CHAPTER

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).

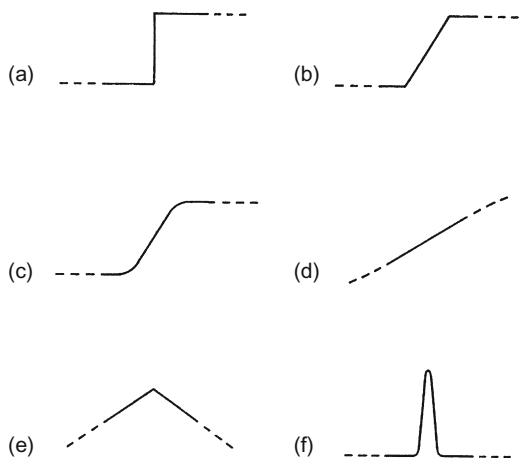

FIGURE 5.1

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:

$$g = \max\,(g_i : i = 1, \ldots, n) \tag{5.1}$$

where n is usually 8 or 12.

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

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

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

$$g = |g_x| + |g_y| \tag{5.3}$$

or

$$g = \max\left(|g_x|, |g_y|\right) \tag{5.4}$$

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:

$$\theta = \arctan\frac{g_y}{g_x} \tag{5.5}$$

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:

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

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

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

If vector addition is to be valid, then:

$$g_{45} = \frac{g_0 + g_{90}}{\sqrt{2}} \tag{5.9}$$

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

$$C = \frac{B}{\sqrt{2}} \tag{5.10}$$

$$D = A\sqrt{2} \tag{5.11}$$

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:

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

where $t = \tan 22.5°$, so that:

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

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

	M_x	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):

$$\tan\theta = \frac{2\tan\varphi}{3-\tan^2\varphi} \qquad 0^\circ \le \varphi \le 45^\circ \tag{5.14}$$

leads to the following formula:

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

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.

r	rms error
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

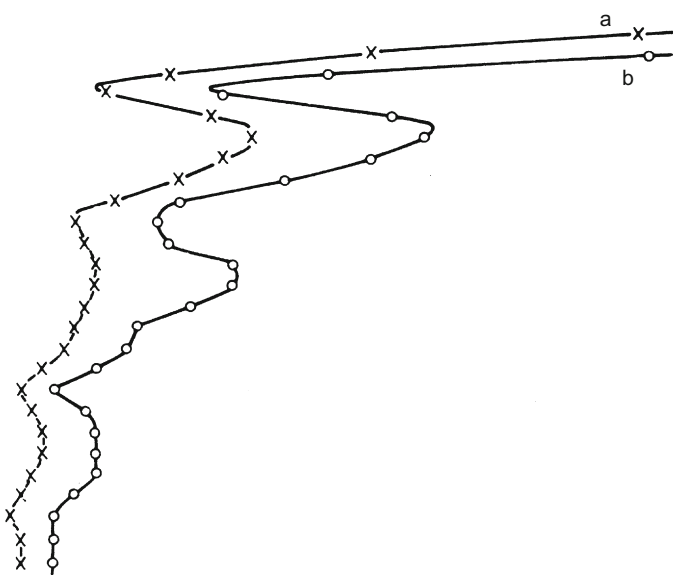

FIGURE 5.2

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.*
[a]*Values are accurate to within ~0.02° 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:

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

Furthermore, for a 3×3 operator of the form

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

applied to the edge

$$\begin{bmatrix} a & a + h(0.5 - E\sec\theta + \tan\theta) & a + h \\ a & a + h(0.5 - E\sec\theta) & a + h \\ a & a + h(0.5 - E\sec\theta - \tan\theta) & a + h \end{bmatrix}$$

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

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

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:

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

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

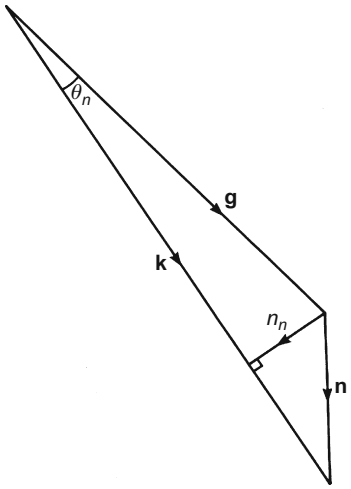

FIGURE 5.3

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,

[1]In 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.

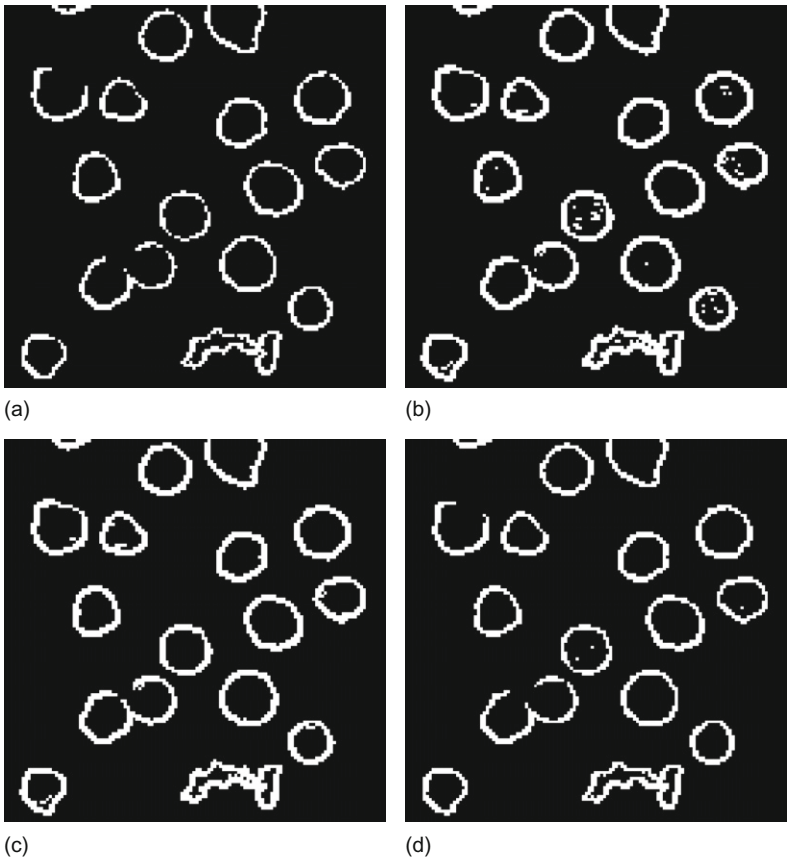

FIGURE 5.4

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:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{5.19}$$

where

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{5.20}$$

and

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{5.21}$$

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
0.008	0.250	2.000	4.000	2.000	0.250	0.008
0.004	0.125	1.000	2.000	1.000	0.125	0.004
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)

1	2	1
2	4	2
1	2	1

(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:

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

where

$$l_1 = \tan \theta \tag{5.23}$$

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

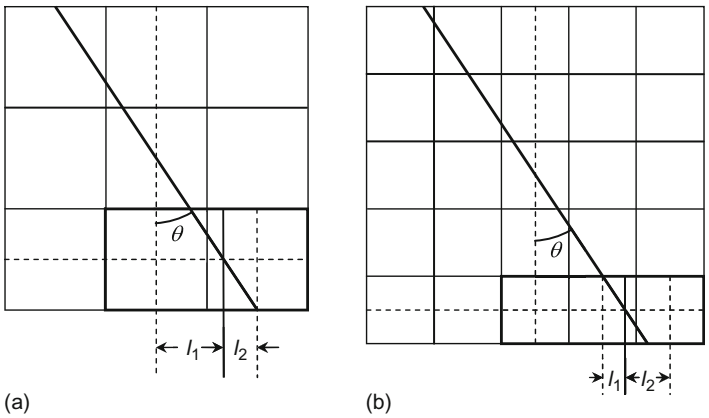

FIGURE 5.6

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:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{5.24}$$

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

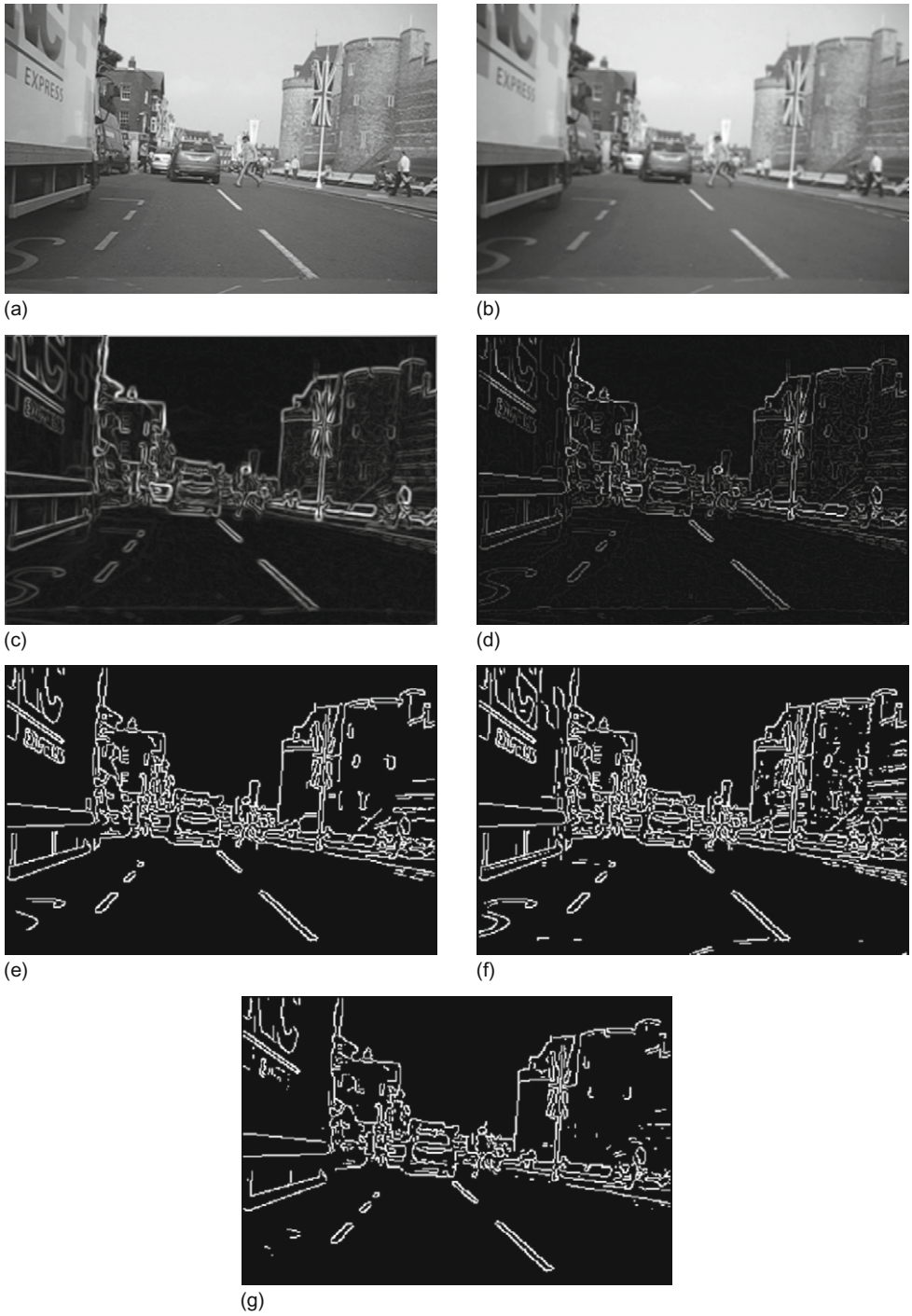

FIGURE 5.7

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.

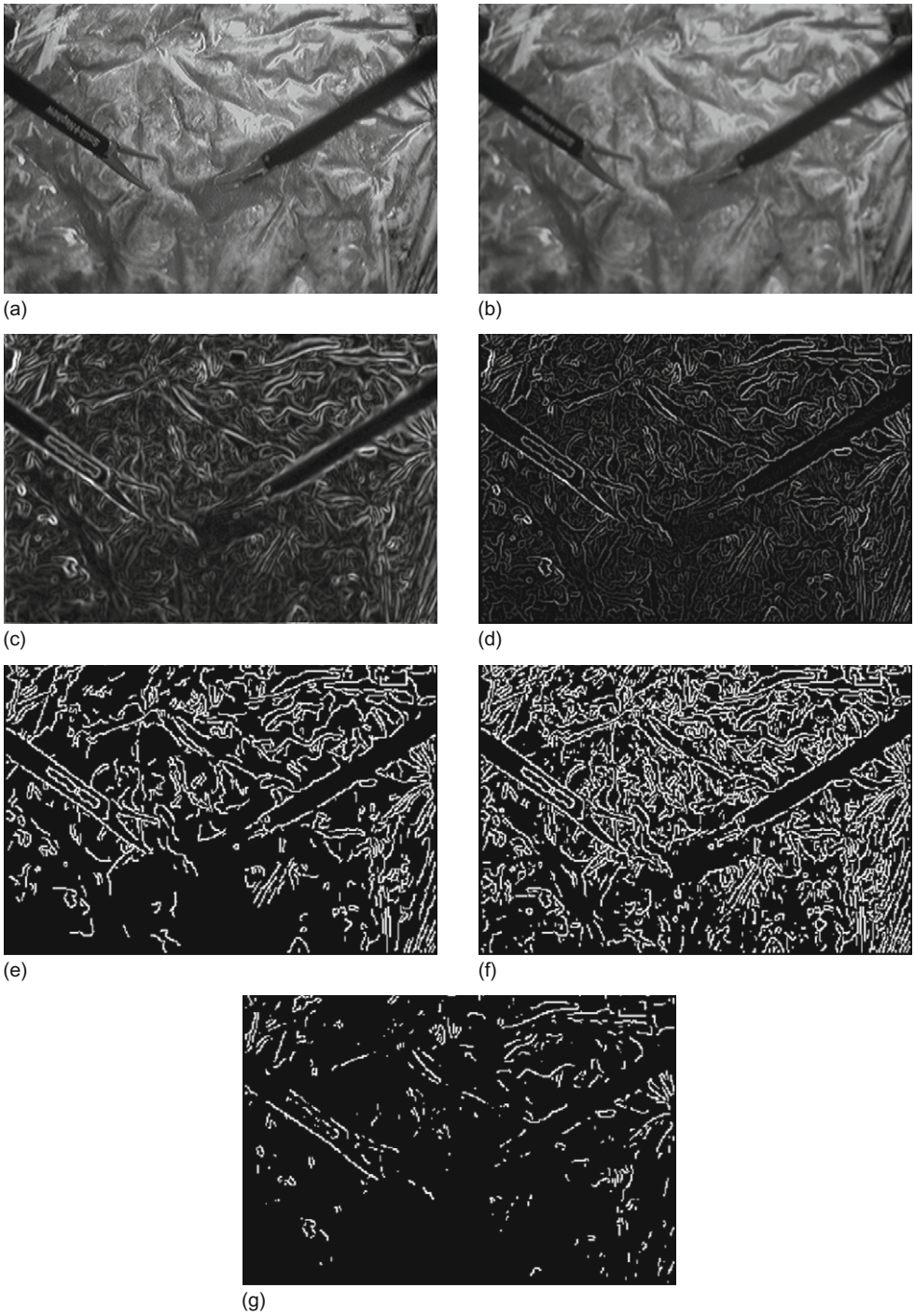

FIGURE 5.8

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:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{5.25}$$

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*

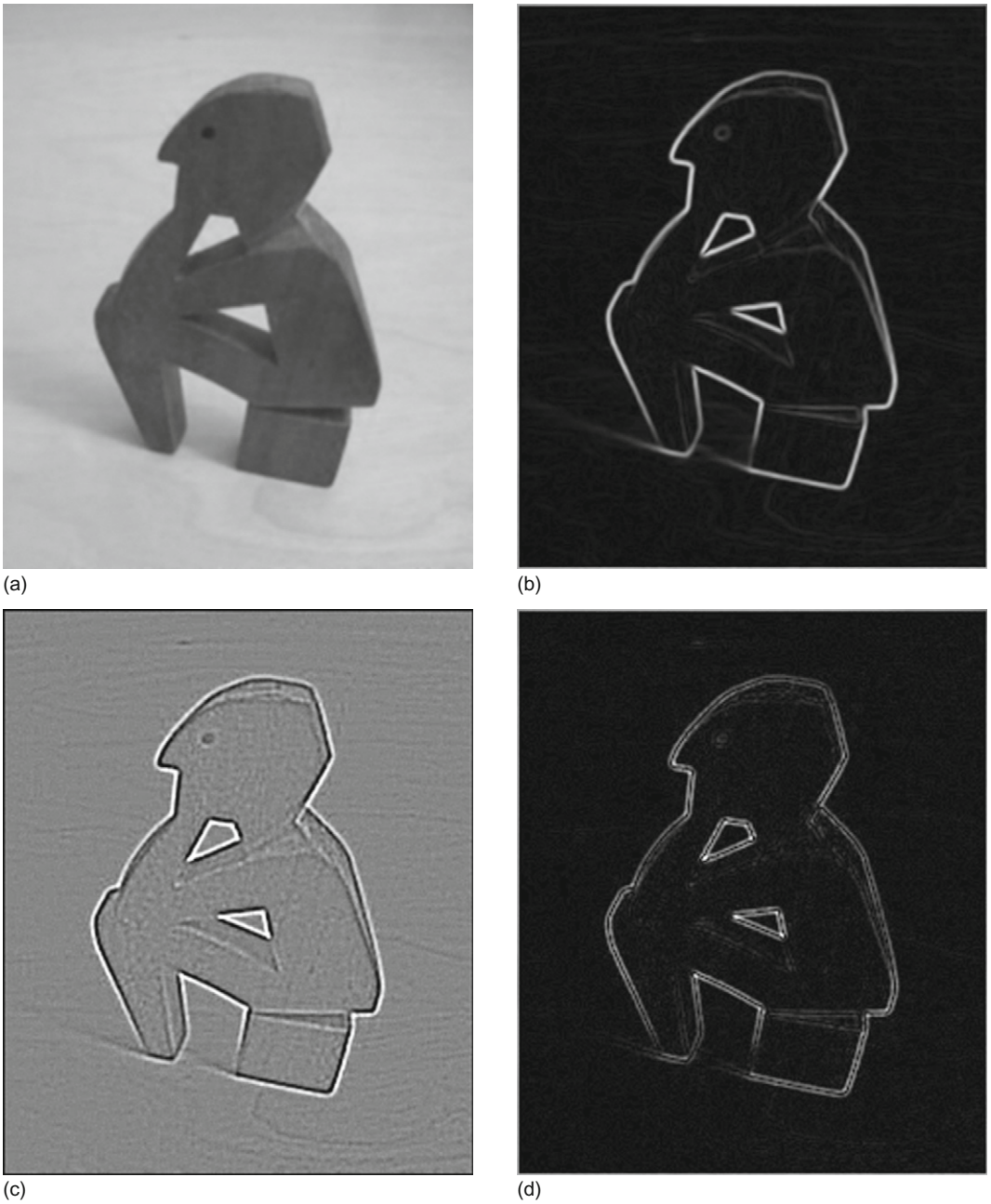

FIGURE 5.9

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:

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

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:

$$E_{\text{stretch}} = \int \kappa(s) \left\| \mathbf{x}_s(s) \right\|^2 \mathrm{d}s \tag{5.27}$$

and

$$E_{\text{bend}} = \int \lambda(s) \left\| \mathbf{x}_{ss}(s) \right\|^2 \mathrm{d}s \tag{5.28}$$

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 $\left|\text{grad } I\right|$, leading to:

$$E_{\text{edge}} = -\int \mu(s) \left\| \text{grad } I \right\|^2 \mathrm{d}s \tag{5.29}$$

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:

$$E_{\text{stretch}} = \sum_{i=1}^{N} \kappa \left\| \mathbf{x}_i - \mathbf{x}_{i+1} \right\|^2 \tag{5.30}$$

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

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

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:

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

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

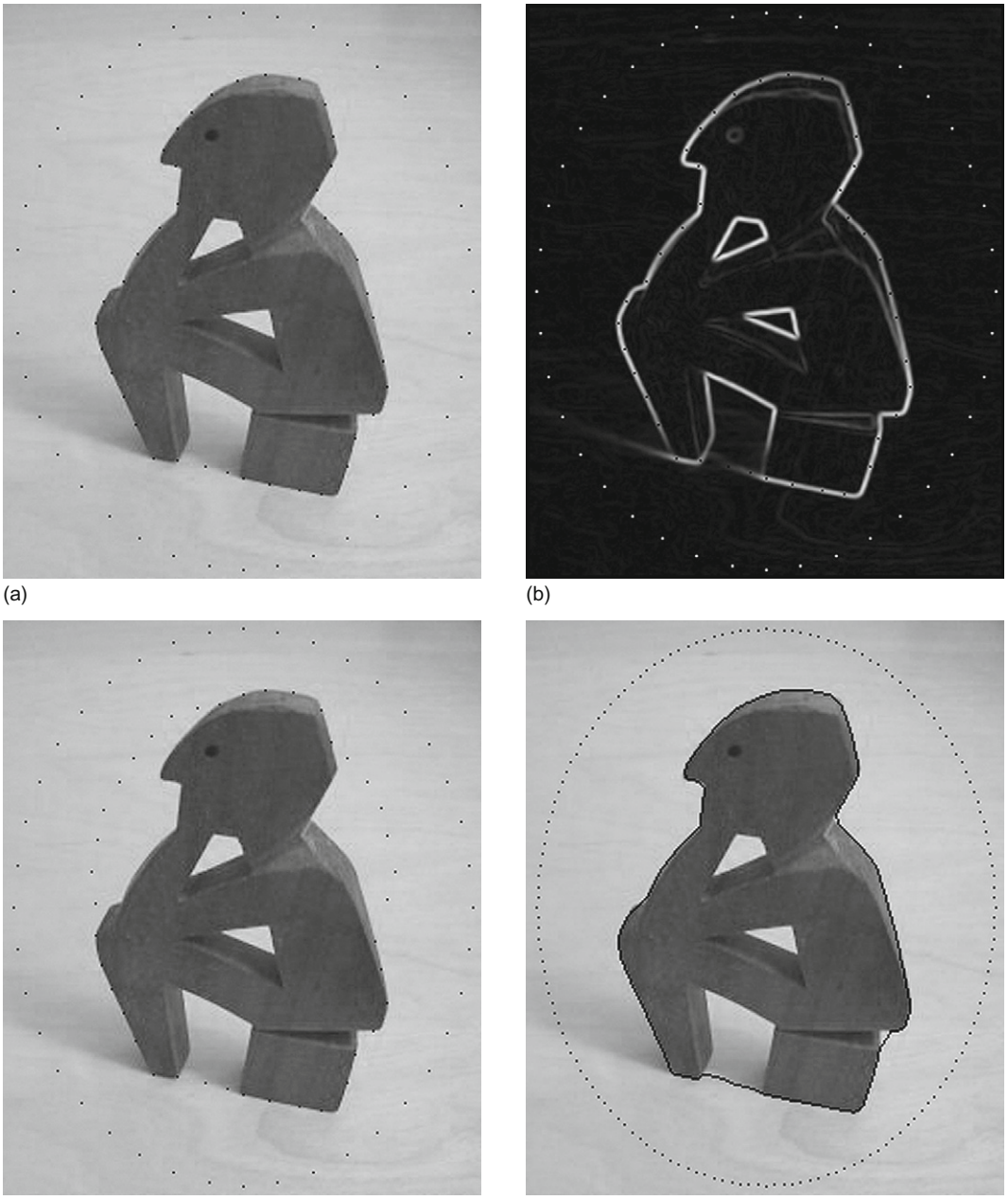

FIGURE 5.10

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:

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

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

$$\varphi(C(t), t) = 0 \tag{5.34}$$

which leads to:

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

Substituting for $\mathbf{n}$ using:

$$\mathbf{n} = \frac{\nabla\varphi}{|\nabla\varphi|} \tag{5.36}$$

we obtain:

$$\frac{\partial \varphi}{\partial t} = -|\nabla\varphi|F \tag{5.37}$$

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

$$\frac{\partial \varphi}{\partial t} = |\nabla \varphi| \, \mathrm{div}\left(g(I) \frac{\nabla \varphi}{|\nabla \varphi|} \right) \tag{5.38}$$

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 ith pixel intensity by I_i, the flow capacity can be minimized at edges

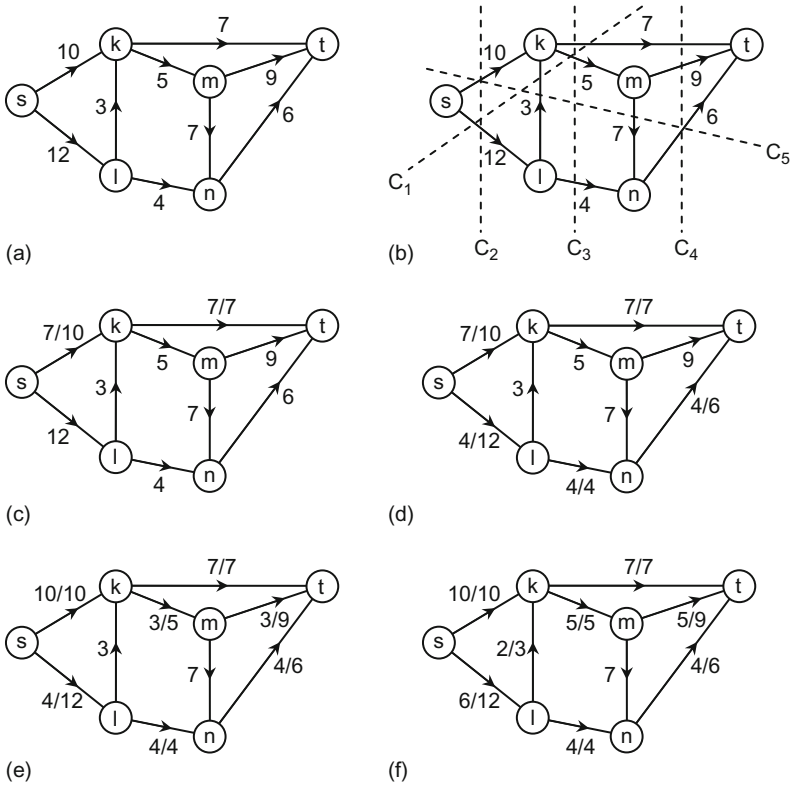

FIGURE 5.11

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 C1–C5 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 C3 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 C1 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:

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

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).