4. Bobrow, D.G., Burchfiel, J.D., Murphy, D.L., and Tomlinson, R.S. TENEX, a paged time sharing system for the PDP-10. *Comm. ACM 15*, 3 (March 1972), 135–143.
5. Bobrow, D.G., and Murphy, D.L. The structure of a LISP system using two level storage. *Comm. ACM 10*, 3 (March 1967). 155–159.
6. Cheney, C.J. A nonrecursive list compacting algorithm. *Comm. ACM 13*, 11 (Nov. 1970), 677–678.
7. Deutsch, L.P. An interactive program verifier. Ph.D. Th., Comptr. Sci. Dep., U. of California, Berkeley, Calif., May 1973.
8. Deutsch, L.P. A LISP machine with very compact programs. Third Int. Joint Conf. on Artificial Intelligence, Stanford, Calif., 1973, pp. 697–703.
9. Fenichel, R.R., and Yochelson, J.C. A LISP garbage-collector for virtual-memory computer systems. *Comm. ACM 12*, 11 (Nov. 1969), 611–612.
10. Hansen, W.J. Compact list representation: Definition, garbage collection, and system implementation. *Comm. ACM 12*, 9 (Sept. 1969), 499-507.
11. Hehner, E.C.R. Matching program and data representations to a computing environment. Ph.D. Th., Comptr. Systems Res. Group, U. of Toronto, Toronto, Canada, Nov. 1974.
12. McCarthy, J., et al. *LISP 1.5 Programmer's Manual.* M.I.T. Press, Cambridge, Mass., 1962.
13. Minsky, M.L. A LISP garbage collector algorithm using serial secondary storage. Artificial Intelligence Proj. Memo 58 (rev.), Project MAC, M.I.T., Cambridge, Mass., Dec. 1963.
14. Quam, L.H. Stanford LISP 1.6 manual. Artificial Intelligence Proj., Stanford University, Stanford, Calif., Sept. 1969.
15. Reboh, R., and Sacerdoti, E. A preliminary QLISP manual. Tech. Note 81, Stanford Res. Inst. AI Center, Menlo Park, Calif., Aug. 1973.
16. Sacerdoti, E. The nonlinear nature of plans. Fourth Int. Joint Conf. on Artificial Intelligence, Tbilisi, Georgia, U.S.S.R., 1975, pp. 206–214.
17. Shannon, C.E. A mathematic theory of communication. *Bell System Tech. J. 27* (July 1948), 379–423.
18. Smith, D.H., Masinter, L.M., and Sridharan, N.S. Heuristic DENDRAL: Analysis of molecular structure. In *Computer Representation and Manipulation of Chemical Information*, W.T. Wipke, S. Heller, R. Feldman, and E. Hyde, Eds., Wiley, New York, 1974.
19. Teitelman, W. INTERLISP Reference manual. Xerox Palo Alto Res. Center, Palo Alto, Calif., 1974.
20. Van der Poel, W.L. A Manual of HISP for the PDP-9. Technical U., Delft, Netherlands.
21. Weissman, C. *LISP 1.5 Primer.* Dickenson Pub. Co., Belmont, Calif., 1967.
22. Wilner, W.T. Design of the B1700. Proc. AFIPS 1972 FJCC, Vol. 41, AFIPS Press, Montvale, N.J., pp. 579–586.
23. Zipf, G.K. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, Reading, Mass., 1949.

Programming Techniques

G. Manacher S.L. Graham
Editors

Convex Hulls of Finite Sets of Points in Two and Three Dimensions

F. P. Preparata and S. J. Hong
University of Illinois at Urbana-Champaign

The convex hulls of sets of n points in two and three dimensions can be determined with O(n log n) operations. The presented algorithms use the "divide and conquer" technique and recursively apply a merge procedure for two nonintersecting convex hulls. Since any convex hull algorithm requires at least O(n log n) operations, the time complexity of the proposed algorithms is optimal within a multiplicative constant.

Key Words and Phrases: computational complexity, convex hull, optimal algorithms, planar set of points, spatial set of points

CR Categories: 4.49, 5.25, 5.32

1. Introduction

The determination of the convex hull of a finite set of points is relevant to several problems in computer graphics, design automation, pattern recognition and operations research: references [3, 4, 10]—just to cite a few—discuss some interesting applications in these areas, which require convex hull computation.

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DAAB-07-72-C-0259.

Authors' addresses: F.P. Preparata, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801; S.J. Hong, IBM Systems Product Division, Poughkeepsie, NY 12602. This work was completed while the second author was on leave at the University of Illinois.

We also mention that other important questions, such as set separability or existence of linear decision rules, are easily solved through the determination of convex hulls.

This problem has received some attention in recent times. Chand and Kapur [2] described a convex hull algorithm for a finite set of n points in a space with an arbitrary number of dimensions. Their approach is based on the so-called "gift wrapping" principle and requires a number of operations $O(n^2)$. The first convex hull algorithm to run in time less than $O(n^2)$ was found by R.L. Graham [5] for a set of points in the plane. Graham's method, based on representing the points in polar coordinates and sorting them according to their azimuth, has a running time $O(n \log n + Cn)$, for some constant C determined by the Cartesian-to-polar coordinate conversion. More recently, R.A. Jarvis [7] presented an alternative algorithm for the planar case, which is the two-dimensional specialization of the Chand-Kapur method and runs in time $O(nm)$, where m is the number of points on the hull. Subsequent to the original submission of this paper, we learned of the enlightening doctoral thesis of M.I. Shamos, which contains several convex hull algorithms with running time $O(n \log n)$ for a set of n points in the plane ([9], problems P3, P15, POL5c).

A problem of long standing has been the computation of convex hulls in more than two dimensions in time less than $O(n^2)$. In this paper we show that the convex hull of a finite set of n points in three dimensions can be computed with at most $O(n \log n)$ operations. We also report an algorithm for the plane, which is original and can be viewed as the two-dimensional specialization of the three-dimensional algorithm. The latter is based on the property that the number of edges of the convex hull of n points is at most linear in n. For this reason, its generalization based on edges is impossible beyond three dimensions, since in more than three dimensions it is known that there exist convex polytopes with n vertices whose numbers of edges are $O(n^2)$ (see [6, p. 193]).

The computation model we shall refer to is that of a random access machine (RAM) in the sense of Aho, Hopcroft, and Ullman ([1, pp. 5–24]), with the only modification that real number arithmetic replaces integer arithmetic. Our algorithms are based on the well-known technique called "divide and conquer." Let E^d denote the d-dimensional Euclidean space and let the set $S = \{a_1, \cdots, a_n \mid a_j \in E^d\}$ be given. Without loss of generality, the points of S are assumed to be given in Cartesian coordinates: in fact, in the adopted computation model, a conversion between coordinate systems can be done in a constant time per point. By $x_i(a)$ we denote the ith coordinate of $a \in E^d$, for $i = 1, \cdots, d$. Here and hereafter we assume that for any two points u and v in E^d we have $x_i(u) \neq x_i(v)$, for $i = 1, \cdots, d$. This simplification helps bring out the basic ideas of the algorithms to be described, while the modifications required for the unrestricted case are straightforward.

As a preliminary step we sort the elements of S according to the coordinate x_1, and relabel them if necessary so that we may assume $x_1(a_i) < x_1(a_j) \Leftrightarrow i < j$. We can now give the following general algorithm for a set of n d-dimensional points.

Algorithm CH

```
Input. A set $S = \{a_1, \ldots, a_n\}$, where $a_i \in E^d$ and $x_1(a_i) < x_1(a_j) \Leftrightarrow i < j$ for $i, j = 1, \ldots, n$.
Output. The convex hull $CH(S)$ of $S$.
Step 1. Subdivide $S$ into $S_1 = \{a_1, \ldots, a_{\lfloor \frac{1}{2}n \rfloor}\}$ and $S_2 = \{a_{\lfloor \frac{1}{2}n \rfloor + 1}, \ldots, a_n\}$.
Step 2. Apply recursively Algorithm CH to $S_1$ and $S_2$ to obtain $CH(S_1)$ and $CH(S_2)$.
Step 3. Apply a merge algorithm to $CH(S_1)$ and $CH(S_2)$ to obtain $CH(S)$ and halt.
```

The initial sorting of the x_1 coordinates of the elements of S requires $O(n \log n)$ operations. Notice that, because of this sorting and of step 1, the sets $CH(S_1)$ and $CH(S_2)$ will define two nonintersecting convex domains. Now, if the merging of two convex hulls with at most n d-dimensional extreme points in total requires at most $P_d(n)$ operations, an upper-bound to the number $C_d(n)$ of operations required by Algorithm CH is given by the equation

$$C_d(n) = 2C_d(\tfrac{1}{2}n) + P_d(n).$$

(Notice that we have assumed that n be even for simplicity, but practically without loss of generality). Thus, if we can show that $P_d(n)$ is $O(n)$, we shall obtain that $C_d(n)$ is $O(n \log n)$, and, taking into account the initial sorting pass, an overall complexity $O(n \log n)$ results for the convex hull determination.

In Sections 3 and 4 we shall show that merging algorithms with number of operations $O(n)$ can be designed for $d = 2, 3$. In the next section we shall establish a lower bound to the number of operations performed by any algorithm for finding the convex hull of a set of n points. Since this computational work is of at least the same order as that of an algorithm for sorting n numbers, i.e. it is $O(n \log n)$, we reach the interesting conclusion that the proposed convex hull algorithms for finite sets in two and three dimensions are optimal in their order of complexity, within a multiplicative constant.

2. Lower Bound

In this section we present a lower bound to the running time of any algorithm which computes the convex hull of a set of n points.

The arguments presented are similar to those developed in connection with finding the maxima of a set of vectors ([8, 11]), which is a problem related to the one being investigated. Incidentally, we note that

Fig. 1. Illustration for the proof of Lemma 3.

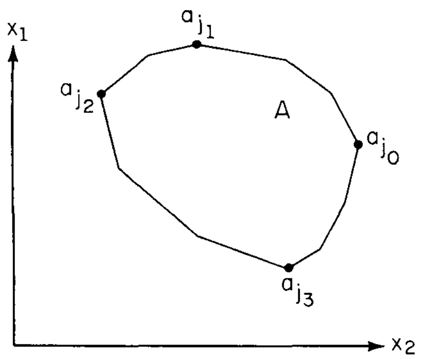
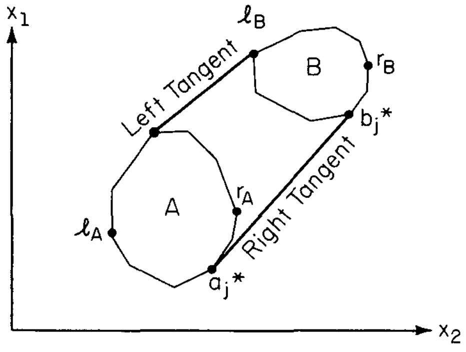


Fig. 2. Illustration of the planar merge procedure.



the same computation model was adopted in [8] and [11].

We begin with a straightforward result.

Lemma 1. $C_{d-1}(n) \leq C_d(n)$ *for* $d \geq 3$.

Proof. Let A_{d-1} be a set of n $(d-1)$-dimensional points for $d \geq 3$, and let A_d be the set of d-dimensional points obtained by extending each point $v \in A_{d-1}$ with the same component v_d. Let $CH(A_d)$ and $CH(A_{d-1})$ be the convex hulls of A_d and A_{d-1}. Clearly the projection of $CH(A_d)$ on the coordinates $x_1, \cdots, x_{d-1}$ is $CH(A_{d-1})$. Thus to find $CH(A_{d-1})$, it suffices to find $CH(A_d)$, whence $C_{d-1}(n) \leq C_d(n)$. □

The next objective is to establish a lower bound to the number of operations $C_2(n)$ required by any algorithm for finding the convex hull of n points in two dimensions. All known convex hull algorithms in two dimensions obtain the so-called "ordered convex hull," i.e. the sequence of the vertices of the polygon coinciding with the convex hull. However, the following result applies also to algorithms which simply provide the identification of the hull points. The bound is based on a previous result concerning maxima of vectors [8, 11] and on a connection between the two problems which Shamos attributes to A.C. Yao ([9, problem P35)]. It is therefore convenient to recall the result on the maxima of two dimensional vectors. Let A be a set of n two-dimensional vectors with real components. A partial ordering is defined on A in a natural way, i.e. for u and $v \in A$, $u > v$ if and only if $x_i(u) \geq x_i(v)$, for $i = 1, 2$. The maximal elements of this partially ordered set are called the *maxima* of A. Let $M_2(n)$ be the maximum running time of any algorithm for finding the maxima of A. We then recall the following lemma.

Lemma 2. [8]. $M_2(n) \geq O(n \log n)$.

We can now establish the result:

Lemma 3. $C_2(n) \geq O(n \log n)$ *for* $n \geq 3$.

Proof. Let $A = \{a_1, \cdots, a_n\}$ be a planar set of points, and assume that $CH(A) = A$: this means that the points $a_1, \cdots, a_n$ are the vertices of a convex polygon and may be thought of as forming a circuit. There are four points in A, a_{j_0}, a_{j_1}, a_{j_2}, and a_{j_3} such that $x_2(a_{j_0}) = \max_i x_2(a_i)$, $x_1(a_{j_1}) = \max_i x_1(a_i)$, $x_2(a_{j_2}) = \min_i x_2(a_i)$, and $x_1(a_{j_3}) = \min_i x_1(a_i)$ (see Figure 1). These four points identify four "quadrants," and there is a quadrant which contains at least $\lceil \frac{1}{4}n \rceil$ points. Without loss of generality, let the quadrant determined by a_{j_0} and a_{j_1} contain $s \geq \lceil \frac{1}{4}n \rceil$ points. All of these points are maxima in the sense defined above and their identification, by Lemma 2, requires time at least $O(s \log s) = O(\frac{1}{4}n \log \frac{1}{4}n) = O(n \log n)$. □

Thus we reach the following conclusion.

Theorem. $C_d(n) \geq C_{d-1}(n) \geq \cdots \geq C_2(n) \geq O(n \log n)$.

3. Merge Algorithm for Two-Dimensional Sets

The algorithm presented in this section finds the *ordered* convex hull of a set of two-dimensional points.[1] The ordered convex hull in two dimensions is not just the set of the vertices of the convex polygon representing the hull, but the sequence of the vertices on the boundary of this polygon.

Let $A = (a_1, \cdots, a_p)$ and $B = (b_1, \cdots, b_q)$ be two convex polygons in the plane where $(a_1, \cdots, a_p)$ is the (clockwise) sequence of the vertices in the boundary of A, and similarly is $(b_1, \cdots, b_q)$ for B. We assume that $x_1(a_i) < x_1(b_j)$ for $i = 1, \cdots, p$ and $j = 1, \cdots, q$, so that A and B are nonintersecting.

By *merging* A and B we mean the determination of the convex hull $CH(A, B)$ of A and B. The convex polygon $CH(A, B)$ is obtained by tracing the two tangents common to A and B and by eliminating the points of A and B which become internal to the resulting polygon (see Figure 2).

We let l_A and r_A be two points of A such that $x_2(l_A) = \min_i x_2(a_i)$ and $x_2(r_A) = \max_i x_2(a_i)$; similarly l_B and r_B are defined in B. For easy reference, we shall call the two tangents to A and B as *left* and *right*

[1] As mentioned in the Introduction, Shamos independently discovered several planar convex hull algorithms exhibiting the same worst-case time bound as ours [7]. Our algorithm is somewhat more complicated to describe than those of Shamos; however, since it employs a different technique, it is in our opinion worth reporting.

tangent. It is easily realized that the determination of, say, the right tangent depends upon the relative ordering of $x_2(r_A)$ and $x_2(r_B)$; the same can be said for the left tangent in relation to $x_2(l_A)$ and $x_2(l_B)$. Therefore in the sequel we shall consider only one case: specifically, the determination of the right tangent under the hypothesis

$$x_1(r_A) < x_1(r_B) \text{ and } x_2(r_A) < x_2(r_B);$$

the other case, as well as the determination of the left tangent, are treated in an analogous manner. Without loss of generality, we shall also assume that $r_A = a_1$ and $r_B = b_1$. Indices of vertices of A and B are assumed to be taken mod p and mod q, respectively.

Given two points u and v in the plane, (u, v) and (u, v) denote, respectively, the line containing u and v and the segment delimited by u and v. The *slope* $sl(u, v)$ is given by $sl(u, v) = (x_1(u) - x_1(v))/(x_2(u) - x_2(v))$.

We must now determine the two vertices a_{i^*} of A and b_{j^*} of B which are the extremes of the right tangent, where $1 \leq i^* \leq$ index $[l_A]$ and $1 \leq j^* \leq$ index $[l_B]$. We begin by defining the slopes:

$$\alpha_{i,i+1} = sl(a_i, a_{i+1}), \beta_{j,j+1} = sl(b_j, b_{j+1}), \gamma_{ij} = sl(a_i, b_j).$$

Notice that in the ranges $1 \leq i <$ index $[l_A]$ and $1 \leq j <$ index $[l_B]$, due to convexity, the sequences $(\alpha_{12}, \alpha_{23}, \cdots)$ and $(\beta_{12}, \beta_{23}, \cdots)$ are strictly monotone decreasing. Thus the extremes a_{i^*} and b_{j^*} of the right tangent are characterized by the following properties:

$$i^* > 1 \Rightarrow \alpha_{i^*,i^*+1} < \gamma_{i^*j^*} \leq \alpha_{i^*-1,i^*};$$
$$i^* = 1 \Rightarrow \alpha_{12} < \gamma_{1j^*};$$
$$j^* \geq 1 \Rightarrow \beta_{j^*,j^*+1} \leq \gamma_{i^*j^*} < \beta_{j^*-1,j^*};$$
$$j^* = 1 \Rightarrow \beta_{12} \leq \gamma_{i^*1}.$$

We claim that the following algorithm uniquely determines a_{i^*} and b_{j^*}.

Algorithm RT (right tangent)

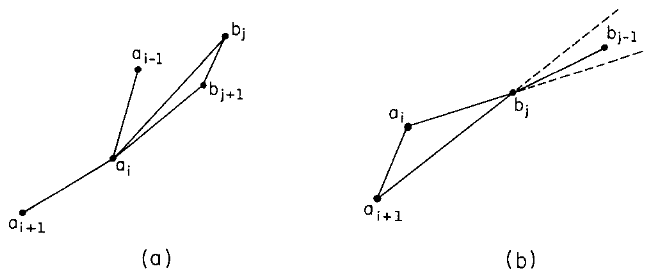
Input. Coordinates of $(a_1, a_2, \ldots, l_A)$ and $(b_1, b_2, \ldots, l_B)$, and slopes $(\alpha_{12}, \alpha_{23}, \ldots)$ and $(\beta_{12}, \beta_{23}, \ldots)$.
Output. i^*, j^*, the indices of the extremes of the right tangent segment.

```
RT1. Set $i \leftarrow 1, j \leftarrow 1$.
RT2. Compute $\gamma_{ij} \leftarrow (x_1(a_i) - x_1(b_j))/(x_2(a_i) - x_2(b_j))$.
RT3. If $\alpha_{i,i+1} \geq \gamma_{ij}$, set $i \leftarrow i + 1$ and go to RT2.
RT4. If $\beta_{j,j+1} > \gamma_{ij}$, set $j \leftarrow j + 1$ and go to RT2.
RT5. Set $i^* \leftarrow i, j^* \leftarrow j$, and halt.
```

We now prove the validity of Algorithm RT. The algorithm halts when the conditions $\alpha_{i,i+1} < \gamma_{j,j+1}$ and $\beta_{j,j+1} \leq \gamma_{ij}$ occur for the first time. Thus all we have to show is that before executing step RT3 we always have $\gamma_{ij} \leq \alpha_{i-1,i}$ and $\gamma_{ij} < \beta_{j-1,j}$. We distinguish two cases: (1) j is incremented or (2) i is incremented.

(1) The index j is incremented when the condition $\alpha_{i,i+1} < \gamma_{ij} < \beta_{j,j+1}$ occurs. Assuming inductively that $\gamma_{ij} \leq \alpha_{i-1,i}$, we have (see Figure 3(a)) $\gamma_{i,j+1} \leq \gamma_{ij} \leq \alpha_{i-1,i}$ and $\gamma_{i,j+1} < \beta_{j,j+1}$: after incrementing j,

Fig. 3. Illustrations for the validity of Algorithm RT.



these conditions become $\gamma_{ij} \leq \alpha_{i-1,i}$ and $\gamma_{ij} < \beta_{j-1,j}$, as desired.

(2) The index i is incremented when $\gamma_{ij} \leq \alpha_{i,i+1}$. Notice that we cannot have $\beta_{j-1,j} \leq \gamma_{i+1,j}$: indeed $\beta_{j-1,j} \leq \gamma_{i+1,j}$ implies $\alpha_{i,i+1} > \gamma_{i,j-1}$, whence, by the formulation of step RT3, the vertex b_j cannot have been reached (see Figure 3(b)) yet by the algorithm; thus we have $\gamma_{i+1,j} < \beta_{j-1,j}$. Next we notice that when $\gamma_{ij} \leq \alpha_{i,i+1}$, we also have $\gamma_{i+1,j} \leq \alpha_{i,i+1}$. The two conditions $\gamma_{i+1,j} < \beta_{j-1,j}$ and $\gamma_{i+1,j} \leq \alpha_{i,i+1}$ become $\gamma_{ij} < \beta_{j-1,j}$ and $\gamma_{ij} \leq \alpha_{i-1,i}$ after incrementing the index i, thus proving our original claim and the validity of the algorithm. It is clear that the number of operations performed by Algorithm RT is $O(i^* + j^*)$.

A procedure analogous to Algorithm RT is required for the determination of the other tangent to A and B (left tangent); clearly, the overall number of operations necessary for determining the two tangents is at most of order $(p + q)$. Finally, we recall that the data structure describing a convex polygon is simply a list giving the circular sequence of its vertices. Thus it is easily realized that the construction of the data structure describing $CH(A, B)$ from the analogous data structures of A and B can be accomplished by modifying a fixed number (two) of pointers. Thus, the overall running time $P_2(n)$ of the merge algorithm of planar sets is at most linear in the total number n of vertices.

4. Merge Algorithm for Three-Dimensional Sets

The merge algorithm for planar sets described in the preceding section can be viewed as constructing a two-dimensional cylinder tangent to two given convex polygons. This idea is the basis for the three-dimensional procedure, which we shall now informally describe.

A convex polyhedron is specified by the sets of its vertices and of the edges connecting them; from these two sets, the set of its faces is readily obtainable. Let A and B be two convex polyhedra with p and q vertices, respectively. Again we assume that for any points a_i of A and b_j of B we have $x_1(a_i) < x_1(b_j)$, so that A and B are nonintersecting.

It is a crucial observation that the sets of vertices and edges of either A or B form a planar graph: if we exclude degeneracies, they form a triangulation. Thus we know that the numbers of edges of A and B are at most $(3p - 6)$ and $(3q - 6)$, respectively, by Euler's theorem (see e.g. [6, p. 189]).

The convex hull $CH(A, B)$ of A and B may be obtained by the following operations (see Figure 4 for an intuitive illustration):

(1) Construction of a "cylindrical" triangulation $\mathfrak{J}$, which is tangent to A and B along two circuits E_A and E_B, respectively.
(2) Removal both from A and from B of the respective portions which have been "obscured" by $\mathfrak{J}$.

Here, the terms "cylindrical" and "obscured" have not been formally defined; rather, they have been used in their intuitive connotations, as suggested by Figure 4.

Alternatively to the generalization of the two-dimensional procedure, the construction of $\mathfrak{J}$ can be viewed as an application of the "gift wrapping" principle of Chand and Kapur [2] to the merging of two convex polyhedra. The gift wrapping principle works as follows. Let C be a polyhedron with n vertices. Assuming a face f of C is given, select an edge of f. This edge and every vertex of C determine a plane; a new face of C belongs to the plane forming with f the largest convex angle. Thus the application of the gift wrapping principle to the construction of the convex hull, as described, requires work $O(n)$ for each new face of C to be determined, yielding a total work $O(n^2)$. We will show below that the properties of convex hulls can be exploited so that *merging* two convex polyhedra by gift wrapping can be done in time at most linear in n.

Fig. 4. Merging two convex hulls. Construction of $\mathfrak{J}$.

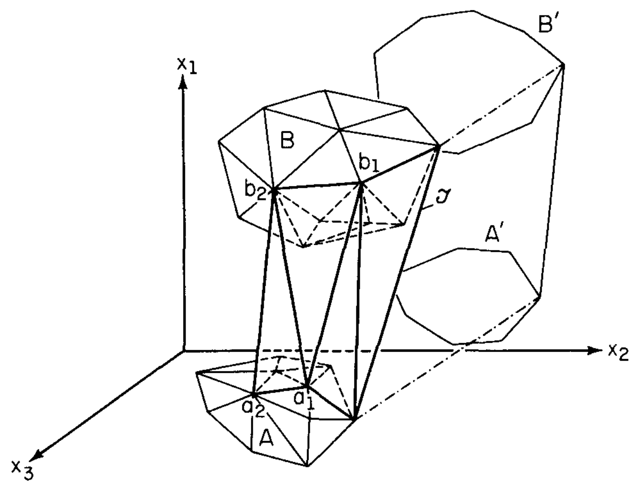
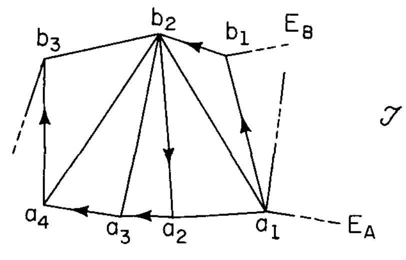


Fig. 5. Fragment of $\mathfrak{J}$ described by the string $a_1b_1b_2a_2a_3a_4b_3$.



The initial step in the construction of the triangulation $\mathfrak{J}$ is the determination of an edge of $\mathfrak{J}$. This is easily done by referring the projections A' and B' on the plane $\langle x_1, x_2\rangle$ (see Figure 4) of the two polyhedra A and B, respectively. We can assume inductively that the convex hulls $CH(A')$ and $CH(B')$ are available at this stage (obviously, $CH(A')$ and $CH(B')$ are nonintersecting): indeed, applying the merge algorithm for planar sets described in Section 3 to $CH(A')$ and $CH(B')$, we obtain two segments tangent to both $CH(A')$ and $CH(B')$. This operation, which runs in time at most $O(p+q)$, not only extends the inductive assumption but also yields a segment whose extreme points are the projections of extreme points of an edge of $\mathfrak{J}$. Thus an edge of $\mathfrak{J}$ has been determined and the construction can be started.

We now describe the advancing mechanism of the procedure. If we temporarily exclude degeneracies, i.e. we assume that each face of $\mathfrak{J}$ is a triangle, each step determines a new vertex, whereby a new face is added to $\mathfrak{J}$. We shall discuss later the case in which the restriction on degeneracies is removed. In our illustration (Figure 4), a_2 and (a_2, b_2, a_1) are, respectively, the vertex and the face of $\mathfrak{J}$ constructed in the previous step. The advancing mechanism makes reference to the most recently constructed face of $\mathfrak{J}$. To initialize the procedure, the reference face is chosen as one of the half planes parallel to the x_3 axis, containing the initially determined edge and delimited by it. Let (a_2, b_2, a_1) be the reference face for the current step. We must now select a vertex $\hat{a}$, connected to a_2, such that the face $(a_2, b_2, \hat{a})$ forms the largest convex angle with (a_2, b_2, a_1) among the faces (a_2, b_2, v), for all $v \neq a_1$ connected to a_2; similarly we select $\hat{b}$ among the vertices connected to b_2. For reasons to become apparent later, we call these comparisons of *type 1*.

Next, once the "winners" $(a_2, b_2, \hat{a})$ and $(a_2, b_2, \hat{b})$ have been selected, we have a run-off comparison, called of *type 2*. If $(a_2, b_2, \hat{a})$ forms with (a_2, b_2, a_1) a larger convex angle than $(a_2, b_2, \hat{b})$, then $\hat{a}$ is added to $\mathfrak{J}$ ($\hat{b}$ is added in the opposite case) and the step is complete. Practically, the triangulation $\mathfrak{J}$ is entirely specified by the circular sequence E_{AB} of the vertices which are successively acquired by the advancing mechanism just illustrated. In fact, this sequence E_{AB} is some interleaving of the two sequences of vertices of E_A and E_B; the interleaving exactly specifies the edges of $\mathfrak{J}$ not belonging to E_A or E_B (see Figure 5).

In the case of a degeneracy, the advancing mechanism fails to construct a new face and simply extends the previously constructed one. Indeed, in this case a type 1 winner face (or, both winner faces) forms with

Fig. 6.

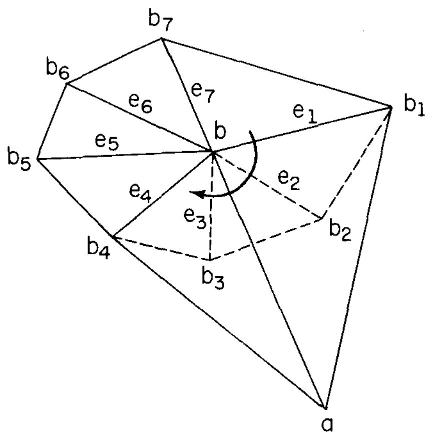
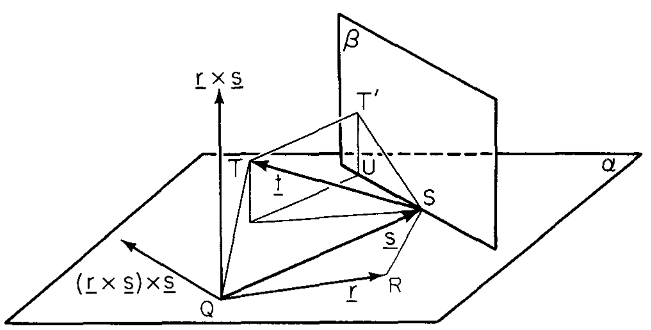


Fig. 7. Illustration of the cotangent calculation.


β
$\underline{r} \times \underline{s}$
T'
T
U
$\underline{t}$
S
α
$\underline{s}$
$(\underline{r} \times \underline{s}) \times \underline{s}$
$\underline{r}$
R
Q

the reference face an angle equal to π, i.e. it is coplanar with it; thus a new vertex (or, new vertices) and possibly new edges are added to the reference face.

To efficiently implement the outlined step, we make the following considerations. First we describe a criterion for uniquely ordering the edges incident on any vertex of A or B. For any a in A (b in B) the edges incident on a (on b) are numbered in ascending order so that they form a counterclockwise (clockwise in B) sequence for an external observer. For concreteness of illustration, suppose now that b and (b, a) are the most recently added vertex and edge of $\mathfrak{I}$, respectively, and let (b_1, b) be the edge of E_B reaching b (see Figure 6). Without loss of generality, we may assume that the numbering of the edges incident on b and of their terminals $b_1, b_2, \cdots, b_k$ be as shown in Figure 6, where $k = 7$. Let (b_s, b, a) be the face which forms the largest convex angle with (b_1, b, a) among the faces (b_i, b, a) for $i = 2, \cdots, k$ (in our case, $s = 4$). It is clear that any (b, b_i) for $1 < i < s$ is an internal edge of the final hull $CH(A, B)$ and need not be further considered.

Thus we can easily upper-bound the number of comparisons of angles between pairs of planes required by the construction of $\mathfrak{I}$. First of all, we notice that each type 1 comparison definitively eliminates one edge of either A or B from those considered by the procedure which constructs $\mathfrak{I}$. Since the numbers of edges of A and B are at most $(3p - 6)$ and $(3q - 6)$, respectively, the number of type 1 comparisons is bounded by $[(3p - 6) - 1 + (3q - 6) - 1] = 3(p + q) - 14$. Next, each type 2 comparison adds a new vertex to either E_A or E_B: since the numbers of vertices of E_A and E_B are at most p and q, respectively, the number of type 2 comparisons is bounded by $(p + q - 1)$. We conclude that the number of angle comparison grows no faster than linearly in the total number of vertices of A and B. Notice that this result rests crucially on the property that the numbers of edges of A and B are linear in their respective numbers of vertices.

It is now worth considering the implementation of the operation of comparing two angles, which is central to the outlined algorithm. We first notice that, due to convexity, all angles to be considered belong to the range $[0, \pi]$. Referring now to Figure 7, consider the convex angle formed by the face QST with the face QRS, lying in plane α. Let β be a plane orthogonal to QS and T' be the projection of T on β: $(\pi - \measuredangle T'SU)$ is the angle between QST and QRS. Since the function cotangent: $[0, \pi] \rightarrow [-\infty, +\infty]$ is an order-reversing mapping, we shall replace the comparison of two angles with the comparison of their cotangents, thereby avoiding costly computations of inverse trigonometric functions. Thus we must compute $\cot(\pi - \measuredangle T'SU) = -\cot(\measuredangle T'SU) = \mathrm{SU}/\mathrm{T'U}$. We shall use vector notation and let "$\times$" and "$\circ$" denote "outer" and "inner" products of 3-dimensional vectors, respectively; also, we let QS = s, and ST = t. Referring to Figure 7, it is obvious that $\mathrm{SU} = K_1 \mathbf{t} \circ ((\mathbf{r} \times \mathbf{s}) \times \mathbf{s})$ and $\mathrm{T'U} = -K_2 \mathbf{t} \circ (\mathbf{r} \times \mathbf{s})$, where $K_1^{-1} = |\mathbf{r}| \cdot |\mathbf{s}|^2 \sin\theta$ and $K_2^{-1} = |\mathbf{r}| \cdot |\mathbf{s}| \sin\theta$, θ being the angle between r and s. It follows that $\mathrm{SU}/\mathrm{T'U} = -\mathbf{t} \circ ((\mathbf{r} \times \mathbf{s}) \times \mathbf{s}) / |\mathbf{s}| \cdot \mathbf{t} \circ (\mathbf{r} \times \mathbf{s})$. If, as is the case with our algorithm, the vector s is the same for all planes whose angles are to be compared, we may replace the comparison of cotangents with that of cotangents multiplied by $|\mathbf{s}|$. It is then straightforward to show that the computation of $|\mathbf{s}| \cdot \mathrm{SU}/\mathrm{T'U}$ requires four multiplications, four additions, and one division.

Once the construction of the triangulation $\mathfrak{I}$ has been completed, i.e. the interleaving E_{AB} of E_A and E_B has been obtained, we must remove those portions of A and B which have become internal to $CH(A, B)$. Concretely, this is done by constructing the data structure describing $CH(A, B)$ from E_{AB} and from the structures describing A and B. The data structure describing a spatial set C may be realized as a collection of lists $\{L(c)\}$, each list $L(c)$ corresponding to a vertex c of C and giving the sequence of the edges incident on c, ordered according to the previously described criterion. By means of a vector of pointers, each list is accessible in fixed time.

We consider the lists of vertices in E_A and E_B. Let $E_A = a_{i_1}, a_{i_2}, \cdots, a_{i_r}$ and $E_B = b_{j_1}, b_{j_2}, \cdots,$

b_{j_t}. Suppose we are currently updating the list $L(a_{i_k})$: typically, E_{AB} contains the substring $\gamma a_{i_k} b_{j_h} b_{j_{h+1}} \cdots b_{j_t} a_{i_{k+1}}$, where $b_{j_h} \cdots b_{j_t}$ is possibly empty and γ is either $a_{i_{k-1}}$ or $b_{i_{h-1}}$. Then we will remove from $L(a_{i_k})$ the edges comprised between $(a_{i_k}, a_{i_{k-1}})$ and $(a_{i_k}, a_{i_{k+1}})$, and insert the sequence $(a_{i_k}, \gamma)(a_{i_k}, b_{j_h}), \cdots, (a_{i_k}, b_{j_t})$: this effects the updating of $L(a_{i_k})$. Note that the work required by this part of the updating procedure is proportional to the number of edges which have to be added when reconstructing the lists of vertices in E_A and E_B, i.e. it is bounded by $O(p + q)$.

The updating task is completed by a simple deletion of the lists pertaining to vertices which have become internal to the hull. Referring, for example, to the polyhedron A, the circuit E_A divides the hull into two portions, the interior one of which is to be deleted. The latter is a planar graph and has $O(p)$ edges at most. Traversing this graph and marking its vertices involves inspecting each edge twice. It follows that the deletion of lists of vertices originally in A and B requires work at most $O(p + q)$.

Therefore, since both the construction of the triangulation $\mathfrak{T}$ and the deletion of obscured portions of A and B are procedures which require a number of operations at most linear in the number of vertices of A and B, this property holds for the merging algorithm as a whole, that is, $P_3(n) = O(n)$.

Acknowledgment. The authors acknowledge with gratitude the constructive criticism of M.I. Shamos, whose extremely valuable suggestions significantly contributed to improving the quality and organization of this paper.

Received May 1975; revised May 1976

References

1. Aho, A.V., Hopcroft, J.E., and Ullman, J.D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass, 1974.
2. Chand, D.R., and Kapur, S.S. An algorithm for convex polytopes. *J. ACM 17,* 7 (Jan. 1970), 78–86.
3. Freeman, H., and Shapira. R. Determining the minimum-area incasing rectangle for an arbitrary closed curve. *Comm. ACM 18,* 7 (July 1975), 409–413.
4. Gilbert, E.N., and Pollak, H. Steiner minimal trees. *SIAM J. Appl. Math. 16,* (1968), 1–29.
5. Graham, R.L. An efficient algorithm for determining the convex hull of a finite planar set, *Inform. Proc. Lett. 1* (1972), 132–133.
6. Grünbaum, B. *Convex Polytopes.* Wiley Interscience, New York, 1967.
7. Jarvis, R.A. On the identification of the convex hull of a finite set of points in the plane. *Inform. Proc. Lett. 2,* (1973), 18–21.
8. Kung, H.T., Luccio, F., and Preparata, F.P. On finding the maxima of a set of vectors. *J. ACM 22,* 4 (Oct. 1975), 469–476.
9. Shamos, M.I. Problems in computational geometry. Dep. Comptr. Sci., Yale U., New Haven, Conn., May 1975.
10. Sklansky, J. Measuring concavity on a rectangular mosaic, *IEEE Trans. Comptrs. C-21* (Dec. 1972), 1355–1364.
11. Yao, F.F. On finding the maximal elements in a set of plane vectors. Comptr. Sci. Dep. Rep., U. of Illinois at Urbana-Champaign, Urbana, Ill., July 1974.

Transient-Free Working-Set Statistics

M.C. Easton and B.T. Bennett
IBM Thomas J. Watson Research Center

Transient-free average working-set size and transient-free missing-page rate for a finite sample of a reference string are defined. Use of these statistics is appropriate if the contents of the working set at the start of the recorded string are unknown. If a certain stationarity condition holds, these statistics provide unbiased estimates of expected working-set sizes, missing-page probabilities, and interreference distance probabilities. Two other pairs of estimators are shown to be biased. Expressions for the transient-free statistics are obtained in terms of interval statistics. Several methods of computation are discussed, the usefulness of each depending on length of the sample, number of distinct references, and the amount of main storage available to the computer performing the calculations. In particular, methods are described for handling long strings containing many distinct page names.

Key Words and Phrases: working set, estimation program behavior

CR Categories: 4.3, 4.6, 5.5

Authors' address: Computer Sciences Department, IBM Thomas J. Watson Research Center, P.O. Box 218. Yorktown Heights, NY 10598.