1983

A Linear Time Algorithm for the Hausdorff Distance Between Convex Polygons

Mikhail J. Atallah
Purdue University, mja@cs.purdue.edu

Report Number:
83-442

TR 442

A Linear Time Algorithm for the Hausdorff Distance Between Convex Polygons

Mikhail J. Atallah
Department of Computer Science
Purdue University
West-Lafayette, IN 47907

Computational Geometry, computational complexity, analysis of algorithms

1. Introduction

We give an $O(m+n)$ time algorithm for computing the Hausdorff distance [GR, page 6] between a convex n-gon and a convex m-gon. First we review the notion of Hausdorff distance between two polygons A and B. The distance from a point x to a polygon P is defined in the obvious way : It is the Euclidean distance from x to the point of P which is nearest to x. However the obvious definition of the distance between A and B as the shortest distance between any point of A and any point of B is unsatisfactory in the sense that two polygons may have points situated close together and yet be widely separated from each other (Fig. 1).

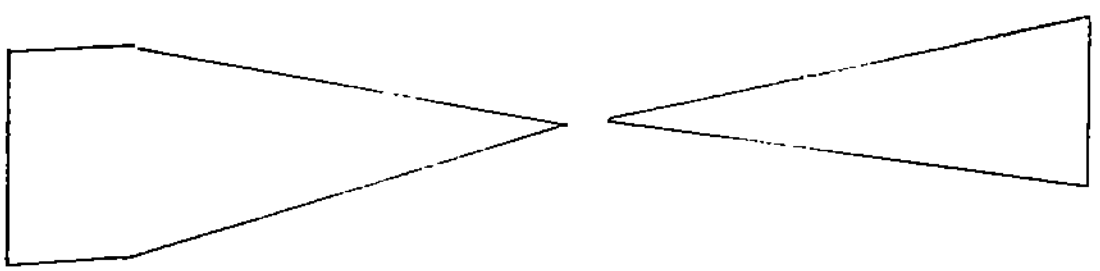

Figure 1 : The above polygons have some close points, yet they cannot be considered as being close

It is natural to expect that a small distance between polygons should mean that

no point on one polygon is far from the other polygon (in this sense the two polygons shown in Fig. 1 are not close). The Hausdorff distance satisfies this expectation and plays a very significant role in mathematics in spite of its apparent complexity. Formally, the Hausdorff distance from polygon A to polygon B is defined by

$$D(A,B) = \underset{x\in A}{MAX}\ \underset{y\in B}{MIN}\ d(x,y)$$

where $d(x,y)$ denotes the Euclidean distance between points x and y. Then the Hausdorff distance *between* A and B is equal to $MAX\{D(A,B),D(B,A)\}$ (In general, $D(A,B)$ is not equal to $D(B,A)$). In words, to get $D(A,B)$ we form all distances between points x and B, where x belongs to A, and then *maximize* over all such points x.

In the next Section, we present some simple facts that are later used in Section 3, where we give an $O(m+n)$ algorithm for computing $D(A,B)$ in the case when A is a convex n-gon and B a convex m-gon.

2. Preliminaries

Throughout the rest of this paper, A is a convex n-gon, B a convex m-gon that is disjoint from A (i.e. they do not intersect and none of them contains the other), and by distance from A to B we always mean the Hausdorff distance $D(A,B)$.

Lemma 1 *Let* $d(x,y)=D(A,B)$, where x is a point of A and y is a point of B. Then

(i) The perpendicular to xy at x is a supporting line of A, and A is on the same side as B relative to that supporting line, and

(ii) The perpendicular to xy at y is a supporting line of B, and x and B are

on different sides relative to that line.

Lemma 2 There is a vertex x of A such that the distance from x to B is equal to $D(A,B)$

(The proofs of the above two Lemmas are easy and are omitted)

Let $v_0,v_1,\cdots,v_{n-1}$ be the vertices of A listed in counterclockwise cyclic order, and let $w_0,w_1,\cdots,w_{m-1}$ be the vertices of B listed in counterclockwise cyclic order. If we let d_i be the distance from v_i to B $(0\le i\le n-1)$, then Lemma 2 says that we need only compute the largest of the d_i's. Computing every d_i separately, however, may take $O(m)$ time for every d_i, resulting in a total of $O(mn)$ time. We will show that it is possible to compute all the d_i's (and hence $D(A,B)$) in $O(m+n)$ time.

For every d_i, let y_i be the point of B for which $d_i=d(v_i,y_i)$. Note that y_i is either a vertex of B, or the foot of the perpendicular from v_i to one of the edges of B. Let σ denote the sequence $y_0,y_1,\cdots,y_{n-1},y_0$. We say that σ is *moving counterclockwise at* y_i if v_{i+1} is to the left of y_iv_i (Fig. 2). If v_{i+1} is to the right of y_iv_i then we say that σ is *moving clockwise at* y_i. If v_{i+1} belongs to the line y_iv_i then we say that σ is *stationary at* y_i.

We say that σ *changes direction at* y_i if it is stationary or moving clockwise at y_{i-1} and it is moving counterclockwise at y_i, or if it is stationary or moving counterclockwise at y_{i-1} and it is moving clockwise at y_i. For example, in Fig. 2 σ changes direction at y_2 and at y_4.

Lemma 3 σ changes direction exactly twice.

Proof : The proof follows from the convexity of the polygons and is left to the reader. ▪

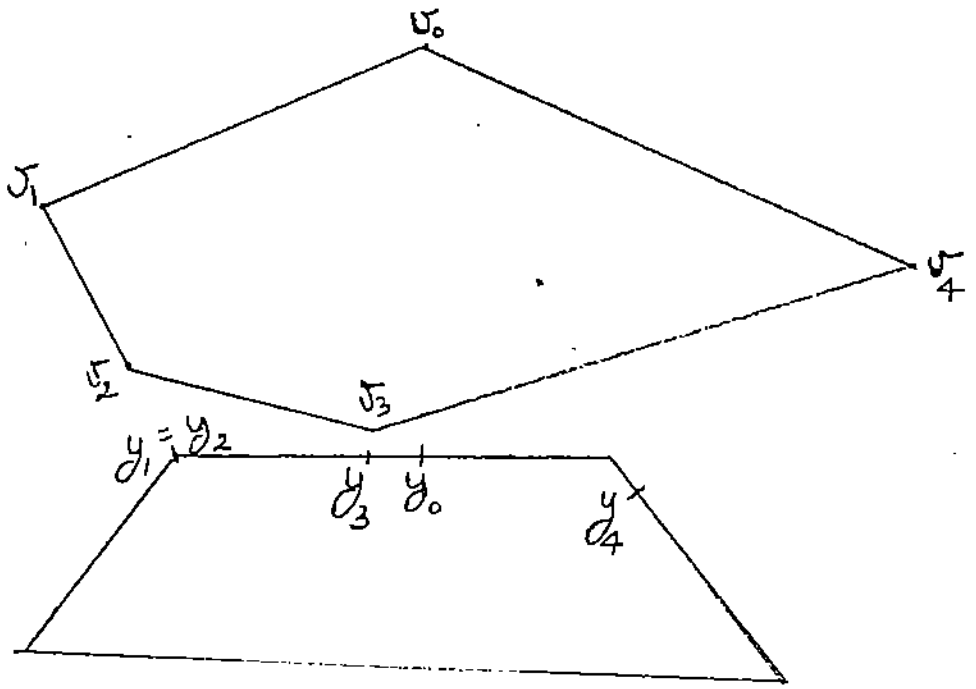

Figure 2 : σ is moving counterclockwise at y_0 and is stationary at y_1

The algorithm presented in the next Section hinges on the above Lemmas.

3. The Algorithm

The following is an informal description of the algorithm for computing $D(A,B)$:

Step 1 Compute d_0 and find y_0 (this can clearly be done in $O(m)$ time). Set $D(A,B) \leftarrow d_0$ and $k \leftarrow 0$.

Step 2 (Finding y_{k+1})

Case 1 v_{k+1} is to the left of $y_k v_k$: Search for y_{k+1} by "scanning" polygon B counterclockwise, starting at y_k, until one of the following two conditions is satisfied :

(i) The edge of B being considered (call it $w_q w_{q+1}$) is such that the foot (call it z) of the perpendicular from v_{k+1} to $w_q w_{q+1}$ is between w_q and w_{q+1}. In this case set $y_{k+1} \leftarrow z$.

(ii) The vertex of B being considered (call it w_q) is such that the perpendicular to $v_{k+1} w_q$ at w_q is a supporting line of B. In this case set $y_{k+1} \leftarrow w_q$.

Case 2 v_{k+1} is to the right of $y_k v_k$: This case is similar to Case 1, except that the search for y_{k+1} is done by "scanning" polygon B clockwise, starting at y_k.

Case 3 v_{k+1} is on $y_k v_k$ (but not necessarily between y_k and v_k) : Set $y_{k+1} \leftarrow y_k$.

Step 3 Set $D(A,B) \leftarrow MAX\left\{D(A,B), d(v_{k+1}, y_{k+1})\right\}$ and $k \leftarrow k+1 \bmod n$. If $k \neq 0$ then go to Step 2, otherwise stop.

(End of Algorithm)

The correctness proof is straightforward and is omitted. That the algorithm runs in $O(m+n)$ time is a consequence of Lemma 3, which guarantees that the repetitive execution of Step 2 will not cause polygon B to be "scanned" more than twice.

To compute the distance *between* A and B, the above algorithm is used twice (once for the computation of $D(A,B)$, and once for that of $D(B,A)$) and then $MAX\left\{D(A,B), D(B,A)\right\}$ is computed.

3. Conclusion

We gave a linear time algorithm for computing the Hausdorff distance between two convex polygons. Even though we only considered the case when

the two polygons are disjoint (i.e. they do not intersect and none of them contains the other), it is easy to modify our algorithm for the case when the two polygons intersect or when one of them contains the other (the details of these modifications are not particularly enlightening and are therefore omitted). It would be interesting to know if a linear time algorithm exists for computing the Hausdorff distance between two convex 3-dimensional polyhedra (the algorithm presented here does not seem to generalize to 3 dimensions).

Acknowledgement

A stimulating conversation with D.T. Lee is gratefully acknowledged.

References

[GR] B. Grunbaum, "Convex Polytopes", Wiley, 1967.