Mathl Modelling, Vol. 9, No. 1, pp. 63–68, 1987
Printed in Great Britain.

0270-0255/87 $3.00 + 0.00

A LOCALLY OPTIMAL FAST OBSTACLE-AVOIDING PATH-PLANNING ALGORITHM

Y. LIROV
Department of Systems Science and Mathematics, Washington University, St Louis, MO 63130, U.S.A.

(*Received* 22 *December* 1986)

Communicated by E. Y. Rodin

Abstract—Two algorithms to compute the shortest collision-free paths in the Euclidean plane are presented. The f obstacles are assumed to be described by disjoint convex polygons having N vertices in total. After preprocessing time $O(N + f \log N)$, a suboptimal shortest path between two arbitrary query points can be found in $O(f + N \log N)$ time using Dijkstra's algorithm and in $\Theta(N)$ time using the A^* algorithm. The space complexity is $O(N + f)$.

1. INTRODUCTION

The class of problems of collision-free paths finding for a moving object in the Euclidean plane cluttered with obstacles is known as collision-free path planning. The solutions to this class of problems find their applications in such areas as mobile robots path planning and computer graphics.

Previous attempts at collision-free path planning can be classified into two different approaches [1]: a model-based approach and a non-model-based approach. In the model-based approach models of the object and the obstacles are preprocessed and stored in the computer, and used as references in the path-planning algorithms. In the non-model-based approach the image maps based on visual input are utilized when planning a path.

The non-model-based approach reduces the representation cost to a minimum while the planning cost is usually high. Examples of utilizing this approach can be found in Refs [1, 2].

The model-based approach can be further subdivided into the configuration space approach and the free-space representation space approach. The configuration space approach [3] is based on the idea to shrink the object into a single point while at the same time expanding the obstacles according to the object shape. The visibility graph is constructed and the search for the minimum path is subsequently performed on that graph. The visibility graph indicates all collision-free straight-line paths among the expanded polygonal obstacle vertices.

The free-space approach is based on direct representation of the free space by using basic shape primitives. Chatila [4] uses convex polygons to represent the free space. The connectivity graph, where each node is a free-space convex polygon and each link corresponds to the common edge segments shared by adjacent polygons, is constructed. The overlapping generalized cones [5] can also be used to represent the free space. The path planning reduces to the search through the connectivity graph where each node corresponds to the intersection of two generalized cone axes and each link corresponds to the cone axis.

The main advantage of the model-based algorithms over the non-model-based algorithms is the faster running speed. The main disadvantage is the high preprocessing time demands that the model-based algorithms require before the actual planning can start. The best recent result is proposed by Ronhert [6], who achieves $O(f^2 + N \log N)$ running time and $O(N + f^2 \log N)$ preprocessing time complexities by cleverly constructing the visibility graph and then implementing Dijkstra's algorithm.

In this paper we present a solution to the collision-free path-planning problems using the model-based, configuration space approach. By employing Voronoi diagrams in the preprocessing stage, we achieve the $O(N + f \log N)$ preprocessing time complexity and $O(f + N \log N)$ running time complexity using Dijkstra's algorithm. This result is further improved to $\Theta(N)$, linear expected running time complexity, by utilizing the A^* search algorithm.

MM-9/1-E

2. VISIBILITY GRAPH CONSTRUCTION

We assume throughout the paper that the obstacles are represented by f disjoint convex polygons with N vertices in total. Furthermore, we assume that the two query points s and t describing the optimal collision-free path do not lie in the interior of any polygon and that no three vertices are colinear.

Lemma 1 [6]

The shortest path between s and t in the plane avoiding convex polygonal obstacles is defined by edges of the polygons and supporting segments of pairs of polygons.

The problem of obstacle-avoiding path planning, therefore, is transformed to the problem of supporting segments efficient construction and search in the resulting visibility graph. The visibility graph is constructed of all N vertices, obstacle edges and those supporting segments that do not intersect any of the polygons.

3. SUPPORTING-LINE GENERATION

Lemma 2 [6]

For two disjoint convex polygons there exist at most four common supporting segments.

Lemma 3 [7]

The four supporting segments between two polygons P_i and P_j can be computed in $O(\log(n_i \cdot n_j))$ time, where n_i and n_j are the number of vertices of P_i and P_j, respectively.

Efficient supporting segments generation becomes the key issue in the preprocessing stage. Two approaches can be used on this matter: full set of supporting segments generation $\left[\text{total of} \frac{f(f-1)}{2}\right]$, as done by Ronhert [6], or selective supporting-line generation. Most of the supporting lines in the full set are not useful, since it is intuitively clear that most of them will cross the interior of other convex polygons. This intuition is supported also by the results of Ronhert [6], who generates the full set in quadratic time and then eliminates the non-useful ones in $O(f^2 \log f)$ time.

Our approach is to selectively generate only those supporting segments that have the most chances to be included in the shortest path. We propose to construct the supporting segments only between the neighboring obstacles.

Efficient determination of the neighboring obstacles can be performed by introducing the Voronoi diagram of the set of obstacles. The f regions partition of the plane, such that every region consists of the set of points which are closest to a particular obstacle, is called the Voronoi diagram of the set of obstacles. However, the Voronoi diagram of the convex polygons is a complex curve which is difficult to compute [8].

For our purposes it is enough to use only the polygon centers to define the neighborhood relations.

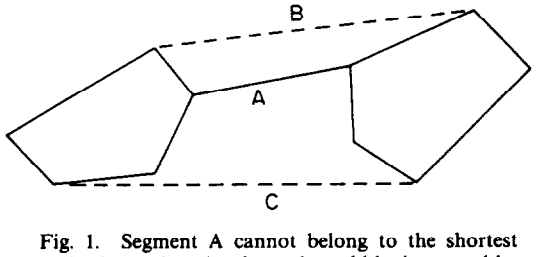

Fig. 1. Segment A cannot belong to the shortest path, since otherwise the path could be improved by taking B or C.

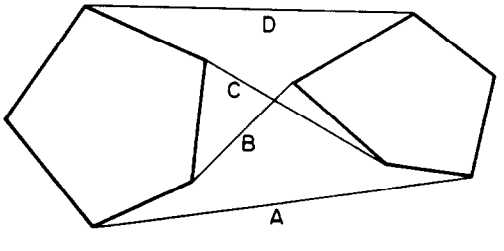

Fig. 2. The four supporting segments A, B, C and D.

Let the set of points $(T_1, \ldots, T_f)$, each corresponding to some polygon, be defined by the respective polygon's arithmetic sum of some of its vertices.

Lemma 4

The nearest neighbors for every point in the set $(T_1, \ldots, T_f)$ are computed in $O(f \log f)$ time and this is optimal.

Proof. Use the Voronoi diagram and apply Theorem 5.15 of Preparata and Shamos [9].

Lemma 5

Every nearest neighbor defines an edge in the Voronoi diagram (Theorem 5.9 in Preparata and Shamos [9]).

Lemma 6

The Voronoi diagram has at most $3f - 6$ edges and $2f - 5$ vertices.

Proof. Corollary 5.2 in Preparata and Shamos [9].

Theorem 1

There are at most $4(3f - 6)$ supporting lines between the nearest neighboring polygonal obstacles.

Proof. Follows from Lemmas 2, 5 and 6.

Theorem 2

The supporting lines between the neighboring polygonal obstacles can be computed in $O\left(f \log \frac{N}{f}\right)$ time.

Proof. In the worst case there will be $3f - 6$ neighbors, each having $\left\lceil \frac{N}{f} \right\rceil$ vertices, so that every pair will have all 4 supporting lines. Every pair requires $O(\log n_i + \log n_j) = O\left(\log \frac{N}{f}\right)$ time to construct the supporting lines. Summation over all the neighbors gives at most $O\left(f \log \frac{N}{f}\right)$ total running time.

As mentioned in Section 2, the shortest paths traverse obstacles and the supporting segments of pairs of polygons. However, a supporting segment that intersects the interior of a polygon does not belong to the shortest path. Following Ronhert [6], we call the segments that intersect the interior of an obstacle non-useful.

Lemma 7

The non-useful supporting segments can be eliminated in $O(N)$ time and $O(f)$ storage.

Proof. Since the supporting lines were constructed only between neighboring polygons, they can intersect only a neighboring polygon. The total of neighboring polygons at every Voronoi diagram node is 3. In the worst case, each has $\left\lceil \frac{N}{f} \right\rceil$ edges. So at most there will be 12 tests for segments intersection of a polygon having $\left\lceil \frac{N}{f} \right\rceil$ edges, which takes $O\left(\frac{N}{f}\right)$ time. Since the number of vertices on the Voronoi diagram is at most $2f - 5$, in total, there will be at most $(2f - 5)\, O\left(\frac{N}{f}\right) = O(N)$ time.

The visibility graph constructed using the supporting lines only between the neighboring polygons has at most $4(3f - 6)$ supporting lines, according to Theorem 1.

4. SUBOPTIMALITY vs SPEED

Observe, that, using our approach there may be cases when some supporting lines that could belong to the optimal path are not constructed at all. For example, in Fig. 4, the polygons B and D are not neighbors and therefore, no supporting segments are constructed. If there was an optimal path from D to B, then it would have to touch either A or C, and no straight supporting line between B and D would exist.

In order to judge the above described observation, let us first consider the main merits which are usually used when choosing an algorithm for the shortest path planning. In practical applications three criteria play major roles in choosing a planning algorithm:

(1) the resulting path must connect the two given points and must avoid the obstacles;
(2) it must be as short as possible; and
(3) it must be computed as fast as possible.

In real-time applications, when the terminal point is allowed to move, or, when the obstacles set is changing its configuration or, when the obstacles themselves change their shapes, the requirement for the global path optimality becomes senseless. On the other hand, the demand for the computational speed becomes the major consideration when choosing the shortest path-planning algorithm, since the planning process is repeatedly performed at every predefined time increment.

In such cases our approach for the visibility graph construction is the most useful one (cf. Lemma 4, Theorem 2 and Lemma 7). Moreover, since the resulting visibility graph is as sparse as it can possibly be, the shortest path-search algorithm will require also the least running time.

5. VISIBILITY GRAPH SEARCH

We may use either of the following two approaches to search for the solution in the visibility graph. The first employs Dijkstra's algorithm and the second employs the A^* algorithm.

Theorem 3

The shortest suboptimal path between two points s and t in the presence of f disjoint convex polygonal obstacles with N vertices in total can be computed in $O(N + f)$ space, and $O(f + N \log N)$ time after $O(N + f \log N)$ preprocessing time.

Proof. The input has cost $O(N)$. The Voronoi diagram construction is done in $O(f \log f)$ time. The supporting segments are constructed in $O\left(f \log \frac{N}{f}\right)$ time, according to Theorem 2. The elimination of the non-useful ones can be done in $O(N)$ time, according to Lemma 7. So, the total

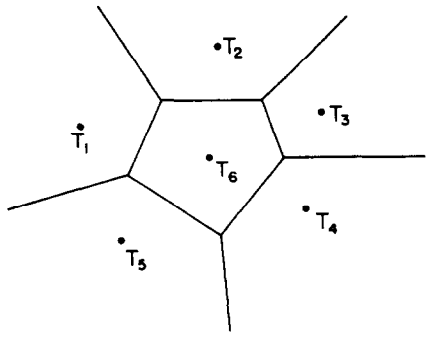

Fig. 3. The Voronoi diagram.

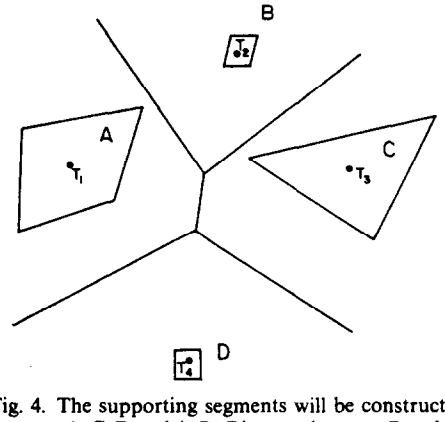

Fig. 4. The supporting segments will be constructed between A, C, D and A, B, C but not between B and D.

is $O(N + f \log N)$ time for the preprocessing stage to construct the connectivity graph. Note that $N > f$.

Dijkstra's algorithm can be implemented according to Fredman and Tarjan [10] in $O(|E| + |V| \log |V|)$ time. In our case, $|V| = N + 2$, and $|E| < 4(3f - 6) + 4f + N$. Therefore, we get a run time of $O(f + N \log N)$.

Space complexity is $O(N + f)$, since every polygon and every vertex and all the information pertaining to the above must be stored.

The second approach to the search is based on the use of the A^* algorithm [11]. The evaluation function f used to evaluate an expanded node is of the form

$$f(n) = g(n) + h(n)$$

for every node n, where

$$g(n') = g(n) + c(n, n'),$$

n' belongs to the set of successors of n, $g(n)$ is the cost of the path from s to n, $h(n)$ is the heuristic estimate of the cost of the remaining path from n to n' and $c(n, n')$ is the cost of the path from n to n'.

We take the cost function to be defined by the distance between two nodes, and the estimate of remaining cost to be the straight-line distance to the goal.

Theorem 4

The A^* algorithm returns an optimal solution when one exists in linear expected time complexity $\Theta(N)$.

Proof. The admissibility follows from the choice of the heuristic function h, since

$$h(n) \leqslant h^*(n), \qquad \forall n$$

where $h^*(n)$ is the cheapest cost of the path going from n to t. The linear expected time complexity is proved by Pearl [11] for the case where the N vertices are randomly scattered with uniform density.

Note that the A^* algorithm performance is independent of the visibility graph complexity.

6. SUMMARY

Three criteria play major roles in choosing an algorithm for the shortest path planning in obstacle-cluttered space between two given points:

(1) the resulting path must connect the two given points and must avoid the obstacles;
(2) the resulting path must be as short as possible; and
(3) the resulting path must be computed in the shortest possible time.

The visibility space approach is utilized to solve the collision-free shortest path in the Euclidean plane cluttered with convex polygonal obstacles. The preprocessing stage is shown to have $O(N + f \log N)$ time complexity to construct the visibility graph. Dijkstra's algorithm performing the search on the resulting graph runs in $O(f + N \log N)$ time complexity. These results improve the best known time complexities of $O(N + f^2 \log N)$ and $O(f^2 + N \log N)$, respectively. The use of the A^* algorithm yields the $\Theta(N)$ expected running time.

The Voronoi diagram of the set of obstacles provides an efficient mechanism to construct such a visibility graph.

The significant reductions in the reprocessing and running times are achieved by giving up the global optimality requirement of the resulting shortest path. This requirement relaxation enables one to consider the visibility graph consisting of the obstacles edges and of the supporting segments between the neighboring obstacles only.

REFERENCES

1. E. K. Wong and K. S. Fu, A hierarchical orthogonal space approach to three dimensional planning. *IEEE Jl Robotics Automn* **RA-2**(1), 42–53 (1986).
2. S. Kambhampati and L. Davis, Multiresolution path planning for mobile robots. *IEEE Jl Robotics Automn* **RA-2**(3), 135–145 (1986).
3. T. Lozano-Perez, Automatic planning of manipulator transfer movements. *IEEE Trans. Syst. Man Cybernet.* **SMC-11**, 681–698 (1981).
4. R. Chatila, Path planning and environment learning in a mobile robot system. In *Proc. Eur. Conf. on Artificial Intelligence*, Orsay, France (1982).
5. R. A. Brooks, Solving the find-path problem by good representation of free space. *IEEE Trans. Syst. Man. Cybernet.* **SMC-13**(3), 190–197 (1983).
6. H. Ronhert, Shortest paths in the plane with convex polygonal obstacles. *Inf. Process. Lett.* **23**, 71–76 (1986).
7. H. Edelsbrunner, Finding extreme distances between convex polygons. *J. Algorithms* **6**(2), 213–224 (1985).
8. R. L. Drysdale, Generalized Voronoi diagrams and geometric searching. Stanford, CS Report STAN-es-79-705, Stanford, Calif. (1979).
9. F. P. Preparata and M. I. Shamos, *Computational Geometry*. Springer, New York (1985).
10. M. Fredman and R. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms. In *Proc. 25th A. IEEE Symp. on Foundations of Computer Science*, pp. 338–346 (1984).
11. J. Pearl, *Heuristics—Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, Mass. (1984).