Technical Report
A 86/02

New Algorithms for Shortest Paths avoiding Convex Polygonal Obstacles

Hans Rohnert

FB 10, University of Saarbrücken, West-Germany

Abstract: Two algorithms are presented for computing shortest paths in the plane avoiding convex polygonal obstacles. For the first algorithm the obstacles are assumed to consist of f disjoint convex polygons, for the second algorithm the f polygons may intersect pairwise at most twice. For finding the shortest path between two arbitrary query points the first algorithm takes time $O(n \log n + f^2 \log n)$ and space $O(n)$, the second algorithm preprocesses the polygons in time $O(n \log n + f^3)$ and space $O(n + f^2)$. Thereafter one query takes time $O(n \log n + f^2)$.

Keywords: Shortest paths, convex polygonal obstacles, computational geometry.

1. Introduction

Shortest paths algorithms and methods of computational geometry can be combined in various ways to obtain new algorithms for solving problems which arise in robotics, motion planning and computer graphics. The main interest is focused on 2D- and 3D-space. Obstacles are assumed to consist of disks, rectilinear barriers, line segments, polygons, polyhedra, etc. Perhaps the most general problem in 2D-space is that of n line segments as obstacles. For disjoint line segments Welzl [7] and Ta. Asano, Te. Asano, Guibas, Hershberger and Imai [1] derived independently $O(n^2)$ time and space algorithms. In this paper two special cases of the general problem are considered:

(i) The obstacles are f disjoint convex polygons.

(ii) The obstacles are f convex polygons whose pairwise intersection consists of at most two points.

The paper is organized as follows: Section 2 gives an algorithm for nonintersecting polygons and Section 3 for pairwise intersecting polygons. Boths algorithms are discussed and compared with lower bounds in Section 4. Throughout the paper we borrow heavily from [6].

2. The Obstacles are Disjoint

In [6] the obstacles are assumed to consist of f disjoint convex polygons with a total of n vertices. The paper [6] shows how to preprocess the obstacles in time $O(n + f^2 \log n)$ and by that to construct a graph G of size $O(n + f^2)$ which is a subgraph of the visibility graph. Graph G enables us to compute the shortest path between two arbitrary query points in time $O(n \log n + f^2)$. The authors of [1] discuss this algorithm and propose another algorithm with worse time complexity $O(fn \log n)$, but linear space $O(n)$. The idea behind is to drop the preprocessing step, i.e., consume less space, and to compute on-line visibility from the vertex under consideration in Dijkstra's algorithm. In this section we derive the complete algorithm (in [1] only complexity bounds are given) with improved time complexity $O(n \log n + f^2 \log n)$. Note that $f \leq n$.

Dijkstra's algorithm computes the shortest path from point s to point t as follows: A set S of vertices is maintained whose distance from s is already known ($S = \{s\}$ in the beginning). In every iteration the vertex $v \in V - S$ with minimum distance from s is added to S and all tentative distances from s of vertices adjacent to v are updated. In [6] we prove that shortest paths amidst disjoint convex polygonal obstacles use only the edges (boundaries) of the polygons and supporting segments of the form $\overline{uv}$, $u, v \in V$, where the line supporting $\overline{uv}$ is a supporting

line of two polygons (u is a vertex of the first and v a vertex of the second polygon) and $\overline{uv}$ does not intersect any third polygon.

Here we assume for simplicity that no three vertices are collinear and that the query points lie outside the polygons. Generalization is possible without weakening the complexity bounds.

Then it is natural to define graph $G = (V, E)$ onto which Dijkstra's algorithm can be applied by letting V be the set of the vertices of the polygons and

$$\begin{aligned} E \;=\; & \{\text{edges of the polygons}\} \cup \\ & \{\overline{uv};\ \text{the line through vertices } u \text{ and } v \text{ is a supporting line of} \\ & \quad \text{two polygons and } \overline{uv} \text{ does not intersect any third polygon}\}. \end{aligned}$$

We call the line segments of the form $\overline{uv} \in E$ **useful** supporting segments since shortest paths may use them. Graph $G = (V, E)$ is a subgraph of the visibility graph $VG = (V, E')$ whose edge set E' consists of all line segments $\overline{uv}$ which connect vertices and do not intersect any edge of the polygons (except in endpoints). The preprocessing step of [6] computes G in time $O(n + f^2 \log n)$. The size of G is $O(n + f^2)$ and implies space complexity. Using the ingenious implementation of Dijkstra's algorithm by Fredman and Tarjan [2] one query for the shortest path between two arbitrary points is answered in time $O(n \log n + f^2)$.

In our new algorithm we do not construct set E at once but piece by piece and forget pieces again. Let $D[v]$ denote the tentative distance of v from s, P_{curr} the polygon under consideration and $\ell(\overline{vw})$ the length of line segment $\overline{vw}$, if $(v, w) \in E$, otherwise $\ell(\overline{vw}) = \infty$. Now we can proceed to establish a high-level description of the algorithm (see Figure 1).

```
(1)  S := {s};
(2)  D[s] := 0; P_curr := s;
(3)  for each v ∈ V − {s} do D[v] := ℓ(sv) od;
(4)  while t ∉ S
(5)  do choose v ∈ V − S with minimal D[v];
(6)      S := S ∪ {v};
(7)      let P be the polygon which v belongs to;
(8)      if P_curr ≠ P
(9)      then forget supporting segments emanating from P_curr;
(10)          P_curr := P;
(11)          compute useful supporting segments emanating from P_curr
(12)     fi;
(13)     for all useful supporting segments vw emanating from v
         and for the two edges incident to v
(14)     do D[w] := min{D[w], D[v] + ℓ(vw)} od
(15) od.
```

Fig. 1. Algorithm for Disjoint Polygons.

The correctness of the algorithm should be obvious from the preceding discussions. A good realization of the high-level statements is more difficult. Line (3) requires the knowledge of the useful supporting segments belonging to s. Since a point is a trivial polygon we can solve this problem by simplification of line (11).

Set S is not explicitly maintained, instead, we maintain $V - S$ by a priority queue, here a binary heap. The heap is ordered according to the values $D[v]$. For every vertex v a pointer is maintained on its position in the heap and on the polygon it belongs to. The polygons are assumed to be input in clockwise order and recorded as doubly-connected cyclic lists. Then lines (5)-(7) essentially perform one Deletemin-Operation on the heap taking time logarithmic in the size of the heap, i.e., $O(\log n)$.

Line (9) is executed by using $O(f)$ storage cells for the supporting segments emanating from the polygon under consideration and by overwriting their values in line (11). The supporting segments emanating from P_{curr} are recorded twice: once scattered as linked lists attached to the vertices of P_{curr} which are each one endpoint of a supporting segment and once as one list of size $O(f)$. Then line (9) is implemented by one pass over the big list and then by deletion of the respective lists attached to the vertices. Hence one execution of lines (9)-(10) has time complexity $O(f)$.

Line (11) is realized in two steps: first we compute all $4f - 2$ supporting segments emanating from P_{curr} ($4f - 4$ go to the other $f - 1$ polygons and 2 to the sink) in time $O(\sum_{1 \le i \le f}(\log n_{curr} + \log n_i)) = O(f \log n_{curr} + f \log(n/f))$, where n_i is the number of vertices of the i-th polygon P_i. Then we filter out the useful supporting segments in time $O(f \log f)$ by a cyclic sweep around P_{curr}. Since every polygon is visited on any shortest path at most once lines (8)-(12) have overall time complexity $O(n + f^2 \log(n/f) + f^2 \log f) = O(n + f^2 \log n)$.

Execution of line (14) implicates the adjustment of the position of w in the heap in logarithmic time. Since G has $O(n + f^2)$ edges and every element of E is examined at most twice lines (13)-(14) have $O(n \log n + f^2 \log n)$ overall time complexity. We summarize in

Theorem 1. The shortest path between two points s and t in the Euclidean plane avoiding f disjoint convex polygons with a total of n vertices can be computed in time $O(n \log n + f^2 \log n)$ and linear space $O(n)$. ∎

3. The Obstacles intersect pairwise

Our second algorithm is more general than the algorithm of Section 2 since pairs of convex polygons may be pairwise disjoint, touch in one point or intersect twice. Note that there are no known $O(n^2)$ time shortest paths algorithms for arbitrary

intersecting line segments. But an interesting fact on planar Jordan curves which was discovered by Livne and Sharir [4] helps us to efficiently compute the contour (or boundary) of $K = \bigcup_{1 \le i \le f} P_i$ of the polygons P_i.

Lemma 1. The boundary of f simple Jordan curves in the plane intersecting one another in at most two points contains at most $\max(2, 6f - 12)$ intersection points of the curves.

Proof: see [4] ∎

The significance of Lemma 1 is based on the fact that shortest paths which avoid pairwise intersecting polygons use only edges of the contour and supporting segments which neither intersect the contour nor run inside K. The contour can be computed by an algorithm due to Ottmann, Widmeyer and Wood [5] in time $O((n+s)\log n)$, where s is the number of intersection points. Although Lemma 1 says that the number of intersection points on the boundary of K is small, i.e., $O(f)$, the contour algorithm of [5] considers all $O(f^2)$ intersection points and implies time complexity $O(n \log n + f^2 \log n)$. But this does no harm because the following parts dominate the time complexity.

Having computed the contour of K we compute the useful supporting segments in three steps:

(1) compute all $O(f^2)$ supporting segments,

(2) find the supporting segments having one or two endpoints not on K,

(3) find the supporting segments intersecting properly any polygon-edge.

The useful supporting segments are those *not* found in steps (2) and (3). This property is attached to every supporting segment by a boolean flag. Two non-intersecting (resp. once or twice intersecting) convex polygons P_i and P_j have four (resp. two) common supporting lines and these supporting lines can be computed in time $O(\log n_i + \log n_j)$. Summation over all pairs of polygons yields time $O(f^2 \log(n/f))$ for step (1). Step (2) takes $O(f^2)$ time since we mark every vertex in the contour algorithm whether or not it belongs to the contour of K. Then we can examine each supporting segment in constant time and see whether both of its endpoints lie on the boundary of K or whether one or two of them are interior to K.

Step (3) is realized in time $O(f^3)$ by performing a cyclic sweep around each polygon: first the supporting segments emanating from the considered polygon P_i are arranged in cyclic order according to their angle with the x-axis. The resulting sequence is scanned by comparing the current supporting segment with all other supporting segments emanating from P_i. If the current supporting segment falls into a wedge formed by two supporting segments which join the current polygon with this other polygon then we can check in constant time if the current supporting segment intersects this other polygon. The correctness of this procedure follows from Lemma 5 of [6] that remains intact for the case of intersecting polygons. Yet,

time complexity for one polygon P_i rises from $O(f \log f)$ in [6] to $O(f^2)$ since we do not have such a well defined order of the polygons as in [6].

With knowledge of contour and useful supporting segments it remains to be decided whether two query points s and t lie inside the same component of the connected regions formed by the polygons. This can be done by constructing the inclusion tree of [3] and locating s and t in it in time $O(n \log n)$. We summarize in

Theorem 2. Single source shortest paths problems in the Euclidean plane clustered with f pairwise at most twice intersecting polygons can be solved in time $O(n \log n + f^2)$ after constructing a graph of size $O(n + f^2)$ by a preprocessing step of time complexity $O(n \log n + f^3)$.

Proof: The preprocessing step is described above. The resulting graph consists of the $O(f^2)$ useful supporting segments and the contour which has two kinds of vertices: first $\leq n$ original vertices and then (according to Lemma 1) $O(f)$ points of local non-convexity caused by intersections of polygons. Hence the size of the graph G on which to perform Dijkstra's algorithm is $O(n + f^2)$. This implies space complexity and query time according to [2]. ∎

4. Concluding Remarks

(1) It is not possible to generalize Lemma 1 and thereby to conceive efficient algorithms for cases where convex polygons are allowed to intersect pairwise more than twice: there are instances where convex polygons intersect pairwise four times yielding $\Omega(n^2)$ intersection points (think of $n/2$ pencils oriented vertically in the plane and $n/2$ pencils oriented horizontally, forming altogether a grid).

(2) Our algorithms are only good for small f, say $f = O(n^{1-\epsilon})$ for $\epsilon > 0$. But since $f \leq n$ this should be the case for many applications. Yet, if every polygon has only a constant number of vertices and the polygons are disjoint we prefer the $O(n^2)$ time and space algorithms.

(3) $\Omega(n \log n)$ is the trivial lower bound and is reached by our first algorithm for $f = O(n^{1/2})$, i.e., if every polygon has $\geq c\sqrt{n}$ vertices for some $c \in \mathbf{R}$. The second algorithm reaches the lower bound for $f = O(n^{1/3})$.

5. References

[1] Ta. ASANO, Te. ASANO, L. GUIBAS, J. HERSHBERGER, H. IMAI : "Visibility-Polygon Search and Euclidean Shortest Paths", *Proc. 26th Annual IEEE Symp. on Found. of Comput. Science* (1985)

[2] M. FREDMAN, R. TARJAN : "Fibonacci Heaps and their Uses in Improved Network Optimization Algorithms", *Proc. 25th Annual IEEE Symp. on Found. of Comput. Science*, pp. 338–346 (1984)

[3] K. KEDEM, M. SHARIR : "An Efficient Algorithm for Planning Collision-free Translational Motion of a Convex Polygonal Object in 2-dimensional Space amidst Polygonal Obstacles", *Proc. 17th Annual ACM Symp. on Theory of Computing* (1985)

[4] R. LIVNE, M. SHARIR : "On Intersection of Planar Jordan Curves", N.Y. Univ., Courant Institute, Techn. Rep. 153 (1985)

[5] T. OTTMANN, P. WIDMEYER, D. WOOD : "A Fast Algorithm for Boolean Mask Operations", Inst. für Angew. Mathematik und Formale Beschreibungsverfahren, D-7500 Karlsruhe, Rept. No. 112 (1982)

[6] H. ROHNERT : "Shortest Paths in the Plane with Convex Polygonal Obstacles", Techn. Rep. A 85/06, University of Saarbrücken, West-Germany, to appear in *Inf. Proc. Lett.*

[7] E. WELZL : "Constructing the visibility graph for n line segments in $O(n^2)$ time", *Inf. Proc. Lett.* Vol. **20** (1985), pp. 167–171