Contents

List of Algorithms

Chapter 6

Shape representation and description

The last chapter was devoted to image segmentation methods which showed how to construct homogeneous regions of images and/or their boundaries. Recognition of image regions is an important step on the way to understanding image data, and requires an exact region description in a form suitable for a classifier (Chapter 7). This description should generate a numeric feature vector, or a non-numeric syntactic description word, which characterizes properties (for example, shape) of the region. Region description is the third of the four levels given in Chapter 3, implying that the description already comprises some abstraction – for example, 3D objects can be represented in a 2D plane and shape properties that are used for description are usually computed in two dimensions. If we are interested in a 3D object description, we have to process at least two images of the same object taken from different viewpoints (stereo vision), or derive the 3D shape from a sequence of images if the object is in motion. A 2D shape representation is sufficient in the majority of practical applications, but if 3D information is necessary – if, say, 3D object reconstruction is the processing goal, or the 3D characteristics bear the important information – the object description task is much more difficult; these topics are introduced in Chapter 9. In the following sections, we will limit our discussion to 2D shape features and proceed under the assumption that object descriptions result from the image segmentation process.

Defining the shape of an object can prove to be very difficult. Shape is usually represented verbally or in figures and people use terms like *elongated*, *rounded*, *with sharp edges*, etc. The computer era has introduced the necessity to describe even very complicated shapes precisely, and while many practical shape description methods exist, there is no generally accepted methodology of shape description. Further, it is not known what in shape is important. Current approaches have both positive and negative attributes; computer graphics [Woodwark 86] or mathematics [Lord and Wilson 84] use effective shape representations which are unusable in shape recognition [Juday 88] and vice versa. In spite of this, it is possible to find features common to most shape description approaches. Location and description of substantial variations in the first derivative of object boundaries often yield suitable information. Examples include alphanumeric optical character description (OCR), technical drawings, ECG curve characterization, etc.

Shape is an object property which has been carefully investigated in recent years and

many papers may be found dealing with numerous applications – OCR, ECG analysis, EEG analysis, cell classification, chromosome recognition, automatic inspection, technical diagnostics, etc. Despite this variety, differences of many approaches are limited mostly to terminology. These common methods can be characterized from different points of view [Pavlidis 78, Pavlidis 80, Ballard and Brown 82, Brady 84, Besl 88, Marshall 89b, Koenderink 90, Watt 93, Hogg 93].

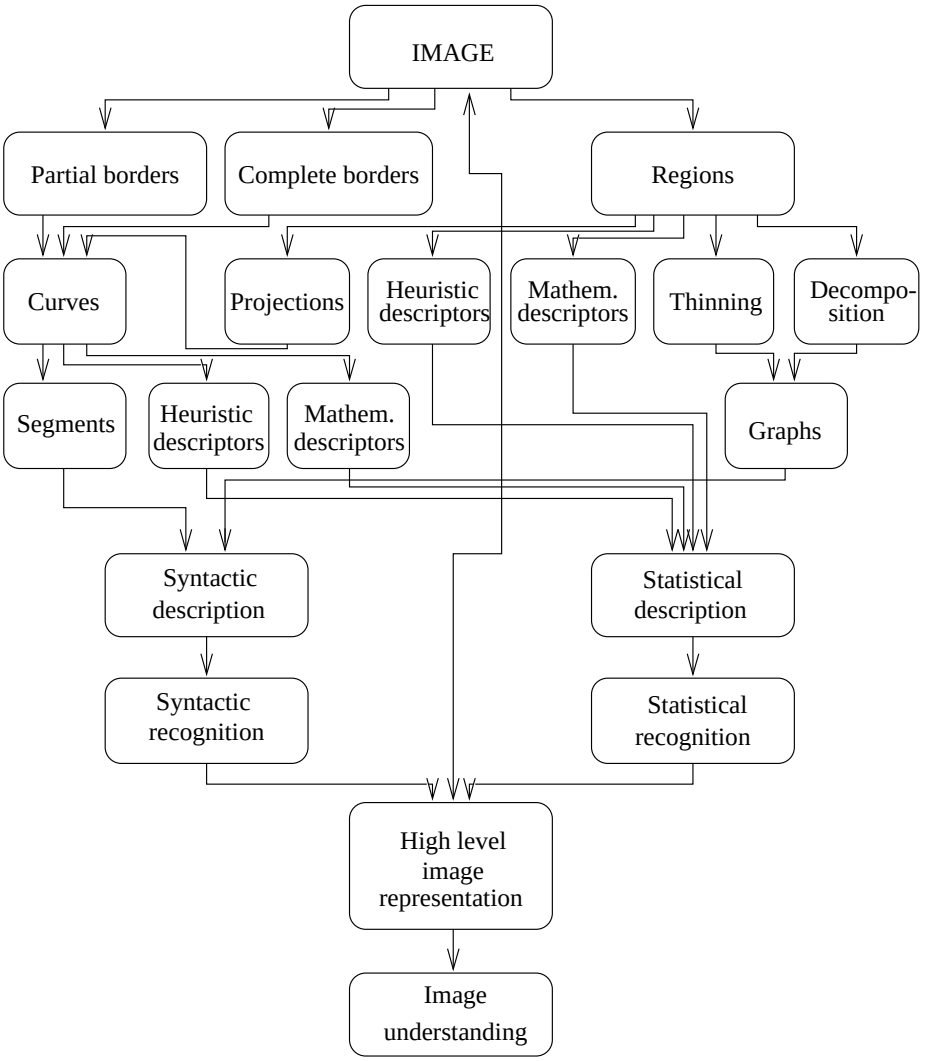


Figure 6.1: *Image analysis and understanding methods.*

- Input representation form: Object description can be based on boundaries (contour-based, external) or on more complex knowledge of whole regions (region-based, internal).
- Object reconstruction ability: That is, whether an object's shape can or cannot be reconstructed from the description. Many varieties of shape-preserving methods exist. They differ in the degree of precision with respect to object reconstruction.

- Incomplete shape recognition ability: That is, to what extent an object's shape can be recognized from the description if objects are occluded and only partial shape information is available.

- Local/global description character: Global descriptors can only be used if complete object data are available for analysis. Local descriptors describe local object properties using partial information about the objects. Thus, local descriptors can be used for description of occluded objects.

- Mathematical and heuristic techniques: A typical mathematical technique is shape description based on the Fourier transform. A representative heuristic method may be elongatedness.

- Statistical or syntactic object description (Chapter 7).

- A robustness of description to translation, rotation, and scale transformations: Shape description properties in different resolutions.

The role of different description methods in image analysis and image understanding is illustrated by the flowchart shown in Figure 6.1.

Problems of scale (resolution) are common in digital images. Sensitivity to scale is even more serious if a shape description is derived, because shape may change substantially with image resolution. Contour detection may be affected by noise in high resolution, and small details may disappear in low resolution (see Figure 6.2). Therefore, shape has been studied in multiple resolutions which again causes difficulties with matching corresponding shape representations from different resolutions. Moreover, the conventional shape descriptions change discontinuously. A **scale-space** approach has been presented in [Babaud et al. 86, Witkin 86, Yuille and Poggio 86, Maragos 89] that aims to obtain continuous shape descriptions if the resolution changes continuously. This approach is not a new technique itself, but is an extension of existing techniques, and more robust shape methods may result from developing and retaining their parameters over a range of scales [Marshall 89b]. This approach will be mentioned in more detail in Section 6.2.4.

In many tasks, it is important to represent classes of shapes properly, e.g. shape classes of apples, oranges, pears, bananas, etc. The **shape classes** should represent the generic shapes of the objects belonging to the same classes well. Obviously, shape classes should emphasize shape differences among classes while the influence of shape variations within classes should not be reflected in the class description. Current research challenges includes development of approaches to automated learning about shape and reliable definition of shape classes (Section 6.4).

Object representation and shape description methods discussed in the following sections are not an exhaustive list – we will try to introduce generally applicable methods. It is necessary to apply a problem-oriented approach to the solution of specific problems of description and recognition. This means that the following methods are appropriate for a large variety of descriptive tasks and the following ideas may be used to build a specialized, highly efficient method suitable for a particular problem description. Such a method will no longer be general since it will take advantage of a priori knowledge about the problem. This is the way human beings can solve their vision and recognition problems, by using highly specialized knowledge.

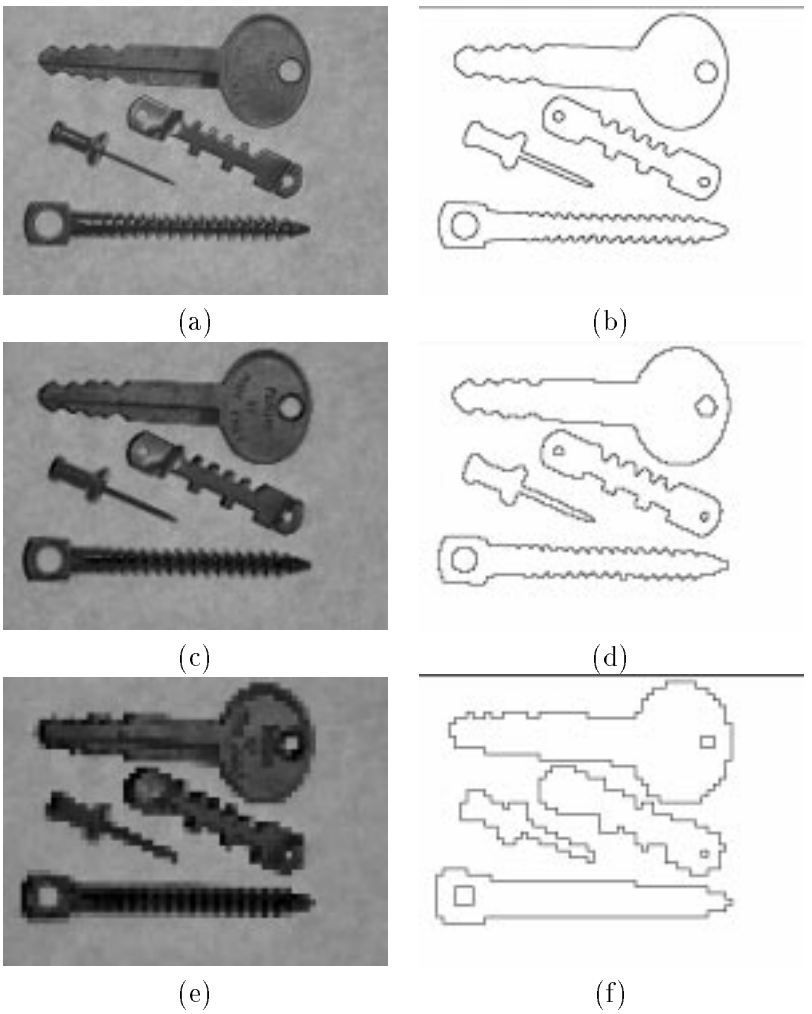


Figure 6.2: *(a) Original image 640 × 480, (b) contours of a, (c) original image 160 × 120 (d) contours of c, (e) original image 64 × 48, (f) contours of e.*

It should be understood that despite the fact that we are dealing with two-dimensional shape and its description, our world is three-dimensional and the same objects, if seen from different angles (or changing position/orientation in space), may form very different 2D projections (see Chapter 9). The ideal case would be to have a universal shape descriptor capable of overcoming these changes – to design projection-invariant descriptors. Consider an object with planar faces and imagine how many very different 2D shapes may result from a given face if the position and 3D orientation of this simple object changes with respect to an observer. In some special cases, like circles which transform to ellipses, or planar polygons, projectively invariant features (called **invariants**) can be found. Unfortunately, no existing shape descriptor is perfect; in fact, they are all far from being perfect. Therefore, a very careful choice of descriptors resulting from detailed analysis of the shape recognition problem must precede any implementation, and whether or not a 2D representation is capable of de-

scribing a 3D shape must also be considered. For some 3D shapes, their 2D projection may bear enough information for recognition – aircraft contours are a good example; successful recognition of airplanes from projections are known even if they change their position and orientation in space. In many other cases, objects must be seen from a specific direction to get enough descriptive information – human faces are such a case.

Object occlusion is another hard problem in shape recognition. However, the situation is easier here (if pure occlusion is considered, not combined with orientation variations yielding changes in 2D projections as discussed above), since visible parts of objects may be used for description. Here, the shape descriptor choice must be based on its ability to describe local object properties – if the descriptor only gives a global object description (e.g. object size, average boundary curvature, perimeter), such a description is useless if only a part of an object is visible. If a local descriptor is applied (e.g. description of local boundary changes), this information may be used to compare the visible part of the object to all objects which may appear in the image. Clearly, if object occlusion occurs, the local or global character of the shape descriptor must be considered first.

In Sections 6.2 and 6.3, descriptors are sorted according to whether they are based on object boundary information (contour-based, external description) or whether the information from object regions is used (region-based, internal description) [Li and Ma 94]. This classification of shape description methods corresponds to previously described boundary-based and region-based segmentation methods. However, both contour-based and region-based shape descriptors may be local or global and differ in sensitivity to translation, rotation, scaling, etc.

6.1 Region identification

Region identification is necessary for region description. One of the many methods for region identification is to label each region (or each boundary) with a unique (integer) number; such identification is called **labeling** or **coloring** (also connected component labeling), and the largest integer label usually gives the number of regions in the image. Another method is to use a smaller number of labels (four is theoretically sufficient [Appel and Haken 77, Saaty and Kainen 77, Nishizeki and Chiba 88, Wilson and Nelson 90]), and ensure that no two neighboring regions have the same label; then information about some region pixel must be added to the description to provide full region reference. This information is usually stored in a separate data structure. Alternatively, mathematical morphology approaches (Chapter 11) may be used for region identification.

Assume that the segmented image R consists of m disjoint regions R_i (as in equation (5.1)). The image R often consists of objects and a background

$$R_b^C = \bigcup_{i=1, i\neq b}^{m} R_i$$

where R^C is the set complement, R_b is considered background, and other regions are considered objects. Input to a labeling algorithm is usually either a binary or multi-level image, where background may be represented by zero pixels, and objects by non-zero values. A multi-level image is often used to represent the labeling result, background being represented

by zero values, and regions represented by their non-zero labels. Algorithm 6.1 presents a sequential approach to labeling a segmented image.

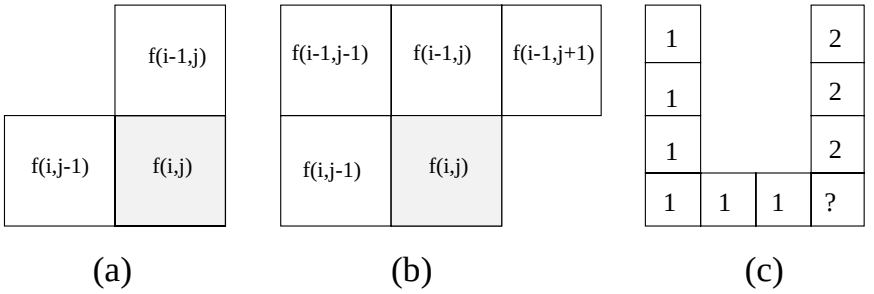


Figure 6.3: *Masks for region identification: (a) In 4-connectivity, (b) in 8-connectivity, (c) label collision.*

Algorithm 6.1: 4-neighborhood and 8-neighborhood region identification

1. First pass: Search the entire image R row by row and assign a non-zero value v to each non-zero pixel $R(i,j)$. The value v is chosen according to the labels of the pixel's neighbors where the property *neighboring* is defined by Figure 6.3. ('neighbors' outside the image R are not considered),
   - If all the neighbors are background pixels (with pixel value zero), $R(i,j)$ is assigned a new (and as yet) unused label.
   - If there is just one neighboring pixel with a non-zero label, assign this label to the pixel $R(i,j)$.
   - If there is more than one non-zero pixel among the neighbors, assign the label of any one to the labeled pixel. If the labels of any of the neighbors differ (*label collision*) store the label pair as being equivalent. Equivalence pairs are stored in a separate data structure – an equivalence table.
2. Second pass: All of the region-pixels were labeled during the first pass but some regions have pixels with different labels (due to label collisions). The whole image is scanned again, and pixels re-labeled using the equivalence table information (for example, with the lowest value in an equivalence class).

Label collision is a very common occurrence – examples of image shapes experiencing this are U-shaped objects, mirrored E ($\exists$) objects, etc. (see Figure 6.3c). The equivalence table is a list of all label pairs present in an image; all equivalent labels are replaced by a unique label in the second step. Since the number of label collisions is usually not known beforehand, it is necessary to allocate sufficient memory to store the equivalence table in an array. A dynamically allocated data structure is recommended. Further, if pointers are used for label specification, scanning the image for the second time is not necessary (the second pass of the algorithm) and only rewriting labels to which these pointers are pointing is much faster.

The algorithm is basically the same in 4-connectivity and 8-connectivity, the only difference being in the neighborhood mask shape (Figure 6.3b). It is useful to assign the region labels incrementally to permit the regions to be counted easily in the second pass. An example of partial results is given in Figure 6.4.

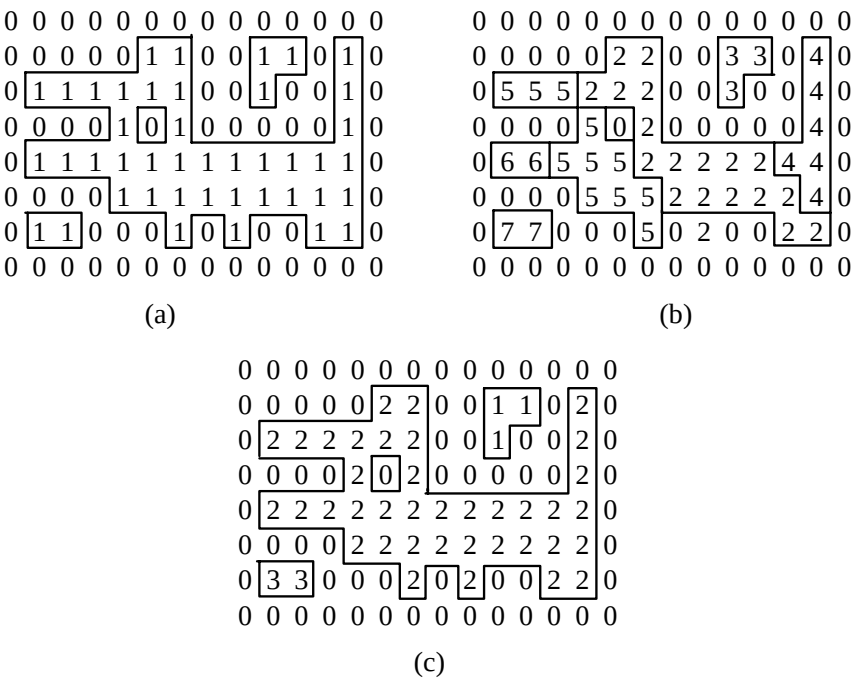


Figure 6.4: *Object identification in 8-connectivity: (a),(b),(c) Algorithm steps. Equivalence table after step (b): 2-5, 5-6, 2-4.*

Region identification can be performed on images that are not represented as straightforward matrices; the following algorithm [Rosenfeld and Kak 82] may be applied to images that are run-length encoded (see Chapter 3).

Algorithm 6.2: Region identification in run-length encoded data

1. First pass: Use a new label for each continuous run in the first image row that is not part of the background.

2. For the second and subsequent rows, compare positions of runs. If a run in a row does not neighbor (in the 4- or 8- sense) any run in the previous row, assign a new label. If a run neighbors precisely one run in the previous row, assign its label to the new run. If the new run neighbors more than one run in the previous row, a label collision has occurred. Collision information is stored in an equivalence table, and the new run is labeled using the label of any one of its neighbors.

3. Second pass: Search the image row by row and re-label the image according to the equivalence table information.

If the segmented image is represented by a quadtree data structure, the following algorithm may be applied:

Algorithm 6.3: Quadtree region identification

1. First pass: Search quadtree nodes in a given order – e.g. beginning from the root and in NW, NE, SW, SE directions. Whenever an unlabeled non-zero leaf node is entered, a new label is assigned to it. Then search for neighboring leaf nodes in the E and S directions (plus SE in 8-connectivity). If those leaves are non-zero and have not yet been labeled, assign the label of the node from which the search started. If the neighboring leaf node has already been labeled, store the collision information in an equivalence table.

2. Repeat step (1) until the whole tree has been searched.

3. Second pass: Re-label the leaf nodes of the quadtree according to the equivalence table.

Algorithmic details and the procedure for looking for neighboring leaf nodes can be found in [Rosenfeld and Kak 82, Samet 84].

The **region counting** task is closely related to the region identification problem. Object counting can be an intermediate result of region identification as we have seen. If it is only necessary to count regions with no need to identify them, a one-pass algorithm is sufficient [Rosenfeld and Kak 82, Atkinson et al. 85].

6.2 Contour-based shape representation and description

Region borders must be expressed in some mathematical form. The **rectangular** representation of $\mathbf{x}_n$ pixel co-ordinates as a function of the path length n is most common. Other useful representations are (see Figure 6.5);

- **polar** co-ordinates, in which border elements are represented as pairs of angle ϕ and distance r;
- **tangential** co-ordinates, which codes the tangential directions $\theta(\mathbf{x}_n)$ of curve points as a function of path length n.

6.2.1 Chain codes

Chain codes describe an object by a sequence of unit-size line segments with a given orientation (see Section 3.2.2). The first element of such a sequence must bear information about its position to permit the region to be reconstructed. The process results in a sequence of numbers (see Figure 6.6); to exploit the position invariance of chain codes the first element, which contains the position information, is omitted. This definition of the chain code is known as **Freeman's** code [Freeman 61]. Note that a chain code object description may easily be

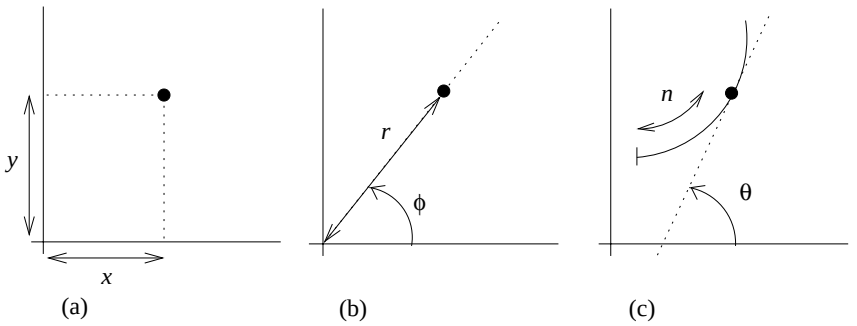


Figure 6.5: *Co-ordinate systems: (a) Rectangular (Cartesian), (b) polar, (c) tangential.*

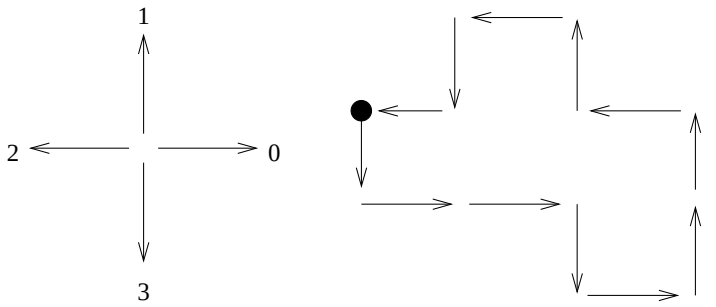


Figure 6.6: *Chain code in 4-connectivity, and its derivative. Code: 3, 0, 0, 3, 0, 1, 1, 2, 1, 2, 3, 2, derivative: 1, 0, 3, 1, 1, 0, 1, 3, 1, 1, 3, 1.*

obtained as a by-product of border detection; see Section 5.2.3 for a description of border detection algorithms.

If the chain code is used for matching it must be independent of the choice of the first border pixel in the sequence. One possibility for normalizing the chain code is to find the pixel in the border sequence which results in the minimum integer number if the description chain is interpreted as a base four number – that pixel is then used as the starting pixel [Tsai and Yu 85]. A *mod 4* or *mod 8* difference code, called a chain code **derivative**, is another numbered sequence that represents relative directions of region boundary elements, measured as multiples of counter-clockwise 90^o or 45^o direction changes (Figure 6.6). A chain code is very sensitive to noise, and arbitrary changes in scale and rotation may cause problems if used for recognition. The smoothed version of the chain code (averaged directions along a specified path length) is less noise sensitive [Li and Zhiying 88].

6.2.2 Simple geometric border representation

The following descriptors are mostly based on geometric properties of described regions. Because of the discrete character of digital images, all of them are sensitive to image resolution.

Boundary length

This is an elementary region property, simply derived from the chain code representation. Vertical and horizontal steps have unit length, and the length of diagonal steps in 8-connectivity

is $\sqrt{2}$. It can be shown that the boundary is longer in 4-connectivity where a diagonal step consists of two rectangular steps with a total length of two. A closed-boundary length (**perimeter**) can also easily be evaluated from run length [Rosenfeld and Kak 82] or quadtree representations [Samet 81, Crowley 84] as well. Boundary length increases as the image raster resolution increases; on the other hand, region area is not affected by higher resolution and converges to some limit (see also the description of fractal dimension given in Section 14.1.6). To provide continuous-space perimeter properties (area computation from the boundary length, shape features, etc.), it is better to define the region border as being the outer or extended border (see Section 5.2.3). If inner borders are used, some properties are not satisfied – e.g. the perimeter of a one-pixel region is four if the outer boundary is used, and one using the inner.

Curvature

In the continuous case, curvature is defined as the rate of change of slope. In discrete space, the curvature description must be slightly modified to overcome difficulties resulting from violation of curve smoothness. The curvature scalar descriptor (also called boundary straightness) finds the ratio between the total number of boundary pixels (length) and the number of boundary pixels where the boundary direction changes significantly. The smaller the number of direction changes, the straighter the boundary. The evaluation algorithm is based on the detection of angles between line segments positioned b boundary pixels from the evaluated boundary pixel in both directions. The angle need not be represented numerically; rather, relative position of line segments can be used as a property. The parameter b determines sensitivity to local changes of the boundary direction (Figure 6.7). Curvature computed from the chain code can be found in [Rosenfeld 74], and the tangential border representation is also suitable for curvature computation. Values of the curvature at all boundary pixels can be represented by a histogram; relative numbers then provide information on how common specific boundary direction changes are. Histograms of boundary angles, such as the β angle in Figure 6.7, can be built in a similar way– such histograms can be used for region description. Another approach to calculating curvature from digital curves is based on convolution with the truncated Gaussian kernel [Lowe 89], and an improved version not suffering from systematic bias caused by curvature smoothing effect is given in [Hlavac et al. 94].

Bending energy

The bending energy of a border (curve) may be understood as the energy necessary to bend a rod to the desired shape, and can be computed as a sum of squares of the border curvature $c(k)$ over the border length L.

$$BE = \frac{1}{L}\sum_{k=1}^{L} c^2(k) \tag{6.1}$$

Bending energy can easily be computed from Fourier descriptors using Parseval's theorem [Oppenheim et al. 83, Papoulis 91]. To represent the border, Freeman's chain code or its smoothed version may be used [Smeulders et al. 80], see Figure 6.8. Bending energy does not permit shape reconstruction.

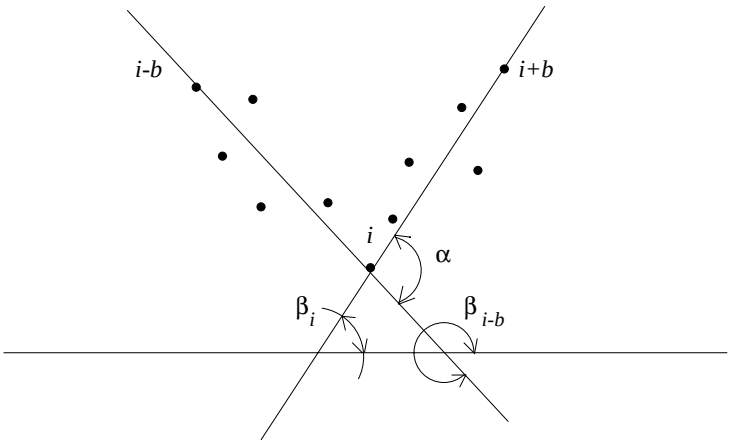


Figure 6.7: *Curvature.*

Signature

The signature of a region may be obtained as a sequence of normal contour distances. The normal contour distance is calculated for each boundary element as a function of the path length. For each border point A the shortest distance to an opposite border point B is sought in a direction perpendicular to the border tangent at point A, see Figure 6.9. Note that *being opposite* is not a symmetric relation (compare Algorithm 5.17). Signatures are noise sensitive, and using smoothed signatures or signatures of smoothed contours reduces noise sensitivity. Signatures may be applied to the recognition of overlapping objects or whenever only partial contours are available [Vernon 87]. Position, rotation, and scale-invariant modifications based on gradient-perimeter and angle-perimeter plots are discussed in [Safaee-Rad et al. 89].

Chord distribution.

A line joining any two points of the region boundary is a chord, and the distribution of lengths and angles of all chords on a contour may be used for shape description. Let $b(x,y) = 1$ represent the contour points, and $b(x,y) = 0$ represent all other points. The chord distribution can be computed (see Figure 6.10a) as

$$h(\Delta x, \Delta y) = \int\int b(x,y)b(x+\Delta x, y+\Delta y)dxdy \tag{6.2}$$

or in digital images as

$$h(\Delta x, \Delta y) = \sum_i \sum_j b(i,j)b(i+\Delta x, j+\Delta y) \tag{6.3}$$

To obtain the rotation-independent radial distribution $h_r(r)$, the integral over all angles is computed (Figure 6.10b).

$$h_r(r) = \int_{-\pi/2}^{\pi/2} h(\Delta x, \Delta y) r d\theta \tag{6.4}$$

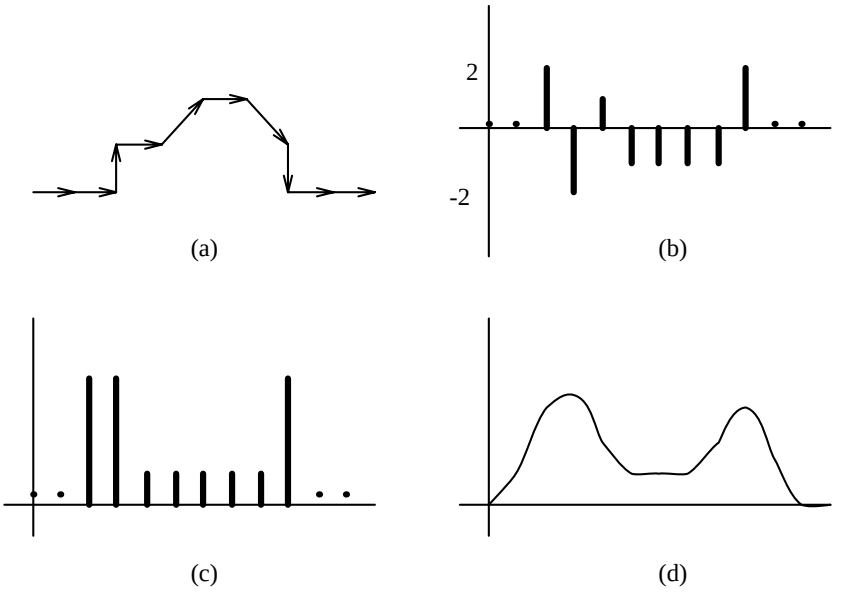


Figure 6.8: *Bending energy: (a) Chain code 0, 0, 2, 0, 1, 0, 7, 6, 0, 0, (b) curvature 0, 2, -2, 1, -1, -1, -1, 2, 0, (c) sum of squares gives the bending energy, (d) smoothed version.*

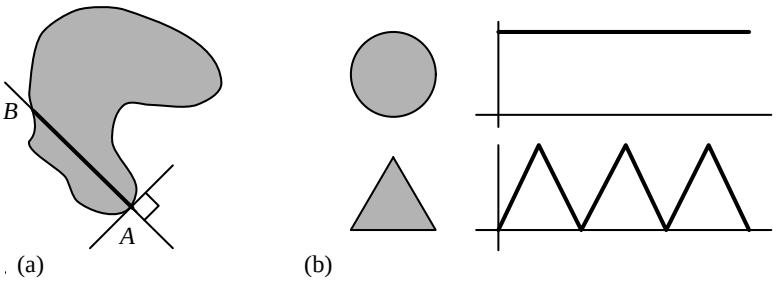


Figure 6.9: *Signature: (a) Construction, (b) signatures for a circle and a triangle.*

where $r = \sqrt{\Delta x^2 + \Delta y^2}$, $\theta = sin^{-1}(\Delta y / r)$. The distribution $h_r(r)$ varies linearly with scale. The angular distribution $h_a(\theta)$ is independent of scale, while rotation causes a proportional offset

$$h_a(\theta) = \int_0^{\max(r)} h(\Delta x, \Delta y) dr \tag{6.5}$$

Combination of both distributions gives a robust shape descriptor [Smith and Jain 82, Cootes et al. 92].

6.2.3 Fourier transforms of boundaries

Suppose C is a closed curve (boundary) in the complex plane (Figure 6.11a). Traveling anti-clockwise along this curve keeping constant speed, a complex function $z(t)$ is obtained, where t is a time variable. The speed should be chosen such that one circumnavigation of the boundary takes time 2π; then a periodic function with period 2π is obtained after multiple

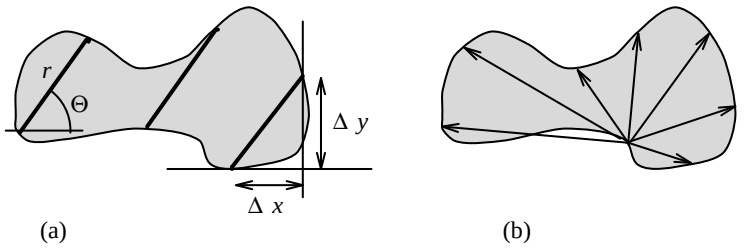


Figure 6.10: *Chord distribution.*

passes around the curve. This permits a Fourier representation of $z(t)$ (see Section 2.1.3);

$$z(t) = \sum_n T_n e^{int} \tag{6.6}$$

The coefficients T_n of the series are called the **Fourier descriptors** of the curve C. It is more useful to consider the curve distance s in comparison to time

$$t = 2\pi s/L, \tag{6.7}$$

where L is the curve length. The Fourier descriptors T_n are given by

$$T_n = \frac{1}{L}\int_0^L z(s)e^{-i(2\pi/L)ns}ds \tag{6.8}$$

The descriptors are influenced by the curve shape and by the initial point of the curve. Working with digital image data, boundary co-ordinates are discrete and the function $z(s)$ is not continuous. Assume that $z(k)$ is a discrete version of $z(s)$, where 4-connectivity is used to get a constant sampling interval; the descriptors T_n can be computed from the discrete Fourier transform (DFT, Section 12.2) of $z(k)$;

$$z(k) \longleftarrow DFT \longrightarrow T_n \tag{6.9}$$

The Fourier descriptors can be invariant to translation and rotation if the co-ordinate system is appropriately chosen [Pavlidis 77, Persoon and Fu 77, Wallace and Wintz 80, Grimmins 82, Lin and Chellappa 87]. They have been used for handwritten alphanumeric character description in [Shridhar and Badreldin 84]; the character boundary in this description was represented by co-ordinate pairs (x_m, y_m) in 4-connectivity, $(x_1, y_1) = (x_L, y_L)$. Then

$$a_n = \frac{1}{L-1}\sum_{m=1}^{L-1} x_m e^{-i(2\pi/(L-1))nm} \tag{6.10}$$

$$b_n = \frac{1}{L-1}\sum_{m=1}^{L-1} y_m e^{-i(2\pi/(L-1))nm} \tag{6.11}$$

The coefficients a_n, b_n are not invariant, but after the transform

$$r_n = (\mid a_n \mid^2 + \mid b_n \mid^2)^{1/2}, \tag{6.12}$$

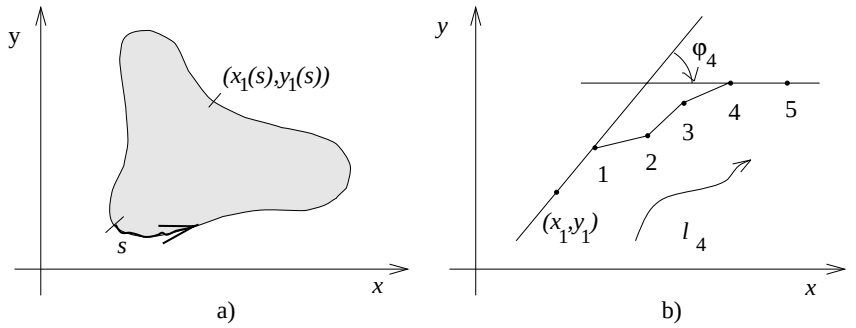


Figure 6.11: *Fourier description of boundaries: (a) Descriptors T_n, (b) descriptors S_n.*

r_n are translation and rotation invariant. To achieve a magnification invariance the descriptors w_n are used:

$$w_n = r_n / r_1 \tag{6.13}$$

The first $10 - 15$ descriptors w_n are found to be sufficient for character description.

A closed boundary can be represented as a function of angle tangents versus the distance between the boundary points from which the angles were determined (Figure 6.11b). Let φ_k be the angle measured at the k^{th} boundary point, and let l_k be the distance between the boundary starting point and the k^{th} boundary point. A periodic function can be defined

$$a(l_k) = \varphi_k + u_k, \tag{6.14}$$

$$u_k = 2\pi l_k / L \tag{6.15}$$

The descriptor set is then

$$S_n = \frac{1}{2\pi} \int_0^{2\pi} a(u) e^{-inu} du \tag{6.16}$$

The Discrete Fourier Transform is used in all practical applications [Pavlidis 77].

The high quality boundary shape representation obtained using only a few lower order coefficients is a favorable property common to Fourier descriptors. We can compare the results of using the S_n and T_n descriptors: The S_n descriptors have more high frequency components present in the boundary function due to more significant changes of tangent angles, and as a result, they do not decrease as fast as the T_n descriptors. In addition the S_n descriptors are not suitable for boundary reconstruction since they often result in a non-closed boundary. A method for obtaining a closed boundary using S_n descriptors is given in [Strackee and Nagelkerke 83]. The T_n descriptor values decrease quickly for higher frequencies and their reconstruction always results in a closed boundary. Moreover, the S_n descriptors cannot be applied for squares, equilateral triangles, etc. [Wallace 81] unless the solution methods introduced in [Wallace and Wintz 80] are applied.

Fourier descriptors can also be used for calculation of region area, location of centroid, and computation of second-order moments [Kiryati and Maydan 89]. Fourier descriptors are a general technique, but problems with describing local information exist. A modified technique using a combined frequency-position space that deals better with local curve properties is described in [Eichmann et al. 90], and another modification invariant under rotation,

translation, scale, mirror reflection, and shifts in starting points is discussed in [Krzyzak et al. 89]. Conventional Fourier descriptors cannot be used for recognition of occluded objects. Nevertheless, classification of partial shapes using Fourier descriptors is introduced in [Lin and Chellappa 87]. Boundary detection and description using elliptic Fourier decomposition of the boundary is described in [Staib and Duncan 92].

6.2.4 Boundary description using segment sequences

Representation of a boundary using **segments** with specified properties is another option for boundary (and curve) description. If the segment type is known for all segments, the boundary can be described as a chain of segment types, a code-word consisting of representatives of a type alphabet. An example is given in Figure 6.14 which will be discussed later in more detail. This sort of description is suitable for syntactic recognition (see Section 7.4). A trivial segment chain is used to obtain the Freeman code description discussed in Section 6.2.1.

A **polygonal representation** approximates a region by a polygon, the region being represented using its vertices. Polygonal representations are obtained as a result of a simple boundary segmentation. The boundary can be approximated with varying precision; if a more precise description is necessary, a larger number of line segments may be employed. Any two boundary points $\mathbf{x}_1$, $\mathbf{x}_2$ define a line segment and a sequence of points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ represents a chain of line segments – from the point $\mathbf{x}_1$ to the point $\mathbf{x}_2$, and from $\mathbf{x}_2$ to $\mathbf{x}_3$. If $\mathbf{x}_1$=$\mathbf{x}_3$, a closed boundary results. There are many types of straight segment boundary representations [Pavlidis 77, Koch and Kashyap 87, Matas and Kittler 93, Lindenbaum and Bruckstein 93, Ji and Haralick 97]; the problem lies in determining the location of boundary vertices, one solution to which is to apply a split-and-merge algorithm. The merging step consists of going through a set of boundary points and adding them to a straight segment as long as a segment straightness criterion is satisfied. If the straightness characteristic of the segment is lost, the last connected point is marked as a vertex and construction of a new straight segment begins. This general approach has many variations, some of which are described in [Pavlidis 77].

Boundary vertices can be detected as boundary points with a significant change of boundary direction using the curvature (boundary straightness) criterion (see Section 6.2.2). This approach works well for boundaries with rectilinear boundary segments.

Another method for determining the boundary vertices is a **tolerance interval approach** based on setting a maximum allowed difference e. Assume that point $\mathbf{x}_1$ is the end-point of a previous segment and so by definition the first point of a new segment. Define points $\mathbf{x}_2$, $\mathbf{x}_3$ positioned a distance e from the point $\mathbf{x}_1$ to be rectilinear – $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$ are positioned on a straight line – see Figure 6.12. The next step is to locate a segment which can fit between parallels directed from points $\mathbf{x}_2$ and $\mathbf{x}_3$. Resulting segments are sub-optimal, although optimality can be achieved with a substantial increase in computational effort [Tomek 74].

The methods introduced above represent single-pass algorithms of boundary segmentation using a segment-growing approach. Often they do not result in the best possible boundary segmentation because the vertex which is located often indicates that the real vertex should have been located a few steps back. The splitting approach of segmenting boundaries into smaller segments can sometimes help and the best results can be anticipated using a combination of both methods. If the splitting approach is used, segments are usually divided

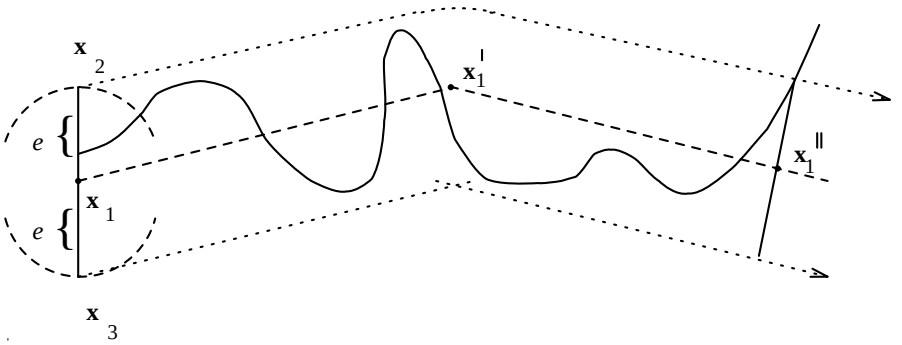


Figure 6.12: *Tolerance interval.*

into two new smaller segments until the new segments meet the final requirements [Duda and Hart 73, Pavlidis 77]. A simple procedure for splitting begins from end-points $\mathbf{x}_1$ and $\mathbf{x}_2$ of a curve; these end-points are connected by a line segment. The next step searches all the curve points for the curve point $\mathbf{x}_3$ with the largest distance from the line segment. If the point located is within a preset distance between itself and the line segment, the segment $\mathbf{x}_1$-$\mathbf{x}_2$ is an end segment and all curve vertices are found, the curve being polygonally represented by vertices $\mathbf{x}_1$ and $\mathbf{x}_2$. Otherwise the point $\mathbf{x}_3$ is set as a new vertex and the process is recursively applied to both resulting segments $\mathbf{x}_1$-$\mathbf{x}_3$ and $\mathbf{x}_3$-$\mathbf{x}_2$ (see Figure 6.13 and Section 5.2.7).

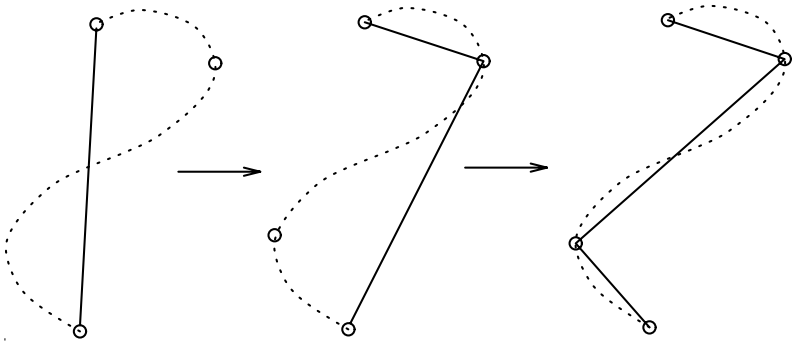


Figure 6.13: *Recursive boundary splitting.*

Boundary segmentation into segments of **constant curvature** is another possibility for boundary representation. The boundary may also be split into segments which can be represented by polynomials, usually of the second order, such as circular, elliptic, or parabolic segments [Costabile et al. 85, Wuescher and Boyer 91]. Curve segmentation into circular arcs and straight lines is presented in [Rosin and West 89]. Segments are considered as primitives for syntactic shape recognition procedures – a typical example is the syntactic description and recognition of chromosomes [Fu 74], where boundary segments are classified as convex segments of large curvature, concave segments of large curvature, straight segments, etc. as illustrated in Figure 6.14.

Other syntactic object recognition methods based on a contour partitioning into primitives from a specified set are described in [Jakubowski 85, Jakubowski 90, Tampi and Sridhar 90]. Partitioning of the contour using location of points with high positive curvatures (corners)

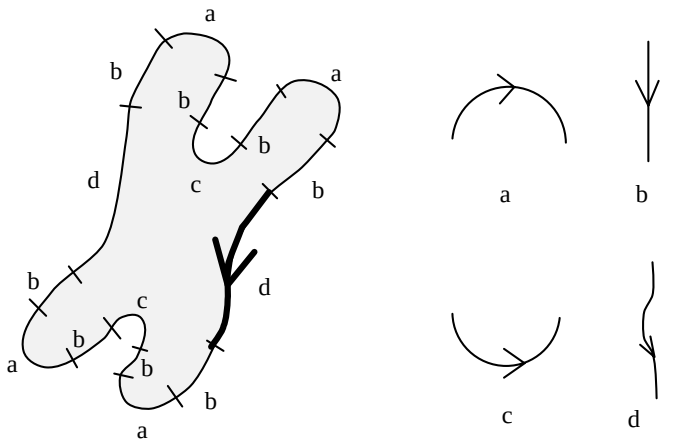


Figure 6.14: *Structural description of chromosomes by a chain of boundary segments, code word: d, b, a, b, c, b, a, b, d, b, a, b, c, b, a, b (adapted from [Fu 74]).*

is described in [Chien and Aggarwal 89] together with applications to occluded contours. A discrete curvature function based on a chain code representation of a boundary is used with a morphological approach to obtain segments of constant curvature in [Leymarie and Levine 89]. Contour partitioning using segments of constant intensity is suggested in [Marshall 89a], and polygonal representation used in a *hypothesize and verify* approach to recognition of occluded objects may be found in [Koch and Kashyap 87].

Sensitivity of shape descriptors to scale (image resolution) has already been mentioned as an undesirable feature of a majority of descriptors. In other words, shape description varies with scale, and different results are achieved at different resolutions. This problem is no less important if a curve is to be divided into segments; some curve segmentation points exist in one resolution and disappear in others without any direct correspondence. Considering this, a **scale-space** approach to curve segmentation that guarantees a continuously changing position of segmentation points is a significant achievement [Babaud et al. 86, Witkin 86, Yuille and Poggio 86, Maragos 89, Florack et al. 92, Griffin et al. 92]. In this approach, only new segmentation points can appear at higher resolutions, and no existing segmentation points can disappear. This is in agreement with our understanding of varying resolutions; finer details can be detected in higher resolution but significant details should not disappear if the resolution increases. This technique is based on application of a unique Gaussian smoothing kernel to a one-dimensional signal (e.g. a curvature function) over a range of sizes and the result is differentiated twice. To determine the peaks of curvature, the zero crossing of the second derivative is detected, the positions of zero crossings give the position of curve segmentation points. Different locations of segmentation points are obtained at varying resolution (different Gaussian kernel size). An important property of the Gaussian kernel is that the location of segmentation points changes continuously with resolution which can be seen in the **scale-space image** of the curve, Figure 6.15a. Fine details of the curve disappear in pairs with increasing size of the Gaussian smoothing kernel, and two segmentation points always merge to form a closed contour showing that any segmentation point existing in coarse resolution must also exist in finer resolution. Moreover, the position of a segmentation point is most accurate in finest resolution and this position can be traced from coarse to fine

resolution using the scale-space image. A multiscale curve description can be represented by an **interval tree**, Figure 6.15b. Each pair of zero crossings is represented by a rectangle, its position corresponding with segmentation point locations on the curve, its height showing the lowest resolution at which the segmentation point can be detected. Interval trees can be used for curve decomposition in different scales keeping the possibility of segment description using higher resolution features.

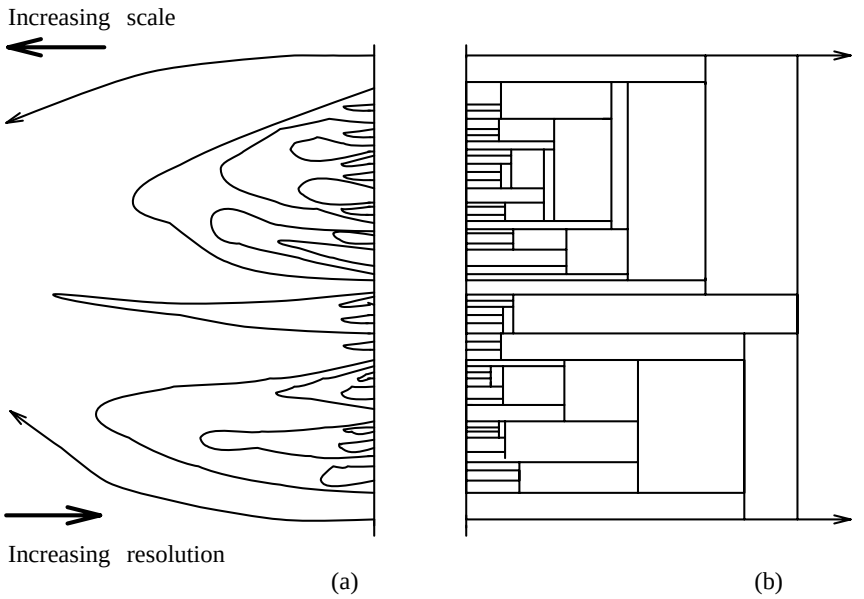


Figure 6.15: *Scale-space image: (a) Varying number and locations of curve segmentation points as a function of scale, (b) curve representation by an interval tree.*

Another scale-space approach to curve decomposition is the **curvature primal sketch** [Asada and Brady 86], (compare Section 9.1.1). A set of primitive curvature discontinuities is defined and convolved with first and second derivatives of a Gaussian in multiple resolutions. The curvature primal sketch is computed by matching the multiscale convolutions of a shape. The curvature primal sketch then serves as a shape representation; shape reconstruction may be based on polygons or splines. Another multiscale border-primitive detection technique that aggregates curve primitives at one scale into curve primitives at a coarser scale is described in [Saund 90]. A robust approach to multiscale curve corner detection that uses additional information extracted from corner behavior in the whole multi-resolution pyramid is given in [Fermuller and Kropatsch 92].

6.2.5 B-spline representation

Representation of curves using piecewise polynomial interpolation to obtain smooth curves is widely used in computer graphics. B-splines are piecewise polynomial curves whose shape is closely related to their control polygon – a chain of vertices giving a polygonal representation of a curve. B-splines of the third-order are most common because this is the lowest order which includes the change of curvature. Splines have very good representation properties and are easy to compute: Firstly, they change their shape less then their control polygon, and

do not oscillate between sampling points as many other representations do. Furthermore, a spline curve is always positioned inside a convex $n+1$-polygon for a B-spline of the n^{th} order – Figure 6.16. Secondly, the interpolation is local in character. If a control polygon vertex changes its position, a resulting change of the spline curve will occur only in a small neighborhood of that vertex. Thirdly, methods of matching region boundaries represented by splines to image data are based on a direct search of original image data. These methods are similar to the segmentation methods described in Section 5.2.6. A spline direction can be derived directly from its parameters.

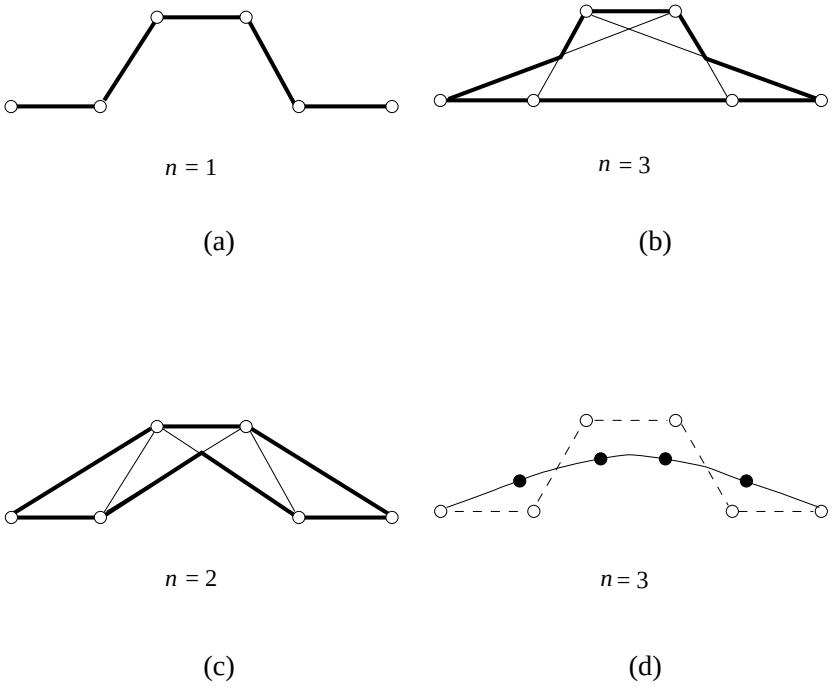


Figure 6.16: *Splines of order n. (a,b,c) Convex $n+1$-polygon for a B-spline of the n^{th} order, (d) 3^{rd} order spline.*

Let $\mathbf{x}_i$, $i = 1, \ldots, n$ be points of a B-spline interpolation curve; call this interpolation curve $\mathbf{x}(s)$. The s parameter changes linearly between points $\mathbf{x}_i$ – that is, $\mathbf{x}_i = \mathbf{x}(i)$. Each part of a cubic B-spline curve is a third-order polynomial, meaning that it and its first and second derivatives are continuous. B-splines are given by

$$\mathbf{x}(s) = \sum_{i=0}^{n+1} \mathbf{v}_i B_i(s), \tag{6.17}$$

where $\mathbf{v}_i$ are coefficients representing a spline curve, and $B_i(s)$ are base functions whose shape is given by the spline order. The coefficients $\mathbf{v}_i$ bear information dual to information about the spline curve points $\mathbf{x}_i$ – the values $\mathbf{v}_i$ can be derived from $\mathbf{x}_i$ values and vice versa. The coefficients $\mathbf{v}_i$ represent vertices of the control polygon and if there are n points $\mathbf{x}_i$, there must be $n+2$ points $\mathbf{v}_i$. The two end-points $\mathbf{v}_0$, $\mathbf{v}_{n+1}$ are specified by binding conditions. If

the curvature of a B-spline curvature is to be zero at the curve beginning and end, then

$$\mathbf{v}_0 = 2\mathbf{v}_1 - \mathbf{v}_2$$

$$\mathbf{v}_{n+1} = 2\mathbf{v}_n - \mathbf{v}_{n-1} \tag{6.18}$$

If the curve is closed then $\mathbf{v}_0 = \mathbf{v}_n$ and $\mathbf{v}_{n+1} = \mathbf{v}_1$.

The base functions are non-negative and are of local importance only. Each base function $B_i(s)$ is non-zero only for $s \in (i-2, i+2)$ meaning that for any $s \in (i, i+1)$, there are only four non-zero base functions for any i : $B_{i-1}(s)$, $B_i(s)$, $B_{i+1}(s)$, and $B_{i+2}(s)$. If the distance between the $\mathbf{x}_i$ points is constant (e.g. unit distances), all the base functions are of the same form and consist of four parts $C_j(t)$, $j = 0, \ldots, 3$.

$$C_0(t) = \frac{t^3}{6}$$

$$C_1(t) = \frac{-3t^3 + 3t^2 + 3t + 1}{6} \tag{6.19}$$

$$C_2(t) = \frac{3t^3 - 6t^2 + 4}{6}$$

$$C_3(t) = \frac{-t^3 + 3t^2 - 3t + 1}{6}$$

Because of equation (6.17) and zero-equal base functions for $s \notin (i-2, i+2)$, $\mathbf{x}(s)$ can be computed from the addition of only four terms for any s.

$$\mathbf{x}(s) = C_{i-1,3}(s)\mathbf{v}_{i-1} + C_{i,2}(s)\mathbf{v}_i + C_{i+1,1}(s)\mathbf{v}_{i+1} + C_{i+2,0}(s)\mathbf{v}_{i+2} \tag{6.20}$$

Here, $C_{i,j}(s)$ means that we use the j^{th} part of the base function B_i (see Figure 6.17). Note that

$$C_{i,j}(s) = C_j(s-i), \tag{6.21}$$

$$i = 0, \ldots, n+1; \qquad j = 0, 1, 2, 3.$$

To work with values inside the interval $[i, i+1)$, the interpolation curve $\mathbf{x}(s)$ can be computed as

$$\mathbf{x}(s) = C_3(s-i)\mathbf{v}_{i-1} + C_2(s-i)\mathbf{v}_i + C_1(s-i)\mathbf{v}_{i+1} + C_0\mathbf{v}_{i+2} \tag{6.22}$$

Specifically if $s = 5$, s is positioned at the beginning of the interval $[i, i+1)$, therefore $i = 5$ and

$$\mathbf{x}(5) = C_3(0)\mathbf{v}_4 + C_2(0)\mathbf{v}_5 + C_1(0)\mathbf{v}_6 = \frac{1}{6}\mathbf{v}_4 + \frac{4}{6}\mathbf{v}_5 + \frac{1}{6}\mathbf{v}_6 \tag{6.23}$$

or if $s = 7.7$ then $i = 7$ and

$$\mathbf{x}(5) = C_3(0.7)\mathbf{v}_6 + C_2(0.7)\mathbf{v}_7 + C_1(0.7)\mathbf{v}_8 + C_0(0.7)\mathbf{v}_9 \tag{6.24}$$

Other useful formulae can be found in [DeBoor 78, Ballard and Brown 82, Ikebe and Miyamoto 82].

Splines generate curves which are usually considered pleasing. They allow a good curve approximation, and can easily be used for image analysis curve representation problems.

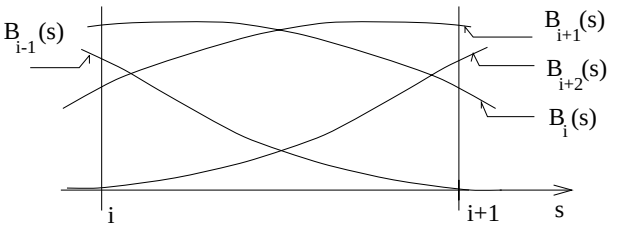


Figure 6.17: *The only four non-zero base functions for* $s \in (i, i+1)$.

A technique transforming curve samples to B-spline control polygon vertices is described in [Paglieroni and Jain 88] together with a method of efficient computation of boundary curvature, shape moments, and projections from control polygon vertices. Splines differ in their complexity; one of the simplest applies the B-spline formula for curve modeling as well as for curve extraction from image data [DeBoor 78]. Splines are used in computer vision to form exact and flexible inner model representations of complex shapes which are necessary in model-driven segmentation and in complex image understanding tasks. On the other hand, splines are highly sensitive to change in scale.

6.2.6 Other contour-based shape description approaches

Many other methods and approaches can be used to describe two-dimensional curves and contours.

The **Hough transform** has excellent shape description abilities and is discussed in detail in the image segmentation context in Section 5.2.6 (see also [McKenzie and Protheroe 90]). Region-based shape description using **statistical moments** is covered in Section 6.3.2 where a technique of contour-based moments computation from region borders is also included. Further, it is necessary to mention the **fractal** approach to shape [Mandelbrot 82, Barnsley 88, Falconer 90] that is gaining growing attention in image shape description [Frisch et al. 87, Chang and Chatterjee 89, Vemuri and Radisavljevic 93, Taylor and Lewis 94].

Mathematical morphology can be used for shape description, typically in connection with region skeleton construction (see Section 6.3.4) [Reinhardt and Higgins 96]. A different approach is introduced in [Loui et al. 90] where a **geometrical correlation function** represents two-dimensional continuous or discrete curves. This function is translation, rotation, and scale invariant and may be used to compute basic geometrical properties.

Neural networks (Section 7.3) can be used to recognize shapes in raw boundary representations directly. Contour sequences of noiseless reference shapes are used for training, and noisy data are used in later training stages to increase robustness; effective representations of closed planar shapes result [Gupta et al. 90]. Another neural network shape representation system uses a modified Walsh-Hadamard transform (Chapter 12) to achieve position-invariant shape representation [Minnix et al. 90].

6.2.7 Shape invariants

Shape invariants represent a very active current research area in machine vision. Although the importance of shape invariance has been known for a long time, the first machine-vision

related paper about shape invariants [Weiss 88] appeared in 1988 followed by a book [Kanatani 90] in 1990. The following section gives a brief overview of this topic and is mostly based on the paper [Forsyth et al. 91] and on the book [Mundy and Zisserman 92] in which additional details can be found. The book [Mundy and Zisserman 92] gives an overview of this topic in its Introduction and its Appendix presents an excellent and detailed survey of projective geometry for machine vision. Even if shape invariance is a novel approach in machine vision, invariant theory is not new and many of its principles were introduced in the nineteenth century.

As has been mentioned many times, object description is necessary for object recognition. Unfortunately, all the shape descriptors discussed so far depend on viewpoint, meaning that object recognition may often be impossible as a result of changed object or observer position, as illustrated in Figure 6.18. The role of shape description invariance is obvious – shape invariants represent properties of such geometric configurations which remain unchanged under an appropriate class of transforms [Mundy and Zisserman 92, Reiss 93]. Machine vision is especially concerned with the class of projective transforms.

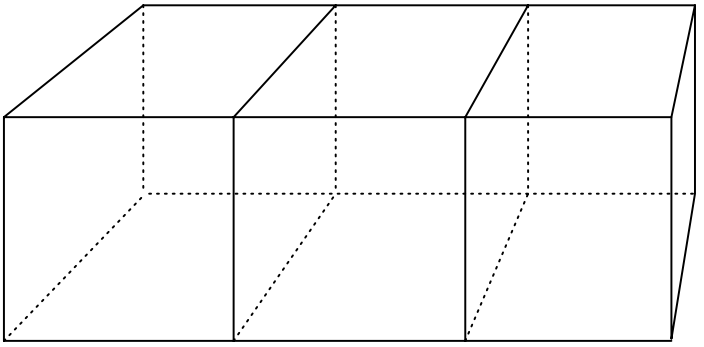


Figure 6.18: *Change of shape caused by a projective transform. The same rectangular cross-section is represented by different polygons in the image plane.*

Collinearity is the simplest example of a projectively invariant image feature. Any straight line is projected as a straight line under any projective transform. Similarly, the basic idea of the projection-invariant shape description is to find such shape features that are unaffected by the transform between the object and the image plane.

A standard technique of projection-invariant description is to hypothesize the pose (position and orientation) of an object and transform this object into a specific co-ordinate system; then shape characteristics measured in this co-ordinate system yield an invariant description. However, the pose must be hypothesized for each object and each image which makes this approach difficult and unreliable.

Application of **invariant theory**, where invariant descriptors can be computed directly from image data without the need for a particular co-ordinate system, represents another approach. In addition, invariant theory can determine the total number of functionally independent invariants for a given situation therefore showing completeness of the description invariant set. Invariant theory is based on a collection of transforms that can be composed and inverted. In vision, the **plane-projective group** of transforms is considered which contains all the perspectives as a subset. The **group approach** provides a mathematical tool

for generating invariants; if the transform does not satisfy the properties of a group, this machinery is not available [Mundy and Zisserman 92]. Therefore, the change of co-ordinates due to the plane-projective transform is generalized as a **group action**. **Lie group** theory is especially useful in designing new invariants.

Let corresponding entities in two different co-ordinate systems be distinguished by large and small letters. An invariant of a linear transformation is defined in [Mundy and Zisserman 92] as follows:

> An invariant, $I(\mathbf{P})$, of a geometric structure described by a parameter vector $\mathbf{P}$, subject to a linear transformation $\mathbf{T}$ of the co-ordinates $\mathbf{x} = \mathbf{TX}$, is transformed according to $I(\mathbf{p}) = I(\mathbf{P})|\mathbf{T}|^w$. Here $I(\mathbf{p})$ is the function of the parameters after the linear transformation, and $|\mathbf{T}|$ is the determinant of the matrix $\mathbf{T}$.

In this definition, w is referred to as the weight of the invariant. If $w = 0$, the invariants are called **scalar invariants**, which are considered below. Invariant descriptors are unaffected by object pose, by perspective projection, and by the intrinsic parameters of the camera.

Several examples of invariants are now given.

1. **Cross ratio**: The cross ratio represents a classic invariant of a projective line. As mentioned earlier, a straight line is always projected as a straight line. Any four collinear points A, B, C, D may be described by the cross-ratio invariant

$$I = \frac{(A-C)(B-D)}{(A-D)(B-C)} \tag{6.25}$$

   where $(A-C)$ represents the distance between points A and C (see Figure 6.19). Note that the cross ratio depends on the order in which the four collinear points are labeled.

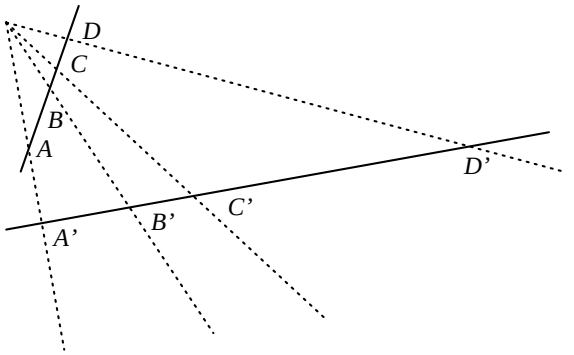


Figure 6.19: *Cross ratio; four collinear points form a projective invariant.*

2. **Systems of lines or points**: A system of four coplanar concurrent lines (meeting at the same point) is dual to a system of four collinear points and the cross ratio is its invariant, see Figure 6.19.

   A system of five general coplanar lines forms two invariants

$$I_1 = \frac{|\mathbf{M}_{431}||\mathbf{M}_{521}|}{|\mathbf{M}_{421}||\mathbf{M}_{531}|} \qquad I_2 = \frac{|\mathbf{M}_{421}||\mathbf{M}_{532}|}{|\mathbf{M}_{432}||\mathbf{M}_{521}|} \tag{6.26}$$

where $\mathbf{M}_{ijk} = (\mathbf{l}_i, \mathbf{l}_j, \mathbf{l}_k)$. $\mathbf{l}_i = (l_i^1, l_i^2, l_i^3)^T$ is a representation of a line $l_i^1 x + l_i^2 y + l_i^3 = 0$, where $i \in [1,5]$, and $|\mathbf{M}|$ is the determinant of $\mathbf{M}$. If the three lines forming the matrix $\mathbf{M}_{ijk}$ are concurrent, the matrix becomes singular and the invariant is undefined.

A system of five coplanar points is dual to a system of five lines and the same two invariants are formed. These two functional invariants can also be formed as two cross ratios of two coplanar concurrent line quadruples, see Figure 6.20. Note that even though combinations other than those given in Figure 6.20 may be formed, only the two presented functionally independent invariants exist.

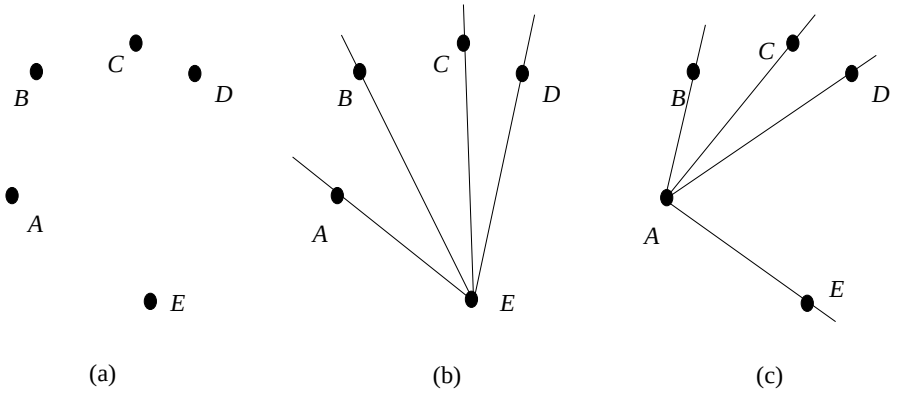


Figure 6.20: *Five coplanar points form two cross-ratio invariants: (a) Coplanar points, (b) five points form a system of four concurrent lines, (c) the same five points form another system of four coplanar lines.*

3. **Plane conics**: A plane conic may be represented by an equation

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{6.27}$$

for $\mathbf{x} = (x, y, 1)^T$. Then the conic may also be defined by a matrix $\mathbf{C}$

$$\mathbf{C} = \begin{vmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{vmatrix}$$

and

$$\mathbf{x}^T \ \mathbf{C} \ \mathbf{x} = 0 \tag{6.28}$$

For any conic represented by a matrix $\mathbf{C}$, and any two coplanar lines not tangent to the conic, one invariant may be defined

$$I = \frac{(\mathbf{l}_1^T \mathbf{C}^{-1} \mathbf{l}_2)^2}{(\mathbf{l}_1^T \mathbf{C}^{-1} \mathbf{l}_1)(\mathbf{l}_2^T \mathbf{C}^{-1} \mathbf{l}_2)} \tag{6.29}$$

The same invariant can be formed for a conic and two coplanar points.

Two invariants can be determined for a pair of conics represented by their respective matrices $\mathbf{C}_1, \mathbf{C}_2$ normalized so that $|\mathbf{C}_i| = 1$

$$I_1 = \text{Trace}[\mathbf{C}_1^{-1}\mathbf{C}_2] \qquad I_2 = \text{Trace}[\mathbf{C}_2^{-1}\mathbf{C}_1] \tag{6.30}$$

(The trace of a matrix is calculated as the sum of elements on the main diagonal.) For non-normalized conics, the invariants of associated quadratic forms are

$$I_1 = \mathrm{Trace}[\mathbf{C}_1^{-1}\mathbf{C}_2]\left(\frac{|\mathbf{C}_1|}{|\mathbf{C}_2|}\right)^{\frac{1}{3}} \qquad I_2 = \mathrm{Trace}[\mathbf{C}_2^{-1}\mathbf{C}_1]\left(\frac{|\mathbf{C}_2|}{|\mathbf{C}_1|}\right)^{\frac{1}{3}} \tag{6.31}$$

and two true invariants of the conics are [Quan et al. 92]

$$I_1 = \frac{\mathrm{Trace}[\mathbf{C}_1^{-1}\mathbf{C}_2]}{(\mathrm{Trace}[\mathbf{C}_2^{-1}\mathbf{C}_1])^2}\frac{|\mathbf{C}_1|}{|\mathbf{C}_2|} \qquad I_2 = \frac{\mathrm{Trace}[\mathbf{C}_2^{-1}\mathbf{C}_1]}{(\mathrm{Trace}[\mathbf{C}_1^{-1}\mathbf{C}_2])^2}\frac{|\mathbf{C}_2|}{|\mathbf{C}_1|} \tag{6.32}$$

An interpretation of these invariants is given in [Maybank 92]. Two plane conics uniquely determine four points of intersection, and any point that is not an intersection point may be chosen to form a five-point system together with the four intersection points. Therefore, two invariants exist for the pair of conics, as for the five-point system.

Many man-made objects consist of a combination of straight lines and conics, and these invariants may be used for their description. However, if the object has a contour which cannot be represented by an algebraic curve, the situation is much more difficult. **Differential invariants** can be formed (e.g. curvature, torsion, Gaussian curvature) which are not affected by projective transforms. These invariants are local – that is, the invariants are found for each point on the curve, which may be quite general. Unfortunately, these invariants are extremely large and complex polynomials, requiring up to seventh derivatives of the curve, which makes them practically unusable due to image noise and acquisition errors, although noise-resistant local invariants are beginning to appear [Weiss 92]. However, if additional information is available, higher derivatives may be avoided. In [Brill et al. 92, Van Gool et al. 92], higher derivatives are traded for extra reference points which can be detected on curves in different projections although the necessity of matching reference points in different projections brings other difficulties.

Designing new invariants is an important part of invariant theory in its application to machine vision. The easiest way is to combine primitive invariants forming new ones from these combinations. Nevertheless, no new information is obtained from these combinations. Further, complete tables of invariants for systems of vectors under the action of the rotation group, the affine transform group, and the general linear transform group may be found in [Weyl 46]. To obtain new sets of functional invariants, several methods (eliminating transform parameters, the infinitesimal method, the symbolic method) can be found in [Forsyth et al. 91, Mundy and Zisserman 92].

Stability of invariants is another crucial property which affects their applicability. The robustness of invariants to image noise and errors introduced by image sensors is of prime importance, although not much is known about this. Results of plane-projective invariant stability testing (cross ratio, five coplanar points, two coplanar conics) can be found in [Forsyth et al. 91, Hopcroft et al. 92]. Further, different invariants have different stabilities and distinguishing powers. It was found, for example [Rothwell et al. 92a], that measuring a single conic and two lines in a scene is too computationally expensive to be worthwhile. It is recommended to combine different invariants to enable fast object recognition.

An example of recognition of man-made objects using invariant description of four coplanar lines, a conic and two lines, and a pair of coplanar conics is given in [Rothwell et al. 92a].

The recognition system is based on a model library containing over thirty object models – significantly more than that reported for other recognition systems. Moreover, the construction of the model library is extremely easy; no special measurements are needed, the object is digitized in a standard way and the projectively invariant description is stored as a model [Rothwell et al. 92b]. Further, there is no need for camera calibration. The recognition accuracy is 100% for occluded objects viewed from different viewpoints if the objects are not severely disrupted by shadows and specularities. An example of such object recognition is given in Figure 6.21.

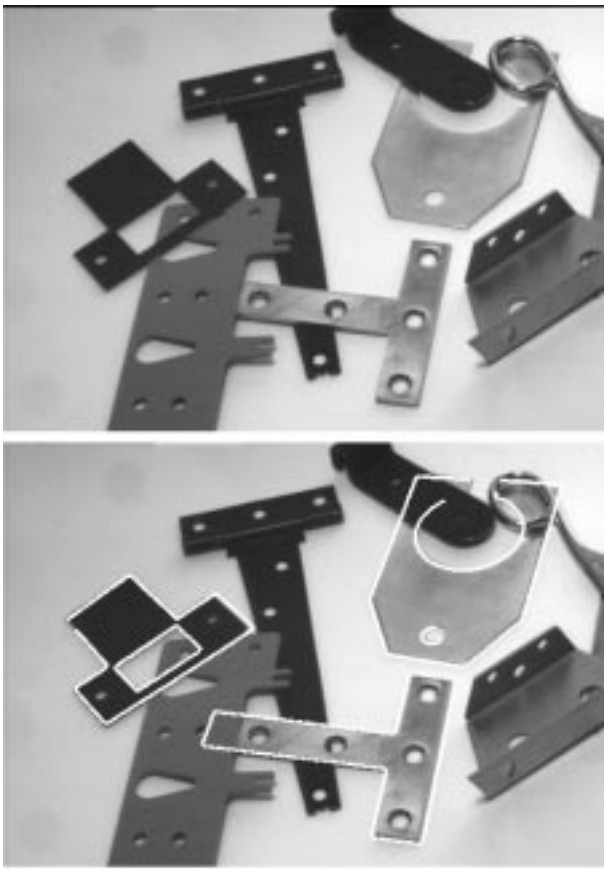


Figure 6.21: *Object recognition based on shape invariants: (a) Original image of overlapping objects taken from an arbitrary viewpoint, (b) object recognition based on line and conic invariants. Courtesy D. Forsyth, The University of Iowa; C. Rothwell, A. Zisserman, University of Oxford; J. Mundy, General Electric Corporate Research and Development, Schenectady, NY.*

6.3 Region-based shape representation and description

We can use boundary information to describe a region, and shape can be described from the region itself. A large group of shape description techniques is represented by heuristic approaches which yield acceptable results in description of simple shapes. Region area,

rectangularity, elongatedness, direction, compactness, etc. are examples of these methods. Unfortunately, they cannot be used for region reconstruction and do not work for more complex shapes. Other procedures based on region decomposition into smaller and simpler subregions must be applied to describe more complicated regions, then subregions can be described separately using heuristic approaches. Objects are represented by a planar graph with nodes representing subregions resulting from region decomposition, and region shape is then described by the graph properties [Rosenfeld 79, Bhanu and Faugeras 84, Turney et al. 85]. There are two general approaches to acquiring a graph of subregions: The first one is region thinning leading to the **region skeleton**, which can be described by a graph. The second option starts with the **region decomposition** into subregions, which are then represented by nodes while arcs represent neighborhood relations of subregions. It is common to stipulate that subregions be convex.

Graphical representation of regions has many advantages; the resulting graphs

- are translation and rotation invariant; position and rotation can be included in the graph definition
- are insensitive to small changes in shape
- are highly invariant with respect to region magnitude
- generate a representation which is understandable
- can easily be used to obtain the information-bearing features of the graph
- are suitable for syntactic recognition

On the other hand, the shape representation can be difficult to obtain and the classifier-learning stage is not easy either (see Chapter 7). Nevertheless, if we are to get closer to the reality of computer vision, and to understand complex images, there is no alternative.

6.3.1 Simple scalar region descriptors

A number of simple heuristic shape descriptors exist which relate to statistical feature description. These methods are basic and are used for description of subregions in complex regions, and may then be used to define graph node classification [Bribiesca and Guzman 80].

Area

The simplest and most natural property of a region is its area, given by the number of pixels of which the region consists. The *real* area of each pixel may be taken into consideration to get the *real size* of a region, noting that in many cases, especially in satellite imagery, pixels in different positions correspond to different areas in the real world. If an image is represented as a rectangular raster, simple counting of region pixels will provide its area. If the image is represented by a quadtree, however, it may be more difficult to find the region area. Assuming that regions have been identified by labeling, the following algorithm may be used.

Algorithm 6.4: Calculating area in quadtrees

1. Set all region area variables to zero, and determine the global quadtree depth H; for example, the global quadtree depth is $H = 8$ for a 256×256 image.
2. Search the tree in a systematic way. If a leaf node at a depth h has a non-zero label, proceed to step (3).
3. Compute:
$$area[region\_label] = area[region\_label] + 4^{(H-h)}$$
4. The region areas are stored in variables $area[region\_label]$.

The region can be represented by n polygon vertices (i_k, j_k), and $(i_0, j_0) = (i_n, j_n)$. The area is given by

$$area = \frac{1}{2}\left|\sum_{k=0}^{n-1}(i_k j_{k+1} - i_{k+1} j_k)\right| \tag{6.33}$$

– the sign of the sum represents the polygon orientation. If a smoothed boundary is used to overcome noise sensitivity problems, the region area value resulting from equation (6.33) is usually somewhat reduced. Various smoothing methods and accurate area-recovering techniques are given in [Koenderink and van Doorn 86].

If the region is represented by the (anti-clockwise) Freeman chain code the following algorithm provides the area;

Algorithm 6.5: Region area calculation from Freeman 4-connectivity chain code representation

1. Set the region *area* to zero. Assign the value of the starting point i co-ordinate to the variable *vertical_position*.
2. For each element of the chain code (values 0, 1, 2, 3) do

```
switch(code) {
    case 0:
        area := area - vertical_position;
        break;
    case 1:
        vertical_position := vertical_position + 1;
        break;
    case 2:
        area := area + vertical_position;
        break;
    case 3:
        vertical_position := vertical_position - 1;
        break;
}
```

3. If all boundary chain elements have been processed, the region area is stored in the variable *area*.

Euler's number

Euler's number ϑ (sometimes called **Genus** or the **Euler-Poincaré characteristic**) describes a simple topologically invariant property of the object. It is based on S, the number of contiguous parts of an object and N, the number of holes in the object (an object can consist of more than one region, otherwise the number of contiguous parts is equal to one (see Section 2.3.1)). Then

$$\vartheta = S - N \tag{6.34}$$

Special procedures to compute Euler's number can be found in [Dyer 80, Rosenfeld and Kak 82, Pratt 91], and in Chapter 11.

Projections,

Horizontal and vertical region projections $p_h(i)$ and $p_v(j)$ are defined as

$$p_h(i) = \sum_j f(i,j) \qquad\qquad p_v(j) = \sum_i f(i,j) \tag{6.35}$$

Region description by projections is usually connected to binary image processing. Projections can serve as a basis for definition of related region descriptors.; for example, the width (height) of a region with no holes is defined as the maximum value of the horizontal (vertical) projection of a binary image of the region. These definitions are illustrated in Figure 6.22. Note that projections can be defined in any direction. A practical example exploiting the use of projections is described in Section 16.1.

Eccentricity

The simplest eccentricity characteristic is the ratio of the length of the maximum chord A to the maximum chord B which is perpendicular to A (the ratio of major and minor axes of an object) – see Section 6.2.2, Figure 6.23. Another approximate eccentricity measure is based on a ratio of main region axes of inertia [Ballard and Brown 82, Jain 89].

Elongatedness

Elongatedness is a ratio between the length and width of the region bounding rectangle. This is the rectangle of minimum area that bounds the shape, which is located by turning in discrete steps until a minimum is located (see Figure 6.24a). This criterion cannot succeed in curved regions (see Figure 6.24b), for which the evaluation of elongatedness must be based on maximum region thickness. Elongatedness can be evaluated as a ratio of the region area and the square of its thickness. The maximum region thickness (holes must be filled if present)

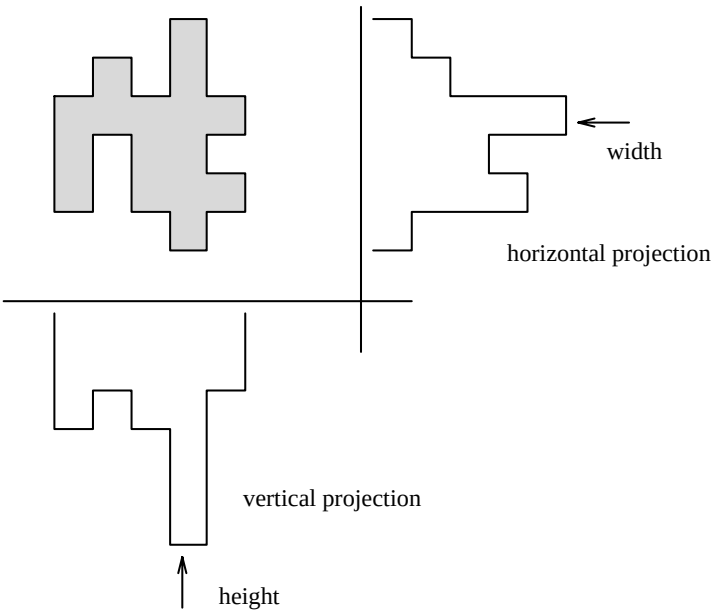


Figure 6.22: *Projections.*

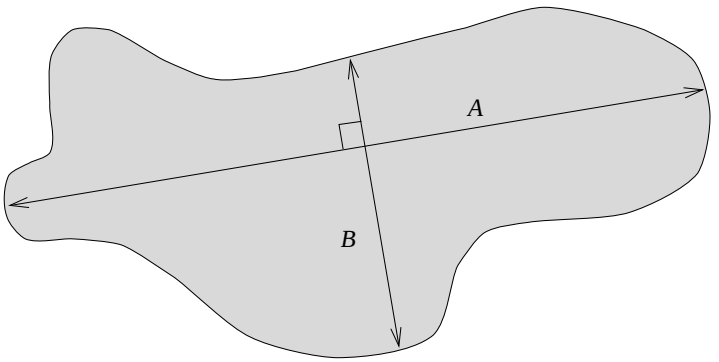


Figure 6.23: *Eccentricity.*

can be determined as the number of erosion steps (see Chapter 11) that may be applied before the region totally disappears. If the number of erosion steps is d, elongatedness is then

$$elongatedness = \frac{area}{(2d)^2} \tag{6.36}$$

Another method based on longest central line detection is described in [Nagao and Matsuyama 80]; representation and recognition of elongated regions is also discussed in [Lipari and Harlow 88].

Note that the bounding rectangle can be computed efficiently from boundary points, if its direction θ is known. Defining

$$\alpha(x,y) = x\cos\theta + y\sin\theta, \quad \beta(x,y) = -x\sin\theta + y\cos\theta \tag{6.37}$$

search for the minimum and maximum of α and β over all boundary points (x,y). The values

of $\alpha_{min}, \alpha_{max}, \beta_{min}, \beta_{max}$ then define the bounding rectangle, and $l_1 = (\alpha_{max} - \alpha_{min})$ and $l_2 = (\beta_{max} - \beta_{min})$ are its length and width.

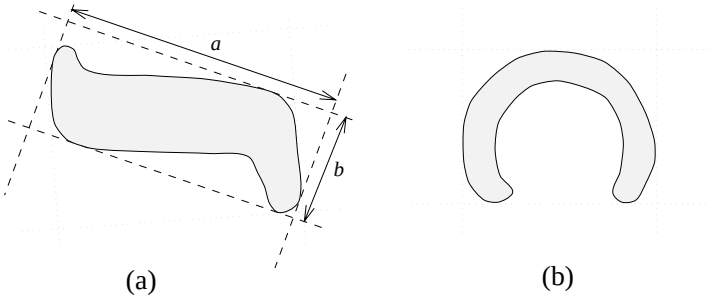


Figure 6.24: *Elongatedness: (a) Bounding rectangle gives acceptable results, (b) bounding rectangle cannot represent elongatedness.*

Rectangularity

Let F_k be the ratio of region area and the area of a bounding rectangle, the rectangle having the direction k. The rectangle direction is turned in discrete steps as before, and **rectangularity** measured as a maximum of this ratio F_k;

$$rectangularity = \max_k(F_k) \tag{6.38}$$

The direction need only be turned through one quadrant. Rectangularity assumes values from the interval $(0, 1]$, with 1 representing a perfectly rectangular region. Sometimes, it may be more natural to draw a bounding triangle; a method for similarity evaluation between two triangles called **sphericity** is presented in [Ansari and Delp 90].

Direction

Direction is a property which makes sense in elongated regions only. If the region is elongated, **direction** is the direction of the longer side of a minimum bounding rectangle. If the shape moments are known (Section 6.3.2), the direction θ can be computed as

$$\theta = \frac{1}{2} tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{6.39}$$

It should be noted that elongatedness and rectangularity are independent of linear transformations – translation, rotation, and scaling. Direction is independent on all linear transformations which do not include rotation. Mutual direction of two rotating objects is rotation invariant.

Compactness

Compactness is a popular shape description characteristic independent of linear transformations given by

$$compactness = \frac{(region\_border\_length)^2}{area} \tag{6.40}$$

The most compact region in a Euclidean space is a circle. Compactness assumes values in the interval $[1, \infty)$ in digital images if the boundary is defined as an inner boundary (see Section 5.2.3), while using the outer boundary, compactness assumes values in the interval $[16, \infty)$. Independence from linear transformations is gained only if an outer boundary representation is used. Examples of a compact and a non-compact region are shown in Figure 6.25.

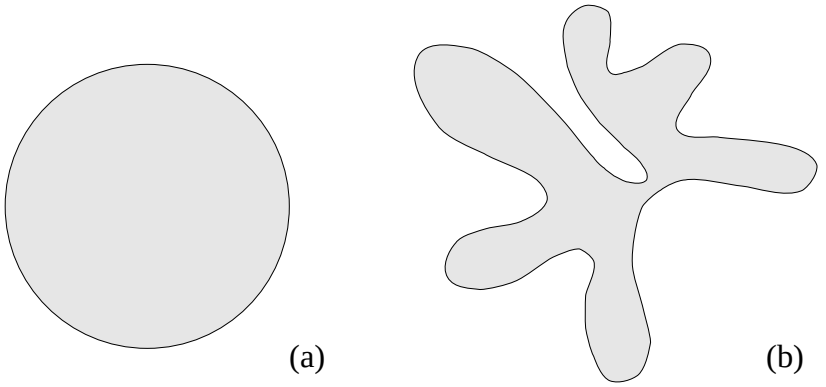


Figure 6.25: *Compactness: (a) Compact, (b) non-compact.*

6.3.2 Moments

Region moment representations interpret a normalized gray level image function as a probability density of a 2D random variable. Properties of this random variable can be described using statistical characteristics – **moments** [Papoulis 91]. Assuming that non-zero pixel values represent regions, moments can be used for binary or gray level region description. A moment of order $(p+q)$ is dependent on scaling, translation, rotation, and even on gray level transformations and is given by

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y)\, dx\, dy \tag{6.41}$$

In digitized images we evaluate sums

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i, j) \tag{6.42}$$

where x, y, i, j are the region point co-ordinates (pixel co-ordinates in digitized images). Translation invariance can be achieved if we use the central moments

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - x_c)^p (y - y_c)^q f(x, y)\, dx\, dy \tag{6.43}$$

or in digitized images

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i, j) \tag{6.44}$$

where x_c, y_c are the co-ordinates of the region's center of gravity (centroid) which can be obtained using the following relationships

$$x_c = \frac{m_{10}}{m_{00}} \tag{6.45}$$

$$y_c = \frac{m_{01}}{m_{00}}$$

In the binary case, m_{00} represents the region area (see equations (6.41) and (6.42)). Scale invariant features can also be found in scaled central moments η_{pq} (scale change $x' = \alpha x, y' = \alpha y$),

$$\eta_{pq} = \frac{\mu'_{pq}}{(\mu'_{00})^\gamma} \tag{6.46}$$

$$\gamma = \frac{p+q}{2} + 1$$

$$\mu'_{pq} = \frac{\mu_{pq}}{\alpha^{(p+q+2)}}$$

and normalized unscaled central moments ϑ_{pq}

$$\vartheta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma} \tag{6.47}$$

Rotation invariance can be achieved if the co-ordinate system is chosen such that $\mu_{11} = 0$ [Cash and Hatamian 87]. Many aspects of moment properties, normalization, descriptive power, sensitivity to noise, and computational cost are discussed in [Savini 88]. A less general form of invariance was given in [Hu 62] and is discussed in [Maitra 79, Jain 89, Pratt 91], in which seven rotation, translation, and scale invariant moment characteristics were used.

$$\varphi_1 = \vartheta_{20} + \vartheta_{02} \tag{6.48}$$

$$\varphi_2 = (\vartheta_{20} - \vartheta_{02})^2 + 4\vartheta_{11}^2 \tag{6.49}$$

$$\varphi_3 = (\vartheta_{30} - 3\vartheta_{12})^2 + (3\vartheta_{21} - \vartheta_{03})^2 \tag{6.50}$$

$$\varphi_4 = (\vartheta_{30} + \vartheta_{12})^2 + (\vartheta_{21} + \vartheta_{03})^2 \tag{6.51}$$

$$\begin{aligned}\varphi_5 = \; & (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{30} + \vartheta_{12})[(\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2] + \\ & (3\vartheta_{21} - \vartheta_{03})(\vartheta_{21} + \vartheta_{03})[3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2]\end{aligned} \tag{6.52}$$

$$\varphi_6 = (\vartheta_{20} - \vartheta_{02})[(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2] + 4\vartheta_{11}(\vartheta_{30} + \vartheta_{12})(\vartheta_{21} + \vartheta_{03}) \tag{6.53}$$

$$\begin{aligned}\varphi_7 = \; & (3\vartheta_{21} - \vartheta_{03})(\vartheta_{30} + \vartheta_{12})[(\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2] - \\ & (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{21} + \vartheta_{03})[3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2]\end{aligned} \tag{6.54}$$

where the ϑ_{pq} values can be computed from equation (6.47).

While the seven moment characteristics presented above were shown to be useful, they are only invariant to translation, rotation, and scaling. Recent algorithms for fast computation of translation-, rotation-, and scale-invariant moments were given in [Li and Shen 91, Jiang

and Bunke 91]. However, these approaches do not yield descriptors that are invariant under general affine transforms. A complete set of four affine moment invariants derived from second- and third-order moments was presented in [Flusser and Suk 93].

$$I_1 = \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4} \tag{6.55}$$

$$I_2 = \frac{\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{21}^3\mu_{03} - 3\mu_{21}^2\mu_{12}^2}{\mu_{00}^{10}} \tag{6.56}$$

$$I_3 = \frac{\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)}{\mu_{00}^7} \tag{6.57}$$

$$\begin{aligned} I_4 = \quad & (\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2 \\ & +12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} \\ & -8\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21}^2 \\ & +12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} + \mu_{02}^3\mu_{30}^2)/\mu_{00}^{11} \end{aligned} \tag{6.58}$$

Details of the process for the derivation of invariants and examples of invariant moment object descriptions can be found in [Flusser and Suk 93], and a complete proof and detailed discussion of the properties of them are given in [Flusser and Suk 91].

All moment characteristics are dependent on the linear gray level transformations of regions; to describe region shape properties, we work with binary image data ($f(i,j) = 1$ in region pixels) and dependence on the linear gray level transform disappears.

Moment characteristics can be used in shape description even if the region is represented by its boundary. A closed boundary is characterized by an ordered sequence $z(i)$ that represents the Euclidean distance between the centroid and all N boundary pixels of the digitized shape. No extra processing is required for shapes having spiral or concave contours. Translation, rotation, and scale invariant one-dimensional normalized contour sequence moments $\overline{m}_r, \overline{\mu}_r$ are defined in [Gupta and Srinath 87]. The r^{th} contour sequence moment m_r and the r^{th} central moment μ_r can be estimated as

$$m_r = \frac{1}{N}\sum_{i=1}^{N}[z(i)]^r \tag{6.59}$$

$$\mu_r = \frac{1}{N}\sum_{i=1}^{N}[z(i) - m_1]^r \tag{6.60}$$

The r^{th} normalized contour sequence moment $\overline{m}_r$ and normalized central contour sequence moment $\overline{\mu}_r$ are defined as

$$\overline{m}_r = \frac{m_r}{(\mu_2)^{r/2}} = \frac{\frac{1}{N}\sum_{i=1}^{N}[z(i)]^r}{[\frac{1}{N}\sum_{i=1}^{N}[z(i) - m_1]^2]^{r/2}} \tag{6.61}$$

$$\overline{\mu}_r = \frac{\mu_r}{(\mu_2)^{r/2}} = \frac{\frac{1}{N}\sum_{i=1}^{N}[z(i)-m_1]^r}{[\frac{1}{N}\sum_{i=1}^{N}[z(i)-m_1]^2]^{r/2}} \tag{6.62}$$

While the set of invariant moments $\overline{m}_r, \overline{\mu}_r$ can be directly used for shape representation, less noise-sensitive results can be obtained from the following shape descriptors [Gupta and Srinath 87]

$$F_1 = \frac{(\mu_2)^{1/2}}{m_1} = \frac{[\frac{1}{N}\sum_{i=1}^{N}[z(i)-m_1]^2]^{1/2}}{\frac{1}{N}\sum_{i=1}^{N} z(i)} \tag{6.63}$$

$$F_2 = \frac{\mu_3}{(\mu_2)^{3/2}} = \frac{\frac{1}{N}\sum_{i=1}^{N}[z(i)-m_1]^3}{[\frac{1}{N}\sum_{i=1}^{N}[z(i)-m_1]^2]^{3/2}} \tag{6.64}$$

$$F_3 = \frac{\mu_4}{(\mu_2)^2} = \frac{\frac{1}{N}\sum_{i=1}^{N}[z(i)-m_1]^4}{[\frac{1}{N}\sum_{i=1}^{N}[z(i)-m_1]^2]^2} \tag{6.65}$$

$$F_4 = \overline{\mu}_5 \tag{6.66}$$

Lower probabilities of error classification were obtained using contour sequence moments than area based moments (equation (6.48) – 6.54) in a shape recognition test; also, contour sequence moments are less computationally demanding.

6.3.3 Convex hull

A region R is convex if and only if for any two points $\mathbf{x}_1, \mathbf{x}_2 \in R$, the whole line segment $\mathbf{x}_1\mathbf{x}_2$ defined by its end-points $\mathbf{x}_1, \mathbf{x}_2$ is inside the region R. The convex hull of a region is the smallest convex region H which satisfies the condition $R \subseteq H$ – see Figure 6.26. The convex hull has some special properties in digital data which do not exist in the continuous case. For instance, concave parts can appear and disappear in digital data due to rotation, and therefore the convex hull is not rotation invariant in digital space [Gross and Latecki 95]. The convex hull can be used to describe region shape properties and can be used to build a tree structure of region concavity.

A discrete convex hull can be defined by the following algorithm which may also be used for convex hull construction. This algorithm has complexity $\mathcal{O}(n^2)$ and is presented here as an intuitive way of detecting the convex hull. Algorithm 6.7 describes a more efficient approach.

Algorithm 6.6: Region convex hull construction

1. Find all pixels of a region R with the minimum row co-ordinate; among them, find the pixel P_1 with the minimum column co-ordinate.
   Assign $\mathbf{P}_k = \mathbf{P}_1$, $\mathbf{v} = (0, -1)$; the vector $\mathbf{v}$ represents the direction of the previous line segment of the convex hull.

2. Search the region boundary in an anti-clockwise direction (Algorithm 5.8) and compute the angle orientation φ_n for every boundary point $\mathbf{P}_n$ which lies after the point $\mathbf{P}_1$ (in the direction of boundary search – see Figure 6.26). The angle orientation φ_n is the angle of vector $\mathbf{P}_k\mathbf{P}_n$. The point $\mathbf{P}_q$ satisfying the condition $\varphi_q = \min_n \varphi_n$ is an element (vertex) of the region convex hull.

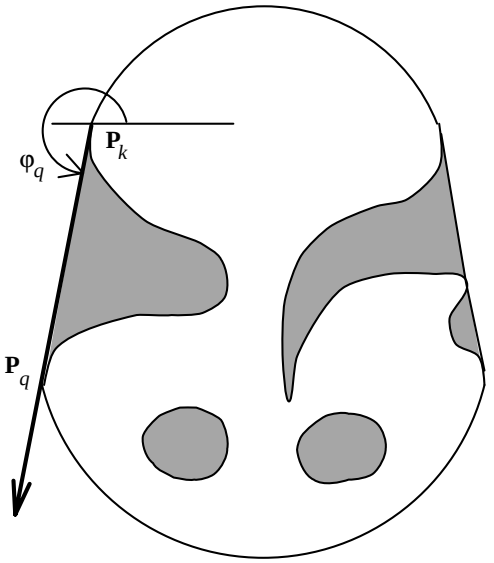


Figure 6.26: *Convex hull.*

3. Assign $\mathbf{v} = \mathbf{P}_k\mathbf{P}_q$, $\mathbf{P}_k = \mathbf{P}_q$.

4. Repeat steps (2) and (3) until $\mathbf{P}_k = \mathbf{P}_1$.

The first point $\mathbf{P}_1$ need not be chosen as described in the given algorithm, but it must be an element of a convex segment of the inner region boundary.

As has been mentioned, more efficient algorithms exist, especially if the object is defined by an ordered sequence $P = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ of n vertices $\mathbf{v}_i$ representing a polygonal boundary of the object. Many algorithms [Toussaint 85] exist for detection of the convex hull with computational complexity $\mathcal{O}(n \log n)$ in the worst case; these algorithms and their implementations vary in speed and memory requirements. As discussed in [Toussaint 91], the code of [Bhattacharya and Toussaint 83] (in which a Fortran listing appears) seems to be the fastest to date using only $5n$ storage space.

If the polygon P is a *simple* polygon (self-non-intersecting polygon) which is always the case in a polygonal representation of object borders, the convex hull may be found in linear time $\mathcal{O}(n)$. In the past two decades, many linear-time convex hull detection algorithms have been published, however more than half of them were later discovered to be incorrect [Toussaint 85, Toussaint 91], with counter-examples published. The algorithm of [McCallum and Avis 79] was the first correct linear-time one. The simplest correct convex hull algorithm was given in [Melkman 87] and was based on previous work [Lee 83, Bhattacharya and Gindy 84, Graham and Yao 84]. Melkman's convex hull detection algorithm is now discussed further.

Let the polygon for which the convex hull is to be determined be a simple polygon $P = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ and let the vertices be processed in this order. For any three vertices $\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$

in an ordered sequence, a directional function δ may be evaluated (Figure 6.27)

$$\begin{aligned} \delta(\mathbf{x},\mathbf{y},\mathbf{z}) &= 1 \quad \text{if } \mathbf{z} \text{ is to the right of the directed line } \mathbf{xy} \\ &= 0 \quad \text{if } \mathbf{z} \text{ is collinear with the directed line } \mathbf{xy} \\ &= -1 \quad \text{if } \mathbf{z} \text{ is to the left of the directed line } \mathbf{xy} \end{aligned}$$

The main data structure H is a list of vertices (deque) of polygonal vertices already processed.

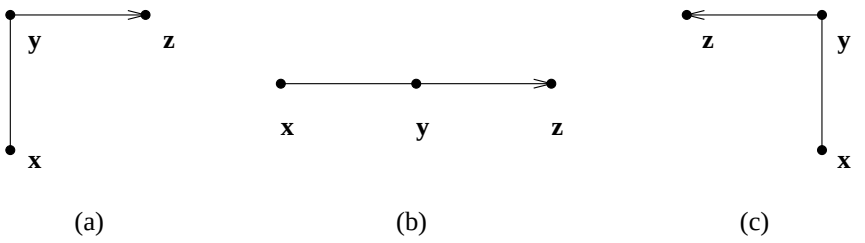


Figure 6.27: *Directional function δ: (a) $\delta(\mathbf{x},\mathbf{y},\mathbf{z}) = 1$, (b) $\delta(\mathbf{x},\mathbf{y},\mathbf{z}) = 0$, (c) $\delta(\mathbf{x},\mathbf{y},\mathbf{z}) = -1$.*

The current contents of H represents the convex hull of the currently processed part of the polygon, and after the detection is completed, the convex hull is stored in this data structure. Therefore, H always represents a closed polygonal curve, $H = \{d_b, \ldots, d_t\}$ where d_b points to the bottom of the list and d_t points to its top. Note that d_b and d_t always refer to the same vertex simultaneously representing the first and the last vertex of the closed polygon.

Here are the main ideas of the algorithm. The first three vertices A, B, C from the sequence P form a triangle (if not collinear) and this triangle represents a convex hull of the first three vertices – Figure 6.28a. The next vertex D in the sequence is then tested for being located inside or outside the current convex hull. If D is located inside, the current convex hull does not change – Figure 6.28b. If D is outside of the current convex hull, it must become a new convex hull vertex (Figure 6.28c) and based on the current convex hull shape, either none, one, or several vertices must be removed from the current convex hull – Figure 6.28c,d. This process is repeated for all remaining vertices in the sequence P.

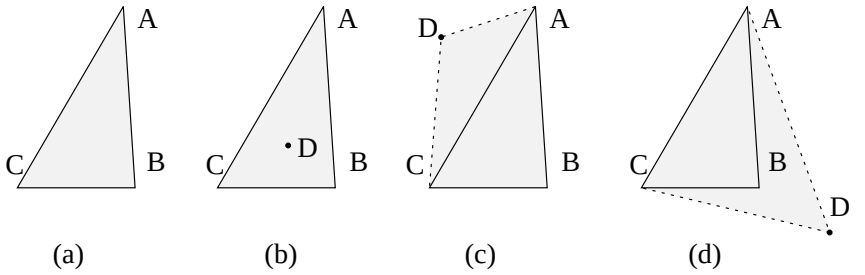


Figure 6.28: *Convex hull detection: (a) First three vertices A, B, C form a triangle, (b) if the next vertex D is positioned inside the current convex hull ABC, current convex hull does not change, (c) if the next vertex D is outside of the current convex hull, it becomes a new vertex of the new current convex hull $ABCDA$, (d) in this case, vertex B must be removed from the current convex hull and the new current convex hull is $ADCA$.*

Following the terminology used in [Melkman 87], the variable $\mathbf{v}$ refers to the input vertex

under consideration, and the following operations are defined:

`push` $\mathbf{v}$	:	$t := t + 1; \quad d_t \rightarrow \mathbf{v}$
`pop` d_t	:	$t := t - 1$
`insert` $\mathbf{v}$	:	$b := b - 1; \quad d_b \rightarrow \mathbf{v}$
`remove` d_b	:	$b := b + 1$
`input` $\mathbf{v}$	:	next vertex is entered from sequence P, if P is empty, stop.

where $\rightarrow$ means 'points to'. The algorithm is then;

Algorithm 6.7: Simple polygon convex hull detection

1. Initialize;
   ```
   .   t := -1;
   .   b := 0;
   .   input v1; input v2; input v3;
   .   if ( δ(v1, v2, v3) > 0 )
   .         { push v1;
   .           push v2; }
   .      else
   .         { push v2;
   .           push v1; }
   .   push v3;
   .   insert v3;
   ```

2. If the next vertex $\mathbf{v}$ is inside the current convex hull H, enter and check a new vertex; otherwise process steps (3) and (4);
   ```
   .   input v;
   .   while ( δ(v, d_b, d_{b+1}) ≥ 0   AND   δ(d_{t-1}, d_t, v) ≥ 0 )
   .      input v;
   ```

3. Rearrange vertices in H, top of the list.
   ```
   .   while ( δ(d_{t-1}, d_t, v) ≤ 0 )
   .      pop d_t;
   .   push v;
   ```

4. Rearrange vertices in H, bottom of the list.
   ```
   .   while ( δ(v, d_b, d_{b+1}) ≤ 0 )
   .      remove d_b;
   .   insert v;
   .   go to step (2);
   ```

The algorithm as presented may be difficult to follow, however, a less formal version would be impossible to implement; a formal proof is given in [Melkman 87]. The following example makes the algorithm more understandable.

Let $P = \{A, B, C, D, E\}$ as shown in Figure 6.29a. The data structure H is created in the first step;

$$\begin{array}{lcccccc} t, b \ldots & & -1 & 0 & 1 & 2 \\ H & = & C & A & B & C \\ & & d_b & & & d_t \end{array}$$

In the second step, vertex D is entered (Figure 6.29b);

$$\begin{array}{lcl} \delta(D, d_b, d_{b+1}) & = & \delta(D, C, A) = 1 > 0 \\ \delta(d_{t-1}, d_t, D) & = & \delta(B, C, D) = -1 < 0 \end{array}$$

Based on the values of the directional function δ, in this case, no other vertex is entered during this step. Step (3) results in the following current convex hull H;

$$\delta(B, C, D) = -1 \longrightarrow \texttt{pop}\ d_t \longrightarrow$$

$$\begin{array}{lcccccc} t, b \ldots & & -1 & 0 & 1 & 2 \\ H & = & C & A & B & C \\ & & d_b & & d_t & \end{array}$$

$$\delta(A, B, D) = -1 \longrightarrow \texttt{pop}\ d_t \longrightarrow$$

$$\begin{array}{lcccccc} t, b \ldots & & -1 & 0 & 1 & 2 \\ H & = & C & A & B & C \\ & & d_b & d_t & & \end{array}$$

$$\delta(C, A, D) = 1 \longrightarrow \texttt{push}\ D \longrightarrow$$

$$\begin{array}{lcccccc} t, b \ldots & & -1 & 0 & 1 & 2 \\ H & = & C & A & D & C \\ & & d_b & & d_t & \end{array}$$

In step (4) – Figure 6.29c;

$$\delta(D, C, A) = 1 \longrightarrow \texttt{insert}\ D \longrightarrow$$

$$\begin{array}{lcccccc} t, b \ldots & & -2 & -1 & 0 & 1 & 2 \\ H & = & D & C & A & D & C \\ & & d_b & & & d_t & \end{array}$$

Go to step (2); vertex E is entered – Figure 6.29d;

$$\begin{array}{lcl} \delta(E, D, C) & = & 1 > 0 \\ \delta(A, D, E) & = & 1 > 0 \end{array}$$

A new vertex should be entered from P, however there is no unprocessed vertex in the sequence P and the convex hull generating process stops. The resulting convex hull is defined by the sequence $H = \{d_b, \ldots, d_t\} = \{D, C, A, D\}$ which represents a polygon $DCAD$, always in the clockwise direction – Figure 6.29e.

A **region concavity tree** is another shape representation option [Sklansky 72]. A tree is generated recursively during the construction of a convex hull. A convex hull of the whole region is constructed first, and convex hulls of concave residua are found next. The resulting convex hulls of concave residua of the regions from previous steps are searched until no concave residuum exists. The resulting tree is a shape representation of the region. Concavity tree construction can be seen in Figure 6.30.

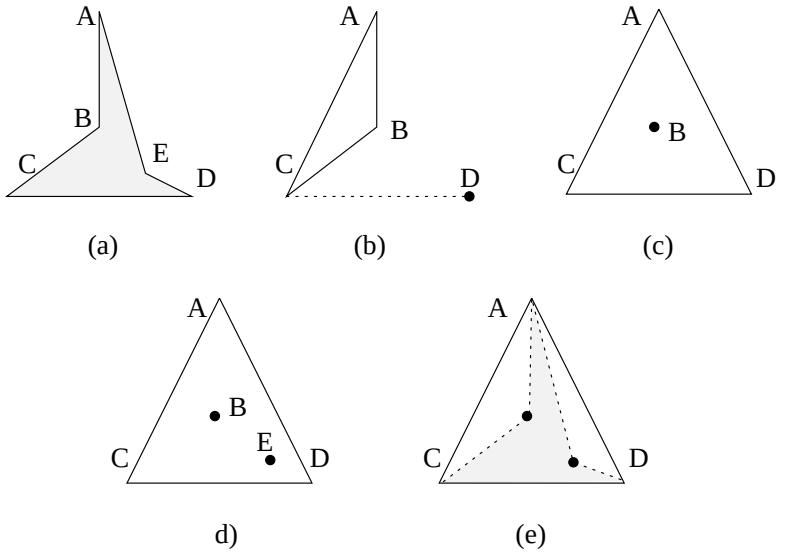


Figure 6.29: *Example of convex hull detection: (a) The processed region – polygon ABCDEA, (b) vertex D is entered and processed, (c) vertex D becomes a new vertex of the current convex hull ADC, (d) vertex E is entered and processed, E does not become a new vertex of the current convex hull, (e) the resulting convex hull DCAD.*

6.3.4 Graph representation based on region skeleton

This method corresponds significantly curving points of a region boundary to graph nodes. The main disadvantage of boundary-based description methods is that geometrically close points can be far away from one another when the boundary is described – graphical representation methods overcome this disadvantage. Shape properties are then derived from the graph properties.

The region graph is based on the region skeleton, and the first step is the skeleton construction. There are four basic approaches to skeleton construction:

- thinning – iterative removal of region boundary pixels
- wave propagation from the boundary
- detection of local maxima in the distance-transformed image of the region
- analytical methods

Most thinning procedures repeatedly remove boundary elements until a pixel set with maximum thickness of one or two is found. The following algorithm constructs a skeleton of maximum thickness two.

Algorithm 6.8: Skeleton by thinning

1. Let R be the set of region pixels, $H_i(R)$ its inner boundary, and $H_o(R)$ its outer boundary. Let $S(R)$ be a set of pixels from the region R which have all their neighbors in

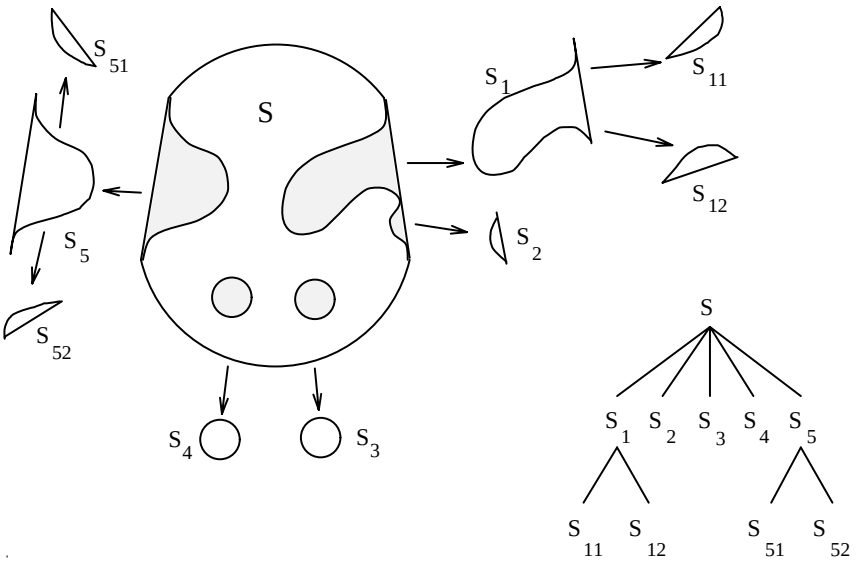


Figure 6.30: *Concavity tree construction: (a) Convex hull and concave residua, (b) concavity tree.*

8-connectivity either from the inner boundary $H_i(R)$ or from the background – from the residuum of R. Assign $R_{old} = R$.

2. Construct a region R_{new} which is a result of one-step thinning as follows

$$R_{new} = S(R_{old}) \cup [R_{old} - H_i(R_{old})] \cup [H_o(S(R_{old})) \cap R_{old}]$$

3. If $R_{new} = R_{old}$, terminate the iteration and proceed to step (4). Otherwise assign $R_{old} = R_{new}$ and repeat step (2).

4. R_{new} is a set of skeleton pixels, the skeleton of the region R.

Steps of this algorithm are illustrated in Figure 6.31. If there are skeleton segments which have a thickness of two in the skeleton, one extra step can be added to reduce those to a thickness of one, although care must be taken not to break the skeleton connectivity.

A large number of thinning algorithms can be found in the image processing literature [Hildich 69, Pavlidis 78]. If special prior conditions apply, these algorithms can be much simpler. Thinning is generally a time-consuming process, although sometimes it is not necessary to look for a skeleton, and one side of a parallel boundary can be used for skeleton-like region representation. Mathematical morphology is a powerful tool used to find the region skeleton, and thinning algorithms which use mathematical morphology are given in Chapter 11; see also [Maragos and Schafer 86] where the morphological approach is shown to unify many other approaches to skeletonization.

Thinning procedures often use a medial axis transform (also symmetric axis transform) to construct a region skeleton [Blum 73, Pavlidis 77, Samet 85, Arcelli and Sanniti di Baja 86,

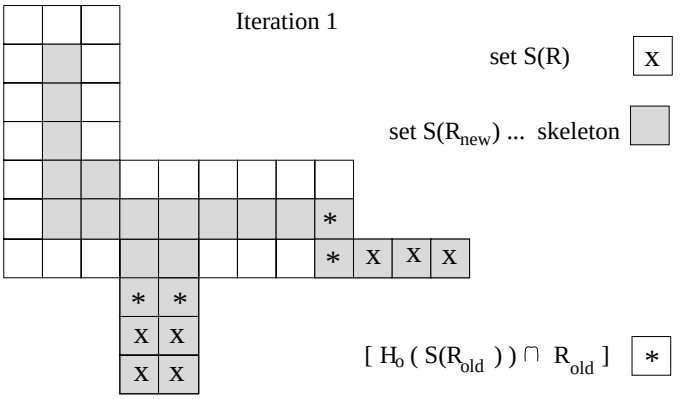


Figure 6.31: *Skeleton by thinning (Algorithm 6.8).*

Pizer et al. 87, Lam et al. 92, Wright and Fallside 93]. Under the medial axis definition, the skeleton is the set of all region points which have the same minimum distance from the region boundary for at least two separate boundary points. Examples of skeletons resulting from this condition are shown in Figures 6.32 and 6.33. Such a skeleton can be constructed using a distance transform which assigns a value to each region pixel representing its (minimum) distance from the region's boundary. The skeleton can be determined as a set of pixels whose distance from the region's border is locally maximal. As a post-processing step, local maxima can be detected using operators that detect linear features and roof profiles [Canny 83, Petrou 90, Wright and Fallside 93]. Every skeleton element can be accompanied by information about its distance from the boundary – this gives the potential to reconstruct a region as an envelope curve of circles with center points at skeleton elements and radii corresponding to the stored distance values. Shape descriptions, as discussed in Section 6.3.1 can be derived from this skeleton but, with the exception of elongatedness, the evaluation can be difficult. In addition, this skeleton construction is time-consuming and a resulting skeleton is highly sensitive to boundary noise and errors. Small changes in the boundary may cause serious changes in the skeleton – see Figure 6.32. This sensitivity can be removed by first representing the region as a polygon, then constructing the skeleton. Boundary noise removal can be absorbed into the polygon construction. A multi-resolution (scale-space) approach to skeleton construction may also result in decreased sensitivity to boundary noise [Pizer et al. 87, Maragos 89]. Similarly, the approach using the Marr-Hildreth edge detector with varying smoothing parameter facilitates scale-based representation of the region's skeleton [Wright and Fallside 93].

A method of skeleton construction based on the Fourier coefficients of a boundary T_n and S_n (see Section 6.2.3) is given in [Persoon and Fu 77]. Neural networks [Krishnapuram and Chen 91] and a Voronoi diagram approach [Brandt and Algazi 92, Ogniewicz and Ilg 92, Mayya and Rajan 95] can also be applied to find the skeleton. Fast parallel algorithms for thinning are given in [Guo and Hall 92]. Use of the intensity axis of symmetry represents an unconventional approach to skeletonization that does not require explicit region segmentation [Gauch and Pizer 93]. If derived from boundary data considering the scale, the intensity axes of symmetry are often called **cores** [Morse et al. 93, Fritsch et al. 97]. The cores are invariant to translation, rotation, linear variation of intensity, and scale, and are insensitive to small-

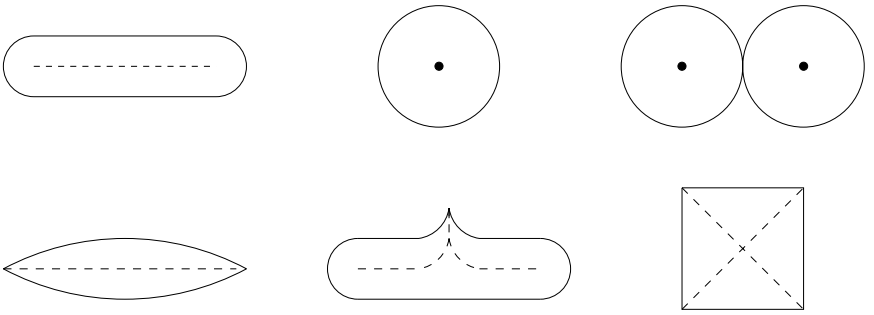
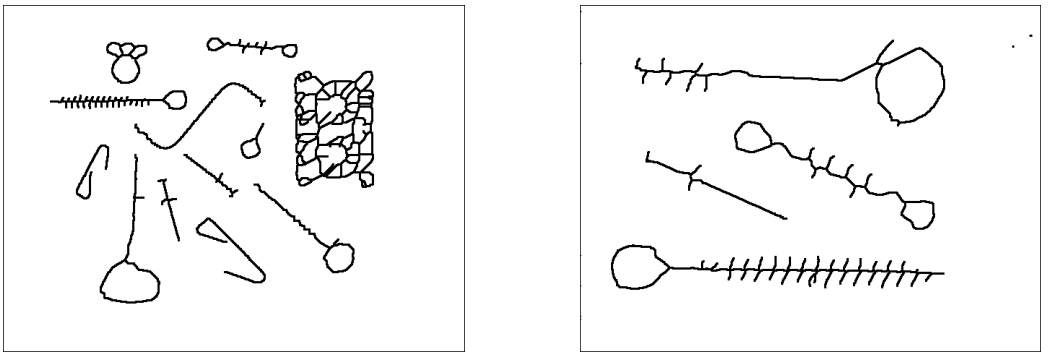


Figure 6.32: *Region skeletons; small changes in border can have a significant effect on the skeleton.*



(a) (b)

Figure 6.33: *Region skeletons, see Figures 5.1a and 6.2a for original images; thickened for visibility.*

scale noise (spatially uncorrelated), small-scale blurring (compared to the object's width), and small-scale local deformation.

Skeleton construction algorithms do not result in graphs but the transformation from skeletons to graphs is relatively straightforward. Consider first the medial axis skeleton, and assume that a minimum radius circle has been drawn from each point of the skeleton which has at least one point common with a region boundary. Let *contact* be each contiguous subset of the circle which is common to the circle and to the boundary. If a circle drawn from its center A has one contact only, A is a skeleton end-point. If the point A has two contacts, it is a normal skeleton point. If A has three or more contacts, the point A is a skeleton node-point.

Algorithm 6.9: Region graph construction from skeleton

1. Assign a point description to all skeleton points – end-point, node-point, normal-point.

2. Let graph node-points be all end-points and node-points. Connect any two graph nodes by a graph edge if they are connected by a sequence of normal-points in the region skeleton.

It can be seen that boundary points of high curvature have the main influence on the graph. They are represented by graph nodes, and therefore influence the graph structure.

If other than medial axis skeletons are used for graph construction, end-points can be defined as skeleton points having just one skeleton neighbor, normal-points as having two skeleton neighbors, and node-points as having at least three skeleton neighbors. It is no longer true that node-points are never neighbors and additional conditions must be used to decide when node-points should be represented as nodes in a graph and when they should not.

6.3.5 Region decomposition

The decomposition approach is based on the idea that shape recognition is a hierarchical process. Shape **primitives** are defined at the lower level, primitives being the simplest elements which form the region. A graph is constructed at the higher level – nodes result from primitives, arcs describe the mutual primitive relations. Convex sets of pixels are one example of simple shape primitives.

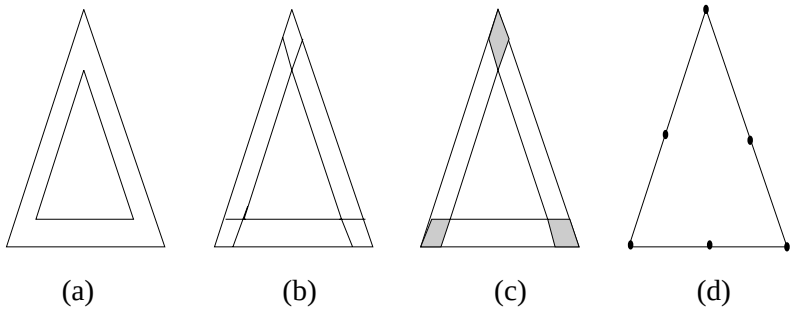


Figure 6.34: *Region decomposition: (a) Region, (b) primary regions, (c) primary subregions and kernels, (d) decomposition graph.*

The solution to the decomposition problem consists of two main steps: The first step is to segment a region into simpler subregions (primitives) and the second is the analysis of primitives. Primitives are simple enough to be successfully described using simple scalar shape properties (see Section 6.3.1). A detailed description of how to segment a region into primary convex subregions, methods of decomposition to concave vertices and graph construction resulting from a polygonal description of subregions are given in [Pavlidis 77]. The general idea of decomposition is shown in Figure 6.34 where the original region, one possible decomposition, and the resulting graph are presented. Primary convex subregions are labeled as primary subregions or kernels. Kernels (shown striped in Figure 6.34c) are subregions which belong to several primary convex subregions. If subregions are represented by polygons, graph nodes bear the following information;

1. Node type representing primary subregion or kernel.

2. Number of vertices of the subregion represented by the node.

3. Area of the subregion represented by the node.

4. Main axis direction of the subregion represented by the node.

5. Center of gravity of the subregion represented by the node.

If a graph is derived using attributes 1-4, the final description is translation invariant. A graph derived from attributes 1-3 is translation and rotation invariant. Derivation using the first two attributes results in a description which is size invariant in addition to possessing translation and rotation invariance.

A decomposition of a region uses its structural properties, and a syntactic graph description is the result. Problems of how to decompose a region and how to construct the description graph are still open; an overview of some techniques that have been investigated can be found in [Feng and Pavlidis 75, Moayer and Fu 75, Pavlidis 77, Stallings 76, Shapiro 80, di Baja and Thiel 94, Held and Abe 94]. Shape decomposition into a complete set of convex parts ordered by size is described in [Cortopassi and Rearick 88], and a morphological approach to skeleton decomposition is used to decompose complex shapes into simple components in [Zhou and Venetsanopoulos 89, Pitas and Venetsanopoulos 90, Kanungo and Haralick 92, Loncaric and Dhawan 95, Xiaoqi and Baozong 95, Wang et al. 95, Reinhardt and Higgins 96]; the decomposition is shown to be invariant to translation, rotation, and scaling. Recursive subdivision of shape based on second central moments is another translation, rotation, scaling, and intensity shift invariant decomposition technique [Zhu and Poh 88]. Hierarchical decomposition and shape description that uses region and contour information, addresses issues of local versus global information, scale, shape parts, and axial symmetry is given in [Rom and Medioni 92, Rom and Medioni 93]. Multiresolution approaches to decomposition are reported in [Loncaric and Dhawan 93, Cinque and Lombardi 95].

6.3.6 Region neighborhood graphs

Any time a region decomposition into subregions or an image decomposition into regions is available, the region or image can be represented by a region neighborhood graph (the region adjacency graph described in Section 3.2.3 being a special case). This graph represents every region as a graph node, and nodes of neighboring regions are connected by edges. A region neighborhood graph can be constructed from a quadtree image representation, from run-length encoded image data, etc. Binary tree shape representation is described in [Leu 89] where merging of boundary segments results in shape decomposition into triangles, their relations being represented by the binary tree.

Very often, the relative position of two regions can be used in the description process – for example, a region A may be positioned to the *left of* a region B, or *above* B, or *close to* B, or a region C may lie *between* regions A and B, etc. We know the meaning of all of the given relations if A, B, C are points, but, with the exception of the relation *to be close*, they can become ambiguous if A, B, C are regions. For instance (see Figure 6.35), the relation *to be left of* can be defined in many different ways;

- All pixels of A must be positioned to the left of all pixels of B.
- At least one pixel of A must be positioned to the left of some pixel of B.
- The center of gravity of A must be to the left of the center of gravity of B.

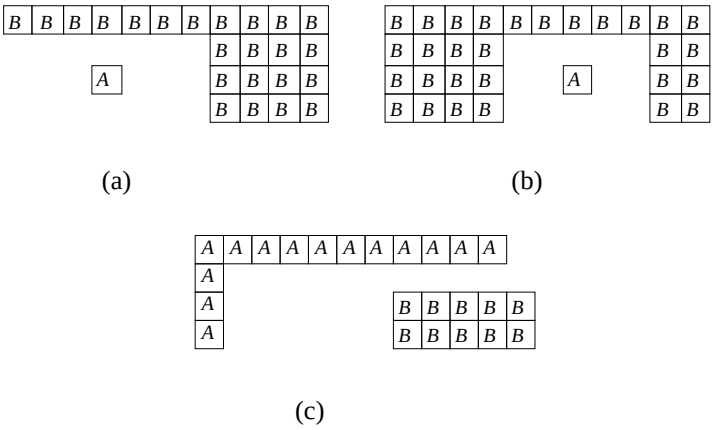


Figure 6.35: *Binary relation* to be left of*; see text.*

All of these definitions seem to be satisfactory in many cases but they can sometimes be unacceptable because they do not meet the usual meaning of *being left of*. Human observers are generally satisfied with the definition:

- The center of gravity of A must be positioned to the left of the leftmost point of B and (logical AND) the rightmost pixel of A must be left of the rightmost pixel of B [Winston 75].

Many other inter-regional relations are defined in [Winston 75] where relational descriptions are studied in detail.

An example of applying geometrical relations between simply shaped primitives to shape representation and recognition may be found in [Shariat 90], where recognition is based on a **hypothesize and verify** control strategy. Shapes are represented by region neighborhood graphs that describe geometrical relations among primitive shapes. The model-based approach increases the shape recognition accuracy and makes partially occluded object recognition possible. Recognition of any new object is based on a definition of a new shape model.

6.4 Shape classes

Representation of **shape classes** is considered a challenging problem of shape description [Hogg 93]. The shape classes are expected to represent the generic shapes of the objects belonging to the class well and emphasize shape differences between classes, while the shape variations allowed within classes should not influence the description.

There are many ways to deal with such requirements. A widely used representation of in-class shape variations is determination of class-specific regions in the feature space. The feature space can be defined using a selection of shape features described earlier in this chapter (for more information about feature spaces, see Chapter 7). Another approach to shape class definition is to use a single prototype shape and determine a planar warping transform that if applied to the prototype produces shapes from the particular class. The prototype shape may be derived from examples.

If a set of landmarks can be identified on the regions belonging to specific shape classes, the landmarks can characterize the classes in a simple and powerful way. Landmarks are usually selected as easily recognizable border or region points. For planar shapes, a co-ordinate system can be defined that is invariant to similarity transforms of the plane (rotation, translation, scaling) [Bookstein 91, Rangarajan et al. 97]. If such a landmark model uses n points per 2D object, the dimensionality of the shape space is $2n$. Clearly, only a subset of the entire shape space corresponds to each shape class and the shape class definition reduces to the definition of the shape space subsets. In [Cootes et al. 92], principal components in the shape space are determined from training sets of shapes after the shapes are iteratively aligned. The first few principal components characterize the most significant variations in shape. Thus, a small number of shape parameters represent the major shape variation characteristics associated with the shape class. Such a shape class representation is referred to as **point distribution models** and is discussed in detail in Section 8.3 in the context of image interpretation.

6.5 Summary

- **Shape representation and description**
  - Region description generates a numeric feature vector or a non-numeric syntactic description word, which characterize properties (for example, shape) of the described region.
  - While many practical shape description methods exist, there is no generally accepted methodology of shape description. Further, it is not known what in shape is important.
  - Shape may change substantially with image resolution. Conventional shape descriptions change discontinuously with changes in resolution. A **scale-space** approach aims to obtain continuous shape descriptions for continuous resolution changes.
  - The **shape classes** represent the generic shapes of the objects belonging to the same classes. Shape classes should emphasize shape differences among classes while the shape variations within classes should not be reflected in the shape class description.
- **Region identification**
  - Region identification assigns unique **labels** to image regions.
  - If nonrepeating ordered numerical labels are used, the largest integer label gives the number of regions in the image.
- **Contour-based shape descriptors**
  - **Chain codes** describe an object by a sequence of unit-size line segments with a given orientation, called **Freeman's** code.
  - **Simple geometric border representations** are based on geometric properties of described regions, e.g.:

    - Boundary length
    - Curvature
    - Bending energy
    - Signature
    - Chord distribution
  - **Fourier shape descriptors** can be applied to closed curves, co-ordinates of which can be treated as periodic signals.
  - Shape can be represented as a sequence of **segments** with specified properties. If the segment type is known for all segments, the boundary can be described as a chain of segment types, a code-word consisting of representatives of a type alphabet.
  - **B-splines** are piecewise polynomial curves whose shape is closely related to their control polygon – a chain of vertices giving a polygonal representation of a curve. B-splines of the third-order are most common, representing the lowest order which includes the change of curvature.
  - **Shape invariants** represent properties of geometric configurations that remain unchanged under an appropriate class of transforms; machine vision is especially concerned with the class of projective transforms.

- **Region-based shape descriptors**
  - Simple geometric region descriptors use geometric properties of described regions:
    - Area
    - Euler's number
    - Projections
    - Height, width
    - Eccentricity
    - Elongatedness
    - Rectangularity
    - Direction
    - Compactness
  - **Statistical moments** interpret a normalized gray level image function as a probability density of a 2D random variable. Properties of this random variable can be described using statistical characteristics – **moments**. Moment-based descriptors can be defined to be independent of scaling, translation and rotation.
  - The **convex hull** of a region is the smallest convex region H which satisfies the condition $R \subset H$.
  - More complicated shapes can be described using region decomposition into smaller and simpler subregions. Objects can be represented by a planar graph with nodes representing subregions resulting from region decomposition. Region shape can then be described by the graph properties. There are two general approaches to acquiring a graph of subregions:

    * Region thinning
    * Region decomposition
  - **Region thinning** leads to the region **skeleton** that can be described by a graph. Thinning procedures often use a medial axis transform to construct a region skeleton. Under the medial axis definition, the skeleton is the set of all region points which have the same minimum distance from the region boundary for at least two separate boundary points.
  - **Region decomposition** considers shape recognition to be a hierarchical process. Shape **primitives** are defined at the lower level, primitives being the simplest elements which form the region. A graph is constructed at the higher level – nodes result from primitives, arcs describe the mutual primitive relations.
  - **Region neighborhood graphs** represents every region as a graph node, and nodes of neighboring regions are connected by edges. The **region adjacency graph** is a special case of the region neighborhood graph.

- **Shape classes**
  - Shape classes represent the generic shapes of the objects belonging to the class and emphasize shape differences among classes.
  - A widely used representation of in-class shape variations is determination of class-specific regions in the feature space.

6.6 Exercises

Short-answer questions

1. What are the prerequisites of shape description?
2. What are the main distinguishing aspects among various shape representation and shape description methods?
3. Explain how the problem of scale affects shape description.
4. Explain what shape classes are and why are they important.
5. Explain the rationale behind projection-invariant shape descriptors.
6. Define the three most common representations of region borders.
7. Define the boundary chain code in 4- and 8-connectivity.
8. Define the chain code derivative in 4- and 8-connectivity.
9. Define the following border-based region descriptors:
   (a) boundary length
   (b) curvature
   (c) bending energy
   (d) signature
   (e) chord distribution
   (f) Fourier transform of boundaries using T_n descriptors

(g) Fourier transform of boundaries using S_n descriptors

(h) polygonal segment representation

(i) constant curvature representation

(j) tolerance interval representation

10. Explain the concept of multiscale curve description using interval trees in the scale-space.

11. Describe the concept of B-spline curve interpolation.

12. Explain the role of the spline order.

13. Define the following projection-invariant shape descriptors:

(a) cross ratio

(b) system of four coplanar concurrent lines

(c) system of five coplanar concurrent lines

(d) system of five coplanar points

(e) plane conics

14. Describe a subclass of boundary shapes to which the invariants listed in Question 6.13 can be applied.

15. Describe how differential invariants can help with invariant description of general shapes.

16. Explain the difference between local and global invariants.

17. Define the following region shape descriptors:

(a) area

(b) Euler's number

(c) horizontal and vertical projections

(d) eccentricity

(e) elongatedness

(f) rectangularity

(g) direction

(h) compactness

(i) statistical moments

(j) convex hull

(k) region concavity tree

18. List the advantages of graph-based region shape descriptors.

19. Describe the principles of region skeletonization by thinning.

20. Describe the medial axis transform.

21. Describe the principles of shape description using graph decomposition.

Problems

1. Write a function (subroutine) for region identification in 4-neighborhood connectivity.
2. Write a function (subroutine) for region identification in 8-neighborhood connectivity.
3. Develop a program for region identification and region counting that will use function(s) developed in Problems 6.1 and 6.2. Test on binary segmented images.
4. Modify the program developed in Problem 6.3 to accept multi-level segmented images, assuming that the background gray level is known.
5. Develop a program for region identification and region counting in run-length encoded image data. Use the program developed in Problem 3.2 to generate run-length encoded image data.
6. Develop a program for region identification and region counting in quadtrees. Use the program developed in Problem 3.5 to generate quadtree image data.
7. Write a function (subroutine) for chain code generation in 4-connectivity. Test on images in which the regions have been identified using one of the programs developed in Problems 6.3–6.6.
8. Write a function (subroutine) for chain code generation in 8-connectivity. Test on images in which the regions have been identified using one of the programs developed in Problems 6.3–6.6.
9. An object is described by the following chain code in 4-connectivity: 10123230.
   (a) Determine the normalized version of the chain code.
   (b) Determine the derivative of the original chain code.
10. Determine the Euler number of the following characters: 0, 4, 8, A, B, C, D.
11. Prove the statement that the most compact region in a Euclidean space is a circle. Compare the compactness values of a square and a rectangle of any aspect ratio – what can you conclude?
12. Write functions (subroutines) determining the following border descriptors:
    (a) boundary length
    (b) curvature
    (c) bending energy
    (d) signature
    (e) chord distribution
    (f) Fourier transform of boundaries using T_n descriptors
    (g) Fourier transform of boundaries using S_n descriptors

    Use the functions in a program to determine shape features of binary objects.
13. Write functions (subroutines) determining the following region shape descriptors:
    (a) area
    (b) area from chain code border representation
    (c) area from quadtree region representation
    (d) Euler's number
    (e) horizontal and vertical projections
    (f) eccentricity
    (g) elongatedness

(h) rectangularity

(i) direction

(j) compactness

(k) affine transform invariant statistical moments

Use the functions in a program to determine shape features of binary objects.

14. Develop a program to determine the shape features listed in Problems 6.12–6.13 in images containing several objects simultaneously. The program should report the features in a table, and the objects should be identified by their centroid co-ordinates.

15. Develop a program to determine the shape features listed in Problems 6.12–6.13 from shapes encoded using run-length code and/or quadtree image data.

16. Develop a program to generate digital images of simple shapes (rectangle, diamond, circle, etc.) in various sizes and rotations. Using the functions prepared in Problems 6.12–6.13, compare the shape features determined by individual shape descriptors as a function of size and a function of rotation.

17. Develop a program for simple polygon convex hull detection.

18. Develop a program for region concavity tree construction.

19. Determine a medial axis skeleton of a circle, square, rectangle, and triangle.

20. Develop a program constructing a medial axis of a binary region.

    (a) Apply the program to computer-generated letters and numerals.

    (b) Apply the program to printed letters and numerals after their digitization using a video camera or a scanner.

    (c) Explain the differences in performance of your algorithm.

    (d) Develop a practically-applicable thinning algorithm that constructs line shapes from scanned characters.

6.7 References

[Ansari and Delp 90] N Ansari and E J Delp. Distribution of a deforming triangle. *Pattern Recognition*, 23(12):1333–1341, 1990.

[Appel and Haken 77] K Appel and W Haken. Every planar map is four colourable: Part I: discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.

[Arcelli and Sanniti di Baja 86] C Arcelli and G Sanniti di Baja. Endoskeleton and exoskeleton of digital figures, an effective procedure. In V Cappellini and R Marconi, editors, *Advances in Image Processing and Pattern Recognition*, pages 224–228, North Holland, Amsterdam, 1986.

[Asada and Brady 86] H Asada and M Brady. The curvature primal sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):2–14, 1986.

[Atkinson et al. 85] H H Atkinson, Gargantini, I, and T R S Walsh. Counting regions, holes and their nesting level in time proportional to the border. *Computer Vision, Graphics, and Image Processing*, 29:196–215, 1985.

[Babaud et al. 86] J Babaud, A P Witkin, M Baudin, and R O Duda. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):26–33, 1986.

[Ballard and Brown 82] D H Ballard and C M Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Barnsley 88] M F Barnsley. *Fractals Everywhere*. Academic Press, Boston, Ma, 1988.

[Besl 88] P J Besl. Geometric modelling and computer vision. *Proceedings of the IEEE*, 76:936–958, 1988.

[Bhanu and Faugeras 84] B Bhanu and O D Faugeras. Shape matching of two–dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):137–155, 1984.

[Bhattacharya and Gindy 84] B K Bhattacharya and H E Gindy. A new linear convex hull algorithm for simple polygons. *IEEE Transactions on Information Theory*, 30:85–88, 1984.

[Bhattacharya and Toussaint 83] B K Bhattacharya and G T Toussaint. Time-and-storage-efficient implementation of an optimal planar convex hull algorithm. *Image and Vision Computing*, 1(3):140–144, 1983.

[Blum 73] H Blum. Biological shape and visual science (part 1). *J. Theor. Biol.*, 38:205–287, 1973.

[Bookstein 91] F L Bookstein. *Morphometric Tools for Landmark Data*. Cambridge University Press, Cambridge, UK, 1991.

[Brady 84] M Brady. Representing shape. In M Brady, L A Gerhardt, and H F Davidson, editors, *Robotics and Artificial Intelligence*, pages 279–300. Springer + NATO, Berlin, 1984.

[Brandt and Algazi 92] J W Brandt and V R Algazi. Continuous skeleton computation by Voronoi diagram. *CVGIP – Image Understanding*, 55(3), 1992.

[Bribiesca and Guzman 80] E Bribiesca and A Guzman. How to describe pure form and how to measure differences in shapes using shape numbers. *Pattern Recognition*, 12(2):101–112, 1980.

[Brill et al. 92] M H Brill, E B Barrett, and P M Payton. Projective invariants for curves in two and three dimensions. In J L Mundy and A Zisserman, editors, *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Ma; London, 1992.

[Canny 83] J F Canny. Finding edges and lines in images. Technical Report AI-TR-720, MIT, Artificial Intelligence Lab., Cambridge, Ma, 1983.

[Cash and Hatamian 87] G L Cash and M Hatamian. Optical character recognition by the method of moments. *Computer Vision, Graphics, and Image Processing*, 39:291–310, 1987.

[Chang and Chatterjee 89] C Chang and S Chatterjee. Fractal based approach to shape description, reconstruction and classification. In *Twenty-Third Annual Asilomar Conference on Signals, Systems and Computers, Pacific Grove, Ca*, pages 172–176, IEEE, Los Alamitos, Ca, 1989.

[Chien and Aggarwal 89] C H Chien and J K Aggarwal. Model construction and shape recognition from occluding contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4):372–389, 1989.

[Cinque and Lombardi 95] L Cinque and L Lombardi. Shape description and recognition by a multiresolution approach. *Image and Vision Computing*, 13:599–607, 1995.

[Cootes et al. 92] T F Cootes, D H Cooper, C J Taylor, and J Graham. Trainable method of parametric shape description. *Image and Vision Computing*, 10(5), 1992.

[Cortopassi and Rearick 88] P P Cortopassi and T C Rearick. Computationally efficient algorithm for shape decomposition. In *Proceedings - CVPR '88: Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, Mi*, pages 597–601, IEEE, Los Alamitos, Ca, 1988.

[Costabile et al. 85] M F Costabile, C Guerra, and G G Pieroni. Matching shapes: A case study in time–varying images. *Computer Vision, Graphics, and Image Processing*, 29:296–310, 1985.

[Crowley 84] J L Crowley. A multiresolution representation for shape. In A Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 169–189. Springer Verlag, Berlin, 1984.

[DeBoor 78] C A DeBoor. *A Practical Guide to Splines*. Springer Verlag, New York, 1978.

[di Baja and Thiel 94] G Sanniti di Baja and E Thiel. Shape description via weighted skeleton partition. In *Proceedings of the 7th Int. Conf. Image Analysis and Processing. Progress in Image Analysis and Processing III*, pages 87–94, 1994.

[Duda and Hart 73] R O Duda and P E Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.

[Dyer 80] C R Dyer. Computing the Euler number of an image from its quadtree. *Computer Graphics and Image Processing*, 13:270–276, 1980.

[Eichmann et al. 90] G Eichmann, C Lu, M Jankowski, and R Tolimeiri. Shape representation by Gabor expansion. In *Hybrid Image and Signal Processing II, Orlando, Fl*, pages 86–94, Society for Optical Engineering, Bellingham, Wa, 1990.

[Falconer 90] K Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, Chichester, New York, 1990.

[Feng and Pavlidis 75] H Y Feng and T Pavlidis. Decomposition of polygons into simpler components. *IEEE Transactions on Computers*, 24:636–650, 1975.

[Fermuller and Kropatsch 92] C Fermuller and W Kropatsch. Multi-resolution shape description by corners. In *Proceedings, 1992 Computer Vision and Pattern Recognition, Champaign, Il*, pages 271–276, IEEE, Los Alamitos, Ca, 1992.

[Florack et al. 92] L M J Florack, B M Haar-Romeny, J J Koenderink, and M A Viergever. Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388, 1992.

[Flusser and Suk 91] J Flusser and T Suk. Classification of objects by affine moment invariants. Technical Report UTIA-1736, Czechoslovak Academy of Sciences, Prague, 1991.

[Flusser and Suk 93] J Flusser and T Suk. Pattern recognition by affine moment invariants. *Pattern Recognition*, 26:167–174, 1993.

[Forsyth et al. 91] D Forsyth, J L Mundy, A Zisserman, C Coelho, A Heller, and C Rothwell. Invariant descriptors for 3D object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, 1991.

[Freeman 61] H Freeman. On the encoding of arbitrary geometric configuration. *IRE Transactions on Electronic Computers*, EC–10(2):260–268, 1961.

[Frisch et al. 87] A A Frisch, D A Evans, J P Hudson, and J Boon. Shape discrimination of sand samples using the fractal dimension. In *Coastal Sediments '87, Proceedings of a Specialty Conference on Advances in Understanding of Coastal Sediment Processes, New Orleans, LA*, pages 138–153, ASCE, Dallas, Tx, 1987.

[Fritsch et al. 97] D Fritsch, S Pizer, L Yu, V Johnson, and E Chaney. Segmentation of medical image objects using deformable shape loci. In J Duncan and G Gindi, editors, *Information Processing in Medical Imaging*, pages 127–140, Springer Verlag, Berlin, New York, 1997.

[Fu 74] K S Fu. *Syntactic Methods in Pattern Recognition*. Academic Press, New York, 1974.

[Gauch and Pizer 93] J M Gauch and S M Pizer. The intensity axis of symmetry and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:753–770, 1993.

[Graham and Yao 84] R L Graham and F F Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4:324–331, 1984.

[Griffin et al. 92] L D Griffin, A C F Colchester, and G P Robinson. Scale and segmentation of grey-level images using maximum gradient paths. *Image and Vision Computing*, 10(6):389–402, 1992.

[Grimmins 82] T R Grimmins. A complete set of Fourier descriptors for two–dimensional shapes. *IEEE Transactions on Systems, Man and Cybernetics*, 12(6):923–927, 1982.

[Gross and Latecki 95] A Gross and L Latecki. Digital geometric invariance and shape representation. In *Proceedings Int. Symp. Comp. Vision*, pages 121–126. IEEE, 1995.

[Guo and Hall 92] Z Guo and R W Hall. Fast fully parallel thinning algorithms. *CVGIP – Image Understanding*, 55(3):317–328, 1992.

[Gupta and Srinath 87] L Gupta and M D Srinath. Contour sequence moments for the classification of closed planar shapes. *Pattern Recognition*, 20(3):267–272, 1987.

[Gupta et al. 90] L Gupta, M R Sayeh, and R Tammana. Neural network approach to robust shape classification. *Pattern Recognition*, 23(6):563–568, 1990.

[Held and Abe 94] A Held and K Abe. On the decomposition of binary shapes into meaningful parts. *Pattern Recognition*, 27:637–647, 1994.

[Hildich 69] C J Hildich. Linear skeletons from square cupboards. In B Meltzer and D Michie, editors, *Machine Intelligence IV*, pages 403–420. Elsevier, New York, 1969.

[Hlavac et al. 94] V Hlavac, T Pajdla, and M Sommer. Improvements of the curvature computation. In *International Conference on Pattern Recognition*, pages 536–538, IEEE, Los Alamitos, CA, 1994.

[Hogg 93] D C Hogg. Shape in machine vision. *Image and Vision Computing*, 11:309–316, 1993.

[Hopcroft et al. 92] J E Hopcroft, D P Huttenlocher, and P C Wayner. Affine invariants for model-based recognition. In J L Mundy and A Zisserman, editors, *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Ma; London, 1992.

[Hu 62] M K Hu. Visual pattern recognition by moment invariants. *IRE Transactions Information Theory*, 8(2):179–187, 1962.

[Ikebe and Miyamoto 82] Y Ikebe and S Miyamoto. Shape design, representation, and restoration with splines. In K S Fu and H Kunii, editors, *Picture Engineering*. Springer Verlag, Berlin, 1982.

[Jain 89] A K Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Jakubowski 85] R Jakubowski. Extraction of shape features for syntactic recognition of mechanical parts. *IEEE Transactions on Systems, Man and Cybernetics*, 15(5):642–651, 1985.

[Jakubowski 90] R Jakubowski. Decomposition of complex shapes for their structural recognition. *Information Sciences*, 50(1):35–71, 1990.

[Ji and Haralick 97] Q Ji and R M Haralick. Corner detection with covariance propagation. In *Computer Vision and Pattern Recognition*, pages 362–367, IEEE Computer Society, Los Alamitos, CA, 1997.

[Jiang and Bunke 91] X Y Jiang and H Bunke. Simple and fast computation of moments. *Pattern Recognition*, 24:801–806, 1991.

[Juday 88] R D Juday, editor. *Digital and Optical Shape Representation and Pattern Recognition, Orlando, Fl*, Bellingham, Wa, 1988. SPIE.

[Kanatani 90] K Kanatani. *Group-Theoretical Methods in Image Understanding.* Springer Verlag, Berlin, 1990.

[Kanungo and Haralick 92] T Kanungo and R M Haralick. Vector-space solution for a morphological shape-decomposition problem. *J. Math. Imag. and Vision*, 2:51–82, 1992.

[Kiryati and Maydan 89] N Kiryati and D Maydan. Calculating geometric properties from Foutier representation. *Pattern Recognition*, 22(5):469–475, 1989.

[Koch and Kashyap 87] M W Koch and R L Kashyap. Using polygons to recognize and locate partially occluded objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):483–494, 1987.

[Koenderink 90] J J Koenderink. *Solid Shape.* MIT Press, Cambridge Ma, 1990.

[Koenderink and van Doorn 86] J J Koenderink and A J van Doorn. Dynamic shape. Technical report, Dept. Medical and Physiological Physics, State University, Utrecht, The Netherlands, 1986.

[Krishnapuram and Chen 91] R Krishnapuram and L F Chen. Iterative neural networks for skeletonization and thinning. In *Intelligent Robots and Computer Vision IX: Neural, Biological, and 3D Methods, Boston, Ma*, pages 271–281, Soc for Optical Engineering, Bellingham, Wa, 1991.

[Krzyzak et al. 89] A Krzyzak, S Y Leung, and C Y Suen. Reconstruction of two-dimensional patterns from Fourier descriptors. *Machine Vision and Applications*, 2(3):123–140, 1989.

[Lam et al. 92] L Lam, S W Lee, and C Y Suen. Thinning methodologies - a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.

[Lee 83] D T Lee. On finding the convex hull of a simple polygon. *International Journal of Computer and Information Sciences*, 12:87–98, 1983.

[Leu 89] J G Leu. View-independent shape representation and matching. In *IEEE International Conference on Systems Engineering, Fairborn, Oh*, pages 601–604, IEEE, Piscataway, NJ, 1989.

[Leymarie and Levine 89] F Leymarie and M D Levine. Shape features using curvature morphology. In *Visual Communications and Image Processing IV, Philadelphia, Pa*, pages 390–401, SPIE, Bellingham, Wa, 1989.

[Li and Ma 94] B Li and S D Ma. On the relation between region and contour representation. In *International Conference on Pattern Recognition*, pages 352–355, IEEE, Los Alamitos, CA, 1994.

[Li and Shen 91] B C Li and J Shen. Fast computation of moment invariants. *Pattern Recognition*, 24:807–813, 1991.

[Li and Zhiying 88] X Li and Z Zhiying. Group direction difference chain codes for the representation of the border. In *Digital and Optical Shape Representation and Pattern Recognition, Orlando, Fl*, pages 372–376, SPIE, Bellingham, Wa, 1988.

[Lin and Chellappa 87] C C Lin and R Chellappa. Classification of partial 2D shapes using Fourier descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):686–690, 1987.

[Lindenbaum and Bruckstein 93] M Lindenbaum and A Bruckstein. On recursive, o(n) partitioning of a digitized curve into digital straight segments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:949–953, 1993.

[Lipari and Harlow 88] C Lipari and C A Harlow. Representation and recognition of elongated regions in aerial images. In *Applications of Artificial Intelligence VI, Orlando, Fl*, pages 557–567, SPIE, Bellingham, Wa, 1988.

[Loncaric and Dhawan 93] S Loncaric and A P Dhawan. A morphological signature transform for shape description. *Pattern Recognition*, 26:1029–1037, 1993.

[Loncaric and Dhawan 95] S Loncaric and A P Dhawan. Near-optimal MST-based shape description using genetic algorithm. *Pattern Recognition*, 28:571–579, 1995.

[Lord and Wilson 84] E A Lord and C B Wilson. *The Mathematical Description of Shape and Form*. Halsted Press, Chichester, UK, 1984.

[Loui et al. 90] A C P Loui, A N Venetsanopoulos, and K C Smith. Two-dimensional shape representation using morphological correlation functions. In *Proceedings of the 1990 International Conference on Acoustics, Speech, and Signal Processing - ICASSP 90, Albuquerque, NM*, pages 2165–2168, IEEE, Piscataway, NJ, 1990.

[Lowe 89] D G Lowe. Organization of smooth image curves at multiple scales. *International Journal of Computer Vision*, 1:119–130, 1989.

[Maitra 79] S Maitra. Moment invariants. *Proceedings IEEE*, 67(4):697–699, 1979.

[Mandelbrot 82] B B Mandelbrot. *The Fractal Geometry of Nature*. Freeman, New York, 1982.

[Maragos 89] P Maragos. Pattern spectrum and multiscale shape representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:701–716, 1989.

[Maragos and Schafer 86] P A Maragos and R W Schafer. Morphological skeleton representation and coding of binary images. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(5):1228–1244, 1986.

[Marshall 89a] S Marshall. Application of image contours to three aspects of image processing; compression, shape recognition and stereopsis. In *Third International Conference on Image Processing and its Applications, Coventry, England*, pages 604–608, IEE, Michael Faraday House, Stevenage, England, 1989.

[Marshall 89b] S Marshall. Review of shape coding techniques. *Image and Vision Computing*, 7(4):281–194, 1989.

[Matas and Kittler 93] J Matas and J Kittler. Junction detection using probabilistic relaxation. *Image and Vision Computing*, 11:197–202, 1993.

[Maybank 92] S J Maybank. The projection of two non-coplanar conics. In J L Mundy and A Zisserman, editors, *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Ma; London, 1992.

[Mayya and Rajan 95] N Mayya and V T Rajan. An efficient shape representation scheme using voronoi skeletons. *Pattern Recognition Letters*, 16:147–160, 1995.

[McCallum and Avis 79] D McCallum and D Avis. A linear algorithm for finding the convex hull of a simple polygon. *Information Processing Letters*, 9:201–206, 1979.

[McKenzie and Protheroe 90] D S McKenzie and S R Protheroe. Curve description using the inverse Hough transform. *Pattern Recognition*, 23(3-4):283–290, 1990.

[Melkman 87] A V Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, 1987.

[Minnix et al. 90] J I Minnix, E S McVey, and R M Inigo. Multistaged neural network architecture for position invariant shape recognition. In *Visual Communications and Image Processing '90, Lausanne, Switzerland*, pages 58–68, SPIE, Bellingham, Wa, 1990.

[Moayer and Fu 75] B Moayer and K S Fu. A tree system approach for fingerprint pattern recognition. *IEEE Transactions on Computers*, 24(4):436–450, 1975.

[Morse et al. 93] B S Morse, S M Pizer, and A Liu. Multiscale medial analysis of medical images. In Barrett and Gmitro, editors, *Information Processing in Medical Imaging*, pages 112–131. Springer Verlag, Berlin, 1993.

[Mundy and Zisserman 92] J L Mundy and A Zisserman. *Geometric Invariance in Computer Vision.* MIT Press, Cambridge, Ma; London, 1992.

[Nagao and Matsuyama 80] M Nagao and T Matsuyama. *A Structural Analysis of Complex Aerial Photographs.* Plenum Press, New York, 1980.

[Nishizeki and Chiba 88] T Nishizeki and N Chiba. *Planar Graphs: Theory and Algorithms.* North Holland, Amsterdam-New York-Tokyo, 1988.

[Ogniewicz and Ilg 92] R Ogniewicz and M Ilg. Voronoi skeletons: Theory and applications. In *Proceedings, 1992 Computer Vision and Pattern Recognition, Champaign, Il*, pages 63–69, IEEE, Los Alamitos, Ca, 1992.

[Oppenheim et al. 83] A V Oppenheim, A S Willsky, and I T Young. *Signals and Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[Paglieroni and Jain 88] D W Paglieroni and A K Jain. Control point transforms for shape representation and measurement. *Computer Vision, Graphics, and Image Processing*, 42(1):87–111, 1988.

[Papoulis 91] A Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw Hill, New York, 3rd edition, 1991.

[Pavlidis 77] T Pavlidis. *Structural Pattern Recognition.* Springer Verlag, Berlin, 1977.

[Pavlidis 78] T Pavlidis. A review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7:243–258, 1978.

[Pavlidis 80] T Pavlidis. Algorithms for shape analysis of contours and waveforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):301–312, 1980.

[Persoon and Fu 77] E Persoon and K S Fu. Shape discrimination using Fourier descriptors. *IEEE Transactions on Systems, Man and Cybernetics*, 7:170–179, 1977.

[Petrou 90] M Petrou. Optimal convolution filters and an algorithm for the detection of linear features. Technical Report Dept. of Electronic and Electrical Eng., University of Surrey, United Kingdom, 1990.

[Pitas and Venetsanopoulos 90] I Pitas and A N Venetsanopoulos. Morphological shape decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):38–45, 1990.

[Pizer et al. 87] S M Pizer, W R Oliver, and S H Bloomberg. Hierarchical shape description via the multiresolution symmetric axis transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):505–511, 1987.

[Pratt 91] W K Pratt. *Digital Image Processing.* John Wiley and Sons, New York, 2nd edition, 1991.

[Quan et al. 92] L Quan, P Gros, and R Mohr. Invariants of a pair of conics revisited. *Image and Vision Computing*, 10(5):319–323, 1992.

[Rangarajan et al. 97] A Rangarajan, H Chui, and F L Bookstein. The Softassign Procrustes matching algorithm. In J Duncan and G Gindi, editors, *Information Processing in Medical Imaging*, pages 29–42, Springer Verlag, Berlin, New York, 1997.

[Reinhardt and Higgins 96] J M Reinhardt and W E Higgins. Efficient morphological shape representation. *IEEE Transactions on Image Processing*, 5:89–101, 1996.

[Reiss 93] T H Reiss. *Recognizing Planar Objects using Invariant Image Features*. Springer Verlag, Berlin; New York, 1993.

[Rom and Medioni 92] H Rom and G Medioni. Hierarchical decomposition and axial shape description. In *Proceedings, 1992 Computer Vision and Pattern Recognition, Champaign, Il*, pages 49–55, IEEE, Los Alamitos, Ca, 1992.

[Rom and Medioni 93] H Rom and G Medioni. Hierarchical decomposition and axial shape description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:973–981, 1993.

[Rosenfeld 74] A Rosenfeld. Digital straight line segments. *IEEE Transactions on Computers*, 23:1264–1269, 1974.

[Rosenfeld 79] A Rosenfeld. *Picture Languages – Formal Models for Picture Recognition*. Academic Press, New York, 1979.

[Rosenfeld and Kak 82] A Rosenfeld and A C Kak. *Digital Picture Processing*. Academic Press, New York, 2nd edition, 1982.

[Rosin and West 89] P L Rosin and G A W West. Segmentation of edges into lines and arcs. *Image and Vision Computing*, 7(2):109–114, 1989.

[Rothwell et al. 92a] C A Rothwell, A Zisserman, D A Forsyth, and J L Mundy. Fast recognition using algebraic invariants. In J L Mundy and A Zisserman, editors, *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Ma; London, 1992.

[Rothwell et al. 92b] C A Rothwell, A Zisserman, J L Mundy, and D A Forsyth. Efficient model library access by projectively invariant indexing functions. In *Proceedings, 1992 Computer Vision and Pattern Recognition, Champaign, Il*, pages 109–114, IEEE, Los Alamitos, Ca, 1992.

[Saaty and Kainen 77] T L Saaty and P C Kainen. *The Four Colour Problem*. McGraw Hill, New York, 1977.

[Safaee-Rad et al. 89] R Safaee-Rad, B Benhabib, K C Smith, and K M Ty. Position, rotation, and scale-invariant recognition of 2 dimensional objects using a gradient coding scheme. In *IEEE Pacific RIM Conference on Communications, Computers and Signal Processing, Victoria, BC, Canada*, pages 306–311, IEEE, Piscataway, NJ, 1989.

[Samet 81] H Samet. Computing perimeters of images represented by quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:683–687, 1981.

[Samet 84] H Samet. A tutorial on quadtree research. In A Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 212–223. Springer Verlag, Berlin, 1984.

[Samet 85] H Samet. Reconstruction of quadtree medial axis transforms. *Computer Vision, Graphics, and Image Processing*, 29:311–328, 1985.

[Saund 90] E Saund. Symbolic construction of a 2D scale-space image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:817–830, 1990.

[Savini 88] M Savini. Moments in image analysis. *Alta Frequenza*, 57(2):145–152, 1988.

[Shapiro 80] L Shapiro. A structural model of shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(2):111–126, 1980.

[Shariat 90] H Shariat. A model-based method for object recognition. In *IEEE International Conference on Robotics and Automation, Cincinnati, Oh*, pages 1846–1851, IEEE, Los Alamitos, Ca, 1990.

[Shridhar and Badreldin 84] M Shridhar and A Badreldin. High accuracy character recognition algorithms using Fourier and topological descriptors. *Pattern Recognition*, 17(5):515–524, 1984.

[Sklansky 72] J Sklansky. Measuring concavity on a rectangular mosaic. *IEEE Transactions on Computers*, 21(12):1355–1364, 1972.

[Smeulders et al. 80] A W M Smeulders, A M Vossepoel, J Vrolijk, J S Ploem, and C J Cornelisse. Some shape parameters for cell recognition. In *Proceedings of Pattern Recognition in Practice*, pages 131–142, North Holland, Amsterdam, 1980.

[Smith and Jain 82] S Smith and A Jain. Cord distribution for shape matching. *Computer Graphics and Image Processing*, 20:259–265, 1982.

[Staib and Duncan 92] L H Staib and J S Duncan. Boundary finding with parametrically deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1061–1075, 1992.

[Stallings 76] W Stallings. Approaches to Chinese character recognition. *Pattern Recognition*, 8(1):87–98, 1976.

[Strackee and Nagelkerke 83] J Strackee and N J D Nagelkerke. On closing the Fourier descriptor presentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(6):660–661, 1983.

[Tampi and Sridhar 90] K R Tampi and C S Sridhar. Shape detection using word like image description. In *Proceedings of the 1990 International Conference on Acoustics, Speech, and Signal Processing - ICASSP 90 Albuquerque, NM*, pages 2041–2043, IEEE, Piscataway, NJ, 1990.

[Taylor and Lewis 94] R I Taylor and P H Lewis. 2D shape signature based on fractal measurements. *IEE Proceedings-Vision, Image and Signal Processing*, 141:422–430, 1994.

[Tomek 74] I Tomek. Two algorithms for piecewise linear continuous approximation of functions of one variable. *IEEE Transactions on Computers*, 23(4):445–448, 1974.

[Toussaint 85] G Toussaint. A historical note on convex hull finding algorithms. *Pattern Recognition Letters*, 3(1):21–28, 1985.

[Toussaint 91] G Toussaint. A counter-example to a convex hull algorithm for polygons. *Pattern Recognition*, 24(2):183–184, 1991.

[Tsai and Yu 85] W H Tsai and S S Yu. Attributed string matching with merging for shape recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):453–462, 1985.

[Turney et al. 85] J L Turney, T N Mudge, and R A Volz. Recognizing partially occluded parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):410–421, 1985.

[Van Gool et al. 92] L J Van Gool, T Moons, E Pauwels, and A Oosterlinck. Semi-differential invariants. In J L Mundy and A Zisserman, editors, *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Ma; London, 1992.

[Vemuri and Radisavljevic 93] B C Vemuri and A Radisavljevic. From global to local, a continuum of shape models with fractal priors. In *Computer Vision and Pattern Recognition*, pages 307–313, IEEE, Los Alamitos, CA, 1993.

[Vernon 87] D Vernon. Two-dimensional object recognition using partial contours. *Image and Vision Computing*, 5(1):21–27, 1987.

[Wallace 81] T P Wallace. Comments on algorithms for shape analysis of contours and waveforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5), 1981.

[Wallace and Wintz 80] T P Wallace and P A Wintz. An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors. *Computer Graphics and Image Processing*, 13:99–126, 1980.

[Wang et al. 95] D Wang, V Haese-Coat, and J Ronsin. Shape decomposition and representation using a recursive morphological operation. *Pattern Recognition*, 28:1783–1792, 1995.

[Watt 93] R J Watt. Issues in shape perception. *Image and Vision Computing*, 11:389–394, 1993.

[Weiss 88] I Weiss. Projective invariants of shapes. In *Proceedings of Image Understanding Workshop*, volume 2, pages 1125–1134, Cambridge, Ma, 1988.

[Weiss 92] I Weiss. Noise resistant projective and affine invariants. In *Proceedings, 1992 Computer Vision and Pattern Recognition, Champaign, Il*, pages 115–121, IEEE, Los Alamitos, Ca, 1992.

[Weyl 46] H Weyl. *The Classical Groups and Their Invariants*. Princeton University Press, Princeton, NJ, 1946.

[Wilson and Nelson 90] R Wilson and R Nelson. *Graph Colourings*. Longman Scientific and Technical; Wiley, Essex, England and New York, 1990.

[Winston 75] P H Winston, editor. *The Psychology of Computer Vision*. McGraw Hill, New York, 1975.

[Witkin 86] A P Witkin. Scale space filtering. In A P Pentland, editor, *From Pixels to Predicates*, pages 5–19. Ablex Publishing Corporation, Norwood, NJ, 1986.

[Woodwark 86] J Woodwark. *Computing Shape: An Introduction to the Representation of Component and Assembly Geometry for Computer-Aided Engineering*. Butterworths, London-Boston, 1986.

[Wright and Fallside 93] M W Wright and F Fallside. Skeletonisation as model-based feature detection. *IEE Proceedings Communic., Speech and Vision*, 140:7–11, 1993.

[Wuescher and Boyer 91] D M Wuescher and K L Boyer. Robust contour decomposition using a constant curvature criterion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):41–51, 1991.

[Xiaoqi and Baozong 95] Z Xiaoqi and Y Baozong. Shape description and recognition using the high order morphological pattern spectrum. *Pattern Recognition*, 28:1333–1340, 1995.

[Yuille and Poggio 86] A L Yuille and T A Poggio. Scaling theorems for zero-crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):15–25, 1986.

[Zhou and Venetsanopoulos 89] Z Zhou and A N Venetsanopoulos. Generic ribbons: A morphological approach towards natural shape decomposition. In *Visual Communications and Image Processing IV, Philadelphia, Pa*, pages 170–180, Soc for Optical Engineering, Bellingham, Wa, 1989.

[Zhu and Poh 88] Q Zhu and L Poh. Transformation-invariant recursive subdivision method for shape analysis. In *9th International Conference on Pattern Recognition, Rome, Italy*, pages 833–835, IEEE, New York, 1988.