Implicit Representations in Computer Animation : a Compared Study

Marie-Paule Cani

Implicit Representations in Computer Animation: a Compared Study

Marie-Paule Cani

iMAGIS[†]-GRAVIR, INRIA Rhône-Alpes
655 avenue de l'Europe, 38330 Montbonnot, France
Marie-Paule.Cani@imag.fr
http://www-imagis.imag.fr/Membres/Marie-Paule.Cani/

Abstract

How can Implicit Surfaces be used in the context of high-end Computer Animation ? This paper compares two different representations of field functions – the constructive approach and the field image approach. Their respective advantages and limitations for the definition of animation and morphing algorithms, and for the visualization of an animation are discussed. We show that efficient solutions to the animation of textured objects can be provided by hybrid methods that combine these representations together and/or with parametric surfaces. This point is illustrated by two case studies: the animation of deformable characters and the simulation of textured lava-flows.

Categories and subject descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling– Solid and object representations, Object hierarchies; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism–Animation, Texture.

Additional Key Words and Phrases: Implicit Surfaces, Constructive soft geometry, Field image representation.

1 Introduction

The use of implicit representations is quickly spreading over scientific visualization, geometric design and computer animation. Implicit surfaces bring several advantages over parametric forms: they ease the construction of smooth surfaces of any topology and geometry, provide an in/out function which can be used for testing intersections, and allow the modeling of large deformations including separations and fusions. These features have proved very convenient, in particular, in several applications of Computer Animation such as 3D morphing, character animation, and soft substances simulation. However, severe drawbacks, such as the difficulty to get an interactive display, and to map 2D textures on implicit surfaces, still prevent this representation from being really usable in high-end applications.

This paper tries to open a discussion: In which cases do we have good reasons for choosing Implicit Surfaces ? Should they be used alone, or combined with a parametric representation ? Should they be defined using a constructive approach (ie. as the successive combination of primitives) or should we merely store a field function sampled on a 3D grid ? Rather than trying to provide general answers, we focus here on the Computer Animation field, where the efficiency of computations is a key point, while the quality and novelty of effects are a constant challenge. In particular, we will discuss the problem of animating deformations, including for objects on which a 2D texture needs to be mapped.

Animating an implicit surface is an intricate process since the surface is defined as an iso-surface of a scalar field over the 3D space. Thus, animating the surface consists in animating the field function, which is a volumetric entity. While a constructive approach to implicit design defines the field function as the successive combination of implicit primitives, most implicit surfaces used in scientific visualization are defined from a 3D field sampled on a grid. The choice of one of these formulations is a key point in our discussion since the modeling, animation, and rendering steps of the process will be quite different. The remainder of this paper reviews these two approaches, compares their benefits for designing animation and morphing algorithms, and for rendering an animation. Hybrid solutions are then discussed in the context of the animation of textured objects that deform over time. This point is illustrated by two case studies, the animation of deformable characters, and the simulation of a visually-realistic lava-flow.

2 Constructive soft geometry versus field images

There are two main ways of representing the field function that defines an implicit surface. The first one relies on a constructive approach, while the second one directly stores a 3D "image" of the field, ie. field values that are sampled over space.

2.1 Constructive soft Geometry

This approach, which is the most widely spread in design and animation applications, consists in successively combining field primitives using various operators such as blends (the simplest of which is summation), set-theoretic operations, and warping functions [25, 5]. Primitives may be defined either analytically, or using functions of the distance to a geometric "skeleton". This last class includes both convolution surfaces [3], for which the distance function is integrated over the skeleton, and distance surfaces such as blobs, meta-balls, and soft objects [37, 4]. An example of object created with this approach is depicted at left of Figure 1.

[†]iMAGIS is a joint project of CNRS, INRIA, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

Figure 1: A column created using the constructive approach, with blends and warps of primitives (left). Two views of a sculpture modeled with Eric Ferley's system, based on the field image approach (right)

2.2 Field images

An alternative approach consists in directly storing the values of the field function sampled over the 3D space i.e. a "field image" (this terminology was introduced in [17]). From this representation, the field can be defined anywhere in space using the trilinear interpolation of the 8 nearest sampled values. This approach has been used for a long time in scientific visualization [18], since many datasets in medical imaging or hydrodynamic flow simulations consist in values defined over a 3D grid. It is also popular in 3D image processing, where "level-set approaches" rely on field images for animating a deformable iso-surface that fits and smoothes scattered data [32]. Lastly, this representation is now emerging in modeling [19, 31, 17, 13] and animation [33, 11] applications. Figure 1 shows an example of object created from a virtual sculpturing system relying on the field image representation [13].

2.3 Evaluation time versus memory cost

Although both representations can be used for modeling and animating complex implicit surfaces, their performances regarding time and memory costs are very different.

The constructive approach offers a compact storage of the field function, since the field is stored as a hierarchical structure where leaves are primitives defined by a few coefficients and nodes are operators. However, each query of the field value at a given point in space requires a full tree-traversal and function evaluation. During a modeling process, the field evaluation time will linearly depend on the number of tree nodes, so it will increase with the complexity of the objet[1], even if the use of bounding boxes around local field primitives can avoid some of the computations.

On the opposite, the field image approach insures that any field query will be answered in almost constant time, whatever the complexity and extend in space of the model. However, storing a full 3D grid of values may require too much memory. As noted in [32, 31, 33, 11], only storing grid nodes that belong to a tubular neighborhood of the iso-surface of interest is sufficient, since other field value do not affect this surface. An efficient solution is to use a search tree for storing the active grid nodes [13]. Alternative solutions would consist in storing them in an octree or in a multi-grid data-structure. In all these representations, we should note that the more complex the data-structure is, the more costly field evaluations will be, since field values at grid nodes need to be searched for in the structure before being interpolated. For instance, with the search tree implementation, the number of nodes that are stored is proportional to the area of the iso-surface of interest (since the tubular neighborhood has a constant extend around this surface), and searching for a node is proportional to the logarithm of this number. For complex model, this approach is still much more efficient than evaluating a construction tree.

A last solution for offering a compact storage of the field image consists in using wavelet compression [31, 17]. In this approach, wavelet coefficients need to be stored instead of field values, and a specific "reconstruction" process is required for answering field queries. The wavelet representation has several advantages: it offers an analytical formulation of the field function, yields compression when the function is smooth, and may be used for multi-resolution rendering of the implicit surface. However, we do not discuss this representation in the remainder of this paper, since it has not been used yet in Computer Animation applications. We can simply guess that since wavelet coefficients need to be attached to fixed space locations, most features of this representation should be the same than for the field image solution.

3 Animation and morphing algorithms

Whatever their representation, implicit surface provide specific features that are very useful in Computer Animation applications:

- they allow the animation of large deformations including topological changes (separations, fusions) which would be very difficult to model using parametric surfaces [35]. This feature is essential both for animation and for morphing applications.

- the use of implicit representations accelerates collision detection, since in-out functions are provided. This makes them a good tool for physically-based simulation. Moreover, contact surfaces between soft colliding objects can be generated, both with the constructive and field image representations [7, 21, 13].

However, in addition to different performances in terms of time and memory cost, the representations of implicit surfaces greatly differ in the way motion and deformation can be defined and controlled. The following sections review these differences.

3.1 Controlling motion and deformations

A first remark is that constructive and image field models are respectively related to the Lagrangian versus Eulerian approaches for simulating motion: while Lagrangian methods follow the motion of the material, such as we do when we animate implicit primitives, Eulerian methods rely on a grid of voxels and capture what goes in and out of each voxel over time. The image field representation, where field values at fixed grid points are edited to model material motion over the grid, belongs to the latter approach.

The constructive representation is very popular in Computer Animation since it stores a structure (typically, a hierarchy of implicit primitives) which can be used for controlling the animation. Any motion or change of parameters of the primitives will immediately result into an adequate motion and deformation of the surface. Constructive soft geometry is thus very easily embedded into a layered model for Computer Animation [7, 6], where the motion of implicit primitives can be linked to any "inner structure" easy to animate (for instance, to an articulated skeleton to perform character animation, or to a physically-based particle system for animating soft substances experimenting separations and fusions).

At the other end of the spectrum, field images provide no structure at all. This does not mean that animation cannot be performed. However, it may be more intricate [33, 11]. Animating the iso-surface implies progressively modifying the sampled field

[1] An alternative to using a lot of simple primitives is to rather use a few complex ones, defined from more general skeletons and/or from anisotropic field functions [8]. The construction tree will then be smaller, but the leaves evaluation may become quite expensive.

function that defines it. This can be done through the numerical integration of differential equations applied to the field values. Let $f(X,t)$ be the time varying field function and $S = \{X \in \mathbb{R}^3 / f(X,t) = 0\}$ be the iso-surface of interest, of normal vectors $n(X,t) = -\nabla f(X,t)/\|\nabla f(X,t)\|$. Suppose the motion we want to apply to the surface is defined by a velocity field $V(X)$ (i.e. a vector field giving the desired speed vector for point $X \in \mathbb{R}^3$). Then, assuming that the path of point X satisfies $f(X,t) = 0$ during motion yields:

$$\frac{df}{dt}(X,t) = \frac{\partial f}{\partial t}(X,t) + \nabla f(X,t).\frac{dX}{dt} = 0$$

The time-variation we should apply to f in order to set $\frac{dX}{dt}$ to $V(t)$ is thus:

$$\frac{\partial f}{\partial t}(X,t) = -\nabla f(X,t).V(t) = \|\nabla f(X,t)\|\, V(t).n(X,t) \qquad (1)$$

where $\nabla f(X,t)$ is easily computed using finite differences. Simple motion strategies such as following an implicit target surface or smoothing the current surface have already been proposed [33, 11]. For instance, following a target surface $T(X) = iso$ that may move over time can be done using:

$$V(t) = \alpha(T(X) - iso)n(X)$$

where $n = -\nabla f/\|\nabla f\|$ is the normal vector to the field image iso-surface. This method was used to render mud-flow simulations while filtering the deformations of a poping iso-surface $T(X) = iso$ defined by a time-varying number of skeleton-points [11, 16].

An idea for providing a more direct control on the animation would be to leave the user directly specify the motion (and thus the velocity vector) of some "control points" on the surface. This approach has been explored in a constructive soft geometry framework [34], where primitive parameters where optimized to make $\frac{\partial f}{\partial t}(X,t)$ obey equation 1 (then, the number of control points had to be directly related to the total number of primitive parameters). Experimenting with a similar approach in the image field case would be promising. Of course, the velocity field needs to be defined everywhere. The use of the smoothing strategy or of a constant curvature strategy for points that are not over the user's control could be a solution.

3.2 Morphing applications

Implicit Surfaces allow easy morphing between 3D shapes of any, and may be different, topologies. A naïve approach consists into using linear interpolation between the field functions defining the initial and the final shapes. However, this generally results into incoherent intermediates shapes. For instance, transforming a man into a rabbit model (ie. metamorphosis between quite "similar" objects) can yield intermediate shapes that are made of several disconnected components (see [2]).

Solutions for ensuring shape consistency during the transformation have been proposed both for the constructive and for the image field approaches.

For implicit surfaces built from a combination of primitives, the best method consists in associating primitives together (thus controlling which part of the objet will morph to a specific part of the target object), and then progressively transforming all the primitives into their target primitives. When primitives are generated by skeletons such as points, line segments, polygons, or polyhedra, the weighted Minkowski sum of the initial and final skeletons can be used as the skeleton at the intermediate states [14]. A specific control may be needed to avoid "amorphous" intermediate shapes: trajectories including translation and rotation are assigned to skeletons, allowing to compute Minkowski sums in local coordinate systems thus avoiding changes of skeleton dimension during the transformation [15]. For instance, if the initial and final skeletons are two non-co-planar polygons, the sum is computed in a rotating plane which moves from the initial to the final polygon planes, yielding a planar Minkowski sum (the direct computation of this sum in the 3D space would have resulted into a polyhedric skeleton).

Performing fully automatic yet consistent 3D morphing is easier using the image field representation. Equation 1 is integrated using the target following strategy of Section 3.1, the target being given by the desired final shape. During the transformation, the initial object locally inflates where the target object is larger, and deflates where it is smaller, thus performing one of the shortest path transforms between the two shapes (see Figure 2, taken from [11]). The process is purely automatic, the only constraint being that the two initially object intersect.

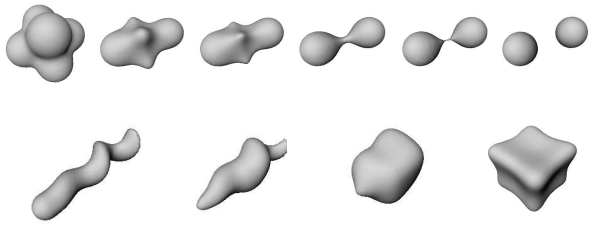


Figure 2: Two morphing examples using the image field representation.

3.3 Constant volume deformations

Controlling the volume of an implicit object during an animation or a morphing process is an important point: constant volume deformations are a key feature for making virtual objects look like real ones. In morphing applications, large volume variations at intermediate stages are not desirable in most cases. Lastly, being able to locally specify some local volume changes during an animation can be very useful in some applications such as character animation. Two solutions were proposed, respectively in the constructive and in the field image methodologies. Both of them are approximate solutions, which is not a problem in practice, since several volume optimization steps can be performed, if needed, between two consecutive animation steps. However, one of them only performs local volume control, while the other one is restricted to global control.

The constructive approach is well suited to the local control of volume. Constructive soft geometry offers no easy way for computing the volume embedded inside the implicit surface. Moreover, when an animation is defined by moving some of the primitives around, volume variations should be corrected *where* the shape of the object is actually changing. The solution proposed [10, 7] relies on the notion of "territories" associated with each implicit primitive. The territory of a primitive is the region of the implicit volume where the field generated by this specific primitive is the higher than any of the others. During an animation, a sampling of territories is maintained over time in order to capture local volume variations. Then, a PID controller is used to tune the primitive "strength" in order to make the implicit surface locally fit the desired volume value.

On the opposite, the image field approach (which does not provide local primitives to be tuned) can easily track global volume changes by counting the number of voxels the iso-surface goes through during an animation (this number is counted positively or negatively depending if the voxel is entering or going out of the

implicit volume). Then, an inflate/deflate strategy can be used for maintaining the volume towards the desired value [11], by adding a penalty term to the velocity field $V(X)$ used in equation 1. We should note that trying to use this strategy locally would not work: the volume of a translating rigid object will be interpreted, in the field image representation, as locally increasing at one side, and decreasing at the other side. These local "volume variations" must not be compensated, since the global volume is not actually changing!

4 Rendering an animation

Efficient rendering is a key point in Computer Animation: Interactive visualization is highly desirable at early design stages, in order to give the animator a good feedback on what he is doing. Moreover, due to the number of images to generate, efficient and good quality final rendering will be needed. Direct ray-tracing of the implicit surface is a solution for performing this last task, although the projective rendering of a polygonisation can give quite good results in much smaller time. The remainder of this section focuses on solutions for getting or trying to get an interactive display of an implicit surface that moves and deforms over time, and for consistently mapping a texture on it.

4.1 Global versus local visualization methods

Usual methods for polygonizing implicit surfaces belong to the spatial partitioning approach [37, 18], which consists into scanning the field function through a fixed 3D grid, and then triangulating the voxels that intersect the surface. This approach is very relevant indeed to render field image models: the set of voxels intersecting the surface and the associated field values at nodes being already computed and stored, polygonisation can be performed efficiently, by just incrementally editing the polygonisation in the regions where the iso-surface is moving [11, 13]. This yield real-time performances, as shown in [13].

In the case of implicit surface models built from the constructive approach, the convenience of spatial partitioning algorithms is more doubtful: basically, these methods will pre-convert[2] the construction-tree representing the object into a field image (by computing field values at grid nodes) before performing triangulation. As a consequence, they do not yield real-time performances anymore. Moreover, the use of this methodology for rendering an animation raises a number of drawbacks: the conversion to a field image and subsequent polygonisation must be recomputed from scratch for each animation frame, even if the animator knows that the implicit objects are just performing rigid displacements; moreover, scanning such motions through a fixed grid often creates aliasing artifacts on the surface shapes.

A totally different global visualization method consists in visualizing mutually repulsive particles, called "floaters", that sample the implicit surfaces [34]. The differential equation 1 is used again, but in the other way: this time, the particles velocities are computed from the time derivative of the field, so that the particles are constrained to stay onto the surface. Repulsive forces applied between particles, together with a fission/death process, yield a regular sampling even when the implicit surface experiences large deformations. This approach can be applied whatever the representation of the implicit surface is, although polygonisation based on spatial partitioning still seems more appropriate for the field image representation. Contrary to the spatial partitioning approach, particles rendering allows to take benefits of temporal coherence during the animation. However, the visualization it provides (each particle being represented as a small polygon oriented along the local tangent plane, see Figure 3, left) is not always sufficient for adequately visualizing an animation where a lot of objects move and deform. A solution would be to compute a Delaunay triangulation (which should ideally be incrementally modified at each animation step) of the particle set. However, it should be noted that generalizing Delaunay criteria for triangulating a curved surface embedded into a 3D space is not as straightforwards as computing the Delaunay tetrahedrization of a volume (Section 5.2 comes back to this point).

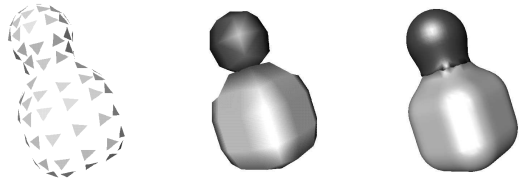


Figure 3: Rendering with particles (left) compared with local, primitive-based, polygonisation (results for two different mesh resolutions are shown).

In the case of implicit objects modeled using constructive soft geometry, recomputing a triangulation from a set of sample points can be avoided using a local, primitive-based, approach (see Figure 3, center and right). The idea is to associate a given local polygonisation to each individual implicit primitive. When several primitives are blended together, the polygonisation nodes migrate along a fixed axis (w.r.t. their associated primitive) to the iso-surface, or to the border of the territory to which they belong. This results into a piecewise polygonisation of the implicit surface, that can be generated in real-time during animations, thanks to temporal coherence [12]. Snapshots from our animation system are depicted in Figure 4. Little "gaps" between primitives can be

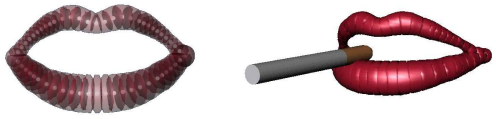


Figure 4: Snapshots from our animation system: left: internal structure of talking lips; right: an animation frame.

avoided through local overlapping as suggested in [28]. If a full polygonisation of the implicit surface has to be computed, the local polyhedrizations can be clipped near the territory borders, and then reconnected together [9].

4.2 Textures

Modeling and animating textured objects is not straithforwards using implicit surfaces. Indeed, a volumetric texture can be defined in the global frame of a field image. However, the surface will just move into the texture space when the object deforms, resulting into strange visual effects. In the constructive soft geometry approach, different 3D textures can be attached to each implicit primitive, and blended together at surface points where several primitives have a non-zero contribution [36]. However, animating the surface will then result into visual artifacts such as time-varying interferences between texture patterns. Anyway, being able to map and coherently animate a 2D texture on an implicit surface that deforms over time would be much more likely than 3D textures to model a kind of "skin" covering the animated object. However, this is known as one of the hardest problem in implicit surfaces modeling, since

[2] This does not mean that a full field image needs to be stored; polygonisation is usually performed "on the fly", using a continuation method for following the voxels that intersect the iso-surface.

implicit surfaces provide *no parameterization* on which to attach texture coordinates.

Contrary to what is often stated [38], I do not think that defining an initial texture mapping is much of a problem. Since a polygonisation or at least sample points are needed for visualizing the implicit surface, an initial mapping can be attached to those points. This can be done using any standard approach that works whatever the surface topological type. For instance, direct painting, interactive decoration [26], or texturing with triangular patterns [20] can be used for defining the initial texture map.

A much harder problem in Computer Animation is the way the texture will be animated when the object surface deforms over time (these deformations may even include separations and fusions!). A first practical approach, suggested in [1] and dedicated to the constructive methodology, consists in attaching patches of 2D textures to the individual primitives. During the animation, texture patterns may either be blended together in zones influenced by several primitives, as was done for 3D textures, or clipped at the boundary of the implicit territories in order to produce sharp color transitions. An easy way to implement this solution is to use the local primitive-based polygonisation [12], where texture coordinates may be directly associated to the sample points defined in each local primitive frame. However, this approach, that works well enough for uniforms color, will produce quite poor results when arbitrary texture patterns are used: parts of the texture would experience rigid motion during a smooth deformation of the implicit surface, while other parts, corresponding to blending areas, would simply appear or disappear during motion. In practice, this solution was successfully used for painting eyes and mouth on a deformable character [1], these texture patterns being adequately surrounded by the same uniform color.

A much wiser solution to animate 2D textures on an implicit surface which deforms over time is to use a vector-field based approach [28]. Sample points of fixed texture coordinates are animated using various vector fields, equation 1 being used again for constraining them to stay onto the implicit surface. Vector fields implementing various behaviors such as “sticking”, “twisting”, “elastic”, “sucking” and “shivering” textures are provided. Most of these fields are related to the individual motion of underlying primitives, since the implementation was performed in a constructive implicit surface representation. However, using the same ideas in the field image approach should not be difficult. An easy way to do it would be to attach texture coordinates to mutually repulsive floaters, as those of [34].

5 Animating textured objects: two case studies

The existence of theoretical solutions do not always give practical answers to “real-life” problems. The following case studies, chosen so that the well known advantages of implicit surfaces animation (see Section 3) immediately apply, will illustrate this point. Giving practical solutions to these problems yields hybrid solutions which combine several representations together.

5.1 Character animation

Experimenting with animated characters is quite natural when using implicit surfaces for animation:

- characters are a good application field for constructive soft geometry, since this representation is very easily embedded into layered animated models [6].

- efficient collision processing is an important feature in character animation, since interpenetrations between the body of the character and its arms, legs, clothes, and hair have to be prevented. Modeling contact through local deformations of the flesh [21, 6] is useful in this framework.

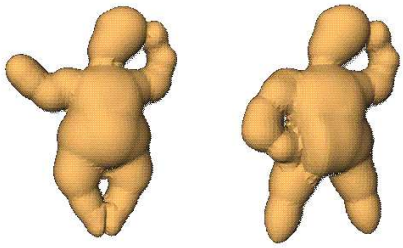


Figure 5: Character modeled with a fully implicit solution. Deformations due to contact are generated for parts that do not blend.

A first solution consists in entirely relying on constructive implicit surfaces for modeling the muscle, flesh and skin layers of the character. The volumetric primitives generating the surface are attached to various parts of the articulated skeleton of the character, sometime through intermediate springs in order to give a more dynamic behavior to the flesh [23, 24]. Muscles can be animated by tuning some of the primitive parameters over time. Moreover, the use of the resulting implicit surface as the skin layer generates smooth junctions between the different body parts. However, such a direct use of implicit surfaces raises two problems, which may greatly alter the quality of the resulting animation:

- Unwanted blending between different parts of the character must be avoided. A solution is to specify which of the primitives should blend together using a blending graph [22, 7], and to process collision between the parts of the character that do not blend [21, 6] (see Figure 5). However a closer look at the blending graph approach shows that it does not fully solves the problem: it is just a new formulation for an old solution [1] which consists in defining the field function as the maximum of field contributions from groups of blended primitives (these groups are the fully-connected subgraphs in the blending-graph terminology). The problem is that this solution does not insure order one continuity of the implicit surface: since the result is the mere union of distinct implicit volumes, creases may appear between different parts of the surface.

- Modeling and animating skin texture may be difficult with the implicit surface approach. Skin should ideally be able to slide over the muscle and flesh layers, and even to create wrinkles. Implicit surfaces seem not be the best model for solving these problems, although the vector-field based texturing approach reviewed in Section 4.2 could give a partial answer to the problem.

More practical solutions to character animation consist in embedding the implicit volumetric model into a parametric skin [30, 27]. This solution is perfectly wise since the character has little chances to break into pieces during the animation: a skin mesh of fixed topology, whose points are re-projected[3] onto the surface at each time step, is sufficient. The parametric skin will automatically smooth the tangent discontinuities due to the use of different blending groups in the implicit layer. This model for the skin will ease texture mapping. Lastly, sliding and wrinkling behaviors

[3]This is usually done using the iso-surface in-out function, although equation 1 could also be used for constraining the motion of skin points.

can be integrated to it. This solution shows that implicit surfaces can be much more useful in an hybrid representation framework than alone. Here, the implicit layer is still a good choice for easily modeling and animating the body volume while efficiently avoiding penetrations inside of it.

5.2 Animating a visually-realistic lava-flow

The motion and deformations of virtual lava flowing down slope can be simulated using an hydrodynamic particle system [29]. Contrary to the last case, using parametric surfaces for coating the flow would be almost impossible, since large deformations, and possibly separations into several disconnected components are to be generated. Implicit surfaces, that were already used for rendering particles [10], thus seem the right solution. However, the answer is not so straightforwards considering that several thousands of particles are to be animated, and that a texture should be coherently attached to the flow in order to render lava-crust.

The solution developed in [29] tackles the number of primitives problem thanks to an hybrid representation taking the best features of the constructive and of the image field approaches. Both point-primitives (the particles) and a grid structure are used. The former allows a direct control of the implicit surface from the particle flow. The latter, *i.e.* a grid of voxels, is used for treating field value queries in almost constant time. Field functions with a limited radius of influence are associated with particles. Then, the grid is used to keep track of the list of particles going in and out of each voxel at each time step. Since attraction-repulsion forces prevent particles from clustering, the number of particles in a voxel remains small (a few tenths). Field computation at a given point is performed by only summing the contributions of particles that lie in the current voxel and in some of its neighbors (depending on the value which has been given to the field radius of influence). Lastly, storing a full 3D grid, which would take too much memory, can be avoided considering the fact that the vertical extend of a lava flow always remains small (see Figure 6).

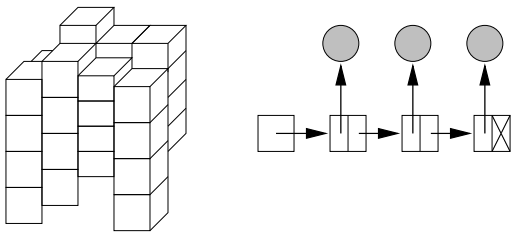


Figure 6: Grid data-structure used in the lava-flow animation.

Concerning the polygonisation and texturing problems, the aim was to generate a lava-crust texture that would *follows the flow*. Thus, the main texture patterns (i.e. the lava "clinkers") must be attached to the particles, while consistent crust texture has to be generated between them. The solution adopted in [29] can be seen as a very simplified version of the local visualization method [12]: due to the number and the small size of the particles, a single sample point is attached to each of them. A triangulation connecting these points is generated at each time step. Using a Delaunay triangulation would not ease the generation of lava clinkers patterns. We rather tile the implicit surface into pseudo-Voronoï regions surrounding each sample point, these regions being themselves triangulated (see Figure 7). To achieve this, we use a variant of the usual planar Delaunay criteria: A "pseudo-Delaunay triangle" is detected between three sample points when the projection[4] of these points onto the surface tangent plane satisfies Delaunay criteria. Then, the center of the circle defined by the three points is incorporated to the triangulation, since it belongs to the border of the three Voronoï regions. In areas of high curvature, some of the pseudo-Delaunay triangles may be of quite bad quality, since they were selected using a criteria in the tangent plane. If one of their angles exceeds 90^o (so that the center of the circle does not lie in the triangle anymore), we set this center to the middle of the closest triangle edge. This results into almost regular Voronoï regions, in which a lava clinker pattern is generated. This is done by generating displacement texture maps from stochastic noise functions that maintain continuity constraints along triangle edges (see [29, 20]). Color ranges and roughness parameters are computed from the current temperature of the associated particle. This process results into an animated texture whose patterns closely follow the underlying particles motion, and whose surface aspect changes while lava cools down. See Figure 8.

[4] We use a modified projection method, which preserves the distances.

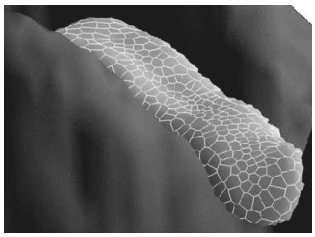


Figure 7: Pseudo-Voronoï tiling of the surface is generated by incorporating the centers of Delaunay circles into the triangulation.

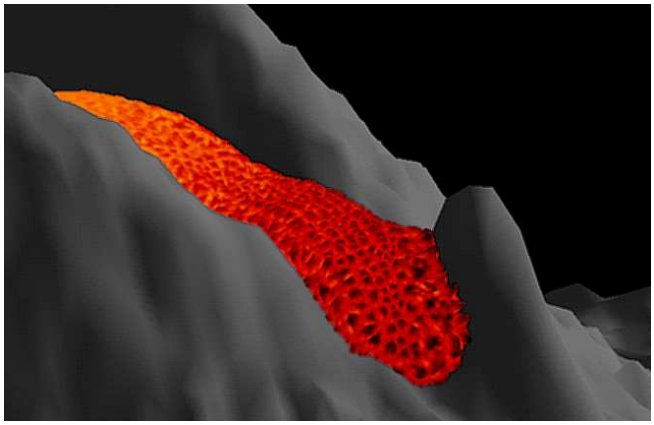


Figure 8: Animation of a textured lava-flow.

6 Conclusion

The first concluding remark is that implicit surfaces are not necessarily the best choice in Computer Animation applications. Before choosing this model, we should ask ourselves if we really need it, i.e. which specific advantages in terms of control, quality and efficiency it will bring to the current application. Then, a specific implicit representation has to be chosen. Constructive approaches seem to be the best model for providing the user with an intuitive control over motion and deformations. They can be combined with parametric surfaces to ease rendering. The image field approach is a more efficient representation for complex objects, eases polygonisation, and has proved useful in automatic morphing applications. Its use in Computer Animation has probably not been fully explored yet.

Experimenting with hybrid representations that would combine the advantages of both constructive and image field approaches seems very promising. The lava animation already uses such an hybrid model, although the grid structure does not store a full image field. The opposite way of using hybridation would be to animate and blend local image fields attached to a set of moving frames. We are currently studying an approach of that kind in the context of character animation.

Lastly, the ability of a model to provide an adaptive level of detail is an important feature in Computer Animation. This point has almost not being studied for animated implicit surfaces. The constructive representation seems to offer no simple solution. The wavelet representation of field images provides multi-resolution edition of the whole field. However, its use for the animation of a specific iso-surface still has to be studied. I guess that these points should inspire further researches within the next few years.

Acknowledgments: I would like to thanks the students and colleagues who worked with me on implicit surfaces, especially Eric Ferley, Dan Stora, Mathieu Desbrun, Nicolas Tsingos, Agata Opalach and Jean-Dominique Gascuel.

References

[1] Thaddeus Beier. Practical uses for implicit surfaces in animation. In *Modeling, Visualizing and Animating with Implicit Surfaces (SIGGRAPH'93 course notes Number 25)*, Anaheim, CA, August 1993.

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

[3] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[4] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990. Proceedings of Symposium on Interactive 3D Graphics.

[5] Andrew Guy Brian Wyvill and Eric Galin. Extending the CSG tree – warping, blending and boolean operations in an implicit surface modeling system. In *Implicit Surfaces'98—Eurographics and ACM-Siggraph Workshop*, pages 113–122, Seattle, USA, June 1998.

[6] Marie-Paule Cani-Gascuel. Layered deformable models with implicit surfaces. In *Graphics Interface (GI'98) Proceedings*, Vancouver, Canada, June 1998. Invited paper.

[7] 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.

[8] Benoit Crespin, Carole Blanc, and Christophe Schlick. Implicit sweep objects. *Computer Graphics Forum*, 15(3):165–174, August 1996. Proceedings of Eurographics'96, Poitiers, France.

[9] Benoit Crespin, Pascal Guitton, and Christophe Schlick. Efficient and accurate tesselation of implicit sweeps. In *CSG'98 (Set-theoretic Solid Modeling Techniques and Applications*, pages 49–63, 1998.

[10] M. Desbrun and M.P. 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.

[11] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Active implicit surface for computer animation. In *Graphics Interface (GI'98) Proceedings*, Vancouver, Canada, June 1998.

[12] Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Gascuel. Adaptive sampling of implicit surfaces for interactive modelling and animation. *Computer Graphics Forum*, 15(5):319–327, December 1996. A preliminary version of this paper appeared in *Implicit Surfaces'95*, Grenoble, France, may 1995.

[13] Marie-Paule Cani Eric Ferley and Jean-Dominique Gascuel. Practical volumetric sculpting. In *Implicit Surfaces'99—Eurographics and ACM-Siggraph Workshop*, Bordeaux, France, September 1999.

[14] Eric Galin and Samir Akkouche. Blob metamorphosis based on Minkowski sums. *Computer Graphics Forum*, 15(3):143–153, August 1996. Proceedings of Eurographics'96, Poitiers, France.

[15] Eric Galin and Samir Akkouche. Shape constrained blob metamorphosis. In *Implicit Surfaces'96—Eurographics and ACM-Siggraph Workshop*, pages 9–24, Eindhoven, the Netherlands, October 1996.

[16] J.D. Gascuel, M.P. Cani, M. Desbrun, E. Leroy, and C. Mirgon. Simulating landslides for natural disaster prevention. In *9th Eurographics Workshop on Computer Animation and Simulation (EGCAS'98)*, September 1998.

[17] Laurent Grisoni and Christophe Schlick. Multiresolution representation of implicit objects. In *Implicit Surfaces'98—Eurographics and ACM-Siggraph Workshop*, pages 1–10, Seattle, USA, June 1998.

[18] 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).

[19] 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.

[20] Fabrice Neyret and Marie-Paule Cani. Pattern-based texturing revisited. In *SIGGRAPH'99 Conference Proceedings*, Annual Conference Series. ACM SIGGRAPH, Addison Wesley, August 1999. Los Angeles, CA.

[21] Agata Opalach and Marie-Paule Cani-Gascuel. Local deformations for animation of implicit surfaces. In *SCCG'97*, Bratislava, Slovakia, 1997.

[22] Agata Opalach and Steve Maddock. Implicit surfaces: Appearance, blending and consistency. In *Fourth Eurographics Workshop on Animation and Simulat ion*, pages 233–245, Barcelona, Spain, September 1993.

[23] Agata Opalach and Steve Maddock. Disney effects with implicit surfaces. In *5th Eurographics Workshop on Animation and Simulation*, Oslo, Norway, September 1994.

[24] Agata Opalach and Steve Maddock. High level control of implicit surfaces for character animation. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 223–232, Grenoble, France, April 1995.

[25] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: Concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[26] Hans K. Pedersen. Decorating implicit surfaces. *Computer Graphics*, pages 291–300, August 1995. Proceedings of SIGGRAPH'95 (Los Angeles, California, August 1995).

[27] Jianhua Shen and Daniel Thalmann. Interactive shape design using metaballs and splines. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 187–195, Grenoble, France, April 1995.

[28] Jean-Paul Smets-Solanes. Vector field based texture mapping of animated implicit objects. *Computer Graphics Forum*, 15(3):289–300, August 1996. Proceedings of Eurographics'96, Poitiers, France.

[29] Dan Stora, Pierrer-Olivier Agliati, Marie-Paule Cani, Fabrice Neyret, and Jean-Dominique Gascuel. Animating lava flows. In *Graphics Interface'99*, May 1999.

[30] Russel Turner and Daniel Thalmann. The elastic surface layer model for animated character construction. In *Computer Graphics International'93*, June 1993.

[31] 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.

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

[33] Ross Whitaker and David Breen. Level-set models for the deformation of solid objects. In *Implicit Surfaces'98—Eurographics and ACM-Siggraph Workshop*, pages 19–36, Seattle, USA, June 1998.

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

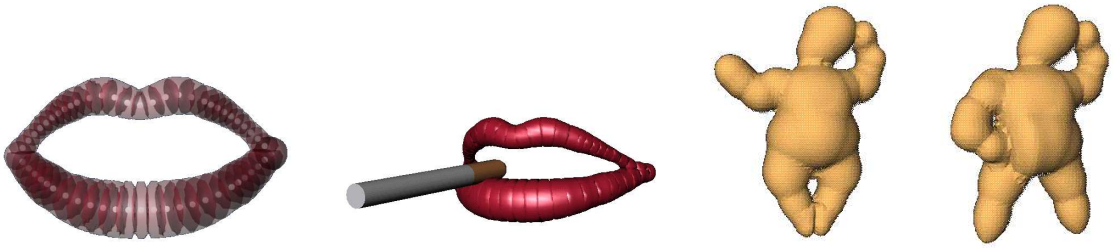
[35] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Animating soft objects. *The Visual Computer*, 2(4):235–242, August 1986.

[36] G. Wyvill, B. Wyvill, and C. McPheeters. Solid texturing of soft objects. *IEEE Computer Graphics and Applications*, pages 20–26, December 1987.

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

[38] Ruben Zonenschein, Jonas Gomes, Luiz Velho, and Luiz Henrique de Figueiredo. Controlling texture mapping onto implicit surfaces with particle systems. In *Implicit Surfaces'98—Eurographics and ACM-Siggraph Workshop*, pages 131–138, Seattle, USA, June 1998.

Figure 1



Figures 4 and 5

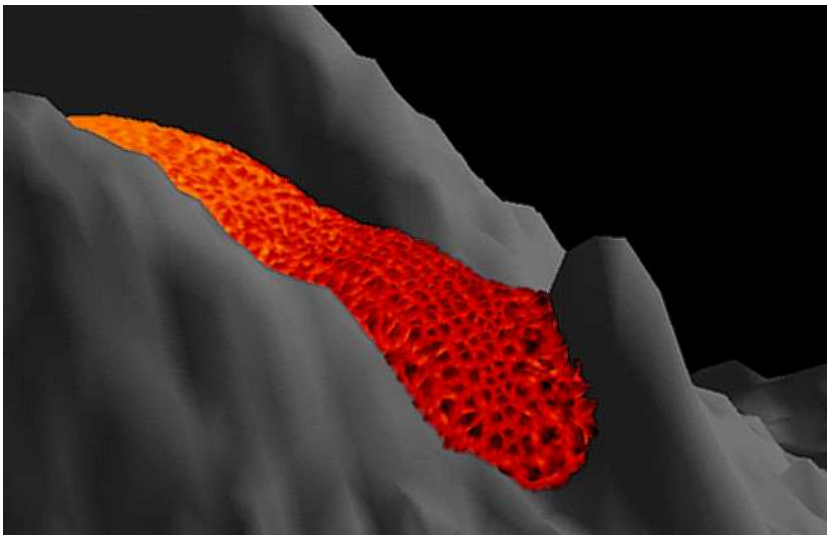


Figure 8