

Implicit Modelling with Skeleton Curves: Controlled Blending in Contact Situations

Alexis Angelidis, Pauline Jepp* and Marie-Paule Cani
iMAGIS[†]GRAVIR, INRIA Rhône-Alpes

655 avenue de l'Europe, 38330 Montbonnot, France

Alexis.Angelidis@imag.fr | pj@cpsc.ucalgary.ca | Marie-Paule.Cani@imag.fr

Abstract

Interactive implicit modelling with complex skeletons was recently improved with the introduction of primitives defined at different LODs. These implicit primitives use a subdivision curve as a skeleton. In this paper an extension to this representation is presented, in order to make it suitable for the interactive modelling and animation of soft objects in contact situations. The first contribution improves the display method for subdivision-based implicit primitives; this uses an adaptive polygonisation which locally refines, where necessary, according to the currently selected LOD. The second contribution consists of a new method for preventing unwanted blending when a skeleton-curve folds back onto itself. The third introduces local deformations where surfaces that should not blend come into contact. We illustrate the benefits of this methodology by describing two applications: the interactive modelling of complex organic shapes in contact situations and the physically-based animation of such organic shapes.

1 Introduction

Although parametric surfaces have been more widely used, the blending properties of implicit surfaces make them particularly suited to organic modelling. In such cases *soft* and branching objects are more easily achieved than with their parametric counterparts. In particular, skeleton-based implicit surfaces generate a smooth coating along a skeleton, which can be animated. In this paper we focus on implicit surfaces generated by a graph of branching curves, following the

methodology first introduced in [6]. The field function that generates the implicit surface is computed using convolution, which ensures bulge-free blending between implicit primitives at branching points. Moreover, the use of subdivision-curves as skeleton elements enables the generation of the implicit surface at different levels of detail (LODs), which makes this representation very convenient in an interactive modelling and animation framework.

However, making implicit surfaces behave as organic shapes when their skeleton moves and deforms is not easy: body-parts generated by distant pieces of curve on the skeleton graph should not blend, but rather squash each other when they come into contact. This requires careful control of blending, which was not provided by previous solutions.

This paper extends implicit subdivision curve primitives, giving them the ability to capture the shape and local deformations of organic objects in contact situations. The first contribution is to improve the display of the surfaces using an adaptive interactive polygonisation, which provides more details in highly curved areas. The second is the introduction of a local convolution method, which prevents unwanted blends whilst preserving smoothness, even when a skeleton curve coils or folds-back on itself. Lastly, we extend the Precise Contact Modelling (PCM) methodology [5, 11] to the subdivision-curve skeletons' framework. Two applications illustrate the benefits of this methodology: the interactive modelling of organic shapes in contact situations, and their physically-based animation.

This paper is structured as follows. In Section 2, some of the relevant background work is presented. Section 3 presents the adaptive interactive polygonisation. Section 4 introduces the new unwanted blending solution. The extension of PCM to the subdivision-curve primitives is described in Section 5. Applications are presented in Section 6.

*Currently at Departement of Computer Science, University of Calgary, Canada.

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

2 Related Works

2.1 Skeleton-Based Implicit Surfaces

An implicit surface is the set of points for which field function $f(p)$ takes value iso , so $f(p) - iso = 0$; $f(p)$ is a scalar field function and iso is the value at which the iso-surface is drawn. An evaluation of the field function at any point permits the *inside/outside* test which simplifies computations for controlled blending, collision detection and ray tracing.

The field function can be generated by points [1, 10, 15] or by more complex geometric primitives (such as line-segments or triangles) that form a *skeleton*. The skeleton, surrounded by its implicit *skin*, provides an intuitive way of controlling aspects of an object in a modelling or animation environment [8, 4]. There are two main ways to define an implicit surface from a skeleton:

Distance surfaces [4] calculate the field value at a given point P from the distance between P and the closest point on the skeleton. Blending the contributions of several skeleton elements is then usually performed by summing their field contributions. Although this eliminates creases on the surface where the underlying skeletal elements join, it also creates bulges. For example, if a skeleton is created from a polyline, blending of the surface is desirable but not always to the extent that a bulge is created where the lines join.

If a bulge at the join point of skeletal elements is considered an artifact, *Convolution Surfaces* [3] can be used. In this representation, the field value at a point P is calculated by integrating all the contributions from the different points on the skeleton. Smooth complex surfaces can be created by summing the integrals of individual field contributions of relatively simple skeletal elements. The introduction of closed-form solutions for the convolution integral allows efficient computation of convolution surfaces for specific sets of skeleton primitives, such as line-segments and arcs of circles [14].

2.2 Subdivision-curves as Skeletons

One of the main weaknesses of implicit surfaces compared to parametric ones has long been the lack of methods for generating them at different Levels of Detail (LOD). Since implicit surfaces are iso-surfaces of field functions, defining such LODs is equivalent to generating the field functions at different resolutions.

Such a method was recently proposed in the specific case of implicit surfaces generated by graphs of interconnected skeleton curves [6]. Subdivision curves are used for representing the curve-segments, enabling

the approximation of the skeleton, and thus of the field function it generates, at different resolutions. At each resolution, the skeleton is represented by a polyline. Closed-form convolution is used for efficiently generating a bulge-free and crease-free implicit surface. This is done by using $\frac{1}{d^3}$ as convolution kernel, which gives an integrated field that varies as the inverse of the squared distance to the skeleton elements. Interactive visualization is performed by generating a closed mesh around each skeleton curve. These meshes locally overlap near junction points where several curves meet, and their vertices are migrated to the iso-surface. The meshes resolution is uniform, but can be tuned according to the current LOD of the underlying implicit surface. Examples of objects created with this method are depicted in Figure 1.



Figure 1. From [6]: an implicit object defined from a graph of subdivision curves with two different LODs.

In this paper we will rely on the same methodology for defining implicit surfaces from graphs of subdivision curves. To complement the LODs on the field function, one of our contributions is to automatically generate LODs on the meshes that sample the surface. This is done by locally refining these meshes according to the local error from the iso-surface, considering its current LOD. This extension makes the model very well suited for interactive modelling and animation.

2.2.1 Controlled Blending

When parts of an implicit surface come into contact, careful blending between the field contributions of skeletal elements is required: indeed, parts of the surface generated by elements that are distant in the skeleton graph should not blend; moreover, if the aim is to model organic shapes, or more generally any kind of deformable bodies, object parts should not locally intersect, but rather deform when they come into contact.

Previous works have recommended the use of a restricted blending graph to avoid unwanted blending situations [4, 12]. However, defining the field as the max-

imum contribution from blending groups (i.e. maximal set of skeletal elements that are authorized to blend) introduces creases (the surface is not C^1 everywhere) [12, 5].

In the case of skeletons defined as graphs of branching curves, the skeleton’s topology was used for automatically defining the blending graph [6]. Moreover, a solution was proposed for guaranteeing C^1 continuity everywhere. It consisted in computing the field contribution at a point P by propagating blending along the skeleton curve as long as the next skeletal elements had a non-zero contribution at the query point. However, this solution was not fully satisfying: large folds that did not blend could be generated (for instance, in Figure 1 the tail does not blend with the head), but the authors stressed that blending could not be avoided between smaller folds, as illustrated in Figure 8.

In this paper we propose a solution that fully solves this problem, by introducing a *local convolution* formulation, which will be described in Section 4.

2.2.2 Contact Modelling

In the context of a modelling and animation environment, situations where neither blending nor intersection are considered appropriate call for contact modelling. The ability of objects to respond to collisions by deforming locally around a contact surface is indeed essential for a visually realistic modelling of contact situations [9]. This requires that the contact between objects, or parts of an object, and the deformations that occur as a result be modelled.

The technique known as Precise Contact Modelling (PCM) [5, 11] examines the interaction of objects and generates the resulting local deformations. The idea is to locally modify the objects’ field functions, once a collision has been detected, by adding a “deformation term” to them. Since this term is computed from the other object’s field function, it can be seen as a different way of blending the two shapes. More precisely, two different deformation field terms are computed for each object (or each part of an object) in contact:

- In the region where the two surfaces originally intersect, the field function f_1 of the first object is replaced by $f_1 + (iso - f_2)$, while f_2 is replaced by $f_2 + (iso - f_1)$. This moves the two surfaces to the same location, where $f_1 = f_2$.
- The deformation is propagated around the contact region in order to preserve surface smoothness, and to compensate for the loss of volume due to the first deformation. Opalach [11] gives an efficient solution for doing so: f_1 is replaced by $f_1 + b_1(f_2)$ (similarly f_2 by $f_2 + b_2(f_1)$), the

functions b_1 and b_2 being chosen so that the resulting new local bulge around the contact region smoothly connects with the rest of the object.

PCM was only applied to distance surfaces for which the field function was almost linearly decreasing around the iso-value [5, 11]. Its application to convolution surfaces such as those of [6], where the integrated field function varies as the inverse of the squared distance to the skeleton, introduces a new set of problems, that will be discussed in Section 5.

3 Adaptive Polygonisation

The categories for visualising iso-surfaces are ray-tracing, particles, or triangulation of the surface [2]. Even with improvements, the first method is too slow to allow interaction with the skeleton, although providing high quality images. The second produces too rough a display of the surface. The third has the advantage of benefitting from hardware acceleration. The method we propose is an adaptive triangulation method which is an extension of sampling and display method introduced by Desbrun [5] and improved by Cani [6], in which sample points are sent along given rays attached to skeletal elements, and sample its element’s *territory*, i.e. the regions where the elements field contribution is the highest. In order to produce an adaptive sampling, we propose to attach a rough polygonisation to a few sampling points, and to refine recursively the polygonisation using an error criterion.

Sample points are defined in a coordinate set attached to the control skeleton. As shown in 2, the point’s ray needs an origin on the skeleton and a direction along which the point moves.

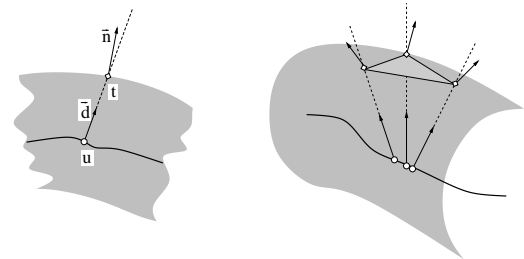


Figure 2. Sampling point. Left: a single point. Right: triangle supported by sampling points.

3.1 Setting local meshes

A rough polygonisation that coats the skeleton is needed for initialisation. It is not obvious how to define it in branching regions. Thus, the skeleton is cut into

its maximal linear pieces (see Figure 3). Closed meshes are generated around each of those pieces. To avoid visual discontinuities, the meshes are made to locally overlap at joints, as was done in [6] (see sub-section 2.2).

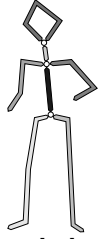


Figure 3. A coarse skeleton made of seven surface pieces.

The closed mesh is initialized as follows: on each linear piece of connected segments, sampling axes are sent around the surface in planes such that: if the joint is an extremity, the plane is normal to the segment; else the joint is of valence 2, and the plane is the medial plane between the two segments. Additional axes are sent at vertices of valence 1, aligned with the segment they are attached to. Note that each axis t_i is attached along a segment at a parameter u_i . The rough triangulation is then build using sample points ¹ on the axes as vertices (see Figure 4).

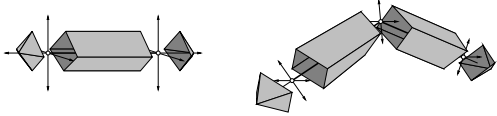


Figure 4. Exploded construction of a coarse local mesh.

3.2 Adaptive local refinement

The refinement of the coarse triangulation we have defined can be done in two ways: the first is a trivial uniform subdivision of all the coarse triangles; the second is an adaptive refinement using a local error metric.

The best method to choose whether a triangle needs subdivision or not would be to measure the volume that separates the triangle from the mathematically defined iso-surface. This would however be time consuming. We have chosen to measure the error using directly the field function: thus, a triangle edge is subdivided if the difference between the field value at its midpoint and the iso-value iso is greater than a constant (which could be view-dependent to refine areas of interest).

¹Convergence of the sample points towards the iso-surface is done by finding the root of $(f(\text{root} + t * \text{dir}) - iso)^2$ using Newton-Raphson method, thus knowing t , benefiting of coherence between positions, or can be done using methods proposed in [5].

$$\epsilon = \left| f\left(\frac{v_0 + v_1}{2}\right) - iso \right|$$

The subdivision of an edge then requires the insertion of a new axis: its origin and direction are interpolated linearly between the two axes of the vertices of that edge. Figure 5 illustrates the benefits of adaptive polygonisation: curved areas are more finely sampled than flatter areas.

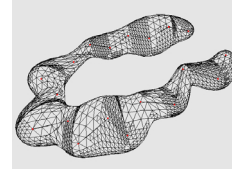


Figure 5. Adaptive triangulation of an implicit surface with varying radius along a skeleton curve.

4 Avoiding Unwanted Blending using Local Convolution

As mentioned previously, shape modeling and animation with skeleton defined implicit surfaces is intuitive, as the shape corresponds to a coating of the skeleton. Thanks to the blending property of this model, the resulting iso-surface is smooth and has an organic aspect without any effort. Unfortunately, this property also turns out to be a drawback, as when skeleton primitives are too close, all can blend with each other, without any control on the topology of the final shape. As stressed in Section 2, none of the previous solutions to this problem was satisfying.

Our solution can be seen as a local convolution, since we use a restricted skeleton range for computing the field value at a query point: a parameter u_i along the skeleton is assigned to each query point $p_i \in \mathbb{R}^3$. See Figure 6. We then define the portion of the skeleton to be used for computing $f(p_i)$ as a neighborhood of constant size D around u_i (D is expressed in Euclidian distance along the skeleton). The field value is computed exactly the same way as before, except that only the useful part of the skeleton is considered during this computation.

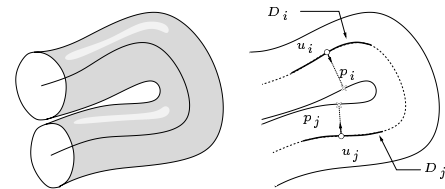


Figure 6. Local convolution

The only difficulty consists in assigning a skeleton parameter u_i to each point $p_i \in \mathbb{R}^3$. Defining u_i as the parameter of the closest point on the skeleton would not work, since it would produce creases in concavities.

We solve the problem by defining a parameter u_i on each skeleton-curve of the skeleton. The total field value near a branching point is then defined by summing these contributions.

Let us consider a given skeleton-curve. If the query point p_i is one of the sample points used for displaying the surface, finding u_i is easy: we just use the parameter of the reference point on the skeleton which was used for defining p_i (see Section 3).

In the general case, we need to define a similar reference point on the skeleton. This is done by re-using the set of planes defined at each vertex of the skeleton, and depicted in Figure 4 (for vertices of valence 1, the plane is normal to the segment, and for vertices of valence 2, it is the medial plane between the two adjacent segments). In-between planes are defined along segments using linear interpolation between the planes associated with vertices. Finding u_i for p_i then simply consists in finding which of these planes passes through p_i , and using the corresponding interpolation parameter. This is done by solving:

$$\begin{cases} r_i = (1 - u_i) * r_0 + u_i * r_1 \\ d_i = (1 - u_i) * d_0 + u_i * d_1 \\ d_i * (p_i - r_i) = 0 \\ u_i \in [0, 1] \end{cases}$$

where r_0 and r_1 are vertices that define the current curve-segment, and d_0 and d_1 are the normal vectors that define the planes at r_0 and r_1 , respectively. See Figure 7 for a two dimensional illustration.

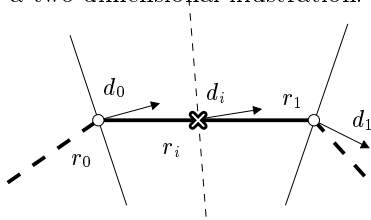


Figure 7. Assignment of a parameter to a point in \mathbb{R}^3 for a segment of a linear skeleton.

When several different planes generated by different line-segments of the curve pass through p_i , the one whose parameter generates the maximal field value is selected.

The benefits of our new controlled blending method, in comparison with the solution in [6], is shown on Figure 8. The local convolution solution enables us to prevent unwanted blends between small folds, which was impossible using the previous solution.

5 Precise Contact Modelling

Previous work on PCM dealt with distance implicit surfaces [5] and has been further developed to be more

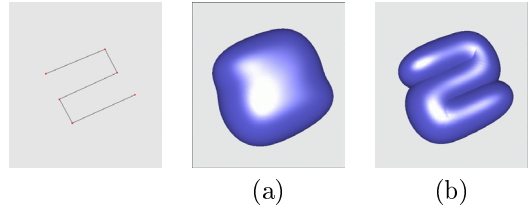


Figure 8. Controlled blending: comparison between the solution in [6] (a) and our new solution based on local convolution (b).

suitable to interactive modeling and animation environments [11]. In the context of convolution surfaces [6], the main difference lies with the shape of the convolution kernel; unlike the previous method, the field function is not linearly decreasing. Also, the subdivision structure of our skeleton can be used to decrease the computation cost of collision detection. This section will first describe how we deal with the collision detection and precise contact modeling (PCM), and then we will focus on the issues introduced by convolution surfaces.

The PCM process has three main steps: detect a collision; find the contact surface; and generate the deformations.

Detecting the collision is first performed using axes aligned bounding boxes, as in the previous method. The bounding boxes are constructed using sample points from the object which completely enclose the surface even after the PCM process (the deformed surface has a different shape and perhaps volume after contact). Each of the skeletal elements have their own bounding box, as an object might be in collision with itself. An element's bounding box is sized using its sample points (defined as described in Section 3). The size is finally increased to compensate for roughly sampled curved areas, using the current value of the sampling error.

A bounding box is associated with a skeletal element, so when a skeleton is refined the resulting bounding boxes are better fitted to the surface. These refined or sub-bounding boxes allow us to use the benefits associated with the levels of detail for the collision detection.

Sampling points use the bounding box in two stages: first a check is performed to detect if the bounding boxes of two non-adjacent skeletal elements (or skeletal elements of different objects) come into contact. If this test is true the next stage queries the sample points of the first object to detect if they are inside the bounding box of the second. When the sampling point is detected to be potentially in contact with another primitive (ie, inside both bounding boxes), the inside/outside test is

performed. If the result of this test is true, the object is flagged as having been in collision and the deformation term, presented below, is added to the sample point.

The deformation term has two roles: to deform the surfaces such that surfaces that penetrated each other are in exact contact; and to add bulges to the surrounding surface to keep them smooth around contact areas.

For two functions $f_i(p)$ and $f_j(p)$ whose iso-surfaces are in contact, the separation limit verifies $f_i(p) = f_j(p)$. Thus, for sampling points p_i of i that stand on object j 's side, adding the term $iso - f_j(p_i)$ to $f_i(p_i)$ makes p_i converge to the desired surface (which is done symmetrically for sampling points of j). On the border of this first deformation, the surface is however discontinuous; thus, a deformation term is added to sampling points within a propagation area around the contact surface.

The bulge area is modeled with a dilation term that is added around the objects. See for instance Figure 9. It is such that surfaces' C^1 continuity is preserved. Considering object i 's deformation by object j , the bulge function $b_i(f_j(p_i))$ which surrounds object j is chosen as to satisfy: $b_i(iso) = 0$, $b'_i(iso) = -1$, $b_i(c_j) = 0$ and $b'_i(c_j) = 0$, where $c_j < iso$ controls the end of the bulge created by object j . (it is a coating field around j that stops at c_j).

Opalach [11] did not give any equation for function b_i . We are experimenting with three versions of the function. The first one is very simple to implement and to use, since its only parameter is the bulge's extent in space, controlled by c_j : we use it in animation situations, where the use requires an efficient solution.

The first uses the fourth Hermite basis function. This does not use the h parameter, so offers less control of the shape of the bulge (see Figure 9):

$$\begin{aligned} \text{if } f_j(p) \in [c_j, iso] \\ b_i(x) = x^2(1-x) \\ \text{where } x = (f_j(p) - c_j)/(iso - c_j) \end{aligned}$$

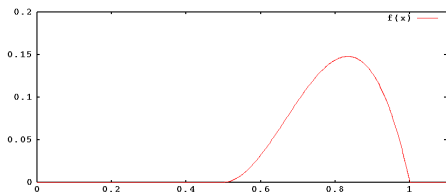


Figure 9. Fourth Hermite basis function.

The second one is a cubic piecewise polynomial which provides more control on the bulge's shape, since the local maximum of the bulge can be controlled in position and height: the position of the bulge is controlled with a field value m verifying $iso < m < c_j$, at which the function reaches $1/3 < h$, its global

maximum. The constraint on h is to ensure a unique maximum.

$$\begin{aligned} \text{if } f_j(p) \in [c_j, m] \\ b_i(f_j(p)) = (3-2x)x^2h \\ \text{where } x = (f_j(p) - c_j)/(m - c_j) \\ \text{if } f_j(p) \in [m, iso] \\ b_i(f_j(p)) = h + ((1-3h) + (2h-1)x)x^2 \\ \text{where } x = (f_j(p) - m)/(iso - m) \end{aligned}$$

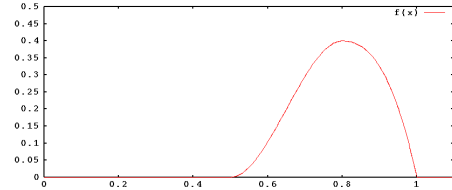


Figure 10. Height control with a piecewise polynomial.

The third piecewise function is non-polynomial. It is an adaptation of the second function, where the left part has been modified by a cosine. The control over the position of the bulge is lost to ensure that the periods lengths are constant: $m = c_j + (iso - c_j)(1 - \frac{1}{4n+3})$. The number of ripples is specified through n .

$$\begin{aligned} \text{if } f_j(p) \in [c_j, m] \\ b_i(f_j(p)) = (3-2x)x^2h \frac{\cos(((2n+1)x+1)\pi)+1}{2} \\ \text{where } x = (f_j(p) - c_j)/(m - c_j) \\ \text{if } f_j(p) \in [m, iso] \\ b_i(f_j(p)) = h + ((1-3h) + (2h-1)x)x^2 \\ \text{where } x = (f_j(p) - m)/(iso - m) \end{aligned}$$

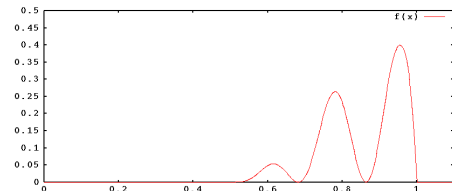


Figure 11. Ripples modeled using function with local maxima.

The height h is constrained to be at least lower than iso , otherwise, an unwanted implicit surface would appear, coating both surfaces. In the case of convolution surfaces, this does not provide enough constraints: unlike the previous method [11], the field function is not linearly decreasing; at some distance from the skeleton the field decreases too slowly and is almost constant. If such a value of iso is chosen then the result of the bulge function may propagate and coat the other object (see Figure 13). In practice, we need to choose an iso-value greater than 0.86 to achieve a sensible result.

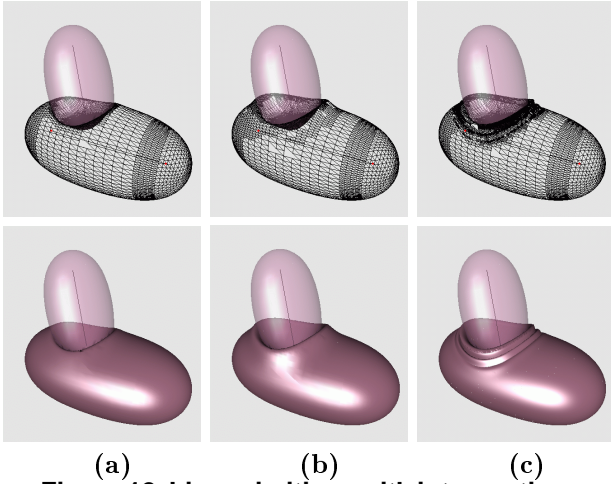


Figure 12. Line primitives with intersecting regions: (a) with fast bulge function, (b) with height controlling function, (c) with ripple function.

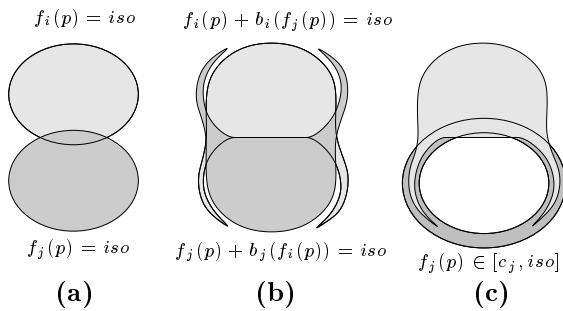


Figure 13. (a) Without PCM (b) With PCM (c) object i 's iso-surface and object j bulge function.

The bulges of Figure 12 were obtained using the following coefficients:

	iso	m	c_j	h	n
<i>column(a)</i>	2	—	0.4	—	—
<i>column(b)</i>	2	1.3	0.4	0.5	—
<i>column(c)</i>	2	—	0.4	0.5	2

6 Applications

6.1 Modelling organic shapes in contact situations

Figure 14 (a) and (b) show a part of a hand where the finger and thumb that squash a small ball are deformed using PCM. Figure 14 (c) and (d) compare the effect of not using and using PCM.

Our unwanted blending solution is illustrated in Figure 15, with the snail character. The wireframe pictures show various levels of detail on the iso-surface.

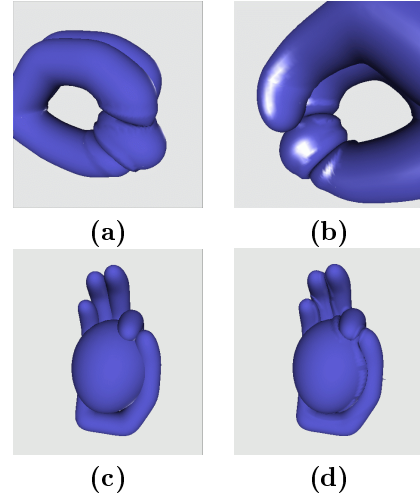


Figure 14. Hand: (a) and (b): finger and thumb grabbing a small ball (with PCM); (c) and (d): hand grabbing a ball, without and with PCM respectively.

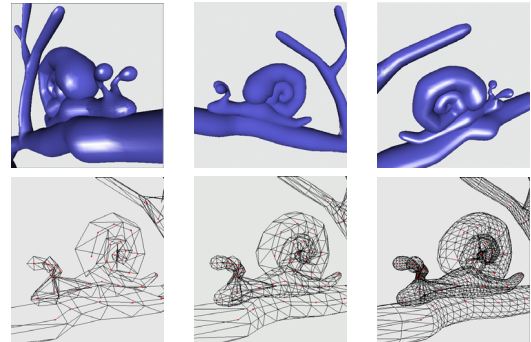


Figure 15. Top: snail character. Bottom: 3 error bounds for the adaptive polygonisation.

6.2 Animation of soft, organic shapes

Another application of the system is to animate deformable bodies made of skeleton curves coated with an implicit “skin”. Motion and the main deformations are controlled through the animation of the skeleton. The implicit surface that coats it may locally deform when different objects, or parts of an object, come into contact.

Figure 16 shows an example of a such an animation. Here, physically-based animation has been used for generating the motion of the object’s skeleton, which is modelled as a chain of masses and springs. The object falls under the action of gravity, sliding along a wall. A penalty method is used for computing collision response with the wall and between different folds of the object. Since physically-based motions are tuned using a number of trial and errors, being able to choose the

LOD of the implicit surface is very helpful. The animation is first tuned using a very low resolution. When the user is satisfied, high quality images are computed.

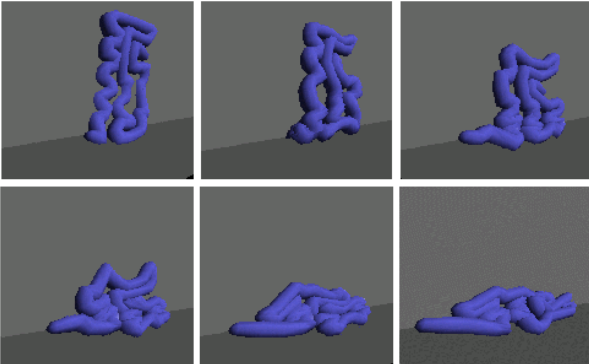


Figure 16. Controlled blending in a physically-based animation framework, where a deformable object falls under gravity.

7 Conclusion and Future Work

In this paper, we have extended the subdivision-curve implicit primitive representation to make it more suitable for the interactive modelling and animation of complex, soft objects in contact situations. Blending control is performed using a local convolution: only a portion of the skeleton is taken into account for computing the field contribution at a query point. Contact modelling is essential for modelling the deformation of non-blending shapes when they come into contact. It uses a specific combination of the different objects field functions to generate contact surfaces while preserving smoothness. Lastly, the surfaces are displayed using an adaptive polygonization scheme. This increases efficiency without dropping quality, since a higher resolution is generated in highly curved areas.

The system has been used to create a virtual hand that uses contact modelling to model interactions with soft objects. This hand could be used as a tool in a virtual sculpting environment [7, 13]. At present tools in such systems tend to be limited to the addition or removal of material. It would be interesting to use a virtual hand with precise contact modelling in order to emulate the actions of a real hand and its effect on the clay. We have also shown that our modelling method can be used in a physically-based animation framework, the implicit “skin” generated with our method being used for coating an animated skeleton. We are planning to use this methodology in the context of a pedagogic intestinal surgery simulator, aimed at surgeons’ training.

Acknowledgments: This research was supported by the EU Training and Mobility of Researchers network (TMR FMRX-CT96-0036) “Platform for Animation and Virtual Reality”.

The authors would like to thank François Faure for his contributions to the physically-based simulation.

References

- [1] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.
- [2] Jules Bloomenthal, Chandrajit Bajaj, Jim Blinn, Marie-Paule Cani, Alyn Rockwood, Brian Wyvill, and Geoff Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann, jul 1997.
- [3] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, jul 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] Marie-Paule Cani and Mathieu Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, mar 1997.
- [6] Marie-Paule Cani and Samuel Hornus. Subdivision curve primitives: a new solution for interactive implicit modeling. In *Shape Modelling International*, Italy, May 2001.
- [7] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Practical volumetric sculpting. *the Visual Computer*, 16(8):469–480, dec 2000.
- [8] 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.
- [9] J. Harrison and D. Forsey. A kinematic model for collision response. In *In Fifth Eurographics Workshop on Animation and Simulation*, Oslo, Norway, September 1994.
- [10] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Objects modeling by distribution function and a method of image generation (in japanese). *The Transactions of the Institute of Electronics and Communication Engineers of Japan*, J68-D(4):718–725, 1985.
- [11] Agata Opalach and Marie-Paule Cani. Local deformations for animation of implicit surfaces. In *SCCG’97*, Bratislava, Slovakia, 1997.
- [12] Agata Opalach and Steve Maddock. Implicit surfaces: Appearance, blending and consistency. In *Fourth Eurographics Workshop on Animation and Simulation*, pages 233–245, Barcelona, Spain, sep 1993.
- [13] Ronald N. Perry and Sarah F. Frisken. Kizamu: A system for sculpting digital characters. *Proceedings of SIGGRAPH 2001*, pages 47–56, August 2001. ISBN 1-58113-292-1.
- [14] Andrei Sherstyuk. Kernel functions in convolution surfaces: a comparative analysis. *The Visual Computer*, 15(4), 1999.
- [15] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, August 1986.