Delaunay Deformable Models: Topology-Adaptive Meshes Based on the Restricted Delaunay Triangulation

Jean-Philippe Pons
WILLOW, INRIA / ENS / École des Ponts
Paris, France
pons@certis.enpc.fr

Jean-Daniel Boissonnat
GEOMETRICA, INRIA
Sophia-Antipolis, France
boissonnat@sophia.inria.fr

Abstract

In this paper, we propose a robust and efficient Lagrangian approach, which we call Delaunay Deformable Models*, for modeling moving surfaces undergoing large deformations and topology changes. Our work uses the concept of restricted Delaunay triangulation, borrowed from computational geometry. In our approach, the interface is represented by a triangular mesh embedded in the Delaunay tetrahedralization of interface points. The mesh is iteratively updated by computing the restricted Delaunay triangulation of the deformed objects. Our method has many advantages over popular Eulerian techniques such as the level set method and over hybrid Eulerian-Lagrangian techniques such as the particle level set method: localization accuracy, adaptive resolution, ability to track properties associated to the interface, seamless handling of triple junctions. Our work brings a rigorous and efficient alternative to existing topology-adaptive mesh techniques such as T-snakes.*

1. Introduction

Deformable models, also known in the literature as snakes, active contours/surfaces, deformable contours/surfaces, constitute a widely used computerized technique to address various shape reconstruction problems in image processing. They have been initially proposed for the purpose of image segmentation by Kass, Witkin and Terzopoulos in [24], but they have proven successful in many other contexts, in computer vision and in medical imaging, including region tracking and shape from X. More generally, modeling dynamic interfaces between several materials undergoing large deformations is a ubiquitous task in science and engineering. Let us mention computer aided design, physics simulation and computer graphics.

The existing techniques roughly fall into two categories: Eulerian and Lagrangian formulations. It is commonly admitted that both viewpoints have strengths and weaknesses and that only a hybrid approach can overcome the limitations of both. In this paper, we propose a method which has the particularity of being purely Lagrangian with all the associated advantages, while achieving topological adaptivity with a comparable robustness and efficiency to Eulerian methods.

1.1. Eulerian methods

The Eulerian formulation casts deformation as a time variation of quantities defined over a fixed grid. The interfaces have to be represented implicitly, since the grid does not conform to them. In computational physics, this is also known as the *front capturing* method. Two notable front capturing techniques are the *level set method*, introduced by Osher and Sethian [34], and the *volume-of-fluid (VOF) method*, pioneered by Hirt and Nichols [23].

Hereafter, we will mainly focus on the level set method because it is an established technique in computer vision. Basically, this method consists in representing the interface as the zero level set of a higher-dimensional scalar function. The movement of the interface can be cast as an evolution of the embedding level set function by an Eulerian PDE (partial differential equation). We refer the reader to some good reviews [33, 40] for all the details about the theory, the recent developments, the implementation and the applications of the level set method.

On the one hand, this approach has several advantages over an explicit Lagrangian representation of the interface: no parameterization is needed, topology changes are handled automatically, intrinsic geometric properties such as normal or curvature can be computed easily from the level set function. Last but not least, the theory of *viscosity solutions* provides robust numerical schemes and strong mathematical results to deal with the evolution PDE. These advantages explain the popularity of the level set method not only in computer vision but also for multi-phase fluid flow simulation [42] in CFD (computational fluid dynamics), as well as for computer animation of fluids with free surfaces [18, 19, 20, 21, 28].

On the other hand, several serious shortcomings limit the applicability of the level set method:

First, the higher dimensional embedding makes the level set method much more expensive computationally than explicit representations. Much effort has been done to alleviate this drawback, leading to the *narrow band* methodology [1] and to the *PDE-based fast local level set method* [36]. More recently, octree decompositions have been proposed [10, 19, 27, 28] to circumvent the typically fixed uniform sampling of the level set method, in order to reach high resolution (typically an effective resolution of 512^3) while keeping the computational and memory cost sustainable. However, these methods somewhat lose the simplicity of the original level set method, as an efficient implementation of such tree-based methods turns out to be a tricky task.

Second, as discussed and numerically demonstrated by Enright *et al.* in [18], the level set method is strongly affected by mass loss, smearing of high curvature regions and inability to resolve very thin parts. These limitations have motivated the development of some hybrid Eulerian-Lagrangian methods, such as the *particle level set method* outlined by Foster and Fedkiw [21] and later improved by Enright and coworkers [18, 19, 20]. While the latter method yields state-of-the-art results, an objection could be the large number of parameters controlling the particle reseeding strategy included in this approach.

Third, purely Eulerian formulation is not very appropriate for tracking interface properties such as color or texture coordinates, as may be needed in computer graphics applications. Some approaches based on a coupled system of Eulerian PDEs were recently proposed to overcome this limitation within the level set framework [2, 37, 46], but this capability comes at a significant additional computational cost.

1.2. Previous Lagrangian methods

The Lagrangian formulation adopts a more “natural” point of view. It explicitly tracks the interfaces between the different materials with some points advected by the motion. There are mainly two classes of Lagrangian techniques: *mesh-based* methods and *particle-based* methods. We first tackle the mesh-based approach, which corresponds to the so-called ”snake” methodology in computer vision. and which is also known as the *front tracking* method in computational physics. When dealing with large deformations, this approach is hampered by distortion and entanglement of the mesh, source of numerical instabilities or even of breakdowns of the simulation. For instance, if corners or cusps develop in the evolving front, front tracking methods usually form “swallowtail” solutions. These defective parts of the interface must be detected, then removed through intricate delooping procedures.

Another major shortcoming of the mesh-based Lagrangian approach is that a fully automatic, robust and efficient handling of topology changes remains an open issue, despite several heuristic solutions proposed in computer vision [8, 12, 13, 17, 25, 26, 30, 31].

McInerney and Terzopoulos [30, 31] propose topology adaptive deformable curves and meshes, called *T-snakes* and *T-surfaces*. During the evolution, the model is periodically resampled by computing its intersections with a regular simplicial decomposition of space. A labeling of the vertices of the simplicial grid as inside or outside of the model is maintained. This procedure loses the desirable adaptivity of the Lagrangian formulation, by imposing a fixed uniform spatial resolution. Also, not all motions are admissible: this approach only works when the model inflates or deflates everywhere, which considerably restricts the range of applications.

Several authors have proposed alternatives to the *T-snakes* approach: Lachaud and coworkers [25, 26], Bredno *et al.* [8], Duan and Qin [17], Delingette and Montagnat [12, 13]. Basically, these approaches consist in detecting self-intersections in the evolving mesh and in merging the colliding regions using a set of heuristic remeshing rules. Unfortunately, the detection of intersection is computationally expensive, even when optimizing pairwise distance computations with an octree structure. In practice, it requires the most part of total computation time. Also, these approaches lack a systematic and provably correct remeshing strategy. Consequently, they are very likely to break in some complex or degenerate practical cases.

This major shortcoming of mesh-based methods have gained popularity to the particle-based approach [9, 14, 15, 22, 32, 35, 38, 43, 44] for representing dynamic interfaces undergoing complex topology changes. However, it is generally admitted that with the meshless approach, the localization of the interface and the computation of interface properties such as normal and curvature gets cumbersome.

1.3. Novelty of our method

In this paper, we present a purely Lagrangian approach, which combines the advantages of front tracking and front capturing methods, while discarding their respective drawbacks. Our work brings a robust and efficient solution to remeshing and topological adaptivity for Lagrangian dynamic interfaces, thanks to the concept of restricted Delaunay triangulation, borrowed from computational geometry. In our approach, the interface is represented by a triangular mesh embedded in the Delaunay tetrahedralization of interface points. Interestingly, this increase of dimension bears similarity to the spirit of the level set method. This embedding tetrahedralization enforces directly watertight interfaces free of loops, swallowtails or self intersections at all times. The mesh is iteratively updated by computing the restricted Delaunay triangulation of the deformed objects. Thus, our approach does not need heuristic rules to update the mesh connectivity. The proximity structure encoded in the Delaunay triangulation allows to detect topology changes efficiently and to handle them naturally.

Moreover, improving the geometric quality and adapting the resolution of the deformable model is far easier than in previous approaches, since the adequate connectivity is automatically obtained through the restricted Delaunay triangulation.

Being purely Lagrangian, our method does not suffer from mass loss which plagues the level set method, and can track material properties such as color or texture coordinates during motion at no additional cost, while being free of the localization problem of particle-based approaches.

Finally, our method is not limited to two-phase motion. It seamlessly accommodates any number of materials, whereas this requires special care in most existing Lagrangian and Eulerian methods (*cf* for example [41] on the treatment of triple junctions with the level set method). To achieve this, it suffices to label each Delaunay tetrahedron with the type of material it contains.

The remainder of this paper is organized as follows. Section 2 gives some background on the basic computational geometry concepts needed in our approach: Voronoi diagrams, Delaunay triangulations and restricted Delaunay triangulations. Our algorithm is described in Section 3. In Section 4, we demonstrate the applicability and the efficiency of our approach with some numerical experiments, including the multi-label segmentation of real medical images.

2. Background

2.1. Voronoi diagram and Delaunay triangulation

Voronoi diagrams are versatile structures which encode proximity relationships between objects. They are particularly relevant to perform nearest neighbors search and motion planning (*e.g*. in robotics), and to model growth processes (*e.g*. crystal growth in materials science). Delaunay triangulations, which are geometrically dual to Voronoi diagrams, are a classical tool in the field of mesh generation and mesh processing due to its optimality properties.

Most of the following definitions are taken from [6]. We also refer the interested reader to some recent computational geometry textbooks [7, 11].

In the sequel, we call k-*simplex* the convex hull of $k+1$ affinely independent points. For example, a 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle and a 3-simplex is a tetrahedron.

Let $E = \{p_1, \ldots, p_n\}$ be set of points in $\mathbb{R}^d$. Note that

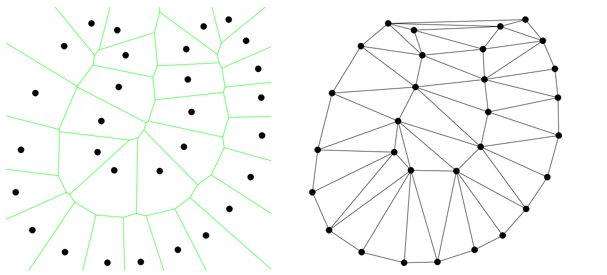


Figure 1. *Left:* Voronoi diagram of a set of points in the plane. *Right:* Its dual Delaunauy triangulation.

in this work, we are mainly interested in $d = 2$ and $d = 3$. The *Voronoi region*, or *Voronoi cell*, denoted by $V(p_i)$, associated to a point p_i is the region of space that is closer from p_i than from all other points in E:

$$V(p_i) = \{p \in \mathbb{R}^d \ : \ \forall j, \ \|p - p_i\| \leq \|p - p_j\|\} \ . \qquad (1)$$

$V(p_i)$ is the intersection of $n-1$ half-spaces bounded by the bisector planes of segments $[p_i p_j], \ j \neq i$. $V(p_i)$ is therefore a convex polytope, possibly unbounded. The *Voronoi diagram* of E, denoted by $\mathrm{Vor}(E)$, is the partition of space induced by the Voronoi cells $V(p_i)$.

See Figure 1 for a two-dimensional example of a Voronoi diagram In two dimensions, the edges shared by two Voronoi cells are called *Voronoi edges* and the points shared by three Voronoi cells are called *Voronoi vertices*. Similarly, in three dimensions, we term *Voronoi facets, edges* and *vertices* the geometric objects shared by one, two and three Voronoi cells, respectively. The Voronoi diagram is the collection of all these k-dimensional objects, with $0 \leq k \leq d$, which we call *Voronoi objects*. In particular, note that Voronoi cells $V(p_i)$ correspond to d-dimensional Voronoi objects.

The *Delaunay triangulation* $\mathrm{Del}(E)$ of E is defined as the geometric dual of the Voronoi diagram: there is an edge between two points p_i and p_j in the Delaunay triangulation if and only if their Voronoi cells $V(p_i)$ and $V(p_j)$ have a non-empty intersection. It yields a *triangulation* of E, that is to say a partition of the convex hull of E into d-dimensional simplices (*i.e*. into triangles in 2D, into tetrahedra in 3D and so on).

The fundamental property of the Delaunay triangulation is called the *empty circle* (resp. *empty sphere* in 3D) property: in 2D (resp. in 3D), a triangle (resp. tetrahedron) belongs to the Delaunay triangulation if and only if its circumcircle (resp. circumsphere) does not contain any other points of E.

The algorithmic complexity of the Delaunay triangulation of n points is $\mathcal{O}(n \log n)$ in 2D, and $\mathcal{O}(n^2)$ in 3D. Fortunately, as was recently proven in [4], the complexity in 3D drops to $\mathcal{O}(n \log n)$ when the points are distributed on a smooth surface, which is the case of interest here.

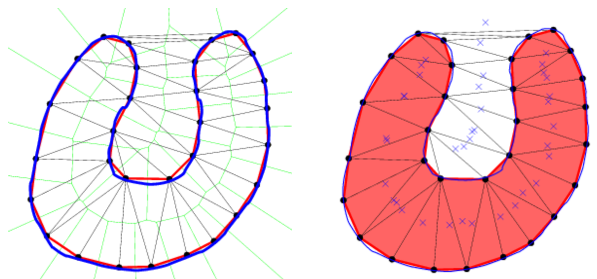


Figure 2. *Left:* The Delaunay triangulation restricted to the blue curve is plotted with a thick red line. *Right:* The Delaunay triangulation restricted to the region bounded by the blue curve is composed of the filled red triangles, whose circumcenters (blue crosses) are inside the region.

2.2. Restricted Delaunay triangulation

Each k-simplex in the Delaunay triangulation is dual to a $(d-k)$-dimensional Voronoi object. For instance, in 3D, the dual of a Delaunay tetrahedron is the Voronoi vertex which coincides with the circumcenter of the tetrahedron, the dual of a Delaunay facet is a Voronoi edge, the dual of a Delaunay edge is a Voronoi facet, and the dual of a Delaunay vertex p_i is the Voronoi cell $V(p_i)$.

Given a subset $\Omega \in \mathbb{R}^d$, typically a manifold of dimension $k \leq d$, we call the *Delaunay triangulation of E restricted to* Ω, and we note $\text{Del}|_\Omega(E)$ the subcomplex of $\text{Del}(E)$ composed of the Delaunay simplices whose dual Voronoi objects intersect Ω. For example, in 2D, as illustrated in Figure 2 *(left)*, the Delaunay triangulation restricted to a curve C is composed of the Delaunay edges whose dual Voronoi edges intersect C. Similarly, as shown in Figure 2 *(right)*, the Delaunay triangulation restricted to a region R is composed of the Delaunay triangles whose circumcenters are contained in R. The attentive reader may have noticed that in both cases the restricted Delaunay triangulation forms a good approximation of the object.

Actually, this is a general property of the restricted Delaunay triangulation. It can be shown that, under some assumptions, and especially if E is a "sufficiently dense" sample of Ω, in some sense defined in [3], $\text{Del}|_\Omega(E)$ is a good approximation of Ω, both in a topological and in a geometric sense: as regards topology, $\text{Del}|_\Omega(E)$ is homeomorphic to Ω; as regards geometry, the Hausdorff distance between $\text{Del}|_\Omega(E)$ and Ω can be made arbitrarily small; normals and curvatures of Ω can be consistently approximated from $\text{Del}|_\Omega(E)$.

Based on these approximation properties, a family of provably correct algorithms for mesh generation and mesh reconstruction from point clouds have been designed in the last decade. We refer the reader to [6] and references therein for more details.

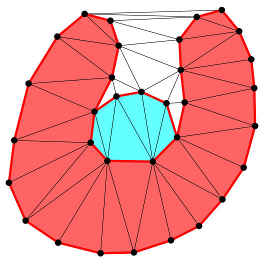


Figure 3. Embedding labeled Delaunay triangulation, including three materials and triple junctions. The different materials are color-coded. Thick red edges indicate the interfaces between different materials.

3. Methods

3.1. Embedding labeled Delaunay triangulation

A determinant feature of our approach is that the deforming objects and the interfaces between them are modeled as a subset of the Delaunay simplices of a point sample E.

More specifically, each tetrahedron of $\text{Del}(E)$ is labeled with the type of material it contains. For instance, in a multi-phase fluid simulation, each tetrahedron would be mapped to one of the phases. This information, that we call for short the *label* of a tetrahedron, can be conveniently represented by an integer value. The interfaces of interest are embedded in the Delaunay triangulation: they are composed of the triangular facets adjacent to two tetrahedra having different labels.

This representation has many virtues. It automatically enforces watertight interfaces. It seamlessly accommodates any number of materials. It makes the request of the type of material at some point inexpensive, since there exist very efficient location algorithms dedicated to triangulations [11]. An example of this embedding Delaunay triangulation in 2D, including three materials with some triple junctions, is displayed in Figure 3.

3.2. Algorithm

We now describe how our embedding labeled Delaunay triangulation is consistently updated during motion, in order to faithfully reflect the deformations and the topology changes of the different materials. For sake of simplicity, the following description only deals with the case of two materials (inside and outside), the most common in practice. However, the described algorithm extends to any number of materials with a few minor changes.

Let us denote by E_n the point sample at iteration n, and by D_n the subset of the embedding Delaunay triangulation which is labeled as inside, *i.e.* foreground objects, at the corresponding time. The core idea of our approach is to obtain D_n by taking the Delaunay triangulation of E_n restricted to some region Ω_n:

$$D_n = \mathrm{Del}|_{\Omega_n}(E_n) \ . \qquad (2)$$

Given the approximation properties of the restricted Delaunay triangulation evoked in the previous section, this approach is valid as soon as E_n is a good point sample of the interfaces and Ω_n faithfully represents the interior of objects.

With this construction on hand, we only need to compute the point sample E_{n+1} at the subsequent iteration and the domain Ω_{n+1} after deformation, given the current E_n and Ω_n. As for the connectivity of interface meshes, it is automatically determined through Equation (2). No heuristic and intricate remeshing rule is needed, as opposed to previous work [8, 12, 13, 17, 25, 26].

E_{n+1} is obtained simply by advecting the points in E_n with the desired velocity field, then by adapting the sampling resolution if needed. Specifically, for each pair of points of E_n corresponding to an interface edge in D_n:

- if the distance between the two displaced points is too small, the points are discarded and replaced by their midpoint in E_{n+1};
- otherwise, the displaced points are included in E_{n+1};
- if the distance between the two displaced points is too large, their midpoint is also added to E_{n+1}.

Of course, the aforementioned length thresholds may be dependent on the input data or on the geometry of the interface itself, *e.g.* on the structure tensor of the image as in [26], or on the curvature of the object. Also, we emphasize that, in contrast with existing approaches, the above resampling procedure can be applied without taking care of the connectivity of interface meshes. Again, the adequate connectivity after resampling is recovered through Equation (2).

The ability of our approach to deal with topology changes is intimately related to the computation of the domain of objects after deformation. Ω_{n+1} is obtained as the union of tetrahedra in D_n, after advecting their vertices with the velocity field, and after discarding tetrahedra which have become inverted. This can be written as

$$\Omega_{n+1} = \bigcup \{\tilde{T} \mid T \in D_n, \ \text{orientation}(\tilde{T}) > 0\} \ , \qquad (3)$$

where T denotes a tetrahedron and $\tilde{T}$ is the same tetrahedron after displacement.

For sake of clarity, the different substeps of an iteration of our algorithm are illustrated in Figure 4 *(a-f)*, in two dimensions:

- *(a)* shows the embedding triangulation D_n before deformation.
- *(b)* shows the triangulation after deformation, with its connectivity kept constant. Note that it is no longer a valid geometric triangulation, since there are some inverted triangles, dotted with yellow edges.
- *(c)* shows the domain Ω_{n+1}, *i.e.* the union of displaced triangles still having a correct orientation. Note that this domain accounts for the change in topology.
- *(d)* shows the Delaunay triangulation of the new point sample E_{n+1} along with the associated circumcenters, dotted with blue crosses.
- *(e)* shows the Delaunay triangulation of E_{n+1} restricted to the domain Ω_{n+1}. It is obtained by labeling Delaunay triangles as inside if the circumcenters displayed in *(d)* are inside the domain shown in *(c)*.
- In some cases, especially after a topology change, some vertices of the new triangulation may be surrounded by simplices all sharing the same label. In the example shown in *(e)*, four vertices are in this case. These vertices are of no use for representing the interfaces, so they are removed from the triangulation, yielding the final embedding Delaunay triangulation to be used in the next iteration, shown in *(f)*.

As regards removing useless points in the last substep of each iteration, using a Delaunay triangulation rather than an arbitrary triangulation plays an important role in 3D. Indeed, although it may be impossible to tetrahedralize the hole created by removing a point from an arbitrary 3D triangulation [39], it is always possible in a 3D *Delaunay* triangulation, and there exist efficient algorithms to do it [16].

4. Numerical experiments

By using *CGAL* (Computational Geometry Algorithms Library, homepage: www.cgal.org) [5], we have been able to implement our approach with only 1000 lines of C++ code. CGAL defines all the needed geometric primitives and provides an excellent algorithm to compute the Delaunay triangulation in 3D: it is robust to degenerate configurations and floating-point error, thanks to the use of exact geometric predicates, while being able to process millions of points per minute on a standard workstation. As regards the efficient computation of the restricted Delaunay triangulation, which requires to test if a point intersects a "soup" of tetrahedra, we use a *segment tree* data structure [11], also implemented in CGAL.

4.1. Imbricated toruses

In our first experiment, we demonstrate that our approach naturally handles topology changes and resolution

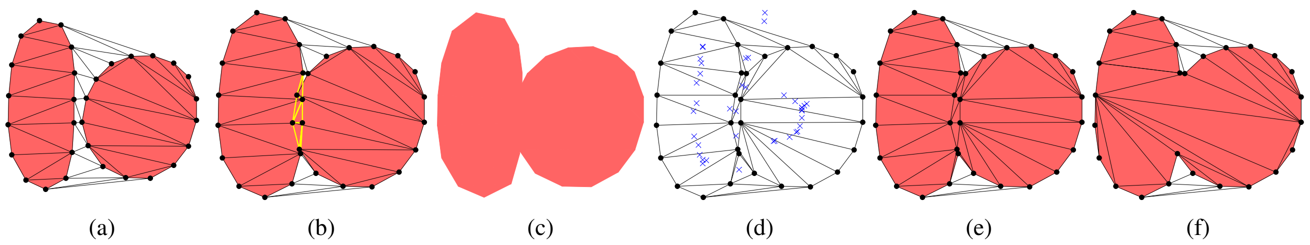


Figure 4. Substeps of one iteration of our algorithm (see text).

adaptivity, by segmenting a synthetic image of two imbricated toruses, starting from a sphere. The obtained evolution is shown in Figure 5. Note that we have purposely enforced different resolutions in the lower and upper parts of the object. Whereas this is straightforward to achieve with our approach, it is intricate when using Eulerian techniques such as the level set method.

In this experiment, the number of iterations to reach convergence is 120, the number of vertices ranges from 8,000 to about 22,000 along the evolution, and the total computation time is 171 s. Thus, the efficiency of our algorithm is comparable to the recent work of Lachaud and Taton [26]: roughly one second per iteration for 10,000 vertices. Comparing these timings to other related methods [8, 12, 13, 17, 30, 31] turned out to be a tricky task, due to missing information such as number of iterations, number of vertices, or hardware specifications. Moreover, as the complexity of these algorithms is not linear, a fair comparison would suppose to use the same number of vertices, if not the same dataset.

4.2. Medical image segmentation

In our second experiment, we apply our approach to the segmentation of real 3D medical data. In order to demonstrate the ability of our approach to seamlessly handle any number of materials, we use three different labels: air, soft tissues (skin, muscles, . . .) and bone. We set the velocity of the model to a combination of mean curvature motion and of a region-based data attachment term (see [29, 45] for more details). Figure 6 shows the input CT image that drives the motion and the different stages of the evolution. Note how the initial random seeds corresponding to the different tissues grow and progressively merge, while creating triple junctions, until they successfully partition the head.

Some movies of these experiments are available as supplemental material. Also, we plan to make our code available at the time of the conference.

5. Conclusion and future work

We have proposed a robust and efficient Lagrangian approach for modeling dynamic interfaces between different materials undergoing large deformations and topology changes, based on the rigorous concept of restricted Delaunay triangulation, borrowed from computational geometry. We have demonstrated the applicability and the efficiency of our approach with some numerical experiments, including the multi-label segmentation of real medical images. As our algorithm easily extends to any number of dimensions, our future work includes investigating its effectiveness in 4D, *e.g.* for dynamic scene reconstruction from multi-view image sequences, or for spatio-temporal MRI segmentation.

References

[1] D. Adalsteinsson and J. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995.

[2] D. Adalsteinsson and J. Sethian. Transport and diffusion of material quantities on propagating interfaces via level set methods. *Journal of Computational Physics*, 185(1):271–288, 2003.

[3] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.

[4] D. Attali, J.-D. Boissonnat, and A. Lieutier. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *Annual Symposium on Computational Geometry*, pages 201–210, 2003.

[5] J.-D. Boissonnat, O. Devillers, M. Teillaud, and M. Yvinec. Triangulations in CGAL. In *Annual Symposium on Computational Geometry*, pages 11–18, 2000.

[6] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67:405–451, 2005.

[7] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.

[8] J. Bredno, T. M. Lehmann, and K. Spitzer. A general discrete contour model in two, three, and four dimensions for topology-adaptive multichannel segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):550–563, 2003.

[9] M.-P. Cani and M. Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, 1997.

[10] T. Cecil, S. Osher, and J.-L. Qian. Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. *Journal of Computational Physics*, 213(2):458–473, 2006.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. SpringerVerlag, 1997.

[12] H. Delingette. General object reconstruction based on simplex meshes. *The International Journal of Computer Vision*, 32(2):111–146, 1999.

[13] H. Delingette and J. Montagnat. Shape and topology constraints on parametric active contours. *Computer Vision and Image Understanding*, 83(2):140–171, 2001.

[14] M. Desbrun and M.-P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation*, pages 61–76, 1996.

[15] M. Desbrun and M.-P. Gascuel. Animating soft substances with implicit surfaces. In *Proceedings of ACM SIGGRAPH*, pages 287–290, 1995.

[16] . Devillers and M. Teillaud. Perturbations and vertex removal in a 3d delaunay triangulation. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 313–319, 2003.

[17] Y. Duan and H. Qin. A subdivision-based deformable model for surface reconstruction of unknown topology. *Graphical Models*, 66(4):181–202, 2004.

[18] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[19] D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures*, 83:479–490, 2005.

[20] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *Proceedings of ACM SIGGRAPH*, pages 736–744, 2002.

[21] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH*, pages 23–30, 2001.

[22] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[23] C. Hirt and B. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.

[24] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *The International Journal of Computer Vision*, 1(4):321–331, 1987.

[25] J.-O. Lachaud and A. Montanvert. Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.

[26] J.-O. Lachaud and B. Taton. Deformable model with a complexity independent from image resolution. *Computer Vision and Image Understanding*, 99(3):453–475, 2005.

[27] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 2006. To appear.

[28] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. In *Proceedings of ACM SIGGRAPH*, pages 457–462, 2004.

[29] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[30] T. McInerney and D. Terzopoulos. Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Transactions on Medical Imaging*, 18(10):840–850, 1999.

[31] T. McInerney and D. Terzopoulos. T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4(2):73–91, 2000.

[32] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 141–151, 2004.

[33] S. Osher and R. Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational Physics*, 169(2):463–502, 2001.

[34] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[35] M. Pauly, R. Keiser, B. Adams, P. Dutr, M. Gross, and L. Guibas. Meshless animation of fracturing solids. *ACM Transactions on Graphics*, 24(3):957–964, 2005.

[36] D. Peng, B. Merriman, S. Osher, H.-K. Zhao, and M. Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[37] J. Pons, G. Hermosillo, R. Keriven, and O. Faugeras. Maintaining the point correspondence in the level set framework. *Journal of Computational Physics*, 2006. To appear.

[38] S. Premože, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*, 22(3):401–410, 2003.

[39] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry*, 7(3):227–253, 1992.

[40] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences*. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press, 1999.

[41] K. Smith, F. Solis, and D. Chopp. A projection method for motion of triple junctions by level sets. *Interfaces and Free Boundaries*, 4(3):263–276, 2002.

[42] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159, 1994.

[43] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *SIGGRAPH*, pages 185–194, 1992.

[44] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models (from goop to glop). *Graphics Interface*, pages 219–216, 1989.

[45] C. Xu, D. Pham, and J. Prince. Medical image segmentation using deformable models. In J. Fitzpatrick and M. Sonka, editors, *Handbook of Medical Imaging*, volume 1, chapter 3, pages 129–174. SPIE Press, 2000.

[46] J.-J. Xu and H.-K. Zhao. An Eulerian formulation for solving partial differential equations along a moving interface. *Journal of Scientific Computing*, 19:573–594, 2003.

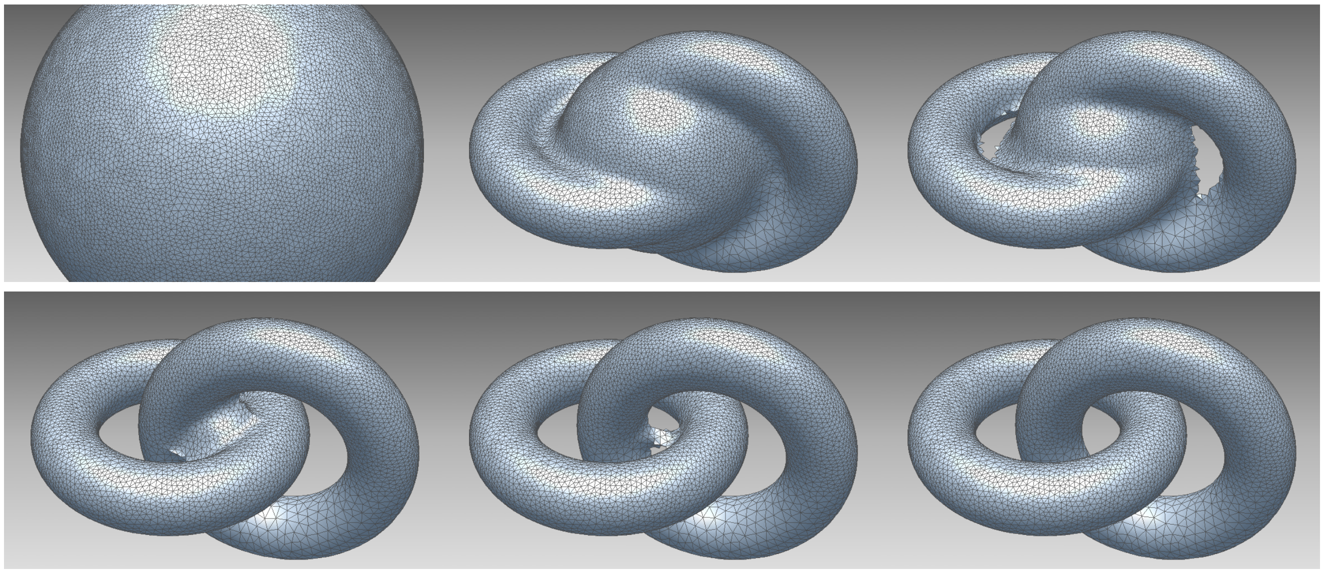
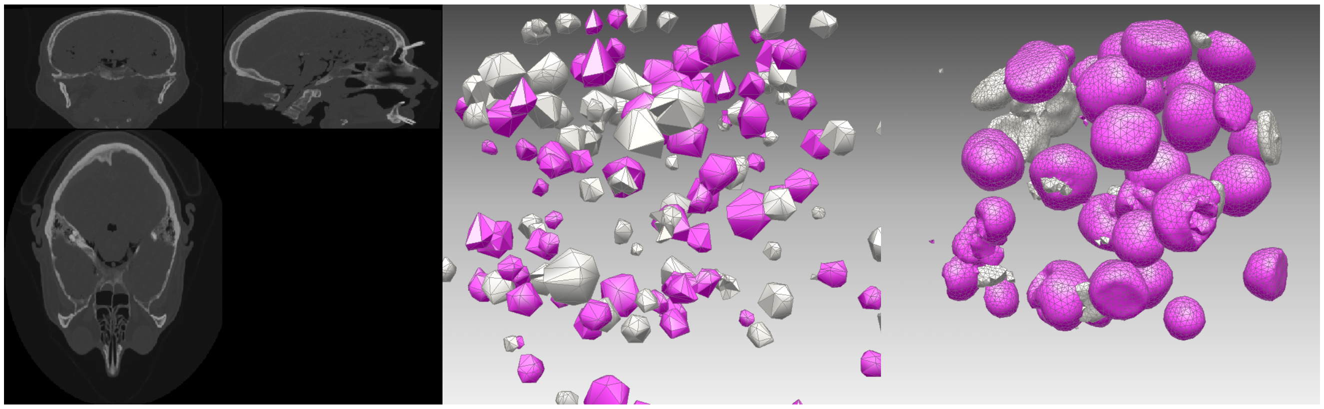


Figure 5. From left to right and top to bottom, different stages of the segmentation of two imbricated toruses, starting from a sphere, using a non-uniform sampling resolution.


(a) (b) (c)

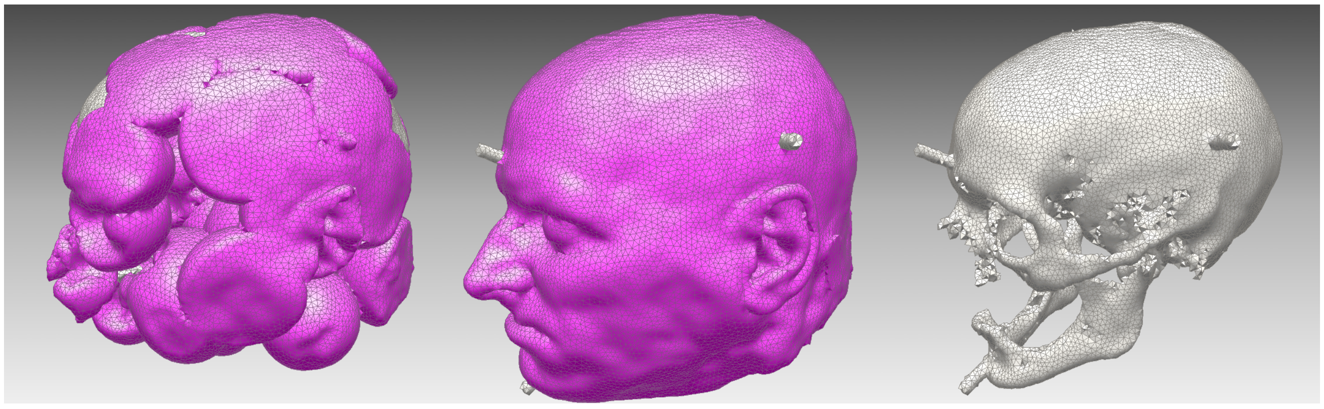

(d) (e) (f)

Figure 6. Application to multi-tissue medical image segmentation: (a) Some cuts of the input CT image. (b) Random initial seeds (soft tissues in pink, bone in white). (c-d) Different stages of the evolution. (e) Final three-material reconstruction. Note that the head positioning system is segmented as bone, which creates triple junctions in the final reconstruction. (f) Final bone surface.