CHAPTER 2 – SNAKES: ACTIVE CONTOUR MODELS

Active contour models – known colloquially as *snakes* – are energy-minimising curves that deform to fit image features. Snakes lock on to local minima in the *potential energy* generated by processing an image. (This energy is minimised by iterative *gradient descent*, moving the model according to equations of motion derived using the calculus of variations; a simple Euler time-stepping scheme can be used for this purpose, but a semi-implicit scheme is theoretically more stable.) In addition, *internal (smoothing) constraints* produce *tension* and *stiffness* that keep the model smooth and continuous, and prevent the formation of sharp corners; *external* constraints may be specified by a supervising process or a human user. A *pressure* term can also be added to make the models expand like balloons or bubbles.

Active contour models provide a unified solution to several image processing problems such as the detection of light and dark lines and edges; they are often used to segment spatial and temporal image sequences. Unfortunately, the energy minimisation process is prone to oscillation unless a very small time step is used, with the side-effect that convergence is slow. Other limitations are (i) that the models usually only incorporate edge information, ignoring other image characteristics such as texture and colour; and (ii) that they must be initialised close to the feature of interest to avoid being distracted by noise and clutter.

2.1 INTRODUCTION

As explained in Chapter 1, active contour models (snakes) are one example of the general technique of matching a deformable model to an image using energy minimisation. From any starting point, subject to certain constraints, a snake will deform into alignment with the nearest salient feature in an image; such features correspond to local minima in the energy generated by processing the image. In addition, high-level mechanisms can interact with snakes – for example, to guide them towards features of interest. Unlike most other techniques for segmenting images, snakes are always active; changes in high-level interpretation can therefore affect the energy minimisation process, and even in the absence of such changes the models will respond to moving image features.

(a) Initial Snake **(b) Final Snake**

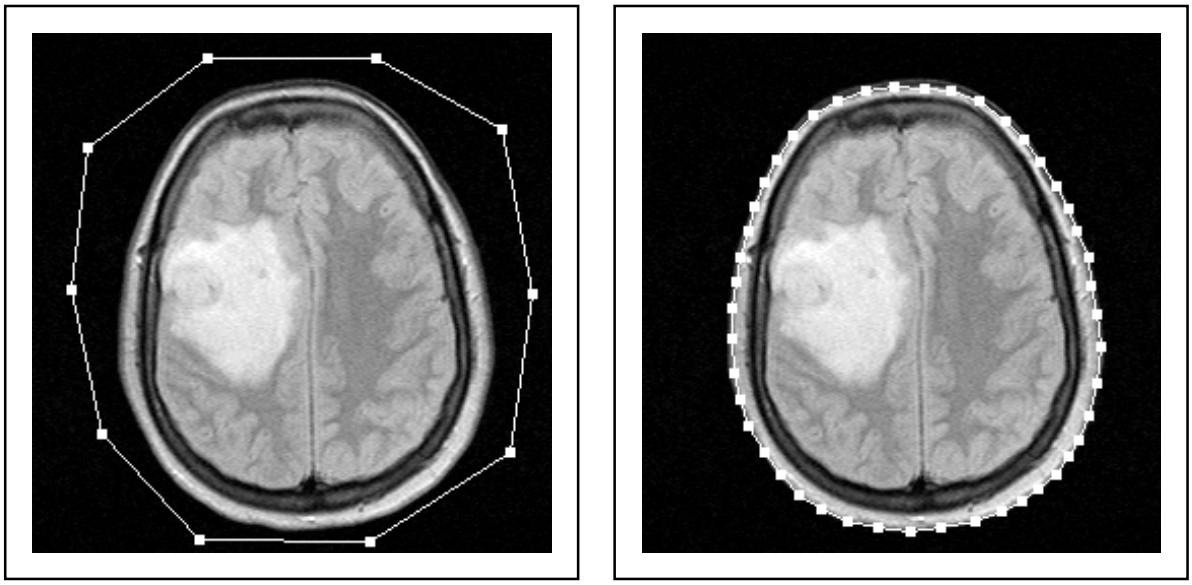

Figure 2.1.1: An Active Contour Model. This figure shows the NMR data from Figure 1.1.1; the potential energy generated by smoothing the image and convolving it with a gradient operator is shown in Figure 2.2.2. (**a**) The initial snake marked by the user. (**b**) The final snake, after energy minimisation, modelling the skin over the skull. (The model has been re-parameterised during energy minimisation, as explained in Chapter 4.)

Most snakes do not try to solve the entire problem of finding salient image features; they rely on other mechanisms to place them somewhere near a desired solution. For example, image processing techniques can be used to estimate the locations of interesting features, which are then refined using snakes. However, snakes can still be used for image interpretation in cases where automatic initialisation is not possible. An expert user need only place a snake near an

image feature, and the energy minimisation process will fit the model to the data. This behaviour has been exploited in numerous interactive image processing systems – for example, see **Porrill and Ivins (1994)**.

In addition to minimising the image energy, snakes must also satisfy some *internal constraints* – typically, they must remain smooth and continuous. Sometimes the user imposes additional *external constraints* such as making parts of the image attract or repel the models. Internal and external constraints are reviewed in Section 2.2, followed by some simple image energy terms. Energy minimisation is reviewed in Section 2.3. First, the *calculus of variations* is used to derive the *Euler-Lagrange equation* which is satisfied at minima in the energy of an active contour model. The nearest of these minima can be found by iterative gradient descent using a simple Euler time-stepping scheme; however, the original models proposed by **Kass et al (1987; 1988)** used a more stable semi-implicit scheme based on a fast matrix inversion algorithm. Section 2.4 explains how active contour models can be represented using B-spline basis functions. Section 2.5 describes how a pressure force can be added to make the models expand like balloons or bubbles. Finally, Section 2.6 discusses the main limitations of active contour models and suggests some ways to improve them.

2.2 SNAKE ENERGY FUNCTIONALS

A snake is an energy-minimising parametric contour that deforms over a series of time steps. Each element along the contour **u** depends on two parameters, where the space parameter s is taken to vary between 0 and N–1, and t is time (iteration):

$$\mathbf{u}(s,t) \quad = \quad (\, x(s,t), y(s,t)\,)$$

At a given time this function represents a mapping from the parametric domain $s \in [0, N)$ into the image plane $\mathbf{R}^2$. An image is regarded as an intensity function defined all over $\mathbf{R}^2$, with a value of zero outside some finite limits.

The total energy E_{snake} of the model is given by the sum of the energy for the individual snake elements:†

$$E_{\text{snake}} \quad = \quad \int_0^{N-1} E_{\text{element}}(\, \mathbf{u}(s,t)\,) \quad ds \tag{2.2.1}$$

The integral notation used in this equation implies an open-ended snake; however, joining the first and last elements makes a closed loop as shown in Figure 2.1.1.

The energy for each element can be decomposed into three basic energy terms:

$$E_{\text{element}} \quad = \quad E_{\text{internal}}(\mathbf{u}) \quad + \quad E_{\text{external}}(\mathbf{u}) \quad + \quad E_{\text{image}}(\mathbf{u}) \tag{2.2.2}$$

The parameters s and t are omitted where no ambiguity arises. The internal energy E_{internal} incorporates regularising constraints that give the model tension and stiffness. The external energy E_{external} represents external constraints imposed by high-level sources such as human operators or automatic initialisation procedures. The image (potential) energy E_{image} drives the model towards salient features; it is usually generated by processing the image to enhance light and dark lines, edges, or terminations. Each of these energy terms produces a corresponding force which can be used to move the model and hence minimise its energy as described in Section 2.3.

2.2.1 Internal Energy

The internal energy of a snake element is defined as:

$$E_{\text{internal}}(\mathbf{u}) \quad = \quad \underbrace{\alpha(s) \left| \frac{\partial \mathbf{u}}{\partial s} \right|^2}_{\text{Tension}} \quad + \quad \underbrace{\beta(s) \left| \frac{\partial^2 \mathbf{u}}{\partial s^2} \right|^2}_{\text{Stiffness}} \tag{2.2.1.1}$$

† To implement an active contour model in computer software the continuous representation must be approximated by N discrete snake elements; however, continuous notation is used wherever possible because of its greater mathematical elegance.

This energy contains a first-order term controlled by $\alpha(s)$, and a second-order term controlled by $\beta(s)$. Minimising the first-order energy term makes the snake contract by introducing *tension*; minimising the second-order term makes the snake resist bending by producing *stiffness*. In other words, the curve is predisposed to have minimal (preferably zero) velocity and acceleration with respect to the parameter *s*.

In the absence of other constraints, an active contour model simply collapses to a point like a strip of infinitely-elastic material; however, if the ends of the model are anchored then it forms a straight line along which its elements are evenly spaced. The weights $\alpha(s)$ and $\beta(s)$ control the relative importance of the tension and stiffness terms. For example, setting $\beta(s) = 0$ in one part of the model allows it to become second-order discontinuous and develop a corner. For simplicity, throughout this document the tension and stiffness weights are assumed to be uniform so that $\alpha(s) = \alpha$ and $\beta(s) = \beta$.

2.2.2 External Energy

Both automatic and manual supervision can be used to control external *attraction* and *repulsion* forces that drive active contour models towards or away from specific features.

A spring-like attractive force can be generated between a snake element **u** and a point **i** in an image using the following external energy term:

$$E_{\text{external}}(\mathbf{u}) \quad = \quad k\,|\mathbf{i}-\mathbf{u}|^2 \qquad (2.2.2.1)$$

This energy is zero when **u** = **i**, and takes the value of *k* when **i** – **u** = ±1 as in Figure 2.2.1.

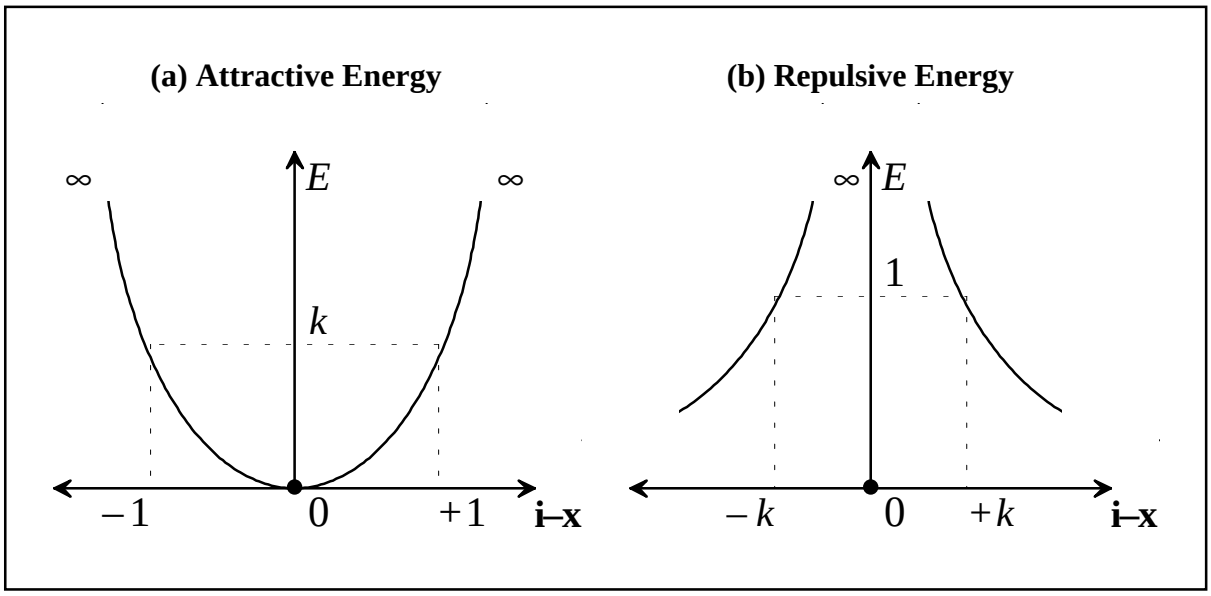

Figure 2.2.1: Attraction And Repulsion Energy. These graphs show key values of the attractive and repulsive energy terms. Both functionals have maximal values that are infinite; the minima are zero.

External energy can also be used to repel an active contour model:

$$E_{\text{external}}(\mathbf{u}) = \frac{k}{|\mathbf{i}-\mathbf{u}|^2} \qquad (2.2.2.2)$$

This energy is maximal (infinite) when $\mathbf{u} = \mathbf{i}$; it is unity when $\mathbf{i} - \mathbf{u} = \pm k$. Because of the singularity, the repulsion term must be clipped as the denominator approaches zero.

Negating the (positive) constant k in these equations converts attraction to pseudo-repulsion, and repulsion to pseudo-attraction; however, these pseudo energy terms are unusable because their minima are infinite. (During energy minimisation, these singularities completely dominate the behaviour of the model, at the expense of all other energy terms.) External energy terms are not considered further; however, the idea of a repulsive force is used in a different context to prevent boundary intersections in Chapter 4.

2.2.3 Image (Potential) Energy

A potential energy P generated by processing an image $I(x, y)$ is used to drive snakes towards features of interest. Two potential energy terms are described below; these drive snakes towards lines (regions) and edges; a third term for detecting corners (terminations) was proposed by **Kass et al (1987; 1988)** but will not be reviewed. The total image energy can be expressed as a weighted combination of these functionals:

$$P(\mathbf{u}) = E_{\text{image}}(\mathbf{u}) = w_{\text{line}}E_{\text{line}}(\mathbf{u}) + w_{\text{edge}}E_{\text{edge}}(\mathbf{u}) + w_{\text{term}}E_{\text{term}}(\mathbf{u}) \qquad (2.2.3.1)$$

The image energy is thus a linear combination of line and edge (and termination) energy terms, all computed from the raw image.

The effect of image energy is very local; however, if just a small portion of an active contour model finds a low-energy feature then the internal constraints will pull neighbouring elements towards that feature. This effect can be enhanced by spatially smoothing the potential energy and minimising it using scale-continuation – see **Witkin et al (1986)**. A snake can first be allowed to reach equilibrium on a very smooth potential; the blurring is then gradually reduced. At very coarse scales the model does a poor job of localising features, and fine detail is lost; however, it is attracted to local minima from far away. Reducing the amount of blurring allows the snake to form a more accurate model of the underlying image.

Line (Region) Functional. The simplest potential energy is the smoothed image intensity:

$$E_{\text{line}}(\mathbf{u}) \quad = \quad G_\sigma * I(\mathbf{u}) \tag{2.2.3.2}$$

The Gaussian filter G has standard deviation σ. According to the sign of w_{line} in Equation 2.2.3.1, the snake will be attracted either to light or to dark lines in the image.

Edge Functional. By far the most common use for active contour models is as semi-global edge-detectors that minimise a potential energy in which minima correspond to strong edges – see Figure 2.2.2. A snake can be given an affinity for edges using a gradient-based potential energy:†

$$E_{\text{edge}}(\mathbf{u}) \quad = \quad -\left| \frac{\partial}{\partial \mathbf{u}} G_\sigma * I(\mathbf{u}) \right|^2 \tag{2.2.3.3}$$

Unfortunately, the limitations of edge-based energy terms are well-known. The models often either fail to lock on to weak edges, or become distracted from the feature of interest by stronger but less interesting edges. These limitations initially lead to the development of the balloon models reviewed in Section 2.4, and later to the region models which are the subject of this document.

(a) Unprocessed Image **(b) Potential (Edge) Energy**

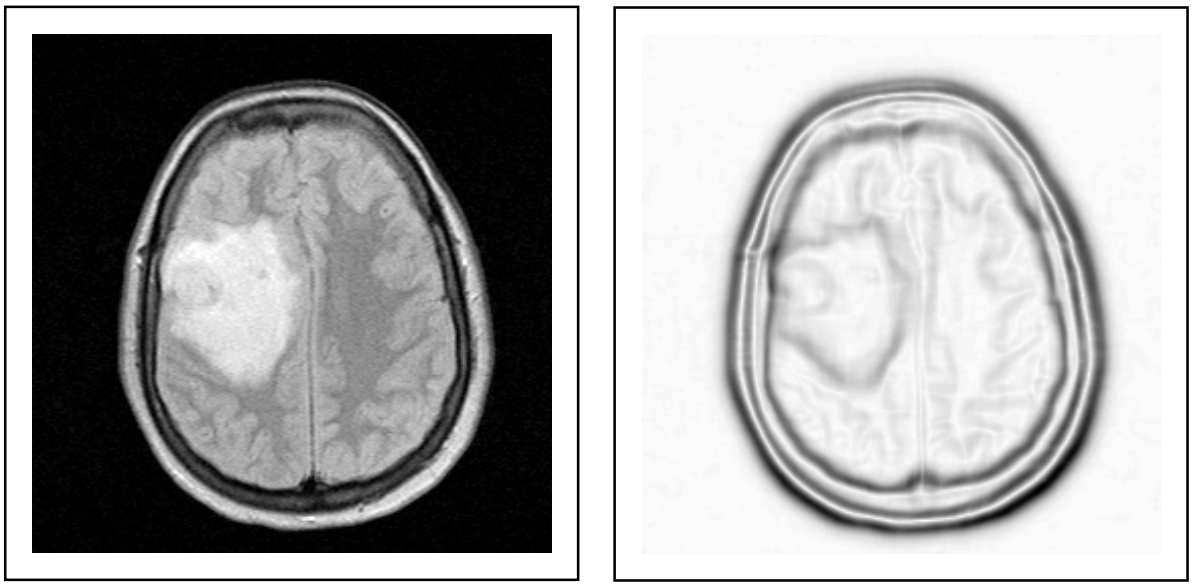

Figure 2.2.2: Potential (Edge) Energy. (**a**) The unprocessed 256-by-256 pixel NMR image from Figure 1.1.1. (**b**) The potential energy generated by smoothing the image, convolving it with a simple gradient operator, and negating the result (the image has been rescaled for display). Strong edges produce correspondingly low (dark) local minima; however, fine detail is lost during the smoothing process, which is necessary to eliminate noise and spread out edges.

† The Canny operator finds edges by applying non-maximal suppression to E_{edge}.

2.3 ENERGY MINIMISATION

This section reviews two related schemes for minimising the energy of active contour models. The nearest local minimum in the energy can be found using iterative gradient descent, moving the elements of the model according to either an Euler time-stepping scheme or the semi-implicit scheme proposed by **Kass et al (1987; 1988)**.

Given an energy functional $E(\mathbf{u})$ the change arising from a small change $\delta\mathbf{u}$ to the snake is approximated using the first variation of E with respect to $\mathbf{u}$:

$$\delta E = \int_0^{N-1} \frac{\delta E}{\delta \mathbf{u}} \cdot \delta\mathbf{u} \; ds \tag{2.3.1}$$

The *Euler Lagrange equation* describes the equilibrium condition which is satisfied at minima in the energy:

$$\frac{\delta E}{\delta \mathbf{u}} = 0 \tag{2.3.2}$$

Section 2.3.1 uses the *calculus of variations* to derive the Euler-Lagrange equation that describes the minimum energy condition for an active contour model. This is a complicated implicit equation which cannot be solved analytically; iterative schemes based on *gradient descent* are therefore used instead.

The *descent condition* states that the energy must decrease with each change $\delta\mathbf{u}$:

$$\delta E < 0 \tag{2.3.3}$$

This is achieved by movement in the *descent direction*:

$$\delta\mathbf{u} = \delta t \, \mathbf{f}(\mathbf{u}) \qquad \mathbf{f}(\mathbf{u}) \equiv -\frac{\delta E}{\delta \mathbf{u}} \tag{2.3.4}$$

Continuous movement in this direction is described by the *time evolution equation* for gradient descent which is derived for an active contour model in Section 2.3.2:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(\mathbf{u}) \tag{2.3.5}$$

This process can be speeded up using an appropriate positive definite matrix $\mathbf{S}$ (see Chapter 5) to give a 'modified' gradient descent method:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{S}(\mathbf{u}) \, \mathbf{f}(\mathbf{u}) \tag{2.3.6}$$

Equation 2.3.5 (or 2.3.6) can be integrated using various methods; for example, an Euler time-stepping scheme, or a semi-implicit scheme.†

The equation for the *Euler time-stepping scheme* is:

$$\frac{\mathbf{u}^{t+\delta t} - \mathbf{u}^t}{\delta t} = \mathbf{f}^t \quad \Rightarrow \quad \mathbf{u}^{t+\delta t} = \mathbf{u}^t + \delta t \, \mathbf{f}^t \tag{2.3.7}$$

† For an introduction to numerical methods see **Press et al (1992 B)**.

This scheme performs conventional gradient descent; it is simple to implement but is also theoretically unstable, particularly when large time steps are used. This method is considered in detail in Chapter 5 where it is used to drive active region models.

The equation for the *semi-implicit scheme* is:

$$\frac{\mathbf{u}^{t+\delta t} - \mathbf{u}^{t}}{\delta t} = \underbrace{\mathbf{f}_1^{t+\delta t}}_{\text{Implicit}} + \underbrace{\mathbf{f}_2^{t}}_{\text{Explicit}} \qquad (2.3.8)$$

The force is split into an image component which is evaluated at the current time step (the explicit part) and an internal component which is pushed forward in time (the implicit part). The semi-implicit scheme is computationally more expensive than the Euler time-stepping scheme, but offers improved stability with respect to internal forces; however, it still cannot prevent oscillation arising from image forces.

In both schemes the time step for iteration must be very small or instability and oscillation may result.

2.3.1 Euler-Lagrange Equations

As explained in Section 2.2 the basic active contour model $\mathbf{u}(s) = (x(s), y(s))$ is characterised by an energy functional that includes tension and stiffness terms, and a potential energy generated by processing an image:

$$E = \underbrace{\int_0^{N-1} P(\mathbf{u})\ ds}_{\text{Potential Energy}} + \underbrace{\frac{\alpha}{2}\int_0^{N-1} \left|\frac{\partial \mathbf{u}}{\partial s}\right|^2 ds}_{\text{Tension}} + \underbrace{\frac{\beta}{2}\int_0^{N-1} \left|\frac{\partial^2 \mathbf{u}}{\partial s^2}\right|^2 ds}_{\text{Stiffness}} \qquad (2.3.1.1)$$

For compactness, this functional can be rewritten as follows:

$$E = \int_0^{N-1} F(s, \mathbf{u}(s), \mathbf{u}'(s), \mathbf{u}''(s))\ ds \quad \begin{cases} \mathbf{u} \equiv \mathbf{u}(s) \\ \mathbf{u}' \equiv \partial\mathbf{u}/\partial s \\ \mathbf{u}'' \equiv \partial^2\mathbf{u}/\partial s^2 \end{cases} \qquad (2.3.1.2)$$

Making a small change $\delta\mathbf{u}$ to the vector $\mathbf{u}$ generates a corresponding change in E; this can be approximated using the Taylor expansion, ignoring terms above first order:

$$E + \delta E \approx \int_0^{N-1} \left(F + \frac{\partial F}{\partial \mathbf{u}} \cdot \delta\mathbf{u} + \frac{\partial F}{\partial \mathbf{u}'} \cdot \delta\mathbf{u}' + \frac{\partial F}{\partial \mathbf{u}''} \cdot \delta\mathbf{u}'' \right) ds \qquad (2.3.1.3)$$

Subtracting 2.3.1.2 from 2.3.1.3 gives the energy change:

$$\delta E \approx \int_0^{N-1} \frac{\partial F}{\partial \mathbf{u}} \cdot \delta\mathbf{u} + \frac{\partial F}{\partial \mathbf{u}'} \cdot \delta\mathbf{u}' + \frac{\partial F}{\partial \mathbf{u}''} \cdot \delta\mathbf{u}''\ ds \qquad (2.3.1.4)$$

Terms in $\delta\mathbf{u}'$ and $\delta\mathbf{u}''$ are eliminated using integration by parts:

$$\delta E \approx \int_0^{N-1} \frac{\partial F}{\partial \mathbf{u}} \cdot \delta\mathbf{u} - \frac{d}{ds}\left(\frac{\partial F}{\partial \mathbf{u}'}\right) \cdot \delta\mathbf{u} + \frac{d^2}{ds^2}\left(\frac{\partial F}{\partial \mathbf{u}''}\right) \cdot \delta\mathbf{u}\ ds \qquad (2.3.1.5)$$

Factorising:

$$\delta E \approx \int_0^{N-1} \left(\frac{\partial F}{\partial \mathbf{u}} - \frac{d}{ds}\left(\frac{\partial F}{\partial \mathbf{u}'} \right) + \frac{d^2}{ds^2}\left(\frac{\partial F}{\partial \mathbf{u}''} \right) \right) \cdot \delta \mathbf{u} \; ds \qquad (2.3.1.6)$$

At extrema in E a small change in $\mathbf{u}$ produces almost no change in the energy functional:

$$\int_0^{N-1} \left(\frac{\partial F}{\partial \mathbf{u}} - \frac{d}{ds}\left(\frac{\partial F}{\partial \mathbf{u}'} \right) + \frac{d^2}{ds^2}\left(\frac{\partial F}{\partial \mathbf{u}''} \right) \right) \cdot \delta \mathbf{u} \; ds \approx 0 \qquad (2.3.1.7)$$

The fact that $\delta\mathbf{u}$ is non-zero leads to the *Euler-Lagrange equation* for extrema in functionals of the form $F(s, \mathbf{u}(s), \mathbf{u}'(s), \mathbf{u}''(s))$.

$$\frac{\partial F}{\partial \mathbf{u}} - \frac{d}{ds}\left(\frac{\partial F}{\partial \mathbf{u}'} \right) + \frac{d^2}{ds^2}\left(\frac{\partial F}{\partial \mathbf{u}''} \right) = 0 \qquad (2.3.1.8)$$

This is a necessary condition for a local minimum in F.

The functional F can represent the potential energy, tension and stiffness of an active contour model as follows:

$$F = P(\mathbf{u}) + \frac{\alpha}{2} \mathbf{u}'^2 + \frac{\beta}{2} \mathbf{u}''^2 \qquad (2.3.1.9)$$

If the tension and stiffness parameters are constant then the partial derivatives for Equation 2.3.1.8 are as follows:

$$\frac{\partial F}{\partial \mathbf{u}} = \frac{\partial P}{\partial \mathbf{u}} \qquad \frac{\partial F}{\partial \mathbf{u}'} = \alpha \, \mathbf{u}' \equiv \alpha \, \frac{\partial \mathbf{u}}{\partial s} \qquad \frac{\partial F}{\partial \mathbf{u}''} = \beta \, \mathbf{u}'' \equiv \beta \, \frac{\partial^2 \mathbf{u}}{\partial s^2}$$

Substituting these derivatives gives the condition for extrema in the energy of an active contour model:

$$\frac{\delta E}{\delta \mathbf{u}} = \frac{\partial P}{\partial \mathbf{u}} - \alpha \frac{\partial^2 \mathbf{u}}{\partial s^2} + \beta \frac{\partial^4 \mathbf{u}}{\partial s^4} = 0 \qquad (2.3.1.10)$$

This equation is satisfied at all types of extrema: maxima, minima, and points of inflection. The second-order partial derivative (which will be positive at minima, negative at maxima, and zero at points of inflection) can be calculated to distinguish between these extrema. Unfortunately, Equation 2.3.1.10 cannot be solved analytically because $\mathbf{u}$ must be known before $\partial P/\partial \mathbf{u}$ can be found; the energy is therefore minimised using iterative gradient descent.

Two schemes are now considered for minimising this energy: a simple Euler time-stepping scheme, and a semi-implicit scheme which is theoretically more stable.

2.3.2 Euler Time-Stepping Scheme

A snake energy functional $E(\mathbf{u})$ can be minimised by altering $\mathbf{u}$ by an amount $\delta\mathbf{u}$ that is guaranteed to reduce the value of the functional:

$$\mathbf{u} \rightarrow \mathbf{u} + \delta\mathbf{u} \qquad (2.3.2.1)$$

Local linear approximation based on the appropriate functional derivative (first variation) gives an expression for the new energy:

$$E(\mathbf{u}+\delta\mathbf{u}) \quad \approx \quad E(\mathbf{u}) \quad + \quad \int_0^{N-1} \frac{\delta E}{\delta \mathbf{u}} \cdot \delta\mathbf{u} \tag{2.3.2.2}$$

Steps down the energy hypersurface (see Figure 2.3.1) can be guaranteed by making small changes in the direction of the negated gradient:

$$\delta\mathbf{u} \quad \propto \quad -\frac{\delta E}{\delta \mathbf{u}} \tag{2.3.2.3}$$

Substituting this expression back into Equation 2.3.2.2, and introducing a time step δt, gives the new value of the energy functional:

$$E(\mathbf{u}+\delta\mathbf{u}) \quad \approx \quad E(\mathbf{u}) \quad - \quad \delta t \int_0^{N-1} \left(\frac{\delta E}{\delta \mathbf{u}}\right)^2 \quad ds \tag{2.3.2.4}$$

The negative sign and square power guarantee that E will decrease at each iteration until the minimum is reached; however, the time step must be small enough to prevent oscillation (see Chapter 5) and is almost invariably less than unity.

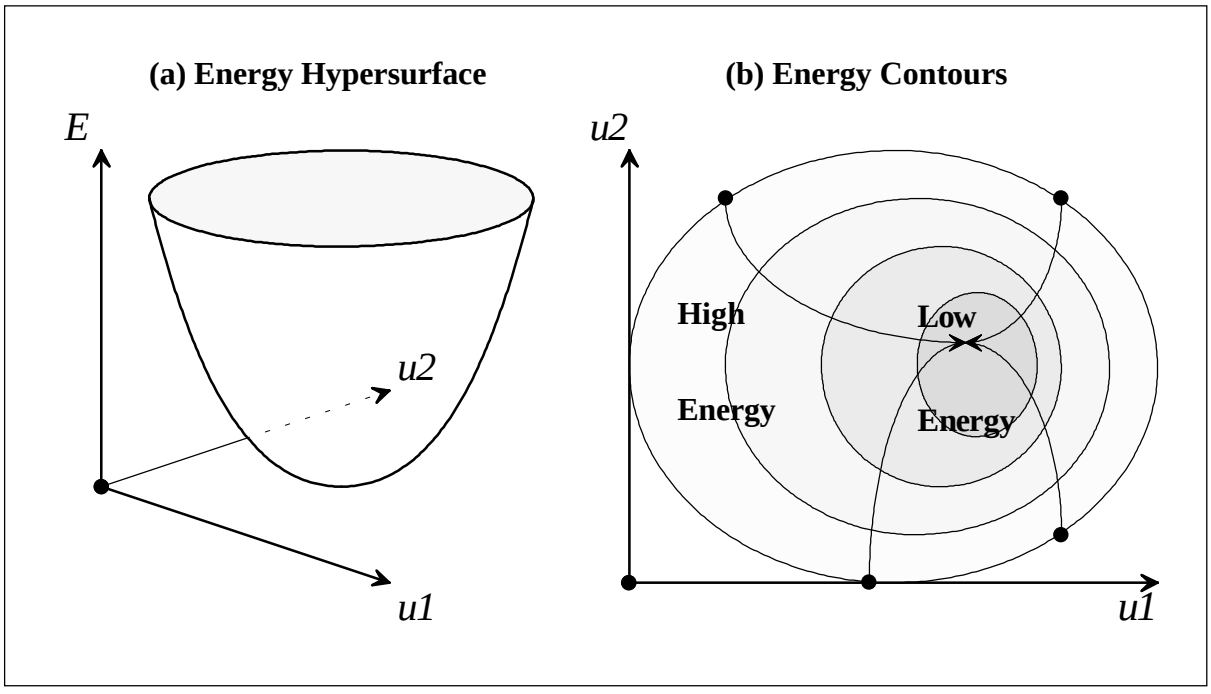

Figure 2.3.1: Gradient Descent. This diagram shows four alternative paths down a simple three-dimensional energy surface. At each iteration the gradient descent algorithm moves the energy value towards the nearest local minimum by making a small change in the direction given by the negated energy gradient (orthogonal to the local energy contours). The process is repeated until the gradient is zero.

From Equation 2.3.1.10 the equations of motion for minimising the basic snake energy functional by iterative gradient descent are:

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{\delta E}{\delta \mathbf{u}} = \alpha \frac{\partial^2 \mathbf{u}}{\partial s^2} - \beta \frac{\partial^4 \mathbf{u}}{\partial s^4} - \frac{\partial P}{\partial \mathbf{u}} \qquad (2.3.2.5)$$

These equations can be used to implement an Euler time-stepping scheme in which each element in the model is moved according to the local forces (RHS) acting on it. This method has the advantage of being computationally cheap and simple to implement; it also allows for boundary collision detection and re-parameterisation as discussed in Chapter 4. The main problem with the method is that a small time step is necessary if oscillation is to be avoided.

2.3.3 Semi-Implicit Scheme

As an alternative to the Euler time-stepping scheme, a semi-implicit scheme can be used to solve Equations 2.3.3.1 for all elements of the snake simultaneously. After converting to discrete notation, where $\mathbf{u}_s$ is an approximation to $\mathbf{u}(s)$ such that $s = 0, 1, \ldots, N-1$, the necessary derivatives are approximated using the finite difference formulae given in Appendix C. The x and y components of the vector terms are treated separately and are represented by u:

$$\frac{\partial u}{\partial t} \rightarrow \frac{u_s^{t+\delta t} - u_s^t}{\delta t} \qquad \frac{\partial^2 u}{\partial s^2} \rightarrow \frac{u_{s+\delta s}^{t+\delta t} + u_{s-\delta s}^{t+\delta t} - 2u_s^{t+\delta t}}{\delta s^2}$$

$$\frac{\partial^4 u}{\partial s^4} \rightarrow \frac{1}{\delta s^4}\left(u_{s+2\delta s}^{t+\delta t} - 4u_{s+\delta s}^{t+\delta t} + 6u_s^{t+\delta t} - 4u_{s-\delta s}^{t+\delta t} + u_{s-2\delta s}^{t+\delta t}\right)$$

Superscript t is used to denote time (iteration); the second and fourth derivatives are estimated as though at the *next* time step†; for simplicity, the space step δs is taken to be unity.

Combining the discrete approximations gives the following finite difference equation:

$$\frac{u_s^{t+\delta t} - u_s^t}{\delta t} = \alpha\left(u_{s+1}^{t+\delta t} + u_{s-1}^{t+\delta t} - 2u_s^{t+\delta t}\right) - \beta\left(u_{s+2}^{t+\delta t} - 4u_{s+1}^{t+\delta t} + 6u_s^{t+\delta t} - 4u_{s-1}^{t+\delta t} + u_{s-2}^{t+\delta t}\right) - \frac{\partial P}{\partial u_s^t} \qquad (2.3.3.1)$$

† This mathematical trick produces a set of equations that describe the behaviour of the model over time. The snake is first moved according to the image forces at the current time step; the resulting model is then smoothed immediately at the start of the next iteration. At equilibrium these processes cancel out. The scheme is semi-implicit because some terms are evaluated at the current iteration while others are not.

Moving terms that cannot be estimated at time t over to the LHS gives:

$$bu_{s+2}^{t+\delta t} \quad - \quad (a+4b)u_{s+1}^{t+\delta t} \quad + \quad (1+2a+6b)u_s^{t+\delta t} \quad - \quad (a+4b)u_{s-1}^{t+\delta t} \quad + \quad bu_{s-2}^{t+\delta t}$$

$$= \quad u_s^t \quad + \quad \delta t \, \frac{\partial P}{\partial u_s^t} \qquad \text{Note:} \quad \begin{cases} a & \equiv & \alpha \, \delta t/\delta s^2 \\ b & \equiv & \beta \, \delta t/\delta s^4 \end{cases} \tag{2.3.3.2}$$

The RHS of Equation 2.3.3.3 can be evaluated using the potential energy at time t:

$$pu_{s+2}^{t+\delta t} + qu_{s+1}^{t+\delta t} + ru_s^{t+\delta t} + qu_{s-1}^{t+\delta t} + pu_{s-2}^{t+\delta t} \quad = \quad \tilde{u}_s^{t+\delta t} \quad \begin{cases} p & \equiv & b \\ q & \equiv & -a-4b \\ r & \equiv & 1+2a+6b \end{cases}$$

$$\tilde{u}_s^{t+\delta t} \quad \equiv \quad u_s^t \quad + \quad \delta t \, \frac{\partial P}{\partial u_s^t} \tag{2.3.3.3}$$

This equation leads to two sets of N simultaneous linear equations (for the x and y co-ordinates of each element in the snake) that can be written using a smoothing matrix $\mathbf{M}$ which is symmetric and pentadiagonal. If $\mathbf{u}_0 = \mathbf{u}_N$ so the model forms a closed loop then $\mathbf{M}$ is also cyclic:

$$\begin{bmatrix} r & q & p & & & p & q \\ q & r & q & p & & & p \\ p & q & r & q & p & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & p & q & r & q & p \\ p & & & p & q & r & q \\ q & p & & & p & q & r \end{bmatrix} \begin{pmatrix} u_0^{t+\delta t} \\ u_1^{t+\delta t} \\ u_2^{t+\delta t} \\ \vdots \\ u_{N-3}^{t+\delta t} \\ u_{N-2}^{t+\delta t} \\ u_{N-1}^{t+\delta t} \end{pmatrix} = \begin{pmatrix} \tilde{u}_0^{t+\delta t} \\ \tilde{u}_1^{t+\delta t} \\ \tilde{u}_2^{t+\delta t} \\ \vdots \\ \tilde{u}_{N-3}^{t+\delta t} \\ \tilde{u}_{N-2}^{t+\delta t} \\ \tilde{u}_{N-1}^{t+\delta t} \end{pmatrix} \tag{2.3.3.4}$$

Each row of the matrix can be thought of as a convolution mask for evaluating the derivatives; the constant values making up the matrix are as follows:

$$p \quad \equiv \quad \beta \, \frac{\delta t}{\delta s^4} \qquad q \quad \equiv \quad -\alpha \, \frac{\delta t}{\delta s^2} - 4\beta \, \frac{\delta t}{\delta s^4} \qquad r \quad \equiv \quad 1 + 2\alpha \, \frac{\delta t}{\delta s^2} + 6\beta \, \frac{\delta t}{\delta s^4}$$

Equation 2.3.3.4 can be written more compactly:

$$\mathbf{M} \, \mathbf{x}^{t+\delta t} \quad = \quad \tilde{\mathbf{x}}^{t+\delta t} \quad = \quad \mathbf{x}^t \quad + \quad \delta t \, \frac{\partial P}{\partial \mathbf{x}^t}$$

$$\mathbf{M} \, \mathbf{y}^{t+\delta t} \quad = \quad \tilde{\mathbf{y}}^{t+\delta t} \quad = \quad \mathbf{y}^t \quad + \quad \delta t \, \frac{\partial P}{\partial \mathbf{y}^t} \tag{2.3.3.5}$$

The vectors $\mathbf{x}$ and $\mathbf{y}$ contain the x and y co-ordinates respectively of each element in the model; they represent the positions of the snake elements, both before and after adjustment to conform with the internal forces.

Multiplying both sides of Equation 2.3.3.5 by the inverse of $\mathbf{M}$ gives the final solution:

$$\mathbf{x}^{t+\delta t} \quad = \quad \mathbf{M}^{-1}\left(\mathbf{x}^t + \delta t \, \frac{\partial P}{\partial \mathbf{x}^t}\right) \qquad \mathbf{y}^{t+\delta t} \quad = \quad \mathbf{M}^{-1}\left(\mathbf{y}^t + \delta t \, \frac{\partial P}{\partial \mathbf{y}^t}\right) \tag{2.3.3.6}$$

This equation provides an iterative method for minimising the energy of an active contour model. Each iteration is implicit with respect to the internal energy, and explicit with respect to the external and image energy. The minimisation process is therefore stable in the presence of high tension and stiffness. Furthermore, with ordinary relaxation methods the propagation of forces along a snake is slow; however, the semi-implicit procedure allows forces to travel arbitrary distances in a single iteration.

The pentadiagonal matrix can be inverted using the algorithm described by **Benson and Evans (1973; 1977)** making the solution of Equation 2.3.3.6 an O(N) process rather than O(N^3). If the tension and stiffness parameters and the number of elements are all constant then the inverse matrix need only be calculated once because it too will be constant; however, the matrix must be recomputed and inverted if the smoothing parameters are changed either locally or globally, and the same is true if the model is re-parameterised (the matrix has one row and one column for each element in the model). Another limitation is that the elements of the model cannot be moved individually and so hard constraints, such as not allowing the model to self-intersect, cannot be imposed using this method. (These issues are dealt with in more detail in Chapters 4 and 5.) For these reasons an Euler time-stepping scheme is used to move elements individually in the chapters that follow.

2.4 B-SPLINE MODELS

This section reviews two snake models implemented using B-splines: B-snakes and dynamic contours. Both models incorporate the main advantages of B-splines – compact representation, implicit smoothness, and local control – and can deal with corners by varying the degree of continuity between adjacent spline sections. Appendix B provides a brief introduction to cubic B-splines.

Let s be the (space) parameter describing a uniform B-spline $\mathbf{q}(s) = (x(s), y(s))$ that is to represent a snake approximated in discrete form by N elements such that $s = 0, 1, \ldots, N–1$. Each spline section is a polynomial of order K obtained from a linear combination of M basis functions $B_i(v_s)$ and control points $\mathbf{c}_i = (x_i, y_i)$ where $i = 0, 1, ..., M–1$:

$$\mathbf{q}(s) = \sum_{i=0}^{M-1} B_i(v_s)\, \mathbf{c}_i \qquad v_s = s\,\frac{M-1}{N-1} \tag{2.4.1}$$

Thus s naturally parameterises the spline elements, while v_s parameterises the control points.

The entire model can be represented using a banded matrix containing spline basis function values, with one row for each element; q and c indicate the x or y co-ordinates of the elements and controls respectively:

$$\underbrace{\begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_{N-1} \end{pmatrix}}_{\text{Elements}} = \underbrace{\begin{bmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,M-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N-1,0} & b_{N-1,1} & \cdots & b_{N-1,M-1} \end{bmatrix}}_{\text{Basis Functions}} \underbrace{\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{M-1} \end{pmatrix}}_{\text{Controls}} \tag{2.4.2}$$

This matrix maps control points to spline elements as shown in Figure 2.4.1a; its inverse does the reverse.† Most of the (constant) entries in the matrix are zero (it is banded).

2.4.1 B-Snakes

Menet et al (1990; 1991), and **Saint-Marc and Medioni (1990)** used parametric B-spline approximation to implement a new model, called a *B-snake*.

† In fact these formulae are quite general; they are valid for arbitrary linear algebra bases such as the Fourier representation.

Substituting $\mathbf{q}(s)$ for $\mathbf{u}(s)$ in the basic snake energy equation yields a system in which the unknowns are the M control points, rather than the N elements of the snake:

$$E = \sum_{s=0}^{N-1} P\left(\sum_{i=0}^{M-1} \mathbf{c}_i B_i(v_s)\right) + \sum_{s=0}^{N-1}\left(\frac{\alpha}{2}\left|\sum_{i=0}^{M-1} \mathbf{c}_i B_i'(v_s)\right|^2 + \frac{\beta}{2}\left|\sum_{i=0}^{M-1} \mathbf{c}_i B_i''(v_s)\right|^2\right) \quad (2.4.1.1)$$

The first and second derivatives in this equation, shown using dashes, are calculated from the usual spline derivatives (see Appendix B):

$$B'(v) \equiv \frac{\partial B}{\partial v} \qquad B''(v) \equiv \frac{\partial^2 B}{\partial v^2}$$

To minimise the energy of the model it is necessary to find control points $\mathbf{c}_j$ such that:

$$\forall \quad j \in (0, 1, ..., M-1) \quad \left\{\frac{\partial E}{\partial \mathbf{c}_j} = 0 \right. \quad (2.4.1.2)$$

Note the following partial derivatives obtained from Equation 2.4.1.1:

$$\frac{\partial}{\partial \mathbf{c}_j}\left(\sum_{i=0}^{M-1} \mathbf{c}_i B_i'(v_s)\right) = B_j'(v_s) \qquad \frac{\partial}{\partial \mathbf{c}_j}\left(\sum_{i=0}^{M-1} \mathbf{c}_i B_i''(v_s)\right) = B_j''(v_s)$$

The minimum energy condition (Equation 2.3.2) therefore yields:

$$\frac{\delta E}{\delta \mathbf{c}_j} = \sum_{s=0}^{N-1}\left(\alpha B_j'(v_s)\sum_{i=0}^{M-1} \mathbf{c}_i B_i'(v_s) + \beta B_j''(v_s)\sum_{i=0}^{M-1} \mathbf{c}_i B_i''(v_s)\right) + \sum_{s=0}^{N-1} B_j(v_s)\frac{\partial}{\partial \mathbf{c}_j} P\left(\sum_{i=0}^{M-1} \mathbf{c}_i B_i(v_s)\right) = 0 \quad (2.4.1.3)$$

The force term $\mathbf{f}$ generated from the image energy P can be written as:

$$\mathbf{f}_j \equiv -\sum_{s=0}^{N-1} B_j(v_s)\frac{\partial}{\partial \mathbf{c}_j} P(\mathbf{q}(s))$$

Equation 2.4.2.5 can therefore be rewritten as:

$$\sum_{i=0}^{M-1} \mathbf{c}_i\left(\alpha \sum_{s=0}^{N-1} B_j'(v_s)B_i'(v_s) + \beta \sum_{s=0}^{N-1} B_j''(v_s)B_i''(v_s)\right) - \mathbf{f}_j = 0 \quad (2.4.1.4)$$

This system of M equations – where $j = 0, 1, ..., M–1$ – can be written using a banded matrix $\mathbf{B}$ of spline basis functions and column matrices $\mathbf{c}_x$ and $\mathbf{c}_y$ containing the x and y co-ordinates respectively of the M unknown control points:

$$\mathbf{B}\,\mathbf{c}_x - \mathbf{f}_x = 0 \qquad \mathbf{B}\,\mathbf{c}_y - \mathbf{f}_y = 0 \quad (2.4.1.5)$$

The vectors $\mathbf{f}_x$ and $\mathbf{f}_y$ contain the x and y components of the forces acting on the controls, computed from the energy gradients at the elements.

The equations of motion for gradient descent are:

$$\frac{\partial \mathbf{c}}{\partial t} \quad = \quad -\frac{\delta E}{\delta \mathbf{c}} \quad = \quad \mathbf{f} - \mathbf{B}\,\mathbf{c}$$

Equations 2.4.1.5 can therefore be solved using a semi-implicit scheme:

$$\mathbf{c}_{\mathbf{x}}^{t} - \mathbf{c}_{\mathbf{x}}^{t-1} \quad = \quad \delta t\,(\mathbf{f}_{\mathbf{x}}^{t-1} - \mathbf{B}\,\mathbf{c}_{\mathbf{x}}^{t}) \qquad\qquad \mathbf{c}_{\mathbf{y}}^{t} - \mathbf{c}_{\mathbf{y}}^{t-1} \quad = \quad \delta t\,(\mathbf{f}_{\mathbf{y}}^{t-1} - \mathbf{B}\,\mathbf{c}_{\mathbf{y}}^{t}) \qquad (2.4.1.6)$$

Rearranging:

$$(\delta t\,\mathbf{B} + \mathbf{I})\,\mathbf{c}_{\mathbf{x}}^{t} \quad = \quad \mathbf{c}_{\mathbf{x}}^{t-1} + \delta t\,\mathbf{f}_{\mathbf{x}}^{t-1} \qquad\qquad (\delta t\,\mathbf{B} + \mathbf{I})\,\mathbf{c}_{\mathbf{y}}^{t} \quad = \quad \mathbf{c}_{\mathbf{y}}^{t-1} + \delta t\,\mathbf{f}_{\mathbf{y}}^{t-1} \qquad (2.4.1.7)$$

This system can be solved using matrix inversion:

$$\mathbf{c}_{\mathbf{x}}^{t} \quad = \quad (\delta t\,\mathbf{B} + \mathbf{I})^{-1}\,(\mathbf{c}_{\mathbf{x}}^{t-1} + \delta t\,\mathbf{f}_{\mathbf{x}}^{t-1}) \qquad\qquad \mathbf{c}_{\mathbf{y}}^{t} \quad = \quad (\delta t\,\mathbf{B} + \mathbf{I})^{-1}\,(\mathbf{c}_{\mathbf{y}}^{t-1} + \delta t\,\mathbf{f}_{\mathbf{y}}^{t-1}) \qquad (2.4.1.8)$$

This equation represents an iterative semi-implicit energy minimisation scheme based on propagating $\mathbf{c}^{t-1}$ forward in time. The matrix $(\delta t\,\mathbf{B} + \mathbf{I})^{-1}$ is banded, so it can be inverted in O(*N*) time.

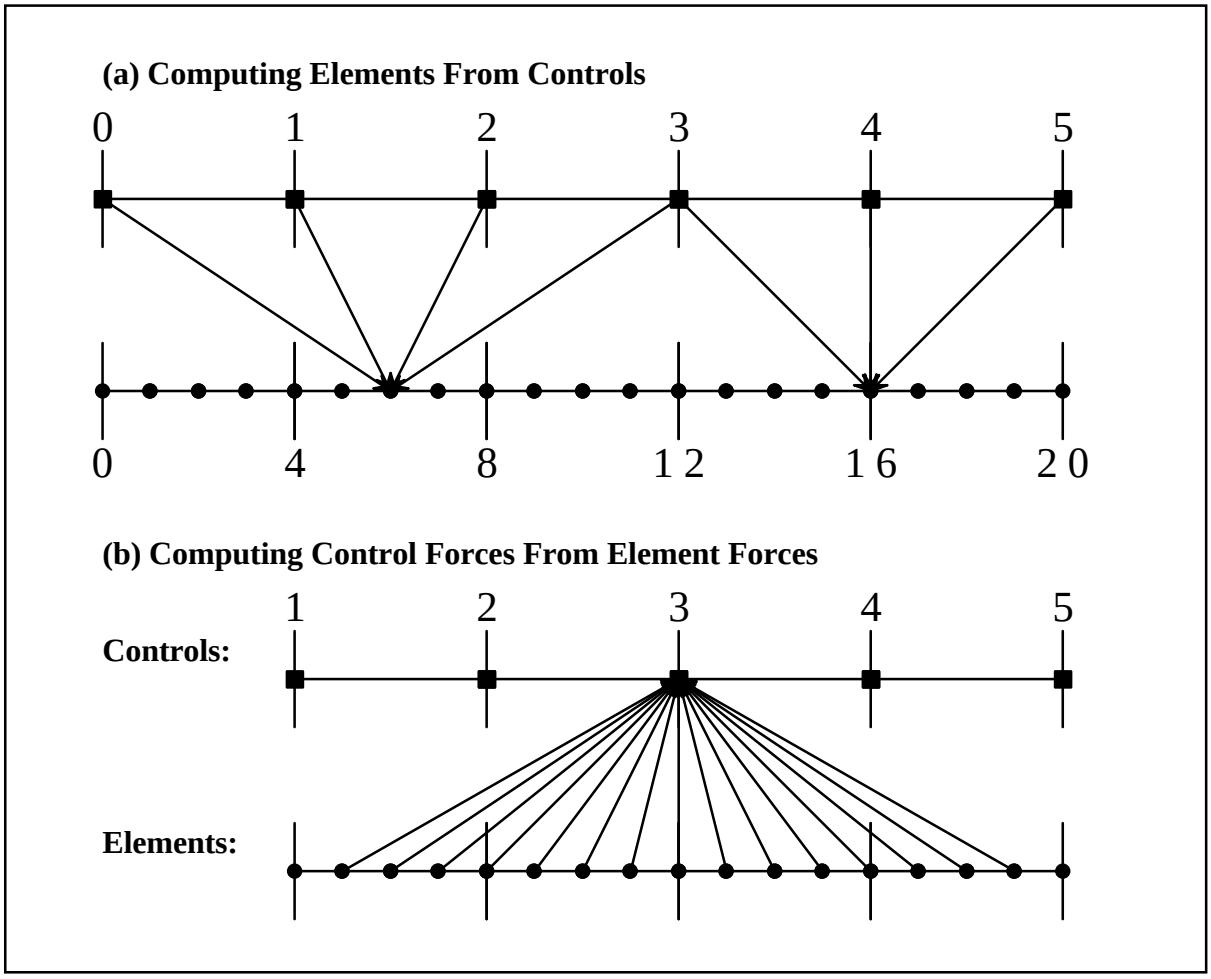

Figure 2.4.1: Computing Elements And Forces. (**a**) Each element in a cubic spline is influenced by four control points (except for elements lying on span boundaries, which have only three controls). (**b**) The force acting on a control point is computed from the forces at each of the elements to which the control contributes.

The original B-snake model behaves in a similar way to the active contour model from which it evolved, although a B-snake tends to be slightly smoother, and to require fewer points than the equivalent explicit representation. The main theoretical advantage of using B-splines is that there are fewer equations to solve (*M* controls rather than *N* elements); however, the models are more difficult to implement.

2.4.2 Dynamic Contours

Dynamic contours, like B-snakes, are defined parametrically using B-splines – see **Cipolla and Blake (1990)**, **Curwen et al (1991)**, and **Curwen and Blake (1992)**. However, unlike B-snakes, dynamic contours are not constrained by tension or stiffness energy terms because smoothness is implicitly guaranteed by the spline formulation. Unconstrained dynamic contours therefore run very quickly because, in addition to requiring fewer equations than active contour models, there is no need to calculate smoothing forces; they are therefore often the model of choice for real-time applications.

The energy of a closed dynamic contour $\mathbf{q}(s)$ is approximated discretely as:

$$E \quad = \quad \sum_{s=0}^{N-1} P(\ \mathbf{q}(s)\ ) \quad = \quad \sum_{s=0}^{N-1} P\left(\sum_{i=0}^{M-1} B_i(v_s)\ \mathbf{c}_i \right) \qquad (2.4.2.1)$$

The force (for gradient descent) acting on control point j is:

$$\delta\mathbf{c}_j \quad = \quad -\frac{\partial E}{\partial \mathbf{c}_j} \quad = \quad -\frac{\partial}{\partial \mathbf{c}_j} \sum_{s=0}^{N-1} P(\ \mathbf{q}(s)\ ) \qquad (2.4.2.2)$$

Differentiating with the product rule, and using $\delta\mathbf{q}(s)$ to represent the force on each element:

$$\delta\mathbf{c}_j \quad = \quad -\sum_{s=0}^{N-1} \frac{\partial P}{\partial \mathbf{q}(s)} \cdot \frac{\partial \mathbf{q}(s)}{\partial \mathbf{c}_j} \quad = \quad -\sum_{s=0}^{N-1} \delta\mathbf{q}(s)\ B_j(v_s) \qquad \delta\mathbf{q} \equiv \frac{\partial P}{\partial \mathbf{q}(s)} \qquad (2.4.2.3)$$

The forces acting on a dynamic contour can be represented using a banded matrix of spline basis functions with one row for each element as in Equation 2.4.2; δq and δc represent changes to the x or y co-ordinates of the elements and controls respectively:

$$\begin{pmatrix} \delta c_0 \\ \delta c_1 \\ \vdots \\ \delta c_{M-1} \end{pmatrix} \quad = \quad \mathbf{B}^{\mathrm{T}} \begin{pmatrix} \delta q_0 \\ \delta q_1 \\ \vdots \\ \delta q_{N-1} \end{pmatrix} \qquad \mathbf{B} \equiv \begin{bmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,M-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N-1,0} & b_{N-1,1} & \cdots & b_{N-1,M-1} \end{bmatrix} \qquad (2.4.2.4)$$

Alternatively:

$$\delta\mathbf{c}_\mathbf{x} = \mathbf{B}^{\mathrm{T}} \delta\mathbf{q}_\mathbf{x} \qquad \delta\mathbf{c}_\mathbf{y} = \mathbf{B}^{\mathrm{T}} \delta\mathbf{q}_\mathbf{y} \qquad (2.4.2.5)$$

The force at a given control point is therefore calculated from the forces acting on the elements within the span of that control (Figure 2.4.1b).

Unconstrained dynamic contours are free to deform and, because there are no tension or stiffness terms, have no unique relaxed state. However, it is usually desirable to bias the models to a particular configuration, especially when the shape of the target feature is approximately known. Incorporating prior knowledge greatly reduces the tendency for a dynamic contour to be distracted when the feature of interest moves over background clutter, and also prevents it from self-intersecting – see **Blake et al (1993)** and **Baumberg and Hogg (1994)**.

2.5 BALLOONS: PRESSURISED SNAKES

When not subjected to image forces, a closed active contour model reaches equilibrium by collapsing to point. **Cohen (1991)** and **Sullivan et al (1990)** therefore added a *pressure* term to make the model behave like an inflatable *balloon* or *bubble* that is trapped by strong edges but expands through edges that are weak relative to the pressure and smoothing forces. This section reviews the pressurised active contour model.

A balloon $\mathbf{u}(s) = (x(s), y(s))$, approximated discretely by N elements $\mathbf{u}_s$ where $s = 1, \ldots, N-1$, can be formed by looping an open-ended snake into a closed band such that $\mathbf{u}_0 \equiv \mathbf{u}_N$. The mechanical properties of the balloon are specified by an energy functional:

$$E_{\text{balloon}} = \underbrace{\frac{\alpha}{2}\oint \left|\frac{\partial \mathbf{u}}{\partial s}\right|^2 ds}_{\text{Tension}} + \underbrace{\frac{\beta}{2}\oint \left|\frac{\partial^2 \mathbf{u}}{\partial s^2}\right|^2 ds}_{\text{Stiffness}} - \underbrace{\frac{\rho}{2}\oint \frac{\partial \mathbf{u}}{\partial s}\times\mathbf{u}\; ds}_{\text{Pressure}} + \underbrace{\zeta\oint P(I(\mathbf{u}))\; ds}_{\text{Potential}} \tag{2.5.1}$$

The third integral in this equation is an isotropic pressure potential that controls the evolution of the area enclosed by the model.

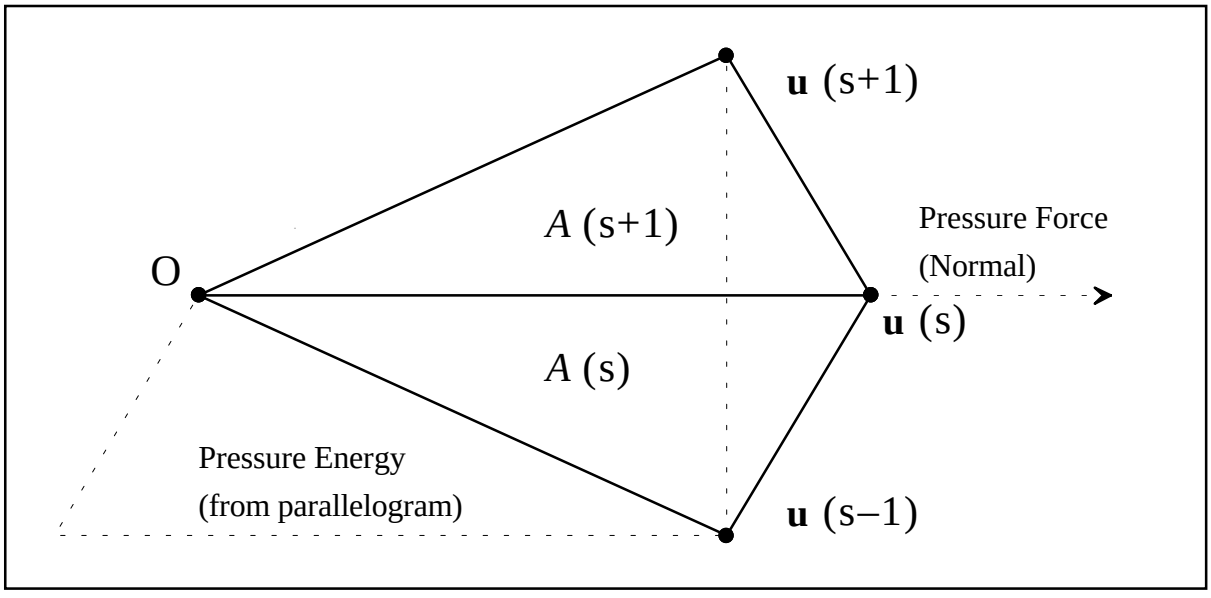

Figure 2.5.1: Discrete Approximation For Pressure Energy And Force. The area of a pressurised snake can be approximated using the patchwork of triangles formed between the individual elements and the origin. The sum of the areas of these triangles, some of which will be negative, gives the total area (pressure energy) of the model; the corresponding force used to minimise this energy is given by the normal to the model boundary.

Since the balloon is represented by N discrete elements, its area can be approximated using triangles as illustrated in Figure 2.5.1. The area A_s of the triangle formed by the origin O and the elements $\mathbf{u}_{s-1}$ and $\mathbf{u}_s$ can be calculated as follows:

$$A_s \quad = \quad \frac{1}{2}\,\mathbf{u}_s \times (\mathbf{u}_s - \mathbf{u}_{s-1}) \quad = \quad \frac{1}{2}\,\mathbf{u}_{s-1} \times \mathbf{u}_s \quad = \quad \frac{1}{2}\,(x_{s-1}y_s - x_s y_{s-1}) \tag{2.5.2}$$

Converting to continuous notation, negating and integrating this expression around the model gives the total pressure energy:

$$E_{\text{pre}} \quad = \quad -\oint A(s) \quad ds \quad = \quad -\frac{\rho}{2}\oint \left(x\,\frac{\partial y}{\partial s} - y\,\frac{\partial x}{\partial s} \right) \quad ds \tag{2.5.3}$$

The constant ρ governs the rate at which the area is allowed to increase or decrease according to the equations of motion.

The pressure term from Equation 2.5.3 can be rewritten using dashes to indicate derivatives:

$$E_{\text{pre}}(\mathbf{u}) \quad = \quad -\frac{\rho}{2}\oint \mathbf{u} \times \mathbf{u}' \quad ds \tag{2.5.4}$$

If the snake changes slightly then the pressure energy of the new configuration is:

$$E_{\text{pre}}(\mathbf{u} + \delta\mathbf{u}) \quad = \quad -\frac{\rho}{2}\oint (\mathbf{u} + \delta\mathbf{u}) \times \left(\mathbf{u}' + \delta\mathbf{u}' \right) \quad ds \tag{2.5.5}$$

This equation expands as follows:

$$E_{\text{pre}}(\mathbf{u} + \delta\mathbf{u}) \quad = \quad -\frac{\rho}{2}\oint \{\mathbf{u} \times \mathbf{u}' \;+\; \mathbf{u} \times \delta\mathbf{u}' \;+\; \delta\mathbf{u} \times \mathbf{u}' \;+\; \delta\mathbf{u} \times \delta\mathbf{u}'\} \quad ds \tag{2.5.6}$$

Ignoring the (very small) final term in this expansion, and subtracting 2.5.4 from 2.5.6 gives a linear approximation for the energy change:

$$\delta E \quad = \quad -\frac{\rho}{2}\oint \{\mathbf{u} \times \delta\mathbf{u}' \;+\; \delta\mathbf{u} \times \mathbf{u}'\} \quad ds \tag{2.5.7}$$

Using integration by parts, and the anti-symmetry of the cross product:

$$\delta E \quad = \quad -\frac{\rho}{2}\oint \{-\mathbf{u}' \times \delta\mathbf{u} \;-\; \mathbf{u}' \times \delta\mathbf{u}\} \quad ds \quad = \quad \rho \oint \mathbf{u}' \times \delta\mathbf{u} \quad ds \tag{2.5.8}$$

This can be rewritten using the symbol $\perp$ to indicate rotation through a right-angle:

$$\delta E \quad = \quad \rho \oint \left(\frac{\partial \mathbf{u}}{\partial s} \right)^{\perp} \cdot \delta\mathbf{u} \quad ds \tag{2.5.9}$$

Using the methods from Section 2.3, Equation 2.5.9 can be incorporated into an expression for the total energy change due to making a small change $\delta\mathbf{u}$ to a pressurised snake:

$$\delta E \quad = \quad \oint \left(\frac{\partial P}{\partial \mathbf{u}} \quad - \quad \alpha\,\frac{\partial^2 \mathbf{u}}{\partial s^2} \quad + \quad \beta\,\frac{\partial^4 \mathbf{u}}{\partial s^4} \quad + \quad \rho \left(\frac{\partial \mathbf{u}}{\partial s} \right)^{\perp} \right) \cdot \delta\mathbf{u} \quad ds \tag{2.5.10}$$

Assuming it is not already at a minimum, the energy of the model will decrease at each iteration if $\delta\mathbf{u}$ is a negative fraction $-\delta t$ of this energy gradient:

$$\delta\mathbf{u} \quad = \quad -\delta t \left(\frac{\partial P}{\partial \mathbf{u}} \quad - \quad \alpha\,\frac{\partial^2 \mathbf{u}}{\partial s^2} \quad + \quad \beta\,\frac{\partial^4 \mathbf{u}}{\partial s^4} \quad + \quad \rho \left(\frac{\partial \mathbf{u}}{\partial s} \right)^{\perp} \right) \tag{2.5.11}$$

At the limit of infinitesimal steps, the continuous descent equation is:

$$\frac{\partial \mathbf{u}}{\partial t} = \underbrace{\alpha \frac{\partial^2 \mathbf{u}}{\partial s^2}}_{\text{Tension Force}} - \underbrace{\beta \frac{\partial^4 \mathbf{u}}{\partial s^4}}_{\text{Stiffness Force}} - \underbrace{\rho \left(\frac{\partial \mathbf{u}}{\partial s} \right)^{\perp}}_{\text{Pressure Force}} - \underbrace{\frac{\partial P}{\partial \mathbf{u}}}_{\text{Image Force}} \tag{2.5.12}$$

In the absence of an image, the steady-state solution to this equation is a circle (assuming uniform tension, stiffness and pressure parameters) because of the symmetry in the cyclic boundary parameter *s*. If the stiffness is zero then the radius of the balloon is proportional to the pressure-to-tension ratio; adding stiffness will slightly reduce the radius. Unfortunately, in the presence of image data, the feature of interest is not always extracted because the model may reach equilibrium on another (stronger) feature as shown in Figure 2.5.2.

(a) Initial Balloon **(b) Final Balloon**

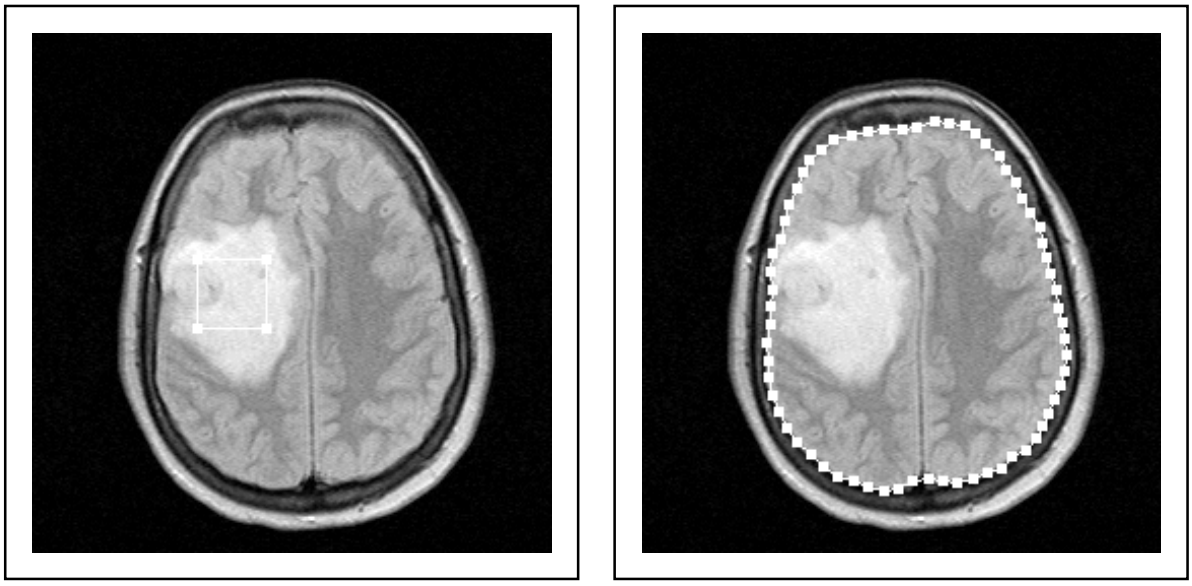

Figure 2.5.2: A Pressurised Snake (Balloon). This figure shows one of the main problems with balloons – that they expand through weak edges. (**a**) A balloon with four elements is initialised inside the pale area of inflammation (oedema) in the left hemisphere. (**b**) Unfortunately, the edges produced by the oedema are not strong enough to overcome the inflation force, so the model expands until it encounters the stronger edge produced by the cerebral cortex. (The model has also been re-parameterised, as explained in Chapter 4.)

2.6 DISCUSSION AND CONCLUSIONS

Active contour models (snakes) are essentially semi-global edge-detectors which are constrained to be smooth and continuous. Snakes thus form freely deformable models of image features, which change when the features change; this makes them very useful for analysing image sequences. However, snakes are very sensitivity to initialisation; if a model is initialised too far away from the feature of interest it may fail to locate the appropriate energy minimum.

During energy minimisation, there is a trade-off between speed and stability: snakes converge quickly on energy minima when large time steps are used, but are then prone to oscillation; improved stability comes from using small time steps, but at the cost of slow convergence. The semi-implicit energy minimisation scheme can deal with large tension and stiffness without the need for very small time steps; however, oscillation can still arise because of image forces. The Euler time-stepping scheme is mathematically less elegant, and is theoretically more susceptible to oscillation; however, it is also computationally less expensive because it avoids the need to invert a pentadiagonal smoothing matrix.

The addition of a pressure term reduces the sensitivity of active contour models to initialisation by making them expand towards image features (usually edges). However, strong features must be present if the image force is to overcome the pressure so that the model can reach equilibrium. Similarly, if the feature of interest gets smaller between consecutive images then it will have to produce a large image force to overcome the pressure. Pressurised snakes are therefore unlikely to contract in response to changing image data, and consequently they do not always perform well when analysing image sequences.

Both ordinary snakes, and their pressurised variants are usually restricted to detecting features according to the strength of the edges in an image, ignoring other features; in particular, they do not exploit texture or colour information. Snakes and balloons are difficult to use when the edges of a feature are weak, noisy and diffuse because they tend to be distracted by nearby edges that do not belong to the desired feature. These deficiencies suggest that some extension to the pressurised model is needed to link it more strongly with the image data; this extension might exploit image characteristics such as texture and colour. The next chapter therefore introduces a new alternative to active contour models and balloons – the statistical snake or active region model.