JOURNAL OF COMPUTATIONAL PHYSICS 84, 134–144 (1989)

Does a Point Lie Inside a Polygon?

M. S. MILGRAM

Reactor Physics Branch, Chalk River Nuclear Laboratories, Chalk River, Ontario, Canada K0J 1J0

Received June 13, 1988; revised November 7, 1988

A new algorithm is presented for determining whether or not a point lies inside a polygon. The method is particularly useful when the coordinates of point and polygon are given with respect to a multi-dimensional coordinate basis. The polygon must be convex and connected, but may be open. As a corollary, a method to determine the convexity of a polygon is revealed.

1. INTRODUCTION

An interesting, and superficially simple problem in computational geometry is that of determining whether or not a query point P lies in the interior of a polygon if it lies in the plane of the polygon. This is a significant question whose answer is often required when tracking particles in a Monte Carlo program; usually the question is asked many times and an efficient algorithm is crucial. Littlefield [1] has recently rediscovered Shimrat's algorithm [5], noting that a well-known textbook [3] claims that a "simple, quick solution for arbitrary non-convex polygons" does not exist.

In separate work, Wooff [6], Preparata and Shamos [4], and Mehlhorn [2], as well as Yamaguchi [7] give different algorithms to solve the same problem. The first of these algorithms is valid for arbitrary non-convex polygons in the plane, the second is limited to convex and "star-shaped" polygons but can be generalized, and the third is designed for problems in computer graphics; most suffer from deficiencies, described in Section 3 in more detail.

A practical algorithm to answer this question when the polygon lies in a plane skewed in space (i.e., the vertices are specified by three space coordinates with respect to some global coordinate system) is not immediately evident from most of these oeuvres. Additionally all but one of them fails when the polygon in question is open—that is two sides extend to infinity. (Such polygons are easily generated in, for example, hidden surface computations where they may represent the outlines of the shadow of some polygon R projected onto a plane Q from a point source S, if a plane parallel to Q through S happens to cut R—see Fig. 1.) The intent of this paper is to review the known methods and to present a new, efficient algorithm to answer this question, valid for all convex polygons, open or closed, in

0021-9991/89 $3.00

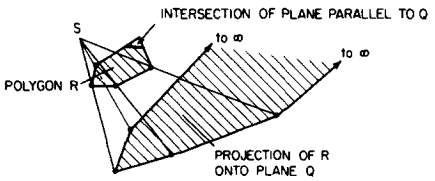

FIG. 1. How to generate an open polygon.

n-dimensional space ($n \geqslant 2$). However, they *must* be convex and topologically connected (contain no holes). The particular application is to Monte Carlo computer programs where the (N) vertices of geometrical objects (polygons) are computed once and for all, but the query point P is generated many times and the question must be answered quickly for each point, for a relatively small number of vertices at a time. As a corollary, a method has been found to test for the convexity of a polygon in n-space, and the method easily generalizes to spaces of more than three dimensions.

2. NOTATION

Although conventional methods of computational geometry use homogeneous (Desarguesian) coordinates, we shall confine ourselves to traditional methods of vector algebra. A polygon with N-sides is determined by N vertices $\mathbf{V}_i$ ($i = 1, ..., 1, N$) using three coordinates $\mathbf{V}_i = (V_x, V_y, V_z)_i$ with respect to a global reference system (Fig. 2a). An open polygon may be conveniently represented by the same N vertices, but only $N-1$ sides (Fig. 2b). Thus a list of:

(i) $3N$ vertex coordinates,

(ii) the number of items in the list (N)

(iii) the number of sides (N or $N-1$) or equivalently a logical flag,

contains sufficient information to define a polygon and its open/closedness. If two bounding edges extend to infinity, we use the convention that $\mathbf{V}_1$ and $\mathbf{V}_N$ are any

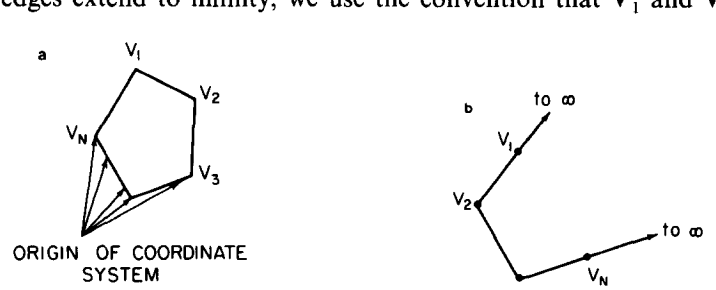

FIG. 2. (a) A polygon skewed in space, viewed from "above." (b) An open polygon skewed in space, viewed from "below."

two points on each of the bounding edges, provided they lie between the vertices $\mathbf{V}_2$ and $\mathbf{V}_{N-1}$, respectively, and the extension to the point at infinity. Furthermore, we use the convention that $\mathbf{V}_i$ $(i = 1, ..., N)$ are ordered so that they appear in clockwise order if the polygon is viewed in the direction opposite to the outward facing normal (front ways). It is also assumed that the point $\mathbf{P} = (P_x, P_y, P_z)$ is known to lie in the plane containing $\mathbf{V}_i$ $(i = 1, ..., N)$. The extension to more than three dimensions is obvious.

3

(a) *Shimrat's Algorithm*

Temporarily, let us assume we are working with two component vectors—i.e., all coordinates are given with respect to a system with $x-y$ axes lying in the plane of the polygon. In References [1, 4, 5] the suggestion is made that a horizontal line be drawn from the point **P** to the point at infinity. If:

(A) the y-coordinate of **P** is greater than or equal to the minimum value and less than the maximum value of the y-cordinates of two contiguous vertices, then

(B) the x-coordinate of the point of intersection is found.

If this coordinate is less than the x-coordinate of point **P**, it is counted, otherwise not; the test is repeated for all pairs of vertices. An odd/even number of counts means that P is within/without the polygon. The algorithm works for non-convex and disconnected polygons (Fig. 3a).

Suppose the polygon is open to infinity, however (Fig. 3b). A uniform convention such as that described in Section 2 cannot be relied upon, since the vertex V_1 and/or V_N may or may not satisfy the condition (A). This may be overcome by using two slopes plus vertices V_2 and V_{N-1} to define the bounding edges of an open polygon, but these two edges must then be treated differently from the others, introducing a further degree of complexity. More importantly, when points and vertices are specified in full 3-dimensional generality, there are only two clear ways to proceed, each of which has problems or complications.

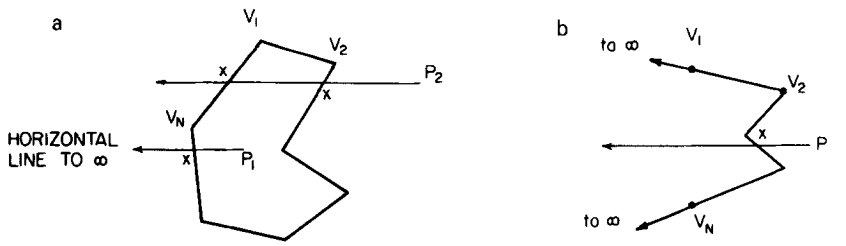

FIG. 3. (a) Shimrat's algorithm for points P_1 and P_2 uses a horizontal line in the plane extending to ∞. "x" indicates the intersection point(s) that will be calculated. (b) The horizontal line algorithm applied to an open polygon does not work.

The most obvious method requires that the coordinates of every point **P** be rotated into a coordinate system with z-axis perpendicular to the plane of the polygon. All the polygon vertices must also be given with respect to this local coordinate frame although these may be precomputed. This transformation can be computationally expensive when done once for each of a number of polygons for each point P. However, if the transformation is performed, the method ought to work.

The second possibility is to attempt to generalize the algorithm to 3-space. In that case, the "horizontal line to infinity" must be replaced by an unbounded horizontal plane passing through **P**. The intersection of this plane with an edge defined by the vertices $\mathbf{V}_i$ and $\mathbf{V}_{i+1}$, lies between the vertices if the parameter

$$\alpha = \frac{(\mathbf{P}-\mathbf{V}_i)_z}{(\mathbf{V}_{i+1}-\mathbf{V}_i)_z}$$

satisfies $0 \leqslant \alpha \leqslant 1$. The intersection point

$$\mathbf{X} = \alpha \mathbf{V}_{i+1} + (1-\alpha)\mathbf{V}_i.$$

must then be found and tested to determine if it lies between P and the point at ∞ in the horizontal plane, unless α is indeterminate, in which case the plane of the polygon is already horizontal and the 2-dimensional algorithm applies.

(b) *Wooff's Algorithm*

Wooff's algorithm [6] "uses the property that the sum of all angles formed between lines connecting a point **P** and the ith and $(i+1)$th vertices of a given polygon is zero if **P** lies outside the polygon and $\pm 2\pi$ if **P** lies inside the polygon." The implementation in two dimensions may be done by simply noting (Fig. 4) that the angle Θ_i between V_i and V_{i+1} is given by

$$\Theta_i = \tan^{-1}\frac{(V_y-P_y)_{i+1}}{(V_x-P_x)_{i+1}} + \tan^{-1}\frac{(V_y-P_y)_i}{(V_x-P_x)_i}.$$

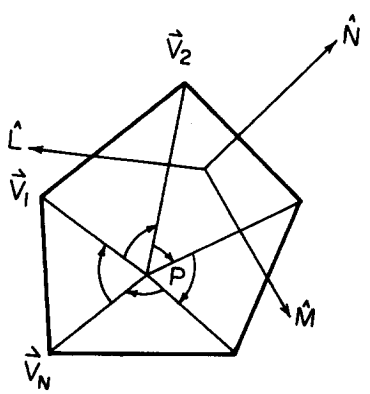

FIG. 4. Wooff's algorithhm in 3-space sums the angles joining **P** and $\mathbf{V}_i$. Note the "quadrant ambiguity" (branch cut) when crossing the negative M axis.

The first evaluation of the arc tan function may be kept since it becomes the second value of the arc tan when $i \to i+1$. Thus for a polygon with N vertices, N values of Θ_i must be added and N evaluations of the arc tangent are required. In addition to the generally accepted truism that the arc tangent is an expensive computation, Wooff alludes to the accepted reality that "a number of problems" will arise with ATAN2 due to "quadrant ambiguities." In my experience, these ambiguities are usually non-trivial to detect and computationally expensive to resolve. However, the algorithm can be generalized to 3-space by the simple expedient of introducing a standard local normalized vector coordinate system $\hat{\mathbf{L}}$, $\hat{\mathbf{M}}$, $\hat{\mathbf{N}}$, for each polygon where $\hat{\mathbf{M}}$ and $\hat{\mathbf{N}}$ lie in the plane of the polygon. With this convention, we have

$$\Theta_i = \tan^{-1} \frac{(\mathbf{V}_i - \mathbf{P}) \cdot \hat{\mathbf{N}}}{(\mathbf{V}_i - \mathbf{P}) \cdot \hat{\mathbf{M}}} + \tan^{-1} \frac{(\mathbf{V}_{i+1} - \mathbf{P}) \cdot \hat{\mathbf{N}}}{(\mathbf{V}_{i+1} - \mathbf{P}) \cdot \hat{\mathbf{M}}}$$

with the same comments holding as before. The resolution of quadrant ambiguities is considerably more difficult, and the case of open polygons must be specially treated.

(c) *Convex Inclusion*

The method of convex inclusion [2, 4] can be used to solve this problem in the case of convex and "star-shaped" polygons. The algorithm is two part (see Fig. 5). From a predetermined internal point Q (say the centroid), draw rays to each vertex, then:

(1) identify the vertices $\mathbf{V}_i$, $\mathbf{V}_{i+1}$ defining the wedge containing point $\mathbf{P}$;

(2) test for the handedness of the angle $\mathbf{V}_i\mathbf{V}_{i+1}\mathbf{P}$ to determine if $\mathbf{P}$ lies within/without the polygon.

It is immediately evident that this algorithm is not applicable to the case of an open polygon, since a suitable point $\mathbf{Q}$ does not exist (dashed lines of Fig. 5).

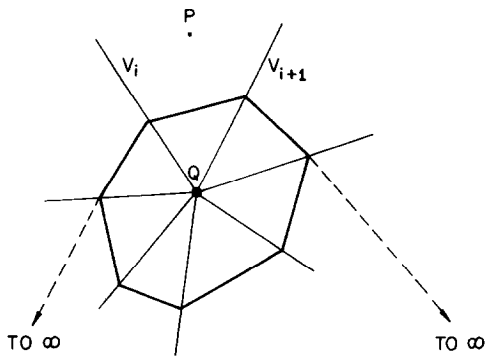

FIG. 5. Identify the wedge bounded by V_i, V_{i+1} containing P and test for the chirality of the angle $V_iV_{i+1}P$. This method (convex inclusion) fails if the polygon is open, since no point Q can be found.

The implementation in three dimensions, for closed, but convex polygons, is computationally expensive. For step 1, define the determinant

$$d_i = \begin{vmatrix} \mathbf{P} \\ \mathbf{Q} \\ \mathbf{V}_i \end{vmatrix}.$$

Then point $\mathbf{P}$ lies inside the wedge bounded by $\mathbf{V}_i$ and $\mathbf{V}_{i+1}$ if $d_i < 0$ and $d_{i+1} > 0$ and $\mathbf{V}_i$ are ordered clockwise. The method retains its validity if $\mathbf{P}$ lies on one of the rays ($d_i = 0$). Each such test requires nine multiplications and five additions.

Once the proper wedge has been identified, the variable η is evaluated:

$$\eta = \det \begin{vmatrix} \mathbf{V}_i \\ \mathbf{V}_{i+1} \\ \mathbf{P} \end{vmatrix}.$$

If $\eta < 0$ then angle $\mathbf{V}_i\mathbf{V}_{i+1}\mathbf{P}$ is clockwise and $\mathbf{P}$ is internal, with obvious extensions if it lies on the boundary or is external. Note that d_i and η change sign if $\mathbf{V}_i$ and $\mathbf{V}_{i+1}$ are interchanged, so this test requires a further element of preconditioning: determine the orientation of the normal rather than letting the ordering carry this information (see Section 2).

(d) *Triangulation*

Yamaguchi's triangulation method [7] is based on a simple test of the location of a point $\mathbf{P}(x, y, z, w)$, with respect to the half-plane containing a triangle, and is natural when a homogeneous coordinate system is being used. Given three points $\mathbf{R}_0$, $\mathbf{R}_1$, and $\mathbf{R}_2$ defining the triangle "i" with homogeneous coordinates $\mathbf{R}_j = (x_j, y_j, z_j, w_j)$ define:

$$\tilde{x}_i = \det \begin{vmatrix} y_0, z_0, w_0 \\ y_1, z_1, w_1 \\ y_2, z_2, w_2 \end{vmatrix} \qquad \tilde{y}_i = \det \begin{vmatrix} z_0, x_0, w_0 \\ z_1, x_1, w_1 \\ z_2, x_2, w_2 \end{vmatrix}$$

$$\tilde{z}_i = \det \begin{vmatrix} x_0, y_0, w_0 \\ x_1, y_1, w_1 \\ x_2, y_2, w_2 \end{vmatrix} \qquad \tilde{D}_i = \det \begin{vmatrix} x_0, y_0, z_0 \\ x_1, y_1, z_1 \\ x_2, y_2, z_2 \end{vmatrix}.$$

Form the test variable $\sigma_i = \mathbf{P} \cdot \mathbf{S}_i$, where $\mathbf{S}_i = (\tilde{x}_i, \tilde{y}_i, \tilde{z}_i, -D_i)$ is a 4-dimensional vector. The sign of σ_i indicates whether $\mathbf{P}$ lies above or below the plane containing the vertices $\mathbf{R}_i$ (Fig. (6a)).

For a convex polygon with vertices $\mathbf{V}_i$, the above forms a basis for a test for point inclusion. Construct a "wall" along the i^{th} edge of the convex polygon by choosing $\mathbf{R}_0 = \mathbf{V}_{i+1}$, $\mathbf{R}_1 = \mathbf{V}_i$, and $\mathbf{R}_2$ to be the point at infinity in the direction of

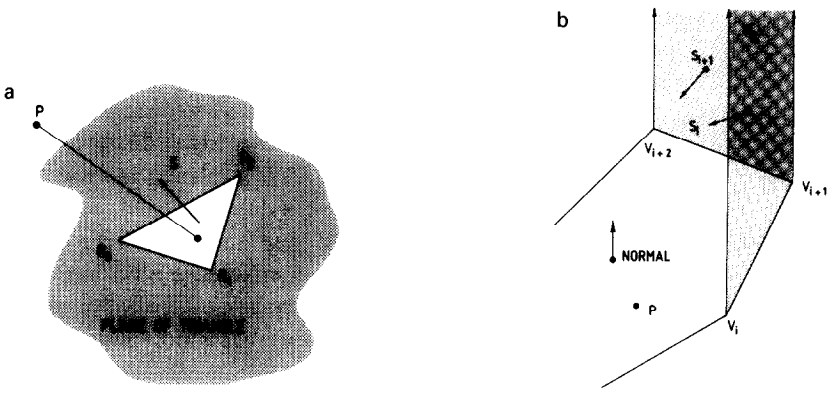

FIG. 6. (a) Yamaguchi's test determines whether P lies above or below the plane containing a triangle in homogeneous coordinates. (b) P lies inside a convex polygon if it lies on the same side of all normal planes each of which includes one bounding edge.

the normal to the polygon (set $w_2 = 0$) then construct $\mathbf{S}_i$ corresponding to each side (Fig. (6b)). This may be precomputed. According to Yamaguchi, $\mathbf{P}$ lies inside the polygon if $\sigma_i < 0$ for all i when $\mathbf{V}_i$ are given in counter clockwise order when seen from the front. A more general test would specify that the sign of σ_i does not change if P lies inside the polygon; this removes the dependence on the ordering of $\mathbf{V}_i$. The determination of σ_i requires four multiplications and three additions, is valid for open polygons by omitting the missing side, and can possibly be extended to more dimensions. The method is extendible to a general polygon by subdividing that polygon into triangles employing an algorithm given in Ref. [7].

4. A New Algorithm

The algorithm to be presented here makes use of vector invariant quantities throughout. Thus it is valid in either two or three (or more) dimensions without reference to any local coordinate system. It has the added advantage that it uses a geometrical quantity that is a function only of the geometry of the polygon being tested; this quantity—the edge vector—may be precomputed prior to any test. From now on we require that the polygon be convex.

Consider Fig. 7a. If the point P lies inside the polygon, it most lie to the *right* of all the edges of that polygon, when the polygon is traversed in a clockwise direction along the edges. The vector $(\mathbf{P}-\mathbf{A})\times(\mathbf{A}-\mathbf{B})$ will point out of the plane of the polygon, if $\mathbf{P}$ lies to the right of the line $\mathbf{A}-\mathbf{B}$. So will the vector $(\mathbf{A}-\mathbf{B})\times(\mathbf{B}-\mathbf{C})$, provided the angle at $\mathbf{B}$ is less than π. Form the test variable

$$\omega = (\mathbf{P}-\mathbf{A})\times(\mathbf{A}-\mathbf{B})\cdot(\mathbf{A}-\mathbf{B})\times(\mathbf{B}-\mathbf{C})$$
$$= (\mathbf{P}-\mathbf{A})\cdot\mathbf{E}$$

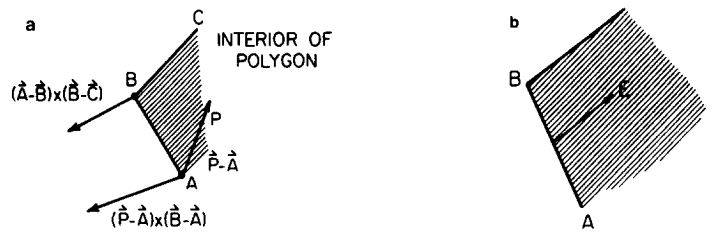

FIG. 7. (a) The vectors forming ω when the line $\mathbf{A}-\mathbf{B}$ forms one of the edges of the polygon. (b) The "edge vector" $\mathbf{E}$ for a bounding edge of a polygon.

using elementary vector identities, where the "edge vector" $\mathbf{E}$, defined by

$$\mathbf{E} = (\mathbf{A}-\mathbf{B})\alpha_1 - (\mathbf{B}-\mathbf{C})\alpha_2$$

with

$$\alpha_1 = (\mathbf{A}-\mathbf{B})\cdot(\mathbf{B}-\mathbf{C})$$
$$\alpha_2 = (\mathbf{A}-\mathbf{B})\cdot(\mathbf{A}-\mathbf{B})$$

is perpendicular to the edge $\mathbf{A}-\mathbf{B}$ and points "inward," since $\mathbf{E}\cdot(\mathbf{A}-\mathbf{B})=0$ and $\mathbf{E}\cdot(\mathbf{B}-\mathbf{C})<0$. See Fig. (7b). Notice that ω is invariant under reversal of the ordering of points, but changes sign if the angle at vertex B is greater than π, since $\mathbf{E}$ will point "outward" in this eventuality.

In what follows let us generalize to an N-gon by replacing vertices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ with $\mathbf{V}_i$, $\mathbf{V}_{i+1}$, $\mathbf{V}_{i+2}$. Then the test variable associated with each vertex of a polygon is

$$\omega_i = (\mathbf{P}-\mathbf{V}_i)\cdot\mathbf{E}_i,$$

where

$$\mathbf{E}_i = (\mathbf{V}_i-\mathbf{V}_{i+1})(\mathbf{V}_i-\mathbf{V}_{i+1})\cdot(\mathbf{V}_{i+1}-\mathbf{V}_{i+2}) - (\mathbf{V}_{i+1}-\mathbf{V}_{i+2})(\mathbf{V}_i-\mathbf{V}_{i+1})\cdot(\mathbf{V}_i-\mathbf{V}_{i+1})$$

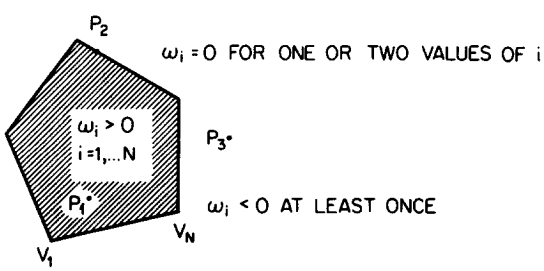

FIG. 8. $P_{1,2,3}$ are internal/boundary/external points. The associated ω_i lie in the ranges shown, respectively.

is precomputed. If the N-gon is known to be convex (see Section 5), then **P** must lie to the right of each and every edge when viewed from above. There are three possibilities for each vertex, see Fig. 8. If **P** lies inside the polygon, then $\omega_i > 0$ for all values of $i = 1, ..., N$. If **P** lies on one of the edges, then $\omega_i > 0$ for all values of $i = 1, ..., N$ but one or perhaps two ($i = M$) where $\omega_M = 0$ (within computational tolerance). If $\omega_i < 0$ for any value of i, the point **P** lies outside the polygon.

Note that:

(1) the vectors $\mathbf{E}_i$ depend only on the geometry of the polygon and may be precomputed;

(2) ω_i is a scalar invariant and is thus independent of the coordinate system;

(3) only five additions and three multiplications are needed to find ω_i when $\mathbf{E}_i$ is precomputed;

(4) only if P lies inside the polygon is it always necessary to make N determinations of ω_i;

(5) the method is valid for open or closed polygons by simply reducing the value of N by one, provided the convention of Section 2 is adhered to;

(6) ω_i does not change sign upon reversing the order of the vertices (viewing the polygon from below);

(7) the method trivially generalizes to more than three dimensions.

On most counts the test appears better suited for physics computations in many dimensions than the older algorithms, except if the vector $\mathbf{E}_i$ cannot for some reason be precomputed, if the polygon is known to be non-convex, or if homogeneous coordinates are being employed. If the polygon is known to be non-convex, the method of Yamaguchi may be used to subdivide it, in which case the test can be applied to the primal triangular subdivisions. The method of convex inclusion should be considered for those problems in which N is known to be large, the polygons are known to be closed, and ordering invariance is not needed, since this method scales as $\log N$. Although both the method of Yamaguchi and the new method scale as N (with different coefficients in the worst case (P inside the polygon)), in the best case both of these methods could detect an exterior point in as little as one determination of their respective test variables. A stricture concerning the wisdom of blind adherence to asymptotic analysis can be found in Section 1.2 of Ref. [4].

5. A Test for Convexity

As a corollary, we find that the new algorithm generates a simple test for the convexity of a polygon in n-space. Consider any ternary grouping of continguous vertices (Fig. 9). The centroid $\mathbf{P}_i$ given by

$$\mathbf{P}_i = (\mathbf{V}_{i-1} + \mathbf{V}_i + \mathbf{V}_{i+1})/3$$

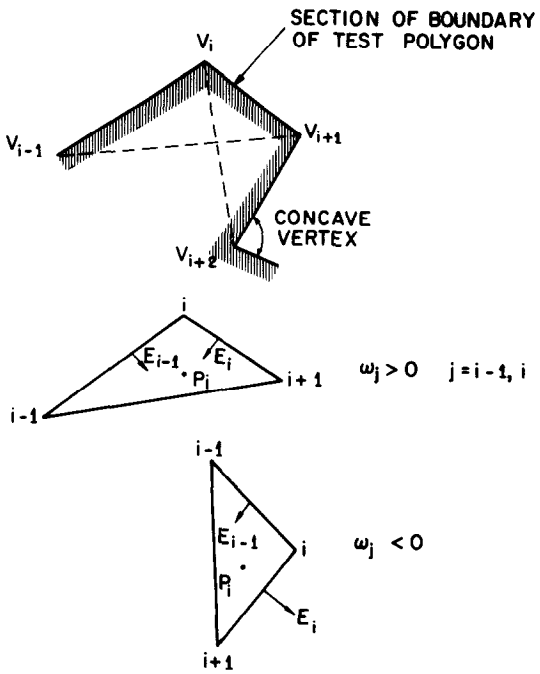

FIG. 9. A test for convexity. Test the centroid of ternary groupings of vertices for inclusion. The test fails at the grouping prior to a concave vertex, if one exists.

must be an interior point of each triangle. Apply the new algorithm to each of the exterior edges of the triangle by evaluating

$$\omega_j = (\mathbf{P}_i - \mathbf{V}_j) \cdot \mathbf{E}_j, \qquad j = i-1, i.$$

If $\omega_j < 0$, the angle bounded by V_j, V_{j+1}, V_{j+2} is $> \pi$ and the polygon is non-convex. If $\omega_j > 0$ for all contiguous ternary groupings, the polygon is convex.

ACKNOWLEDGMENTS

This work was sponsored by HTFS, the Heat Transfer and Fluid Flow Service of Atomic Energy Research Establishment, Harwell, UK and Chalk River Nuclear Laboratories, Atomic Energy of Canada Limited.

REFERENCES

1. R. J. LITTLEFIELD, "Basic Geometrical Algorithms for Use with Graphic Input," Pacific Northwest Laboratory report PNL-SA-12161, National Computer Graphics Association, Anaheim, CA, May 1984.

2. K. MEHLHORN, *Data Structures and Algorithms 3: Multi-Dimensional Searching and Computational Geometry* (Springer-Verlag, Berlin, 1984).
3. T. PAVLIDIS, *Algorithms for Graphics and Image Processing* (Computer Science, Rockville MD, 1982), p. 330.
4. F. P. PREPARATA AND M. I. SHAMOS, *Computational Geometry, An Introduction* (Springer-Verlag, New York, 1985), Sect. 2.2.
5. M. SHIMRAT, *Commun. ACM*, pp. 434, 606 (1962).
6. C. WOOFF, *Comput. Phys. Commun.* **36**, 219 (1985).
7. F. YAMAGUCHI, *A Unified Approach to Interference Problems Using a Triangle Processor*, Proceedings, SIGGRAPH'85, ACM, New York, 1985, p. 141.