AD-A058 787 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/2
FAST INTERSECTION OF HALF SPACES.(U)
JUN 78 K Q BROWN N00014-76-C-0829
UNCLASSIFIED CMU-CS-78-129 NL

| OF |
AD
A058 787

END
DATE
FILMED
-11-78
DDC

CMU-CS-78-129

LEVEL II 12

Fast Intersection of Half Spaces

Kevin Q. Brown

June 29, 1978

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

CMU-CS-78-129

Fast Intersection of Half Spaces

Kevin Q. Brown

June 29, 1978

Department of Computer Science
Carnegie - Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

The problem of intersecting N half-spaces in K space is transformed to ~~2*K~~ problems of constructing the convex hull of N points in K space and a simple intersection problem. This enables one to intersect the N K-dimensional half spaces in O(K*H(N,K)) time, where H(N,K) is the time required to construct the convex hull of N points in K space. For two and three dimensions the algorithm takes O(NlogN) time in the worst case, but under fairly robust conditions the expected time is only O(N). It is also shown that an algorithm for intersection of half spaces can be used to construct the convex hull of points in K space. Thus, the intersection of half spaces and convex hull of points problems are essentially equivalent.

This research was partially supported by the Office of Naval Research under contract number N00014-76-C-0829.

1. Introduction

There has been much recent work on the problem of intersecting half spaces. Shamos[75] and Shamos and Hoey[76] describe an O(NlogN) time algorithm in two dimensions, and Zolnowsky[77] and Preparata and Muller[77] have developed O(NlogN) time algorithms for three dimensions. These algorithms are optimal within a constant factor because there is an Ω(NlogN) lower bound for two or more dimensions (Shamos[75]).

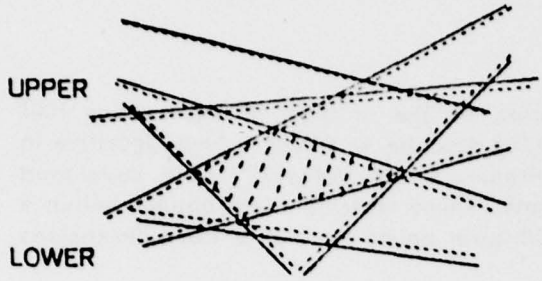
We will show that an algorithm for constructing the convex hull of a set of points in K space can be used to intersect half spaces in K space, and vice-versa. In fact, within a factor of K, the upper and lower bounds for one problem apply to the other. Thus, (much of) the knowledge concerning convex hulls of points in K space applies to the intersection of half spaces in K space. Graham[72] and Preparata and Hong[77] describe O(NlogN) time algorithms for constructing the convex hull of N points in the plane. Bentley and Shamos[78] and Eddy[77] give fast expected time algorithms in the plane. Preparata and Hong[77] have also produced an O(NlogN) time algorithm for the convex hull of N points in 3 space. However, in four dimensions there is an $\Omega(N^2)$ lower bound because the convex hull can have $\theta(N^2)$ edges (Grunbaum[67], p.193). Chand and Kapur[70] describe a convex hull algorithm for an arbitrary number of dimensions whose complexity has yet to be analyzed.

This paper will first describe the two dimensional problem and the solution obtained by use of the transform. Then we move up to three dimensions, using the transform to solve the problem in O(NlogN) time. Finally, four (and more) dimensions will be tackled, followed by some fast expected time algorithms and applications. All algorithms in this paper use the RAM (with reals) model of computation.

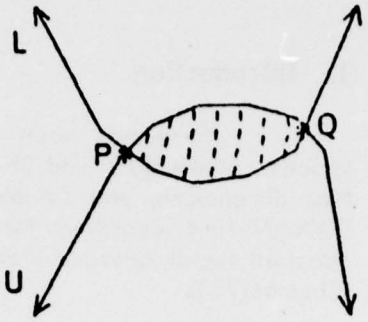
2. The two dimensional problem

Shamos[75] and Shamos and Hoey[76] show that the intersection of N half planes has time complexity θ(NlogN). Their algorithm for constructing the intersection in O(NlogN) time relies on Shamos' O(N) time algorithm for intersection of convex polygons. In this paper an entirely different O(NlogN) time algorithm is described.

In Figure 1 the intersection of N half planes is indicated by the shaded region. The half planes are partitioned into two sets, UPPER and LOWER. A half plane is in set UPPER if the line at its boundary is above the rest of the half plane. Similarly, a half plane is in the set LOWER if the line at its boundary is below the rest of the half plane. (If any boundary lines are vertical, then rotate all N half planes a small angle.) The reason for producing this partition will not be explained now, but it will be apparent later. Thus, the problem is now divided into three parts: (1) the intersection of the UPPER half planes, (2) the intersection of the LOWER half planes, and (3) the intersection of the results of parts (1) and (2).



(1)



(2)

Figure 1: Intersection of N half planes., Figure 2: Intersection of regions U and L.

As shown in Figure 2, part (3) is relatively easy. If U and L in Figure 2 have O(N) vertices then the intersection (shaded region) can be constructed in O(N) time. The algorithm is described in detail in ALGORITHM INTERSECTCHAINS below.

ALGORITHM INTERSECTCHAINS

Input: Integers N1, N2 (where N = N1 + N2) and real vectors $U_x[0:N1+1]$, $U_y[0:N1+1]$, $L_x[0:N2+1]$, $L_y[0:N2+1]$ such that

$$-\infty = U_x[0] < U_x[1] < \ldots < U_x[N1] < U_x[N1+1] = \infty$$
$$-\infty = L_x[0] < L_x[1] < \ldots < L_x[N2] < L_x[N2+1] = \infty.$$

Output: Integer K (number of vertices on the intersection), $H_x[1:K]$, $H_y[1:K]$.
Time: O(N), Space: O(N).

(1) Scan U and L (vectors U_x, U_y, L_x, and L_y) from left to right until two segments intersect at a point P. (If no segments intersect then the intersection of U and L is empty.) The scan can be done in O(N) time in a manner similar to the O(N) time merge in the merge sort algorithm.

(2) Scan U and L from right to left until two segments intersect at a point Q.

(3) If $P \neq Q$, then return (in vectors H_x and H_y) the concatenation of the chains of line segments of U and L between points P and Q.

(4) If P = Q, then the intersection is unbounded (or just the point P = Q). In the case of an unbounded intersection one must determine whether to return the chains to the left of P or the chains to the right of P. This can be determined by comparing the slopes of the rays bounding the left and right sides of U and L. If the slope of the left ray of U is less than the slope of the left ray of L, then return the chains to the left of P. Otherwise, return the chains to the right of P.

This leaves parts (1) and (2), the intersection of the UPPER half planes and the intersection of the LOWER half planes. Only (1) will be described since (2) can be solved similarly. Assume that the N half planes are all UPPER half planes. Some of these half planes, such as half plane k in Figures 3a and 3b, do not bound any side of the region of intersection. It would be nice to be able to find all such half planes and throw them away since they do not contribute to the final result. Once that is done one can find the intersection of the UPPER half planes rather easily. As one can see in Figure 2, the slopes of the sides of the chain U are monotonic decreasing as one travels from left to right. Thus, given the lines determined by the sides, one need only sort the lines by slope to determine the order in which they intersect to form the sides. The sort costs O(NlogN) time so once all the redundant half planes, such as k in Figures 3a and 3b, are removed, the intersection of the UPPER half planes can be determined in O(NlogN) time.

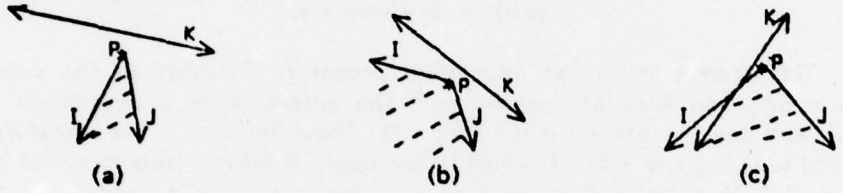


Figure 3: (a) & (b) - k is redundant, (c) - k is nonredundant.

How does one determine which half planes are redundant and which are not? There are two conditions which need to be checked.

(1) Line k (Figure 3) must be above the point P where lines i and j meet.

(2) The slope of line k must lie between the slopes of lines i and j. That is, slope(i) < slope(k) < slope(j) or slope(j) < slope(k) < slope(i). This is true if i and j have slopes of different sign (Figure 3a) or of the same sign (Figure 3b).

Figure 3c shows what will happen if the first condition is satisfied but the second is not. A half plane k is therefore redundant iff there exist half planes i and j such that the two above conditions are satisfied.

How fast can one determine (non)redundancy for each of the N UPPER half planes? Certainly one approach is to test all pairs of half planes i and j for each halfplane k. That costs $O(N^3)$ time, though, which is not good. One could, perhaps, modify this approach and obtain a faster algorithm. However, there is an entirely different way of looking at this problem that is clearly better and that is the approach which will be taken.

3. The Transform

The transform exploits a natural duality between points and lines in the plane. For an arbitrary point (x,y), consider the set of all lines in slope-intercept form which pass through the point. (Thus this won't work for vertical lines.) If a line is represented by an ordered pair of the form (slope,intercept), then this set is { (a,b) | y = a*x + b }. Note that this transforms a point (x,y) to a line b = -x*a + y. Note also that the representation of a line y = a*x + b in the form (a,b) is a natural transform of a line to a point. Thus, points transform to lines and lines transform to points by the formulas

$$y = a*x + b \rightarrow (a,b), \text{ and}$$
$$(x,y) \rightarrow b = -x*a + y.$$

This transform has an interesting property: Distances *in the y-coordinate* between points and lines are preserved.[1] The difference in y coordinate between point (c,d) and line y = e*x + f is d - (e*c + f). The difference in the transforms b = -c*a + d and (e,f) is (-c*e + d) - f, which is the same. It follows from this that incidence is preserved.[2] If point (c,d) is on line y = e*x + f, then it holds also for their transforms - point (e,f) is on line b = -c*a + d. Note further that not only is the magnitude of the distance (in the y-coordinate) preserved but also its sign. Thus, above/below-ness is preserved. If (c,d) is above (below) line y = e*x + f, then the transform of (c,d) is above (below) the transform of y = e*x + f.

There is another property of the transform which should be mentioned. The transform is not involutory, but composition of it *four* times produces the following:

$$(x,y) \rightarrow b = -x*a + y \rightarrow (-x,y) \rightarrow b = x*a + y \rightarrow (x,y)$$

Only a slight change is required to make the transform its own inverse. Express lines in the form y + a*x + b = 0 rather than y = a*x + b. Then it is true that y + a*x + b = 0 $\leftrightarrow$ (a,b). But this has the unfortunate side effect that above/below-ness between points and lines is not preserved; it is reversed. If point (c,d) is above line y + e*x + f = 0 then the transform of (c,d) will be *below* the transform of line y + e*x + f = 0.

---

[1] The restriction *in the y coordinate* is important because it can be shown to be impossible to preserve the *Euclidean* distance between a point and a line under a duality transform.

[2] There are other duality transforms which preserve incidence, such as Plucker's transform [Shamos 77].

4. Application of the transform to the two dimensional problem

Now it will be shown how the transform enables one to intersect the UPPER (or LOWER) half planes fast. More specifically, the transform enables an efficient mechanism for eliminating the "redundant" half planes. Recall the two conditions for redundancy of an UPPER half plane: a half plane k is redundant iff there exist half planes i and j such that (1) line k is above the point P where lines i and j intersect, and (2) the slope of k is between the slopes of lines i and j. In the ab plane there is a corresponding interpretation.

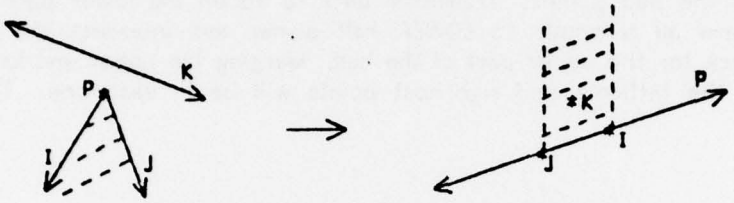


Figure 4: Transform of a redundant half plane.

In Figure 4, line k is above point P in the xy plane. This is transformed to a *point* k which is above *line* P in the ab plane. (Above/below-ness between lines and points is preserved by the transform.) The slope of a line in the xy plane is the a coordinate of the corresponding point in the ab plane. Thus, as in Figure 4, line k with a slope between the slopes of lines i and j transforms to a point k with a coordinate between the a coordinates of points i and j in the ab plane. Figure 5 shows the result of applying this procedure to several half planes. In the ab plane the points connected by line segments are the transforms of the "nonredundant" half planes. The points above these segments correspond to the redundant half planes.

Figure 5: Transform of N UPPER half planes.

Look closely at those line segments in the ab plane. They are exactly the bottom part of the *convex hull* of the points in the ab plane! The problem of intersecting N UPPER half planes has been reduced to the problem of constructing the (bottom part of the) convex hull of N points. The convex hull of N points in the plane can be constructed in O(NlogN) time (Graham[72]). This leaves only the detail of separating the top from the bottom part of the hull. To do that, find the leftmost point of the hull in O(N) time. Then traverse the hull on the *bottom* side until the rightmost point is reached.

The problem of intersecting N half planes has been broken into three parts. Part (1), the intersection of the UPPER half planes, has just been shown to cost only O(NlogN) time. Part (2), the intersection of the LOWER half planes, is equivalent to part (1) so it can also be done in O(NlogN) time. Part (3), the intersection of the results of parts (1) and (2), has been shown to cost only O(N) time. Thus, the entire problem can be solved in O(NlogN) time.

It is interesting to note that one can also use an intersection of half planes algorithm to produce a convex hull of points algorithm. First transform all N points to UPPER half planes by the formula

$$(x,y) \rightarrow b = x*a + y$$

and intersect the half planes. Transform back to obtain the lower part of the hull. Then transform all N points to LOWER half planes and intersect the half planes. Transform back for the upper part of the hull. Merging the upper and lower parts is trivial, since the leftmost and rightmost points will be in each one. Total time is O(NlogN).

5. The three dimensional problem

The technique just used in two dimensions can be extended to three dimensions. The half spaces are, as before, first partitioned into the two sets UPPER and LOWER. A half space is in the UPPER set if the plane at its boundary is above the rest of the half space. Similarly for LOWER. The problem of intersecting the half spaces is then broken into the subproblems (1) intersect the UPPER half spaces, (2) intersect the LOWER half spaces, and (3) intersect the results of (1) and (2). This schema will still work in three dimensions, but the algorithm for part (3) is more complicated and does not seem to generalize well to higher dimensions. To avoid these complications a slightly different method for intersecting half spaces in three dimensions will be presented.

Recall that the partition of the half spaces into the sets UPPER and LOWER is determined by a comparison in the z coordinate. The choice of the z coordinate is totally arbitrary. Any of the 3 coordinates will do just as well.[1] Thus, there are 3 possible sets UPPER and LOWER. To distinguish them from each other, use the notation $UPPER_1$ and $LOWER_1$ for the first (x) coordinate, $UPPER_2$ and $LOWER_2$ for the second (y) coordinate, and $UPPER_3$ and $LOWER_3$ for the third (z) coordinate. Also, U and L, the intersections of the UPPER and LOWER sets, will now be written U_i and L_i for the ith coordinate. The new method for solving the intersection of U and L (that is, U_3 and L_3), uses all 3 of the pairs U_i and L_i. (See Figure 6 for an illustration of the three sets U and L for an octahedron.) It will be shown how the extra U_i and L_i make the intersection much simpler. Then it will be shown how to construct U_3, L_3, U_2, L_2, U_1, and L_1.

[1] In fact, so will any linear combination of the coordinates.

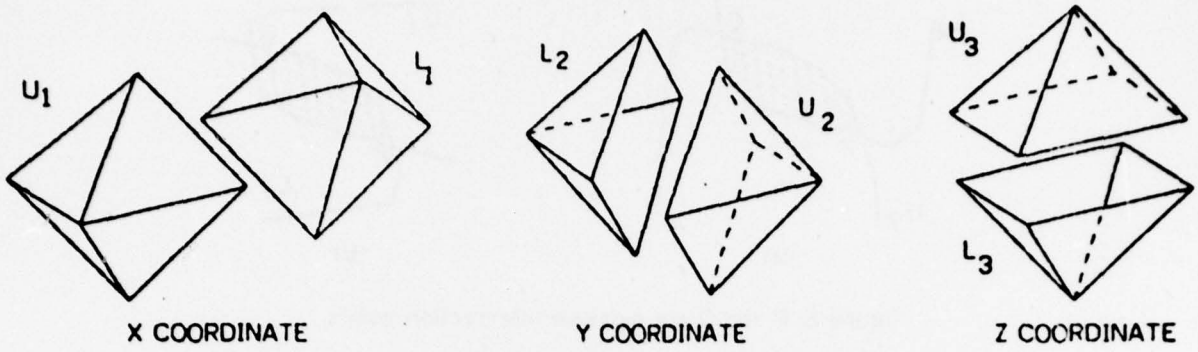


Figure 6: The three sets U and L for an octahedron.

An example of the two dimensional case is shown in Figure 7. The curves U_2 and L_2 of Figure 7a meet at two points, P and Q. Once P and Q are found, it is easy to construct the entire polygon defining the intersection of *regions* U_2 and L_2. (See the last part of INTERSECTCHAINS in Section 2.) The leftmost intersection point P of U_2 and L_2 in 7(a) is simply the leftmost point of L_1 in 7(b). Thus, rather than using algorithm INTERSECTCHAINS (Section 2) to find point P, one can simply construct L_1 and find the leftmost point. (Similarly for point Q.) Does this always work? No, but the exceptions are not a serious problem.

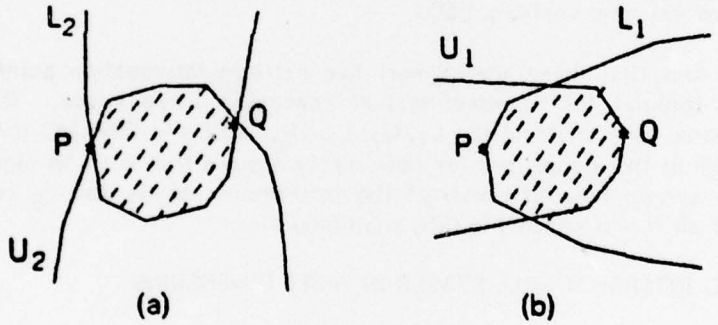


Figure 7: An easy way to find P and Q.

Suppose that point P is not only the leftmost point of the shaded region, but also the highest (or lowest) point (Figure 8). Then P will be one of the two intersection points for not only U_2 and L_2, but also for U_1 and L_1. Such points will be called *extreme intersection points*. (The cases of unbounded or null intersections will be treated later.) In fact, as shown in Figure 8, P and Q can be extreme intersection points simultaneously. Thus, in the two dimensional worst case, the use of U_1 and L_1 to find P and Q does not seem entirely successful, since the two extreme intersection points may still be left unknown. However, in higher dimensions the situation looks relatively better.

Figure 8: P and Q are extreme intersection points.

Thm: In K dimensions there are at most two extreme intersection points of N half spaces.

Pf: To be an extreme intersection point, the vertex must be extreme in all K coordinates. (That is, greatest or smallest of all the vertices in each coordinate.) It is assumed that no vertices have any coordinates in common, since the probability of that happening is zero. Assume that P, Q, and R are distinct extreme intersection points. Since P and Q are extreme intersections, then if the ith coordinate of P is the smallest (largest) over the vertices, then the ith coordinate of Q is the largest (smallest). However, the ith coordinate of R must also be the largest or smallest i coordinate and that can't be because points P and Q exhaust both of those possibilities. Thus there are at most two extreme vertices. QED.

The fact that there are at most two extreme intersection points is used in the algorithm (below) for intersecting half spaces in three space. This algorithm assumes that one can construct U_1, L_1, U_2, L_2, U_3, and L_3 in O(NlogN) time. That will be proven later in this paper, but for now simply assume that such an algorithm exists and it will be shown how to construct the intersection of U_3 and L_3 (and thus the intersection of all N half spaces) in O(N) additional time.

ALGORITHM TO INTERSECT HALF SPACES IN THREE DIMENSIONS

Input: Integer N > 0, real vectors X[1:N], Y[1:N], Z[1:N].
Output: Integers K (number of vertices of the polytope), E (number of edges), F (number of faces), real vectors H_x[1:K], H_y[1:K], H_z[1:K], integer arrays EndPointsOfEdges[1:E,1:2], FacesOfEdges[1:E,1:2], EdgesOfFacesPointer[1:F], EdgesOfFaces[1:2*E].
Time: O(NlogN), Space: O(N).

(1) Construct U_3, L_3, U_2, L_2, U_1, and L_1 in O(NlogN) time. (Section 6 of this paper.) (These structures are represented the same as the output for this algorithm.)

(2) For the most part, the edges where U_3 and L_3 meet have already been constructed as edges of U_2 and L_2 (and U_1 and L_1). For each of the O(N) faces of U_2 and L_2 one can determine in O(1) time whether it belongs in $UPPER_3$ or $LOWER_3$. Thus,

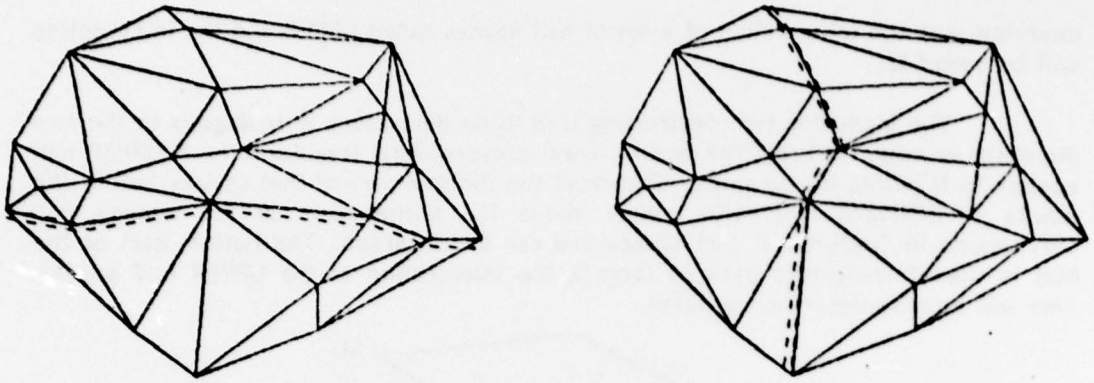


Figure 9: Dashes indicate boundaries of (a) U_3 and L_3 and (b) U_2 and L_2

in O(N) time one can search all of the faces of U_2 and L_2 and determine where the $UPPER_3$ faces of U_2 and L_2 meet the $LOWER_3$ faces. The edges where these faces meet are edges of the intersection of U_3 and L_3. (See Figure 9.)

(3) Compare U_3 and L_3 with U_1 and L_1 as in (2) in O(N) time. By the above theorem, there will be at most two vertices of $U_3 \cap L_3$ which have not yet been determined in steps (2) and (3).

(4) If the chain of edges connecting U_3 and L_3 (constructed in steps (2) and (3)) is a closed loop then there are no extreme vertices and the intersection of U_3 and L_3 is complete.

(5) If the chain of edges connecting U_3 and L_3 is open at one end then either (a) the intersection of U_3 and L_3 is an unbounded region, or (b) there is one extreme intersection. If the chain is broken into two parts then either (c) there are two extreme intersections, or (d) there is one extreme intersection and the intersection of U_3 and L_3 is unbounded. For these cases of one or two breaks in the chain, apply step (6) to find any remaining vertices.

(6) To distinguish an extreme intersection point from an unbounded intersection: (a) find the two edges at the two ends of the chain and call them A and B, (b) find the faces above and the faces below A and B which have A or B as edges. Call them A_u, A_l, B_u, and B_l. (c) Find the (infinite) faces at the intersection of A_u and A_l, and B_u and B_l. Call them A_I and B_I. (d) In O(N) time find the intersection, if any, of A_I and B_I.

It has just been shown that given U_3, L_3, U_2, L_2, U_1, and L_1, the intersection of U_3 and L_3 can be constructed in O(N) time. Now it must be shown how to construct U_3, L_3, U_2, L_2, U_1, and L_1 in O(NlogN) time. Since constructing one of these six intersections is essentially the same as constructing any other, the following text will

describe just the intersection of a set of half spaces called UPPER and the intersection will be called U.

The algorithm for constructing U in three dimensions is analogous to the two dimensional case. In brief, this is how it will proceed: First transform the N UPPER half spaces to N points in abc space. Construct the (bottom part of the) convex hull of the points in O(NlogN) time. The points above the bottom part of the convex hull correspond to "redundant" half spaces and can be discarded. The bottom part of the hull is then transformed back to form U, the intersection of the UPPER half spaces. This will now be described in detail.

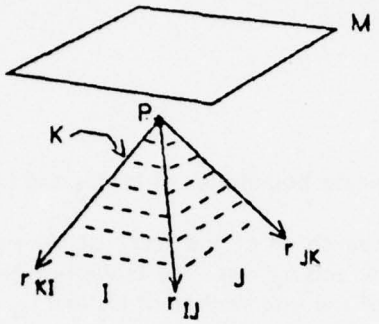


Figure 10: A redundant half space.

Assume that there are N UPPER half spaces. Some of these half spaces contribute to the intersection U and some are "redundant," such as half space M in Figure 10. For the plane there are two simple conditions for redundancy of a half plane. In three dimensions there are two analogous conditions for a half space. The first condition for redundancy of a half space M is that plane M be above the point P where planes I, J, and K intersect, as in Figure 10. The second condition, the "betweenness of slopes" condition, is more complicated to express. The purpose of the "betweenness of slopes" condition is to insure that a plane above the point P can't drop down fast enough to enter the region below planes I, J, and K. But this is insured by requiring that plane M remain above the rays r_{IJ}, r_{JK}, and r_{KI} which define the edges of the three faces bounding the enclosed region.

Plane M can be written in the form $z = a*x + b*y + c$. (No vertical planes satisfying the two conditions are possible, so this form will always work.) Plane I can be written $z = a_I*x + b_I*y + c_I$. Similarly for planes J and K. The three rays r_{IJ}, r_{JK}, and r_{KI} can be expressed as vectors originating at the point P. The cross products of normals to planes I, J, and K give vectors along the lines determined by these three vectors. Thus,

$$r_{IJ} = (\alpha_{IJ}, \beta_{IJ}, \gamma_{IJ}) = -\begin{vmatrix} \vec{I} & \vec{J} & \vec{K} \\ a_I & b_I & -1 \\ a_J & b_J & -1 \end{vmatrix} \times \text{SGN} \begin{vmatrix} a_I & b_I \\ a_J & b_J \end{vmatrix}$$

Similarly for r_{JK} and r_{KI}. The first condition, that plane M be above (or on) point $P = (P_x, P_y, P_z)$ is

(1) $$a*P_x + b*P_y + c \geq P_z.$$

The second (betweenness of slopes) condition depends only on the slopes a and b, not the intercept c. Thus, it is the same for a plane M which passes through point P as one which is strictly above P. The condition that M not drop below vectors r_{IJ}, r_{JK}, or r_{KI} is therefore

(2)
$$a * \alpha_{IJ} + b * \beta_{IJ} \geq \gamma_{IJ}$$
$$a * \alpha_{JK} + b * \beta_{JK} \geq \gamma_{JK}$$
$$a * \alpha_{KL} + b * \beta_{KL} \geq \gamma_{KL}.$$

The four inequalities in (1) and (2) have an equivalent interpretation in the transformed (abc) space.

6. Application of the transform to the three dimensional problem

The transform used in three dimensions is a straightforward extension of the two dimensional transform. Planes transform to points and points transform to planes. The formulas for the transform are:[1]

$$z = a*x + b*y + c \rightarrow (a,b,c)$$
$$(x,y,z) \rightarrow c = -x*a + -y*b + z.$$

[1] Dantzig[63] uses the above transform in the context of linear programming and Huffman[77] uses an almost identical transform for an analysis of polyhedral scenes.

The distance between a point and a plane *in the z coordinate* is preserved by this transform. And, most importantly, the sense of above/below-ness (in the z coordinate) between points and planes is preserved also.

Plane M ($z = a*x + b*y + c$) of Figure 10 transforms to the point (a,b,c) in abc space, and planes I, J, and K transform to the points $P_I = (a_I,b_I,c_I)$, $P_J = (a_J,b_J,c_J)$, and $P_K = (a_K,b_K,c_K)$. The inequality (1) (of the previous section) for redundancy of half space M (and point (a,b,c)) is interpreted in abc space as a half space whose boundary plane passes through all three points P_I, P_J, and P_K. Thus condition (1) requires that the point (a,b,c) be above this plane. Inequalities (2) (of the previous section) define three vertical planes, each of which passes through two of the three points P_I, P_J, and P_K. Thus, the set of half spaces M which half spaces I, J, and K make redundant is represented in abc space by the set of all points (a,b,c) which are *directly above* some point in (or on) the triangle determined by points P_I, P_J, and P_K. When conditions (1) and (2) *are applied over* all N points in abc space, the only points (half spaces) which are not made redundant by some points (half spaces) I, J, and K are those points on the bottom part of the convex hull of the N points.

How long does it take to construct the (bottom part of the) convex hull of N points in three dimensions? Preparata and Hong[77] describe an algorithm to construct the (entire) convex hull in O(NlogN) time. One way to separate the bottom of the hull from the top requires one to augment Preparata and Hong's algorithm by *maintaining* with each face of the convex hull a vector perpendicular to the face which points toward the inside (as opposed to the outside) of the hull. The bottom faces of the hull are those faces whose vectors point upward. The vectors for the top faces point downward. (Note that *there will be some vertices in both the top and bottom parts of* the hull. These are the vertices which bound both top and bottom faces.) The bottom and top parts of the convex hull are *therefore* separable in O(N) time.

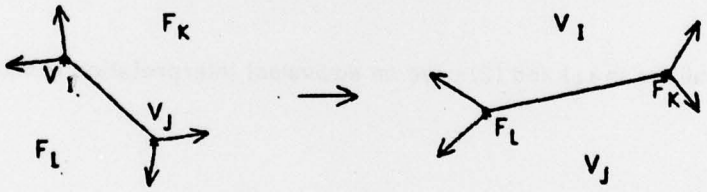


Figure 11: Transform of a convex hull.

To find the intersection of the UPPER half spaces, the bottom part of the convex hull must be transformed back to xyz space. The O(N) vertices transform, of course, to planes. But there's much more information than that in the convex hull. For instance, defining each face of the hull there are three coplanar vertices I, J, and K.[1] The plane which these vertices define is transformed to a point in xyz space. This point is where *planes* I, J, and K (of xyz space) intersect. (This follows from the fact that the transform preserves incidence between points and planes.) Also, as illustrated

[1] Assuming that the vertices are in general position. If not, then there may be four or more coplanar vertices.

in Figure 11, if faces F_K and F_L of the (bottom part of the) convex hull share an edge V_IV_J, then in xyz space *faces* V_I and V_J share an edge F_KF_L. In fact, even the unbounded faces of U in xyz space can be obtained from the transform. These faces of U correspond to vertices at the boundary between the top and bottom parts of the convex hull in abc space. Thus, little computation will be required to construct U after the transform from abc space since all the faces, vertices, edges, etc. are directly obtainable from the transform. Since the transform costs only O(N) time, the total time to construct U is dominated by the time to construct the convex hull in abc space, which is O(NlogN) time.

It has just been shown that the time to construct U, the intersection of the UPPER half spaces is O(NlogN). Thus, U_3, L_3, U_2, L_2, U_1, and L_1 can be constructed in O(NlogN) time. Once the U_i and L_i are constructed, it costs only O(N) more time to intersect U_3 and L_3. Thus, the total time to intersect N half spaces in 3 space is O(NlogN).

There is one more item of interest in the three dimensional problem. As with the two dimensional case, an intersection of half spaces algorithm can be used as a convex hull (of points) algorithm also. Simply transform all N points to UPPER (LOWER) half spaces, intersect the half spaces, and transform back to obtain the bottom (top) part of the convex hull. Merging the top and bottom parts of the hull is, again, trivial since they will share several vertices at the boundary of the top and bottom parts.

7. Intersecting half spaces in four or more dimensions

Suppose that in K dimensions the convex hull of N points can be constructed in H(N,K) time. Then the intersection of N half spaces can be constructed in O(K*H(N,K)) time. The problem is solved by a straightforward extension of the three dimensional solution. The U_i and L_i are first constructed in O(K*H(N,K)) time and then U_K and L_K are intersected with the aid of the other U_i and L_i as in the three dimensional case. To construct the U_i and L_i, one transforms the $UPPER_i$ ($LOWER_i$) half spaces to points, constructs the convex hull of the points in H(N,K) time, partitions the top from the bottom parts of the hull, and transforms the bottom (top) part of the hull back.

Before proving that this actually works in K dimensions some terminology will first have to be introduced. Let the UPPER set of half spaces, refer, by default, to the $UPPER_K$ set of half spaces. That is, "up" will be measured in the x_K coordinate. Also, let a "j-face" of a K dimensional polytope by denoted as follows: a vertex is a 0-face, an edge (a line segment) is a 1-face, etc.

Theorem: Let S be a set of N UPPER, K - dimensional half spaces H_i defined by

$$H_i: \; a_{ik} + \sum_{j=1,k-1} a_{ij}*x_j \le x_k, \; i=1,\ldots N$$

and let T be the convex polytope formed by the intersection of these N half spaces. If the transform of half space $H_i \in S$ is

$$H_i \rightarrow A_i = (a_{i1}, a_{i2}, \ldots a_{ik}), \; i=1,\ldots N,$$

then the nonredundant half spaces of S correspond to the points on the bottom half of the convex hull of the points A_i. Furthermore, the j-faces of the convex hull correspond to the K-j-1 faces of T.

Proof: The proof is divided into five parts:

(1) For each redundant half space H_t there is a set $G_t \subset (S-H_t)$ of half spaces such that [INTERSECT($x \in G_t$) x] $\subset H_t$ and $|G_t| = K$ (if the half spaces are in general position).

(2) The algebraic description of the redundancy of H_t with respect to G_t.

(3) The interpretation of the algebraic description in the transformed space.

(4) The interpretation in (3) implies that the nonredundant half spaces correspond to the points on the bottom part of the convex hull of the points A_i.

(5) The j-faces of the convex hull correspond to the K-j-1 faces of T.

PART (1): T is a convex polytope because it is an intersection of half spaces. If a halfspace H_t is redundant then its boundary plane lies completely above T. Let V_t be the point of T which is closest to the boundary plane of H_t. (If the closest point to the boundary of H_t is not unique, then let V_t be any vertex in the set of closest points.) Since T is a convex polytope, V_t is (or can be chosen to be) a vertex of T. Let $G_t \subset S$ be the set of half spaces of S whose boundaries meet at vertex V_t. If the half spaces of S are in general position then the set G_t will contain exactly K half spaces. Since T is convex, one can travel from V_t along T in any direction and the distance to the boundary of H_t will be nondecreasing. It follows that [INTERSECT($x \in G_t$) x] $\subset H_t$. That is, H_t is redundant with respect to the half spaces in G_t. As in two and three dimensions, redundancy can be expressed by two sets of conditions: (1) the boundary of H_t is above vertex V_t, and (2) the "betweenness of slopes" condition. These conditions will now be shown algebraically.

PART (2): Let H_t be the half space

$$a_{tk} + \sum_{j=1,K-1} a_{tj} * x_j \leq x_k,$$

and let the half spaces in G_t be, without loss in generality, the half spaces 1 through K:

$$a_{ik} + \sum_{j=1,K-1} a_{ij} * x_j \leq x_k, \; i=1,\ldots K.$$

The point $V_t = (V_{t1}, V_{t2}, \ldots V_{tk})$ closest to the boundary plane of H_t is defined as the solution to the set of equations

$$a_{ik} + \sum_{j=1,K-1} a_{ij} * V_{tj} = V_{tk},\ i=1,\ldots K.$$

Letting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots\cdots & a_{1,k-1} & -1 \\ a_{21} & a_{22} & \cdots\cdots & a_{2,k-1} & -1 \\ & & \cdots\cdots & & \\ a_{k1} & a_{k2} & \cdots\cdots & a_{k,k-1} & -1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} a_{1k} \\ a_{2k} \\ \vdots \\ a_{kk} \end{pmatrix}$$

this can be rewritten

$$A * V_t' = B,$$

where V_t' is the transpose of V_t.

Now the two conditions for redundancy of H_t can be described. The condition that the boundary of H_t is above point V_t is

(1) $$a_{tk} + \sum_{j=1,K-1} a_{tj} * V_{tj} \geq V_{tk}.$$

The "betweenness of slopes" conditions are

(2) $$\sum_{j=1,K} M_{ij} * C_j \geq 0,\ i=1,\ldots K,$$

where $M_{ij} = (-1)^K * cof_{ij}(A) * SGN(cof_{ik}(A))$ and
$C_j = a_{tj}$ for $j \leq K-1$ and $C_k = -1$.

Equation (1) is straightforward and believable, but (2) requires some explanation. Describing (2), the "betweenness of slopes" condition, is difficult to do by directly comparing slopes of the K flats of G_t. Instead, the alternative developed for the three dimensional case will now be applied to the K dimensional problem. In three dimensions one constructs the three rays r_{ij}, r_{ik}, and r_{jk} determined by the vertex P where the three planes meet and the intersection of pairs of planes I and J, I and K, and J and K, respectively. Condition (2) is satisfied if the redundant plane stays above these rays. In K dimensions one constructs the K rays determined by the vertex V_t and the K sets of K-1 elements of G_t. The notation used for the K dimensional case will be slightly different than for the three dimensional case. Rather than denoting a ray by the set of K-1 flats which determine it, the ray will instead be denoted by the one flat which is *not* used to determine it. If $(\alpha_{i1}, \alpha_{i2}, \ldots \alpha_{ik})$ is a vector parallel to

ray i (the ray which does *not* include flat i), then ray i may be written in parametric form as

$$(x_1, x_2, \ldots x_k) = (V_{i1}, V_{i2}, \ldots V_{ik}) + u * (\alpha_{i1}, \alpha_{i2}, \ldots \alpha_{ik}).$$

Now it must be shown how to construct such a vector $(\alpha_{i1}, \alpha_{i2}, \ldots \alpha_{ik})$. Ray i is also the intersection of the other K-1 half spaces of G_t. Thus,

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1,k-1} & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i-1,1} & a_{i-1,2} & \cdots & a_{i-1,k-1} & -1 \\ a_{i+1,1} & a_{i+1,2} & \cdots & a_{i+1,k-1} & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{k1} & a_{k2} & \cdots & a_{k,k-1} & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} a_{1k} \\ \vdots \\ a_{i-1,k} \\ a_{i+1,k} \\ \vdots \\ a_{kk} \end{pmatrix}.$$

By combining these two equations one obtains

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1,k-1} & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i-1,1} & a_{i-1,2} & \cdots & a_{i-1,k-1} & -1 \\ a_{i+1,1} & a_{i+1,2} & \cdots & a_{i+1,k-1} & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{k1} & a_{k2} & \cdots & a_{k,k-1} & -1 \end{pmatrix} \begin{pmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \alpha_{ik} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

The problem is now to find a general solution to this system of K-1 equations in K unknowns. Recall the following property of cofactors:

$$\sum_{j=1,K} a_{ij} * \mathrm{cof}_{mj}(A) = \det(A) \text{ if } i=m$$
$$= 0 \text{ if } i \neq m.$$

It follows that the general solution is

$$(\alpha_{i1}, \alpha_{i2}, \ldots \alpha_{ik}) = \text{constant} * (\mathrm{cof}_{i1}(A), \mathrm{cof}_{i2}(A), \ldots \mathrm{cof}_{ik}(A)).$$

The constant is chosen as $(-1)^K * SGN(cof_{ik}(A))$ so that the vectors always point downward. It can now be seen that the inequality expressed in condition (2) is simply a generalization of the formulas for the three dimensional case.

PART (3): In the transformed space, half space H_t becomes a point $H_t = (a_{t1}, a_{t2}, \ldots a_{tk})$ and the elements of G_t transform to points similarly. Vertex V_t transforms to a flat V_t defined by

$$V_{tk} - \sum_{j=1,K-1} V_{tj} * z_j = z_k.$$

Since incidence is preserved by the transform, flat V_t passes through all of the points G_t and is, in fact, determined by them. Also, since above/below-ness in the x_k coordinate is preserved, point H_t is above flat V_t in the transform space. Using the formula for flat V_t, this can be expressed as

$$V_{tk} - \sum_{j=1,K-1} V_{tj} * a_{tj} \leq a_{tk},$$

which is equivalent to condition (1) for redundancy of H_t! Condition (2) defines K "vertical" half spaces which further constrain point H_t to be not only above flat V_t but also directly above some point in the (K-1 dimensional) *simplex* of the points G_t. (The K half spaces are described as "vertical" because their formulas do not include the term a_{tk} and thus the K coordinate is free.)

PART (4): In Parts (1) through (3) attention has focused only on the conditions for redundancy of a particular half space H_t. Given the interpretation of part (3) above, what can now be said over all N of the half spaces of S? It has been shown that if half space H_t is redundant with respect to a set of K half spaces G_t, then in the transform space point H_t is directly above some point of the (K-1 dimensional) simplex of the points G_t, and conversely. Thus, a half space $H_i \in S$ is redundant iff there exists a set $G_i \subset S-H_i$ of K half spaces such that in the transform space *point* H_i is directly above a point of the (K-1 dimensional) simplex of the *points* of G_i! But this eliminates all points except those on the bottom of the *convex hull* of the N points H_i. This proves the main part of the theorem. Now it must be shown that the convex hull produces not only the nonredundant half spaces but also the edges, faces, etc. of T also.

PART (5): To show that a j-face of the bottom part of the convex hull of the points H_i corresponds to a K-j-1 face of T, a generalized version of the transform must first be described. The general transform maps a j-space to a K-j-1 space.

> General Transform: The transform of a j dimensional subspace of K space is the set of all flats which contain it. This is a K-j-1 dimensional subspace of flats. However, since each flat can be readily represented as a point, the transform of the j dimensional subspace is represented as a K-j-1 dimensional subspace of points.

Theorem: The general transform preserves incidence. In other words, if a j_1 dimensional subspace of K space is a subspace of a j_2 dimensional subspace, then the

transform of the j_2 dimensional subspace is a subspace of the transform of the j_1 dimensional subspace.

Proof: Since the j_2 dimensional subspace can be interpreted as an intersection of $K-j_2$ flats, it will be sufficient to show that if a K-j dimensional subspace is a subspace of a flat, then the transform of the flat is a (zero dimensional) subspace of the transform of the K-j-space. Let the K-j-space be represented as an intersection of j flats

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1,k-1} & -1 \\ a_{21} & a_{22} & \cdots & a_{2,k-1} & -1 \\ & \cdots & \cdots & \cdots & \\ a_{j1} & a_{j2} & \cdots & a_{j,k-1} & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = - \begin{pmatrix} a_{1k} \\ a_{2k} \\ \vdots \\ a_{jk} \end{pmatrix}$$

The transform of the K-j space is a j-1 space with points

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{j1} \\ a_{12} & a_{22} & \cdots & a_{j2} \\ & \cdots & \cdots & \\ a_{1k} & a_{2k} & \cdots & a_{jk} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{j-1} \\ 1 - U_1 - U_2 - \cdots U_{j-1} \end{pmatrix}$$

It is easy to see that this j-1 space contains the points

$$(a_{i1}, a_{i2}, \cdots a_{i,k-1}, a_{ik}),\ i=1,\ldots j$$

which are the transforms of the j flats.

Since the general transform preserves incidence and j-spaces transform to K-j-1 spaces, it follows that j-faces of the convex hull transform to K-j-1 faces of T. QED.

As with the two and three dimensional cases, an algorithm for intersecting N half spaces in K dimensions can be used to construct the convex hull of N points in K

dimensions. The procedure is a straightforward extension of the three dimensional case: Transform all N points to N UPPER (LOWER) half spaces in K space. Intersect the half spaces, then transform back to the LOWER (UPPER) part of the convex hull. Joining the UPPER and LOWER parts of the hull should be easy because of the vertices which they have in common on their boundaries.

8. Fast expected time algorithms

There have been recent advances in algorithms for constructing the convex hull of a set of points in fast *expected* time. Since the time to intersect N half spaces is dominated by the time to construct the convex hull of N *points*, these fast convex hull algorithms lead to fast intersection of half space algorithms. Bentley and Shamos[78] have shown that when the expected number of points on the convex hull is $O(N^p)$ for some $p < 1$, the convex hull of N points in two and three dimensions can be constructed in O(N) expected time, while maintaining an O(NlogN) worst case time. Thus, N half planes or N half 3-spaces (under the above conditions) can be intersected in O(N) expected time and O(NlogN) worst case time.[1] In four or more dimensions the worst case time is at least $\Omega(N^2)$. However, it is anticipated that one may still construct fast expected time algorithms.

9. Applications

(A) Linear programming.

The feasible region of a linear programming problem in K variables and N constraints is an intersection of N K-dimensional half spaces. For large K, the time O(K*H(N,K)) to construct this may not be an improvement over the simplex method. But for $K \leq 3$, the feasible region can be constructed in O(NlogN) time worst case and (under the conditions described in Section 8) O(N) expected time. Thus, the total time to solve the linear programming problem is only O(NlogN) worst case and O(N) expected time. Shamos and Hoey[76] have previously shown that a linear programming problem in two variables (two dimensions) can be solved in O(NlogN) time. Bentley and Shamos[78] later showed the O(N) expected time for the two dimensional case.

For $K > 3$ dimensions a different approach is recommended. To solve a linear programming problem it is not really necessary to construct the entire feasible

[1] Bentley and Shamos[78] showed this result for the two dimensional case using a duality result of Ziezold[70].

region. Certainly the simplex method avoids doing any such thing. Dantzig[63] applies the duality transform to a standard linear programming problem of K equations in N variables. (The interpretation of a linear programming problem as an intersection of N half spaces in K dimensions is shown equivalent to this standard form by the duality theorem of linear programming.) The objective function is transformed to a vertical line in K space and the N variables are each transformed to points in K space. The optimization problem is transformed to the problem of determining where the vertical line intersects the convex hull of the N points. The simplex method provides one way to determine this intersection without constructing the convex hull of the N points. It remains to be determined whether or not modern algorithm techniques can produce an improvement over the simplex algorithm in the K dimensional case.

(B) Intersection of convex polyhedra

A convex polyhedron of N faces can be described as an intersection of the N half spaces which determine the faces (and which include the polyhedron). Thus, an intersection of two convex polyhedra of N faces can be solved as an intersection of 2*N half spaces, which can be done in O(NlogN) time. (See Muller and Preparata[77] for a different O(NlogN) time algorithm for intersection of convex polyhedra.) But it remains an open question whether or not the algorithm can be improved to run in O(N) time. (It certainly can in two dimensions.)

10. Conclusion

The use of a geometric transform has been shown to be very useful for construction of a fast algorithm for intersection of half spaces. This algorithm, in turn, has been used in algorithms for a fast *expected* time algorithm for intersection of half spaces, intersection of convex polyhedra, and linear programming in three variables. The geometric transform is applicable to many more problems. The union of a set of (the interiors of) N arbitrary planar circles can be constructed by transforming to a set of N half 3 - spaces which are then intersected in O(NlogN) time. The Euclidean diameter of a set of N points in 3 - space can be determined in O(NlogN) time through use of a transform very similar to the one in this paper. The nearest and farthest point Voronoi diagrams of N points on a sphere or on a Euclidean plane can be constructed in O(NlogN) time through use of inversion and the methods in this paper. Also, several two dimensional problems with lines have been solved quickly through use of this transform (Brown[77]). In addition to the duality transform described in this paper, other transforms such as inversion, rotation, and various kinds of projection (gnomonic, orthographic, stereographic) have been successfully used to construct fast geometric algorithms. It is anticipated that the use of geometric transforms will become a standard tool in the construction of geometric algorithms.

Acknowledgements

Michael Shamos introduced the author to the problem of intersecting N half spaces in less than $O(N^2)$ time. Also, he and Jon Bentley made several constructive comments on early versions of the paper which improved its readability.

REFERENCES

Bentley, J.L. and Shamos, M.I., 1978, *Divide and Conquer for Linear Expected Time*, Information Processing Letters, Feb. 1978.

Brown, K.Q., 1977, *A Simple Transform for Fast Geometric Algorithms*, unpublished notes.

Chand, D.R. and Kapur, S.S., 1970, *An algorithm for convex polytopes.* JACM 17,7 Jan, p. 78-86.

Dantzig, G.B., 1963, *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey.

Eddy, W., 1977, *A New Convex Hull Algorithm for Planar Sets.* ACM Transactions on Mathematical Software, Vol. 3, No. 4, Dec., p.398-403.

Graham, R.L., 1972, *An efficient algorithm for determining the convex hull of a planar set.*, Information Processing Letters, No. 1, p.132-133.

Grunbaum, B., 1967, *Convex Polytopes*, Wiley Interscience, New York.

Harary, F., 1969, *Graph Theory*, Addison - Wesley Publishing Company.

Huffman, D.A., 1977, *A Duality Concept for the Analysis of Polyhedral Scenes*, p.475-492, Machine Intelligence 8, Elcock, E.W. and Michie, D. editors, Edinburgh University Press and Halsted Press (a subsidiary of John Wiley & Sons, Inc.)

Lee, D.T. and Preparata, F.P., 1976, *Location of a Point in a Planar Subdivision and Its Applications*, 8th SIGACT, p.231-235. Also in SIAM Journal of Computing, Sept. 1977, Vol.6, No.3, p.594-606.

Lipton, R.J. and Tarjan, R.E., 1977, *Applications of a Separator Theorem for Planar Graphs*, 18th Annual Symposium on Foundations of Computer Science, Oct.31-Nov.2.

Muller, D.E., and Preparata, F.P., 1977, *Finding the Intersection of Two Convex Polyhedra*, Report R-793 UILU-ENG 77-2240, Coordinated Science Laboratory, Applied Computation Theory Group, University of Illinois, Urbana, Illinois. Also submitted to Theoretical Computer Science.

Preparata, F.P. and Hong, S.J., 1977, *Convex hulls of finite sets of points in two and three dimensions*, CACM, 20,2, Feb., p.87-93

Preparata, F.P., and Muller, D.E., 1977, *Finding the Intersection of a Set of n Half-Spaces in Time O(nlogn)*, Report R-803, UILU-ENG 77-2250, Coordinated Science Laboratory, Applied Computation Theory Group, University of Illinois, Urbana, Illinois.

Shamos, M.I., 1975, *Geometric Complexity*, Proc. Seventh Annual ACM Symposium on Automata and Computability Theory, p.224-233.

Shamos, M.I, and Hoey, D., 1975, *Closest-Point problems.* Proc. Sixteenth Annual IEEE Symposium on Foundations of Computing, October, 1975, p. 151-162.

Shamos, M.I., and Hoey, D., 1976, *Geometric Intersection Problems*, Proc. Seventeenth Annual IEEE Symposium on Foundations of Computer Science, October, 1976, p. 208-215.

Shamos, M.I., 1977, *Problems in Computational Geometry*, unpublished notes. To appear as Computational Geometry, Springer-Verlag, 1977.

Ziezold, H., 1970, *Uber die Eckenzahl zufalliger konvexer Polygone.* Izv. Akad. Nauk. Armjan. SSR. Ser. Mat. Vol. 5, p. 296-312.

Zolnowsky, J., 1977, *Topics in Computational Geometry*, Stanford Ph.D. thesis, August.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CMU-CS-78-129	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): FAST INTERSECTION OF HALF SPACES	5. TYPE OF REPORT & PERIOD COVERED: Interim rept.
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): Kevin Q. Brown	8. CONTRACT OR GRANT NUMBER(s): N00014-76-C-0829
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 26p.
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Arlington, VA 22217	12. REPORT DATE: June 1978
	13. NUMBER OF PAGES: 25
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): same as above	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The problem of intersecting N half-spaces in K space is transformed to 2*K problems of constructing the convex hull of N points in K space and a simple intersection problem. This enables one to intersect the N K-dimensional half spaces in O(K*H(N,K)) time, where H(N,K) is the time required to construct the convex hull of N points in K space. For two and three dimensions the algorithm takes O(NlogN) time in the worst case, but under fairly robust conditions the expected time is only O(N). It is

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601 |

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

also shown that an algorithm for intersection of half spaces can be used to construct the convex hull of points in K space. Thus, the intersection of half spaces and convex hull of points problems are essentially equivalent.