tersection at lower levels. In the converse case prune the tree sequentially by first intersecting the resultant pruned tree with the right region strip.

Algorithm 8.5: Curve–Region Intersection

```
comment A Reference Procedure returns a pointer;
reference procedure CurveRegionInt(T1,T2)
 begin
 A := T2;
 comment R is a global used by CRInt;
 return (CRInt(T1,T2));
 end;

reference procedure CRInt(T1,T2)
 begin
 begin Case StripInt(T1,T2) of
  [Null or Primitive]
   if intersection (T1,R,TRUE) = null then
    if Inside(T1,R) then return (T1)
    else return (null);
   else return (T1);
   [Possible] if T1 is "fatter" then
   begin
    NT := NewRecord;
    x_b(NT) := x_b (T);
    x_e(NT) := x_e (T);
    w_l(NT) := w_l(T);
    w_r(NT) := w_r(T);
    LSon(NT) := CRInt (LSon(T1),T2);
    RSon(NT) := CRInt (RSon(T1),T2);
    return(NT);
    end
   else comment T2 is "fatter"
    Return (CRInt(CRInt(T1,LSon(T2)),RSon(T2)));
   end;
end Case;
end;
```

The problem of intersecting two regions can be decomposed into two curve-region intersection problems (Fig. 8.16). Thus algorithm 8.5 can also be used to solve the region-region intersection problem.

8.3 REGION REPRESENTATIONS

8.3.1 Spatial Occupancy Array

The most obvious and quite a useful representation for a region on a raster is a membership predicate $p(x, y)$ which takes the value 1 when point (x, y) is in the

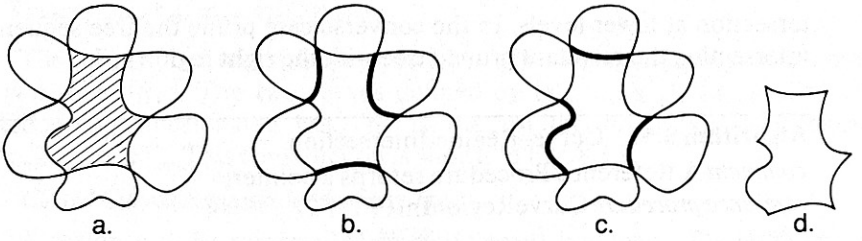


Fig. 8.16 Decomposition of Region-Region Intersection. (a) Desired result. (b) Portion of boundary generated by treating three-lobed region as a curve. (c) Portion of boundary generated by treating five-lobed region as a curve. (d) Result of union operation.

region and the value 0 otherwise. One easy way to implement such a function is with a *membership array*, an array of 1's and 0's with the obvious interpretation. Such arrays are quicky interrogated and also quite easily unioned, merged and intersected by AND and OR operations, applied elementwise on the operand arrays. The disadvantages of this representation are that it requires much space and does not represent the boundary in a useful way.

8.3.2 *y*-Axis

A representation that is more compact and which offers reasonable algorithms for intersection, merging, and union is the *y*-axis representation [Merrill 1973]. This is a run-length encoding of the membership array, and as such it provides no explicit boundary information. It is a list of lists. Each element on the main list corresponds to a row of constant y in the image raster. Each row of constant y is encoded as a list of x-coordinate points; the first x point at which the region is entered while moving along that y row, then the x point at which the region is exited, then the x point at which it next is entered, and so forth. The y-rows with no region points are omitted from the main list. Thus, in a notation where successive levels of sublist are surrounded by successive levels of parentheses, the y-axis encoding of a region is shown in Fig. 8.17; here the first element of each sublist is the y coordinate, followed by a list of "into" and "out of" x coordinates. Where a y coordinate con-

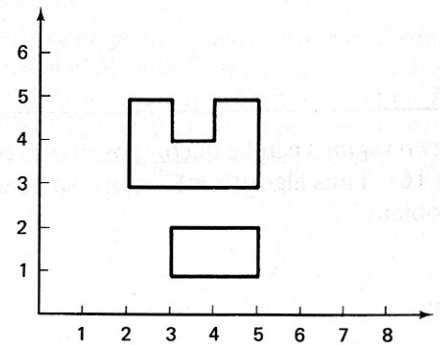


Fig. 8.17 *y*-axis region representation.

tains an isolated point in the region, this point is repeated in the x-axis representation, as shown by the example in Fig. 8.17. Thus "lines" (regions of unit width) can be easily (although not efficiently) represented in this system.

Union and intersection are implemented on y-axis representations as merge-like operations which take time linearly proportional to the number of y rows. Two instances of y-axis representations and the representation of their union are shown in Fig. 8.18. Note that the union amounts to a merge of x elements along rows organized within a merge of rows themselves.

The y-axis representation is wasteful of space if the region being represented is long, thin, and parallel to the y axis. In this case one is invited to encode it in x-axis format, in an obvious extension. Working with mixed x-axis and y-axis formats presents no conceptual difficulties, but considerable loss of convenience.

8.3.3 Quad Trees

Quad trees [Samet 1980] are a useful encoding of the spatial occupancy array. The easiest way to understand quad trees is to consider pyramids as an intermediate representation of the binary array. Figure 8.19 shows a pyramid (Section 3.7) made from the base image (on the left). Each pixel in images above the lowest level has one of three values, BLACK, WHITE, or GRAY. A pixel in a level above the base is BLACK or WHITE if all its corresponding pixels in the next lower level are BLACK or WHITE respectively. If some of the lower level pixels are BLACK and others are WHITE, the corresponding pixel in the higher level is GRAY.

Such a pyramid is easy to construct. To convert the pyramid to a quad tree, simply search the pyramid recursively from the top to the base. If an array element in the pyramid is either BLACK or WHITE, form a terminal node of the corresponding type. Otherwise, form a GRAY node with pointers to the results of

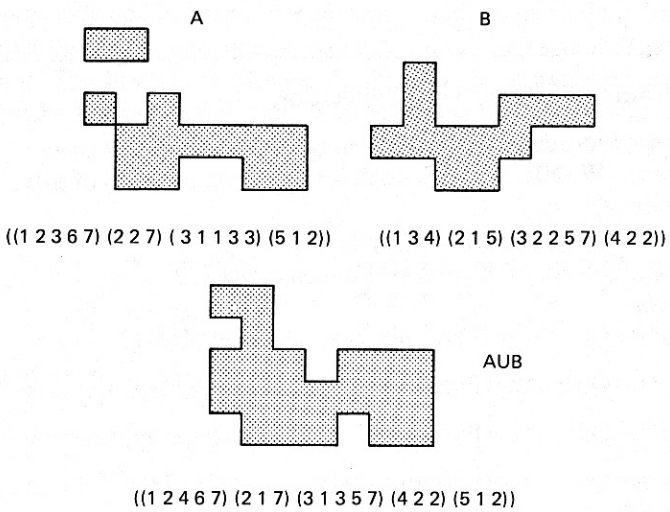


Fig. 8.18 Two point sets A, B, and $A \cup B$, with their y-axis representations.

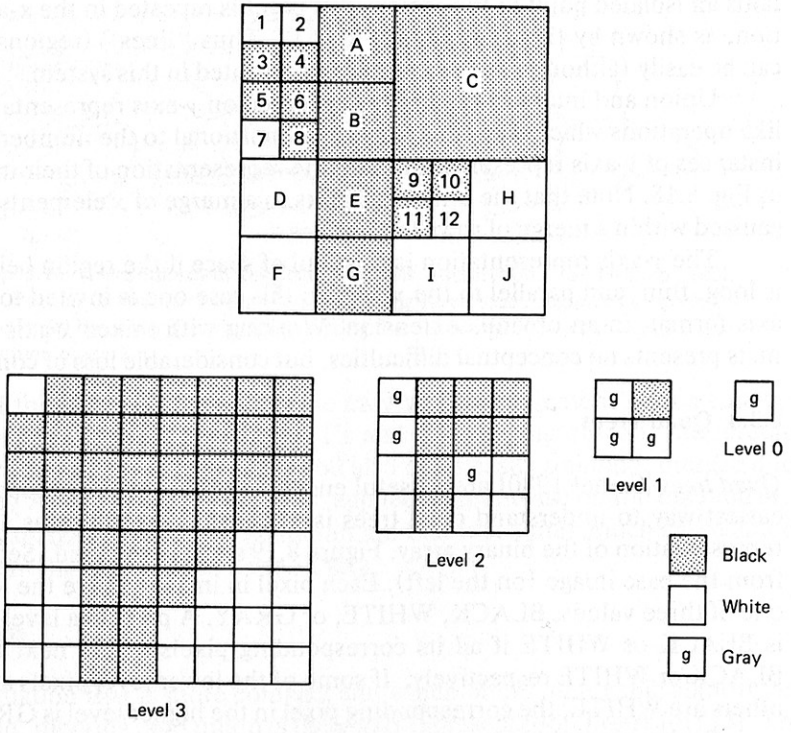


Fig. 8.19 Pyramid used in quad tree construction. Letters correspond to pixels in the pyramid that are either BLACK or WHITE.

the recursive examination of the four elements at the next level in the tree (Algorithm 8.6).

Algorithm 8.6: Quad Tree Generation

```
Reference Procedure QuadTree (integer array pyramid; integer x, y, level);
Comment NW, NE, SW, SE are fields denoting the sons of a quadtree node;
Newnode(P);
TYPE(P) := Pyramid(IND (x,y,Level));
if TYPE(P) = BLACK or WHITE then return (P)
else begin
  SW(P):=QuadTree(Pyramid, 2*x, 2*y, Level + 1);
  SE(P):=QuadTree(Pyramid, 2*x + 2*Level, 2*y, Level + 1);
  NW(P):=QuadTree(Pyramid, 2*x, 2*y + 2*Level, Level + 1);
  NE(P):=QuadTree(Pyramid, 2*(x + Level), 2*(y + Level), Level + 1);
  return (P)
  end;
```

Here an implementational point is that the entire pyramid fits into a linear array of size $2(2^{2\times level})$. IND is an indexing function which extracts the appropriate value given the x, y and level coordinates. The reader can apply this algorithm to the example in Fig. 8.19 to verify that it creates the tree in Fig. 8.20.

The quad tree can be created directly from the base of the pyramid, but the algorithm is more involved. This is because proceeding upward from the base, one must sometimes defer the creation of black and white nodes. This algorithm is left for the exercises [Samet 1980].

Many operations on quad trees are simple and elegant. For example, consider the calculation of area [Schneier 1979]:

Algorithm 8.7: Area of a Quad Tree

```
Integer Procedure Area (reference QuadTree; integer height)
  Begin
  Comment NW, NE, SW, SE are fields denoting the sons of
  a quadtree node;
  BlackArea := 0;
  if TYPE(QuadTree) = GRAY then
   for I in the set {NW, NE, SW, SE} do
    BlackArea = BlackArea + Area(I(QuadTree), height-1)
  else if TYPE(QuadTree) = BLACK then
   BlackArea = BlackArea + 2^2*height;
  return(BlackArea)
  end;
```

Other examples may be found in the References and are pursued in the Exercises.

The quad tree and the associated pyramid have two related disadvantages as a representation. The first is that the resolution cannot be extended to finer resolution after a grid size has been chosen. The second is that operations between quad

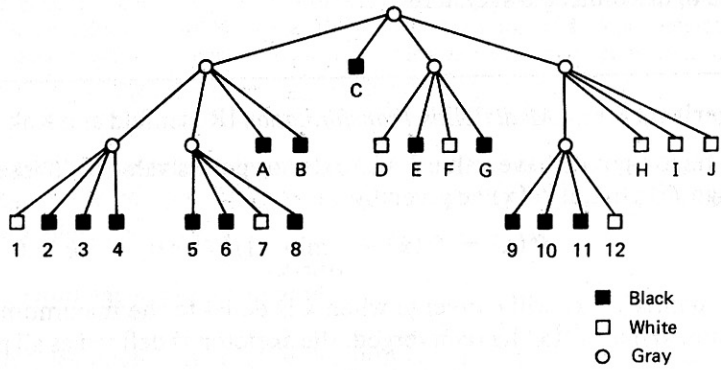


Fig. 8.20 Quad tree for the example in Fig. 8.19.

trees tacitly assume that their pyramids are defined on the same grids. The grids cannot be shifted or scaled without cumbersome conversion routines.

8.3.4 Medial Axis Transform

If the region is made of thin components, it can be well described for many purposes by a "stick-figure" *skeleton.* Skeletons may be derived by thinning algorithms that preserve connectivity of regions; the medial axis transform (MAT), of [Blum 1973; Marr 1977] is a well-known thinning algorithm.

The skeleton is defined in terms of the distance of a point $\mathbf{x}$ to a set A:

$$d_s(\mathbf{x}, A) = \inf\{d(\mathbf{x}, \mathbf{z}) | \mathbf{z} \text{ in } A\} \tag{8.15}$$

Popular metrics are the Euclidean, city block, and chessboard metrics described in Chapter 2.

Let B be the set of boundary points. For each point P in a region, find its closest neighbors (by some metric) on the region boundary. If *more than one* boundary point is the minimum distance from $\mathbf{x}$, then $\mathbf{x}$ is on the skeleton of the region. The skeleton is the set of pairs $\{\mathbf{x}, d_s(\mathbf{x}, B)\}$ where $d_s(\mathbf{x}, B)$ is the distance from $\mathbf{x}$ to the boundary, as defined above (this is a definition, not an efficient algorithm.) Since each $\mathbf{x}$ in the skeleton retains the information on the minimum distance to the boundary, the original region may be recovered (conceptually) as the union of "disks" (in the proper metric) centered on the skeleton points.

Some common shapes have simply structured medial axis transform skeletons. In the Euclidean metric, a circle has a skeleton consisting of its central point. A convex polygon has a skeleton consisting of linear segments; if the polygon is nonconvex, the segments may be parabolic or linear. A simply connected polygon has a skeleton that is a tree (a graph with no cycles). Some examples of medial axis transform skeletons appear in Fig. 8.21.

The figure shows that the skeleton is sensitive to noise in the boundary. Reducing this sensitivity may be accomplished by smoothing the boundary, using a polygonal boundary approximation, or including only those points in the skeleton that are greater than some distance from the boundary. The latter scheme can lead to disconnected skeletons.

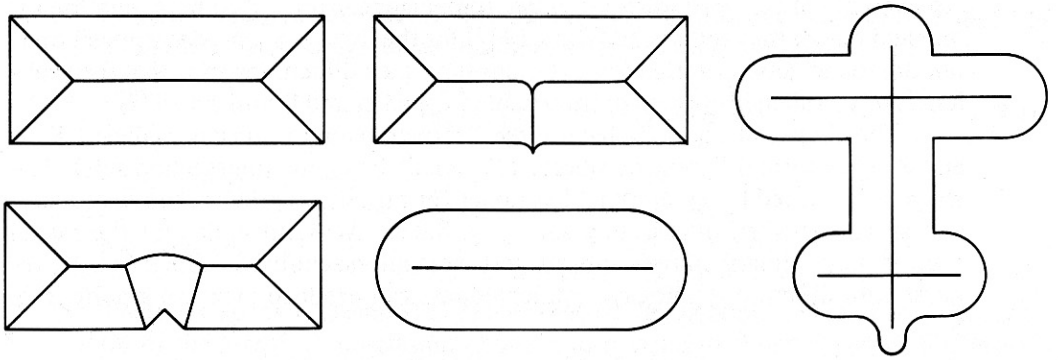
Algorithm 8.8: *Medial Axis Transformation* [Rosenfeld and Kak 1976]

Let region points have value 1 and exterior points value 0. These points define an image $f^0(\mathbf{x})$. Let $f^k(\mathbf{x})$ be given by

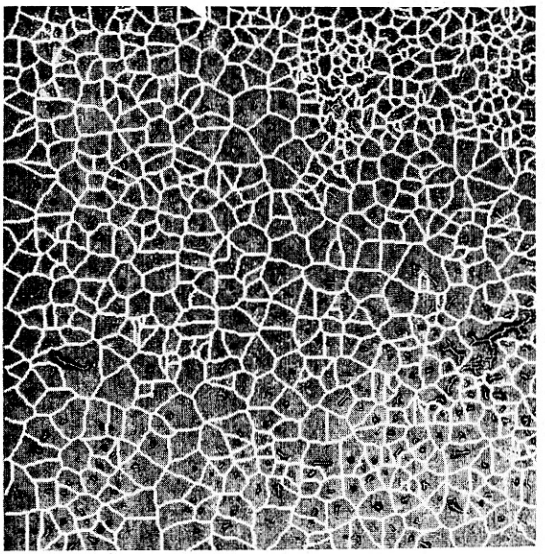
$$f^k(\mathbf{x}) = f^0(\mathbf{x}) + \min_{d(x,z)\leq 1} [f^{k-1}(\mathbf{z})], \qquad k > 0$$

The points $f^k(\mathbf{x})$ will converge when k is equal to the maximum thickness of the region. Where $f^k(\mathbf{x})$ has converged, the skeleton is defined as all points $\mathbf{x}$ such that

$$f^k(\mathbf{x}) \geqslant f^k(\mathbf{z}), \qquad d(\mathbf{x}, \mathbf{z}) \leqslant 1.$$



(a)



(b)

Fig. 8.21 Medial Axis Transform skeletons (a), and the technique applied to human cell nuclei (b). Shown in (b) are both the "normal" skeleton obtained by measuring distances interior to the boundaries, and the exo-skeleton, obtained by measuring distances exterior to the boundary.

This algorithm can produce disconnected skeletons for excursions or lobes off the main body of the region. Elegant thinning algorithms to compute skeletons are given in [Pavlidis 1977].

8.3.5 Decomposing Complex Regions

Much work has been done on the decomposition of point sets (usually polygons) into a union of convex polygons. Such convex decompositions provide structural analysis of a complex region that may be useful for matching different point sets.

An example of the desired result in two dimensions is presented here, and the interested reader may refer to [Pavlidis 1977] for the details. Such a decomposition is not unique in general and in three dimensions, such difficulties arise that the problem is often called ill-formed or intractable [Voelcker and Requicha 1977].

The shapes of Fig. 8.22 have three "primary convex subsets" labeled *X, Y,* and *Z*. They form different numbers of "nuclei" (roughly, intersection sets). The shape is described by a graph that has nodes for nuclei and primary convex subsets and an arc between intersecting sets (Fig. 8.22c). Without nodes for the nuclei (i.e., if only primary convex subsets and their intersections are represented), regions with different topological connectedness can produce identical graphs (Fig. 8.22b).

8.4 SIMPLE SHAPE PROPERTIES

8.4.1 Area

The *area* of a region is a basic descriptive property. It is easily computed from curve boundary representations (8.3.1) and thus also for chain codes (8.3.2); their con-

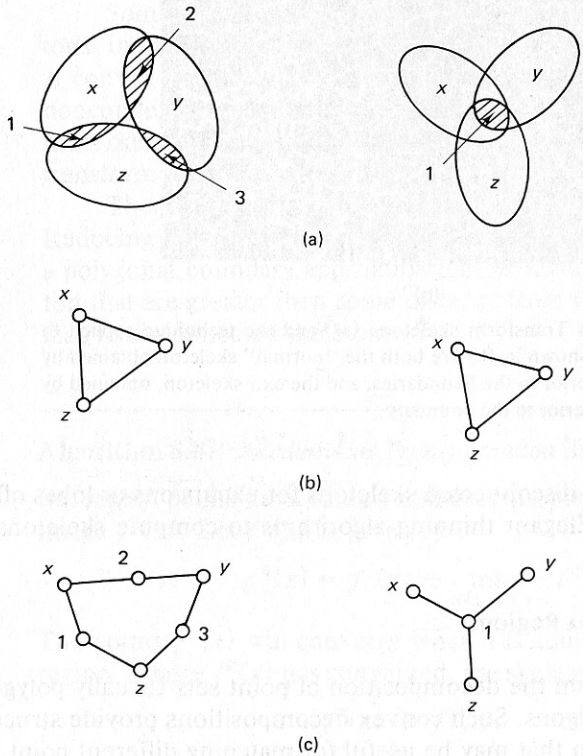


Fig. 8.22 Decomposition of polygon into primary convex subsets and nuclei (see text).

tinuous analog is also useful. Consider a curve parameterized on arc length s so that points (x, y) are given by functions $(x(s), y(s))$

$$\text{area} = \int_0^P \left(x \frac{dy}{ds} - y \frac{dx}{ds}\right) ds \tag{8.16}$$

where P is the perimeter.

8.4.2 Eccentricity

There are several measures of *eccentricity*, or "elongation". One of them is the ratio of the length of maximum chord A to maximum chord B perpendicular to A (Fig. 8.23).

Another reasonable measure is the ratio of the principal axes of inertia; this measure can be based on boundary points or the entire region [Brown 1979]. An (approximate) formula due to Tenenbaum for an arbitrary set of points starts with the mean vector

$$\mathbf{x}_0 = \frac{1}{n} \sum_{\mathbf{x} \text{ in } R} \mathbf{x} \tag{8.17}$$

To compute the remaining parameters, first compute the ijth moments M_{ij} defined by

$$M_{ij} = \sum_{\mathbf{x} \text{ in } R} (x_0 - x)^i (y_0 - y)^j \tag{8.18}$$

The orientation, θ, is given by

$$\theta = \frac{1}{2} \tan^{-1}\left(\frac{2M_{11}}{M_{20} - M_{02}}\right) + n\left(\frac{\pi}{2}\right) \tag{8.19}$$

and the approximate eccentricity e is

$$e = \frac{(M_{20} - M_{02})^2 + 4M_{11}}{\text{area}} \tag{8.20}$$

8.4.3 Euler Number

The Euler number is a topological property defining the set of objects that are equivalent under "rubber-sheet" deformations of the plane. It describes the connectedness of a region, not its shape. A *connected region* is one in which all pairs of

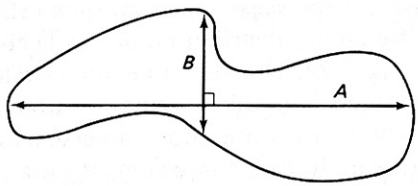


Fig. 8.23 An eccentricity measure: A / B.

points may be connected by a curve lying entirely in the region. If a complex two-dimensional object is considered to be a set of connected regions, where each one can have holes, the *Euler number* for such an object is defined as

$$\text{(number of connected regions)} - \text{(number of holes)}$$

The number of holes is one less than the connected regions in the set complement of the object.

8.4.4 Compactness

One measure of *compactness* (not compactness in the sense of point-set topology) is the ratio (perimeter^2)/area, which is dimensionless and minimized by a disk. This measure is computed easily from the chain-code representation of the boundary where the length of an individual segment of eight-neighbor chain code is given by $(\sqrt{2})$ if the (eight-neighbor) direction is odd and by 1 if the direction is even. The area is computed by a modification of Algorithm 8.2 and the perimeter may be accumulated at the same time.

For small discrete objects, this measure may not be satisfactory; another measure is based on a model of the boundary as a thin springy wire [Young et al. 1974]. The normalized "bending energy" of the wire is given by

$$E = \frac{1}{P}\int_0^P |\kappa(s)|^2 ds \tag{8.21}$$

where κ is *curvature*. This measure is minimized by a circle. E can be computed from the chain code representation by recognizing that $\kappa = d\theta/dS$, and also from the Fourier coefficients mentioned below since

$$|\kappa(s)|^2 = \left(\frac{d^2x}{ds^2}\right)^2 + \left(\frac{d^2y}{ds^2}\right)^2 \tag{8.22}$$

so that E, using Parseval's theorem, is

$$\sum_{k=-\infty}^{\infty} (kw_0)^4(|X_k|^2 + |Y_k|^2) \tag{8.23}$$

where $\mathbf{X}_k = (X_k, Y_k)$ are the Fourier descriptor coefficients in (8.2).

8.4.5 Slope Density Function

The ψ–s curve can be the basis for the *slope density function* (SDF) [Nahin 1974]. The SDF is the histogram or frequency distribution of ψ collected over the boundary. An example is shown in Fig. 8.24. The SDF is flat for a circle (or in a continuous universe, any shape with a monotonically varying ψ); straight sides stand out sharply, as do sharp corners, which in a continuous universe leave gaps in the histogram. The SDF is the signature of the ψ–s curve along the ψ axis.

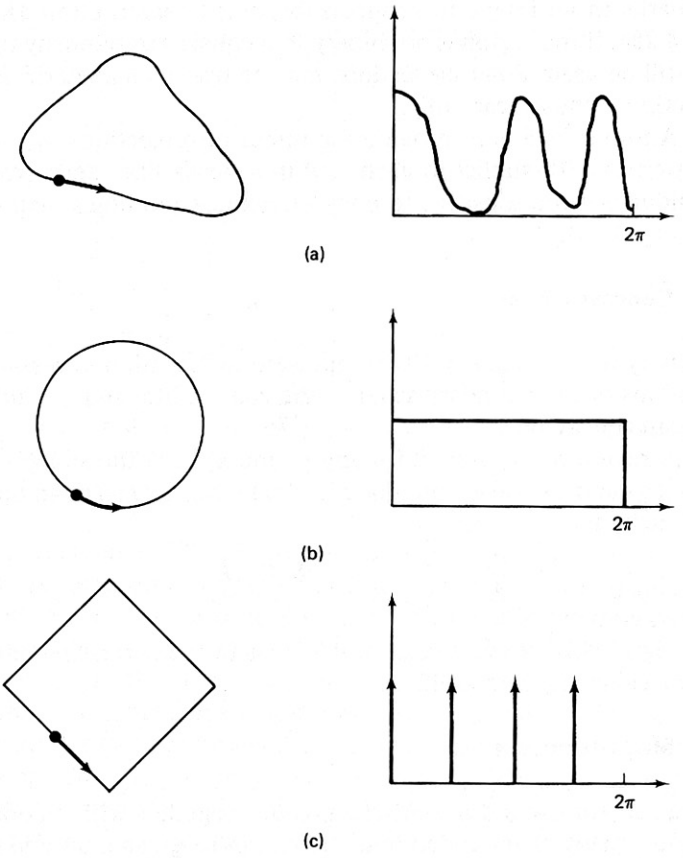


Fig. 8.24 The Slope Density Function for three curves: a triangular blob, a circle, and a square.

8.4.6 Signatures

By definition, a projection is not an information-preserving transformation. But Section 2.3.4 showed that (as with Fourier descriptors,) enough projections allow reconstruction of the region to any desired degree of accuracy. (This observation forms the basis for computer assisted tomography.)

Given a binary image $f(\mathbf{x}) = 0$ or 1, define the horizontal *signature* $p(x)$ as

$$p(x) = \int_y f(x, y) \tag{8.24}$$

$p(x)$ is simply the projection of p onto the x axis. Similarly, define $p(y)$, the vertical signature, as

$$p(y) = \int_x f(x, y) \tag{8.25}$$

Maxima and minima of signatures are often useful for establishing preliminary

landmarks in an image to reduce subsequent search effort [Kruger et al. 1972] (Fig. 8.25). If the region is not binary, but consists of a density function, Eq. (8.24) may still be used. Polar projections may be useful characterizations if the point of projection is chosen carefully.

Another idea is to provide a number of projections, $q_1, \ldots, q_n$, the ith one based on the ith sublist in each row in a y-axis-like region representation. This technique is more sensitive to non-convexities and holes than is a regular projection (Fig. 8.26).

8.4.7 Concavity Tree

Concavity trees [Sklansky 1972] represent information necessary to fill in local indentations of the boundary as far as the convex hull and to study the shape of the resultant concavities.

A region S is *convex* iff for any $\mathbf{x}_1$ and $\mathbf{x}_2$ in S, the straight line segment connecting $\mathbf{x}_1$ and $\mathbf{x}_2$ is also contained in S. The *convex hull* of an object S is the smallest H such that

$$S \subset H$$

and H is convex.

Figure 8.27 shows a region, the steps in the derivation of the concavity tree, and the concavity tree itself.

8.4.8 Shape Numbers

For closed curves and a 3-bit chain code (together with a controlled digitization scheme), many chain-coded boundaries can be given a unique *shape number* [Bri-

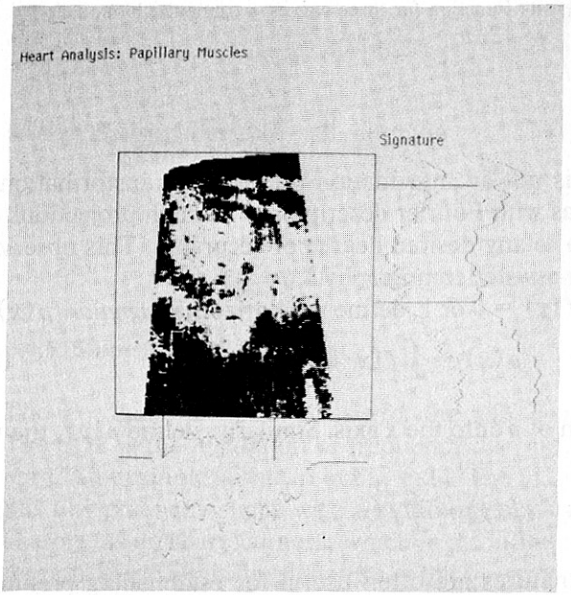


Fig. 8.25 The use of signatures to locate a left ventricle cross section in ultrasound data. (Outer curves are smoothed versions of inner signatures.)

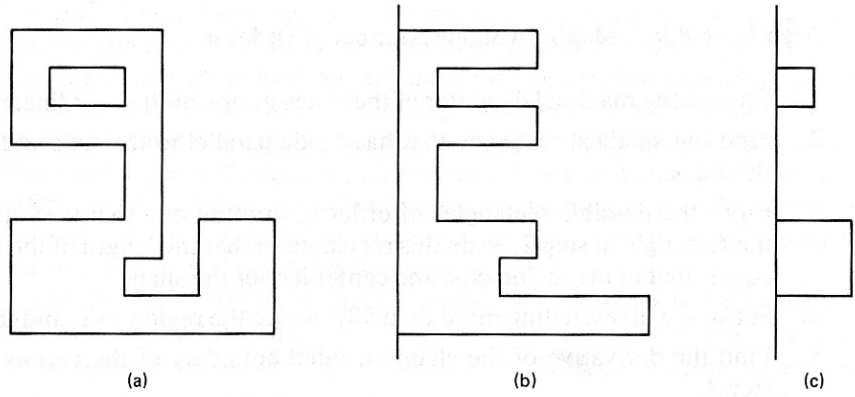


Fig. 8.26 A shape (a) and projections; from the first (b) and second (c) sublists of the y-axis representation.

biesca and Guzman 1979]. The shape number is related to the resolution of the digitization scheme. In a multiple resolution pyramid of digitization grids, every possible shape can be represented as a path through a tree. At each grid resolution corresponding to a level in the tree, there are a finite number of possible shapes. Moving up the tree, the coarser grids tend to blur distinctions between different shapes until at some resolution they are identical. This level can be used as a similarity measure between shapes. The basic idea behind shape numbers is the following. Consider all the possible closed boundaries with n chain segments. These form the possible shapes of "order n." The chain encoding for a particular boundary can be made unique by interpreting the chain-code direction sequence as a number and picking the start point that minimizes this number. Notice that the orders of shape numbers must be even on rectangular grids since a curve of odd order cannot close.

Algorithm 8.9 generates a shape number of order n.

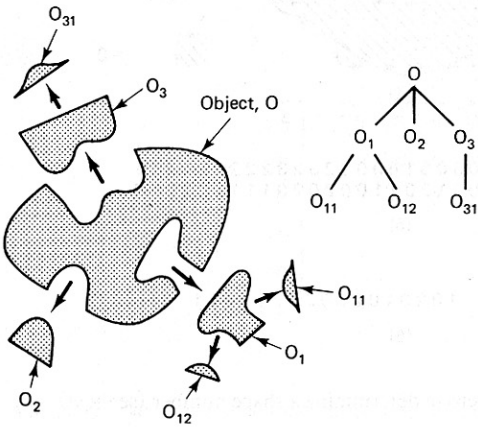


Fig. 8.27 Concavities of an object and the concavity tree.

Algorithm 8.9: Making a Shape Number of Order n

1. Choose the maximal diameter of the shape as one of the coordinate axes.
2. Find the smallest rectangle that has a side parallel to this axis and just covers the shape.
3. From the possible rectangles of order n, find the one that best approximates the rectangle in step 2. Scale this rectangle so that the length of the longest side equals that of the major axis, and center it over the shape.
4. Set all the pixels falling more than 50% inside the region to 1, and the rest to 0.
5. Find the derivative of the chain encoded boundary of the region of 1's from step 4.
6. Normalize this number by rotating the digits until the number is minimum. The normalized number is the shape number.

Figure 8.28 shows these steps.

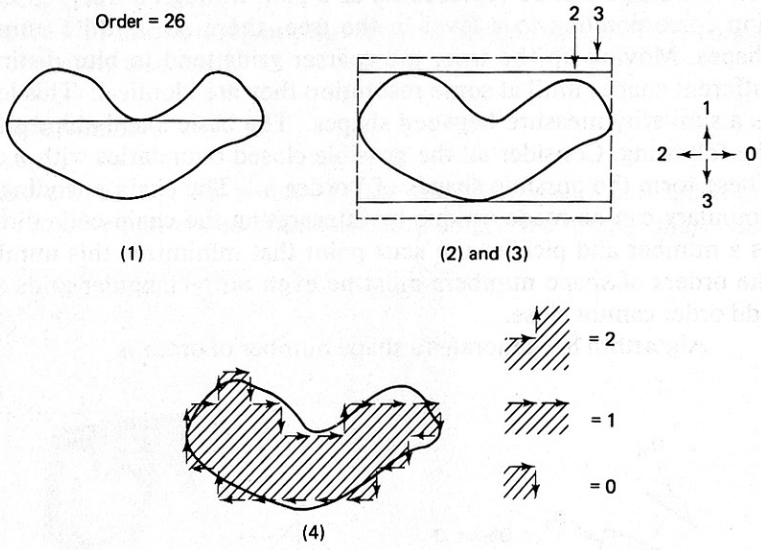


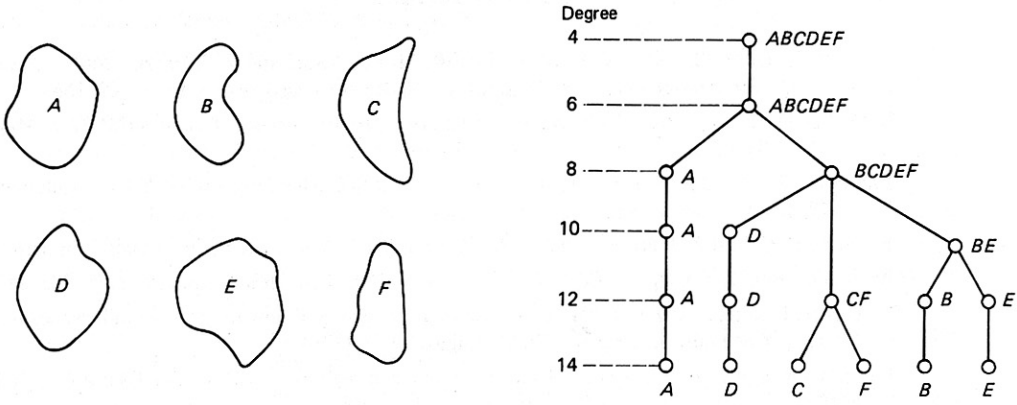
Fig. 8.28 Steps in determining a shape number (see text).

Generating a shape number of a specific order may be tricky, as there is a chance that the resulting shape number may be greater than order n due to deep concavities in the boundary. In this case, the generation procedure can be repeated for smaller values of n until a shape number of n digits is found. Even this strategy may sometimes fail. The shape number may not exist in special cases such as boundaries with narrow indentations. These features may cause step 4 in Algorithm 8.11 to fail in the following way. Even though the rectangle of step 3 was of order n, the resultant boundary may have a different order. Nevertheless, for the vast majority of cases, a shape number can be computed.

The degree of similarity for two shapes is the largest order for which their shape numbers are the same. The "distance" between two shapes is the inverse of their degree of similarity. This distance is an *ultradistance* rather than a norm:

$$\begin{aligned} d(S, S) &= 0 \\ d(S_1, S_2) &\geqslant 0 \qquad \text{for } S_1 \neq S_2 \\ d(S_1, S_3) &\leqslant \max(d(S_1, S_2), d(S_2, S_3)) \end{aligned} \tag{8.26}$$

Figure 8.29 shows the similarity tree for six shapes as computed from their shape numbers. When the shape number is well defined, it is a useful measure since it is unique (for each order), it is invariant under rotation and scale changes of an object, and it provides a metric by which shapes can be compared.



	A	B	C	D	E	F
A	∞	6	6	6	6	6
B		∞	8	8	10	8
C			∞	8	8	12
D				∞	8	8
E					∞	8
F						∞

Fig. 8.29 Six shapes, their similarity trees, and the ultradistances between the shapes.

EXERCISES

8.1 Consider a region segmentation where regions are of two types: (1) filled in and (2) with holes. Relate the number of junctions, boundaries, and filled-in regions to the Euler number.

8.2 Write a procedure for finding where two chain codes intersect.

8.3 Devise algorithms to intersect and union two regions in the y-axis representation.

8.4 Show that the number of intersections of the curves under a clear strip intersection is odd.

8.5 Modify Algorithm 8.4 to work with strip trees with varying numbers of sons.

8.6 Derive Eq. (8.9) from Eq. (8.7).

8.7 Show that Eqs. (8.12) and (8.13) are equivalent.

8.8 Given two points $\mathbf{x}_1$ and $\mathbf{x}_2$ and slopes $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_2)$, find the ellipse with major axis a that fits the points.

8.9 Write a procedure to intersect two regions represented by quad trees, producing the quad tree of the intersection.

8.10 Determine the shape numbers for (a) a circle and (b) an octagon. What is the distance between them?

REFERENCES

AMBLER, A. P., H. G. BARROW, C. M. BROWN, R. M. BURSTALL, and R. J. POPPLESTONE. "A versatile system for computer controlled assembly." *Artificial Intelligence 6*, 2, 1975, 129–156.

BALLARD, D. H. "Strip trees: A hierarchical representation for curves." *Comm. ACM 24*, 5, May 1981, 310–321.

BARNHILL, R. E. and R. F. RIESENFELD. *Computer Aided Geometric Design.* New York: Academic Press, 1974, 160.

BARROW, H. G. and R. J. POPPLESTONE. "Relational descriptions in picture processing." In *MI6*, 1971.

BLUM, H. "Biological shape and visual science (Part I)." *J. Theoretical Biology 38*, 1973, 205–287.

BRIBIESCA, E. and A. GUZMAN. "How to describe pure form and how to measure differences in shapes using shape numbers." *Proc.*, PRIP, August 1979, 427–436.

BRICE, C. R. and C. L. FENNEMA. "Scene analysis using regions." *Artificial Intelligence 1*, 3, Fall 1970, 205–226.

BROWN, C. M. "Two descriptions and a two-sample test for 3-d vector data." TR49, Computer Science Dept., Univ. Rochester, February 1979.

DEBOOR, C. *A Practical Guide to Splines.* New York: Springer-Verlag, 1978.

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

FREEMAN, H. "Computer processing of line drawing images." *Computer Surveys 6*, 1, March 1974, 57–98.

GALLUS, G. and P. W. NEURATH. "Improved computer chromosome analysis incorporating preprocessing and boundary analysis." *Physics in Medicine and Biology 15*, 1970, 435.

GORDON, W. J. "Spline-blended surface interpolation through curve networks." *J. Mathematics and Mechanics 18*, 10, 1969, 931–952.

HOROWITZ, S. L. and T. P. PAVLIDIS. "Picture segmentation by a tree traversal algorithm." *J. ACM 23*, 2, April 1976, 368–388.

KRUGER, R. P., J. R. TOWNE, D. L. HALL, S. J. DWYER, and G. S. LUDWICK, "Automatic radiographic diagnosis via feature extraction and classification of cardiac size and shape descriptors." *IEEE Trans. Biomedical Engineering 19*, 3, May 1972.

MARR, D. "Representing visual information." AI Memo 415, AI Lab, MIT, May 1977.

MERRILL, R. D. "Representations of contours and regions for efficient computer search." *Comm. ACM 16*, 2, February 1973, 69–82.

NAHIN, P. J. "The theory and measurement of a silhouette descriptor for image preprocessing and recognition." *Pattern Recognition 6*, 2, October 1974.

PATON, K. A. "Conic sections in automatic chromosome analysis." In *MI5*, 1970.

PAVLIDIS, T. *Structural Pattern Recognition.* New York: Springer-Verlag, 1977.

PERSOON, E. and K. S. FU. "Shape discrimination using Fourier descriptors." *Proc.*, 2nd IJCPR, August 1974, 126–130.

REQUICHA, A. A. G. "Mathematical models of rigid solid objects." TM-28, Production Automation Project, Univ. Rochester, November 1977.

ROBERTS, L. G. "Machine perception of three-dimensional solids." In *Optical and Electro-optical Information Processing*, J.P. Tippett et al. (Eds.). Cambridge, MA: MIT Press, 1965.

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

SAMET, H. "Region representation: quadtrees from boundary codes." *Comm. ACM 23*, 3, March 1980, 163–170.

SCHNEIER, M. "Linear time calculations of geometric properties using quadtrees." TR-770, Computer Science Center, Univ. Maryland, May 1979.

SHIRAI, Y. "Analyzing intensity arrays using knowledge about scenes." In *PCV*, 1975.

SKLANSKY, J. "Measuring concavity on a rectangular mosaic." *IEEE Trans. Computers 21*, 12, December 1972.

SKLANSKY, J. and D. P. KIBLER. "A theory of non-uniformly digitizing binary pictures." *IEEE Trans. SMC 6*, 9, September 1976, 637–647.

TOMEK, I. "Two algorithms for piecewise linear continuous approximation of functions of one variable." *IEEE Trans. Computers 23*, 4, April 1974, 445–448.

TURNER, K. J. "Computer perception of curved objects using a television camera." Ph.D. dissertation, Univ. Edinburgh, 1974.

VOELCKER, H. B. and A. A. G. REQUICHA. "Geometric modelling of mechanical parts and processes." *Computer 10*, December 1977, 48–57.

WU, S., J. F. ABEL, and D. P. GREENBERG. "An interactive computer graphics approach to surface representations." *Comm. ACM 20*, 10, October 1977, 703–711.

YOUNG, I. T., J. E. WALKER, and J. E. BOWIE. "An analysis technique for biological shape I." *Information and Control 25*, 1974.