



PERGAMON

Computers & Graphics 23 (1999) 429–437

COMPUTERS  
& GRAPHICS

Technical Section

## Tribox bounds for three-dimensional objects

A. Crosnier<sup>a,\*</sup>, J.R. Rossignac<sup>b</sup>

<sup>a</sup>LIMM, 161 rue Ada, 34392 Montpellier Cedex 5, France

<sup>b</sup>GVU Center, College of Computing, Georgia Institute of Technology, Atlanta, USA

### Abstract

A convex hull  $H(S)$  of a 3D set  $S$  is rarely used to accelerate interference detection or as a substitute for rendering small projection of  $S$ , because it typically has too many faces. An axis aligned bounding box  $B(S)$  is cheaper to display and more effective at detecting that two distant objects are clearly disjoint, but is a conservative approximation of  $S$ . We propose to use a tribox  $T(S)$  as a compromise.  $T(S)$  is a tighter bound than  $B(S)$  and is cheaper to display and to test for interference than  $H(S)$ .  $T(S)$  is the intersection of three bounding boxes formed in three different coordinate systems, each obtained by rotating the global coordinate system by  $45^\circ$  around one of the principal axes.  $T(S)$  has at most 18 polygonal faces. We present an algorithm for computing the boundary of  $T(S)$ , given its 18 parameters – the endpoints of the intervals containing the projection of the vertices of  $S$  onto 9 directions. Given the 18 projection bounds, our algorithm requires only 24 shifts and at most 112 additions. We also describe a simple technique for building a hierarchical multi-resolution representation, which approximates a 3D shape at different level of details by unions of triboxes. © 1999 Elsevier Science Ltd. All rights reserved.

### 1. Introduction

A geometric bound (also called bounding volume) is a simple solid (box, sphere) that contains a given shape  $S$ . Such bounds have been used to speed up interference detection, rendering and swept volume generation. Two objects are disjoint if their bounds are. Complex models that appear small on the screen may often be replaced by simplified shapes [1] for interactive graphic applications. Rectangular bounding boxes are popular for both purposes, but bounding spheres and convex hulls have also been considered.

An axis-aligned mini-max bounding box,  $B(S)$ , of a polyhedron  $S$  may be constructed efficiently by finding the minima and maxima of the vertex coordinates along each axis.  $B(S)$  may be rendered efficiently. Interferences of mini-max boxes may be tested by performing at most six comparisons. Yet mini-max boxes are rarely used in

practice, because they are often much larger than the shape they bound. The most blatant example is a long and thin cylinder with an axis in the (1, 1, 1) direction. Such boxes are usually not acceptable as graphic substitutes and often intersect although the objects they bound do not.

Given  $B(S)$ , a bounding sphere for  $S$  may be obtained by placing its center at the center of  $B(S)$  and by setting its radius to be the largest distance between this center and the vertices of  $S$ . In general, the radius of such a sphere is only 5% larger than the radius of the smallest sphere containing  $S$ , which is significantly more expensive to compute [2]. Sphere/sphere intersection is easily tested, but spheres are more expensive to display than boxes. Their main advantage is their invariable under rotation, which is important for animation. Still, spheres are rarely tight bounds and thus are rarely used as graphic substitute or as filters for interference tests.

The convex hull,  $H(S)$ , of  $S$  may be computed efficiently [3] and is the tightest convex bound for  $S$ . Unfortunately, convex hulls of complex objects may be bounded by large numbers of polyhedral faces, which make them expensive to render and inappropriate for interference calculations.

\*Corresponding author. Tel.: + 33-67-41-85-85; fax: + 33-67-41-85-00.

E-mail address: crosnier@lirmm.fr (A. Crosnier)

Given a set  $D$  of directions, a constrained convex envelope,  $C(S, D)$ , of a polyhedron  $S$  is the intersection of a finite number of slabs. The slab for a direction  $\mathbf{d}$  of  $D$  is the largest set that has the same projection as  $S$  on a line parallel to  $\mathbf{d}$ . Thus, the slab is the space between two parallel planes with normals parallel to  $\mathbf{d}$ . It may be computed by finding the minimum and maximum projections of the vertices of  $S$  onto a line parallel to  $\mathbf{d}$ . Interference between  $C(S, D)$  and  $C(R, D)$  requires  $2k$  comparisons, where  $k$  is the number of directions in  $D$ . The cost of rendering  $C(S, D)$  is also proportional to  $k$ .  $B(S)$  is a particular case of  $C(S, D)$  for which  $D$  contains only the three principal directions of the global coordinate system. A larger set of directions for  $D$  will in general yield a tighter bound than  $B(S)$ . A  $C(S, D)$  with the 12 principal and diagonal directions has been explored at Stony Brook by Mitchell and his colleagues [4].

We introduce here a slightly simpler shape, called tribox and denoted  $T(S)$ . Its main advantage over the

work reported in [4] is the simplicity of the computation of its boundary. General techniques for computing the faces of a convex polyhedron from its definition as the intersection of planar half-spaces are slow. The special purpose techniques, developed in [4] for the 24-sided constrained convex envelopes, are faster, but may still prove too expensive for some interactive applications, where bounds on evolving shapes or groups of shapes must be maintained. In contrast, building the bounding faces of  $T(S)$  takes only 112 additions and 24 shifts, once the projection bounds on the nine directions have been established.

$T(S)$  is a  $C(S, D)$  with a set  $D$  of 9 directions: the 3 principal axes and the 6 diagonals:  $\mathbf{x} + \mathbf{y}$ ,  $\mathbf{x} - \mathbf{y}$ ,  $\mathbf{y} + \mathbf{z}$ ,  $\mathbf{y} - \mathbf{z}$ ,  $\mathbf{z} + \mathbf{x}$ , and  $\mathbf{z} - \mathbf{x}$  (see Figs. 1 and 2). In general,  $T(S)$  is significantly tighter than  $B(S)$  and may often be used as a visual substitute for low levels-of-detail. Although it is at most three times more expensive to test whether  $T(S)$  and  $T(R)$  interfere than to test whether  $B(S)$  and  $B(R)$  do, a significantly larger number of detailed interference tests

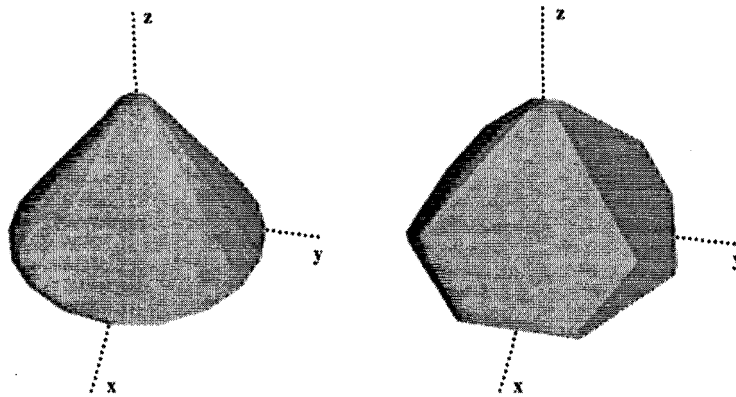


Fig. 1. A 3D object and its tribox.

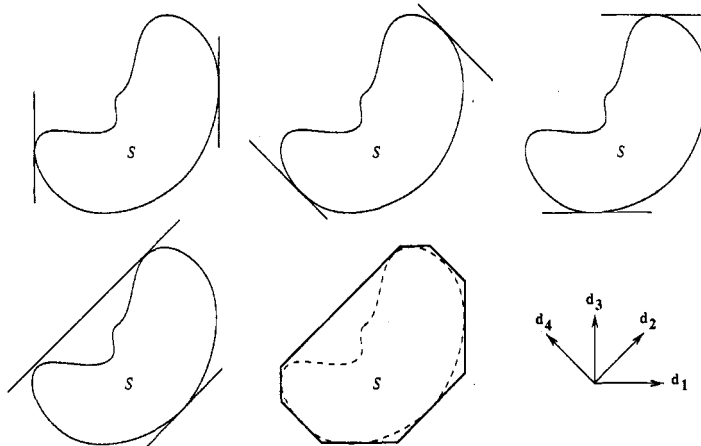


Fig. 2. Example of the construction of the tribox in 2D.

may be avoided if triboxes, rather than mini-max boxes, are used to prune out disjoint candidate pairs.

Shapes with large cavities or holes are poorly approximated by convex outer bounds. A multi-resolution hierarchy of simple bounds (such as arbitrarily oriented boxes) may be used to provide an adaptive level-of-detail envelope around the surface of the object [5]. We present an efficient procedure for recursively subdividing a polyhedron to generate a hierarchy of triboxes (all with the same set of directions  $D$ ) that provide increasingly finer approximations of the original object.

## 2. Constrained convex envelope: the tribox

In this section, we provide a more formal definition of triboxes and study the structure of their boundary.

### 2.1. Definition

Let  $S$  be a polyhedron or a finite set of points of the three-dimensional Euclidian space,  $E^3$ . Given the following set of 9 directions:  $D = \{\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{x} + \mathbf{y}, \mathbf{x} + \mathbf{z}, \mathbf{y} + \mathbf{z}, \mathbf{x} - \mathbf{z}, \mathbf{y} - \mathbf{x}, \mathbf{z} - \mathbf{y}\}$ , we wish to compute the convex polyhedron that bounds the region corresponding to the intersection of the 18 half-spaces associated with the bounds of  $S$ , which are defined by the min and max values of the projections of the input vertices or points of  $S$  on the directions of  $D$ . The tribox  $T$  of a set  $S$  is thus defined as the intersection of 9 pairs  $(H_{\min}(S, \mathbf{d}_i), H_{\max}(S, \mathbf{d}_i))$  of linear half-spaces that contain  $S$  and have each a fixed orientation according to their outward pointing normals defined by  $D$ . We have:

$$T(S, D) = \bigcap_{\mathbf{d}_i \in D} H_{\min}(S, \mathbf{d}_i) \cap H_{\max}(S, \mathbf{d}_i). \quad (1)$$

Fig. 2 illustrates the construction of the tribox in 2D. In that case, the set  $D$  is composed of the four following directions:  $\mathbf{x}, \mathbf{y}, \mathbf{x} + \mathbf{y}, -\mathbf{x} + \mathbf{y}$ . The projection of the set  $S$  on the directions of  $D$  exhibits four slabs whose intersection defines the 2D tribox.

### 2.2. Boundary of the tribox

The topology of the graph representing the face-edge-vertex incidence relations of the boundary of the tribox depends on the values of the 18 min-max projection-bound parameters that define the tribox.

That graph and the location of its vertices could be computed through a sequence of Boolean operations by considering that the tribox is the intersection of the 18 half-spaces. They could also be computed in dual space, as the convex hull of points that each correspond to one of the half-spaces [6].

Instead of these approaches, we exploit the particular configuration of the planes and use a special procedure to

compute the boundary of the tribox given its 18 projection-bound parameters. The essence of our approach may be better understood by considering that the tribox is obtained by chamfering the edges of a block  $\mathcal{B}$  whose faces are orthogonal to the principal axes  $\mathbf{x}, \mathbf{y}$ , and  $\mathbf{z}$ . The topology of the entire tribox may be computed by analyzing the local topology at each corner of the chamfered block, independently of the other corners. The topological configurations of the eight corners may be easily composed to define the faces, edges, and vertices of the tribox. The validity of this divide-and-conquer approach is justified as follows.

Consider the 18 planes that bound the half-spaces whose intersection is the tribox. Each one of these 18 planes must intersect the boundary of the tribox in at least one point, otherwise, we could have used a smaller half-space in the same direction. Our algorithm creates 18 faces, one per plane, although in singular situations, some of these faces or some of their edges may degenerate to a single point. Such degenerate edges and degenerate zero-area faces may be removed through an optional post-processing step, which may be important when the resulting boundary is to be used for repeated rendering or for geometric calculations.

Each face of the tribox is defined by its bounding loop – the ordered list of its vertices. For each face, that list is assembled from two or four sub-lists computed independently for different corners. Each sub-list contains two or four vertices. Each corner is responsible for computing the location of its four vertices and for arranging their references into 6 sub-lists, one per incident face. These computations and the decisions on how to form the sub-lists are reduced to simple addition and shift operations and to three comparisons as explained below.

At each corner of  $\mathcal{B}$ , the set of planes contributing to the boundary of the tribox is defined by three main planes  $\{h_A, h_B, h_C\}$  associated with the three principal directions  $\mathbf{x}, \mathbf{y}$  and  $\mathbf{z}$ , and by three chamfer planes  $\{h_{AB}, h_{AC}, h_{BC}\}$  associated with three diagonals. Fig. 3 represents a configuration of the planes  $\{h_A, h_{AB}, h_B, h_{BC}, h_C, h_{AC}\}$ , respectively, associated with the directions  $\mathbf{x}, \mathbf{x} + \mathbf{y}, \mathbf{y}, \mathbf{y} + \mathbf{z}, \mathbf{z}, \mathbf{x} + \mathbf{z}$ .

A naive analysis of this arrangement would require the computation and the classification of a number of vertices equal to:  $n_0 = \binom{6}{3} = 20$ . Instead, we reduce this process to the classification of a single point, the intersection  $\mathbf{p} = (x_p, y_p, z_p)$  of the three chamfers  $\{h_{AB}, h_{BC}, h_{AC}\}$  that meet at the corner, against the principal planes (Fig. 4). There are only four possible configurations if one ignores the singular cases and accepts to represent degenerate zero-length edges:  $\mathbf{p}$  is either in all three half-spaces or outside of only one of them (see Fig. 5). Other cases are impossible, because if  $\mathbf{p}$  was outside of two of the principal half-spaces, the chamfers that cut the corner between these two planes would not be a tight bound.

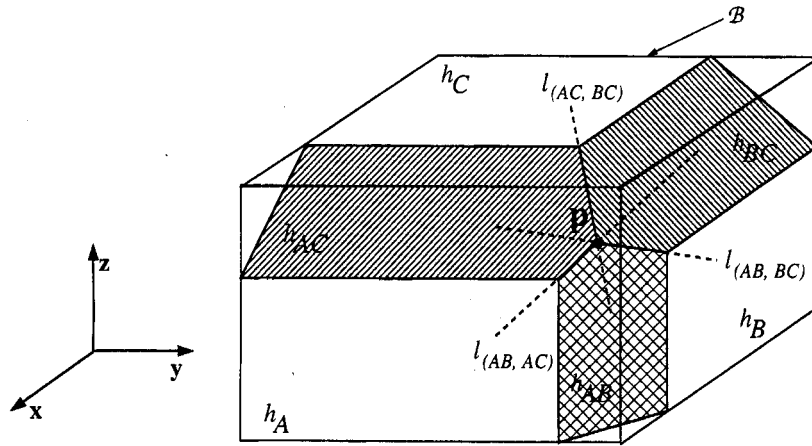


Fig. 3. Set of planes abutting a corner.

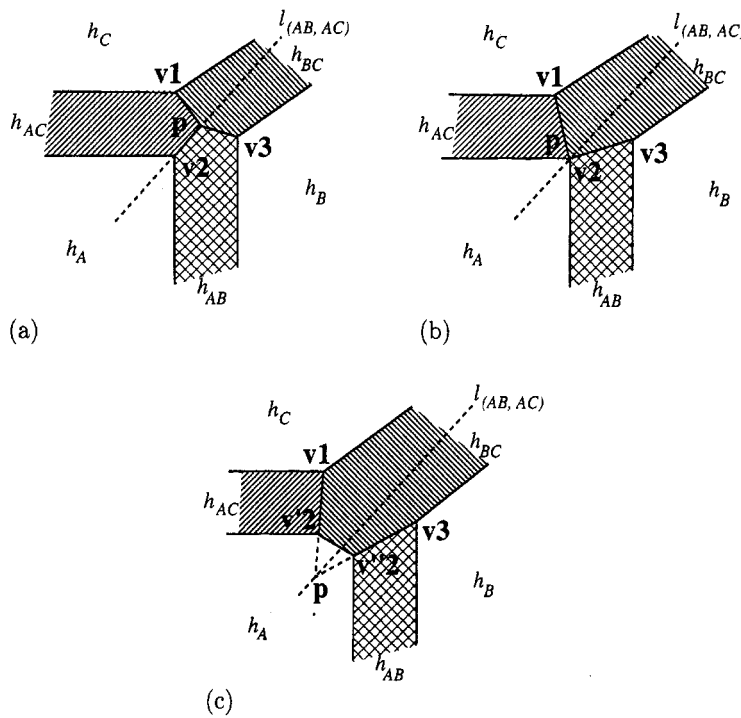


Fig. 4. Change of the local topology at a corner.

To illustrate this situation, consider that  $p$  is also the intersection point of three lines  $l_{(AB, AC)}$ ,  $l_{(AB, BC)}$ ,  $l_{(AC, BC)}$ , respectively, associated with the three couples of intersecting chamfers (see Fig. 3). If we change one of the 18 input parameters to displace one of the chamfers, for

example  $h_{BC}$ , the point  $p$  moves on the line  $l_{(AB, AC)}$  that is not affected by the translation of  $h_{BC}$  (see Fig. 4). When  $p$  passes a principal plane, as illustrated on Fig. 4(c), a change of the local topology of the tribox appears. Complete topology of the tribox is represented in Fig. 6.

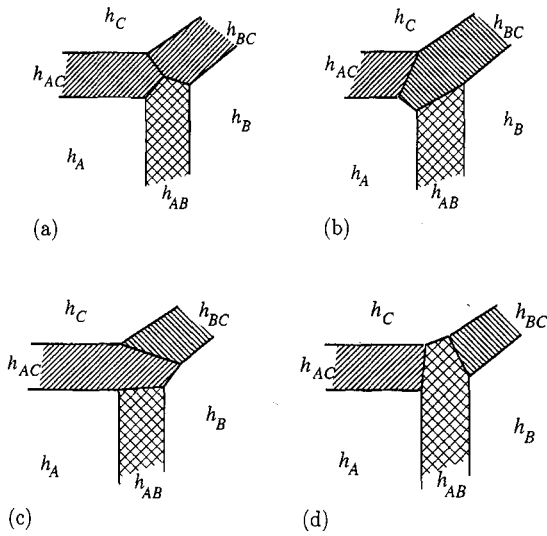


Fig. 5. Topological configurations at a corner.

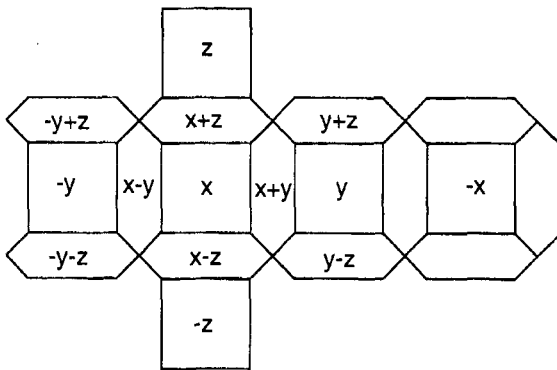


Fig. 6. Complete topology of the tribox.

### 3. Boundary evaluation of the tribox

The boundary evaluation aims at computing the faces of the tribox. Depending on the application, the boundary evaluation takes as input either the set of points  $S$  associated with the object, or the set of the projection bounds.

For an exact tribox, the projection bounds may be computed by traversing all the vertices and by updating the min and max values of their projections on 9 lines. We denote afterwards  $\alpha_A, \alpha_B$  and  $\alpha_C$  the bounds of the principal planes  $h_A, h_B$  and  $h_C$ , and  $\alpha_{AB}, \alpha_{AC}$  and  $\alpha_{BC}$  the bounds of the chamfer planes  $h_{AB}, h_{AC}$  and  $h_{BC}$ .

For graphic purposes, each tribox may be computed in an object's local coordinate system or even using an orientation that minimizes the overall volume or area of the tribox. However, for interference detection, it may be

useful to represent each tribox bound in the same coordinate system.

When objects or groups of objects for which tribox bounds are important are rotated frequently, it may be beneficial to precompute a convex hull and use its vertices to update the tribox. If an approximate tribox is acceptable, the vertices of an initial tribox may be used to compute triboxes for the various orientations of the model.

Given the 18 projection bounds, the topology of the boundary at each corner may be easily characterized by calculating the intersection point  $\mathbf{p}$  and by testing it against the three principal planes. For Fig. 3, the three chamfer planes are characterized by the following equations:  $x + y = \alpha_{AB}$ ,  $x + z = \alpha_{AC}$  and  $y + z = \alpha_{BC}$ , and the intersection point may be computed as:

$$x_p = \frac{\alpha_{AB} + \alpha_{AC} - \alpha_{BC}}{2}, \tag{2}$$

$$y_p = \frac{\alpha_{AB} - \alpha_{AC} + \alpha_{BC}}{2}, \tag{3}$$

$$z_p = \frac{-\alpha_{AB} + \alpha_{AC} + \alpha_{BC}}{2}. \tag{4}$$

If we test  $\mathbf{p}$  against the three principal planes  $x = \alpha_A, y = \alpha_B, z = \alpha_C$  of  $\mathcal{B}$ , we can distinguish four cases:

- case 1:  $x_p < \alpha_A, y_p < \alpha_B$  and  $z_p < \alpha_C$  (see Fig. 5(a)),
- case 2:  $x_p > \alpha_A, y_p < \alpha_B$  and  $z_p < \alpha_C$  (see Fig. 5(b)),
- case 3:  $x_p < \alpha_A, y_p > \alpha_B$  and  $z_p < \alpha_C$  (see Fig. 5(c)),
- case 4:  $x_p < \alpha_A, y_p < \alpha_B$  and  $z_p > \alpha_C$  (see Fig. 5(d)).

Given that the four situations are exclusive, the test for each corner may be structured as follows:

```

if  $x_p > \alpha_A$  then "case 2",
else if  $y_p > \alpha_B$  then "case 3",
else if  $z_p > \alpha_C$  then "case 4",
then "case 1".
    
```

We illustrate this process by first considering a typical situation depicted in Fig. 4a, where point  $\mathbf{p}$  is located inside all principal half-spaces. Thus, the following inequalities:  $x_p < \alpha_A, y_p < \alpha_B$  and  $z_p < \alpha_C$  are satisfied, and three vertices  $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ , which, with  $\mathbf{p}$ , contribute to the boundary of the tribox at the corner, may be computed as follows:

- $\mathbf{v}_1$  that is the intersection point of the planes:  $x + z = \alpha_{AC}, y + z = \alpha_{BC}$  and  $z = \alpha_C$  is defined by
 
$$\mathbf{v}_1 = (\alpha_{AC} - \alpha_C, \alpha_{BC} - \alpha_C, \alpha_C), \tag{5}$$

- $\mathbf{v}_2$ , intersection point of the planes:  $x = \alpha_A, x + y = \alpha_{AB}$  and  $x + z = \alpha_{AC}$ , is defined by
 
$$\mathbf{v}_2 = (\alpha_A, \alpha_{AB} - \alpha_A, \alpha_{AC} - \alpha_A) \tag{6}$$

- and  $\mathbf{v}_3$ , intersection point of the planes:  $x + y = \alpha_{AB}$ ,  $y = \alpha_B$  and  $y + z = \alpha_{BC}$ , is defined by

$$\mathbf{v}_3 = (\alpha_{AB} - \alpha_B, \alpha_B, \alpha_{BC} - \alpha_B). \quad (7)$$

A different topology is obtained if we displace  $\mathbf{p}$  past a principal plane (for example  $h_A$  in Fig. 4(c)). Such situation is easily detected when the conditions:  $x_p > \alpha_A$ ,  $y_p < \alpha_B$  and  $z_p < \alpha_C$  are satisfied. In that case, the point  $\mathbf{p}$  located outside of  $\mathcal{B}$  does not contribute to the boundary of the tribox, and the four other vertices  $\mathbf{v}_1, \mathbf{v}'_2, \mathbf{v}''_3$  contributing to the boundary of the tribox at that corner can be calculated as follows:

- $\mathbf{v}_1$  and  $\mathbf{v}_3$  are defined by the expressions (5) and (7),
- $\mathbf{v}'_2$  is the intersection point of the planes:  $x = \alpha_A$ ,  $x + z = \alpha_{AC}$  and  $y + z = \alpha_{BC}$  and it may be computed as follows:

$$\mathbf{v}'_2 = (\alpha_A, \alpha_{BC} - \alpha_{AC} + \alpha_A, \alpha_{AC} - \alpha_A) \quad (8)$$

- and  $\mathbf{v}''_3$ , intersection point of the planes:  $x = \alpha_A$ ,  $x + y = \alpha_{AB}$  and  $y + z = \alpha_{BC}$ , is defined by

$$\mathbf{v}''_3 = (\alpha_A, \alpha_{AB} - \alpha_A, \alpha_{BC} - \alpha_{AB} + \alpha_A). \quad (9)$$

The computation of each vertex that is the intersection point between one face from  $\mathcal{B}$  and two chamfer planes, requires only two additions.

Each face of the tribox is represented by a list of its bounding vertices ordered around the face. Our algorithm will compute these lists and the associated vertices. The vertex list of each principal face  $F$  will be the concatenation of vertices computed in the four sub-lists associated with  $F$ . The chamfer faces have only two sub-lists each. We reserve an array of four vertex-indices for each sub-list.

The sub-lists are filled in when processing the corners of the tribox. Each corner is responsible for one sub-list in each one of its six incident faces: three sub-lists of principal faces and three of chamfer faces. This correspondence is hard-coded. As a corner is analyzed, its four vertices are computed and appended into a vertex array. The indices for these vertices are then entered into the appropriate sub-list arrays and the vertex count for each sub-list is also stored. The indices and the counts are hard-coded for the various 1, 2, 3 and 4 cases of our corner-testing algorithm. Then, the vertex-indices for each sub-list are merged into a single array for each face.

For example, in Fig. 7, face  $F_1$  has four sub-lists  $L_{11}, L_{12}, L_{13}$ , and  $L_{14}$  and the chamfer face  $F_2$  has only two lists,  $L_{21}$  and  $L_{22}$ . Corner  $C_1$  is responsible for the lists  $L_{11}, L_{21}, L_{31}, L_{41}, L_{51}$ , and  $L_{61}$ . Corner  $C_2$  is responsible for the lists  $L_{12}, L_{22}, L_{32}, L_{72}, L_{82}$ , and  $L_{92}$ . Other corners and faces are omitted in this example.

When singular cases appear, the boundary may be optionally cleaned up by removing all the edges and zero-area faces. This option may be useful for improving the performances of certain applications, like rendering.

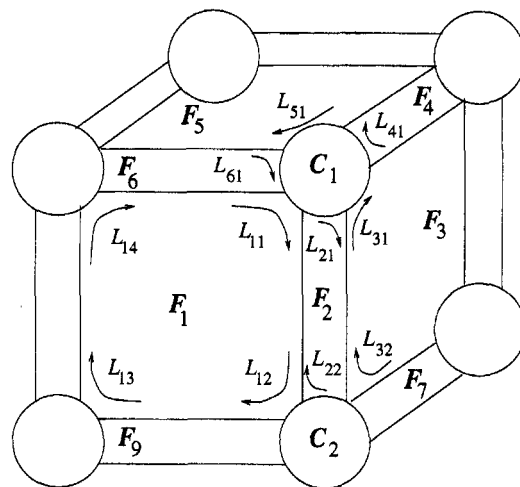


Fig. 7. Sub-lists associated with the faces.

On the other hand, when small faces appear in the boundary of the tribox, it is possible to perform a simplification phase that consists of applying a small outward displacement on the bounding planes in order to obtain a simplified tribox. The maximum displacement may be specified by a tolerance factor which allows to snap some of the projection bounds in order to force some faces to have zero area.

Our algorithm is much faster than computing a general intersection of 18 half-spaces. The boundary calculation of the tribox performs at most 24 tests. For each corner, we perform six additions and three shifts (which implement a multiplication by 2) to compute the intersection point  $\mathbf{p}$ . Then the coordinates of  $\mathbf{p}$  are compared to projection bounds. Depending on the classification of the corner we calculate the three or four vertices associated with the local topology at the corner, that requires 6 or 8 additions. In conclusion, the tests which determine the configuration of all 8 corners require between 96 and 112 additions and 24 shifts.

#### 4. Hierarchical triboxes

Tribox bounds perform well for convex shapes. For a shape  $S$  with large cavities, we have developed a recursive method, which computes a set of smaller covering triboxes that are mutually disjoint and whose union is usually a tighter bound for  $S$ . We explain the overall recursion and discuss the details of computing the vertex sets at each stage of the recursion.

Given a set  $S$  and its box tribox  $T(S)$ , we obtain a more accurate representation  $T(S) \cup T(S_2)$  of the object  $S$  if we split  $S$  by a plane  $h$  into  $S_1$  and  $S_2$ . We choose  $h$  to be orthogonal to one of the principal or diagonal directions in  $D$ .

Let  $H^+$  and  $H^-$  be the two open half-spaces defined by  $h$ . Let  $S_1 = S \cap H^+$  and  $S_2 = S \cap H^-$ .  
The difficulties of our approach are:

- the selection of  $h$ ,
- the evaluation of the criteria for stopping the recursive subdivision,

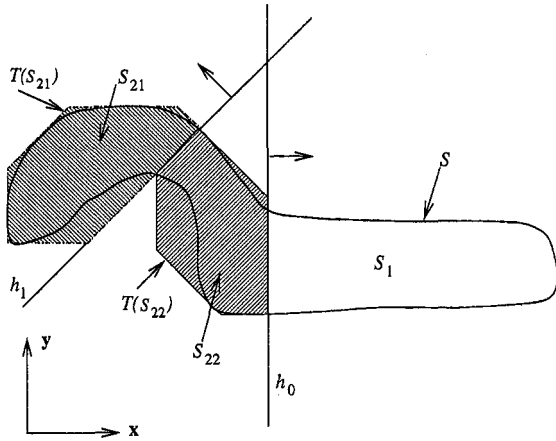


Fig. 8. Subdivision process.

- the computation of the projection bounds for  $S_1$  and  $S_2$ .

We do not have an efficient solution for selecting the best direction for  $h$  and for computing its optimal position. We have explored a simple greedy approach focussed on minimizing the total area of the boundaries of  $T(S_1)$  and  $T(S_2)$ . Although we believe that a more appropriate candidate set for  $h$  may be derived by identifying local extrema in all directions [7], our solution considered first the plane  $h$  that bisects the thickest slab of  $T(S)$ . The subdivision would stop when no split by a slab-bisecting plane would reduce the total area of the covering triboxes by a sufficient proportion.

Our recursive procedure splits  $S$  into small cells. Each such cell  $S_i$  is the intersection  $S \cap C$  of  $S$  with a convex and possibly unbounded set  $C$  that is the intersection of at most 18 half-spaces, each aligned with one of the principal or diagonal directions. We need to compute the vertices of  $S_i$  to update its projection bounds and compute the tribox. The vertices of  $S_i$  fall into the following categories:

- the vertices of  $S$  in  $C$ . These may be computed by simply classifying the vertices of  $S$  against the half-spaces of  $C$ ,

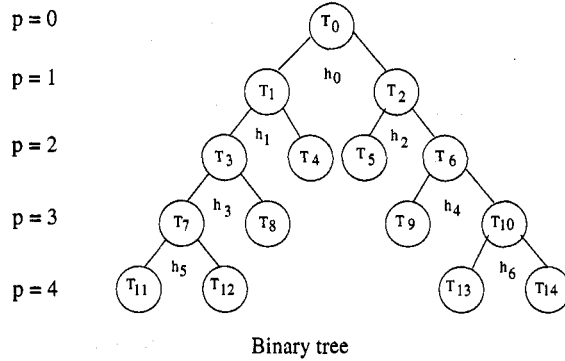
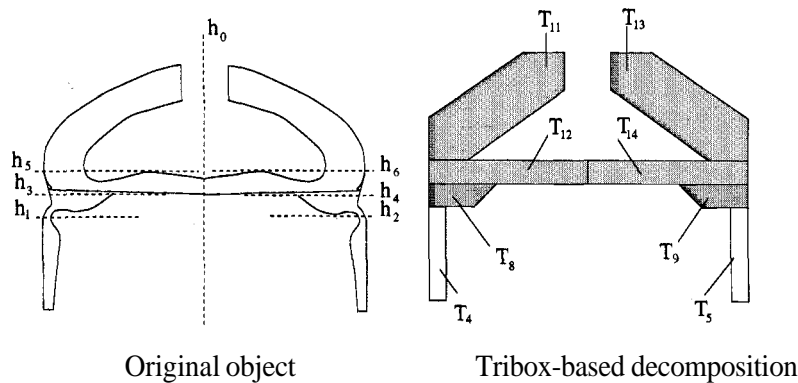


Fig. 9. Binary tree and tribox-based decomposition

- the vertices of  $C$  in  $S$ . These are easily obtained by computing the vertices of  $C$  using a simple extension of the tribox-boundary evaluation procedure outlined above and by classifying the resulting vertices against  $S$  using a ray-casting approach,
- the intersections of edges of  $S$  with faces of  $C$ . These are easily obtained by first intersecting edges of  $S$  with the planes bounding  $C$  and then testing these points against the other half-spaces of  $C$ . These tests are inexpensive because  $C$  has usually very few, and never more than 18, faces and because these faces have simple normals,
- the intersections of edges of  $C$  with faces of  $S$ . These may be computed by taking all the edges of  $C$  (which have tangent directions in a predefined set) and by intersecting them with the faces of  $S$ .

When the above split is performed recursively, there is no need to start from the whole set  $S$  at each level, but suffices to use the parent set instead. So the whole process is incremental and requires lower costs as the cells become simpler. The resulting vertices are used to incrementally update the projection bounds passed down the recursion.

In Fig. 8, the plane  $h_0$ , parallel to  $\mathbf{x}$ , splits  $S$  into the cells  $S_1$  and  $S_2$ . Then the plane  $h_1$ , parallel to  $\mathbf{x} + \mathbf{y}$ , splits  $S_2$  into the cells  $S_{21}$  and  $S_{22}$ . The tribox  $T(S_{22})$  is computed from the projection bounds associated with the cell  $S_{22}$ , that derives from the intersection  $S \cap C$ , where  $C = H_0^+ \cap H_1^+$ .

The result of the subdivision process may be stored into a hierarchical structure represented by a binary tree, as illustrated by the 2D example of Fig. 9. Each node  $i$  of the binary tree represents the tribox associated with cell  $S_i$ .

This tribox contains the triboxes associated with all the children of  $i$ .

Simplified representations of the original object can be constructed as the union of the triboxes associated with the leaves of an upper portion of the binary tree. The complexity/accuracy trade-off may be adjusted by changing the depth of this portion. For example, in Fig. 9, the collection of triboxes  $\tau = \{T_7, T_8, T_4, T_5, T_9, T_{10}\}$  is used as a tight bound.

Fig. 10 gives two examples of the use of the decomposition technique for rendering 3D objects. Fig. 10(a) compares the original object (left column) having 450 faces with three levels of detail (right column): simple tribox of

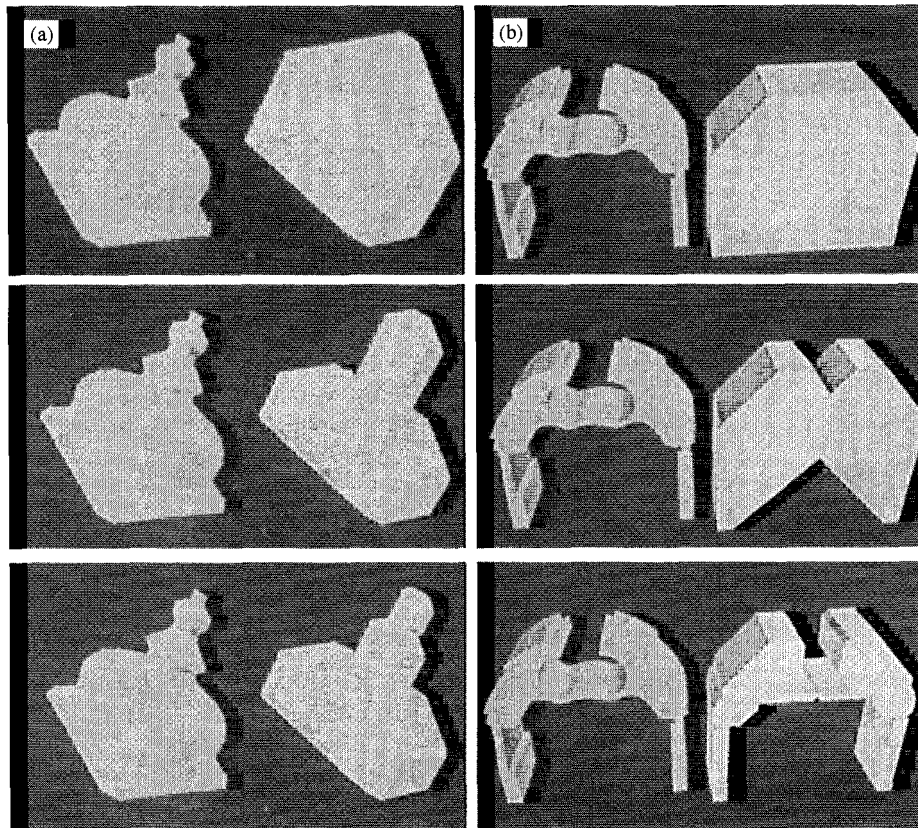


Fig. 10. Hierarchical triboxes resulting from 3D object decomposition.

Table 1  
Compression ratios

Example	$N$	$M$	Number of triboxes	$\tau$
Fig. 10(a)	450	14	1	32.14
		28	2	16.07
		46	3	9.78
Fig. 10(b)	950	8	1	118.75
		18	2	52.77
		72	8	13.19

the original object (top), two triboxes resulting from splitting the object by a horizontal plane (middle), three triboxes obtained from the previous model by splitting one cell by another plane (bottom). Fig. 10(b) uses the same structure to compare the original model (left column) discussed in Fig. 9 with increasing accuracy tribox-based decomposition. For these two examples, a simplification phase has been performed on all the triboxes used into the structures. For Fig. 10 the polyhedral compression  $\tau$ , expressed as the ratio of the number of faces of the original object ( $N = 450$  and  $950$  faces, respectively) over the total number of faces, denoted  $M$ , associated with the collection of triboxes is depicted in Table 1.

It is helpful to have an estimation of the approximation resulting from the use of the tribox, which can be defined according to how closely the tribox matches the original object. The Hausdorff distance is a useful tool for computing the error due to the approximation. In that case, the bound of the error can be expressed by

$$\rho = \max_{s_i \in S} [\max_{d_j \in A(s_i)} (s_i \cdot d_j)], \quad (10)$$

where  $A(s_i)$  denotes the cone of normals at the vertex  $s_i$  of  $S$ . The error can be controlled through a user-defined accuracy factor acting as a threshold for defining the

acceptable minimum distance between two vertices of the tribox.

## 5. Conclusion

Enclosing bounds based on bounding spheres and axis-aligned min-max boxes are too loose. Convex hulls are too complex. The tribox provides a tighter bound and is useful for graphic acceleration and interference detection. Its definition as a convex 3-polytope constrained to a predefined set of 9 directions requires only the knowledge of 18 parameters. The calculation of its boundary may be easily performed in real time because it requires only 24 shifts and at most 112 additions.

## References

- [1] Ronfard R, Rossignac JR. Full-range approximation of triangulated polyhedra. Proceedings of Eurographics'96. 1996;15(3):C-67.
- [2] Welzl E. Smallest enclosing disks(balls and ellipsoids), volume 555 of Lectures and Notes in Computer Science. Berlin:Springer, 1991:359-70.
- [3] Halperin D, Sharir M. New bounds for lower envelopes in three dimensions with application to visibility in terrains. Proceedings of 9th ACM Symposium on Computational Geometry. 1993: pp. 11-18.
- [4] Klosowski J, Held M, Mitchell JSB, Sowizral H, Zikan K. Efficient collision detection using bounding volume hierarchies of k-DOPs, In SIGGRAPH'96 Visual Proceedings, 1996.
- [5] Gottschalk S, Lin MC, Manocha D. Obbtree: A hierarchical structure for rapid interference detection. Technical report, Department of Computer Science:University of North Carolina, 1996.
- [6] Preparata FP, Shamos MI. Computational Geometry: an introduction. Berlin: Springer, 1987.
- [7] Barequet G, Chazelle B, Guibas L, Mitchell JSB, Tal A. Boxtree: a hierarchical representation for surfaces in 3D, Computer Graphics Forum, Proceedings of Eurographics'96, 1996, C-387-C-484.