Active Implicit Surface for Animation

Mathieu Desbrun‡ Marie-Paule Cani-Gascuel

iMAGIS/GRAVIR-IMAG[1]
BP53, 38041 Grenoble Cedex 9 - France

‡ Currently at:
Department of Computer Science
1200 E. California Blvd., MS 256-80
California Institute of Technology
Pasadena, CA 91125 - USA

e-mail: {Mathieu.Desbrun,Marie-Paule.Cani}@imag.fr

Abstract

This paper introduces a new model of deformable surfaces designed for animation, which we call active implicit surfaces. The underlying idea is to animate a potential field defined by discrete values stored in a grid, rather than directly animating a surface. This surface, defined as an iso-potential of the field, has the ability to follow a given object using a snake-like strategy. Surface tension and other characteristics such as constant surface area or constant volume may be added to this model. The implicit formulation allows the surface to easily experience topology changes during simulation. We present an optimized implementation: computations are restricted to a close neighborhood around the surface. Applications range from the coating of deformable materials simulated by particle systems (the surface hides the granularity of the underlying model) to the generation of metamorphosis between shapes that may not have the same topology.

Résumé

Nous présentons dans ce papier un nouveau modèle de surface déformable pour l'animation en Synthèse d'Images, appelé surface implicite active. Cette surface est définie comme une isopotentielle d'un champ discret, dont on modifie les valeurs afin d'animer la surface en retour. Le comportement de la surface permet de suivre le contour d'un objet à la manière des snakes en vision, tout en simulant une tension de surface et/ou un volume constant. La formulation implicite permet tout changement de topologie. Nous proposons une implémentation optimisée, où les calculs sont restreints à un voisinage proche de la surface. Les applications peuvent s'étendre de l'enrobage de systèmes de particules, où la surface permet de dissimuler la granularité de la simulation, à la création de métamorphoses automatiques entre objets de topologie quelconque.

[1] iMAGIS is a joint project between CNRS, INRIA, INPG and UJF.

1 Introduction

Implicit surfaces have been progressively used in modeling and animation during the past decade. Their advantages for modeling purposes are numerous, from ease of use to compact storage or blending properties. Their use in physically based animation is also interesting [1] since they efficiently handle collision detection and topology changes with their volumetric representation (*V-rep*).

Physics-based particle systems provide a typical example of the capabilities of implicit surfaces. As they are discrete, particles are not sufficient to render a deformable body, but they can be "coated" with an implicit surface that will be used for display [13, 19, 20]. This is done by associating a field contribution, or *potential*, with each particle, and then defining the body's surface as an iso-surface for the sum of these fields. In this way, the object has a well-defined volume, and its shape is controlled by (and *only* by) the positions of the particles. A further improvement is to use this coating to detect collisions, to model precise contacts, and to provide other advantages such as the preservation of volume during the animation [9, 5]. The particles are then really "fleshed" with the implicit layer, which is no longer used solely for visualization.

Nevertheless, this use of implicit surfaces as a *purely geometric coating* (as directly defined from particle positions) is often insufficient. Indeed, the surface of a deformable body should exhibit specific properties such as surface tension. Moreover, a new trend in animation is to use levels of detail to optimize computations [3]. If such an internal physics-based model suddenly changes, so will a purely geometric coating. This popping artefact, betraying the internal model, can be avoided if an *active* surface is defined.

This paper presents an active surface model, based on an implicit formulation in order to allow any topological changes and fast inside/outside detection. Similarly to snakes in vision, the surface (or *skin*) can track a target and/or exhibit surface tension under its own inertia. A discrete formulation allows the model to be efficient both in motion and display computations. This active coating may have different applications, providing a more complex behavior than conventional implicit surfaces while keeping all of their advantages.

The remainder of the paper is organized as follows: Section 2 briefly presents previous approaches for deformable surfaces in computer graphics. We present our active implicit surface model in Section 3, while Section 4 details the implementation. We propose extensions of this model in Section 5, before showing applications to the coating of particle systems and to metamorphoses in Section 6. We conclude in Section 7.

2 Background on active surfaces

In this section, we briefly review the two main approaches for the animation of a deformable surface with physical properties.

2.1 Lagrangian approach

A first method to simulate a surface is to discretize it into a set of finite mass elements, in a Lagrangian approach. Various physics-based methods, such as mass-spring networks or particle systems, can be used to compute its evolution over time [18, 11, 21]. The main problem with these techniques appears when the surface deforms significantly: discretization points may be lacking in some regions of the surface. As a consequence, topology changes such as separations or fusions are difficult to handle. Fixed lattices of mass elements indeed need delicate updating stages when a separation occurs, whereas particles systems, easily adapting to those changes [24], do not provide the interactive polygonization that is needed for display.

2.2 Eulerian approach

Rather than discretizing matter and following the evolution of each mass element, the Eulerian approach consists of partitioning space into a fixed grid that encodes the presence of matter. Motion is produced when some amount of matter moves from one grid node to another. While this technique is particularly suited to simulation of fluid transport [6, 7], it has not been used, to our knowledge, for surface animation. Yet using a fixed grid offers a relevant property: discretization node relationships are automatically known, so that each grid node can access its six neighbors instantly at any time with no need for compute-intensive updates. This property, making the approach robust to topology changes, seems attractive, especially if we combine it with implicit surfaces.

3 Active Implicit Surfaces

In this section, we present our method for developing an active implicit surface with a Eulerian-type approach.

3.1 Introduction of an evolving potential

The formal definition of an evolving implicit surface $S(t)$ requires the use of an evolving potential f :

$$S(t) = \left\{ \mathbf{X} \in \mathbb{R}^3 / f(\mathbf{X}, t) = 0 \right\} \tag{1}$$

In the remainder of this paper, we adopt the following conventions: we define the interior of an implicit object as the part where the potential f takes its positive values. We also define the normals of an object as being directed outward. Thus, the normal vector is: $\mathbf{n} = -\nabla f / ||\nabla f||$.

3.2 Discrete potential

In the spirit of Eulerian approaches, we can think of using a *discrete potential*, with values stored on a regular grid. Contrary to most of the implicit surfaces used in modeling and animation, the potential has no analytic formulation, just sampled values. A continuous potential can be constructed from it through tri-linear interpolation for instance.

This discrete formulation offers several advantages. First, the evaluation cost of the potential of a point in space can be performed in constant time, *independently of the surface complexity*, via a basic interpolation of the closest values. Secondly, finding sample points on the resulting surface is quite straightforward compared to general implicit surfaces: Since we have at hand a grid of potential values, a simple spatial partitioning method (also called Marching Cubes) [25, 10] can be used. We choose a tri-linear interpolation between potential values so that the null potential between two adjacent grid nodes of opposite values can be found in a simple linear interpolation, without any need for compute-intensive root-finding methods. Moreover, such a sampling method provides a trivial polygonization of the surface from the obtained sample points. Lastly, a further advantage given by this discrete potential is *local control* of the surface. Indeed, we can act on a single potential value to affect the surface slightly and locally, without making the formulation more complicated.

3.3 Differential equation of the potential

As the evolving potential characterizes a moving isosurface, it follows a simple differential equation. Assuming that the path of a point $\mathbf{X}(t)$ during the evolution of the surface satisfies $f(\mathbf{X}(t), t) = 0$ for any time t, it yields:

$$\frac{df}{dt}(\mathbf{X}(t), t) = \frac{\partial f}{\partial t}(\mathbf{X}(t), t) + \nabla f(\mathbf{X}(t), t) \cdot \frac{d\mathbf{X}(t)}{dt} = 0 \tag{2}$$

A way to animate the implicit surface $S(t)$ is then to define a *velocity field*, the only unknown quantity in the equation, and integrate Equ. (2) over time. This technique has already been used for interactive sampling purposes [24], as well as for texture animation [17] and computer vision [12, 23].

Controlling the evolution of the surface thus consists in defining at each surface point an instantaneous motion speed. Through integration of Equ. (2), potential values are modified so that the surface experiences the desired motion. This *indirect* control of the surface motion via the potential allows us to deform the surface while maintaining the implicit formulation.

3.4 Making the implicit surface active

To finally give "life" to the surface, we only need to define a velocity field that will impose a behavior to the surface. We call this definition *motion strategy*.

Tracking another iso-surface

The implicit skin should be designed to cover an internal physics-based model such as a particle system. Using a purely geometric implicit surface as in [2], that naturally defines a *boundary* between the inside and the outside, is not sufficient in the general case since it cannot simulate surface tension and may also give rise to visible temporal discontinuities if the internal model switches between different levels of detail [3].

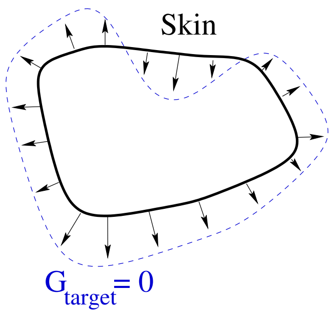
However, the geometric coating can be used as a *target*: the skin will always be *attracted by this boundary*. In this way, we get around the problem of purely geometric surfaces by simulating a deformable surface tending towards a given surface. The corresponding motion strategy can be simply written as follows:

$$\frac{d\mathbf{X}_{target}}{dt}(t) = \alpha \left(G_{target}(\mathbf{X}) - iso\right) \mathbf{n}(\mathbf{X}), \qquad (3)$$

where α is a scalar, G_{target} the potential corresponding to the target surface, and iso the chosen isopotential for G_{target}. It intuitively means that the skin will locally inflate if inside the object, and deflate if outside, so that the skin will always try to fit the boundary. This strategy is closely related with *balloon snakes* in computer vision [4].

Adding surface tension

Modifying the strategy can confer surface tension to the skin. Analogously to an elastic skin under tension, surface tension is mimicing with a penalty term to minimize



curvature. The strategy can then be written:

$$\begin{aligned} \frac{d\mathbf{X}}{dt}(t) &= \frac{d\mathbf{X}_{target}}{dt}(t) + \frac{d\mathbf{X}_{tension}}{dt}_{tension}(t) \\ &= \alpha \left(G_{target}(\mathbf{X}) - iso\right)\mathbf{n}(\mathbf{X}) + \beta\, \kappa(\mathbf{X})\, \mathbf{n}(\mathbf{X}) \\ &= \left[\alpha \left(G_{target}(\mathbf{X}) - iso\right) + \beta\, \kappa(\mathbf{X})\right] \mathbf{n}(\mathbf{X}) \end{aligned} \qquad (4)$$

where κ is the mean curvature, and β a coefficient allowing us to tune the weight of surface tension. With this last equation, we constrain the skin to follow an iso-surface while minimizing curvature at each point. It thus tends to converge to a smoothened version of the target surface.

The motion strategy we defined is valid for any potential G_{target}, although its complexity will affect the cost of the simulation. G_{target} does not even have to be C^0, since the skin will be smoothed by surface tension. As the target may also be moving, we should formally write $G_{target}(\mathbf{X}, t)$; but we omit the time dependency whenever possible for clarity.

4 Implementation

Once the main ideas have been set up, we must choose the implementation carefully in order to offer good performances. A complete and precise mathematical framework for implementing Hamilton-Jacobi differential equations such as Equ. (2) can be found in [16]; while this theory is now well elaborated, we have developed another approach that offers a fast solution to this problem.

To alleviate computation, we propose a number of simplifications that will result in an optimized implementation of our general approach.

4.1 Simplifications

A skin being only a surface, it seems logical to restrict the discrete potential to a nearby area around the current surface.

Close neighborhood

As Velho and Gomes pointed out in [22], the fundamental characteristics of a potential are its variations around the implicit surface it defines. Outside of this *close neighborhood* (or enveloping volume) of its surface, potential values have no influence on the surface itself.

In our case, where the potential is discrete, this neighborhood represents a set of nodes of the grid all around the surface (see Fig. 1). The information stored in these nodes allows us to deduce the potential gradient on the surface with finite differences. It then appears that integrating the variations of f over time on the whole grid is not necessary: we just have to *propagate a front with a width of a few voxels* to have at hand only the information strictly necessary to the evolution of the surface.

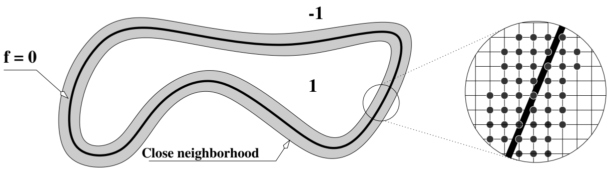


Figure 1: Close neighborhood of a surface ($f = 0$).

Restricting computations to a close neighborhood of the surface will accelerate both the motion and polygonization calculations. The theoretical algorithmic cost for the global animation of the surface turns out to be proportional to the covered area, resulting in a mean complexity in $O(n^2)$ for a grid of size $n \times n \times n$.

Thresholding the inside and the outside

Regions out of the surface neighborhood are not relevant for the surface motion. Nevertheless, we must keep track of whether they are inside or outside the skin, for collision detection purposes. We then threshold the potential as suggested in [23]: every node of a potential value greater than 1 is thresholded at 1 and in turn, considered as interior. Reciprocally, every node of value less than -1 is thresholded at -1 and considered as being outside of the skin. The choice of the threshold value is arbitrary: any other value would have changed the scale of the discrete potential, but not the surface.

4.2 Data structures

Coding the potential and the surface neighborhood

The potential being a set of discrete values on a 3D grid, we store these data in a $n \times n \times n$ array, where n is chosen according to the desired sharpness of the polygonization. Observe that for each node, we can access to its six neighbors in constant time.

In order to quickly access relevant nodes, which are those lying in the neighborhood of the current surface, we employ a simple linked list, where each element points to a corresponding node in the potential grid.

A run-length encoding data structure may also be feasible, which would reduce the data memory size while keeping the same efficiency.

4.3 Reliable integration

The motion strategy defined in section 3.4 is valid only for points *on* the skin surface. But the integration must take place at all the grid nodes in a neighborhood around the surface: therefore, we must extend the velocity field notion.

Oscillation with nave integration

A naïve integration of Equ. (2) on all the nodes close to the surface gives rise to inevitable oscillations. The velocity $\frac{d\mathbf{X}}{dt}$ is actually zero only when the node $\mathbf{X}$ is right on the target surface, which is a quite rare case. Thus in general, the skin will never come to a rest state, keeping on oscillating around the target. The reason is that both the differential equation and the velocity field are formulated only for surface points.

To solve this difficulty, it seems judicious to extend the motion strategy at each grid node *according to the closest point on the surface*. In this way, if the skin passes right on the target iso-surface, the neighboring grid nodes will be assigned a zero speed. It all comes down to move the neighboring isopotentials at the same speed that the iso-surface, as depicted on Fig. 2: it is exactly the prescribed method of Malladi in vision [12], which was not used in practice as it raised a new problem. How indeed can we find the closest surface point rapidly?

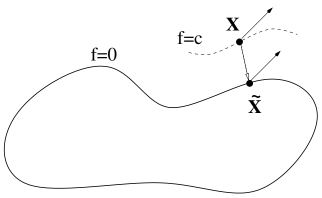


Figure 2: Use of the closest point (after [MSV95])

Evaluation of the target potential on polygonal mesh

Instead of finding the projection $\tilde{\mathbf{X}}$ of a grid node $\mathbf{X}$ on the current surface, we propose to approximate $G_{target}(\tilde{\mathbf{X}})$ using local considerations. During the previous polygonization of the skin, if some voxels around the grid node $\mathbf{X}$ were intersecting the surface, a set of p points $\{\mathbf{P}_i\}$ on the current surface have been determined through linear interpolations (see Fig. 3). We then replace the evaluation of $G_{target}(\tilde{\mathbf{X}})$ by:

$$\frac{1}{p} \sum_{i=1..p} G_{target}(\mathbf{P}_i).$$

This approximation can easily be implemented, but requires up to six target potential evaluations for each grid node. By stamping already computed potential values, we avoid redundant computations and the whole process only requires as many target potential evaluations as the current number of sampling points available for the skin. In the case where a node has no sampling point in its immediate neighborhood, we just evaluate $G_{target}(\mathbf{X})$ without giving rise to problems : the grid node is "far" enough from the surface for that approximation to be valid.

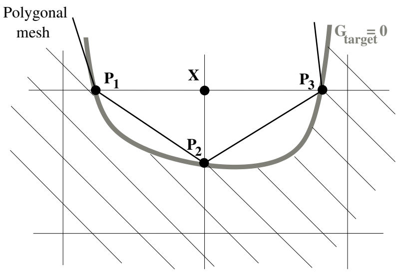


Figure 3: Shifted evaluation using available sample points

With this evaluation process, the algorithm appears to be reliable without the need of artificial viscosity. The potential evaluation, shifted from a grid node to nearby sample points on the current polygonal mesh, introduces a filtering that is sufficient for our needs.

Surface tension handling

To include surface tension to our motion strategy as explained in section 3.4, we must first express the local mean curvature of the skin. Its surface being defined implicitly, the expression for the mean curvature becomes:

$$\begin{aligned}\kappa(\mathbf{X}) &= div\,(\mathbf{n}(\mathbf{X}))\\ &= div\,(-\tfrac{\nabla f(\mathbf{X})}{\|\nabla f(\mathbf{X})\|})\\ &= -\big(f_x^2(f_{yy}+f_{zz}) + f_y^2(f_{xx}+f_{zz})\\ &\quad +f_z^2(f_{xx}+f_{yy}) - 2f_xf_yf_{xy} - 2f_yf_zf_{yz}\\ &\quad -2f_xf_zf_{xz}\big)\,/(f_x^2+f_y^2+f_z^2)^{3/2}\end{aligned} \tag{5}$$

where $f_x, f_{xy}, f_{xx}, \ldots$ represent partial derivatives approximated at $\mathbf{X}$ with finite differences.

Once these mean curvatures have been computed for all grid nodes in the neighborhood of the surface, we can add to the previous evaluation the surface tension term $\beta\ \kappa(\mathbf{X})$ for each grid node. A more sophisticated approximation appeared unnecessary, contrary to the advective term as already reported in [16].

4.4 A quick note on normals

Slight shading artifacts, resembling a bump mapping effect, can arise if no surface tension is set. Indeed, even if the polygonization is adequate, the normals of the skin are computed by linear interpolation [10] between finite differences approximations: small errors are inevitable. To get rid of the problem, we can think of a post-process for smoothing. But a small amount of surface tension proves to result in the same effect : it smoothes out the normals and the surface appears correct.

4.5 Updating the surface neighborhood

We must cope with the dynamic update of the nodes supposed to be in the current neighborhood of the surface. That is to say, we must find out when a grid node must be added to the linked list of neighboring nodes, or, conversely, when a node must be removed from it.

A node must be "awakened" when the surface comes to a close neighborhood of one of its neighbors. An easy implementation can then be done during the evaluation of the velocity $d\mathbf{X}/dt$ of a node: if the surface goes through one of the adjacent voxels, we check that each neighbor node is already active, and we wake up those which are not. In this way, nodes are available just in time as the surface propagates.

As for deletion, it can take place during the thresholding stage: as soon as a potential value drops below -1 or goes above 1, the corresponding node is both thresholded and removed from the linked list. We can also remove nodes too far away from the surface, as being no longer relevant: it allows us to maintain only a small number of nodes around the surface whatever the slope of the target potential. With these easy tests, the linked list remains up to date at a low processing cost.

4.6 Global algorithm

The global algorithm can be implemented in two passes: we evaluate the velocities of each nodes in the linked list, and then update them all. The following pseudo-code sums up the process:

At each time step dt,
- For all points $\mathbf{X}$ of the grid stored in the linked list
  - Calculate $\nabla f(\mathbf{X}, t)$ with finite differences.
  - Evaluate $\frac{d\mathbf{X}}{dt}$ as explained in sections 4.3 and 3.4.
  - Deduce $\frac{\partial f}{\partial t}(\mathbf{X}, t) = -\nabla f(\mathbf{X}, t) \cdot \frac{d\mathbf{X}}{dt}$
- For the same points $\mathbf{X}$
  - Update the potential values :
    $f(\mathbf{X}, t+dt) = f(\mathbf{X}, t) + \frac{\partial f}{\partial t}(\mathbf{X}, t)\, dt$
- Visualize the surface by polygonizing modified voxels.

4.7 Time stepping

The last point to mention is the time stepping required to ensure a stable and precise integration.

Without surface tension

The differential equation (2) is purely hyperbolic when the surface tension coefficient β is zero. The adequate time step must follow the Courant-Friedrichs-Levy criterion [14], written in our case as:

$$dt \leq \frac{\Delta x}{V} \tag{6}$$

where Δx represents the size of spatial discretization, and $V = \max || \frac{d\mathbf{X}_i}{dt} ||$ is the nodes' current maximal velocity. This criterion is typically analogous to the Shannon theorem, as it stipulates a time step inversely proportional to the maximal speed.

We can observe here that this criterion also guarantees a displacement smaller than the size of a grid voxel, ensuring that our algorithm will be robust as the surface will not pass over a node in a time step.

With surface tension

The introduction of a surface tension, involving a curvature term, changes the nature of the differential equation [16]: A parabolic contribution is added. We have chosen the time step as follows:

$$dt \leq \frac{\Delta x^2}{2D}$$

where $D = \beta \max ||\nabla f(\mathbf{X}_i)||$ is the diffusion due to surface tension.

Finally, we use the following time step in our implementation:

$$dt = \min(\frac{\Delta x}{V}, \frac{\Delta x^2}{2D}) \tag{7}$$

5 Properties extensions

The basic model being defined, we can extend the strategy of the active skin model in order to add further properties, such as area or volume preservation.

5.1 Controlling the surface area

During polygonization, we can sum the area of each triangle generated to obtain the global area of the surface. From this approximation $\mathcal{A}(t)$, we are able to define another penalty strategy to ensure an approximate area preservation. This can be done by adding the penalty term:

$$\frac{d\mathbf{X}_{area}}{dt}(t) = \gamma \frac{(\mathcal{A}_0 - \mathcal{A}(t))}{\mathcal{A}_0} \mathbf{n}(\mathbf{X})$$

The skin will thus globally inflate or deflate to keep its surface area at the given value $\mathcal{A}_0$.

5.2 Controlling the inner volume

Similarly, the total volume can be updated at each polygonization. Then, the penalty term that will result in a global volume control can be written:

$$\frac{d\mathbf{X}_{volume}}{dt}(t) = \gamma \frac{(V_0 - V(t))}{V_0} \mathbf{n}(\mathbf{X}).$$

It is interesting to observe here that our *balloon strategy* for the volume control is justified *a posteriori*. Indeed, it has been proved [15] that the gradient of the incompressibility constraint of a meshed object at one of its vertices is directed along the normal to the mesh. Thus, a displacement along the normal of the skin is optimal to control volume.

6 Results

The implicit active surface has been originally designed for visualizing physics-based particle systems, but may also be used for very different applications such as computing metamorphosis between shapes of various topology.

6.1 Skin over particles

Our surface model has been used as an active coating for particle system animation. The target iso-surface is directly defined from field contributions associated with the particles. The surface is animated in order to follow the surface evolving while simulating surface tension, which hides the granularity of the underlying model. Since the amount of surface tension is tunable, different renderings of a given particle animation can be generated (see Fig. 4(g-i), where an excessive surface tension was used without controlling the inner volume). Since particle systems can lead to separation and fusion, the implicit formulation of the skin is critical.

For this test (Fig. 4), there is no retroaction from skin to particles — the skin is not used to detect collisions with obstacles.

6.2 Automatic metamorphosis

Another direct application of the technique described through this paper is the automatic metamorphosis between any shapes. Indeed, if the surface is initialized to a given shape, it will smoothly and autonomously turn into the target shape we have chosen.

The specificity of such a method compared to conventional morphing between, for instance, implicit objects defined by skeletons [8], is that there is no need for specifying a correspondence between parts of the implicit body. The surface turns from one shape to the other one

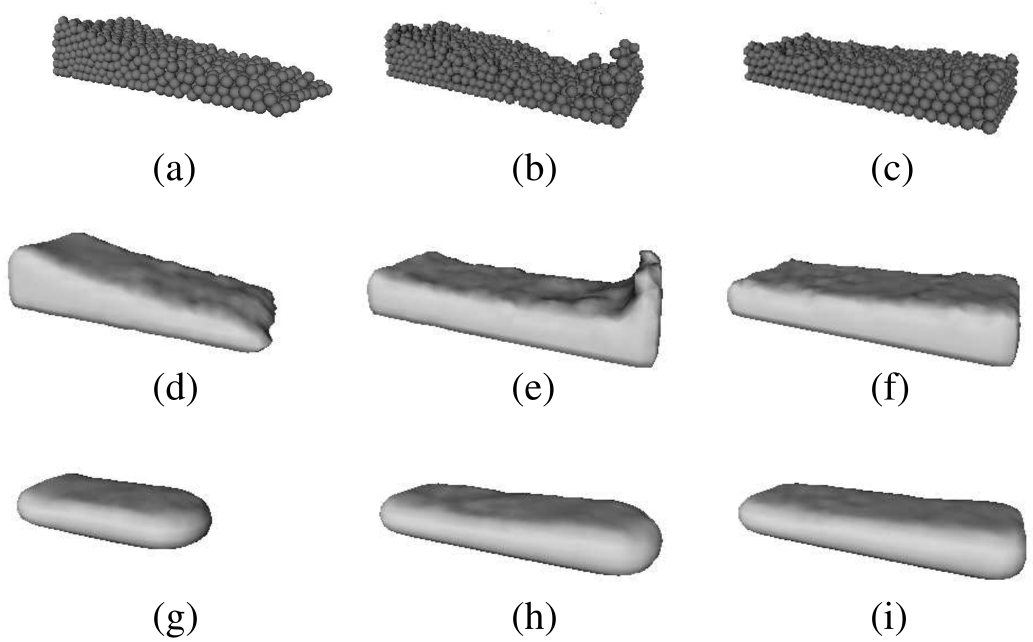


Figure 4: Examples of particle coating, for $\beta = .1$ (d-e-f) and $\beta = .4$ (g-h-i), with a grid size of $40 \times 20 \times 20$.

without any pre-processing time. This is its main drawback too: the user cannot specify a given way to morph between the two shapes.

Fig. 5 shows a metamorphosis without change of topology. The initial object is a warped implicit surface generated by a segment-skeleton, while the target object is analytic. The parameters we use are $\alpha = 2.0, \beta = .01$, and the discrete potential is stored on a $50 \times 50 \times 50$ grid to obtained a detailed representation. The average computation time between two frames is less than two seconds on a *Indy R4600 SGI*.

The second example takes advantage of the ease of dealing with topology changes. We morph, as depicted in Fig. 6, a six point-skeleton implicit surface into two disconnected spheres. Observe that four lobes deflate while the other two seem to move apart. Some intermediate frames exhibits quite sharp edges. The filament between the two spheres gets slowly thin to finally disappears.

7 Conclusions and future work

We have presented a new model for animating a deformable surface with physical properties. A discrete implicit formulation, capable of dealing with arbitrary topological change, is updated according to a motion strategy. Several properties can be simulated such as tracking a given shape, surface tension, or volume preservation. An iso-surface of the discrete field is then polygonized and displayed. In this way, the surface evolves under control of the discrete potential, maintaining the implicit formulation. By restricting computations to a close neighborhood of the current surface, we obtain a rapid and robust way to model evolving implicit surfaces with variable complexity.

Work in progress includes the definition of an adaptive particle system, which refines or simplifies itself according to local deformation. With such an adaptive model, the surface described above would be aa appropriate feature that adds both surface tension to the simulation and smooth visualization despite the ever-changing granularity of the internal model.

Acknowledgements

The authors would like to thank Julie Dorsey and Hans Pedersen as they provided considerable help in sharing their ideas with us and in revising this paper. We are also grateful to François X. Sillion for correction of an early version, and to George Drettakis for the final re-reading.

References

[1] Jules Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July 1997.

[2] Marie-Paule Cani-Gascuel and Mathieu Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, March 1997.

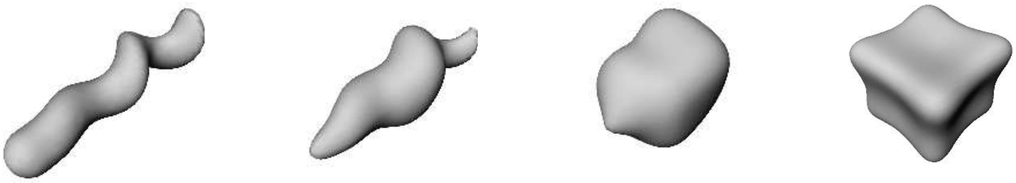


Figure 5: Morphing between two objects of same topology.

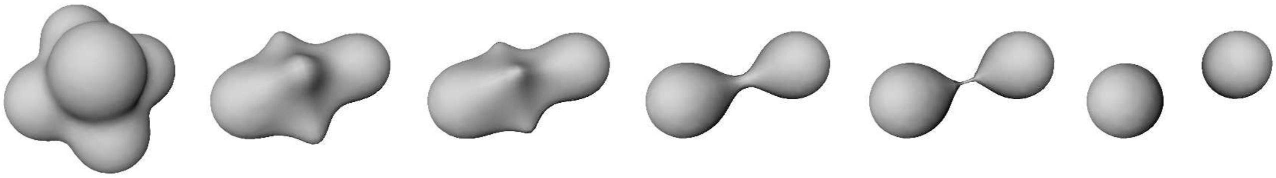


Figure 6: Morphing between two objects of different topologies.

[3] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In *Graphics Interface'97*, pages 1–8, Kelowa, British Columbia, 1997.

[4] Laurent D. Cohen. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing : Image Understanding*, 53(2):211–218, March 1991.

[5] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 287–290. ACM SIGGRAPH, Addison Wesley, August 1995. Los Angeles, CA.

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

[7] Nick Foster and Dimitri Metaxas. Modeling the motion of a hot, turbulent gas. *Computer Graphics*, pages 181–188, 1997. Proceedings of SIGGRAPH'97 (Los Angeles, California).

[8] E. Galin and S. Akkouche. Blob metamorphosis based on Minkowski sums. In *Eurographics'96*, Poitiers, France, August 1996.

[9] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, 27:313–320, August 1993. Proceedings of SIGGRAPH'93 (Anaheim, CA).

[10] William Lorensen and Harvey Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[11] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. A unified view of multitude behaviour, flexibility, plasticity, and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991.

[12] Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, February 1995.

[13] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 89. This paper also appeared in SIGGRAPH'89 Course notes number 30.

[14] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C, second edition*. Cambridge University Press, New York, USA, 1992.

[15] Emmanuel Promayon, Pierre Baconnier, and Claude Puech. Physically-based deformations constrained in displacements and volume. In *Eurographics'96*, Poitiers, France, August 1996.

[16] James A. Sethian. *Level Set Methods*. Cambridge Press, 1996.

[17] Jean-Paul Smets-Solanes. Vector field based texture mapping of animated implicit objects. In *Eurographics'96*, pages 289–300, Poitiers, France, August 1996.

[18] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California).

[19] Demetri Terzopoulos, John Platt, and Kurt Fleisher. Heating and melting deformable models (from goop to glop). In *Graphics Interface'89*, pages 219–226, London, Ontario, June 1989.

[20] David Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface'91*, pages 255–262, Calgary, AL, June 1991.

[21] Russel Turner. Leman: A system for construsting and animating layered elastic characters. In *Computer Graphics- Developments in Virtual Environments*, pages 185–203, Academic Press, San Diego, CA, June 1995.

[22] Luiz Velho and Jonas Gomez. Approximate conversion of parametric to implicit surfaces. *Computer Graphics Forum*, 15(5):327–337, December 1996. A preliminary version of this paper appeared in *Implicit Surfaces'95*, Grenoble, France, may 1995.

[23] Ross Whitaker. Algorithms for implicit deformable models. In *The International Conference of Computer Vision*, Boston, Mass, 1995.

[24] Andrew Witkin and Paul Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, pages 269–278, July 1994. Proceedings of SIGGRAPH'94.

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