The HybridTree: Mixing Skeletal Implicit Surfaces, Triangle Meshes and Point Sets in a Free-form Modeling System

Rémi Allègre *, Eric Galin, Raphaëlle Chaine, Samir Akkouche

LIRIS CNRS, Université Claude Bernard Lyon 1, France

Abstract

In this paper, we present a hybrid modeling framework for creating complex 3D objects incrementally. Our system relies on an extended CSG tree that assembles skeletal implicit primitives, triangle meshes and point set models in a coherent fashion: we call this structure the HybridTree. Editing operations are performed by exploiting the complementary abilities of implicit and polygonal mesh surface representations in a complete transparent way for the user. Implicit surfaces are powerful for combining shapes with Boolean and blending operations, while triangle meshes are well-suited for local deformations such as FFD and fast visualization. Our system can handle point sampled geometry through a mesh surface reconstruction algorithm. The HybridTree may be evaluated through four kinds of queries, depending on the implicit or explicit formulation is required: field function and gradient at a given point in space, point membership classification, and polygonization. Every kind of query is achieved automatically in a specific and optimized fashion for every node of the HybridTree.

Key words: shape modeling, implicit surfaces, triangle meshes, point sets, blending, free-form deformations

* Corresponding author.
Email addresses: remi.allegre@liris.cnrs.fr (Rémi Allègre), eric.galin@liris.cnrs.fr (Eric Galin), raphaelle.chaine@liris.cnrs.fr (Raphaëlle Chaine), samir.akkouche@liris.cnrs.fr (Samir Akkouche).
URL: http://liris.cnrs.fr/remi.allegre (Rémi Allègre).

1 Introduction

For the purpose of modeling complex free-form shapes, a large number of geometric representation have been developed, each with specific properties and limitations. For certain modeling operations, some surface representations are thus more advantageous than others. Our goal is to overcome this kind of restriction by mixing multiple shape representations into a single coherent modeling framework that takes benefit from the complementary advantages of the different models. We focus here on three fundamental representations: implicit surfaces, triangle meshes and point sets.

Implicit surfaces [1,2] are powerful for representing objects of complex geometry and topology. They naturally lend themselves for blending [3] and CSG Boolean operations, and can be deformed by space warping techniques [4]. Pasko et al. [5] and later Wyvill et al. [6] have proposed two hierarchical models that incorporate Boolean, blending and warping operations in a unified system. We have contributed to develop the *BlobTree* model [6] that is characterized by a combination of skeletal primitives, rather than R-Functions, in a tree data-structure. The BlobTree has proven to be an intuitive and effective tool for modeling and animating complex and realistic organic shapes [7]. However, in this framework as in most implicit modeling frameworks, local deformations are difficult to implement and are restrictive. Visualizing complex implicit surfaces is also a computationally expensive task.

In contrast, triangle meshes can be efficiently visualized thanks to common graphic hardware. These surfaces can be edited interactively by a variety of powerful tools, such as free-form deformations [8] or Laplacian editing [9], that provide very intuitive local control over geometry. However, combining parametrically defined surfaces with Boolean operations is a complicated task, which is prone to topological inconsistencies. Polygonal meshes also do not naturally blend themselves together.

Point sampled geometry, that can be obtained from scanning devices, can be efficiently visualized and edited through point-based implicit surface models, such as Moving Least Squares [10]. This kind of representation is strongly dependent on the sampling density of the input point set and extrapolating reliable topological information can be a hard task. A connectivity structure can be provided by a surface reconstruction process [11].

In this paper, we describe a hybrid shape representation mixing implicit and polygonal mesh representations for incremental modeling of complex 3D shapes. This paper extends and improves the framework presented in [12] in several ways. Our model is characterized by a tree data-structure that combines skeletal implicit surfaces, triangle meshes and point set models by means

of Boolean, blending and warping operations, including free-form deformations. We call our model the *HybridTree*, which may be seen as a generalization of the BlobTree [6]. We evaluate this structure on-the-fly through three fundamental queries: field function, gradient and point membership classification, and a polygonization process. The originality of our model relies in the evaluation system that dynamically switches from one surface model to the other so as to use the most suitable representation for every type of editing operation. The core of our current implementation is a dual skeletal implicit/triangle mesh representation for every node. Each kind of node in the HybridTree is evaluated automatically in a specific and optimized fashion, depending on the formulation required by each operation. To handle point set models, we rely on an intermediate mesh representation that is obtained through the reconstruction technique proposed in [13].

In our system, the user can perform Boolean, blending and warping on either skeletal implicit, triangle mesh or point set data without worrying about the nature of the objects that are manipulated. Point sets are a new feature of our model that allows us to directly deal with digitally acquired objects. This paper develops improved methods for evaluating our model more efficiently, e.g. through point membership classification queries. We also present a new local polygonization technique for blending nodes that allows to reduce computation time significantly for objects that blend together. All the algorithms are described in detail with performance analysis.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of related shape modeling frameworks. Section 3 describes the architecture of our system and present how implicit and mesh representations are combined together. We explain in Section 4 how fundamental queries are performed on the HybridTree, and detail our polygonization algorithms in Section 5. Applications to complex shape modeling are discussed in Section 6. Eventually, in Section 7, we conclude and present future work.

2 Related work

Modeling complex 3D shapes, either from scratch and/or from existing surfaces, e.g. acquired with scanning devices, is an active research area in Geometric Modeling and Computer Graphics. Conversion techniques from one model to the other make it possible for objects in different representations to coexist and interact in the same environment through a unified representation. Recent developments in implicit, mesh and point set modeling have lead to new interesting solutions to tighten the gap between these three kinds of representations, from the surface editing point of view. The strategy that consists in combining the abilities of implicit and mesh models into a single hybrid

model has received attention only recently for efficient geometry processing and shape modeling.

2.1 Conversion techniques

Techniques to translate geometry from the implicit to the polygonal mesh representation or from the polygonal mesh to the implicit representation have been extensively studied, but still remain a challenging research field. In the Computer Graphics community, state-of-the-art implicit surface meshing techniques include 3D-space cell decompositions [14,15], particle systems [16], and surface marching methods [17,18]. Recent work in Computational Geometry focused on how to produce a mesh approximation of an implicit surface with guaranteed topology and geometry [19,20,21]. A polygonal mesh surface can be converted into an implicit surface either through surface reconstruction techniques from point sets [22,23,24] or from triangle meshes [25,26].

2.2 Implicit surface editing

In recent work, discrete implicit representations have been proposed as a general-purpose model for editing complex shapes. The level set framework proposed by Museth et al. [27] is based on a distance field sampled at the vertices of a voxel grid. The system provides conversion algorithms from many other representations, including point sets and polygonal meshes, and a wide range of editing tools. This model is memory consuming and the quality of the result is dependent on the resolution of the grid. The Adaptively Sampled Distance Fields introduced by Frisken et al. [28] rely on a hierarchical structure that provides local control over geometric error. In constrast, the framework in this paper manipulates a continuous implicit representation and preserves existing surfaces when possible. As other implicit surface models, these representations also do not accommodate large deformations.

A hybrid system based on the the F-Rep model by Pasko et al. [5] that mixes volumetric and function implicit representations has been studied in [29]. The proposed framework relies on a conversion to voxel representation so as to make both representations to interact in a unified fashion.

Schmitt et al. [30] have proposed a framework for local deformation of functionally-defined implicit surfaces. They rely on specific skeletal elements and field functions to simulate free-form deformations. This approach is not as intuitive as mesh deformation tools and does not offer as many degrees of freedom.

2.3 Mesh surface editing

Surface mesh editing has been recently enriched with new techniques based on differential coordinates. The Laplacian representation encodes the location of each vertex relatively to its neighborhood, which provides an approximation of the Laplacian of the underlying surface. Sorkine et al. [9] developed local deformation and blending tools that preserve geometric details. All operations require to solve least-squares systems, which can be achieved at interactive rates for not too dense meshes. A similar approach based on the discrete Poisson equation was introduced by Yu et al. [31]. Their technique is formulated by manipulation of the gradients of the mesh. The mesh surface is retrieved by solving the least-squares system resulting from discretizing the Poisson equation with Dirichlet boundary conditions. In both methods, mesh blending requires to solve a vertex matching problem between corresponding mesh boundaries. The meshes that are blended together should have near equal edge length, which may require an initial remeshing process.

Kanai et al. [32] have proposed a method for cutting and pasting mesh parts. After selection of mesh parts of interest, the correspondence between boundary vertices is first established. A registration process is applied between the two boundaries and a B-spline function is used to interpolate vertex locations smoothly. Using this technique, Funkhouser et al. [33] developed a new kind of "data-driven" mesh modeling framework. Starting from a mesh model, the user can select parts to edit thanks to an interactive segmentation algorithm, and then query a mesh database for similar parts. The desired parts can be extracted from the retrieved models and then merged smoothly with the base mesh.

Polygonal mesh blending is also closely related to shape interpolation and morphing techniques [34]. A source polygonal mesh and a target one can be locally interpolated so as to achieve local blending effects. Related work in this domain include the work by Alexa [35], in which Laplacian coordinates are linarly interpolated. Xu et al. [36] proceed by non-linear interpolation of gradient fields by solving Poisson equations defined on meshes. This approach involves numerous parameterization and remeshing issues [37,38]. The very limitation of mesh blending based on morphing methods is that the source and target models should have the same topology.

2.3.1 Point-based modeling

Due to the recent advances of 3D digital acquisition, shape modeling from point-sampled geometry has also attracted particular attention for a few years. The Moving Least Squares implicit surface model introduced by Levin [39] has

proven abilities for both interactive surface editing [10] and physical simulation [40]. Its major interest is that it can handle digitally acquired surfaces without preliminary surface reconstruction step. Explicit information about topology is not maintained, which has advantages for some operations. Whenever neighborhood information is needed, connectivity relations based on k-neighborhoods are computed on-the-fly using a spatial search data-structure. However, establishing correct connectivity relations is not always a trivial problem, that depends on the sampling distribution of the points on the input surfaces [41,42]. Moreover, the sampling density has to be updated frequently to maintain a coherent surface, which requires surface reconstruction steps. While the original Moving Least Squares fitting technique used to be capable of representing smooth surfaces only, an extended model proposed by Fleishman et al. [43] can now handle sharp features robustly, but at the expense of decreased performance.

The variational technique presented in [44] generates an implicit surface from point-sets via an interpolation scheme based on compactly supported radial basis functions. The resulting shapes can be locally controlled in an intuitive way by acting on the constraint points. The evaluation may be performed at interactive rates using an octree data-structure. However, this approach remains computationally demanding when manipulating dense point sampled geometry, and sharp features are difficult to handle in this framework.

2.3.2 Hybrid modeling techniques

Depending on the surface representation, some operations cannot be performed easily in a direct way. In some cases, this issue can be addressed with the help of an intermediate representation. Over the past few years, hybrid models have been investigated by several authors for this purpose. In the field of geometry processing, some specific problems may be efficiently solved using a hybrid modeling approach [45]. For shape modeling, this approach has attracted attention for mesh deformation and blending. Several implicit models have been used to deform meshes [46,47,48]. Decaudin [49] and Singh and Parent [50] introduced mesh blending techniques based on an intermediate implicit representation that requires the input meshes to be star-shaped or locally star-shaped. In the first method, the resulting shape is completely retriangulated, while the second one only retriangulates blending regions using the Marching Cubes algorithm [14].

3 The HybridTree

The HybridTree model relies on a tree data-structure whose leaves can hold either complex skeletal implicit primitives as described in [51], triangle meshes with manifold topology or point set models. Those models are combined by Boolean, blending and warping operations located at the nodes of the tree. Warping nodes include affine transformations, Barr deformations [52] and free-form deformations [8]. Constructive operations are binary whereas warping operations are unary operations.

Figure 1 shows the HybridTree structure of a winged snake-woman model. The snake-woman model (a) has been entirely built from skeletal implicit primitives. Using Boolean difference, only the body has been conserved, which has been then blended with the Igea point set model (b) so as to obtain the result in (c). The wings of a mesh model of the Victory of Samothrace (d) have been extracted by intersecting the model with a box. The wings and the modified snake-woman model have been finally blended together in (e).

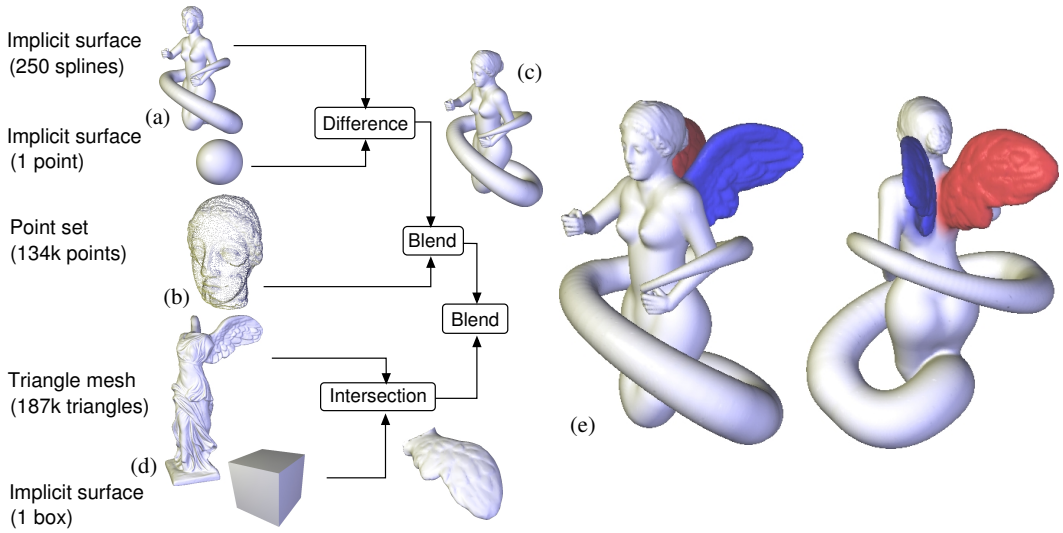


Fig. 1. The HybridTree structure of a winged snake-woman model.

The evaluation of the HybridTree is achieved in an incremental way by recursively traversing the tree data-structure. The architecture of our evaluation system is presented in Figure 2. Each pole corresponds to a geometric representation that provides the set of operations for which it is the most well-suited. Arrows depict the gateways from one model to the other, that correspond to conversion procedures. Starting from an implicit, mesh or point set object, it is first converted on-the-fly into the required representation before applying a given operation. The gateways available in our current implementation are depicted by solid arrows in the diagram. Skeletal implicit surfaces and triangle meshes are currently the core of our system. Every node of the HybridTree can generate both a potential field in space and a triangle mesh. Point sets mo-

dels are plugged through a surface reconstruction technique that produces a triangle mesh representation. The completeness of the system is thus achieved by transitivity. Conversion from the point set representation to the implicit one could be performed directly using Moving Least Squares or variational techniques. However, these techniques are not compatible with our skeletal implicit model, that requires the signed distance function to the surface to be reliably evaluable everywhere in space. Depending on the sampling density, converting a triangle mesh into a point set model may require a resampling process, as proposed in [19], that is not currently included in our system. Conversion from a skeletal implicit surface into a point set is the object of current work.

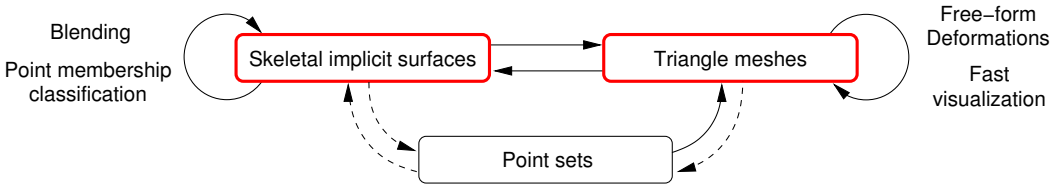


Fig. 2. The HybridTree's evaluation system.

We convert point set models into triangle meshes using the convection-driven surface reconstruction technique developed by Allègre et al. in [13], that offers user control over the level of detail of the resulting mesh. An application of this technique is illustrated in Figure 3. The input point set on the left has 134,344 points, uniformly distributed on the surface. The center and right images show the reconstruction results. Given a prescribed level of detail, the points set has been automatically simplified while reconstructing the surface so that the resulting mesh only incorporates 31,126 vertices (76% decimation) and 62,323 triangles. The result was obtained in 64 second on a Pentium IV 3.0GHz - 1GB RAM workstation.

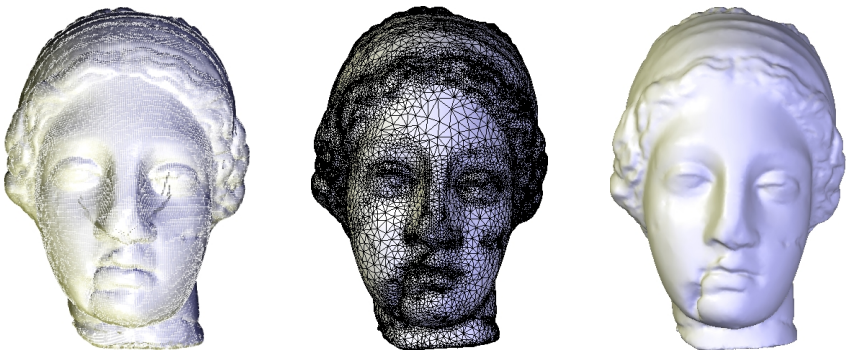


Fig. 3. Illustration of the surface reconstruction technique on the Igea model. Left: original point set (134,344 points); Center and right: reconstructed mesh (31,126 vertices, 62,323 triangles).

The HybridTree is evaluated through three fundamental queries and a polygonization process that are implemented in a specific fashion for each kind of

node. Field function queries at a given point in space are performed whenever the implicit formulation is required. We essentially rely on the implicit formulation to achieve blending operations. Gradient queries allow to obtain the exact normal at sample points. Some operations only require to know whether a point lies inside or outside the surface, such as Boolean operations. Instead of evaluating the sign of the field function explicitely, we developed specific methods to perform accelerated point membership queries. The last query type in our framework is an incremental polygonization process that is invoked at a given node if the mesh formulation is needed, for local deformations or visualization. Local results are combined into a coherent fashion by binary operations.

Notations

An implicit surface is mathematically defined by a field function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ as the points in space that satisfy the equation:

$$S = \{\mathbf{p} \in \mathbb{R}^3 | f(\mathbf{p}) = T\}$$

where T is a threshold level.

The field function of a node A will be denoted as f_A, and the corresponding gradient as ∇f_A. We will call $c_A(\mathbf{p})$ the point membership function of A at point $\mathbf{p}$, that can take three different values $\{1, 0, -1\}$ depending on $\mathbf{p}$ respectively lies inside, on, or outside the surface of A. The notation $\mathcal{M}_A$ will refer to the mesh of the surface of A, and the bounding box of the object A will be denoted as $\mathcal{B}_A$.

4 Fundamental queries

In the following paragraphs, we detail how the field function, gradient and point membership are evaluated for the different kinds of node in the Hybrid-Tree.

4.1 Skeletal implicit primitives

Skeletal implicit primitives are built around a geometric object called *skeleton*. The field function for a given skeleton is evaluated analytically using the

following formulation:

$$f(\mathbf{p}) = g \circ d(\mathbf{p})$$

where $d : \mathbb{R}^3 \rightarrow \mathbb{R}_+$ denotes the Euclidean distance to the skeleton, and $g : \mathbb{R}_+ \rightarrow \mathbb{R}$ refers to the potential field function. The latter is a compactly supported radial basis function that is parameterized by a maximum field value $I \in \mathbb{R}$ reached on the skeleton, and a radius of influence that will be denoted as $R \in \mathbb{R}_+$. The associated region of influence, characterized by non-zero field values, will be denoted as Ω. In our system, we use polynomial potential field functions of the form:

$$g(r) = \begin{cases} I\left(1 - \dfrac{r^2}{R^2}\right)^n, & n \geq 2 \quad \text{if } r \in [0, R] \\ 0 \quad \text{otherwise} \end{cases}$$

The corresponding inverse potential field functions $g^{-1} : \mathbb{R} \rightarrow \mathbb{R}_+$ is defined as follows:

$$g^{-1}(t) = \begin{cases} R\sqrt{1 - \left(\dfrac{t}{I}\right)^{\frac{1}{n}}}, & n \geq 2 \quad \text{if } 0 < t \leq I \text{ or } I \leq t < 0 \\ 0 \quad \text{otherwise} \end{cases}$$

Normals can be obtained directly from the gradient of the field function $\nabla f(\mathbf{p})$ which is evaluated as follows:

$$\nabla f(\mathbf{p}) = g' \circ d(\mathbf{p})\, \nabla d(\mathbf{p})$$

Since $d(\mathbf{p})$ is the Euclidean distance function, $\nabla d(\mathbf{p})$ is computed as the vector between the orthogonal projection of $\mathbf{p}$ onto the skeleton and $\mathbf{p}$.

The HybridTree implements a wide range of complex skeletal primitives including curve, surface and volume skeletons as described by Barbier et al. in [51]. Figure 4 shows a bottle built using surface of revolution, spline, circle and hollow cylinder skeletons.

Every type of primitive implements a specific algorithm that computes the distance $d(\mathbf{p})$ to its skeleton analytically in an optimized fashion. Although the computation of the distance between a point in space and a simple skeleton such as a point or a line segment is straightforward and computationally efficient, algorithms become more sophisticated as the complexity of the skeletons increases. Some detailed algorithms can be found in Schneider and Eberly [53] and Barbier and Galin [54].

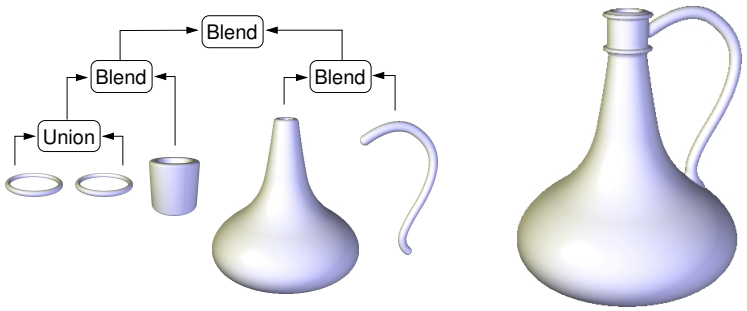


Fig. 4. A bottle model built using complex skeletal implicit primitives.

For implicit surfaces, point membership classification is usually obtained through an evaluation of the field function and comparing its value to the threshold level. Since we manipulate skeletal implicit primitives, we do not need to compute the full field function for point membership queries. The T level surface of these primitives may indeed be defined by sweeping a sphere of constant radius $r_T = g^{-1}(T)$ along the boundary of the skeleton. This iso-surface exists if and only if $r_T \geq 0$. Therefore, for a given point $\mathbf{p}$ in space, the point membership function $c(\mathbf{p})$ may be defined as follows:

$$c(\mathbf{p}) = \begin{cases} -1 \text{ if } 0 \leq r_T < d(\mathbf{p}) \\ \quad 1 \text{ if } 0 < d(\mathbf{p}) < r_T \\ \quad 0 \text{ otherwise} \end{cases}$$

The radius r_T is computed once for each primitive. For volume skeletal primitives, the location of query points with respect to the interior of the skeleton is obtained analytically as part of the computation of the distance $d(\mathbf{p})$.

Skeletal implicit primitives are defined only by a few parameters, which yields particularly low storage cost. Complex skeletons are also easier to control than a combination of simple primitives, but distance evaluations are more computationally demanding. A basic acceleration technique consists in pre-computing and caching some results required for several evaluations, e.g. along a ray [51]. When the evaluation of $f(\mathbf{p})$ and $\nabla f(\mathbf{p})$ are both required, common intermediate results are also computed only once. At global scale, every primitive is equiped with a bounding box that allows to save useless evaluations.

4.2 Polygonal meshes

For a polygonal mesh, the field function is computed using the same formulation as for skeletal implicit primitives. The distance function $d_{\mathcal{M}}$ from a point $\mathbf{p} \in \mathbb{R}^3$ to a triangle mesh $\mathcal{M}$ is defined as the minimal Euclidean distance

between $\mathbf{p}$ and any triangle $\mathcal{T}$ of the boundary of $\mathcal{M}$:

$$d_{\mathcal{M}}(\mathbf{p}) = \min_{\mathcal{T} \in \mathcal{M}} d(\mathbf{p}, \mathcal{T})$$

The implicit surface generated by the skeletal mesh for a given threshold T is a rounded surface S which differs for the original mesh $\mathcal{M}$. This surface may be defined by sweeping a sphere of constant radius $r_T = g^{-1}(T)$ along the boundary of $\mathcal{M}$ (Figure 5, left). To make the boundary of $\mathcal{M}$ and the T level surface to correspond independently from the field function parameters, we incorporate the threshold as an offset in a pseudo-distance function which is defined as follows:

$$d(\mathbf{p}) = \begin{cases} d_{\mathcal{M}}(\mathbf{p}) + r_T & \text{if } \mathbf{p} \text{ is outside } \mathcal{M} \\ r_T - d_{\mathcal{M}}(\mathbf{p}) & \text{if } \mathbf{p} \text{ is inside } \mathcal{M} \text{ and } d_{\mathcal{M}}(\mathbf{p}) < r_T \\ 0 & \text{otherwise} \end{cases}$$

Our distance function guarantees that the isosurface and the mesh boundary mesh are the same for any value of T, as shown in Figures 5(right) and 6(right).

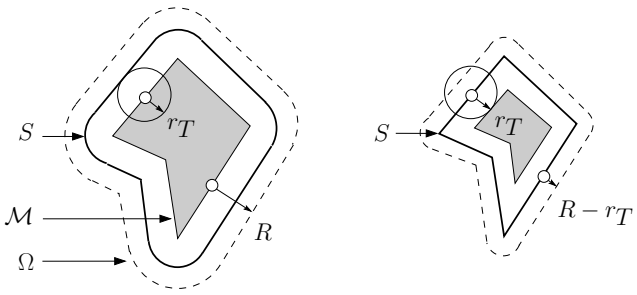


Fig. 5. The distance offset mechanism on a 2D polygon. On the left, the basic distance formula is used. The effect of our distance function is illustrated on the right.

The user keeps control on every parameter of the field function, and can precisely control the range of the blend between two objects. The radius of influence of the mesh, which falls from R to $R - r_T$, is rescaled so that the distance offset is hidden. If the user specifies a radius of influence R, the field function is evaluated with a radius of influence R' such that $R' - r'_T = R$, where $r'_T = g^{-1}(T)$ depends on R' and I. For our potential field functions, we have:

$$\frac{r'_T}{R'} = \frac{r_T}{R} = \sqrt{1 - \left(\frac{T}{I}\right)^{\frac{1}{n}}}$$

that only depends on fixed parameters. We then compute R' as follows:

$$R' = \frac{R}{\left(1 - \frac{r_T}{R}\right)}$$

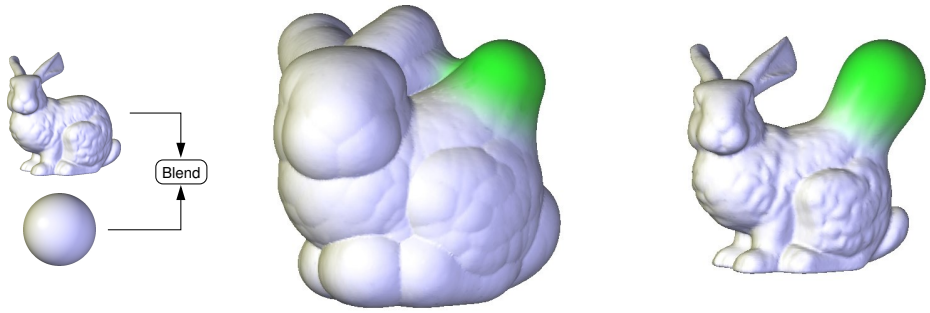


Fig. 6. The Stanford Bunny mesh model (69,674 triangles) blended with an implicit sphere. The result on the left uses the distance $d_{\mathcal{M}}(\mathbf{p})$ whereas the model on the right has been computed using our pseudo-distance function $d(\mathbf{p})$.

The gradient of the field function $\nabla d(\mathbf{p})$ is evaluated as follows:

$$\nabla d(\mathbf{p}) = \begin{cases} \nabla d_{\mathcal{M}}(\mathbf{p}) & \text{if } \mathbf{p} \text{ is outside } \mathcal{M} \\ -\nabla d_{\mathcal{M}}(\mathbf{p}) & \text{if } \mathbf{p} \text{ is inside } \mathcal{M} \text{ and } d_{\mathcal{M}}(\mathbf{p}) < r_T \\ 0 & \text{otherwise} \end{cases}$$

As for implicit primitives, $\nabla d(\mathbf{p})$ is computed as the vector between the orthogonal projection of $\mathbf{p}$ onto the mesh $\mathcal{M}$ and $\mathbf{p}$, that can be obtained as part of the computation of $d(\mathbf{p})$.

Computing the minimum distance between a point $\mathbf{p}$ and all the triangles $\mathcal{T}$ of the mesh $\mathcal{M}$ is computationally expensive. Acceleration techniques have been widely studied, not only in the Computer Graphics community [55]. Our framework implements an algorithm inspired by Johnson and Cohen's lower-upper bound strategy [56]. We rely on a bounding box hierarchy based on a Binary Space Partition tree, which is traversed breadth-first for each point-to-mesh distance query. For each node we compute a lower and an upper bound of the minimum point-to-mesh distance, which yields efficient space-pruning. Moreover, we use the fact that the potential field falls to zero beyond the distance R from the mesh boundary so as to reject more useless point-to-mesh distance computations.

As illustrated in Figure 7, the effectiveness of our optimization varies according to the geometry of $\mathcal{M}$ and the size of R. We measured the time taken to perform 100^3 point-to-mesh distance queries for different meshes, with an

increasing radius of influence. For each mesh, vertex coordinates were normalized to fit in a unit bounding box. Query points correspond to a regular 3D sampling of this box. Solid line timings take the radius of influence optimization into account, while dashed line timings do not. It can be observed that a lot of time is saved when R is small regarding the overall size of the mesh. Only a very reduced set of bounding boxes of the hierarchy have to be tested in this case. The benefit of our acceleration then decreases as R increases, and the running times tend to stabilize.

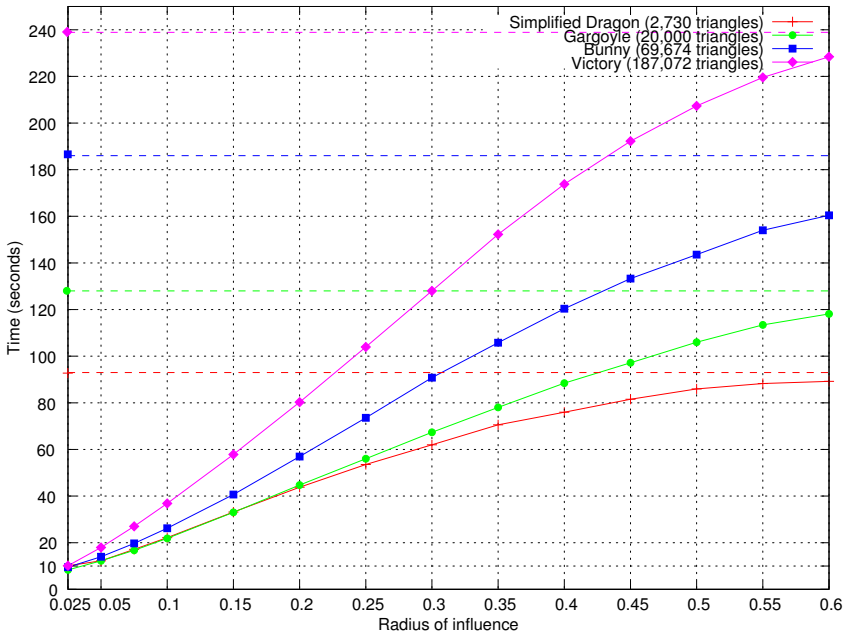


Fig. 7. Timings for 100^3 point-to-mesh distance queries (computations performed on a Pentium IV 3.0GHz - 1GB RAM workstation).

The given timings do not take into account the time taken to build the bounding box hierarchy. Preprocessing times are given in Table 1, that also indicates the depth of the bounding box hierarchy for all the models we tested. The leaf boxes contain at most 4 triangles so as to avoid too much bounding box tests.

Model	#Triangles	Depth	Pre. Time
Simplified Dragon (a)	2,730	23	0.05
Gargoyle (b)	20,000	22	0.39
Bunny (c)	69,674	28	1.31
Victory (d)	187,072	25	4.82

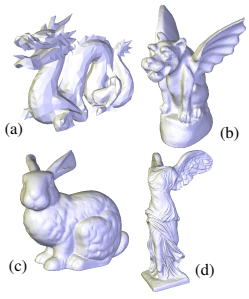


Table 1
Number of triangles, depth of the bounding box hierarchy and preprocessing timings for four different mesh models.

The cost of point-to-mesh distance computations could be even more reduced using a triangle fan decomposition of the mesh [57], that allows to save some redundant operations by factorizing computations between neighboring triangles in each triangle fan.

Point membership classification is obtained by computing the number of intersections between a ray and the mesh using the bounding box hierarchy. In order to reduce the number of cells to be traversed, the direction and the orientation of the ray are chosen so that the distance between the query point and the intersection point between the ray and the mesh bounding box is minimized.

4.3 Blending operations

In our system, we propose two kinds of formulations for blending two surfaces : one global and one local. Let A and B denote two models that blend together. Global blending between two objects is functionally defined as originally proposed by Blinn [3]:

$$f_{A+B} = f_A + f_B$$

Figure 8 shows two mesh models blended together using our global blending operation. The major drawback of this basic approach is the well-known unwanted blending problem [58]. As shown on the right image in Figure 8, the right wing is undesirably blended with the Bunny's right ear.

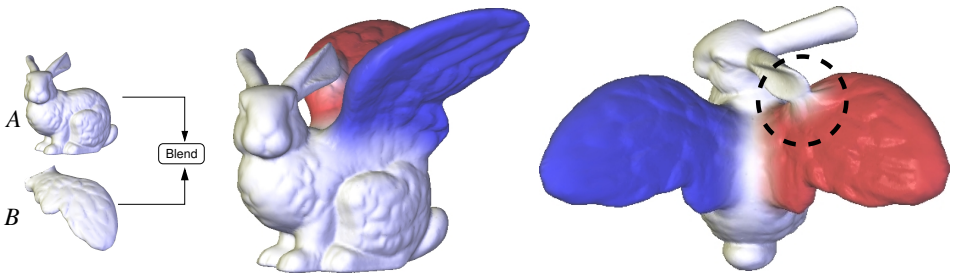


Fig. 8. Global blending between two mesh models: the Stanford Bunny (69,674 triangles) and a wing pair (8,170 triangles).

To perform blending with better local control, we have implemented a new local blending operation adapting the local blending technique described by Pasko et al. in [59]. This operation has three children. The first two, denoted as A and B, represent the two models that will be partially blended together, whereas the third, denoted as C, represents the region of space where blending will occur.

In our system, C is characterized by a potential field that is defined as a union of implicit primitives denoted as C_i. Such a combination makes it possible to build complex blending regions with predictible results. The field function f_C characterizes the blending region and is used to scale the amount of blending between the two sub-trees A and B. The values taken by the field functions f_{C_i} should range between 0 and 1. At a given point in space $\mathbf{p}$, if $f_C(\mathbf{p}) = 0$ then only union occurs, which is the case for any point outside the region of influence of C. In contrast, if $f_C(\mathbf{p}) = 1$, full blending takes place normally.

The evaluation of the local blending operation is performed as follows. We first compute the potential field value resulting from the blending of the children nodes $f_{A+B}(\mathbf{p}) = f_A(\mathbf{p}) + f_B(\mathbf{p})$, and the field function value $f_{A \cup B}(\mathbf{p})$. We define the resulting field function as a weighted average:

$$f_{A+B}(\mathbf{p}) = f_C(\mathbf{p})\, f_{A+B}(\mathbf{p}) + (1 - f_C(\mathbf{p}))\; f_{A \cup B}(\mathbf{p})$$

Primitives build from a volume skeleton are very useful to define regions in space where full blending occurs. In Figure 9, the local blending region C is defined as the union of two implicit cylinders.

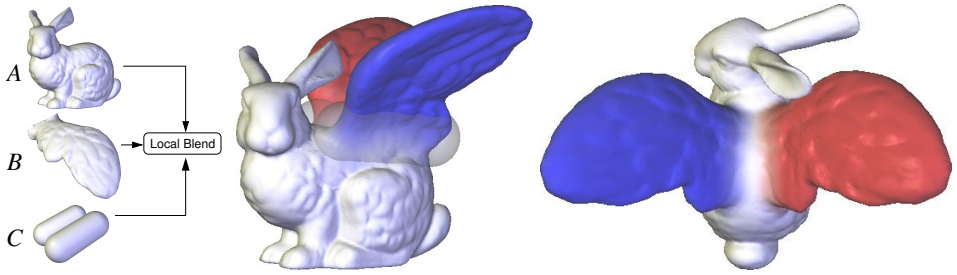


Fig. 9. Local blending. Two implicit cylinders define the blending region between the Bunny and the wing pair.

For both blending techniques, the gradient is obtained by deriving the field functions. For point membership classification, we distinguish *positive* and *negative* potential fields. A model A will be said to generate a positive potential field if one of its primitive is such that $I > 0$. Conversely, a model A will be said to generate a negative potential field if and only if every primitive it consits of is such that $I < 0$.

Let A and B two models that globally blend together such that A and B both generate a positive potential field. Point membership classification is then achieved as follows:

(1) If $\mathbf{p}$ is located inside $\mathcal{B}_A \cap \mathcal{B}_B$, then evaluate $v = f_{A+B}(\mathbf{p})$.
   (a) If $v < T$, $\mathbf{p}$ lies inside the surface: return 1.
   (b) Else, if $v > T$, $\mathbf{p}$ lies outside the surface: return -1.
   (c) Else, $\mathbf{p}$ is on the surface: return 0.

(2) Else, if $\mathbf{p}$ belongs to $\mathcal{B}_A \setminus \mathcal{B}_B$ (resp. $\mathcal{B}_B \setminus \mathcal{B}_A$), then query point membership on A (resp. B) and return the result.
(3) Else, $\mathbf{p}$ lies outside the surface: return -1.

For local blending nodes such that A, B and C generate positive potential fields, the previous algorithm is slightly modified. In Step 1, we consider the bounding box $\mathcal{B}_C$ of the local blending region and evaluate f_{A+B}. In Step 2, we consider the boxes $\mathcal{B}_A \setminus \mathcal{B}_C$ and $\mathcal{B}_A \setminus \mathcal{B}_C$.

For global blending nodes such that A generates a positive potential field and B generates a negative potential field, the previous algorithm is modified as follows: In Step 2, if $\mathbf{p}$ belongs to $\mathcal{B}_B \setminus \mathcal{B}_A$, then $\mathbf{p}$ is outside the surface. If $\mathbf{p}$ belongs to $\mathcal{B}_A \setminus \mathcal{B}_B$, then point membership is queried on A. If both A and B generate negative potential fields, $\mathbf{p}$ lies outside the surface: -1 is returned.

Let us finally consider a local blending nodes such that C generates a negative potential field. If $\mathbf{p}$ is located inside $\mathcal{B}_C$, then $\mathbf{p}$ lies outside the surface: 1 is returned. Otherwise, if $\mathbf{p}$ belongs to $\mathcal{B}_A \setminus \mathcal{B}_C$ (resp. $\mathcal{B}_B \setminus \mathcal{B}_C$), then point membership is queried on A (resp. B) and the result is returned.

4.4 Boolean operations

The min and max functions prescribed in [6] for union and intersection produce gradient discontinuities in the potential function. This results in visible unwanted normal discontinuities on the surface.

Contrary to min and max functions, R-Functions define a field function with C^n continuity almost everywhere in space, except on the surface, which avoids unwanted discontinuities. The functions prescribed in [5] operate on field functions that have an infinite support, whereas our model operates on field functions that have a compact support. Moreover, R-Functions have been designed for implicit surfaces characterized by a null threshold value: $T = 0$.

We have adapted those functions to our model by incorporating the threshold value as an offset in the previous equations. A weighting coefficient appears so as to guarantee that the resulting field function still has a compact support. We have:

$$f_{A\cup B} = T + \frac{1}{2-\sqrt{2}}\left[(f_A - T) + (f_B - T) + \sqrt{(f_A - T)^2 + (f_B - T)^2}\right]$$

$$f_{A\cap B} = T + \frac{1}{2+\sqrt{2}}\left[(f_A - T) + (f_B - T) - \sqrt{(f_A - T)^2 + (f_B - T)^2}\right]$$

Although min and max functions on the one hand, and R-Functions on the other hand produce different potential fields in space, both representations produce the same implicit surface if the Boolean nodes are located at the top of the tree structure. In this case, the computation of the min and max is computationally inexpensive compared to R-Functions. In contrast, we use the modified R-Function equations to create a continuously differentiable potential field if blending nodes are located above Boolean operations in the HybridTree. Our system automatically adapts the function used to evaluate Boolean operations depending on the context during the evaluation. Figure 10 illustrates the two union operations.

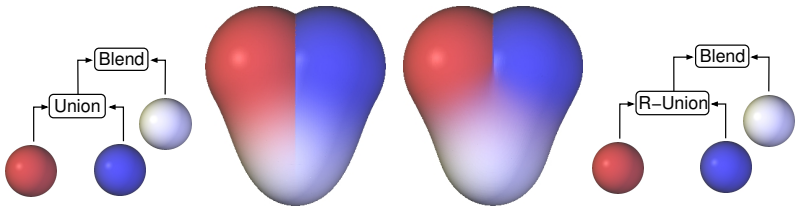


Fig. 10. With the Union operation (left), the discontinuity of the field function produced by the max function is propagated to the blending region. This problem is fixed with the R-Union continuous function (right).

For Boolean operations defined using the min and max functions, the gradient at a given point in space is the gradient of the minimum or maximum contributing field function between the two input models. Using the R-Function formulation, the gradient is simply derived from the field functions.

Point membership classification is obtained through the standard operations of Boolean algebra. For instance, the point membership classification for a difference operation between two models A and B is performed as follows:

(1) If $c_A(\mathbf{p}) = 1$, then:
  (a) If $c_B(\mathbf{p}) = -1$, then return 1.
  (b) Else, if $c_B(\mathbf{p}) = 0$, then return 0.
  (c) Else, return -1.
(2) Else, if $c_A(\mathbf{p}) = 0$ and $c_B(\mathbf{p}) \in \{0, 1\}$ then return 0.
(3) Else, return -1.

4.5 Warping operations

In our system, the shape of a surface can be distorted by locally warping space. Our warping operations include affine transformations and Barr's twist, taper and bend deformations [52]. We also handle free-form deformations [8], denoted as FFD, so as to perform local deformations. Throughout the following paragraphs, A will denote the child object of a warping node, w a

space transformation that maps $\mathbb{R}^3$ into $\mathbb{R}^3$, and w^{-1} the corresponding inverse transformation.

4.5.1 Barr deformations

When the implicit formulation is required, twist, taper and bend deformations are applied as warp functions. The resulting field function is defined using the inverse warp function as follows:

$$f_w(\mathbf{p}) = f_A \circ w^{-1}(\mathbf{p})$$

The gradient of the field function may be evaluated as:

$$\nabla f_w(\mathbf{p}) = J^{\mathsf{T}}_{w^{-1}}(\mathbf{p}) \times \nabla f_A \circ w^{-1}(\mathbf{p})$$

where $J^{\mathsf{T}}_{w^{-1}}(\mathbf{p})$ denotes the transpose Jacobian matrix of the inverse warp function w^{-1} at $\mathbf{p}$. For these deformations, the closed form expressions of w^{-1} and $J^{\mathsf{T}}_{w^{-1}}(\mathbf{p})$ can be easily computed. The detailed equations can be found in [52] (it should be noted that there are some mistakes in the formulas related to the inverse bend transformation).

Point membership classification is achieved by computing the location $w^{-1}(\mathbf{p})$ of the query point $\mathbf{p}$ in the unwarped space and then querying point membership on A as follows:

$$c_w(\mathbf{p}) = c_A \circ w^{-1}(\mathbf{p})$$

Figure 11 shows an application of the twist operation to two blended implicit cylinders.

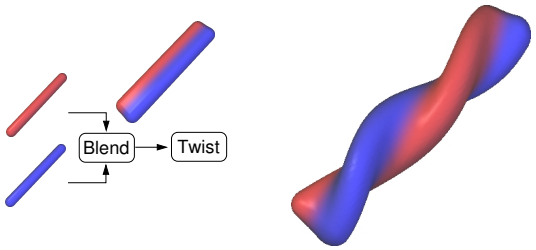


Fig. 11. Twist operation applied to two blended implicit cylinders.

4.5.2 Free-form deformations

Free-form deformations have been first introduce by Sederberg and Parry in [8], and then have been extended by several authors [60,61,62]. Applying

local deformations in the implicit formulation is not straightforward as there is no easy way of computing an analytical formulation for w^{-1}. In our framework, the field function for FFD operations is evaluated using an intermediate mesh representation. The algorithm proceeds as follows:

(1) Generate the mesh $\mathcal{M}_A$ of A.
(2) Apply the deformation to $\mathcal{M}_A$ by transforming the vertices of $\mathcal{M}_A$ according to w.
(3) Evaluate the field function using the distance to the deformed mesh $\mathcal{M}_A$.

FFD nodes hold a mesh representation of their own resulting surface using this method. Subsequent field function, gradient and point membership queries are performed directly on this mesh without further recursion down to the subtree. If A is a single primitive, the field function is evaluated using the potential field function of A. If the local deformation extends to several primitives, their local blending properties are replaced by a new potential field function that is associated with the computed mesh. Therefore, these nodes should be located at the lowest levels of the tree in order to preserve these local properties if they are involved in further operations.

Figure 12 illustrates our free-form deformation tool, applied to the bottle model of Figure 4. Before applying the deformation, this model has been first polygonized as described in Section 5.

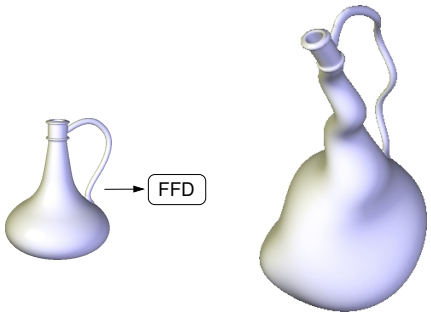


Fig. 12. Free-form deformation applied on the implicit bottle model of Figure 4.

4.5.3 Affine transformations

Affine transformations can be applied to implicit surfaces as warp functions. The resulting field function can be defined using the inverse transformation in the same way as for Barr deformations. However, this requires to evaluate w^{-1} for every queries on the subtree. Benefitting from the distributivity of affine transformations over Boolean and blending operations, Fox et al. [63] prescribed to remove affine transformation nodes and directly integrate them into the parameters of the primitives. In their method, the process is blocked by warping nodes. In our system, we have extended the algorithm to our local blending nodes and optimized it so that Euclidean similarities, including rigid

transformations plus uniform scaling, can be cast through Barr and FFD nodes either. For these three operations, affine transformations are transmitted to both the arguments and space parameters of the operation.

Let $a\{N\}$ denote an affine transformation operation with child node N. Our algorithm proceeds as follows for each kind of node:

- If N is a primitive, we have:

  $$a\{N\} = N'$$

  where N' is the transformed primitive.
- If $o\{A, B\}$ is a Boolean or a global blending operation with child nodes A and B, we have the following equivalence:

  $$a\{o\{A, B\}\} = o\{a\{A\}, a\{B\}\}$$

- If $o\{A, B, R\}$ is a local blending operation with child nodes A, B and C, we have:

  $$a\{o\{A, B, C\}\} = o\{a\{A\}, a\{B\}, a\{C\}\}$$

- Let $FFD_{\mathcal{G}}\{N\}$ a FFD operation with parameter $\mathcal{G}$ denoting the grid in which the child node N is embedded to be deformed. If a is an Euclidean similarity, we have:

  $$a\{FFD_{\mathcal{G}}\{N\}\} = FFD_{\mathcal{G}'}\{a\{N\}\}$$

  where $\mathcal{G}'$ is the transformed grid.
- Let $w_{\mathcal{F}}\{N\}$ a Barr operation with child node N that is performed in a local frame $\mathcal{F}$. Is a is an Euclidean similarity, we have:

  $$a\{w_{\mathcal{F}}\{N\}\} = w_{\mathcal{F}'}\{a\{N\}\}$$

  where $\mathcal{F}'$ denotes the transformed local frame.

Figure 13 illustrates the transmission of a rotation to a complex object through a FFD node.

5 Polygonizing the HybridTree

The resulting surface of a HybridTree may be triangulated using standard implicit surface meshing techniques, thanks to the previously defined field functions. However, these techniques rely on many evaluations of the potential field function, which is computationally demanding in the general case. In

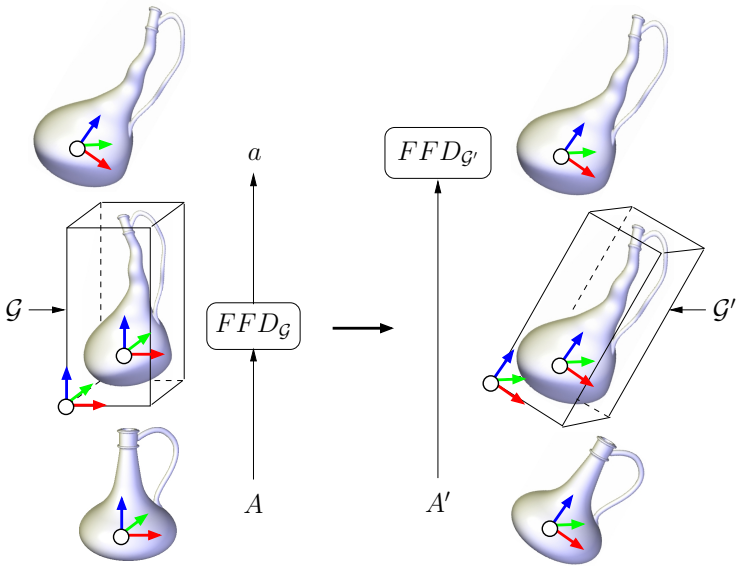


Fig. 13. Transmission of a rotation through a FFD node.

particular, sampling the field function of a mesh primitive is an expensive task that is clearly unprofitable if the mesh surface only interacts locally with other primitives.

For efficient meshing, we developed an incremental approach that preserves existing mesh surfaces as much as possible and optimizes the mesh generation for every kind of node using specific meshing algorithms. Local meshes are merged together at blending and Boolean nodes to form the resulting mesh surface. The following paragraphs detail our meshing methods for skeletal implicit primitives and for the different editing operations.

5.1 Skeletal implicit primitives

In our system, every implicit primitive automatically generates its mesh representation very efficiently for a target level of detail. For every kind of skeleton, we developed a specific and optimized meshing procedure that outputs a manifold mesh characterized by an almost uniform sampling distribution and regular connectivity (Figure 14).

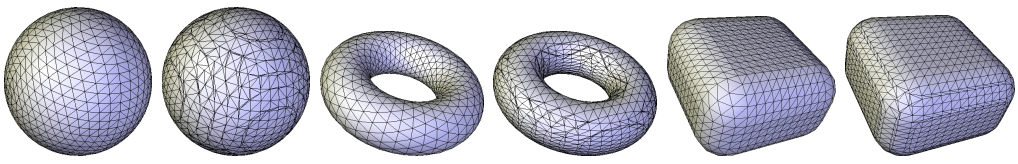


Fig. 14. Meshes obtained by direct meshing (left) vs. Marching Cubes results (right) for identical target edge length. From left to right: point, circle and box skeletons.

As mentioned previously, the T level surface of a skeletal implicit primitive

can be described by sweeping of a sphere of constant radius $r_T = g^{-1}(T)$ along the skeleton. This isosurface corresponds to a $2d$-manifold and will be polygonized if and only if $r_T > 0$. For most primitives, this surface can be defined as a patchwork of simple surface pieces such as portions of sphere or cylinder, planes, disks, which facilitates direct meshing. After having identified the different components for a given primitive, we sample each part iteratively while establishing consistent connectivity relations. The level of detail is fixed by the choice of a maximum edge length. During the sampling process, we take benefit from symmetries whenever possible. For instance, we just need to sample an octant of a sphere for a point primitive [64]. The sampled geometry is then replicated with respect to symmetry axes or symmetry planes. For a box primitive, we need to sample an octant of a sphere, a quadrant of a cylinder and a rectangle.

Primitive	Direct meshing		Marching Cubes		
	Time	#Triangles	Time	#Triangles	#Cells
Point	4	57,800	567	75,356	166^3
Circle	1	57,000	353	87,280	116^3
Box	4	84,780	765	105,912	139^3

Table 2
Polygonization timings (in milliseconds) and number of triangles for both direct meshing and Marching Cubes for a fine target resolution (computations performed on a Pentium IV 3.0GHz - 1GB RAM workstation).

In comparison to standard implicit surface triangulation algorithms like the Marching Cubes algorithm [14,15], timings demonstrate that our direct meshing approach accelerates the polygonization process up to 200 times and produces up to 30% fewer, better shaped triangles. Figure 14 shows the meshes of point, circle and box primitives produced by our algorithms which compares favorably to the mesh outputs of the Marching Cubes algorithm [14] for the same level of detail. Computation timings and number of triangles are given in Table 2 for a finer sampling resolution. The Marching Cubes algorithm we employed relies on a brute-force voxel decomposition of space.

5.2 Blending operations

We generate the mesh at blending nodes using the implicit representation of the surface. The computation of each sample point on the isosurface is an expensive task that requires several field function evaluations. To save computational time and preserve existing mesh surfaces, we restrict the computation

of new sample points to blending regions. Wherever two objects do not overlap very much, we observed that it is better to generate the meshes of these objects before combining them rather than computing the overall mesh from scratch.

In the following paragraphs, we describe our meshing algorithms for global and local blending nodes. For the global case, we will first assume that the objects that blend together are associated with positive potential fields, which is the most common situation. The algorithm for blending nodes involving negative potential fields will be exposed subsequently.

5.2.1 Global blending

In [12], we proposed to use the Marching Triangles technique to generate the mesh in blending regions. This technique works well for smooth surfaces with smooth gradient variations, but often fails at producing satisfying results in other cases, particularly in the presence of sharp features. We develop here an alternative approach based on the Marching Cubes technique [14]. We present two kinds of algorithms that differ in the way they process a HybridTree. The first *node-based* algorithm recursively traverses the tree data-structure, every node generating the mesh of its own subtree as prescribed in [12]. The second algorithm, called *primitive-based*, is more independent from the tree structure and focuses on the interactions between primitives.

Node-based algorithm

Let A and B to objects that globally blend together into an object C. We suppose that A and B generate positive potential fields. If A and B are only partially blended, then we create the mesh of C after the meshes $\mathcal{M}_A$ and $\mathcal{M}_B$. We use the Marching Cubes algorithm to generate the mesh in the blending region. Otherwise, the whole mesh of C is generated by applying the Marching Cubes algorithm from scratch. In both cases, Marching Cubes sample points are computed by evaluating the local field function $f_{A+B}(\mathbf{p})$.

To determine the most favorable approach, we estimate how much the models A and B overlap, i.e. which proportion of volume each one shares in the blending region. Between A and B, blending occurs in the regions of space where f_A and f_B are both positive. These blending regions are enclosed in the intersection of $\mathcal{B}_A$ and $\mathcal{B}_B$. We introduce a ratio ρ called *overlapping ratio* such that $0 \leq \rho \leq 1$, which is computed as follows:

$$\rho = \frac{V_{A \cap B}}{\min(V_A, V_B)}$$

where V_A, V_B and $V_{A\cap B}$ denote the volume of the bounding boxes $\mathcal{B}_A$, $\mathcal{B}_B$ and $\mathcal{B}_A \cap \mathcal{B}_B$ respectively. Let $0 \leq \rho_0 \leq 1$ denote a fixed threshold. Without loss of generality, assume $V_A \leq V_B$. Our algorithm then proceeds as follows:

(1) If $\rho > \rho_0$, then:
   (a) If $\frac{V_A}{V_B} \leq \rho_0$, then apply the Marching Cubes algorithm in the box that bounds $\mathcal{B}_A \cup \mathcal{B}_B$.
   (b) Else:
      - Create the mesh $\mathcal{M}_B$ of B.
      - Remove the triangles of $\mathcal{M}_B$ that have at least one vertex $\mathbf{p}_i$ such that $\mathbf{p}_i \in \mathcal{B}_A$.
      - Apply the Marching Cubes algorithm in $\mathcal{B}_A$.

(2) Else:
   (a) Create the meshes $\mathcal{M}_A$ and $\mathcal{M}_B$ of A and B respectively.
   (b) Remove the triangles of $\mathcal{M}_A$ and $\mathcal{M}_B$ that have at least one vertex $\mathbf{p}_i$ such that $\mathbf{p}_i \in \mathcal{B}_A \cap \mathcal{B}_B$.
   (c) Apply the Marching Cubes algorithm in $\mathcal{B}_A \cap \mathcal{B}_B$.

We invoke a crack fixing algorithm after each local Marching Cubes meshing process in order to bridge the narrow gap between boundary triangles and output a closed manifold mesh. Our algorithm creates new triangles from the boundary edges. For one given pair of contours, we identify the closest neighbors from one side to the other and corresponding vertices are connected together. This technique works well if the meshes $\mathcal{M}_A$ and $\mathcal{M}_B$ have triangles of almost the same size and if corresponding boundaries are very close to each other. The second condition is achieved by clipping the triangles of $\mathcal{M}_A$ and/or $\mathcal{M}_B$ that intersect the bounding box of the blending region against this box [65]. When the lengths of the facing edges are too different, the boundary triangles are refined locally in order to make the scale of the boundary edges compatible.

Before launching the polygonization process, we evaluate the ratio ρ for terminal blending nodes and propagate the information up the tree structure. We cluster consecutive blending nodes whose child nodes strongly overlap along the same branch so as to treat them in one single Marching Cubes pass at the highest possible level of the branch, out of efficiency. The clustering process along a branch stops as soon as a non-blending node is encountered.

The user can provide a value for ρ_0 or directly specify which method should be used for each blending operation involved in a tree. We choose $\rho_0 = 0.5$ as a default threshold, which appears as a good guess in most cases. The motivation for this choice is illustrated in Figure 15. We compared the time taken to polygonize two blended implicit spheres using our local approach and a global Marching Cubes polygonization for random locations of the two primitives. It can be observed on the diagram that our node-based local approach is up to

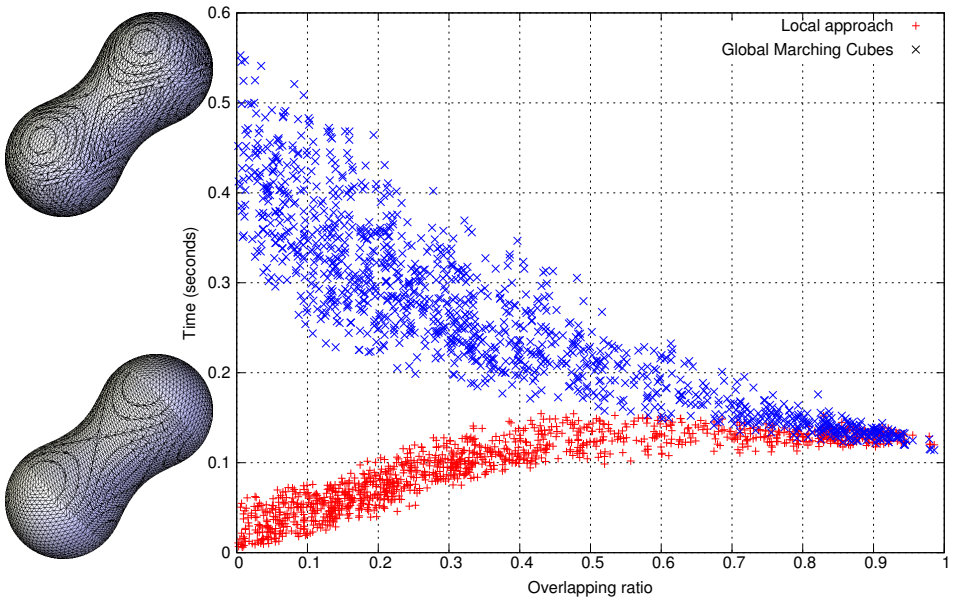


Fig. 15. Polygonization time vs. overlapping ratio for 10^3 polygonization queries of two blended implicit spheres with varying locations in space. The top-left image shows a mesh result for the global Marching Cubes and the bottom-left image shows a mesh result for the local approach.

five times faster when the overlapping ratio is less than 0.5.

Based on the analysis of the bounding box hierarchy, this approach is particularly simple and systematic. However, its performance clearly depends on the structuration of the HybridTree. Some blending mesh parts generated from different blending subtrees may be locally destroyed and remeshed several times due to a non optimal binary tree structure. This limitation could be compensated by a restructuration of the tree, but this kind of optimization is known to be computationally expensive [51], and would not improve every configurations, as illustrated by the model in Figure 16. Here, the blending node would be globally polygonized using Marching Cubes although blending only occurs locally.

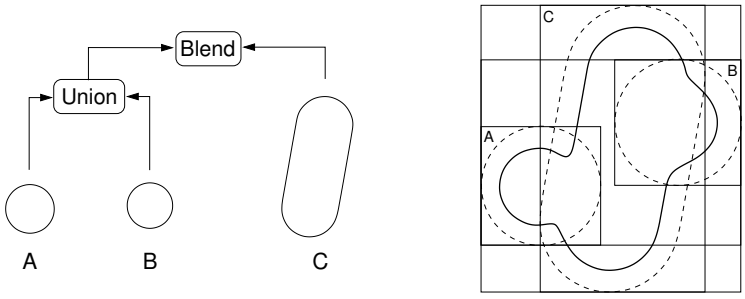


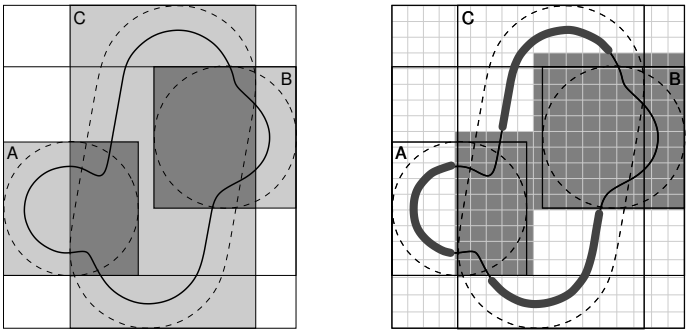
Fig. 16. A HybridTree configuration for which our node-based polygonization technique for blending nodes is not optimal.

The difficulty is in fact intrinsic to the binary tree representation that cannot

explicitely describe the interactions that occur between more than two primitives. To cope with non optimal HybridTree structures, we propose to extend our first algorithm in the spirit of the space decomposition approach introduced by Fox et al. in [63] that focuses on how primitives interact in space rather than on the global tree structure.

Primitive-based algorithm

We distinguish two kinds of regions of space: the regions $\mathcal{R}_P$ that are influenced by a single primitive (Figure 17(a), light gray), and the regions $\mathcal{R}_+$ where blending occurs (Figure 17(a), dark grey). Our goal is to polygonize primitives using direct meshing in regions $\mathcal{R}_P$ and to apply the Marching Cubes algorithm in a single pass in regions $\mathcal{R}_+$. For this purpose, the HybridTree is embedded in a regular grid and we rely on a modified Marching Cubes algorithm to compute sample points in the bounding boxes of the blending regions.



(a) (b)

Fig. 17. Our primitive-based optimized local meshing approach.

Let $\{P_j\}, j = 1..n$ denote the set of primitives involved in a tree, with bounding boxes $\mathcal{B}_j$. We define the overlapping ratio ρ_i for a primitive P_i as follows :

$$\rho_i = \frac{V_O(\mathcal{B}_i)}{V(\mathcal{B}_i)}$$

where $V(\mathcal{B}_i)$ denotes the volume of $\mathcal{B}_i$ and V_O is the amount of volume of $\mathcal{B}_i$ that is shared with the bounding boxes of the other primitives, i.e.:

$$V_O(\mathcal{B}_i) = V\left(\bigcup_{j \neq i} \mathcal{B}_i \cap \mathcal{B}_j\right)$$

where $V\left(\cup_{k=1}^{n}\mathcal{B}_k\right)$ is computed using the inclusion-exclusion formula:

$$V\left(\bigcup_{k=1}^{n}\mathcal{B}_k\right)=\sum_{l=1}^{n}(-1)^{l+1}\sum_{m_1<m_2<\ldots<m_l}V\left(\mathcal{B}_{m_1}\cap\mathcal{B}_{m_2}\cap\ldots\cap\mathcal{B}_{m_l}\right)$$

Let $\mathcal{G}$ denote a regular grid in which the HybridTree is embedded. All the grid cells are first initialized to "0". We proceed as follows for all primitives P_i:

(1) Compute the set $\{\mathcal{B}_i\cap\mathcal{B}_j\}$, $j\neq i$, $\mathcal{B}_i\cap\mathcal{B}_j\neq\emptyset$, such that the first common ancestor A of P_i and P_j is a blending node and P_j is not a right descendent of a difference operation located below A.
(2) Align $\mathcal{B}_i$ and every box $\mathcal{B}_i\cap\mathcal{B}_j$, $j\neq i$ on the grid $\mathcal{G}$.
(3) Compute ρ_i.
(4) If $\rho_i<\rho_0$ then:
  (a) Create the mesh $\mathcal{M}_i$ of P_i.
  (b) Remove the triangles of $\mathcal{M}_i$ that have at least one vertex $\mathbf{p}_i$ such that $\mathbf{p}_i\in\cup_{j\neq i}\mathcal{B}_j$.
  (c) Mark "1" all cells of $\mathcal{G}$ that lie in $\cup_{j\neq i}\mathcal{B}_j$.
(5) Else, mark "1" all cells of $\mathcal{G}$ that lie in $\mathcal{B}_i$.

Our modified Marching Cubes algorithm then computes sample points along the edges of "1" cells and triangulates these cells.

Using this method, more implicit primitive mesh parts are obtained by direct meshing and more existing mesh parts can be preserved. In Figure 17(b), the regions that are polygonized using the direct meshing strategy are depicted with a bold contour. Figure 18 shows three different mesh outputs of the restored Igea model from Figure 23. In this model, several implicit primitives are locally blended with the reconstructed mesh model of the Igea point set, yielding a configuration that is similar to the one in figure 16. The polygonization results were obtained by applying the global Marching Cubes algorithm (left), then our node-based algorithm (center), and finally our primitive-based algorithm (right). Computational timings are reported in Table 3, that also shows the number of new sample points computed on the surface through the Marching Cubes technique. For this model, our primitive-based algorithm is two times faster than our node-based algorithm.

The computation of the intersection boxes between every pair of primitives is achieved in time $O(n^2)$, where n denotes the number of primitives. For a given primitive, the overlapping ratio is evaluated in time $O(n^2)$ in the worst case. As a consequence, the performance of our primitive-based algorithm may decline over a set of primitives that are all tightly blended together. However, in practice, the number of primitives that effectively contribute to the final potential field at a given point in space is generally small compared to the overall number of primitives involved in a particular model.

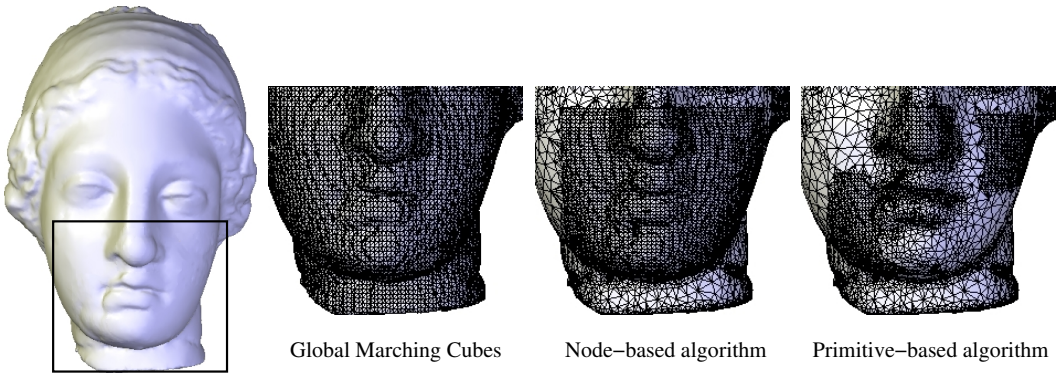


Fig. 18. Several polygonization results for the restored Igea model from Figure 23.

Method	Polyg. time	#Sample points M.C.	#Triangles
Global M.C.	1,533.02	54,420	108,836
Node-based	19.82	8,299	61,980
Primitive-based	9.67	2,592	58,458

Table 3
Polygonization timings (in seconds), number of new sample points computed by the Marching Cubes technique and number of triangles for global Marching Cubes, node-based and primitive-based meshing (computations performed on a Pentium IV 3.0GHz - 1GB workstation).

Our primitive-based method also requires to store a grid of size m^3 with only 1 bit per cell. For a grid with 300^3 cells, which was the maximum in our tests, this represents less than 3.5 megabytes of main memory. The time for traversing the set of cells is negligible against the polygonization process. If more precision is needed or if memory is a critical resource, then the node-based approach may be more profitable.

5.2.2 Negative blending

Suppose that A generates a positive potential field and B a negative one. In this case, our algorithm proceeds as follows:

(1) Create the mesh $\mathcal{M}_A$ of A.
(2) Remove the triangles of $\mathcal{M}_A$ with at least one vertex $\mathbf{p}_i$ such that $\mathbf{p}_i \in \mathcal{B}_B$.
(3) Apply the Marching Cubes algorithm in $\mathcal{B}_B$ and invoke the crack fixing algorithm to close the gap.

5.2.3 Local blending

Here we suppose that A and B generate positive potential fields, which is not required for C. The polygonization local blending nodes is achieved as follows:

(1) Create the meshes $\mathcal{M}_A$ and $\mathcal{M}_B$ of A and B respectively.
(2) Compute the mesh $\mathcal{M}_D$ of the union D between A and B.
(3) Remove the triangles of $\mathcal{M}_D$ that have at least one vertex $\mathbf{p}_i$ such that $\mathbf{p}_i \in \mathcal{B}_A \cap \mathcal{B}_C$ or $\mathbf{p}_i \in \mathcal{B}_A \cap \mathcal{B}_C$.
(4) Apply the Marching Cubes algorithm in $\mathcal{B}_R$ and invoke the crack fixing algorithm to close the gap.

5.3 Boolean operations

Computing the mesh resulting from Boolean operations is achieved as performed by standard B-Rep modelers. Our approach takes advantage of the dual implicit/mesh representation of the HybridTree. We rely on the implicit representation of the child nodes to perform point membership classification efficiently. The algorithm may be written as follows for any of the union, intersection or difference operations:

(1) Create the meshes $\mathcal{M}_A$ and $\mathcal{M}_B$ of A and B respectively.
(2) If $\mathcal{B}_A$ and $\mathcal{B}_B$ overlap, then compute the resulting mesh surface using the point membership function of A and B for point membership classification.

To determine whether two triangles overlap and clip them properly, we use the fast and robust triangle-triangle overlap test proposed by Guigue and Devillers [66].

5.4 Warping operations

We first create the mesh $\mathcal{M}_A$ of the child node A. Then the deformation is applied to the mesh $\mathcal{M}_A$ by simply changing the coordinates of the vertices of the mesh $\mathbf{p}_i$ into $w(\mathbf{p}_i)$ so as to obtain the deformed mesh. Translation, rotation and uniform scaling preserve the aspect ratio of the triangles, whereas non uniform scaling or twisting, tapering and bending may stretch the triangles into flat triangles. In those cases, we apply a simple local remeshing process based on edge collapse [67] and vertex insertion to get better shaped triangles.

6 Results and discussion

In this section, we present some complex models created by combining and deforming skeletal implicit models built from hundreds of implicit primitives, and meshes and point sets with tens of thousands of elements. Table 4 reports the timings for polygonizing the final models (in seconds), as well as the overall number of triangles. The given preprocessing timings take into account the time taken to build the bounding box hierarchy for mesh models and the initialization of the Marching Cubes grid when the second local meshing algorithm is used for blending nodes. These timings do not include the time needed to reconstruct a mesh model from an input point set when involved in our hybrid models. For the Igea point set that we used, the time taken to produce a triangle mesh was 64 seconds. Measures were performed on a Pentium IV 3.0GHz - 1GB RAM workstation.

Figure	Preproc. time	Polyg. time	#Triangles
1, 19	4.63	63.85	171,562
9	1.72	48.16	105,467
20	0	14.02	121,271
21	1.97	21.24	94,862
22	6.85	56.35	269,698
23	1.58	9.12	58,458

Table 4
Preprocessing and polygonization timings (in seconds) and number of triangles for polygonizing several complex hybrid models.

6.1 Free-form modeling

The winged snake-woman Figures 1 and 19 show blending and Boolean operations applied to implicit and mesh input models. The original snake-woman (Figure 1(a)) is an implicit model built from 250 spline implicit primitives blended together, which is stored in our own library of models. The body has been first blended with a mesh of the Igea model (62,323 triangles) that was automatically reconstructed from the point set in Figure 3, and with the wings of the Victory of Samothrace (16,340 triangles). The mesh creation process first invokes the polygonization of the implicit snake-woman model. The Marching Cubes algorithm is used as all implicit primitives are overlapping much. The resulting mesh consists of 121,524 triangles, and took 6 seconds to generate. The head has been removed using Boolean difference

with an implicit sphere primitive, and the body has been blended with the Igea model using our local meshing method. The wings were extracted from the Victory of Samothrace mesh model by intersecting the original model with an implicit box. The wings and the modified snake-woman model have finally been blended together using the local meshing method.

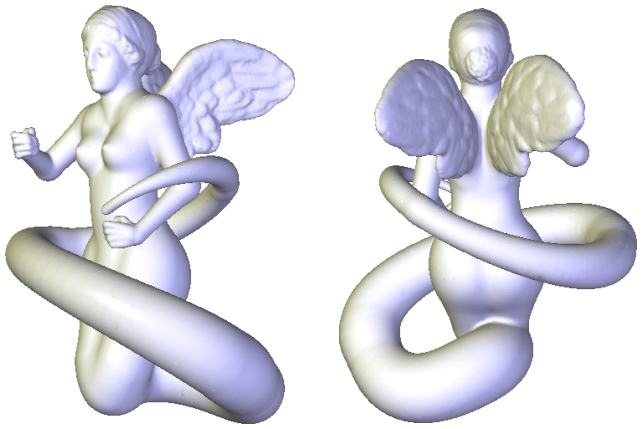


Fig. 19. The winged snake-woman model of Figure 1.

The bowl The bowl in Figure 20 has been created using blending operations. The interior of the Igea model has been carved using a negative potential field generated by a cylinder implicit primitive. Handles built from two implicit circle primitives have then been added using our local blending operation.

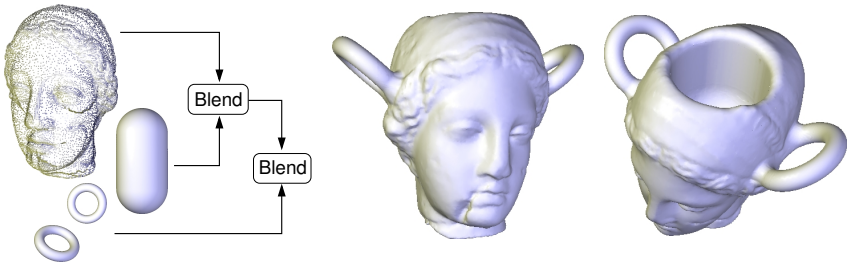


Fig. 20. A bowl created from the Igea model.

The bottle The bottle in Figure 21 has been create from the implicit bottle model of Figure 4 that incorporates 5 complex skeletal implicit primitives. We first applied our Free-Form Deformation tool, which necessitates the polygonization of the bottle model. Additionally, 12 holes have been created using Boolean differences with implicit spheres.

The Victory Figure 22 shows a statue model based on the Victory of Samothrace mesh model (187,072 triangles), that has no head and no arms. We picked up the arms of the original snake-woman implicit model, that consist of 18 implicit spline primitives each, and we have blended them locally

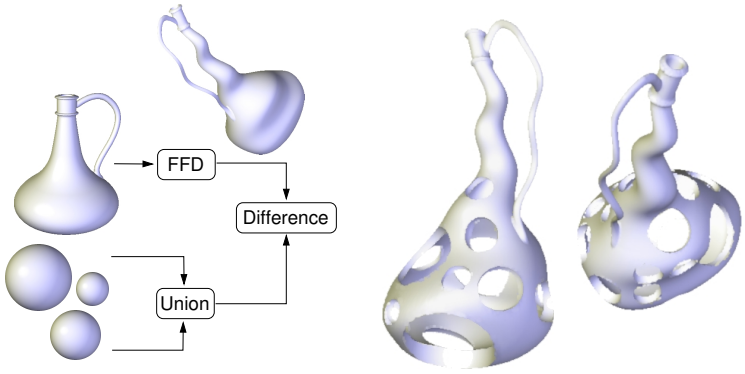


Fig. 21. Bottle with holes.

with the Victory of Samothrace mesh model using implicit spheres located at each shoulder. We have blended the resulting surface locally with the Igea head using an implicit cylinder placed around the neck. We have finally completed our custom Victory model by adding a shepherd's crook in the right hand.

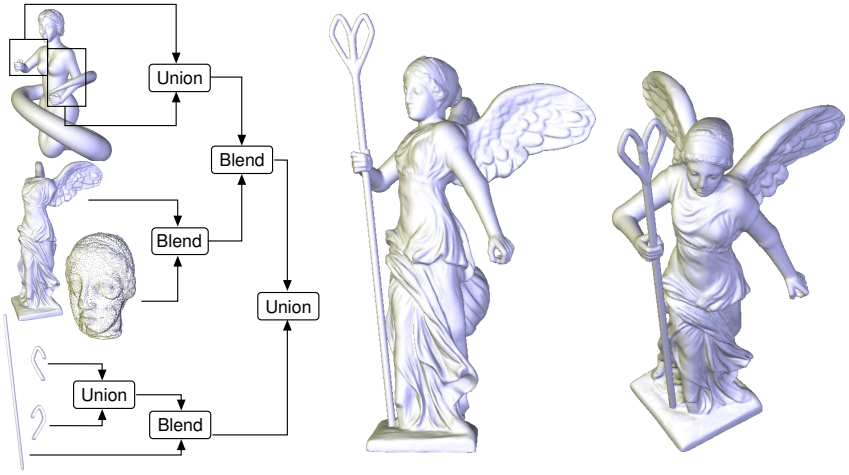


Fig. 22. A Victory model.

6.2 Virtual restoration of artwork

Our model is well-suited for modeling complex shapes either from existing models or from scratch. It could also be advantagely used for the purpose of digital preservation of cultural heritage artwork, which has become a very challenging research domain. Our HybridTree structure can be efficiently used to simulate restoration or natural phenomena [68] effects on digitized pieces of artwork and it naturally maintains the history of every operation, which is useful for archiving purposes.

Figure 23 shows a virtual restoration process on the Igea model using the HybridTree. We were interested in filling in the ridges on the right of the chin

and on the left cheek, and restoring the nose, exactly as a specialist could do. We used our blending tools to simulate cementing in a very intuitive and realistic way.

We have manually placed implicit spline primitives along each ridge and one implicit point primitive at the tip of the nose. The parameters of the potential field functions have also been set by hand for each primitive. The primitives have been then blended with the Igea mesh model so as to produce the final mesh representation. We have built an independent subtree for the set of primitives of the chin and the another for the nose. The former has been polygonized using the Marching Cubes algorithm, as the primitives overlap much. Then, the resulting mesh has been blended with the Igea model using the local method. The same approach has been used for the nose.

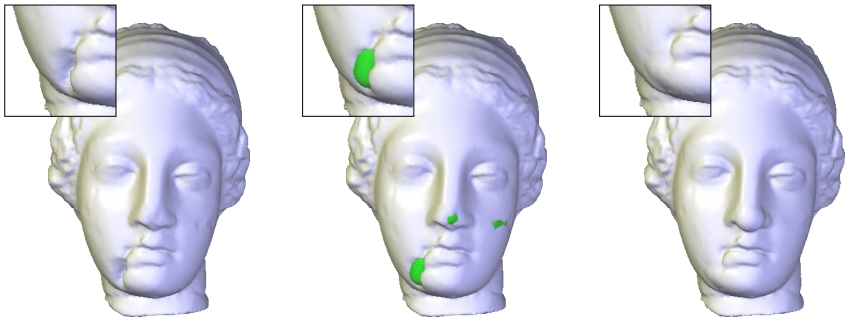


Fig. 23. Virtual restoration stages of the Igea Greek artifact. The initial reconstructed Igea mesh is on the left. In the center image, material has been added to fill in some cavities. On the right is shown the partially restored model.

6.3 Discussion

Performance Our system can handle complex implicit primitives and polygonal meshes of up to 25,000 triangles at interactive rates. Free-form deformations as well as local blending may be performed interactively for not too fine resolutions. Boolean operations combining small implicit primitives or meshes compared to the overall size of the final object may also be performed at interactive rates.

The conversion step between triangles meshes and implicit surfaces is a critical limiting factor regarding computational performance. The computation of the potential field function generated by a mesh at a given point in space remains computationally expensive, despite our acceleration technique. Experiments demonstrate that a field function query performed on a complex mesh can have a cost in time that is up to several hundreds times the cost of the same evaluation performed on a point primitive.

For interactive shape design or animation, the evaluation could be accelerated

by sampling the potential field on a regular grid and caching computed field values [69]. An approximation of the surface can be retrieved by tricubic interpolation. This approach involves an increased cost in memory and possible loss of geometric and topological information.

Storage The HybridTree data-structure significantly reduces the amount of memory needed for storing complex models. Contrary to Level Set [27] or Adaptive Distance Fields models [28], we do not store any voxel grid or octree, which saves memory. The use of complex implicit skeletal primitives enables us to design complex shapes with a very compact representation. The snake-woman model represented in Figure 1 was created by blending a few hundred spline skeletal primitives together. The corresponding HybridTree representation takes less than 64 kilobytes in memory.

Shape control The ability to combine mesh models and skeletal implicit surfaces in a coherent framework not only extends the range of models that can be created but also permits us to have a tight control when editing our models.

The implicit surface representation enables blending of meshes of arbitrary topology and geometry. This compares favorably with other specific mesh blending methods such as [50] or [32] that impose some geometric or topological restrictions. Moreover, our local blending technique provides fine control on the way shapes blend together. The designer may simply tune the radius of influence for mesh or implicit primitives so as to control the geometry of the blend with other objects. The implicit representation also provides means of creating negative blending between shapes, which is useful for simulating carvings. Eventually, our Free-Form Deformation tool enables very intuitive, non restrictive local deformations on hybrid models.

7 Conclusion and future work

In this paper, we have proposed a new hybrid shape representation. Our model combines skeletal implicit surfaces, triangle meshes and point set models in a coherent framework. The HybridTree's evaluation system is designed to exploit the complementary advantages of these geometric models. The core of our current system is based on a dual skeletal implicit/triangle mesh representation. Editing operations are performed in the most suitable representation in a totally transparent way for the user. The mesh representation is useful for fast visualization and free-form deformations, and the implicit one lend themselves for Boolean, and local and global blending. The HybridTree is evaluated

through field function, gradient, point membership classification and polygonization queries that are optimized for every kind of node.

In the near future, we plan to extend the integration of the point set representation into the HybridTree. This representation, that avoids the management of connectivity relations, could be interesting for interactive visualization. We will also investigate the automatic management of levels of detail in the HybridTree. We think that it should be possible to combine skeletal implicit primitives with levels of detail as presented in [70] with multiresolution meshes and subdivision surfaces. Another interesting research field would be to bring unicity in the representation of the information and achieve reversibility in the evaluation process.

Acknowledgements

This work is part of the Art3D project (ACI Masses de données) and is supported by the Ministère de l'Éducation Nationale, de la Recherche et de la Technologie (MENRT) and by the Centre National de la Recherche Scientifique (CNRS).

The snake-woman implicit model is courtesy Tiphaine Accary. The model of the Victory of Samothrace is provided courtesy of Pascal Lefebvre-Albaret from Technodigit. Other point set and mesh models are courtesy of Stanford Scanning Repository and Cyberware Inc.

References

[1] C. Bajaj, J. Blinn, J. Bloomenthal, M.-P. Cani-Gascuel, A. Rockwood, B. Wyvill, G. Wyvill, Introduction to Implicit Surfaces, Morgan-Kaufmann, 1997.

[2] L. Velho, J. Gomes, L. H. Figueiredo, Implicit Objects in Computer Graphics, Springer Verlag, New York, 2002.

[3] J. F. Blinn, A generalization of algebraic surface drawing, ACM Transactions on Graphics 1 (3) (1982) 235–256.

[4] B. Crespin, C. Blanc, C. Schlick, Implicit Sweep Objects, in: Proc. Eurographics, Vol. 15, 1996, pp. 165–174.

[5] A. Pasko, V. Adzhizev, A. Sourin, V. Savchenko, Function Representation in Geometric Modeling: Concepts, Implementation and Applications, The Visual Computer 11 (8) (1995) 429–446.

[6] B. Wyvill, E. Galin, A. Guy, Extending The CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System, Computer Graphics Forum 18 (2) (1999) 149–158.

[7] A. Barbier, E. Galin, S. Akkouche, Controlled Metamorphosis of Animated Objects, in: Proc. Shape Modeling International, 2003, pp. 184–196.

[8] T. W. Sederberg, S. R. Parry, Free-Form Deformation of Solid Geometric Models, in: Proc. SIGGRAPH, 1986, pp. 151–160.

[9] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, H.-P. Seidel, Laplacian Surface Editing, in: Proc. Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2004, pp. 179–188.

[10] M. Pauly, R. Keiser, L. P. Kobbelt, M. Gross, Shape Modeling with Point-Sampled Geometry, ACM Transactions on Graphics 22 (3) (2003) 641–650.

[11] F. Cazals, J. Giesen, Delaunay Triangulation based Surface Reconstruction: Ideas and Algorithms, Tech. Rep. 5393, INRIA (November 2004).

[12] R. Allègre, A. Barbier, E. Galin, S. Akkouche, A hybrid shape representation for free-form modeling, in: Proc. Shape Modeling International, 2004, pp. 7–18.

[13] R. Allègre, R. Chaine, S. Akkouche, Convection-Driven Dynamic Surface Reconstruction, in: Proc. Shape Modeling International, IEEE Computer Society Press, 2005, pp. 33–42.

[14] W. E. Lorensen, H. E. Cline, Marching Cubes : A high Resolution 3D surface reconstruction algorithm, Computer Graphics (Proc. SIGGRAPH) 21 (4) (1987) 163–169.

[15] L. P. Kobbelt, M. Botsch, U. Schwanecke, H.-P. Seidel, Feature Sensitive Surface Extraction from Volume Data, in: Proc. SIGGRAPH, 2001, pp. 57–66.

[16] A. P. Witkin, P. S. Heckbert, Using particles to sample and control implicit surfaces, Computer Graphics 28 (2) (1994) 269–277.

[17] A. Hilton, A. J. Stoddart, J. Illingworth, T. Windeatt, Marching Triangles: Range image fusion for complex object modelling, in: IEEE International Conference on Image Processing, 1996, pp. 381–384.

[18] S. Akkouche, E. Galin, Adaptive Implicit Surface Polygonization using Marching Triangles, Computer Graphics Forum 20 (2) (2001) 67–80.

[19] J.-D. Boissonnat, S. Oudot, Provably Good Surface Sampling and Approximation, in: Proc. Symposium on Geometry Processing, 2003, pp. 9–18.

[20] J.-D. Boissonnat, S. Oudot, An effective Condition for Sampling Surfaces with Guarantees, in: Proc. Symposium on Solid Modeling and Applications, 2004, pp. 101–112.

[21] J.-D. Boissonnat, D. Cohen-Steiner, G. Vegter, Isotopic Implicit Surface Meshing, in: Proc. ACM Symposium on Theory of Computing, 2004, pp. 301–309.

[22] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface Reconstruction from Unorganized Points, Computer Graphics (Proc. SIGGRAPH) 26 (2) (1992) 71–78.

[23] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, Reconstruction and Representation of 3D Objects with Radial Basis Functions, in: Proc. SIGGRAPH, 2001, pp. 67–76.

[24] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.-P. Seidel, Multi-level Partition of Unity Implicits, ACM Transactions on Graphics 22 (3) (2003) 463–470.

[25] G. Turk, J. F. O'Brien, Modelling with Implicit Surfaces that Interpolate, ACM Transactions on Graphics 21 (4) (2002) 855–873.

[26] C. Shen, J. F. O'Brien, J. R. Shewchuk, Interpolating and Approximating Implicit Surfaces from Polygon Soup, ACM Transactions on Graphics 23 (3) (2004) 896–904.

[27] K. Museth, D. E. Breen, R. T. Whitacker, A. H. Barr, Level Set Surface Editing Operators, ACM Transactions on Graphics 21 (3) (2002) 330–338.

[28] S. F. Frisken, R. N. Perry, A. P. Rockwood, T. R. Jones, Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics, in: Proc. SIGGRAPH, 2000, pp. 249–254.

[29] V. Adzhiev, M. Kazakov, A. Pasko, V. Savchenko, Hybrid System Architecture for Volume Modeling, Computers & Graphics 24 (1) (2000) 194–203.

[30] B. Schmitt, A. Pasko, C. Schlick, Shape-Driven Deformations of Functionally Defined Heterogeneous Volumetric Objects, in: Proc. ACM Graphite 2003, 2003, pp. 127–134.

[31] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, H.-Y. Shum, Mesh Editing with Poisson-based Gradient Field Manipulation, ACM Transactions on Graphics (Proc. SIGGRAPH) 23 (3) (2004) 644–651.

[32] T. Kanai, H. Suzuki, J. Mintani, F. Kimura, Interactive Mesh Fusion Based on Local 3D Metamorphosis, in: Proc. Graphics Interface '99, 1999, pp. 148–156.

[33] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. Dobkin, Modeling by Example, ACM Transactions on Graphics (Proc. SIGGRAPH) 23 (3) (2004) 652–663.

[34] M. Alexa, Recent Advances in Mesh Morphing, Computer Graphics Forum 21 (2) (2002) 173–196.

[35] M. Alexa, Differential Coordinates for Local Mesh Morphing and Deformation, The Visual Computer 19 (2–3) (2003) 105–114.

[36] D. Xu, H. Zhang, Q. Wang, H. Bao, Poisson Shape Interpolation, in: Proc. ACM Symposium on Solid and Physical Modeling, 2005, pp. 267–274.

[37] V. Kraevoy, A. Sheffer, Cross-Parameterization and Compatible Remeshing of 3D Models, ACM Transactions on Graphics (Proc. SIGGRAPH) 23 (3) (2004) 861–869.

[38] J. Schreiner, A. Asirvatham, E. Praun, H. Hoppe, Inter-Surface Mapping, ACM Transactions on Graphics (Proc. SIGGRAPH) 23 (3) (2004) 870–877.

[39] D. Levin, Mesh-Independent Surface Interpolation, Geometric Modeling for Scientific Visualization.

[40] M. Mueller, R. Keiser, A. Nealen, M. Pauly, M. Gross, M. Alexa, Point-Based Animation of Elastic, Plastic, and Melting Objects, in: Proc. ACM Siggraph/Eurographics Symposium on Computer Animation, 2004, pp. 141–151.

[41] M. Pauly, N. Mitra, L. Guibas, Uncertainty and Variability in Point Cloud Surface Data, in: Proc. Symposium on Point-Based Graphics, 2004, pp. 77–84.

[42] R. Kolluri, Provably Good Moving Least Squares, in: Proc. ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 1008–1018.

[43] S. Fleishman, C. T. Silva, D. Cohen-Or, Robust Moving Least-squares Fitting with Sharp Features, ACM Transactions on Graphics (Proc. SIGGRAPH) 24 (3) (2005) 544–552.

[44] P. Reuter, I. Tobor, C. Schlick, S. Dedieu, Point-based Modelling and Rendering using Radial Basis Functions, in: Proc. ACM Graphite 2003, 2003, pp. 111–118.

[45] L. P. Kobbelt, M. Botsch, Freeform Shape Representations for Efficient Geometry Processing, in: Proc. Shape Modeling International, 2003, pp. 111–115.

[46] K. Singh, R. Parent, Implicit Surface Based Deformations of Polyhedral Objects, in: Proc. Implicit Surfaces, 1995.

[47] B. Crespin, Implicit Free-Form Deformations, in: Proc. Implicit Surfaces, 1999, pp. 17–23.

[48] P. Decaudin, Geometric Deformation by Merging a 3D-Object with a Simple Shape, in: Proc. Graphics Interface, 1996, pp. 55–60.

[49] P. Decaudin, A. Gagalowicz, Fusion of 3D Shapes, in: Proc. Computer Animation and Simulation, 1994, pp. 1–14.

[50] K. Singh, R. Parent, Joining Polyhedral Objects using Implicitly Defined Surfaces, The Visual Computer 17 (7) (2001) 415–428.

[51] A. Barbier, E. Galin, S. Akkouche, Complex Skeletal Implicit Surfaces with Levels of Detail, Journal of WSCG 12 (1) (2004) 35–42.

[52] A. H. Barr, Global and Local Deformations of Solid Primitives, Proc. SIGGRAPH 18 (3) (1984) 21–30.

[53] P. Schneider, D. H. Eberly, Geometric Tools for Computer Graphics, Morgan Kaufman Series in Computer Graphics and Geometric Modeling, 2002.

[54] A. Barbier, E. Galin, Fast distance computation between a point and cylinders, cones, line swept spheres and cone-spheres, Journal of Graphics Tools 9 (2) (2004) 31–39.

[55] A. Guéziec, Meshsweeper: Dynamic Point-to-Polygonal-Mesh Distance and Applications, IEEE Transactions on Visualization and Computer Graphics 7 (1) (2001) 47–61.

[56] D. Johnson, E. Cohen, A Framework for Efficient Minimum Distance Computation, in: Proc. Conf. Robotics and Automation, 1998, pp. 3678–3683.

[57] E. Galin, S. Akkouche, Fast Processing of Triangle Meshes using Triangle Fans, in: Proc. Shape Modeling International, 2005, pp. 326–331.

[58] M.-P. Cani, M. Desbrun, Animation of Deformable Models using Implicit Surfaces, IEEE Transactions on Visualization and Computer Graphics 3 (1) (1997) 39–50.

[59] G. Pasko, A. Pasko, M. Ikeda, T.Kunii, Bounded Blending Operations, in: Proc. Shape Modeling International, 2002, pp. 95–104.

[60] S. Coquillart, Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling, in: Proc. SIGGRAPH, 1990, pp. 187–196.

[61] P. Borrel, D. Bechmann, Deformation of n-dimensional objects, in: Proc. Solid Modeling and Applications, 1991, pp. 351–369.

[62] R. MacCracken, K. I. Joy, Free-form Deformations with Lattices of Arbitrary Topology, in: Proc. SIGGRAPH, 1996, pp. 181–188.

[63] M. Fox, C. Galbraith, B. Wyvill, Efficient Implementation of the Blobtree for Rendering Purposes, in: Proc. Shape Modeling International, 2001, pp. 306–314.

[64] J. Ahn, Fast Generation of Ellipsoids, Graphics Gems V (1995) 179–190.

[65] T. A. Möller, Fast 3D Triangle-Box Overlap Testing, Journal of Graphics Tools 6 (1) (2001) 29–33.

[66] P. Guigue, O. Devillers, Fast and Robust Triangle-Triangle Overlap Test Using Orientation Predicates, Journal of Graphics Tools 8 (1) (2003) 25–32.

[67] H. Hoppe, Progressive Meshes, in: Proc. SIGGRAPH, 1996, pp. 99–108.

[68] A. Martinet, E. Galin, B. Desbenoit, S. Akkouche, Procedural Modeling of Cracks and Fractures, in: Proc. Shape Modeling International, 2004, pp. 346–349.

[69] R. Schmidt, B. wyvill, E.Galin, Interactive Implicit Modeling With Hierarchical Spatial Caching, in: Proc. Shape Modeling International, 2005, pp. 104–113.

[70] A. Angelidis, M.-P. Cani, Adaptive Implicit Modeling using Subdivision Curves and Surfaces as Skeletons, in: Proc. Solid Modeling and Applications, 2002, pp. 45–52.