# Shape Representation

# 6

*The Creator has a model for every creation.*

## 6.1 Introduction

A shape is a binary image. Mathematically, it is defined as

$$f(x,y) = \begin{cases} 1 & if\ (x,y) \in D \\ 0 & otherwise \end{cases} \tag{6.1}$$

where $D$ is the domain or area of the binary image.

Most of the objects in this world can be identified by their shapes, such as fruits, trees, plant leaves, buildings, furniture, birds, fishes, etc. Figure 6.1 shows a few examples of shape images, the first two are shapes with a contour while the last two are shapes with interior content.

A shape can be defined by its boundary/contour like Fig. 6.1a, b, or by its interior content like Fig. 6.1c, d. There are a variety of shape methods; they can be generally grouped into either contour-based method or region-based method. A number of perceptual shape descriptors have also been proposed to capture both contour and region features. The design of a shape descriptor usually follows the principles suggested by MPEG-7: good retrieval accuracy, compact features, general application, low computation complexity, robust retrieval performance (affine invariance and noise resistance), and hierarchically coarse to fine representation.

However, shape description is a difficult task because it is difficult to define perceptual shape features and measure the similarity between shapes. To make the problem more complex, the shape is often corrupted with noise, defection, arbitrary distortion, and occlusion.



D. Zhang, *Fundamentals of Image Data Mining*, Texts in Computer Science, https://doi.org/10.1007/978-3-030-17989-2_6

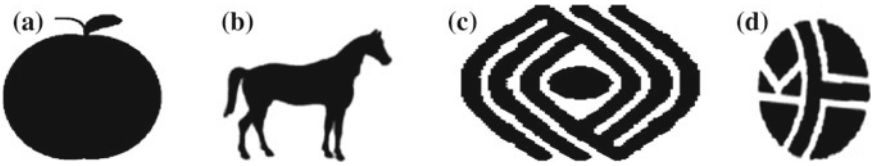



**Fig. 6.1** Examples of shape images

## 6.2 Perceptual Shape Descriptors

There are a number of simple shape descriptors which can be computed according to human perception, such as perimeter, area, compactness, Euler number, circularity, eccentricity, major axis orientation, bending energy, convexity, etc. These individual shape descriptors can be combined to create a more powerful descriptor to represent a shape.

### 6.2.1 Circularity and Compactness

*Circularity* represents how a shape is close to a circle, it also indicates how compact the shape is or not. Mathematically, it is defined as the ratio of the area of a shape ($A_s$) to the area of a circle ($A_c$) having the same perimeter:

$$C = A_s/A_c \tag{6.2}$$

Assume the perimeter of the shape is $p$, the area of the circle with the same perimeter $p$ is given by

$$A_c = p^2/4\pi \tag{6.3}$$

Therefore,

$$C = 4\pi A_s/p^2 \tag{6.4}$$

By this definition, the shape with the largest circularity is a circle with circularity of 1. Since $4\pi$ is a constant, circularity can be simply defined as

$$C = A_s/p^2 \tag{6.5}$$

*Rectangularity* and *ellipse variance* can be defined in a similar way to circularity.

The circularity descriptor defined in this way can cause confusions in certain situations. For example, the following two shapes would have the same

**Fig. 6.2** Two different shapes with same circularity

circularity or compactness according to the above definition, although perceptually they look very different (Fig. 6.2).

Furthermore, the above circularity is too sensitive to noise and irregularity. Therefore, a more robust circularity descriptor has also been defined. It is defined as the following ratio:

$$C = \sigma_R / \mu_R \tag{6.6}$$

where $\mu_R$ stands for the mean of the radial distance from the centroid of the shape to shape boundary points and $\sigma_R$ stands for the standard deviation of the radial distance from the centroid to shape boundary points.

### 6.2.2 Eccentricity and Elongation

*Eccentricity* is defined as the ratio of the length of the longest chord of the shape to the longest chord perpendicular to it. Eccentricity can be computed using either the principle axes method (Fig. 6.3a) or the minimum bounding box method (Fig. 6.3b).

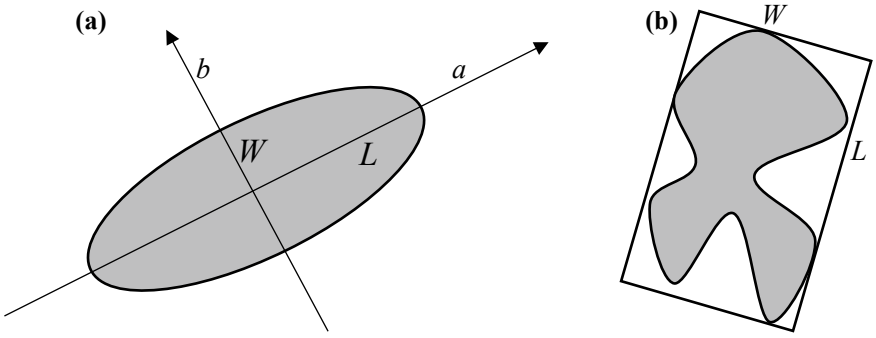



**Fig. 6.3** Computation of Eccentricity. **a** Eccentricity with principle axes; **b** eccentricity with minimum bounding box

**Fig. 6.4** A curled eel with 0 elongation

In the above figure, the eccentricity of the shapes is given by

$$E = L/W \tag{6.7}$$

The principle axis can also be found using the Principle Component Analysis (PCA) method. *Eccentricity* indicates the *elongation* of a shape, the larger the eccentricity the more elongated the shape.

*Elongation* is defined as

$$El = 1 - W/L \tag{6.8}$$

$0 \leq El \leq 1$. A circle, a square or any symmetric shape would have the least elongation (0), while objects like eels, poles, road, etc. would have an elongation close to 1. *Elongation*, however, can fail when an elongated shape is bent. For example, a curled eel (Fig. 6.4) would have a small or even 0 elongation although perceptually it is still an elongated object.

### 6.2.3 Convexity and Solidarity

A region is convex if for any two points with the region, the entire line segment linked by the two points are also inside the region. A convex hull of a shape is the smallest convex region that includes the shape (Fig. 6.5).

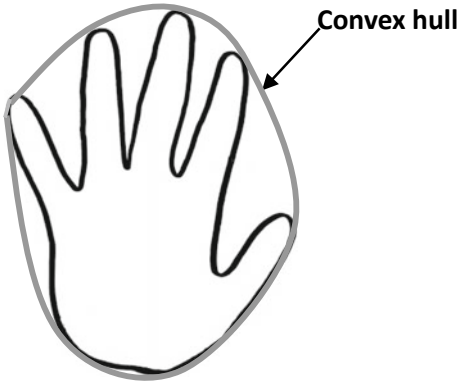



**Fig. 6.5** A hand shape and its convex hull

*Convexity* is then defined as the ratio of the perimeter of the convex hull of the shape ($P_h$) to the perimeter of the shape ($P_s$).

$$Convexity = P_h/P_s \tag{6.9}$$

*Solidarity* is defined as the ratio of the area of the shape ($A_s$) to the area of its convex hull ($A_h$).

$$Solidarity = A_s/A_h \tag{6.10}$$

The convexity and solidarity of a convex shape are always 1, convexity and solidarity of non-convex shapes are smaller than 1.

### 6.2.4 Euler Number

Topology is the study of the properties which are unaffected by any deformation (e.g., rubber-sheet distortion). It is found that when a shape is deformed, the number of holes does not change by the deformation. Therefore, a useful topological descriptor is the *Euler number* which is defined as the difference between the number of holes $H$ and the number of connected components $C$. A small *Euler number* indicates more holes in a shape.

$$En = C - H \tag{6.11}$$

For example, the Euler numbers of number 3, letter A and B are 1, 0, and −1, respectively (Fig. 6.6).

A Hole Area Ratio (*HAR*) can also be defined in relation to Euler number:

$$HAR = A_h/A_s \tag{6.12}$$

where $A_h$ is the total area of holes and $A_s$ is the area of the shape.

**Fig. 6.6** Shapes with different Euler numbers

### 6.2.5 Bending Energy

The *bending energy* (*BE*) is defined by [1]

$$BE = \frac{1}{N}\sum_{t=0}^{N} K(t)^2$$

and

$$K(t) = (\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t))/(\dot{x}^2(t) + \dot{y}^2(t))^{3/2} \tag{6.14}$$

where $K(t)$ is the curvature function, and $N$ is the number of points on a contour. In order to compute a robust bending energy, the shape boundary is usually Gaussian smoothed before the *BE* calculation. It can be shown that a circle is the shape having the minimum bending energy.

The shape descriptors described in this section are computed according to human perception of the shape patterns. The advantage of using them is that they usually have a semantic meaning. However, the downside of these descriptors is that they are usually sensitive as shown in the circularity and eccentricity sections. It's difficult to describe a shape effectively using a single shape descriptor. Therefore, these perceptual descriptors are usually used as filters to eliminate shapes of large difference. They are often used together with other more powerful descriptors described in the following sections.

## 6.3 Contour-Based Shape Methods

Contour shape techniques only exploit shape boundary information. There are generally two types of very different approaches for contour shape modeling: *continuous approach* (global) and *discrete approach* (structural) [1]. *Continuous approaches* do not divide shape into subparts, and a multidimensional feature vector derived from the integral boundary is used to describe the shape. It starts to derive a 1D continuous function, called shape signature, from the shape boundary. After that, variety of techniques from signal processing, time series, and statistics can be used to extract a feature vector from the shape signature. The matching between shapes is a straightforward process, which is usually a calculation of the Euclidean distance or city block distance between feature vectors.

*Discrete approaches* break the shape boundary into segments, called *primitives* using a particular criterion. The final representation is usually a string or a graph (or tree), the similarity measure is done by string matching or graph matching.

### 6.3.1 Shape Signatures

The first step of contour-based shape methods is to obtain a 1D function from the shape boundary points, called shape signature. Many shape signatures exist, including *complex coordinates, polar coordinates, central distance, tangent angle, cumulative angle, curvature, area*, and *chord length*.

In general, a *shape signature* $u(t)$ is any 1D function representing 2D areas or boundaries. A shape signature captures the perceptual feature of the shape, it uniquely describes a shape. In the following, we assume the shape boundary coordinates $(x(t), y(t))$, $t = 0, 1, \ldots, N-1$, have been extracted in the preprocessing stage, $t$ usually means arclength. The preprocessing usually consists of a denoising procedure or a smoothing procedure and a contour tracing procedure.

#### 6.3.1.1 Position Function

*Position function*, or *complex coordinates*, is simply the complex number generated from the boundary coordinates:

$$z(t) = [x(t) - x_c] + i[y(t) - y_c] \tag{6.15}$$

where $(x_c, y_c)$ is the centroid of the shape, which is the average of the boundary coordinates

$$x_c = \frac{1}{N}\sum_{t=0}^{N-1} x(t), \quad y_c = \frac{1}{N}\sum_{t=0}^{N-1} y(t) \tag{6.16}$$

$z(t)$ is a complex number which captures the spoke features of a shape boundary. $z(t)$ is a translation invariant signature due to the subtraction of the centroid. Rotation causes a circular shift to $z(t)$, and scaling of shape introduces linear change in $z(t)$. The use of position function as shape signature involves little computation. However, the position function needs to be further processed for matching, for example, a centroid distance signature can be computed from $z(t)$.

#### 6.3.1.2 Centroid Distance

The *centroid distance* function is defined as the magnitude of $z(t)$, and it is expressed by the distance of the boundary points to the centroid $(x_c, y_c)$ of the shape [1]

$$r(t) = \left([x(t) - x_c]^2 + [y(t) - y_c]^2\right)^{1/2} \tag{6.17}$$

Same as $z(t)$, $r(t)$ is also invariant to translation. Rotation causes $r(t)$ circular shift and scaling of shape changes $r(t)$ by a linear term. Different from $z(t)$ however, $r(t)$ is a real function which can be used for matching two shapes directly. Figure 6.7 (top) shows the centroid distance signatures of a tree shape. Due to the use of the centroid as the reference point, both $z(t)$ and $r(t)$ attenuate protruding features of a

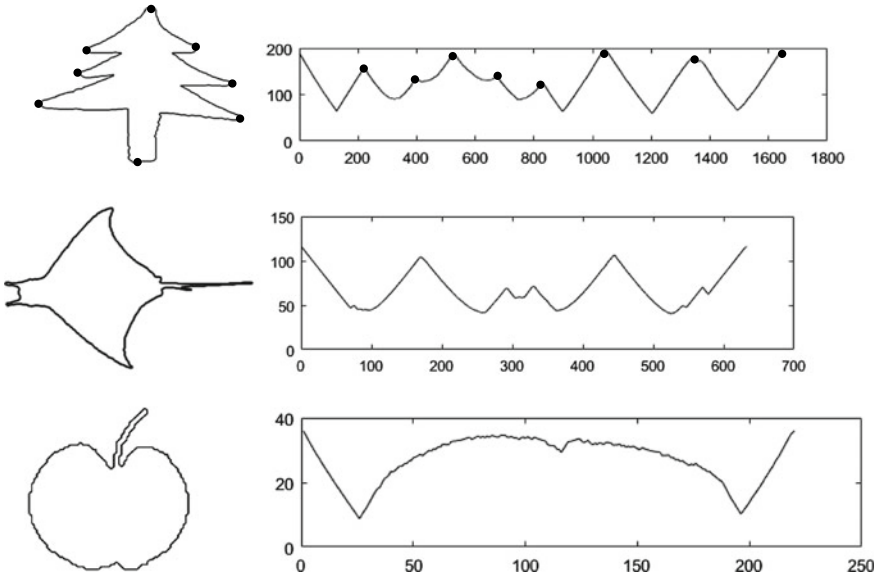



**Fig. 6.7** Examples of centroid distance signatures. Top row: a tree shape on the left and its centroid distance signature on the right; middle row: a ray fish on the left and its signature on the right; bottom row: an apple shape on the left and its signature on the right

shape. As can be seen, the $r(t)$ function can generally capture the variations of a shape boundary well, like the eight protruding corners on the tree boundary (marked by black dots). However, it does not capture the most prominent feature in a shape well if it is too thin like the tail of fish or the tail of an apple in Fig. 6.7 (center and bottom).

Another problem with $r(t)$ is that it uses the centroid as reference point. For shapes with high irregularity or low compactness, the centroid often falls outside the shape body. Consequently, the structure of the shape cannot be preserved by $r(t)$ when the shape is under distortion. For example, Fig. 6.8a and b are, respectively, the $r(t)$ functions of two sea snakes, it can be seen that the number of peaks in Fig. 6.8b is doubled compared with that of Fig. 6.8a.

In order to overcome the reference point problem of $r(t)$, *chord length signature* (CLS) $r^*(t)$ has been proposed. The chord length function $r^*(t)$ is derived from shape boundary without using any reference point. For each boundary point $\mathbf{P}(t)$, its $r^*(t)$ is defined as the distance between $\mathbf{P}$ and another boundary point $\mathbf{P}'$ such that $\mathbf{PP}'$ is perpendicular to the tangent vector at $\mathbf{P}$ and $|\mathbf{PP}'|$ is the shortest chord if there are more than one $\mathbf{P}'$s perpendicular to $\mathbf{P}$ (Fig. 6.8c). $r^*(t)$ is invariant to translation. Rotation causes circular shift to $r^*(t)$. Scaling causes linear changes to $r^*(t)$.

$r^*(t)$ is more sensitive to noise than $r(t)$ due to the numeric approximation in computing the tangents at each boundary point and the angles needed to find the chords. To reduce noise sensitivity, an average filter or a median filter can be used to smooth the shape boundary before the signature extraction.

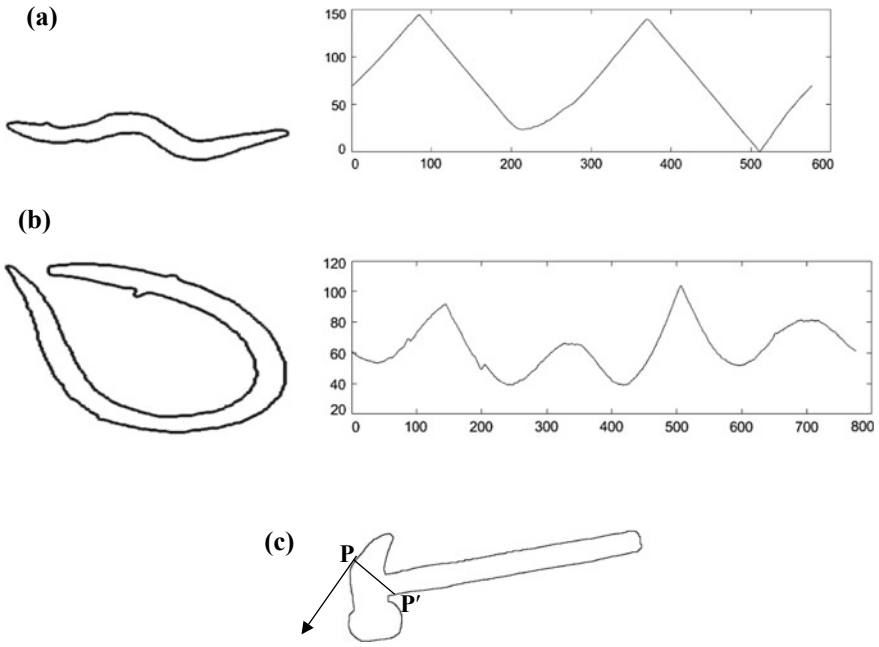



**Fig. 6.8** Computation of chord length signature. **a** A sea snake shape on the left and its $r(t)$ function on the right; **b** another sea snake shape on the left and its $r(t)$ function on the right; **c** illustration of computing $r^*(t)$ at point **P** of a hammer shape

#### 6.3.1.3 Angular Functions

Intuitively, the tangent angles of the shape boundary indicate the change of angular directions of the shape boundary. The change of angular directions is important to human perception. Therefore, shape can be represented by its boundary tangent angles

$$\theta(t) = \arctan\frac{y(t) - y(t-w)}{x(t) - x(t-w)} \tag{6.18}$$

where $w$, an integer, is a jump step used in practice to smooth the boundary. However, the tangent angle function $\theta(t)$ can only assume values in a range of length $\pi$, usually in the interval of $[-\pi/2, \pi/2]$. Therefore, $\theta(t)$ in general contains vertical jump/drop discontinuities at $\theta(t) = \pm\pi/2$ (Fig. 6.9b). One solution is to use the absolute function $|\theta(t)|$ instead of $\theta(t)$, as shown in Fig. 6.9c, the vertical jump/drop discontinuities are removed from $|\theta(t)|$ and the structure of the shape is also preserved. However, the signature function still has sharp corners. Another solution is to use a *cumulative angular function* $\varphi(t)$ which is the net amount of angular bend between the starting position $z(0)$ and position $z(t)$ on the shape boundary

$$\varphi(t) = \theta(t) - \theta(0) \tag{6.19a}$$

The computation of $\varphi(t)$ is illustrated in Fig. 6.9f. As shown in Fig. 6.9d, there is no vertical jump/drop in $\varphi(t)$ at places where $\theta(t) = \pm\pi/2$, and the two sharp angular changes on the heart boundary has been accurately captured. Because of the accumulation, $\varphi(t)$ has captured a linear trend in the function, therefore, a $\psi(t')$ (Fig. 6.9d) can be created by normalizing $t$ into $[0, 2\pi]$ using $t' = \frac{2\pi}{L}t$ and taking away a linear term $t'$ from $\varphi(t')$ if it is obtained in counter clockwise order (or adding $t'$ if it is obtained in clockwise order).

$$\psi(t') = \varphi\left(\frac{L}{2\pi}t'\right) - t' \tag{6.19b}$$

where $t \in [0, L]$, $L$ is the length of the shape boundary and $t' \in [0, 2\pi]$. The cumulative angular signature $\psi$ $(t)$ is invariant to both translation and scaling. Rotation causes a shift in the signature.

#### 6.3.1.4 Curvature Signature

Curvature is an important boundary feature. It is used widely for shape representation in the literature. *Curvature function* is given by (6.20a, 6.20b, 6.20c):

$$\kappa(t) = \frac{d\theta}{dt} \tag{6.20a}$$

$$= \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{6.20b}$$

$$= \frac{\frac{d^2y}{dx^2}}{\left(1 + \left(\frac{dy}{dx}\right)^2\right)^{3/2}} \tag{6.20c}$$

where $\theta$ is defined in (6.18). A perfect circular shape would have a constant $\kappa(t)$ based on the definition of (6.20a). Curvature is an important boundary feature, however, due to $\theta(t)$ is typically piecewise continuous and jumps at discontinuities, $\kappa(t)$ is zero almost everywhere and jumps at where $\theta(t)$ jumps. For example, Fig. 6.10 (Top) is the $\kappa(t)$ of the tree shape in Fig. 6.7, while it successfully captures the seven shape corners on the shape boundary, it is jaggy. In order to use $\kappa(t)$ for shape representation, a shape boundary needs to be smoothed before curvature extraction. One way to smooth the shape boundary using a Gaussian filter (Fig. 6.10 (Bottom)).

Curvature is invariant to translation, rotation causes circular shift to the signature, and curvature signature is invariant to scaling if all shapes are normalized to the same number of points.

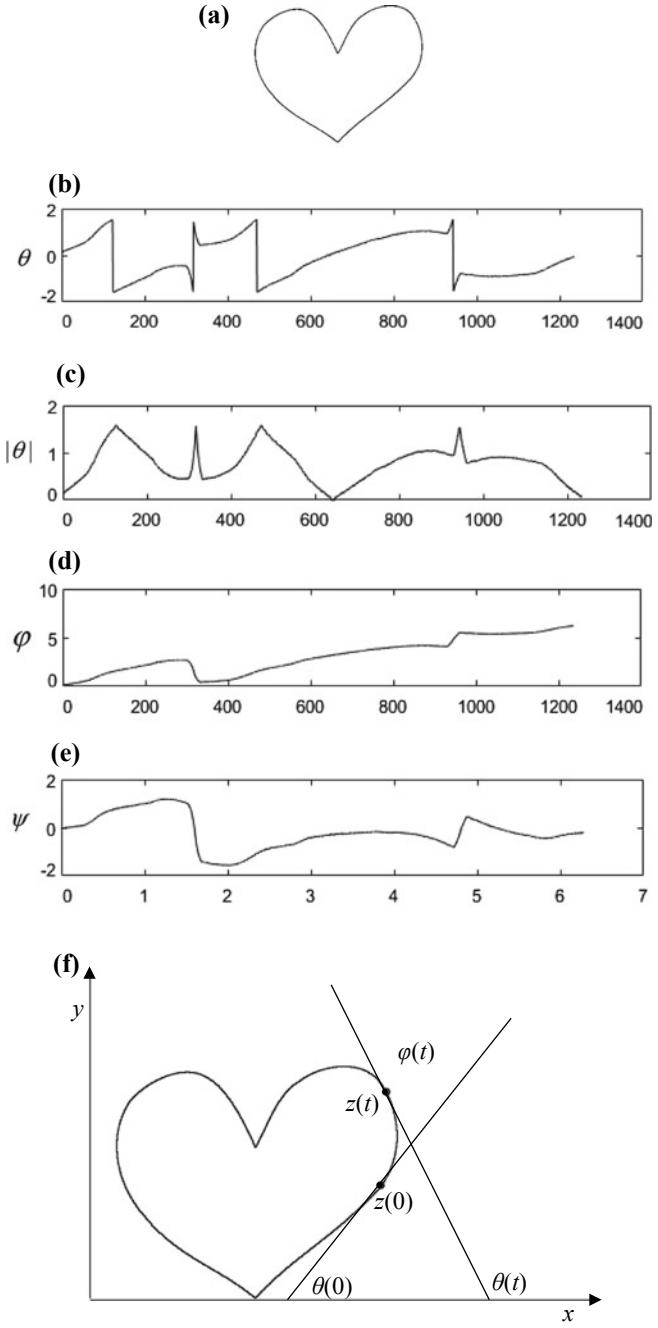



**Fig. 6.9** Computation of angular signatures. **a** A heart shape; **b** $\theta(t)$ of (**a**); **c** $|\theta(t)|$ of (**a**); **d** $\varphi(t)$ of (**a**); **e** $\psi(t)$ of (**a**); **f** illustration of the computation of $\varphi(t)$

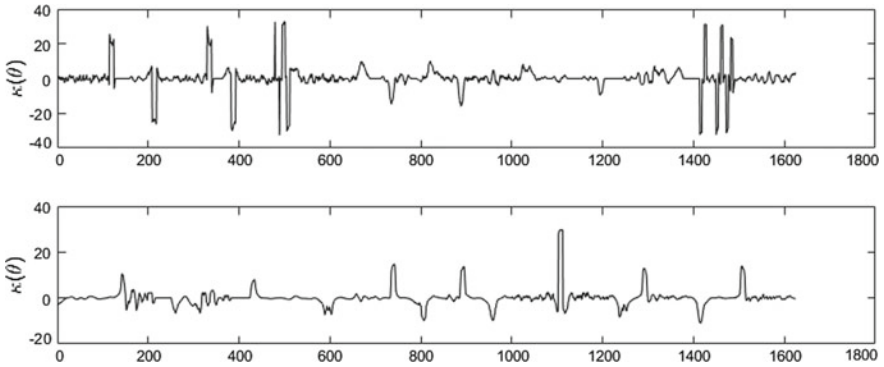



**Fig. 6.10** Curvature signatures. Top: curvature signature of a tree shape from Fig. 6.7 without smoothing; Bottom: curvature signature of the same tree shape with a Gaussian smoothing

#### 6.3.1.5 Area Function

It is known that the area of a triangle changes linearly under affine transformation. Linearity is a desirable property for shape representation because normalization of linearity is equivalent to scale normalization which is simple. Therefore, an area signature is used in attempt to acquire a signature invariant to affine distortion [1].

When the boundary points change along the shape boundary, the area of the triangle formed by the two boundary points and the center of gravity also changes (Fig. 6.11a). For each boundary points, the area of the triangle with $\alpha$ degree angle at vertex **o** is calculated (Fig. 6.11b). This forms an *area function* which can be employed as shape representation.

For the triangle $\Delta\mathbf{OP}_1\mathbf{P}_2$ formed by $\mathbf{O}$, $\mathbf{P}_1$, and $\mathbf{P}_2$ in Fig. 6.11b, its area is given by the following difference:

$$\Delta \mathrm{OP}_1\mathrm{P}_2 = \Delta \mathrm{OP}_2 x_2 - \Delta \mathrm{OP}_1 x_1 - \square x_1 \mathrm{P}_1 \mathrm{P}_2 x_2$$

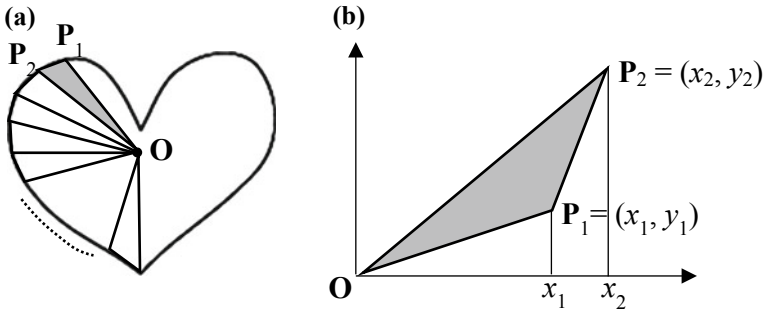



**Fig. 6.11** Computation of area signature. **a** Triangulation of a heart shape; **b** illustration on the calculation of the area of the shaded triangle

Mathematically, the area is given by (6.21):

$$\begin{aligned} A(t) &= \frac{1}{2}x_2y_2 - \frac{1}{2}x_1y_1 - \frac{1}{2}(x_2 - x_1)(y_2 - y_1) - (x_2 - x_1)y_1 \\ &= \frac{1}{2}|x_1y_2 - x_2y_1| \end{aligned} \tag{6.21}$$

Because the area of a triangle and the central distance inside the triangle has a linear relationship, $A(t)$ is similar to $r(t)$ [1], however, due to the numerical approximation of the area of a triangle, $A(t)$ is usually more jaggy than $r(t)$. Therefore, $A(t)$ needs to be smoothed for further feature extraction. $A(t)$ is linear under affine transformation.

#### 6.3.1.6 Discussions

Shape signatures reduce shape matching in 2D space into 1D space, this reduces the complexity of feature extraction. Shapes are usually normalized to be translation and scale invariant before signature extraction. Translation invariance is achieved by either using a reference point such as the centroid or using relative positions such as in the cases of curvature, angles and chord length. Scale invariance is achieved by first scaling all shape images to the same size, and then normalizing shape boundaries to the same number of points. If shape signatures are used for shape matching, a shift matching is needed to find the best matching between two shapes. Alternatively, a signature can be quantized into a signature histogram, which is rotation invariant and can be used for matching.

Shape signatures are generally sensitive to noise and irregularities. Therefore, a shape boundary is typically smoothed before signature extraction to remove noise and small irregularities. However, significant irregularities can still cause large error in the matching, therefore, it is impractical to use shape signature for direct representation. Further processing is necessary to improve both matching efficiency and accuracy. The following sections describe methods on feature extraction from shape signatures and other robust contour shape methods.

### 6.3.2 Shape Context

The idea of shape context is similar to the shape signature methods discussed in Sect. 6.3.1. It also computes contour features from a shape boundary point by point. However, instead of computing a single feature value for each boundary point in the shape signature methods, a feature vector (histogram) is computed in shape context. Furthermore, the computation of the feature vector of each boundary point makes use of all the points on the boundary instead of just two points in the shape signature methods. This makes the shape context features more robust to boundary irregularities. The algorithm of shape context computation is summarized in the following:

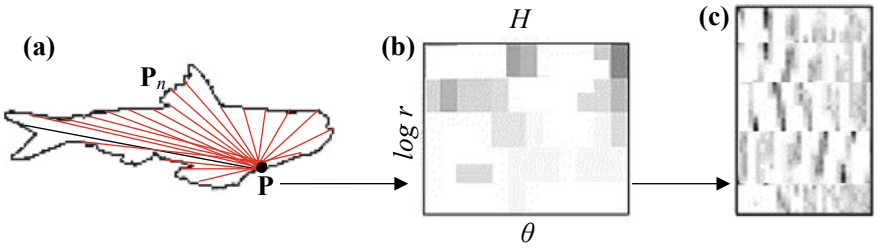



**Fig. 6.12** Computation of shape context. **a** A point **P** on a shape boundary and all the vectors started from **P**; **b** the log-polar histogram $H$ of the vectors from **P**, the histogram $H$ is the context of point **P**; **c** the shape context map of shape of (**a**), each row of the context map is the flattened histogram of a point context, the number of rows is the number of sampled points

1. Normalize a shape boundary to $N$ points;
2. For each of the boundary points **P**, find the vectors between **P** and all the other boundary points $\mathbf{P}_n$ (Fig. 6.12a);
3. Quantize both the angles $\theta$ and the length $r$ of the vectors at **P**;
4. Create a 2D *logarithmic* histogram $H$ of $r$ on $\theta$ for each point **P** (Fig. 6.12b);
5. Flat the 2D histogram $H$ into a 1D histogram $h$ by concatenating the rows of $H$;
6. Concatenate $h$'*s* from all the boundary points $\mathbf{P}_n$ to form a histogram map which is the shape context (Fig. 6.12c).

The logarithm of $r$ is to raise the contribution from neighboring points of **P** which would otherwise contribute too little due to significantly shorter vector length than that of points farther away from **P**. The process of computing a shape context is shown in Fig. 6.12 [2]. In Fig. 6.12a, a point **P** on the shape boundary and its vectors to all the boundary points $\mathbf{P}_n$ are shown. Figure 6.12b shows the logarithmic histogram of the vectors in Fig. 6.12a, c shows the shape context map/matrix of the concatenated flatten histograms from all boundary points.

Shape context is invariant to translation due to the use of relative point position. Scaling invariance can be achieved by normalizing all shape boundaries to N points and normalizing the radial distances by the mean distance between all point pairs. Rotation invariance is done by finding the minimum of all shift matching of two shape context maps.

The matching of two shape contexts is complex. It minimizes the total cost of matching between one context matrix and all the permutations of another context matrix. This can affect the robustness of shape context significantly.

### 6.3.3 Boundary Moments

Moments can be computed to reduce the dimension of a boundary representation. Assume shape boundary has been represented as a *shape signature* $z(i)$, the $r$th moment $m_r$ and central moment $\mu_r$ can be estimated as [1]

$$m_r = \frac{1}{N}\sum_{i=1}^{N}[z(i)]^r \quad \text{and} \quad \mu_r = \frac{1}{N}\sum_{i=1}^{N}[z(i) - m_1]^r \tag{6.22}$$

where $N$ is the number of boundary points. The normalized moments $\bar{m}_r = m_r/(\mu_2)^{r/2}$ and $\bar{\mu}_r = \mu_r/(\mu_2)^{r/2}$ are invariant to shape translation, rotation and scaling.

Boundary moments can also be computed from the boundary histogram. Suppose the amplitude of shape signature function $z(i)$ is quantized and a histogram $p(v_i)$ is created from the quantized $z(i)$. Then, the $r$th moment is obtained by

$$\mu_r = \sum_{i=1}^{K}(v_i - m)^r p(v_i) \quad \text{and} \quad m = \sum_{i=1}^{K} v_i p(v_i) \tag{6.23}$$

The advantage of boundary moment descriptors is that they are simple to compute and they are more robust than a shape signature. However, only a few low-order moments have physical meaning. In practice, the following three moment descriptors are usually used for shape description: $F_1 = (\mu_2)^{1/2}/m_1$, $F_2 = \mu_3/(\mu_2)^{3/2}$, and $F_3 = \mu_4/(\mu_2)^2$, which describes the variance, skewness, and kurtosis of the boundary.

### 6.3.4 Stochastic Method

Time-series models and especially autoregressive (AR) modeling have been used for calculating shape descriptors. A linear autoregressive model expresses a value of a function $f(x)$ as a linear combination of a certain number of preceding values. Specifically, each function value in the sequence has some correlation with previous function values and can, therefore, be predicted through a number of, say, $M$ observations of previous function values. The autoregressive model is a simple prediction of the current radius by a linear combination of $M$ previous radii plus a constant term and an error term:

$$f_t = \alpha + \sum_{j=1}^{m}\theta_j f_{t-j} + \sqrt{\beta}\omega_t \tag{6.24}$$

where $\theta_j$, $j = 1, 2, \ldots, m$ are the AR-model coefficients, $m$ is the model order, it tells how many preceding function values the model uses. $\sqrt{\beta}\omega_t$ is the current error term or residual, reflecting the accuracy of the prediction. $\alpha$ is proportional to the mean of function values. The parameters $\{\alpha, \theta_1, \ldots, \theta_m, \beta\}$ are estimated by using the *least square* (LS) criterion. The estimated $\{\theta_j\}$ are translation, rotation, and scale invariant. Parameters $\alpha$ and $\beta$ are not scale invariant. But the quotient $\alpha/\sqrt{\beta}$, which

reflects the signal-to-noise ratio of the boundary, is regarded as an invariant. Therefore, the feature vector $\left[\theta_1, \ldots, \theta_m, \alpha/\sqrt{\beta}\right]$ is used as the shape descriptor.

The AR descriptors can capture the cyclic patterns of shape, it works well for a regular and smooth shape. However, AR is an optimization process, for irregular and complex shapes, it may not have a solution. Furthermore, the choice of $m$ is a complicated problem and is usually decided empirically.

### 6.3.5 Scale Space Method

#### 6.3.5.1 Scale Space

The problem of noise sensitivity and boundary variations in most spatial domain shape methods inspire the use of scale space analysis. The scale space representation of a shape is created by tracking the position of *inflection points* in a shape boundary or signature. This is done by repeatedly applying low-pass Gaussian filters of variable widths $\sigma$ at the signature function $f(x)$.

$$L(x;\sigma) = g(x;\sigma) * f(x) \tag{6.25}$$

The inflection points that remain present in the Gaussian filtered signature functions are expected to be "significant" object characteristics. The result is usually an interval tree, called "fingerprint", consisting of inflection points shown in Fig. 6.12 [3]. As can be seen from Fig. 6.12a, as the scale $\sigma$ goes up, the number of inflection points on the function $f(x)$ decreases. However, at each point of the function, the inflection point disappears at different scale, this feature has been captured in the interval tree shown in Fig. 6.13b.

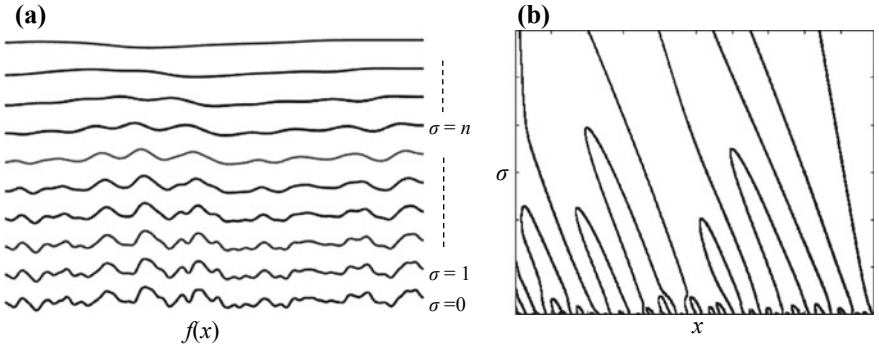



**Fig. 6.13** Shape signature in scale space. **a** An original signature function $f(x)$ at the bottom and its successively smoothed versions on the top of it (up to scale 512), where $\sigma$ is the scale of the smoothed function; **b** the interval tree derived from the zero-crossings of the second derivatives of the smoothed functions at the left, each $(x, \sigma)$ in the interval tree corresponds to a zero-crossing point at position $x$ and scale $\sigma$ of the function at left-hand side

### 6.3.5.2 Curvature Scale Space

The difficulty with scale space method is the interpretation of the interval tree. Mokhtarian and Mackworth [4] developed a *curvature scale space* (CSS) descriptor by finding the peaks of the interval tree. The computation of the CSS descriptor consists of two procedures [5]. The first is to compute a CSS contour map or the interval tree. The second is to extract the branch peaks from the interval tree.

*Algorithm of computing CSS contour map*:

1. Normalize shape to a fixed number of boundary points;
2. Create an array $ZC[\ ][\ ]$ to record curvature zero-crossing points at each scale;
3. Set $\sigma = 0$;
4. Compute curvatures of each position at scale $\sigma$;
5. Record each curvature zero-crossing point at current scale $\sigma$ to $ZC[\sigma][x]$;
6. Set $\sigma = \sigma + 1$;
7. Smooth the boundary with a Gaussian filter $g(x; \sigma)$;
8. Repeat step 3–7 until no curvature zero-crossing points are found;
9. Plot $ZC[\sigma][x]$ onto a Cartesian space to create CSS contour map.

*Algorithm of extracting CSS contour peaks*:

1. Scanning from the top row of CSS contour map;
2. If a zero-crossing point is found at a location $(i, j)$, check the above neighbor points $(i - 1, j - 1)$, $(i - 1, j)$, and $(i - 1, j + 1)$. If the three above neighbor points are nonzero-crossing points, then the location $(i, j)$ is a peak candidate; find all the peak candidates in row $i$;
3. For each peak candidate $(i, j)$ at row $i$, check its neighbor peak candidates, if a neighbor candidate $(i, k)$ is found over five points away, then $(i, j)$ is a peak. If a neighbor candidate is found within five points, there is a peak at the middle $(i, (j + k)/2)$;
4. Repeat step 2 and 3 for each row, until all the CSS peaks are found.

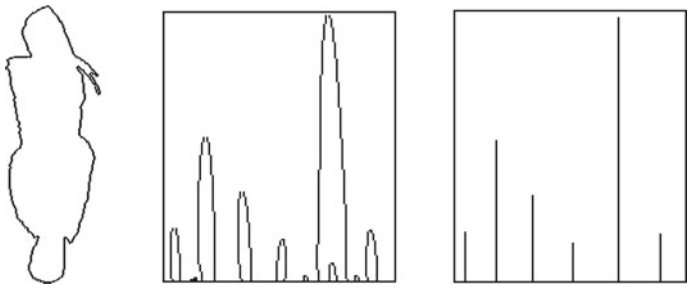


**Fig. 6.14** Computation of curvature scale space. A fish shape (left), its CSS contour map (center) and CSS peaks (right)

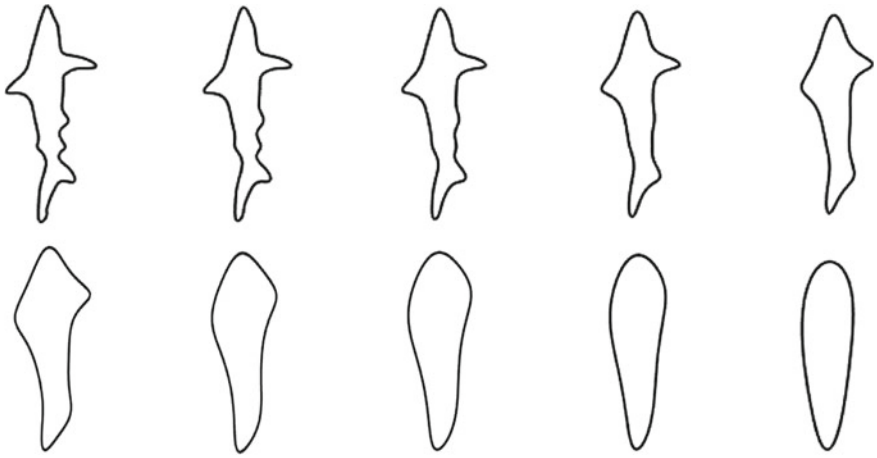


**Fig. 6.15** The evolution of shape boundary as scale $\sigma$ increases

Figure 6.14 shows an example of a fish shape, its CSS contour and peaks [1]. The CSS contour successfully captures the seven major corners of the shape.

The morphing of the fish shape in Fig. 6.14 during the Gaussian smoothing process is shown in Fig. 6.15. As can be seen from Fig. 6.15, as the scale $\sigma$ goes up, the shape becomes smoother and smoother until it is completely smoothed.

The matching of two CSS descriptors is complex. Assume all shapes have been normalized into the same number of boundary points, the peaks need to be normalized and circularly shifted to find the best match between two shapes. However, due to the sensitivity of both the height and position of the highest peaks, certain tolerance needs to be accepted during the matching of two peaks. The complex matching can affect the performance significantly. In practice, CSS is combined with a few robust single descriptors such as compactness, elongation, etc. [5]. But this introduces a new issue on the weight given to each type of descriptors.

### 6.3.6 Fourier Descriptor

For any 1D signature function $f(x)$ derived from Sect. 6.3.1, its discrete Fourier transform is given by

$$a_n = \frac{1}{N}\sum_{t=0}^{N-1} f(x)\exp(-j2\pi nx/N), \quad n = 0, 1, \ldots, N-1 \tag{6.26}$$

This results in a set of Fourier coefficients $\{a_n\}$, which is a representation of the shape. Since shapes generated through rotation, translation, and scaling (called *similarity transformation*) of the same shape are similar shapes, a shape

representation should be invariant to these operations. The selection of different start point on the shape boundary to derive $f(x)$ should not affect the representation.

The magnitudes of Fourier coefficients $|a_n|$ are invariant to rotation and starting point because rotation and starting point only affect the phases of the coefficients. Translation invariance can be achieved by normalizing a shape to its center of gravity or subtraction of the mean. Scale invariance is done by normalizing the magnitudes of the coefficients by the DC component $a_0$, which is the average energy of the signal and is the largest coefficient:

$$|b_n| = |a_n|/a_0, \quad n = 1, 2, \ldots, N-1 \tag{6.27}$$

The set of magnitudes of the normalized Fourier coefficients $\{|b_n|,\ 1 < n < N-1\}$ are used as the shape descriptor, denoted as FD: $\{\mathrm{FD}_n,\ 1 < n < N-1\}$.

FD has several desirable features compared with other shape descriptors. First, it is efficient to compute due to the use of fast Fourier transform or FFT. Second, it provides a coarse to fine representation of a shape. Figure 6.16 examples show how a shape can be reconstructed or represented to a different level of details using different number of Fourier coefficients [1]. In practice, only a small number of low-frequency FDs are used to describe a shape to reduce the sensitivity to noise and irregularities.

Third, all FDs have physical meaning, they capture the different frequency components or different level of details of a shape boundary. Fourth, the matching of two FDs are very simple, it is done by either the city block or Euclidean distance. Because of these desirable features, FD is one of the most robust shape descriptors for contour based shapes. The study by Zhang et al. [5] shows that FD outperforms CSS descriptor which has been adopted by MPEG-7.


**Fig. 6.16** Reconstructed shapes of an apple shape using Fourier coefficients from **a** $r(t)$; **b** $z(t)$. In both (**a**) and (**b**), from left to right, the shapes are reconstructed using 5, 10, 20, 30, and all the Fourier coefficients, respectively

### 6.3.7 Discussions

The global shape descriptors described above are basically a mathematical summarization of the boundary samples. However, just like any mathematics, it is based on ideal assumptions and constraints. Typically, these descriptors only work on ideal applications where objects are located in isolation and the boundary of the objects can be viewed or kept in completeness. If the objects overlap each other or certain part of the object boundary is missing, these descriptors do not work anymore. For example, an apple with a bite is still perceived as an apple, like the famous Apple Inc. logo. But using the global shape descriptors, the bite on the apple will change the mathematical summarization dramatically and cause a mismatch. In such kind of applications, *structural shape methods* can be used to analyze the structure of a shape boundary and match shapes using part of the shape boundary. In the following, different types of *structural shape methods* are described in details.

### 6.3.8 Syntactic Analysis

Syntactic analysis is inspired by the phenomenon that composition of a natural scene is analog to the composition of a language, that is, sentences are built up from phrases, phrases are built up from words and words are built up from alphabets, etc. [6, 7]. In syntactic methods, a shape is represented with a set of predefined primitives. The set of predefined primitives is called the *codebook* and the primitives are called *codewords*. For example, given the codewords in the right of Fig. 6.17, the hammer shape at the left can be represented as a grammatical string of $S$:

$$S = a\ b\ b\ b\ c\ b\ b\ c\ b\ d\ b\ b \tag{6.28}$$

The matching between shapes can use string matching by finding the minimal number of edit operations to convert one string into another.

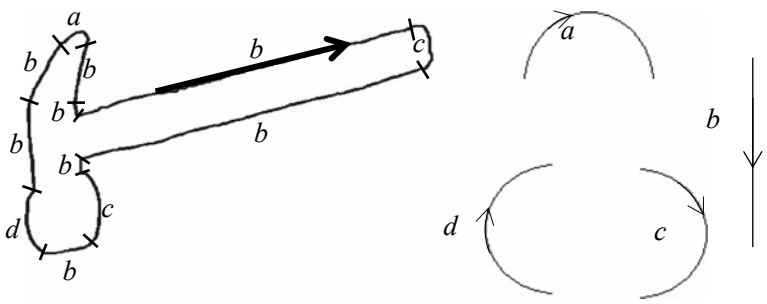



**Fig. 6.17** Syntactic analysis of a hammer shape

A more general method is to formulate the representation as a string grammar. Each primitive is interpreted as an alphabet of some grammar, where grammar is a set of rules of syntax that govern the generation of sentences formed from symbols of the alphabet. The set of sentences generated by a grammar, $G$ is called its language and is denoted as $L(G)$. Here, sentences are strings of symbols (which in turn represent patterns), and languages correspond to pattern class. After grammar has been defined, the matching is straightforward. For a sentence representing an unknown shape, the task is to decide in which language the shape represents a valid sentence.

In practice, however, it is difficult to infer a pattern grammar which can generate only the valid patterns. In addition, this method needs a priori knowledge of the database in order to define codewords or alphabets. The knowledge, however, is usually unavailable.

### 6.3.9 Polygon Decomposition

Polygon can be used to capture the overall shape of a contour and discard the minor variations or noise along the shape boundary. In general, there are two methods to create a polygon from a shape contour: merging and splitting, both are based on applying a distance threshold on the cumulated distance (or errors) between the shape boundary and the polygon line segments [8].

Merging methods add successive pixels to a line segment if each new pixel that is added doesn't cause the segment to deviate too much from a straight line.

In the merging method, it chooses one point as a starting point on the contour, for each new point to be added, let a line segment go from the starting point to this new point. Then, the total squared error of all the boundary points to the line segment is computed. If the error exceeds some threshold, the line from the starting point to the previous point is kept and new line segment is started. For example, in the Fig. 6.18 [8], to find out if the boundary points between $\mathbf{P}_i$ and $\mathbf{P}_k$ should be merged, the total distance or error of all the boundary points to the line segment $\mathbf{P}_i\mathbf{P}_k$ is computed. If the error is larger than the threshold, keep $\mathbf{P}_i\mathbf{P}_j$ and add a new line segment $\mathbf{P}_j\mathbf{P}_k$. The distance $d_j$ from $\mathbf{P}_j$ to $\mathbf{P}_i\mathbf{P}_k$ is given as (6.29) [9]. $d_j$ is equal

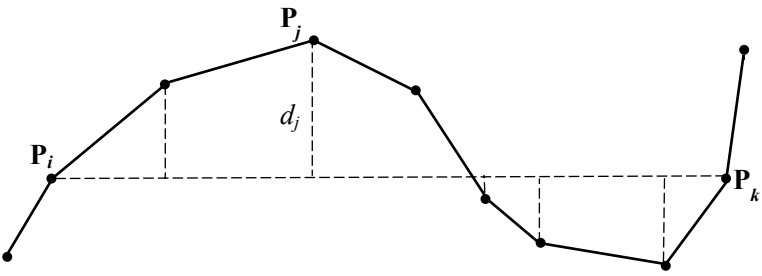



**Fig. 6.18** Polygon approximation by merging

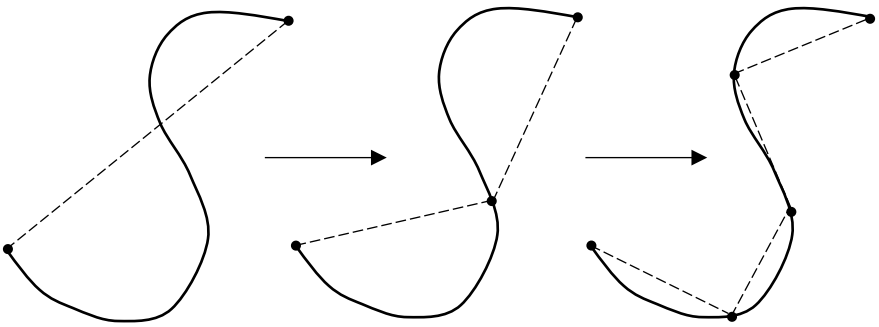


**Fig. 6.19** Polygon approximation by splitting

to twice the area of triangle $\Delta \mathbf{P}_i\mathbf{P}_j\mathbf{P}_k$ divided by the distance between $\mathbf{P}_i\mathbf{P}_k$ and the area of $\Delta \mathbf{P}_i\mathbf{P}_j\mathbf{P}_k$ is given in (6.21).

$$d_j = \frac{\left|(x_k - x_i)(y_j - y_i) - (x_j - x_i)(y_k - y_i)\right|}{\sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}} \tag{6.29}$$

Splitting methods work by first drawing a line from the start point of the boundary to the end point of the boundary. Then, the perpendicular distance from each point along the boundary to the line is computed. If this exceeds some threshold, the boundary is broken into two segments with equal length. The process is repeated for each of the two new segments until no boundary segment needs to be broken (Fig. 6.19).

Once a polygon has been approximated, each segment of the polygon is regarded as a primitive and can be described by its *length* and *angle* in relation to the previous segment. Other features can also be used such as *length ratio* and *triangle area* between two adjacent segments. The polygon is then represented as a string of primitives. The length of each primitive is normalized by the shape boundary to achieve scale invariance.

The matching between two shapes involves shift and best match. Typically, the matching between shapes involves two steps: feature-by-feature matching in the first step and model-by-model matching (shape-by-shape matching) in the second step. In the first step, given a feature(s) of a query shape, the feature(s) is searched through the indexed database, if a particular model feature in the database is found to be similar to the query feature(s), the list of shapes associated with the model feature is retrieved. In the second step, the matching between the query shape and a retrieved model is matched based on the editing distance between the two string of primitives.

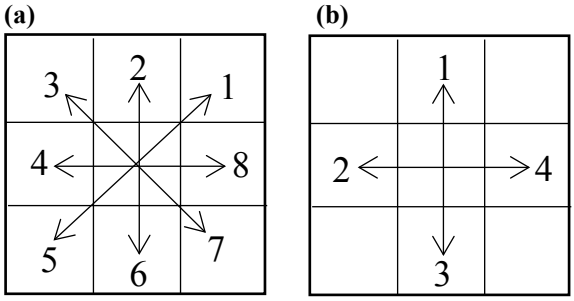



**Fig. 6.20** Computation of chain code. **a** Chain code in eight-connectivity; **b** chain code in four-connectivity

### 6.3.10 Chain Code Representation

Chain codes describe an object by a sequence of unit-size line segments with a given orientation. In the implementation, a digital boundary of an image is superimposed with a grid, the boundary points are approximated to the nearest grid point, then a sampled image is obtained. From a selected starting point, a chain code can be generated by using a four-connectivity or an eight-connectivity chain code (Fig. 6.20).

Chain code is invariant to translation because the code values are based on directions only. Rotation invariance can be achieved by finding the pixel in the border sequence which results in the minimum integer number, that pixel is then used as the starting pixel. For matching, a *chain code histogram* (of directions) normalized by the chain code length is used. This not only reduces the dimensions and sensitivity to noise but also achieves scaling invariance.

Chain code is itself a fine polygon of equal side length and can be further merged to create a coarse polygon of different side length. In other words, chain code can be used to create a polygon of a shape.

### 6.3.11 Smooth Curve Decomposition

A contour shape can also be segmented into boundary segments using curvature threshold. The idea is to first smooth the boundary with a Gaussian filter and then calculate the curvature at each point of the smoothed boundary. The boundary is then segmented at points where the curvature exceeds the threshold [10]. The boundary segments are then regarded as the primitives. An example is shown in Fig. 6.21.

The feature for each primitive is its maximum *curvature* and its *orientation*, and the similarity between two primitives is measured by the weighted Euclidean distance. Shapes in database are then indexed with the primitives. Shapes can then be matched using the two steps matching used in the polygon matching.

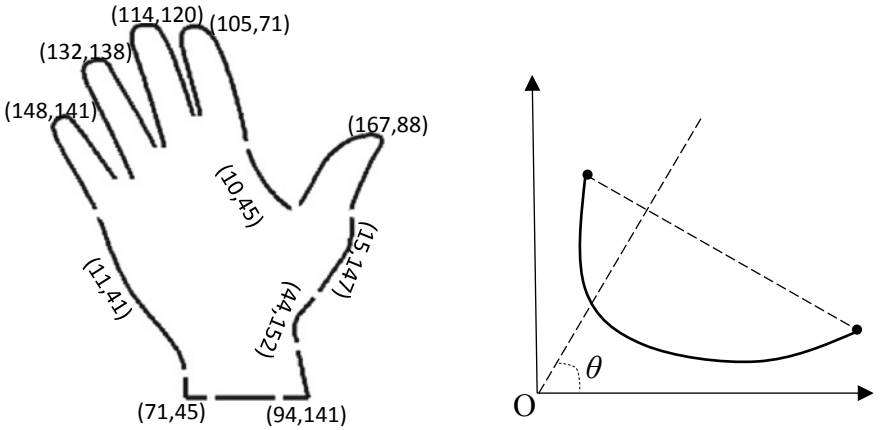



**Fig. 6.21** Smooth curve decomposition. Left: a horse shape and its boundary segments; right: the calculation of the angle of a segment

### 6.3.12 Discussions

The advantage of structural approach is its capability of handling *occlusion* problem in the scene and allowing partial matching. However, structural approach suffers from ambiguity of primitives, expensive computation, and complex matching.

Both global and structural approaches are sensitive to boundary noise and irregularity. This can be overcome by extracting features from shape interior content, or using region-based feature extraction approaches.

## 6.4 Region-Based Shape Feature Extraction

In region-based methods, all the pixels within a shape region are taken into account to obtain shape features. Common region-based feature extractions methods are based on shape moments. Other region-based methods include grid method, shape matrix, convex hull, and media axis.

### 6.4.1 Geometric Moments

Mathematically, *geometric moments* are projections of a function onto a polynomial basis in a similar way to the FT which is a projection onto a basis of harmonic sinusoid functions. The geometric moment of order $(p + q)$ of a general function $f(x, y)$ is given as (6.30)

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy, \quad p,q = 0,1,2,\ldots \tag{6.30}$$

Two simple properties can be derived from geometric moment:

$$Mass = M_{00} \tag{6.31}$$

$$\text{Centroid} = \left\{ \bar{x} = \frac{M_{10}}{M_{00}}, \bar{y} = \frac{M_{01}}{M_{00}} \right\} \tag{6.32}$$

The *central moments* of order $p + q$ of a shape image $f(x, y)$ are given by

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x,y) \quad p,q = 0,1,2\ldots \tag{6.33}$$

The normalized central moments $\mu_{pq}/\mu_{00}$ are invariant to both translation and scaling. The following properties can be observed from geometric moments.

- $(M_{10}/M_{00}, M_{01}/M_{00})$ defines the *center of gravity* or *centroid* of a shape.
- $M_{20}$ and $M_{02}$ describe the distribution of mass of the shape with respect to the coordinate axes. They are also called the *moments of inertia*.
- $\mu_{10}/\mu_{00}$ and $\mu_{01}/\mu_{00}$ are the *horizontal mean* and *vertical mean* of the shape, respectively.
- $\mu_{20}/\mu_{00}$ and $\mu_{02}/\mu_{00}$ are the *horizontal variance* and *vertical variance* of the shape, respectively.
- $\mu_{11}$ is the *covariance* of the shape.
- $\mu_{30}/\mu_{00}$ and $\mu_{03}/\mu_{00}$ represent the *horizontal skewness* and *vertical skewness* of the shape, respectively. The *skewness* measures the *symmetry* of a shape. The skewness of a symmetric shape equals to zero.
- $\mu_{40}/\mu_{00}$ and $\mu_{04}/\mu_{00}$ represent the *horizontal kurtosis* and *vertical kurtosis* of the shape, respectively. The kurtosis measures the *peakedness* or *sharpness* of the pixel distribution inside the shape.

These are important properties to describe how pixels are distributed inside the shape. Since both $x^p$ and $y^q$ are monomials, they amplify the moment values of pixels farther away from the centroid, higher order moments reflect how dramatic a shape changes in relation to its centroid.

However, geometric moments are not rotation invariant and it's difficult to derive moment invariants of high order. Hu [11] has derived seven *moment invariants* up to order three:

$$\left.\begin{aligned}
\Phi_1 &= \eta_{20} + \eta_{02} \\
\Phi_2 &= (\eta_{20} - \eta_{02})^2 + 4(\eta_{11})^2 \\
\Phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\Phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\Phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\
&\quad + (3\eta_{21} - \eta_{03})\{\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
\Phi_6 &= (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\Phi_7 &= (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \\
&\quad + (3\eta_{12} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]
\end{aligned}\right\} \tag{6.34}$$

where $\eta_{pq} = \mu_{pq}/(\mu_{00})^{\gamma}$ and $\gamma = 1 + (p + q)/2$ for $p + q = 2, 3, \ldots$.

The geometric moment transform can be extended to generalized form by replacing the conventional transform kernel $x^p y^q$ with a more general kernel of $P_p(x)$ $P_q(y)$.

### 6.4.2 Complex Moments

The rotation invariance issue of the geometric moments can be easily addressed by using *complex moments* and polar sampling because the magnitude of a complex function is invariant to rotation and polar sampling is also invariant to rotation. The idea is to replace the real polynomials in the geometric moment with complex polynomials and sample the shape in a polar coordinate system. The general form of complex moments is defined as (6.35)

$$C_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x + jy)^p (x - jy)^q f(x, y) dx dy \tag{6.35}$$

where $j = \sqrt{-1}$. The orthogonal *Zernike moments* are derived from Zernike polynomials:

$$V_{nm}(x, y) = V_{nm}(\rho \cos\theta, \rho \sin\theta) = R_{nm}(\rho) \exp(jm\theta) \tag{6.36}$$

where

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n - s)!}{s!\left(\frac{n+|m|}{2} - s\right)!\left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s} \tag{6.37}$$

where $\rho$ is the radius from $(x, y)$ to the shape centroid, $\theta$ is the angle between $\rho$ and $x$-axis, $n$ and $m$ are integers and subject to $n - |m| = even$, $|m| \leq n$. Zernike

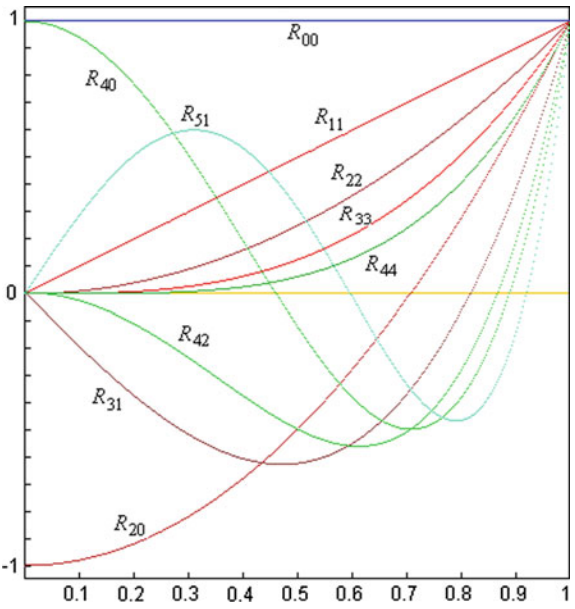



**Fig. 6.22** The first ten real Zernike polynomials

polynomials are a complete set of complex-valued function orthogonal over the unit disk, i.e., $x^2 + y^2 = 1$. Therefore, a shape is first normalized into a unit disk to compute Zernike moments.

For example, the first six real Zernike polynomials are given in the following:

$$\begin{array}{lll} R_{00}(\rho) = 1, & R_{11}(\rho) = \rho, & R_{20}(\rho) = 2\rho^2 - 1 \\ R_{22}(\rho) = \rho^2, & R_{31}(\rho) = 3\rho^3 - 2\rho, & R_{33}(\rho) = \rho^3 \end{array}$$

The first 10 real Zernike polynomials of up to order 5 is shown in Fig. 6.22.

The complex Zernike moments of order $n$ with repetition $m$ are thus defined as

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V_{nm}^*(x,y), \quad x^2 + y^2 \leq 1 \tag{6.38}$$

or

$$A_{nm} = \frac{n+1}{\pi} \sum_\rho \sum_\theta f(\rho\cos\theta, \rho\sin\theta) R_{nm}(\rho) \exp(jm\theta), \quad \rho \leq 1 \tag{6.39}$$

where * means complex conjugate. Due to the constraint of $n - |m| = even$ and $m < n$, there are $[n/2]$ repetition of moments in each order $n$. As an example, the first 36 Zernike moments of up to order 10 are given in Table 6.1. The table only shows the moments with positive $m$, the moments of negative $m$ are just the rotational versions of the moments of positive $m$. It can be seen from the table, from

**Table 6.1** List of Zernike moments up to order 10

| Order ($n$) | Zernike moment of order $n$ with repetition $m$ ($A_{nm}$) | Number of moments in each order $n$ | Total number of moments up to order 10 |
|---|---|---|---|
| 0 | $A_{0,\ 0}$ | 1 | 36 |
| 1 | $A_{1,\ 1}$ | 1 | |
| 2 | $A_{2,\ 0}$, $A_{2,\ 2}$ | 2 | |
| 3 | $A_{3,\ 1}$, $A_{3,\ 3}$ | 2 | |
| 4 | $A_{4,\ 0}$, $A_{4,\ 2}$, $A_{4,\ 4}$ | 3 | |
| 5 | $A_{5,\ 1}$, $A_{5,\ 3}$, $A_{5,\ 5}$ | 3 | |
| 6 | $A_{6,\ 0}$, $A_{6,\ 2}$, $A_{6,\ 4}$, $A_{6,\ 6}$ | 4 | |
| 7 | $A_{7,\ 1}$, $A_{7,\ 3}$, $A_{7,\ 5}$, $A_{7,\ 7}$ | 4 | |
| 8 | $A_{8,\ 0}$, $A_{8,\ 2}$, $A_{8,\ 4}$, $A_{8,\ 6}$, $A_{8,\ 8}$ | 5 | |
| 9 | $A_{9,\ 1}$, $A_{9,\ 3}$, $A_{9,\ 5}$, $A_{9,\ 7}$, $A_{9,\ 9}$ | 5 | |
| 10 | $A_{10,\ 0}$, $A_{10,\ 2}$, $A_{10,\ 4}$, $A_{10,\ 6}$, $A_{10,\ 8}$, $A_{10,\ 10}$ | 6 | |

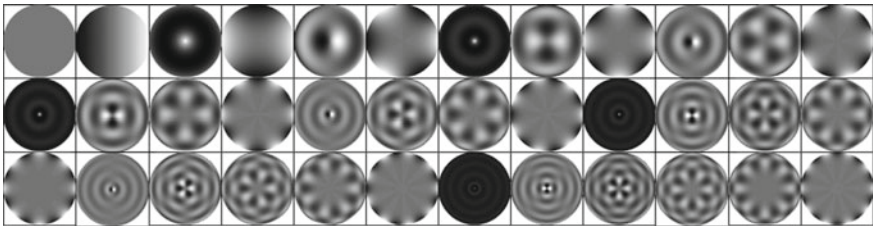


**Fig. 6.23** The first 36 Zernike moments from order 1 to 10

the second order (row), there are $[n/2]$ repetition of moments which capture different circular frequencies. The images of Zernike moment functions from order 1 to 10 are shown in Fig. 6.23 [12].

A simplified complex moment is used by MPEG-7, called *angular radial transformation* (ART).

$$ART_{nm} = \frac{1}{2\pi}\sum_{\rho}\sum_{\theta} f(\rho\cos\theta, \rho\sin\theta)V_{nm}(\rho,\theta), \quad \rho \le 1 \tag{6.40}$$

where $V_{nm}$ is the ART basis function

$$V_{nm} = R_n(\rho)\exp(jm\,\theta) \tag{6.41}$$

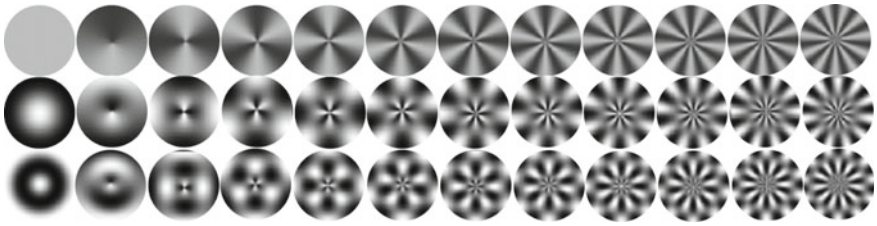


**Fig. 6.24** Real parts of the ART basis functions

and $R_n(\rho)$ is the radial basis function

$$R_n(\rho) = \begin{cases} 1 & if\ n = 0 \\ 2\cos(n\pi\rho) & if\ n \neq 0 \end{cases} \tag{6.42}$$

The real part of the first 36 ART basis functions are shown in Fig. 6.24.

Compared with geometric moments, complex moments are invariant to rotation and they are also more robust due to capturing the spatial information within a shape. They have minimum information redundancy due to orthogonal basis.

However, the computation of complex moments is more expensive than geometric moments. The image needs to be normalized to a unit disk. For an irregular shape, this may either cut out part of the shape if an interior circle is used or include irrelevant part if an exterior circle is used. Depending on how much the shape is cut out or irrelevant part is included, the unit disk normalization can affect the accuracy.

### 6.4.3 Generic Fourier Descriptor

Complex moment improves the geometric moment with three advantages: rotation invariance, spatial or frequency information, and orthogonality. These three features can be achieved more naturally and efficiently using Fourier transform. The generic Fourier descriptor or GFD is a method just based on this idea [13].

The idea is to first transform a shape into a rectangular polar image with sides $r$ and $\theta$ by a polar raster sampling around the shape centroid. Next, a 2D Fourier transform is applied on the transformed rectangular image. The normalized Fourier coefficients are then used as the shape descriptor.

Figure 6.25 demonstrates the polar raster transform [13]. For example, Fig. 6.25a is the original shape image in polar space, Fig. 6.25b is the polar raster sampled image plotted into Cartesian space.

Given a shape image $I = \{f(x, y); 0 \leq x < M, 0 \leq y < N\}$. To apply the polar Fourier transform or PFT, the shape image is converted from Cartesian space to polar space $I_p = \{f(r, \theta); 0 \leq r < R, 0 \leq \theta < 2\pi\}$, $R$ is the maximum radius of the shape. The origin of the polar space is set to be the centroid of the shape so that the shape is translation invariant. The centroid $(x_c, y_c)$ is given by (6.43)


**Fig. 6.25** Polar raster transform. **a** An original shape image in polar space; **b** polar raster sampled image of (**a**) plotted into Cartesian space

$$x_c = \frac{1}{M}\sum_{x=0}^{N-1} x, \quad y_c = \frac{1}{N}\sum_{y=0}^{M-1} y \tag{6.43}$$

and $(r, \theta)$ is given by

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}, \theta = \arctan\frac{y - y_c}{x - x_c} \tag{6.44}$$

The shape image is then polar raster sampled around the centroid and transformed into a rectangular polar image. The polar image, e.g., Figure 6.25b, is a normal rectangular image. Therefore, a 2D FT is applied on this polar rectangle (PFT). The PFT has a similar form to the normal discrete 2D FT in Cartesian space and is defined as

$$PF(\rho, \theta) = \sum_r \sum_i f(r, \theta_i) \exp\left[j2\pi\left(\frac{r}{R}\rho + \frac{2\pi i}{T}\theta\right)\right] \tag{6.45}$$

where $0 \leq r < R$ and $\theta_i = i(2\pi/T)$ $(0 \leq i < T)$; $0 \leq \rho < R$, $0 \leq \theta < T$. $R$ and $T$ are the radial frequency resolution and angular frequency resolution, respectively.

In addition to the three advantages of complex moment, the PFT has another desirable feature of capturing shape information in a smaller number of low-frequency coefficients. This is particularly suitable for shape representation. Figure 6.26 shows an example of PFT on two shape images with different orientations. Conventional 2D FT on the two images results in two different spectra, as they are rotated each other. It can be observed from the middle row of Fig. 6.26 that rotation of shapes in Cartesian space results in circular shift in polar space. However, the circular shift does not change the spectra distribution on polar space, e.g., the bottom row of Fig. 6.26.

Since $f(x, y)$ is a real function, the spectra is circularly symmetric, only the first quarter of the spectra features is needed to describe the shape.

The acquired coefficients of the PFT are translation invariant due to the use of centroid as polar space origin. Rotation invariance is achieved by ignoring the phase information in the coefficients and only retaining the magnitudes of the

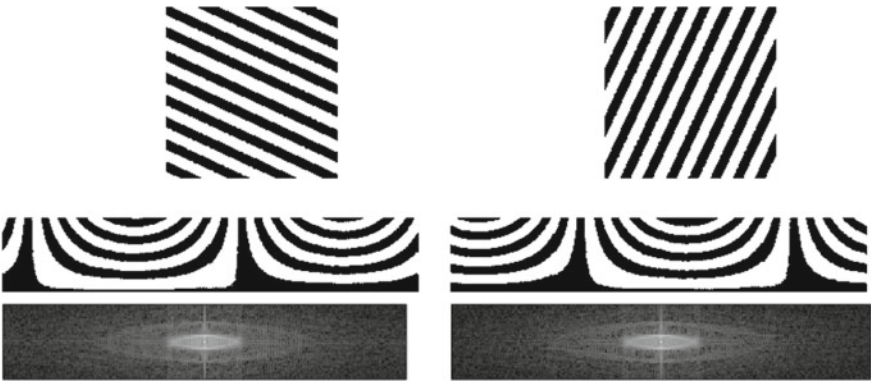


**Fig. 6.26** Rotation invariant GFD. Top row: two shape images with different orientations; middle row: the polar raster sampled images of the two corresponding shapes at the top row; bottom row: the Fourier spectra images of the two corresponding images at the middle row

coefficients. To achieve scale invariance, the first magnitude value is normalized by the area of the circle (*area*) in which the polar image resides or the mass of the shape (*mass*), and all the other magnitude values are normalized by the magnitude of the first coefficient. The translation, rotation, and scale normalized PFT coefficients are used as the shape descriptor. To summarize, the shape descriptor derived from the PFT is shown as follows:

$$\mathbf{GFD} = \left\{ \frac{|PF(0,0)|}{area}, \frac{|PF(0,1)|}{|PF(0,0)|}, \ldots, \frac{|PF(0,n)|}{|PF(0,0)|}, \ldots, \frac{|PF(m,0)|}{|PF(0,0)|}, \ldots, \frac{|PF(m,n)|}{|PF(0,0)|} \right\} \tag{6.46}$$

where $m$ is the maximum number of the radial frequencies selected and $n$ is the maximum number of angular frequencies selected. $m$ and $n$ can be adjusted to achieve hierarchical coarse to fine representation requirement. Normally, the first coefficient, or the *DC* component is used as the normalization factor and is discarded after normalization. However, this component is used as an additional feature in a shape descriptor because it reflects the average energy (scale) of the shape which is useful for shape description.

For efficient shape description, only a small number of the acquired GFD features are selected for shape representation. The selected GFD features form a feature vector which is used for indexing the shape.

Compared with Zernike moment descriptor (ZMD), GFD is simpler and more efficient to compute.

### 6.4.4 Shape Matrix

The most intuitive way to represent a shape is simply to binarize the shape within a bounding box, the result is a binary *shape matrix*. To acquire the shape matrix of a shape $S$, a square is centered at the *center of gravity* $G$ of $S$ (Fig. 6.27). The side length of the square is equal to $2L$, where $L$ is the maximum distance from $G$ to a point $M$ on the boundary of the shape, or $L = \overline{GM}$. All shape squares are normalized with the same length $L$ and are aligned with line $L$. The square is then divided into $N \times N$ blocks $b_{ij}$ and the shape matrix is defined as $SM = [c_{ij}]$ where $c_{ij}$ is given by

$$c_{ij} = \begin{cases} 1 & if\, A(S \cap b_{ij}) > A(b_{ij})/2 \\ 0 \end{cases} \tag{6.47}$$

where $A(\bullet)$ is the area function.

It is easy to show that the shape matrix acquired this way is invariant to translation, scale and rotation. The similarity of two shape matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ is given by

$$d(A,B) = 1 - \frac{1}{N^2}\sum_{i=0}^{N}\sum_{j=0}^{N}\left|a_{ij} - b_{ij}\right| \tag{6.48}$$

In [14], a binary shape number is created by concatenating the rows of a shape matrix into a vector.

The shape matrix acquired this way is sensitive to boundary noise because the size and orientation of the square bounding box can easily be influenced by the noise. In practice, there needs to be multiple guesses of longest radii, therefore, the best matching of multiple shape matrices is needed.

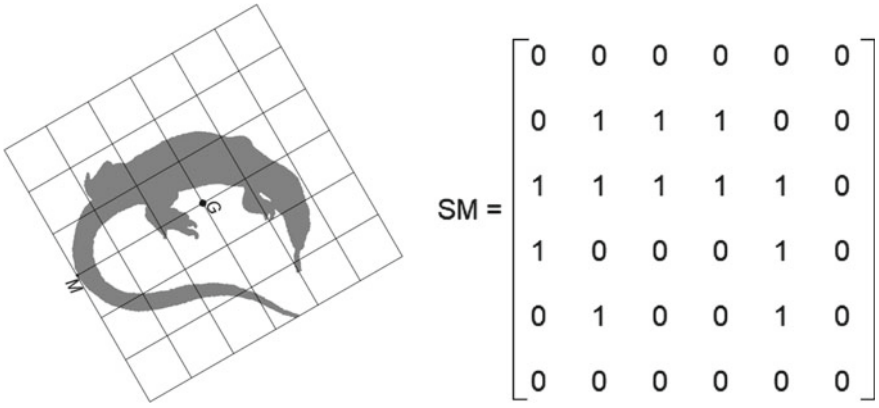
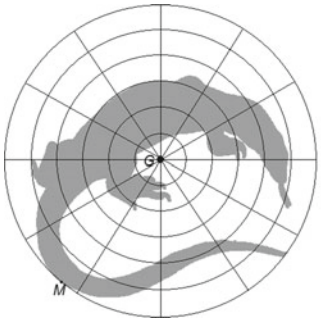



**Fig. 6.27** Computation of a shape matrix. A shape on the left and its shape matrix on the right



$$SM = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

**Fig. 6.28** Computation of a polar raster shape matrix. Left: polar raster sampling of a shape; Right: its polar shape matrix

A more robust shape matrix can be created by a using polar grid [15]. The idea is similar to the square shape matrix, however, instead of using a square grid, a polar grid is used. For example, the polar shape matrix of a shape is shown in Fig. 6.28.

The polar model of shape matrix is more robust than the square model because rotation of a shape only causes a horizontal shift of the shape matrix, the matching become simply a shift best matching. However, since the sampling density of the polar raster is not constant at all rings, a weighed shape matrix is necessary.

## 6.4.5 Shape Profiles

### 6.4.5.1 Shape Projections

Shape profiles can be extracted from the projections of the shape. The profiles are the projections of the shape onto $x$-axis and $y$-axis on the Cartesian coordinate system. By vertical and horizontal projections, two 1D functions are obtained:

$$\left.\begin{aligned} P_v(x) &= \sum_{y_{min}}^{y_{max}} f(x,y) \\ P_h(y) &= \sum_{x_{min}}^{x_{max}} f(x,y) \end{aligned}\right\} \tag{6.49}$$

The vertical profile $P_v(x)$ counts the number of pixels on each column of the shape and the horizontal profile $P_h(y)$ counts the number of pixels on each row of the shape (Fig. 6.29). The profiles are unique to each type of objects, they can be used as shape signatures to describe shapes.

Polar shape profiles can also be obtained in a similar way by counting the number of pixels at each angle and radius on polar coordinates.

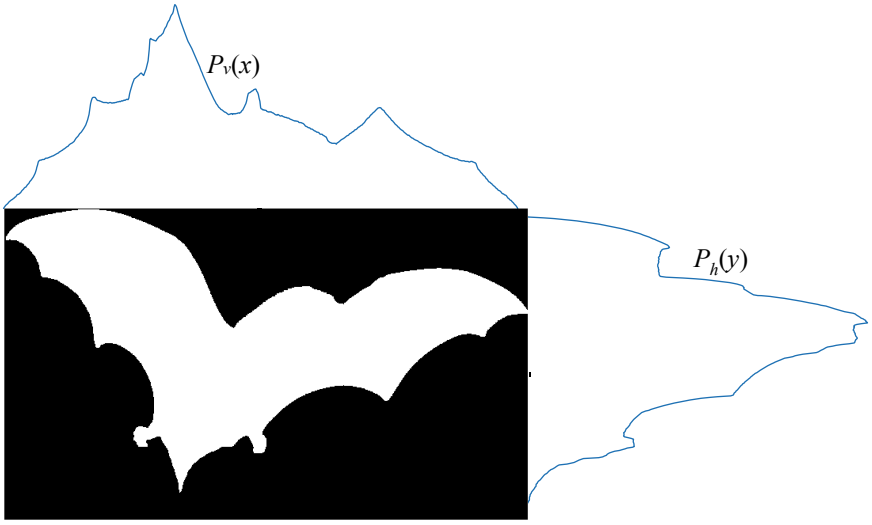



**Fig. 6.29** Computation of shape profiles. A binary shape image with its vertical profile $P_v(x)$ (top) and horizontal profile $P_h(y)$ (right-hand side)

#### 6.4.5.2 Radon Transform

Multiple shape profiles projected from different directions $\theta$ can be obtained using the *Radon transform*. When all these profiles are aligned on the $\theta$ axis in 3D, they create a Radon spectrum of the shape and the spectrum captures the content information of the shape region.

A Radon transform works by creating a shape profile at each angle. Formally, it is defined as

$$R(\rho,\theta) = \iint_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy \tag{6.51}$$

where $\rho = x\,cos\theta + y\,sin\theta$ is the line the shape is to be projected; $\theta$ and $\rho$ are the angle of the line and distance of the line to the origin, respectively. $\delta(x)$ is the Dirac delta-function and is given by (6.51)

$$\delta(x-a) = 0 \quad \text{for} \quad x \neq a \tag{6.51}$$

$$\int_{-\infty}^{\infty} f(x)\delta(x-a)dx = f(a) \tag{6.52}$$

The projection of a shape image at a particular angle $\theta$ is shown in Fig. 6.30.

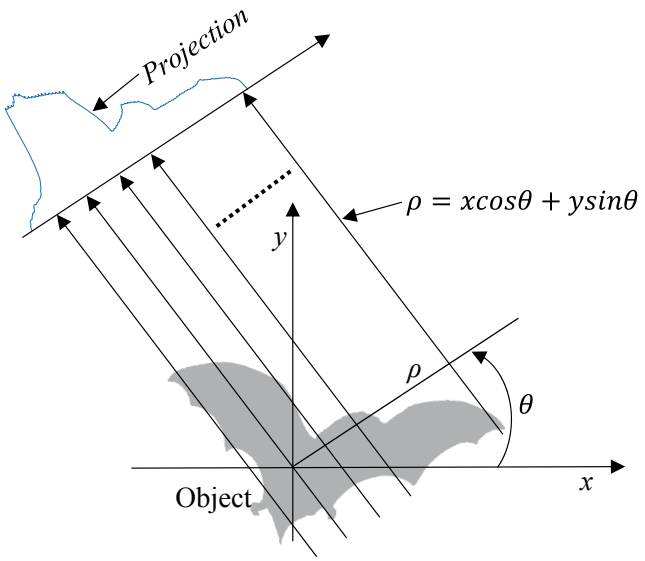



**Fig. 6.30** A shape profile from Radon transform

The top row of Fig. 6.31 shows the projections of a hammer shape from two different directions. It can be seen, the projection at 8° has a much higher amplitude than that at 90° due to the projection at 8° captures the profile of the long handle. The two profiles are shown at the bottom row of Fig. 6.31 after zooming in and realignment with the projection angles.

In order to capture the complete information of a shape, projections from all directions are created. If the projections from all directions are created and plotted into an angle-magnitude plane, it creates a spectrum of the transformed shape image. For example, the Radon transform spectrum of the hammer shape is shown in Fig. 6.32. The bright spots on the spectrum indicate high amplitudes on the projections. In this case, there is a single brightest spot at around 8°, it exactly captures the handle angle of the hammer. The next brightest area is around 100°, pointing to the hammerhead. Another Radon transform example is shown in Fig. 6.33, where the most bright spots are at 0° and 90°, pointing to the vertical legs and horizontal body of the dog.

A histogram or a GFD can be computed from the Radon spectrum image as a shape descriptor to match between two shapes.

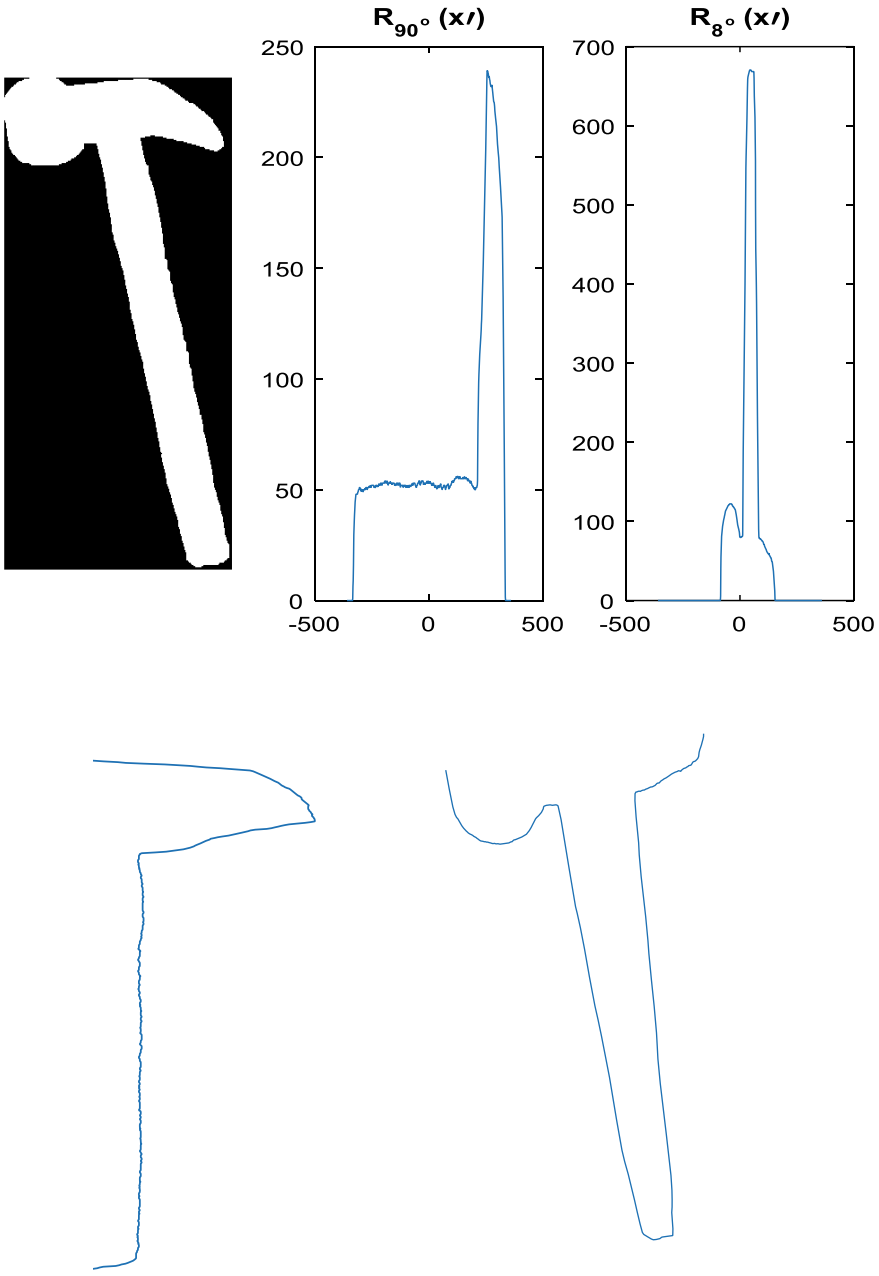



**Fig. 6.31** Shape profiles of a hammer image. Top row: A hammer shape and its Radon transforms at 90° and 8°; Bottom row: the 90° and 8° profiles of the hammer shape after zooming in and realignment

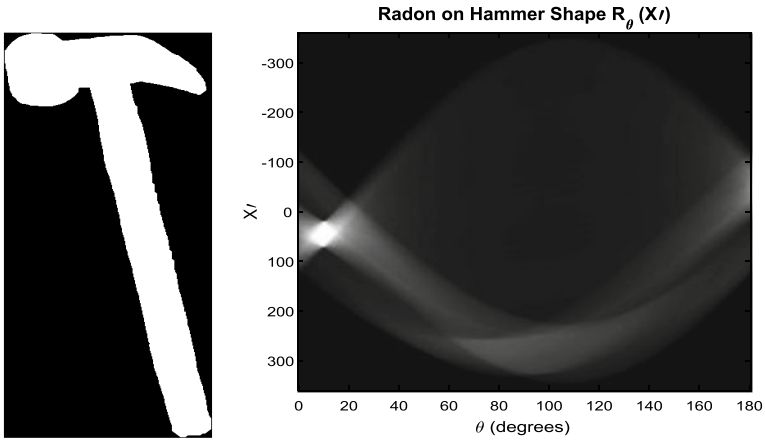



**Fig. 6.32** Radon transform of hammer shape. A hammer shape (left) and its Radon transform spectrum (right)

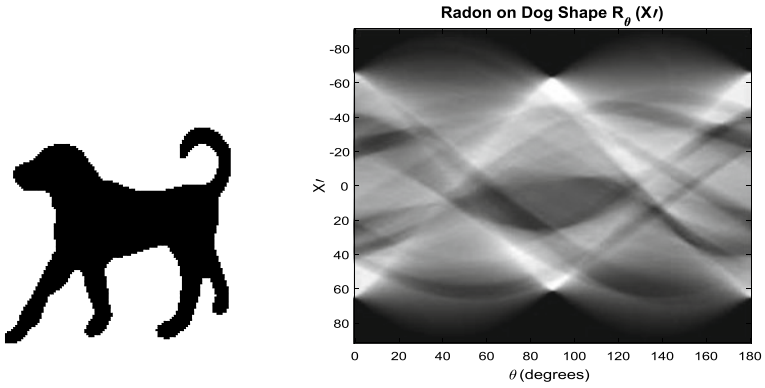



**Fig. 6.33** A dog shape and its Radon transform spectrum

### 6.4.6 Discussions

Global region based methods treat a shape region as a whole instead of just using the boundary, and make effective use of all the pixel information within the region. These methods measure *pixel distribution* within the shape region, which are less likely affected by noise and variations. This makes them more robust than contour-based methods. Methods like complex moments and GFD are more powerful region shape descriptors than conventional moments because they not only capture the pixel distribution within a shape but also capture the spatial details or spatial relationship between pixels. This spatial feature gives them a significant advantage

over other region-based methods, such as geometric moments, shape matrix, shape profiles, etc.

However, similar to the global boundary-based methods, global region-based methods cannot deal with overlapped shapes or shapes with missing parts. To address this issue, structural methods are used. Similar to the contour structural methods, *region-based structural methods* decompose a shape region into individual parts which are then used for shape representation and description. In the following, we discuss two of the region-based structural methods.

### 6.4.7 Convex Hull

A region $R$ is convex if and only if any two points $\mathbf{x}_1$, $\mathbf{x}_2 \in R$, the whole line segment $\mathbf{x}_1\mathbf{x}_2$ is inside the region. The *convex hull* of a region is the smallest convex region $H$ which satisfies the condition $R \subset H$. The difference $H - R$ is called the *convex deficiency* $D$ of the region $R$. Methods of computing a convex hull from a shape include morphological methods [16, 17] and polygon approximation. Shape boundaries tend to be irregular because of digitization, noise, and variations in segmentation; these irregularities and noise usually result in a convex deficiency that has small, insignificant components scattered randomly throughout the boundary. Common practice is to first smooth a boundary prior to convex hull computation. The polygon approximation is particularly attractive because it reduces the computation of extracting convex hull from $O(n^2)$ to $O(n)$.

The extracting of convex hull features can be a single process which finds the most significant convex deficiencies along the boundary. The shape can then be represented by a string of concavities. A fuller representation of the shape may be obtained by a recursive process which results in a concavity tree. To do this, the convex hull of an object is first obtained and its convex deficiencies are detected, they are level 1 convex hull and convex deficiencies. Next, the convex hulls and deficiencies of the level 1 convex deficiencies are found. Then, the convex hulls and deficiencies of the level 2 convex deficiencies are found. So on so forth until all the derived convex deficiencies are convex.

Figure 6.34a illustrates the computation of convex hull and concavities [7]. The shape is then represented as a concavity tree in Fig. 6.34b. Each concavity can be described by its area, bridge length which is the line connecting the cut of the concavity, maximum curvature, distance from maximum curvature point to the bridge. The matching between shapes becomes a string matching or a graph matching. A shape boundary needs to be smoothed to remove small irregularities before convex hull and concavity extraction, otherwise, the concavity tree would be very complex and sensitive. For example, if the boundary of the apple shape in Fig. 6.34a had been smoothed, the concavity tree would only have four branches $S_1$–$S_4$, which accurately represent the shape boundary.

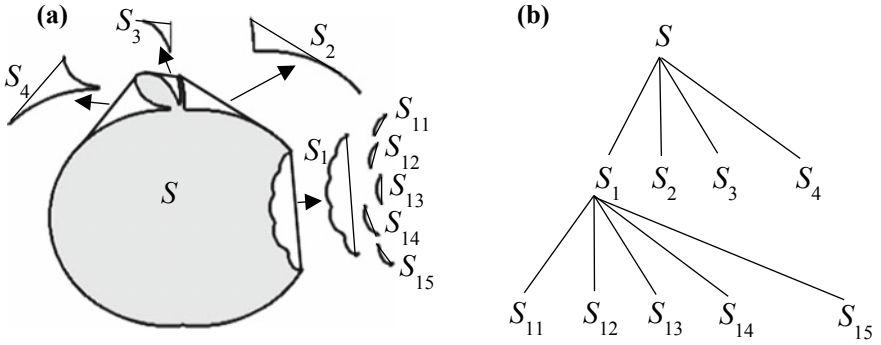



**Fig. 6.34** Convex hull and concavity tree of an apple shape. **a** The convex hull of an apple shape and its concavities; **b** Concavity tree representation of the convex hull

### 6.4.8 Medial Axis

Like the convex hull, region *skeleton* can also be employed for shape representation and description. A skeleton may be defined as a connected set of medial lines along the limbs of a figure [7]. For example, in the case of thick hand-drawn characters, the skeleton may be supposed to be the path traveled by the pen. The basic idea of the skeleton is that eliminating redundant information while retaining only the topological information concerning the structure of the object that can help with recognition.

Shape skeleton can be found by using Blum's *medial axis transform* (MAT) [18]. In MAT, the medial axis is the locus of centers of maximal disks or bi-tangent circles that fit entirely within the shape as illustrated in Fig. 6.35. The bold line in the figure is the skeleton of the hand shape. The skeleton can then be

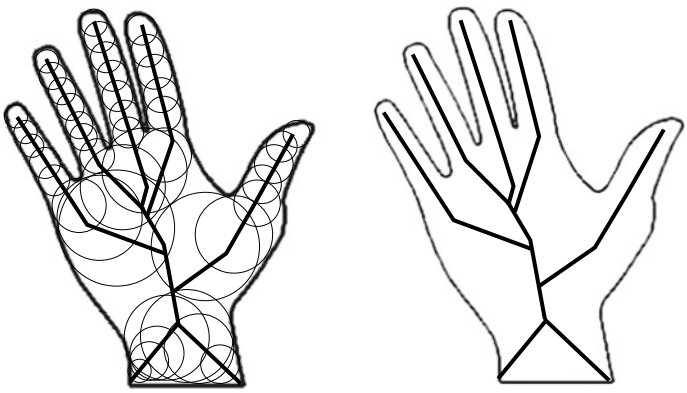


**Fig. 6.35** Computation of media axis. Left: construction of medial axis of a rectangle shape using locus of circles; Right: the medial axis or skeleton of the hand shape

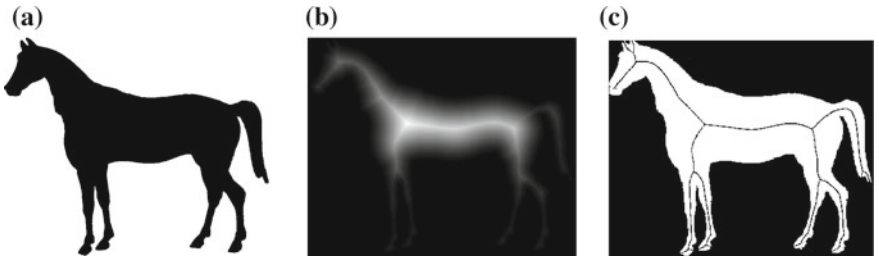



**Fig. 6.36** Computation of a shape skeleton. **a** A horse shape image; **b** distance map of (**a**); **c** skeleton of the shape

decomposed into segments and represented as a graph according to certain criteria. The matching between shapes becomes a graph matching problem.

The computation of medial axis in this way is a rather challenging problem. In addition, medial axis tends to be very sensitive to boundary noise and variations. Therefore, it is suggested that the contour of a shape be smoothed before the media axis computation.

The medial axis can be computed from scale space. The medial axis acquired in this way is called the core of the shape [19].

An alternative way of finding the medial axis is to use a distance transform (calculate the distance from each shape point to the background) to convert the binary shape into a gray-level distance map (Fig. 6.36b). A ridge detection is then applied on the distance map followed by a linking process. The ridge points are local extrema which can be found by scanning the distance map both horizontally and vertically. An extrema is found at where the gradient changes from positive (uphill) to negative (downhill). The skeleton of the shape is finally shown up after the linking process (Fig. 6.36c).

The limbs and spine of the skeleton are then detected, features of the limbs and spine such as *length*, *angle*, *curvature* are computed for indexing.

The region structural methods are useful in applications which require partial matching due to object overlapping and missing parts. However, they suffer from similar drawbacks to the contour base structural approaches. Apart from complex computation and implementation, the graph matching is also a complex issue. These issues can affect the performance of region structural methods significantly.

## 6.5 Summary

This chapter describes three types of shape descriptors: perceptual, contour-based, and region-based. Perceptual shape descriptors are very intuitive and easy to understand, however, they are not powerful enough to be used alone. Typically, they are used as filters to eliminate large number of irrelevant shapes before other shape descriptors are used to refine the retrieval list.