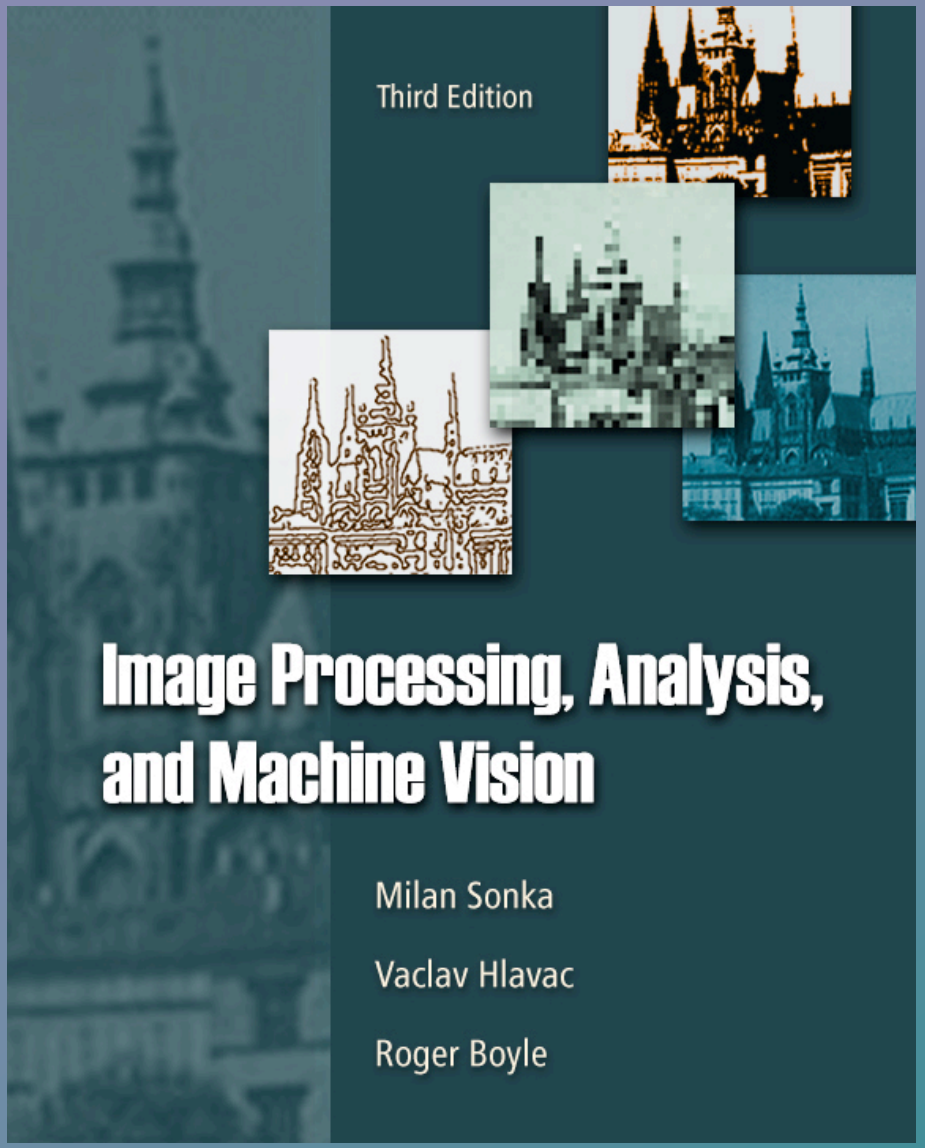


Chapter 8 – Shape representation and description

8.1 Region identification

8.2 Contour-based shape representation

8.3 Region-based shape representation and description

8.4 Shape classes

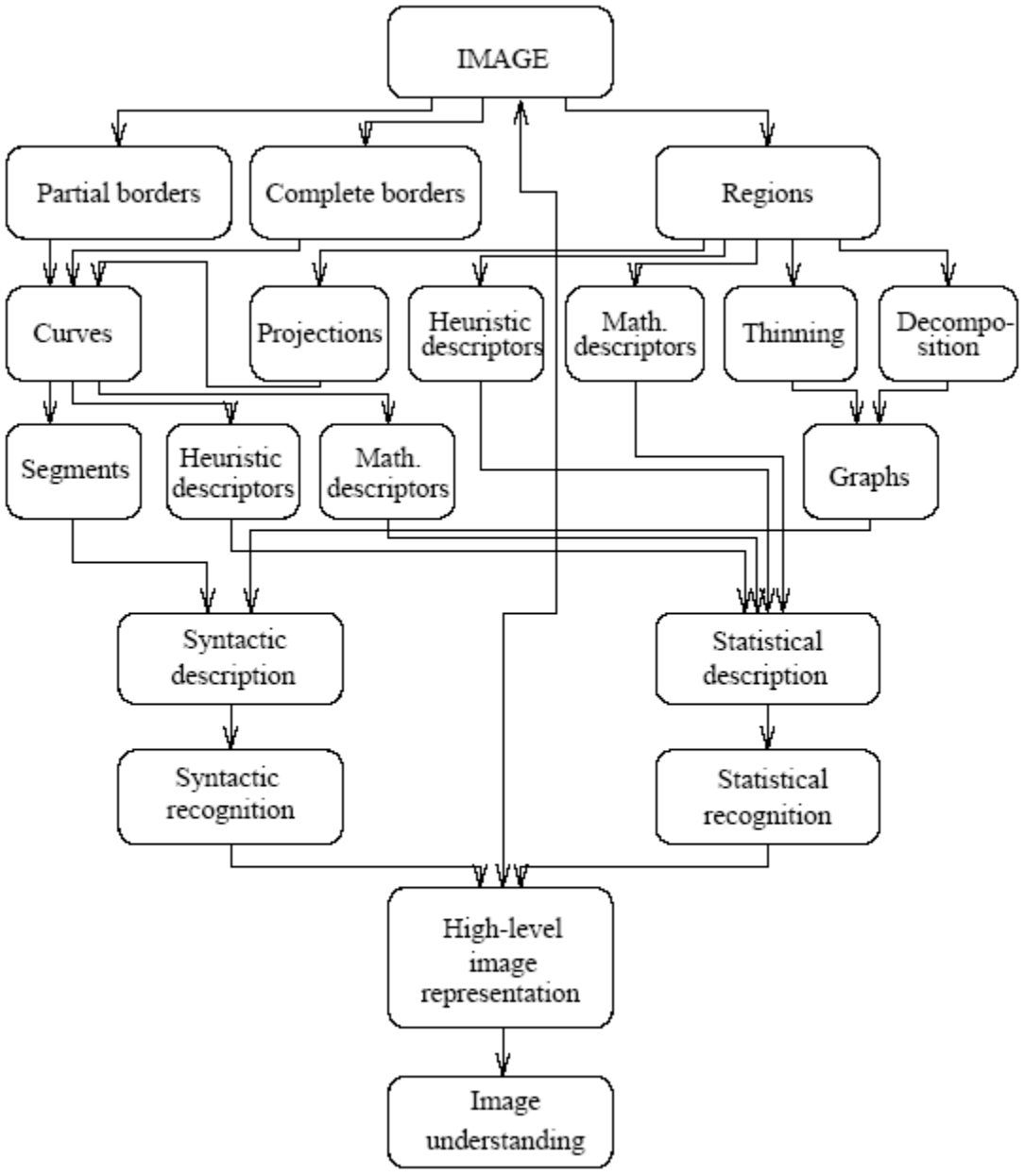


Figure 8.1: Image analysis and understanding methods.

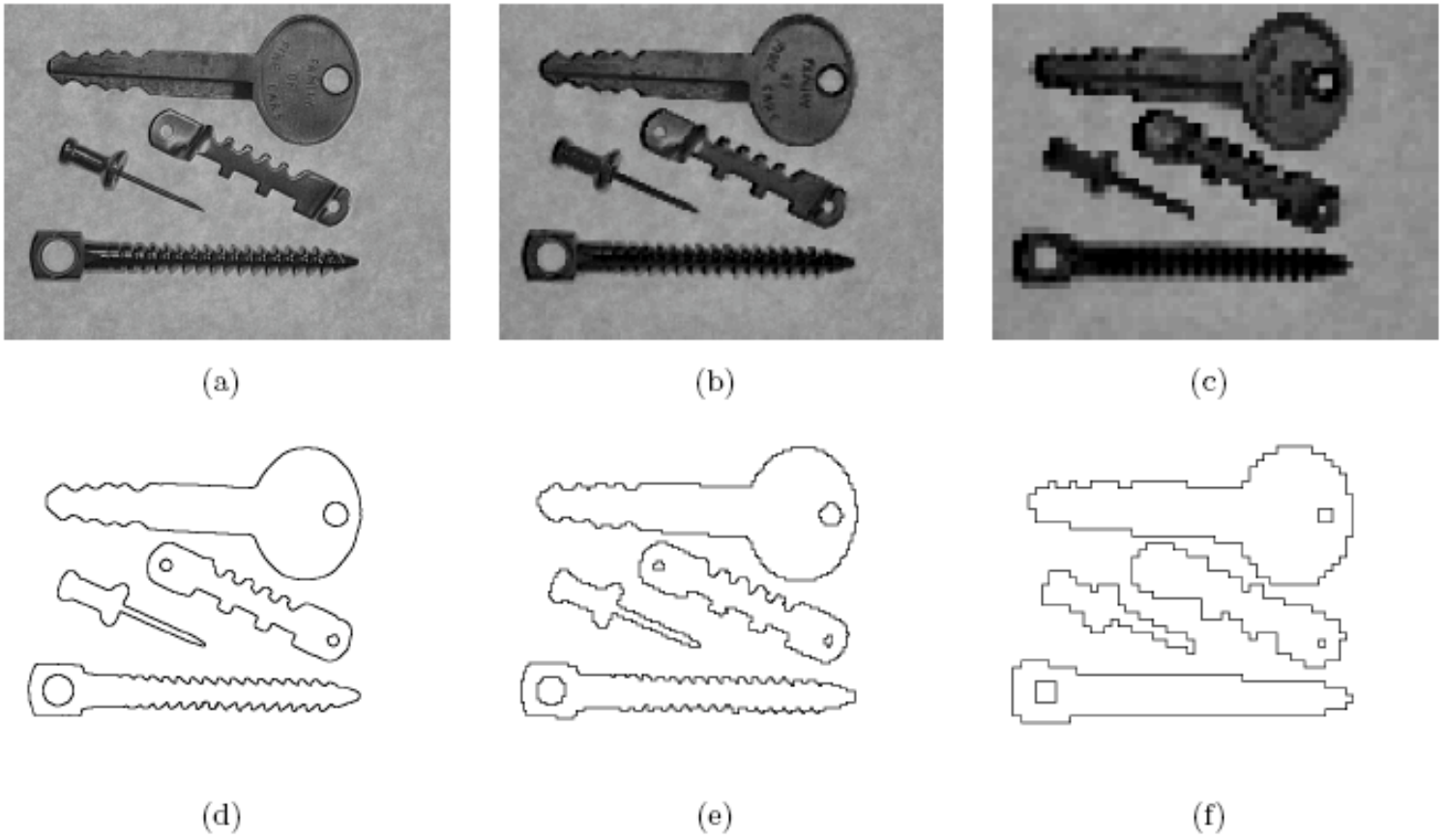


Figure 8.2: (a) Original image 640×480. (d) Contours of (a). (b) Original image 160×120. (e) Contours of (b). (c) Original image 64×48. (f) Contours of (c).

Algorithm 8.1: 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 8.3. ('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 are re-labeled using the equivalence table information (for example, with the lowest value in an equivalence class).

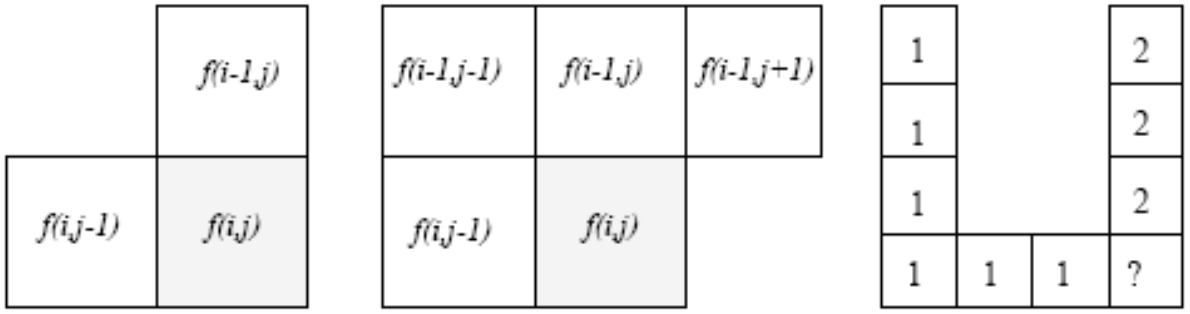
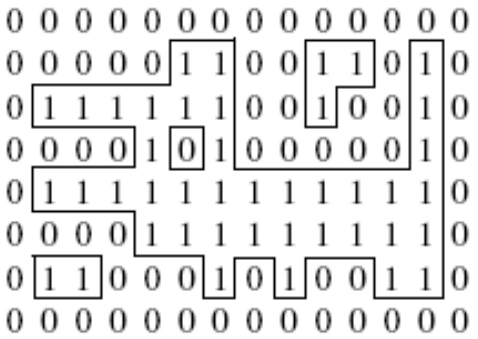


Figure 8.3: Masks for region identification. (a) 4-connectivity. (b) 8-connectivity. (c) Label collision.

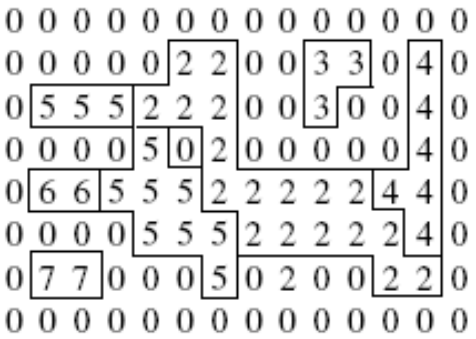
Algorithm 8.2: Region identification in run length encoded data

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.

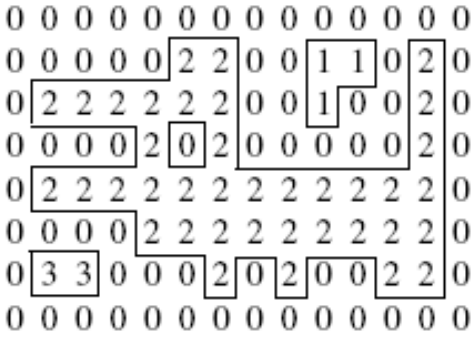
   Collision information is stored in an equivalence table, and the new run is labeled using the label of any one of its neighbors.
3. Second pass: Search the image row by row and re-label the image according to the equivalence table information.



(a)



(b)



(c)

Figure 8.4: Object identification in 8-connectivity. (a), (b), (c) Algorithm steps. Equivalence table after step (b): 2-5, 5-6, 2-4.

Algorithm 8.3: Quadtree region identification

1. First pass: Search quadtree nodes in a given order—e.g., beginning from the root and in the 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.

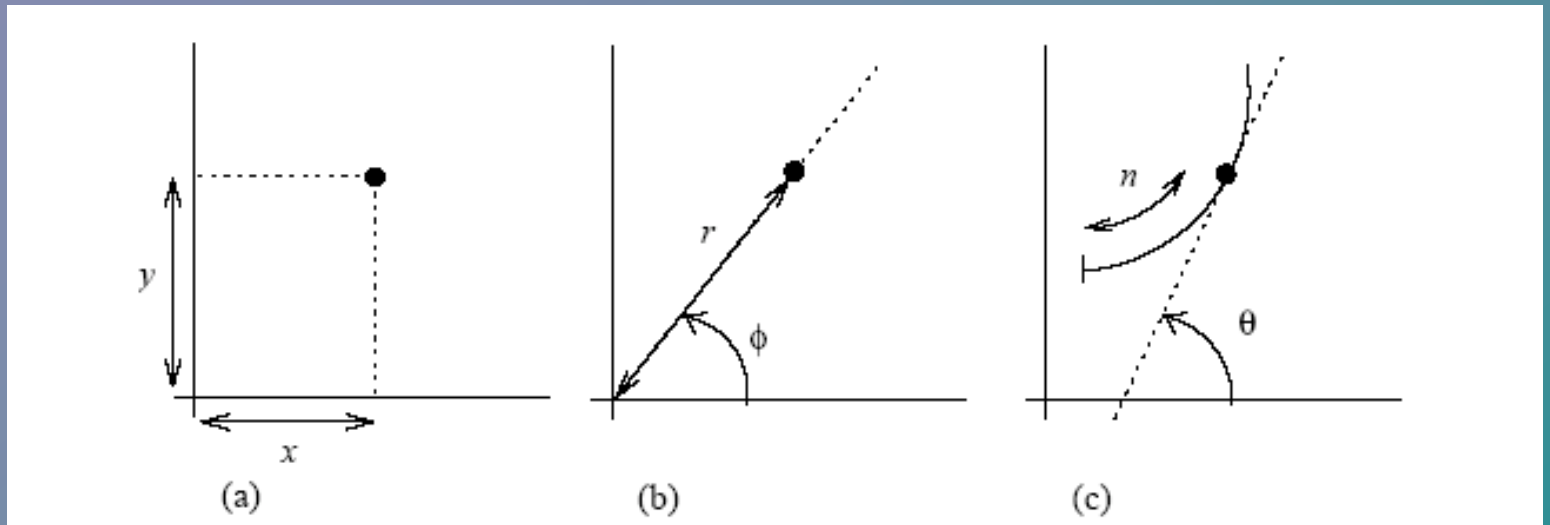


Figure 8.5: Co-ordinate systems. (a) Rectangular (Cartesian). (b) Polar. (c) Tangential.

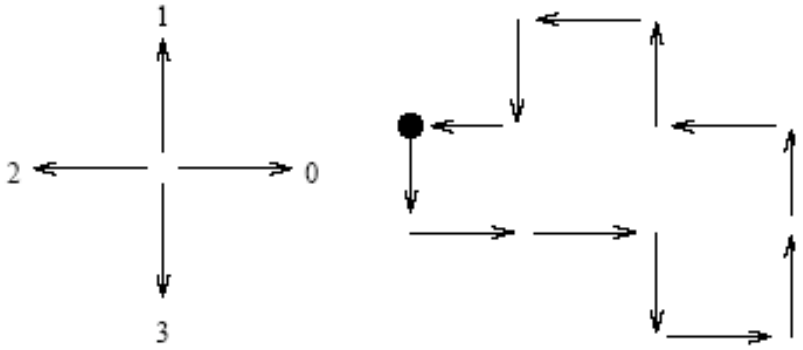


Figure 8.6: Chain code in 4-connectivity, and its derivative. Code: **3**, 0, 0, 3, 0, 1, 1, 2, 1, 2, 3, 2; derivative: 1, 0, 3, 1, 1, 0, 1, 3, 1, 1, **3**, 1.

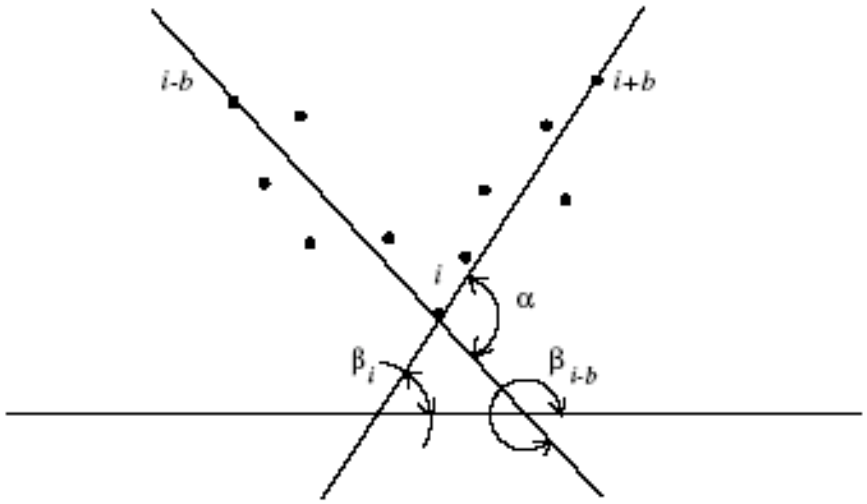


Figure 8.7: Curvature.

$$\mathrm{BE} = \frac{1}{L}\sum_{k=1}^{L} c^2(k)\,. \qquad (8.1)$$

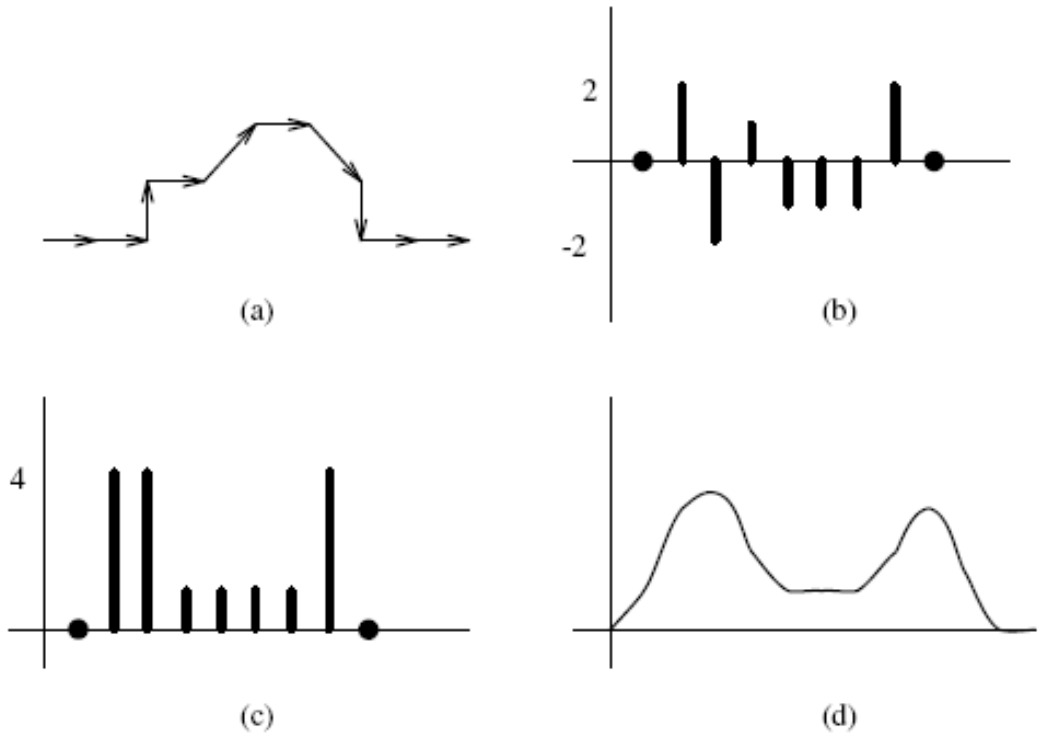


Figure 8.8: Bending energy. (a) Chain code 0, 0, 2, 0, 1, 0, 7, 6, 0, 0. (b) Curvature 0, 2, -2, 1, -1, -1, -1, 2, 0. (c) Sum of squares gives the bending energy. (d) Smoothed version.

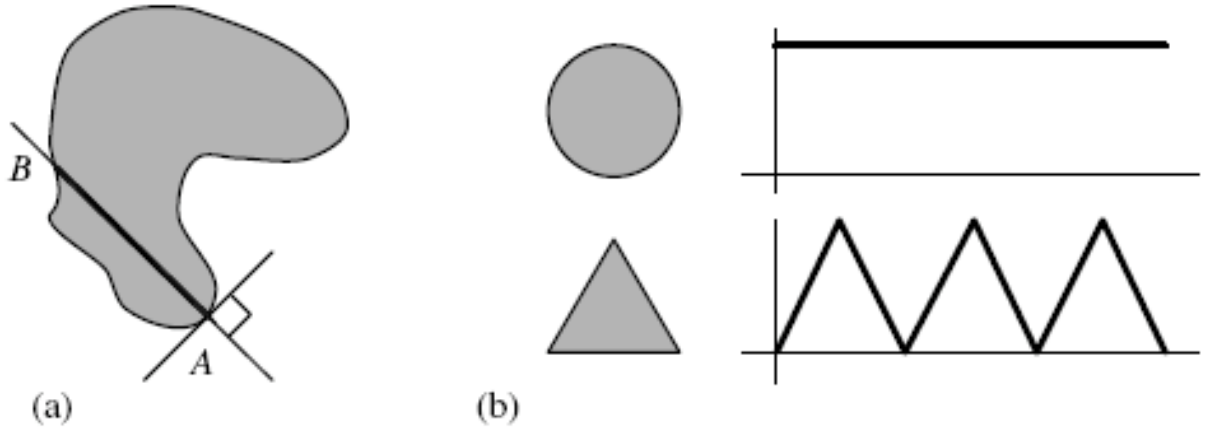


Figure 8.9: Signature. (a) Construction. (b) Signatures for a circle and a triangle.

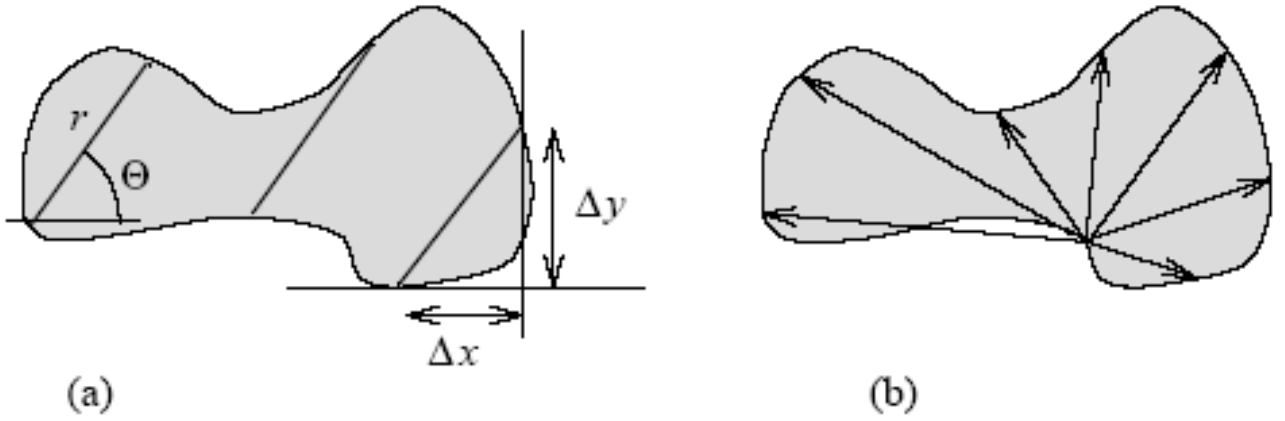


Figure 8.10: Chord distribution.

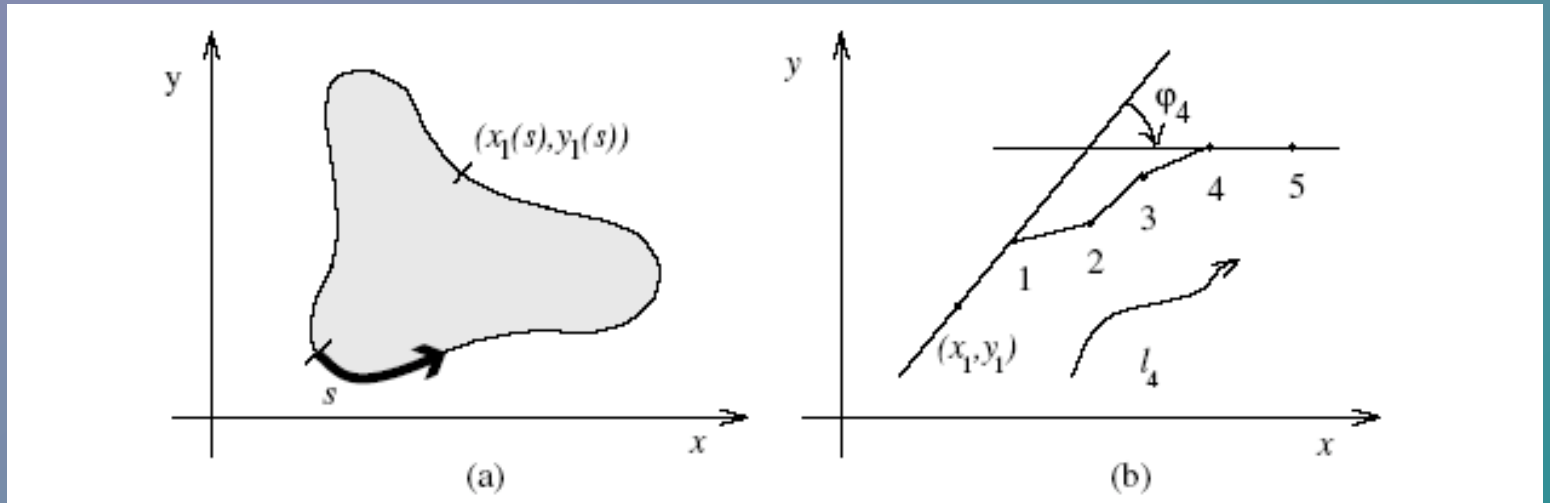


Figure 8.11: Fourier description of boundaries. (a) Descriptors T_n. (b) Descriptors S_n.

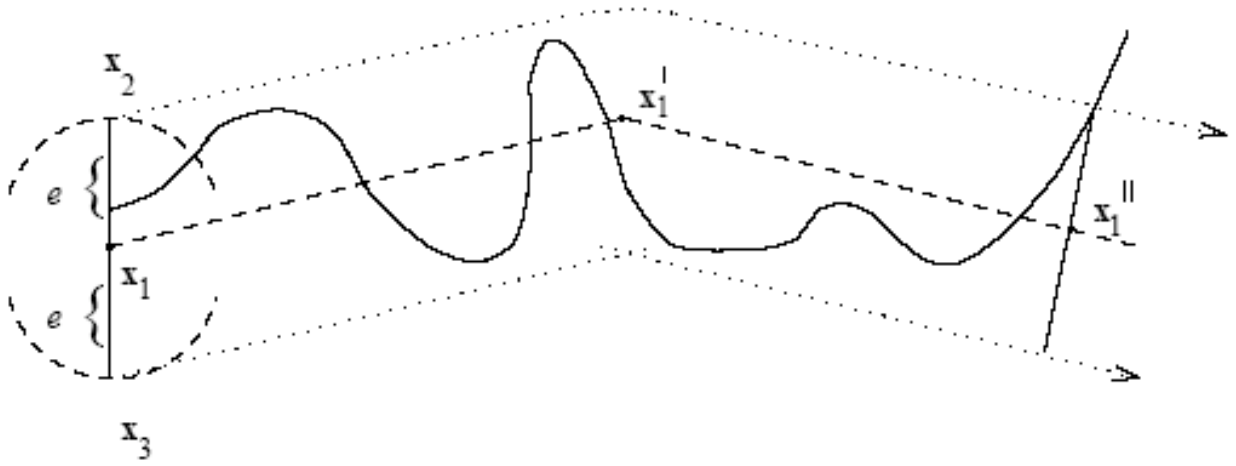


Figure 8.12: Tolerance interval.

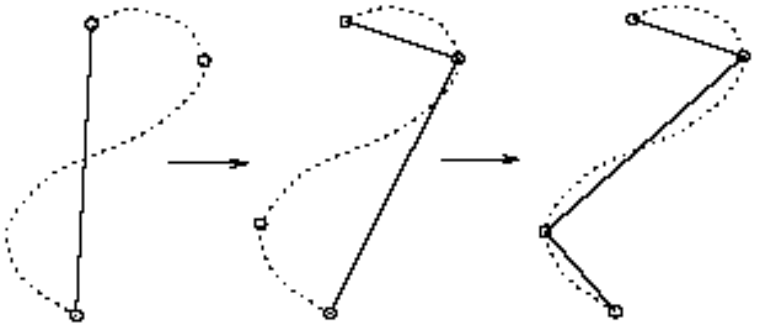


Figure 8.13: Recursive boundary splitting.

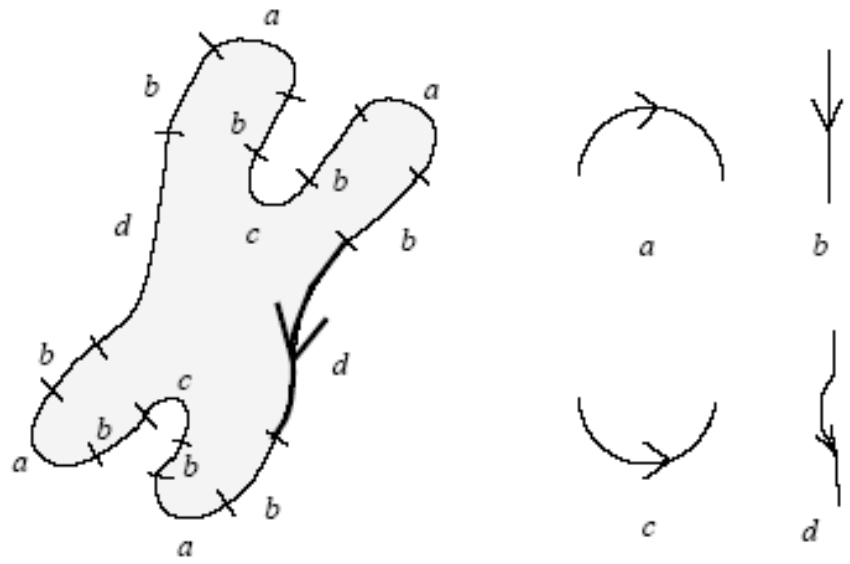


Figure 8.14: Structural description of chromosomes by a chain of boundary segments, code word: d, b, a, b, c, b, a, b, d, b, a, b, c, b, a, b. *Adapted from [Fu, 1974].*

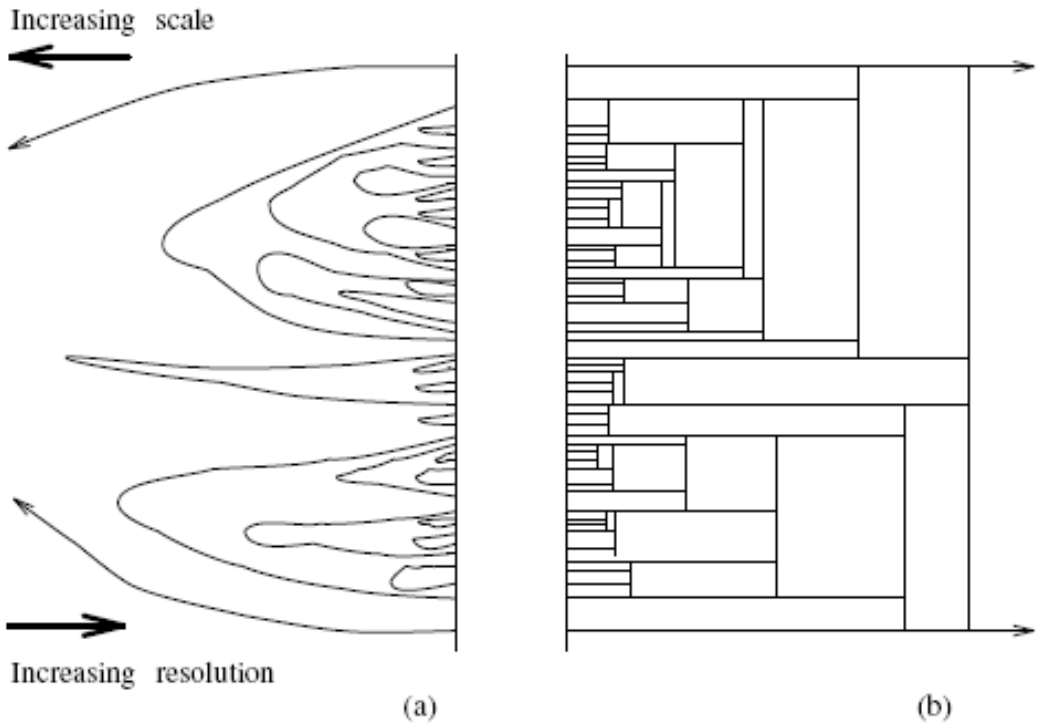


Figure 8.15: Scale-space image. (a) Varying number and locations of curve segmentation points as a function of scale. (b) Curve representation by an interval tree.

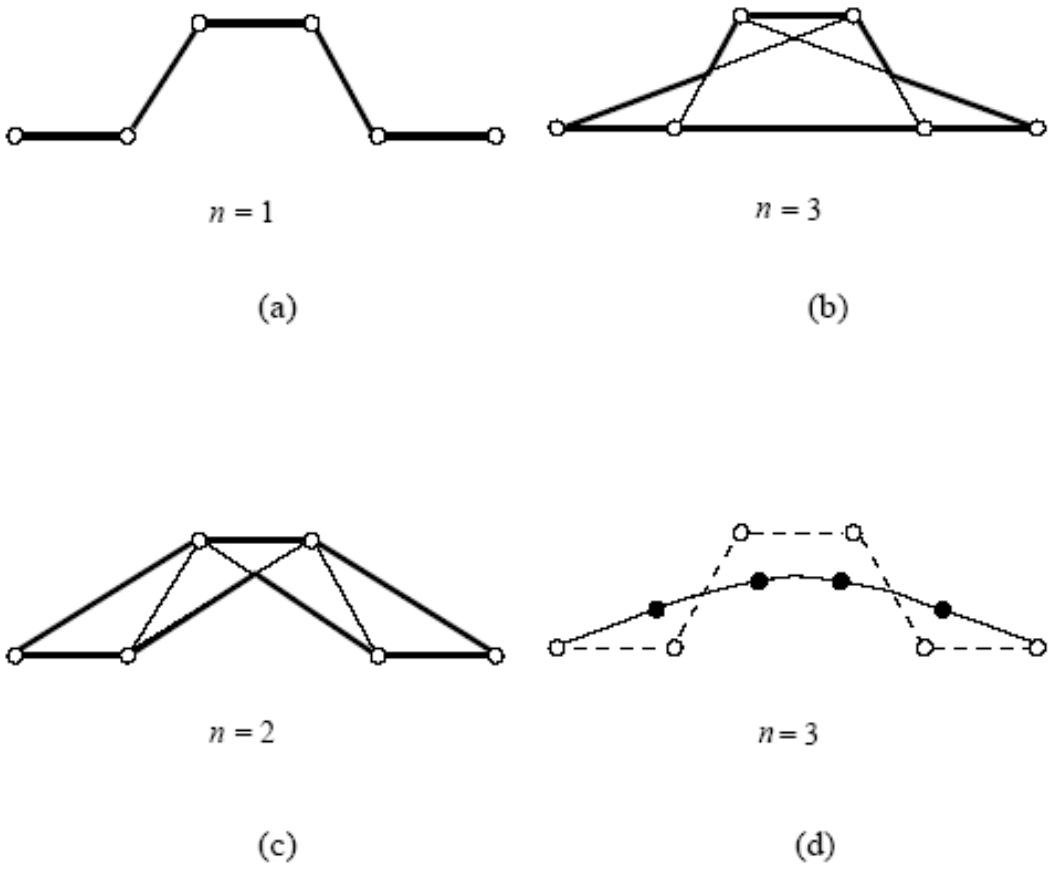


Figure 8.16: Splines of order n. (a), (b), (c) Convex $n+1$-polygon for a B-spline of the n^{th} order. (d) 3^{rd}-order spline.

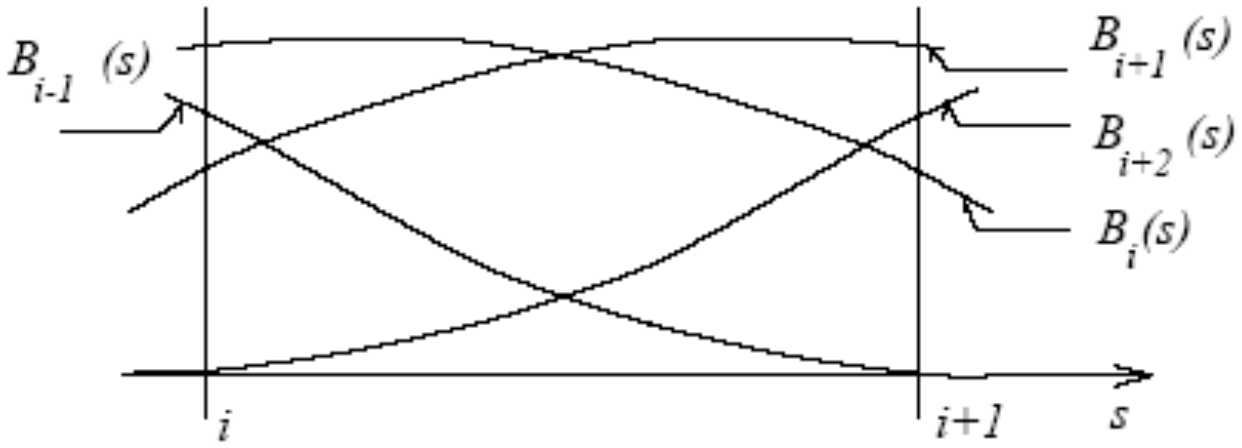


Figure 8.17: The only four non-zero base functions for $s \in (i, i+1)$.

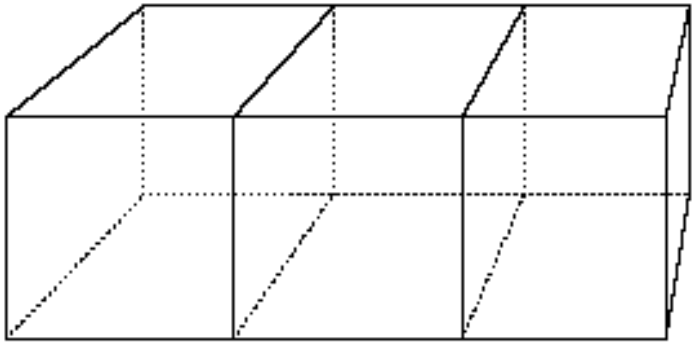


Figure 8.18: Change of shape caused by a projective transform. The same rectangular cross section is represented by different polygons in the image plane.

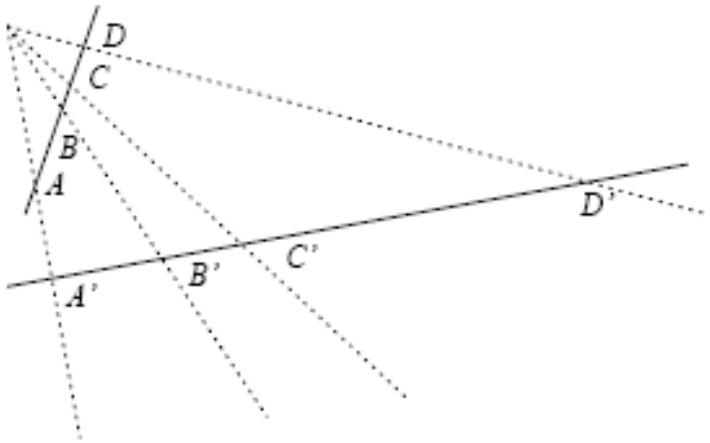


Figure 8.19: Cross ratio; four collinear points form a projective invariant.

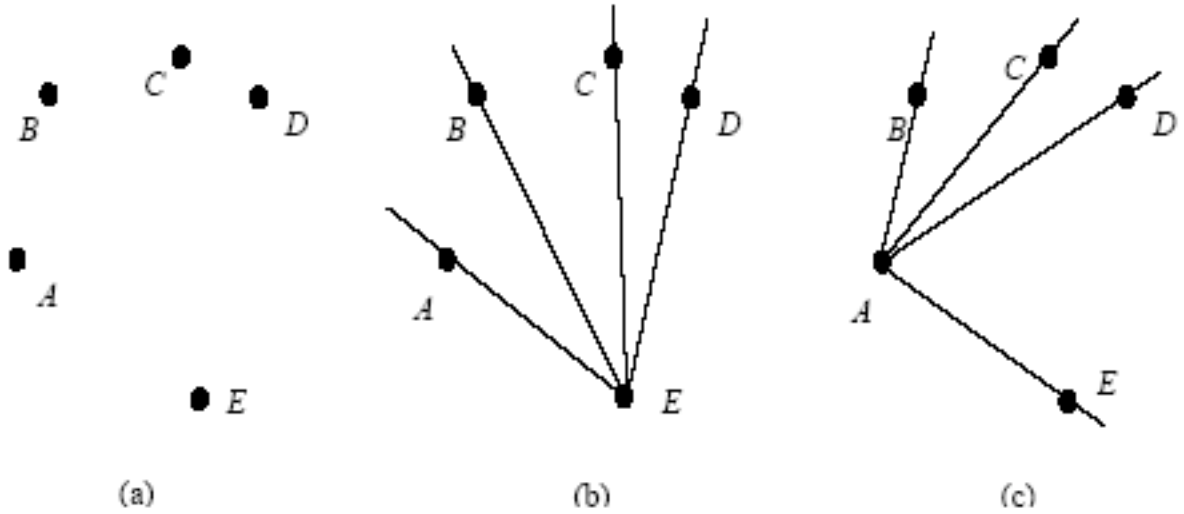
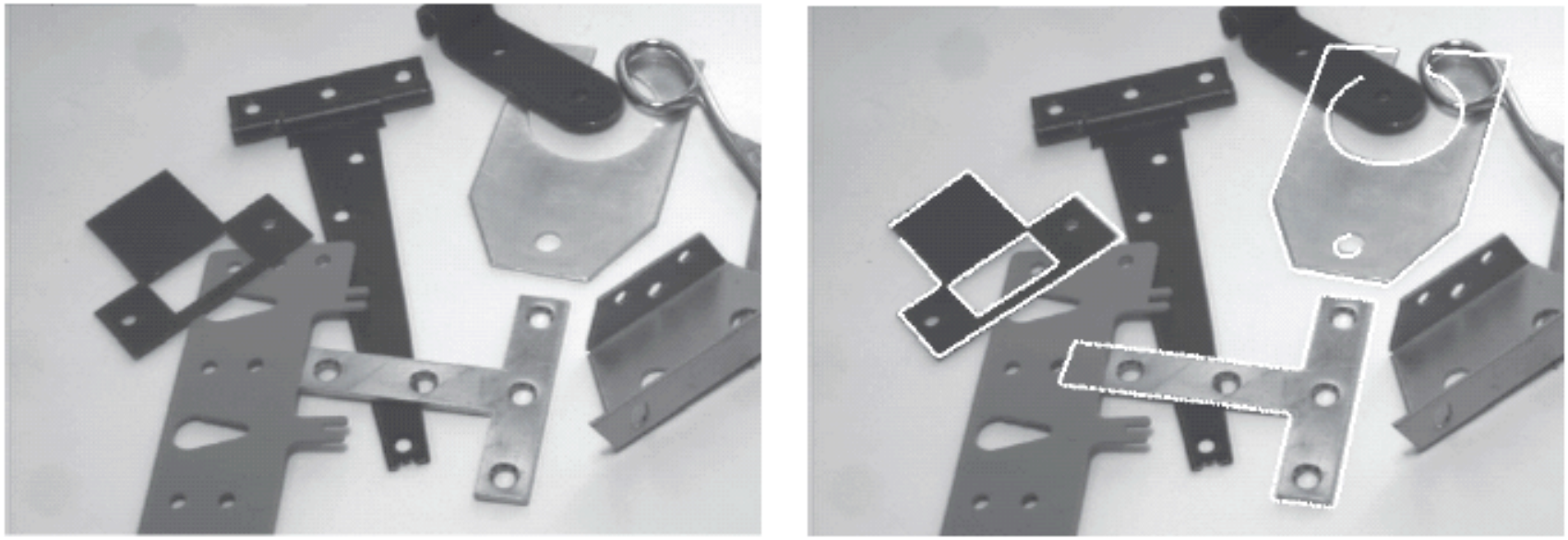


Figure 8.20: Five co-planar points form two cross-ratio invariants. (a) Co-planar points. (b) Five points form a system of four concurrent lines. (c) The same five points form another system of four co-planar lines.



(a) (b)

Figure 8.21: 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 of D. Forsyth, The University of Iowa; C. Rothwell, A. Zisserman, University of Oxford; J. Mundy, General Electric Corporate Research and Development, Schenectady, NY.*

Algorithm 8.4: 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 $area[region\_label]$.

Algorithm 8.5: 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*.

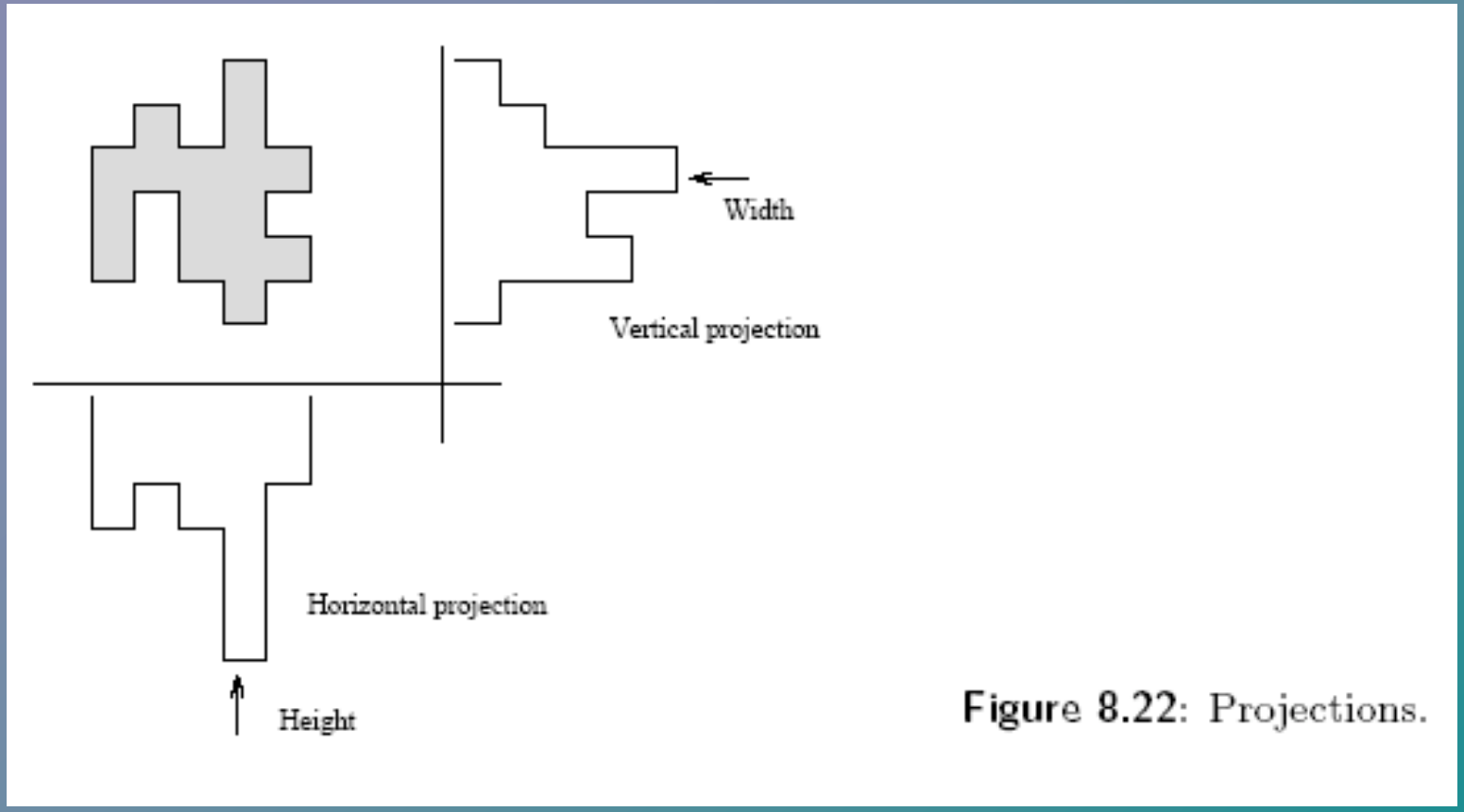


Figure 8.22: Projections.

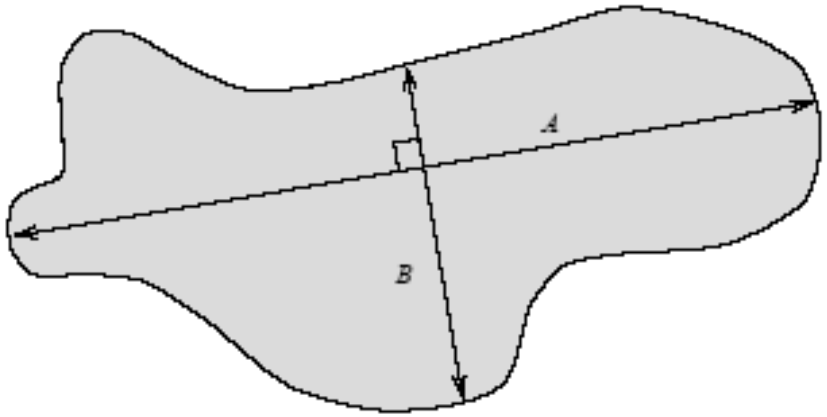


Figure 8.23: Eccentricity.

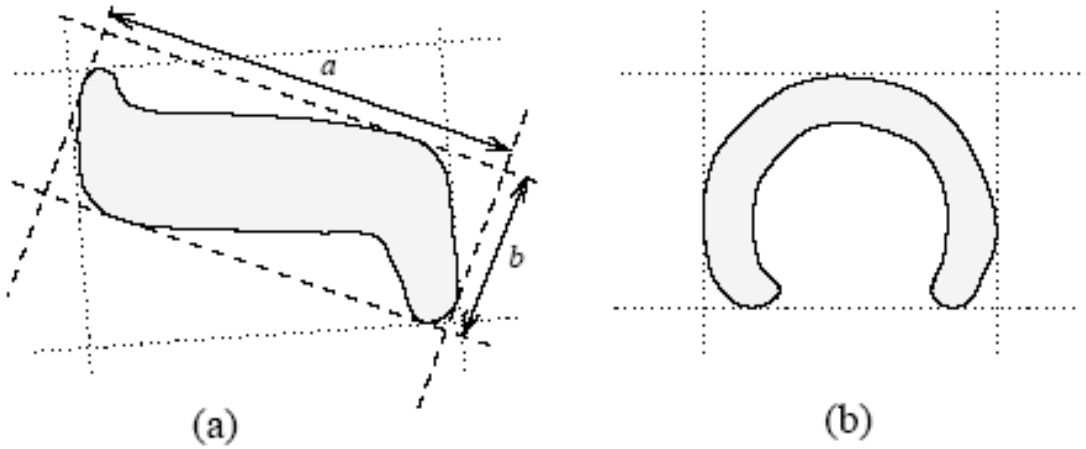


Figure 8.24: Elongatedness: (a) bounding rectangle gives acceptable results; (b) bounding rectangle cannot represent elongatedness.

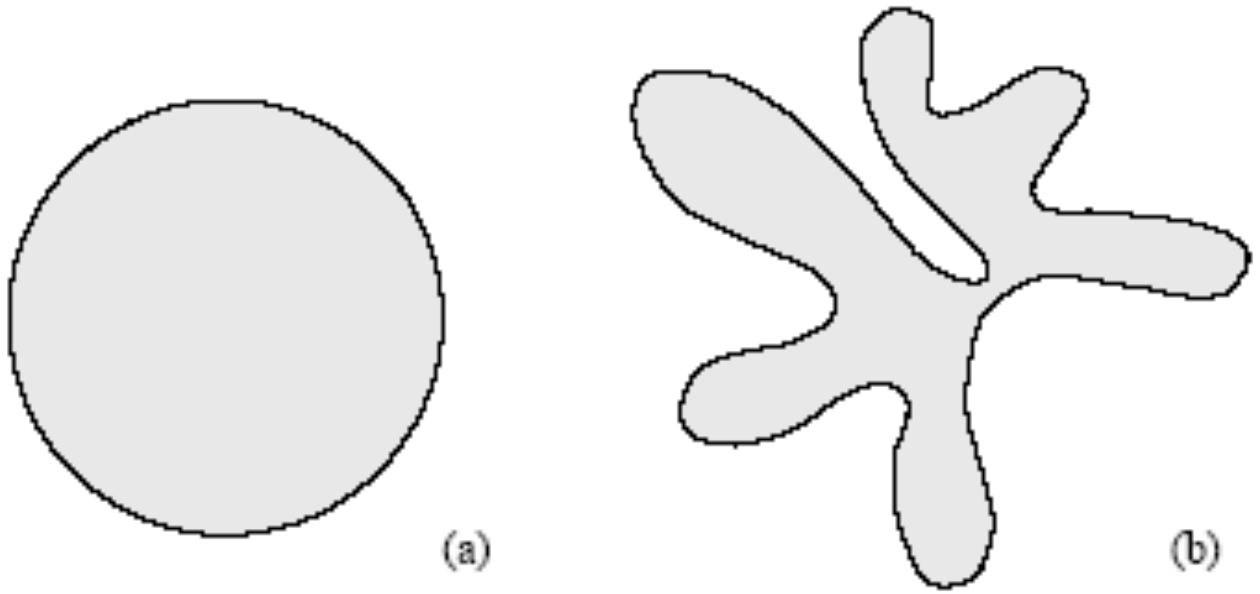


Figure 8.25: Compactness: (a) compact; (b) non-compact.

Algorithm 8.6: 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 6.7) 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 8.26). 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.
3. Assign $\mathbf{v} = \mathbf{P}_k\mathbf{P}_q$, $\mathbf{P}_k = \mathbf{P}_q$.
4. Repeat steps 2 and 3 until $\mathbf{P}_k = \mathbf{P}_1$.

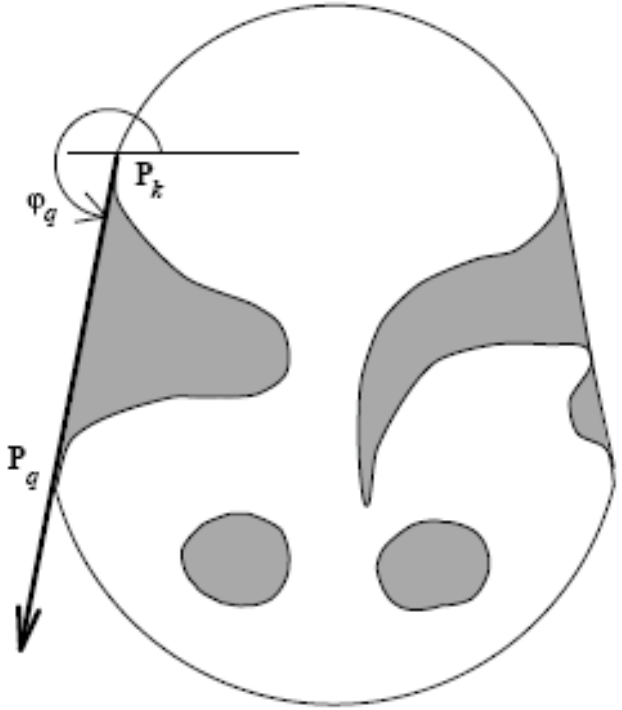


Figure 8.26: Convex hull.

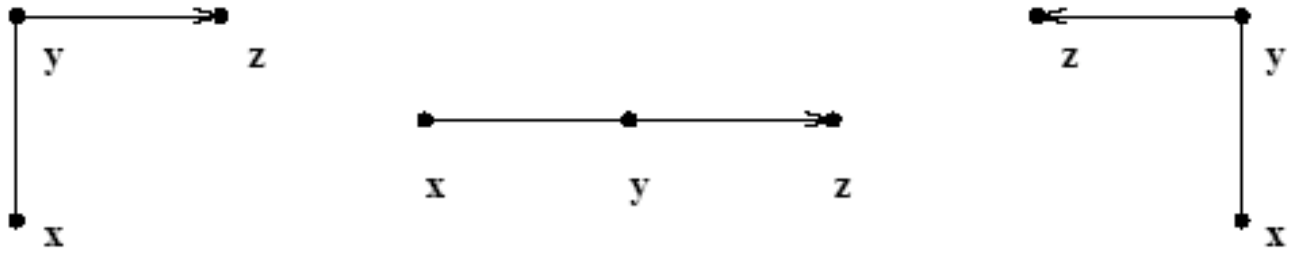


Figure 8.27: Directional function δ. (a) $\delta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 1$. (b) $\delta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$. (c) $\delta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = -1$.

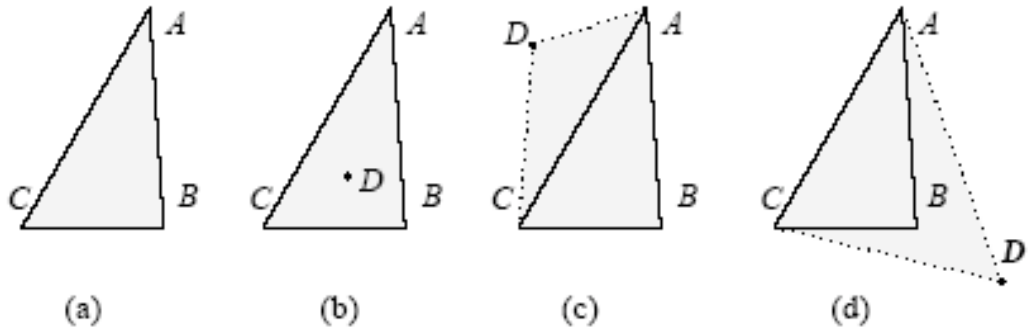


Figure 8.28: Convex hull detection. (a) First three vertices A, B, C form a triangle. (b) If the next vertex D is positioned inside the current convex hull ABC, current convex hull does not change. (c) If the next vertex D is outside of the current convex hull, it becomes a new vertex of the new current convex hull $ABCDA$. (d) In this case, vertex B must be removed from the current convex hull and the new current convex hull is $ADCA$.

Algorithm 8.7: Simple polygon convex hull detection

1. Initialize.
   . $t := -1$;
   . $b := 0$;
   . input v_1; input v_2; input v_3;
   . if ($\delta(v_1, v_2, v_3) > 0$)
   . { push v_1;
   . push v_2; }
   . else
   . { push v_2;
   . push v_1; }
   . push v_3;
   . insert v_3;
2. If the next vertex v is inside the current convex hull H, enter and check a new vertex; otherwise process steps 3 and 4;
   . input v;
   . while ($\delta(v, d_b, d_{b+1}) \geq 0$ AND $\delta(d_{t-1}, d_t, v) \geq 0$)
   . input v;
3. Rearrange vertices in H, top of the list.
   . while ($\delta(d_{t-1}, d_t, v) \leq 0$)
   . pop d_t;
   . push v;
4. Rearrange vertices in H, bottom of the list.
   . while ($\delta(v, d_b, d_{b+1}) \leq 0$)
   . remove d_b;
   . insert v;
   . go to step 2;

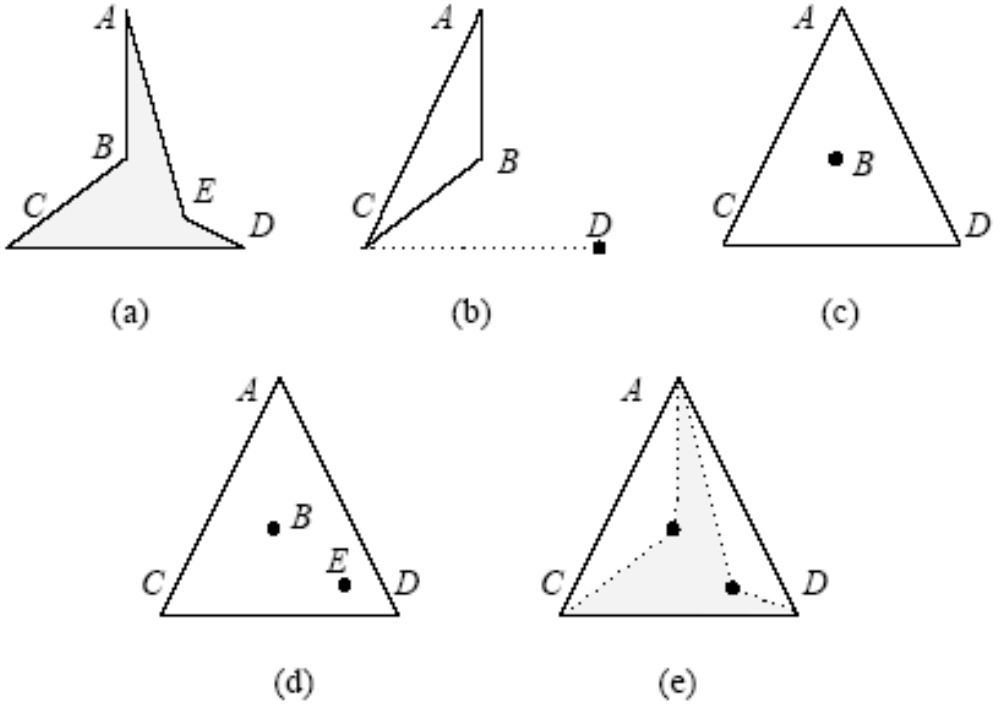


Figure 8.29: Example of convex hull detection. (a) The processed region—polygon $ABCDEA$. (b) Vertex D is entered and processed. (c) Vertex D becomes a new vertex of the current convex hull ADC. (d) Vertex E is entered and processed, E does not become a new vertex of the current convex hull. (e) The resulting convex hull $DCAD$.

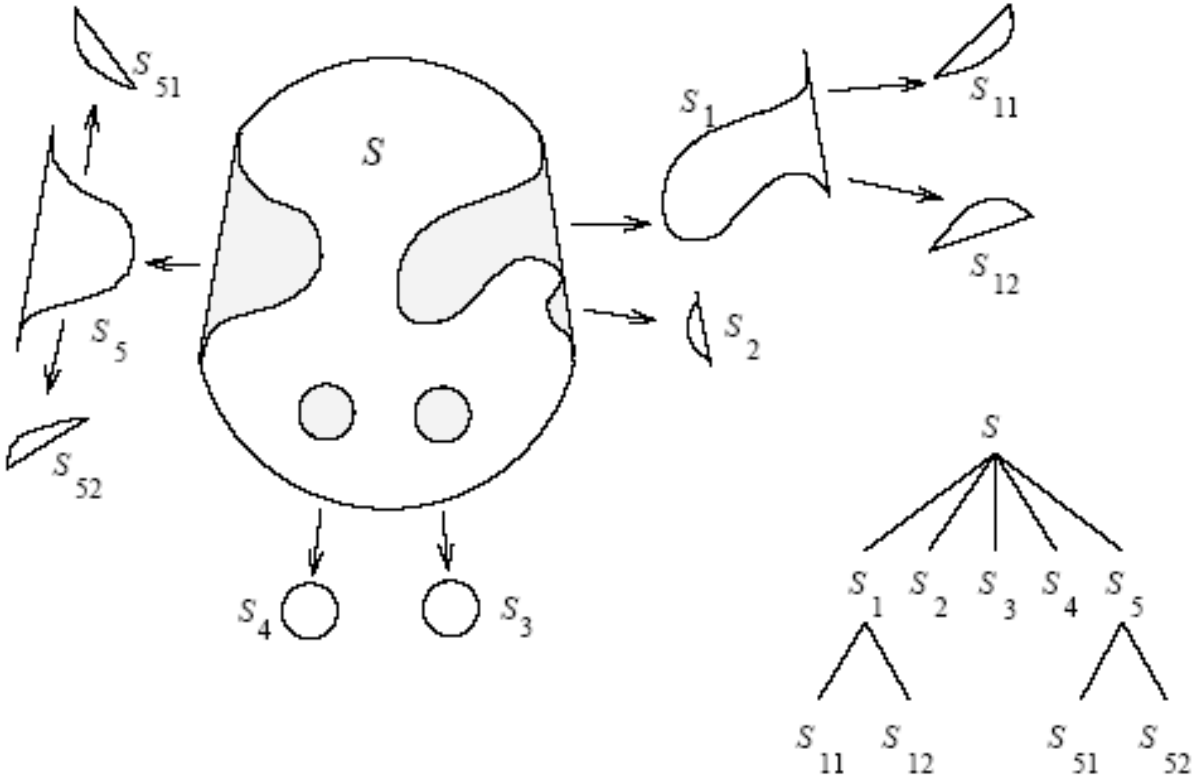


Figure 8.30: Concavity tree construction. (a) Convex hull and concave residua. (b) Concavity tree.

Algorithm 8.8: 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 8-connectivity either from the inner boundary $H_i(R)$ or from the background—from the residuum of R. Assign $R_{\text{old}} = R$.
2. Construct a region R_{new} which is a result of one-step thinning as follows

$$R_{\text{new}} = S(R_{\text{old}}) \cup \left[R_{\text{old}} - H_i(R_{\text{old}})\right] \cup \left[H_o\big(S(R_{\text{old}})\big) \cap R_{\text{old}}\right] .$$

3. If $R_{\text{new}} = R_{\text{old}}$, terminate the iteration and proceed to step 4. Otherwise assign $R_{\text{old}} = R_{\text{new}}$ and repeat step 2.
4. R_{new} is a set of skeleton pixels, the skeleton of the region R.

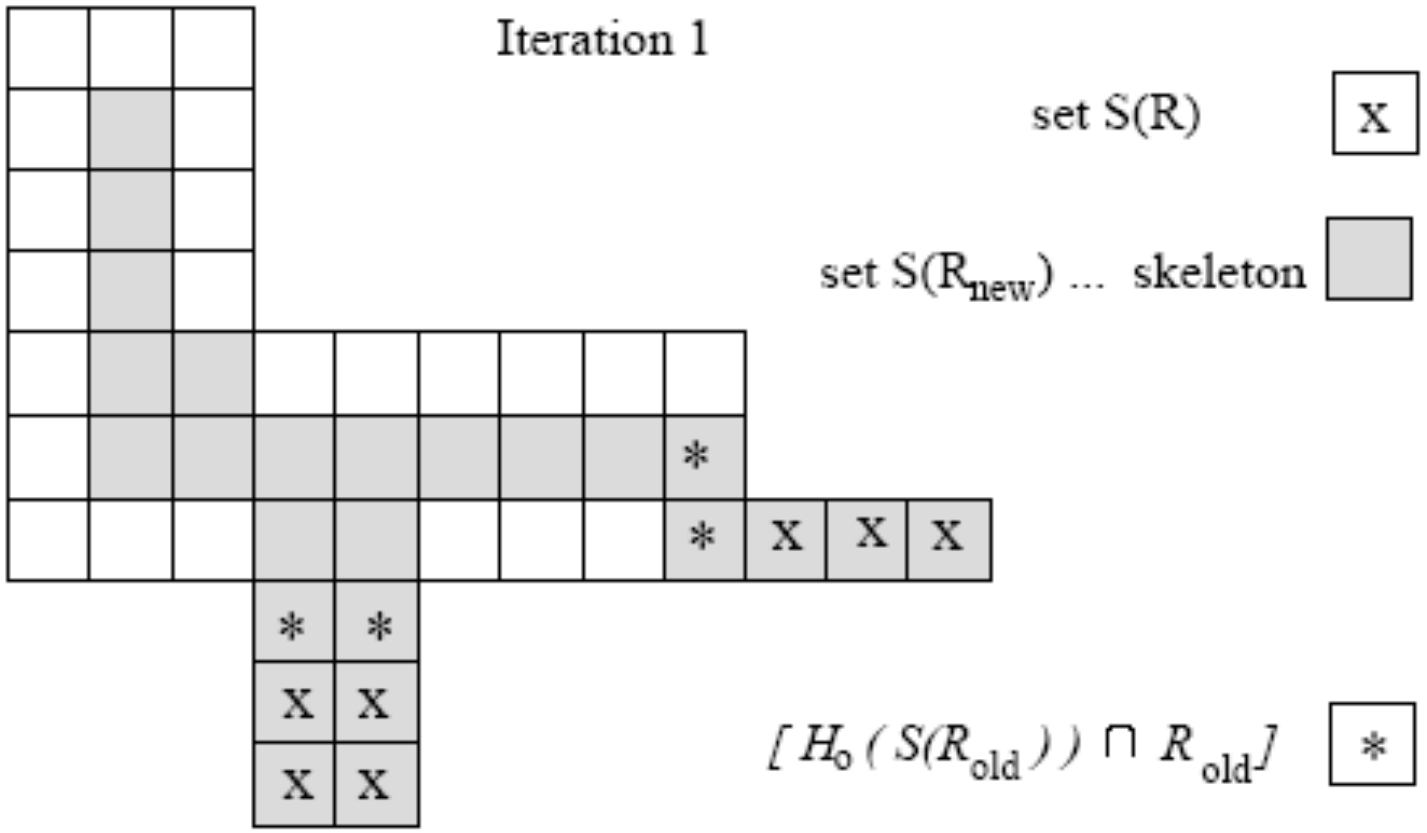


Figure 8.31: Skeleton by thinning (Algorithm 8.8).

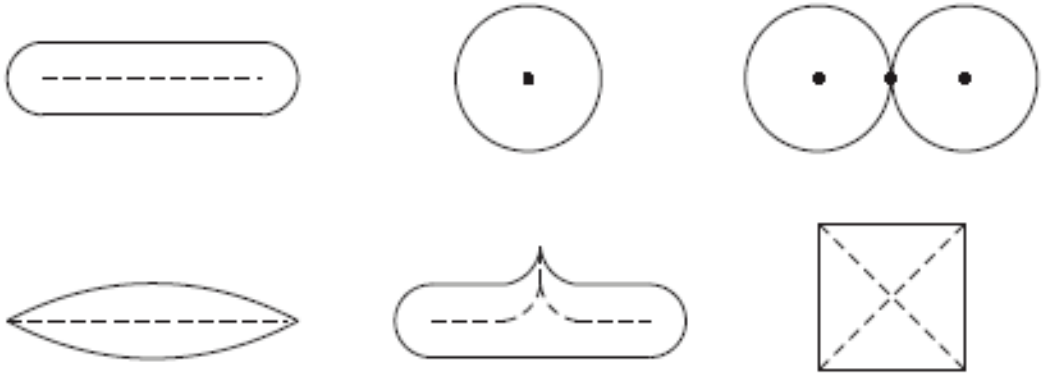


Figure 8.32: Region skeletons; small changes in border can have a significant effect on the skeleton.

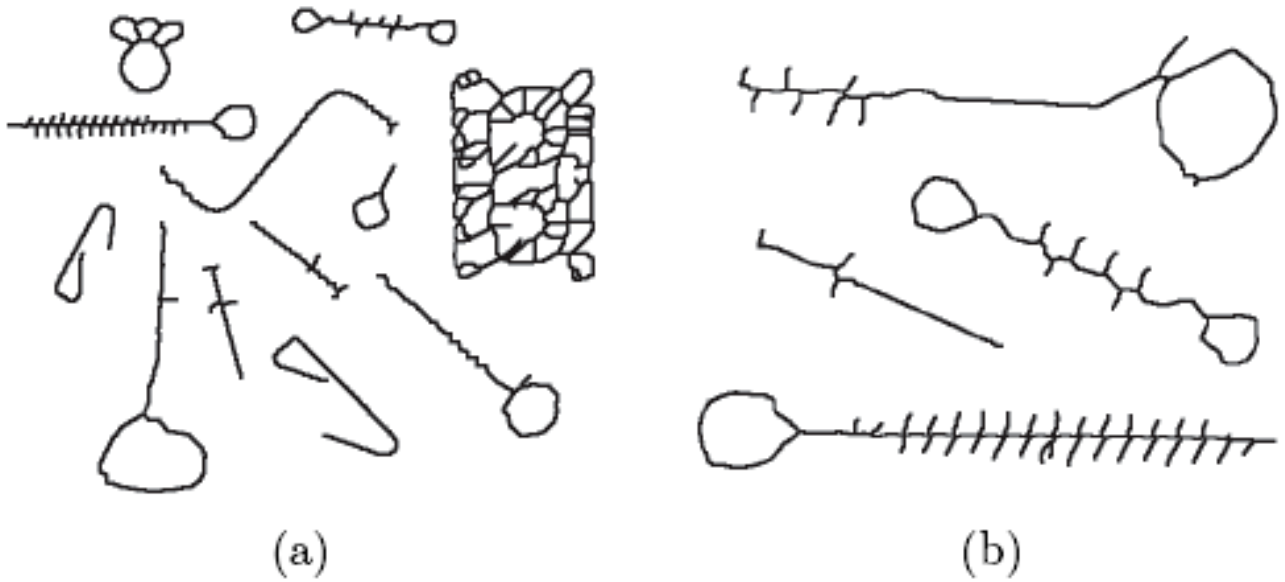


Figure 8.33: Region skeletons, see Figures 6.1a and 8.2a for original images; thickened for visibility.

Algorithm 8.9: 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.

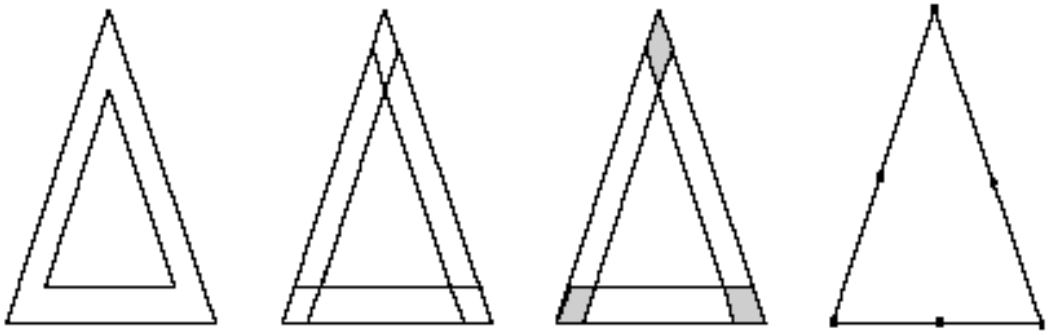


Figure 8.34: Region decomposition. (a) Region. (b) Primary regions. (c) Primary sub-regions and kernels. (d) Decomposition graph.

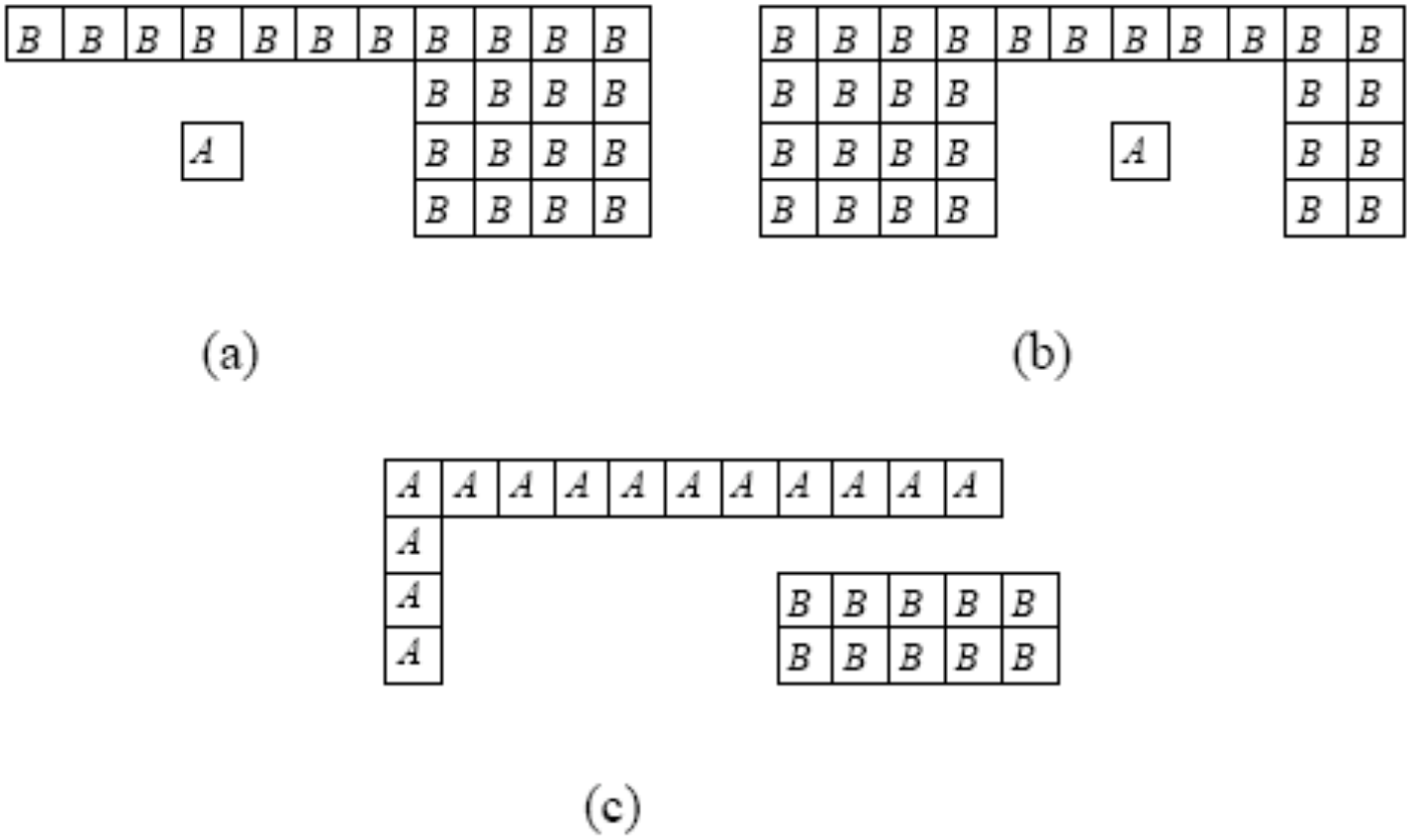


Figure 8.35: Binary relation *to be left of*; see text.

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 is important in shape.
- 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.

Summary

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.

Summary

Contour-based shape descriptors cont.

- **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 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.

Summary

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.

Summary

Region-based shape descriptors cont.

- The **convex hull** of a region is the smallest convex region H which satisfies the condition R H.
- More complicated shapes can be described using region decomposition into smaller and simpler sub-regions. Objects can be represented by a planar graph with nodes representing sub-regions resulting from region decomposition. Region shape can then be described by the graph properties. There are two general approaches to acquiring a graph of sub-regions:
    * Region thinning.
    * Region decomposition.

Summary

Region-based shape descriptors cont.

- **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.

Summary

Region-based shape descriptors cont.

- **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.