ROMA TRE

Università degli Studi di Roma Tre
Dipartimento di Discipline Scientifiche
Via della Vasca Navale, 84 – 00146 Roma, Italy.

Bijective Dimension-Independent Boundary to Interior Mapping with BSP Trees

Claudio Baldazzi, Alberto Paoluzzi

TR-INF-17-96 **August 1996**

ABSTRACT

In this paper we discuss two algorithms for performing both ways the conversion between the boundary and the interior of d-dimensional polyhedra. Both a d-polyhedron and its $(d-1)$-faces are represented as BSP trees. An algorithm for boundary to BSP conversion starting from a standard B-rep was given by Thibault and Naylor in [15]. In this paper we assume no structure, no ordering and even no orientation on the set of boundary BSP-trees. The resulting algorithm can be executed on a parallel computing architecture. Also, if the incidence relations between faces of various dimension are known, then the algorithm can be executed iteratively, so allowing for reconstructing a d-polyhedron from its $(d-k)$-faces, $1 \leq k \leq d$. The converse algorithm allows to compute the BSP of the intersection of a generic hyperplane in $\mathcal{E}^d$ with the BSP representation of a d-polyhedron. If such section hyperplane is the support of a $(d-1)$-face, then a BSP tree of the face is generated. This second algorithm may be used iteratively to compute the k-skeletons of a d-polyhedron.

1 Introduction

Problem statement A very interesting representation without topology of d-polyhedra was given by Naylor [8] by using BSP trees [4]. Naylor represents a solid as the union of quasi-disjoint "full" convex cells of the space partition generated by a binary tree of hyperplanes. In this paper we solve the following two problems: (a) construct the BSP representation of the intersection of a BSP tree with a given hyperplane, and specialize it to the case of a face hyperplane; (b) construct a cell decomposition, via a BSP tree, of the interior of a d-dimensional polyhedron (bounded by an orientable hypersurface) starting from the unordered collection of the BSP trees associated to its *unoriented* $(d-1)$-faces.

Naylor and Thibault [15] discuss an algorithm for converting a B-rep of a d-polyhedron P into a BSP tree in the hypothesis that the $(d-1)$-faces of P have an orientation. In the present paper we relax this assumption, by assuming no orientation for the boundary faces. Our algorithm also uses a BSP representation of the faces, so allowing for closure on the set of BSP trees.

Motivation A boundary representation of a 3D solid can be seen [17] as a connected subgraph of the complete oriented graph having as nodes the sets V, E, F of vertices, edges and faces of the solid. For the representation to be complete, it is necessary to add an ordering (i.e. an orientation) to some subsets of the chosen relationships between vertices, edges and faces. E.g., vertices and edges upon the boundary of a given face loop must be circularly ordered, and so must be the faces incident on a vertex neighborhood [6]. This ordering information is particularly useful when representations of non-manifolds are defined (see, e.g., Weiler [16]). Such ordering information is closely linked to the orientation of the boundary of the solid. A good discussion of the importance of ordering in the description of solids can be found in Rossignac [12].

A very different viewpoint is sometime assumed, where little or no topology at all is kept in the representation of the solid [14, 5, 10]. This is quite usual in computer graphics, where the standard representation of a 3D polyhedron is the collection of its boundary polygons, usually considered oriented in order to efficiently culling the back-faces.

When considering representations for d-dimensional objects it is actually crucial to be able to avoid as much as possible the consideration of topology and in particular the orientation and/or the ordering of subsets of incidence relations. It is not difficult to understand that to maintain the coherent orientation of the incidence structures of boundary subsets may become too hard or too complex for generic d-dimensional solids, or even worst, for k-dimensional objects embedded in some d-dimensional space, $k < d$, as is the case for the k-skeletons of a d-polyhedron.

If the incidence relations between faces of dimension $k-1$ and k are known, for $1 \leq k \leq d$, then our Boundary $\rightarrow$ Interior algorithm can be executed iteratively, so allowing for reconstructing a d-polyhedron from its $(d-k)$-faces. E.g., such an approach might be used in Photogrammetry to reconstruct a solid model from the edges of stereo pairs. Also, the iterate execution of the converse algorithm (Interior $\rightarrow$ Boundary) may allow for computation of a BSP representation of the k-skeletons of a d-polyhedron. For example, this approach might be useful for quickly interacting with pictures of internal structures of higher dimensional geometric objects.

Previous work Fuchs, Kedem and Naylor [4] introduced the BSP trees for computing hidden surface removed scenes. Naylor, Amanatides and Thibault started using BSP trees as representations of solids, and defined regularized Boolean operations by merging BSP trees [7]. Cell-decompositions of non-manifold solids starting from a B-rep were studied by Bajaj and Dey [1], Thibault and Naylor [15] and Shapiro and Vossler [13]. Baldazzi [2] implemented regularized Boolean operations on d-dimensional BSP trees using Linear Programming techniques. Baldazzi and Paoluzzi have recently discussed in [3] a conversion algorithm from 2D polygons to BSP trees, so obtaining a cell decomposition of the polygon interior. That paper describes an approach based on the Boolean XOR of unbounded plane stripes associated to the polygon edges. Such an approach is extended in Section 4 of the present paper to work with polyhedra of whatever dimension.

Preview In Section 2 some background concepts are recalled and definitions are given concerning Binary Space Partition trees, the BSP representation scheme and Boolean set operations on polyhedra. In Section 3 the problem of computing a B-rep based on BSP trees by starting from a BSP decompositive representation is solved by introducing algorithms for computing "section BSP", "face BSP" and "boundary BSP" trees. In Section 4 the converse problem of computing a BSP representation of the interior starting from the unordered collection of BSP trees associated to the unoriented faces of the boundary is discussed. Some examples of computation of the concepts here discussed are given in the whole paper.

2 Background

Given a set of hyperplanes in $\mathcal{E}^d$, a Binary Space Partition (BSP) tree defined on such hyperplanes establishes a hierarchical partitioning of the $\mathcal{E}^d$ space.

A node ν of such a binary tree represents a convex and possibly unbounded region of $\mathcal{E}^d$ denoted by R_ν. The two sons of an internal node ν are denoted as $below(\nu)$ and $above(\nu)$, respectively. Leaves correspond to unpartitioned regions, which are called either *empty* (OUT) or *full* (IN) *cells*. Each internal node ν of the tree is associated with a *partitioning hyperplane* h_ν, which intersects the interior of R_ν. The hyperplane h_ν partitionates R_ν into three subsets:

- the subregion $R^0_\nu = R_\nu \cap h_\nu$ of dimension $d-1$;
- the subregion $R^-_\nu = R_\nu \cap h^-_\nu$ where h^-_ν is the negative halfspace of h_ν. The halfspace h^-_ν is associated with the tree edge $(\nu, below(\nu))$. The region R^-_ν is associated with the *below* subtree, i.e. $R^-_\nu = R_{below(\nu)}$;
- the subregion $R^+_\nu = R_\nu \cap h^+_\nu$ where h^+_ν is the positive halfspace of h_ν. The halfspace h^+_ν is associated with the tree edge $(\nu, above(\nu))$. The region R^+_ν is associated with the *above* subtree, i.e. $R^+_\nu = R_{above(\nu)}$;

R_ν is the intersection of the closed halfspaces on the path from the root to ν. The region described by any node ν is:

$$R_\nu = \bigcap_{e \in E(\nu)} h_e$$

where $E(\nu)$ is the edge set on the path from the root to ν and h_e is the halfspace associated to the edge e.

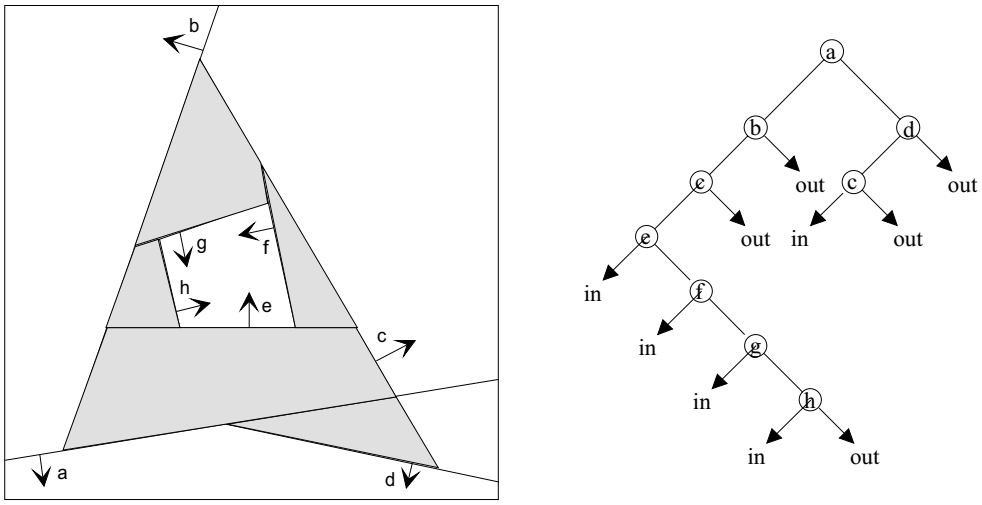

Figure 1: Concave polyhedron with a hole and corresponding BSP tree.

Definition 1 (Representation Space) *We call $\mathcal{BSP}^d$ the set of all BSP trees which partitionate $\mathcal{E}^d$.*

According to the Requicha's approach and terminology [11]:

Definition 2 (Representation scheme) *We define a BSP representation scheme as a mapping*

$$\mathrm{BSP} : \mathcal{P}^d \rightarrow \mathcal{BSP}^d,$$

where $\mathcal{P}^d$ is the set of linear solid polyhedra in Euclidean space $\mathcal{E}^d$.

So, BSP(P) will denote a BSP representation of the polyhedron P. Let notice that the BSP scheme is complete but not unique, since different trees can be associated to the same polyhedron.

We suppose that regularized set operations of union ($|$), intersection ($\&$), difference ($-$) and symmetric difference ($\wedge$) are available on the space $\mathcal{BSP}^d$ by using an algorithmic approach [2] derived by that of Naylor [7], and claim that for all $P, Q \in \mathcal{P}^d$:

$$\begin{aligned} eval(\mathrm{BSP}(P) \mid \mathrm{BSP}(Q)) &= P \cup^* Q; \\ eval(\mathrm{BSP}(P) \,\&\, \mathrm{BSP}(Q)) &= P \cap^* Q; \\ eval(\mathrm{BSP}(P) - \mathrm{BSP}(Q)) &= P \,/^*\, Q; \\ eval(\mathrm{BSP}(P) \wedge \mathrm{BSP}(Q)) &= P \,\triangle^*\, Q, \end{aligned}$$

where $eval$ just means the set union of full cells in the argument tree, and where, as usual,

$$P \; op^* \; Q = \mathrm{clos}(\mathrm{int}(P \; op \; Q)), \qquad op \in \{\cup, \cap, /, \triangle\}.$$

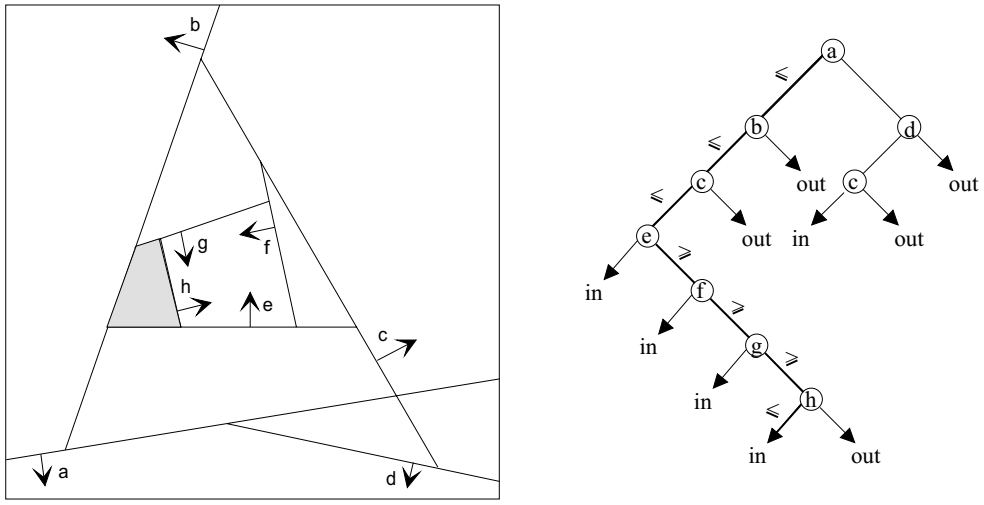

Figure 2: The halfspaces along the path give a set of linear inequalities. The set of common solutions of such inequalities is a convex cell.

It is recalled that the symmetric difference is also called exclusive OR (also XOR), since

$$P \triangle Q = (P \cup Q) \ / \ (P \cap Q) = (P \ / \ Q) \cup (Q \ / \ P).$$

Finally, a unary operation of complement ($\sim$) on BSP trees is defined, such that

$$eval(\sim \mathrm{BSP}(P)) = -P,$$

where $-P$ is the complement of the P polyhedron.

3 From Decompositive BSP to Boundary BSP

Let be given a BSP representation of a d-dimensional polyhedron P in $\mathcal{E}^d$. In this section we study a mapping between BSP(P) and a set of BSP trees associated to the $(d-1)$-faces of P. This goal is accomplished by computing the BSP of the intersection of BSP(P) with any affine hyperplane h in $\mathcal{E}^d$. In particular, the "section" tree BSP($P \cap h$) is computed by properly traversing BSP(P). To be consistent with the standard definition of BSP trees such a new tree must be affinely transformed into the $(d-1)$-dimensional coordinate subspace $x_d = 0$ and then projected into $\mathcal{E}^{d-1}$. In a subsequent subsection an algorithm is given to compute the so-called "face BSP" associated to a face of P. This face is possibly non-convex and/or unconnected and/or even unorientable (see Figure 9c). A formal definition of "Boundary BSP" is finally given.

```
Algorithm Section-BSP (ν: BSP node; h: hyperplane): BSP node;
    loop {
        if ν is a leaf then return ν
        else if (h ∩ h_ν ∩ R_ν) ≠ ∅ then {
            below(ν) = Section-BSP(below(ν),h);
            above(ν) = Section-BSP(above(ν),h);
            return ν }
        else if (h ∩ h_ν^- ∩ R_ν) ≠ ∅ then {
            delete above(ν) and ν;
            ν = below(ν) }
        else {                          // (h ∩ h_ν^+ ∩ R_ν) ≠ ∅
            delete below(ν) and ν;
            ν = above(ν) }
    }
```

Figure 3: The algorithm Section-BSP

3.1 Section BSP trees

Definition 3 (Section) *We call* section *operator*

$$\sigma : \mathcal{BSP}^d \times \Re^{d+1} \rightarrow \mathcal{BSP}^{d-1}$$

a function which, for any given pair $(\mathrm{BSP}(P), h)$*, with a tree and a hyperplane, returns a tree* $\mathrm{BSP}(P \cap h)$*, which represents the partition of* h *induced by* $\mathrm{BSP}(P)$*. Such a tree* $\sigma(\mathrm{BSP}(P), h)$ *is returned properly transformed into the* $x_d = 0$ *coordinate subspace.*

The algorithm to compute the tree $\mathrm{BSP}(P \cap h)$ starting from $\mathrm{BSP}(P)$ and from the equation of an hyperplane h is made of two steps:

1. Traversal of $\mathrm{BSP}(P)$ with a suitable tree pruning;
2. some proper transformations to map h into the subspace $x_d = 0$.

Tree traversal The traversal can be described as follows. Let the root node ν be a formal parameter of a recursive algorithm: If ν is a leaf then return ν; else if h and h_ν intersect inside R_ν then traverse both the below and the above subtrees; else if h and R_ν intersect inside h_ν^- then delete the above subtree and substitute ν with the root of its below subtree; else (if h and R_ν intersect inside h_ν^+) delete the below subtree and substitute ν with the root of its above subtree. This algorithm is given more formally in Figure 3.

Tree transformation The computation which follows the tree traversal is composed of some elementary steps:

1. *Translation.* Let denote with $h = (h_0, h_1, \ldots, h_d)$ the affine set $h_0 + h_1 x_1 + \cdots + h_d x_d = 0$. First translate to the origin the point p intersection of h with the

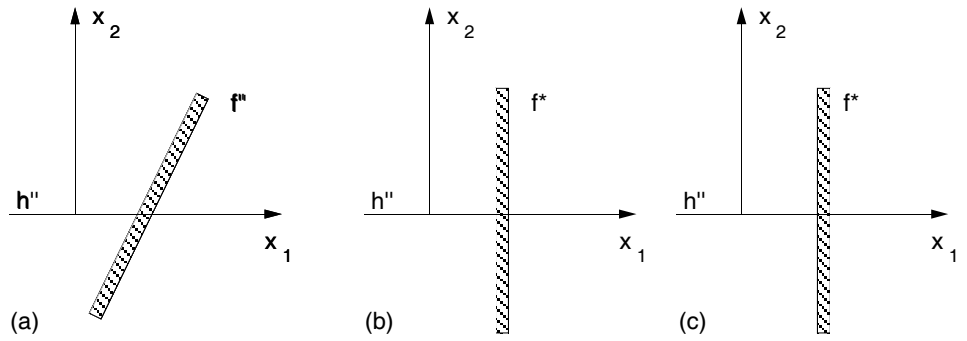

Figure 4: Orthogonalization of a hyperplane. (a) The above subspace of f''; (b) Coherent orientation of f^*; (c) Non coherent orientation of f^*.

subspace $x_1 = \cdots = x_{d-1} = 0$, in the hypothesis that $h_d \neq 0$. So we have $p = (1, 0, \ldots, 0, -h_0/h_d)$, where the first coordinate is the homogeneous one. Let call $T = T(0, \ldots, 0, h_0/h_d)$ the corresponding translation matrix.

2. *Rotation.* The translated $h' = h\,T^{-1}$ hyperplane is then rotated to coincide with the $x_d = 0$ coordinate subspace. Let call R the corresponding transformation matrix. The matrix R can be computed as the product of $d-1$ elementary rotations. An elementary rotation is an isometry with a fixed coordinate subspace of codimension 2.

3. *Orthogonalization.* For all hyperplanes f in $S = \text{Section-BSP}(P, h)$, first transform it into $f'' = f\,T^{-1}R^{-1}$, then make it orthogonal to the subspace $x_d = 0$. This is done by computing an hyperplane bundle and choosing the bundle element f^* which is orthogonal to $x_d = 0$, i.e. such that $f^*_d = 0$. So, we have:

$$f^* = h'' + \lambda f'', \qquad \text{with} \qquad \lambda = -\frac{h''_d}{f''_d},$$

in the hypothesis that $f''_d \neq 0$. Indeed, it is necessary to guarantee a coherent orientation for the above and below subspaces of f'' and f^*. This can be done by taking a point $x \in h''$ such that $x \notin f''$. If the sign of $f''(x)$ is different from that of $f^*(x)$ then multiply f^* by -1 (See Figure 4).

4. *Elimination of the affine support.* A special case arises when the section hyperplane h is coincident with a face hyperplane f. In such a case f would be transformed into the null covector $f^* = (0, \ldots, 0)$, i.e. into the target space $\mathcal{E}^{d-1}$, and its node should be eliminated from the output tree. A check for such a case must be so performed. If and only if h is the affine support of some face f (and hence it is a node in the original tree), then it must be deleted as follows from the output tree:

   (a) compute $S^+ = S \;\&\; H^+$ and $S^- = S \;\&\; H^-$, where we denote with H^- and H^+ the BSP trees with root h and with only two leaves labeled IN, OUT and OUT, IN, respectively, as shown in Figure 5.

   (b) In S^+ delete the node ν such that $h = h_\nu$ and delete the below subtree; In S^- delete the node ν and the above subtree; in both cases substitute ν with the root of the remaining subtree.

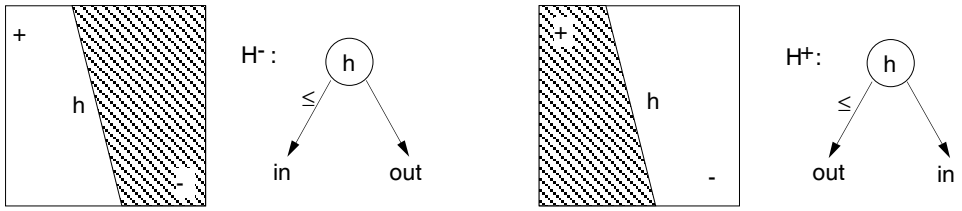

Figure 5: The basic BSP trees associated to the two opposite halfspaces of the h hyperplane.

(c) Compute $S = S^+ \mid S^-$.

5. *Projection.* Finally eliminate the x_d coordinate from the hyperplane equations associated to the nodes of S. It becomes so possible to represent every σ-BSP tree in the standard way, as an element of $\mathcal{BSP}^{d-1}$.

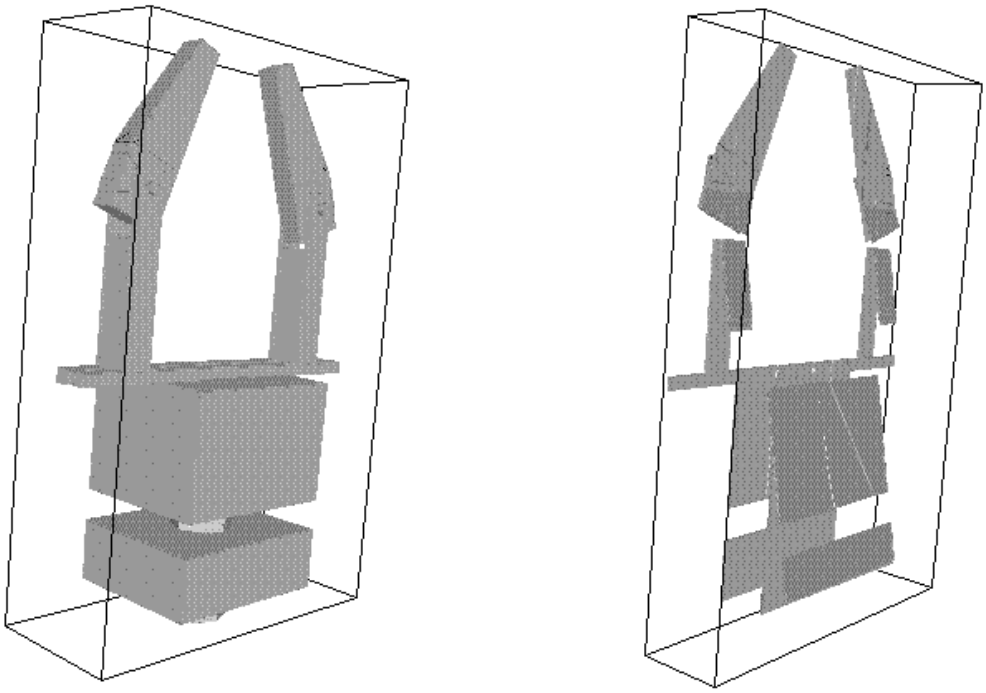

Figure 6: (a) A 3D model; (b) two sections of the object.

Example 1
In Figure 6a the BSP of a robot pliers generated by using the geometric language `PLaSM` [9] is shown. In Figure 6b two σ-BSP trees are displayed in the same picture.

3.2 Face BSP

In this section we discuss the computation of the tree BSP(f) $\in \mathcal{BSP}^{d-1}$ associated to the generic face f of the polyhedron P, when a representation $A = \text{BSP}(P) \in \mathcal{BSP}^d$ is given. The computing procedure can be summarized as follows:

1. Consider the hyperplane h_f, affine support of the face f. Let denote with H_f^- and H_f^+ the basic BSP trees with root h_f and two leaves labeled IN, OUT and OUT, IN, respectively.

   Compute the tree $B =\sim A$, and intersect both A and B with both H_f^+ and H_f^-:

$$A_f^+ = A \;\&\; H_f^+, \qquad A_f^- = A \;\&\; H_f^-,$$
$$B_f^+ = B \;\&\; H_f^+, \qquad B_f^- = B \;\&\; H_f^-.$$

2. Do extract from every such tree the partial tree of the only planes which are incident[1] on h_f, and transform it into an element of $\mathcal{BSP}^{d-1}$. This is exactly the task performed by the operator σ previously discussed. In formal terms, let compute the following trees in $\mathcal{BSP}^{d-1}$:

$$A^+ = \sigma(A_f^+, h_f), \qquad A^- = \sigma(A_f^-, h_f),$$
$$B^+ = \sigma(B_f^+, h_f) \qquad B^- = \sigma(B_f^-, h_f).$$

3. The required BSP representation of the boundary face f is finally computed as:

$$\mathrm{BSP}(f) = (A^+ \;\&\; B^-) \mid (A^- \;\&\; B^+)$$

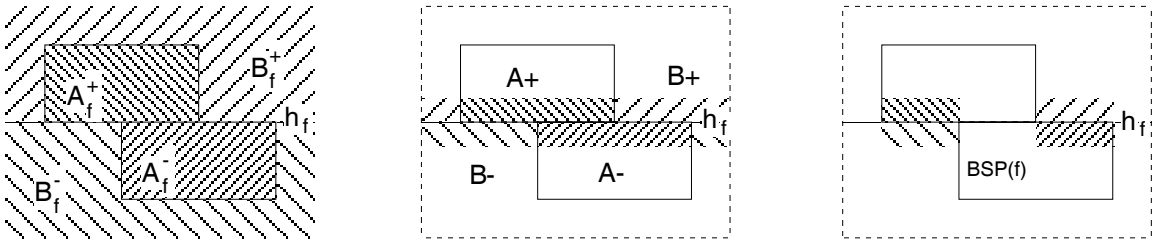

Figure 7: Graphical illustration of the computation of a face BSP.

Some motivation of the procedure for computing a face BSP tree is given in the sequel. Steps 1 is needed to compute the result when the input polyhedron is non-convex. Is such a case some interior space can be contained both in the "above" and in the "below" halfspaces of the face f. Step 2 performs the required σ operations using as section hyperplane the affine support h_f of the f face. Notice that this computation always contains the special case 4 (elimination of affine support) of Section 3.1. Step 3 is needed to bring together the portions of a face possibly associated to the two opposite orientations of the exterior surface. See Figure 7, where the computation of a face BSP is displayed with reference to a 2D case.

3.3 Boundary BSP

Let be given a polyhedron P in $\mathcal{E}^d$.

Definition 4 (Boundary BSP) *We call* Boundary BSP *a set of pairs* $(\mathrm{BSP}(f), M_f)$ *where f is any $(d-1)$-face of P,* $\mathrm{BSP}(f) \in \mathcal{BSP}^{d-1}$, *and M_f is the product of two affine maps. The first map is the identity embedding of $\mathcal{E}^{d-1}$ into the $x_d = 0$ subspace of $\mathcal{E}^d$; the second map is from the $x_d = 0$ subspace to the hyperplane h_f, affine support of f in $\mathcal{E}^d$.*

[1] Obviously, directly inside the represented object, conforming to its BSP.

Example 2
A solid polyhedron Q, represented as $\mathrm{BSP}(Q) \in \mathcal{E}^3$, is shown in Figure 8. The solid is generated as the Boolean union of three translated and/or rotated parallelepipeds. In Figure 8 it is possible to distinguish the four full cells associated to the BSP representation of Q. Figure 9 shows the full cells of the BSP representation of four boundary faces. Notice that they are unconvex, non 2-manifold and even (face dotted in Figure 9c) non coherently oriented.[2] Each of the four pictures of Figure 9 shows the cells of the BSP of a face of Q as embedded in $\mathcal{E}^3$.

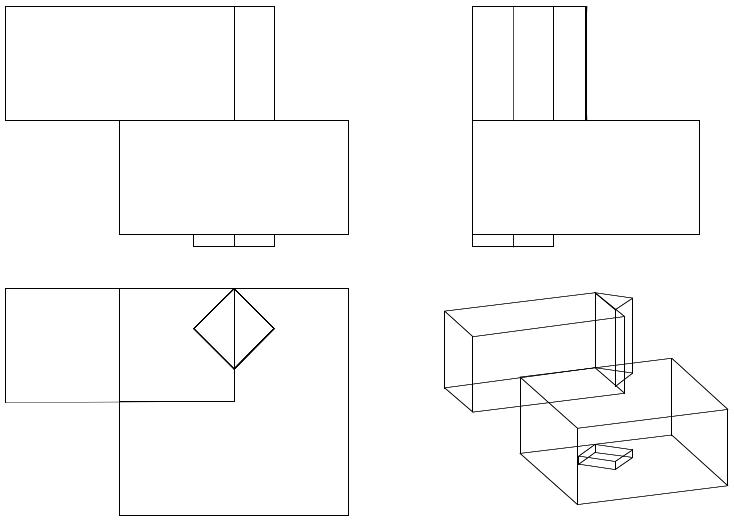

Figure 8: Three orthographic projections and one dimetric projection of a 3D solid.

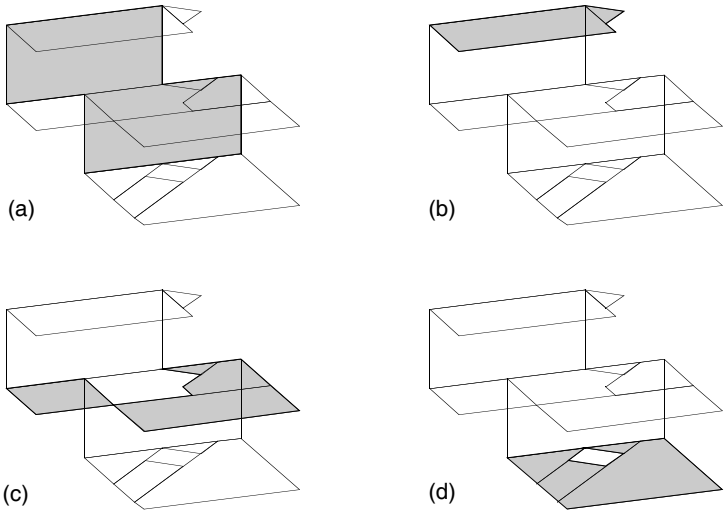

Figure 9: The face BSP cells of four boundary faces of the solid displayed in Figure 8.

[2]Using the standard terminology of algebraic topology we should say "non orientable".

4 From Boundary BSP to Decompositive BSP

The goal of the algorithm given in this section is to transform a boundary BSP, i.e. a collection of "face BSP" trees and embedding maps, which describe the boundary of a d-polyhedron P in $\mathcal{E}^d$, into a single tree BSP(P) which describes a cell decomposition of P.

Two main steps can be identified in the computation:

1. First, for each face f of P, a "stripe BSP" tree BSP(S_f) is properly built starting from BSP(f), where S_f is a solid unbounded stripe generated by extruding the projection of f into a coordinate subspace of dimension $d-1$.

2. Then, the Boolean XOR of all such stripes is evaluated, so giving the desired BSP representation of P:

$$\mathrm{BSP}(P) = \bigwedge_{f \in F(P)} \mathrm{BSP}(S_f), \tag{1}$$

   where $F(P)$ is the set of boundary faces of P and BSP(S_f) is the BSP tree which represents the solid unbounded stripe associated to the face f.

4.1 Stripe BSP

It is here discussed how to transform the BSP representation of a face of the input polyhedron into an unbounded solid stripe (once again represented as a BSP tree) to be later combined with stripes associated to the other faces.

Given an origin and an orthonormal basis $\{e_i\}$ in $\mathcal{E}^d$, let denote with h^i the coordinate hyperplane normal to e_i.

Definition 5 (Projection) *The i-th* projection *of a BSP tree embedded in a hyperplane of $\mathcal{E}^d$ is a mapping*

$$\Pi_i : \mathcal{BSP}^{d-1} \times \Re^{d+1} \to \mathcal{BSP}^{d-1}, \qquad i = 1, \ldots, d$$

such that

$$\Pi_i(A, h^i) = A.$$

The tree $\Pi_i(A, h)$, where $A \in \mathcal{BSP}^{d-1}$is embedded in the hyperplane h of $\mathcal{E}^d$, is computed as follows. For all hyperplanes h_ν in A, (the embedded) h_ν is substituted by the element of the hyperplane boundle

$$h + \lambda h_\nu = 0, \qquad \lambda \in \Re,$$

whose normal is perpendicular to e_i.

In other words, given a face BSP tree A with support hyperplane h, each hyperplane equation h_ν in A is transformed into the hyperplane (member of the bundle generated by h and h_ν) which is perpendicular to h^i. This operation results in a BSP tree in $\mathcal{BSP}^d$ projected in such i-th coordinate subspace (see Figure 11). After the elimination of the i-th coordinate, the output tree will belong to $\mathcal{BSP}^{d-1}$.

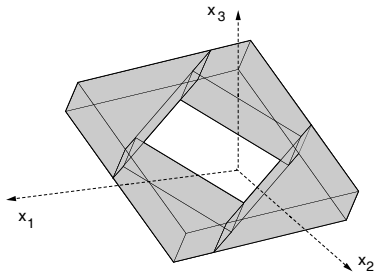

Figure 10: The BSP decomposition of a 3D solid in general position.

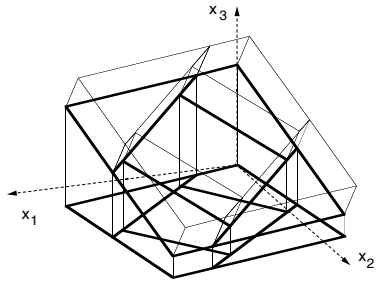

Figure 11: A face BSP and its Π_3 projection.

Example 3
In Figure 10 the BSP of a 3D solid in general position in $\mathcal{E}^3$ is shown. This example solid is used to show the generation of a Stripe BSP associated to its bottom face. In particular, Figure 11 shows both the bottom face f and the projection $\Pi_3(BSP(f), h_f)$.

Definition 6 (Extrusion) *The* extrusion *of a BSP tree is a unary operation*

$$\mathrm{Extr} : \mathcal{BSP}^d \to \mathcal{BSP}^{d+1},$$

such that

$$\Pi_{d+1}(\sigma(\mathrm{Extr}(A), h), h) = A,$$

for all $A \in \mathcal{BSP}^d$ *and all hyperplanes* h *in* $\mathcal{E}^{d+1}$.

So, if $A \in \mathcal{BSP}^d$, then the notation $\mathrm{Extr}(A)$ denotes the BSP tree associated to the partition of $\mathcal{E}^{d+1}$ defined as the set of cells

$$\{b_i \mid b_i = a_i \times \Re\}$$

where a_i is any d-cell in the partitioning of $\mathcal{E}^d$ induced by A.

Let be given a tree $F \in \mathcal{BSP}^{d-1}$, embedded in a hyperplane h of $\mathcal{E}^d$. As a preliminary test, check for coherent orientation of h by computing the dot product $n(h) \cdot e_d$, where $n(h), e_d$ are the normal vector to h and the last element in a base $\{e_i\}$, respectively. If the dot product is negative h must be multiplied by -1.

Definition 7 (Stripe BSP) *We call* Stripe BSP *the tree* $S \in \mathcal{BSP}^d$ *defined as the intersection of the halfspace BSP tree* H^+ *(defined in Section 3.1), with the BSP tree obtained by extruding the projection of* F *in a coordinate subspace:*

$$S = H^+ \ \& \ (\mathrm{Extr} \circ \Pi_d)(F, h)$$

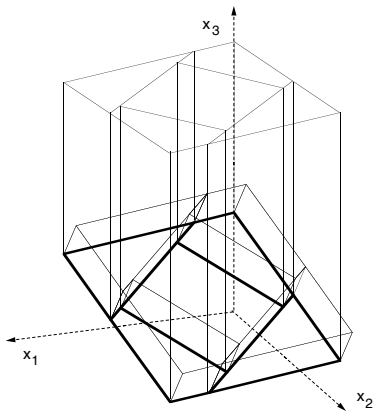

Figure 12: The Stripe BSP associated to a face.

Example 4
In Figure 12 it is shown the Stripe BSP associated to the bottom face of the 3D solid defined in the Example 3. Such figure shows a truncated solid, but it should be considered that the solid stripe is actually unbounded along the d-th coordinate direction.

4.2 XOR tree evaluation

It is known, from the Jordan's theorem, that the number of intersection points between a ray and the boundary of a polyhedron is odd if the ray is shot from the interior region, even if it is shot from the exterior region. The observation that the XOR of a number n of instances of a set A is either the empty set or the set A itself, depending on the parity (even or odd) of n, motivates the approach here discussed.

In particular, we extend to d-dimensional solids the algorithm [3], so generating a decompositive BSP from the set of stripes associated to the boundary $(d-1)$-faces, according to the formula:

$$\mathrm{BSP}(P) = \bigwedge_{f \in F(P)} \mathrm{BSP}(S_f). \tag{2}$$

Notice that the XOR is a commutative and associative operation. Due to associativity, any pattern of parenthesis can be used to pair-wise compute the sequence of XOR operations of Equation 2. Any such pattern will result in a different binary tree of the computation.

Definition 8 (XOR Tree) *We call XOR tree of dimension d a CSG tree where each internal node is the XOR set operation and each leaf node is a stripe BSP of the same dimension d.*

We remember that a CSG (Constructive Solid Geometry) tree is a binary tree with Boolean set operations as internal nodes and primitive objects as leaf nodes.

The XOR tree representation is traversed with Boolean traversal [7, 2] to generate a BSP tree of the result. Later on, a set of convex cells may be produced with standard BSP traversal. Due to XOR commutativity, no special ordering of faces is needed in order to compute the resulting BSP. This approach is hence well suited for parallel implementations, since the computing load can be easily distributed between different processors.

```
algorithm QuickXOR
                ( faces: array of face BSP, n: number of faces ): BSP tree;
    return QuickXORProc( faces, 0, n-1 );
end algorithm

algorithm QuickXORProc
                ( faces: array of face BSP, i, j: faces range ): bsp tree;
    t1, t2: bsp tree;
    if j-i == 0
        return StripeBSP( faces[i] );
    if j-i == 1
        t1 = StripeBSP( faces[i] );
        t2 = StripeBSP( faces[j] );
        return XOR( t1, t2 );
    t1 = QuickXORProc( faces, i, (i+j)/2 );
    t2 = QuickXORProc( faces, 1+(i+j)/2, j );
    return XOR( t1, t2 );
end algorithm
```

Figure 13: The algorithm QuickXOR

The pseudocode of an evaluation algorithm of XOR trees called QuickXOR, since it strongly resembles the Quick Sort of an array, is given in Figure 13.

5 Conclusion

This paper has discussed a both ways transformation between a boundary and a decompositive BSP representation of dimension-independent polyhedra. The presented algorithms can be used to generate a cell-decomposition of a d-polyhedron starting from the BSP trees of its boundary faces, as well as to generate such boundary BSP trees starting from its standard BSP representation. Such an approach may be useful to calculate the k-skeletons of a d-polyhedron, $0 \leq k \leq d$, as well to give insight into the problem of computing non-regularized Booleans over multidimensional polyhedra, which is our ultimate goal. Since the presented approach does not depend from neither the orientation nor the ordering of boundary faces, it easily allows for parallel implementations.

References

[1] Bajaj, C. L., and Dey, T. K. Convex Decomposition of Polyhedra and Robustness. *SIAM Journal on Computing*, 21(2), April 1992, 339–364.

[2] Baldazzi, C. *Boolean Set Operations on Multidimensional Polyhedra.* Graduation Thesis, Università "La Sapienza", Rome, Italy, February 1996. (In Italian).

[3] Baldazzi, C., and Paoluzzi, A. *From Polyline to Polygon via XOR tree.* Tech. Rep. INF-04-96, Dip. Disc. Scient., Università Roma Tre, Rome, Italy, May 1996.

[4] Fuchs, H., Kedem, Z. M., and Naylor, B. F. On visible surface generation by a priori tree structures. *Computer Graphics*, 14(3), August 1980, 124–133. (Proc. of ACM Siggraph'80).

[5] Günther, O., and Wong, E. Convex polyhedral chains: A representation for geometric data. *Computer Aided Design*, 21(3), March 1989, 157–164.

[6] Mäntylä, M. *An Introduction to Solid Modeling.* Computer Science Press, Rockville, MD, 1988.

[7] Naylor, B. F., Amanatides, J., and Thibault, W. Merging BSP trees yields polyhedral set operations. *Computer Graphics*, 24(4), August 1990, 115–124. (Proc. of ACM Siggraph'90).

[8] Naylor, B. F. Binary space partitioning trees as an alternative representation of polytopes. *Computer Aided Design*, 22(4), April 1990, 250–252.

[9] Paoluzzi, A., Pascucci, V., and Vicentino, M. Geometric programming: A programming approach to geometric design. *ACM Transactions on Graphics*, 14(4), July 1995, 266–306.

[10] Pascucci, V., Ferrucci, V., and Paoluzzi, A. Dimension-Indipendent Convex-Cell based HPC: Skeletons and Product. *International Journal of Shape Modeling*, 2(1), January 1996, 37–67.

[11] Requicha, A. A. G. Representations for rigid solids: Theory, methods and systems. *ACM Computing Surveys*, 12(4), December 1980, 437–464.

[12] Rossignac, J. R. Through the Cracks of the Solid Modeling Milestone. In *From Object Modelling to Advanced Visual Communication*, S. Coquillart, W. Straßer, and P. Stucki, Eds., Springer-Verlag, 1994, 1–75.

[13] Shapiro, V., Vossler, D. L. Separation for Boundary to CSG Conversion. *Computer Graphics*, 12(1), January 1993, 35–55. (Proc. of ACM Siggraph'93).

[14] Takala, T. Geometric Boundary Modelling Without Topological Data Structures. In *Eurographics '86*, A.A.G. Requicha, Ed., North Holland, 1986.

[15] Thibault, W., and Naylor, B. F. Set operations on polyhedra using binary space partitioning trees. *Computer Graphics*, 21(4), July 1987, 153–162. (Proc. of ACM Siggraph'87).

[16] Weiler, K. The radial edge structure: A topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications*, M. J. Wozny, H. W. McLaughlin, and J. L. Encarnaçao, Eds., North Holland, 1988, 3–36.

[17] Woo, T. A Combinatorial Analysis of Boundary Data Structure Schemata. *IEEE Computer Graphics & Applications*, 5(3), March 1985, 19–27.