

# Three-Dimensional Halfspace Constructive Solid Geometry Tree Construction from Implicit Boundary Representations

Suzanne F. Buchele  
Southwestern University  
Georgetown, Texas

bucheles@southwestern.edu

Richard H. Crawford  
The University of Texas at Austin  
Austin, Texas

rhc@mail.utexas.edu

## ABSTRACT

This paper presents a new method to compute constructive solid geometry (CSG) tree representations of an object whose faces consist of planar and non-planar surfaces. The algorithm described accepts as input a valid boundary representation of an object consisting of piecewise implicit surfaces, and computes a halfspace CSG representation of the object. A class of objects that are *describable* by the surfaces bounding them are valid input for the algorithm of this work, although methods currently exist to compute the additional information necessary to process *non-describable* quadric objects as well. This work builds on and complements the other work in this area, in which *dominating halfspaces* are used to simplify the b-rep to CSG conversion process. We include *factored faces* to enable the factorization of dominating halfspaces throughout the algorithm. Thus, an efficient disjoint decomposition of the solid is obtained as a matter of course in the algorithm, so that CSG minimization is generally not necessary.

This work is motivated by reverse engineering of mechanical parts, in which a model of a part is recovered from information obtained by some sort of sensing technique (e.g. CAT scanning, laser range finding). The recovery of a valid CSG-tree description of an object from a boundary representation of it can provide useful information to an engineer in the area of reverse engineering and in other areas related to solid modeling as well. The CSG tree also provides a relatively neutral representation that can enhance form feature recognition and translation.

## Categories and Subject Descriptors

I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—*boundary representations; curve, surface, solid, and object representations*; J.6 [Computer Aided Engineering]: [computer-aided design]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SM'03, June 16–20, 2003, Seattle, Washington, USA.  
Copyright 2003 ACM 1-58113-706-0/03/0006 ...\$5.00.

## General Terms

Algorithms, Design

## Keywords

Representation conversion, reverse engineering, geometric and topological representations, product and assembly modeling

## 1. INTRODUCTION AND MOTIVATION

There are two primary focuses for the conversion of boundary representations (b-reps) to halfspace Constructive Solid Geometry (CSG) tree representations, namely the representation of 3-D solids in CAD systems, and reverse engineering of 3-D solids.

CAD systems typically use a combination of object representation techniques, including boundary representations (b-reps), in which objects' boundaries and other neighborhood and orientation information is stored, and CSG representations, in which a tree representation is stored using solid primitives and Boolean operations [12]. Hybrid CAD systems, in which some combination of representations are used in tandem or conjunction with each other in a single CAD system is the norm for current CAD modelers. To facilitate the most efficient and flexible computations in a CAD system, the conversion between representations may be useful. Algorithms are well-known for the CSG to b-rep conversion process, primarily to facilitate rendering of solid models by the CAD systems. The problem of b-rep to CSG conversion has historically been considered too difficult for general purpose implementation, although fundamental research and development of a general purpose b-rep to CSG algorithm has been presented [14, 15, 16, 21].

Reverse engineering of mechanical parts is the general process of recovering a model of a physical mechanical object from information obtained by some sort of sensing technique. These sensing techniques include, but are not limited to, CAT and MRI scans, laser rangefinder scans, stereoscopic sensing, and contact sensing devices. Reverse engineering may be needed when parts do not conform to their associated model, or when an original CAD model of the part does not exist. Most current advances in reverse engineering of mechanical parts have focused on recovering a boundary representation (b-rep) of an object. Recovered boundary representations of these surfaces can vary from a triangulation of points to piecewise parametric surfaces (common representations include parametric spline surfaces and NURBS surfaces) to piecewise algebraic surfaces [5]. For

reverse engineering, a boundary representation may fail to sufficiently recover the original geometry of the object due to the varying range of types of surfaces fit, the orientation of surface patches, and inherent limitations in parametric representations. The extraction of a CSG representation of a solid may be considered a “user friendly” form for the geometric data in the field of reverse engineering. It is easier than a boundary representation of the object for an engineer to parameterize and manipulate, and could be entered into CSG, b-rep, and hybrid modeling CAD systems through a CSG user interface.

## 2. DEFINITIONS

A halfspace  $\psi$  of  $\mathbb{R}^3$  is a set of the form  $\psi = \{(x, y, z) : g(x, y, z) \geq 0\}$  for some function  $g : \mathbb{R}^3 \rightarrow \mathbb{R}$ . When the function  $g$  is restricted to be a plane, then the halfspace  $\psi$  associated with  $g$  is a halfplane. In this work we consider only halfspaces that are non-empty, regular, pathwise connected sets in  $\mathbb{R}^3$ , such that the regularized complement halfspace  $\bar{\psi}^*$ , is also a non-empty pathwise connected set (and is regular by definition). We refer to the surface associated with a halfspace  $\psi = \{(x, y, z) : g(x, y, z) \geq 0\}$  defined by some function  $g$ , as  $S_\psi = bd(\psi) = \{(x, y, z) : g(x, y, z) = 0\}$ , and we call the set  $S_\psi$  the *surface of  $\psi$* . We say the surface  $S_\psi = \{(x, y, z) : g(x, y, z) = 0\}$  defined by the function  $g$  induces the two halfspaces  $\psi = \{(x, y, z) : g(x, y, z) \geq 0\}$  and  $\bar{\psi}^* = \{(x, y, z) : g(x, y, z) \leq 0\}$ . Buchele [6] points out that non-degenerate quadric surfaces of a single sheet (ellipsoids, hyperboloids of one sheet, cones, hyperbolic and elliptic paraboloids, and elliptic, hyperbolic, and parabolic cylinders) induce two non-empty regular, pathwise connected sets.

We define a solid to be a regular subset of  $\mathbb{R}^3$ . The boundary of the solid  $\Gamma$  is denoted  $bd(\Gamma)$ . We define the natural surfaces of a solid  $\Gamma$  to be the minimal set of surfaces  $\mathcal{S}$  such that  $bd(\Gamma) \subseteq \mathcal{S}$ . A *face*  $F$  of a solid  $\Gamma$  is a subset of a natural surface  $S$  of the solid such that  $F$  is a maximally connected component of  $bd(\Gamma)$ . If we denote  $\mathcal{F}$  to be the set of all faces of a solid  $\Gamma$ , then we can define the boundary representation, or b-rep, of  $\Gamma$  as the union of all faces of the solid  $\Gamma$ , that is,  $bd(\Gamma) = \bigcup F$  for faces  $F \in \mathcal{F}$ .

The surfaces used in the definition of the boundary, faces, and b-rep of a solid are typically defined either parametrically or implicitly. An implicit (or algebraic) surface of  $\mathbb{R}^3$  is a set of the form  $S = \{(x, y, z) : g(x, y, z) = 0\}$ , for some algebraic function  $g : \mathbb{R}^3 \rightarrow \mathbb{R}$  [17]. We use an implicit surface representation in our work, since we use the fact that the implicit definition induces two regular halfspaces in a convenient notational form. Surfaces represented implicitly also have advantages in the solid modeling environment, such as the ease of point membership classification and the simplicity of extracting geometric information directly from the surface equation [5, 7]. Implicit surfaces have recently gained attention in the fields of computer graphics, solid modeling, and reverse engineering (eg., [1, 4, 3]).

A CSG-tree is a singly-rooted binary tree in which the internal nodes are the regularized set operations union ( $\cup^*$ ), intersection ( $\cap^*$ ), and set difference ( $-^*$ ), or rigid-body transformations, and the leaves are regular sets. In  $\mathbb{R}^3$  there are two types of CSG-tree representations, halfspace and bounded solid CSG representations. Halfspace CSG representations operate on regular sets that are halfspaces, commonly algebraic halfspaces in which an algebraic function

$g : \mathbb{R}^3 \rightarrow \mathbb{R}$  separates  $\mathbb{R}^3$  into two pieces. Bounded solid CSG representations operate on a predefined set of canonical solid primitives which are also regular sets.

## 3. PREVIOUS WORK

In 2-D, the problem of b-rep to CSG conversion is considered solved for cases with modest and realistic limitations ([11, 20, 15, 14]). In 3-D, early or limited attempts to obtain CSG-type information from b-reps of an object were done by Lin and Chen [9], Woo [22] and Tang and Woo [18, 19], and Menon and Kim [10].

Shapiro and Vossler present a solution to b-rep to halfspace CSG conversion for 3-D solids [14, 16, 21], in which they solve the general problem of converting from a boundary representation (b-rep) of a solid to a halfspace CSG representation of it, for solids defined as regular subsets of  $\mathbb{R}^3$  bounded by quadric surfaces. Central to Shapiro and Vossler’s work is the idea of a *canonical intersection term*. In  $\mathbb{R}^3$  (or  $\mathbb{R}^d$  for  $d \geq 1$ ), a collection of  $n$  sets (or, specifically, halfspaces) partition  $\mathbb{R}^3$  into  $2^n$  subsets (some possibly empty). We can represent these  $2^n$  subsets, or, cells, as intersections of the  $n$  halfspaces and their complements.

Also central to Shapiro and Vossler’s work is their Describability Theorem ([15, 14, 16, 21]). A set  $\Gamma \subseteq \mathbb{R}^3$  is *describable* by halfspaces  $\Psi = \{\psi_1, \dots, \psi_N\}$  if there exists a (halfspace) CSG expression for  $\Gamma$  containing only halfspaces in  $\Psi$ . Shapiro and Vossler’s Describability Theorem is as follows: given a set  $\Psi = \{\psi_1, \dots, \psi_N\}$  of halfspaces such that  $bd(\Gamma) \subset S_{\psi_1} \cup^* \dots \cup^* S_{\psi_N}$ , solid  $\Gamma$  is (halfspace) describable by  $\Psi$  iff for each canonical intersection term, all points in the canonical intersection term have the same point membership classification with respect to  $\Gamma$ . Shapiro and Vossler also use the concept of *separating halfspaces*. A separating halfspace is defined as a halfspace that is necessary in the (halfspace) CSG representation of a solid but is not a natural halfspace of it. Not all solids can be represented as a halfspace CSG tree in which leaves consist only of natural halfspaces of the solid. Shapiro and Vossler [16, 21] present an algorithm to compute any necessary separating halfspaces to represent the solid when the solid is not describable by its natural halfspaces alone.

Shapiro and Vossler point out that *dominating halfspaces* can simplify the CSG representation of a solid if factored out at the beginning of the algorithm. A dominating halfspace  $\psi_i$  of a solid  $\Gamma$  is a halfspace such that  $\psi_i \subseteq \Gamma$ . Similarly, a dominating halfspace  $\psi_j$  of the regularized complement of a solid  $\Gamma$ ,  $\bar{\Gamma}^*$ , is a halfspace such that  $\psi_j \subseteq \bar{\Gamma}^*$ .

The basic b-rep to CSG approach of Shapiro and Vossler is to first factor any dominating halfspaces of the solid or the complement of the solid, deriving a resultant solid after each pass. When no more dominating halfspaces can be factored from the original or a subsequent resultant solid, they compute all canonical intersection terms induced by the remaining halfspaces. If any canonical cell contains points that are both interior and exterior to the solid, separating halfspaces are computed and the canonical cells are re-evaluated. Once a set  $\Psi$  of sufficient and necessary halfspaces is determined and the canonical CSG representation of the solid or resultant solid is determined, Shapiro and Vossler then attempt to minimize this halfspace CSG representation using prime implicants. [14, 2]

Our b-rep to CSG development builds on and complements Shapiro and Vossler’s theory in a number of ways.

Buchele [6] introduces a new method for b-rep to halfspace CSG conversion to allow for BSP and bounded solid CSG conversion from a b-rep of an object as well as halfspace CSG conversion. The BB family of algorithms convert from a b-rep to a BSP (Binary Space Partitioning) tree representation of the solid (the BB algorithm) and then converts from the BSP tree to a halfspace CSG tree representation as well (the BBHC algorithm). We consolidate Buchele’s BB and BBHC algorithms here, bypassing the BSP to halfspace CSG conversion.

## 4. B-REP TO HALFSPACE CSG CONVERSION

### 4.1 Overview of BHC Algorithm

We present a new algorithm based on Buchele’s BB and BBHC algorithms [6] to directly convert from a b-rep of a 3-D solid to a CSG tree representation of it. We call this consolidated algorithm BHC, for B-rep to Halfspace CSG conversion. Like Shapiro and Vossler’s algorithm, we take advantage of dominating halfspaces and factor them from the representation at the start of the algorithm. Because of our addition of *factored faces*, we do not fully compute a resultant solid after each pass of dominating halfspace factorization, and we are able to factor dominating halfspaces throughout the algorithm. Thus, a simplified factorization is generally possible as a matter of course in the algorithm.

A stack of regions to be processed (initially one region, all of  $\mathbb{R}^3$ ) is maintained. Only one region is the current region at any time. Regions are popped from the stack as they are processed, and any resulting regions to be processed in the future are pushed onto the stack. Along with the region is stored the *bounding structure*, which gives enough information about the current solid on that region, so that computing a complete boundary representation of the current (partially factored) solid is not necessary. The bounding structure  $\mathcal{B}$  for a region consists of all original faces that intersect the interior of the region ( $\mathcal{F}_O$ ), all factored faces that have been associated with the region ( $\mathcal{F}_F$ ), and all halfspaces that may be used to factor the current region ( $\Psi$ ).

Halfspaces are factored one at a time in the algorithm. At each iteration, we check to see if a dominating halfspace can be factored from the current representation of the current solid. If a dominating halfspace cannot be factored, we choose a partitioning halfspace, and split the solid and region into two pieces. Each piece is then treated as a solid in a recursive call, and dominating halfspaces are factored from each piece if possible.

Factored faces are subsets of surfaces of halfspaces that have been factored from the solid. They are used to assure the overall correctness of the algorithm. The purpose of factored faces is to identify regions of space that are inside or outside the current solid. We identify four types of factored faces, called In, In/Out, Out, and Out/In factored faces. Specifically, In and Out/In factored faces alert us to regions of space that must be correctly identified as “in” the resulting solid, while Out and In/Out factored faces alert us to regions of space that must be correctly classified as “out” of the resulting solid. In/Out and Out/In factored faces are identified whenever a halfspace is factored whose surface contains an original face of the solid. In addition, In

and Out factored faces are identified whenever a non-planar halfspace is factored.

We identify dominating halfspaces using the bounding structure and factored faces in the following manner. A halfspace  $\psi$  is a dominating halfspace of a current solid  $\Gamma_{current}$  if there are no original faces of the solid, or In/Out or Out factored faces of the solid, on the interior of the halfspace ( $int(\psi)$ ). Similarly, a halfspace  $\psi$  is a dominating halfspace of the regularized complement of the solid,  $\overline{\Gamma}_{current}^*$ , if there are no original faces of the solid, or Out/In or In factored faces of the solid, on  $int(\psi)$ .

Whenever we factor a halfspace, we remove any original faces completely contained in the halfspace surface from the current bounding structure of the current region  $\lambda$ , and add the factored faces to the bounding structure. If the halfspace  $\psi$  is factored as a dominating halfspace of the solid, one of the resulting regions will be identified as an “in” or “out” region;  $\lambda \cap^* \psi$  if  $\psi$  is a dominating halfspace of the solid, or  $\lambda \cap^* \overline{\psi}^*$  if  $\psi$  is a dominating halfspace of the regularized complement of the solid. In this case the entire bounding structure associated with that region, including any factored faces just identified on that region, will be discarded.

Because the BHC algorithm checks for dominating halfspaces on each pass through each region the object is partitioned into, the resulting halfspace CSG tree is of small size overall. If only dominating halfspaces are factored, the halfspace CSG tree will be of minimal size (the number of leaves in the tree will equal the number of halfspaces in the boundary representation). Each time we partition, we duplicate at least one halfspace in the CSG tree (the one we partitioned on), and split the solid region into two. Because we choose a halfspace to partition by choosing the one that splits a minimal number of faces, halfspaces that must be duplicated on each of the two resulting regions after a partition are minimized, at least in a greedy fashion.

### 4.2 Discussion of BHC Algorithm

We illustrate the BHC algorithm given in Figure 1 using the 2-D example shown in Figure 2 as a running example. In this case, the original solid  $\Gamma$  is shown as the shaded region, the original current region  $\lambda_{current}$  is  $\mathbb{R}^2$ , and  $\Psi = \{\psi_1, \psi_2, \dots, \psi_8\}$ . We begin the procedure by determining the set  $\Psi_{Incl}$  to be the set of all halfspaces whose surface contains original faces ( $\mathcal{F}_O$ ) that lie on the interior, as opposed to the boundary, of  $\lambda_{current}$ . In the initial case,  $\Psi_{Incl} = \Psi$  (Line 2).

If  $\Psi_{Incl} \neq \emptyset$ , that is, if there are original faces on  $\lambda_{current}$  that have not yet been factored, then we construct the set  $\Psi_{Dom}$  to be the set of all dominating halfspaces of  $\Gamma$  and  $\overline{\Gamma}^*$  on  $\lambda_{current}$  (Line 4). A halfspace  $\psi$  is a dominating halfspace of the current solid  $\Gamma_{current}$  if  $\psi$  or  $\overline{\psi}^* \in \Psi$ ,  $\nexists$  face  $F \in \mathcal{F}_O$  such that  $F \cap^* int(\psi) \neq \emptyset$ , and  $\nexists$  face  $F_f \in \mathcal{F}_F$ ,  $F_f$  an In/Out or Out factored face, such that  $F_f \cap^* int(\psi) \neq \emptyset$ . Similarly, halfspace  $\psi$  is a dominating halfspace of the complement of the current solid  $\overline{\Gamma}_{current}^*$  if  $\psi$  or  $\overline{\psi}^* \in \Psi$ ,  $\nexists$  face  $F \in \mathcal{F}_O$  such that  $F \cap^* int(\psi) \neq \emptyset$ , and  $\nexists$  face  $F_f \in \mathcal{F}_F$ ,  $F_f$  an Out/In or In factored face, such that  $F_f \cap^* int(\psi) \neq \emptyset$ . In our example, we find that  $\psi_1, \psi_2$ , and  $\psi_3$  are dominating halfspaces of  $\Gamma$ , and  $\overline{\psi}_5^*$  is a dominating halfspace of  $\overline{\Gamma}^*$ .

Since  $\Psi_{Dom} \neq \emptyset$  then we can factor each of these dominating halfspaces from the solid  $\Gamma$  (Line 6). Pseudo code of **FactorDomHalfspaces** is given in Figure 3. We may choose the first dominating halfspace to factor arbitrarily

### Algorithm BHC

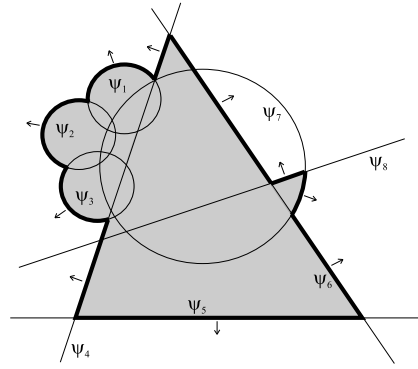
Input: Nil CSG tree,  $\lambda_{current} = \mathbb{R}^3$ ,  
 $\mathcal{B} = \{\mathcal{F}_O = \{F_1, \dots, F_k\}, \mathcal{F}_F = \emptyset, \Psi = \{\psi_1, \dots, \psi_N\}\}$

1. **While**  $\lambda_{current} \neq \emptyset$   
 $\{\lambda_{current} \not\subseteq \Gamma \text{ and } \lambda_{current} \not\subseteq \bar{\Gamma}^*\}$
2.  $\Psi_{Incl} = \text{GetOriginalHalfspaces}(\lambda_{current}, \mathcal{B})$
3. **If**  $\Psi_{Incl} \neq \emptyset$
4.  $\Psi_{Dom} = \text{ComputeDomHalfspaces}(\lambda_{current}, \mathcal{B})$
5. **If**  $\Psi_{Dom} \neq \emptyset$
6.  $\lambda_{new} = \text{FactorDomHalfspaces}(\Psi_{Dom}, \lambda_{current}, \mathcal{B})$
7. **Else**
8. Choose Partitioning Halfspace  $\psi_{part} \in \Psi_{Incl}$
9. **PartitionAndPushRegions** $(\psi_{part}, \lambda_{current}, \mathcal{B})$
10.  $\lambda_{new} = \text{PopPendingRegion}$
11. **End If**
12. **Else**  
 $\{\nexists \text{ original face on } \text{int}(\lambda_{current}), \text{ and } \exists \text{ both}$   
 $\text{In or Out/In and Out or In/Out factored faces}$   
 $\text{on } \lambda_{current}\}$
13. Choose Partitioning Halfspace  $\psi_{part} \in \Psi_{Incl}$
14. **If**  $\psi_{part} = \text{nil}$
15. **Abort** with Not Describable Error
17. **Else**
18. **PartitionAndPushRegions** $(\psi_{part}, \lambda_{current}, \mathcal{B})$
19.  $\lambda_{new} = \text{PopPendingRegion}$
20. **End If**
20. **End If**
21.  $\lambda_{current} = \lambda_{new}$
22. **While** **CheckInOutSolid** $(\lambda_{current})$
23. **TerminateCSGBranch** $(\lambda_{current})$
24.  $\lambda_{current} = \text{PopPendingRegion}$
25. **End While**
26. **End While**
27. **Exit**

**Figure 1: BHC Algorithm for the B-rep to Halfspace CSG Tree Conversion Process**

(Line 1, Figure 3). In practice, we first factor dominating halfspaces of  $\Gamma_{current}$ , and then factor dominating halfspaces of  $\bar{\Gamma}_{current}^*$ . Buchele [6] proved that the order of dominating halfspace factorization does not affect the validity of the previously detected dominating halfspaces.

For each dominating halfspace  $\psi_D$  that we factor, we remove any portions of any factored faces that lie on the interior of the dominating halfspaces from the factored faces (Lines 3-7, Figure 3). Since  $\mathcal{F}_F = \emptyset$  initially, no removal is required. We then remove each original face of the dominating halfspace from the set of original faces in the bounding structure for this region,  $\mathcal{F}_O$  (Line 10). In addition, for each dominating halfspace we create a factored face of type In/Out or Out/In (depending on if the halfspace is a dominating halfspace of  $\Gamma_{current}$  or  $\bar{\Gamma}_{current}^*$ ) (Lines 9,11) and add this factored face (corresponding to the original face) to the set of factored faces in the bounding structure for this region  $\mathcal{F}_F$  (Line 12). If the surface of the face is non-planar, as it is for dominating halfspaces  $\psi_1$ ,  $\psi_2$ , and  $\psi_3$ , we also create a factored face corresponding to the complement of the original face on  $\lambda_{current}$  (Lines 13, 14, 15). Because of the data structure used to store faces, the **CreateComplementFace** procedure is simply a reversal of the edge representation on the surface for each face. If the complement face intersects the current region, as it does in our example, then it is also added to the bounding structure of the current region as an In or Out factored face (Lines 16-19). We then remove the halfspace we have factored (or its complement) from the set of halfspaces in the bounding structure for this region (Line 21). So, in our example we remove  $\psi_1$ ,



**Figure 2: Solid  $\Gamma$ .**

$\psi_2$ ,  $\psi_3$ , and  $\psi_5$  from  $\mathcal{F}_O$  and from  $\Psi$ , we add each of the faces on  $\psi_1$ ,  $\psi_2$ , and  $\psi_3$ , to  $\mathcal{F}_F$  as a factored face of type In/Out (since each halfspace is a dominating halfspace of  $\Gamma_{current}$ ), we add the face of  $\psi_5$  to  $\mathcal{F}_F$  as factored face of type Out/In, (since it is a dominating halfspace of  $\bar{\Gamma}_{current}^*$ ), and we create complement faces on  $\psi_1$ ,  $\psi_2$ , and  $\psi_3$ , and add these complement faces to  $\mathcal{F}_F$  as factored faces of type In.

Lastly, we update the CSG tree. If  $\psi_D$  is a dominating halfspace of  $\Gamma_{current}$ , we add a union node as the next node in the CSG tree (Line 23, Figure 3), and we add  $\psi_D$  as the node's left child (Line 24). If  $\psi_D$  is a dominating halfspace of  $\bar{\Gamma}_{current}^*$ , we add an intersection node as the next node in the CSG tree (Line 26), and add  $\psi_D$  as the node's left child (Line 27). In either case, the node's right child becomes  $\lambda_{current} \cap^* \bar{\psi}_D$ , the next current region (Line 29). Once all dominating halfspaces in  $\Psi_D$  have been factored, we return with the new region  $\lambda_{current}$  being the one that resulted from factoring the last dominating halfspace in  $\Psi_{Dom}$ . Figure 4a shows the CSG tree after the initial dominating halfspace factorizations of our example. After the factoring process, we have  $\Gamma = \psi_1 \cup^* \psi_2 \cup^* \psi_3 \cup^* (\psi_5 \cap^* (\lambda_{new} \cap^* \Gamma))$  for  $\lambda_{new}$  pictured in Figure 4b. Figure 4b also shows the bounding structure of  $\lambda_{new}$ , containing the original faces of  $\Gamma$  remaining on  $\lambda_{new}$ , with arrows indicating surface normals, as well as the factored faces added as a result of our procedure.

Before continuing another pass of the algorithm, we first check if the new  $\lambda_{current}$  is wholly contained in  $\Gamma_{current}$  or  $\bar{\Gamma}_{current}^*$  (Line 22, Figure 1). We can accomplish this by checking if the region contains no original faces of the solid, and contains either only In or Out/In faces, or, only Out or In/Out faces. In this case,  $\lambda_{new}$  is not wholly contained inside or outside the solid, so we continue processing  $\lambda_{new}$  at Figure 1, Line 2. Since we know that the region is not contained in  $\Gamma$  or  $\bar{\Gamma}^*$ , the assertion preceding Line 2 holds.

We continue with with the current region  $\lambda_{current}$  (in our example,  $\lambda_{current}$  is the previous  $\lambda_{new}$  shown in Figure 4b), and  $\Psi = \{\psi_4, \psi_6, \psi_7, \psi_8\}$ . Again,  $\Psi_{Incl} = \Psi$ . On the new  $\lambda_{current}$  there are no dominating halfspaces of  $\Gamma$  or  $\bar{\Gamma}^*$ , so we must choose a partitioning halfspace (Line 8, Figure 1). Note that without the In factored faces resulting from factoring  $\psi_1$ ,  $\psi_2$ , and  $\psi_3$ , we might mistakenly identify  $\psi_4$  as a dominating halfspace of  $\bar{\Gamma}^*$ , or, have to compute the closure of the resultant solid on  $\lambda_{current}$ .

We evaluate each  $\psi \in \Psi_{Incl}$  as to its goodness as a parti-

**FactorDomHalfspaces**( $\Psi_{Dom}, \lambda_{current}, \mathcal{B} = \{\mathcal{F}_O, \mathcal{F}_F, \Psi\}$ )

```

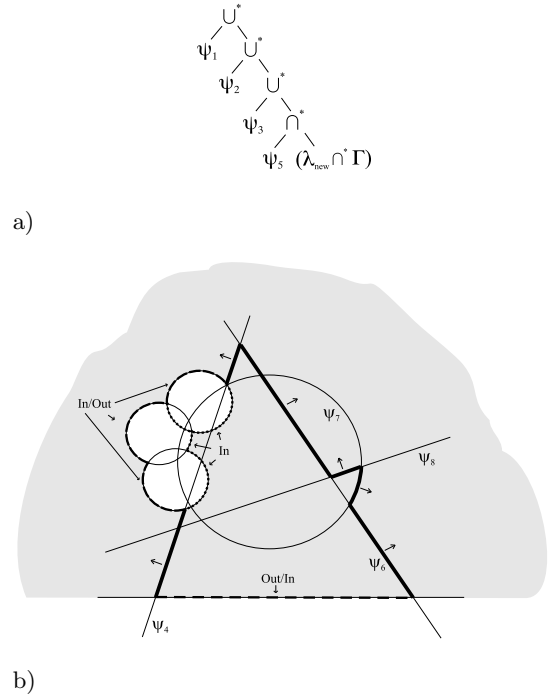
1.  $\psi_D = \text{GetFirstDominatingHalfspace}(\Psi_{Dom})$ 
2. While  $\psi_D \neq \text{nil}$ 
3.   For every face  $F \in \mathcal{F}_F$ 
4.     If  $(F \cap \text{int}(\overline{\psi_D^*}) \neq \emptyset)$ 
5.       Replace  $F$  with  $F \cap \overline{\psi_D^*}$  in  $\mathcal{F}_F$ 
6.     End If
7.   End For
8.    $F = \text{GetHalfspaceFaces}(\psi_D, \mathcal{F}_O)$ 
9.    $F_f = F$ 
10.  Remove Face(s)  $F$  from  $\mathcal{F}_O$ 
11.  Set Face Type of  $F_f$  to In/Out or Out/In
12.  Add Face  $F_f$  to  $\mathcal{F}_F$ 
13.  If  $S_{\psi_D}$  a Non-Planar Surface
14.     $F_C = \text{CreateComplementFace}(F_f)$ 
15.     $F_C = F_C \cap \lambda_{current}$ 
16.    If  $F_C \neq \text{nil}$ 
17.      Set Face Type of  $F_C$  to In or Out
18.      Add Face  $F_C$  to  $\mathcal{F}_S$ 
19.    End If
20.  End If
21.  Remove Halfspace  $\psi_D$  or  $\overline{\psi_D^*}$  from  $\Psi$ 
22.  If  $\psi_D$  dominating halfspace of  $\Psi$ 
23.    AddCSGInternalNode( $\cup^*$ ,  $\lambda_{current}$ )
24.    AddCSGChildNode( $\psi_D$ , left,  $\lambda_{current}$ )
25.  Else
26.    AddCSGInternalNode( $\cap^*$ ,  $\lambda_{current}$ )
27.    AddCSGChildNode( $\psi_D$ , left,  $\lambda_{current}$ )
28.  End If
29.   $\lambda_{current} = \text{GetChildNode}(\text{right}, \lambda_{current})$ 
30.   $\psi_D = \text{GetNextDominatingHalfspace}(\Psi_{Dom})$ 
31. End While
32. Return  $\lambda_{current}$ 

```

**Figure 3: Algorithm to factor dominating halfspaces.**

tioning halfspace, noting that at this point in the algorithm if  $\psi$  is a halfspace in  $\Psi_{Incl}$ ,  $S_\psi$  contains at least a portion of an original face on  $\lambda_{current}$ . For our purposes, we choose a halfspace that minimizes the number of original, In/Out, Out/In, In, and Out faces that must be split in order to perform the partitioning, although another criteria may be used if desired. If some number of halfspaces of  $\Psi_{Incl}$  split the same number of faces, we prefer halfspaces that split the fewest number of original faces. If more than one halfspace fits this criteria, we prefer planar to non-planar halfspaces for partitioning. Otherwise, we choose a partitioning halfspace randomly from the set of halfspaces meeting these criteria. We choose  $\psi_6$  as the partitioning halfspace (since it splits no faces) and partition (Figure 1, Lines 8,9).

Pseudo code of **PartitionAndPushRegions** is given in Figure 5. The overall algorithm is similar to a single pass through the main loop of **FactorDomHalfspaces**, except two resulting sub-regions are generated instead of one. We first insert a union node and two child intersection nodes into the CSG tree (Lines 1-5, Figure 5). The right child of the left intersection node is the partitioning halfspace  $\psi_6$ , while the left child of the right intersection node is the complement of the partitioning halfspace,  $\overline{\psi_6^*}$ . The two resulting regions are  $\lambda_1 = \lambda_{current} \cap \psi_6$ , and  $\lambda_2 = \lambda_{current} \cap \overline{\psi_6^*}$  (Lines 6, 7). Since in this case we know  $\psi_6 \in \Psi_{Incl}$  (Figure 1 Line 8), we know that  $S_{\psi_6}$  contains a face from  $\mathcal{F}_O$  and so in Figure 5, Lines 8 and 9,  $F$  will not be nil. We include the original face as a factored face in both the resulting regions  $\lambda_1$  and  $\lambda_2$  (Lines 11-13). In this case,  $S_{\psi_6}$  is a planar, surface, so we next remove the face of  $\psi_6$ ,  $F$ , from  $\mathcal{F}_O$  (Line 24), and the halfspace  $\psi_6$  from  $\Psi$  (Line 25). We



**Figure 4: a) The CSG tree after initial dominating halfspaces are factored. b) Resulting  $\lambda_{new}$ .**

then split the bounding structure associated with  $\lambda_{current}$  into two bounding structures,  $\mathcal{B}_1 = \{\mathcal{F}_{O_1}, \mathcal{F}_{F_1}, \Psi_1\}$  and  $\mathcal{B}_2 = \{\mathcal{F}_{O_2}, \mathcal{F}_{F_2}, \Psi_2\}$  associated with each resulting region  $\lambda_1$  and  $\lambda_2$ . All halfspaces remaining in  $\Psi$  (all halfspaces associated with  $\lambda_{current}$  except  $\psi_6$ ) are included in each of  $\Psi_1$  and  $\Psi_2$ . Only faces of  $\mathcal{F}_O$ ,  $\mathcal{F}_F$  that intersect the region  $\lambda_1$  are included in  $\mathcal{F}_{O_1}$ ,  $\mathcal{F}_{F_1}$ . Similarly, only faces of  $\mathcal{F}_O$ ,  $\mathcal{F}_S$  that intersect the region  $\lambda_2$  are included in  $\mathcal{F}_{O_2}$ ,  $\mathcal{F}_{F_2}$  (Lines 27-30). Finally, the two resulting regions  $\lambda_1$  and  $\lambda_2$  are pushed as pending regions to be processed in the future (Line 33). The bounding structure information for each region is stored with the region pushed on the stack.

We continue processing (Line 10, Figure 1) by popping one of the pushed regions, say,  $\lambda_1$  shown in Figure 6a.  $\lambda_1$  is not an “in” or “out” cell (Lines 21-26, Figure 1), so we continue processing it as our next  $\lambda_{current}$ .

At this point,  $\Psi_{Incl} = \{\psi_4\}$ , and  $\Psi_{Dom} = \psi_4$  since  $\psi_4$  is a dominating halfspace of  $\Gamma$  on  $\lambda_{current}$  (Lines 2-4, Figure 1). We factor  $\psi_4$  (Line 6), resulting in the CSG tree depicted in Figure 6b and the resulting region  $\lambda_{new}$  shown in Figure 6c. The resulting region  $\lambda_{new}$  is not wholly contained in the solid, so we continue processing with  $\lambda_{current} = \lambda_{new}$  (Lines 21-26).

In this case,  $\Psi = \{\psi_7, \psi_8\}$  and  $\Psi_{Incl} = \emptyset$ , so we proceed to Line 13 in Figure 1. Since  $\Psi_{Incl} = \emptyset$ , then we have a current region that contains no original faces (if it did,  $\Psi_{Incl}$  would not be empty) and contains inconsistent factored faces (Out/In or In, and, In/Out or Out). We know that the region contains inconsistent factored faces, because if it did not, the region would have been classified as wholly inside or outside the current solid. Thus, the assertion before Line 13 holds. We choose a partitioning halfspace (Line 13)  $\psi_{part}$  from  $\Psi$  such that the surface of  $\psi_{part}$  at least partially partitions In/Out and Out factored faces from Out/In

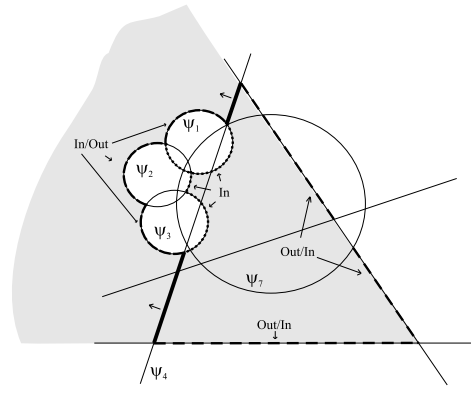
**PartitionAndPushRegions**( $\psi_{part}$ ,  $\lambda_{current}$ ,  $\mathcal{B} = \{\mathcal{F}_O, \mathcal{F}_F, \Psi\}$ )

1. **AddCSGInternalNode**( $\cup^*$ ,  $\lambda_{current}$ )
2. **AddCSGInternalNode**( $\cap^*$ ,  $\lambda_{current}.Left$ )
3. **AddCSGChildNode**( $\psi_{part}$ , right,  $\lambda_{current}.Left$ )
4. **AddCSGInternalNode**( $\cap^*$ ,  $\lambda_{current}.Right$ )
5. **AddCSGChildNode**( $\overline{\psi_{part}}$ , left,  $\lambda_{current}.Right$ )
6.  $\lambda_1 = \lambda_{current} \cap^* \psi_{part}$
7.  $\lambda_2 = \lambda_{current} \cap^* \overline{\psi_{part}}$
8.  $F = \mathbf{GetHalfspaceFace}(\psi_{part}, \mathcal{F}_O)$
9. **If**  $F \neq nil$
10.     **For**  $i = 1, 2$
11.          $F_{f_i} = F \cap \lambda_i$
12.         Set Face Type of  $F_{f_i}$  to In/Out or Out/In
13.         AddFace  $F_{f_i}$  to  $\mathcal{F}_{F_i}$
14.     **End If**
15. **If**  $S_{\psi_{part}}$  a Non-Planar Surface
16.      $F_C = \mathbf{CreateComplementFace}(F)$
17.     **If**  $F_C \neq nil$  **SplitFaceIntoComponents**( $F_C, \mathcal{F}_O$ )
18.     **For** each component  $F_C(j)$
19.         **For**  $i = 1, 2$
20.              $F_{C_i}(j) = F_C(j) \cap \lambda_i$
21.             Set Face Type of  $F_{C_i}(j)$  to In or Out
22.             Add Face  $F_{C_i}(j)$  to  $\mathcal{F}_{F_i}$
23.     **End If**
24.     Remove Face  $F$  from  $\mathcal{F}_O$
25.     Remove Halfspace  $\psi_{part}$  from  $\Psi$
26.      $\Psi_1 = \Psi$ ;  $\Psi_2 = \Psi$
27.     **For** every face  $F \in \mathcal{F}_{O_1}$  or  $\mathcal{F}_{F_1}$
28.         **If** ( $F \cap int(\psi_{part}) \neq \emptyset$ )
29.             Add Face  $F \cap \psi_{part}$  to  $\mathcal{F}_{O_1}$  or  $\mathcal{F}_{F_1}$
30.         **If** ( $F \cap int(\overline{\psi_{part}}) \neq \emptyset$ )
31.             Add Face  $F \cap \overline{\psi_{part}}$  to  $\mathcal{F}_{O_2}$  or  $\mathcal{F}_{F_2}$
32.     **PushPendingRegions**( $\lambda_1, \lambda_2$ )
33. **Return**

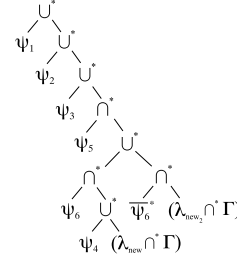
**Figure 5: Algorithm to partition.**

and In factored faces. Of the possible partitioning halfspaces that partially partition factored faces, we choose the halfspace that minimizes the number of factored faces that must be split at partitioning. If some number of halfspaces split the same number of factored faces, we prefer planar to non-planar halfspaces. Otherwise, we choose a partitioning halfspace randomly from the set of halfspaces meeting these criteria. If no partitioning halfspace can be found (that is, no halfspaces of  $\Psi$  partially split the In/Out and Out faces from the Out/In and In faces), then we have detected a region that is not describable by the halfspaces of  $\Psi$  and we abort processing (Line 15). At this point in the algorithm the computation of separating halfspaces might be added to the algorithm, as in Vossler and Shapiro’s method [21], in order to make the solid describable. In this case,  $\psi_7$  partitions the In/Out faces from the In faces, so we partition using  $\psi_7$ .

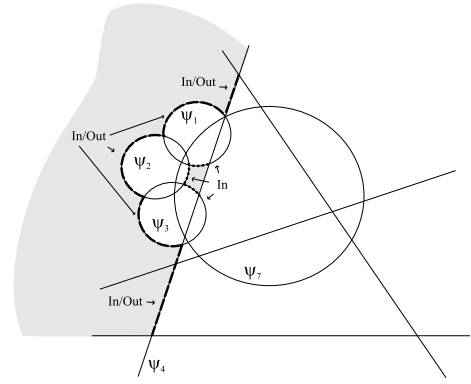
Returning to **PartitionandPushRegions** (Figure 5), we first update the CSG tree (Lines 1-5) and compute the new  $\lambda_1$  and  $\lambda_2$ . The partitioning surface  $S_{\psi_7}$  contains no faces on  $\lambda_{current}$ , so  $F = nil$  in this case (Line 9), and  $F_C = S_{\psi_7}$  (Line 16). Since the partitioning surface is non-planar (Line 15), we split  $S_{\psi_{part}} \cap \lambda_{current}$  into connected surface components (Line 17), and examine each surface component separately. The connected surface components, (determined by **SplitFaceIntoComponents**) are connected surface regions in  $\lambda_{current}$  partitioned by original faces in  $\mathcal{F}_O$ . In this case,  $S_{\psi_{part}} \cap \lambda_{current} = \emptyset$ , so no factored faces are added to both of the next regions (Lines 19-22). We remove  $\psi_7$  from  $\Psi$  (line 25), and put each of the special faces into either the new  $\lambda_1$  or  $\lambda_2$  (Lines 27-30). We push the two resulting re-



a)



b)



c)

**Figure 6: a)  $\lambda_1$  after  $\psi_6$  is partitioned. b) CSG tree after  $\psi_4$  is factored. c) The resulting  $\lambda_{new}$  after  $\psi_4$  is factored.**

gions onto the pending regions stack (Line 31) and continue by popping one of the recently pushed regions in Figure 1 at Line 19. Both of the regions we just pushed are inside and outside the solid (one contains only In/Out faces, the other contains only In faces), so we consolidate the CSG tree by replacing the (internal) parent node of the CSG tree with its left child (Line 23). Since this consolidation is done for both of the recently pushed regions, we obtain the resulting CSG tree shown in Figure 7a.

We continue with Line 2, Figure 1) with the region  $\lambda_{current} = \lambda_2$  shown in Figure 7b. This was the second resulting region after  $\psi_6$  was partitioned. We have  $\Psi_{Incl} = \{\psi_7, \psi_8\}$ , and  $\Psi_D = \{\overline{\psi_8^*}\}$  since  $\overline{\psi_8^*}$  is a dominating halfspace of  $\overline{\Gamma^*}$  on  $\lambda_{current}$ . We factor  $\overline{\psi_8^*}$ , resulting in the CSG tree depicted in Figure 7c. The resulting region  $\lambda_{new}$  is shown in Figure 7d.  $\lambda_{new}$  is not an “in” or “out” cell, so we continue (Line



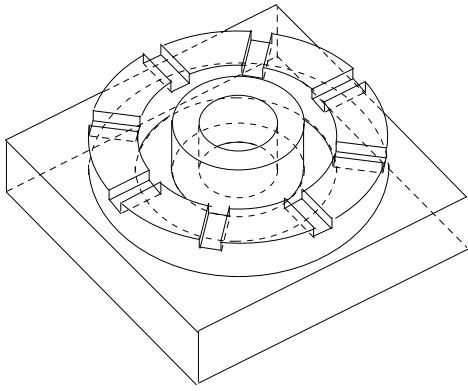


Figure 9: First test object

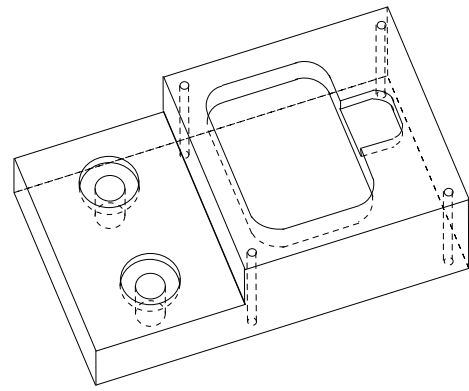


Figure 10: Second test object

representation of the object defined by the input b-rep, using the BHC algorithm. This experimental implementation is designed to work for objects bounded by quadric surfaces of a single sheet, although implementations for more general surfaces (implicit surfaces that induce two non-empty, regular, pathwise-connected halfspaces) may also be developed using the BHC algorithm. The output is a text-formatted CSG tree listed in a pre-order traversal order, in which internal nodes are the operators regularized union and regularized intersection, and leaf nodes are halfspaces. An extension to the BHC algorithm was implemented, in that if separating surfaces are added to the input b-rep and the object is describable using its natural halfspaces plus the separating halfspaces induced by these separating surfaces, the separating halfspaces are used with the natural halfspaces to compute the CSG tree representation of the object.

## 5. TEST RESULTS

The experimental implementation of the BHC algorithm was tested on several objects. Three objects were chosen that tested several aspects of the algorithm, including the detection of non-describable objects. Two objects were chosen that were not describable by their natural halfspaces. In both cases, separating halfspaces were easily determined (by hand) and added to the b-rep for each object. With the addition of the separating halfspace information, a correct halfspace CSG tree representation of the object was then computed by the BHC implementation.

The first test object used is shown in Figure 9. This object was taken from a paper on feature-based design and recognition [8]. The b-rep of the object consists of 21 halfspaces, 45 faces, 114 edges, and 78 vertices. Using the BHC implementation, a halfspace CSG tree was computed consisting of 22 levels, 27 internal nodes (regularized union/intersection operators) and 28 leaves (halfspaces). Although seven of the halfspaces are duplicated in the resulting halfspace CSG tree, the tree is of minimal total size for this object.

The second test object used is shown in Figure 10. This object was taken from a paper on recognizing shape features in solid models [13]. The b-rep of the object consists of 32 halfspaces, 33 faces, 76 edges, and 56 vertices. This object is not describable by its natural halfspaces. To see this, consider the round edges at the corners of the pocket at the top of the object. These rounded edges are blends, that

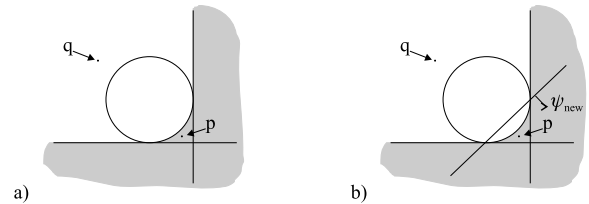
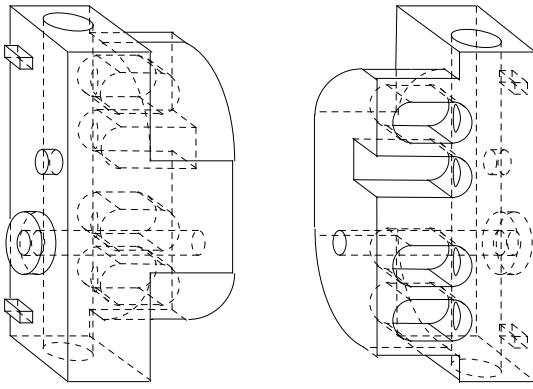


Figure 11: a) 2-D cross section in neighborhood of rounded edge of cavity. b) Same edge with separating halfspace added.

can be thought of as being formed by a cylindrical surface intersecting the neighboring planes in tangent lines. A 2-D view looking down on a cylinder forming a rounded edge is shown in Figure 11a. Point  $p$  shown is inside the solid  $\Gamma$  (the shaded region shown), while point  $q$  is outside the solid  $\Gamma$ . However, points  $p$  and  $q$  lie in the same canonical cell, and so, the solid is not describable. The BHC implementation applied to the b-rep of this object correctly determined that the object is not describable.

We can easily determine a set of separating halfspaces that would make the second test object describable, however. An example of a halfspace added,  $\psi_{new}$ , is shown in Figure 11b. Because of the unique geometry of this object, two of the six separating halfspaces actually coincide. We augmented the original b-rep to include these separating halfspaces, so that the new augmented b-rep of the object consists of the original 32 halfspaces, 33 faces, 76 edges, and 56 vertices of the object, plus the 4 unique separating halfspaces. Using the BHC implementation on this augmented b-rep, a halfspace CSG tree was computed consisting of 35 levels, 38 internal nodes (regularized union/intersection operators), and 39 leaf nodes (halfspaces). Here, three of the halfspaces are duplicated in the resulting halfspace CSG tree, resulting in a nearly minimal total tree size for this object.

The third test object used is shown in Figure 12. This object was taken from the same paper on feature-based design and recognition as the first test object [8]. The b-rep of the object consists of 37 halfspaces, 48 faces, 123 edges, and 85 vertices. This object is not describable by its natural halfspaces, due to the rounded edges in the inner cavities and along the outside of the object. The BHC implementation



**Figure 12: Front and rear view of the third test object.**

applied to the b-rep of this object correctly determined that the object is not describable.

Once again, we can easily determine a set of separating halfspaces that would make the third test object describable, however. Three planar halfspaces are needed. Two planar halfspaces separate the rounded edges along the outside of the object, while one additional planar halfspace separates the rounded edges at the upper side of the inner cavities. The rounded edges at the lower side of the inner cavities are already separated by the plane defining the upper base of the object. We augmented the original b-rep to include these separating halfspaces, so that the new augmented b-rep of the object consists of the original 37 halfspaces, 48 faces, 123 edges, and 85 vertices of the object, plus the 3 separating halfspaces. Using the BHC implementation on this augmented b-rep, a halfspace CSG tree was computed consisting of 24 levels, 55 internal nodes, and 56 leaf nodes. Sixteen of the halfspaces are duplicated in the resulting halfspace CSG tree. Six of these duplicates occur because six partitions are needed for this object. The other ten halfspaces are due to multiple faces on the same halfspace occurring in different regions after partitioning. The resulting tree size is not minimal, but of small total tree size overall.

Compared to Buchele's BB and BBHC algorithms, the consolidated BHC algorithm runs in approximately the same time. The BHC algorithm is slightly faster, due to the elimination of the BSP to halfspace CSG conversion step.

## 6. CONCLUSIONS

This research complements the work of Shapiro and Vossler [15, 14, 16, 21] in solving the problem of computing a halfspace CSG tree representation of a three-dimensional object bounded by quadric surfaces. This work is motivated by the representation of 3-D solids and by the problem of reverse engineering of 3-D solids. The algorithm presented here is an encapsulation of Buchele's BB and BBHC algorithms [6] which compute a binary space partitioning (BSP) tree representation from a b-rep of a solid object, and convert the BSP tree to a halfspace CSG tree representation. Conversion to a bounded solid CSG tree representation of the solid may also be possible, as in Buchele's BBC algorithm [6].

The theory of this work is applicable to any solid whose

bounding surfaces induce two non-empty, regular halfspaces. The BHC algorithm is implemented for solids bounded by non-degenerate quadric surfaces of a single sheet. Many mechanical objects, particularly those manufactured by turning or simple machining operations, consist of such planar and quadric surfaces. Solids whose bounding surfaces are higher degree polynomial surfaces, or non-polynomial surfaces, would work in theory using this method, although the implementation of surface-surface intersection methods for more complex surfaces may limit its usefulness in practice. Because of the direct, halfspace by halfspace factorization process of the algorithm, no post-processing to minimize the resultant CSG tree is necessary. The BHC algorithm naturally produces an efficient CSG representation from the input b-rep.

The current BHC implementation accepts a b-rep of a solid that is describable by its natural halfspaces. Alternatively, a solid that is not describable by its natural halfspaces, but whose b-rep has been appended with separating halfspaces is also acceptable by this algorithm. The computation of separating halfspaces for a quadric solid has been solved by Shapiro and Vossler [16]. The solution to the problem is most completely given in the patented work of Vossler and Shapiro [21]. Thus, the Shapiro and Vossler/Vossler and Shapiro separating halfspace algorithm might be integrated with the BHC algorithm to compute necessary separating halfspaces as non-describable regions of the object are detected.

## 7. REFERENCES

- [1] F. Bernardini, C. L. Bajaj, J. Chen, and S. D. R. Automatic Reconstruction of 3D CAD Models from Digital Scans. *International Journal of Computational Geometry & Applications*, 9(4 & 5):327–369, 1999.
- [2] G. Birkhoff and T. C. Bartree. *Modern Applied Algebra*. McGraw-Hill, 1970.
- [3] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufman, 1997.
- [4] J. Bloomenthal and B. Byvill, editors. *Implicit Surfaces*. Morgan Kaufman, 1996.
- [5] P. Brunet and A. Vinacua. Modeling Closed Surfaces: A Comparison of Existing Methods. In *Mathematical Methods in Computer Aided Geometric Design II*. Academic Press, 1992.
- [6] S. F. Buchele. *Three-Dimensional Binary Space Partitioning Tree and Constructive Solid Geometry Tree Construction from Algebraic Boundary Representations*. PhD thesis, The University of Texas at Austin, 1999.
- [7] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture*. John Wiley & Sons, 1979.
- [8] J. Han and A. A. G. Requicha. Integration of Feature Based Design and Feature Recognition. *Computer Aided Design*, 29(5):393–403, 1997.
- [9] W. Lin and T. Chen. CSG-Based Object Recognition using Range Images. In *Proceedings of the 9th International Conference on Pattern Recognition*, pages 99–103, November 1988.
- [10] S. Menon and Y. S. Kim. Handling Blending Features in Form Feature Recognition using Convex Decomposition. In *Proceedings 14th Annual ASME*

- International Computers in Engineering Conference and Exhibition*, volume 1, pages 79–92, 1994.
- [11] D. P. Peterson. Boundary to Constructive Solid Geometry Mappings: A Focus on 2D Issues. *Computer Aided Design*, 18(1):3–14, 1986.
- [12] A. A. G. Requicha and J. R. Rossignac. Solid Modeling and Beyond. *IEEE Computer Graphics and Applications*, pages 31–44, September 1992.
- [13] H. Sakurai and D. C. Gossard. Recognizing Shape Features in Solid Models. *IEEE Computer Graphics and Applications*, pages 22–32, September 1990.
- [14] V. Shapiro and D. L. Vossler. Construction and Optimization of CSG Representations. *Computer Aided Design*, 23(1):4–20, 1991.
- [15] V. Shapiro and D. L. Vossler. Efficient CSG Representations of Two-Dimensional Solids. *Transactions of the ASME*, 113:292–305, September 1991.
- [16] V. Shapiro and D. L. Vossler. Separation for Boundary to CSG Conversion. *ACM Transactions on Graphics*, 12(1):25–55, January 1993.
- [17] D. M. Y. Sommerville. *Analytical Geometry of Three Dimensions*. Cambridge University Press, 1939.
- [18] K. Tang and T. Woo. Algorithmic Aspects of Alternating Sum of Volumes: Part 1: Data Structure and Difference Operation. *Computer Aided Design*, 23(5):357–366, June 1991.
- [19] K. Tang and T. Woo. Algorithmic Aspects of Alternating Sum of Volumes: Part 2: Nonconvergence and its Remedy. *Computer Aided Design*, 23(6):435–443, July 1991.
- [20] D. L. Vossler. Sweep-to-CSG Conversion using Pattern Recognition Techniques. *IEEE Computer Graphics and Applications*, 5(8):61–68, August 1985.
- [21] D. L. Vossler and V. Shapiro. System and Method for Converting Boundary Representations to Constructive Solid Geometry Representations for Three-Dimensional Solid Object Modeling. Technical report, United States Patent No. 5,537,519, July 1996.
- [22] T. C. Woo. Feature Extraction by Volume Decomposition. In *Proceedings on CAD/CAM Technology in Mechanical Engineering*, pages 76–94, 1982.