A Counter-Example to a Fast Algorithm for Finding the Convex Hull of a Simple Polygon

Godfried Toussaint
School of Computer Science
McGill University
Montreal

Published in: ***Computer Aided Drafting, Design and Manufacturing***,
Vol. 4, No. 2, December 1994, pp. 1-4.

ABSTRACT

A linear-time algorithm was recently published (*International Conference Proceedings of Pacific Graphics'94 / CADDM'94*, August 26-29, 1994, Beijing, China) for computing the convex hull of a simple polygon. In this note we present a counter-example to that algorithm by exhibiting a family of polygons for which the algorithm discards vertices that are on the convex hull.

Keywords: *simple-polygons, crossing-polygons, convex-hull, algorithms, Graham-scan, computational geometry.*

1. Introduction

Since McCallum and Avis first published a correct linear time algorithm for computing the convex hull of a simple polygon back in 1979 [6] a score of similar algorithms have appeared which used simpler data structures or were conceptually easier to understand. The simplest and most elegant of these algorithms are the Graham-Yao algorithm [4], Lee's algorithm [5], the Bhattacharya-ElGindy algorithm [1], and Melkman's algorithm [7]. Unfortunately many other such algorithms turned out to be incorrect.

The latest algorithm along these lines is due to Deng, Shen and Wang [2]. In this note we show that their algorithm is also incorrect by exhibiting a family of polygons for which their algorithm deletes vertices that are on the convex hull.

2. The Algorithm of Deng, Shen and Wang

We refer the reader to Deng, Shen and Wang [2] for the detailed algorithm and here merely describe the essential features necessary to understand the counter-example. Their algorithm consists of two fundamental operations. The first is a pre-processing step used to reduce the initial problem to four simpler problems. The second step is a variant of the well known Graham-scan used by Graham [3] to find the convex hull of a star-shaped polygon. This step is used in each of the four sub-problems and the four sub-solutions are then stitched together to complete the convex hull.

The pre-processing step itself is further divided into two steps. The first step consists of dividing the polygon *P* into four polygonal chains. This is accomplished by finding the vertices of the polygon with maximum and minimum *x* and *y* coordinates and splitting the polygon at these four points into the four polygonal chains. We may assume that these extreme vertices are unique since if there should be duplications it is a straightforward matter to rotate the polygon by a suitable amount in linear time so that this assumption is satisfied. This decomposition yields a bounding box with four polygonal chains which we may label as NW (north west), NE (north east), SW (south west) and SE (south east). The bounding box, the NW-chain and its end points A and B are illustrated in Fig. 1. Thus we now have four convex hull problems since all we need do is compute the convex hull of each of these chains separately and concatenate the four hulls to obtain the final hull. Therefore we restrict our attention to one of these chains. Without loss of generality let us consider the NW-chain.

The second step in the preprocessing step of the algorithm of Deng, Shen and Wang [2] consists of constructing a straight line through A and B, deleting all vertices of *P* that lie below this line, linking all the remaining vertices in the order in which they appeared originally in *P* and adding the line segment AB between the vertices A and B. These operations are illustrated in Fig. 2 where the white vertices are thrown out and the black vertices are kept as the new polygon. Note that in this example the final polygon is simple. However, as the authors[2] point out in these op-

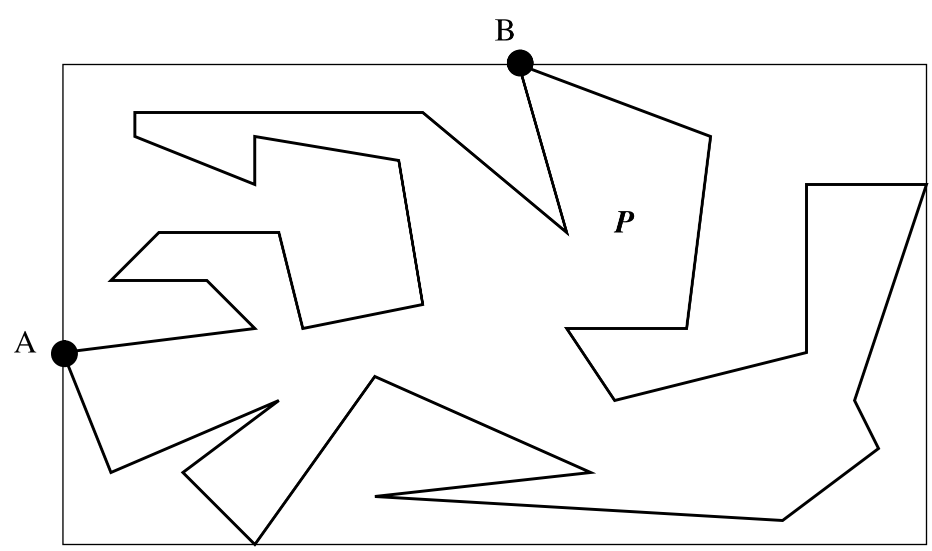

Fig. 1 The bounding box and the extreme points of polygon *P*.

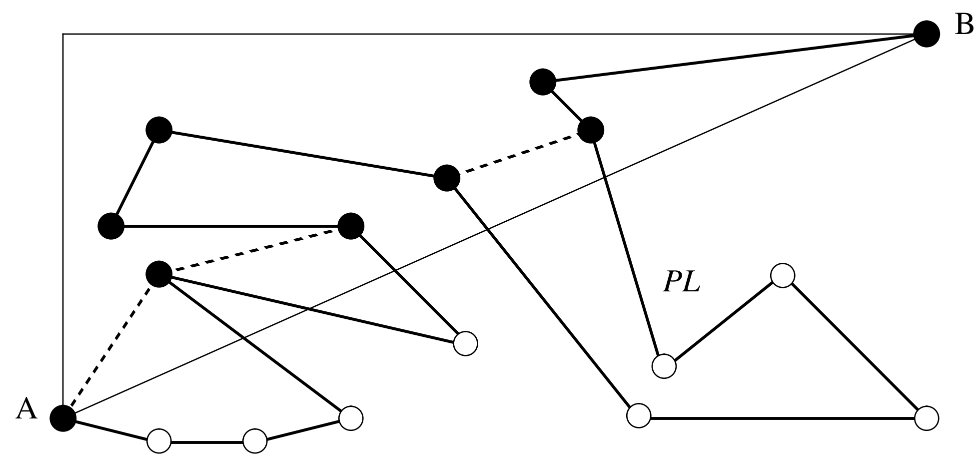

Fig. 2 The vertices of the old polygonal line *PL* that lie below the line through A and B are deleted and the remaining vertices are linked in the same sequence they enjoyed in the original polygon to form a new polygonal line.

erations may in fact yield a non-simple polygon.

The second step in the algorithm then takes as input a polygon obtained from the above preprocessing steps and intends to compute its convex hull by using a variant of the Graham-scan and exploiting the slope structure of the convex hull in the north-west corner of the bounding box. We shall refer to this scan as the DSW-scan for Deng, Shen and Wang. In this step the DSW-scan begins at point A and advances along the polygon in a clockwise manner maintaining a convex polygonal chain in a linked list until the convex chain contains vertex B at its end. The algorithm then exits with a convex polygon. Like the Graham-scan, the DSW scan consists of two fundamental modes: the *advancing* mode and the *backtracking* mode. The backtracking mode in both scans are the same. It is in the advancing mode that the scans differ. Therefore let us consider in detail the logic of the advancing mode of the DSW-scan.

As in [2] let [A=p_1, p_2,...,p_{k-1}], be the portion of the polygon processed so far and let the current result be the convex chain stored in a linked list and denoted by [A=q_1, q_2,...,q_h]. At this time the next input vertex is p_k and is called the *current* vertex. Then they distinguish three different cases where $x(p_k)$ denotes the x-coordinate of p_k:

(1) $x(p_k) = x(q_h)$. The current vertex p_k is either rejected if $y(p_k) < y(q_h)$ or accepted otherwise. In the latter sub-case q_h will be erased.

(2) $x(p_k) < x(q_h)$. Vertex p_k will be rejected if it does not lie above the line defined by q_h and q_{h-1}. Otherwise, it is accepted and q_h removed (see Fig. 3).

(3) $x(p_k) > x(q_h)$. If the slope K_2 of the line defined by p_k and q_h is not greater than zero, the current vertex p_k will be deleted. It is accepted but q_h removed if K_2 is greater than

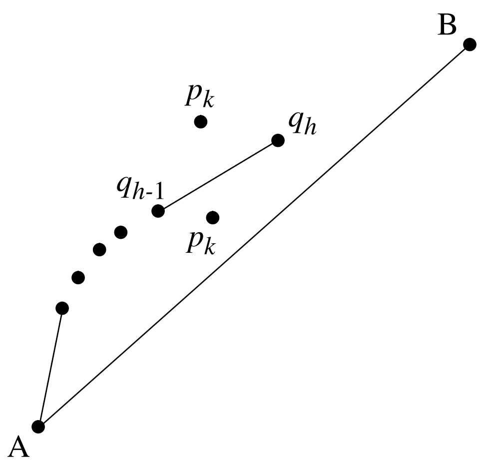

Fig. 3 The current vertex p_k lies to the left of q_h.

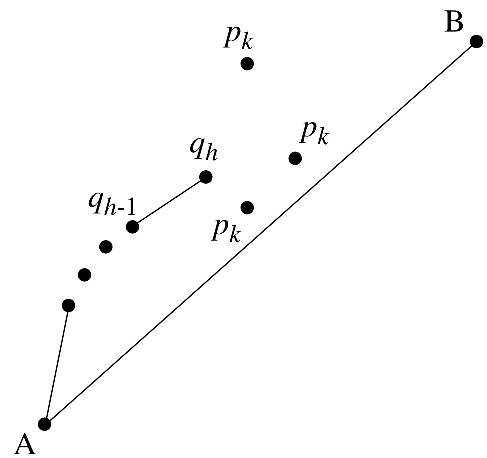

Fig. 4 The current vertex p_k lies to the right of q_h.

that of the line determined by q_h and $q_{h\text{-}1}$. Otherwise, p_k will be accepted without any deletion (see Fig. 4).

The authors then describe the backtracking step which we paraphrase with the following words: note that if q_h is deleted and p_k accepted we will have to check p_k against the previous result (a convex chain obtained when the current vertex is $p_{k\text{-}1}$). Vertex $q_{h\text{-}1}$ and its predecessors will be removed until we encounter a vertex q_f $(1 \leq f \leq h\text{-}1)$ such that the slope of line segment $p_k q_f$ is less than that of segment $q_f q_{f\text{-}1}$.

3. The Counter-example

We now present a simple polygon on which the DSW algorithm fails. An example of such a polygon is illustrated in Fig. 5. Let us step through the algorithm with this polygon in mind. In the pre-processing step we find the bounding box and learn that we only need to compute the NW-convex-hull from vertices A to B in order to find the complete hull. Next, the line through A and B is constructed and all vertices below this line are discarded. In our polygon this means one vertex (the white vertex labelled C) is discarded. The new polygon for which we must compute the convex hull is therefore the non-simple polygon [A, $q_{h\text{-}2}$, $q_{h\text{-}1}$, q_h, p_k, B]. Now let us consider the DSW-scan in part two of the algorithm as it executes on this polygon. At the start of the scan we have that case (3) applies several times, therefore accepting vertices without deletion and creating a convex chain A, $q_{h\text{-}2}$, $q_{h\text{-}1}$, q_h. In the next step p_k is as indicated in Fig. 5. Now we have that case (2) applies. Furthermore, since p_k does not lie above the line defined by q_h and $q_{h\text{-}1}$ it is rejected. However, note that p_k is a convex hull vertex of the polygon. Therefore the DSW algorithm may discard vertices which are on the convex hull and the algorithm fails.

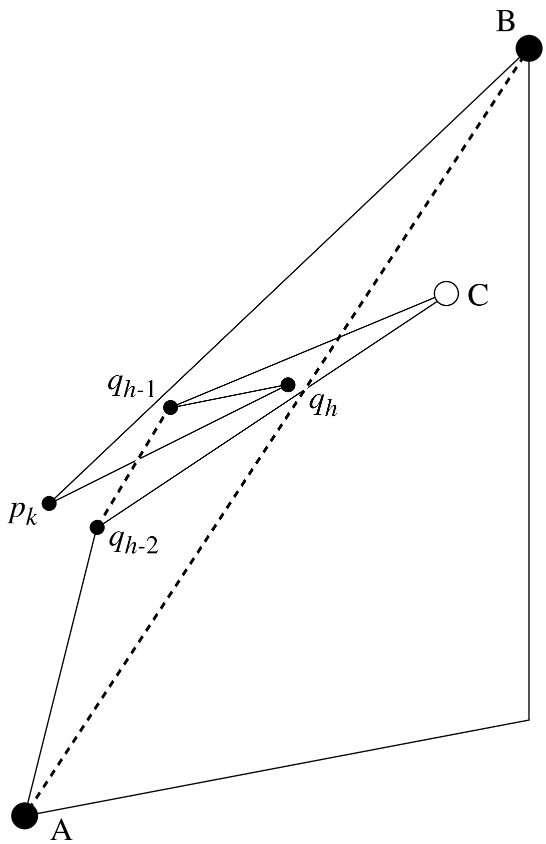

Fig. 5 A polygon on which the DSW algorithm fails.

4. References

1 Bhattacharya B K, ElGindy H. A new linear convex hull algorithm for simple polygons. IEEE Transactions on Information Theory. 1994, IT-30(1):85-88

2 Deng C, Shen M, Wang J. A fast algorithm for finding the convex hull of a simple polygon. International Conference Proceedings of Pacific Graphics'94 / CAD-DM'94, August 26-29, 1994, Beijing, China

3 Graham R L. An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters. 1972, 1:132-133

4 Graham R L, Yao F F. Finding the4convex hull of a simple polygon. Journal of Algorithms. 1983,4:324-331

5 Lee D T. On finding the convex hull of a simple polygon. International Journal of Computer & Information Science. 1983, 12:87-98

6 McCallum D, Avis D. A linear time algorithm for finding the convex hull of a simple polygon. Information Processing Letters. 1979, 8:201-206

7 Melkman A. On-line construction of the convex hull of a simple polyline. Information Processing Letters. 1987, 25:11-12