Animation of Deformable Models using Implicit Surfaces

Marie-Paule Cani-Gascuel Mathieu Desbrun
iMAGIS[†]-GRAVIR / IMAG
BP 53, F-38041 Grenoble cedex 09, France
Marie-Paule.Gascuel@imag.fr Mathieu.Desbrun@imag.fr

Abstract—
This paper presents a general approach for designing and animating complex deformable models with implicit surfaces. Implicit surfaces are introduced as an extra layer coating any kind of structure that moves and deforms over time. Offering a compact definition of a smooth surface around an object, they provide an efficient collision detection mechanism. The implicit layer deforms in order to generate exact contact surfaces between colliding bodies. A simple physically-based model approximating elastic behavior is then used for computing collision response. The implicit formulation also eases the control of the object's volume with a new method based on local controllers.

We present two different applications that illustrate the benefits of these techniques. First, the animation of simple characters made of articulated skeletons coated with implicit flesh exploits the compactness and enhanced control of the model. The second builds on the specific properties of implicit surfaces for modeling soft inelastic substances capable of separation and fusion that maintain a constant volume when animated.

***Keywords*— animation, implicit surfaces, deformable models, collision detection, collision response, inelasticity.**

I. Introduction

In traditional animation systems based on key-framing, specifying the motion and the successive shapes of objects interacting with a simulated world, requires a great amount of specialized knowledge and intuition from the animator. Models based on simplified physical laws have been proposed for automating these tasks. They generate motion and deformation from initial conditions and from a set of externally applied forces over time, and automatically detect and respond to collisions. These models are particularly appropriate for facilitating the animation of deformable objects. They can either be used alone for the simulation of inanimate bodies, or be combined with user-controlled structures as has been done for instance in character animation [25], [7].

This paper shows that a number of problems that are difficult to solve with previous models can be easily handled by combining them with an external deformable layer based on implicit surfaces. Indeed, the implicit formulation defines a smooth surface around the object that can be used to perform efficient collision detection, to enable exact contact modeling, and to ease volume preservation. This approach leads to a variety of applications such as the animation of elastic bodies, the modeling of soft substances that can separate or melt, or the animation of simple characters made of articulated skeletons coated with an implicitly specified volume.

A. Related work

Deformable models give a method for computing the alteration of an object's shape due to a set of externally applied forces. Deformations may be *elastic* or *inelastic* depending on whether the original shape is restored when external forces are removed. Most deformable models in Computer Graphics result from "nodal approaches", in the sense that they approximate deformations by the displacements of elementary nodes inside a flexible body. Some of them derive from the elasticity theory. Differential equations of motion are discretized in space, and then integrated over time by resolving a matrix equation at each time step. This scheme has been used for modeling both elastic [37], [39], [19] and inelastic [36] deformations. However, since the topology of the network of nodes does not vary over time, this approach is restricted to the animation of structured objects. A solution for modeling soft inelastic bodies that absorb deformations and may separate into pieces or melt during an animation is a physically-based particle system [26], [38], [40], [24]. In this case, a set of elementary masses called "particles" interact by means of forces that vary with the distance, such as Lennard-Jones forces that combine short range repulsion with long range attraction. Motion is computed by independently solving the equations of motion for each particle.

Nodal approaches are often compute intensive, since very small integration steps may be required. Another class of methods, called "global approaches", has been introduced to reduce computational costs [33], [45], [2]. The idea is to perform global shape transformations rather than simulating deformations that progressively propagate over a deformable body. However, this leads to a restricted range of deformations, and can only be applied to the animation of homogeneous visco-elastic material.

The way a deformable model detects and responds to collisions with other objects is very important, since it influences all subsequent motions and deformations. However, collision between soft objects is a complex phenomenon that has not been widely studied in physics[1]. Consequently,

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

[1]Elasticity theory [18] studies small oscillations around equilibrium states, but does not provide any model for collisions. Moreover, collisions between flexible bodies have finite time duration and consume some energy in deformations, so the solutions developed for rigid solids [27], [1] cannot be applied.

most of the solutions used in computer graphics have been especially designed for this application.

A first issue is the way interactions between deformable bodies can be detected. Quite surprisingly, the surface that is displayed to represent the object during an animation is seldom used for collision detection. For instance, deformable bodies have been represented by splines surfaces controlled by mass-nodes located at control points [25], [17], but no precise collision detection was performed. Soft inelastic substances animated with a particle system have been displayed with implicit surfaces [38], [40], but once again, the implicit surface itself was not used to collision detection. Besides being inaccurate, detecting collisions with mass-nodes is expensive [28]. A better solution is used by Pentland and Williams [32], who exploit the implicit inside/outside function that defines the surfaces of their models for precise and efficient collision detection. They test if the points that sample one object's surface are inside or outside another one in linear time.

Secondly, a method for computing response to collisions must be designed. Most of the models used so far[2] compute response forces from penalty methods [28]. These methods do not generate any contact surface between interacting flexible bodies, but use instead the amount of interpenetration for computing a force that pushes the objects apart. A quite promising approach [2] extends the analytical interaction processing used for rigid solids [1] to a global deformable model [45]. However, contact surfaces are approximated by discrete sets of contact points which, as the authors emphasize, is somewhat unsatisfactory.

To conclude the review of related work, most previous deformable models do not present a convincing way of processing collision and contacts between objects. Exact contact surfaces should be generated rather than local interpenetrations or bouncing before visual contact. Moreover, these problems may be exacerbated by the fact that soft collisions can last for a finite time. The combination of implicit surfaces and deformable models described in this paper supplies a systematic approach to treating collisions among deformable objects.

B. Overview

This paper presents an integrated set of methods that use implicit surfaces for animating a wide variety of deformable models. Implicit surfaces will be used as an extra layer that coats a base structure, deformable or not, with some smooth elastic flesh. While the base structure controls the large scale behavior, the implicit layer performs collision processing and generates local deformations due to contacts. It also handles constant volume deformations and topological changes such as separation or fusion. This allows simulation of behaviors that would be extremely difficult to treat with other methods.

Section II presents the layered approach we use. Section III details the collision processing method associated with the implicit layer. Section IV introduces volume preservation to the model. Section V presents two different applications of this formalism: the design of simplified characters made of articulated skeletons coated with elastic flesh, and the animation of soft inelastic substances capable of separation and fusion.

[2]As for instance Terzopoulos et al. models [37], [39], [36], and Pentland's "Thing World" system [32], [35].

II. Building layered models using implicit surfaces

A. Implicit surfaces generated by skeletons

Implicit isosurfaces such as "distance surfaces" [47], [5] allow the design of free form shapes through the manipulation of "skeletons" generating potential fields. Because they are simple to define and to control, they constitute a good alternative to traditional implicit surfaces defined by analytical equations.

An implicit isopotential surface generated by a set of skeletons s_i $(i = 1 \dots n)$ with associated "field functions" f_i is defined, at the isovalue c, by[3]:

$$\{P \in I\!R^3 \mid f(P) = c\} \text{ where } f(P) = \sum_{i=1}^{n} f_i(P) \qquad (1)$$

In this paper, f will be called the "field function", and the f_i will be designated as the "implicit contributions" of the different skeletons. The implicit surface surrounds a solid whose points satisfy $(f(P) \geq c)$, which may have several disconnected components. Normal vectors are directed along the field's gradient.

The skeletons s_i can be any geometric primitive admitting a well defined distance function: points, curves, parametric surfaces, simple volumes, etc. The field contributions f_i are decreasing functions of the distance to the associated skeleton:

$$f_i(P) = F_i(d(P, s_i)) \qquad (2)$$

where $d(., s_i)$ is the distance to s_i, and F_i can be defined for instance by pieces of polynomials [47] or by more sophisticated anisotropic functions [21], [3]. Most field functions associate a restricted scope of influence to each skeleton in order to provide local control of the surface and to optimize the computations. In practice, we use piecewise polynomial field contributions that are parametrized by three parameters t_i, k_i and R_i, called respectively thickness, stiffness and radius of influence, as sketched in Figure 1.

B. Embedding implicit surfaces into a layered construction

As emphasized in [7], a layered construction is a very efficient tool for creating complex models for animation. It often provides the user with parameters that are more intuitive and easier to use. Implicit surfaces generated by skeletons are especially well suited to this kind of approach.

In our framework, the user defines a deformable object by specifying:

[3]In the remainder of this paper, upper-case letters are used for points and vectors, and lower-case letters for scalar values.

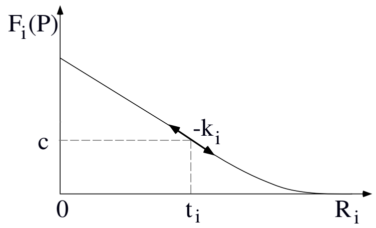

Fig. 1. A typical field contribution, with its three parameters t_i, k_i and R_i.

1. An internal physically-based model which will be used as a base structure during the animation. This model may be for instance a rigid solid defined by a mass and an inertia tensor, an articulated structure made of several such solids, a mass/spring network, a particle system, or any other model.
2. An implicit layer, that "coats" the base structure. This layer is built by defining skeletons that generate the implicit surface in local coordinate systems animated by the base structure. Skeletons may be points, line segments, triangles, or any graphic primitives.

During animations, the implicit layer immediately follows the motion and deformations generated by the base structure, defining a smooth surface that can be used for display. The topology of this surface may change over time, since separation or blending may be produced by the relative motion of skeletons.

C. *Benefits of the approach*

As stressed before, the coherence between the representation of an object and the model used for collision processing is very important for generating convincing motion. Using an implicit representation of the surface brings several benefits, such as a precise yet efficient collision detection mechanism, and a solution to precise contact modeling between deformable bodies.

Instead of using a purely geometric definition for the implicit layer, we use the deformable implicit model first introduced in [15]. This model, which will be reviewed in the next section, defines a correspondence between applied forces and deformations of the implicit surface that approximates elastic behavior. The latter can therefore be used for collision detection and response.

D. *Animation algorithm*

The general scheme for animating the resulting layered model develops as follows:

1. Animate the base structure by integrating the equations of motion according to the set of applied forces. This computes new positions for the skeletons that generate the implicit layer.
2. Process interactions between objects:
   (a) Use the implicit layer for detecting interpenetrations.
   (b) Model contact by locally deforming the implicit layer in order to generate exact contact surfaces between colliding bodies.
   (c) Integrate reaction and friction forces along contact surfaces. Add them to the set of external actions to be applied to the base structure at the next time step.

Detailing dynamic equations and integration schemes usable for the base structure is beyond the scope of this paper. An overview can be found for instance in [42]. The next section just presents the elastic model we use for the implicit layer, and details the associated collision detection and response algorithm.

III. Processing collisions with the implicit layer

In previous deformable models, when a collision is detected, response forces are approximated first. These forces are then used to compute subsequent deformation and motion of the objects, without producing an exact contact surface when collision endures in time. The implicit layer defined in this section uses a different approach. When a collision is detected it first performs precise contact modeling with the other object with local deformations. Compression along contact surfaces is then used to compute collision response.

A. *Collision detection*

Collision detection is performed between each pair of objects by first testing interpenetration between axis-aligned bounding boxes. When boxes intersect, the implicit representations of the surfaces and a set of sample points on them are used for a more precise detection: the surface points of each object that lie within the bounding box of the other object are evaluated by the implicit function of the other object. When an interpenetration is detected, the purely geometric contact modeling process described in the following section is applied. Issues for maintaining samples on the implicit surface will be discussed in Section III-D.

B. *Modeling contact between objects*

Once detected, an interpenetration must be avoided by locally deforming the implicit layer of each object. Both exact contact surfaces and deformations in propagation regions that model the *transverse propagation* of deformation must be generated (see Figure 2). We accomplish this by adding new local terms, called "deformation terms", to the field functions defining the implicit layers of each object.

- A negative field g modeling compression is added in the interpenetration region in order to generate a contact surface with the other object.
- A positive field p modeling the transverse propagation of deformations is added in the propagation region.

B.1 Deformation in the interpenetration region

The deformation field terms g_{ji} and g_{ij} to be added to the equations of the objects i and j should generate an exact contact surface. Thus the two equations

$$f_i(P) + g_{ji}(P) = c \tag{3}$$

$$f_j(P) + g_{ij}(P) = c \tag{4}$$

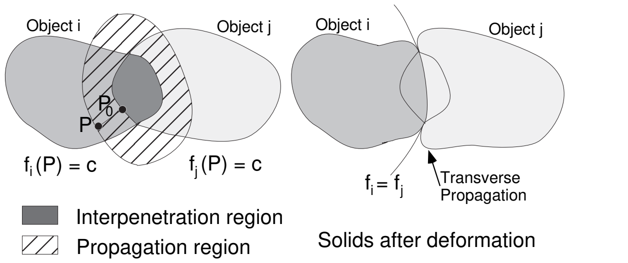

Fig. 2. Modeling contact consists in applying different deformation fields in the interpenetration region and in the propagation regions associated with each object (cross-sectional views).

must have a common solution. A simple and symmetric solution is to define the compression field terms by:

$$g_{ji}(P) = c - f_j(P) \tag{5}$$

$$g_{ij}(P) = c - f_i(P) \tag{6}$$

This choice is appropriate since deformation field terms are negative in the interpenetration region, and locally generate in this region a contact surface defined by the set of points such that:

$$f_i(P) = f_j(P) \tag{7}$$

We can remark at this point that an object can be defined as rigid if no deformation term is added to its surface equation. In that case, collision with a deformable implicit object can easily be modeled. The compression field term applied to the deformable object i is:

$$g_{ji}(P) = (c - f_j(P)) + (c - f_i(P)) \tag{8}$$

This makes the implicit layer of the deformable body exactly fit with the rigid object in the whole interpenetration region.

B.2 Deformation in the propagation areas

Our aim is to optimize the contact modeling process by directly computing deformed shapes in contact situations rather than simulating deformations that progressively propagate over the implicit layer. The bulge generated in the propagation region (see Figure 2) must be computed such that there is a smooth junction between the interpenetration region and the region where the object remains undeformed.

The user controls s_i's propagation field term $p_{ji}(P)$ (due to the collision with s_j) through two additional parameters in s_i's description:

- A value w_i giving the offset distance where deformations propagate around the interpenetration region (see Figure 2). No deformations will be generated outside this area.
- An "attenuation value" α_i giving the ratio between the maximal value desired for p_{ji} and the current maximal compression term in the interpenetration area.

We then define the propagation field term p_{ji} to be applied in the propagation zone of the object s_i as:

$$p_{ji}(P) = a_{k,a_0,w_i}(d(P, P_0)) \tag{9}$$

where P_0 the closest point to s_j in s_j's gradient direction (see Figure 2), $k = ||\nabla f_j(P_0)||$, a_0 is the maximal propagation value equal to α_i times the maximal compression field value, and $a_{k,a_0,w}(x)$ is a piecewise polynomial function as depicted in Figure 3. P_0 is computed in practice by iteratively performing small steps from P in the opposite direction of the object j's gradient. Our choice for the slope k ensures that the shape of the implicit object stays C^1 at the border of the interpenetration zone. One can use the following equation for $a_{k,a0,w}(x)$:

$$a_{k,a_0,w}(x) = \begin{cases} p_1x^3 + p_2x^2 + kx & \text{if } x \in [0, w/2] \\ \frac{4a_0 + (x-w)^2(4x-w)}{w^3} & \text{if } x \in [w/2, w] \end{cases} \tag{10}$$

where $p_1 = 4(wk - 4a_0)/w^3$ and $p_2 = 4(3a_0 - wk)/w^2$.

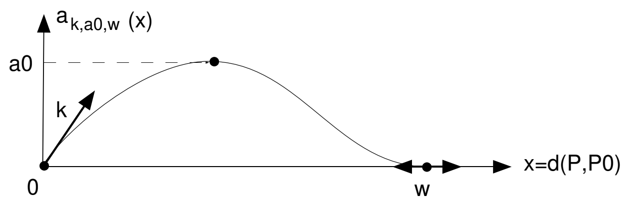

Fig. 3. Attenuation function defining the field in propagation areas. The slope at $x = 0$ is k, the maximum is a_0, and the function has zero values and derivatives for $x \geq w$.

Fig. 4. (left) Contact between two colliding deformable objects. (right) Cross-sectional view showing the exact contact modeling. A step-by-step description of this animation is described in Appendix.

C. *Response to collision*

Now that correctly deformed shapes are generated beween colliding objects as depicted in Figure 4, the deformation of the implicit layer must be used to deduce response forces. To achieve this, we describe in this section a model designed to approximate elastic behavior, first introduced in [15].

C.1 Modeling elasticity with implicit surfaces

A deformable model is defined by a given correspondence between applied forces and deformations. Both linear [39],

[19], [45] and non-linear [37] elasticity have been used in previous models. Linear elasticity states that stiffness near a given point P of a solid remains constant during deformations. The displacement of P from an initial position $X_0 = (x_0, y_0, z_0)$ to a final position $X(P) = (x(P), y(P), z(P))$ is then a linear function of the applied force $R(P)$:

$$R(P) = k_P(X(P) - X_0) \tag{11}$$

In non-linear models, the stiffness k is not only a function of the point P, but may also depend on its current location inside the solid. The force applied during a displacement of P from $X_0 = (x_0, y_0, z_0)$ to $X(P) = (x(P), y(P), z(P))$ is then:

$$R(P) = \int_{X_0}^{X(P)} k_P(Y)\, dY \tag{12}$$

For generality, the implicit layer we are defining should be able to exhibit both linear and non-linear behaviors.

Deformations of an implicit surface can be modeled by variations in its field function f. To express non-linear elasticity with this formalism, we let $dR(Y)$ be a infinitesimal radial force and dY the resulting infinitesimal radial displacement. From equation (12) they must satisfy: $k_P(Y)dY = dR(Y)$. We find:

$$df(Y) = \nabla f(Y) \cdot dY = \nabla f(Y) \cdot \frac{dR(Y)}{k_P(Y)} \tag{13}$$

We can then make the following observation: the set of points P satisfying $f(P) = c$, where f is the field function, is sufficient to define a surface. This set of points being fixed, the variation of f around the isosurface can then be used to model physical properties. Consequently, we choose to model stiffness with the field's gradient:

$$\forall Y \quad \nabla f(Y) = -k_P(Y)\, N(Y) \tag{14}$$

This choice simplifies equation (13), yielding:

$$\int_{X_0}^{X(P)} df(Y) = -\int_{X_0}^{X(P)} (N(P) \cdot dR(Y)) \tag{15}$$

Let $g(P) = f(X(P)) - f(X_0)$ be the deformation field term associated at equilibrium with the radial force R. If the normal vector $N(P)$ has remained constant during the deformation (which is then said to be "radial"), the formula giving the correspondence between applied forces and deformations, obtained by integrating equation (15), is:

$$g(P) = -N(P) \cdot R(P) \tag{16}$$

In practice, we will use a rewritten version of this latter equation:

$$R(P) = -g(P)\, N(P). \tag{17}$$

The correspondence between applied forces and deformations will only be used to integrate radial response forces during collisions. It can be noted that this formulation only approximates elastic behavior since only the radial component of forces is computed.

C.2 Stiffness control

As shown in [15], defining stiffness with the field's gradient has a simple geometric interpretation since field contributions are decreasing functions of the distance to the associated skeleton: the user defines local stiffness as the opposite of the slope of the field function. Both linear and non-linear elastic behaviors are easily designed, as depicted in Figure 5.

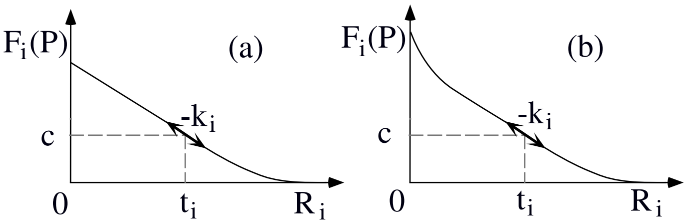

Fig. 5. Examples of field functions for the implicit layer. (a) Linear elasticity: stiffness, represented by the opposite of the slope, is constant during deformations. (b) Non-linear elasticity: stiffness increases during compressions.

In practice, the field's gradient also affects the object geometry when contributions from different skeletons blend together (equation (1)). It is difficult to specify the object's shape and its dynamic behavior at the same time by adjusting the different field contributions. We partially overcome this problem by introducing a scaling parameter K relating the effective stiffness and the field function's gradient:

$$k_P(Y) = K||\nabla f(Y)|| \tag{18}$$

This parameter K enables us to adjust the object stiffness according to its mass and inertia tensor without modifying its geometry.

C.3 Radial response forces

We have defined a way to model contact and a correspondence between deformations of the implicit layer and radial applied forces. Thus evaluating the resulting normal reaction force $R(P)$ at a point P on the contact surface during collision is straightforward. Equations (6) and (17) give us a value for the normal response force:

$$R_i(P) = -g_{ji}(P)N_i(P) = (f_j(P) - c)N_i(P) \tag{19}$$

where $N_i(P)$ is the normal vector to the deformed surface of object i at point P. Since g_{ji}, which models local compression, is negative, R_i has the same orientation as the normal vector, and models the internal force that tends to restore the initial shape of the object.

An important remark is that our model is consistent with the action/reaction principle: along contact surfaces, the deformation terms applied to both objects are equal, since $f_i = f_j$, while the normal vectors are opposite. Thus, opposite reaction forces $R_i(P) = -R_j(P)$ are generated.

C.4 Friction and damping forces

To model both tangential friction in contact areas and damping due to the progressive compression of the solids,

Fig. 6. Flexible clover falling on a rigid staircase.

we include a friction coefficient λ_i in the description of each object. When a collision occurs, the friction and damping force F_i at a point P of the contact surface between two objects i and j is expressed by:

$$F_i(P) = \lambda_i \lambda_j (V_j(P) - V_i(P)) \tag{20}$$

where $V_i(P)$ (respectively $V_j(P)$) is the speed of P, a point on the surface of the object i (respectively j).

The sum of radial and frictional forces is transmitted to the base structure of each object, and will be taken into account in subsequent motion. An example of animation using this technique is depicted on Figure 6.

D. Implementation

D.1 Sampling the isosurface

Sample points on the implicit surface are needed for both collision detection and numerical integration of response forces. Points that appear to be inside another object are moved to the deformed isosurface (for instance by performing search along the gradient direction), and the value of the deformation field at each new location directly gives the intensity of radial contact force along the small surface area sampled. These forces are added to friction forces and stored to be integrated by the base structure at the next time step.

Any method could be used for generating sample points on the implicit surface. The most widely used are spatial partitioning techniques [47], [23], [4], [43], [29]. However, approaches that take advantage of temporal coherence are more efficient for our application, since sample points do not move much between two consecutive steps of an animation. One method of this kind [44] maintains sample points called "floaters" on the isosurface, and ensures a good sampling distribution by connecting them with repulsive interaction laws. During deformations, more floaters may be automatically generated, or some of them may be removed, according to the local sampling density. The method we use, first introduced in [10], is slightly different. We present it briefly in the next paragraph since it has the advantage of also being convenient for volume preservation, as will be described below.

D.2 An adaptive sampling technique

The central idea is the following: each skeleton s_i contributing to the implicit layer emits a set of sample points in directions that are fixed in its local coordinate system, and are well distributed around it as illustrated in the stages 1 to 3 of Figure 7. Those of the points that reach the isosurface without going through an area already sampled by another skeleton are said to be "valid", and will be used as samples on the isosurface at a current time step (see stage 4 in Figure 7).

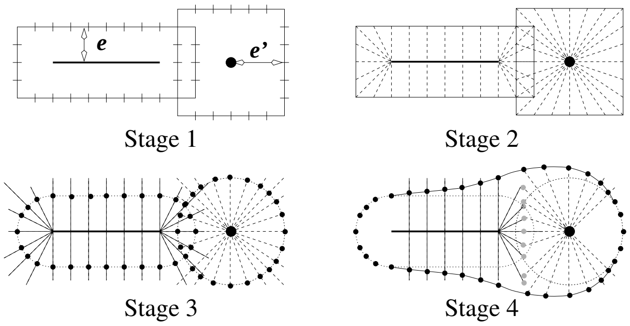

Fig. 7. Different stages of sampling initialization

This process can be defined more formally by associating a *territory* $\mathcal{T}_i$ to each skeleton s_i:

$$\mathcal{T}_i = \{P \in I\!R^3 \mid (f(P) \geq c) \text{ and } (\forall j \; f_i(P) \geq f_j(P))\} \tag{21}$$

$\mathcal{T}_i$ is the part of the implicit object where s_i's field contribution is the highest. This is equivalent to splitting the implicit volume into Voronoï regions defined by the skeletons, the "distance" from a point P to a skeleton s_i being defined by the field function. Sample points sent from s_i stop at $\mathcal{T}_i$'s boundary, and are valid if they lie on the isosurface.

Sample points then start from their previous position in s_i's local coordinate system at each time step to meet the surface again. Thus using temporal coherence increases efficiency. In addition, sample points that were located between two territories at a given animation step may be brought to the surface during later deformations. The sampling of the isosurface automatically adapts to large deformations and changes in topology.

D.3 Interactive visualization

A further benefit of the sampling method above is that the implicit surface can be displayed, during an interactive animation process, as a set of polygonal meshes built on the sample points (valid or not) that belong to the same skeleton. This provides a legible solid representation of the surface with no extra cost [10] as depicted in Figure 8. Note that we do not use the sample points for final high quality rendering of an animation. Storing the parameters of the isosurface such as the positions of the skeletons and the lists of colliding objects offers a much more compact representation that can then be used for computing direct ray-tracing of the implicit surfaces [13].

Fig. 8. Left: visualization of sample points as *scales* on the surface. Right: visualization of the same sampling but with a piecewise polygonization.

IV. Controlling volume during animations

The preservation of constant volume of deformable objects is desirable in animation [22]. Additionally, the user may desire precise control over volume variation in order to emphasize certain motion. The problem of volume preservation has been solved by methods based on Lagrange multipliers in the specific case of objects discretized into lattices of fixed topology [33], [34]. We present in this section the only solution to our knowledge that has been proposed for controlling the volume of bodies that undergo large deformations and topological changes such as separation and fusion [9].

Unwanted volume variations are exacerbated in implicit surface animation. They are produced by the field blending process during the relative motion of the skeletons, and they may be particularly annoying when the object undergoes separation or fusion. Although the problem has already been identified [47], [8], previous approaches only provide partial solutions that ensure volume preservation between an initial and a final state. But they do not ensure volume preservation during intermediate deformations, while drastically restricting the range of field functions that can be used.

This section presents a general method, applicable to any field function and isovalue, for controlling the volume of objects defined by implicit surfaces.

A. Local volume variations

The first problem is the detection of volume variation. The volume of an implicitly-defined object is given by: $V = \int\int\int_{f(P)\geq c} dx\, dy\, dz$. This expression cannot be computed analytically for most field functions. A simple method for volume approximation consists of discretizing space into voxels, and expressing the volume as the sum of voxels that lie inside the object. However, this technique would not provide a solution to the problem, since we also need to know near *which skeleton* the volume is changing.

Suppose the volume has been modified, as in Figure 9 (a), by the relative motion of some skeletons of the implicit layer. A solution for avoiding the variation consists of adjusting the strength of the field functions so that the

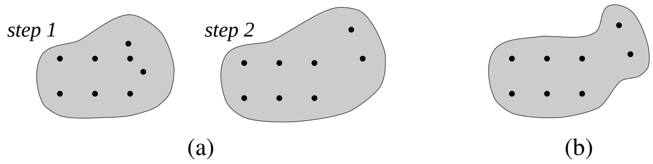

Fig. 9. (a) Volume variations of an implicit surface generated by point skeletons between two steps of animation. (b) Volume controlled locally in step 2.

volume keeps its initial value. However, these adjustments should not be done in areas where the object has not been deformed. As a consequence, volume variations should be detected and treated locally.

To define local volumes we use the notion of skeleton territory, already introduced in Section III-D.2: we define the *local volume* V_i associated with a skeleton s_i as the volume of its territory $\mathcal{T}_i$. The total volume of the implicit object can then be expressed as the sum of local volumes.

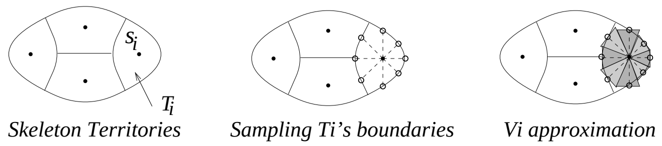

Fig. 10. Particle territories and sample points used for volume approximation.

Computing local volumes is straightforward when the adaptive sampling method of Section III-D is used. As shown in Figure 10, a local volume V_i is approximated as the sum of small pyramidal volumes defined around each sample point sent by s_i:

$$V_i = b_i \sum_{P\in\mathcal{P}_i} d(P, s_i)^3 \qquad (22)$$

where $\mathcal{P}_i$ is the set of sample points sent by a skeleton s_i, d is the distance, and b_i only depends on the angular distribution of samples for s_i. In practice, the factor b_i can be omitted in volume computations, since controlling the value of $\sum_{P\in\mathcal{P}_i} d(P, s_i)^3$ is sufficient for avoiding volume variations.

B. Volume control

We control local volume variations by associating a proportional-derivative controller to each skeleton. This controller can be seen as a black box that, given the current local volume $V_{i,t}$ and the value $V_{i,0}$ to reach or to maintain, outputs an adequate adjustment of the field function f_i of this particular skeleton.

For our application, the way to modify f_i must be chosen carefully since the norm of f_i's gradient gives the object local stiffness (see Section III-C.1). In order to adjust the volume of skeleton territories without modifying the object's physical properties, we combine the original field function with a translation $\epsilon_{i,t}$. At each time step, the field originally defined by the decreasing function of the distance

$f_i(P) = F_i(d(P, s_i))$ is replaced by:

$$f_{i,t} = F_i\left(d(P, s_i) - \epsilon_{i,t}\right). \qquad (23)$$

Since we need regular shape variations, we control the time derivative $\dot{\epsilon}_{i,t}$ of the translation parameter rather than its value. The inputs of the controller are then the normalized volume variation $\Delta_{i,t}$ and its time derivative $\dot{\Delta}_{i,t}$:

$$\Delta_{i,t} = \frac{V_{i,t} - V_{i,0}}{V_{i,0}} \qquad \dot{\Delta}_{i,t} = \frac{V_{i,t} - V_{i,t-dt}}{V_{i,0}\, dt} \qquad (24)$$

Its output is:

$$\dot{\epsilon}_{i,t} \;=\; \alpha\Delta_{i,t} + \beta\dot{\Delta}_{i,t} \qquad (25)$$

where α and β are appropriately chosen parameters. A simple example of volume control is given in Figure 11.

Fig. 11. Preserving volume during a blend between two point-skeletons. The leftmost picture shows the initial configuration and the speed vector. (a) shows the blending without any control, whereas (b) depicts a controlled blending ensuring constant volume.

This method used for maintaining a constant volume during an animation can be extended in order to impose specific volume variations that may be locally specified by the user: the target volumes $V_{i,0}$ simply have to be changed over time. These capabilities are useful in a broad range of applications in the field of animation with implicit surfaces. The next section presents two of them.

V. Applications

A very simple use of the implicit layer consists of combining it with a rigid internal structure, as was done in [15]. This leads to animations of elastic objects that locally deform during contacts, but return to their initial shape when no external force is applied. Snapshots from such an animation [12] are displayed in Figure 12.

This section presents two further applications of the model. The first is an animation of simple characters made of articulated skeletons coated with implicit flesh, which exploits the compactness and enhanced control offered by a layered structure. The second builds on the specific properties of implicit surfaces for modeling soft inelastic substances capable of separation and fusion, and that preserve their volume during the animation.

A. Modeling simplified characters

Specifying complex motion is greatly simplified when the animation system is able to abstract basic shapes from the representation of an object. Motion can then be refined with these shapes, and the animator only switches to the detailed representation when necessary. This is particularly true in character animation, where animators often spend a lot of time specifying motion and deformations of an articulated "skeleton" representing the character. Much less time is spent on animating the skin deformations, which may be generated automatically from the skeleton motion. Layered models are particularly well adapted to this context. Various approaches, either purely geometric [6], [11] or physically-based [7], [17], [31], [41] have already been proposed for the automatic animation of the skin from the motion of the underlying skeleton.

Fig. 12. Four frames of the animation *Simply Implicit*.

This section explains how to adapt the implicit layer model we have developed for this applications. One advantage of using an implicit representation of the flesh and skin is the compactness it offers: only skeletons and field functions need to be specified since the field models geometric and elastic properties. Another advantage is the ability to automatically detect collisions and model contact with other objects. This is particularly useful since characters often need to interact with the simulated world. Moreover, the volume control method we have developed can be used for creating more lively animations, by animating muscles for instance.

A.1 Structure used

In the terminology we have developed in Section II, a character will be represented by:

- **Base structure**: an articulated structure composed of a set of "links" connected by hinges. This structure may be animated with key frames, inverse kinematics, through the use of physically-based animation, or by any other technique.
- **Implicit layer**: each skeleton contributing to the implicit surface is defined in the local coordinate systems of one of the links.

The animation of this model is straightforward in the general framework we have defined: the general algorithm described in Section II-D is used. However, two problems due to the relative motion of skeletons inside the implicit layer have to be discussed:

1. Unwanted blending effects must be avoided during deformations.
2. Intercollisions should be detected between the different parts of a character.

The two next paragraphs explain how we deal with these problems.

A.2 Avoiding unwanted blending effects

The unwanted blending case is a difficult problem that has been known for a long time [46]. When we implicitly model characters for instance, we want their arms to blend with their shoulders, but not with another part of the body, as illustrated by Figure 13.

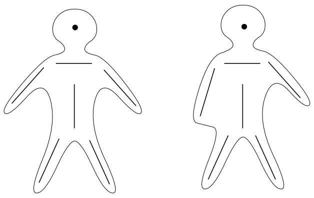

Fig. 13. The unwanted blending problem

A solution, first suggested in [46] and further developed in [30], consists in defining a neighboring graph between the different skeletons, and stating that a skeleton's field only blends with contributions from neighboring skeletons. More precisely, the field function f is replaced by the following procedure for computing the field value at a point P:

1. Compute all the field contributions at point P,
2. Select the predominant contribution from those of groups of skeletons that blend together,
3. Return this value without summing the other field contributions.

This algorithm avoids surface discontinuity during the controlled blending process, as explained in [20]. However, the method can not guarantee $\mathcal{C}^1$ continuity everywhere.

A.3 Processing intercollisions

Instead of processing collisions between pairs of objects, we use the algorithm described in Section III-B for processing collisions between pairs of skeleton territories that do not blend. The sampling method described in Section III-D provides us with a set of sample points, so we can easily compute local bounding boxes from the position of valid sample points and associate them with each skeleton territory. These boxes (enlarged by the maximal distance between sample points) cover the isosurface, allowing for precise collision detection even when the surface separates into several components.

When the underlying links are animated with a physically-based approach, contact forces computed between skeleton territories are transmitted to the reference link of the skeletons, to be integrated at the next time step.

A.4 Results

An example of animation performed with this model is depicted in Figure 14. A simulation method based on displacement constraints [14] is used for animating the base structure of the characters. We can observe several intercollisions as the character falls.

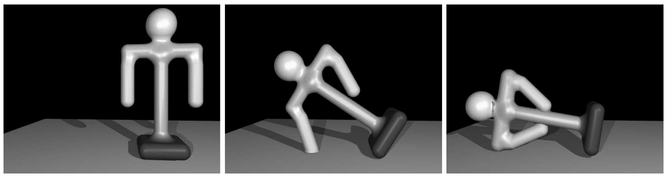

Fig. 14. Collisions with the ground and intercollisions of a simple articulated implicit object falling on its side.

B. Animating soft substances

This section presents a quite different application of the techniques we have described. The soft substance model we develop here, first introduced in [9], is particularly interesting since it benefits from the specific capability of implicit surfaces to model separation and fusion. The constant volume deformations generated by our model are very important in this case, otherwise an important increase of volume would be produced during fusion.

B.1 Structure used

As emphasized in the introduction, a simple and unified way of modeling a large variety of behaviors, including inelasticity and fractures, is to use physically-based particle systems. The main drawback of these systems, when used alone, is the lack of a method for defining a smooth surface for the objects. This is not a problem when several thousands of particles are used for accurately simulating fluids for instance. For animation purposes, the use of far fewer particles is sufficient for producing sufficient deformations. A surface, defined around the particles, should be used for visualization and for processing contact with other objects. The general framework we have defined provides a solution to this problem since the implicit layer seems well adapted for coating the particles. A piece of soft substance will then be composed of:

- **Base structure:** a particle system, made of a few tens of particles. Interactions between particles are modeled by attraction/repulsion forces such as Lennard-Jones forces, combined with friction forces that depend on the local density of particles. In our implementation, we use the following expressions for the interaction force and the friction force between two particles:

$$F_{int}(P_i \to P_j) = \gamma \left(\left(\frac{r_0}{r}\right)^8 - \left(\frac{r_0}{r}\right)^4 \right) \frac{P_j - P_i}{r^2} \quad (26)$$

$$F_{fr}(P_i \to P_j) = \mu(r)\; ||\dot{P}_i - \dot{P}_j||\; (\dot{P}_i - \dot{P}_j) \quad (27)$$

where $r = ||\, P_j - P_i \,||$, γ is the stiffness parameter, $\dot{P}_i$ stands for the speed vector of particle P_i, and μ is a decreasing continous function with finite support.

- **Implicit layer:** an implicit surface defined by point skeletons located on each particle. We use field contributions with relatively large thickness and radius of influence in order to give a smooth aspect to the simulated material even if only a few particles are used. Local volume controllers are associated with each skeleton, in order to prevent volume variations.

The general animation algorithm we have developed also applies to this case. The next paragraphs explain how we handle separation and fusion of the substance.

B.2 Modeling separation

When the particle system moves and deforms, a piece of substance may separate into several components, due to the relative motion of point skeletons defining the surface. However, if these disconnected chunks come back close to each other, they will blend as in Figure 11 rather than collide, since they are considered to be parts of the same surface.

This artifact is related to the unwanted blending problem we have referred to in the previous section. However, the problem is more complicated here since we cannot use a predefined blending graph: the blending properties between the set of point skeletons must change during the animation, according to the separation that is detected.

As a consequence, our method is based on the computation of a time varying blending graph. At each animation step, the current blending graph is stored as a list of neighbors, so called "blending list", associated with each point skeleton. Processing unwanted blending is done by reducing blending lists each time the implicit surface breaks into disconnected components that must not blend any more. The algorithm we use is the following:

- Before the animation, the blending graph is initialized as a complete graph, where each skeleton is connected to every other one. This corresponds to the standard field function, computed as the sum of all the field contributions.
- At each animation step:
  1. For each pair of point skeletons that blend together, we check if their spheres of influence, defined by the radius of influence of their fields, intersect. This relation defines an "influence graph". We then use the transitive closure of this graph for computing the blending graph we are looking for. For instance, in Figure 15, point A is detected to be in the same components than point B, while the separation with the other part is detected.
  2. Collisions are detected between skeleton territories not connected in the blending graph, as was done in Section V-A.3. As a result, pieces of substance that separate from the same body collide instead of blending when they come back close to each other.

Note that intersection tests on spheres of influence do not detect disconnections as soon as they appear, but when there is no more implicit contribution between the disconnected parts. However, we cannot reduce the blending list earlier without a sudden alteration of the shape of the disconnected components.

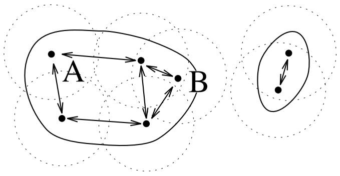

Fig. 15. The influence graph and its connected components. Particles A and B lie in the same component, and thus their fields will blend if they come close to each other.

B.3 Modeling fusion under compression

Blocks of soft substance such as clay or dough merge under compression forces that exceed a specified threshold. This behavior can easily be simulated with our model.

A fusion threshold is associated with each substance. Each time a collision is computed between two components of the same substance, the compression force along the contact surface between two skeleton territories is compared with the fusion threshold. If the threshold is exceeded, those skeleton that contribute to the contact surface for one component are added to the blending list for those skeletons contributing to the other component's contact surface. At the next time step, fields from the two pieces will then locally blend in this area, while collisions will still be computed between the rest of the components as illustrated in Figure 16. This merging will endure in time, unless the two pieces happen to be disconnected again by subsequent deformations. This method do not ensure, however, $\mathcal{C}^1$ continuity everywhere, as observed in [20]: tangent discontinuities may appear locally at some intermediate steps of the fusion.

Fig. 16. Progressive fusion under compression of two soft substances.

Instant fusion can be handled even more easily. Figure 17 shows four steps of an animation where a piece of soft substance is grabbed away by pliers and then released. The substance is made of nine particles only, and the fusion threshold has been set to zero, so that the substance *immediately* merges back after a collision. Here, volume preservation is essential. Otherwise, a very large and sudden increase of volume would be produced between the two last frames.

Figure 18 exhibits four frames from the animation *Kitchen Fiction* [16]. It shows the application of all the techniques detailed above to a more complex animation where a set of rigid tools are manipulating three different soft substances.

Fig. 17. Soft substance grabbed away by pliers and released.

Fig. 18. Four frames of the animation *Kitchen Fiction*.

VI. Conclusion

This paper has presented a general framework for building layered deformable models with implicit surfaces. The implicit formulation is particularly well adapted to a layered construction. It can be used for coating any reference component as, for instance, a rigid solid, an articulated structure, a mass-springs network, or a particle system. It defines a smooth surface around the object that can be used for rendering, and offers simple yet precise processing of collisions and contacts. The implicit inside-outside function faciliates collision detection, while the deformation of the implicit layer generates exact contact surfaces between colliding bodies. The physically-based model associated with the implicit layer approximates elasticity and allows the computation of response forces due to compression and friction. Moreover, preservation of deformed objects' volume is possible, even when the objects undergo significant changes such as separation or fusion.

We have illustrated this framework by detailing two very different applications: the animation of rigid articulated bodies coated with implicit flesh and the simulation of soft substances performing separation and fusion. The first of these applications should lead to interesting developments in the character animation area. Our layered framework would offer a compact way of modeling both geometry and the physical characteristics of simplified characters. Local adjustments of volume through time could be used for generating more expressive animations. Lastly, the capability of processing collisions and contact between a character and other objects of the scene would be an essential benefit of our approach.

Acknowledgements

We wish to thank Jean-Dominique Gascuel and Nicolas Tsingos for their contributions to the developement of the animation software. Many thanks to Agata Opalach and Jules Bloomenthal for fruitful discussions, to reviewers for very helpful comments, and to Andrew Hanson and George Drettakis for carefully rereading this paper.

References

[1] David Baraff. Dynamic simulation of non-penetrating rigid bodies. *PHD Thesis*, Cornell University, May 1992.

[2] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics*, 26(2):303–308, July 1992. Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992).

[3] Carole Blanc and Christophe Schlick. Extended field functions for soft objects. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 21–32, Grenoble, France, April 1995.

[4] Jules Bloomenthal. Polygonisation of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.

[5] 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.

[6] Nicholas Burtnyk and Mark Wein. Interactive skeleton technique for enhancing motion dynamics in key frame animation. *Communications of the ACM*, 19(10):564–569, October 1976.

[7] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3):243–252, July 1989.

[8] Mathieu Desbrun and Marie-Paule Gascuel. Highly deformable material for animation and collision processing. In *5th Eurographics Workshop on Animation and Simulation*, pages 89–102, Oslo, Norway, September 1994.

[9] 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.

[10] Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Gascuel. Adaptive sampling of implicit surfaces for interactive modeling and animation. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 171–185, Grenoble, France, April 1995.

[11] David R. Forsey. A surface model for skeleton-based character animation. In *Second Eurographics Workshop on Animation and Simulation, Vienna, Austria*, pages 55–73, September 1991.

[12] Morgane Furio, Marie-Paule Gascuel, Jean-Dominique Gascuel, and Alexis Lamouret. Simply implicit. *An animation produced by iMAGIS/LIENS, Paris, France*, July 1993.

[13] Jean-Dominique Gascuel. Implicit patches: An optimized and powerful ray intersection algorithm for implicit surfaces. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 143–159, Grenoble, France, April 1995.

[14] Jean-Dominique Gascuel and Marie-Paule Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, 10(4):191–204, March 1994.

[15] 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).

[16] Marie-Paule Gascuel, Mathieu Desbrun, Jean-Dominique Gascuel, and Stéphane Réhel. Kitchen fiction. *Eurographics'96*, Video Competition, August 1996.

[17] Marie-Paule Gascuel, Anne Verroust, and Claude Puech. A modeling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2(3):82–91, August 1991. A shorter version of this paper appeared in *Graphics Interface'91*.

[18] Henry Goldstein. *Classical Mechanics*. Addison-Wesley Publishing Company, 1950.

[19] Jean-Paul Gourret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics*, 23(3):21–29, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[20] Andrew Guy and Brian Wyvill. Controlled blending for implicit surfaces using a graph. In *Implicit Surfaces'95—the First Eurographics Workshop on Implicit Surfaces*, pages 107–112, Grenoble, France, April 1995.

[21] Zoran Kacic-Alesic and Brian Wyvill. Controlled blending of procedural implicit surfaces. In *Graphics Interface'91*, pages 236–245, Calgary, Canada, June 1991.

[22] John Lasseter. Principles of traditional animation applied to 3d computer animation. *Computer Graphics*, 21(4):35–43, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[23] 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).

[24] 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.

[25] Gavin Miller. The motion dynamics of snakes and worms. *Computer Graphics*, 22(4):169–177, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, Georgia).

[26] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *SIGGRAPH '89 Courses 30 notes*, pages 305–309, August 89.

[27] Brian Mirtich and John Canny. Impulse based simulation of rigid bodies. In *Proceedings of Symposium on interactive 3D Graphics*, pages 181–188. ACM SIGGRAPH, April 1995.

[28] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[29] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, November 1993.

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

[31] 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.

[32] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[33] John Platt and Alan Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4):279–288, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, Georgia).

[34] Ari Rappoport, Alla Sheffer, Daniel Youlus, and Michel Bercovier. Volume-preserving free-form deformations. In *ACM Solid Modeling'95*, pages 361–372, Salt Lake City, Utah, May 1995.

[35] Stan Sclaroff and Alex Pentland. Generalized implicit functions for computer graphics. *Computer Graphics*, 25(4):247–250, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[36] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformations: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4):269–278, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, Georgia).

[37] 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).

[38] 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.

[39] Demetri Terzopoulos and Andrew Witkin. Physically based model with rigid and deformable components. *IEEE Computer Graphics and Applications*, pages 41–51, December 1988.

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

[41] 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.

[42] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley Publishing Company, 1992.

[43] Jane Wilhelms and Allen Van Gelder. Octree for faster isosurface generation. *Transactions on Graphics*, 11(3):210–227, July 1992.

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

[45] Andrew Witkin and William Welch. Fast animation and control for non-rigid structures. *Computer Graphics*, 24(4):243–252, August 1990. Proceedings of SIGGRAPH'90 (Dallas, Texas, August 1990).

[46] Brian Wyvill and Geoff Wyvill. Field functions for implicit surfaces. *The Visual Computer*, 5:75–82, December 1989.

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

Appendix

A step-by-step description of a simple animation

We show next how to create animations and what our typical data structure is. It will focus on a simple animation of two colliding balls as depicted in Figure 4. The lower ball is considered as motionless while both are deformable. The data structure for each ball is defined as follows:

Ball #1:

<u>Base structure</u>: motionless mass-point

- constants
  - *mass* 10.0
  - *position* P_1 $(0, 0, 0)$

<u>Implicit Layer</u>

- skeletons
  - *point-skeleton* s_1 *located at* P_1
- field with linear elasticity
  - *thickness* $t_1 = 0.03$
  - *stiffness* $k_1 = 1.0$
  - *radius* $R_1 = 0.1$
- constants
  - stiffness scale $K = 1.0$
  - sampling rate $\sigma = 10^{-2}$
  - surface friction $\lambda = 0.1$

Ball #2:

<u>Base structure</u>: mass-point

- constants
  - *mass* 2.96

time-varying parameters

position P_2	$(0, 0, 1)$
speed	$(0, 0, 0)$

<u>Implicit Layer</u>

skeletons

point-skeleton s_2 *located at* P_2

field with linear elasticity

thickness	$t_2 = 0.01$
stiffness	$k_2 = 1.0$
radius	$R_2 = 0.04$

constants

stiffness scale	$K = 0.8$
sampling rate	$\sigma = 10^{-2}$
surface friction	$\lambda = 0.2$

General data:

dt	0.04
gravity	9.8

The other data needed, such as w_1 and w_2 giving the extent of the propagation regions, or $V_{1,0}$ and $V_{2,0}$ for volume preservation, are automically derived from given parameters.

Once this data structure is defined, the sampling method of section III-D is used to initialize sample points and to deduce axis-aligned bounding boxes around each of the two balls. The animation is then computed by numerically integrating the classical equations of motion over time for the mass-point to animate:

$$a_2^t = F_2^t / m_2$$

$$v_2^{t+dt} = v_2^t + a_2^t \; dt$$

$$P_2^{t+dt} = P_2^t + (v_2^t + v_2^{t+dt})/2 \; dt$$

where F_2^t is the accumulation of all collision forces computed on the sample points from the ball #2 during the time step and of gravity. Collision detection and computation of response forces are performed as described in this paper at each time step, on each sample point.

Marie-Paule Cani-Gascuel is an Assistant Professor of Computer Science at ENSIMAG. A graduate of the Ecole Normale Superieure (Paris, France) in both mathematics and computer science, she received the PhD degree in Computer Science from the University of Paris XI in 1990. She received the "Habilitation à diriger des recherches" degree from INPG (Grenoble, France) in March 1995. Her research interests include physically-based simulation, motion control of linked figures, and implicit surfaces applied to interactive modeling and animation. Marie-Paule Gascuel has served on the program committee of several Eurographics Workshops and Conferences since 1993.

Mathieu Desbrun was awarded an engineering degree in Computer Science at ENSIMAG with distinction and a graduate degree in Computer Graphics and Vision, both in 1994. He is currently a doctoral candidate in Computer Science at the INPG (Grenoble, France). His research interests include physically-based animation and implicit surfaces.