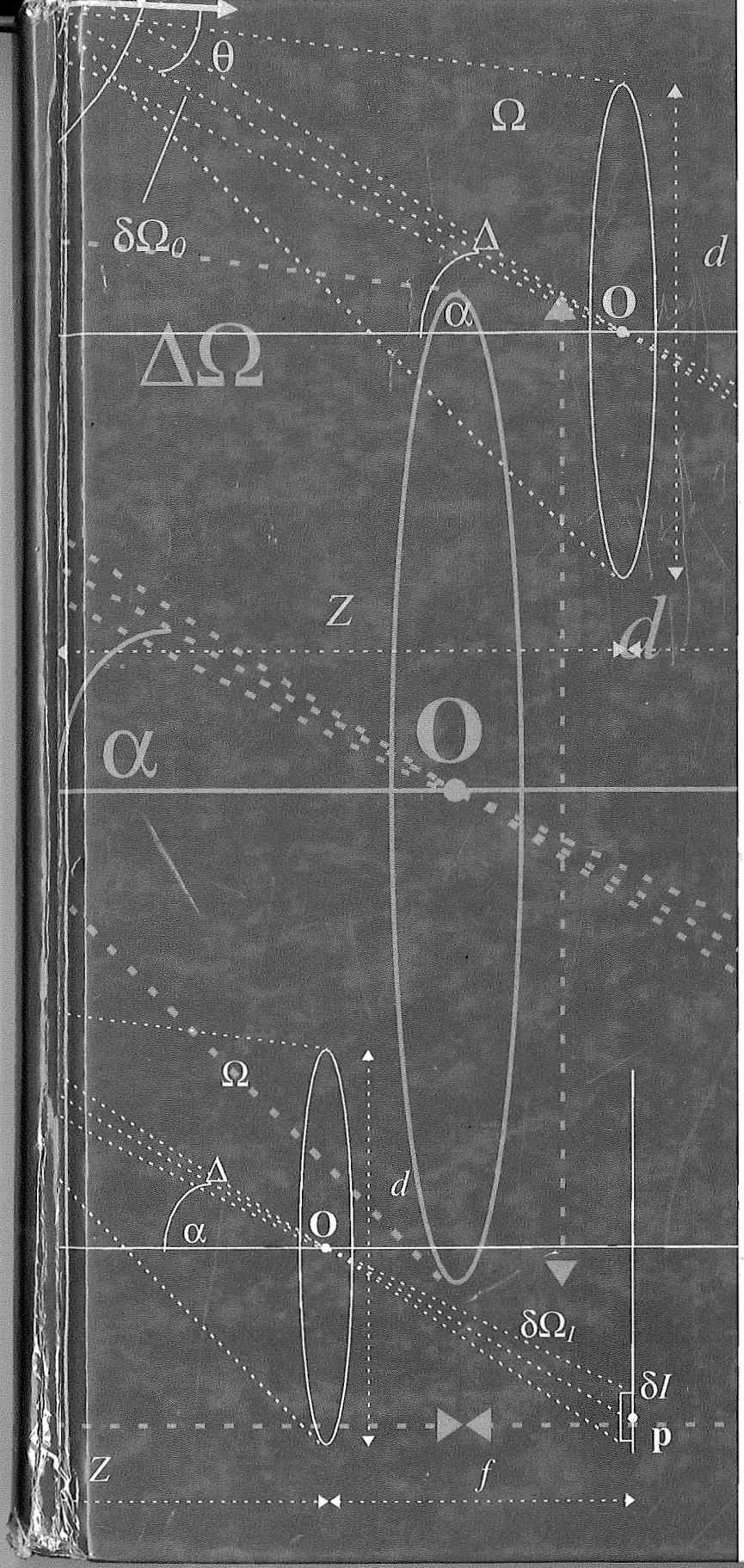
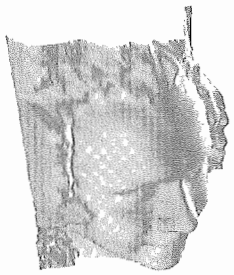


INTRODUCTORY TECHNIQUES *for* 3-D COMPUTER VISION



Emanuele Trucco
Alessandro Verri

5.4 Deformable Contours

Having discussed how to fit simple curves, we now move on to the general problem of fitting a curve of arbitrary shape to a set of image edge points. We shall deal with *closed contours* only.

A widely used computer vision model to represent and fit general, closed curves is the *snake*, or *active contour*, or again *deformable contour*. You can think of a snake as an elastic band of arbitrary shape, sensitive to the intensity gradient. The snake is located initially near the image contour of interest, and is attracted towards the target contour by forces depending on the intensity gradient.

☞ Notice that the snake is applied to the *intensity image*, not to an image of edge points as the line and ellipse detectors of the previous sections.

We start giving a description of the deformable contour model using the notion of *energy functional* and continuous image coordinates (no pixelization). We then discuss a simple, iterative algorithm fitting a deformable contour to a chain of egde points of a real, pixelized image.

5.4.1 The Energy Functional

The key idea of deformable contours is to associate an *energy functional* to each possible contour shape, in such a way that the image contour to be detected corresponds to a minimum of the functional. Typically, the energy functional used is a sum of several terms, each corresponding to some force acting on the contour.

Consider a contour, $\mathbf{c} = \mathbf{c}(s)$, parametrized by its arc length,[2] s. A suitable energy functional, $\mathcal{E}$, consists of the sum of three terms:

$$\mathcal{E} = \int \left(\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image}\right) ds, \tag{5.10}$$

where the integral is taken along the contour $\mathbf{c}$ and each of the energy terms, E_{cont}, E_{curv}, and E_{image}, is a function of $\mathbf{c}$ or of the derivatives of $\mathbf{c}$ with respect to s. The parameters α, β and γ control the relative influence of the corresponding energy term, and can vary along $\mathbf{c}$. Let us now define more precisely the three energy terms in (5.10).

5.4.2 The Elements of the Energy Functional

Each energy term serves a different purpose. The terms E_{cont} and E_{curv} encourage *continuity* and *smoothness* of the deformable contour, respectively; they can be regarded as a form of internal energy. E_{image} accounts for *edge attraction*, dragging the contour toward the closest image edge; it can be regarded as a form of external energy. What functions can achieve these behaviors?

Continuity Term. We can exploit simple analogies with physical systems to devise a rather natural form for the continuity term:

$$E_{cont} = \left\| \frac{d\mathbf{c}}{ds} \right\|^2 .$$

[2] Given an arbitrary parametrization of a curve, $\mathbf{c} = \mathbf{c}(t)$, with t the parameter and $0 \leq t \leq T$, the *arc length* s is defined as

$$s = \int_0^t \left\| \frac{d\mathbf{c}}{d\tau} \right\| d\tau.$$

In the arc length parametrization, the tangent vector $d\mathbf{c})ds$ is always a unit vector (Exercise 5.7).

In the discrete case, the contour $\mathbf{c}$ is replaced by a chain of N image points, $\mathbf{p}_1, \ldots, \mathbf{p}_N$, so that

$$E_{cont} = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|^2.$$

A better form for E_{cont}, preventing the formation of clusters of snake points, is

$$E_{cont} = \left(\bar{d} - \|\mathbf{p}_i - \mathbf{p}_{i-1}\|\right)^2, \tag{5.11}$$

with $\bar{d}$ the average distance between the pairs $(\mathbf{p}_i, \mathbf{p}_{i-1})$. If $\|\mathbf{p}_i - \mathbf{p}_{i-1}\| >> \bar{d}$, we have

$$E_{cont} \simeq \|\mathbf{p}_i - \mathbf{p}_{i-1}\|^2,$$

while for smaller distances (5.11) promotes the formation of equally spaced chains of points and avoids the formation of point clusters.

Smoothness Term. The aim of the smoothness term is to *avoid oscillations* of the deformable contour. This is achieved by introducing an energy term penalizing high contour curvatures. Since E_{cont} encourages equally spaced points on the contour, the curvature is well approximated by the second derivative of the contour (Exercise 5.8); hence, we can define E_{curv} as

$$E_{curv} = \|\mathbf{p}_{i-1} - 2\mathbf{p}_i + \mathbf{p}_{i+1}\|^2. \tag{5.12}$$

Edge Attraction Term. The third term corresponds to the energy associated to the external force attracting the deformable contour towards the desired image contour. This can be achieved by a simple function:

$$E_{image} = -\|\nabla I\|, \tag{5.13}$$

where ∇I is the spatial gradient of the intensity image I, computed at each snake point. Clearly, E_{image} becomes very small (negative) wherever the norm of the spatial gradient is large, (that is, near images edges), making $\mathcal{E}$ small and attracting the snake towards image contours. Note that E_{image}, unlike E_{cont} and E_{curv}, depends only on the contour, not on its derivatives with respect to the arc length.

5.4.3 A Greedy Algorithm

We are now ready to describe a method for fitting a snake to an image contour. The method is based on the minimization of the energy functional (5.10). First of all, let us summarize the assumptions and state the problem.

Assumptions

Let I be an image and $\bar{p}_1, \ldots, \bar{p}_N$ the chain of image locations representing the initial position of the deformable contour, which we assume close to the image contour of interest.

Problem Statement

Starting from $\bar{p}_1, \ldots, \bar{p}_N$, find the deformable contour $p_1, \ldots, p_N$ which fits the target image contour best, by minimizing the energy functional

$$\sum_{i=1}^{N} \left(\alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image} \right),$$

with $\alpha_i, \beta_i, \gamma_i \geq 0$, and E_{cont}, E_{curv}, and E_{image} as in (5.11), (5.12), and (5.13) respectively.

Of the many algorithms proposed to fit deformable contours, we have selected a *greedy algorithm*. A greedy algorithm makes *locally optimal choices*, in the hope that they lead to a *globally* optimal solution. Among the reasons for selecting the greedy algorithm instead of other methods, we emphasize its simplicity and low computational complexity.

The algorithm is *conceptually simple* because it does not require knowledge of the calculus of variations.[3] It has a *low computational complexity* because it converges in a number of iterations proportional to the number of contour points times the number of locations in which each point can move at each iteration, whereas other snake algorithms take much longer.

The core of a greedy algorithm for the computation of a deformable contour consists of two basic steps. First, at each iteration, each point of the contour is moved within a small neighborhood to the point which minimizes the energy functional. Second, before starting a new iteration, the algorithm looks for corners in the contour, and takes appropriate measures on the parameters $\beta_1, \ldots, \beta_N$ controlling E_{curv}. Let us discuss these two steps in more detail.

Step 1: Greedy Minimization. The neighborhood over which the energy functional is locally minimized is typically small (for instance, a 3×3 or 5×5 window centered at each contour point). Keeping the size of the neighborhood small lowers the computational load of the method (the complexity being linear in the size of the neighborhood). The local minimization is done by direct comparison of the energy functional values at each location.

Step 2: Corner Elimination. During the second step, the algorithm searches for corners as curvature maxima along the contour. If a curvature maximum is found at point $\mathbf{p}_j$, β_j is set to zero. Neglecting the contribution of E_{curv} at $\mathbf{p}_j$ makes it possible to keep the deformable contour piecewise smooth.

☞ For a correct implementation of the method, it is important to normalize the contribution of each energy term. For the terms E_{cont} and E_{curv}, it is sufficient to divide by the largest

[3] The calculus of variations is the mathematical technique for determining the minimum of a functional, in the same way as calculus provides the tools for determining the minimum of an ordinary function.

value in the neighborhood in which the point can move. For E_{image}, instead, it may be useful to normalize the norm of the spatial gradient, $\|\nabla I\|$, as

$$\frac{\|\nabla I\| - m}{M - m},$$

with M and m maximum and minimum of $\|\nabla I\|$ over the neighborhood, respectively.

The iterations stop when a predefined fraction of all the points reaches a local minimum; however, the algorithm's greed does not guarantee convergence to the global minimum. It usually works very well as far as the initialization is not too far from the desired solution. Let us enclose the algorithm details in the usual box.

Algorithm SNAKE

The input is formed by an intensity image, I, which contains a closed contour of interest, and by a chain of image locations, $\mathbf{p}_1, \ldots, \mathbf{p}_N$, defining the initial position and shape of the snake.

Let f be the minimum fraction of snake points that must move in each iteration before convergence, and $U(\mathbf{p})$ a small neighborhood of point $\mathbf{p}$. In the beginning, $\mathbf{p}_i = \bar{\mathbf{p}}_i$ and $d = \bar{d}$ (used in E_{cont}).

While a fraction greater than f of the snake points move in an iteration:

1. for each $i = 1, \ldots, N$, find the location of $U(\mathbf{p}_i)$ for which the functional $\mathcal{E}$ defined in (5.10) is minimum, and move the snake point $\mathbf{p}_i$ to that location;
2. for each $i = 1, \ldots, N$, estimate the curvature k of the snake at $\mathbf{p}_i$ as

$$k = |\mathbf{p}_{i-1} - 2\mathbf{p}_i + \mathbf{p}_{i+1}|,$$

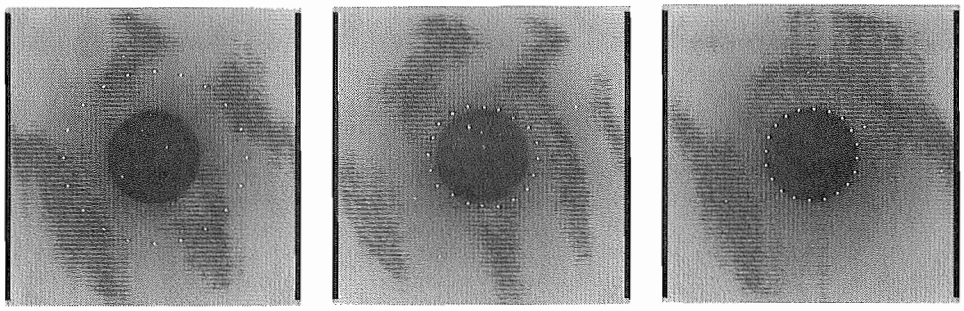
and look for local maxima. Set $\beta_j = 0$ for all $\mathbf{p}_j$ at which the curvature has a local maximum and exceeds a user-defined minimum value;
3. update the value of the average distance, $\bar{d}$.

On output this algorithm returns a chain of points $\mathbf{p}_i$ that represent a deformable contour.

☞ We still have to assign values to α_i, β_i, and γ_i. One possible choice is to initialize all of them to 1; another possibility is $\alpha_i = \beta_i = 1$ and $\gamma_i = 1.2$, which gives edges attraction more relevance in the minimization stage.

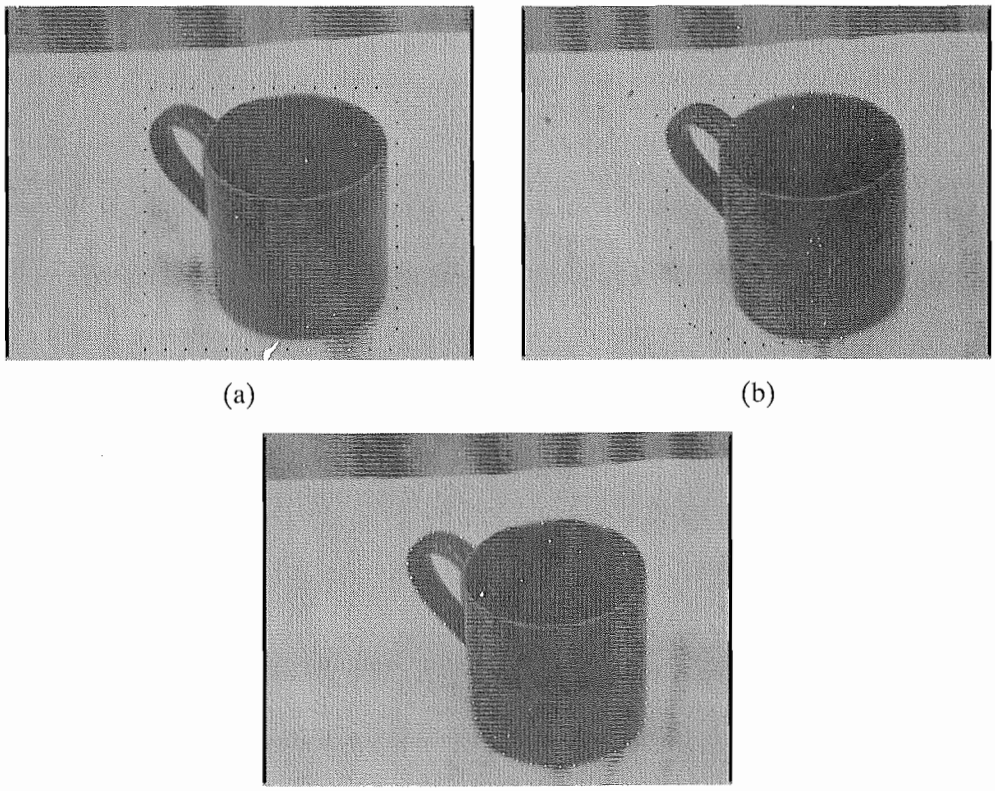
☞ To prevent the formation of noisy corners, it may be useful to add an extra condition: point $\mathbf{p}_j$ is a corner if and only if the curvature is locally maximum at $\mathbf{p}_j$ *and* the norm of the intensity gradient at $\mathbf{p}_j$ is sufficiently large. This ignores corners formed too far away from image edges.

Examples of the application of SNAKE to synthetic and real images are shown in Figures 5.7 and 5.8.



(a) (b) (c)

Figure 5.7 (a–c): Initial position of the snake, intermediate position (6th iteration), final result (20th iteration). Parameters were $\alpha_i = \beta_i = 1$, $\gamma_i = 1.2$. SNAKE used 7×7 local neighborhoods, and stopped when only 4 or fewer points changed in an iteration.



(a) (b)

(c)

Figure 5.8 (a): Initial position of the snake. (b): Intermediate position (84th iteration). (c): Final result (130th iteration). Parameters were $\alpha_i = \beta_i = 1$, $\gamma_i = 1.2$. SNAKE used 7×7 local neighborhoods, and stopped when only 9 or fewer points changed in an iteration.