Blum, H. 1967 'A transformation for extracting new descriptors of shape', in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn (ed.), MIT Press Cambridge, Massachusetts, pp. 153–71.

Brady, M. and Asada, H. 1984 'Smoothed local symmetries and their implementation', *The International Journal of Robotics Research*, Vol. 3, No. 3, pp. 36–61.

Castleman, K.R. 1979 *Digital Image Processing*, Prentice Hall, New York.

Gonzalez, R.C. and Wintz, P. 1977 *Digital Image Processing*, Addison-Wesley, Reading, Massachusetts.

Hall, E.L. 1979 *Computer Image Processing and Recognition*, Academic Press, New York.

Hilditch, C.J. 1983 'Comparison of thinning algorithms on a parallel processor', *Image and Vision Computing*, Vol. 1, No. 3, pp. 115–32.

Kenny, P.A., Dowsett, D.J., Vernon, D. and Ennis J.T. 1990 'The application of spatial warping to produce aerosol ventilation images of the lung immediately after perfusion with the same labelled isotope', *Physics in Medicine and Biology*, Vol. 35, No. 5, 679–85.

Motzkin, Th. 1935 'Sur Quelques Proprietes Caracteristiques des Ensembles Bornes Non Convexes', *Atti. Acad. Naz. Lincei*, 21, pp. 773–9.

Nackman, L.R. and Pizer, S.M. 1985 'Three dimensional shape description using the symmetric axis transform 1: Theory', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 2, pp. 187–202.

Pratt, W.K. 1978 *Digital Image Processing*, Wiley, New York.

Rosenfeld, A. and Kak, A. 1982 *Digital Picture Processing*, Academic Press, New York.

Rosenfeld, A. 1975 'A characterization of parallel thinning algorithms', *Information and Control*, Vol. 29, pp. 286–91.

Serra, J. 1982 *Image Analysis and Mathematical Morphology*, Academic Press, London.

Tamura, H. 1978 'A comparison of line-thinning algorithms from a digital geometry viewpoint', *Proceedings 4th International Joint Conference on Pattern Recognition*, pp. 715–19.

Zhang, T.Y. and Suen, C.Y. 1984 'A fast parallel algorithm for thinning digital patterns', *Communications of the ACM*, Vol. 27, No. 3, pp. 236–9.

5

The segmentation problem

5.1 Introduction: region- and boundary-based approaches

Segmentation is a word used to describe a grouping process in which the components of a group are similar with respect to some feature or set of features. The inference is that this grouping will identify regions in the image which correspond to unique and distinct objects in the visual environment.

There are two complementary approaches to the problem of segmenting images and isolating objects: boundary detection and region growing. Region growing effects the segmentation process by grouping elemental areas (in simple cases, individual image pixels) sharing a common feature into connected two-dimensional areas called regions. Such features might be pixel grey-level or some elementary textural pattern, e.g. the short thin bars present in a herringbone texture.

Boundary-based segmentation is concerned with detecting or enhancing the boundary pixels of objects within the image and subsequently isolating them from the rest of the image. The boundary of the object, once extracted, may easily be used to define the location and shape of the object, effectively completing the isolation.

An image comprising boundaries alone is a much higher level representation of the scene than is the original grey-scale image, in that it represents important information explicitly. In cases where the boundary shape is complicated, the gap between these two representations is wide and it may be necessary to introduce an intermediate representation which is independent of the shape. Since boundaries of objects are often manifested as intensity discontinuities, a natural intermediate representation is composed of local object-independent discontinuities in image intensity, normally referred to as 'edges'. The many definitions of the term *edge* can be summarized by the observation (or premise) that an edge occurs in an image

if some image attribute (normally image intensity) changes its value discontinuously. In particular, edges are seen as local intensity discontinuities while boundaries are global ones. The usual approach to segmentation by boundary detection is to first construct an edge image from the original grey-scale image, and then to use this edge to construct the boundary image without reference to the original grey-scale data by edge linking to generate short-curve segments, edge-thinning, gap-filling, and curve segment linking, frequently with the use of domain-dependent knowledge. This association of local edges is normally referred to as *boundary detection* and the generation of the local edge image is referred to as *edge detection*.

Boundary detection algorithms vary in the amount of domain-dependent information or knowledge which they incorporate in associating or linking the edges, and their effectiveness is obviously dependent on the quality of the edge image. The more reliable the edge elements in terms of their position, orientation and, indeed, authenticity, the more effective the boundary detector will be. However, for relatively simple, well-defined shapes, boundary detection may become redundant, or at least trivial, as edge detection performance improves. Since computational complexity for the segmentation process as a whole is a function of the complexity of both the edge detection and boundary detection then minimal segmentation complexity may be achieved by a trade-off between the sophistication of the edge detector and the boundary detector.

It is worth noting, however, that since edge detection is essentially a filtering process and can often be effected in hardware, while boundary detection will require more sophisticated software, the current (and, probably, correct) trend is to deploy the most effective and sophisticated edge detector (e.g. the Canny operator or the Marr–Hildreth operator) and to simplify the boundary detection process.

The remainder of this chapter is devoted to a discussion of a region-based segmentation technique (thresholding), edge detection, region growing and, finally, boundary detection.

5.2 Thresholding

Grey-level thresholding, which we covered briefly in Chapter 4, is a simple region-based technique. However, we include it in a section on its own here because it is a very commonly used and popular technique. As we saw in Chapter 4, in situations where an object exhibits a uniform grey-level and rests against a background of a different grey-level, thresholding will assign a value of 0 to all pixels with a grey-level less than the threshold level and a value of 255 (say) to all pixels with a grey-level greater than the threshold level. Thus, the image is segmented into two disjoint regions, one corresponding to the background, and the other to the object.

5.2.1 Global, local, and dynamic approaches

In a more general sense, a threshold operation may be viewed as a test involving some function of the grey-level at a point, some local property of the point, e.g. the average grey-level over some neighbourhood, and the position of the point in the image. Thus, a threshold operation may be viewed as a test involving a function T of the form:

$$T(x, y, N(x, y), g(x, y))$$

where $g(x, y)$ is the grey-level at the point (x, y) and $N(x, y)$ denotes some local property of the point (x, y). If $g(x, y) > T(x, y, N(x, y), g(x, y))$ then (x, y) is labelled an object point, otherwise it is labelled a background point, or conversely. This is the most general form of the function T, however, and three classes of thresholding (global, local, and dynamic) may be distinguished on the basis of restrictions placed on this function (see Weszka, 1978). These are:

$T = T(g(x, y))$	Global thresholding: the test is dependent only on the grey-level of the point.
$T = T(N(x, y), g(x, y))$	Local thresholding: the test is dependent on a neighbourhood property of the point and on the grey-level of the point.
$T = T(x, y, N(x, y), g(x, y))$	Dynamic thresholding: the test is dependent on the point coordinates, a neighbourhood property of the point and on the grey-level of the point.

It is pertinent to note, however, that most systems utilize the simplest of these three approaches, global thresholding: the threshold test is based exclusively on the global threshold value and on the grey-level of a test-point, irrespective of its position in the image or of any local context. The approach is facilitated either by constraining the scene to ensure that there is no uneven illumination or by photometrically decalibrating the image before thresholding. The advantage of this approach is that the thresholding can be accomplished by commonly available hardware using a look-up table, as described in Chapter 4.

5.2.2 Threshold selection

The selection of an appropriate threshold is the single major problem for reliable segmentation. Of the several techniques which have been proposed, most are based on the analysis of the grey-level histogram, selecting thresholds which lie in the region between the two modes of the (bi-modal) histogram. The assumption that the histogram is indeed bi-modal, with one mode corresponding to the grey-level representing the object and the other to the grey-level representing the background, is often not valid; histograms are frequently noisy and the two modes may be

difficult to detect (see Figure 5.1). Just as often, the object will generate a single mode while the background will comprise a wide range of grey-level, giving rise to a uni-modal histogram (see Figure 5.2). While several applications of a simple smoothing (or local averaging) operator to a noisy histogram will help with noisy bi-modal histograms, it will be of little use with uni-modal histograms. Figure 5.3

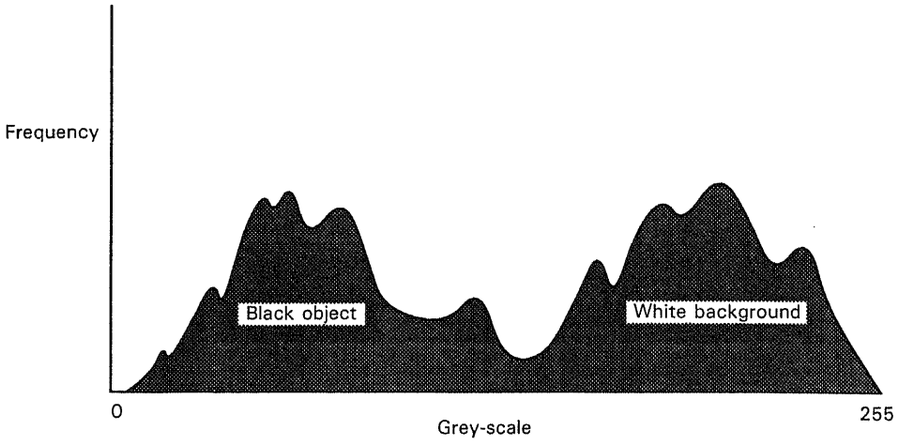

Figure 5.1 Noisy bi-modal grey-scale histogram.

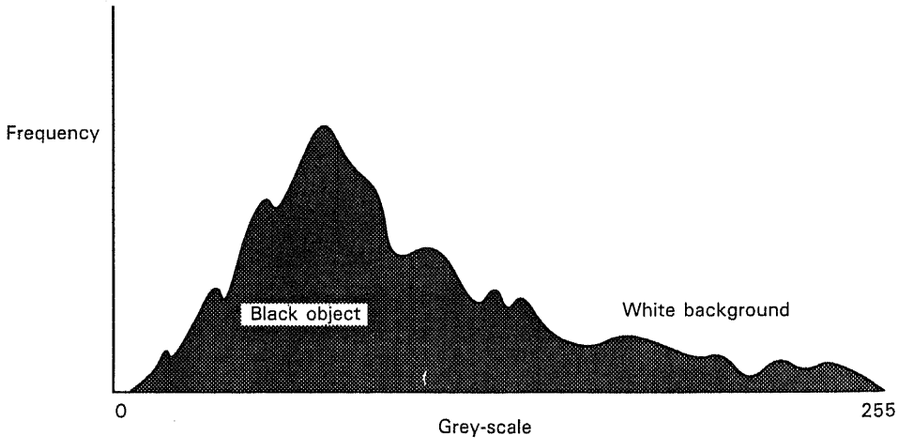

Figure 5.2 Uni-modal grey-scale histogram.

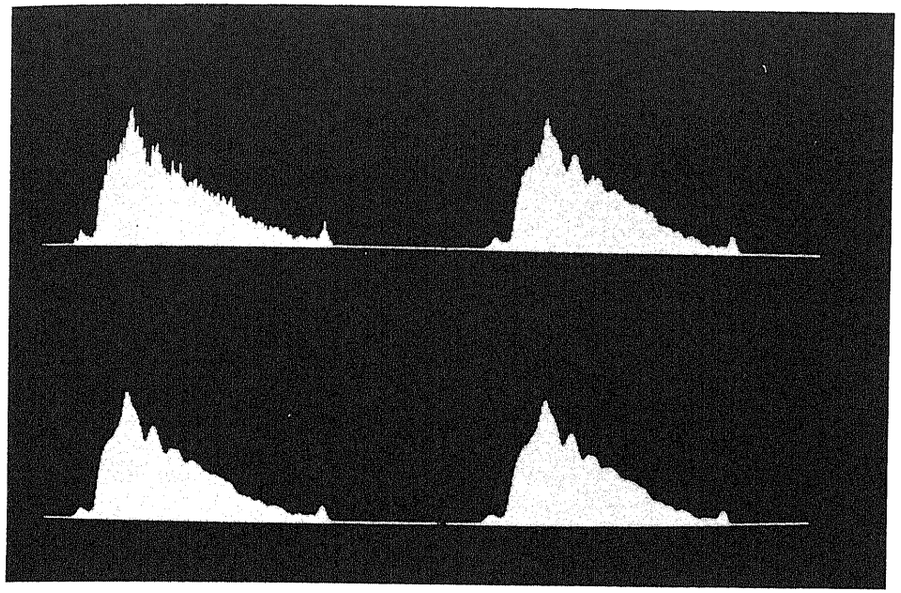

Figure 5.3 Grey-scale histogram smoothing: top-left: no smoothing; top-right: one application of a 3×1 neighbourhood average operator; bottom-left: two applications; bottom-right: three applications.

illustrates the effect of smoothing a grey-scale histogram by iterative application of a local 3×1 neighbourhood averaging operator.

Another useful approach to thresholding selection is to use the average grey-level of those pixels which are on the boundary between the object and the background as an estimate of the threshold value. As the grey-level of this boundary pixel will typically lie between those of the object and the background, it provides a good indication of the threshold value (see Figure 5.4). The difficulty, of course, lies in deciding which pixels *are* on the boundary.

One of the best approaches is to use a reliable edge detector, such as the Marr–Hildreth operator described in the next section, to identify these boundary points. The threshold selection procedure first uses a Marr–Hildreth operator to locate edges in the image and the mean grey-level of the image pixels at these edge locations is computed. This mean represents the global threshold value. To illustrate this approach, Figure 5.5 shows the binary image generated by thresholding the original grey-scale image at a threshold equal to the mean grey-level of the boundary points generated using the Marr–Hildreth operator. It should be noted that, although this threshold selection technique is computationally complex and may take a significant amount of time to compute, it is only a calibration exercise and need not be performed before every threshold operation.

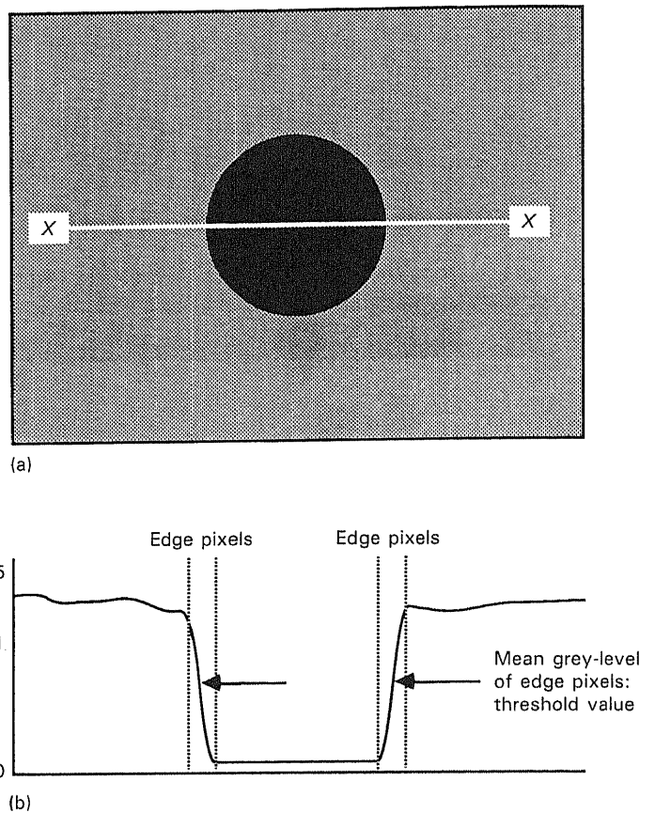

Figure 5.4 Using edge pixels to select a threshold: (a) image of dark, round object on a light background with section $X-X$ shown; (b) profile of image intensity along section $X-X$.

5.3 An overview of edge detection techniques

As might be expected when dealing with a process which is fundamental to image segmentation, the literature concerning edge detection is large. This section will not attempt to review all detectors in detail; rather it will survey and describe the different approaches to edge detection and illustrate the approach with specific algorithms.

There are four distinct approaches to the problem of edge detection:

(a) gradient- and difference-based operators;
(b) template matching;

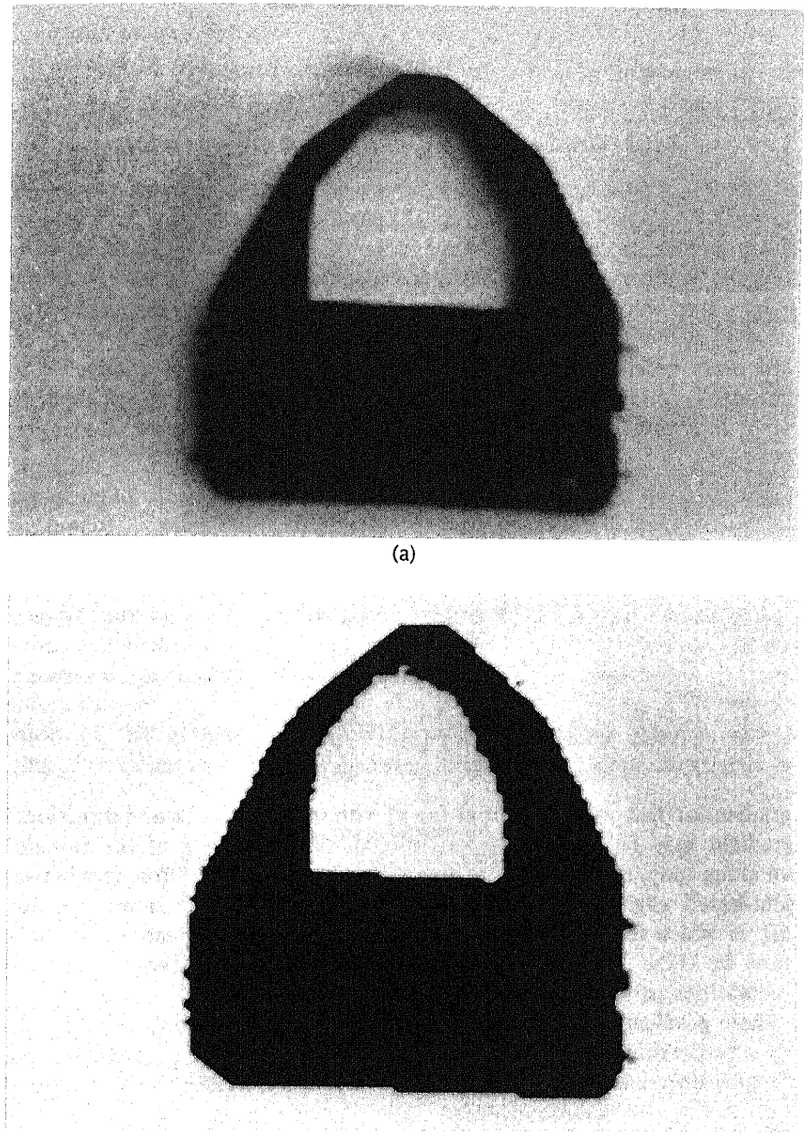

Figure 5.5 Automatic threshold selection using the Marr–Hildreth theory of edge detection: (a) original grey-scale image; (b) automatically thresholded binary image.

(c) edge fitting;
(d) statistical edge detection.

Each of these four approaches will be considered in turn.

5.3.1 Gradient- and difference-based operators

If we define a local edge in an image to be a transition between two regions of significantly different intensities, then the gradient function of the image, which measures the rate of change, will have large values in these transitional boundary areas. Thus gradient-based, or first-derivative-based, edge detectors enhance the image by estimating its gradient function and then signal that an edge is present if the gradient value is greater than some defined threshold.

In more detail, if $\partial/\partial x$ and $\partial/\partial y$ represent the rates of change of a two-dimensional function $f(x, y)$ in the x- and y-directions, respectively, then the rate of change in a direction θ (measured in the positive sense from the X-axis) is given by:

$$\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$

The direction θ, at which this rate of change has the greatest magnitude is given by:

$$\arctan\left[\frac{\partial f}{\partial y}\Big/\frac{\partial f}{\partial x}\right]$$

with magnitude:

$$\sqrt{\left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]}$$

The gradient of $f(x, y)$ is a vector at (x, y) with this magnitude and direction. Thus the gradient may be estimated if the directional derivatives of the function are known along (any) two orthogonal directions. The essential differences between all gradient-based edge detectors are the directions which the operators use, the manner in which they approximate the one-dimensional derivatives of the image function in these directions, and the manner in which they combine these approximations to form the gradient magnitude.

These gradient functions are intuitively easy to understand when we confine ourselves to the discrete domain of digital images where partial derivatives become simple first differences. For example, the first difference of a two-dimensional function in the x-direction is simply:

$$f(x+1, y) - f(x, y)$$

Similarly, the first difference of a two-dimensional function in the y-direction is simply:

$$f(x, y+1) - f(x, y)$$

An operator due to Roberts estimates the derivatives diagonally over a 2×2 neighbourhood. The magnitude of the gradient $g(x, y)$, at an image point (x, y), is approximated by taking the RMS of the directional differences:

$$g(x,y) \approx R(x,y) = \sqrt{[\{f(x,y) - f(x+1,y+1)\}^2 + \{f(x,y+1) - f(x+1,y)\}^2]}$$

$R(x, y)$ is usually referred to as the Roberts cross operator. The differences may be combined in a way other than the RMS to provide a computationally simpler version, the Roberts absolute value estimate of the gradient function, given by:

$$g(x,y) \approx R(x,y) = |f(x,y) - f(x+1,y+1)| + |f(x,y+1) - f(x+1,y)|$$

Rosenfeld and Kak have argued that a third version, the Roberts max operator, given by:

$$g(x,y) \approx R(x,y) = \text{Max}\ (|f(x,y) - f(x+1,y+1)|, |f(x,y+1) - f(x+1,y)|)$$

affords better invariance to edge orientation. Applying the Roberts max operator to edges of equal strength, but of different orientation, produces less variation in the resultant magnitude value than if the cross operator were used.

To illustrate these edge detectors, a test image with fairly fine structure (a tray of electrical wires) was acquired: see Figure 5.6. The Roberts RMS, absolute value, and Roberts max operators are shown in Figures 5.7, 5.8, and 5.9 respectively.

One of the main problems with the Roberts operator is its susceptibility to noise because of the manner in which it estimates the directional derivatives, i.e. the first differences, of the image function $f(x, y)$. This has prompted an alternative estimation of the gradient by combining the differencing process with local averaging. For example, the Sobel operator estimates the partial derivative in the

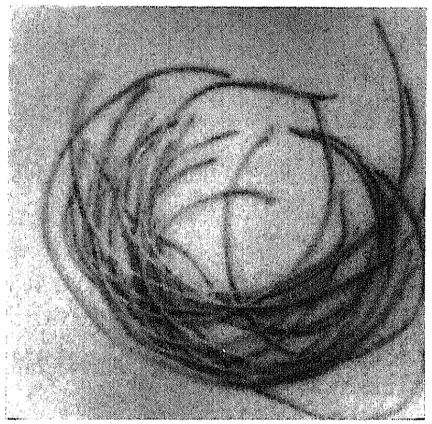

Figure 5.6 A tray of wires.

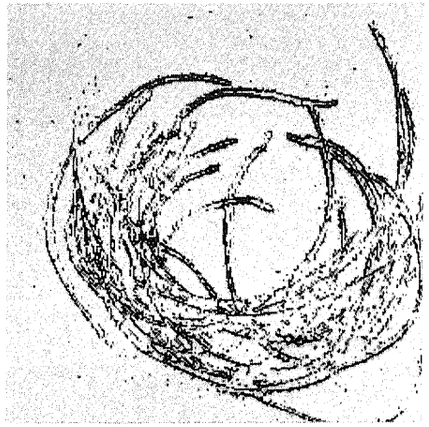

Figure 5.7 Roberts RMS edge detection operator.

Figure 5.8 Roberts absolute value edge detection operator.

x-direction over a 3×3 region centred at $f(x, y)$ by:

$$S_x = \{f(x+1, y-1) + 2f(x+1, y) + f(x+1, y+1)\} - \{f(x-1, y-1) + 2f(x-1, y) + f(x-1, y+1)\}$$

This essentially takes the difference of a weighted average of the image intensity on either side of $f(x, y)$. Similarly:

$$S_y = \{f(x-1, y+1) + 2f(x, y+1) + f(x+1, y+1)\} - \{f(x-1, y-1) + 2f(x, y-1) + f(x+1, y-1)\}$$

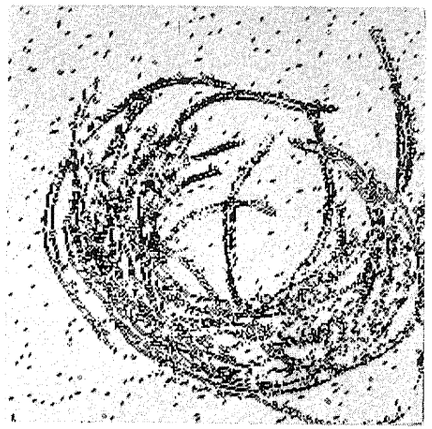

Figure 5.9 Roberts max edge detection operator.

Figure 5.10 Sobel RMS edge detection operator.

The gradient may then be estimated as before by either calculating the RMS (see Figure 5.10):

$$g(x, y) \approx S = \sqrt{(S_x^2 + S_y^2)}$$

or by taking the absolute values (see Figure 5.11):

$$g(x, y) \approx S = |S_x| + |S_y|$$

In an analogous manner Prewitt suggests an approximation of the partial

derivatives by:

$$P_x = \{f(x+1, y-1) + f(x+1, y) + f(x+1, y+1)\} - \{f(x-1, y-1) + f(x-1, y) + f(x-1, y+1)\}$$

$$P_y = \{f(x-1, y+1) + f(x, y+1) + f(x+1, y+1)\} - \{f(x-1, y-1) + f(x, y-1) + f(x+1, y-1)\}$$

and the gradient may be estimated as before (see Figures 5.12 and 5.13).

Quite often, the directional differences are estimated using simple convolution

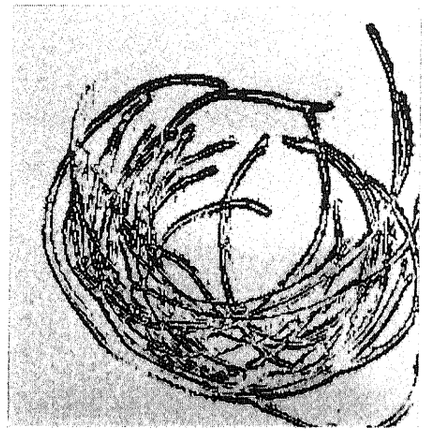

Figure 5.11 Sobel absolute value edge detection operator.

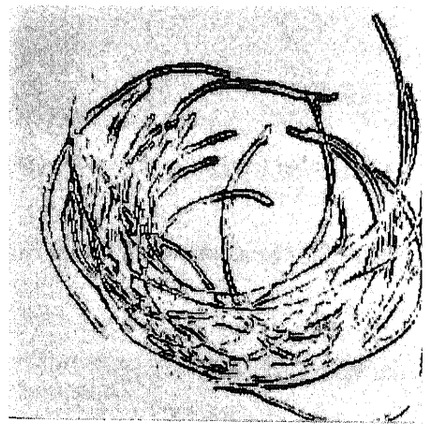

Figure 5.12 Prewitt RMS edge detection operator.

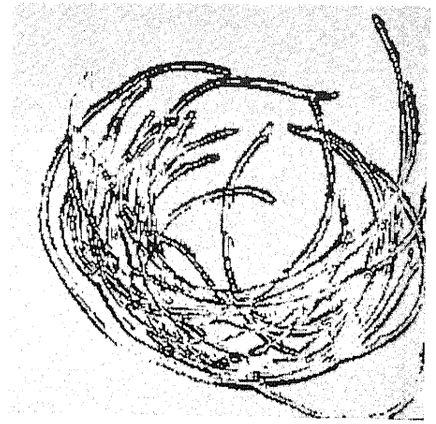

Figure 5.13 Prewitt absolute value edge detection operator.

kernels, one kernel for each different operator. These yield two partial derivative images, which are then combined on a point by point basis, either as the RMS or as the sum of absolute values, to produce the final gradient estimate. Figure 5.14 illustrates the convolution kernels for the Roberts, Sobel, and Prewitt operators. Strictly speaking, the kernel should first be rotated by 180° before the convolution is performed (see Section 4.2.1). However, this is normally omitted since the resultant error of 180° in the gradient direction can be ignored.

Once the gradient magnitude has been estimated, a decision as to whether or not an edge exists is made by comparing it to some predefined value; an edge is deemed to be present if the magnitude is greater than this threshold. Obviously the choice of threshold is important and in noisy images threshold selection involves a trade-off between missing valid edges and including noise-induced false edges.

So far, edge detection has been discussed on the basis of first-derivative directional operators. However, an alternative method uses an approximation to the Laplacian:

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

i.e. the sum of second-order, unmixed, partial derivatives. The standard approximation is given by:

$$L(x, y) = f(x, y) - 1/4\{f(x, y+1) + f(x, y-1) + f(x+1, y) + f(x-1, y)\}$$

The digital Laplacian has zero response to linear ramps (and thus gradual changes in intensity) but it does respond on either side of the edge, once with a positive sign and once with a negative sign. Thus in order to detect edges, the image is enhanced by evaluating the digital Laplacian and isolating the points at which the resultant

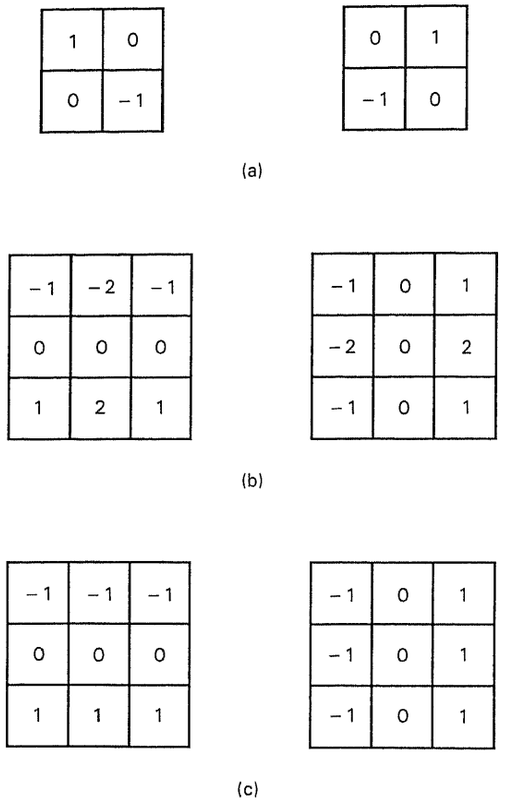

Figure 5.14 Convolution kernels for estimation of the partial derivatives with (a) Roberts; (b) Sobel; and (c) Prewitt edge detection operators.

image goes from positive to negative, i.e. at which it crosses zero. The Laplacian has one significant disadvantage: it responds very strongly to noise.

A different and much more successful application of the Laplacian to edge detection was proposed by Marr and Hildreth in 1980. This approach first smooths the image by convolving it with a two-dimensional Gaussian function, and subsequently isolating the zero-crossings of the Laplacian of this image:

$$\nabla^2\{I(x, y) * G(x, y)\}$$

where $I(x, y)$ represents the image intensity at a point (x, y) and $G(x, y)$ is the two-dimensional Gaussian function, of a given standard deviation σ, defined by:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp[-(x^2 + y^2)/2\sigma^2]$$

Despite some criticism of this technique, it is very widely used. The operator possesses a number of useful properties: for example: the evaluation of the Laplacian and the convolution commute so that, for a Gaussian with a given standard deviation, we can derive a single filter: *the Laplacian of Gaussian*:

$$\nabla^2\{I(x, y) * G(x, y)\} = \nabla^2 G(x, y) * I(x, y)$$

Furthermore, this two-dimensional convolution is separable into four one-dimensional convolutions (see Appendix I for a derivation):

$$\nabla^2\{I(x, y) * G(x, y)\} = G(x) * \left\{I(x, y) * \frac{\partial^2}{\partial y^2} G(y)\right\} + G(y) * \left\{I(x, y) * \frac{\partial^2}{\partial x^2} G(x)\right\}$$

Bearing in mind that an implementation of the operator requires an extensive support, e.g. 63×63 pixels for a Gaussian with standard deviation of 9.0, this separability facilitates significant computational savings, reducing the required number of multiplications from n^2 to $4n$ for a filter kernel size of n pixels. The Laplacian of Gaussian operator also yields thin continuous closed contours of zero-crossing points. This property is most useful in subsequent processing, such as when characterizing the intensity discontinuities or edges as object boundaries.

The location of the zero-crossings in space is not the only information that can be extracted from the convolved image: the amplitude (which is related to the slope) and orientation of the gradient of the convolved image at the zero-crossing point provide important information about edge contrast and orientation. The slope of a zero-crossing is the rate at which the convolution output changes as it crosses zero and is related to the contrast and width of the intensity change.

Since the Gaussian is used to smooth the image and since different standard deviations yield edges detected at different scales within the image (successively smoothing out image detail), Marr's theory also requires the correlation of edge segments derived using Gaussians of different standard deviation. However, the edges detected by one operator alone are often sufficiently reliable for many industrial applications: see Figure 5.15. In any event, we will return to this issue of multi-scale or multi-resolution edge detection when we discuss image understanding in Chapter 9.

5.3.2 Template matching

Since an ideal edge is essentially a step-like pattern, one straightforward approach to edge detection is to try to match templates of these ideal step edges with regions of the same size at every point in the image. Several edge templates are used, each template representing an ideal step at a different orientation. The degree of match can, for example, be determined by evaluating the cross-correlation between the

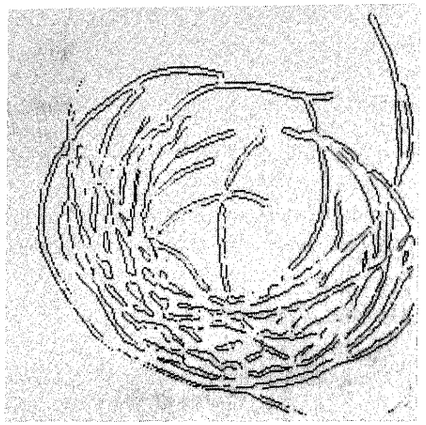

Figure 5.15 Marr–Hildreth edge detection operator.

template and the image.* The template producing the highest correlation determines the edge magnitude at that point and the edge orientation is assumed to be that of the corresponding template. Detection is accomplished by thresholding in the same manner as discussed for gradient approaches. These templates are often referred to as edge masks.

Such a set of masks, due to Kirsch, is:

```
 1  1  1    1  1  1   -1  1  1   -1 -1  1
 1 -2  1   -1 -2  1   -1 -2  1   -1 -2  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 -2  1    1 -2 -1    1 -2 -1    1 -2 -1
 1  1  1    1  1  1    1  1 -1    1 -1 -1
```

Figure 5.16 illustrates the effect of the application of this set of masks.

Another set, due to Prewitt, is:

```
 5  5  5   -3  5  5   -3 -3  5   -3 -3 -3
-3  0 -3   -3  0  5   -3  0  5   -3  0  5
-3 -3 -3   -3 -3 -3   -3 -3  5   -3  5  5

-3 -3 -3   -3 -3 -3    5 -3 -3    5  5 -3
-3  0 -3    5  0 -3    5  0 -3    5  0 -3
 5  5  5    5  5 -3    5 -3 -3   -3 -3 -3
```

This edge detection technique is shown in Figure 5.17.

* Cross-correlation is discussed in Chapter 6 on image analysis.

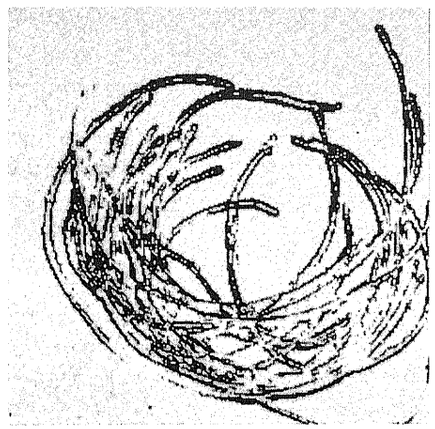

Figure 5.16 Kirsch template edge detection operator.

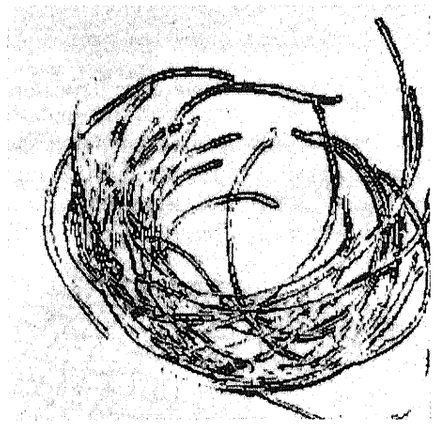

Figure 5.17 Prewitt template edge detection operator.

All the masks discussed so far, with the exception of the Laplacian of Gaussian, when convolved with the image, will produce an enhanced image with large values, not only at the centre of the edge, but also at points close to that edge. Subsequent thresholding of such an enhanced image will generate an edge map with thick edges. Nevatia and Babu suggested a template-matching algorithm (see Figure 5.18) which produces a thin edge. In this case, six 5×5 masks (corresponding to edges at 0°, 30°, 60°, 90°, 120°, and 150° orientations) are correlated with the

image. An edge is deemed present at a particular orientation if:

(a) the response at that orientation exceeds a set threshold; and
(b) it is not dominated by responses at neighbouring points in a direction that is normal to the candidate edge.

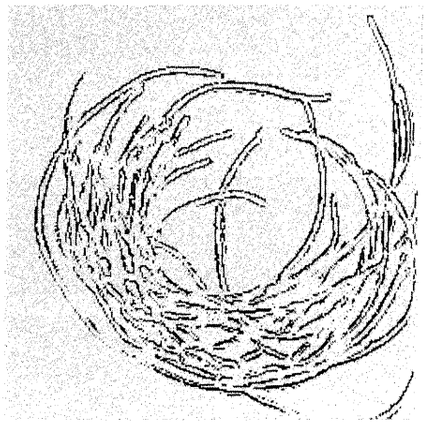

Figure 5.18 Nevatia–Babu template edge detection operator.

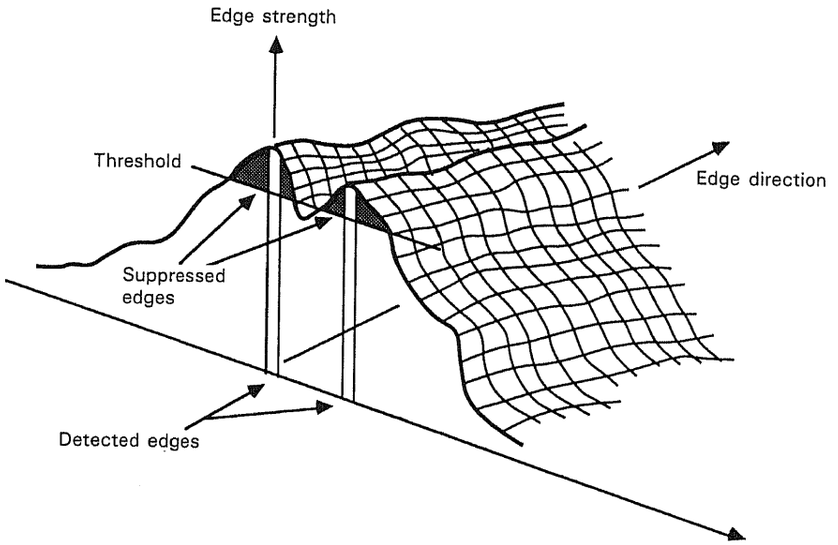

Figure 5.19 Non-maxima suppression.

In particular, the edge magnitude must be higher than the edge magnitude of the pixels on either side of it, in a direction normal to the edge orientation (see Figure 5.19), and the neighbouring pixels are also required to have edge orientations similar to (within thirty degrees) that of the central point. This general technique is referred to as 'non-maxima suppression' and was originally proposed by Rosenfeld and Thurston in 1971.

5.3.3 Edge fitting

As mentioned above, an ideal edge can be modelled as a step discontinuity in intensity at a particular location in the image. A step function can be defined in a circular region by a function $S(x, y)$ defined as follows (refer to Figure 5.20):

$$S(x, y) = \begin{cases} b & (x \cos\theta + y \sin\theta) < \rho \\ b + h & (x \cos\theta + y \sin\theta) \geqslant \rho \end{cases}$$

where

h is the step height (intensity difference);
b is the base intensity;
ρ and θ define the position and orientation of the edge line with respect to the origin of this circular region.

The approach to edge detection in this case is to determine how closely this model fits a given image neighbourhood and to identify the values of b, h, ρ, and θ that minimize some measure of the distance between this circular step function and a corresponding area in the image. This is the basis of Hueckel's operator. The error, $\mathscr{E}$ between the ideal step $S(x, y, b, h, \rho, \theta)$, defined over a circular region C

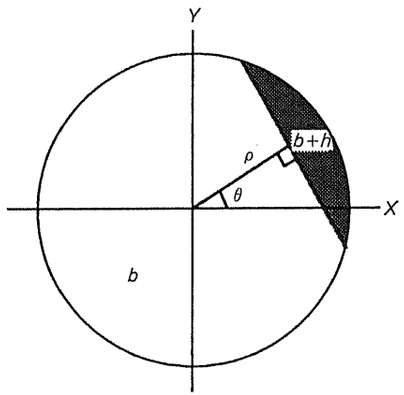

Figure 5.20 Parameters defining a step function in a circular region.

and (some) circular sub-image $f(x, y)$ of the same size can be given by:

$$\mathscr{E}^2 = \sum_{x,y \in C} (f(x, y) - S(x, y, b, h, \rho, \theta))$$

Note that $f(x, y)$ is the image intensity at a point (x, y) and $S(x, y, b, h, \rho, \theta)$ is the ideal step edge for chosen values of b, h, ρ, and θ at a point (x, y). The summation is to be performed over the entire region C and b, h, ρ, and θ are to be chosen so that $\mathscr{E}^2$ is minimized. This minimization process is simplified by expanding both $f(x, y)$ and $S(x, y, b, h, \rho, \theta)$ in terms of a set of orthogonal (Fourier) basis functions and using the corresponding coefficients (f_i and s_i, say) in the error measure given above instead of $f(x, y)$ and $S(x, y, b, h, \rho, \theta)$. In practice, only the first eight terms of the expansion are used. The operator was designed to detect the presence of two edge elements in the circular neighbourhood simultaneously, reporting only the more dominant edge. Each edge requires four parameters to describe it and it was for this reason that the eighth term was chosen as the cut-off.

Thus, the problem is now to minimize:

$$E^2 = \sum_{i=0}^{7} (f_i - s_i)$$

Hueckel then presents and proves a theorem which, in effect, reduces this problem to one of extremization of a function in θ and he derives subsequent expressions for ρ, b, and h. The decision as to whether an edge is present or not is based on the amplitude of the computed step as well as the degree of fit. Bearing in mind the computational complexity of the Hueckel edge fitting technique, the results are, in general, quite disappointing (see Figure 5.21).

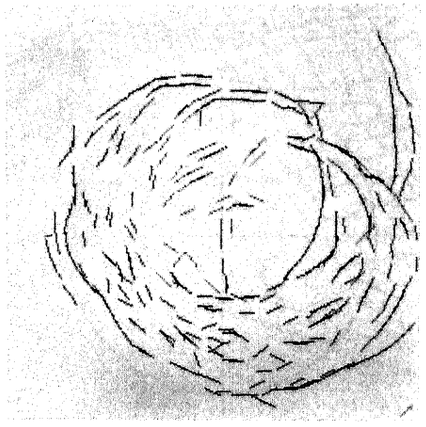

Figure 5.21 Hueckel edge detection operator.

5.3.4 Statistical techniques

If you consider a small region or window in an image (e.g. a 7×7 pixel area), you can view an edge as a boundary between two inhomogeneous sub-regions. On the other hand, if the entire region is homogeneous, then no edge exists. This is the basis of an edge detection technique due to Yakimovsky (see Figure 5.22) who treated edge detection as an exercise in hypothesis testing, choosing between the two following hypotheses:

H0: The image values on two sides of a line through the window are taken from the same region.

H1: The image values on one side are from one region and on the other side from a different region.

Assuming that each region comprises pixels having normally distributed grey-levels and each pixel in the region is mutually independent, Yakimovsky argued that the choice can be made by considering the ratio of (a) the grey-level standard deviation of the combined region (raised to the power of the number of pixels in the region) to (b) the product of the grey-level standard deviations of both individual regions, each with standard deviation raised to the power of the number of pixels in the respective region.

To decide whether an edge of a given orientation exists in a window, one can bisect the window at that orientation, thus creating two sub-regions, calculate the appropriate standard deviations and decide as to the presence of an edge based on some selected threshold. This may be done for several orientations, for example $0°$, $30°$, $60°$, $90°$, $120°$, and $150°$. There are several other statistical edge detectors but this one serves to illustrate the approach.

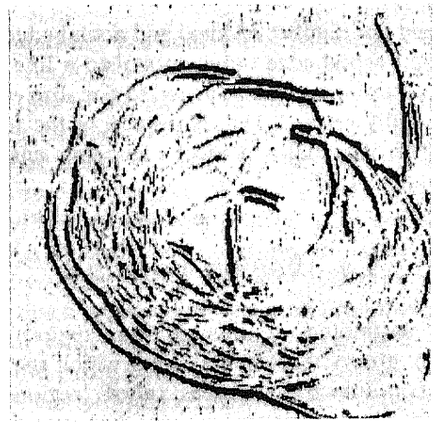

Figure 5.22 Yakimovsky edge detection operator.

5.3.5 Assessment of edge detection

One point on which there is a definite consensus in the computer vision community is that it is difficult quantitatively to evaluate and compare edge detector performance. Reasons for this include the fact that there are a large number of detectors and each detector incorporates its own inherent edge model. Thus, certain edge detectors may be more appropriate or more successful in certain circumstances. Also, edge detectors will display differing degrees of noise immunity: some may be too sensitive to noise; others will cater for (certain types of) noise but possibly at the cost of missing valid edges. A generic mathematical analysis of detectors is difficult, even for simple images, due to the frequent complexity and non-linearity of the detectors (but see Hildreth, 1985, and Torre and Poggio, 1986, for discussions of the topic). It is also worth noting that many of the detectors which can claim some degree of optimality (e.g. Canny, Marr–Hildreth, Fleck) are optimal only with respect to the criteria they choose as relevant. To coin a phrase: 'You pick your optimality criteria and you take your choice.'

In spite of these difficulties, a quantitative evaluation is clearly desirable even if only to provide a very rough guide to the relative performance of detectors in limited circumstances. If a clear-cut objective analytic measure is difficult to establish (but, again, see Torre and Poggio, 1986), an empirical assessment may often suffice. It cannot be stressed too much, however, that empirical tests are only relevant in the application domain in which they are carried out. One figure of measure that can be used in such tests was suggested by Pratt (Pratt, 1978; Abdou and Pratt, 1979) and provides a way of assessing the performance of edge detectors in localizing the position of edges. Specifically, a figure of merit R, yielding a value in the interval 0.0–1.0, is defined:

$$R = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha d^2}$$

where I_I and I_A represent the number of ideal and actual edge map points, d is the separation distance of an actual edge point normal to a line of ideal edge points. The value α is a scaling constant and is adjusted to penalize edges offset from their true location. It is typically set at 0.111. Normalizing the figure of merit by the maximum of the number of actual and ideal edgepoints ensures that fragmented and smeared edges are penalized.

5.4 Region growing

As we noted at the beginning of this section, region growing effects the segmentation process by grouping elemental areas which share a common feature into large connected two-dimensional areas called 'regions', with the implicit assumption that these resultant regions correspond to some real-world surface or object. Thus, the central idea underlying region-growing techniques is to merge

initially small regions (e.g. individual pixels) into large ones. This merging process must then address two questions: upon what criterion will two regions be merged and at what point will merging cease? To answer the first question, we must define more rigorously what we understand by the term *region*. A region is an aggregate (collection) of pixels all of which satisfy some *uniformity predicate* $U(p)$, such that the value of $U(p)$ is true if some local property of the neighbourhood of pixel p satisfies the uniformity predicate and false otherwise. Most region-growing techniques base the test for uniformity on some feature of the grey-level. The answer to the question: 'what is the criteria for merging two regions?' is thus, quite simply, the uniformity predicate. Two *adjacent* regions are merged if the merged region satisfies the uniformity predicate. The answer to the second question: 'at what point does merging cease?' is now clear. Merging (or growing) ceases when no two adjacent regions satisfy the uniformity predicate. Thus, the region-growing procedure is an iterative one: small regions are merged to form larger ones; these are then merged (if appropriate) to form still larger ones, and so on until no more regions can be merged. A natural consequence of this approach is that sophisticated data structures are required to keep track of the intermediate regions and their features.

In the trivial global thresholding technique, the uniformity predicate is simply that the grey-level of the pixel (region) in question should lie within a pre-set range of grey-levels; the final regions are grown in one step as only one test is needed for each pixel and there are no intermediate regions.

5.4.1 The split and merge procedure using quad-trees

This region-growing algorithm, developed by Horowitz and Pavlidis in 1976, makes use of an image representation called the quad-tree. A quad-tree is a tree (a collection of nodes organized in a hierarchical manner) in which each node has either four sons or no sons.

The root of the quad-tree is a node which represents the average grey-level of the entire image. The leaves of a full quad-tree, i.e. the nodes which do not have any offspring, correspond to the individual pixels of the image. The parent of a group of four nodes corresponds to the average grey-level of a 2×2 neighbourhood of the image: if each of these four pixels/nodes have the same grey-level, then, obviously, there is no need to represent them explicitly as the parent node can do the job just as well and they can be deleted. Similarly, the parent of each of these nodes corresponds to a further 2×2 grouping of these 2×2 neighbourhoods; again, the off-spring are present in the quad-tree if and only if they have different grey-levels. The hierarchical aggregation continues until one reaches the root of the tree which clearly represents the average grey-level of its sons, and hence the average grey-level of the entire image. Thus, one can see that the quad-tree is a useful method of image compression if there are large regions of uniform grey-level since much of these regions can be represented by nodes close to the root, without the need to have nodes for the individual pixels; see Figure 5.23.

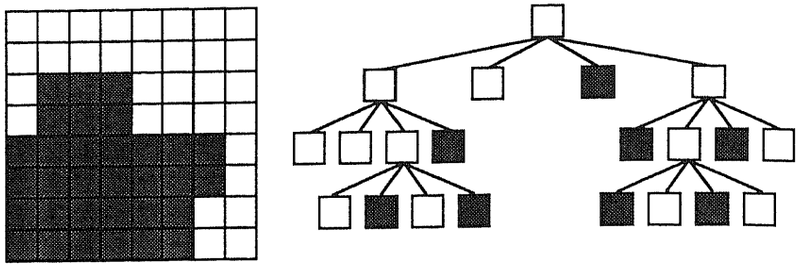

Figure 5.23 A quad-tree representation of an 8 × 8 binary image.

The split and merge procedure begins with a quad-tree of a given depth, typically where the depth is such that the leaves correspond not to pixels but to larger blocks. Leaves are then merged: if four leaves from a parent satisfy the uniformity predicate then the leaves are deleted and the parent inherits their average grey-level (assuming that we are going to use grey-level as a way of labelling a region). This merging continues at the next highest level, and then at subsequent levels, until no more merging is possible. At this stage a splitting procedure is initiated. Here, the quad-tree is traversed and at each leaf the uniformity predicate of the corresponding group is evaluated; if the value is false then the block is split and the four corresponding nodes are added to the node. This process continues until all leaves satisfy the uniformity predicate. At this stage, the quad-tree is traversed horizontally. The neighbours of each leaf are examined and adjacent leaves from different parents which satisfy the uniformity predicate are linked together to form a region.

It is worth noting that, for industrial applications, region-growing techniques are considered too computationally complex (and hence too slow) to be of much use and should only be considered when thresholding or edge detection techniques are incapable of yielding satisfactory results.

5.5 Boundary detection

As we discussed previously, edge detection is only the first stage of the boundary-based segmentation process. We also need to aggregate these local edge elements, which are a relatively featureless representation, into structures better suited to the process of interpretation. This is normally achieved using processes such as edge thinning (recall that gradient-based template matching edge detectors produce thick edges), edge linking, gap-filling, and curve-segment linking in order to generate a distinct, explicit, and unambiguous representation of the boundary. There are several techniques for boundary detection and they vary in the amount of knowledge or domain-dependent information that is used in the grouping process. These approaches include, in order of decreasing use of domain-dependent

information, boundary refining, the Hough transform, graph searching, dynamic programming, and contour following. Since the Hough transform will be described in detail in the next chapter, we will confine ourselves here to the other four techniques, placing emphasis on contour following. However, the reader can, if he/she likes, skip ahead and preview the Hough transform section without any difficulty.

5.5.1 Boundary refining

Boundary refining methods use an initial *a priori* estimate of the position of the boundary to guide a search for the actual, or real, boundary and subsequently to refine this initial estimate. The estimate may have been generated by the analysis of lower-resolution images or, alternatively, explicit high-level knowledge can be utilized, for example knowledge that the boundary lies on, or close to, a given curve. For example, Bolles incorporated this technique in an algorithm which builds boundaries by searching in a direction at right-angles to the *a priori* boundary for local edges and selects the element with the highest gradient value, provided its direction is (almost) parallel to the boundary direction (Bolles, 1977). This is repeated at regular intervals along the approximate *a priori* boundary and, if sufficient edge elements are extracted, then these locations are fitted with some analytic function, e.g. a low-degree polynomial. This parameterized curve then represents the actual boundary.

Another approach, referred to as *divide-and-conquer* or *iterative end-point fit* boundary detection, is often applicable where a low curvature boundary is known to exist between two edge elements. The general idea here is to fit an initial line between the two points; if the normal distance from the line to the point of maximum gradient magnitude is less than some pre-set tolerance, the approximation is complete, otherwise the point of greatest normal distance from the line becomes a break-point on the boundary, forming two new line segments. All new segments are then subjected, in an iterative manner, to this same process.

5.5.2 Graph-theoretic techniques

A second approach to boundary detection, which need not depend significantly on the existence of *a priori* knowledge, treats the aggregation or association of edge points into boundaries as a (low-cost) graph traversal. In particular, edge points are viewed as nodes in a graph and there is a cost function or weight associated with connecting two neighbouring points or nodes; the desired boundary may be interpreted as the minimal cost (or any low-cost) path through the graph. The cost function or weight associated with connecting two edge points is normally defined to be a function of the distance between them, the difference in their directions, and edge strength.

5.5.3 Dynamic programming

This approach formulates the boundary-following procedure as a dynamic programming problem by defining a cost function which embodies a notion of the 'best boundary'. This is not dissimilar to the idea of the graph-theoretic technique in that the path specified by the boundary minimizes the cost function. For example, suppose that a local edge detection operator is applied to a grey-level image to produce edge magnitude and direction information at points $x_1, \ldots, x_n$. One possible criterion for a good boundary is a weighted sum of high cumulative edge strength and low cumulative curvature, that is, for a curve with n segments:

$$H(x_1, \ldots, x_n) = \sum_{k=1}^{n} s(x_k) + c \sum_{k=1}^{n-1} q(x_k, x_{k+1})$$

with the implicit constraint that consecutive points must be grid neighbours:

$$\| x_{k+1} - x_k \| < 2$$

The function $q(x_k, x_{k+1})$ embodies the difference in direction between edge element x_k and element x_{k+1}. Note that c is a negative constant. The function $s(x_k)$ is the edge strength (or gradient magnitude). The evaluation function $H(x_1, \ldots, x_n)$ is in the form of a serial optimization problem where not all variables in the evaluation function are simultaneously inter-related, and can be solved by a multi-stage optimization process referred to as *serial dynamic programming* (Bellman and Dreyfus, 1962).

5.5.4 Contour following

Contour following is a simple approach which uses no domain-dependent information and 'follows' the boundary or contour exclusively on the basis of locally derived data. The basis of the technique is, essentially, to start with a point that is believed to be on the boundary (some local edge point, say) and to extend the boundary by adding a neighbouring point in the contour direction (i.e. the direction which is normal to the gradient direction). This process of extension is reiterated, starting at this new boundary pixel. In this way a contour is followed around the boundary (see Figure 5.24). The basis for selecting a candidate varies from task to task and from algorithm to algorithm but normally is dependent at least on the gradient direction and on the gradient magnitude. Since contour following techniques are often based on gradient edge detectors, these techniques are normally most successful with images in which there is little noise. For the purpose of illustration, the following section details a simple contour following algorithm.

To begin with, contour following algorithms cannot assume that the boundary constitutes a closed curve. Thus, the boundary is followed in two directions: first in the forward boundary direction and then in the reverse direction. The forward boundary direction is arbitrarily designated the direction equal to the

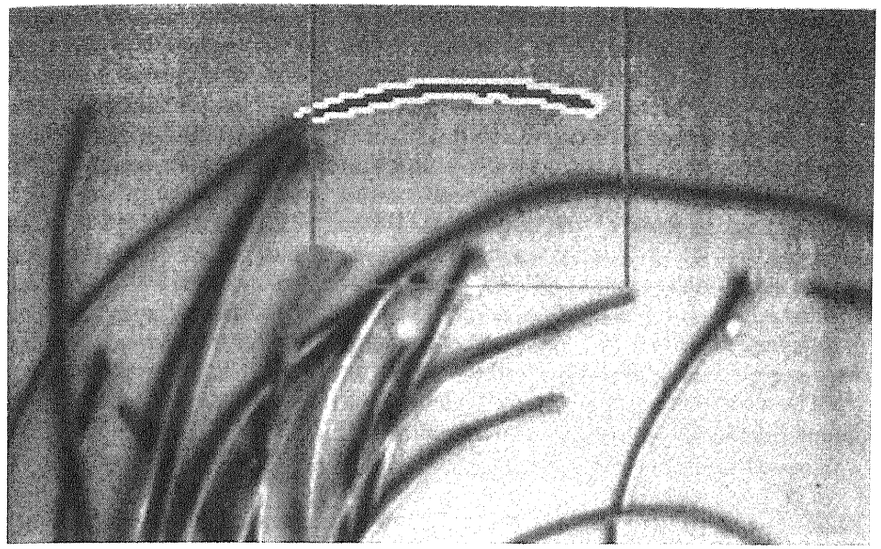

Figure 5.24 Contour following.

edge normal (or gradient) direction $+90^\circ$ and the reverse direction corresponds to the edge normal direction -90°. The boundary following algorithm proceeds on a pixel to pixel basis, tracing the local maximum gradient given by the gradient direction at that point on the boundary to the next pixel.

Tracing continues as long as the difference between the current and candidate pixel gradient directions is not too large; this helps to avoid following boundaries into noisy areas which are characterized by frequent changes in edge direction. If no acceptable edge is encountered when tracing, a search is made in two zones ahead of the boundary: the first zone separated by a gap of one pixel from the current boundary point, the second by two pixels. If a suitable edge is found, the intervening gap pixels are filled in and the trace is restarted from this point. If none is found then the boundary is traced in the reverse direction from the original start point, provided this has not already been done.

The boundary following algorithm terminates when the boundary can be followed no further: that is, if neither the tracing nor the searching algorithms yield a valid boundary point. Boundary following also terminates if the original start point is re-encountered during tracing, signifying that a boundary is represented by a closed curve. Thus, the extraction of both open and closed contours is facilitated.

As the algorithm traces around the boundary, it builds a boundary chain code (BCC) representation of the contour. The BCC is defined and discussed in detail in Chapter 7. For the purposes of this section it is sufficient to summarize its essential structure and properties. A boundary chain code comprises an integer pair, denoting the coordinates of an origin point on the contour, and a sequence

(or chain) of integer values representing the direction of the next pixel on the contour. The range of possible directions is coarsely quantized and there are just eight possible directions corresponding to each of the eight pixels which are adjacent to the centre pixel in a 3×3 neighbourhood. The boundary following algorithm adheres to the Freeman chain code convention: the pixel to the right of centre (east neighbour) is given by the direction 0, the north-east pixel is given by the direction 1. Continuing anti-clockwise around the neighbourhood, the south-east neighbour is given by the direction 7. Figure 5.25 illustrates these direction codes.

To avoid following multiple close parallel boundaries caused by the presence of 'thick' edge responses, pixels in directions normal to the boundaries extracted by the following algorithm are suppressed, i.e they are labelled, so as to exclude them from later consideration by either the trace or search modules.

The next two sections discuss the component trace and search modules of the contour following algorithm in more detail.

□ *The trace algorithm*

The purpose of the trace algorithm is to follow a sequence of boundary pixels and to generate a model of the boundary, i.e. the BCC, by recording all the boundary pixels visited. It is assumed that the boundary pixels correspond to the pixels exhibiting the local maximum gradient magnitude and that the gradient direction indicates, approximately at least, the neighbouring boundary pixel. The boundary direction, D, of an (edge) point is given by the edge gradient direction G as $D = G + 2$ for the forward direction and as $D = G - 2$ for the reverse direction ($+90^\circ$ and -90° respectively). All directional additions and subtractions are modulo 8 operations. The tracing algorithm chooses as candidates for inclusion in the boundary those pixels given by directions $D, D + 1$, and $D - 1$, that is, the pixel directly ahead of the current pixel and one pixel on either side of it. For example,

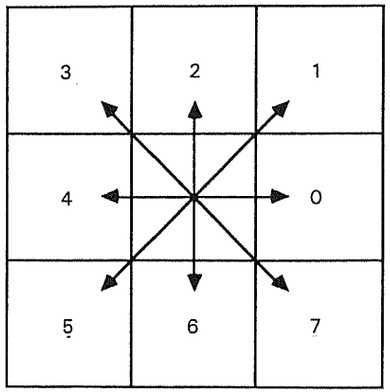

Figure 5.25 Freeman direction codes.

if $D = 1$ (boundary direction is 45°), then pixels 0, 1, and 2 are chosen as candidates (pixel labels adhere to the convention described in Chapter 3: see Figure 3.6, i.e. they are labelled sequentially from 0 to 7 in an anti-clockwise rotation from the middle right-hand pixel in a 3×3 neighbourhood).

The potential of each of these candidates to be a boundary point is evaluated as the difference between the points gradient magnitude and the predefined gradient magnitude threshold. If the pixel has been visited previously, either by the tracing or searching algorithms, or if the pixel overlaps the image boundary, then it is assumed to have a negative potential. The candidate with the highest positive potential is selected as the next boundary point from which to continue the trace and is implicitly included in the list of boundary points by updating the BCC. If no candidate satisfies this condition, tracing is suspended and the search algorithm is invoked.

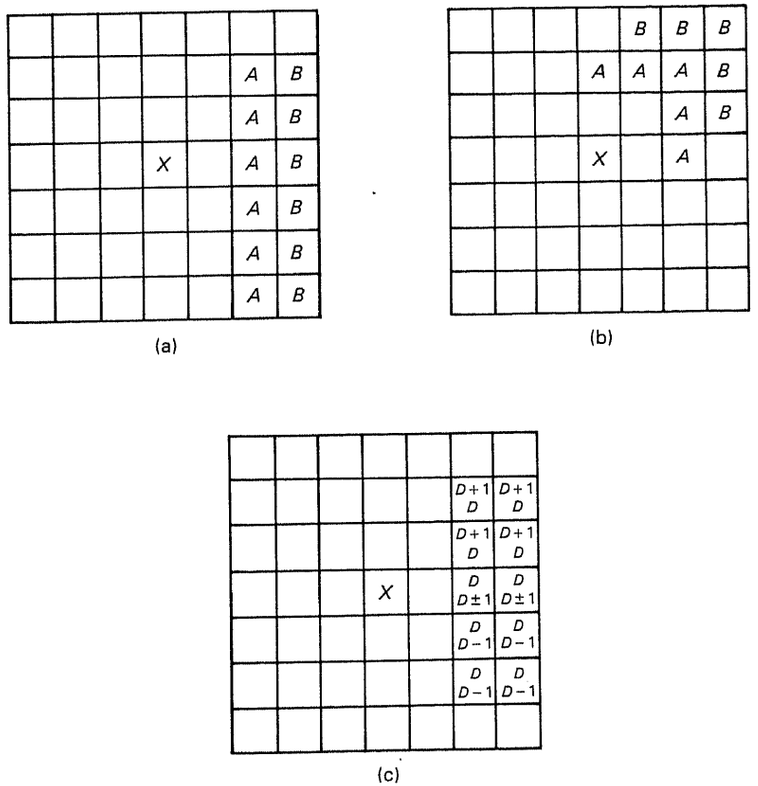

Figure 5.26 (a) Zone-of-influence *A*. (b) Zone-of-influence *B*. (c) Allowable directional variation of zones *A* and *B*.

□ *The search algorithm*

If all three neighbouring candidate boundary pixels selected by the trace algorithm fail to satisfy the criteria for inclusion in the boundary, then the boundary follower searches for other potential boundary pixels which are not immediate neighbours of the current boundary pixel. The position and direction of this search is dependent on the direction of the current boundary point; the associated region for searching is termed the 'zone of influence' of the current boundary pixel. Two zones of influence, labelled A and B, of a boundary pixel are defined, based on the boundary direction D. Zone A comprises those pixels ahead of the boundary direction, separated from the current point by a one-pixel gap. Pixels in zone B are separated by a two-pixel gap. Figure 5.26(a) shows the pixels in zones A and B with $D = 0$. Figure 5.26(b) shows these zones when $D = 1$. The zone A configuration for directions $D = 2, 4$, and 6 is simply a rotation of the configuration where $D = 0$; similarly, simple rotations of zones A and B corresponding for direction $D = 1$ define the configurations for directions $D = 3, 5$, and 7.

A pixel in a particular zone of influence is selected for inclusion in the boundary if it satisfies the following two criteria:

(a) the gradient magnitude is greater than the given threshold;
(b) the gradient direction is within a certain directional range.

This directional range is based on the orientation of the current boundary point and the position of this candidate pixel relative to the current boundary point. Figure 5.26(c) defines the allowable directional variations for zones A and B, with the boundary direction equal to zero. The directional variations corresponding to other orientations may be determined simply by substituting the new direction for D in the diagram.

The order in which the individual pixels are tested within a zone is important. In general, they are searched in the following order:

Direction of boundary → 3 1 0 2 4

Since searching terminates as soon as an acceptable edge point is encountered, this ordering favours the extraction of a point that coincides with the boundary direction. In addition, zone A is searched before zone B; thus, the bridging of short boundary gaps is favoured over larger gaps. If a search is successful the BCC is updated to include the selected pixel and the intermediate gap pixel(s). If the search is not successful, boundary following in the current bias (forward or reverse) is terminated.

Figure 5.24 illustrates the operating of this contour following algorithm.

Exercises

1. Discuss thresholding as a segmentation process. Describe three possible methods for automatic threshold selection.
2. Gradient-based edge detection operators require some method for approximating the partial derivatives of an intensity function (i.e. an image) in two orthogonal directions. Why? How can the gradient direction and magnitude be estimated from these partial derivatives? Identify one simple gradient operator and explain how it affects each of the above components of edge detection.
3. What is the relationship of the Laplacian operator and the gradient operator? Identify a significant shortcoming of the Laplacian operator and state how it is ameliorated by the Marr–Hildreth operator.
4. Describe how the Marr–Hildreth theory of edge detection might be used to facilitate automatic threshold selection for binary image segmentation.
5. How would the contour following algorithm described in this chapter fare in a noisy image?

References and further reading

Abdou, I.E. and Pratt, W.K. 1979 'Quantitative design and evaluation of enhancement/thresholding edge detectors', *Proceedings of the IEEE*, Vol. 67, No. 5, (May), pp. 753–63.

Argyle, E. 1971 'Techniques for edge detection', *Proc. IEEE*, Vol. 59, pp. 285–7.

Bellman, R. and Dreyfus, S. 1962 *Applied Dynamic Programming*, Princeton University Press, Princeton, New Jersey.

Canny, J. 1986 'A computational approach to edge detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp. 679–98.

Cooper, D. and Sung, F. 1983 'Multiple-window parallel adaptive boundary finding in computer vision', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 3, pp. 299–316.

Davis, L.S. 1975 'A survey of edge detection techniques', *Computer Graphics and Image Processing*, Vol. 4, No. 3, pp. 248–70.

Fischler, M.A. and Bolles, R.C. 1986 'Perceptual organisation and curve partitioning', *IEEE Transactions of Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1, pp. 100–5.

Fram, J.R. and Deutsch, E.S. 1975 'On the quantitative evaluation of edge detection schemes and their comparison with human performance', *IEEE Transactions of Computers*, Vol. C-24, No. 6, pp. 616–28.

Frei, W. and Chen, C-C. 1977 'Fast boundary detection: a generalization and a new algorithm', *IEEE Transactions on Computers*, pp. 988–98.

Giordano, A., Maresca, M., Sandini, G., Vernazza, T. and Ferrari, D. 1985 *A Systolic*

Convolver for Parallel Multiresolution Edge Detection, Internal Report, DIST – University of Genoa.

Giordano, A., Maresca, M., Sandini, G., Vernazza, T. and Ferrari, D. 1987 'VLSI-based systolic architecture for fast Gaussian convolution', *Optical Engineering*, Vol. 26, No. 1, pp. 63–8.

Griffith, A.K. 1973 'Edge detection in simple scenes using *a priori* information', *IEEE Transactions on Computers*, Vol. 22, No. 4, pp. 371–80.

Grimson, W.F.L. and Hildreth, E.C. 1985 'Comments on digital step edges from zero crossings of second directional derivatives', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 1, pp. 121–7.

Guari, E. and Wechsler, H. 1982 'On the difficulties involved in the segmentation of pictures', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 3.

Haralick, R.M. 1984 'Digital step edges from zero-crossing of second directional derivatives', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 1, pp. 58–68.

Hildreth, E.C. 1981 'Edge detection in man and machine', *Robotics Age*, Sept/Oct, pp. 8–14.

Hildreth, E.C. 1985 *Edge Detection*, AI Memo No. 858, MIT AI Lab.

Hueckel, M. 1971 'An operator which locates edges and digitises pictures', *JACM*, Vol. 18, No. 1, pp. 113–25.

Hueckel, M. 1973 'A local visual operator which recognises edges and lines', *JACM*, Vol. 20, No. 4, pp. 643–7.

Hueckel, M. 1973 'Erratum to "a local visual operator which recognizes edges and lines"', *JACM*, Vol. 21, No. 2, p. 350.

Huertas, A. and Medioni, G. 1986 'Detection of intensity changes with subpixel accuracy using Laplacian–Gaussian masks', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 5, pp. 651–64.

Jacobus, C. and Chien, R. 1981 'Two new edge detectors', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 5, pp. 581–92.

Juvin, D. and de Cosnac, B. 1984 'ANIMA 2: Un Systeme Generale de Vision Pour la Robotique', *Proceedings of the Premier Colloque Image*, CESTA, Biarritz, pp. 165–9.

Kasvand, T. 1975 'Iterative edge detection', *Computer Graphics and Image Processing*, Vol. 4, pp. 279–86.

Kelly, M.D. 1971 'Edge detection in pictures by computer using planning', *Machine Intelligence*, B. Meltzer and D. Michie (eds.), Vol. 6, pp. 397–409, Edinburgh, Edinburgh University Press.

Kirsch, R. 1971 'Computer determination of the constituent structure of biological images', *Computers and Biomedical Research*, Vol. 4, No. 3, pp. 315–28.

Lee, C.C. 1983 'Elimination of redundant operations for a fast Sobel operator', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 3, pp. 242–5.

Lunscher, W.F. and Beddoes, M.P. 1986 'Optimal edge detector design I: Parameter selection and noise effects', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2, pp. 164–77.

Lunscher, W.F. and Beddoes, M.P. 1986 'Optimal edge detector design II: Coefficient quantisation', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2, pp. 178–87.

Marr, D. 1976 'Early processing of visual information', *Philosophical Transactions of the Royal Society of London*, **B275**, pp. 483–524.

Marr, D. and Hildreth, E. 1980 'Theory of edge detection', *Proceedings of the Royal Society of London*, **B207**, pp. 187–217.

Mero, L. and Vassy, Z. 1975 'A simplified and fast version of the Hueckel operator for finding optimal edges in pictures', *Proceedings of the International Joint Conference on Artificial Intelligence, Tbilisi, Georgia, USSR*, pp. 650–5.

Nalwa, V.S. and Binford, T.O. 1986 'On detecting edges', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp. 699–714.

Nazif, A.M. and Levine M.D. 1984 'Low level segmentation: an expert system', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 5, pp. 555–77.

Nevatia, R. 1977 'Note: Evaluation of a simplified Hueckel edge-line detector', *Journal of Computer Graphics and Image Processing*, pp. 582–8.

Nevatia, R. and Babu, K. 1980 'Linear feature extraction and description', *Computer Graphics and Image Processing*, Vol. 13, pp. 257–69.

Perkins, W.A. 1980 'Area segmentation of images using edge points', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 1, pp. 8–15.

Prewitt, J.M.S. 1970 'Object enhancement and extraction' in *Picture Processing and Psychopictorics*, B. Lipkin and A. Rosenfeld (eds.), Academic Press, New York, pp. 75–149.

Roberts, L.G. 1965 'Machine perception of three-dimensional solids' in *Optical and Electro-Optical Information Processing*, J.T. Tippett *et al.* (eds.), MIT Press, Cambridge, Massachusetts, pp. 159–97.

Rosenfeld, A. and Thurston, M. 1971 'Edge and curve detection for visual scene analysis', *IEEE Transactions on Computers*, Vol. C-20, No. 5, pp. 562–9.

Sandini, G. and Torre, V. 1985 'Thresholding techniques for zero-crossings', *Proceedings of 'Winter '85 Topical Meeting on Machine Vision', Incline Village, Nevada.*

Torre, V. and Poggio, T.A. 1986 'On edge detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2, pp. 147–63.

Weszka, J.S. 1978 'A survey of threshold selection techniques', *Computer Graphics and Image Processing*, Vol. 7, pp. 259–65.

Weszka, J.S., Nagel, R.N. and Rosenfield, A. 1974 'A threshold selection technique', *IEEE Transactions on Computers*, Vol. C-23, No. 12, pp. 1322–7.

Wiejak, J.S. 1983 'Regions estimation and boundary estimation', *Image and Vision Computing*, Vol. 1, No. 2, May 1983, pp. 99–102.

Yakimovsky, Y. 1976 'Boundary and object detection in real world images', *JACM*, Vol. 23, No. 4, pp. 599–618.