Path planning in polygonal domains for robots with limited turning abilities

Mohammad Reza Ranjbar Divkoti
Computer Engineering Department
Ferdowsi University Of Mashhad
Mashhad, Iran
Email: mohammadreza.ranjbardivkoti@stu.um.ac.ir

Mostafa Nouri-Baygi
Computer Engineering Department
Ferdowsi University Of Mashhad
Mashhad, Iran
Email: nouribaygi@um.ac.ir

***Abstract*—Path planning among polygonal obstacles is a well-known problem in robotics. In this paper, we consider the problem of planning a collision-free path for a robot in a polygonal domain from a given source point to a given target point. The robot has two basic limitations: an upper bound on the angle of rotation and a lower bound on the distance between two consecutive turns. We describe an algorithm that runs in $O(n^4)$ time and finds a path in accordance with the above limitations. As shown by experiments, the output of the algorithm is much close to the shortest path with the requirements. We further demonstrate how to decompose the algorithm into two phases, preprocessing time and query time. In this way, given a fixed start point and a set of obstacles, we can preprocess a data-structure of size $O(n^4)$ in $O(n^4)$ time, such that for any query target point we can find the above-mentioned path in $O(n^2)$ time.**

I. Introduction

The path planning problem used in various domains like robotics, road-building, navigation, aviation industry and so on. In this paper, we study the problem of path planning among polygonal obstacles. The goal is to generate a path from a source point to a target point such that the path does not pass through the interior of obstacles, the angles of rotations are less than a fixed given value and the distance between two consecutive turns is greater than a fixed given length.

If all inputs of the problem is given at the same time we solve the problem in $O(n^4)$ time. In another version of the problem, we may need the path to different target points, from a fixed source point through a fixed set of obstacles. In this case, we preprocess a data structure of $O(n^4)$ size in $O(n^4)$ time, and answer any query target point in $O(n^2)$ time.

The remaining of the paper is organized as follows: The related work is summarized in Section II. The problem is described in Section III. In Section IV, we discuss the geometry prerequisites necessary to solve the problem. We present the algorithm to solve the problem in Section V. The extension of the algorithm is described in Section VI and its evaluation is given in Section VII. Section VIII concludes the paper.

II. Related Work

The exact problem we study in this paper, as far as we know, was not studied by theoretical computer scientists. A related problem studied extensively is the shortest path problem in a polygonal domain. Kapoor and Maheshwari [1] solved this problem in $O(m^2 \log n + n \log n)$ time, where n is the total number of vertices of obstacles and m is the number of obstacles. Inkulu *et al.* [2] gave an algorithm that computes the shortest path in $O(T+(m \log m)(\log n))$ time and $O(n)$ space, where $O(T)$ is the time to triangulate the polygonal domain. Kapoor *et al.* [3] presented an algorithm that preprocesses the scene in $O(n + h^2 \log n)$ time and finds the shortest path for a query target point t in time $O(\log n)$.

Unlike theoretical results, there are some heuristic algorithms for path planning that have similar limitations to our problem. Liu *et al.* [4] use OARPER (operational area restriction polar extending recursively) method to evade obstacles and generates superior path trees and then obtains the desired path by multi-attribute fuzzy optimization (MAFO) method. Samar and Kamal [5] give a path planner for Unmanned Aerial Vehicle (UAV). Wang *et al.* [6], Chen and Xu [7] and Su and LI [8] also present path planning algorithms based on genetic algorithm.

III. Problem Description

We here formally define the problem. In the plane, a set $O = \{P_1, P_2, \ldots, P_h\}$ of h disjoint simple polygons, called obstacles, which totally have n vertices, a source point s, a target point t and two constant values l and α are given. We want to find a path in the exterior space of polygons of O, called the free space, from s to t with the following properties:

- *Minimum turning distance*: the path consists of straight line segments, each one of length at least l.
- *Connectedness*: the consecutive segments are connected at turn points.
- *Maximum turning angle*: the turning angles at the turn points are at most α.

Figure 1 depicts the set $O = \{P_1, P_2, P_3\}$ of three obstacles, with $n = 14$ vertices, source point s and target point t. Each path from s to t is composed of a chain of segments. The illustrated path $\{e_1, e_2, e_3\}$ in the figure have the above properties if $l \leqslant |e_1|, |e_2|, |e_3|$ and $\theta_1, \theta_2 \leqslant \alpha$.

IV. Preliminaries

In this section, we introduce some geometric tools used for solving the problem. In Section IV-A we define regular chains

978-1-5386-0804-3/17/$31.00

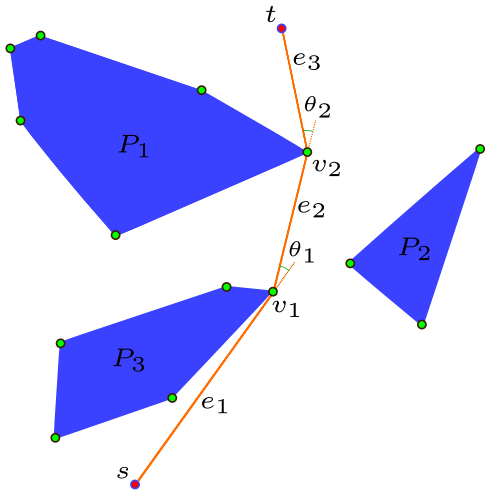

Fig. 1. Path planning in the presence of polygon obstacles with connectedness, minimum turning distance, and maximum turning angle properties.

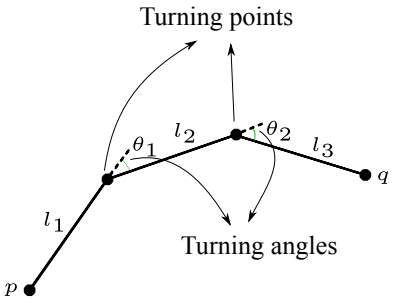

Fig. 2. chain of segments from p to q with condition $l_1, l_2, l_3 \leqslant l$, $\theta_1, \theta_2 \leqslant \alpha$.

of segments. In Section IV-B the ray shooting data structure briefly explained.

A. Regular chain of segments

In our problem we consider only paths consisting of a sequence of connected line segments. We call such a sequence a *chain of segments*. Each two consecutive segments have a common endpoint, called the *turning point*. The angle between a segment and the extension of the previous segment in the turning point is called the *turning angle*. (Figure 2)

We call a chain of segments with all turning angles equal to α and equal turning distances a *regular chain of segments*. Intuitively, it is part of a regular polygon with exterior angle α.

We here calculate the position of vertices of a regular chain of segments with different number of turning points, while the position of the start and the end points are known.

Let C denote a regular chain of segments with A_0 as the starting point and k turning points. We assume the x-axis is the rightward horizontal line through the origin and the y-axis is the upward vertical line through the origin. Let A_i, $1 \leqslant i \leqslant k$, be the i^{th} turning point of C and A_{k+1} be the last point. Let e denote the length of the segments of C. For ease of illustration, we consider A_0 as an imaginary turning point

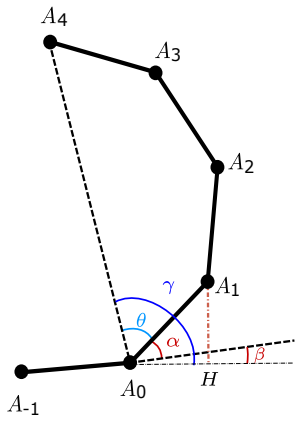

Fig. 3. A regular chain of segments from point A_0 to A_4 with three turning points.

and denote its previous point by A_{-1} (Figure 3). Let β denote the angle between the extension of $\overline{A_{-1}A_0}$ and the x-axis. Since C is a regular chain of segments, the angle between the extension of $\overline{A_{-1}A_0}$ and $\overline{A_0A_1}$ is α. Let the angle between $\overline{A_0A_1}$ and $\overline{A_0A_{k+1}}$ (the segment from the starting point to the end-point) be denoted by θ.

Draw a line from A_0 parallel to x-axis and project A_1 on this line. Let H denote the projection point (Figure 3). In triangle $\triangle\, A_0HA_1$ we have

$$x_{A_1} - x_{A_0} = |A_0A_1| \cos(\alpha + \beta)$$
$$y_{A_1} - y_{A_0} = |A_0A_1| \sin(\alpha + \beta).$$

Similarly, we can draw a horizontal line through A_1 and project A_2 on this line and get the following equations:

$$x_{A_2} - x_{A_1} = |A_1A_2| \cos(2\alpha + \beta)$$
$$y_{A_2} - y_{A_1} = |A_1A_2| \sin(2\alpha + \beta)$$

As it follows, the position of each point A_i, can be calculated from the following recurring relations:

$$x_{A_i} - x_{A_{(i-1)}} = |A_{(i-1)}A_i| \cos(i\alpha + \beta)$$
$$y_{A_i} - y_{A_{(i-1)}} = |A_{(i-1)}A_i| \sin(i\alpha + \beta)$$

Since $e = |A_{i-1}A_i|$, for all $1 \leqslant i \leqslant k+1$, we solve the above recurring relations as follows

$$x_{A_j} = e \sum_{i=1}^{j} \cos(i\alpha + \beta) + x_{A_0}, \tag{1}$$

$$y_{A_j} = e \sum_{i=1}^{j} \sin(i\alpha + \beta) + y_{A_0}. \tag{2}$$

where $j = 1, \ldots, k+1$.

In a simple polygon, the sum of the interior angles with d vertices is equal to $(d-2)\pi$. If we connect A_0 to A_{k+1}, we have a simple polygon. In this polygon, the interior angle at A_0 and A_{k+1} are equal because C is part of a regular polygon. If d is the number of vertices of the polygon and since k is the

number of turning points in C, we have $k = d-2$ and so the sum of the interior angles of the simple polygon is $(d-2)\pi = k\pi$. The interior angle of the first and the last vertices in the chain is θ and the other angles are $\pi - \alpha$. Therefore we have $k(\pi-\alpha)+2\theta = k\pi$, or equivalently $\theta = \frac{k\alpha}{2}$.

Let the angle between $\overline{A_0A_{k+1}}$ and the x-axis be denoted by γ. If we know the position of A_0 and A_{k+1}, we have the value of γ. Since $\gamma = \beta + \alpha + \theta$, we can rewrite this equation to compute β as $\beta = \gamma - (\alpha + \theta)$.

If we further know the number of turning points of C, i.e. k, the length of segments of C can be computed with each of the following equations

$$e = \frac{x_{A_{k+1}} - x_{A_0}}{\sum_{i=1}^{k+1} \cos(i\alpha+\beta)} = \frac{y_{A_{k+1}} - y_{A_0}}{\sum_{i=1}^{k+1} \sin(i\alpha+\beta)}.$$

Obtaining the value of e, the position of each vertex in the chain will be determined by Equations 1 and 2.

B. Ray shooting

One of the problems we encounter for path planning is to recognize if a segment intersects a set of segments. Formally, given a set of segments $S = \{s_1, s_2, \ldots, s_n\}$, we are asked if segment t intersect at least one of the segments of S.

It is obvious that this problem could be solved by checking the intersection of the segment t with all the segments of S in $\Theta(n)$ time. But, since the problem must be solved a lot of times for a fixed given set of segments, an algorithm with less query time is more suitable. We use ray shooting algorithm for this problem. In ray shooting problem, a set of segments S is known. We need to preprocess the set S such that given a query ray (its start point and direction), the first intersection position of the ray with the segments of S will be recognized quickly. We should also detect the case in which the ray does not intersect any segment of S.

By the solution of this problem; the problem of intersection of a segment with a given set of segments will be solved as follow: first, we preprocess S for solving the ray shooting problem. During the query time by receiving the segment t, we shoot a ray from one end-point of t in the direction of the other end-point. If the first intersection point between this ray and the segments of S is between the end-points of t, the answer to our solution is positive, in other words, t has intersection with the segments of S. On the other hand, if the intersection position is not between the end-points of t or there is not any intersection at all, the answer to our problem is negative.

Different algorithms are suggested for ray shooting problem. The last results belong to Chan [9] and Chen and Wang [11]. Chan [9] introduced a randomized algorithm which in$O(n\log^3 n)$ preprocessing time and in $O(n\log^2 n)$ space, the problem of ray shooting for an arbitrary query ray will be solved in $O(\sqrt{n}\log^2 n)$ average time. Chen and Wang [10] constructed a data structure of size $O(n+h^2)$ in $O(n + h^2\log^c n)$ time that answer ray shooting queries in $O(\log n)$ time. In the preprocessing time bound above, c is a constant.

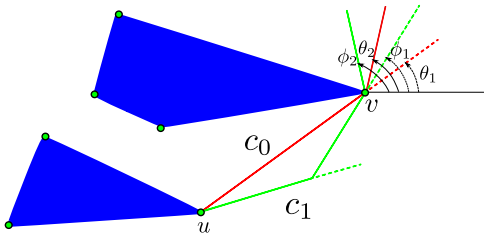

Fig. 4. Following the chain c_0, where the object moves directly towards v, it can leave v with an angle in the range $[\theta_1, \theta_2]$. For the chain c_1 the valid leaving angle is in the range $[\phi_1, \phi_2]$. We call this range, valid leave range of chain c, and denote it by VLR(c).

By combining the latest results about ray shooting and the technique used by Matouek [11] we concludes that for any arbitrary parameter b, where $n\log^2 n \leqslant b \leqslant n^2$ and a constant c, a data structure of size $O(b)$ can be constructed in $O(b\log^c n)$ preprocessing time such that the problem of ray shooting for an arbitrary query segment can be solved in $O((\frac{n}{\sqrt{b}})\log^c n)$ time.

V. Path Planning Algorithm

In this section, we describe our algorithm for finding a path with the given requirements. Since we prefer to make the path smooth, we try to spread out the break points along the path evenly. Therefore, we use regular chains of segments when moving from an obstacle vertex to another one.

Let u and v be two obstacle vertices. We want to find all paths starting from u to v not passing through other vertices of obstacles. To do this, we examine all regular chains of segments with different number of break points starting from u and ending at v. If all edges of a chain do not intersect any edges of the obstacles, then the desired chain is considered as a *valid* path from u to v. In other words, we first look at the regular chain of segments without any break point, that is the straight line segment from u to v. If the only edge of this chain does not intersect any obstacle then we consider this chain as a valid path from u to v. Similarly, we examine the regular chain of segments with one break point, two break points, and so on. We continue this process until the distance between two adjacent break point in the chain gets smaller than l. If the first and the last points of a regular chain are fixed, the distance between two consecutive break points decreases when the number of break points increases. Therefore there are a limited number of valid regular chains of segments from u to v.

Assume the object follows a given regular chain of segments from u to v when traveling from the start point to the target. Because of the "maximum turning angle property", when the object moves away from v, its *leaving angle* must be within a fixed range. This observation is depicted in Figure 4. Following the chain c_0, where the object moves directly towards v, it can leave v with an angle in the range $[\theta_1, \theta_2]$. Similarly, for the chain c_1 with a single break point, the valid leaving angle is in the range $[\phi_1, \phi_2]$. We call this range, valid leave range of chain c, and denote it by VLR(c).

In the following, we construct a graph $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$, such that the shortest path between two vertices in $\mathcal{G}$ corresponds to the shortest path from the starting point s to the target point t in the original configuration, consisting only of regular chains of segments. We use Dijkstra's algorithm to find the shortest path in $\mathcal{G}$. For an obstacle vertex v and valid leaving range $r = [\theta_1, \theta_2]$, we add a node v_r to $\mathcal{G}$. In addition, for each pair u_r and $v_{r'}$ of nodes in $\mathcal{G}$, if the following conditions hold, we add an edge from u_r to $v_{r'}$.

1) There exists a regular chain of segments c from u to v.
2) The chain c does not intersect any obstacle.
3) It leaves u with an angle in the range r.
4) $\text{VLR}(c) = r'$.

The weight of this edge is equal to the length of c.

Now let's explain how we construct the graph. We consider the starting point s and the target point t as obstacle vertices. We have a set of unprocessed graph nodes, denoted by $U_\mathcal{G}$, which initially contains only node $s_{[0,2\pi]}$, corresponding to vertex s with a full valid leaving range. During the construction of $\mathcal{G}$, we remove a node u_r from $U_\mathcal{G}$ and find all valid regular chains we could draw from vertex u to other obstacle vertices which leave u with an angle in the range r. For each valid chain c with this property from u to a vertex v, let $r' = \text{VLR}(c)$. We add a new node $v_{r'}$ to $U_\mathcal{G}$, which must be processed later and added to $\mathcal{G}$. We also add an edge to the set of edges of $\mathcal{G}$ from node u_r to node $v_{r'}$ corresponding to c. Finally, when u_r processed completely, we add it to $V_\mathcal{G}$, which is the set of vertices of $\mathcal{G}$. We repeat this process until there is no node in $U_\mathcal{G}$.

Lemma 1. *For a polygonal domain of n vertices, $|V_\mathcal{G}|$, that is the number of nodes of $\mathcal{G}$, is $O(\frac{2\pi}{\alpha}n^2)$ and $|E_\mathcal{G}|$, that is the number of edges is $O(|V_\mathcal{G}|^2)$.*

Proof. For each pair of vertices u and v, we have at most $\frac{2\pi}{\alpha} + 1$ valid regular chains of segments. This is because the maximum number of break points is $\frac{2\pi}{\alpha}$. For a vertex v and a valid chain c, a single node will be added to $\mathcal{G}$. Therefore, the number of nodes is at most equal to the number of different pairs of vertices times the number of valid regular chains from the first vertex to the second one, that is $O(\frac{2\pi}{\alpha}n^2)$.

The second claim is trivial as $\mathcal{G}$ is a simple graph. □

Lemma 2. *The shortest path in $\mathcal{G}$ from node $s_{[0,2\pi]}$ to all nodes t_r for some valid leaving range r is equal to the shortest path in the polygonal domain from vertex s to vertex t that consists of only regular chains of segments.*

Proof. We construct $\mathcal{G}$ such that any valid regular chain of segments from a vertex u to a vertex v have a corresponding edge in the graph with the same weight as the length of the chain. We can establish a one to one mapping between each path in $\mathcal{G}$ and each path in the polygonal domain which consists of only regular chain of segments. Therefore the shortest path in the plane has a corresponding path in $\mathcal{G}$, which is the shortest path in $\mathcal{G}$ from $s_{[0,2\pi]}$ to some node in $\mathcal{G}$ related to t. □

Corollary 1. *From the previous lemma, we conclude that running Dijkstra's algorithm on $\mathcal{G}$ from $s_{[0,2\pi]}$ until we reach some node related to vertex t, solves the path planning problem in polygonal domain.*

Theorem 1. *The algorithm for path planning with "maximum turning angle", "connectedness" and "minimum turning distance" properties from the starting point s to the target point t can be run in $O(n^4)$ time, where n is the total number of obstacle vertices.*

Proof. The running time of Dijkstra's algorithm on a graph with $|V|$ vertices and $|E|$ edges is $O(|E| + |V| \log |V|)$ using Fibonacci heap data structure. According to Lemma 1, the running time on $\mathcal{G}$ will be $O(\frac{4\pi^2}{\alpha^2}n^4)$. Since π and α are constants, the claim proved. □

VI. Extension: Problem in the Query Mode

In the original problem, it was assumed that the target point t is given at the same time as the other input. In this section we divide the algorithm into two parts, preprocessing phase, and query phase. In the preprocessing phase, the set of obstacles and the starting point are given and in the query phase the target point is determined. The problem in this mode is suitable when we want to find the best moving path to different position of the target in a fixed environment.

We solve the problem in the query mode as follows. Given the set of obstacles and the starting point in the preprocessing phase, we construct the graph $\mathcal{G}$ and run Dijkstra's algorithm on $\mathcal{G}$ to find the shortest path from $s_{[0,2\pi]}$ to all other nodes. In the query phase, given the target point t, we consider each node u_r in $\mathcal{G}$ and find the valid regular chain of segments from u to t with the smallest number of break points and leaving angle in the range r. If there exists such a chain, we add the length of the chain to the length of the shortest path to u_r and store it as a possible path from s to t. The final solution is the shortest path among such possible paths.

Theorem 2. *For the problem of path planning with "maximum turning angle", "connectedness" and "minimum turning distance" properties, we can preprocess a data structure in $O(n^4)$ time, using $O(n^4)$ space to answer the query for any target point t in $O(n^2)$ time. In these upper bounds n is the total number of obstacle vertices.*

Proof. We can construct the graph using $O(n^4)$ space and run Dijkstra's algorithm in $O(n^4)$ time as proved in Lemma 1 and Theorem 1, respectively. In the query time, we need to process each node of a graph in $O(1)$ time to find a possible path to t. Processing all nodes and finding the shortest path to t takes $O(|V_G|) = O(n^2)$ time. □

VII. Algorithm Evaluation

In this section we report our experiments to evaluate the algorithm in practice. We evaluate the algorithm in two cases. In the first case, we run the algorithm on two different polygonal domain. These configurations are depicted in Figure 5 and 6. The results can be seen in Table I. As it can be seen,

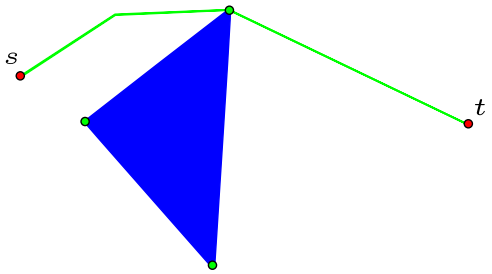

Fig. 5. The first configuration with $\alpha = \pi/6$ and $l = 50$.

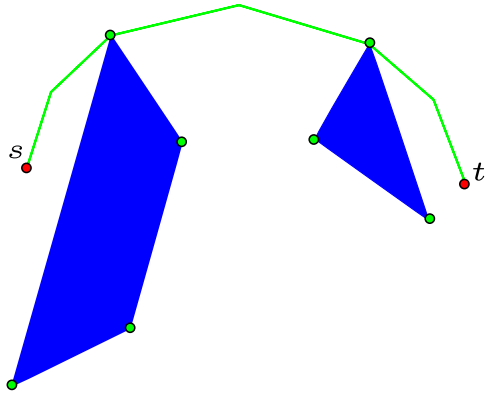

Fig. 6. The second configuration with $\alpha = \pi/6$ and $l = 50$.

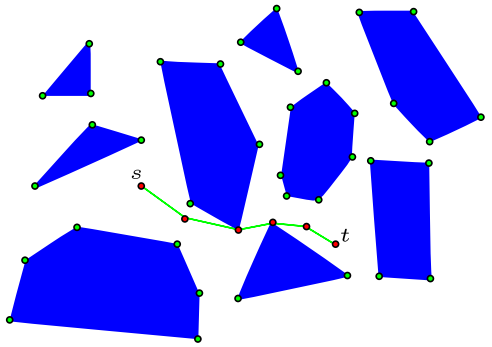

Fig. 7. A polygonal domain with 9 obstacles and 39 vertices is illustrated in which we find a path with $\alpha = \pi/18$ and $l = 60$.

the length of the path found by the algorithm, the length of the shortest path with the required properties and the relative error of the result are shown for each configuration.

In the second case, we use six different polygonal domains with various size, and run the algorithm to find the required preprocessing and query time of the algorithm for each one. In these test cases, the maximum turning angle $\alpha = \pi/9$ and the minimum turning distance $l = 60$. For each test case, the number of obstacles and vertices, and the preprocessing and query time in milliseconds are shown in Table II.

In Figure 7 a polygonal domain with 9 obstacles and 39 vertices is illustrated in which we find a path with $\alpha = \pi/18$ and $l = 60$.

VIII. Conclusion

In this paper we studied the problem of path planning for a robot with "maximum turning angle" and "minimum turning distance" in a polygonal domain. We proposed an algorithm to find a path with the given requirements. We prove that the required space and the running time of the algorithm are $O(n^4)$. We further decompose the algorithm into two parts, the preprocessing phase and the query phase, to reduce the running time of the algorithm when a set of target points are going to be used in a fixed polygonal domain.

Although our main goal was not to find the shortest path, as presented by the experiments, the solution is not much longer than the shortest path. For future work, we try to improve the algorithm and find the shortest path with the given requirements.

References

[1] S. Kapoor and S. Maheshwari, "Efficient algorithms for euclidean shortest path and visibility problems with polygonal obstacles," in *Proceedings of the fourth annual symposium on Computational geometry*. ACM, 1988, pp. 172–182.

[2] R. Inkulu, S. Kapoor, and S. Maheshwari, "A near optimal algorithm for finding euclidean shortest path in polygonal domain," *arXiv preprint arXiv:1011.6481*, 2010.

[3] S. Kapoor, S. N. Maheshwari, and J. S. Mitchell, "An efficient algorithm for euclidean shortest paths among polygonal obstacles in the plane," *Discrete & Computational Geometry*, vol. 18, no. 4, pp. 377–383, 1997.

[4] G. Liu, S.-y. Lao, L.-l. Hou, Y. Li, and D.-f. Tan, "Oarper-mafo algorithm for anti-ship missile path planning," *Aerospace Science and Technology*, vol. 47, pp. 135–145, 2015.

[5] R. Samar and W. A. Kamal, "Optimal path computation for autonomous aerial vehicles," *Cognitive Computation*, pp. 1–11, 2012.

[6] Q. Wang, L. Ma, and H. H. Deng, "Adaptive path planning of the uav based on genetic algorithm," *Jisuanji Xitong Yingyong- Computer Systems and Applications*, vol. 22, no. 1, pp. 200–203, 2013.

[7] H. Chen and Z. Xu, "Path planning based on a new genetic algorithm," in *Neural Networks and Brain, 2005. ICNN&B'05. International Conference on*, vol. 2. IEEE, 2005, pp. 788–792.

[8] J. T. SU and J. F. LI, "Path planning of anti-submarine cruise missile based on genetic algorithm," *Computer Knowledge and Technology*, vol. 16, p. 056, 2013.

[9] T. M. Chan, "Optimal partition trees," *Discrete & Computational Geometry*, vol. 47, no. 4, pp. 661–690, 2012.

[10] D. Z. Chen and H. Wang, "Visibility and ray shooting queries in polygonal domains," *Computational Geometry*, vol. 48, no. 2, pp. 31–41, 2015.

[11] J. Matoušek, "Range searching with efficient hierarchical cuttings," in *Proceedings of the eighth annual symposium on Computational geometry*. ACM, 1992, pp. 276–285.

TABLE I
COMPARING THE RESULT OF THE ALGORITHM WITH THE SHORTEST PATH IN TWO DIFFERENT CONFIGURATIONS.

Test case	#Obstacles	#Vertices	Shortest path	Path length	Relative error
1	1	3	652.30	662.14	1.51%
2	2	7	693.51	712.31	2.71%

TABLE II
PREPROCESSING AND QUERY TIME OF THE ALGORITHM FOR POLYGONAL DOMAINS WITH VARIOUS SIZES.

Test case	#Obstacles	#Vertices	Preprocessing time (ms)	Query time (ms)
1	1	3	2	3
2	2	7	25	3
3	9	39	1166	18
4	26	100	2033	26
5	84	300	28860	188
6	180	500	223134	570