29

Representation, Boundary Computation and Fast Display of CSG Models with NURBS Primitives

Shankar Krishnan, Subodh Kumar and Dinesh Manocha
Department of Computer Science
University of North Carolina
Chapel Hill NC 27599
{krishnas,kumar,manocha}@cs.unc.edu
http://www.cs.unc.edu/~geom/geom.html

Abstract

We present efficient and accurate algorithms for Boolean combinations of solids composed of sculptured models. The boundary of each solid is represented as a collection of trimmed spline surfaces and a connectivity graph. Based on algorithms for trapezoidation of polygons, partitioning of polygons using polygonal chains, surface intersection of high degree spline surfaces and ray-shooting, the boundaries of the resulting solids and its connectivity graph after the Boolean operation are computed. We also present accurate representations of the intersection curves and boundary of the resulting solids. The resulting boundaries are used for interactive display and walkthrough applications. The system has been used to convert parts of a submarine storage and handling system model represented as more than 2,000 CSG trees. The B-rep consists of more than 30,000 trimmed spline surfaces and is displayed at interactive rates on the Pixel-Planes 5 graphics system.

Keywords

surface intersection, NURBS, boundary representation, robustness, trimming curves, solid modeling

M. J. Pratt et al. (eds.), *Product Modeling for Computer Integrated Design and Manufacture*

1 INTRODUCTION

Rational spline surfaces are routinely used to represent solids in engineering design. In addition, solid models composed of polyhedra, spheres, cones, tori, prisms, solids of revolution etc. are widely used in CAD/CAM, virtual reality, engineering simulation and animation [Hof89]. All these solids can also be easily represented in terms of rational spline surfaces. These solids and their Boolean combinations, i.e. union, intersection and difference, are used to generate large-scale models like those of airplanes, ships, automobiles, submarines etc. These models are typically represented as thousands of CSG trees with up to a hundred Boolean operations on primitives corresponding to surfaces of high parametric degree. Many of the current solid modelers do not compute analytic and accurate B-rep for these models and represent them as millions of polygons. Current graphics systems are not able to render such complex polygonal models at interactive frame rates. Thus there is a wide body of application that benefits from accurate CSG display based on a spline visualizer.

Main Contribution: We present algorithms and systems to display CSG and spline models at interactive frame rates. The main components of the system are:

- *Boundary Computation*: Computation of accurate B-reps from the CSG models and their representation in terms of trimmed spline models.
- *Trimming Curves Representation*: An accurate and efficient representation of the high degree trimming curves.
- *Display System*: Fast and accurate rendering of models composed of trimmed spline surfaces. Immersive design and design validation are the target applications of our system.

1.1 Prior Work

There is a great deal of work in the modeling and rendering literature related to model conversion, displaying large data sets and multiresolution modeling. We categorize it into:
CSG to B-rep Conversion: There is considerable literature on computing B-reps from CSG solids defined using polyhedral primitives, as surveyed in [Hof89, RR92]. A number of techniques to improve the robustness of such systems have been proposed as well [Seg90]. However, no such robust algorithms and systems are known for solid models composed of curved or spline primitives [Hof89, RR92]. The main problem lies in accurate, robust and efficient computation of the intersection of spline surfaces. Surface intersection is an active area of research and some of the recent papers have proposed algorithms to compute all components and piecewise linear or spline approximations to the intersection curves based on tracing methods [SN91, Hoh91, Hof89, KM94]. However, these representations are either inaccurate or inconsistent for Boolean operations on complex models. Therefore, some of the current solid modelers use polyhedral approximations of these primitives and compute the B-reps of the resulting solids in terms of polygons. This method can potentially lead to data proliferation and inaccurate representations.

Rendering Spline Surfaces: Many algorithms are known in the literature for tessellating spline surfaces and rendering the resulting polygons [AES91, RHD89, LC93, KML95, BR94]. In particular, Rockwood et. al. [RHD89] presented the first real time algorithm

for rendering trimmed surfaces. Given a NURBS model, Rockwood et. al. decompose it into a series of trimmed Bézier surfaces, split the trimmed surfaces into patches bounded by monotonic curve segments and triangulate the resulting patches. However, it is not fast enough for complex models and its implementation as part of SGI's GL library can render surfaces composed of at most a few hundred trimmed Bézier surfaces at interactive rates on an SGI Onyx [KML95].

1.2 Overview

Given a CSG model, our solid modeler represents each primitive as a collection of Bézier surfaces, performs accurate Boolean operations and represents the B-rep as a collection of trimmed Bézier surfaces. It represents the trimming curves as piecewise algebraic space curves along with bounding volumes. It makes use of algorithms for surface intersection, polygon triangulation, domain partitioning and ray-shooting to compute the B-reps. Given a collection of trimmed Bézier surfaces, our display system tessellates them into triangles and renders them on the graphics pipeline.

The ability to compute varying resolutions of the B-reps and accurate representations in terms of trimmed curves and surfaces is fundamental to the performance of the overall system. As compared to earlier systems for rendering such models, it has the following advantages:

- **Accuracy:** A high degree of accuracy of the curves is essential for accurate boundary computation as well as meaningful visualization and design validation. Our B-reps for CSG models are more accurate than those generated by modelers using polygonal representation for the curved primitives and the final solids. Besides rendering, it is also useful for other applications like collision detection.
- **Rendering:** Our algorithms based on visibility culling and dynamic tessellation generate fewer polygons for the spline models. Furthermore, we can easily generate on-line any *level-of-detail* with correct topology using incremental computations. This is in contrast with the difficulty of computing multiresolution models for large polygonal datasets of arbitrary topology with visible artifacts introduced due to few and discrete levels of detail. Our method results in *better images and faster display.*

The rest of the paper is organized in the following manner. Section 2 presents the notation used in the rest of the paper and the multiresolution representation for the trimming curves. Section 3 describes model generation, representation and the underlying algorithms. We present the display system in Section 4 and discuss the overall implementation and performance on parts of the submarine storage and handling system model in Section 5.

2 MODEL REPRESENTATION

We use bold face letters to represent vector quantities and surfaces. Lower case letters are used for representing points and curves in a plane and upper case letters for points, curves and surfaces in $\mathcal{R}^3$. A tensor product Bézier surface $\mathbf{F}(s,t)$ is represented as:

$$\mathbf{F}(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{V}_{ij} B_i^m(s) B_j^n(t)$$

where $\mathbf{V}_{ij} = \langle w_{ij}x_{ij}, w_{ij}y_{ij}, w_{ij}z_{ij}, w_{ij} \rangle$ are the control points of the patch in homogeneous coordinates and $B_i^m(s) = \binom{m}{i} s^i(1-s)^{m-i}$ is the Bernstein polynomial [Far93]. The domain of the surface is defined on the unit square $0 \leq s,t \leq 1$ in the (s,t) plane. Trimmed surfaces are defined on a subset of $[s,t] \in [0,1] \times [0,1]$ domain using trimming curves, $t_1, t_2, \ldots, t_p$. The trimming curves are represented as piecewise linear curves, Bézier curves or piecewise algebraic space curves. Each trimming curve *trims out* the part of the surface that lies on its right. It does not allow self intersecting trimming curves. The region of the surface that is not trimmed out is also referred to as the trimmed region.

2.1 Representation of Solids

Each solid $\mathbf{S}$ compsists of planar faces and trimmed Bézier surfaces. The data structure for the solid includes the number of surfaces, representation of each surface and an adjacency graph $\Gamma(\mathbf{S})$ representing the connectivity between all the surfaces (Fig. 2). Each planar face is defined as an anti-clockwise ordering of the vertices and for each Bézier surface $\mathbf{F}(s,t)$, the cross-product $\mathbf{F}_s \times \mathbf{F}_t$ points to the outer normal. As a result, we can perform *local* in/out tests at any point on the boundary of the solid.

Each vertex v_i in the adjacency graph corresponds to a surface $\mathbf{F}_i$ of the solid. Two vertices v_j and v_k are connected by an edge, iff the two surfaces $\mathbf{F}_j$ and $\mathbf{F}_k$ share a common boundary. The common boundary may be along the edges of a planar face or a boundary curve or trimming curve for a Bézier surface.

2.2 Representation of Trimming Curves

The intersection curves of two surfaces correspond to the vector equation $\mathbf{F}(s,t) = \mathbf{G}(u,v)$. This results in the following three scalar equations:

$$\begin{aligned} F_1(s,t,u,v) = X(s,t)\overline{W}(u,v) - \overline{X}(u,v)W(s,t) &= 0 \\ F_2(s,t,u,v) = Y(s,t)\overline{W}(u,v) - \overline{Y}(u,v)W(s,t) &= 0 \\ F_3(s,t,u,v) = Z(s,t)\overline{W}(u,v) - \overline{Z}(u,v)W(s,t) &= 0, \end{aligned} \tag{1}$$

where the domain of the variables is restricted to $0 \leq s,t,u,v \leq 1$. We use a recently developed algorithm for computing the intersection of rational parametric surfaces [KM94]. In this algorithm, we maintain an accurate analytic representation of the intersection curve in the domain of the two surfaces. This can be used to obtain accurate trimming curves during the CSG operation. The algorithm computes a start point on each component of the intersection curve (including loops) and decomposes the domain such that

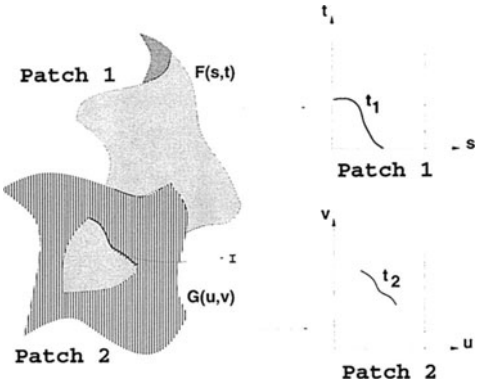

Figure 1 Intersection of two surfaces

each resulting region contains at most one component of the intersection curve. Along with the analytic representation, the algorithm also computes a series of points on each component. In order to guarantee a fairly close approximation to the actual intersection curve, a densely tessellated piecewise linear curve needs to be generated using a small step-size. However, the problems of generating such densely tessellated curves for the purposes of rendering are twofold. Firstly, the number of triangles generated for a trimmed patch by our rendering algorithm is bounded below by the number of points on the trimmed curve. A densely tessellated trimmed curve produces an over-tessellated patch. Secondly, it causes data proliferation, which limits the size of the models that can be handled by the modeling and rendering system.

Replacing the piecewise linear trimmed curves by parametric curves is an obvious solution. It is a well known that the intersection curve between two rational parametric surfaces generally cannot have an exact representation in rational parametric form. Therefore, we can only hope to approximate the intersection curve within a tolerance value. The intersection algorithm generates points in the domain of the two surfaces (as a (u, v, s, t) tuple such that $\mathbf{F}(s,t) = \mathbf{G}(u,v)$). Given this set of points, let the parametric approximation to the intersection curve be denoted by $(\mathbf{u}(\alpha), \mathbf{v}(\alpha), \mathbf{s}(\alpha), \mathbf{t}(\alpha))$. However, the mapping functions that take the respective curves from domain to $\mathcal{R}^3$ are different, and hence these curves might not match well in 3-space. This shows up as *cracks* during the visualization process, and is an undesirable artifact. This problem can be prevented as follows. Instead of parameterizing just the (u, v, s, t) tuple, we generate a 7-tuple (u, v, s, t, x, y, z) by mapping one pair of domain values to $\mathcal{R}^3$ using the appropriate patch functions. The parameterization of these points in $\mathcal{R}^7$ results in functions $(\mathbf{u}(\alpha), \mathbf{v}(\alpha), \mathbf{s}(\alpha), \mathbf{t}(\alpha), \mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{z}(\alpha))$. During the display process, the new functions are used to map points from domain to $\mathcal{R}^3$ thereby resulting in a consistent boundary between adjacent trimmed patches in the model.

Parameterization of the curve: Given a piecewise linear curve in $\mathcal{R}^7$, our task is to fit a curve of degree n (n is chosen depending on the number of points on the curve and the degree of the two patches intersecting) which lies within a tolerance ϵ to the piecewise linear curve. The main requirements of the curve fitting algorithm are to preserve the topology and not proliferate the number of piecewise curve segments.

The first step in the algorithm is to subdivide the curve into monotonic (in all dimensions) non-intersecting segments. This provides a strategy to preserve the topology of the curve. Further, using monotonic segments to do curve fitting usually gives much better results in terms of deviation from the original curve. The basic idea of fitting a curve of degree n is to choose $(n+1)$ equally spaced points from the given sequence and interpolate the curve through them. The surface intersection algorithm generates roughly equally spaced points on the intersection curve. Therefore, we are justified in assigning equally spaced parameter values (steps of $\frac{1}{n}$) to the chosen points. We use the Bézier-Bernstein basis for our parametric curve. We shall describe the method for one of the dimensions (say u). Let the curve be: $\mathbf{u}(\alpha) = \Sigma_{i=0}^{n} \begin{pmatrix} n \\ i \end{pmatrix} \mathbf{U_i}(1-\alpha)^{n-i}\alpha^i$. $(n+1)$ equations are obtained by substituting appropriate parameter values at each of the chosen points. This results in a linear system with $(n+1)$ unknowns $\mathbf{U_i}$'s $(i = 0, 1, \ldots, n)$. Solving the linear system gives the control points for the parametric curve. The functions for the other variables are obtained similarly. The maximum distance between any point on the piecewise linear curve and the parametric curve is determined by solving a system of non-linear equations using a reduction to the eigenvalue problem [Man93]. This estimates the error between the two curves. If the error is not within the tolerance limit ϵ, the original piecewise linear curve is divided into two halves and the whole process is repeated again.

3 MODEL GENERATION

In this section, we describe the solid modeling system used for computing B-reps from CSG trees. Free–form surfaces have been used previously in modeling systems. The primitives may correspond to polyhedra or solids whose boundaries can be represented in terms of rational spline surfaces.

Our algorithms and systems for boundary evaluation make the following set of assumptions:

- All primitives are *regularized* solids [Hof89] and all Boolean operations result in regularized solids. That is, the closure of the interior of the solid corresponds to the original model.
- The intersection between all pairs of overlapping surfaces is well-conditioned and there are no degeneracies.

The system does not explicitly check whether these assumptions are satisfied. In some cases, it can detect these cases while running the algorithm.

The algorithm for B-rep computation performs a depth first traversal of the CSG tree and computes the B-rep of solids at each intermediate node of the tree. The algorithm for Boolean operation between a pair of solids involves trimmed surface intersection, ray-shooting, adjacency graph computation and surface normal orientation of the resulting solid. In this section we give a brief overview of the algorithm.

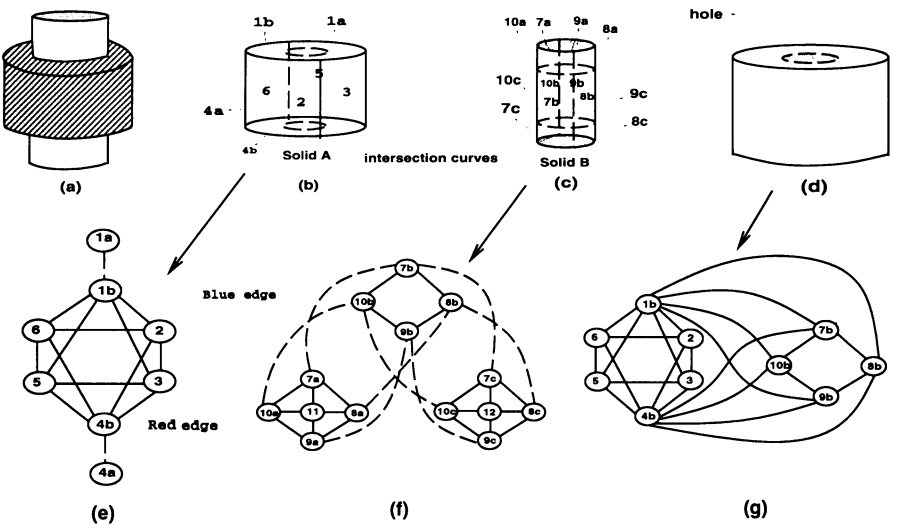

Figure 2 Adjacency Graph Computation for a Difference Operation

3.1 Surface Intersection

A basic operation in the computation of B-reps from a CSG model is the intersection of two surfaces. Given two Bézier surfaces, $\mathbf{F}(s,t) = (X(s,t), Y(s,t), Z(s,t), W(s,t))$ and $\mathbf{G}(u,v) = (\overline{X}(u,v), \overline{Y}(u,v), \overline{Z}(u,v), \overline{W}(u,v))$, represented in homogeneous coordinates, the intersection curve is defined as the set of common points in $\mathcal{R}^3$. However, the degree of the intersection curve can be as high as $4m^2n^2$ for two $m \times n$ tensor-product surfaces and the curve cannot be exactly represented as a Bézier curve (for most cases) [Hof89]. Typical values of m and n are two or three and can be as high as ten or fifteen in practice. In addition, the intersection curve may consist of multiple components, closed loops, singularities etc., which add to its geometric complexity. A number of algorithms based on subdivision, lattice evaluation and marching [SN91, Hoh91, Hof89, KM94] are known. These algorithms compute piecewise linear or spline approximations of the intersection curve. However, when it comes to computing the B-reps of CSG models defined using a series of Boolean operations, these representations may not guarantee robustness, accuracy or consistent representation.

3.2 Intersection between Trimmed Surfaces

Given two solids, $\mathbf{S}_1$ and $\mathbf{S}_2$, with m and n surfaces, respectively, the algorithm for Boolean operations initially computes the intersection curve between their boundaries. This includes computation of intersection between all possible overlapping surfaces. In the worst case, all the mn pairs can intersect, but that is uncommon. To speed up the computation, the algorithm first checks bounding volumes and convex hulls for intersection. It encloses each surface by an axis-aligned bounding box, projects the bounding box along the X, Y and Z axes, sorts the resulting intervals and computes the overlapping bounding boxes.

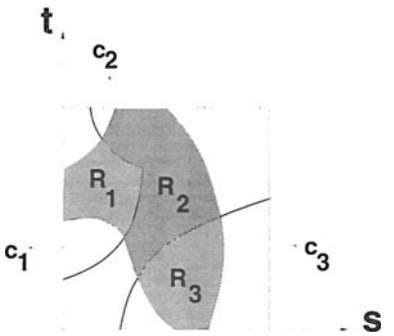

Figure 3 Partitioning the domain based on intersection

Each Bézier surface is contained in the convex hull of its control points [Far93]. For each pair of overlapping bounding boxes, the algorithm checks whether the convex hulls of their control points intersect. This is reduced to a linear programming problem in three dimensions and solved using Seidel's incremental linear time algorithm [Sei90].

Let $\mathbf{F}(s,t)$ be a trimmed surface and its trimmed subset in the domain $0 \leq s,t \leq 1$ be bounded by curve segments $t_1, t_2, \ldots, t_k$. To compute its intersection with another trimmed or non-trimmed surface $\mathbf{G}(u,v)$, it initially treats each parameterization as a non-trimmed surface and computes all the components of the intersection curves in the domain $0 \leq s,t,u,v \leq 1$ [KM94]. Let these components be denoted by $c_1, c_2, \ldots, c_p$, as shown in Fig. 3. Let the corresponding components in the domain of $\mathbf{G}(u,v)$ be $\overline{c}_1, \overline{c}_2, \ldots, \overline{c}_p$. Given these components, it computes a partition of the domain of $\mathbf{F}(s,t)$ using the following algorithm:

1. Compute a dense linear approximation for each algebraic space curve and a polygonal approximation (P) of the domain of $\mathbf{F}(s,t)$. Let Q be the corresponding polygonal approximation of the domain of $\mathbf{G}(u,v)$. Compute a triangulation of the domain (we use Seidel's triangulation algorithm [Sei91]).
2. Using a dense linear approximation of c_i and $\overline{c}_i$, compute their maximal subsets that lie within P and Q. Approximations to the intersection points (with either P or Q) are computed between the resulting line segments. Fig. 3 shows this process on the domain of $\mathbf{F}(s,t)$.
3. Each point of intersection is used as a starting guess for the accurate intersection between the algebraic representation of the trimming curves. The actual intersection is represented as a solution of six algebraic equations in six unknowns and computed using Newton's method. The accurate solution is used to split the intersection curve and the trimming curves.

The new trimming curves arising from intersection with various surfaces are merged together and eventually the domain of intersecting surfaces is partitioned into two or more regions. The boundaries of each component are composed of portions of t_i's and c_i's.

3.3 Computation of the New Solid

The location of a point with respect to a solid (in/out) is a fundamental operation in the computation of the new solid. Given a point $\mathbf{P}$, we shoot a ray from the point in any direction and compute all its intersections with the boundaries of the solid. If the number of intersections is odd, the point is inside the solid, otherwise it is outside. The intersection of rays with trimmed Bézier surfaces is computed using eigenvalue methods [KM94]. Checking whether the resulting point lies inside the trimmed domain is based on planar triangulation [Sei91].

The intersection curves between the boundaries of the two solids partition the surfaces into multiple regions. These regions have the property that all points in their interior are either inside or outside the other solid. Further, it is easy to show that two adjacent regions, separated by the intersection curve, cannot *both* lie inside or outside. Depending on the Boolean operation (union, intersection or difference), the regions composing the B-rep of the solid are chosen. In practice, doing a containment classification test (inside/outside test) for each region of $\mathbf{S}_1$ and $\mathbf{S}_2$ is expensive. The number of surfaces and regions tends to grow rapidly with each Boolean operation. To speed up the process we make use of adjacency graphs of the two solids, $\Gamma(\mathbf{S}_1)$ and $\Gamma(\mathbf{S}_2)$. The partitioning of surfaces into multiple regions induced by the intersection curves changes the structure of the adjacency graphs. For example, in Fig. 2(e) the vertex 1 corresponding to a surface in solid $\mathbf{S}_1$ is split into regions corresponding to $1a$ and $1b$. New adjacencies are introduced between the vertices due to intersection curves. We refer to the new edges as *red* edges and the original set of edges as *blue* edges. All the red edges in Fig. 2(e) and 2(f) are shown using dashed lines. After computing the new vertices and edges, the algorithm computes all connected components of the graph considering only the blue edges. For example, the subgraph consisting of vertices $7a, 8a, 9a, 10a, 11$ represents one connected component in Fig. 2(f). Each connected component of one solid lies completely inside or outside the other solid. We perform ray shooting tests on one of these components. Based on the result, we propagate it to the rest of the graph such that no two adjacent components have the same result. The adjacency graph of the solid resulting from the difference operation is shown in Fig. 2(g). The new edges between these solids correspond to pairs of intersecting surfaces between $\mathbf{S}_1$ and $\mathbf{S}_2$ (e.g. 1b and 7b).

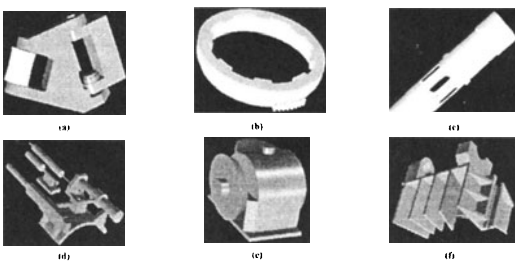

Figure 4 Application of the Algorithm and System to Different Solids

4 DISPLAY SYSTEM

In this section, we briefly describe the display algorithm [KM95] used for trimmed Bézier surfaces. This algorithm tessellates the surfaces into triangles and renders these triangles using the standard graphics system. This algorithm relies on the dual trim curve representation to avoid cracks in the image between adjacent surfaces along the common boundaries and intersections curves. The system makes use of algorithms for back-patch culling [KML95] (an extension of back-face culling for polygons to Bézier patches), bounds for sampling the surface [KML95] and coherence between successive frames for Bézier surfaces.

In brief, the rendering algorithm for trimmed patches is as follows:

1. Create rectangular cells from the uniform tessellation of the surface based on TOL for the surface, where TOL is the user specified tolerance in the screen space.
2. **For each** trimming curve:

   (a) Compute the required tessellation for the space curve: $n_t(TOL)$.
   (b) Tessellate the domain curve into $n_t(TOL)$ straight line segments.
   (c) March along the piecewise linear curve segments marking all the domain cells it crosses. Since the trimming curves are tessellated into piecewise linear segments, there is no need to calculate the exact intersections of the high degree algebraic curves with the cell boundaries. We only need to do the following:

      i. Identify the cells whose bounding edges are intersected by the curve segments.
      ii. For each bounding edge of the rectangular cell, maintain the order in which the segments cross that cell.
      iii. If the trimming curve is contained in a cell, it intersects no cell edges. Mark such cells also.

3. Triangulate each unmarked cell that lies in the trimmed region of the surface by adding a diagonal.
4. The marked cells form a staircase like polygonal chain*. the trimming curve forms another polygonal chain offset from the staircase. Partition this region across cell edges.
5. What results is a set of planar straight line graphs (PSLGs). Triangulate each PSLG. (We use Siedel's algorithm [Sei91].)
   To prevent cracks in the display, no additional points on the curve are introduced in the region partitioning step or the triangulation step.
6. Evaluate the uniformly sampled points on the surface. Evaluate the $n_t(TOL)$ uniformly sampled points on the space curve. These are the vertices of the final triangles.

5 IMPLEMENTATION AND PERFORMANCE

The algorithms presented above have been implemented and applied to a number of solids comprising the model of a submarine storage and handling system, made available to us by

*This chain can be degenerate, e.g. it is null if a trimming curve is contained in a cell.

Model	Number of primitives	Number of CSG operations	Running time(in min.)	Number of patches (in B-Rep)
Fig. 4(a)	23	20	2.6	137
Fig. 4(b)	6	5	0.8	89
Fig. 4(c)	6	5	0.7	116
Fig. 4(d)	28	27	3.4	169
Fig. 4(e)	11	10	1.1	69
Fig. 4(f)	22	21	3.1	146

Table 1 Performance of the Solid Modeling System

Electric Boat, a division of General Dynamics. The model consists of about 2,000 solids. Many of the primitives are composed of polyhedra and conicoids like spheres, cylinders. Additional primitives include prisms and surfaces of revolution of degrees six and more. A few of the primitives are composed of Bézier surfaces of degree as high as twelve. Most of the CSG trees have heights ranging between six and twelve and some of them are as high as 30. The B-reps of many of the solids consist of more than $40-45$ trimmed Bézier surfaces and some of them have up to 148 surfaces.

5.1 Model Generation

The running time of the system varies on different solids. In particular, it depends on the number of Boolean operations, number of intersecting pairs of surfaces and the number of connected components generated. In most cases, it spends about half the time in computing intersections between pairs of surfaces and the other half in ray-shooting and computing the components of new solids. Its performance on a small subset of solids from the model has been highlighted in Table 1. The solids are shown in Fig. 4.

A major problem in the application of our system to different models is numerical accuracy of computations and its impact on the robustness of the entire system. The problem of building robust solid modeling systems based on floating-point computation is fairly open and no good solutions are known [Hof89]. In our case, the algorithm uses ϵ-tolerances at different parts of the overall algorithm. Depending on the values of the tolerances, the robustness of the algorithm can vary considerably.

As an input, the user specifies a set of four tolerance values and they are used as part of the surface intersection algorithm, for ray-shooting, merging intersection curves and detecting planar overlaps. The surface intersection algorithm normalizes the input surface parameterizations and ensures that the output of the intersection algorithm has certain digits of accuracy. The intersection algorithm is based on iterative numerical algorithms and we set the termination criterion accordingly. Similar criteria are used in computing the intersection of trimming curves represented as piecewise algebraic curves. At the end of every CSG operation, the system makes sure that the topology of the resulting solid is consistent (*i.e.*, the solid boundary partitions $\mathcal{R}^3$ into two or more regions).

5.2 Display

The display system has been implemented on an SGI Onyx and Pixel-Planes 5 graphics system. On Pixel-planes 5 it uses multiple graphics processors (GP's) for visibility computations, evaluating Bézier functions and triangulating polygons. The trimmed Bézier surfaces are evenly distributed over different GP's and the system associates each surface with the parent solid for visibility computations. Each GP has about $2.5 Megabytes$ of memory for storing the surface representations and caching the triangle vertices and their normals. The system uses a dynamic memory allocation scheme for caching triangles.

The rendering algorithm produces topologically correct triangulations. As we zoom in or out on a model, it produces varying levels of detail incrementally and *no visual artifacts* appear. The trimming algorithm also works for trimming curves represented as piecewise linear or spline curves. Compared to Rockwood et al.'s algorithm [RHD89], our trimming algorithm is faster by a factor of seven or eight on models consisting of about 200 surfaces on an SGI Onyx. It also produces fewer triangles. We used SGI's GL library implementation based on the [RHD89] algorithm for comparison.

We evaluated the performance of our system on parts of a submarine storage and handling system. The model was designed using Catia CAD system. This version of Catia does not support Boolean operations on curved geometries † and, therefore generates a dense polygonal B-rep for each model. Pixel-Planes 5 graphics system can barely render $3-10$ frames per second on the polygonal representation on a combination of these models ($389,721$ polygons). Furthermore, the image quality is poor when we zoom onto a part of a model. On the other hand our display system can render the multiresolution representation ($20,928$ trimmed Bézier surfaces) at $12-25$ frames a second at good resolutions.

6 ACKNOWLEDGEMENTS

We are grateful to Fred Brooks, Anselmo Lastra and members of UNC Walkthrough group for productive discussions. The CSG model of the submarine storage and handling system was provided to us by Greg Angelini, Jim Boudreaux and Ken Fast at Electric Boat; thanks goes to them. This research is supported in part by Alfred P. Sloan Foundation Fellowship, ARO Contract P-34982-MA, NSF Grant CCR-9319957, NSF Grant CCR-9625217, ONR Contract N00014-94-1-0738, ARPA Contract DABT63-93-C-0048 and NSF/ARPA Center for Computer Graphics and Scientific Visualization

REFERENCES

[AES91] S.S. Abi-Ezzi and L.A. Shirman. Tessellation of curved surfaces under highly varying transformations. *Proceedings of Eurographics'91*, pages 385–97, 1991.

[BR94] C.L. Bajaj and A. Royappa. Triangulation and display of rational parametric surfaces. In *Proceedings of Visualization'94*, pages 69–76, IEEE Computer Society, Los Alamitos, CA, 1994.

†The latest version of Catia does support curved geometries.

[Coh83] E. Cohen. Some mathematical tools for a modeler's workbench. *IEEE Computer Graphics and Applications*, 3(7):63–66, 1983.

[Far93] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press Inc., 1993.

[Hof89] C.M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, California, 1989.

[Hoh91] M.E. Hohmeyer. A surface intersection algorithm based on loop detection. *International Journal of Computational Geometry and Applications*, 1(4):473–490, 1991. Special issue on Solid Modeling.

[KM94] S. Krishnan and D. Manocha. An efficient surface intersection algorithm based on the lower dimensional formulation. Technical Report TR94-062, Department of Computer Science, University of North Carolina, 1994.

[KM95] S. Kumar and D. Manocha. Efficient rendering of trimmed NURBS surfaces. *Computer-Aided Design*, pages 509–521, 1995.

[KML95] S. Kumar, D. Manocha, and A. Lastra. Interactive display of large scale NURBS models. In *Proc. of ACM Interactive 3D Graphics Conference*, pages 51–58, 1995.

[KNM95] S. Krishnan, A. Narkhede, and D. Manocha. Boole: A system to compute boolean combinations of sculptured solids. Technical Report TR95-008, Department of Computer Science, University of North Carolina, 1995.

[LC93] W.L. Luken and Fuhua Cheng. Rendering trimmed NURB surfaces. Computer science research report 18669(81711), IBM Research Division, 1993.

[Man93] D. Manocha. Solving polynomial systems for curve, surface and solid modeling. In *ACM/SIGGRAPH Symposium on Solid Modeling*, pages 169–178, 1993.

[RHD89] A. Rockwood, K. Heaton, and T. Davis. Real-time rendering of trimmed surfaces. In *Proceedings of ACM Siggraph*, pages 107–17, 1989.

[RR92] A.A.G. Requicha and J.R. Rossignac. Solid modeling and beyond. *IEEE Computer Graphics and Applications*, pages 31–44, September 1992.

[Seg90] M. Segal. Using tolerances to guarantee valid polyhedral modeling results. In *Proceedings of ACM Siggraph*, pages 105–114, 1990.

[Sei90] R. Seidel. Linear programming and convex hulls made easy. In *Proc. 6th Ann. ACM Conf. on Computational Geometry*, pages 211–215, Berkeley, California, 1990.

[Sei91] R. Seidel. A simple and fast randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry Theory & Applications*, 1(1):51–64, 1991.

[SN91] T.W. Sederberg and T. Nishita. Geometric hermite approximation of surface patch intersection curves. *Computer Aided Geometric Design*, 8:97–114, 1991.

[Til83] W. Tiller. Rational B-splines for curve and surface representation. *IEEE Computer Graphics and Applications*, 3(6):61–69, 1983.