Classification of a point with respect to a polyhedron vertex

Robert Juan
Alvar Vinacua
Pere Brunet

Report LSI–90–19

1. Introduction

The Extended Octrees (EO), as introduced in [Ayala 85] and [Brunet 85], are a generalization of the classical Octrees [Meagher 80] based on the inclusion of the extended terminal nodes. In addition to the classical node types (White, Black and Grey), the EO model incorporates Face, Edge and Vertex nodes. These new terminal nodes contain a piece of the boundary of the modelled object.

The extended nodes are defined [Ayala 85], [Navazo 89] as follows,

- Face nodes are crossed by a single face of the object. They are coded by the node type and the oriented equation of the plain which supports the face.
- Edge nodes contain only two neighboring faces together with a part of their common edge. They are coded by the node type, the oriented equations of the plains which support the faces and the configuration of the node that is defined either, as the faces' semi-spaces intersection when the edge is convex or as the faces' semi-spaces union if the edge is concave.
- Vertex nodes contain only one vertex of the object and part of the edges and faces converging to it. They are coded by the node type and the configuration of the node that is defined as the set of configurations of the cyclically ordered edges plus the equations of the planes supporting the faces.

One of the major advantages of EO scheme is that the algorithms for boolean operations are simple [Ayala 85], [Navazo 86]. Given two EO, the desired boolean operation is carried out by traversing both trees simultaneously and performing the boolean operation on the nodes at the same tree depth level and in the same spatial location [Meagher 80], [Oliver 83], [Ayala 85], [Navazo 89].

It is well known [Tilove 80], [Tilove 81] that algorithms for boolean operations are strongly based on classifying points with respect to solids and there are several algorithms in the literature addressed to solve the point-in-polyhedron problem. Some of them, [Kalay 82], [Lane 84], [Horn 89], are devised to classify a point with respect to a polyhedron represented by means of a Boundary Representation scheme. In the case of polyhedral solids represented as EO one only needs to classify points with respect to extended terminal nodes [Ayala 85], [Brunet 85], [Navazo 89]. To classify a point with respect to a Face node or Edge node is straightforward. Dealing with extended Vertex nodes is a little more difficult, and the general approach would be too expensive.

Other algorithms operate on polyhedra represented via Constructive Solid Geometry (CSG) and use the divide-and-conquer paradigm [Tilove 80], [Voelcker 78]. As EO does not actually store CSG trees except for the edges, this kind of algorithms do not apply.

This work describes an algorithm to classify an arbitrary point in an extended Vertex node with respect to the polyhedral region defined by the piece of solid boundary inside the node. The algorithm takes advantage of the EO data structure in order to reduce the computations. Furthermore, the algorithm is simple and does not suffer from singularities.

2. The solution proposed

In this paper we propose to solve this problem using the kinetic framework of Guibas *et al.* [Guibas 83]. To do this, we will reduce the problem to an equivalent problem in 2D; this two-dimensional problem will be of a kind not directly addressable

by Guibas' method, and hence we will suitably extend it. Finally we will show that this solution can be computed directly in three-dimensional space, without ever needing to "go down" to 2D.

Dürst and Kunii have also studied this problem in [Dürst 89b]. There, they mention that their first approach was also to extend the kinetic framework of Guibas *et al.*, but the resulting algorithm was cumbersome because it has to use several tables, although they do not give any hint as to exactly how they do this. Note that our algorithm does no suffer from this need. In fact the solution we propose here has been implemented and thoroughly tested, as was reported in [Brunet 90].

Let's first set up some notation. Let x be the point we want to classify with respect to a solid $\mathcal{P}$, and let v be the vertex of $\mathcal{P}$ in the same octree node as x.

A first idea might be to consider a small ball centered at v, and to look at the intersection of the surface of that ball with $\mathcal{P}$, and the central projection from v of x onto this surface. The classification problem is thus reduced to a two dimensional problem, but extending the *exclussive* or algorithm to this case is non-trivial, because of the different topology of the sphere and the plane.

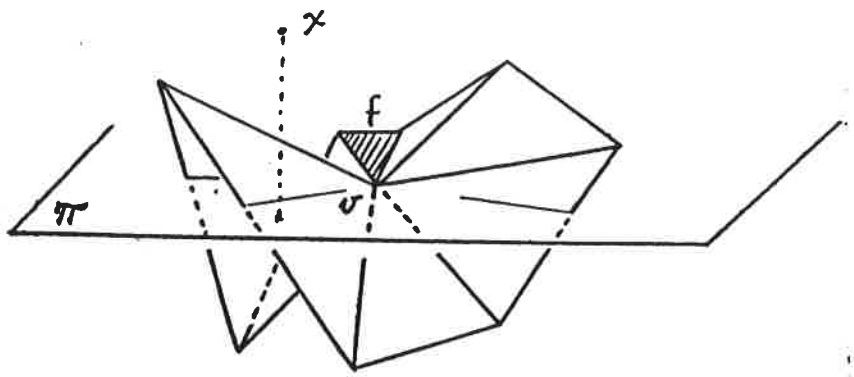

Figure 1: Plane π that supports a face f through a vertex v.

Instead, we will follow the steps below (see Figure 1 and Figure 2):

1– Find a plane π through v not containing x. Note that this can always be done unless $x \equiv v$. However not all the feasible ways of doing this are appropriate for the ensuing steps. This question is further discussed in the following section.

2– Consider a plane $\bar{\pi}$ parallel to π but closer to x than π. Let P be the (possibly multiply connected and unbounded) polygon resulting from the intersection of $\mathcal{P}$ with the new plane. Let $\bar{x}$ be the central projection of x onto $\bar{\pi}$ from v. Then classifying x with respect to $\mathcal{P}$ is equivalent to classifying $\bar{x}$ with respect to P. Note that in some cases this may be extremely simple, as it is possible that P be empty if π separated $\mathcal{P}$ and x. P may also be all of $\bar{\pi}$. Note also that how much we shift π is immaterial, as long as we do not move it so much that it sweeps past another vertex of $\mathcal{P}$. In fact, since $\mathcal{P}$ is locally star-shaped, all the resulting intersections would be homothetically equivalent, where the center of the homotecy is the projection of v onto $\bar{\pi}$ in the direction of the normal to $\bar{\pi}$.

3– In section 5 we will show how the kinetic framework can be adapted to this case, and how the resulting expression to be evaluated can be computed directly in

three-dimensional space, without needing to intersect $\mathcal{P}$ and $\bar{\pi}$, nor project x onto $\bar{\pi}$.

3. Finding the splitting plane

We call the plane π a splitting plane because the strategy outlined above amounts to discarding all of $\mathcal{P} \cap \pi_-$. The plane π used in the first step of section 2 can be rather arbitrary. It need only satisfy two conditions:

a. The vertex v is **on** π, and

b. The point x is **not on** π.

These are very easy to fulfill. For instance the plane of the points y such that $(y - v) \cdot (x - v) = 0$ is a possible choice (unless $v \equiv x$ in which case we are done anyway). In fact let us call this choice Option 1.

Another possibility is to loop over the planes of the faces incident on the vertex v, and choose the first one that satisfies condition b. above (since they all obviously satisfy condition a). Let this be Option 2.

Option 1 is clearly very inexpensive computation-wise. Option 2 is instead more costly (since it implies a traversal of the list of faces converging to v). Its strength —at first sight— may lie in that it has a chance of producing a separating plane in the case of a convex vertex. Thus, if the solids are expected to have a very large proportion of convex vertices, it may prove worthwhile to spend the extra processing at this stage. For general and very complicated bodies it may not. As all heuristic matters, this one will require in each case some experimentation to assess the best strategy, although not very great differences are to be expected, since the ensuing computation is not exceedingly intricate.

More important, however, is the fact that Option 2 allows a smoother extension of the kinetic framework to the present problem, and thus we adopt it: in what follows we will assume that a plane π satisfying both conditions a. and b. above has been determined along the lines of Option 2. Furthermore, we will assume that all of the solid $\mathcal{P}$ with respect to which we want to classify x lies on the same side of π as x itself. If this were not true, it would be enough to take as the new solid $\bar{\mathcal{P}}$ the regularized intersection of the solid $\mathcal{P}$ and the hemispace defined by π that contains x (denoted here π_+). Classifying x with respect to $\bar{\mathcal{P}}$ is equivalent to classifying it with respect to $\mathcal{P}$, since both solids do not differ in the hemispace with respect to π to which x belongs. Note that $\bar{\mathcal{P}}$ may be empty, but this ends the question of classifying x: it is necessarily *out*, since we have found a plane that separates it and the solid.

4. Boiling it down to 2D

Now let us return to our problem: we have a vertex node $\mathcal{Q}$ of an extended octtree, and a point in it, and wish to determine whether the point is *in*, *on* or *out*side the solid. The answer to this problem is not modified —obviously— if we change the solid outside the node. Let us then erase all of the solid outside the octree node, and substitute it with the cone emanating from v whose edges are the continuation of the edges in the node, and whose faces are the continuation of the faces in the node.

Formally, for each point y, let ℓ_y denote the semi infinite line

$$\ell_y = \Big\{ z \big| z = v + \lambda(y - v), \lambda \geq 0 \Big\}.$$

Furthermore, let $\bar{\mathcal{P}}_{\mathcal{Q}}$ denote the portion of the solid $\bar{\mathcal{P}}$ contained in the node we are considering. Then instead of solving the problem of classifying x with respect to $\bar{\mathcal{P}}$, we purport to solve the problem of classifying x with respect to $\hat{\mathcal{P}} = \bigcup_{y \in \bar{\mathcal{P}}_{\mathcal{Q}}} \ell_y$. These two problems are equivalent because clearly by construction we have $\bar{\mathcal{P}} \cap \mathcal{Q} = \hat{\mathcal{P}} \cap \mathcal{Q}$ and by hypothesis $x \in \mathcal{Q}$.

Let $\bar{\pi}$ be the plane parallel to π that contains x. Then $P = \hat{\mathcal{P}} \cap \bar{\pi}$ is a nonempty collection of disjoint polygons. Note that these polygons will be unbounded, since $\hat{\mathcal{P}}$ has edges on π. (see Figure 2).

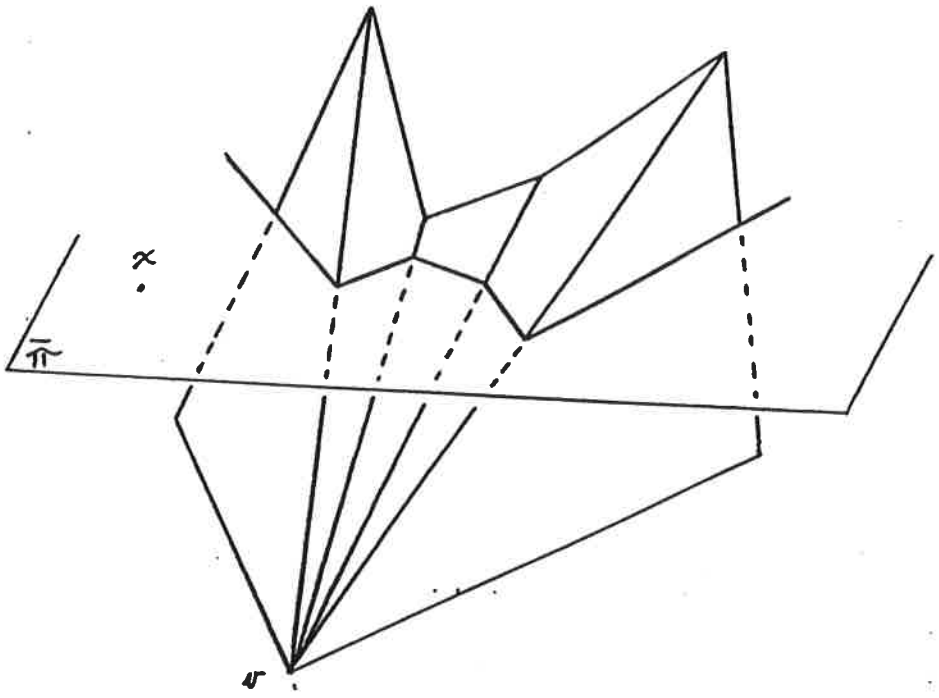

Figure 2: Splitting $\mathcal{P}$ near a vertex v.

Now by construction $x \in \bar{\pi}$ (Figure 2). Then

$$x \text{ in } \hat{\mathcal{P}} \iff (x \in X \text{ for some connected component } X \text{ of } P),$$

and a similar statement holds for a point on $\hat{\mathcal{P}}$.

Now the problem has been reduced to an equivalent one in two dimensions. The kinetic framework of Guibas *et al.* cannot yet be applied because the component polygons of P are unbounded. A simple extension can, however, be defined for this case. This we do in the next section.

5. Extending the kinetic framework to our problem

A different approach of constructing boolean formulas defining 2D polygonal regions using information at polygon vertices is given in [Guibas 83]. The approach is stated in the following way. Let $L_i(x)$ be a boolean predicate being true for all those points x inside the halfspace defined by the line supporting edge e_i of the polygon P. Let $S_i(x)$ be $L_i(x) \wedge L'_{i+1}(x)$ if the vertex defined by edges (e_i, e_{i+1}) is convex and

$L_i'(x) \wedge L_{i+1}(x)$ otherwise where the index $i+1$ is taken modulo the number of polygon vertices n and L' denotes the complement of L. Then

Theorem 1.- The point x is inside the simple polygon P if and only if the exclusive or $S_0(x) \oplus S_1(x) \oplus \cdots \oplus S_{n-1}(x)$ yields false.

Although a formula of this style is not desirable as a solid modelling representation, it is trivial to write down and provides an efficient test for wether a point x is inside a simple polygon [Guibas 83].

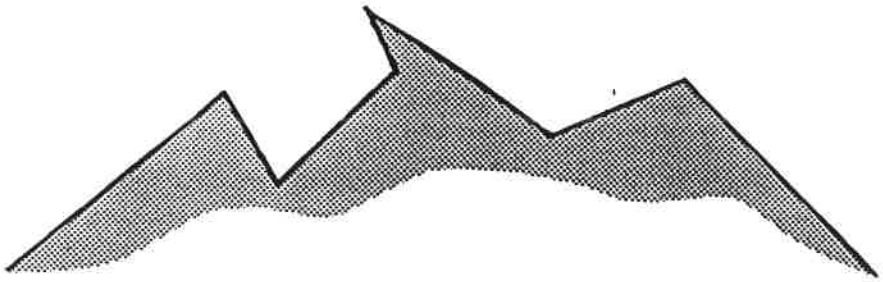

Figure 3: A simple bi-infinite polygonal chain.

As it has been said above, our polygonal regions are unbounded so they are not simple polygons and therefore, the theorem does not apply. Let us refer to the boundary of any unbounded simple polygonal region like that showed in Figure 3 as a simple bi-infinite polygonal chain [Dobkin 88]. Such a chain is terminated by two semi-infinite rays and in between contains an arbitrary finite number of sides. The only three possible types of unbounded polygonal regions produced when intersecting a octree vertex node with the plane π are those depicted in Figure 4 because unbounded polygonal regions like those showed in Figure 5 would mean that the two vertex faces which produce on the plain π the two semi-infinite edges do not converge to the original vertex in the extended octree node.

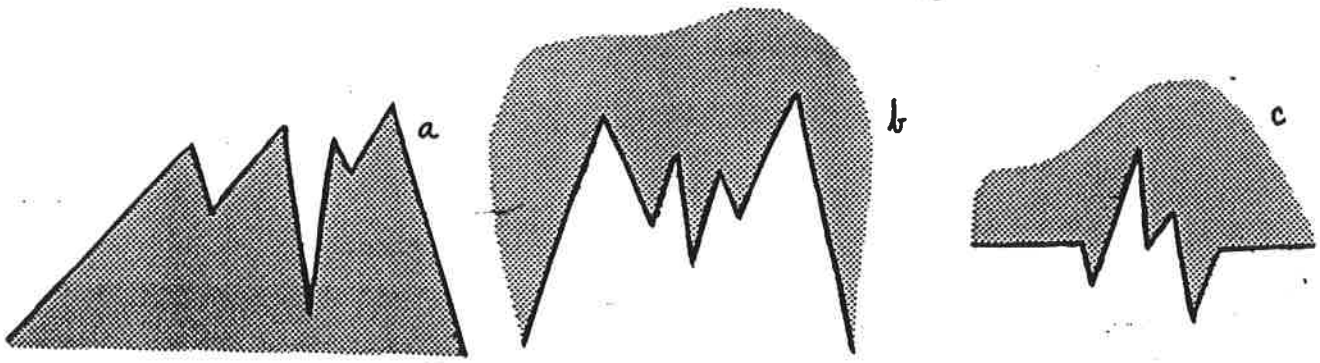

Figure 4: Possible bi-infinte polygonal chains. a) Convex in the large. b) Concave in the large. c) Parallel.

The extension to unbounded polygonal regions of the kinetic framework of Guibas *et al.* is easy to obtain. Let us consider first a convex *in the large* unbounded polygonal region as the one showed in Figure 6. Given an arbitrary point x it is always possible to find a ball centered in x which contains all the bi-infinite polygonal chain vertices. It is easy to see that the virtual vertex v_r on the right semi-infinite edge contributes to the winding number [Guibas 83] with $\Delta w_r = 1$ if point x is outside the halfspace which supports edge e_r and $\Delta w_r = 0$ otherwise.

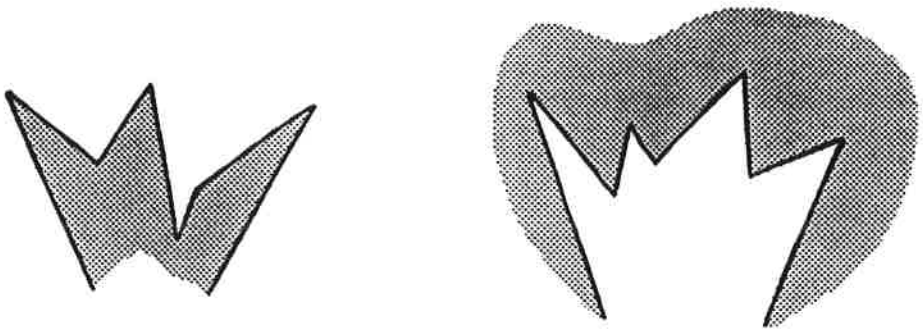

Figure 5: Impossible bi-infinite polygonal chains.

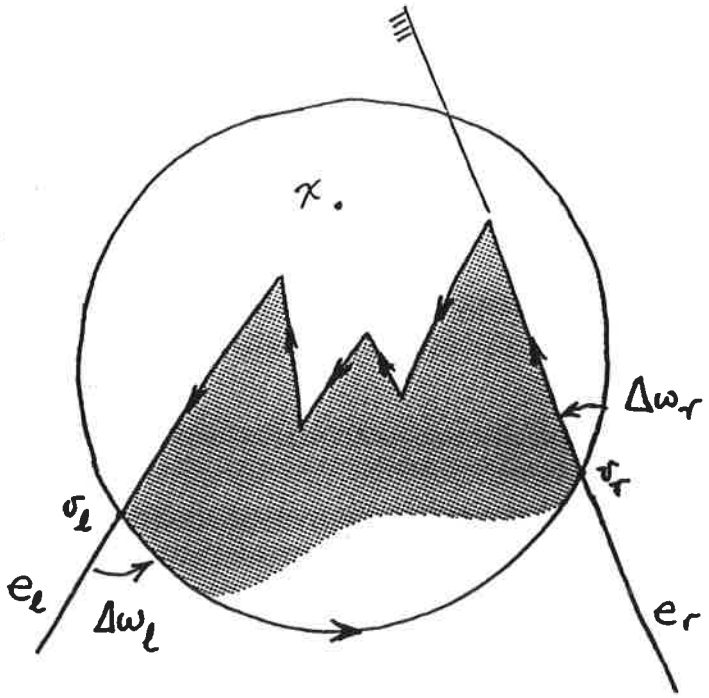

Figure 6: Convex *in the large* unbounded polygonal region and enclosing ball for an arbitrary point x.

On the other hand, there is no way of sweeping past point x in the turn at the virtual vertex v_l so $\Delta w_l = 0$.

Let us consider now the concave *in the large* polygonal region, shown in Figure 7. Reasoning in the same way as before leads us to state that $\Delta w_r = 1$ if point x is outside the halfspace supporting e_r, $\Delta w_r = 0$ otherwise and $\Delta w_l = 0$.

The rule obviously works when semi-infinite edges in the polygonal chain are parallel. So if $\{v_0, v_1, \ldots, v_{n-1}\}$ are the vertices of the unbounded polygonal region, we have shown the following

Theorem 2.- The point x is inside the simple unbounded polygonal region P if and only if the following exclusive or yields false, $S_0(x) \oplus S_1(x) \oplus \cdots \oplus S_{n-1}(x) \oplus L_r'(x)$.

Notice that changing the choosen orientation in the polygonal chain just inter-

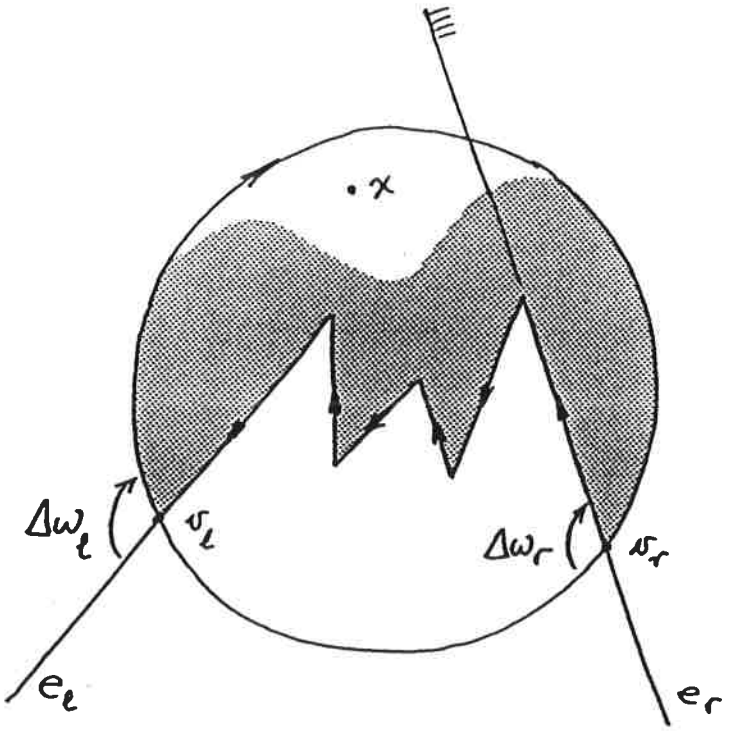

Figure 7: Concave *in the large* unbounded polygonal region and enclosing ball for an arbitrary point x.

changes both the Δw_r and Δw_l winding number contributions at the virtual vertices and the role played by L_r and L_l.

It should be noted here that the whole preceding construction is necessary to reduce the problem to the known plane case, but the resulting method can be applied in 3D directly without ever needing to compute the intersection of the polihedron with $\bar{\pi}$, nor carrying out any other computation on that plane. In fact, each of the terms in expression is either of the form $L_i \wedge L'_{i+1}$ or $L'_i \wedge L_{i+1}$, or L'_r. Note however that L_i is the intersection of the plain π_i supporting a face with $\bar{\pi}$, and by our construction necessarily

$$L_i(x) = \pi_i(x)$$

and therefore this expression can be completely evaluated in 3D replacing all L's by π's. This leads to the algorithm in the following section.

Some implementations of EO define also a *quasi-vertex* node to further reduce the complexity of the representation. These nodes contain several edges converging to a vertex, although the vertex itself lies outside (see [Dürst 89a] and [Brunet 90]). Since the $\bar{\mathcal{P}}$ constructed from this information will only differ from the $\bar{\mathcal{P}}$ associated to the vertex outside of the node, the same solution is valid in the case of point in quasi-vertex node classification.

6. The resulting algorithm

Here we present pseudo-code for the algorithm proposed.

```
function point_in_vertex (x : point, v : vertex) returns {on, in, out}
        { Returns the appropriate classification of x, either on, in or out of v}
        Search for the first plane π of a face converging to v such that x ∉ π
        if there is no such π
          then return on
          else
               if x is not on the positive side π+ of π
                 then change the sign of the coefficients of π
               endif
               Form a list of all the edges of v that lie in the half-space π+
                    (but are not contained in π)
               if the list is empty
                 then return the classification of x with respect to the original plane π
                              (that is before changing signs)
                 else
                      Split the list in lists of consecutive edges
                      while there are lists to process
                            {let us say that the current list is {e1, ..., ek} }
                            {Furthermore, let πi be the plane spanned by ei and ei+1 }
                            {Also let π0 be the plane of the face containing e1}
                            {and e0(∈ π−)}
                            if x on Le1 then non := 1
                                        else non := 0
                            endif
                            if x not in Le1 then nin := 1
                                            else nin := 0
                            endif
                            for i := 1 to k do
                                if ei is convex then cl := classif(x, πi ∩ π'i+1)
                                                 else cl := classif(x, π'i ∩ πi+1)
                                endif
                                case cl of
                                     on : non := non + 1
                                     in : nin := nin + 1
                                endcase
                            endfor
                            if non mod 2 = 1 then return on endif
                            if nin mod 2 = 0 then return in endif
                      endwhile
                      { It is not on or in any component, so...}
                      return out
               endif
        endif
endfunction
```

At most, the algorithm processes the list of the node faces three times: one in the search for the splitting plane, one for forming the list of faces in half-space π_+ and one more to split the list in lists of consecutive edges. The first and second list traversal can be subsumed in just one. This results in a complexity of $O(n)$, where n is the number of faces connected to the vertex.

7. Conclusions

We have presented a simple and robust algorithm for point_in_vertex classification which represents an extension of the kinetic framework of Guibas *et al.* to this 3D problem. The algorithm has been succesfully implemented and thoroughly tested. It is specially well suited for application in EO.

References

[Ayala 85] D. Ayala, P. Brunet, R. Juan and I. Navazo, "Object representation by means of non minimal division quadtrees and octrees", ACM Transactions on Computer Graphics 4(1), pp. 41-59 (1985)

[Brunet 85] P. Brunet and I. Navazo, "Geometric modeling using exact octree representation for polyhedral objects", Proceedings Eurographics'85, pp. 159-169, Nice, September 1985

[Brunet 90] P. Brunet and I. Navazo, "Solid representation and operation using extended octrees", ACM Transactions on Graphics 9(4), 1990.

[Dobkin 88] D. Dobkin, L. Guibas, J.E. Hershberger and J. Soneyink, "An efficient algorithm for finding the CSG representation of a simple polygon", Computer Graphics 4(22), pp. 31-40, Proc. SIGGRAPH 1988.

[Dürst 89a] M. Dürst and T. Kunii, "Integrated polytrees: A generalized model for the integration of spatial decomposition and boundary representation", Theory and practice of Geometric Modeling, W. Strasser and H.-P Seidel, Eds., Springer-Verlag, Heidelberg, 1989, pp. 329-348.

[Dürst 89b] M. Dürst and T. Kunii, "Vertex classification using the convex hull on a sphere", COMP 89-53, Univerity of Tokyo, 1989.

[Horn 89] W.P. Horn and D.L. Taylor, "A theorem to determine the spatial containement of a point in a planar polyhedron", Computer Vision, Graphics and Image Processing 26, pp. 118-125 (1989)

[Kalay 82] Y.E. Kalay, "Determining the spatial containement of a point in general polyhedra", Computer Graphics and Image Processing, 19, 1982, pp. 303-334

[Lane 84] J. Lane, B. Magedson and M. Rarick, "An efficient point in polyhedron algorithm", Computer Vision, Graphics and Image Processing 26, pp. 118-125 (1984)

[Meagher 80] D. Meagher, "Octree encoding: A new technique for the representation, manipulation and display of arbitrary three dimensional objects by computer", Tech. Report IPL-TR-80-111, Rensselaer Polytechnic Inst.

[Navazo 86] I. Navazo, "Geometric modelling of octree encoded polyhedral solids", PhD dissertation. January 1986. Universitat Politecnica de Catalunya.

[Navazo 89] I. Navazo, "Extended octrees representation of general solids with plane faces: Model structure and algorithms", Computer and graphics 13(1), pp. 5-16 (1989)

[Oliver 83] M. Oliver and N. Wiseman, "Operation on quadtree encoded images", The Computer Journal 26(1), pp. 83-91 (1983)

[Tilove 80] R.B. Tilove, "Set membership classification: A unified approach to geometric intersection problems", IEEE Transactions on Computers c29(10), pp. 874-883 (1980)

[Tilove 81] R.B. Tilove, "Line/polygon classification: A study of the complexity of geometric computation", IEEE Computer Graphics and Applications, April 1981, pp. 75-83

[Voelcker 78] H.B. Voelcker *et al.*, "The PADL-1.0/2 system for defining and displaying solid objects", Computer Graphics, vol. 12, pp. 257-263, August 1978.