Technical Report TR-NCCA-2008-01

EMBEDDED IMPLICIT STAND-INS FOR ANIMATED MESHES: A CASE OF HYBRID MODELLING

Denis Kravtsov, Oleg Fryazinov, Valery Adzhiev, Alexander Pasko, Peter Comninos

The National Centre for Computer Animation
Bournemouth Media School
Bournemouth University
Talbot Campus,
Poole, Dorset BH12 5BB
United Kingdom

2008

Technical Report TR-NCCA-2008-01

ISBN: 1-85899-123-4

Title: Embedded Implicit Stand-ins for Animated Meshes: a Case of Hybrid Modelling

Authors: Denis Kravtsov, Oleg Fryazinov, Valery Adzhiev, Alexander Pasko, Peter Comninos

Key words and Phrases: Convolution surfaces, blending, implicit surfaces, fluid, character animation, hybrid modelling

Abstract:

In this paper we address shape modelling problems, encountered in computer animation and computer games development that are difficult to solve just using polygonal meshes. Our approach is based on a hybrid modelling concept that combines polygonal meshes with implicit surfaces. A hybrid model consists of an animated polygonal mesh and an approximation of this mesh by a convolution surface stand-in that is embedded within it or is attached to it. The motions of both objects are synchronised using a rigging skeleton. This approach is used to model the interaction between an animated mesh object and a viscoelastic substance, normally modelled in implicit form. The adhesive behaviour of the viscous object is modelled using geometric blending operations on the corresponding implicit surfaces. Another application of this approach is the creation of metamorphosing implicit surface parts that are attached to an animated mesh. A prototype implementation of the proposed approach and several examples of modelling and animation with near real-time preview times are presented.

Report date: 14 August, 2008

Web site to download from: URL: http://ncca.bournemouth.ac.uk/index.html

The authors' e-mail addresses: dkravtsov@bournemouth.ac.uk, ofryazinov@bournemouth.ac.uk, pasko@acm.org, vadzhiev@bournemouth.ac.uk, pcomninos@bournemouth.ac.uk

Supplementary Notes:

The National Centre for Computer Animation
Bournemouth Media School
Bournemouth University
Talbot Campus,
Poole, Dorset BH12 5BB
United Kingdom

1 Introduction

In modern computer graphics and animation systems polygonal and NURBS meshes are predominantly used for modelling. On the other hand, implicit surfaces (defined by continuous scalar fields and discrete level sets) have been shown to have a great potential in various related application areas, in particular in the modelling of human and animal figures [Shen and Thalmann 1995], in the simulation of natural phenomena [Wei et al. 2003], in simulation of object cracking and explosions [Barbier et al. 2005], and in geometric operations (such as blending and metamorphosis [Galin et al. 2000]).

Hybrid models combining different geometric representations, for example polygonal meshes and implicit surfaces [Allègre et al. 2006], are emerging in geometric modelling as a potentially promising area of investigation. It is anticipated that computer animation and computer games in particular would benefit greatly from the closer integration of mesh and implicit surface models. One of the many aspects of hybrid modelling concerns the generation of a continuous scalar field around an animated polygonal mesh. Another aspect concerns the combination of meshes and implicit surface components into a single hybrid model. Such hybrid models can then be used to create animation effects which are very hard or impossible to achieve using pure mesh models.

Animated meshes and certain types of implicit surfaces (such as soft objects, distance-based objects and convolution surfaces) share a common characteristic - a rigging skeleton. So, we consider a skeleton as a proper base for their integration. Previously published work has concentrated on the generation of animated scalar fields using point skeletons [Wyvill et al. 1986] or discretized fields [Osher and Fedkiw 2002]. However, point skeleton-based implicit surfaces have quite limited expressive power. Discrete scalar fields are better suited for physical simulations, but are hard to animate using skeletons. Distance based skeleton implicits [Cani-Gascuel 1998] extend the geometric domain of modelled and animated shapes, but still generate models with a "blobby" appearance to them. Convolution surfaces with skeletons of various types (line segments, arcs or triangle meshes) allow for modelling of a much wider class of objects than point-based implicits, have naturally smoothly blended surfaces and are easily animated using skeletons.

The main contributions of this paper are the following: a hybrid model combining skeleton driven animated meshes with skeleton-based implicit surfaces; a procedure for fitting a convolution surface to an animated mesh; applications of such hybrid models for the simulation of viscous object behaviour, which requires blending between mesh models animated using conventional skeleton techniques, and for the creation of easily metamorphosed parts for animated characters.

We propose embedding convolution implicit surfaces inside animated meshes and coupling them to each other such that the motions of both types of objects are synchronised. Thus, skeletons serve for motion synchronisation. The objects can either share a common skeleton or have individual synchronously moving skeletons.

An embedded convolution surface has to closely approximate the embedding mesh such that its motion requires no changes or minimal changes of the convolution surface parameters. This may require a procedure for fitting a convolution surface to an initial mesh taking into account its specified motion. Interaction of a viscous object with an animated object is modelled using geometric blending operations on the corresponding implicit surfaces, which achieves near real-time rendering speeds. Note that the initial animated mesh is rendered in the final animation together with the blending surface, which creates the visual effect of the blending of the mesh itself. Thus the embedded convolution surface serves as an implicit stand-in for the animated mesh. In practical terms, we aim to develop a simple technique allowing the user of conventional animation tools to model specific types of object interaction with having a non-time-consuming pre-processing step.

2 Related works

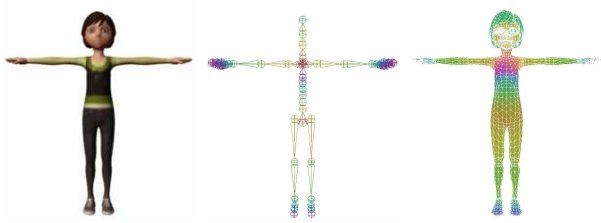
2.1 Implicit and hybrid models in animation

Many authors have used implicit surfaces for character animation. Elliptical blobs for skeletal animation were used in [Blinn 1982], where the transformation of the blob is inherited from the transformation of the joints of the skeleton. In [Opalach and Maddock 1995], blobby objects were used, which are quite easy to define. However, with this method it is difficult to control the resulting "blobby" mesh and to model a proper mesh one needs a large number of primitives.

One of the earliest attempts of using hybrid modelling involved embedding mesh objects into implicit surface primitives [Singh and Parent 1995] to implement polyhedral object deformations of articulated deformable bodies. Skeleton-based implicits for non-polygonal animated objects were examined, for instance, in [Cani-Gascuel 1998], where geometric primitives producing distance fields were used for character animation - although this technique may lead to C^1 discontinuities in the resulting surfaces. The coating of arbitrary animated models by implicit surfaces, employed in this technique, is not always acceptable to animators. We consider our approach complementary to the coating technique. Polygonal meshes and implicit primitives are combined together in a HybridTree [Allègre et al. 2006] using blending, Boolean and other operations supported by the conversion procedures between two different models. Embedding, attachments and skeleton-based motion synchronisation of meshes and implicits were not directly addressed.

Implicit surfaces were also used for the approximation of polygonal meshes using different approaches, such as Radial Basis Functions (RBFs) [Turk and O'Brien 1999] and Multi-level Partition of Unity implicits (MPUs) [Ohtake et al. 2003]. These methods generally work well with static meshes, but are less suitable for animation because dynamic models require per-frame re-fitting and can not be easily edited by the user due to the complexity of the implicit surface.

One of the interesting alternatives among implicit surfaces is that of convolution surfaces [Bloomenthal and Shoemake 1991]. Convolution surfaces proved to be useful for the modelling of organic shapes and objects with a complex topology [McCormack and Sherstyuk 1998]. Convolution surfaces can be smoothly blended with each other and provide a good approximation for polygonal meshes typical of animated characters. The authors of [Angelidis and Cani 2002] proposed using convolution surfaces with adaptive skeletal curves and surfaces for character modelling. As the issues of the animation of the newly generated skeleton were not discussed in their paper, it is hard to evaluate the applicability of this technique to the problems related to animated meshes. In this paper we utilise convolution surfaces with line segment skeletons for hybrid modelling applications concerned with both the creation of scalar fields around animated meshes and for the modelling of implicit surface parts attached to and synchronised with mesh-based characters.



(a) (b) (c)

Figure 1: *Animated mesh information: (a) Polygonal mesh, (b) Skeleton, (c) Skinning information. Model courtesy of John Doublestein.*

2.2 Interaction of viscous objects with polygonal meshes

The main technique for modelling viscous objects is fluid simulation. Thus, in [Foster and Fedkiw 2001] the use of a combination of level-set implicit surfaces and inertia-free particles was proposed for modelling the interaction of fluids with polygonal meshes. Also, [Thürey et al. 2006] describe the generation of control particles with respect to the underlying mesh model. Despite the good control of the shape of the viscous object that these methods exhibit, they are computationally expensive. The approach described in [Clavet et al. 2005] uses smoothed particle hydrodynamics and additional physical simulation to model viscoelastic objects and their interaction with rigid bodies. In [Jin et al. 2005] blobby objects are used to approximate a static mesh. These blobby objects are then used in an interpolation process to achieve morphing liquid effects.

The authors of [Shi and Yu 2005] used the signed distance to a skeleton to control the animation of fluids. This approach provides the animator with more control over the resulting animation sequence, but the computation times involved are still relatively high. The combination of different techniques appears to be common practice in animation production involving viscous materials. In our work we simulate viscous objects using geometric blending between the implicit objects generated from given polygonal meshes.

3 Problem statement and approach outline

We address problems that appear in computer animation practice and are difficult or impossible to solve using polygonal meshes exclusively. Our approach relies upon hybrid modelling combining polygonal meshes with implicit surfaces. From a general point of view, there are three main ways of achieving this: (a) by coating of meshes with implicit surfaces, (b) by attaching implicit surfaces to mesh surfaces, and (c) by embedding implicit surfaces into mesh objects. Coating was discussed in [Cani-Gascuel 1998] (see the previous section). In this paper, embedding is considered as a general approach and attaching as its special case. An important constraint that applies to our approach is the near real-time rendering of all hybrid models.

Let an animated object be defined by a polygonal mesh shown in (Fig. 1a), with a rigging skeleton shown in (Fig. 1b), skinning information shown in (Fig. 1c), and a set of animation transformations for the skeleton nodes. A rigging skeleton is a set of hierarchically connected joints used to specify the mesh motion in the animation sequence. If there is no skeleton provided, it can still be extracted from the polygonal mesh using one of the published techniques [Katz and Tal 2003] [Tierny et al. 2006] [Liu et al. 2003].

An important application area for embedded implicit surfaces is the modelling of viscoelastic object adhesive behaviour in its interaction with an animated mesh object. To obtain visually plausible results with near real-time preview, we propose to apply geometric blending between the implicit surfaces representing both interacting objects.

A viscoelastic object can be represented by either an implicit surface or by another polygonal mesh. We will mainly concentrate on the former case to simulate such viscous liquids as jam, honey or tar, and to show how such liquids interact with an animated object. Thus we will deal with the adhesion of the liquid to the surface, its stretching following the object's motion and other related topics.

Natural controllable blending is one of the best-known useful properties of implicit surfaces. We can use this property for modelling adhesive behaviour. This, in general, assumes the conversion of the animated mesh into an implicit surface. However, an exact conversion is a complex task. We propose to use a hybrid model which includes a polygonal mesh and an approximation of this mesh by some implicit surface embedded within it using a fitting procedure. It is impractical to perform this fitting to the mesh for each frame of animation. Thus, it is preferable that an implicit surface is made to follow the motion of the animated mesh. A convolution surface satisfies this requirement when its skeleton is built using the rigging skeleton of the animated mesh and the motions of both skeletons are synchronised. This derived convolution surface can be blended with the implicit surface, representing the viscous liquid, to mimic their adhesive interaction. The resulting surface can be polygonized to obtain a near real-time preview or can be ray-traced to produce the final animation sequence.

An alternative application is that of the attachment of implicit surfaces to meshes for the creation of such parts of animated characters that are difficult or impossible to model with meshes. An attachment can be considered a special case of embedding, when a rigging skeleton is still used for motion synchronisation, but surfaces are fitted only at the area of their contact and the implicit surface does not necessarily penetrate the mesh.

4 Background

In this section we describe the necessary basics of convolution surfaces and the operation of blending union both of which are used in our approach.

4.1 Convolution surfaces

Convolution surfaces were first introduced in [Bloomenthal and Shoemake 1991]. Given a scalar field function f, such as:
$f(\mathbf{p}) = \int_S g(\mathbf{r})h(\mathbf{p}-\mathbf{r})dp; \mathbf{p}, \mathbf{r} \in R^3$
where S is a skeleton specifying the resulting surface, $\mathbf{r} \in R^3$ is a set of points that belong to the skeleton S, $g(\mathbf{r})$ defines the geometry of primitives (skeleton function), $h(\mathbf{p})$ is a kernel function. Thus, the convolution surface on the primitive is a point set satisfying the equality:
$f(\mathbf{p}) - T = 0$ where T is a threshold scalar value for the convolution function.

In our method we use convolution surfaces based on line segments and a Cauchy kernel [McCormack and Sherstyuk 1998]. Thus,
$h(d) = \frac{1}{(1+s^2d^2)^2}; d > 0$
where d denotes the Euclidean distance to a point of interest and s is a scalar value controlling the radius of the convolution surface's cross-section.
Given a line segment

$\mathbf{r}(t) = \mathbf{b} + t\mathbf{a}; 0 \le t \le l$
where $\mathbf{b}$ is the segment base (vector), $\mathbf{a}$ is the segment axis (vector) and l is the segment length, for an arbitrary point $\mathbf{p} \in R^3$ the squared distance between $\mathbf{r}(t)$ and $\mathbf{p}$ would be:
$d^2(t) = |\mathbf{q}|^2 + t^2 - 2t\mathbf{q} \odot \mathbf{a}$
where $\mathbf{q} = \mathbf{p} - \mathbf{b}$.
A field function for an arbitrary point $\mathbf{p}$ would be:
$f(\mathbf{p}) = \int_0^l \frac{dt}{(1+s^2r^2(t))^2} = \frac{x}{2m^2\left(m^2+s^2x^2\right)} + \frac{l-x}{2m^2n^2} + \frac{1}{2sm^3}\left(\arctan\left[\frac{sx}{m}\right] + \arctan\left[\frac{s(l-x)}{m}\right]\right)$
where: x is the coordinate on the segment's axis, $x = (\mathbf{p} - \mathbf{b}) \odot \mathbf{a}$, $m^2 = 1 + s^2(q^2 - x^2)$, and $n^2 = 1 + s^2(q^2 + l^2 - 2lx)$.

The main advantage of a convolution surface is the smooth transition between its parts that are defined by different skeletal elements as illustrated in (Fig. 2b). When moving skeletal elements, the convolution surface follows the motion quite naturally, which is useful in animation.

4.2 The blending union of implicit surfaces

Given two implicit surfaces, $f_1(x, y, z) = 0$ and $f_2(x, y, z) = 0$, the blending operation between these surfaces is defined as:
$blend(f_1, f_2) = union(f_1, f_2) + disp(f_1, f_2)$,
where $union$ is the set-theoretic union function and $disp$ is a displacement function, that is defined as:
$disp(f_1, f_2) = \frac{a_0}{1+\left(\frac{f_1}{a_1}\right)^2+\left(\frac{f_2}{a_2}\right)^2}$
where a_0, a_1 and a_2 are blending parameters. a_0 controls the overall resulting shape, a_1 and a_2 specify the contributions to the blending of the shapes defined by functions f_1 and f_2.

Using R-functions (see [Pasko et al. 2005]) the formula for the blending operation can be obtained:
$blend(f_1, f_2) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1+\left(\frac{f_1}{a_1}\right)^2+\left(\frac{f_2}{a_2}\right)^2}$

R-function based blending is more suitable for user interaction when compared with the direct algebraic summation of the defining functions, because several parameters are provided to control the overall shape of the blend and its symmetry. If it is preferable that blending takes place inside a particular volumetric shape, we can use the bounded blending operation [Pasko et al. 2005].

5 The proposed approach

In this section we systematically describe our approach for the construction of hybrid models combining animated meshes with embedded or attached convolution surfaces. The proposed solution outlined in Section 3 can be subdivided into the following steps:

1. The creation of the initial approximation of the given mesh with bounding volumes using the skeleton information.
2. The tuning of the initial approximation.
3. The creation of an embedded convolution surface for the initial polygonal mesh.
4. The application steps: (1) the definition of the blending between the convolution surface and the viscous object for the modelling of the adhesive behaviour of a viscoelastic object and its interaction with an animated object; (2) the creation of metamorphosing implicit parts for an animated mesh.

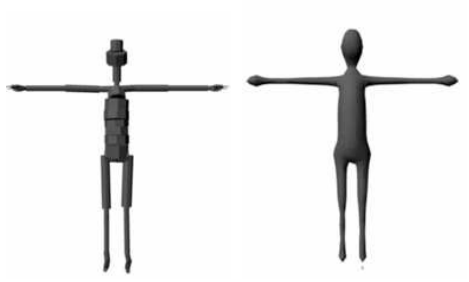
Each step requires the rendering of the current convolution surface and either the blending surface or the attached convolution surface. Note that both application steps can be performed together, when an animated mesh with attached implicit parts is interacting with a viscous material.

5.1 The initial mesh approximation with bounding volumes

To start with, we create an approximation of the given polygonal mesh using the embedded convolution surface. A similar problem in 2D space was addressed in [Tai et al. 2004] using a silhouette curve sketched on a plane as the desired shape of a convolution surface. We extend this problem formulation to the 3D case of the convolution surface embedding into the polygonal mesh. In our formulation
$\mathbf{V} = \{v_1, .., v_n\}$ are mesh vertices;
$\mathbf{C} = \{c_1, .., c_m\}$ are the centroids of mesh triangles;
$\mathbf{P} = \mathbf{V} \cup \mathbf{C} = \{p_1, .., p_{m+n}\} = \{v_1, .., v_n, c_1, .., c_m\}$ are estimation points (mesh vertices and centroids);
$\mathbf{L} = \{l_1, .., l_n\}$ are convolution line segments;
$\mathbf{F_i} = [F(p_1, L_i), .., F(p_{m+n}, L_i)]^\top$ is a transposed vector of field values produced by the i-th convolution line segment at the estimation points;
$\mathbf{F} = [\mathbf{F_1}, .., \mathbf{F_N}]$ are field values produced by all line segments at all estimation points;
$\mathbf{T} = [T, .., T]^\top$ is a transposed vector of threshold values for the convolution surface. We need to solve the constrained least-squares problem:
$\min_{\substack{\Lambda \ge 0 \\ \mathbf{D}_k - T > 0}} (\mathbf{F\Lambda} - \mathbf{T})^\top (\mathbf{F\Lambda} - \mathbf{T})$
for the unknown weights of field contribution from each line segment $\mathbf{\Lambda} = [\lambda_1, .., \lambda_N]^\top$. The constraint $\lambda \ge 0$ is given for the topological correctness of the resulting convolution surface. The constraint $\mathbf{D}_k - T > 0$, where $\mathbf{D} = \mathbf{F\Lambda}$ and $k = 1...(m+n)$ is intended to ensure that the convolution surface is completely embedded into the mesh (at least at the estimation points). We need to apply a numerical search in the N-dimensional space of the parameters $\mathbf{\Lambda}$ to minimise the above least squares criterion with the given constraints. The paper [Tai et al. 2004] mentions a method suitable for this sort of problems: the Hildreth-d'Esopo method, which is time consuming and provides good quality of the approximation. Another well-known method, the Levenberg-Marquart method, is less time consuming, but provides a poor quality approximation. Since the initial approximation is performed only once and does not influence the speed of further interactions, the former method is preferable.

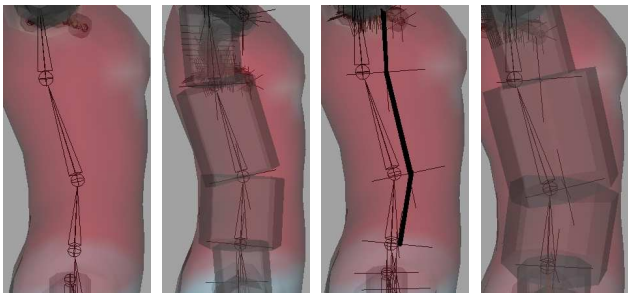
At the moment we approximate the mesh using a convolution with a constant radius along a line segment. In the case of large variations of the distance between the mesh and its medial axis, a better approximation could be achieved by using weighted convolution surfaces [Jin et al. 2001]. In this case, there is a need to solve a constrained truncated cones fitting problem. When the radii for both endpoints of the convolution segment are found, they can be used for the computation of the weighted convolution surface. Such approximation would decrease the median distance between the convolution surface and the polygonal mesh. It is worth observing that, if we wish to apply the blending union operation described in the following sections, the quality of the initial approximation does not play a significant part in this process.

As the first step of the global minimisation procedure, we can estimate the parameters of the convolution surface using the available information. For the initial approximation we use the rigging skeleton. Given the set of bones of the rigging skeleton, where each bone



(a) (b)

Figure 2: *Initial approximation: (a) Initial placement of bounding volumes inside the mesh, (b) Produced convolution surface*



(a) (b) (c) (d)

Figure 3: *Tuning of the initial approximation: (a) Original mesh with the rigging skeleton bones, (b) Initial placement of bounding volumes, (c) Migration of the convolution skeleton, (d) Bounding volumes around the produced convolution surfaces*

Figure 4: *Behaviour of convolution surfaces during animation*

is a line segment in 3D space, we use the set of these segments as the basis for an initial convolution skeleton. We denote the start vertex and the end vertex for each such skeleton segment as markers. To calculate the radius of the convolution surface for each segment, we calculate the minimal distance between each line segment specified by the markers and the polygonal mesh. At this stage we can build bounding volumes around each line segment for the real-time preview of the convolution surface. The bounding volume for each segment is a cylinder. For the set of rigging skeleton bones $s_i \in \mathbf{S}$ (where $\mathbf{S}$ is a set of skeleton bones) the radius of the i-th cylinder associated with the i-th bone is:
$r_i = \min_{s_i \in \mathbf{S}, p_j \in \mathbf{P}} (dist(s_i, p_j))$, where p_j is the j-th face of the polygonal mesh ($\mathbf{P}$ - connected set of faces) and $dist(s_i, p_j)$ is a distance between the bone s_i and the face p_j. Thus, each bounding volume is fitted inside the mesh in its initial position. Rendered bounding volumes help the user to better understand how the resulting approximating convolution surface is embedded into the mesh (Fig. 2).

The problem of unwanted blending between different parts of the convolution surface can be partially solved using the union operation based on R-functions. Separate lists of line segments are created for branching skeleton structures. The field contributions from primitives within the same list are summed up, thus defining a convolution surface branch: $G_j(p) = \sum_{i}^{N_j} F_i(p)$ is the field contribution from the list j, where N_j is the number of line segments in the j-th list. The final implicit surface is a union of branches with the field computed as a union R-function of the field contributions from every list: $H_j = union(G_j(p), H_{j-1})$, where j = 1,...,N. If there is still unwanted blending between the non-branching parts of the skeleton (e.g., lower and upper arm), a user specified blending graph can be used.

5.2 Tuning of the initial approximation

After the initial approximation is generated, the user has the option to change it to achieve the desired result. This is needed sometimes because the rigging skeleton used for the mesh animation can be placed at arbitrary locations inside the mesh, making it more suitable for animation (Figs. 3a, 3b). As the field produced by the line segments is symmetrical, a better approximation of the mesh is achieved if the segments are placed closer to the medial axis of the particular mesh clusters. In this case, the segments of the convolution surface skeleton have to be moved away from the original bones positions of the rigging skeleton as shown in Figs. 3c, 3d. More specifically, the convolution line segments migrate from the rigging skeleton to the medial axis. The distance-based field produced by the extracted medial axis is used to determine the direction of migration for each vertex of the convolution line segments. The elements of the convolution surface skeleton are still defined relative to the joints of the rigging skeleton and follow them during the motion.

The fitting step has to be repeated after the skeleton migration. In case the user is not interested in the approximation of particular clusters of the mesh, some of the generated markers can be discarded.

5.3 The creation of a convolution surface for the initial polygonal mesh

The embedded convolution surface is created by using the segments of the skeleton produced in the above two steps. For rendering purposes we use a polygonization, which provides an approximation of the implicit surface as a polygonal mesh. For relatively simple skeletons the polygonization of the convolution surface can be obtained in near real-time. As the segments of the convolution skeleton are transformed relative to the transformation of the rigging skeleton, the motion of the convolution surface is synchronised with the motion of the animated mesh (Fig. 4).

We automatically perform the approximate convolution surface fitting only for the bind pose at the first frame of the animation. Thus, during the animation the bounding volumes and the convolution surface itself may not fit inside the mesh. This can happen because the distances between the mesh vertices and the bones change noticeably for those vertices that are influenced significantly by more than one joint. Such vertices are usually positioned near the skeleton joints. Performing fitting of the convolution parameters for each key-frame of the animation can be a lengthy process. This also means that each time the user adds a key-frame to the animation sequence the fitting procedure has to be repeated for these new frames. Thus, we let the user choose the key-frames for which refitting needs to be done - for instance, when the distance between the convolution surface and the bone exceeds the distance between the

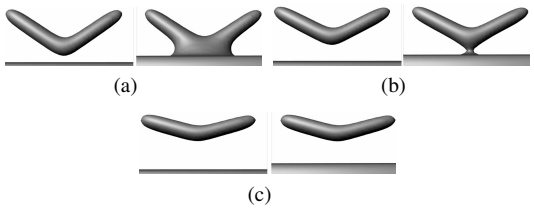


Figure 5: *Cases of object interaction without (left) and with (right) blending: (a) Two implicit surfaces and a single shape during blending, (b) Boundary case before two shapes disconnect, (c) Two separate shapes with some deformation showing objects' reciprocal attraction*

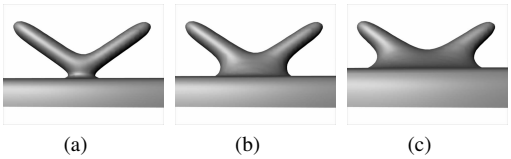


Figure 6: *Viscosity: (a) low, (b) medium, (c) high*

bone and the mesh. The re-estimated parameters are updated at the key-frames for the convolution primitives and then are interpolated during the animation sequence. This allows the user to concentrate on the process of mesh animation by decreasing the delays caused by the implicit surface re-fitting. Also, there is an opportunity for the user to assign custom values to the parameters of the implicit surface over time - for instance, to change the parameter controlling the overall surface radius. This can be used to achieve a desired artistic effect for a particular animation sequence.

5.4 The blending between the convolution surface and the viscous object

As the first application of our technique, we simulate the interaction of the viscous object with the animated object using the blending union of two implicit surfaces. As we mentioned above, the implicit surface corresponding to the initial mesh is an embedded convolution surface. The second implicit surface representing the viscous object can be modelled using a set of implicit primitives. The shape and the defining functions of both implicit surfaces are used to model the interaction between the two objects. If both defining functions have distance properties, the shape of the surface resulting from the blending operation depends on the distance between the original implicit surfaces. The further the objects are from each other the less they are deformed. There exist three main cases of object interaction: the "continuous interaction" when the two implicit surfaces form a single blend shape (see Fig. 5a), the separation of two objects (see Fig. 5b) and the objects' reciprocal attraction resulting in the directional deformation which decreases proportionately to the distance between the two objects (see Fig. 5c).

A blending union can dramatically change the resulting surface and its topology. As the result of the mutual deformation, a part of the convolution surface embedded within the mesh becomes visible contributing to the material interacting with the mesh. Thus, the quality of the initial approximation of the mesh by the convolution surface does not play a significant part in this application. It is more important just to fully embed the convolution surface into the mesh when no deformation is applied.

Modification of the blending parameters produces an effect visually mimicking the viscous object's physical parameter adjustment (Fig. 6). Thus, providing the user with the ability to emulate a specific phenomenon by modifying its set of parameters (for instance, by varying the liquid viscosity or gravitation instead of the three parameters of blending), as well as supplying predefined templates for different materials (such as tar, honey, oil, etc.) can help achieve an even more intuitive control over the interaction process.

5.5 Implicit parts for animated meshes

The proposed hybrid modelling technique involving animated meshes combined with implicit surfaces can be used not only to simulate viscoelastic objects but also to generate character models and other elements of the scene. In this case, the user creates and animates a usual skeleton made of line segments. One group of joints is used to deform the mesh and another directly specifies the form of resulting convolution surfaces. Convolution surfaces specified in this way are also driven by the skeleton animation. There is only a need to fit the implicit surface at the boundary attachment to the polygonal mesh, because in this case the surface is not hidden under the mesh. This approach might be suitable for effects that are very hard to achieve with conventional polygonal techniques (e.g., for modelling unusual characters made of liquids which can change the form of their limbs over time). In this case, no additional blending is required as these convolution surfaces are automatically blended with each other.

6 Implementation and results

In this section we describe the implementation of the proposed approach, present some experimental results and discuss areas of application.

6.1 The Maya plug-in

We have implemented the proposed approach as a plug-in for the Alias Maya 7.0 animation system. We have chosen Maya as it is a popular tool for modelling and animation used by a lot of professional artists. Our plug-in requires the user to specify both the skeletons and polygonal meshes, which are used to calculate the initial parameters of all the convolution surface skeletal primitives. Given these initial parameters, we can show the approximate bounding volumes for the convolution surface. This is done to provide the user with a finer control over the individual skeletal elements of the resulting convolution surface. Intermediate results of the implicit surface polygonization can be seen in the editor window (Fig. 7) in near real-time (the actual times for a number of experiments are shown in Table 1). The polygonization domain and the grid resolution for this operation can be modified by the user to control the quality of the resulting implicit surface and the time needed for its calculation. There is also an option to choose between different types of blending. In the case of bounded blending, the user can choose a shape specifying the region where the blending operation should be performed. Each parameter can be animated over time thus providing the user with more flexibility to produce various effects.

We have also used the NVIDIA CUDA SDK thus performing the computations on the GPU (Table 2). There are a number of improvements that could be made to enhance the performance of our basic GPU implementation. From our preliminary results, however, it is already apparent that this sort of task is ideally suitable for a parallel real-time implementation.

The integration of such functionality into an existing animation

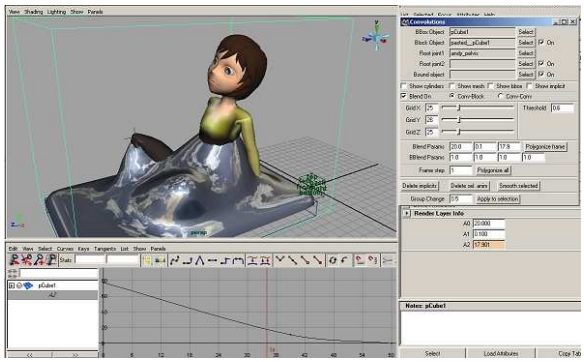


Figure 7: *A screenshot of the working environment*

package decreases the learning curve for the user. The user is free to produce an animation in the customary way within the familiar animation environment having the opportunity to see the expected results in near real-time. Thus, the incorporation of the plug-in in a general-purpose animation software package allows the user to easily integrate the produced animation into complex scenes developed using this package.

6.2 The interaction of an animated object with viscous liquid

The result of the interaction between an animated object and a viscous liquid is represented as an implicit surface. The user can preview intermediate results as a low- or medium-resolution polygonization of the implicit surface in near real-time (see the grey surface in Fig. 8a). A high-resolution polygonization grid can then be used for the final offline rendering with the additional application of complex material and shading properties (Fig. 8b). The following algorithm is used to generate an animation sequence:

- Calculate the positions of the convolution skeletal primitives based on the joint transformation matrix and the field parameters.
- Calculate the field function produced by the blending between the set of convolution surfaces and the implicit viscous object.
- Polygonize the resulting implicit surface.
- Render the initial deformed mesh and the polygonized implicit surface.

6.3 Bounded blending applied to two animated meshes

A viscous object can also be defined by an animated mesh. In this case, it can be approximated by a convolution surface. The interaction between two animated meshes with embedded implicit surfaces is shown in Fig. 9. Here, the operation of bounded blending [Pasko et al. 2005] is applied to localise the blending surface inside the specified region.

6.4 "Supra-natural" liquid behaviour

Another possible application of this technique is the modelling of a special "live" liquid covering the animated meshes (Fig. 10). In such animations the convolution radius is increased over time, which creates the effect of the liquid flowing up the mesh and completely engulfing it. It is possible to automatically generate this sort

Grid Resolution for polygonization	Cactus (11 segments)	Andy (45 segments)	Hybrid Andy (10 segments)
20x20x20	25 ms	80 ms	30 ms
30x30x30	80 ms	220 ms	60 ms
50x50x50	310 ms	930 ms	260 ms
70x70x70	810 ms	2580 ms	670 ms

Table 1: *Average time for mesh generation (milliseconds/frame) on PC with Dual Core Intel Xeon (2.66 GHz), 2 GB of RAM; Andy is a mesh model with an embedded convolution surface (Fig. 8a), Hybrid Andy is a mesh with an attached implicit surface limb (Fig. 11)*

Grid Resolution for polygonization	Cactus (11 segments)	Andy (45 segments)	Hybrid Andy (10 segments)
32x32x32	10 ms	15 ms	10 ms
64x64x64	30 ms	75 ms	30 ms

Table 2: *Average time for mesh generation (milliseconds/frame) on an NVIDIA GeForce 8800 Ultra, 768 MB of RAM*

of animation. The user just needs to specify the first and last joint of the skeletal chain as well as the final thickness of the liquid flowing over the mesh.

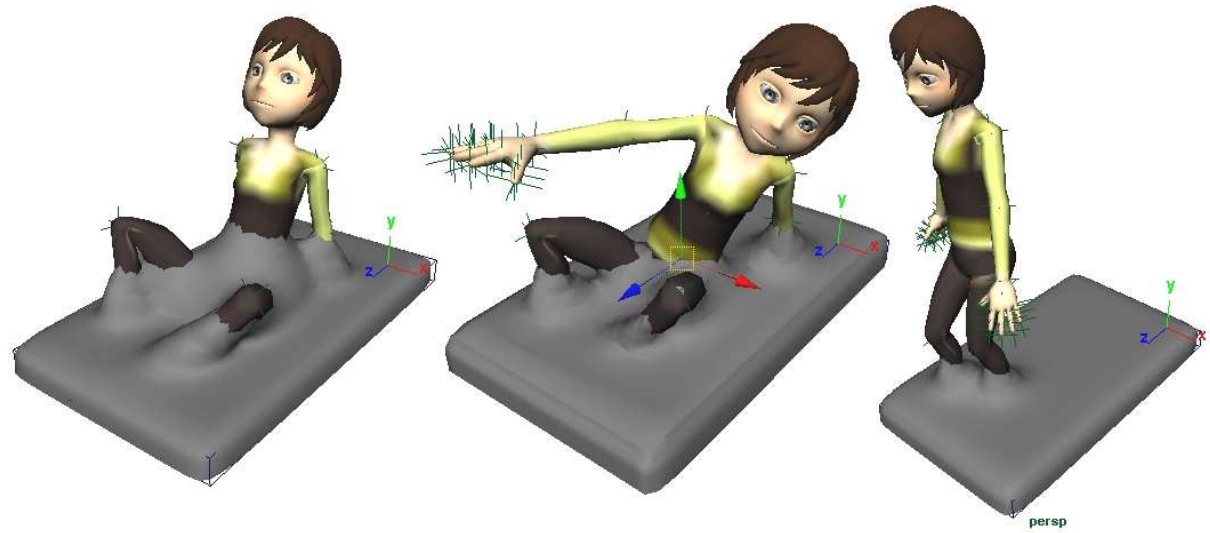
6.5 Animated mesh character with an implicit limb

Here we illustrate how the proposed technique can be used for the creation of easily metamorphosing parts for animated characters. The example shown in Fig. 11 deals with a mesh animated with a rigging skeleton. One of the character's limbs that is represented by convolution surfaces is also animated by a part of the skeleton. Thus, it is possible to dramatically vary the shape of this limb. At first, the limb is just a blob, subsequently a hand grows out of it, then it metamorphoses into a sabre and finally transforms into a hammer. The whole animation can be generated in real-time.

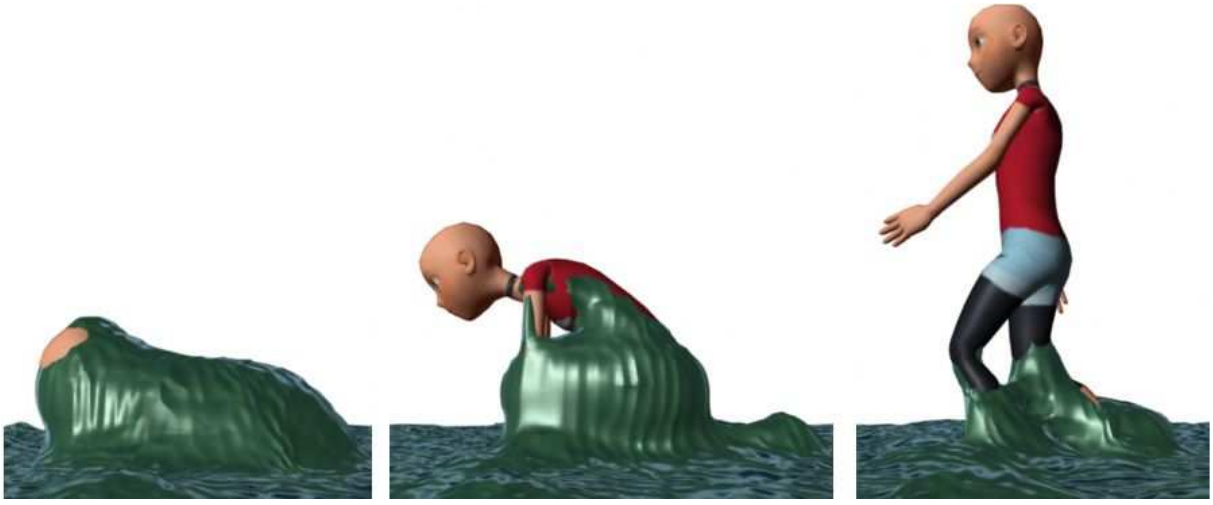
6.6 Discussion

There are several issues that require additional consideration and can be addressed in future work. The first is concerned with the fact that the applied blending operation is based on the distance properties of the functions defining its arguments. The scalar fields produced by known convolution surface kernels significantly decrease with the growth of the distance from the line segment. At a particular distance from the line segment the values of such a field are almost equal to zero and no blend shape is generated at these distances by the blending operation. Thus, it is hard to model the interaction of the mesh and the viscous object at large distances. In such cases, an ellipsoidal approximation of the mesh could provide better results. Another limitation related to the blending surface is that of texturing, as texture matching for dynamic implicit surfaces is still an open research question.

As the distance between the two blended objects increases, the deformation of convolution surfaces decreases until these surfaces are again embedded into the polygonal mesh and are no longer visible. The proposed method does not allow us to easily model the separation of droplets of the viscous liquid from the mesh. If this effect is desired, some additional particles modelling this effect could be attached to the mesh. It is also possible to add particles to the viscous object. These can improve the visual quality and dynamism of the resulting image. Simplified particle based physical models can be applied to the implicit model to improve the default behaviour of the viscous object. A metaball representation of the particles is



(a) Preview of intermediate results as seen in the tool window



(b) Offline rendering with complex materials

Figure 8: *The interaction of an animated object with viscous liquid*

Figure 9: *Bounded blending of animated meshes*

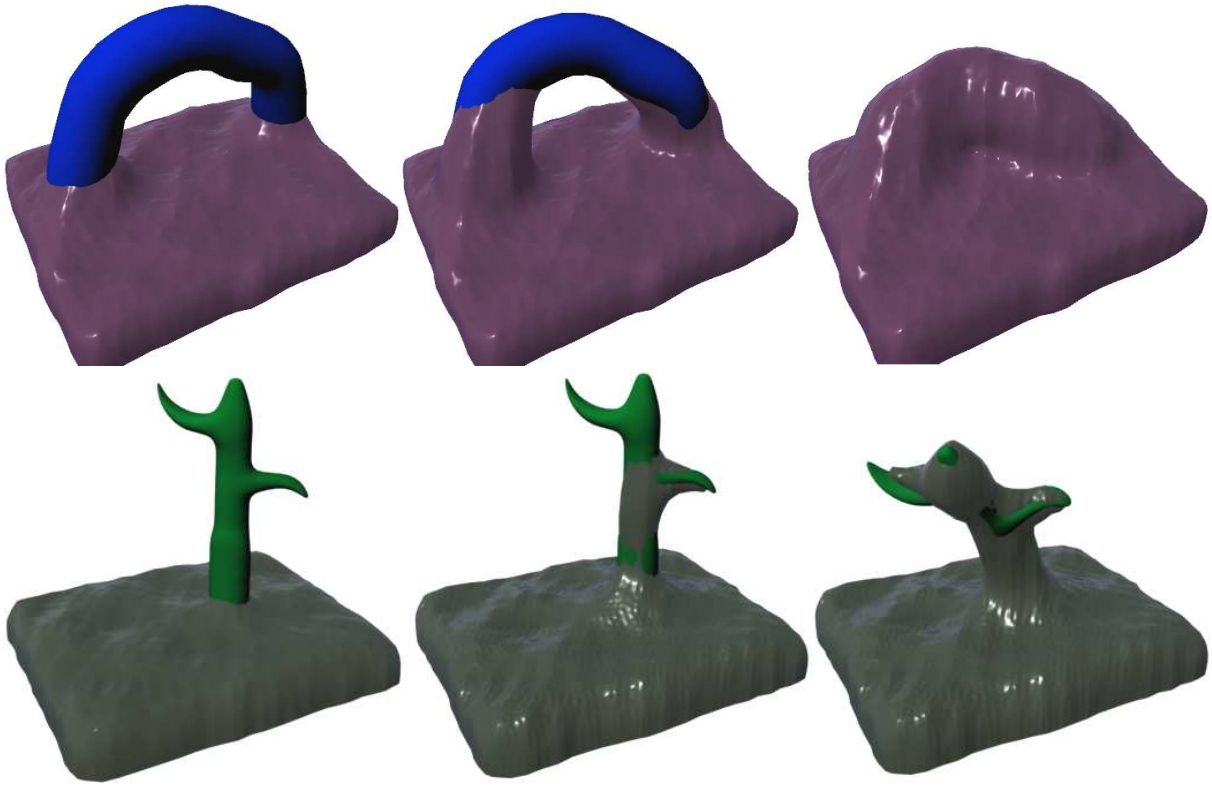


Figure 10: *Liquid covering animated mesh*

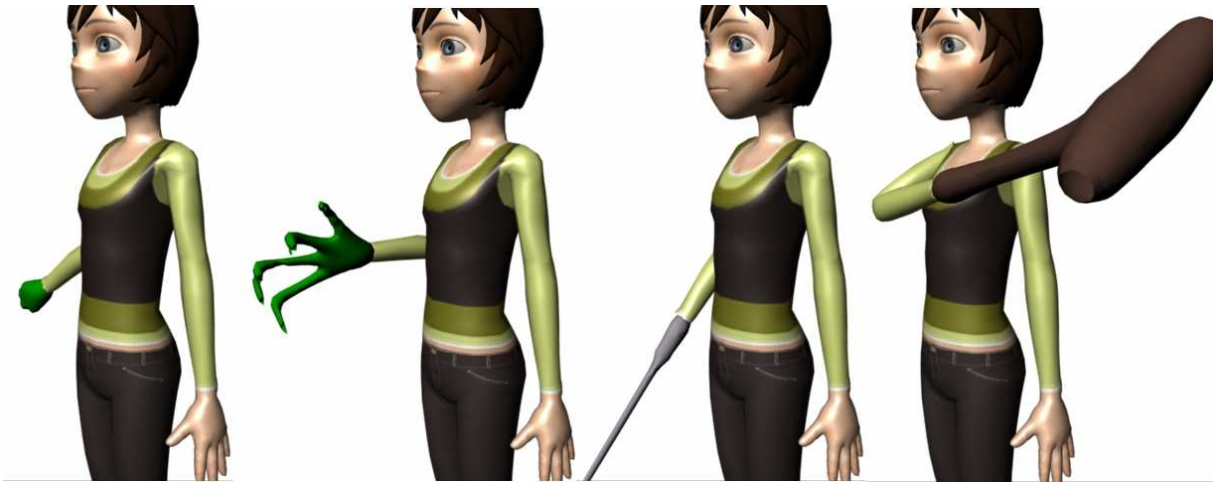


Figure 11: *Hybrid model with a polygonal body and an attached "metamorphosing" implicit limb*

frequently used to integrate these particles into the implicit model [Clavet et al. 2005].

If the size of the generated convolution surface is not significantly smaller than the size of the object modelling the viscous material, volume preservation techniques may have to be used [Cani-Gascuel and Desbrun 1997]. We can estimate the amount of intersection between the bounding volumes of the skeletal primitives and the viscous object. This estimation can then be used to adjust the blending union parameters. These parameters influence the strength of the deformation which leads to changes in the resulting surface volume.

In the research presented in this paper we were mainly concerned with embedding an implicit surface inside a mesh that defines the moving object, but in fact other criteria of fitting could be used. The criteria could be chosen depending on the specific animation effect desired and the particular requirements of the user. For instance, there might be a need for the implicit surface to completely cover the mesh or only certain parts of the mesh.

For particular asymmetrical parts of the model (e.g., the hips or feet of human characters) a better approximation could be achieved by using line segments producing anisotropic fields [Tai et al. 2004]. This would significantly increase the time needed both to perform the fitting operation and to calculate the final field produced by a set of line segments. In any case, the ability to perform this type of fitting could be useful for the production of specific animation effects. Another useful option is to let the user draw an outline curve for the blend shape between the implicit surfaces and then to estimate the parameters of the blending fitting this curve.

7 Conclusions

In this paper we have proposed a method for hybrid modelling involving animated meshes combined with implicit surfaces. An implicit convolution surface is built around the rigging skeleton of the animated mesh and is then embedded inside or attached to this mesh. This method allows for a high level of control over the animation of both the mesh and the implicit surface components of the model.

We have applied this approach to model the adhesive behaviour of viscoelastic objects in their interaction with moving surfaces. The physical effect of adhesive coating of moving surfaces by liquids is modelled using geometric blending between the approximations of the animated surface meshes by implicit surfaces. Another application is concerned with the augmentation of animated meshes by attaching implicit surface parts to them.

The proposed method is based on purely geometric properties and operations on the interacting objects. In contrast to known fluid dynamics techniques which are based on physical simulation, our technique is not computationally expensive and allows for near real-time preview using a polygonization of the resulting isosurface. This technique can be employed in a conventional animation pipeline with near real-time preview. It might even be suitable for real-time applications such as computer games where visual results and low computation times are more important than physical correctness. Our prototype implementation provides the user of a commercial animation system with a number of parameters to specify the desired behaviour of the animated hybrid model. Our work shows that not only simple implicits, such as blobs/metaballs, but more complex implicit surfaces are useful in computer animation and games. In particular, we have shown that hybrid models including convolution surfaces show great promise for modelling dynamic effects.

References

ALLÈGRE, R., GALIN, E., CHAINE, R., AND AKKOUCHE, S. 2006. The hybridtree: mixing skeletal implicit surfaces, triangle meshes, and point sets in a free-form modeling system. *Graph. Models 68*, 1, 42–64.

ANGELIDIS, A., AND CANI, M.-P. 2002. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications*, ACM, 45–52.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3, 72.

BARBIER, A., GALIN, E., AND AKKOUCHE, S. 2005. A framework for modeling, animating, and morphing textured implicit models. *Graph. Models 67*, 3, 166–188.

BLINN, J. F. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3, 235–256.

BLOOMENTHAL, J., AND SHOEMAKE, K. 1991. Convolution surfaces. *SIGGRAPH Comput. Graph. 25*, 4, 251–256.

CANI-GASCUEL, M.-P., AND DESBRUN, M. 1997. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics 3*, 1, 39–50.

CANI-GASCUEL, M.-P. 1998. Layered deformable models with implicit surfaces. In *Graphics Interface*, 201–208.

CLAVET, S., BEAUDOIN, P., AND POULIN, P. 2005. Particle-based viscoelastic fluid simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 219–228.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 23–30.

GALIN, E., LECLERCQ, A., AND AKKOUCHE, S. 2000. Morphing the BlobTree. *Comput. Graph. Forum 19*, 4, 257–270.

JIN, X., TAI, C.-L., FENG, J., AND PENG, Q. 2001. Convolution surfaces for line skeletons with polynomial weight distributions. *J. Graph. Tools 6*, 3, 17–28.

JIN, X., LIU, S., WANG, C. C. L., FENG, J., AND SUN, H. 2005. Blob-based liquid morphing: Natural phenomena and special effects. *Comput. Animat. Virtual Worlds 16*, 3-4, 391–403.

KATZ, S., AND TAL, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *ACM Transactions on Graphics (Proc. SIGGRAPH'03)*, ACM, 954–961.

LIU, P.-C., WU, F.-C., MA, W.-C., LIANG, R.-H., AND OUHYOUNG, M. 2003. Automatic animation skeleton construction using repulsive force field. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, 409–413.

MCCORMACK, J., AND SHERSTYUK, A. 1998. Creating and rendering convolution surfaces. *Comput. Graph. Forum 17*, 2, 113–120.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Transactions on Graphics (Proc. SIGGRAPH'03) 22*, 3, 463–470.

OPALACH, A., AND MADDOCK, S. C. 1995. High level control of implicit surfaces for character animation. In *Proc. 1st International Eurographics Workshop on Implicit Surfaces*, 223–232.

OSHER, S., AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer.

PASKO, G., PASKO, A., AND KUNII, T. 2005. Bounded blending for function-based shape modeling. *Computer Graphics and Applications, IEEE 25*, 2, 36–45.

SHEN, J., AND THALMANN, D. 1995. Interactive shape design using metaballs and splines. In *Implicit Surfaces'95*, 187–196.

SHI, L., AND YU, Y. 2005. Taming liquids for rapidly changing targets. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, 229–236.

SINGH, K., AND PARENT, R. 1995. Implicit function based deformations of polyhedral objects. In *Proc. 1st International Eurographics Workshop on Implicit Surfaces*, 113–128.

TAI, C.-L., ZHANG, H., AND FONG, J. C.-K. 2004. Prototype modeling from sketched silhouettes based on convolution surfaces. *Comput. Graph. Forum 23*, 1, 71–84.

THÜREY, N., KEISER, R., PAULY, M., AND RÜDE, U. 2006. Detail-preserving fluid control. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 7–12.

TIERNY, J., VANDEBORRE, J.-P., AND DAOUDI, M. 2006. 3d Mesh Skeleton Extraction Using Topological and Geometrical Analyses. In *14th Pacific Conference on Computer Graphics and Applications*, 85–94.

TURK, G., AND O'BRIEN, J. F. 1999. Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 335–342.

WEI, X., LI, W., AND KAUFMAN, A. 2003. Melting and flowing of viscous volumes. In *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, IEEE Computer Society, 54–59.

WYVILL, B., MCPHEETERS, C., AND WYVILL, G. 1986. Animating Soft objects. *The Visual Computer 2*, 4, 235–242.