

Shape and Topology Constraints on Parametric Active Contours

H. Delingette and J. Montagnat¹

Epidaure project, INRIA, 2004 route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex, France

Received January 26, 2001; accepted April 4, 2001

In recent years, the field of active contour-based image segmentation has seen the emergence of two competing approaches. The first and oldest approach represents active contours in an explicit (or parametric) manner corresponding to the *Lagrangian* formulation. The second approach represents active contours in an implicit manner corresponding to the *Eulerian* framework. After comparing these two approaches, we describe several new topological and physical constraints applied to parametric active contours in order to combine the advantages of these two contour representations. More precisely, we introduce three algorithms related to the control of the contour topology, geometry, and deformation. The first algorithm controls both vertex spacing and contour smoothness in an independent and intrinsic manner. The second algorithm controls the contour resolution (number of vertices) while the third algorithm automatically creates or fuses connected components on closed or opened contours. The efficiency of these algorithms is demonstrated on several images including medical images and a comparison with the level-sets method is also provided. © 2001 Academic Press

Key Words: deformable contours; topology change; level-sets; curvature; medical imaging.

1. INTRODUCTION

1.1. Previous Work

Image segmentation based on active contours has achieved considerable success in the past few years [1]. Deformable models are often used for bridging the gap between low-level computer vision (feature extraction) and high-level geometric representation. In their seminal paper [2], Kass *et al.* choose to use a parametric contour representation with a semiimplicit integration scheme for discretizing the law of motion. Several authors have proposed different representations [3], including the use of finite element models [4],

¹ Now working at Creatis, INSA 502, 20 bd Albert Einstein, 69621 Villeurbanne cedex, France.



TABLE 1
Properties of Parametric and Implicit Active Contour Representations

	Parametric contours	Implicit contours
Efficiency	***	*
Ease of implementation	***	**
Topology change	No	Yes
Open contours	Yes	No
Interactivity	Good	Poor

subdivision curves [5], and analytical models [6]. Implicit active contour representation was introduced by Malladi *et al.* [7], following the work of Osher and Sethian [8]. This approach has been developed by several other researchers including Caselles *et al.* [9], who have introduced the concept of “geodesic snakes.”

The opposition between parametric and implicit contour representation corresponds to the opposition between Lagrangian and Eulerian frameworks. In Lagrangian frameworks, the deformation of a body is described with a parameterization corresponding to the body rest position. In Eulerian frameworks, this deformation is described with a parameterization corresponding to the current body position. Lagrangian frames are usually well-suited for solid mechanic computations, whereas Eulerian frames are well-suited for fluid mechanic computations.

In the following, we summarize the respective advantages and disadvantages of these two approaches (Table 1). The comparison of the efficiency and the implementation of these two frameworks is difficult because of the large variety of algorithms existing in the literature. However, in general, implicit representations are regarded as being less efficient than parametric ones. This is because the update of an implicit contour requires the update of at least a narrow band around each contour. Furthermore, on parametric contours vertex sampling may not be constant or uniform whereas on implicit contours the resolution is constrained by the resolution of a regular grid. Speed-up algorithms of level-set methods were proposed either by constraining the contour evolution through the Fast-Marching method [8] or by using an asynchronous update of the narrow-band [10].

If the implementation of basic parametric contours is usually straightforward, the implementation of level-set algorithms requires a little more care due to the handling of the narrow band and the discretization of the hyperbolic term in the differential equation.

The main advantage of the implicit representation is obviously its ability to change automatically the contour topology during the deformation. This property makes it well-suited for reconstructing contours of complex geometry, for instance, tree-like structures. Also, by merging different intersecting contours, it is possible to initialize a deformable contour with a set of growing seeds. For parametric contours, it is in general not possible to achieve any automatic topological changes. However, several algorithms have been proposed to overcome this limitation [11–13]. These approaches are discussed later in this paper.

Sometimes, it is necessary to handle open active contours for proper image segmentation as shown with the work of Berger *et al.* [14]. Open contours are even more widely used when considering the more general problem of regularization and reconstruction from a cloud of points. Because the implicit representation describes a contour as the zero-crossing of a scalar field usually defined on a regular grid, only open contours having their extremities

on the border of the regular grid can be considered with this representation. Therefore, in practice, only a parametric representation can properly describe the deformation of an active open contour.

Finally, we address the issue of user interactivity which, we believe, should be considered as being of high importance. Indeed, because of their sensitivity to the initial position, in most applications, active contours require some level of user interaction in order to locally or globally modify the contour shape. In medical applications, this feature is regarded as essential.

1.2. Contributions

This paper includes three distinct contributions corresponding to the three different modeling levels of parametric active contours:

1. Discretization. We propose two algorithms for controlling the relative spacing between the vertices and the total number of vertices. On one hand, the vertex spacing is controlled through the tangential component of the internal force applied at each vertex. On the other hand, the total number of contour vertices is periodically updated in order to constrain the distance between vertices. This approach based on a periodic update is more efficient than performing any global reparameterization.

2. Shape. We introduce one intrinsic internal force expressions that does not depend on contour parameterization. This force regularizes the contour curvature profile without producing any contour shrinkage.

3. Topology. A new algorithm automatically creates or merges different connected components of a contour based on the detection of edge intersections. Our algorithm can handle opened and closed contours.

Furthermore, we propose a framework where the algorithms for controlling the discretization, shape, and topology of active contours are completely independent of each other. This is in sharp contrast with most previous approaches where at least two of these components were related. For instance, in the level-set framework [8], topology changes are naturally influenced by the regular grid size which also determines the contour resolution. Also, in a typical implementation of active contours [2], since tangential and normal components of the internal force are not considered independently, shape and vertex spacing are not controlled separately which often leads to undesirable coupled effects.

We believe that having algorithmic independence is important for two reasons. First, each modeling component may be optimized separately and therefore this leads to computationally more efficient algorithms. Second, a large variety of active contour behaviors may be obtained by combining different algorithms for each modeling component.

This paper is organized as follows. In a next section, we address the issue of contour discretization by introducing two algorithms: the former algorithm modifies the relative vertex spacing for instance as a function of curvature, whereas the latter algorithm dynamically controls the contour resolution during its deformation. In Section 3, we present new regularizing forces that are independent of the contour parameterization and that create a smooth curvature profile along the contour. In Section 4, we propose a new approach for performing automatic topology changes based on the detection of edge intersections. This algorithm is independent of the contour resolution and can be applied on opened or

closed contours. In Section 5 we show different results based on medical images. Finally, in Section 6, we provide a qualitative and quantitative comparison between our parametric active contours and the level set method.

2. DISCRETIZATION OF ACTIVE CONTOURS

2.1. Parametric Contour Deformation

In the remaining part of this paper, we consider the deformation over time of a two-dimensional parametric contour, $\mathcal{C}(u, t) \in \mathbb{R}^2$, where u designates the contour parameter (which in general differs from the contour arc length) and t designates the time. The parameter u belongs to the range $[0, 1]$ with $\mathcal{C}(0, t) = \mathcal{C}(1, t)$ if the contour is closed. We formulate the contour deformation with a Newtonian law of motion,

$$\frac{\partial^2 \mathcal{C}}{\partial t^2} = -\gamma \frac{\partial \mathcal{C}}{\partial t} + f_{\text{internal}} + f_{\text{external}}, \quad (1)$$

where f_{internal} and f_{external} correspond, respectively, to the internal and external forces. A contour may include several connected components, each component being a closed or opened contour.

Temporal and spatial discretizations of $\mathcal{C}(u, t)$ are based on finite differences. Thus, the set of N^t vertices $\{\mathbf{p}_i^t\}$, $i = 0 \dots N^t - 1$ represents the contour $\mathcal{C}(u, t)$ at time t . The discretization of Eq. (1) using centered and right differences for the acceleration and speed terms leads to

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + (1 - 2\gamma)(\mathbf{p}_i^t - \mathbf{p}_i^{t-1}) + 2\delta t(f_{\text{internal}})_i + 2\delta t(f_{\text{external}})_i,$$

where δt is the time step. We simplify Eq. (1) by hiding the time step δt inside two coefficients, α_i and β_i , weighting the internal and external forces, respectively. Indeed, by removing the time step parameter, the stability of this explicit scheme depends on the choice of the two distinct parameters, α_i and β_i . Therefore, the actual discrete law of motion is

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + (1 - 2\gamma)(\mathbf{p}_i^t - \mathbf{p}_i^{t-1}) + \alpha_i(f_{\text{internal}})_i + \beta_i(f_{\text{external}})_i.$$

In order to simplify the notation, in the remainder of this paper, we will write \mathbf{p}_i instead of \mathbf{p}_i^t for the vertex position at time t .

2.2. Discrete Contour Geometry

We now describe the basic geometry of a discrete contour. At each vertex \mathbf{p}_i , we define a local tangent vector \mathbf{t}_i , a normal vector \mathbf{n}_i , a metric parameter ϵ_i , and a curvature k_i . We propose to define the tangent vector at \mathbf{p}_i , as the direction of the line joining its two neighbors,

$$\mathbf{t}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2r_i},$$

where r_i is the half distance between the two neighbors of \mathbf{p}_i ,

$$r_i = \frac{\|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\|}{2}.$$

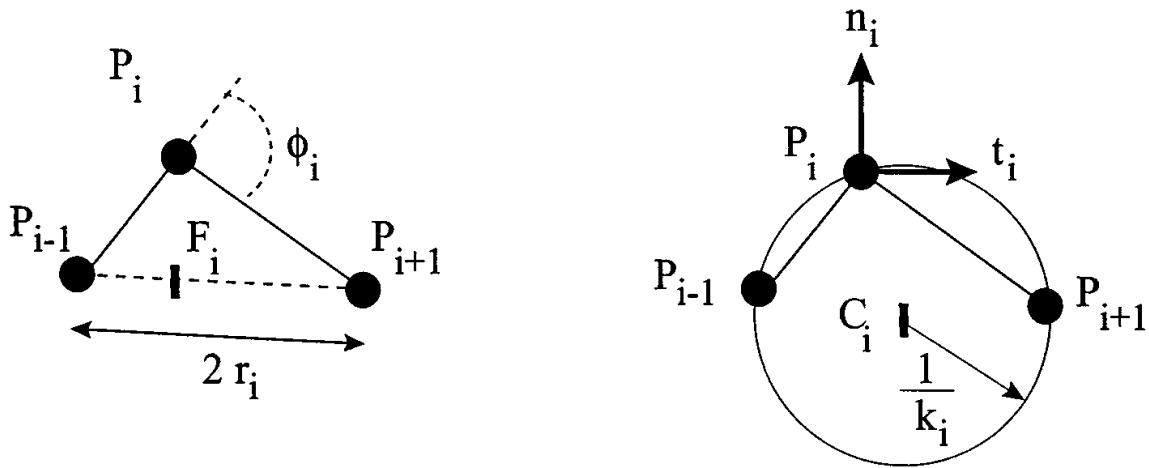


FIG. 1. The geometry of a discrete contour. Definition of the tangent vector \mathbf{t}_i , normal vector \mathbf{n}_i , curvature k_i , angle ϕ_i , and foot \mathbf{F}_i .

The normal vector \mathbf{n}_i is defined as the vector directly orthogonal to \mathbf{t}_i : $\mathbf{n}_i = \mathbf{t}_i^\perp$ with $(x, y)^\perp = (-y, x)$. The curvature k_i is naturally defined as the curvature of the circle circumscribed at triangle $(\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1})$. If we write as ϕ_i , the oriented angle between segments $[\mathbf{p}_{i-1}, \mathbf{p}_i]$ and $[\mathbf{p}_i, \mathbf{p}_{i+1}]$, then the curvature is given by

$$k_i = \frac{\sin(\phi_i)}{r_i}. \quad (2)$$

Finally, the metric parameter ϵ_i measures the relative spacing of \mathbf{p}_i with respect to its two neighboring vertices, \mathbf{p}_{i-1} and \mathbf{p}_{i+1} . If \mathbf{F}_i is the projection of \mathbf{p}_i on the line $[\mathbf{p}_{i-1}, \mathbf{p}_{i+1}]$, then the metric parameter is

$$\epsilon_i = \frac{\|\mathbf{F}_i - \mathbf{p}_{i+1}\|}{2r_i} = 1 - \frac{\|\mathbf{F}_{i+1} - \mathbf{p}_{i-1}\|}{2r_i}.$$

In other words, ϵ_i and $1 - \epsilon_i$ are the barycentric coordinates of \mathbf{F}_i with respect to \mathbf{p}_{i-1} and \mathbf{p}_{i+1} :

$$\mathbf{F}_i = \epsilon_i \mathbf{p}_{i-1} + (1 - \epsilon_i) \mathbf{p}_{i+1}.$$

Other definitions for the tangent and normal vectors could have been chosen. For instance, Gunn *et al.* [15] and Lobregt *et al.* [16] have chosen the tangent of the circumscribed circle at \mathbf{p}_i as the tangent vector. However, our tangent and normal vector definitions have the advantage of providing a simple local shape description,

$$\mathbf{p}_i = \epsilon_i \mathbf{p}_{i-1} + (1 - \epsilon_i) \mathbf{p}_{i+1} + L(r_i, \phi_i, \epsilon_i) \mathbf{n}_i, \quad (3)$$

where $L(r_i, \phi_i, \epsilon_i)$ is the following function:

$$L(r_i, \phi_i, \epsilon_i) = \frac{r_i}{\tan \phi_i} (1 + \mu \sqrt{1 + 4\epsilon_i(1 - \epsilon_i) \tan^2 \phi}),$$

with $\mu = 1$ if $|\phi| < \pi/2$ and $\mu = -1$ if $|\phi| > \pi/2$. Equation (3) simply decomposes vertex position \mathbf{p}_i into a tangential and normal component. The importance of this equation will be revealed in Section 2.3.

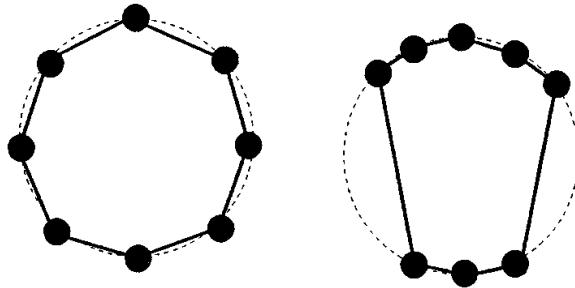


FIG. 2. Two discrete circles with different parameterizations. This illustrates that a contour parameterization modifies its shape.

2.3. Parameterization Control

For a continuous active contour $\mathcal{C}(u, t)$, the contour parameterization is characterized by the metric function,

$$g(u, t) = \left\| \frac{\partial \mathcal{C}}{\partial u} \right\|.$$

If $g(u, t) = 1$ then the parameter of $\mathcal{C}(u, t)$ coincides with the contour arc length. For a discrete contour, the parameterization corresponds to the relative spacing between vertices and is characterized by $g_i = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|$. If all the spacings g_i^t are equal for $i = 0 \dots N - 1$, then the current discrete parameterization is proportional to the arc length.

For a continuous representation, parameterization is clearly independent of the contour shape. For a discrete contour represented by finite differences, shape and parameterization are not completely independent, as illustrated in Fig. 2. The effect of parameterization changes is especially important at parts of high curvature. Therefore, parameterization is an important issue for handling discrete parametric contours.

In this section we propose a simple algorithm to enforce two types of parameterization:

- **uniform parameterization:** the spacing between consecutive vertices is uniform. This property may be of interest for further processing such as performing 3D reconstruction from a set of planar contours [17].

- **curvature-based parameterization:** vertices are concentrated at parts of high curvature. This parameterization tends to optimize the shape description for a given number of vertices.

To modify a contour parameterization, only the tangential component of the internal force should be considered. Indeed, Kimia *et al.* [18] have proven that only the normal component of the internal force applied on a continuous contour $\mathcal{C}(u, t)$ has an influence on the resulting contour shape. This is why only the deformation component along the gradient direction is considered in the level-set method.

Therefore, if \mathbf{t} and \mathbf{n} are the tangent and normal vectors, respectively, at a point $\mathcal{C}(u, t)$, then the contour evolution may be written as

$$\frac{\partial \mathcal{C}}{\partial t} = f_{\text{internal}} = a(u, t)\mathbf{t} + b(u, t)\mathbf{n}.$$

In Kimia *et al.* [18] show that only the normal component of the internal force $b(u, t)$ modifies the contour shape, whereas the metric function $g(u, t) = \|\frac{\partial \mathcal{C}}{\partial u}\|$ evolution is dependent

on $a(u, t)$ and $b(u, t)$:

$$\frac{\partial g}{\partial t} = \frac{\partial a(u, t)}{\partial u} + b(u, t)kg. \quad (4)$$

On the contrary, with parametric contours, it is important to define a tangential component of the internal force $a(u, t)\mathbf{t}$ that constrains the nature of the parameterization. We propose to apply this principle on discrete parametric contours as well by decomposing the internal force f_{internal} into its normal and tangential components:

$$\begin{aligned} (f_{\text{internal}})_i &= (f_{\text{tangent}})_i + (f_{\text{normal}})_i, \\ (f_{\text{tangent}})_i \cdot \mathbf{n}_i &= 0, \\ (f_{\text{normal}})_i \cdot \mathbf{t}_i &= 0. \end{aligned}$$

Several researchers [15, 16] have previously proposed to decompose the internal and external forces along the normal and tangential directions. However, to our knowledge, none of them have considered simultaneously updating the vertex spacing with the tangential component and the contour shape with the normal component. Furthermore, we propose a simple and coherent formulation of these forces based on the two geometric parameters: ϵ_i and ϕ_i .

More precisely, since the tangent direction \mathbf{t}_i at a vertex is the line direction joining its two neighbors, we use a simple expression for the tangential component,

$$(f_{\text{tangent}})_i = (\epsilon_i^* - \epsilon_i)(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) = 2r_i(\epsilon_i^* - \epsilon_i)\mathbf{t}_i,$$

where ϵ_i^* is the reference metric parameter whose value depends on the type of parameterization to enforce.

2.4. Uniform Vertex Spacing

Uniform vertex spacing corresponds to the normal parameterization of a contour where all contour edges have the same length. Such parameterization is well-suited when no specific information is available. To obtain evenly spaced vertices, we simply choose:

$$\epsilon_i^* = \frac{1}{2}.$$

The tangential force has then a simple geometric interpretation: each vertex is moved in the tangent direction toward the middle of its two neighbors:

$$(f_{\text{tangent}})_i = \left(\left(\frac{\mathbf{p}_{i-1} + \mathbf{p}_{i+1}}{2} - \mathbf{p}_i \right) \cdot \mathbf{t}_i \right) \mathbf{t}_i. \quad (5)$$

When the contour reaches its equilibrium, i.e., when $(f_{\text{tangent}})_i = \mathbf{0}$, then \mathbf{p}_i is equidistant from \mathbf{p}_{i-1} and \mathbf{p}_{i+1} . Equation (5) can be seen as the discretization of the force:

$$(f_{\text{tangent}})_i = \left(\frac{\partial^2 \mathcal{C}}{\partial u^2} \cdot \mathbf{t} \right) \mathbf{t} = \frac{\partial g}{\partial u} \mathbf{t}.$$

Because the second derivative vector $\frac{\partial^2 \mathcal{C}}{\partial u^2}$ is the first variation of the weak string $\int_u \|\frac{\partial \mathcal{C}}{\partial u}\|^2 du$ internal energy, Eq. (5) is somewhat related to the classical “snakes” approach proposed by Kass *et al.* [2].

Also, using Eq. (4), we can analyze the evolution of parameterization caused by this tangential force. By setting $a(u, t) = \frac{\partial g}{\partial u}$, we obtain the following differential equation of the metric parameter $g(u, t)$,

$$\frac{\partial g}{\partial t} = \frac{\partial^2 g}{\partial u^2} + b(u, t)kg.$$

When the normal component $b(u, t)$ of the force is small enough, this equation corresponds to the diffusion of the metric parameter along the contour. The steady state is reached when the metric parameter g is uniform: $\frac{\partial g}{\partial u} = 0$ which implies that the parameter u is proportional to the arc length s .

2.5. Curvature-Based Vertex Spacing

The uniform vertex spacing describes with the same spatial resolution all contour parts independently of their curvature. To obtain an optimal description of shape, it is necessary that the vertices are concentrated at parts of high curvature and that the flat parts are only described with few vertices. To obtain such a parameterization, we present a method where edge length is inversely proportional to curvature. If e_i is the edge joining \mathbf{p}_i and \mathbf{p}_{i+1} , then we compute its edge curvature K_i^{i+1} as the mean absolute curvature of its two vertices:

$$K_i^{i+1} = \frac{|k_i| + |k_{i+1}|}{2}.$$

Then at each vertex \mathbf{p}_i , we can compute the local relative variation of absolute curvature $\Delta K_i \in [-1, 1]$ as

$$\Delta K_i = \frac{K_i^{i+1} - K_{i-1}^i}{K_i^{i+1} + K_{i-1}^i}.$$

To enforce a curvature-based vertex spacing, we compute the reference metric parameter ϵ_i^* as

$$\epsilon_i^* = \frac{1}{2} - 0.4 * \Delta K_i. \quad (6)$$

When vertex \mathbf{p}_i is surrounded by two edges having the same curvature then $\Delta K_i = 0$ and therefore ϵ_i^* is set to $\frac{1}{2}$, which implies that \mathbf{p}_i becomes equidistant from its two neighboring vertices. On the contrary, when the absolute curvature of \mathbf{p}_{i+1} is greater than the absolute curvature of \mathbf{p}_{i-1} then ΔK_i becomes close to 1 and therefore ϵ_i^* is close to 0.1 which implies that \mathbf{p}_i moves toward \mathbf{p}_{i+1} .

2.6. Results of Vertex Spacing Constraints

In this section, we demonstrate the influence of these two tangential internal energies on the parameterization of a contour. To illustrate the ability to decouple parameterization and shape properties, we propose to apply an internal force that modifies the vertex spacing on a

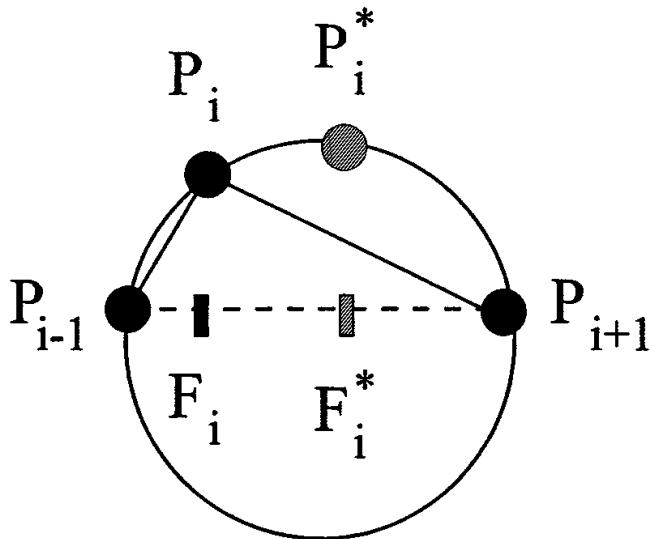


FIG. 3. The internal force associated with the curvature-conservative flow is proportional to $\mathbf{p}_i^* - \mathbf{p}_i$.

contour without changing its shape. For this purpose, we cannot choose $f_{\text{normal}} = \mathbf{0}$ because it would result in shrinking the contour toward a point by applying a mean curvature flow. Instead we define a *curvature conservative* regularizing force that moves \mathbf{p}_i in the normal direction in order to keep the same local curvature:

$$f_{\text{normal}} = (L(r_i, \phi_i, \epsilon_i^*) - L(r_i, \phi_i, \epsilon_i))\mathbf{n}_i. \quad (7)$$

Equation (7) has a simple geometric interpretation if we note that the total internal force $f_{\text{internal}} = f_{\text{tangent}} + f_{\text{normal}}$ is simply equal to $\mathbf{p}_i^* - \mathbf{p}_i$, where \mathbf{p}_i^* is the point having the same curvature as \mathbf{p}_i but with a metric parameter ϵ_i^* . From Fig. 3, we can see that f_{tangent} corresponds to the displacement between \mathbf{F}_i^* and \mathbf{F}_i , whereas f_{normal} corresponds to the difference of elevation between \mathbf{p}_i^* and \mathbf{p}_i .

Given an open or closed contour we iteratively apply differential Eq. (1) with the internal force expression described above. To ensure stability, it is sufficient to choose $\alpha_i = \frac{1}{2}$. Figure 4 shows an example of the vertex spacing constraint enforced on a closed contour consisting of 150 vertices. The initial vertex spacing is uneven as shown by the diagram on the right column that displays the distribution of edge curvature as a function of edge length. When applying the uniform vertex spacing tangential force ($\epsilon_i^* = 0.5$), after 1000 iterations, all contour edge lengths become equal within less than 5% without greatly changing the contour shape, as seen in the second row of Fig. 4. Similarly, with the same number of iterations, the contour evolution using the curvature-based vertex spacing force tends to concentrate vertices at parts of high curvature. The corresponding diagram clearly shows that the edge length is inversely proportional to edge curvature. There is no bijective relationship between edge length and edge curvature because the local variation of length $\|\mathbf{p}_i - \mathbf{p}_{i+1}\|/\|\mathbf{p}_i - \mathbf{p}_{i-1}\|$ is related not only to the metric parameter ϵ_i but also to the local angle ϕ_i . In both cases, the computation time on a Pentium II 400 Mhz was 112 s.

2.7. Contour Resolution Control

In addition to constraining the relative spacing between vertices, it is important to control the total number of vertices. Indeed, the computational complexity of discrete parametric contours is typically linear in the number of vertices. By removing unnecessary vertices,

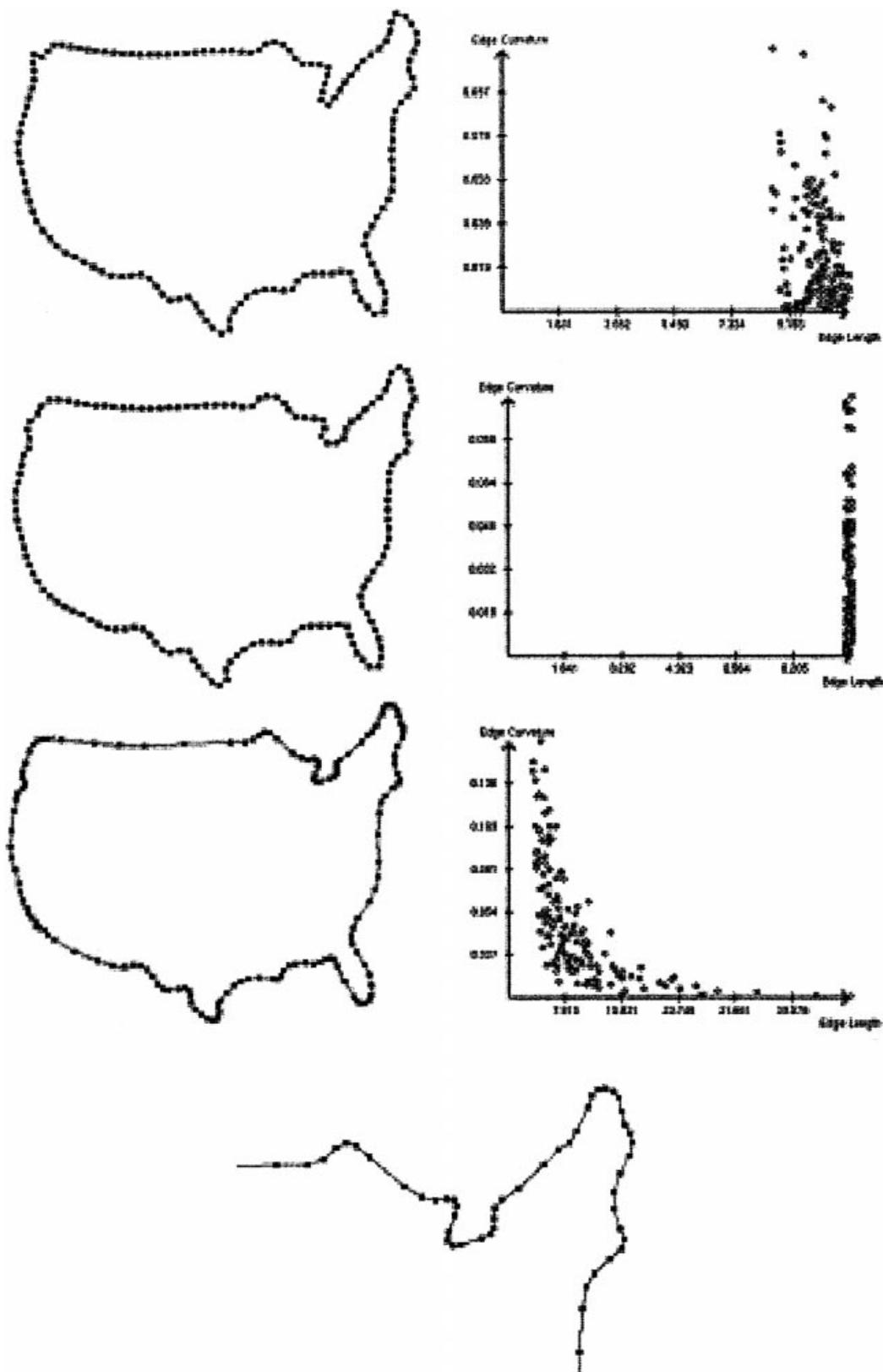


FIG. 4. (First row) The initial contour with the edge curvature-length diagram: (Second row) Contour after applying the uniform vertex spacing tangential force. All edges have nearly the same length: (Third row) Contour after applying the curvature-based vertex spacing tangential force. The edge curvature is inversely proportional to edge length: (Fourth row) Enlarged view of the previous contour.

a significant speed-up may be achieved. Conversely, the shape complexity of a discrete contour is directly linked to the total number of vertices and their relative spacing.

In order to add or remove vertices, we do not use any global contour reparameterization as performed in the level-set method [8] because of its high computational cost. Instead,

we propose to add or remove locally a vertex if the edge length does not belong to a given distance range, similarly to Ivins *et al.* [19] and Lobregt *et al.* [16]. This approach is simpler to implement and it fits nicely with the vertex spacing algorithm presented previously. Indeed, for instance when the uniform vertex spacing is chosen, the combination of the two constraints converges toward a contour where all edges have the same length belonging to a given range. When curvature-based vertex spacing is applied, we must use a rather large range of edge lengths to prevent the creation of vertex accumulations.

Our resolution constraint algorithm proceeds as follows. Given two thresholds, s_{\min} and s_{\max} , corresponding to the minimum and maximum edge length, we scan all existing contour edges. If the current edge length is greater than s_{\max} and $2 \times s_{\min}$ then a vertex is added. Otherwise, if current edge length is less than s_{\min} and if the sum of the current and previous edge length is less than s_{\max} , then the current vertex is removed. In general, this procedure is called every five deformation iterations.

3. SHAPE REGULARIZATION

3.1. Previous Work

In the previous section, we have defined two different tangential force expressions in order control the vertex spacing on active contours. These forces have little influence on the contour shape evolution because they are related only to the contour parameterization. In this section, we deal with the internal force normal component which determines the contour shape regularization. The most widely used internal forces on active contours are:

- **mean curvature motion** used in the level-set method [7], where $f_{\text{internal}} = k\mathbf{n}$, corresponding to the minimization of the contour length $\int_u \|\frac{\partial \mathcal{C}}{\partial u}\| du$;
- **weak string or Laplacian smoothing**, where $f_{\text{internal}} = \frac{\partial^2 \mathcal{C}}{\partial u^2}$ corresponding to the minimization of the elastic energy $\int_u \|\frac{\partial \mathcal{C}}{\partial u}\|^2 du$;
- **thin rod smoothing**, where $f_{\text{internal}} = -\frac{\partial^4 \mathcal{C}}{\partial u^4}$ corresponding to the minimization of the bending energy $\int_u \|\frac{\partial^2 \mathcal{C}}{\partial u^2}\|^2 du$,
- **balloon force**, where $f_{\text{internal}} = p\mathbf{n}$ corresponding to the minimization of the area enclosed by the contour,
- **spring force**, where $f_{\text{internal}} = \sum_j (l_j - l_j^0)(\mathbf{p}_j - \mathbf{p}_i)/\|\mathbf{p}_j - \mathbf{p}_i\|$.

First, we note that Laplacian smoothing and mean curvature motion have an identical regularizing effect since their normal components are equal:

$$\frac{\partial^2 \mathcal{C}}{\partial u^2} = \frac{\partial g}{\partial u} \mathbf{t} + g^2 k \mathbf{n}.$$

In fact, Laplacian smoothing can be seen as a combination of a uniform vertex spacing constraint on the tangent direction with the mean curvature motion in the normal direction. Figure 5 compares the equilibrium position of an open active contour fixed at its two extremities and submitted to mean curvature motion and Laplacian smoothing.

Laplacian smoothing and mean curvature motion have the drawback of significantly shrinking the contour. This shrinking effect introduces a bias in the contour deformation since image structures located inside the contour are more likely to be segmented than structures located outside the contour. Furthermore, the amount of shrinking often prevents active contours from entering inside fine structures.

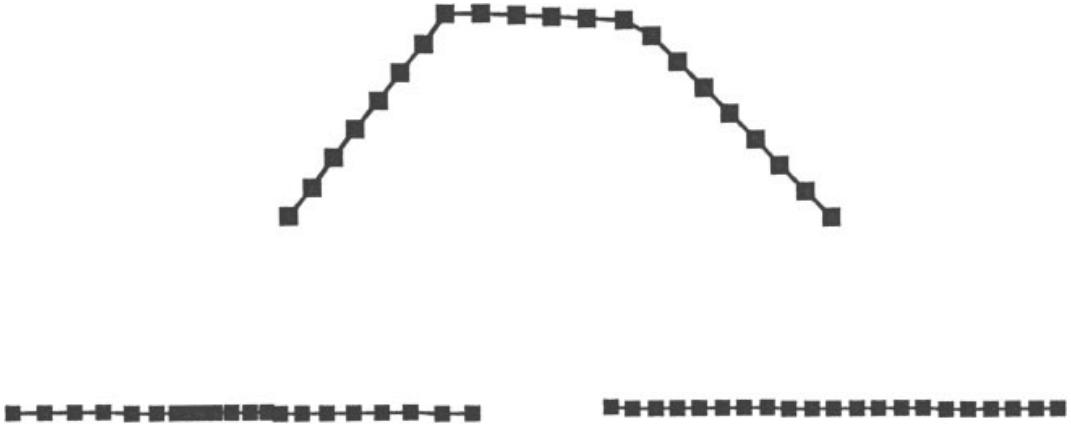


FIG. 5. (Top) Initial contour fixed at its two end vertices. (Bottom left) Equilibrium position after applying the *mean curvature motion*: vertices are unevenly spaced. (Bottom right) Equilibrium position after applying the *Laplacian smoothing*: vertices are evenly spaced.

To prevent the shrinking effect, several approaches have been proposed. Cohen [20] propose to move each vertex a constant distance along the normal direction to compensate for oversmoothing. This balloon energy leads to reaction–diffusion differential equations in the level-set method. Similarly, Taubin [21] proposes to apply a linear filter to curves and surfaces in order to reduce the shrinking effect of Gaussian smoothing. However, these two methods remove the shrinking effect for only a given curvature scale. For instance, when smoothing a circle, this circle would stay invariant only for one given circle radius which is related to a set of filtering parameters. Therefore, in these methods, the choice of these parameters are important but difficult to estimate prior to the segmentation process.

A regularizing force with higher degrees of smoothness than the Laplacian smoothing such as the *thin rod smoothing* causes significantly less shrinking since it is based on fourth derivatives along the contour. However, the normal component of this force $-(\frac{\partial^4 \mathcal{C}}{\partial u^4} \cdot \mathbf{n})\mathbf{n}$ is dependent on the nature of the parameterization, which is a serious limitation. Also, minimizing the curvature integral $\int_u k^2(s) ds$ is not appropriate because it also creates a shrinking effect [22].

3.2. Curvature Diffusion Regularization

We propose to use the following expression as the governing regularizing force:

$$f_{\text{internal}} = \left(\frac{1}{2s_0} \int_{-s_0}^{s_0} k ds - k \right) \mathbf{n} \quad (8)$$

This force tends to smooth the contour curvature profile by replacing the current curvature $k(s)$ with the average curvature $\frac{1}{2s_0} \int_{-s_0}^{s_0} k ds$ in the neighborhood of $\mathbf{p}(s)$. In this equation, s_0 represents half the neighborhood length around $\mathbf{p}(s)$ which is used to compute the average curvature. In fact, s_0 corresponds to a scale factor for the regularization of the curvature profile and it is important to note that $s_0(s)$ may not be the same along the contour.

The contours for which $f_{\text{internal}} = \mathbf{0}$ verifies $k(s) = \frac{1}{2s_0} \int_{-s_0}^{s_0} k ds$ have a linear curvature profile. Thus, all closed contours converge toward circles, independently of their radii. For open contours, clothoid curves having a linear curvature profile $k = as + b$ are the optimal curves associated with this force.

Equation (8) is related to two different regularizing forces that have been previously proposed. First, for closed contours, when s_0 is equal to the contour length \mathcal{L} , then the

proposed internal force corresponds to the *volume preserving mean curvature flow*,

$$f_{\text{internal}} = (\bar{k} - k)\mathbf{n}, \quad (9)$$

where \bar{k} is the mean contour curvature. Contours evolving according to this flow have been studied by Gage [23] and Huisken [24]. In particular, it has the interesting property of preserving the area enclosed by the contour.

When the scale parameter s_0 converges toward zero, $\int_{-s_0}^{s_0} k ds - k$ converges toward $\frac{d^2 k}{ds^2}$. Therefore, the internal force of Eq. (8) is also related to the surface diffusion flow:

$$f_{\text{internal}} = \frac{d^2 k}{ds^2} \mathbf{n}.$$

The properties of contours following this flow have also been largely studied by several researchers including Escher [25].

We propose to generalize these two regularizing force expressions by including a spatial parameter s_0 in Eq. (8). This additional parameter naturally corresponds to the notion of deformation scale. The choice for s_0 depends on the maximum level of detail size required on the reconstructed curve but also on the noise level of external data sets. On the other hand, parameters α_i and β_i are related to the relative confidence given to these datasets. Of course these two factors are often correlated since a low confidence value is often given to noisy data. However, the parameter s_0 enables us to properly scale the internal force deformation. For instance, if a first contour has the same shape than a second contour but with half the number of vertices, then they would have the same evolution if they have the same scale parameter s_0 . This would not have been the case if we had used an internal force like $f_{\text{internal}} = \frac{d^2 k}{ds^2} \mathbf{n}$, because its discrete value strongly depends on the contour discretization. Finally, a major advantage of Eq. (8) over the surface diffusion flow of Eq. (3.2) is that it does not involve any fourth-derivatives of the position. It only requires the estimate of the contour curvature for which Eq. (2) provides a stable computational scheme.

For the discretization of Eq. (8), we do not use straightforward finite differences, since this would lead to complex and potentially unstable schemes. Instead, we propose a geometry-based implementation that is similar to Eq. (7),

$$f_{\text{normal}} = (L(r_i, \phi_i^\star, \epsilon_i^\star) - L(r_i, \phi_i, \epsilon_i))\mathbf{n}_i, \quad (10)$$

where ϕ_i^\star is the angle at a point \mathbf{p}_i^\star for which $\frac{d^2 k}{ds^2} = 0$. The geometric interpretation of Eq. (10) is also straightforward, since the internal force $f_{\text{internal}} = f_{\text{tangent}} + f_{\text{normal}}$ corresponds to the displacement $\mathbf{p}_i^\star - \mathbf{p}_i$ (Fig. 6). The angle ϕ_i^\star is simply computed from Eq. (2):

$$\phi_i^\star = \arcsin(k_i^\star * r_i),$$

where k_i^\star is the average curvature around \mathbf{p}_i^\star . If $\|k_i^\star * r_i\| > 1$ then we consider that $\phi_i^\star = 0$.

If we take into account only the two closest neighbors, then the average curvature k_i^\star can be written as

$$k_i^\star = \frac{\|\mathbf{p}_i \mathbf{p}_{i-1}\| k_{i+1} + \|\mathbf{p}_i \mathbf{p}_{i+1}\| k_{i-1}}{\|\mathbf{p}_i \mathbf{p}_{i+1}\| + \|\mathbf{p}_i \mathbf{p}_{i-1}\|}.$$

Furthermore, we can compute the local average curvature over a greater neighborhood which results in increased smoothness and faster convergence. If $\sigma_i > 0$ is the topological *scale* parameter, and $l_{i,i+j}$ is the distance between \mathbf{p}_i and \mathbf{p}_{i+j} then we compute

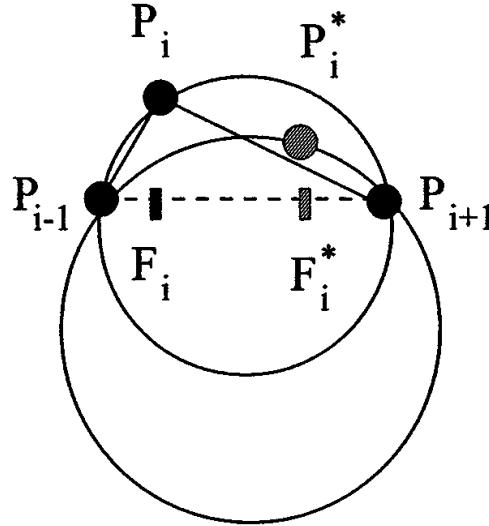


FIG. 6. The internal force associated with the curvature continuity constraint which is proportional to $\mathbf{p}_i^* - \mathbf{p}_i$.

k_i as

$$k_i^* = \frac{\sum_{j=1}^{\sigma_i} l_{i,i-j} k_{i+j} + l_{i,i+j} k_{i-j}}{\sum_{j=1}^{\sigma_i} l_{i,i-j} + l_{i,i+j}},$$

with

$$l_{i,i+j} = \sum_{k=1}^j \|\mathbf{p}_{i+k-1} \mathbf{p}_{i+k}\|.$$

This scheme generalizes the *intrinsic polynomial stabilizers* proposed in Delingette [22] that required a uniform contour parameterization. Because of this regularizing force is geometrically intrinsic, we can combine it with a curvature-based vertex spacing tangential force, thus leading to optimized computations. Finally, the stability analysis of the explicit integration scheme is linked to the choice of α_i . We have found experimentally, without yet having a formal proof, that we obtain a stable iterative scheme if we choose $\alpha_i \leq 0.5$. We believe that this simple and remarkable result is related to the geometric nature of the internal force $f_{\text{internal}} = \mathbf{p}_i^* - \mathbf{p}_i$.

Parametric contours are well-suited for implementing this curvature diffusive force. Indeed, Picinbonno [26] reported an approximative implementation of this force on level-sets. Instead of estimating the average curvature along the contour, he resorted to estimating the average curvature in a two-dimensional neighborhood of each pixel. To obtain a better estimate of the average curvature along each level-set, it would be necessary to extract geometrically all level-sets. Similarly, Chopp and Sethian [27] have reported difficulties in implementing the surface diffusion flow $\frac{\partial C}{\partial t} = \frac{d^2 k}{ds^2} \mathbf{n}$.

In Fig. 7, we show an example of contour evolution under the influence of this curvature diffusion force. We use an open contour consisting of 50 vertices and initialized as a straight line. The boundary conditions consist of the curvatures at the two contour extremities: $k_0 = -\pi$ and $k_{n-1} = \pi$. Finally, we use the maximal value of σ_i in order to speed up the convergence. We display in Fig. 7 the contour after 100, 200, and 400 iterations. It took 2.20 s on a Pentium II 400 MHz to iterate 400 times. The resulting curve is a simple clothoid curve having equation $C(u) = \{FC(u), FS(u)\}$, where $FC(u)$ and $FS(u)$ are the Fresnel cosine and sine functions, respectively. In order to estimate the accuracy of the

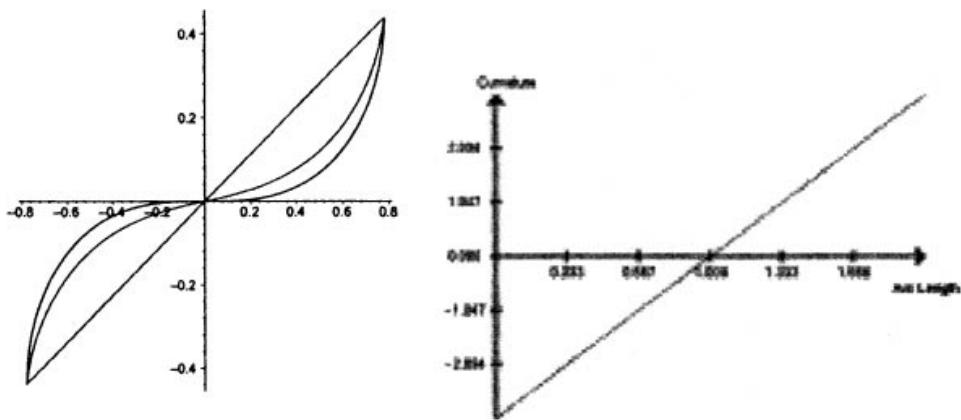


FIG. 7. (Left) Curve evolution from a line to a clothoid curve fixed at its two extremities. (Right) Curvature profile of the deformed contour.

resulting contour, we have fixed the two extremities at positions $\{FC(-1), FS(-1)\}$ and $\{FC(1), FS(1)\}$. We have verified that the relative position error of each vertex was less than 1% after 400 iterations. The curvature profile of the resulting curve follows equation: $k = \pi s$.

Figure 8 shows the evolution of a closed curved under the flow $\frac{\partial C}{\partial t} = (\frac{1}{2s_0} \int_{-s_0}^{s_0} k ds - k)\mathbf{n}$. The initial contour consists of the contour displayed in Fig. 4 with a curvature-based vertex spacing. We chose a deformation scale parameter $\sigma_i = 5$. The final circle is obtained after 7500 iterations. During the deformation, we fix all metric parameters. Therefore, the circle resulting from this contour evolution is also nonuniformly discretized. This example demonstrates that this regularizing force is clearly independent on the contour parameterization.

Finally, in Fig. 9, we show the importance of the shrinking effect on the reconstruction of a curvy boundary. The rightmost image corresponds to the action of Laplacian smoothing that

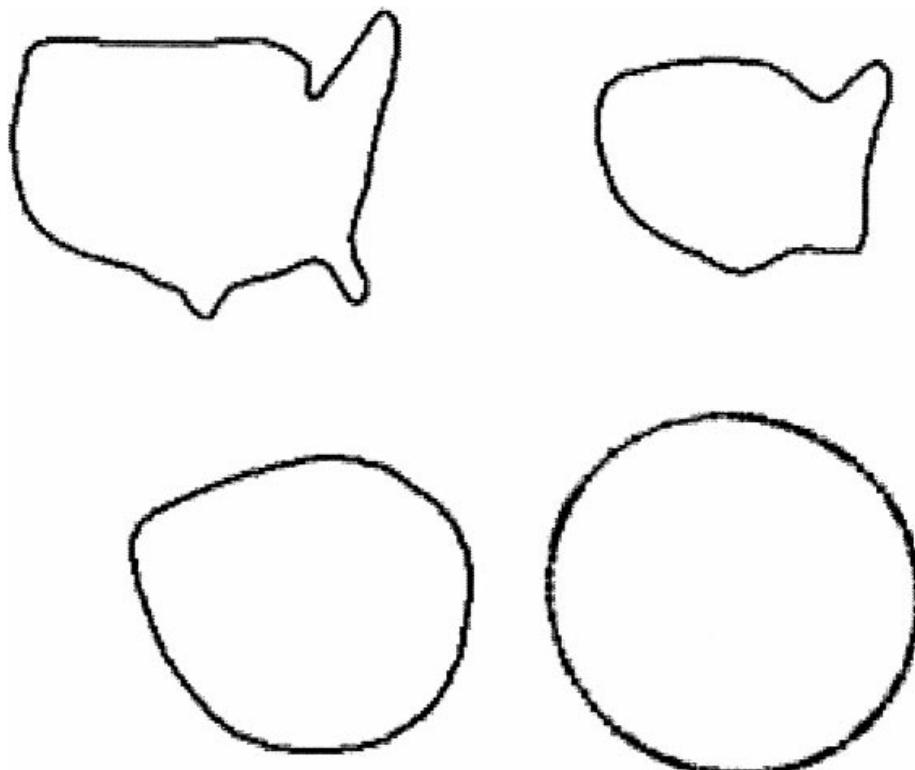


FIG. 8. Curve evolution under the curvature diffusive flow starting from Fig. 4 toward a circle: (Bottom right) Nonuniformly discretized circle resulting from the curve evolution.

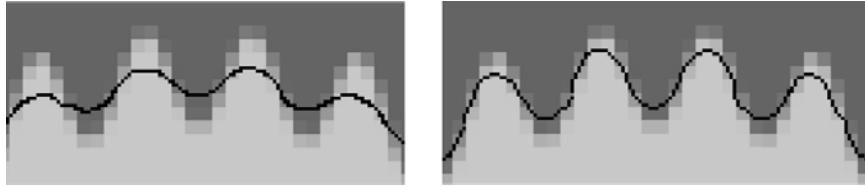


FIG. 9. Segmentation of a curvy shape using (left) Laplacian smoothing and (right) curvature diffusive internal force expressions.

tries to minimize the total curve length. On the other hand, when applying the curvature diffusion regularization of Eq. (8) with the same parameters α_i , the deformable contour is able to capture most of the curvy shape while enforcing a higher order of geometric continuity.

4. TOPOLOGY CONSTRAINTS

Automatic topology changes of parametric contours have been previously proposed by Leitner and Cinquin [11], McInerney and Terzopoulos [1], and Lachaud and Montanvert [13]. Using McInerney *et al.* approach, all topological changes occur by computing the contour intersections with a simplicial decomposition of space. The contour is reparameterized at each iteration, the intersections with the simplicial domain being used as the new vertices. Recently Lachaud *et al.* [13] introduced topologically adaptive deformable surfaces where self-intersections are detected based on the distance between vertices. Our algorithm is also based on a regular lattice for detecting all contour intersections. However, the regular grid is not used for changing the contour parameterization and furthermore topology changes result from the application of topological operators. Therefore unlike previous approaches, we propose to completely decouple the physical behavior of active contours (contour resolution and geometric regularity) with their topological behavior in order to provide a very flexible scheme. Finally, our framework applies to closed or opened contours.

In this section, we propose an algorithm for changing the topology of a contour. A contour topology is defined by the number of its connected components and whether each of its components is closed or opened. Therefore, the topology constraint consists in changing the number of connected components or in closing or opening a given component. Our approach consists in using two basic topological operators. The first operator illustrated in Fig. 10 consists in merging two contour edges. Depending whether the edges belong to the same connected component or not, this operator creates or removes a connected component. The second topological operator consists in closing or opening a connected component.

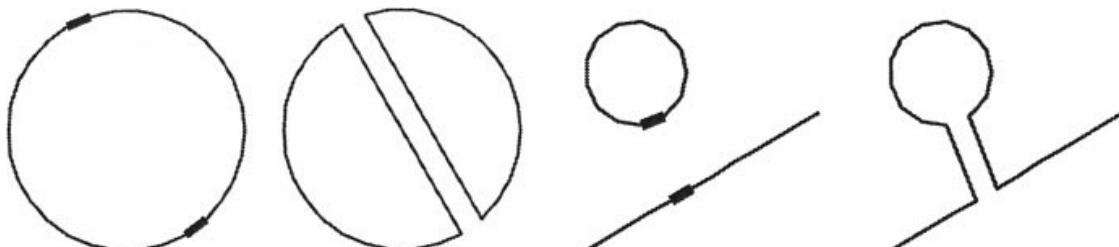


FIG. 10. Topological operator applied to two edges of the same connected component (left) or two different connected components (right).

Our approach for modifying a contour topology can be decomposed into three steps:

1. building of a grid-based data structure;
2. detection of intersecting edges;
3. application of topological operators based on these intersecting edges.

The first step creates a data structure where the collision detection between contour connected components is computationally efficient. The second determines the geometric intersection between edges and the last step actually performs all topological modifications. Figure 11 details the topology change algorithm steps explained in the following sections.

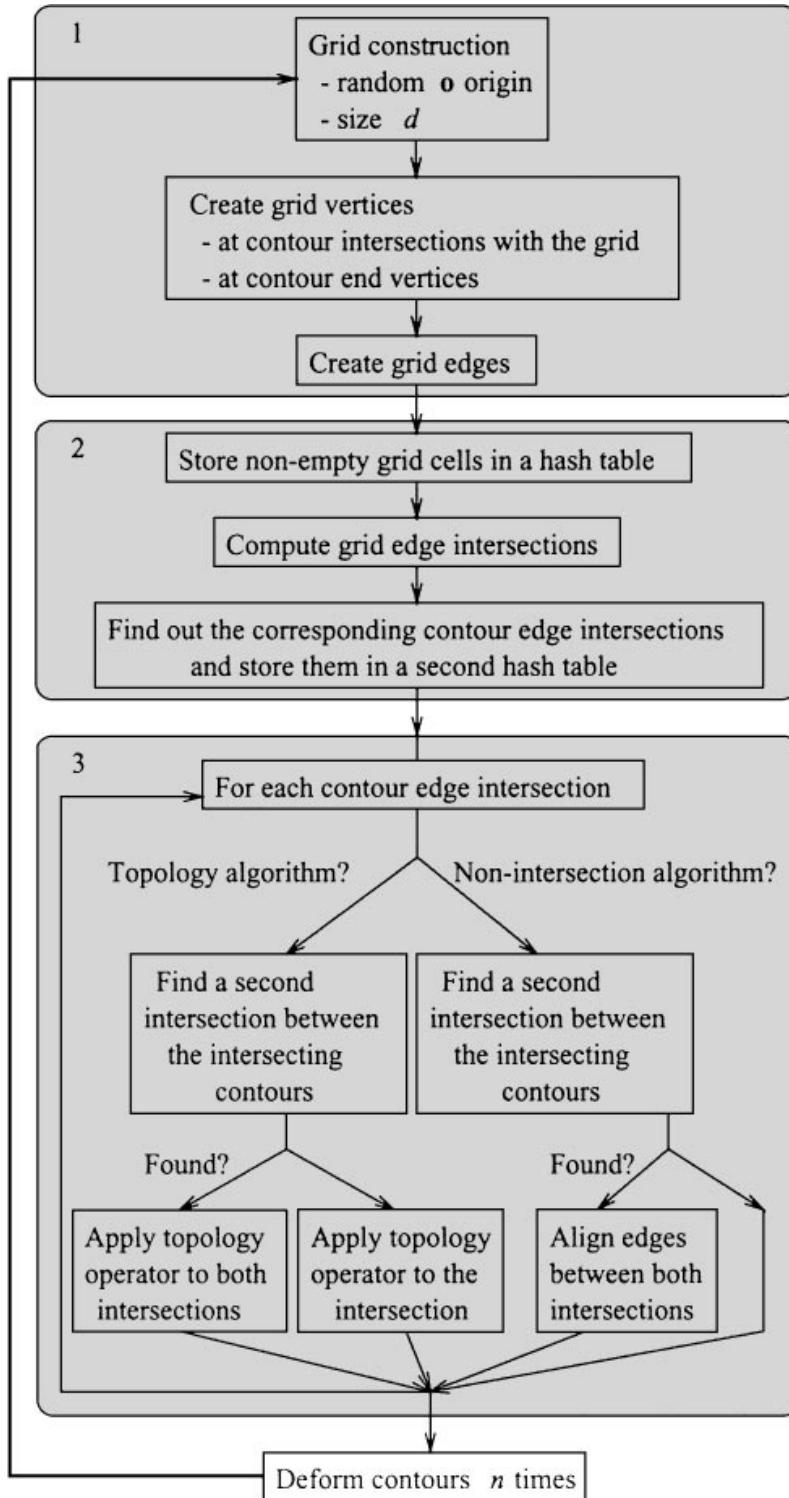


FIG. 11. Topology change algorithm steps.

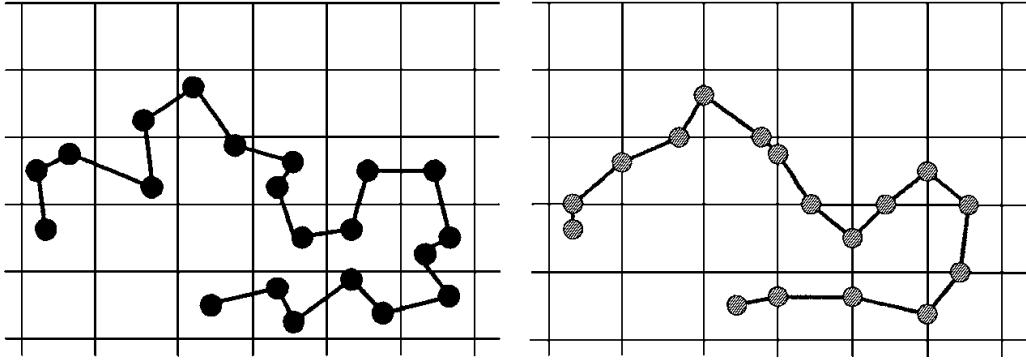


FIG. 12. (Left) the original contour with the regular grid. (Right) The contour decomposed on the regular grid.

4.1. Data Structure for the Detection of Contour Intersections

Efficiently finding pairs of intersecting edges with a line-sweep algorithm has a complexity of $O(n \log n)$ where n is the number of vertices (or edges). Our algorithm is based on a regular grid of size d and has a complexity linear with the ratio \mathcal{L}/d where \mathcal{L} is the length of the contour. Therefore, unlike the approach proposed in [1], our approach is not region-based (inside or outside regions) but only uses the polygonal description of the contour.

The two-dimensional Euclidean space with a reference frame $(\mathbf{o}, \mathbf{x}, \mathbf{y})$ is decomposed into a regular square grid whose size d is user-defined. The influence of the grid size d is discussed in Section 5.1. In this regular lattice, we define a point of row and column indices r and c as the point of Cartesian coordinates $\mathbf{o}_{\text{grid}} + r\mathbf{x} + c\mathbf{y}$, where \mathbf{o}_{grid} is the grid origin point. This point is randomly determined each time topology constraints are activated in order to make the algorithm independent of the origin choice. Furthermore, we define a square cell of index (r, c) as the square determined by the four points of indices $(r, c), (r + 1, c), (r + 1, c + 1)$, and $(r, c + 1)$. In each cell of the regular grid, we store the list of *grid edges* corresponding to the decomposition of a contour on the regular lattice (Fig. 12).

In order to build the sampled contour, we scan all edges of each connected component. For each edge, we test if it intersects any row or columns of the regular lattice. Since the row and column directions correspond to the directions \mathbf{x} and \mathbf{y} of the coordinate frame, these intersection tests are efficiently computed. When a vertex position coincides with a grid point (which is very unlikely), then one cell is consistently chosen. Each time an intersection with the row or column direction is found, a *grid vertex* is created and the intersecting contour edge is stored in the grid vertex. Furthermore, a grid vertex is stored in a grid edge structure. A grid edge is either a pair of grid vertices or a grid vertex associated with an end vertex (when the connected component is an opened line). Finally, the grid edge is appended to the list of grid edges inside the corresponding grid cell.

4.2. Finding Intersecting Grid Edges

In order to optimize memory space, we store all nonempty grid cells in a hash table, hashed by its row and column indices. The number of grid cells is proportional to the length \mathcal{L} of the contour. In order to detect possible contour intersections, each entry to the hash table is scanned. For each cell containing n grid edges with $n > 1$, we test the intersection between all pairs of grid edges (see Fig. 13, left). Since each grid edge is geometrically

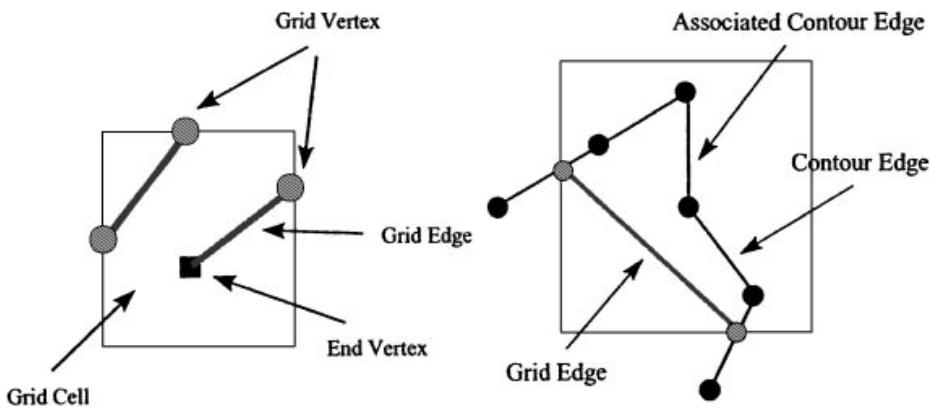


FIG. 13. (Left) Definition of a grid vertex, a grid edge, and grid cell. (Right) Contour edge associated with a grid edge.

represented by a line segment, this intersection test requires the evaluation of only two dot products.

Once a pair of grid edges has been found to intersect, a pair of contour edges must be associated for the application of topological operators (see Section 4.3). Because a contour edge is stored in each grid vertex, one contour edge can be associated with each grid edge. Thus, we associate with each grid edge, the middle of these two contour edges (in terms of topological distance), as shown in Fig. 13b.

Our contour edge intersection algorithm has the following properties:

1. If one pair of grid edges intersects then there is at least one pair of contour edges that intersects inside this grid cell.
2. If a pair of contour edges intersects and if the corresponding intersecting area is greater than $d \times d$ then there is a corresponding pair of intersecting grid edges.

In another words, our method does not detect all intersections but is guaranteed to detect all intersections having an area greater than $d \times d$. In practice, since the grid origin \mathbf{o}_{grid} is randomly determined each time the topology constraint is enforced, we found that our algorithm detected all intersections that are relevant for performing topology changes.

4.3. Applying Topological Operators

All pairs of intersecting contour edges are stored inside a hash table for efficient retrieval. Since in general two connected components intersect each other at two edges, given a pair of intersecting contour edges, we search for the closest pair of intersecting contour edges based on topological distance. If such a pair is found, it is processed right after the current pair.

The processing of a pair of intersecting contour edges consists in the following tasks. If both edges belong to the same connected component, then the smallest number of edges between the two edges is counted. If the two edges are too close to each other, then the two edges are not merged in order to avoid creating connected components that are too small. This topological distance threshold must be greater or equal to three (for the creation of a triangle) and is equal to 8 in the examples shown in this paper. In all other cases, the two edges are merged with the topological operator presented in Fig. 10. Finally, we update the list of intersecting edge pairs by removing from the hash table all edge pairs involving any of the two contour edges that have been merged.

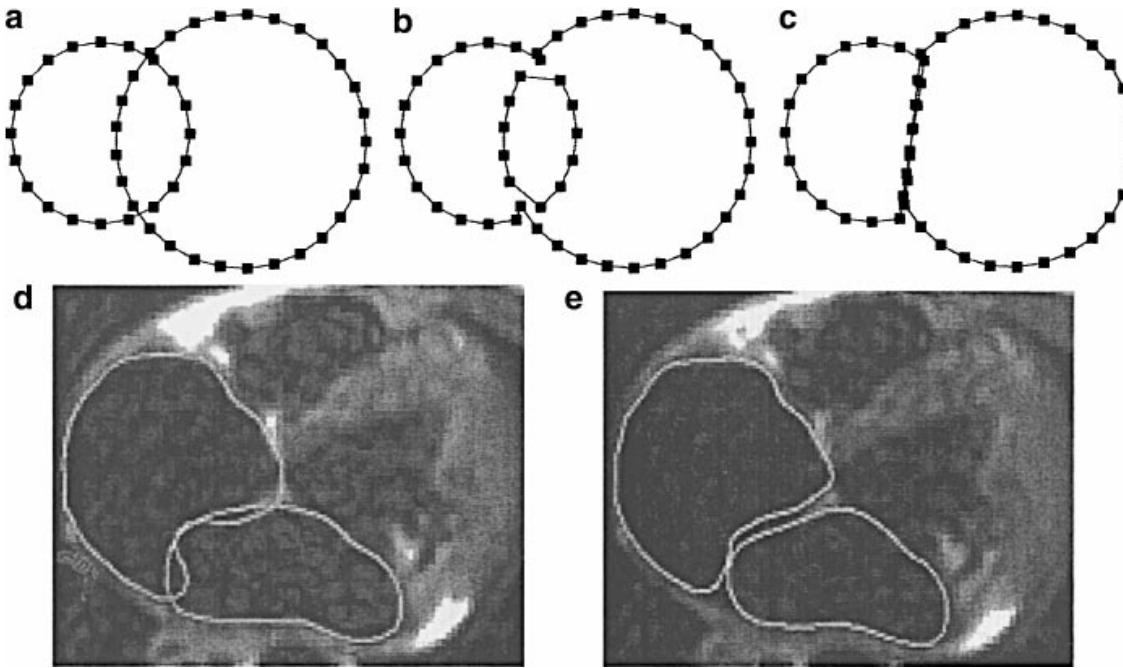


FIG. 14. (a) Two intersecting connected components; (b) after merging the two pairs of intersecting edges; (c) after aligning vertices along the intersecting edges. (d and e) Examples of two active contours reconstructing the right and left atrium in an MR image without any topology changes (c) and with a repulsive behavior between each components.

4.4. Preventing the Fusion of Connected Components

The algorithm presented in the previous sections merges intersecting edges regardless of the nature of the intersection. If it corresponds to a self-intersection, then a new connected component is created, otherwise two connected components are merged.

With our framework, it is possible to prevent the merging of two distinct connected components while allowing the removal of self-intersections. To do so, given a pair of intersecting contour edges belonging to distinct connected components, we search for the closest pair of intersecting contour edges as described in the previous section. If such pair is found, instead of performing two edge merging operations, we simply align all the vertices that are located between intersecting edges belonging to the same connected component (see Fig. 14, top). Thus, each component pushes back all neighboring components. In Fig. 14 (bottom), we show an example of image segmentation where this repulsive behavior between components is very useful in segmenting the two heart ventricles.

Therefore, our scheme not only allows us to change contour topologies but it also provides a simple mechanism for controlling these topology changes. This is not the case of the typical implementation of the level-set method where we cannot prevent the fusion of two distinct connected components. Several researchers including Zhao *et al.* [28] have proposed multiple interface extensions of the level-set method. However, it currently seems to be limited to a limited number of interfaces with constant speeds.

4.5. Image Borders

Image borders are natural boundary conditions for the contour deformation process. When a contour reaches an image border, image gradient information becomes meaningless. Furthermore, if a contour is submitted to a balloon force, it would indefinitely inflate outside the image borders. Therefore, since we deal with close as well as open curves, we propose to

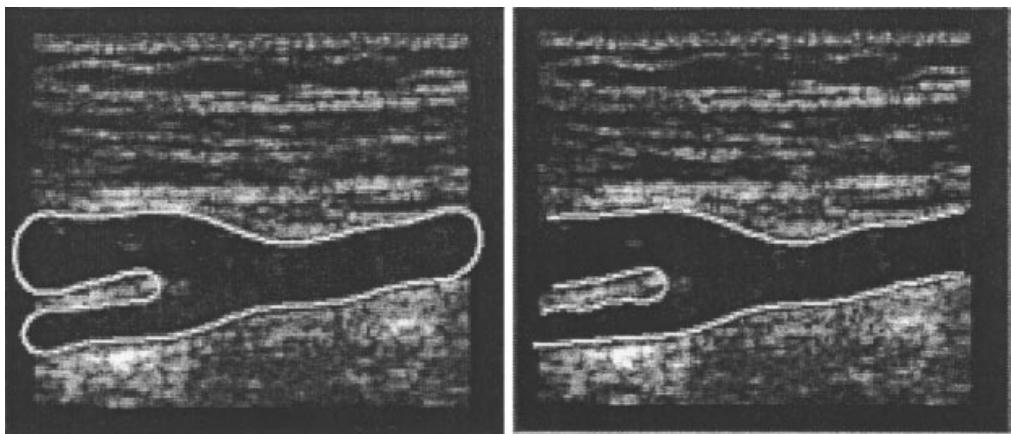


FIG. 15. Segmentation of a carotid branching from an ultrasound image without (left) and with (right) automatic topology changes of the contour on the image borders.

break all contours reaching the borders of an image. Figure 15 shows the result of a carotid branching segmentation from an ultrasound image using a deformable contour initialized as an ellipsoid. In the leftmost image, no border constraint has been used and the contour flows outside the image borders. On the rightmost image, the contour has been automatically cut at the contact of the image borders.

Figure 16 details the contour evolution from its initial position (upper left) to its carotid branching shape. Each contour vertex is displayed as a small square. The contour is regularized by a curvature diffusion constraint. The contour resolution constraint is used every 10 iterations to dynamically add vertices when the contour length increases. Image border

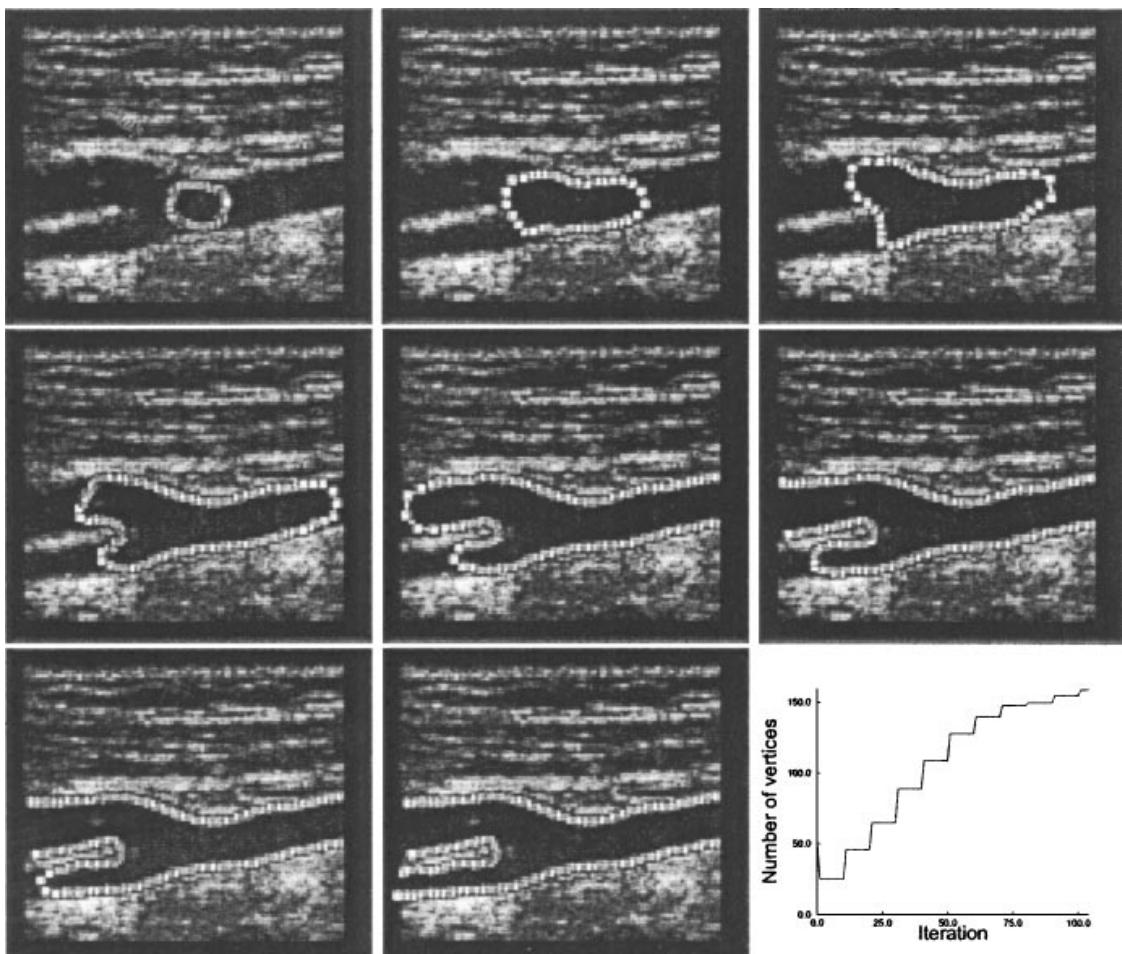


FIG. 16. Evolution of a closed curve toward an open carotid branching in an ultrasound image.

constraints are computed every 10 iterations as well to detect all contour intersections with the image border.

Figure 16 (bottom right) shows the evolution of the number of contour vertices over time.

5. RESULTS

5.1. Automatic Topology Adaptation

We evaluate the performance of our automatic topology adaptation algorithm on the example in Fig. 17. The contour, consisting of 50 vertices, is deformed from a circular shape toward two circular connected components. The computation time for building the data structure described in Section 4.1 is displayed in Fig. 18 as a function of the grid size d . It varies from 175 to 1 ms when the grid size increases from 0.17 to 10 image pixels on an Alphastation 500 MHz. The computation time for applying the topological operators can be neglected in general. When the grid size is equal to the mean edge distance (around 2 pixels), the computation time needed to detect edge intersections becomes almost equal to the computation time needed to deform the contour during one iteration (4.8 ms).

When the grid size increases, the contour sampling on the regular grid becomes sparse and therefore some contour intersections may not be detected. However, we have verified that topological changes still occur if we choose a grid size corresponding to 20 image pixels with contour intersections checked every 20 iterations. In practice, we choose a conservative option with a grid size equal to the average edge length and with a frequency for topology changes of 5 iterations which implies an approximate additional computation time of 20%.

5.2. Evolution of Closed and Opened Contours

In Fig. 19, we show the evolution under Laplacian smoothing (equivalent to curvature motion as described in Section 3) of the Ying-Yang symbol represented with three circles

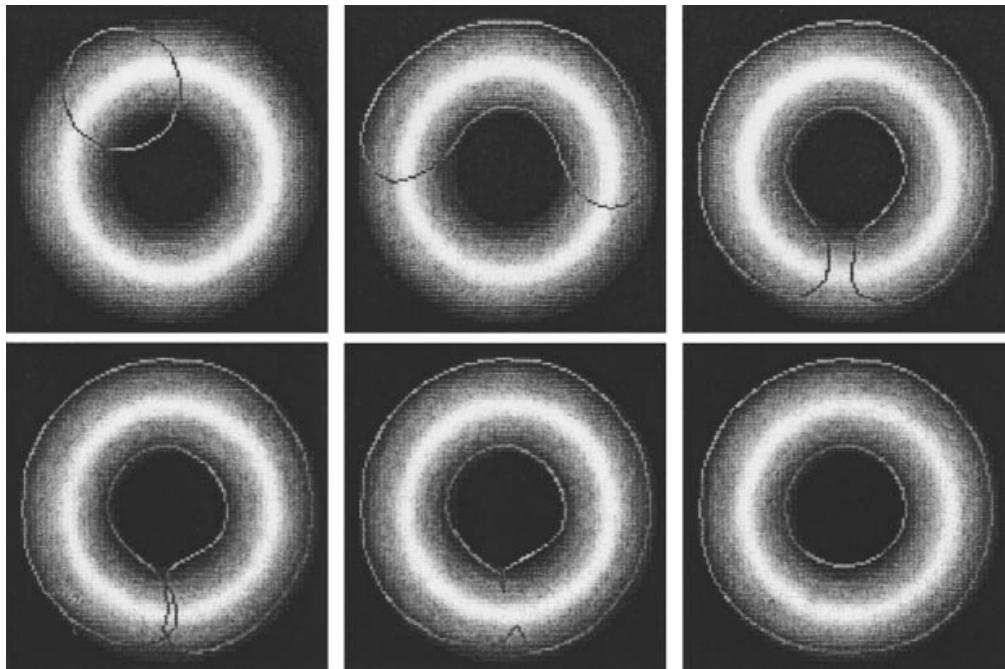


FIG. 17. Segmentation of a ring shape by an active contour with a balloon force.

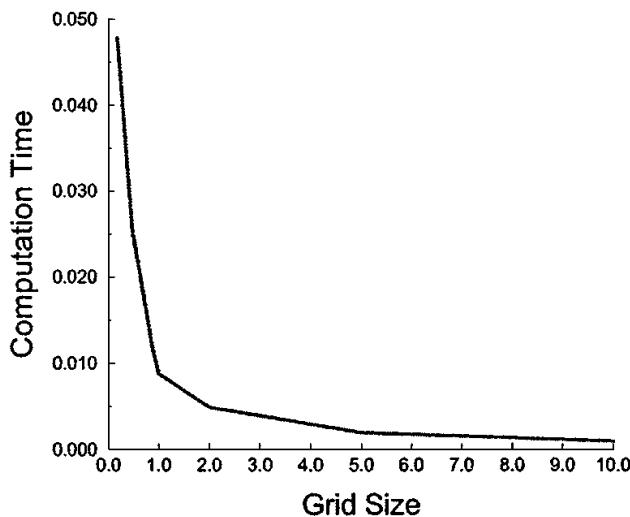


FIG. 18. Plot of the computation time for testing contour intersections as a function of the grid size.

and one central opened curve. We apply topology change and resolution constraints on these contours at each iteration in order to ensure a smooth contour evolution.

5.3. Medical Image Example

This last example illustrates the segmentation of an aortic arch angiography. Figure 20 shows the initial contour (top left) and its evolution toward the aorta and the main vessels. The image segmentation is made difficult by the contrast variations. The contour is attracted toward the highest gradient value in the vicinity of each vertex. External forces are computed as a function of the vertex distance to a gradient point to avoid oscillations around image edges. The contour is regularized by a curvature diffusion constraint. Internal forces are responsible for the contour parameterization (tangential displacement) and the external forces are backprojected on the normal direction. The contour resolution constraint is applied every 10 iterations which makes the resampling overhead very low. Topology constraints are computed every 5 iterations on a 4-pixel grid size to fuse the self-intersecting contour parts. Intersections with image borders are computed every 10 iterations.

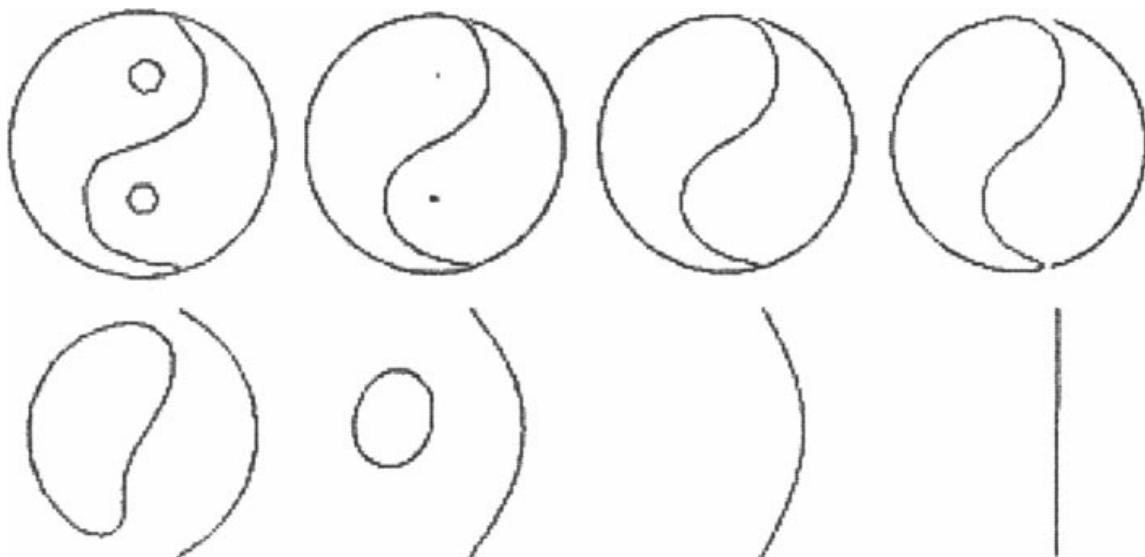


FIG. 19. Evolution of three closed contours and one opened contour under Laplacian smoothing.

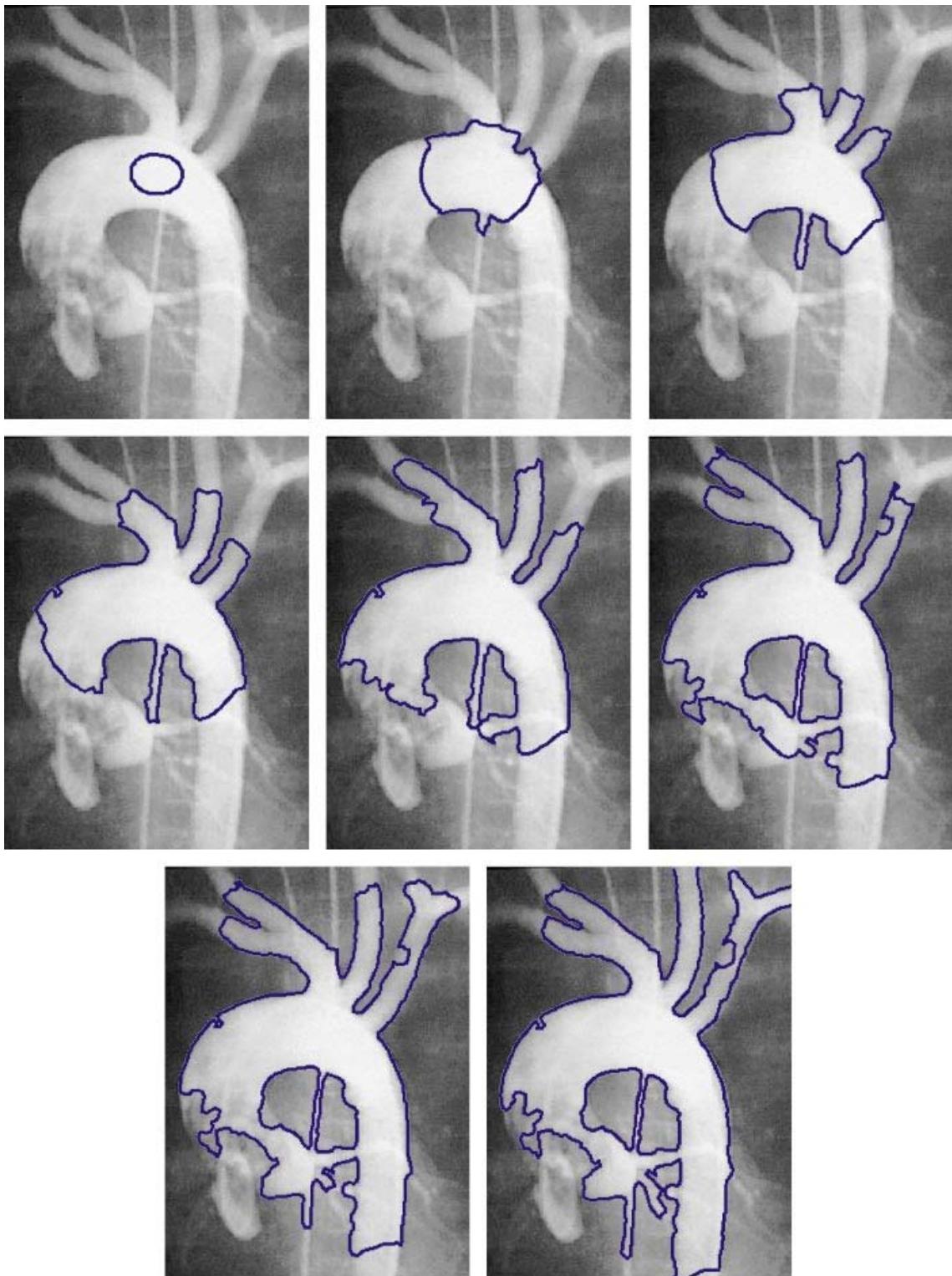


FIG. 20. Evolution of a closed curve toward the aortic arch and the branching vessels.

6. COMPARISON WITH THE LEVEL-SET METHOD

The level-set method has been introduced in the computer vision community by Malladi *et al.* [7]. It is based on an implicit representation of contours that allows topology changes during the contours evolution. However, the major drawbacks of level-set methods are related to their difficult user interaction and their computational cost, although some speed-up algorithms based on narrow bands have been proposed.

The key idea of the implicit representation of contours is to embed the two-dimensional contour in a three-dimensional surface. A contour is defined as the level-set of that surface

corresponding to $z = 0$. Let $\Psi : \mathbb{R}^2 \rightarrow \mathbb{R}$ denote the level-set function. The contour \mathcal{C} is defined by

$$\mathcal{C} = \{\mathbf{p} \in \mathbb{R}^2 \mid \Psi(\mathbf{p}) = 0\}.$$

Since it corresponds to the zero-crossing over a scalar field defined on a regular grid, \mathcal{C} represents a set of closed curves and opened curves having their extremities on the regular grid boundary. However, in practice, authors have considered only the case of closed contours. As the surface Ψ continuously evolves over time, \mathcal{C} is deformed and may encounter topology changes. The evolution equation is governed by a Hamilton-Jacobi equation,

$$\frac{\partial \Psi}{\partial t} + v \|\nabla \Psi\| = 0, \quad (11)$$

where v is a propagation speed term. Given that v is regular enough, Ψ remains a function as it evolves.

A formal comparison between both the parametric and level-set approaches has been established recently by Aubert *et al.* [29] in the case of geodesic snakes. In this section, we propose a practical comparison between both approaches including implementation issues.

6.1. Level-Set Implementation

We describe here the level-set implementation used to compare our discrete contours with the level-set method. The level-set function Ψ is discretized on a rectangular grid whose resolution corresponds to the image pixel size.

The propagation speed term v is designed to attract \mathcal{C} toward object boundaries extracted from the image using a gradient operator with an additional regularizing term. Thus

$$v(\mathbf{p}) = \beta(\mathbf{p})(\kappa(\mathbf{p}) + c), \quad (12)$$

where

- $\kappa(\mathbf{p}) = \frac{\frac{\partial^2 \Psi}{\partial x^2} \frac{\partial \Psi^2}{\partial y} - 2 \frac{\partial \Psi}{\partial x} \frac{\partial \Psi}{\partial y} \frac{\partial^2 \Psi}{\partial x \partial y} + \frac{\partial^2 \Psi}{\partial y^2} \frac{\partial \Psi^2}{\partial x}}{\left(\frac{\partial \Psi^2}{\partial y} \frac{\partial \Psi^2}{\partial x} \right)^{\frac{3}{2}}}$ denotes the contour curvature at point \mathbf{p} . This is a regularizing term corresponding to the mean curvature flow: each vertex is moved toward its center of curvature.

• c is a constant resulting in a uniform contour evolution inward or outward depending on its sign. It is equivalent to the application of a balloon force [20].

• $\beta(\mathbf{p}) \in [0, 1]$ is a multiplicative coefficient dependent on the image gradient norm at point \mathbf{p} . A truncated decreasing function illustrated in Fig. 21 is used, where I represents

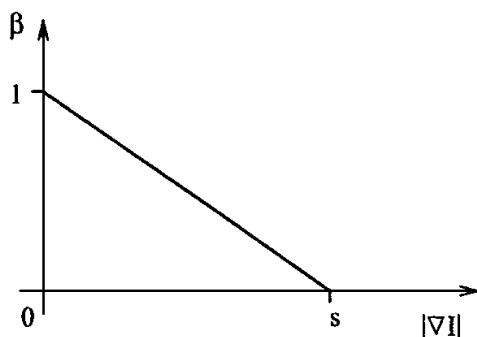


FIG. 21. Multiplicative coefficient $\beta(\mathbf{p})$ for the speed term.

the image intensity. When \mathcal{C} moves across pixels of high gradient, this term slows down the level-set propagation. The parameter s is determined as a threshold on the minimum gradient norm (i.e., the minimal image boundary strength) required to stop a level-set evolution.

Equation (11) is discretized in space using finite differences in time using an explicit scheme. It reduces to

$$\Psi_{ij}^{t+\delta t} = \Psi_{ij}^t - \delta t v_{ij} \|\nabla_{ij} \Psi_{ij}^t\|,$$

where Ψ_{ij} and v_{ij} denote the values of Ψ and v at pixel (i, j) , δt is the discrete time step, and ∇_{ij} is a finite difference operator.

In order to speed up the level-set algorithm, we use a narrow band method [7], which requires periodical reinitialization of the level-set contour.

In order to compare active contours to level-sets, we compute external forces similar to Eq. (12) at discrete contour vertices. First, we use mean curvature motion described in Section 3 as the governing internal force. Then we have implemented for the external force, a balloon force (a constant force applied along the contour normal either inward or outward) weighted by the coefficient β proposed in Fig. 21. Therefore, we were able to use the same gradient threshold in both approaches.

6.2. Torus Example

We first propose to compare both approaches on the synthetic image shown in Fig. 22 on the left. This image has two distinct connected components. It was obtained by intersecting a 3D torus with a plane as illustrated in center image. Due to partial volume effect, the gradient magnitude image shown on the right is not completely dark near the center region.

A discrete contour is initialized around the two components as illustrated in Fig. 23 (left). In this example, the contour is regularized with a curvature continuity internal force. A medium grid size (8 pixels resolution) is used and topology constraints are computed every 10 iterations. Throughout the deformation process, vertices are added and removed to have similar edge length along the contour.

The corresponding initial level-set is shown in Fig. 23 (center). The irregular square corners are due to a rounding approximation of the level-set initialization procedure but they are quickly regularized and have no effect on the deformation process. A 7-pixel-wide narrow band appeared to optimize the convergence time. A 0.3 time step is used. It is the maximal value below which the evolving curve is stable.

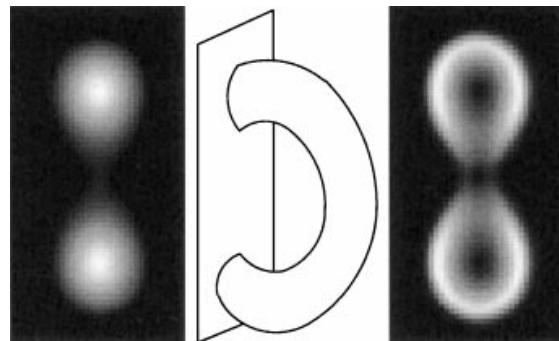


FIG. 22. Image with two connected components (left) obtained by intersecting a torus with a plane (center) and its gradient norm magnitude (right).

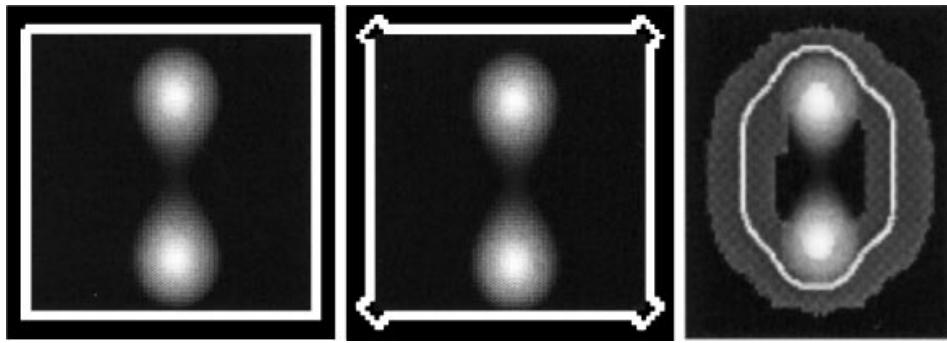


FIG. 23. Initial discrete contour (left), initial level-set (center), and level-set narrow band (right).

Figure 24 shows the convergence of the discrete contour (top line) and the level-set (bottom line). Due to different regularizing constraints, final shapes are slightly different in both cases. The discrete contour converges in 0.42 s, opposed to 3.30 s for the level-set; that is a 7.85 acceleration factor in favor of the discrete contour. These time measures do not include any display procedures since it would greatly penalize the level-set method. The difference of computational time is due to the small vertex number used for the discrete contour (varying between 36 and 48 vertices) compared to the much greater number of sites (from 1709 up to 3710) updated in the level-set narrow band.

6.3. Parameter Comparison

As much as possible, the same parameter values were used to compare discrete contours and level-sets. The time step parameter is specific to level-sets and in our experiments it appeared to be difficult to optimize. Only observing the level-set behavior for different time steps allowed us to set an optimal value. Also, few parameters needed to be set for the discrete contour topology algorithm. However, the grid size and the topology algorithm frequency are much more intuitive parameters to set since they are clearly related to pixel size and the number of iterations.

The gradient threshold s is common to both discrete contour and level-set methods. In this experiment we have set s to a value of 140. This value is high enough to “break” the contour through the gradient barrier between the two connected components but it is still low enough to prevent it from entering each component. Using a slightly lower value slows

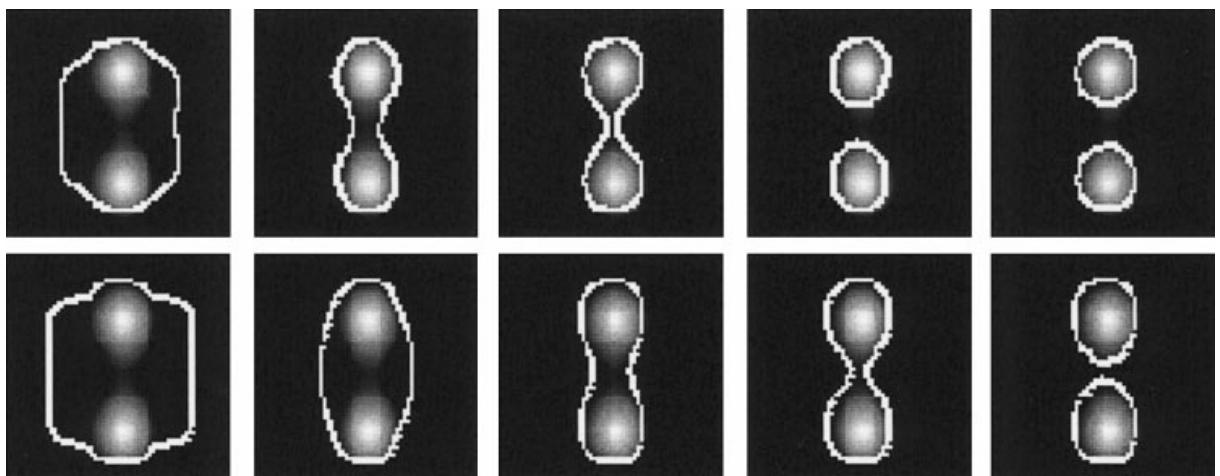


FIG. 24. Discrete contour (top row) and level-set (bottom row) deformations.

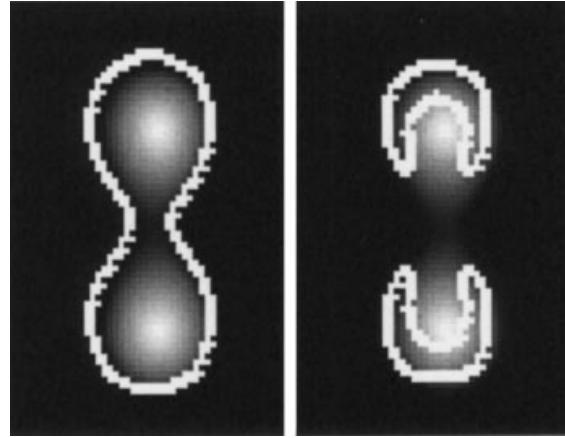


FIG. 25. Level-set convergence with a too-weak (left) or a too-strong (right) s parameter value.

downs the level-set evolution and stops it on the center potential barrier (see Fig. 25, left). On the contrary, with a slightly higher value, the level-set contour crosses the barrier and converges toward the components inner regions (Fig. 25, right). In practice, we found that the level-set evolution was particularly sensitive to s variations. On the other hand, due to the force computation method of the discrete contour implying a wider range scanning and the different regularizing constraint, the discrete contour is much less sensitive to s variations.

6.4. Synthetic Data

This experiment shows the ability of the discrete contour topology algorithm to follow difficult topology changes. We use a synthetic fractal image showing a number of small connected components. Figure 26 shows the discrete contour convergence in the image while Fig. 27 shows the level set convergence.

In both cases, the initial contour is a square located at the image border. It evolves under a deflation force that stops on strong image boundaries. For the discrete contour, a small grid size is used due to the small image structure size (4-pixels grid size and 5-iteration algorithm frequency). A weak regularizing constraint allows the contour to segment the square corners. The contour is checked every 10 iterations to add the necessary vertices.

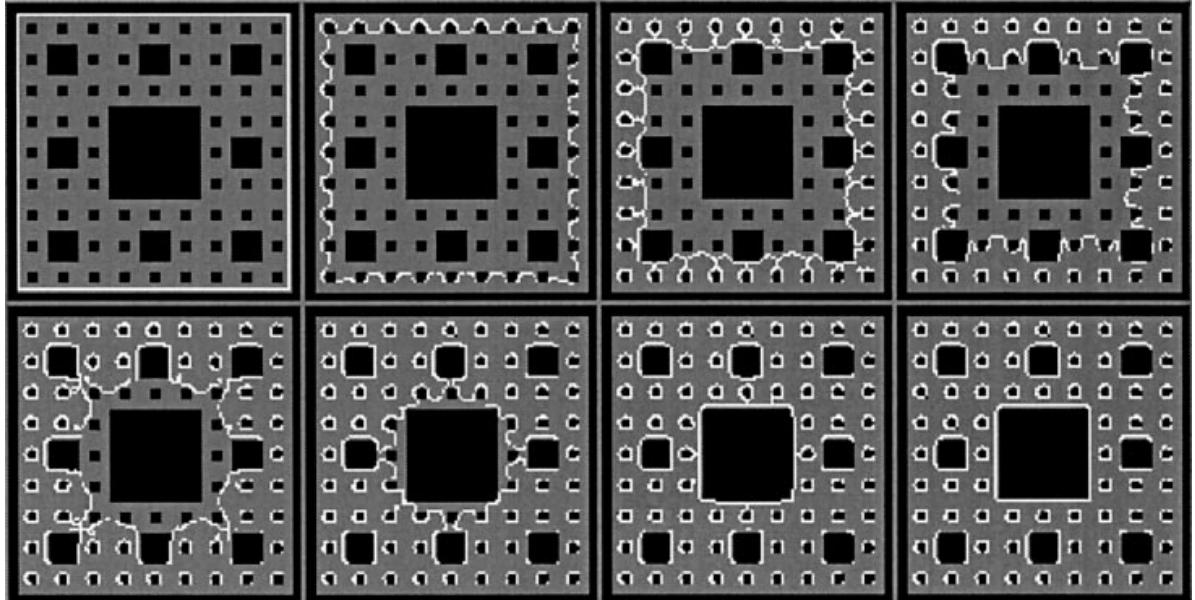


FIG. 26. Discrete contour convergence in a fractal image.

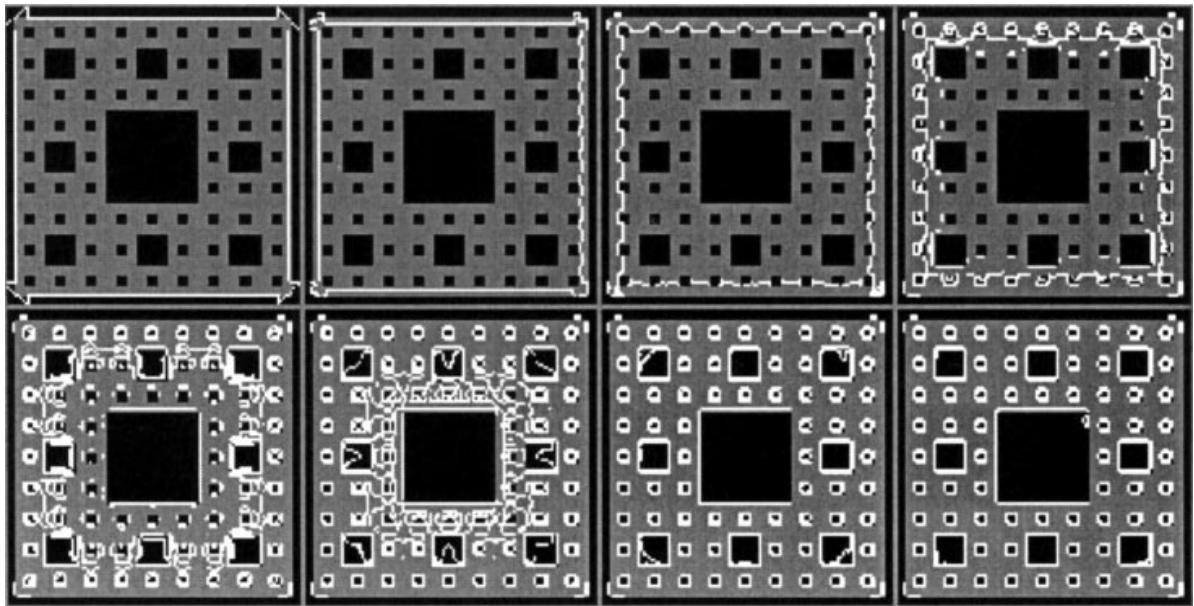


FIG. 27. Level-set convergence in a fractal image.

A 0.3 time step is used for the level-set. This high value leads to a rather unstable behavior as can be seen in Fig. 27. As the level-set contours gradually fill in the whole image, we have verified that the convergence time is not minimized by using any narrow bands.

Figure 28 shows the computational time function as a function of the number of iterations until the contour and the level-set converge. The level-set convergence time is linear while the discrete contour slows down as vertices are added. However, the discrete contour converges almost four times faster than the level-set method.

6.5. Medical Images

Finally, we show a medical image segmentation example using both approaches. Figure 29 shows a discrete contour deformation in a vertebra CT-scan image. The contour is resampled and the topology algorithm is applied every 5 iterations. It converges in 3.2 s. The corresponding segmentation using a level-set is shown in Fig. 30. A 0.5 time step value and a strong inflating force are used to speed up the convergence. The deformation

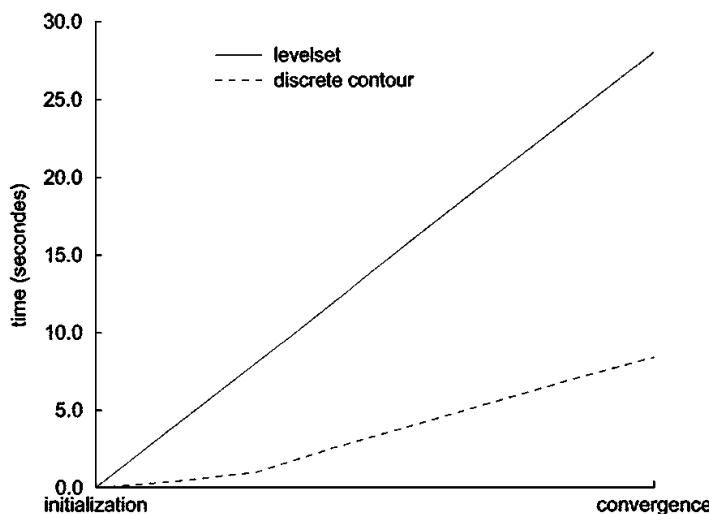


FIG. 28. Level-set (full line) and discrete contour (dashed line) convergence time.

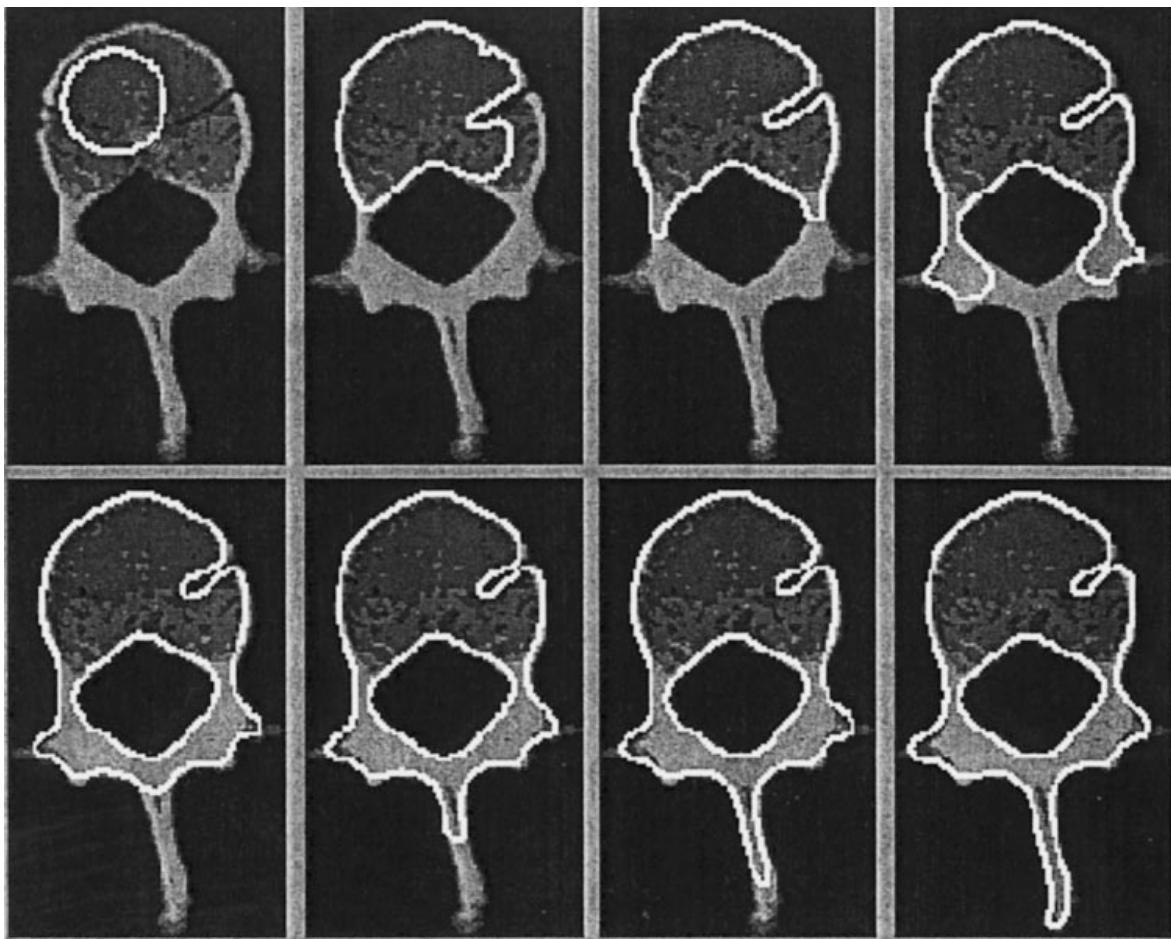


FIG. 29. Discrete contour convergence in a vertebra CT-scan image.

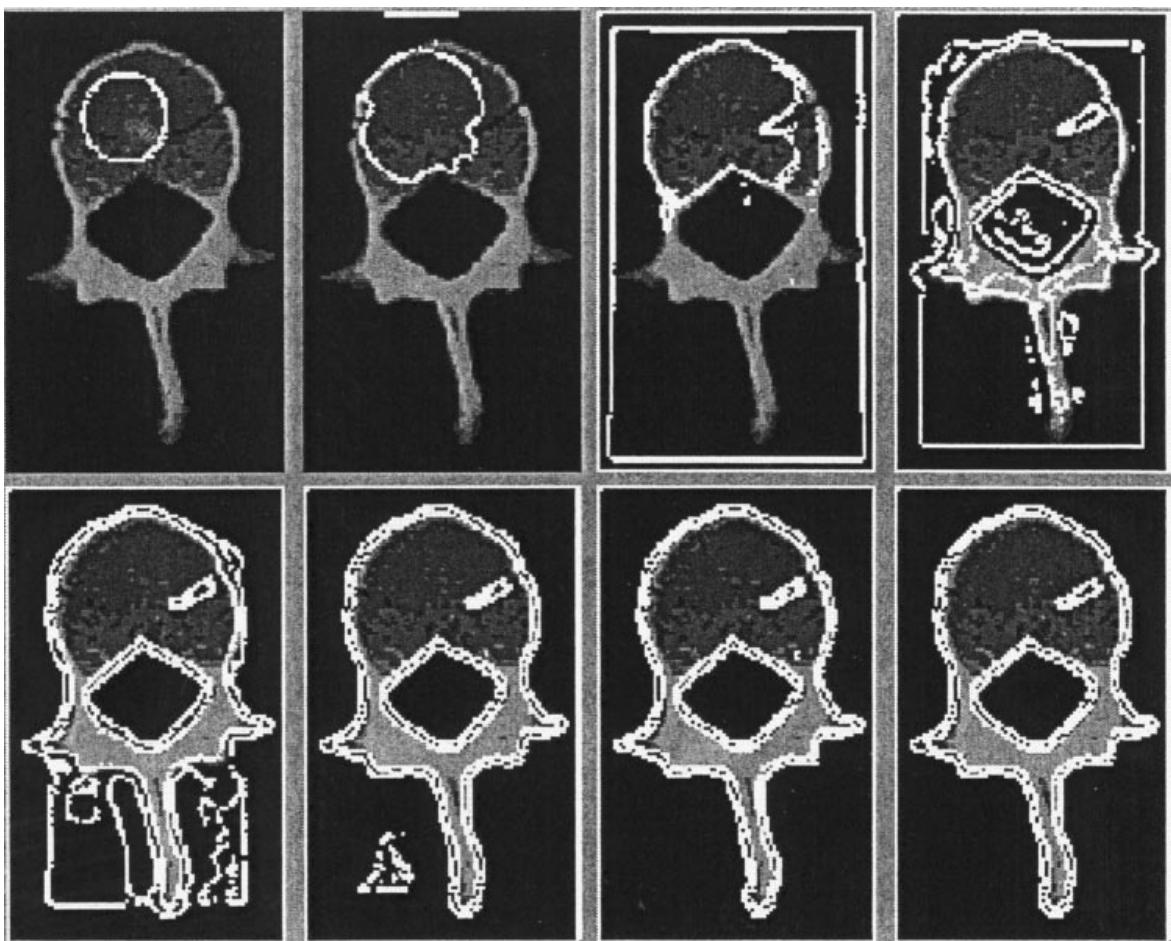


FIG. 30. Level-set convergence in a vertebra CT-scan image.

process is unstable but almost stabilizes. The instabilities cause a double segmentation of the inner and the outer contour of the vertebra. The image border is also segmented due to level-set update method border effects. The convergence time for the level-set is 4.2 s.

7. CONCLUSION

We have introduced three contributions related to contour discretization, regularization, and topology changes. We believe that these three algorithms greatly improve the generality of parametric active contours while preserving their computational efficiency. Furthermore, these algorithms are controlled by simple parameters that are easily understood by the user.

For the internal force, two parameters should be set. The first parameter α_i must be chosen between 0 and 1 is used to set the amount of smoothing. Because the resulting shape only depends on the ratio α_i/β_i , we often set α_i to 1.0 and chose to modify β_i instead. The second parameter is the deformation scale parameter σ_i that is an integer that belongs to the range $[1, \dots, N/2]$. In general, we choose a value of 1. When the data is sparse or when the contour resolution is greater than the finest level of detail required, we increase this value. Setting the external force parameter β_i typically depends on the confidence in the data set. For resolution and topology constraint algorithms, distance parameters must be provided as well as the frequency at which they apply. Given an image, all these parameters can be initialized automatically to meaningful values providing good results in most cases.

Finally, we have compared the efficiency of this approach with the level set method. These experiments seem to conclude that our approach is at least twice as fast as the implicit implementation. Above all, we believe that the most important advantage of parametric active contours is their user interactivity.

ACKNOWLEDGMENTS

We thank the members of the Epidiaure project for stimulating discussion related to this topic. Also, we thank J. Darcourt from the Hospital University of Nice, France for providing the MR cardiac images.

REFERENCES

1. T. McInerney and D. Terzopoulos, Topologically adaptable snakes, in *International Conference on Computer Vision (ICCV'95)*, pp. 840–845, Cambridge, MA, 1995.
2. M. Kass, A. Witkin, and D. Terzopoulos, Snakes: Active contour models. *Internat. J. Comput. Vision* **1**, 1988, 321–331.
3. S. Menet, P. Saint-Marc, and G. Medioni, Active contour models: Overview, implementation and applications. *IEEE Trans. Syst. Man Cybernetics* 1993, pp. 194–199.
4. I. Cohen, L. Cohen, and N. Ayache, Using deformable surfaces to segment 3-D images and infer differential structures. *Comput. Vision, Graphics, Image Process. Image Understand.* **56**(2), 1992, 242–263.
5. J. Hug, C. Brechbuhler, and G. Szekely, Tamed snake: A particle system for robust semi-automatic segmentation, in *Second International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'99)* (C. Taylor and A. Colchester, Eds.), No. 1679 in LNCS, pp. 106–115, Springer, Cambridge, 1999.
6. D. Metaxas and D. Terzopoulos, Constrained deformable superquadrics and nonrigid motion tracking, in *International Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pp. 337–343, Maui, HI, 1991.

7. R. Malladi, J. Sethian, and B. Vemuri, Shape modeling with front propagation: A level set approach. *IEEE Trans. Pattern. Anal. Mach. Intelligence* **17**(2), 1995, 158–174.
8. J. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Science*, Cambridge University Press, 1996.
9. V. Caselles, R. Kimmel, and G. Sapiro, Geodesic active contours. *Internat. J. Comput. Vision* **22**(1), 1997, 61–79.
10. N. Paragios and R. Deriche, A PDE-based level-set approach for detection and tracking of moving objects, in *International Conference on Computer Vision (ICCV'98)*, pp. 1139–1145, Bombay, India, 1998.
11. F. Leitner and P. Cinquin, Complex topology 3D objects Segmentation, in *SPIE Conference on Advances in Intelligent Robotics Systems*, Vol. 1609, Boston, 1991.
12. T. McInerney and D. Terzopoulos, Deformable models in medical image analysis: A survey. *Med. Image Anal.* **1**(2), 1996, 91–108.
13. J.-O. Lachaud and A. Montanvert, Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction. *Med. Image Anal.* **3**(2), 1999, 187–207.
14. M.-O. Berger and R. Mohr, Towards autonomy in active contour models, in *International Conference on Pattern Recognition (ICPR'90)*, Atlantic City, NJ, pp. 847–851, 1990.
15. S. Gunn and M. Nixon, A robust snake implementation: A dual active contour. *IEEE Trans. Pattern Anal. Mach. Intelligence* **19**(1), 1997, 63–68.
16. S. Lobregt and M. Viergever, A discrete dynamic contour model. *IEEE Trans. Med. Imaging* **14**(1), 1995, 12–23.
17. J. Boissonnat and B. Geiger, Three dimensional reconstruction of complex shapes based on the delaunay triangulation, in *SPIE Conference on Biomedical Image Processing and Biomedical Visualization* (R. Acharya and D. Goldgof, Eds.), Vol. 1905, San Jose, CA.
18. B. Kimia, A. Tannenbaum, and S. Zucker, On the evolution of curves via a function of curvature i. the classical case. *J. Math. Anal. Appl.* **163**, 1992, 438–458.
19. J. Ivins and J. Porritt, Active region models for segmenting textures and colours. *Image Vision Comput.* **13**(5), 1995, 431–438.
20. L. Cohen, On active contour models and balloons. *Comput. Vision Graphics Image Process. Image Understand.* **53**(2), 1991, 211–218.
21. G. Taubin, Curve and surface smoothing without shrinkage, in *ICCV95*, pp. 852–857, 1995.
22. H. Delingette, Intrinsic stabilizers of planar curves, in *3rd European Conference on Computer Vision (ECCV'94)*, Stockholm, Sweden, 1994.
23. M. Gage, On an area preserving evolution equation for plane curves. *J. Contemp. Math.* **51**, 1986, 51–62.
24. G. Huisken, The volume preserving mean curvature flow. *J. Reine Angew. Math.* **382**, 1987, 35–48.
25. J. Escher, U. Mayer, and G. Simonett, The surface diffusion flow for immersed hypersurfaces. *SIAM J. Math. Anal.* **29**(6), 1998, 1419–1433.
26. G. Picinbono, *Modèle géométrique de contour déformable implicite pour la segmentation et la simulation*, Master's thesis, université de Nice-Sophia Antipolis, 1997.
27. D. Chopp and J. Sethian, Motion by intrinsic laplacian of curvature. *Interfaces Free Boundaries* **1**, 1999, 1–18.
28. H.-K. Zhao, T. Chan, B. Merriman, and S. Osher, A variational level set approach to multiphase motion. *J. Comput. Phys.* **127**, 1996, 179–195.
29. G. Aubert and L. Blanc-Féraud, Some remarks on the equivalence between 2D and 3D classical snakes and geodesic active contours. *Internat. J. Comput. Vision* **34**(1), 1999, 5–17.