[10] F. P. Preparata and S. J. Hong, “Convex hulls of finite sets of points in two and three dimensions,” *Communications of the ACM*, vol. 20, pp. 87-93, 1977.

[11] K. Ichida and T. Kiyono, “Segmentation of plane curves,” *Trans. Elec. Commun. Eng.*, Japan, vol. 58-D, pp. 689-696, 1975.

[12] H. Imai and M. Iri, “Polygonal approximations of a curve: Formulations and solution algorithms,” in *Computational Morphology*, G. T. Toussaint, Ed. Amsterdam, The Netherlands: North-Holland, to be published.

[13] Y. Kurozumi and W. A. Davis, “Polygonal approximation by the minimax method,” *Computer Graphics and Image Processing*, vol. 19, pp. 248-264, 1982.

[14] G. T. Toussaint, “Solving geometric problems with the rotating calipers,” in *Proc. MELECON’83*, Athens, Greece, 1983.

[15] G. T. Toussaint, “Approximating polygonal curves in three-dimensions,” manuscript in preparation.

4. Conclusion

It has been shown that the width of a convex polyhedron in 3-space can be computed in O($n + I$) time. The main open question regarding this problem is to determine whether this algorithm is optimal. No non-trivial lower bound is known on the complexity of this problem. Another area for further investigation concerns the generalization of these results to dimensions higher than three. Finally, an interesting open problem concerns the related area of minimax approximation. We have seen in Sections 2-D and 3-C that the width of a set, as defined in this paper, solved the problems of finding a minimax approximating line of a set of points in the plane as well as that of finding the minimax approximating plane of a set of points in 3-space. An interesting intermediate problem calls for finding the minimax approximating line of a set of points in 3-space. The concepts developed in this paper do not seem applicable in this case. Note that this is equivalent to finding the thinnest "tube" that encloses the set, and the diameter of the cross-section of this "tube" is an alternate definition of the width of a set in 3-space and has applications to approximating polygonal curves in three dimensions [15].

5. Acknowledgment

The authors are grateful to the referees for helpful suggestions leading to improvements of the manuscript. Following the publication of [8], both the referees and the authors independently observed that the result of [8] could be used to improve the complexity of computing the width of a polyhedron to O($n + I$) time.

6. References

[1] G. T. Toussaint, "Movable separability of sets," in *Computational Geometry*, G. T. Toussaint, Ed. Amsterdam, The Netherlands: North-Holland, 1985, pp. 335-375.

[2] M. I. Shamos, "Computational geometry," Ph.D. dissertation, Yale University, 1978.

[3] K. Q. Brown, "Geometric transforms for fast geometric algorithms," Dept. Computer Science, Carnegie-Mellon University, 1979.

[4] D. G. Kirkpatrick and R. Seidel, "The ultimate planar convex hull algorithm?", Dept. Comput. Sci., Cornell University, Tech. Rep. 83-557, Oct. 1982

[5] M. M. McQueen and G. T. Toussaint, "On the ultimate convex hull algorithm in practice," *Pattern Recognition Letters*, pp. 29-34, Jan. 1985.

[6] D. McCallum and D. Avis, "A linear time algorithm for finding the convex hull of a simple polygon," *Inform. Processing Lett.*, pp. 201-205, Dec. 1979.

[7] D. T. Lee, "On finding the convex hull of a simple polygon," Tech. Northwestern University, Tech. Rep. 80-03-FC-01, Mar. 1980.

[8] L. Guibas and R. Seidel, "Computing convolutions by reciprocal search," in *Proc. 2nd Symposium on Computational Geometry,* Yorktown Heights, NY, 1986, pp. 90-99.

[9] M. E. Houle and G. T. Toussaint, "Computing the width of a set," Tech. Rep. SOCS-84.22, Dec. 1984.

Algorithm 3.1 - Width in Three Dimensions:

Begin

> *Step 1:* Construct the convex hull in O(n log n) time using Preparata and Hong's algorithm [10].
>
> *Step 2:* Transform the polyhedron into two planar subdivisions in O(n) time as described in [3].
>
> *Step 3:* Compute the overlay of the two planar subdivisions using Guibas and Seidel's algorithm [8] in O(n + I) time.
>
> *Step 4:* Examine all vertices and all parallel rays of unbounded regions of the overlay to generate V-F and E-E pairs in O(I) time, and report the smallest antipodal pair distance as being the width.

End

Theorem 3.2: Algorithm 3.1 correctly finds the width of a set of points $P = \{p_1, p_2,..., p_n\}$ in three dimensions in O(n log n + I) time and O(n) space, where I is the number of antipodal edge-edge pairs of *CH(P)*.

Corollary 3.2: The width of a polyhedron P in three dimensions can be computed in O(n + I) time and in O(n) space, where I is the number of antipodal edge-edge pairs of *CH*(P).

Note that in the worst case, I is $\Omega(n^2)$, as in the example shown in Fig. 3. Also, note that if the diameter is desired, one may examine all O(I) regions of the convex overlay in step 4, and generate V-V pairs instead of V-F and E-E pairs. The pair with the largest vertex-vertex distance determines the diameter. Thus, using these methods, the complexity of finding the diameter of a set of three dimensions is the same as that of finding the width.

C. Minimax Approximating Plane

Given a set of points P in 3-space, the minimax-approximating-plane problem consists of determining a plane H such that the maximum distance between a point $p \in P$ and H is minimized. Here the distance used is as in Section 2-D.

Theorem 3.3: Given a set of points $P = \{p_1, p_2,..., p_n\}$ in three dimensions, the minimax approximating plane of P can be computed in O(n log n + I) time and O(n) space.

Proof: From Theorem 3.2 it follows that the width of P can be computed in O(n log n + I) time. From Theorem 3.1 we obtain that the width of P is determined by either a V-F or E-E pair. In both cases the minimax approximating plane can be computed in constant time. Q.E.D.

of the other (a V-F pair). Because the orientation of the convex hull of P is such that no face is vertical, no V-F pairs may be located at infinity in the subdivision overlay. Unfortunately, this is not true of E-E pairs. These pairs may occur at vertices of the overlay, or at the intersection of parallel rays at infinity.

To generate all V-F and E-E pairs, we examine all vertices and some pairs of parallel rays of the overlay. Since these rays occur in increasing order of slope as one travels along the convex hull of overlay vertices in the counter-clockwise direction, we need only check rays bordering the same unbounded region for parallelism. This we can do for all O(I) unbounded regions in constant time each.

Finally, we must note that the distance between parallel planes of support containing an E-E pair may be computed in constant time, as long as the edges in question are parallel. If they are not parallel, then the pair may be ignored.

Lemma 3.2: Two nonparallel lines l_1 and l_2 in 3-space uniquely determine a pair of parallel planes p_1 and p_2 such that p_1 contains l_1 and p_2 contains l_2.

Lemma 3.3: Given two nonparallel lines in 3-space, the minimum distance between parallel planes containing the lines can be computed in O(1) time.

We now summarize the algorithm.

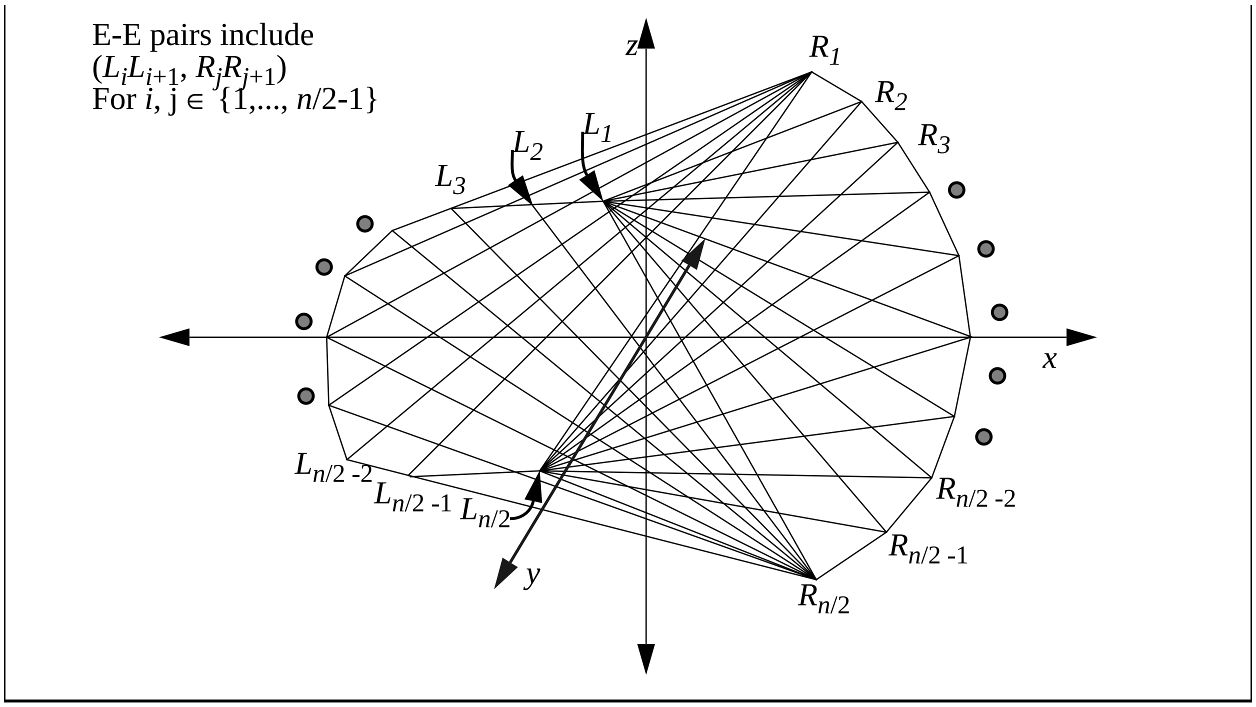

Fig. 3

As in the two-dimensional case, the polyhedron is divided into an *upper* and a *lower* half-hull, and each half-hull is then transformed into a planar subdivision. A face on the half-hull maps into a vertex of the subdivision, an edge maps into an edge, and a vertex maps into a region (see Fig. 2). Brown outlines a method for computing this transformation in linear time [3]. He finds the diameter of the polyhedron by generating antipodal pairs of vertices. Noting that parallel planes of support must map onto the same point in $\Re^2$ under the transformation, and that these planes must meet the polyhedron in different half-hulls, each antipodal pair of vertices then corresponds to a non-null intersection of two regions of the two subdivisions.

To find the width, we generate all antipodal edge-edge and vertex-face pairs, by first computing the convex subdivision overlay of the *upper* and *lower* subdivisions. Using Guibas and Seidel's algorithm [8], which uses a variant of the planar sweep technique, this may be accomplished optimally in O(n + I) time and O(n) space, where I is the number of edges of the overlay. Each vertex of the overlay is formed either by the intersection of an edge from the *upper* subdivision and an edge of the *lower* subdivision (an E-E pair), or by a vertex of one subdivision located in a region

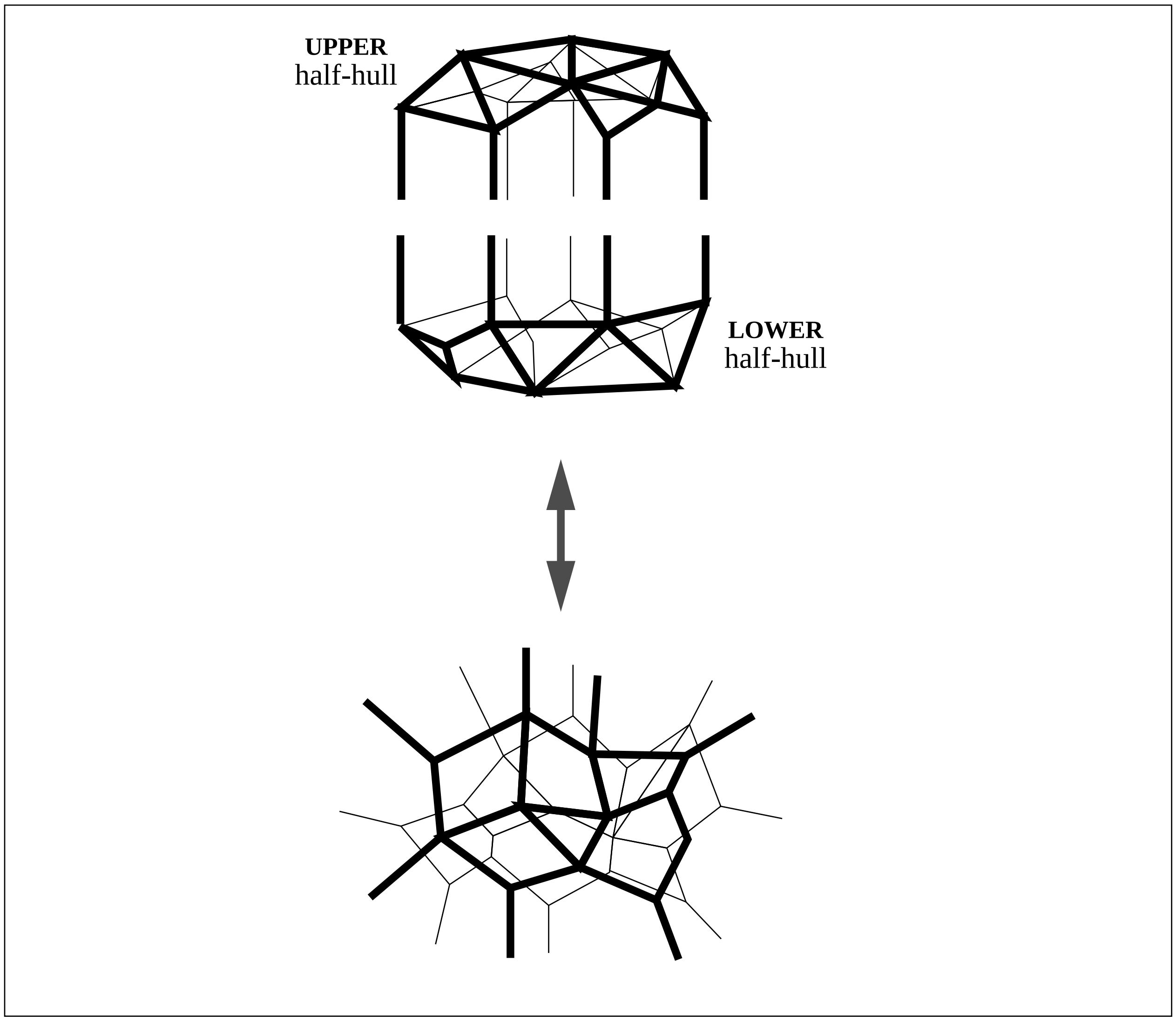

Fig. 2.

Lemma 3.1: Let *a* and *b* be parallel lines in 3-space. Let π_1 and π_2 be distinct parallel planes containing *a* and *b*, respectively. Then there exists a preferred direction of rotation such that if π_1 and π_2 are rotated about *a* and *b* in that direction to form the new parallel planes π_1' and π_2', then we have that $d(\pi_1', \pi_2') < d(\pi_1, \pi_2)$.

Theorem 3.1: The width of a set of points *P* in three dimensions is the minimum distance between parallel planes of support passing through either an antipodal vertex-face pair or an antipodal edge-edge pair of *CH(P)*.

Proof: Assume otherwise. Then the width must have been strictly determined by and E-F, F-F, V-V or V-E pair.

Case 1 (F-F pair): Here the width is the distance between two particular extended faces of the convex hull. Since this distance is the same as the distance between a vertex of one face and the other extended face, it reduces to the case of a V-F pair.

Case 2 (E-F pair): The argument of case 1 also applies here to reduce this case to that of a V-F pair.

Case 3 (V-V pair): Since the V-V pair admits parallel lines of support, we may rotate the parallel planes of support in the PDR in order to decrease the distance between them. This contradicts the assumption that the V-V pair determined the width.

Case 4 (V-E pair): Since the V-E pair admits parallel lines of support, the argument of case 3 applies. Q.E.D.

Corollary 3.1: The width of a polyhedron *P* in three dimensions is the minimum distance between parallel planes of support passing through an antipodal vertex-face pair or an antipodal edge-edge pair of *CH*(*P*).

The tetrahedra in Fig. 1 are examples of polyhedra where the minimum distance between parallel planes of support occurs at a vertex-face pair and an edge-edge pair, thus establishing both possibilities. Note that from Lemma 3.1, it is clear that an E-E pair with parallel edges does not determine the width.

B. Three-Dimensional Width Algorithm

We now extend the method used in Section 2-C to solve the three-dimensional width problem. The transform we use is based on the "slope" of the planes of support of the convex hull of the point set *P*. These planes can be divided into two categories: those which are orthogonal with respect to the *xy*-plane (called "vertical") and those which are not. We shall restrict ourselves to polygons with non-vertical faces, noting that if vertical faces do exist, the polyhedron may be rotated so as to produce a polyhedron without vertical faces in time proportional to the number of faces. The slope of a non-vertical plane is really a two-dimensional vector, so every such plane maps into a point in $\Re^2$.

Plane: $z = ax + by + c \rightarrow (a, b) \in \Re^2$.

where $d(p, x)$ is the euclidean distance between x and p. From the properties of the solution line, it follows that once the width of P is known, l can be computed in O(1) time [13]. We therefore have the following result.

Theorem 2.3: Given a set of points $P = \{p_1, p_2, ..., p_n\}$ in two dimensions, the minimax approximation line of P can be computed in $O(n \log h)$ time.

3. The Width in Three Dimensions

A. Characterization of the Width

We turn now to the principal contribution of this paper. In extending the notion of the width of a planar set to three dimensions, one might expect at first glance that it will always be determined by a vertex and a face of the convex hull of the set. It turns out that this is not true in general - the width may also be determined by parallel planes of support, each of which contains an edge of the polyhedron. As in two dimensions, we define two edges of the convex hull of a set of points P in three dimensions as being an *antipodal edge-edge* (E-E) *pair* if parallel planes of support of P contain these edges. Similarly, we define *vertex-vertex* (V-V), *vertex-edge* (V-E), *vertex-face* (V-F), *edge-face* (E-F), and *face-face* (F-F) *pairs*. Since a plane of support may only meet a convex polyhedron at a vertex, edge, or face, these are the only combinations possible.

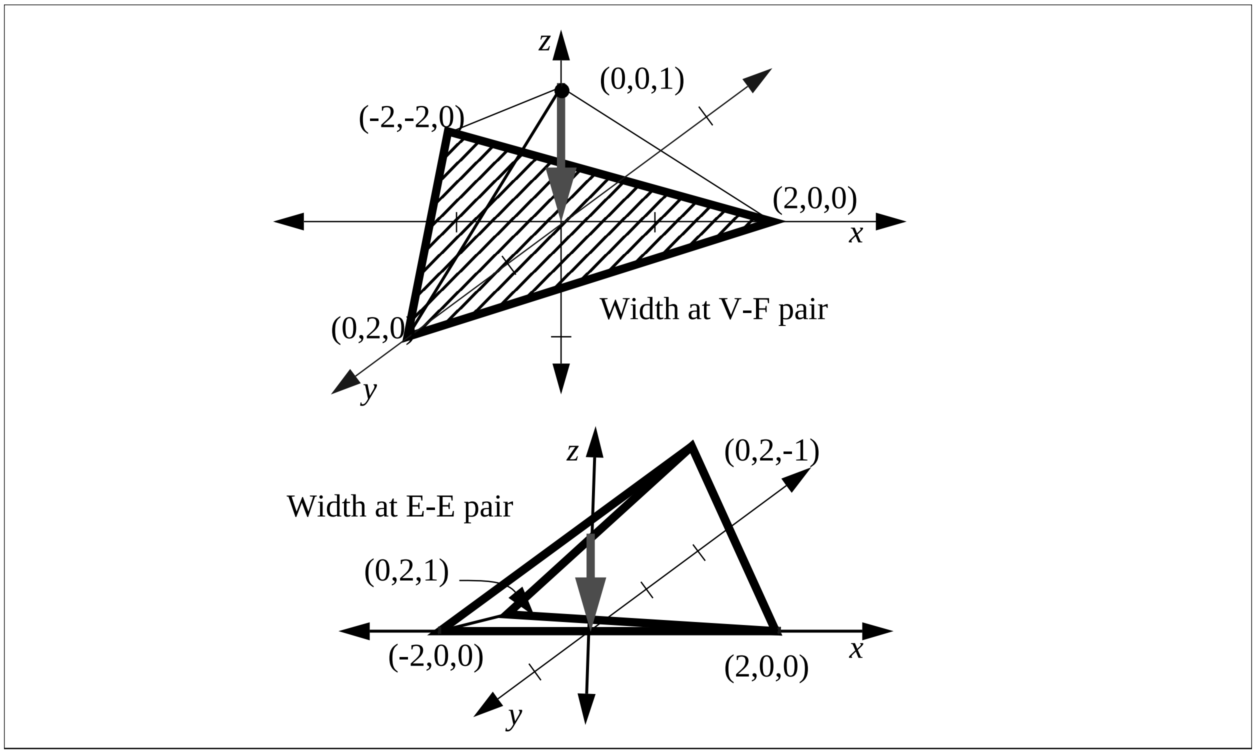

Fig. 1.

To find the diameter of the convex *n*-gon *P*, all of the possible pairs of antipodal vertices are generated by considering where the intervals of the *upper* and *lower* lines overlap. Each such intersection corresponds to a unique pair of vertices of *P* with lines of support with the same slopes. All intersections are determined in O(*n*) time by scanning the set of intervals of one line to determine the interval of the other line in which it lies.

Finding the width of a set of points *P* now reduces to considering where all points on each of the *upper* and *lower* lines lie with respect to the intervals of the other. If parallel lines of support exist at an edge and a vertex of the convex hull of *P*, then the point and the interval of the *upper* and *lower* lines corresponding to the edge and the vertex of *CH(P)*, respectively, must intersect. Also, every such intersection corresponds to a unique V-E pair. The scans for these intersections may be done in O(*h*) time because of the sorted order of the intervals.

To summarize, the width of a set of points *P* may be computed in O(*n* log *h*) time in the following manner.

Algorithm 2.2 - Width Using Geometric Transforms:

Begin

Step 1: Construct the convex hull in O(*n* log *h*) time as in step 1 of Algorithm 2.1

Step 2: Apply the transform to obtain two ordered sets of intervals in O(*h*) time.

Step 3: Scan the sets for intersections between the intervals of one set and the interval endpoints of the other, generating V-E pairs in O(*h*) time. For each such pair, compute the distance between the vertex and the extended edge, and note the smallest such distance. When the scan is complete, that distance is the width.

End

We therefore have the following result.

Theorem 2.2: Algorithm 2.2 correctly finds the width of a set of points $P = \{p_1, p_2, ..., p_n\}$ in two dimensions in O(*n* log *h*) time, where *h* is the number of vertices of the convex hull of *P*.

Corollary 2.2: The width of a simple *n*-gon may be computed in O(*n*) time using geometric transforms.

D. Minimax Approximation Line of a Set

Given a set of points *P* in the plane, the minimax approximation problem consists of determining a straight line *l* such that the maximum distance between a point $p \in P$ and the line *l* is minimized. Here the distance between *p* and *l* is defines as

$$d(p, l) = \min_{x \varepsilon l} \{d(p, x)\}$$

B. Rotating Caliper Algorithm

The first algorithm presented here is pattered after that of Shamos [2], where the diameter of a set of points is obtained through the use of “rotating calipers” [14]. This algorithm was also suggested independently by Iri and Imai [12] recently. We include a sketch for the sake of completeness.

The convex hull *CH*(*P*) of *P* is first constructed. Then an initial antipodal V-E pair is found by choosing an arbitrary edge of *CH*(*P*) and finding its antipodal vertex in O(log *n*) time using binary search as suggested in [13]. However, repeating this for every edge, as is done in [13], leads to a complexity of O(*n* log *n*). Instead, we apply the rotating calipers method to generate all V-E pairs in O(*n*) time. Let *h* denote the number of vertices on the convex hull of *P*.

Algorithm 2.1 - Width Using Rotating Calipers:

Begin

Step 1: Compute *CH(P)*.

Step 2: Find initial V-E pair in O(log *h*) time (*h* is the number of convex hull vertices).

Step 3: Use the rotating calipers to generate all V-E pairs in O(*h*) time, noting the distance between vertex and edge.

Step 4: Report the minimum V-E distance as the width.

End

Step 1 of the algorithm may be done in O(*n* log *h*) time using the convex hull algorithm of Kirkpatrick and Seidel, yielding O(*n* log *h*) time overall [4], [5]. Since the convex hull of a simple *n*-gon may be computed in O(*n*) time [6], [7], step 1 may be changed to give an O(*n)* time algorithm for simple *n*-gons.

C. Computing the Width Using Geometric Transforms

Brown’s algorithm [3] to find the diameter of a convex polygon may also be modified to yield the width. He notes that the most important feature of parallel lines of support is that they have the same slope, which allows us to ignore the actual Cartesian coordinates of the vertices of the polygon in favor of the slopes of the lines of support that are admitted by each vertex and edge.

His first step is to divide the polygon into *upper* and *lower* chains, so that when one line of support meets the polygon on the *upper* chain, the other lies on the *lower* chain. The transform of an edge of the polygon is defined as the slope of the line containing that edge. The transform of a vertex of the polygon is defined as the set of slopes of the lines of support passing through it. Since every vertex lies between two edges, this set of slopes is a closed interval. When the *upper* and *lower* sets of points on the line are generated, they appear in sorted order because the slopes of the edges of the polygon are already sorted. This leaves us with *upper* and *lower* lines of slopes, divided into intervals corresponding to vertices of the polygon, which can be obtained in linear time.

of the polyhedron that admit parallel planes of support. In the worst case I can be $\Omega(n^2)$ but typically, it is only $O(n)$. Brown's original diameter algorithm has worst-case time complexity of $O(n^2 \log n)$. However, its complexity can be reduced to $O(n + I)$ time if Guibas and Seidel's convex subdivision overlay algorithm [8] is used to report intersections between two sets of regions in the plane.

2. The Width in Two Dimensions

A. Characterization of the Width

Before presenting the width-finding algorithms, it is useful to characterize the width of a set of points, and to introduce some terminology. When we say that A is *contained* in B, we mean that every point of A also lies in B. We call any two edges of the convex hull $CH(P)$ of a set of points P an *antipodal edge-edge* (E-E) pair if parallel lines of support of P contain these edges. Similarly, we can define *vertex-vertex* (V-V) and *vertex-edge* (V-E) pairs. Note that these are the only possible combinations, since a line of support of P must intersect $CH(P)$ either at a single vertex, or along one of its edges. Define $d(l_1, l_2)$ to be the minimum Euclidean distance between a point $x \in l_1$ and a point $y \in l_2$. Note that in the interest of brevity, proofs of simple lemmas are omitted and left as exercises for the reader. For the lazy reader they can be found in [9].

Lemma 2.1: Let a and b be vertices in the plane. Let l_1 and l_2 be distinct parallel lines containing a and b, respectively. Then there exists a *preferred direction of rotation* (PDR) such that if l_1 and l_2 are rotated about a and b in the PDR to form new parallel lines l_1' and l_2', $d(l_1', l_2') < d(l_1, l_2)$.

Theorem 2.1: The width of a set of points P in two dimensions is the minimum distance between parallel lines of support passing through an antipodal vertex-edge pair of *CH(P)*.

Proof: Assume otherwise. Then the width must have been determined by either a strict E-E or V-V pair.

Case 1 (E-E pair): Here the width of the set of points is the distance between two particular (extended) edges of its convex hull. Since this distance is the same as the distance between an endpoint of one of the edges and the other (extended) edge, it reduces to the case of a V-E pair.

Case 2 (V-V pair): We may rotate the parallel lines of support in the PDR in order to decrease the distance between them. This contradicts the assumption that the width occurred at this V-V pair. Q.E.D.

Corollary 2.1: The width of a simple polygon P in two dimensions is the minimum distance between parallel lines of support passing through an antipodal vertex-edge pair of *CH(P)*.

(1)

We note here that this result has been known for some time [11].

Computing the Width of a Set

Michael E. Houle
and
Godfried Toussaint,

ABSTRACT

Given a set of points $P = \{p_1, p_2, ..., p_n\}$ in three dimensions, the width of P, $W(P)$, is defined as the minimum distance between parallel planes of support of P. It is shown that $W(P)$ can be computed in $O(n \log n + I)$ time and $O(n)$ space, where I is the number of antipodal pairs of edges of the convex hull of P, and in the worst case $I = \Omega(n^2)$. For convex polyhedra, the time complexity becomes $O(n + I)$. If P is a set of points in the plane, the complexity can be reduced to $O(n \log n)$. Finally, for simple polygons linear time suffices.

Index Terms - Algorithms, antipodal pairs, artificial intelligence, computational geometry, convex hull, geometric complexity, geometric transforms, image processing, minimax approximating line, minimax approximating plane, pattern recognition, rotating calipers, width.

1. Introduction

The width of a set of points P (or a simple polygon $\boldsymbol{P}$) in two dimensions is the minimum distance between parallel lines of support of P (or $\boldsymbol{P}$). In three dimensions, it is the minimum distance between parallel planes of support. This notion of width is closely related to that of diameter (the maximum distance between parallel lines or planes of support), and the methods that will be presented reflect this. Computation of the width has applications in collision avoidance problems [1], and in approximating polygonal curves [11]-[13].

The two-dimensional width problem has received some attention from the computational point of view in [13], where two algorithms are proposed for computing the width of a convex polygon with n vertices. One algorithm runs in $O(n^2)$ time in the worst case and the other incorporates binary search to reduce this complexity to $O(n \log n)$.

In this paper two algorithms to find the width of a convex polygon will be presented, one based on Shamos' "rotating calipers" method for computing the diameter [2], and the other on Brown's method using geometric transforms [3]. Both of these algorithms run in $O(n)$ time and $O(n)$ space. A modification of Brown's algorithm for finding the diameter of a convex polyhedron in three dimensions computes the width in $O(n + I)$ time, where I is the number of pairs of edges

Manuscript received June 12, 1986; revised April 3, 1987. Recommended for acceptance by J. O'Rourke. This research was supported by NSERC Grant A9293, FCAC Grant EQ1678, and a Killam Senior Research Fellowship award to G. T. Toussaint by the Canada Council.
The authors are with the School of Computer Science, McGill University, Montreal, P.Q. H3A 2K6, Canada
IEEE Log Number 8823183.