

# Digital Image Processing

## Shape representation and description

### Region identification

#### Shape representation and description

- Defining the shape of an object can prove to be very difficult. Shape is usually represented verbally or in figures.
- 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 or mathematics use effective shape representations which are unusable in shape recognition and vice versa.
- In spite of this, it is possible to find features common to most shape description approaches.
- Common shape description methods can be characterized from different points of view
  - Input representation form: Object description can be based on boundaries or on more complex knowledge of whole regions.
  - Object reconstruction ability: That is, whether an object's shape can or cannot be reconstructed from the description.
  - 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.
- A robustness of description to translation, rotation, and scale transformations: Shape description properties in different resolutions.

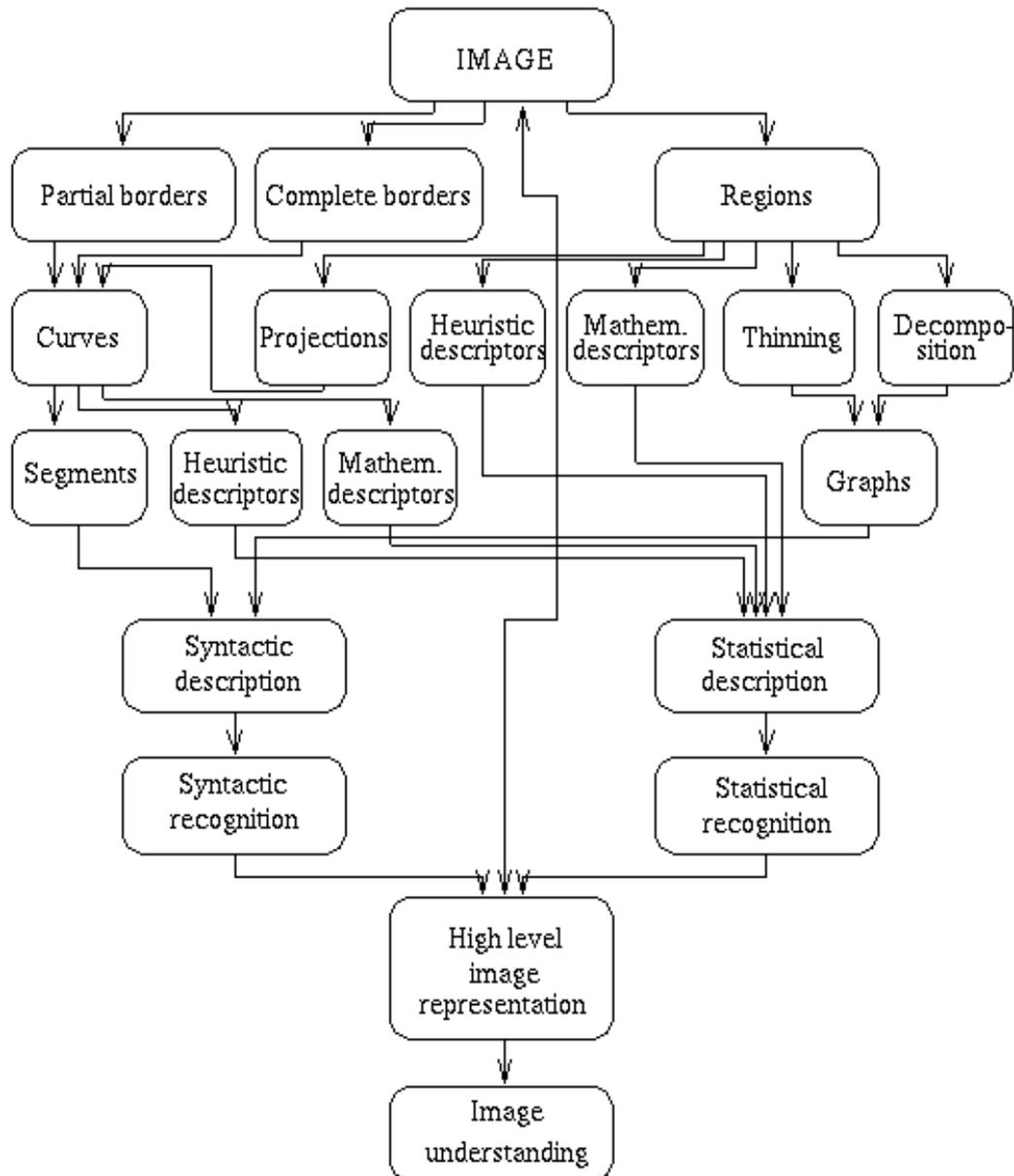


Fig 1: Image analysis and understanding methods

- Sensitivity to scale is even more serious if a shape description is derived, because shape may change substantially with image resolution.
- Therefore, shape has been studied in multiple resolutions which again cause difficulties with matching corresponding shape representations from different resolutions.
- Moreover, the conventional shape descriptions change discontinuously.
- A **scale-space** approach aims to obtain continuous shape descriptions if the resolution changes continuously.

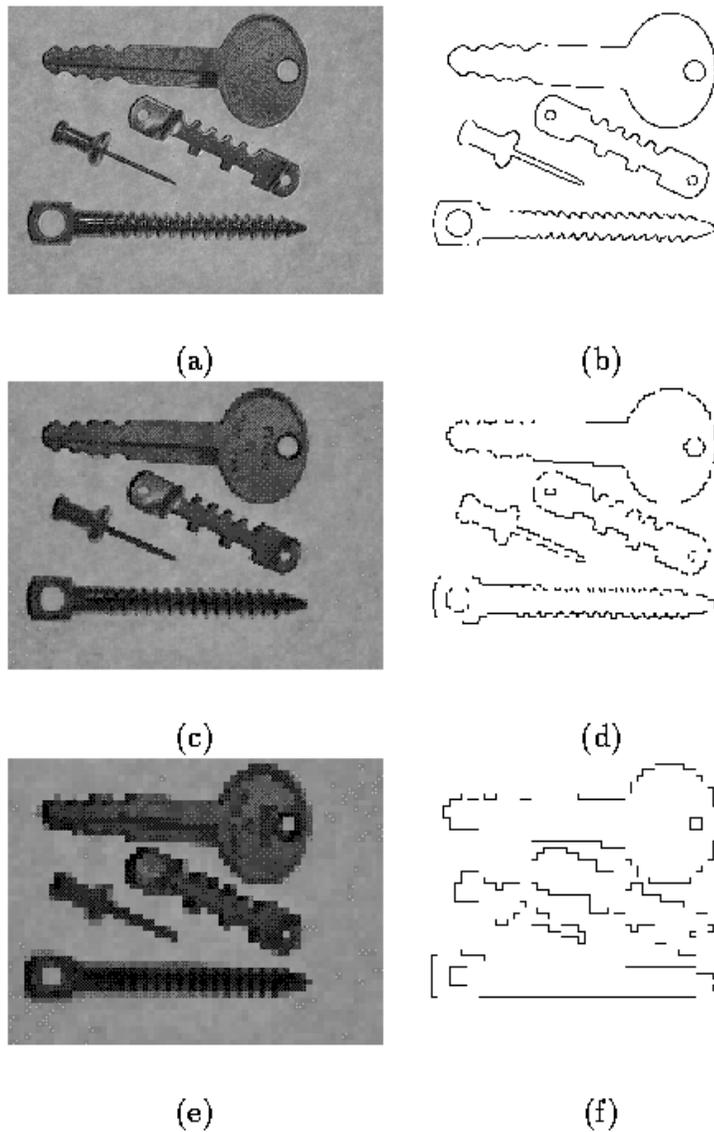


Fig 2: (a)Original image 640x480 (b)Contours of (a) (c)Original image 160x120 (d)Contours of (c) (e)Original image 64x48 (f)Contours of (e)

- 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.
- 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.
- 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 (**invariants**) can be found.
- 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), 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; such a description is useless if only a part of an object is visible.
- If a local descriptor is applied, 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.

## 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 gives number of regions in the image.
- Another method is to use smaller number of labels by ensuring 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 stored in a separate data structure.
- Goal of segmentation was to achieve complete segmentation, now, the regions must be labeled.  $R$  is segmented image consists of  $m$  disjoint regions  $R_i$ . The image  $R$  consists of objects and a background

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

$R^c$  is the set of complement,  $R_b$  is background, and other regions are considered as objects. Either binary or multilevel image will be the input where background is represented by zero pixels, and objects by non-zero values.

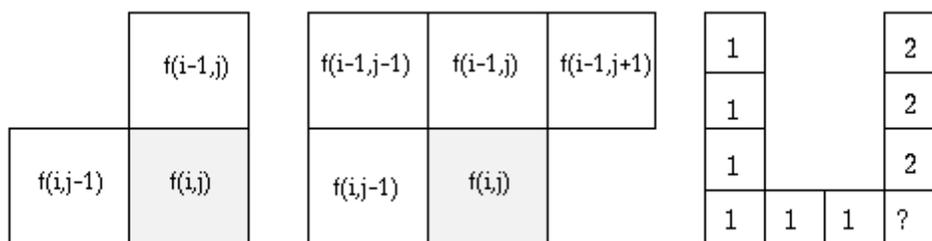


Fig 3: Masks for region identification (a) 4-connectivity (b) 8-connectivity (c) Label collision

## Algorithm: 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 pixels neighbors where the property neighboring is defined by Fig 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.
  - 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.
  - The algorithm is basically the same in 4-connectivity and 8-connectivity, the only difference being in the neighborhood mask shape.
  - It is useful to assign the region labels incrementally to permit the regions to be counted easily in second pass. Example for partial result is in Fig 3.
  - The region counting task is closely related to the region identification problem.

- Object counting can be an intermediate result of region identification. If it is only to count regions with no need to identify them, a one-pass algorithm is sufficient.

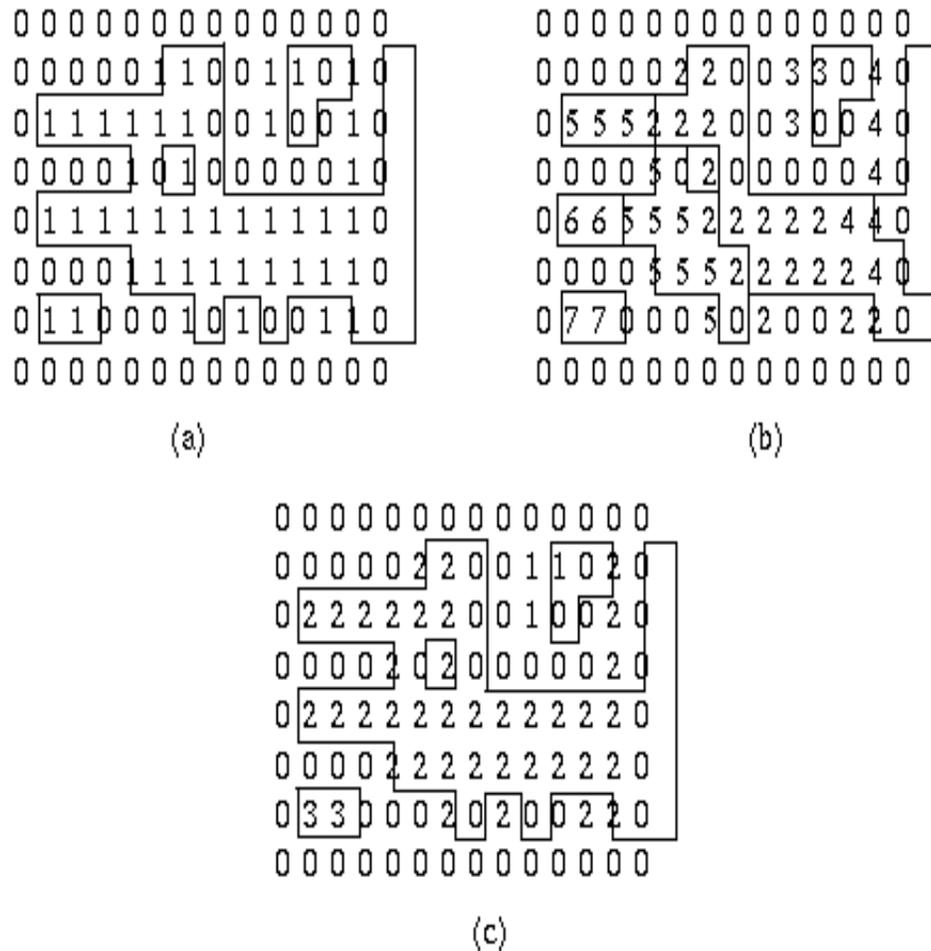


Fig 4: Object identification in 8-connectivity. (a), (b), (c) Algorithm steps. Equivalence table after step (b):  
2-5,5-6,2-4

### Algorithm: 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 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 quad tree data structure, the following algorithm may be applied.

**Algorithm: Quad tree region identification**

1. First pass: search quad tree 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. 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 quad tree according to the equivalence table.

## Contour-based shape representation and description

- Region borders must be expressed in some mathematical form.
  - **Polar** co-ordinates, in which border elements are represented as pairs of angle  $\phi$  and distance  $r$ ;
  - **Tangential** co-ordinates, which codes the tangential directions  $\theta(x_n)$  of curve points as a function of path length  $n$ .

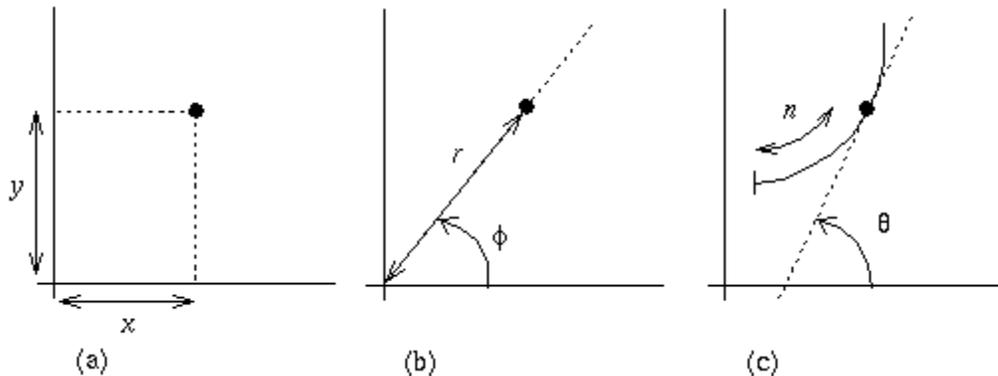


Fig 5: Co-ordinate systems. (a) Rectangular (Cartesian). (b) Polar (c) Tangential

### Chain codes

- Chain codes describe an object by a sequence of unit-size line segments with a given orientation.
- 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 (Fig 6).

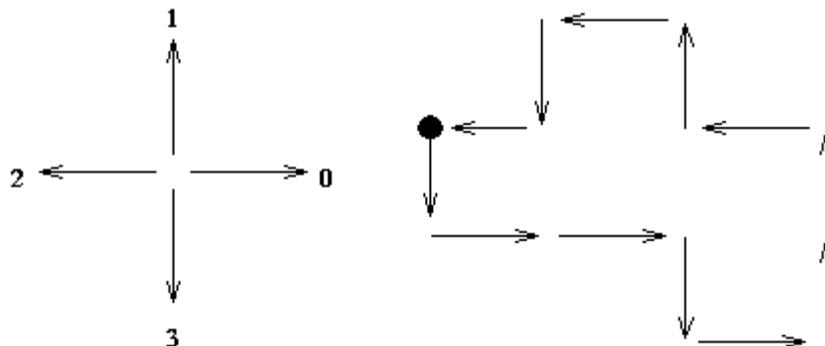


Fig 6: Chain code in 4-connectivity, and its derivative. Code: 3,0,0,3,0,1,1,2,1,2,3,2; derivative: 1,0,3,1,1,0,1,3,1,1,3,1.

- 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.
  - A *mod 4* or *mod 8* differences is called a chain code **derivative**, is another numbered sequence that represents relative directions of region boundary elements, measured as multiples of counter-clockwise  $90^0$  or  $45^0$  direction changes (Fig 6).

### 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*

Boundary length is derived from chain code representation. Vertical and horizontal steps have unit length of diagonal steps in 8-connectivity is  $\sqrt{2}$ . A closed boundary length is evaluated from run length or quad tree representations. Boundary length increases as the image raster resolution increases.

- *Curvature*

Curvature is defined as the rate of change of slope. 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. Histogram of boundary angles, such as the  $\beta$  angle in Fig 7, can be built in a similar way—such histograms can be used for region description.

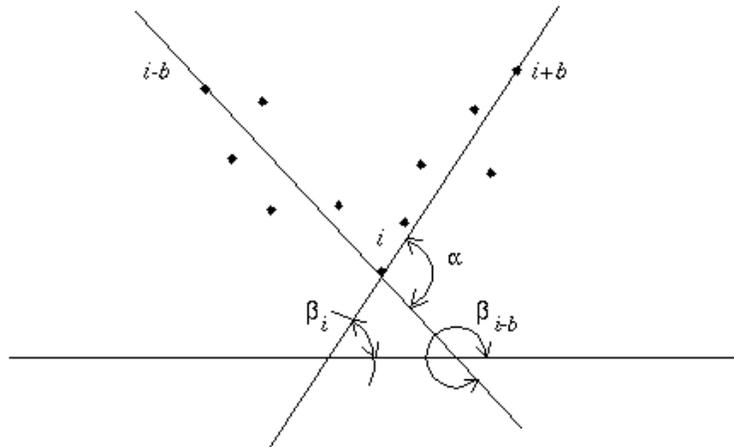


Fig 7: Curvature

- *Bending energy*

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

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

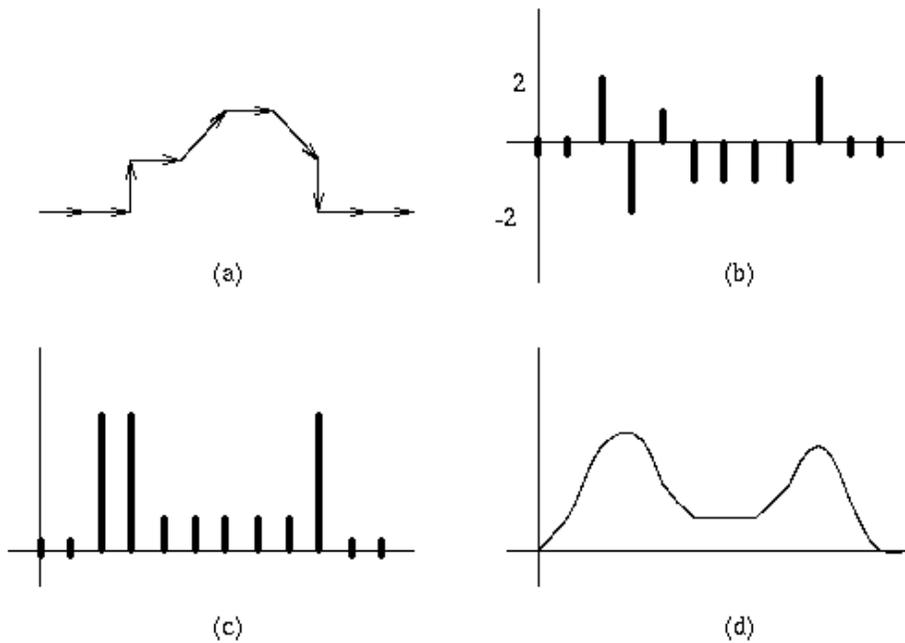


Fig 8: Bending energy (a)chain code 0,0,,0,1,0,7,6,0,0. (b)Curvature 0,2,-2,1,-1,-1,-1,2,0. (c)Sum of the squares gives the bending energy. (d) Smoothed version.

o *Signature*

The signature of a region is 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 (Fig 9). Signatures may be applied to the recognition of overlapping objects or whenever only partial contours are available.

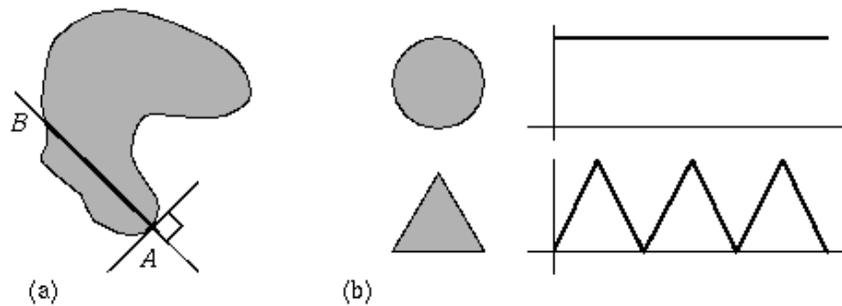


Fig 9: Signature (a) Construction (b) Signatures for a circle and a triangle

- *Chord distribution*

Chord is a line joining any two points of the region boundary. 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 (Fig 10(a)) as

$$h(\Delta x, \Delta y) = \iint b(x, y)b(x + \Delta x, y + \Delta y)dx dy$$

or in digital image as

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

Rotation-independent radial distribution (Fig 6(b)):

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

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

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

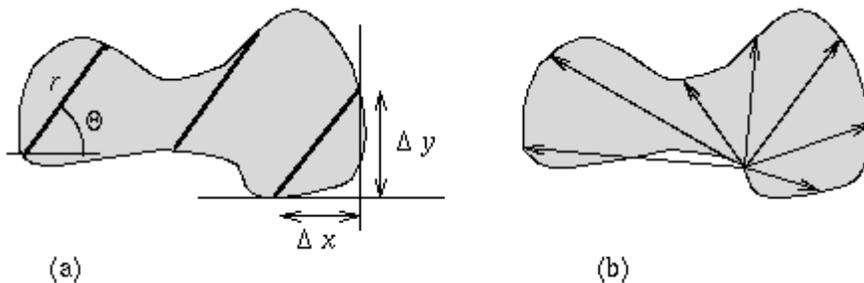


Fig 10: Chord distribution

## Fourier transforms of boundaries

- Suppose  $C$  is a closed curve (boundary) in the complex plane (Fig 11(a)).
- Traveling anti-clockwise along this curve keeping constant speed, a complex function  $z(t)$  is obtained, where  $t$  is a time variable.

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

- 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

$$t = \frac{2\pi s}{L},$$

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

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

- 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.

$$z(k) \leftarrow DFT \rightarrow T_n$$

- The Fourier descriptors can be invariant to translation and rotation if the co-ordinate system is appropriately chosen.

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

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

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

$$r_n = (|a_n|^2 + |b_n|^2)^{1/2}$$

$r_n$  are translation and rotation invariant.

- To achieve magnification invariance the descriptors  $w_n$  are used:

$$w_n = r_n/r_1$$

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

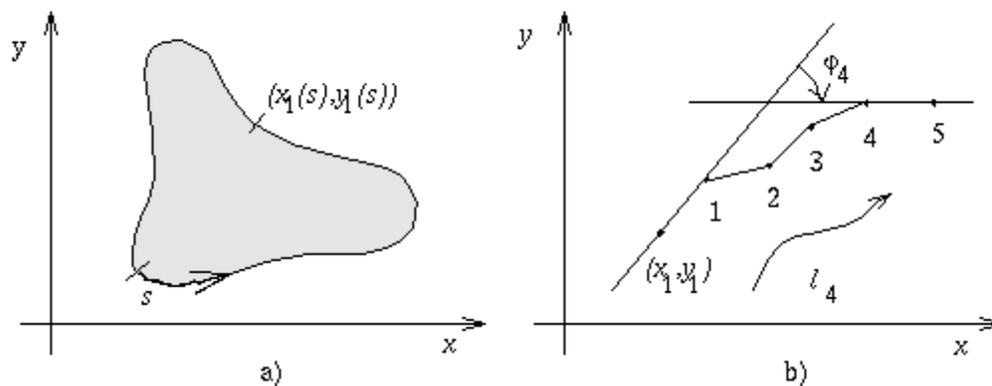


Fig 11: Fourier description of boundaries (a) Descriptors  $T_n$  (b) Descriptors  $S_n$

- 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

$$a(l_k) = \varphi_k + u_k$$

$$u_k = 2\pi l_k/L$$

- The descriptor set is then

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

- The high quality boundary shape representation obtained using only a few lower order coefficients is a favorable property common to Fourier descriptors.

### Boundary description using segment sequences

- 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.
- A **polygonal representation** approximates a region by a polygon, the region being represented using its vertices. Polygonal representations are obtained as a result of simple boundary segmentation.
- Another method for determining the boundary vertices is a **tolerance interval approach** based on setting a maximum allowed difference  $\epsilon$ .

Assume that point  $x_1$  is the end point of a previous segment and so by definition the first point of new segment. Define points  $x_2, x_3$  positioned a distance  $\epsilon$  from the point  $x_1$  to be rectilinear--  $x_1, x_2, x_3$  are positioned on a straight line (Fig 8). The next step is to locate a segment which can fit between parallels directed from points  $x_2$  and  $x_3$ . Resulting segments are sub-optimal, although optimality can be achieved with a substantial increase in computational effort.

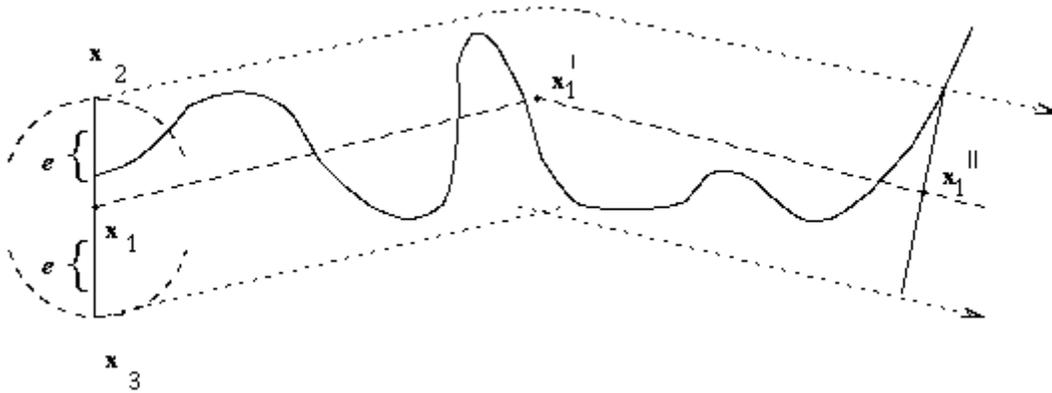


Fig 12: Tolerance interval

- Recursive boundary splitting.

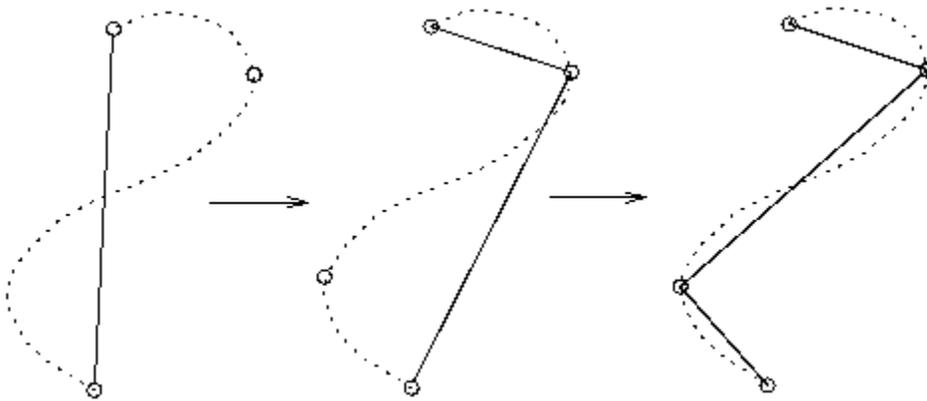


Fig 13: Recursive boundary splitting

- Boundary segmentation into segments of constant curvature or curve segmentation into circular arcs and straight lines is used.
  - Segments are considered as primitives for syntactic shape recognition procedures.

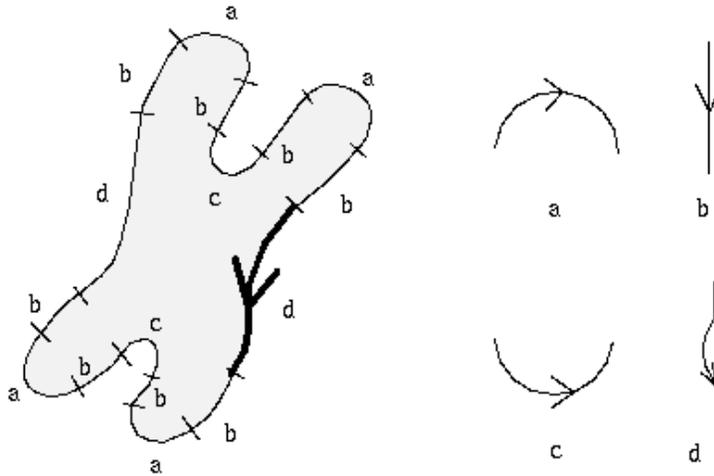


Fig 14: Structural description of chromosomes by a chain of boundary segments, code word:  
d,b,a,b,c,b,a,b,d,b,a,b,c,b,a,b.

- Sensitivity of shape descriptors to scale is also important if a curve is to be divided into segments -- a **scale-space approach** to curve segmentation.
- Only new segmentation points can appear at higher resolutions, and no existing segmentation points can disappear.
- 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**.

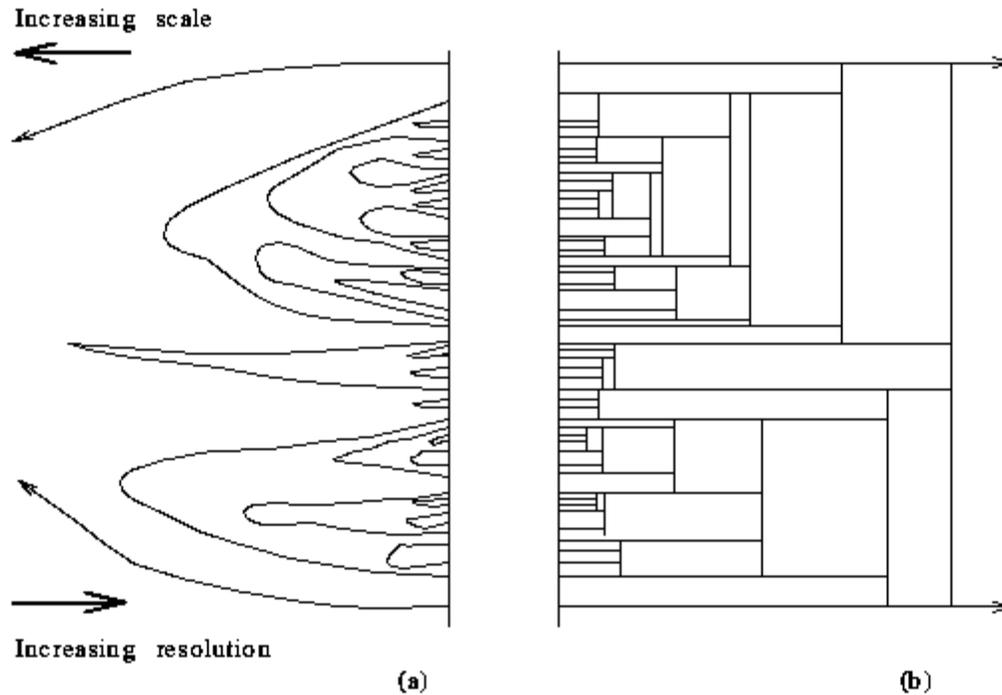


Fig 15: Scale-space image (a) Varying number and locations of curve segmentation points as a function of scale. (b) Curve representation by an interval tree.

## 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 than their control polygon, and do not oscillate between sampling points as many other representations do. A spline curve is always positioned inside a convex  $n+1$ -polygon for a B-spline of the  $n$ -th order.

- 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.

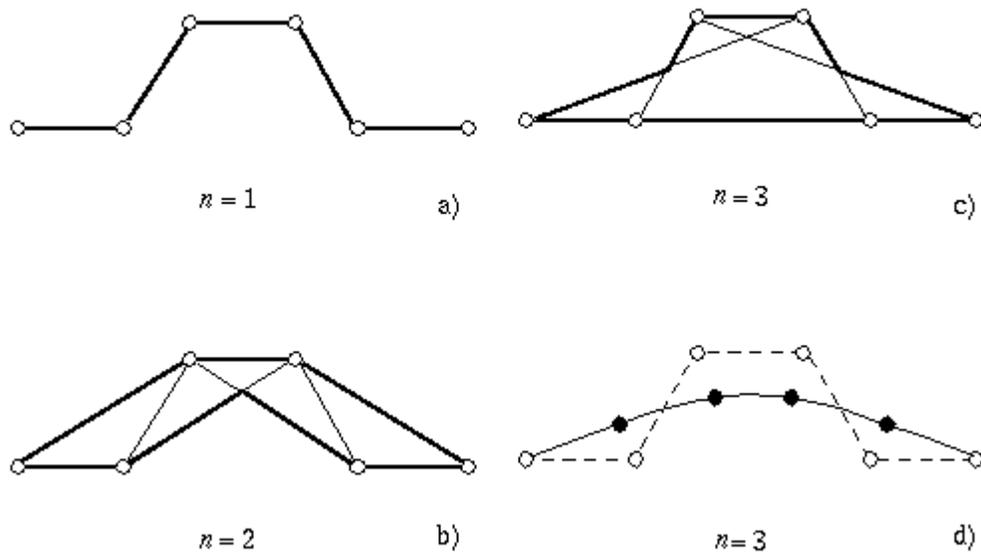


Fig 16: Splines of order  $n$ . (a),(b),(c) Convex  $n+1$  polygon for a B-spline of the  $n$ th order. (d) 3<sup>rd</sup> order spline.

- 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

$$X(s) = \sum_{i=0}^{n+1} v_i B_i(s)$$

$$v_0 = 2v_1 - v_2$$

$$v_{n+1} = 2v_n - v_{n-1}$$

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

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

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

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

$$X(s) = C_{i-1,3}(s)v_{i-1} + C_{i,2}(s)v_i + C_{i+1,1}(s)v_{i+1} + C_{i+2,0}(s)v_{i+2}$$

$$C_{i,j}(s) = c_j(s - i), \quad i = 0, \dots, n + 1; \quad j = 0, 1, 2, 3.$$

$$X(s) = C_3(s - i)v_{i-1} + C_2(s - i)v_i + C_1(s - i)v_{i+1} + C_0v_{i+2}$$

$$X(5) = C_3(0)v_4 + C_2(0)v_5 + C_1(0)v_6 = v_4 + v_5 + v_6,$$

$$X(5) = C_3(0.7)v_6 + C_2(0.7)v_7 + C_1(0.7)v_8 + C_0(0.7)v_9$$

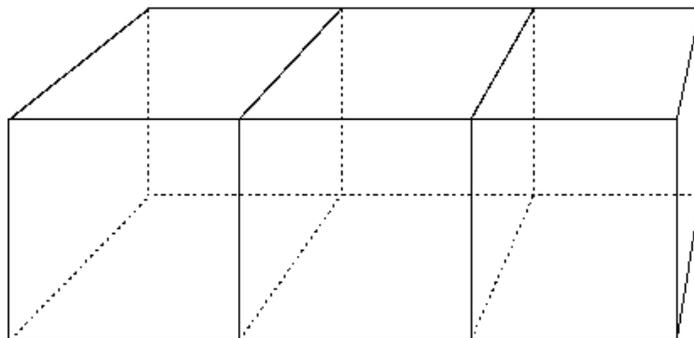
### Other contour-based shape description approaches

- Many other methods and approaches can be used to describe two-dimensional curves and contours.
- The **Hough transform** has excellent shape description abilities.
- Region-based shape description using **statistical moments** is covered below.
- **Fractal** approach to shape is gaining attention in image shape description.
- **Mathematical morphology** can be used for shape description, typically in connection with region skeleton construction.

- **Neural networks** 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.

## Shape invariants

- Shape invariants represent a very active current research area in machine vision.
- Importance of shape invariance has been known for a long time however it is somewhat novel approach in machine vision.
- Invariant theory is not new and many of its principles were introduced in the nineteenth century.
- 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.
- 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.
- Machine vision is especially concerned with the class of projective transforms.



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.
- Let corresponding entities in two different co-ordinate systems be distinguished by large and small letters. An invariant of a linear transformation may be defined as:
  - An invariant,  $I(P)$ , of a geometric structure described by a parameter vector  $P$ , subject to a linear transformation  $T$  of the co-ordinates  $x=TX$ , is transformed according to  $I(p)=I(P)|T|^w$ .
  - Here  $I(p)$  is the function of the parameters after the linear transformation, and  $|T|$  is the determinant of the matrix  $T$ .
- 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.

### ***Cross ratio:***

- The cross ratio represents a classic invariant of a projective line.
- 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

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

where (A-C) represents the distance between points A and C(Fig 17). Note that the cross ratio depends on the order in which the four collinear points are labeled.

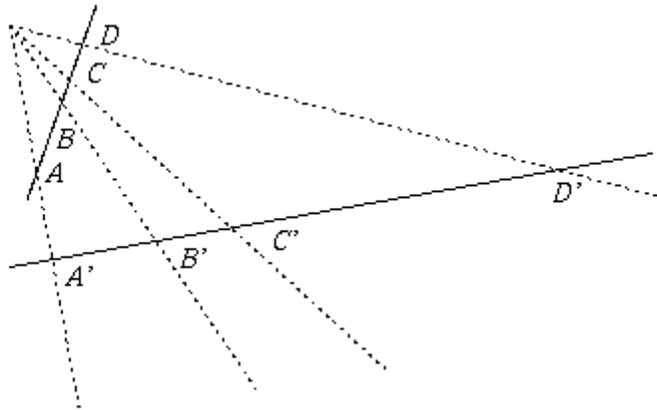


Fig 17: Cross ratio; four collinear points form a projective invariant.

### **Systems of lines or points**

- A system of four general coplanar lines forms two invariants .

$$I_1 = \frac{|M_{431}||M_{521}|}{|M_{421}||M_{531}|}, \quad I_2 = \frac{|M_{421}||M_{532}|}{|M_{432}||M_{521}|}$$

Where  $M_{ijk} = (I_i, I_j, I_k)$ .  $I_i = (l_i^1, l_i^2, l_i^3)^T$  is a representation of a line  $l_i^1 x + l_i^2 y + l_i^3 = 0$ , where  $i \in [1,5]$ , and  $|M|$  is the determinant of  $M$

- If the three lines forming the matrix  $M_{ijk}$  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 (Fig 18).

- Note that even though combinations other than those given in Figure may be formed, only the two presented functionally independent invariants exist.

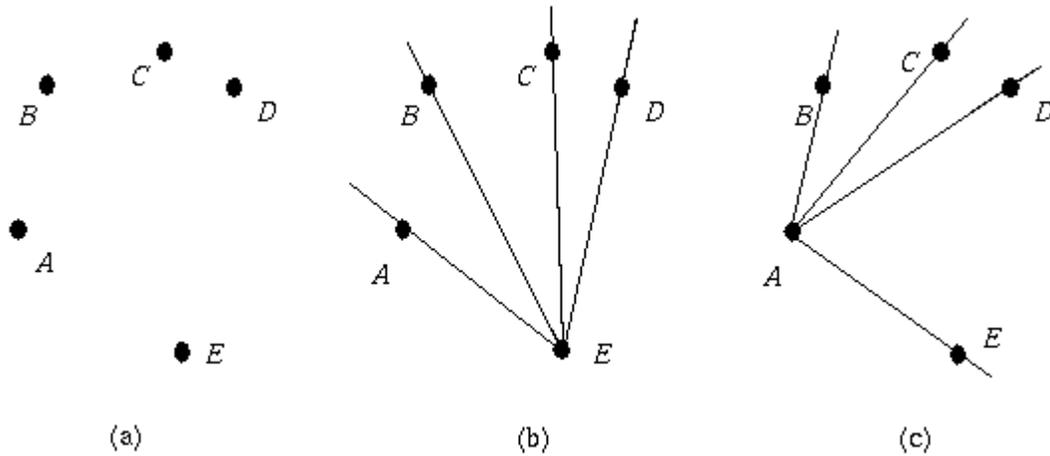


Fig 18: Five co-planar points form two cross-ratio invariants. (a) Co-planar points (b) Five points form a system of four concurrent lines (c) The same five points form another system of four co-planar lines

**Plane conics:**

- A plane conic may be represented by an equation

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

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

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

and

$$X^T C x = 0$$

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

$$I = \frac{(I_1^T C^{-1} I_2)^2}{(I_1^T C^{-1} I_1)(I_2^T C^{-1} I_2)}$$

- 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  $C_1, C_2$  normalized so that  $|C_i|=1$

$$I_1 = \text{Trace}[C_1^{-1}C_2], \quad I_2 = \text{Trace}[C_1^{-1}C_1]$$

- For non-normalized conics, the invariants of associated quadratic forms are

$$I_1 = \text{Trace}[C_1^{-1}C_2] \left( \frac{|C_1|}{|C_2|} \right)^{1/3}, \quad I_2 = \text{Trace}[C_2^{-1}C_2] \left( \frac{|C_2|}{|C_1|} \right)^{1/3}$$

and two true invariants of the conics are

$$I_1 = \frac{\text{Trace}[C_1^{-1}C_2] |C_1|}{(\text{Trace}[C_2^{-1}C_1])^2 |C_2|}, \quad I_2 = \frac{\text{Trace}[C_2^{-1}C_1] |C_2|}{(\text{Trace}[C_1^{-1}C_2])^2 |C_1|}$$

- 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.
- However, if additional information is available, higher derivatives may be avoided.
  
- 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.
- Different invariants have different stability and distinguishing powers.
  
- 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.
  - 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.

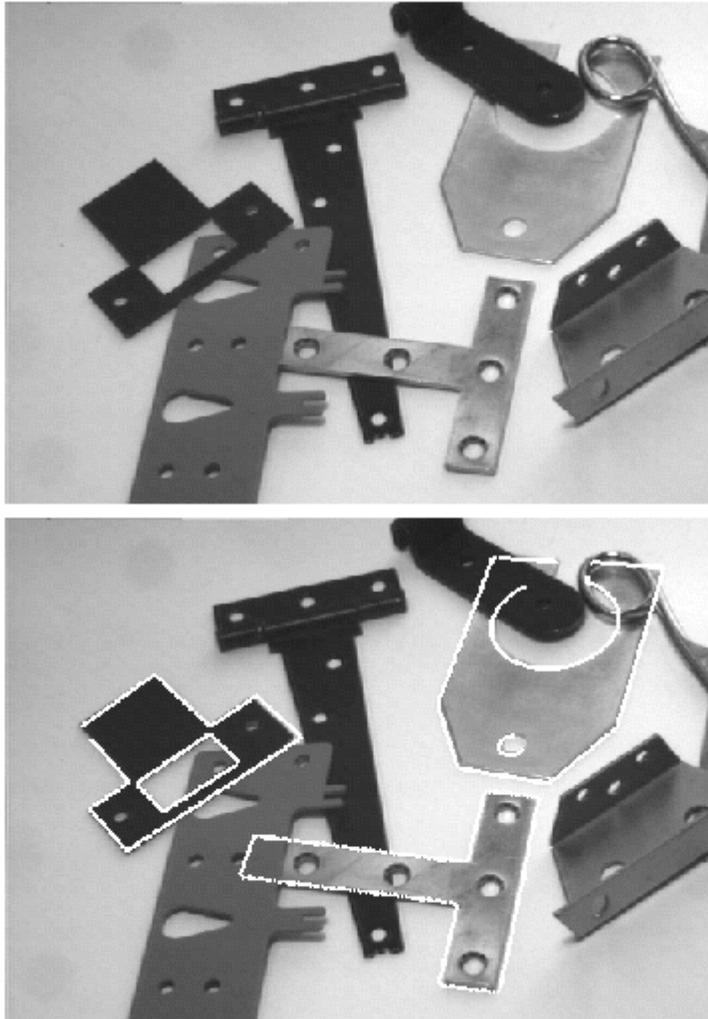


Fig 19: Object recognition based on shape invariants. (a) Original image of overlapping objects taken from an arbitrary viewpoint. (b) Object recognition based on line and conic invariants.

## Region-based shape representation and description

- A large group of shape description techniques is represented by heuristic approaches which yield acceptable results in description of simple shapes.
- Heuristic region descriptors: area, rectangularity, elongatedness, direction, compactness, etc.
- These descriptors cannot be used for region reconstruction and do not work for more complex shapes.
- Procedures based on region decomposition into smaller and simpler subregions must be applied to describe more complicated regions, and then sub regions can be described separately using heuristic approaches.

### Simple scalar region descriptors

- **Area**
- Area is 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.
- 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 quad tree, then:

### Algorithm: Calculating area in quad trees

1. Set all region area variables to zero, and determine the global quad tree depth  $H$ ; for example, the global quad tree depth is  $H=8$  for a  $256 \times 256$  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 also be represented by  $n$  polygon vertices  $(i_k, j_k)$ , and  $(i_0, j_0) = (i_n, j_n)$ . The area is given by

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

the sign of the sum represents the polygon orientation.

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

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

1. Set the region area to zero. Assign the value of the starting point  $i$  coordinate 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**

- (Sometimes called **Genus** or the **Euler-Poincare characteristic**) describes a simple topologically invariant property of the object.

- $S$  is the number of contiguous parts of an object and  $N$  is the number of holes in the object (an object can consist of more than one region).

$$\vartheta = S - N$$

- **Projections**

Horizontal and vertical region projections  $P_h(i)$  and  $P_v(j)$  are defined as

$$P_h(i) = \sum_j f(i, j) , \quad P_v(j) = \sum_i f(i, j).$$

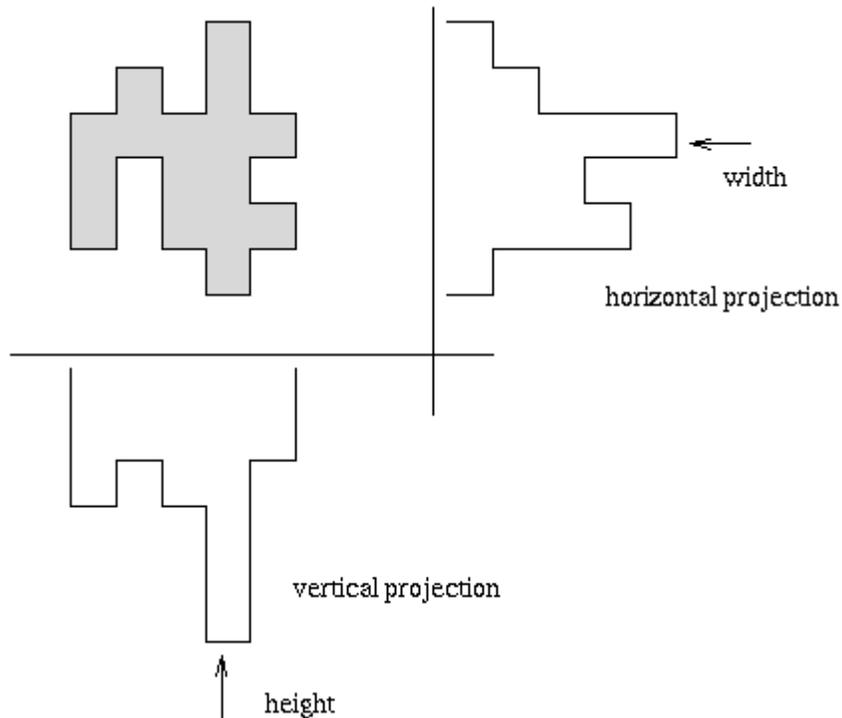


Fig 20: Projections

Region description by projections is connected to binary image processing. Projections can serve as a basis for definition of related region descriptors.

- **Eccentricity**

The simplest is the ratio of major and minor axes of an object. Another approximate eccentricity measure is based on a ratio of main region axes of inertia.

- **Elongatedness**

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 (Fig 2(a)).

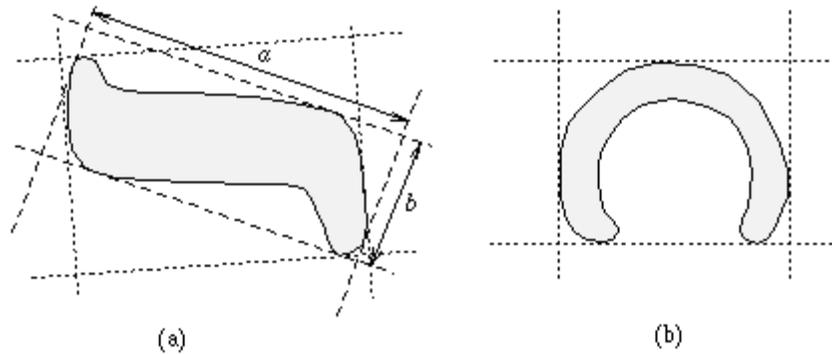


Fig 21: Elongatedness: (a) Bounding rectangle gives acceptable results; (b) Bounding rectangle cannot represent elongatedness

This criterion cannot succeed in curved regions (Fig 2(b)), 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) can be determined as the number  $d$  of erosion steps that may be applied before the region totally disappears.

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

- **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  $F_k$

$$rectangularity = \max_k(F_k)$$

- **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, the direction  $\theta$  can be computed as

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

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 independent of linear transformations

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

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, 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.

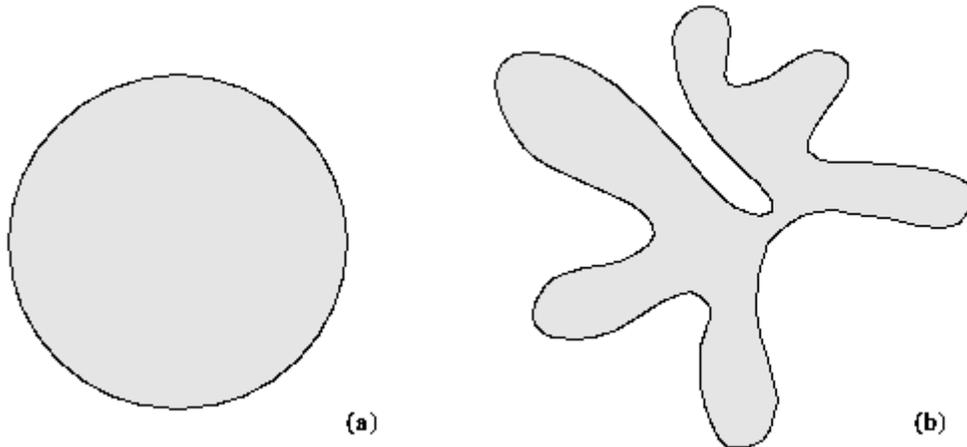


Fig 22: Compactness: (a) Compact; (b) Non-compact.

## 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**.
- 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

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

In digitized images we evaluate sums

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

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

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

or in digitized images

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

where  $x_c, y_c$  are the co-ordinates of the region's centroid

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}}$$

In the binary case,  $m_{00}$  represents the region area.

Scale invariant features can also be found in scaled central moments  $\eta_{pq}$

$$\eta_{pq} = \frac{\mu'_{pq}}{(\mu'_{00})^\gamma}, \quad \gamma = \frac{p+q}{2} + 1, \quad \mu'_{pq} = \frac{\mu_{pq}}{\alpha^{p+q+2}}$$

and normalized un-scaled central moments  $\vartheta_{pq}$

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

- Rotation invariance can be achieved if the co-ordinate system is chosen such that  $\mu_{11} = 0$ .
- A less general form of invariance is given by seven rotation, translation, and scale invariant moment characteristics

$$\varphi_1 = \vartheta_{20} + \vartheta_{02},$$

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

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

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

$$\varphi_5 = (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{30} + \vartheta_{12})((\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2) \\ + (3\vartheta_{21} - \vartheta_{03})(\vartheta_{21} + \vartheta_{03})(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2),$$

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

$$\varphi_7 = (3\vartheta_{21} - \vartheta_{03})(\vartheta_{30} + \vartheta_{12})((\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2) \\ - (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{21} + \vartheta_{03})(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2)$$

- While the seven moment characteristics presented above were shown to be useful, they are only invariant to translation, rotation, and scaling.
- A complete set of four affine moment invariants derived from second- and third-order moments is

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

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

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

$$I_4 = (\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2 \\ + 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} \\ - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} - 18\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} \\ + 9\mu_{20}\mu_{02}^2\mu_{21}^2 + 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{12} \\ + \mu_{02}^3\mu_{30}^2) / \mu_{00}^{11} .$$

- 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 can be estimated as

$$m_r = \frac{1}{N} \sum_{i=1}^N (z(i))^r ,$$

$$\mu_r = \frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^r .$$

- The  $r$ -th normalized contour sequence moment and normalized central contour sequence moment are defined as

$$\overline{m}_r = \frac{m_r}{\mu_2^{r/2}} = \frac{\frac{1}{N} \sum_{i=1}^N (z(i))^r}{\left( \frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2 \right)^{r/2}} ,$$

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

- Less noise-sensitive results can be obtained from the following shape descriptors

$$F_1 = \frac{(\mu_2)^{1/2}}{m_1} = \frac{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2\right)^{1/2}}{\frac{1}{N} \sum_{i=1}^N z(i)} ,$$

$$F_2 = \frac{\mu_3}{(\mu_2)^{3/2}} = \frac{\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^3}{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2\right)^{3/2}} ,$$

$$F_3 = \frac{\mu_4}{(\mu_2)^2} = \frac{\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^4}{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2\right)^2} ,$$

$$F_4 = \overline{\mu_5} .$$

## Convex hull

- A region  $R$  is convex if and only if for any two points  $x_1, x_2$  from  $R$ , the whole line segment defined by its end-points  $x_1, x_2$  is inside the region  $R$ .
- The convex hull of a region is the smallest convex region  $H$  which satisfies the condition  $R$  is a subset of  $H$ .
- 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.
- The convex hull can be used to describe region shape properties and can be used to build a tree structure of region concavity.

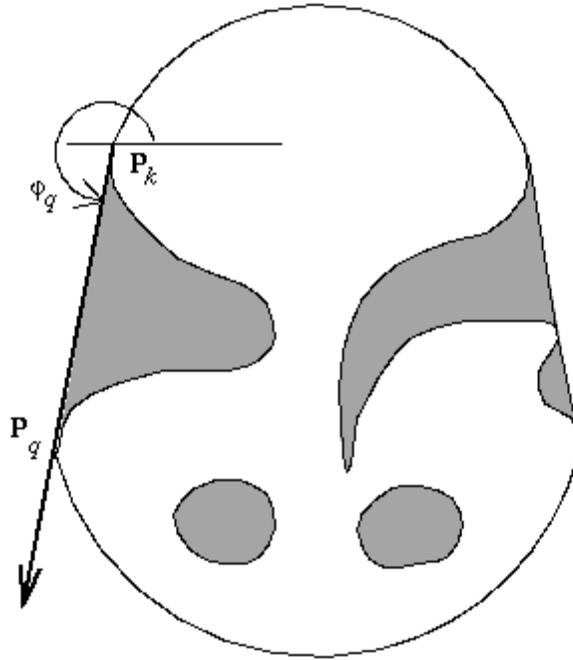


Fig 23: Convex hull

- 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: 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  $P_k = P_1$ ,  $v = (0, 1)$ ; the vector  $v$  represents the direction of the previous line segment of the convex hull.
2. Search the region boundary in an anti-clockwise direction and compute the angle orientation  $\varphi_n$  for every boundary point  $P_n$  which lies after the point  $P_1$  (in the direction of boundary search-see Fig 4). The angle orientation  $\varphi_n$  is the angle of vector  $P_k P_n$ . The point  $P_q$  satisfying the condition  $\varphi_q = \min_n \varphi_n$  is an element (vertex) of the region convex hull.
3. Assign  $v = P_k P_q$ ,  $P_k = P_q$ .
4. Repeat steps 2 and 3 until  $P_k = P_1$ .

- More efficient algorithms exist, especially if the object is defined by an ordered sequence of  $n$  vertices representing a polygonal boundary of the object.
- 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 with counter-examples published.
- The simplest correct convex hull algorithm was developed by Melkman and is now discussed further.
  - Let the polygon for which the convex hull is to be determined be a simple polygon  $P = v_1, v_2, \dots, v_n$  and let the vertices be processed in this order.
  - For any three vertices  $x, y, z$  in an ordered sequence, a directional function delta may be evaluated

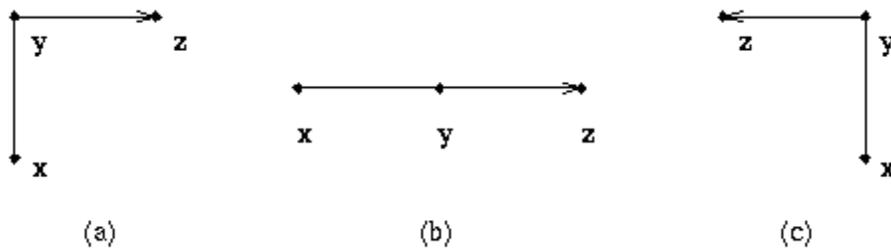


Fig 24: Directional function  $\delta$ . (a)  $\delta(x, y, z) = 1$ . (b)  $\delta(x, y, z) = 0$ . (c)  $\delta(x, y, z) = -1$ .

- The main data structure  $H$  is a list of vertices (deque) of polygonal vertices already processed.
- 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,  $H = \{d_b \dots d_t\}$  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.

- Main ideas of the algorithm:
  - The first three vertices  $A, B, C$  from the sequence  $P$  forms a triangle (if not collinear) and this triangle represents a convex hull of the first three vertices.
  - 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.
  - If  $D$  is outside of the current convex hull, it must become a new convex hull vertex and based on the current convex hull shape, either none, one, or several vertices must be removed from the current convex hull.
  - This process is repeated for all remaining vertices in the sequence  $P$ .

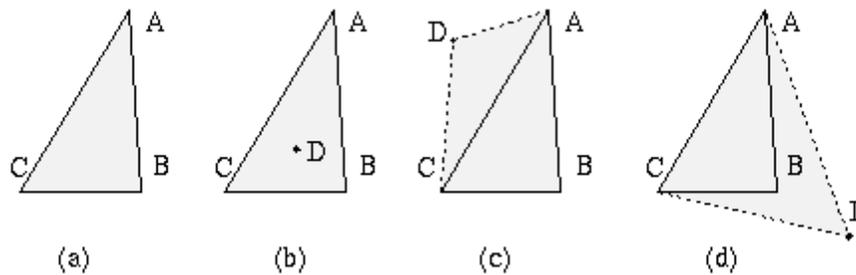


Fig 25: Convex hull detection. (a) First three vertices  $A, B, C$  forms a triangle. (b) If the next vertex  $D$  is positioned inside the current convex hull  $ABC$ , current convex hull does not change. (c) If the next vertex  $D$  is outside of the current convex hull, it becomes a new vertex of the new current convex hull  $ABCD$ . (d) In this case, vertex  $B$  must be removed from the current convex hull and the new current hull is  $ADCA$ .

- The variable  $v$  refers to the input vertex under consideration, and the following operations are defined:

*push*  $v$  :  $t := t + 1; d_t \rightarrow v,$

*pop*  $d_t$  :  $t := t - 1$

*insert*  $v$  :  $b := b - 1; d_b \rightarrow v,$

*remove*  $d_b$  :  $b := b + 1,$

*input*  $v$  : next vertex is entered from sequence  $P$ , if  $P$  is empty, stop,

- The algorithm is then;

## Algorithm: 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, top of the list.

- $while(\delta(d_{t-1}, d_t, v) \leq 0)$
- $pop\ d_t;$
- $push\ v;$

### 4. Rearrange vertices in, bottom of the list.

- $while(\delta(v, d_b, d_{b+1}) = 0)$
- $remove\ d_b$
- $insert\ v;$
- $go\ to\ step\ 2;$

- The algorithm as presented may be difficult to follow, however, a less formal version would be impossible to implement.
- The following example makes the algorithm more understandable.

Let  $P = \{A, B, C, D, E\}$  as shown in Fig 7(a). The data structure  $H$  is created in the first step;

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

In the second step, vertex D is entered (Fig 7(b));

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

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

$$\begin{aligned} \delta(B, C, D) &= -1 \rightarrow \text{pop } d_t \rightarrow \\ &\begin{array}{cccc} -1 & 0 & 1 & 2 \\ t, b \dots H &= C & A & B & C \\ & d_b & & & d_t \end{array} \end{aligned}$$

$$\begin{aligned} \delta(A, B, D) &= -1 \rightarrow \text{pop } d_t \rightarrow \\ &\begin{array}{cccc} -1 & 0 & 1 & 2 \\ t, b \dots H &= C & A & B & C \\ & d_b & d_t & & \end{array} \end{aligned}$$

$$\begin{aligned} \delta(C, A, D) &= 1 \rightarrow \text{push } D \rightarrow \\ &\begin{array}{cccc} -1 & 0 & 1 & 2 \\ t, b \dots H &= C & A & D & C \\ & d_b & & & d_t \end{array} \end{aligned}$$

In step (4)—Fig 26(c)

$$\begin{aligned} \delta(D, C, A) &= 1 \rightarrow \text{insert } D \rightarrow \\ &\begin{array}{ccccc} -2 & -1 & 0 & 1 & 2 \\ t, b \dots H &= D & C & A & D & C \\ & d_b & & & & d_t \end{array} \end{aligned}$$

Go to step (2); vertex E is entered – Fig 26(d)

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

- 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

$H = \{d_b, \dots, d_t\} = \{D, C, A, D\}$  which represents a polygon DCAD, always in the clockwise direction.

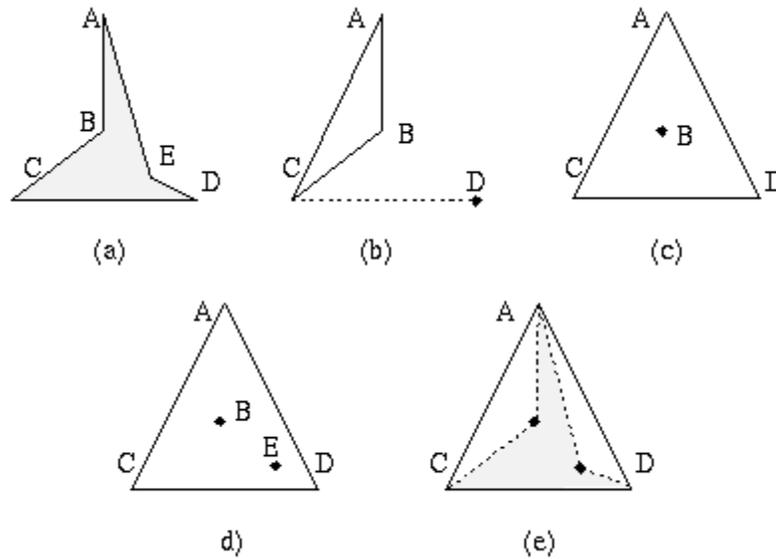


Fig 26: Example of convex hull detection. (a) The processed region-polygon ABCDEA. (b) Vertex D is entered and processed. (c) Vertex D becomes a new vertex of the current convex hull ADC. (d) Vertex E is entered and processed E does not become a new vertex of the current convex hull. (e) The resulting convex hull DCAD.

- A **region concavity 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.

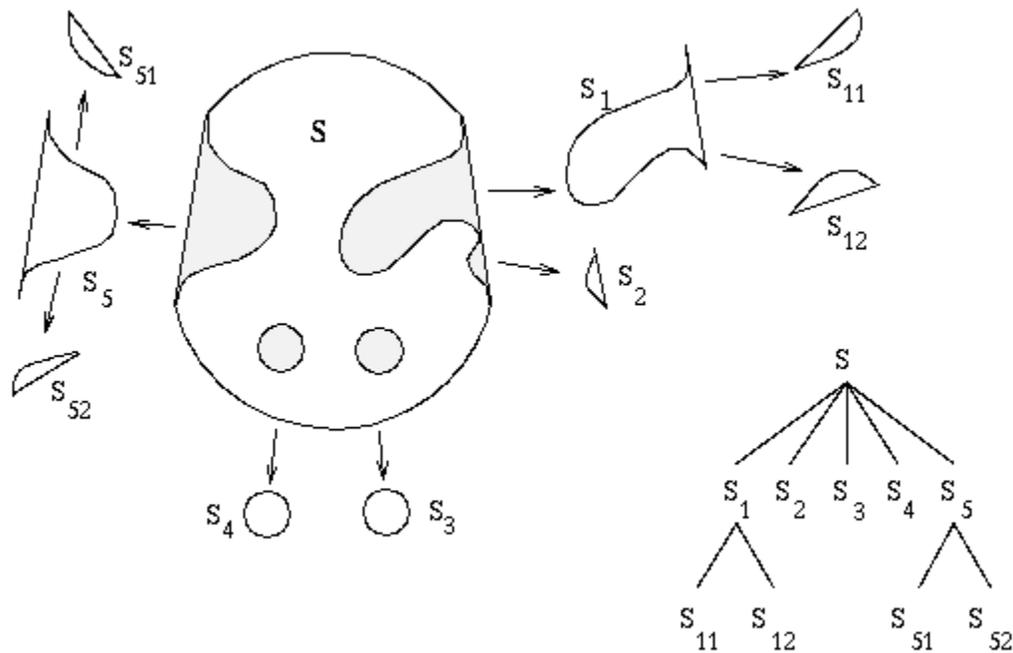


Fig 27: Concavity tree construction. (a) Convex hull and concave residua. (b) Concavity tree.

### Graph representation based on region skeleton

- Objects are represented by a planar graph with nodes representing sub regions resulting from region decomposition, and region shape is then described by the graph properties.
- There are two general approaches to acquiring a graph of sub regions:
  - 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 sub regions, which are then represented by nodes while arcs represent neighborhood relations of sub regions.
- 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
- 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.
- 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: 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  $R_{old} = R$ .
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 the next Figure.
- If there are skeleton segments which have a thickness of two, 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.

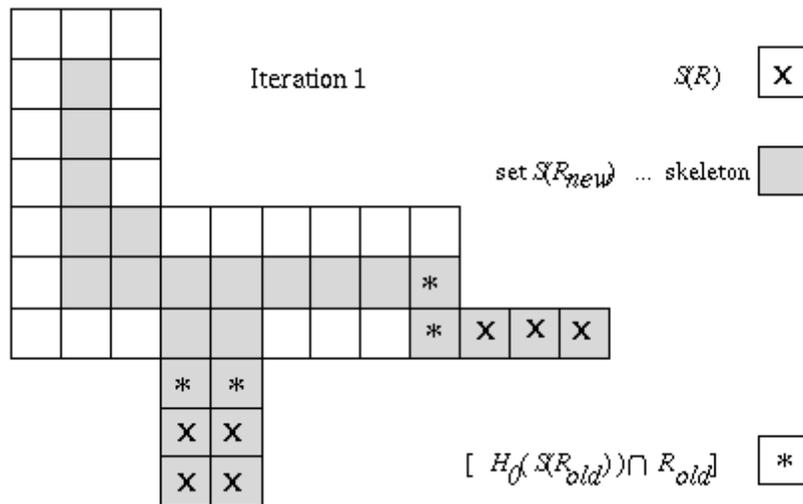


Fig 28: Skeleton by thinning

- 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.
- 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.
- 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.
- 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.
- Small changes in the boundary may cause serious changes in the skeleton.
- 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 approach to skeleton construction may also result in decreased sensitivity to boundary noise.
- Similarly, the approach using the Marr-Hildreth edge detector with varying smoothing parameter facilitates scale-based representation of the region's skeleton.

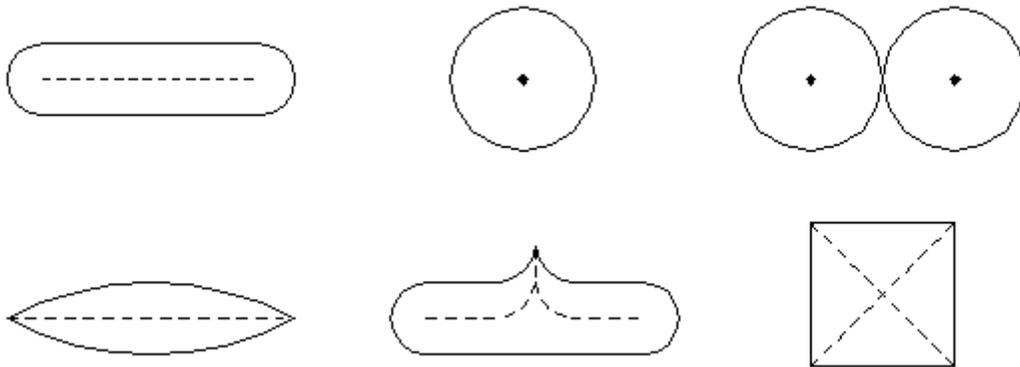


Fig 29: Region skeletons; small change in border can have a significant effect on the skeleton

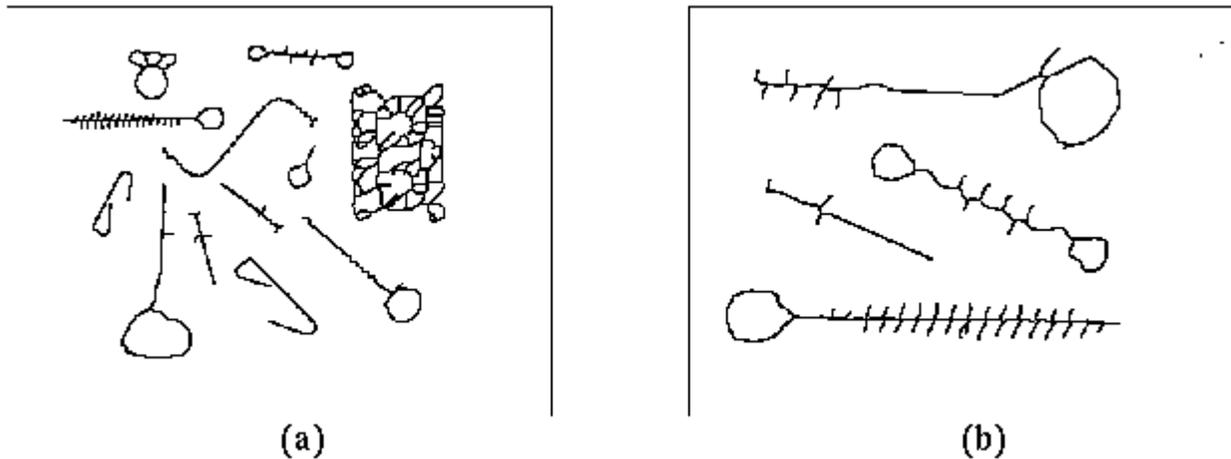


Fig 30: Region skeletons: Thickened for visibility.

- 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: Region graph construction from skeleton

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

### 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.

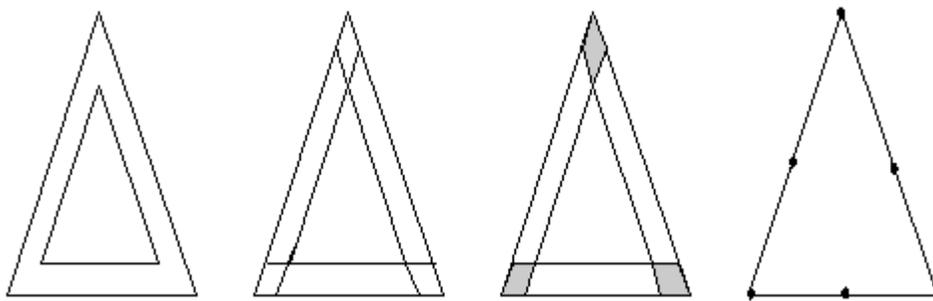


Fig 31: Region decomposition (a) Region. (b) Primary regions. (c) Primary sub-regions and kernels. (d) Decomposition graph.

- The solution to the decomposition problem consists of two main steps:
  - The first step is to segment a region into simpler sub regions (primitives)
  - Second is the analysis of primitives.
- Primitives are simple enough to be successfully described using simple scalar shape properties.
- If sub regions are represented by polygons, graph nodes bear the following information;
  1. Node type representing primary sub region or kernel.
  2. Number of vertices of the sub region represented by the node.
  3. Area of the sub region represented by the node.
  4. Main axis direction of the sub region represented by the node.
  5. Center of gravity of the sub region 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.

### **Region neighborhood graphs**

- Any time region decomposition into sub regions or image decomposition into regions is available, the region or image can be represented by a region neighborhood graph (the region adjacency graph 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 quad tree image representation, from run-length encoded image data, etc.

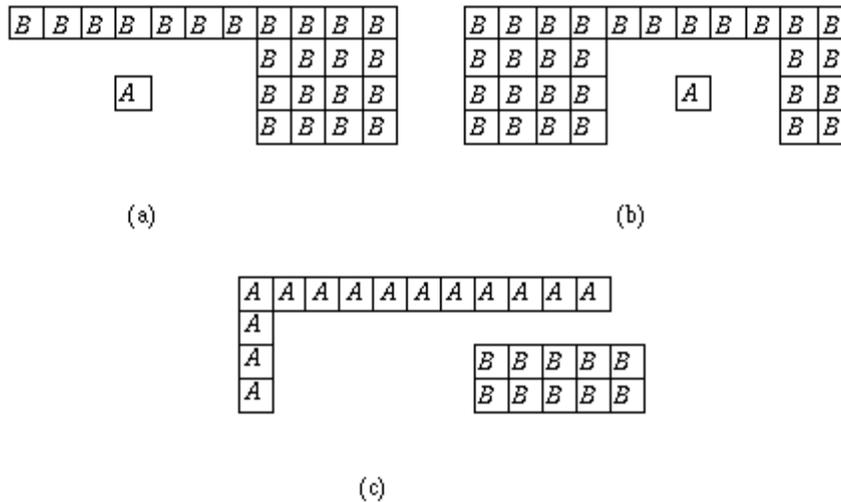


Fig 32: Binary relation.

- 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, 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

# Shape Classes

## Shape classes

- Representation of **shape classes** is considered a challenging problem of shape description.
- 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.
- 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).
- 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.
- Cootes et al. determine principal components in the shape space 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**.