Initializing 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

In this paper, we propose a snake-based approach that lets a user specify only the distant end points of the curve he wishes to delineate without having to supply an almost complete polygonal approximation. We achieve much better convergence properties than those of traditional snakes by using the image information around these end points to provide boundary conditions and by introducing an optimization schedule that allows the snake to take image information into account first only near its extremities and then, progressively, towards its center.

These snakes could be used to alleviate the often repetitive task practitioners have to 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.

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 [1, 2] and have since given rise to a large body of literature ([3, 4, 5, 6] 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 [7, 8], it is ineffective for delineating complex objects from scratch.

The optimization of the traditional snakes is typically 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, this often results in the snake getting 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 an objective function and a dissipation potential that enforces convergence. This potential tends to prevent the snake from moving far away from its current position, thereby contributing to forcing the snake's initial position to be close from the desired solution in order to achieve convergence and not get stuck in an undesirable local minimum. In the remainder of the paper, we will refer to these snakes as "dynamic snakes" because their position is computed by solving the dynamics equation of a physical system.

Here, we describe a snake approach that allows a user to specify only the end points of the curve he wishes to delineate instead of a complete polygonal approximation. This way we may abolish the need to outline the desired structure very precisely, that is, to perform a painstaking, almost complete, manual segmentation. The optimization progresses from the end points towards the center thereby effectively propagating edge-information along the curve without the need for a dissipation potential. The user-supplied end points and the automatically computed edge gradient in their vicinity serve as anchors. They are first used to compute an initial state that is approximately correct in the anchors' vicinity. While the image term is "turned on" progressively from the snake's extremities towards its middle section, the snake's position is iteratively recomputed. As a result, the snake eventually finds the smooth path, as defined by the regularization term, that best matches the edge connecting the two end points and has the right orientation at these points. We will refer to these snakes as "static" snakes.

In the following section, we describe the traditional or dynamic snakes. Next, we introduce our own breed of static snakes. Finally, we present results on both synthetic and real images that demonstrate the improved performance of the static snakes for initializations that are far away from the desired result.

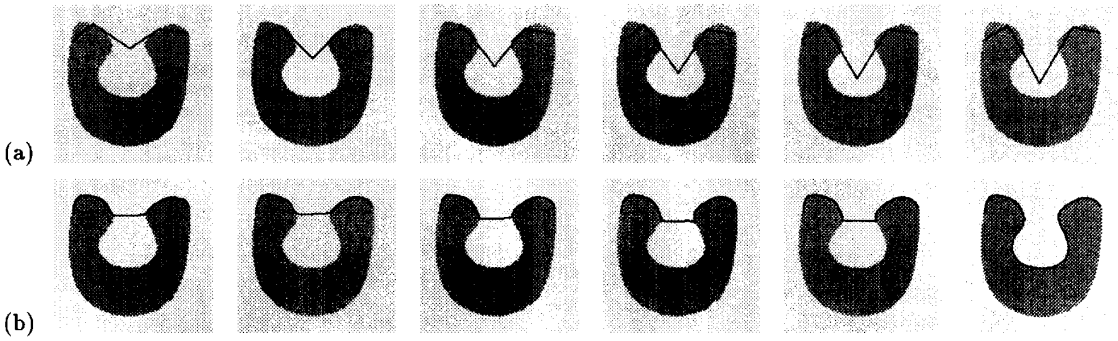

Figure 1: Ill conditioned behavior of the dynamic snakes with respect to initialization . (a) Slightly different initializations: the snake is initialized using a polygon with five vertices. While the third vertex moves closer to the shape, the other vertices are the same for all six situations. (b) The corresponding results: the snake detects the correct contour (6th image pair) when the third vertex is close enough to the object's border. However, it is not intuitively obvious what the threshold is.

2 Dynamic snakes

The original snakes [2] are modeled as time dependent 2–D curves defined parametrically as

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

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 $E_I(\vec{v}) = -\int_0^1 P(\vec{v}(s,t))\, ds$, 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. 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 [4] or replace the gradient magnitude by its logarithm [3]. 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 [9]. The results of all these approaches are very similar. In our implementation, described in section 3, we use the latter where the Euclidean distances are computed using the Danielsson distance transform [10]. Unfortunately, whatever the choice of P, $E_I(\vec{v})$ is typically not a convex functional.

To perform the optimization, following Terzopoulos *et al.*, one must minimize an energy $E(\vec{v})$ that is the sum of $E_I(\vec{v})$ and of a regularization term $E_D(\vec{v})$. Using the thin-plate model, $E_D(\vec{v})$ is taken to be

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

where $\alpha(s)$ and $\beta(s)$ are arbitrary functions that regulate the curve's tension and rigidity. In the implementation described in section 3, α and β are taken to be constant and are supplied by the user. We have shown [3] that they can be chosen in a fairly image-independent way. Several techniques, however, have been proposed to dynamically adjust the values of α and β along the curve (see [11] for example).

Variational calculus 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

$$-\nabla P(\vec{v}(s,t)) = -\frac{\partial}{\partial s}\left(\alpha(s)\frac{\partial \vec{v}(s,t)}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta(s)\frac{\partial^2 \vec{v}(s,t)}{\partial s^2}\right) \quad (3)$$

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

In practice, to minimize $E(\vec{v})$, one must discretize the curve $\vec{v}$ by sampling it at regular intervals. We therefore take $\vec{v}$ to be a polygonal curve defined by a set of vertices $\vec{v}^t = (x_i^t, y_i^t)_{0 \le i \le n}$.

Using finite differences, the snake energy $E(\vec{v})$ becomes $E(\vec{v}) = E_I(\vec{v}) + E_D(\vec{v})$ where

$$E_I(\vec{v}^t) = -\sum_i P(x_i^t, y_i^t)$$

$$E_D(\vec{v}^t) = \frac{1}{2}\sum_i \alpha_i \left[(x_i^t - x_{i-1}^t)^2 + (y_i^t - y_{i-1}^t)^2\right] + \frac{1}{2}\sum_i \beta_i \Big[(2x_i^t - x_{i-1}^t - x_{i+1}^t)^2 + (2y_i^t - y_{i-1}^t - y_{i+1}^t)^2\Big] .$$

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

$$E_D(\vec{v}^t) = \frac{1}{2} X_t^T K X_t + \frac{1}{2} Y_t^T K Y_t , \qquad (4)$$

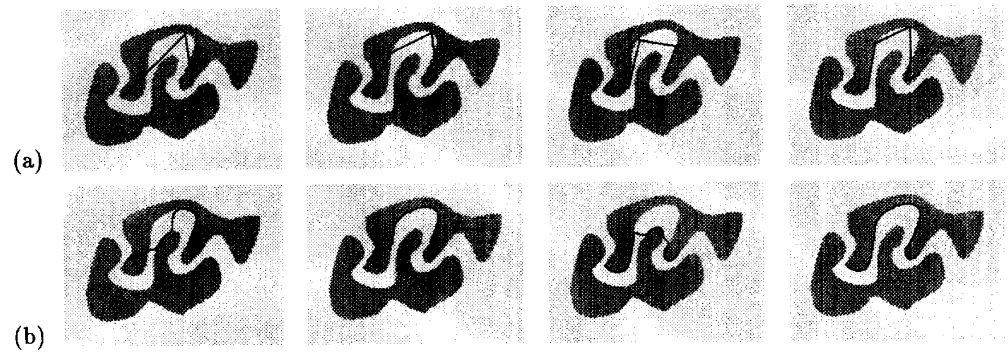

Figure 2: Sensitivity of dynamic snakes to nearby contours. (a) Different initializations: the snake is initialized using a polygon with an increasing number of vertices. (b) The corresponding results: The fact that the two objects are lying close to each other forces the user to outline the desired contour segment precisely. If the snake touches the influence region of a nearby object it will get stuck on the wrong contour.

where K is a $(n+1) \times (n+1)$ penta-diagonal matrix, and $X = (x_0, x_1, \ldots, x_n)$ and $Y = (y_0, y_1, \ldots, y_n)$ are the vectors of the x and y vertex coordinates.
A curve that minimizes the energy E must be such that

$$\frac{\partial E}{\partial \vec{v}} = \frac{\partial E_D}{\partial \vec{v}} + \frac{\partial E_I}{\partial \vec{v}} = 0\,. \tag{5}$$

Since the deformation energy E_D in equation (4) is quadratic and decouples the x and y coordinates of the curve, equation (5) can be rewritten as a system of equations

$$K\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} F_X \\ F_Y \end{pmatrix} \text{ where } F_X = -\frac{\partial E_I}{\partial X},\ F_Y = -\frac{\partial E_I}{\partial Y} \tag{6}$$

which are coupled by the "image forces," F_X and F_Y. Note that F_X and F_Y depend on the snake's position, making the system semi-linear. The matrix K, however, is not invertible and these equations cannot be solved directly. This stems from the fact that the Euler differential equation (3) has a unique solution only when boundary conditions are supplied. 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 in which the minimization of $E(\vec{v})$ is achieved by embedding the curve into a viscous medium and solving the equation of the dynamics. This amounts to adding a Rayleigh dissipation functional to the energy and leads to solving the following differential equation:

$$\frac{\partial E}{\partial \vec{v}} + \gamma \frac{d\vec{v}}{dt} = 0\,,$$

where γ is the viscosity of the medium. As in the case of equation (6), after time discretization, this can be rewritten as a set of two equations in X and Y:

$$\begin{aligned} (K + \gamma I)X_t &= \gamma X_{t-1} + F_X \\ (K + \gamma I)Y_t &= \gamma Y_{t-1} + F_Y \end{aligned} \tag{7}$$

where K, F_X and F_Y are defined in equation (6). The matrix $(K + \gamma I)$ is positive definite for $\gamma > 0$. The dynamic snakes are effective when the initial position of $\vec{v}$ is close from the desired solution. However, as illustrated by figures 1 and 2, 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. In the next section, we introduce a different breed of snakes, the "static snakes" that alleviate these problems by replacing the viscosity term by boundary conditions that produce better solutions of equation (3).

3 Static snakes

To improve upon the snakes' convergence properties, we must use constraints that better reflect the image properties than the Rayleigh dissipation functional of section 2. In our implementation, we assume that the user specifies snake end points in the vicinity of clearly visible edge segments, which implies a well defined edge direction. It becomes natural to use these points and their associated edge directions as anchor points and to propagate the edge information along the curve starting from them.

We use these anchor points to derive an initial position for the snake which will, in general, be close to the desired answer in the vicinity of those points but

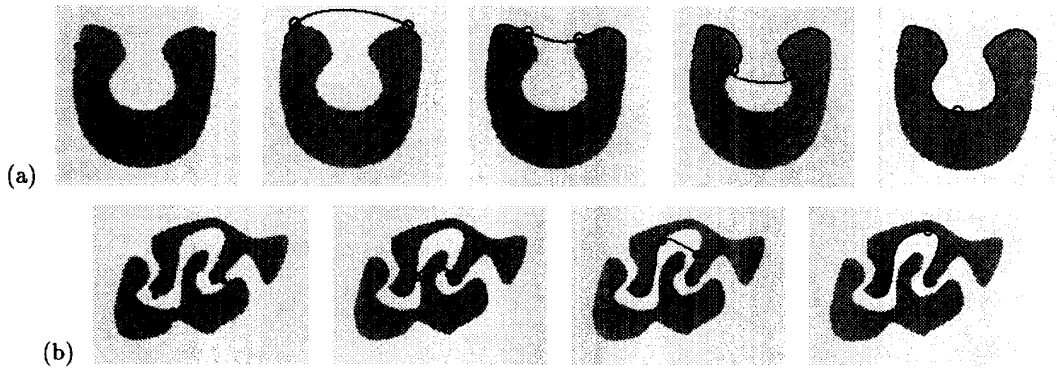

Figure 3: Evolution of a static snake on the synthetic images of figures 1 and 2. The circles denote the farthest vertices away from the end points for which the image forces are turned on. The optimization stops when the two circles meet (section 3.3).

nowhere else. We will therefore "turn on" the image forces (6) 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. Our approach is closely related to perturbation theory [12]: we start with an unperturbed solution of our minimization problem and progressively perturb it by considering more and more of the image forces.

Dynamic programming provides an alternative approach to find an optimal path between selected points. However, there are strict contrains for the applicable cost function and the method is basically restricted to 2-D path finding problems. The lack of these limitations in the snake approach allows the generalization of our method to a broader class of problems.

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 (3) which leads, after discretization, the semi-linear system of the two equations (6). By fixing the curve's end points (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:

$$K'X' = F'_X \quad ; \quad K'Y' = F'_Y$$

where X' is the $n-1$ vector $(x_1, \ldots, x_{n-1})$, Y' the $n-1$ vector $(y_1, \ldots, y_{n-1})$ and K' an $(n-1)\times(n-1)$ penta diagonal matrix that is now invertible. The system is still semi-linear and cannot, in general, be solved in closed form. Instead, one must still use a time discretization and iteratively solve the system

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

3.2 Initialization

In order to successfully optimize our snake, we must start from an initial position that is approximately correct in the neighborhood of the end points. The easiest way to achieve this result is to solve the homogeneous equations that correspond to the system of Euler equations (3). As discussed in section 2, we take α and β to be constant, and the homogeneous system becomes

$$-\alpha\frac{d^2v(s)}{ds^2} + \beta\frac{d^4v(s)}{ds^4} = 0 \qquad (8)$$

where v stands for either x or y and $0 \le s \le 1$.

We assume, that the user has chosen the snake's head and tail close to dominant edge fragments. In order to find the true edge location in the near neighborhood of the selected start and end points we first perform a linear search. The snake tangent at its head and tail is then given by the valley direction in the potential surface corresponding to the selected contour fragments. It is obtained from the orientation map that can be computed at the same time as the Canny edge-map. While the tangent direction at the end points can be computed, its orientation cannot be determined. By default, the boundary conditions are chosen so that the initial snake defines acute angles

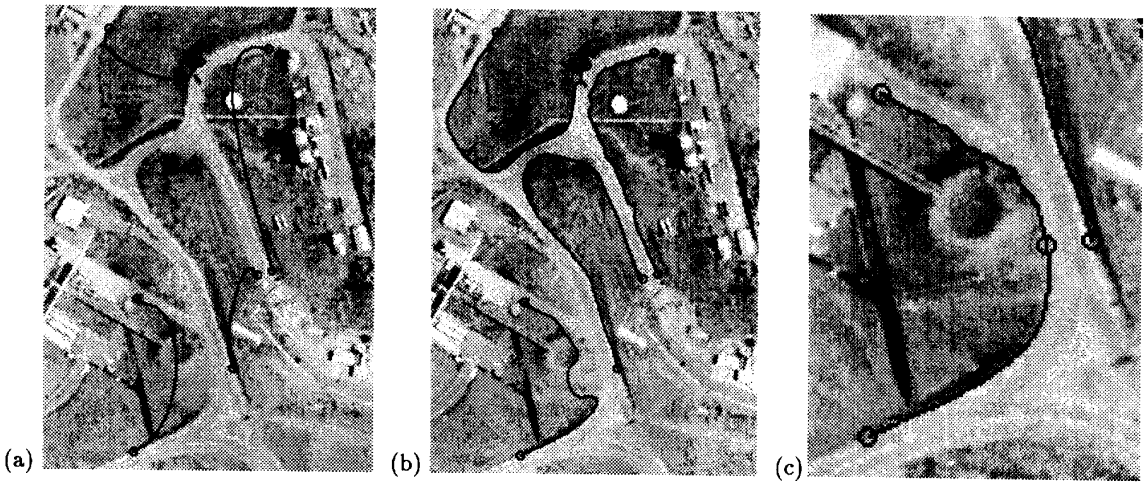

Figure 4: Outlining roads in an aerial image. (a) Static snakes initializations. (b) Final results. All the road edges are correctly outlined except the bottom left one. (c) The erroneous result is corrected by adding a new control point.

with the line joining the two end-points. Since this heuristic may fail, we provide the user with the possibility to flip the orientation at both ends. This could be automated by initializing the snake in the four possible ways (2 possible orientations at each end) and retaining the best final result.

By construction, this solution has the right tangent at the end points 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.

3.3 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 (8). 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 end points nearby dominant edge fragments in the image, this initialization ensures that the snake lies already close to its optimal position at both ends. We define the "force boundaries" as the location of the vertices farthest away from the end points that feel the image forces. These boundaries approach each other during the ongoing optimization process according to the following rule:

- Each boundary is moved at every iteration step by at least one vertex. To speed up the convergence, we use the fact that our potential surface is the Danielsson distance map. We shift both boundaries across the contiguous vertices whose potential does not exceed the one of the currently active ones by more than 1.

This strategy allows the snake to leave valleys and to close small gaps. However, further investigation is required to better control this "gap closing" mechanism. We are also working on modifying these heuristics to deal with a potential surface derived directly from the image gradients.

4 Results

In this section, using both synthetic and real images, we compare the dynamic snakes with our static snakes and show that the formers' initialization must typically be much closer to the desired answer to achieve comparable results. To achieve a fair comparison, we use the same tension and rigidity parameters, α and β defined in equation (2), for both kinds of snakes.

Figure 3 illustrates a static snakes' behavior on two synthetic images and its ability to outline the cavity and distinguish between two nearby objects. Figure 4 shows that our static snakes can be used to delineate roads in an aerial image using very distant end-points. Note, however, that our snakes can still become confused in the presence of junctions. This is the case for the snake drawn in the bottom left corner of figure 4(b), which is being trapped by 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 (see figure 4(c)). Figures 5 and 6 show the snake's performance on an image of an apple and illustrate similar results on a low contrast face image.

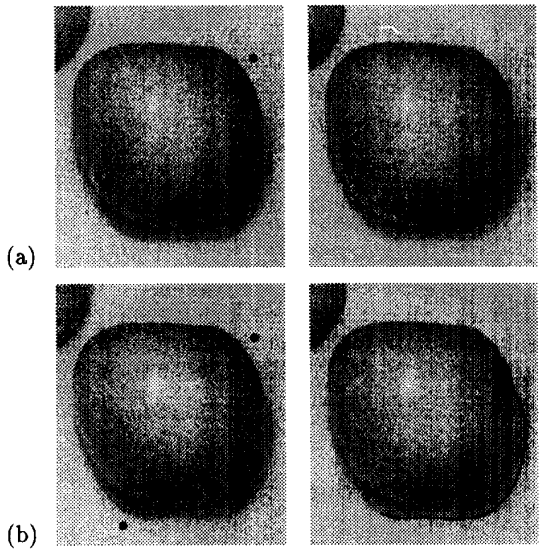

Figure 5: Detection of different image features by static snakes. The apple's outline is detected in (a) whereas (b) shows the detection of the projected shadow.

Figure 6: Outlining facial features. (a) Three pairs of end points on a face image (Courtesy of INRIA). (b) Final results.

5 Conclusion

We have proposed a snake-based approach to semi-automated delineation that allows a user to outline an open contour by only specifying very distant end points and allowing the computer to propagate edge-information from the extremities towards the center. This yields excellent convergence properties for the snakes and diminishes very substantially the probability getting trapped into an undesirable local minimum. By sequentially defining end points of adjoining open snake fragments we are able to segment more complex shapes.

Intelligent initialization of snakes is required to make them into an operational tool for image segmentation where existing implementations leave almost all the work to an "expert user" [2]. Our method has been designed for practitioners of image evaluation without a professional background in image understanding.

We concentrate our future research on generalizing our approach to handle more general constrains and problems of higher dimensionality.

References

[1] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking Models for 3D Object Reconstruction. *IJCV*, 1(3):211–221, October 1987.

[2] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*,1(4):321-331,1988.

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

[4] F. Leymarie and M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE PAMI*, 15(6):617–634, 1993.

[5] I. Cohen, L. D. Cohen, and N. Ayache. Using Deformable Surfaces to Segment 3-D Images and Infer Differential Strucutres. *CVGIP: IU*, 56(2):242–263, September 1992.

[6] L.H. Staib and J.S. Duncan. Boundary Finding with Parametrically Deformable Models. *IEEE PAMI*, 14(11):1061–1075, November 1992.

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

[8] B. Bascle and R. Deriche. Stereo Matching, Reconstruction and Refinement of 3D Curves Using Deformable Contours. In *ICCV*, pages 421–430, Berlin, Germany, 1993.

[9] J. Canny. A computational approach to edge detection. *IEEE PAMI*, 8(6):679–698, 1986.

[10] P.E. Danielsson. Euclidean distance mapping. *CVGIP*, 14:227–248, 1980.

[11] R. Samadani. Changes in connectivity in active contour models. In *Proceedings of the IEEE Workshop on Visual Motion, Irvine, California*, pages 337–343, March 1989.

[12] A. W. Bush. *Perturbation methods for engineers and scientists*. CRC Press, 1992.