/2}} \quad (19.7)$$

Clearly, we need to deduce the component of \mathbf{v} along the line given by Eq. (19.6). However, there is no purely local means of achieving this with first derivatives of the intensity function. The accepted solution (Horn and Schunck,

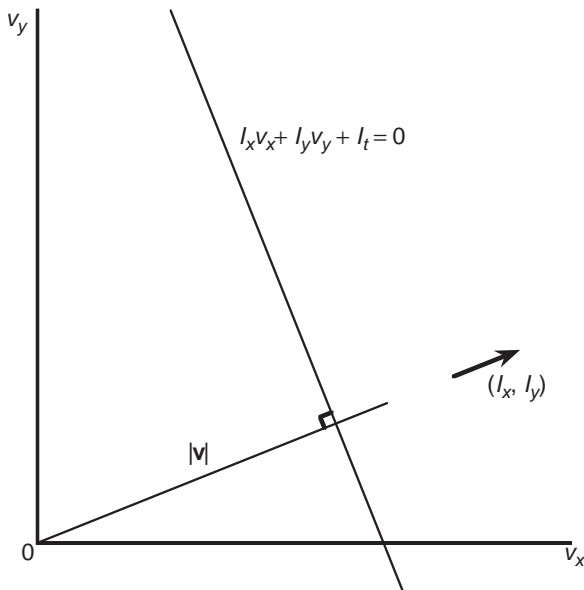


FIGURE 19.3

Computation of the velocity field. The graph shows the line in velocity space on which the velocity vector \mathbf{v} must lie. The line is normal to the direction (I_x, I_y) and its distance from the origin is known to be $|\mathbf{v}|$ (see text).

1981) is to use relaxation labeling to arrive iteratively at a self-consistent solution that minimizes the global error. In principle, this approach will also minimize the noise problem indicated earlier.

In fact, there are still problems with the method. Essentially, these arise as there are liable to be vast expanses of the image where the intensity gradient is low. In that case, only very inaccurate information is available about the velocity component parallel to ∇I , and the whole problem becomes ill-conditioned. On the other hand, in a highly textured image, this situation should not arise (assuming the texture has a large enough grain size to give good differential signals).

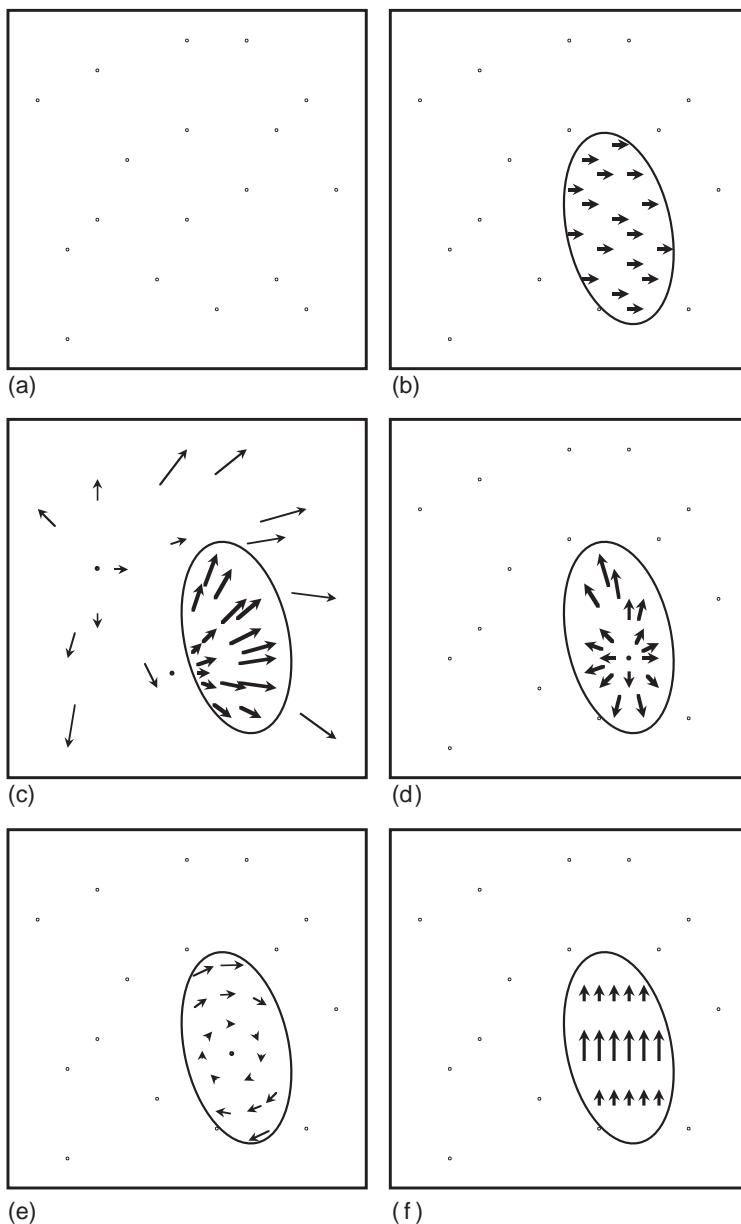
Finally, we return to the idea mentioned at the beginning of this section—that edges parallel to the direction of motion would not give useful motion information. Such edges will have edge normals normal to the direction of motion, so ∇I will be normal to \mathbf{v} . Thus, from Eq. (19.5), I_t will be zero. In addition, regions of constant intensity will have $\nabla I = 0$, so again I_t will be zero. It is interesting and highly useful that such a simple equation (19.5) embodies all the cases that were suggested earlier on the basis of intuition.

In what follows, we assume that the optical flow (velocity field) image has been computed satisfactorily, that is, without the disadvantages of inaccuracy or ill-conditioning. It must now be interpreted in terms of moving objects and in some cases a moving camera. In fact, we shall ignore motion of the camera by remaining within its frame of reference.

19.3 INTERPRETATION OF OPTICAL FLOW FIELDS

We start by considering a case where no motion is visible. In that case, the velocity field image contains only vectors of zero length (Fig. 19.4(a)). Next, we take a case where one object is moving toward the right, with a simple effect on the velocity field image (Fig. 19.4(b)). Next, we consider the case where the camera is moving forward; in this case, all the stationary objects in the field of view appear to be diverging from a point, which is called the *focus of expansion* (FOE)—see Fig. 19.4(c); this image also shows an object that is moving rapidly past the camera and has its own separate FOE. Figure 19.4(d) shows the case of an object moving directly toward the camera. In this case, its FOE lies within its outline. Similarly, objects that are receding appear to move away from the *focus of contraction*. Next, there are objects that are stationary but rotating about the line of sight. For these, the vector field appears as in Fig. 19.4(e). There is a final case that is also quite simple: an object that is stationary but rotating about an axis normal to the line of sight; if the axis is horizontal, then the features on the object will appear to be moving up or down, while paradoxically the object itself remains stationary (Fig. 19.4(f))—although its outline could oscillate as it rotates.

So far, we have only dealt with cases in which pure translational or pure rotational motion is occurring. If a rotating meteor is rushing past, or a spinning cricket ball is approaching, then both types of motion will occur together. In that

**FIGURE 19.4**

Interpretation of velocity flow fields. (a) A case where the object features all have zero velocity. (b) A case where an object is moving to the right. (c) A case where the camera is moving into the scene, and the stationary object features appear to be diverging from a focus of expansion (FOE), while a single large object is moving past the camera and away from a separate FOE. In (d), an object is moving directly toward the camera that is stationary: the object's FOE lies within its outline. In (e), an object is rotating about the line of sight to the camera, and in (f), the object is rotating about an axis perpendicular to the line of sight. In all cases, the length of the arrow indicates the magnitude of the velocity vector.

case, unraveling the motion will be far more complex. We shall not solve this problem here, but refer the reader to more specialized texts (e.g., Maybank, 1992). However, the complexity is due to the way depth (Z) creeps into the calculations. First, note that pure rotational motion with rotation about the line of sight does not depend on Z . All we have to measure is the angular velocity, and this can be done quite simply.

19.4 USING FOCUS OF EXPANSION TO AVOID COLLISION

We now take a simple case in which an FOE is located in an image and show how it is possible to deduce the distance of closest approach of the camera to a fixed object of known coordinates. This type of information is valuable for guiding robot arms or robot vehicles and helping to avoid collisions.

In the notation of Chapter 15, we have the following formulas for the location of an image point (x, y, z) resulting from a world point (X, Y, Z) :

$$x = \frac{fX}{Z} \quad (19.8)$$

$$y = \frac{fY}{Z} \quad (19.9)$$

$$z = f \quad (19.10)$$

Assuming the camera has a motion vector $(-\dot{X}, -\dot{Y}, -\dot{Z}) = (-u, -v, -w)$, fixed world points will have velocity (u, v, w) relative to the camera. Now a point (X_0, Y_0, Z_0) will after a time t appear to move to $(X, Y, Z) = (X_0 + ut, Y_0 + vt, Z_0 + wt)$ with image coordinates:

$$(x, y) = \left(\frac{f(X_0 + ut)}{Z_0 + wt}, \frac{f(Y_0 + vt)}{Z_0 + wt} \right) \quad (19.11)$$

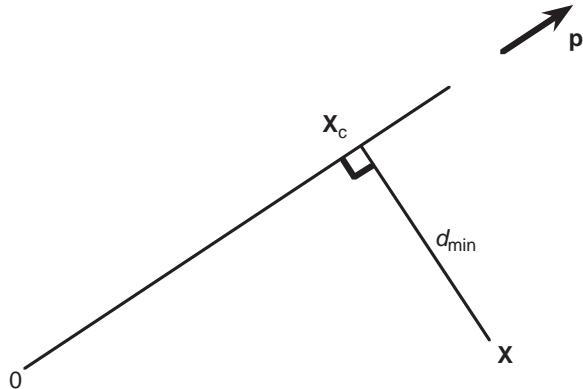
and as $t \rightarrow \infty$ this approaches the focus of expansion $F(fu/w, fv/w)$. This point is in the image, but the true interpretation is that the actual motion of the center of projection of the imaging system is toward the point:

$$\mathbf{p} = \left(\frac{fu}{w}, \frac{fv}{w}, f \right) \quad (19.12)$$

(This is of course consistent with the motion vector (u, v, w) assumed initially.) The distance moved during time t can now be modeled as:

$$\mathbf{X}_c = (X_c, Y_c, Z_c) = \alpha t \mathbf{p} = f \alpha t \left(\frac{u}{w}, \frac{v}{w}, 1 \right) \quad (19.13)$$

where α is a normalization constant. To calculate the distance of closest approach of the camera to the world point $\mathbf{X} = (X, Y, Z)$, we merely specify that the vector

**FIGURE 19.5**

Calculation of distance of closest approach. Here, the camera is moving from 0 to \mathbf{X}_c in the direction \mathbf{p} , not in a direct line to the object at \mathbf{X} . d_{\min} is the distance of closest approach.

$\mathbf{X}_c - \mathbf{X}$ be perpendicular to \mathbf{p} (Fig. 19.5) so that:

$$(\mathbf{X}_c - \mathbf{X}) \cdot \mathbf{p} = 0 \quad (19.14)$$

$$\text{i.e., } (\alpha t \mathbf{p} - \mathbf{X}) \cdot \mathbf{p} = 0 \quad (19.15)$$

$$\therefore \alpha t \mathbf{p} \cdot \mathbf{p} = \mathbf{X} \cdot \mathbf{p} \quad (19.16)$$

$$\therefore t = \frac{\mathbf{X} \cdot \mathbf{p}}{\alpha(\mathbf{p} \cdot \mathbf{p})} \quad (19.17)$$

Substituting in the equation for \mathbf{X}_c now gives:

$$\mathbf{X}_c = \frac{\mathbf{p}(\mathbf{X} \cdot \mathbf{p})}{\mathbf{p} \cdot \mathbf{p}} \quad (19.18)$$

Hence, the minimum distance of approach is given by:

$$\begin{aligned} d_{\min}^2 &= \left[\frac{\mathbf{p}(\mathbf{X} \cdot \mathbf{p})}{\mathbf{p} \cdot \mathbf{p}} - \mathbf{X} \right]^2 = \frac{(\mathbf{X} \cdot \mathbf{p})^2}{(\mathbf{p} \cdot \mathbf{p})} - \frac{2(\mathbf{X} \cdot \mathbf{p})^2}{(\mathbf{p} \cdot \mathbf{p})} + (\mathbf{X} \cdot \mathbf{X}) \\ &= (\mathbf{X} \cdot \mathbf{X}) - \frac{(\mathbf{X} \cdot \mathbf{p})^2}{(\mathbf{p} \cdot \mathbf{p})} \end{aligned} \quad (19.19)$$

which is naturally zero when \mathbf{p} is aligned along \mathbf{X} . Clearly, avoidance of collisions requires an estimate of the size of the machine (e.g., robot or vehicle) attached to the camera and the size to be associated with the world point feature \mathbf{X} . Finally, note that while \mathbf{p} is obtained from the image data, \mathbf{X} can only be deduced from the image data if the depth Z can be estimated from other information. In fact, this information should be available from time-to-adjacency analysis (see below) if the speed of the camera through space (and specifically w) is known.

19.5 TIME-TO-ADJACENCY ANALYSIS

In this section, we consider the extent to which the depths of objects can be deduced from optical flow. First, note that features on the same object share the same FOE, and this can help us to identify them. But how can we get information on the depths of the various features on the object from optical flow? The basic approach is to start with the coordinates of a general image point (x, y) , deduce its flow velocity, and then find an equation linking this with the depth Z .

Taking the general image point (x, y) given in Eq. (19.11), we find:

$$\begin{aligned}\dot{x} &= \frac{f[(Z_0 + wt)u - (X_0 + ut)w]}{(Z_0 + wt)^2} \\ &= f \frac{(Zu - Xw)}{Z^2}\end{aligned}\quad (19.20)$$

and

$$\dot{y} = f \frac{(Zv - Yw)}{Z^2} \quad (19.21)$$

Hence:

$$\begin{aligned}\frac{\dot{x}}{\dot{y}} &= \frac{(Zu - Xw)}{(Zv - Yw)} = \frac{(u/w - X/Z)}{(v/w - Y/Z)} \\ &= \frac{(x - x_F)}{(y - y_F)}\end{aligned}\quad (19.22)$$

This result was to be expected, as the motion of the image point has to be directly away from the focus of expansion (x_F, y_F) . Without loss of generality, we now take a set of axes such that the image point considered is moving along the x -axis. Then we have:

$$\dot{y} = 0 \quad (19.23)$$

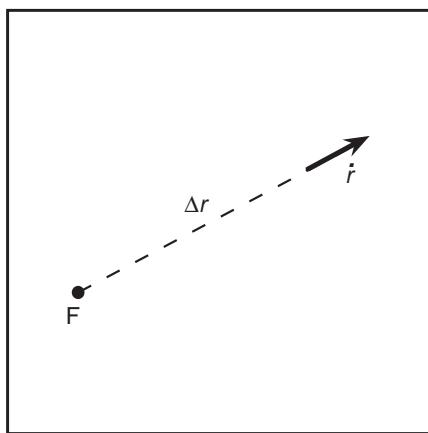
$$y_F = y = \frac{fY}{Z} \quad (19.24)$$

Defining the distance from the focus of expansion as Δr (see Fig. 19.6), we find:

$$\Delta r = \Delta x = x - x_F = \frac{fX}{Z} - \frac{fu}{w} = \frac{f(Xw - Zu)}{Zw} \quad (19.25)$$

$$\therefore \frac{\Delta r}{\dot{r}} = \frac{\Delta x}{\dot{x}} = -\frac{Z}{w} \quad (19.26)$$

Defining the *time to adjacency* T_a as the time it will take for the origin of the camera coordinate system to arrive at the object point, Eq. (19.26) means that T_a is the same (Z/w) when the object is observed in real-world coordinates as when it is observed in image coordinates $(-\Delta r/\dot{r})$. Hence, it is possible to relate the

**FIGURE 19.6**

Calculation of time to adjacency. Here, an object feature is moving directly away from the focus of expansion F with speed \dot{r} . At the time of observation, the distance of the feature from F is Δr . These measurements permit the time to adjacency and hence also the relative depth of the feature to be calculated.

optical flow vectors for object points at different depths in the scene. This is important, as the assumption of identical values of w now allows us to determine the relative depths of object points merely from their apparent motion parameters:

$$\frac{Z_1}{Z_2} = \frac{\Delta r_1 / \Delta r_2}{\dot{r}_1 / \dot{r}_2} \quad (19.27)$$

This is thus the first step in the determination of structure from motion. In this context, note how the implicit assumption that the objects under observation are rigid is included—namely, that all points on the same object are characterized by identical values of w . The assumption of rigidity underlies much of the work on interpretation of motion in images.

19.6 BASIC DIFFICULTIES WITH THE OPTICAL FLOW MODEL

When the optical flow ideas presented above are tried on real images, certain problems arise that are not apparent from the above model. First, not all edge points that should appear in the motion image are actually present. This is due to the contrast between the moving object and the background vanishing locally and limiting visibility. The situation is exactly as for edges that are located by edge detection operators in nonmoving images. The contrast simply drops to a low value in certain localities and the edge peters out. This signals that the edge model, and now the velocity flow model, is limited and such local procedures are *ad hoc* and too impoverished to permit proper segmentation unaided.

Here, we take the view that simple models can be useful, but they become inadequate on certain occasions and robust methods are required to overcome the problems that then arise. Some of the problems were noticed by Horn as early as 1986 (Horn, 1986). First, a smooth sphere may be rotating but the motion will not show up in an optical flow (difference) image. We can if we wish regard this as a simple optical illusion, as the rotation of the sphere may well be invisible to the eye too. Second, a motionless sphere may *appear* to rotate as the light rotates around it. The object is simply subject to the laws of Lambertian optics, and again we may if we wish regard this effect as an optical illusion. (The illusion is relative to the baseline provided by the *normally correct* optical flow model.)

We next return to the optical flow model and see where it could be wrong or misleading. The answer is at once apparent: we stated in writing Eq. (19.2) that we were assuming the *image* is being shifted. Yet it is not images that shift but the objects imaged within them. Thus, we ought to be considering the images of objects moving against a fixed background (or a variable background if the camera is moving). This will then permit us to see how sections of the motion edge can go from high to low contrast and back again in a rather fickle way, which we must nevertheless allow for in our algorithms. With this in mind, it should be permissible to go on using optical flow and difference imaging, even though these concepts have distinctly limited theoretical validity. (For a more thoroughgoing analysis of the underlying theory, see Faugeras, 1993.)

19.7 STEREO FROM MOTION

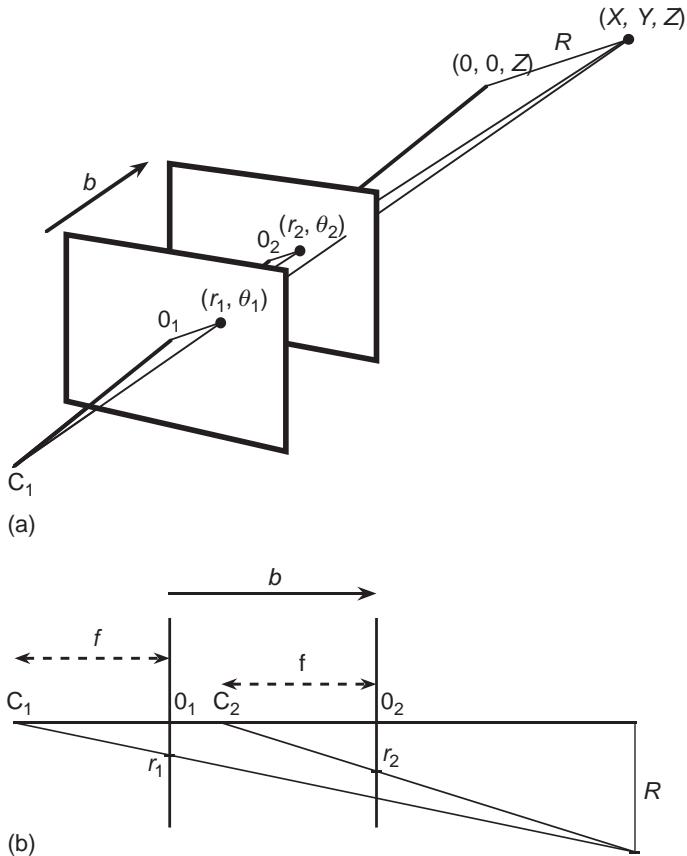
An interesting aspect of camera motion is that over time the camera sees a succession of images that span a baseline in a similar way to binocular (stereo) images. Thus, it should be possible to obtain depth information by taking two such images and tracking object features between them. The technique is in principle more straightforward than normal stereo imaging in that feature tracking is possible, so the correspondence problem should be nonexistent. However, there is a difficulty in that the object field is viewed from almost the same direction in the succession of images so that the full benefit of the available baseline is not obtained (Fig. 19.7). We can analyze the effect as follows.

First, in the case of camera motion, the equations for lateral displacement in the image depend not only on X but also on Y , although we can make a simplification in the theory by working with R , the radial distance of an object point from the optical axis of the camera, where:

$$R = (X^2 + Y^2)^{1/2} \quad (19.28)$$

We now obtain the radial distances in the two images as:

$$r_1 = \frac{Rf}{Z_1} \quad (19.29)$$

**FIGURE 19.7**

Calculation of stereo from camera motion. (a) Depiction of how stereo imaging can result from camera motion, the vector \mathbf{b} representing the baseline. (b) The simplified planar geometry required to calculate the disparity. It is assumed that the motion is directly along the optical axis of the camera.

$$r_2 = \frac{Rf}{Z_2} \quad (19.30)$$

so the disparity is:

$$D = r_2 - r_1 = Rf \left(\frac{1}{Z_2} - \frac{1}{Z_1} \right) \quad (19.31)$$

Writing the baseline as:

$$b = Z_1 - Z_2 \quad (19.32)$$

and assuming $b \ll Z_1, Z_2$, and then dropping the suffices, gives:

$$D = \frac{Rbf}{Z^2} \quad (19.33)$$

While this would appear to mitigate against finding Z without knowing R , we can overcome this problem by observing that:

$$\frac{R}{Z} = \frac{r}{f} \quad (19.34)$$

where r is approximately the mean value $\frac{1}{2}(r_1 + r_2)$. Substituting for R now gives:

$$D = \frac{br}{Z} \quad (19.35)$$

Hence, we can deduce the depth of the object point as:

$$Z = \frac{br}{D} = \frac{br}{(r_2 - r_1)} \quad (19.36)$$

This equation should be compared with Eq. (15.5) representing the normal stereo situation. The important point to note is that for motion stereo, the disparity depends on the radial distance r of the image point from the optical axis of the camera, whereas for normal stereo the disparity is independent of r ; as a result, motion stereo gives no depth information for points on the optical axis, and the accuracy of depth information depends on the magnitude of r .

19.8 THE KALMAN FILTER

When tracking moving objects, it is desirable to be able to predict where they will be in future frames, as this will make maximum use of preexisting information and permit the least amount of search in the subsequent frames. It will also serve to offset the problems of temporary occlusion, such as when one vehicle passes behind another, or one person passes behind another, or even when one limb of a person passes behind another. (There are also many military needs for tracking prediction, and others on the sports field.) The obvious equations to employ for this purpose involve sequentially updating the position and the velocity of points on the object being tracked:

$$x_i = x_{i-1} + v_{i-1} \quad (19.37)$$

$$v_i = x_i - x_{i-1} \quad (19.38)$$

assuming, for convenience, a unit time interval between each pair of samples.

In fact, this approach is too crude to yield the best results. First, it is necessary to make three quantities explicit: (1) the raw measurements (e.g., x), (2) the best estimates of the values of the corresponding variables *before* observation (denoted by $\bar{\cdot}$), and (3) the best estimates of these same model parameters *following* observation (denoted by $^+$). In addition, it is necessary to include explicit noise terms so that rigorous optimization procedures can be derived for making the best estimates.

In the particular case outlined above, the velocity—and possible variations on it which we shall ignore here for simplicity—constitutes a best estimate model parameter. We include position measurement noise by the parameter u and velocity (model) estimation noise by the parameter w . The above equations now become:

$$x_i^- = x_{i-1}^+ + v_{i-1} + u_{i-1} \quad (19.39)$$

$$v_i^- = v_{i-1}^+ + w_{i-1} \quad (19.40)$$

In the case that the velocity is constant and the noise is Gaussian, we can spot the optimum solutions to this problem:

$$x_i^- = x_{i-1}^+ \quad (19.41)$$

$$\sigma_i^- = \sigma_{i-1}^+ \quad (19.42)$$

these being called the *prediction* equations, and

$$x_i^+ = \frac{x_i/\sigma_i^2 + (x_i^-)/(\sigma_i^-)^2}{1/\sigma_i^2 + 1/(\sigma_i^-)^2} \quad (19.43)$$

$$\sigma_i^+ = \left[\frac{1}{1/\sigma_i^2 + 1/(\sigma_i^-)^2} \right]^{1/2} \quad (19.44)$$

these being called the *correction* equations.¹ (In these equations, σ^\pm are the standard deviations for the respective model estimates x^\pm , and σ is the standard deviation for the raw measurement x .)

What these equations show is how repeated measurements improve the estimate of the position parameter and the error upon it at each iteration. Note the particularly important feature—that the noise is being modeled as well as the position itself. This permits all positions earlier than $i - 1$ to be forgotten. The fact that there were many such positions, whose values can all be averaged to improve the accuracy of the latest estimate, is of course rolled up into the values of x_i^- and σ_i^- , and eventually into the values for x_i^+ and σ_i^+ .

The next problem is how to generalize this result, both to multiple variables and to possibly varying velocity and acceleration. This is the function of the widely used Kalman filter. It achieves this by continuing with a linear approximation and by employing a state vector comprising position, velocity, and acceleration (or other relevant parameters), all in one state vector s . This constitutes the dynamic model. The raw measurements x have to be considered separately.

In the general case, the state vector is not updated simply by writing:

$$s_i^- = s_{i-1}^+ \quad (19.45)$$

¹The latter are nothing more than the well-known equations for weighted averages (Cowan, 1998).

but requires a fuller exposition because of the interdependence of position, velocity, and acceleration; hence, we have:²

$$s_i^- = K_i s_{i-1}^+ \quad (19.46)$$

Similarly, the standard deviations σ_i , σ_i^\pm in Eqs. (19.42)–(19.44) (or rather, the corresponding variances) have to be replaced by the covariance matrices Σ_i , Σ_i^\pm , and the equations become significantly more complicated. We will not go into the calculations fully here as they are nontrivial and need several pages to iterate. Suffice it to say that the aim is to produce an optimum linear filter by a least-squares calculation (see, e.g., Maybeck, 1979).

Overall, the Kalman filter is the optimal estimator for a linear system for which the noise is zero mean, white, and Gaussian, although it will often provide good estimates even if the noise is not Gaussian.

Finally, it will be noted that the Kalman filter itself works by averaging processes, which will give erroneous results if any outliers are present. This will certainly occur in most motion applications. Thus, there is a need to test each prediction to determine if it is too far away from reality. If this is the case, it is not unlikely that the object in question has become partially or fully occluded. A simple option is to assume that the object continues in the same motion (albeit with a larger uncertainty as time goes on), and to wait for it to emerge from behind another object. At the very least, it is prudent to keep a number of such possibilities alive for some time, but the extent of this will naturally vary from situation to situation and from application to application.

19.9 WIDE BASELINE MATCHING

The need for wide baseline matching was noted in Chapter 6, where considerable discussion was included on detection of suitable invariant features (see Section 6.7 and its various subsections). The topic has been left until the present chapter because it is relevant for both 3-D vision and motion analysis, and the latter topic has only been covered in this chapter. The wide baseline scenario arises from situations where the same object is viewed from widely different directions, with the result that its appearance may change dramatically so that it may become extremely difficult to recognize. While narrow baseline stereo is the norm for depth estimation using two cameras, wide baselines are common in surveillance applications—e.g., where a pedestrian precinct is viewed by several independent cameras that are widely separated, as described in Chapter 22. They are also the norm when objects are being sought in image databases. However, one of the most likely situations when they occur is with objects that are in motion. While this may appear to be immaterial in surveillance or

²Some authors write K_{i-1} in this equation, but it is only a matter of definition whether the label matches the previous or the new state.

with driver assistance systems, because every pair of frames will give instances of narrow baseline stereo, it can easily happen that objects will be *temporarily occluded* and come back into view with different orientations or backgrounds; in addition, the *attention* of the software (like that of a human operator) may only be on part of the scene for part of the time. Hence, wide baseline viewing is bound to be a common consequence of motion. Overall, then, wide baseline matching techniques will be needed in a variety of instances of 3-D viewing and motion tracking.

Chapter 6 showed how features could be designed to cover a variety of wide baseline views as far apart as 50° . In these circumstances, an important factor in designing suitable feature detectors is to make them invariant to scale and affine distortions. However, that alone is not enough. The feature detectors must also provide descriptors of each feature that are sufficiently rich in information that matching between views is made as unambiguous as possible. In that way, wide baseline matching has a chance of being highly reliable. Lowe (2004) has found that reliable recognition of objects is possible with as few as three features. Indeed, it is highly important to aim to achieve this when images typically contain several thousand features that come from many different objects as well as background clutter. In this way, the number of false positives is reduced to minimal levels, and there is a high chance of detecting all objects of the chosen type in the input image. That this can be possible is underlined by the richness of Lowe's SIFT features whose descriptors contain 128 parameters. (As discussed in Chapter 6, features devised by other workers may contain fewer parameters, but in the end risk not working in all possible scenarios.)

Granted that wide baseline matching is desirable, and that SIFT and other features have rich descriptor sets, how should the matching actually be achieved? Ideally, all that is necessary is to compare the feature descriptors from each pair of images, and find which ones match well, and which therefore lead to co-recognition of objects in the two views. Clearly, the first requirement is a similarity test for pairs of features. Lowe (2004) achieved this using a nearest neighbor (Euclidean) distance measure in his 128-dimensional descriptor space. He then used a Hough transform to identify clusters of features giving the same interpretations of poses for objects appearing in the two images. Because of the relatively small number of inliers that may occur in this type of situation, he found that the Hough transform approach performed significantly better than RANSAC (Section 11.6). Mikolajczyk and Schmid (2004) used a Mahalanobis distance measure for selecting the most similar descriptors to obtain a set of initial matches; they then used cross-correlation to reject low-score matches; finally, they performed a robust estimation of the transformation between the two images using RANSAC. Tuytelaars and Van Gool (2004) developed this further, using semilocal constraints involving geometric consistency and photometric constraints to refine the selection of matches before (again) relying on RANSAC to perform the final robust estimation of poses. In contrast to the approaches outlined above, Bay

et al. (2008) fed the descriptor information to a naive Bayes' classifier working on a “bag-of-words” representation (Dance et al., 2004) in order to perform object recognition. Bay et al. (2008) make no mention of determination of object pose in this application, which was targeted more at recognizing objects in an image database—although it could equally well have been targeted at repeated recognition of cars on the road for which pose would not be especially relevant.

Overall, it is clear that the new regime of utilizing invariant feature detectors with rich descriptors of local image content forms a powerful approach to wide baseline object matching and takes much of the heat out of the subsequent algorithms.

19.10 CONCLUDING REMARKS

In Sections 19.2 and 19.3, we described the formation of optical flow fields and showed how a moving object or a moving camera leads to a focus of expansion. In the case of moving objects, the focus of expansion can be used to decide whether a collision will occur. In addition, analysis of the motion taking account of the position of the focus of expansion led to the possibility of determining structure from motion. Specifically, this can be achieved via time-to-adjacency analysis, which yields the relative depth in terms of the motion parameters measurable directly from the image. We then demonstrated some basic difficulties with the optical flow model, which arise since the motion edge can have a wide range of contrast values, making it difficult to measure motion accurately. In practice, this means that larger time intervals may have to be employed to increase the motion signal. Otherwise, feature-based matching related to that of Chapter 14 can be used. Corners are the features that are most widely used for this purpose because they are ubiquitous and highly localized in 3-D. Space prevents details of this approach from being described here. Details may be found in Barnard and Thompson (1980), Scott (1988), Shah and Jain (1984), and Ullman (1979). However, the value of the Kalman filter for alleviating the difficulties of temporary occlusion has been considered, and the use of invariant features for wide baseline matching (which includes motion tracking applications) has been covered.

Further work on motion as it arises in real applications will be dealt with in Chapters 22 and 23, which address the problems of surveillance and in-vehicle vision systems.

The obvious way to understand motion is by image differencing and determination of optical flow. This chapter has shown that the “aperture problem” is a difficulty that is avoidable by using corner tracking. Further difficulties are caused by temporary occlusions, thus necessitating techniques such as occlusion reasoning and Kalman filtering.

(Section 14.3.2). We also present applications of location recognition (Section 14.3.3).

In the second half of the chapter, we address the most challenging variant of recognition, namely the problem of category recognition (Section 14.4). This includes approaches that use bags of features (Section 14.4.1), parts (Section 14.4.2), and segmentation (Section 14.4.3). We show how such techniques can be used to automate photo editing tasks, such as 3D modeling, scene completion, and creating collages (Section 14.4.4). Next, we discuss the role that context can play in both individual object recognition and more holistic scene understanding (Section 14.5). We close this chapter with a discussion of databases and test sets for constructing and evaluating recognition systems (Section 14.6).

While there is no comprehensive reference on object recognition, an excellent set of notes can be found in the ICCV 2009 short course (Fei-Fei, Fergus, and Torralba 2009), Antonio Torralba's more comprehensive MIT course (Torralba 2008), and two recent collections of papers (Ponce, Hebert, Schmid *et al.* 2006; Dickinson, Leonardis, Schiele *et al.* 2007) and a survey on object categorization (Pinz 2005). An evaluation of some of the best performing recognition algorithms can be found on the PASCAL Visual Object Classes (VOC) Challenge Web site at <http://pascallin.ecs.soton.ac.uk/challenges/VOC/>.

14.1 Object detection

If we are given an image to analyze, such as the group portrait in Figure 14.2, we could try to apply a recognition algorithm to every possible sub-window in this image. Such algorithms are likely to be both slow and error-prone. Instead, it is more effective to construct special-purpose *detectors*, whose job it is to rapidly find likely regions where particular objects might occur.

We begin this section with face detectors, which are some of the more successful examples of recognition. For example, such algorithms are built into most of today's digital cameras to enhance auto-focus and into video conferencing systems to control pan-tilt heads. We then look at pedestrian detectors, as an example of more general methods for object detection. Such detectors can be used in automotive safety applications, e.g., detecting pedestrians and other cars from moving vehicles (Leibe, Cornelis, Cornelis *et al.* 2007).

14.1.1 Face detection

Before face recognition can be applied to a general image, the locations and sizes of any faces must first be found (Figures 14.1c and 14.2). In principle, we could apply a face recognition algorithm at every pixel and scale (Moghaddam and Pentland 1997) but such a process would be too slow in practice.



Figure 14.2 Face detection results produced by Rowley, Baluja, and Kanade (1998a) © 1998 IEEE. Can you find the one false positive (a box around a non-face) among the 57 true positive results?

Over the years, a wide variety of fast face detection algorithms have been developed. Yang, Kriegman, and Ahuja (2002) provide a comprehensive survey of earlier work in this field; Yang’s ICPR 2004 tutorial² and the Torralba (2007) short course provide more recent reviews.³

According to the taxonomy of Yang, Kriegman, and Ahuja (2002), face detection techniques can be classified as feature-based, template-based, or appearance-based. Feature-based techniques attempt to find the locations of distinctive image features such as the eyes, nose, and mouth, and then verify whether these features are in a plausible geometrical arrangement. These techniques include some of the early approaches to face recognition (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991), as well as more recent approaches based on modular eigenspaces (Moghaddam and Pentland 1997), local filter jets (Leung, Burl, and Perona 1995; Penev and Atick 1996; Wiskott, Fellous, Krüger *et al.* 1997), support vector machines (Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007), and boosting (Schneiderman and Kanade 2004).

Template-based approaches, such as active appearance models (AAMs) (Section 14.2.2), can deal with a wide range of pose and expression variability. Typically, they require good initialization near a real face and are therefore not suitable as fast face detectors.

² <http://vision.ai.uiuc.edu/mhyang/face-detection-survey.html>.

³ An alternative approach to detecting faces is to look for regions of skin color in the image (Forsyth and Fleck 1999; Jones and Rehg 2001). See Exercise 2.8 for some additional discussion and references.

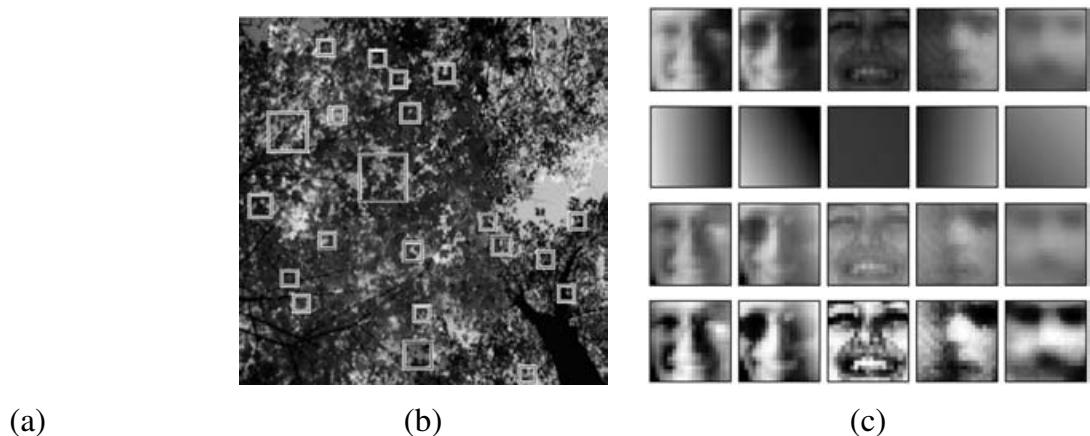


Figure 14.3 Pre-processing stages for face detector training (Rowley, Baluja, and Kanade 1998a) © 1998 IEEE: (a) artificially mirroring, rotating, scaling, and translating training images for greater variability; (b) using images without faces (looking up at a tree) to generate non-face examples; (c) pre-processing the patches by subtracting a best fit linear function (constant gradient) and histogram equalizing.

Appearance-based approaches scan over small overlapping rectangular patches of the image searching for likely face candidates, which can then be refined using a *cascade* of more expensive but selective detection algorithms (Sung and Poggio 1998; Rowley, Baluja, and Kanade 1998a; Romdhani, Torr, Schölkopf *et al.* 2001; Fleuret and Geman 2001; Viola and Jones 2004). In order to deal with scale variation, the image is usually converted into a sub-octave pyramid and a separate scan is performed on each level. Most appearance-based approaches today rely heavily on training classifiers using sets of labeled face and non-face patches.

Sung and Poggio (1998) and Rowley, Baluja, and Kanade (1998a) present two of the earliest appearance-based face detectors and introduce a number of innovations that are widely used in later work by others.

To start with, both systems collect a set of labeled face patches (Figure 14.2) as well as a set of patches taken from images that are known not to contain faces, such as aerial images or vegetation (Figure 14.3b). The collected face images are augmented by artificially mirroring, rotating, scaling, and translating the images by small amounts to make the face detectors less sensitive to such effects (Figure 14.3a).

After an initial set of training images has been collected, some optional pre-processing can be performed, such as subtracting an average gradient (linear function) from the image to compensate for global shading effects and using histogram equalization to compensate for varying camera contrast (Figure 14.3c).

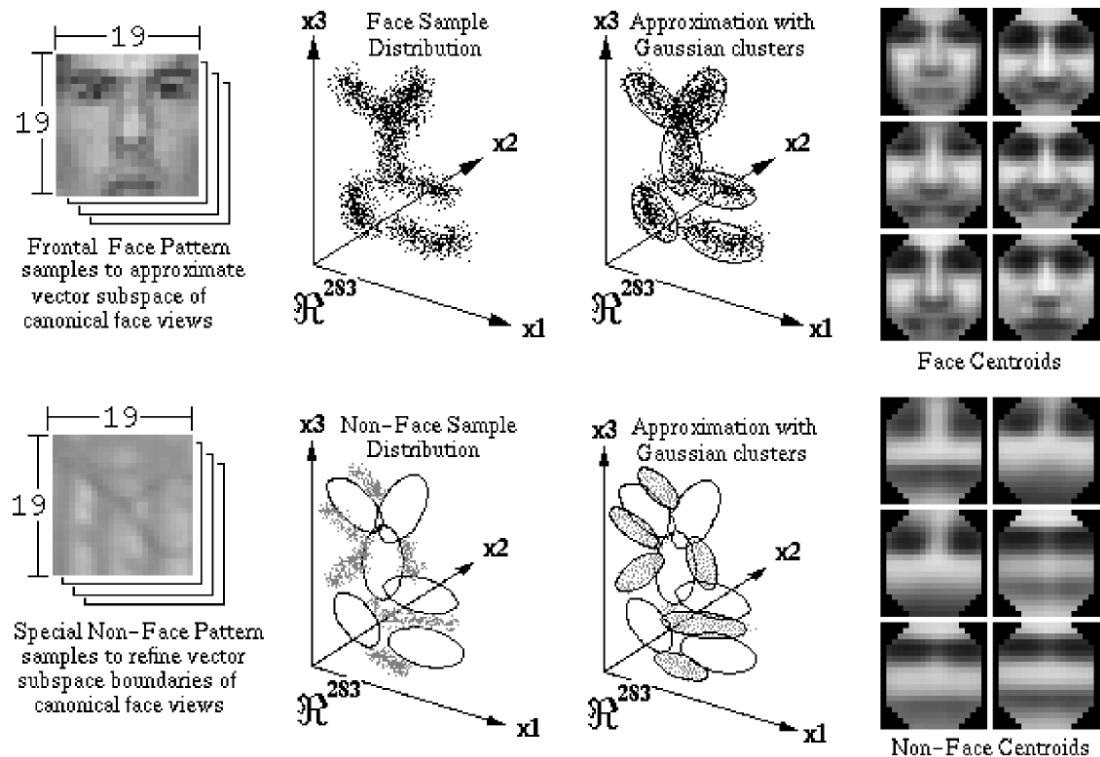


Figure 14.4 Learning a mixture of Gaussians model for face detection (Sung and Poggio 1998) © 1998 IEEE. The face and non-face images (19^2 -long vectors) are first clustered into six separate clusters (each) using k-means and then analyzed using PCA. The cluster centers are shown in the right-hand columns.

Clustering and PCA. Once the face and non-face patterns have been pre-processed, Sung and Poggio (1998) cluster each of these datasets into six separate clusters using k-means and then fit PCA subspaces to each of the resulting 12 clusters (Figure 14.4). At detection time, the DIFS and DFFS metrics first developed by Moghaddam and Pentland (1997) (see Figure 14.14 and (14.14)) are used to produce 24 Mahalanobis distance measurements (two per cluster). The resulting 24 measurements are input to a multi-layer perceptron (MLP), which is a neural network with alternating layers of weighted summations and sigmoidal non-linearities trained using the “backpropagation” algorithm (Rumelhart, Hinton, and Williams 1986).

Neural networks. Instead of first clustering the data and computing Mahalanobis distances to the cluster centers, Rowley, Baluja, and Kanade (1998a) apply a neural network (MLP) directly to the 20×20 pixel patches of gray-level intensities, using a variety of differently sized hand-crafted “receptive fields” to capture both large-scale and smaller scale structure (Figure 14.5). The resulting neural network directly outputs the likelihood of a face at the center

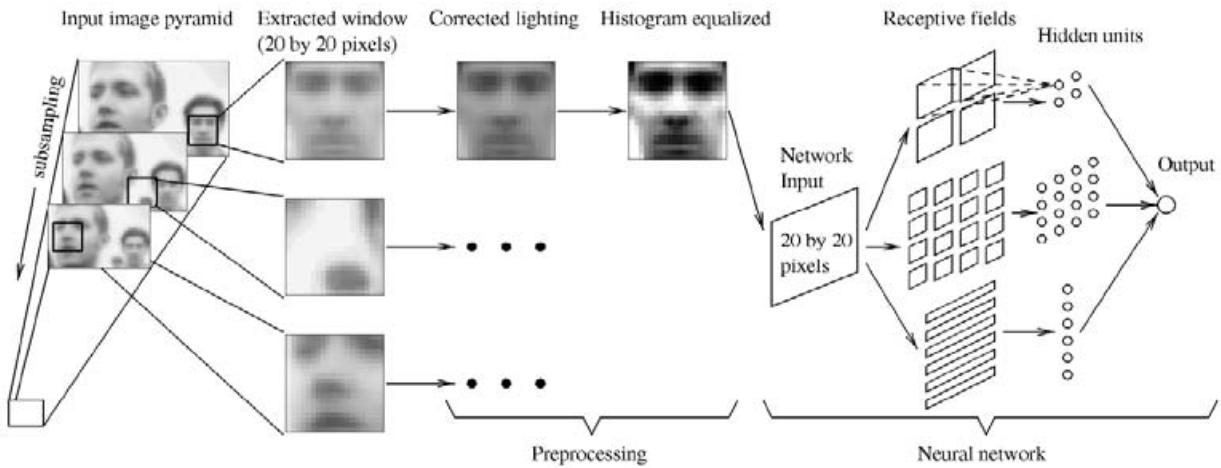


Figure 14.5 A neural network for face detection (Rowley, Baluja, and Kanade 1998a) © 1998 IEEE. Overlapping patches are extracted from different levels of a pyramid and then pre-processed as shown in Figure 14.3b. A three-layer neural network is then used to detect likely face locations.

of every overlapping patch in a multi-resolution pyramid. Since several overlapping patches (in both space and resolution) may fire near a face, an additional merging network is used to merge overlapping detections. The authors also experiment with training several networks and merging their outputs. Figure 14.2 shows a sample result from their face detector.

To make the detector run faster, a separate network operating on 30×30 patches is trained to detect both faces and faces shifted by ± 5 pixels. This network is evaluated at every 10th pixel in the image (horizontally and vertically) and the results of this “coarse” or “sloppy” detector are used to select regions on which to run the slower single-pixel overlap technique. To deal with in-plane rotations of faces, Rowley, Baluja, and Kanade (1998b) train a *router network* to estimate likely rotation angles from input patches and then apply the estimated rotation to each patch before running the result through their upright face detector.

Support vector machines. Instead of using a neural network to classify patches, Osuna, Freund, and Girosi (1997) use a *support vector machine* (SVM) (Hastie, Tibshirani, and Friedman 2001; Schölkopf and Smola 2002; Bishop 2006; Lampert 2008) to classify the same preprocessed patches as Sung and Poggio (1998). An SVM searches for a series of *maximum margin* separating planes in feature space between different classes (in this case, face and non-face patches). In those cases where linear classification boundaries are insufficient, the feature space can be lifted into higher-dimensional features using *kernels* (Hastie, Tibshirani, and Friedman 2001; Schölkopf and Smola 2002; Bishop 2006). SVMs have been used by other researchers for both face detection and face recognition (Heisele, Ho, Wu *et al.* 2003;

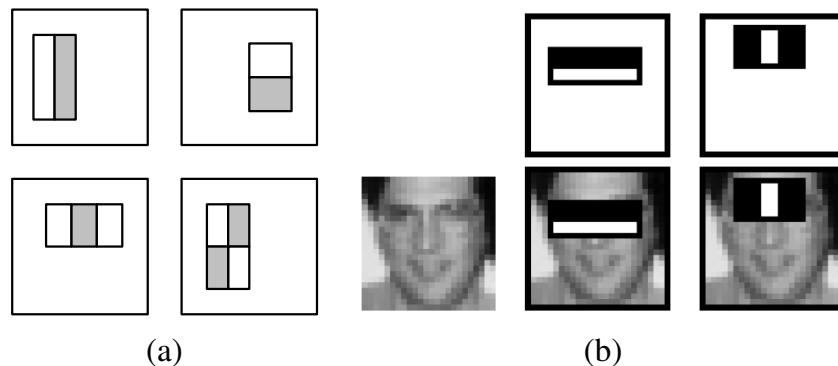


Figure 14.6 Simple features used in boosting-based face detector (Viola and Jones 2004) © 2004 Springer: (a) difference of rectangle feature composed of 2–4 different rectangles (pixels inside the white rectangles are subtracted from the gray ones); (b) the first and second features selected by AdaBoost. The first feature measures the differences in intensity between the eyes and the cheeks, the second one between the eyes and the bridge of the nose.

Heisele, Serre, and Poggio 2007) and are a widely used tool in object recognition in general.

Boosting. Of all the face detectors currently in use, the one introduced by Viola and Jones (2004) is probably the best known and most widely used. Their technique was the first to introduce the concept of *boosting* to the computer vision community, which involves training a series of increasingly discriminating simple classifiers and then blending their outputs (Hastie, Tibshirani, and Friedman 2001; Bishop 2006).

In more detail, boosting involves constructing a *classifier* $h(\mathbf{x})$ as a sum of simple *weak learners*,

$$h(\mathbf{x}) = \text{sign} \left[\sum_{j=0}^{m-1} \alpha_j h_j(\mathbf{x}) \right], \quad (14.1)$$

where each of the weak learners $h_j(\mathbf{x})$ is an extremely simple function of the input, and hence is not expected to contribute much (in isolation) to the classification performance.

In most variants of boosting, the weak learners are threshold functions,

$$h_j(\mathbf{x}) = a_j[f_j < \theta_j] + b_j[f_j \geq \theta_j] = \begin{cases} a_j & \text{if } f_j < \theta_j \\ b_j & \text{otherwise,} \end{cases} \quad (14.2)$$

which are also known as *decision stumps* (basically, the simplest possible version of *decision trees*). In most cases, it is also traditional (and simpler) to set a_j and b_j to ± 1 , i.e., $a_j = -s_j$, $b_j = +s_j$, so that only the feature f_j , the threshold value θ_j , and the polarity of the threshold $s_j \in \pm 1$ need to be selected.⁴

⁴Some variants, such as that of Viola and Jones (2004), use $(a_i, b_i) \in [0, 1]$ and adjust the learning algorithm

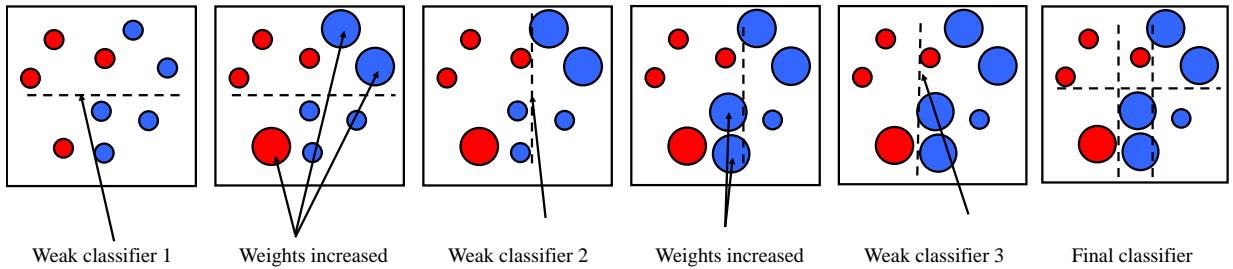


Figure 14.7 Schematic illustration of boosting, courtesy of Svetlana Lazebnik, after original illustrations from Paul Viola and David Lowe. After each weak classifier (decision stump or hyperplane) is selected, data points that are erroneously classified have their weights increased. The final classifier is a linear combination of the simple weak classifiers.

In many applications of boosting, the features are simply coordinate axes x_k , i.e., the boosting algorithm selects one of the input vector components as the best one to threshold. In Viola and Jones' face detector, the features are differences of rectangular regions in the input patch, as shown in Figure 14.6. The advantage of using these features is that, while they are more discriminating than single pixels, they are extremely fast to compute once a summed area table has been pre-computed, as described in Section 3.2.3 (3.31–3.32). Essentially, for the cost of an $O(N)$ pre-computation phase (where N is the number of pixels in the image), subsequent differences of rectangles can be computed in $4r$ additions or subtractions, where $r \in \{2, 3, 4\}$ is the number of rectangles in the feature.

The key to the success of boosting is the method for incrementally selecting the weak learners and for re-weighting the training examples after each stage (Figure 14.7). The AdaBoost (Adaptive Boosting) algorithm (Hastie, Tibshirani, and Friedman 2001; Bishop 2006) does this by re-weighting each sample as a function of whether it is correctly classified at each stage, and using the stage-wise average classification error to determine the final weightings α_j among the weak classifiers, as described in Algorithm 14.1. While the resulting classifier is extremely fast in practice, the training time can be quite slow (in the order of weeks), because of the large number of feature (difference of rectangle) hypotheses that need to be examined at each stage.

To further increase the speed of the detector, it is possible to create a *cascade* of classifiers, where each classifier uses a small number of tests (say, a two-term AdaBoost classifier) to reject a large fraction of non-faces while trying to pass through all potential face candidates (Fleuret and Geman 2001; Viola and Jones 2004). An even faster algorithm for performing cascade learning has recently been developed by Brubaker, Wu, Sun *et al.* (2008).

accordingly.

1. Input the positive and negative training examples along with their labels $\{(\mathbf{x}_i, y_i)\}$, where $y_i = 1$ for positive (face) examples and $y_i = -1$ for negative examples.
2. Initialize all the weights to $w_{i,1} \leftarrow \frac{1}{N}$, where N is the number of training examples. (Viola and Jones (2004) use a separate N_1 and N_2 for positive and negative examples.)
3. For each training stage $j = 1 \dots M$:
 - (a) Renormalize the weights so that they sum up to 1 (divide them by their sum).
 - (b) Select the best classifier $h_j(\mathbf{x}; f_j, \theta_j, s_j)$ by finding the one that minimizes the weighted classification error

$$e_j = \sum_{i=0}^{N-1} w_{i,j} e_{i,j}, \quad (14.3)$$

$$e_{i,j} = 1 - \delta(y_i, h_j(\mathbf{x}_i; f_j, \theta_j, s_j)). \quad (14.4)$$

For any given f_j function, the optimal values of (θ_j, s_j) can be found in linear time using a variant of weighted median computation (Exercise 14.2).

- (c) Compute the modified error rate β_j and classifier weight α_j ,

$$\beta_j = \frac{e_j}{1 - e_j} \quad \text{and} \quad \alpha_j = -\log \beta_j. \quad (14.5)$$

- (d) Update the weights according to the classification errors $e_{i,j}$

$$w_{i,j+1} \leftarrow w_{i,j} \beta_j^{1-e_{i,j}}, \quad (14.6)$$

i.e., downweight the training samples that were correctly classified in proportion to the overall classification error.

4. Set the final classifier to

$$h(\mathbf{x}) = \text{sign} \left[\sum_{j=0}^{m-1} \alpha_j h_j(\mathbf{x}) \right]. \quad (14.7)$$

Algorithm 14.1 The AdaBoost training algorithm, adapted from Hastie, Tibshirani, and Friedman (2001), Viola and Jones (2004), and Bishop (2006).

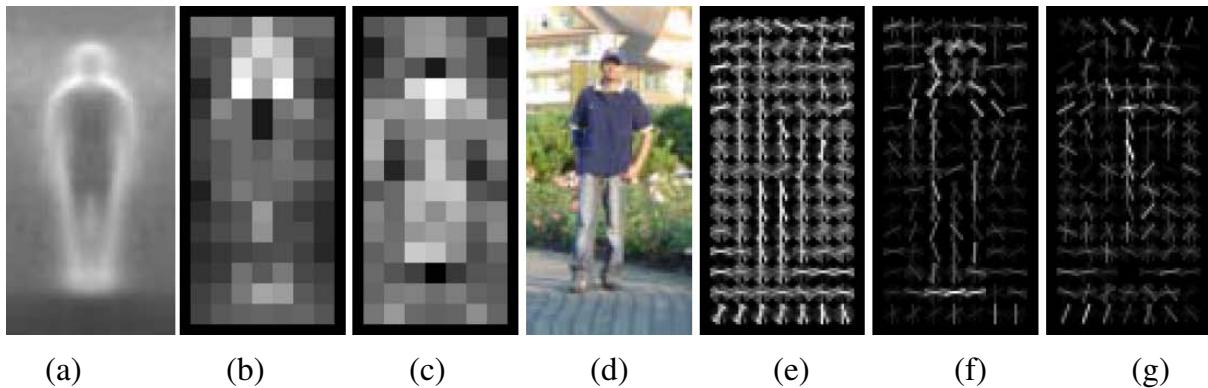


Figure 14.8 Pedestrian detection using histograms of oriented gradients (Dalal and Triggs 2005) © 2005 IEEE: (a) the average gradient image over the training examples; (b) each “pixel” shows the maximum positive SVM weight in the block centered on the pixel; (c) likewise, for the negative SVM weights; (d) a test image; (e) the computed R-HOG (rectangular histogram of gradients) descriptor; (f) the R-HOG descriptor weighted by the positive SVM weights; (g) the R-HOG descriptor weighted by the negative SVM weights.

14.1.2 Pedestrian detection

While a lot of the research on object detection has focused on faces, the detection of other objects, such as pedestrians and cars, has also received widespread attention (Gavrila and Philomin 1999; Gavrila 1999; Papageorgiou and Poggio 2000; Mohan, Papageorgiou, and Poggio 2001; Schneiderman and Kanade 2004). Some of these techniques maintain the same focus as face detection on speed and efficiency. Others, however, focus instead on accuracy, viewing detection as a more challenging variant of generic class recognition (Section 14.4) in which the locations and extents of objects are to be determined as accurately as possible. (See, for example, the PASCAL VOC detection challenge, <http://pascallin.ecs.soton.ac.uk/challenges/VOC/>.)

An example of a well-known pedestrian detector is the algorithm developed by Dalal and Triggs (2005), who use a set of overlapping *histogram of oriented gradients* (HOG) descriptors fed into a support vector machine (Figure 14.8). Each HOG has cells to accumulate magnitude-weighted votes for gradients at particular orientations, just as in the scale invariant feature transform (SIFT) developed by Lowe (2004), which we discussed in Section 4.1.2 and Figure 4.18. Unlike SIFT, however, which is only evaluated at interest point locations, HOGs are evaluated on a regular overlapping grid and their descriptor magnitudes are normalized using an even coarser grid; they are only computed at a single scale and a fixed orientation. In order to capture the subtle variations in orientation around a person’s outline, a large number of orientation bins is used and no smoothing is performed in the central difference gradient computation—see the work of Dalal and Triggs (2005) for more implementation details.

Figure 14.8d shows a sample input image, while Figure 14.8e shows the associated HOG descriptors.

Once the descriptors have been computed, a support vector machine (SVM) is trained on the resulting high-dimensional continuous descriptor vectors. Figures 14.8b–c show a diagram of the (most) positive and negative SVM weights in each block, while Figures 14.8f–g show the corresponding weighted HOG responses for the central input image. As you can see, there are a fair number of positive responses around the head, torso, and feet of the person, and relatively few negative responses (mainly around the middle and the neck of the sweater).

The fields of pedestrian and general object detection have continued to evolve rapidly over the last decade (Belongie, Malik, and Puzicha 2002; Mikolajczyk, Schmid, and Zisserman 2004; Leibe, Seemann, and Schiele 2005; Opelt, Pinz, and Zisserman 2006; Torralba 2007; Andriluka, Roth, and Schiele 2009, 2010; Dollàr, Belongie, and Perona 2010). Munder and Gavrila (2006) compare a number of pedestrian detectors and conclude that those based on local receptive fields and SVMs perform the best, with a boosting-based approach coming close. Maji, Berg, and Malik (2008) improve on the best of these results using non-overlapping multi-resolution HOG descriptors and a histogram intersection kernel SVM based on a spatial pyramid match kernel from Lazebnik, Schmid, and Ponce (2006).

When detectors for several different classes are being constructed simultaneously, Torralba, Murphy, and Freeman (2007) show that sharing features and weak learners between detectors yields better performance, both in terms of faster computation times and fewer training examples. To find the features and decision stumps that work best in a shared manner, they introduce a novel *joint boosting* algorithm that optimizes, at each stage, a summed expected exponential loss function using the “gentleboost” algorithm of Friedman, Hastie, and Tibshirani (2000).

In more recent work, Felzenszwalb, McAllester, and Ramanan (2008) extend the histogram of oriented gradients person detector to incorporate flexible parts models (Section 14.4.2). Each part is trained and detected on HOGs evaluated at two pyramid levels below the overall object model and the locations of the parts relative to the parent node (the overall bounding box) are also learned and used during recognition (Figure 14.9b). To compensate for inaccuracies or inconsistencies in the training example bounding boxes (dashed white lines in Figure 14.9c), the “true” location of the parent (blue) bounding box is considered a latent (hidden) variable and is inferred during both training and recognition. Since the locations of the parts are also latent, the system can be trained in a semi-supervised fashion, without needing part labels in the training data. An extension to this system (Felzenszwalb, Girshick, McAllester *et al.* 2010), which includes among its improvements a simple contextual model, was among the two best object detection systems in the 2008 Visual Object Classes detection challenge. Other recent improvements to part-based person detection and pose estimation in-

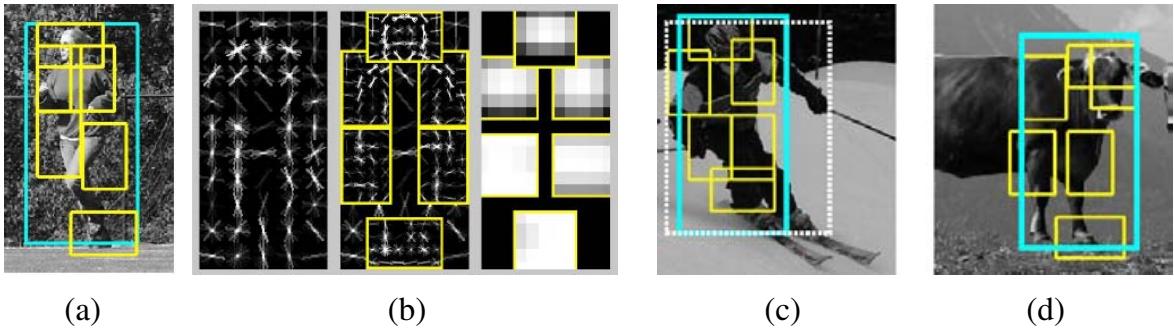


Figure 14.9 Part-based object detection (Felzenszwalb, McAllester, and Ramanan 2008) © 2008 IEEE: (a) An input photograph and its associated person (blue) and part (yellow) detection results. (b) The detection model is defined by a coarse template, several higher resolution part templates, and a spatial model for the location of each part. (c) True positive detection of a skier and (d) false positive detection of a cow (labeled as a person).

clude the work by Andriluka, Roth, and Schiele (2009) and Kumar, Zisserman, and H.S.Torr (2009).

An even more accurate estimate of a person's pose and location is presented by Rogez, Rihan, Ramalingam *et al.* (2008), who compute both the phase of a person in a walk cycle and the locations of individual joints, using random forests built on top of HOGs (Figure 14.11). Since their system produces full 3D pose information, it is closer in its application domain to 3D person trackers (Sidenbladh, Black, and Fleet 2000; Andriluka, Roth, and Schiele 2010), which we discussed in Section 12.6.4.

One final note on person and object detection. When video sequences are available, the additional information present in the optic flow and motion discontinuities can greatly aid in the detection task, as discussed by Efros, Berg, Mori *et al.* (2003), Viola, Jones, and Snow (2003), and Dalal, Triggs, and Schmid (2006).

14.2 Face recognition

Among the various recognition tasks that computers might be asked to perform, face recognition is the one where they have arguably had the most success.⁵ While computers cannot pick out suspects from thousands of people streaming in front of video cameras (even people cannot readily distinguish between similar people with whom they are not familiar (O'Toole, Jiang, Roark *et al.* 2006; O'Toole, Phillips, Jiang *et al.* 2009)), their ability to distinguish

⁵Instance recognition, i.e., the re-recognition of known objects such as locations or planar objects, is the other most successful application of general image recognition. In the general domain of *biometrics*, i.e., identity recognition, specialized images such as irises and fingerprints perform even better (Jain, Bolle, and Pankanti 1999; Pankanti, Bolle, and Jain 2000; Daugman 2004).

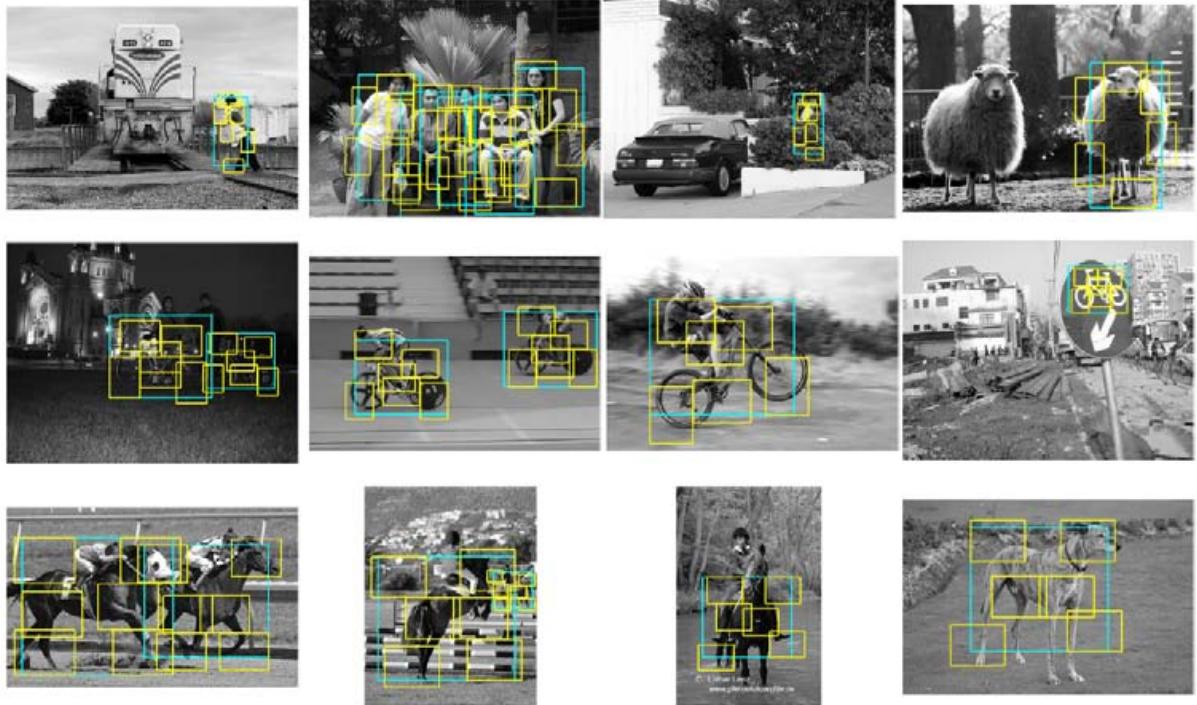


Figure 14.10 Part-based object detection results for people, bicycles, and horses (Felzenszwalb, McAllester, and Ramanan 2008) © 2008 IEEE. The first three columns show correct detections, while the rightmost column shows false positives.

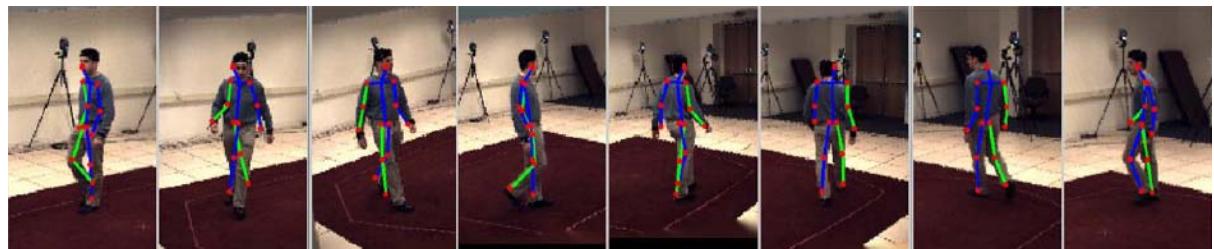


Figure 14.11 Pose detection using random forests (Rogez, Rihan, Ramalingam *et al.* 2008) © 2008 IEEE. The estimated pose (state of the kinematic model) is drawn over each input frame.



Figure 14.12 Humans can recognize low-resolution faces of familiar people (Sinha, Balas, Ostrovsky *et al.* 2006) © 2006 IEEE.

among a small number of family members and friends has found its way into consumer-level photo applications, such as Picasa and iPhoto. Face recognition can also be used in a variety of additional applications, including human–computer interaction (HCI), identity verification (Kirovski, Jovic, and Jancke 2004), desktop login, parental controls, and patient monitoring (Zhao, Chellappa, Phillips *et al.* 2003).

Today’s face recognizers work best when they are given full frontal images of faces under relatively uniform illumination conditions, although databases that include large amounts of pose and lighting variation have been collected (Phillips, Moon, Rizvi *et al.* 2000; Sim, Baker, and Bsat 2003; Gross, Shi, and Cohn 2005; Huang, Ramesh, Berg *et al.* 2007; Phillips, Scruggs, O’Toole *et al.* 2010). (See Table 14.1 in Section 14.6 for more details.)

Some of the earliest approaches to face recognition involved finding the locations of distinctive image features, such as the eyes, nose, and mouth, and measuring the distances between these feature locations (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991). More recent approaches rely on comparing gray-level images projected onto lower dimensional subspaces called *eigenfaces* (Section 14.2.1) and jointly modeling shape and appearance variations (while discounting pose variations) using *active appearance models* (Section 14.2.2).

Descriptions of additional face recognition techniques can be found in a number of surveys and books on this topic (Chellappa, Wilson, and Sirohey 1995; Zhao, Chellappa, Phillips *et al.* 2003; Li and Jain 2005) as well as the Face Recognition Web site.⁶ The survey on face recognition by humans by Sinha, Balas, Ostrovsky *et al.* (2006) is also well worth reading; it includes a number of surprising results, such as humans’ ability to recognize low-resolution images of familiar faces (Figure 14.12) and the importance of eyebrows in recognition.

⁶ <http://www.face-rec.org/>.

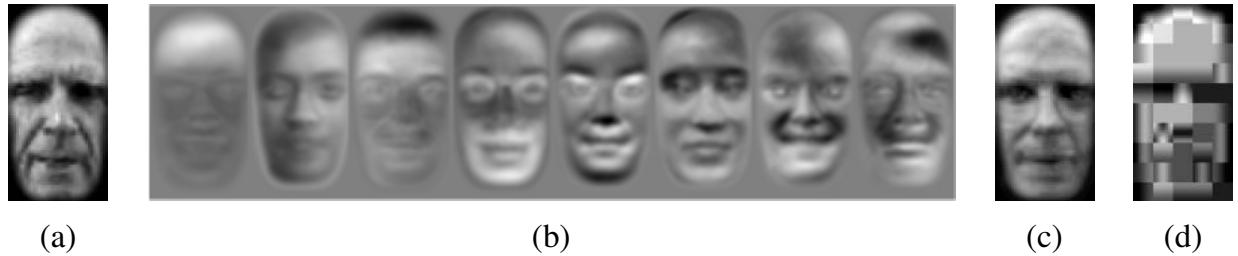


Figure 14.13 Face modeling and compression using eigenfaces (Moghaddam and Pentland 1997) © 1997 IEEE: (a) input image; (b) the first eight eigenfaces; (c) image reconstructed by projecting onto this basis and compressing the image to 85 bytes; (d) image reconstructed using JPEG (530 bytes).

14.2.1 Eigenfaces

Eigenfaces rely on the observation first made by Kirby and Sirovich (1990) that an arbitrary face image \mathbf{x} can be compressed and reconstructed by starting with a mean image \mathbf{m} (Figure 14.1b) and adding a small number of scaled signed images \mathbf{u}_i ,⁷

$$\tilde{\mathbf{x}} = \mathbf{m} + \sum_{i=0}^{M-1} a_i \mathbf{u}_i, \quad (14.8)$$

where the signed basis images (Figure 14.13b) can be derived from an ensemble of training images using *principal component analysis* (also known as *eigenvalue analysis* or the *Karhunen–Loève transform*). Turk and Pentland (1991a) recognized that the coefficients a_i in the eigenface expansion could themselves be used to construct a fast image matching algorithm.

In more detail, let us start with a collection of *training images* $\{\mathbf{x}_j\}$, from which we can compute the mean image \mathbf{m} and a *scatter* or *covariance* matrix

$$\mathbf{C} = \frac{1}{N} \sum_{j=0}^{N-1} (\mathbf{x}_j - \mathbf{m})(\mathbf{x}_j - \mathbf{m})^T. \quad (14.9)$$

We can apply the eigenvalue decomposition (A.6) to represent this matrix as

$$\mathbf{C} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T = \sum_{i=0}^{N-1} \lambda_i \mathbf{u}_i \mathbf{u}_i^T, \quad (14.10)$$

where the λ_i are the eigenvalues of \mathbf{C} and the \mathbf{u}_i are the *eigenvectors*. For general images, Kirby and Sirovich (1990) call these vectors *eigenpictures*; for faces, Turk and Pentland

⁷ In previous chapters, we used I to indicate images; in this chapter, we use the more abstract quantities \mathbf{x} and \mathbf{u} to indicate collections of pixels in an image turned into a vector.

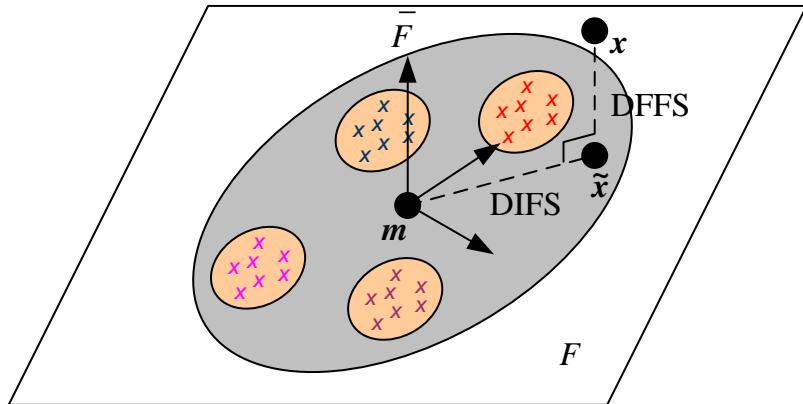


Figure 14.14 Projection onto the linear subspace spanned by the eigenface images (Moghaddam and Pentland 1997) © 1997 IEEE. The distance from face space (DFFS) is the orthogonal distance to the plane, while the distance in face space (DIFS) is the distance along the plane from the mean image. Both distances can be turned into Mahalanobis distances and given probabilistic interpretations.

(1991a) call them *eigenfaces* (Figure 14.13b).⁸

Two important properties of the eigenvalue decomposition are that the optimal (best approximation) coefficients a_i for any new image \mathbf{x} can be computed as

$$a_i = (\mathbf{x} - \mathbf{m}) \cdot \mathbf{u}_i, \quad (14.11)$$

and that, assuming the eigenvalues $\{\lambda_i\}$ are sorted in decreasing order, truncating the approximation given in (14.8) at any point M gives the best possible approximation (least error) between $\tilde{\mathbf{x}}$ and \mathbf{x} . Figure 14.13c shows the resulting approximation corresponding to Figure 14.13a and shows how much better it is at compressing a face image than JPEG.

Truncating the eigenface decomposition of a face image (14.8) after M components is equivalent to projecting the image onto a linear subspace F , which we can call the *face space* (Figure 14.14). Because the eigenvectors (eigenfaces) are orthogonal and of unit norm, the distance of a projected face $\tilde{\mathbf{x}}$ to the mean face \mathbf{m} can be written as

$$\text{DIFS} = \|\tilde{\mathbf{x}} - \mathbf{m}\| = \sqrt{\sum_{i=0}^{M-1} a_i^2}, \quad (14.12)$$

where DIFS stands for *distance in face space* (Moghaddam and Pentland 1997). The remaining distance between the original image \mathbf{x} and its projection onto face space $\tilde{\mathbf{x}}$, i.e., the

⁸ In actual practice, the full $P \times P$ scatter matrix (14.9) is never computed. Instead, a smaller $N \times N$ matrix consisting of the inner products between all the signed deviations $(\mathbf{x}_i - \mathbf{m})$ is accumulated instead. See Appendix A.1.2 (A.13–A.14) for details.

distance from face space (DFFS), can be computed directly in pixel space and represents the “faceness” of a particular image.⁹ It is also possible to measure the distance between two different faces in face space as

$$\text{DIFS}(\mathbf{x}, \mathbf{y}) = \|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\| = \sqrt{\sum_{i=0}^{M-1} (a_i - b_i)^2}, \quad (14.13)$$

where the $b_i = (\mathbf{y} - \mathbf{m}) \cdot \mathbf{u}_i$ are the eigenface coefficients corresponding to \mathbf{y} .

Computing such distances in Euclidean vector space, however, does not exploit the additional information that the eigenvalue decomposition of our covariance matrix (14.10) provides. If we interpret the covariance matrix \mathbf{C} as the covariance of a multi-variate Gaussian (Appendix B.1.1),¹⁰ we can turn the DIFS into a log likelihood by computing the *Mahalanobis distance*

$$\text{DIFS}' = \|\tilde{\mathbf{x}} - \mathbf{m}\|_{\mathbf{C}^{-1}} = \sqrt{\sum_{i=0}^{M-1} a_i^2 / \lambda_i^2}. \quad (14.14)$$

Instead of measuring the squared distance along each principal component in face space F , the Mahalanobis distance measures the *ratio* between the squared distance and the corresponding *variance* $\sigma_i^2 = \lambda_i$ and then sums these squared ratios (per-component log-likelihoods). An alternative way to implement this is to pre-scale each eigenvector by the inverse square root of its corresponding eigenvalue,

$$\hat{\mathbf{U}} = \mathbf{U}\Lambda^{-1/2}. \quad (14.15)$$

This *whitening* transformation then means that Euclidean distances in feature (face) space now correspond directly to log likelihoods (Moghaddam, Jebara, and Pentland 2000). (This same whitening approach can also be used in feature-based matching algorithms, as discussed in Section 4.1.3.)

If the distribution in eigenface space is very elongated, the Mahalanobis distance properly scales the components to come up with a sensible (probabilistic) distance from the mean. A similar analysis can be performed for computing a sensible difference from face space (DFFS) (Moghaddam and Pentland 1997) and the two terms can be combined to produce an estimate of the likelihood of being a true face, which can be useful in doing face detection (Section 14.1.1). More detailed explanations of probabilistic and Bayesian PCA can be found in textbooks on statistical learning (Hastie, Tibshirani, and Friedman 2001; Bishop 2006), which also discuss techniques for selecting the optimum number of components M to use in modeling a distribution.

⁹ This can be used to form a simple face detector, as mentioned in Section 14.1.1.

¹⁰ The ellipse shown in Figure 14.14 denotes an equi-probability contour of this multi-variate Gaussian.



Figure 14.15 Images from the Harvard database used by Belhumeur, Hespanha, and Kriegman (1997) © 1997 IEEE. Note the wide range of illumination variation, which can be more dramatic than inter-personal variations.

One of the biggest advantages of using eigenfaces is that they reduce the comparison of a new face image \mathbf{x} to a prototype (training) face image \mathbf{x}_k (one of the colored \mathbf{x} s in Figure 14.14) from a P -dimensional difference in pixel space to an M -dimensional difference in face space,

$$\|\mathbf{x} - \mathbf{x}_k\| = \|\mathbf{a} - \mathbf{a}_k\|, \quad (14.16)$$

where $\mathbf{a} = \mathbf{U}^T(\mathbf{x} - \mathbf{m})$ (14.11) involves computing a dot product between the signed difference-from-mean image $(\mathbf{x} - \mathbf{m})$ and each of the eigenfaces \mathbf{u}_i . Once again, however, this Euclidean distance ignores the fact that we have more information about face likelihoods available in the distribution of training images.

Consider the set of images of one person taken under a wide range of illuminations shown in Figure 14.15. As you can see, the *intrapersonal* variability within these images is much greater than the typical *extrapersonal* variability between any two people taken under the same illumination. Regular PCA analysis fails to distinguish between these two sources of variability and may, in fact, devote most of its principal components to modeling the intrapersonal variability.

If we are going to approximate faces by a linear subspace, it is more useful to have a space that *discriminates* between different classes (people) and is less sensitive to within-class variations (Belhumeur, Hespanha, and Kriegman 1997). Consider the three classes shown as different colors in Figure 14.16. As you can see, the distributions within a class (indicated by the tilted colored axes) are elongated and tilted with respect to the main face space PCA,

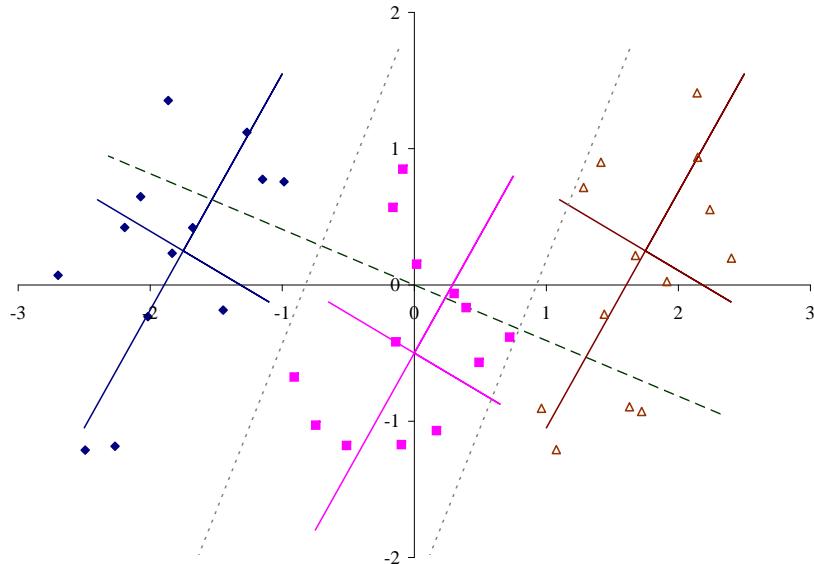


Figure 14.16 Simple example of Fisher linear discriminant analysis. The samples come from three different classes, shown in different colors along with their principal axes, which are scaled to $2\sigma_i$. (The intersections of the tilted axes are the class means \mathbf{m}_k .) The dashed line is the (dominant) Fisher linear discriminant direction and the dotted lines are the linear discriminants between the classes. Note how the discriminant direction is a blend between the principal directions of the between-class and within-class scatter matrices.

which is aligned with the black x and y axes. We can compute the total *within-class* scatter matrix as

$$\mathbf{S}_W = \sum_{k=0}^{K-1} \mathbf{S}_k = \sum_{k=0}^{K-1} \sum_{i \in C_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T, \quad (14.17)$$

where \mathbf{m}_k is the mean of class k and \mathbf{S}_k is its within-class scatter matrix.¹¹ Similarly, we can compute the *between-class* scatter as

$$\mathbf{S}_B = \sum_{k=0}^{K-1} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T, \quad (14.18)$$

where N_k are the number of exemplars in each class and \mathbf{m} is the overall mean. For the three distributions shown in Figure 14.16, we have

$$\mathbf{S}_W = 3N \begin{bmatrix} 0.246 & 0.183 \\ 0.183 & 0.457 \end{bmatrix} \quad \text{and} \quad \mathbf{S}_B = N \begin{bmatrix} 6.125 & 0 \\ 0 & 0.375 \end{bmatrix}, \quad (14.19)$$

¹¹ To be consistent with Belhumeur, Hespanha, and Kriegman (1997), we use \mathbf{S}_W and \mathbf{S}_B to denote the scatter matrices, even though we use \mathbf{C} elsewhere (14.9).

where $N = N_k = 13$ is the number of samples in each class.

To compute the most discriminating direction, *Fisher's linear discriminant* (FLD) (Belhumeur, Hespanha, and Kriegman 1997; Hastie, Tibshirani, and Friedman 2001; Bishop 2006), which is also known as *linear discriminant analysis* (LDA), selects the direction \mathbf{u} that results in the largest ratio between the projected between-class and within-class variations

$$\mathbf{u}^* = \arg \max_{\mathbf{u}} \frac{\mathbf{u}^T \mathbf{S}_B \mathbf{u}}{\mathbf{u}^T \mathbf{S}_W \mathbf{u}}, \quad (14.20)$$

which is equivalent to finding the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem

$$\mathbf{S}_B \mathbf{u} = \lambda \mathbf{S}_W \mathbf{u} \quad \text{or} \quad \lambda \mathbf{u} = \mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{u}. \quad (14.21)$$

For the problem shown in Figure 14.16,

$$\mathbf{S}_W^{-1} \mathbf{S}_B = \begin{bmatrix} 11.796 & -0.289 \\ -4.715 & 0.3889 \end{bmatrix} \quad \text{and} \quad \mathbf{u} = \begin{bmatrix} 0.926 \\ -0.379 \end{bmatrix} \quad (14.22)$$

As you can see, using this direction results in a better separation between the classes than using the dominant PCA direction, which is the horizontal axis. In their paper, Belhumeur, Hespanha, and Kriegman (1997) show that Fisherfaces significantly outperform the original eigenfaces algorithm, especially when faces have large amounts of illumination variation, as in Figure 14.15.

An alternative for modeling within-class (intrapersonal) and between-class (extrapersonal) variations is to model each distribution separately and then use Bayesian techniques to find the closest exemplar (Moghaddam, Jebara, and Pentland 2000). Instead of computing the mean for each class and then the within-class and between-class distributions, consider evaluating the difference images

$$\Delta_{ij} = \mathbf{x}_i - \mathbf{x}_j \quad (14.23)$$

between all pairs of training images $(\mathbf{x}_i, \mathbf{x}_j)$. The differences between pairs that are in the same class (the same person) are used to estimate the intrapersonal covariance matrix Σ_I , while differences between different people are used to estimate the extrapersonal covariance Σ_E .¹² The principal components (eigenfaces) corresponding to these two classes are shown in Figure 14.17.

At recognition time, we can compute the distance Δ_i between a new face \mathbf{x} and a stored training image \mathbf{x}_i and evaluate its intrapersonal likelihood as

$$p_I(\Delta_i) = p_{\mathcal{N}}(\Delta_i; \Sigma_I) = \frac{1}{|2\pi\Sigma_I|^{1/2}} \exp -\|\Delta_i\|_{\Sigma_I^{-1}}^2, \quad (14.24)$$

¹² Note that the difference distributions are zero mean because for every Δ_{ij} there corresponds a negative Δ_{ji} .

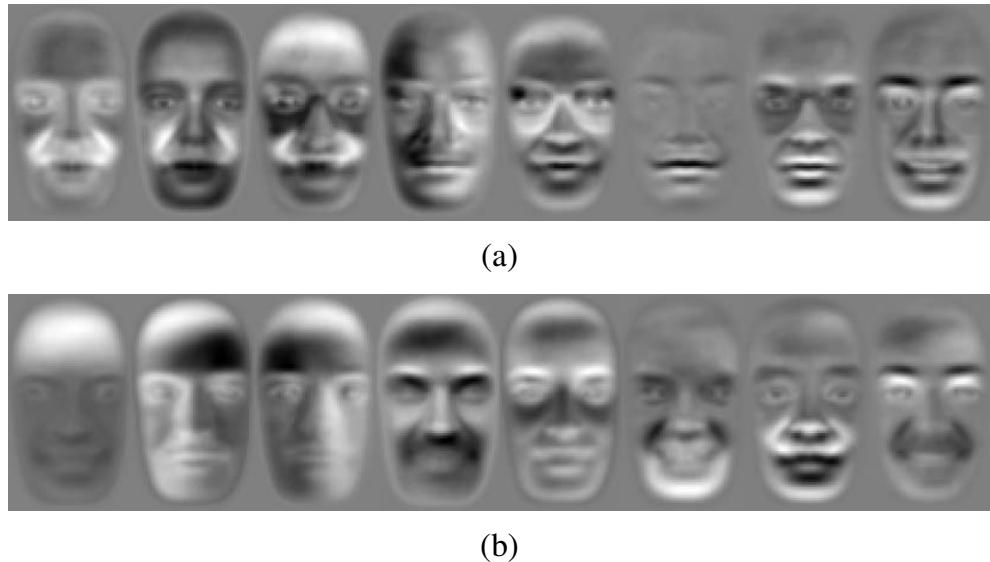


Figure 14.17 “Dual” eigenfaces (Moghaddam, Jebara, and Pentland 2000) © 2000 Elsevier: (a) intrapersonal and (b) extrapersonal.

where $p_{\mathcal{N}}$ is a normal (Gaussian) distribution with covariance Σ_I and

$$|2\pi\Sigma_I|^{1/2} = (2\pi)^{M/2} \prod_{j=1}^M \lambda_j^{1/2} \quad (14.25)$$

is its volume. The Mahalanobis distance

$$\|\Delta_i\|_{\Sigma_I^{-1}}^2 = \Delta_i^T \Sigma_I^{-1} \Delta_i = \|\mathbf{a}^I - \mathbf{a}_i^I\|^2 \quad (14.26)$$

can be computed more efficiently by first projecting the new image \mathbf{x} into the whitened intrapersonal face space (14.15)

$$\mathbf{a}^I = \hat{\mathbf{U}}^I \mathbf{x} \quad (14.27)$$

and then computing a Euclidean distance to the training image vector \mathbf{a}_i^I , which can be pre-computed offline. The extrapersonal likelihood $p_E(\Delta_i)$ can be computed in a similar fashion.

Once the intrapersonal and extrapersonal likelihoods have been computed, we can compute the Bayesian likelihood of a new image \mathbf{x} matching a training image \mathbf{x}_i as

$$p(\Delta_i) = \frac{p_I(\Delta_i)l_I}{p_I(\Delta_i)l_I + p_E(\Delta_i)l_E}, \quad (14.28)$$

where l_I and l_E are the prior probabilities of two images being in the same or in different classes (Moghaddam, Jebara, and Pentland 2000). A simpler approach, which does not require the evaluation of extrapersonal probabilities, is to simply choose the training image with

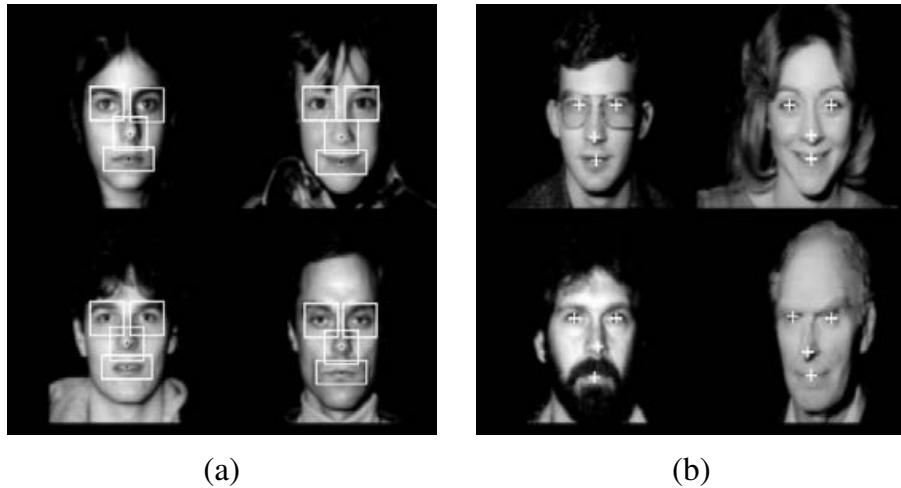


Figure 14.18 Modular eigenspace for face recognition (Moghaddam and Pentland 1997) © 1997 IEEE. (a) By detecting separate features in the faces (eyes, nose, mouth), separate eigenspaces can be estimated for each one. (b) The relative positions of each feature can be detected at recognition time, thus allowing for more flexibility in viewpoint and expression.

the highest likelihood $p_I(\Delta_i)$. In this case, nearest neighbor search techniques in the space spanned by the precomputed $\{\mathbf{a}_i^I\}$ vectors could be used to speed up finding the best match.¹³

Another way to improve the performance of eigenface-based approaches is to break up the image into separate regions such as the eyes, nose, and mouth (Figure 14.18) and to match each of these *modular eigenspaces* independently (Moghaddam and Pentland 1997; Heisele, Ho, Wu *et al.* 2003; Heisele, Serre, and Poggio 2007). The advantage of such a modular approach is that it can tolerate a wider range of viewpoints, because each part can move relative to the others. It also supports a larger variety of combinations, e.g., we can model one person as having a narrow nose and bushy eyebrows, without requiring the eigenfaces to span all possible combinations of nose, mouth, and eyebrows. (If you remember the cardboard children's books where you can select different top and bottom faces, or Mr. Potato Head, you get the idea.)

Another approach to dealing with large variability in appearance is to create *view-based* (view-specific) eigenspaces, as shown in Figure 14.19 (Moghaddam and Pentland 1997). We can think of these view-based eigenspaces as local descriptors that select different axes depending on which part of the face space you are in. Note that such approaches, however, potentially require large amounts of training data, i.e., pictures of every person in every possible pose or expression. This is in contrast to the shape and appearance models we study in

¹³ Note that while the covariance matrices Σ_I and Σ_E are computed by looking at differences between *all* pairs of images, the run-time evaluation selects the *nearest* image to determine the facial identity. Whether this is statistically correct is explored in Exercise 14.4.

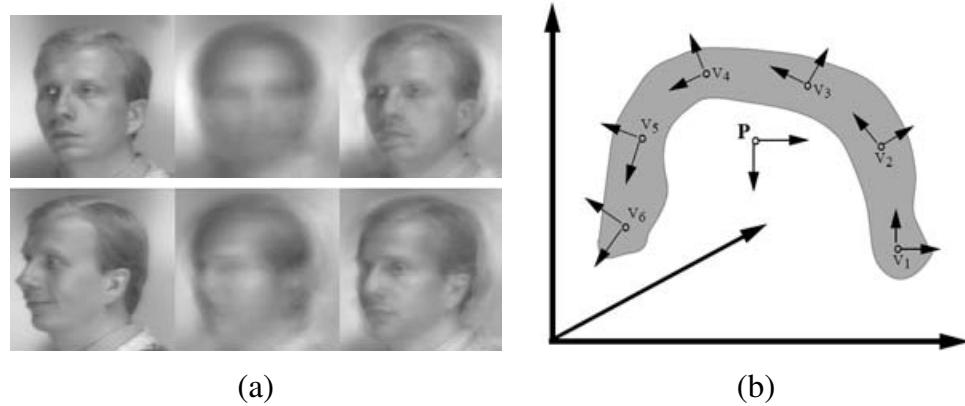


Figure 14.19 View-based eigenspace (Moghaddam and Pentland 1997) © 1997 IEEE. (a) Comparison between a regular (parametric) eigenspace reconstruction (middle column) and a view-based eigenspace reconstruction (right column) corresponding to the input image (left column). The top row is from a training image, the bottom row is from the test set. (b) A schematic representation of the two approaches, showing how each view computes its own local basis representation.

Section 14.2.2, which can learn deformations across all individuals.

It is also possible to generalize the bilinear factorization implicit in PCA and SVD approaches to multilinear (tensor) formulations that can model several interacting factors simultaneously (Vasilescu and Terzopoulos 2007). These ideas are related to currently active topics in machine learning such as *subspace learning* (Cai, He, Hu *et al.* 2007), *local distance functions* (Frome, Singer, Sha *et al.* 2007), and *metric learning* (Ramanan and Baker 2009). Learning approaches play an increasingly important role in face recognition, e.g., in the work of Sivic, Everingham, and Zisserman (2009) and Guillaumin, Verbeek, and Schmid (2009).

14.2.2 Active appearance and 3D shape models

The need to use modular or view-based eigenspaces for face recognition is symptomatic of a more general observation, i.e., that facial appearance and identifiability depend as much on *shape* as they do on color or texture (which is what eigenfaces capture). Furthermore, when dealing with 3D head rotations, the *pose* of a person's head should be discounted when performing recognition.

In fact, the earliest face recognition systems, such as those by Fischler and Elschlager (1973), Kanade (1977), and Yuille (1991), found distinctive feature points on facial images and performed recognition on the basis of their relative positions or distances. Newer techniques such as *local feature analysis* (Penev and Atick 1996) and *elastic bunch graph matching* (Wiskott, Fellous, Krüger *et al.* 1997) combine local filter responses (jets) at distinctive

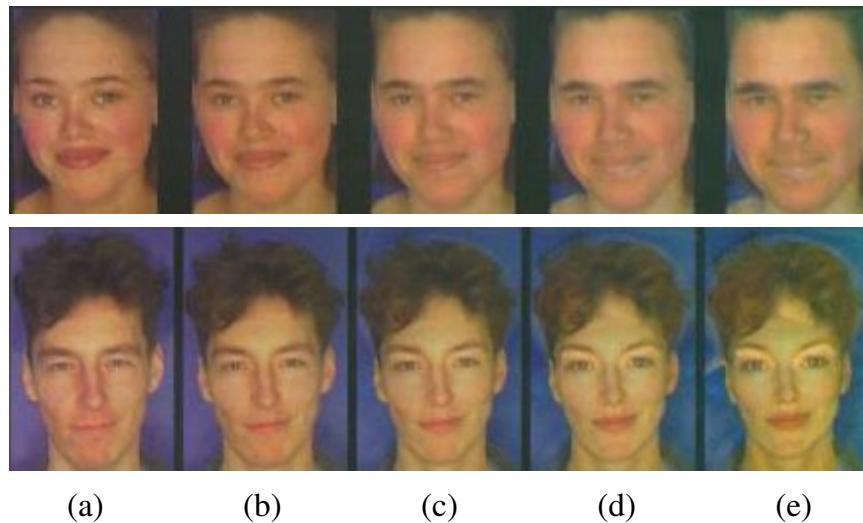


Figure 14.20 Manipulating facial appearance through shape and color (Rowland and Perrett 1995) © 1995 IEEE. By adding or subtracting gender-specific shape and color characteristics to (b) an input image, different amounts of gender variation can be induced. The amounts added (from the mean) are: (a) +50% (gender enhancement), (c) -50% (near “androgyny”), (d) -100% (gender switched), and (e) -150% (opposite gender attributes enhanced).

feature locations together with shape models to perform recognition.

A visually compelling example of why both shape and texture are important is the work of Rowland and Perrett (1995), who manually traced the contours of facial features and then used these contours to normalize (warp) each image to a canonical shape. After analyzing both the shape and color images for deviations from the mean, they were able to associate certain shape and color deformations with personal characteristics such as age and gender (Figure 14.20). Their work demonstrates that both shape and color have an important influence on the perception of such characteristics.

Around the same time, researchers in computer vision were beginning to use simultaneous shape deformations and texture interpolation to model the variability in facial appearance caused by identity or expression (Beymer 1996; Vetter and Poggio 1997), developing techniques such as Active Shape Models (Lanitis, Taylor, and Cootes 1997), 3D Morphable Models (Blanz and Vetter 1999), and Elastic Bunch Graph Matching (Wiskott, Fellous, Krüger *et al.* 1997).¹⁴

Of all these techniques, the *active appearance models* (AAMs) of Cootes, Edwards, and Taylor (2001) are among the most widely used for face recognition and tracking. Like other shape and texture models, an AAM models both the variation in the shape of an image s , which is normally encoded by the location of key feature points on the image (Figure 14.21b),

¹⁴ We have already seen the application of PCA to 3D head and face modeling and animation in Section 12.6.3.

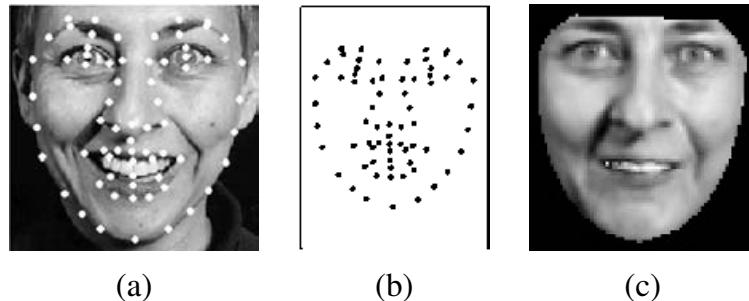


Figure 14.21 Active Appearance Models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE: (a) input image with registered feature points; (b) the feature points (shape vector s); (c) the shape-free appearance image (texture vector t).

as well as the variation in texture t , which is normalized to a canonical shape before being analyzed (Figure 14.21c).¹⁵

Both shape and texture are represented as deviations from a mean shape \bar{s} and texture \bar{t} ,

$$s = \bar{s} + \mathbf{U}_s a \quad (14.29)$$

$$t = \bar{t} + \mathbf{U}_t a, \quad (14.30)$$

where the eigenvectors in \mathbf{U}_s and \mathbf{U}_t have been pre-scaled (whitened) so that unit vectors in a represent one standard deviation of variation observed in the training data. In addition to these principal deformations, the shape parameters are transformed by a global similarity to match the location, size, and orientation of a given face. Similarly, the texture image contains a scale and offset to best match novel illumination conditions.

As you can see, the same appearance parameters a in (14.29–14.30) simultaneously control both the shape and texture deformations from the mean, which makes sense if we believe them to be correlated. Figure 14.22 shows how moving three standard deviations along each of the first four principal directions ends up changing several correlated factors in a person’s appearance, including expression, gender, age, and identity.

In order to fit an active appearance model to a novel image, Cootes, Edwards, and Taylor (2001) pre-compute a set of “difference decomposition” images, using an approach related to other fast techniques for incremental tracking, such as those we discussed in Sections 4.1.4, 8.1.3, and 8.2 (Gleicher 1997; Hager and Belhumeur 1998), which often *learn* a discriminative mapping between matching errors and incremental displacements (Avidan 2001; Jurie and Dhome 2002; Liu, Chen, and Kumar 2003; Sclaroff and Isidoro 2003; Romdhani and Vetter 2003; Williams, Blake, and Cipolla 2003).

¹⁵ When only the shape variation is being captured, such models are called *active shape models* (ASMs) (Cootes, Cooper, Taylor *et al.* 1995; Davies, Twining, and Taylor 2008). These were already discussed in Section 5.1.1 (5.13–5.17).

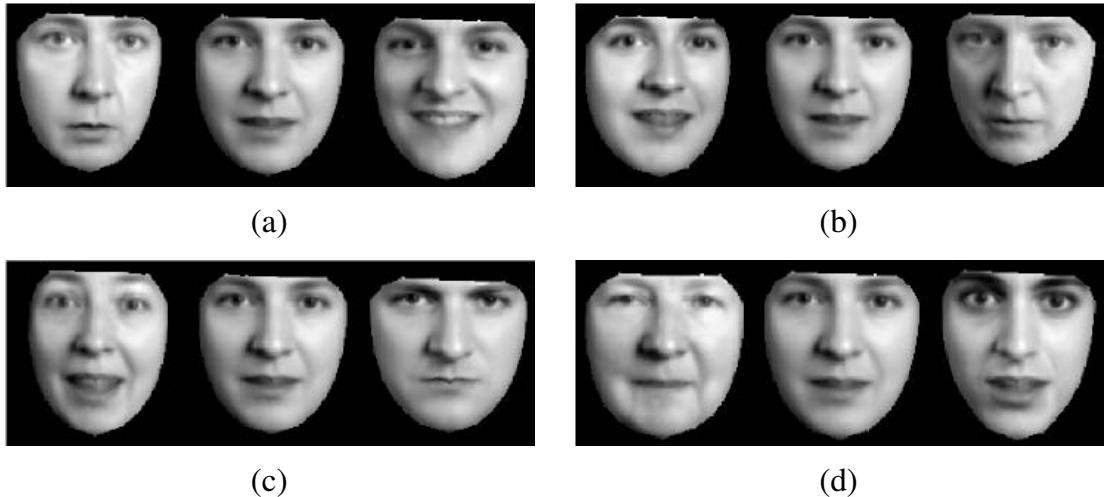


Figure 14.22 Principal modes of variation in active appearance models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE. The four images show the effects of simultaneously changing the first four modes of variation in both shape and texture by $\pm\sigma$ from the mean. You can clearly see how the shape of the face and the shading are simultaneously affected.

In more detail, Cootes, Edwards, and Taylor (2001) compute the derivatives of a set of training images with respect to each of the parameters in \mathbf{a} using finite differences and then compute a set of *displacement weight* images

$$\mathbf{W} = \left[\frac{\partial \mathbf{x}^T}{\partial \mathbf{a}} \quad \frac{\partial \mathbf{x}}{\partial \mathbf{a}} \right]^{-1} \frac{\partial \mathbf{x}^T}{\partial \mathbf{a}}, \quad (14.31)$$

which can be multiplied by the current error residual to produce an update step in the parameters, $\delta \mathbf{a} = -\mathbf{W} \mathbf{r}$. Matthews and Baker (2004) use their *inverse compositional method*, which they first developed for parametric optical flow (8.64–8.65), to further speed up active appearance model fitting and tracking. Examples of AAMs being fitted to two input images are shown in Figure 14.23.

Although active appearance models are primarily designed to accurately capture the variability in appearance and deformation that are characteristic of faces, they can be adapted to face recognition by computing an identity subspace that separates variation in identity from other sources of variability such as lighting, pose, and expression (Costen, Cootes, Edwards *et al.* 1999). The basic idea, which is modeled after similar work in eigenfaces (Belhumeur, Hespanha, and Kriegman 1997; Moghaddam, Jebara, and Pentland 2000), is to compute separate statistics for intrapersonal and extrapersonal variation and then find discriminating directions in these subspaces. While AAMs have sometimes been used directly for recognition (Blanz and Vetter 2003), their main use in the context of recognition is to align faces into a canonical pose (Liang, Xiao, Wen *et al.* 2008) so that more traditional methods of face



Figure 14.23 Multiresolution model fitting (search) in active appearance models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE. The columns show the initial model, the results after 3, 8, and 11 iterations, and the final convergence. The rightmost column shows the input image.

recognition (Penev and Atick 1996; Wiskott, Fellous, Krüger *et al.* 1997; Ahonen, Hadid, and Pietikäinen 2006; Zhao and Pietikäinen 2007; Cao, Yin, Tang *et al.* 2010) can be used. AAMs (or, actually, their simpler version, Active Shape Models (ASMs)) can also be used to align face images to perform automated morphing (Zanella and Fuentes 2004).

Active appearance models continue to be an active research area, with enhancements to deal with illumination and viewpoint variation (Gross, Baker, Matthews *et al.* 2005) as well as occlusions (Gross, Matthews, and Baker 2006). One of the most significant extensions is to construct 3D models of shape (Matthews, Xiao, and Baker 2007), which are much better at capturing and explaining the full variability of facial appearance across wide changes in pose.

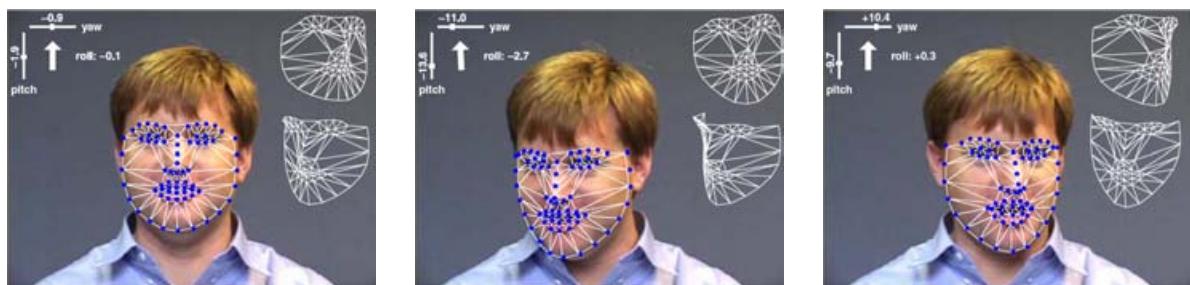


Figure 14.24 Head tracking with 3D AAMs (Matthews, Xiao, and Baker 2007) © 2007 Springer. Each image shows a video frame along with the estimate yaw, pitch, and roll parameters and the fitted 3D deformable mesh.

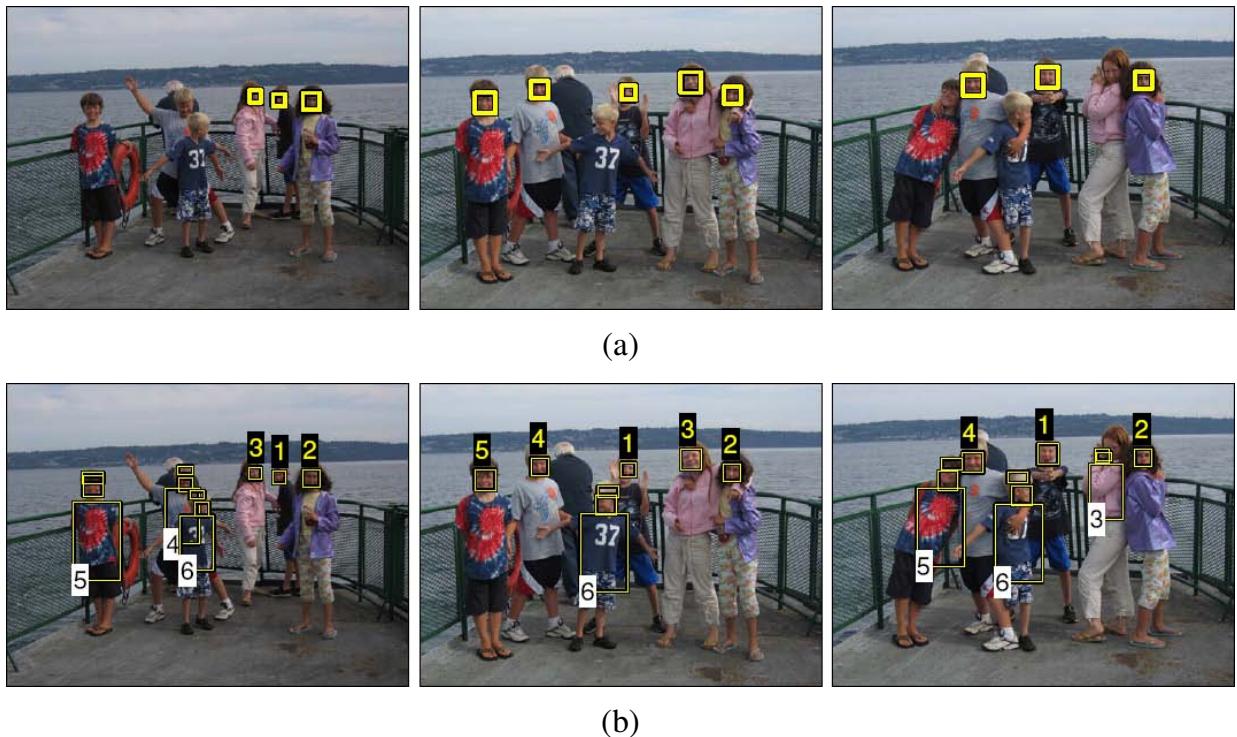


Figure 14.25 Person detection and re-recognition using a combined face, hair, and torso model (Sivic, Zitnick, and Szeliski 2006) © 2006 Springer. (a) Using face detection alone, several of the heads are missed. (b) The combined face and clothing model successfully re-finds all the people.

Such models can be constructed either from monocular video sequences (Matthews, Xiao, and Baker 2007), as shown in Figure 14.24, or from multi-view video sequences (Ramnath, Koterba, Xiao *et al.* 2008), which provide even greater reliability and accuracy in reconstruction and tracking. (For a recent review of progress in head pose estimation, please see the survey paper by Murphy-Chutorian and Trivedi (2009).)

14.2.3 Application: Personal photo collections

In addition to digital cameras automatically finding faces to aid in auto-focusing and video cameras finding faces in video conferencing to center on the speaker (either mechanically or digitally), face detection has found its way into most consumer-level photo organization packages, such as iPhoto, Picasa, and Windows Live Photo Gallery. Finding faces and allowing users to tag them makes it easier to find photos of selected people at a later date or to automatically share them with friends. In fact, the ability to tag friends in photos is one of the more popular features on Facebook.

Sometimes, however, faces can be hard to find and recognize, especially if they are small,

vision systems, where moving cameras are used to monitor both stationary and moving objects.

“Tis Cinna; I do know him by his gait”

(William Shakespeare, 1599)

22.1 INTRODUCTION

Visual surveillance is a long-standing area of computer vision, and one of its main early uses was to obtain information on military activities—whether from high flying aircraft or from satellites. However, with the advent of ever cheaper video cameras, it subsequently became widely used for monitoring road traffic, and most recently it has become ubiquitous for monitoring pedestrians. In fact, its application has actually become much wider than this, the aim being to locate criminals or people acting suspiciously—for example, wandering around car-parks with the potential purpose of theft. However, by far the majority of visual surveillance cameras are connected to video recorders and gather miles of videotape, most of which will never be looked at—though, following criminal or other activity, some hours of videotape may be scanned for relevant events. Further cameras will be attached to closed circuit television monitors where human operators may be able to extract some fraction of the events displayed, though human attentiveness and reliability when overseeing a dozen or so screens will not be high. Clearly, it would be far better if video cameras could be connected to automatic computer vision monitoring systems, which would call human operators’ attention to potential hazards or misdemeanors of various types. Even if this were not carried out in real time in specific applications, it would be useful if it could be achieved at high speed with selected videotapes: this could save huge amounts of police time in locating and identifying perpetrators of crime.

Surveillance can cover other useful activities, including riot control, monitoring of crowds on football pitches, checking for overcrowding on underground stations, and generally helping with safety as well as crime. To some extent, human privacy must suffer when surveillance is called into play, and there is clearly a tradeoff between privacy and security: suffice it to say here that many would be happier to have increased levels of security, a small loss of privacy being a welcome price to pay to achieve it.

In fact, there are many difficulties to be solved before the “people tracking” aspects of surveillance are fully solved. First, in comparison to cars, people are articulated objects that change shape markedly as they move: that their motion is often largely periodic can help visual analysis, though the irregularities in human motion may be considerable—especially if obstacles have to be avoided. Second, human motions are partly self-occluding, one leg regularly disappearing behind another, while arms can similarly disappear from view. Third, people vary in size and

apparent shape, having a variety of clothes that can disguise their outlines. Fourth, when pedestrians are observed on a pavement, or on the underground, there is some possibility of losing track when one person passes behind another, as the two outlines tend to coalesce before re-emerging from the combined object shape.

It could be said that all these problems have been solved. However, many of the algorithms that have been applied to these tasks have limited intelligence: indeed, some employ rather simplified algorithms, as the need to operate continuously in real time generally overrides the need for absolute accuracy. In any case, given the visual data that the computer actually receives, it is doubtful whether a human operator could always guarantee making correct interpretations: for example, there are occasions when humans turn around in their tracks because they have forgotten something, and this could cause confusion when trying to track every person in a complex scene. Further complexities can be caused by varying illumination, fixed shadows from buildings, moving shadows from clouds or vehicles, and so on.

In the following sections we cover two main areas of surveillance—those in which people or pedestrians are the prime targets, and those in which vehicles are the prime targets. Of course, there are many transport scenarios where both would be observed by the same systems. In addition, similar techniques and considerations would apply in both cases. The next section, on the geometry underlying camera positioning, talks mainly about pedestrians: this is done for definiteness, though most of the considerations apply equally when vehicles are the prime targets, as happens on motorways, for example.

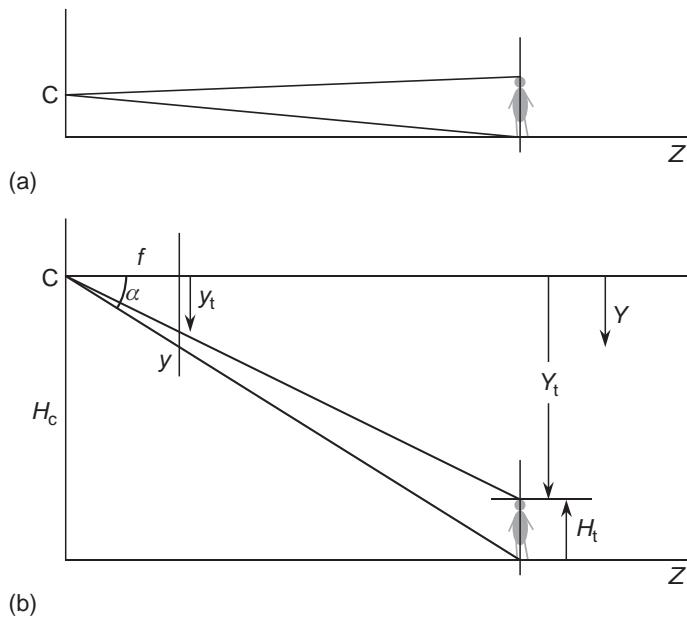
22.2 SURVEILLANCE—THE BASIC GEOMETRY

Perhaps the most obvious way of monitoring pedestrians is indicated in Fig. 22.1(a). As we have seen in Chapter 15, this leads to the following relations between real-world (X, Y, Z) and image coordinates (x, y):

$$x = \frac{fX}{Z} \quad (22.1)$$

$$y = \frac{fY}{Z} \quad (22.2)$$

Here, Z is the (horizontal) depth in the scene, X represents lateral position, Y represents vertical position (downward from the camera axis), and f is the focal length of the camera lens. This method of observation is useful in providing undistorted profiles of pedestrians from which they may be recognized. However, it provides virtually no information about depth in the scene beyond what can be deduced from knowledge of the pedestrian's size; and as size may be one of the key parameters to be determined by the vision system, this is an unsatisfactory situation. Note also that this view of the scene is subject to gross occlusion of one pedestrian by another.


FIGURE 22.1

3-D monitoring: camera axis horizontal. (a) Camera axis mounted at eye-level. (b) Camera mounted higher up to obtain a less restricted view.

To overcome these problems, an overhead view would be better. However, it is difficult to obtain views from directly overhead; in any case, any one view would give a highly restricted range, and again pedestrian height could not be measured. An alternative approach is to place the camera in Fig. 22.1(a) higher up, as shown in Fig. 22.1(b), so that the positions of the feet of any pedestrian on the ground plane can be seen: this makes it possible to obtain a reasonable estimate of depth in the scene. In fact, if the camera is at a height H_c above the ground plane, Eq. (22.2) gives the depth Z as:

$$Z = \frac{fH_c}{y} \quad (22.3)$$

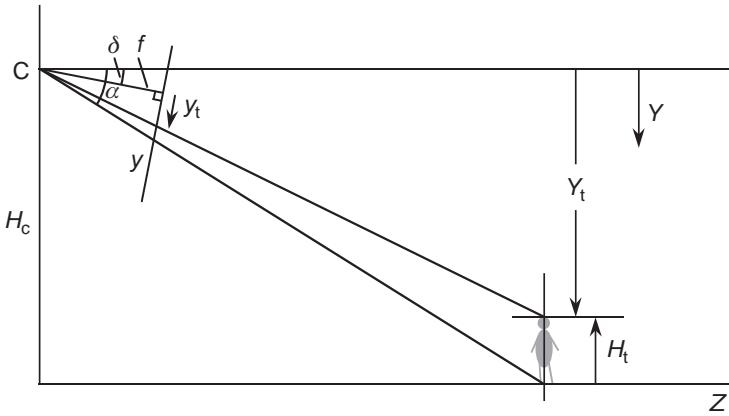
while the modified value of y at the top of the pedestrian is given by:

$$y_t = \frac{fY_t}{Z} = \frac{yY_t}{H_c} \quad (22.4)$$

The height of the pedestrian H_t can now be estimated from the following equation:

$$H_t = H_c - Y_t = H_c(1 - y_t/y) \quad (22.5)$$

Note that to achieve this, H_c must be known from prior on-site measurements, or alternatively by camera calibration using test objects.

**FIGURE 22.2**

3-D monitoring: camera tilted downwards. δ is the angle of declination of the camera optical axis.

In practice, it is better to modify the above scheme by tilting the optical axis of the camera slightly downward (see Fig. 22.2), as this allows the range of observation to be increased, and particularly for nearby pedestrians to be kept in view. However, the geometry of the situation becomes somewhat more complicated, leading to the following basic formulae:

$$\tan \alpha = \frac{H_c}{Z} \quad (22.6)$$

$$\tan (\alpha - \delta) = \frac{y}{f} \quad (22.7)$$

where δ is the angle of declination of the camera. Substituting for $\tan(\alpha-\delta)$ using the formula:

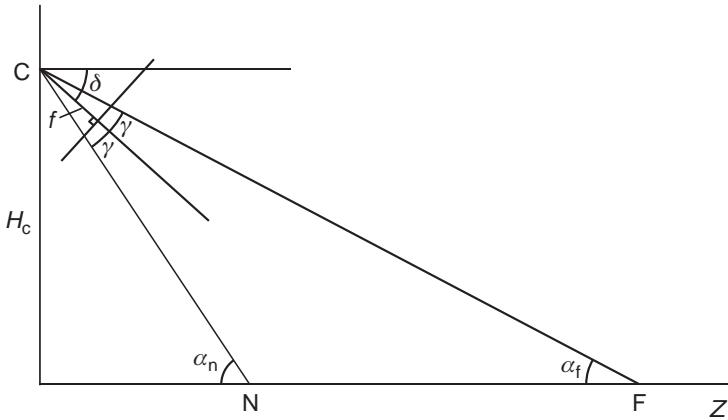
$$\tan (\alpha - \delta) = \frac{(\tan \alpha - \tan \delta)}{(1 + \tan \alpha \tan \delta)} \quad (22.8)$$

and using the above equations to eliminate α , we obtain the following formula for Z in terms of y :

$$Z = H_c \frac{(f - y \tan \delta)}{(y + f \tan \delta)} \quad (22.9)$$

So far we have not allowed for the heights of any objects, but have only considered points on the ground plane. To estimate the heights of pedestrians, we need to bring in the additional equation:

$$Z = Y_t \frac{(f - y_t \tan \delta)}{(y_t + f \tan \delta)} \quad (22.10)$$


FIGURE 22.3

Geometry for considering optimum camera tilt. δ is the angle of declination of the camera optical axis. 2γ is the overall vertical field of view of the camera.

which is simply derived by substituting Y_t for H_c and y_t for y in Eq. (22.9). Eliminating Z between these two equations now allows us to find Y_t :

$$Y_t = H_c \frac{(f - y \tan \delta)(y_t + f \tan \delta)}{(y + f \tan \delta)(f - y_t \tan \delta)} \quad (22.11)$$

thereby permitting $H_t = H_c - Y_t$ to be calculated in this case too.

Next, we consider the optimum value for the angle of declination δ of the camera optical axis. We assume that the viewing range of the camera has to vary from a near point given by Z_n to a far point given by Z_f , corresponding to respective values of α , α_n , and α_f (Fig. 22.3). We also assume that the overall vertical field of view of the camera is 2γ . This immediately results in the following formulae:

$$\frac{H_c}{Z_n} = \tan \alpha_n = \tan(\delta + \gamma) \quad (22.12)$$

$$\frac{H_c}{Z_f} = \tan \alpha_f = \tan(\delta - \gamma) \quad (22.13)$$

Taking the ratio between these equations now shows that:

$$\eta = \frac{Z_n}{Z_f} = \frac{\tan(\delta - \gamma)}{\tan(\delta + \gamma)} \quad (22.14)$$

so specifying either Z_n or Z_f immediately gives the alternate value. In the case that Z_f is taken to be infinity, Eq. (22.13) shows that δ has to be equal to γ , in which case Eq. (22.12) leads to the relation $Z_n = H_c \cot 2\gamma$. Note that $\delta = \gamma = 45^\circ$ is a limiting case that covers all points on the ground plane, i.e. $Z_n = 0$ and

$Z_f = \infty$. For smaller values of γ , values of Z_n and Z_f are determined by δ , e.g. for $\gamma = 30^\circ$, the optimum value of η (namely, zero) occurs both at $\delta = 30^\circ$ and at $\delta = 60^\circ$, and the worst case ($\eta \approx 0.072$) occurs at $\delta = 45^\circ$.

Finally, it is instructive to consider the minimum separation Z_s that is needed between pedestrians if they are not to occlude each other at all. Equating $\tan \alpha$ to both H_t/Z_s and H_c/Z (see Eq. (22.6)), we find:

$$Z_s = \frac{H_t Z}{H_c} \quad (22.15)$$

As might have been expected, this varies inversely with camera height; but note that it is also proportional to Z .

Overall, we have seen that placing the camera high up permits both depth and height to be estimated and the incidence of occlusion to be considerably reduced. In addition, tilting the camera downward permits the maximum range to be achieved. Importantly, two cameras placed at the far ends of a courtyard should be able to cover it quite well. Pedestrians can be identified as having a particular position on the ground plane, though they could then be recognized pictorially from knowledge of their size, shape, and coloring. The formulae that are involved reflect all the complications of perspective projection, and some are quite complex. Note that even in the simple case of Fig. 22.1(b), the inverse relation between y and Z is highly nonlinear (see Eq. (22.3)), and equal intervals in the Z direction by no means correspond to equal vertical intervals in the image plane: see Section 17.8 for further theory underpinning this point.

22.3 FOREGROUND–BACKGROUND SEPARATION

One of the first problems of surveillance is to locate the targets that are to be placed under observation. In principle, we could follow all the recognition methods of earlier chapters (see also Chapter 24), and just proceed to recognize the targets individually. However, there are two reasons why we should approach this differently. First, cars moving along a road, or pedestrians in a precinct, are highly variegated, unlike the situation for products appearing on a product line. Second, there is usually a significant real-time problem, especially when vehicles are moving at up to 100 mph on a highway, and cameras typically deliver 30 frames per second under highly variable conditions. Thus, it pays to capitalize on the motion of the targets and perform motion-based segmentation.

In these circumstances, it is natural to think of frame differencing and optical flow. Indeed, frame differencing has been applied to this task, but it is prone to noise problems and consequent unreliability. In any case, when applied between adjacent frames, it locates only limited sections of target outlines—in accordance with the $-\nabla I \cdot v$ formula of Chapter 19. The simplest way out of this difficulty is that of background modeling.

22.3.1 Background Modeling

The idea of background modeling is to create an idealized background image that can be subtracted from any frame to yield the target or foreground image. To achieve this, the simplest strategy is to take a frame when there are known to be no targets present and use that as the background model. In addition, to eliminate noise, it is useful to average a number of frames prior to making observations of targets. The problems with this strategy are: (a) How to know when there are no targets present, so that frames represent true background? (b) How to cope with the usual outdoor situation of illumination that varies with the weather and the time of day?

To solve the latter problem, only the most recent frames can reasonably be used, and if this path is followed, it is difficult to tackle the former problem (in any case, on highways with a lot of traffic, or precincts with a continuous mêlée of people, there may seldom be a chance of obtaining a clear background frame). One compromise solution is to take an average of many background frames over the most recent period Δt , whether or not targets are present. If targets are reasonably rare, most of the frames will be clear and a good approximation to an ideal background model will be built: of course, any targets will not be eliminated so much as averaged in, and the result will sometimes be visible “tails” in the model. To optimize the model, Δt can be increased, thereby minimizing problem (a), or decreased, thereby minimizing problem (b). Clearly, there is a tradeoff between the two difficulties: while it can be adjusted to suit the time of day and prevailing weather and illumination levels, this approach is limited.

Part of the problem is due to the “averaging in” mentioned above, and this can be partially eliminated by using a temporal median filter. Note that this means applying a median filter to the I (intensity and color) values of each pixel, over the sequence of frames arising during the most recent period Δt . This is a computation intensive process, but is considerably better than taking a raw average, as mentioned above. It is effective to take the median because it eliminates outliers, but ultimately it will still lead to biased estimates. In particular, if we suppose that vehicles are on the whole darker than the road, then the temporal median will also tend to end up darker than the road. To overcome this problem, a temporal mode filter can be used, and hopefully, the intensity distribution will have separate modes—one from the road and one from the vehicles, so the former can be used, and could probably be identified even if it became a minor mode when there were a lot of vehicles. However, there is no guarantee that there would only be one mode for vehicles, or even that any such modes would be clearly separated from the one corresponding to the road, and bias would again be the most likely result. [Figures 22.4–22.6](#) illustrate some of the problems.

In fact, there are significant further problems with background modeling. In many situations the background objects are themselves subject to motion. In particular, shadows will move with time and their crispness will change with the weather; while leaves, branches of trees, and flags will flutter and sway in the wind, with highly variable frequencies; even the camera may sway, especially if it is mounted on a pole,

**FIGURE 22.4**

Background subtraction using a temporal median filter. The lines of black graphics dots demarcate the relevant road region: almost all of the fluttering vegetation lies outside this region (it is indicated by fainter boundaries than for the foreground objects: see (b)). Note the plethora of stationary shadows that are completely eliminated during the process of background subtraction. The stationary bus is progressively eaten away in (a) and (b), while in (c) and (d) the ghost of the bus appears and then starts to merge back into the background. These problems are largely eliminated in [Fig. 22.5](#), which includes the same four frames.

but we defer that type of problem until Chapter 23. Motions of small animals and birds may also have to be considered. At this point we shall concentrate on fluttering vegetation, which is often prevalent in outdoor scenarios, even within cities.

The fluttering of vegetation can be more serious than might at first be imagined. It can result in the I values of some pixels oscillating between those of leaves, branches, and sky (or ground, buildings, etc.). Thus, the distributions of intensities and colors for any pixel may best be regarded as the superposition of several distributions corresponding to two or three component sources. Here what

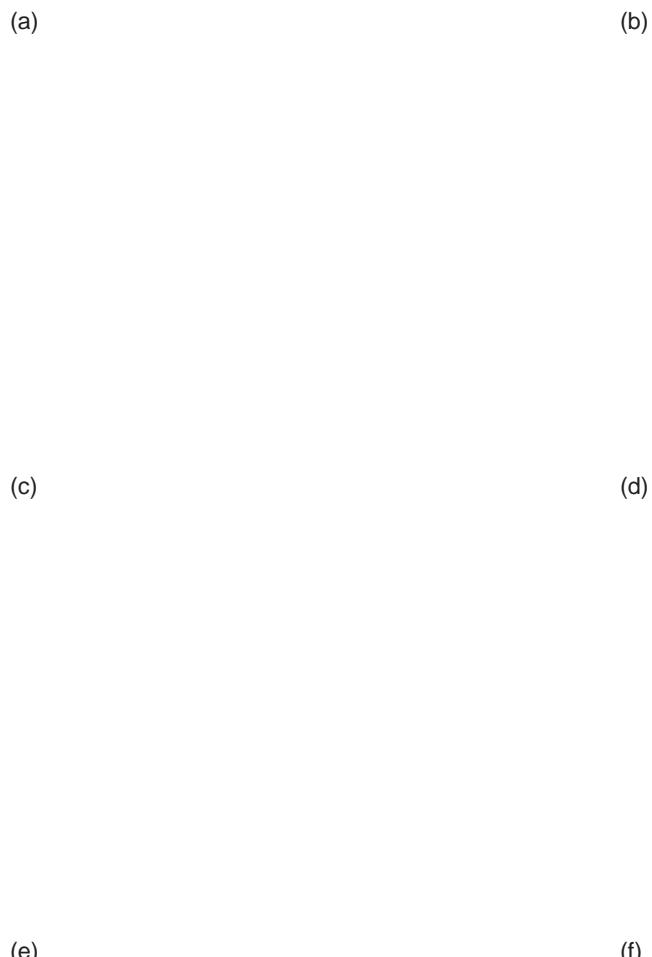


FIGURE 22.5

Background subtraction using a restrained temporal median filter. This figure shows a much more comprehensive set of frames than Fig. 22.4 because the method is more accurate. In particular, its responses (d, e, g, h) to the bus problems of Fig. 22.4 are vastly improved. Fluttering vegetation problems are indicated by fainter boundaries than for vehicles, but are entirely absent from the road region. In all frames the stationary shadows are completely eliminated by background subtraction: even the prominent bridge shadow is ignored; neither does it have much effect on the integrity of foreground objects. Note the low false negative rate for vehicles, and the fact that they only tend to be joined together in the distance. Overall, foreground object fragmentation and false shapes (including the effects of moving shadows) are the worst problems.

(g)

(h)

(i)

(j)

(k)

(l)

FIGURE 22.5

(Continued)

(a) (b)

FIGURE 22.6

Problems arising immediately after background subtraction. These two frames show clearly the noise problems that arise during background subtraction: the white pixels indicate where the current image fails to closely match the background model. Most of the noise effects occur for fluttering vegetation outside the road region. Morphological operations (see text) are used to largely eliminate the noise and to integrate the vehicle shapes as far as possible, as shown in [Figs. 22.4 and 22.5](#).

is important is that each of the component distributions could be quite narrow and well defined. This means that if each is known from ongoing training, any current intensity \mathbf{I} can be checked to determine whether it is likely to correspond to background. If not, it has to correspond to a new foreground object.

Models formed from multiple component distributions are commonly called mixture models: in practice, the component distributions are approximated by Gaussians, because the odd shape of the overall distributions is largely attributable to the existence of the separate components. Thus, we arrive at the terms Gaussian mixture models (GMMs) and mixtures of Gaussians (MoG). Note that the number of components at any pixel is initially unknown; indeed, a large proportion of pixels will have only a single component, and it may seem unlikely that the number would be much larger than three in practice. However, the fact that every pixel will have to be analyzed to determine its GMM is computationally burdensome, while the analysis can be unstable if the component distributions are not as tidy as suggested above. These factors mean that a computation intensive algorithm, the expectation maximization (EM) algorithm, has to be used to analyze the situation. In fact, while it is usual to use this rigorous approach to *initialize* the background generation process, many workers use simpler more efficient techniques for updating it, so that the ongoing process can proceed in real time. The GMM method determines for itself the number of component distributions to use, the judgment being based on a threshold value for the fraction of the total weight given to the background model.

Unfortunately, the GMM approach fails when the background has very high frequency variations. Essentially, this is because the algorithm has to cope with rapidly varying distributions that change dramatically over very short periods of time, so the statistics become too poorly defined. To tackle this problem, Elgammal et al. (2000) moved away from the parametric approach of the GMM (the latter essentially finds the weights and variances of the component distributions, and thus is parametric). Their nonparametric method involves taking a kernel smoothing function (typically a Gaussian) and for each pixel, applying it to the N samples of \mathbf{I} for frames appearing during the period Δt prior to the current time t . This approach is able to rapidly adapt to jumps from one intensity value to another, while at the same time obtaining the local variances at each pixel. Thus, its value lies in its capability to quickly forget old intensities and to reflect local variances rather than random intensity jumps. In addition, it is a probabilistic approach, but has no need of the EM algorithm, and this enables it to run highly efficiently in real time. In addition, it is capable of sensitive detection of foreground objects coupled with low false alarm rates. To achieve all this, it incorporates two further features:

1. It assumes independence between three different color channels, each having its own kernel bandwidth (variance). Together with the adoption of a Gaussian kernel function, this leads to a probability estimate given by:

$$P(\mathbf{I}) = \frac{1}{N} \sum_{i=1}^N \prod_{j=1}^C \frac{1}{(2\pi\sigma_j^2)^{1/2}} e^{-(I_j - I_{j,i})^2 / 2\sigma_j^2} \quad (22.16)$$

where i runs over the N samples taken in the time period Δt and j runs over the C color channels; this function is simple to calculate, though computation is further speeded up by using pre-calculated kernel function lookup tables.

2. It uses chromaticity coordinates for suppressing shadows. As these coordinates are independent of the level of illumination, and shadows can be regarded as poorly lit background, this means that they should largely merge back into the background. Hence, the foreground is much less likely to have shadows accompanying it after background subtraction. The chromaticity coordinates r , g , b are derived from the usual R , G , B coordinates by equations $r = R/(R + G + B)$, and so on, with $r + g + b = 1$.

In fact, shadows can be particularly problematic: not only do they distort the apparent shapes of foreground objects after background subtraction, but also they can connect separate foreground objects, and thus cause under-segmentation. The problem is reviewed by Prati et al. (2003), while Xu et al. (2005) have proposed a hybrid shadow removal method that makes use of morphology; see also Guan (2010).

Whatever method is used for background modeling leading to background subtraction and hence to foreground detection, the various blobs will need to be clustered and labeled using connected components analysis. Frame-to-frame tracking is then carried out by making correspondences between the blobs in the different frames. As in the case of Xu et al. (2005), morphology can be used to help with this process. Nevertheless, false positives tend to arise because of

shadows and illumination effects, while false negatives can arise from color similarities between foreground and background.

Overall, failures arise in two categories: (1) *the stationary background problem*, in which the shape of the foreground object is not defined accurately enough; (2) *the transient background problem*, in which the start and stop of the foreground object aren't found quickly enough. If the accuracy or reactivity of the background model are inadequate, background subtraction will lead to the detection of false objects: these are called “ghosts” by Cucchiara et al. (2003). In addition, as indicated above, shadows tend to compound these problems.

22.3.2 Practical Examples of Background Modeling

To add concreteness to the above discussion, a traffic surveillance video was taken and submitted to some of the algorithms mentioned above. For illustrative purposes the algorithms were kept as simple as possible. The raw data consisted of an AVI video from a digital camera (Canon Ixus 850 IS), which was decompiled into individual JPG frames, and though the JPG artifacts were fairly severe, no specific attempt was made to eliminate them. The frame size was 320×240 pixels in RGB color, but only the 8-bit lightness component was used for the main tests. While the video was taken at 15 frames per second, only every tenth frame was used for the test, which comprised 113 frames. Of these, the first 10 can be regarded as initialization training material and these are not considered further. During the test, a bus arrived and was stationary at a bus stop for some time. The overall sequence is illustrated in Fig. 22.5; however, for reasons of space the only frames included in the figures are those that illustrate the problems well. Note that the video was taken on a sunny day, and that there are a great many shadows, which over the minute or so of the video did not change markedly. On the other hand, some camera motion is detectable, possibly due to movement of the bridge. Thus, there are many ways in which the raw data were not ideal; these therefore impose exacting conditions on the success of any algorithm.

Figure 22.4 shows some of the results obtained by applying a temporal median. Frames (a) and (b) show the bus stationary at the bus stop and being progressively eaten up as it starts to merge with the background. Frame (c) shows the bus moving away from the bus stop, leaving a large “ghost” behind it. Frame (d) shows that the ghost remains for some time and is a substantial factor to be taken into account by any foreground interpretation procedure.

To overcome these problems, the median was restrained so that it could only take into account pixel intensities within a limited number of gray levels of the current median value; in this way, it took on something of the characteristics of a mode filter (a temporal mode filter *per se* would lock on to the current value too inflexibly and not adapt well to the changing intensity distribution). The results are shown in Fig. 22.5. It is clear that the restrained median largely eliminates the two problems mentioned above (*viz.* the observed vehicle being eaten away while stationary and

leaving a ghost behind it when moving on). For this reason the remainder of the tests used only the restrained median. Problems seen in Fig. 22.5 include:

1. Eating away of foreground objects, leaving unusual shapes (e.g. (d), (i)).
2. Fragmentation of foreground objects (e.g. (b), (f)).
3. Shadows accompanying the moving foreground objects (e.g. (c), (g), (j)).
4. Joining of foreground objects that should appear separated (e.g. (i), (j)).
5. Signals from fluttering vegetation (e.g. (a), (k)).

Item 2 can be considered as an extreme case of item 1. Item 3 is bound to arise as the shadows are moving at the same speed as the vehicles that give rise to them, and straightforward background suppression or alternatively moving object detection alone will not eliminate them. In general, unless color interpretation will help (we return to this possibility below), high-level interpretation is needed to achieve satisfactory elimination. Item 4 is due partly to the effects of vehicle shadows, which tend to connect vehicles, especially when seen in the distance. The morphological operations that were applied (see below) also tended to make vehicles become joined. Item 5 is never manifest in the road region, i.e. between the lines of black graphics dots shown in the frames. This is because, in this case, the vegetation is high up, away from the road region. In addition, it is largely eliminated by morphological operations. In fact, Fig. 22.6 shows the results obtained immediately after background subtraction. It is clear that there is a serious noise problem, caused largely by (a) camera noise, (b) the effects of JPG artifacts, (c) fluttering vegetation, and (d) the effects of slight camera motion. Interestingly, two applications of a single pixel erosion operation were sufficient to eliminate the noise almost completely, these being followed by four applications of a single dilation operation to help restore vehicle shapes. (Overall, this corresponds to a 2-pixel opening operation followed by a 2-pixel dilation operation.) These morphological operations were selected to give roughly optimal results—in particular, low probability of failing to capture foreground objects in any individual frame, coupled with pressure to maintain object shapes as far as reasonable, and not to join vehicles together unavoidably. The point is that background subtraction must aim to pass on sufficient useful information to subsequent foreground object identification, tracking, and interpretation stages.

One of the remarkable aspects of the results is the total elimination of stationary shadows and lack of problems arising from this. On the other hand, two other sorts of shadows are manifest—those arising from moving objects, and those falling on moving objects (the latter arise in the video both from the bridge and from the other causes of ground shadows): neither of these sorts of shadows are eliminated. Other problems are those of reflections, particularly from the windows of the bus (see the frame in Fig. 22.5(g)), and secondary illumination from moving vehicles.

Lastly, it was felt worthwhile to attempt utilizing the original color images and augmenting the background model by using the chromaticity coordinates as

outlined in the previous section. In fact, while in some respects improvements occurred, these were more than canceled out by increased numbers of false negatives and fragmented shapes of the foreground objects. No results are shown here, but whereas Elgammal et al. (2000) were able to show excellent results obtained in this way, with the video used here no improvement seemed to be achievable by this approach. This requires some explanation. High up among the reasons is the effect of the highly variegated colors and intensities of the many different vehicles. In particular, some vehicles turned out to have body intensities close to those of shadows, whereas others had windows or transparent roofs of similar intensity. These had the effect of eliminating large portions of vehicles together with the shadows, thereby increasing the incidence of false negative and false shape information. However, even Elgammal et al. (2000) point out that intensities have to be used carefully in a way that will bolster up the shadow removing capabilities of the chromaticity information, and here there appeared to be no way this could be achieved. Overall, all the color and grayscale information has to be taken into account in a more considered and strategic way, and this demands a thoroughgoing statistical pattern recognition approach in which objects are identified one by one in a high-level schema rather than by relatively chancy *ad hoc* methods: the latter definitely have their place but their use must not be pushed beyond what is reasonable. One example of the use of object-by-object recognition is the identification of road markings, which need to be, and could easily be, identified whether or not vehicle shadows cover them. This could then lead to a much more viable strategy for identifying, tracking, and eliminating vehicle shadows. Meanwhile, in situations where the road has almost no color content, as in the traffic surveillance trials described above, it is difficult to remove shadows effectively merely by using chromaticity information.

22.3.3 Direct Detection of the Foreground

In the previous subsection it has been seen that background modeling followed by background subtraction constitutes a powerful strategy for the location of moving targets in image sequences. Nevertheless, for all sorts of reasons it is limited in what it can achieve. While these reasons devolve into problems such as changes in ambient illumination, effects of shadows, irrelevant motions such as fluttering leaves, and color similarities between foreground and background, there is one whole tranche of information that is absent; specifically, there is a total lack of information on the nature of the target objects, including size, shape, location, orientation, color, speed, and probability of occurrence. If this sort of information were obtainable, there would be some chance of incorporating it into a complete target detection system and achieving close to perfect detection capability. Indeed, it seems possible that in some cases ignoring the background and attempting direct detection of the foreground might be a better first approximation. Such a procedure might well be both effective and efficient for the case of face

detection, for example. In what follows we consider how direct foreground detection might be achieved.

Direct foreground detection is only possible if a suitable foreground model is available or can be constructed. It would seem that this requires a specialization to each particular application, such as pedestrian detection or vehicle detection. However, some workers (e.g. Khan and Shah, 2000) have managed to achieve it more generically by a bootstrapping process. They start with background modeling and background subtraction, locate foreground objects by an “exception to background” procedure, and thus create initial foreground models. In subsequent frames these are enhanced, mostly using Gaussian-based models: GMMs and non-parametric models have been employed for this. However, a difference relative to background modeling is that the latter applies continuously (with updates) for the same camera, whereas each foreground object must have its own individual model that is learnt anew for that object. So, background modeling is only applied to initially locate the foreground object: thereafter, the foreground model is built and tracked, albeit in a similar way to what happens with background modeling.

More recently, in a new class of algorithm, Yu et al. (2007) use a GMM for simultaneously modeling both foreground and background. In this way, a tension is built between foreground and background that potentially leads to higher segmentation accuracy, and this does seem to have been achieved in practice. Against this, the algorithm has to be initialized by marking areas of definite foreground and background, and then it continues to track autonomously. However, there seems to be no reason why initialization should not also be carried out autonomously with the help of an initial stage of background modeling.

22.4 PARTICLE FILTERS

When trying to track foreground objects, independent detection in each frame, followed by appropriate linking, does not make best use of available information; neither does it achieve optimum sensitivity: this will be obvious when noting that averaging slowly moving objects over a number of frames can boost signal-to-noise ratio. In addition, over time—and sometimes over very few frames—objects can change radically in appearance, so tracking is needed in order to ensure continued capture. Nowhere is this more obvious than in the case of a guided missile approaching a target over several miles, as during the time of flight the size, scale, and the resolution will increase dramatically. But even in cases where a person is being tracked, rotation of the head will present if anything even more dramatic changes in appearance. With the radically changing backgrounds arising with moving and rotating objects, sensitive robust tracking is clearly of fundamental importance. To achieve this, optimal methods are needed. In particular, in the face of radical change, we need to know what is the most *likely* position of an object that is being tracked. Optimal estimation of likelihood implies the need for Bayesian filtering.

To achieve this, we start by considering the observations \mathbf{z}_1 to \mathbf{z}_k of an object in successive frames, and the corresponding deduced states of the object \mathbf{x}_0 to \mathbf{x}_k (there is no zeroth value of \mathbf{z} because it takes at least two frames to estimate the velocity \mathbf{v}_k , which forms part of the state information). At each stage, we need to estimate the most probable state of the object, and Bayes rule gives us the *a posteriori* probability density:¹

$$p(\mathbf{x}_{k+1}|\mathbf{z}_{1:k+1}) = \frac{p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})p(\mathbf{x}_{k+1}|\mathbf{z}_{1:k})}{p(\mathbf{z}_{k+1}|\mathbf{z}_{1:k})} \quad (22.17)$$

where the normalizing constant is:

$$p(\mathbf{z}_{k+1}|\mathbf{z}_{1:k}) = \int p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1})p(\mathbf{x}_{k+1}|\mathbf{z}_{1:k})d\mathbf{x}_{k+1} \quad (22.18)$$

The prior density is obtained from the previous time-step:

$$p(\mathbf{x}_{k+1}|\mathbf{z}_{1:k}) = \int p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k})d\mathbf{x}_k \quad (22.19)$$

but note that this is only valid because of the Markov process (of order one) assumption commonly taken to simplify Bayesian analysis, which leads to:

$$p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{z}_{1:k}) = p(\mathbf{x}_{k+1}|\mathbf{x}_k) \quad (22.20)$$

In other words, the transition probability for the update $\mathbf{x}_k \rightarrow \mathbf{x}_{k+1}$ depends only indirectly on $\mathbf{z}_{1:k}$, via previous updates.

General solutions of this set of equations—in particular, Eqs. (22.17) and (22.19)—do not exist. However, restricted solutions are possible, as in the case of the Kalman filter (see Chapter 19), which assumes that all posterior densities are Gaussian. In addition, particle filters can be used to approximate the optimal Bayesian solution when Gaussian constraints are inapplicable.

The particle filter, also known as sequential importance sampling (SIS), the sequential Monte Carlo approach, bootstrap filtering and condensation, is a recursive (iteratively applied) Bayesian approach that at each stage employs a set of samples of the posterior density function. It is an attractive concept because in the limit of large numbers of samples (or “particles”), the filter is known to approach the optimal Bayesian estimate (Arulampalam et al., 2002).

To apply this method, the posterior density is reformulated as a sum of delta function samples:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^N w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \quad (22.21)$$

¹It is much easier to see the relation to Bayes rule if conditional dependence on $\mathbf{z}_{1:k}$ is eliminated; once this is done, all remaining subscripts are equal to $k + 1$, and suppressing them, Eqs. (22.17) and (22.18) become standard Bayes rule. Reinstating dependence on $\mathbf{z}_{1:k}$ is of course necessary when dealing with tracking over $k + 1$ frames involving previous observations $\mathbf{z}_{1:k}$.

where the weights are normalized by:

$$\sum_{i=1}^N w_k^i = 1 \quad (22.22)$$

Substituting into Eqs. (22.17)–(22.19), we obtain the posterior:

$$p(\mathbf{x}_{k+1} | \mathbf{z}_{1:k+1}) \propto p(\mathbf{z}_{k+1} | \mathbf{x}_{k+1}) \sum_{i=1}^N w_k^i p(\mathbf{x}_{k+1} | \mathbf{x}_k^i) \quad (22.23)$$

where the prior now takes the form of a mixture of N components.

In principle, this gives us a discrete weighted approximation to the true posterior density. In fact, it is often difficult to sample directly from the posterior density: this problem is normally solved by sequential importance sampling (SIS) from a suitable “proposal” density function $q(\mathbf{x}_{0:k} | \mathbf{z}_{1:k})$. It is useful to take an importance density function that can be factorized:

$$q(\mathbf{x}_{0:k+1} | \mathbf{z}_{1:k+1}) = q(\mathbf{x}_{k+1} | \mathbf{x}_{0:k}, \mathbf{z}_{1:k+1}) q(\mathbf{x}_{0:k} | \mathbf{z}_{1:k}) \quad (22.24)$$

following which, the weight update equation can be obtained (Arulampalam et al., 2002) in the form:

$$\begin{aligned} w_{k+1}^i &= w_k^i \frac{p(\mathbf{z}_{k+1} | \mathbf{x}_{k+1}^i) p(\mathbf{x}_{k+1}^i | \mathbf{x}_k^i)}{q(\mathbf{x}_{k+1}^i | \mathbf{x}_{0:k}^i, \mathbf{z}_{1:k+1})} \\ &= w_k^i \frac{p(\mathbf{z}_{k+1} | \mathbf{x}_{k+1}^i) p(\mathbf{x}_{k+1}^i | \mathbf{x}_k^i)}{q(\mathbf{x}_{k+1}^i | \mathbf{x}_k^i, \mathbf{z}_{k+1})} \end{aligned} \quad (22.25)$$

where the path $\mathbf{x}_{0:k}^i$ and history of observations $\mathbf{z}_{1:k}$ have been eliminated—as this is necessary if the particle filter is to be able to track recursively in a manageable way.

In fact, pure SIS has the largely unavoidable problem that all but one particle will have negligible weight after a few iterations. More precisely, the variance of the importance weights is only able to increase over time, leading ineluctably to this degeneracy problem. However, one simple means of limiting the problem is to resample particles so that those with small weights are eliminated, while those with large weights are enhanced by duplication. Duplication can be implemented relatively easily, but it also leads to so-called “sample impoverishment,” i.e. it still results in some loss of diversity among the particles, which is itself a form of degeneracy. Nevertheless, if there is sufficient process noise, the result may prove to be adequate.

One basic algorithm for performing the resampling is “systematic resampling,” and involves taking the cumulative discrete probability distribution (in which the original delta function samples are integrated into a series of steps) and subjecting it to uniform cuts over the range 0–1 to find appropriate indexes for the new samples. As seen in Fig. 22.7, this leads to small samples being eliminated and strong samples being duplicated, possibly several times. The result is called

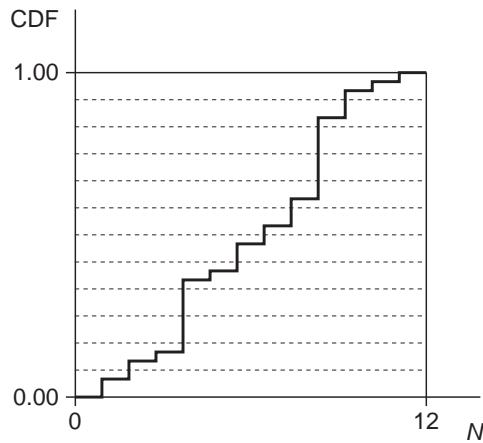


FIGURE 22.7

Use of the cumulative distribution function (CDF) to perform systematic resampling. Applying the regularly spaced horizontal sampling lines shows the cuts needed to find appropriate indexes (N) for the new samples. The cuts tend to ignore the small steps in the CDF and to accentuate the large steps by duplicating samples.

sampling importance resampling (SIR), and is a useful first step on the way to producing stable sets of samples. With this particular approach, the importance density is chosen to be the prior density:

$$q(\mathbf{x}_{k+1} | \mathbf{x}_k^i, \mathbf{z}_{k+1}) = p(\mathbf{x}_{k+1} | \mathbf{x}_k^i) \quad (22.26)$$

Appealing to Eq. (22.25) shows that the weight update equation becomes enormously simplified to:

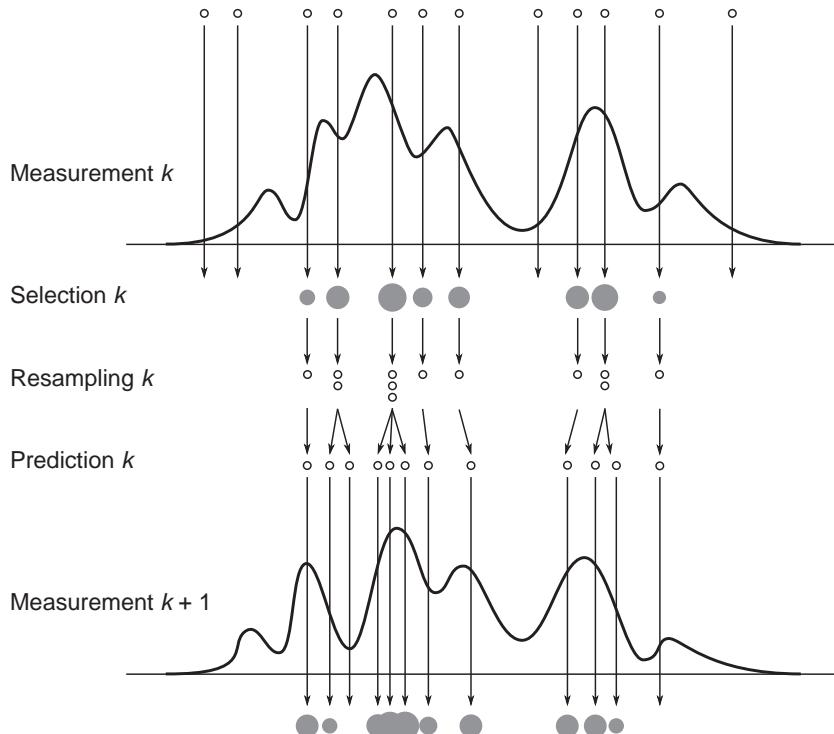
$$w_{k+1}^i = w_k^i p(\mathbf{z}_{k+1} | \mathbf{x}_{k+1}^i) \quad (22.27)$$

Moreover, as resampling is applied at every time index, previous weights w_k^i are all given the value $1/N$, so we can simplify this equation to:

$$w_{k+1}^i \propto p(\mathbf{z}_{k+1} | \mathbf{x}_{k+1}^i) \quad (22.28)$$

As can be seen in Eq. (22.26), the importance density is taken to be independent of measurement \mathbf{z}_{k+1} , so the algorithm is restricted with regard to observational evidence, and this is one cause of the loss of particle diversity mentioned earlier.

The Condensation method of Isard and Blake (1996) goes some way to eliminating these problems by following the resampling with a prediction phase during which a diffusion process separates any duplicated samples, thereby helping to maintain sample diversity. This is achieved by applying a stochastic dynamical model that has been trained on sample object motions. Figure 22.8 gives an overall perspective on the approach, and includes all the sampling and other processes that have been discussed above.

**FIGURE 22.8**

Perspective on the processes involved in particle filtering. Note how the filter cycles repeatedly through the same basic sequence.

The concept is taken further in the condensation approach (Isard and Blake, 1998) by using a mixture of samples, some using standard SIR and some using an importance function depending on the most recent measurement \mathbf{z}_{k+1} but ignoring the dynamics. Thus, this complex method reflects the need to ensure continued sample diversity: it also aims to combine low and high-level approaches to tracking by noting that model switching may be necessary when handling real-world tasks such as tracking human hands.

Similar ideas and motivation were employed by Pitt and Shephard (1999) in their auxiliary particle filter (APF). This generates particles from an importance distribution depending on the most recent observations, and then samples the posterior using this importance density. The algorithm involves an additional likelihood computation for each particle, but overall the computational efficiency is improved because fewer particles are needed. Nevertheless, Nait-Charif and McKenna (2004) found that the method gave only limited improvement relative to SIR. They went on to make a comparison with the iterated likelihood weighting (ILW) scheme. In this approach, after an initial iteration of SIR, the sample set is split randomly into two sets of equal size; one of these is migrated to regions of high likelihood and the other is handled

normally. The purpose is to cope on the one hand with situations where the prior is sound and on the other hand with situations where it is not and regions of high likelihood need to be explored. When tracking human heads, the method proved to be a significantly more robust tracker than either SIR or the APF. Perhaps oddly, the ILW is designed to reduce approximation error rather than to give unbiased estimates of a posterior. This means that it is not based completely on probabilistic methods. On the other hand, as for the Isard and Blake condensation approach mentioned above, it is intended to match a variety of scenarios that can be found when tracking under real-world conditions, where it is difficult to model all probabilities accurately.

Many more particle filter methods have been developed over the past decade or so. Several incorporate the Kalman filter and its extended and “unscented” versions, in attempts to optimize likelihoods when sample diversity turns out to be insufficient. More recently “regularized” and “kernel” particle filters (Schmidt et al., 2006) have been developed to tackle the problem of sample impoverishment. These perform resampling using a continuous approximation to the posterior density, typically using the Epanechnikov kernel (Comaniciu and Meer, 2002). The mean shift approach is in this category. Essentially, the mean shift algorithm is a means of climbing density gradients to identify underlying modes in sparse distributions, and involves moving a sampling sphere around the space being searched. This makes it a good iterative search technique, though it is only suitable for locating one mode at a time. It complements the particle filter formalism well, as it can be used to refine the accuracy with which objects may be found, and works well even if a limited number of particles are employed. It has recently been applied by Chang and Lin (2010) for tracking various parts of the moving human body.

At this stage it is starting to become apparent that different tracking applications will demand different types of particle filter. This will depend on a variety of factors including how jerky the motion is, whether rotations will be involved, whether occlusions will occur and if so for how long—and of course on the appearance and variability of the objects being tracked. It should be noted that all the theory and most of the ideas presented above reflect abstract situations and the concentration is on relatively small, well localized objects that are considered locally, i.e. the filters themselves will not have a global understanding of the situation. Thus, they must be categorized as low or intermediate level vision. In contrast, the human eye is an excellent tracker by virtue of its capability for thinking about what objects are present and which ones have moved where, including passing behind other objects or even temporarily out of the scene. Clearly, we must not expect too much from particle filters just because they are based on probabilistic models.

An important advantage of particle filters is that they can be used to track multiple objects in an image sequence. This is because there is no record of which object is being tracked by which particles. However, this possibility arises only because no restrictions are placed on the posterior densities; in particular, they are

not assumed to be Gaussian as in the case of Kalman filters. Indeed, if Kalman filters are used for tracking, each object must be tracked by its own Kalman filter.

Once a suitable approach to particle filtering has been arrived at, it is necessary to determine how to implement it. The basic means of achieving this is via appearance models. In particular, color and shape models are frequently used for this purpose. However, before delving into this topic, it will be useful to find what can be achieved by color analysis using what is by now quite an old approach—that of color indexing via color histogram matching.

22.5 USE OF COLOR HISTOGRAMS FOR TRACKING

One of the most useful tools that is available for object tracking was developed as early as 1991 by Swain and Ballard (1991) in a paper called “Color indexing.” The aim of that work was to index from a color image into a large database of models. In a sense that idea is the inverse of the tracking problem, as its purpose was to search for the model with the best match to a given image rather than to search for instances of a given model in frames from an image sequence.² However, apart from one important difference which will be discussed below, this is a rather minor point.

The main idea behind the color indexing approach is that of matching color histograms rather than the images themselves. There is an obvious validity in this approach, in that if the images match, so will their color histograms. What is more, as the histograms have no memory of where in the image a particular color originated, histograms are invariant to translation and rotation about the viewing axis (so-called “in-plane rotation”). Also relevant is the fact that a planar object that is subject to out of plane rotation will still have the same color histogram, although it will involve different numbers of pixels, so normalization will be required. The same applies to objects that are at different depths in the scene: the histogram profile will be unchanged, but it will have to be normalized to allow for the different numbers of pixels that are involved. Finally, a spherical or cylindrical object with the same set of colors distributed similarly over its surface will again have the same histogram. While exact adherence to this scenario might be relatively rare, it would apply almost perfectly for a ball of wool or a football, and with varying degrees of exactness for a shaven human head or torso. In fact, the main problem with use of histograms for recognition is the possible ambiguity it could bring, but when tracking a known object that has moved only a small distance between frames, this problem should be a minor one.

The above explanation demonstrates the potential power of the histogram approach, but raises an important question about what happens when the object moves in such a way as to be larger or smaller than the model, whether through

²Actually, database searches fall in the realm of classification, whereas the process of tracking assumes implicitly that the object in question has already been identified.

depth scaling or through out of plane rotation. In particular, if it becomes smaller, this will mean that the model will be matched partly against the object background. Swain and Ballard sought to minimize this effect by taking the following intersection measure rather than any sort of correlation between the image I and model M histograms:

$$\sum_{i=1}^n \min(I_i, M_i) \quad (22.29)$$

This would have the effect of discounting any pixels of a given color in excess of those expected in the model histogram (including both those whose colors are simply not represented in the model and those that have limited representation). The above expression was then normalized by the number of pixels in the model histogram. However, here we follow Birchfield (1998) in normalizing by the number of pixels in the image histogram, to reflect the point made earlier that we are searching for the best image match rather than the best model match:

$$H_N(I, M) = \frac{\sum_{i=1}^n \min(I_i, M_i)}{\sum_{i=1}^n I_i} \quad (22.30)$$

At first sight, this formula might appear wrong, in that a match over fewer pixels would be normalized out, still representing perfect agreement and giving a normalized intersection of unity. Note for example that in shape matching, it is common to use the formula $(A \cap B)/(A \cup B)$, where A and B are sets representing object areas, which would give a value less than unity in the case $A \supset B$. However, Eq. (22.30) is designed to cope well with partial occlusions in the image, which will lead to the intersection with M being reduced, yielding the I values, which would then cancel with the denominator, giving the answer 1.

The overall effect of using the normalized intersection of Eq. (22.30) is that the method has the twin advantages of minimizing the effects of background and canceling the effects of occlusion, while also coping well (and in some cases exactly) with varying viewpoints. The problem of varying scale remains, but this can be countered by preliminary segmentation of the object and scaling its histogram to the size of the model histogram.

There remains one further important consideration when matching an image against a model—that the model might have become out of date under varying levels of illumination. To a large extent, the latter can be considered as varying levels of *luminance*, with the *chrominance* parameters remaining more or less unchanged. This problem can be addressed by changing to different color representations. For example, we can move from the RGB representation to the HSI representation (see Chapter 20), and then use the hue (H) and saturation (S) parameters. However, more protection will be available by color normalization

(dividing by I)—though it is far easier and less computation intensive to normalize the RGB parameters directly:

$$r = \frac{R}{(R + G + B)} \quad (22.31)$$

$$g = \frac{G}{(R + G + B)} \quad (22.32)$$

$$b = \frac{B}{(R + G + B)} \quad (22.33)$$

but because $r + g + b = 1$, we should ignore one of the parameters, e.g. b .

While the above arguments suggest that luminance should be totally ignored, this is inadvisable, as colors that are close to the black–white line in color space (where saturation $S \approx 0$) would be indistinguishable. Indeed, Birchfield (1998) cites the “dangerous” case of dark brown hair looking similar to a white wall if luminance is ignored. For these reasons, most workers use different sizes and numbers of histogram bins for luminance and chrominance information. Here, we have to remember that a full-sized color histogram with 256 bins in each of the color dimensions would be both large and clumsy, and would not easily be searchable in real time—an especially important factor in tracking applications. In addition, such a histogram would not be well populated and would lead to very noisy statistics. For this reason, 16–40 bins per color dimension are much more typical. In particular, $16 \times 16 \times 8$ bins are widely used, 8 being the number in the luminance channel. Note that these numbers correspond, respectively, to 16, 16 and 32 levels per channel, and that a 512×512 image would lead to an average occupation number of 128 per bin; however, a 256×256 image would give an average occupation of just 32 per bin, which is distinctly low and liable to be inaccurate (though this would depend very much on the type of data).

Birchfield (1998) reported that when used for head tracking, the color histogram method was able to follow a head reliably, though it became “unstable” when the head was in front of a white board whose color was quite close to that of skin. This behavior is understandable, as it has already been noted that the histogram approach is invariant to translation (hence, as long as the head is somewhere within the image, the histogram tracker will be unlikely to lose it). These points show that ultimately the histogram tracker approach is limited, and needs to be enhanced by other means, in particular some means of detecting object outlines. To achieve this, Fieguth and Terzopoulos (1997) used (1) simple M -ary hypothesis testing of position around the previous position, the displacements merely being those at the $M = 9$ points in a 3×3 window; (2) a highly nonlinear velocity prediction scheme involving step incremental corrections for acceleration, deceleration, and damping to avoid oscillations; and (3) color histograms bins based exclusively on chrominance. The reason for these simplifications was

to achieve real-time operation for full frames (640×480 pixels) at 30 frames per second—at which rate object displacements become much smaller and easier to track.

Birchfield (1998) developed a more sophisticated approach, based on approximating the shape of the human head by a vertical ellipse with a fixed aspect ratio of 1:2. Then, in common with previous contour trackers he measured the goodness of match by computing the normalized sum of gradient magnitudes around the boundary of the ellipse, though (a) he summed the gradient values at all points on the boundary rather than just at selected points,³ and (b) he took the component of the gradient along the perpendicular to the boundary. This led to a shape model $\mathbf{s}(x, y, \sigma)$ with three parameters— x , y denoting the ellipse location and σ denoting its semi-minor axis—and the following goodness of fit parameter:

$$\psi(\mathbf{s}) = \frac{1}{N_\sigma} \sum_{i=1}^{N_\sigma} |\mathbf{n}_\sigma(i) \cdot \mathbf{g}_s(i)| \quad (22.34)$$

Here, N_σ is the number of pixels on the boundary of an ellipse with semi-minor axis σ , $\mathbf{n}_\sigma(i)$ is the unit vector normal to the ellipse at pixel i , and $\mathbf{g}_s(i)$ is the local intensity gradient vector. Normalized goodness of fit parameters for boundary shape (ψ_b) and color (ψ_c) are added and used to obtain an optimum fit:

$$\mathbf{s}_{\text{opt}} = \arg \max_{\mathbf{s}_i} \{\psi_b(\mathbf{s}_i) + \psi_c(\mathbf{s}_i)\} \quad (22.35)$$

As discussed earlier, when the color module was tested on its own, it performed well, but somewhat unstably when the background color was close to that of skin. All this was corrected by adding the gradient module. However, the gradient module on its own performed less well than the color module on its own. As time progressed the gradient model tended to become distracted by the background, not having any inbuilt design features to counteract this. Moreover, in a cluttered background it behaved even less well; and it was not able to handle large accelerations adequately because of its limited ability to probe for high gradient regions, and its consequent propensity for attaching itself to the wrong ones. Fortunately, the two modules were able to complement each other's capabilities; in particular, the color module helped the gradient module by its ability to ignore background clutter, and by providing a larger region of attraction. Finally, when the human subject turned around, so that only his hair was visible, the gradient module was able to take over and handle rescaling correctly as the subject moved; and it was able to prevent the color tracker from slipping down the subject's neck, which had a similar color histogram. All this signals that two or more strategies for tracking can be useful in real-world situations where enough

³In fact, this is unusual: most workers sample at a set of 100 or so points on the boundary.

information needs to be brought to bear to provide correct tracking interpretations on an ongoing basis. It also signals that the color histogram type of tracking module is exceptionally powerful, and tends to need only minor tweaks to keep it properly on lock. Overall, however, the outstanding factor that needs further detailed attention and development is the handling of occlusion: this needs to be arranged by design rather than by tweaks, as we shall see in a later section.

22.6 IMPLEMENTATION OF PARTICLE FILTERS

The particle filter formalism is a very powerful one, mediated by generic probability-based optimization, yet needing to be taken further to achieve its promise. To arrange this, it needs to be applied to real objects and thus appearance models have to be taken into account—though in the present context it will be more accurate and general to refer to them as observation models. Our particle filter formalism already embodies these in the form of conditional densities $p(\mathbf{z}_k|\mathbf{x}_k)$, see Eq. (22.28).

At this stage we need to specialize the observations. Here, we illustrate the process by considering the color and assumed elliptical shape of a human head: these can be thought of as region-based (r) and boundary-based (b) properties, each with their own likelihoods. Taking the latter to be conditionally independent, we can factorize $p(\mathbf{z}_k|\mathbf{x}_k)$ as follows:

$$p(\mathbf{z}_k|\mathbf{x}_k) = p(\mathbf{z}_k^r|\mathbf{x}_k)p(\mathbf{z}_k^b|\mathbf{x}_k) \quad (22.36)$$

Clearly, the region-based likelihood will depend not only on the color but also on the shape of the region it is in. Nevertheless, the conditional independence assumption will be valid, as we are really interested in the colors within the boundary and the gradient values along it.

To proceed further, we assume that color histograms I and M have been obtained for the image and the target model, within the current region r . Following Nummiaro et al. (2003) and many other workers—and at this point abandoning the Swain and Ballard (1991) normalized intersection formalism—we normalize them to unity as p^I, p^M , respectively. To compare these distributions, it is convenient to use the Bhattacharyya coefficient (here expressed as a sum rather than an integral) that expresses the similarity between the distributions:

$$\rho(p^I, p^M) = \sum_{i=1}^m \sqrt{p_i^I p_i^M} \quad (22.37)$$

To display the *distance* between the distributions, we simply apply the measure:

$$d = \sqrt{1 - \rho(p^I, p^M)} \quad (22.38)$$

Ideally, the color distribution will be close to the target distribution, so these should differ only as a Gaussian error function. Remembering that p^I is actually a function of \mathbf{x}_k , we now find the region (and color) conditional likelihood:

$$p(\mathbf{z}_k^r | \mathbf{x}_k) = \frac{1}{(2\pi\sigma_r^2)^{1/2}} e^{-(d^2/2\sigma_r^2)} = \frac{1}{(2\pi\sigma_r^2)^{1/2}} e^{-[(1 - \rho(p^I(\mathbf{x}_k), p^M)))/2\sigma_r^2]} \quad (22.39)$$

Making a similar assumption that the estimated gradient positions in the image I will differ from those in the target model M by a Gaussian error function, we find the boundary conditional likelihood:

$$p(\mathbf{z}_k^b | \mathbf{x}_k) = \frac{1}{(2\pi\sigma_b^2)^{1/2}} e^{-(G^2/2\sigma_b^2)} \quad (22.40)$$

where G represents the sum of the gradient magnitude values perpendicular to the local boundary positions.

Combining the last two equations, as specified by Eq. (22.36), now provides the required estimate of $p(\mathbf{z}_k | \mathbf{x}_k)$, which in turn leads via the particle filter formulation to an estimate of $p(\mathbf{x}_k | \mathbf{z}_{1:k})$. This essentially completes the long series of arguments and calculations comprising the particle filter scenario.

In fact, there are several further aspects to consider. The first is that it is natural to weight the contributions made by the various pixels to the color histograms. In particular, the pixels nearest to the centers of the ellipses should be weighted higher than those near their boundaries, so that any inaccuracies in the center locations will be minimized. For example, Nummiaro et al. (2003) used the weighting function:

$$k(r) = \begin{cases} 1 - \frac{r^2}{r_0^2} & : r < r_0 \\ 0 & : r \geq r_0 \end{cases} \quad (22.41)$$

with $r_0 = \sqrt{a^2 + b^2}$, a and b being the semi-major and semi-minor axes of the ellipses. In fact, Nummiaro et al. (2003) placed such reliance on this weighting that their particle filter did not use a separate boundary likelihood $p(\mathbf{z}_k^b | \mathbf{x}_k)$. In contrast, Zhang et al. (2006) used both, almost exactly as described above, albeit with an auxiliary particle filter incorporating mean shift filtering.

Another important aspect not so far mentioned is the need to adapt the target model M to keep it up to date, e.g. with regard to the size and orientation of the real-world target. Nummiaro et al. (2003) achieved this using the commonly applied “learning/forgetting” operation:

$$p_{k+1,i}^M = \alpha p_{k,i}^M + (1 - \alpha)p_{k,i}^I, \quad i = 1, 2, \dots, m \quad (22.42)$$

which mixes in a little of the recent image data while forgetting a correspondingly small amount of the old model data. During this process care is taken to avoid

mixing in outlier data, such as when an object is partly occluded. Even with this precaution, it should be borne in mind that use of an adaptive model is potentially dangerous; while it helps by valid adaptation to appearance changes, it gives an increased sensitivity to extended occlusions and loss of target.

While heads are typically tracked using 2-D position (x, y) and ellipse shape parameters (a, b), it can normally be assumed that the ellipse is vertically aligned. However, when viewed from overhead, in-plane orientation (θ) is also an important parameter. Sometimes, similar models are used for individual human limbs, though rectangles have also been employed. However, ellipses provide a simple, easily parametrized shape, and can be specified with as few as three parameters (x, y, b); these can even be used to track whole human figures using 3 or 4 parameters (Nummiaro et al., 2003). On the other hand, when torsos or hands are being tracked, closed curves may not be appropriate, and it is common to use parametric spline curves.

With the type of particle filter design outlined above, performance in the event of occlusions is a vexed question.⁴ In principle, if a significant change such as a strong partial occlusion occurs, the simple artifice of putting the tracker on hold is often sufficient to allow it to recover and continue tracking. However, to ensure recovery, the tracker might have to wait for a background subtraction routine to signal that the object is again present (Nait-Charif and McKenna, 2006). In any case, a background subtraction module is useful for signaling when a totally new object has entered the scene. Finally, when objects leave the scene, some memory of their appearance and position is useful in case they re-enter the scene after a short time in the same or other location (when humans appear indoors, there are usually a limited number of entry and exit points, and re-entry via the same one will generally be the most likely possibility). However, there is a danger of instituting a rather *ad hoc* set of algorithms to solve such problems, when what is needed is a more absolute object recognition module to positively identify individuals, or at least to search for the most likely identifications, together with sets of probabilities. A particular example of this type of situation is when two pedestrians walking in opposite directions (a) pass each other without interacting, but with the one momentarily occluding the other, or (b) stop, shake hands, and then proceed, or (c) stop, shake hands, and then retrace their steps. Scenario (c) involves merging of profiles and can be as difficult to handle as occlusion: in any case temporary partial occlusion involves merging of the figures; only seldom does complete occlusion and total disappearance of one figure occur. It ought to be stressed that scenario (a) is handled well by a Kalman filter module that uses

⁴In this area, many claims and counter-claims about relative effectiveness of tracking and occlusion handling capabilities are made in various papers. As the claims are often made on different datasets, it is difficult to know the true position. However, the particle filter has quite a high level of intrinsic robustness. This is because “less likely object states have a chance to temporarily remain in the tracking process, [so] particle filters can deal with short-lived occlusions” (Nummiaro et al., 2003). Hence minor popping up in a judicious way using other modules can often boost performance significantly.

continuity of velocity to aid interpretation; scenario (b) is handled badly or not at all by such a module, depending on the time delay; while scenario (c) is not handled at all by such a module.⁵ In general, when processing human interactions, the Kalman filter has to be used tentatively, to throw up *hypotheses* about motion. However, it is possible to incorporate Kalman filters usefully into a particle filter (van der Merwe et al., 2000); equally, they can be incorporated into supervisory programs that oversee the whole tracking process, as indicated above (see also Comaniciu et al., 2003).

22.7 CHAMFER MATCHING, TRACKING, AND OCCLUSION

As we have seen, one of the perennial problems of matching and tracking is that of occlusion of objects within the field of view. A variety of measures can be applied to make single camera systems as robust as possible against overlap. Leibe et al. (2005) have devised methods based on chamfer matching and segmentation, together with a minimum description length procedure for hypothesis verification. The latter evaluates hypotheses in terms of the savings that can be made by explaining part of the image by the hypotheses. Here, we concentrate on the concept of chamfer matching, as it has achieved considerable use for matching pedestrians, notably by Gavrila (1998, 2000).

The basic idea behind chamfer matching relates to the process of matching objects to templates via their boundaries—a strategy that should be much less computation intensive than matching via whole object regions. However, since this would not give much indication of a potential match until very close to the match position, some means is required of making the approach to a match far smoother. This should also permit substantial speedup of the process by employing a hierarchical coarse-to-fine search. To achieve a smoother transition, edge points in the image are first located, and then a distance function image is generated, starting with the edge points, which are initialized to zero distance values. Application of the template, also in the form of edge points, will ideally yield a zero sum (of image distance function values) along the template points: this will rise to a higher value when the template is misplaced or the shape of the object is distorted, corresponding to the sum of distances of each image point from the ideal position. Taking the distance function as $DF_I(i)$, we can express the degree of match by the average “chamfer” distance, i.e. the average distance from each edge point to the nearest edge point in the template T :

$$D_{\text{chamfer}}(T, I) = \frac{1}{N_T} \sum_{i=1}^{N_T} DF_I(i) \quad (22.43)$$

where N_T is the number of edge points within the template. $D_{\text{chamfer}}(T, I)$ is actually a dissimilarity measure, having a value of zero for a perfect match.

⁵These points about use of Kalman filters are well illustrated by Nummiaro et al. (2003) in relation to a quite different scenario—that of a bouncing ball.

In fact, there is no necessity to take edge points for the image and the template: corner points or other feature points can be utilized, and the method is quite general. However, the method works best when the point set is sparse, so that (a) accurate location is achieved, and (b) computation is reduced. On the other hand, reducing the number of points too far will result in lack of sensitivity and robustness as parts of the image and template will not be adequately represented.

As it stands, this approach is limited because any outliers (caused by occlusion or segmentation errors, for example) will lead to substantial matching problems. To limit this problem Leibe et al. (2005) used a truncated distance for matching:

$$D_{\text{chamfer}}(T, I) = \frac{1}{N_T} \sum_{i=1}^{N_T} \min(DF_I(i), d) \quad (22.44)$$

with a suitable empirical value of d . On the other hand, Gavrila (1998) applied an order-based method for limiting the number of interfering distance function values, taking the k th of the ordered values (1 to N_T) as the solution value:

$$D_{\text{chamfer}}(T, I) = \arg \text{order}_k^{i=1:N_T} DF_I(i) \quad (22.45)$$

When applying this formula, it may seem attractive to use the median value, for which $k = \frac{1}{2}(N_T + 1)$. However, it can easily happen that a large proportion of the template area is obscured, so we usually need to take a smaller value of k (e.g. $0.25N_T$) that reflects this. In fact, this will reduce accuracy when none of the template is obscured, so, in the end, Eq. (22.44) might give a more useful result. We take this discussion no further here as a lot depends on the type of data that is involved. In passing, it is worth observing that when $k = N_T$, Eq. (22.45) gives the well-known Hausdorff distance (Huttenlocher et al., 1993):

$$D_{\text{chamfer}}(T, I) = \max_{i=1:N_T} DF_I(i) \quad (22.46)$$

(This formula for the Hausdorff distance may appear different from the usual one that involves a max–min operation; however, as computation of a distance function involves taking local minima of possible distances (see Chapter 9), there is concurrence in the two formulations.)

Note that, in the foregoing discussion, the distance function of the image is used rather than that of the template. This is because in practical situations many templates will have to be applied in order to cover expected variations in the objects being detected. For example, if the method is being applied for pedestrian detection, various sizes, poses, positions of limbs, and types of clothing will have to be allowed for, as well as variations in the background and possible overlaps. In these circumstances it is far more efficient to use DF_I than DF_T . Gavrila (1998) showed with considerable success how all the variations listed above can be dealt with and how the method can be made to work well to detect pedestrians.

Finally, returning to the work of Leibe et al. (2005), limitations of the chamfer matching technique were compensated by using segmentation information. This

meant obtaining a similarity function from the chamfer distance (which is a dissimilarity measure), and then combining with a Bhattacharyya coefficient representing overlap with the hypothesized segmentation $\text{Seg}_I(i)$ to produce an overall similarity measure:

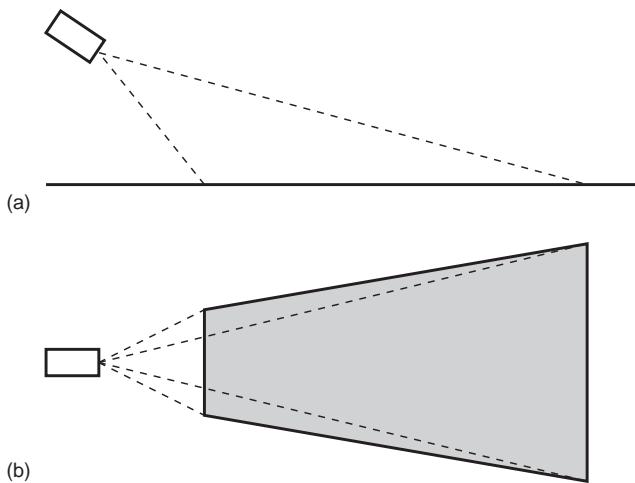
$$S = a \left[1 - \frac{1}{b} D_{\text{chamfer}}(T, I) \right] + (1 - a) \sum_i \sqrt{\text{Seg}_I(i) R_T(i)} \quad (22.47)$$

Here, $R_T(i)$ is the region within T , and the sum covers the pixels in this region. In addition, a somewhat arbitrary but nonetheless reasonable pair of weights is applied to balance the two similarity measures: a is the proportion of the overall similarity allotted to chamfer matching, and b is a weight expressing the fact that chamfer matching is applied over a significant boundary distance; in the work of Leibe et al. (2005), a and b were taken to be 0.45 and 50, respectively. The overall effect was to produce much improved solutions in respect of placement accuracy and elimination of false positives, relative to the chamfer distance method taken on its own (Eq. (22.44)).

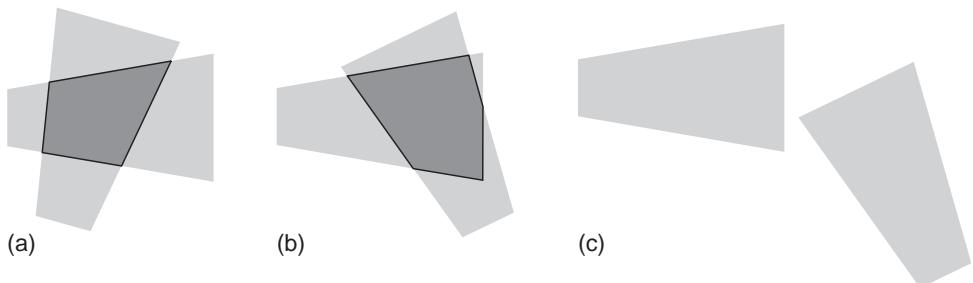
22.8 COMBINING VIEWS FROM MULTIPLE CAMERAS

Over the past decade or so there has been a surge of interest in multi-camera surveillance systems. Multiple cameras are clearly necessary if, e.g. long stretches of motorway are to be monitored, or if pedestrians are to be tracked around cities or shopping precincts. The field of view (FOV) of a single camera is quite restricted and the resolution available for viewing in the distance will almost certainly be inadequate for detailed observation. Another reason for the use of several cameras is that of viewing in stereo and obtaining sufficient depth information. A further reason is that pedestrians in a precinct will frequently be partially or wholly occluded by architectural features such as statues or other pedestrians, but the chance of missing a pedestrian will be much less if the scene is viewed by multiple cameras; this sort of situation will also apply on roads, where many other possibilities for occlusion exist.

On roads, cameras are often mounted on overhead gantries, and maintaining observation over long distances will require many cameras. This raises the question of whether the observation should be unbroken, i.e. whether the cameras will have overlapping, contiguous, or nonoverlapping views. On motorways, cameras may be separated by several miles, and can usefully be sited at junctions, so it will be possible to keep track of all vehicles without too much expense, though breakdowns at intermediate locations may not be observed. On the other hand, in a shopping precinct, if pedestrians are to be monitored closely enough for attacks or terrorist activities to be detected, contiguous, or overlapping views will be mandatory. In fact, there will be a problem in ensuring that all pedestrians are positively identified as they progress from one FOV to the next: to facilitate this, and for ease of setting up the system, overlapping views are normally required.

**FIGURE 22.9**

Area on the ground plane viewed by a camera. (a) Side view with the camera canted slightly downwards. (b) Plan view of the symmetrical trapezium seen by the camera on the ground plane.

**FIGURE 22.10**

Areas on the ground plane viewed by multiple cameras. (a) Overlapping trapezia forming a quadrilateral. (b) Overlapping trapezia forming another shape—here a pentagon. (c) Trapezia that do not overlap, though tracking across the gap can in some cases be achieved by making spatial and temporal correspondences (see text).

Next we consider the layout of a multi-camera system. To do this we must examine the area of the ground plane that lies within the FOV of the camera. First, note that the optical axis of the camera passes through the center of the image plane and that the latter has a rectangular shape given by the minimum and maximum values of x and y , $\pm x_m$ and $\pm y_m$. The FOV is therefore limited by four planes, at horizontal and vertical angles $\pm \alpha$ and $\pm \beta$, where $\tan \alpha = x_m/f$ and $\tan \beta = y_m/f$, f being the focal length of the camera lens. Each plane will

intersect with the ground plane in a line, and for a camera with a horizontal x -axis, the viewed area on the ground plane will be a symmetrical trapezium (Fig. 22.9). However, following on from the discussion in Section 22.2, if the camera is not inclined slightly downward, the distant side of the trapezium will not be visible. Since this would not make the most of the camera FOV, we will assume that it has been arranged for the distant side to fall on the ground plane.

When an adjacent camera views an adjacent section of the ground plane, there are two possibilities: (1) it will view the next stretch in the same direction, as on a motorway; (2) it will not be restricted to lie, or point, in the same direction, but just to overlap in some convenient way. For example, in a precinct or park a typical placement would be as shown in Fig. 22.10(a), where two opposite sides of the common viewing area would arise from the FOV of the first camera and the other two from that of the second camera—thereby forming a quadrilateral rather than a trapezium. However, other situations are possible, as shown in Fig. 22.10(b), where the trapezia of the two cameras overlap in a more complex way, and the common viewing area is not a quadrilateral.

No matter which of the reasons for using a multi-camera system apply, there is a need to relate the views from the separate cameras in order to obtain a consistent labeling of the objects passing between them. The obvious means of achieving this is by appearance, i.e. to apply recognition algorithms to establish that the same person or vehicle is being tracked across the various camera fields of view. Unfortunately, while this correspondence problem can normally be solved straightforwardly in binocular vision, when the two cameras are close together and pointing in a similar direction, this is by no means true for wide baseline cases such as those shown in Fig. 22.10. This is so for two reasons: (1) a person seen in two disparate views may have an altogether different appearance, e.g. the face may be visible in one and the back of the head in the other; or the back of a shirt may have a different design or color from the front; (2) the illumination may be quite different for each of the views, and this will make it even more difficult to confirm the person's identity from the other camera.

The obvious solution to this problem is to confirm identity not by appearance but by position and time. If we know that person P is at position X in the scene at time t , this must be the case in all views. So all that has to be done is to relate the common areas of the ground plane uniquely between cameras. Following the widely used and usually sufficiently accurate assumption that everything is happening on the same flat ground plane, we only need to set up a homography between the two cameras to arrange for the same correct interpretation from any view. Under perspective projection, it requires a minimum of four common feature points to set up a homography (the number is as small as this because of the planar constraint, as is made clear in Table 16.1), though more points can be used to improve accuracy; note also that at least one more point is needed to validate the homography.

In the work of Calderara et al. (2008) greater accuracy was achieved by finding the straight lines bounding the common quadrilateral and using its corners as

highly accurate points by which to define the homography. While this might seem trivial, in fact the common quadrilateral has to be located by experiment. This can be achieved most easily when the scene is empty (e.g. overnight), and one individual can be sent to walk repeatedly around the site until a sufficient number of boundary points—as determined by the individual entering or leaving one of the fields of view—have been measured in both views. Note that to ensure that this gives sound results, temporal synchronization of the two camera systems is crucial. Once all this has been carried out, methods such as Hough transforms or RANSAC are applied to collate the boundary points into the straight lines bounding the quadrilateral: and because of the averaging inherent in this process, the straight lines will be known accurately; therefore, the corner positions will be known accurately, so there will be no need to use more points to establish an accurate homography.

Interestingly, Khan and Shah (2003) consider this approach an overkill to solve the consistent labeling problem. They assert that there is no need to determine the homography in this numerical sort of way: rather, it should be done by finding the FOV boundary lines and then merely noting when a pedestrian passes over one of these lines and making the identity at that point in time, i.e. if an individual crosses a line at time t , this will be detected at the same time t in each camera and the person's identity can be passed across at that moment. This process is commonly called camera “handoff” (whereas it might appear to be more natural to call it “handover,” there is a subtlety in that the latter term would tend to imply that the fields of view are contiguous rather than overlapping). However, if a group of people all cross the line together, this could obviously give rise to difficulties. Indeed, the whole problem of tracking groups of individuals is a difficult one, and becomes almost insuperable in dense crowd situations.

While finding FOV boundary lines can be carried out when no crowds are present, and ideally when a single individual walks around, there are limits to the performance of the trained system. This is because a homography relates to a plane, and the simplest way of defining and using a plane is to use the foot locations to provide the plane contact points. (In principle, this is easily done by taking the lowest point on the individual.) However, when the calibrated system is used, the feet of one individual will often be obscured by another individual—a situation that will be virtually unavoidable in crowds. Consequently, there has been a fair amount of attention to recognizing and locating individuals from the tops of their heads (e.g. Eshel and Moses, 2008, 2010). Clearly, tops of heads are much less likely than feet to be occluded. Hence, even in crowd conditions, as long as cameras are quite high up and canted down at quite high angles (say 40°), all but the shortest individuals should be identifiable. Interestingly, apart from orientation, tops of heads may actually look similar in different views. As the camera cant angle will be known, altered head orientation can be allowed for and recognition and cross identification between cameras can proceed. With fully calibrated cameras (see Chapter 18), tops of heads can be located in 3-D space, and the positions of feet and heights of individuals can be deduced. Unfortunately,

full camera calibration is a tedious process and may need frequent updating, so it is better not to rely on that approach in “informal” (and therefore changeable) surveillance situations such as shopping centers. Instead, camera views can be related using the fundamental matrix formulation (Chapter 18), which only requires that epipoles should be known so that epipolar lines can be determined; however, finding them requires considerable computation, though this can be done offline prior to actual use (Calderara et al., 2008).

An intriguing approach to top-of-head location is to try various homographies differing only in the parameter H signifying distance from the floor. When a homography is found that indicates the same value of H , the foot locations can be calculated for each camera view, even though the feet themselves are obscured. However, to achieve this a somewhat complex and subtle process is required (Eshel and Moses, 2008, 2010). Four vertical poles are set up at the corners of each viewing quadrilateral (or other convenient location), each pole having three bright lights along it (e.g. at the top, bottom and middle of the pole). Then standard homographies are set up for each of these, so that at any location in an image, three heights can be deduced. Finally, a height that is to be measured can be related to the three known ones for that location, a cross-ratio calculated along a vertical line, and the actual height deduced; at the same time the foot position in each camera view can be identified unambiguously.

Overall, the simplest and most powerful approach is that of prior training by getting someone to walk around the site and thus demarcate the boundaries of each common viewing zone. Then, applying the fundamental matrix for pairs of cameras will permit homographies to be set up relating all the mutually viewable regions of ground planes. The paper by Calderara et al. (2008) contains a number of other subtleties, but space prevents them from being described in detail here. Finally, if heights and exact locations of people are to be found from top-of-head positions, elegant though fairly complex methods using several homographies have to be used, but in some applications, such as observation of crowds, the additional complexity may well be justified. However, segmentation of crowd views and identification of all individuals remains a research topic, especially when the people are tightly packed—as can easily happen in metro stations and football matches.

22.8.1 The Case of Nonoverlapping Fields of View

Next we move on to the case of nonoverlapping fields of view. Here there seems to be no basis for homographies or for reliable camera handoff. However, some degree of similarity in appearance will still be detectable between views; in addition, there will be strong correlations between the time of leaving one FOV and arriving in another. The situation will often be helped if there is some restriction of access, such as would occur if there is a single adjoining door. (On a motorway, there is anyway such a restriction, and temporal correlations can be strong.)

Pflugfelder and Bischof (2008) have obtained significant success in this sort of situation, and made no assumptions about appearance. In particular, they have found how to relate the camera calibration matrices when overlapping views are not available. While this seems intrinsically impossible because no common image points can be found and hence no equations can be obtained linking the parameters (recall that the 8-point algorithm requires eight points in order to obtain a sufficient number of equations), they have found that if velocities are assumed to be more or less constant across the intervening space, this provides the continuity needed to permit enough equations to be found. Thus, a minimum of two positions per view for each trajectory is sufficient, these being immediately before and after camera handoff. Strict temporal correspondences are required, as is data on relative camera orientations, but a common ground plane is not assumed. Under these conditions, tracking across gaps of up to 4 m was achieved (Fig. 22.10(c)). The method works by making use of Rother and Carlsson's (2001) 2-point technique for determining the relative positions of two cameras with overlapping views: the new method simulates this situation by utilizing a separate pair of points in the second nonoverlapping view in order to emulate and replace the two points that would ideally have been present in an overlapping view.

For a differently motivated probabilistic strategy tackling this problem, based on transition probabilities between nonoverlapping views, see Makris et al. (2004): what is special about this approach is that it is quite general as it is entirely unsupervised and has no direct knowledge of camera placement or camera characteristics.

22.9 APPLICATIONS TO THE MONITORING OF TRAFFIC FLOW

22.9.1 The System of Bascle et al.

One important area of surveillance is the visual analysis of traffic flow. In an early study (Bascle et al., 1994) it was found that the complexity of the analysis was reduced because vehicles run on the roadway and because their motions are generally smooth. Nevertheless, the methods that had to be used to make scene interpretation reliable and robust were nontrivial.

First, motion-based segmentation is used to initialize the interpretation of the sequence of scenes. The motion image is used to obtain a rough mask of the object, and then the object outline is refined by classical edge detection and linking. B-splines are used to obtain a smoother version of the outline, which is fed to a snake-based tracking algorithm. The latter updates the fit of the object outline and proceeds to repeat this for each incoming image.

However, snake-based segmentation concentrates on isolation of the object boundary, and therefore ignores motion information from the main region of the object. It is therefore more reliable to perform motion-based segmentation of the entire region bounded by the snake, and to use this information to refine the description of the motion and to predict the position of the object in the next

image. The overall process is thus to feed the output of the snake boundary estimator into a motion-based segmenter and position predictor that re-initializes the snake for the next image—so both constituent algorithms perform the operations they are best adapted to. It is especially relevant that the snake has a good starting approximation in each frame, both to help eliminate ambiguities and to save on computation. The motion-based region segmenter operates principally by the analysis of optical flow, though in practice the increments between frames are not especially small: this means that while true derivatives are not obtained, the result is not as bedevilled by noise as it might otherwise be.

Various refinements were incorporated into the basic procedure:

- B-splines are used to smooth the outlines.
- The motion predictions are carried out using an affine motion model that works on a point-by-point basis. (The affine model is sufficiently accurate for this purpose if perspective is weak so that motion can be approximated locally by a set of linear equations.)
- A multi-resolution procedure is invoked to perform a more reliable analysis of the motion parameters.
- Temporal filtering of the motion is performed over several image frames.
- The overall trajectories of the boundary points are smoothed by a Kalman filter.⁶

The affine motion model used in the algorithm involves six parameters:⁷

$$\begin{bmatrix} x(t+1) \\ y(t+1) \end{bmatrix} = \begin{bmatrix} a_{11}(t) & a_{12}(t) \\ a_{21}(t) & a_{22}(t) \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} b_1(t) \\ b_2(t) \end{bmatrix} \quad (22.48)$$

This leads to an affine model of image velocities, also with six parameters:

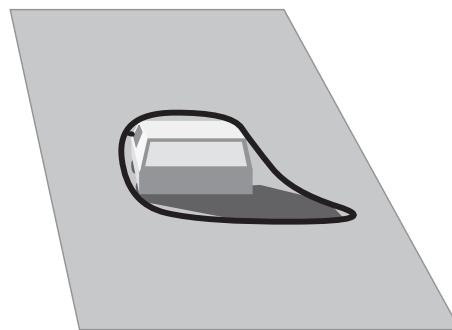
$$\begin{bmatrix} u(t+1) \\ v(t+1) \end{bmatrix} = \begin{bmatrix} m_{11}(t) & m_{12}(t) \\ m_{21}(t) & m_{22}(t) \end{bmatrix} \begin{bmatrix} u(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} c_1(t) \\ c_2(t) \end{bmatrix} \quad (22.49)$$

Once the motion parameters have been found from the optical flow field, it is straightforward to estimate the following snake position.

An important factor in the application of this type of algorithm is the degree of robustness it permits. In this case, both the snake algorithm and the motion-based region segmentation scheme are claimed to be relatively robust to partial occlusions: the abundance of available motion information for each object, the insistence on consistent motion, and the recursive application of smoothing procedures including a Kalman filter, all help to achieve this end. However, no specific

⁶A basic treatment of Kalman filters is given in Section 19.8.

⁷An affine transformation is one which is linear in the coordinates employed. This type of transformation includes the following geometric transformations: translation, rotation, scaling and skewing (see Chapter 18). An affine motion model is one which takes the motion to lead to co-ordinate changes describable by affine transformations.

**FIGURE 22.11**

Vehicles located with their shadows. In many practical situations, shadows move with the objects that cause them, and simple motion segmentation procedures produce composite objects that include the shadows. Here a snake tracker envelops the car and its shadow.

nonlinear outlier rejection process is mentioned, which could help if two vehicles merged together and became separated later on or if total occlusion occurred.

Finally, the initial motion segmentation scheme locates the vehicles with their shadows since these are also moving (see Fig. 22.11); subsequent analysis seems able to eliminate the shadows and arrive at smooth vehicle boundaries.

22.9.2 The System of Koller et al.

Another scheme for automatic traffic scene analysis was described by Koller et al. (1994). This contrasts with the system described above by placing heavy reliance on high-level scene interpretation through the use of belief networks. The basic system incorporates a low-level vision system employing optical flow, intensity gradient, and temporal derivatives. These provide feature extraction, and lead to snake approximations to contours; since convex polygons would be difficult to track from image to image (because the control points would tend to move randomly), the boundaries are smoothed by closed cubic splines having 12 control points; tracking is then achieved using Kalman filters. The motion is again approximated by an affine model, though in this case only three parameters are used, one being a scale parameter and the other two being velocity parameters:

$$\Delta \mathbf{x} = s(\mathbf{x} - \mathbf{x}_m) + \Delta \mathbf{x}_m \quad (22.50)$$

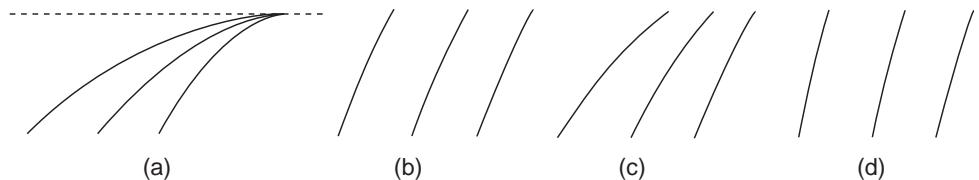
Here the second term gives the basic velocity component of the center of a vehicle region, and the first term gives the relative velocity for other points in the region, s being the change in scale of the vehicle ($s = 0$ if there is no change in scale). The rationale for this is that vehicles are constrained to move on the roadway, and rotations will be small. In addition, motion with a component toward the camera will result in an increase in size of the object and a corresponding increase in its apparent speed of motion.

Occlusion reasoning is achieved by assuming that the vehicles are moving along the roadway, and are proceeding in a definite order, so that later vehicles (when viewed from behind) may partly or wholly obscure earlier ones. This depth ordering defines the order in which vehicles are able to occlude each other, and appears to be the minimum necessary to rigorously overcome problems of occlusion.

As stated above, belief networks are employed in this system to distinguish between various possible interpretations of the image sequence. Belief networks are directed acyclic graphs in which the nodes represent random variables and arcs between them represent causal connections. In fact, each node has an associated list of the conditional probabilities of its various states corresponding to assumed states of its parents (i.e. the previous nodes on the directed network). Thus, observed states for subsets of nodes permit deductions to be made about the probabilities of the states of other nodes. The reason for using such networks is to permit rigorous analysis of probabilities of different outcomes when a limited amount of knowledge is available about the system. Likewise, once various outcomes are known with certainty (e.g. a particular vehicle has passed beneath a bridge), parts of the network will become redundant and can be removed; however, before removal their influence must be “rolled up” by updating the probabilities for the remainder of the network. Clearly, when applied to traffic, the belief network has to be updated in a manner appropriate to the vehicles that are currently being observed; indeed, each vehicle will have its own belief network that will contribute a complete description of the entire traffic scene. However, one vehicle will have some influence on other vehicles, and special note will have to be taken of stalled vehicles or those making lane changes. In addition, one vehicle slowing down will have some influence on the decisions made by drivers in following vehicles. All these factors can be encoded into the belief network and can aid in arriving at globally correct interpretations. General road and weather conditions can also be taken into account.

Further work was planned to enable the vision part of the system to deal with shadows, brake lights and other signals, and a wide enough variety of weather conditions. Overall, the system was designed in a very similar manner to that of Bascle et al. (1994), though its use of belief networks made it rather more sophisticated.

In a later version of the system (Coifman et al., 1998), it was decided that a greater degree of robustness with regard to partial occlusion was required. Hence, the idea of tracking objects as a whole was abandoned and corner features were used for detection. This led to a different problem—that of grouping corner features to infer the presence of the vehicles, a process that was simplified by using a common motion constraint, so that features that were seen to be rigidly moving together were grouped together. The new version of the system also applied a homography between the image plane and the ground plane. The reason for this was to generate world parameters so that ground-based positions, trajectories, velocities, and densities could be established. Note for example that

**FIGURE 22.12**

Adjusting the inverse perspective mapping of the roadway. (a) shows the roadway as observed by the camera. (b) shows an inverse perspective mapping with the roadway adjusted for constant width. (c) and (d) show cases of incorrect adjustment of the mapping.

a vehicle traveling at constant speed on the road would have variable speed when viewed in an image. In addition, the right information could more easily be brought to bear when problems of partial or total occlusion are being investigated.

When designing a much later system, Magee (2004) made several interesting observations: (a) corner features are unreliable because of the small size of the objects of interest; (b) connected components analysis is a poor tool for combining parts of vehicles because of fragmentation and similarity of some object foreground points to background; (c) particle filter trackers have high computational cost that does not scale linearly with the number of objects present—a serious matter when 30 or more vehicles in close proximity are being tracked simultaneously. He found that a sound way to track vehicles was to dynamically model vehicle invariants such as size, color, and speed: in other words, object appearance and recognition were important to systematic and accurate tracking; and the only way they could be achieved was by establishing a homography between the image and the ground plane. In that way vehicle parameters properly became invariants as required. The homography is expressible as a nonlinear perspective transformation (or “inverse perspective mapping”),⁸ and some care is required in setting it up. However, if the camera x -axis is horizontal, the homography only requires a rotation through an angle θ about the image x -axis, together with a scaling, in order to relate the image coordinates to the ground plane coordinates. Ignoring the scaling, there is only one parameter (θ) to be determined. Magee adopted the simple strategy of estimating θ as the angle required to make the roadway appear to have constant width, a procedure that proved to be adequate in his particular application (Fig. 22.12). The calculation was made sufficiently accurate by approximating the road centerlines and outlines by three polynomials and performing a fit by iteratively adjusting θ . The reason for adopting this procedure is

⁸Note that such a mapping is mathematically valid only for points known to lie on the ground plane. When points not lying on the ground plane are back-projected to it, they give rise to weird, nonsensical effects, such as buildings that appear to lean backwards.

that the roadway has no absolute predefined shape so a heuristic approach seemed appropriate. Ideally, however, the ground truth for the road centerlines and outlines would be known and the value of θ could be adjusted to fit the ground truth without having to assume that the roadway has constant width.

22.10 LICENSE PLATE LOCATION

Over the past decade there has been intensive effort to identify vehicles automatically by their license plates. Although license plates were introduced many years ago for the purpose of checking ownership and detecting stolen vehicles, nowadays two other important reasons for automatically identifying vehicles are (1) for taxation within tolling zones and (2) for exacting fines in the case of parking offences—because considerable sums of money can be obtained in these ways with very little human intervention. Also, considering all the possible applications of computer vision in surveillance, identification of license plates represents a potentially straightforward application of current methodology. Nevertheless, there are many problems, not least because of the different styles of license plate from different countries.

Identification of license plates progresses through three main stages: (1) location and segmentation of the license plate; (2) segmentation of the individual characters; (3) recognition of the individual characters. Here, we concentrate on the first of these stages, as the other two are more specialized and less generic, considering the different styles, fonts, and character sets in use in different countries. In any case, the first stage is probably the most difficult to engineer.

A priori, it might be thought that the best way of locating license plates would be via their colors, which are generally well specified for each country. However, many problems arise from variations in ambient lighting, particularly with the seasons, the weather, and the time of day, while shadows are also a source of difficulty. In this milieu, one of the best starting points has been found to be use of a simple Sobel or other vertical edge detection operator, in conjunction with horizontal nonmaximum suppression and thresholding. This has been found to locate not only the vertical lines at the ends of the number plates but also the vertical lines at the sides of the characters (Zheng et al., 2005). This generally gives a relatively dense set of vertical edges within the region of the license plate. To proceed further, long background edges and short noise edges are eliminated. Finally, moving a rectangle of license plate size over the image and counting the edge pixels within it turns out to be a highly reliable way of locating license plates (in fact, this process is a form of correlation). The whole process is shown in Fig. 22.13, with the difference that in the case shown, the final stages are carried out solely using morphological operations (horizontal closing followed by horizontal opening, in each case by 16 pixels).

**FIGURE 22.13**

Simple procedure for locating license plates. (a) Original image with license plate pixelated to prevent identification. (b) Vertical edges of original image. (c) Vertical edges selected for length. (d) Region of license plate located by horizontal closing followed by horizontal opening, each over a substantial distance (in this case 16 pixels).

This method has been developed considerably further by Abolghasemi and Ahmadyfard (2009) using color and texture cues. They found that a particular advantage of color object analysis is robustness to viewpoint changes. They also used morphological closing to link all the vertical edge points, and followed this by opening to eliminate the effects of isolated noise points.

Before characters can be segmented and recognized, another stage is needed—that of license plate distortion correction. This arises because license plates may not be observed from the most ideal viewpoint. This is something that requires careful attention. If vehicles are too far away from the camera, the resolution will be too low to permit vertical edges to be found; likewise, accurate identification of the characters will not be possible. If the license plate is viewed obliquely it will appear misorientated and will not even appear rectangular. However, if license plates were always viewed at a particular distance and location, a standard perspective transformation could be applied to correct such distortions. While it is acknowledged in the literature (e.g. Chang et al., 2004) that adding such a step would improve the performance of license number recognition, few systems seem to incorporate such a step. The reason is probably that OCR systems are already very accurate even when characters are slightly sheared and rotated.

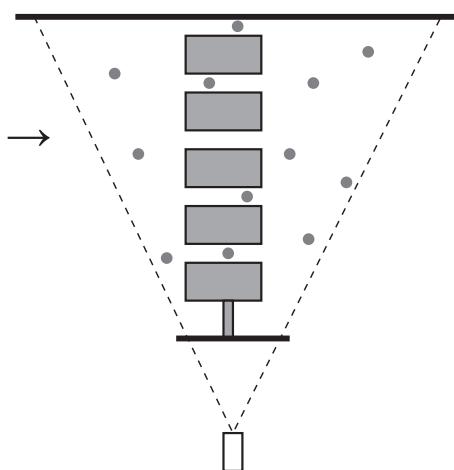


FIGURE 22.14

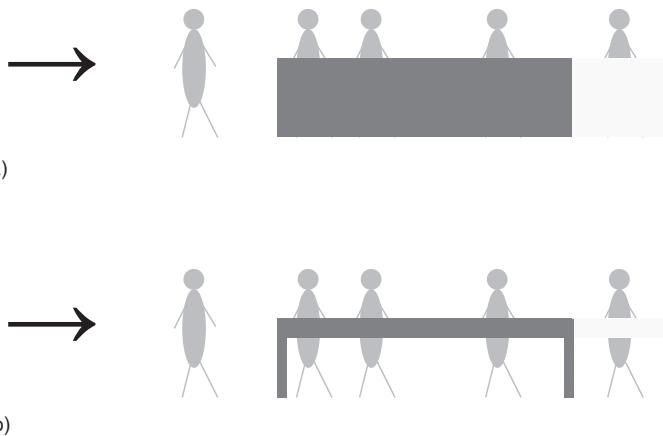
Typical situation of occlusion. This illustrates the case of turnstiles leading to underground trains, viewed from the side. The dots represent people (moving in the direction of the arrow) ranged at different distances from the camera.

22.11 OCCLUSION CLASSIFICATION FOR TRACKING

It will be clear from the many remarks made about occlusion on the preceding pages that this is a serious problem that needs in-depth analysis and careful algorithm design, particularly with regard to people tracking. To this end, Vezzani and Cucchiara (2008), and Vezzani et al. (2011) have made a careful analysis of the means by which occlusion can arise, starting with the definition of *nonvisible regions* as the parts of objects that are not visible in the current frame. They proceeded to classify these as “dynamic,” “scene,” or “apparent” occlusions:

1. *Dynamic occlusions* are due to moving objects that are readily identified.
2. *Scene occlusions* are due to static objects that are part of the background, but can nevertheless be in front of moving objects.
3. *Apparent occlusions* are sets of pixels that arise from shape variations of objects being tracked.

Here it is important to note the distinction between background and foreground. To the layman, “background” merely means a backdrop in front of which the actors perform: it is regarded as static, while the foreground is considered to consist of more interesting moving objects. However, in computer vision we have to consider the background as static wherever it is, with the moving “foreground” objects ranged at different distances from the camera and sometimes moving *behind* background objects (Fig. 22.14). Note that the background that is identified by background modeling algorithms is the static part of the scene. Of course, a complication that can disrupt this tidy situation is that the background may be

**FIGURE 22.15**

Further examples of occlusion. (a) Case of people walking behind a fence or barrier, potentially resulting in only the head and shoulders being tracked afterwards. (b) Case of people walking behind a table, potentially resulting in two parts of the body being tracked independently afterwards.

composed partly of objects that have come to rest, either permanently or temporarily, and it will be up to the vision algorithm to consider the available evidence from watching the scene and to assess various possibilities and probabilities.

Another factor to consider is whether occlusions are partial or total. For many static scenes and static situations, total occlusion is an eventuality that is normally disregarded; hence, all occlusions are taken to be partial, and they are simply referred to as “occlusions.” However, when tracking objects, total occlusion is a possibility that has to be borne in mind indefinitely (though in practical situations a time-limit may have to be set).

So, when viewing image sequences containing motion, objects may temporarily be totally occluded, *or* they may be partially occluded—in which case they may be broken into several sections. And when objects re-emerge later on, these sections need to be reassembled into whole objects: this scenario arises when a person passes behind a table, for example. In addition, when a person passes behind a low fence, and the lower body is temporarily invisible, it is necessary for the model of the complete person to be remembered; for if the model becomes adapted to the changed situation, it may not be able to cope properly when the whole person re-emerges and it will continue to track only the top of the body. Clearly, to cope successfully with such situations (Fig. 22.15), the computer needs to have the means to deal with them holistically. Similarly, when two people merge into a larger blob when walking together, the computer will need to have the means to recall that two people were involved so that their identities will be preserved and reinstated when they separate again. Thus, we require substantial intelligence to be incorporated into tracking algorithms.

The various components that have to be incorporated into the algorithm would appear to be the following: (a) the usual background extraction capability, (b) the usual blob tracking capability, (c) full appearance and identity recall, (d) merge capability, (e) split capability, and (f) probabilistic analysis of interpretations. Indeed, (f) will probably have to be the unifying force that drives the whole algorithm.

All these aspects are included in the work of Vezzani and Cucchiara (2008) and Vezzani et al. (2011). In particular, they employ an appearance-based formalism that integrates the possible shape variations of each object and represents them by probabilistic maps. This means that when part of an object is obscured, its shape model hallucinates the whole of the object in the probabilistic shape it ought to have, so that when it re-emerges it is automatically re-integrated virtually instantaneously into its natural form.

So far we have not examined item 3 (see the beginning of this section) that mentioned apparent partial occlusions due to changes in shape. These arise because as a body rotates or bends slightly, or otherwise deforms, new parts will become visible though other parts will become invisible. While these could be regarded as arising through self-occlusion, this may not be the only possibility, e.g. if stretching is involved. We shall not delve further into this point, but merely underline that apparent partial occlusions are not caused by any other objects. This is quite an innovative observation relating to occlusion, and may be part of the reason why progress with occlusion has been drawn out over many years. Suffice it to say that the work of Vezzani et al. represents a sound and impressive advance through its cognizance of the many aspects of occlusion in tracking scenarios.

The overall system is very robust and fast and is well able to cope with more than 40 people in videos from the PETS2006 dataset. Nevertheless, it gives rise to some failures that devolve into the following categories: (a) identity change of one person, (b) split head/feet, (c) incorrect splitting of groups containing two or three people, (d) identity change of luggage. In fact, it appears that these are failures not of the *overall* system, including the handling of objects and occlusions, but of the part of the system handling appearance, which is arguably the part to which relatively little design effort has been devoted. Furthermore, it is not clear from the two papers whether a human observer could have performed better using the same video input. Nevertheless, the way forward, including the ability to handle problems (b) and (c) above, is probably to enhance the system using stick-figure based models of humans, which can take proper account of limb articulation constraints (see Section 22.13): the performance of a system that looks at the body as a whole and models it as a holistic probabilistic shape profile must in the end be limited without suitable enhancement.

22.12 DISTINGUISHING PEDESTRIANS BY THEIR GAIT

This section outlines a method for distinguishing pedestrians by their gait. Clearly, unlike many other moving objects such as vehicles, pedestrians have

FIGURE 22.16

Portions of frames extracted from video sequences by motion detector. Left: Three frames of moving vehicle. Right: Respective frames of pedestrian, runner and group of walkers.

Source: © IET 2007

cyclical motions, and it is actually possible to recognize individual people by their gait. However, here we consider only the methodology needed to locate pedestrians in image sequences.

The basis of the approach is to perform spatiotemporal differencing operations, in which spatiotemporal averaging is followed by temporal differencing. This “motion distillation” method (Sugrue and Davies, 2008) is implemented as a Haar wavelet and leads to a nonbinary motion map of the video at each time-step, according to the equation:

$$W = \sum_{t=t_0}^t \sum_i \sum_j x_{tij} - \sum_{t=t_1}^t \sum_i \sum_j x_{tij} \quad (22.51)$$

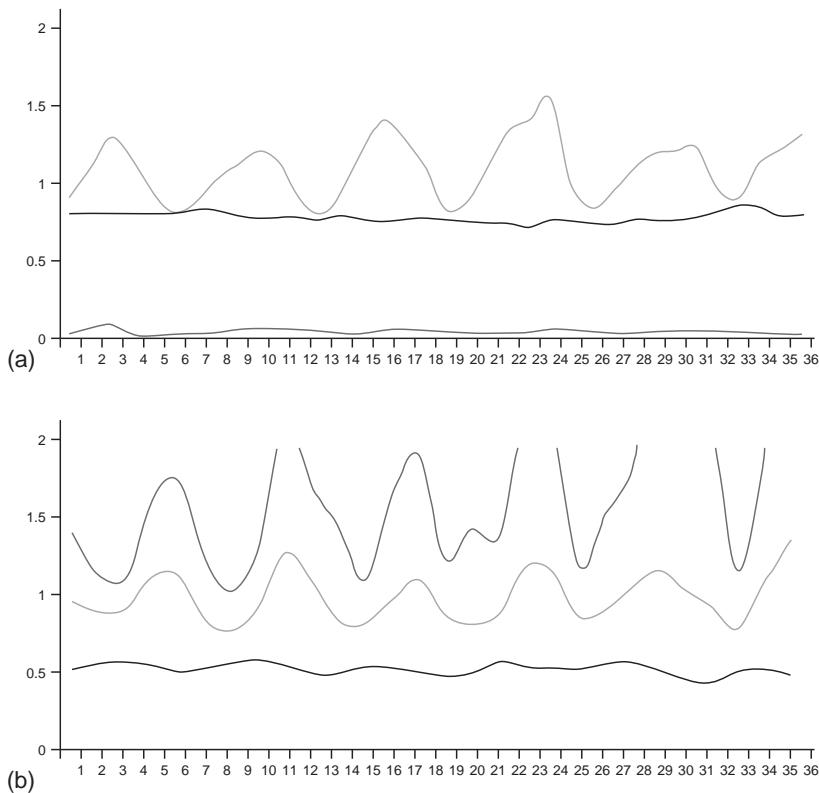
where x_{tij} represents the video pixel data at the point (t, i, j) in spatiotemporal space.

In this method, undesirable contrast dependence is removed by normalizing W values across the detected object: the process involves taking the ratio R of positive (W_+) to negative (W_-) filter outputs:

$$R = \frac{\sum |W_+|}{\sum |W_-|} \quad (22.52)$$

For a rigid object that retains its orientation relative to the camera, R will remain approximately constant over time. On the other hand, pedestrians deform as they move, and can quickly be detected by testing for changes, and particularly oscillations in the “rigidity parameter” R .

Figure 22.17(a) compares the motion signals R of a typical vehicle and a pedestrian (see Fig. 22.16 for typical frames from the original videos). While the vehicle signal changes only gradually as a result of slight rotation, changing perspective and noise, the pedestrian signal is highly variable and oscillatory because of gait motion. Note that over the period shown in Fig. 22.17(a), the area of the vehicle changes by a factor ~ 10 , while the area of the pedestrian changes by only a few percent. This makes it all the more significant that R is so constant for the vehicle, demonstrating that it is a useful invariant of the motion.

**FIGURE 22.17**

Motion analysis using rigidity and box parameters. (a) Top to bottom: result of applying the rigidity parameter R for the pedestrian; result of applying R for the vehicle; result of applying box parameter η for the pedestrian. The horizontal scales indicate video frames. (b) Top to bottom: result of applying the respective parameters R_{ex} , R , η for the runner. In all cases the originals are shown in Fig. 22.16.

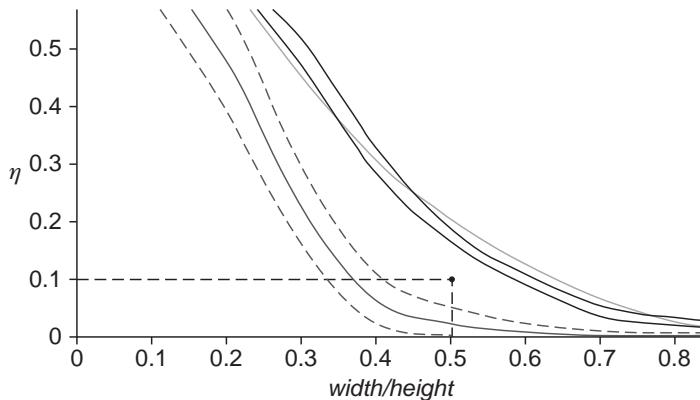
Source: © IET 2007

After detection, a pedestrian's motion field can be further analyzed for a variety of behavior patterns. Normal behavior can be modeled by fitting a rectangular box to the subject's motion field. The rectangle is the full height of the figure with a width typically set at half the height (see below). The total motion area A is calculated as:

$$A = \sum |W_+| + \sum |W_-| \quad (22.53)$$

where the sums are taken over the whole object; in addition, the corresponding area A_{ex} is calculated for the region outside the box. The box parameter η is then defined as the ratio of the two areas:

$$\eta = \frac{A_{\text{ex}}}{A} \quad (22.54)$$

**FIGURE 22.18**

Discrimination permitted by box parameter η . The lower solid line represents the mean value of a sample of walkers (the broken lines indicate $\pm\sigma$ error bars), and the upper solid lines record runners. To discriminate between walkers and runners, the best operating point is close to (0.5, 0.1), as shown.

Source: © IET 2007

This parameter should also be an invariant both for rigid motion and for “compact” motion where A_{ex} is small, giving a measure of the type of behavior: this is because η is dimensionless and compares like with like but still contrasts two motions (*viz.* exterior to the box and overall). If the pedestrian is walking normally, the η value will be low at all times (see, for example, the bottom trace in Fig. 22.17(a)): the higher values typical of runners are demonstrated clearly in Fig. 22.17(b). In addition, individual sudden actions such as waving and jumping will result in spikes in η .

A third type of invariant R_{ex} has also been developed to help discriminate other more complex cases. This has the same definition as for R , except that it applies only to the part of the object external to the box. It provides additional useful information helping to discriminate runners from groups of walkers (see Fig. 22.16). Both of these categories have been found to have values of η around 0.5, so R_{ex} is useful in enabling them to be discriminated. (Specifically, $R_{ex} \approx 1.5R$ for runners and $R_{ex} \approx R$ for groups of walkers, though additionally R and R_{ex} are only well synchronized for runners.) While the R_{ex} information cannot be described as being specific to groups, it is nevertheless valuable, though ultimately only detailed analysis leading, e.g., to stick-figure models of people may provide the information that is required in a particular application (see Section 22.13).

Because of the importance of box size in determining the values of η and R_{ex} , a careful study was made to optimize discrimination between single walkers and runners. This gave the optimum box width/height ratio as close to 0.5, for which the walker–runner threshold was best set at about 0.1 (see Fig. 22.18).

Overall, the methods described here have been found to distinguish motions of rigid and nonrigid objects with $\sim 97\%$ accuracy. They are also able to classify single walkers with $\sim 95\%$ accuracy, and runners and groups of walkers with

~87% accuracy; in addition, they give useful indications of “extravagant” activities such as waving and jumping. Interestingly, all this was achieved via use of specially designed invariants, which save complexity and computation while being straightforward to set up and adjust.

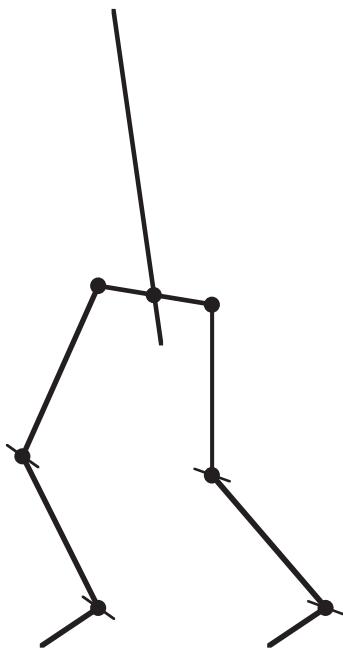
22.13 HUMAN GAIT ANALYSIS

For several decades human motion has been studied using conventional cinematography. Often the aim of this work has been to analyze human movements in the context of various sports—in particular, tracking the swing of a golf club and thus helping the player to improve his game. To make the actions clearer, stroboscopic analysis coupled with bright markers attached to the body have been employed, and have resulted in highly effective action displays. In the 1990s, machine vision was applied to the same task. At this point the studies became much more serious and there was increased focus on accuracy. The reason for this was a widening of the area of application not only to other sports but also to medical diagnosis and to animation for modern types of film containing artificial sequences.

Because high accuracy is needed for many of these purposes—not least measuring limps or other imperfections of human gait—analysis of the motion of the whole human body in normally lit scenes proved insufficient, and body markers remained important. Typically, two are needed per limb, so that the 3-D orientation of each limb is deducible. Some work has been done to analyze human motions using single cameras, but the majority of the work employs two or more cameras: multiple cameras are valuable because of the occlusion that occurs when one limb passes behind another, or behind the body.

To proceed with the analysis, a kinematic model of the human body is required. In general, such models assume that limbs are rigid links between a limited number of ball-and-socket joints, which can be approximated as point junctions between stick limbs. For example, one such model (Ringer and Lazenby, 2000) employs two rotation parameters at the point where the hips join the backbone, three for the joint where the thigh bone joins the hips, plus one for the knee and another for the ankle. Thus, each leg has seven degrees of freedom, two of these being common (at the backbone): this leads to a total of 12 parameters covering leg movements ([Fig. 22.19](#)). Clearly, it is part of the nature of the skeleton that the joints are basically rotational, though there is some slack in the system, especially in the shoulders, while the knees have some lateral freedom. Finally, the whole situation is made more complex by constraints such as the inability of the knee to extend the lower leg too far forward.

Once a kinematic model has been established, tracking can be undertaken. It is relatively straightforward to identify the markers on the body with reasonable accuracy. The next problem is to distinguish one marker from another and to label them. Considering the huge number of combinations of labels that are possible,

**FIGURE 22.19**

Stick skeleton model of the lower human body. This model takes the main joints on the skeleton as being universal ball-and-socket joints, which can be approximated by point junctions—albeit with additional constraints on the possible motions (see text). Here a thin line through a joint indicates the single rotational axis of that joint.

and the frequency with which occlusions of parts of a leg or arm are bound to take place, special association algorithms are required for the purpose. These include the Kalman filter that helps to predict how unseen markers will move until they come back into view. Such models can be improved by including acceleration parameters as well as position and velocity parameters (Dockstader and Tekalp, 2002). Their model is not merely theoretically deduced: it has to be trained, typically on sequences of 2500 images each separated by 1/30 s. In addition, the stick model of each human subject has to be initialized manually. Considerable training is necessary to overcome the slight inaccuracies of measurement and to build up the statistics sufficiently for practical application when testing. Errors are greatest when measuring hand and arm movements, because of the frequent occlusions they are subject to.

Overall, articulated motion analysis involves complex processing and a lot of training data. It is a key area of computer vision and the subject is evolving rapidly. It has already reached the stage of producing useful output, but accuracy will improve over the next few years and this will set the scene for practical medical monitoring and diagnosis, completely natural animation, detailed help with sports activities at affordable costs, not to mention recognition of criminals by their characteristic gaits. Certain requirements—such as multiple cameras—will

probably remain, though the trend to markerless monitoring can be expected to continue. For further information, the reader could start by referring to the monograph by Nixon et al. (2006).

22.14 MODEL-BASED TRACKING OF ANIMALS

This section is concerned with the care of farm animals. Good stockmen noted many aspects of the behavior of the animals, and learnt to respond to them. Fighting, bullying, tail biting, activity, resting behavior, and posture are useful indicators of states of health, potential lameness, or heat stress, while group behavior may indicate the presence of predators or human intruders. In addition, feeding behavior is all-important, as is the incidence of animals giving birth or breaking away from the confinement of the pen. In all these aspects, automatic observation of animals by computer vision systems is potentially useful.

Some animals such as pigs and sheep are lighter than their usual backgrounds of soil and grass, and thus they can in principle be located by thresholding. However, the backgrounds may be cluttered with other objects such as fences, pen walls, drinking troughs, and so on—all of which can complicate interpretation. Thus, straightforward thresholding will rarely work well in normal farm scenes. McFarlane and Schofield (1995) tackled this problem by background subtraction. They used a background image obtained by temporal median filtering for a whole range of images taken over a fair period: during this process care was taken to mask out regions where piglets were known to be resting. Their algorithm modeled piglets as simple ellipses, and achieved fair success in its task of monitoring the animals.

We next examine the more rigorous modeling approach adopted by Marchant and Onyango (1995), and developed further by Onyango and Marchant (1996) and Tillett et al. (1997). These workers aimed to track movements of pigs within a pen by viewing them from overhead under not very uniform lighting conditions. The main aim of the work at this early stage was tracking the animals, though, as indicated above, it was intended to lead on to behavioral analysis in later work. To find the animals, some form of template matching is required. Shape matching is an attractive concept, but with live animals such as pigs, the shapes are highly variable: specifically, animals which are standing up or walking around will bend from side to side, and may also bend their necks sideways or up and down as they feed. It is insufficient to use a small number of template masks to match the shapes, as there is an infinity of shapes related by various values of the shape parameters mentioned. These parameters are additional to the obvious ones of position, orientation, and size.

Careful trials showed that matching with all these parameters is insufficient, as the model is quite likely to be shifted laterally by variations in illumination: if one side of a pig is closer to the source of illumination, it will be brighter, and hence the final template used for matching will also shift in that direction. The

resulting fit could be so poor that many possible “goodness of fit” criteria will deny the presence of a pig. These factors mean that possible variations in lighting have to be taken into account in fitting the animal’s intensity profile.

A rigorous approach involves principal components analysis (PCA). The deviation in position and intensity between the training objects and the model at a series of carefully chosen points is fed to a PCA system: the highest energy eigenvalues indicate the main modes of variation to be expected; then any specific test example is fitted to the model and amplitudes for each of these modes of variation are extracted, together with an overall parameter representing the goodness of fit. Unfortunately, this sort of approach is highly computation intensive because of the large number of free parameters; in addition, the position and intensity parameters are disparate measures that require quite different scale factors to be used to coax the schema into working. This means that some means is required for decoupling the position and intensity information. This is achieved by performing two independent principal components analyses in sequence—first on the position coordinates and then on the intensity values.

When this procedure is carried out, three significant shape parameters are found, the first being lateral bending of the pig’s back, accounting for 78% of the variance from the mean; and the second being nodding of the pig’s head: as the latter corresponded to only $\sim 20\%$ of the total variance, it was ignored in later analysis. In addition, the gray-level distribution model had three modes of variation amounting to a total of 77% of the intensity variance: the first two modes corresponded to (a) a general amplitude variation in which the distribution is symmetrical about the backbone, and (b) a more complex variation in which the intensity distribution is laterally shifted relative to the backbone (this arises largely from lateral illumination of the animal), see Fig. 22.20.

While principal components analyses yield the important modes of variation in shape and intensity, in any given case the animal’s profile has still to be fitted using the requisite number of parameters—one for shape and two for intensity. The Simplex algorithm (Press et al., 1992) proved effective for this purpose. The objective function to be minimized to optimize the fit takes account of (a) the average difference in intensity between the rendered (gray-level) model and the image over the region of the model; and (b) the negative of the local intensity gradient in the image normal to the model boundary averaged along the model boundary (the local intensity gradient will be a maximum right around the animal if this is correctly outlined by the model).

One crucial factor has been skirted around in the preceding discussion: that the positioning and alignment of the model to the animal must be highly accurate (Cootes et al., 1992). This applies both for the initial PCA and later when fitting individual animals to the model is in progress. Here we concentrate on the PCA task. When using PCA, it should be borne in mind that it is a method of characterizing deviations: this means that the deviations must already be minimized by referring all variations to the mean of the distribution. Thus, it is very important when setting up the data to bring all objects to a common position, orientation, and scale

**FIGURE 22.20**

Effect of one mode of intensity variation found by PCA. This mode clearly arises from lateral illumination of the pig.

Source: ©World Scientific 2000

before attempting PCA. In the present context the PCA relates to shape analysis, and it is assumed that prior normalizations of position, orientation, and scale have already been carried out. (Note that in more general cases scaling may be included within PCA if required. However, PCA is a computation intensive task, and it is best to encumber it as little as possible with unnecessary parameters.)

Overall, the achievements outlined above are notable, particularly in the effective method for decoupling shape and intensity analysis. In addition, the work holds significant promise for application in animal husbandry, demonstrating that animal monitoring and ultimately behavioral analysis should be attainable with the aid of computer vision.

22.15 CONCLUDING REMARKS

This chapter has shown something of the purpose of surveillance, which is largely to do with monitoring the behavior patterns of people and vehicles on roads and precincts. It has also shown a number of the principles and methods by which surveillance may be implemented: these include identification and elimination of background; detection and tracking of moving objects; identification of the ground plane; occlusion reasoning; Kalman and particle filtering; capability for modeling complex motions including those of articulated objects; and use of multiple cameras for widening coverage in time and space.

Over time, some specialized application areas have appeared, such as location and identification of license plates, identification of vehicles exceeding the speed limit, human gait analysis, and even animal tracking: the chapter has aimed to indicate how all these can be achieved. Early methods included Kalman filters and chamfer matching, and later ones included particle filters, which rely on a probabilistic approach to tracking. Particle filters have come a long way, but it is doubtful whether they can go much further if based on probability assessment alone, as it is clear that humans bring to bear huge databases of relevant information when tracking moving objects.

An important lesson is that detection and tracking are distinct, complementary functions, and there is no reason why the same algorithms will be optimal for both. As indicated in [Section 22.3.3](#), foreground detection requires the application of a suitable foreground model, or else a bootstrapping process involving an “exception to background” procedure. But once detection has been achieved, tracking can in principle proceed as a much simpler, more blinkered process. It remains to be seen whether future work will find ways of streamlining the detection + tracking model. Most likely it will be found lacking, because objects such as people radically alter in appearance as they walk by; hence, it is more natural to have both processes working in parallel (not necessarily at constant rates) all the time, rather than being applied serially. The same applies when a guided missile approaches a tank but can be misled into tracking a different object in the background as the scale of the target radically changes by several orders of magnitude: again the tracking algorithm needs to be monitored by a continuously acting detection algorithm. These points are labored because detection and tracking are at the core of surveillance, whatever the application area, and are thus very much the generic backcloth to this chapter.

While the chapter has covered the situation of static cameras being used to monitor moving objects, the following chapter covers the intrinsically more complex case of in-vehicle vision systems, where moving cameras are used to monitor both stationary and moving objects. This will call for a radical rethink of vision system strategy, because all parts of the scene will be eternally shifting and changing, and it will generally not be possible to rely on relatively trivial preliminary identification of a stationary background.

This chapter has shown that surveillance is largely about the detection and tracking of moving objects, and that different types of algorithm will often be needed to achieve each of these functions. In most cases locating the ground plane is a necessary first step in the analysis, while occlusion reasoning, Kalman filtering, capability for modeling complex motions, and multiple cameras will often be needed to achieve the ultimate aim of analyzing developing behavior patterns.

22.16 BIBLIOGRAPHICAL AND HISTORICAL NOTES

As we have seen, surveillance involves many factors, from 3-D to motion, but paramount among these is the tracking of moving objects—in particular, vehicles and

people. For some years tracking meant the use of Kalman filters, but the deficiencies of this approach led in the 1990s to the development of particle filters, including particularly the work of Isard and Blake (1996, 1998), Pitt and Shepherd (1999), van der Merwe et al. (2000), Nummiaro et al. (2003), Nait-Charif and McKenna (2004, 2006), Schmidt et al. (2006), and many others. Much of the early work is summarized in a tutorial paper by Arulampalam et al. (2002), though Doucet and Johansen (2011) have justifiably felt it necessary to produce another. The first of these and a 2008 preprint of the second might better be called reviews than tutorials, as the going is difficult—partly because of the lack of explanatory figures—and often it is easier to appeal to the original works than to them.

In parallel with these developments, much work took place on background modeling using both parametric and nonparametric methods, see for example Elgammal et al. (2000). Cucchiara et al. (2003) helped by defining the stationary and transient background problems and by clarifying the problem of “ghosts.” Shadows have been a source of problems over the whole period, not least because they can be static or moving, and also because they can fall on static or moving objects: Elgammal et al. (2000) and Prati et al. (2003) carried out seminal work on this topic.

Khan and Shah (2000, 2003, 2009) were responsible for a thoroughgoing approach to the tracking of people both with single and with multiple cameras, this work being followed up by Eshel and Moses (2008, 2010) who found how to make good use of top-of-head tracking in crowd scenes. Pflugfelder and Bischof (2008, 2010) developed the approach to cover nonoverlapping views—a task that had previously (Makris et al., 2004) been solved with some degree of generality, but without knowledge of scene geometry, by learning transition probabilities for objects passing between views.

Vezzani and Cucchiara (2008) and Vezzani et al. (2011) made a careful analysis of the means by which occlusions can arise, and this enabled them to devise algorithms that cope better with temporary partial or full occlusions or temporary merging of moving objects—in the sense of not getting confused, and recovering faster from such events.

Work on traffic monitoring has stretched over many years (e.g. Fathy and Siyal, 1995; Kastrinaki et al., 2003). Early work on the application of snakes to tracking was carried out by Delagnes et al. (1995); on the use of Kalman filters for tracking by Marslin et al. (1991); and on the recognition of vehicles on the ground plane by Tan et al. (1994). For details of belief networks see Pearl (1988). Note that corner detectors (Chapter 6) have also been widely used for tracking, see Tissainayagam and Suter (2004) for an assessment of performance.

A huge amount of work has been carried out on the analysis of human motions (Aggarwal and Cai, 1999; Gavrila, 1999; Collins et al., 2000; Haritaoglu et al., 2000; Siebel and Maybank, 2002; Maybank and Tan, 2004). See Sugrue and Davies (2007) for a simple method of distinguishing pedestrians. However, note that rigorous analysis of human motion involves studies of articulated motion (Ringer and Lazenby, 2000; Dockstader and Tekalp, 2001), one of the earliest enabling techniques being that of Wolfson (1991). As a result, a number of

workers have been able to characterize or even recognize human gait patterns (Foster et al., 2001; Dockstader and Tekalp, 2002; Vega and Sarkar, 2003): see Nixon et al. (2006) for a recent monograph on the subject. A particular purpose for this type of work has been the identification and avoidance of pedestrians from moving vehicles (Broggi et al., 2000; Gavrila, 2000). Much of this work has its roots in the early farsighted paper by Hogg (1983), which was later followed up by crucial work on eigenshape and deformable models (Cootes et al., 1992; Baumberg and Hogg, 1995; Shen and Hogg, 1995). Gavrila's work on pedestrian detection (Gavrila, 1998, 2000) used chamfer matching, while Leibe et al. (2005) developed the method further, albeit with the help of a minimum distance length (MDL) top-down segmentation scheme capable of handling multiple hypotheses.

While focussing on complex topics such as articulated motion and complications caused by occlusion, it is important not to lose sight of simple but elegant developments such as histograms of orientated gradients (HOGs), which have only appeared relatively recently (Dalal and Triggs, 2005). These were designed for, and are well-matched to, the detection of human shapes. Basically, they focus on the straight limbs of the human body, which have many edge points aligned along the same direction—though the latter will naturally change with walking or other motions. The basis of the method is to divide the image into “cells” (sets of pixels) and to produce orientation histograms for each of them. Voting into the orientation histogram bins takes place with weighting proportional to gradient magnitude. To provide strong illumination invariance, a robust normalization method is used. The cells are combined into larger overlapping blocks in several ways, with the result that some of the blocks end up with larger signals indicating the presence of human limbs. However, the result is that the HOG detectors cue mainly on silhouette contours and emphasize the head, shoulders, and feet. In a later paper, Dalal et al. (2006) combined the HOG detector with motion detectors and were able to achieve even better results (motion detection improved the false alarm rate by a factor of 10 relative to the best appearance-based detector). An interesting feature of the HOG approach is that it outperforms wavelet analysis because the latter eliminates vital abrupt edge information by prematurely blurring the image data.

Overall, work on surveillance has stretched over many years, but has vastly accelerated since the mid-1990s as workers have had access to more powerful computers that made it realistic to think of real-time implementation, both for experimentation and for on-road systems. Note that the past few years have seen developments in real-time systems involving FPGAs (a trend that was already present around 2000) and GPUs (a trend that is especially recent, and has arisen as a result of natural interaction between the video games industry and computer vision): for further details see Chapter 26.

22.16.1 More Recent Developments

Among the most recent works, Kim et al. (2010) have proposed a robust method for recognizing humans by their gait by using a hierarchical active shape model.

The approach is novel in that it is prediction-based and overcomes the drawbacks of existing methods by extracting a set of model parameters instead of directly analyzing the gait. Feature extraction proceeds by motion detection, object region detection, and Kalman prediction of the active shape model parameters. The method is able to alleviate tasks such as background generation, shadow removal, and obtaining high recognition rates. Ramanan (2006) has obtained good results by a new iterative parsing method for analyzing motions of articulated bodies ranging from humans playing games to horses frolicking and cantering. The approach has the advantage of being generic and does not depend on the location of skin or human faces. Lian et al. (2011) have obtained impressive performance when tracking pedestrians between camera view separations of more than 20 m—much greater than the separations ~ 4 m obtained by Pflugfelder and Bischof (2008, 2010).

Ulusoy and Yuruk (2011) have analyzed the problems of fusing data from visual and thermal images in order to make good use of their complementary properties to improve overall performance. They show that fusion should lead to a better recall rate (fewer false negatives), but at the same time result in a decrease in precision rate (more false positives); they also note that the infrared (thermal) domain always has higher precision (the underlying reasons for these observations are that thermal images effectively provide the *foreground* information containing the object pixels). In fact, it is only worth attempting fusion when an improved recall rate is required. This paper presents a more efficient method for fusing the data from the two domains and at the same time obtaining recall rates better than those previously obtained. The method was tested on outdoor images of human groups including those from a well-known database. The work in this paper leans heavily on the earlier work of Davis and Sharma (2007). Both papers refer to thermal imagery in spite of the title of the first which refers to “infrared” images.

Finally, three key papers have highlighted recent progress with the surveillance of road users: Buch et al. (2010) have employed 3-D wire-frame models for the classification of road users; Lazarevic-McManus et al. (2008) have demonstrated the value of the F-measure for optimising motion detection on the roads; and Xu et al. (2011) have obtained improved accuracy and robustness for tracking partially observable targets on the roads.

22.17 PROBLEM

- When an inverse perspective mapping onto the ground plane is carried out, points on the ground plane are well represented in the new representation. Explain why this does not apply for buildings or people, and why they always appear to lean backward when presented in this representation.

23 In-Vehicle Vision Systems

This chapter considers the value of in-vehicle vision as part of the means for providing driver assistance systems. To achieve this, many objects have to be identified, including not only the roadway itself but also the lane and other markings on it, road signs, other vehicles, and pedestrians. The latter are particularly important as their actions are relatively unpredictable, and people who wander into the roadway are liable to cause accidents—unless the driver assistance system can help to avoid them.

Designing in-vehicle vision systems is anything but trivial, as they necessarily deploy moving cameras, which means that all objects in a scene are moving; hence it becomes quite difficult to eliminate the background from consideration. For these reasons, it becomes necessary to rely more on recognition of individual objects than on motion-based segmentation.

Look out for:

- how the roadway, road signs, and road markings may be located.
- the availability of several distinct methods for locating vehicles.
- what information can be obtained by viewing licence plates and wheels.
- how pedestrians may be located.
- how vanishing points can be used to provide a basic understanding of the scene.
- how the ground plane may be identified.
- how a plan view of the ground plane can be obtained and used to help with navigation.
- how vehicles can be guided using vision to compensate for roll, pitch, and yaw.

While it is easy to set out strategies for building in-vehicle vision systems that will work well in normal conditions on the roadway, it is far from simple to design them to operate on the less structured environments of farms or fields. Indeed, much additional reliance on GPS and other methodologies will often be needed for the purpose.

23.1 INTRODUCTION

This chapter provides an introduction to in-vehicle vision systems. The topic clearly overlaps with many of the ideas of the previous chapter, particularly regarding traffic surveillance, as here we are regarding the flow from inside a vehicle rather than from a stationary camera mounted (typically) on an overhead gantry. However, although the environment may be similar, the situation is essentially different, because the camera platform is in motion and almost nothing that is viewed appears stationary ([Table 23.1](#)). This means that it is extremely difficult to use methods such as background subtraction. Note that while it is theoretically possible to find a general perspective transformation that makes a sequence of frames exactly coincide so that background subtraction can be achieved, to do this would be to replace a technique that is intended to be a simple way of cueing into images into one that is highly complex; and the process of finding a sufficiently exact perspective transformation would itself require considerable computation, so this is unlikely to provide a useful strategy for analyzing image sequences.

Given the more difficult problem of analyzing scenes containing moving objects from a moving platform, we have to find ways of tackling the task equitably. Fortunately, with vehicles on a road, the range of types of scene is highly restricted. In particular, the roadway is always present in the image foreground, and thus is easily identifiable. Likewise, it normally has a characteristic dark intensity and thus, its recognition right into the distance need not be too problematic: the fact that it is moving relative to the camera is relatively immaterial. In fact, it may even be quite difficult to detect motion by looking downward toward the road surface. Next, there is a host of standard types of objects that are likely to be visible from within a vehicle—buildings, other vehicles, pedestrians, road markings, road signs, telegraph poles, lamp standards, bollards, and so on. The high frequency with which each of these can appear indicates that it will be necessary to have the capability of recognizing each of them independently, at any range and at any speed. This means that it is better, *as a first stage in the analysis*, to revert to ignoring speed of motion and to concentrate on pattern recognition. In fact, recognition can be helped by considering the range, which is readily deduced approximately at first from the lowest location on the object, which is where it meets the road (it is here assumed that the road has already been segmented from the remainder of the scene as an important preliminary stage of the analysis). Note that depending on the aim of the analysis (a point to which we shall return below), it is likely to be more

Table 23.1 Levels of Difficulty When Motions Can Occur

1. Locating stationary objects from a stationary platform
2. Locating moving objects from a stationary platform
3. Locating stationary objects from a moving platform
4. Locating moving objects from a moving platform

important to identify objects that lie within the road region, so segmentation of the latter is all the more important as a first stage. Then these objects—now restricted mainly to the subset, other vehicles, pedestrians, road markings, road signs, traffic lights—each needs to be identified in its own right. Later, the exact motion of the moving platform, and subsequently its location relative to all the other objects, will need to be ascertained.

We next consider the aims of implementing in-vehicle vision systems. Broadly, there are two: (1) navigation along the road, including staying in lane and finding out from road signs and traffic signals where to go, when to stop, and other such information (here, to simplify matters, we ignore use of GPS and other types of help, and how information from the various sources can be fused together reliably); (2) driver assistance, which can include a variety of matters, particularly informing the driver of all aspects included in (1), and alerting him or her to important factors, such as vehicles that are braking, or pedestrians who are moving onto the roadway. In fact, much of the information that is acquired by the vision system will need to be conveyed to the driver in one way or another. However, of particular interest is the fact that drivers will sometimes not be able to act rapidly enough to avoid pedestrians, vehicles that brake unpredictably, overtaking vehicles that suddenly cut in, and so on. There is also the problem that drivers may be drowsy or may for various reasons—e.g., because of distractions from other occupants or those caused by the simultaneous need to navigate—react too slowly, so that an accident could become imminent. In such cases, driver assistance that could automatically initiate breaking or swerving might be crucial. We can also envisage various situations where the vision system would be part of a fully automatic driving system: here there is bound to be a problem of legality, and who or what would be to blame for an accident (*viz.* the driver, car manufacturer, vision system designer, or whoever). We shall not delve into such problems here, but just consider the vision system as an enabling technology. However, once vision and driver assistance systems become sufficiently powerful, they will doubtless become part of other schemes such as those for driving in tight convoys—deemed by many to be the best way of achieving rapid safe transit along our motorways. In addition, there are other ways in which driver assistance can be valuable: these range from cruise control to automatic parking.

In this chapter, we focus generally on providing a vision system that can perceive all that might be needed for vehicle guidance and driver assistance, with emphasis on locating the roadway and road lanes, identifying other vehicles, and locating pedestrians close to or on the roadway. As indicated above, the whole process starts by locating the roadway, as discussed in the following section.

23.2 LOCATING THE ROADWAY

Chapter 4 described a technique that was capable of locating the roadway using a multilevel thresholding approach (see Fig. 4.9(b)). In fact, the roadway was identified by the third and fourth thresholds as that section of the image with gray

levels in the approximate range 100–140. Similar results are obtained in other cases, e.g., Fig. 23.1(b), where the two threshold values demarcate an even greater grayscale range, approximately 60–160. While these can be construed as being reasonably ideal cases, thresholding is such a basic technique that it should be possible to extend it to cover less ideal situations. For example, if shadows appear on the roadway, the latter would in many cases appear as two contiguous sets of regions with two prominent intensity levels, and could indeed be identified by the



FIGURE 23.1

Frame of video taken from a moving vehicle. (a) Original image. (b) Doubly thresholded image. (c) Result of only applying the lower threshold. (d) Top: intensity histogram of the original scene. Middle: result of applying the global valley transformation and smoothing. Bottom: the dotted line shows the two thresholds used in (b) being located automatically. For further details, see text in Chapter 4. The graphics dots in (c) demonstrate that, within the road area, the lower threshold predominantly identifies under-vehicle shadows.

Source: © IET 2007

same method. Note also that varying illumination levels would be likely to make one intensity elide smoothly into another, and if a suitable range of intensities between thresholds (as in [Figs. 4.9 and 23.1](#)) were taken into account, the segmentation problem might still be solved in exactly the same way. However, ultimately the problem is one of pattern recognition, and can be solved by (a) eliminating other objects, such as road lane markings, (b) identifying the limits to the roadway, and (c) taking other features such as color or texture into account. Note that as the color of the roadway is often a bland gray, it may only be made to stand out by noting the colors of the other surroundings, such as grass, trees, or brickwork on buildings. Clearly, this would make the whole system more complex, but in a well known and well worn way—pattern recognition by now is a reasonably mature subject. To some extent the situation may be helped by bringing the motion of the vehicle into the picture (we have so far resisted this, to bring the discussion to the simplest possible base level). In that case, without calculating the exact motion of the vehicle, we can take account of the fact that the roadway stretches for a long distance ahead, so any part of it that is established to be roadway will remain so until the vehicle passes over it. Furthermore, on the road ahead, any vehicle that is located evidently runs on the roadway, so parts of it are continuously being identified. Thus, the camera vehicle merely needs to keep a record of all candidate regions that have been positively identified, so that any ambiguities from identification *via* intensities can be eliminated. Finally, this time taking motion parameters into account, keeping a tally on the road boundaries with the aid of Kalman filters will solve many of the remaining issues.

23.3 LOCATION OF ROAD MARKINGS

It has been noticed from [Figs. 4.9 and 23.1](#) that the multilevel thresholding technique used to locate the gray surface of the road simultaneously segments white road markings. However, white road markings are seldom pure white and may be worn or even partly duplicated by older markings. In any case, segmenting them by thresholding is not the same as absolute identification. One way around this dilemma is that of fitting the road markings to suitable models. Often straight lines are adequate, although sometimes parabolas are used for the purpose. [Figure 23.2](#) shows a case where continuous and broken road markings have been identified using the RANSAC technique, which helps to locate the vanishing point on the horizon to a reasonable approximation. The widths of the road lane markings can also be measured in this way. [Figure 23.3](#) takes this even further. In this case, a greater degree of reliability and accuracy is obtained by locally bisecting each lane marking horizontally before feeding the data to RANSAC. In this way, extraneous signals can be eliminated—if necessary by filtering the horizontal widths. Note how RANSAC is able to find the best fit straight line section even when the road lane markings are curved. Likewise, it is able to eliminate lane markings that have been distorted by the presence of older lane

(a) (b)

FIGURE 23.2

Application of RANSAC for locating road lane markings. (a) Original image of road scene with lane markings identified by RANSAC. (b) The edge point local maxima used by RANSAC for locating the road lane markings. While the lane markings converge to approximately the right point on the horizon line, the parallel sides of the individual lane markings do not converge quite so accurately, indicating the limits achievable with so few edge points. This is more a failure of the edge detector than of RANSAC itself.

markings ([Fig. 23.3\(a\)](#)). As described in Chapter 11, the version of RANSAC used for the tests successively eliminates the data points used to fit line segments, and the width delete threshold d_d is made larger than the fit threshold d_f so that no data points are retained that could mislead the algorithm while searching for subsequent line segments (see the algorithm flowchart in [Fig. 23.4](#)).

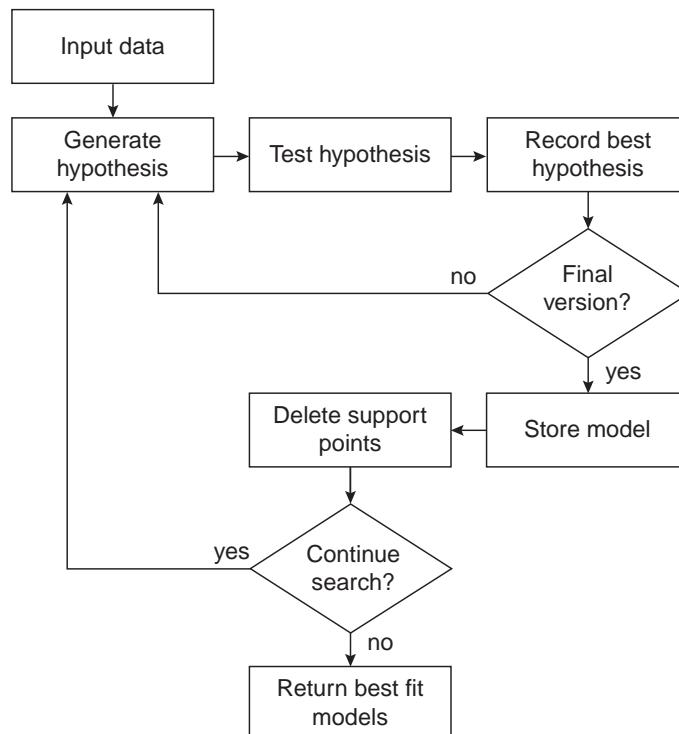
23.4 LOCATION OF ROAD SIGNS

We now continue with the process of analyzing the vehicle’s environment and consider the most relevant remaining stationary parts that lie on or adjacent to the roadway. These include the traffic signs. It will not be possible to examine more than one or two cases, but among these are various relevant warnings, including those for road bumps and “GIVE WAY”: note that many others appear in the same style—with the message in black on a white background and enclosed in a red triangle. To locate these signs, some tests were made without using the color aspect as this might represent too easy an approach (note also that in the wrong lighting conditions, color can be misleading): instead, an idealized small binary template of size 22×19 pixels was employed. While apparently crude, this small template had the advantage of requiring very little computation to locate the relevant objects. In fact, the chamfer matching technique (Section 22.7) was used for detecting the traffic signs shown in [Fig. 23.5](#). While the template was primarily designed to detect

FIGURE 23.3

Further tests of RANSAC for locating road lane markings. (a) Original image 1: a distorted set of double line road markings. (b) Thresholded version of (a). (c) 3–3. (d) 3–6. (e) 3–10. (f) 3–11. (The notation “ $d_f - d_d$ ” means that d_f is the “fit distance” and d_d is the “delete distance;” see text.) (g) Original image 2, already thresholded: the central section of the road containing no markings has been eliminated to save space. (h) 3–3, (i) 3–6, (j) 3–11. (f) and (j) show the final results as dense dotted lines: in other cases, dots and dashes are used to distinguish the different lines. Note that immediately after thresholding, the horizontal bisector algorithm finds the midpoints of white regions along horizontal lines, and feeds them to RANSAC for fitting.

Source: © IET 2007

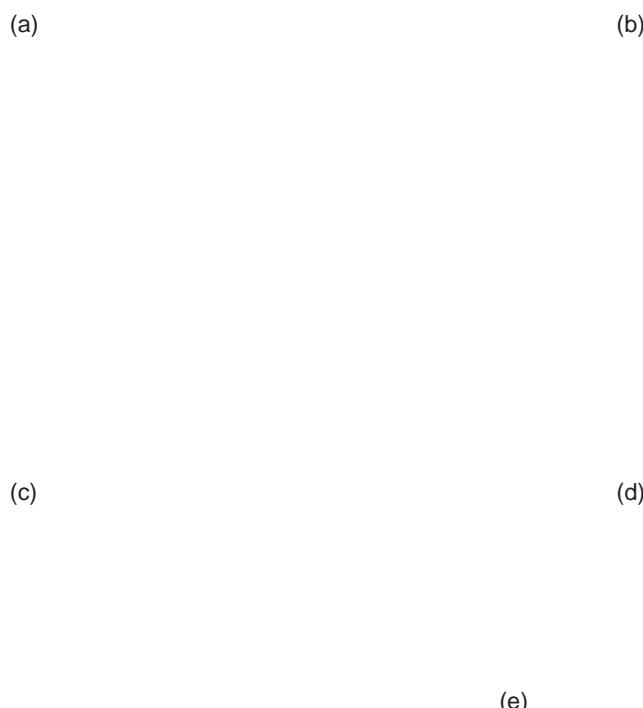
**FIGURE 23.4**

Flowchart of the lane detector algorithm used for the tests in Fig. 23.3.

Source: © IET 2007

the road bump sign, it also gave a sizeable signal for the GIVE WAY sign. Indeed, the two signals found using the template were both well above the signal-to-noise ratio elsewhere in the image, the closest possible false alarms being high up in the trees, which contain a plethora of random shapes. Note that the picture was taken under highly nonideal conditions on a wet day when there were a number of reflective areas on the road. Overall, the chamfer matching technique seems well suited to rapidly locating fixed road signs of various sorts.

There is some possibility of designing a single idealized template for locating all triangular signs. Note first that a blank white interior would be more suitable than the road bump structure in Fig. 23.5(e): this corresponds to disregarding the center of the template, taking it as being composed of “don’t care” locations. In fact, the template should really be designed by a suitable training approach such as the one outlined by Davies (1992d). In this method, a matched filter approach is used in designing templates with local variability of training samples (represented by standard deviation $\sigma(x)$) being taken to correspond to noise, thereby necessitating reduced local weighting: the local matched filter weighting is thus (Davies, 1992d) taken as $\bar{S}(x)/\sigma(x)^2$ rather than $\bar{S}(x)$, where

**FIGURE 23.5**

Locating road signs using chamfer matching. (a) Original image showing two triangular road signs (indicating a road bump and “GIVE WAY”): each of the signs is marked with a white cross where it has been located by the chamfer matching algorithm. (b) Thresholded edge image after nonmaximum suppression. (c) Distance function image: note that with the display enhancement factor of 20 used here, the distance appears to saturate at 13 pixels. (d) The response obtained when moving the template (e) over the image.

$\bar{S}(\mathbf{x})$ is the mean local signal at \mathbf{x} during training. For the types of road sign considered above, variable distributions of black within the central white area would be treated optimally by this method.

23.5 LOCATION OF VEHICLES

In recent years a number of algorithms have been designed for locating vehicles on the road, whether in surveillance applications or by in-vehicle vision systems. One notable means for achieving this has been by looking for the shadows induced by vehicles (Tzomakas and von Seelen, 1998; Lee and Park, 2006). Importantly, the strongest shadows are those appearing beneath the vehicle, not least because these are present even when the sky is overcast and no other shadows are visible. Such shadows are again identified by the multilevel thresholding approach of Chapter 4. [Figure 23.1](#) shows a particular instance of this, where almost the only dark pixels appearing within the roadway region are the under-vehicle shadows. In fact, as under-vehicle shadows lie under vehicles, an excellent way of locating nearby vehicles is to move upward from the lowest part of the roadway until a dark entity appears: there is then a high probability that it will only locate vehicles. Note that in [Fig. 23.1\(c\)](#) the other main candidates are trees, but these are discounted as being well above the road region—as indicated by the dotted triangle.

As pointed out earlier when considering methods for locating the road region, it is useful to have a number of methods available for locating objects such as vehicles, in case of peculiar illumination conditions or other factors. Following this line of analysis we consider symmetry, which was first used for this purpose some years ago (e.g., Kuehnle, 1991; Zielke et al., 1993). [Figure 23.6](#) shows a number of trials in which symmetry is applied to locate objects exhibiting a vertical axis of symmetry. The approach used is the 1-D Hough transform, taking the form of a histogram in which the bisector positions from pairs of edge points along horizontal lines through the image are accumulated. When applied to face detection, the technique is so sensitive that it will locate not only the centerlines of faces but also those of the eyes. In the case of [Fig. 23.6\(c\)](#), the algorithm was confused by the metal object at the bottom when locating the eye on the left, but when tested without that present it was found without difficulty. Note that some bias occurs there because the algorithm is averaging the contribution of the whole eye and the displacement between the iris and the rest of the eye becomes important. Similarly, the set of leaves in [Fig. 23.6\(e\)](#) is located without trouble, but the exact vertical axis that is located represents the combined peak signal from the lower two leaves and the uppermost leaf: in such a case it would be better to identify each one separately. These sorts of problem are less important in [Fig. 23.6\(g\)](#) where both vehicles are located quite accurately—in spite of the fact that the car on the right is not exactly horizontal. Interestingly, both vehicles would also be found using the under-vehicle shadow method. The fact that they both lie within their respective lanes also aids positive identification.

In spite of these successful applications of symmetry, note that the approach needs to be used with caution. In particular, the building on the left in [Fig. 23.6\(g\)](#) gives a plethora of signals because of the multiple symmetries between its windows. An interesting lesson is that three equally spaced vertical lines at locations $x = 1, 3, 5$ will have a symmetry not only at $x = 3$ but also at $x = 2$ and 4.

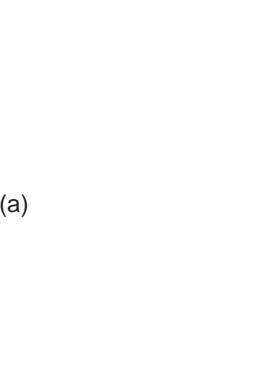
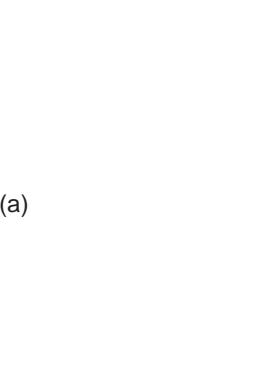

(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)

FIGURE 23.6

Searching for symmetry in images. (a) Original image of a face with a vertical axis of symmetry. (b) Edge image used for determining the axis of symmetry in (a). (c) Original image with symmetry axes of the eyes. (d) Slightly restricted edge image used

Finally, rotation symmetries and reflection symmetries about nonvertical axes are not especially useful in the present context. However, just as a 1-D HT can be used to locate symmetries about vertical axes, so 2-D HTs can be used to locate symmetries about lines of arbitrary direction. Thus, one can build a single 2-D parameter space, each horizontal line of which represents the symmetry in a different direction in the image. Such a parameter space might be expected to have a minor amount of coherence in the vertical direction, but we do not consider this further here.

23.6 INFORMATION OBTAINED BY VIEWING LICENCE PLATES AND OTHER STRUCTURAL FEATURES

Licence plate location has already been covered in Section 22.10. In this section, we consider what can be deduced from an oblique view of a licence plate of length R . We simplify the situation by assuming that both the image plane and the licence plate are vertical, and that they have their main axes aligned horizontally and vertically. [Figure 23.7\(a\) and \(b\)](#) shows respectively the oblique and plan views of the licence plate horizontal axis. The apparent horizontal projection (CQ) of the centerline of the licence plate is $R \cos \alpha$ when viewed in the direction PT. Following [Fig. 23.7\(c\)](#), its vertical projection (QT) is $R \sin \alpha \tan \beta$. However, when viewed in the more general direction PT', with lateral angle λ , its horizontal projection is CQ', which is equal to $R \cos \alpha - R \sin \alpha \tan \lambda$. From [Fig. 23.7\(d\)](#), we deduce that its apparent angle γ and length R' are given by the equations:

$$\tan \gamma = \frac{\tan \alpha \tan \beta}{1 - \tan \alpha \tan \lambda} \quad (23.1)$$

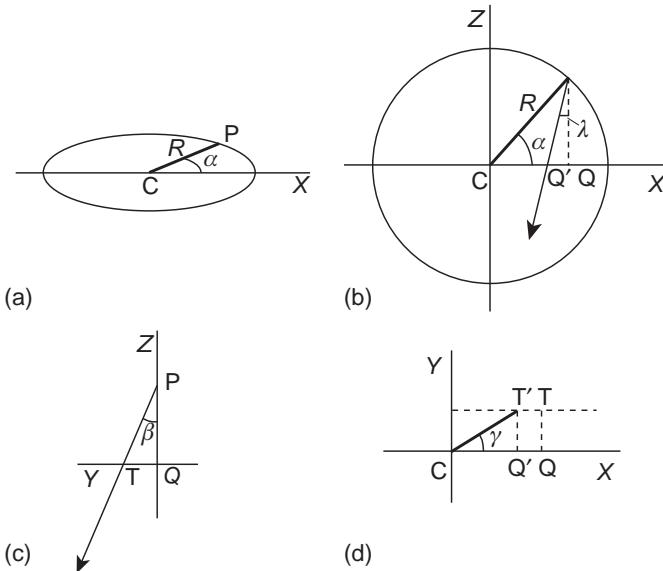
$$\begin{aligned} R' &= R \cos \alpha (1 - \tan \alpha \tan \lambda) \sec \gamma \\ &= R \sin \alpha \tan \beta \operatorname{cosec} \gamma \end{aligned} \quad (23.2)$$

These formulas seem intuitively correct, as for example, $\gamma = 0$ if $\alpha = 0$ or $\beta = 0$. In addition, under nonoblique viewing, $\beta = 0$, $\lambda = 0$, and $\gamma = 0$, so [Eq. \(23.2\)](#) reverts to the standard result for nonoblique viewing, $R' = R \cos \alpha$.

Perhaps a more important case is that of $\alpha = \pi/2$, leading to $\tan \gamma = -\tan \beta / \tan \lambda$. We can interpret this result by taking image plane coordinates (x, y) and 3-D coordinates (X, Y, Z) . Noting that $\tan \beta = y/f$ and $\tan \lambda = x/f$, we deduce that

FIGURE 23.6 (Continued)

for determining the symmetry axes of the eyes. (e) Original image of a leaf triplet, with symmetry axis. (f) Vertical edge image used to determine the symmetry axis in (e). (g) Original image of a traffic scene, with symmetry axes marked. (h) Vertical edge image used for determining the symmetry axes in (f). The slight bias of the left-most symmetry axis in (c) is not surprising in view of the few pixels involved and the interfering effects of other edge pixels in the image.

**FIGURE 23.7**

Horizontal line pose viewing geometry. (a) Oblique view of a horizontal straight line of length R , rotated through an angle α from the X -axis. (b) Plan view of the line. (c) Side view showing the viewing direction, along PT' , with a lateral angle λ ; the angle of elevation β is that of T , not of T' . (d) Front view in the X - Y plane, which is parallel to the image plane x - y . Note that the horizontal line CP in (b) appears to lie at an angle γ in (d): it has an apparent length (CT') of R' .

$\tan \gamma = -y/x = -Y/X$. This corresponds to viewing perspective lines on the roadway that are parallel to the optical axis of the camera. (Note that the minus sign in these equations corresponds to the fact that γ will be viewed in the range $\pi/2$ to π when $\alpha = \pi/2$.)

Finally, note that instead of obtaining the projection of the line as it would appear in the direction of viewing, we have determined its projection in the vertical plane X - Y , which is parallel to the image plane x - y . As a result, the equations correspond *exactly* to projective projection into the image plane, rather than merely to orthographic projection.

We now need to obtain an equation for α in terms of the other parameters. Solving Eq. (23.1) for α , we find:

$$\tan \alpha = \frac{\tan \gamma}{\tan \beta + \tan \gamma \tan \lambda} \quad (23.3)$$

Next, taking the projections of the centerline of the licence plate along the image x and y axes to be δx , δy , we find that the parameters β , γ , λ are all measurable, so α can be estimated:

$$\tan \alpha = \frac{\delta y / \delta x}{(y/f) + (\delta y / \delta x)(x/f)} = \frac{f \delta y}{y \delta x + x \delta y} \quad (23.4)$$

Thus, we now know the orientation in space of the licence plate. In principle, we can use Eq. (23.2) to estimate the range of the licence plate. To achieve this, we need to know the value of R . In fact, for standard UK licence plates, R is reasonably well defined (this assumes that the number of characters in the licence plate is known), and so Eq. (23.2) can be used to estimate R' . Next, the ratio of R' to the apparent length r of the licence plate gives the range Z :

$$Z = \frac{fR'}{r} = \frac{fR'}{[(\delta x)^2 + (\delta y)^2]^{1/2}} \quad (23.5)$$

If we had also made use of the apparent lengths and orientations of the shorter sides of the licence plate, we could have eliminated dependence on the assumptions that the latter are vertical. However, it is unlikely that these short lines could be measured accurately enough to improve the situation significantly: instead we presume that the best that can be done is to use measurements on the longer sides to obtain preliminary estimates of the positions of vehicles, which can then be improved by other measurements.

Unfortunately, all the above theory is somewhat confounded by the variable camber of the road. But note that, while the camber will be considerably different on the opposite side of the road, its effects will tend to cancel when observing the licence plates of vehicles on the same side of the road. Next, the size of γ depends on y , and hence on the height of the camera above the target feature: this means that the observed value of γ will be smaller for the licence plate than for the rear wheels; hence, if the rear wheels are not occluded, it is likely that they will give a more accurate estimate of α than that from the licence plate. Nevertheless, licence plates are more satisfactory indicators than rear wheels both because they are less likely to be occluded and because they are uniquely recognizable: in fact, the rear wheels of one vehicle can sometimes be confused with those of other vehicles, and even the front wheels can cause confusion. Finally, another factor needs to be borne in mind—that we are attempting to estimate an often small quantity α from another small quantity γ when both are comparable to the interfering effect of the camber angle. Interestingly, this problem can be overcome more effectively by estimating $\tilde{\alpha} = \pi/2 - \alpha$ from $\tilde{\gamma} = \pi/2 - \gamma$ and applying these measures to views of the sides (particularly the sides of the wheels) of other vehicles. All this can be achieved by recalling that $\tan \alpha$ and $\tan \gamma$ should respectively be replaced by $\cot \tilde{\alpha}$ and $\cot \tilde{\gamma}$ in Eqs. (23.1) and (23.2). Overall, it might be expected that side views of vehicles will be more valuable for estimating orientation than rear views, whether the latter use rear wheels or licence plates as indicators (although, obviously, only the rear view of a vehicle will be relevant when driving directly behind it). Consideration of Figs. 23.1, 23.6, 23.8 and 23.9 will provide adequate confirmation of these observations.

Finally, it might be asked why so much emphasis has been placed on measurement of angles *vis-à-vis* distances. This is basically because angles represent ratios of distances and thus they tend to provide scale-invariant information. In addition, they do not demand knowledge of absolute distances for interpretation.

(a)

(b)

FIGURE 23.8

Vehicles viewed obliquely. More accurate information about orientation is often obtained from the side of the vehicle than from its rear.

(a)

(b)



(c)

(d)

FIGURE 23.9

Chamfer matching to locate pedestrians from their lower legs. (a) and (b) show original images of road scenes containing pedestrians. The white dots are the peak signals after chamfer matching using an idealized binary U template. Note the plethora of false positives because of the number of vertical edges able to stimulate signals—as seen in (c) and (d).

23.7 LOCATING PEDESTRIANS

In principle, locating whole pedestrians would require many chamfer templates of varying shapes and sizes, to cover the many body profiles of moving people. The alternative chosen here is to look for specific sub-shapes that would be more general and invariant. Possibilities include leg, arm, head and body sections. [Figure 23.9](#) shows lower legs being located using an idealized “U” template with parallel sides. However, a plethora of false positives arises because of the large number of vertical edges that are able to stimulate signals. Their presence means that the distance functions do not have the ideal maximum values that might be expected because the spurious edges reset the distance functions to zero in many places. This does not affect the sensitivity of the method in the sense that the templates are bound to locate instances of the profiles they represent. However, it does affect the numbers of false positives that are detected. In fact, in the examples shown, the result is not disastrous, because the lowest objects found, once road markings are eliminated, are the feet of the pedestrians. However, the fact that the method does not give ideal results makes it essential to back it up using alternative methods.

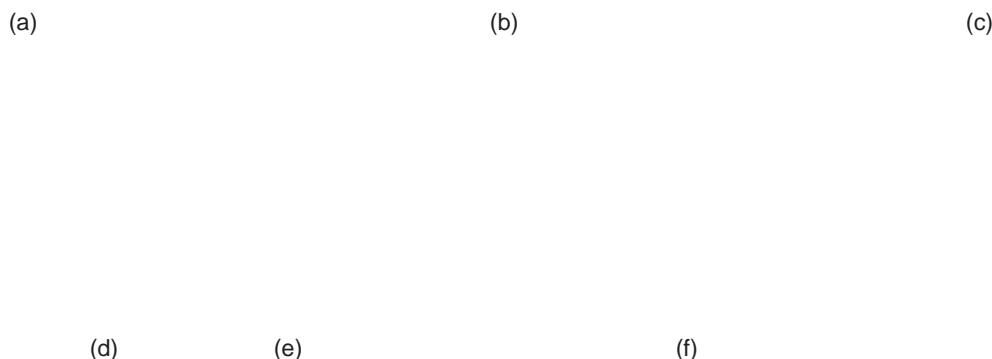
The Harris operator provides a useful alternative approach. As [Fig. 23.10](#) shows, it is able to locate a range of features, including feet and heads, as well as road lane markings. Note that in the case shown in [Fig. 23.10\(a\)](#), the right foot has not been found as it is larger than the other foot and the particular Harris operator employed stretched over a range of only 7 pixels. Note that the Harris operator has no sense of polarity (preference for black or white): in the case of pedestrians this is useful as the clothing and shoes (or feet) are unpredictable and can appear dark on a light background or *vice versa*. (Lack of polarity also applies to chamfer matching, but for different reasons.)

(a)

(b)

FIGURE 23.10

Alternative approach to pedestrian location using the Harris operator. Here the operator has the effect of locating corners and interest points, some of which include pedestrian feet and heads: above all, road lane markings are also located with high probability. The operator has not been tuned in any way to recognize such features. In addition, it has no sense of polarity (preference for black or white).

**FIGURE 23.11**

Another approach to pedestrian location via skin color detection. (a) and (b) show that a lot can be achieved via skin color detection, detecting not only faces but also neck, chest, arms and feet: see also the detail in (c) and (d). With proper color classifier training, even more can be achieved, as shown in (e) and (f) (see also larger version of figure in color Plate 7).

Further approaches are useful to back up the two mentioned above and also to confirm detections that have already been made. In this respect unique identification of human skin color can be useful. That this is possible is shown in Fig. 23.11, one of the main problems clearly being the rather small numbers of pixels in the face regions. To carry out skin detection rigorously, it is necessary to train the color classifier on a set of training images. This was carried out for Fig. 23.11(e). While the method was highly successful (see Fig. 23.11(f)), it corresponded to supervised learning of skin color; in practice, with less tight control of the training images, this process could be compromised by the presence of sand, stone, cement, and a host of brown variants, which have colors close to those of darker or lighter people. Another important factor is that in-vehicle vision systems will not have sufficient time to gather enough training data, considering particularly that the whole point of a vehicle is to travel and thus adaptation from dark to light and other environmental factors are bound to be a source of serious problems. In this respect, in-vehicle systems are subject to far worse conditions than will be usual for surveillance systems.

Overall, we find that in-vehicle pedestrian detection systems involve a demanding set of pattern recognition problems. Earlier we emphasized the potential value of pattern recognition when moving objects are being detected from moving platforms: this approach to the subject was also useful for didactic

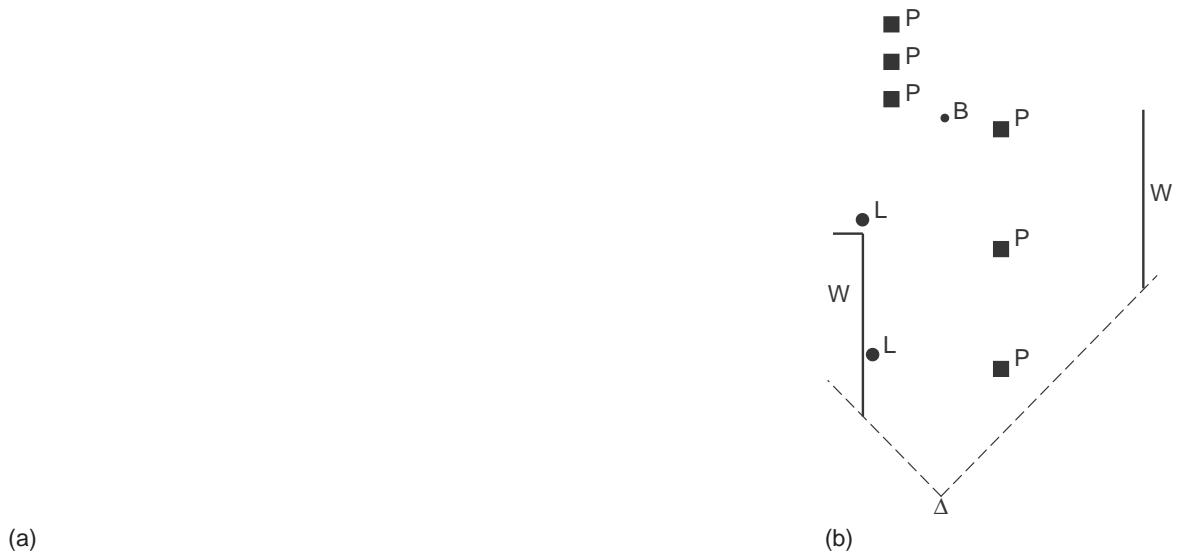
reasons. However, we are now finding that there are limits to this. In fact, it would be an artificial restriction not to make use of motion by at least tracking features and grouping them according to velocity (a process that was already mentioned in Chapter 22). The problem with this approach is the large number of, e.g., interest point features that exist in an entire image, where almost all the features are moving. If each of them (say N) is to be compared with all others in a pair of adjacent frames, then $O(N^2)$ operations will have to be undertaken. However, by acknowledging the individuality and different characteristics of the various features, and their spatial arrangements, this vast number can be cut down to manageable proportions. In particular, feature points should only move a limited distance between frames, so there will only be a small number n of candidates that match a given feature as it moves from one frame to the next. This leaves us with $O(Nn)$ pairs of feature points to consider, a number that can be further minimized by examining the relative strengths and colors of the various pairs (ideally, the final result will be $O(N)$). Here, some of the ideas of Section 6.7, where features were characterized by a great many descriptors, may prove useful, even though wide baseline matching is not relevant for frame to frame tracking.

23.8 GUIDANCE AND EGOMOTION

An important aspect of driver assistance systems is that of vehicle guidance. In fact, this aspect is important both for vehicles with human drivers and for autonomous robot vehicles. In either case, vehicle egomotion is handled by a controlling computer that has to be fully aware of the situation. Incoming images contain complex information and reliable cues have to be found to key into them. Among the most widely used such cues are vanishing points (VPs), which are often very evident in city scenes (e.g., Fig. 17.11).

One of the ways in which VPs are most useful is in helping to identify the ground plane, and a lot of other information follows from this. In particular, local scale can be deduced: for example, objects on the ground plane have width that is referable to, and a known fraction of, the local width of the ground plane; in addition, VPs permit an estimate to be made of distance along the ground plane, by measuring the distance from the relevant image point to the VP, as we shall see below. Thus, they are useful for initiating the process of recognizing and measuring objects, determining their positions and orientations, and helping with the task of navigation.

Here, a lot will depend on the type of environment and the type of vehicle. There are many possibilities such as vacuum-cleaning robots, window-cleaning robots, lawn-mowing robots, invalid chair robots, weeding and spraying robots, maze-running robots, not to mention vehicles running autonomously on roads, or cars that park themselves automatically. In some cases, robots will have to undertake mapping, path planning, and navigational modeling and engage in detailed high-level analysis: this sort of situation has been explored by Kortenkamp et al. (1998). This approach will be important if a path has obstacles such as bollards or pillars

**FIGURE 23.12**

Plan view obtained for navigation. (a) View of a scene showing the obstacles to be avoided. (b) Plan view of the ground plane showing what is visible from viewpoint Δ (for clarity, the full areas of the pillars P, bollards B, and litter-bins L are shown). The walls are marked W.

(Fig. 23.12); and it will be vital for a maze-running robot. In many such cases, vision or other sensors will provide only limited information about the working area, and knowledge will have to be augmented in a suitable representation: this makes a plan view model of the working area a natural solution. To proceed with this idea, we need to transfer the information from individual images into the plan view representation (see the algorithm of Table 23.2).

Basically, to construct a plan view of the ground plane, we start with a single view of a scene in which the vanishing point V has been determined and significant feature points on the ground plane (particularly regarding its boundaries) have been identified. Next, distance along the ground plane can be deduced as shown in Fig. 23.13. The angle of declination α of a general feature point P(X, H, Z) on the ground plane, seen in the image as point (x, y), is given by:

$$\tan \alpha = \frac{H}{Z} = \frac{y}{f} \quad (23.6)$$

The value of Z is therefore given by:

$$Z = \frac{Hf}{y} \quad (23.7)$$

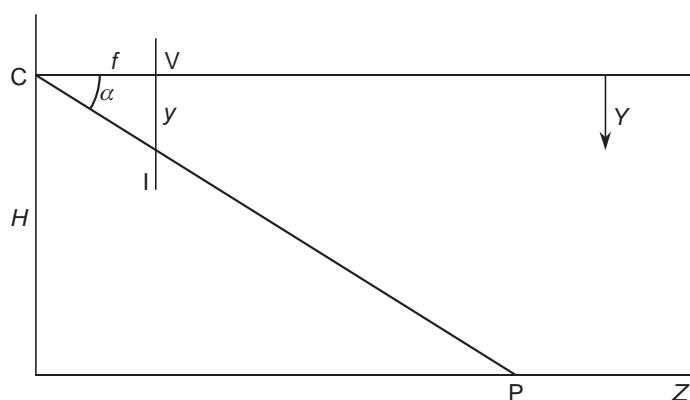
After obtaining a similar formula giving the lateral distance X , we deduce that:

$$X = \frac{Hx}{y} \quad (23.8)$$

Table 23.2 Computing Ongoing Plan Views of the Ground Plane

1. Detect all edges in the current frame.
2. Locate all straight lines in the current frame: e.g., use a Hough transform.
3. Locate all VPs: use a further HT, as described in Section 17.7.
4. Find the VP closest to the direction of motion: eliminate all other VPs.
5. Determine the closest section of G: this should be the part of the frame immediately in front of the robot.
6. Use this and other information to determine which lines through the primary VP lie on G: eliminate all other lines.
7. Segment objects on G.
8. Eliminate object boundaries on G that are unrelated to lines passing although the primary VP.
9. Tentatively identify as shadows any dark regions lying on G.
10. Take the remaining object and shadow boundaries and check for consistency between frames: e.g., use the 5-point cross-ratio values, as described in Section 17.3.
11. Label all remaining feature points on G with their (X, Z) coordinates: use [Eqs. \(23.7\) and \(23.8\)](#).
12. Check for consistency with previous frames.
13. Update list of objects with inconsistent boundaries as not lying on G, or as being otherwise unreliable: these could be due to moving shadows or noise.
14. Update history of feature point coordinates on G.

This table presents an algorithm showing how a plan view of the ground plane G may be computed. It is assumed that the robot sees a sequence of video frames and that it has to update its knowledge base as each frame comes along. The algorithm is set up assuming that it is best to analyze each frame ab initio, and then to look for consistency with previous frames.

**FIGURE 23.13**

Geometry relating the image and the ground plane. C is the center of projection of the camera, I is the image plane, V is the vanishing point, and P is a general point on the ground plane. f is the focal length of the camera lens and H is the height of C above the ground plane. The optical axis of the camera is assumed to be parallel to the ground plane.

The world (plan view) coordinates (X, Z) have now been found in terms of the image coordinates (x, y). Note that y has to be measured from the vanishing point V rather than the top of the image. Note also that as X and Z vary inversely with y , they vary rapidly when y is small, so digitization and other errors will markedly affect the accuracy with which far away objects can be located from the plan view.

When the optical axis of the camera is not parallel to the ground plane, the calculations are best dealt with using homogeneous coordinates as shown in Chapter 18.

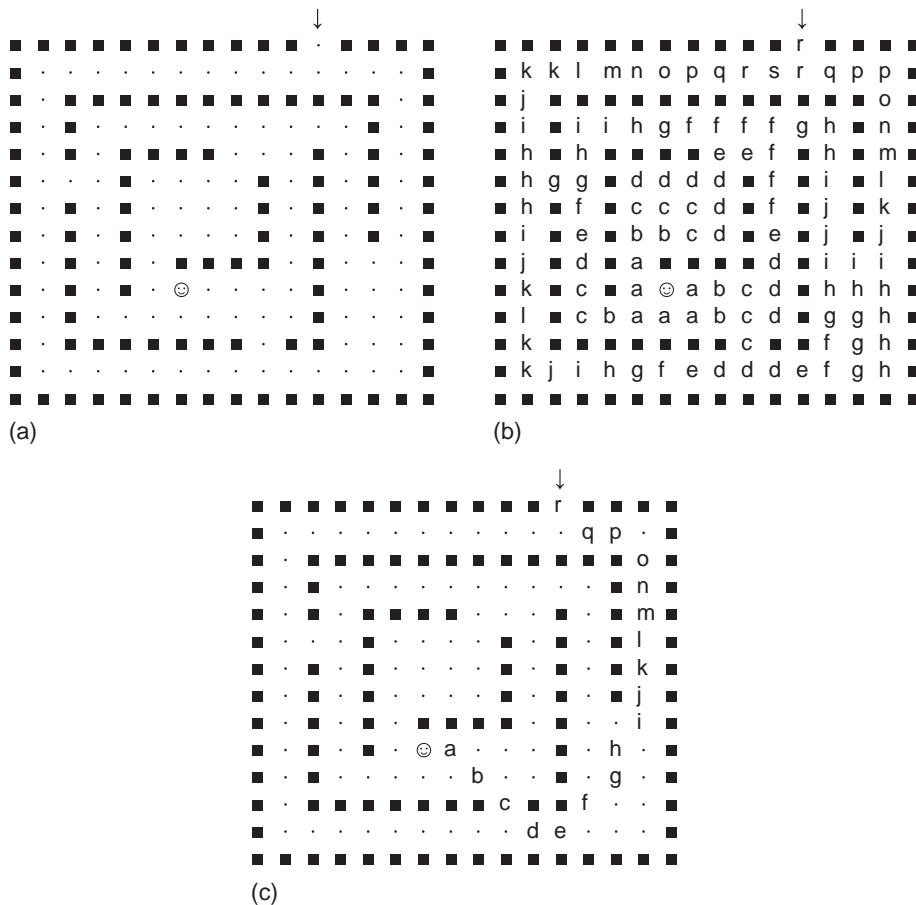
23.8.1 A Simple Path Planning Algorithm

In this subsection, we assume that a plan view of the environment has been built up using the methods of the previous section. While it is by no means clear that humans use an instantaneous plan view model to help them to walk or drive around an environment (an image-based representation seems more likely), it is clear that they use plan views for deductive, logical analysis of the situation and when reading maps. In any case, plan views probably constitute the most natural means for storing navigational knowledge and arriving at globally optimal routes. Here we leave aside conjecture of exactly how humans juggle the information between the two representations, and concentrate on how a robot might reasonably undertake path planning using a plan view it has built up. In fact, a maze-running robot would need to be provided with a suitable algorithm for this purpose.

[Figure 23.14\(a\)](#) shows a simple maze in which the robot has to proceed from the entrance E to the final goal G (respectively marked “ \downarrow ” and “ \odot ” in the figure). We assume that a plan view of the maze has been built up and that a systematic means is needed to find the optimum path to the goal G . The envisaged algorithm starts from G and propagates a distance function over the whole region, constrained only by the walls of the maze ([Fig. 23.14](#)). If a parallel algorithm is used, it is terminated when the distance function arrives at E ; if a sequential algorithm is used, it must carry on until the whole maze has been covered—assuming that an optimal path is required. When the distance function has been completed, finding an optimum path necessitates proceeding downhill along the distance function until G is reached: at each point, the locally greatest gradient must be used (Kanesalingam et al., 1998). Connected components analysis could be used to confirm that a path exists, but a distance function has to be used to guarantee finding the shortest path. Note that the method will find only one of several paths of equal length: these arise because of the limitations of this type of method, which assigns integer values to distances between adjacent pixels.

23.9 VEHICLE GUIDANCE IN AGRICULTURE

In recent years, there has been increasing pressure on farmers to reduce the quantities of chemicals used for crop protection. This cry has come both from

**FIGURE 23.14**

Method for finding an optimal path through a maze. (a) Plan view of maze. (b) Distance function of the maze, starting at the goal (marked ☺), and presenting distance values by successive letters, starting with $a = 1$. (c) Optimum path obtained by tracking from the maze entrance (marked \downarrow) along maximum gradient directions.

environmentalists and from the consumers themselves. The solution to this problem lies in more selective spraying of crops. For example, it would be useful to have a machine that would recognize and spray weeds with herbicides, leaving the vegetable crops themselves unharmed: alternatively, the individual plants could be sprayed with pesticides. This case study relates to the design of a vehicle that is capable of tracking plant rows and selecting individual plants for spraying (Marchant and Brivot, 1995; Marchant, 1996; Brivot and Marchant, 1996; Sanchiz et al., 1996; Marchant et al., 1998).¹

¹Many of the details of this work are remarkably similar to those for the totally independent project undertaken in Australia by Billingsley and Schoenfisch (1995).

FIGURE 23.15

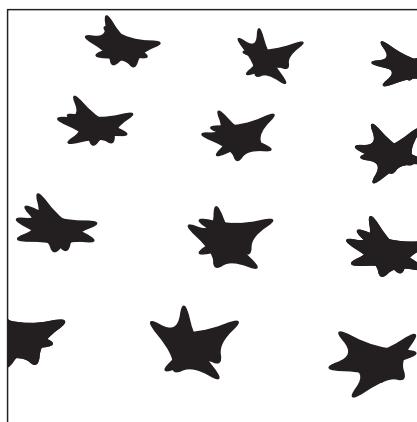
Value of color in agricultural applications. In agricultural scenes such as this, color helps with segmentation and with recognition. It may be crucial in discriminating between weeds and crops if selective robot weedkilling is to be carried out (see also color Plate 1).

Source: © World Scientific 2000

The problem would be enormously simplified if plants grew in highly regular placement patterns, so that the machine could tell from their positions whether they were weeds or plants, and deal with them accordingly. However, the growth of biological systems is somewhat unpredictable and renders such a simplistic approach impracticable. Nevertheless, if plants are grown from seed in a greenhouse, and transplanted to the field when they are approaching 100 mm high, they can be placed in straight parallel rows, which will be approximately retained as they grow to full size. There is then the hope (as in the case shown in Fig. 23.15) that the straight rows can be extracted by relatively simple vision algorithms, and the plants themselves located and identified straightforwardly.

At this stage the main problems are: (1) the plants will have grown to one side or another, and will thus be out of line; (2) some will have died; (3) weeds will have appeared near some of the plants; (4) some plants will have grown too slowly, and will not be recognized as plants. Thus, a robust algorithm will be required to perform the initial search for the plant rows. The Hough transform (HT) approach is well adapted to this type of situation: specifically, it is well suited to looking for line structure in images.

The first step in the process is to locate the plants. This can be achieved with reasonable accuracy by thresholding the input images (this process is eased if

**FIGURE 23.16**

Perspective view of plant rows after thresholding. In this idealized sketch, no background clutter is shown.

Source: © World Scientific 2000

infrared wavelengths are used to enhance contrast). However, at this stage the plant images become shapeless blobs or clumps (Fig. 23.16). These contain holes and lobes (the leaves, in the case of cabbages or cauliflowers), but a certain amount of tidying up can be achieved either by placing a bounding box around the object shape or by performing a dilation of the shape that will regularize it and fill in the major concavities (the real-time solution employed the first of these methods). Then the position of the center of mass of the shape is determined, and it is this that is fed to the HT straight line (plant row) detector. In common with the usual HT approach, votes are accumulated in parameter space for all possible parameter combinations consistent with the input data. Here this means taking all possible line gradients and intercepts for lines passing through a given plant center and accumulating them in parameter space. To help find the most meaningful solution, it is useful to accumulate values in proportion to the plant area. In addition, note that if three rows of plants appear in any image, it will not initially be known which plant is in which row, and therefore each plant should be allowed to vote for all the row positions: this will naturally only be possible if the inter-row spacing is known and can be assumed in the analysis. However, if this procedure is followed, the method will be far more resistant to missing plants and to weeds that are initially mistakenly assumed to be plants.

The algorithm is improved by preferentially eliminating weeds from the images before applying the HT. Weed elimination is achieved by three techniques—hysteresis thresholding, dilation, and blob size filtering. Dilation refers to the standard shape expansion technique described in Chapter 7 and is used here to fill in the holes in the plant blobs. Filtering by blob area is reasonable since the weeds are seldom as strong as the plants, which were transplanted only when they had become well established.

Hysteresis thresholding is a widely used technique that involves use of two threshold levels. In this case, if the intensity is greater than the upper level t_u , the object is taken to be plant; if lower than the lower level t_l , it is taken to be weed; if at an intermediate level and next to a region classified as plant, it is taken to be plant; the plant region is allowed to extend sequentially as far as necessary, given only that there is a contiguous region of intensity between t_l and t_u connecting a given point to a true plant ($\geq t_u$) region. Note that this application is unusual in that whole-object segmentation is achieved using hysteresis thresholding: more usually the technique is used to help create connected object boundaries (see Section 5.10).

Once the HT has been obtained, the parameter space has to be analyzed to find the most significant peak position. Normally, there will be no doubt as to the correct peak—even though the method of accumulation permits plants from adjacent rows to contribute to each peak. The reason for this is that with three rows each permitted to contribute to adjacent peaks, the resultant voting patterns in parameter space are: 1,1,1,0,0; 0,1,1,1,0; 0,0,1,1,1—totaling 1,2,3,2,1—thereby making the true center position the most prominent (actually, the position is more complicated than this as several plants will be visible in each row, thus augmenting the central position further). However, the situation could be erroneous if any plants are missing. It is therefore useful to help the HT arrive at the true central position. This can be achieved by applying a Kalman filter (Section 19.8) to keep track of the previous central positions, and anticipate where the next one will be—thereby eliminating false solutions. This concept is taken furthest in the paper by Sanchiz et al. (1996), where the individual plants are all identified on a reliable map of the crop field and errors from any random motions of the vehicle are systematically allowed for.

23.9.1 3-D Aspects of the Task

So far we have assumed that we are looking at simple 2-D images that represent the true 3-D situation in detail. In practice this is not so. The reason for this is that the rows of plants are being viewed obliquely and therefore appear as straight lines but with perspective distortions, which shift and rotate their positions. The full position can only be worked out if the vehicle motions are kept in mind. In practice, vehicles moving along the rows of plants exhibit variations in speed and are subject to roll, pitch, and yaw. The first two of these motions correspond respectively to rotations about horizontal axes along and perpendicular to the direction of motion: these are less relevant and are ignored here. The last is important as it corresponds to rotation about a vertical axis and affects the immediate direction of motion of the vehicle.

To proceed, we have to relate the position (X, Y, Z) of a plant in 3-D with its location (x, y) in an image. We can achieve this with a general translation:

$$T = (t_x, t_y, t_z)^T \quad (23.9)$$

together with a general rotation:

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \quad (23.10)$$

giving:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad (23.11)$$

The lens projection formulae are also relevant:

$$x = \frac{fX}{Z} \quad (23.12)$$

$$y = \frac{fY}{Z} \quad (23.13)$$

We shall not give a full analysis here, but assuming that roll and pitch are zero, and that the heading angle (direction of motion relative to the rows of plants) is ψ , and that this is small, we obtain a quadratic equation for ψ in terms of t_x . This means that two sets of solutions are in general possible. However, it is soon found that only one solution matches the situation, as the wrong solution is not supported by the other feature point positions. This shows the complications introduced by perspective projection—even when highly restrictive assumptions can be made about the geometrical configuration (in particular, ψ being small).

23.9.2 Real-Time Implementation

Finally, it was found to be possible to implement the vehicle guidance system on a single processor augmented by two special hardware units—a color classifier and a chaincoder. The latter is useful for fast shape analysis following boundary tracking. The overall system was able to process the input images at a rate of 10 Hz, which is sufficient for reliable vehicle guidance. Perhaps more important, the claimed accuracy was in the region of 10 mm and 1° of angle, making the whole guidance system adequate to cope with the particular slightly constrained application considered. A later implementation (Marchant et al., 1998) did a more thorough job of segmenting the individual plants (although still not using the blob size filter), obtaining a final 5 Hz sampling rate—again fast enough for real-time application in the field. All in all, this case study demonstrates the possibility of highly accurate selective spraying of weeds, thereby very significantly cutting down the amount of herbicide needed for crops such as cabbages, cauliflowers, and wheat.

23.10 CONCLUDING REMARKS

This chapter has considered the value of in-vehicle vision as part of the means for providing driver assistance systems. It has also considered the design of such systems. This process is rendered far from trivial because the camera is necessarily moving, so all objects in a scene will appear to be in motion. Hence, it becomes quite difficult to eliminate the background from consideration and less easy to rely on motion-based segmentation. This makes it natural to adopt the alternative approach of placing reliance on recognition of individual objects. [Sections 23.2 and 23.3](#) showed how this concept can be applied to the location not only of the roadway, but also of road markings and road signs. The principle also applied to location of vehicles, but as these vary in appearance, it proved necessary to have several distinct methods for locating them, including under-vehicle shadows, symmetry, wheels, and licence plates (the latter acting not merely as unique vehicle identifiers but also as characteristics of vehicles in general). Curiously, licence plates offered a possible means of finding the orientations of vehicles on roads as well as their locations, although the result was dependent on the relative heights of the camera and licence plate under observation. This meant that, when they are not occluded, tire and wheel location will probably be more accurate indicators of vehicle orientation.

Pedestrian location was also seen to be a challenge—particularly as people are articulated objects, and walk with bobbing motions, and also because they tend to have unique appearances and clothing. This makes it natural to use specific templates for leg, arm, head, and body detection rather than whole-body templates. Here, symmetry is also a possible cue as well as skin color. All these approaches were studied in [Section 23.7](#) and tallied with findings in the literature.

The chapter also included aspects of path planning consequent on projecting vehicles and other obstructions onto a plan view of the ground plane: this has some consequence for robot egomotion and navigation. It is also relevant for guidance of agricultural vehicles that are being used for cultivation, selective spraying, and so on. Here, it is also important to consider the much greater degrees of roll, pitch, and yaw that will be experienced by a tractor or other vehicle moving over ploughed fields, and the visual compensation needed to cope with this. Some indication was given about how these factors have been coped with: because the principles are known, it seemed better for readers to refer to the original papers for further details.

Finally, we should remark on the almost explosive growth of interest in in-vehicle driver assistance systems, particularly since 2000. This is so important that the following section looks very closely at developments in this area and provides separate bibliographies relating to the various aspects. It was felt that it would be clearer presenting these separately once the principles of the subject had been dealt with, as has been done relatively didactically in the preceding sections.

In-vehicle vision systems necessarily deploy moving cameras, so the usual surveillance strategy of eliminating the stationary background becomes difficult to apply. However, considerable success can be achieved using the alternative strategy of directly locating the most relevant objects, such as the roadway, road signs, road markings, vehicles (e.g., *via* their symmetry, shadows, wheel, and licence plates), and pedestrians (e.g., *via* their legs, arms, body, and head). Plan views of the ground plane form useful adjuncts to the information obtained in these ways.

23.11 MORE DETAILED DEVELOPMENTS AND BIBLIOGRAPHIES RELATING TO ADVANCED DRIVER ASSISTANCE SYSTEMS

As indicated earlier in the chapter, in recent years (and particularly since 2000) there has been an almost explosive growth of interest in in-vehicle vision systems. The prime although often unwritten underlying aim has been that of driver assistance—a general term that ultimately includes vehicle guidance. However, in 1998, it at first appeared that Bertozzi and Broggi (1998) had largely solved the problem. In fact, they had laid down many of the ground rules, including finding lane markings with the aid of morphological filters, locating obstacles without constraints on symmetry or shape, analyzing stereo images to find free space on the road ahead, removing the perspective effect, implementing the system on a rapidly operating software plus massively parallel hardware architecture, presenting feedback information to the driver *via* a TV monitor and control panel, testing the system on the road, and above all demonstrating robustness with respect to shadows, changing illumination conditions, varying road texture, and typical motions on the road. Nevertheless, the system was subject to basic assumptions such as the road being flat and road markings being visible; in addition, it placed a great deal of reliance on the stereo system, which had limited range; furthermore, it treated each pair of stereo images individually, and was unable to exploit temporal correlations. Finally, while it never failed to detect vehicles on the road ahead, it sometimes detected false obstacles because of noise arising from the various image remapping processes.

In the light of this work, other workers continued development with increased pace, pressing to eliminate deficiencies with the basic strategy; interestingly, many abandoned the stereo vision approach which brings with it many complications: in fact, appeal to the human vision system demonstrates all too clearly that stereo brings few real advantages for the restricted tasks involved in driving a vehicle (whatever is the case when assembling a gyroscope or other instrument on a workbench). We shall return to this point below.

First, it is worth outlining the findings of Connolly (2009) who has described in a general way the gains to be achieved by advanced driver assistance systems (ADASs). The main keys to success appear to be the provision of lane departure

warnings, help for lane changing, collision avoidance, adaptive cruise control, and driver vigilance monitoring. However, it is important that the ADAS should not give too many warnings, or the driver may become annoyed and deactivate it: neither should it fail to act soon enough or give the driver too much confidence or too much freedom. In fact, it is vital for drowsiness to be detected because approximately 30% of motorway accidents are caused by drivers undergoing micro- or macro-sleeps. While much work has been carried out on blink-rate analysis for detecting these conditions, the method has limited effectiveness in probing the state of the brain itself. Nevertheless, it is clear that vision systems can do much to monitor the driver's behavior, and specifically to monitor his direction of gaze and state of *apparent* awareness. Overall, it is probably in the realm of lane departure warnings and of collision avoidance that an ADAS can do the most good, without annoying the driver. Indeed, in the event of the driver's unawareness of an impending collision, or incapability of acting soon enough, the ADAS should be permitted to act autonomously. While this could in principle be legally contentious, it is not without precedent, as anti-lock braking systems are in common use.

There are many causes of collision, and a large proportion of them are due to driver error, even when drowsiness is not a specific factor. Failure to see a vehicle or pedestrian because of preoccupation with other events on or off the road, failure to estimate speeds or trajectories of vehicles sufficiently accurately, failure to judge how rapidly braking can be performed in the prevailing conditions, and lack of awareness of what other drivers intend to do are all involved in causing accidents: this list does not include gross vehicle malfunctions, such as unpredictable tire bursts. In fact, all these factors arise from or are exacerbated by lack of the right information being available soon enough. Thus, it is obvious that vision has a large part to play in overcoming the problems. While radar, lidar, ultrasonics or other technologies may help, vision provides far more of the right sort of information with the right sort of response rates, and computer vision should be able to cope reliably and rapidly enough to make this possible. The main questions are: What will be the cost? Where will the cameras be placed? Can enough of them be used to ensure that relevant information is made available? Fortunately, cameras are by now so cheap that cost—relative to that of a vehicle or of the damage caused in a crash—is no longer a serious problem. On the other hand, the real problems are the sophistication and speed of the associated software (or in the latter case, how the system is to be implemented in hardware—a consideration that is largely postponed until Chapter 26). For the remainder of the chapter, we therefore concentrate mainly on the sophisticated software aspects and what has been achieved since the turn of the Millennium.

23.11.1 Developments in Vehicle Detection

One area of vital concern has been the detection of other vehicles, especially those overtaking (Zhu et al., 2004; Wang et al., 2005; Hilario et al., 2006; Cherng

et al., 2009). The last of these papers considers patterns of driving, such as “cutting in” after overtaking, but more subtly how interactions between events involving more than two vehicles can cause distractions that prevent optimal actions being taken: this is because not all dynamic obstacles are predictable; in fact, multiple critical situations can occur simultaneously. The paper takes the line that the computer must follow attention patterns that emulate those of the human brain, and concentrate cyclically on eliminating the various critical phases that are being experienced. The necessary dynamic visual model is in this case tackled using a spatiotemporal attention (STA) neural network. The system of Kuo et al. (2011) concentrates on detecting vehicles on the road ahead, but is also able to assess longitudinal distance information and thus to provide adaptive cruise control (albeit no indications of accuracy are given in the paper). Note that this system uses a monocular camera and thus avoids the difficulties of stereo systems mentioned earlier.

Sun et al. (2004, 2006) reviewed the methods used by various workers to detect vehicles. They reported knowledge-based methods using symmetry, color, shadow, corners, horizontal and vertical edges, texture, and lights. In addition, stereo and motion approaches have been used. They also reported template matching and appearance-based methods, and noted that sensor fusion is needed to ensure that sufficient information is brought to bear to make vehicle detection reliable. They emphasized that hypothesis generation and verification are important for obtaining reliable solutions. Overall, they offered no silver bullet solution, apart from sensor fusion, although (looking at their conclusions as a whole) *method* fusion appears to be rather more important. Among the worst challenges they found were those of “all hours—all weathers” operation. In particular, bad illumination (especially at night) and the results of rain and snow will affect many well-known algorithms for vehicle detection, including those based on shadows. While in principle vehicle lights should provide an easy way of detecting vehicles, in the dark they can prove confusing, especially when rain-soaked roads cause reflections. Sun et al. therefore “believe that these cues have limited employability.” However, there are bound to be conditions under which some methods will not work well, but by using method fusion in a dynamic way, giving different methods different weights in different conditions, viable solutions should in the end be obtainable. Whereas humans could be confused in dark situations where no information at all is available, it is difficult to imagine them not being able to solve vehicle detection problems because of rain, snow, or random reflections, and certainly not simply because no shadows are visible.

While the difficulties of dealing with the problems of driving at speed on a motorway can be hugely complicated, with vehicles overtaking on either side and sometimes cutting in, the solution is often to drive more slowly thereby minimizing risks and lowering the data rate to manageable levels. However, the problems of dealing with pedestrians are considerably more complicated. This is because, in contrast to the case of vehicles that travel at more or less constant speeds in constant directions for considerable periods of time—and also have a fair amount

of free space immediately around them—pedestrians are unpredictable, sometimes running to get across roads between vehicles, sometimes jay-walking, and sometimes moving in groups having even more unpredictable behavior. A basic problem is that it is unknown when a stationary pedestrian might suddenly move into the roadway, and with a temporary acceleration that exceeds that of most vehicles. Hence, a great many workers have been, and are producing algorithms for pedestrian detection and tracking.

23.11.2 Developments in Pedestrian Detection

Geronimo et al. (2010) have recently reviewed pedestrian detection systems for ADASs. As their paper is very thorough and contains 146 references, the reader is recommended to work carefully through it. Nevertheless, some useful points can be made here. They emphasize that pedestrians exhibit high variability in size, pose, clothing, objects carried, and so on; they appear in cluttered scenes, can be partially occluded, and may be in poor contrast regions; they have to be identified in dynamically varying scenes when both they and the camera are moving; they often appear radically different when viewed from different directions. Geronimo et al. note that silhouette matching, e.g., using the chamfer matching technique, is widely used for detection, yet it needs to be augmented by an additional appearance-based step. (This is not an argument against silhouette matching, but one for using it as a cue in accordance with the idea expressed above that method fusion is required—i.e., method redundancy is needed to cope robustly with *real* scenes containing substantial clutter.) Geronimo et al. (2010) underline the need for verification and refinement. Interestingly, they note that the Kalman filter is (still) by far the most heavily used tracking algorithm—a surprising fact considering that pedestrian motions along pavements, in precincts or crossing the road exhibit far from steady motion (in fact their motions tend to be jerky and indecisive, as they find their way around obstacles and other people). Finally, Geronimo et al. emphasize the need for all hours—all weathers performance; here they note that NIR imaging gives pictures not dissimilar to visible light images, so similar algorithms can be used for analysis. This is less true for thermal (far infrared or FIR) images, which are commonly called “night vision.” In any case the latter respond to relative temperature, which is useful for distinguishing hot targets, including pedestrians for vehicles, but inappropriate for examining most of the background or objects such as road signs. Thus, thermal cameras need to be backed up with visible light cameras in the day or NIR cameras in the night, and so would generally constitute an unnecessary expense.

Gavrila and Munder (2007) describe a multi-cue pedestrian detection system: after extensive field tests in difficult urban traffic conditions, they reasonably claim it to be at the (2007) leading edge. The four main detection modules are sparse stereo-based ROI generation, shape-based detection, texture-based classification, and verification using dense stereo, these being complemented by a tracking module. In fact, the paper builds on earlier work (Gavrila et al., 2004), and its

main contributions are the method of integration into a multi-cue system for pedestrian detection and a systematic ROC-based procedure for parameter setting and system optimization. In part, the success of the system is due to the use of a novel mixture-of-experts architecture for shape and texture-based classification: here the idea is to take the known shape information and to use texture to partition the feature space into regions of reduced variability—a process that matches well the types of clothing worn by humans. Importantly, the approach using a texture-based mixture-of-experts weighted by the outcome of shape matching was found to outperform an approach based on single texture classifiers. Also notable is the (continued) use of chamfer matching for shape detection, prominent in much of Gavrila's earlier work.

It was remarked earlier that stereo adds considerable complication to a vision system, which may not be justified for an in-vehicle system when most of the objects being viewed will be many meters away. This makes it no surprise that the review article by Enzweiler and Gavrila (2009) concentrates on monocular pedestrian detection. The paper also included descriptions of a number of experimental comparisons of methods for pedestrian detection. Apart from temporal integration and tracking, methods that were tested included the following: (1) Haar wavelet-based cascades, (2) neural networks using local receptive fields, (3) histograms of orientated gradients (HOGs) together with linear SVM classifiers, and (4) combined shape and texture-based approaches. The fourth of these was subsequently disregarded as its main advantage was processing speed, which was not considered relevant to the comparison. The investigation found that the HOG approach outperformed the wavelet and neural network approaches (Section 22.16 contains a brief outline of the HOG approach and also explains why it outperforms the wavelet approach in this type of application). In particular, at a sensitivity of 70%,² the respective false positive rates were 0.045, 0.38, and 0.86, representing huge reduction factors for false positives. Similarly, at a sensitivity of 60%, the precision rates were vastly improved for the HOG approach, particularly relative to the neural network approach. It should be emphasized that these results apply for intermediate resolutions with pedestrian images $\sim 48 \times 96$ pixels, while earlier low-resolution work with pedestrian images $\sim 18 \times 36$ pixels led to Haar wavelets being the most viable option. Overall, there seemed to be slight doubt about what the critical factors actually are: in particular, the authors state “perhaps it is the data that matters most, after all,” meaning that increased performance may be at least partly due to increases in the size of the training set. In addition, quite a bit depends on the processing constraints that are applied, and for tighter constraints the Haar wavelet approach comes back into its own. However, as ever, it is difficult to standardize or specify image data, or *a fortiori*,

²This assumes that the term “detection rate” used by the authors actually means “sensitivity” (or “recall”): see Chapter 24. In this paragraph, note that sensitivity gives a reverse measure of false negative rate, $1 - FN/(TP + FN)$, while precision gives a reverse measure of false positive rate, $1 - FP/(TP + FP)$.

image sequence data, so this paper is not able to tell the whole story. Finally, it should be noted that at this point in time, shape-based detection, and in particular the chamfer matching approach, has dropped out of sight because its main advantage was that of speed, and here recognition accuracy measures were the main performance criteria.

Looking back to the work of Curio et al. (2000)—who use Hausdorff distance rather than chamfer matching for template matching—the attention is very much on analyzing limb movements, modeling human walking, and observing human gait patterns. However, they note that the upper body shows a high degree of variation in its appearance, so it is better to restrict pedestrian detection to the lower body: in fact this strategy is both more reliable and more computationally efficient. They also point out that exact modeling is more complicated for women wearing skirts. (A similar situation must apply for men wearing robes or mackintoshes.) Overall, just as the driver is aware of motion and gait as well as the body models of pedestrians, these need to be incorporated into practical pedestrian detection algorithms in order to provide maximum reliability and robustness.

Zhang et al. (2007) performed tests on pedestrian detection in “IR images” (these were actually thermal images taken with a camera operating in the spectral range 7–14 μm). Their motivation was to make a system that was capable of working at night time, although they also noted that many undesirable activities occur at night or in relative darkness, so the methodology should be useful in other applications as well. They found that IR images are by no means dissimilar to visible light images, so similar algorithms can be used for analyzing them: i.e., there is no need to invent radically different methods for the IR domain. In particular, they found that edgelet and histogram of orientated gradients (HOG) methods (see Dalal and Triggs, 2005) could be adapted to work with IR images, and similarly for boosting and SVM cascade classification methods (Viola and Jones, 2001). Hence they achieved detection performance for IR images comparable to state-of-the-art results for visible light. The underlying reason for this seemed to be that IR and visible light lead to similar silhouettes.

23.11.3 Developments in Road and Lane Detection

Zhou et al. (2006) developed a lane detection and tracking system using a monocular monochromatic camera. They used a deformable template model to initially locate the lane markings, with tabu search for optimal location; then they used a particle filter for tracking the markings. Their experimental results showed that the resulting system was robust against broken lane markings, curved lanes, shadows, distracting edges, and occlusions. Kim (2008) also used a particle filter for tracking lane markings, but employed RANSAC for initial detection. Similarly, Mastorakis and Davies (2011) used RANSAC for detection but modified it for increased reliability, as described in Sections 11.6 and 23.3: see also Borkar et al. (2009). Finally, Marzotto et al. (2010) showed how a RANSAC-based system

could be implemented in real time using an FPGA platform: for more details, see Chapter 26.

While the above approaches are suitable for urban roads, which normally have well-defined lane markings, many roads, especially in rural regions, are unstructured and lacking in markings—and the road boundaries may be overgrown with vegetation. Cheng et al. (2010) devised a system with the ability to handle both structured and unstructured types of road using a monocular camera. To achieve this they devised a hierarchical lane detection strategy which was able to achieve high accuracy using quite simple algorithms. First, environment classification of pixels was carried out with high dimensional feature vectors using eigenvalue decomposition regularized discriminant analysis (EDRDA). For unstructured roads, mean-shift segmentation was used and then road boundary candidates were selected from the region boundaries: Bayes rule was used to select the most probable of these as actual boundaries. When the vehicle moved from one type of road to another, the environment classifier indicated that a different algorithm should be used so that accuracy could be maintained.

There is one way in which road and lane mapping schemes are restricted—namely, by the view available from the chosen camera. Typically, this will give an overall viewing angle of up to $\sim 45^\circ$. In fact, ideally, a vehicle-borne camera should have a full 360° viewing angle, so that overtaking vehicles and pedestrians about to approach from the side can be seen clearly. Omnidirectional (catadioptric) cameras may be the best answer to this problem, and many workers are actively pursuing this possibility. Cheng and Trivedi (2007) tested a system which used an omnidirectional camera for the dual tasks of lane detection and monitoring the head pose of the driver (the reason for monitoring head pose is to check that the driver is aware of the situation on the road). Their tests showed that accuracy of lane detection is reduced by a factor of (only) 2–3 because of the reduced resolution available with this sort of camera. Thus, it should prove possible to make savings in the numbers of sensors employed in practical implementations.

23.11.4 Developments in Road Sign Detection

It is a sign of the seriousness with which ADASs are nowadays being taken that a good many papers describing research into the detection and recognition of road signs have been published since the turn of the Millennium. Fang et al. (2003) describe a system that uses neural networks for detecting and tracking road signs by their color and shape. The shapes considered are circles, triangles, octagons, diamonds, and rectangles. Initial detection takes place at some distance, where the road signs appear small and relatively undistorted, and tracking is carried out by a Kalman filter. At each distance, due account is taken of changes in size and shape due to increasing projective distortions, and when a potential sign has become large enough the system verifies that it is a road sign or discards it.

Actual recognition is not discussed in the paper, but detection and tracking are said to be accurate and robust: although speed was slow on a single PC, the neural networks could conveniently be run in parallel on other processors. A related paper by Fang et al. (2004) describes the types of neural network used in this sort of application. Kuo and Lin (2007) describe a similar system, again involving use of neural networks. The latter paper makes use of greater amounts of structural analysis of the images at the detection stage, e.g., using corner detection, Hough transforms and morphology. De la Escalera et al. (2003) describe a system which starts the analysis using color classification, uses genetic algorithms for narrowing down the search, and employs neural networks for sign classification.

McLoughlin et al. (2008) describe practically orientated work on road sign detection and also on the detection of “cats eyes.” Their aim is to assess the road signage quality rather than to use it, and to this end they relate the signs to GPS information. They focus particularly on reflectivity aspects of the signs and are able to detect defective road studs and road signs. Their system is fully autonomous and thus the methodology is largely transferrable to ADASs.

Prieto and Allen (2009) describe a vision-based system for detecting and classifying traffic signs using self-organizing maps (SOM)—a type of neural network. A two-stage detection process is adopted—of first detecting potential road signs by analyzing the distribution of red pixels within the image, and then identifying the road signs from the distribution of dark pixels in their central pictograms. The HT approach and other structural analysis approaches were eschewed because they were felt to operate too slowly for (efficient) real-time operation, so the SOM approach was adopted. To achieve recognition of the pictogram, it was divided into 16 blocks arranged in the form of a triangle (or whatever shape the particular sign was found to possess). It was found necessary to normalize brightness over the region of the sign. The hardware of the embedded machine vision used for this application was a hybrid consisting of an FPGA together with a digital implementation of a SOM. Experiments showed that the system had good performance, being able to tolerate substantial changes in position, scale, orientation and partial occlusion of the road signs, and also being trainable, at least to within the model of colored surround and black on white pictograms. For further details of the SOM and the hybrid implementation, see the original paper and the references mentioned therein.

Ruta et al. (2010) have developed a system not based on neural networks (as for many of the above) but on color distance transforms, coupled with a nearest neighbor recognition system. The color distance transform is actually a set of three distance transforms, one for each color (RGB). If a particular color is absent during testing, it is accorded a maximum distance value of 10 pixels to avoid confusing the system. The color distance transform was tested for dependence on a variety of conditions, such as strong incident light, reflections, and deep shade, and was found to be robust to substantial illumination changes. Perhaps more important, it was found to be reasonably invariant to the effects of affine transformations, which a moving camera would be subject to. This is almost certainly because chamfer matching is subject to graceful degradation as distortions occur,

so the distances at any template (edge) locations will *gradually* increase with the changing levels of distortion. When compared with other methods, the method performed well, the percentages of correct classifications being 22.3 for HOG/PCA, 62.6 for Haar/AdaBoost, 74.5 for HOG/AdaBoost, and 74.4 for the new method using the color distance transform. The main competitor to the new method, HOG/AdaBoost, offers an elegant solution but is much more complex than the new method, and did not outperform it in any real sense. Hence the new method seemed well adapted to the task it was set.

23.11.5 Developments in Path Planning, Navigation, and Egomotion

The subjects of vehicle guidance and egomotion date from as long ago as 1992 (Brady and Wang, 1992; Dickmanns and Mysliwetz, 1992), while automatic visual guidance in convoys dates from a similar period (Schneiderman et al., 1995; Stella et al., 1995). Mobile robots and the need for path planning were discussed by Kanesalingam et al. (1998) and by Kortenkamp et al. (1998), and later a survey was carried out by DeSouza and Kak (2002): see also Davison and Murray (2002). Guidance of outdoor vehicles, particularly on roads, has undergone increasingly rapid development: see for example Bertozzi and Broggi (1998), Guiducci (1999), Kang and Jung (2003), and Kastrinaki et al. (2003). Zhou et al. (2003) considered the situation for elderly pedestrians—although clearly such work could also be relevant for blind people or wheelchair users. Hofmann et al. (2003) showed that vision and radar can profitably be used together to combine the excellent spatial resolution of vision with the accurate range resolution of radar.

In spite of the evident successes, there is still only a limited number of fully automated visual vehicle guidance systems in everyday use. The main problem would appear to be *potential* lack of the robustness and reliability required to trust the system in “all hours—all weathers” situations—although there are also legal implications for a system that is to be used for control rather than merely for vehicle monitoring.

23.12 PROBLEM

1. Check that the path through the maze shown in Fig. 23.14(c) is optimal, (a) by a hand calculation and (b) by a computer calculation. Confirm that several other paths are also optimal. Obtain a more accurate result by taking the horizontal and vertical neighbors of any pixel as being 2 units away, and taking diagonal neighbors as being 3 units away.