A FAST CONVEX HULL ALGORITHM *

Selim G. AKL and Godfried T. TOUSSAINT
School of Computer Science, McGill University, Montreal, Quebec, Canada

Received 31 October 1977, revised version received 28 December 1977

Convex hull, algorithm

Introduction

Several algorithms for determining the convex hull of a set of points in two and three dimensions have been published recently [1–6]. It has also been established [7] that the complexity of the 2-dimensional convex hull problem is O(n log n). In this note we describe a new algorithm for obtaining the convex hull of a set of points in the plane and empirically compare it to one of the best known algorithms. The simplicity and speed of the proposed algorithm make it worth reporting.

The basic ideas

The algorithm is based on the following simple ideas:

(1) Determining the four extremal points of the set and discarding all points interior to the convex quadrilateral they form.

(2) Breaking the problem into four subproblems determined by the extremal points.

(3) Using the vector cross-product to find the convex path in each problem.

These ideas are now explained in detail. We assume throughout the following discussion that points are given by their cartesian coordinates.

1. Extremal points

These are the four points with minimum and maximum X and Y coordinates: say XMIN, XMAX, YMIN, YMAX, respectively. From Fig. 1, two facts are obvious,

(a) The extremal points must belong to the convex hull.

(b) Any point interior to the convex quadrilateral whose corners are the extremal points cannot belong to the convex hull.

It follows that by identifying the extremal points one adds these points to the convex hull and discards all points falling inside the quadrilateral the form. We call this the "throw-away" principle.

If there are several candidates with the same ex-

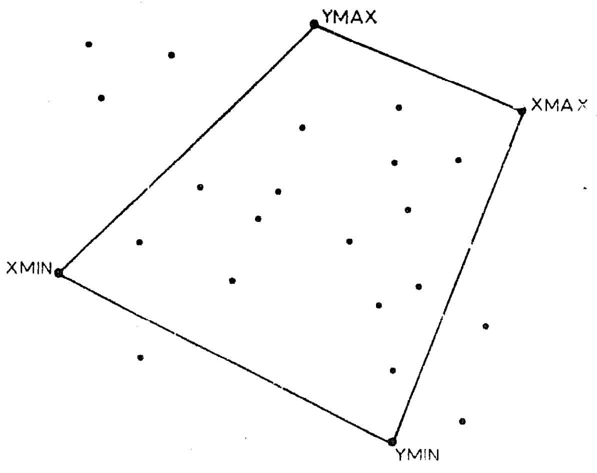

Fig. 1.

* This research was supported by the National Research Council of Canada under grants NRC–A9293 and NRC–A3599.

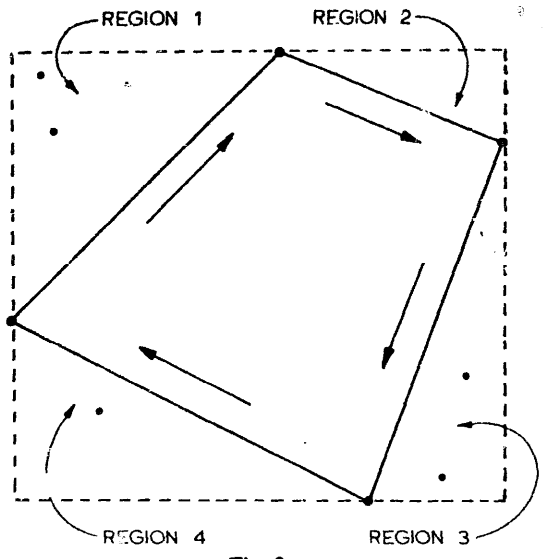

Fig. 2.

treme x or y values their endpoints are kept as extremal points. For example, if several points have the same maximum y coordinates only the leftmost and rightmost of these are kept as YMAX extrema thus yielding a possibly pentagonal "throw-away" region.

2. Subproblems

Once the four extremal points have been determined, and some points eventually discarded, one can break the remaining set of points into four regions, as shown in Fig. 2. All that remains is to find a convex "path" from one extremal point to the other in the same region.

3. Vector cross-product

While examining the points in one of the regions for inclusion in (or exclusion from) the convex hull, assume that we are advancing along an edge of the quadrilateral such that the region is at our left, as shown in Fig. 2. Assume further that we are looking at three consecutive points k, $k+1$, and $k+2$. Obviously if point $k+1$ is as shown in Fig. 3(a) it is to be kept temporarily, while it is to be discarded from further consideration if it is as in Fig. 3(b).

If a, b and θ are as shown in Fig. 3(a) and (b), then the cross-product of the two vectors is given by

$$S = ab \sin\theta$$
$$= (y_{k+1} - y_k)(x_{k+2} - x_{k+1}) + (x_k - x_{k+1})(y_{k+2} - y_{k+1}) .$$

In Fig. 3(a), S is positive and in Fig. 3(b) it is negative. We thus have the following simple rule:

If $S \geqslant 0$ keep point $(k+1)$,

else delete point $(k+1)$.

Before presenting the algorithm we point out that in some cases two extremal points may coincide as shown in Fig. 4. The sole effect of these situations is that the number of subproblems is reduced.

The algorithm

Step 1. Find the extremal points and delete all other points falling inside the polygon they form.

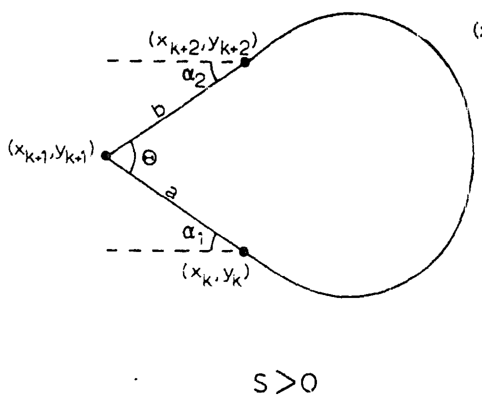

Fig. 3(a).

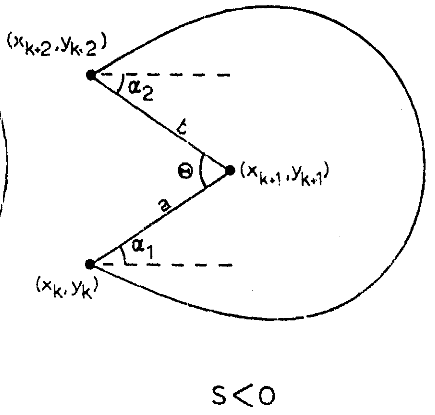

Fig. 3(b).

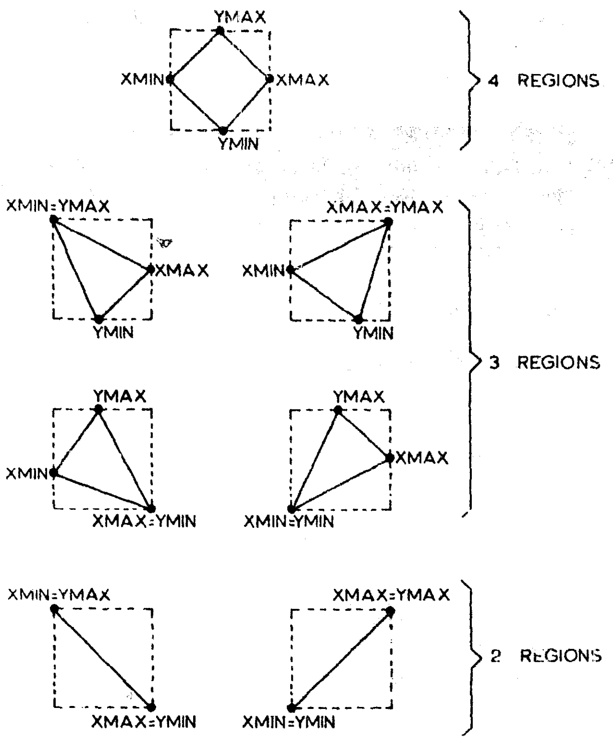

Fig. 4.

Step 2. Sort the remaining points on their x-coordinate: in ascending order for regions 1 and 2 and descending order for regions 3 and 4.

Step 3. For each region find the convex path from one extremal point to the other using the following rule:

```
<<(1) Starting with one extremal point do (a) and
      (b) below for every three consecutive points
      $k$, $k + 1$, $k + 2$ until the other extremal point
      is reached.
        (a) Compute $S$.
        (b) If $S \geqslant 0$ move one point forward; else
            delete point $(k + 1)$ and move one point
            backward.
  (2) If (1) is completed without any deletion,
      stop; else repeat (1)>>
```

In Step 1 a point is assigned to either the quadrilateral or one of the subregions using the point-in-polygon algorithm described in [10,11]. Since at every iteration of Step 3(1) a finite number of points is removed, convergence of the algorithm is guaranteed. The remaining points form the convex hull of the original set.

Worst-case analysis

It is obvious that the computation is dominated by the sorting procedure in Step 2 (the two other steps having a running time proportional to n, the number of points). In the worst case, when all points lie on a straight line, the sorting has a complexity of 0 $(n \log n)$.

Comparison with the algorithm of Jarvis

The new algorithm and the algorithm of Jarvis [2] were programmed in FORTRAN for the IBM 370/158. Test problems were generated randomly in the unit square using a uniform random number generator. In all problems tried the proposed algorithm ran more than 3 times faster than the algorithm of Jarvis. Typically, run-time averages were as follows: $12 \times 10^{-5} \times n \log n$ seconds for the new algorithm and $40 \times 10^{-5} \times n \log n$ seconds for Jarvis' algorithm. (Note that the algorithm of Jarvis was programmed following all possible improvements suggested in [2].)

Advantages of the new algorithm

The principal advantages of the new algorithm can be summarized as follows:

(1) For large values of n it is safe to say that at least half the points are discarded in Step 1. This was generally observed.

(2) Breaking the problem into subproblems by distributing the remaining points into the different regions makes it easier (and therefore faster) to solve. This is a good illustration of the divide-and-conquer concept [5,8].

(3) The cross product criterion is very simple and inexpensive especially when compared to computation of angles, as suggested by Graham [1] shifting of axes, as suggested by Jarvis [2], or evaluation of trigonometric functions as suggested by Anderson [15].

Note that Step 3 in the above algorithm differs from Step 5 in Graham's algorithm [1] in that, not only are angles not computed, but the endpoints in the convex path in each subregion are known thus yielding a simpler termination criterion than that suggested in [16].

Conclusion

A new algorithm has been presented for determining the convex hull of a set of points in the plane. A very efficient preprocessing procedure as well as a novel criterion for identifying convex points were described. Due to its high speed the algorithm is recommended for applications where computation time is the primary factor, such as the Monte Carlo studies considered in [9].

An iterative version of this algorithm that repeatedly uses the "throw-away" principle and does not sort appears in [12]. K.R. Anderson [14] of M.I.T. has independently [illegible]ered a similar implementation of this basic [illegible]ring the reviewing process. We show in [12] and [13] that the iterative algorithm has worst case complexity of O(n log n) and an asymptotic expected run time behaviour of O(n).

References

[1] R.L. Graham, An efficient algorithm for determining the Convex Hull of a Finite Planar Set, Information Processing Lett., 1 (1972) 132–133.

[2] R.A. Jarvis, On the Identification of the Convex Hull of a finite set of points in the plane, Information Processing Lett., 2 (1973) 18–21.

[3] M.I. Shamos and D. Hoey, closet point problems,, Proc. of the Sixteenth Annual Symposium on Foundations of Comp. Sc., IEEE, N.Y. (1975) 151–162.

[4] A. Appel and P.W. Will, Determining the three dimensional convex hull of a polyhedron, IBM J. Res. and Devel., 20 (Nov. 1976) 590–601.

[5] F.P. Preparata and S.J. Hong, convex hulls of finite sets of points in two and three dimensions, CACM, 20 (Feb. 1977) 87–93.

[6] R.A. Jarvis, Computing the Shape Hull of Points in the Plane, Proc. of the IEEE Comp. Soc. Conf. on Pattern Recognition and Image Processing, Troy, N.Y. (June 1977) 231–241.

[7] M.I. Shamos, Geometric Complexity, Proc. of the Seventh Annual ACM Symposium on the Theory of Computing, Albuquerque, N.M. (1975) 224–233.

[8] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison–Wesley, Reading, MA, 1976).

[9] S. Akl, On a generalization of Sylvester's Problem and Some Convex Hull Expectations, Internal Report, School of Computer Science, McGill University.

[10] M. Shimrat, Algorithm 112: Position of point relative to polygon, Comm. A.C.M., 5 (August 1962) 434.

[11] R. Hacker, Certification of algorithm 112 position of point relative to polygon, Comm. A.C.M., 5 (December 1962) 606.

[12] G.T. Toussaint and S.G. Akl, Convex hull algorithms in two and more dimensions (manuscript in preparation).

[13] S.G. Akl and G.T. Toussaint, Efficient convex hull algorithms for pattern recognition applications (submitted for publication).

[14] K.R. Anderson, personal communication.

[15] K.R. Anderson, A reevaluation of an efficient algorithm for determining the convex hull of a finite planar set, Information Processing Lett. 7 (January 1978) 53–55.

[16] J. Koplowitz and D. Jouppi, A more efficient convex hull algorithm, Information Processing Lett. 7 (January 1978) 56–57.