New Algorithms for Controlling Active Contours Shape and Topology

H. Delingette and J. Montagnat

Projet Epidaure
I.N.R.I.A.
06902 Sophia-Antipolis Cedex, BP 93, France

Abstract

In recent years, the field of active-contour based image segmentation have 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 represent 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 on parametric active contours in order to combine the advantages of these two contour representations. We introduce three key algorithms for independently controlling active contour parameterization, shape and topology. We compare our result to the level-set method and show similar results with a significant speed-up.*

1 Introduction

Image segmentation based on active contours has achieved considerable success in the past few years [15]. 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 [8], Kass *et al* choose to use a parametric contour representation with a semi-implicit integration scheme for discretizing the law of motion. Several authors have proposed different representations [16] including the use of finite element models [3], subdivision curves [6] and analytical models [17]. Implicit active contour representation were introduced in [13] following [19]. This approach has been developed by several other researchers including "geodesic snakes" introduced in [2].

The opposition between parametric and implicit contour representation corresponds to the opposition between Lagrangian and Eulerian frameworks. Qualifying the efficiency and the implementation issues of these two frameworks is difficult because of the large number of different algorithms existing in the literature. On one hand, implicit representations are in general regarded as being less efficient than parametric contours. This is because the update of an implicit contour requires the update of at least a narrow band around each contour. On the other hand, parametric contours cannot in general achieve any automatic topological changes, also several algorithms have been proposed to overcome this limitation [11, 14, 10].

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

1. **Discretization**. We propose two algorithms for controlling the relative vertex spacing 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.
2. **Shape**. We introduce an intrinsic internal force expressions that do 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.

We propose a framework where algorithms for controlling the discretization, shape and topology of active contours are completely independent of each other. Having algorithmic independence is important for two reasons. First, each modeling component may be optimized separately leading to computationally more efficient algorithms. Second, a large variety of active contour behaviors may be obtained by combining different algorithms for each modeling component.

2 Discretization of active contours

In the remainder, 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 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:

$$\frac{\partial^2 \mathcal{C}}{\partial t^2} = -\gamma \frac{\partial \mathcal{C}}{\partial t} + f_{\text{int}} + f_{\text{ext}} \tag{1}$$

where f_{int} and f_{ext} correspond respectively to 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 $\{\mathbf{p}_i^t\}$, $i = 0 \ldots N^t - 1$ represents the contour $\mathcal{C}(u,t)$ at time t. The discretization of equation 1 using centered and right differences for the acceleration and speed term leads to:

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

In order to simplify the notation, we will write $\mathbf{p}_i$ instead of $\mathbf{p}_i^t$ the vertex position at time t. At each vertex $\mathbf{p}_i$, we define a local tangent vector $\mathbf{t}_i$, normal vector $\mathbf{n}_i$, metric parameter ϵ_i and curvature k_i. We propose to define the tangent vector at $\mathbf{p}_i$, as the direction of the line joining its two neighbors: $\mathbf{t}_i = (\mathbf{p}_{i+1} -$

$\mathbf{p}_{i-1})/(2r_i)$ where $r_i = \|\mathbf{p}_{i+1} - \mathbf{p}_{i-1}\|/2$ is the half distance between the two neighbors of $\mathbf{p}_i$. 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 $k_i = \sin(\phi_i)/r_i$. 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: $\epsilon_i = \|\mathbf{F}_i - \mathbf{p}_{i+1}\|/(2r_i) = 1 - \|\mathbf{F}_{i+1} - \mathbf{p}_{i-1}\|/(2r_i)$. In another words, ϵ_i and $1 - \epsilon_i$ are the barycentric coordinates of $\mathbf{F}_i$ with respect to $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$: $\mathbf{F}_i = \epsilon_i \mathbf{p}_{i-1} + (1 - \epsilon_i)\mathbf{p}_{i+1}$.

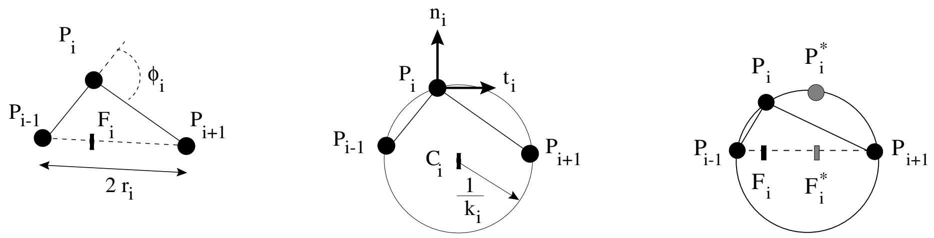

Fig. 1. Left: The geometry of a discrete contour; definition of $\mathbf{t}_i$, $\mathbf{n}_i$, k_i, ϕ_i, and $\mathbf{F}_i$. Right: The internal force associated with the curvature-conservative flow is proportional to $\mathbf{p}_i^\star - \mathbf{p}_i$.

Other definitions for the tangent and normal vectors could have been chosen. However, our tangent and normal vectors definitions has 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 $L(r_i, \phi_i, \epsilon_i) = \frac{r_i}{\tan \phi_i}(1 + \mu\sqrt{1 + 4\epsilon(1-\epsilon)\tan^2\phi})$ 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 sections 3.

3 Parameterization control

For a continuous active contour $\mathcal{C}(u,t)$, the contour parameterization is characterized by the metric function: $g(u,t) = \|\frac{\partial \mathcal{C}}{\partial u}\|$. 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 $g_i = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|$.

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. 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:

1. **uniform parameterization:** the spacing between consecutive vertices is uniform.
2. **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* [9] have proved 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. Therefore, if $\mathbf{t}$, $\mathbf{n}$ are the tangent and normal vector at a point $\mathcal{C}(u,t)$, then the contour evolution may be written as: $\frac{\partial \mathcal{C}}{\partial t} = f_{\text{int}} = a(u,t)\mathbf{t} + b(u,t)\mathbf{n}$. Kimia *et al* [9] 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 $b(u,t)$:

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

The tangential component of the internal force $a(u,t)\mathbf{t}$ constrains the nature of the parameterization. We propose to apply this principle on discrete parametric contours as well by decomposing the internal force f_{int} into its normal and tangential components: $(f_{\text{int}})_i = (f_{\text{tg}})_i + (f_{\text{nr}})_i$ with $(f_{\text{tg}})_i \cdot \mathbf{n}_i = 0$, and $(f_{\text{nr}})_i \cdot \mathbf{t}_i = 0$. 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: $(f_{\text{tg}})_i = (\epsilon_i^\star - \epsilon_i)(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}) = 2r_i(\epsilon_i^\star - \epsilon_i)\mathbf{t}_i$ where $\epsilon_i^\star$ is the reference metric parameter whose value depends on the type of parameterization to enforce.

3.1 Uniform vertex spacing

To obtain evenly spaced vertices, we simply choose: $\epsilon_i^\star = \frac{1}{2}$. This tangential force moves each vertex in the tangent direction towards the middle of its two neighbors. When the contour reaches its equilibrium, i.e. when $(f_{\text{tg}})_i = \mathbf{0}$, $\mathbf{p}_i$ is then equidistant from $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$. It equals to $(f_{\text{tg}})_i = \left(\frac{\partial^2 \mathcal{C}}{\partial u^2} \cdot \mathbf{t}\right)\mathbf{t} = \frac{\partial g}{\partial u}\mathbf{t}$. Because the second derivative vector $\frac{\partial^2 \mathcal{C}}{\partial u^2}$ is the first variation of the weak string internal energy ($\int_u \|\frac{\partial \mathcal{C}}{\partial u}\|^2 du$), this force is somewhat related to the classical "snakes" approach proposed in [8].

3.2 Curvature based vertex spacing

To obtain an optimal description of shape, it is required that vertices concentrate at parts of high curvature and that flat parts are only described with few vertices.

To obtain such 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: $K_i^{i+1} = (|k_i| + |k_{i+1}|)/2$. Then at each vertex $\mathbf{p}_i$, we can compute the local relative variation of absolute curvature $\Delta K_i \in [-1, 1]$ as: $\Delta K_i = (K_i^{i+1} - K_{i-1}^{i})/(K_i^{i+1} + K_{i-1}^{i})$. To enforce a curvature-based vertex spacing, we compute the reference metric parameter $\epsilon_i^\star$ as: $\epsilon_i^\star = \frac{1}{2} - 0.4 * \Delta K_i$.

When vertex $\mathbf{p}_i$ is surrounded by two edges having the same curvature then $\Delta K_i = 0$ and therefore $\epsilon_i^\star$ 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 $\epsilon_i^\star$ is close to 0.1 which implies that $\mathbf{p}_i$ moves towards $\mathbf{p}_{i+1}$.

3.3 Results of vertex spacing constraints

To illustrate the ability to decouple parameterization and shape properties, we propose to apply an internal force that modifies the vertex spacing on a contour without changing its shape. We define a *curvature conservative* regularizing force that moves $\mathbf{p}_i$ in the normal direction in order to keep the same local curvature:

$$f_{\mathrm{nr}} = (L(r_i, \phi_i, \epsilon_i^\star) - L(r_i, \phi_i, \epsilon_i))\mathbf{n}_i. \tag{5}$$

This equation has a simple geometric interpretation if we note that the total internal force $f_{\mathrm{int}} = f_{\mathrm{tg}} + f_{\mathrm{nr}}$ is simply equal to $\mathbf{p}_i^\star - \mathbf{p}_i$ where $\mathbf{p}_i^\star$ is the point having the same curvature as $\mathbf{p}_i$ but with a metric parameter $\epsilon_i^\star$. From right of figure 1, we can see that f_{tg} corresponds to the displacement between $\mathbf{F}_i^\star$ and $\mathbf{F}_i$ whereas f_{nr} corresponds to the difference of elevation between $\mathbf{p}_i^\star$ and $\mathbf{p}_i$.

Given an open or closed contour we iteratively apply differential equation 2 with the internal force expression described above. Figure 2 shows an example of vertex spacing constraint enforced on a closed contour consisting of 150 vertices. The initial vertex spacing is uneven. When applying the uniform vertex spacing tangential force ($\epsilon_i^\star = 0.5$), after 1000 iterations, all contour edge lengths become equal within less 5 percent without greatly changing the contour shape, as shown in figure 2 (upper row). The diagram displays the distribution of edge curvature as a function of edge length. 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 edge length is inversely proportional to edge curvature.

3.4 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. In order to add or remove vertices, we do not use any global contour reparameterization as performed in the level-set method [19] because of its high computational cost.

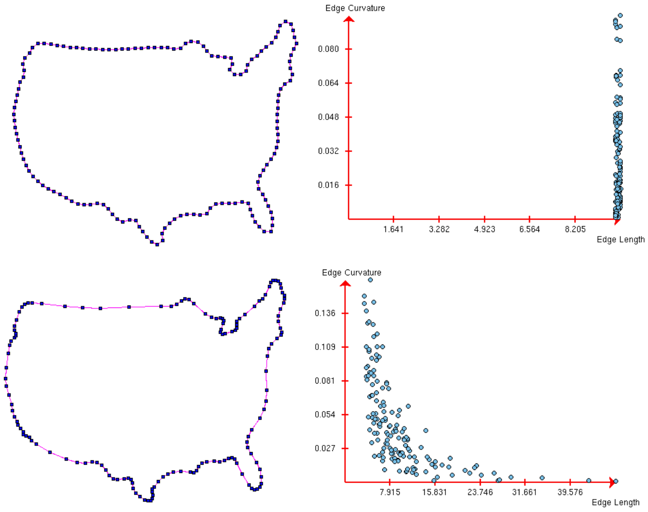

Fig. 2. (**up**) contour after applying the uniform vertex spacing tangential force; (**bottom**) contour after applying the curvature-based vertex spacing constraint.

Instead, we propose to locally add or remove a vertex if the edge length does not belong to a given distance range, similarly to [7, 12]. 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 $2 * s_{\min}$ 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 $fr_{\text{resolution}} = 5$ deformation iterations.

4 Shape regularization

The two internal forces defined in previous section have little influence on the contour shape evolution because they are only related 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 the mean curvature motion [13], Laplacian smoothing, thin rod smoothing or spring forces.

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.

To decrease the shrinking effect, Taubin [20] proposes to apply a linear filter to curves and surfaces in order to reduce the shrinking effect of Gaussian smoothing. However, these two methods only remove the shrinking effect for 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 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.

4.1 Curvature diffusion regularization

We propose to use the second derivative of curvature with respect to arc length as the governing regularizing force:

$$f_{\text{int}} = \frac{d^2 k}{ds^2}\mathbf{n} \tag{6}$$

This force tends to diffuse the curvature along the contour, thus converging towards circles, independently of their radii, for closed contours. For the discretization of equation 6, we do not use straightforward finite differences, since it would lead to complex and potentially unstable schemes. Instead, we propose a geometry-based implementation that is similar to equation 5:

$$f_{\text{nr}} = (L(r_i, \phi_i^\star, \epsilon_i^\star) - L(r_i, \phi_i, \epsilon_i))\mathbf{n}_i \tag{7}$$

where $\phi_i^\star$ is the angle at a point $\mathbf{p}_i^\star$ for which $\frac{d^2 k}{ds^2} = 0$. The geometric interpretation of equation 7 is also straightforward, since the internal force $f_{\text{int}} = f_{\text{tg}} + f_{\text{nr}}$ corresponds to the displacement $\mathbf{p}_i^\star - \mathbf{p}_i$. The angle $\phi_i^\star$ is simply computed by $\phi_i^\star = \arcsin(k_i^\star * r_i)$ where $k_i^\star$ is the curvature at $\mathbf{p}_i^\star$. Therefore, $k_i^\star$ is simply computed as the local average curvature weighted by arc length:

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

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 *scale* parameter, and $l_{i,i+j}$ is the distance between $\mathbf{p}_i$ and $\mathbf{p}_{i+j}$ then we compute k_i as :

$$k_i^\star = \frac{\sum_{j=1}^{\sigma_i} l_{i,i-j} k_{i+j} + l_{i,i+j} k_{i-j}}{\sum_{j=1}^{\sigma_i} l_{i,i-j} + l_{i,i+j}} \quad \text{with} \quad l_{i,i+j} = \sum_{k=1}^{j} \|\mathbf{p}_{i+k-1}\mathbf{p}_{i+k}\|.$$

This scheme generalizes the *intrinsic polynomial stabilizers* proposed in [5] 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 having a formal proof yet, that we obtain a stable iterative scheme if we choose $\alpha_i \leq 0.5$.

5 Topology constraints

Automatic topology changes of parametric contour has been previously proposed in [11, 14, 10]. In 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* [10] introduced topologically adaptive deformable surfaces where self-intersections are detected based on 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.

A contour topology is defined by the number of its connected components and whether each of its components is closed or opened. Our approach consists in using two basic topological operators. The first operator illustrated in figure 3 consists in merging two contour edges. Depending whether the edges belong to the same connected component or not, this operator creates or remove a connected component. The second topological operator consists in closing or opening a connected component.

Fig. 3. Topological operator applying on (**left**) two edges on the same connected component or (**right**) two different connected components.

Our approach for modifying a contour topology can be decomposed into three stages. The first stage 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 stage actually performs all topological modifications.

5.1 Data structure for the detection of contour intersections

Finding pairs of intersecting edges has an *a priori* complexity of $O(n^2)$ 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 [14], 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 which size d is user-defined. The influence of the grid size d is discussed in section 6.1. 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 $\mathbf{o}_{\text{grid}}$ 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 order to build the sampled contour, we scan all edges of each connected components. 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. 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.

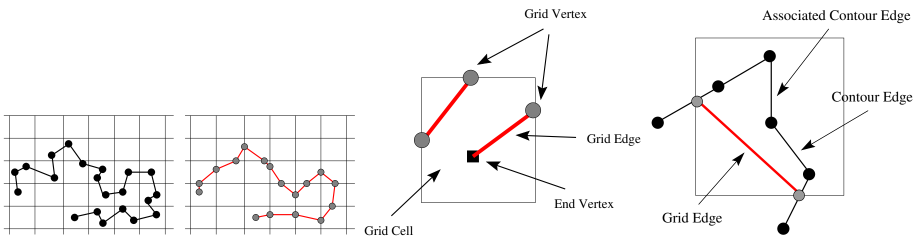

Fig. 4. (**left**) The original contour with the regular grid the contour decomposed on the regular grid; (**right**) Definition of grid vertex, grid edge, grid cell and contour edge associated with a grid edge.

5.2 Finding intersecting grid edges

In order to optimize memory space, we store all non-empty grid cells inside 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 figure 4, left). Since each grid edge is geometrically represented by a line segment, this intersection test only requires the evaluation of 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 5.3). 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 figure 4, right.

Our contour edges intersection algorithm has the following properties: (i) if one pair of grid edges intersects then there is at least one pair of contour edges that intersects inside this grid cell; and (ii) if a pair of contour edges intersects and if the corresponding intersecting area is greater than $d * d$ 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 intersection having an area greater than $d * d$. In practice, since the grid origin $\mathbf{o}_{\text{grid}}$ 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.

5.3 Applying topological operators

All pairs of intersecting contour edges are stored inside another hash table for an 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, we perform the following tasks. If both edges belong to the same connected component, then the they are merged if their topological distance is greater than a threshold (usually equal to 8). This is to avoid creating too small connected components. In all other cases, the two edges are merged with the topological operator presented in figure 3. 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.

5.4 Other applications of the collision detection algorithm

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. As in [14] our framework can prevent the merging of two distinct connected components while allowing the removal of self-intersections. To do so, when a pair of intersecting contour edges belonging to distinct connected components is found, instead of merging this edges, we align all vertices located between intersecting edges belonging to the same connected component (see figure 5). Thus, each component pushes back all neighboring components. In figure 5, right, we show an example of image segmentation where this repulsive behavior between components is very useful in segmenting the two heart atriums.

6 Results

6.1 Topology algorithm cost

We evaluate the performance of our automatic topology adaptation algorithm on the example of figure 6. The contour consisting of 50 vertices, is deformed from a circular shape towards a vertebra in a CT image. The computation time for building the data structure described in section 5.1 is displayed in figure 6,

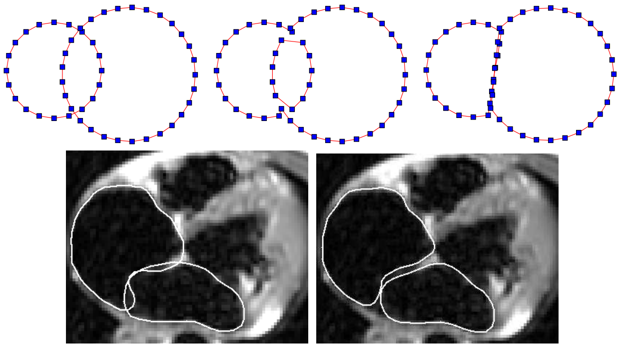

Fig. 5. (from left to right) Two intersecting connected components; after merging the two pairs of intersecting edges; after aligning vertices along the intersecting edges. Example of 2 active contours reconstructing the right and left ventricles in a MR image with a repulsive behavior between each components.

right, as a function of the grid size d. It varies from 175 ms to 1 ms when the grid size increases from 0.17 to 10 image pixels on a Digital PWS 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).

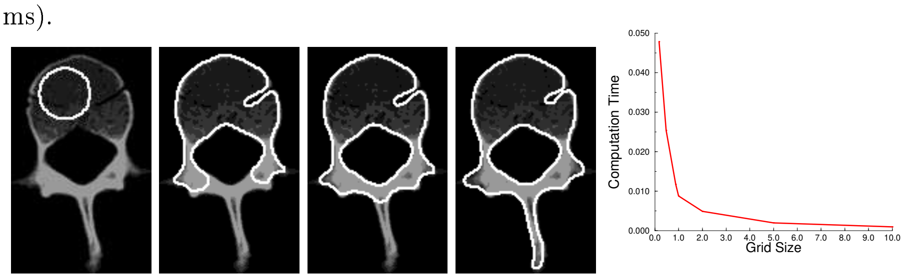

Fig. 6. Segmentation of a vertebra in a CT image; Topology algorithm computation time.

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 percent.

6.2 Segmentation example

This example illustrates the segmentation of an aortic arch angiography. Figure 7 shows the initial contour (up left) and its evolution towards the aorta and the main vessels. External forces are computed as a function of vertex distance to a gradient point to avoid oscillations around image edges and are projected on the vertex normal direction. The contour is regularized by a curvature diffusive constraint. 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 and the contour is opened as it reaches the image border.

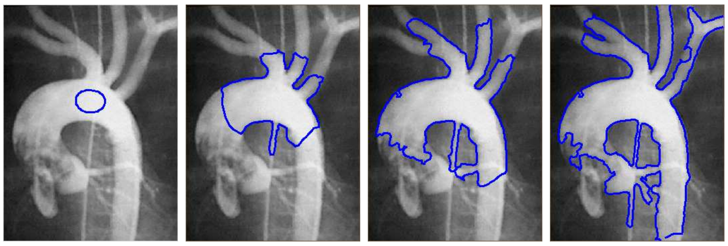

Fig. 7. Evolution of a closed curve towards the aortic arch and the branching vessels.

7 Comparison with the level-set method

The main advantage of the level-set method is obviously its ability to automatically change the contour topology during the deformation. This property makes it well-suited for reconstructing contours of complex geometries 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. However, the major drawbacks of level-sets methods are related to their difficult user interaction and their computational cost, although some speed-up algorithms based on constraining the contour evolution through the Fast-Marching method [19] or by using an asynchronous update of the narrow-band [18] have been proposed. The formal comparison between both parametric and level-set approaches have been recently established in the case of geodesic snakes [1]. In this section, we propose a practical comparison between both approaches including implementation issues.

7.1 Level-set implementation

The level-set function Ψ is discretized on a rectangular grid whose resolution corresponds to the image pixel size. The evolution equation is discretized in space using finite differences in time using an explicit scheme, leading to : $\Psi_{ij}^{t+\Delta t} = \Psi_{ij}^{t} + \Delta t \nu_{ij} \|\nabla_{ij} \Psi_{ij}^{t}\|$ [13] where ν_{ij} denotes the propagation speed term and Δt is the discrete time step.

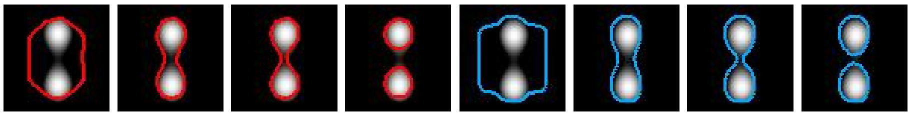

Fig. 8. (Five left figures) Discrete contour deformation; (Five right figures) level-set deformation.

The propagation speed term ν is designed to attract $\mathcal{C}$ towards object boundaries extracted from the image using a gradient operator with an additional regularizing term: $\nu(\mathbf{p}) = \beta(\mathbf{p})\,(\kappa(\mathbf{p}) + c)$. $\kappa(\mathbf{p})$ denotes the contour curvature at point $\mathbf{p}$ while c is a constant resulting in a balloon force [4] on the contour. Finally, $\beta(\mathbf{p}) \in [0, 1]$ is a multiplicative coefficient dependent on the image gradient norm at point $\mathbf{p}$. When $\mathcal{C}$ moves across pixels of high gradients, this term slows down the level-set propagation. A threshold parameter determines the minimal image boundary strength required to stop a level-set evolution. We speed-up the level-set by using a narrow band method [13] which requires to periodically reinitialize the level-set contour.

In order to compare active contours to level-sets, we compute external forces similar to level-set propagation at discrete contour vertices. First, we use mean curvature motion as the governing internal force and we have implemented for the external force, a balloon force weighted by the coefficient β proposed above. Finally, we were able to use the same gradient threshold in both approaches.

7.2 Torus example

We first propose to compare both approaches on the synthetic image shown in figure 8. This image has two distinct connected components.

A discrete contour is initialized around the two components. 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. A corresponding level-set is initialized at the same place that the discrete contour. 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.

Figure 8 shows the convergence of the discrete contour (left) and the level-set (right). The discrete contour converges in 0.42 seconds opposed to 3.30 seconds for the level-set, that is a 7.85 acceleration factor in favor of the discrete contour. 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.

7.3 Synthetic data

This experiments shows the ability of the discrete contour topology algorithm to follow difficult topology changes. We use a synthetic fractal image showing a

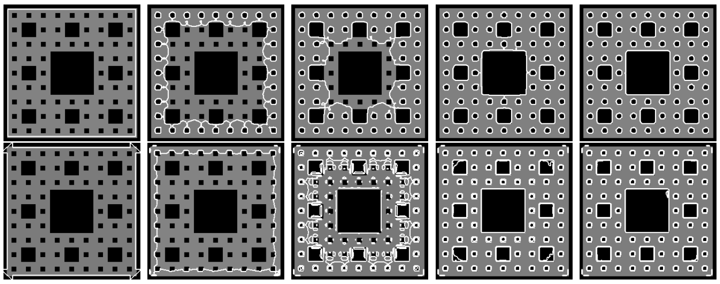

Fig. 9. (upper row) Discrete contour convergence in a fractal image; (bottom row) Level-set convergence in the same image.

number of small connected components. Figure 9, upper row, shows the discrete contour convergence in the image while the bottom row 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 iterations 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. 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 figure 9. As the level-set contours gradually fills-in the whole image, we have verified that the convergence time is not minimized by using any narrow bands. Again, the speed-up is 3.84 in favor of the discrete contour.

8 Conclusion

We have introduced three algorithms that greatly improve the generality of parametric active contours while preserving their computational efficiency. Furthermore, these algorithms are controlled by simple parameters that are easy to understand. For the internal force, a single parameter α between 0 and 1 is used to set the amount of smoothing. 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 set automatically to meaningful values providing good results in most cases.

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

References

1. G. Aubert and L. Blanc-Féraud. Some Remarks on the Equivalence between 2D and 3D Classical Snakes and Geodesic Active Contours. *International Journal of*

Computer Vision, 34(1):5–17, September 1999.
2. V. Caselles, R. Kimmel, and G. Sapiro. Geodesic Active Contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.
3. I. Cohen, L.D. Cohen, and N. Ayache. Using Deformable Surfaces to Segment 3-D Images and Infer Differential Structures. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 56(2):242–263, September 1992.
4. L.D. Cohen. On Active Contour Models and Balloons. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2):211–218, March 1991.
5. H. Delingette. Intrinsic stabilizers of planar curves. In *third European Conference on Computer Vision (ECCV'94)*, Stockholm, Sweden, June 1994.
6. J. Hug, C. Brechbüler, and G. Székely. Tamed Snake: A Particle System for Robust Semi-automatic Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI'99)*, volume 1679 of *Lectures Notes in Computer Science*, pages 106–115, Cambridge, UK, September 1999. Springer.
7. J. Ivins and J. Porrill. Active Region models for segmenting textures and colours. *Image and Vision Computing*, 13(5):431–438, June 1995.
8. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1:321–331, 1988.
9. B. Kimia, A. Tannenbaum, and S. Zucker. On the evolution of curves via a function of curvature i. the classical case. *Journal of Mathematical Analysis and Applications*, 163:438–458, 1992.
10. J.-O. Lachaud and A. Montanvert. Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.
11. F. Leitner and P. Cinquin. Complex topology 3d objects segmentation. In *SPIE Conf. on Advances in Intelligent Robotics Systems*, volume 1609, Boston, November 1991.
12. S. Lobregt and M. Viergever. A Discrete Dynamic Contour Model. *IEEE Transactions on Medical Imaging*, 14(1):12–23, 1995.
13. R. Malladi, J.A. Sethian, and B.C. Vemuri. Shape Modeling with Front Propagation : A Level Set Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–174, 1995.
14. T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In *International Conference on Computer Vision (ICCV'95)*, pages 840–845, Cambridge, USA, June 1995.
15. T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.
16. S. Menet, P. Saint-Marc, and G. Medioni. Active Contour Models: Overview, Implementation and Applications. *IEEE Trans. on Systems, Man and Cybernetics*, pages 194–199, 1993.
17. D. Metaxas and D. Terzopoulos. Constrained Deformable Superquadrics and non-rigid Motion Tracking. In *International Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 337–343, Maui, Hawai, June 1991.
18. 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)*, pages 1139–1145, Bombay, India, 1998.
19. J.A. Sethian. *Level Set Methods : Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Science*. Cambridge University Press, 1996.
20. G. Taubin. Curve and Surface Smoothing Without Shrinkage. In *International Conference on Computer Vision (ICCV'95)*, pages 852–857, 1995.