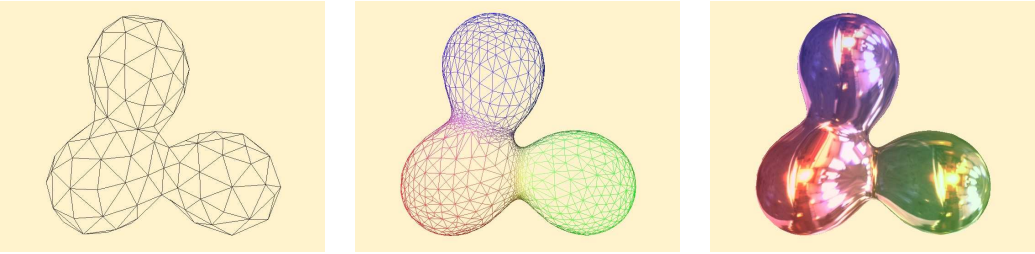
Twinned Meshes for Dynamic Triangulation of Implicit Surfaces

Antoine Bouthors, Matthieu Nesme

Twinned Meshes for Dynamic Triangulation of Implicit Surfaces

Antoine Bouthors Matthieu Nesme

EVASION - LJK/INRIA



Mechanical mesh Geometric mesh Rendering

ABSTRACT

We introduce a new approach to mesh an animated implicit surface for rendering. Our contribution is a method which solves stability issues of implicit triangulation, in the scope of real-time rendering. This method is robust, moreover it provides interactive and quality rendering of animated or manipulated implicit surfaces.
This approach is based on a double triangulation of the surface, a *mechanical* one and a *geometric* one. In the first triangulation, the vertices are the nodes of a simplified mechanical finite element model. The aim of this model is to uniformly and dynamically sample the surface. It is robust, efficient and prevents the inversion of triangles. The second triangulation is dynamically created from the first one at each frame. It is used for rendering and provides details in regions of high curvature. We demonstrate this technique with skeleton-based and volumetric animated surfaces.

Keywords: implicit surfaces, real-time, triangulation

1 INTRODUCTION

Implicit surfaces are a powerful tool for modeling and animating deformable objects such as organic structures or fluids [7]. Ray tracing is a natural choice for rendering implicit surfaces because ray-surface intersections are easily found with substitution and root finding. But ray tracing is generally expensive, and new real time methods require the latest programmable graphics hardware [1, 14, 19]. An alternative is to sample the implicit surface, which allows for point-based or triangle-based rendering. Moreover, sampling does not only permit rendering, it is useful in other tasks, such as modeling [30, 23, 6] or tracking surface features [4].

Among existing sampling approaches, tessellation methods such as the marching cubes algorithm [31, 22] have seen much evolution [16]. Re-tessellating when and where the surface changes [13] is computationally expensive and the resulting triangulation is not as regular as one could wish. Particle-based sampling methods are a very fast way to sample and follow dynamic implicit surfaces, but obtaining a rendering-friendly triangulation from a set of particles is far from trivial and is time-consuming. The goal of this paper is to propose a particle-based dynamic triangulation method that is robust, fast, and well adapted for rendering.

1.1 Related work

Particles-based techniques have been introduced [11] to sample implicit surfaces. The main idea is to constrain self-repulsing or self-attracting [28] particles on the surface and let them populate the surface. This model has been improved with approaches to create or remove particles in regions where they are too sparse or too dense [30], to take into account the curvature of the surface in the particle system [8], to handle sharp features [27], or to speed up the creation of the initial particle system [20].

Point-based rendering methods [21] are less desirable in comparison to triangles because overlapping the points (*i.e.,* surface elements) can produce noticeable artifacts. A triangular mesh can be obtained from a particle system via Delaunay triangulation at each animation step [11]. A faster method is to take into account neighboring information given by the particle system [9]. More time can be saved by exploiting temporal coherence making the triangulation evolve with the particle system [28]. The idea is to start from an initial triangulation and convert each vertex into a particle and each edge into a spring linking the two particles. The vertices (or particles) evolve under the forces of the springs, constrained on the surface. In addition, triangles and springs are inserted or deleted accordingly when particles are created or removed.

A difficulty of this kind of technique is topology changes. When surfaces split or merge, triangles have to be cut or created accordingly to create or merge different triangulations. [29] introduced a method to guarantee the topology of a dynamic implicit surface by searching for and tracking *critical points* [24] in the field function and modifying the polygonization when necessary.

As a dynamic implicit surface changes shape, the triangulation that we use for rendering must be fine enough along the directions of high curvature to capture the desired quality (*e.g.,* for reflections). One approach for achieving this is to modify the rest lengths of the springs depending on the local curvature so that additional detail is pulled into areas as required. But changing the rest shape of the mechanical elements through time and adapting their sizes to curvature makes the system unstable and increases computations. The additional detail that is ultimately necessary for rendering is unnecessary and a waste if used with the mechanical system.

In addition, triangles sometimes become inverted as the system is updated. If no means are taken against this issue, the triangulation becomes non-manifold and stays invalid. A solution to this problem was very briefly introduced in [28], but was based on collapsing edges, which can create additional inverted elements. Such kinds of stresses on the polygonization have been pointed out in [15], along with some solutions.

1.2 Overview

We propose to solve these stability and computation issues by decorrelating the needs for a stable particle system and those for a rendering-friendly geometry. We consider two separate representations of the surface. The first one, that we call the *mechanical mesh*, is used to follow dynamically the animated surface. The second one, called *geometric mesh*, is generated from the mechanical mesh and used for rendering purposes.

The **mechanical mesh** acts like a spring-based particle system [28]. The main criterions for its design are robustness and speed. Instead of a mass-spring system, we use a more robust Finite Element Method (FEM) with quasi-static solving. We give all the triangles a common size, aspect ratio, and stiffness, and we do not add more mechanical triangles in high curvature areas. We identify origins of inversions and propose additional means to avoid and recover from them.

The **geometric mesh** is generated by refining the mechanical mesh in areas of high curvature. As a result, each mechanical triangle can correspond to one or more geometric triangles. It is the geometric mesh that is used for rendering.

2 The Mechanical Mesh

In this section the terms mesh, triangles, vertices, edges and particles are all related to the mechanical mesh.

2.1 Mesh initialization

We must have an initial mesh for the implicit surface before we can start tracking how it changes. We use a marching cubes algorithm to construct the initial mesh, however, any convenient can be used. Each vertex of this initial mesh becomes a particle on which forces are applied. Though the quality of this initial mesh can be poor due to near degenerate triangles, our method will quickly modify the triangulation to produce a mesh consisting of near equilateral triangles (see Figure 1). Note that the user does not have to go through a mesh relaxation before animating the implicit function. The animation can start with the initial mesh. Our method is robust enough to track the surface with the initial mesh and relax it on the fly.

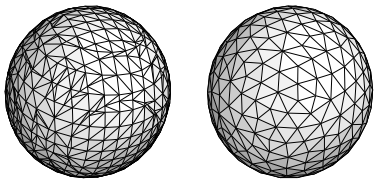


Figure 1: *Left:* the initial state of the mechanical model, corresponding to the result of a marching cubes algorithm. *Right:* at equilibrium, the mechanical model has evolved to a more regular mesh.

2.2 Quasi Static Time Integration

We use a quasi static solution to solve for the successive equilibrium states of the mechanical mesh. Since each step is independent, the errors of previous steps are not accumulated. At each frame t, we search for the ideal positions $\mathbf{x}_t$ of the particles. For a time step length dt the new positions are given by $\mathbf{x}_{t+dt} = \mathbf{x}_t + \triangle\mathbf{x}_t$. The ideal positions can be found by solving the non-linear system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ respecting the force equilibrium. To find a solution, we use the modified implicit solver of [3] based on a conjugate gradient method that omits the mass and velocity terms, which results in

$$\mathbf{I} - dt\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \triangle \mathbf{x} = dt\mathbf{f}(\mathbf{x}). \tag{1}$$

2.3 A FEM-based Approach

Several methods for modeling forces and force derivatives exist. The two most popular in computer graphics are the mass-spring system and the finite element method. Mass-spring systems are very fast but have many known drawbacks. Particularly, they cannot precisely represent triangles because resulting forces do not take into account the entire surface. We base our approach on the fast and robust FEM-based method presented in [26]. The system forces are described by

$$\mathbf{f}_j(\mathbf{x}) = \mathbf{R}_j\mathbf{K}_j(\mathbf{R}_j^{\mathrm{T}}\mathbf{x} - \mathbf{x}_0) \tag{2}$$

where $\mathbf{K}_j$ is the stiffness matrix of the element, $\mathbf{x}$ and $\mathbf{x}_0$ the current and the initial positions of the sampling points, respectively. The forces are therefore a simple linear function of the displacement $(\mathbf{x} - \mathbf{x}_0)$. Matrix $\mathbf{R}_j$, which encodes the rotation of a local frame attached to the element with respect to its initial orientation, is updated at each frame. Because a perfect application of the laws of physics is not necessary here, we speed up computations significantly by precomputing stiffness matrices, as explain in [25].

2.4 Surface Constraint

In order to keep the particles on the implicit surface, most methods [30, 28] project the velocity of each particle along the surface gradient at each time step. In this manner, particles are always moving following the surface, but at the end of a step their positions are not guaranteed to be exactly on the surface. Enforcing this constraint with Lagrange multipliers augments the linear system of Equation 1, and can degrade the conditioning of the system which can result in numerical instabilities [3]. Moreover, the augmented system is no longer guaranteed to be positive definite with the conjugate gradient we use. For these reasons, we prefer to use a filtered conjugate gradient in the same way as [3]. The idea is to project initial residual and successive iterative solutions along the surface tangent plane during the conjugate gradient resolution.

Let an implicit surface be defined as the zero level-set $\{\mathbf{x} \in \mathbb{R}^3, F(\mathbf{x}) = 0\}$ of a field function $F : \mathbb{R}^3 \to \mathbb{R}$, which is positive inside the objects. We constrain the particles in two ways. First, we correct the position variation at each conjugate gradient iteration by projecting it onto the tangent plane of the surface. Secondly, we correct the particle positions directly by projecting them to the surface after the conjugate gradient resolution. The projection of a point $\mathbf{p}$ on the implicit surface corresponds to solving $f_{\mathbf{p}}(s) = 0$ with $f_{\mathbf{p}}(s) = F(\mathbf{p} + s\dot{F}(\mathbf{p}))$. This new position is computed using a Newton-Raphson method. At each step k, the minimized point $\mathbf{p}$ moves along the surface gradient $\dot{F}(\mathbf{p})$:

$$\mathbf{p}_{k+1} = \mathbf{p}_k - F(\mathbf{p}_k)\frac{\dot{F}(\mathbf{p}_k)}{\left\|\dot{F}(\mathbf{p}_k)\right\|^2} \tag{3}$$

This method is very efficient and works well for any field function whose derivative is continuous and nonzero in the neighborhood of the solution. In these cases, very few steps are necessary to find a good projection.

To handle sharp features we apply a second constraint at each conjugate gradient iteration, as in [27]. This constraint forces the normals of the mechanical triangles to be aligned with the potential gradient. We apply

$$\mathbf{p}'_k = \mathbf{p}_k + \frac{1}{\sum_\tau A_\tau}\sum_\tau A_\tau\left[(\mathbf{c}_\tau - \mathbf{p}_k)\cdot\dot{F}(\mathbf{p}_k)\right]\frac{\dot{F}(\mathbf{p}_k)}{\left\|\dot{F}(\mathbf{p}_k)\right\|^4} \tag{4}$$

where A_τ is the area of the triangle τ and $\mathbf{c}_\tau$ its centroid. The summations are taken over all $\mathbf{p}$-incident triangles.

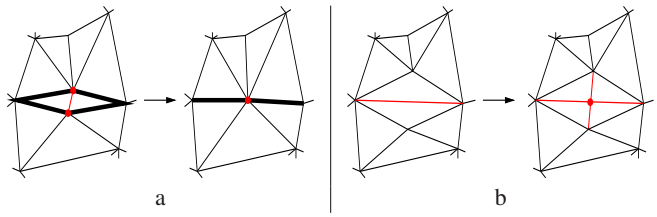


Figure 2: *(a)* Collapse: when an edge is too compressed (in red) and has not stretched since the last time step, the two vertices at the ends of this edge are joined in its middle (red point). It results in the removal of two triangles (in bold) and the welding of two edges. *(b)* Split: when an edge is too stretched (in red) and has not been compressed since the last time step, it is splitted into two parts by inserting a new vertex in its center (red point). It results in the creation of two new triangles and two new edges (in red).

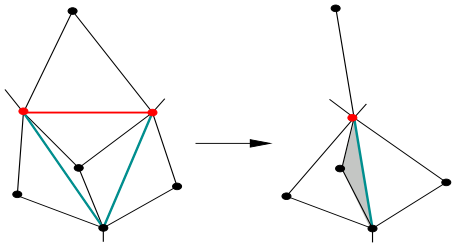


Figure 3: Example of collapsing creating a non-manifold mesh: the red edge is collapsed, and the two green edges become identical, yielding two superimposed triangles (grey). The new green edge has four incident triangles.

2.5 Maintaining Mesh Quality

We enforce the aspect ratio of all the triangles. We set a fixed size, shape and material for all the mechanical elements. As a result, we obtain only one initial shape $\mathbf{x_0}$ and one stiffness matrix $\mathbf{K}$ for all the elements, which can be precomputed. So for an element j, forces applied to its vertices are $\mathbf{f}_j(\mathbf{x}) = \mathbf{R}_j\mathbf{K}(\mathbf{R}_j^{\mathrm{T}}\mathbf{x} - \mathbf{x}_0)$, where $\mathbf{K}$ and $\mathbf{x}_0$ are the same for all elements. The common rest shape is computed as an equilateral triangle with edge lengths $\hat{e}$ given by the user.

When the surface is deformed, stretching of elements can pass over a threshold e_{max}, so new triangles must appear, and some triangles must disappear when elements are too compressed (threshold e_{min}). The mesh is modified using edge collapse and edge splitting, as shown in Figure 2. As in previous works, criteria for mesh changes are based on geometrical information (that is, edge lengths $\| e \|$). This information is faster to compute than information based on mechanics (such as deformation directions). For this, we loop on edges and check their length $\| e \|$, as explained in Figure 2. As in [28], we collapse or split an edge if $\| e \| > e_{max} \times \hat{e}$ *and* $\| e_t \| - \| e_{t-1} \| > \varepsilon_e \times \hat{e}$ or $\| e \| < e_{min} \times \hat{e}$ *and* $\| e_t \| - \| e_{t-1} \| < \varepsilon_e \times \hat{e}$, respectively. At each collapse or split, all modified edges are flagged as unmodifiable for the next time step. This avoids oscillation during the search of equilibrium state with the new created mesh. In certain cases, collapsing can generate a non-manifold mesh [12], where an edge is shared by more than two triangles (see the example in Figure 3). In these cases, collapsing is forbidden, to ensure the mesh is always manifold. This mesh alteration, which loops on edges and splits or collapses them if necessary, is called before the quasi-static integration step.

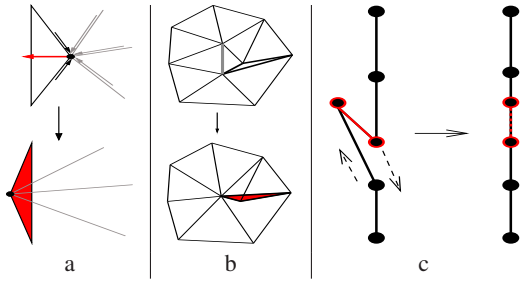


Figure 4: *(a)* Inversion resulting from a discrete integration. At time t, the gray forces are stronger than the black ones and push the point inside the triangle. At time $t+dt$, the point is pushed too far, resulting in an inversion. *(b)* Inversion resulting from collapsing the gray edge: the bold triangle becomes inverted (red). *(c)* Forces can create wrinkles that give inverted triangles when projected to the surface (red).

2.6 Handling Triangle Inversions

An inverted element is a triangle facing the inside of the object instead of the outside. We consider that a triangle $(\mathbf{a},\mathbf{b},\mathbf{c})$ of normal $\mathbf{n}$ is inverted if

$$\mathbf{n}\cdot\dot{F}(\mathbf{a}) > 0 \textit{ and } \mathbf{n}\cdot\dot{F}(\mathbf{b}) > 0 \textit{ and } \mathbf{n}\cdot\dot{F}(\mathbf{c}) > 0$$

We found several cases that yield such inverted triangles:

1. discrete integration can let points cross an edge and create a fold, as shown in Figure 4*a*,
2. collapsing an edge may invert a neighboring triangle (see Figure 4*b*),
3. the constraint may invert the triangles when projecting the points to the surface (see Figure 4*c*),
4. splitting an edge belonging to an inverted triangle inherently inserts new inverted elements,
5. user interaction or surface animationmay modify particles positions too fast for the mechanical system to follow them.

If the mechanical system allows such inversions as a rest state, as is the case with a mass-spring system, they will hardly be removed, and may even expand when the splitting criterion is satisfied. We address this problem by two means: trying to have as few inversions as possible, and remove them when they appear.

2.6.1 Preventing Inversions

Our FEM is better at avoiding inversions than a mass-spring system since the forces increase much more when a triangle is compressed and approaches an inversion state (case 1, Figure 4*a*). Moreover, we can keep the elements in a stretched rather than compressed state by simply setting the split and collapse criteria e_{max} and e_{min} to 2 and 1, respectively. This means the system is always stretched and never makes wrinkles that could lead to inversions (case 3, Figure 4*c*). The result mimics in fact a particle system approach in which particles attract each other instead of repulsing each other. We avoid collapsing an edge if doing so creates an inversion (case 2), and we forbid splitting of edges belonging to an inverted triangle (case 4, Figure 4*b*). Despite these measures, case 5 can still create inversions which are handled as explained below.

2.6.2 Recovering from Inversions

Forces of inverted elements are modified so that they fall back to an uninverted state, mixing the elegant approach of [17] and the fast approach of [26]. The local frame presented in [26] is chosen such that its third vertex (assimilated as the inverted one) has the smallest height as in [17] (see Figure 5). The displacements of the inverted element j then become

$$\mathbf{R}_j^T\mathbf{x}-\mathbf{x}_0=\begin{bmatrix}0\\0\\v^x-v_0^x\\0\\w^x-w_0^x\\-w^y-w_0^y\end{bmatrix}$$

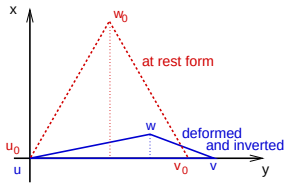


Figure 5: An element known as inverted and its rest form in their local frame

where $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ are the coordinates of the first, second and third vertex of the element j in the local frame given by the rotation $\mathbf{R}_j$, respectively. With these displacements, forces are computed classically as in Equation 2, so they do not require more computations than in case of a non-inverted element. Only the inversion detection needs one more evaluation of the function gradient.

3 The Geometric Mesh

After each update of the mechanical system, we create an initial geometric mesh by copying the mechanical one. We then refine it where the curvature is higher than a user-defined value α_r. We approximate the curvature by comparing the normals of both adjacent vertices and adjacent triangles. If the angle between the normals of two vertices of an edge is higher than α_r, this edge is marked for refinement. If the angle between the normals of two neighboring triangles is higher than α_r, the other edges of both triangles are marked for refinement. The normals are computed by measuring the gradient $\vec{F}$ of the field function at the vertices and triangle centroids. For each edge that needs to be refined, we create a new point at its center and project its position onto the implicit surface using the Newton-Raphson technique described in Section 2.4.

Depending on how many edges of a triangle need refinement, we replace it by two, three or four smaller triangles as shown in Figure 7*a*. In the case where two out of the three edges of a triangle need to be refined, two solutions exist. We choose the one that best fits the local curvature direction. This refinement pass is repeated several times in order to reach the desired level of detail. The number of passes is limited to a user-defined value N_p to avoid infinite refinement in sharp areas. Note that if α_r or N_p are set to the minimal value (that is $-\pi$ and 0, respectively), the geometric mesh is an exact copy of the mechanical mesh. Example results can be seen in Figures 6 and 8.

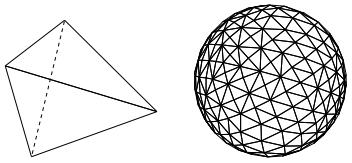


Figure 8: Geometric refinement on an extremely coarse mesh: although the metaball is mechanically represented by only four triangles (*left*), the refinement with $\alpha_r = 20°$ gives a well detailed mesh (*right*).

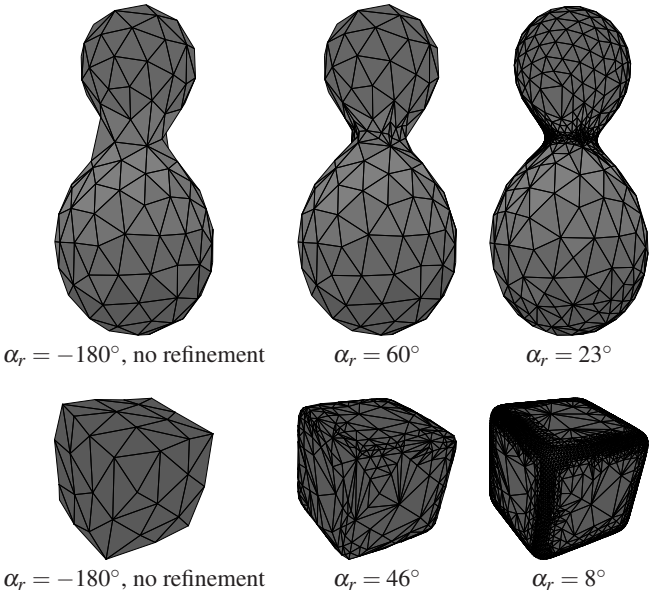


Figure 6: Results of geometry refinements with varying α_r.

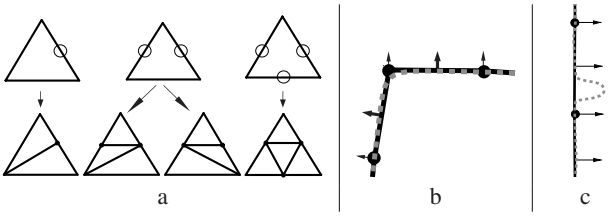


Figure 7: *(a)* Refinement scheme. Marked lines on the top row are to be refined. Case 2 has two refinement solutions. We choose the one that best fits the direction of curvature. *(b)* Refinement on a highly curved edge: a criterion based only on the vertex normals would not refine the top triangle. Our criterion also compares the triangle normals and refines both triangles. *(c)* Limitation of using sampled normals as refinement criterion: since all normals are aligned, the geometry will not be refined to sample the hernia.

Using triangle normals is helpful in cases of sharp or highly curved features. In these cases the normals at the vertices located on the edges may be wrong (see Figure 7*b*). The main limitation of this choice is that the criteria may miss areas were refinement should happen, *e.g.,* see Figure 7*c*. If one does not want to miss these regions, either a finer mechanical resolution must be chosen or a different criterion must be used, such as one based on the Hessian of the field function [10]. Since we evaluate the gradient at the particle locations *and* at the triangle centers, we are less likely to miss a refinement area than other approaches, such as [28], which take only the particle normals into account.

The refinement procedures and criteria described here are not inherent to our method. Any other one can be used. For example, a red-green refinement scheme [5, 2] can be preferred over ours. The one we chose allows for an T-vertex-free, anisotropic geometry in which the anisotropy suits the curvature directions.

Constant	Meaning	Value
E	Young's modulus (for finite element method)	100–3000 N
ν	Poisson's ratio (for finite element method)	0.3
k	Springs stiffness (for mass-springs comparison)	100
dt	Time step	0.04 s
e_{min}	Maximal edge compression (Section 2.5)	$0.6-1$
e_{max}	Maximal edge stretching (Section 2.5)	$1.2-2$
$\hat{e}$	Desired mechanical edge length (Section 2.5)	user-defined
ε_e	Limit for test on edge length variation (Section 2.5)	0.1
α_r	Max. angle between normals of geometric points (Sec. 3)	user-defined
N_p	Max. number od refinement passes (Sec. 3)	≈ 4

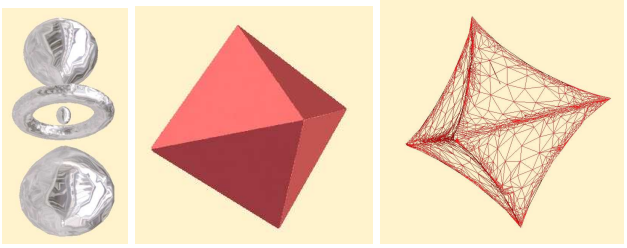
Table 1: Used constants

4 Results

The method was run on a Pentium M 2GHz with 2GB of RAM and a nVidia Quadro FX Go1400, using the constants of table 1. Note that only two parameters must be user-defined. We tested our method on several classes of implicit surfaces. Figures 1, *right*, 6, 8, *right*, 10 and teaser show skeleton-based implicit surfaces with $\frac{1}{d^2}$-like field functions, d being the distance to the skeleton element. We used different norms for d, such as $||_{10}$, $||_1$ and $||_{0.6}$ (Figures 6, *bottom*, 9*b* and 9*c*, respectively). Figures 9*a*, 11, 12 show our method on volume data sets.

Our method produces surfaces that are well sampled and the geometric refinement adds details to areas of high curvature. The mechanical mesh is well-shaped, and manages to follow fast movements of the surfaces. Triangle inversions are rare as they only occur when the implicit surface changes very quickly, *e.g.,* with fast manipulation. When they do occur, our method gracefully recovers from them within one or two time steps.

For C^1-continuous field functions, we obtain a stable mechanical solution with vertices lying on the sharp features (Figure 9), thanks to the constraint described in Equation 4.



a b c

Figure 9: C^1-continuous examples: *(a)* Volume data. *(b–c)* Skeleton-based $||_1$-ball and $||_{0.8}$-ball. The system finds a solution even though there are discontinuities in the gradient.

Speed results for different geometric refinement criteria are presented in table 2 and on Figures 10–12. The speed measurements include notably the handling of the mechanical system, the creation of the geometry and the rendering.

We compared our Twinned Meshes approach against a classical adaptive mass-spring solution, using the same surface constraint and solver. In terms of speed, our approach is just as fast as the mass-spring approach for the same number of triangles. However, it is much more robust: it converges more rapidly, and nicely handles inversions. The mass-spring method takes more time to find a solution, thus limiting the speed of possible animations and manipulations. It does not often recover from inversions, leading to useless meshes. These problems sometimes even make the system diverge and explode. Moreover, we are able to achieve very fine levels of triangulation, whereas the mass-spring method is limited in details because of computation time and stability issues. In addition, the improved robustness of the FEM approach allows us to save computation time by modifying the mesh less often and increasing the time step dt.

α_r	37°	26°	18°	8°
Number of geometric triangles	326	1378	2130	12054
Frames per second	52	22	18	7

Table 2: Rendering speeds (in frames per second) for the model shown on teaser with various values of α_r.

5 Conclusion and Future Work

We have presented a method based on two dynamic triangulations of an implicit surface, a *mechanical* one and a *geometric* one. Thanks to this decoupling, we can address the needs for a stable mechanical system and the detail requirements of rendering at the same time.
The mechanical resolution is robust and fast thanks to several features. First, it uses a finite element method where all elements have a common and constant ideal rest state. Secondly, the quasi-static integration avoids accumulated errors. Finally, the treatment of inverted elements allows the handling of degenerate configurations.
By refining the geometric triangulation in areas of high curvature, we generate a mesh that is detailed while keeping the number of triangles low and without adding more strain on the particle system.
As a result, this method provides robust, interactive and detailed rendering of animated or manipulated implicit surfaces.

There are areas in which our method could be improved. For instance, the update of the mechanical system could be frozen in areas where it is at rest and the surface is not changing. Additional or different criteria can be applied on the refinement method, such as maintaining a minimum aspect ratio [18].

We believe the major research topic left in this area is the treatment of topology changes. We started some experiments on that matter that indicate it is possible to handle simple cases of blending and splitting in a dynamic and interactive fashion. Hart *et al.* 's works [29, 15] on this matter can be applied to this method and future work should concentrate on extending it. This would allow particle-based dynamic triangulation of a wide range of dynamic implicit surfaces.

6 Acknowledgments

We would like to thank Benoit Laurent for his early work, Mathieu Coquerelle for the fluid simulation, Gabriel Peyré and the whole EVASION team for their reviews and advices.

REFERENCES

[1] Ricardo S. Avila and Lisa M. Sobierajski. A haptic interaction method for volume visualization. In *Proceedings of the 7th conference on Visualization*, 1996.

[2] Randolph E. Bank, Andrew H. Sherman, and Alan Weiser. *Refinement Algorithms and Data Structures for Regular Local Mesh Refinement*. 1983.

[3] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, 1998.

[4] Adam W. Bargteil, Tolga G. Goktekin, James F. O'Brien, and John A. Strain. A semi-lagrangian contouring method for fluid simulation. volume 25, 2006.

[5] F. Bornemann and R. Erdmann, B.and Kornhuber. Adaptive multilevel-methods in three space dimensions. *Intl. J. for Numer. Meth. in Eng.*,, 1993.

[6] Antoine Bouthors and Fabrice Neyret. Modeling clouds shape. In E. Galin M. Alexa, editor, *Eurographics (short papers)*, august 2004.

[7] Marie-Paule Cani. Implicit representations in computer animation : a compared study. In *Proceedings EUROGRAPHICS Workshop on Implicit Surfaces'99*, 1999.

[8] Patricia Crossno and Edward Angel. Isosurface extraction using particle systems. In *Proceedings of IEEE Visualization '97*, 1997.

[9] Patricia J. Crossno and Edward S. Angel. Spiraling edge: Fast surface reconstruction from partially organized sample points. In *Proceedings of IEEE Visualization '99*, 1999.

[10] Bruno Rodrigues de Araújo and Joaquim Armando Pires Jorge. Curvature dependent polygonization of implicit surfaces. In *Proceedings. 17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 266–273, 2004.

[11] Luiz Henrique de Figueiredo, Jonas de Miranda Gomes, Demetri Terzopoulos, and Luiz Velho. Physically-based methods for polygonization of implicit surfaces. In *Proceedings of the conference on Graphics interface '92*, 1992.

[12] Tamal K. Dey, Herbert Edelsbrunner, Sumanta Guha, and Dmitry V. Nekhayev. Topology preserving edge contraction. *Publications de l' Institut Mathematique (Beograd)*, 1999.

[13] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Practical volumetric sculpting. *The Visual Computer*, 2000.

[14] Markus Hadwiger, Christian Sigg, Henning Scharsach, Khatja Bühler, and Markus Gross. Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum*, 2005.

[15] John C. Hart, Antoinne Durr, and Douglas Harsh. Critical points of polynomial metaballs. In *Proceedings of Implicit Surfaces '98*, pages 69–76, June 1998.

[16] Chien.-Chang Ho, Fu-Che Wu, Bing-Yu Chen, Yung-Yu Chuang, and Ming Ouhyoung. Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum*, 24(3):537–546, September 2005.

[17] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004.

[18] Xiangmin Jiao, Andrew Colombi, Xinlai Ni, and John C. Hart. Anisotropic mesh adaptation for evolving triangulated surfaces. In *Meshing Roundtable*, pages 173–190, 2006.

[19] Takashi Kanai, Yutaka Ohtake, Hiroaki Kawata, and Kiwamu Kase. Gpu-based rendering of sparse low-degree implicit surfaces. In *Proceedings of GRAPHITE 2006*, pages 165–171, 2006.

[20] Florian Levet, Xavier Granier, and Christophe Schlick. Fast sampling of implicit surfaces by particle systems. In *Proceedings of Shape Modeling International*, 2006.

[21] Florian Levet, Julien Hadim, Patrick Reuter, and Christophe Schlick. Anisotropic sampling for differential point rendering of implicit surfaces. In *Proceedings of the Winter School of Computer Graphics'05*, 2005.

[22] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 1987.

[23] Lee Markosian, Jonathan M. Cohen, Thomas Crulli, and John F. Hughes. Skin: A constructive approach to modeling free-form shapes. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 393–400, August 1999.

[24] John Willard Milnor. *Topology from the Differentiable Viewpoint*. University Press of Virginia, 1965.

[25] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002.

[26] M. Nesme, Y. Payan, and F. Faure. Efficient, physically plausible finite elements. In *Eurographics (short papers)*, 2005.

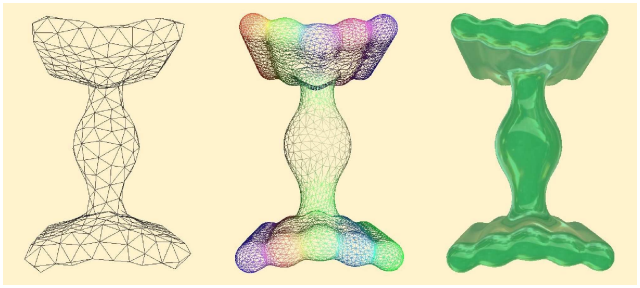
[27] Yutaka Ohtake, Alexander Belyaev, and Alexander Pasko. Dynamic mesh optimization for polygonized implicit surfaces with sharp features. *The Visual Computer*, 19(2):115–126, 2003.

[28] Hans-Christian Rodrian and Hardy Moock. Dynamic triangulation of animated skeleton-based implicit surfaces. In *Proceedings of the EUROGRAPHICS Workshop on Implicit Surfaces'96*, 1996.

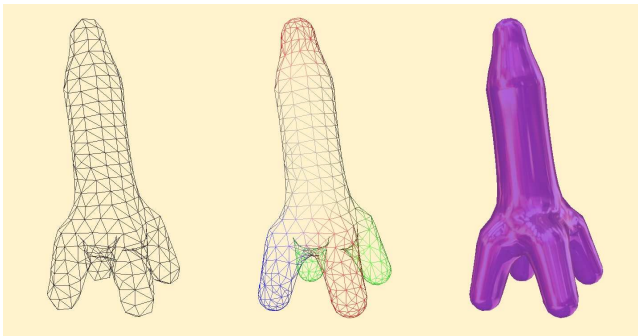
[29] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proceedings of SIGGRAPH 97*, 1997.

[30] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH 94*, 1994.

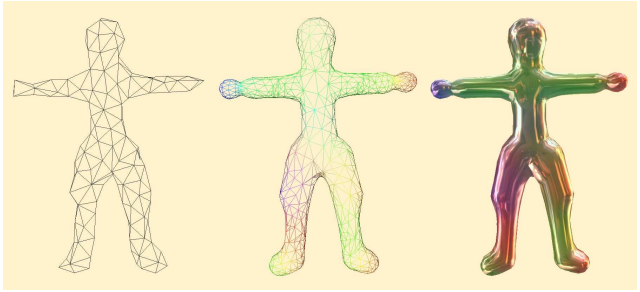
[31] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.



≈ 450 particles, ≈ 900 mechanical triangles, ≈ 15000 geometric triangles
18 skel. elements, $\alpha_r = 18°$, ≈ 3 fps



≈ 390 particles, ≈ 780 mechanical triangles, ≈ 1600 geometric triangles
8 skel. elements, $\alpha_r = 37°$, ≈ 7 fps



≈ 150 particles, ≈ 300 mechanical triangles, ≈ 2000 geometric triangles
14 skel. elements, $\alpha_r = 46°$, ≈ 4 fps

Figure 10: Skeleton-based examples, with a $\frac{1}{d^2}$-like field function. The first column shows the mechanical mesh, the second is the geometric mesh and the last is a textured rendering.

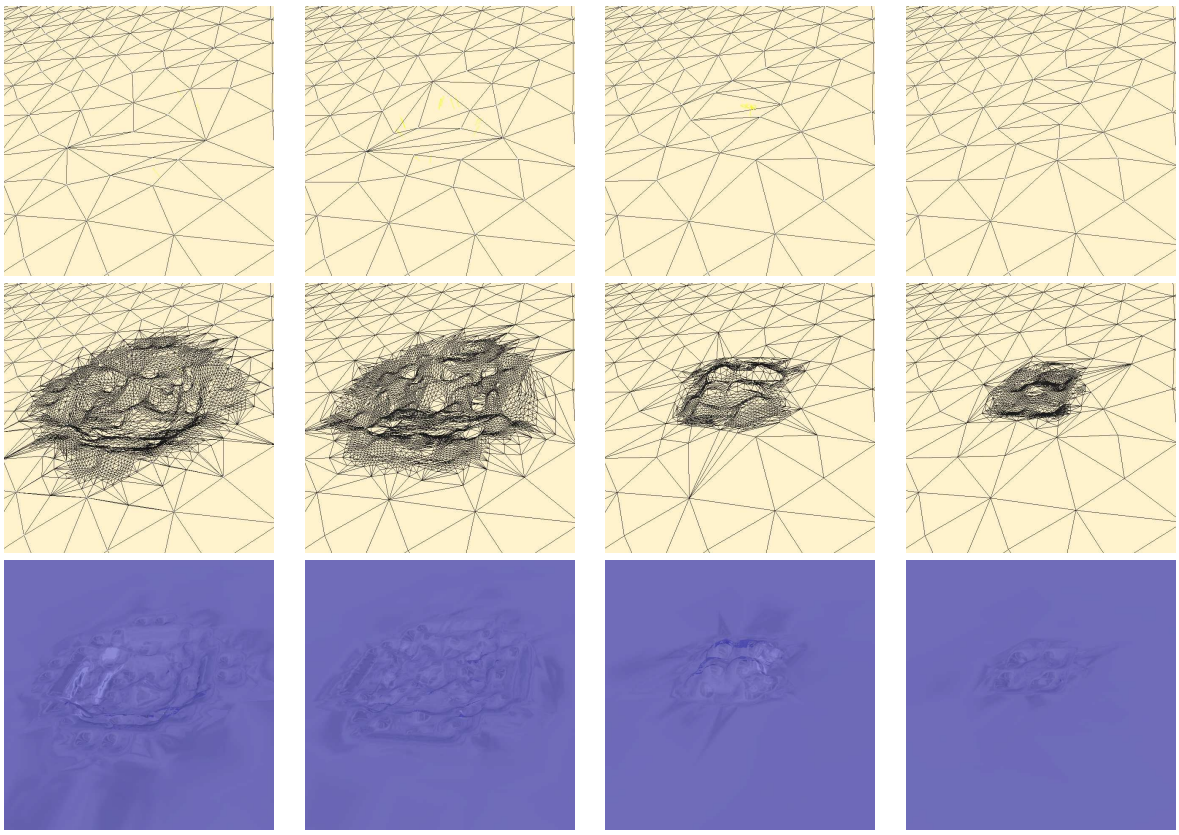


Figure 11: Results on a water-like simulation (an animated grid of potentials). First row: mechanical mesh. Second row: geometric mesh. Third row: textured rendering. $\approx$ 15 fps with simulation

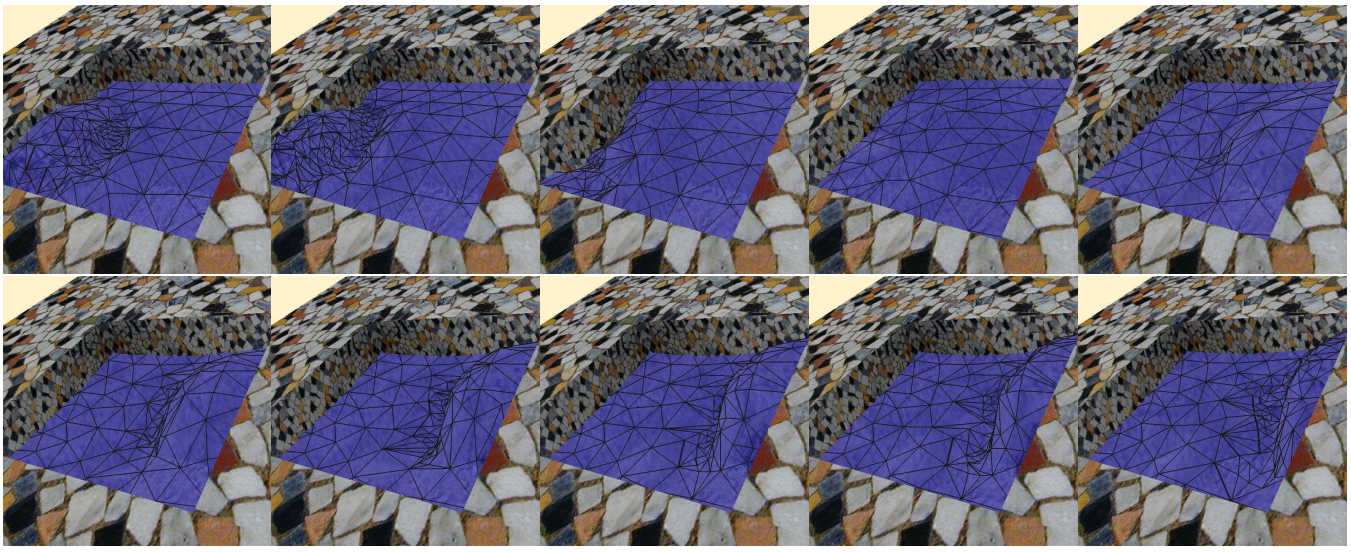


Figure 12: Real time triangulation of an precomputed fluid simulation (an animated grid of potentials) $\approx$ 40 fps