EUROGRAPHICS 2003 / P. Brunet and D. Fellner
(Guest Editors)

Volume 22 (2003), Number 3

Sub-Voxel Topology Control for Level-Set Surfaces

Stephan Bischoff and Leif Kobbelt

Abstract

Active contour models are an efficient, accurate, and robust tool for the segmentation of 2D and 3D image data. In particular, geometric deformable models (GDM) that represent an active contour as the level set of an implicit function have proven to be very effective. GDMs, however, do not provide any topology control, i.e. contours may merge or split arbitrarily and hence change the genus of the reconstructed surface. This behavior is inadequate in settings like the segmentation of organic tissue or other objects whose genus is known beforehand. In this paper we describe a novel method to overcome this limitation while still preserving the favorable properties of the GDM setup. We achieve this by adding (sparse) topological information to the volume representation at locations where it is necessary to locally resolve topological ambiguities. Since the sparse topology information is attached to the edges of the voxel grid, we can reconstruct the interfaces where the deformable surface touches itself at sub-voxel accuracy. We also demonstrate the efficiency and robustness of our method.

1. Introduction

Deformable models were originally devised for feature detection in two dimensional images. In particular in medical imaging applications, like the segmentation of organic structures from MRT images, these models are ubiquitous. In recent years however, deformable models have also widely spread into such diverse areas as physical simulation, geometric modeling, 3D reconstruction, computer vision, etc. and can now be considered as a standard tool for computer graphics applications.

Deformable models come in two flavors: *parametric* (or *explicit*) and *geometric* (or *implicit*) models, depending on whether the contour is represented as the range or the kernel of a function.

Parametric models represent the contour by an explicit spline or polygonal mesh. As is usual in parametric representations, the sampling rate has to be adapted to the contour geometry, i.e. in regions of high curvature, the contour has to be represented by more patches than in flat areas. This resampling has to be done over and over, which is a tedious and error-prone process, in particular in three dimensions. Furthermore, as collision detection is costly, it is hard to avoid self-intersections of the contour. On the other hand, parametric representations automatically guarantee that the contour does not merge or split during evolution, i.e. the contour will neither remove nor create handles or cavities.

Geometric models on the other hand, represent the contour as the level set of a scalar-valued function. Geometric models are free of parameterization artifacts and will never self-intersect. Furthermore they can be efficiently implemented using e.g. narrow band or fast marching techniques. On the other hand, however, geometric models provide no way to control the topology of the contour, i.e. in general the contour may merge or split.

In many applications, the deformable model has to reflect some a priory knowledge about the topology of the object. For example, when reconstructing a cortical surface from a set of MRT scans, the model should reflect the anatomical fact, that this surface is homeomorphic to a sphere. Hence, the contour has to be prevented from forming handles or cavities. Also, when two or more objects in an image are segmented simultaneously, the contours should obviously not be allowed to merge.

In this paper we devise an algorithm that combines the topology preserving properties of parametric models with the parameterization free, implicit representation of geometric models. The topology control is achieved by placing "cuts" on the edges of the voxel grid whenever the model is about to change its topology. As the cuts can subdivide the edge in an arbitrary ratio, this results in a sub-voxel accurate reconstruction of the contour. We further note that our algorithm does affect the evolution of the underlying geometric model only along the contact surface, hence it can be implemented as an "add-on" to existing level-set implementations.

For simplicity, we will build our exposition on the easily-

understandable fast marching method. However, the algorithm can also be applied (with minor modifications) to other geometric models.

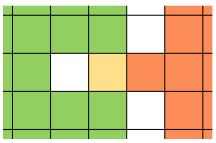
After an overview of previous work in Section 2, we will give a short introduction to deformable models and the fast marching method in Section 3. A detailed description of topological framework is presented in Section 4. In Section 5 we show some results and finally conclude with future work in Section 6.

2. Previous Work

Active contour models were first introduced in 1987 by Kass et al. [9]. Their explicit model employed a parametric curve whose evolution is driven by an energy-minimization process. Since then numerous extensions and refinements have been made for explicit models [12, 5, 10, 3, 2]. Explicit models are mostly topology preserving, i.e. the contour does not merge or split during its evolution. For this reason they are in particular favored in medical imaging applications like the segmentation of brain tissue, where the topology of the segmented object is known beforehand [4]. It is also possible to add topology control to explicit models [12], however, this becomes very elaborate in three dimensions.

Implicit models as introduced by Sethian and Osher [14] in 1988 represent the contour as the level set of a scalar-valued function. Since then level set methods have been applied in various areas, like image segmentation [11], shape morphing [6], physical simulations [7], 3D reconstruction [18] and geometric modeling [13]. For a thorough overview we refer to the books [17, 15]. Level set methods are in particular favored for their ease of implementation and their lack of parameterization artifacts.

Implicit methods do not suffer from the (re-)parameterization problems encountered with explicit methods (in particular in three dimensions), however, they do not allow to control the topology of the contour, i.e. they cannot prevent the contour from merging or splitting. Han, Xu and Prince [8] overcome this problem by modifying the update rule for the level set function to respect the topology. The value of a voxel that is rejected for topological reasons is constrained to a constant $\pm\varepsilon$. However, this "wrong" value not only changes the computation of the level set function, but may also prevent parts of the object from being conquered:



Consider the configuration on the left. The green voxels have already been conquered, the yellow voxel, however, cannot be conquered without topology change, hence the red part is effectively cut off from the level set evolution. A second drawback of their method is, that it is not sub-voxel accurate along the collision lines of two different parts of the contour, hence aliasing-artefacts can occur. In our case we use linear approximation and therefore achieve better accuracy.

3. Preliminaries

3.1. Deformable Models

Deformable models are typically used to segment objects from their background in 2D images and 3D volume data. For this the user places an initial (approximating) surface $S \subset \mathbb{R}^3$ into the volume. Then S is continuously deformed such as to lock on the boundaries of the desired object.

The idea of deformable models is to track the evolution of the surface S over time, i.e. $S = S(t)$. This process can conveniently be described by a partial differential equation

$$D_{\mathbf{n}} S = f$$

where $D_{\mathbf{n}}$ is the derivative in direction of the outward surface normal and f is a user-defined force.

The force f may depend on various parameters, like time, position in space, curvature of the surface etc. One typically chooses f such that $S(t)$ strives to minimize an energy functional

$$\mathcal{E}(S) = \mathcal{E}_{\text{int}}(S) + \mathcal{E}_{\text{ext}}(S)$$

as t goes to infinity. Here, $\mathcal{E}_{\text{int}}$ represents the internal energy of S. It is typically a weighted combination of stretching and bending energies and used to smooth and regularize the shape of S. The external energy $\mathcal{E}_{\text{ext}}$ is derived from the underlying segmentation problem. The corresponding external forces are typically gradient forces that attract the surface S into the direction of the image features that represent the object's boundary.

Although our results are valid for arbitrary forces f, we will assume for the ease of presentation that f is strictly positive, i.e. S moves only outward. This allows us to describe our method within the setup of the efficient and easily-understandable fast marching method (see below). Other than that we make no assumptions about f, hence f can be considered as a black-box that is provided by the user and that captures all the relevant internal and external forces.

3.2. The Fast Marching Method

Let us consider a surface $S(t) \subset \mathbb{R}^3$ that evolves over time t, starting at an initial configuration $S(0)$. Let us further denote by $\eta(\mathbf{x})$ the time when $S(t)$ passes a point $\mathbf{x} \in \mathbb{R}^3$, see Figure 1. Note that as $S(t)$ moves only outward it cannot pass a point $\mathbf{x}$ twice, hence $\eta(\mathbf{x})$ is well-defined. By this definition, $S(t)$ is just the level set of $\eta(\mathbf{x})$ at time t, i.e.

$$S(t) = \{\mathbf{x} : \eta(\mathbf{x}) = t\}$$

The fast marching method [16, 17, 15] is an algorithm that reconstructs an approximation of $\eta(\mathbf{x})$ not on all of $\mathbb{R}^3$ but only on the discrete Cartesian grid $\mathbb{Z}^3$. To be more precise, the algorithm computes for each grid point $\mathbf{x} \in \mathbb{Z}^3$ a value $\eta_{\mathbf{x}}$ that approximates the real $\eta(\mathbf{x})$. In order to speed up the algorithm, the values $\eta_{\mathbf{x}}$ are computed along a front of grid

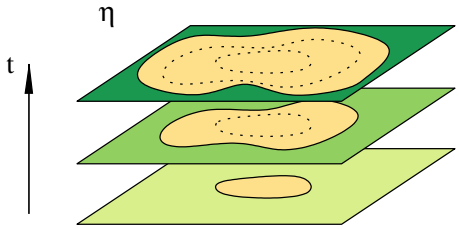


Figure 1: *Level set representation: The arrival time of the surface S at point $\mathbf{x}$ is designated as $\eta(\mathbf{x})$. Hence $S(t)$ is the level set $\{\mathbf{x} : \eta(\mathbf{x}) = t\}$ of η at time t.*

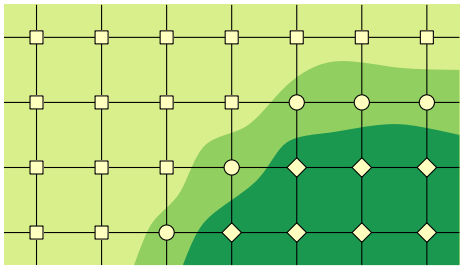


Figure 2: *The fast marching methods assigns to each grid point one of the three states* conquered ⋄, front ∘, free □. *Note that each front point is connected to at least one conquered point.*

points that is propagated over the entire volume until finally all values $\eta_{\mathbf{x}}$ have been computed.

To each grid point $\mathbf{x} \in \mathbb{Z}^3$ one of the three states $\{$ *free, front, conquered* $\}$ is assigned.

- For *conquered* points $\mathbf{x}$ the algorithm has already computed the final value $\eta_{\mathbf{x}}$.
- *Front* points $\mathbf{x}$ are connected to at least one conquered point. The value $\eta_{\mathbf{x}}$ of a front point is only tentative and may be updated during the run of the algorithm.
- Points $\mathbf{x}$ that are neither conquered nor front are called *free*, $\eta_{\mathbf{x}} = \infty$.

Figure 2 depicts a typical configuration of grid point states in the two-dimensional case.

The fast marching method proceeds by successively conquering front points. Hence the front advances over the grid until all grid points $\mathbf{x}$ are conquered and assigned a value $\eta_{\mathbf{x}} < \infty$. The pseudo-code in Figure 3 illustrates the basic structure of the algorithm.

Note that in the initialization step **1** the exact distance of a grid point to the initial front $S(0)$ has only to be evaluated for the front grid points. Step **3** is efficiently implemented using a min-heap data structure. The updating of the neighbors arrival times in step **5** is done using a first order approximation. Details on the implementation of steps **1**-**5** can be found in [16, 17, 15].

Fast Marching Method

1 Initialization:

$$C^0 = \{\mathbf{x} : \mathrm{dist}(\mathbf{x}, S(0)) < 0\}$$
$$F^0 = \{\mathbf{x} : \mathbf{x} \text{ is a face neighbor to } C\}$$
$$\eta_{\mathbf{x}}^0 = \begin{cases} 0 & \text{if } \mathbf{x} \in C^0 \\ \mathrm{dist}(\mathbf{x}, S) & \text{if } \mathbf{x} \in F^0 \\ \infty & \text{otherwise} \end{cases}$$

2 For $k = 0, 1, 2, 3, \ldots$ do

3 Select the grid point $\mathbf{x}^k \in F^k$ with smallest arrival time $\eta_{\mathbf{x}}^k$

4 Let

$$N = \{\mathbf{y} : \mathbf{y} \text{ is a free neighbor of } \mathbf{x}\}$$

and set

$$C^{k+1} \leftarrow C^k \cup \{\mathbf{x}^k\}$$
$$F^{k+1} \leftarrow F^k \setminus \{\mathbf{x}^k\} \cup N$$

5 Update the values $\eta_{\mathbf{y}}$ of the neighbors $\mathbf{y} \in N$ according to the forces $f(\mathbf{y})$.

Figure 3: *Fast Marching Method*

4. Algorithm

4.1. Cut-edge Grids

Let us consider a set C of conquered grid points. A grid point $\mathbf{v}$ is said to be simple, if C and $C \cup \{\mathbf{v}\}$ have the same topology, otherwise it is called complex. The precise meaning of the term "same topology" will be explained in the following sections, but for now we will stick to the intuitive meaning that the number of components, cavities and handles are the same for C and $C \cup \{\mathbf{v}\}$. During a run of the fast marching algorithm, grid points are successively conquered resulting in a sequence

$$C^0 \to C^1 \to \cdots \to C^n$$

where $C^{k+1} = C^k \cup \{\mathbf{x}^k\}$. If all grid points $\mathbf{x}^k$ are simple, the topology of C^n will be the same as that of C^0. In general, however, grid points $\mathbf{x}^k$ cannot be guaranteed to be simple. Adding the a grid point could connect two parts of an object, therefore forming a handle or a hole. The original fast marching method approach does not prevent complex grid points from being conquered. Hence the surface $S(t)$ may change its topology during the run of the algorithm.

In the following we will present a framework, that allows to solve the aforementioned problems on a sub-voxel scale without modifying the underlying level set computation. To achieve sub-voxel accuracy we propose to insert information in between the integer grid points, i.e. on the edges of the grid. We do this by placing "cuts" on the edges in order to separate the two adjacent grid points from each other. In our example, placing cuts on the edges to the left and to the top of the center voxel (see Figure 9,c,e) will conveniently keep the two components separated.

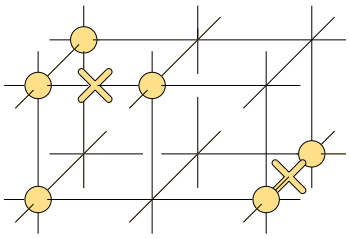


Figure 4: *A cut-edge grid (C,X) consists of a set $C \subset \mathbb{Z}^3$ of integer grid points $\bullet$ and a set $X \subset C \times C$ of cuts $\times$. Note that cuts may only be placed in between two grid points that belong to C* and *are adjacent.*

To be more precise, we define a *cut-edge grid* to be a tuple (C,X) where $C \subset \mathbb{Z}^3$ is a set of integer grid points and

$$X \subset \{(\mathbf{c},\mathbf{d}) \in C \times C : \mathbf{c} \text{ is adjacent to } \mathbf{d}\}$$

are the cut edges. A cut-edge grid can conveniently be implemented by assigning to each edge a "cut"-flag, see Figure 4.

Whenever the fast marching algorithm is about to conquer a grid point $\mathbf{x}$, we first check the grid point's topological status. If it is a simple point, the algorithm proceeds as usual. However, if it is not a simple point, we will place some cuts around $\mathbf{x}$ such as to maintain the topology of C.

In the following sections we describe this framework in more detail. First, we will put the rather vague notion of a cut-edge grid on solid mathematical ground by mapping our cut-edge grids onto so-called digital sets. In the second part, we will describe, how to resolve topological inconsistencies, i.e. which edges to cut in order to avoid topology changes. Thirdly, we will describe the geometric positions of the cuts, i.e. we will determine the ratio by which an edge is subdivided by the cut. Finally we describe how to extract an explicit representation, i.e. a polygonal mesh from a cut-edge grid.

4.2. Topological Embedding

In this section we will use the concepts of digital topology to assign a precise topological meaning to our cut-edge grids. This is needed to properly detect topology changes as well as to avoid inconsistencies and ambiguities when extracting an explicit surface representation from a cut-edge grid, see Figure 5. We do this by mapping each cut-edge grid (C,X) onto a digital set $\mathcal{E}(C,X) \subset \mathbb{Z}^3$. Then we define the topology of (C,X) to be the one of $\mathcal{E}(C,X)$. This approach allows us to transfer the well-known characterization theorems of simple points from digital topology into our framework.

Remark: In the following we will deal with two kind of integer vectors $\in \mathbb{Z}^3$. Elements $\mathbf{c} \in C \subset \mathbb{Z}^3$ are called grid points and are thought of as infinitely small points. Elements $\mathbf{v} \in V \subset \mathbb{Z}^3$ of a digital set are called voxels and are thought of as unit cubes centered at $\mathbf{v}$.

Let us first recap some definitions from digital topology. We

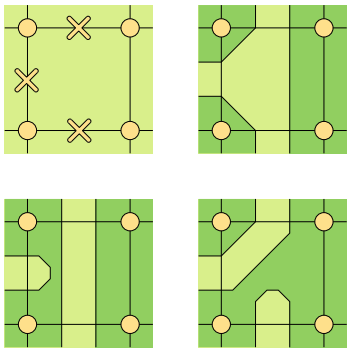


Figure 5: *Different topological interpretations of a cut-edge grid.*

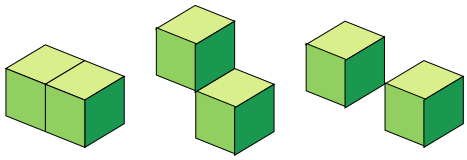


Figure 6: *Neighborhood relations: 6,18,26-neighborhood (from left to right)*

say that a voxel $\mathbf{w} \in \mathbb{Z}^3$ is an $6,18,26$-neighbor of a voxel $\mathbf{v} \in \mathbb{Z}^3$ if $\|\mathbf{v}-\mathbf{w}\|_2 \leq 1, \sqrt{2}, \sqrt{3}$ resp. see Figure 6. A digital set V is a subset of $\mathbb{Z}^3$, the so called interior voxels. The complement $\overline{V} = \mathbb{Z}^3 \setminus V$ is called exterior. To avoid a topological connectivity paradox, we assume that the interior voxels are 6-connected and that the exterior voxels are 26-connected.

Let $\mathbf{v} \in \mathbb{Z}^3$ be a voxel and let $V \subset \mathbb{Z}^3$ be a voxel set. We say that $\mathbf{v}$ is simple with respect to V, if V and $V \cup \{\mathbf{v}\}$ have the same number of components, handles and cavities. Otherwise $\mathbf{v}$ is called a complex voxel. In the following we will give a characterization of simple voxels in terms of connected components of a neighborhood of $\mathbf{v}$.

Let us denote by

$$N_n(\mathbf{v}) = \{\mathbf{w} : \mathbf{w} \text{ is an } n\text{-neighbor of } \mathbf{v}\}$$

the n-neighborhood ($n \in \{6,18,26\}$) of $\mathbf{v}$ and set

$$N_n^*(\mathbf{x}) = N_n(\mathbf{x}) \setminus \{\mathbf{x}\}.$$

Furthermore we define the geodesic n-neighborhood of order k of a voxel $\mathbf{v}$ with respect to a voxel set $V \subset \mathbb{Z}^3$ recursively by

$$\begin{aligned} N_n^1(\mathbf{v},V) &= V \cap N_n(\mathbf{v}) \\ N_n^{k+1}(\mathbf{v},V) &= V \cap \bigcup \left\{ N_n(\mathbf{w}) \,:\, \mathbf{w} \in N_n^k(\mathbf{v},V) \right\} \end{aligned}$$

The number of n-connected components of a voxel set $V \subset \mathbb{Z}^3$ is denoted by $c_n(V)$. Furthermore, we define the *topological numbers* n_{int} and n_{ext} as follows

$$\begin{aligned} n_{\text{ext}}(\mathbf{v},V) &= c_{26}\left(\overline{V} \cap N_{26}^*(\mathbf{v})\right) \\ n_{\text{int}}(\mathbf{v},V) &= c_6\left(N_6^2(\mathbf{v},V) \cap N_{26}^*(\mathbf{v})\right) \end{aligned}$$

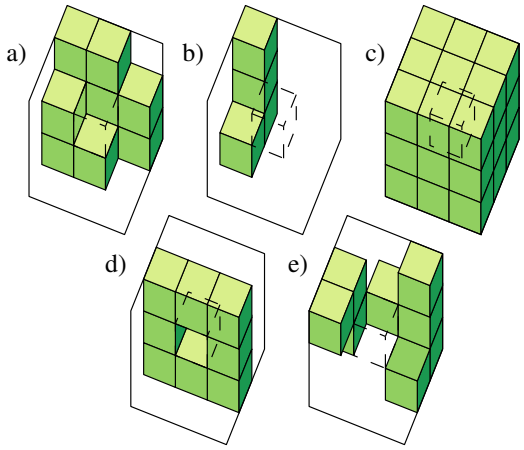


Figure 7: *Characterization of the center voxel (dashed) with respect to its neighborhood (solid). Simple: a)* $n_{int} = 1, n_{ext} = 1$. *Complex: b)* $n_{int} = 0, n_{ext} = 1$, *c)* $n_{int} = 1, n_{ext} = 0$, *d)* $n_{int} = 1, n_{ext} = 2$, *e)* $n_{int} = 2, n_{ext} = 1$.

Roughly stated, $n_{int}(\mathbf{v}, V)$ designates the number of interior components of V that touch $\mathbf{v}$ and $n_{ext}(\mathbf{v}, V)$ the number of exterior components that touch $\mathbf{v}$. Now we are able to formulate the characterization theorem.

Characterization Theorem Let $V \subset \mathbb{Z}^3$ be a digital set. A voxel $\mathbf{v}$ is simple with respect to V if and only if

$$n_{int}(\mathbf{v}, V) = n_{ext}(\mathbf{v}, V) = 1$$

A proof of this theorem can be found in [1]. Figure 7 depicts the characterization theorem.

In the following we describe how to embed a cut-edge grid (C, X) as a voxel set, such as to form a digital set $\mathcal{E}(C, X)$. Intuitively we do that by removing from the solid volume $\mathbb{Z}^3$ voxels that correspond to either non-grid points $\mathbb{Z}^3 \setminus C$ or to cuts X. For this we think of $\mathbb{Z}^3$ at double resolution, hence each grid point $\mathbf{v}$ corresponds to an even-valued voxel $2\mathbf{v}$. Note that we do not actually implement a double resolution grid — we just use it as a convenient concept to resolve the topology of a cut-edge grid.

We first define an auxiliary mapping

$$\phi : \mathbb{Z}^3 \cup \left(\mathbb{Z}^3 \times \mathbb{Z}^3 \right) \to \mathcal{P}(\mathbb{Z}^3)$$

that maps grid points and cut edges to voxels sets as follows

$$\begin{aligned} \phi(\mathbf{v}) &= N_{26}(2\mathbf{v}) \\ \phi(\mathbf{v}, \mathbf{w}) &= N_{26}(\mathbf{v} + \mathbf{w}) \cap \\ &\quad \left\{ \mathbf{u} \in \mathbb{Z}^3 \ : \ \|\mathbf{u} - 2\mathbf{v}\|_2 = \|\mathbf{u} - 2\mathbf{w}\|_2 \right\} \end{aligned}$$

(Here $+$ and $-$ mean addition and subtraction of coordinates in $\mathbb{Z}^3$). See Figure 8 for a depiction of this mapping.

A cut edge configuration (C, X) is then embedded as a digital

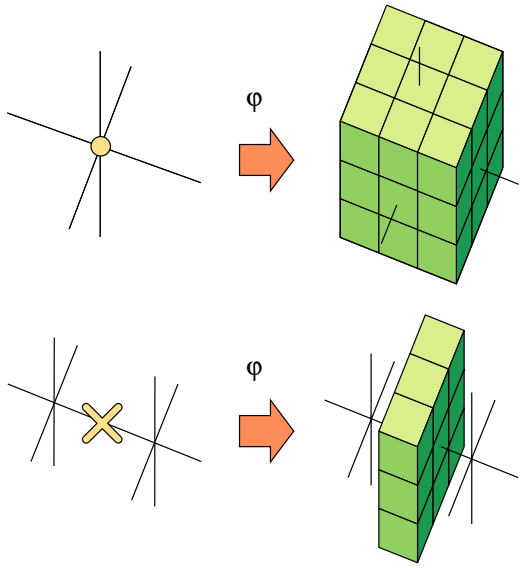


Figure 8: *The auxiliary mapping* ϕ *maps grid points and cut-edges to voxel sets* $\subset \mathbb{Z}^3$.

set $\mathcal{E}(C, X) \subset \mathbb{Z}^3$ by

$$\mathcal{E}(C, X) = \mathbb{Z}^3 \setminus \bigcup_{\mathbf{c} \notin C} \phi(\mathbf{c}) \setminus \bigcup_{(\mathbf{v}, \mathbf{w}) \in X} \phi(\mathbf{v}, \mathbf{w})$$

see Figure 9.

We can now define the topological equivalence of two cut-edge configurations (C, X) and (C', X').

Definition A Two cut-edge grids (C, X) and (C', X') are topologically equivalent, if there exists a sequence of digital sets

$$\mathcal{E}(C, X) = V_0 \to \cdots \to V_n = \mathcal{E}(C', X')$$

such that V_i and V_{i+1} differ only by a simple voxel.

Direct implementation of definition A is, albeit possible, cumbersome and inefficient as it amounts to testing each sequence $V_0 \to \cdots \to V_n$ for complex voxels. The following criterion facilitates this task.

Cut-edge Criterion Let two cut-edge grids (C, X) and (C', X') be given such that

$$\begin{aligned} C' &= C \cup \{\mathbf{v}\} \quad \text{and} \\ X' &= X \cup Y \end{aligned}$$

where $Y \subset (\mathbf{v}, N_6(\mathbf{v}))$. Then (C, X) and (C', X') are topologically equivalent, if and only if $2\mathbf{v}$ is a simple voxel with respect to $\mathcal{E}(C', X')$.

The proof of this intuitive criterion amounts to finding a sequence as required by Definition A and is not carried out here. Figure 9 depicts the application of this criterion in the two dimensional case.

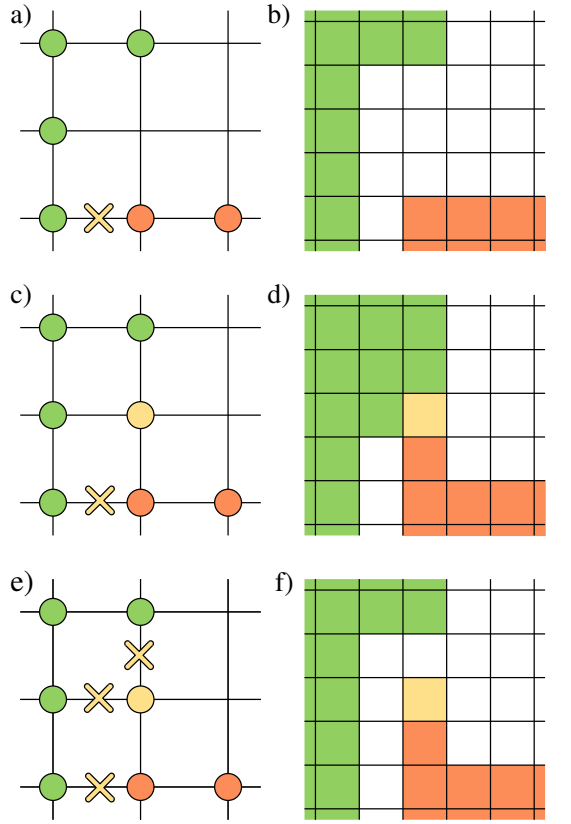


Figure 9: *Topological embeddings and Cut-edge Criterion. The figure above depicts three cut-edge grids (C_i, X_i) (Figures a,c,e) and their corresponding embeddings $\mathcal{E}_i = \mathcal{E}(C_i, X_i)$ (Figures b,d,f). (C_1, X_1) is not equivalent to (C_0, X_0) as the yellow voxel is not simple w.r.t. $\mathcal{E}_1$. This can be remedied by introducing additional cuts as in (C_2, X_2). Now (C_2, X_2) is equivalent to (C_0, X_0) as the yellow voxel is simple w.r.t. $\mathcal{E}_2$.*

4.3. Choice of Cuts

Whenever a voxel $\mathbf{x}^k$ is about to be conquered in the k-th step of the fast marching method (see Figure 3) we first test it for simplicity. If it is simple, the algorithm proceeds as usual. If not, we have to introduce some additional cuts around $\mathbf{x}^k$ in order to prevent a topology change.

To be more precise, we extend the original algorithm (see Figure 3) by setting $X^0 = \emptyset$ in step **1** and altering the update rule **4** to

$$\begin{aligned} C^{k+1} &\leftarrow C^k \cup D^k \\ X^{k+1} &\leftarrow X^k \cup Y^k \end{aligned}$$

where $D^k \subset \{\mathbf{x}^k\}$ and $Y^k \subset \{(\mathbf{x}^k, \cdot)\}$ is a set of cuts that is introduced at the k-th step of the algorithm. We can distinguish 5 cases, see Figure 7, where $n_{int} = n_{int}\left(\mathbf{x}^k, \mathcal{E}(C^k, X^k)\right)$ and $n_{ext} = n_{ext}\left(\mathbf{x}^k, \mathcal{E}(C^k, X^k)\right)$:

1. $n_{int} = n_{ext} = 1$: $\mathbf{x}^k$ is simple, $D^k = \{\mathbf{x}^k\}, Y^k = \emptyset$.
2. $n_{int} = 0$: The algorithm is about to create a new component of conquered voxels. Obviously, this is not possible with the fast marching approach, hence this case will never occur.
3. $n_{ext} = 0$: The algorithm is about to close a cavity. We could handle this by introducing an arbitrary cut on one of $\mathbf{x}^k$'s adjacent edges, however, for symmetry reasons, we choose to simply reject $\mathbf{x}^k$, and set $D^k = Y^k = \emptyset$.
4. $n_{ext} \geq 2$: The algorithm tries to close a handle. Here again we just reject $\mathbf{x}^k$, $D^k = Y^k = \emptyset$.
5. $n_{int} \geq 2$: The algorithm tries to merge two interior components. This is the standard case when two parts of the front or two different fronts collide. To resolve this case we search the nearest (in terms of arrival time η) 6-component $A \subset N_{26}^*(\mathbf{x}^k) \cap C^k$ that is adjacent to $\mathbf{x}^k$. Then we introduce cuts to all other adjacent components, $D^k = \{\mathbf{x}^k\}$, $Y^k = \{(\mathbf{x}^k, \mathbf{w}) : \mathbf{w} \in C^k \cap N_{26}^*(\mathbf{x}^k) \setminus A\}$.

4.4. Cut Positioning

In order to resolve the topology of a cut-edge configuration it is sufficient to know for all edges, whether they are cut or not. However, if we want to actually extract the surface, we also have to know, where the cuts are placed, i.e. in which ratio the edge is subdivided by the cut. For this, we proceed as follows. Let $e = (\mathbf{v}_0, \mathbf{v}_1)$ be a cut edge and $\mathbf{d} = \mathbf{v}_1 - \mathbf{v}_0$ the direction of the edge. We approximate the slopes m_0, m_1 of the arrival time function η in $\mathbf{v}_0$, $\mathbf{v}_1$ by letting

$$m_0 = \eta(\mathbf{v}_0) - \eta(\mathbf{v}_0 - \mathbf{d}) \quad \text{and} \quad m_1 = \eta(\mathbf{v}_1) - \eta(\mathbf{v}_1 + \mathbf{d})$$

Then we extrapolate the contact point $(1-s)\mathbf{v}_0 + s\mathbf{v}_1$ where s is given by

$$s = \frac{\eta(\mathbf{v}_1) - \eta(\mathbf{v}_0) + m_0}{m_0 + m_1}$$

4.5. Surface Extraction

In order to extract an explicit representation, i.e. a polygonal mesh, from a cut-edge grid (C, X), we proceed as follows (conf. Figure 10).

1. Map (C, X) onto its embedding $\mathcal{E}(C, X)$. Note that this can be done locally for each voxel, hence there is no need to actually use a voxel grid of double resolution.
2. $\mathcal{E}(C, X)$ is viewed as a grid, and the ordinary marching cubes algorithm is applied.
3. Vertices that do lie on the original grid edges are removed by collapsing one of their adjacent edges.
4. The remaining vertices are moved to their final positions as described in the previous section.

Note that depending on the underlying representation of polygonal meshes under certain circumstances not all vertices can be collapsed. In this case we move them to the position of one of their neighbors.

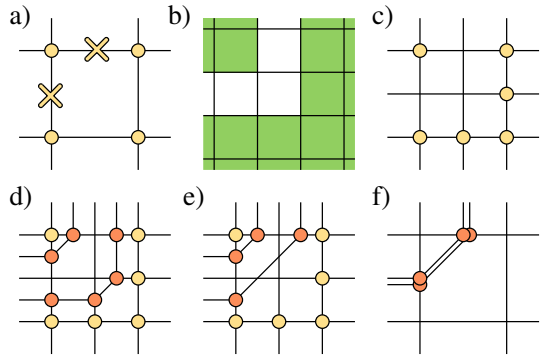


Figure 10: *Extracting a polygonal mesh from a cut-edge grid: The cut-edge configuration (a) is mapped to its corresponding digital set (b). This digital set is then viewed as a grid (c) and the original marching cubes algorithm is applied (d). Vertices that do not correspond to edges of the original configuration are removed (e) and the remaining vertices are moved into the final position of the contact surface (double vertices) (f).*

5. Results

We have implemented and tested our framework on various datasets. In all Figures, the contact area is shown in green, while the remaining surface is shown in red.

Figure 11 shows a trefoil knot as an example of a genus zero initial surface. Expanding this knot at unit speed leads to a self-contact and to a halt of the corresponding part of the surface.

Figure 12 shows some spheres as an example of a surface consisting of multiple components. Expanding these spheres at unit speed within a bounding box results in a Voronoi-like decomposition of the box. Note again, the the surfaces do not self-intersect but come to a halt as soon as they touch each other.

Figure 13 shows the reconstruction of a human brain from an MRT dataset. For this we set a small sphere into the interior of the brain and let it grow with speed proportional to the image intensity, i.e.

$$f(\mathbf{x}) = \max(i_{\mathbf{x}} - t, 0)$$

where we iteratively lower the threshold t in order to successively conquer the brain. It is elaborate to implement such a strategy for the fast-marching framework, hence we created these images using a narrow-band implementation of the standard level-set framework. The topological framework, however, remained the same.

Note that the detection of simple points and the placement of cuts is only a local process and hence very efficient. The running times for our examples range from a few seconds to some minutes. Adding topology control slows down the running time of the fast marching method by a factor of about 6. This is not astonishing, as the fast marching method is very efficient in itself, hence even a small computational overhead will have a strong impact on the running time. Adding topology control to other, computationally more involved level-set implementations, like narrow-band implementations, results in a more moderate slow-down of about 10%, see also [8].

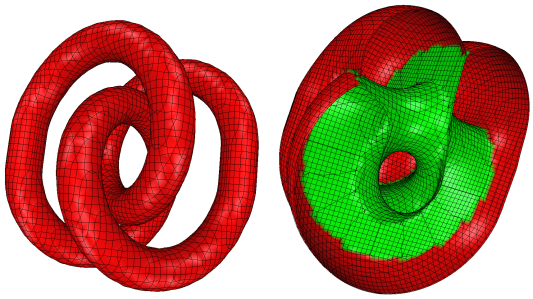


Figure 11: *A genus zero trefoil knot is expanded at unit speed. Original knot (top), expanded knot hiddenline (left) and colored, smoothed (right). The green color designates the area where the surface has touched itself. The resolution of the cut-edge grid is* 64^3.

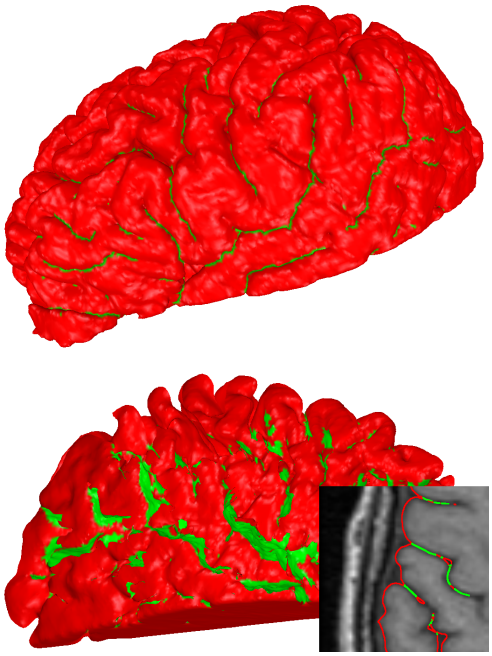


Figure 13: *Reconstruction of the cortex of a brain. Upper image: The green color signifies areas where the surface has touched itself. Without topology control, these areas would have merged together resulting in a reconstruction of wrong genus. Lower image: Interior of the brain cortex. The resolution of the cut-edge grid is* $95 \times 256 \times 88$

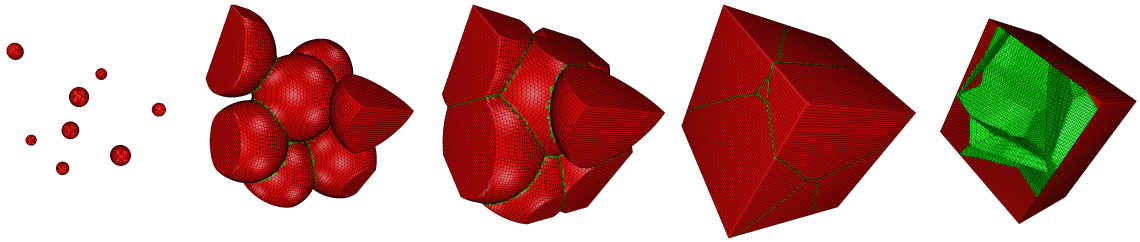


Figure 12: *A number of spheres with different radii are expanded at unit speed within a bounding box. The result is a Voronoi-like decomposition of the box. The resolution of the underlying cut-edge grid is* 64^3.

6. Conclusion and Future Work

We have presented a novel framework for controlling the topology of geometric deformable models. The topology constraints of the initial surface are preserved by introducing cuts on the grid edges during the deformation process. The main features of our cut-edge framework are its

- efficiency,
- sub-voxel accuracy and
- compatibility to existing level-set frameworks.

In all our experiments, the cut-edge framework produced valid meshes of correct topology. However, we have to point out that in our present implementation small oscillations on the contact surfaces may occur. This problem is of geometric rather than of topological nature. We plan to address it by using higher-order approximation methods for the fast marching method as well as for the computation of the cut positions (conf. Section 4.4).

References

1. G. Bertrand. Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pattern recognition letters*, 15:1003–1011, 1994.
2. I. Cohen, L. 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, 1992.
3. L. Cohen. On active contour models and balloons. *Computer Vision, Graphics and Image Processing: Image Understanding*, 53(2):211–218, 1991.
4. C. A. Davatzikos and J. L. Prince. An active contour model for mapping the cortex. *IEEE Trans. on Medical Imaging*, 14(1):112–115, 1995.
5. H. Delingette and J. Montagnat. Shape and topology constraints on parametric active contours. *Computer Vision and Image Understanding*, 83:140–171, 2001.
6. M. Desbrun and M.-P. Cani. Active implicit surface for animation. In *Graphics Interface*, pages 143–150, 1998.
7. N. Foster and R. Fedkiw. Practical animation of liquids. In *SIGGRAPH 01 proceedings*, pages 23–30, 2001.
8. X. Han, C. Xu, and J. L. Prince. A topology preserving deformable model using level sets. In *Computer Vision and Pattern Recognition Proceedings*, pages 765–770, 2001.
9. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Internation Journal of Computer Vision*, 1:321–331, 1988.
10. J.-O. Lachaud and A. Montanvert. Deformable meshes with automatic topology changes for coarse-to-fine three-dimensional surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.
11. R. Malladi, J. A. Sethian, and B. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–174, 1995.
12. T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In *International Conference on Computer Vision*, pages 840–845, 1995.
13. K. Museth, D. E. Breen, R. T. Whitaker, and A. H. Barr. Level set surface editing operators. In *SIGGRAPH 02 proceedings*, volume 21, pages 330–338, 2002.
14. S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.
15. S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.
16. J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93(4):1591–1595, 1996.
17. J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.
18. R. Whitaker. A level-set approach to 3d reconstruction from range data. In *International Journal of Computer Vision*, volume 29, pages 203–231, 1998.