*Published in the Proc. of the Fifth Int. Conf. on Computer Vision (ICCV '95), Cambridge, MA, USA, June, 1995, 840–845.*840

Topologically Adaptable Snakes

Tim McInerney and Demetri Terzopoulos
Department of Computer Science, University of Toronto
Toronto, ON, Canada M5S 1A4

Abstract

This paper presents a topologically adaptable snakes model for image segmentation and object representation. The model is embedded in the framework of domain subdivision using simplicial decomposition. This framework extends the geometric and topological adaptability of snakes while retaining all of the features of traditional snakes, such as user interaction, and overcoming many of the limitations of traditional snakes. By superposing a simplicial grid over the image domain and using this grid to iteratively reparameterize the deforming snakes model, the model is able to flow into complex shapes, even shapes with significant protrusions or branches, and to dynamically change topology as necessitated by the data. Snakes can be created and can split into multiple parts or seamlessly merge into other snakes. The model can also be easily converted to and from the traditional parametric snakes model representation. We apply a 2D model to various synthetic and real images in order to segment objects with complicated shapes and topologies.

1 Introduction

Image segmentation remains a fundamental goal in computer vision research. In recent years, segmentation techniques which combine a local edge extraction operation with the use of active contour models, or snakes [5], to perform a global region extraction have achieved considerable success for certain applications [3, 1, 4, 6, 9]. These models simulate elastic material which can dynamically conform to object shapes in response to internal forces, external image forces, and user specified constraints. The result is an elegant method of linking sparse or noisy local edge information into a coherent object description.

Traditional snakes models are not without limitations. Most algorithms based on active contour models can only handle geometrically and topologically simple objects. They are inadequate for objects with deep cavities or multipart objects. Snakes are sensitive to their initial conditions and therefore were designed as interactive models, allowing the user to initialize them near objects of interest. The internal energy constraints of snakes models can limit their geometric flexibility and prevent them from representing long tube-like shapes or shapes with significant protrusions or bifurcations. Furthermore, the topology of the structure of interest must be known in advance since traditional snakes models are parametric and are incapable of topological transformations without additional machinery.

Several researchers have attempted to address some of these limitations. Cohen [3] used an internal "inflation" force to expand the snake past spurious edges towards the real edges of the structure (as was done for deformable surfaces in [11]), making the model less sensitive to initial conditions. Samadani [9] used a heuristic technique based on deformation energies to split and merge active contours. More recently, Malladi *et al.* [8] and Caselles *et al.* [2] independently developed a topology independent active contour scheme based on the modeling of propagating fronts with curvature dependent speeds, where the propagating front is viewed as an evolving level set of some implicitly defined function.

Most active contour models are parametric models whose parameterization is defined initially and does not change automatically throughout the deformation process. If the topology of an object is fixed and known a priori, such models are the most appropriate since they will provide greater constraint. Implicit models on the other hand, such as the formulation used in [8], provide topological and geometric flexibility through their level sets. They are best suited to the recovery of objects with complex shapes and unknown topologies. Unfortunately, implicit models are not as convenient as parametric models in terms of mathematical formulation, for shape analysis and visualization, and for user interaction.

In this paper, we develop a parametric snakes model that has the power of an implicit formulation by using a superposed simplicial grid to quickly and efficiently reparameterize the model during the deformation process. That is, we embed the traditional snakes model within the framework of simplicial domain decomposition — a theoretically sound decomposition method based on classical results from algebraic topology. This framework enables our model to maintain the traditional properties associated with snakes, such as user interaction, while overcoming many of the limitations described above. Using the simplicial grid to iteratively reparameterize the model allows it to flow into complex shapes and change its topology when necessary. Multiple instances of the model can be dynamically created or destroyed, or can seamlessly split or merge. Conversion to and from the traditional snakes model formulation is simply a matter of discarding or imposing the grid at any time. Thus, the grid provides a simple and effective means to extend the geometric and topological adaptability of snakes.

We apply our topologically adaptable snakes model to segment objects with complex shapes and topologies that cannot easily be segmented with traditional snakes. In this paper, we consider the 2D case only, although the model is readily extensible to deformable surfaces in higher dimensions.

2 Model Implementation

We define our snakes model as a closed elastic 2D contour consisting of a set of nodes interconnected by adjustable springs [1]. The elastic contour model is a discrete approximation to the traditional snakes model and retains all of the snake properties. That is, an "inflation" force pushes the model towards image edges until it is opposed by external image forces, the internal spring forces act as a smoothness constraint, users can interact with the model using spring forces and other constraints, and the deformation of the model is governed by discrete Lagrangian equations of motion.

Unlike traditional snakes, the set of nodes and interconnecting springs of our model does not remain constant during its motion. That is, we decompose the image domain into a grid of discrete cells. As the model moves under the influence of external and internal forces, we reparameterize the model with a new set of nodes and springs by efficiently computing the intersection points of the model with the superposed grid. By reparameterizing the model at each iteration of the evolutionary process, we create a simple, elegant and automatic model subdivision technique as well as an unambiguous framework for topological transformations. This allows the model to be relatively independent of its initial placement and "flow" into complex shapes with complex topologies in a stable manner. Furthermore, conversion to and from a traditional parametric snakes model representation is simply a matter of discarding or superposing the grid at any time during the evolutionary process.

2.1 Discrete snake model

A 2D deformable contour or snake can be thought of as an energy minimizing spline in the x-y image plane. We define a discrete snake as a set of N nodes indexed by $i = 1, \ldots, N$. We associate with these nodes time varying positions $\mathbf{x}_i(t) = \left[x_i(t), y_i(t)\right]$ and a mass m_i along with compression forces which make the snake act like a series of unilateral springs that resist deformation from a rest length, rigidity forces which make the snake act like a thin wire that resists bending, and external forces that act in the image plane. We connect the nodes in series using nonlinear springs to form a discrete dynamic system whose behavior is governed by the set of ordinary differential equations of motion

$$m_i\ddot{\mathbf{x}}_i + \gamma_i\dot{\mathbf{x}}_i + \boldsymbol{\alpha}_i + \boldsymbol{\beta}_i = \mathbf{f}_i. \tag{1}$$

where $\ddot{\mathbf{x}}_i$ is the acceleration of node i, $\dot{\mathbf{x}}_i$ is its velocity, m_i is the mass, γ_i is a damping coefficient that controls the rate of dissipation of the kinetic energy of the nodes and $\mathbf{f}_i$ is an external force that attracts the model toward salient image edges [5]. The force

$$\boldsymbol{\alpha}_i = a_i e_i \hat{\mathbf{r}}_i - a_{i-1} e_{i-1} \hat{\mathbf{r}}_{i-1} \tag{2}$$

makes the snake resist expansion or compression, where $\mathbf{r}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$ and the caret denotes a unit vector, a_i is the spring stiffness and $e_i = ||\mathbf{r}_i|| - L_i$, where L_i is the spring "rest" length. Since a new set of model nodes and springs is computed during every iteration, we update these rest lengths by setting them equal to the new spring lengths.

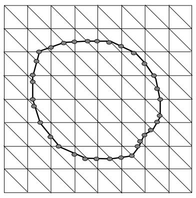


Figure 1: Simplicial approximation of a contour model using a Freudenthal triangulation. The model nodes (intersection points) are marked.

The "rigidity" forces

$$\begin{aligned}\boldsymbol{\beta}_i =& b_{i+1}(\mathbf{x}_{i+2} - 2\mathbf{x}_{i+1} + \mathbf{x}_i) - \\ & 2b_i(\mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i-1}) + \\ & b_{i-1}(\mathbf{x}_i - 2\mathbf{x}_{i-1} + \mathbf{x}_{i-2}),\end{aligned} \tag{3}$$

make the snake resist bending. When computing this force, we "normalize" the spring lengths to account for the uneven node spacing. Finally, an "inflation" force, $\mathbf{h}_i = k\mathbf{n}_i$, is used to push the model towards image edges until it is opposed by external image forces, where k is the force scale factor and $\mathbf{n}_i$ is the unit normal to the contour at node i. The use of an inflation force essentially eliminates the need for an inertial force term. For this reason, we have simplified the equations of motion, while still preserving useful dynamics, by setting the mass density m_i in equation (1) to zero to obtain a model which has no inertia and which comes to rest as soon as the applied forces balance the internal forces. We integrate this first-order dynamic system forward through time using an explicit Euler method.

2.2 Simplicial decomposition

The grid of discrete cells used to approximate the snakes model is an example of space partitioning by simplicial decomposition. There are two main types of domain decomposition methods: non-simplicial and simplicial. Most nonsimplicial methods employ a regular tessellation of space. The marching cubes algorithm [7] is an example of this type. These methods are fast and easy to implement but they cannot be used to represent surfaces or contours unambiguously without the use of a disambiguation scheme.

Simplicial methods, on the other hand, are theoretically sound because they rely on classical results from algebraic topology. In a simplicial decomposition, space is partitioned into cells defined by open simplices, where an n-simplex is the simplest geometrical object of dimension n. A simplicial cell decomposition is also called a triangulation. The simplest triangulation of Euclidean space $\mathcal{R}^n$ is the Coxeter-Freudenthal triangulation (Fig. 1). It is constructed by subdividing space using a uniform cubic grid and the triangulation is obtained by subdividing each cube into $n!$ simplices.

Simplicial decompositions provide an unambiguous framework for the creation of local polygonal approximations of a contour or surface model. In an n-simplex, the negative vertices can always be separated from the positive vertices by a single plane; thus an unambiguous polygonalization of the simplex always exists and as long as neighboring cubes are decomposed so that they share common

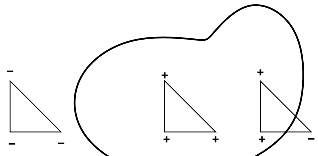


Figure 2: Cell classification.

edges (or faces in 3D) at their boundaries, a consistent polygonization will result. The set of simplices (or triangles in 2D) of the grid that intersect the surface or contour (the boundary triangles) form a two dimensional combinatorial manifold that has as its dual a one dimensional manifold that approximates the contour. The one dimensional manifold is constructed from the intersection of the true contour with the edges of each boundary triangle, resulting in one line segment that approximates the contour inside this triangle (Fig. 1). The contour intersects each triangle in two distinct points, each located on a different edge. The set of all these line segments constitute the combinatorial manifold that approximates the true contour.

The cells of the triangulation can be classified in relation to the partitioning of space by a closed contour model by testing the "sign" of the cell vertices during each time step. If the signs are the same for all vertices, the cell must be totally inside or outside the contour. If the signs are different, the cell must intersect the contour (Fig. 2).

The simplicial decomposition of the image domain also provides a framework for efficient boundary traversal or contour tracing. This property is useful when models intersect and topological changes must take place. Each node stores the edge and cell number it intersects and in a complementary fashion, each boundary cell keeps track of the two nodes which form the line segment cutting the cell. Any node of the model can be picked at random to determine its associated edge and cell number. The model can then be traced by following the neighboring cells indicated by the edge number of the connected nodes.

2.3 Topological transformations

When a snake collides with itself or with another snake, or when a snake breaks into two or more parts, a topological transformation must take place. In order to effect consistent topological changes, consistent decisions must be made about disconnecting and reconnecting snake nodes. The simplicial grid provides us with an unambiguous framework from which to make these decisions. Each boundary triangle can contain only one line segment to approximate a closed snake in that triangle. This line segment must intersect the triangle on two distinct edges. Furthermore, each vertex of a boundary triangle can be unambiguously classified as inside or outside the snake. When a snake collides with itself, or when two or more snakes collide, there are some boundary triangles that will contain two or more line segments. We then choose two line segment end-points on different edges of these boundary triangles and connect them to form a new line segment. The two end-points are chosen such that they are the closest end-points to the outside vertices of the triangle and such that the line segment joining them separates the inside and outside vertices (Fig. 3). Any unused node points are discarded.

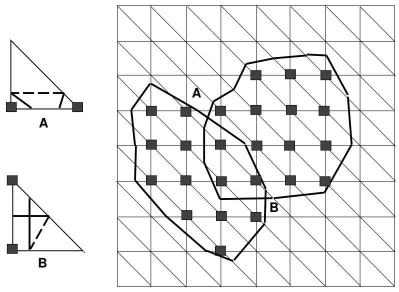


Figure 3: Intersection of two snakes with "inside" grid cell vertices marked. Snake nodes in triangles A and B are reconnected as shown.

With this simple strategy, topological transformations are handled automatically and consistently.

To determine inside vertices of a boundary triangle, we use a simple, efficient ray casting technique. That is, we count the number of intersections that a ray cast from a triangle vertex along its grid row makes with the enclosing snake. An odd number of intersections indicates that the vertex is inside the snake. Counting the number of intersections is simply a matter of a lookup into an active edge-list table constructed during each time step when a snake is projected onto the grid.

Once the topological transformations have taken place, we can run through the list of nodes generated by the procedure above and perform contour tracings via the grid cells, marking off all nodes visited during the tracings. In this fashion we find all new snakes generated by the topological transformation phase and assign each a unique identifier. The result is that at any time during the evolutionary process, we can track, control, and interact with each snake created.

3 Experimental Results

In this section we discuss implementation issues and present a number of experiments, using various synthetic and real image datasets, to demonstrate the model capabilities. The most common method to initiate the segmentation process using topologically adaptable snakes is to have the user draw an initial contour (or contours, if desired) within the objects of interest. Automatic initialization procedures can also be used. These contours are then closed and converted to a parametric model representation using the superposed grid. The snakes are then updated during each time step by: calculating the forces on each snake node and updating the node positions using the simplified equation (1), reparameterizing each snake (computing a new set of nodes and springs) by finding the intersection points of the snake with the grid, computing the new spring rest lengths L_i, performing topological transformations within grid cells as necessary, and traversing each snake via the grid cells, identifying all new snakes.

The reparameterization process can be performed every iteration or every nth iteration to improve computational efficiency. Furthermore, the grid can be turned off (and on) at any time, in which case the topologically adaptable snake reverts to the traditional snake formulation. In addition, different grid resolutions can be used at any time. For example, a coarse resolution grid can be used initially

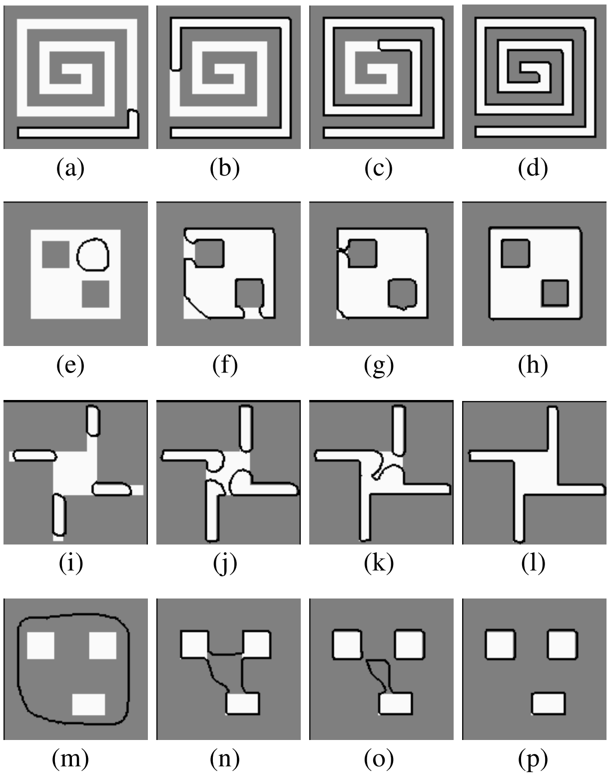


Figure 4: Segmentation of objects with complex geometries and topologies.

and as the algorithm progresses, the grid resolution can be increased. Finally, in our implementation, we compute the edges and vertices of grid cells as they are needed during the evolution of the snake. Consequently, the only memory requirement for the grid is a pointer for each cell to keep track of the snake segments cutting the cell. This "on-the-fly" scheme allows us to use pixel or even subpixel resolution grids, if necessary, without incurring excessive memory or computational costs.

3.1 Experiments with synthetic data

In the first experiment we demonstrate the "flowing" property of the snake by segmenting a spiral shaped object (Figs. 4a–d). We superposed onto a 128×128 pixel image a 40×40 square cell grid, where each cell is divided into two triangles. The parameter values for all of the experiments with synthetic data sets are: $\Delta t = 0.002, a_i = 10.0, b_i = 5.0, k = 20.0, \kappa = 20.1$ (note: κ is the external image force scale factor):

In the second set of experiments we demonstrate the topological transformation capabilities of the model. In Figures 4e–h, a snake flows around two "holes" in the object, collides with itself and splits into three closed snakes. In Figures 4i–l, several snakes are initialized in the protrusions of the object, flow towards each other, and merge. In Figures 4m–p, the snake shrinks, wraps and finally splits to segment each object.

3.2 Segmentation of tool images

In this set of experiments we use the snake to automatically segment various tools with a wide range of shapes and topologies. In each example, a snake was manually placed inside the tool image and allowed to expand outward towards the tool boundary. Note that the final result is relatively independent of the initial placement of the snake, although parameter settings, such as the image force strength, are still determined empirically. Figures 5a–h show the final results, with the original image on the left in each case. The parameter values for all of the experiments are: $\Delta t = 0.002, a_i = 30.0, b_i = 20.0, k = 60.0, \kappa = 62.0$. In Figure 5b, the intermediate states of the evolving snake are shown.

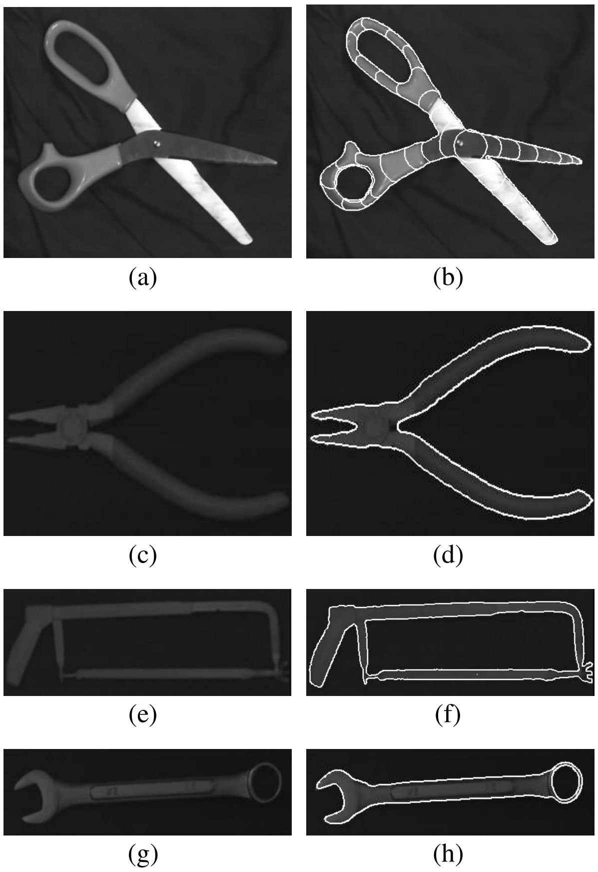
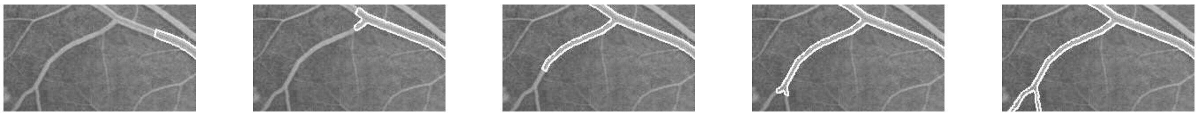


Figure 5: Segmentation of tools.

3.3 Segmentation of retinal vasculature

In the third set of experiments, we applied several snakes to a 1024×1024 retinal image (Fig. 6) in order to segment the vascular "tree", a structure with extended branches and bifurcations. We initialized a snake at the source of each major "branch" and performed the segmentation one branch at a time. To enable a snake to flow along the narrow vessels, a pixel-resolution grid was required. We also manually "freeze" one half of each snake at the branch source to force it to flow into the branch structure. Note that the arteries and veins do not physically intersect as they appear to do in this 2D image projection. Note also that we did not attempt to automatically identify vessel bifurcations or arterio-venous crossings. As a result, we manually freeze a snake once it begins to flow into a crossing branch.



(a) (b) (c) (d) (e)

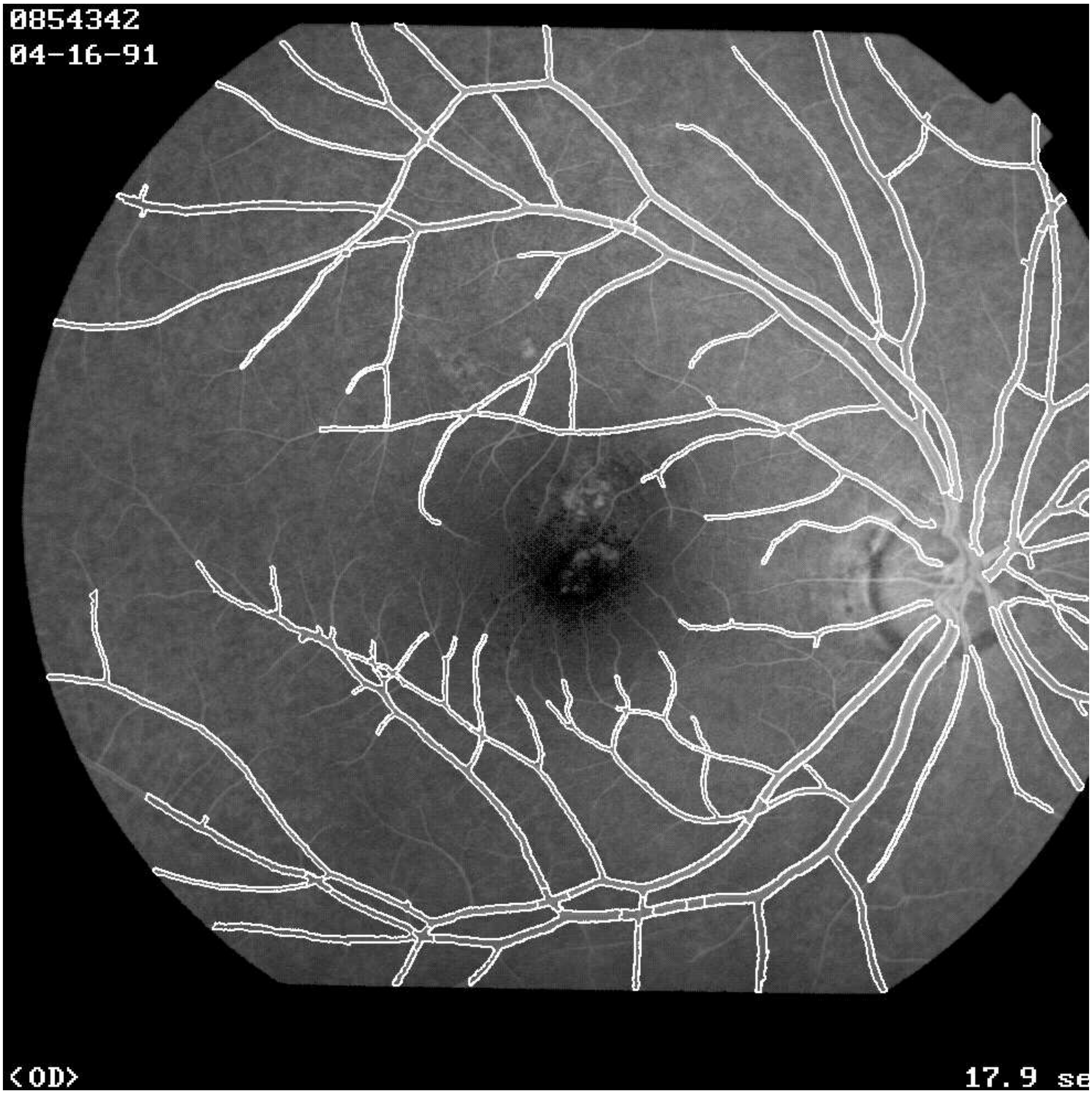


Figure 6: Segmentation of the blood vessels in angiogram of retina. The top row is an image sequence showing a snake flowing and branching along a vessel.

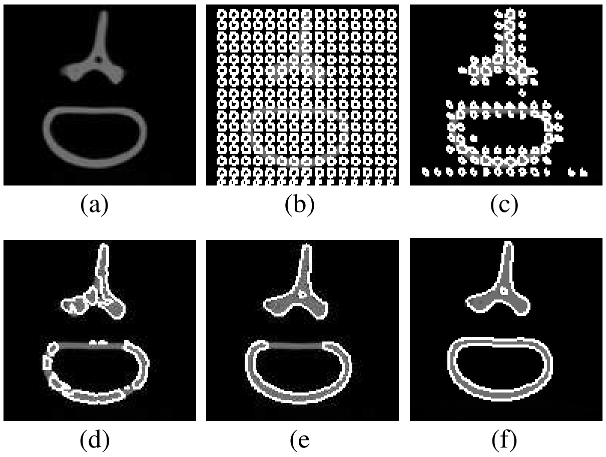


Figure 7: Seed snakes growing, shrinking, merging, splitting and disappearing to automaticlly recover vertebral phantom parts.

3.4 Automatic segmentation using seed snakes

In the final experiment, we demonstrate an interesting automatic segmentation technique. The automatic segmentation of images containing multiple objects of interest, objects embedded inside other objects, or objects containing holes creates an initialization problem that cannot be resolved using a single snake. If the snake is initialized such that it surrounds the objects of interest and then is made to shrink and split around these objects, only the outer boundaries will be recovered. Conversely, to grow a snake from within each object requires prior knowledge of the objects. Malladi *et al.* [8] solved this initialization problem by using a two stage algorithm, capturing the outer boundaries in the first stage and the inner boundaries in the second stage. In our approach, similar to the that taken in [10], we uniformly distribute a set of small circular snake "seeds" over the image domain. These snakes then progressively expand, shrink, merge, and/or split to recover larger and larger regions of each object, including both the inner and outer boundaries (Figs. 7a–f). If there is no object of interest to attract a snake, it will shrink and eventually disappear. This technique results in a single stage process and requires no user interaction.

4 Conclusion

We have developed a 2D topologically adaptable snakes model for image segmentation and object representation. By combining a domain decomposition technique with parametric snakes, we have considerably extended the capabilities of snakes and overcome many of their limitations while keeping all of the traditional properties associated with these models. By iteratively reparameterizing the snake using the superposed grid, we have created a simple and efficient automatic subdivision technique as well as an unambiguous framework for topological transformations, allowing the model to be relatively independent of its initial placement and flow into complex shapes with complex topologies in a stable manner. This domain decomposition framework does not unduly limit the shape recovery scheme and conversion to and from a traditional snakes model representation is simply a matter of discarding or superposing the grid at any time during the evolutionary process. Furthermore, our topologically adaptable snakes model has all of the functionality of the implicit level set techniques described in [8, 2], but unlike these techniques it does not require any mathematical machinery beyond that of traditional snakes and retains their parametric formulation. This allows users to control and interact with the topologically adaptable model as they would with traditional snakes. We have applied the model to various 2D synthetic and real images in order to segment structures of interest that have complicated shapes and topologies. The model is readily extensible to a three dimensional surface representation based on a tetrahedral decomposition of the image domain.

Acknowledgements

TM is grateful for the financial support of an NSERC postgraduate scholarship. DT is a fellow of the Canadian Institute for Advanced Research. This work was made possible by the financial support of the Information Technologies Research Center of Ontario.

References

[1] I. Carlbom, D. Terzopoulos, and K. Harris. Computer-assisted registration, segmentation, and 3D reconstruction from images of neuronal tissue sections. *IEEE Transactions on Medical Imaging*, 13(2):351–362, 1994.

[2] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours in image processing. Technical Report 9210, CEREMADE, 1992.

[3] L.D. Cohen. On active contour models and balloons. In *CVGIP: Image Understanding*, volume 53(2), pages 211–218, March 1991.

[4] P. Fua and Y. Leclerc. Model driven edge detection. *Machine Vision Applications*, 1:45–56, 1990.

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

[6] F. Leymarie and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(1):56–75, 1992.

[7] W.E. Lorenson and H.E. Cline. Marching cubes, a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[8] R. Malladi, J. Sethian, and B. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

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

[10] H. Tek and B. Kimia. Shock–based reaction–diffusion bubbles for image segmentation. Technical Report LEMS–138, Division of Engineering, Brown University, August 1994.

[11] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, 1988.