Object description

7

CHAPTER OUTLINE HEAD

7.1 Overview

Objects are represented as a collection of pixels in an image. Thus, for purposes of recognition we need to describe the properties of groups of pixels. The description is often just a set of numbers—the object's *descriptors*. From these, we can compare and recognize objects by simply matching the descriptors of objects in an image against the descriptors of known objects. However, in order to be useful for recognition, descriptors should have four important properties. First, they

Table 7.1 Overview of Chapter 7

Main Topic	Subtopics	Main Points
Boundary descriptions	How to determine the **boundary** and the **region** it encloses. How to form a description of the boundary and necessary **properties** in that description. How we describe a curve/boundary by **Fourier** approaches	Basic approach: *chain codes*. *Fourier descriptors*: discrete approximations; cumulative angular function and *elliptic Fourier descriptors*
Region descriptors	How we describe the **area** of a shape. Basic shape measures: heuristics and properties. **Describing** area by statistical moments: need for **invariance** and more sophisticated descriptions. What do the moments describe, and **reconstruction** from the moments	Basic *shape measures*: area; perimeter; *compactness*; and dispersion. *Moments*: basic; *centralized*; *invariant*; *Zernike*. Properties and reconstruction

should define a *complete set*, i.e., two objects must have the same descriptors if and only if they have the same shape. Secondly, they should be *congruent*. As such, we should be able to recognize **similar** objects when they have **similar** descriptors. Thirdly, it is convenient that they have **invariant** properties. For example, **rotation**-invariant descriptors will be useful for recognizing objects whatever their **orientation**. Other important invariance properties naturally include scale and position and also invariance to affine and perspective changes. These last two properties are very important when recognizing objects observed from different viewpoints. In addition to these three properties, the descriptors should be a **compact** set, namely, a descriptor should represent the essence of an object in an efficient way, i.e., it should only contain information about what makes an object unique or different from the other objects. The quantity of information used to describe this characterization should be less than the information necessary to have a complete description of the object itself. Unfortunately, there is no set of complete and compact descriptors to characterize general objects. Thus, the best recognition performance is obtained by carefully selected properties. As such, the process of recognition is strongly related to each particular application with a particular type of object.

In this chapter, we present the characterization of objects by two forms of descriptors. These descriptors are summarized in Table 7.1. **Region** and **shape** descriptors characterize an arrangement of pixels within the **area** and the arrangement of pixels in the **perimeter** or **boundary**, respectively. This region versus

perimeter kind of representation is common in image analysis. For example, edges can be located by **region growing** (to label area) or by **differentiation** (to label perimeter), as covered in Chapter 4. There are actually many techniques that can be used to obtain descriptors of an object's boundary. Here, we shall just concentrate on three forms of descriptors: *chain codes* and two forms based on *Fourier characterization*. For region descriptors, we shall distinguish between basic descriptors and statistical descriptors defined by moments.

7.2 Boundary descriptions

7.2.1 Boundary and region

A region usually describes *contents* (or interior points) that are surrounded by a *boundary* (or perimeter) which is often called the region's *contour*. The form of the contour is generally referred to as its *shape*. A point can be defined to be on the boundary (contour) if it is part of the region and there is at least one pixel in its neighborhood that is not part of the region. The boundary itself is usually found by contour following: we first find one point on the contour and then progress round the contour either in a clockwise direction or in an anticlockwise, finding the nearest (or next) contour point.

In order to define the interior points in a region and the points in the boundary, we need to consider neighboring relationships between pixels. These relationships are described by means of *connectivity* rules. There are two common ways of defining connectivity: *4-way* (or 4-neighborhood) where only immediate **neighbors** are analyzed for connectivity; or *8-way* (or 8-neighborhood) where all the eight pixels surrounding a chosen pixel are analyzed for connectivity. These two types of connectivity are shown in Figure 7.1. In this figure, the pixel is shown in

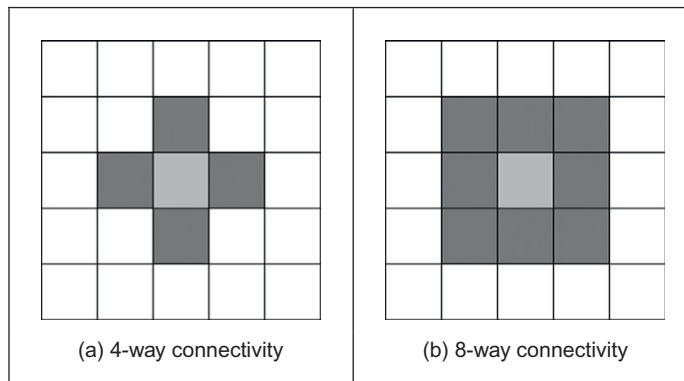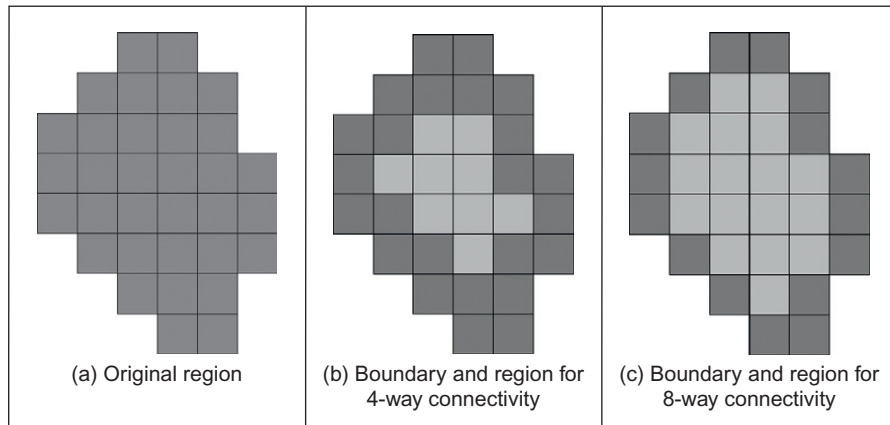(a) 4-way connectivity   (b) 8-way connectivity

FIGURE 7.1

Main types of connectivity analysis.

FIGURE 7.2

Boundaries and regions.

light gray and its neighbors in dark gray. In 4-way connectivity (Figure 7.1(a)) a pixel has four neighbors in the directions: North, East, South, and West, its immediate neighbors. The four extra neighbors in 8-way connectivity (Figure 7.1(b)) are those in the directions: North East, South East, South West, and North West, the points at the **corners**.

A boundary and a region can be defined using both types of connectivity and they are always **complementary**, i.e., if the boundary pixels are connected in 4-way, the region pixels will be connected in 8-way and vice versa. This relationship can be seen in the example shown in Figure 7.2, where the boundary is shown in dark gray and the region in light gray. We can observe that for a diagonal boundary, the 4-way connectivity gives a staircase boundary, whereas the 8-way connectivity gives a diagonal line formed from the points at the corners of the neighborhood. Note that all the pixels that form the region in Figure 7.2(b) have 4-way connectivity, while the pixels in Figure 7.2(c) have 8-way connectivity. This is complementary to the pixels in the border.

7.2.2 Chain codes

In order to obtain a representation of a contour, we can simply store the coordinates of a sequence of pixels in the image. Alternatively, we can just store the relative position between consecutive pixels. This is the basic idea behind *chain codes.* Chain codes are actually one of the oldest techniques in computer vision originally introduced in the 1960s (Freeman, 1961) (an excellent review came later; Freeman, 1974). Essentially, the set of pixels in the border of a shape is translated into a set of connections between them. Given a complete border, one
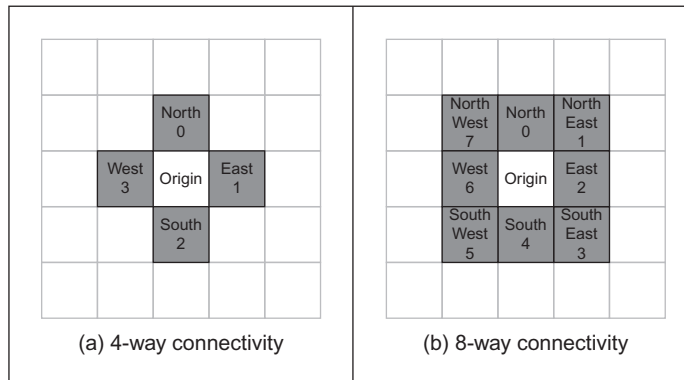
FIGURE 7.3

Connectivity in chain codes.

that is a set of connected points, then starting from one pixel we need to be able to determine the direction in which the next pixel is to be found, namely, the next pixel is one of the adjacent points in one of the major compass directions. Thus, the chain code is formed by concatenating the number that designates the direction of the next pixel, i.e., given a pixel, the successive direction from one pixel to the next pixel becomes an element in the final code. This is repeated for each point until the start point is reached when the (closed) shape is completely analyzed.

Directions in 4- and 8-way connectivity can be assigned as shown in Figure 7.3. The chain codes for the example region in Figure 7.2(a) are shown in Figure 7.4. Figure 7.4(a) shows the chain code for the 4-way connectivity. In this case, we have that the direction from the start point to the next is South (i.e., code 2), so the first element of the chain code describing the shape is 2. The direction from point P1 to the next, P2, is East (code 1), so the next element of the code is 1. The next point after P2 is P3 that is South giving a code 2. This coding is repeated until P23 that is connected eastward to the starting point, so the last element (the 12th element) of the code is 1. The code for 8-way connectivity shown in Figure 7.4(b) is obtained in an analogous way, but the directions are assigned according to the definition in Figure 7.3(b). Note that the length of the code is shorter for this connectivity, given that the number of boundary points is smaller for 8-way connectivity than it is for 4-way.

Clearly this code will be different when the start point changes. Accordingly, we need *start point invariance*. This can be achieved by considering the elements of the code to constitute the digits in an integer. Then, we can shift the digits *cyclically* (replacing the least significant digit with the most significant one, and shifting all other digits left one place). The smallest integer is returned as the *start*
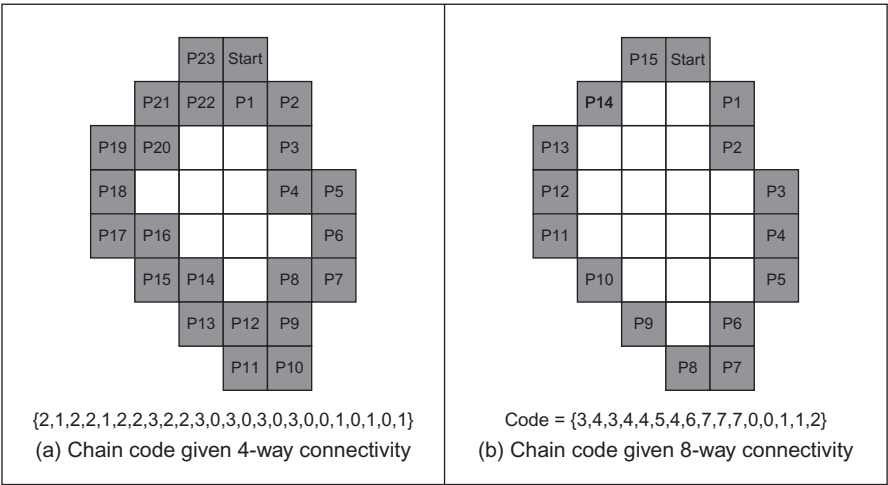
FIGURE 7.4

Chain codes by different connectivity.

FIGURE 7.5

Start point invariance in chain codes.

point invariant chain code description. This is shown in Figure 7.5 where the initial chain code is that from the shape in Figure 7.4. Here, the result of the first shift is given in Figure 7.5(b), which is equivalent to the code that would have been derived by using point P1 as the starting point. The result of two shifts, in Figure 7.5(c), is the chain code equivalent to starting at point P2, but this is not a code corresponding to the minimum integer. The minimum integer code, as in Figure 7.5(d), is the minimum of all the possible shifts and is actually the chain code which would have been derived by starting at point P11. That fact could not be used in application since we would need to find P11; naturally, it is much easier to shift to achieve a minimum integer.

In addition to starting point invariance, we can also obtain a code that does not change with **rotation**. This can be achieved by expressing the code as a

difference of chain code: relative descriptions remove rotation dependence. Change of **scale** can complicate matters greatly since we can end up with a set of points which is of different size to the original set. As such, the boundary needs to be **resampled** before coding. This is a tricky issue. Furthermore, **noise** can have drastic effects. If salt and pepper **noise** were to remove, or to add, some points the code would change. Clearly, such problems can lead to great difficulty with chain codes. However, their main virtue is their **simplicity** and as such they remain a popular technique for shape description. Further developments of chain codes have found application with **corner detectors** (Liu and Srinath, 1990; Seeger and Seeger, 1994). However, the need to be able to handle noise, the requirement of connectedness, and the local nature of description naturally motivates alternative approaches. Noise can be reduced by **filtering**, which naturally leads back to the **Fourier transform**, with the added advantage of a **global** description.

7.2.3 Fourier descriptors

Fourier descriptors, often attributed to early work by Cosgriff (1960), allow us to bring the power of Fourier theory to shape description. The main idea is to characterize a contour by a set of numbers that represent the frequency content of a whole shape. Based on frequency analysis, we can select a **small** set of numbers (the Fourier coefficients) that describe a shape rather than any noise (i.e., the noise affecting the spatial position of the boundary pixels). The general recipe to obtain a Fourier description of the curve involves two main steps. First, we have to define a representation of a curve. Secondly, we expand it using Fourier theory. We can obtain alternative flavors by combining different curve representations and different Fourier expansions. Here, we shall consider Fourier descriptors of angular and complex contour representations. However, Fourier expansions can be developed for other curve representations (Persoon and Fu, 1977; Van Otterloo, 1991).

In addition to the curve's definition, a factor that influences the development and properties of the description is the choice of Fourier expansion. If we consider that the trace of a curve defines a periodic function, we can opt to use a Fourier series expansion. However, we could also consider that the description is not periodic. Thus, we could develop a representation based on the Fourier transform. In this case, we could use alternative Fourier integral definitions. Here, we will develop the presentation based on expansion in Fourier series. This is the common way used to describe shapes in pattern recognition.

It is important to note that although a curve in an image is composed of discrete pixels, Fourier descriptors are developed for continuous curves. This is convenient since it leads to a discrete set of Fourier descriptors. Additionally, we should remember that the pixels in the image are actually the sampled points of a continuous curve in the scene. However, the formulation leads to the definition of the integral of a continuous curve. In practice, we do not have a continuous curve

but a sampled version. Thus, the expansion is actually approximated by means of numerical integration.

7.2.3.1 Basis of Fourier descriptors

In the most basic form, the coordinates of boundary pixels are x and y point coordinates. A Fourier description of these essentially gives the set of spatial frequencies that fit the boundary points. The **first** element of the Fourier components (the d.c. component) is simply the average value of the x and y coordinates, giving the coordinates of the center point of the boundary, expressed in complex form. The **second** component essentially gives the radius of the circle that best fits the points. Accordingly, a circle can be described by its zero- and first-order components (the d.c. component and first harmonic). The **higher**-order components increasingly describe detail, as they are associated with higher frequencies.

This is shown in Figure 7.6. Here, the Fourier description of the ellipse in Figure 7.6(a) is the frequency components in Figure 7.6(b), depicted in logarithmic form for purposes of display. The Fourier description has been obtained by using the ellipse boundary points' coordinates. Here, we can see that the low-order components dominate the description, as to be expected for such a smooth shape. In this way, we can derive a set of numbers that can be used to **recognize** the boundary of a shape: a similar ellipse should give a similar set of numbers, whereas a completely different shape will result in a completely different set of numbers.

We do, however, need to check the result. One way is to take the descriptors of a circle since the first harmonic should be the circle's radius. A better way though is to **reconstruct** the shape from its descriptors; if the reconstruction
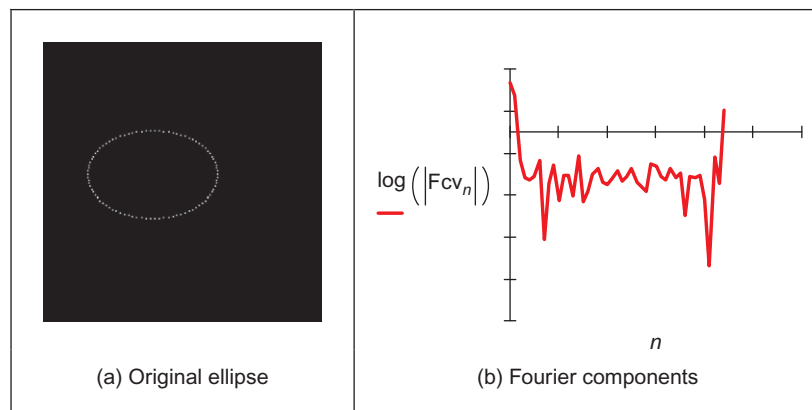

(a) Original ellipse   (b) Fourier components

FIGURE 7.6

An ellipse and its Fourier description.

matches the original shape, then the description would appear correct. Naturally, we can reconstruct a shape from this Fourier description since the descriptors are **regenerative**. The zero-order component gives the position (or origin) of a shape. The ellipse can be reconstructed by adding in all spatial components to extend and compact the shape along the *x*- and *y*-axes, respectively. By this inversion, we return to the original ellipse. When we include the zero and first descriptor, then we reconstruct a circle, as expected, shown in Figure 7.7(b). When we include all Fourier descriptors the reconstruction (Figure 7.7(c)) is very close to the original (Figure 7.7(a)) with slight difference due to discretization effects.

But this is only an outline of the basis to Fourier descriptors since we have yet to consider descriptors that give the same description whatever be an object's position, scale, and rotation. Here, we have just considered an object's description that is achieved in a manner that allows for reconstruction. In order to develop practically useful descriptors, we shall need to consider more basic properties. As such, we first turn to the use of Fourier theory for shape description.

7.2.3.2 Fourier expansion

In order to define a Fourier expansion, we can start by considering that a continuous curve $c(t)$ can be expressed as a summation of the form

$$c(t) = \sum_k c_k f_k(t) \tag{7.1}$$

where c_k defines the coefficients of the expansion, and the collection of functions $f_k(t)$ define the basis functions. The expansion problem centers on finding the coefficients given a set of basis functions. This equation is very general and different basis functions can also be used. For example, $f_k(t)$ can be chosen such that the expansion defines a polynomial. Other bases define splines, Lagrange, and
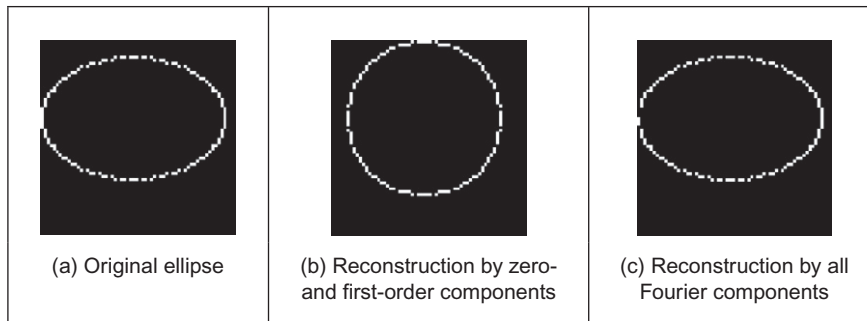

| (a) Original ellipse | (b) Reconstruction by zero- and first-order components | (c) Reconstruction by all Fourier components |

FIGURE 7.7

Reconstructing an ellipse from a Fourier description.

Newton interpolant functions. A Fourier expansion represents periodic functions by a basis defined as a set of infinite complex exponentials, i.e.,

$$c(t) = \sum_{k=-\infty}^{\infty} c_k\, e^{jk\omega t} \tag{7.2}$$

Here, ω defines the fundamental frequency and it is equal to $2\pi/T$, where T is the period of the function. The main feature of the Fourier expansion is that it defines an orthogonal basis. This simply means that

$$\int_0^T f_k(t)f_j(t)\mathrm{d}t = 0 \tag{7.3}$$

for $k \neq j$. This property is important for two main reasons. First, it ensures that the expansion does not contain redundant information (each coefficient is **unique** and contains no information about the other components). Secondly, it simplifies the computation of the coefficients, i.e., in order to solve for c_k in Eq. (7.1), we can simply multiply both sides by $f_k(t)$ and perform integration. Thus, the coefficients are given by

$$c_k = \int_0^T c(t)f_k(t) \Big/ \int_0^T f_k^2(t) \tag{7.4}$$

By considering the definition in Eq. (7.2), we have that

$$c_k = \frac{1}{T}\int_0^T c(t)e^{-jkwt} \tag{7.5}$$

In addition to the exponential form given in Eq. (7.2), the Fourier expansion can also be expressed in trigonometric form. This form shows that the Fourier expansion corresponds to the summation of trigonometric functions that increase in frequency. It can be obtained by considering that

$$c(t) = c_0 + \sum_{k=1}^{\infty} (c_k e^{jk\omega t} + c_{-k}e^{-jk\omega t}) \tag{7.6}$$

In this equation, the values of $e^{jk\omega t}$ and $e^{-jk\omega t}$ define a pair of complex conjugate vectors. Thus, c_k and c_{-k} describe a complex number and its conjugate. Let us define these numbers as

$$c_k = c_{k,1} - jc_{k,2} \quad \text{and} \quad c_{-k} = c_{k,1} + jc_{k,2} \tag{7.7}$$

By substitution of this definition in Eq. (7.6), we obtain

$$c(t) = c_0 + 2 \sum_{k=1}^{\infty} \left(c_{k,1} \left(\frac{e^{jk\omega t} + e^{-jk\omega t}}{2} \right) + jc_{k,2} \left(\frac{-e^{jk\omega t} + e^{-jk\omega t}}{2} \right) \right) \qquad (7.8)$$

That is,

$$c(t) = c_0 + 2 \sum_{k=1}^{\infty} (c_{k,1} \cos(k\omega t) + c_{k,2} \sin(k\omega t)) \qquad (7.9)$$

If we define

$$a_k = 2c_{k,1} \quad \text{and} \quad b_k = 2c_{k,2} \qquad (7.10)$$

we obtain the standard trigonometric form given by

$$c(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(k\omega t) + b_k \sin(k\omega t)) \qquad (7.11)$$

The coefficients of this expansion, a_k and b_k, are known as the *Fourier descriptors*. These control the amount of each frequency that contributes to make up the curve. Accordingly, these descriptors can be said to **describe** the curve since they do not have the same values for different curves. Note that according to Eqs (7.7) and (7.10), the coefficients of the trigonometric and exponential form are related by

$$c_k = \frac{a_k - jb_k}{2} \quad \text{and} \quad c_{-k} = \frac{a_k + jb_k}{2} \qquad (7.12)$$

The coefficients in Eq. (7.11) can be obtained by considering the orthogonal property in Eq. (7.3). Thus, one way to compute values for the descriptors is

$$a_k = \frac{2}{T} \int_0^T c(t)\cos(k\omega t)dt \quad \text{and} \quad b_k = \frac{2}{T} \int_0^T c(t)\sin(k\omega t)dt \qquad (7.13)$$

In order to obtain the Fourier descriptors, a curve can be represented by the complex exponential form of Eq. (7.2) or by the sin/cos relationship of Eq. (7.11). The descriptors obtained by using either of the two definitions are equivalent, and they can be related by the definitions of Eq. (7.12). Generally, Eq. (7.13) is used to compute the coefficients since it has a more intuitive form. However, some works have considered the complex form (e.g., Granlund, 1972). The complex form provides an elegant development of rotation analysis.

7.2.3.3 Shift invariance

Chain codes required special attention to give start point invariance. Let us see if that is required here. The main question is whether the descriptors will change when the curve is shifted. In addition to Eqs (7.2) and (7.11), a Fourier expansion can be written in another sinusoidal form. If we consider that

$$|c_k| = \sqrt{a_k^2 + b_k^2} \quad \text{and} \quad \varphi_k = a\tan^{-1}(b_k/a_k) \tag{7.14}$$

then the Fourier expansion can be written as

$$c(t) = \frac{a_0}{2} + \sum_{k=0}^{\infty} |c_k|\cos(k\omega t + \varphi_k) \tag{7.15}$$

Here $|c_k|$ is the **amplitude** and φ_k is the **phase** of the Fourier coefficient. An important property of the Fourier expansion is that $|c_k|$ does not change when the function $c(t)$ is shifted (i.e., translated), as in Section 2.6.1. This can be observed by considering the definition of Eq. (7.13) for a shifted curve $c(t + \alpha)$. Here, α represents the shift value. Thus,

$$a_k' = \frac{2}{T}\int_0^T c(t' + \alpha)\cos(k\omega t')\mathrm{d}t \quad \text{and} \quad b_k' = \frac{2}{T}\int_0^T c(t' + \alpha)\sin(k\omega t')\mathrm{d}t \tag{7.16}$$

By defining a change of variable by $t = t' + \alpha$, we have

$$a_k' = \frac{2}{T}\int_0^T c(t)\cos(k\omega t - k\omega\alpha)\mathrm{d}t \quad \text{and} \quad b_k' = \frac{2}{T}\int_0^T c(t)\sin(k\omega t - k\omega\alpha)\mathrm{d}t \tag{7.17}$$

After some algebraic manipulation, we obtain

$$a_k' = a_k\cos(k\omega\alpha) + b_k\sin(k\omega\alpha) \quad \text{and} \quad b_k' = b_k\cos(k\omega\alpha) - a_k\sin(k\omega\alpha) \tag{7.18}$$

The amplitude $|c_k'|$ is given by

$$|c_k'| = \sqrt{(a_k\cos(k\omega\alpha) + b_k\sin(k\omega\alpha))^2 + (b_k\cos(k\omega\alpha) - a_k\sin(k\omega\alpha))^2} \tag{7.19}$$

That is,

$$|c_k'| = \sqrt{a_k^2 + b_k^2} \tag{7.20}$$

Thus, the amplitude is independent of the shift α. Although shift invariance could be incorrectly related to translation invariance, actually, as we shall see, this property is related to rotation invariance in shape description.

7.2.3.4 Discrete computation

Before defining Fourier descriptors, we must consider the numerical procedure necessary to obtain the Fourier coefficients of a curve. The problem is that Eqs (7.11) and (7.13) are defined for a **continuous** curve. However, given the discrete nature of the image, the curve $c(t)$ will be described by a collection of **points**. This discretization has two important effects. First, it limits the number of frequencies in the expansion. Secondly, it forces numerical approximation to the integral defining the coefficients.

Figure 7.8 shows an example of a discrete approximation of a curve. Figure 7.8(a) shows a continuous curve in a period, or interval, T. Figure 7.8(b) shows the approximation of the curve by a set of discrete points. If we try to obtain the curve from the sampled points, we will find that the sampling process reduces the amount of detail. According to the Nyquist theorem, the maximum frequency f_c in a function is related to the sample period τ by

$$\tau = \frac{1}{2f_c} \tag{7.21}$$

Thus, if we have m sampling points, then the sampling period is equal to $\tau = T/m$. Accordingly, the maximum frequency in the approximation is given by

$$f_c = \frac{m}{2T} \tag{7.22}$$

Each term in Eq. (7.11) defines a trigonometric function at frequency $f_k = k/T$. By comparing this frequency with the relationship in Eq. (7.15), we have that the maximum frequency is obtained when
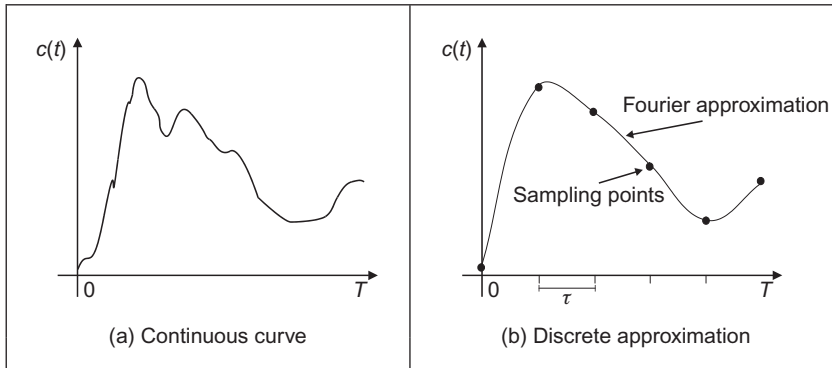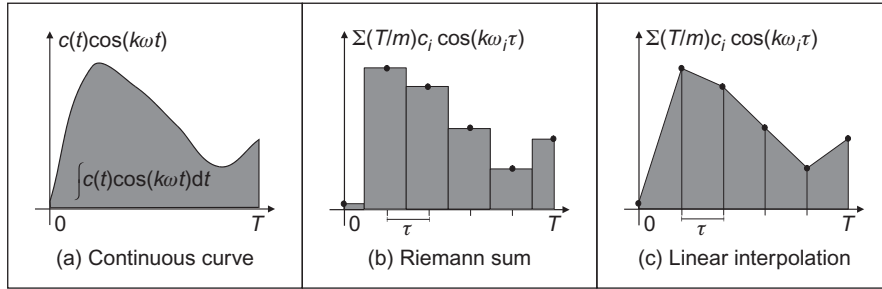
$$k = \frac{m}{2} \tag{7.23}$$

FIGURE 7.8

Example of a discrete approximation.

FIGURE 7.9

Integral approximation.

Thus, in order to define a curve that passes through the m sampled points, we need to consider only $m/2$ coefficients. The other coefficients define frequencies higher than the maximum frequency. Accordingly, the Fourier expansion can be redefined as

$$c(t) = \frac{a_0}{2} + \sum_{k=1}^{m/2}(a_k \cos(k\omega t) + b_k \sin(k\omega t)) \qquad (7.24)$$

In practice, Fourier descriptors are computed for fewer coefficients than the limit of $m/2$. This is because the low-frequency components provide most of the features of a shape. High frequencies are easily affected by noise and only represent detail that is of little value to recognition. We can interpret Eq. (7.22) the other way around: if we know the maximum frequency in the curve, then we can determine the appropriate number of samples. However, the fact that we consider $c(t)$ to define a continuous curve implies that in order to obtain the coefficients in Eq. (7.13), we need to evaluate an integral of a continuous curve. The approximation of the integral is improved by increasing the number of sampling points. Thus, as a practical rule, in order to improve accuracy, we must try to have a large number of samples even if it is theoretically limited by the Nyquist theorem.

Our curve is only a set of discrete points. We want to maintain a continuous curve analysis in order to obtain a set of discrete coefficients. Thus, the only alternative is to approximate the coefficients by approximating the value of the integrals in Eq. (7.13). We can approximate the value of the integral in several ways. The most straightforward approach is to use a Riemann sum. Figure 7.9 shows this approach. In Figure 7.9(b), the integral is approximated as the summation of the rectangular areas. The middle point of each rectangle corresponds to each sampling point. Sampling points are defined at the points whose parameter

is $t = i\tau$, where i is an integer between 1 and m. We consider that c_i defines the value of the function at the sampling point i, i.e.,

$$c_i = c(i\tau) \tag{7.25}$$

Thus, the height of the rectangle for each pair of coefficients is given by $c_i \cos(k\omega i\tau)$ and $c_i \sin(k\omega i\tau)$. Each interval has a length $\tau = T/m$. Thus,

$$\int_0^T c(t)\cos(k\omega t)\mathrm{d}t \approx \sum_{i=1}^m \frac{T}{m}c_i \cos(k\omega i\tau) \quad \text{and} \quad \int_0^T c(t)\sin(k\omega t)\mathrm{d}t \approx \sum_{i=1}^m \frac{T}{m}c_i \sin(k\omega i\tau) \tag{7.26}$$

Accordingly, the Fourier coefficients are given by

$$a_k = \frac{2}{m}\sum_{i=1}^m c_i \cos(k\omega i\tau) \quad \text{and} \quad b_k = \frac{2}{m}\sum_{i=1}^m c_i \sin(k\omega i\tau) \tag{7.27}$$

Here, the error due to the discrete computation will be reduced with increase in the number of points used to approximate the curve. These equations actually correspond to a linear approximation to the integral. This approximation is shown in Figure 7.9(c). In this case, the integral is given by the summation of the trapezoidal areas. The sum of these areas leads to Eq. (7.26). Note that b_0 is zero and a_0 is twice the average of the c_i values. Thus, the first term in Eq. (7.24) is the average (or center of gravity) of the curve.

7.2.3.5 Cumulative angular function

Fourier descriptors can be obtained by using many boundary representations. In a straightforward approach, we could consider, for example, that t and $c(t)$ define the angle and modulus of a polar parameterization of the boundary. However, this representation is not very general. For some curves, the polar form does not define a single valued curve, and thus we cannot apply Fourier expansions. A more general description of curves can be obtained by using the angular function parameterization. This function was already defined in Chapter 4 in the discussion about curvature.

The angular function $\varphi(s)$ measures the angular direction of the tangent line as a function of arc length. Figure 7.10 shows the angular direction at a point in a curve. In Cosgriff (1960), this angular function was used to obtain a set of Fourier descriptors. However, this first approach to Fourier characterization has some undesirable properties. The main problem is that the angular function has discontinuities even for smooth curves. This is because the angular direction is
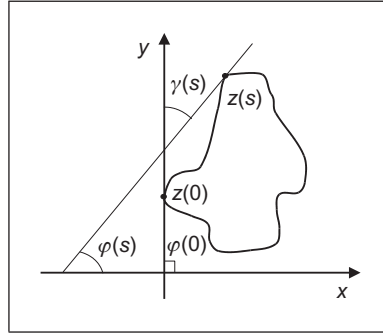
FIGURE 7.10

Angular direction.

bounded from zero to 2π. Thus, the function has **discontinuities** when the angular direction increases to a value of more than 2π or decreases to be less than zero (since it will change abruptly to remain within bounds). In Zahn and Roskies' approach (Zahn and Roskies, 1972), this problem is eliminated by considering a normalized form of the cumulative angular function.

The *cumulative angular function* at a point in the curve is defined as the amount of angular change from the starting point. It is called **cumulative** since it represents the summation of the angular change to each point. Angular change is given by the derivative of the angular function $\varphi(s)$. We discussed in Chapter 4 that this derivative corresponds to the curvature $\kappa(s)$. Thus, the cumulative angular function at the point given by s can be defined as

$$\gamma(s) = \int_0^s \kappa(r)\mathrm{d}r - \kappa(0) \tag{7.28}$$

Here, the parameter s takes values from zero to L (i.e., the length of the curve). Thus, the initial and final values of the function are $\gamma(0) = 0$ and $\gamma(L) = -2\pi$, respectively. It is important to note that in order to obtain the final value of -2π, the curve must be traced in a clockwise direction. Figure 7.10 shows the relation between the angular function and the cumulative angular function. In the figure, $z(0)$ defines the initial point in the curve. The value of $\gamma(s)$ is given by the angle formed by the inclination of the tangent to $z(0)$ and that of the tangent to the point $z(s)$. If we move the point $z(s)$ along the curve, this angle will change until it reaches the value of -2π. In Eq. (7.28), the cumulative angle is obtained by adding the small angular increments for each point.

The cumulative angular function avoids the discontinuities of the angular function. However, it still has two problems. First, it has a discontinuity at the

end. Secondly, its value depends on the length of curve analyzed. These problems can be solved by defining the normalized function $\gamma^*(t)$, where

$$\gamma^*(t) = \gamma\left(\frac{L}{2\pi}t\right) + t \tag{7.29}$$

Here, t takes values from 0 to 2π. The factor $L/2\pi$ normalizes the angular function such that it does not change when the curve is scaled, i.e., when $t = 2\pi$, the function evaluates the final point of the function $\gamma(s)$. The term t is included to avoid discontinuities at the end of the function (remember that the function is periodic), i.e., it makes that $\gamma^*(0) = \gamma^*(2\pi) = 0$. Additionally, it causes the cumulative angle for a circle to be zero. This is consistent as a circle is generally considered the simplest curve and, intuitively, simple curves will have simple representations.

Figure 7.11 shows the definitions of the cumulative angular function with two examples. Figure 7.11(b)−(d) defines the angular functions for a circle in Figure 7.11(a). Figure 7.11(f)−(h) defines the angular functions for the rose in Figure 7.11(e). Figure 7.11(b)−(f) defines the angular function $\varphi(s)$. We can observe the typical toroidal form. Once the curve is greater than 2π, there is a discontinuity while its value returns to zero. The position of the discontinuity actually depends on the selection of the starting point. The cumulative function $\gamma(s)$ shown in Figure 7.11(c) and (g) inverts the function and eliminates discontinuities. However, the start and end points are not the same. If we consider that this function is periodic, there is a discontinuity at the end of each period. The normalized form $\gamma^*(t)$ shown in Figure 7.11(d) and (h) has no discontinuity and the period is normalized to 2π.

The normalized cumulative functions are very nice indeed. However, it is tricky to compute them from images. Additionally, since they are based on measures of changes in angle, they are very sensitive to noise and difficult to compute at inflexion points (e.g., corners). Code 7.1 illustrates the computation of the angular functions for a curve given by a sequence of pixels. The matrices X and Y store the coordinates of each pixel. The code has two important steps. First, the computation of the angular function stored in the matrix A. Generally, if we use only the neighboring points to compute the angular function, then the resulting function is useless due to noise and discretization errors. Thus, it is necessary to include a procedure that can obtain accurate measures. For purposes of illustration, in the presented code, we average the position of pixels in order to filter out noise; however, other techniques such as the fitting process discussed in Section 4.4.1 can provide a suitable alternative. The second important step is the computation of the cumulative function. In this case, the increment in the angle cannot be computed as the simple difference between the current and precedent angular values. This will produce as result a discontinuous function. Thus, we need to consider the periodicity of the angles. In the code, this is achieved by checking the increment in the angle. If it is greater than a threshold, we consider that the angle has exceeded the limits of 0 or 2π.
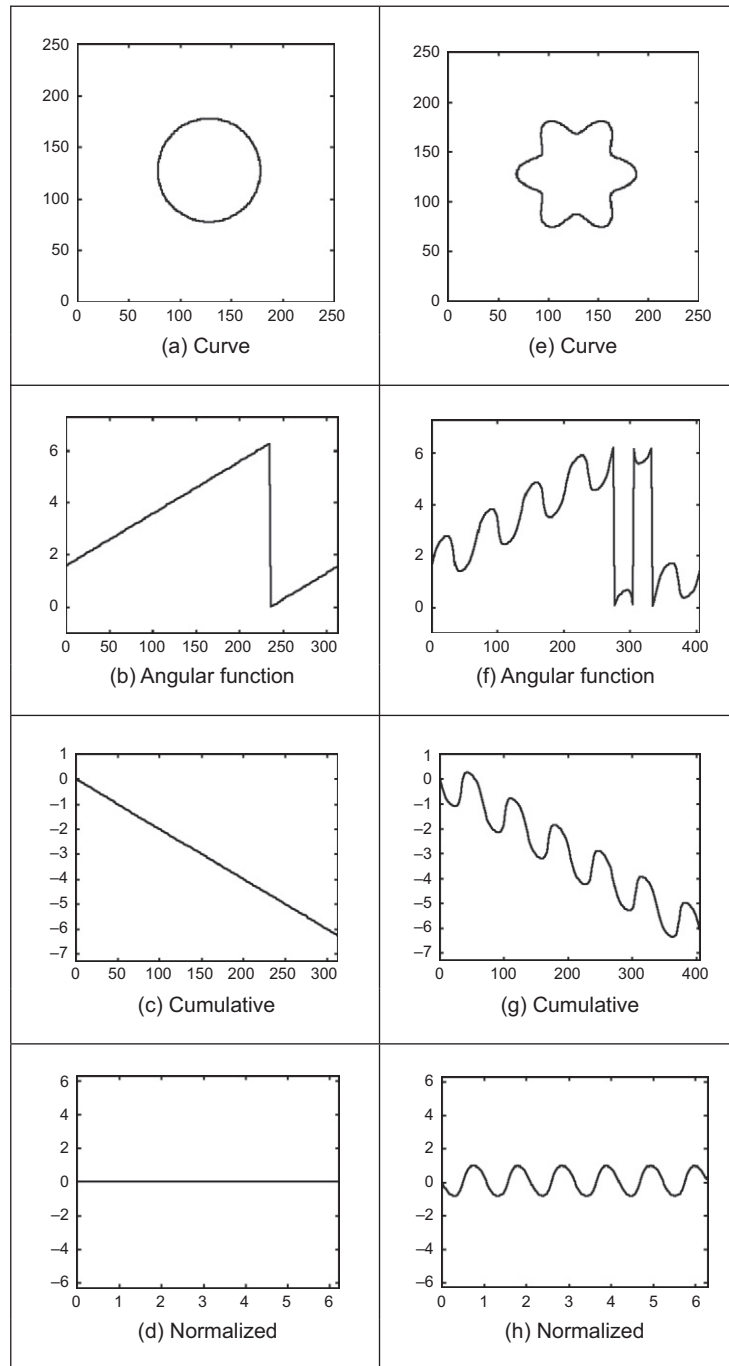
FIGURE 7.11

Angular function and cumulative angular function.

```
%Angular function
 function AngFuncDescrp(curve)

%Function
 X=curve(1,:);  Y=curve(2,:);
 M=size(X,2);   %number points

%Arc length
 S=zeros(1,m);
 S(1)=sqrt((X(1)-X(m))^2+(Y(1)-Y(m))^2);
 for i=2:m
    S(i)=S(i-1)+sqrt((X(i)-X(i-1))^2+(Y(i)-Y(i-1))^2);
 End
 L=S(m);

%Normalised Parameter
 t=(2*pi*S)/L;

%Graph of the curve
 subplot(3,3,1);
 plot(X,Y);
 mx=max(max(X),max(Y))+10;
 axis([0,mx,0,mx]);   axis square;       %Aspect ratio

%Graph of the angular function y'/x'
 avrg=10;
 A=zeros(1,m);
 for i=1:m
    x1=0; x2=0; y1=0; y2=0;
    for j=1:avrg
       pa=i-j; pb=i+j;
       if(pa<1) pa=m+pa; end
       if(pb>m) pb=pb-m; end
       x1=x1+X(pa); y1=y1+Y(pa);
       x2=x2+X(pb); y2=y2+Y(pb);
    end
       x1=x1/avrg; y1=y1/avrg;
       x2=x2/avrg; y2=y2/avrg;
       dx=x2-x1;    dy=y2-y1;

       if(dx==0) dx=.00001; end
          if dx>0 & dy>0
          A(i)=atan(dy/dx);
          elseif dx>0 & dy<0
          A(i)=atan(dy/dx)+2*pi;
       else
          A(i)=atan(dy/dx)+pi;
       end
    end
```

CODE 7.1

Angular functions.

```
        subplot(3,3,2);
        plot(S,A);
        axis([0,S(m),-1,2*pi+1]);

%Cumulative angular    G(s)=-2pi
 G=zeros(1,m);
 for i=2:m
    d=min(abs(A(i)-A(i-1)),abs(abs(A(i)-A(i-1))-2*pi));

    if d>.5
       G(i)=G(i-1);
    elseif (A(i)-A(i-1))<-pi
       G(i)=G(i-1)-(A(i)-A(i-1)+2*pi);
    elseif (A(i)-A(i-1))>pi
       G(i)=G(i-1)-(A(i)-A(i-1)-2*pi);
    else
       G(i)=G(i-1)-(A(i)-A(i-1));
    end
 end

 subplot(3,3,3);
 plot(S,G);
 axis([0,S(m),-2*pi-1,1]);

%Cumulative angular Normalised
 F=G+t;

 subplot(3,3,4);
 plot(t,F);
 axis([0,2*pi,-2*pi,2*pi]);
```

CODE 7.1

(Continued)

Figure 7.12 shows an example of the angular functions computed using Code 7.1, for a discrete curve. These are similar to those in Figure 7.11(a)−(d) but show noise due to discretization which produces a ragged effect on the computed values. The effects of noise will be reduced if we use more points to compute the average in the angular function. However, this reduces the level of detail in the curve. Additionally, it makes it more difficult to detect when the angle exceeds the limits of 0 or 2π. In a Fourier expansion, noise will affect the coefficients of the high-frequency components, as seen in Figure 7.12(d).

In order to obtain a description of the curve, we need to expand $\gamma^*(t)$ in Fourier series. In a straightforward approach, we can obtain $\gamma^*(t)$ from an image and apply the definition in Eq. (7.27) for $c(t) = \gamma^*(t)$. However, we can obtain a computationally more attractive development with some algebraic simplifications. By considering the form of the integral in Eq. (7.13), we have that

$$a_k^* = \frac{1}{\pi}\int_0^{2\pi}\gamma^*(t)\cos(kt)\mathrm{d}t \quad \text{and} \quad b_k^* = \frac{1}{\pi}\int_0^{2\pi}\gamma^*(t)\sin(kt)\mathrm{d}t \qquad (7.30)$$
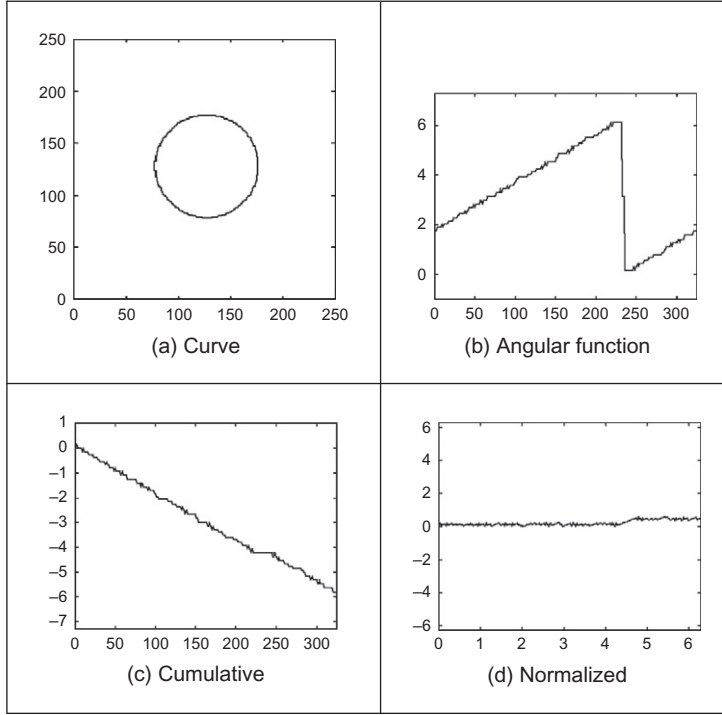
FIGURE 7.12
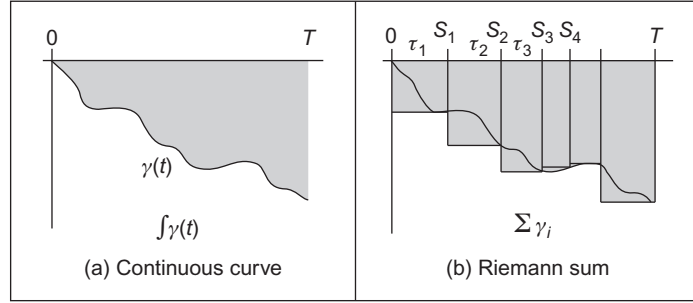
Discrete computation of the angular functions.

By substitution of Eq. (7.29), we obtain

$$a_0^* = \frac{1}{\pi} \int_0^{2\pi} \gamma((L/2\pi)t)\mathrm{d}t + \frac{1}{\pi} \int_0^{2\pi} t \ \mathrm{d}t$$

$$a_k^* = \frac{1}{\pi} \int_0^{2\pi} \gamma((L/2\pi)t)\cos(kt)\mathrm{d}t + \frac{1}{\pi} \int_0^{2\pi} t \cos(kt)\mathrm{d}t \qquad (7.31)$$

$$b_k^* = \frac{1}{\pi} \int_0^{2\pi} \gamma((L/2\pi)t)\sin(kt)\mathrm{d}t + \frac{1}{\pi} \int_0^{2\pi} t \sin(kt)\mathrm{d}t$$

By computing the second integrals of each coefficient, we obtain a simpler form as

$$a_0^* = 2\pi + \frac{1}{\pi} \int_0^{2\pi} \gamma((L/2\pi)t)\mathrm{d}t$$

$$a_k^* = \frac{1}{\pi} \int_0^{2\pi} \gamma((L/2\pi)t)\cos(kt)\mathrm{d}t \qquad (7.32)$$

$$b_k^* = -\frac{2}{k} + \frac{1}{\pi} \int_0^{2\pi} \gamma((L/2\pi)t)\sin(kt)\mathrm{d}t$$

FIGURE 7.13

Integral approximations.

In an image, we measure distances, thus it is better to express these equations in arc-length form. For that, we know that $s = (L/2\pi)t$. Thus,

$$dt = \frac{2\pi}{L}ds \tag{7.33}$$

Accordingly, the coefficients in Eq. (7.32) can be rewritten as

$$a_0^* = 2\pi + \frac{2}{L}\int_0^L \gamma(s)ds$$

$$a_k^* = \frac{2}{L}\int_0^L \gamma(s)\cos\left(\frac{2\pi k}{L}s\right)ds \tag{7.34}$$

$$b_k^* = -\frac{2}{k} + \frac{2}{L}\int_0^L \gamma(s)\sin\left(\frac{2\pi k}{L}s\right)ds$$

In a similar way to Eq. (7.26), the Fourier descriptors can be computed by approximating the integral as a summation of rectangular areas. This is shown in Figure 7.13. Here, the discrete approximation is formed by rectangles of length τ_i and height γ_i. Thus,

$$a_0^* = 2\pi + \frac{2}{L}\sum_{i=1}^{m}\gamma_i\tau_i$$

$$a_k^* = \frac{2}{L}\sum_{i=1}^{m}\gamma_i\tau_i \cos\left(\frac{2\pi k}{L}s_i\right) \tag{7.35}$$

$$b_k^* = -\frac{2}{k} + \frac{2}{L}\sum_{i=1}^{m}\gamma_i\tau_i \sin\left(\frac{2\pi k}{L}s_i\right)$$

where s_i is the arc length at the ith point. Note that

$$s_i = \sum_{r=1}^{i} \tau_r \qquad (7.36)$$

It is important to observe that although the definitions in Eq. (7.35) use only the discrete values of $\gamma(t)$, they obtain a Fourier expansion of $\gamma^*(t)$. In the original formulation (Zahn and Roskies, 1972), an alternative form of the summation is obtained by rewriting the coefficients in terms of the increments of the angular function. In this case, the integrals in Eq. (7.34) are evaluated for each interval. Thus, the coefficients are represented as a summation of integrals of constant values as

$$a_0^* = 2\pi + \frac{2}{L} \sum_{i=1}^{m} \int_{s_{i-1}}^{s_i} \gamma_i \, ds$$

$$a_k^* = \frac{2}{L} \sum_{i=1}^{m} \int_{s_{i-1}}^{s_i} \gamma_i \cos\left(\frac{2\pi k}{L} s\right) ds \qquad (7.37)$$

$$b_k^* = -\frac{2}{k} + \frac{2}{L} \sum_{i=1}^{m} \int_{s_{i-1}}^{s_i} \gamma_i \sin\left(\frac{2\pi k}{L} s\right) ds$$

By evaluating the integral, we obtain

$$a_0^* = 2\pi + \frac{2}{L} \sum_{i=1}^{m} \gamma_i(s_i - s_{i-1})$$

$$a_k^* = \frac{1}{\pi k} \sum_{i=1}^{m} \gamma_i \left(\sin\left(\frac{2\pi k}{L} s_i\right) - \sin\left(\frac{2\pi k}{L} s_{i-1}\right) \right) \qquad (7.38)$$

$$b_k^* = -\frac{2}{k} + \frac{1}{\pi k} \sum_{i=1}^{m} \gamma_i \left(\cos\left(\frac{2\pi k}{L} s_i\right) - \cos\left(\frac{2\pi k}{L} s_{i-1}\right) \right)$$

A further simplification can be obtained by considering that Eq. (7.28) can be expressed in discrete form as

$$\gamma_i = \sum_{r=1}^{i} \kappa_r \tau_r - \kappa_0 \qquad (7.39)$$

where κ_r is the curvature (i.e., the difference of the angular function) at the rth point. Thus,

$$a_0^* = -2\pi - \frac{2}{L}\sum_{i=1}^{m}\kappa_i s_{i-1}$$

$$a_k^* = -\frac{1}{\pi k}\sum_{i=1}^{m}\kappa_i \tau_i \sin\left(\frac{2\pi k}{L}s_{i-1}\right) \qquad (7.40)$$

$$b_k^* = -\frac{2}{k} - \frac{1}{\pi k}\sum_{i=1}^{m}\kappa_i \tau_i \cos\left(\frac{2\pi k}{L}s_{i-1}\right) + \frac{1}{\pi k}\sum_{i=1}^{m}\kappa_i \tau_i$$

Since

$$\sum_{i=1}^{m}\kappa_i \tau_i = 2\pi \qquad (7.41)$$

thus,

$$a_0^* = -2\pi - \frac{2}{L}\sum_{i=1}^{m}\kappa_i s_{i-1}$$

$$a_k^* = -\frac{1}{\pi k}\sum_{i=1}^{m}\kappa_i \tau_i \sin\left(\frac{2\pi k}{L}s_{i-1}\right) \qquad (7.42)$$

$$b_k^* = -\frac{1}{\pi k}\sum_{i=1}^{m}\kappa_i \tau_i \cos\left(\frac{2\pi k}{L}s_{i-1}\right)$$

These equations were originally presented in Zahn and Roskies (1972) and are algebraically equivalent to Eq. (7.35). However, they express the Fourier coefficients in terms of increments in the angular function rather than in terms of the cumulative angular function. In practice, both implementations (Eqs (7.35) and (7.40)) produce equivalent Fourier descriptors.

It is important to note that the parameterization in Eq. (7.21) does not depend on the position of the pixels but only on the change in angular information. That is, shapes in different position and with different scale will be represented by the same curve $\gamma^*(t)$. Thus, the Fourier descriptors obtained are **scale** and **translation** invariant. **Rotation**-invariant descriptors can be obtained by considering the shift invariant property of the coefficients' amplitude. Rotating a curve in an image produces a shift in the angular function. This is because the rotation changes the starting point in the curve description. Thus, according to Section 7.2.3.2, the values

$$|c_k^*| = \sqrt{(a_k^*)^2 + (b_k^*)^2} \qquad (7.43)$$

provide a rotation, scale, and translation-invariant description. The function AngFourierDescrp in Code 7.2 computes the Fourier descriptors in this equation by using the definitions in Eq. (7.35). This code uses the angular functions in Code 7.1.

```
%Fourier descriptors based on the Angular function
 function AngFuncDescrp(curve,n,scale)
                        %n=number coefficients
                        %if n=0 then n=m/2
                        %Scale amplitude output

%Angular functions
 AngFuncDescrp(curve);

%Fourier Descriptors
 if(n==0) n=floor(m/2); end;  %number of coefficients

 a=zeros(1,n); b=zeros(1,n);  %Fourier coefficients

for k=1:n
    a(k)=a(k)+G(1)*(S(1))*cos(2*pi*k*S(1)/L);
    b(k)=b(k)+G(1)*(S(1))*sin(2*pi*k*S(1)/L);
       for i=2:m
         a(k)=a(k)+G(i)*(S(i)-S(i-1))*cos(2*pi*k*S(i)/L);
         b(k)=b(k)+G(i)*(S(i)-S(i-1))*sin(2*pi*k*S(i)/L);
       end
    a(k)=a(k)*(2/L);
    b(k)=b(k)*(2/L)-2/k;
end

%Graphs
 subplot(3,3,7);
 bar(a);
 axis([0,n,-scale,scale]);

 subplot(3,3,8);
 bar(b);
 axis([0,n,-scale,scale]);

%Rotation invariant Fourier descriptors
    CA=zeros(1,n);
    for k=1:n
        CA(k)=sqrt(a(k)^2+b(k)^2);
    end

%Graph of the angular coefficients
 subplot(3,3,9);
 bar(CA);
 axis([0,n,-scale,scale]);
```
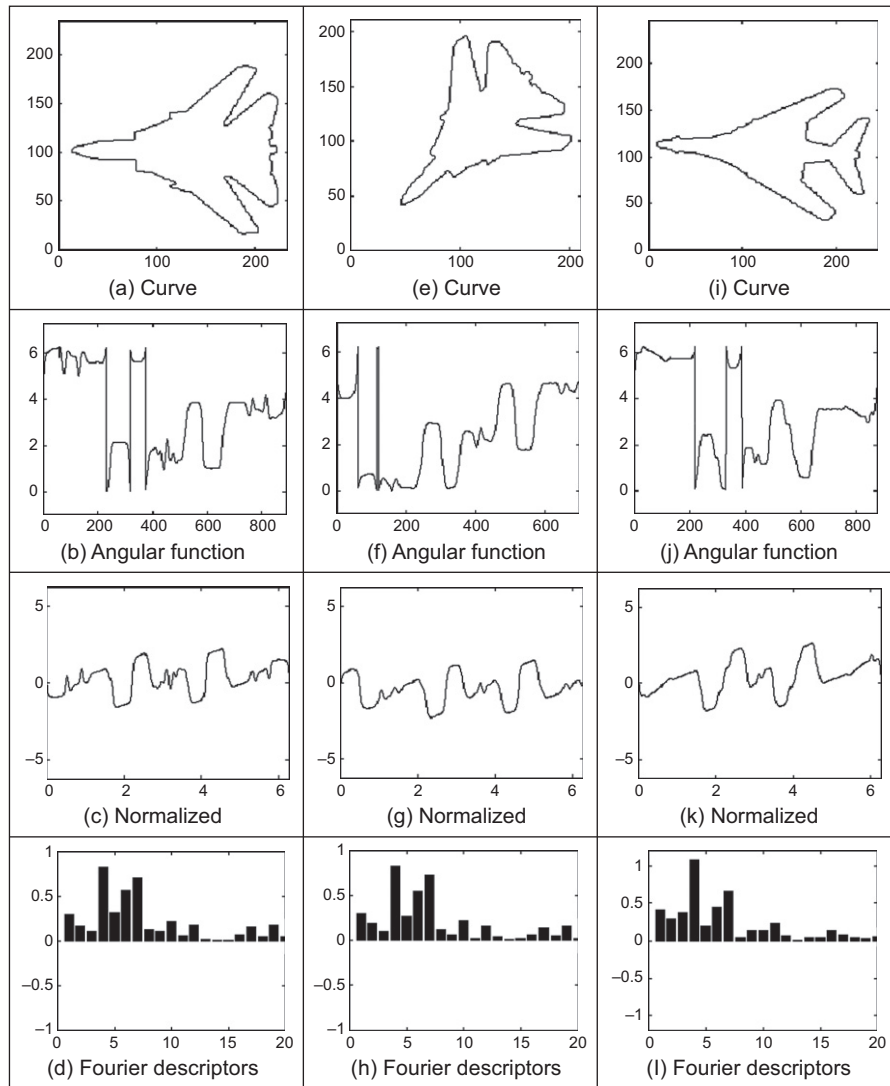
CODE 7.2

Angular Fourier descriptors.

FIGURE 7.14

Example of angular Fourier descriptors.

Figure 7.14 shows three examples of the results obtained using Code 7.2. In each example, we show the curve, the angular function, the cumulative normalized angular function, and the Fourier descriptors. The curves in Figure 7.14(a) and (e) represent the same object (the contour of an F-14 fighter), but the curve in Figure 7.14(e) was scaled and rotated. We can see that the angular function
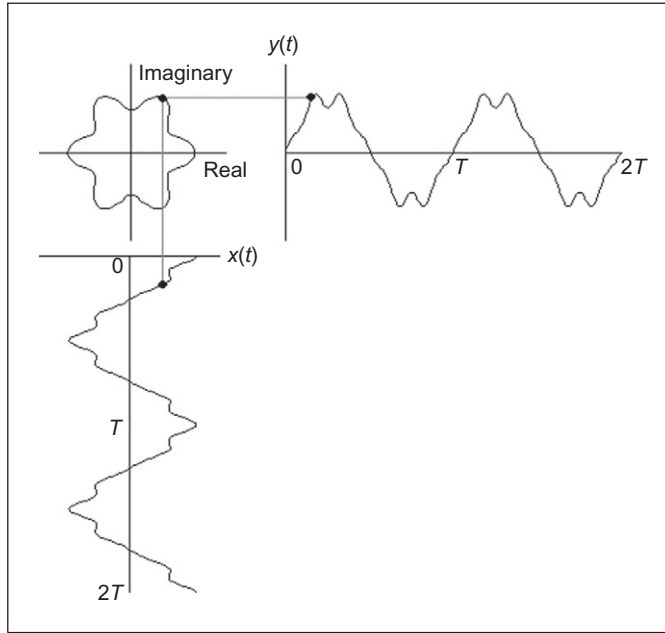
FIGURE 7.15

Example of complex curve representation.

changes significantly, while the normalized function is very similar but with a remarkable shift due to the rotation. The Fourier descriptors shown in Figure 7.14(d) and (h) are quite similar since they characterize the same object. We can see a clear difference between the normalized angular function for the object presented in Figure 7.14(i) (the contour of a different plane, a B1 bomber). These examples show that Fourier coefficients are indeed invariant to scale and rotation and that they can be used to characterize different objects.

7.2.3.6 Elliptic Fourier descriptors

The cumulative angular function transforms the 2D description of a curve into a 1D periodic function suitable for Fourier analysis. In contrast, *elliptic Fourier descriptors* maintain the description of the curve in a 2D space (Granlund, 1972). This is achieved by considering that the image space defines the complex plane, i.e., each pixel is represented by a complex number. The first coordinate represents the real part, while the second coordinate represents the imaginary part. Thus, a curve is defined as

$$c(t) = x(t) + jy(t) \tag{7.44}$$

Here, we will consider that the parameter t is given by the arc-length parameterization. Figure 7.15 shows an example of the complex representation of a

curve. This example illustrates two periods of each component of the curve. Generally, $T = 2\pi$, thus the fundamental frequency is $\omega = 1$. It is important to note that this representation can be used to describe open curves. In this case, the curve is traced twice in opposite directions. In fact, this representation is very general and can be extended to obtain the elliptic Fourier description of irregular curves (i.e., those without derivative information) (Montiel et al., 1996, 1997).

In order to obtain the elliptic Fourier descriptors of a curve, we need to obtain the Fourier expansion of the curve in Eq. (7.44). The Fourier expansion can be performed by using the complex or trigonometric form. In the original work in Granlund (1972), the expansion is expressed in the complex form. However, other works have used the trigonometric representation (Kuhl and Giardina, 1982). Here, we will pass from the complex form to the trigonometric representation. The trigonometric representation is more intuitive and easier to implement.

According to Eq. (7.5), we have that the elliptic coefficients are defined by

$$c_k = c_{xk} + \mathrm{j}c_{yk} \tag{7.45}$$

where

$$c_{xk} = \frac{1}{T}\int_0^T x(t)\mathrm{e}^{-\mathrm{j}k\omega t} \quad \text{and} \quad c_{yk} = \frac{1}{T}\int_0^T y(t)\mathrm{e}^{-\mathrm{j}k\omega t} \tag{7.46}$$

By following Eq. (7.12), we note that each term in this expression can be defined by a pair of coefficients, i.e.,

$$
\begin{aligned}
c_{xk} &= \frac{a_{xk} - \mathrm{j}b_{xk}}{2} & c_{yk} &= \frac{a_{yk} - \mathrm{j}b_{yk}}{2} \\
c_{x-k} &= \frac{a_{xk} + \mathrm{j}b_{xk}}{2} & c_{y-k} &= \frac{a_{yk} + \mathrm{j}b_{yk}}{2}
\end{aligned}
\tag{7.47}
$$

Based on Eq. (7.13), the trigonometric coefficients are defined as

$$
\begin{aligned}
a_{xk} &= \frac{2}{T}\int_0^T x(t)\cos(k\omega t)\mathrm{d}t & \text{and} \quad b_{xk} &= \frac{2}{T}\int_0^T x(t)\sin(k\omega t)\mathrm{d}t \\
a_{yk} &= \frac{2}{T}\int_0^T y(t)\cos(k\omega t)\mathrm{d}t & \text{and} \quad b_{yk} &= \frac{2}{T}\int_0^T y(t)\sin(k\omega t)\mathrm{d}t
\end{aligned}
\tag{7.48}
$$

That according to Eq. (7.27) can be computed by the discrete approximation given by

$$
\begin{aligned}
a_{xk} &= \frac{2}{m}\sum_{i=1}^{m} x_i \cos(k\omega i\tau) & \text{and} \quad b_{xk} &= \frac{2}{m}\sum_{i=1}^{m} x_i \sin(k\omega i\tau) \\
a_{yk} &= \frac{2}{m}\sum_{i=1}^{m} y_i \cos(k\omega i\tau) & \text{and} \quad b_{yk} &= \frac{2}{m}\sum_{i=1}^{m} y_i \sin(k\omega i\tau)
\end{aligned}
\tag{7.49}
$$

where x_i and y_i define the value of the functions $x(t)$ and $y(t)$ at the sampling point i. By considering Eqs (7.45) and (7.47), we can express c_k as the sum of a pair of complex numbers, i.e.,

$$c_k = A_k - jB_k \quad \text{and} \quad c_{-k} = A_k + jB_k \tag{7.50}$$

where

$$A_k = \frac{a_{xk} + ja_{yk}}{2} \quad \text{and} \quad B_k = \frac{b_{xk} + jb_{yk}}{2} \tag{7.51}$$

Based on the definition in Eq. (7.45), the curve can be expressed in the exponential form given in Eq. (7.6) as

$$c(t) = c_0 + \sum_{k=1}^{\infty}(A_k - jB_k)e^{jk\omega t} + \sum_{k=-\infty}^{-1}(A_k + jB_k)e^{jk\omega t} \tag{7.52}$$

Alternatively, according to Eq. (7.11), the curve can be expressed in trigonometric form as

$$c(t) = \frac{a_{x0}}{2} + \sum_{k=1}^{\infty}\left(a_{xk}\cos(k\omega t) + b_{xk}\sin(k\omega t) + j\left(\frac{a_{y0}}{2} + \sum_{k=1}^{\infty}a_{yk}\cos(k\omega t) + b_{yk}\sin(k\omega t)\right)\right) \tag{7.53}$$

Generally, this equation is expressed in matrix form as

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix} + \sum_{k=1}^{\infty}\begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix}\begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{7.54}$$

Each term in this equation has an interesting geometric interpretation as an elliptic phasor (a rotating vector). That is, for a fixed value of k, the trigonometric summation defines the locus of an ellipse in the complex plane. We can imagine that as we change the parameter t, the point traces ellipses moving at a speed proportional to the harmonic number k. This number indicates how many cycles (i.e., turns) give the point in the time interval from zero to T. Figure 7.16(a) shows this concept. Here, a point in the curve is given as the summation of three vectors that define three terms in Eq. (7.54). As the parameter t changes, each vector defines an elliptic curve. In this interpretation, the values of $a_{x0}/2$ and $a_{y0}/2$ define the start point of the first vector (i.e., the location of the curve). The major axes of each ellipse are given by the values of $|A_k|$ and $|B_k|$. The definition of the ellipse locus for a frequency is determined by the coefficients as shown in Figure 7.16(b).

(a) Sum of three frequencies

(b) Elliptic phasor
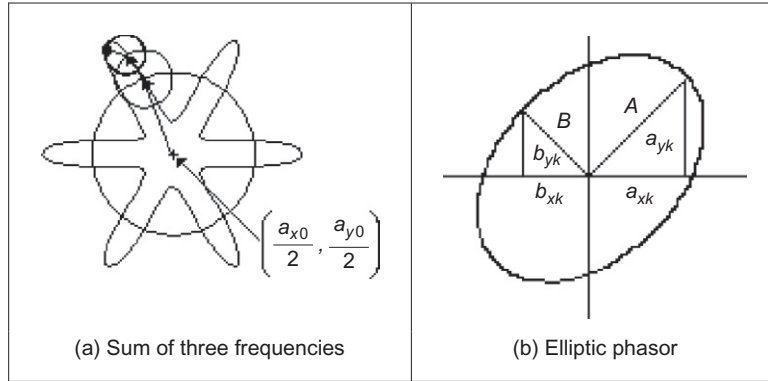
FIGURE 7.16

Example of a contour defined by elliptic Fourier descriptors.

7.2.3.7 Invariance

As in the case of angular Fourier descriptors, elliptic Fourier descriptors can be defined such that they remain invariant to geometric transformations. In order to show these definitions, we must first study how geometric changes in a shape modify the form of the Fourier coefficients. Transformations can be formulated by using both the exponential and the trigonometric form. We will consider changes in translation, rotation, and scale using the trigonometric definition in Eq. (7.54).

Let us denote $c'(t) = x'(t) + jy'(t)$ as the transformed contour. This contour is defined as

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} a'_{x0} \\ a'_{y0} \end{bmatrix} + \sum_{k=1}^{\infty} \begin{bmatrix} a'_{xk} & b'_{xk} \\ a'_{yk} & b'_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{7.55}$$

If the contour is translated by t_x and t_y along the real and the imaginary axes, respectively, we have that

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix} + \sum_{k=1}^{\infty} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{7.56}$$

that is,

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} a_{x0} + 2t_x \\ a_{y0} + 2t_y \end{bmatrix} + \sum_{k=1}^{\infty} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{7.57}$$

Thus, by comparing Eqs (7.55) and (7.57), we have that the relationship between the coefficients of the transformed and original curves is given by

$$a'_{xk} = a_{xk} \quad b'_{xk} = b_{xk} \quad a'_{yk} = a_{yk} \quad b'_{yk} = b_{yk} \quad \text{for } k \neq 0$$
$$a'_{x0} = a_{x0} + 2t_x \quad a'_{y0} = a_{y0} + 2t_y \tag{7.58}$$

Accordingly, all the coefficients remain invariant under translation except a_{x0} and a_{y0}. This result can be intuitively derived by considering that these two coefficients represent the position of the center of gravity of the contour of the shape, and translation changes only the position of the curve.

The change in scale of a contour $c(t)$ can be modeled as the dilation from its center of gravity, i.e., we need to translate the curve to the origin, scale it, and then return it back to its original location. If s represents the scale factor, then these transformations define the curve as

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix} + s \sum_{k=1}^{\infty} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{7.59}$$

Note that in this equation, the scale factor does not modify the coefficients a_{x0} and a_{y0} since the curve is expanded with respect to its center. In order to define the relationships between the curve and its scaled version, we compare Eqs (7.55) and (7.59). Thus,

$$a'_{xk} = sa_{xk} \quad b'_{xk} = sb_{xk} \quad a'_{yk} = sa_{yk} \quad b'_{yk} = sb_{yk} \quad \text{for } k \neq 0$$
$$a'_{x0} = a_{x0} \quad a'_{y0} = a_{y0} \tag{7.60}$$

That is, under dilation, all the coefficients are multiplied by the scale factor except a_{x0} and a_{y0}, which remain invariant.

Rotation can be defined in a similar way to Eq. (7.59). If ρ represents the rotation angle, then we have that

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix} + \begin{bmatrix} \cos(\rho) & \sin(\rho) \\ -\sin(\rho) & \cos(\rho) \end{bmatrix} \sum_{k=1}^{\infty} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{7.61}$$

This equation can be obtained by translating the curve to the origin, rotating it, and then returning it back to its original location. By comparing Eqs (7.55) and (7.61), we have that

$$a'_{xk} = a_{xk} \cos(\rho) + a_{yk} \sin(\rho) \quad b'_{xk} = b_{xk} \cos(\rho) + b_{yk} \sin(\rho)$$
$$a'_{yk} = -a_{xk} \sin(\rho) + a_{yk} \cos(\rho) \quad b'_{yk} = -b_{xk} \sin(\rho) + b_{yk} \cos(\rho) \tag{7.62}$$
$$a'_{x0} = a_{x0} \quad a'_{y0} = a_{y0}$$

That is, under rotation, the coefficients are defined by a linear combination dependent on the rotation angle, except for a_{x0} and a_{y0}, which remain invariant. It is important to note that rotation relationships are also applied for a change in the starting point of the curve.

Equations (7.58), (7.60), and (7.62) define how the elliptic Fourier coefficients change when the curve is translated, scaled, or rotated. We can combine these results to define the changes when the curve undergoes the three transformations. In this case, transformations are applied in succession. Thus,

$$
\begin{aligned}
a'_{xk} &= s(a_{xk}\cos(\rho) + a_{yk}\sin(\rho)) & b'_{xk} &= s(b_{xk}\cos(\rho) + b_{yk}\sin(\rho)) \\
a'_{yk} &= s(-a_{xk}\sin(\rho) + a_{yk}\cos(\rho)) & b'_{yk} &= s(-b_{xk}\sin(\rho) + b_{yk}\cos(\rho)) \\
a'_{x0} &= a_{x0} + 2t_x & a'_{y0} &= a_{y0} + 2t_y
\end{aligned}
\tag{7.63}
$$

Based on this result, we can define alternative invariant descriptors. In order to achieve invariance to translation, when defining the descriptors coefficient for $k = 0$ is not used. In Granlund (1972), invariant descriptors are defined based on the complex form of the coefficients. Alternatively, invariant descriptors can be simply defined as

$$
\frac{|A_k|}{|A_1|} + \frac{|B_k|}{|B_1|}
\tag{7.64}
$$

The advantage of these descriptors with respect to the definition in Granlund (1972) is that they do not involve negative frequencies and that we avoid multiplication by higher frequencies that are more prone to noise. By considering the definitions in Eqs (7.51) and (7.63), we can prove that

$$
\frac{|A'_k|}{|A'_1|} = \frac{\sqrt{a_{xk}^2 + a_{yk}^2}}{\sqrt{a_{x1}^2 + a_{y1}^2}} \quad \text{and} \quad \frac{|B'_k|}{|B'_1|} = \frac{\sqrt{b_{xk}^2 + b_{yk}^2}}{\sqrt{b_{x1}^2 + b_{y1}^2}}
\tag{7.65}
$$

These equations contain neither the **scale** factor, s, nor the **rotation**, ρ. Thus, they are **invariant**. Note that if the square roots are removed, invariance properties are still maintained. However, high-order frequencies can have undesirable effects.

The function EllipticDescrp in Code 7.3 computes the elliptic Fourier descriptors of a curve. The code implements Eqs (7.49) and (7.64) in a straightforward way. By default, the number of coefficients is half of the number of points that define the curve. However, the number of coefficients can be specified by the parameter n. The number of coefficients used defines the level of detail of the characterization. In order to illustrate this idea, we can consider the different curves that are obtained by using a different number of coefficients. Figure 7.17 shows an example of the reconstruction of a contour. In Figure 7.17(a), we can observe that the first coefficient represents an ellipse. When the second coefficient is considered (Figure 7.17(b)), then the ellipse changes into a triangular shape. When adding more coefficients, the contour is refined until the curve represents an accurate approximation of the original contour. In this example, the contour is represented by 100 points. Thus, the maximum number of coefficients is 50.

```
%Elliptic Fourier Descriptors
 function EllipticDescrp(curve,n,scale)
          %n=num coefficients
          %if n=0 then n=m/2
          %Scale amplitud output
%Function from image
 X=curve(1,:);
 Y=curve(2,:);
 m=size(X,2);

%Graph of the curve
        subplot(3,3,1);
        plot(X,Y);
        mx=max(max(X),max(Y))+10;
        axis([0,mx,0,mx]); %Axis of the graph pf the curve
        axis square;       %Aspect ratio

%Graph of X
 p=0:2*pi/m:2*pi-pi/m;        %Parameter
        subplot(3,3,2);
         plot(p,X);
         axis([0,2*pi,0,mx]); %Axis of the graph pf the curve

%Graph of Y
        subplot(3,3,3);
        plot(p,Y);
        axis([0,2*pi,0,mx]); %Axis of the graph pf the curve
%Elliptic Fourier Descriptors
 if(n==0) n=floor(m/2); end; %number of coefficients

 %Fourier Coefficients
 ax=zeros(1,n); bx=zeros(1,n);
 ay=zeros(1,n); by=zeros(1,n);

t=2*pi/m;

  for k=1:n
    for i=1:m
     ax(k)=ax(k)+X(i)*cos(k*t*(i-1));
     bx(k)=bx(k)+X(i)*sin(k*t*(i-1));
     ay(k)=ay(k)+Y(i)*cos(k*t*(i-1));
     by(k)=by(k)+Y(i)*sin(k*t*(i-1));
    end
  ax(k)=ax(k)*(2/m);
  bx(k)=bx(k)*(2/m);
  ay(k)=ay(k)*(2/m);
  by(k)=by(k)*(2/m);
end
%Graph coefficient ax
 subplot(3,3,4);
 bar(ax);
 axis([0,n,-scale,scale]);
```

CODE 7.3

Elliptic Fourier descriptors.

```
%Graph coefficient ay
subplot(3,3,5);
bar(ay);
axis([0,n,-scale,scale]);

%Graph coefficient bx
 subplot(3,3,6);
 bar(bx);
 axis([0,n,-scale,scale]);

%Graph coefficient by
 subplot(3,3,7);
 bar(by);
 axis([0,n,-scale,scale]);

%Invariant
 CE=zeros(1,n);
 for k=1:n
   CE(k)=sqrt((ax(k)^2+ay(k)^2)/(ax(1)^2+ay(1)^2))+
         sqrt((bx(k)^2+by(k)^2)/(bx(1)^2+by(1)^2));
 end

%Graph of Elliptic descriptors
 subplot(3,3,8);
 bar(CE);
 axis([0,n,0,2.2]);
```
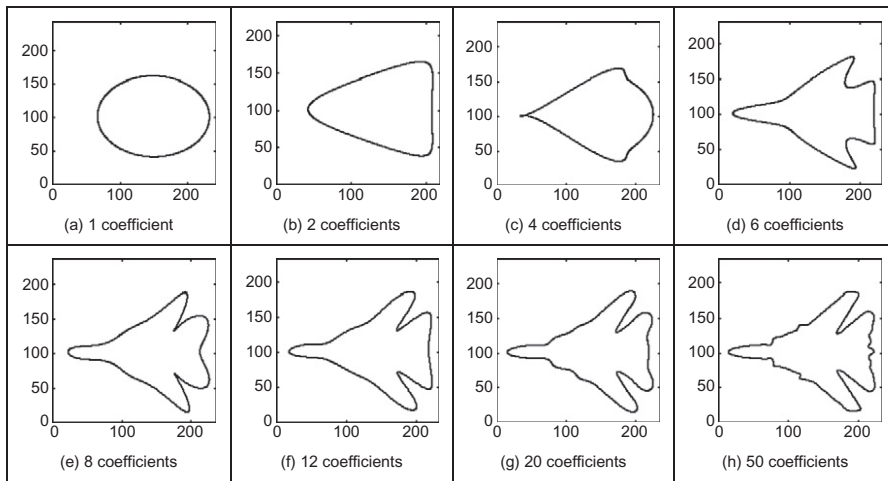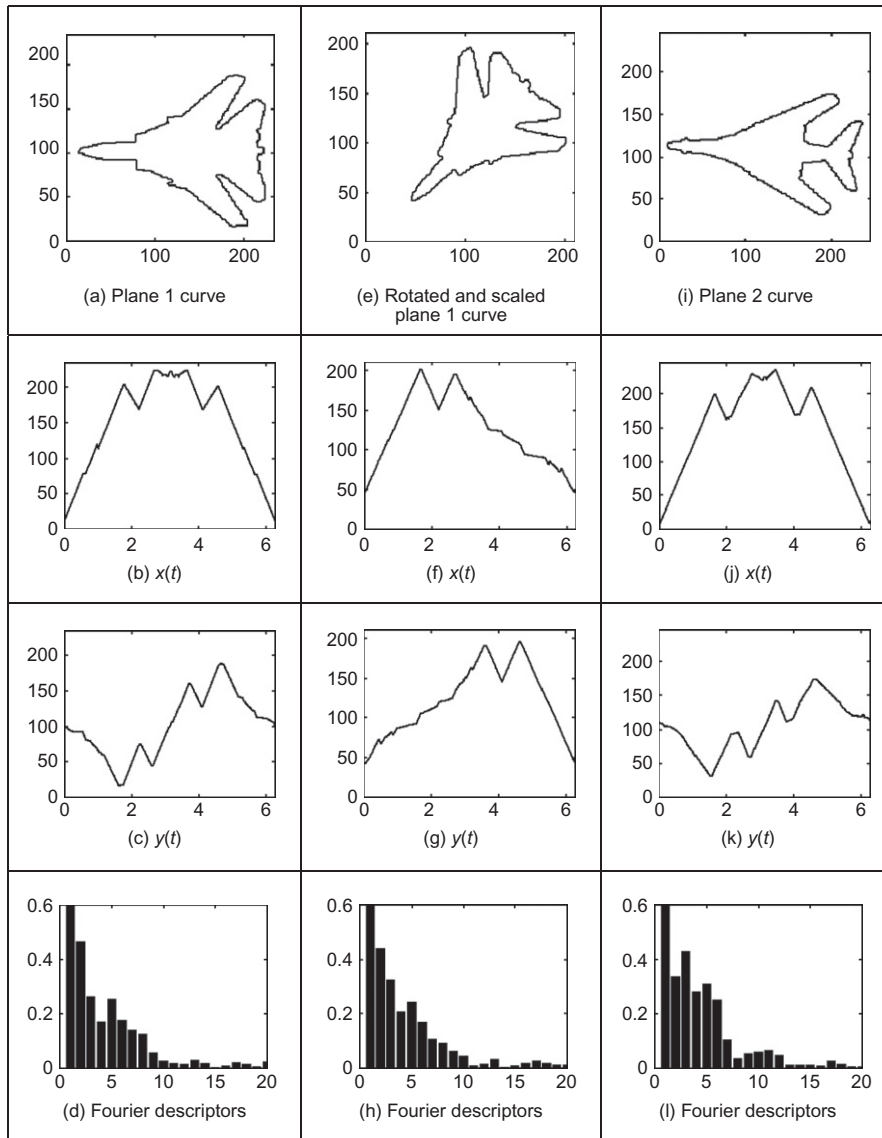
CODE 7.3

(Continued)

FIGURE 7.17

Fourier approximation.

FIGURE 7.18

Example of elliptic Fourier descriptors.

Figure 7.18 shows three examples of the results obtained using Code 7.3. Each example shows the original curve, the x and y coordinate functions and the Fourier descriptors defined in Eq. (7.64). The maximum in Eq. (7.64) is equal to two and is obtained when $k = 1$. In the figure, we have scaled the Fourier

descriptors to show the differences between higher-order coefficients. In this example, we can see that the Fourier descriptors for the curves in Figure 7.18(a) and (e) (F-14 fighter) are very similar. Small differences can be explained by discretization errors. However, the coefficients remain the same after changing its location, orientation, and scale. The descriptors of the curve in Figure 7.18(i) (B1 bomber) are clearly different, showing that elliptic Fourier descriptors truly characterize the shape of an object.

Fourier descriptors are one of the most popular boundary descriptions. As such, they have attracted considerable attention and there are many further aspects. Naturally, we can use the descriptions for shape recognition (Aguado et al., 1998). It is important to mention that some work has suggested that there is some ambiguity in the Fourier characterization. Thus, an alternative set of descriptors has been designed specifically to reduce ambiguities (Crimmins, 1982). However, it is well known that Fourier expansions are unique. Thus, Fourier characterization should uniquely represent a curve. Additionally, the mathematical opacity of the technique in Crimmins (1982) does not lend itself to tutorial type presentation. Interestingly, there has not been much study on alternative decompositions to Fourier though Walsh functions have been suggested for shape representation (Searle, 1970), and recently wavelets have been used (Kashi et al., 1996) (though these are not an orthonormal basis function). The 3D Fourier descriptors were introduced for analysis of simple shapes (Staib and Duncan, 1992) and have recently been found to give good performance in application (Undrill et al., 1997). Fourier descriptors have also been used to model shapes in computer graphics (Aguado et al., 1999). Naturally, Fourier descriptors cannot be used for occluded or mixed shapes, relying on extraction techniques with known indifference to occlusion (the HT, say). However, there have been approaches aimed to classify partial shapes using Fourier descriptors (Lin and Chellappa, 1987).

7.3 Region descriptors

So far, we have concentrated on descriptions of the perimeter or boundary. The natural counterpart is to describe the **region**, or the **area**, by *regional shape descriptors*. Here, there are two main contenders that differ in focus: basic regional descriptors characterize the geometric properties of the region; moments concentrate on density of the region. First though, we shall look at the simpler descriptors.

7.3.1 Basic region descriptors

A region can be described by considering scalar measures based on its geometric properties. The simplest property is given by its size or area. In general, the area of a region in the plane is defined as

$$A(S) = \int_x \int_y I(x, y)\mathrm{d}y \ \mathrm{d}x \tag{7.66}$$

where $I(x,y) = 1$ if the pixel is within a shape, $(x,y) \in S$, and 0 otherwise. In practice, integrals are approximated by summations, i.e.,

$$A(S) = \sum_x \sum_y I(x, y) \Delta A \tag{7.67}$$

where ΔA is the area of one pixel. Thus, if $\Delta A = 1$, then the area is measured in pixels. Area changes with changes in scale. However, it is invariant to image rotation. Small errors in the computation of the area will appear when applying a rotation transformation due to discretization of the image.

Another simple property is defined by the perimeter of the region. If $x(t)$ and $y(t)$ denote the parametric coordinates of a curve enclosing a region S, then the perimeter of the region is defined as

$$P(S) = \int_t \sqrt{x^2(t) + y^2(t)} \mathrm{d}t \tag{7.68}$$

This equation corresponds to the sum of all the infinitesimal arcs that define the curve. In the discrete case, $x(t)$ and $y(t)$ are defined by a set of pixels in the image. Thus, Eq. (7.68) is approximated by

$$P(S) = \sum_i \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{7.69}$$

where x_i and y_i represent the coordinates of the ith pixel forming the curve. Since pixels are organized in a square grid, then the terms in the summation can only take two values. When the pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are 4-neighbors (as shown in Figure 7.1(a)), the summation term is unity. Otherwise, the summation term is equal to $\sqrt{2}$. Note that the discrete approximation in Eq. (7.69) produces small errors in the measured perimeter. As such, it is unlikely that an exact value of $2\pi r$ will be achieved for the perimeter of a circular region of radius r.

Based on the perimeter and area, it is possible to characterize the compactness of a region. *Compactness* is an oft-expressed measure of shape given by the ratio of perimeter to area, i.e.,

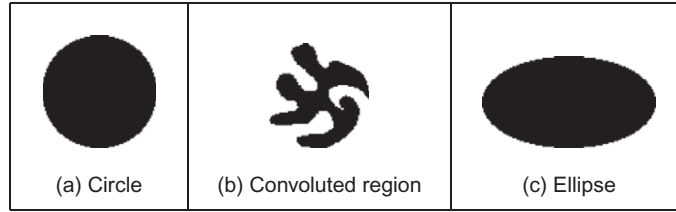$$C(S) = \frac{4\pi A(s)}{P^2(s)} \tag{7.70}$$

FIGURE 7.19

Examples of compactness.

In order to show the meaning of this equation, we can rewrite it as

$$C(S) = \frac{A(s)}{P^2(s)/4\pi} \tag{7.71}$$

Here, the denominator represents the area of a circle whose perimeter is $P(S)$. Thus, compactness measures the ratio between the area of the shape and the circle that can be traced with the same perimeter, i.e., compactness measures the efficiency with which a boundary encloses area. In mathematics, it is known as the *isoperimetric quotient*, which smacks rather of grandiloquency. For a perfectly circular region (Figure 7.19(a)), we have that $C(circle) = 1$, which represents the maximum compactness value: a circle is the most compact shape. Figure 7.19(b) and (c) shows two examples in which compactness is reduced. If we take the perimeter of these regions and draw a circle with the same perimeter, we can observe that the circle contains more area. This means that the shapes are not compact. A shape becomes more compact if we move region pixels far away from the center of gravity of the shape to fill empty spaces closer to the center of gravity. For a perfectly square region, $C(square) = \pi/4$. Note that neither for a perfect square nor for a perfect circle, does the measure include size (the width and radius, respectively). In this way, compactness is a measure of **shape** only. Note that compactness alone is not a good discriminator of a region; low values of C are associated with convoluted regions such as the one in Figure 7.19(b) and also with simple though highly elongated shapes. This ambiguity can be resolved by employing additional shape measures.

Another measure that can be used to characterize regions is *dispersion*. Dispersion (irregularity) has been measured as the ratio of major chord length to area (Chen et al., 1995). A simple version of this measure can be defined as *irregularity*

$$I(S) = \frac{\pi \max((x_i - \bar{x})^2 + (y_i - \bar{y})^2)}{A(S)} \tag{7.72}$$

where $(\bar{x}, \bar{y})$ represent the coordinates of the center of mass of the region. Note that the numerator defines the area of the maximum circle enclosing the region.

Thus, this measure describes the density of the region. An alternative measure of dispersion can, actually, also be expressed as the ratio of the maximum to the minimum radius, i.e., an alternative form of the irregularity

$$\text{IR}(S) = \frac{\max\left(\sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2} \right)}{\min\left(\sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2} \right)} \tag{7.73}$$

This measure defines the ratio between the radius of the maximum circle enclosing the region and the maximum circle that can be contained in the region. Thus, the measure will increase as the region spreads. In this way, the irregularity of a circle is unity, $\text{IR}(\text{circle}) = 1$; the irregularity of a square is $\text{IR}(\text{square}) = \sqrt{2}$, which is larger. As such the measure increases for irregular shapes, whereas the compactness measure decreases. Again, for perfect shapes, the measure is irrespective of size and is a measure of shape only. One **disadvantage** of the irregularity measures is that they are insensitive to slight **discontinuity** in the shape, such as a thin crack in a disk. On the other hand, these discontinuities will be registered by the earlier measures of compactness since the perimeter will increase disproportionately with the area. Naturally, this property might be desired and so irregularity is to be preferred when this property is required. In fact, the perimeter measures will vary with rotation due to the nature of discrete images and are more likely to be affected by noise than the measures of area (since the area measures have inherent averaging properties). Since the irregularity is a ratio of distance measures and compactness is a ratio of area to distance, then intuitively it would appear that irregularity will vary less with noise and rotation. Such factors should be explored in application to check that desired properties have indeed been achieved.

Code 7.4 shows the implementation for the region descriptors. The code is a straightforward implementation of Eqs (7.67), (7.69), (7.70), (7.72), and (7.73). A comparison of these measures for the three regions shown in Figure 7.19 is presented in Figure 7.20. Clearly, for the circle, the compactness and dispersion measures are close to unity. For the ellipse, the compactness decreases while the dispersion increases. The convoluted region has the lowest compactness measure and the highest dispersion values. Clearly, these measurements can be used to characterize and hence discriminate between areas of differing shape.

Other measures, rather than focus on the geometric properties, characterize the structure of a region. This is the case of the *Poincarré measure* and the *Euler number*. The Poincarré measure concerns the number of holes within a region. Alternatively, the Euler number is the difference of the number of connected regions from the number of holes in them. There are many more potential measures for shape description in terms of structure and geometry. Recent interest has developed a measure (Rosin and Zunic, 2005) that can discriminate *rectilinear* regions,

```
%Region descriptors (compactness)

function RegionDescrp(inputimage)

%Image size
[rows,columns]=size(inputimage);

%area
A=0;
  for x=1:columns
    for y=1:rows
       if inputimage(y,x)==0 A=A+1; end
    end
  end

%Obtain Contour
C=Contour(inputimage);

%Perimeter & mean
X=C(1,:);   Y=C(2,:);  m=size(X,2);
mx=X(1); my=Y(1);
P=sqrt((X(1)-X(m))^2+(Y(1)-Y(m))^2);
for i=2:m
  P=P+sqrt((X(i)-X(i-1))^2+(Y(i)-Y(i-1))^2);
  mx=mx+X(i); my=my+Y(i);
end

mx=mx/m; my=my/m;

%Compactness
Cp=4*pi*A/P^2;

%Dispersion
max=0; min=99999;

   for i=1:m
     d=((X(i)-mx)^2+(Y(i)-my)^2);
     if (d>max) max=d; end
     if (d<min) min=d; end
   end

   I=pi*max/A;
   IR=sqrt(max/min);

%Results
disp('perimeter=');     disp(P);
disp('area=');          disp(A);
disp('Compactness=');   disp(Cp);
disp('Dispersion=');    disp(I);
disp('DispersionR=');   disp(IR);
```

CODE 7.4

Evaluating basic region descriptors.

$A(S) = 4917$	$A(S) = 2316$	$A(S) = 6104$
$P(S) = 259.27$	$P(S) = 498.63$	$P(S) = 310.93$
$C(S) = 0.91$	$C(S) = 0.11$	$C(S) = 0.79$
$I(S) = 1.00$	$I(S) = 2.24$	$I(S) = 1.85$
$IR(S) = 1.03$	$IR(S) = 6.67$	$IR(S) = 1.91$
(a) Descriptors for the circle	(b) Descriptors for the convoluted region	(c) Descriptors for the ellipse

FIGURE 7.20

Basic region descriptors.

e.g., for discriminating buildings from within remotely sensed images. We could evaluate global or local **curvature** (convexity and concavity) as a further measure of geometry; we could investigate **proximity** and **disposition** as a further measure of structure. However, these do not have the advantages of a unified structure. We are simply suggesting measures with descriptive ability, but this ability is reduced by the correlation between different measures. We have already seen the link between the Poincarré measure and the Euler number; there is a natural link between circularity and irregularity. But the region descriptors we have considered so far lack structure and are largely heuristic—though clearly they may have sufficient descriptive ability for some applications. As such we shall now look at a **unified** basis for shape description which aims to reduce this correlation and provides a unified theoretical basis for region description, with some similarity to the advantages of the frequency selectivity in a Fourier transform description.

7.3.2 Moments
7.3.2.1 Basic properties
Moments describe a shape's **layout** (the arrangement of its pixels), a bit like combining area, compactness, irregularity, and higher-order descriptions together. Moments are a **global** description of a shape, accruing this same advantage as Fourier descriptors since there is selectivity which is an in-built ability to discern, and filter, noise. Further, in image analysis, they are *statistical moments*, as opposed to **mechanical** ones, but the two are analogous. For example, the mechanical moment of inertia describes the rate of change in momentum; the statistical second-order moment describes the rate of change in a shape's area. In this way, statistical moments can be considered as a global region description. Moments for image analysis were again originally introduced in the 1960s (Hu, 1962) (an exciting time for computer vision researchers too!) and an excellent review is available (Prokop and Reeves, 1992).

Moments are actually often associated more with **statistical** pattern recognition, than with **model-based** vision since a major assumption is that there is an

unoccluded view of the target shape. Target images are often derived by thresholding, usually one of the optimal forms that can require a single object in the field of view. More complex applications, including handling occlusion, could presuppose **feature extraction** by some means, with a model to in-fill for the missing parts. However, moments do provide a global description with invariance properties and with the advantages of a compact description aimed to avoid the effects of noise. As such, they have proved popular and successful in many applications.

The *2D Cartesian moment* is actually associated with an order that starts from low (where the lowest is zero) up to higher orders. The moment of order p and q, m_{pq} of a function $I(x,y)$ is defined as

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q I(x, y) \mathrm{d}x \, \mathrm{d}y \qquad (7.74)$$

For discrete images, Eq. (7.74) is usually approximated by

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y) \Delta A \qquad (7.75)$$

where ΔA is again the area of a pixel. These descriptors have a **uniqueness** property in that it can be shown that if the function satisfies certain conditions, then moments of all orders exist. Also, and conversely, the set of descriptors uniquely determines the original function, in a manner similar to reconstruction via the inverse Fourier transform. However, these moments are descriptors rather than a specification which can be used to reconstruct a shape. The *zero-order moment*, m_{00}, is

$$m_{00} = \sum_x \sum_y I(x, y) \Delta A \qquad (7.76)$$

which represents the total mass of a function. Note that this equation is equal to Eq. (7.67) when $I(x,y)$ takes values of zero and one. However, Eq. (7.76) is more general since the function $I(x,y)$ can take a range of values. In the definition of moments, these values are generally related to density. The two *first-order moments*, m_{01} and m_{10}, are given by

$$m_{10} = \sum_x \sum_y x \, I(x, y) \Delta A \quad m_{01} = \sum_x \sum_y y \, I(x, y) \Delta A \qquad (7.77)$$

For binary images, these values are proportional to the shape's center coordinates (the values merely require division by the shape's area). In general, the *center of mass* $(\overline{x}, \overline{y})$ can be calculated from the ratio of the first-order to the zero-order components as

$$\overline{x} = \frac{m_{10}}{m_{00}} \quad \overline{y} = \frac{m_{01}}{m_{00}} \qquad (7.78)$$

The first 10 x-axis moments of an ellipse are shown in Figure 7.21. The moments rise exponentially so are plotted in logarithmic form. Evidently, the
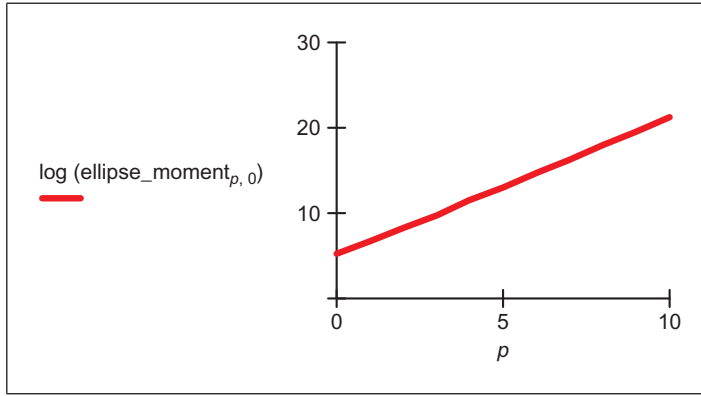
FIGURE 7.21

Horizontal axis ellipse moments.

moments provide a set of descriptions of the shape: measures that can be collected together to differentiate between different shapes.

Should there be an intensity transformation that **scales** brightness by a particular factor, say α, such that a new image $I'(x,y)$ is a transformed version of the original one $I(x,y)$ given by

$$I'(x, y) = \alpha I(x, y) \tag{7.79}$$

Then the transformed moment values m'_{pq} are related to those of the original shape m_{pq} by

$$m'_{pq} = \alpha m_{pq} \tag{7.80}$$

Should it be required to distinguish *mirror symmetry* (reflection of a shape about a chosen axis), then the rotation of a shape about the, say, x-axis gives a new shape $I'(x,y)$ which is the reflection of the shape $I(x,y)$ given by

$$I'(x, y) = I(-x, y) \tag{7.81}$$

The transformed moment values can be given in terms of the original shape's moments as

$$m'_{pq} = (-1)^p m_{pq} \tag{7.82}$$

However, we are usually concerned with more basic invariants than mirror images, namely invariance to **position**, **size**, and **rotation**. Given that we now have an estimate of a shape's center (in fact, a reference point for that shape), the *centralized moments*, μ_{pq}, which are invariant to **translation** can be defined as

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \Delta A \tag{7.83}$$

Clearly, the zero-order **centralized** moment is again the shape's area. However, the first-order centralized moment μ_{01} is given by

$$
\begin{aligned}
\mu_{01} &= \sum_x \sum_y (y-\overline{y})^1 I(x,y)\Delta A \\
&= \sum_x \sum_y yI(x,y)\Delta A - \sum_x \sum_y \overline{y}I(x,y)\Delta A \\
&= m_{01} - \overline{y}\sum_x \sum_y I(x,y)\Delta A
\end{aligned}
\tag{7.84}
$$

From Eq. (7.77), $m_{01} = \sum_x\sum_y yI(x,y)\Delta A$ and from Eq. (7.78), $\overline{y} = m_{01}/m_{00}$, so

$$
\begin{aligned}
\mu_{01} &= m_{01} - \frac{m_{01}}{m_{00}}m_{00} \\
&= 0 \\
&= \mu_{10}
\end{aligned}
\tag{7.85}
$$

Clearly, neither of the first-order centralized moments has any description capability since they are both zero. Going to higher order, one of the second-order moments, μ_{20}, is

$$
\begin{aligned}
\mu_{20} &= \sum_x \sum_y (x-\overline{x})^2 I(x,y)\Delta A \\
&= \sum_x \sum_y (x^2 - 2x\overline{x} + \overline{x}^2)I(x,y)\Delta A \\
&= \sum_x \sum_y x^2 I(x,y)\Delta A - 2\overline{x}\sum_x \sum_y xI(x,y)\Delta A + \overline{x}^2\sum_x \sum_y I(x,y)\Delta A
\end{aligned}
\tag{7.86}
$$

Since $m_{10} = \sum_x \sum_y xI(x,y)\Delta A$ and $\overline{x} = m_{10}/m_{00}$

$$
\begin{aligned}
\mu_{20} &= m_{20} - 2\frac{m_{10}}{m_{00}}m_{10} + \left(\frac{m_{10}}{m_{00}}\right)^2 m_{00} \\
&= m_{20} - \frac{m_{10}^2}{m_{00}}
\end{aligned}
\tag{7.87}
$$

and this has descriptive capability.

The use of moments to describe an ellipse is shown in Figure 7.22. Here, an original ellipse (Figure 7.22(a)) gives the second-order moments in Figure 7.22(d). In all cases, the first-order moments are zero, as expected. The moments (Figure 7.22(e)) of the translated ellipse (Figure 7.22(b)) are the same as those of the original ellipse. In fact, these moments show that the greatest rate of change in mass is around the horizontal axis, as consistent with the ellipse. The second-order moments (Figure 7.22(f)) of the ellipse when rotated by $90°$ (Figure 7.22(c)) are simply swapped around, as expected: the rate of change of mass is now greatest

(a) Original ellipse	(b) Translated ellipse	(c) Rotated ellipse
$\mu_{02} = 2.4947 \times 10^6$ $\mu_{20} = 6.4217 \times 10^5$	$\mu_{02} = 2.4947 \times 10^6$ $\mu_{20} = 6.4217 \times 10^5$	$\mu_{02} = 6.4217 \times 10^5$ $\mu_{20} = 2.4947 \times 10^6$
(d) Second-order centralized moments of original ellipse	(e) Second-order centralized moments of translated ellipse	(f) Second-order centralized moments of rotated ellipse

FIGURE 7.22

Describing a shape by centralized moments.

around the vertical axis. This illustrates how centralized moments are invariant to translation but not to rotation.

7.3.2.2 Invariant moments

Centralized moments are only **translation** invariant: they are constant only with change in position, and no other appearance transformation. In order to accrue invariance to scale and rotation, we require *normalized central moments*, η_{pq}, defined as (Hu, 1962)

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{7.88}$$

where

$$\gamma = \frac{p+q}{2} + 1 \quad \forall p + q \geq 2 \tag{7.89}$$

Seven *invariant moments* can be computed from these given by

$$
\begin{aligned}
M1 &= \eta_{20} + \eta_{02} \\
M2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
M3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
M4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
M5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
M6 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
M7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
&\quad + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})(3(\eta_{12} + \eta_{30})^2 - (\eta_{21} + \eta_{03})^2)
\end{aligned}
\tag{7.90}
$$

The first of these, $M1$ and $M2$, are second-order moments—those for which $p + q = 2$. Those remaining are third-order moments since $p + q = 3$. (The first-order moments are of no consequence since they are zero.) The last moment $M7$ is introduced as a skew invariant deigned to distinguish mirror images.

Code 7.5 shows the Mathcad implementation that computes the invariant moments $M1$, $M2$, and $M3$. The code computes the moments by straight implementation of Eqs (7.83) and (7.90). The use of these invariant moments to describe three shapes is shown in Figure 7.23. Figure 7.23(b) corresponds to the same plane in Figure 7.23(a) but with a change of scale and a rotation. Thus, the invariant moments for these two shapes are very similar. In contrast, the invariant moments for the plane in Figure 7.23(c) differ. These invariant moments have the most important invariance properties. However, these moments are not **orthogonal**, as such there is potential for **reducing** the size of the set of moments required to describe a shape accurately.

```
μ(p,q,shape):= | cmom← 0
               |
               | xc← ─────────── · ∑(from i=0 to rows(shape)-1) (shape_i)_0
               |      rows(shape)
               |
               | yc← ─────────── · ∑(from i=0 to rows(shape)-1) (shape_i)_1
               |      rows(shape)
               |
               | for s∈0..rows(shape)-1
               |
               |    cmom←cmom+[(shape_s)_0−xc]^p·[(shape_s)_1−yc]^q·(shape_s)_2
               |
               | cmom

              μ(p,q,im)
η(p,q,im):= ──────────────
             μ(0,0,im)^(p+q/2 +1)

M1(im):=η(2,0,im)+η(2,0,im)

M2(im):=(η(2,0,im)−η(0,2,im))^2+4·η(1,1,im)^2

M3(im):=(η(3,0,im)−3·η(1,2,im))^2+(3·η(2,1,im)−η(0,3,im))^2
```

CODE 7.5

Computing $M1$, $M2$, and $M3$.

7.3.2.3 Zernike moments

Invariance can be achieved by using *Zernike moments* (Teague, 1980) that give an orthogonal set of rotation-invariant moments. These find greater deployment where **invariant** properties are required. **Rotation** invariance is achieved by using polar representation, as opposed to the Cartesian parameterization for centralized moments. The **complex** Zernike moment, Z_{pq}, is

$$Z_{pq} = \frac{p+1}{\pi} \int_0^{2\pi} \int_0^\infty V_{pq}(r,\theta)^* f(r,\theta) r \, dr \, d\theta \qquad (7.91)$$

FIGURE 7.23

Describing a shape by invariant moments.

where p is now the radial magnitude, and q is the radial direction and where * again denotes the complex conjugate (as in Section 5.3.2) of a **Zernike polynomial**, V_{pq}, given by

$$V_{pq}(r, \theta) = R_{pq}(r)e^{jq\theta} \quad \text{where } p - q \text{ is \textbf{even} and } 0 \leq q \leq |p| \qquad (7.92)$$

R_{pq} is a real-valued polynomial given by

$$R_{pq}(r) = \sum_{m=0}^{\frac{p-|q|}{2}} (-1)^m \frac{(p-m)!}{m!\left(\frac{p+|q|}{2} - m\right)!\left(\frac{p-|q|}{2} - m\right)!} r^{p-2m} \qquad (7.93)$$

The order of the polynomial is denoted by p and the repetition by q. The repetition q can take negative values (since $q \leq |p|$), so the radial polynomial uses its magnitude and thus the inverse relationship holds: $R_{p,q}(r) = R_{p,-q}(r)$ (changing the notation of the polynomial slightly by introducing a comma to make clear the moment just has the sign of q inverted). The polynomials of lower degree are

$$\begin{aligned}
R_{00}(r) &= 1 \\
R_{11}(r) &= r \\
R_{22}(r) &= r^2 \\
R_{20}(r) &= r^2 - 1 \\
R_{31}(r) &= 3r^2 - 2r \\
R_{40}(r) &= 6r^4 - 6r^2 + 1
\end{aligned} \qquad (7.94)$$

and some of these are plotted in Figure 7.24. In Figure 7.24(a), we can see that the frequency components increase with the order p and the functions approach unity as $r \rightarrow 1$. The frequency content reflects the level of detail that can be
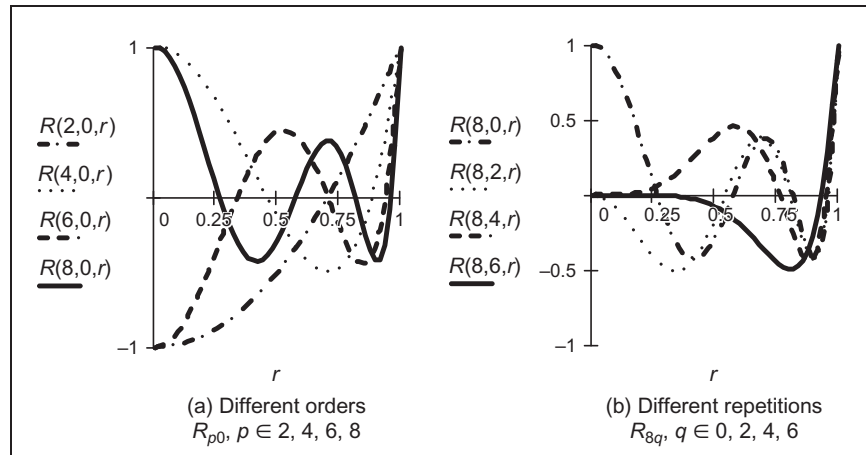
FIGURE 7.24

Zernike polynomials.
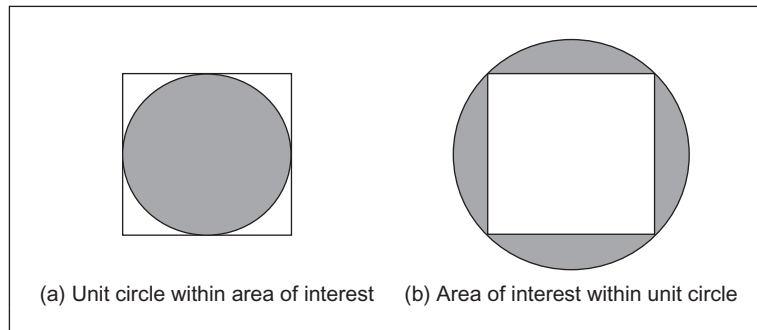
FIGURE 7.25

Mapping a unit circle to an area of interest.

captured by the particular polynomial. The change between the different polynomials shows how together they can capture different aspects of an underlying signal, across the various values of r. The repetition controls the way in which the function approaches unity: the influence along the polynomial and the polynomials for different values of q are shown in Figure 7.24(b).

These polynomials are orthogonal within the unit circle, so the analyzed shape (the area of interest) has to be remapped to be of this size before calculation of its moments. This naturally implies difficulty in mapping a unit circle to a Cartesian grid. As shown in Figure 7.25, (a) the circle can be within the area of interest, losing corner information (but that is information rarely of interest) or (b) around
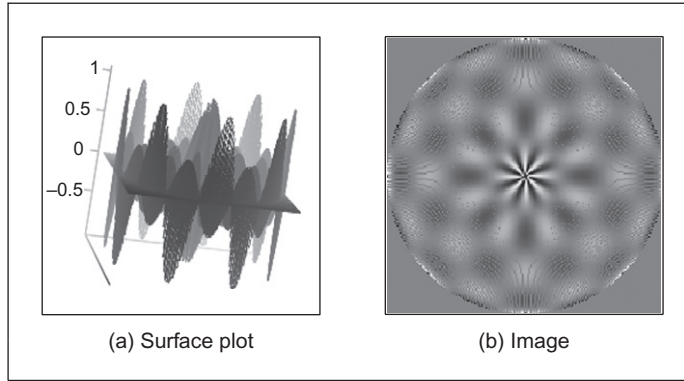
FIGURE 7.26

Zernike polynomial kernel.

(encompassing) the area of interest which then covers areas where there is no information but ensures that all the information within the area of interest is included.

The **orthogonality** of these polynomials assures the **reduction** in the set of numbers used to describe a shape. More simply, the radial polynomials can be expressed as

$$R_{pq}(r) = \sum_{k=q}^{p} B_{pqk} r^k \tag{7.95}$$

where the Zernike coefficients are

$$B_{pqk} = (-1)^{\frac{p-k}{2}} \frac{((p+k)/2)!}{((p-k)/2)!((k+q)/2)!((k-q)/2)!} \tag{7.96}$$

for $p - k =$ even. The Zernike moments can actually be calculated from centralized moments as

$$Z_{pq} = \frac{p+1}{\pi} \sum_{k=q}^{p} \sum_{l=0}^{t} \sum_{m=0}^{q} (-\mathrm{j})^m \binom{t}{l} \binom{q}{m} B_{pqk} \mu_{(k-2l-q+m)(q+2l-m)} \tag{7.97}$$

where $t = (k - q)/2$ and where

$$\binom{t}{l} = \frac{t!}{l!(t-l)!} \tag{7.98}$$

A Zernike polynomial kernel is shown in Figure 7.26. This shows that the kernel can capture differing levels of shape detail (and that multiple kernels are needed to give a shape's description). This kernel is computed in radial form, which is how it is deployed in shape analysis. Note that differing sets of parameters such as order

and repetition control the level of detail that is analyzed by application of this kernel to a shape. The plot shows the real part of the kernel; the imaginary part is similar but rotated.

Analysis (and by Eq. (7.83)), assuming that x and y are constrained to the interval $[-1,1]$, gives

$$Z_{00} = \frac{\mu_{00}}{\pi}$$

$$Z_{11} = \frac{2}{\pi}(\mu_{01} - j\mu_{10}) = 0 \tag{7.99}$$

$$Z_{22} = \frac{3}{\pi}(\mu_{02} - j2\mu_{11} - \mu_{20})$$

which can be extended further (Teague, 1980), and with remarkable similarity to the Hu invariant moments (Eq. (7.90)).

The magnitude of these Zernike moments remains invariant to rotation, which affects only the phase; the Zernike moments can be made **scale** invariant by normalization. An additional advantage is that there is a *reconstruction* theorem. For *Nm* moments, the original shape *f* can be reconstructed from its moments and the Zernike polynomials as

$$f(x,y) \approx \sum_{p=0}^{Nm} \sum_{q} Z_{pq} V_{pq}(x,y) \tag{7.100}$$

This is shown in Figure 7.27 for reconstructing a simple binary object, the letter A, from different numbers of moments. When reconstructing this up to the 10th order of a Zernike moment description (this requires 66 moments), we achieve a gray level image, which contains much of the overall shape, shown in Figure 7.27(b). This can be thresholded to give a binary image (Figure 7.27(e)) that shows the overall shape, without any corners. When we use more moments, we increase the detail in the reconstruction: Figure 7.27(c) is up to 15th order (136 moments) and Figure 7.27(d) is 20th order (231 moments). The latter of these is much closer to the original image, especially in its thresholded form (Figure 7.27(d)). This might sound like a lot of moments, but the compression from the original image is very high. Note also that even though we can achieve **recognition** from a smaller set of moments, these might not represent the hole in the shape that is not present at the 10th order, which just shows the overall shape of the letter A. As such, reconstruction can give insight as to the shape contribution of selected moments: their **significance** can be assessed by this and other tests.

These Zernike descriptors have been shown to good effect in application by reconstructing a good approximation to a shape with only few descriptors (Boyce and Hossack, 1983) and in recognition (Khotanzad and Hong, 1990). As ever, fast computation has been of (continuing) interest (Mukundan and Ramakrishnan, 1995; Gu et al., 2002).
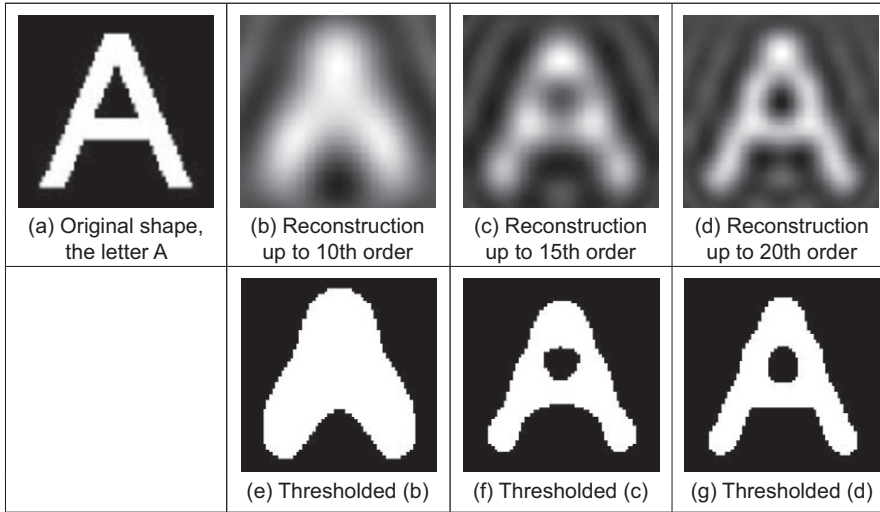
FIGURE 7.27

Reconstructing a shape from its moments (Prismall et al., 2002).

7.3.2.4 Other moments

There are *pseudo Zernike moments* (Teh and Chin, 1988) aimed to relieve the restriction on normalization to the unit circle. Also, there are *complex moments* (Abu-Mostafa and Psaltis, 1985), again aimed to provide a simpler moment description with invariance properties. In fact, since there is an infinite variety of functions that can be used as the basis function, we also have *Legendre* (Teague, 1980) and, more recently, *Tchebichef* (though this is sometimes spelt as *Chebyshev*) moments (Mukundan, 2001). There is no detailed comparison yet available, but there are advantages and disadvantages to the differing moments, often exposed by application.

Finally, there are *affine invariant moments* which do not change with position, rotation, and different scales along the coordinate axes, as a result, say, of a camera not being normal to the object plane. Here, the earliest approach appears to be by Flusser and Suk (1993). One of the reviews (Teh and Chin, 1988) concentrates on information content (redundancy), noise sensitivity, and on representation ability, comparing the performance of several of the more popular moments in these respects.

It is actually possible to explore the link between moments and Fourier theory (Mukundan and Ramakrishnan, 1998). The discrete Fourier transform of an image (Eq. (2.22)) can be written as

$$\mathbf{FP}_{u,v} = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathbf{P}_{x,y} \, e^{-j\frac{2\pi}{N}ux} \, e^{-j\frac{2\pi}{N}vy} \tag{7.101}$$

By using the Taylor expansion of the exponential function

$$\mathrm{e}^z = \sum_{p=0}^{\infty} \frac{z^p}{p!} \tag{7.102}$$

we can substitute for the exponential functions as

$$\mathbf{FP}_{u,v} = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathbf{P}_{x,y} \sum_{p=0}^{\infty} \frac{\left(-\mathrm{j}\frac{2\pi}{N}ux\right)^p}{p!} \sum_{q=0}^{\infty} \frac{\left(-\mathrm{j}\frac{2\pi}{N}vy\right)^q}{q!} \tag{7.103}$$

which by collecting terms gives

$$\mathbf{FP}_{u,v} = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} x^p y^q \mathbf{P}_{x,y} \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} \frac{\left(-\mathrm{j}\frac{2\pi}{N}\right)^{p+q}}{p!q!} u^p v^q \tag{7.104}$$

and by the definition of Cartesian moments (Eq. (7.74)), we have

$$\mathbf{FP}_{u,v} = \frac{1}{N} \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} \frac{\left(-\mathrm{j}\frac{2\pi}{N}\right)^{p+q}}{p!q!} u^p v^q m_{pq} \tag{7.105}$$

This implies that the Fourier transform of an image can be derived from its moments. There is then a link between the Fourier decomposition and that by moments, showing the link between the two. But we can go further since there is the inverse Fourier transform (Eq. (2.23))

$$\mathbf{P}_{x,y} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \mathbf{FP}_{u,v} \, \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}ux} \, \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}vy} \tag{7.106}$$

So the original image can be computed from the moments as

$$\mathbf{P}_{x,y} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}ux} \, \mathrm{e}^{\mathrm{j}\frac{2\pi}{N}vy} \frac{1}{N} \sum_{p=0}^{\infty} \sum_{q=0}^{\infty} \frac{\left(-\mathrm{j}\frac{2\pi}{N}\right)^{p+q}}{p!q!} u^p v^q m_{pq} \tag{7.107}$$

and this shows that we can get back to the image from our moment description, though care must be exercised in the choice of windows from which data are selected. This is *reconstruction*: we can reconstruct an image from its moment description. There has not been much study on reconstruction from moments, despite its apparent importance in understanding the potency of the description that has been achieved. Potency is usually investigated in application by determining the best set of moment features to maximize recognition capability (and we shall turn to this in Chapter 8). Essentially, reconstruction from basic geometric (Cartesian) moments is impractical (Teague, 1980), and the orthogonal bases functions such as the Zernike polynomials offer a simpler route to reconstruction, but these still require thresholding. More recently, Prismall et al. (2002) used (Zernike) moments for the reconstruction of moving objects.

7.4 **Further reading**

This chapter has essentially been based on unified techniques for border and region description. There is actually much more to contour and region analysis than indicated at the start of the chapter, for this is one of the start points of morphological analysis. There is an extensive review available (Loncaric, 1998) with many important references in this topic. The analysis neighborhood can be extended to be larger (Marchand and Sharaiha, 1997), and there is consideration of appropriate distance metrics for this (Das and Chatterji, 1988). A much more detailed study of boundary-based representation and application can be found in Otterloo's fine text (Van Otterloo, 1991). Naturally, there are many other ways to describe features, though few have the unique attributes of moments and Fourier descriptors. Naturally, there is an interrelation between boundary and region description: **curvature** can be computed from a chain code (Rosenfeld, 1974); Fourier descriptors can also be used to calculate region descriptions (Kiryati and Maydan, 1989). There have been many approaches to boundary approximation by fitting curves to the data. Some of these use polynomial approximation, or there are many *spline-based* techniques. A spline is a local function used to model a feature in sections. There are **quadratic** and **cubic** forms (for a good review of spline theory, try Ahlberg et al., 1967 or Dierckx, 1995); of interest, **snakes** are actually energy minimizing splines. There are many methods for polygonal approximations to curves, and recently, a new measure has been applied to compare performance on a suitable curve of techniques based on dominant point analysis (Rosin, 1997). To go with the earlier-mentioned review (Prokop and Reeves, 1992), there is a book available on moment theory (Mukundan and Ramakrishnan, 1998) showing the whole moment picture. As in the previous chapter, the skeleton of a shape can be used for recognition. This is a natural target for *thinning* techniques that have not been covered here. An excellent survey of these techniques, as used in character description following extraction, can be found in Trier et al. (1996)—describing use of moments and Fourier descriptors.

7.5 **References**

Abu-Mostafa, Y.S., Psaltis, D., 1985. Image normalisation by complex moments. IEEE Trans. PAMI 7, 46−55.

Aguado, A.S., Nixon, M.S., Montiel, E., 1998. Parameterising arbitrary shapes via Fourier descriptors for evidence-gathering extraction. CVIU: Comput. Vision Image Understand. 69 (2), 202−221.

Aguado, A.S., Montiel, E., Zaluska, E., 1999. Modelling generalised cylinders via Fourier morphing. ACM Trans. Graph. 18 (4), 293−315.

Ahlberg, J.H., Nilson, E.N., Walsh, J.L., 1967. The Theory of Splines and Their Applications. Academic Press, New York, NY.

Boyce, J.F., Hossack, W.J., 1983. Moment invariants for pattern recognition. Pattern Recog. Lett. 1, 451−456.

Chen, Y.Q., Nixon, M.S., Thomas, D.W., 1995. Texture classification using statistical geometric features. Pattern Recog. 28 (4), 537−552.

Cosgriff, R.L., 1960. Identification of Shape, Rep. 820-11, ASTIA AD 254792. Ohio State University Research Foundation, Columbus, OH.

Crimmins, T.R., 1982. A complete set of Fourier descriptors for two-dimensional shapes. IEEE Trans. Syst. Man Cybernetics 12 (6), 848−855.

Das, P.P., Chatterji, B.N., 1988. Knight's distances in digital geometry. Pattern Recog. Lett. 7, 215−226.

Dierckx, P., 1995. Curve and Surface Fitting with Splines. Oxford University Press, Oxford.

Flusser, J., Suk, T., 1993. Pattern-recognition by affine moment invariants. Pattern Recog. 26 (1), 167−174.

Freeman, H., 1961. On the encoding of arbitrary geometric configurations. IRE Trans. EC-10 (2), 260−268.

Freeman, H., 1974. Computer processing of line drawing images. Comput. Surv. 6 (1), 57−95.

Granlund, G.H., 1972. Fourier preprocessing for hand print character recognition. IEEE Trans. Comp. 21, 195−201.

Gu, J., Shua, H.Z., Toumoulinb, C., Luoa., L.M., 2002. A novel algorithm for fast computation of Zernike moments. Pattern Recog. 35 (12), 2905−2911.

Hu, M.K., 1962. Visual pattern recognition by moment invariants. IRE Trans. Inform. Theor. IT-8, 179−187.

Kashi, R.S., Bhoj-Kavde, P., Nowakowski, R.S., Papathomas, T.V., 1996. 2-D shape representation and averaging using normalised wavelet descriptors. Simulation 66 (3), 164−178.

Khotanzad, A., Hong, Y.H., 1990. Invariant image recognition by Zernike moments. IEEE Trans. PAMI 12, 489−498.

Kiryati, N., Maydan, D., 1989. Calculating geometric properties from Fourier representation. Pattern Recog. 22 (5), 469−475.

Kuhl, F.P., Giardina, C.R., 1982. Elliptic Fourier descriptors of a closed contour. CVGIP 18, 236−258.

Lin, C.C., Chellappa, R., 1987. Classification of partial 2D shapes using Fourier descriptors. IEEE Trans. PAMI 9 (5), 686−690.

Liu, H.C., Srinath, M.D., 1990. Corner detection from chain-coded curves. Pattern Recog. 23 (1), 51−68.

Loncaric, S., 1998. A survey of shape analysis techniques. Pattern Recog. 31 (8), 983−1001.

Marchand, S., Sharaiha, Y.M., 1997. Discrete convexity, straightness and the 16-neighbourhood. Comput. Vision Image Understand. 66 (3), 316−329.

Montiel, E., Aguado, A.S., Zaluska, E., 1996. Topology in fractals. Chaos, Solitons Fractals 7 (8), 1187−1207.

Montiel, E., Aguado, A.S., Zaluska, E., 1997. Fourier series expansion of irregular curves. Fractals 5 (1), 105−199.

Mukundan, R., 2001. Image analysis by Tchebichef moments. IEEE Trans. IP 10 (9), 1357−1364.

Mukundan, R., Ramakrishnan, K.R., 1995. Fast computation of Legendre and Zernike moments. Pattern Recog. 28 (9), 1433−1442.

Mukundan, R., Ramakrishnan, K.R., 1998. Moment Functions in Image Analysis: Theory and Applications. World Scientific, Singapore.

Van Otterloo, P.J., 1991. A Contour-Oriented Approach to Shape Analysis. Prentice Hall, Hertfordshire.

Persoon, E., Fu, K-S, 1977. Shape description using Fourier descriptors. IEEE Trans. SMC 3, 170−179.

Prismall, S.P., Nixon, M.S., Carter, J.N., 2002. On moving object reconstruction by moments. Proceedings of the BMVC, pp. 73−82.

Prokop, R.J., Reeves, A.P., 1992. A survey of moment-based techniques for unoccluded object representation and recognition. CVGIP: Graph. Models Image Process. 54 (5), 438−460.

Rosenfeld, A., 1974. Digital straight line segments. IEEE Trans. Comput. 23, 1264−1269.

Rosin, P., 1997. Techniques for assessing polygonal approximations to curves. IEEE Trans. PAMI 19 (6), 659−666.

Rosin, P., Zunic, J., 2005. Measuring rectilinearity. Comput. Vision Image Understand. 99 (2), 175−188.

Searle, N.H., 1970. Shape analysis by use of Walsh functions. In: Meltzer, B., Mitchie, D. (Eds.), Machine Intelligence, vol. 5. Edinburgh University Press.

Seeger, U., Seeger, R., 1994. Fast corner detection in gray-level images. Pattern Recog. Lett. 15, 669−675.

Staib, L., Duncan, J., 1992. Boundary finding with parametrically deformable models. IEEE Trans. PAMI 14, 1061−1075.

Teague, M.R., 1980. Image analysis by the general theory of moments. J. Opt. Soc. Am. 70, 920−930.

Teh, C.H., Chin, R.T., 1988. On image analysis by the method of moments. IEEE Trans. PAMI 10, 496−513.

Trier, O.D., Jain, A.K., Taxt, T., 1996. Feature extraction methods for character recognition—a survey. Pattern Recog. 29 (4), 641−662.

Undrill, P.E., Delibasis, K., Cameron, G.G., 1997. An application of genetic algorithms to geometric model-guided interpretation of brain anatomy. Pattern Recog. 30 (2), 217−227.

Zahn, C.T., Roskies, R.Z., 1972. Fourier descriptors for plane closed curves. IEEE Trans. Comput. C-21 (3), 269−281.