CHAPTER

High-level feature extraction: deformable shape analysis

6

CHAPTER OUTLINE HEAD

6.1 Overview

The previous chapter covered finding shapes by matching. This implies knowledge of a model (mathematical or template) of the target shape (feature). The shape is the **fixed** in that it is flexible only in terms of the parameters that define the shape or the parameters that define a template's appearance. Sometimes, however, it is not possible to model a shape with sufficient accuracy or to provide a template of the target as needed for the GHT. It might be that the exact shape is **unknown** or it might be that the perturbation of that shape is **impossible** to parameterize. In this case, we seek techniques that can **evolve** to the target solution or adapt their result to the data. This implies the use of flexible shape formulations. This chapter presents four techniques that can be used to find flexible

Feature Extraction & Image Processing for Computer Vision.

Table 6.1 Overview of Chapter 6

Main Topic	Subtopics	Main Points
Deformable templates	**Template matching** for **deformable** shapes. Defining a way to **analyze** the best match.	**Energy** maximization, computational considerations, optimization. **Parts-**based shape analysis.
Active contours and snakes	Finding shapes by **evolving contours**. **Discrete** and **continuous** formulations. **Operational** considerations and new active contour approaches.	Energy minimization for curve evolution. *Greedy* algorithm. *Kass snake*. Parameterization; initialization and performance. *Gradient vector field* and *level set* approaches.
Shape skeletonization	Notions of **distance**, **skeletons**, and **symmetry** and its measurement. Application of symmetry detection by **evidence gathering**. **Performance** factors.	*Distance transform* and shape skeleton; *medial axis transform*. *Discrete symmetry operator*. Accumulating evidence of symmetrical point arrangements. Performance: speed and noise.
Active shape models	Expressing shape variation by **statistics**. Capturing shape **variation** within feature extraction.	*Active shape model*. *Active appearance* model. *Principal components* analysis.

shapes in images. These are summarized in Table 6.1 and can be distinguished by the matching functional used to indicate the extent of match between image data and a shape. If the shape is flexible or **deformable**, so as to match the image data, we have a *deformable template*. This is where we shall start. Later, we shall move to techniques that are called *snakes*, because of their movement. We shall explain two different implementations of the snake model. The first one is based on discrete minimization and the second one on finite element analysis. We shall also look at determining a shape's skeleton, by *distance* analysis and by the *symmetry* of their appearance. This technique finds any symmetric shape by gathering evidence by considering features between pairs of points. Finally, we shall consider approaches that use the **statistics** of a shape's possible appearance to control selection of the final shape, called *active shape models (ASMs)*.

6.2 Deformable shape analysis

6.2.1 Deformable templates

One of the earlier approaches to deformable template analysis (Yuille, 1991) was aimed to find facial features for purposes of recognition. The approach considered

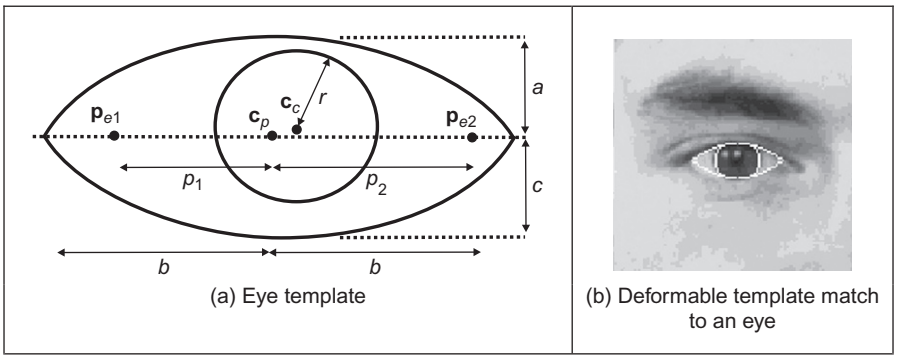


FIGURE 6.1

Finding an eye with a deformable template.

an eye to be comprised of an iris which sits within the sclera and which can be modeled as a combination of a circle that lies within a parabola. Clearly, the circle and a version of the parabola can be extracted by using Hough transform techniques, but this cannot be achieved in combination. When we combine the two shapes and allow them to change in size and orientation, while retaining their spatial relationship (that the iris or circle should reside within the sclera or parabola), then we have a deformable template.

The parabola is a shape described by a set of points (x,y) related by

$$y = a - \frac{a}{b^2}x^2 \tag{6.1}$$

where, as illustrated in Figure 6.1(a), a is the height of the parabola and b is its radius. As such, the maximum height is a and the minimum height is zero. A similar equation describes the lower parabola, in terms of b and c. The "center" of both parabolae is $\mathbf{c}_\mathrm{p}$. The circle is as defined earlier, with center coordinates $\mathbf{c}_\mathrm{c}$ and radius $\boldsymbol{r}$. We then seek values of the parameters which give a best match of this template to the image data. Clearly, one fit we would like to make concerns matching the edge data to that of the template, like in the Hough transform. The set of values for the parameters which give a template which matches the most edge points (since edge points are found at the boundaries of features) could then be deemed to be the best set of parameters describing the eye in an image. We then seek values of parameters that maximize

$$\{\mathbf{c}_\mathrm{p}, a, b, c, \mathbf{c}_\mathrm{c}, r\} = \max\left(\sum_{x,y \in \text{circle.perimeter,parabolae.perimeter}} \mathbf{E}_{x,y}\right) \tag{6.2}$$

Naturally, this would prefer the larger shape to the smaller ones, so we could divide the contribution of the circle and the parabolae by their perimeter to give an edge energy contribution E_e:

$$E_e = \sum_{x,y \in \text{circle.perimeter}} \mathbf{E}_{x,y} \Big/ \text{circle.perimeter} + \sum_{x,y \in \text{parabolae.perimeter}} \mathbf{E}_{x,y} \Big/ \text{parabolae.perimeter} \tag{6.3}$$

and we seek a combination of values for the parameters $\{\mathbf{c}_p, a, b, c, \mathbf{c}_c, r\}$ which maximize this energy. This however implies little knowledge of the **structure** of the eye. Since we know that the sclera is white (usually...) and the iris is darker than it, then we could build this information into the process. We can form an energy E_v functional for the circular region which averages the brightness over the circle area as

$$E_v = -\sum_{x,y \in \text{circle}} \mathbf{P}_{x,y} \Big/ \text{circle.area} \tag{6.4}$$

This is formed in the negative, since maximizing its value gives the best set of parameters. Similarly, we can form an energy functional for the light regions where the eye is white as E_p:

$$E_p = \sum_{x,y \in \text{parabolae} - \text{circle}} \mathbf{P}_{x,y} \Big/ \text{parabolae-circle.area} \tag{6.5}$$

where parabolae-circle implies points within the parabolae but not within the circle. We can then choose a set of parameters which maximize the combined energy functional formed by adding each energy when weighted by some chosen factors as

$$E = c_e \cdot E_e + c_v \cdot E_v + c_p \cdot E_p \tag{6.6}$$

where c_e, c_v, and c_p are the weighting factors. In this way, we are choosing values for the parameters which simultaneously maximize the chance that the edges of the circle and the perimeter coincide with the image edges, that the inside of the circle is dark and that the inside of the parabolae are light. The value chosen for each of the weighting factors controls the influence of that factor on the eventual result.

The energy fields are shown in Figure 6.2 when computed over the entire image. Naturally, the valley image shows up regions with low image intensity and the peak image shows regions of high image intensity, like the whites of the eyes. In its original formulation, this approach actually had five energy terms and the extra two are associated with the points $\mathbf{p}_{e1}$ and $\mathbf{p}_{e2}$ either side of the iris in Figure 6.1(a).

This is where the problem starts, as we now have eleven parameters (eight for the shapes and three for the weighting coefficients). We could of course simply

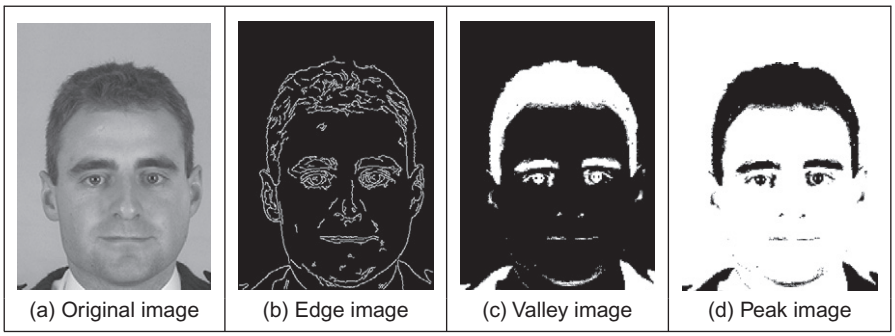


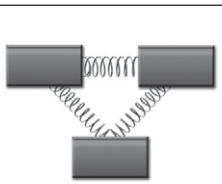
FIGURE 6.2

Energy fields over whole face image (Benn et al., 1999).

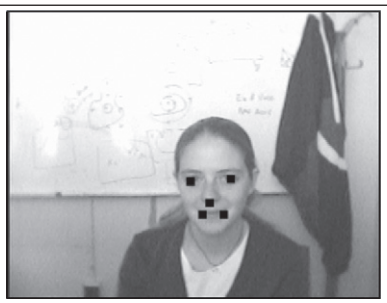
cycle through every possible value. Given, say, 100 possible values for each parameter, we then have to search 10^{22} combinations of parameters which would be no problem given multithread computers with terra hertz processing speed achieved via optical interconnect, but computers like that are not ready yet (on our budgets at least). Naturally, we can reduce the number of combinations by introducing constraints on the relative size and position of the shapes, e.g., the circle should lie wholly within the parabolae, but this will not reduce the number of combinations much. We can seek two alternatives: one is to use *optimization* techniques. The original approach (Yuille, 1991) favored the use of gradient descent techniques; currently, the *genetic algorithm* approach (Goldberg, 1988) seems to be most favored in many approaches which use optimization and this has been shown to good effect for deformable template eye extraction on a database of 1000 faces (Benn et al., 1999) (this is the source of the images shown here).

6.2.2 Parts-based shape analysis

A more recent class of approaches is called "*parts-based*" object analysis: rather than characterizing an object by a single feature (as in the previous chapter), objects are represented as a collection of parts arranged in a deformable structure. This follows an approach which predates the previous template approach (Fischler and Elschlager, 1973). Essentially, objects can be modeled as a network of **masses** which are connected by **springs**. This is illustrated in Figure 6.3(a) where, for a face, the two upper masses could represent the eyes and the lower mass could represent the face. The springs then constrain the mouth to be beneath and between the eyes. The springs add context to the position of the shape; the springs control the relationships between the objects and allow the object parts to move relative to one another. The extraction of the representation is then a compromise between the match of the features (the masses) to the image and the interrelationships (the springs) between the locations of the features. A result by a



(a) Mechanical equivalent



(b) Finding face features (Felzenszwalb and Huttenlocher, 2005)

FIGURE 6.3

Parts-based shape model.

later technique is shown in Figure 6.3(b) which shows that the three mass model of face features can be extended to one with five parts (in a star arrangement) and the image shows the best fit of this arrangement to an image containing a face.

Let us suggest that we have n parts (in Figure 6.3, $n = 3$) and $m_i(l_i)$ represents the difference from the image data when each feature $f_i(f_1, f_2,$ and f_3 in Figure 6.3) is placed at location l_i. The features can differ in relative position, and so a measure of the misplacement within the configuration (by how much the springs extend) can be a function $d_{ij}(l_i,l_j)$ which measures the degree of deformation when features f_i and f_j are placed at locations l_i and l_j, respectively. The best match L^* of the model to the image is then

$$L^* = \arg\left[\min\left(\sum_{i=1}^{N} m_i(l_i) + \sum_{f_i,f_j \in \Re} d_{ij}(l_i, l_j)\right)\right] \tag{6.7}$$

These components can be weighted; thus the optimization is the form of Eq. (6.6). The parameters thus derived are those which are the best compromise between the positions of the parts and the deformation. Determining these parameters is computationally very challenging, as it was for deformable templates. In the earliest approach, the optimization strategy was dynamic programming (the Viterbi algorithm). (It's fantastic that they even tried. In 1973, computers had the computational power of a modern doorbell and perhaps the same amount of memory—in the paper the resulting images are by character printing!) More recently, the minimization has been phrased as a statistical problem, and the solution requires structure to be imposed on the models, in order that an efficient solution is achieved (Felzenszwalb and Huttenlocher, 2005). In this way, machine learning approaches are used to learn—or train—from examples of the target

structures, and these models are then applied in an efficient manner by these methods. The method was demonstrated in its earliest forms capable of determining facial features in images, as shown in Figure 6.3(b) and of locating the human body by representing it as a set of interconnected parts.

An extension to the approach (Felzenszwalb et al., 2010) uses HoG (Section 5.4.2.2) at different scales to represent spatial models and again employs techniques from machine learning to improve the matching procedure. The approach was evaluated on the PASCAL VOC Challenge (Section 5.6) and clearly offers state-of-art performance on quite challenging datasets. The implementation of the approach is also available at the PASCAL site. Arguably, a model needs to be built before the technique can be applied, but that is central to any model-based approach (e.g., HoG or GHT). As computers' speeds increase, training on large sets of data will clearly improve too.

An alternative to deformable models- and parts-based analysis is to seek a different technique that uses fewer parameters. This is where we move to **snakes** that are a much more popular approach. These snakes evolve a set of **points** (a contour) to match the image data, rather than evolving a shape.

6.3 Active contours (snakes)

6.3.1 Basics

Active contours or *snakes* (Kass et al., 1988) are a completely different approach to feature extraction. An active contour is a set of points which aims to enclose a target feature, the feature to be extracted. It is a bit like using a balloon to "find" a shape: the balloon is placed outside the shape, enclosing it. Then by taking air out of the balloon, making it smaller, the shape is found when the balloon stops shrinking, when it fits the target shape. By this manner, active contours arrange a set of points so as to describe a target feature, by enclosing it. Snakes are actually quite recent compared with many computer vision techniques and their original formulation was as an interactive extraction process, though they are now usually deployed for automatic feature extraction.

An initial contour is placed outside the target feature and is then evolved so as to enclose it. The process is illustrated in Figure 6.4 where the target feature is the perimeter of the iris. First, an initial contour is placed outside the iris (Figure 6.4(a)). The contour is then minimized to find a new contour which shrinks so as to be closer to the iris (Figure 6.4(b)). After seven iterations, the contour points can be seen to match the iris perimeter well (Figure 6.4(d)).

Active contours are actually expressed as an **energy minimization** process. The target feature is a minimum of a suitably formulated energy functional. This energy functional includes more than just edge information: it includes properties that control the way the contour can stretch and curve. In this way, a snake represents a **compromise** between its **own** properties (like its ability to bend

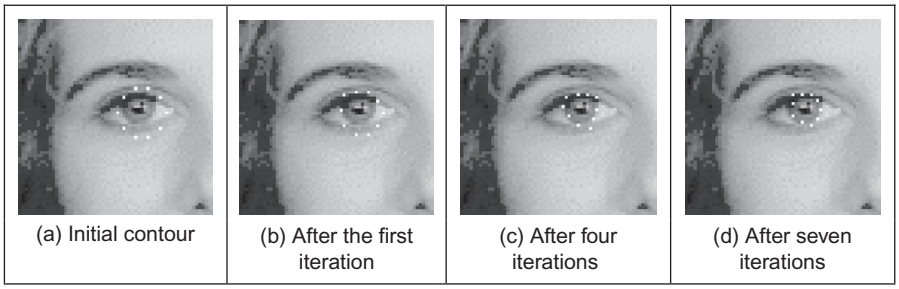


FIGURE 6.4

Using a snake to find an eye's iris.

and stretch) and **image** properties (like the edge magnitude). Accordingly, the energy functional is the addition of a function of the contour's internal energy, its constraint energy, and the image energy: these are denoted E_{int}, E_{con}, and E_{image}, respectively. These are functions of the set of points which make up a snake, $\mathbf{v}(s)$, which is the set of x and y coordinates of the points in the snake. The energy functional is the integral of these functions of the snake, given $S \in [0,1)$ is the normalized length around the snake. The energy functional E_{snake} is then

$$E_{\text{snake}} = \int_{s=0}^{1} E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s)) + E_{\text{con}}(\mathbf{v}(s))\text{d}s \tag{6.8}$$

In this equation, the internal energy, E_{int}, controls the natural behavior of the snake and hence the arrangement of the snake points; the image energy, E_{image}, attracts the snake to chosen low-level features (such as **edge** points); and the constraint energy, E_{con}, allows higher level information to control the snake's evolution. The aim of the snake is to evolve by **minimizing** Eq. (6.8). New snake contours are those with lower energy and are a better match to the target feature (according to the values of E_{int}, E_{image}, and E_{con}) than the original set of points from which the active contour has evolved. In this manner, we seek to choose a set of points $\mathbf{v}(s)$ such that

$$\frac{\text{d}E_{\text{snake}}}{\text{d}\mathbf{v}(s)} = 0 \tag{6.9}$$

This can of course select a maximum rather than a minimum, and a second-order derivative can be used to discriminate between a maximum and a minimum. However, this is not usually necessary as a minimum is usually the only stable solution (on reaching a maximum, it would then be likely to pass over the top to minimize the energy). Prior to investigating how we can minimize Eq. (6.8), let us first consider the parameters which can control a snake's behavior.

The energy functionals are expressed in terms of functions of the snake and of the image. These functions contribute to the snake energy according to values

chosen for respective weighting coefficients. In this manner, the internal image energy is defined to be a weighted summation of first- and second-order derivatives around the contour:

$$E_{\text{int}} = \alpha(s)\left|\frac{\mathrm{d}\mathbf{v}(s)}{\mathrm{d}s}\right|^2 + \beta(s)\left|\frac{\mathrm{d}^2\mathbf{v}(s)}{\mathrm{d}s^2}\right|^2 \tag{6.10}$$

The first-order differential, $\mathrm{d}\mathbf{v}(s)/\mathrm{d}s$, measures the energy due to **stretching** which is the **elastic** energy since high values of this differential imply a high rate of change in that region of the contour. The second-order differential, $\mathrm{d}^2\mathbf{v}(s)/\mathrm{d}s^2$, measures the energy due to **bending**, the **curvature** energy. The first-order differential is weighted by $\alpha(s)$ which controls the contribution of the elastic energy due to point spacing; the second-order differential is weighted by $\beta(s)$ which controls the contribution of the curvature energy due to point variation. Choice of the values of α and β controls the shape the snake aims to attain. Low values for α imply the points can change in spacing greatly, whereas higher values imply that the snake aims to attain evenly spaced contour points. Low values for β imply that curvature is not minimized and the contour can form corners in its perimeter, whereas high values predispose the snake to smooth contours. These are the properties of the contour itself, which is just part of a snake's compromise between its own properties and measured features in an image.

The image energy attracts the snake to low-level features, such as brightness or edge data, aiming to select those with least contribution. The original formulation suggested that lines, edges, and terminations could contribute to the energy function. Their energy is denoted E_{line}, E_{edge}, and E_{term}, respectively, and are controlled by weighting coefficients w_{line}, w_{edge}, and w_{term}, respectively. The image energy is then

$$E_{\text{image}} = w_{\text{line}}E_{\text{line}} + w_{\text{edge}}E_{\text{edge}} + w_{\text{term}}E_{\text{term}} \tag{6.11}$$

The **line** energy can be set to the image intensity at a particular point. If black has a lower value than white, then the snake will be extracted to dark features. Altering the sign of w_{line} will attract the snake to brighter features. The **edge** energy can be that computed by application of an edge detection operator, the magnitude, say, of the output of the Sobel edge detection operator. The **termination** energy, E_{term}, as measured by Eq. (4.52), can include the curvature of level image contours (as opposed to the curvature of the snake, controlled by $\beta(s)$), but this is rarely used. It is most common to use the edge energy, though the line energy can find application.

6.3.2 The Greedy algorithm for snakes

The implementation of a snake, to evolve a set of points to minimize Eq. (6.8), can use finite elements, or finite differences, which is complicated and follows later. It is easier to start with the *Greedy algorithm* (Williams and Shah, 1992)

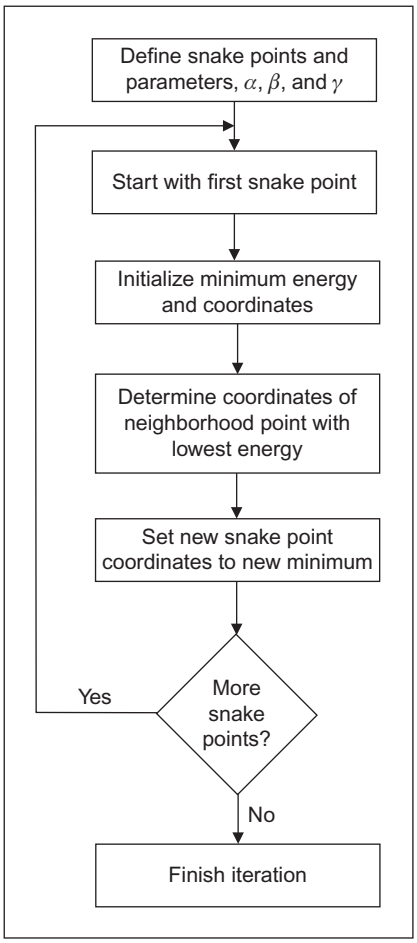


FIGURE 6.5

Operation of the Greedy algorithm.

which implements the energy minimization process as a purely discrete algorithm, illustrated in Figure 6.5. The process starts by specifying an initial contour. Earlier, Figure 6.4(a) used a circle of 16 points along the perimeter of a circle. Alternatively, these can be specified manually. The Greedy algorithm then evolves the snake in an iterative manner by local neighborhood search around contour points to select new ones which have lower snake energy. The process is called Greedy by virtue of the way the search propagates around the contour. At each iteration, all contour points are evolved and the process is actually repeated for the first contour point. The index to snake points is computed modulo S (the number of snake points).

For a set of snake points $\mathbf{v}_s$, $\forall s \in 0, S-1$, the energy functional minimized for each snake point is

$$E_{\text{snake}}(s) = E_{\text{int}}(\mathbf{v}_s) + E_{\text{image}}(\mathbf{v}_s) \tag{6.12}$$

This is expressed as

$$E_{\text{snake}}(s) = \alpha(s)\left|\frac{d\mathbf{v}_s}{ds}\right|^2 + \beta(s)\left|\frac{d^2\mathbf{v}_s}{ds^2}\right|^2 + \gamma(s)E_{\text{edge}} \tag{6.13}$$

where the first- and second-order differentials are approximated for each point searched in the local neighborhood of the currently selected contour point. The weighting parameters, α, β, and γ are all functions of the contour. Accordingly, each contour point has associated values for α, β, and γ. An implementation of the specification of an initial contour by a function `point` is given in Code 6.1. In this implementation, the contour is stored as a matrix of vectors. Each vector has five elements: two are the x and y coordinates of the contour point, the remaining three parameters are the values of α, β, and γ for that contour point, set here to be 0.5, 0.5, and 1.0, respectively. The `no` contour points are arranged to be in a circle, radius `rad`, and center (`xc`,`yc`). As such, a vector is returned for each snake point, $\texttt{point}_s$, where $(\texttt{point}_s)_0$, $(\texttt{point}_s)_1$, $(\texttt{point}_s)_2$, $(\texttt{point}_s)_3$, $(\texttt{point}_s)_4$ are the x coordinate, the y coordinate, and α, β, and γ for the particular snake point s: $\mathbf{x}_s$, $\mathbf{y}_s$, α_s, β_s, and γ_s, respectively.

```
points(rad,no,xc,yc):= | for s∈0..no-1
                       |   x_s ← xc+floor(rad·cos((s·2·π)/no)+0.5)
                       |   y_s ← yc+floor(rad·sin((s·2·π)/no)+0.5)
                       |   α_s ← 0.5
                       |   β_s ← 0.5
                       |   γ_s ← 1
                       |   point_s ← [x_s, y_s, α_s, β_s, γ_s]
                       | point
```

CODE 6.1

Specifying an initial contour.

The first-order differential is approximated as the modulus of the difference between the average spacing of contour points (evaluated as the Euclidean distance between them), and the Euclidean distance between the currently selected image point $\mathbf{v}_s$ and the next contour point. By selection of an appropriate value of $\alpha(s)$ for each contour point $\mathbf{v}_s$, this can control the spacing between the contour points:

$$\left|\frac{d\mathbf{v}_s}{ds}\right|^2 = \left|\sum_{i=0}^{S-1}\|\mathbf{v}_i - \mathbf{v}_{i+1}\|/S - \|\mathbf{v}_s - \mathbf{v}_{s+1}\|\right| = \left|\sum_{i=0}^{S-1}\sqrt{(\mathbf{x}_i - \mathbf{x}_{i+1})^2 + (\mathbf{y}_i - \mathbf{y}_{i+1})^2}/S - \sqrt{(\mathbf{x}_s - \mathbf{x}_{s+1})^2 + (\mathbf{y}_s - \mathbf{y}_{s+1})^2}\right| \tag{6.14}$$

as evaluated from the x and the y coordinates of the adjacent snake point ($\mathbf{x}_{s+1}$, $\mathbf{y}_{s+1}$) and the coordinates of the point currently inspected ($\mathbf{x}_s$,$\mathbf{y}_s$). Clearly, the first-order differential, as evaluated from Eq. (6.14), drops to zero when the contour is evenly spaced, as required. This is implemented by the function `Econt` in Code 6.2 which uses a function `dist` to evaluate the average spacing and a function `dist2` to evaluate the Euclidean distance between the currently searched point ($\mathbf{v}_s$) and the next contour point ($\mathbf{v}_{s+1}$). The arguments to E_{cont} are the x and y coordinates of the point currently being inspected, `x` and `y`, the index of the contour point currently under consideration, `s`, and the contour itself, `cont`.

```
dist(s,contour):= | s1←mod(s,rows(contour))
                  | s2←mod(s + 1,rows(contour))
                  | √[(contour_s1)_0 - (contour_s2)_0]^2 + [(contour_s1)_1 - (contour_s2)_1]^2

dist2(x,y,s,contour):= | s2←mod(s + 1,rows(contour))
                       | √[(contour_s2)_0 - x]^2 + [(contour_s2)_1 - y]^2

Econt(x,y,s,cont):= | D← 1/rows(cont) · Σ_{s1=0}^{rows(cont)-1} dist(s1, cont)
                    | |D - dist2(x,y,s,cont)|
```

CODE 6.2

Evaluating the contour energy.

The second-order differential can be implemented as an estimate of the curvature between the next and previous contour points, $\mathbf{v}_{s+1}$ and $\mathbf{v}_{s-1}$,

respectively, and the point in the local neighborhood of the currently inspected snake point $\mathbf{v}_s$:

$$\left|\frac{d^2\mathbf{v}_s}{ds^2}\right|^2 = |(\mathbf{v}_{s+1} - 2\mathbf{v}_s + \mathbf{v}_{s-1})|^2 = (\mathbf{x}_{s+1} - 2\mathbf{x}_s + \mathbf{x}_{s-1})^2 + (\mathbf{y}_{s+1} - 2\mathbf{y}_s + \mathbf{y}_{s-1})^2 \tag{6.15}$$

This is implemented by a function `Ecur` in Code 6.3, whose arguments again are the x and y coordinates of the point currently being inspected, `x` and `y`, the index of the contour point currently under consideration, `s`, and the contour itself, `con`.

```
Ecur(x,y,s,con) := | s1←mod(s-1+rows(con),rows(con))
                   | s3←mod(s+1,rows(con))
                   | [(con_s1)_0 - 2·x + (con_s3)_0]^2 + [(con_s1)_1 - 2·y + (con_s3)_1]^2
```

CODE 6.3

Evaluating the contour curvature.

E_{edge} can be implemented as the magnitude of the Sobel edge operator at point x,y. This is normalized to ensure that its value lies between zero and unity. This is also performed for the elastic and curvature energies in the current region of interest. This is achieved by normalization using Eq. (3.2) arranged to provide an output ranging between 0 and 1. The edge image could also be normalized within the current window of interest, but this makes it more possible that the result is influenced by noise. Since the snake is arranged to be a minimization process, the edge image is inverted so that the points with highest edge strength are given the lowest edge value (0), whereas the areas where the image is constant are given a high value (1). Accordingly, the snake will be attracted to the edge points with greatest magnitude. The normalization process ensures that the contour energy and curvature and the edge strength are balanced forces and ease appropriate selection of values for α, β, and γ. This is achieved by a balancing function (`balance`) that normalizes the contour and curvature energy within the window of interest.

The Greedy algorithm then uses these energy functionals to minimize the composite energy functional, Eq. (6.13), given in the function `grdy` in Code 6.4. This gives a single iteration in the evolution of a contour wherein all snake points are searched. The energy for each snake point is first determined and is stored as the point with minimum energy. This ensures that if any other point is found to have equally small energy, then the contour point will remain in the same position. Then, the local 3×3 neighborhood is searched to determine whether any other point has a lower energy than the current contour point. If it does, that point is returned as the new contour point.

```
grdy(edg,con) := |for s1∈0..rows(con)
                 |   s←mod(s1,rows(con))
                 |   xmin←(con_s)_0
                 |   ymin←(con_s)_1
                 |   forces←balance[(con_s)_0,(con_s)_1,edg,s,con]
                 |   Emin←(con_s)_2.Econt(xmin,ymin,s,con)
                 |   Emin←Emin+(con_s)_3·Ecur(xmin,ymin,s,con)
                 |   Emin←Emin+(con_s)_4·(edg_0)_((con_s)_1,(con_s)_0)
                 |   for x∈(con_s)_0-1..(con_s)_0+1
                 |     for y∈(con_s)_1-1..(con_s)_1+1
                 |       if check(x,y,edg_0)
                 |           |xx←x-(con_s)_0+1
                 |           |yy←y-(con_s)_1+1
                 |           |Ej←(con_s)_2·(forces_(0,0))_(yy,xx)
                 |           |Ej←Ej+(con_s)_3·(forces_(0,1))_(yy,xx)
                 |           |Ej←Ej+(con_s)_4·(edg_0)_(y,x)
                 |           |if Ej<Emin
                 |           |   |Emin←Ej
                 |           |   |xmin←x
                 |           |   |ymin←y
                 |   con_s←[ xmin      ]
                 |         [ ymin      ]
                 |         [ (con_s)_2 ]
                 |         [ (con_s)_3 ]
                 |         [ (con_s)_4 ]
                 |con
```

CODE 6.4

The Greedy algorithm.

A verbatim implementation of the Greedy algorithm would include three thresholds. One is a threshold on tangential direction and another on edge magnitude. If an edge point were adjudged to be of direction above the chosen threshold, and with magnitude above its corresponding threshold, then β can be set to zero for that point to allow corners to form. This has not been included in Code 6.4, in part because there is mutual dependence between α and β. Also, the original presentation of the Greedy algorithm proposed to continue evolving the snake until it becomes static, when the number of contour points moved in a single iteration is below the third threshold value. This can lead to instability since it can lead to a situation where contour points merely oscillate between two solutions and the process would appear not to converge. Again, this has not been implemented here.

The effect of varying α and β is shown in Figures 6.6 and 6.7. Setting α to zero removes influence of **spacing** on the contour points' arrangement. In this manner, the points will become unevenly spaced (Figure 6.6(b)) and

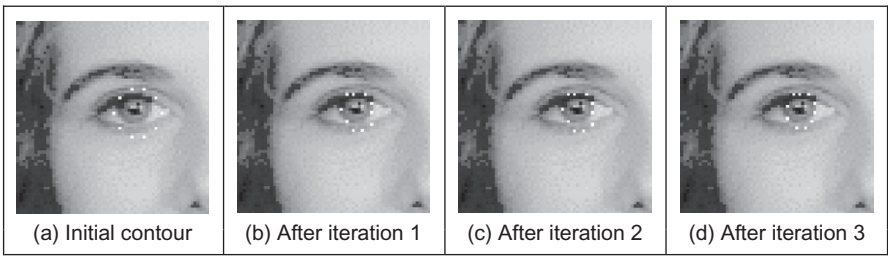


FIGURE 6.6

Effect of removing control by spacing.

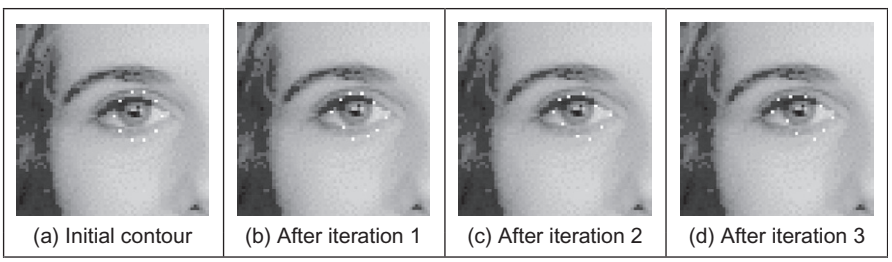


FIGURE 6.7

Effect of removing low curvature control.

eventually can be placed on top of each other. Reducing the control by spacing can be desirable for features that have high localized curvature. Low values of α can allow for bunching of points in such regions, giving a better feature description.

Setting β to zero removes influence of **curvature** on the contour points' arrangement, allowing corners to form in the contour, as illustrated in Figure 6.7. This is manifested in the first iteration (Figure 6.7(b)) and since with β set to zero for the whole contour, each contour point can become a corner with high curvature (Figure 6.7(c)) leading to the rather ridiculous result in Figure 6.7(d). Reducing the control by curvature can clearly be desirable for features that have high localized curvature. This illustrates the mutual dependence between α and β, since low values of α can accompany low values of β in regions of high localized curvature. Setting γ to zero would force the snake to ignore image data and evolve under its own forces. This would be rather farcical. The influence of γ is reduced in applications where the image data used is known to be noisy. Note that one fundamental problem with a discrete version is that the final solution can oscillate when it swaps between two sets of points which are both with equally low energy. This can be prevented by detecting the occurrence of oscillation.

A further difficulty is that as the contour becomes smaller, the number of contour points actually constrains the result as they cannot be compressed into too small a space. The only solution to this is to resample the contour.

6.3.3 Complete (Kass) snake implementation

The Greedy method iterates around the snake to find local minimum energy at snake points. This is an **approximation**, since it does not necessarily determine the "best" local minimum in the region of the snake points, by virtue of iteration. A *complete snake implementation*, or *Kass snake*, solves for all snake points in one step to ensure that the snake moves to the best local energy minimum. We seek to choose snake points ($\mathbf{v}(s) = (\mathbf{x}(s), \mathbf{y}(s))$) in such a manner that the energy is minimized, Eq. (6.9). Calculus of variations shows how the solution to Eq. (6.8) reduces to a pair of differential equations that can be solved by finite difference analysis (Waite and Welsh, 1990). This results in a set of equations that iteratively provide new sets of contour points. By calculus of variation, we shall consider an admissible solution $\hat{\mathbf{v}}(s)$ perturbed by a small amount, $\varepsilon\,\delta\mathbf{v}(s)$, which achieves minimum energy, as

$$\frac{dE_{\text{snake}}(\hat{\mathbf{v}}(s) + \varepsilon\,\delta\mathbf{v}(s))}{d\varepsilon} = 0 \tag{6.16}$$

where the perturbation is spatial, affecting the x and y coordinates of a snake point:

$$\delta\mathbf{v}(s) = (\delta_x(s), \delta_y(s)) \tag{6.17}$$

This gives the perturbed snake solution as

$$\hat{\mathbf{v}}(s) + \varepsilon\,\delta\mathbf{v}(s) = (\hat{x}(s) + \varepsilon\,\delta_x(s), \hat{y}(s) + \varepsilon\,\delta_y(s)) \tag{6.18}$$

where $\hat{x}(s)$ and $\hat{y}(s)$ are the x and y coordinates, respectively, of the snake points at the solution ($\hat{\mathbf{v}}(s) = (\hat{x}(s), \hat{y}(s))$). By setting the constraint energy E_{con} to zero, the snake energy, Eq. (6.8), becomes

$$E_{\text{snake}}(\mathbf{v}(s)) = \int_{s=0}^{1} \{E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s))\}ds \tag{6.19}$$

Edge magnitude information is often used (so that snakes are attracted to edges found by an edge-detection operator), so we shall replace E_{image} by E_{edge}. By substitution for the perturbed snake points, we obtain

$$E_{\text{snake}}(\hat{\mathbf{v}}(s) + \varepsilon\,\delta\mathbf{v}(s)) = \int_{s=0}^{1} \{E_{\text{int}}(\hat{\mathbf{v}}(s) + \varepsilon\,\delta\mathbf{v}(s)) + E_{\text{edge}}(\hat{\mathbf{v}}(s) + \varepsilon\,\delta\mathbf{v}(s))\}ds \tag{6.20}$$

By substituting from Eq. (6.10), we obtain

$$E_{\text{snake}}(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathbf{v}(s))= \int_{s=0}^{s=1}\left\{\alpha(s)\left|\frac{\mathrm{d}(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathbf{v}(s))}{\mathrm{d}s}\right|^2+\beta(s)\left|\frac{\mathrm{d}^2(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathbf{v}(s))}{\mathrm{d}s^2}\right|^2+E_{\text{edge}}(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathbf{v}(s))\right\}\mathrm{d}s \tag{6.21}$$

By substituting from Eq. (6.18),

$$E_{\text{snake}}(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathbf{v}(s))= \int_{s=0}^{s=1}\left\{\begin{array}{l}\alpha(s)\left\{\begin{array}{l}\left(\dfrac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\right)^2+2\varepsilon\dfrac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\dfrac{\mathrm{d}\delta_x(s)}{\mathrm{d}s}+\left(\varepsilon\dfrac{\mathrm{d}\delta_x(s)}{\mathrm{d}s}\right)^2\\ \quad+\left(\dfrac{\mathrm{d}\hat{y}(s)}{\mathrm{d}s}\right)^2+2\varepsilon\dfrac{\mathrm{d}\hat{y}(s)}{\mathrm{d}s}\dfrac{\mathrm{d}\delta_y(s)}{\mathrm{d}s}+\left(\varepsilon\dfrac{\mathrm{d}\delta_y(s)}{\mathrm{d}s}\right)^2\end{array}\right\}\\ +\beta(s)\left\{\begin{array}{l}\left(\dfrac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\right)^2+2\varepsilon\dfrac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\dfrac{\mathrm{d}^2\delta_x(s)}{\mathrm{d}s^2}+\left(\varepsilon\dfrac{\mathrm{d}^2\delta_x(s)}{\mathrm{d}s^2}\right)^2\\ \quad+\left(\dfrac{\mathrm{d}^2\hat{y}(s)}{\mathrm{d}s^2}\right)^2+2\varepsilon\dfrac{\mathrm{d}^2\hat{y}(s)}{\mathrm{d}s^2}\dfrac{\mathrm{d}^2\delta_y(s)}{\mathrm{d}s^2}+\left(\varepsilon\dfrac{\mathrm{d}^2\delta_y(s)}{\mathrm{d}s^2}\right)^2\\ \quad+E_{\text{edge}}(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathbf{v}(s))\end{array}\right\}\end{array}\right\}\mathrm{d}s \tag{6.22}$$

By expanding E_{edge} at the perturbed solution by Taylor series, we obtain

$$\begin{aligned}E_{\text{edge}}(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathrm{v}(s))&=E_{\text{edge}}(\hat{x}(s)+\varepsilon\,\delta_x(s),\hat{y}(s)+\varepsilon\,\delta_y(s))\\&=E_{\text{edge}}(\hat{x}(s),\hat{y}(s))+\varepsilon\delta_x(s)\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{\hat{x},\hat{y}}+\varepsilon\delta_y(s)\frac{\partial E_{\text{edge}}}{\partial y}\bigg|_{\hat{x},\hat{y}}+O(\varepsilon^2)\end{aligned} \tag{6.23}$$

This implies that the image information must be twice differentiable which holds for edge information, but not for some other forms of image energy. Ignoring higher-order terms in ε (since ε is small) by reformulation, Eq. (6.22) becomes

$$\begin{aligned}E_{\text{snake}}(\hat{\mathbf{v}}(s)+\varepsilon\,\delta\mathbf{v}(s))&=E_{\text{snake}}(\hat{\mathbf{v}}(s))\\&+2\varepsilon\int_{s=0}^{s=1}\alpha(s)\frac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\frac{\mathrm{d}\delta_x(s)}{\mathrm{d}s}+\beta(s)\frac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\frac{\mathrm{d}^2\delta_x(s)}{\mathrm{d}s^2}+\frac{\delta_x(s)}{2}\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{\hat{x},\hat{y}}\mathrm{d}s\\&+2\varepsilon\int_{s=0}^{s=1}\alpha(s)\frac{\mathrm{d}\hat{y}(s)}{\mathrm{d}s}\frac{\mathrm{d}\delta_y(s)}{\mathrm{d}s}+\beta(s)\frac{\mathrm{d}^2\hat{y}(s)}{\mathrm{d}s^2}\frac{\mathrm{d}^2\delta_y(s)}{\mathrm{d}s^2}+\frac{\delta_y(s)}{2}\frac{\partial E_{\text{edge}}}{\partial y}\bigg|_{\hat{x},\hat{y}}\mathrm{d}s\end{aligned} \tag{6.24}$$

Since the perturbed solution is at a minimum, the integration terms in Eq. (6.24) must be identically zero:

$$\int_{s=0}^{s=1} \alpha(s)\frac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\frac{\mathrm{d}\delta_x(s)}{\mathrm{d}s} + \beta(s)\frac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\frac{\mathrm{d}^2\delta_x(s)}{\mathrm{d}s^2} + \frac{\delta_x(s)}{2}\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{\hat{x},\hat{y}} \mathrm{d}s = 0 \tag{6.25}$$

$$\int_{s=0}^{s=1} \alpha(s)\frac{\mathrm{d}\hat{y}(s)}{\mathrm{d}s}\frac{\mathrm{d}\delta_y(s)}{\mathrm{d}s} + \beta(s)\frac{\mathrm{d}^2\hat{y}(s)}{\mathrm{d}s^2}\frac{\mathrm{d}^2\delta_y(s)}{\mathrm{d}s^2} + \frac{\delta_y(s)}{2}\frac{\partial E_{\text{edge}}}{\partial y}\bigg|_{\hat{x},\hat{y}} \mathrm{d}s = 0 \tag{6.26}$$

By integration we obtain

$$\begin{aligned}
&\left[\alpha(s)\frac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\delta_x(s)\right]_{s=0}^{1} - \int_{s=0}^{s=1}\frac{\mathrm{d}}{\mathrm{d}s}\left\{\alpha(s)\frac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\right\}\delta_x(s)\mathrm{d}s \\
&\left[\beta(s)\frac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\frac{\mathrm{d}\delta_x(s)}{\mathrm{d}s}\right]_{s=0}^{1} - \left[\frac{\mathrm{d}}{\mathrm{d}s}\left\{\beta(s)\frac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\right\}\delta_x(s)\right]_{s=0}^{1} \\
&+ \int_{s=0}^{s=1}\frac{\mathrm{d}^2}{\mathrm{d}s^2}\left\{\beta(s)\frac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\right\}\delta_x(s)\mathrm{d}s + \frac{1}{2}\int_{s=0}^{1}\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{\hat{x},\hat{y}}\delta_x(s)\mathrm{d}s = 0
\end{aligned} \tag{6.27}$$

Since the first, third, and fourth terms are zero (since for a closed contour, $\delta_x(1) - \delta_x(0) = 0$ and $\delta_y(1) - \delta_y(0) = 0$), this reduces to

$$\int_{s=0}^{s=1}\left\{-\frac{\mathrm{d}}{\mathrm{d}s}\left\{\alpha(s)\frac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\right\} + \frac{\mathrm{d}^2}{\mathrm{d}s^2}\left\{\beta(s)\frac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\right\} + \frac{1}{2}\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{\hat{x},\hat{y}}\right\}\delta_x(s)\mathrm{d}s = 0 \tag{6.28}$$

Since this equation holds for all $\delta_x(s)$, then

$$-\frac{\mathrm{d}}{\mathrm{d}s}\left\{\alpha(s)\frac{\mathrm{d}\hat{x}(s)}{\mathrm{d}s}\right\} + \frac{\mathrm{d}^2}{\mathrm{d}s^2}\left\{\beta(s)\frac{\mathrm{d}^2\hat{x}(s)}{\mathrm{d}s^2}\right\} + \frac{1}{2}\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{\hat{x},\hat{y}} = 0 \tag{6.29}$$

Similarly, by a similar development of Eq. (6.26), we obtain

$$-\frac{\mathrm{d}}{\mathrm{d}s}\left\{\alpha(s)\frac{\mathrm{d}\hat{y}(s)}{\mathrm{d}s}\right\} + \frac{\mathrm{d}^2}{\mathrm{d}s^2}\left\{\beta(s)\frac{\mathrm{d}^2\hat{y}(s)}{\mathrm{d}s^2}\right\} + \frac{1}{2}\frac{\partial E_{\text{edge}}}{\partial y}\bigg|_{\hat{x},\hat{y}} = 0 \tag{6.30}$$

This has reformulated the original energy minimization framework, Eq. (6.8), into a pair of differential equations. To implement a complete snake, we seek the solution to Eqs (6.29) and (6.30). By the method of finite differences, we substitute for $\mathrm{d}\mathbf{x}(s)/\mathrm{d}s \cong \mathbf{x}_{s+1} - \mathbf{x}_s$, the first-order difference, and the second-order difference is $\mathrm{d}^2\mathbf{x}(s)/\mathrm{d}s^2 \cong \mathbf{x}_{s+1} - 2\mathbf{x}_s + \mathbf{x}_{s-1}$ (as in Eq. (6.13)), which by substitution into Eq. (6.29), for a contour discretized into S points equally spaced by an arc

length h (remembering that the indices $s \in [1,S]$ to snake points are computed modulo S), gives

$$-\frac{1}{h}\left\{\alpha_{s+1}\frac{(\mathbf{x}_{s+1}-\mathbf{x}_s)}{h}-\alpha_s\frac{(\mathbf{x}_s-\mathbf{x}_{s-1})}{h}\right\}$$
$$+\frac{1}{h^2}\left\{\beta_{s+1}\frac{(\mathbf{x}_{s+2}-2\mathbf{x}_{s+1}+\mathbf{x}_s)}{h^2}-2\beta_s\frac{(\mathbf{x}_{s+1}-2\mathbf{x}_s+\mathbf{x}_{s-1})}{h^2}\right.$$
$$\left.+\beta_{s-1}\frac{(\mathbf{x}_s-2\mathbf{x}_{s-1}+\mathbf{x}_{s-2})}{h^2}\right\}+\frac{1}{2}\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{\mathrm{x}_s,\mathrm{y}_s}=0 \tag{6.31}$$

By collecting the coefficients of different points, Eq. (6.31) can be expressed as

$$f_s = a_s\mathbf{x}_{s-2}+b_s\mathbf{x}_{s-1}+c_s\mathbf{x}_s+d_s\mathbf{x}_{s+1}+e_s\mathbf{x}_{s+2} \tag{6.32}$$

where

$$f_s = -\frac{1}{2}\frac{\partial E_{\text{edge}}}{\partial x}\bigg|_{x_s,y_s},\quad a_s=\frac{\beta_{s-1}}{h^4},\quad b_s=-\frac{2(\beta_s+\beta_{s-1})}{h^4}-\frac{\alpha_s}{h^2}$$
$$c_s=\frac{\beta_{s+1}+4\beta_s+\beta_{s-1}}{h^4}+\frac{\alpha_{s+1}+\alpha_s}{h^2},\quad d_s=-\frac{2(\beta_{s+1}+\beta_s)}{h^4}-\frac{\alpha_{s+1}}{h^2},\quad e_s=\frac{\beta_{s+1}}{h^4}$$

This is now in the form of a linear (matrix) equation:

$$\mathbf{Ax} = fx(\mathbf{x},\mathbf{y}) \tag{6.33}$$

where $fx(\mathbf{x},\mathbf{y})$ is the first-order differential of the edge magnitude along the x axis, where

$$\mathbf{A}=\begin{bmatrix} c_1 & d_1 & e_1 & 0 & \cdots & a_1 & b_1 \\ b_2 & c_2 & d_2 & e_2 & 0 & \cdots & a_2 \\ a_3 & b_3 & c_3 & d_3 & e_3 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \\ e_{S-1} & 0 & \cdots & a_{S-1} & b_{S-1} & c_{S-1} & d_{S-1} \\ d_S & e_S & 0 & \cdots & a_S & b_S & c_S \end{bmatrix}$$

Similarly, by analysis of Eq. (6.30), we obtain

$$\mathbf{Ay} = fy(\mathbf{x},\mathbf{y}) \tag{6.34}$$

where $fy(\mathbf{x},\mathbf{y})$ is the first-order difference of the edge magnitude along the y axis. These equations can be solved iteratively to provide a new vector $\mathbf{v}^{<i+1>}$ from an initial vector $\mathbf{v}^{<i>}$, where i is an evolution index. The iterative solution is

$$\frac{(\mathbf{x}^{<i+1>} - \mathbf{x}^{<i>})}{\Delta} + \mathbf{A}\mathbf{x}^{<i+1>} = fx(\mathbf{x}^{<i>}, \mathbf{y}^{<i>}) \tag{6.35}$$

where the control factor Δ is a scalar chosen to control convergence. The control factor, Δ, actually controls the rate of evolution of the snake: large values make the snake move quickly, small values make for slow movement. As usual, fast movement implies that the snake can pass over features of interest without noticing them, whereas slow movement can be rather tedious. So the appropriate choice for Δ is again a compromise, this time between selectivity and time. The formulation for the vector of y coordinates is

$$\frac{(\mathbf{y}^{<i+1>} - \mathbf{y}^{<i>})}{\Delta} + \mathbf{A}\mathbf{y}^{<i+1>} = fy(\mathbf{x}^{<i>}, \mathbf{y}^{<i>}) \tag{6.36}$$

By rearrangement, this gives the final pair of equations that can be used to iteratively evolve a contour; the complete snake solution is then

$$\mathbf{x}^{<i+1>} = \left(\mathbf{A} + \frac{1}{\Delta}\mathbf{I}\right)^{-1}\left(\frac{1}{\Delta}\mathbf{x}^{<i>} + fx(\mathbf{x}^{<i>}, \mathbf{y}^{<i>})\right) \tag{6.37}$$

where $\mathbf{I}$ is the identity matrix. This implies that the new set of x coordinates is a weighted sum of the initial set of contour points and the image information. The fraction is calculated according to specified snake properties, the values chosen for α and β. For the y coordinates, we have

$$\mathbf{y}^{<i+1>} = \left(\mathbf{A} + \frac{1}{\Delta}\mathbf{I}\right)^{-1}\left(\frac{1}{\Delta}\mathbf{y}^{<i>} + fy(\mathbf{x}^{<i>}, \mathbf{y}^{<i>})\right) \tag{6.38}$$

The new set of contour points then becomes the starting set for the **next** iteration. Note that this is a continuous formulation, as opposed to the discrete (Greedy) implementation. One **penalty** is the need for matrix inversion, affecting speed. Clearly, the benefits are that coordinates are calculated as **real** functions and the **complete** set of new contour points is provided at each iteration. The result of implementing the complete solution is illustrated in Figure 6.8. The initialization Figure 6.8(a) is the same as for the Greedy algorithm, but with 32 contour points. At the first iteration (Figure 6.8(b)), the contour begins to shrink and move toward the eye's iris. By the sixth iteration (Figure 6.8(c)), some of the contour points have snagged on strong edge data, particularly in the upper part of the contour. At this point, however, the excessive curvature becomes inadmissible, and the contour releases these points to achieve a smooth contour again, one which is better matched to the edge data and the chosen snake features. Finally,

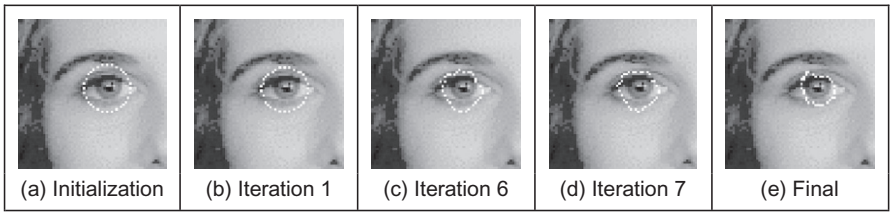


FIGURE 6.8

Illustrating the evolution of a complete snake.

Figure 6.8(e) is where the contour ceases to move. Part of the contour has been snagged on strong edge data in the eyebrow, whereas the remainder of the contour matches the chosen feature well.

Clearly, a different solution could be obtained by using different values for the snake parameters; in application the choice of values for α, β, and Δ must be made very carefully. In fact, this is part of the difficulty in using snakes for practical feature extraction; a further difficulty is that the result depends on where the initial contour is placed. These difficulties are called **parameterization** and **initialization**, respectively. These problems have motivated much research and development.

6.3.4 Other snake approaches

There are many further considerations to implementing snakes and there is a great wealth of material. One consideration is that we have only considered **closed** contours. There are, naturally, *open contours*. These require slight difference in formulation for the Kass snake (Waite and Welsh, 1990) and only minor modification for implementation in the Greedy algorithm. One difficulty with the Greedy algorithm is its sensitivity to noise due to its local neighborhood action. Also, the Greedy algorithm can end up in an oscillatory position where the final contour simply jumps between two equally attractive energy minima. One solution (Lai and Chin, 1994) resolved this difficulty by increase in the size of the snake neighborhood, but this incurs much greater complexity. In order to allow snakes to **expand**, as opposed to contracting, a *normal force* can be included which inflates a snake and pushes it over unattractive features (Cohen, 1991; Cohen and Cohen, 1993). The force is implemented by the addition of

$$F_{\text{normal}} = \rho \mathbf{n}(s) \tag{6.39}$$

to the evolution equation, where $\mathbf{n}(s)$ is the normal force and ρ weights its effect. This is inherently sensitive to the magnitude of the normal force that, if too large, can force the contour to pass over features of interest. Another way to allow

expansion is to modify the elasticity constraint (Berger, 1991) so that the internal energy becomes

$$E_{\text{int}} = \alpha(s)\left(\left|\frac{\mathrm{d}\mathbf{v}(s)}{\mathrm{d}s}\right|^2 - (L+\varepsilon)\right)^2 + \beta(s)\left|\frac{\mathrm{d}^2\mathbf{v}(s)}{\mathrm{d}s^2}\right|^2 \tag{6.40}$$

where the length adjustment ε when positive, $\varepsilon > 0$, and added to the contour length L causes the contour to expand. When negative, $\varepsilon < 0$, this causes the length to reduce and so the contour contracts. To avoid imbalance due to the contraction force, the technique can be modified to remove it (by changing the continuity and curvature constraints) without losing the controlling properties of the internal forces (Xu et al., 1994) (and which, incidentally, allowed corners to form in the snake). This gives a contour no prejudice to expansion or contraction as required. The technique allowed for integration of prior shape knowledge; methods have also been developed to allow **local shape** to influence contour evolution (Berger, 1991; Williams and Shah, 1992).

Some snake approaches have included factors that attract contours to regions using statistical models (Ronfard, 1994) or texture (Ivins and Porrill, 1995), to complement operators that combine edge detection with region growing. Also, the snake model can be generalized to higher dimensions and there are 3D snake **surfaces** (Cohen et al., 1992; Wang and Wang, 1992). Finally, a new approach has introduced shapes for moving objects, by including velocity (Peterfreund, 1999).

6.3.5 Further snake developments

Snakes have been formulated not only to include local shape but also phrased in terms of *regularization* (Lai and Chin, 1995) where a single parameter controls snake evolution, emphasizing a snake's natural compromise between its own forces and the image forces. Regularization involves using a single parameter to control the balance between the external and the internal forces. Given a regularization parameter λ, the snake energy of equation (6.37) can be given as

$$E_{\text{snake}}(\mathbf{v}(s)) = \int_{s=0}^{1} \{\lambda E_{\text{int}}(\mathbf{v}(s)) + (1-\lambda)E_{\text{image}}(\mathbf{v}(s))\}\mathrm{d}s \tag{6.41}$$

Clearly, if $\lambda = 1$, then the snake will use the internal energy only, whereas if $\lambda = 0$, the snake will be attracted to the selected image function only. Usually, regularization concerns selecting a value in between zero and one guided, say, by knowledge of the likely confidence in the edge information. In fact, Lai's approach calculates the regularization parameter at contour points as

$$\lambda_i = \frac{\sigma_\eta^2}{\sigma_i^2 + \sigma_\eta^2} \tag{6.42}$$

where σ_i^2 appears to be the variance of the point i and σ_η^2 is the variance of the noise at the point (even digging into Lai's PhD thesis provided no explicit clues here, save that "these parameters may be learned from training samples"—if this is impossible a procedure can be invoked). As before, λ_i lies between zero and one, and where the variances are bounded as

$$1/\sigma_i^2 + 1/\sigma_\eta^2 = 1 \tag{6.43}$$

This does actually link these **generalized** active contour models to an approach we shall meet later, where the target shape is extracted conditional upon its expected variation. Lai's approach also addressed **initialization** and showed how a GHT could be used to initialize an active contour and built into the extraction process. A major development of new external force model, which is called the *gradient vector flow* (GVF) (Xu and Prince, 1998). The GVF is computed as a diffusion of the gradient vectors of an edge map. There is however natural limitation on using a single contour for extraction, since it is never known precisely where to stop.

In fact, many of the problems with initialization with active contours can be resolved by using a **dual contour** approach (Gunn and Nixon, 1997) that also includes local shape and regularization. This approach aims to enclose the target shape within an inner and an outer contour. The outer contour **contracts** while the inner contour **expands**. A balance is struck between the two contours to allow them to allow the target shape to be extracted. Gunn showed how shapes could be extracted successfully, even when the target contour was far from the two initial contours. Further, the technique was shown to provide better immunity to initialization, in comparison with the results of a Kass snake and Xu's approach.

Later, the dual approach was extended to a discrete space (Gunn and Nixon, 1998), using an established search algorithm. The search algorithm used dynamic programming which has already been used within active contours to find a global solution (Lai and Chin, 1995) and in matching and tracking contours (Geiger et al., 1995). This new approach has already been used within an enormous study (using a database of over 20,000 images no less) on automated cell segmentation for cervical cancer screening (Bamford and Lovell, 1998), achieving more than 99% accurate segmentation. The approach is formulated as a discrete search using a dual contour approach, illustrated in Figure 6.9. The inner and the outer contours aim to be inside and outside the target shape, respectively. The space between the inner and the outer contours is divided into lines (like the spokes on the wheel of a bicycle) and M points are taken along each of the N lines. We then have a grid of $M \times N$ points, in which the target contour (shape) is expected to lie. The full lattice of points is shown in Figure 6.10(a). Should we need higher resolution, then we can choose large values of M and of N, but this in turn implies more computational effort. One can envisage strategies which allow for linearization of the coverage of the space in between the two contours, but these can make implementation much more complex.

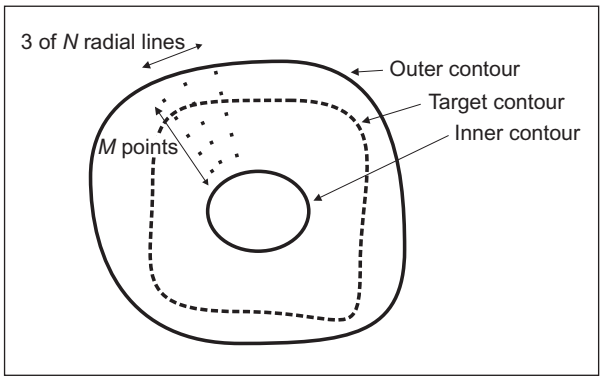


FIGURE 6.9

Discrete dual contour search.

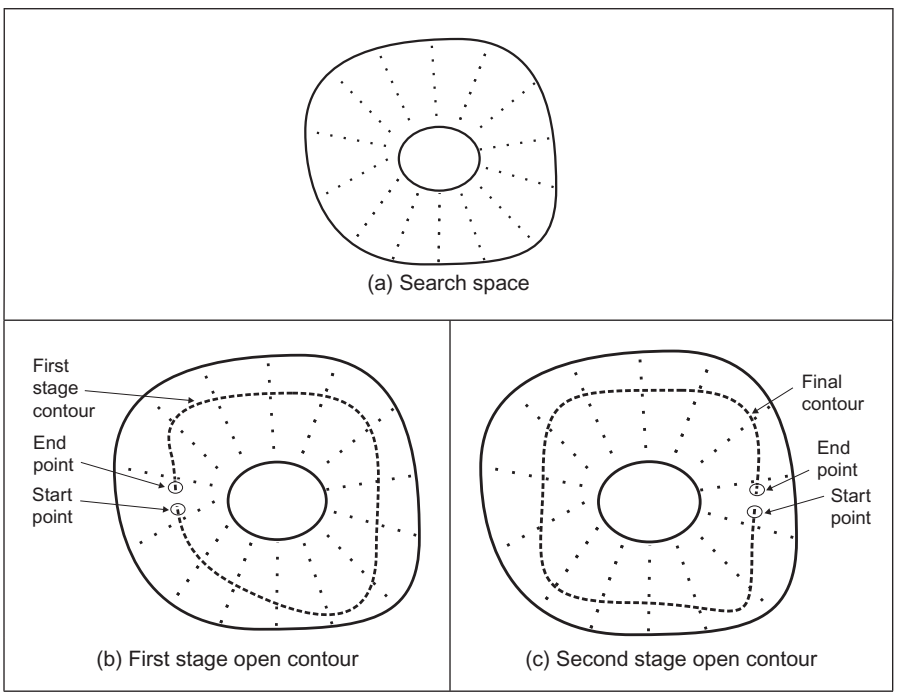


FIGURE 6.10

Discrete dual contour point space.

The approach again uses **regularization**, where the snake energy is a discrete form to Eq. (6.41), so the energy at a snake point (unlike earlier formulations, e.g., Eq. (6.12)) is

$$E(\mathbf{v}_i) = \lambda E_{\text{int}}(\mathbf{v}_i) + (1 - \lambda) E_{\text{ext}}(\mathbf{v}_i) \tag{6.44}$$

where the internal energy is formulated as

$$E_{\text{int}}(\mathbf{v}_i) = \left(\frac{|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}|}{|\mathbf{v}_{i+1} - \mathbf{v}_{i-1}|} \right)^2 \tag{6.45}$$

The numerator expresses the curvature, seen earlier in the Greedy formulation. It is scaled by a factor that ensures the contour is *scale invariant* with no **prejudice** as to the **size** of the contour. If there is no prejudice, the contour will be attracted to smooth contours, given appropriate choice of the regularization parameter. As such, the formulation is simply a more sophisticated version of the Greedy algorithm, dispensing with several factors of limited value (such as the need to choose values for three weighting parameters: one only now need be chosen; the elasticity constraint has also been removed, and that is perhaps more debatable). The interest here is that the search for the optimal contour is constrained to be between two contours, as in Figure 6.9. By way of a snake's formulation, we seek the contour with minimum energy. When this is applied to a contour which is bounded, then we seek a minimum cost path. This is a natural target for the well-known Viterbi (dynamic programming) algorithm (for its application in vision, see, for example, Geiger et al., 1995). This is designed precisely to do this: to find a minimum cost path within specified bounds. In order to formulate it by dynamic programming, we seek a cost function to be minimized. We formulate a cost function C between one snake element and the next as

$$C_i(\mathbf{v}_{i+1}, \mathbf{v}_i) = \min[C_{i-1}(\mathbf{v}_i, \mathbf{v}_{i-1}) + \lambda E_{\text{int}}(\mathbf{v}_i) + (1 - \lambda) E_{\text{ext}}(\mathbf{v}_i)] \tag{6.46}$$

In this way, we should be able to choose a path through a set of snake that minimizes the total energy, formed by the compromise between internal and external energy at that point, together with the path that led to the point. As such, we will need to store the energies at points within the matrix, which corresponds directly to the earlier tessellation. We also require a position matrix to store for each stage (i) the position ($\mathbf{v}_{i-1}$) that minimizes the cost function at that stage ($C_i(\mathbf{v}_{i+1},\mathbf{v}_i)$). This also needs initialization to set the first point, $C_1(\mathbf{v}_1,\mathbf{v}_0) = 0$. Given a **closed** contour (one which is completely joined together) then for an arbitrary start point, we separate optimization routine to determine the best starting and end points for the contour. The full search space is illustrated in Figure 6.10(a). Ideally, this should be searched for a closed contour, the target contour of Figure 6.9. It is computationally less demanding to consider an **open** contour, where the ends do not join. We can approximate a closed contour by considering it to be an open contour in **two** stages. In the first stage (Figure 6.10(b)) the midpoints of the two lines at the start and end are taken as the starting

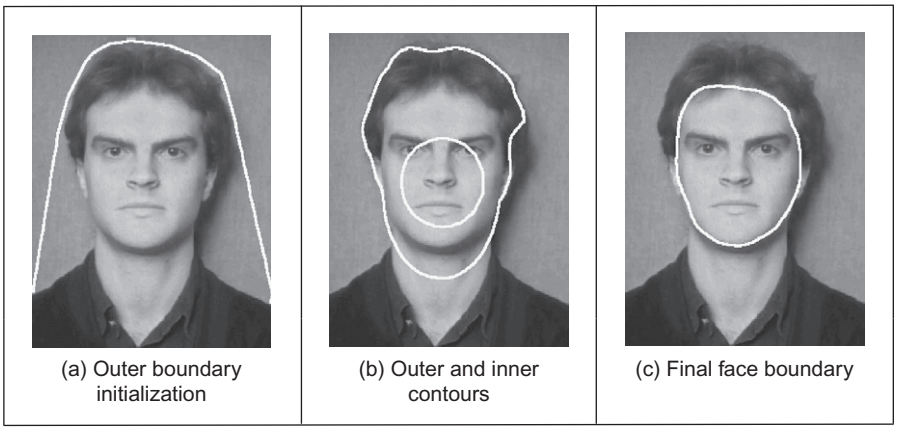


FIGURE 6.11

Extracting the face outline by a discrete dual contour.

conditions. In the second stage (Figure 6.10(c)) the points determined by dynamic programming halfway round the contour (i.e., for two lines at $N/2$) are taken as the start and the end points for a new open-contour dynamic programming search, which then optimizes the contour from these points. The premise is that the points halfway round the contour will be at, or close to, their optimal position after the first stage and it is the points at, or near, the starting points in the first stage that require refinement. This reduces the computational requirement by a factor of M^2.

The technique was originally demonstrated to extract the face boundary, for feature extraction within automatic face recognition, as illustrated in Figure 6.11. The outer boundary (Figure 6.11(a)) was extracted using a convex hull which in turn initialized an inner and an outer contour (Figure 6.11(b)). The final extraction by the dual discrete contour is the boundary of facial skin (Figure 6.11(c)). The number of points in the mesh naturally limits the accuracy with which the final contour is extracted, but application could naturally be followed by use of a continuous Kass snake to improve final resolution. In fact, it was shown that human faces could be discriminated by the contour extracted by this technique, though the study highlighted potential difficulty with facial organs and illumination. As mentioned earlier, it was later deployed in cell analysis where the inner and the outer contours were derived by the analysis of the stained-cell image.

6.3.6 Geometric active contours (level-set-based approaches)

Problems discussed so far with active contours include initialization and poor convergence to concave regions. Also, parametric active contours (the snakes discussed earlier) can have difficulty in segmenting multiple objects simultaneously because of the explicit representation of curve. *Geometric active contour* (GAC)

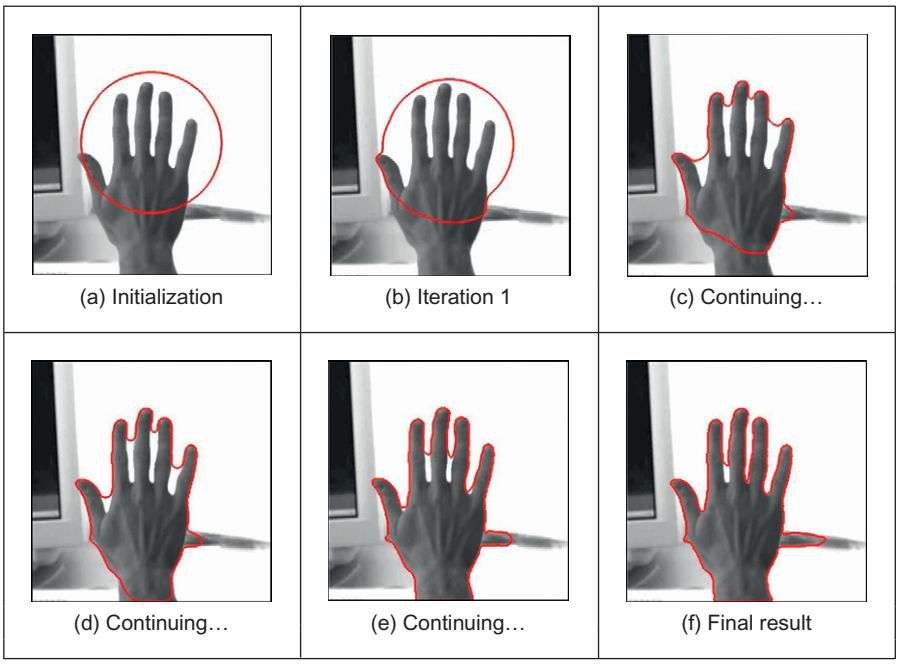


FIGURE 6.12

Extraction by curve evolution (a diffusion snake) (Cremers et al., 2002).

models have been introduced to solve this problem, where the curve is represented implicitly in a *level set function*. Essentially, the main argument is that by changing the representation, we can improve the result, and there have indeed been some very impressive results presented. Consider for example the result in Figure 6.12 where we are extracting the boundary of the hand, by using the initialization shown in Figure 6.12(a). This would be hard to achieve by the active contour models discussed so far: there are concavities, sharp corners, and background contamination which it is difficult for parametric techniques to handle. It is not perfect, but it is clearly much better (there are techniques to improve on this result, but this is far enough for the moment). On the other hand, there are no panaceas in engineering, and we should not expect them to exist. The new techniques can be found to be complex to implement, even to understand, though by virtue of their impressive results there are new approaches aimed to speed application and to ease implementation. As yet, the techniques do not find routine deployment (certainly not in real-time applications), but this is part of the evolution of any technique. The complexity and scope of this book mandates a short description of these new approaches here, but as usual we shall provide pointers to more in-depth source material.

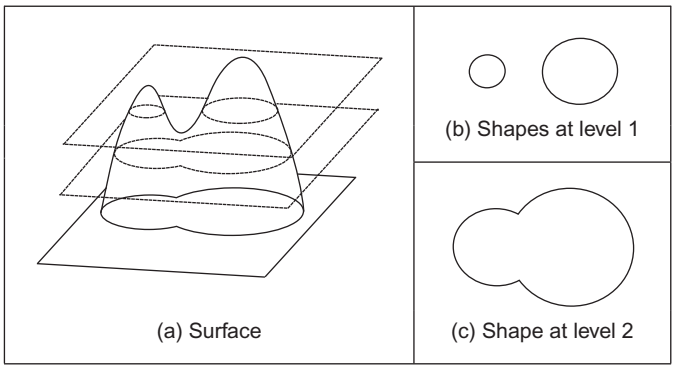


FIGURE 6.13

Surfaces and level sets.

Level set methods (Osher and Sethian, 1988) essentially find the shape without parameterizing it, so the curve description is **implicit** rather than explicit, by finding it as the zero level set of a function (Sethian, 1999; Osher and Paragios, 2003). The zero level set is the interface between two regions in an image. This can be visualized as taking slices through a surface shown in Figure 6.13(a). As we take slices at different levels (as the surface evolves) the shape can split (Figure 6.13(b)). This would be difficult to parameterize (we would have to detect when it splits), but it can be handled within a level set approach by considering the underlying surface. At a lower level (Figure 6.13(c)) we have a single composite shape. As such, we have an extraction which evolves with time (to change the level). The initialization is a closed curve and we shall formulate how we want the curve to move in a way analogous to minimizing its energy.

The level set function is the signed distance to the contour. This distance is arranged to be negative inside the contour and positive outside it. The contour itself, the target shape, is where the distance is zero—the interface between the two regions. Accordingly, we store values for each pixel representing this distance. We then determine new values for this surface, say by expansion. As we evolve the surface, the level sets evolve accordingly, equivalent to moving the surface where the slices are taken, as shown in Figure 6.13. Since the distance map needs renormalization after each iteration, it can make the technique slow in operation (or need a fast computer).

Let us assume that the interface C is controlled to change in a constant manner and evolves with time t by propagating along its normal direction with speed F (where F is a function of, say, curvature (Eq. (4.61)) and speed) according to

$$\frac{\partial C}{\partial t} = F \cdot \frac{\nabla \phi}{|\nabla \phi|} \tag{6.47}$$

Here, the term $\frac{\nabla\phi}{|\nabla\phi|}$ is a vector pointing in the direction normal to the surface—previously discussed in Section 4.4.1, Eq. (4.53). (The curvature at a point is measured perpendicular to the level set function at that point.) The curve is then evolving in a normal direction, controlled by the curvature. At all times, the interface C is the zero level set

$$\phi(C(t),t)=0 \tag{6.48}$$

The level set function ϕ is positive outside of the region and negative when it is inside and it is zero on the boundary of the shape. As such, by differentiation we get

$$\frac{\partial\phi(C(t),t)}{\partial t}=0 \tag{6.49}$$

and by the chain rule we obtain

$$\frac{\partial\phi}{\partial C}\frac{\partial C}{\partial t}+\frac{\partial\phi}{\partial t}=0 \tag{6.50}$$

By rearranging and substituting from Eq. (6.47), we obtain

$$\frac{\partial\phi}{\partial t}=-F\frac{\partial\phi}{\partial C}\cdot\frac{\nabla\phi}{|\nabla\phi|}=-F|\nabla\phi| \tag{6.51}$$

which suggests that the propagation of a curve depends on its gradient. The analysis is actually a bit more complex since F is a scalar and C is a vector in (x,y) we have that

$$\begin{aligned}\frac{\partial\phi}{\partial t}&=-\frac{\partial C}{\partial t}F\cdot\frac{\nabla\phi}{|\nabla\phi|}=-F\frac{\partial C}{\partial t}\cdot\frac{\nabla\phi}{|\nabla\phi|}\\&=-F(\phi_x,\phi_y)\cdot\frac{(\phi_x,\phi_y)}{\sqrt{\phi_x^2+\phi_y^2}}\\&=-F\frac{\phi_x^2+\phi_y^2}{\sqrt{\phi_x^2+\phi_y^2}}\\&=-F|\nabla\phi|\end{aligned}$$

where (ϕ_x,ϕ_y) are components of the vector field, so indeed the curve evolution depends on gradient. In fact, we can include a (multiplicative) stopping function of the form

$$S=\frac{1}{1+|\nabla\mathbf{P}|^n} \tag{6.52}$$

where $\nabla\mathbf{P}$ is the magnitude of the image gradient giving a stopping function (like the one in anisotropic diffusion in Eq. (3.42)) which is zero at edge points (hence stopping evolution) and near unity when there is no edge data (allowing movement). This is in fact a form of the Hamilton−Jacobi equation which is a partial

differential equation that needs to be solved so as to obtain our solution. One way to achieve this is by finite differences (as earlier approximating the differential operation) and a spatial grid (the image itself). We then obtain a solution which differences the contour at iterations $<n+1>$ and $<n>$ (separated by an interval Δt) as

$$\frac{\phi(i,j,\Delta t)^{<n+1>} - \phi(i,j,\Delta t)^{<n>}}{\Delta t} = -F|\nabla_{ij}\phi(i,j)^{<n>}| \tag{6.53}$$

where $\nabla_{ij}\phi$ represents a spatial derivative, leading to the solution

$$\phi(i,j,\Delta t)^{<n+1>} = \phi(i,j,\Delta t)^{<n>} - \Delta t(F|\nabla_{ij}\phi(i,j)^{<n>}|) \tag{6.54}$$

and we then have the required formulation for iterative operation.

This is only an introductory view, rather simplifying a complex scenario and much greater detail is to be found in the two major texts in this area (Sethian, 1999; Osher and Paragios, 2003). The real poser is how to solve it all. We shall concentrate on some of the major techniques, but not go into their details. Caselles et al. (1993) and Malladi et al. (1995) were the first to propose GAC models, which use gradient-based information for segmentation. The gradient-based GAC can detect multiple objects simultaneously but it has other important problems, which are boundary leakage, noise sensitivity, computational inefficiency and difficulty of implementation. There have been formulations (Caselles et al., 1997; Siddiqi et al., 1998; Xie and Mirmehdi, 2004) introduced to solve these problems; however, they can just increase the tolerance rather than achieve an exact solution. Several numerical schemes have also been proposed to improve computational efficiency of the level set method, including *narrow band* (Adalsteinsson and Sethian, 1995) (to find the solution within a constrained distance, i.e., to compute the level set only near the contour), *fast marching methods* (Sethian, 1999) (to constrain movement) and *additive operator splitting* (Weickert et al., 1998). Despite substantial improvements in efficiency, they can be difficult to implement and can be slow (we seek the zero level set only but solve the whole thing). These approaches show excellent results, but they are not for the less than brave—though there are numerous tutorials and implementations available on the Web. Clearly, there is a need for unified presentation, and some claim this—e.g., Caselles et al. (1997) (and linkage to parametric active contour models).

The technique which many people compare the result of their own new approach with is a GAC called the *active contour without edges*, introduced by Chan and Vese (2001), which is based on the Mumford−Shah functional (Mumford and Shah, 1989). Their model uses **regional** statistics for segmentation, and as such is a region-based level set model. The overall premise is to **avoid** using gradient (edge) information since this can lead to boundary leakage and cause the contour to collapse. A further advantage is that it can find objects when boundary data is weak or diffuse. The main strategy is to minimize energy, as in an active

contour. To illustrate the model, let us presume we have a bimodal image $\mathbf{P}$ which contains an object and a background. The object has pixels of intensity $\mathbf{P}^{\mathrm{i}}$ within its boundary and the intensity of the background is $\mathbf{P}^{\mathrm{o}}$, outside of the boundary. We can then measure a fit of a contour, or curve, C to the image as

$$F^{\mathrm{i}}(C) + F^{\mathrm{o}}(C) = \int_{\mathrm{inside}(C)} |\mathbf{P}(x,y) - c^{\mathrm{i}}|^2 \, \mathrm{d}x \, \mathrm{d}y + \int_{\mathrm{outside}(C)} |\mathbf{P}(x,y) - c^{\mathrm{o}}|^2 \, \mathrm{d}x \, \mathrm{d}y \tag{6.55}$$

where the constant c^{i} is the average brightness inside the curve, depending on the curve, and c^{o} is the brightness outside of it. The boundary of the object C_{o} is the curve which minimizes the fit derived by expressing the regions inside and outside the curve as

$$C_{\mathrm{o}} = \min_{C}(F^{\mathrm{i}}(C) + F^{\mathrm{o}}(C)) \tag{6.56}$$

(Note that the original description is excellent, though Chan and Vese are from a maths department, which makes the presentation a bit terse. Also, the strict version of minimization is actually the infimum or greatest lower bound; $\inf(X)$ is the biggest real number that is smaller than or equal to every number in X.) The minimum is when

$$F^{\mathrm{i}}(C_{\mathrm{o}}) + F^{\mathrm{o}}(C_{\mathrm{o}}) \approx 0 \tag{6.57}$$

i.e., when the curve is at the boundary of the object. When the curve C is inside the object, $F^{\mathrm{i}}(C) \approx 0$ and $F^{\mathrm{o}}(C) > 0$; conversely, when the curve is outside the object, $F^{\mathrm{i}}(C) > 0$ and $F^{\mathrm{o}}(C) \approx 0$. When the curve straddles the two and is both inside and outside the object, then $F^{\mathrm{i}}(C) > 0$ and $F^{\mathrm{o}}(C) > 0$; the function is zero when C is placed on the boundary of the object. By using regions, we are avoiding using edges and the process depends on finding the best separation between the **regions** (and by the **averaging** operation in the region, we have better **noise immunity**). If we constrain this process by introducing terms which depend on the length of the contour and the area of the contour, we extend the energy functional from Eq. (6.55) as

$$\begin{aligned} F(c^{\mathrm{i}}, c^{\mathrm{o}}, C) = {} & \mu \cdot \mathrm{length}(C) + \upsilon \cdot \mathrm{area}(C) \\ & + \lambda_1 \cdot \int_{\mathrm{inside}(C)} |P(x,y) - c^{\mathrm{i}}|^2 \, \mathrm{d}x \, \mathrm{d}y + \lambda_2 \cdot \int_{\mathrm{outside}(C)} |P(x,y) - c^{\mathrm{o}}|^2 \, \mathrm{d}x \, \mathrm{d}y \end{aligned} \tag{6.58}$$

where μ, υ, λ_1, and λ_2 are parameters controlling selectivity. The contour is then, for a fixed set of parameters, chosen by minimization of the energy functional as

$$C_{\mathrm{o}} = \min_{c^{\mathrm{i}}, c^{\mathrm{o}}, C}(F(c^{\mathrm{i}}, c^{\mathrm{o}}, C)) \tag{6.59}$$

A level set formulation is then used wherein an approximation to the unit step function (the Heaviside function) is defined to control the influence of points

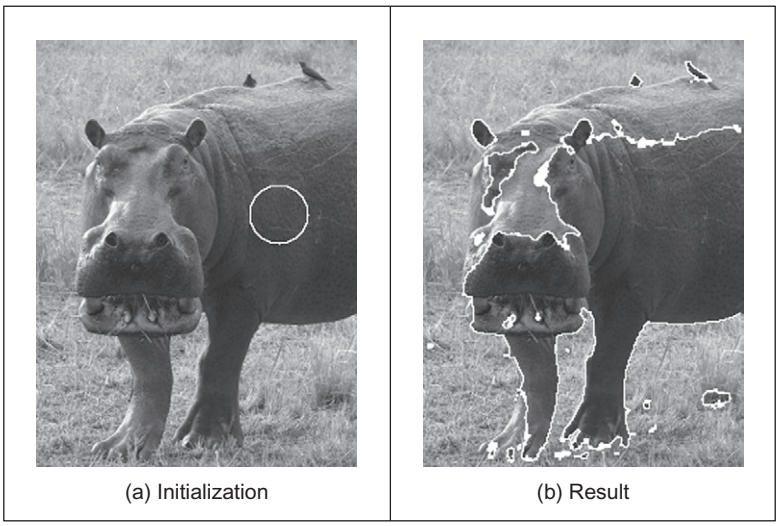


FIGURE 6.14

Extraction by a level-set-based approach.

within and without (outside) the contour, which by differentiation gives an approximation to an impulse (the Dirac function), and with a solution to a form of equation (6.51) (in discrete form) is used to update the level set.

The active contour without edges model can address problems with initialization, noise, and boundary leakage (since it uses regions, not gradients) but still suffers from computational inefficiency and difficulty in implementation because of the level set method. An example result is shown in Figure 6.14 where the target aim is to extract the hippo—the active contour without edges aims to split the image into the extracted object (the hippo) and its background (the grass). In order to do this, we need to specify an initialization which we shall choose to be within a small circle inside the hippo, as shown in Figure 6.14(a). The result of extraction is shown in Figure 6.14(b) and we can see that the technique has detected much of the hippo, but the result is not perfect. The values used for the parameters here were $\lambda_1 = \lambda_2 = 1.0$; $\upsilon = 0$ (i.e., area was not used to control evolution); $\mu = 0.1 \times 255^2$ (the length parameter was controlled according to the image resolution) and some internal parameters were $h = 1$ (a 1 pixel step space); $\Delta t = 0.1$ (a small time spacing) and $\varepsilon = 1$ (a parameter within the step and hence the impulse functions). Alternative choices are possible and can affect the result achieved. The result here has been selected to show performance attributes, the earlier result (Figure 6.12) was selected to demonstrate finesse.

The regions with intensity and appearance that are most similar to the selected initialization have been identified in the result: this is much of the hippo, including the left ear and the region around the left eye but omitting some of the upper

body. There are some small potential problems too: there are some birds extracted on the top of the hippo and a small region underneath it (was this hippo's breakfast we wonder?). Note that by virtue of the regional level set formulation, the image is treated in its entirety and multiple shapes are detected, some well away from the target shape. By and large, the result looks encouraging as much of the hippo is extracted in the result and the largest shape contains much of the target; if we were to seek to get an exact match, then we would need to use an exact model such as the GHT or impose a model on the extraction, such as a statistical shape prior. That the technique can operate best when the image is bimodal is reflected in that extraction is most successful when there is a clear difference between the target and the background, such as in the lower body. An alternative interpretation is that the technique clearly can handle situations where the edge data is weak and diffuse, such as in the upper body.

Techniques have moved on and can now include statistical priors to guide shape extraction (Cremers et al., 2007). One study shows the relationship between parametric and GACs (Xu et al., 2000). As such, snakes and evolutionary approaches to shape extraction remain an attractive and stimulating area of research, so as ever it is well worth studying the literature to find new, accurate, techniques with high performance and low computational cost. We shall now move to determining **skeletons** which, though more a form of low-level operation, can use evidence gathering in implementation thus motivating its inclusion rather late in this book.

6.4 Shape skeletonization

6.4.1 Distance transforms

It is possible to describe a shape not just by its perimeter, or its area, but also by its *skeleton*. Here we do not mean an anatomical skeleton, more a central **axis** to a shape. This is then the axis which is equidistant from the borders of a shape and can be determined by a *distance transform*. In this way we have a representation that has the same topology, the same size, and orientation, but contains just the essence of the shape. As such, we are again in morphology and there has been interest for some while in binary shape analysis (Borgefors, 1986).

Essentially, the distance transform shows the distance from each point in an image shape to its central axis. (We are measuring distance here by difference in coordinate values, other measures of distance such as Euclidean are considered later in Chapter 8.) Intuitively, the distance transform can be achieved by successive erosion and each pixel is labeled with the number of erosions before it disappeared. Accordingly, the pixels at the border of a shape will have a distance transform of unity, those adjacent inside will have a value of two, and so on. This is illustrated in Figure 6.15 where Figure 6.15(a) shows the analyzed shape (a rectangle derived by, say, thresholding an image—the superimposed pixel values

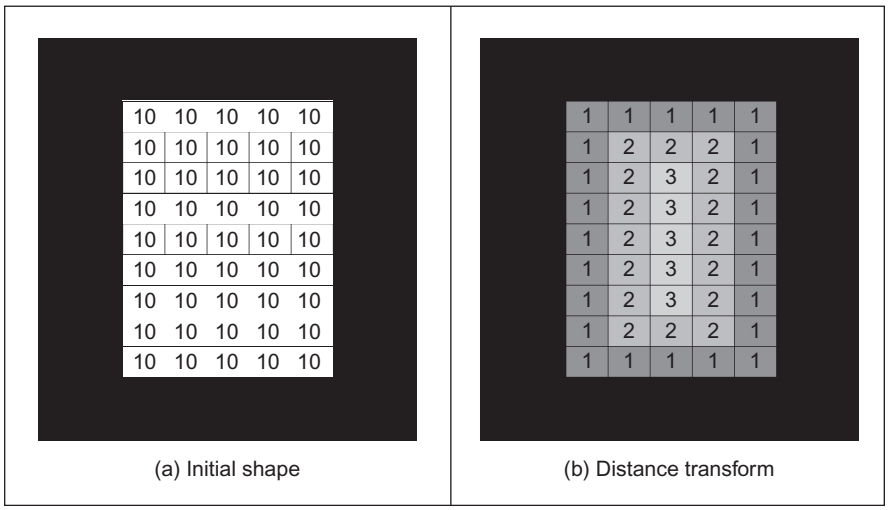


FIGURE 6.15

Illustrating distance transformation.

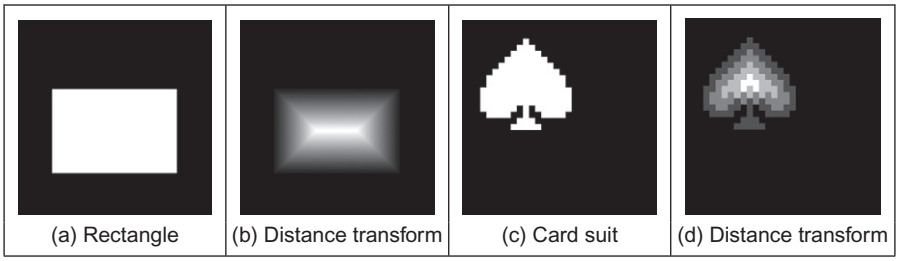


FIGURE 6.16

Applying the distance transformation.

are arbitrary here as it is simply a binary image) and Figure 6.15(b) shows the distance transform where the pixel values are the distance. Here the central axis has a value of three as it takes that number of erosions to reach it from either side.

The application to a rectangle at higher resolution is shown in Figure 6.16(a) and (b). Here we can see that the central axis is quite clear and actually includes parts that reach toward the corners (and the central axis can be detected (Niblack et al., 1992) from the transform data). The application to a more irregular shape is shown applied to that of a card suit in Figure 6.16(c) and (d).

The natural difficulty is of course the effect of noise. This can change the resulting, as shown in Figure 6.17. This can certainly be ameliorated by using the earlier morphological operators (Section 3.6) to clean the image, but this can

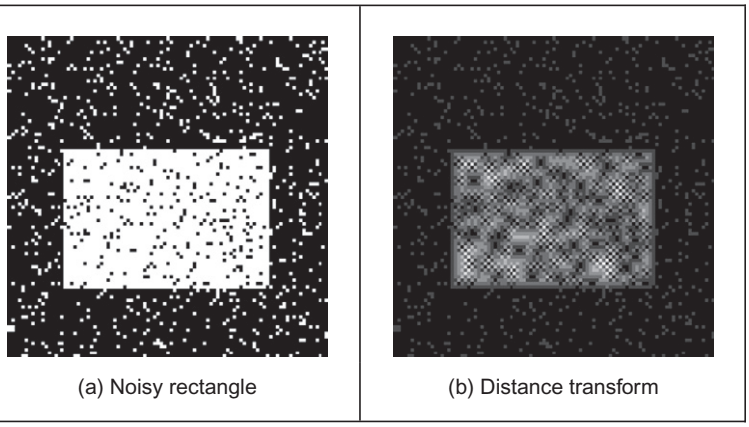


FIGURE 6.17

Distance transformation on noisy images.

obscure the shape when the noise is severe. The major point is that this noise shows that the effect of a small change in the object can be quite severe on the resulting distance transform. As such, it has little tolerance of occlusion or in change to its perimeter.

The natural extension from distance transforms is to the *medial axis transform* (Blum, 1967), which determines the skeleton that consists of the locus of all the centers of maximum disks in the analyzed region/shape. This has found use in feature extraction and description so naturally approaches have considered improvement in **speed** (Lee, 1982). One more recent study (Katz and Pizer, 2003) noted the practical difficulty experienced in **noisy** imagery: "It is well documented how a tiny change to an object's boundary can cause a large change in its medial axis transform." To handle this, and hierarchical shape decomposition, the new approach "provides a natural parts-hierarchy while eliminating instabilities due to small boundary changes." In fact, there is a more authoritative study available on medial representations (Siddiqi and Pizer, 2008) which describes formulations and properties of medial axis and distance transformations, together with applications. An alternative is to seek an approach which is designed implicitly to handle noise, say by averaging, and we shall consider this type of approach next.

6.4.2 Symmetry

Symmetry is a natural property, and there have been some proposed links with human perception of beauty. Rather than rely on finding the border of a shape, or its shape, we can locate features according to their **symmetrical properties**. So it is a totally different basis to find shapes and is intuitively very appealing since it exposes **structure**. (An old joke is that "symmetry" should be a palindrome, fail!)

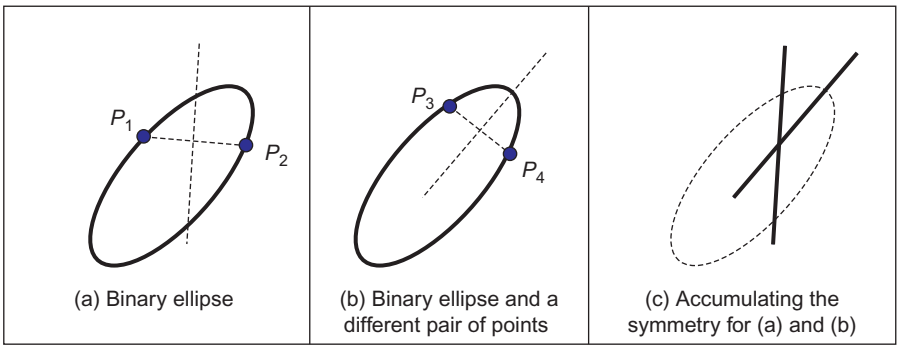


FIGURE 6.18

Primitive symmetry operator—basis.

There are many types of symmetry which are typified by their invariant properties such as **position**-invariance, *circular symmetry* (which is invariant to **rotation**), and **reflection**. We shall concentrate here on *bilateral reflection symmetry* (**mirror**-symmetry), giving pointers to other approaches and analysis later in this section.

One way to determine reflection symmetry is to find the midpoint of a pair of edge points and to then draw a line of votes in an accumulator wherein the gradient of the line of votes is normal to the line joining the two edge points. When this is repeated for all pairs of edge points, the maxima should define the estimates of maximal symmetry for the largest shape. This process is illustrated in Figure 6.18 where we have an ellipse. From Figure 6.18(a), a line can be constructed that is normal to the line joining two (edge) points P_1 and P_2 and a similar line in Figure 6.18(b) for points P_3 and P_4. These two lines are the lines of votes that are drawn in an accumulator space as shown in Figure 6.18(c). In this manner, the greatest number of lines of votes will be drawn along the ellipse axes. If the shape was a circle, the resulting accumulation of symmetry would have the greatest number of votes at the center of the circle, since it is a totally symmetric shape. Note that one major difference between the symmetry operator and a medial axis transform is that the symmetry operator will find the two axes, whereas the medial transform will find the largest.

This is shown in Figure 6.19(b) which is the accumulator for the ellipse in Figure 6.19(a). The resulting lines in the accumulator are indeed the two axes of symmetry of the ellipse. This procedure might work well in this case, but lacks the selectivity of a practical operator and will be sensitive to noise and occlusion. This is illustrated for the accumulation for the shapes in Figure 6.19(c). The result in Figure 6.19(d) shows the axes of symmetry for the two ellipsoidal shapes and the point at the center of the circle. It also shows a great deal of noise (the mush in the image) and this renders this approach useless. (Some of the noise in

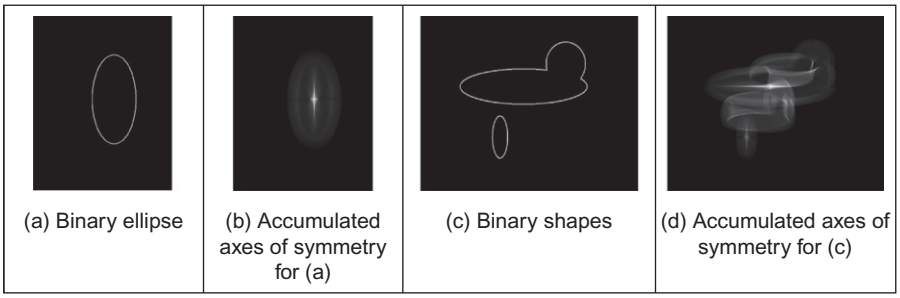


FIGURE 6.19

Application of a primitive symmetry operator.

Figure 6.19(b) and (d) is due to implementation but do not let that distract you.) Basically, the technique needs greater selectivity.

To achieve selectivity, we can use edge direction to filter the pairs of points. If the pair of points does not satisfy specified conditions on gradient magnitude and direction, then they will not contribute to the symmetry estimate. This is achieved in the *discrete symmetry operator* (Reisfeld et al., 1995) which essentially forms an **accumulator** of points that are measures of symmetry between image points. Pairs of image points are attributed symmetry values that are derived from a **distance** weighting function, a **phase** weighting function, and the **edge** magnitude at each pair of points. The distance weighting function controls the scope of the function to control whether points which are more distant contribute in a similar manner to those which are close together. The phase weighting function shows when edge vectors at the pair of points point to each other and is arranged to be zero when the edges are pointing in the same direction (were that to be the case, they could not belong to the same shape—by symmetry). The symmetry accumulation is at the center of each pair of points. In this way, the accumulator measures the degree of symmetry between image points controlled by the edge strength. The distance weighting function D is

$$D(i,j,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{|\mathbf{P}_i - \mathbf{P}_j|}{2\sigma}} \tag{6.60}$$

where i and j are the indices to two image points $\mathbf{P}_i$ and $\mathbf{P}_j$ and the deviation σ controls the scope of the function by scaling the contribution of the distance between the points in the exponential function. A small value for the deviation σ implies local operation and detection of local symmetry. Larger values of σ imply that points that are further apart contribute to the accumulation process as well as ones that are close together. In, say, application to the image of a face, large and small values of σ will aim for the whole face or the eyes, respectively.

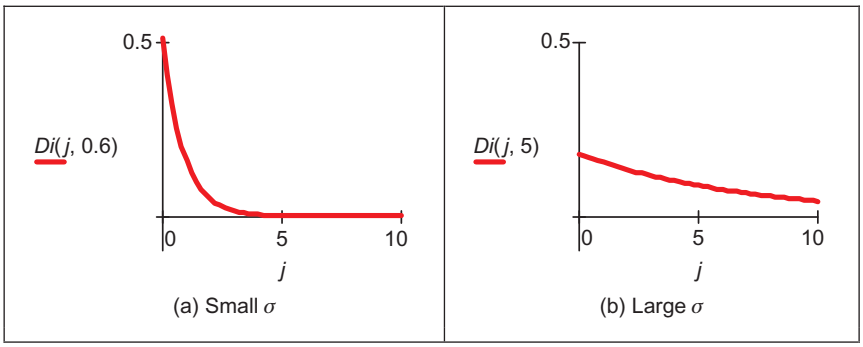


FIGURE 6.20

Effect of σ on distance weighting.

The effect of the value of σ on the scalar distance weighting function expressed as Eq. (6.61) is illustrated in Figure 6.20:

$$Di(j,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{j}{\sqrt{2}\sigma}} \tag{6.61}$$

Figure 6.20(a) shows the effect of a small value for the deviation, $\sigma = 0.6$, and shows that the weighting is greatest for closely spaced points and drops rapidly for points with larger spacing. Larger values of σ imply that the distance weight drops less rapidly for points that are more widely spaced, as shown in Figure 6.20(b) where $\sigma = 5$, allowing points which are spaced further apart to contribute to the measured symmetry. The phase weighting function P is

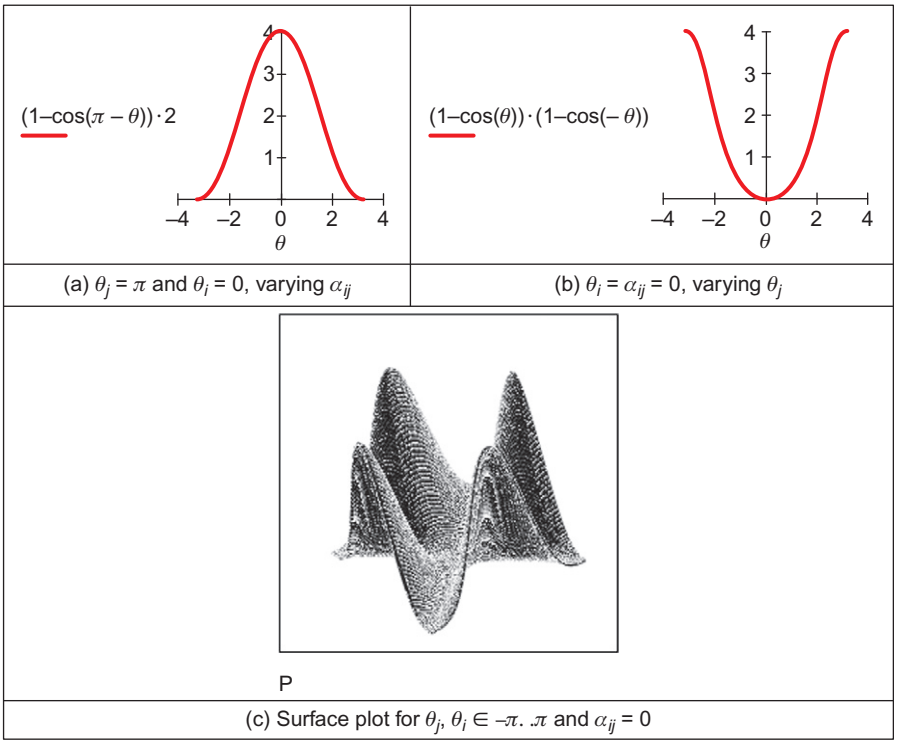
$$P(i,j) = (1 - \cos(\theta_i + \theta_j - 2\alpha_{ij})) \times (1 - \cos(\theta_i - \theta_j)) \tag{6.62}$$

where θ is the edge direction at the two points and α_{ij} measures the direction of a line joining the two points:

$$\alpha_{ij} = \tan^{-1}\left(\frac{y(\mathbf{P}_j) - y(\mathbf{P}_i)}{x(\mathbf{P}_j) - x(\mathbf{P}_i)}\right) \tag{6.63}$$

where $x(\mathbf{P}_i)$ and $y(\mathbf{P}_i)$ are the x and y coordinates of the point $\mathbf{P}_i$, respectively. This function is minimum when the edge direction at two points is in the same direction ($\theta_j = \theta_i$) and is maximum when the edge direction is away from each other ($\theta_i = \theta_j + \pi$), along the line joining the two points ($\theta_j = \alpha_{ij}$).

The effect of relative edge direction on phase weighting is illustrated in Figure 6.21 where Figure 6.21(a) concerns two edge points that point toward each other and describes the effect on the phase weighting function by varying α_{ij}. This shows how the phase weight is maximum when the edge direction at the two points is along the line joining them, in this case when $\alpha_{ij} = 0$ and $\theta_i = 0$. Figure 6.21(b) concerns one point with edge direction along the line joining two points, where the edge direction at the second point is varied. The phase weighting function is maximum when the edge direction at each point is toward each



(a) $\theta_j = \pi$ and $\theta_i = 0$, varying α_{ij}

(b) $\theta_i = \alpha_{ij} = 0$, varying θ_j

(c) Surface plot for θ_j, $\theta_i \in -\pi . . \pi$ and $\alpha_{ij} = 0$

FIGURE 6.21

Effect of relative edge direction on phase weighting.

other, in this case when $|\theta_j| = \pi$. Naturally, it is more complex than this and Figure 6.21(c) shows the surface of the phase weighting function for $\alpha_{ij} = 0$, with its four maxima.

The symmetry relation between two points is then defined as

$$C(i,j,\sigma) = D(i,j,\sigma) \times P(i,j) \times E(i) \times E(j) \tag{6.64}$$

where E is the edge magnitude expressed in logarithmic form as

$$E(i) = \log(1 + M(i)) \tag{6.65}$$

where M is the edge magnitude derived by application of an edge detection operator. The symmetry contribution of two points is accumulated at the midpoint of the line joining the two points. The total symmetry $S_{\mathbf{P}_m}$ at point $\mathbf{P}_m$ is the sum of the measured symmetry for all pairs of points which have their midpoint at $\mathbf{P}_m$, i.e., those points $\Gamma(\mathbf{P}_m)$ given by

$$\Gamma(\mathbf{P}_m) = \left[(i,j) \left| \frac{\mathbf{P}_i + \mathbf{P}_j}{2} \right| = \mathbf{P}_m \quad \forall i \neq j \right] \tag{6.66}$$

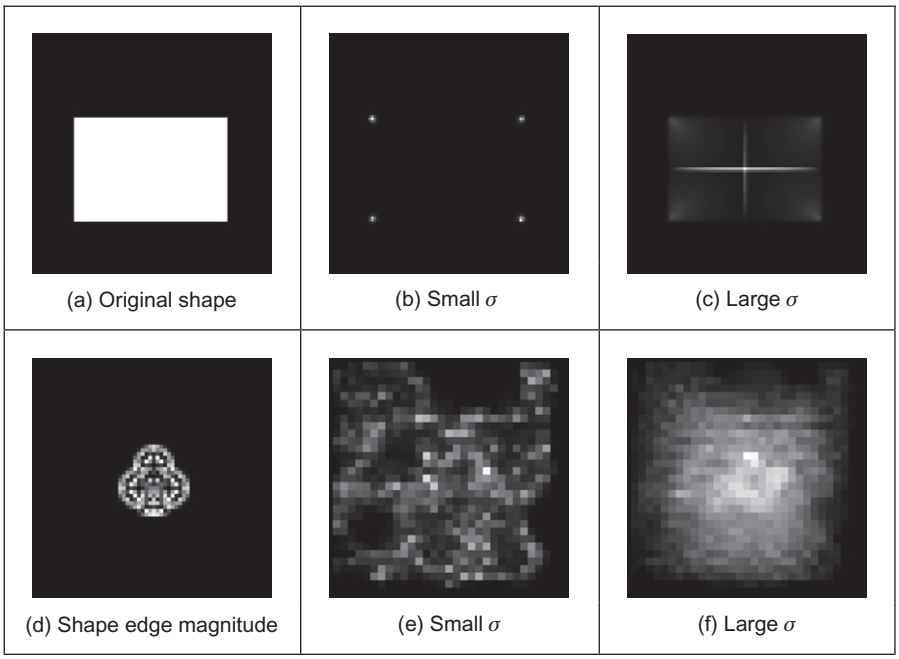


FIGURE 6.22

Applying the symmetry operator for feature extraction.

and the accumulated symmetry is then

$$S_{\mathbf{P}_m}(\sigma) = \sum_{i,j \in \Gamma(\mathbf{P}_m)} C(i,j,\sigma) \tag{6.67}$$

The result of applying the symmetry operator to two images is shown in Figure 6.22, for small and large values of σ. Figure 6.22(a) and (d) shows the image of a rectangle and the club, respectively, to which the symmetry operator was applied, Figure 6.22(b) and (e) for the symmetry operator with a **low** value for the deviation parameter, showing detection of areas with high localized symmetry; Figure 6.22(c) and (f) are for a **large** value of the deviation parameter which detects overall symmetry and places a peak near the center of the target shape. In Figure 6.22(b) and (e), the symmetry operator acts as a corner detector where the edge direction is discontinuous. (Note that this rectangle is one of the synthetic images we can use to test techniques, since we can understand its output easily. We also tested the operator on the image of a circle, since the circle is completely symmetric and its symmetry plot is a single point at the center of the circle.) In Figure 6.22(e), the discrete symmetry operator provides a peak close to the position of the accumulator space peak in the GHT. Note that if the reference point

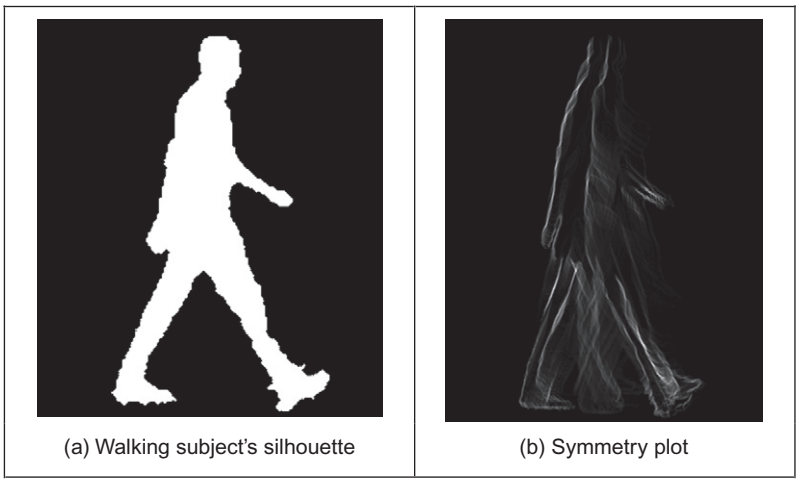


FIGURE 6.23

Applying the symmetry operator for recognition by gait (Hayfron-Acquah et al., 2003).

specified in the GHT is the center of symmetry, the results of the discrete symmetry operator and the GHT would be the same for large values of deviation.

There is a **review** of the performance of state-of-art symmetry detection operators (Park et al., 2008) which compared those new operators which offered multiple symmetry detection, on standard databases. We have been considering a discrete operator, a *continuous symmetry operator* has been developed (Zabrodsky et al., 1995), and a later clarification (Kanatani, 1997) was aimed to address potential practical difficulty associated with **hierarchy** of symmetry (namely that symmetrical shapes have subsets of regions, also with symmetry). More advanced work includes symmetry between pairs of points and its extension to constellations (Loy and Eklundh, 2006) thereby imposing structure on the symmetry extraction, and analysis of local image symmetry to expose structure via derivative of Gaussian filters (Griffin and Lillholm, 2010), thereby accruing the advantages of a frequency domain approach. There have also been a number of sophisticated approaches to detection of *skewed symmetry* (Gross and Boult, 1994; Cham and Cipolla, 1995), with later extension to detection in *orthographic projection* (Van Gool et al., 1995). Another generalization addresses the problem of **scale** (Reisfeld, 1996) and extracts points of symmetry together with scale. A *focusing* ability has been added to the discrete symmetry operator by reformulating the distance weighting function (Parsons and Nixon, 1999) and we were to deploy this when using symmetry in an approach which recognize people by their gait (the way they walk) (Hayfron-Acquah et al., 2003). Why symmetry was chosen for this task is illustrated in Figure 6.23: this shows the main axes of symmetry of the walking subject

(Figure 6.23(b)) which exist within the body, largely defining the skeleton. There is another axis of symmetry between the legs. When the symmetry operator is applied to a sequence of images, this axis grows and retracts. By agglomerating the sequence and describing it by a (low-pass filtered) Fourier transform, we can determine a set of numbers which are the same for the same person and different from those for other people, thus achieving recognition. No approach as yet has alleviated the computational burden associated with the discrete symmetry operator, and some of the process used can be used to reduce the requirement (e.g., judicious use of thresholding).

6.5 Flexible shape models—active shape and active appearance

So far, our approaches to analyzing shape have concerned a match to image data. This has concerned usually a match between a model (either a template that can deform or a shape that can evolve) and a single image. An active contour is flexible, but its evolution is essentially controlled by local properties, such as the local curvature or edge strength. The chosen value for, or the likely range of, the parameters to weight these functionals may have been learnt by extensive testing on a database of images of similar type to the one used in application, or selected by experience. A completely different approach is to consider that if the database contains all possible **variations** of a shape, like its appearance or pose, the database can form a model of the likely variation of that shape. As such, if we can incorporate this as a global constraint, while also guiding the match to the most likely version of a shape, then we have a deformable approach which is guided by the **statistics** of the likely variation in a shape. These approaches are termed *flexible templates* and use **global** shape constraints formulated from exemplars in training data.

This major new approach is called *active shape modeling*. The essence of this approach concerns a model of a shape made up of points: the variation in these points is called the *point distribution model*. The chosen **landmark** points are labeled on the training images. The set of training images aims to capture all possible variations of the shape. Each point describes a particular point on the boundary, so order is important in the labeling process. Example choices for these points include where the curvature is high (e.g., the corner of an eye) or at the apex of an arch where the contrast is high (e.g., the top of an eyebrow). The statistics of the variations in position of these points describe the ways in which a shape can appear. Example applications include finding the human face in images, for purposes say of automatic face recognition. The only part of the face for which a distinct model is available is the round circle in the iris—and this can be small except at very high resolution. The rest of the face is made of unknown shapes and these can change with change in face expression. As such, they are

well suited to a technique which combines shape with distributions, since we have a known set of shapes and a fixed interrelationship, but some of the detail can change. The variation in detail is what is captured in an ASM.

Naturally, there is a lot of data. If we choose lots of points and we have lots of training images, we shall end up with an enormous number of points. That is where *principal components analysis* comes in as it can compress data into the most significant items. Principal components analysis is an established mathematical tool: help is available in Chapter 12, Appendix 3, on the Web and in the literature *Numerical Recipes* (Press et al., 1992). Essentially, it rotates a coordinate system so as to achieve maximal discriminatory capability: we might not be able to see something if we view it from two distinct points, but if we view it from some point in between then it is quite clear. That is what is done here: the coordinate system is rotated so as to work out the most significant variations in the morass of data. Given a set of N training examples where each example is a set of n points, for the ith training example $\mathbf{x}_i$, we have

$$\mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{ni}) \quad i \in 1, N \tag{6.68}$$

where x_{ki} is the kth variable in the ith training example. When this is applied to shapes, each element is the two coordinates of each point. The average is then computed over the whole set of training examples as

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \tag{6.69}$$

The deviation of each example from the mean $\delta\mathbf{x}_i$ is then

$$\delta\mathbf{x}_i = \mathbf{x}_i - \overline{\mathbf{x}} \tag{6.70}$$

This difference reflects how far each example is from the mean at a point. The $2n \times 2n$ covariance matrix $\mathbf{S}$ shows how far all the differences are from the mean as

$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N} \delta\mathbf{x}_i \, \delta\mathbf{x}_i^{\mathrm{T}} \tag{6.71}$$

Principal components analysis of this covariance matrix shows by how much these examples, and hence a shape, can change. In fact, any of the exemplars of the shape can be approximated as

$$\mathbf{x}_i = \overline{\mathbf{x}} + \mathbf{P}\mathbf{w} \tag{6.72}$$

where $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_t)$ is a matrix of the first t eigenvectors and $\mathbf{w} = (w_1, w_2, \ldots, w_t)^{\mathrm{T}}$ is a corresponding vector of weights where each weight value controls the contribution of a particular eigenvector. Different values in $\mathbf{w}$ give different occurrences of the model or shape. Given that these changes are within specified limits, then the new model or shape will be similar to the basic (mean)

shape. This is because the modes of variation are described by the (unit) eigenvectors of $\mathbf{S}$, as

$$\mathbf{S}\mathbf{p}_k = \lambda_k \mathbf{p}_k \tag{6.73}$$

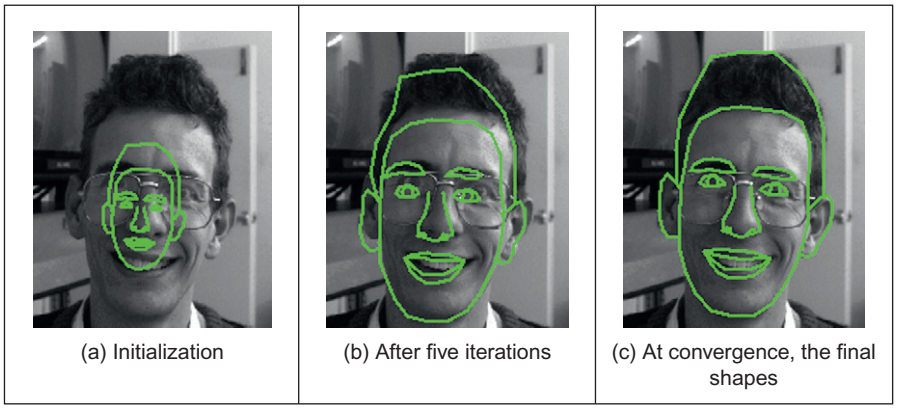
where λ_k denotes the eigenvalues and the eigenvectors obey orthogonality such that

$$\mathbf{p}_k \mathbf{p}_k^{\mathrm{T}} = 1 \tag{6.74}$$

and where the eigenvalues are rank ordered such that $\lambda_k \geq \lambda_{k+1}$. Here, the largest eigenvalues correspond to the most significant modes of variation in the data. The proportion of the variance in the training data, corresponding to each eigenvector, is proportional to the corresponding eigenvalue. As such, a limited number of eigenvalues (and eigenvectors) can be used to encompass the majority of the data. The remaining eigenvalues (and eigenvectors) correspond to modes of variation that are hardly present in the data (like the proportion of very high-frequency contribution of an image; we can reconstruct an image mainly from the low-frequency components, as used in image coding). Note that in order to examine the statistics of the labeled landmark points over the training set applied to a new shape, the points need to be aligned and established procedures are available (Cootes et al., 1995).

The process of application (to find instances of the modeled shape) involves an iterative approach to bring about increasing match between the points in the model and the image. This is achieved by examining regions around model points to determine the best nearby match. This provides estimates of the appropriate translation, scale rotation, and eigenvectors to best fit the model to the data. This is repeated until the model converges to the data, when there is little change to the parameters. Since the models only change to better fit the data and are controlled by the expected appearance of the shape, they were called ASMs. The application of an ASM to find the face features of one of the technique's inventors (yes, that's Tim behind the target shapes) is shown in Figure 6.24 where the initial position is shown in Figure 6.24(a), the result after five iterations in Figure 6.24(b), and the final result in Figure 6.24(c). The technique can operate in a coarse-to-fine manner, working at low resolution initially (and making relatively fast moves) while slowing to work at finer resolution before the technique result improves no further at convergence. Clearly, the technique has not been misled either by the spectacles or by the presence of other features in the background. This can be used either for enrollment (finding the face automatically) or for automatic face recognition (finding and describing the features). Naturally, the technique cannot handle initialization which is too poor—though clearly by Figure 6.24(a) the initialization needs not to be too close either.

ASMs have been applied in face recognition (Lanitis et al., 1997), medical image analysis (Cootes et al., 1994) (including 3D analysis, Hill et al., 1994), and in industrial inspection (Cootes et al., 1995). A similar theory has been used to



(a) Initialization (b) After five iterations (c) At convergence, the final shapes

FIGURE 6.24

Finding face features using an ASM.

develop a new approach that incorporates texture, called *active appearance models* (AAMs) (Cootes et al., 1998a,b). This approach again represents a shape as a set of landmark points and uses a set of training data to establish the potential range of variation in the shape. One major difference is that AAMs explicitly include texture and updates model parameters to move landmark points closer to image points by matching texture in an iterative search process. The essential differences between ASMs and AAMs include:

1. ASMs use texture information local to a point, whereas AAMs use texture information in a whole region.
2. ASMs seek to minimize the distance between model points and the corresponding image points, whereas AAMs seek to minimize distance between a synthesized model and a target image.
3. ASMs search around the current position—typically along profiles normal to the boundary, whereas AAMs consider the image only at the current position.

One comparison (Cootes et al., 1999) has shown that although ASMs can be faster in implementation than AAMs, the AAMs can require fewer landmark points and can converge to a better result, especially in terms of texture (wherein the AAM was formulated). We await with interest further developments in these approaches to flexible shape modeling. An example result by an AAM for face feature finding is shown in Figure 6.25. Clearly, this cannot demonstrate computational advantage, but we can see the inclusion of hair in the eyebrows has improved segmentation there. Inevitably, interest has concerned improving computational requirements, in one case by an efficient fitting algorithm based on the inverse compositional image alignment algorithm (Matthews and Baker, 2004). Recent interest has concerned ability to handle occlusion (Gross et al., 2006), as occurring either by changing (3D) orientation or by gesture.

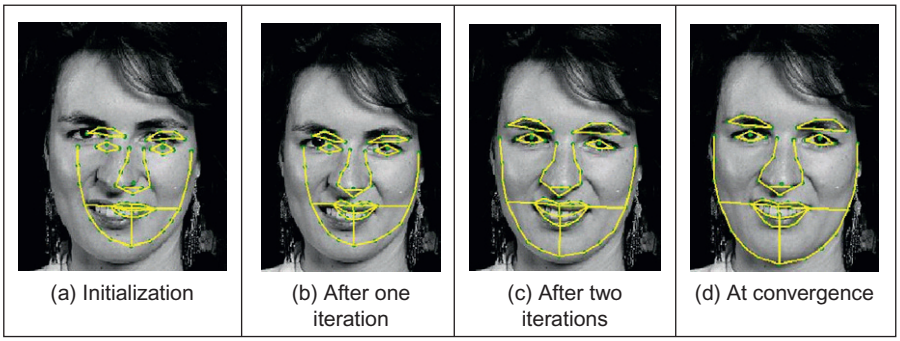


FIGURE 6.25

Finding face features using an AAM.

6.6 Further reading

The majority of further reading in finding shapes concerns papers, many of which have already been referenced. An excellent survey of the techniques used for **feature extraction** (including template matching, deformable templates, etc.) can be found in Trier et al. (1996), while a broader view was taken later (Jain et al., 1998). A comprehensive survey of flexible extractions from **medical** imagery (McInerney and Terzopolous, 1996) reinforces the dominance of snakes in medical image analysis, to which they are particularly suited given a target of smooth shapes. (An excellent survey of history and progress of medical image analysis is available (Duncan and Ayache, 2000).) Few of the textbooks devote much space to shape extraction and snakes, especially level set methods are too recent a development to be included in many textbooks. One text alone is dedicated to shape analysis (Van Otterloo, 1991) and contains many discussions on symmetry, and there is a text on distance and medial axis transformation (Siddiqi and Pizer, 2008). A visit to Prof. Cootes' (personal) web pages http://www.isbe.man.ac.uk/~bim/ reveals a lengthy report on flexible shape modeling and a lot of support material (including Windows and Linux code) for active shape modeling. Alternatively, a textbook from the same team is now available (Davies et al., 2008). For a review of work on level set methods for image segmentation, see Cremers et al. (2007).

6.7 References

Adalsteinsson, D., Sethian, J., 1995. A fast level set method for propagating interfaces. J. Comput. Phys. 118 (2), 269–277.

Bamford, P., Lovell, B., 1998. Unsupervised cell nucleus segmentation with active contours. Signal Process. 71, 203−213.

Benn, D.E., Nixon, M.S., Carter, J.N., 1999. Extending concentricity analysis by deformable templates for improved eye extraction. Proceedings of the Second International Conference on Audio- and Video-Based Biometric Person Authentication AVBPA99, pp. 1−6.

Berger, M.O., 1991. Towards dynamic adaption of snake contours. Proceedings of the Sixth International Conference on Image Analysis and Processing, Como, Italy, pp. 47−54.

Blum, H., 1967. A transformation for extracting new descriptors of shape. In: Wathen-Dunn, W. (Ed.), Models for the Perception of Speech and Visual Form. MIT Press, Cambridge, MA.

Borgefors, G., 1986. Distance transformations in digital images. Comput. Vision Graph. Image Process. 34 (3), 344−371.

Caselles, V., Catte, F., Coll, T., Dibos, F., 1993. A geometric model for active contours. Numerische Math. 66, 1−31.

Caselles, V., Kimmel, R., Sapiro, G., 1997. Geodesic active contours. Int. J. Comput. Vision 22 (1), 61−79.

Cham, T.J., Cipolla, R., 1995. Symmetry detection through local skewed symmetries. Image Vision Comput. 13 (5), 439−450.

Chan, T.F., Vese, L.A., 2001. Active contours without edges. IEEE Trans. IP 10 (2), 266−277.

Cohen, L.D., 1991. On active contour models and balloons. CVGIP: Image Understanding 53 (2), 211−218.

Cohen, L.D., Cohen, I., 1993. Finite-element methods for active contour models and balloons for 2D and 3D images. IEEE Trans. PAMI 15 (11), 1131−1147.

Cohen, I., Cohen, L.D., Ayache, N., 1992. Using deformable surfaces to segment 3D images and inter differential structures. CVGIP: Image Understanding 56 (2), 242−263.

Cootes, T.F., Hill, A., Taylor, C.J., Haslam, J., 1994. The use of active shape models for locating structures in medical images. Image Vision Comput. 12 (6), 355−366.

Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J., 1995. Active shape models—their training and application. CVIU 61 (1), 38−59.

Cootes, T.F., Edwards, G.J., Taylor, C.J., 1998a. A Comparative evaluation of active appearance model algorithms. In: Lewis, P.H., Nixon, M.S. (Eds.), Proceedings of the British Machine Vision Conference 1998, BMVC98, vol. 2, pp. 680−689.

Cootes, T., Edwards, G.J., Taylor, C.J., 1998b. Active appearance models. In: Burkhardt, H., Neumann, B. (Eds.), Proceedings of the ECCV 98, vol. 2, pp. 484−498.

Cootes, T.F., Edwards, G.J., Taylor, C.J., 1999. Comparing active shape models with active appearance models. In: Pridmore, T., Elliman, D. (Eds.), Proceedings of the British Machine Vision Conference 1999, BMVC99, vol. 1**,** pp. 173−182.

Cremers, D., Tischhäuser, F., Weickert, J., Schnörr, C., 2002. Diffusion snakes: introducing statistical shape knowledge into the Mumford−Shah functional. Int. J. Comput. Vision 50 (3), 295−313.

Cremers, D., Rousson, M., Deriche, R., 2007. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. Int. J. Comput. Vision 72 (2), 195−215.

Davies, R., Twining, C., Taylor, C.J., 2008. Statistical Models of Shape: Optimisation and Evaluation. Springer.

Duncan, J.S., Ayache, N., 2000. Medical image analysis: progress over two decades and the challenges ahead. IEEE Trans. PAMI 22 (1), 85–106.

Felzenszwalb, P.F., Huttenlocher, D.P., 2005. Pictorial structures for object recognition. Int. J. Comput. Vision 61 (1), 55–79.

Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D., 2010. Object detection with discriminatively trained part based models. Trans. PAMI 32 (9), 1627–1645.

Fischler, M.A., Elschlager, R.A., 1973. The representation and matching of pictorial structures. IEEE Trans. Comp. C-22 (1), 67–92.

Geiger, D., Gupta, A., Costa, L.A., Vlontsos, J., 1995. Dynamical programming for detecting, tracking and matching deformable contours. IEEE Trans. PAMI 17 (3), 294–302.

Goldberg, D., 1988. Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley.

Griffin, L.D., Lillholm, M., 2010. Symmetry sensitivities of derivative-of-Gaussian filters. IEEE Trans. PAMI 32 (6), 1072–1083.

Gross, A.D., Boult, T.E., 1994. Analysing skewed symmetries. Int. J. Comput. Vision 13 (1), 91–111.

Gross, R., Matthews, I., Baker, S., 2006. Active appearance models with occlusion. Image Vision Comput. 24 (6), 593–604.

Gunn, S.R., Nixon, M.S., 1997. A robust snake implementation: a dual active contour. IEEE Trans. PAMI 19 (1), 63–68.

Gunn, S.R., Nixon, M.S., 1998. Global and local active contours for head boundary extraction. Int. J. Comput. Vision 30 (1), 43–54.

Hayfron-Acquah, J.B., Nixon, M.S., Carter, J.N., 2003. Automatic gait recognition by symmetry analysis. Pattern Recog. Lett. 24 (13), 2175–2183.

Hill, A., Cootes, T.F., Taylor, C.J., Lindley, K., 1994. Medical image interpretation: a generic approach using deformable templates. J. Med. Informat. 19 (1), 47–59.

Ivins, J., Porrill, J., 1995. Active region models for segmenting textures and colours. Image Vision Comput. 13 (5), 431–437.

Jain, A.K., Zhong, Y., Dubuisson-Jolly, M.-P., 1998. Deformable template models: a review. Signal Process. 71, 109–129.

Kanatani, K., 1997. Comments on "symmetry as a continuous feature". IEEE Trans. PAMI 19 (3), 246–247.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: active contour models. Int. J. Comput. Vision 1 (4), 321–331.

Katz, R.A., Pizer, S.M., 2003. Untangling the blum medial axis transform. Int. J. Comput. Vision 55 (2-3), 139–153.

Lai, K.F., Chin, R.T., 1994. On regularisation, extraction and initialisation of the active contour model (snakes). Proceedings of the First Asian Conference on Computer Vision, pp. 542–545.

Lai, K.F., Chin, R.T., 1995. Deformable contours—modelling and extraction. IEEE Trans. PAMI 17 (11), 1084–1090.

Lanitis, A., Taylor, C.J., Cootes, T., 1997. Automatic interpretation and coding of face images using flexible models. IEEE Trans. PAMI 19 (7), 743–755.

Lee, D.T., 1982. Medial axis transformation of a planar shape. IEEE Trans. PAMI 4, 363–369.

Loy, G., Eklundh, J.-O., 2006. Detecting symmetry and symmetric constellations of features. Proceedings of the ECCV 2006, Part II, LNCS, vol. 3952, pp. 508–521.

Malladi, R., Sethian, J.A., Vemuri, B.C., 1995. Shape modeling with front propagation: a level set approach. IEEE Trans. PAMI 17 (2), 158–175.

Matthews, I., Baker, S., 2004. Active appearance models revisited. Int. J. Comput. Vision 60 (2), 135–164.

McInerney, T., Terzopolous, D., 1996. Deformable models in medical image analysis, a survey. Med. Image Anal. 1 (2), 91–108.

Mumford, D., Shah, J., 1989. Optimal approximation by piecewise smooth functions and associated variational problems. Comms. Pure Appl. Math 42, 577–685.

Niblack, C.W., Gibbons, P.B., Capson, D.W., 1992. Generating skeletons and centerlines from the distance transform. CVGIP: Graph. Models Image Process. 54 (5), 420–437.

Osher, S.J., Paragios, N. (Eds.), 2003. Vision and Graphics. Springer, New York, NY.

Osher, S.J., Sethian, J., (Eds.), 1988. Fronts propagating with curvature dependent speed: algorithms based on the Hamilton–Jacobi formulation. J. Comput. Phys. 79, 12–49.

Park, M., Leey, S., Cheny, P.-C., Kashyap, S., Butty, A.A., Liu, Y., 2008. Performance evaluation of state-of-the-art discrete symmetry detection algorithms. Proceedings of the CVPR, 8 pp.

Parsons, C.J., Nixon, M.S., 1999. Introducing focus in the generalised symmetry operator. IEEE Signal Process. Lett. 6 (1), 49–51.

Peterfreund, N., 1999. Robust tracking of position and velocity. IEEE Trans. PAMI 21 (6), 564–569.

Press, W.H., Teukolsky, S.A., Vettering, W.T., Flannery, B.P., 1992. Numerical Recipes in C—The Art of Scientific Computing, second ed. Cambridge University Press, Cambridge.

Reisfeld, D., 1996. The constrained phase congruency feature detector: simultaneous localization, classification and scale determination. Pattern Recog. Lett. 17 (11), 1161–1169.

Reisfeld, D., Wolfson, H., Yeshurun, Y., 1995. Context-free attentional operators: the generalised symmetry transform. Int. J. Comput. Vision 14, 119–130.

Ronfard, R., 1994. Region-based strategies for active contour models. Int. J. Comput. Vision 13 (2), 229–251.

Sethian, J., 1999. Level Set Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press, New York, NY.

Siddiqi, K., Pizer, S. (Eds.), 2008. Medial Representations: Mathematics, Algorithms and Applications (Computational Imaging and Vision). Springer.

Siddiqi, K., Lauziere, Y., Tannenbaum, A., Zucker, S., 1998. Area and length minimizing flows for shape segmentation. IEEE Trans. IP 7 (3), 433–443.

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

Van Gool, L., Moons, T., Ungureanu, D., Oosterlinck, A., 1995. The characterisation and detection of skewed symmetry. Comput. Vision Image Underst. 61 (1), 138–150.

Van Otterloo, P.J., 1991. A Contour-Oriented Approach to Shape Analysis. Prentice Hall International (UK) Ltd., Hemel Hempstead.

Waite, J.B., Welsh, W.J., 1990. Head boundary location using snakes. Br. Telecom J. 8 (3), 127–136.

Wang, Y.F., Wang, J.F., 1992. Surface reconstruction using deformable models with interior and boundary constraints. IEEE Trans. PAMI 14 (5), 572–579.

Weickert, J., Ter Haar Romeny, B.M., Viergever, M.A., 1998. Efficient and reliable schemes for nonlinear diffusion filtering. IEEE Trans. IP 7 (3), 398–410.

Williams, D.J., Shah, M., 1992. A fast algorithm for active contours and curvature estimation. CVGIP: Image Underst. 55 (1), 14–26.

Xie, X., Mirmehdi, M., 2004. RAGS: region-aided geometric snake. IEEE Trans. IP 13 (5), 640–652.

Xu, C., Prince, J.L., 1998. Snakes, shapes, and gradient vector flow. IEEE Trans. IP 7 (3), 359–369.

Xu, C., Yezzi, A., Prince, J.L., 2000. On the relationship between parametric and geometric active contours and its applications. Proceedings of the 34th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, pp. 483–489.

Xu, G., Segawa, E., Tsuji, S., 1994. Robust active contours with insensitive parameters. Pattern Recog. 27 (7), 879–884.

Yuille, A.L., 1991. Deformable templates for face recognition. J. Cognitive Neurosci. 3 (1), 59–70.

Zabrodsky, H., Peleg, S., Avnir, D., 1995. Symmetry as a continuous feature. IEEE Trans. PAMI 17 (12), 1154–1166.