SRI International

Ziplock Snakes

September 19, 1994

Technical Note No. 548

Prepared By: W. Neuenschwander*, ETH
P. Fua†, SRI International
G. Székely*, ETH
O. Kübler*, ETH

Approved by: C. Raymond Perrault, Director
Artificial Intelligence Center
Computing and Engineering Sciences Division

* Communication Technology Laboratory
Swiss Federal Institute of Technology ETH
CH-8092 Zurich, Switzerland

† Computer Scientist for the Artificial Intelligence Center
Computing and Engineering Sciences Division

Ziplock Snakes

W. Neuenschwander*, P. Fua†, G. Székely*, and O. Kübler*

* Communication Technology Laboratory
Swiss Federal Institute of Technology ETH
CH-8092 Zurich, Switzerland

† SRI International
Artificial Intelligence Center
333 Ravenswood Avenue
Menlo Park, CA 94025, USA

Abstract

We propose a snake-based approach that lets a user specify only the distant endpoints of the curve he wishes to delineate without having to supply an almost complete polygonal approximation. We greatly simplify the initialization process and achieve much better convergence properties than those of traditional snakes by using the image information around these endpoints to provide boundary conditions and by introducing an optimization schedule that allows a snake to take image information into account first only near its extremities and then, progressively, toward its center. In effect, the snakes are clamped onto the image contour in a manner reminiscent of a ziplock being closed.

These snakes can be used to alleviate the often repetitive task practitioners face when segmenting images by abolishing the need to sketch a feature of interest in its entirety, that is, to perform a painstaking, almost complete, manual segmentation.

Keywords : Snakes, Deformable models, Interactive initialization, Boundary conditions.

1 Introduction

In recent years, snakes have emerged as a very powerful tool for semiautomated object delineation. They have been originated by Terzopoulos, Kass, and Witkin [Terzopoulos *et al.*, 1987, Kass *et al.*, 1988] and have since given rise to a large body of literature ([Fua and Leclerc, 1990, Leymarie and Levine, 1993, Cohen *et al.*, 1992, Staib and Duncan, 1992] among many others) that explores theoretical and implementation issues as well as new applications.

In most of these papers, however, it is assumed that the initial position of the snake is relatively close to the desired solution. While this is a reasonable assumption for applications such as motion tracking [Terzopoulos and Szeliski, 1992, Bascle and Deriche, 1993], it is ineffective for delineating complex objects from scratch.

Here, we describe a snake approach that allows a user to specify only the endpoints of the curve he wishes to delineate instead of a complete polygonal approximation. Our efforts are aimed at alleviating the often repetitive task practitioners face when segmenting images. In particular, we aim at abolishing the need to outline the desired structure very precisely, that is, to perform a painstaking, almost complete, manual segmentation. In our implementation, the user needs only to supply a few discrete points through which the contour must pass; the system then propagates the information along the contour starting from these points. As a result, considerably fewer control points are needed than for conventional implementations to which we refer as *traditional snakes.*

Traditional snakes are polygonal curves to which is associated an objective function that combines an "image term" that measures either edge strength or edge proximity, and a regularization term that accounts for tension and curvature. The curve is deformed so as to optimize the score and, as a result, to match the image edges. The optimization typically is global and takes edge information into account along the whole curve simultaneously. When the snake's initial position is far away from the desired result, the snake often gets stuck in an undesirable local minimum because it uses irrelevant edge information. The minimization problem is solved by treating the snake as a physical system evolving under the influence of the potential that is the sum of the objective function and a dissipation term that enforces convergence. This term tends to prevent the snake from moving far away from its current position. As a result, to ensure convergence to the desired answer, one must supply an initial position that is close to it.

By contrast, in our approach, the optimization progresses from the endpoints toward the center, thereby effectively propagating edge information along the curve without the need for a dissipation potential. The user-supplied endpoints and the automatically computed edge gradients in their vicinities serve as anchors. They are first used to compute, without using the image potential, an initial state that is approximately correct in each anchor's vicinity. The image term is then "turned on" progressively from the snake's extremities toward its middle section. Still using the endpoints as anchors, the snake's position is iteratively recomputed until the image term is fully turned on. As a result, the snake is progressively clamped onto an image contour so that it smoothly connects the two endpoints and has the right orientation at these points. This behavior is analogous to the closing of a ziplock, hence the name of our snakes.

The paper is organized as follows. We first review the properties of traditional snakes and introduce our *ziplock snakes.* We then compare their respective behaviors, and present results on

both synthetic and real images to demonstrate the improved performance of ziplock snakes for initializations that are far away from the desired result. Finally, we propose extensions to deal with ribbon-like structures and to facilitate interactive segmentation by using key-point information for initialization.

2 Traditional Snakes

The original snakes [Kass *et al.*, 1988] are modeled as time-dependent 2–D curves defined parametrically as

$$\vec{v}(s,t) = (x(s,t), y(s,t))_{0 \leq s \leq 1} , \tag{1}$$

where s is proportional to the arc length, t the current time, and x and y the curve's image coordinates. The snake deforms itself as time progresses so as to minimize an image potential $E_I(\vec{v})$, with

$$E_I(\vec{v}) = -\int_0^1 P(\vec{v}(s,t))\, ds \tag{2}$$

where $P(\vec{v}(s,t))$ is a function of the image. One typical choice is to take $P(\vec{v}(s,t))$ to be equal to the magnitude of the image gradient, that is

$$P(\vec{v}(s,t)) = |\nabla I(\vec{v}(s,t))| , \tag{3}$$

where I is either the image itself or the image convolved by a Gaussian kernel. We have shown [Fua and Leclerc, 1990] that, when P is defined in such a fashion, the points along a curve that minimizes E_I are maxima of the gradient in the direction normal to the curve wherever the curvature of the curve is small and the variation in the gradient magnitude is slow. Therefore, such a curve approximates edges well except at corners.

However, because the gradient magnitude can vary rapidly due to contrast changes and to noise, it has proven effective to either clip the derived potential force [Leymarie and Levine, 1993] or to replace the gradient magnitude by its logarithm [Fua and Leclerc, 1990]. Alternatively, one can take $P(\vec{v}(s,t))$ to be the Euclidean distance to the closest edge point in an edge map computed using an operator such as the Canny edge detector [Canny, 1986, Cohen *et al.*, 1992]. The results of all these approaches are very similar. Our implementation, as described in Section 3, uses the gradient magnitude of the Gaussian smoothed image.

Whatever the choice of P, $E_I(\vec{v})$ is typically not a convex functional. To overcome this problem, Terzopoulos *et al.* have proposed to introduce a regularization term $E_D(\vec{v})$ that is convex and to minimize a total energy term $E(\vec{v})$ that is the sum of $E_I(\vec{v})$ and $E_D(\vec{v})$. Using the thin-plate model, $E_D(\vec{v})$ is taken to be

$$E_D(\vec{v}) = \frac{1}{2}\int_0^1 \alpha(s)\left|\frac{\partial \vec{v}(s,t)}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 \vec{v}(s,t)}{\partial s^2}\right|^2 ds , \tag{4}$$

where $\alpha(s)$ and $\beta(s)$ are arbitrary functions that regulate the curve's tension and rigidity. For generality's sake, we show that our framework handles α and β functions that are nonconstant.

Although the behavior of snakes can be made extremely flexible in this way, we assert that tension and rigidity parameters should not be allowed to vary in an unconstrained manner. It is the mark of good theory to use considerably fewer free parameters than there are degrees of freedom. In the implementation described below $\alpha(s)$ and $\beta(s)$ are taken to be constant and supplied by the user. We have shown previously [Fua and Leclerc, 1990] that constant α and β can be chosen in a fairly image-independent way.

Variational calculus [Courant and Hilbert, 1989] shows that if v minimizes $E = E_D + E_I$ and is sufficiently regular, that is at least $C^4(0,1)$, then it must be a solution of the Euler differential equation

$$-\frac{\partial}{\partial s}\left(\alpha(s)\frac{\partial \vec{v}(s,t)}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta(s)\frac{\partial^2 \vec{v}(s,t)}{\partial s^2}\right) = -\nabla P(\vec{v}(s,t)) \tag{5}$$

Note that, in order for this equation to have a unique solution, one must specify boundary conditions such as the values and derivatives of $\vec{v}(s,t)$ for $s = 0$ and $s = 1$.

In practice, to implement the minimization $E(\vec{v})$, the curve $\vec{v}$ is discretized by sampling it at regular intervals. We therefore take $\vec{v}$ to be a polygonal curve defined by a set of $n+1$ x and y vertices

$$\vec{v}^t = (x_i^t, y_i^t)_{0 \le i \le n} \,. \tag{6}$$

Using finite differences, the snake energy $E(\vec{v})$ becomes

$$\begin{aligned} E(\vec{v}) &= E_I(\vec{v}) + E_D(\vec{v}) \\ E_I(\vec{v}) &= -\sum_i P(x_i^t, y_i^t) \\ E_D(\vec{v}) &= 1/2 \sum_i \alpha_i((x_i^t - x_{i-1}^t)^2 + (y_i^t - y_{i-1}^t)^2) \\ &+ 1/2 \sum_i \beta_i((2x_i^t - x_{i-1}^t - x_{i+1}^t)^2 + (2y_i^t - y_{i-1}^t - y_{i+1}^t)^2) \,. \end{aligned} \tag{7}$$

Note that $E_D(\vec{v})$ is quadratic and can be rewritten as

$$E_D(\vec{v}) = 1/2 X_t^T K X_t + 1/2 Y_t^T K Y_t \,, \tag{8}$$

where K is a $(n+1) \times (n+1)$ pentadiagonal matrix, and X and Y are the vectors of the x and y vertex coordinates.
A curve that minimizes the energy E must be such that

$$\begin{aligned} 0 &= \frac{\partial E}{\partial \vec{v}} \\ &= \frac{\partial E_D}{\partial \vec{v}} + \frac{\partial E_I}{\partial \vec{v}} \,. \end{aligned} \tag{9}$$

Since the deformation energy E_D in Equation 8 is quadratic and decouples the x and y coordinates of the curve, Equation 9 can be rewritten as a system of two linear equations

$$KX = F_X \quad , \quad KY = F_Y \tag{10}$$

which are coupled by the "image forces," F_X and F_Y,

$$F_X = -\frac{\partial E_I}{\partial X} \quad , \quad F_Y = -\frac{\partial E_I}{\partial Y} \quad . \tag{11}$$

These image forces are the $n+1$ vectors

$$(\frac{\partial P(x_0, y_0)}{\partial v}, \frac{\partial P(x_1, y_1)}{\partial v}, ..., \frac{\partial P(x_n, y_n)}{\partial v}) \tag{12}$$

where v stands for either x or y. Note that F_X and F_Y depend on the snake's position, making the system semilinear. Equivalently, the Euler differential equation can be rewritten as a set of two coupled differential equations

$$\begin{aligned} -\frac{\partial}{\partial s}\left(\alpha(s)\frac{\partial x(s,t)}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta(s)\frac{\partial^2 x(s,t)}{\partial s^2}\right) &= -\frac{\partial P}{\partial x} \\ -\frac{\partial}{\partial s}\left(\alpha(s)\frac{\partial y(s,t)}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta(s)\frac{\partial^2 y(s,t)}{\partial s^2}\right) &= -\frac{\partial P}{\partial y} \end{aligned} \tag{13}$$

The discretization of the system of Equation 13 using finite differences also leads to the system of Equation 10 [Leymarie and Levine, 1993]. The matrix K, however, is not invertible and these equations cannot be solved directly. This stems from the fact that the Euler differential equation (Equations 5 and 13) has a unique solution only when boundary conditions are supplied. Without them, the system is underconstrained.

In Section 3, we will show how we can solve this system of equations by supplying the boundary conditions. This will be one of the key differences between our approach and more traditional snake implementations. Traditionally, the minimization of $E(\vec{v})$ is achieved by first generalizing to a dynamical physical model in which the snake is endowed with mass and assumed to move in a viscous medium, and then solving the equation of the dynamics. This amounts to adding a kinetic energy term to the regularization term $E_D(\vec{v})$ and a Rayleigh dissipation functional to the energy and leads to solving the following differential equation [Terzopoulos and Szeliski, 1992]:

$$\frac{\partial E}{\partial \vec{v}} + \mu\frac{\partial^2 \vec{v}(s,t)}{\partial t^2} + \gamma\frac{\partial \vec{v}(s,t)}{\partial t} = 0 \, , \tag{14}$$

where μ represents the mass density of the snake and γ is the viscosity of the medium. Both mass and damping are assumed to be independent of position. As in the case of Equation 10, after time discretization using backward and central differences, this can be rewritten as a set of two equations in X and Y:

$$\begin{aligned} (K + (\gamma+\mu)\, I)\; X_t &= (\gamma + 2\mu)\; X_{t-1} - \mu X_{t-2} + F_X \\ (K + (\gamma+\mu)\, I)\; Y_t &= (\gamma + 2\mu)\; Y_{t-1} - \mu Y_{t-2} + F_Y \end{aligned} \tag{15}$$

where K, F_X, and F_Y are defined in Equation 10. The matrix $(K + (\gamma+\mu)\, I)$ is positive definite for $\mu + \gamma > 0$ and, because it is pentadiagonal, the solution of this set of equations can be computed efficiently in $O(n)$ time using LU decomposition and back-substitution. Note that the LU decomposition need be recomputed only when the parameters α and β change.

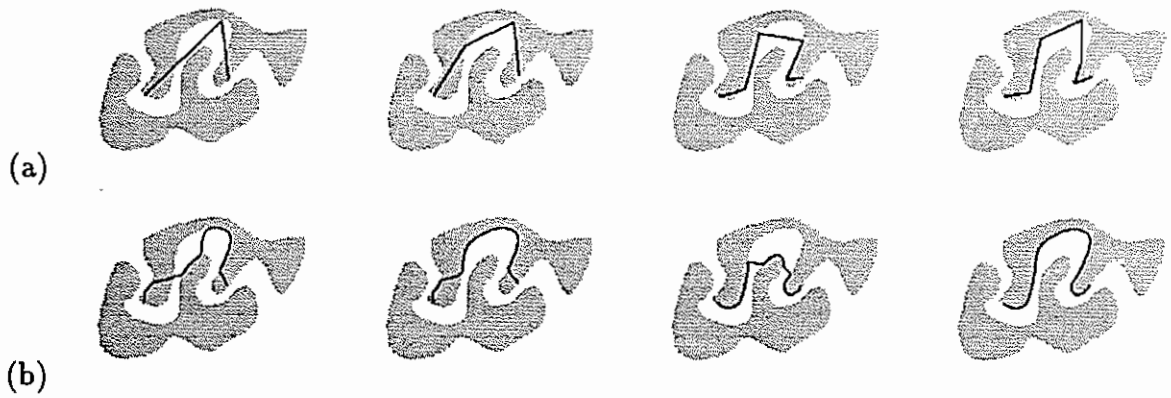


Figure 1: Sensitivity of traditional snakes to nearby contours. (a) Different initializations: the snake is initialized using a polygon with an increasing number of vertices. (b) The corresponding results: the fact that the two objects are lying close to each other forces the user to outline the desired contour segment precisely. If the snake touches the influence region of a nearby object it will get stuck on the wrong contour.

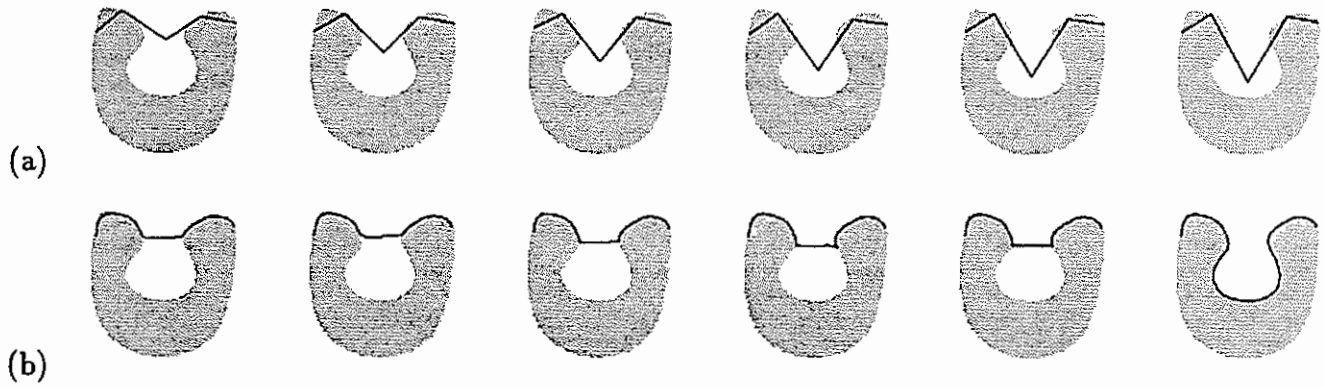


Figure 2: Ill-conditioned behavior of the traditional snakes with respect to initialization. (a) Slightly different initializations: the snake is initialized using a polygon with five vertices. While the first, second, fourth, and fifth vertices are the same for all six situations, the the third vertex moves closer to the shape. (b) The corresponding results: when the third vertex is close enough to the object's border, the snake is able to detect the correct contour (6th image pair). However, it is not intuitively obvious what the threshold is.

The traditional snakes are effective when the initial position of $\vec{v}$ is close from the desired solution. However, they are very sensitive to initial conditions. They can easily get caught in local minima when the desired outline presents large concavities that force the snake to extend itself or

when there are other edges in the vicinity of the desired one that may "catch" the snake as shown in Figure 1. Furthermore, from a user's point of view, there is not always a simple intuitive way of defining how close an initial state has to be in order to detect the correct object boundary. Figure 2 illustrates this problem.

This is to be expected since the Rayleigh dissipation functional constrains the result to be the minimum of the objective function that is in some sense "closest" from the initial position. To a certain extent, the kinetic energy can be expected to help the snake move across shallow minima in the potential surface $P(\vec{v})$. While the tracking of contours in image sequences may benefit from such a kinetic energy term [Terzopoulos and Szeliski, 1992], we have observed that as far as extracting features from static images is concerned, the effect of this term is marginal.

In the next section, we introduce a different breed of snakes — the ziplock snakes — that alleviate these problems by resorting directly to Equation 5 and solving it by supplying appropriate boundary conditions.

3 Ziplock Snakes

To improve upon the snakes' convergence properties, we have sought to introduce mechanisms that better reflect the nature of the problem than the indiscriminate global optimization described above. Also, we should take into account the type of input that a user can provide with reliability and facility. In our implementation, we assume that the user specifies snake endpoints in the vicinity of clearly visible edge segments, which implies a well-defined edge direction. As explained below, the system actually optimizes the location of the user-supplied points to ensure that they are indeed good edge points, and extracts the associated edge directions. It then becomes natural to use these anchor elements as boundary conditions and to propagate the edge information along the curve starting from them.

We use the anchor elements to derive an initial position for the snake independently of the image under consideration. This initial state will, in general, be close to the desired answer in the vicinity of the snake endpoints but nowhere else. We will therefore "turn on" F_X and F_Y, the image forces of Equation 12, in those areas and compute a new position for the snake using the same boundary conditions as before. A longer part of the new solution will be closer to the actual image edge than before; the image forces can then turned on on this longer part and the snake's position recomputed. By iterating this process, we eventually turn on the image forces over the whole length of the snake, thereby achieving the propagation of the edge information from the anchor points to the snake's middle and the progressive ziplock-style clamping. Our approach is closely related to perturbation theory [Bush, 1992]: we start with an unperturbed solution of our minimization problem and progressively perturb it by considering more and more of the image forces.

In the remainder of this section, we first discuss the use the boundary conditions to solve our minimization problem. We then derive our initial unperturbed snake position from the endpoints and finally we present the optimization schedule that defines the gradual "turning on" of the image forces.

3.1 Solving the Minimization Problem with Boundary Conditions

As discussed in Section 2, minimizing the snake's energy amounts to solving the Euler differential Equation 5 (or Equation 13) which leads, after discretization, to the semilinear system of Equation 10. In the appendix we show that by fixing the curve's endpoint (x_0, y_0) and (x_n, y_n) and giving the curve's tangent at those points, the above system of equations reduces to

$$\begin{aligned} K'X' &= F'_X \\ K'Y' &= F'_Y \,, \end{aligned} \tag{16}$$

where X' is the $n-1$ vector $(x_1.....,x_{n-1})$, Y' the $n-1$ vector $(y_1.....,y_{n-1})$, and K' an $(n-1)\times(n-1)$ pentadiagonal matrix that is now invertible. Of course, since F'_X and F'_Y depend on the snake's current position, the system is still only semilinear and cannot, in general, be solved in closed form. Instead, one must use an iteration scheme that can be concisely rewritten as

$$\begin{aligned} K'X'_t &= F'_X|_{X'=X'_{t-1},Y'=Y'_{t-1}} \\ K'Y'_t &= F'_Y|_{X'=X'_{t-1},Y'=Y'_{t-1}} \end{aligned} \tag{17}$$

where t is the iteration index.

3.2 Initialization

To successfully optimize our snake, we must start from an initial position that is approximately correct in the neighborhood of the endpoints. The easiest way to achieve this result is to solve the homogeneous equations that correspond to the system of Equation 13. As discussed in Section 2, we take α and β to be constant, and the homogeneous system becomes

$$-\alpha\frac{d^2v(s)}{ds^2} + \beta\frac{d^4v(s)}{ds^4} = 0 \tag{18}$$

where v stands for either x or y and $0 \leq s \leq 1$. Below, we derive boundary conditions for this equation and its analytical solution. By construction, this solution has the specified tangent at the endpoints and is close to the right answer near these points. It can therefore serve as an initial curve for the following minimization of the energy functional.

3.2.1 Image-compatible Boundary Conditions

When interactively defining the snake's endpoints, which we will subsequently refer to as the head and tail, the user may miss the accurate edge position. It is safe to assume, however, that these points are chosen close to dominant edge fragments. Hence we first perform a linear search to find the true edge location in the near neighborhood of the selected startpoints and endpoints. The snake tangent at its head and tail is then taken as the valley direction in the potential surface corresponding to the selected contour fragments. It is computed by optimizing the orientation of a short straight-line snake.

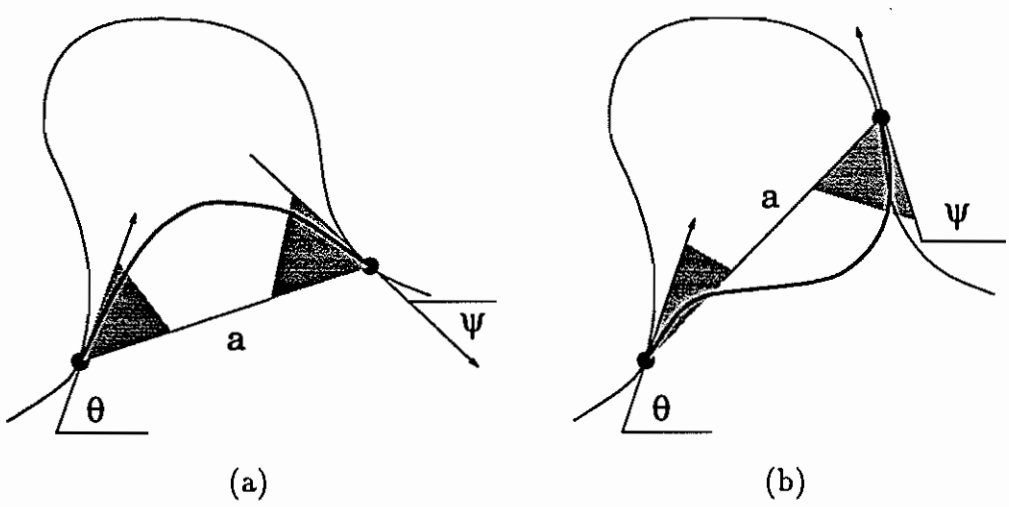


Figure 3: Defining the boundary conditions. The initial snake defines acute angles (light shaded) with the straight line a between the two endpoints. (a) The two endpoints are selected such that the acute angles are on the same side of a. The snake is initialized correctly. (b) In this case the acute angles are lying on opposite sides of a. Our heuristic initialization fails.

While the tangent direction at the endpoints can be computed, its orientation cannot be determined, as illustrated by Figure 3. By default, the boundary conditions are chosen so that the initial snake defines acute angles with the line joining the two endpoints. Since this heuristic may fail, we provide the user with the possibility of flipping the orientation at both ends. This could be automated by initializing the snake in the four possible ways (two possible orientations at each end) and retaining the best final result.

3.2.2 Analytical Solution of the Homogeneous Euler Equation

The homogeneous Euler Equation 18 has analytical solutions of the form

$$v(s) = C_1 + C_2 s + C_3 e^{-\sqrt{\frac{\alpha}{\beta}}s} + C_4 e^{+\sqrt{\frac{\alpha}{\beta}}s} \tag{19}$$

where $C_1, \ldots, C_4$ are integration constants that can be determined uniquely by the four boundary conditions

$$v(0) = A, \quad v(1) = B, \quad v'(0) = a, \quad v'(1) = b\,.$$

Let $w = \sqrt{\alpha/\beta}$ and $q = e^w$. We can now write

$$v(s) = C_1 + C_2 s + C_3 e^{-ws} + C_4 e^{+ws}, \quad v'(s) = C_2 - C_3 w e^{-ws} + C_4 w e^{+ws}\,,$$

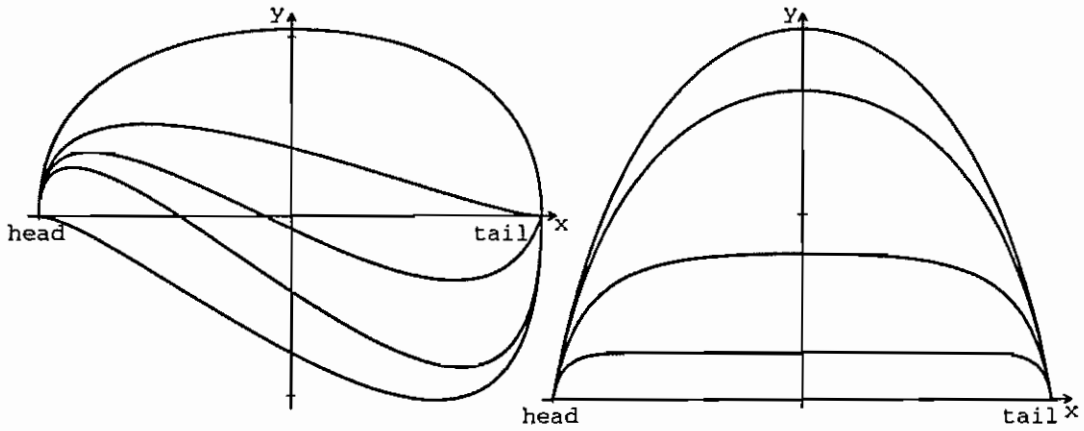
and the four constants $C_1, \ldots, C_4$ must satisfy the system of linear equations:

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & q^{-1} & q \\ 0 & 1 & -w & w \\ 0 & 1 & -wq^{-1} & wq \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{pmatrix} = \begin{pmatrix} A \\ B \\ a \\ b \end{pmatrix}$$

Recall that Equation 18 and its solution in Equation 19 are defined for $0 \leq s \leq 1$, and v stands for both the x and y coordinates. Therefore, the user-supplied endpoints are mapped from the image coordinate system into a normalized coordinate system. To get the boundary conditions for the uncoupled system, we have to split the tangent angles at the head and the tail into x and y components. Let θ and ψ, respectively, be the tail's and head's tangent angles. The parametric snake definition of Equation 1 requires the following two equations to hold:

$$\tan(\theta) = \frac{y'(0)}{x'(0)} = \frac{a_y}{a_x} \quad ; \quad \tan(\psi) = \frac{y'(1)}{x'(1)} = \frac{b_y}{b_x}$$

where a_x, b_x are the two first order boundary conditions for the x component and a_y, b_y the conditions for the y component. In practice, depending on the value of θ, we take either a_x or a_y to be ± 1 and use the above ratio to derive the other a. We compute b_x and b_y similarly.



(a) (b)

Figure 4: Different solutions of the homogeneous Euler Equation. (a) Solutions for various boundary conditions and constant w. (b) Solutions for decreasing parameter w and constant boundary conditions.

The solution of Equation 18 depends only on the ratio $w = \sqrt{\alpha/\beta}$ and the boundary conditions. Figure 4 shows the wide range of different curves that can be covered by varying these parameters.

3.3 Optimization Procedure

We start the optimization of the energy term by defining the initial snake as the solution of the homogeneous differential system of Equation 18. At this stage the snake "feels" absolutely no external potential forces. During the ongoing iterative optimization process the image potential is turned on progressively for all the snake vertices, starting from the extremities. Assuming that the user selects the endpoints near dominant edge fragments in the image, this initialization ensures that the snake already lies close to its optimal position at both ends.

We define the *force boundaries* as the location of the vertices farthest away from the endpoints that feel the image forces. These boundaries approach each other during the ongoing optimization process by moving forward at every iteration step if the displacement between the last two iterations is below a certain threshold.

Starting the optimization from both endpoints, we can proceed in two possible ways:

1. By moving the force boundaries we can fix all the snake elements between these boundaries and both endpoints. We can then iteratively redefine new boundary values and plug them into Equation 18. Therefore, we construct a "static solution" of Equation 5 out of piecewise solutions of Equation 18. At the end of the optimization process we release all snake elements and take this solution as an initial contour for a traditional-snake global optimization. Simple observation reveals the quality of the initialization: we regard it as good when the global optimization produces no noticeable deviations to the initial state while the quality is questionable whenever large changes occur, just as observed for the case of poor manual intialization of traditional snakes. The user is then advised to supply new, less separated, anchor points and to start the procedure again.

2. Alternatively, we can introduce, for each snake element between the force boundaries and both endpoints, a viscosity factor that actually defines a blending function between the old and the new snake positions. Note, however, that the middle part of the snake, for which the potential field and the viscosity have not yet been switched on, is still computed by solving the Euler equation of Equation 18. The last two snake elements on either side of the force boundaries serve implicitly as boundary values for the differential equation.

The first method incrementally builds a solution by locally optimizing the boundary values and only at the end performs global optimization across the whole snake. While this seems like an obvious way to implement the clamping, it has the problem that there is no mechanism to recover from mistakes. A solution that starts to go off track will keep on doing so and will stray further and further from the desired answer. However, this problem can be alleviated by subjecting the partially clamped ziplock snake to traditional snake optimization at regular intervals. By freeing the snake vertices, one allows it to wiggle into the best local minimum in its immediate vicinity, thereby correcting small deviations and preventing them from growing into large ones.

The second method achieves this by performing global optimization during the iteration and, as a result, converges much better. We define a varying viscosity factor $\gamma(s,t)$ such that $\gamma(s,0) \equiv 0$, $s \in [0,1]$. The initial coordinate vector $(X_0', Y_0')^T$ is the solution of Equation 18. We then

iteratively solve the following system for $t = 1, 2, 3, \ldots$

$$(K' + \gamma_t I)\, X'_t = \gamma_t\, X'_{t-1} + F'_X\Big|_{\substack{X' = X'_{t-1} \\ Y' = Y'_{t-1}}}$$

$$(K' + \gamma_t I)\, Y'_t = \gamma_t\, Y'_{t-1} + F'_Y\Big|_{\substack{X' = X'_{t-1} \\ Y' = Y'_{t-1}}}$$

The viscosity $\gamma(s,t)$ is recomputed each time the force boundaries move so that the initial displacement of each vertex is on the average of a given a-priori magnitude, typically 1 pixel. The damping factor $\gamma(s,t)$ is then increased progressively until the snake stabilizes and the force boundaries can be moved again.

4 Discussion

We use synthetic and real images of tutorial character to compare the traditional snakes with our ziplock snakes and show that initialization of the former must typically be much closer to the desired answer to achieve comparable results. To make the comparison fair, we use the same tension and rigidity parameters, α and β as defined in Equation 4, for both kinds of snakes.

Figure 5 shows three sequences of snakes evolving on a synthetic image. The first row (a) illustrates the behavior of our model. The ziplock snake can always be initialized with only two points. While the traditional snake undergoes a global deformation, we observe that the ziplock snake is only deformed locally. During the iteration, the force boundaries move along the contour until they meet. Sequence (b) illustrates the deficient behavior of the traditional snake initialized in exactly the same way. The third sequence (c) shows the traditional snake now initialized by a polygon with five vertices. These additional points are needed for traditional snakes to succeed.

The ability of ziplock snakes to trace the cavity stems from the gradual taking into account of the image forces from the ends. Conversely, the poor performance of the traditional snakes is a direct consequence of the immediate onset of those forces along the whole contour. The initialization of the traditional snake did not provide enough length; the penalty on expansion then prevented it from adequately conforming to the contour. The ability of ziplock snakes to outline cavities and to distinguish between nearby objects is further illustrated by Figure 6.

Figure 7 compares the performance of ziplock and traditional snakes, using an image of an apple. Although the object looks quite simple, the segmentation problem is surprisingly treacherous. Shadows and illumination effects produce edges having nothing to do with the apple's outline. Sequence (a) shows the initialization and optimization of a ziplock snake. The two endpoints are selected on the apple's outline and the snake eventually deforms to the correct shape, even though, depending on the choice of α and β, the initial guess may be very far from the final answer. Sequence (b) illustrates the bad performance of the traditional snake model initialized using only the same two endpoints: the snake gets stuck on the bright specular reflection on the apple's skin. In sequence (c), the initialization is closer to the desired answer but the snake is still attracted by other salient features near the contour and fails to outline the apple. To achieve this result using a

traditional snake, one must supply the almost "correct" polygonal approximation of Sequence (d). It is so close to the desired answer that only slight smoothing occurs during the evaluation, without any real position adjustment. In this case, the traditional snake acts only as a regularization tool smoothing out the first-order discontinuities of the initialization .

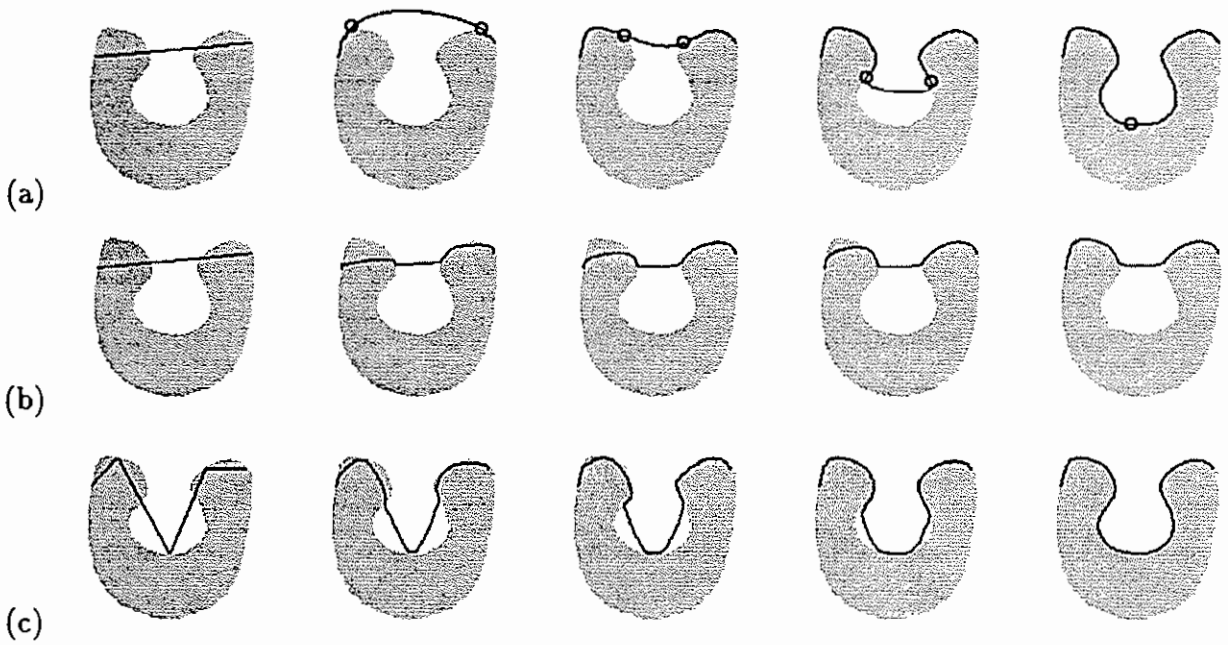


Figure 5: The effect of initialization for ziplock and traditional snakes. The rows show the evolution for different snakes from the initial state in the first column. (a) Evolution of a ziplock snake. The circles indicate the movement of the *force boundaries* during optimization. (b) Traditional snake with same initialization. (c) Traditional snake initialized in three additional points.

Figure 6: Evolution of a ziplock snake on the synthetic image of Figure 2. The circles denote the farthest vertices away from the endpoints for which the image forces are turned on, that is the *force boundaries* of Section 3.3. As discussed there, the optimization stops when the two circles meet.

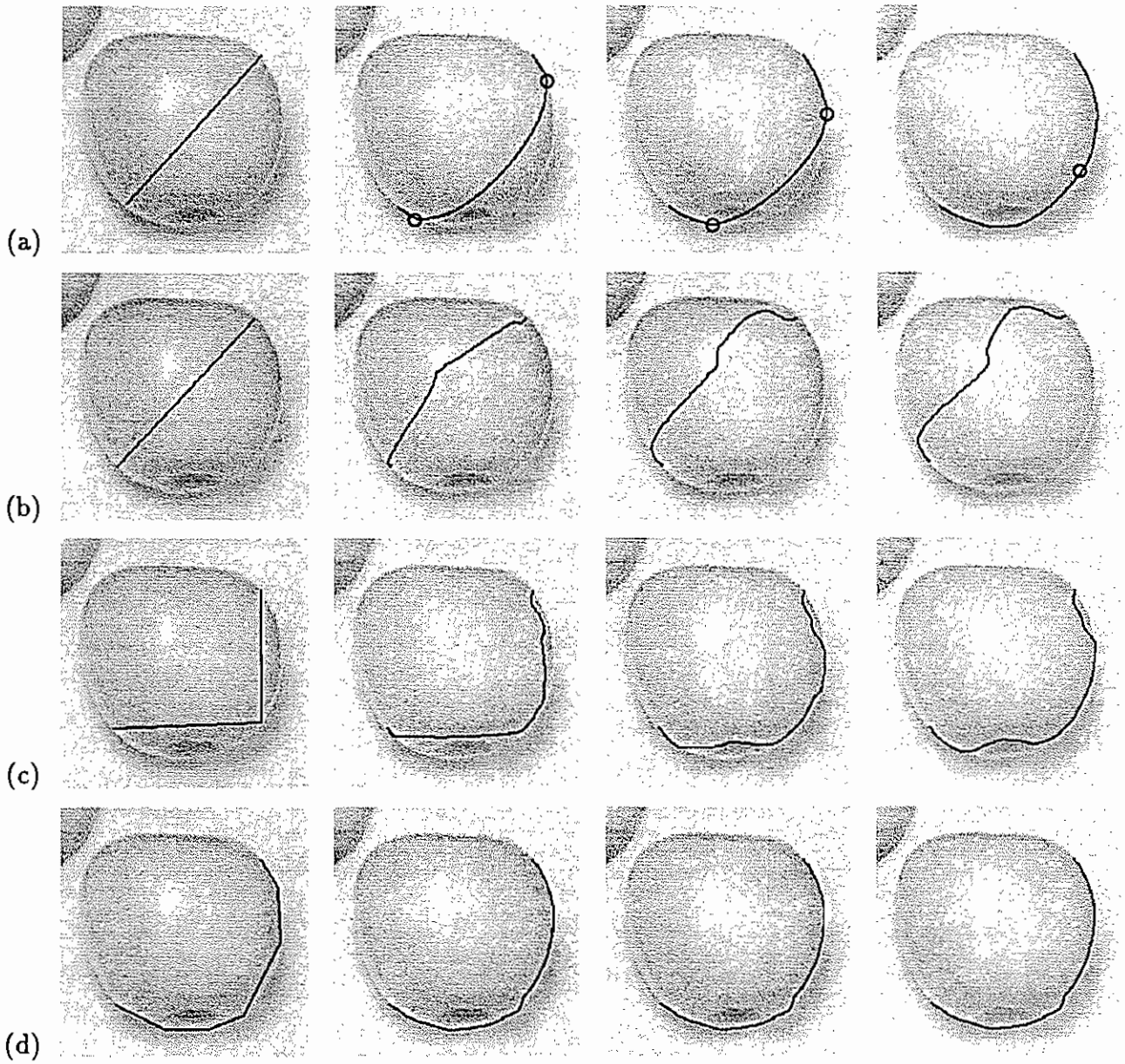


Figure 7: Compared behaviors of ziplock and traditional snakes on an image of an apple. The rows show the initialization and three stages of the optimization process for (a) a ziplock snake, (b) a traditional snake with the same initialization, (c) a traditional snake with three vertices, (d) a traditional snake with seven vertices.

5 Results

The contour-following capability of our ziplock snakes makes them especially useful for segmenting scenes where the separation of nearby features is necessary. Figure 8 illustrates, on the previously used image of the apple, how easily the contour of the apple and its shadow can be separated. The first row depicts the extraction of the apple's contour. To outline the shadow as shown in the second row, the initial points simply have to be moved into its vicinity.

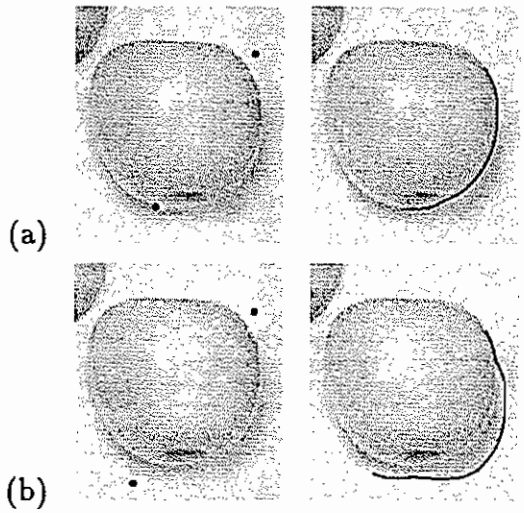
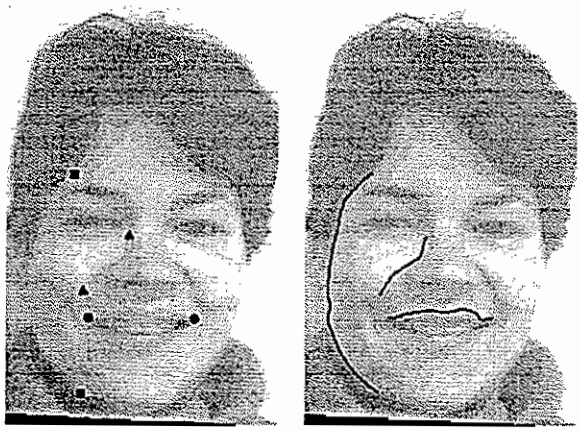


Figure 8: Detection of different image features by ziplock snakes. (a) Detection of the apple's outline. Some liberty can be used in placing the upper right initial point since there is no danger of ambiguity in the automatic adjustment to nearby image structures. More precision is required for the lower left initial point in order to avoid confusing apple and shadow boundaries. (b) Detection of the projected shadow; the lower left initial point is now closer to the shadow boundary.

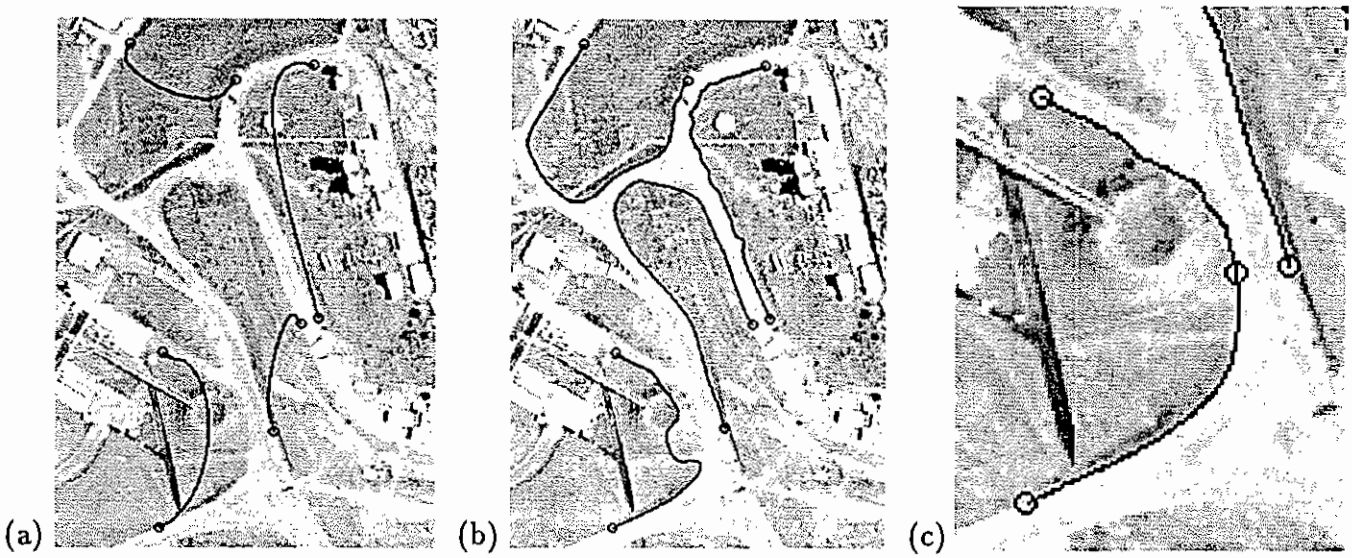
Figure 9 demonstrates that the contour-following property does not impair the gap-closing capability of ziplock snakes by showing the extraction of some low-contrast contours on an image of a face.

Figure 10 shows that ziplock snakes can be used to delineate roads in an aerial image by using very distant endpoints. Note, however, that our snakes can still become confused in the presence of junctions. In the case of the curve drawn in the bottom left corner of Figure 10(b), there is a secondary road that leaves from the main one and traps the snakes in an undesirable local minimum. In practice, when such problems arise, our interface allows the user to add a new point in the middle of the curve, thereby splitting it into two snakes.



(a) (b)

Figure 9: Outlining facial features. (a) Three pairs of endpoints on a face image (Courtesy of INRIA). (b) Final results.



(a) (b) (c)

Figure 10: Outlining roads in an aerial image. (a) Ziplock snake initializations. (b) Final results. All the road edges are correctly outlined except the bottom left one. (c) The erroneous result is corrected by adding one new control point.

Another mechanism for the user to interact with snakes and fix problems such as the one described above is to allow him to use a mouse-controlled cursor to nudge the snake off the unintended feature. Alternatively, we have extended our formulation of the snake equations to include a term related to the length and direction of the vector from each snake vertex to the mouse-cursor. In

this way, we have simulated a magnetic force between the cursor and the snake, which allows the user to push the curve in a desired direction.

To segment more complex shapes we need a simple way to sequentially define endpoints for adjoining open snake fragments. We have implemented an interactive initialization tool in which the contour-finding process starts as soon as the user has clicked on two initial points. Sequentially adding salient points will then extend the initial segment under immediate visual control in a sort of contour-following rubber line process. This mechanism can also be utilized to detect closed contours since the boundary conditions can be shared between neighboring snakes to guarantee a smooth transition between them. Figure 11 illustrates this process on the segmentation of a corpus callosum in an MR image. Note, in particular, how the correct outline is found by the second segment in spite of its crossing over a wrong contour during the initialization phase.

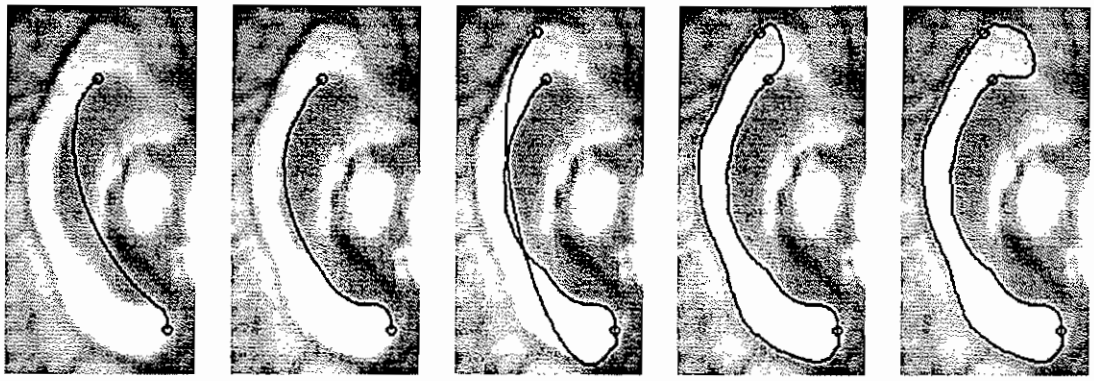


Figure 11: Segmentation process of a *corpus callosum* (MR image slice turned by 90°). The closed contour is built up by three open snakes linking three points supplied sequentially by the user and denoted by the circles.

6 Extensions

6.1 Ribbon Snakes

Following [Fua and Leclerc, 1990], we have implemented a tool for interactive road delineation, called *ribbon snake*. The model vector $\vec{v}$ is augmented by a third component, the varying width $w(s,t)$ of the road. The expression for the deformation energy (4) still holds for $\vec{v}(s,t) = (x(s,t), y(s,t), w(s,t))^T$, where the width is subject to the same "tension" and "rigidity" constraints as the two coordinate components. The ribbon forms the center of the road, while the assigned width defines two curves that are the actual deforming road boundaries. Note that the image information is taken into account along these two curves only. Figure 12 depicts the delineation of complete road boundaries using the ribbon snake model. Each ribbon is initialized by the two corresponding endpoints and the road direction, which is the average direction of the left and right road boundary.

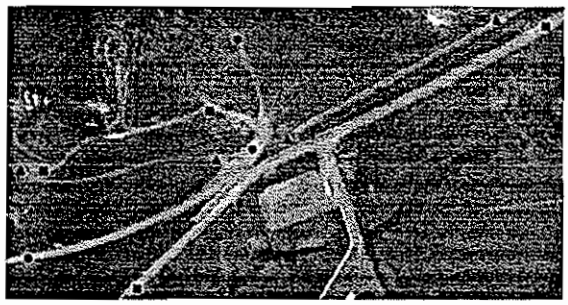
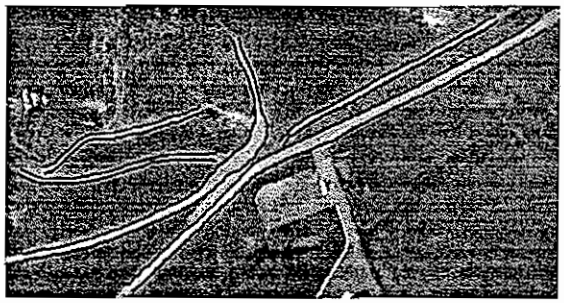
(a)  (b) 

Figure 12: Delineating roads on an aerial image using ribbon snakes. (a) Several pairs of endpoints. (b) Final detected road segments. Note that the actual ribbon, the centerline of the road, is not shown in the image on the right.

6.2 Snake Initialization by Key-points

A general drawback of the snake model is the difficulty of adapting to boundaries with strong orientation discontinuities. For this reason, Terzopoulos [Terzopoulos, 1987] proposed to introduce tangent discontinuities on the snake by setting the parameter $\beta(s^*)$ to zero for specific s^*. The snake can be bent arbitrarily at such points because high curvature is not penalized anymore. Even though this parameter manipulation is possible, the detection of correct breakpoints along the snake is not trivial. The difficulties are compounded at contour junctions, which are notoriously hard to detect.

In [Rosenthaler *et al.*, 1992], we have described a method to extract strong 2-D intensity variations (such as corners, junctions, strong curvature, ...) from the image. These specific image configurations have been subsumed under the concept of *key-point*. The detection algorithm computes their positions as well as the number and directions of attached contour segments. The connectivity between key-points can then be established and encoded in a graph-like data structure as described in [Henricsson and Heitger, 1994].

We have implemented a supervised contour tracking scheme that combines this connectivity graph with ziplock snakes so as to preserve corners and junctions. Connected pairs of key-points serve as initializing anchor elements. An outline is generated by using the snakes to connect them in succession. First-order continuity at the key-point is naturally controlled by the orientations of the emanating contour segments encoded in the graph-like data structure.

Figure 13 illustrates how key-points and their correspondences can be used to extract object boundaries in a semiautomated fashion. Figure 13(a) shows an image of a set of peppers with key-points overlayed as black circles. In this experiment, the goal was to get the outline of the long, light-gray capsicum. By selecting an arbitrary key-point (indicated by the arrow in Figure 13(a)) on the image, all possible direct connections with this point are shown and the user can select one of them. When he does, a snake is automatically initialized and the optimization procedure starts. As soon as the snake has reached a steady state, all key-points that are connected with one of the snake's endpoints are searched for. As long as there is a unique ongoing direction, the cycle of initialization followed by optimization is carried out autonomously. Only at branching points does

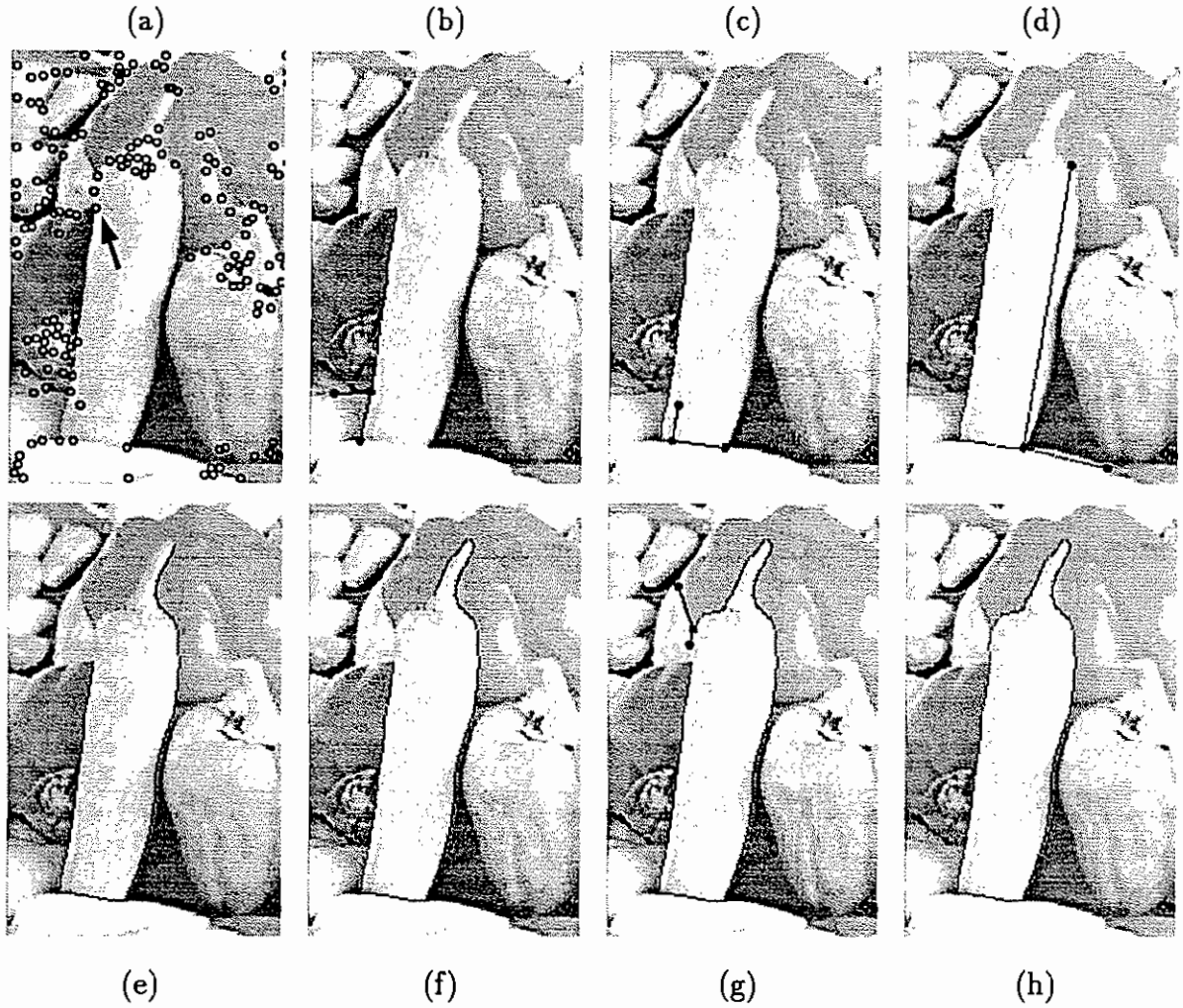


Figure 13: Tracking algorithm with snakes based on key-point information. (a) Image with overlayed key-points. (b) Optimized snake, supplying two alternative tracking directions. (c) (d) (e) (f) and (g) Intermediate steps as tracking proceeds. (h) Final pepper's outline consisting of ziplock snakes linked by key-points (not displayed).

the algorithm stop. It then displays the alternative tracking directions indicated by the rays shown in Figures 13(b), (c), (d), and (g). The user then has to choose the path to follow by interactively selecting one specific connection. Image 13(h) shows the final closed outline consisting of several ziplock snakes linked together at key-points.

7 Conclusion

We have proposed a snake-based approach to semiautomated delineation that allows a user to outline an open contour by specifying only very distant endpoints and allowing the computer to propagate edge information from the extremities toward the center. Our approach presents two main differences from more traditional snake implementations:

- It relies on the original time-independent snake formulation and derives boundary conditions from the image to constrain the minimization.
- It uses a schedule that allows the snake to take image information into account first only near its extremities and then, progressively, toward its center.

Combining these two ideas yields excellent convergence properties for the snakes and diminishes very substantially the probability of their getting trapped into an undesirable local minimum, even for initializations that are very far away from the desired answer. The only constraint on the user is the fact that the endpoints he specifies must be close to well-defined edges so that their directions can be estimated reliably. This is a fairly natural constraint that can be explained to a photoanalyst, even if he has no knowledge whatsoever of image processing. In other words, we have proposed a natural initialization procedure that is completely in line with both the practitioner's task and the mathematical problem so that the "expert user" [Kass *et al.*, 1988] of the original snake papers need not be that much of an expert anymore.

We have implemented a standalone version of our ziplock snakes that incorporates an interactive initialization tool in which the contour-finding process starts as soon as the user has clicked on two initial points. Sequentially adding salient points will then extend the initial segment under immediate visual control in a sort of contour-following rubber line process. In case of error, trace-back and editing are supported. We have extended the formalism to deal with ribbon-like structures — for example roads in aerial images — and we have experimented with semiautomatic initialization using key-point information. Ziplock snakes have been ported into the RADIUS Common Development Environment [Mundy *et al.*, 1991] where they can be used to optimize or connect curves and ribbons in a semiautomated fashion while taking advantage of RCDE's extensive editing capabilities.

We will concentrate our future research on generalizing our approach to handle more general constraints and problems of higher dimensionality. In particular, the deformable-surface models described in [Fua and Leclerc, 1994a, Fua and Leclerc, 1994b] are optimized using a traditional snake approach but are amenable to ziplock optimization, which should enhance their performance.

Acknowledgement

We thank Friedrich Heitger and Olof Henricsson of ETH-Zurich for their ideas and active support in supplementing the ziplock snake initialization with key-point information. We also wish to thank Tom Strat from SRI for his various suggestions and for christening the ziplock snakes.

Appendix:
Discretizing the Euler Equation with Boundary Conditions

We discretize the system of two fourth order differential equations of Equation 13, using boundary conditions. We write the equations as

$$-\frac{d}{ds}\left(\alpha(s)\frac{dv(s)}{ds}\right)+\frac{d^2}{ds^2}\left(\beta(s)\frac{d^2v(s)}{ds^2}\right)=-\frac{\partial P}{\partial v}\,. \tag{A.1}$$

where v stands for either x or y. Let the discretized snake $\vec{v}$ consist of $n+1$ vertices $(x_i,y_i)_{0\le i\le n}$. By combining forward, backward and central differences, as proposed in [Leymarie and Levine, 1993], we derive the following discretization:

$$-\frac{d}{ds}\left(\alpha(s)\frac{dv(s)}{ds}\right)+\frac{d^2}{ds^2}\left(\beta(s)\frac{d^2v(s)}{ds^2}\right)\Bigg|_{s=ih}=c_{i-2}v_{i-2}+b_{i-1}v_{i-1}+a_iv_i+b_iv_{i+1}+c_iv_{i+2} \tag{A.2}$$

where

$$\begin{aligned} a_i &= h^{-4}(\beta_{i-1}+4\beta_i+\beta_{i+1}+h^2\alpha_i+h^2\alpha_{i+1})\,, \\ b_i &= h^{-4}(-2\beta_i-2\beta_{i+1}-h^2\alpha_{i+1})\,, && \text{(A.3)} \\ c_i &= h^{-4}\beta_{i+1}\,, && \text{(A.4)} \end{aligned}$$

for $2\le i\le n-2$ and $h=1/n$. This yields $n-3$ equations in the $n+1$ unknown $(v_0,v_1,\ldots,v_n)$. Fixing four snake vertices would reduce number of unknown to $n-3$, making the system's solution unique. A better way to achieve this result, however, is to fix the head and the tail of the snake and to define the derivatives at those points. Let $v'(0)=\gamma$ and $v'(1)=\delta$. The two derivatives have to be approximated by central differences as well. Introducing two new fictive v coordinates v_{-1} and v_{n+1}, we write

$$v'(0)\approx\frac{v_1-v_{-1}}{2h}=\gamma\ ,\qquad v'(1)\approx\frac{v_{n+1}-v_{n-1}}{2h}=\delta\,. \tag{A.5}$$

The discretization of Equation A.2 now becomes valid for $1\le i\le n-1$, yielding a system of $n-1$ equations that can be rewritten in matrix form as

$$-\frac{d}{ds}\left(\alpha(s)\frac{dv(s)}{ds}\right)+\frac{d^2}{ds^2}\left(\beta(s)\frac{d^2v(s)}{ds^2}\right)\Bigg|_{s=ih}\approx K_{n+3}\ (v_{-1},v_0,\ldots,v_n,v_{n+1})^T \tag{A.6}$$

where K_{n+3} denotes the $(n-1) \times (n+3)$ matrix

$$K_{n+3} = \begin{pmatrix} c_{-1} & b_0 & a_1 & b_1 & c_1 & & & & & \\ & c_0 & b_1 & a_2 & b_2 & c_2 & & & & \\ & & c_1 & b_2 & a_3 & b_3 & c_3 & & & \\ & & & & & \ddots & & & & \\ & & & & c_{n-5} & b_{n-4} & a_{n-3} & b_{n-3} & c_{n-3} & \\ & & & & & c_{n-4} & b_{n-3} & a_{n-2} & b_{n-2} & c_{n-2} \\ & & & & & & c_{n-3} & b_{n-2} & a_{n-1} & b_{n-1} & c_{n-1} \end{pmatrix}. \tag{A.7}$$

Using Equation A.5, we substitute the fictive coordinates v_{-1} $(= v_1 - 2\gamma h)$ and v_{n+1} $(= v_{n-1} + 2\delta h)$ in Equation A.6. Furthermore, since the head v_0 and the tail v_n are fixed, we can eliminate the first two and the last two columns of the matrix in Equation A.7. This finally yields a system of $n-1$ linear equations in $n-1$ unknowns

$$\underbrace{\begin{pmatrix} a_1 + c_{-1} & b_1 & c_1 & & & & & \\ b_1 & a_2 & b_2 & c_2 & & & & \\ c_1 & b_2 & a_3 & b_3 & c_3 & & & \\ & & & \ddots & & & & \\ & & c_{n-5} & b_{n-4} & a_{n-3} & b_{n-3} & c_{n-3} & \\ & & & c_{n-4} & b_{n-3} & a_{n-2} & b_{n-2} & \\ & & & & c_{n-3} & b_{n-2} & a_{n-1} + c_{n-1} & \end{pmatrix}}_{K'} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-3} \\ v_{n-2} \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 - b_0 v_0 + 2c_{-1} h\gamma \\ f_2 - c_0 v_0 \\ f_3 \\ \vdots \\ f_{n-3} \\ f_{n-2} - c_{n-2} v_n \\ f_{n-1} - b_{n-1} v_n - 2c_{n-1} h\delta \end{pmatrix}, \tag{A.8}$$

where the $f_i = \frac{\partial P(x_i, y_i)}{\partial v}$ are elements of the image force vector of Equation 12. K' is the symmetric, invertible and pentadiagonal matrix that appears in Equations 16 and 17. It is known as the *stiffness matrix.*

References

[Bascle and Deriche, 1993] B. Bascle and N. Deriche. Stereo Matching, Reconstruction und Refinement of 3D Curves Using Deformable Contours. In *International Conference on Computer Vision*, pages 421–430, Berlin, Germany, 1993.

[Bush, 1992] A. W. Bush. *Perturbation Methods for Engineers and Scientists.* CRC Press, 1992.

[Canny, 1986] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[Cohen *et al.*, 1992] I. Cohen, L. D. Cohen, and N. Ayache. Using Deformable Surfaces to Segment 3-D Images and Infer Differential Structures. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 56(2):242–263, September 1992.

[Courant and Hilbert, 1989] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1. Wiley: New York, 1989.

[Fua and Leclerc, 1990] P. Fua and Y.G. Leclerc. Model Driven Edge Detection. *Machine Vision and Applications*, 3:45–56, 1990.

[Fua and Leclerc, 1994a] P. Fua and Y. G. Leclerc. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, 1994. Accepted for publication, available as Tech Note 535, Artificial Intelligence Center, SRI International.

[Fua and Leclerc, 1994b] P. Fua and Y. G. Leclerc. Using 3–Dimensional Meshes To Combine Image-Based and Geometry-Based Constraints. In *European Conference on Computer Vision*, pages 281–291, Stockholm, Sweden, May 1994.

[Henricsson and Heitger, 1994] O. Henricsson and F. Heitger. The Role of Key-Points in Finding Contours. In *European Conference on Computer Vision*, pages 371–382, Stockholm, Sweden, May 1994.

[Kass *et al.*, 1988] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[Leymarie and Levine, 1993] F. Leymarie and M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–634, June 1993.

[Mundy *et al.*, 1991] J.L. Mundy, R. Welty, L. Quam, T. Strat, W. Bremmer, M. Horwedel, D. Hackett, and A. Hoogs. The RADIUS Common Development Environment. In *Proc. of AIPR, Washington, DC*, October 1991. (also *in Proc. of DARPA Image Understanding Workshop*, San Diego, California, 1992.).

[Rosenthaler *et al.*, 1992] L. Rosenthaler, F. Heitger, O. Kübler, and R. von der Heydt. Detection of general edges and keypoints. In O. Faugeras, editor, *European Conference on Computer Vision*, pages 78–86. Springer-Verlag, 1992.

[Staib and Duncan, 1992] L.H. Staib and J.S. Duncan. Boundary Finding with Parametrically Deformable Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1061–1075, November 1992.

[Terzopoulos and Szeliski, 1992] D. Terzopoulos and R. Szeliski. Tracking with Kalman Snakes. In A. Blake and A. Yuille, editors, *Active Vision*. The MIT Press, 1992.

[Terzopoulos *et al.*, 1987] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking Models and 3D Object Reconstruction. *International Journal of Computer Vision*, 1:211–221, 1987.

[Terzopoulos, 1987] D. Terzopoulos. On matching deformable models to images. *Topical Meeting on Machine Vision Tech. Digest Series*, 12:160–167, 1987.