Volume 29 (2010), Number 1, pp. 128–140

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

D. Kravtsov, O. Fryazinov, V. Adzhiev, A. Pasko, P. Comninos

The National Centre for Computer Animation, Bournemouth University, UK

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. We model the interaction between an animated mesh object and a viscoelastic substance, which is normally represented in an implicit form. Our approach is aimed at achieving verisimilitude rather than physically based simulation. 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.

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 [ST95], in the simulation of natural phenomena [WLK03], in the simulation of object cracking and explosions [BGA05], and in geometric operations (such as offsetting, blending and metamorphosis [PASS95]).

Hybrid models combining different geometric representations, for example polygonal meshes and implicit surfaces [AKK*02, AGCA06], 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 to make it visually behave as an implicit surface. 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. For the creation of a continuous scalar field around an animated mesh, we consider an embedded implicit surface (meaning a surface completely hidden inside the mesh) synchronously moving with it and used only when special effects involving scalar fields are required. An example of such an effect is the blending between the embedded implicit surface and another implicit surface representing a viscous object. One form of combining a mesh with an implicit surface would be to attach the implicit surface to it. An attachment area is a surface patch which is shared by both the implicit surface and the mesh surface [AKK*02].

Polygonal meshes and certain types of implicit surfaces can be animated using a rigging skeleton. We consider a skeleton as a proper base for their integration into hybrid models. There are many candidates for such integration among implicit surfaces, namely soft objects, distance-based blobs, ellipsoids, convolution surfaces, constructive solids built of cylinders, spheres, and other primitives. The main requirements for an implicit surface are: a relatively simple defining function, which is fast to evaluate, easy manipulation us-

submitted to COMPUTER GRAPHICS *Forum* (9/2009).

ing skeletons and an absence of buldges and other unwanted artefacts, which require additional processing. We have selected convolution surfaces [BS91, MS98] for this purpose, as they satisfy all these requirements, however other types of implicit surfaces can be also utilized within the proposed approach.

We propose to embed an implicit convolution surface inside an animated mesh or to attach it to the mesh such that the motions of both types of object are synchronised. 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, which can be achieved using a global minimization of the overall algebraic distance of the mesh nodes to the convolution surface. Interaction of a viscous object with an animated object is modelled using geometric blending operations on the corresponding implicit surfaces. 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.

The main contributions of this paper are the following: (1) a hybrid model combining skeleton driven animated meshes with skeleton-based implicit surfaces, (2) a procedure for fitting a convolution surface to an animated mesh and (3) applications of such hybrid models for the simulation of viscous object behaviour. Some of these applications call for blending between mesh models animated using conventional skeleton techniques. We also present another application concerned with the creation of easily metamorphosing parts for animated characters.

In the areas of computer animation and digital special effects production it is a well established fact that physically correct simulation is often an inappropriate technique to use, as it often interferes with the intended development of the narrative. Consider for instance the following two examples which may arise in such productions. In a cartoon-style animation, a character may step over the edge of a precipice but he does not fall immediately as would be physically correct. He remains suspended in mid-air until he realises that the ground no longer supports his weight and only then he begins to fall. Similarly in a digital effects sequence, a digital stand-in character may leap from building to building in defiance of gravity or a column of water is suspended in mid-air, morphs into a human face and starts talking to a real actor. In such scenarios what animators are looking for is not physical correctness of the event or phenomenon, but for some form of believable semblance of reality (i.e., verisimilitude), which is inspired by physical reality but bends this reality to allow them to advance the story narrative. Instead of a physically correct simulation what is required is a set of techniques and tools that would allow the animator to alter, to tune and to cheat reality. These may be physically inspired, but not physically correct, and must be able to be directable by the animator, so that they produce the desired visual effects. Physically-correct simulation techniques can often be combined with physically-inspired verisimilitude techniques but they must be directable by the artist and subordinated to the story-telling process. Additionally when such techniques are used in the development stage of computer animation and digital effects sequences or in a computer game they must produce believable visual results in real-time or near-real-time.

Having a good understanding of computer animation production requirements, we aim to provide the animator with a simple tool based on purely geometric methods which allows the creation of complex animations satisfying the specified requirements.

2. Related works

In this section we discuss two groups of publications, the first group relates to the general usage of implicit surfaces in computer animation and the second is concerned with specific applications of viscous objects simulation.

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 [Bli82], where the transformation of the blob is inherited from the transformation of the joints of the skeleton. In [OM95], blobby objects were used, which are quite easy to define. However, with this method it is difficult to control the resulting "blobby" mesh. In addition, a large number of primitives is usually needed to model an appropriate mesh.

One of the earliest attempts of using hybrid modelling involved embedding mesh objects into implicit surface primitives [SP95] to implement polyhedral object deformations of articulated deformable bodies. Skeleton-based implicits for non-polygonal animated objects were examined, for instance in [CG98], where skeletal geometric primitives that produce 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. Mixing of implicit surfaces and polygonal models was performed in [LAG01]. In this work specific regions of an animated mesh were deformed using implicit primitives attached to the animated skeleton. Polygonal meshes and implicit primitives were also combined together in a HybridTree [AGCA06] using blending, Boolean and other operations supported by

the conversion procedures between two different models. However, embedding, attachments and skeleton-based motion synchronisation of meshes and implicits as well as an implementation in a general animation system were not directly addressed.

Implicit surfaces were also used for the approximation of polygonal meshes using different approaches, such as Radial Basis Functions (RBFs) [SPOK95] and Multi-level Partition of Unity implicits (MPUs) [OBA*03]. These methods generally work well with static meshes, but are less suitable for animation because dynamic models require per-frame refitting and can not be easily edited by the user due to the complicated handling of the implicit surface.

One of the interesting alternatives among implicit surfaces is that of convolution surfaces [BS91]. Convolution surfaces can be smoothly blended with each other and provide a good approximation for polygonal meshes typical for skeletal characters with axial symmetry [MS98]. In this paper we utilise convolution surfaces with line segment skeletons for hybrid modelling applications.

2.2. Interaction of viscous objects with meshes

The main technique for modelling viscous objects is fluid simulation. Thus, in [FF01] 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, [TKPR06] describes 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 and do not consider the animation of the mesh. The approach described in [CBP05] uses smoothed particle hydrodynamics and additional physical simulation to model viscoelastic objects and their interaction with rigid bodies. This approach is not well suited for the modelling of interactions with animated polygonal objects and it may require a significant number of particles for the simulation of liquid substances. In [JLW*05] 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 presented mesh fitting is a lengthy offline process. This approach also does not allow for the metamorphosis between animated meshes.

[MTPS04] proposed a method to control fluid simulations through a number of keyframes supplied by the user. Interactions of viscoelastic materials with the meshes was performed in [GBO04]. Two-way coupling between rigid bodies and fluids was implemented in [CMT04]. This allows for the creation of realistic animations involving interactions of solid objects with fluids. The drawbacks of the methods are: a lengthy simulation process and poor artistic control (i.e. directability) of the resulting effects.

The authors of [SY05] 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 and the animator is required to manually tune the physical parameters of the simulation for different types of meshes and animation sequences. Thus, even the initial tuning can be time consuming.

It is often the case that a combination of different techniques is used for the emulation of viscous materials. In our work we simulate objects composed of viscous materials using a geometric blending between the implicit objects generated from given polygonal meshes. We aim to provide the user with a simple tool which allows the creation of complex animations with convincing visual results in real-time or near-real-time.

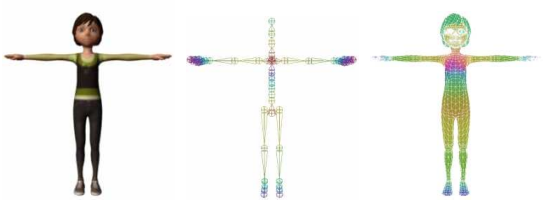
3. Problem statement and approach outline

In this paper we address problems that appear in computer animation practice and are difficult or impossible to solve using polygonal meshes exclusively. We particularly examine problems associated with the creation of effects such as objects blending using scalar fields associated with polygonal meshes and the modelling of freely transformable objects involving meshes with implicit surface components. Our approach relies upon hybrid modelling combining polygonal meshes with implicit surfaces. In general, there are three main ways of achieving this: (a) by coating of meshes with implicit surfaces, (b) by embedding implicit surfaces inside mesh objects, (c) by attaching external implicit surfaces to mesh surfaces. Coating was discussed in [CG98] (see the previous section). In this paper, embedding is applied in order to achieve blending effects and attaching is used in order to construct metamorphosing parts of hybrid models. 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 (Fig. 1a), with a rigging skeleton (Fig. 1b), skinning information (Fig. 1c) and a set of animation transformations for its skeletal nodes. A rigging skeleton is a set of hierarchically connected joints used to specify the motion of a mesh model in an animation sequence. If there is no skeleton provided, it can be automatically extracted from the polygonal mesh using one of the published techniques [BP07].

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 replace the mesh object with an implicit surface stand-in and then to apply geometric blending between the implicit surfaces representing both interacting objects.

A viscoelastic object can be represented either by an implicit surface or by another polygonal mesh (which has to be converted to an implicit surface). We will mainly concentrate on



(a) (b) (c)

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

the former case to simulate such viscous substances 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 will use this property for modelling the adhesive behaviour. This, in general, assumes the conversion of the animated mesh into an implicit surface. However, an exact conversion of this type 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 the 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 fitting procedure provides the convolution surface with a minimal distance measure to the mesh. This is required in order to create the visual effect of the mesh being blended with the viscous liquid. The resulting convolution surface can be polygonized to obtain a near real-time preview or can be ray-traced to produce the final animation sequence.

4. Background

In this section we describe the 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 [BS91]. Given a scalar field function f, such as:

$$f(\mathbf{p}) = \int_S g(\mathbf{r})h(\mathbf{p}-\mathbf{r})d\mathbf{r}; \mathbf{p}, \mathbf{r} \in R^3$$

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 the primitives (i.e. the skeleton function), $h(\mathbf{p})$ is a kernel function and $\mathbf{p}$ is an arbitrary point in space. 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 [MS98]. Thus,

$$h(d) = \frac{1}{(1+s^2d^2)^2}; d > 0$$

where d denotes the Euclidean distance from a point of interest and s is a scalar value controlling the convolution surface.
Given a line segment

$$\mathbf{r}(t) = \mathbf{b} + t\mathbf{a}; 0 \le t \le l$$

where $\mathbf{b}$ is the segment base (poistion vector), $\mathbf{a}$ is the segment axis (direction 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:

$$d^2(t) = |\mathbf{q}|^2 + t^2 - 2t(\mathbf{q}\cdot\mathbf{a})$$

where $\mathbf{q} = \mathbf{p} - \mathbf{b}$.
A field function for an arbitrary point $\mathbf{p}$ would be:

$$f(\mathbf{p}) = \int_0^l \frac{dt}{(1+s^2d^2(t))^2} = \frac{x}{2m^2\left(m^2+s^2x^2\right)} + \frac{l-x}{2m^2n^2} + \frac{1}{2sm^3}\left(\arctan\left[\frac{sx}{m}\right] + \arctan\left[\frac{s(l-x)}{m}\right]\right)$$

where x is the coordinate on the segment's axis,

$$x = (\mathbf{p}-\mathbf{b})\cdot\mathbf{a}$$

$$m^2 = 1 + s^2(q^2 - x^2)$$

$$n^2 = 1 + s^2(q^2 + l^2 - 2lx)$$

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

4.2. The blending union of implicit surfaces

Given two implicit surfaces,

$$f_1(x,y,z) = 0 \text{ and } f_2(x,y,z) = 0\,,$$

the blending operation between these surfaces is defined as [PASS95]:

$$blend(f_1, f_2) = union(f_1, f_2) + disp(f_1, f_2),$$

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. Here a_0 controls the overall resulting shape, a_1 and a_2 specify the contributions to the blending by the shapes defined by functions f_1 and f_2.

Using R-functions the formula for the following blending operation can be obtained as:

$$blend(f_1, f_2) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + \left(\frac{f_1}{a_1}\right)^2 + \left(\frac{f_2}{a_2}\right)^2}$$

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 use a bounded blending operation [PPK05].

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 skeletal information.
2. The tuning of the initial approximation.
3. The creation of an embedded convolution surface for the initial polygonal mesh.
4. One of two application steps: (a) the definition of the blending between the convolution surface and the viscous object for the modelling of the adhesive behaviour of this viscoelastic object and its interaction with an animated object or (b) 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

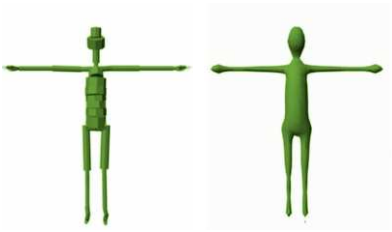
As was mentioned earlier, the procedure for embedding an implicit surface inside the mesh requires a global minimisation of the algebraic distance measure between the mesh nodes and the convolution surface.

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 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 and end vertices for each such a skeletal 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 skeletal bones $\mathbf{s_i} \in \mathbf{S}$ (where $\mathbf{S}$ is a set of skeletal bones) the radius of the i-th cylinder associated with the i-th bone is:

$$r_i = \min_{\mathbf{p_j} \in \mathbf{P}} (dist(\mathbf{s_i}, \mathbf{p_j}))\,,$$

where $\mathbf{p_j}$ is the j-th face of the polygonal mesh, $\mathbf{P}$ is a connected set of faces and $dist(\mathbf{s_i}, \mathbf{p_j})$ is the distance between the bone $\mathbf{s_i}$ and the face $\mathbf{p_j}$. Thus, each bounding volume is fitted inside the mesh in its initial position. Rendering these bounding volumes helps the user to better understand how the resulting approximating convolution surface is embedded into the mesh (Fig. 2).

In the next step we perform a global optimisation to achieve a better approximation of the given polygonal mesh using the embedded convolution surface. A similar problem in 2D space was addressed in [TZF04] 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 embedment into the polygonal mesh. In our formulation
$F(\mathbf{p}) = \sum_{i=1}^{N} f_i(\mathbf{p})$ is a field produced by all the line segments at a point p;
$\mathbf{V} = \{\mathbf{v_1}, .., \mathbf{v_n}\}$ are mesh vertices;
$\mathbf{C} = \{\mathbf{c_1}, .., \mathbf{c_m}\}$ are the centroids of mesh triangles;



(a) (b)

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

$\mathbf{D} = \{\mathbf{d_1}, .., \mathbf{d_k}\}$ are points on the mesh closest to the skeleton line segments;
$\mathbf{P} = \mathbf{G}(\mathbf{V} \cup \mathbf{C} \cup \mathbf{D}) = \mathbf{G}(\mathbf{v_1}, .., \mathbf{v_n}, \mathbf{c_1}, .., \mathbf{c_m}, \mathbf{d_1}, .., \mathbf{d_k})$ are filtered estimation points; where

$$\mathbf{G}(\mathbf{p_1}, .., \mathbf{p_{n+m+k}}) = \mathbf{G}(\mathbf{p_1}) \cup .. \cup \mathbf{G}(\mathbf{p_{n+m+k}}) = \{\mathbf{p'_1}, .., \mathbf{p'_M}\}$$

$$\mathbf{G}(\mathbf{p_i}) = \begin{cases} \emptyset, F(\mathbf{p_i}) < T \\ \mathbf{p_i}, F(\mathbf{p_i}) \geq T \end{cases}$$

The filtering of the estimation points helps to reduce the dimensionality of the task. This reduces the processing time needed for the optimisation procedure. We apply the optimisation considering only the points on the mesh which are situated inside the convolution surface (i.e., in regions where the initial convolution surface is not embedded into the mesh).

Given that,
$\mathbf{L} = \{\mathbf{l_1}, .., \mathbf{l_h}\}$ are convolution line segments;
$\mathbf{F_i} = [f_i(\mathbf{p'_1}), .., f_i(\mathbf{p'_M})]^\top$ 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 the 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. Thus we need to solve the constrained least-squares problem:

$$\min_{0 \leq \Lambda \leq 1} (\mathbf{F}\Lambda - \mathbf{T})^\top (\mathbf{F}\Lambda - \mathbf{T})$$

for the unknown weights of field contribution from each line segment $\Lambda = [\lambda_1, .., \lambda_N]^\top$. The constraint $\lambda_k \geq 0$ is given for the topological correctness of the resulting convolution surface. The constraints $\lambda_k \leq 1; k = 1...N$ are intended to ensure that the convolution surface is completely embedded into the mesh (at least at all the estimation points) and that the conditions of the initial approximation were not violated. We also take into account the embedding constraint $F(p) \leq 0$ to ensure the convolution surface is completely inside the mesh.

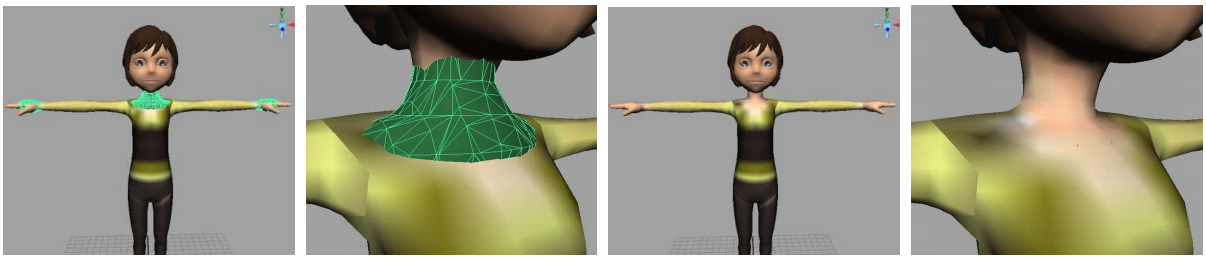
We need to apply a numerical search in the N-dimensional space of the parameters Λ to minimise the above least squares criterion with the given box constraints. We use the constrained Levenberg-Marquardt method [KYF05] to solve this problem. The constrained optimisation in the general case may worsen the quality of the approximation, for instance, it may increase the average distance between the mesh and the convolution surface. This may happen because the optimisation will only cause a decrease of the line segment weights thus decreasing the volume of convolution surface in order to embed it into the mesh. If this effect is undesirable, the segments with decreased weights can be further subdivided and the optimisation procedure repeated. In such a case it is possible to improve the quality of the approximation. Final field produced by the weighted line segments is defined as follows:

$$F(\mathbf{p}) = \sum_{i=1}^{N} f_i(\mathbf{p})\lambda_i$$

Usually the tuning procedure needs to be performed only once for the character's bind pose (Fig. 3). In our experiments, it took between 10 and 45 seconds for this operation. The time required depends on the number of vertices in the mesh and the number of skeletal branches. We perform tuning only in "problematic areas" (i.e. locations in proximity to skeletal branches where the implicit surface is not embedded into the mesh). This helps to significantly reduce the number of estimation points and leads to a speedup of the fitting procedure. It is also possible to reduce the number of line segments whose parameters are tuned. This applies to line segments which do not produce a significant contribution in the "problematic areas". Usually there are only between 5% and 10% of line segments satisfying this criterion. If there is a need to perform additional tuning during some phases of the animation, the parameters retrieved after fitting are interpolated between those frames.

There are default parameters for vertex filtering, which is a search for "problematic areas", and default parameters for the penalty function, but the user has the option to change both if he/she is not satisfied with the result. For example, the user can control how "deep" the implicit surface is embedded inside the volume described by the mesh. We also let the user control parameters for each line-segment or a group of line segments manually if needed. For some effects this tuning step is not needed at all.

In our current implementation, we approximate the mesh using a convolution surface with a constant radius along a line segment. If there are large variations in distance between the mesh and its medial axis, a better approximation could be achieved by using weighted convolution surfaces [JTFP01]. 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 an approximation would decrease the median distance between



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

Figure 3: *Approximation results: (a,b) Before the optimisation with the implicit surface outside the mesh in the "problematic areas" (c,d) After the optimisation the fully embedded implicit surface.*

the convolution surface and the polygonal mesh. It is worth observing that, if we wish to apply the blending union operation described in section 4.2, the quality of the initial approximation does not play a significant part in this process.

The problem of unwanted blending between different parts of the convolution surface can also be partially solved using a union operation based on R-functions. Separate lists of line segments are created for branching skeletal structures. The field contributions from primitives within the same list are summed up, thus defining a convolution surface branch:

$$G_j(\mathbf{p}) = \sum_{i}^{N_j} f_i(\mathbf{p})\lambda_i$$

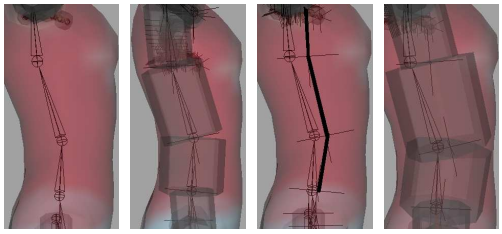
This is the field contribution from the list j, where N_j is the number of line segments in the j-th list. The final implicit surface is the union of branches with the field computed as an R-function union of the field contributions from every list:

$$H_j(\mathbf{p}) = union(G_j(\mathbf{p}), H_{j-1}(\mathbf{p}))\ ,$$

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 modify 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 volume described by the mesh, making it more suitable for animation (Figs. 4a, 4b). 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 bone positions of the rigging skeleton as shown in Figs. 4c, 4d. More specifically, the convolution line segments migrate from the rigging skeleton to the medial axis (a 1D curve skeleton). The distance-based field produced by the extracted curve skeleton is used to



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

Figure 4: *Tuning of the initial approximation: (a) Original mesh with the rigging skeletal bones, (b) Initial placement of bounding volumes, (c) Migration of the convolution skeleton, (d) Bounding volumes around the produced convolution surfaces*

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 their movement.

This 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 convolution segments can be discarded.

5.3. The creation of a convolution surface

The embedded convolution surface is created by using the segments of the skeleton produced in the above two steps. For the rendering purposes we use a polygonization procedure, 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. 5).

We automatically perform the approximate convolution sur-

Figure 5: *Synchronised motions of the embedded convolution surfaces during animation*

face 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 could happen because the distances between the mesh vertices and the bones change noticeably for those vertices that are influenced significantly by more than one joints. 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 time-consuming 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 take place - for instance, when the distance between the convolution surface and the bone exceeds the distance between the bone and the mesh. The re-estimated parameters are updated at the key-frames for the convolution primitives and then are interpolated during the playback of 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. Applying the blending operation

As the first application of our technique, we simulate the interaction of a viscous object with an 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. 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 phases of object interaction: the "continuous interaction" when the two implicit surfaces form a single blend shape (see Fig. 6a), the "separation of two objects" (see Fig. 6b) and "the objects' reciprocal attraction" resulting in the directional deformation which decreases proportionately to the distance between the two objects (see Fig. 6c).

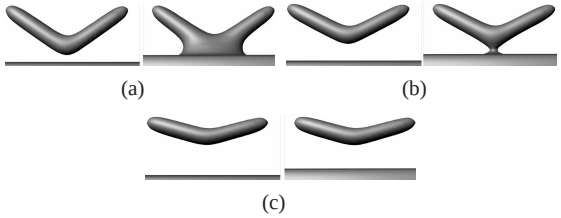


Figure 6: *Phases of interaction between animated objects without (left) and with (right) blending: (a) Two implicit surfaces and a single blend shape during blending, (b) The boundary case before two shapes separate, (c) Two separate shapes with some deformation showing the objects' reciprocal attraction*

Figure 7: *Viscosity: (a) low, (b) medium, (c) high*

A blending union can dramatically change the resulting surface and its topology. As a result of the mutual deformation, a part of the convolution surface embedded within the mesh becomes visible thus 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. 7). Thus, the user can control a specific phenomenon by modifying a meaningful set of parameters. Then, instead of using those abstract parameters of geometric blending (see sec. 4.2), the user can operate with intuitive parameters representing liquid viscosity or gravitation. A set of predefined templates for different materials (such as tar, honey, oil, etc.) allows the user to achieve easier control over the interaction process.

6. Implementation and results

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

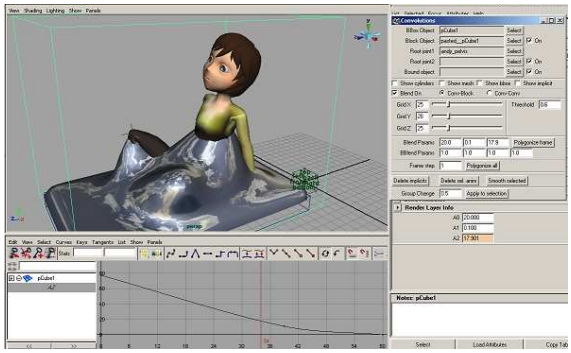


Figure 8: *A screenshot of the working environment*

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 skeletal primitives of the convolution surface. 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. 8) 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.

6.2. The interaction of an animated object with a 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 (Fig. 8). 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. 9). 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 original deformed mesh and the polygonized implicit surface.

The example shown in Fig. 10 deals with a mesh animated with a rigging skeleton, while some parts of the character are represented by convolution surfaces (Fig. 12).

Figure 9: *Adhesive behaviour of the viscous object*

6.3. "Supra-natural" liquid behaviour

Another possible application of this technique is the modelling of a special "live" liquid covering the animated meshes (Fig. 11). In such an animation effect the convolution surface radii are increased over time, which creates the effect of the liquid flowing up the mesh and gradually engulfing it. It is possible to automatically generate this sort 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.

6.4. Results

In this project our main aim was to achieve near-real time frame rates while providing flexibility for the end user. Using the CPU, the average times required for the mesh generation of various models are shown in Table 1.

We have also used the NVIDIA CUDA SDK thus performing the computations on the GPU (see Table 2). This is a basic implementation, where only function values are computed on the GPU, while triangles are still generated on the

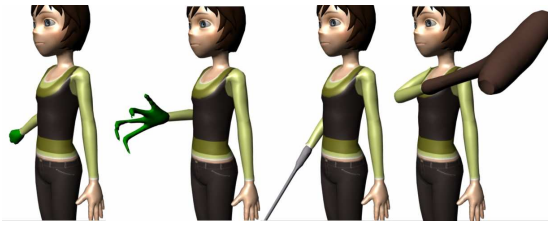


Figure 10: *Hybrid Andy (the hybrid model with a polygonal body and an attached "metamorphosing" implicit limb)*

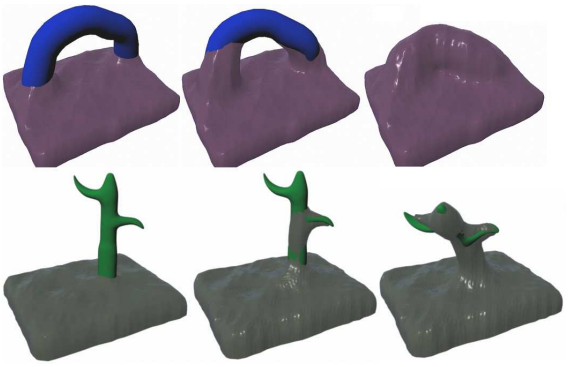


Figure 11: *Interactions of the meshes and the liquid object*

(a) (b)

Figure 12: *Metamorphosis (a) Two skeletons (shown in red and blue) specifying metamorphosis (b) Metamorphosis between convolution surfaces specified by two different skeletons*

CPU. Triangle generation is a relatively fast process compared to the function evaluation. Since it was required to communicate the generated data back to the tool, we needed to feed the generated triangles back to the RAM. For other applications, it is possible to perform the process of mesh generation and rendering completely on the GPU. This task can also be performed on GPUs without CUDA support. In the fashion of a GPGPU, the function computations across the grid can be performed in a pixel/fragment shader. This data could then be fed back to the RAM or used directly in a vertex shader [Pas04] or a geometry shader [TSD07] to create a mesh. Alternatively, ray-casting can be used to render the generated volume data directly [TSD07].

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 suited for a parallel real-time implementation. Given the current hardware design trends towards an ever increasing number of streaming processors within a GPU and cores within a CPU, it will be possible, in the near future, to increase both the resolution and complexity of a model while maintaining real-time rendering rates.

The integration of such functionality into an existing animation package decreases the learning curve for the user. The user is free to produce an animation sequence in a way

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 times for mesh generation (milliseconds/frame) on a PC with a Dual Core Intel Xeon (2.66 GHz), 2 GB of RAM; Andy is a mesh model with an embedded convolution surface (Fig. 5), Hybrid Andy is shown in figure 10)*

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 times for mesh generation (milliseconds/frame) on an NVIDIA GeForce 8800 Ultra, 768 MB of RAM*

that he is accustomed to within the familiar software environment, while having an opportunity to see the results of his actions 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.

6.5. Discussion

There are several issues that require additional consideration and will 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 the initial geometric objects being blended. The scalar fields produced by known convolution surface kernels significantly decrease as the distance from the line segment increases. 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 between the mesh and the viscous object at large distances. In such cases, an ellipsoidal approximation of the mesh could provide better results.

Additionally as the distance between the two blended objects increases, the deformation of the 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 frequently used to integrate these particles into the implicit model. Particle positions retrieved after the physical simulation could be used to add metaballs to the final model. This would allow for partial modelling of physically correct behaviour within the existing geometric model. A hybrid approach could help to overcome the significant growth of the number of particles needed to simulate some special effects [CBP05]. In the case of grid-based fluid simulation, the field produced by a convolution surface could be used for the creation of force fields interacting with fluids. It is also possible to employ temporal coherence. Results of the polygonization retrieved from the previous frame can be used to find the new grid cells where the function needs to be evaluated. The function could be evaluated only in the neighbourhood of the cells which contained the surface in the previous frame. In this case the velocity of the skeleton elements needs to be considered as well, in order not to miss surface changes due to fast motion of the skeleton. This could significantly improve the performance of the method, especially for lengthy animation sequences. 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 [CGD97]. We could also estimate the "amount of intersection" between the bounding volumes of the skeletal primitives and the viscous object. This estimation could 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.

Another limitation of our approach, related to the blending surface, is that of texturing, as texture matching for dynamic implicit surfaces is still an open research question.

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 [TZF04]. 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. 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 used this approach to model the adhesive behaviour of viscoelastic objects in their interaction with moving surfaces. The physical effect of the 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. We have implemented a prototype plug-in for a commercial animation system. Our plug-in provides the user 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

[AGCA06] ALLÈGRE R., GALIN E., CHAINE R., AKKOUCHE S.: The hybridtree: mixing skeletal implicit surfaces, triangle meshes, and point sets in a free-form modeling system. *Graph. Models 68*, 1 (2006), 42–64.

[AKK*02] ADZHIEV V., KARTASHEVA E., KUNII T., PASKO A., SCHMITT B.: Hybrid cellular-functional modelling of heterogeneous objects. In *Journal of Computing and Information Science in Engineering, Transactions of the ASME* (2002), vol. 4, pp. 312–322.

[BGA05] BARBIER A., GALIN E., AKKOUCHE S.: A framework for modeling, animating, and morphing textured implicit models. *Graph. Models 67*, 3 (2005), 166–188.

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3 (2007), 72–80.

[BS91] BLOOMENTHAL J., SHOEMAKE K.: Convolution surfaces. *SIGGRAPH Comput. Graph. 25*, 4 (1991), 251–256.

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 219–228.

[CG98] CANI-GASCUEL M.-P.: Layered deformable models with implicit surfaces. In *Graphics Interface* (1998), pp. 201–208.

[CGD97] CANI-GASCUEL M.-P., DESBRUN M.: Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics 3*, 1 (1997), 39–50.

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. 23*, 3 (2004), 377–384.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 23–30.

[GBO04] GOKTEKIN T. G., BARGTEIL A. W., O'BRIEN J. F.: A method for animating viscoelastic fluids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 463–468.

[JLW*05] JIN X., LIU S., WANG C. C. L., FENG J., SUN H.: Blob-based liquid morphing: Natural phenomena and special effects. *Comput. Animat. Virtual Worlds 16*, 3-4 (2005), 391–403.

[JTFP01] JIN X., TAI C.-L., FENG J., PENG Q.: Convolution surfaces for line skeletons with polynomial weight distributions. *J. Graph. Tools 6*, 3 (2001), 17–28.

[KYF05] KANZOW C., YAMASHITA N., FUKUSHIMA M.: Levenberg-marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints. *J. Comput. Appl. Math. 173*, 2 (2005), 321–343.

[LAG01] LECLERCQ A., AKKOUCHE S., GALIN E.: Mixing triangle meshes and implicit surfaces in character animation. In *Proceedings of the Eurographic workshop on Computer animation and simulation* (New York, NY, USA, 2001), Springer-Verlag New York, Inc., pp. 37–47.

[MS98] MCCORMACK J., SHERSTYUK A.: Creating and rendering convolution surfaces. *Comput. Graph. Forum 17*, 2 (1998), 113–120.

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 449–456.

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

[OM95] OPALACH A., MADDOCK S. C.: High level control of implicit surfaces for character animation. In *Proc. 1st International Eurographics Workshop on Implicit Surfaces* (1995), pp. 223–232.

[Pas04] PASCUCCI V.: Isosurface computation made simple: hardware acceleration, adaptive refinement and tetrahedral stripping. In *In Joint Eurographics - IEEE TVCG Symposium on Visualization (VisSym* (2004), pp. 293–300.

[PASS95] PASKO A., ADZHIEV V., SOURIN A., SAVCHENKO V.: Function representation in geometric modeling: Concepts, implementation and applications. *The Visual Computer*, 11 (1995), 429–446.

[PPK05] PASKO G., PASKO A., KUNII T.: Bounded blending for function-based shape modeling. *Computer Graphics and Applications, IEEE 25*, 2 (2005), 36–45.

[SP95] SINGH K., PARENT R.: Implicit function based deformations of polyhedral objects. In *Proc. 1st International Eurographics Workshop on Implicit Surfaces* (Apr. 1995), pp. 113–128.

[SPOK95] SAVCHENKO V., PASKO A., OKUNEV O., KUNII T.: Function representation of solids reconstructed from scattered surface points and contours. *Comput. Graph. Forum 14*, 4 (1995), 181–188.

[ST95] SHEN J., THALMANN D.: Interactive shape design using metaballs and splines. In *Implicit Surfaces'95* (1995), pp. 187–196.

[SY05] SHI L., YU Y.: Taming liquids for rapidly changing targets. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 229–236.

[TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), Eurographics Association, pp. 7–12.

[TSD07] TATARCHUK N., SHOPF J., DECORO C.: Real-time isosurface extraction using the gpu programmable geometry pipeline. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), ACM, pp. 122–137.

[TZF04] TAI C.-L., ZHANG H., FONG J. C.-K.: Prototype modeling from sketched silhouettes based on convolution surfaces. *Comput. Graph. Forum 23*, 1 (2004), 71–84.

[WLK03] WEI X., LI W., KAUFMAN A.: Melting and flowing of viscous volumes. In *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)* (2003), IEEE Computer Society, pp. 54–59.