Boolean Operations with Implicit and Parametric Representation of Primitives Using R-Functions

Article *in* IEEE Transactions on Visualization and Computer Graphics · October 2005

DOI: 10.1109/TVCG.2005.72 · Source: PubMed

5 authors, including:

Yohan D Fougerolle
University of Burgundy
67 PUBLICATIONS **380** CITATIONS
SEE PROFILE

Andrei Gribok
University of Tennessee
105 PUBLICATIONS **1,321** CITATIONS
SEE PROFILE

Frederic Truchetet
University of Burgundy
233 PUBLICATIONS **1,814** CITATIONS
SEE PROFILE

Mongi Abidi
University of Tennessee
252 PUBLICATIONS **4,708** CITATIONS
SEE PROFILE

Boolean Operations with Implicit and Parametric Representation of Primitives Using R-Functions

Yohan D. Fougerolle, Andrei Gribok, Sebti Foufou,
Frédéric Truchetet, *Member*, *IEEE*, and Mongi A. Abidi, *Member*, *IEEE*

Abstract—We present a new and efficient algorithm to accurately polygonize an implicit surface generated by multiple Boolean operations with globally deformed primitives. Our algorithm is special in the sense that it can be applied to objects with both an implicit and a parametric representation, such as superquadrics, supershapes, and Dupin cyclides. The input is a Constructive Solid Geometry tree (CSG tree) that contains the Boolean operations, the parameters of the primitives, and the global deformations. At each node of the CSG tree, the implicit formulations of the subtrees are used to quickly determine the parts to be transmitted to the parent node, while the primitives' parametric definition are used to refine an intermediary mesh around the intersection curves. The output is both an implicit equation and a mesh representing its solution. For the resulting object, an implicit equation with guaranteed differential properties is obtained by simple combinations of the primitives' implicit equations using R-functions. Depending on the chosen R-function, this equation is continuous and can be differentiable everywhere. The primitives' parametric representations are used to directly polygonize the resulting surface by generating vertices that belong exactly to the zero-set of the resulting implicit equation. The proposed approach has many potential applications, ranging from mechanical engineering to shape recognition and data compression. Examples of complex objects are presented and commented on to show the potential of our approach for shape modeling.

Index Terms—Computational geometry and object modeling, constructive solid geometry, object representation, volume visualization, R-functions, superquadrics, supershapes, Dupin cyclides.

1 Introduction

Solid modeling deals with the representation, analysis, and combination of various shapes to increase the ease of use and the accuracy of algorithms needed for the representation of complex objects. Developments in these representations have become necessary for technological improvements and are widely used in many applications, including the automotive and airplane industries, architecture, medicine, and even the advertising sector. In solid modeling, two complementary approaches can be considered, the first dealing with the primitives and their properties and the second studying the methods to efficiently combine them. This paper addresses both approaches by considering primitives with both parametric and implicit representations and their combination through Boolean operations using R-functions. The contributions of this paper are:

- An extension of the literature in solid modeling by the combination of recent and versatile primitives such as supershapes using R-functions.
- An algorithm to polygonize an implicit surface generated by multiple Boolean operations with globally deformed primitives characterized by its low time complexity and its accuracy.

1.1 Related Work

Two methods for surface representation can be distinguished, namely, implicit and parametric representations. The most widely used parametric surfaces are based on Bernstein polynomials and B-spline basis functions [1]. Parametric surfaces also include subdivision surfaces in which an initial coarse mesh is refined to converge to a limit surface that can be represented by a set of eigenbasis functions everywhere [2]. One of the most useful characteristics of the parametric representations is the direct computation of vertices, normals, and higher order derivatives. Due to the useful geometrical relation between the surface and its control mesh, shape design using parametric surfaces is easy, natural, and has become a quasistandard modeling set in CAD systems. Unfortunately, parametric representations lack a characteristic function to evaluate if a point is inside or outside an object. Boolean operations on NURBS [3] or subdivision surfaces [4] are therefore slower and more complicated to perform due to "face-to-face" intersection algorithms and additional preprocessing steps. Well-suited for solid modeling, the implicit representation uses a single-valued function of multiple variables, where the surface is the locus of points for which the function is constant. There exist

- *Y.D. Fougerolle, A. Gribok, and M.A. Abidi are with the University of Tennessee, Knoxville, TN 37996-2100. E-mail: {yfougero, agribok, abidi}@utk.edu.*
- *S. Foufou is with the Laboratoire Le2i, UMR CNRS 5158, UFR Sciences et Techniques, Université de Bourgogne, BP 47870, 21078 Dijon Cedex, France. E-mail: Sebti.Foufou@u-bourgogne.fr.*
- *Y.D. Fougerolle and F. Truchetet is with IUT Le Creusot, Laboratoire Le2i, UMR CNRS 5158, 12, rue de la Fonderie, 71200 Le Creusot, France. E-mail: y.fougerolle@u-bourgogne.fr, f.truchetet@ieee.org.*

Manuscript received 16 June 2004; revised 12 Jan. 2005; accepted 24 Jan. 2005; published online 11 July 2005.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0058-0604.

many types of implicit representations, ranging from skeletal-based approaches, distance functions, and algebraic surfaces to partial differential equations. Implicit methods provide simple solutions for Boolean operations and blending [5], [6], [7]. Unfortunately, the resulting surface may be difficult to compute exactly and, most of the time, only an approximation of the surface is available when using algorithms such as Marching cubes [8], [9].

The two major approaches used in solid modeling are constructive solid geometry (CSG) and boundary representation (B-Rep). In CSG, solids are represented as a Boolean expression of solid objects of a simpler structure. Both the surface and the interior of an object are implicitly defined. A B-Rep only describes the oriented surface of a solid as a data structure composed of vertices, edges, and faces. Strengths and weaknesses of both methods have been presented and discussed in [10], [11], [12], [13], [14], [15]. Recently, Rappoport and Spitz proposed a method for interactive display of CSG models using widely available graphics hardware [16]. Wyvill et al. proposed the BlobTree, a structure to combine Boolean operations, deformations, and skeletal implicit surface system (Blinn Objects) [5]. Biermann et al. proposed a method to model free-form solids using subdivision surfaces [4]. Recent methods, known as hybrid methods, try to combine different solid modeling representations or consider one representation applied to different types of primitives. For example, Adzhiev et al. [17] combine volume representation by voxels and real continuous functions, whereas Allegre et al. [18] apply the CSG framework to skeletal implicit functions and polygonal meshes.

1.2 Outline

The rest of the paper is organized as follows: Section 2 is dedicated to R-functions and their properties. Section 3 deals with the primitives considered in this paper. Our algorithm is presented in Section 4. Section 5 presents some experimental results and a discussion about our assumptions and hypothesis. The last section presents our conclusions and future work.

2 R-Functions and Their Properties

The theory of R-functions was developed by Rvachev in the 1960s [19]; a concise tutorial in English has been written by Shapiro [20]. An R-function is a real-valued function characterized by some property that is completely determined by the corresponding property of its arguments (such as sign, membership in $[0, 1]$, etc.). As a special case, some real-valued functions of real variables have the property that their sign is completely determined by the signs of their arguments. For example, the sign of the function $f(x) = xy$ is completely determined by the sign of its arguments, as opposed to the function $f(x) = \sin(xy)$, where not only the sign but also the magnitude of the arguments influence the sign of the function.

We are specifically interested in applications of R-functions theory to the inverse problem of analytical geometry. By solving the problem, we mean that an arbitrary closed shape can be described by a single analytical equation, $f = 0$, or inequality, $f \geq 0$, which would be satisfied only on the borders or in the interior of the object, respectively. Additionally, the resulting function would be differentiable a given number of times. In [21], [22], Rvachev showed that combining Boolean algebra —and, more generally, n-values algebra—with a set of functions which is slightly richer than polynomials, the inverse problem of analytic geometry can be solved in general. To introduce R-functions, let us start with a Venn diagram presented in Fig. 1a.

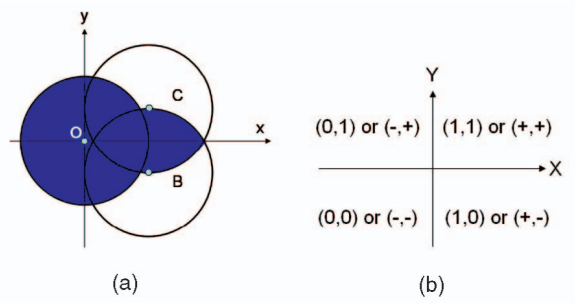


Fig. 1. An example of object and Boolean quadrants. (a) Venn diagram. (b) Boolean quadrants.

The shaded area D can be represented by the following Boolean function performed for each point of disks C_O, C_A, and C_B centered, respectively, at O, A, and B.

$$D = C_O \vee (C_A \wedge C_B). \tag{1}$$

We can introduce the following predicate to map a real variable x into a Boolean variable depending on its sign.

$$S(x) = \frac{1}{2}(1 + sign(x)), x \neq 0. \tag{2}$$

Since we are considering two-valued logic, zero belongs exclusively to one or the other interval. We refer the reader to Section 1.6 in [20] for discussions about the zero case and the potential partitions of the real axis.

Let the inequalities $\omega_O(x, y) \geq 0$, $\omega_A(x, y) \geq 0$, and $\omega_B(x, y) \geq 0$ define the unit disks centered at O, A, and B, respectively. Substituting these inequalities into (1) using predicate (2), we obtain the predicate equation

$$F(S(\omega_O(x, y)), S(\omega_A(x, y)), S(\omega_B(x, y))) = 1, \tag{3a}$$

or, on our example,

$$S(\omega_O(x, y)) \vee (S(\omega_A(x, y)) \wedge S(\omega_B(x, y))) = 1, \tag{3b}$$

where F and S are Boolean functions. This predicate equation defines the same area as (1). Using predicate (2), we can now introduce R-functions. Every nondegenerative point on a plane having coordinates (x, y) can be mapped into a set of Boolean variables $\{X, Y\}$ by

$$X = S(x), Y = S(y).$$

Notice that there are four different combinations of two Boolean variables (for couples (0, 0), (0, 1), (1, 0), and (1, 1)), as shown in Fig. 1b. It means that each real valued point on the plane can be mapped into one of these combinations of Boolean variables. A function $f(x, y)$ which has a constant sign in each of the quadrants is called an R-function. That is, for an R-function

$$S(f(x, y)) = F = const,$$

where F is a Boolean variable. Fig. 1b represents the different cases defining a Boolean variable F over the real

plane and illustrates that the sign of the R-function is defined by the sign of its arguments. So, we can write

$$S(f(x,y)) = F(S(x), S(y)).$$

The link with solid modeling appears more clearly when considering that every solid can be represented by a real-valued function. The zero-set of this function usually represents the surface of the object and positive values describe its interior (or its exterior, depending on the conventions). In other words, a function f may be used as a characteristic function for a point set to distinguish if a point P belongs to the set or not. As a binary variable, the sign can be treated as a Boolean variable. The theory of R-functions [22] provides an algorithmic method for constructing functions that exactly represent virtually any geometric shape in engineering. Just as any logical function can be written using only the three operations, $\neg$, $\vee$, and $\wedge$, every R-function may be written as a composition of the three corresponding elementary R-functions. Differential properties of the three most common branches of functions, R_α, R_m^0, and R_p, are studied in [23]. For conciseness, in the following paragraphs, we will denote R-conjunction and R-disjunction with the same notation (symbol $\pm$), with R-conjunction obtained by considering $-$ and R-disjunction by considering $+$. R-negation is simply the opposite sign function $\overline{x} \equiv -x$.

2.1 Properties of R_α

$$R_\alpha : \frac{1}{1+\alpha}\left(x + y \pm \sqrt{x^2 + y^2 - 2\alpha xy}\right), \tag{4}$$

where $\alpha(f_1, f_2)$ is an arbitrary symmetric function such that $-1 < \alpha(f_1, f_2) \leq 1$. Setting α to 1 leads to the simplest R-functions: R-conjunction $min(x, y)$ and R-disjunction $max(x, y)$. Unfortunately, these functions are not differentiable along the line $x = y$.

2.2 R_0^m and R_p: An Improvement for R_α Differential Drawbacks

To solve the loss of differentiability along the line $x = y$, two other functions, R_0^m and R_p, are proposed.

$$R_0^m : \left(x + y \pm \sqrt{x^2 + y^2}\right)\left(x^2 + y^2\right)^{\frac{m}{2}}, \tag{5}$$

where m is any even positive integer. Shapiro and Tsukanov show in [23] that R_0^m is m times differentiable everywhere. Unfortunately, R_0^m is not normalized. Normalization is achieved by the R_p-function. R_p-function is differentiable everywhere except at the corner point and is defined as

$$R_p : x + y \pm (x^p + y^p)^{\frac{1}{p}}, \tag{6}$$

for any even positive integer p.

An illustration of implicit function representing multiple Boolean operations using R_p-function is shown in Fig. 2. The color coding corresponds to the interior of the objects, and the gray scale to the outside.

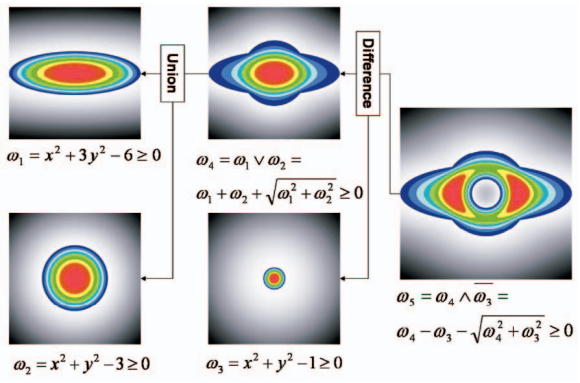


Fig. 2. Boolean operations using R_p-functions, $p = 2$.

2.3 N-Ary R-Functions

R_α, R_0^m, and R_p functions are defined to handle two arguments. Rvachev proposed the N-ary R-conjunction and R-disjunction to handle more than two arguments. We refer to these functions as "chain" functions. m is an integer and corresponds to the parameter used for R_m functions.

$$\bigwedge_{i=1}^{i=n^{(m)}} x_i \equiv \sum_{i=1}^{n} (-1)^m x_i^m (x_i - |x_i|) + \prod_{i=1}^{n} x_i^m (x_i + |x_i|). \tag{7a}$$

$$\bigvee_{i=1}^{i=n^{(m)}} x_i \equiv \sum_{i=1}^{n} x_i^m (x_i + |x_i|) - \prod_{i=1}^{n} x_i^m (-1)^m (|x_i| - x_i). \tag{7b}$$

3 Primitives and Global Deformations

The earliest primitives used by solid modelers (cube, cylinder, sphere, etc.) can be represented by superquadrics, introduced by Barr [24], [25]. Superquadrics have both a parametric and an implicit representation that are, respectively, defined by

$$\begin{pmatrix} x(\theta,\phi) \\ y(\theta,\phi) \\ z(\theta,\phi) \end{pmatrix} = \begin{pmatrix} a_1 \cos^{\varepsilon_2}\theta \cos^{\varepsilon_1}\phi \\ a_2 \sin^{\varepsilon_2}\theta \cos^{\varepsilon_1}\phi \\ a_3 \sin^{\varepsilon_1}\phi \end{pmatrix}, \tag{8}$$

and

$$F(x,y,z) \equiv \left(\left(\frac{x}{a_1}\right)^{\frac{2}{\varepsilon_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{\varepsilon_2}}\right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{z}{a_3}\right)^{\frac{2}{\varepsilon_1}} = 1. \tag{9}$$

Superquadrics have been used as a quantitative model for various applications, both in computer graphics and in computer vision [26], [27], [28]. Much effort has been put into increasing the degrees of freedom of superquadrics: Löffelmann and Gröller deal with parameterization [29], DeCarlo and Metaxas blend multiple models [30], and freeform deformations are considered by Bardinet et al. [31], [32].

Superquadrics are a special case of supershapes, recently presented by Gielis et al. [33], [34]. Compared to the superquadrics formulation, a term $\frac{m\phi}{4}$, $m \in \mathbb{R}^+$, is introduced to allow a rational or irrational number of symmetry. Three shape coefficients, n_1, n_2, and n_3, are considered. By setting

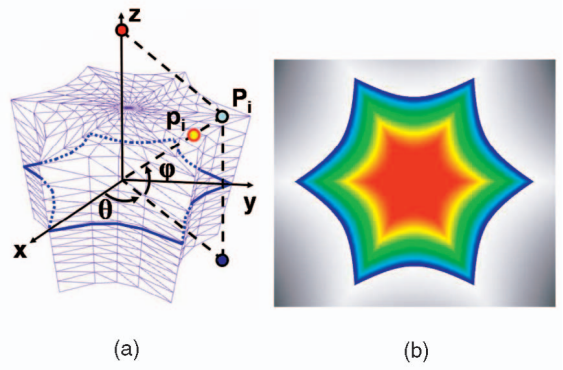


Fig. 3. Supershapes. (a) Geometric interpretation. (b) Intensity of the scalar field generated for a planar section z = const.

$n_2 \neq n_3$, asymmetrical shapes can be generated. In polar coordinates, the radius r of a superpolygon is defined by

$$r(\phi) = \frac{1}{\sqrt[n_1]{\left|\frac{1}{a}\cos\left(\frac{m\phi}{4}\right)\right|^{n_2} + \left|\frac{1}{b}\sin\left(\frac{m\phi}{4}\right)\right|^{n_3}}}, \tag{10}$$

with a, b, and $n_i \in \mathbb{R}^+$, and $m \in \mathbb{R}^+_*$. Parameters a and b control the scale, m represents the number of symmetry axes and can also be seen as the number of sectors in which the plane is folded. Coefficients n_1, n_2, and n_3 control the shape. Since we consider non-self-intersecting surfaces, m is considered as a natural number. Regular polygons can be generated by setting the shape coefficients to specific values, as shown in [34]. For $m = 4$ and $n_2 = n_3$, the original superellipses are obtained. Extension to 3D is performed by the spherical product of two superpolygons, as done for superquadrics in [25]. We note r_1 and r_2, the radii of the first and second superpolygons, respectively. Parameters of the second superpolygon are noted in capital letters (M for the number of symmetry and N_1, N_2, N_3 for the shape coefficients). To simplify the notation and stay close to the superquadric representation, we consider unit supershapes ($a = b = 1$) that are scaled among the three axes (x, y, z). A parametric version can be written as

$$\begin{pmatrix} x(\theta,\phi) \\ y(\theta,\phi) \\ z(\theta,\phi) \end{pmatrix} = \begin{pmatrix} r_1(\theta) r_2(\phi) \cos\theta \cos\phi \\ r_1(\theta) r_2(\phi) \sin\theta \cos\phi \\ r_2(\phi) \sin\phi \end{pmatrix}, \tag{11}$$

with $-\pi \leq \theta < \pi$ and $-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$.

In [34], two distance functions are proposed as an implicit function for supershapes. An equivalent implicit function can be deduced from (11).

$$F(x,y,z) = 1 - \frac{x^2+y^2+r_1^2(\theta)z^2}{r_1^2(\theta) r_2^2(\phi)}. \tag{12}$$

Notice that angles θ and ϕ are not the common spherical angles—the scale factor on the z coordinate should be $r_1 r_2$ instead of r_2 for both formulations to be equivalent. Let us consider $P_i(X, Y, Z) \in \mathbb{R}^3$ and $p_i(x, y, z) = tP_i$, with $t > 0$, that is, the intersection between the half line $]O, P_i)$ and the surface of the supershape. It can be seen from (11), that $\frac{Y}{X} = \frac{y}{x} = \tan\theta$, which leads to θ. Once θ is known, radius r_1 can be computed, then the angle ϕ can be deduced using (11) and the coordinates of P_i.

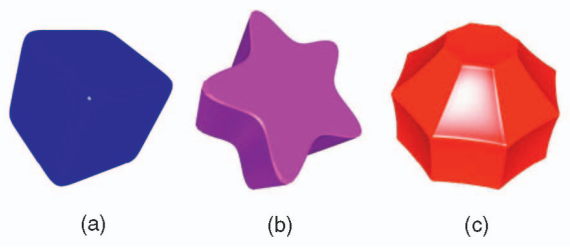


Fig. 4. Example of 3D supershapes with m a natural number. (a) $m = M = 4$. (b) $m = 5$, $M = 4$. (c) $m = M = 8$.

Fig. 3b shows the intensity of the scalar field for a plane $z = const$. Positive values define the inside of the object and range from blue ($F = 0^+$) to red (F is maximum). The outside of the object is defined by negative values and is represented in gray scale (white = 0^-, black = minimum). Examples of supershapes are shown in Fig. 4.

To increase the range of shapes that can be modeled, we also consider hierarchical deformations as proposed by Wyvill et al. in [5]. Additionally, other primitives can be used if they have both a parametric and an implicit representation, such as supertoroids or Dupin cyclides, as illustrated in Fig. 6.

4 Algorithm

The key idea of our algorithm is to combine simple and fast Boolean operations with direct and exact evaluation of the surface of the resulting object. This is the reason why we consider primitives with both implicit and parametric representations and Boolean operations using R-functions. Instead of using a single complex parametric surface, we consider multiple Boolean operations between globally deformed primitives to represent solids of arbitrary complexity. The input of the algorithm is a CSG tree of supershapes. Each node is defined by a Boolean operation and a negation flag to invert the orientation of the surface. Each node also contains the parameters of the deformations, the position, and the orientation of the primitive. The leaves of the tree contain the primitives parameters. At the beginning of the process, the vertices and faces of the primitives are stored in the leaves of the CSG tree. At the end of the process, the resulting 3D mesh, which is a subset of the union of all the oriented primitive surfaces, is stored at the root. Applied at each intermediate node, the algorithm is composed of two main operations, described in the following subsections. In Section 4.1, a fast direct solution is proposed to evaluate the intersection between two or more subtrees to determine the parts of the subtrees to be kept. In Section 4.2, we present our solution to split intersecting faces along intersection curves before merging the frontiers to build the resulting closed mesh within an ε-accuracy. The complete process applied to each node of the tree can be represented by the flowchart in Fig. 5.

4.1 Intersection Evaluation

Since we are dealing with parametric objects, we can directly generate vertices lying exactly on the surface to efficiently polygonize the surface of every primitive. Instead of considering a polygon/polygon intersection approach, we use the R-function associated with every subtree to flag the vertices: For each vertex of each subtree, the sign of the

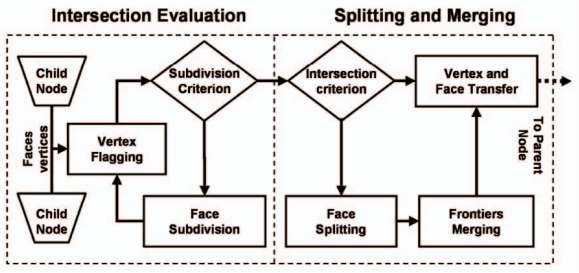


Fig. 5. Node evaluation flowchart.

implicit function (implicit function if the subtree is a superquadric or resulting R-function if the subtree is not a single primitive) determines if a vertex is lying in, on, or outside the other subobjects. Therefore, the time complexity of this operation becomes linear in the number of vertices to be evaluated because a single function is called for each vertex. A simple example is shown with two Dupin cyclides in Fig. 6a. The 3D points that are detected "inside" the other subtree are in deep blue and deep red. Faces where the sign of the implicit function is constant are completely inside or outside the object and are kept, depending on the Boolean operation performed. For example, Fig. 6b shows the faces that are kept for the Boolean union between two Dupin cyclides.

The only assumption made to isolate the intersecting faces is that an edge is crossed only once by one intersection curve. If this hypothesis is not verified, the parametric space of the primitive is locally subdivided to refine the mesh as much as desired, as shown in [4]. In theory, the only way to know if such a condition is verified is to solve the equation $Tree_{left} = Tree_{right}$, which is not the purpose. We can afford additional vertex evaluations because the time complexity for a vertex inside/outside test is constant. At the end of this operation, the faces and vertices of the resulting object are known except the faces crossed by intersection curves that are treated next.

4.2 Splitting and Merging

In this section, we discuss the last operations needed to construct the mesh representing the surface created by a Boolean operation between two subtrees. The first step is to split faces that are crossed by the intersection curves. The second operation consists of merging the split faces from the two subtrees to close the final mesh.

As mentioned before, we consider that an intersecting face is crossed only once by one intersection curve, as illustrated in Fig. 7a. This means that if we consider an arbitrary edge crossed by an intersection curve, the signs of

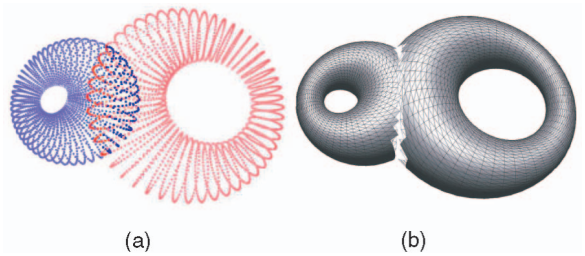


Fig. 6. Union of two Dupin cyclides. Flagged vertices and faces kept depend on the Boolean operation. (a) Detection of vertices belonging to the intersection. (b) Faces belonging to the union.

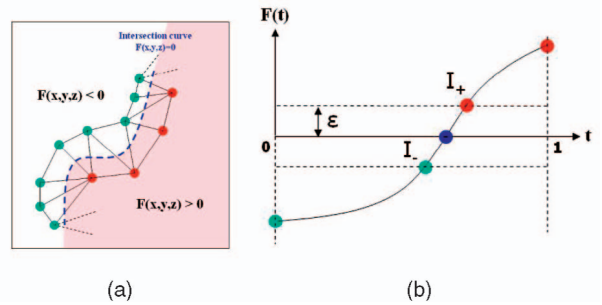


Fig. 7. Splitting process. (a) Sequence of faces crossed by an intersection curve. (b) On a parametric segment, the approximating intersection point I^+ or I^- depends on the Boolean operation.

the implicit function at its extremities are opposite. In this case, the implicit R-function applied to an edge is continuous and monotonous and, therefore, provides a zero-set point, as shown in Fig. 7b. A dichotomy approach is used on the parametric domain of the primitive to approximate this 3D point. As a vertex generated by a parametric primitive, the implicit function $F_1(P)$ associated will be zero. This vertex is therefore lying exactly on the resulting surface, but is only an approximation of the theoretical intersection point. In practice, we consider $|F_2(P)| < \varepsilon = 10^{-6}$ sufficiently small to quickly obtain accurate intersection points. It is also possible to consider a minimum distance along the gradient because the resulting implicit function can be differentiable everywhere if R_m^0 or $R_p-\text{functions}$ are used. This last technique is completely independent of the scale of the objects and makes use of normalization properties of R_p-function. For a higher accuracy, intersecting faces can be split by introducing an additional parameter, δ, that controls the maximum distance between two consecutive vertices along the approximation of the intersection curve. Its role is to fill the gaps in term of sampling or scale that may arise when evaluating important CSG trees with numerous primitives with various scales. Parameter δ is automatically adjusted in function of the smallest edge length of the two objects that are combined to avoid cracks and holes when combining primitives with very different scales or different samplings. Parameter δ is therefore transparent for the user during the execution. In the case of a large disparity of scale or sampling, it behaves like a security to preserve a minimum sample density along intersection curves. Fig. 8 presents two results for the difference between two spheres without merging. Fig. 8a shows the result with common values used in our experiments, i.e., a minimum sampling for the primitive surfaces of 10,000 faces, $\epsilon = 10^{-6}$, and $\delta = 10^{-2}$. These values increase the number of faces generated and, therefore, the number of faces to be split, but avoid the introduction of degenerated triangles. Fig. 8b shows the result using a coarser surface sampling and a larger $\epsilon = 10^{-3}$ without merging. Notice the hole between the two parts due to the larger ϵ and the effect of parameter δ to avoid coarse sampling around the intersection curve on Fig. 8c. After merging and triangulation update, the final vertices will be within an ε-stripe along the frontier.

The last operation performed is the merging of the frontiers of the subtrees: Vertices along both object frontiers, representing an approximation of intersection curves on each object, are merged to close the resulting mesh. We refer

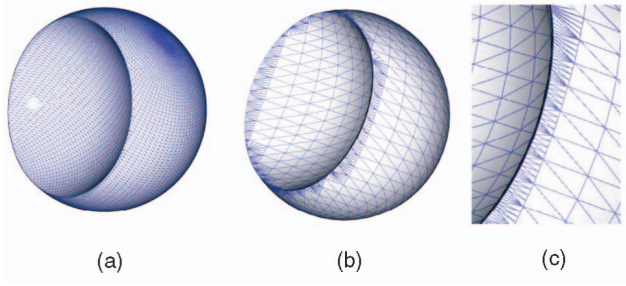


Fig. 8. Intermediate results of Boolean difference between two spheres. (a) Common values. (b) If the sampling rate is not fine enough, higher point density around the intersection curve can be obtained by using smaller δ values. (c) Hole created by the gap between the primitive frontiers (in black), generated with larger ε.

to [4] for a possible technique. In our case, we fuse vertices belonging to the same ε-ball into their center of gravity, as shown in the results section. An additional fusion and replacement of identical or close vertices is also performed to reduce the number of faces used and to avoid degenerate triangles or identical points.

As a remark, directly computing the R-function at the root of the CSG tree and applying it to the vertices is also possible and has been tested. We apply our algorithm node by node for the following reasons:

- The simpler the R-function is, the smaller the computational error is. This becomes crucial during intersection curve approximation.
- It is easier and faster to merge the frontiers of two objects than many frontiers from many objects, especially if these curves are very close: By considering binary trees and working node by node, we avoid other problems about the topology of the object near the merged frontiers.

4.3 Discussion

Our main concern for discussion is the definition of the resulting surface. It is commonly considered that the surface represents an iso-value of the implicit function, the zero-set in most of cases. Unfortunately, in some cases, zero values of the resulting R-Function are inside the object. This situation happens when the union of tangent objects is performed as shown in Fig. 9a. This issue leads to the concept of regularized Boolean operations, introduced in the very beginning of the research on CSG by Voelcker and Requicha [15]. Therefore, considering the surface only as the zero-set of the function is not sufficient to describe all the cases and some additional assumptions on the model must be made to avoid ambiguities. Actually, the problem of tangency comes from applying a 2-valued logic when 3-valued logic is required [19], [21], [22]. If there is no technique to solve the equation $F = G$, where F and G represent the implicit equation of the two considered CSG trees, which represents most of the cases, the authors are obliged to use heuristics. We present in this section some concepts used in the literature to avoid this problem and discuss their major advantages and drawbacks.

An immediate solution is not only to consider the surface as the function zero-set, but also to assert its sign changes in a given neighborhood. Related to the algorithm, one final step is added to evaluate the R-function on a neighborhood of points. But, in practice, evaluating sampled values on small balls enclosing zero value points does not guarantee a correct result or solve the problem of zeros inside.

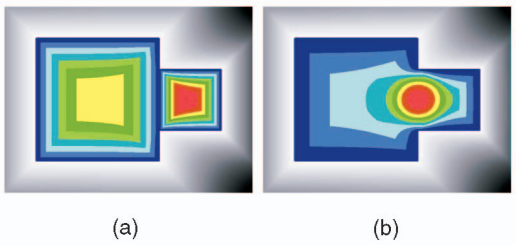


Fig. 9. Example of zero values inside the object. (a) R-function created by two tangent squares. (b) Circular patch added to overlap the zero-set within the object.

Another solution is to consider a blending function as mentioned by Wyvill et al. in [5]. Intuitively, this corresponds to adding a quantity of material depending on the distance between the two primitives. This approach is well-known, widely used, and is especially useful to generate smooth links between objects. But, if this solution solves the problem of zero values inside the object, its drawback is the loss of the parametric domain to generate the solutions for the resulting R-function. Using this approach, the resulting mesh is no longer defined by a subset of the primitive surfaces. In other words, the method to generate the resulting surface must use classical implicit surface meshing techniques, such as the marching cubes [8], [9]. This implies that we lose the ability to efficiently and exactly generate points on the surface using the parametric form of the primitives.

The last two solutions come from Rvachev, who considers patches enclosing the zero value domain. In other words, he eliminates object tangency by using overlapping patches, as shown in Fig. 9b. This solution also solves the inside zero value problem, but is not applicable in practice. First, these tangent parts must be defined, which implies directly solving the equation $Object_a = Object_b$ and leads to complicated if not unsolvable equation systems. Then, even if the inside zero values could be efficiently determined, the definition of the patch itself is critical and also leads to complex equation solving problems. The second solution proposed by Rvachev uses a three-valued logic instead of two-valued logic. If $f \geq 0$ defines a closed area P and f is a continuous function, generally speaking, it does not automatically mean that $f = 0$ defines the boundary of the area P. As shown in [22], this property can only be guaranteed for arbitrary primitives if one uses three-valued logic, but this leads to other practical problems and is not as comfortable and natural as Boolean logic.

So, in practice and in the context of solid modeling, a slight blending can be used to perform unions of tangent objects if the user is aware of the tangency, but if the zero-set of the function and the definition domain of the solutions are the primary goals, it is up to the user to properly model his CSG tree and to assert that there is no tangency between primitives. In practice, we detect when zero value points are inside the resulting object because the sign of the function must change in a small neighborhood.

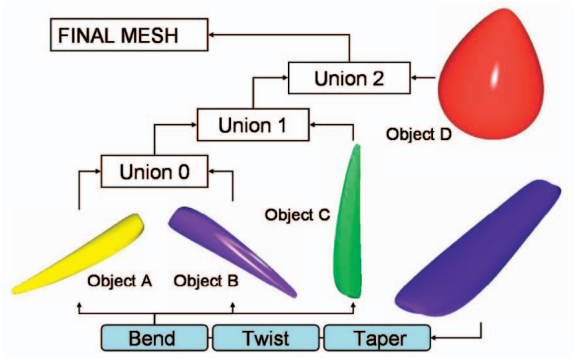


Fig. 10. A CSG tree for the propeller model.

5 Results

We present in this section three objects modeled with our algorithm. We first present detailed results for a propeller model, including the main intermediary steps and the values for every parameter. The resulting mesh and the implicit function intensity are presented in Fig. 13. The bolt presented in Fig. 14 is obtained by the composition of five supershapes without deformation. The bolt could also be represented by superquadrics, except that its head—which has a hexagonal section—would require more superquadrics to be represented. The piston head in Fig. 15 is a more complex object composed of nine supershapes. For each object, we show a possible binary CSG tree, the resulting surface, one or many examples of planar sections to show the intensity of the resulting implicit function, and details on the areas where the frontiers of the primitives have been merged.

The CSG tree of the propeller model is composed of four supershapes and three Boolean operations, as shown in Fig. 10. Parameters for supershapes A, B, and C are identical except for the position and orientation. The supershape coefficients are presented in Table 1 and parameters of deformations and transformations in Table 2. Parameters k_x, k_y are the tapering parameters, k and α correspond to the bending as in [27]. For the twisting, O_x and O_y are the coordinates of a rotation axis parallel to the z-axis and ψ is the total angle rotation. In the following results, we use R_p-functions with $p = 2$ to obtain an implicit equation for the whole object. We note Ω_i intermediate objects and ω_i, their respective implicit equations.

First, we perform Ω_1, the union of supershapes A and B. Ω_1 implicit equation is then defined by

$$\omega_1 = A + B + \sqrt{A^2 + B^2} \geq 0.$$

TABLE 1
Supershape Parameters for the Propeller Model

	2D-Supershape 1				2D-Supershape 2			
Label	m	n_1	n_2	n_3	M	N_1	N_2	N_3
Object A	1	0.5	0.5	0.5	4	5	5	5
Object B	1	0.5	0.5	0.5	4	5	5	5
Object C	1	0.5	0.5	0.5	4	5	5	5
Object D	4	2	2	2	1	10	5	5

The next intermediate object Ω_2 is defined by the union of Ω_1 and supershape C and its implicit equation ω_2 is

$$\omega_2 = C + \omega_1 + \sqrt{C^2 + {\omega_1}^2} \geq 0.$$

Notice that there is no intersection between the three supershapes A, B, and C, which avoid split and merge operations during the first two unions. The last union between supershape D and intermediate object Ω_2 is then performed. The equation ω_3, corresponding to the final object Ω_3, is therefore defined by

$$\omega_3 = D + \omega_2 + \sqrt{D^2 + {\omega_2}^2} \geq 0.$$

The expanded equation is then

$$\omega_3 = A+B+C+D+\sqrt{A^2+B^2}+\sqrt{\left(A+B+\sqrt{A^2+B^2}\right)^2+C^2}+\sqrt{\left(A+B+C+\sqrt{A^2+B^2}+\sqrt{\left(A+B+\sqrt{A^2+B^2}\right)^2+C^2}\right)^2+D^2} \geq 0. \quad (13)$$

The intensity of the scalar field generated by this function can be seen for different planar sections in Fig. 13b and Fig. 13c. The color coding represents the intensity of the function over the considered section (from blue for $f = 0^+$ to red for f_{max}). During the last union, intersections between Ω_3 and D are detected; therefore, intersecting faces must be split and merged to obtain the final result as shown in Fig. 11.

As mentioned before, we use an implicit function to evaluate the parts of the primitives to be kept. For each vertex of each primitive, this operation is constant in time and depends on the number of primitives of the CSG tree. For classical methods, at each node, each vertex of each subtree must be evaluated regarding each face of the other subtree [35]. The complexity for a Boolean operation between two objects is therefore $O(n + m)$ for all the implicit methods, whereas it is $O(nm)$ for classical methods, with n and m representing the number of faces of the object. As an illustration, we measured the time execution to compute the final mesh for three objects. Processing time for

TABLE 2
Deformations and Positioning Parameters

	Scaling			Tapering		Twisting			Bending		Rotation			Translation		
Label	S_x	S_y	S_z	k_x	k_y	ψ	O_x	O_y	k	α	R_x	R_y	R_z	T_x	T_y	T_z
Object A	1.7	0.75	7.3	0.65	0.85	0.7	0.0	0.0	0.012	1.57	3.664	0	-0.8	0	4	-6.928
Object B	1.7	0.75	7.3	0.65	0.85	0.7	0.0	0.0	0.012	1.57	1.571	0	-0.8	0	-8	0
Object C	1.7	0.75	7.3	0.65	0.85	0.7	0.0	0.0	0.012	1.57	-0.524	0	-0.8	0	4	6.928
Object D	2.3	2.3	2.9	-0.45	-0.45	0.0	0.0	0.0	0	0	0	1.571	-0.524	1.8	0	0

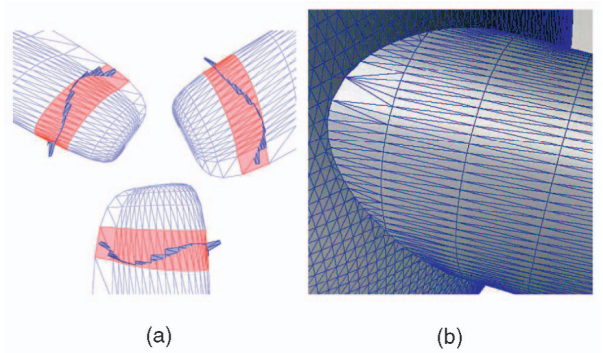


Fig. 11. Propeller example. (a) Intersecting faces detected. (b) Zoom on merged frontiers of the final result.

splitting and merging is not taken in account in this comparison because it is the same in both cases. Fig. 12 shows the results for both methods. The number of faces in abscissa represents the sum of all the faces of all the primitives.

The naive approach we implemented for comparison has no optimization using k-trees or bounding boxes. Notice in Fig. 12b that the propeller, which is only composed of four primitives, is the first model to become extremely slow to be computed. There are two main explanations: The propeller is composed of fewer primitives, so the number of faces and vertices per node is higher than for the other objects. Moreover, the number of faces kept at each operation is higher than other models. There is no intersection between supershapes A, B, and C, which implies no face is removed during these operations. So, as the total number of faces increases, the number of faces per intermediate node increases faster than for the bolt and the piston models. As we can see from Fig. 12a, our method based on R-function has a lower time complexity than the naive approach. The time complexity of our method is linear in the number of faces considered and the slope depends on the number of Boolean operations performed. Since objects can be described by an implicit equation, their volume can be computed using a Monte Carlo integration by taking a large number of uniformly random points within the bounding box of the object and counting the ones that are within the object using its implicit function as a characteristic function. The volume of the object is

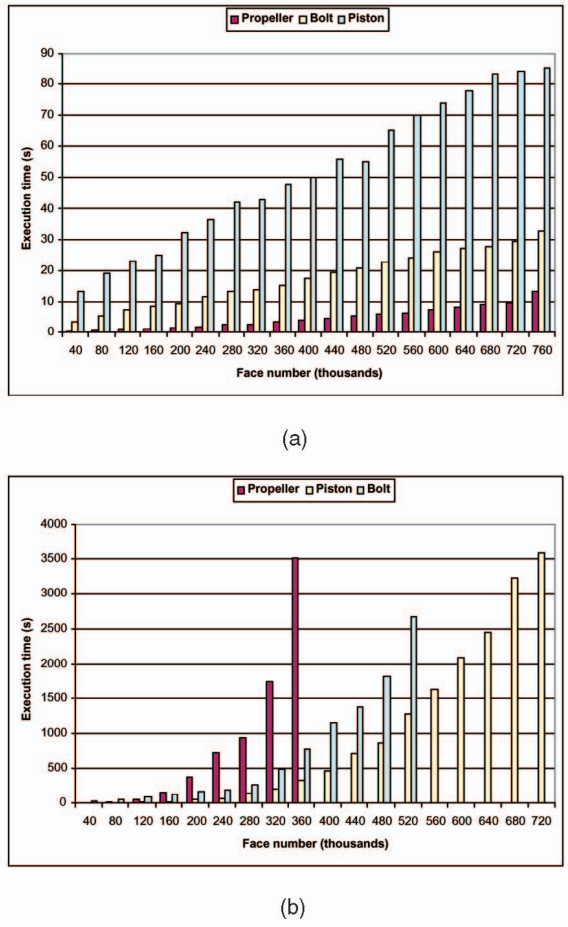


Fig. 12. Execution time in function of the total number of faces for the classical and implicit methods. (a) Using R-functions. (b) Naive approach.

$$V_{obj} \approx \frac{n}{N} V_{bb},$$

where n corresponds to the number of points within the object O, N is the total number of points used, and V_{bb} is the volume of the bounding box.

The same technique can also be applied for arbitrary section area: Any arbitrary surface can be used to get a section of the object. As long as the area of the complete surface patch can be evaluated, the area of the section of the

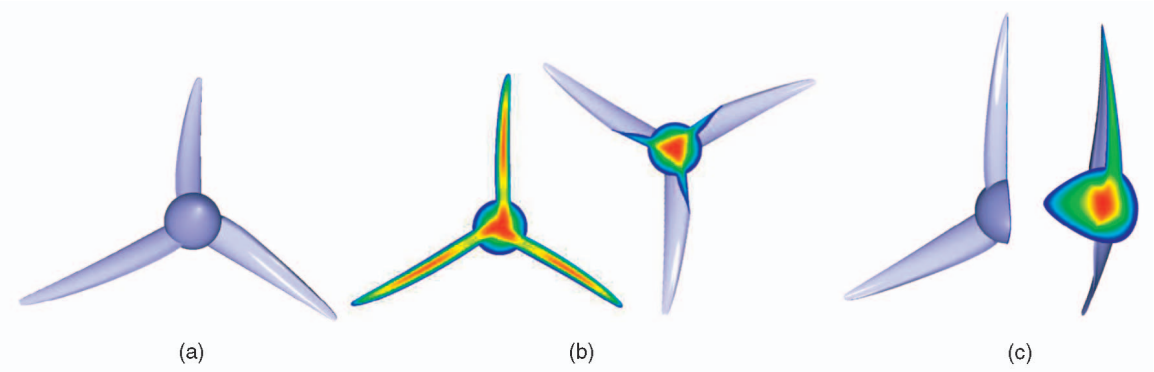


Fig. 13. Propeller model. (a) Final mesh. (b) Horizontal sections and intensity of the resulting R-function. (c) Transversal section and intensity of the resulting R-function.

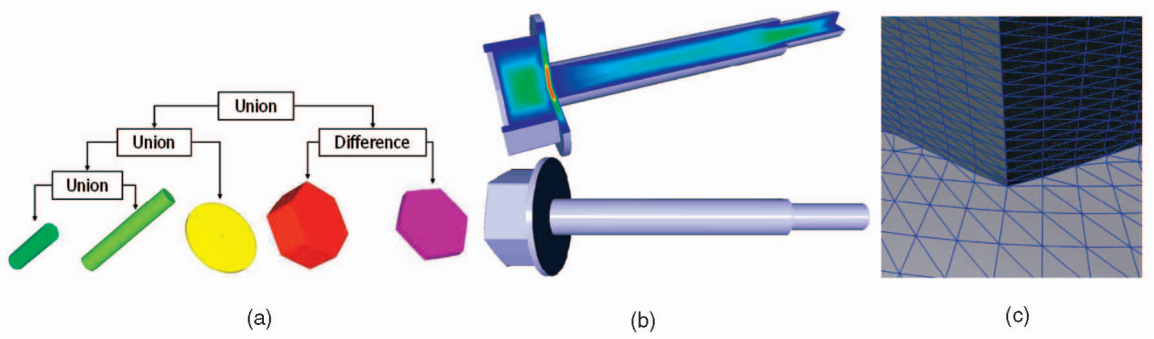


Fig. 14. Bolt model. (a) A binary CSG tree. (b) Final mesh and scalar field generated by the resulting R-Function. (c) Zoom on merged frontiers of two supershapes.

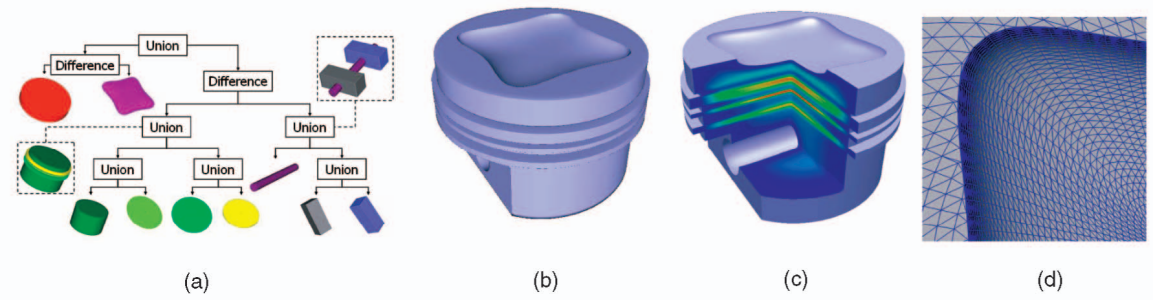


Fig. 15. Piston head model. (a) A binary CSG tree. (b) Final mesh. (c) Scalar field generated by the resulting R-Function. (d) Zoom on merged frontiers of two supershapes.

object can be obtained by using the implicit equation, a characteristic function to determine if randomly generated points on the surface patch are within the object or not. The area of the section of the object is then given by the product of the area of the surface patch by the ratio of points in the object over the total number of points computed.

6 Conclusions and Future Work

In this paper, we have proposed a new and efficient algorithm based on R-functions theory to compute multiple Boolean operations between primitives with both an implicit and a parametric representation. The resulting object is described by an implicit equation and the vertices composing the resulting mesh lie exactly on its zero-set. The time complexity for node evaluation stays linear in the number of faces and the polygonization of the surface is kept simple due to the parametric form of the primitives. The theory of R-functions is used to obtain an implicit function with guaranteed differential properties and the parametric forms of the primitives are used to generate the solutions, i.e., polygonizing the surface. In input, we consider a CSG tree that contains the parameters of the supershapes, the Boolean operations, and the global deformations. The output includes two components: an implicit equation whose zero-set represents the surface and a 3D mesh whose points lie exactly on the resulting surface. The only approximation made is on the borders of the parametric domain for the solutions. As a subset of the union of all the primitive parametric intervals, the parametric interval considered as a solution is a subset of the theoretical parametric solution. These two intervals tend to be equal when ε tends to zero. This approximation is due to the dichotomy process used to split the faces and the parametric interval: We are not able to explicitly solve the equation representing the intersection of objects, so we approximate the parametric solutions within a ε-precision. Moreover, our algorithm can handle any primitive with both a parametric and an implicit, which allows us to consider objects such as supertoroids or Dupin cyclides.

Being now able to represent complex shapes with a CSG tree of primitives and a single implicit equation, we are going to study possible solutions to recover the CSG tree and the parameters of the primitives from a cloud of points or a 3D mesh to apply our method to reverse engineering and computer vision.

Acknowledgments

The authors would like to thank David Page and Brad Grinstead for their valuable comments, and Michaël Roy for his implementation advice. Dr. Gribok is thankful to Anton Bougaev for introducing him to the theory of R-functions. This work is supported by the University Research Program in Robotics under grant DOE-DE-FG02-86NE37968, by the DOD/RDECOM/NAC/ARC Program, R01-1344-18, by the US Army under grant Army-W56HC2V-04-C-0044, and by FAA/NSSA Program, R01-1344-129/130 and R01-1344-088.

References

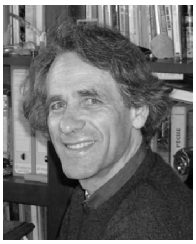
[1] L. Piegl and W. Tiller, *The NURBS Book*, second ed. Springer, 1997.
[2] J. Stam, "Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values," *Proc. SIGGRAPH '98, Ann. Conf. Series*, pp. 395-404, 1998.
[3] S. Krishnan, D. Manocha, M. Gopi, T. Culver, and J. Keyser, "Boole: A Boundary Evaluation System for Boolean Combinations of Sculptured Solids," *Int'l J. Computationnal Geometry & Applications*, no. 1, pp. 105-144, 2001.

[4] H. Biermann, D. Kristjansson, and D. Zorin, "Approximate Boolean Operations on Free-Form Solids," *Proc. SIGGRAPH '01,* pp. 185-194, 2001.
[5] 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, 1999.
[6] J.F. Blinn, "A Generalization of Algebraic Surface Drawing," *ACM Trans. Graphics,* vol. 1, no. 3, pp. 235-256, 1982.
[7] 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, 1995.
[8] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics,* vol. 21, no. 4, pp. 163-169, 1987.
[9] J. Bloomenthal, "Polygonization of Implicit Surfaces," *Computer Aided Geometric Design,* vol. 5, no. 4, 1988.
[10] C.M. Hoffmann, *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann, 1989.
[11] J.R. Rossignac, "Constraints in Constructive Solid Geometry," *Proc. 1986 Workshop Interactive 3D Graphics,* 1986.
[12] J.R. Rossignac and H.B. Voelcker, "Active Zones in CSG for Accelerating Boundary Evaluation, Redundancy Elimination, Interference Detection, and Shading Algorithms," *ACM Trans. Graphics,* vol. 8, no. 1, 1988.
[13] J.R. Rossignac and A.A. G. Requicha, "Constructive Non-Regularized Geometry," *Computer-Aided Design,* vol. 23, no. 1, pp. 21-32, 1991.
[14] J.R. Rossignac, "Blist: A Boolean List Formulation of CSG Trees," Technical Report GIT-GVU-99-04, GVU Tech., 1998.
[15] H.B. Voelcker and A.A.G. Requicha, "Research in Solid Modeling at the University of Rochester: 1972-1987," *Fundamental Developments in Computer-Aided Modeling,* L. Piegl, ed., London: Academic Press, 1993.
[16] A. Rappoport and S. Spitz, "Interactive Boolean Operations for Conceptual Design of 3-D Solids," *Proc. of SIGGRAPH '97, Ann. Conf. Series,* pp. 269-278, 1997.
[17] V. Adzhiev, M. Kazakov, A. Pasko, and V. Savchenko, "Hybrid System Architecture for Volume Modeling," *Computers & Graphics,* vol. 24, no. 1, pp. 194-203, 2000.
[18] R. Allègre, A. Barbier, E. Galin, and S. Akkouche, "A Hybrid Shape Representation for Free-Form Modeling," *Proc. Shape Modeling Int'l,* 2004.
[19] V.L. Rvachev, *Geometric Applications of Logic Algebra.* Naukova Dumka, 1967, in Russian.
[20] V. Shapiro, "Theory of R-Functions and Applications: A Primer," Technical Report TR91-1219, Computer Science Dept., Cornell Univ., Ithaca, N.Y., 1991.
[21] V.L. Rvachev, *Methods of Logic Algebra in Mathematical Physics.* Naukova Dumka, 1974, in Russian.
[22] V.L. Rvachev, *Theory of R-Functions and Some Applications.* Naukova Dumka, 1982, in Russian.
[23] V. Shapiro and I. Tsukanov, "Implicit Functions with Guaranteed Differential Properties," *Proc. Symp. Solid Modeling and Applications,* pp. 258-269, 1999.
[24] A.H. Barr, "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications,* vol. 1, no. 1, pp. 481-484, 1981.
[25] A.H. Barr, "Global and Local Deformation of Solid Primitives," *Computer Graphics,* vol. 18, no. 3, pp. 21-30, 1984.
[26] M.E. Montiel, A.S. Aguado, and E. Zaluska, "Surface Subdivision for Generating Superquadrics," *The Visual Computer,* vol. 14, no. 1, pp. 1-17, 1998.
[27] A. Jaklič, A. Leonardis, and F. Solina, *Segmentation and Recovery of Superquadrics.* Dordrecht: Kluwer Academic, 2000.
[28] A. Jaklic and F. Solina, "Moments of Superellipsoids and Their Application to Range Image Registration," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 33, no. 4, pp. 648-657. 2003.
[29] H. Löffelmann and E. Gröller, "Parameterizing Superquadrics," *Proc. (WSCG '95),* pp. 162-172, 1995.
[30] D. DeCarlo and D. Metaxas, "Shape Evolution with Structural and Topological Changes Using Blending," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, pp. 1186-1205, 1998.
[31] E. Bardinet, L.D. Cohen, and N. Ayache, "Fitting of Iso-Surfaces Using Superquadrics and Free-Form Deformations," *Proc. Int'l Conf. Pattern Recognition (ICPR),* pp. 79-83, 1994.
[32] E. Bardinet, L.D. Cohen, and N. Ayache, "Superquadrics and Free-Form Deformations: A Global Model to Fit and Track 3D Medical Data," *Proc. Conf. Virtual Reality in Medicine (CVRMed),* pp. 319-326, 1995.
[33] J. Gielis, "A Generic Geometric Transformation that Unifies a Wide Range of Natural and Abstract Shapes," *Am. J. Botany,* vol. 90, pp. 333-338, 2003.
[34] J. Gielis, B. Beirinckx, and E. Bastiaens, "Superquadrics with Rational and Irrational Symmetry," *Proc. Symp. Solid Modeling and Applications,* pp. 262-265, 2003.
[35] J. O'Rourke, *Computational Geometry in C,* second ed. Cambridge Univ. Press, 1998.

Yohan D. Fougerolle received the MS degree from the University of Burgundy, Dijon, France, in 2002, in electrical engineering. He is a PhD student in electrical and computer engineering at the University of Burgundy, Le Creusot, France, and is currently a visiting research scholar in the Imaging, Robotics, and Intelligent Systems Laboratory, Knoxville, Tennessee. His research interests include computer vision, solid modeling, and surface reconstruction.

Andrei Gribok received the BS degree in system science and the MS degree in nuclear engineering from the Moscow Institute of Physics and Engineering. He received the PhD degree from the Moscow Institute of Biological Physics in the area of acoustical pattern recognition, artificial intelligence and nondestructive testing. He is a research assistant professor in the Department of Computer and Electrical Engineering at the University of Tennessee, Knoxville, and also holds a position as an adjunct professor of statistics in the Department of Statistics at the same university. His area of expertise is inverse and ill-posed problems in engineering, statistical learning, and model misspecification in statistics. His publications list includes three book chapters and numerous journal and conference papers. He has 15 years of experience in industry as well as in academia. Dr. Gribok worked as an invited scientist at the Cadarache Nuclear Research Centre in France, where his work focused on nuclear power plants monitoring, diagnostics, and ultrasonic imaging.

Sebti Foufou received the PhD degree in computer science in 1997 from the University of Claude Bernard Lyon I, France, for a dissertation on parametric surfaces intersections. Since then, he has been working as an associate professor in the Computer Science Department at the University of Burgundy, France. His research interests concern geometric modeling and CAD-CAM topics and include surfaces blending using Dupin cyclides, subdivision surfaces, and geometric constraints solving. Currently, he is working as a temporary guest researcher at the National Institute of Standards and Technology, Gaithersburg, Maryland, where he is contributing in product engineering-related research: smart machining systems, tolerances, assembly modeling, and PLM.



Frédéric Truchetet received the master's degree in physics from Dijon University, France, in 1973 and the PhD degree in electronics from the same University in 1977. He was with Thomson-CSF for two years as a research engineer and he is currently a full professor in Le2i and vice president of the Université de Bourgogne, France, delegate for research. His research interests are focused on image processing for artificial vision inspection and particularly on wavelets transform, multiresolution, edge detection, and image compression. He has authored or coauthored more than 150 international publications, three books, and holds one patent. He is member of GRETSI, ASTI, IEEE, SPIE, chairman of SPIE's Conference on Wavelet Applications in Industrial Processing, and a member of numerous technical committees of international conferences in the area of computer vision.

Mongi A. Abidi received the PhD degree in electrical engineering from the University of Tennessee in 1987, the MS degree in electrical engineering from the University of Tennessee in 1985, and principal engineer in electrical engineering from the National Engineering School of Tunis, Tunisia in 1981. He is a professor and associate department head in the Department of Electrical and Computer Engineering and directs activities in the Imaging, Robotics, and Intelligent Systems Laboratory at the University of Tennessee, Knoxville. He conducts research in the field of 3D imaging, specifically in the areas of scene building, scene description, and data visualization. He is a member of the IEEE.