Precise Construction and Control of Implicit Fillets in the BlobTree

Herbert Grasberger*, Andrea Weidlich†, Alexander Wilkie‡, Brian Wyvill§
*University of Victoria, Victoria, Canada, grassi@cs.uvic.ca
†Vienna University of Technology, Vienna, Austria, weidlich@cg.tuwien.ac.at
‡Charles University, Prague, Czech Republic, wilkie@cgg.mff.cuni.cz
§University of Victoria, Victoria, Canada, blob@cs.uvic.ca

***Abstract*—Skeletal implicit modelling systems have been used to design models of both organic and man-made structures, however existing systems lack convenient and accurate methods for users to de ne llets when building prototype engineering models.**

In this work we extend the methodology and skeletal primitives found in the BlobTree and introduce an improved method for modelling llets. This is done by interpolating between hard-edged and soft-edged representations of a primitive, using the eld to control the interpolation. Blending planes are introduced as a way of providing user control for this interpolation. The new methods are expressed as blend operators between two objects and can thus be implemented as a new node in the BlobTree. In addition we introduce an ef cient methodology for modelling hard edged primitives.

***Keywords*-Computer Graphics, implicit modelling, CSG, llets, the BlobTree**

I. Introduction

The roots of our proposed technique lie in the desire to have an intuitive method for the modelling of *fillets* i.e. smooth transitions between adjoining surfaces that appear in a wide variety of modelling contexts. In Constructive Solid Geometry (CSG) models, which are traditionally an exclusive domain of hard edged shape representations, such smooth transitions occur in many real-world objects and can be put into one of the following groups:

1) There are fillets with strong functional constraints, such as the blend surface in the area where an aircraft wing is joined to the fuselage.
2) There are what can be termed *aesthetic blends*, such as the smooth transition between the stem and the cup of a wine glass, that are introduced on purpose by the designer of an object.
3) Finally, there are fillets that are mere side-effects of a manufacturing process, such as welding beads.

While other distinctions are possible, this categorization is useful when viewing the problem from a computer graphics modelling perspective. It is worth noting that blending surfaces of the first kind have a form that is dependent on complex external factors, and are therefore something that is usually not modeled by hand. In such cases, one would e.g. seek to import a CAD model that is the result of numerical optimizations, or similar processes. However, fillets of the second and third type share the notable characteristic that their *exact* shape (in a numerical sense) is often not important, not even for highly accurate predictive image synthesis. They are typically more numerous in an average scene than the first type (welding seams come to mind here), so a reliable and expressive method for placing them makes for a more efficient modelling workflow

The work of this paper is to present a new approach for easy creation of fillets of the second and third type, but as we will see later, it can also be used for modelling purposes beyond simple filleting. The proposed technique is named **C**onstructive **S**olid **B**lobs, since we use the BlobTree [1] as the underlying data structure, but still retain the widely known CSG modelling paradigm, as first described in [2]. Figure 1 shows a comparison of roughly similar models that were created with a traditional CSG approach (only hard edges), the BlobTree (without hard-edged primitives), and finally our approach (a healthy, i.e. semantically sensible, mix of both).

The main contribution of this work is the application of the simple blending technique defined by the BlobTree to the definition of fillets. In addition the BlobTree uses inefficient complex primitives requiring many field evaluations to define simple hard-edged objects such as a cube. We replace these by simple and efficient hard-edged primitives.

The remainder of the paper is structured as follows: we first give an overview on related work and the background of the BlobTree. In the main part we describe our extension to the BlobTree in detail, present some results modeled by our approach, and conclude this paper by giving an outlook to future work.

II. Related Work

A. Other modelling Methods

In the 1980s methods for filleting arbitrary surfaces were developed by [3], who specified a set of desirable properties a blend has to fulfill:

1) tangency with the base surfaces to be blended,
2) curves of tangency with the base surfaces which are a constant distant *(sic)* from the alternate base surface,
3) cross section profiles defined and controlled by the 'shape' of the profile curve and the distance of the tangency curves from the alternate base surface.

The methods described in that paper generate a new surface that meets the above conditions, which can be a complex problem in certain cases. According to [3], such a blend between two surfaces can, in some circumstances,

978-0-7695-4072-6/10 $26.00 © 2010 IEEE
DOI 10.1109/SMI.2010.37

(a) CSG (b) BlobTree (c) CSB

Figure 1: Comparison of a tap modeled with different modelling approaches. Only the different representations of the single primitives and the operators are changed to allow for comparable results. In comparison to CSG, our approach provides mathematically simpler blend operators while still allowing for hard-edged objects as primitives.

yield surfaces of very high order, and can involve polynomial equations up to degree 23. Which is one reason why the general applicability of these results is limited: blends between blend surfaces (which is not an exotic configuration; consider two fillet surfaces, such as welding beads, meeting in a corner) become an almost intractable problem due to the high order of the resulting surfaces. Although a possible solution is discussed, it is still quite complicated to compute.

The approach described in [4] uses cross sections created by parametric curves in order to control the shape of the blend. It allows blends with continuity starting from C^0 up till C^2, depending on the curve used for the cross section.

Another approach for constructing blends was proposed in [5]. The main constraint there was the fact that only blends between objects defined by between Function Representations (FReps) [6] were considered. Using these techniques, is possible to robustly blend all objects that can be created by FReps. In [7] a methodology to control the influence region of the blend and to create round edges was introduced.

In many cases where fillets have to be inserted, blends with a circular cross section are required, as discussed in [8] or [9]. Such blends are closely related to the boundaries of canal surfaces [10], which are created by a rolling ball that remains in tangential contact to the involved surfaces. In the approach described in [11], a ball with constant radius is used to create this surface, while other approaches [12] use a ball with varying radius. However, such canal surfaces can be of a very high order as well, which considerably complicates their evaluation in a renderer due to numerical stability issues, and the high computational cost associated with the necessary interval arithmetic operations.

B. The BlobTree

The BlobTree extended existing skeletal implicit surface modelling techniques [13] by introducing a unified structure in which nodes could represent arbitrary blends between objects as well as Boolean operations and warping at local and global level. For visualisation the tree may be traversed and a field-value and a gradient returned for an arbitrary point in space.

One reason why it makes sense to use the BlobTree as an underlying data structure for an interactive modelling application is that it can easily and quickly be converted to a number of more conventional data structures for display purposes. For example, [14], [15], [16] propose methods for converting an implicit surface into a triangle-mesh that has an optimised, non-uniform triangle distribution. Other approaches allow for voxelization [17] of the generated surface, or to evaluate the BlobTree with common ray casting approaches, such as interval analysis [18], or ray marching. More recent approaches show how the BlobTree can be used in interactive modelling systems as well [19].

Another reason to use the BlobTree is that it also includes a scene graph and is thus a very compact description of a complex model. Since the data structure takes up considerably less storage than using triangular meshes it could be used for collaborative design across a network.

There have been some attempts to convert models into representations that could potentially be used in the BlobTree. [20] proposed an approach that would convert CSG objects into volume data stored in voxels, which could potentially be used with the BlobTree but it has many shortcomings due to the restricted resolution of the finite space. [21] in comparison added the functionality to combine the BlobTree with mesh surfaces and point sets. This approach does permit hard edged objects but has the added complexity of having to deal with a different representation. In our work we propose a method that stays within the framework of using only implicit primitives and

from which a simple user interface can be built.

C. Skeletal Primitives

Most of the primitives used in the BlobTree are built from skeletal primitives which are incorporated in many implicit modelling software packages such as BlobTree.net [22] or ShapeShop [23]. They are ideally suited to quickly prototype shapes of arbitrary topology [13]. The use of skeletal primitives in the above publications demonstrate that skeletal primitives can lead to a simple and intuitive user modelling methodology.

The basic building block of a skeletal primitive is a skeleton S. Usually the skeleton itself is a very simple shape such as a point or a line, but also more complex skeletons can be used.

To create a skeletal primitive the entire distance-field has to be computed as described in [24]. This is done by computing the distance to the skeleton for each point in the volume encapsulating the final shape. The distance function is defined as

$$D_S : \mathbb{R}^3 \to \mathbb{R}. \tag{1}$$

As a result the distance field is a volume of scalar values which is not bounded as the distance itself can be infinitely large.

In the next step the distance field D_S has to be modified by a field function to be bound to a finite range. This field function is defined as $G : \mathbb{R} \to \mathbb{R}$ and as a result the skeletal primitive $L_{S,G}$ is formed by applying this function to the given distance D_s. Usually it is a function that maps the distances to the range $[0, 1]$.
The implicit function of one skeletal primitive is

$$f_{L_{S,G}}(p) = G(D_S(p)). \tag{2}$$

The most widely used field function was developed by [25] as a simplification of the original Soft Objects field function called the Wyvill field function. It maps a distance d to the field value $G(d)$ by the following formula

$$G(d) = \left(1 - \frac{d^2}{r^2}\right)^3. \tag{3}$$

In this formula r is a constant value that states the distance where the field value equals zero. Figure 2 shows a plot of the function. The main advantage of this field function is that it is C^2 continuous.

After applying the field function to the distance field we call the resulting field the potential field. One further step needs to be done to actually compute the surface of the shape. A classification which part of the finite space is inside and which part is outside has to be done.
This can be seen as an extension to the traditional space partition approach described in [2] because the binary approach is still here, but there is also additional distance information which can be used later on.
By defining an *iso*-value c it is possible to construct the surface of the shape and classify the surrounding space as mentioned above. Usually the iso-value $c = 0.5$ is chosen because it provides surfaces that lie exactly in the middle of the finite space defined by the field values.

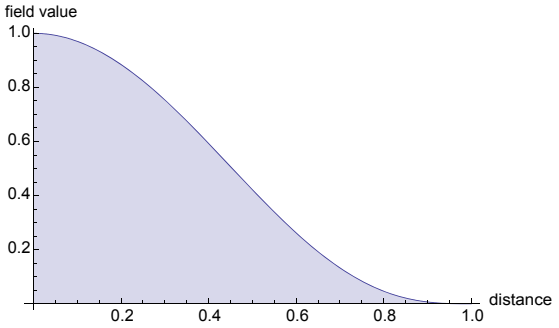


Figure 2: The plot of the commonly used Wyvill field function.

D. Blend Operators

When a Boolean operation is performed on such blobby primitives, it is actually performed on the field-values f. This makes it possible to go beyond the classical Boolean operators, and define general *blend operators* that e.g. create smooth transitions between shapes.

The most common operator that creates a smooth transition between several values is called the *summation blend* [13]

$$f_R(p) = \sum_{n \in N} f_n(p) \tag{4}$$

where the resulting field-value at a point p in space $f_R(p)$ is the sum of the field-values of all involved objects. This example offers only very basic functionality, but it does exhibit one property typical for most blend operators: its results are no longer necessarily in the range $[0, 1]$ of the original potential fields involved in the blend. While seemingly problematic, this does not affect the usefulness of such operators; unlike the original skeleton distance field, they are still bounded (just not to the $[0, 1]$ range), and one can visualize iso-surfaces within them.

More complex operators, such as e.g. those described in [26], [27] or the blending functions that are based on R-functions [28], [6], [29], allow for a fine control on the resulting blend shape. By using them it is possible to create complex blended shapes similar to the ones proposed for CSG in [4]. Something that is also worth noting here is that so far, no-one appears to have used an *interpolation* of the involved field values as a blending operator.

III. Our Approach - Constructive Solid Blobs

Our proposed ***C**onstructive **S**olid **B**lob* approach to solve the filleting problem in CSG makes use of the BlobTree, and the different blending possibilities between blobby objects that exist there. In this section, we propose the idea of separation planes as blending tools in general, explain how this can be used to blend between two potential fields generated by objects, define fillets on CSG objects using this approach and define the extended potential fields for

our new skeletal primitives. Finally we show how these primitives can be combined with the separation plane approach to create interpolations between objects

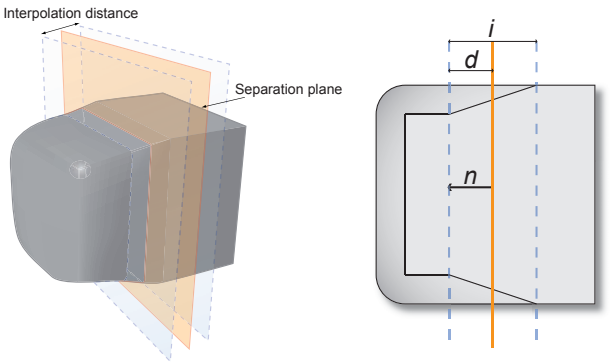
A. Separation Planes as Blending Tools

While any number of schemes to control the blend of two potential fields are possible, we propose the comparatively simple concept of using a *separation plane* for the task. As the name implies, this is a plane which separates the two fields. On one side of such a plane the first field is active, and in the region on the other side of the plane, the other field will be the prominent one as shown in figure 3. The user positions the plane shown in yellow.

A hard separation between the two fields would result in discontinuities at the separation plane, and would not be a particularly useful technique, however, if the new field is derived by interpolating between the two fields in a well-defined area around the separation plane, a smooth transition is created. To bound the volume used in the interpolation and give the user a convenient control, a blend region is computed by using a normal distance from the separation plane specified by the user. This amounts to the implicit definition of two additional planes that limit the blend range, and figure 3 shows a sketch of such an arrangement. The *separation plane*, the tool to control such a blend, is shown in orange. The extent of the *blend range* is shown in blue; here, it is defined by two planes parallel to the separation plane. The skeletons for the two parts of the shape are usually *not* the same size. However, the relative size of the skeletons can be easily precomputed so that the side faces of the two versions of the object are exactly aligned, and that only the edges change from hard to blobby. While all sorts of interpolation modalities can be used, we found that symmetrical interpolation around the separation plane yields useful and good-looking results. In such a configuration, the separation plane contains a cross-section that is an exact average of the two forms as can be seen in figure 3b. Here i refers to the full interpolation region, d is the user set interpolation distance and n refers to the normal vector of the set plane. The way that the interpolation is done to avoid undesired surface behavior such as discontinuities is described in great detail in section III-D. All the user has to do to control the interpolation is to set the plane and the distance d in order to achieve the desired result. Since planes are infinite regions it might occur that the user wants to limit the range of one separation plane. This can be done by using a Boolean combination of several planes to gain the desired result.

B. Blending Between Different Potential Fields

Within a standard BlobTree, it is already common practice to perform blending operations between the potential fields of different skeletons; this allows the aggregation of complex compound objects. However, even if hard-edged primitives are made available in a BlobTree (e.g. via the extended potential fields discussed in section III-C), a basic problem remains: on blending rounded, blobby objects



(a) Separation plane in 3D (b) Separation plane in 2D

Figure 3: A hybrid soft/hard edged cube.

fillets can be specified between adjoining objects almost for free, but unfortunately with their rounded corners, do not necessarily resemble machine parts. Conversely, the much more useful hard-edged objects proposed below do not blend nicely, not even when they are actually BlobTree primitives. Neither do we want generalised blending between them in the first place; we want fillets exactly in those locations where we place them, and nowhere else.

The objectives are:

- To create fillets *between different objects* .
- Additionally we want smooth transitions between a hard edged and a soft edge for a single object.

The method presented in [21] overcomes this problem by mixing the field-value based BlobTree with meshes and point clouds. We have taken the view that a unified modelling approach based on the BlobTree leads to a resolution independent representation and pushes a polygonised approximation down the production pipeline until the end.

In the following sections we describe how the idea of the separating planes can achieve both of these objectives and provide the user with an easy to use and accurate way of building a fillet. By achieving the first objective the second case can be done by co-locating both a hard edged and soft edged version of the same object and interpolating.

C. Defining Extended Potential Fields

As mentioned earlier, it is possible to construct a potential field around any hard edged object contour. A 2D example of this can be seen in figure 4. However, in its original form, the BlobTree lacks objects with sharp edges, such as cubes, cylinders and cones. BlobTree versions of these objects always have rounded corners, even though the skeletons used for them have hard edges. It has to be mentioned that it is possible to create e.g. a hard edged cube by subtracting 6 blobby cubes in the right order from a sphere, but this involves 7 objects and 6 Boolean operations instead of creating it directly with one primitive. A similar approach was proposed by [30] where 6 planes are blended together to create a cube with

almost hard edges. For various reasons, which probably include the lack of a clear incentive to do so up to now, the incorporation of hard-edged objects as genuine BlobTree primitives does not seem to have been attempted so far.

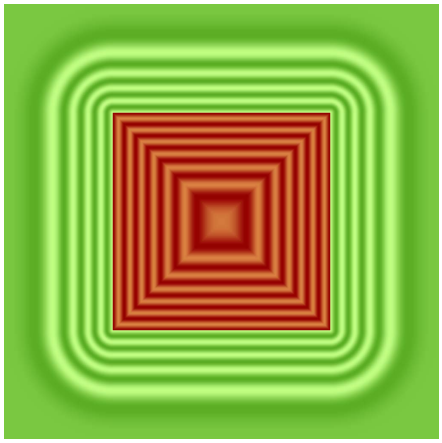


Figure 4: A cut through the extended potential field for a cube skeleton. The difference to normal BlobTree potential fields is that the inside of the skeleton also has field values < 1. For visualization purposes, the function is color-coded (green outside the skeleton, red inside), and modulated with a wave function to show its structure. One can clearly see that for outer iso-values, the resulting shape becomes more and more rounded, the further one gets from the basic shape. In the interior field, on the other hand, iso-lines define shapes with creases. Image created with BlobTree.net [22].

Theoretically it would be possible to use an iso-value of 1 to render those primitives directly from an unmodified BlobTree representation. However, the entire inside of the skeleton is mapped to 1 in classical BlobTree usage, though, so that algorithms for the detection of iso-surfaces run into problems in this case. Which is why we use *extended potential fields* that also contain valid field-values *inside* the skeleton to obtain hard-edged shapes; see figure 4 for an example. The shapes defined by iso-values in such an extended field usually contain creases for values inside the skeleton (the red part of the figure), which stands in contrast to the smooth outside field (shown in green).

In principle, this idea can be applied to any hard-edged CSG primitive, but for our initial work, we concentrated on what arguably are the three most useful basic CSG primitives with edges: the cube, the cylinder and the cone. In the following paragraphs, we describe how to define such extended potential fields for these shapes; additional primitives can be included in an analogous fashion.

To generate the potential field for an object, first its distance field has to be created from the corresponding blobby primitive with the skeleton at the same size as the desired object. Equation 6 shows this for the case of the cube. The desired surface is now at the distance field value zero, positive outside and zero inside. To make this compatible with the BlobTree the next step is to add the distance at which equation 3 yields the iso-value c. We call this the iso-distance D_{iso}, i.e.

$$c = G(D_{iso}) = (1 - \frac{D_{iso}^2}{r^2})^2 \tag{5}$$

Unfortunately there is a problem when this is done without modifying the distance field beforehand. If each value of the current distance field is incremented by the iso-distance all points inside the skeleton have value c, which would be a problem for further blending operations. The desired field is shown in figure 4. In order to fix those wrong values all distances within the skeleton have to be replaced by values computed in a different way than originally as shown e.g. in equation 7. The upcoming formulas assume a coordinate system that is Z-up, $\uparrow$ refers to the maximum and $\downarrow$ the minimum of the given values.

Cube. The distance function for a cube skeleton S centered at q with three perpendicular arms a_x, a_y and a_z describing the orientation of the three axes of the cube is:

$$\begin{aligned} D_S(p) &= \sqrt{\uparrow(0,v_x)^2 + \uparrow(0,v_y)^2 + \uparrow(0,v_z)^2} \\ v_x &= d_x - \|a_x\| \\ v_y &= d_y - \|a_y\| \\ v_z &= d_z - \|a_z\| \\ d_x &= \frac{a_x(p-q)}{\|a_x\|} \\ d_y &= \frac{a_y(p-q)}{\|a_y\|} \\ d_z &= \frac{a_z(p-q)}{\|a_z\|} \end{aligned} \tag{6}$$

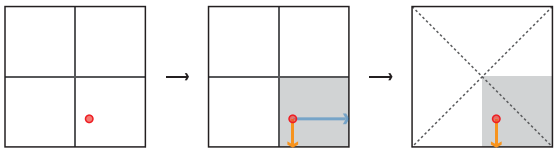
where d_x, d_y and d_z are the distances between q and the projection of p onto a_x, a_y and a_z. They are normalised to the length of a_x, a_y and a_z. To compute the proper distance for the cube with hard-edges the result of this distance computation has to be used in the following formulas :

$$D_{final} = \begin{cases} D_S(p) - \downarrow(v_x, v_z, v_z) + D_{iso} & if\ con \\ D_S(p) + D_{iso} \end{cases} \tag{7}$$

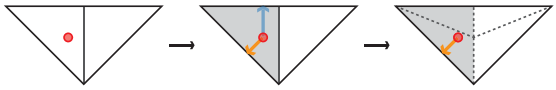
$$con = D_S(p) = 0 \wedge d_x \neq D_{iso} \wedge d_y \neq D_{iso} \wedge d_z \neq D_{iso}$$

A diagram of this computation is shown in figure 5a. The different branches in the computation of D_{final} make sure, that we just modify the inside of the cube skeleton before subtracting the iso-distance D_{iso}. In the computation for $D_S(p)$ this area results in distance values of zero, since they are within the skeleton boundary. The resulting D_{final} is not continuous within the skeleton but it is not intended to be used for blending purposes. Only the blobby version should be used for that.

Cylinder. The approach for computing the distance within the cylinder skeleton is similar to the approach used in computing the distance for the cube. The distance



(a) Cube and cylinder distance computation



(b) Cone distance computation

Figure 5: Illustration of the algorithm to compute the distance for the inside region of either the cube or the cylinder (figure 5a) and the cone (figure 5b) in 2D. For the current point (red circle) first determine in which symmetry area the current point lies (see grey area in the middle image). The shorter of the two distances (blue and orange lines) is chosen (orange). The dashed lines in the right image show where discontinuities in the potential field are created by this algorithm.

function for a cylinder skeleton S with radius r, centred at q having the normal n and height h is:

$$\begin{aligned} D_S(p) &= \sqrt{a^2+b^2} \\ a &= \uparrow(0,c) \\ b &= \begin{cases} \uparrow(0,e) & if\ d>0 \\ \downarrow(0,e) & if\ d\le 0 \end{cases} \\ c &= \|p-q\| - e - r \\ d &= \begin{cases} e-\frac{n}{2} & if\ e>0 \\ e+\frac{n}{2} & if\ e\le 0 \end{cases} \\ e &= (p-q)n \end{aligned}$$

To compute the proper distance for the cylinder with hard-edges the result of this distance computation has to be used in the following formulas:

$$D_{final} = \begin{cases} D_S(p) - \downarrow(0,d) + D_{iso} & if\ con \\ D_S(p) + D_{iso} \end{cases}$$

$$\begin{aligned} con = {} & D_S(p) = 0 \wedge \|(p-q)_x\| \neq D_{iso} \wedge \\ & \|(p-q)_y\| \neq D_{iso} \wedge \|(p-q)_z\| \neq D_{iso} \wedge \\ & \sqrt{\|(p-q)_x\|^2 + \|(p-q)_y\|^2} \neq D_{iso} \end{aligned}$$

The basic principle is similar to the one shown in figure 5a. Similar to the computation of D_{final} for the cube here the branch makes sure that p lies within the skeleton. We do this in the last part of $cond$ by comparing the distance of the point projected onto the XY plane with the iso-distance D_{iso} the additional part in the if statement makes sure that the current point lies within the circle of the cylinder base when projected onto the XY plane.

Cone. For the cone skeleton the distance computation is a little bit more complicated (see figure 5b). The distance function for a cone skeleton S with the tip at point q, having normal n, height h and radius r at the base is:

$$\begin{aligned} D_S(p) &= \begin{cases} \|p_1 - s\| & if\ s > \|t\| \\ \|p_1 - s\ \ t\| & if\ \|t\| > 0 \\ \|p_1\| & if\ s \le 0 \end{cases} \\ s &= \frac{(p_1)t}{\|t\|^2} \\ t &= h\ \ n + \frac{r\ \ u}{\|u\|} \\ u &= p_1 - n((p_1)n) \\ p_1 &= (p-q) \end{aligned}$$

To compute the proper distance for the cone with hard-edges the result of this distance computation has to be used in the upcoming formulas. The cone is projected onto the XZ plane forming a triangle with inradius w.

$$D_{final} = \begin{cases} D_{new} + D_{iso} & if\ con \\ D_S(p) + D_{iso} \end{cases}$$

$$con = 0 < p_{1z} < 1 \wedge \sqrt{p_{1x}^2 + p_{1y}^2} < p_{1x}\ \ p_{1z}$$

$$D_{new} = \begin{cases} -(1-p_{2y})/w\ \ D_{iso} & if\ p_{2y} \ge a \\ -(\|p_2 - c\|)/w\ \ D_{iso} \end{cases}$$

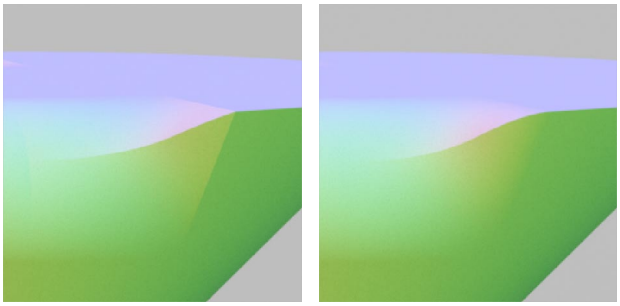
$$\begin{aligned} p_2 &= \begin{pmatrix} \sqrt{p_{1x}^2 + p_{1y}^2} \\ p_{1z} \end{pmatrix} \in \mathbb{R}^2 \\ a &= w\ \ p_{2x} - (1-w) \\ b &= (p_{2x} + p_{2y})/2 \\ c &= \begin{pmatrix} b \\ b \end{pmatrix} \in \mathbb{R}^2 \end{aligned}$$

By multiplying the result of D_{new} by the iso-distance D_{iso} it is guaranteed that there will not be a resulting distance $D_{final} < 0$. Note that p_2 and v are two dimensional vectors, since the computation can be reduced to two dimensions because of the rotational symmetry.

Other primitives. All of the above computations can be described as the medial axis transform of an object as first described in [31]. In our application we use the medial axis transform to compute the "skeleton of an implicit skeleton" that is then used to modify the original distance computation. The extended potential field of an object can be computed from the the medial axis transform of the already known implicit skeletons. Most of the algorithms for computing the medial axis transform have been described for discrete objects. Algorithms for computing continuous medial axes can be found in [32], [33], [34].

D. Shape Interpolation Within the Blend Range

With the separation plane technique described above we are able to apply blending operations to different potential fields of the same basic object, placed at the same location, i.e. we are able to switch between varying intensities of blobbiness in one and the same shape. Figure 3 shows one possible result of such an operation. The key to this is to simply interpolate between the field values of the



(a) linear interpolation (b) Hermite interpolation

Figure 6: Comparison between a simple linear interpolation operator, and an interpolation based on cubical Hermite spline base functions. This image is a close-up of the top of the blended cone shape shown in figure 10, with a special BRDF that changes color depending on the surface normal. In case (a), a discontinuity on the outer mantle, which starts at the point where the normal hard edge of the cone top starts to slope downward, and which runs down the front side of the object, is clearly visible. In case (b), Hermite interpolation prevents such an unwanted crease from appearing.

involved representations. From a modelling point of view, all that has to be done is to define a separation plane and an interpolation distance.

As noted in section II-B, this is a blending operator that is not useful for blending between *different* shapes, and is therefore not used in normal BlobTree modelling.

The easiest way to define the interpolation blending operator we use in the region around the separation plane is in a linear fashion, e.g. by multiplication with a factor j, as given in equation (8). There, d is the interpolation distance of the separation plane, and $d(p)$ the distance between the separation plane and the current point.

$$j \quad = \quad \left(0.5\frac{d(p)}{d}\right) \; + 0.5 \tag{8}$$

The main drawback of this simple operator is that it causes visible surface discontinuities at the borders of the blend range. This can be clearly seen in figure 6a, where an unwanted crease in the cone mantle along the limits of the blend range is clearly evident.

Since one usually wants a blended object with more than just C^0 continuity at the borders of the blend zone, one has to use some sort of higher order interpolation instead. We opted for the third of the four base functions of cubic Hermite splines [35]. By using the function defined in equation (9), we weight the factor j used in the linear interpolation to create a new interpolation factor h.

$$h(j) \quad = \quad j^2\left(3 - 2j\right. \tag{9}$$

This yields an interpolated surface which is C^2 continuous and smooth in the whole blend area, and with the adjoining surfaces of the two representations.

Figure 6 shows the difference between the linear and the spline basis function based interpolation. In most cases the Hermite interpolation is preferable. All the subsequent examples are rendered using the Hermite based interpolation. It has to be noted that the Hermite interpolation involves more floating point operations than the linear interpolation.

Another possibility to blend between the two representations would be to use the potential field of an altogether different blobby object to define the $[0, 1]$ interpolation range, similar to a blending approach described in [7] and integrated into the BlobTree in [36]. Our approach could also be used to create entire blends, as described in [12]. All these could also be used together with the BlobTree, and would yield interesting results in their own right.

E. Defining Fillets With Separation Planes

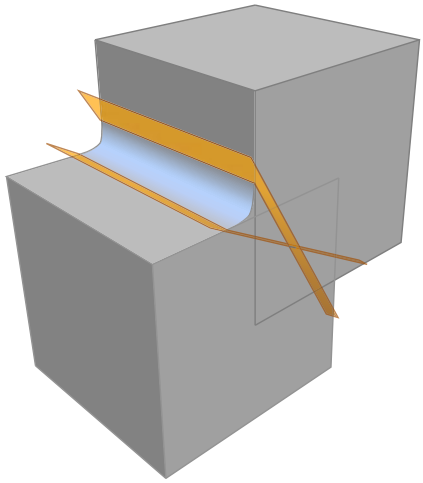


Figure 7: Use of separation planes to define a fillet region between hard-edged shapes. The shape of the fillet is dependent on the chosen blend operator. Only the interpolation region is defined by using the planes.

Hybrid shapes like the mixtures between blobby and hard-edged versions of geometric primitives shown in figure 10 are a nice extension of the standard BlobTree. Figure 7 shows an example where two cubes are blended together with our approach. Each cube has it's own separation plane facing the other cube. As a result the blobby representations which blend well are restricted to the area between the planes. For blending the objects we now just use the blobby representation.

By setting the separation plane manually the user can decide where and how the transition from the hard-edged representation to the round-edged blobby one takes place. This transition is independent of the chosen blend operator since it is just a way to define the region where the user wants to have the blobby part of the object that has the capability to blend properly as described above. This allows for a high number of possible blends between two objects since there are many different ways that the two objects can be placed, which blend operator can be chosen and how the user restricts the object representations using one or more separation planes.

To create a blend as shown above, it is actually necessary to define a new blend operator which allows for interpolation between two different blend operators, for example by reusing our separation plane approach. Otherwise a simple blend would create bulges on the sides of the cubes. This image shows the blend just happening at the above junction of the two shapes whereas on the opposite side there is the normal CSG-union transition. This is due to the fact that the normal vectors of the planes do not point into the area enclosed by the planes at that connection. It is possible to extend our approach to create a smooth junction at the opposite side as well by e.g. defining that when two planes cross, the normal vectors always are oriented towards the other plane.

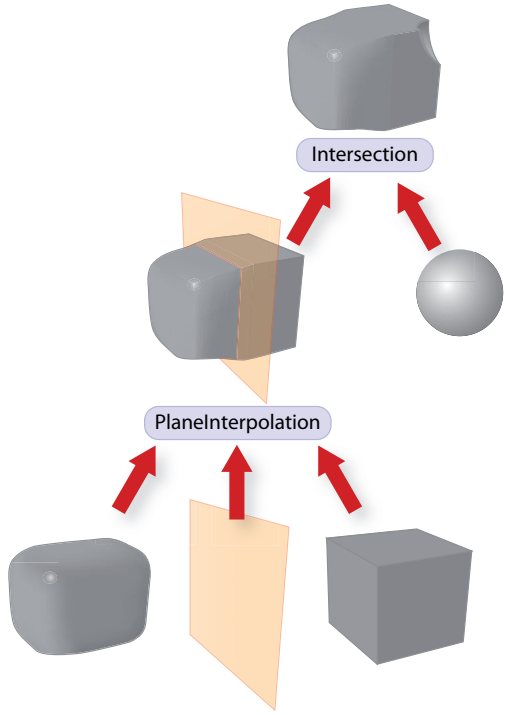


Figure 8: The possible structure of a new BlobTree node incorporating the described separation plane approach.

Figure 8 shows how the described approach can be implemented as a new BlobTree node. Since there are not just two shapes involved but also an additional separation plane the most convenient way will be a ternary node that connects those three components. When the field-value for this node is computed it evaluates the field-values of the two child-shapes and interpolates according to the algorithm described above.

The modelling workflow to set up the interpolation is very similar to any other blend node: Two shapes have to be positioned in the scene ideally at the same position having the same size. After that the user has to place the separation plane and set the normal into the direction where the interpolation should take place and adjust the interpolation distance to get the desired result. Figure 11 shows an example where the separation plane is used to smooth just one edge of the cube. The other edges stay

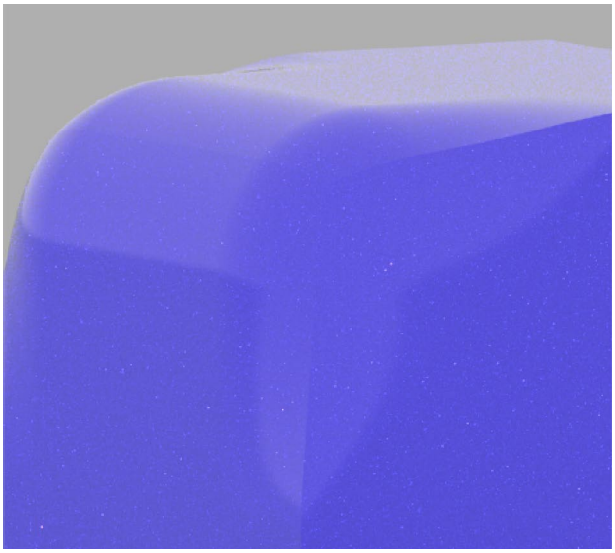


Figure 9: One single edge smoothed by placing a separation plane so that it cuts through the cube so that the normal vector faces the edge.

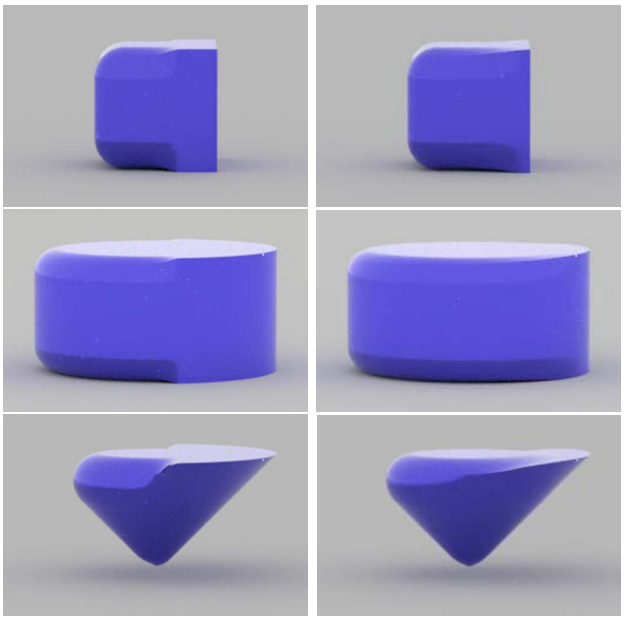
hard edged until they come into the interpolation region. By adjusting the interpolation distance it is moreover possible to get a slight edge in the smooth part as well depending on the set distance defined as d in equation 8.

With our approach a fillet can be defined in a few simple steps. First, the separation planes have to be defined; they specify where the shapes should have hard edges and where they should have blobby parts. The latter describes the filleting region of the shape. Afterwards one has to assign the desired blend operator to the two shapes and make sure that just the blobby part of the shape is used for the fillet. This can be done e.g. by using blend operators as defined in [27] that give the user great control over the shape of the resulting blend between the objects.

Furthermore the modelling paradigm introduced with the separation planes can be combined with a general blend node as well. Since the main objective of a separation plane is to provide a volume that defines interpolation values it is not just possible to interpolate between the field values of objects but it also allows for an interpolation between the resulting field values of two different blend operators. This can be used to produce similar modelling results as achieved using a bounding implicit object [7] as can be seen in figure 11 at the upper right edge that smoothly interpolates into the rounded edge. A proper user study to compare the use of the separation plane to a bounding object and associated parameters awaits future work.

IV. Results

The approach we presented in this paper exploits the advantages of both the blobby representation and traditional CSG. However, our approach also allows a smooth interpolation between blobby and hard edged representations within an object by simply defining a separation plane.



(a) (b)

Figure 10: Interpolation between traditional CSG and blobby representations for a cube, a cylinder and a cone. In all examples, the separation plane lies in the (y, z)-plane. In the left row the interpolation is done only within 0.2 distance from the plane, in the right row the interpolation is done on the whole size of the shape. The issue with the edge not appearing to be straight with the cone is the result of the combination of the used shading model and the surface created.

Figure 10 shows a change in the representation on three basic primitives, a cube, a cylinder and a cone. Here the separation plane is set exactly at the z-axis facing left so that the left part of the primitive is represented as blobby object and the right part as traditional CSG primitive. As it can be seen, a smooth interpolation is performed between both objects. The hard edges of the CSG objects gradually become rounder until no hard edges can be seen anymore. Moreover, by changing the interpolation distance we are able the change the appearance of the object in an additional way.

One advantage of this approach is, that the implicit field at the rounded edge is fully intact and as a result it can be used in any further blending situation as shown in figure 11.

Such modelling techniques can be used to create realistic complex objects that are very hard or even impossible to model with traditional CSG or the blend and fillet operators since complex transitions are involved. Figure 12 (respectively 1c) shows an example of such an object. The water tap in this image consists of all representations described in this paper:

1) blobby objects (e.g. the cylinder at the base part of the handle),
2) traditional CSG objects (e.g. the cube that connects the two water inlets),

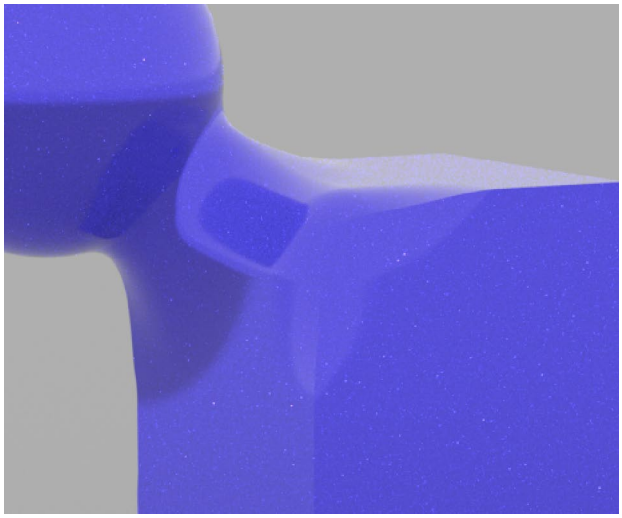


Figure 11: Any object can be blended with the rounded edge without problems that are the result of field discontinuities.

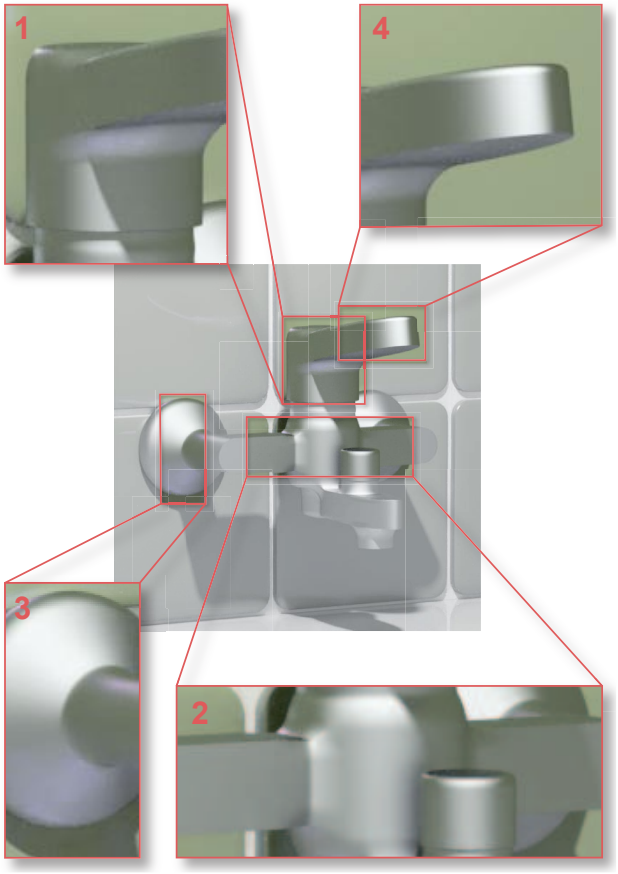


Figure 12: Several close-ups of the water tap CSB modelshown in figure 1 (c). Note the presence of both soft and hard edges, and the smooth interpolation between both representations.

3) filleting between blobby and CSG objects (e.g. the inlets) and

4) interpolation between classic CSG and blobby objects (e.g. the cylinder at the handle).

With traditional CSG it would be very time-consuming to compute the smooth transitions of the inlets and the round edges of certain primitives like e.g. the main part of the tap. On the other side, the hard edge of the cross piece cannot be created with blobby objects alone. The rendering time of the CSB model was roughly equal to the rendering time of the blobby object. In theory some parts of the model could be constructed with the original BlobTree as well by creating the hard edged objects with additional Boolean operations, but this increase in complexity comes with the cost of higher rendering times since the number of BlobTree evaluation increases as well.

This example shows that by using our technique it is possible to achieve the same modelling result as with the BlobTree but using fewer primitives and Boolean operations. On the other hand it is still possible to make use of the mathematically simpler blending operators provided by the BlobTree compared to special purpose filleting operations used in CSG. Since only the underlying math is different from previously known approaches it is still possible to expose the user to standard user interfaces.

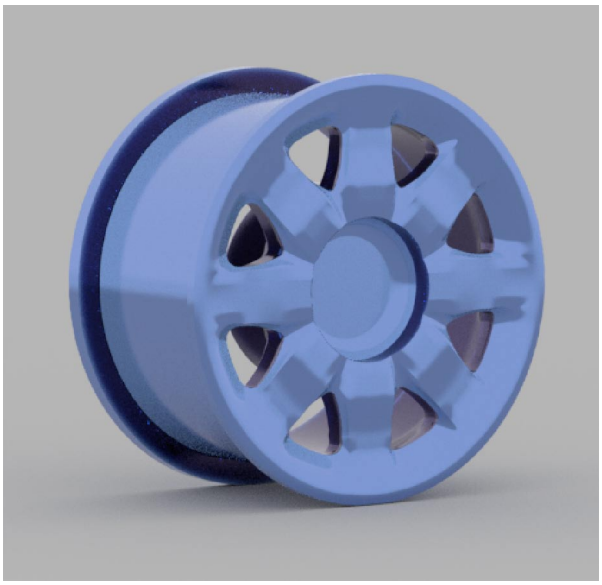
One of the drawbacks of the filleting methods discussed in section II-A is that they run into troubles when blends with blends are involved. As it can be seen in figure 13a where the spokes have smooth transitions between themselves and the outer part of the rim, our approach is capable of performing such blends since creating these blends is just a matter of applying the desired operator; no additional work has to be done.

Of course one could argue that such models could be created easily using a mesh or subdivision surface approach. However, figure 13b and 13c clearly illustrates the advantage of our approach compared to any mesh approach. By using CSG or blobby objects we are able to resize and move certain parts of a model without remodelling the whole object. For example, we resized the thickness of the spokes by simply widening the cubes the spokes are made of. The fillets are created automatically – a mesh approach would require to remodel large parts of the whole wheel.

The models created for this paper were produced from a scripting system. Interactive sketch based use of the BlobTree has already been demonstrated [23] and our approach does not increase the time complexity for BlobTree traversal. We are currently exploring a sketch based interface for the BlobTree using our presented approach.

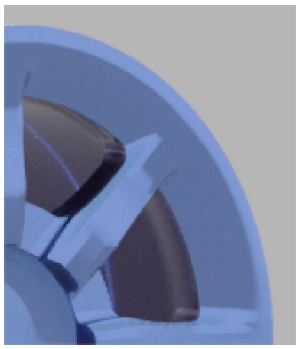
V. Conclusions and Future Work

Modelling with just CSG or the BlobTree has certain disadvantages which are *the lack of rounded edges and fillets* in standard CSG and the *lack of hard edges* in standard BlobTree modelling. As the modelling approach is similar in both methods the work done in this paper shows how the features of both can be combined into one approach, namely **C**onstructive **S**olid **B**lobs (CSB).



(a)

(b) Thick spoke



(c) Thin spoke

Figure 13: The spokes in this model of a car wheel have sharp corners, but also smooth transitions between themselves and the outer part of the rim. It can be changed easily by resizing certain primitives since the fillets are generated automatically.

CSB is entirely based on the modelling approach of the BlobTree and extends it to support the shapes that are different to the standard CSG approach. The shapes that differ in their representations are extended to create accurate field values which are currently not continuous in the inside of the objects. It can be seen that CSB does not completely change the modelling process of CSG or the BlobTree as it is just an extension to the BlobTree. The problem of creating blends or so called fillets is overcome by using CSB instead of CSG as round edges and blends are just a matter of summations which are easier to compute than the higher order equations described in [3] or the approaches described in [8] or [5]. Moreover CSB enables to define the shape of the transition by the implemented blend operators.

One major enhancement of CSB would be to create a continuous potential field for the extended CSG shapes.

1: **if** $(D_S(p) == 0) \wedge (\|d_x\| \neq D_{iso}) \wedge (\|d_y\| \neq D_{iso}) \wedge (\|d_z\| \neq D_{iso})$ **then**
2: $D_{final} \leftarrow D_S(p) - \downarrow (\uparrow (0, d_x - \|a_x\|)^2, \uparrow (0, d_y - \|a_y\|)^2, \uparrow (0, d_z - \|a_z\|)^2) + D_{iso}$
3: **else**
4: $D_{final} \leftarrow D_S(p) + D_{iso}$
5: **end if**

Figure 14: distance computation within the cube skeleton

1: **if** $(D_S(p) == 0) \wedge (\|p_x\| \neq D_{iso}) \wedge (\|p_y\| \neq D_{iso}) \wedge (\|p_z\| \neq D_{iso}) \wedge (\sqrt{p_x^2 + p_y^2} \neq D_{iso})$ **then**
2: $D_{final} \leftarrow D_S(p) - \downarrow (\sqrt{\|p-q\| - ((p-q)\ n)^2} - r, \|(p - q)\ n\| + \frac{h}{2}) + D_{iso}$
3: **else**
4: $D_{final} \leftarrow D_S(p) + D_{iso}$
5: **end if**

Figure 15: distance computation within the cylinder skeleton

This would enable better blending capabilities of these objects. As a result it might be possible to use these shapes even without the presented separation plane approach in a blending situation. Another issue with the current implementation is that the round corners of an object have the same dimensions on the whole shape.

In some modelling situations it might be desired to have different rounded corners with different radii at the same object or even corners where the radius changes along the edge. Something similar was already presented by [7] with the *partial edge blending* approach or the rolling ball approach of [12]. If this approach can be combined with defining individual edge radii and a continuous field this would lead to easier application of the introduced approach combined with a higher flexibility.

Another interesting topic for research will be to give the user the opportunity to have the positions of the separation planes computed automatically according to a chosen blend operator and a certain placement of the involved shapes.

ACKNOWLEDGMENT

This research is made possible in part by a grant from the Natural Sciences and Engineering Council of Canada and the Vienna University of Technology.

APPENDIX

The algorithms for the extended distance computation have the basic distance $D_S(p)$ as an input and compute the modified distance D_{final} are shown in figures 14, 15 and 16. The distance values within the skeletons are in the range $[0, D_{iso}]$. As in the formulas presented throughout this paper $\uparrow$ refers to the maximum and $\downarrow$ to the minimum value of the given ones.

1: **if** $(D_S(p) == 0) \wedge (0 < p_{1z} < 1) \wedge (\sqrt{p_{1x}^2 + p_{1y}^2} < p_{1z}\ \ radius)$ **then**
2: $point_{2D} \leftarrow (\sqrt{p_{1x}^2 + p_{1y}^2}, p_{1z})$
3: $reference_y \leftarrow inradius\ \ point_x + (1 - inradius)$
4: **if** $point_y \geq reference_y$ **then**
5: $D_{new} \leftarrow -\frac{1-point_y}{inradius \cdot D_{iso}}$
6: **else**
7: $u \leftarrow \frac{point_x + point_y}{2}$
8: $footPos \leftarrow (u, u)$
9: $D_{new} \leftarrow -\|point - footPos\|$
10: **end if**
11: $D_{final} \leftarrow D_{new} + D_{iso}$
12: **else**
13: $D_{final} \leftarrow D_S(p) + D_{iso}$
14: **end if**

Figure 16: distance computation within the cone skeleton

REFERENCES

[1] B. Wyvill, A. Guy, and E. Galin, "Extending the CSG tree. warping, blending and boolean operations in an implicit surface modeling system," *Computer Graphics Forum*, vol. 18, no. 2, pp. 149–158, Jan 1999.

[2] A. Ricci, "A constructive geometry for computer graphics," *The Computer Journal*, vol. 16, no. 2, pp. 157–160, 1973.

[3] A. Middleditch and K. Sears, "Blend surfaces for set theoretic volume modelling systems," *SIGGRAPH 85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, Jul 1985.

[4] G. Elber, "Generalized filleting and blending operations toward functional and decorative applications," *Graphical Models*, vol. 67, no. 3, pp. 189–203, Dec 2005.

[5] V. Adzhiev, R. Cartwright, E. Fausett, A. Ossipov, A. Pasko, and V. Savchenko, "HyperFun project: a framework for collaborative multidimensional F-rep modeling," *Proceedings of Implicit Surfaces 99*, pp. 59–69, Sept. 1999.

[6] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko, "Function Representation in Geometric Modeling: Concepts, Implementation and Applications," *The Visual Computer*, vol. 11, no. 8, pp. 429–446, Oct 1995.

[7] G. Pasko, A. Pasko, M. Ikeda, and T. Kunii, "Bounded Blending Operations," *Proc. of Shape Modeling International 2002*, Dec 2002.

[8] J. Rossignac and A. Requicha, "Constant-Radius Blending in Solid Modeling," *ASME Computers In Mechanical Engineering (CIME)*, vol. 3, pp. 65–73, 1984.

[9] C. Hoffmann and J. Hopcroft, "The potential method for blending surfaces and corners," Cornell University, Ithaca, NY, USA, Tech. Rep., 1985.

[10] M. Peternell and H. Pottmann, "Computing rational parametrizations of canal surfaces," *Journal of Symbolic Computation*, Dec 1997.

[11] X. Chen and C. Hoffmann, "Trimming and closure of constrained surfaces," Purdue University, Tech. Rep., Dec 1993.

[12] B. Whited and J. Rossignac, "Relative blending," *Computer-Aided Design*, vol. 41, no. 6, pp. 456–462, May 2009.

[13] J. Bloomenthal, *Introduction to Implicit Surfaces*. Morgan Kaufmann, ISBN 1-55860-233-X, 1997, edited by Jules Bloomenthal With Chandrajit Bajaj, Jim Blinn, Marie-Paule Cani-Gascuel, Alyn Rockwood, Brian Wyvill, and Geoff Wyvill.

[14] C. Ho, F. Wu, B. Chen, Y. Chuang, and M. Ouhyoung, "Cubical marching squares: Adaptive feature preserving surface extraction from volume data," *Computer Graphics Forum*, vol. 24, no. 3, pp. 537–545, Dec 2005.

[15] K. van Overveld and B. Wyvill, "Shrinkwrap: An efficient adaptive algorithm for triangulating an iso-surface," *Vis. Comput.*, vol. 20, no. 6, pp. 362–379, 2004.

[16] J. Bloomenthal, "Polygonization of Implicit Surfaces," *Computer Aided Geometric Design*, vol. 5, no. 4, pp. 341–355, 1988.

[17] N. Stolte, "Robust voxelization of surfaces," State University of New York at Stony Brook, Tech. Rep., Dec 1997.

[18] J. Snyder, "Interval analysis for computer graphics," *SIGGRAPH 92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, Jul 1992.

[19] R. Schmidt, B. Wyvill, and E. Galin, "Interactive implicit modeling with hierarchical spatial caching," in *SMI 05: Proceedings of the International Conference on Shape Modeling and Applications 2005*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 104–113.

[20] D. E. Breen, S. Mauch, and R. T. Whitaker, "3d scan conversion of CSG models into distance volumes," in *VVS 98: Proceedings of the 1998 IEEE symposium on Volume visualization*. New York, NY, USA: ACM, 1998, pp. 7–14.

[21] R. Allegre, E. Galin, R. Chaine, and S. Akkouche, "The hybridtree: mixing skeletal implicit surfaces, triangle meshes, and point sets in a free-form modeling system," *Graph. Models*, vol. 68, no. 1, pp. 42–64, 2006.

[22] E. de Groot, "Blobtree modelling," Ph.D. dissertation, The University of Calgary, 2008.

[23] R. Schmidt, B. Wyvill, M. Costa-Sousa, and J. A. Jorge, "Shapeshop: Sketch-based solid modeling with the blobtree," in *Proc. 2nd Eurographics Workshop on Sketch-based Interfaces and Modeling*, Eurographics. Eurographics, 2005, pp. 53–62, dublin, Ireland, August 2005.

[24] A. Barbier and E. Galin, "Fast distance computation between a point and cylinders, cones, line-swept spheres and cone-spheres," *Journal of Graphics, GPU, and Game Tools*, vol. 9, no. 2, pp. 11–19, 2004.

[25] B. Wyvill and G. Wyvill, "Field Functions for Implicit Surfaces," *The Visual Computer*, vol. 5, no. 1-2, pp. 75–82, 1989.

[26] L. Barthe, N. A. Dodgson, M. A. Sabin, B. Wyvill, and V. Gaildrat, "Two-dimensional potential fields for advanced implicit modeling operators," *Computer Graphics Forum*, vol. 22, no. 1, pp. 23–34, 2003.

[27] L. Barthe, B. Wyvill, and E. de Groot, "Controllable binary CSG operators for soft objects," *International Journal of Shape Modeling*, Dec 2004.

[28] V. Shapiro, "Real Functions for Representation of Rigid Solids," *Computer-Aided Geometric Design*, vol. 11, no. 2, 1994.

[29] A. A. Pasko and V. V. Savchenko, "Blending Operations for the Functionally Based Constructive Geometry," *CSG 94 Set-Theoretic Solid Modeling: Techniques and Applications, Information Geometers*, pp. 151–161, Dec 1998.

[30] L. Barthe, V. Gaildrat, and R. Caubet, "Combining implicit surfaces with soft blending in a CSG tree," *CSG Conference Series*, pp. 17–31, Apr 1998.

[31] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed. Cambridge: MIT Press, 1967, pp. 362–380.

[32] L. Cao, Z. Jia, and J. Liu, "Computation of medial axis and offset curves of curved boundaries in planar domains based on the cesaro's approach," *Computer Aided Geometric Design*, vol. 26, no. 4, pp. 444 – 454, 2009, geometric Modeling and Processing 2008, 5th International Conference on Geometric Modeling and Processing.

[33] E. Remy and E. Thiel, "Exact medial axis with euclidean distance," *Image and Vision Computing*, vol. 23, no. 2, pp. 167 – 175, 2005, discrete Geometry for Computer Imagery.

[34] J.-H. Chuang, C.-H. Tsai, and M.-C. Ko, "Skeletonization of Three-Dimensional Object Using Generalized Potential Field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1241–1251, 2000.

[35] D. D. Hearn and M. P. Baker, *Computer Graphics with OpenGL*. Prentice Hall Professional Technical Reference, 2003.

[36] C. Galbraith, L. Mundermann, and B. Wyvill, "Implicit Visualization and Inverse Modeling of Growing Trees ," *Computer Graphics Forum*, vol. 23, no. 3, pp. 337–348, 2004.