CHAPTER 2

Active Contours for Tracking

“The earth doth like a snake renew Her winter weeds outworn.”

—Percy Bysshe Shelley

“Which of you fathers, if your son asks for a fish, will give him a snake instead?”

—Jesus Christ

2.1 OVERVIEW

The flexible outline provided by an active contour (a.k.a. the snake) is mated well with the amorphous, nonrigid boundaries found in many biomedical applications. The purpose of this chapter is to put forth the active contour model as a tool for object tracking. Essentially, the active contour is a contour that seeks to optimize energy, where the energy quantifies the “goodness” of the contour—how smooth it is and how well localized it is with respect to the image edges. The movement of the active contour is then motivated by minimizing this energy, making moves that improve the goodness of the contour in some manner. We first discuss the concept of gradient descent as an energy minimization tool for active contour computation. Employment of the active contour for cell tracking is addressed next. In this context, we illustrate several external forces for active contours—the external
forces that guide the active contour to reside on the desired image edges. Here, the minimax method for determining active contour parameter values is outlined. Finally, dynamic programming for snake energy minimization is discussed as an alternative to the gradient descent method.

2.2 THE BASIC SNAKE MODEL

In the eighties, disco died and the snake was born. And as disco band “The Bee Gees” had three founders, so the snake had its three creators: Kass, Witkin, and Terzopoulos [1]. After nearly three decades of deformation, biomedical imaging researchers have a mature tool that accommodates segmentation and tracking of nonrigid objects.

The basic snake model introduced by Kass et al. is a good starting point for exploration of the world of active contours. We are first going to introduce this basic snake model and then we will utilize the model to track objects of interest in biomedical images. The crux of the snake-based tracking will be using an active contour to lock onto the boundary of a moving object whilst the object moves in a video sequence.

A snake is simply the mathematical description of a contour used to delineate a boundary. As mentioned, the goal is moving the contour such that an energy measure is minimized. To understand the working principles of the snake model, let us first consider an image of a circle as shown in Fig. 2.1(a). Figure 2.1(b) shows the image intensity profile as a surface plot. The negative (additive inverse) of the image gradient magnitude is shown in Fig. 2.1(c), and corresponding surface plot is shown in Fig. 2.1(d). The additive inverse of the gradient magnitude forms a topographic surface and serves as the potential energy for the snake. Object boundaries correspond to the valleys of this topographic surface, e.g. the annular valley shown in Fig. 2.1(d). Let us consider point A on the circle image as shown in Fig. 2.1(e), which is a magnified version of Fig. 2.1(d). The potential energy at A may be considered as the height of the potential surface (negative gradient magnitude) at A, measured from some reference. According to the principle of
FIGURE 2.1: Illustration of the gradient descent method. (a) A circle image. (b) The circle image intensity as a surface. (c) Negative of the image gradient magnitude of the circle image. (d) Negative image gradient magnitude as a surface plot. (e) Points A and B on the negative gradient magnitude surface. The point A scrolls down to B along the path shown here. (f) Contours (collection of ordered points) on the negative gradient magnitude surface settle down to the valley.
8 BIOMEDICAL IMAGE ANALYSIS: TRACKING

energy minimization, point A will always attempt to lower its potential energy; consequently, this point will amble down the potential surface and rest in a local minimum of the surface indicated by B (see Fig. 2.1(e)). A similar situation arises in the computation of the snake. We may approximate a contour (viz., a snake) by a collection of points called snaxels. When a contour is placed on the circle image shown in Fig. 2.1(f), the total potential energy of the contour is given by the sum of potential surface heights (measured from a base reference) at the constituent snaxels. In this case too the snaxels migrate down the potential surface and reach the nearest local minima. This evolution of a contour on the potential surface is shown in Fig. 2.1(f).

The curious reader would immediately raise concerns regarding the collective behavior of the snaxels, for example whether the snaxels maintain smoothness of the contour while evolving or after reaching the minima. The good news preached in this chapter is that the inventors of the snake model already answer the question. We can reframe the question as, is the movement of a snaxel influenced by the movements of other snaxels? If independent movements of the snaxels are allowed, then it is very likely that after they reach their respective local minima, the contour shape would be jagged and sensitive to minor variations in the image gradient. In fact in biomedical imaging, this would almost always be the case since the potential surface is generally noisy.

So, the movement of a snaxel is typically dependent on the movements of a few neighboring snaxels so that the contour does not become rough or irregular. How can one make this happen in the energy minimization framework? The trick is to associate another kind of energy with the snake. This second energy function should assume a small value where the contour is smooth and should explode as the snake becomes jagged. Notice that now we have two energies associated with the snake—one is the potential energy computed from image data and the other is the snake smoothness energy. And we want both these energies to be minimized ideally. But is it possible to achieve this goal of simultaneous minimization of the energies? Certainly it is not guaranteed that a snake minimizes both the image
potential energy and the smoothness energy—these energies may lead to conflicting constraints.

Thus, some sort of compromise has to be met between these two types of energies. In order to formally achieve this joint optimality, the total energy of the snake is considered to be the sum of the image potential energy and the smoothness energy. And we now want the snake to minimize this total energy. Again the curious and the cautious reader would ask whether the total energy minimization would lead to our desired contour location. In fact, we ask a more subtle question—what is our “desired” snake location? Intuitively we want the snake to delineate our desired object from the image, and at the same time we want a smooth snake boundary.

To formally express the total snake energy as devised by Kass et al. we need to parameterize the snaxel coordinates as \((X(s), Y(s))\), where \(s \in [0, 1]\). In other words, the contour is parameterized via \(s\), hence we have a parametric contour. Let \(I(x, y)\) denote the image intensity at position \((x, y)\) within the image domain. Further, let \(f(x, y) = |\nabla I(x, y)|^2\) denote the squared image gradient magnitude. We can now express the total snake energy of Kass et al. as follows:

\[
E(X, Y) = \frac{1}{2} \int_0^1 \alpha \left( \left| \frac{dX}{ds} \right|^2 + \left| \frac{dY}{ds} \right|^2 \right) + \beta \left( \left| \frac{d^2X}{ds^2} \right|^2 + \left| \frac{d^2Y}{ds^2} \right|^2 \right) ds + \int_0^1 f [X(s), Y(s)] ds. \tag{2.1}
\]

The first integral embraces the smoothness energy or the so-called “internal energy” of the snake, and the second integral expresses the “external energy” computed over the entire contour. The negative sign in front of the second integral (external energy term) implies that we want to maximize the sum of image gradient magnitude over the entire contour. Note that the internal energy of the snake has two components weighted by non-negative weighting factors \(\alpha\) and \(\beta\). The first of the two internal energy components is known as the stretching energy (likewise, the tension) and the second term is called the bending energy (a.k.a. the rigidity) for the snake. As
the names connote, the stretching energy prevents the snake from getting stretched while the bending energy prevents bending of the snake.

Thanks to the high school calculus lessons given by Dr. Brown at Oakton High School and Dr. Bajani at Howrah Vivekananda Institution, we know that in order to minimize a function of single variable (argument), we simply compute the first derivative of the function with respect to its argument, equate the derivative to zero, and solve the equation. To discover whether the function actually achieves a local minimum value, we need to examine the value of the second derivative of the function at the extremum. If this value is positive then the point is a local minimum.

So, on the basis of the lesson of Dr. Brown, one expects that something similar should be done to the snake energy (2.1). Let us first realize that Dr. Brown did not tell us everything we needed to know in life; this high school approach does not work in this case.

The snake energy (2.1) is actually a function of the snake location, which itself is a multi-valued function of $s$. The energy (2.1) is called a functional (a function of a function). So, in the Dr. Brown paradigm, we need to differentiate the snake energy functional with respect to the snake position function. But how to differentiate a function with respect to a function? We need an advanced mathematical toolbox called the calculus of variations (likewise, variational calculus) to perform this minimization [2, 3]. The calculus of variations defines a functional derivative—the derivative of a functional with respect to its argument function(s).

So the rest is similar to high school calculus: equate the functional derivative to zero and solve the equation to ferret out the minimizing snake.

In order not to burden the reader uninterested in derivation, we banish the details of computing functional derivatives $\frac{\delta E}{\delta X}$ and $\frac{\delta E}{\delta Y}$ of the energy functional (2.1) to Appendix A. (Note that $\delta$ is used here as the functional derivative operator.) First, we equate the functional derivative of (2.1) with respect to the $X$ to zero:

$$\frac{\delta E}{\delta X} = -\alpha \frac{d^2 X}{ds^2} + \beta \frac{d^4 X}{ds^4} - \frac{\partial f}{\partial x} = 0.$$  \hspace{1cm} (2.2)
Similarly the functional derivative of (2.1) with respect to $Y$ equated to zero is given as follows:

$$\frac{\delta E}{\delta Y} = -\alpha \frac{d^2 Y}{ds^2} + \beta \frac{d^4 Y}{ds^4} - \frac{\partial f}{\partial y} = 0. \quad (2.3)$$

Equations (2.2) and (2.3) are known as Euler equations. Closed form solutions for $X$ and $Y$ from (2.2) and (2.3) in general cases, where $f$ is any arbitrary potential surface, cannot be computed. We resort to numerical techniques to solve them. One such method is known as the gradient descent method. Consider points A and B in Fig. 2.1(e) again, where the path of the movement of the point from location A to location B is shown. We show the gradient vectors of this potential surface in Fig. 2.2, where we also indicate the point locations A and B. We observe that the gradient vectors are oppositely aligned with the path from A to B. We

**FIGURE 2.2:** Gradient vectors of the potential surface in Fig. 2.1(d). Points A and B are also shown here. The direction of gradient is from B to A. Thus the negative of the gradient direction pushes the point A to point B.
may intuitively conclude that a point on a potential surface moves in the opposite direction of the gradient of the potential surface. The point stops where the gradient magnitude is zero (for example the point location B). Formally, in gradient descent the velocity of a point \((P_x, P_y)\) is proportional to the negative gradient of the potential surface, \(E(x, y)\):

\[
\frac{dP_x}{d\tau} \propto -\frac{\partial}{\partial x} E(x, y),
\]

and

\[
\frac{dP_y}{d\tau} \propto -\frac{\partial}{\partial y} E(x, y).
\]

Note that left sides of Eqs. (2.4) and (2.5) represent the rate of change of the position of the point with respect to time \(\tau\), i.e., the velocity of the point. While generating the path from A to B we have actually simulated the movement of the point by Eqs. (2.4) and (2.5). Carefully notice that the velocity of the point at B is zero, where the point stops. In an analogous way, the gradient descent method dictates that the velocity of the snake will be proportional to the negative of the functional gradient of (2.1) (see Appendix A for a derivation):

\[
\frac{\partial X}{\partial \tau} = \alpha \frac{d^2 X}{ds^2} - \beta \frac{d^4 X}{ds^4} + \frac{\partial f}{\partial x},
\]

and

\[
\frac{\partial Y}{\partial \tau} = \alpha \frac{d^2 Y}{ds^2} - \beta \frac{d^4 Y}{ds^4} + \frac{\partial f}{\partial y}.
\]

As discussed, we crave the snake locations where the velocity (left sides of (2.6) and (2.7)) is zero because then (2.6) and (2.7) fulfill the condition designated by (2.2) and (2.3). A physicist might interpret (2.6) and (2.7) as a force that drives the snake. This resultant force is composed of an internal force (stretching and bending terms) and an external force (the terms involving the image edge function \(f\)). When the resultant force is zero, the snake reaches its equilibrium position and we obtain the desired snake location.
To reach the state of equilibrium, we enact the snake motion governed by Eqs. (2.6) and (2.7), starting from an initial snake location. In order to realize the snake on a discrete grid (viz., a digital image) the continuous contour is approximated by a polygon (in case of a closed contour) or a polyline (for an open contour). As with sampling a function in time, we can increase the number of samples to better approximate the contour. The sampling theorem of Michiganian Claude Shannon can be used to determine the number of samples needed to perfectly reconstruct a continuous and bandlimited (i.e., the frequency of undulations on the contour is bandlimited) contour. These sampled vertices are also commonly referred to as snaxels. In the discrete snake description essentially the continuous parameter, \( s \in [0, 1] \) is indexed by \( i \in \{0, 1, \ldots, n-1\} \), where \( n \) is the total number of snaxels. A snaxel, located at \((X(s), Y(s))\) on the continuous contour, is denoted by \((X_i, Y_i)\) within the discrete facsimile.

The discrete versions of the Eqs. (2.6) and (2.7) for the \( i \)th snaxel are given as follows:

\[
\frac{X_{i}^{\tau+1} - X_{i}^{\tau}}{\zeta} = \alpha (X_{i+1}^{\tau} - 2X_{i}^{\tau} + X_{i-1}^{\tau}) - \beta (X_{i+2}^{\tau} - 4X_{i+1}^{\tau} + 6X_{i}^{\tau} - 4X_{i-1}^{\tau} + X_{i-2}^{\tau}) + f_x(X_{i}^{\tau}, Y_{i}^{\tau}),
\]

(2.8)

and

\[
\frac{Y_{i}^{\tau+1} - Y_{i}^{\tau}}{\zeta} = \alpha (Y_{i+1}^{\tau} - 2Y_{i}^{\tau} + Y_{i-1}^{\tau}) - \beta (Y_{i+2}^{\tau} - 4Y_{i+1}^{\tau} + 6Y_{i}^{\tau} - 4Y_{i-1}^{\tau} + Y_{i-2}^{\tau}) + f_y(X_{i}^{\tau}, Y_{i}^{\tau}),
\]

(2.9)

where the superscripts \( \tau \) and \( \tau + 1 \) respectively represent two successive discrete time instants, and the + and − operators appearing in subscripts of \( Xs \) and \( Ys \) in (2.8) and (2.9) denote modulo \( n \) addition and subtraction in order to take care of the wraparound effect for a closed contour. For example, the operation \((n - 1) + 2\) would yield 1, as opposed to \(n + 1\). The parameter \( \zeta \) is the time step that controls the magnitude of steps taken in the discrete updates.
The edge strength terms $f_x$ and $f_y$ are as follows:

$$f_x(x, y) = \frac{\partial}{\partial x} f(x, y)$$

(2.10)

and

$$f_y(x, y) = \frac{\partial}{\partial y} f(x, y).$$

(2.11)

($f_x(x, y)$, $f_y(x, y)$), or in shorthand notation ($f_x$, $f_y$), forms a vector field over the image domain that acts as an external force for the snake. To express (2.8) and (2.9) in matrix-vector notation collectively for all the snakelets, we employ linear algebra:

$$x^\tau \equiv \begin{bmatrix} X^\tau_0 \ldots, X^\tau_{n-1} \end{bmatrix}^T,$$

$$y^\tau \equiv \begin{bmatrix} Y^\tau_0 \ldots, Y^\tau_{n-1} \end{bmatrix}^T,$$

$$f^\tau_x \equiv \begin{bmatrix} f_x(X^\tau_0, Y^\tau_0), \ldots, f_x(X^\tau_{n-1}, Y^\tau_{n-1}) \end{bmatrix}^T,$$

and

$$f^\tau_y \equiv \begin{bmatrix} f_y(X^\tau_0, Y^\tau_0), \ldots, f_y(X^\tau_{n-1}, Y^\tau_{n-1}) \end{bmatrix}^T.$$

Now (2.8) is rewritten for the entire active contour as

$$\frac{x^{\tau+1} - x^\tau}{\zeta} = -A x^\tau + f^\tau_x,$$

(2.12)

and similarly (2.9) is rewritten as

$$\frac{y^{\tau+1} - y^\tau}{\zeta} = -A y^\tau + f^\tau_y.$$

(2.13)

Here, $A$ is an $n$-by-$n$ sparse matrix written as

$$A = \begin{bmatrix} c & b & a & a & b \\ b & c & b & a & a \\ a & b & c & b & a \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a & b & c & b & a \\ a & a & b & c & b \\ b & a & a & b & c \end{bmatrix}.$$

(2.14)
where in turn $a$, $b$, and $c$ are as follows:

\[ a = \beta, \quad b = -(4\beta + \alpha), \quad c = 6\beta + 2\alpha. \quad (2.15) \]

To iteratively solve for the snake location (the set of snaxel positions) at time $\tau + 1$, given the position at time $\tau$, we rewrite (2.12) and (2.13) respectively as

\[ x^{\tau+1} = x^\tau - \zeta (A x^\tau - f_x^\tau), \quad (2.16) \]

and

\[ y^{\tau+1} = y^\tau - \zeta (A y^\tau - f_y^\tau). \quad (2.17) \]

Equations (2.16) and (2.17) are known as *explicit* solution techniques [4]. The numerical stability of (2.16) and (2.17) depends on the time step $\zeta$.

An *implicit* procedure [4] that is stable for a wide range of $\zeta$ values is followed by rewriting (2.16) and (2.17) as

\[ \frac{x^{\tau+1} - x^\tau}{\zeta} = -A x^{\tau+1} + f_x^\tau, \quad (2.18) \]

and

\[ \frac{y^{\tau+1} - y^\tau}{\zeta} = -A y^{\tau+1} + f_y^\tau. \quad (2.19) \]

Immediately we obtain the following iterative form from (2.18) and (2.19):

\[ x^{\tau+1} = (I_n + \zeta A)^{-1}(x^\tau + \zeta f_x^\tau), \quad (2.20) \]

and

\[ y^{\tau+1} = (I_n + \zeta A)^{-1}(y^\tau + \zeta f_y^\tau). \quad (2.21) \]

where $I_n$ is the $n$-by-$n$ identity matrix. It should be noted that the matrix, $(I_n + \zeta A)$ is positive definite (see Appendix B), and so it is invertible. Equations (2.20) and (2.21) are sometimes referred to as the snake evolution equations. Staring from an initial snake location, these evolution equations can be used iteratively to move
the snake while tracking a biomedical object, whether it is a cell or the myocardial border or a tumor boundary.

Another way to acquire the snake evolution equations is to represent the snake energy functional approximately in a discrete framework as follows:

$$E(X_0, \ldots, X_{n-1}, Y_0, \ldots, Y_{n-1}) = \frac{1}{2} \sum_{i=0}^{n-1} \alpha (X_{i+1} - X_i)^2 + \alpha (Y_{i+1} - Y_i)^2$$

$$+ \frac{1}{2} \sum_{i=0}^{n-1} \beta (X_{i+1} - 2X_i + X_{i-1})^2 + \beta (Y_{i+1} - 2Y_i + Y_{i-1})^2 - \sum_{i=0}^{n-1} f(X_i, Y_i).$$

(2.22)

The addition and subtraction operations in the subscript are modulo $n$ addition operations as mentioned in the context of describing Eqs. (2.8) and (2.9). Note that the snake energy functional (2.22) is a function of $2n$ variables $X_0, \ldots, X_{n-1}, Y_0, \ldots, Y_{n-1}$. So if we want to compute the minima of (2.22), we need to take the partial derivatives of (2.22) with respect to these $2n$ variables:

$$\frac{\partial E}{\partial X_i} = -\alpha (X_{i+1} + X_{i-1} - 2X_i) + \beta (X_{i+2} - 4X_{i+1} + 6X_i - 4X_{i-1} + X_{i-2})$$

$$- f'_x(X_i, Y_i)$$

(2.23)

and

$$\frac{\partial E}{\partial Y_i} = -\alpha (Y_{i+1} + Y_{i-1} - 2Y_i) + \beta (Y_{i+2} - 4Y_{i+1} + 6Y_i - 4Y_{i-1} + Y_{i-2})$$

$$- f'_y(X_i, Y_i), \forall i \in \{0, 1, \ldots, n-1\}.$$ 

(2.24)

From Eqs. (2.23) and (2.24) we can reach the gradient descent Eqs. (2.8) and (2.9), and derive the snake evolution Eqs. (2.20) and (2.21) in the same way as before. Note that in this latter derivation we altogether bypass the use of calculus of variations, because here we are essentially dealing with functions, not functionals.

Consider again the toy image of Fig. 2.1(a). We have observed in Fig. 2.1 that the snake scrolls down the edge potential surface. (The potential $f$ of Eq. (2.1) in this case is the edge potential surface, i.e., $f(x, y) = |\nabla I(x, y)|^2$.) Let us now look at the snake evolution on the image plane as shown in Fig. 2.3(a). The initial
**FIGURE 2.3:** (a) Snake evolution on the circle of Fig. 2.1(a) from the edge potential force. The darker contour represents the initial contour location. The lighter contours represent intermediate contours during the contour evolution. The final contour delineating the circle is shown in Fig. 2.3(b).

Another example of snake evolution via edge potential force is given in Fig. 2.4. In Fig. 2.4(a) a leukocyte (white blood cell), as seen from an intravital video microscopy.

**FIGURE 2.4:** (a) A leukocyte as seen from intravital video microscopy. (b) Capturing the leukocyte with a snake and edge potential force. Initial final and intermediate contours are shown in the figure.
microscopy, is shown. Figure 2.4(b) shows snake evolution. These two simulations are performed via Eqs. (2.20) and (2.21).

The final contour positions in these two cases are determined by a prescribed number of iterations for Eqs. (2.20) and (2.21). The iterative process can also be stopped if the average or maximum distance between snaxels in consecutive iterations does not exceed a prescribed limit. Another interesting stopping criterion is given in [5], where the ratio of the number of oscillating snaxels over the total number of snaxels is counted in each iteration of Eqs. (2.20) and (2.21), and once the ratio exceeds a prescribed limit the process is stopped. A snaxel is deemed to be oscillating if in two consecutive iterations, the snaxel encounters opposing directions of the external force. Another very straightforward way for enforcing a stopping criterion is to account for the change in snake energy (2.1). Once the energy reaches a minimum (locally or otherwise) the process can be stopped.

2.3 SNAKE EXTERNAL FORCES

Recall that the overall goal of this chapter is the design of a snake that can be used to capture a moving object in biomedical imaging. It is the external energy that acts as a chemoattractant, drawing the snake to the proper boundary. So far we have seen that the squared image gradient magnitude as the basis for snake external energy. First, we illustrate a basic limitation of this potential energy source for the snake. Let us imagine that the initial active contour is far away (5–10 pixels away, for example) from the coveted edges and residing in a perfectly homogeneous region in an image (as shown in Fig. 2.5). Implementing the snake with the external force field \((f_x, f_y)\) is unsuccessful in this case, because inside the homogeneous region, the image gradient magnitude would be zero and consequently there will be little/no edge force \((f_x, f_y)\) acting on the snake. Unable to “sense” the force due to the edges and guided only by the internal force, the active contour may not move toward the desired edge. In essence, the gradient-based edge potential force field \((f_x, f_y)\) has a limited capture range.
FIGURE 2.5: Illustrations of the failure of the snake method using edge potential force in object boundary delineation. Snake evolutions are shown on (a) synthetic image and on a (b) real image.

The distance potential force is a remedy to this limitation for binary images (a binary image has only two intensity levels, 0 and 1). In this case, the distance surface acts as the source of the snake external force. The distance surface (or distance map) $D(x, y)$ is computed from a binary image $I(x, y)$ in the following way:

$$D(x, y) = \min_{(p, q) \in ((a, b): I(a, b) = 1)} [d(x, y; p, q)], \quad (2.25)$$

where $d(x, y; p, q)$ is a distance metric between two locations $(x, y)$ and $(p, q)$. For example, $d$ may be Euclidean distance metric:

$$d(x, y; p, q) = \sqrt{(x - p)^2 + (y - q)^2}. \quad (2.26)$$

The minimum in (2.25) is computed over all locations $(p, q)$ where the binary image intensity is unity. Figure 2.6(a) shows an example distance potential surface. Once we construct $D(x, y)$, we compute the distance potential force field $(-D_x(x, y), -D_y(x, y))$ [Note the appearance of the negative sign: since the
negative derivative of the potential is the force], and utilize it in the gradient descent equations:

\[
\frac{\partial X}{\partial \tau} = \alpha \frac{d^2 X}{ds^2} - \beta \frac{d^4 X}{ds^4} - \frac{\partial D}{\partial x},
\]

(2.27) and

\[
\frac{\partial Y}{\partial \tau} = \alpha \frac{d^2 Y}{ds^2} - \beta \frac{d^4 Y}{ds^4} - \frac{\partial D}{\partial y}.
\]

(2.28)

Note that Eqs. (2.27) and (2.28) are identical to Eqs. (2.6) and (2.7) except that the edge force \((f_x, f_y)\) is replaced by the distance potential force \((-D_x(x, y), -D_y(x, y))\). The distance potential force is shown in Fig. 2.7(a) and the corresponding snake evolution is illustrated in Fig. 2.7(b).

2.3.1 The Balloon Force

A balloon force is another type of external force for an active contour [6]. True to the name, the balloon force tries either to inflate or to deflate a closed contour. The balloon force exerts a force that is normal to the active contour (outward or inward). If it is an inflating force then the direction is outward normal; otherwise, it is directed
FIGURE 2.7: (a) Distance potential force. (b) Snake evolution via distance potential force to the inward normal. The normal direction to a parameterized contour at a point \((X(s), Y(s))\) is given by the direction of \((-Y_s, X_s)\), \((i.e., -Y_s, X_s)\). The inward and the outward directions of the normal are determined by the parameterization \(s\): whether \(s\) increases or decreases while traveling in a particular direction (for example clockwise) along the curve. So, the balloon force may be quantified by \(\epsilon_t(\alpha - Y_s, X_s)\). In other words, the balloon force is proportional to the normal to the curve with a proportionality constant \(\epsilon_t\), which in general may be evolution time dependent. The snake evolution in this case is governed by

\[
\frac{\partial X}{\partial \tau} = \alpha \frac{\partial^2 X}{\partial s^2} - \beta \frac{\partial^4 X}{\partial s^2} - \epsilon_t Y_s,
\]

and

\[
\frac{\partial Y}{\partial \tau} = \alpha \frac{\partial^2 Y}{\partial s^2} - \beta \frac{\partial^4 Y}{\partial s^2} + \epsilon_t X_s.
\]

The balloon force has gained popularity in medical imaging applications in which an initial contour may be easily placed inside the region to be segmented, as with the lung segmentation application in [7].

In general, we might have more than one type of external force acting on a snake. For example, in Fig. 2.8, we show snake evolution based on two external
FIGURE 2.8: Snake evolution with balloon force and edge potential force. The contour is seen to “leak” through the edge gaps because of balloon force forces, an edge force and a balloon force. In such a case the snake evolution equations are given by

\[
\frac{\partial X}{\partial \tau} = \alpha \frac{\partial^2 X}{\partial s^2} - \beta \frac{\partial^4 X}{\partial s^4} + \lambda f_x(X, Y) - \epsilon \tau Y, \tag{2.31}
\]

and

\[
\frac{\partial Y}{\partial \tau} = \alpha \frac{\partial^2 Y}{\partial s^2} - \beta \frac{\partial^4 Y}{\partial s^4} + \lambda f_y(X, Y) + \epsilon \tau X. \tag{2.32}
\]

Here \(\lambda\) is a non-negative weight for the edge force.

2.3.2 Gradient Vector Flow

Gradient vector flow (GVF) represents a noteworthy advance in active contour design for biomedical image analysis. In GVF, Xu and Prince [8] construct an external force field \((u(x, y), v(x, y))\) by diffusing the edge force \((f_x, f_y)\), away from edges to the homogeneous regions, at the same time keeping the constructed field...
as close as possible to the edge force near the edges. They achieve this goal through
the minimization of the following energy functional:
\[
E_{\text{corr}}(u, v) = \frac{1}{2} \int \int \mu (u_x^2 + u_y^2 + v_x^2 + v_y^2) + (f_x^2 + f_y^2)(u - f_x)^2 \\
+ (v - f_y)^2) \, dx \, dy,
\]
where \( \mu \) is a non-negative parameter expressing the degree of smoothness of the
field \((u, v)\). The interpretation of (2.33) is straightforward—the first integrand
keeps the field, \((u, v)\), smooth. This term is quite similar to the solution for the
classical Laplace’s equation. The second integrand forces the vector field to resemble
the initial edge force near the edges \((i.e.,\text{ where the edge force strength is high})\).
Variational minimization of (2.33) results in the following two Euler equations (see
Appendix C for derivation):
\[
\mu \nabla^2 u - (f_x^2 + f_y^2)(u - f_x) = 0,
\]
and
\[
\mu \nabla^2 v - (f_x^2 + f_y^2)(v - f_y) = 0.
\]
Solving (2.34) and (2.35) for \((u, v)\) results in gradient vector flow (GVF) that acts
as an external force field for the active contour. The derivation details of GVF field
are given in Appendix C. Figure 2.9(a) shows the GVF force computed on the
circle image of Fig. 2.1(a). Since the GVF vectors exist in homogeneous regions
\((i.e.,\text{ where edges are absent})\) as well, the capture range of the edge force has been
effectively increased and the snake correctly captures the circle. In addition to being
capable of attracting the active contour from a distance toward the edge, GVF can
drag the active contour inside a long concavity (formed by the edges) [8]. Once
the GVF force field \((u, v)\) is computed via (2.34) and (2.35), it is utilized in the
following snake evolution equations:
\[
\frac{\partial X}{\partial \tau} = \alpha \frac{\partial^2 X}{\partial s^2} - \beta \frac{\partial^4 X}{\partial s^4} + u(X, Y),
\]
FIGURE 2.9: (a) GVF force field on the circle image of Fig. 2.1(a). (b) Snake evolution via GVF. Same initial contour as Fig. 2.5(a) is used here. (c) GVF snake evolution on leukocyte image. Same initial contour as Fig. 2.5(b) is used here.
\[
\frac{\partial Y}{\partial \tau} = \alpha \frac{\partial^2 Y}{\partial s^2} - \beta \frac{\partial^4 Y}{\partial s^4} + v(X, Y).
\] (2.37)

Figure 2.9(b) shows snake evolution via GVF. Notice that the initial contour for this snake is identical to that of Fig. 2.5(a). In contrast to the previous case (using the edge potential force), the snake is able to delineate the circle. Figure 2.9(c) shows GVF snake evolution on the leukocyte image of Fig. 2.4(b). These figures exemplify the increase in capture range provided by GVF. Later in this chapter, we will see that GVF is extremely important in tracking applications, where accurate initialization of the contour is impossible.

### 2.4 CASE STUDY: TRACKING WITH SNAKES

Snakes are perhaps most widely used in biomedical image segmentation. The treatment of snakes in this book, however, is directed toward the important task of biomedical tracking. We believe the most straightforward way to communicate the snake tracking approach is to use an application-based case study.

*Tracking* is defined here as the task of following an object through a temporal sequence of images. Tracking in general is composed of two subtasks—object detection and correspondence resolution. Often these two tasks are performed simultaneously, because in general, they can be viewed as interdependent. Correspondence resolution is the problem of identifying a specific object in the current frame that appeared in the previous frame. Of course, when we are tracking only one target and no other targets are present in the video, we do not need any correspondence resolution strategy. Sometimes correspondence resolution is performed through a nearest neighbor assumption. In the nearest neighbor paradigm (used in this case study), the closest detection in space with respect to a previous target position (or predicted target position) is matched to a given target.

Let us now turn our attention to the application of tracking rolling leukocytes observed *in vivo* for this case study. Although the tracking method described in this section is tailored to the rolling leukocyte tracking, this approach may also
FIGURE 2.10: Six frames from an intravital video sequence showing rolling leukocytes

be applied to tracking cells with well-defined shapes. In Fig. 2.10, a few frames obtained via intravital video microscopy show the motion of rolling leukocytes. Rolling leukocytes are activated leukocytes that move at a much slower speed than the blood flow [9, 10]. With the nearest neighbor assumption, the leukocyte-tracking algorithm with active contours can be described as follows.

Algorithm 2.1

1. Leukocyte detection. To initiate this algorithm, a leukocyte on the first frame is detected either manually or automatically, and then active contour evolution is performed to delineate the detected leukocyte.

2. Tracking. From the second frame onwards for each frame execute following steps:
   a. Initial active contour placement. The final contour delineating the leukocyte from the previous frame is placed over the current video frame.
   b. Active contour evolution. Starting from the initial active contour, evolution is performed on the current video frame to delineate the displaced rolling leukocyte.
To delineate a leukocyte with an active contour, we have convinced ourselves that instead of (or in addition to) the smoothness internal energy as described in the work of Kass et al., a shape and size constrained contour is needed. The reason is obvious—a leukocyte is somewhat circular in shape and has a predictable size. Typically a leukocyte has a radius of about 4–7 microns. Thus we require the contour not to deviate much from a circle of a specified radius. This constraint serves as the internal energy for the active contour. We place another constraint on the contour evolution. This constraint comes from the observation about the movement of leukocytes basically follows the blood flow direction. This means the movement of a rolling leukocyte in the direction orthogonal to the blood flow is insignificant. If we align the $x$-axis in the direction of blood flow, which can be approximated by the venule centerline, then the inter-frame rolling leukocyte movement along the $y$-axis is limited.

The constrained active contour for leukocyte delineation is expressed as a parametric curve via a reference point (typically the center point), $(P, Q)$, and the polar coordinates $(R(t), t)$; the Cartesian coordinates of the contour points are $(P + R(t)\cos(t), Q + R(t)\sin(t))$. Figure 2.11 depicts such a “radial” active contour. This active contour must be collocated with positions of high gradient magnitude in the image, at the same time the contour should not be deviated significantly from a circular shape of a desired radius. The following energy functional of a shape–size constrained snake serves to delineate

\[ E = \int \left( \frac{1}{2} \alpha \left( \frac{\partial^2 P}{\partial t^2} \right)^2 + \beta \left( \frac{\partial P}{\partial t} \right)^2 \right) dt + \gamma \int \left( 1 - \frac{1}{R(t)^2} \right) \left( \frac{\partial P}{\partial t} \right)^2 dt \]

\[ + \int \left( \frac{1}{2} \alpha \left( \frac{\partial^2 Q}{\partial t^2} \right)^2 + \beta \left( \frac{\partial Q}{\partial t} \right)^2 \right) dt + \gamma \int \left( 1 - \frac{1}{R(t)^2} \right) \left( \frac{\partial Q}{\partial t} \right)^2 dt \]

\[ + \int \left( \frac{1}{2} \alpha \left( \frac{\partial^2 R}{\partial t^2} \right)^2 + \beta \left( \frac{\partial R}{\partial t} \right)^2 \right) dt + \gamma \int \left( 1 - \frac{1}{R(t)^4} \right) \left( \frac{\partial R}{\partial t} \right)^2 dt \]

\[ + \int \left( \frac{1}{2} \alpha \left( \frac{\partial^2 \theta}{\partial t^2} \right)^2 + \beta \left( \frac{\partial \theta}{\partial t} \right)^2 \right) dt \]

\[ + \gamma \int \left( 1 - \frac{1}{R(t)^4} \right) \left( \frac{\partial \theta}{\partial t} \right)^2 dt \]

\[ + \int \left( \frac{1}{2} \alpha \left( \frac{\partial^2 \ell}{\partial t^2} \right)^2 + \beta \left( \frac{\partial \ell}{\partial t} \right)^2 \right) dt + \gamma \int \left( 1 - \frac{1}{R(t)^4} \right) \left( \frac{\partial \ell}{\partial t} \right)^2 dt \]

\[ + \int \left( \frac{1}{2} \alpha \left( \frac{\partial^2 \phi}{\partial t^2} \right)^2 + \beta \left( \frac{\partial \phi}{\partial t} \right)^2 \right) dt + \gamma \int \left( 1 - \frac{1}{R(t)^4} \right) \left( \frac{\partial \phi}{\partial t} \right)^2 dt \]

\[ + \int \left( \frac{1}{2} \alpha \left( \frac{\partial^2 \gamma}{\partial t^2} \right)^2 + \beta \left( \frac{\partial \gamma}{\partial t} \right)^2 \right) dt \]
leukocytes:

\[ E_{\text{f--snake}}(P, Q, R) = E_{\text{edge}}(P, Q, R) + \mu_{\text{cons}} E_{\text{cons}}(R) + \mu_{\text{pos}} E_{\text{pos}}(P, Q, R), \]  

(2.38)

where \( E_{\text{edge}} \) is the external/edge force, \( E_{\text{cons}} \) is the shape–size constraint, and \( E_{\text{pos}} \) is the position constraint.

The edge constraint for this application is written as

\[ E_{\text{edge}}(P, Q, R) = -\frac{1}{L_s} \int_0^{2\pi} w[P + R(t) \cos(t), Q + R(t) \sin(t)] R(t) \, dt, \]  

(2.39)

where \( w \) is a surface that achieves its maxima at edges (such as with the image gradient magnitude), and \( L_s \) is the length of the active contour. When the contour is residing on the ridges of this surface \( w \), this energy term is minimized.

The shape–size constraint is expressed as

\[ E_{\text{cons}}(R) = \frac{1}{2} \int_0^{2\pi} [R(t) - \rho]^2 \, dt, \]  

(2.40)

and penalizes deviations of radial distance \( R(t) \) from the desired radius, \( \rho \).

The position constraint, for flow along the \( x \)-axis, is

\[ E_{\text{pos}}(P, Q, R) = \frac{1}{2} (Q - P_Y)^2, \]  

(2.41)

The position constraint \( E_{\text{pos}} \) prevents large deviation of the active contour from the estimated direction of leukocyte rolling \( (P_Y \) indicates \( y \)-coordinate of estimated leukocyte location). The energy functional (2.38) contains three components with non-negative weights \( \mu_{\text{cons}} \) and \( \mu_{\text{pos}} \) expressing the importance of the respective energy components in the functional. In the subsequent section we discuss how proper values of these parameters can be chosen analytically.
Gradient descent equations for minimizing (2.38) are obtained by variational calculus as follows (see Appendix D):

\[
\frac{\partial P}{\partial \tau} = \overline{w_x}, \tag{2.42}
\]

where \( \tau \) here is pseudotime,

\[
\frac{\partial Q}{\partial \tau} = \overline{w_y} - \mu_{\text{pos}}(Q - P_y), \tag{2.43}
\]

and

\[
\frac{\partial R(t)}{\partial \tau} = \frac{1}{L_s} \left[ w + R(t) \frac{\partial w}{\partial x} \cos(t) + R(t) \frac{\partial w}{\partial y} \sin(t) - \overline{w} \right] - \mu_{\text{cons}} [R(t) - \rho], \tag{2.44}
\]

where

\[
\overline{w} = \frac{1}{L_s} \int_0^{2\pi} w \left[ P + R(t) \cos(t), Q + R(t) \sin(t) \right] R(t) dt, \tag{2.45}
\]

\[
\overline{w_x} = \frac{1}{L_s} \int_0^{2\pi} \frac{\partial w}{\partial x} \left[ P + R(t) \cos(t), Q + R(t) \sin(t) \right] R(t) dt, \tag{2.46}
\]

and

\[
\overline{w_y} = \frac{1}{L_s} \int_0^{2\pi} \frac{\partial w}{\partial y} \left[ P + R(t) \cos(t), Q + R(t) \sin(t) \right] R(t) dt. \tag{2.47}
\]

In steps (2a) and (2b) of Algorithm 2.1, we iteratively employ Eqs. (2.42)–(2.44) starting from an initial contour to obtain a new active contour configuration that locally minimizes (2.38) and delineates a leukocyte in the process. In the next section we discuss a suitable construction process for the edge potential surface \( w \) that facilitates leukocyte delineation via shape–size constrained active contours.

### 2.4.1 External Force for Cell Tracking Case Study

An edge potential force derived from the image gradient is limited to the local proximity of the edges. Tracking with a method such as Algorithm 2.1 would be
successful with such an external force if the frame-to-frame displacement of the
target were not large. However, if the displacements exceed several pixels, the target
will be lost. Consider a rolling leukocyte observed at the standard video capture rate
of 30 frames per second and a resolution of three pixels per micron; if the leukocyte
velocity exceeds 60 microns/s, the displacement will be on the order of 6 pixels per
frame. In such a case, it is unlikely that the snake will “see” the boundary if the
guiding force is based solely on the intensity gradient. One remedy is to increase
the frame rate. However, then computational expense increases because there are
more frames to process. Another remedy is to have a slightly modified version
of Algorithm 2.1 where we advance the contour from the previous frame in the
direction of blood flow so that the advanced contour becomes close to the leukocyte
boundary on the current frame. At this point, the knee-jerk question would be—
how far should the contour be advanced? One approach could involve learning the
leukocyte movement pattern; then predicting this advancement using the Kalman
filter or some other predictor and estimator. However rolling leukocytes exhibit
various mechanisms of movement while rolling along the microvessel wall—they
may halt briefly, then make a sudden jump, and then continue steadily at a constant
velocity. It is also not uncommon that a leukocyte exhibits combinations of some
or even all of these movement patterns. Therefore, prediction of movement with a
constant velocity model is likely to be unsuccessful.

2.4.2 Motion Gradient Vector Flow

What if we used a gradient vector flow field that was biased in the known direction
of motion? We can design an external force field for the shape, size, and position
constrained snake taking into account cell movement direction so that a lagging
initial contour will be drawn toward the cell edge on the current frame. The external
force should also be able to handle the case where the frame-to-frame leukocyte
displacement is small, or nearly zero. As we have already stated while describing the
position constraint, the cell motion direction more or less follows the blood flow.
Therefore this direction of cell movement can be estimated a priori when extracting
the microvessel boundary. (See Section 2.5 for a discussion of microvessel boundary detection.)

Motion gradient vector flow (MGVF) is an external force that can be utilized in Algorithm 2.1 to track a moving object with a frame-to-frame displacement that is less than its diameter. We first want to represent the gradient magnitude surface \( f = |\nabla I| \) by a surface \( w \); then gradient of this surface (i.e., \( \nabla w \)) will serve as the external force field for the snake. We may argue that this surface should have two properties: (a) the slope of this surface inside a cell should be such that a contour from the previous frame, even with minimal overlap with the cell, will be dragged by the surface slope toward leukocyte delineation; (b) once the contour reaches the leukocyte edge, it should cease movement (i.e., it achieves a state of equilibrium). Minimizing the following energy functional we create such a surface \( w \) [9].

\[
E_{MGVF}(w) = \frac{1}{2} \iint \left\{ \mu H_\varepsilon(\nabla w \cdot (\mathbf{v}_x, \mathbf{v}_y)) |\nabla w|^2 + f(w - f)^2 \right\} dxdy, \tag{2.48}
\]

where \((\mathbf{v}_x, \mathbf{v}_y)\) is the known blood flow direction, and \( H_\varepsilon \) is a regularized (by the positive parameter \( \varepsilon \)) Heaviside function that is a continuously differentiable approximation to the unit step function:

\[
H_\varepsilon(z) = \frac{1}{2} \left( 1 + \frac{2}{\pi} \tan^{-1} \left( \frac{z}{\varepsilon} \right) \right). \tag{2.49}
\]

Because of the presence of the Heaviside function in (2.48), the diffusion of \( w \) is maximized when the Heaviside function achieves a value of unity, and it is minimized when the Heaviside function is zero. In other words, when the vectors \( \nabla w \) (\( \nabla w \) serves as the external force in MGVF) and \((\mathbf{v}_x, \mathbf{v}_y)\) are aligned with each other, the diffusion is maximal. Also because of the \( f(w - f)^2 \) term in the integral of (2.48), the surface \( w \) remains close to \( f \) whenever the value of \( f \) is high. The parameter \( \mu \) is a non-negative constant controlling the contribution of the first (diffusivity) term.

Applying the variational principles along with a bag of minor mathematical tricks (see Appendix E) for the minimization of (2.48), we obtain a gradient descent
equation that can be used to derive the motion gradient vector flow field:

\[
\frac{\partial w}{\partial \tau} = \mu \text{div}\{H_c[\nabla w \cdot (v^x, v^y)]\nabla w\} - f(w - f). \tag{2.50}
\]

The diffusion mechanism is clearly understood from the “div” (divergence) term in Eq. (2.50). It is an anisotropic diffusion with a diffusion coefficient \(H_c(\nabla w \cdot (v^x, v^y))\) that encourages diffusion where \(\nabla w\) and \((v^x, v^y)\) form acute angles and discourages diffusion where they form obtuse angles. Once (2.50) is applied, the solution surface \(w\) serves as the negative of the potential for the snake (viz., \(\nabla w\)), acting as the external force for the snake. We refer to this force field \(\nabla w\) as the motion gradient vector flow force.

### 2.4.3 Computation of Motion Gradient Vector Flow Field

To obtain a solution to (2.50), we follow an eight-neighborhood system on the discrete Cartesian image domain and utilize a Jacobian solution procedure as was used in solving the traditional anisotropic diffusion equation [11].

\[
\begin{align*}
    w^{\tau+1}_{i,j} &= w^\tau_{i,j} + \frac{\mu}{\lambda} \sum_{l=-1}^{1} \sum_{m=-1}^{1} H_c[|\nabla w^\nu + m v^\nu|(w^\nu_{i+l,j+m} - w^\nu_{i,j})](w^\nu_{i+l,j+m} - w^\nu_{i,j}) \\
    &\quad - \frac{1}{\lambda} f_{i,j} (w^\nu_{i,j} - f_{i,j}).
\end{align*}
\tag{2.51}
\]

Here \(w^0_{i,j} = f_{i,j}, w^\tau_{i,j}, \text{ and } f_{i,j}\) respectively denote the value of the surface \(w\) and the edge-map \(f\) at the \((i, j)\)th location in the discrete domain, \(\tau\) denotes the iteration number, and \(\lambda\) denotes inverse of the time-step. The following proposition illustrates the convergence conditions and the speed of convergence for (2.51).

**Proposition 1.** The numerical implementation given by (2.51) is convergent, and the rate of convergence is that of a geometric series of common ratio \(\sigma\), provided

(i). the edge-map \(f\) is normalized such that

\[
0 < \sigma \leq f_{i,j} \leq 1, \quad \forall i, j, \tag{2.52}
\]

and
FIGURE 2.12: MGVF force field on the synthetic circle image. The direction of motion \((v^x, v^y)\) is from right to left here

(ii). we select the multiplicative inverse of the time-step as

\[
\lambda \geq 1 + 8\mu. \tag{2.53}
\]

Proof. See Appendix F.

Let us illustrate the efficacy of MGVF through a couple of examples. Figure 2.12 shows MGVF field \(\nabla w\) for the circle image of Fig. 2.1(a). We have assumed \(v^x = -1, v^y = 0\) (i.e., the circle is moving in the negative \(x\)-direction). The vector field MGVF is quite different from the GVF vector field for the circle shown in Fig. 2.9(a). A lagging active contour can now be attracted to the circle edges.

The next set of figures illustrates this point for a leukocyte-tracking example. Figure 2.13(a) shows the leukocyte and Fig. 2.13(b) shows the corresponding MGVF force field for \(v^x = -1, v^y = 0\). Figures 2.13(c)–2.13(d) shows leukocyte delineation by implementing a snake via gradient descent Eqs. (2.42)–(2.44) in conjunction with the computed \(w\) via (2.51).
2.5 CHOOSING PARAMETER VALUES

One of the aspects of snake-based tracking that is apt to a knob-tweaking solution is the question of weighting parameter selection. The minimax criterion provides an analytical way to determine the parameters such as $\mu_{\text{cons}}$ and $\mu_{\text{pos}}$ of (2.38) involved in multi-component energy functionals [12]. To apply the minimax criterion, the individual energy components must be non-negative and the energy functional must be a convex combination of the constituent energy components. The combination
may be linear, quadratic or cubic, and so on. For example, the weighting parameters in the quadratic combination are non-negative and when squared they add up to unity. To conform to these requirements of the minimax criterion, we adopt the energy functional

\[
E_{\text{nonlin-rsk}}(P, Q, R, \mu_{\text{cons}}, \mu_{\text{pos}}) = \sqrt{(1 - \mu_{\text{cons}}^2 - \mu_{\text{pos}}^2)}[1 + E_{\text{edge}}(P, Q, R)]
\]

\[+ \mu_{\text{cons}} E_{\text{cons}}(R) + \mu_{\text{pos}} E_{\text{pos}}(P, Q, R),\]

(2.54)

where \(E_{\text{edge}}, E_{\text{cons}},\) and \(E_{\text{pos}}\) are defined by (2.39), (2.40), and (2.41) respectively. Note that the value of \(E_{\text{edge}}\) defined via (2.39) lies between \(-1\) and 0; therefore, in order to make this component non-negative we add 1 to \(E_{\text{edge}}\) in (2.54).

The minimax principle minimizes (2.54) as follows:

\[
(P^*, Q^*, R^*) = \arg \min_{P, Q, R} \arg \max_{\mu_{\text{cons}}, \mu_{\text{pos}}} \left[E_{\text{nonlin-rsk}}(P, Q, R, \mu_{\text{cons}}, \mu_{\text{pos}})ight]
\]

\[= \arg \max_{\mu_{\text{cons}}, \mu_{\text{pos}}} \left[E_{\text{nonlin-rsk}}^*(\mu_{\text{cons}}, \mu_{\text{pos}})\right],\]

(2.55)

where

\[
E_{\text{nonlin-rsk}}^*(\mu_{\text{cons}}, \mu_{\text{pos}}) = \min_{P, Q, R} E_{\text{nonlin-rsk}}(P, Q, R, \mu_{\text{cons}}, \mu_{\text{pos}}).
\]

(2.56)

We know that the function \(E_{\text{nonlin-rsk}}^*\) is concave up [12]. In consequence, the parameter value set, \((\mu_{\text{cons}}^*, \mu_{\text{pos}}^*)\), corresponding to the minimax criterion, is now determined uniquely by (see [12])

\[
(\mu_{\text{cons}}^*, \mu_{\text{pos}}^*) = \arg \max_{\mu_{\text{cons}}, \mu_{\text{pos}}} E_{\text{nonlin-rsk}}^*(\mu_{\text{cons}}, \mu_{\text{pos}}).
\]

(2.57)

Unfortunately, multiple minimization computations for the energy functional (2.54) are required to determine the required parameter value set \((\mu_{\text{cons}}^*, \mu_{\text{pos}}^*)\). Since the function (2.56) is concave up, we can set up a simple “steepest ascent” type search method to crawl up the top of the surface \(E_{\text{nonlin-rsk}}^*\):
Algorithm 2.4.1

1. Start with initial values $\mu^*_{\text{cons}} = \mu^0_{\text{cons}}$ and $\mu^*_{\text{pos}} = \mu^0_{\text{pos}}$.

2. Choose a step size $h > 0$.

3. for $n = 1: \text{Max}\_\text{Iterations}$

   $$(\mu^n_{\text{cons}}, \mu^n_{\text{pos}}) = \arg \max_{\mu_1 \in \{\mu^{n-1}_{\text{cons}}-h, \mu^{n-1}_{\text{cons}}, \mu^{n-1}_{\text{cons}}+h\}} \arg \max_{\mu_2 \in \{\mu^{n-1}_{\text{pos}}-h, \mu^{n-1}_{\text{pos}}, \mu^{n-1}_{\text{pos}}+h\}} \left[ E^*_{\text{nonlin-rsnk}}(\mu_1, \mu_2) - E^*_{\text{nonlin-rsnk}}(\mu^n_{\text{cons}}, \mu^n_{\text{pos}}) \right]$$

4. Output: $\mu^*_{\text{cons}} = \mu^n_{\text{cons}}$ and $\mu^*_{\text{pos}} = \mu^n_{\text{pos}}$.

The loop in Algorithm 2.4.1 performs hill-climbing on the surface $E^*_{\text{nonlin-rsnk}}$. Instead of a maximum iteration value for the loop, the iterative process can also be terminated when the change between the values $(\mu^n_{\text{cons}}, \mu^n_{\text{pos}})$ and $(\mu^{n-1}_{\text{cons}}, \mu^{n-1}_{\text{pos}})$ becomes insignificant. Note that once $(\mu^*_{\text{cons}}, \mu^*_{\text{pos}})$ is found via Algorithm 2.4.1, we achieve the desired minimax solution for $P$, $Q$, and $R$ from (2.54).

Figure 2.14(a) shows the surface plot of $E^*_{\text{nonlin-rsnk}}$ vs $(\mu_{\text{cons}}, \mu_{\text{pos}})$ for a sample leukocyte image. Figure 2.14(b) shows a plot of solution quality vs $(\mu_{\text{cons}}, \mu_{\text{pos}})$. To find out the solution quality value for a set of values $(\mu_{\text{cons}}, \mu_{\text{pos}})$, we first compute (2.56) and then calculate the Pratt figure of merit (FOM) for the solution [13]:

$$FOM = \frac{1}{\max(N_d, N_i)} \sum_{n=1}^{N_d} \frac{1}{1 + \alpha d_n^2}, \quad (2.58)$$

where $N_d$ and $N_i$ are respectively the detected and the actual (true) number of edge points on an image, $d_n$ is the distance between the $n$th true edge point from its nearest detected edge point, and $\alpha$ is a weighting parameter ($\alpha = 1/9$ here). Pratt’s FOM is bound between 0 and 1, with unity representing the perfect segmentation. Note from Figs. 2.14(a) and 2.14(b) that the minimax parameter value approximately corresponds to highest solution quality.

On a side note, we introduce the Pratt figure of merit here as a quantitative measure of segmentation success. We believe that there is a striking absence of established quantitative measures of success in image analysis tasks. We feel that
FIGURE 2.14: (a) Energy values vs. weighting parameters. (b) Solution quality vs. weighting parameter values.
the establishment of such measures is critical to biomedical image analysis, where the results may affect the health of a patient or the acceptance of a novel drug.

2.6 DYNAMIC PROGRAMMING FOR SNAKE EVOLUTION

As an alternative to gradient descent based snake solutions, we explore dynamic programming (DP) [14]. DP is effective when not all variables are interrelated simultaneously in the energy functional that we seek to minimize. To make this point clear, let us consider the following energy functional of five variables, \( v_1, v_2, v_3, v_4, \) and \( v_5 \):

\[
E(v_1, v_2, v_3, v_4, v_5) = E_1(v_1, v_2) + E_2(v_2, v_3) + E_3(v_3, v_4) + E_4(v_4, v_5).
\]  

(2.59)

Let us assume that each decision variable \( v_i \) can take on only \( m \) possible values. So if we want to minimize \( E \) via an exhaustive search we first need to compute \( m^5 \) values of \( E \) for all possible combinations of the values of the five variables and then pick the combination yielding the minimum \( E \). In contrast, DP exploits the additive form of the energy functional (2.59) and solves the minimization in five sequential stages as follows:

\[
\min_{v_1, v_2, v_3, v_4, v_5} E(v_1, v_2, v_3, v_4, v_5) = \min_{v_5} \left( \min_{v_4} \left( \min_{v_3} \left( \min_{v_2} \left( \min_{v_1} \left[ E_1(v_1, v_2) + E_2(v_2, v_3) \right] + E_3(v_3, v_4) + E_4(v_4, v_5) \right] \right) \right) \right).
\]  

(2.60)

In other words, DP solves the minimization problem by generating a sequence of functions of single variable called optimal value functions:

\[
\begin{align*}
D_1(v_2) &= \min_{v_1} [E_1(v_1, v_2)], \\
D_2(v_3) &= \min_{v_2} [D_1(v_2) + E_2(v_2, v_3)], \\
D_3(v_4) &= \min_{v_3} [D_2(v_3) + E_3(v_3, v_4)], \\
D_4(v_5) &= \min_{v_4} [D_3(v_4) + E_4(v_4, v_5)], \\
D_5 &= \min_{v_5} [D_4(v_5)].
\end{align*}
\]  

(2.61)
FIGURE 2.15: Dynamic programming: optimal value functions in an example situation. The optimal path is shown with boldface arrows. Four values for each of the five variables \((v_1, v_2, \ldots, v_5)\) are possible. Consequently there are four possible values for each of the five optimal value functions \((D_1, D_2, \ldots, D_5)\). See text for description.

Since \(D_1(v_2)\) is a function of \(v_2\), which can assume \(m\) values, \(D_1(v_2)\) can be represented by an array of length \(m\). To compute \(D_1(v_2)\), for each value of \(v_2\), we find a value of \(v_1\) that yields the minimum value for \(E_1(v_1, v_2)\). This minimum value is assigned to the corresponding element of the \(D_1(v_2)\) array. Each element of the \(D_1(v_2)\) array also points to the corresponding value of \(v_1\) which has yielded the minimum \(E_1(v_1, v_2)\).

For the sake of illustration, let \(m = 4\). In Fig. 2.15 we show only the pointers from the \(D_1(v_2)\) array pointing to the index array: 1 through 4. By pointing the second element of \(D_1(v_2)\) array to value 1, we indicate that out of the four values \(E_1[v_1(1), v_2(2)], E_1[v_1(2), v_2(2)], E_1[v_1(3), v_2(2)], \) and \(E_1[v_1(4), v_2(2)]\), the minimum energy value is \(E_1[v_1(1), v_2(2)]\). Similarly, to indicate that \(E_1[v_1(2), v_2(3)]\) is the minimum of the four values \(E_1[v_1(1), v_2(3)], E_1[v_1(2), v_2(3)], E_1[v_1(3), v_2(3)],\) and \(E_1[v_1(4), v_2(3)]\), the third element of array \(D_1(v_2)\) points to value 2. To compute \(D_2(v_3)\), for each value of \(v_3\) we find the value of \(v_2\) that yields the minimum \([D_1(v_2) + E_2(v_2, v_3)]\). This minimum value is assigned to \(D_2(v_3)\) along with a pointer to the corresponding element of the \(D_1(v_2)\) array. In
Fig. 2.15, we show the pointers from $D_2(\nu_3)$. As an example, we point $D_2[\nu_3(2)]$ to $D_1[\nu_3(3)]$ to indicate that \{ $D_1[\nu_3(3)]+E_2[\nu_2(3), \nu_3(2)]$ \} is the minimum of the four values: \{ $D_1[\nu_3(1)]+E_2[\nu_2(1), \nu_3(2)]$ \}, \{ $D_1[\nu_3(2)]+E_2[\nu_2(2), \nu_3(2)]$ \}, \\
\{ $D_1[\nu_2(3)]+E_2[\nu_2(3), \nu_3(2)]$ \} and \{ $D_1[\nu_3(4)]+E_2[\nu_2(4), \nu_3(2)]$ \}. Arrays $D_3(\nu_4)$ and $D_4(\nu_5)$ are computed in like manner. Note that $D_5$ is a single element array pointing to $D_4[\nu_5(4)]$ indicating that $D_4[\nu_5(4)]$ is the minimum of the four values: $D_4[\nu_5(1)]$, $D_4[\nu_5(2)]$, $D_4[\nu_5(3)]$, and $D_4[\nu_5(4)]$. Now we can trace the path (shown in bold in Fig. 2.15) starting at $D_5$. Figure 2.15 shows a set of links in bold that depicts $\nu_1(2)$, $\nu_2(1)$, $\nu_3(1)$, $\nu_4(2)$, and $\nu_5(4)$ as the values of the five variables resulting in the minimum energy functional $E$ of (2.59).

Let us now analyze the computational complexity of the minimization of (2.59) via DP. In computing $D_1(\nu_2)$ we evaluate the objective function component $E_1(\nu_1, \nu_2)$ $m^2$ times. Similarly in computing $D_2(\nu_3)$, $D_3(\nu_4)$, and $D_4(\nu_5)$ it takes $m^2$ such evaluations of the minimands (the objective functions to be minimized) in each case. Finally in order to compute $D_5$, it takes $m$ evaluations of minimands. Thus, in total, DP takes $4m^2 + m$ evaluations, as opposed to $m^5$ evaluations resulting from the exhaustive search method. Likewise, if the objective function $E$ in (2.59) had $n$ constituent function components in total, then DP would take $(n-1)m^2 + m$ evaluations. So DP has a computational complexity of $O(nm^2)$ for minimizing (2.59) as opposed to $O(m^n)$ for the exhaustive search.

Let us now consider an objective function with the following form:

$$E(\nu_1, \nu_2, \nu_3, \nu_4, \nu_5) = E_1(\nu_1, \nu_2) + E_2(\nu_2, \nu_3) + E_3(\nu_3, \nu_4) + E_4(\nu_4, \nu_5) + E_5(\nu_5, \nu_1).$$

(2.62)

In this case DP generates the following optimal value functions of two variables:

$$D_1(\nu_1, \nu_3) = \min_{\nu_2} [E_1(\nu_1, \nu_2) + E_2(\nu_2, \nu_3)],$$

$$D_2(\nu_1, \nu_4) = \min_{\nu_3} [D_1(\nu_1, \nu_3) + E_3(\nu_3, \nu_4)],$$

$$D_3(\nu_1, \nu_5) = \min_{\nu_4} [D_2(\nu_1, \nu_4) + E_4(\nu_4, \nu_5)],$$

$$D_4 = \min_{\nu_5, \nu_1} [D_3(\nu_1, \nu_5) + E_5(\nu_5, \nu_1)].$$

(2.63)
Note that the computational complexity of DP, to minimize objective functions with the form of (2.61), is \( O(nm^3) \) where \( n \) is the total number of constituent energy terms. Sometimes the objective function is composed of constituent functions of three variables. An example is as follows:

\[
E(v_1, v_2, v_3, v_4, v_5) = E_1(v_1, v_2, v_3) + E_2(v_2, v_3, v_4) + E_3(v_3, v_4, v_5).
\]

(2.64)

DP then generates the following optimal value functions of two variables:

\[
\begin{align*}
D_1(v_2, v_3) &= \min_{v_1} [E_1(v_1, v_2, v_3)], \\
D_2(v_3, v_4) &= \min_{v_2} [D_1(v_2, v_3) + E_2(v_2, v_3, v_4)], \\
D_3(v_4, v_5) &= \min_{v_3} [D_2(v_3, v_4) + E_3(v_3, v_4, v_5)], \\
D_4 &= \min_{v_4, v_5} D_3(v_4, v_5).
\end{align*}
\]

The energy functional of the form (2.64) is referred to as energy functional of second order interaction terms, whereas (2.59) is known as energy functional of first order interactions. It is not difficult to see that DP minimizes objective functions with second order interactions in \( O(nm^3) \), where \( n \) is the total number of terms in the energy functional. But how do we apply DP to tracking with snakes?

For the sake of simplicity, we consider DP snake computation where the snake energy functional has only first order interaction terms as follows:

\[
E(X_0, \ldots, X_{n-1}, Y_0, \ldots, Y_{n-1}) = \frac{1}{2} \sum_{i=0}^{n} \alpha(X_{i+1} - X_i)^2 + \alpha(Y_{i+1} - Y_i)^2 - \sum_{i=0}^{n} f(X_i, Y_i).
\]

(2.65)

If the snake is a closed contour, then the functional (2.64) is similar to (2.61). In this case, DP leads to the following optimal value functions:

\[
\begin{align*}
D_0(X_0, Y_0, X_2, Y_2) &= \min_{X_1, Y_1} [E_0(X_0, Y_0, X_1, Y_1) + E_1(X_1, Y_1, X_2, Y_2)], \\
D_1(X_0, Y_0, X_3, Y_3) &= \min_{X_2, Y_2} [D_0(X_0, Y_0, X_2, Y_2) + E_2(X_2, Y_2, X_3, Y_3)], \\
& \vdots
\end{align*}
\]
The optimal snake location is found by tracing the variable values from $D_{n-1}$ to $D_0$. Note that in the snake computation the energy functional has to be minimized iteratively until the energy functional value reaches a local minimum. Therefore the optimal value functions (2.65) are generated in each iteration of snake computation until no further change in snake position occurs. A snake energy functional with second order interactions can be similarly handled via DP. However, the computational complexity will be slightly higher.

Figure 2.16 shows DP snake evolutions used to compute the microvessel boundary from an in vivo microscopy observation.
2.7 CONCLUSIONS

Our first tool for tracking the boundaries of biological objects is the traditional snake, powered by the expeditious mechanism of gradient descent. Where optimization variables can be sequentially treated, dynamic programming may provide a superior solution.

Several key concepts such as internal energy, external energy, and the snake force are necessary in mastering the snake. This chapter also puts forth a toolbox of snake external forces including the state-of-the-art gradient vector flow. For objects in motion, we prescribe a specially designed external force, motion gradient vector flow. The chapter provides Algorithm 2.1, which is a simple yet powerful method that can be exploited in many object-tracking applications. We show how the tracking algorithm can be tailored to certain biological objects using a shape–size–position constrained snake. Finally, a potential Achilles’ heel in snake tracking—how to choose the weighting parameter values in the energy functional—is tackled by way of the minimax method.