ORTHOGONAL POLYHEDRA AS GEOMETRIC BOUNDS IN CONSTRUCTIVE SOLID GEOMETRY

A. Aguilera
Universidad de las Américas-Puebla
Puebla, México
antonio@udlapvms.pue.udlap.mx
aguilera@goliat.upc.es

D. Ayala
Universitat Politècnica de Catalunya
Barcelona, Spain
ayala@lsi.upc.es

Abstract

Set membership classification and, specifically, the evaluation of a CSG tree, are problems of a certain complexity. Several techniques to speed up these processes have been proposed. This include Active Zones, Geometric Bounds and the Extended Convex Differences Tree.

Boxes are the most commonly studied geometric bounds, although other bounds such as spheres, convex hulls and prisms have also been proposed.

On the other hand, there is an extended bibliography dealing with convex polyhedra and solving problems for this class of polyhedra. Orthogonal polyhedra are also a class of polyhedra and several problems have been solved for them.

In this work we propose orthogonal polyhedra as geometric bounds in the CSG model. CSG primitives are approximated by orthogonal polyhedra, and the orthogonal bound of the object is obtained by applying the corresponding boolean algebra. A specific model for orthogonal polyhedra is presented that facilitates a simple and robust boolean operations algorithm between orthogonal polyhedra. This algorithm has linear complexity (is based on a merging process) and avoids floating-point computation.

1 Introduction

Constructive Solid Geometry (CSG) is a non-ambiguous 3D model that allows complicated shapes to be built up from simple ones. This model is represented by a tree in which internal nodes represent boolean regularized operations and leaf nodes represent simple shapes or primitives [15].

Set membership classification [22] and, specifically, the boundary evaluation of a CSG tree, are problems of a certain complexity. Up to now, several accelerating techniques have been proposed to speed up geometric computations in CSG including Active Zones [18], [4] the Extended Convex Differences Tree [14] and Approximating Shapes or Geometric Bounds [8], [5].

The most extensively used geometric bounds are the well-known bounding boxes, although other shapes such as spheres and convex hulls have also been proposed and studied [5].

On the other hand, in several disciplines such as Solid Modeling and Computational Geometry, it is very common to start studying problems with simpler classes of polyhedra rather than the general case. The most commonly chosen class is that of the convex polyhedra. Convexity enables the use of efficient and simple algorithms [13], [6]. Orthogonal polyhedra are a less used simple class. Nevertheless, some works have been published dealing with this simpler class [10], [9], [3]. The restricted class of both convex and orthogonal polyhedra, i.e., orthogonal boxes, have been widely used in many applications [13], [8], [19].

In this work we propose orthogonal polyhedra as geometric bounds in CSG. We define a specific model - the Extreme Vertices (EV) Model - to represent this class of polyhedra. Then, in order to compute the orthogonal bound for a CSG object, we have developed a robust algorithm for regularized boolean operations. This algorithm has linear complexity (it is based on a merging process) and avoids floating-point computations.

The paper is arranged as follows. Sections 2 and 3 deal respectively with geometric bounds and orthogonal polyhedra, and analyze related work on both disciplines. Section 4 defines the Extreme Vertices, EV, model while section 5 describes the corresponding boolean operations algorithm. Section 6 discusses the advantages and drawbacks of using orthogonal polyhedra instead of classical boxes. Finally, section 7 summarizes conclusions and also shows possible directions for future work.

2 Geometric Bounds

A common way of reducing the complexity of geometric computations in CSG is to use geometric bounds or approximating shapes. After fixing a class $\sum$ of approximating shapes, the process to be carried out consists of [8]:

1. All the primitives p in the tree are approximated with their corresponding approximating shape, $p \rightarrow AS(p) \epsilon \sum$
2. A postorder tree traversal is carried out by applying the following rules:
   (a) if $T = T1 \cup T2 \rightarrow AS(T) = AS(AS(T1) \sqcup AS(T2)) = AS(T1) \sqcup AS(T2)$
   (b) if $T = T1 \cap T2 \rightarrow AS(T) = AS(AS(T1) \sqcap AS(T2)) = AS(T1) \sqcap AS(T2)$
   (c) if $T = T1 - T2 \rightarrow AS(T) = AS(T1)$

Hence, the approximating shapes for all the internal nodes and for the root representing the object are determined.

The symbols $\sqcup$ and $\sqcap$ refer to operators equivalent to the boolean operations but closed within the $\sum$ class.

In [4] the S-bound theory is introduced and formally developed in [5]. A class of totally consistent bounding functions is defined and the initial principle working with geometric bounds is extended by the application of the so-called upward and downward rules:

upward rule : the same as the above mentioned postorder tree traversal

downward rule : the geometric bound of each node is refined by intersecting it with the geometric bound of its father,

$$AS(T) = AS(T) \sqcap AS(T.father)$$

Both rules are continuously applied until convergence is reached. For a detailed discussion concerning S-bounds, see [5].

Boxes are the most widely used geometric bounds. A box is defined as:

$$A =< x_{Am}, y_{Am}, z_{Am}, x_{AM}, y_{AM}, z_{AM} >= \{(x,y,z) | x_{Am} \leq x \leq x_{AM}, y_{Am} \leq y \leq y_{AM}, z_{Am} \leq z \leq z_{AM}\}$$

And the corresponding operators are defined as [11]:
$C = A \sqcup B$
where,

$$x_{Cm} = min(x_{Am}, x_{Bm})$$
$$x_{CM} = max(x_{AM}, x_{BM})$$
$$y_{Cm} = min(y_{Am}, y_{Bm})$$
$$y_{CM} = max(y_{AM}, y_{BM})$$
$$z_{Cm} = min(z_{Am}, z_{Bm})$$
$$z_{CM} = max(z_{AM}, z_{BM})$$

and
$C = A \sqcap B$
where

$$x_{Cm} = max(x_{Am}, x_{Bm})$$
$$x_{CM} = min(x_{AM}, x_{BM})$$
$$y_{Cm} = max(y_{Am}, y_{Bm})$$
$$y_{CM} = min(y_{AM}, y_{BM})$$
$$z_{Cm} = max(z_{Am}, z_{Bm})$$
$$z_{CM} = min(z_{AM}, z_{BM})$$

We can easily observe that while the operator $\sqcap$ coincides with the intersection, the $\sqcup$ operator does not correspond to the union operation. The operators $\sqcup$ and $\sqcap$ over the class bounding boxes are a non-distributive lattice instead of a boolean algebra [11].

An order relation can be defined in a lattice such that:

$$a \preceq b \Leftrightarrow a \sqcap b = a$$

and, from lattice theory, the following distributive inequalities are obtained:

$$(a \sqcup b) \sqcap c \succeq (a \sqcap c) \sqcup (b \sqcap c)$$
$$(a \sqcap b) \sqcup c \preceq (a \sqcup c) \sqcap (b \sqcup c)$$

Based on these inequalities, in [11] the authors show that the bounding box size of a CSG depends on the form of its algebraic expression and that the smallest bounding box is obtained when this algebraic expression is in the normal disjunctive form (DF) or union of intersections form (UOI). The authors also show that, in general, this technique produces better bounds than the S-bounds technique. In [7] an algorithm is presented that converts a CSG expression into its DF.

In [16] other advantages of DF are shown:

1. DF only requires a stack of depth 1 and so it can be used for evaluating CSG trees in parallel.

2. When CSG primitives are halfspaces, intersections are convex polyhedra and so the CSG object can be represented as the union of convex polyhedra.

3. We can avoid visiting all the primitives of all intersections. When the intersection currently being visited contains a combination of primitives that resulted in empty bounds for a previously visited intersection, we can state that this current intersection is empty without visiting its remaining terms. This fact is referred to as culling up empty intersections.

Nevertheless, the size of the DF grows exponentially in the number of primitives of the original tree. So, in order to alleviate the need for storing such a large tree, an algorithm is presented in [16] that processes the DF directly from the initial tree.

3 Orthogonal Polyhedra

Orthogonal polyhedra (OP) are polyhedra with all their faces oriented in three orthogonal directions. In this work we will consider only two-manifold OP.

This class of polyhedra implies a restriction of the general case concerning the geometry. In an OP, all planes and lines are parallel to three orthogonal axes and the number of incident edges for any vertex can be only three, four or six [9]. These geometric characteristics make OP a more restricted class than convex polyhedra. However, in terms of topology, OP do not imply any restrictions. OP allow any number of rings on faces, holes (they can be of any genus) and shells. Thus, they represent a radically different class of polyhedra from the convex class.

There is a large amount of work concerning convex polyhedra whose study is not the purpose of the present work.

OP are a less-used simple class though some studies have been published dealing with or using them. In [10] a B-Rep to CSG conversion algorithm is presented that works for a restricted class of OP. The obtained CSG expression is a Peterson-style formula and the restricted class is the acyclic OP. In [9] the same author extends the domain for a certain class of cyclic OP. In [12] an octree to B-Rep conversion algorithm is presented and an OP is obtained. In [3] an algorithm that simplifies geometry is presented for the particular case of OP; a more complex algorithm is needed for the general case of polyhedra [2].

Boxes, which are both convex and orthogonal, have been widely used in many applications [13], [8], [19] and have been used as approximations, as was explained in the previous section.

As mentioned in this section, in an OP the number of edges incident on a vertex can be three, four or six. From now on we will refer to them as V3, V4 or V6. Moreover, Vertices V3 present two possible configurations depending on the way in which three halfspaces can be arranged to generate a V3 vertex (see Figure 1). Let h_1, h_2 and h_3 be the supporting halfspaces of all three faces incident to a V3 vertex. In configuration A the vertex enclosing region is represented by $h_1 \cap h_2 \cap h_3$ or by $h_1 \cup h_2 \cup h_3$ whereas in configuration B the vertex enclosing region is represented by $(h_1 \cup h_2) \cap h_3$ or by $(h_1 \cap h_2) \cup h_3$ [9].

4 Extreme Vertices Model for Orthogonal Polyhedra

In this section we present a model for two-manifold OP. We consider that all the OP, as well as their geometric elements (faces and edges) with which we operate, are in the same iso-oriented coordinate system.

The Extreme Vertices model, EV, represents OP in a complete and compact way. The model is complete because we can infer from it all the topological and geometric information of the polyhedron.

Splitting and boolean set operations can be done on EV in linear time. Although input data (i.e., vertices coordinates) are floating-point values, no time-consuming floating-point arithmetic is ever performed and so there are no propagation errors. All results are obtained by merely classifying the vertices coordinates of the initial data.

Other operations such as computing the perimeter, area and volume of OP, as well as conversion algorithms between EV and hierarchical B-Rep and Classical Octrees, have also been developed [1].

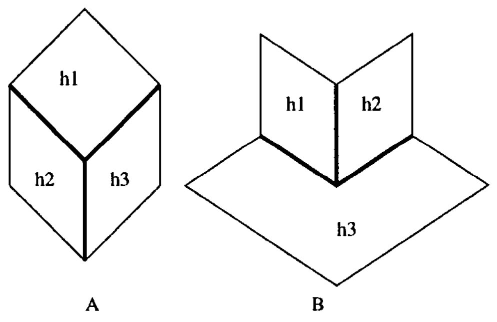


Figure 1: Configutation A and B for V3 vertices.

Definition 4.1 *A* brink *is the longest uninterrupted segment built out of a sequence of collinear and contiguous two-manifold edges of an OP.*

Every edge belongs to a brink, whereas every brink consists of one or more edges and contains as many vertices as the number of edges plus one (see Figure 2 below).

Edges meeting at a V3 vertex are all linearly independent, whereas edges meeting at V4 or V6 vertices are not. Edges meeting at a V4 (V6) vertex belong to two (three) perpendicular directions, that is, they are members of two (three) perpendicular brinks and, hence, they appear as two (three) couples of collinear edges (see Figure 2 above). Furthermore, every V4 or V6 incident edge has a neighbour in the brink corresponding to its direction.

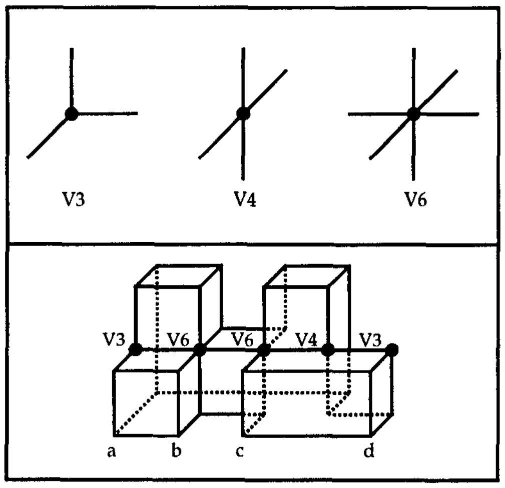


Figure 2: Above) Edges meeting at a V3, V4 and V6 vertex. Below) Example of a brink containing four edges and five vertices. These vertices are respectively V3 (Configuration B), V6, V6, V4 and V3 (Configuration A). Edges (a, b) and (c, d) are collinear but are not contiguous and thus (a, b) is a brink and (c, d) is another brink.

Lemma 4.1 *In a brink both ending vertices are V3 and the remaining (interior) are V4 or V6.*

Proof: Every edge meeting at vertices V4 or V6 has a neighbour in the same brink, thus such vertices cannot appear at the end of any brink. Moreover, any edge meeting at vertices V3 has no neighbour in the same brink and therefore such vertices must appear only at the end of a brink. □

Definition 4.2 *We will call* Extreme Vertices *(EV) of an OP the ending vertices of all the OP brinks, i.e., the V3 vertices of the polyhedron.*

Definition 4.3 *We define the* EV model *for OP as a model that only stores all EV (V3) vertices.*

Lemma 4.2 *Let P be an OP and OH(P) be its isooriented orthogonal hull or minimum bounding box. Then, only a subset of V3 vertices of P lies on the boundary of OH(OP) and, therefore, all V4 and V6 vertices lie in the interior of OH(P).*

Proof: The proof comes from the well-known concept of supportability [21]. In an OP, V3 vertices are locally or complementary supportable and vertices V4 and V6 are non-supportable [9]. Only supportable vertices of an OP can lie on its minimum bounding box. □

Let $VX = \{x_1, x_2, \ldots, x_{nx}\}$ be the ordered set of different values for the x coordinate of every V3 vertex, nx being the total number of different x values (and VY and VZ analogously for their y and z coordinates, with sizes ny and nz).

Lemma 4.3 *For every vertex V4 or V6 with coordinates* (x_i, y_i, z_i), $x_i \in \{x_2, x_3, \ldots, x_{nx-1}\} = VX - \{x_1, x_{nx}\}$ *(and analogously for its y and z coordinates).*

Proof: Vertices V4 (V6) are in the interior of 2 (3) perpendicular brinks (they are in fact the intersection of these brinks) and so their coordinates can be obtained from the coordinates of the V3 ending vertices of these brinks. Thus $x_i \in VX$. However, from lemma 4.2, $x_1 < x_i < x_{nx}$, and therefore V4 and V6 are in the interior of the bounding box of their OP. □

Theorem 4.1 *The EV model for orthogonal polyhedra is a valid B-Rep model.*

Proof: To prove this theorem we must prove that all the geometry, topology and correct orientation of the OP boundary can be obtained from the EV model. Concerning with geometry, from lemma 4.3, all coordinates of vertices V4 and V6 appear as coordinates of vertices V3. Then, although vertices V4 and V6 do not appear in the model, they can be inferred from it. Concerning with topology and orientation, in [1] a conversion algorithm from the EV model to B-Rep is presented that proves this theorem. □

5 Boolean Operations in the EV model

Our approach computes an orthogonal bound for a CSG tree. We consider CSG trees without geometric transformations.

First, all the primitives are approximated by their bounding boxes and then a postorder tree traversal is carried out applying the corresponding operations. In our case the operations are the classical operations of boolean algebra and so the orthogonal bound of the CSG does not depend on the form of the CSG algebraic expression as occurred with boxes (see section 2).

Nevertheless, the advantage of the DF form concerning with culling up empty intersections, also mentioned in section 2, can be applied to our approach and, therefore, the tree traversal is carried out by using the method proposed in [16]. Furthermore, since the CSG primitives are boxes, the intersections are also boxes (when they are not empty) and so the CSG bound is represented as the union of boxes.

In this section the boolean operations algorithm for OP is presented. The algorithm basically performs a geometric merge between OP represented in a sorted EV model. The algorithm computes a sequence of 2D sections from the 3D model and the same algorithm is recursively applied to each of these 2D sections to obtain 1D sections. Then 1D boolean operations are performed on these 1D sections. The recursion upwards by converting the resulting sequence of sections into an EV model thus produces the corresponding result.

5.1 Operations on the EV model

Definition 5.1 *An* ABC-sorted EV model *is an EV model where vertices V3 are sorted first by coordinate A, then by B and then by C.*

EV models can be sorted in six different ways: XYZ, XZY, YXZ, YZX, ZXY and ZYX. In a ZXY-sorted EV model, for instance, its vertices are arranged in planes perpendicular to the Z axis (i.e., with the same z coordinate). In each such plane they are arranged in lines parallel to the Y axis (i.e., with the same x coordinate). Finally, they appear as y intervals (see Figure 3).

Let us consider an ABC-sorted EV model,

Definition 5.2 *A* plane of vertices *of an OP is the set of vertices lying on a plane perpendicular to the A axis. We will also refer to the set of vertices lying on a line parallel to the C axis within a plane of vertices as a* line of vertices.

Definition 5.3 *A* strip *is the region between two consecutive planes (lines) of vertices.*

Definition 5.4 *A* section *is the polygon resulting from the intersection between an OP and an orthogonal plane perpendicular to the A axis which does not coincide with any plane of vertices.*

All the orthogonal planes intersecting an OP in the same strip give the same section. Hence, every strip is represented by a section. Furthermore, as an OP can be interpreted as a sequence of strips, we can define the sequence of sections for an OP.

All these concepts related to sections can also be defined in 2D. A section is also a 1D polygon resulting from the intersection between a 2D orthogonal polygon and an orthogonal line perpendicular to both the A and B axes which does not coincide with any line of vertices.

A sorted-EV model is a sequence of planes (lines) of vertices. The number of elements of this sequence, np, is the number of different A coordinates in the model. The number of sections is $ns = np+1$ because the empty sections S_0 and S_{np} are also considered. Figure 4 shows the sections and planes of vertices for an OP.

An ABC-sorted EV model can represent n-dimensional OP ($n \leq 3$) by taking into account the last n coordinates. Thus, the planes and lines of vertices of an OP will also be represented in this model. Moreover, as a section is actually an OP, 1D and 2D sections will also be represented in this model.

Then, we define the *ABCsorted* type with the following operations:

```
FUNCTION IniEv () RETURN ABCsorted
{Returns an empty EV model}

PROCEDURE Put (INPUT plv: ABCsorted,
               I/O P: ABCsorted,
               INPUT dim:INTEGER)
{Appends a plane (dim=2) or a line
(dim=1) to an EV model, P}

FUNCTION Read (P: ABCsorted,
               dim:INTEGER)
               RETURN ABCsorted
{Extracts the next plane (dim=2) or
line (dim=1) from an EV model, P}

FUNCTION End (P: ABCsorted)
              RETURN BOOLEAN
{Returns TRUE if the end of P has
been reached}

FUNCTION MergeXor (P, Q: ABCsorted;
                   dim: INTEGER)
                   RETURN ABCsorted
{Applies the Exclusive OR operation to
the vertices of P and Q and returns the
resulting set}

PROCEDURE SetCoord (I/O P: ABCsorted,
                    INPUT Coord: REAL,
                    INPUT dim: INTEGER)
{Sets the A (dim=2) or the B (dim=1)
coordinate to Coord on every vertex
of the plane (line) of vertices P}

FUNCTION GetCoord (P: ABCsorted,
                   dim:INTEGER) RETURN REAL
{Gets the common A (dim=2) or B (dim=1)
coordinate of the plane (line) of vertices P}
```

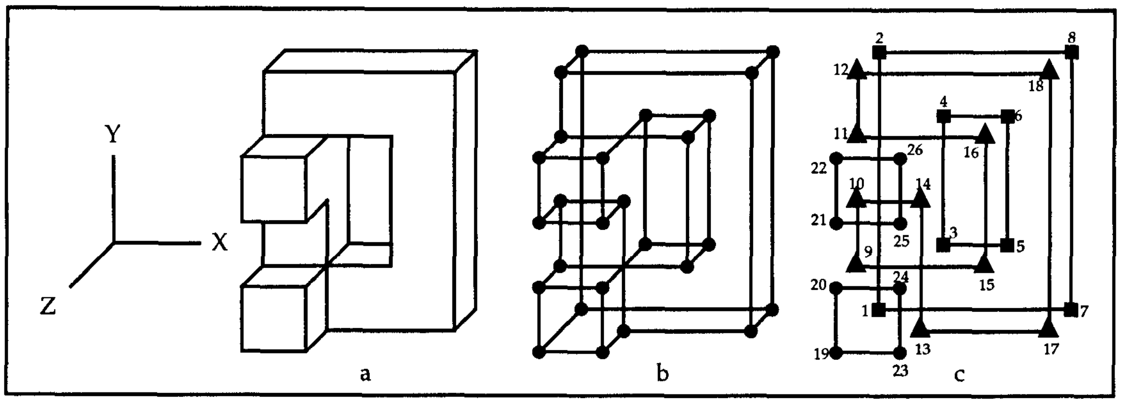


Figure 3: ZXY-sorted EV model. a) A hidden line representation of an OP with one V6, one V4 and 26 V3s. b) Its corresponding wire-frame representation. c) This representation shows the order number for each V3 vertex and the three planes of vertices of the model (with different marks).

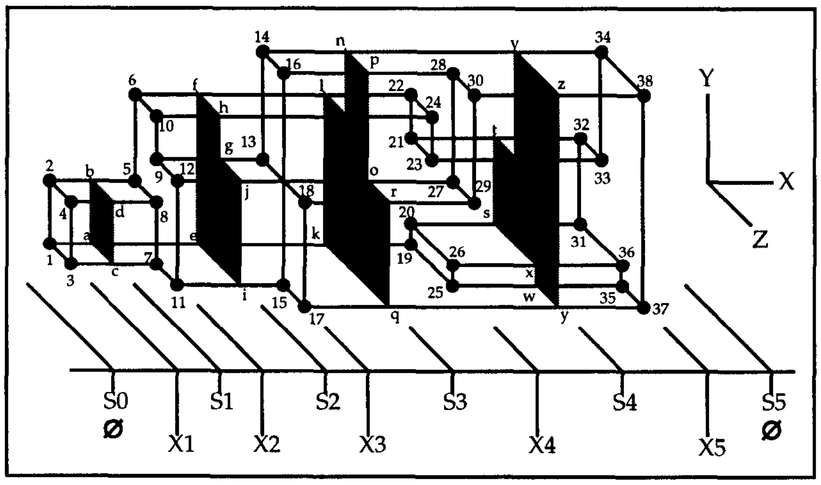


Figure 4: This OP is represented in an XZY-sorted EV model. It has 5 planes of vertices, X_1 to X_5. Each of them corresponds to the set of vertices with the same coordinate X. The shaded-in polygons represent the four sections S_1, S_2, S_3 and S_4. There are two more empty sections, S_0 and S_5.

5.2 Computing sections from planes (lines) of vertices and vice versa

A section S_i is computed by carrying out an exclusive OR between its previous section S_{i-1} and its previous plane (line) of vertices P_i:

$$S_0 = \emptyset$$

$$S_i = S_{i-1} \otimes P_i, \forall i \in [1, np]$$

$\otimes$ corresponds to the exclusive OR operation.

From this recurrence relation it follows that $S_{np} = \emptyset$.

Then, we define the corresponding function GetSection:

```
FUNCTION GetSection (S: ABCsorted,
                     plv: ABCsorted,
                     dim: INTEGER)
                     RETURN ABCsorted
{returns the next section of an OP whose
previous section is S. This function works
for dimension 2 or 1. If dim=2 (dim=1),
plv is the previous plane (line)
of vertices and S is a 2D (1D) section}

   RETURN (MergeXor(S, plv, dim))
ENDFUNCTION
```

An algorithm that computes the sequence of sections of an OP from its EV model using functions IniEv and GetSection is presented in [1].

A plane (line) of vertices P_i of an OP is computed by carrying out an exclusive OR between its previous S_{i-1} and next S_i section:

$$P_i = S_{i-1} \otimes S_i, \forall i \in [1, np]$$

Then, we define the corresponding function GetPlv:

```
FUNCTION GetPlv (Si: ABCsorted,
                 Sj: ABCsorted,
                 dim: INTEGER)
                 RETURN ABCsorted
{This function also works for dimensions
2 or 1. If dim=2 (dim=1), Si and Sj are 2D
(1D) consecutive sections and returns the
plane (line) of vertices between Si and Sj}

   RETURN (MergeXor (Si, Sj, dim)
ENDFUNCTION
```

Actually, this function performs the same computations as the GetSection function, i.e., an exclusive OR between two sets of vertices although, as they are conceptually different, we will use both of them.

An algorithm that computes the EV model from a sequence of sections of an OP is also presented in [1].

5.3 Boolean Operations algorithm

Now, we are able to present the boolean operations algorithm. The algorithm merges two OP, say P and Q, represented in the same ABC-sorted EV model, in such a way that the corresponding planes of vertices are also merged. We consider all the resulting strips. Some of them will correspond to untouched strips of P or Q and only one section will have to be considered. However, some other strips will correspond to a part of a P strip and a part of a Q strip with their corresponding sections. The algorithm considers these sections as 2D OP and operates them in the same way.

We can explain the algorithm as follows. The sequence of sections for objects P and Q are computed. Then, these sections are merged in order to compute the sequence of sections of the R resulting object. Finally, from this sequence of sections, the EV model of the resulting object R is obtained. Nevertheless, the implemented algorithm does not work in this sequential form; it actually works in a wholly merged form and only needs to store one section for each of the P and Q operands and two consecutive sections for the result R. Thus, the algorithm is $O(n)$ as merging-like algorithms are, being n the total number of vertices of P and Q. This conclusion is based on the fact that $\sum_i nPLV_i \approx \sum_i nS_i$, i.e., the total number of extreme vertices of all planes of vertices (which is the number of extreme vertices of the OP) is approximatelly equal to the total number of extreme vertices of all the sections of the OP. Figure 5 shows the boolean operations algorithm.

Function OpBool1D performs 1D boolean operations between P and Q that are now collinear lines of vertices.

Procedure GetPlane obtainss the next plane (line) of vertices `plvi` of P or Q, with its common coordinate `coord`, and shows to which of these objects `obj` it belongs. The plane (line) of vertices is obtained using function Read and its common coordinate using function GetCoord (see section 5.1). This procedure works as in a merging process.

Functions GetSection and GetPlv perform an exclusive OR between the sets of vertices of their operands (see subsection 5.2).

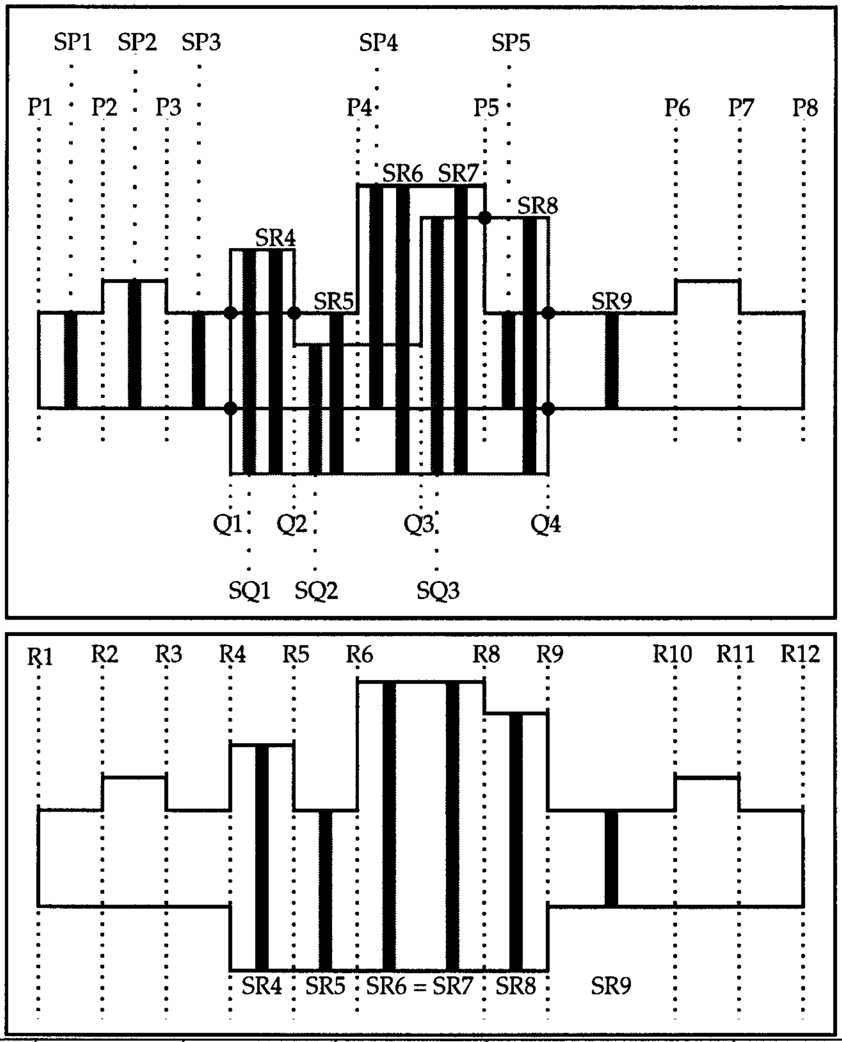
OpBool works for 3D OP (dim=3) and for 2D orthogonal polygons (dim=2). The recursive case of this procedure is a merging-like algorithm.

When the end of one of the objects is reached, the main iteration finishes and the remaining planes (lines) of vertices of the other object are either appended or not to the resulting object depending on the boolean operation considered. Procedure PutBool performs this boolean operation based appending process.

Figure 6 shows a 2D running example and Figure 7 shows a 3D example.

```
TYPE Object = ENUM {P, Q} ENDTYPE

FUNCTION OpBool (P, Q: ABCsorted,  {the input objects}
                 dim: INTEGER,     {dimension of P and Q}
                 op: BoolOp)       {the Boolean operation}
                 RETURN ABCsorted
   VAR
    s[P..Q]: ABCsorted  {s[P], s[Q]: current sections of P, Q}
    sRprevious, sRcurrent: ABCsorted  {sections of the result, R}
    plvi, plvo: ABCsorted  {input and output planes (lines) of vertices}
    obj: Object  {the current selected object}
    coord: REAL {The common coordinate of a plane (line) of vertices}
   ENDVAR

   IF dim = 1 THEN
      RETURN (OpBool1D(P, Q, op))
   ELSE
      dim:= dim - 1
      s[P]:= IniEv()
      s[Q]:= IniEv()
      sRcurrent:= IniEv()

      GetPlane(P, Q, dim, plvi, coord, obj)
      WHILE NOT End(P) AND NOT End(Q) DO
          S[obj]:= GetSection(plvi, S[obj], dim)
          sRprevious:= sRcurrent
          sRcurrent:= OpBool(s[P], s[Q], dim, op)
          plvo:= GetPlv(sRprevious, sRcurrent, dim)
          SetCoord(plvo, coord, dim)
          Put(plvo, R, dim)
          GetPlane(P, Q, dim, plvi, coord, obj)
      ENDWHILE

      WHILE NOT End(P)  DO
          PutBool(plvi, R, op)
          plvi:= Read(P, dim)
      ENDWHILE
      WHILE NOT End(Q)  DO
          PutBool(plvi, R, op)
          plvi:= Read(Q, dim)
      ENDWHILE
      RETURN (R)
   ENDIF
ENDFUNCTION
```

Figure 5: The boolean operation algorithm.



plvi	s[P]	s[Q]	sRanterior	sRactual	plvo
P1	SP1 = P1	∅	∅	SP1	R1 = P1
P2	SP2	∅	SP1	SP2	R2 = P2
P3	SP3	∅	SP2	SP3	R3 = P3
Q1	SP3	SQ1 = Q1	SP3	SR4 = SP3 ∪ SQ1	R4
Q2	SP3	SQ2	SR4	SR5 = SP3 ∪ SQ2	R5
P4	SP4	SQ2	SR5	SR6 = SP4 ∪ SQ2	R6
Q3	SP4	SQ3	SR6	SR7 = SP4 ∪ SQ3	R7= ∅
P5	SP5	SQ3	SR7	SR8 = SP5 ∪ SQ3	R8
Q4	SP5	SQ4 = ∅	SR8	SR9 = SP5 ∪ SQ4	R9
P6					R10 = P6
P7					R11 = P7
P8					R12 = P8

Figure 6: Boolean Operations running example. End(Q)is TRUE when Q4 is selected. We can observe that SR6=SR7 since SP4 ∪ SQ2 = SP4 ∪ SQ3, thus making R7 = ∅

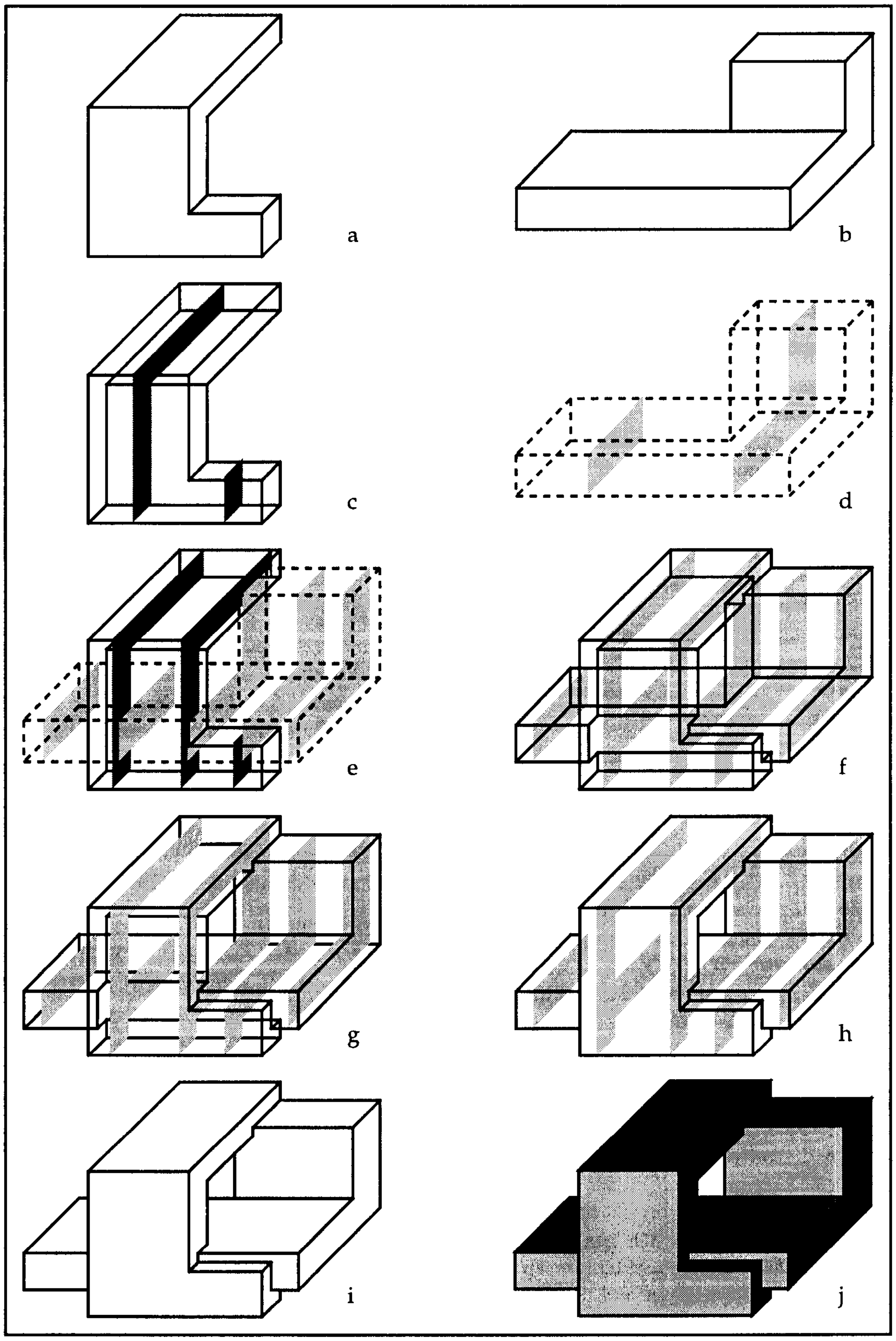


Figure 7: Boolean Operations: 3D example. (a),(b) Two OP. (c),(d) Sections of these OP. (e) OP in overlapping position and the corresponding overlapping sections. (f),(g),(h) The resulting sections and OP (wireframe and HLR). (i),(j) The resulting OP (HLR and shaded).

6 Comparison between geometric bounds

In this work we propose the EV Model as a new model for representing valid OP in a compact way.

We have also developed classification algorithms for OP. In [1] an $O(n)$ splitting algorithm and an $O(lgn)$ point classification algorithm are proposed.

We want now to compare OP with simple boxes as geometric bounds. Boolean operations on boxes are of constant complexity, whereas boolean operations on OP are $O(n)$. Classification algorithms are also more complex for OP than for boxes (since boxes are of constant complexity). So, our approach will be more time-consuming than boxes-based approaches when the CSG geometric bound is computed and when classification tests are performed on it. Nevertheless, OP are tighter than boxes and so classification tests will be more deterministic.

Moreover, the bounding OP used for a primitive is merely its bounding box and we will traverse the CSG tree in its DF using the method presented in [16] i.e., performing unions of intersections. Thus, when performing intersections our method also deals with boxes and has constant complexity. Obviously, the method must finally perform unions between the intersection results and so complexity is $O(n)$.

7 Conclusions and Future work

In this work we have proposed the use of OP as geometric bounds in CSG. The proposal is based on the fact that a simple boolean operations algorithm can be applied for OP. This boolean operations algorithm is a merging-like algorithm and runs in O(n).

Although input data (i.e., vertices coordinates) are floating-point values, no time-consuming floating-point arithmetic is ever performed and so there are no propagation errors. All results are obtained by simply classifying vertices coordinates of the initial data. Moreover, round-off errors in the input data can be avoided by performing a space discretization based on the primitive bounding boxes.

Working with OP instead of boxes as geometric bounds is more time-consuming when computing the bound and when classification algorithms are applied. However, OP are tighter than boxes and so classification tests will be more deterministic.

In the future we intend to compare these theoretical results with experimental ones. We will also perform a rigorous study of the complexity of the presented algorithm. Furthermore, we are extending the EV model and the corresponding operations for pseudomanifold OP. A psedomanifold polyhedron [20] is an r-set with non-manifold boundary [17]. Finally, we will study other applications of OP.

8 Acknowledgments

The present work has been partially supported by CICYT grants TIC-95-630-C03.

The authors would like to thank P. Brunet and R. Joan-Arinyo for their valuable comments and suggestions.

References

[1] A. Aguilera and D. Ayala. The Extreme Vertices Model model EVM for Orthogonal Polyhedra. Technical Report LSI-97-6-R, LSI-Universitat Politècnica de Catalunya, 1997.

[2] C. Andújar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Solé. Automatic generation of multiresolution boundary representations. volume 15. EUROGRAPHICS'96, 1996.

[3] D. Ayala, C. Andújar, and P. Brunet. Automatic simplification of orthogonal polyhedra. In D. Fellner, editor, *Modeling, Virtual Worlds, Distributed Graphics: Proceedings of the International MVD'96 Workshop.* Infix, 1995.

[4] S. Cameron and J.R. Rossignac. Relationship between S-bounds and Active Zones in Constructive Solid Geometry. Proceedings of the International Conference on the Theory and Practice of Geometric Modelling. FRG, 1988.

[5] S. Cameron and C. Yap. Refinement methods for geometric bounds in Constructive Solid Geometry. *ACM Transactions on Graphics*, 11(1):12 – 39, 1992.

[6] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer Verlag, 1987.

[7] J. Goldfeather, S. Molnar, G. Turk, and H. Fuchs. Near real-time CSG rendering using tree normalization and geometric pruning. *IEEE Computer Graphics & Applications.*

[8] C. M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kauffmann Publishers, Inc., 1989.

[9] R. Joan-Arinyo. Domain extension of isothetic polyhedra with minimal CSG representation. *Computer Graphics Forum*, (5):281 – 293, 1995.

[10] R. Juan-Arinyo. On Boundary to CSG and Extended Octrees to CSG conversions. In W. Strasser, editor, *Theory and Practice of Geometric Modeling*, pages 349–367. Springer-Verlag, 1989.

[11] M. Mazzetti and L. Ciminiera. Computing CSG-tree boundaries as algebraic expressions. *CAD*, 26(6):417 – 425, 1994.

[12] C. Montani and R. Scopigno. *Graphics Gems II*, chapter IV.7 Quadtree/Octree to boundary conversion, pages 202 – 218. Academic Press, Inc, 1991.

[13] F.P. Preparata and M.I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, 1985.

[14] A. Rappoport. The n-dimensional Extended Convex Differences Tree (ECDT) for representing polyhedra. In J. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, 1991.

[15] A. Requicha. Representations for rigid solids: Theory, methods, and systems. *Computing Surveys of the ACM*, 12:437–464, 1980.

[16] J. Rossignac. Proceesing disjunctive forms directly from CSG grafs. In *CSG 94. Set-theoretic Solid Modelling Techniques and Applications*, pages 55 – 70. Information Geometers Ltd, 1994.

[17] J. R. Rossignac and A. A. G. Requicha. Contructive Non-Regularized Geometry. *Computer-aided design*, 23(1):21 – 32, 1991.

[18] J. R. Rossignac and H. B. Voelcker. Active zones in CSG for accelerating boundary evaluation, redundancy elimination, interference detection and shading algorithms. *ACM Transactions on Graphics*, 8(1):51 – 87, 1989.

[19] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley Publ., Reading, MA, 1989.

[20] K. Tang and T. Woo. Algorithmic aspects of alternating sum of volumes. Part 1: Data structure and difference operation. *CAD*, 23(5):357 – 366, 1991.

[21] K. Tang and T. Woo. Algorithmic aspects of alternating sum of volumes. Part 2: Nonconvergence and its remedy. *CAD*, 23(6):435 – 443, 1991.

[22] R. B. Tilove. Set membership classification: a unified approach to geometric intersection problems. *IEEE Transactions on Computers*, 29(10):874 – 883, 1980.