Comput. & Graphics Vol. 14, Nos. 3/4, pp. 445–447, 1990
Printed in Great Britain.

0097-8493/90 $3.00 + .00

Technical Section

A QUICK POINT-IN-POLYHEDRON TEST

JOHANN LINHART
Universität Salzburg, Institut für Mathematik, Hellbrunnerstr. 34, A-5020 Salzburg, Austria

Abstract—To decide whether a point is contained in a polyhedron, one may use the parity of the number of intersections of a halfline with the boundary of the polyhedron. Difficulties arise if the halfline meets an edge or a vertex. To resolve such singularities, we propose to associate to each point of intersection a certain positive or negative weight and then to consider the sum of these weights.

1. INTRODUCTION

To test the containment of a point P in a planar-faced polyhedron R, several methods have been studied already (*e.g.*, [2–4]). A main problem seems to be the treatment of singular cases.

If we ignore the singular cases, the simplest and quickest way probably consists in counting the number of points of intersection of a halfline (originating at the test point P) with the boundary of the polyhedron: P is contained in R if this number is odd. Kalay[3] discussed two methods to resolve the singularities, the "projection method" and the "intersection method." In the present paper, a modification of the projection method is presented which seems to be easier and faster. It is applicable to all sorts of planar-faced polyhedral solids, including those with nonconvex faces, holes, or several connected components.

Perhaps the method is best understood if the two-dimensional analogon (which is possibly also new) is described first. Instead of simply counting the intersections of the halfline with the sides of the polygon, we count -1 if the halfline enters the polygon and $+1$ if it leaves it. In case the halfline meets a vertex, we count $-\frac{1}{2}$ or $+\frac{1}{2}$ for each of the two adjacent sides. If a side is collinear with the halfline, it is ignored (unless the test point lies on this side, in which case we are done). The sum will be zero if and only if the point lies outside the polygon.

As to the case of three dimensions, imagine the halfline to be a very thin cylinder. Then to each face F which is met by the halfline, we assign a number s which indicates the extent to which the halfline enters ($s < 0$) or leaves ($s > 0$) the polyhedron at this face (*i.e.*, the fraction of the cross-sectional area of the thin cylinder which gets into or out of the polyhedron). The sum of these numbers will be 1 if P is inside R and 0 if it is outside. The algorithm automatically recognizes the case that P lies on such a face.

The exact definition of s is as follows: Let u be the unit vector in the direction of the halfline H, and F a face of R which intersects H in a point X. Let v be the outer normal vector of F. We distinguish three cases:

1. X lies in the relative interior of F:

$$s := \operatorname{sign}(u \cdot v).$$

2. X lies in the relative interior of an edge of F:

$$s := \tfrac{1}{2} \operatorname{sign}(u \cdot v).$$

3. X coincides with a vertex of F: In this case, let α be the (inner) angle of the normal projection of F in the direction of the halfline at this vertex ($0 < \alpha < 2\pi$). Then

$$s := \frac{\alpha}{2\pi} \operatorname{sign}(u \cdot v).$$

By this we achieve that at each point where the halfline really penetrates the boundary, the s sum up to 1 or -1 (compare Fig. 1), while a value of zero indicates a tangency. If the halfline H lies in the plane of a face, the sum of the numbers s corresponding to all neighboring faces met by H will (in general) again be $1, -1$, or 0 (Fig. 2), unless the test point lies on the face. Fig. 3 illustrates the case where H contains an edge.

A formal proof of the correctness of the algorithm is not given here, since it seems to be fairly clear from intuition and we are mainly interested in practical applications.

Kalay[3] claimed that penetration takes place only if the "senses of the vector components of the normals on the test line of all the faces which are directly adjacent to the coincidental element are the same." This is not true, as should be clear from Fig. 1.

2. THE ALGORITHM

Since we may take any direction for the halfline, the best will be to choose the direction of a coordinate axis, say the positive x-axis. If several or many points have to be tested against the same polyhedron, computing time may be reduced considerably by calculating in advance the minimal and maximal coordinates for each face. If we assume this kind of preprocessing, the algorithm may be described as follows:

1. Initialize the counter s, which has to be a real variable: $s := 0$.
2. For each face F, do the following: Let P' and F' be the orthogonal projection of P and F onto the yz-plane.
   a. If P' lies outside the circumscribed box of F' (determined by the min-max coordinates), go to the next face.
      If the x-coordinate of P is greater than the maximal x-coordinate of F, go to the next face.

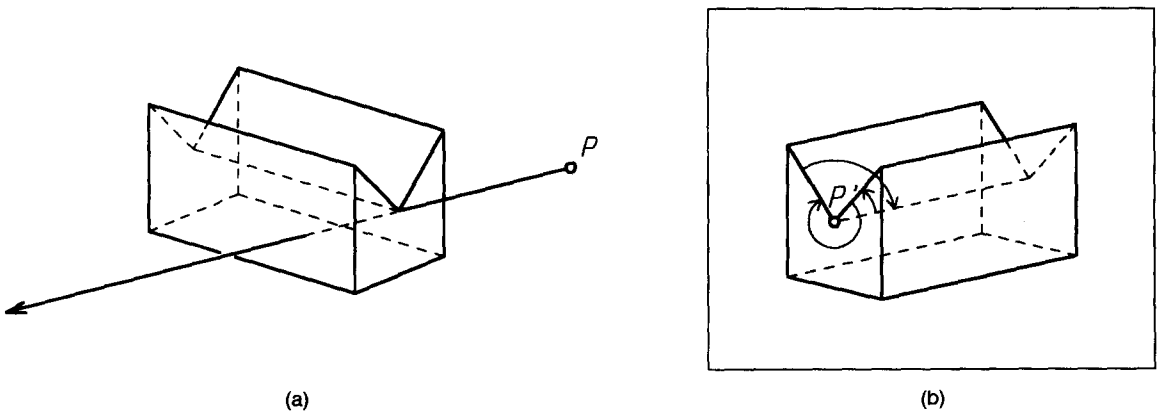

Fig. 1. (a) Halfline passing through a vertex of the polyhedron. (b) Image of the projection parallel to the halfline. The arrows indicate the signed angles which are used to define the "extent of penetration" s.

b. Let H be the halfline from P in the direction of the positive x-axis. Calculate the intersection of H with the plane E of F.
   If H and E are disjoint, go to the next face.
   If H lies in E, determine the containment of P in F by a 2D containment test, using a suitable projection onto a coordinate plane. If P is contained in F then stop, else go to the next face.
   If H intersects E, continue.
c. Apply a 2D containment test to P' with respect to F' and increment s according to the definition in the introduction.
   If P lies in the plane E of F and P' is inside F' or on the boundary of F', stop here.

3. If $s = 1$ (up to some generous tolerance, like 0.2), P is inside F. If $s = 0$, P is outside F. If s has a value (very) different from 0 and 1, an error has occured (*e.g.*, the polyhedron is not well defined).

Of course it would be possible to improve this algorithm (especially step 2a) by using a suitable data structure like those discussed in [5] or [1]. By that means one could get directly a list of relevant faces, so that the average time-complexity could probably be reduced from $O(n)$ to $O(\log n)$ (where n denotes the number of faces), while the preprocessing would become substantially more complicated. From the practical point of view, this is only important for polyhedra with a very large number of faces (say more than several thousand), and if a great number of points is tested against the same polyhedron.

3. REALIZATION

The algorithm described above has been implemented and tested during the development of the CAD-System MEMOPLOT at the University of Salzburg. The main purpose was to obtain a sufficiently fast method to perform Boolean operations on polyhedral sets.

We used an IBM-AT-compatible microcomputer equipped with an additional board (DSI-020) with a 32-bit processor (Motorola MC68020) and a floating-point accelerator (MC68881) on it. The program was written in FORTRAN with no attention to the translation by the compiler and no sophisticated data structure. To test a point against a polyhedron with 1,000 faces, the CPU time was typically about 0.025 seconds. In particular, the time was not significantly longer in the case of singularities.

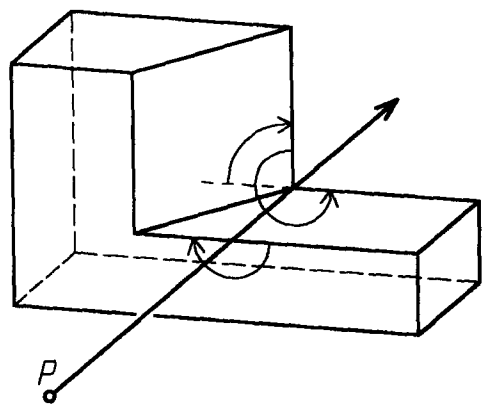

Fig. 2. Halfline lying in the plane of a face.

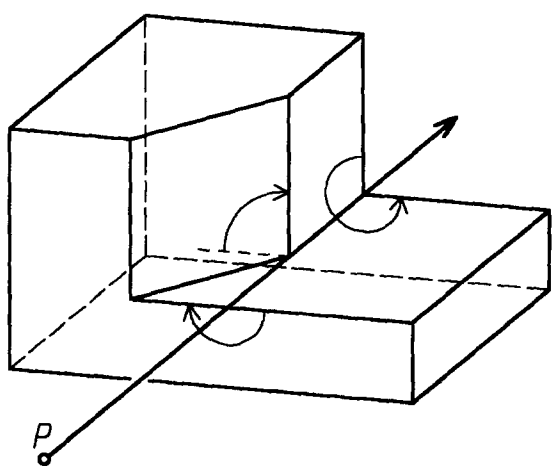

Fig. 3. Halfline containing an edge.

4. COMPARISON WITH OTHER METHODS

From the known algorithms, the projection-elimination algorithm of Kalay[3] comes closest to the one discussed here. To resolve the case where the halfline meets an edge or a vertex, he proposes to merge all faces incident to that edge or vertex into a single, possibly nonplanar face. One problem with this method is the fact (discussed by Kalay) that it may be necessary to iterate the process of merging several times.

The singularity problems seem to have motivated the papers [4] and [2]. In [4], essentially the solid angles spanned by the faces at the test point are summed up. The point lies in the interior if and only if this sum equals 4π. This method is very elegant from the mathematical point of view, but the computations needed are rather time-consuming: more than three seconds for one point against a polyhedron with 232 faces (on a VAX 11/780).

In the recent paper by Horn and Taylor[2], the possible test points are classified according to whether they are closest to a vertex, an edge, or a face of the polyhedron. This method seems to be somewhat more complicated than the one presented here, and no concrete information on CPU time or practical experiences has been given.

REFERENCES

1. H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin (1987).
2. W. P. Horn and D. L. Taylor, A theorem to determine the spatial containment of a point in a planar polyhedron. *Comput. Vision Graphics Image Process.* **45**, 106–116 (1989).
3. Y. E. Kalay, Determining the spatial containment of a point in general polyhedra. *Comput. Vision Graphics Image Process.* **19**, 303–334 (1982).
4. J. Lane, B. Magedson, and M. Rarick, An efficient point in polyhedron algorithm. *Comput. Vision Graphics Image Process.* **26**, 118–125 (1984).
5. F. P. Preparata and M. I. Shamos, *Computational Geometry—an Introduction*, Springer-Verlag, New York (1985).